



AECENAR

Association for Economical and Technological Cooperation
in the Euro-Asian and North-African Region

www.aecenar.com

Karlsruhe/Heidelberg, Germany



Karlsruhe/Heidelberg, Germany



Final Report for TEMOLeb-Mintad/VaEf-Airship (1999 - 2020)

1. Development of a flight control system for an airship (System Development and Hardware Development) (1999-2006)
2. Development of a communication system (2013) / Communication System with HarckRF (2020)
3. Aerodynamic Investigations for a High-Altitude Airship (Platform) (2017)
4. Building of a small prototype of an airship (2017)
5. Development of a GCS (Ground Control System) (Software Development and System Integration) (2018)
6. Development of a FCS (Flight Control System) (Software Development and System Integration) (2018/2019)
7. Marketing Concept (2017-2019)

Last Update: 31.07.2020 02:42

Project Manager: Dr. Samir Mourad

Based on following research reports:

- Mourad S., Forstmann S., Subhan M., E.J., Farkh R., Abdelhaq N., Faquir M., Mikou M.Y., Ammar A., Flight Control System for an Wind Data Measuring Airship (Project Report for 2000 - 2006), Karlsruhe, 2006
- Zohby, Mourad; *Emergency Communication System for IAP-SAT*, Jan 2014
- Mourad S, Bakri S, Shaker A, Mourad A, *TEMO-Leb Airship Report 2017*
- MMJZ, *TEMO-Leb Airship FCS*, 2018
- MMJZ, *TEMO-Leb Airship GCS*, 2018/9

@AECENAR 2020 - Heidelberg/Germany

Content in Short

PROJECT MANAGEMENT	1
1 SYSTEM REQUIREMENTS ANALYSIS (تحليل متطلبات المنظومة)	2
2 POSTERS.....	5
DEVELOPMENT OF A FLIGHT CONTROL SYSTEM FOR AN AIRSHIP (1999-2006)	11
ABSTRACT.....	26
3 INTRODUCTION	28
4 PROJECT MANAGEMENT	29
5 FLIGHT CONTROL SYSTEM OF AN AIRSHIP	30
6 OPTIMIZATION OF THE FLIGHT CONTROL SYSTEM ARCHITECTURE.....	62
7 DEVELOPMENT OF THE EREIGNISDIENSTES FOR THE MIDDLEWARE OSA+	79
8 INERTIAL MEASUREMENT UNIT.....	128
LITERATURE	171
9 ACTUATOR BOARD	173
LITERATURE	236
10 COMMUNICATION AND USER INTERFACE	249
FEATURES	296
OR THE BOARD USED TO EQUIPS THE STD-402 IS THE MB-STD-RS232, SO THE THIS MODE (AUTO MODE OPERATION GUIDE FOR CPU INTERFACE) IS USED LIKE MODE OF THE STD-402 TRANSCEIVER.	322
11 SOME REPAIRS ON THE SENSOR CARD AND FIRST STEP INTEGRATION	376
12 INTEGRATION.....	410
13 INTEGRATION CODE IN C.....	489
DEVELOPMENT OF A COMMUNICATION SYSTEM (2013)	497
ABBREVIATIONS	501
14 ABSTRACT	503
15 PROJECT MANAGEMENT	505
16 BASICS.....	507
FIGURE 6.6: STD-402 TRANSCEIVER – <i>CIRCUIT DESIGN</i>	546
OR THE MB-STD-RS232 EQUIPS STD-402 TRANSCEIVER MODULE AND PERFORMS PACKET COMMUNICATION USING CPU INTERFACE MODE OF THE TRANSCEIVER.	547
17 SPECIFICATION	551
18 SYSTEM DESIGN	553
19 MECHANICS.....	557
20 SCS-SMS	559
21 AES ENCRYPTION.....	583
22 HARDWARE OF ECS DEMO SYSTEM.....	599

23	FURTHER WORK: SYSTEM INTEGRATION AND INTEGRATION TEST OF ECS DEMO SYSTEM.....	611
	APPENDIX A: ALTERNATIVE PROJECT PLANS.....	612
	APPENDIX B: ALL ABOUT HACKRF.....	616
	APPENDIX C: ALTERNATIVE SYSTEM DESIGNS.....	624
	LITERATURE.....	625
	COMMUNICATION SYSTEM WITH HARCKRF (FROM IAP-SAT, 6TH PROJECT REPORT) (2020).....	626
24	TELEMETRY SYSTEM WITH HARCKRF.....	627
	AERODYNAMIC INVESTIGATIONS FOR A HIGH-ALTITUDE AIRSHIP (PLATFORM) (2017).....	644
25	BASICS.....	645
26	AERODYNAMIC INVESTIGATION OF A HIGH ALTITUDE AIRSHIP.....	667
27	REFERENCES.....	692
	APPENDIX A: CONTACT DATA OF SPECIALISTS (معلم), WORKERS,.....	694
	APPENDIX B: FOR AERODYNAMIC INVESTIGATION.....	695
	BUILDING OF THE FRAME INCLUDING ACTUATORS FOR A SMALL PROTOTYPE OF AN AIRSHIP (2017).....	697
28	BASICS CONCERNING ACTUATORS USED IN AEROSPACE.....	698
29	ACTUATOR SYSTEM OF TEMOLEB-MINTAD.....	716
30	PROTOTYPE CONSTRUCTION.....	735
	APPENDIX C: FOR ACTUATOR SYSTEM.....	741
	APPENDIX D: FOR PROTOTYPE CONSTRUCTION.....	743
	DEVELOPMENT OF A GCS (GROUND CONTROL SYSTEM) (2018).....	745
	CONTENTS.....	746
31	APP STRUCTURE:.....	747
32	WIDGET:.....	750
33	CONFIGURATION PAGE:.....	753
34	CONTROLLERS.....	758
35	USER CONTROL CODE SUMMARY:.....	759
36	REFERENCES:.....	764
	DEVELOPMENT OF A FCS (FLIGHT CONTROL SYSTEM) (2018).....	765
37	SYSTEM ARCHITECTURE.....	768
38	SERVO ACTUATOR SYSTEM.....	769
39	REALIZATION OF INERTIAL MEASUREMENT UNIT (IMU).....	778
40	WIFI COMMUNICATION WITH NRF24L01.....	788
41	SOFTWARE DEVELOPMENT ON ARDUINO SIDE WITH THE ARDUINO INTEGRATED DEVELOPMENT ENVIRONMENT (IDE).....	792
42	ASSEMBLY.....	808

43 FINAL IMPLEMENTATION816
44 MINTAD PROTOTYPE TEST RIG QUICK GUIDE817
MARKETING CONCEPT (2017-2019)818
LITERATURE820

Content

PROJECT MANAGEMENT	1
1 SYSTEM REQUIREMENTS ANALYSIS (تحليل متطلبات المنظومة)	2
2 نشر اولي للمشروع في 18.2.2017	1.1
3 حساب التكلفة (Calculation)	1.1.1
3 لقاء مع الاستاذ باسل الايوبي، مدير OGERO في 19.2.2017 في مسجد القبيسي ...	1.1.2
3 متطلبات المنظومة (SYSTEM REQUIREMENTS)	1.2
4 تصميم وتصنيع المنطاد (TEMOLEB-MINTAD/MECHANICS)	1.3
4 ايلكترونيك في المنطاد (TEMOLEB-MINTAD/ELECTRONICS)	1.4
2 POSTERS.....	5
DEVELOPMENT OF A FLIGHT CONTROL SYSTEM FOR AN AIRSHIP (1999-2006)	11
ABSTRACT.....	26
3 INTRODUCTION	28
3.1 THE LOTTE PROJECT AND THE "ALTERNATIVE LOTTE"	28
4 PROJECT MANAGEMENT	29
4.1 COSTS.....	29
5 FLIGHT CONTROL SYSTEM OF AN AIRSHIP	30
5.1 BLOCK DIAGRAM OF THE CONTROL LOOPS.....	61
6 OPTIMIZATION OF THE FLIGHT CONTROL SYSTEM ARCHITECTURE.....	62
6.1 THE ARCHITECTURE OF THE FLIGHT CONTROL SYSTEM AND ITS SIMULATION ON THE MIDDLEWARE OSA+ RUNNING ON THE OPERATING SYSTEM WINDOWS	62
6.2 RESULTS	63
6.2.1.1 System Architecture of Airship Flight Control System	71
6.2.1.2 Simulation Code (in programming language C)	71
6.2.1.3 Simulation results	71
7 DEVELOPMENT OF THE EREIGNISDIENSTES FOR THE MIDDLEWARE OSA+	79
7.1 ABRIDGED VERSION	82
7.2 THE NATURE OF THE TASK	82
7.3 THE OSA+ ARCHITECTURE	82
7.3.1 Introduction	83
7.3.2 The components of OSA+	84
7.3.2.1 The Platform	84
7.3.2.2 Generic Services.....	85
7.4 COMPONENTS OF THE OSA EVENTING SERVICE	86
7.4.1 The time functions	86
7.4.1.1 The handling of the time.....	86
7.4.1.2 Initialization	87
7.4.1.3 Functions for time management	89
7.4.2 The event service.....	90
7.4.2.1 General Information	90
7.4.2.2 The initialization of the Event Service.....	92
7.4.2.3 Features of the Ereignisdienstes.....	92

7.5	SHORT OVERVIEW	94
7.5.1	<i>Time functions/variables</i>	94
7.5.2	<i>Features of the Event Service (Ereignisdienst)</i>	94
7.6	CONFIGURATION	94
7.7	THEORY OF OPERATION	96
7.7.1	<i>Osagettime()</i>	96
7.7.2	<i>Osasettime(int,int)</i>	96
7.7.3	<i>Osaaddtime(int,int)</i>	98
7.7.4	<i>Char * osaPrintTime(int, int)</i>	98
7.7.5	<i>Osainitevents()</i>	99
7.7.6	<i>Int osaAddEvent(uint,uint, uint, uint,uint, uint,uint, uint, function)</i>	99
7.7.7	<i>Int osaDelEvent(uint,uint,uint,UINT)</i>	101
7.7.8	<i>OSA_Error osaGetEventResult(uint,UINT)</i>	101
7.7.9	<i>Internal functions of the Ereignisdienstes</i>	102
7.8	PROGRAMMING EXAMPLES	102
7.8.1	<i>For example: Changing the Time</i>	102
7.8.2	<i>Example: Add an Event</i>	103
7.8.2.1	Start a function at a specified time.....	103
7.8.2.2	Repeated start a function	105
7.8.3	<i>Queries of results</i>	106
7.8.4	<i>Delete a Events</i>	106
7.9	TEST RUNS.....	107
7.9.1	<i>Deliver-Test</i>	109
7.9.2	<i>Test run of a cyclical events with open</i>	110
7.9.3	<i>Test the maximum temporal resolution on different systems</i>	113
7.9.3.1	System 1	113
7.9.3.2	System 2	116
7.9.3.3	System 3	118
7.9.4	<i>Stability tests</i>	121
7.9.4.1	Exceeding the maximum acceptable number of Events	121
7.9.4.2	Exceeding the maximum acceptable number of events per unit time	122
7.9.4.3	Overrun of the events per unit time through Zeituberschneidung	123
7.10	RESULTS AND OUTLOOK	126
7.11	ANNEX	127
7.11.1	<i>List of all Windows specific functions</i>	127
7.12	LITERATURE.....	127
8	INERTIAL MEASUREMENT UNIT.....	128
8.1	INTRODUCTION	133
8.1.1	<i>Task</i>	133
8.1.2	<i>Overview</i>	134
8.2	BASICS	135
8.2.1	<i>Coordinate System</i>	136
8.2.2	<i>Calculation of the current position</i>	137
8.2.3	<i>Compass</i>	138
8.3	ARCHITECTURE DESIGN OF THE IMU	139
8.3.1	<i>Measurement Data Collection (Meßdatenaufnahme)</i>	139
8.3.2	<i>Measurement Data Processing (Meßdatenaufbereitung)</i>	139
8.4	DEVELOPMENT ENVIRONMENT.....	141
8.4.1	<i>Hardware-Entwicklungsumgebung</i>	141

8.4.2	<i>Software Development Environment</i>	142
8.5	REALIZATION OF THE IMU	143
8.5.1	<i>Meßdatenaufnahme</i>	143
8.5.1.1	Circuit Design for the Meßdatenaufnahme	143
8.5.1.2	Reference Voltage	150
8.5.2	<i>Board layout design for the Meßdatenaufnahme (Measurement data collection)</i>	151
8.5.3	<i>Meßdatenaufbereitung</i>	153
8.5.3.1	Circuit Design for the Meßdatenaufbereitung	153
8.5.4	<i>Board layout design for the Meßdatenaufbereitung</i>	155
8.5.4.1	The software for the Meßdatenaufbereitung	157
8.5.4.2	User Interface	161
8.6	TEST RESULTS	165
8.6.1	<i>Hardware-Test</i>	165
8.7	TEST OF THE OVERALL SYSTEM WITH THE MICROCONTROLLER (C167)	166
8.7.1	<i>Gesamtsystem-Test 1</i>	166
8.7.2	<i>Gesamtsystem-Test 2</i>	168
8.7.3	<i>Gesamtsystem-Test 3</i>	168
8.8	SUMMARY AND OUTLOOK	170
LITERATURE		171
9	ACTUATOR BOARD	173
9.1	INTRODUCTION	181
9.1.1	<i>The Solarluftschiff Lotte</i>	181
9.1.2	<i>Task</i>	182
9.1.3	<i>Overview</i>	182
9.2	BASICS	183
9.2.1	<i>Real-time systems</i>	183
9.2.2	<i>Hardware</i>	183
9.2.2.1	Components and classifications	183
9.2.2.2	The microcontroller family C166	184
9.2.2.3	The C167 microcontroller family	185
9.2.2.4	The memory organization of the C167)	186
9.2.2.5	The interrupt system of the C167)	188
9.2.2.6	The Timereinheiten	190
9.2.2.7	Capture Compare Unit (Capcom)	191
9.2.2.8	The PWM Pulsweiten-Einheit	193
9.2.2.9	Analog Digital Converter (ADC)	194
9.2.3	<i>Control of servo motors</i>	195
9.3	ARCHITECTURE DESIGN OF THE AKTORIK-ANSTEUERUNGSEINHEIT (ENGL, <i>ACTUATOR CONTROL UNIT (ACU)</i>)	197
9.3.1	<i>Voltage regulation</i>	197
9.3.2	<i>Betriebsspannungsüberwachung</i>	198
9.3.3	<i>Basisbetriebszustands-Einstellung</i>	198
9.3.4	<i>Steuersignal-Erzeugung</i>	198
9.3.5	<i>Notsteuerungsbaugruppe</i>	199
9.3.6	<i>Operating conditions of the ACU</i>	200
9.3.7	<i>Modularity of the ACU</i>	200
9.4	DEVELOPMENT ENVIRONMENT	202
9.4.1	<i>Hardware-Entwicklungsumgebung</i>	202
9.4.2	<i>Software Development Environment</i>	203
9.5	REALIZATION OF THE ACU	204

9.5.1	<i>Versorgungsspannungs-Stabilisierung</i>	204
9.5.1.1	Circuit Design for the Versorgungsspannungs-Stabilisierung	205
9.5.2	<i>Betriebsspannungsüberwachung</i>	207
9.5.3	<i>Basisbetriebszustands-Einstellung</i>	208
9.5.3.1	Circuit Design for the Basisbetriebszustands-Einstellung	211
9.5.4	<i>Steuersignal-Erzeugung</i>	212
9.5.4.1	The hardware of the Steuersignal-Erzeugung.....	214
9.5.4.2	The monitoring and the forwarding of the PWM-signals from the remote control (normal operation (B)).....	214
9.5.4.3	Level 1 Notsteuersignal-Erzeugung	219
9.5.5	<i>Notsteuerungsbaugruppe</i>	222
9.5.5.1	Circuit Design for the monitoring of the control signals on the output of the ACU and the channel5-output of the Fernsteuerempfangers	222
9.5.5.2	Circuit Design for the Notsteuerungsbaugruppe	223
9.5.5.3	Circuit Design for the forwarding of the PWM-signals from Fernsteuerungsempfanger (Luftschiff-Startphasen-operation)	224
9.5.6	<i>Layout design of board 1 ("Voltage regulation", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe ")</i>	224
9.5.7	<i>The circuit boards of the ACU</i>	225
9.6	EXPERIMENTAL RESULTS	227
9.6.1	<i>Structure of the Testplatzes</i>	227
9.6.2	<i>Experiments and test sequence</i>	228
9.6.3	<i>The series of tests</i>	228
9.6.3.1	Test 1	229
9.6.3.2	Test 2	230
9.6.3.3	TEST3	231
9.6.3.4	Test 4	232
9.6.3.5	Test 5	233
9.7	SUMMARY AND OUTLOOK	235
LITERATURE		236
9.8	APPENDIX A: CIRCUIT DESIGN FOR THE STABILIZATION OF POWER SUPPLY VOLTAGE	237
9.9	ANNEX B: TEST PROGRAM 1.....	240
10	COMMUNICATION AND USER INTERFACE	249
10.1	INTRODUCTION	249
10.1.1	<i>The LOTTE project and the "alternative Lotte"</i>	249
10.1.2	<i>Development method</i>	250
10.1.3	<i>Working task and realization approach</i>	250
10.1.4	<i>Overview</i>	250
10.2	BASICS	251
10.2.1	<i>The V-Model</i>	251
10.2.2	<i>Structured Analysis (SA) / Structured Design (SD)</i>	252
10.2.3	<i>Transceivers</i>	254
10.3	REQUIREMENTS SPECIFICATION	254
10.3.1	<i>The Graphical User Interface (GUI)</i>	254
10.3.1.1	Software Requirements Specification.....	255
10.3.2	<i>Specifications for the transceiver</i>	259
10.3.3	<i>Communication Protocol for User Data</i>	259
10.3.4	<i>Handshaking</i>	261
10.4	ARCHITECTURE DESIGN	261
10.5	DEVELOPMENT ENVIRONMENTS	263

10.6	DESIGN AND IMPLEMENTATION.....	263
10.6.1	<i>SA / SD and Implementation for the base station program</i>	263
10.6.1.1	SA (Structured Analysis).....	263
10.6.1.2	SD (Structured Design).....	267
10.6.1.3	Implementation	270
10.6.2	<i>Communication part on the board system on the airship</i>	273
10.6.2.1	The transceiver	273
10.6.2.2	The communication software on the embedded board computer	278
10.7	COMPONENT TESTS	279
10.7.1	<i>Transceiver test</i>	279
10.7.2	<i>User interface laboratory test with two PCs</i>	282
10.7.3	<i>Test of the user interface on the target system</i>	284
10.8	ANNEX A.....	285
10.9	ANNEX B	295
10.9.1	<i>Block diagram of the STD-402 transceiver in Direct Mode</i>	309
10.10	ANNEX C.....	310
	◆ STD-402TR [Auto Mode Operation Guide]	310
10.10.1	<i>General</i>	310
10.10.2	<i>Features</i>	311
10.10.3	<i>Application Examples</i>	311
10.10.4	<i>Configuration</i>	311
10.11	REGISTRATION OF ID	315
10.11.1	<i>diagram of the STD-402 transceiver in Auto Mode</i>	321
10.12	ANNEX D	322
10.13	COMMUNICATION PROTOCOL	326
10.13.1	<i>STD-402 Data Format</i>	326
10.13.2	<i>Transmitter : Initial setting flow chart</i>	328
10.13.2.1	Receiver : Initial setting flow chart	329
10.13.3	<i>Receiver: Flow chart at power ON</i>	334
10.14	ANNEX E: VISUAL BASIC CODE FOR THE USER INTERFACE PROGRAM	341
10.15	ANNEX F: C PROGRAM CODE.....	368
10.16	ANNEX G	372
11	SOME REPAIRS ON THE SENSOR CARD AND FIRST STEP INTEGRATION	376
12	INTEGRATION	410
12.1	ABSTRACT	410
12.2	INTRODUCTION	414
12.2.1	<i>The Lotte-Projekt and the "alternative Lotte"</i>	414
12.3	DEVELOPMENT ENVIRONMENT.....	416
12.4	ARCHITECTURE DESIGN AND IMPLEMENTATION.....	417
12.5	LITERATURE.....	487
13	INTEGRATION CODE IN C	489
	DEVELOPMENT OF A COMMUNICATION SYSTEM (2013)	497
	ABBREVIATIONS	501
14	ABSTRACT	503
15	PROJECT MANAGEMENT	505
15.1	PROJECT DEFINITION HISTORY	505

Content

15.2	SYSTEM BUDGET (TIME AND COST) FOR DEMO SYSTEM	505
15.3	AT 21 JAN STILL OPEN TASKS FOR IAP ECS DEMO SYSTEM WHEN USING (ONLY INTEGRATION)	505
16	BASICS.....	507
16.1	COMMUNICATION BASICS.....	507
16.1.1	<i>Transmitter design from http://en.wikibooks.org/wiki/Electronics/Transmitter_design</i>	<i>519</i>
16.1.1.1	Frequency synthesis and frequency multiplication	520
16.1.1.2	Frequency mixing and Modulation	521
16.1.1.3	RF power amplifiers	527
16.1.1.4	Linking the transmitter to the aerial.....	527
16.1.1.5	EMC matters	528
16.2	RECEIVER DESIGN FROM HTTP://EN.WIKIPEDIA.ORG/WIKI/TUNER_(ELECTRONICS)	533
16.3	ANTENNA	534
16.4	SOFTWARE DEFINED RADIO (SDR)	536
16.5	HSDR (HIGH DEFINITION SOFTWARE DEFINED RADIO).....	536
16.6	EXTIO.DLL	537
16.7	HOW DO I DEVELOP AN EXTIO.DLL ?	537
16.8	VISUAL C++ 2008 EXPRESS	537
16.9	QT	537
16.10	RF HARDWARE (USB STICK)	537
16.10.1	<i>TERRATEC ran T stick DVB-T/DAB/DAB + Stick USB 2.0</i>	<i>537</i>
16.10.2	<i>Hackrf (an-open-source-SDR-platform)</i>	<i>538</i>
16.11	RF OVERVIEW.....	538
16.12	RF FREQUENCIES POLICIES	539
16.13	RF MODULES	543
16.13.1	<i>STD-402</i>	<i>543</i>
16.13.1.1	Special for MB-STD-RS232	543
16.13.1.2	Special for STD-402 (Transceiver)	545
16.13.2	<i>RFM42B-RFM31B 433MHz</i>	<i>547</i>
16.13.3	<i>BOWITZ W.T.</i>	<i>549</i>
16.13.4	<i>Comparison between modules</i>	<i>549</i>
17	SPECIFICATION	551
17.1	SYSTEM REQUIREMENTS	551
17.2	HARDWARE REQUIREMENTS.....	551
17.3	SOFTWARE REQUIREMENTS.....	551
18	SYSTEM DESIGN	553
18.1	SYSTEM OVERVIEW.....	553
18.2	CENTRAL STATION	553
18.2.1	<i>Architecture</i>	<i>553</i>
18.2.2	<i>SDR development side</i>	<i>553</i>
18.2.3	<i>Graphical User Interface.....</i>	<i>554</i>
18.3	MOBILE STATIONS.....	555
19	MECHANICS.....	557
19.1	MECHANICAL DESIGN	557
19.2	PROTOTYPE WITHOUT COVER.....	557
20	SCS-SMS	559
20.1	ABSTRACT OF SCS-SMS.....	559

20.2	SYSTEM DESIGN	559
20.3	ARCHITECTURES	560
20.4	PIC SOFTWARE	566
20.5	TEST	579
21	AES ENCRYPTION.....	583
21.1	INTRODUCTION	583
21.2	AES ALGORITHM	583
21.3	CODING	586
22	HARDWARE OF ECS DEMO SYSTEM.....	599
22.1	REALIZATION OF RF MODULE	599
22.1.1	<i>Using STD-402</i>	<i>599</i>
22.1.2	<i>Realization of RF Module Using RFM42B-RFM31B – 433MHz</i>	<i>603</i>
22.1.2.1	Serial Peripheral interface (SPI)	603
22.1.2.2	The new hardware design.....	603
22.1.2.3	MSSP module to establishing (SPI)	605
23	FURTHER WORK: SYSTEM INTEGRATION AND INTEGRATION TEST OF ECS DEMO SYSTEM.....	611
	APPENDIX A: ALTERNATIVE PROJECT PLANS.....	612
	APPENDIX B: ALL ABOUT HACKRF	616
	<i>B.1 HackRF overview</i>	<i>616</i>
	<i>B.2 Jawbreaker.....</i>	<i>618</i>
	<i>B.3 Jellybean</i>	<i>618</i>
	<i>B.4 Lemondrop.....</i>	<i>619</i>
	APPENDIX C: ALTERNATIVE SYSTEM DESIGNS.....	624
	LITERATURE	625
	COMMUNICATION SYSTEM WITH HARCKRF (FROM IAP-SAT, 6TH PROJECT REPORT) (2020).....	626
24	TELEMETRY SYSTEM WITH HARCKRF	627
24.1	TIME PLAN.....	627
24.2	INTRODUCTION	628
24.2.1	<i>Links and references:</i>	<i>630</i>
24.2.2	<i>SDR (Software defined radio):</i>	<i>631</i>
24.2.3	<i>HSDR:</i>	<i>631</i>
24.2.4	<i>SDRSharp:</i>	<i>632</i>
24.2.5	<i>GnuRadio:</i>	<i>634</i>
24.2.5.1	<i>Links:</i>	<i>635</i>
24.3	GETTING STARTED	635
24.3.1	<i>Getting Started with HackRF and GNU Radio.....</i>	<i>637</i>
24.3.2	<i>HackRF with Raspberry PI:.....</i>	<i>638</i>
24.4	SYSTEM 1	641
24.5	SYSTEM 2	643
24.6	SYSTEM 3	643
	AERODYNAMIC INVESTIGATIONS FOR A HIGH-ALTITUDE AIRSHIP (PLATFORM) (2017).....	644
25	BASICS.....	645

Content

25.1	HISTORY OF AIRSHIPS	645
25.2	EVOLUTION OF AIRSHIP.....	653
25.2.1	<i>Hot air balloons</i>	653
25.2.2	<i>Dirigible or Airship</i>	656
25.3	EVOLUTION OF THE AIRSHIP INDUSTRY	658
25.4	THE END OF AIRSHIP.....	658
25.5	RETURN OF AIRSHIP	659
25.6	ARCHIMEDES PRINCIPLE	660
25.7	TYPES OF AIRSHIPS.....	660
25.7.1	<i>Non-rigid Airship</i>	660
25.7.2	<i>Semi-rigid airship</i>	661
25.7.3	<i>Rigid Airship</i>	662
25.8	HOW DO RIGID AIRSHIP WORK?.....	663
25.9	WHY DO WE USE HELIUM NOT HYDROGEN?.....	664
25.10	CONTROL OF AIRSHIPS.....	664
26	AERODYNAMIC INVESTIGATION OF A HIGH ALTITUDE AIRSHIP.....	667
26.1	AEROSTATIC OF AIRSHIP.....	667
26.1.1	<i>Bouncy force</i>	667
26.1.2	<i>The relation between altitude and density</i>	668
26.1.3	<i>The relation between altitude and pressure</i>	668
26.1.4	<i>The relation between altitude and temperature</i>	669
26.1.5	<i>Pressure Altitude</i>	670
26.1.6	<i>Influence of pressure, temperature, density and volume on the airship during its rise</i>	670
26.1.7	<i>Sizing the airship hull</i>	671
26.2	TEMO-LEB AIRSHIP	672
26.2.1	<i>Design of "TEMO-Leb Airship"</i>	674
➤	Propeller.....	675
➤	Rudder and Elevator	675
➤	Frame of the airship	676
➤	Outer shell of the airship	677
➤	Airship with the internal gasbags	678
➤	Airship with balloons	679
26.2.2	<i>Mass of framework</i>	680
26.2.3	<i>Mass of gasbags</i>	682
26.2.4	<i>Mass of external fabric</i>	683
26.3	LOW ALTITUDE TEST TEMO-LEB AIRSHIP.....	684
26.3.1	<i>New design with new dimensions of the Low altitude test TEMO-Leb Airship</i>	684
26.3.2	<i>Materials of the test device for the Low altitude test TEMO-Leb Airship in Lebanese market</i>	685
1)	Balloons filled with Helium.....	685
2)	Plexiglas.....	686
3)	External envelope.....	686
26.4	RESULTS OF THEORETICAL STUDY	687
26.4.1	<i>Results of "MATLAB"</i>	687
1)	Variation of the density with altitude.....	687
2)	Variation of pressure with altitude.....	687
3)	Variation of temperature with altitude	689
4.1.2.	<i>$V_{TEMOLeb\ Airship}$ and the V_{max}</i>	689
4.1.3.	<i>Positive results</i>	690
27	REFERENCES	692

27.1	REFERENCES FOR CHAPTERS 1.....	692
27.2	REFERENCES FOR SECTIONS 2.1, 2.10-2.15 AND CHAPTER 4.....	692
27.3	REFERENCES FOR SECTIONS 2.2-2.9 AND CHAPTER 3.....	693
APPENDIX A: CONTACT DATA OF SPECIALISTS (معلم), WORKERS,		694
APPENDIX B: FOR AERODYNAMIC INVESTIGATION.....		695
	INITIAL WORKING PACKAGES	695
	ESTIMATION FOR COSTS OF LOGGING SENSORS DURING FLIGHT.....	696
BUILDING OF THE FRAME INCLUDING ACTUATORS FOR A SMALL PROTOTYPE OF AN AIRSHIP (2017).....		697
28	BASICS CONCERNING ACTUATORS USED IN AEROSPACE.....	698
28.1	WHAT'S AN ACTUATOR?.....	698
28.2	TYPES OF ACTUATORS.....	699
28.2.1	<i>Air Motors</i>	699
28.2.2	<i>Clutch/Brake:</i>	700
28.2.3	<i>Stepping Motors:</i>	700
28.2.4	<i>AC Induction Motors</i>	701
28.2.5	<i>Hydraulic motors:</i>	702
28.2.6	<i>Servomotors</i>	703
28.3	TECHNOLOGY COMPARISONS	704
28.4	FIRST ACTUATOR CHOSEN: HYDRAULIC ACTUATOR.....	705
28.4.1	<i>Applications of Hydraulic Motors</i>	705
28.4.2	<i>Types of hydraulic motors</i>	705
28.4.2.1	Gear motor	706
28.4.2.2	Vane Motor.....	707
28.4.2.3	Piston type motor.....	708
28.4.3	<i>Details about axial piston hydraulic motor:</i>	709
28.4.4	<i>Motor Installation Considerations:</i>	709
28.5	SECOND ACTUATOR CHOSEN: SERVO ACTUATOR	711
28.5.1	<i>Reason for choose of servo actuator:</i>	711
28.5.2	<i>General information about servo:</i>	711
28.5.3	<i>Types of servo motors</i>	713
29	ACTUATOR SYSTEM OF TEMOLEB-MINTAD	716
29.1	HYDRAULIC ACTUATOR SYSTEM	716
29.1.1	<i>Work principal for hydraulic motor</i>	716
29.1.2	<i>FreeCAD design proposal</i>	717
29.2	SERVO ACTUATOR SYSTEM.....	719
29.2.1	<i>Adopted Motor (Figure 28)</i>	719
29.2.2	<i>Basic parts of the servo (Figure 29)</i>	719
29.2.3	<i>Servo motor block diagram (Figure 30)</i>	720
29.2.3.1	Control computer:	720
29.2.3.2	Electronic control system:	720
29.2.3.3	Motor:.....	720
29.2.3.4	Gear train:.....	720
29.2.3.5	Position sensor:	721
29.2.3.6	Servo output:.....	721
29.2.4	<i>Design of servo actuator system (FreeCAD)</i>	721
29.2.5	<i>Motor Controller and Interfaces</i>	721
29.2.5.1	Mobile App:	722

Content

29.2.5.2	Bluetooth Module: (BLE Link -A Bluetooth 4.0 module for Arduino).....	722
29.2.6	Power Management: Polymer Lithium Ion Battery - 1000mah 7.4v	726
29.3	SOFTWARE DEVELOPMENT ON ARDUINO SIDE WITH THE ARDUINO INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)	727
29.3.1	Architecture of the program on the Arduino	727
29.3.2	Bluetooth_Blimp_Control: Code and explication.....	728
29.4	FINAL ASSEMBLY	731
29.5	TESTING THE INTEGRATED ACTUATOR	733
30	PROTOTYPE CONSTRUCTION.....	735
	APPENDIX C: FOR ACTUATOR SYSTEM	741
	INITIAL WORKING PACKAGES	741
	QUOTATION FOR ARDUINO BASED ACTUATOR SYSTEM	742
	APPENDIX D: FOR PROTOTYPE CONSTRUCTION	743
	INITIAL WORKING PACKAGES.....	743
	DEVELOPMENT OF A GCS (GROUND CONTROL SYSTEM) (2018).....	745
31	APP STRUCTURE:	747
31.1	HEADLINE WIDGET CONTAIN:	747
31.2	MAIN AREA:	748
31.3	LEFT NAVIGATION PANEL CONTAIN:.....	748
31.4	RIGHT NAVIGATION PANEL CONTAIN:.....	749
31.5	FOOTER WIDGET CONTAIN:	749
32	WIDGET:.....	750
32.1	HOME WIDGET:.....	750
32.2	ASI WIDGET:	750
32.3	MAP WIDGET:.....	751
32.4	CAMERA WIDGET:.....	751
32.5	CONTROLLER WIDGET:.....	752
33	CONFIGURATION PAGE:	753
33.1	GENERAL INFO CONFIGURATION:	753
33.2	CONNECTION CONFIGURATION:	754
33.3	KNOBS CONFIGURATION:.....	754
33.4	PFD CONFIGURATION:.....	755
33.5	MAP CONFIGURATION:	755
33.6	ROUTE CONFIGURATION:.....	756
33.7	TASK CONFIGURATION:	757
34	CONTROLLERS	758
34.1	SIGNAL STRENGTH.....	758
34.2	KNOBS CONTROLLER	758
35	USER CONTROL CODE SUMMARY:.....	759
35.1	CONFIGURATION:.....	759
35.2	CONTROLLER:	759
35.3	FOOTERWIDGET:	759
35.4	HEADLINEWIDGET:.....	760
35.5	HOMEWIDGET:	760

35.6	MAPWIDGET:	760
35.7	PFDWIDGET:	760
35.8	MAINMENU:	761
35.9	APPCONFIG:	762
35.10	LIBRARIES:	763
36	REFERENCES:	764
	DEVELOPMENT OF A FCS (FLIGHT CONTROL SYSTEM) (2018).....	765
37	SYSTEM ARCHITECTURE	768
38	SERVO ACTUATOR SYSTEM	769
38.1	ADOPTED MOTOR	769
38.2	BASIC PARTS OF THE SERVO	769
38.3	SERVO MOTOR BLOCK DIAGRAM	770
38.3.1	<i>Control computer:.....</i>	<i>770</i>
38.3.2	<i>Electronic control system:.....</i>	<i>770</i>
38.3.3	<i>Motor:.....</i>	<i>770</i>
38.3.4	<i>Gear train:</i>	<i>771</i>
38.3.5	<i>Position sensor:.....</i>	<i>771</i>
38.3.6	<i>Servo output:</i>	<i>771</i>
38.4	DESIGN OF SERVO ACTUATOR SYSTEM (FREECAD)	771
38.5	MOTOR CONTROLLER AND INTERFACES	771
38.5.1	<i>Mobile App:</i>	<i>772</i>
38.5.2	<i>Bluetooth Module: (BLE Link -A Bluetooth 4.0 module for Arduino)</i>	<i>772</i>
38.5.3	<i>Arduino controller: Romeo V2-All in one Controller-motor drive built in</i>	<i>773</i>
38.5.4	<i>Motor drive: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N 775</i>	
38.6	POWER MANAGEMENT: POLYMER LITHIUM ION BATTERY - 1000MAH 7.4V	776
39	REALIZATION OF INERTIAL MEASUREMENT UNIT (IMU).....	778
39.1	IMU-SENSOR: LSM9DS0	779
39.2	LSM9DS0 HOOKUP GUIDE	780
39.2.1	<i>Covered In This Tutorial</i>	<i>780</i>
39.2.2	<i>About the LSM9DS0.....</i>	<i>780</i>
39.2.3	<i>Choose Your Own Adventure: SPI or I²C.....</i>	<i>781</i>
39.2.4	<i>Breakout Overview</i>	<i>782</i>
39.2.4.1	<i>The Pinout.....</i>	<i>782</i>
1)	<i>Basic Arduino Example</i>	<i>785</i>
2)	<i>Download and Install the Library.....</i>	<i>785</i>
3)	<i>Simple Hardware Hookup (I²C).....</i>	<i>785</i>
4)	<i>Open the LSM9DS0_Simple Example</i>	<i>786</i>
39.2.5	<i>Resources and Going Further.....</i>	<i>787</i>
40	WIFI COMMUNICATION WITH NRF24L01	788
40.1	ARDUINO LIBRARY	789
41	SOFTWARE DEVELOPMENT ON ARDUINO SIDE WITH THE ARDUINO INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)	792
41.1	ARCHITECTURE OF THE PROGRAM ON THE ARDUINO.....	792
41.2	BLUETOOTH_BLIMP_CONTROL CODE	792

Content

41.3	IMU SENSOR LSM9DS0 CODE	795
41.4	CONTROL LOOP	797
41.5	WIFI COMMUNICATION CODE.....	797
41.6	FULL CODE	800
42	ASSEMBLY	808
42.1	ACTUATOR ASSEMBLY	808
42.2	IMU ASSEMBLY	814
42.3	WIFI MODULE ASSEMBLY	814
42.4	FULL ASSEMBLY	815
43	FINAL IMPLEMENTATION	816
43.1	MIGRATED TO SMALL PROTOTYPE	816
44	MINTAD PROTOTYPE TEST RIG QUICK GUIDE	817
	MARKETING CONCEPT (2017-2019)	818
	LITERATURE	820

Project Management

The project had the following phases:

1. Development of a flight control system for an airship (1999-2006)
2. Development of a communication system (uplink and downlink) (2013)
3. Aerodynamic Investigations for a High-Altitude Airship (Platform) (2017)
4. Building of a small prototype of an airship (2017)
5. Development of a FCS and GCS (Ground Control System) (2018)

1 System Requirements Analysis (تحليل متطلبات المنظومة)

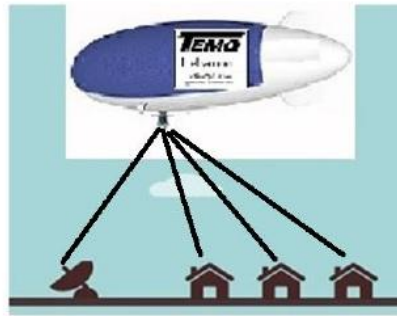
1.1 نشر اولي للمشروع في 18.2.2017

Initial Project (published 18.2.2017, talked to investors)

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Qalamoun-COM_Mintad 2017

نظام لتوزيع انترنت للقلمون والجوار



90.000\$	كلفة المنظومة (بالون واحد زائد ايليكترونيك التحكم ومنظومة الاتصالات)
	كلفة التشغيل سنوياً
\$36,000	كلفة التشغيل (للمراقبة شخص في كل دوام، 3 دوامات = 3 اشخاص)، \$1200 للشخص الواحد شهريا
\$14,000	تكاليف تصليح (سنويا)
	الربح سنوا (مدخول ناقص تكاليف التشغيل والتصليح)
	سعر الانترنت للمستخدم: \$20 في الشهر، 1000 مستخدم -> المدخول سنويا \$240,000
	الربح سنويا \$190,000

1.1.1 حساب التكلفة (Calculation)

The concept is to take a fast internet bundle from OGERO and to distribute it.

من OGERO		عدد المستهلكين الاجمالي	معدل الاستهلاك الشهري [GB]	بيع الخدمة للمستهلك (ل.ل.)
الرسوم الشهري لـ 100 GB (ل.ل.)		3000	20	LBP 30.000
LBP 100.000	ADSL More than 8Mbps			
			عدد المستهلكين تسعهم ال package من OGERO	
			5	
		الربح الاجمالي في الشهر LBP 30.000.000	الربح علي ال package LBP 50.000	

الربح السنوي (\$) \$240.000

الربح هنا هو الدخل السنوي لمشروع المنطاد

1.1.2 لقاء مع الاستاذ باسل الايوبي، مدير OGERO في 19.2.2017 في مسجد القبيسي

نتيجة:

- الدولة اللبنانية اصبحت لا تعطي رخص لشركات خاصة لبث الانترنت. ولكن عن طريق الجامعة ممكن ان شاء الله.
- ما تفعله google هو ان يكون لديها مناطيد لبث الانترنت من فوق 20 كم وهذا خارج الاجواء الدولة التي تحت هذا المنطاد واصبحت اخواء دولية مثل البحر البعيد عن الشاطئ.

1.2 متطلبات المنظومة (System Requirements)

[Sys 10]

المنطاد يجب ان يكون



أ - في علو 2 كم



ب - في علو 20 كم (اي خارج نطاق اجواء الدولة اللبنانية)

[Sys 20]



جسم المنطاد مثبت بأضلع

1.3 تصميم وتصنيع المنطاد (TEMOLeb-Mintad/Mechanics)

TEMOLeb-Mintad/Mechanics: Construction and Manufacturing of Airship

Master Thesises

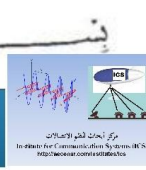
Title	Keywords, additional information	Preferred Faculty/ Student Profile	Project
Construction of a solar powered high altitude airship	Mechanics, FEM analysis, airship, photovoltaics	Energetics	TEMOLeb-Mintad
Aerodynamic investigation for high-altitude airship	Aerodynamics analysis, airship	Energetics	TEMOLeb-Mintad
Mechanical constructing and testing of an actuator system for a high-altitude	CAD, actuator system, airship	Energetics	TEMOLeb-Mintad

1.4 ايلكترونيك في المنطاد (TEMOLeb-Mintad/Electronics)

Master Thesises

Title	Keywords, additional information	Preferred Faculty/ Student Profile	Project
Telemetry system for a high-altitude airship	Remote control, automotive control, sensor integration	Electronics/Control	TEMOLeb-Mintad
Satellite based internet communication through a high-altitude platform	Satellite communication, transponder, antenna control	Telecommunication/ Electronics	TEMOLeb-Mintad

2 Posters



رأس مسقا | طرابلس لبنان،
أذار 2018

TEMOLeb Mintad 1999- 2018



نظام لتوزيع انترنت



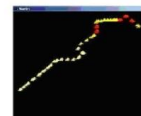
منصة عالية الارتفاع شبه ثابتة توفر وسيلة لإيصال خدمة الانترنت إلى منطقة واسعة
أنه مصمم خصيصا لتعمل على علو كبير (18 كم) او على علو صغير (2 كم)

أجهزة قياس للطاقة البديلة

في الوقت الراهن يتم تطوير و إنتاج جهاز قياس متحرك "المنطاد البديل". بواسطة هذا المنطاد يمكن تركيب أجهزة قياس دقيقة جدا لقياس شدة الرياح و قوة الطاقة الشمسية في مكان معين. بهذا يمكن تحديد الموضع المثالي لمحطة إنتاج طاقة هوائية أو شمسية. هذا المشروع يتم بالتعاون مع جامعة "كارلس روه" و المدرسة الفنية العليا بكارلس روه و جامعة شتوتغارت.



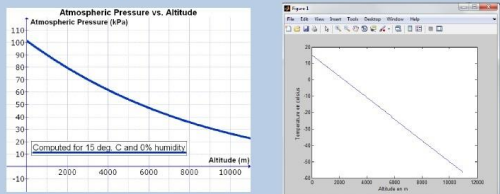
واجهة مستخدم لأجهزة الاستشعار



شدة الإشعاع الشمسية (Simulation) المكان

هذا المشروع "المنطاد البديل" جاري العمل فيه منذ العام 1999 و قد تم من خلال هذا المشروع إنجاز العديد من مشاريع و رسالتات التخرج الجامعية حتى الآن بالإضافة إلى توفير العديد من فرص التدريب

Aerodynamic Investigation

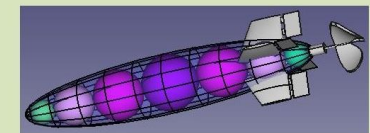
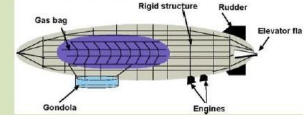


Materials of the test device for the Low altitude test TEMO-Leb Airship in Lebanese market - Balloons filled with Helium

The gasbags are replaced by the balloons, having 1.5 m as a diameter. It is important to know the mass hold by the balloons and the pressure of Helium contained into the balloon. To obtain these parameters, we want to perform this calculation:

Firstly, we calculate the volume of balloon:
 $V_{\text{Balloon}} = \frac{4}{3} \pi (D/2)^3 = \frac{4}{3} \pi (1.5/2)^3 = 1.77 \text{ m}^3$
 We know the density of Helium at 20°C: $\rho_{\text{He}} = 0.178 \text{ kg/m}^3$
 By applying Eq.5, we obtain the mass theirs hold by the balloons:
 $m_{\text{hold by 1 balloon}} = (\rho_{\text{He}} - \rho_{\text{air}}) \cdot V_{\text{Balloon}} = ((1.2041 - 0.178) \cdot 1.77) \text{ kg} = 1.8 \text{ kg}$
 $m_{\text{hold by 15 balloons}} = 4 \cdot 1.8 \text{ kg} = 7.2 \text{ kg}$
 The pressure of Helium into the balloon can be calculated by using the equation:
 $P \cdot V = (m/M) \cdot R \cdot T$ Eq.10
 P: Pressure of Helium into the balloon (Pa)
 V: Volume of balloon (m³)
 m: Mass of balloon (kg)
 M: Molar mass of Helium (kg/m³)
 R: The universal gas constant = 8.314 Jk⁻¹mol⁻¹
 T: The absolute Temperature of Helium (k)
 To obtain the mass, we use Eq.5:
 $\rho = m/V; m = 0.178 \cdot 1.77 = 0.3 \text{ kg/m}^3$
 The pressure is:
 $P = ((m/M) \cdot R \cdot T) / V = ((0.3/4) \cdot 8.314 \cdot 293.15) / 1.77 = 108457.6 \text{ Pa} = 1.07 \text{ atm}$
 ρ: Density of Helium at 20°C = 0.178 kg/m³

Air vehicle Construction



Commercialization

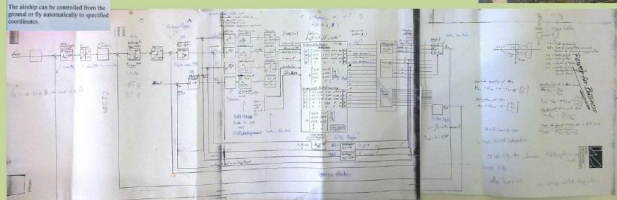
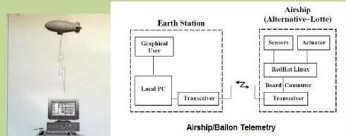
المشروع هو توزيع انترنت فوق العالون 3000 مستخدم

90.000\$	تكلفة البنية التحتية (تأجير وادارة البنية التحتية وخطوط الاتصالات)
تكلفة التشغيل سنويا	
\$36.000	تكلفة التشغيل (تأجير شخص واحد و 3 دوات - 3 ايام) \$1200 للشخص الواحد سنويا
\$14.000	تكاليف صيانة (سنويا)
	الربح سنويا (بعد خصم تكاليف التشغيل والصيانة)
	سعر الانترنت للمستخدم: \$20 في الشهر، 3000 مستخدم > الدخل سنويا \$240.000
	الربح سنويا \$190.000

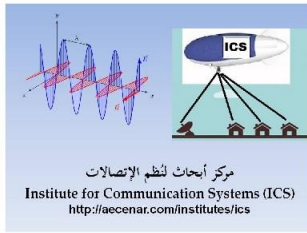
Calculation: The concept is to take a fast internet bundle from OGERO and to distribute it.

من OGERO	عدد المستخدمين الاجمالي	سعر الخدمة للمستهلك (ل.ل.)	سعر الخدمة للمستهلك (دولار)
GB 100	3000	30.000	LBP 30.000
ADSL More than 3Mbps			
LBP 100.000			
عدد المستخدمين من package J OGERO			
5			LBP 50.000
الربح الاجمالي في الشهر			LBP 30.000.000
الربح السنوي (5)			\$240.000
الربح هذا هو الدخل السنوي للمشروع المنطاد			

Flight Control system

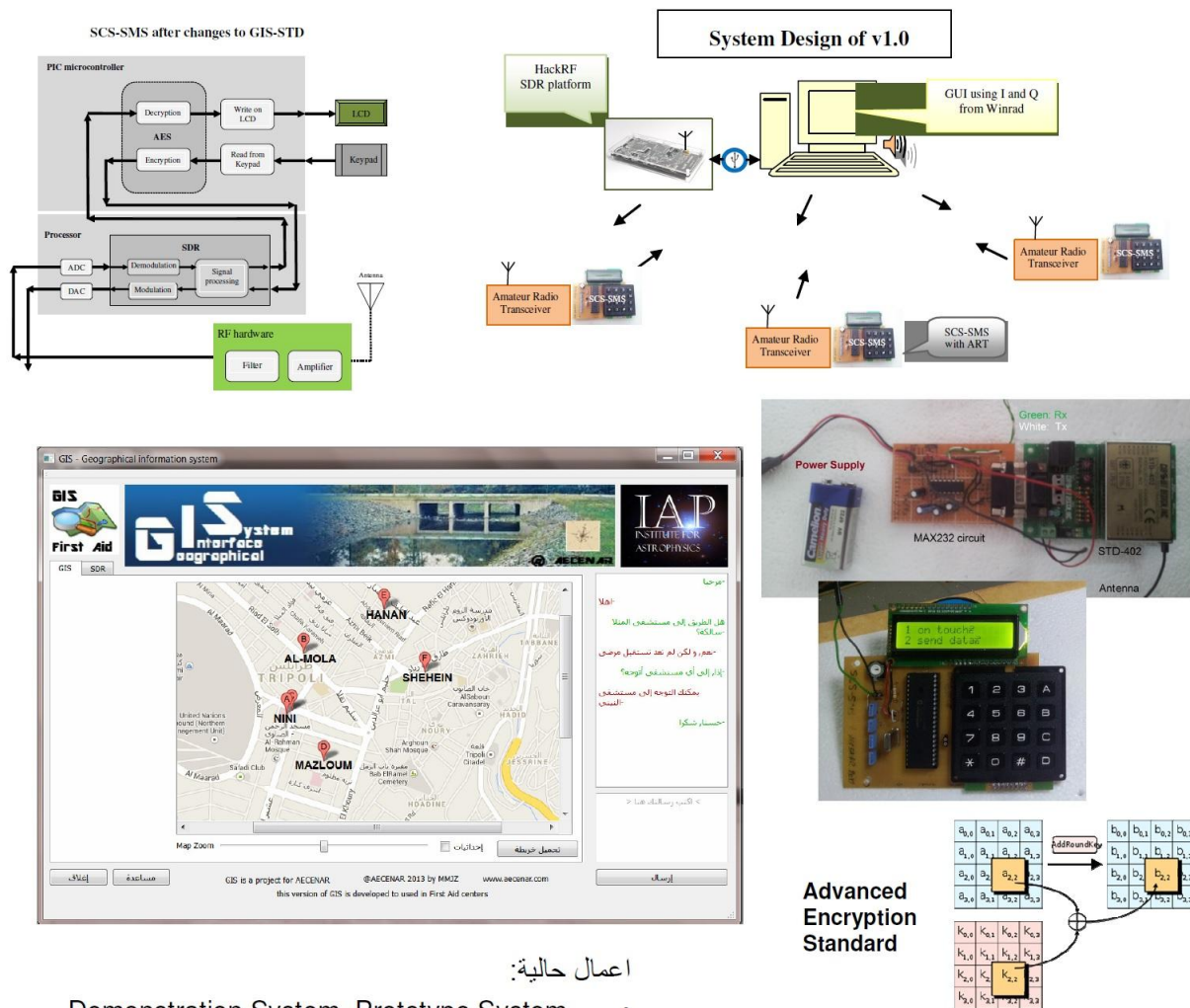


Block diagram of the Control Loops



Emergency SDR Communication System شبكة اتصالات آمنة للطوارئ و الاسعافات

Last update: 03 December 2013



اعمال حالية:

Demonstration System, Prototype System •

الحاجيات لعام 2014:

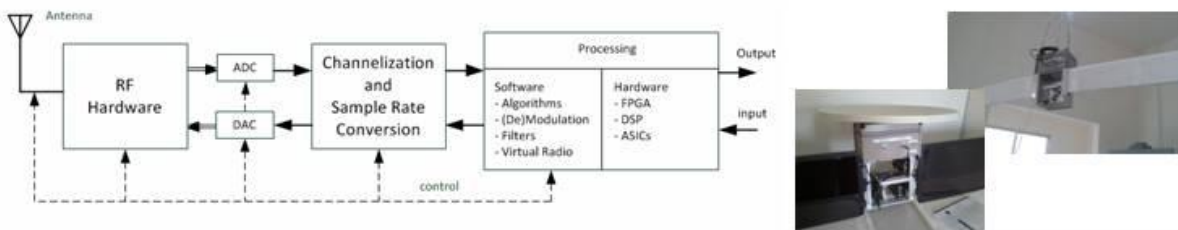
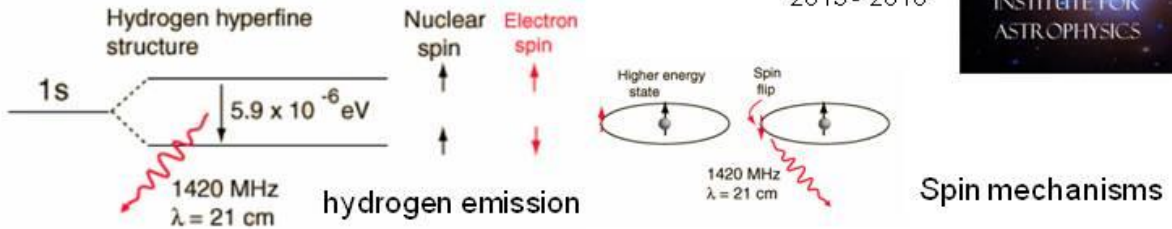
- 2 اشخاص
- \$5.000 للمواد

Contact:
Mahmoud Jandal Zohby
Mobile: +9613671621
mahmoud.j.zohby@aecenar.com

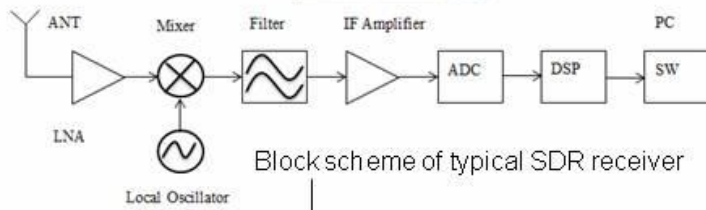


Payload of IAP-SAT

2013 - 2018



Model of Software Radio



System design overview

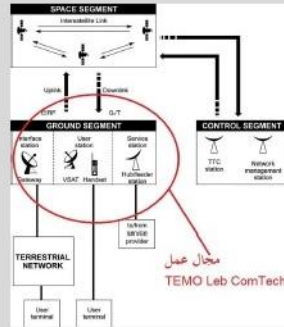
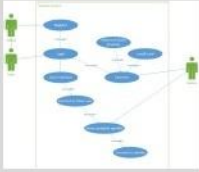
For details see IAP-SAT Final Report (2012-2020)

Satellite Based Communication Systems



مشروع TEMOLeb_SatInterface

من خلال المشروع الحالي نحذف الى انشاء محطات ابراج لتسهيل التواصل بالاقمار الصناعية لتسهيل وتسريع التواصل، ومن اجل هذا الهدف لا بد من انشاء عدة تقنيات من بينها برنامج التحكم بالابرار وايضا لتمكين المستخدمين من الولوج الى خدماتنا من خلال واجهة مستخدم من عدة منصات كالحاسب والهاتف .



البرنامج منقسم الى جزئين



واجهة المدراء (Operators Interface)

من خلال هذه الواجهة يقوم المدراء كل حسب تخصصه بمراقبة الاقمار الصناعية والابرار المنتشرة لمعرفة كمية سر عملها واكتشاف الاخطاء والمشاكل نظارة للسعي الى حلها



واجهة المستخدم (User Interface)

هذه الواجهة مخصصة للمستخدم النهائي او العميل التي من خلالها يستطيع التسجيل والاشتراك بالإضافة الى استخدام الخدمات التوفرة له، يجب ان تتوفر هذه الواجهة على عدة منصات كالهاتف والحاسب وأظفة تشغيل ليتمكّن كل شخص مهما كان نوع جهازه من الاشتراك كما يجب ان تكون بسيطة وسهلة الاستخدام بعيدة عن التعقيد ونوعية قدر الامكان تناسب جميع الاجهزة .

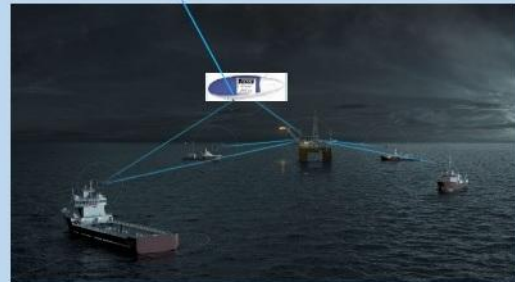
Balloon Based Communication Systems

Applications:



- Offshore internet supply (on sea)

Actual Project: Internet supply for Gas exploration facilities in front of the Lebanese coast



- High Altitude Communication Platform



AECENAR
Association for Economical and Technological Cooperation
in the Euro-Asian and North-African Region

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



TEMOLebanon
منصات للإتصالات
platforms for communication

TEMOLeb-SatellitePlatform

(Communication Satellite Platform) نظام قمر اصطناعي للإتصالات



ICS
المركز التقني للإتصالات والمعلوماتية





3000 Kg Payload


COM Unit for GEO

Payload

Telecommunications

Transporter



About 5000\$ /kg
For 3000 kg = 15 Mio \$

Pre-Development (2012-2019)

Sensors IMU

IMU or Inertial Measurement Unit



The IMU has the following parts:

- Gyro
- Accelerometer
- Magnetometer



Communication SDR

The radio astronomical IAP project supernova radio wave detector and analyzer SRWDA





Chemical Prop. Unit

The satellite chemical propulsion system is based on this system

- Methane tank
- Oxygen tank
- Thrust chamber



Transporter propulsion system design



Electrical Prop. Unit

Pulsed Plasma Thruster



EDDY current sensor




Tasks	Needed Human Resource	External Cost	Year
TEMOLeb-Mintad Gas filling	2 Technicians, 4 MM (man months)	5000\$	2019
TEMOLeb-Mintad& TEMOLeb-SAT solar panels and system	-	2000\$	2019
TEMOLeb-Mintad Control System development & Validation (stimulink/scilab)	Eng., 5 MM	400\$	2019
TEMOLeb-Mintad Actuators, Wings integration	2 Technicians, 2 MM	1000\$	2019
TEMOLeb-Mintad Sensors integration	Eng., 1 MM	300\$	2019
TEMOLeb-SAT Actuator (chem.)	1 Technician, 1 MM	1500\$	2019
TEMOLeb-SAT Platform integration	1 Technician, 1/2 MM	500\$	2019
TEMOLeb-Mintad Telemetry Unit	2 Eng., 5 MM	3000\$	2020
TEMOLeb-SAT Telemetry Unit	2 Eng., 5 MM	3000\$	2020
TEMOLeb-Mintad COM Unit (Payload)			2020
TEMOLeb-SAT COM Unit (Payload)			2020

@AECENAR – ICS May2019 MMJZ
www.aecenar.com/index.php/institutes/ics



Development of a flight control system for an airship (1999-2006)



TEMO
Soft-, Hardware & Cons. e.K.

Association for Alternative Energy Research e.V.,
www.aecenar.com/c_institutes/vaef

TEMO Soft-, Hardware & Consulting e.K.
www.temo-ek.de

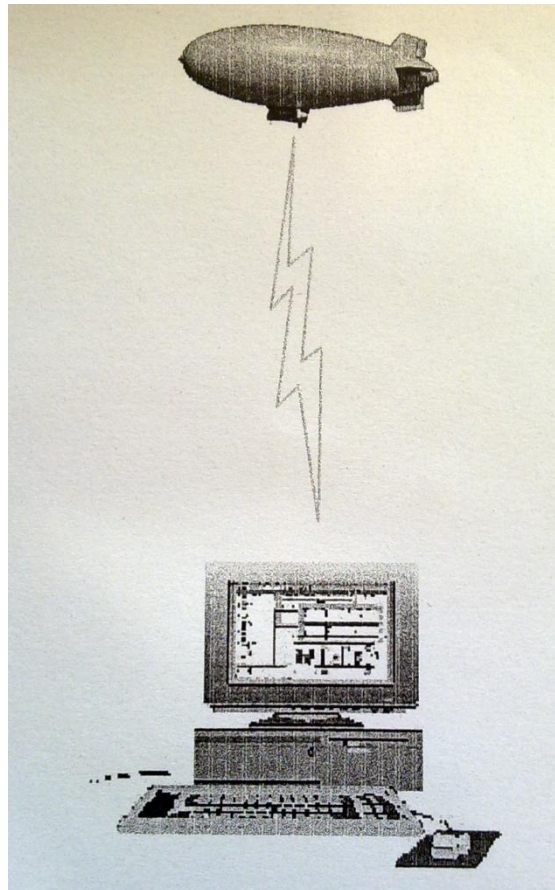
VaEf Airship project "alternative Lotte": Project Report (August 2000 – July 2006)

Development of a Flight Control System for a airship for measuring wind data

Based on student research works of

Samir Mourad, Sven Forstmann, Jamal E., Muhammad Subhan, Rabih al-Farkh, Nasih Abdelhaqq, Yaser Mikou, Abdelfattah Ammar

Project Manager: Samir Mourad, Dipl.-Ing. Dipl.-Inform. (MS Sc. Computer Science, MS Sc. Electrical Engineering)



Mourad et. al.:
Karlsruhe, July 2006

Veröffentlicht von:

Verein für alternative Energieforschung e.V.

CONTENT IN SHORT

PROJECT MANAGEMENT	1
1) تحليل متطلبات المنظومة (SYSTEM REQUIREMENTS ANALYSIS(.....	2
2 POSTERS.....	5
DEVELOPMENT OF A FLIGHT CONTROL SYSTEM FOR AN AIRSHIP (1999-2006)	11
ABSTRACT.....	26
3 INTRODUCTION	28
4 PROJECT MANAGEMENT	29
5 FLIGHT CONTROL SYSTEM OF AN AIRSHIP	30
6 OPTIMIZATION OF THE FLIGHT CONTROL SYSTEM ARCHITECTURE.....	62
7 DEVELOPMENT OF THE EREIGNISDIENSTES FOR THE MIDDLEWARE OSA+	79
8 INERTIAL MEASUREMENT UNIT.....	128
TABLE OF CONTENTS	128
LIST OF FIGURES:	130
LIST OF TABLES	132
LITERATURE	171
9 ACTUATOR BOARD	173
TABLE OF CONTENTS	175
LIST OF FIGURES:	177
LITERATURE	236
APPENDIX A.....	237
ANNEX B.....	240
10 COMMUNICATION AND USER INTERFACE	249
FIGURE 6.6: STD-402 TRANSCEIVER – <i>CIRCUIT DESIGN</i>	276
ACK AUTO RESPONSE SETTING	293
FEATURES	296
➤ <i>REGISTRATION OF ID</i>	315
11 ANNEX D	322
OR THE BOARD USED TO EQUIPS THE STD-402 IS THE MB-STD-RS232, SO THE THIS MODE (AUTO MODE OPERATION GUIDE FOR CPU INTERFACE) IS USED LIKE MODE OF THE STD-402 TRANSCEIVER.	322
12 SOME REPAIRS ON THE SENSOR CARD AND FIRST STEP INTEGRATION	376
13 INTEGRATION.....	410
14 INTEGRATION CODE IN C.....	489
DEVELOPMENT OF A COMMUNICATION SYSTEM (2013)	497
ABBREVIATIONS	501

15	ABSTRACT	503
16	PROJECT MANAGEMENT	505
17	BASICS.....	507
	FIGURE 6.6: STD-402 TRANSCEIVER – <i>CIRCUIT DESIGN</i>	546
	OR THE MB-STD-RS232 EQUIPS STD-402 TRANSCEIVER MODULE AND PERFORMS PACKET COMMUNICATION USING CPU INTERFACE MODE OF THE TRANSCEIVER.....	547
18	SPECIFICATION	551
19	SYSTEM DESIGN	553
20	MECHANICS.....	557
21	SCS-SMS	559
22	AES ENCRYPTION.....	583
23	HARDWARE OF ECS DEMO SYSTEM.....	599
24	FURTHER WORK: SYSTEM INTEGRATION AND INTEGRATION TEST OF ECS DEMO SYSTEM.....	611
	APPENDIX A: ALTERNATIVE PROJECT PLANS.....	612
	APPENDIX B: ALL ABOUT HARCKRF	616
	APPENDIX C: ALTERNATIVE SYSTEM DESIGNS.....	624
	LITERATURE	625
	COMMUNICATION SYSTEM WITH HARCKRF (FROM IAP-SAT, 6TH PROJECT REPORT) (2020).....	626
25	TELEMETRY SYSTEM WITH HARCKRF	627
	AERODYNAMIC INVESTIGATIONS FOR A HIGH-ALTITUDE AIRSHIP (PLATFORM) (2017)	644
	BUILDING OF A SMALL PROTOTYPE OF AN AIRSHIP (2017).....	697
	DEVELOPMENT OF A FCS (FLIGHT CONTROL SYSTEM) AND GCS (GROUND CONTROL SYSTEM) (2018)	745
	MARKETING CONCEPT (2017-2019)	765
	LITERATURVERZEICHNIS	820

Content

1	PROJECT MANAGEMENT	
2	تحليل متطلبات المنظومة (SYSTEM REQUIREMENTS ANALYSIS)	1
2.....	نشر اولي للمشروع في 18.2.2017	1.1
3.....	حساب التكلفة (Calculation)	1.1.1
3.....	لقاء مع الاستاذ باسل الابويبي، مدير OGERO في 19.2.2017 في مسجد القبيسي ...	1.1.2
3.....	متطلبات المنظومة (SYSTEM REQUIREMENTS)	1.2
4.....	تصميم وتصنيع المنطاد (TEMOLEB-MINTAD/MECHANICS)	1.3
4.....	ايلكترونيك في المنطاد (TEMOLEB-MINTAD/ELECTRONICS)	1.4
2	POSTERS	5
	DEVELOPMENT OF A FLIGHT CONTROL SYSTEM FOR AN AIRSHIP (1999-2006)	11
	ABSTRACT	26
3	INTRODUCTION	28
3.1	THE LOTTE PROJECT AND THE "ALTERNATIVE LOTTE".....	28
4	PROJECT MANAGEMENT	29
4.1	COSTS.....	29
5	FLIGHT CONTROL SYSTEM OF AN AIRSHIP	30
5.1	BLOCK DIAGRAM OF THE CONTROL LOOPS.....	61
6	OPTIMIZATION OF THE FLIGHT CONTROL SYSTEM ARCHITECTURE	62
6.1	THE ARCHITECTURE OF THE FLIGHT CONTROL SYSTEM AND ITS SIMULATION ON THE MIDDLEWARE OSA+ RUNNING ON THE OPERATING SYSTEM WINDOWS.....	62
6.2	RESULTS.....	63
6.2.1.1	System Architecture of Airship Flight Control System.....	71
6.2.1.2	Simulation Code (in programming language C).....	71
6.2.1.3	Simulation results.....	71
7	DEVELOPMENT OF THE EREIGNISDIENSTES FOR THE MIDDLEWARE OSA+	79
7.1	ABRIDGED VERSION.....	82
7.2	THE NATURE OF THE TASK.....	82
7.3	THE OSA+ ARCHITECTURE.....	82
7.3.1	Introduction.....	83
7.3.2	The components of OSA+.....	84
7.3.2.1	The Platform.....	84
7.3.2.2	Generic Services.....	85
7.4	COMPONENTS OF THE OSA EVENTING SERVICE.....	86
7.4.1	The time functions.....	86
7.4.1.1	The handling of the time.....	86
7.4.1.2	Initialization.....	87
7.4.1.3	Functions for time management.....	89
7.4.2	The event service.....	90
7.4.2.1	General Information.....	90
7.4.2.2	The initialization of the Event Service.....	92

7.4.2.3	Features of the Ereignisdienstes.....	92
7.5	SHORT OVERVIEW	94
7.5.1	<i>Time functions/variables</i>	94
7.5.2	<i>Features of the Event Service (Ereignisdienst)</i>	94
7.6	CONFIGURATION	94
7.7	THEORY OF OPERATION	96
7.7.1	<i>Osagetime()</i>	96
7.7.2	<i>Osasettime(int,int)</i>	96
7.7.3	<i>Osaaddtime(int,int)</i>	98
7.7.4	<i>Char * osaPrintTime(int, int)</i>	98
7.7.5	<i>Osainitevents()</i>	99
7.7.6	<i>Int osaAddEvent(uint,uint, uint, uint,uint, uint,uint, uint, function)</i>	99
7.7.7	<i>Int osaDelEvent(uint,uint,uint,UINT)</i>	101
7.7.8	<i>OSA_Error osaGetEventResult(uint,UINT)</i>	101
7.7.9	<i>Internal functions of the Ereignisdienstes</i>	102
7.8	PROGRAMMING EXAMPLES	102
7.8.1	<i>For example: Changing the Time</i>	102
7.8.2	<i>Example: Add an Event</i>	103
7.8.2.1	Start a function at a specified time.....	103
7.8.2.2	Repeated start a function	105
7.8.3	<i>Queries of results</i>	106
7.8.4	<i>Delete a Events</i>	106
7.9	TEST RUNS.....	107
7.9.1	<i>Deliver-Test</i>	109
7.9.2	<i>Test run of a cyclical events with open</i>	110
7.9.3	<i>Test the maximum temporal resolution on different systems</i>	113
7.9.3.1	System 1	113
7.9.3.2	System 2	116
7.9.3.3	System 3	118
7.9.4	<i>Stability tests</i>	121
7.9.4.1	Exceeding the maximum acceptable number of Events	121
7.9.4.2	Exceeding the maximum acceptable number of events per unit time	122
7.9.4.3	Overrun of the events per unit time through Zeituberschneidung	123
7.10	RESULTS AND OUTLOOK	126
7.11	ANNEX	127
7.11.1	<i>List of all Windows specific functions</i>	127
7.12	LITERATURE.....	127
8	INERTIAL MEASUREMENT UNIT.....	128
	TABLE OF CONTENTS	128
	LIST OF FIGURES:	130
	LIST OF TABLES	132
8.1	INTRODUCTION	133
8.1.1	<i>Task</i>	133
8.1.2	<i>Overview</i>	134
8.2	BASICS	135
8.2.1	<i>Coordinate System</i>	136
8.2.2	<i>Calculation of the current position</i>	137
8.2.3	<i>Compass</i>	138

8.3	ARCHITECTURE DESIGN OF THE IMU	139
8.3.1	<i>Measurement Data Collection (Meßdatenaufnahme)</i>	139
8.3.2	<i>Measurement Data Processing (Meßdatenaufbereitung)</i>	139
8.4	DEVELOPMENT ENVIRONMENT.....	141
8.4.1	<i>Hardware-Entwicklungsumgebung</i>	141
8.4.2	<i>Software Development Environment</i>	142
8.5	REALIZATION OF THE IMU	143
8.5.1	<i>Meßdatenaufnahme</i>	143
8.5.1.1	Circuit Design for the Meßdatenaufnahme	143
8.5.1.2	Reference Voltage	150
8.5.2	<i>Board layout design for the Meßdatenaufnahme (Measurement data collection)</i>	151
8.5.3	<i>Meßdatenaufbereitung</i>	153
8.5.3.1	Circuit Design for the Meßdatenaufbereitung.....	153
8.5.4	<i>Board layout design for the Meßdatenaufbereitung</i>	155
8.5.4.1	The software for the Meßdatenaufbereitung.....	157
8.5.4.2	User Interface	161
8.6	TEST RESULTS.....	165
8.6.1	<i>Hardware-Test</i>	165
8.7	TEST OF THE OVERALL SYSTEM WITH THE MICROCONTROLLER (C167).....	166
8.7.1	<i>Gesamtsystem-Test 1</i>	166
8.7.2	<i>Gesamtsystem-Test 2</i>	168
8.7.3	<i>Gesamtsystem-Test 3</i>	168
8.8	SUMMARY AND OUTLOOK	170
LITERATURE		171
9 ACTUATOR BOARD		173
TABLE OF CONTENTS		175
LIST OF FIGURES:		177
9.1	INTRODUCTION	181
9.1.1	<i>The Solarluftschiff Lotte</i>	181
9.1.2	<i>Task</i>	182
9.1.3	<i>Overview</i>	182
9.2	BASICS	183
9.2.1	<i>Real-time systems</i>	183
9.2.2	<i>Hardware</i>	183
9.2.2.1	Components and classifications.....	183
9.2.2.2	The microcontroller family C166	184
9.2.2.3	The C167 microcontroller family	185
9.2.2.4	The memory organization of the C167)	186
9.2.2.5	The interrupt system of the C167).....	188
9.2.2.6	The Timereinheiten	190
9.2.2.7	Capture Compare Unit (Capcom).....	191
9.2.2.8	The PWM Pulsweiten-Einheit	193
9.2.2.9	Analog Digital Converter (ADC).....	194
9.2.3	<i>Control of servo motors</i>	195
9.3	ARCHITECTURE DESIGN OF THE AKTORIK-ANSTEUERUNGSEINHEIT (ENGL, <i>ACTUATOR CONTROL UNIT (ACU)</i>)	197
9.3.1	<i>Voltage regulation</i>	197
9.3.2	<i>Betriebsspannungsüberwachung</i>	198
9.3.3	<i>Basisbetriebszustands-Einstellung</i>	198

9.3.4	<i>Steuersignal-Erzeugung</i>	198
9.3.5	<i>Notsteuerungsbaugruppe</i>	199
9.3.6	<i>Operating conditions of the ACU</i>	200
9.3.7	<i>Modularity of the ACU</i>	200
9.4	DEVELOPMENT ENVIRONMENT	202
9.4.1	<i>Hardware-Entwicklungsumgebung</i>	202
9.4.2	<i>Software Development Environment</i>	203
9.5	REALIZATION OF THE ACU	204
9.5.1	<i>Versorgungsspannungs-Stabilisierung</i>	204
9.5.1.1	Circuit Design for the Versorgungsspannungs-Stabilisierung	205
9.5.2	<i>Betriebsspannungsüberwachung</i>	207
9.5.3	<i>Basisbetriebszustands-Einstellung</i>	208
9.5.3.1	Circuit Design for the Basisbetriebszustands-Einstellung	211
9.5.4	<i>Steuersignal-Erzeugung</i>	212
9.5.4.1	The hardware of the Steuersignal-Erzeugung	214
9.5.4.2	The monitoring and the forwarding of the PWM-signals from the remote control (normal operation (B))	214
9.5.4.3	Level 1 Notsteuersignal-Erzeugung	219
9.5.5	<i>Notsteuerungsbaugruppe</i>	222
9.5.5.1	Circuit Design for the monitoring of the control signals on the output of the ACU and the channel5-output of the Fernsteuerempfangers	222
9.5.5.2	Circuit Design for the Notsteuerungsbaugruppe	223
9.5.5.3	Circuit Design for the forwarding of the PWM-signals from Fernsteuerungsempfanger (Luftschiff-Startphasen - operation)	224
9.5.6	<i>Layout design of board 1 ("Voltage regulation", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe ")</i>	224
9.5.7	<i>The circuit boards of the ACU</i>	225
9.6	EXPERIMENTAL RESULTS	227
9.6.1	<i>Structure of the Testplatzes</i>	227
9.6.2	<i>Experiments and test sequence</i>	228
9.6.3	<i>The series of tests</i>	228
9.6.3.1	Test 1	229
9.6.3.2	Test 2	230
9.6.3.3	TEST3.	231
9.6.3.4	Test 4	232
9.6.3.5	Test 5	233
9.7	SUMMARY AND OUTLOOK	235
LITERATURE		236
APPENDIX A		237
ANNEX B		240
10 COMMUNICATION AND USER INTERFACE		249
10.1	INTRODUCTION	249
10.1.1	<i>The LOTTE project and the "alternative Lotte"</i>	249
10.1.2	<i>Development method</i>	250
10.1.3	<i>Working task and realization approach</i>	250
10.1.4	<i>Overview</i>	250
10.2	BASICS	251
10.2.1	<i>The V-Model</i>	251
10.2.2	<i>Structured Analysis (SA) / Structured Design (SD)</i>	252

10.2.3	<i>Transceivers</i>	254
10.3	REQUIREMENTS SPECIFICATION.....	254
10.3.1	<i>The Graphical User Interface (GUI)</i>	254
10.3.1.1	Software Requirements Specification.....	255
10.3.2	<i>Specifications for the transceiver</i>	259
10.3.3	<i>Communication Protocol for User Data</i>	259
10.3.4	<i>Handshaking</i>	261
10.4	ARCHITECTURE DESIGN	261
10.5	DEVELOPMENT ENVIRONMENTS.....	263
10.6	DESIGN AND IMPLEMENTATION.....	263
10.6.1	<i>SA / SD and Implementation for the base station program</i>	263
10.6.1.1	SA (Structured Analysis).....	263
10.6.1.2	SD (Structured Design).....	267
10.6.1.3	Implementation	270
10.6.2	<i>Communication part on the board system on the airship</i>	273
10.6.2.1	The transceiver	273
10.6.2.2	The communication software on the embedded board computer	278
10.7	COMPONENT TESTS	279
10.7.1	<i>Transceiver test</i>	279
10.7.2	<i>User interface laboratory test with two PCs</i>	282
10.7.3	<i>Test of the user interface on the target system</i>	284
10.8	ANNEX A.....	285
10.9	ANNEX B.....	295
10.9.1	<i>Block diagram of the STD-402 transceiver in Direct Mode</i>	309
10.10	ANNEX C.....	310
	◆ STD-402TR [Auto Mode Operation Guide]	310
10.10.1	<i>General</i>	310
10.10.2	<i>Features</i>	311
10.10.3	<i>Application Examples</i>	311
10.10.4	<i>Configuration</i>	311
➤	REGISTRATION OF ID	315
10.10.5	<i>diagram of the STD-402 transceiver in Auto Mode</i>	321
11	ANNEX D	322
11.1	326
11.2	STD-402 DATA FORMAT	326
11.2.1	<i>Transmitter : Initial setting flow chart</i>	328
11.2.2	<i>Receiver: Flow chart at power ON</i>	334
11.3	ANNEX E: VISUAL BASIC CODE FOR THE USER INTERFACE PROGRAM	341
11.4	ANNEX F: C PROGRAM CODE	368
11.5	ANNEX G.....	372
12	SOME REPAIRS ON THE SENSOR CARD AND FIRST STEP INTEGRATION	376
13	INTEGRATION	410
13.1	ABSTRACT.....	410
13.2	INTRODUCTION.....	414
13.2.1	<i>The Lotte-Projekt and the "alternative Lotte"</i>	414
13.3	DEVELOPMENT ENVIRONMENT.....	416
13.4	ARCHITECTURE DESIGN AND IMPLEMENTATION.....	417

13.5	LITERATURE.....	487
14	INTEGRATION CODE IN C.....	489
	DEVELOPMENT OF A COMMUNICATION SYSTEM (2013)	497
	ABBREVIATIONS	501
15	ABSTRACT	503
16	PROJECT MANAGEMENT	505
16.1	PROJECT DEFINITION HISTORY	505
16.2	SYSTEM BUDGET (TIME AND COST) FOR DEMO SYSTEM	505
16.3	AT 21 JAN STILL OPEN TASKS FOR IAP ECS DEMO SYSTEM WHEN USING (ONLY INTEGRATION)	505
17	BASICS.....	507
17.1	COMMUNICATION BASICS.....	507
17.1.1	<i>Transmitter design from http://en.wikibooks.org/wiki/Electronics/Transmitter_design</i>	<i>519</i>
17.1.1.1	Frequency synthesis and frequency multiplication	520
17.1.1.2	Frequency mixing and Modulation	521
17.1.1.3	RF power amplifiers	527
17.1.1.4	Linking the transmitter to the aerial.....	527
17.1.1.5	EMC matters.....	528
17.2	RECEIVER DESIGN FROM HTTP://EN.WIKIPEDIA.ORG/WIKI/TUNER_(ELECTRONICS)	533
17.3	ANTENNA	534
17.4	SOFTWARE DEFINED RADIO (SDR)	536
17.5	HSDR (HIGH DEFINITION SOFTWARE DEFINED RADIO).....	536
17.6	EXTIO.DLL	537
17.7	HOW DO I DEVELOP AN EXTIO.DLL ?	537
17.8	VISUAL C++ 2008 EXPRESS	537
17.9	QT	537
17.10	RF HARDWARE (USB STICK)	537
17.10.1	<i>TERRATEC ran T stick DVB-T/DAB/DAB + Stick USB 2.0</i>	<i>537</i>
17.10.2	<i>Hackrf (an-open-source-SDR-platform)</i>	<i>538</i>
17.11	RF OVERVIEW.....	538
17.12	RF FREQUENCIES POLICIES	539
17.13	RF MODULES	543
17.13.1	<i>STD-402</i>	<i>543</i>
17.13.1.1	Special for MB-STD-RS232	543
17.13.1.2	Special for STD-402 (Transceiver)	545
17.13.2	<i>RFM42B-RFM31B 433MHz</i>	<i>547</i>
17.13.3	<i>BOWITZ W.T.</i>	<i>549</i>
17.13.4	<i>Comparison between modules</i>	<i>549</i>
18	SPECIFICATION	551
18.1	SYSTEM REQUIREMENTS	551
18.2	HARDWARE REQUIREMENTS.....	551
18.3	SOFTWARE REQUIREMENTS.....	551
19	SYSTEM DESIGN	553
19.1	SYSTEM OVERVIEW.....	553
19.2	CENTRAL STATION	553
19.2.1	<i>Architecture</i>	<i>553</i>

19.2.2	<i>SDR development side</i>	553
19.2.3	<i>Graphical User Interface</i>	554
19.3	MOBILE STATIONS	555
20	MECHANICS	557
20.1	MECHANICAL DESIGN	557
20.2	PROTOTYPE WITHOUT COVER	557
21	SCS-SMS	559
21.1	ABSTRACT OF SCS-SMS	559
21.2	SYSTEM DESIGN	559
21.3	ARCHITECTURES	560
21.4	PIC SOFTWARE	566
21.5	TEST	579
22	AES ENCRYPTION	583
22.1	INTRODUCTION	583
22.2	AES ALGORITHM	583
22.3	CODING	586
23	HARDWARE OF ECS DEMO SYSTEM	599
23.1	REALIZATION OF RF MODULE	599
23.1.1	<i>Using STD-402</i>	599
23.1.2	<i>Realization of RF Module Using RFM42B-RFM31B – 433MHz</i>	603
23.1.2.1	Serial Peripheral interface (SPI)	603
23.1.2.2	The new hardware design	603
23.1.2.3	MSSP module to establishing (SPI)	605
24	FURTHER WORK: SYSTEM INTEGRATION AND INTEGRATION TEST OF ECS DEMO SYSTEM	611
	APPENDIX A: ALTERNATIVE PROJECT PLANS	612
	APPENDIX B: ALL ABOUT HACKRF	616
	<i>B.1 HackRF overview</i>	616
	<i>B.2 Jawbreaker</i>	618
	<i>B.3 Jellybean</i>	618
	<i>B.4 Lemondrop</i>	619
	APPENDIX C: ALTERNATIVE SYSTEM DESIGNS	624
	LITERATURE	625
	COMMUNICATION SYSTEM WITH HARCKRF (FROM IAP-SAT, 6TH PROJECT REPORT) (2020)	626
25	TELEMETRY SYSTEM WITH HARCKRF	627
25.1	TIME PLAN	627
25.2	INTRODUCTION	628
25.2.1	<i>Links and references:</i>	630
25.2.2	<i>SDR (Software defined radio):</i>	631
25.2.3	<i>HSDR:</i>	631
25.2.4	<i>SDRSharp:</i>	632
25.2.5	<i>GnuRadio:</i>	634
25.2.5.1	Links:	635

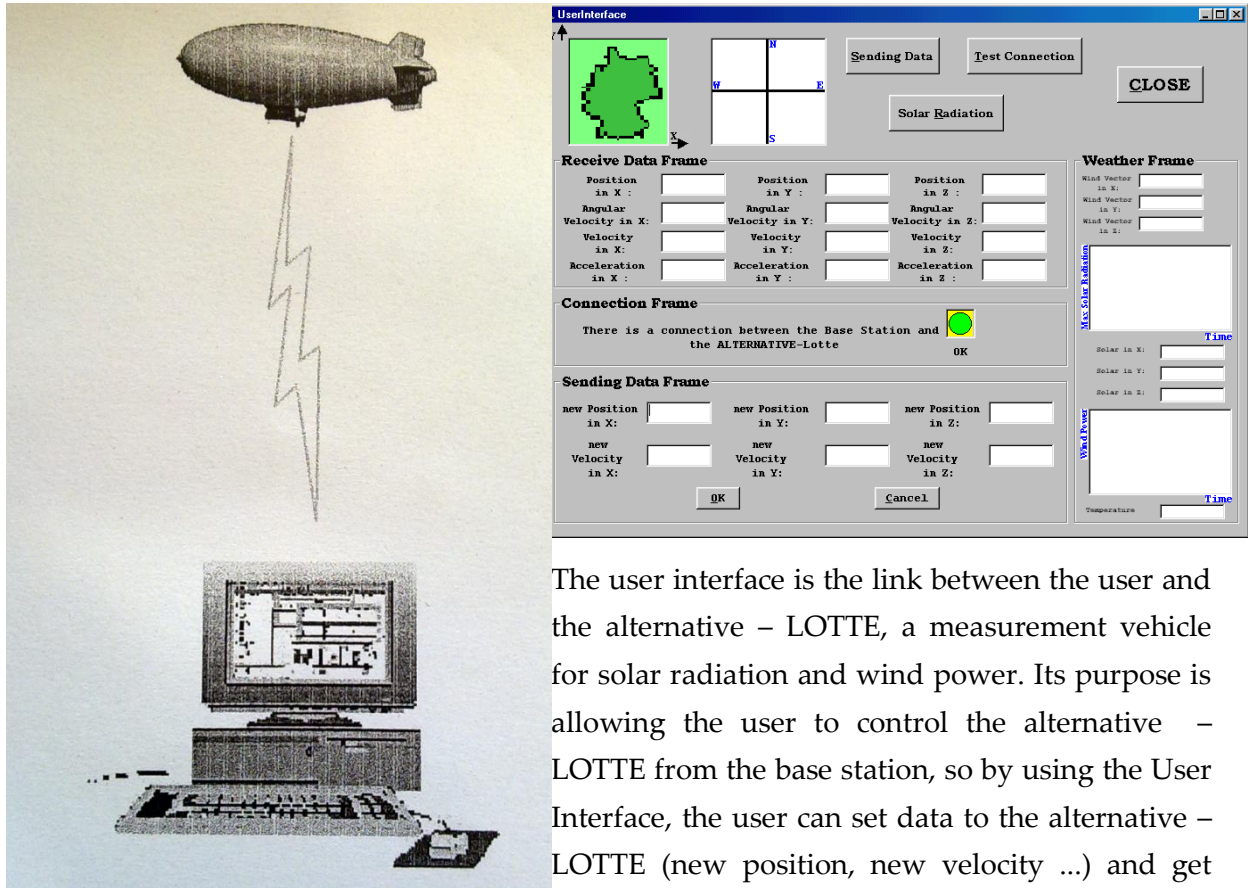
25.3	GETTING STARTED	635
25.3.1	<i>Getting Started with HackRF and GNU Radio</i>	637
25.3.2	<i>HackRF with Raspberry PI</i> :.....	638
25.4	SYSTEM 1	641
25.5	SYSTEM 2	643
25.6	SYSTEM 3	643
	AERODYNAMIC INVESTIGATIONS FOR A HIGH-ALTITUDE AIRSHIP (PLATFORM) (2017)	644
	BUILDING OF A SMALL PROTOTYPE OF AN AIRSHIP (2017)	697
	DEVELOPMENT OF A FCS (FLIGHT CONTROL SYSTEM) AND GCS (GROUND CONTROL SYSTEM) (2018)	745
	MARKETING CONCEPT (2017-2019)	765
	LITERATURVERZEICHNIS	820

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Abstract

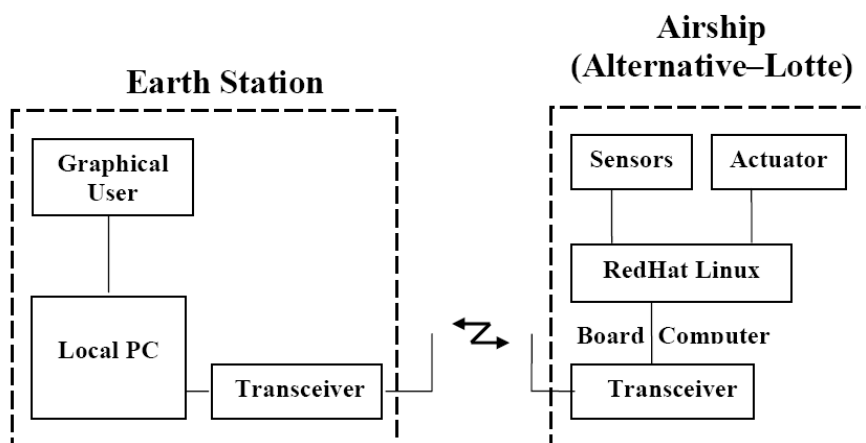
In this project report the "alternative Lotte" airship flight control system project and its results for August 2000 – July 2006 are described. In summer 2006 the project was cancelled.

The alternative Lotte project aimed to improve the flight control system of the solar airship "Lotte" of University of Stuttgart.



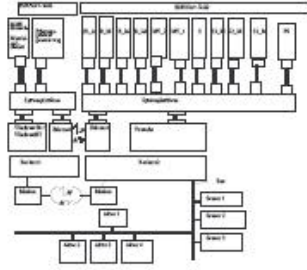
The airship can be controlled from the ground or fly automatically to specified coordinates.

The user interface is the link between the user and the alternative – LOTTE, a measurement vehicle for solar radiation and wind power. Its purpose is allowing the user to control the alternative – LOTTE from the base station, so by using the User Interface, the user can set data to the alternative – LOTTE (new position, new velocity ...) and get data from the alternative – LOTTE (acceleration, angular velocity, azimuth, temperature, wind vector, Solar radiation ...).



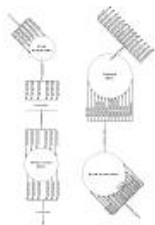
مستقطع من أحد المشروعات (تخطيط النظم)

مشروع تخطيط النظم يتم في ألمانيا في الغالب عن طريق ال "Model-V"، وقد تم في هذا المشروع استخدام هذه الطريقة

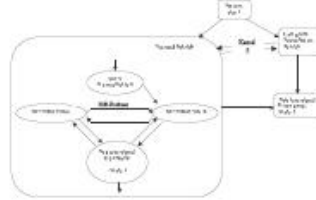


تصميم النظام

تصميم البرمجيات (Software)

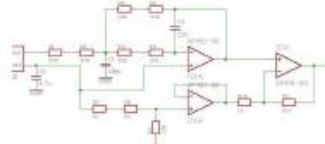


رسم التحليل الهيكلي



رسم الحالة

تصميم المكونات الكهربائية (Hardware)



رسم الدارة الكهربائية



تركيب ال "ICs"
بواسطة EAGLE



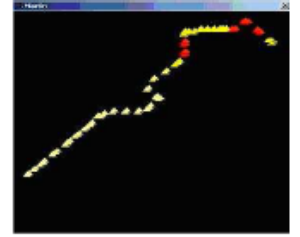
بطاقة معالجة المعلومات
لجهاز الإحساس

أجهزة قياس للطاقة البديلة

في الوقت الراهن يتم تطوير و انتاج جهاز قياس متحرك "المنطاد البديل". بواسطة هذا المنطاد يمكن تركيب أجهزة قياس دقيقة جدًا لقياس شدة الرياح و قوة الطاقة الشمسية في مكان معين. بهذا يمكن تحديد الموضع المثالي لمحطة إنتاج طاقة هوائية أو شمسية. هذا المشروع يتم بالتعاون مع جامعة "كارلس روه" و المدرسة الفنية العليا بكارلس روه و جامعة شتوتغارت.



واجهة مستخدم
لأجهزة الاستشعار



شدة الانبعاث الشمسية
مقابل المكان (Simulation)

هذا المشروع "المنطاد البديل" جاري العمل فيه منذ العام 1999 و قد تم من خلال هذا المشروع إنجاز العديد من مشاريع و رسالات التخرج الجامعية حتى الآن بالإضافة الى توفير العديد من فرص التدريب

3 Introduction

3.1 The LOTTE project and the "alternative Lotte"

Airships are becoming more and more important within the last years. At the "Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktion" at the University of Stuttgart a solar airship was built. In February 1992 the project "Solarluftboot" was initiated. The purpose of this project was to find new materials, new construction methods and a new concept for controlling and navigating an airship run by a solar energy engine.

The "alternative Lotte" – a cooperation project between the universities of Karlsruhe and Stuttgart and the Fachhochschule Karlsruhe - is planned to be an experimental airship which can be controlled directly from the ground (first step of implementation) or (second step of implementation) fly automatically to specified coordinates. The energy supply is going to be conventional batteries in the first step but it is planned to switch to solar energy later. With "alternative Lotte" environmental data are collected during the flight via different sensors which can be mounted on the airship. In the first step the speed of the wind, the temperature and the solar radiation are measured. Apart from that flight data such as acceleration, angular velocity and azimuth angle are measured for navigation purposes.

For the above mentioned Solarluftschiff an alternative flight control system (flight control system - FCS) is developed, which takes into account modern information technology methods.

The "alternative LOTTE" project is lead by the association VaEF e.V. As cooperation main cooperation partners act the Universities of Karlsruhe and Stuttgart. The "alternative LOTTE" project intends to use an airship for taking wind measurement data.

4 Project Management

4.1 Costs

Personal costs: About 3 man years

Material costs: 6000 EUR

Renting rooms, computers etc. 10 000 EUR

The project was undergone through student research works (mostly master theses).

5 Flight Control System of an airship¹

This Flight Control System is based on the IFR Control System of the Lotte airship of the University of Stuttgart / Germany

Lotte-FCS	6
1. Das FCS als Blockschaltbild mit seinen einzelnen Bloecken	6
1.1 Allgemeine Einfuehrung	6
1.1.1 Die flugzeugdynamik - das mathematische Prozessmodell	6
<i>Flugbahngleichungen</i>	6
1.1.2 Randbedingungen zur Flugreglerauslegung	6
<i>Mehrgrößen-Regelstrecke</i>	7
<i>Grosser Parameterbereich</i>	7
<i>Modell-Einschraenkungen</i>	7
<i>Regelungsaufgaben</i>	7
1.1.3 Aufbau von Flugreglungssystemen	8
1.2 Die einzelnen Bloecke des Lotte-FCS	8
1.2.1 CAN/PWM-Schnittstelle	8
1.2.2 Seiten- Hoehen- und querruder	8
1.2.3 Luftschiffdynamik	8
1.2.3.1 <i>Nichtlineare Bewegungsgleichungen</i>	8
1.2.3.1.1 <i>Die Zustandsgleichungen der Luftschiffbewegung</i>	8
1.2.3.1.2 <i>Beschleunigung und Lastvielfaches</i>	8
1.2.3.1.3 <i>DGL fuer die Position (Flugbahn)</i>	8
1.2.3.1.4 <i>Vereinfachte Darstellung einer Anflugbahn</i>	9
1.2.3.1.5 <i>DGL fuer die Drehgeschwindigkeit</i>	9
1.2.3.1.6 <i>DGL fuer die Lagewinkel</i>	9
1.2.3.1.7 <i>Das Gesamtgleichungssystem (Windprozeß, Strömungsmechanik (Triebwerk + Aerodynamik), Kinematik)</i>	9
1.2.3.1.8 <i>Stationäre Flugzustände</i>	10
1.2.3.2 <i>Verhalten des Luftschiffs als Punktmasse</i>	10
1.2.3.3 <i>Linearisierte Zustandsgleichungen</i>	10
1.2.3.4 <i>Analyse des dynamischen Flugzeugverhaltens</i>	10
1.2.3.4.1 <i>Lösung der Zustandsgleichungen</i>	10
<i>Numerische Simulation</i>	11
<i>Numerische Integration der Zustandsgleichungen</i>	11
<i>Lösung der Zustandsgleichung im Laplacebereich</i>	12
1.2.4 <i>Messeinrichtung</i>	12
<i>Ergebnis</i>	13
<i>Bemerkung bezüglich der Softwarearchitektur</i>	13
1.2.5 <i>Sender fuer den Down-link der Messdaten</i>	13
1.2.6 <i>Daempfer</i>	14
2. Konstruktion der dienstorientierten Systemstruktur für das Lotte-FCS	14
2.1 <i>Anforderungen</i>	14
2.2 <i>Die Dienste-Architektur des Lotte-FCS</i>	16
2.3 <i>Die Systemplattformen des Lotte-FCS</i>	17
2.3.1 <i>Die Aufgaben einer Systemplattform</i>	17
2.3.1.1 <i>Funktionalität</i>	17
2.3.1.1.1 <i>Funktionen der Dienstdefinition</i>	17
2.3.1.1.1 <i>Funktionen des Dienstzugangs</i>	18
2.3.1.2 <i>Realisierung</i>	19
2.3.1.3 <i>Problem Echtzeitverhalten</i>	20
2.3.2 <i>Die Systemplattform des ifrsenso</i>	20
2.3.2.1 <i>The OSA+ architecture</i>	21
2.3.2.1.1 <i>Introduction</i>	21

¹Author> Samir Mourad

2.3.2.1.2. The Components of OSA+	21
2.3.2.1.2.1 The platform.....	21
2.3.2.1.2.2 The base services (Basisdienste)	22
2.3.2.1.2.3 Extending services (Erweiterungsdienste)	22
2.3.2.1.3. Description of the interfaces.....	22
2.3.2.1.3.1 The interface of the order administration (Auftragsverwaltung).....	22
2.3.2.1.3.1.1 The interface of the order object (Auftragsobjekt).....	22
2.3.2.1.3.1.2 The interface of the tray object and of the connection	22
2.3.2.1.3.1.3 The interface of the order factory and the order scheduler	23
2.3.2.1.3.2 The interface of the service administration.....	23
2.3.2.1.3.3 The interfaces of the base services (Basisdienste).....	24
2.3.2.1.4. Communication.....	24
2.3.2.1.4.1 Overview.....	24
2.3.2.1.4.2 The communication layers	24
2.3.2.1.4.3 The interaction between the layers.....	25
2.3.2.1.4.4 Interaction between the instances	26
2.3.2.1.5. Remarks on the implementation	26
2.3.2.1.6. Static orders.....	26
2.3.2.1.6.1 Quality of Service.....	26
2.3.2.1.6.2 Constructor and Destructor.....	26
2.3.2.1.6.3 Call interface	27
2.3.2.1.7 Initialization.....	27
2.3.2.2 The real-time operating system VxWorks.....	27
2.3.2.2.1 Overview of the role of VxWorks in the development of real-time applications..	27
2.3.2.2.1 Overview of VxWorks facilities.....	27
2.3.2.2.2 The Basic OS of VxWorks.....	29
2.3.2.2.2.1 Special Wind Features and POSIX Features	29
2.3.2.2.2.2 Tasks.....	30
2.3.2.2.2.2.1 Multitasking	30
2.3.2.3 The adaptation VxWorks and OSA+	30
2.3.2.4 The adaptation of OSA+ and the communication system (RS232, radio ethernet) .	30
2.3.3 Die Systemplattform des Laptops am Boden	30
2.4 The measurement and actuating services (Meß- und Steurdienst, MS service) of the Lotte-FCS	31
2.4.1 The general measurement service (GMS)	31
2.4.2 IMUS	32
2.4.3 GPSS	32
2.4.4 DPMS	32
2.4.5 The barometric hight measurement service (BHMS)	32
2.4.6 RSMS (residually sensors measurement unit)	32
2.5 Die Mensch-Maschine-Dienste des Lotte-FCS	32
2.6 Datenhaltungs-Dienste des Lotte-FCS	32
2.7 Dienste-Architektur des Lotte-FCS	32
Literatur	33

Lotte-FCS

Es soll ein FCS für das Luftschiff Lotte entwickelt werden, welches ein objektorientierte Softwarestruktur hat. Durch die OO-Struktur werden die einzelnen Komponenten des FCS derart gekapselt, so daß später leicht eine Komponente z.B. zu Modernisierungszwecken oder bei Anforderungsänderung im Verlaufe des DFG-Projektes ersetzt werden kann. Dies wird wohl vor allem die Hardwarekomponenten betreffen, die in das FCS eingebunden sind.

Das Ziel ist also, ein FCS zu entwickeln, deren Soft- und Hardwarekomponenten möglichst lose Bindungen haben. In Abb. 1 ist der Regelkreis des Lotte-FCS zu sehen. Die drei verschiedenen Betriebsmodi sind dort eingetragen. Bei Modus 1 greift die Benutzerschnittstelle direkt auf die Stellglieder zu. Bei Modus 2 ist zwischen den Piloten (Benutzerschnittstelle) und den Stellgliedern noch ein Dämpfregelungskreis geschaltet. Bei Modus 3 gibt der Benutzer nur Punkte an, die bei Test-bzw. Meßflügen durchfliegen werden sollen. Die Lenkung wird vom System generiert.

Bei dem zu entwickelnden objektorientierten Flight Control System fuer das Luftschiff soll folgendermassen vorgegangen werden:

1. Zunaechst ist das vollstaendige Flugregelungssystem des Luftschiffs als Blockschaltbild aufzustellen.
2. Die einzelnen Bloecke stellen die einzelnen Module bzw. Dienste dar, welche wiederum aus einem oder mehreren Diensten bestehen können. Ein Dienst besteht aus einem Objekt einer Klasse. Innerhalb der Klassenmethoden sind dann gegebenenfalls die Befehle implementiert, die die physikalischen Teile ansteuern wie z.B. die Ruder. Der Entwurf der Klassenarchitektur soll unter Beruecksichtigung sog. Entwurfsmuster (siehe z.B. [1]) erfolgen. Dadurch soll eine bessere Transparenz erreicht und eine bessere Moeglichkeit zu lokalen Aenderungen gegeben werden.

Die folgende Entwurfsbeschreibung enthaelt im wesentlichen 2 grosse Teile.

Der erste Teil beschreibt die einzelnen Bloecke des Blockschaltbildes des Regelungssystems mit ihren Ein- und Ausgaengen. Es wird auf die Flugregelungs- und die Messtechnik eingegangen.

Der zweite Teil befasst sich mit der Beschreibung der Softwarearchitektur, wobei kurze Einfuehrungen in die verwendeten softwaretechnischen Methoden gegeben werden.

Das Ergebnis ist ein verteiltes, modulares und objektorientiertes Systemmodell

1. Das FCS als Blockschaftbild mit seinen einzelnen Bloecken

In Abb. 1 ist das Lotte-FCS als Blockschaftbild angegeben.

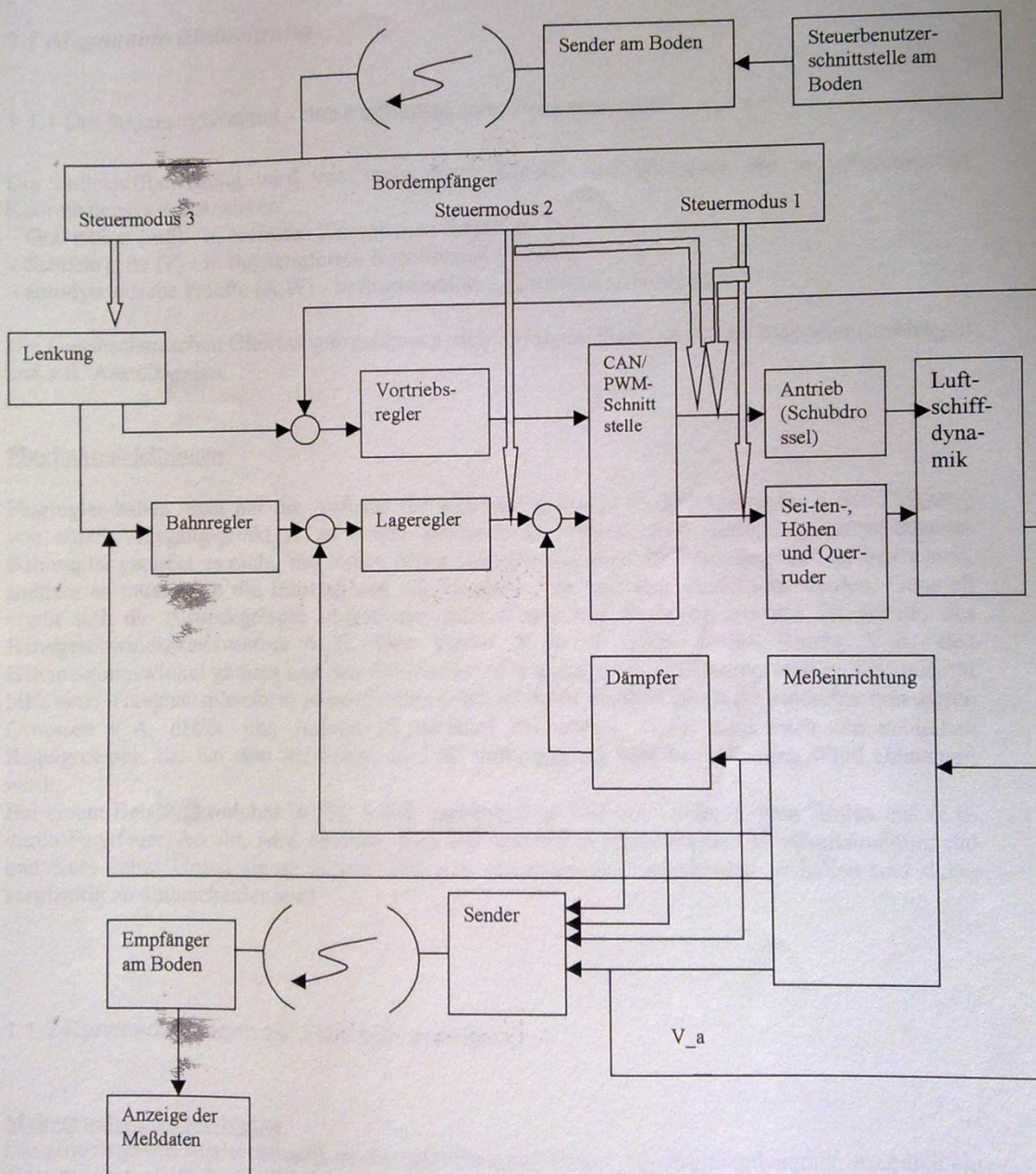


Abb. 1: Das Lotte-FCS als Blockschaftbild

Um einen Einblick in die Funktionsweise des Lotte-FCS zu gewinnen, wird eine kurze allgemeingehaltene Beschreibung der Funktionsweise des Gesamtsystems und teilweise der einzelnen Subsysteme gegeben.
Der Inhalt dieser Einführung ist zum grossen Teil [2] entnommen.

1.1 Allgemeine Einführung

1.1.1 Die Flugzeugdynamik - das mathematische Prozessmodell

Die Luftschiffbewegung wird verursacht durch Kräfte und Momente, die in verschiedenen Koordinatensystemen wirken:

- Gravitation (mg) - in erdfesten Koordinaten (Index g)
- Schubkräfte (F) - in flugzeugfesten Koordinaten (Index f)
- aerodynamische Kräfte (A, W) - in flugwindfesten Koordinaten (Index a)

Die Flugmechanischen Gleichungen geben u. a. den Zusammenhang zwischen Klappenanschlägen und z.B. Anstellwinkel.

Flugbahngleichungen

Flugregler haben nicht nur die Aufgabe der Lagestabilisierung, sondern sollen auch das Flugzeug von einem Ausgangspunkt A zu einem Zielpunkt B führen. Zur Auslegung entsprechender Bahnregler genügt es nicht, nur kleine Abweichungen von einer Referenzflugbahn zu betrachten, sondern es muss auch die Entwicklung der Flugbahn "im Grossen" modelliert werden. Generell ergibt sich die zurückgelegte Flugstrecke nach Länge und Richtung aus der Integration des Bahngeschwindigkeitsvektors V_K . Der Vektor V_K ist durch seinen Betrag V_K , den Bahnneigungswinkel γ und den Bahnazimut ξ gegeben. Alle drei Komponenten sind nur mit Hilfe einer Tragheitsplattform zu bestimmen und sind daher vielfach durch die einfacher messbaren Grössen v_A , dH/dt , und Azimut (Steuerkurs) ψ ersetzt, diese sind auch die üblichen Regelgrössen. Das hat aber zur Folge, dass die Bahnregelung vom herrschenden Wind abhängig wird.

Bei einem Beispiel, welches in [2], S.26f. angegeben ist und auf Lenkung vom Boden aus (z.B. durch Funkfeuer) beruht, wird deutlich, dass Bahnazimut ξ einerseits und Horizontalpeilung ρ und Kurs über Grund σ andererseits sich grundsätzlich verschieden verhalten und daher sorgfältig zu unterscheiden sind.

1.1.2 Randbedingungen zur Flugreglerauslegung

Mehrgrossen-Regelstrecke

Die Grundlage des Reglerentwurfs ist ein axiomatisches Modell der Flugzeugdynamik, nämlich ein nichtlineares Differentialgleichungssystem, dessen wesentliche Koeffizienten sich mit

ausreichender Genauigkeit aus Rechnung und Modellversuch vorherbestimmen lassen. Auf Grund dessen können Flugregler schon vor dem Erstflug eines neuen /flugzeugs fertiggestellt werden.

Grosser Parameterbereich

Der sehr grosse Einsatzbereich (Höhe) und die starken Konfigurationsänderungen (Schwerpunktlage) haben zur Folge, dass die Parameter des Gleichungssystems über grosse Werte schwanken. Insbesondere die aerodynamischen Kennwerte sind stark nichtlineare, mehrdimensionale Funktionen. Dem stehen ausgedehnte Flugphasen gegenüber, in denen sich der Flugzustand nur sehr langsam ändert (z.B. Reiseflug). Auch die heutigen Anflugverfahren setzen sich aus stationären Flugabschnitten zusammen. Deshalb ist es vielfach möglich, die Flugzeugdynamik um einen Betriebspunkt zu linearisieren und durch Näherungsansätze weiter zu vereinfachen.

Modell-Einschränkungen

Die Flugzeugzelle (Flügel, Rumpf, Leitwerk) ist auf Grund ihrer Leichtbaueigenschaften elastisch und in vielen Freiheitsgraden schwingungsfähig. Es besteht zudem eine starke Wechselwirkung zwischen diesen elastischen Freiheitsgraden und den sie anregenden Luftkräften (Aeroelastik). Auch bei den aerodynamischen Rudern besteht eine starke Rückwirkung der Luftkräfte auf den Ausschlag selbst und auf die keineswegs starre Aufhängung. Schliesslich werden durch Elastizität und Laxe zwischen Stellmotoren und Rudern wirkenden Gestänges nichtlineare Einflüsse im Regelkreis wirksam. Wird für den rechnerischen Reglerentwurf das Flugzeug als starr angenommen, so bedeutet das eine Näherung gegenüber dem wirklichen Verhalten. In vielen Fällen ist der Abstand zwischen den Eigenfrequenzen der Strukturschwingungen und denen des starren Flugzeugs gross genug, so dass beide weitgehend voneinander entkoppelt sind und mit Hilfe von Filtern eine Anregung der elastischen Freiheitsgrade durch den Flugregler vermieden werden kann. Alle diese Einflüsse verbieten aber hohe Reglerverstärkungen und hohe Stellaktivität.

Regelungsaufgaben

Die Flugregelung erfüllt gleichermaßen Aufgaben der Flugmechanik und der Flugführung. Erstere betreffen hauptsächlich die Verbesserung der Flugeigenschaften, letztere hat die Aufgabe, einen Flug von A nach B in allen seinen Flugabschnitten bei grösster Sicherheit und Wirtschaftlichkeit zu ermöglichen.

1.1.3 Aufbau von Flugregelungssystemen

Ein heutiges Flugregelungssystem besteht aus der Zusammenfassung einer grossen Zahl von Reglerfunktionen, die aufeinander abgestimmt sind und die vom Piloten in wechselnder Kombination eingesetzt werden können. Es ist eng mit dem Anzeigesystem verknüpft, benutzt die selbe Messinformation und ist mit seiner Hilfe vom Piloten zu überwachen.

Beim Lotte-FCS wird die Messinformation zum Boden gelinkt und dort angezeigt.

In modernen ist generell folgende Arbeitsteilung zwischen Pilot und Flugregler wählbar:

1. Manueller Flug mit teilweiser Reglerunterstützung,
2. Flight-Director-Betrieb, d.h. manueller Flug, unterstützt durch Kommandoanzeigen, die auf Grund reglerähnlicher Funktionen dem Piloten die Bahnführung erleichtern.
3. Vorgaberegung: Bahn-Sollwertvorgabe über die primären Bedienelemente mit automatischer Regelung aller untergeordneter Funktionen
4. Vollautomatischer Flug: automatische Bahnführung einschliesslich aller untergeordneter Regelungsaufgaben ohne Beteiligung des Piloten.

Beim Lotte-FCS gibt es drei moegliche Modi:

1. manueller Flug ohne Regler, d.h. die Stellglieder werden direkt angesprochen
2. manueller Flug mit Daempfer
3. vollautomatischer Flug

1.2 Die einzelnen Bloecke des Lotte-FCS

1.2.1 CAN/PWM-Schnittstelle

1.2.2 Seiten- Hoehen- und querruder

1.2.3 Luftschiffdynamik

1.2.3.1 Nichtlineare Bewegungsgleichungen

In [2], Anhang A.1.4 ist ein in sich geschlossenes gleichungssystem zusammengestellt, welches sich auch als Grundlage fuer eine numerische Simulation der Luftschiffbewegung eignet.

1.2.3.1.1 Die Zustandsgleichungen der Luftschiffbewegung

Die in 6 Freiheitsgraden ablaufende Bewegung des starren Flugzeugs wird durch vier Vektordifferentialgleichungen beschrieben. Die erste dieser Gleichungen folgt aus dem Impulssatz und beschreibt das Kräftegleichgewicht im Flugzeugschwerpunkt. In [2], S.165 ff. wird aus der DGL für die Translationsgeschwindigkeit die Translationsgeschwindigkeit in verschiedenen Koord.systemen gewonnen.

1.2.3.1.2 Beschleunigung und Lastvielfaches

Deutlich vom Beschleunigungsvektor in flugzeugfesten Koordinaten ist der Vektor **b** der von den Beschleunigungsmessern im Flugzeug angezeigten Signale (vgl. [2], Kap. 9.3.1). Anstatt des von den Beschleunigungsmessern im Flugzeug gemessenen Vektors **b** wird haeufig das Lastvielfache verwendet. Dieses ist Regelgroesse in vielen modernen Flugregelungssystemen. Das Lastvielfache ist ein Mass fuer die vom Piloten empfundene Reaktionskraft und steigt an bei Erhoehung der Last, also bei Auftriebserhoehung. Es wird meist in Gegenrichtung zu **b** als n_z angegeben. Def. nach /DIN 9300/, welche nur den aerodynamischen Auftrieb beruecksichtigt:

$$n_z = -Z_a/G = (A_0 + \Delta A)/G = 1 + \Delta n_z$$

Diese Groesse ist nur bei kleinen Winkeln und kleinen Abweichungen vom Horizontalflug zu verwenden.

1.2.3.1.3 DGL fuer die Position (Flugbahn)

Die Flugposition erhaelt man durch die Integration der Translationsgeschwindigkeit relativ zu einem Startpunkt A (siehe die DGL [2], Gl. 5.2.17).

Die Flugzeugposition kann durch die Nord- und Ostkoordinaten Delta N und Delta E und die Flughoehe H ausgedrueckt werden. Die Relativposition zu einem Zielpunkt, ausgedrueckt in Schraegentfernung R, Elavationspeilung epsilon und Horizontalpeilung rho, lautet nach [2], Kap.2.4: (siehe [2], Gl. 2.4.14)

Die DGL fuer die Ablage von einer Sollflugbahn (VOR oder ILS-Leitstrahl): siehe [2], Gl. 5.2.18

1.2.3.1.4 Vereinfachte Darstellung einer Anflugbahn

Zwei Standardsituationen der Kurzstreckenavigation sollen jeweils in einer Ebene naeher erlaeutert werden. Es wird dabei angenommen, dass $V_K = V_{Kc}$ (Sollkursgeschwindigkeit) ist.

Horizontalflug in richtung auf einen Wegpunkt

siehe [2], Bild 5.2 (S. 169)

Sinkflug auf einen Landepunkt

siehe [2], Bild 5.3 (S. 170)

1.2.3.1.5 DGL fuer die Drehgeschwindigkeit

Die DGL fuer die Drehgeschwindigkeit folgt aus dem Drehimpulssatz. Die Gleichung (siehe [2], Gl. 5.2.26) beschreibt das Momentengleichgewicht um den Schwerpunkt. Tragheitstensor des Flugzeugs multipliziert mit der Ableitung des Drehgeschwindigkeitsvektors nach der Zeit ist gleich der Summe aus dem resultierenden aerodynamischen Momentenvektor und dem resultierenden Momentenvektor des Triebwerkschubes. Der Traegheitstensor des Flugzeugs wird als konstant angenommen und lautet in flugzeugfesten Achsen: siehe [2], Gl. 5.2.27.

1.2.3.1.6 DGL fuer die Lagewinkel

Die Integration der Drehgeschwindigkeitskomponenten liefert die Zustandsvariablen der Flugzeuglage (Eulerwinkel): siehe [2], Gl. 2.3.14.

1.2.3.1.7 Das Gesamtgleichungssystem (Windprozeß, Strömungsmechanik (Triebwerk + Aerodynamik), Kinematik)

Blockschaltbild der Zustandsgleichungen

Die Vektor-DGLen [2], Gl. 5.2.1, 5.2.17, 5.2.27 und 2.3.14 sind einschließlic notwendigiger Transformationen in [2], Bild 5.4 als Blockschaltbild dargestellt. Eingangsgroessen sind die Kraft- und Momentenvektoren. Ausgangsgroessen sind die 12 Zustandsgroessen der Flugzeugbewegung, d.h. die Komponenten der Vektoren V_K , Ω_K , s und Φ . Alle vier Zustandsvektoren wirken auf die uebrigen Gleichungen einschliesslic der Windgleichungen zurueck.

Kopplung zwischen Wind und Flugzeugbewegung

In [2], Bild 5.6 sind folgende Blockschaltbilder zusammengefasst:

1. Reduziertes Blockschaltbild von Triebwerk und Aerodynamik (Stoemungsmechanik)
2. Zustands-DGLen der Flugzeugbewegung (Kinematik)
3. Blockschaltbild des Windeinflusses auf das Flugzeug (Windprozess)

Gueltingkeitsbereich des Gleichungssystems

Der Gueltingkeitsbereich dieses Gleichungssystems ist durch die Annahmen am Beginn von Kap.5 aus [2] eingeschraenkt.

1.2.3.1.8 Stationäre Flugzustände

Die sehr aufwendigen nichtlin. Gleichungen, wie sie in [2], Kap. 5.2 zusammengestellt wurden, sind nur unter einschraenkenden Annahmen fuer sonderfaelle analytisch loesbar, vgl. [2], Kap. 6. sie koennen in allgemeiner Form nur numerisch, d.h. iterativ geloest werden. Sie sind als Grundlage fuer eine numerische Simulation im Anhang A.1.4 nochmals zusammengestellt. Als Anfangsbedingung fuer eine solche Simulation ist aber zunaechst ein stationaerer Gleichgewichtszustand festzulegen. Das entspricht genau der Vorgehensweise bei flugversuchen: vor jedem Versuch ist vom Piloten ein ausgetrimmter Zustand herbeizufuehren. Ein solches stationaeres Gleichgewicht einschliesslich der zugehoerigen Parameter ist auch der Ausgangspunkt fuer eine Linearisierung der Gleichungen, wie sie in [2], kap.7 durchgefuehrt wird. In [2], Kap. 5.4 werden der symmetrische Geradeausflug und der horizontale, koordinierte Kurvenflug betrachtet.

1.2.3.2 Verhalten des Luftschiffs als Punktmasse

Reduzierte Differentialgleichungen - Bahnbewegung in drei Freiheitsgraden

Reduzierte Differentialgleichungen - Bahnbewegung in zwei Freiheitsgraden

Energiebetrachtungen

Reaktion des Luftschiffs auf Stellkommandos - Veränderung des aerodynamischen Arbeitspunktes

Reaktion des Luftschiffs auf Stellkommandos - Reaktion des luftschiffs auf Schubänderung

Reaktion des Luftschiffs im Windfeld

1.2.3.3 Linearisierte Zustandsgleichungen

1.2.3.4 Analyse des dynamischen Flugzeugverhaltens

In diesem Abschnitt soll das Flugzeugverhalten "im Kleinen", d.h. bei kleinen Abweichungen von einem stationären Arbeitspunkt analysiert werden. Dafür werden sowohl die Zustandsgleichungen (siehe [2], Kap. 7) selbst herangezogen, als auch die aus ihnen abgeleiteten Übertragungsfunktionen. Zur Vereinfachung werden die linearen Abweichungen (δ) der Bewegungsgrößen nicht gesondert gekennzeichnet.

Bei dieser Analyse steht im Vordergrund erstens die Charakterisierung des dynamischen Verhaltens der Strecke, d.h. ihres Verhaltens mit und ohne äußere Anregungen, zweitens die Einführung der in der Flugregelung gebräuchlichen Näherungen zur Beschreibung einzelner Freiheitsgrade und zur Abschätzung wesentlicher Einflüsse und Eigenschaften und drittens die Auswahl geeigneter Übertragungswege für die Regelung (Regelbarkeitsbetrachtungen). Diese Analyse bildet die Grundlage für die Entwicklung von Reglerarchitekturen.

1.2.3.4.1 Lösung der Zustandsgleichungen

Das mathematische modell der Flugzeugbewegung wurde in [2], Kap.7 auf folgende Zustandsform gebracht:

$$\begin{aligned} \dot{\underline{x}}(t) &= \underline{A}^* \underline{x}(t) + \underline{B}^* \underline{u}(t) \\ \underline{y}(t) &= \underline{C}^* \underline{x}(t) + \underline{D}^* \underline{u}(t) \end{aligned}$$

Hierin hat \underline{x} die Dimension n , \underline{u} die Dimension p und \underline{y} die dimension q . Die Matrizen haben entsprechende Dimensionen.

Die allgemeine Lösung der ersten Gleichung (siehe [2], Gl.8.1.18) setzt sich aus der Summe der homogenen Lösung und der Partikulärlösung zusammen.

Die homogene Lösung $\underline{x}_h(t)$ setzt sich aus n Einzellösungen zusammen. Sie enthält :

1. Reelle Bewegungsanteile, in denen der Eigenwert $\lambda_{i,j}$ und der zugehörige Eigenvektor \underline{v}_i reell sind. Hierdurch wird ein aperiodischer Vorgang beschrieben.
2. Konjugiert komplexe Anteile. Paarweise wird hierdurch ein periodischer Vorgang beschrieben.

Ist $\underline{u}(t)$ gleich einem Vektor von Sprungfunktionen mit dem Amplitudenvektor \underline{u}_0 , so lautet die Partikulärlösung: siehe (Gl. 8.1.20). Auch diese n Lösungsanteile sind eine Funktion der Eigenvektoren. Ihre relativen Amplituden sind außerdem umgekehrt proportional den Eigenwerten. Das läßt sich an den Sprungantworten von Anstellwinkelschwingung und Phygoide in den Bildern [2], 8.11 und [2], 8.12 nachvollziehen.

Numerische Simulation

Eine geschlossene Lösung des Faltungsintegrals in Gl.8.1.13 ist nur in einfachen Fällen möglich. Zur numerischen Berechnung der Systemantwort genügt aber eine in genügend dicht gewählten Abtastschritten mit dem konstanten Abtastintervall T . Dazu wird die kontinuierliche Lösung $\underline{x}(t)$ als eine Wertefolge $\underline{x}(t_k)$ dargestellt, vgl. [2], Bild 8.1.

Die allgemeine Lösung für $\underline{x}(t)$ wird diskretisiert und in rekursive Form geschrieben ([2], Gl.8.1.21). Schließlich folgt die diskrete Zustandsgleichung des Systems:

$$\underline{x}(t_{k+1}) = \underline{F} \cdot \underline{x}(t_k) + \underline{G} \cdot \underline{u}(t_k) \quad ([2], \text{Gl.8.1.22})$$

\underline{F} und \underline{G} sind konstante Matrizen:

$$\underline{F} = \text{Summe von } n \text{ü} = 0 \text{ bis Unendlich } ((\underline{A} \cdot T)^{n \text{ü}} / \text{fakultät}(n \text{ü}))$$

$$\underline{G} = \text{Summe von } n \text{ü} = 0 \text{ bis Unendlich } ((\underline{A} \text{ hoch } n \text{ü}) \cdot (T \text{ hoch } (n \text{ü} + 1)) \cdot \underline{B} \cdot (1 / \text{fakultät}(n \text{ü} + 1)))$$

Hiermit ergibt sich die Möglichkeit der numerischen Simulation, d.h. der diskreten Berechnung der Antwort selbst komplexer Gleichungssysteme auf beliebige Eingangssignale $\underline{u}(t_k)$. Bei genügend großer Schrittzahl läßt sich so ein quasi-kontinuierlicher zeitverlauf berechnen und als Kurvenverlauf darstellen. Die Vorgehensweise ist dabei wie folgt:

- Festlegung der Abtastzeit T (Shannonsches Abtasttheorem beachten!)
- Einmalige Berechnung der konstanten Matrizen \underline{F} und \underline{G}
- Vorgabe des Eingangssignal-Signalverlaufs $\underline{u}(t_k)$, wobei dieser im Abtastintervall T als konstant anzunehmen ist.
- Schrittweise, rekursive Berechnung von Zustands- und Ausgangsvektor nach den diskreten Zustandsgleichungen

$$\underline{x}(t_{k+1}) = \underline{F} \cdot \underline{x}(t_k) + \underline{G} \cdot \underline{u}(t_k)$$

$$\underline{y}(t_k) = \underline{C} \cdot \underline{x}(t_k) + \underline{D} \cdot \underline{u}(t_k)$$

Der Vorteil dieser Lösung besteht darin, daß bei fester Abtastzeit T auch \underline{F} und \underline{G} fest sind. Die Zustandsgleichung enthält somit nur noch Multiplikationen und läßt sich relativ schnell berechnen. Nachteilig ist, daß sie nur auf lineare Zustandsgleichungen anwendbar ist.

Numerische Integration der Zustandsgleichungen

Unter dem Begriff "numerische Integration" wird die direkte, diskrete Lösung der Zustandsgleichung

$$d\underline{x}(t)/dt = \underline{f}[\underline{x}(t), \underline{u}(t)], \quad \underline{x}(t_0) = \underline{x}_0$$

mit Hilfe verschiedener Integrationsalgorithmen verstanden. Das bekannteste ist das Runge-Kutta-Verfahren (siehe [3]). Ein Digitalprogramm zur numerischen Simulation hat den in [2], Bild 8.3 gezeigten prinzipiellen Aufbau.

Lösung der Zustandsgleichung im Laplacebereich

1.2.3.4.2 Das Eigenverhalten des Luftschiffs

1.2.3.4.3 Das Übertragungsverhalten des Luftschiffs

1.2.3.4.4 Näherungsansätze - Näherung für die Anstellerschwingung

1.2.3.4.5 Näherungsansätze - Näherung für die Phygoidebewegung

1.2.3.4.6 Näherungsansätze - Näherung für die Seitenbewegung

1.2.3.4.7 Regelbarkeitsbetrachtungen

1.2.3.4.8 Regelbarkeitsbetrachtungen - Stellverhalten in der Längsbewegung

1.2.3.4.9 Regelbarkeitsbetrachtungen - Allpaßverhalten

1.2.3.4.10 Regelbarkeitsbetrachtungen - Stellverhalten in der Seitenbewegung

1.2.3.4.11 Regelbarkeitsbetrachtungen - Reaktion des Luftschiffs auf Störsignale

1.2.4 Messeinrichtung

Die Messkette von der Detektion einer physikalischen Grösse bis zur Verwendung des Messwertes im Flugregelungssystem oder in der Messwertanzeige lässt sich in fünf Abschnitte aufteilen. Trotz einiger Überschneidungen sind diese Abschnitte charakterisiert durch unterschiedliche Aufgabenstellungen, verschiedene Technologien und typische Fehlerquellen.

1. **Messung:** physikalischer Effekt, mit der eine Grösse möglichst fehlerfrei messbar ist
2. **Wandlung:** Der physikalische Effekt (z.B. Druck) ist zu wandeln in einen Ausschlag (Druckmessdose) und evtl. in eine digitale Zahl. Diese Wandlung vollzieht sich häufig in mehreren Stufen (mechanisch - elektrisch - analog - digital) und enthält, soweit wie möglich, eine Kompensation von Nichtlinearitäten und Fehlereinflüssen (z.B. Temperatur) sowie eine Umrechnung in die gewünschte Messgrösse. Messung und Wandlung stellen eine Steuerkette dar, bei der alle Fehler voll wirksam werden. Diese sind nur bei sorgfältiger Kalibrierung auf ein erträgliches Mass zu reduzieren.
3. **Übertragung:** Der Messwert ist vom Messort zum Regelungs-, Navigations- oder Anzeigesystem zu übertragen und dort evtl. zu speichern. Beim Lotte-FCS wird der CAN-Bus und (zur Übertragung vom IMU-Rechner (iMar) zum Regelungsrechner) die serielle Schnittstelle (RS 232) hierzu verwendet.
4. **Konsolidierung:** Der Messwert wird vor der Weiterverarbeitung konsolidiert, d.h. auf seine Gültigkeit hin überprüft, von kurzzeitigen Ausreissern befreit und zur Unterdrückung fehlerhafter Gleichanteile (bias) und Messrauschen (noise) gefiltert. Dies geschieht beim Lotte-FCS für die IMU-Messwerte im iMar-Rechner, für die übrigen Messwerte im ifrsenso.
5. **Verarbeitung:** Falls nicht schon im Messgeber selbst geschehen, wird der Messwert in die gewünschte flugmechanische Grösse umgerechnet, evtl. durch Verwendung weiterer Messgrößen (Beispiel; Machzahlbestimmung aus Staudruck, statischem Druck und Temperatur).

Fuer ausfuehrliche Informationen ueber diese 5 Messkettenglieder siehe das Literaturverzeichnis zu diesem Thema in [2]. Gemäß [2] sind besonders wichtig fuer flugtechnische Anwendungen die AGARDographen und die AGARD-Conference-Proceedings.

Die grosse Zahl von Messverfahren wird hier wie folgt geordnet:

1. **stroemungsmechanische Groessen** (V_A , angle of attack (Anstellwinkel) α , Schiebewinkel β , Hoehe ueber Meeresspiegel H , dH/dt)
2. **Inertialgroessen** (Lagewinkel Φ , Drehratenvektor (Drehgeschwindigkeitsvektor) Ω , Translationsbeschleunigung a , V_K , s)
3. **Azimumtmessung** (Ψ , Erdmagnetfeld)
4. **funktechnische Messverfahren** (H , V_K , R , ϵ , ρ)
5. **GPS** (3D-Positionsbestimmung auf der Erde (zivile Genauigkeit: besser als 100m))
6. **Bildverarbeitung** (lokale Orientierung)

Ein **Kalmanfilter** kann zur Messwertkonsolidierung eingesetzt werden.

Die Messeinrichtung des Lotte-FCS besteht aus folgenden Komponenten:

- aus 1.
 - Druckdifferenzmesser (Vierflochsonde): liefert α , β und V_a
 - barometrischer-Hoehemesser: liefert H
- aus 2.
 - IMU (inertial measurement unit) mit Beschleunigungsmessern und Gyroskopen: liefert alle Inertialgroessen (s.o.)
- aus 5.
 - GPS: liefert Position (2D+Höhe/Heading + V_{Grund})
- Beim Lotte-FCS wird ein Kalmanfilter zur Messwertkonsolidierung eingesetzt.

Die Daten der IMU werden im iMar-Rechner aufbereitet und ueber eine serielle Schnittstelle auf den ifrsenso-Rechner gegeben, der sie ueber die CAN-Schnittstelle weitergeben kann. Die Rohmeßdaten aller anderen Sensoren kommen ueber CAN-Bus zum ifrsenso werden dort aufbereitet. Die aufbereiteten Daten koennen vom ifrsenso aus ueber den CAN-Bus weitergegeben werden.

Ergebnis

Das Modul bzw. die Klasse "Messeinrichtung" hat folgende Methoden:

```
Messeinrichtung::gib_alpha()
Messeinrichtung::gib_beta()
Messeinrichtung::gib_H()
Messeinrichtung::gib_dH_dt()
...
```

Bemerkung bezüglich der Softwarearchitektur

Ein Objekt der Klasse Messeinrichtung (Singleton) besitzt Objekte, die die einzelnen Sensoren darstellen. Sie delegiert also die Messwernerfassung weiter zu einem virtuellen Sensor, der schliesslich den entsprechenden physikalischen Sensor ansteuert.

Es wird wohl guenstig sein, das Entwurfsmuster "Fassade" zu verwenden.

1.2.5 Sender fuer den Down-link der Messdaten

1.2.6 Daempfer

Im Lotte-FCS sind 3 Daempfer implementiert: Roll-, Nick- und Gierdaempfer.

Die erste Aufgabe von Flugreglern besteht in der Anpassung der Flugeigenschaften an die Wuensche des Piloten. Das betrifft

- die Modifikation des Eigenverhaltens bzgl. der Eigenwerte (Zeitkonstanten, Frequenz, Daempfung) und der Eigenvektoren (Kopplung zwischen den Zustandsgrößen)
-

Die Lagewinkel Teta (Laengsneigung) und Phi (Haengewinkel) stellen eine Abgrenzung der Basis-Reglerfunktionen dar. Diese betreffen alle untergeordneten Regelungsaufgaben unterhalb der Bahnfuehrungsebene. Sie sind eingeschaenkt auf die Regelung der rotatorischen Freiheitsgrade der Starrkoerperbewegung, deren Zeitkonstanten um eine Groessenordnung kleiner sind als die der Bahnfreiheitsgrade. Sie stellen die inneren Regelschleifen des Flugregelungssystems dar, die von den aeusseren Regelschleifen zur Bahnfuehrung naeherungsweise entkoppelt sind, vgl. [2], Kap. 8.4

2. Konstruktion der dienstorientierten Systemstruktur für das Lotte-FCS

Der Inhalt der allgemeinen Beschreibungen des Folgenden ist [4] und [5] entnommen.

Da es sich beim Lotte-FCS um ein Automatisierungssystem handelt, soll es mit Hilfe moderner Konzepte der Automatisierungstechnik realisiert werden.

Bei der Konstruktion von Automatisierungssystemen mit Mikrorechnern wird heute eine verteilte, modulare und objektorientierte Systemstruktur unter Benutzung des Dienstkonzeptes verwandt. Dienste werden meist mittels Software realisiert. Prinzipiell können Dienste auch durch spezielle Hardware erbracht bzw. unterstützt werden.

2.1 Anforderungen

Die Entwicklung heutiger Automatisierungssysteme stellt eine Reihe von Anforderungen an Hard- und Software:

- modularer Aufbau von hard- und Software
- verteilter Aufbau von Hard- und software
- Baukastensysteme (soft- und Hardware-Komponenten, -Busse, -Stecker)
- Echtzeitfähigkeit einzelner Komponenten

Vorteile:

- Wiederverwendbarkeit
- Reduktion von Kosten und Entwicklungszeit
- Verbesserte Wart- und Testbarkeit

Automatisierungssysteme werden aus Komponenten eines Komponentenbaukastens zusammengesetzt.

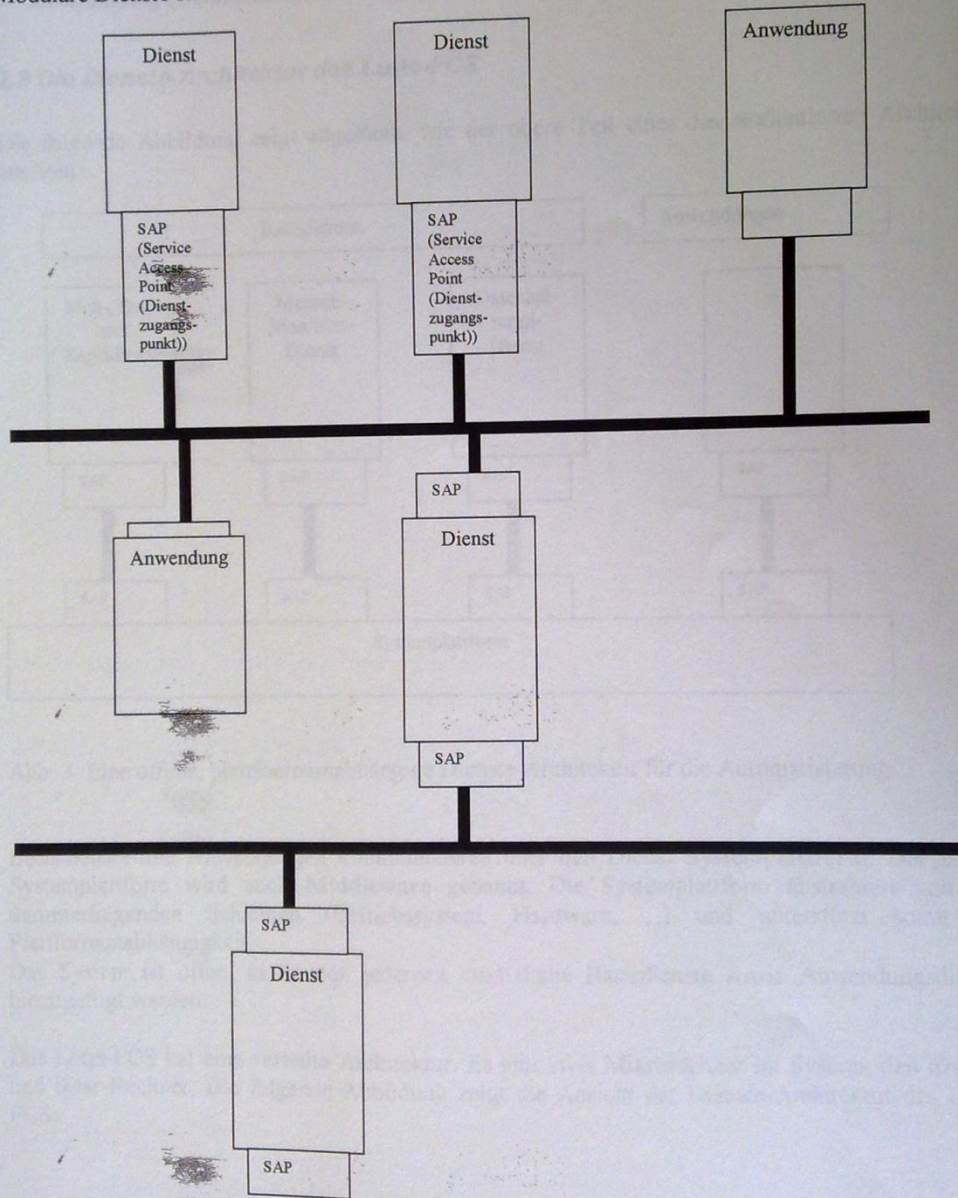
Für Hardware bedeutet das:

- Modulare Mikrorechnersysteme
- Konfigurierbare Hardware-Module
- Genormte Steckkarten und Busse

Für Software:

- Modulare Dienste-Architektur
- Konfigurierbare Dienste
- Genormte dienstzugangspunkte, -protokolle und -primitive (Software-Stecker und -Busse)

Modulare Dienste-Architektur:



Anforderungen an "ideale" Dienste:

- Plattformunabhängigkeit (-> Portierbarkeit)
- Skalierbar und konfigurierbar (-> Anpassung an die Anwendung, Erweiterbarkeit)
- Austauschbar (einfacher Wechsel von Komponenten)
- Kompatibel (-> Datenaustausch)
- Testbar (-> Fehlersuche, Wartung)
- Echtzeitfähig (wenn erforderlich)

2.2 Die Dienste-Architektur des Lotte-FCS

Die folgende Abbildung zeigt allgemein, wie der obere Teil einer dienstorientierten Architektur aussieht.

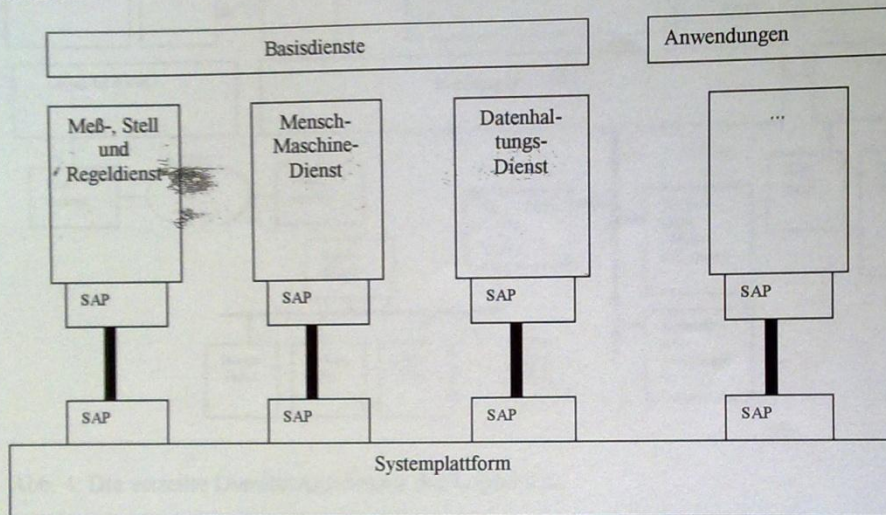


Abb. 3: Eine offene, plattformunabhängige Dienste-Architektur für die Automatisierung

Basisdienste und Anwendungen kommunizieren über den Dienst **Systemplattform**. Der Dienst Systemplattform wird auch **Middleware** genannt. Die Systemplattform abstrahiert von den darunterliegenden Schichten (Betriebssystem, Hardware, ...) und unterstützt somit die Plattformunabhängigkeit.

Das System ist offen, es können jederzeit zusätzliche Basisdienste sowie Anwendungsdienste hinzugefügt werden.

Das Lotte-FCS hat eine verteilte Architektur. Es gibt zwei Mikrorechner im System: den ifrsenso und iMar-Rechner. Die folgende Abbildung zeigt die Ansicht der Dienste-Architektur des Lotte-FCS:

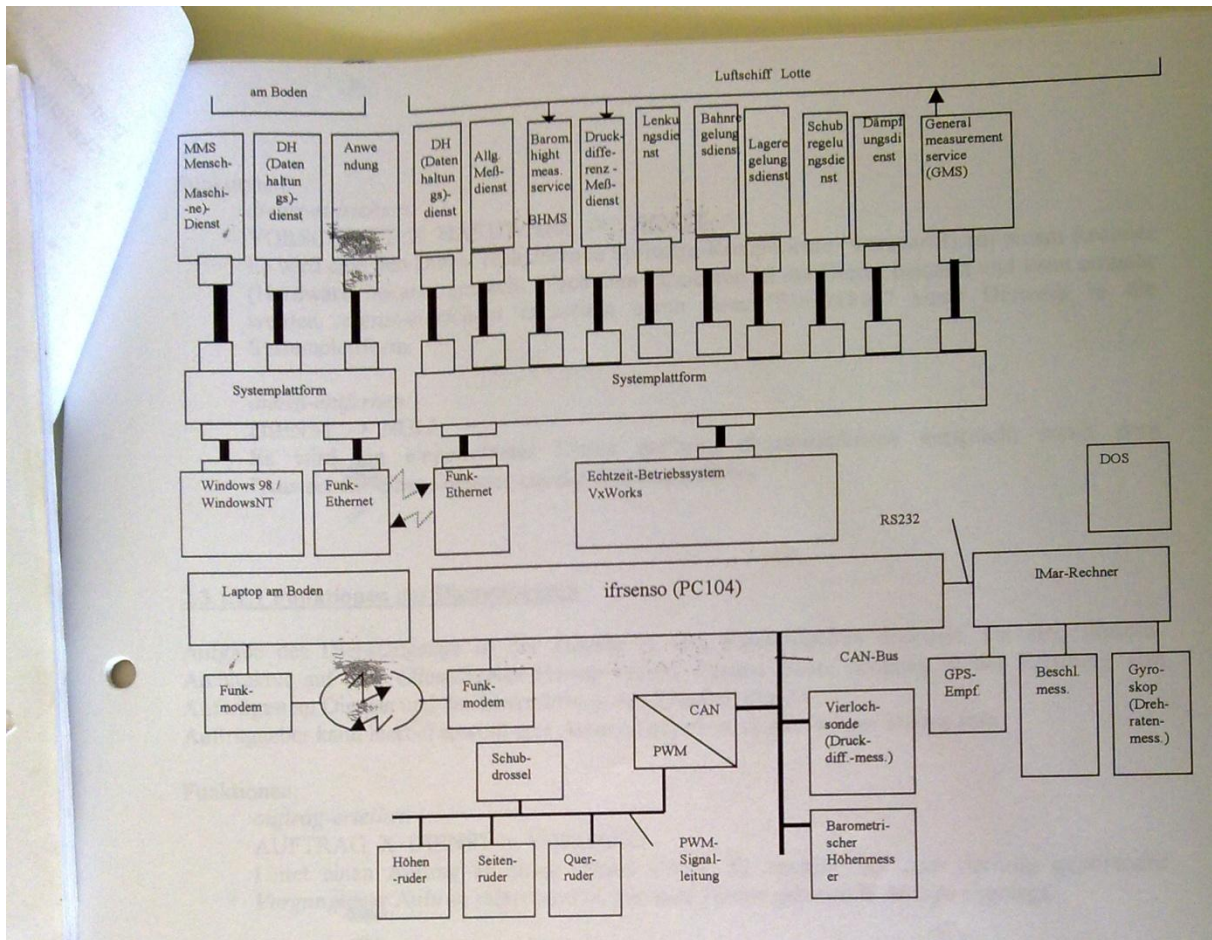


Abb. 4: Die verteilte Dienste-Architektur des Lotte-FCS

2.3 Die Systemplattformen des Lotte-FCS

2.3.1 Die Aufgaben einer Systemplattform

Die Systemplattform verbindet die einzelnen Dienste und Anwendungen untereinander. Eine Systemplattform besitzt im wesentlichen 2 Aufgaben:

- **Dienstdefinition:** Ankoppeln neuer Dienste an die Systemplattform. Es werden hierdurch Dienste im System eingerichtet und bekanntgegeben
- **Dienstzugang:** Zugang zu den angekoppelten Diensten. Es wird die Kommunikation zwischen Anwendungen und Diensten hergestellt, d.h. die SAP der angekoppelten Dienste wird hierdurch zugänglich gemacht.

2.3.1.1 Funktionalität

2.3.1.1.1 Funktionen der Dienstdefinition

Aufgabe des Dienstdefinition ist das Einrichten und Entfernen von Diensten.

Funktionen:

dienst-einrichten:

VORSCHRIFT X HARDWARE -> DIENST

Es wird eine den Dienst realisierende Software-Komponente (Vorschrift) auf einem Rechner (Hardware) bekannt gemacht. Nach dem Einrichten ist der Dienst bekannt und kann erreicht werden. *dienst-einrichten* entspricht somit dem "Einstecken" eines Dienstes in die Systemplattform.

dienst-entfernen:

DIENST -> NULL

Es wird ein eingerichteter Dienst entfernt. *dienst-entfernen* entspricht somit dem "Austecken" eines Dienstes aus der Systemplattform

2.3.1.1.1 Funktionen des Dienstzugangs

Aufgabe des Dienstzugangs ist der Zugang zu den angekoppelten Diensten. Da eine dienst-orientierte Architektur auf dem *Client/Server-Prinzip* basiert, besteht dieser Zugang in der Erteilung von **Aufträgen** an Dienste und der Übermittlung von **Ergebnissen**. Auftraggeber kann hierbei sowohl eine Anwendung als auch ein anderer Dienst sein.

Funktionen:

auftrag-erteilen:

AUFTRAG X DIENST -> VORGANG

Leitet einen Auftrag an einen Dienst weiter. Es entsteht ein zum Auftrag gehörender **Vorgang**. Der Auftrag selbst wird in eine zum Dienst gehörende **Ablage** abgelegt.

auftrag-erwarten:

ABLAGE -> AUFTRAG

Wartet, bis ein Auftrag in der **Ablage des Dienstes** vorhanden ist, holt diesen Auftrag.

existiert-auftrag:

ABLAGE -> WAHRHEITSWERT

Prüft, ob in der **Ablage des Dienstes** ein Auftrag vorhanden ist.

ergebnis-melden:

ERGEBNIS X VORGANG -> ABLAGE

Meldet das Ergebnis eines Vorgangs an den Auftraggeber. Das Ergebnis wird hierbei in eine zum Auftraggeber gehörende Ablage abgelegt.

ergebnis-erwarten:

VORSCHRIFT X ABLAGE -> ERGEBNIS

Wartet, bis ein Ergebnis auf einen Vorgang in der **Ablage des Auftraggebers** vorhanden ist, holt dieses Ergebnis.

existiert-ergebnis:

VORGANG X ABLAGE -> WAHRHEITSWERT

Prüft, ob in der Ablage ein Ergebnis auf einen Vorgang vorhanden ist

2.3.1.2 Realisierung

Die Systemplattform kann entsprechend ihrer Aufgabenbereiche in mehrere Schichten unterteilt werden

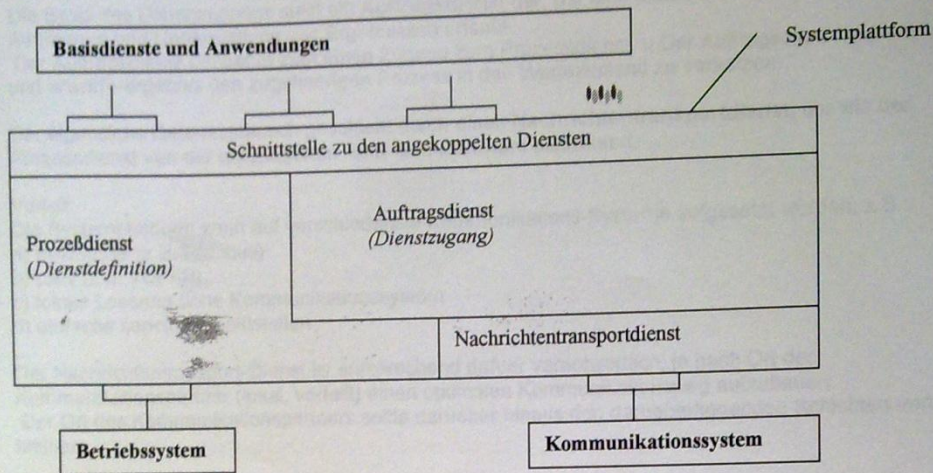


Abb.5: Die Schichten der Systemplattform

Da die Dienste unabhängig voneinander eingerichtet und betrieben werden müssen, entspricht jeder eingerichtete dienst einem oder mehreren Prozessen.

Basis der Dienstdefinition muß somit ein **Prozeßdienst** sein, der über das Betriebssystem die Einrichtung und den Betrieb von Prozessen auf der Zielhardware erlaubt.

Wie die gesamte Systemplattform sollte auch der Prozeßdienst vom unterlagerten Betriebssystem bzw. der unterlagerten Hardware abstrahieren.

Der Vorteil ist, daß die Systemplattform auf verschiedene Hard-/Software-kombinationen aufgesetzt werden kann, z.B.:

- Rechner mit Multitasking-Betriebssystem (z.B. Windows)
- Rechner mit Singletasking Betriebssystem (z.B. MS-DOS)
- Rechner ohne Betriebssystem (z.B. Mikrocontroller, Spezialrechner)

...

Es gibt verschiedene Varianten des Prozeßdienstes innerhalb der Systemplattform

Bei allen Varianten: Prozeßdienst mit plattformunabhängiger Schnittstelle sitzt auf einem Hardware- und Betriebssystemabhängigen Teil.

- Variante a (wenn ein Multitasking-Betriebssystem auf der Hardware sitzt): Hardware- und Betriebssystemabhängiger Teil: Anpassung auf Betriebssystem/-Prozesse
- Variante b (wenn ein Singletasking-Betriebssystem auf der Hardware sitzt): Hardware- und Betriebssystemabhängiger Teil: Eigenes Prozess-System unter Nutzung von Betriebssystem-Funktionen
- Variante c (wenn kein Betriebssystem auf der Hardware sitzt): Hardware- und Betriebssystemabhängiger Teil: Vollständig eigenes Prozess-System

Beim Lotte-FCS handelt es sich bei der Systemplattform des Laptops am Boden um Variante a, beim ifrsenso-Rechner bestehen die Möglichkeiten, entweder Variante a oder b zu benutzen. Die Betrachtung

des iMar-Rechners ist in dieser Beziehung uninteressant, weil dieser als Blackbox mit serieller Schnittstelle zu betrachten ist.

Die Basis des Dienstzugangs stellt ein **Auftragsdienst** dar, der eine plattformunabhängige Erteilung von Aufträgen und Uebermittlung von Ergebnissen erlaubt.
Der Auftragsdienst benötigt zum einen Zugang zum Prozessdienst, u. Der Auftragsdienst *beerwarte-auftrag* und *erwarte-ergebnis* den zugehörigen Prozess in den Wartezustand zu versetzen.

Der eigentliche Datenaustausch geschieht durch einen **Nachrichtentransportdienst**, der wie der Prozessdienst von der unterlagerten Soft- und Hardware abstrahiert.

Vorteil:

Die Systemplattform kann auf verschiedenste Kommunikations-Systeme aufgesetzt werden, z.B.:

- a) Feldbusse (z.B. Profibus)
- b) LAN (z.B. TCP/IP)
- c) lokale Lösung ohne Kommunikationssystem
- d) einfache serielle Schnittstellen

... Der Nachrichtentransport-Dienst ist entsprechend dafür verantwortlich, je nach Ort der Kommunikationspartner (lokal, verteilt) einen optimalen Kommunikationsweg aufzubauen.

Der Ort des Kommunikationspartners sollte darüber hinaus den darüberliegenden Schichten verborgen bleiben.

<Bilder 1 und 3 ([4], Folie iv-21)

2.3.1.3 Problem Echtzeitverhalten

Das Echtzeitverhalten einer Schicht hängt nicht nur von ihr selbst ab, sondern auch in starkem Masse von den darunterliegenden Schichten ab.

Eine noch so schnelle Schicht kann von darunter liegenden langsamen Schichten "ausgebremst" werden.
Vorhersagbarkeit:

Die Einhaltung harter Zeitbedingungen in einer Schicht sind möglich, wenn auch alle benutzten Funktionen der unterlagerten Schicht harte Zeitbedingungen einhalten.

Zur Einhaltung fester Zeitbedingungen müssen die Funktionen der unteren Schichten entweder sehr kurz oder unterbrechbar sein. Eventuelle Änderungen müssen temporär oder in definierter Zeit rückgängig machbar sein.

Weiche Zeitbedingungen stellen die geringsten Anforderungen an die unterlagerten Schichten. Auch hier müssen dann lediglich weiche Zeitbedingungen eingehalten werden.

Für die Systemplattform bedeutet dies:

Selbst wenn Prozess-, Auftrags- und Nachrichtentransportdienst die Einhaltung harter Zeitbedingungen erlauben, hängt das Zeitverhalten der gesamten Systemplattform vom untergelagerten Betriebs- und Kommunikationssystem ab.

2.3.2 Die Systemplattform des ifrsenso

Die Systemplattform des ifrsenso soll auf einem Echtzeitbetriebssystem sitzen, wobei entschieden wurde, VxWorks von WindRiver einzusetzen.

Als Systemplattform wird das OSA+ Programmpaket benutzt, welches am Institut für Mikrorechner und Automation der Uni Karlsruhe entwickelt wurde (siehe [6]). OSA+ liegt in Form einer C-Library vor. Alle Funktionen sind somit aus den Programmiersprachen C und C++ zugänglich.

OSA+ realisiert die in 2.3.1 vorgestellte Funktionalität einer Systemplattform in wesentlichen Teilen.

Ein Auftrag dient hierbei als Mittel zur

- **Kommunikation**
Durch den Inhalt des Auftrags, den der Auftraggeber an einen Dienst weiterleitet. Da die Erteilung eines Auftrages auch eine Antwort erfordert (Auftragsbestätigung, Ergebnis), wird hierdurch implizit eine bidirektionale Kommunikation definiert.
- **Synchronisation**
Durch Erteilung und Erwartung von Aufträgen und Ergebnissen. Durch die Reihenfolge der Auftragserteilung kann zudem die Ausführungsreihenfolge beeinflusst werden
- **Parallelverarbeitung**
Durch gleichzeitige Auftragserteilung, sofern die Beschaffenheit der Hardware dies erlaubt.
- **Definition von Echtzeitanforderungen**
Durch Zuordnung eines Termins zu einem Auftrag

Im folgenden muß OSA+ an VxWorks angepaßt werden.

Aus diesem Grund wird zunächst OSA+ vorgestellt; anschließend wird VxWorks vorgestellt. Danach wird beschrieben, wie OSA+ an VxWorks angepaßt wird.

2.3.2.1. The OSA+ architecture

2.3.2.1.1 Introduction

OSA+ is a scalable realtime middleware. It allows to develop distributed applications, which run in realtime and exist in heterogen networks. The hardware and the operation systems are optimal used for this purpose.

The architecture of OSA+ organizes an application into services, which communicate with each other with orders. The communication is achieved with a platform. Services are "inserted". So all inserted services can communicate by the way of the platform.

The platform itself uses optionally special services to extend its functionality:

- **Base services (Basisdienste)** - to make the platform independent of the OS and the specific HW;
- **extending services (Erweiterungsdienste)**

In the following there will be described the architecture of OSA+ in detail. OSA+ is portable (because of its architecture).

First, the 'naked' architecture is described with the functionality that it offers. Then the base services and the extending services with the additional functionality are described. At last the interfaces of the platform and the of the basis and extending services are described.

2.3.2.1.2. The Components of OSA+

2.3.2.1.2.1 The platform

The platform is the component which is visible to the user of OSA+.

- The platform offers an interface to connect services to the platform.
- The platform allows the application or other services to make orders (Aufträge).

So the platform is divided into 3 basic parts:

1. a **service administration**: knows all inserted services, installs and deinstalls services, finds services.
2. an **order administration**: allows to make orders, to wait for orders, to announce and to wait for results of these orders

3. a **user interface** gives the user a structured interface to the former described two administration instances.

2.3.2.1.2.2 The base services (Basisdienste)

The platform uses the base services to maintain the encapsulation of the OS (operation system) functions, which can be used by the platform. If there are no base services installed in the platform, than the platform can only use the standard of the implementation language (ANSI guide lines for C, C++ and JDK 1.0). If more functionality is to be used it must be offered by the base services. The base services are:

1. **process service** (Prozeßdienst): allows the platform to use leightweighted and/or heavyweighted processes. The process service allows to divide the control flow of the client so that parallel processing subunits are possible. The heavyweighted processes need a mini-plattform, on wich the communication service (IPC service) is installed. So the service wich runs with a heavyweighted process communicates with the platform through a mini-plattform. The process service offers a unified interface to the process administration of a OS.
2. **real time storing service** (Echtzeitspeicherdienst): allows realtime access to the memory. Not realtime accessible memory is unificated in C and C++(malloc in C, new in C++). If the real time storing service is installed the platform recognizes this and in the future the platform will be able to work off orders without a connection in realtime.
3. **communication services** (Kommunikationsdienste): serve the platform to send orders on arbitrary communication mediums. (The interprocess communication is also regarded as communication medium).
4. **event service** (Ereignisdienst): with this service the platform can observe and control the timing.

2.3.2.1.2.3 Extending services (Erweiterungsdienste)

These services add to the platform functionality which the OS does not offer. E.g. protocoll service or safety/security service.

2.3.2.1.3. Description of the interfaces

In the following we describe the components with ist interfaces.

2.3.2.1.3.1 The interface of the order administration (Auftragsverwaltung)

The order administration has 4 objects:

- **order** (Auftrag)
- **depot** (Ablage)
- **order factory** (Auftragsfabrik)
- **order scheduler** (Auftragsscheduler): delivers the orders with special algorithms which send to the servers the orders with consideration to priority, time conditions etc. of the orders.

2.3.2.1.3.1.1 The interface of the order object (Auftragsobjekt)

An order consists of 3 data sectors:

- **connecting data**: client, server, error instance (a service, which is informed if an error occurs or timing limits are violated)
- **order data**: order ID, realtime conditions (the client can choose the following parameters: point of time when the order is delivered, earliest and latest pickup of the order by the server, latest execute, latest end), information about repeated delivering
- **user data**: the client writes the format of the datas about the called function with its data into the order

The scheduler evaluates and supervises the connecting data and order data.

2.3.2.1.3.1.2 The interface of the tray object and of the connection

Trays are resources of the order administration to store orders. If the communication is connection orientated then the depot is allocated when the connection is built. Otherwise (connectionless communication) the tray is allocated when the order is created.

The connection includes information about the involved services and a pointer to the tray which is to be used for the connection.

One connection object can have zero or one tray object.
In the following there are listed some interfaces of both the tray class and the connection class:

```
class OSAJOB_Tray
{
public:
    // Adds a job to the tray. The tray sets the position of the order to the start of a free
    // slot on the tray. The position of the result is the position of the order plus the
    // offset given in function call.
    OSA_Error addJob(OSAJOB_Job& job, OSA_UShort resultOffset);

    // retrieves the current number of jobs in the tray
    OSA_UShort getCurrentNbOfJobs();

    ....
};

class OSAJOB_Connection
{
public:
    // Returns the id of the client service
    OSASRV_ServiceId getClientServiceId();

    ....
};
```

2.3.2.1.3.1.3 The interface of the order factory and the order scheduler

The order factory is responsible for the creation, the storage and destruction of orders. The relation between orders, trays and connections is here administrated.

One job scheduler has zero or one job factory. One job factory can have an arbitrary number of jobs and connections. One job can have an arbitrary number of time constraints.

2.3.2.1.3.2 The interface of the service administration

The service administration is a simple data storing component for the installed services. Because the application is distributed also this data storing is distributed.

```
class OSASRV_Services
{
public:
    // Adds an ARS (Adress Resolution Service - to recognize services in the distributed environment)
    // and an ARS-list id that is used for adress resolution.
    OSA_Error addARS(OSA_PlatformAdress arsPlatformAddr,
                    OSASRV_ServiceId arsRemoteServiceId, OSAARS_ListId listId);

    ...
};
```

2.3.2.1.3.3 The interfaces of the base services (Basisdienste)

The base services are components which offer their services in a very different way which depends on the OS and the hardware for which these base services are implemented.

2.3.2.1.4. Communication

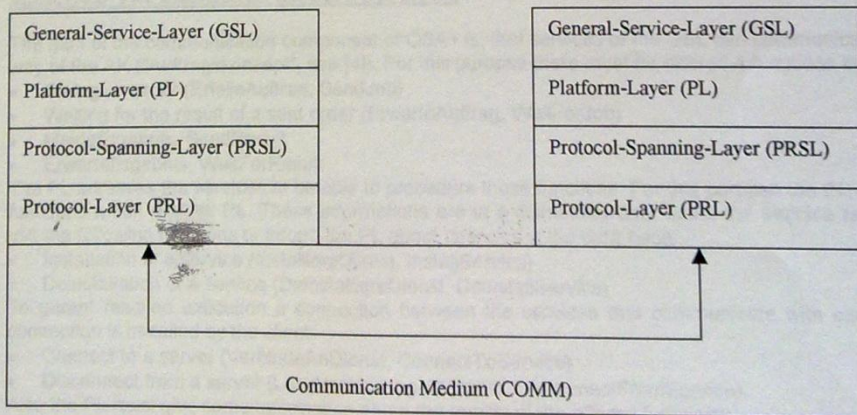
2.3.2.1.4.1 Overview

OSA+ services can give orders which are to be worked out at places not in the same address room or computer. Such orders are worked off by the middleware with a special communication component. For this purpose the communication has the following requirements:

- There must be a **complete network**, so that each platform reach each other platform: The goal is to assembly all platforms to one system. Between these platforms there must be a communication with different communication mediums. The necessary protocols are not specified by OSA+, so that the integration of arbitrary mediums and protocols is possible (e.g. CAN-bus, Ethernet, Internet, ...). The platforms are divided for this purpose into domains and get addresses which allow an identification in the whole system. This address has two parts: a domain number and a platform number within the domain. The passing from one domain to another is where a transition from one protocol or medium to another takes place. In the Lotte-FCS system there are only two platforms to communicate with each other: the platform of the ifrsenso in the airship and the platform of the laptop at the ground. They communicate with radio ethernet.
- **Realtime communication** and **non-realtime communication** must be offered (if the communication system permits this): realtime communication requires that at a determined time point all resources which are needed for the communication are allocated and after this (for the time of the communication) are not changed. For this purpose a static connection between the communication partners (i.e. the services) is built, which garants a fixed answer time.
- **Simple interface** for the platform and other users: the communication offers ist functionality in form of services of the platform and other users. It offers the possibility to construct and destruct connections along these services, to send/receive data and to demand QoS(quality of service)-parameter.

2.3.2.1.4.2 The communication layers

The communication component of OSA+ is achieved by a layer model. For this purpose the complete system is divided into layers (see fig. below):



General Service Layer (GSL)

The top layer is built of the services, which realize the application. These services may not be in the same address room or even the same same computer. They communicate with each other by orders. There are

also services which abstract from the OS or the underlying hardware (Extending services). These services lie in the platform layer and are called by the order concept or the call interface. They don't need the communication component of OSA+ because they are lie in the same addressroom and the same platform.

Platform Layer (PL)

The platform layer offers an interface. With this interface the platform is able to send and to receive orders. This layer consists of a platform. The kernel functions of the platform are the following:

- installation and deinstallation of local available services
- create connections between services
- sending and receiving of orders by the way of these connections or out of these connections
- investigation of QoS parameters

The platform layer has to pass on orders to local available services and to pass on orders for far lying services as packet data to underlying layers.

Protocol Spanning Layer (PRSL)

This layer sends data (not orders) from one instance to another instance. At the interface of this layer the following functions are needed:

- Construction and destruction of connections
- Data sending on these connections or not on these connections
- investigation of QoS parameters

The PRSL must determine a suitable protocol for the connection. If a change of domains is needed also more than only the 2 instances between which the communication takes place. So the PRSL does a part of the **routing activity** of the system.

The PRSL offers its functionality as a service, the **High-Level Communication Service**.

Protocol Layer (PRL)

This layer is the location of all communication protocols on all platforms. For OSA+ ends the communication on this layer. The rest of the layers which are necessary for communication (e.g. ISO-OSI model, internet model) are abstracted by this layer.

There are several services in this layer, the **Low-Level Services**, which support the different communication protocols and communication mediums. Each of this service offers a normed interface so that the PRSL can identify such a service and send data on it.

The Low-Level services must offer the following functionality:

- Construction and destruction of connections
- Data sending on these connections or not on these connections
- investigation of QoS parameters

2.3.2.1.4.3 The interaction between the layers

The goal of the communication component of OSA+ is, that services of the GSL can communicate by the way of the AK ("Auftragskonzept", see [4]). For this purpose there must be offered 4 functions by the PL

- Giving an order (ErteileAuftrag, SendJob)
- Waiting for the result of a sent order (ErwarteAuftrag, WaitForJob)
- MeldeErgebnis, SendResult
- ErwarteErgebnis, WaitForResult

The PL addresses the services to be able to procedure these functions. For this purpose the PL must know all functions which use the PL. These informations are in a distributed data base, the **service table**. Services use the following functions to inform the PL about changes in the data base:

- Installation of a service (InstalliereDienst, InstallService)
- Deinstallation of a service (DeinstalliereDienst, DeinstallService)

To garant realtime execution a connection between the services that communicate with each other. The connection is installed by the client:

- Connect to a server (VerbindeAnDienst, ConnectToService)
- Disconnect from a server (LöseVerbindungMitDienst, DisconnectFromService)

Also the PL must give some information about the quality of the offered functions:

- ErmitteQoS, GetQoS

PL offers also non-realtime delivering and supervision. In this case a static connection is not necessary and the order can be directly given to a service or by the way of an already existing connection to the service.

The differentiation between communication with static connection and communication without static connection is done by PRSL. The PRSL offers to PL the possibility to send data via a connection or outside of a connection:

- Connect to a communication partner (ErstelleVerbindung, Connect)
- Disconnect from a communication partner (LöseVerbindung, Disconnect)
- SendeDaten, SendData
- EmpfangenDaten, ReceiveData
- ErmitteQoS, GetQoS

The PRSL expects the same functionality from the PRL.

2.3.2.1.4.4 Interaction between the instances

The interaction between different instances of one layer is only interesting for the PL, the PRSL and the PRL, because these layer work off the communication task. The instances of the GSL, the services, have to work off a user specific task.

Platform Layer

The platform layer has the following 2 tasks:

- to store information about the services in a database (**service administration** (Dienstverwaltung))
- to deliver orders to these services (**order administration** (Auftragsverwaltung))

The data bases of the service administration are distributed. So the communication must grant consistency of the data bases. In a instance there are only stored the data which are locally needed in order not to handicap the instances of the PL, that means the platforms. This locally needed data consists of information and orders for local services and orders for far services that were contacted by local services.

So an instance of the PL must get the help of other instances, if it wants to build connections to far services.

This can be done in 2 ways:

1. the instance knows one or more central instances that can give the needed information, or
2. the instance doesn't know such an instance and must ask all other instances (Broadcast)

The central instances for adress resolution are offered as services: **Adress Resolution Services (ARS)**

Protocol Spanning Layer

For the PSL it is the same as for the PL. The PSL has also distributed data bas that knows the existing instances of the PL and resolves the addresses in the same way as the PL.

Protocol Layer

The data stored in the PRL and the interactions of the instances are not normed. There is only proposed a metaformat. It is good to follow this proposal because in this way a homogeneous communication can be done.

2.3.2.1.5. Remarks on the implementation

See [6], p.27 for remarks on the implementation of the order administration, the service administration, the realtime storage and the process service.

2.3.2.1.6. Static orders

OSA+ - conform services offer several orders/funcions that allow the platform to work with arbitrary services.

2.3.2.1.6.1 Quality of Service

In OSA+ all services give, if they were asked to do so, information about the quality of its execution of orders.

2.3.2.1.6.2 Constructor and Destructor

At the creation of a service the platform allows the service an initialisation (like a constructor in OO programming languages). Also before deletion of a service the service can give free resources (like a destructor)

2.3.2.1.6.3 Call interface

The call interface allows to use a service like normale object a function interface. In OO languages there must be a interface for this purpose.

2.3.2.1.7 Initialization

The initialisation is done in the following defined order:

- initialize the platform
- register the initialisation service
- install the base services
- install the application services
- give (dt. absetzen) initialization orders (optional)
- blocade: after the blocade initialization orders can be given (dt. absetzen)
- deinstall the application services
- deinstall the base services
- log off initialization service
- end the platform
- end the application

2.3.2.2 The real-time operating system VxWorks

The content of the following is mostly extracted from [7].

2.3.2.2.1 Overview of the role of VxWorks in the development of real-time applications

UNIX and windows hosts are excellent systems for program development and for many interactive applications. However, they are not appropriate for real-time applications.

On the other hand, traditional RTOSs provide poor development environments. The Windriver philosophy is to utilize two complementary and cooperating systems (e.g. VxWorks and Windows) and let each do what it does best. VxWorks handles the critical real-time chores, while the host machine is used for program development and for applications that are not time critical.

You can scale VxWorks to include exactly the feature combinations your application requires. During development, you can include additional features to speed your work (such as networking facilities), then exclude them to save resources in the final version of your application.

You can use the cross-development host machine to edit, compile, link and store real-time code, but then run and debug that real time code on VxWorks. The resulting VxWorks application can run standalone - either in ROM or disk-based - with no further need for the network or the host system.

However, the host machine and VxWorks can also work together in a hybrid application, with the host machine using VxWorks systems as real-time "servers" in a networked environment. For instance, a VxWorks system controlling a robot might itself be controlled by a host machine that runs an expert system, or several VxWorks systems running factory equipment might be connected to host machines that track inventory or generate reports.

In our case the VxWorks system, running on the ifrsenso, controls the airship FCS (a special robot). The FCS gets commands from the ground laptop. So the ground laptop (the host machine) controls the VxWorks system.

2.3.2.2.1 Overview of VxWorks facilities

This section provides a summary of VxWorks facilities:

- **High-Performance Real-time Kernel Facilities**

The VxWorks kernel, wind, includes multitasking with preemptive priority scheduling, intertask synchronization and communications facilities, interrupt handling support, watchdog timers, and memory management.

- **POSIX Compatibility**

POSIX (Portable Operating System Interface) is a set of standard operating system interfaces based on the UNIX operating system. The need for standardization arose because enterprises using computers wanted to be able to develop programs that could be moved among different manufacturer's computer systems without having to be recoded. UNIX was selected as the basis for a standard system interface partly because it was "manufacturer-neutral." However, several major versions of UNIX existed so there was a need to develop a common denominator system.

VxWorks provides most interfaces specified by the 1003.1b standard (formerly the 1003.4 standard), simplifying your ports from other conforming systems.

The 1003.1b standard defines a standard operating system interface and environment to support application portability at the sourcecode level. It is intended to be used by both application developers and system implementors.

The following aspects are part of the standard:

- (1) Terminology, concepts, and definitions and specifications that govern structures, headers, environment variables, and related requirements
- (2) Definitions for system service interfaces and subroutines
- (3) Language specific system services for the C programming language
- (4) Interface issues, including portability, error handling, and error recovery

Not: user interface, graphics interfaces, dbms, ...

- **I/O System**

VxWorks provides a fast and flexible ANSI C-compatible I/O system, including UNIX standard buffered I/O and POSIX standard asynchronous I/O. VxWorks includes the following drivers:

- Network driver - for network devices (Ethernet, shared memory)
- Pipe driver - for intertask communication
- RAM "disk" driver - for memory-resident files
- SCSI driver - for SCSI hard disks, diskettes, and the tape drives
- Keyboard driver - for PC x86 keyboards (x86 BSP only)
- Display driver - for PC x86 VGA displays (x86 BSP only)
- Disk driver - for DDE and floppy disk drives (x86 BSP only)
- Parallel port driver - for PC-style target hardware

- **Local File Systems**

VxWorks provides fast file systems tailored to real-time applications. One file system is compatible with the MS-DOS file system, another with the RT-11 file system, a third is a "raw disk" file system, a fourth supports SCSI tape devices and a fifth supports CD-ROM devices.

- **C++ Development Support**

In addition to general C++ support including the iostream library and the standard template library, the optional component Wind foundation Classes adds the following C++ object libraries:

- VxWorks Wrapper Class library
- Tools.h++ library from Rogue Wave

- **Shared-Memory Objects (VxMP Option)**

The VxMP option provides facilities for sharing semaphores, message queues, and memory regions between tasks on different processors.

- **Virtual Memory (including VxVMI Option)**

VxWorks provides both bundled and unbundled (VxVMI) virtual memory support for boards with an MMU, including the ability to make portions of memory noncacheable or read-only, as well as a set of routines for virtual-memory management.

- **Target-resident Tools**

In the Tornado development tools reside on the host system. However, a target-resident shell, module loader and unloader, and symbol table can be configured into the VxWorks system if necessary.

- **Utility Libraries**

VxWorks provides an extensive set of utility routines, including interrupt handling, watchdog timers, message logging, memory allocation, string formatting and scanning, linear and ring buffer manipulations, linked-list manipulations, and ANSI C libraries.

- **Performance Evaluation Tools**

VxWorks performance evaluation tools include an execution timer **for timing** a routine or a group of routines, and utilities to show CPU utilization percentage by task.

- **Target Agent**

The target agent allows a VxWorks application to be remotely debugged using the Tornado development tools.

- **Board Support Packages**

Board support Packages (BSPs) are available for a variety of boards and provide routines for hardware initialization, interrupt setup, timers, memory mapping, and so on.

- **VxWorks Simulator (VxSim) and Logic Analyzer (Windview)**

Tornado comes with an integrated simulator and software logic analyzer on all host platforms. Vxsim simulates a Vxworks target for use as a prototyping and testing environment. WindView provides advanced debugging tools for the simulator environment.

- **Network Facilities**

VxWorks provides "transparent" access to other VxWorks and TCP/IP-networked systems, a MUX interface (supporting advanced features such as multitasking, polled-mode Ethernet, and zero-copy transmission), a BSD¹ Sockets-compliant programming interface, remote procedure calls (RPC), SNMP (optional), remote file access (including NFS client and server facilities and a non-NFS facility utilizing RSH, FTP, or TFTP), BOOTP, proxy ARP, DHCP, DNS, OSPF (optional), and RIP. All VxWorks network facilities comply with standard Internet protocols, both loosely coupled over a backplane bus using shared memory.

For information on VxWorks network support, see [8].

2.3.2.2.2 The Basic OS of VxWorks

Modern real-time systems are based on the complementary concepts of multitasking and intertask communications. A multitasking environment allows a real-time application to be constructed as a set of independent tasks, each with its own thread of execution and set of system resources. The intertask communication facilities allow these tasks to synchronize and communicate in order to coordinate their activity. In VxWorks, the intertask communication facilities range from fast semaphores to message queues and pipes to network-transparent sockets.

Another key facility in real-time systems is hardware interrupt handling, because interrupts are the usual mechanism to inform a system of external events. To get the fastest possible response to interrupts, *interrupt service routines (ISRs)* in VxWorks run in a special context of their own, outside of any task's context.

This section discusses the multitasking kernel, tasking facilities, intertask communication, and interrupt facilities which are the heart of the VxWorks run-time environment.

2.3.2.2.2.1 Special Wind Features and POSIX Features

¹ BSD stands for Berkley Software Distribution, and refers to a version of UNIX.

The POSIX standard for real-time extensions (1003.1b) specifies a set of interfaces to kernel facilities. To improve application portability, the VxWorks kernel, *wind*, includes both POSIX interfaces and interfaces designed specially for VxWorks.

2.3.2.2.2 Tasks

It is often essential to organize applications into independent, through cooperating, programs. Each of these programs, while executing, is called a *task*. In VxWorks, tasks have immediate, shared access to most system resources, while also maintaining enough separate context to maintain individual threads of control.

2.3.2.2.2.1 Multitasking

2.3.2.3 The adaptation VxWorks and OSA+

In the following the OSA+ implementation for VxWorks is described. There are the following files:

- Service Management:

- Job Management:

- Job Scheduler:

- process service:
osatsk.h

osatsk(vxworks?).c

...
- High level communication service

(hier und in den folgenden Modulen kommen die Funktionen, die in dem paper "Die Kommunikation in OSA+ beschrieben wurden.")

- ARS

The code of the files are listed in the appendix. The interaction diagram of an execution example is also there

2.3.2.4 The adaptation of OSA+ and the communication system (RS232, radio ethernet)

2.3.3 Die Systemplattform des Laptops am Boden

Die Benutzerschnittstelle am Boden läuft auf dem Betriebssystem Windows 98.

2.4 The measurement and actuating services (Meß- und Steuerdienst, MS service) of the Lotte-FCS

A **measurement service** has to measure the physical variables via the sensors. An **actuating service** has to steer the actuators to influence physical systems.

The measurement and actuating services are the connecting link between **the physical system which is to be automatized (zu automatisierendes System)** and the **automation system (Automatisierungssystem)**.

The transformation chains

Physical system (airship) -> automation system (FCS)

Physical variable -> analog electrical signal -> digital values -> coded information

Automation system (FCS) -> physical system (airship)

coded information -> digital values -> analog electrical signal -> physical variable

- **Physical variable <-> analog electrical signal**

This is performed by the sensors and actuators. In most cases a electrical circuit is needed for **signal conditioning (signal matching)** with the following tasks:

- Amplifying
- Elimination of non-linearities
- Electrical transformation to an industry norm for analog interfaces (e.g. 0-20 mA) for save transmission.

Sometimes this circuit is in the sensor/actuator and sometimes it is outside of it.

- **analog electrical signal <-> digital values**

This is performed with DA-/AD-converters

- **digital values <-> coded information**

This is done by measurement and actuating services. The coded information includes normally:

- the measurement value
- time of measurement
- place of measurement (measurement channel)

With an measurement service parts of the signal conditioning of measurement values (the calibration) is done after the conversion to digital values (**Meßwertaufbereitung**).

2.4.1 The general measurement service (GMS)

This service gives orders to all other measurement services and consolidates the results with a kalman filter. The following is a listing of the other measurement services which serve as clients for the GMS:

- BHMS (barometric measurement service)
- DPMS (differential pressure measurement service)
- IMUS (inertial measurement unit service)
- GPSS (GPS service)
- RSMS (residually sensors measurement unit)

It follows a description of the functionality of the GMS:

Creating a service

2.4.2 IMUS

Liefert Beschleunigung \mathbf{a} , die drei Lagewinkel $\mathbf{\Phi}$, die Winkelgeschwindigkeiten p , q und r .

2.4.3 GPSS

Liefert die 2D-Position, die Höhe H , das Heading und $\mathbf{V}_{\text{Grund}}$.

2.4.4 DPMS

Liefert Anstellwinkel α , Schiebewinkel β und \mathbf{V}_a .

2.4.5 The barometric hight measurement service (BHMS)

Liefert die Höhe H durch einen barometrischen Höhenmesser.

2.4.6 RSMS (residually sensors measurement unit)

Can be simillarly implementated like the MSR service for the AK (developed at IMA, Univ. of Karlsruhe ([4]).

2.5 Die Mensch-Maschine-Dienste des Lotte-FCS

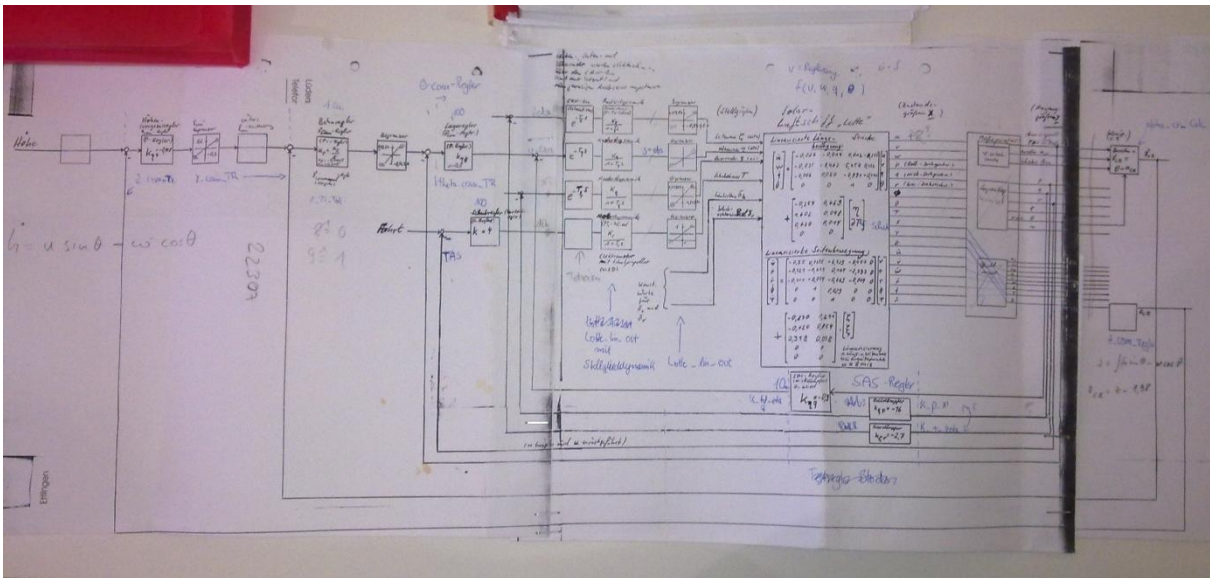
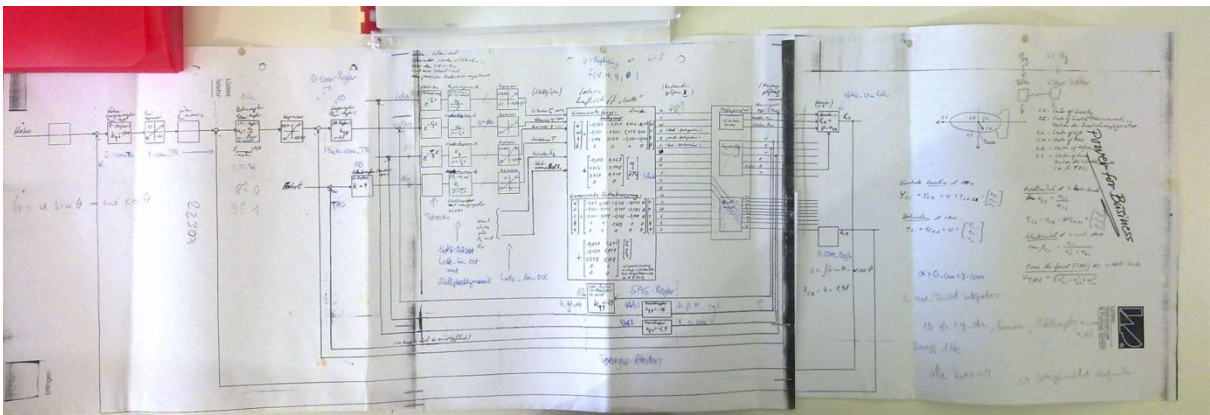
2.6 Datenhaltungs-Dienste des Lotte-FCS

2.7 Dienste-Architektur des Lotte-FCS

Literatur

- [1] Gamma et. al., "Entwurfsmuster", 1996
- [2] R.Brockhaus, "Flugregelung", 1994
- [3] Press, W.H., Flannery, B.P., Teukolsky, S.A., "Numerical Recipies", Cambridge University Press, Cambridge, 1986
- [4] Prof. Brinkschulte, Vorlesungsunterlagen "Mikrorechnertechnik II", SS 99, Institut für Mikrorechner und Automation, Uni Karlsruhe
- [5] Besprechung mit Prof. Schweizer (Institut für Mikrorechner und Automation, Uni Karlsruhe) vom 13.3.2000 im Institut für Mikrorechner und Automation, Uni Karlsruhe.
- [6] Brinkschulte, Krakowski, Riemschneider, „Die OSA+ Architektur“, interner Bericht, Institut für Mikrorechner und Automation, Uni Karlsruhe, Stand: 28.3.2000
- [7] VxWorks Programmer's Guide, 5.4; Edition 1; 6 May 1999; Part #: DOC-12629-ZD-01; WindRiver Systems
- [8] VxWorks Network Programmer's Guide

5.1 Block diagram of the Control Loops



6 Optimization of the Flight Control System architecture

6.1 The architecture of the Flight Control System and its simulation on the middleware OSA+ running on the operating system Windows

Based on:

Samir Mourad, Diplomarbeit (Master Thesis) „Anbindung des Echtzeitbetriebssystems VxWorks an die Middleware OSA+ und Integration dessen in eine dienstorientierte Test-Echtzeitsystem-Umgebung“

Institut für Mikrorechner und Automation, Univ. Karlsruhe, Supervisor: Prof. Dr. U. Brinkschulte“, Institut für Prozessrechnerentechnik und Robotik, Faculty of Computer Science, Universität Karlsruhe (TH), December 2000

Überblick

1. Einleitung

Allgemeingehaltene Einführung in den Aufbau von Echtzeitsystemen. Anforderungen an heutige Echtzeitsysteme: Rechtzeitigkeit der Verarbeitung, wiederverwendbarer Aufbau des Gesamtsystems. Mögliche Lösung: dienstorientierter Aufbau (modular, einfache Erweiterbarkeit des Echtzeitsystems, gibt natürlich den funktionalen Ablauf bei einem Echtzeitsystem wieder) mit Middleware (Trennung von Anwendungssoftware und Betriebssystem bzw. Hardware)

2. Aufgabenstellung und Lösungsansatz

2.1 Aufgabe

Erstellung eines dienstorientierten Test-Echtzeitsystem mit zwei Rechnern, die mit Funk-Ethernet verbunden sind. Der zweite Rechner hat einen CAN-Bus, an dem Sensoren und Aktoren hängen. Über Funkethernet gibt der erste Rechner dem zweiten Rechner Anweisungen bezüglich der Bedienung der Sensoren und Aktoren. In die andere Richtung werden Meßdaten übertragen und auf einer Oberfläche des ersten Rechners dargestellt.

2.2 Vorstellung einiger vorhandener Systeme

2.2 Lösungsansatz

dienstorientierter Systemaufbau (siehe entsprechende Abbildung in Lottefcs_vom_Leptop.doc)

Begründung für die Auswahl von VxWorks: relativ einfache Handhabbarkeit wegen umfangreicher Entwicklungsumgebung, in der Praxis erprobtes Standard-RTOS.

3. OSA+

Beschreibung von Prozeßdienst, Ereignisdienst (SA von Sven), Kommunikationsdienst und Datenhaltungsdienst (Zusammenfassung aus Institutsbericht)

4. VxWorks

Vorstellung des Basic OS, des I/O System, des lokalen Filesystems und des Kommunikationssystems (Zusammenfassung aus Programmer's Guide)

5. Anpassung von VxWorks an OSA+

5.1 Prozeßdienst

5.2 Ereignisdienst (SA von Sven)

5.3 Kommunikationsdienst

6. Integration ins Gesamtsystem

Beschreibung der Dienste, der Treiber, der HW und des Zusammenspiels des Gesamtsystem

6.1 Oberfläche des 1. Rechners und zugehörige Dienste

6.2 Kommunikationsdienst zwischen 1. und 2. Rechner

6.3 Meßdienste und Stelldienste auf 2. Rechner

6.4 Treiber für VxWorks bzgl. CAN-Bus (DA von M. Subhan)

(6.5 Sensoren/Aktoren auf dem CAN-Bus)

7. Ausblick

Variierungsmöglichkeiten: Ersetzung der Sensoren/Aktoren, Austausch des VxWorks durch ein anderes RTOS bzw. direkte Anbindung an die HW, Ersetzung des Funkethernet durch ein anderes Protokoll.

6.2 Results

Entwicklung einer verteilten

Experimentalumgebung fuer die Middleware OSA+
(ESfOSA+)

Diplomarbeit von Samir Mourad

1. Aufgabenstellung

Thema der Arbeit ist die **Entwicklung eines Experimentalsystems fuer die OSA+-Anbindung an Windows NT** und die darauffolgende Durchfuehrung und Dokumentation von Test mit diesem Experimentalsystem.

Das Experimentalsystem wird im folgenden mit **EsfOSA+** (Experimentalsystem fuer OSA+) bezeichnet.

EsfOSA+ soll eine **Simaulation eines embedded system** werden, wobei die Hardwarekomponenten wie Aktoren und Sensoren als digitale Uebertragungsglieder simuliert werden.

[Der Kern von EsfOSA+ ist **komplexer Regelkreis**, der eine **zu automatisierende Anlage regelt**.

Ausgangspunkt war ein der vorliegenden Arbeit ist ein **analoger Regelkreis**, welcher digitalisiert wurde und in dienstorientierter Struktur realisiert wurde. Die Dienste sollen auf der Middleware OSA+ ablaufen.

Es soll **getestet** warden, ob es Bedingungen gibt, unter denen eine solche Regelkreisimplementierung unter OSA+/Windows NT echtzeitfaehig ablaufen kann.

2. Digitalisierung des analogen Regelkreises

Bild 1: Der analoge Regelkreis

Bild 2: Der digitale Regelkreis

3. Konstruktion der dienstorientierten Systemstruktur fuer EsfOSA+

Die Dienste-Architektur des EsfOSA+

Anforderungen an "ideale" Dienste:

- * Plattformunabhaengigkeit (-> Portierbarkeit)
- * Skalierbar und Konfigurierbar (-> Anpassung an die Anwendung, Erweiterbarkeit)
- * Austauschbar (einfacher Wechsel von Komponenten)
- * Kompatibel (-> Datenaustausch)
- * Testbar (-> Fehlersuche, Wartung)
- * Echtzeitfaehig (Wenn erforderlich)

EsfOSA+ soll den in Bild 2.1.1 dargestellten Kaskadierten Regelkreis implementieren.

4. Die OSA+ Architektur

OSA+ ist eine skalierbare echtzeitfaehige Middleware. Sie erlaubt es, verteilte Anwendungen zu entwickeln, die echtzeitfaehig ablaufen und innerhalb eines heterogenen Netzwerks existieren. Die zugrundeliegende Hardware sowie die Betriebssysteme warden dazu optimal ausgenutzt.

Der OSA+-Ereignisdienst

Der Ereignisdienst dient dazu, Jobs zu vorgegebenen Zeiten innerhalb eines festgelegten Intervalls ausfuehren zu lassen. Er besteht aus 2 Teilen: dem **eigentlichen Ereignisdienst** und den **Funktionen fuer die Uhrzeit**.

5. Experimentelle Ergebnisse

5.1 Die Simulationshardware

Die Simulation life auf einem PC mit einem Intel Pentium Prozessor mit 133 MHz und 32 MB RAM ab.

Sensorik und Aktorik wurden softwaremaessig simuliert. Es war keinerlei reale Sensorik oder Aktorik angeschlossen.

5.2 Die Testreihe

Bei den Tests wurden verschiedene Dienste eingesteckt und das Zusammenspiel getestet. Dabei wurde so vorgegangen, dass zunächst nur die Dienste der inneren Regelschleifen mit den zugehörigen Messdiensten und natürlich der Streckensimulation und des Stelldienstes eingesteckt wurden. Später wird sukzessive der ganze kaskadierte Regelkreis von innen nach aussen aufgebaut.

Einstecken von PS, S1_M, S2_M, S, R_In1, R_In2

Test 2.1

Festlegung der max. Antwortzeiten der einzelnen Dienste (durch den Ereignisdienst)

PS	10ms
R_In1	20ms
R_In2	100ms

Wie schon im vorigen Test wurden einige Größen über der Zeit aufgezeichnet (Graphen von links nach rechts):

$y_4, x_1, x_5, x_{12}, x_8, y_1$, (d.h. der 1. Graph v. l. zeigt y_4 , der 2. Graph v. l. zeigt x_1 usw.)

Im Bild R_In2_KnappeAntwortzeit_1.bmp ist zu sehen, dass die Dienste nicht vollständig ausgeführt werden konnten. Im nächsten Versuch (siehe R_In2_KnappeAntwortzeit_2.bmp) wird daher die Antwortzeit für R_In1 erhöht.

R_In2_KnappeAntwortzeit_1.bmp

Test 2.2

Einstellungen wie bei Test 2.1, nur ist die Antwortzeit fuer R_In1 diesmal 30 ms.

R_In2_KnappeAntwortzeit_2.bmp

Einstecken von PS, S1_M, S2_M, S, R_In1, R_In2, MV_2, R_M, MV_2, R_A
(vollstaendiger Regelkreis)

Test 4.1

Festlegung der max. Antwortzeiten der einzelnen Dienste (durch den Ereignisdienst)

PS	10 ms
R_In1	30 ms
R_In2	100 ms
MV_1	100 ms
R_M	200 ms
R_A	300 ms

Testdauer: 30 Sekunden

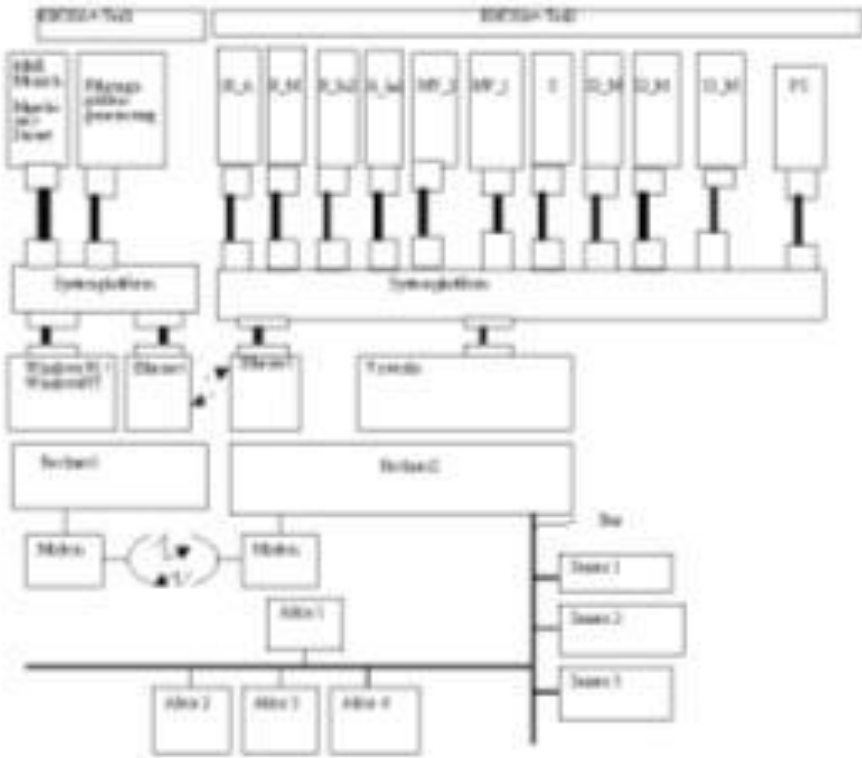
Auf der folgenden Seite ist R_A_KnappeAntwortzeit_1.bmp zu sehen.
(v.l.n.r.: x_12, y_MV1, x_8, y_1, y_4, x_1, x_3, x_5)

6. Bewertung der experimentellen Ergebnisse und Ausblick

Die Experimente ergaben folgende Erkenntnis: Wird im Ereignisdienst die Rueckgabezeit eines Dienstes zu klein eingestellt, dann wird nicht mehr richtig geregelt. Es ergeben sich Bilder wie "R_In1_zuKleineAntwortzeit.bmp". Dies bedeutet: kommt man bei einer OSA+-Anpassung an WindowsNT mit den geforderten Antwortzeiten in den Bereich von ca. 20 ms, ist auch praktisch gesehen kein rechtzeitiges Antwortverhalten garantiert. Dass die zugrundeliegende Simulationshardware keinen grossen Einfluss hat, wird dadurch bestaetigt, dass nahezu die gleichen Tests auf einem AMD Duron 900 MHz Prozessor mit 256 MB gemacht wurden, und dort fast die gleichen Ergebnisse herauskamen.

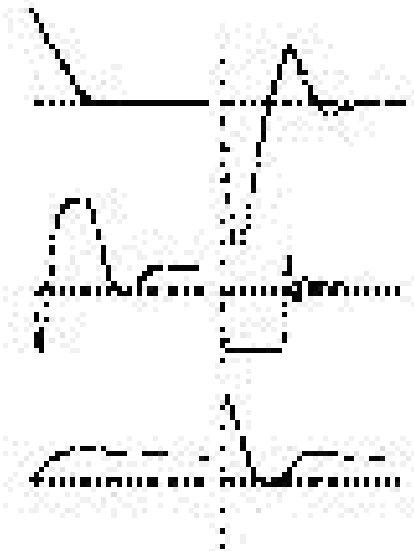
Es waere interessant, OSA+ auf ein Echtzeitbetriebssystem anzupassen, und dort aehnliche Tests ablaufen zu lassen.

6.2.1.1 System Architecture of Airship Flight Control System



6.2.1.2 Simulation Code (in programming language C)

6.2.1.3 Simulation results



Entwicklung einer verteilten Experimentalumgebung für die Middleware OSA+ (ESfOSA+)

Diplomarbeit von Samir Mourad

Inhaltsverzeichnis

1. Aufgabenstellung	2
2. Digitalisierung des analogen Regelkreises	3
3. Konstruktion der dienstorientierten Systemstruktur für EsfOSA+	5
Die Dienste-Architektur des EsfOSA+	5
4. Die OSA+ Architektur	7
<i>Der OSA+-Ereignisdienst</i>	7
5. Experimentelle Ergebnisse	8
5.1 Die Simulationshardware	8
5.2 Die Testreihe	8
Einstecken von PS, S1_M, S2_M, S, R_In1, R_In2	9
<i>Test 2.1</i>	9
<i>Test 2.2</i>	11
Einstecken von PS, S1_M, S2_M, S, R_In1, R_In2, MV_2, R_M, MV_2, R_A (vollständiger Regelkreis)	12
<i>Test 4.1</i>	12
6. Bewertung der experimentellen Ergebnisse und Ausblick	14



2. Digitalisierung des analogen Regelkreises

Bild 1: Der analoge Regelkreis

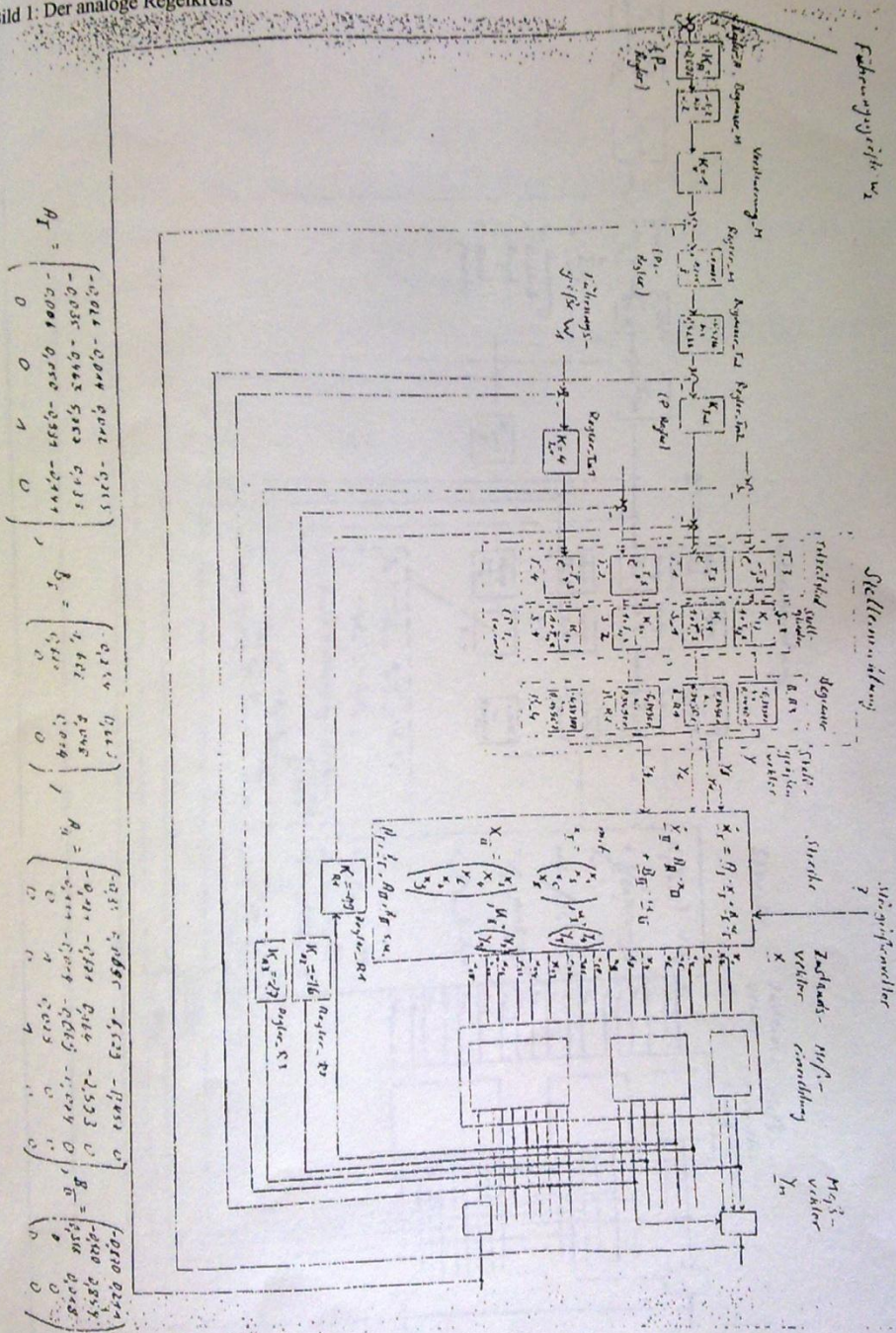
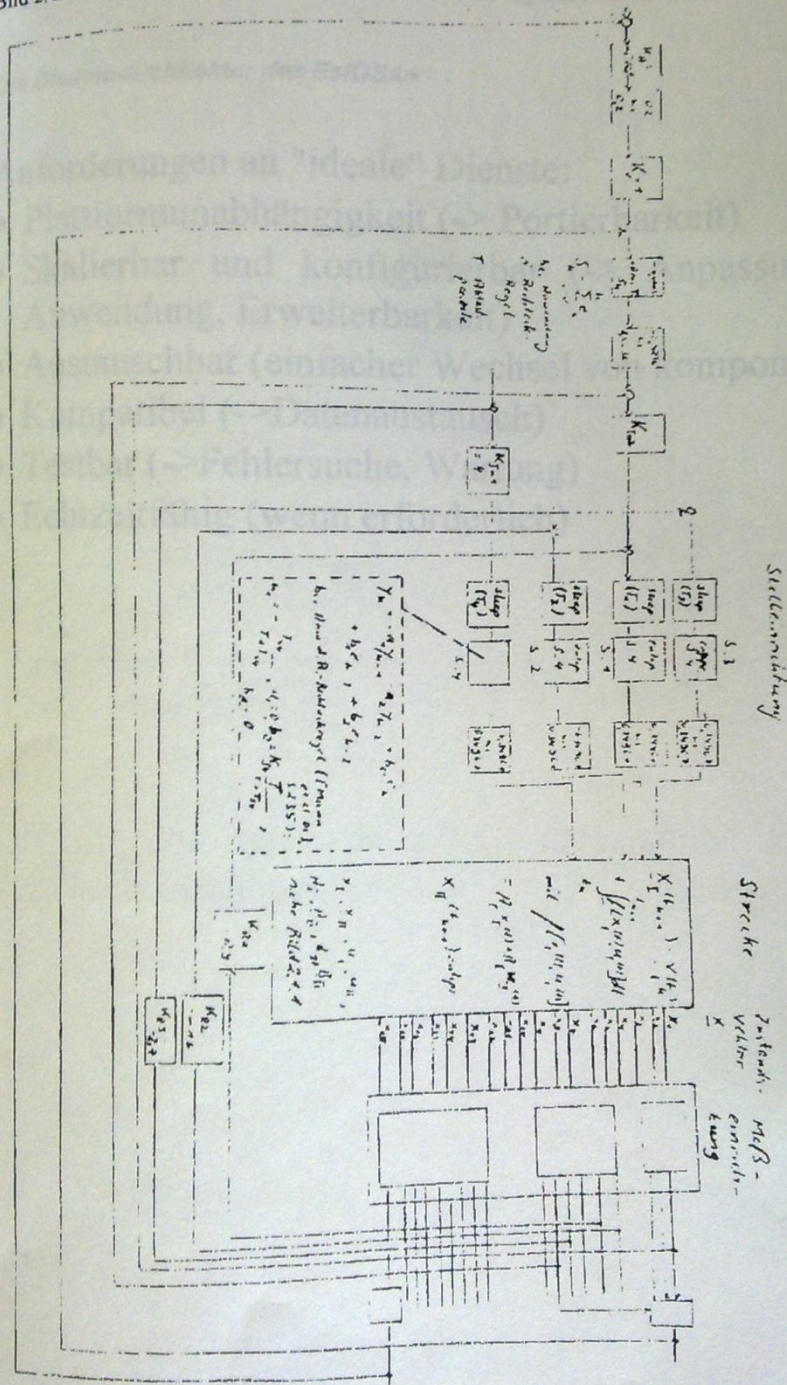
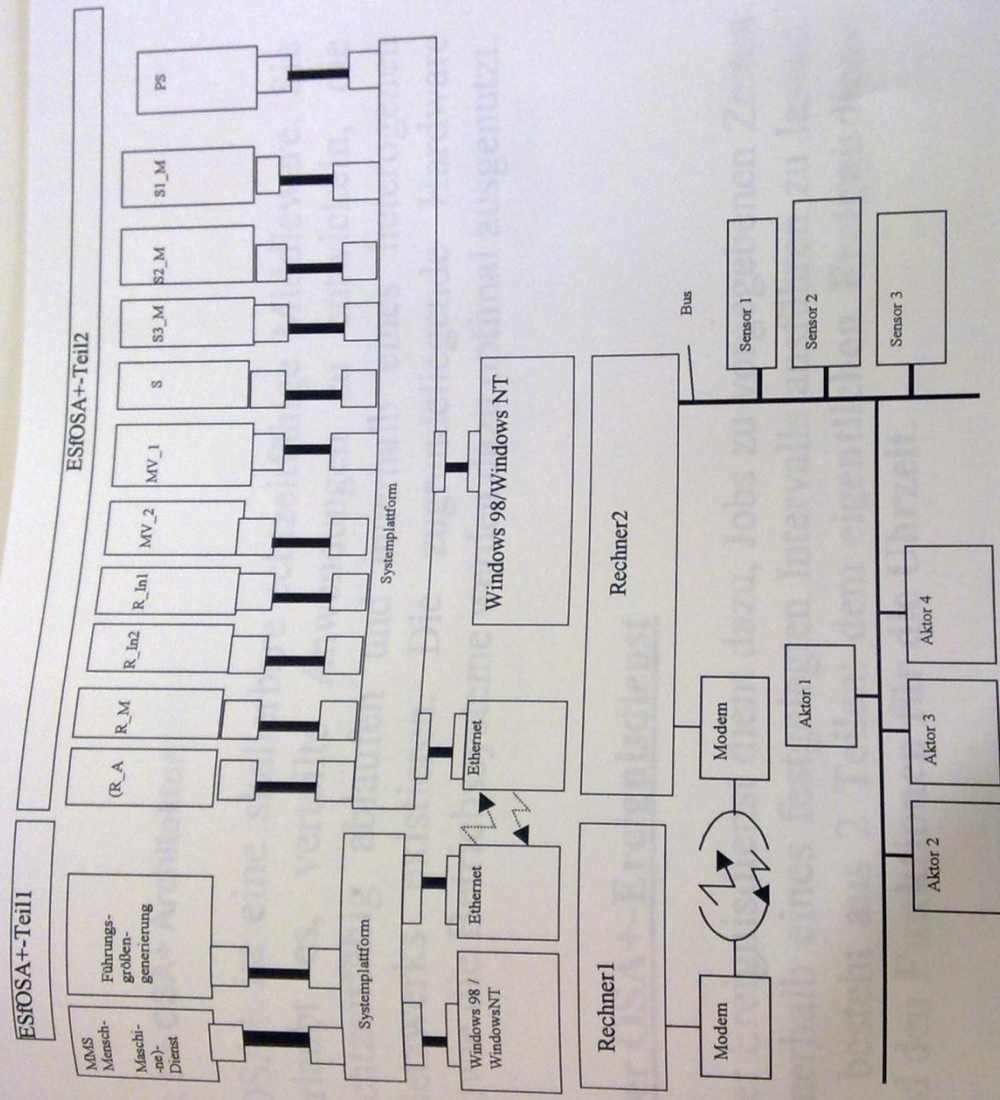


Bild 2: Der digitale Regelkreis

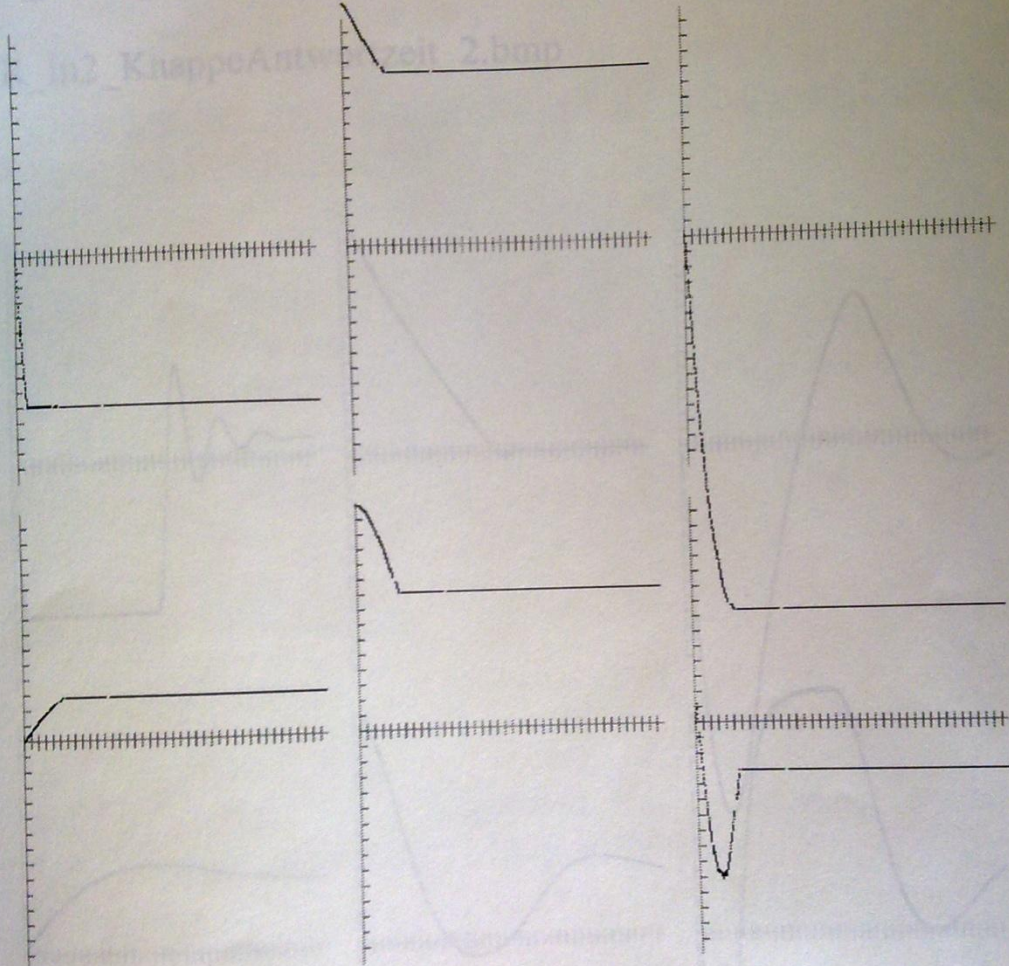




EsfOSA+ soll den in Bild 2.1.1 dargestellten kaskadierten Regelkreis implementieren.

R_In2_KnappeAntwortzeit_1.bmp

Einstellungen wie bei Test 2.1, nur ist die Antwortzeit für R_In1 diesmal 30 ms.

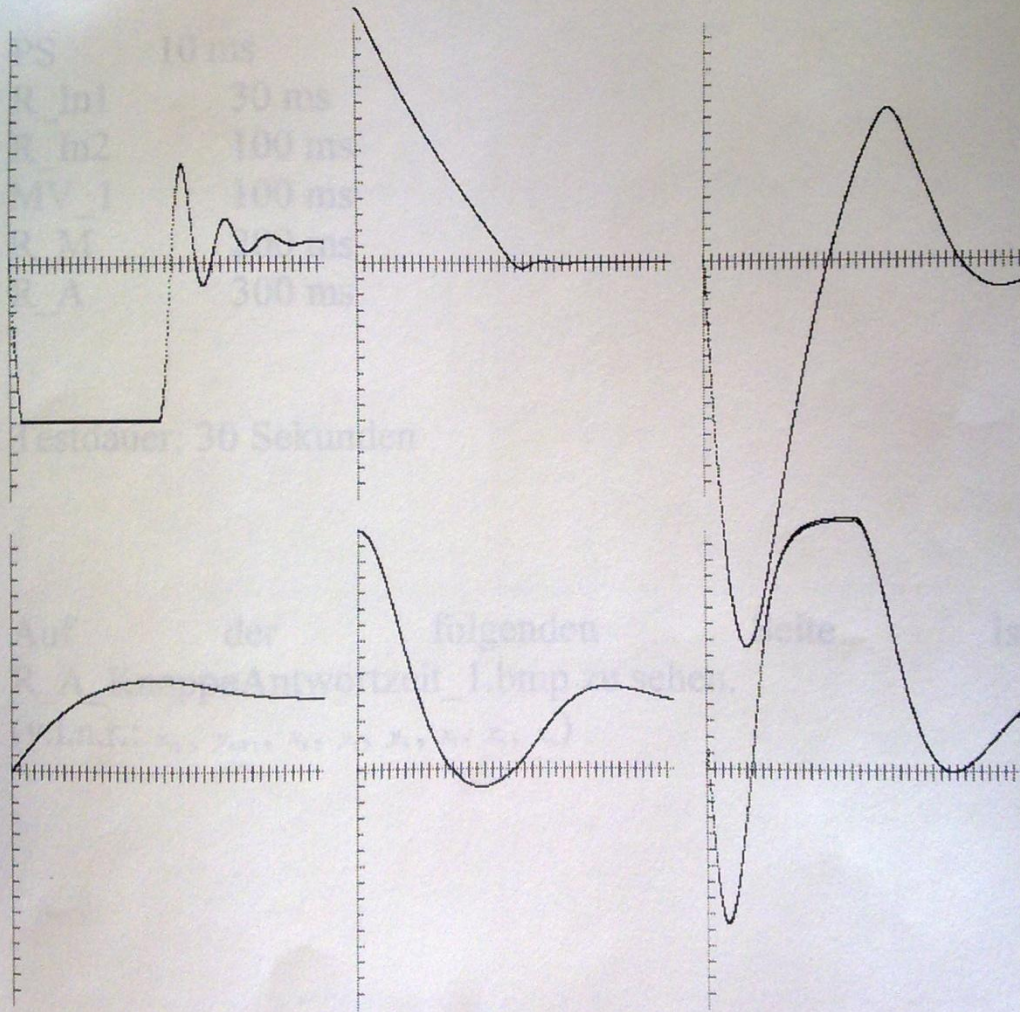


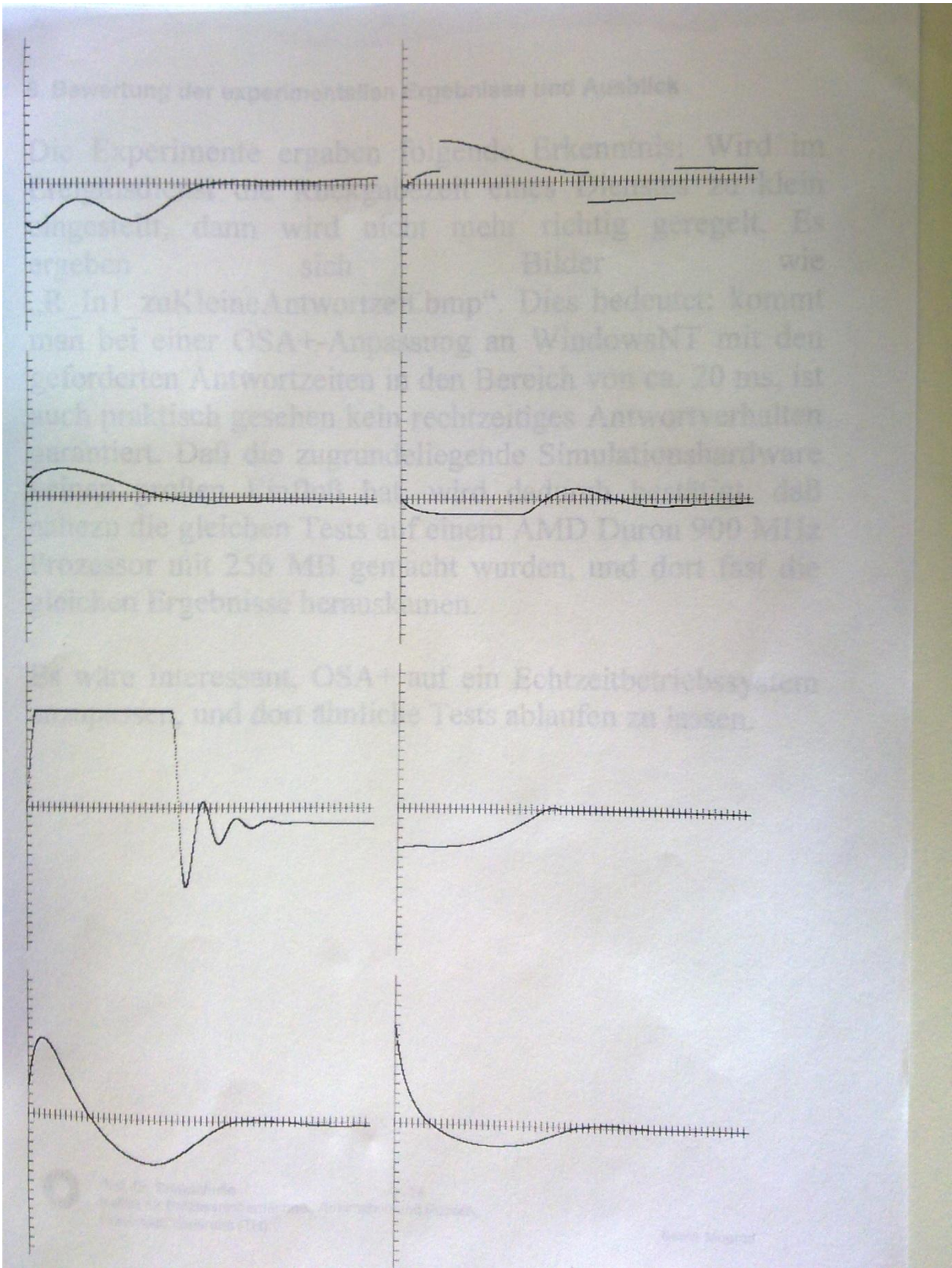
Prof. Dr. Brinkschulte
Institut für Prozessrechenstechnik, Automation und Robotik,
Universität Karlsruhe (TH)

Test 2.2

Einstellungen wie bei Test 2.1, nur ist die Antwortzeit für R_In1 diesmal 30 ms.

R_In2_KnappeAntwortzeit_2.bmp





7 Development of the Ereignisdienstes for the middleware OSA+

Based on *Entwicklung des Ereignisdienstes für die Middleware OSA+*, Studienarbeit von Sven Forstmann, 2001

Studienarbeit (Bachelor Thesis)

of

Sven Forstmann

27 Sep 2001

Institute for Prozeßrechentechnik and automation, Univ. Karlsruhe

Supervisors:

Samir Mourad

Dipl. -Inform. Jens Riemschneider

Prof. Dr. Uwe Brinkschulte

1	ABRIDGED VERSION	82	
2	THE NATURE OF THE TASK.....	82	
3	THE OSA+ ARCHITECTURE	82	
3.1	INTRODUCTION	83	
3.2	THE COMPONENTS OF OSA+.....	84	
3.2.1	<i>The Platform</i>	84	
3.2.2	<i>Generic Services</i>	85	
4	COMPONENTS OF THE OSA EVENTING SERVICE	86	
4.1	THE TIME FUNCTIONS.....	86	
4.1.1	<i>The handling of the time</i>	86	
4.1.2	<i>Initialization</i>	87	
4.1.3	<i>Functions for time management</i>	89	
4.1.3.1 Time Continued	89	99
4.1.3.2 Reading Time	89	99
4.1.3.3 Move Time	90	90
4.1.3.4 Spend time in a string	90	90
4.2	THE EVENT SERVICE.....	90	
4.2.1	<i>General Information</i>	90	
4.2.2	<i>The initialization of the Ereignisdienstes</i>	92	
4.2.3	<i>Features of the Ereignisdienstes</i>	92	
4.2.3.1 Add an Event	92	92
4.2.3.2 Remove a Events	93	93
4.2.3.3 Reading a result/error codes	93	93
5	SHORT OVERVIEW	94	
5.1	TIME FUNCTIONS/VARIABLES	94	
5.2	FEATURES OF THE EREIGNISDIENSTES	94	
6	CONFIGURATION.....	94	
7	THEORY OF OPERATION	96	
7.1	OSAGETTIME()	96	
7.2	OSASETTIME(INT,INT)	96	
7.3	OSAADDTIME(INT,INT)	98	
7.4	CHAR * OSAPRINTTIME(INT, INT).....	98	
7.5	OSAINITEVENTS()	99	
7.6	INT OSAADDEVENT(UINT,UINT, UINT, UINT,UINT, UINT,UINT, UINT, FUNCTION).....	99	
7.7	INT OSADDEVENT(UINT,UINT,UINT,UINT)	101	
7.8	OSA_ERROR OSAGETEVENTRESULT(UINT,UINT)	101	
7.9	INTERNAL FUNCTIONS OF THE EREIGNISDIENSTES	102	
8	PROGRAMMING EXAMPLES	102	
8.1	FOR EXAMPLE: CHANGING THE TIME.....	102	
8.2	EXAMPLE: ADD AN EVENT.....	103	
8.2.1	<i>Start a function at a specified time</i>	103	
8.2.2	<i>Repeated start a function</i>	105	
8.3	QUERIES OF RESULTS.....	106	
8.4	DELETE A EVENTS	106	

9	TEST RUNS.....	107
9.1	DELIVER-TEST.....	109
9.2	TEST RUN OF A CYCLICAL EVENTS WITH OPEN	110
9.3	TEST THE MAXIMUM TEMPORAL RESOLUTION ON DIFFERENT SYSTEMS	113
9.3.1	<i>System 1</i>	113
9.3.2	<i>System 2</i>	116
9.3.3	<i>System 3</i>	118
9.4	STABILITY TESTS	121
9.4.1	<i>Exceeding the maximum acceptable number of Events</i>	121
9.4.2	<i>Exceeding the maximum acceptable number of events per unit time</i>	122
9.4.3	<i>Overrun of the events per unit time through Zeituberschneidung</i>	123
10	RESULTS AND OUTLOOK.....	126
11	ANNEX	127
11.1	LIST OF ALL WINDOWSSPEZIFISCHEN FUNCTIONS	127
12	LITERATURE.....	127

Declaration

I hereby declare, Sven Forstmann, that I have carried out this work independently.

Thanksgiving

First of all, I wish to thank Prof. Brinkschulte for the nice assistance and cooperation and also thank Jens Riemschneider for assistance during the incorporation in OSA+. Samir Mourad was for the friendly help thanked in advance of the thesis.

7.1 Abridged Version

Middleware systems support the development of real-time systems especially in heterogeneous environments, in the present work is the event service with all the details, as well as the Implementationsbeschreibung presented.

The event service extends the existing OSA+ to services for the timed start jobs or functions, as well as to several functions to manage the time.

What is important is that the Ereignisdienstes the OSA+ now one step closer in the direction of the real-time capability brings, this is achieved by, inter alia, that when you start a job / a function also can be specified, in which interval This is to be started, however, the extent to which this real-time capability achieved will depend on the respective operating system, and is also by the handling of the time functions of the same limited.

7.2 The nature of the task

In the framework of a joint project with several industrial partners, the Institute for Prozeßrechentchnik and Automation the open, scalable, and real-time middleware platform OSA+ (Open System Architecture Platform for universal services) developed, today's real-time systems usually operate in environments with many asynchronous events, and the processing of these events is often complex and may, by the introduction of a own service established for this purpose will be much easier. The aim of this thesis is the design of the service, a prototypical implementation and integration into the OSA+ overall architecture.

7.3 The OSA+ architecture

The content of this chapter is [Brinks et al, 00] taken from the description of the individual OSA+ -functions from [Brinks et al, 00] are not listed, it can be read there.

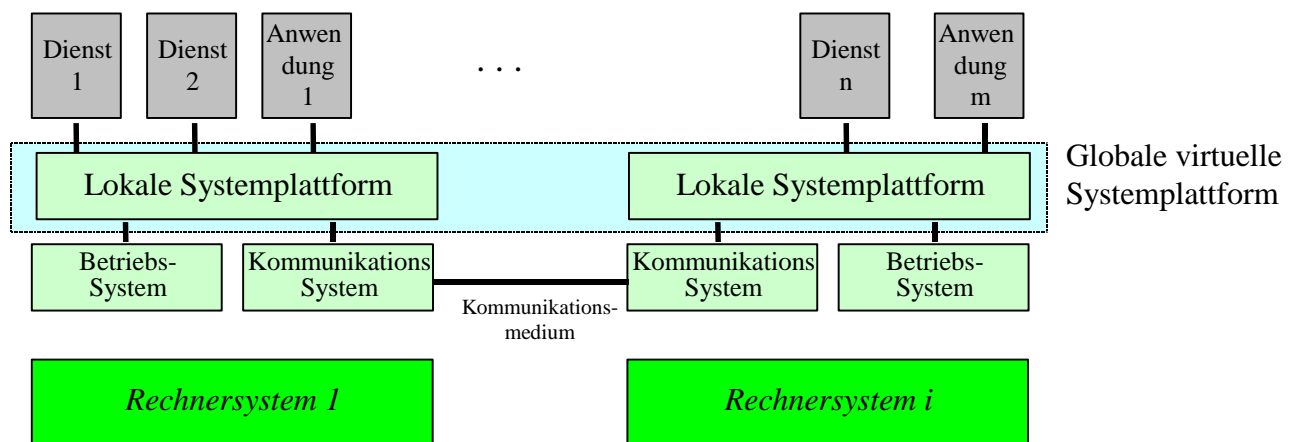
7.3.1 Introduction

OSA+ is a scalable real-time middleware that allows to develop distributed applications, the real timable expire and exist within a heterogeneous network. The underlying hardware, as well as the operating systems to be used optimally.

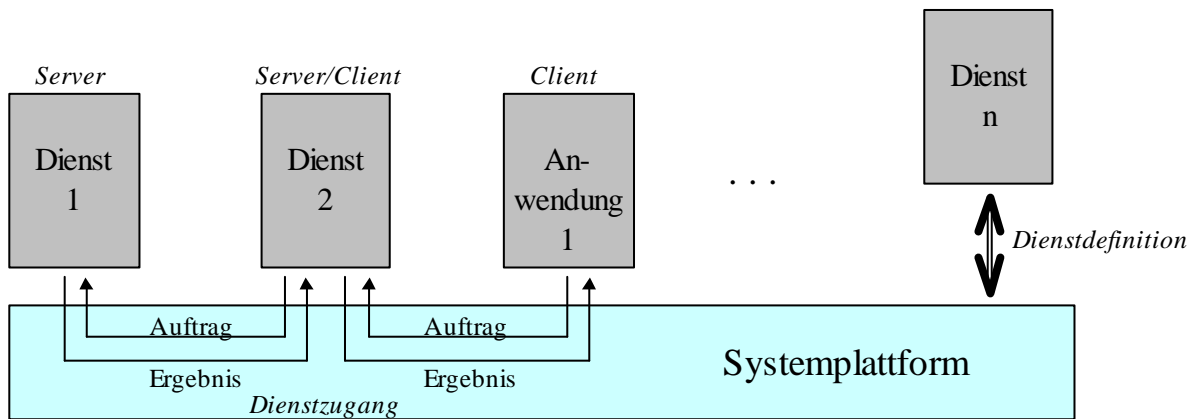
The architecture of OSA+ is divided by a application in services that communicate with each other on jobs, this communication is achieved by means of a platform, in which services in the platform "Plugged in", in this way can inserted into all services with one another using the Platform communicate.

The platform itself specifically used excellent services, to increase its capabilities, including generic services, which ensure that the platform independent of the operating system and the underlying hardware, it is, as well as Erweiterungsdienste, the specific tasks for the platform provide the platform is, however, also run without these services.

The following is the detailed architecture described by the OSA+.



1: OSA+ architecture



2: Communication of services using job and outcome (=job)

First, it is described as the "naked" platform looks like and what functionality it provides, so that you will be the base and Erweiterungsdienste explains and the associated additional possibilities, thus acquires the the platform, and concludes the interfaces of the platform and the basic and Erweiterungsdienste described.

7.3.2 The components of OSA+

7.3.2.1 The Platform

The platform is the component to the user by the OSA+ is visible, it offers a interface, with whose assistance services in the platform can be inserted and allowed the grant of orders, which is divided the platform in three basic parts:

- A **Service Management**: she knows all inserted services and completed the install and uninstall services, as well as locate of services;
- A **job management**: Allows the give and expect of orders and the report of results of this and expect orders;
- A **user interface**: Is there a clear strukturierte the user interface to these two bodies.

7.3.2.2 Generic Services

The platform uses the generic services, to an encapsulation of the operating system functions, which you can use for your tasks are not generic services installed in the platform, so must the platform without these features and can only get to make, what as a recognized standard of the underlying implementation language is recognized, based on the ANSI-Richtlinien for C and C++ or the JDK 1.0 . Beyond functionality of the generic services must be available in each are this:

- **Prozeßdienst:** it allows the platform lightweight and/or heavyweight to take advantage of services, he provides a unified interface to process administration of the operating system, and the OSA+ -Prozeßdienst allows it, split the control flow of the customer, so that the contractor and the customer (quasi) in parallel, each with their Arbeitfortfahren can. This is possible, that of the Prozeßdienst the platform allowed, with a service to connect a Control Flow, heavyweight Kontrollflüsse are situated in their own space and are in need of a Miniplattform, on the A Kmmunikationsdienst (IPC-service) is installed.
- **Echtzeitspeicherdienst:** it allows the realtime access to memory (in particular the allocation and release), not real-time memory is in the above language specifications standardized (e.g. malloc in C or new in C++ or Java). The service is not installed, so the time conditions are checked only a job, but not guaranteed.
- **Communications services:** serve the platform to the shipping of orders over any communication media, it is also the Interprocess Communication as a means of communication.
- **Eventing service:** is the platform for the monitoring and control of temporal processes. He provides specific Auftragspezielle jobs to the platform, so this on certain events can be informed.

7.4 Components of the OSA Eventing service

The event service is divided into the services for the management of the time, as well as in the of the actual Ereignisdienstes. a separation of these two is not impossible, but awkward, as the event service on the functions of the time is dependent on time, to all the processes to start.

7.4.1 The time functions

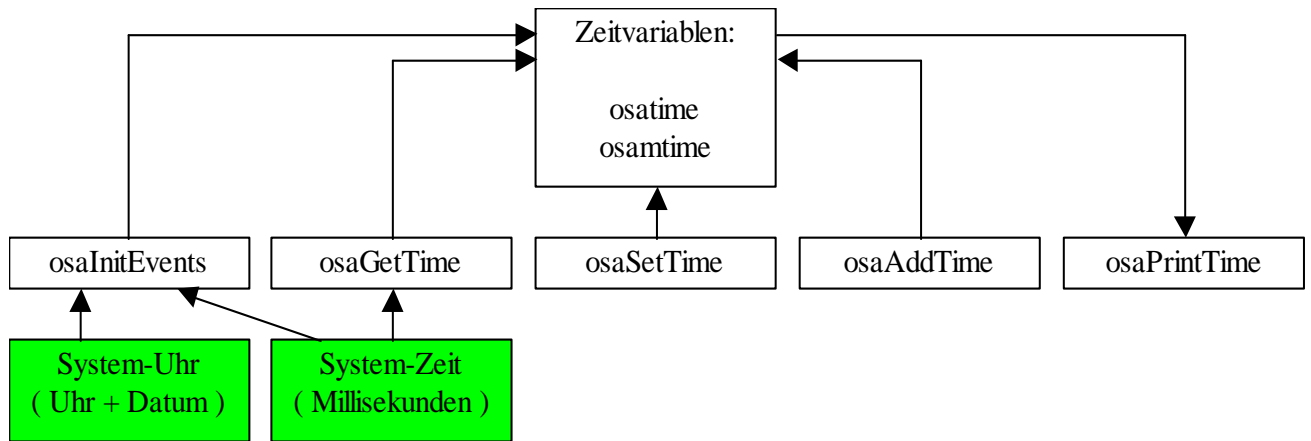
7.4.1.1 The handling of the time

In the event service is managed the time in the UNIX FORMAT, as this is the easiest to handle, and at the same time the implementation on UNIX/LINUX - systems much easier.

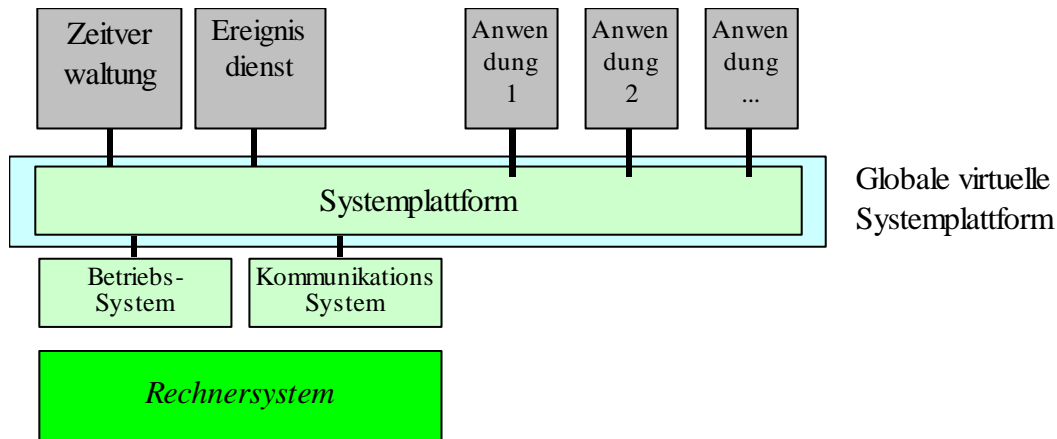
In this format, the seconds since 1.1.1970 12:00:00 in a 32 (soon, perhaps even 64) bit - value paid, which means that extra for Sekunde/Minute/Stunde/ ... not a variable is needed, and also that the invoices are carried out with the time, are now almost easily.

In the OSA-event Service is the time but now not only in a 32-bit value, since there often also in real-time applications to a higher accuracy is essential, therefore it is in addition to the 32-bit variables for the seconds a further variable used for the milliseconds. This makes it difficult and slows down while the calculation of time differences, but it is for the time-exactly timed execution of the job is essential.

It could also have been upgrading the CMOS-clock for all of the operations of the time, as well as to use of the Ereignisdienstes - this had caused some complications, however, first of all it is not accurate enough, because you only on the time the tenth stores, since the event service is to be real timable and there you have a accuracy is promoted in the millisecond range, this is enough and not from, a further difficulty is that when the time change this change affects the entire system and be influenced by other programs, then what in these can create incorrect results.



3: Structure of the time



4: Integration of time management, and in the system of the Ereignisdienstes

7.4.1.2 Initialization

Before the time functions that can be called the initialization must be performed first, which for the time periods required calculated variables.

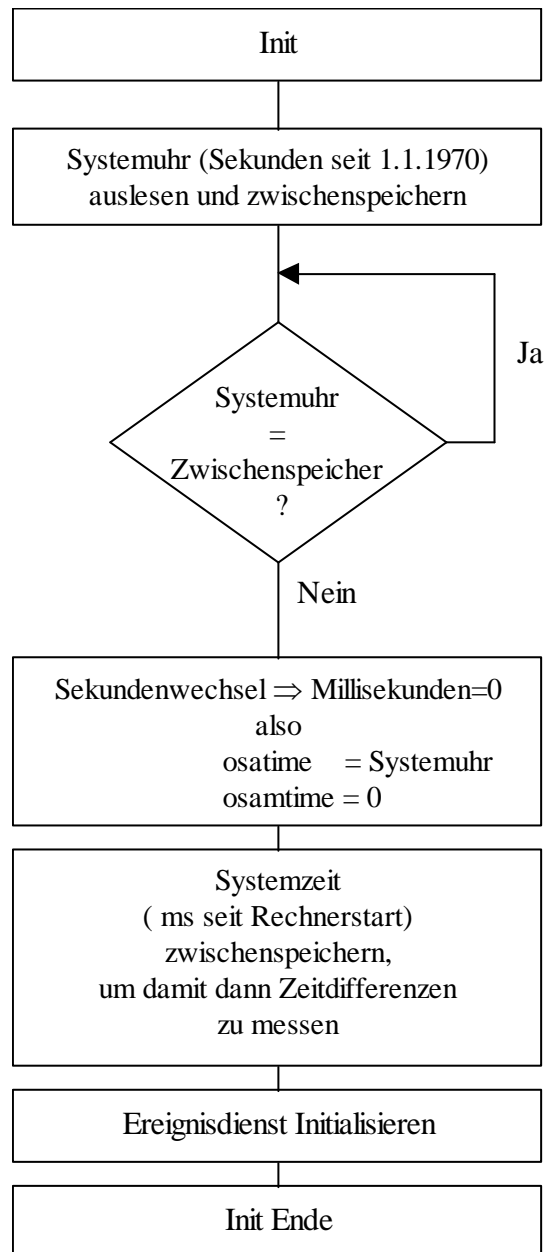
This calculation is necessary, because the time with the help of two different functions is calculated.

With the A is initially read out the time of the system and stored in the time variable, since the system time but only on the second exactly is returned, only to a Sekundenwechsel must be serviced to ensure the Millisekundenanteil can be initialized with zero.

Now, the difference to the value of the other function are calculated, which returns the system time in Millisekundenformat. In this is the number of milliseconds since the start of the system, since you can get with this 32-bit value but only 2^{32}

milliseconds can be used to specify, which corresponds to 49.71 days, you can use this function only to do this, read the time already with the help of this function to update, and is now an offset is calculated at the beginning of the system time and always corresponds to the difference of Millisekundenanteil indicating to system time.

In addition, the initialization of the time and also the Ereignisdienstes starting values for the set, in order to save another function call.



5: PAP of the time initialization

7.4.1.3 Functions for time management

Time Continued

When you set the time, the global time variable updated for seconds, and milliseconds. This also takes into account whether Millisekundenanteile have been specified, the Not in the range 0 to 999. In this case, a carryover for the seconds portion of is calculated, and the Millisekundenanteil corrected accordingly.

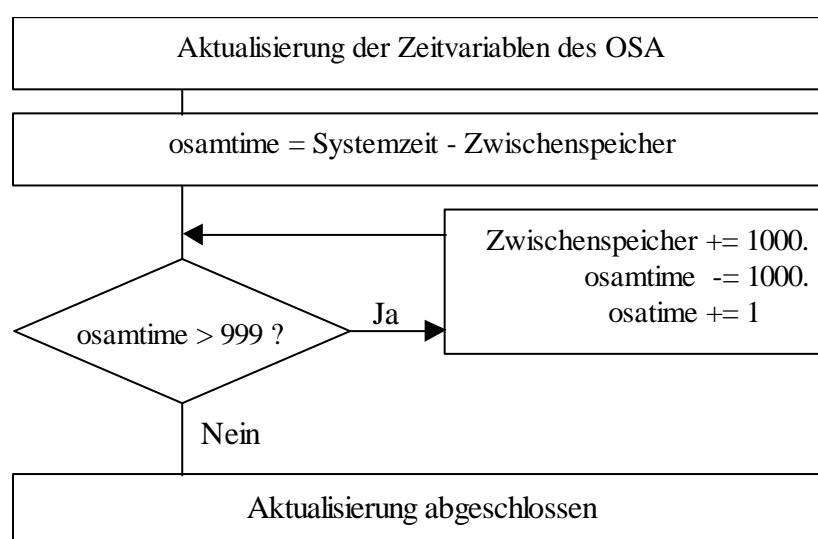
But it is only the time of the OSA+ internally changed - the general CMOS clock of the computer remains untouched.

The time periods necessary to offset the difference between milliseconds of the system and the real time, must also be newly set so that it does not come to the wrong calculations.

Reading Time

When reading the time, the global variable for seconds and milliseconds updated. For this purpose, the difference in milliseconds since the last time the query with the help of the system time calculated and added to the current time.

Since the system time to calculate is used, it should in fact not later than 49 days all the time query be invoked once, but now that the event service is associated with the time, it can be ausgegangen that this due to its timer is called more often, in order to start the events correctly, it must also of course at each time, in the course of which he is called from the timer, the time queries, which is now the reason for this, that the time of an application program does not necessarily have to be called regularly.



Move Time

This function adds to the current time a specified time difference in addition to this, in principle, this function does not have been necessary, since one with the two above functions exactly the same thing can reach had - but so is now saved time, and this may be to match the time on the network can be of use, and the user will work when calculating the carryover disconnected.

Spend time in a string

This function creates a string in which the current date and time is included with milliseconds. (This is especially handy in debug output). This is done by first with the help of the ctime from the current time, in the 32-bit value is stored, generated a string, the date and time already contains, the only thing missing now is the Millisekundenanteil, which simply by various Stringkopieraktionen is inserted.

7.4.2 The event service

7.4.2.1 General Information

The event service is responsible for ensuring that jobs at a specified time with a pre-specified accuracy can be started. He is to guarantee the real-time capability of the OSA, which but limited by the underlying operating system is, for jobs be started via the network, must, however, only the clocks of the two systems are synchronized, this is by a long-term observation of the Pingzeiten possible; as a result of the time offset can then be calculated by which the current time has to be postponed, an alternative would be the average of the Pingzeiten to calculate the last few minutes, and with this to synchronize the clocks.

The event service is from Geschwindigkeitsgunden directly in OSA have been implemented, it is therefore not directly to a LW/HW-service, but rather a supplement to the OSA+ features available.

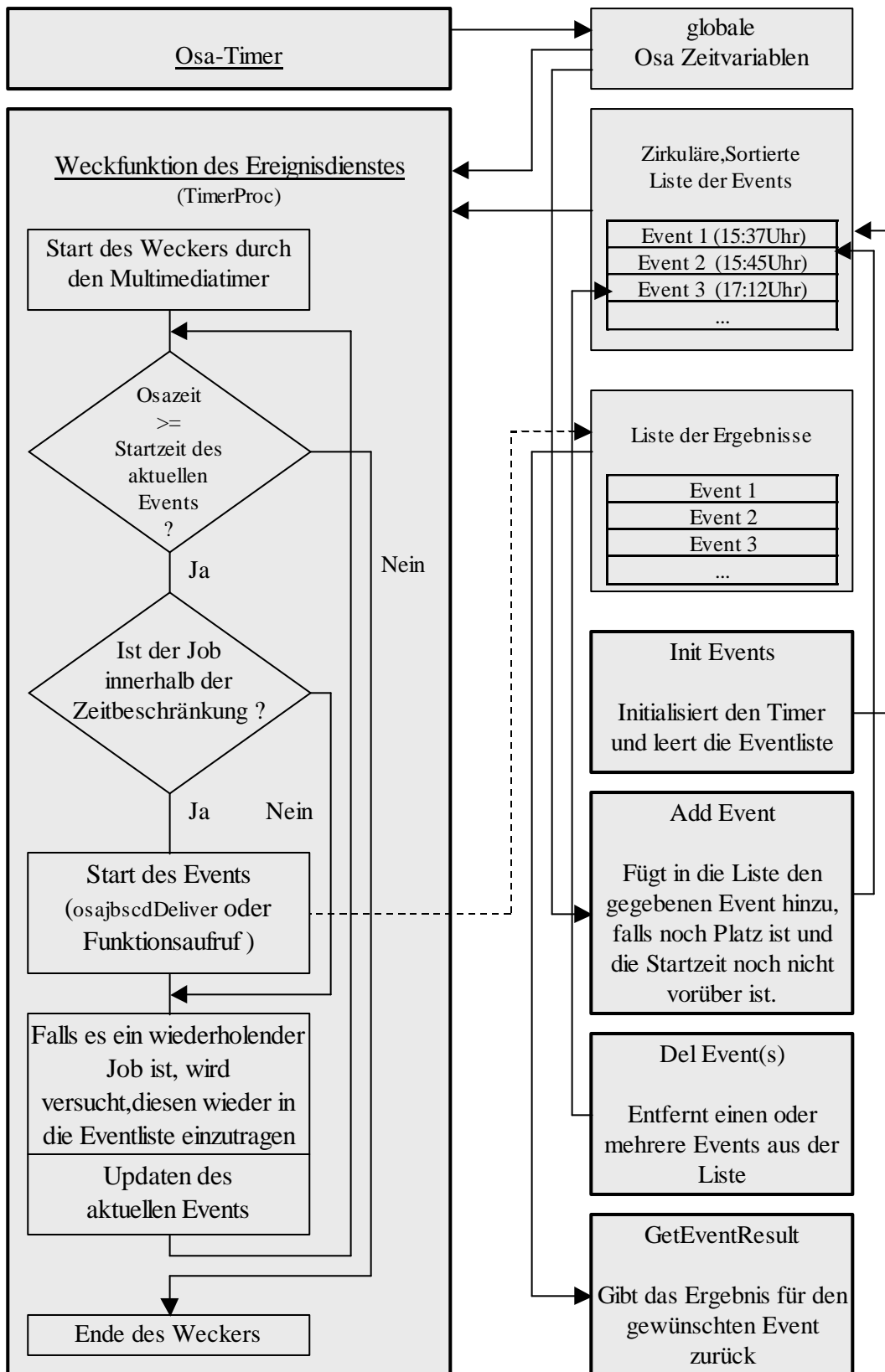


Figure 7: Architecture of the Event Service (Ereignisdienst)

7.4.2.2 The initialization of the Event Service

The initialization of the Ereignisdienstes is automatically after the initialization of the time. This is the first to be the necessary variables for the Eventing service set, the next step is the timer function of the Ereignisdienstes started, what must be called upon regularly in order to make it in time to start the jobs, in which intervals this is to be called, depends on the desired accuracy of the Ereignisdienstes as well as the speed of the underlying system, try showed that, for example on a Duron 900 even an interval of one millisecond is possible.

On a Pentium 100 products, on the other hand, only 5-10 ms intervals possible. (but more about that later in the test runs.

It would also be conceivable, the performance of the Ereignisdienstes to increase that zeitkirtische functions through the implementation of assembly code are replaced.

7.4.2.3 Features of the Ereignisdienstes

Add an Event

When you add new events there are several possibilities: The job can be either at a specific time, or with a relative delay be called, and you can also be used to specify whether the job repeatedly in a certain time interval is to be started (for example, is for the regular queries of sensors helpful), and whether the result of the delivers to the later check is to be saved, and also can be a optional to call a function if it particularly in the short responses to local applications. When you add it can, however, to get any error messages if the job was already over, the list is full, or too many jobs at the same time are to be started, and this number in the same time interval the job to be started is adjustable and should be to the respective system be adapted to maintain speed and in addition must still be given a timeout, the constrains the time window in which the job is to be started.

The job is only called after leaving the window, so is he skipped (this happens but only if it to failures or malfunctions in the system, the calling of the timer function prevent). If it is a repetitive job, it will be the next time but started again, and can still be set, if he is as closely as possible to the Zeitgitter oriented, or the delay as precisely as possible is to be met.

The Add/delete a job may be to delay a few milliseconds, since, as long as the interrupt routine is active, no operations on the list of job to be started are allowed, this of course also applies vice versa for the interrupt routine, if you an insert/delete operation interrupts, this call is ignored, which may of course have the consequence

that a(IGE) Event(s) to a few milliseconds late be started - what but only at very slow computers into the weight cases is expected.

The job that you want to be the Ereignisdienstes in a circular list managed fixed-size, you can, after you have been added to this list, also again be removed - if you have not yet been started.

In order to results of the query again later delivers to, is not yet an additional list managed, in the opened up for each event to an entry exists (the key is the ID of the job), but for reasons of space is only for each EventID keep the latest result.

If the size of the lists is to be changed, is that in the source code simply by changing the #defines in the beginning to reach the files.

In addition, it can not - real-time operating systems that have come to the timer not in time will be called; the result may be that the events do not correctly can be started and is currently are then to be started up to the point that started events at a time, i.e. as long as the timeout of the respective jobs is not exceeded.

Another difficulty arises if a cyclic job has been called for, the back is to be added to the list and an error occurs: a time already past, crowded or list the already too many events to the desired time be called, in order to solve this problem, is, at the moment a part installed, the attempts to the event to matureness, later to start dates.

This will be calculated by either, that either the next possible start times will be taken, or but in the set for the job by increments is attempted, add the job again, the number of retries can be adjusted - but should not be set too high, otherwise if there are too many errors the time interval of the timer not longer sufficient, and the next will be affected, if the rescue but is unsuccessful, the job will no longer be called.

Remove a Events

Back to an event from the table of the job to be started to remove, first waited until requests on the table are allowed and then occupied the semaphore.

The next step will be for the complete table, and skipped the event you want to delete, and the remaining are automatically when.

Reading a result/error codes

Now here is the error code returned, which when you start a job is created, it can here, however, from being purged the last result which emerged only be queried, and internally to a list of all mitzulongenden Events kept, in the then at the start of the event the result will be entered, if there are more events will be logged, as the list is large, after the FI Fo-Prinzip approach.

7.5 Short Overview

7.5.1 Time functions/variables

UINT osatime: Are Using Unixtime in seconds

UINT osamtime: related Millisekundenbruchteil (<1,000)

Osagettime: time reading

Osasettime: Time Set

Osaaddtime: Time-add offset

Osaprinttime: Spend time

7.5.2 Features of the Event Service (Ereignisdienst)

Osainitevents: Initialize event service and Time

Osadelevent: Delete Job

Osaaddevent: Add Job

Osageteventresult: Result of a running job reading

7.6 Configuration

The configuration of the Ereignisdienstes within the source code can be made, and by modifying the various definitions (#define) at the beginning of the source code:

(Alternatively, these parameters from the source code be relocated in the Makefile.

Variable instantaneous value propositions

Precision 1: determines the accuracy with which the event service is supposed to work

EVENTSPERTIME 5 : determines the number of bootable events at the same time

TABLENGTH 255: Indicates the size of the Event-Tabelle (must be a 2^{n-1} number)

ERRORBUFFER 7 : Specifies the size of the Result-Tabelle (must be a 2^{n-1} number)

RETRYCOUNT 4 : specifies how often should be attempted, a recurring event in the

List should be entered, if errors occur

OSA_DEBUG: If defined, debug messages are issued, the

All of the important actions of the Ereignisdienstes describe

To the size of the event, and the table was a 2-he potency as a large selected to the to accelerate for offset calculation for Tabellenzugriffe. (then can namely an and instead of a Modulo-Verkupfung be used - this difference is particularly noticeable on older systems.

7.7 Theory of Operation

7.7.1 `Osagetime()`

Parameters: None

Input integer: NONE

Reads the current time from and updates the global time variable (`osatime` and `osamtime`).

Implementation:

The time is almost to the millisecond, and is calculated from a combination of the internal clock and the system time, the clock will be at the beginning needed to fix the start time in seconds, but not the accuracy is sufficient for OSA as a timer, in that, it not the time to the millisecond that shows exactly, now here comes the system time in the game, the milliseconds since the start of Windows in a 32-bit value, and if you now waits until the clock the seconds count up to 1, then you can use it to calculate a differential value to system time, and with the help of the value they get the missing milliseconds for the time.

If so now `osaGetTime()` is invoked, the milliseconds updated, and at a value greater than 1,000 according to also the seconds, and the difference between this and the system time.

7.7.2 `Osasettime(int,int)`

Parameters: Seconds, Milliseconds

Input integer: NONE

Sets the time seconds is the time in the Unixformat and milliseconds are the associated thousandth.

However, this change also directly to the events that are already in the list are entered, and are to be started; i.e. it quickly in case of major changes can lead to errors.

Implementation:

Here are the global variables are updated; Millisekundenanteile greater than 999 will be automatically converted in seconds.

7.7.3 `Osaaddtime(int,int)`

Parameters:Seconds,Milliseconds

Input integer:NONE

Added to the current time in addition to the time difference of the call-up parameters milliseconds greater than 999 will be converted to the seconds.

(Here are also negative values registered)

Implementation:

This function first reads the time from, and then added the desired time difference, taking into account a Millisekundenuberlaufs, added.

7.7.4 `Char * osaPrintTime(int, int)`

Parameters:Seconds and milliseconds of output time

Input integer:String with the date and time.

This function returns a pointer to a string, the current date and time includes, in debug output is very helpful, since also thousandths of a second to be issued.

Implementation:

The function will access the Function `ctime` back, which from a 32-bit value (Epoch) a string with the date and time calculated. This string will then be modified, so that even milliseconds will be returned.

7.7.5 Osainitevents()

Parameters: None

Input integer: NONE

Initializes the table in which the jobs be entered later and sets the time.

Implementation:

All of the entries will be first in the list of events deleted, and the difference between the internal clock and the system time for `osaGetTime` determined. The next step is now set the timer, later also the starts the job, which is the `Multimediatimer` of Windows, the a function with a definable accuracy can repeatedly. (image 5)5: PAP of the time initialization

7.7.6 `Int osaAddEvent(uint,uint, uint, uint,uint, uint,uint, uint, function)`

Parameters: assigned `id1, NUMCOUNT ID 2, Type, start, MSTART, repeat, mRepeat, timeout, FUNCTION`

Input integer: 0 =OK, 1 =list already full, 2 =too many events to the same Zeit, 3 =time is already over

Assigned `ID1, NUMCOUNT ID 2`: ID's of the job that you want to (`ID1` is the normal ID of a job, `ID2` is

For the future of applications planned)

Type: Bit 0 =0: Start at the specified time (`Start, MSTART`)

Bit 0 =1: Start in `Start.MSTART` Seconds

Bit 1 =0: start only once

Bit 1 =1: repeatedly at intervals of `repeat.Start mRepeat`

Bit 2 =0: return value of not logging `osajbscdDeliver`

Bit 2 =1: return value of `osajbscdDeliver log` for later queries
(costs a little more time.

Bit 3 =0: start times will be respected as precisely as possible

Bit 3 =1:time differences are maintained as closely as possible

Bit 4 =0:When Retries try,the job to start at the soonest Time

Bit 4 =1:In retries the job in time intervals of repeat.Try to start
mRepeat

Start,MSTART:Start at/in Start.MSTART Seconds

Repeat,mRepeat:If in type is specified, the event in repeat.mRepeat seconds called
again

TimeoutGibt the size of the time window, in milliseconds, in which the job is to be
launched

FunctionWenn not zero, the specified function (void function()) called -
osajbscdDeliver is not running in the case

Adds a job to be launched to the Event List in addition to this, the function can be specified the optional, but it is only intended for local use, if short response times are of importance, this feature is this, that you will be called directly, with the same priority as the timer function of the Ereignisdienstes started (on Windows, this is the the highest). You should therefore not be longer than 1-5 ms to use it, as the subsequent jobs this can be affected in some circumstances, since a while(1); in this function, e.g. the system leads to Unbedienbarkeit, this option should go directly to a function can only be used with extreme caution.

Implementation:

In order to prevent the paste that in between the timer function on the Event List is accessing it, is a simple system with semaphores implemented. This checks at the beginning of the paste operation, whether the timer function is currently active and waiting if this is the case, everything is free, the semaphore is occupied, and it can now be checked whether it is possible to add the event, the list is already full, an error will be immediately returned.

If not, now is the time for a relative time into an absolute time converted so that can be compared quickly as possible, whether the job is still is to run, or the time has

already passed, the last thing now to check, is the number of jobs, the at this time are to be started.

Now, if everything is in order, the job is in the Time sorted list is inserted; this prevents that when starting from the correct position must be sought.

7.7.7 Int osaDelEvent(uint,uint,uint,UINT)

Parameters:assigned id1,NUMCOUNT ID 2,start,MSTART

Input integer:number of the deleted events

Deletes one or more events from the list, or if one of the parameters is 0 (up to the Millisekundenanteil), it is not compared; i.e. , osaDelEvent(0,0,0,0) Deletes the entire list, or if the time (start.MSTART) is specified, it is always the absolute time compared (even if the call the relative is specified.

Implementation:

This function runs through the list of events from the front to the rear, and not to delete the copied together events to be deleted with the be skipped, and here again is the system used with semaphores, to avoid conflicts.

7.7.8 OSA_Error osaGetEventResult(uint,UINT)

Parameters:assigned id1, ID2

Input integer:result of the last Delivers

There is the latest result back, at the start of the event with the assigned id1, ID2 has been created.

If OSA_ERR_SERVICE_NOT_FOUND is returned, the job was either not started yet, or the result of change is already have been overwritten.

Implementation:

The list is after the first event with the two ID's being sought, and the returned result of the job in question, and when the event is not found, the function returns OSA_ERR_SERVICE_NOT_FOUND back.

7.7.9 Internal functions of the Ereignisdienstes

Void callback TimerProc(UINT,UINT,DWORD,DWORD,DWORD)

This is the central function of the Ereignisdienstes, which with the Multimediatimer from the desired accuracy of the Ereignisdienstes regularly is called (default are here 2ms). It is responsible for ensuring that the jobs to be started and repetitive jobs again be entered in the list.

Implementation:

At the beginning of the function, it first checks whether the semaphore is busy, and if so, this time interval is skipped; otherwise is now checked, whether the first job needs to be started.

If yes, the next step is still being examined, whether the timeout has not been exceeded and whether a function is to be called, or the job is to be delivered (in the last case, optional, the result will be a list saved).

It is a job to be repeated, again this is in the list, taking into account the different operating modes, it is registered.

7.8 Programming examples

7.8.1 For example: Changing the Time

There are several ways to do this, but one of the simplest is to set the time with the help of the function `osaSetTime` (seconds, milliseconds).

For example, is specified

```
Osasettime( 986604168, 50 )
```

So the clock will be on

Friday the 06.April 2001, 4:42:48 PM set and 50 milliseconds.

However, since this is not too comfortable, and most of the time almost is set correctly, it is often better if the clock is not set to a new, but before or is adjusted, the function is `osaAddTime (seconds, milliseconds)` provided.

So causes for example

```
Osaaddtime ( -3600 , 0 )
```

That the clock one hour is reset.

But you need have no concerns that the time of the system of such actions could be affected - the time is purely OSAintern.

Yet now, in order to check that the Daylight Saving Time was successful, then the time you can also still with the function `osaPrintTime(seconds, milliseconds)` output:

```
Printf(" Date/Time: %s\n", osaPrintTime( osatime , osamtime ));
```

7.8.2 Example: Add an Event

7.8.2.1 Start a function at a specified time

Now here is the first variant customizationsavailable presented:

A unique feature is intended to be with a delay of 5 seconds will be called, is only required once the function that should be called; this has, however, fixed call, and with, here's a little example:

```
OSA_Error test (void)
{
    Printf( "Test OK\n" );
    Return OSA_ERR_OK;
}
```

Now, the Eventing service still be communicated, where he finds the function, and

When he is to call you, and this is done with

```
Osaaddevent ( 1, 0, // assigned id1, ID2
              1, // type ( bit 1 = 1, so start time = delay.
              5, 0, // start in 5 seconds
              0, 0, // time difference when repeats (if set)
              1,000, // 1 second timeout
              &Test // Function
            );
```

As the start time is here now a delay of 5 seconds (+ 0 milliseconds) as well as a timeout of a second is specified at the end of the call must now only the pointer to the function to be called be specified.

A timeout of 1 second means that the function call to a maximum of 1 second may be delayed - for larger delays he will no longer run.

Are Normal delays of 0-10 milliseconds; on fast systems (0-1 milliseconds).

The whole sample program could then may then look like the following:

```
#Include "osa.h" // necessary include - Files
#include "osa_ereignisdienst.h"

OSA_Error test (void)// function that the event Service is Started
{
    Printf( "Test OK\n" ); // as soon as this issue is, the test was successful
    Return OSA_ERR_OK;// everything OK
}
```



```

Main( )// Main Function
{
    // Initialize the Ereignisdienstes
    Osainitevents( );

    // Test-Event entries
    Osaaddevent ( 1.0 , 1, 5.0 , 0.0 , 1000, &test );

    // ... And to the start of the wait dfor
    While(1)sleep(1000);
}

```

7.8.2.2 Repeated start a function

The next example shows how to do a job programd, the will be called repeatedly, which is to the test - function the first time after 5.050 seconds, and then repeatedly at intervals of 1,005 seconds be called, the function is to be called repeatedly, the Eventing service by setting the the 2, Bits communicated in the display type box. The timeout is as well as in the previous example 1 second, and in the display type box the bit 3 is not set, attempts of the Eventing service here, the function exactly as possible to enter the vorherberechneten time - in the present example : Start Time + 5.05 s + X * 1.005 p.

(If the bit would have been set, he had tried, the delays exactly as possible.

The description for the function here is to be called, you can see from the example above, for example, here is the actual function call to the event Service:

```

Osaaddevent ( 1, 0 , // assigned id1, ID2
              1+2 , // type ( delayed start [Bit 1] + repetition [Bit 2] )
              5 , 50, // Start in 5.050 seconds
              1 , 5 ,// repeated Start in 1.005
              1,000 , // Timeout = 1,000 milliseconds
              &Test // Function
            );

```

7.8.3 Queries of results

If a job is started, there is the possibility and subsequently the result query, which is of the type OSA_Error. It plays no role, whether it is a deliver or a direct function call has traded.

```
Osaaddevent (1.0 , 1+8 VDC , F2.0: , 0.0 ,50 , &test); // event in the list entries
Sleep(1000);
Printf( "Eventresult of ID 1 : %d\n", osaGetEventResult(1.0 / * ID of the
event * / ) );
Sleep(2000);
Printf( "Eventresult of ID 1 : %d\n", osaGetEventResult(1.0 / * ID of the
event * / ) );
```

In this example is now the result once before, and the second time after the start of the event read, and if everything is working properly, it is the first time OSA_ERR_SERVICE_NOT_FOUND returned, since the results are not yet available, and the second time then OSA_ERR_OK, what the return value of the test-function corresponds to.

7.8.4 Delete a Events

In the next example will be added first three events, each of which every two seconds will be called once. After 5 seconds, then two of the three will be removed again. The associated program code could appear as follows:

```
Osaaddevent ( 1.0 , 1+2 ), "2, 0, 2.0 ,50, &test1 ) ;// Add the 1 events.
Osaaddevent ( 1.0 , 1+2 ,2,500, 2.0 ,50, &test2 ) ;// Add the 2 events.
Osaaddevent ( 2.0 , 1+2 , 3, 0, 2.0 ,50, &test3 ) ;// Add the 3 events.
Sleep(5000); // wait for 5 seconds ...
Osadelevent ( 1.0 ,// and all with ID 1, * Delete,
0.0 ) ;// without the start times to be taken into account
```

When the program starts, all three events will be initially launched sequentially. After 5 seconds are then the first two of the three events removed from the list, as in the clearing

instructions as the ID 1.0 was specified, what exactly the ID's of the events corresponds to the start times were not taken into account when you delete, since all passed with 0 fields cannot be compared.

7.9 Test Runs

Test Environments

System 1:

Hardware:

CPU:AMD Duron 800

Memory:256 MB

Hard Drive:10GB

Graphics:Nvidia Geforce 2

Operating system:

Windows 2000

System 2:

Hardware:

CPU:Cyrix 6x86

Memory:24MB RAM

Hard Drive:0.5GB

Graphics card: Tseng ET6000

Operating system:

Windows 98

System 3:

Hardware:

CPU: Intel Celeron 450

Memory: 256 MB

Hard Drive: 100GB

Graphics: 3dfx Voodoo 3000

Operating system:

Windows 2000

Configuration of the Ereignisdienstes (if not otherwise specified):

Precision: 2 // The Event-Timer is called every 2 milliseconds

EVENTSPERTIME: 5 // it must not more than 5 events started at the same time

// i.e. there will be maximum 5 events within the 2 ms
started

TABLENGTH: 255 // Number of entries in the Event-Tabelle (2^{n-1} must be)

ERRORBUFFER: 7 // number of jobs, the results keep at the same time

// Should Be (must also be 2^{n-1})

RETRYCOUNT: 4 // Number of retrys if repetitive Job does not immediately
return

// CAN BE ADDED

7.9.1 Deliver-Test

In this test was done with the built-in local modified Serverfunktionstest. The function this is not more testsrvFunc immediately, but only with the delay of 1 second will be called, by using the following changes in the File testsrv.c were made:

```
If (OSA_ERR_OK==osajbscdDeliver(jobId))
{
    ... // Wait for confirmation and result Queries
}
```

Has Been Replaced by the following section

```
Osaaddevent(/ * ID * / jobId,0,
            /* Type */ 1,
            /* Delay */ 1.0 ,
            /* Repeat */ 0.0 ,
            /* Timeout */ 1,000 to
            /* Feature */ /NULL);

If(1)
{
    ... // Wait for confirmation and result Queries
}
```

Now is the result: (with debug mode enabled, to represent the timing of the procedure)

```
+ +> Added ID: 8524656 delay: 1,000 Start: Sun Apr 08 9:48
PM:41,917
--> Started Event 8524656 with error: lms at Sun Apr 08 9:48
PM:41,918
Calculated 1+1
I received a 2!
+ +> Added ID: 8524656 delay: 1,000 Start: Sun Apr 08 9:48
PM:42,920
```

```

--> Started Event 8524656 with error: 0ms at Sun Apr 08 9:48
PM:42,920
Calculated 2+1
I received a 3!
+ +> Added ID: 8524656 delay: 1,000 Start: Sun Apr 08 9:48
PM:43,920
--> Started Event 8524656 with error: 0ms at Sun Apr 08 9:48
PM:43,920
Calculated 3+1
I received a 4!

```

As you can see, the test was successful, and also the temporal error that start when you are born, were within the tolerance: at a set of 2 Accuracy of the Ereignisdienstes milliseconds is a error of 1ms are allowed or not to be excluded.

7.9.2 Test run of a cyclical events with open

In this test is the response of the Ereignisdienstes on longer breaks in the System tested, and the duration of the interruption is in this test here in about 10 seconds, is checked, whether cyclic jobs after this interruption continue to be started correctly. In addition, there will be a demonstration of how the set/do not set of the 3, affects bits to the time difference.

First Test: Bit 3 = 0

Here is the associated function call:

```

Osaaddevent(/ * ID * / 1.0 ,
            /* Type * / 1+2,
            /* Delay * / 2.0 ,
            /* Repeat * / 0.10 ,
            /* Timeout * /10,
            /* Feature * / &test );

```

And the result after the start:

```

+ +> Added id: 1 delay: 9 Start: Sun Apr 22 12:50 PM:28,290
2001
--> Started Event 1 with error: 1ms at Sun Apr 22 12:50
PM:28,291 2001
Test OK
+ +> Added id: 1 delay: 9 Start: Sun Apr 22 12:50 PM:28,300
2001
--> Started Event 1 with error: 1ms at Sun Apr 22 12:50
PM:28,301 2001
Test OK
+ +> Added id: 1 delay: 9 Start: Sun Apr 22 12:50 PM:28,310
2001
--> Started Event 1 with error: 9804ms at Sun Apr 22 12:50
PM:38,114
-> Skipped due to timeout
+ +> Added id: 1 delay: 6 Start: Sun Apr 22 12:50 PM:38,120
2001
--> Started Event 1 with error: 0ms at Sun Apr 22 12:50
PM:38,120 2001
Test OK
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 12:50
PM:38,130 2001
--> Started Event 1 with error: 0ms at Sun Apr 22 12:50
PM:38,130 2001
Test OK
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 12:50 PM:38,140
2001
--> Started Event 1 with error: 0ms at Sun Apr 22 12:50
PM:38,140 2001
Test OK

```

Here you can see now that lasted 10 seconds after the interruption of the job is no longer started, because of the timeout has been exceeded, the time for the next start is now calculated so that the 10ms - Zeitgitter is respected.

Second Test: Bit 3 = 1

Here is the associated function call:

```
Osaaddevent(/ * ID * / 1.0 ,
             /* Type */ 1+2+8,
             /* Delay */ 2.0 ,
             /* Repeat */ 0.10 ,
             /* Timeout */ /10,
             /* Feature */ &test );
```

And the Test Results

```
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 9:36 PM:26,359
2001
--> Started Event 1 with error: 0ms at Sun Apr 22 9:36
PM:26,359 2001
Test OK
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 9:36 PM:26,369
2001
--> Started Event 1 with error: 1ms at Sun Apr 22 9:36
PM:26,370 2001
Test OK
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 9:36 PM:26,380
2001
--> Started Event 1 with error:.. with EN 10204ms at Sun Apr 22
9:36 PM:36,584 2001
-> Skipped due to timeout
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 9:36 PM:36,594
2001
--> Started Event 1 with error: 0ms at Sun Apr 22 9:36
PM:36,594 2001
Test OK
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 9:36 PM:36,604
2001
--> Started Event 1 with error: 1ms at Sun Apr 22 9:36
PM:36,605 2001
Test OK
+ +> Added id: 1 delay: 10 Start: Sun Apr 22 9:36 PM:36,615
2001
--> Started Event 1 with error: 0ms at Sun Apr 22 9:36
PM:36,615 2001
```


Test OK

In this test is now to see that no Zeitgitter more is present, and after the delay of 10 seconds no correction is applied, deleted a part of the calculation, so that this variant a little less processor time.

(However, is expected only on very slow systems make itself felt.

7.9.3 Test the maximum temporal resolution on different systems

7.9.3.1 System 1

Now here is tested in multiple passes, how high the maximum frequency of the calls on this system may be set, without that it comes to errors or problems.

The set of precision timer Ereignisdienstes: 1 millisecond.

Precision:1// The Event-Timer every millisecond is called

Cycle 1:

Set precision of the Ereignisdienstes :1 millisecond (precision= 1)

Set time difference of the Test-Events :1 millisecond.

Associated function call:

```
Osaaddevent( /* ID */      1.0 ,
              /* Type */    1+2,
              /* Delay */   2.0 ,
              /* Repeat */  0.1 ,
              /* Timeout */ /10,
              /* Feature */ &test );
```

And the result of the test:

...

```
+ +> Added id: 1 delay: 1 Start: Mon Apr 23 12:28 PM:36,643 2001
--> Started Event 1 with error: 1ms at Mon Apr 23 12:28 PM:36,644
2001
Test OK
XX> Event 1 emergency added - Time Over
+ +> Added id: 1 delay: 2 Start: Mon Apr 23 12:28 PM:36,646 2001
--> Started Event 1 with error: 0ms at Mon Apr 23 12:28 PM:36,646
2001
Test OK
+ +> Added id: 1 delay: 1 Start: Mon Apr 23 12:28 PM:36,647 2001
--> Started Event 1 with error: 0ms at Mon Apr 23 12:28 PM:36,647
2001
Test OK
...
+ +> Added id: 1 delay: 1 Start: Mon Apr 23 12:28 PM:36,652 2001
--> Started Event 1 with error: 1ms at Mon Apr 23 12:28 PM:36,653
2001
Test OK
XX> Event 1 emergency added - Time Over
+ +> Added id: 1 delay: 2 Start: Mon Apr 23 12:28 PM:36,655 2001
--> Started Event 1 with error: 0ms at Mon Apr 23 12:28 PM:36,655
2001
Test OK
+ +> Added id: 1 delay: 1 Start: Mon Apr 23 12:28 PM:36,656 2001
--> Started Event 1 with error: 0ms at Mon Apr 23 12:28 PM:36,656
2001
Test OK
...
```

As you can see, does the start mostly, but it is out and back to errors, to now is the result of such a high request to improve the accuracy, would be the first consideration, to configure the job in such a way as to that of the Eventing service always the interval of one millisecond complies with with these settings, the test was not, however, the desired result, but there were still the same error.

The real problem here is that the performance was not sufficient, to all text output to be carried out within a millisecond after the text output were omitted in part, the test was almost flawless. There were still error on, but not more in 10ms increments, but only every

few seconds; also the misconduct has improved: were it to the top 2 more failures per error, it is the case in the following test run only has one:

```
...
--> Started Event 1 with error: 0ms at Mon Apr 23 3:54 PM:29,198
2001
--> Started Event 1 with error: 0ms at Mon Apr 23 3:54 PM:29,199
2001
--> Started Event 1 with error: 0ms at Mon Apr 23 3:54 PM:29,200
2001
--> Started Event 1 with error: 0ms at Mon Apr 23 3:54 PM:29,201
2001
--> Started Event 1 with error: 0ms at Mon Apr 23 3:54 PM:29,202
2001
--> Started Event 1 with error: 0ms at Mon Apr 23 3:54 PM:29,203
2001
--> Started Event 1 with error: 1ms at Mon Apr 23 3:54 PM:29,205
2001
XX> Event 1 emergency added - Time Over
--> Started Event 1 with error: 0ms at Mon Apr 23 3:54 PM:29,207
2001
...
```

Cycle 2:

Set precision of the Ereignisdienstes :1 millisecond (precision= 1)

Set time difference of the Test-Events :2 millisecond.

Associated function call:

```
Osaaddevent( /* ID */      1.0 ,
              /* Type */    1+2,
              /* Delay */    2.0 ,
              /* Repeat */   0.2 ,
              /* Timeout */ /10,
```

```
/* Feature */ &test );
```

And the test results:

```
...
--> Started Event 1 with error: 0ms at Thu Apr 26 12:12:42,310
2001
+ +> Added id: 1 delay: 2          Start: Thu Apr 26 12:12:42,312
2001
--> Started Event 1 with error: 0ms at Thu Apr 26 12:12:42,312
2001
+ +> Added id: 1 delay: 2 Start: Thu Apr 26 12:12:42,314 2001
--> Started Event 1 with error: 0ms at Thu Apr 26 12:12:42,314
2001
+ +> Added id: 1 delay: 2 Start: Thu Apr 26 12:12:42,316 2001
--> Started Event 1 with error: 0ms at Thu Apr 26 12:12:42,316
2001
...
```

The test was properly here, without that it came to any errors, i.e. , when working with text files, at least on this system should be a time difference of 2 ms be between 2 Jobs

7.9.3.2 System 2

Here is to be tested, as the replacement of the operating system from Windows 2000 to Windows 98 is noticeable.

Cycle 1:

Set the precision Ereignisdienstes :10 milliseconds (precision= 10)

Set time difference of the Test-Events :100 milliseconds

Function call:

```
Osaaddevent( /* ID */      1.0,
              /* Type */    1+2,
```

```
/* Delay */ 2.0 ,
/* Repeat */ 0.100 ,// cyclic call every 100 ms
/* Timeout */ /10,
/* Feature */ &test );
```

Test run:

In this test Sorry, no results could be collected, as the operating system is either refused the Test (The Multimediatimer has not been activated), or, if this time succeeded in, worked behind the exit and save the logs not, because the system remained at once.

Cycle 2:

Set of precision timer of the Ereignisdienstes: 100 milliseconds

Function call:

```
Osaaddevent( /* ID */ 1.0 ,
/* Type */ 1+2,
/* Delay */ /2.0 ,
/* Repeat */ 1.0 ,
/* Timeout */ /10,
/* Feature */ &test );
```

Test run:

...

```
+ +> Added id: 1 delay: involving 994 Start: Mon Apr 24 11:54
PM:07,000 1995
```

```
--> Started Event 1 with error: 6ms at Mon Apr 24 11:54 PM:07,006
1995
```

```
+ +> Added id: 1 delay: involving 994 Start: Mon Apr 24 11:54
PM:08,000 1995
```

```

--> Started Event 1 with error: Note No. 4465ms at Mon Apr 24
11:54 PM:12,465 1995// done manually delay
  -> Skipped due to timeout
+ +> Added id: 1 delay: experienced 535 recorded Start: Mon Apr
24 11:54 PM:13,000 1995
--> Started Event 1 with error: 6ms at Mon Apr 24 11:54 PM:13,006
1995
+ +> Added id: 1 delay: involving 994 Start: Mon Apr 24 11:54
PM:than 14,000 1995
Started Event 1 with error: 1ms at Mon Apr 24 11:54 PM
...

```

The result shows that it is still possible, on a Windows 98 machine with the OSA to run event service, but may not be a high claims be made to the accuracy!

7.9.3.3 System 3

As there is a on Windows NT-based operating system present, can already at the beginning with a higher accuracy than in 9.3.2 7.9.3.2 be started.

Cycle 1:

Set precision of the Ereignisdienstes :1 millisecond (precision= 1)

Set time difference of the Test-Events :1 millisecond.

Function call:

```

Osaaddevent( /* ID */      1.0 ,
             /* Type */    1+2,
             /* Delay */    2.0 ,
             /* Repeat */   0.100 ,// cyclic call every 100 ms
             /* Timeout */ /10,
             /* Feature */ &test );

```

Test run:

```
...
--> Started Event 1 with error: 1ms at Wed May 11 11:28:43,584
2001
-> Skipped due to timeout
XX> Event 1 emergency added - Time Over
+ +> Added id: 1 delay: 1 Start: Wed May 11 11:28:43,585 2001
--> Started Event 1 with error: 1ms at Wed May 11 11:28:43,586
2001
-> Skipped due to timeout
XX> Event 1 emergency added - Time Over
+ +> Added id: 1 delay: 1 Start: Wed May 11 11:28:43,587 2001
--> Started Event 1 with error: 1ms at Wed May 11 11:28:43,588
2001
-> Skipped due to timeout
XX> Event 1 emergency added - Time Over
+ +> Added id: 1 delay: 1 Start: Wed May 11 11:28:43,589 2001
--> Started Event 1 with error: 2ms at Wed May 11 11:28:43,591
2001
...
```

For this test it is clear that here, in this system even more errors have occurred, as in the same test on the first system. In a further test run however, again, as in 9.3.1 NFS Server Configuration file 7.9.3.1, that mainly blame the text output is to the late call, the same test with only a text per call will be as follows:

```
...
--> Started Event 1 with error: 0ms at Wed May 23 15:18:28,271
2001
--> Started Event 1 with error: 0ms at Wed May 23 15:18:28,272
2001
--> Started Event 1 with error: 0ms at Wed May 23 15:18:28,273
2001
--> Started Event 1 with error: 0ms at Wed May 23 15:18:28,274
2001
--> Started Event 1 with error: 0ms at Wed May 23 15:18:28,275
2001
```

...

So a much better result (Timeouts came even before, however very much less than previously). This is also the reason why should also sub-programs, the of of the timer function in so called short intervals are not too to take a lot of time.

Cycle 2:

In the following test was now the Aufruffrequenz halved, to test whether this time the attempt without Timeouts runs; the test environment is as follows:

Set precision of the Ereignisdienstes :1 millisecond (precision= 1)

Set time difference of the Test-Events :2 milliseconds

Function call:

```
Osaaddevent( /* ID */      1.0 ,
              /* Type */    1+2,
              /* Delay */   2.0 ,
              /* Repeat */  0.100 ,// cyclic call every 100 ms
              /* Timeout */ /10,
              /* Feature */ &test );
```

Test run:

```
--> Started Event 1 with error: 0ms at Wed May 23 3:47 PM:34,566
2001
--> Started Event 1 with error: 0ms at Wed May 23 3:47 PM:34,568
2001
--> Started Event 1 with error: 1ms at Wed May 23 3:47 PM:34,571
2001
--> Started Event 1 with error: 0ms at Wed May 23 3:47 PM:34,572
2001
```



```
--> Started Event 1 with error: Oms at Wed May 23 3:47 PM:34,574
2001
```

Here the test was now easily, although there were also late calls, but this had to literally be sought with the magnifying glass, in the absence of such an then time occurred, it wasn't too bad, because this delay is not Timerout led immediately to a.

7.9.4 Stability tests

7.9.4.1 Exceeding the maximum acceptable number of Events

For this test was the maximum allowable number of 255 entries now intentionally exceeded, to test the risk of errors:

Function call:

```
For(i= 0 ;i< 500 ;i++)
Osaaddevent( /* ID */      i .0,
             /* Type */    1 + 2,
             /* Delay */    i + 2.0 ,
             /* Repeat */   256.0 ,
             /* Timeout */ /100,
             /* Feature */ &test );
```

Test run:

```
+ +> Added id: 0 delay: 2000 Start: Wed May 16 14:30:59,050 2001
+ +> Added id: 1 delay: 3,000 Start: Wed May 16 2:31 PM:00,050
2001
+ +> Added id: 2 delay: 4,000 Start: Wed May 16 2:31 PM:01,050
2001
...
+ +> Added ID: 252 delay: 254000 Start: Wed May 16 2:35
PM:11,050
+ +> Added ID: 253 delay: EUR 255000 per year Start: Wed May 16
2:35 PM:12,050
+ +> Added ID: 254 delay: The value 256000 Start: Wed May 16 2:35 PM:13,050
```

```

XX> Event 255 not added - table full
XX> Event 256 not added - table full
...
XX> Event 498m not added - table full
XX> Event 499 or fewer not added - table full
--> Started Event 0 with error: 0ms at Wed May 16 14:30:59,050
2001
+ +> Added id: 0 delay: The value 256000 Start: Wed May 16 2:35
PM:15,050 2001
--> Started Event 1 with error: 0ms at Wed May 16 2:31 PM:00,050
2001
+ +> Added id: 1 delay: The value 256000 Start: Wed May 16 2:35
PM:16,050 2001
--> Started Event 2 with error: 0ms at Wed May 16 2:31 PM:01,050
200
...

```

As you can see, the test run was successful, and there was no further difficulties

7.9.4.2 Exceeding the maximum acceptable number of events per unit time

It was here that the set from the beginning, the maximum number of 5 Events / Unit time maintained. An attempt is now to exceed this limit, a small difficulty with this attempt is that the add of the 10 events no longer than a millisecond should be allowed to take. This was the reason for this and the next attempt the test system 1 is selected.

Function call:

```

For(i= 0 ;i< 10 ;i++)
Osaaddevent( /* ID */      i .0,
             /* Type */      1 ,
             /* Delay */     1.0 ,
             /* Repeat */    0.0 ,
             /* Timeout */ /100,
             /* Feature */ / &test );

```

Test run: (System (1)

```

+ +> Added id: 0 delay: 1,000 Start: Wed May 15 3:22 PM:08,072
2001
+ +> Added id: 1 delay: 1,000 Start: Wed May 15 3:22 PM:08,072
2001
+ +> Added id: 2 delay: 1,000 Start: Wed May 15 3:22 PM:08,072
2001
+ +> Added id: 3 delay: 1,000 Start: Wed May 15 3:22 PM:08,072
2001
+ +> Added id: 4 delay: 1,000 Start: Wed May 15 3:22 PM:08,072
2001
XX> Event 5 not added - too many events per Time
XX> Event 6 not added - too many events per Time
XX> Event 7 not added - too many events per Time
XX> Event 8 not added - too many events per Time
XX> Event 9 not added - too many events per Time
--> Started Event 0 with error: 0ms at Wed May 15 3:22
PM:08,072 2001
--> Started Event 1 with error: 0ms at Wed May 15 3:22
PM:08,072 2001
--> Started Event 2 with error: 0ms at Wed May 15 3:22
PM:08,072 2001
--> Started Event 3 with error: 0ms at Wed May 15 3:22
PM:08,072 2001
--> Started Event 4 with error: 0ms at Wed May 15 3:22
PM:08,072 2001

```

7.9.4.3 Overrun of the events per unit time through Zeitüberschneidung

This test is similar in principle to attempt from 9.4.2 . above, however, the event service the overlap of the events do not immediately realize. The events are added so that you are all the events until a later date to a time overlap. Here is the behavior of the Ereignisdienstes tested to this situation and will be demonstrated, this is, of course, also the set / do not set of the 4, bits in the Typenfeldes depending on, here is the setting so that, if an error arises during the adding, then simply the closest possible time is selected as a start time (bit 4 is not set). In this example, the minimum delay between the various earliest starts 1 millisecond.7.9.4.2

Since the test this time a bit longer, it was the result of the test for the better understanding still comments.

Function call:

```

For(i= 0 ;i< 10 ;i++)
Osaaddevent( /* ID */      i.0,
             /* Type */    1+2,
             /* Delay */    1+ i,0,
             /* Repeat */   10- i,0,
             /* Timeout */ /100,
             /* Feature */ &test );

```

Test run: (on System 1.

1 Step: Add the 10 Events

```

+ +> Added id: 0 delay: 1,000 Start: Tue May 22 11:48 am:42,200
2001
+ +> Added id: 1 delay: 2000 Start: Tue May 22 11:48 am:43,200
2001
...
+ +> Added id: 8 delay: 9,000 Start: Tue May 22 11:48 am:50,200
2001
+ +> Added id: 9 delay: 10,000 Start: Tue May 22 11:48 am:51,200
2001

```

2 Step:because it is cyclic events, you will be
Re-added

```

--> Started Event 0 with error: 0ms at Tue May 22 11:48 am:42,200
2001
+ +> Added id: 0 delay: 10,000 Start: Tue May 22 11:48 am:52,200
2001
--> Started Event 1 with error: 0ms at Tue May 22 11:48 am:43,200
2001
+ +> Added id: 1 delay: 9,000 Start: Tue May 22 11:48 am:52,200
2001
--> Started Event 2 with error: 0ms at Tue May 22 11:48 am:44,200
2001
+ +> Added id: 2 delay: 8,000 Start: Tue May 22 11:48 am:52,200
2001
--> Started Event 3 with error: 0ms at Tue May 22 11:48 am:45,200
2001

```

```
+ +> Added id: 3 delay: 7,000 Start: Tue May 22 11:48 am:52,200
2001
--> Started Event 4 with error: 0ms at Tue May 22 11:48
am:46,200 2001
+ +> Added id: 4 delay: 6,000 Start: Tue May 22 11:48 am:52,200
2001
```

But since only 5 events/unit (here 1 millisecond) may be called, the remaining 5 to be launched later.

```
--> Started Event 5 with error: 0ms at Tue May 22 11:48 am:47,200
2001
XX> Event 5 not added - too many events per Time
+ +> Added id: 5 delay: 5,001 Start: Tue May 22 11:48 am:52,201
2001
--> Started Event 6 with error: 0ms at Tue May 22 11:48 am:48,200
2001
XX> Event 6 not added - too many events per Time
+ +> Added id: 6 delay: 4001 Start: Tue May 22 11:48 am:52,201
2001
--> Started Event 7 with error: 0ms at Tue May 22 11:48 am:49,200
2001
XX> Event 7 not added - too many events per Time
+ +> Added id: 7 delay: 3001 Focal point Start: Tue May 22 11:48
am:52,201 2001
--> Started Event 8 with error: 0ms at Tue May 22 11:48 am:50,200
2001
XX> Event 8 not added - too many events per Time
+ +> Added id: 8 delay: 2001 Start: Tue May 22 11:48 am:52,201
2001
--> Started Event 9 with error: 0ms at Tue May 22 11:48 am:51,200
2001
XX> Event 9 not added - too many events per Time
+ +> Added id: 9 delay: 1001 Start: Tue May 22 11:48 am:52,201
2001
```

3. Step:Here you just added the starts of the events, only to have a better overview the rows have been shown, in which the events are started.

```
--> Started Event 0 with error: 0ms at Tue May 22 11:48 am:52,200
2001
--> Started Event 1 with error: 0ms at Tue May 22 11:48 am:52,200
2001
--> Started Event 2 with error: 0ms at Tue May 22 11:48 am:52,200
2001
--> Started Event 3 with error: 0ms at Tue May 22 11:48 am:52,200
2001
--> Started Event 4 with error: 0ms at Tue May 22 11:48 am:52,200
2001
--> Started Event 5 with error: 0ms at Tue May 22 11:48 am:52,201
2001
--> Started Event 6 with error: 0ms at Tue May 22 11:48 am:52,201
2001
--> Started Event 7 with error: 0ms at Tue May 22 11:48 am:52,201
2001
--> Started Event 8 with error: 0ms at Tue May 22 11:48 am:52,201
2001
--> Started Event 9 with error: 0ms at Tue May 22 11:48 am:52,201
2001
```

7.10 Results and Outlook

By the enlargement to the event service is now the OSA+ a further major step in the direction toward real-time capability.

This makes it possible that now jobs or functions in an adjustable time in an adjustable time window can be started.

The maximum achievable accuracy is, however, limited by the underlying operating system on Windows 95 testing e.g. in the best case only 50 milliseconds, while under Windows 2000/NT even accuracies of up to 1-2 milliseconds were possible.

If necessary, you can also the jobs started in the same intervals be repeated.

This relatively high accuracy in Windows2000/NT is achieved by the fact that the event service functions to read/set that provides the time, which only during the initialisation of the C use the Standardzeitfunktion. Later, the time, then determined with the help of the system time, which is much more accurate.

This OSA - internal time is also necessary, since there may be times on the network need to be synchronized, and not the CMOS - clock of the system is to be affected.

If you have higher levels of accuracy should be needed, it is recommended that you the porting of the OSA+ on a Echtzeitplattform such as VxWorks or RT-Linux.

Then there are up to the factor of 10 higher accuracy possible and 100% guaranteed real-time capability.

7.11 ANNEX

7.11.1 List of all Windows specific functions

Here is a list of the Windows 95 / 98 / ME/NT/2000 specific functions in the porting to a different platform need to be replaced:

Timegettime()

Reads the current system time from (milliseconds since startup) and there is this as a 32-bit integer value back. (due to the 32-bit, the value is 49.71 days all resettiert)

Timesetevent(PRECISION,1, TimerProc , 0, TIME_PERIODIC);

This function is responsible for ensuring that the Mutimediatimer of the Windows is initialized and then in definable intervals (here:precision) the timer function (TimerProc) invokes the timer function has on the basis of the fixed parameters therefore look like the following:

Void callback TimerProc(UINT uid, UINT uMsg, DWORD dwUser, DWORD DW1, DWORD DW2);

The Windowsspezifischen variables are passed but not evaluated further.

(What the implementation should facilitate to other platforms.

7.12 Literature

[Brinks et al, 00]Brinkschulte, Krakowski, Riemschneider, "The OSA+ architecture", Internal Report, Institute for microcomputer and automation, Uni-Karlsruhe , 28 March 2000)

8 Inertial Measurement Unit

Based on Mohammad Subhan, "Building a inertial measurement system for an experimental environment", Diploma Thesis, 2001/2002, Karlsruhe University of Applied Sciences

University of Technology

Table of Contents

TABLE OF CONTENTS	128
LIST OF FIGURES:	130
LIST OF TABLES	132
LIST OF TABLES	132
1 INTRODUCTION	133
1.1 TASK	133
1.2 OVERVIEW	134
2 BASICS	135
2.1 COORDINATE SYSTEM	136
2.2 CALCULATION OF THE CURRENT POSITION	137
2.3 COMPASS	138
3 ARCHITECTURE DESIGN OF THE IMU	139
3.1 MEßDATENAUFNAHME	139
3.2 MEßDATENAUFBEREITUNG	FEHLER! TEXTMARKE NICHT DEFINIERT.
4 DEVELOPMENT ENVIRONMENT	141
4.1 HARDWARE-ENTWICKLUNGSUMGEBUNG	141
4.2 SOFTWARE DEVELOPMENT ENVIRONMENT	142
5 REALIZATION OF THE IMU	143
5.1 MEßDATENAUFNAHME	143
5.1.1 <i>Circuit Design for the Meßdatenaufnahme</i>	143
5.1.1.1 Acceleration Sensors	143
5.1.1.1.1	Rather than focusing solely on the accelerometer
Fehler! Textmarke nicht definiert.	
5.1.1.1.2	Laying down the bandwidth of the accelerometer
145	

5.1.1.1.3	Definition of the period duration of the accelerometer	
145		
5.1.1.2	The roundabout and the filtering their output signals	146
5.1.1.2.1	Calculation of the order Butterworth	
147		
5.1.1.3	The Kompaßsensoren and filtering their output signals	148
5.1.1.4	Reference Voltage	150
5.1.2	<i>Board layout design for the Meßdatenaufnahme</i>	151
5.2	MEßDATENAUFBEREITUNG	153
5.2.1	<i>Circuit Design for the Meßdatenaufbereitung</i>	153
5.2.1.1	Supply Voltage	153
5.2.1.2	Reference Voltage	153
5.2.1.3	Memory	153
5.2.1.4	Jumper Settings	155
5.2.2	<i>Board layout design for the Meßdatenaufbereitung</i>	155
5.2.3	<i>The software for the Meßdatenaufbereitung</i>	157
5.2.3.1	States and state transitions of the program	159
5.2.3.2	Analysis and Design with SA (Structured Analysis) / SD (Structured Design)	160
5.2.3.3	User Interface	161
5.2.3.4	Collection of data of the acceleration sensors	162
5.2.3.4.1	Decoding of the outputs of the acceleration sensors	
162		
5.2.3.4.2	Calculation of the acceleration with high accuracy	
163		
6	TEST RESULTS	165
6.1	HARDWARE-TEST	165
6.2	TEST OF THE OVERALL SYSTEM WITH THE MICROCONTROLLER (C167)	166
6.2.1	<i>Gesamtsystem-Test 1</i>	166
6.2.2	<i>Gesamtsystem-Test 2</i>	168
6.2.3	<i>Gesamtsystem-Test 3</i>	168
7	SUMMARY AND OUTLOOK	170

List of figures

Fig 1: translational and rotational motion parameters.....	135
Fig 2: Euler-Winkel.....	136
Fig 3: Spatial Representation.....	137
Fig 4: Earth's magnetic field.....	138
Fig 5: The architecture of the IMU the "Alternate Lotte"	139
Fig 6: Development Environment for the hardware development; EAGLE Layout Editor ..	141
Fig 7: Development Environment for the software development; μ Vision2 Text Editor	142
Figure 8: Block diagram of the Meßdatenaufnahme	143
Figure 9: Block diagram of the Beschleunigungsaufnahme	144
Fig 10: Embedding an accelerometer.....	144
Fig 11: Block diagram of the Winkelgeschwindigkeitsaufnahme	146
Fig 12: circuit for the Winkelgeschwindigkeitsaufnahme	147
Fig 13: Butterworth.....	148
Fig 14: Block diagram of the heading indicator	149
Fig 15: on-chip component of the Kompaßsensors.....	149
Fig 16: Set/reset circuit for the Kompaßsensor	150
Fig 17: Reference Voltage	151
Fig 18: Komponentenbesetzung on the circuit board 1 and 2.....	152
Fig 19: Platinenansicht of IMU-overall system.....	152
Fig 20: Block diagram of the Meßdatenaufbereitung (Sensordatenaufbereitung)	153
Fig 21: circuit for the Eingangsspannungsüberwachung	154
Fig 22: serial interface will be started	155
Fig 23: component side of the Mikrocontrollerboards	156
Fig 24: solder side of the Mikrocontrollerboards.....	157

Fig 25: Environment of the microcontroller.....	158
Figure 26: state diagram for the Meßdatenaufbereitungs-Software	159
Fig 27: User Interface of the IMU for testing purposes.	161
Fig 28: Sensor Output of ADXL210.....	162
Fig 29: the accelerometer, in X-direction.....	166
Fig 30: the accelerometer, in Y-direction.....	166
Fig 31: the accelerometer, in Z-direction.....	167
Fig 32: the accelerometer, in X-direction.....	168
Fig 33: the accelerometer, in Y-direction.....	169
Fig 34: the accelerometer, in Z-direction.....	169

List of Tables

Table 1: Bandwidth	145
Table 2: Resistance values for the setting of the period	146
Table 3: jumper settings on the microcontroller board	155
Table 4: Results of the gyros	165

8.1 Introduction

The inertial navigation system [1] is an autonomous system, which, in contrast to the satellites or other turnovers no external information is needed, and using a inertial navigation system is one in the location,

- To measure acceleration and rotation speeds,
- To determine angulation and
- To calculate position and speed - on dynamically moving vehicles.

And is possible

- In every weather.
- At any time,
- In every place on earth

Due to its construction, the present Inertialnavigationssystem assigned the Strapdown-Systemen. As Strapdown-Systeme are called inertial navigation systems, in which the acceleration sensors that are mounted to the vehicle, the equipped of the sensors are usually parallel to the main axis of the vehicle.

The sensors for such a Strapdown-Inertialnavigationssystem consists of accelerometers, the the translatory Beschleunigungskomponenten capture, the gyroscopes (gyros), the capture the Drehwinkelbeschleunigungen. These amounts are constantly in the direction of the axis of the detected vehicle korperfesten. From the measurement of the Drehwinkelbeschleunigungen differential equations can be the Winkelinkremente (azimuth-, Nick and Rollwinkelinkremente) and the angulation (azimuth-, Nick-, and roll angle) continuously calculate. From these values a navigation computer determined the location of the vehicle on the specified Navigationskoordinatensystems.

The Strapdown-Technik provides the direct coupling of the sensors to the vehicle dynamics high dynamic demands on the sensors themselves.

8.1.1 Task

In: solar-powered boat "Lotte" -Project of the University of Stuttgart has been to the identification of the dynamic models of the airship and for the realisation of regulations in a inertial measuring system with GPS procured, which will culminate in various already been used successfully.

For an alternative "low-cost" flight control system is the integration of a 3-D inertial measurement system called for in the system environment, the task is a Inertial Measuring System to design with a microcontroller and then to integrate into the overall system.

The signals from three accelerometers and three roundabouts are to be connected to a microcontroller, this will in turn over the serial interface with the Regelrechner be connected.

8.1.2 Overview

The goal of the national work it is, an IMU (Inertial Measurement Unit) with a "low-cost" to realize sensors.

First, the basics of inertial navigation systems presented.

Then the architectural design of the IMU is presented.

Afterwards, the realization of the individual parts described. The draft of the Board has with the program package EAGLE ver, performed 3.55 out. In the implementation of the Software, . (Microcontrollerprogramming) has been the program μ Vision ver, wedge used 2.03 & amp of the company, it is a shareware version for a limited code size of 16 Kbyte. The programming is in C. the production of the 4-ply Mikrocontroller-Platine is from the company PCB-POOL accepted

Thereafter some test results of tests of the entire IMU presented.

8.2 Basics

The content of this chapter is [1], [3] and [6] issued.

A spatial behavior and/or the movement of a body in the room can be described with six parameters: three Translationsgrößen (x-, Y-, Z-acceleration) and three Rotationsgrößen (x-, Y-, Z- angular velocity), the movement of the body to have to define three accelerometers and three gyroscopes are joined together on a platform (Strapdown-Konzept), so that you form an orthogonal system, and the distance, the journey to be, and the angle to which the body has rotated, can by integration of the individual translations and rotations will be calculated, by accurate calculations using the periodic scans at the sensor output can be made using the ideal system to track movement and the current position [3].

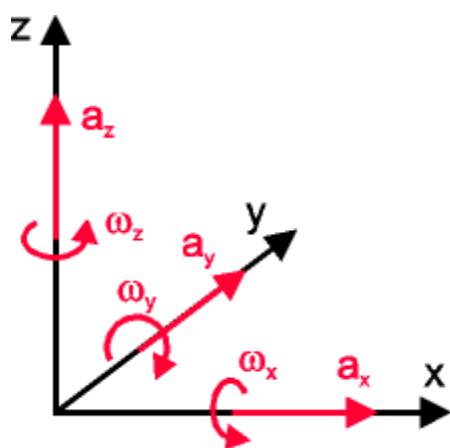


Fig 1

According to Fig 1 can be the distance traveled and the rotation through the following basic formulas calculate.:

$$s(t) = \iint a(t) dt^2 \quad (8-1)$$

$$\varphi(t) = \int \omega(t) dt \quad (8-2)$$

The main limitation of a bet of the system performance is to the limited precision of the given sensors. is a continuous small error in the acceleration once integrated, this will result in the results of the integration to a error in the speed, the speed is still once integrated, it will cause a large error in the distance, and thus are very accurate sensors and bug fixes ("Feedbackalgorithmen") needed to ensure an accurate to obtain Tragheitsnavigationsplattform. An example of a 'cheap' Feedbackalgorithmus is the G-vectoring. It does not require any additional hardware, but simply take to that the average direction of the Z-Beschleunigungsvektors exactly perpendicular to the direction down earth's surface, and its average $-9,81 \text{ m/sec}^2$ is, and a different putting is the inclusion of GPS positional data, will be fed into the the, but this approach requires careful considerations on the upgrade process, so that the entire system is not disturbed.

8.2.1 Coordinate System

Another difficulty is the choice of the coordinates of the orthogonal system. There are different solutions, and some of them allow redundancy with the Add of a improved precision.

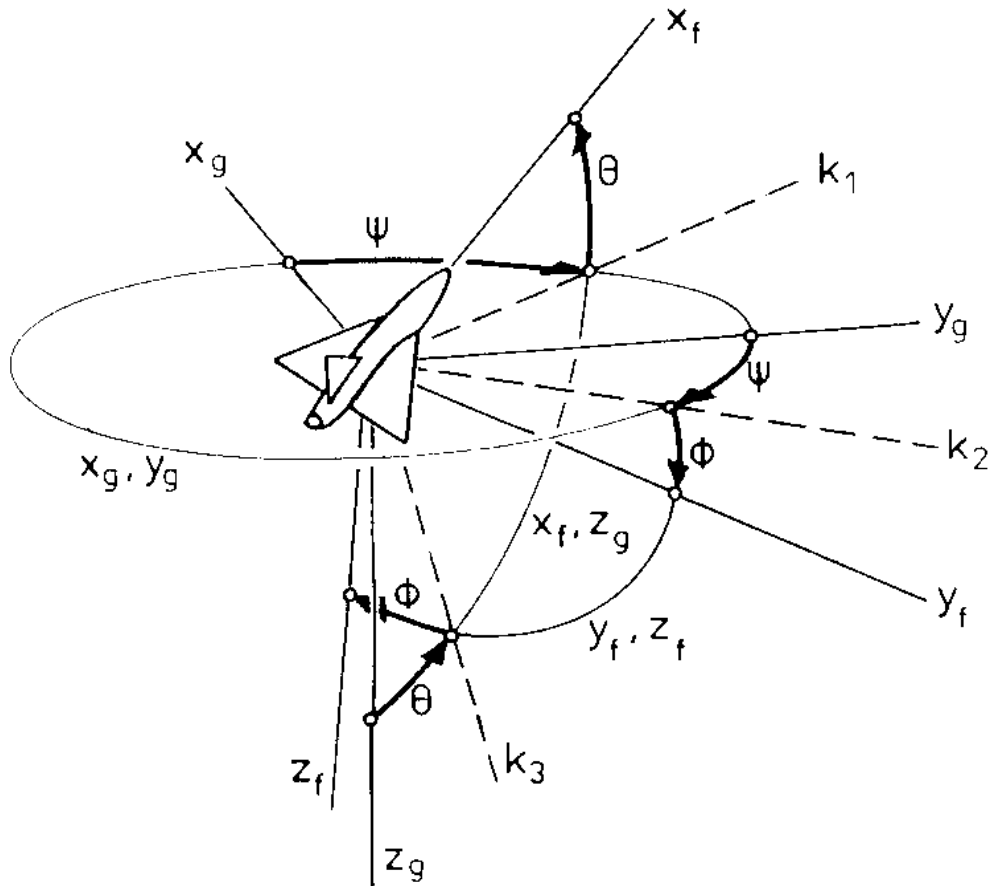


Fig 2

It was the representation with the 'Euler' -Angles [1] is selected, the Euler-Winkel are defined as follows (see Fig 2

- ψ Azimut, heading, *heading* (also yaw angle); Axis LPH
- Θ Longitudinal Tilt, *pitch angle* (also pitching); axis K2
- Φ Hangewinkel, *bank angle* (also roll angle, slope); Axis XF

8.2.2 Calculation of the current position

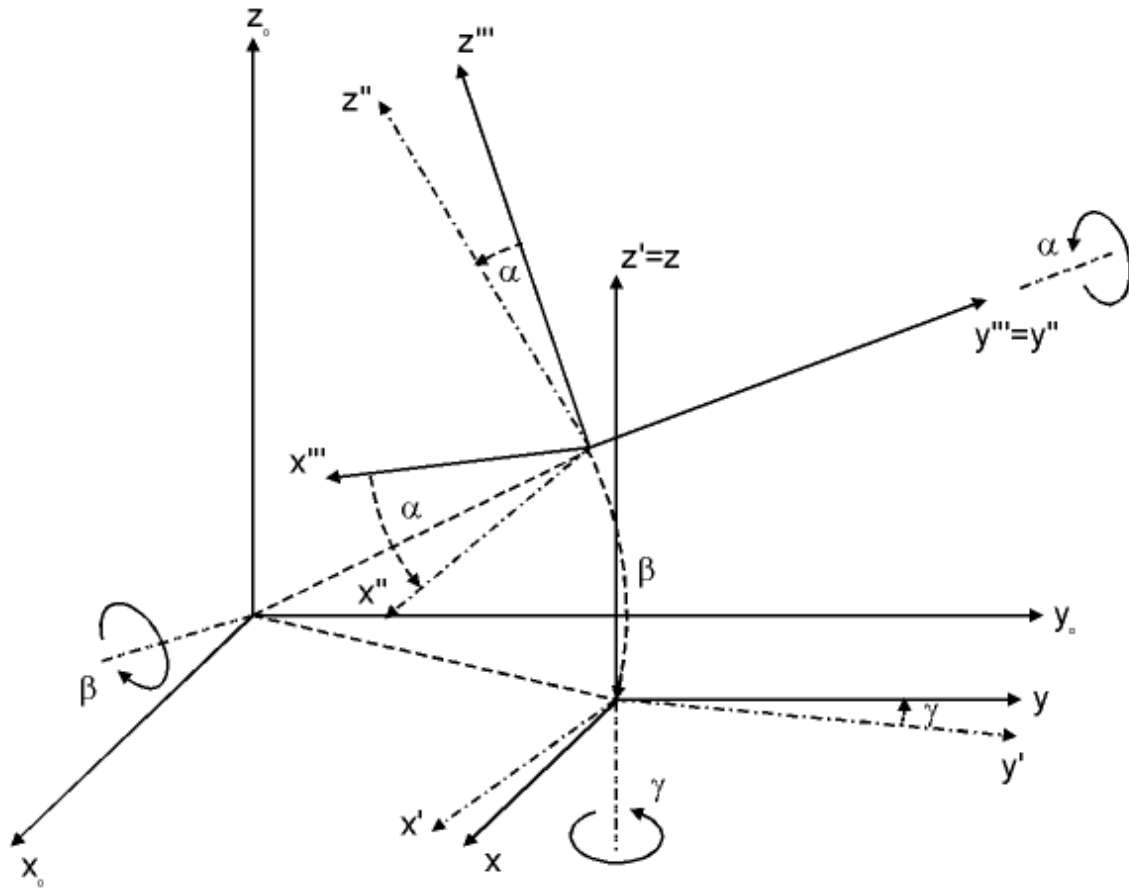


Fig 3

Based on Fig 3 , the current movement based on the Koordinatenreferenz (x, y, z) and a mathematical Transformation (based on the Euler-Winkeln) calculate:

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} \cos(\gamma) * \cos(\alpha) & -\sin(\gamma) * \cos(\beta) & -\cos(\alpha) * \sin(\beta) * \sin(\gamma) \\ -\sin(\gamma) * \sin(\beta) * \sin(\alpha) & \cos(\gamma) * \cos(\beta) & +\sin(\alpha) * \cos(\gamma) \\ \sin(\beta) * \sin(\alpha) * \cos(\gamma) & \cos(\gamma) * \cos(\beta) & \cos(\gamma) * \sin(\beta) * \cos(\alpha) \\ -\cos(\alpha) * \sin(\gamma) & \cos(\gamma) * \cos(\beta) & +\sin(\gamma) * \sin(\alpha) \\ \cos(\beta) * \sin(\alpha) & -\sin(\beta) & \cos(\beta) * \cos(\alpha) \end{pmatrix} \begin{pmatrix} \Delta x''' \\ \Delta y''' \\ \Delta z''' \end{pmatrix}$$

The current Winkelgeschwindigkeitsvektor (α, β, γ) is then calculated using the following formula:

$$\begin{pmatrix} \Delta\alpha \\ \Delta\beta \\ \Delta\gamma \end{pmatrix} = \begin{pmatrix} 1 & \sin(\alpha) * \tan(\beta) & -\cos(\alpha) * \tan(\beta) \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha)/\cos(\beta) & \cos(\alpha)/\cos(\beta) \end{pmatrix} \begin{pmatrix} \Delta\alpha''' \\ \Delta\beta''' \\ \Delta\gamma''' \end{pmatrix}$$

Important here is the sample rate, you must be large enough, if a fast rotation movements is present (because of the Abtasttheorems).

8.2.3 Compass

Most navigation systems today use a compass or something similar, in order to determine the thrust, with the measurement of the magnetic field strength of the earth, can the Electronic Compasses, the magnetoresistischen sensors are based on a rotation to 0.1 degree determine.

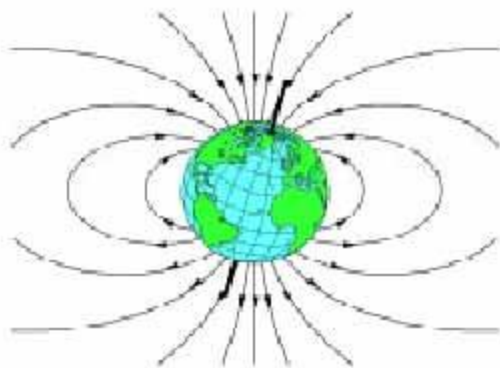


Fig 4

The magnetic field strength of the earth is approximately 0.5 to 0.6 Gauss and has a component that is parallel to the earth's surface, and which to the north shows. This is the basis for all the magnetic compasses, the magnetic field of the earth can be approximated with the Dipolmodell in Fig 4 is shown. This figure illustrates that the direction of the Earth's magnetic field is always on magnetically North shows, this field is used to determine the Kompaßrichtung.

8.3 Architecture design of the IMU

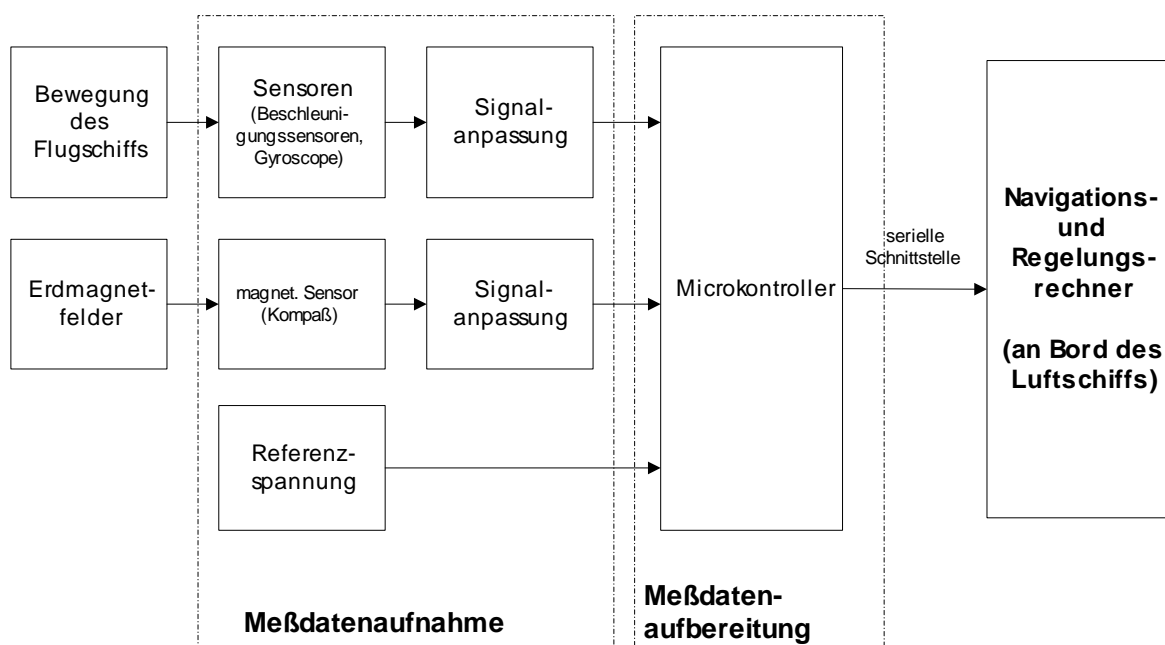


Fig 5

8.3.1 Measurement Data Collection (Meßdatenaufnahme)

The Meßdatenaufnahme consists of sensors, to whose outputs a signal adaptation is switched, the the output signals of the sensors on a reinforced such a voltage level that you, as input signals to the microcontroller (Meßdatenaufbereitung) can be connected, i.e. the Meßdatenaufnahme hardware is realized only.

8.3.2 Measurement Data Processing (Meßdatenaufbereitung)

The Meßdatenaufbereitung is realized with a software to a microcontroller is running, the adapted (partially analog) Sensorausgangssignale are the input signals of Meßdatenaufbereitung. By the Meßdatenaufbereitung are first the in analogue form adapted existing Sensorausgangssignale changed into digital signals.

All of the sensors (accelerometers, gyroscopes and magnetoresistive sensors) have a drift on for temperature fluctuations, and the Meßdatenaufbereitung has a Temperaturmeßeinheit and can to use the known (driftbehafteten) Meßdatenaufnahme-characteristics of the sensors this drift again deduct. Finally, the corrected, digital sensor values are lined up in a row and on a serial output of the microcontroller and the further use of the data on the navigation and Regelungsrechner given on board.

The microcontroller with its external additional elements (additional memory, Treiber, ...) is realized on its own Board, to a hardware of the decoupling of the Meßdatenaufbereitung to achieve Meßdatenaufnahme. This is granted a modularity, the subsequent changes of IMU-systems easier.

Further processing of the IMU-data:

Project increment 1:

As a provisional solution, the IMU-data from the microcontroller to the navigation and Control computer (Regelungsrechner) give you the unprocessed to the ground station via radio data transmission sends, and only find there the integrations instead.

Project inkrement 2:

The navigation and Control computer takes the necessary integrations before and uses the position and order for his control algorithm, and the position and order together with all other be sensor data to the ground station via radio sends.

8.4 Development environment

8.4.1 Hardware-Entwicklungsumgebung

The hardware development is done with the program of the company Cadsoft EAGLE. It is the eagle-Version 3.55 , with Windows 2000, the tool includes a **layout editor**, with the the boards were designed, **EAGLE** also contains a **Bibliotheks-Editor** , a **CAM (Computer Aided Assembly) -processor** and a **text editor**, you can with the **Bibliothekseditor** housing and edit icons.

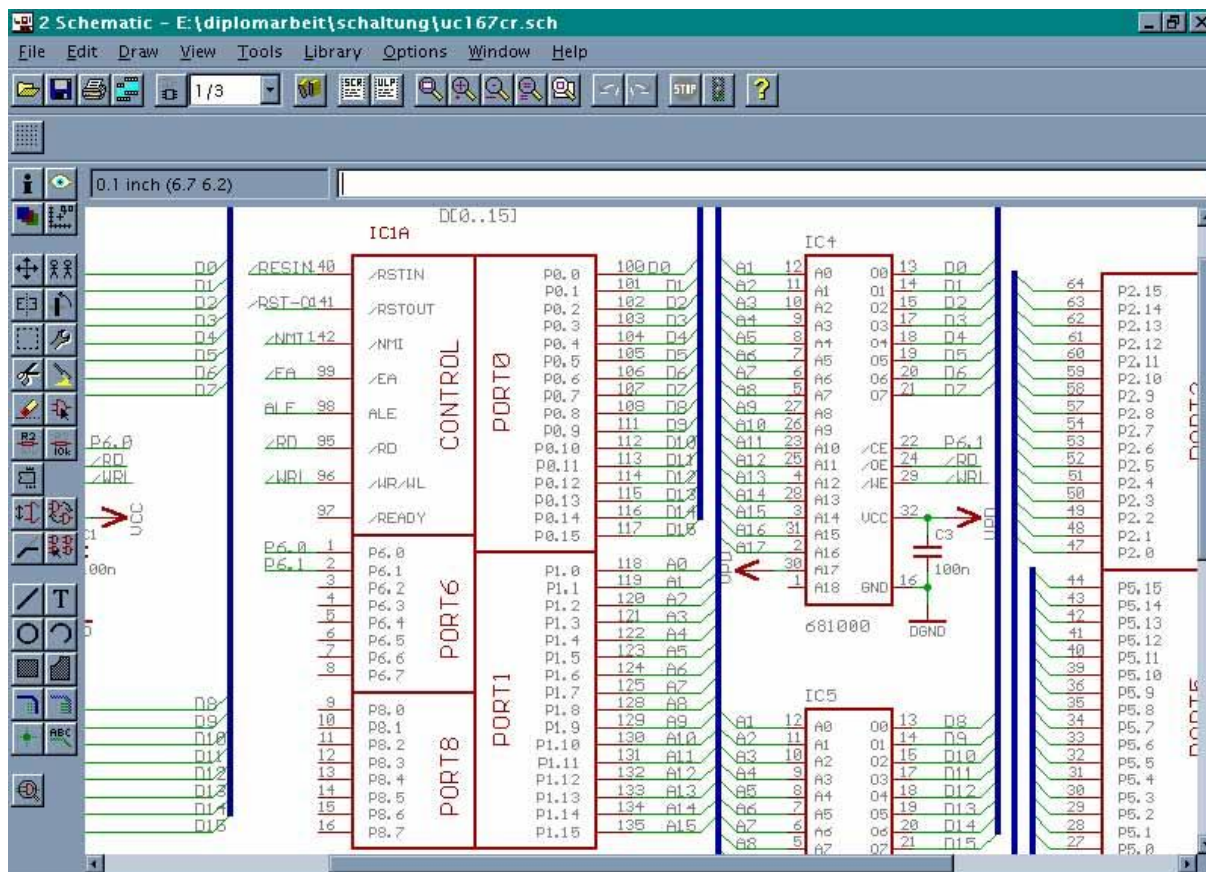


Fig 6

8.4.2 Software Development Environment

The programming of the microcontroller C166 was done with the development μ Vision of the company Keil \square

One finds in μ Vision, Version 2 the basic elements of a modern IDE:

- A specially adapted text editor.
- An ANSI-C-compiler (C166-ANSI-C-compiler),
- A Assembler (A166-Assembler),
- A left/Lokator (L166),
- A debugger (μ Vision2 Debugger)

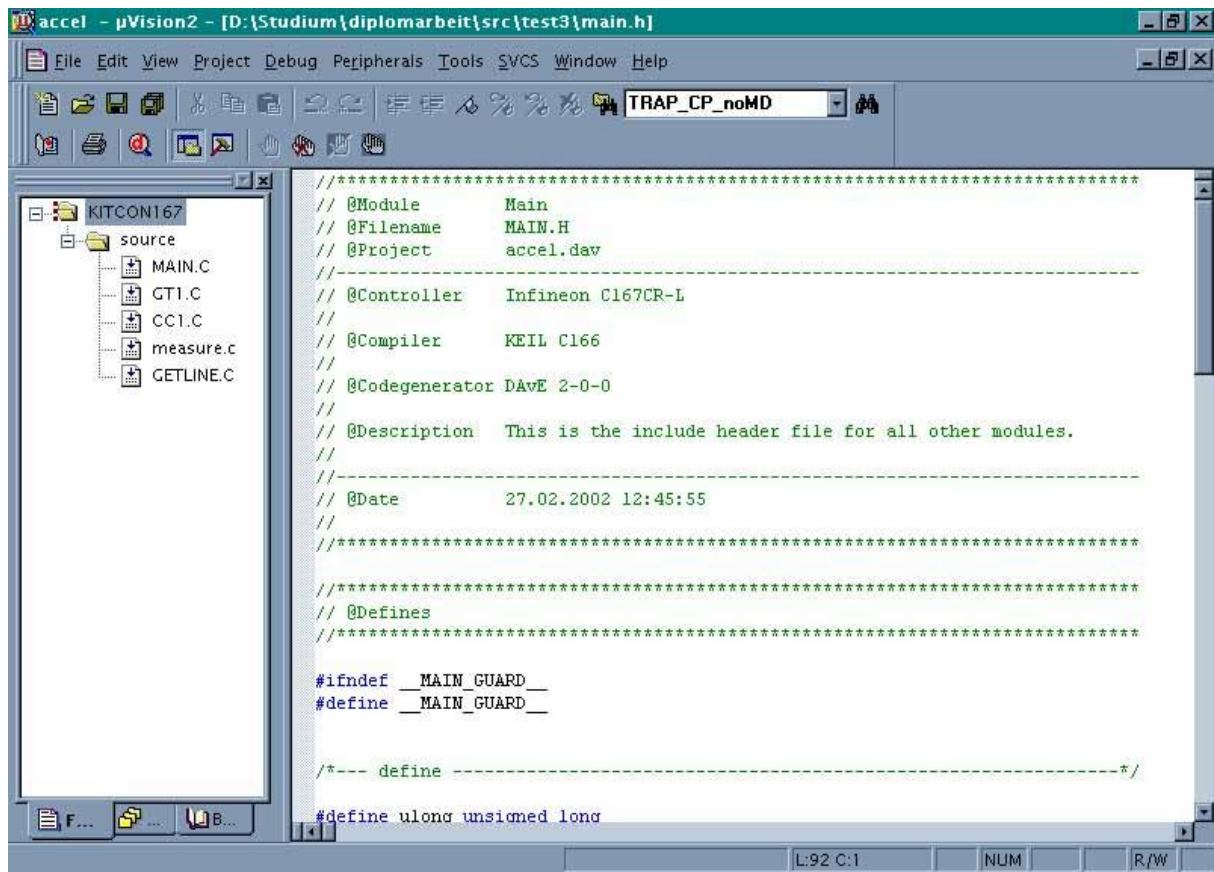


Fig 7

8.5 Realization of the IMU

In this chapter will be the implementation of the Advanced Inertial Measurement System explains. After the block diagram of the Meßdatenaufnahme Meßdatenaufnahme is the shifting of the presented, the individual parts of the circuit are described in detail, on the same principle will be the implementation of the described Meßdatenaufbereitung and, in the case of the Meßdatenaufbereitung even the description of the developed software on the Microcontroller is added.

8.5.1 Meßdatenaufnahme

The task of the Meßdatenaufnahme consists in, and the movement of the airship and the earth's magnetic field to capture the Meßdatenaufnahme is composed of three acceleration sensors, three roundabouts and three magnet sensors together.

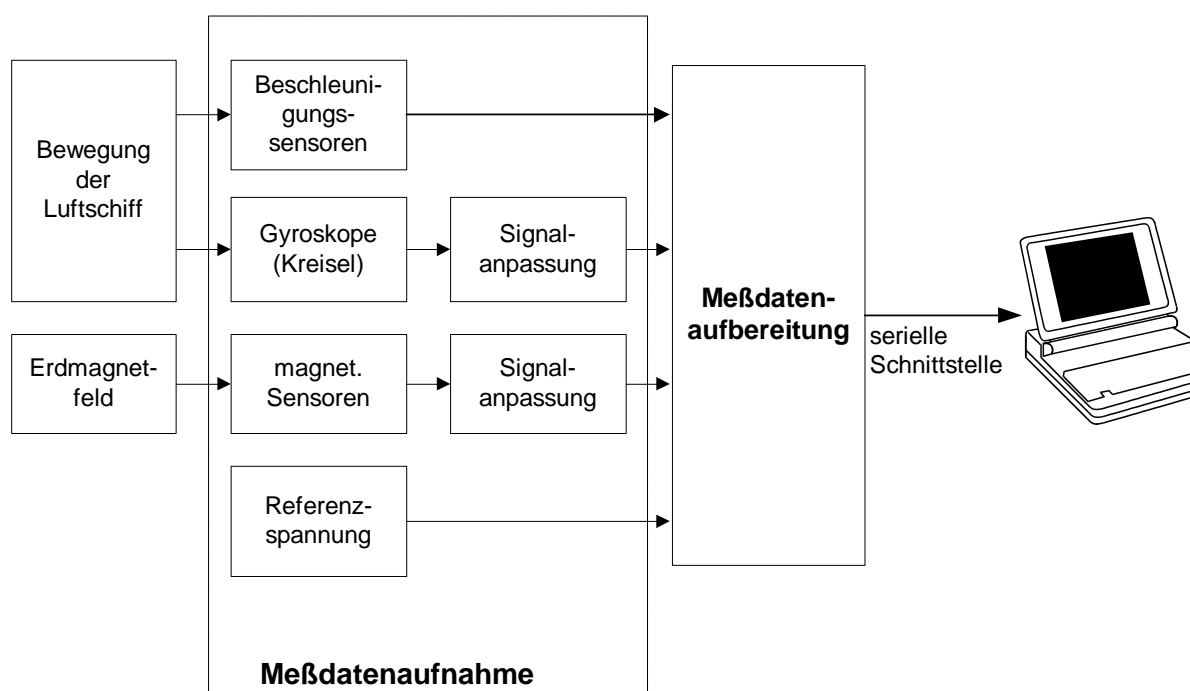


Figure 8

For the adaptation of the Sensorensignale Meßdatenaufbereitung to the additional Signalanpassungsschaltungen be required.

8.5.1.1 Circuit Design for the Meßdatenaufnahme

The whole circuit for the Meßdatenaufnahme is shown in Annex A, the following are the individual areas of the circuit described in more detail.

Acceleration Sensors

Accelerometers measure, as the name already says, the rate of change of velocity, by creating electronic signals, the measured or to control a process can be used, and traditional applications include the activation mechanisms of safety belts and air bags, in which the sudden deceleration of the accelerometer which is responsible for the security techniques in gear sets.

The above sensors use a mechanism, the with of the elongation of a spring can be compared, the force, by the acceleration due to is, extends the spring, with a higher acceleration is even more stretched the spring. The measurement of the Deformationslangen and the acceleration can be determined.

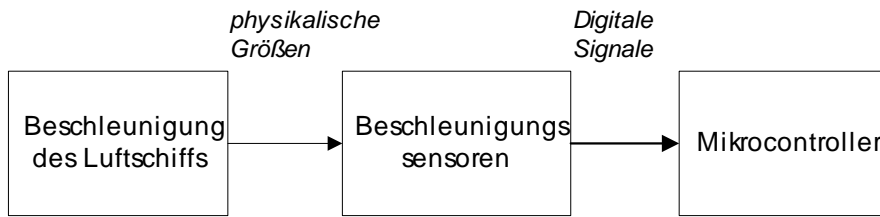


Figure 9

The ADXL210 is a "low cost" 2Achs-Beschleunigungssensor with low power consumption, and the measurement is between - 10g and +10G. The special on the ADXL210 are the digital outputs on the two axles, the outputs are digital signals, whose length (ratio of the pulse width to the period) proportional to the acceleration in two axles are. These outputs can directly be measured with a Mikrocontroller-Zähler. There is no ad-converter needs. The Arbeitszyklusmodulator (DCM) has a resolution of 14 bits, based on this special properties, the acceleration with a simple schema record. After Figure 9 , the digital outputs of the sensor to the microcontroller directly connect the circuit for the embedding of such an accelerometer is in the Fig 10 shown.

The ADXL210 is to configure the following: the length of the period and the bandwidth of the signal output (e.g. , elimination of high-frequency vibrations). These settings are described below.

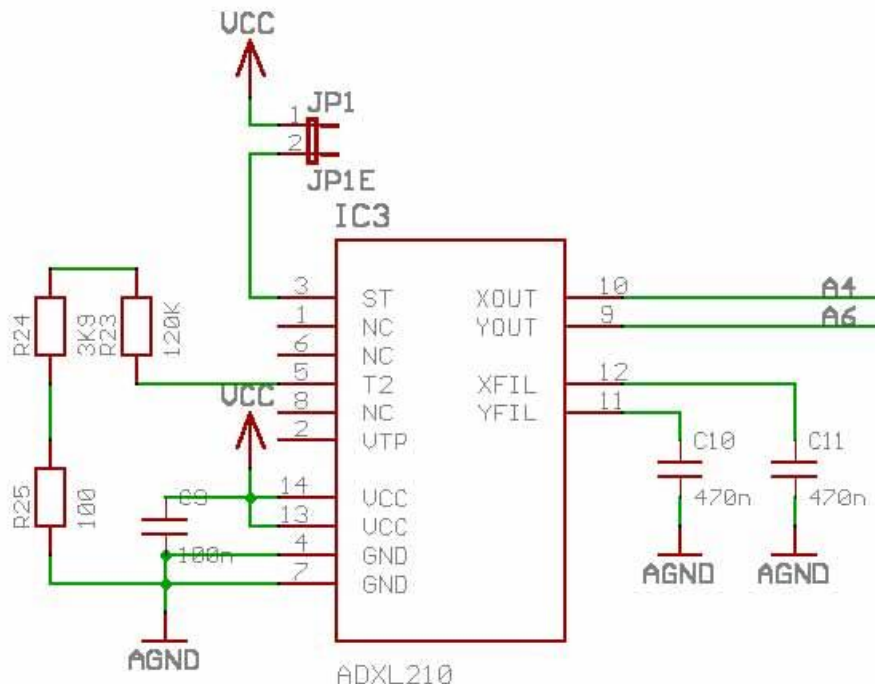


Fig 10

The error behaviour of the acceleration sensor

The sensor failure δa of the ADXL210 is in general from a scale factor error κ , a bias B and a stochastic noise process w , so that the error can be described as follows:

$$\delta a = \kappa a + B + w$$

There is a possibility, before each experiment to determine the bias, so can the error with a much lower bias be assessed, as without this information.

Laying down the bandwidth of the accelerometer

The ADXL210 from *Analog Device* is set to 100 Hz bandwidth. This setting is By C_x and C_y (here $C10$ and $C11$ connector, see 10), determines, two capacities are to pin 11 (Y_{filt}) and 12 (X_{filt}) connected, and the calculation of the capacity is as follows: Fig 10

$$F_{3dB} = \frac{1}{(2\pi(32k\Omega) \times C_{(x,y)})} \quad (8-3)$$

$$F_{3dB} = \frac{5\mu F}{C(x,y)} \quad (8-4)$$

Bandwidth	Capacity
10 Hz	0.47 μ F
50 Hz	0.10 μ F
100 Hz	0.05 μ F
200 Hz	0.027 μ F
500 Hz	0.01 μ F
5 KHz	0.001 μ F

Table 1

Definition of the period duration of the accelerometer

The length of the period at the sensor output is determined by a resistance R_{set} (in the image $R23+R24+R25$), is connected with the GND, the equation is then:

$$T2 = \frac{R_{SET} (\Omega)}{125 M\Omega} \quad (8-5)$$

T2	Rset
1 MS	125 K Ω
2 MS	250 K Ω
5 MS	625 K Ω
10 MS	1.25 M Ω

Table 2

The gyro and the filtering their output signals

The angular velocity with a roundabout you can be record. This recording is in the Fig 11 shown to adapt it to the microcontroller a Butterworth filter be added and an amplifier.

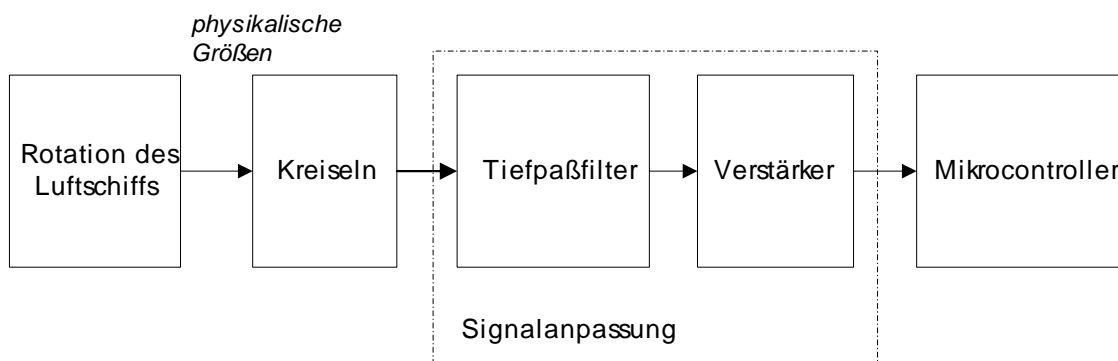


Fig 11

The "heart" of the gyroscope is a Keramikstab, which, in its longitudinal axis is brought to vibrate, the rod is in two places with Metallgabeln welded to the "nodal point" of the rod are, and when the rod to rotate is placed on the vertical plane to a vibration appears through influence, the with the angular velocity is the same, the on the rod mounted piezo-electric plates are both a way to swing the rod longitudinal to bring, as well as the vibration generated by the appears through influence on the vertical plane should be repealed, and the voltage, the repealing of the vibrations on the vertical level is necessary, there is information on how fast the rod (and therefore, the gyroscope) is turning. The gyroscope in this way creates an output voltage that is proportional to the angular velocity is.

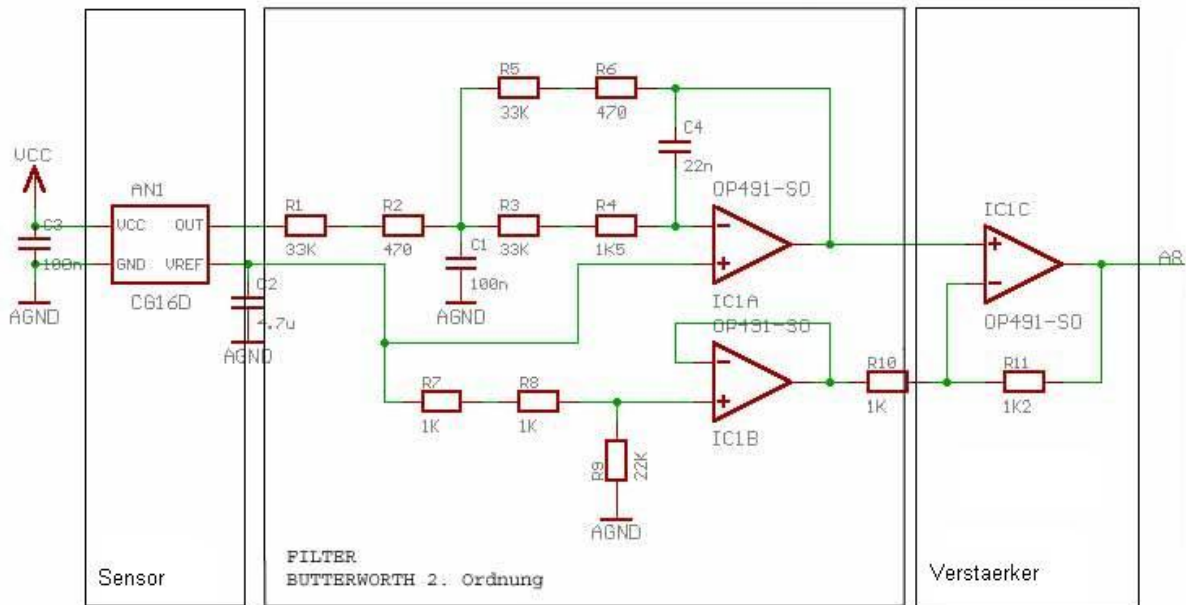


Fig 12

The output of the Kreiselsensoren is an analog voltage. To reduce noise and for antialiasing, the sensor output is filtered by a Butterworth 2nd-order filter. The resulting signal is then amplified by a 1.2-fold gain using an AD-converter led of the microcontroller, implemented with an IC of *analog device* of the type OP491.

The Butterworth filter is implemented using only resistors and capacitors with a fault tolerance of 1%. This is necessary because the filter must be exactly calculated. The values of the resistors and capacitors are C_1 with 22 nF and C_2 with 100 nF. Based on these values, the values of the other resistors can be determined.

Calculation of the order Butterworth

$$A_0 = -\frac{R_2}{R_1} \quad (8-6)$$

$$a_1 = \omega_g C_1 \left(R_2 + R_3 + \frac{R_2 R_3}{R_1} \right) \quad (8-7)$$

$$b_1 = \omega_g^2 C_1 C_2 R_2 R_3 \quad (8-8)$$

The following constants are specified:

$$A_0 = 1; a_1 = 1.4142; b_1 = 1$$

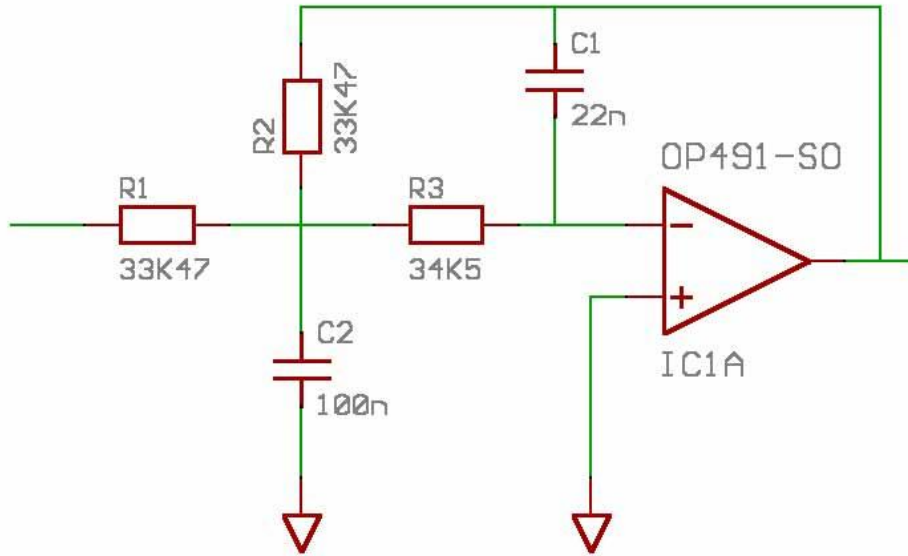


Fig 13

As a fixed values has been $C1$ with 22 nF and $C2$ set with 100 nF. It follows after the forming of equation (8-8

$$R_2 = \frac{a_1 C_2 - \sqrt{a_1^2 C_2^2 - 4C_1 C_2 b_1 (1 - A_0)}}{4\pi f_g C_1 C_2} \quad (8-9)$$

$$R_2 = 33,435k\Omega \approx 33,470k\Omega$$

As a last then you can $R3$ calculate

$$R_3 = \frac{b_1}{4\pi^2 f_g^2 C_1 C_2 R_2} \quad (8-10)$$

$$R_3 = 34,436k\Omega \approx 34,500k\Omega$$

The Compass sensors and filtering their output signals

The current direction of the airship can be seen, in which you have a sensor that the magnetic field on the surface of the earth measures, uses, this compass is in the Fig 14 briefly presented. With a instrumentation amplifier is the output signal from the sensor connected to the microcontroller.

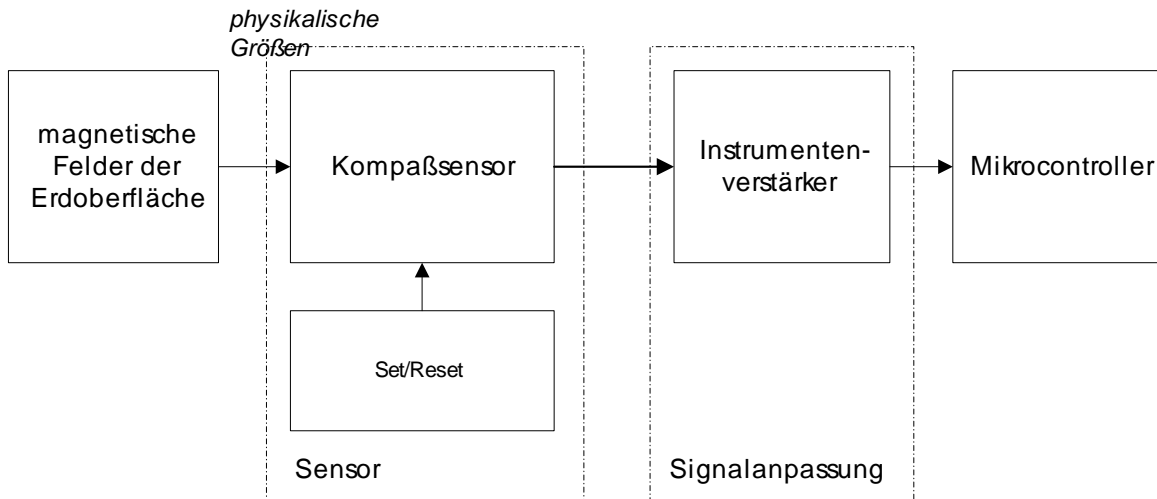


Fig 14

Magnetoresistive sensors from Honeywell are simple bridge connections (Figure 15), the only an input voltage need to to measure magnetic fields, and if a voltage from 0 to 10 volts to Vbridge is connected, begins to the sensor, a magnetic fields to measure in the environment. In addition, the sensor two "on chip straps", the offset and the Enter reset switch. Fig 15

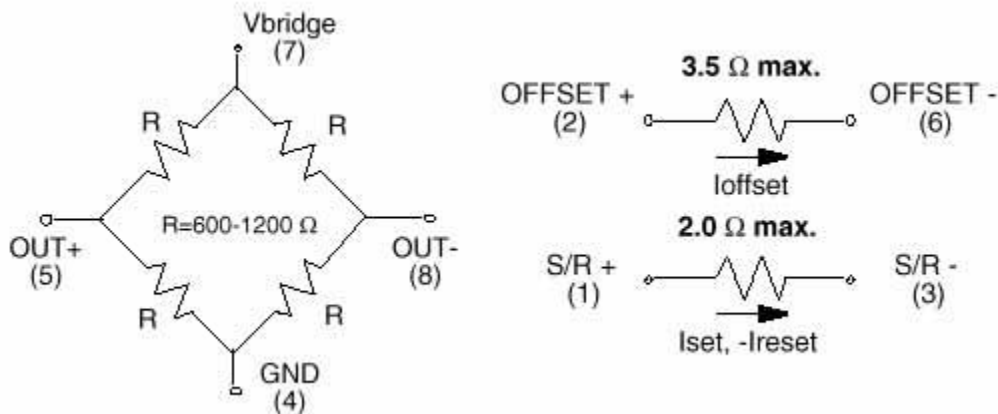


Fig 15

The offset-bridge can be different operating modes, if a DC by the offset-bridge flows. An unwanted, but known magnetic field can be subtracted out.

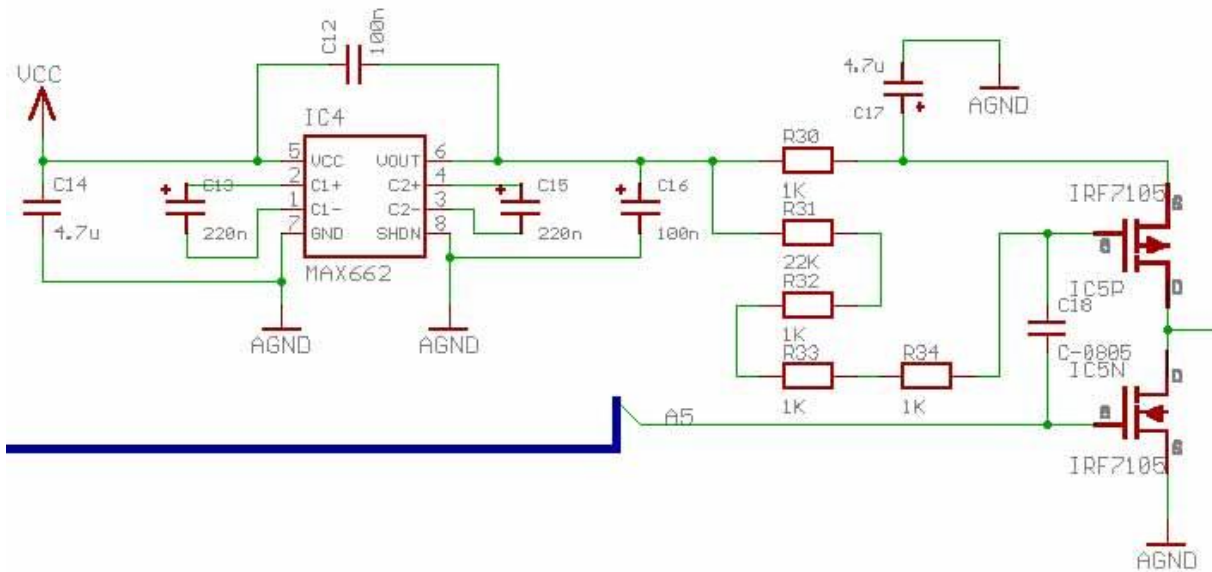


Fig 16

The Set/reset (S/R) can be a high current pulses, which are then very useful, if the sensor in a highly sensitive work mode can be, and most magnetic sensors, measure the low field strengths, are influenced by larger magnetic fields ($> 4 - 20$ gauss), the can lead to a Ausgangssignalverminderung. In order to to reduce this effect, and to the signal at the output to maximize, can a magnetic circuitry (Fig 16) on the S/R will be applied, the eliminated the effect, with the help of IRF7105 brings top is at the exit of the Set/reset circuit with a power output of approximately 3A present, the purpose of Set/reset (S/R) is more sensitive to the sensor back to make small field strengths. This is accomplished, by a large current through the S/R pulsates. Fig 16

8.5.1.2 Reference Voltage

The analog output signals of the sensors require a reference voltage, this reference voltage is the building block by UM780 AD generated, as both Referenzspannungserzeuger, as well as use Temperaturänderungssensor. By AD780 is a Ultrahochpräzision "Bandgap" generated reference voltage, the 2.5 V or 3.0 V provides. The required input voltage can be between 4-36 volts.

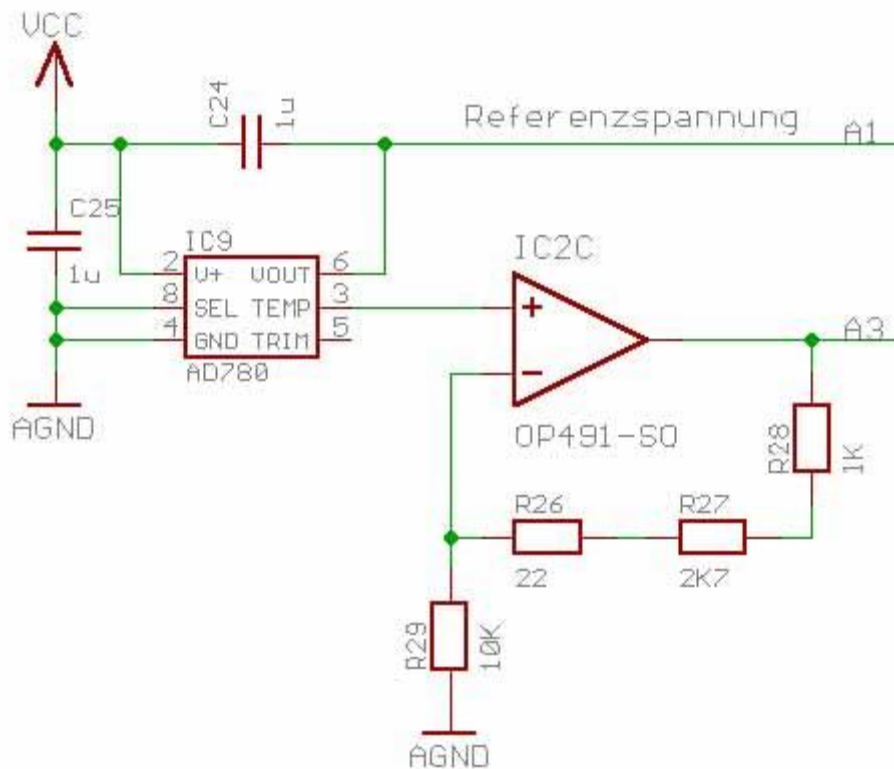


Fig 17

In addition to the reference voltage generated of the AD780 is still a further output voltage, the correlated with the temperature difference (in Fig 17

A room temperature of 25 C is the outlet temperature voltage of 560 mV (according to the data sheet) with a "temperature sensitivity" of 1.9 mV/°C, since the reference voltage of 2.5 volts and AD-converter of the microcontroller a resolution of 10 Bit has, the system has a resolution of:

$$\frac{2,5V}{2^{10}} = 24,4 mV .$$

To the capture of signals to simplify, was elected an amplifier of 2.47 (see Fig 17 1 C temperature change then caused a change in voltage at the output of + 4.7 mV. during the room temperature is the output voltage: 1.48 V.

8.5.2 Board layout design for the Meßdatenaufnahme (Measurement data collection)

The three a wide array (acceleration sensors - Drehwinkelmessung and telemetry, magneticfield measurement, etc.) each have three components (each for the measurement in the x-, Y-and z-direction). In all three Sensorpaketen the sensors must be (responsible for the Meßdatenaufnahme) also in the direction of the corresponding spatial axis (X, Y, and z-axis) be aligned, i.e. , that a package two sensors (e.g. , for the Meßdatenaufnahme in x and y-direction) in a plane, and the third sensor (e.g. for the Meßdatenaufnahme in Z-direction) in a vertical plane.

This can be realized by two sensors each of the three a wide array on a circuit board and the third sensor placed all three a wide array on a second board, perpendicular to the first circuit board is mounted.

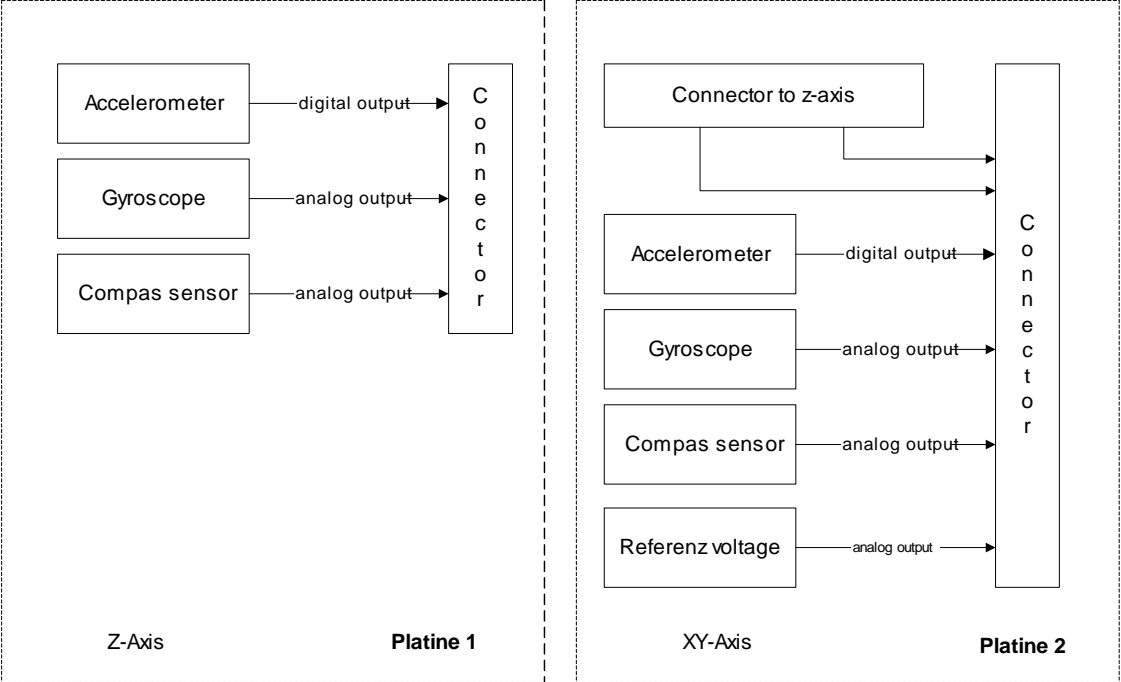


Fig 18

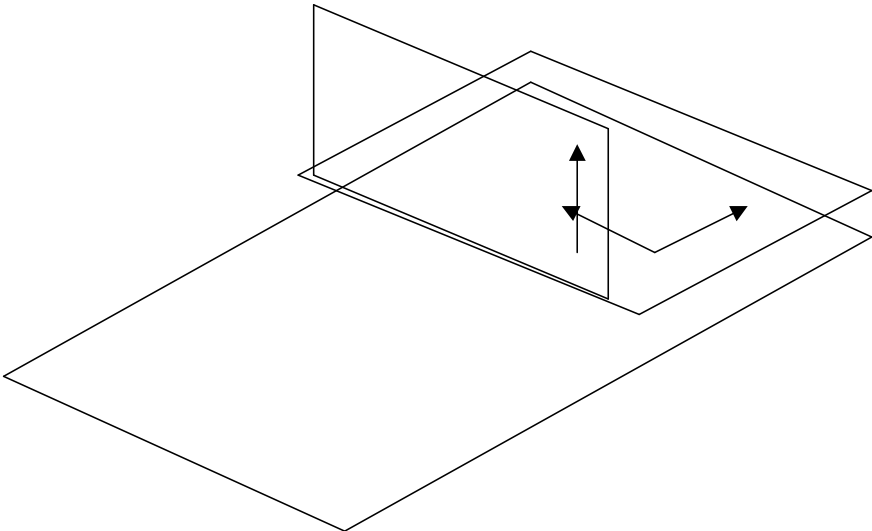


Fig 19

8.5.3 Meßdatenaufbereitung

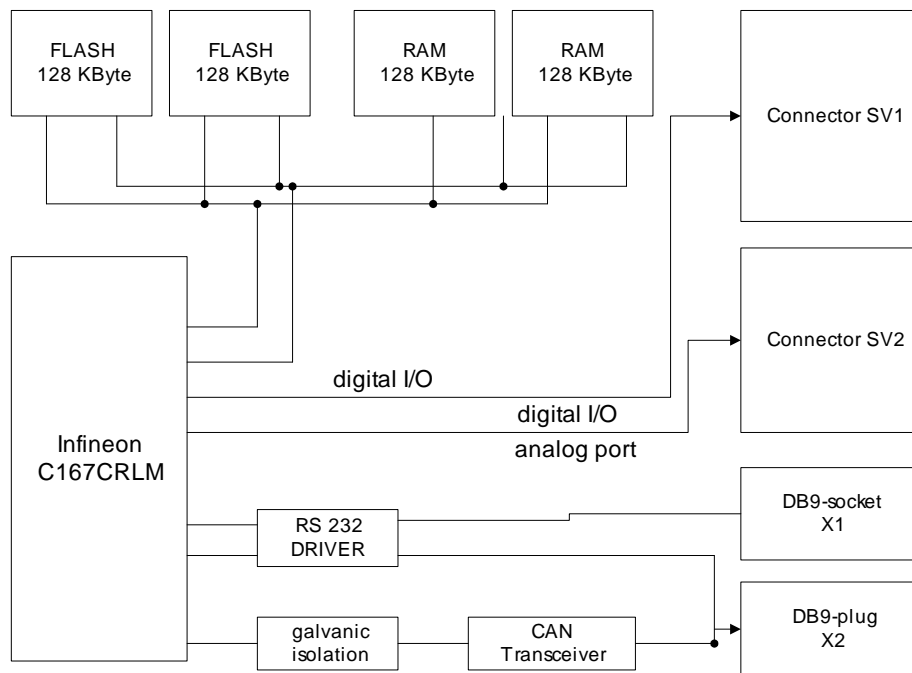


Fig 20

8.5.3.1 Circuit Design for the Meßdatenaufbereitung

The Meßdatenaufbereitung is with a microcontroller (with additional memory and peripheral components) are realized.

The shift, the the microcontroller with the additionally required memory and peripheral components connects, is listed in Appendix B, the following are the individual sections of the circuit described in more detail.

Supply Voltage

Supply voltage is used as a component of the type 78M05. In the component is an input voltage between 7.5 volts and 12 volts input. As a output voltage is 5 volts, all components work with 5 volts of power. The pins of the inputs are via the jumpers JP2 and JP6 (see Fig 23

Reference Voltage

For the reference voltage there are two possibilities: either the supply voltage of 5 Volt be used as a reference voltage, or it can be connected a separate reference voltage, which can be selected by jumper settings (see JP10 in JP11 in the Table 3Fehler! Textmarke nicht definiert.).

Memory

The microcontroller is to 256 KByte ROM (2 blocks per 128 Kbytes) and 256 KByte RAM equipped. The used Rome is of the type 681000, the RAM of the type 29F010 of the Philips company, with the power supply of RAM was a second supply voltage available to potential

losses to avoid the memory, and the building block MAX690 machines only (by Maxim) takes on this task (see Fig 21

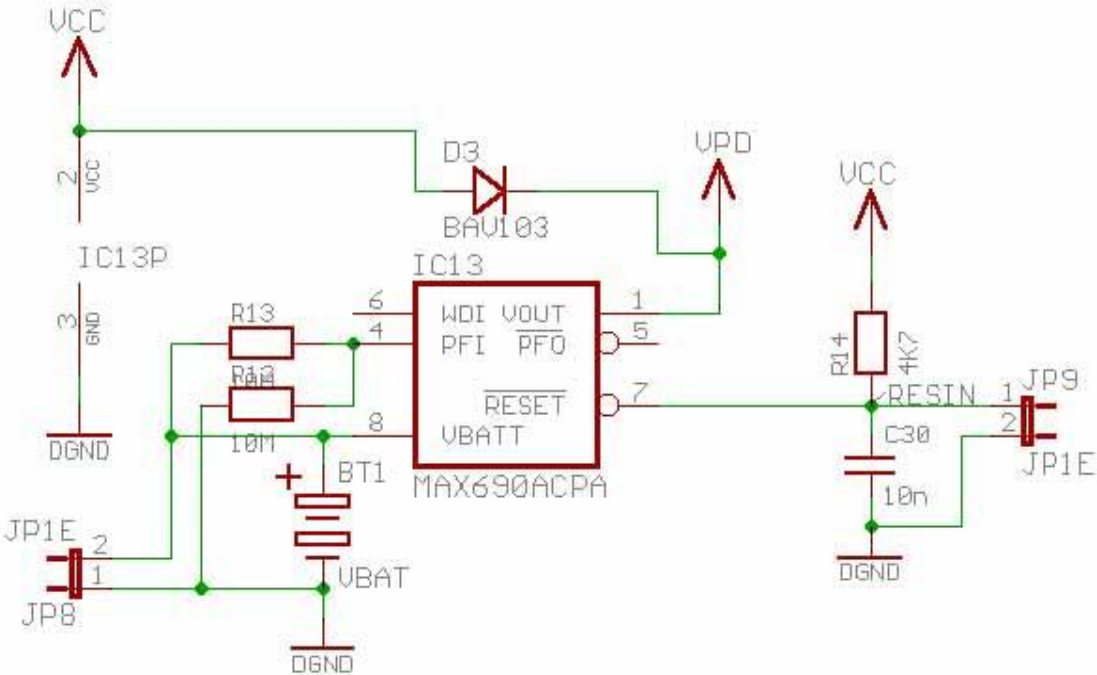


Fig 21

For a connection with other computers or for the programming of the flash RAM, a serial interface and a CAN-bus to the board appropriate. Instead of a port a CAN-bus the connection can also be used as a second serial interface. The first interface X1 (see Fig 22

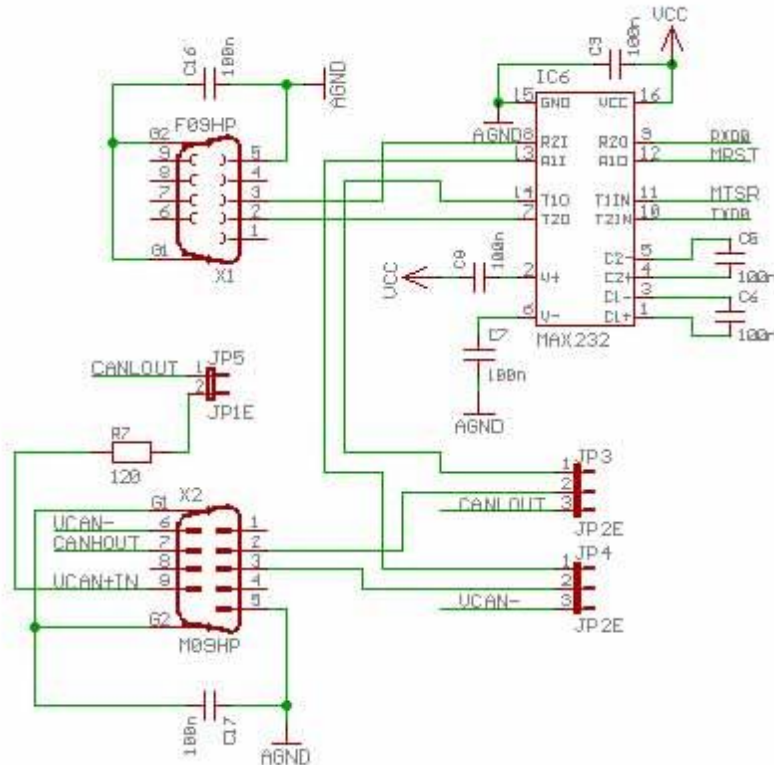


Fig 22

Jumper Settings

The microcontroller board can be configured with different jumper settings, and there is still an additional switch to the RAM or the Rome to configure.

The jumper settings have the following functions:

	Default Setting	Alternative setting
JP1	(1 +2) CAN-VCC generated by Supplay	(2 +3) CAN-VCC generated by CAN-network via DB9-X2
JP8 JP9	(2 +3) DB9-X2 works as CAN	(1 +2) DB9-X2 works as a second serial specific
JP10	(1 +2) VAREF generated by VCC	(2 +3) VAREF generated by external Supplay on SV2
JP11	(1 +2) RGND generated by DGND	(2 +3) RGND generated by external earth via SV2

Table 3

8.5.4 Board layout design for the Meßdatenaufbereitung

As for the components of the Meßdatenaufbereitung no special spatial directions (as in the Meßdatenaufnahme) are required, the entire Meßdatenaufbereitung on a sufficiently large board will be realized.

It shows that the whole circuit for the Meßdatenaufbereitung on a four-layered EURO-board is to realize.

The first of the four layers is the Bestückungsseite, an interlayer is the ground, a second intermediate layer is connected to the supply voltage, and the fourth layer is used as a solder.

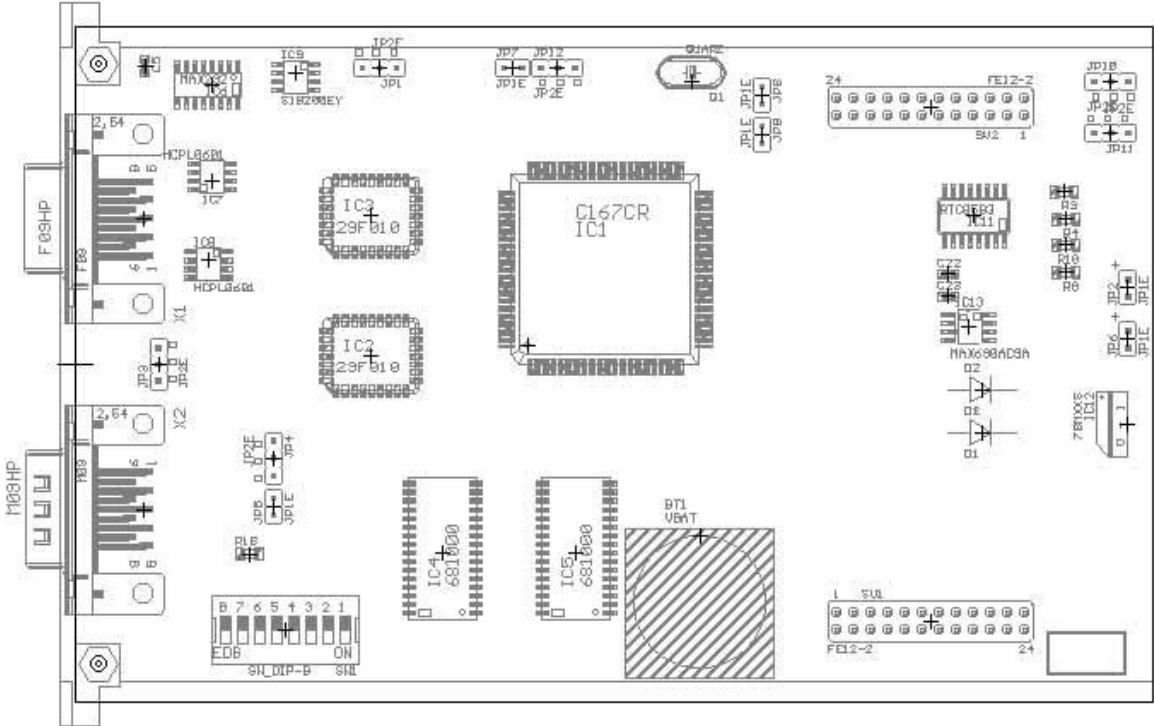


Fig 23

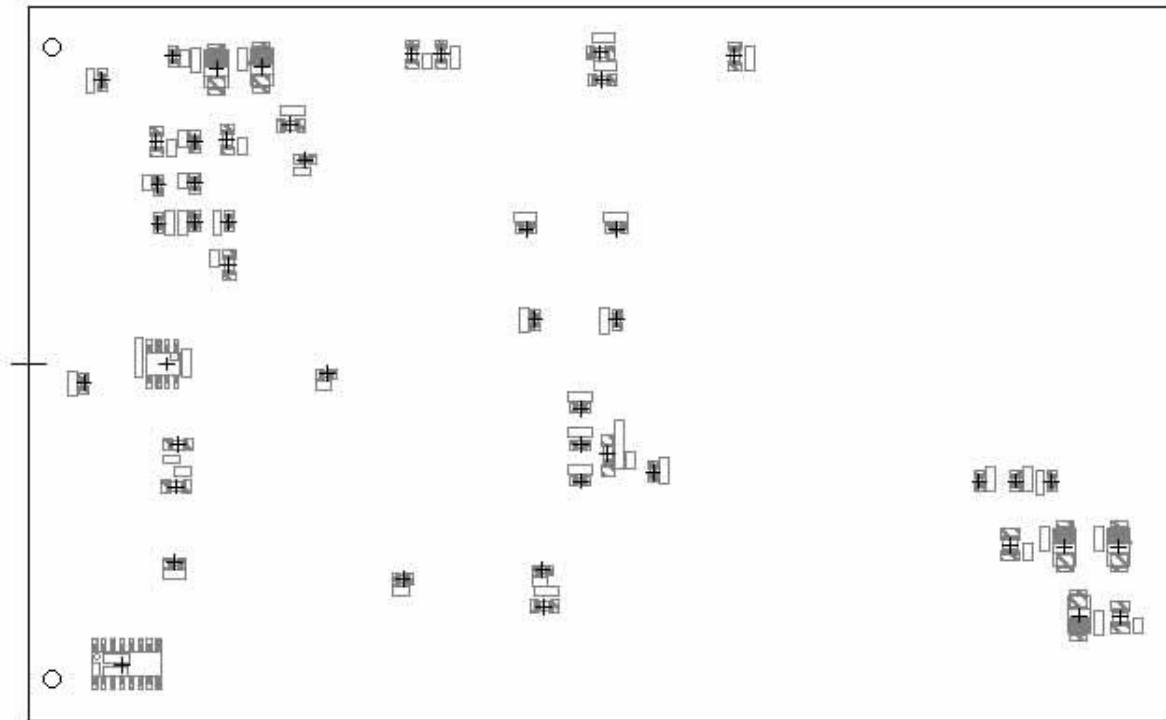


Fig 24

8.5.4.1 The software for the Meßdatenaufbereitung

The actual Meßdatenaufbereitung is realized by a software on the microcontroller (C167) expires.

Signals from sensors are recorded and edited, depending on the switch settings will be different operating modes, and forwarded to multiple LEDs (i.e. , the LEDs show the current mode). The edited data is then given to the serial interface.

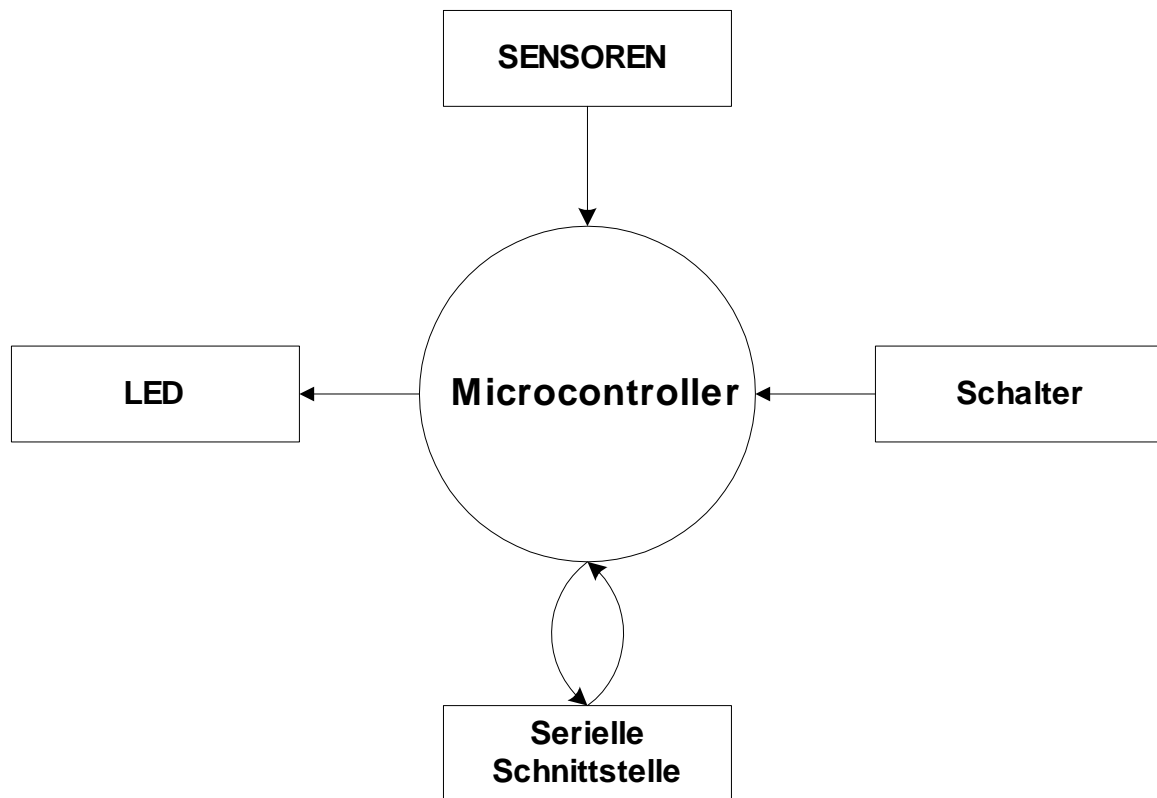


Fig 25

States and state transitions of the program

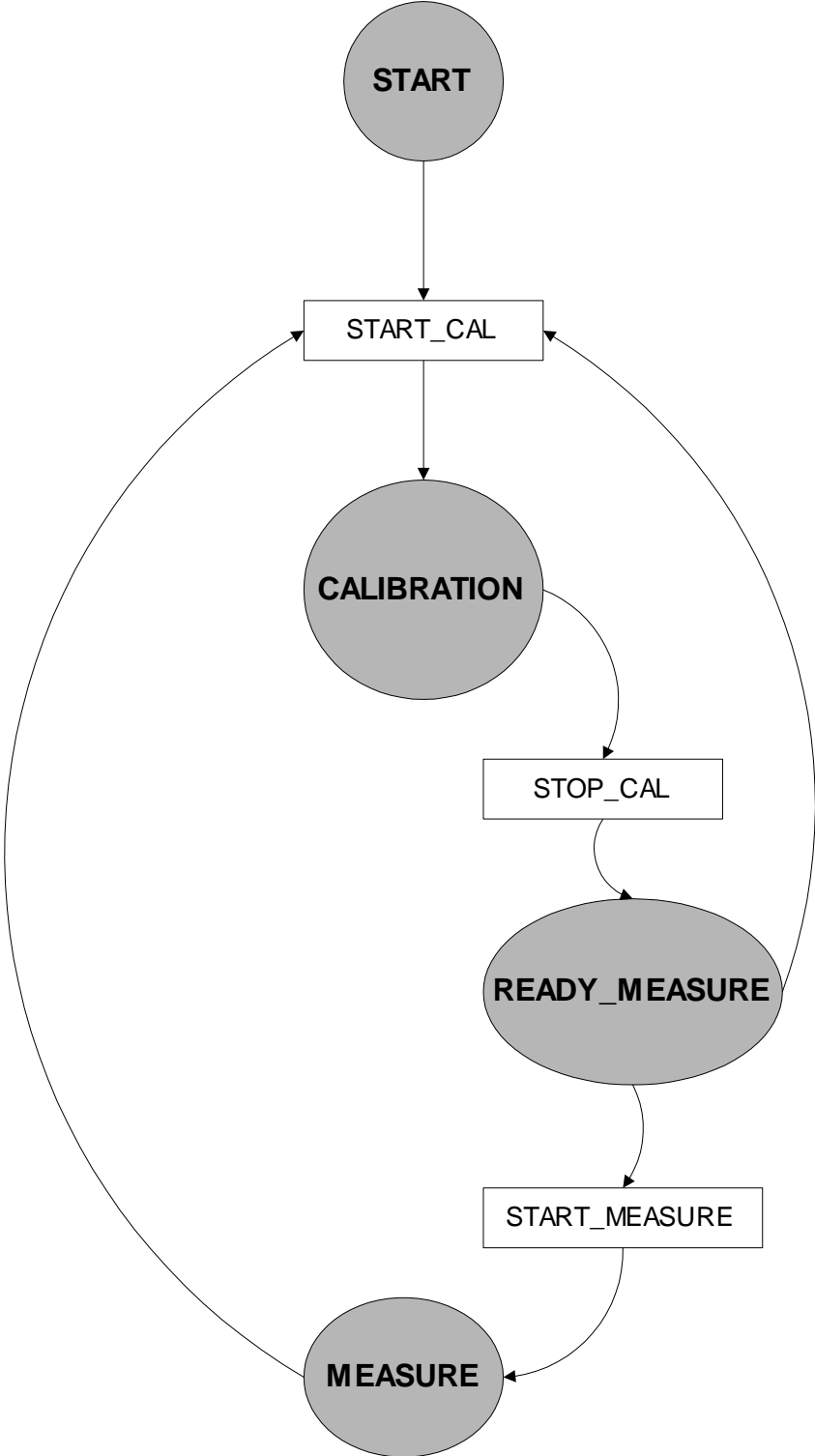


Figure 26

CALIBRATION

In this condition, the initialized and/or calibrated sensors. In the acceleration sensors is the period on the DCM-output measured. The IMU must once again to the x-, Y-, and z-axis will be rotated, this will determine the force of gravity, the later in the Gravitationskompensation is used.

The system can be moved in the Calibration-Zustand, by the START_ Calibration-Befehl there is, after turn of the IMU can then this state with a stop Calibration-Befehl be terminated after calibration, the system in the READY_MEASURE-condition.

READY_MEASURE

This condition is considered a transition state, it will be no data from the sensors detects. This operating mode is used only for the preparation of the next status. With the command START_MEASURE the system goes to the next state.

The condition READY_MEASURE can only be achieved if the system has already been calibrated, which means that the condition is never reached, if the system has not yet been calibrated, which makes it the faulty measurement of the sensors prevent another transition in the calibration state calibration is possible, however, in the man the Start_calibratoin-command is there.

Measure

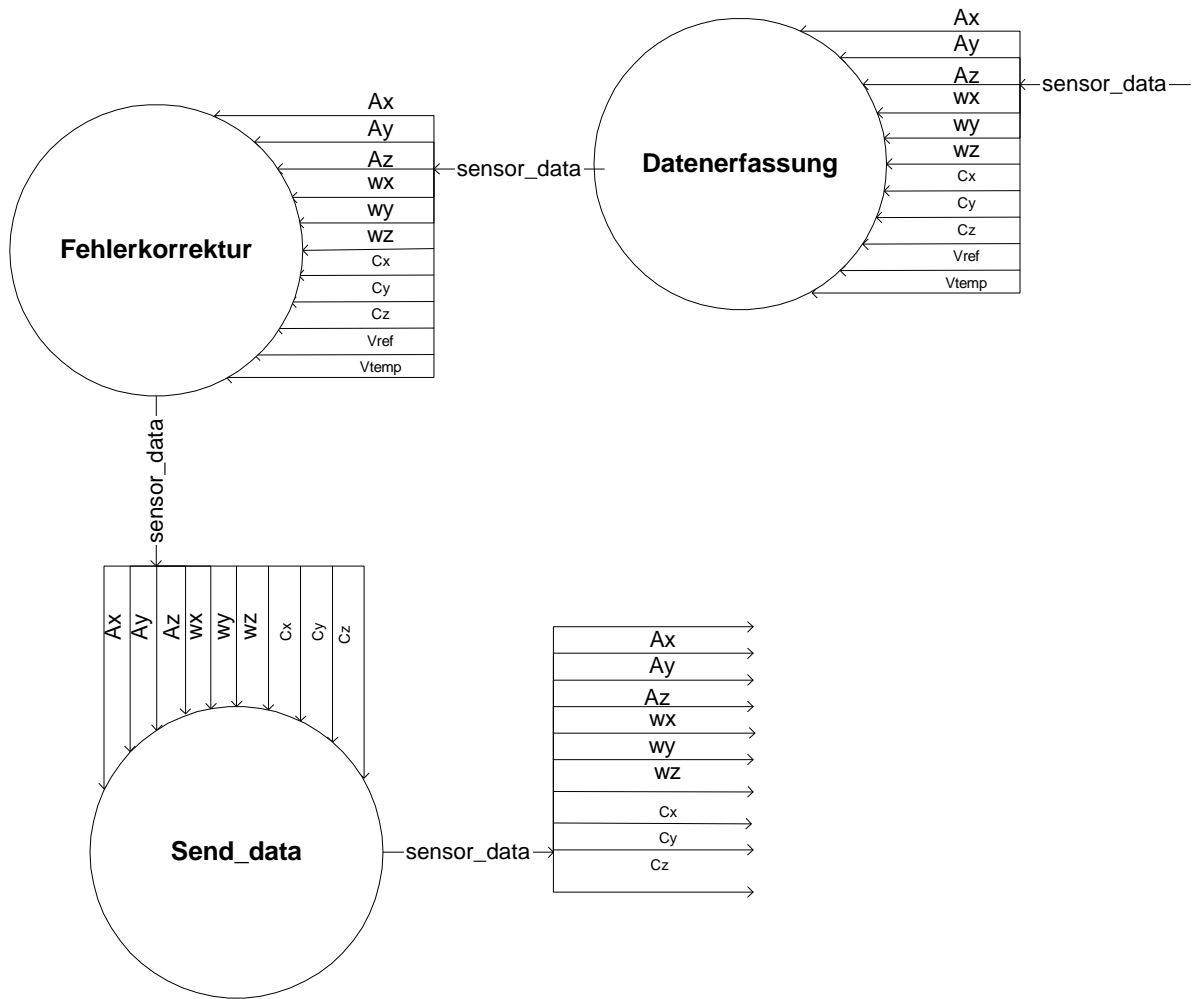
In this condition in certain periodic cycles, is the data collected by the sensors, after error correction, and calculations (Meßdatenaufbereitung) are the data obtained on the serial interface given. This you get the current measured data on the serial interface.

Analysis and Design with SA (Structured Analysis) / SD (Structured Design)

The program processes the data from the sensors and then you are to the serial interface, the user can use serial interface defined commands to the microcontroller to the program of a condition to put in the other (see Figure 26

You can also reach a transition of the program, in which various switch, the with pins of the microcontroller are connected, actuated. Whether the command over the serial interface or the of a switch has a higher priority, can be set a separate switch, should be in the General commands via the serial interface have priority.

The processes of the system are in the following SA-graph (circles mean functions or processes, arrows mean data flows):



8.5.4.2 User Interface

The user interface is used for the test case for this purpose, to communicate with the microcontroller, so between the different modes can be switched.

```

+***** REMOTE MEASUREMENT RECORDER using C167 *****+
| This program is a simple Measurement Recorder. It is based |
| on the 80C167 CPU and controls the data processing from the ADXL |
| 210 (Accelerometer) and the voltage on the Gyro. |
+-----+-----+-----+
| command | syntax | function |
+-----+-----+-----+
| Show    | D      | Show current position |
| Clear   | R      | Set Position to Null  |
| Start Calib. | C      | Start the system in Calibration mode |
| Stop Calib. | E      | Stop Calibration mode |
| Start   | S      | start measurement recording |
+-----+-----+-----+

Command:
    
```

Fig 27

The user has the option, the system with five different commands to affect:

Show

The formula $s = \frac{1}{2} a t^2$ the user receives in the primitive test case the current position in relation to a starting position, in which the Board at the start of the program was. This command can only be executed when the system is already calibrated and has been started.

Clear

Sets all readings back to zero, so a new measurement can be started, if you have already been calibrated.

Start calib.

The system is brought into the calibration mode, the sensors must now by hand to the rotary axis (X-, Y-, and Z-axis) be rotated once.

Stop calib.

Let the program go to a condition when loaded and ready to launch.

Start

Starts the Measurements

Collection of data of the acceleration sensors

In the collection of the accelerometer data a special procedure is necessary.

Roundabout, Kompaßsensoren and temperature and/or reference voltage are on to the AD-converter of microcontroller connected, and the data that will be without a special routine converted into a digital form.

Decoding of the outputs of the acceleration sensors

The outputs of the acceleration sensors are in the form of digital signals before. This is used for decoding a special routine necessary.

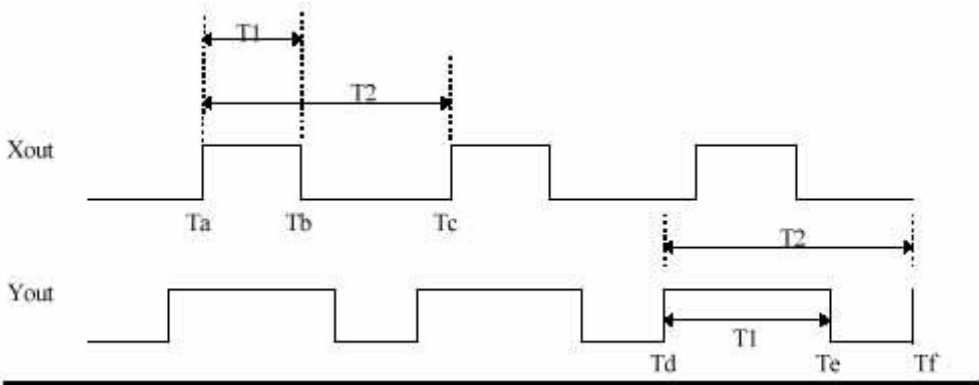


Fig 28

To the two components of a accelerometer to determine, you need to have the Signallangen of T1 and T2 of the two outputs of the sensor (see Fig 28 = 0) started. The count on the signal

covered (TB) is stored during running timer, the contents of Timer is on the following rising edge of the x-output (TC) is stored and, at the same time stopped, a process which will then for the Y-output repeatedly (TD, TE and Tf).

T1 and T2 are then easily calculated with the following formulas.

$$T1 = Tb - Ta \quad (8-11)$$

$$T2 = Tc - Ta \quad (8-12)$$

In normal use, if no high accuracy is required, you can the acceleration with the following simple Naherungs-Formel calculate (see [4]):

$$\text{Beschleunigung (in g)} = \frac{\left(\frac{T1}{T2}\right) - 50\%}{4\%} \quad (8-13)$$

Calculation of the acceleration with high accuracy

With the given "low cost" sensors can be a high level of accuracy only achieved by a suitable software will be in accordance with the data sheet of ADXL210 is to achieve a higher level of accuracy, the following routine to use in the process (see [4]):

At the beginning the system should be placed in the Kalibrier-Zustand, and then with each component of the Beschleunigungs-Sensors be made the following:

The Komponenten-Richtung lies horizontally, then slowly to the sensor is full 360o rotated, and in such a way that the Komponenten-Richtung the zenith (direction that is perpendicular to the top) cuts. If the Komponenten-Richtung in the Zenit shows, should measure the accelerometer -1g, according to +1g, if the Komponenten-Richtung vertically downwards.

Meaning of the variables in the following formulas:

T2cal	The value of T2 during the calibration.
Zcal	The 0 g of T1 during the calibration.
Bit scale factor	Bitnormierungsfaktor
K	Scale Factor

$$Z_{cal} = \frac{T1_{max} - T1_{min}}{2} \quad (8-14)$$

$$K = \frac{T2_{cal} * bit\ scale\ factor}{T1_{max} - T1_{min}} \quad (8-15)$$

The current acceleration can be calculated as follows:

$$Beschleunigung = \frac{K * (T1 - Z_{actual})}{T2_{actual}} \quad (8-16)$$

Where:

$$Z_{actual} = \frac{Z_{cal} * T2_{actual}}{T2_{cal}} \quad (8-17)$$

8.6 Test Results

In this chapter are test results of tests of the IMU-presented overall system, and the first thing will be results of the tests described, then Mikrocontroller-Programm tests to be described.

8.6.1 Hardware-Test

After all boards were manufactured, some of the tests were carried out before the system was put into operation, in this test is the measurement of the output voltages of Gyros and to measurement of DCM-signals that are produced by the acceleration sensors.

An input voltage of 5 volts to the circuit board is connected, the voltages on Gyros and other components are measured with the voltmeter, while in the acceleration sensors the Signallangen be displayed with the oscilloscope.

In general, this test, the function of each of the components tested so you can determine whether the sensors are working or not.

Test Description	Result	Large
Roundabout No1	2.4	Volts
Roundabout No2	2.4	Volts
Roundabout No3	2.3	Volts

Table 4

The acceleration sensors also functioned.

8.7 Test of the overall system with the microcontroller (C167)

After the successful Hardware-Test was the collection of data by using the microcontroller (C167) tested the company PHYTEC. This is not the developed in the present work microcontroller board, but a testboard contains of Phytec. This evaluation module has been with the developed in the present work connected sensor boards.

In the test were the Signallangen every 100 ms (see Fig 28

8.7.1 Gesamtsystem-Test 1

The IMU rested this series (i.e. , no movement has been run).

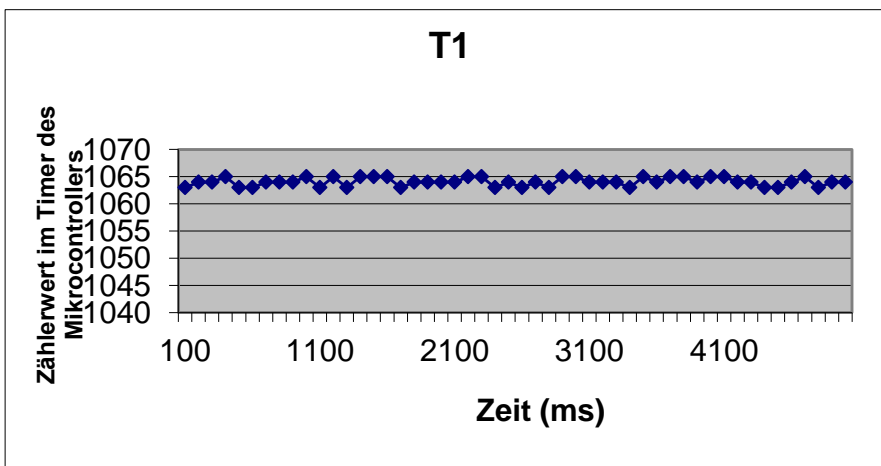


Fig 29

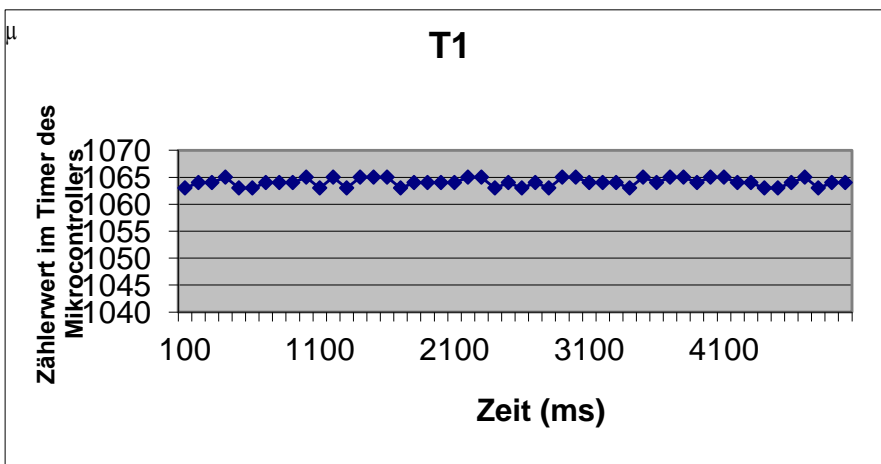


Fig 30

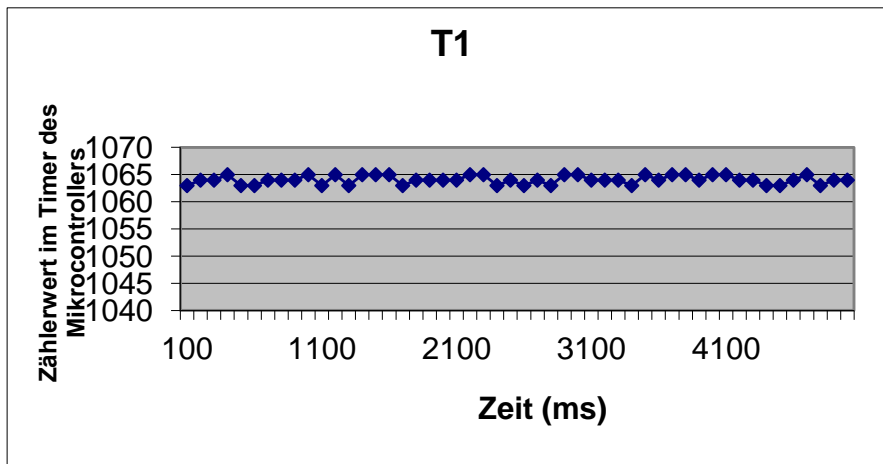


Fig 31

Although the sensors are not moving, were the data collected did not constant, the deviations to the constant value are ± 1 counters (i.e. , about 0.1 %). This is located on the timer of the microcontroller while the clocking.

8.7.2 Gesamtsystem-Test 2

In this attempt, the above error measured by averaging the data be reduced:

The period T1 takes 1ms in test1 was sampled every 100 ms, i.e. each hundredth value of the sensor has been detected in test2 are now all 100ms the first ten values (in the distance of 1ms) detects and of these ten of the mean value is formed, this average is now considered the (every 100 ms) sampled value.

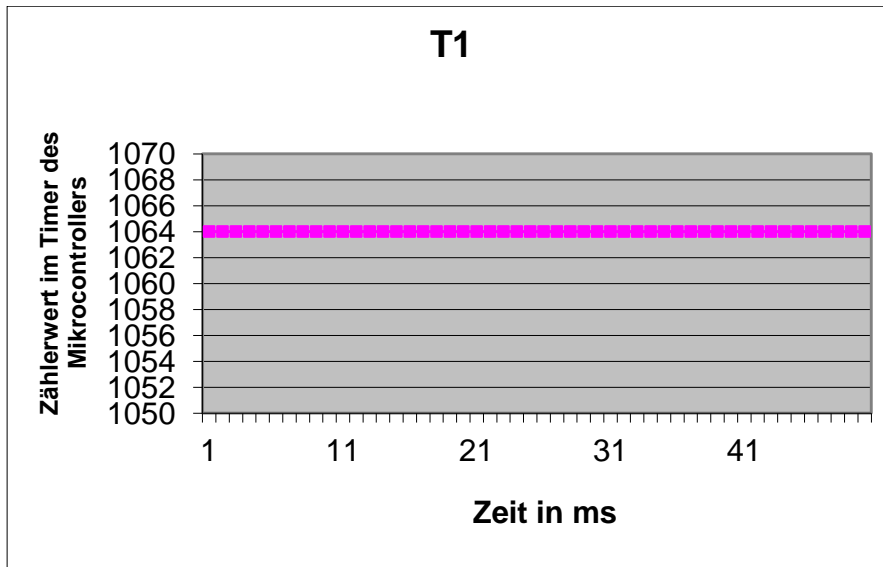


Fig 32

8.7.3 Gesamtsystem-Test 3

Here the data of the roundabout in a similar manner, it showed that coverage without averaging (according to Gesamtsystem-Test 1 for the acceleration sensors) the values of the dormant roundabout more of a constant value which differed (approximately 0.5 %). A averaging was not made.

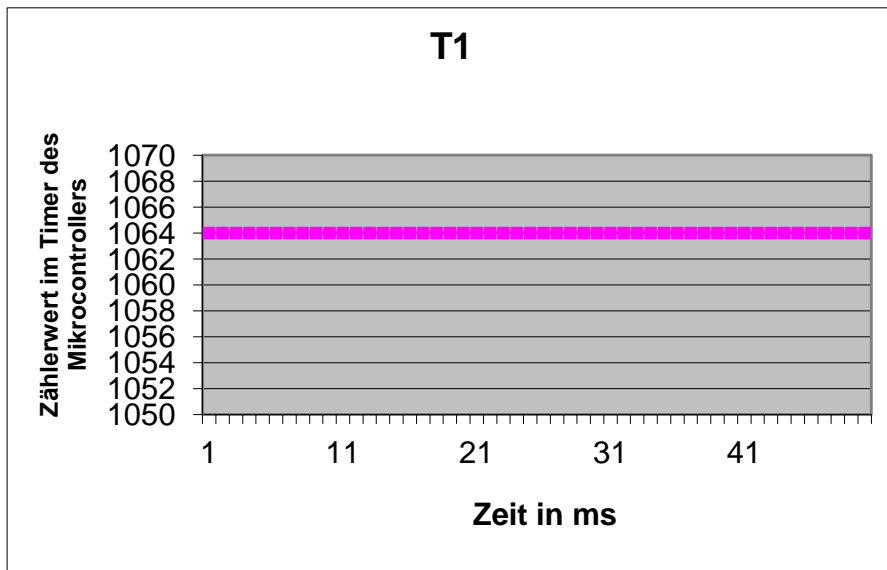


Fig 33

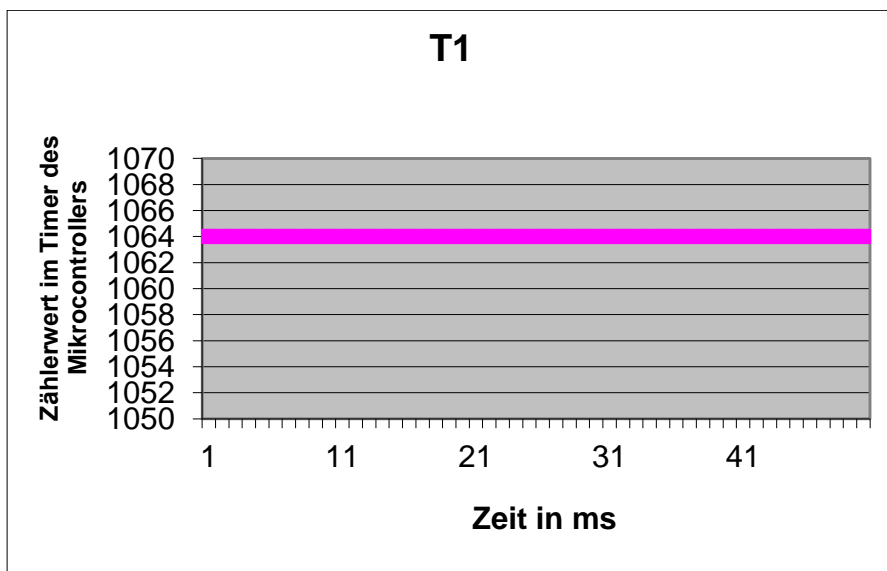


Fig 34

8.8 Summary and outlook

It was a partly functional IMU finished, with only the microcontroller to Meßwertaufbereitung must still be tested for functionality, and the first results of the tests are described in the Chapter 6 under this test is simple tests in dormant sensors, it is not yet to the Error Correction him part of a temperature change of the sensors, the Kompaßsensoren have not yet been tested, because the time was in very short.

The Mikrocontrollerboard could not be tested because some problems occurred during the commissioning of the microcontroller, these issues have been corrected, but it could not be carried out more software testing.

In order to assess the system as a whole, had to be made many more test, with ongoing uptime would, however the differences especially in the roundabouts accumulate, but it became so in regular, not again to large intervals an updated, with the correct reference value (for example, through a GPS-system) the IMU offsets, i.e. , the next step would be the further development to a hybrid system.

Literature

- [1] BROCKHAUS, Rudolf: "Flugregelung"; Springer Verlag; Heidelberg u.a. 1994
- [2] DOROBANTU, Raul: "Simulation des Verhaltens einer low-cost Strapdown IMU unter Laborbedingungen"; Technische Universität München; München.
- [3] MOSER, Luethi/ MOSER, Thomas: "Low Cost Inertial Navigation System"; Electronic Laboratory of Swiss Federal Institute of Technology, Zurich. 2000
- [4] WEINBERG, Harvey: "Using The ADXL202 Duty Cycle Output"; NORWOOD, MASSACHUSETTS.²
- [5] "Low Cost + 2g/+ 10g Dual Axis Accelerometers with Digital Output"; NORWOOD, MASSACHUSETTS³
- [6] CARUSO, Michael J: "Applications of Magnetoresistive Sensors in Navigation Systems"; Honeywell Inc⁴

² Technical data sheets of Analog Device
of <http://products.analog.com/products/info.asp?product=ADXL202>

³ Technical data sheets of Analog Device
of http://www.analog.com/productSelection/pdf/ADXL202_10_b.pdf

⁴ Application note from Honeywell Inc. of
<http://www.ssec.honeywell.com/magnetic/datasheets/sae.pdf>

C: \Documents and Settings\AdiK\Desktop\subhan\Report\Annex A. doc

C: \Documents and Settings\AdiK\Desktop\ \portnumber subhan\Report.doc

C: \Documents and Settings\AdiK\Desktop\subhan\Report\Annex C. doc

C: \Documents and Settings\AdiK\Desktop\subhan\Report\Annex D. doc

C: \Documents and Settings\AdiK\Desktop\subhan\Report\Annex E. doc

9 Actuator Board

Development of a Aktorik-Ansteuerungseinheit of an airship

Jamal E.

Diploma Thesis

2002

IFR

University of Stuttgart

Institute for Flight mechanics and control

Table of Contents

INHALTSVERZEICHNIS I	175	
ABBILDUNGSVERZEICHNIS III	177	
1 INTRODUCTION 1	181	
THE SOLARLUFTSCHIFF LOTTE 1.1	181	
1.2 TASK 2	182	
1.3 MEGA PIXELS OVERVIEW 2	182	
2 BASICS 4	183	
2.1 REAL-TIME SYSTEMS 4	183	
BORDER COOPERATION • 2.2 HARDWARE 4	183	
2.2 .1 components and classifications 4	183	
2.2 .2 The microcontroller family C1666 -	184	
2.2 .3 The C167 microcontroller family 7	185	
2.2 .4 The memory organization of the C1679 tool	186	
2.2 .5 The interrupt system for the C16711	188	
2.2 .6 The Timereinheiten 14	190	
2.2 .7 Capture Compare Unit (Capcom) 15	191	
2.2 .8 The Pulsweiten-Einheit PWM 16	193	
2.2 .9 Analog Digital Converter (ADC) 18	194	
2.3 CONTROL OF SERVO MOTORS 20	195	
3 ARCHITECTURE DESIGN OF THE AKTORIK-ANSTEUERUNGSEINHEIT (ENGL, ACTUATOR CONTROL UNIT (ACU) 22	197	
VERSION 3.1 VOLTAGE REGULATION 22	197	
REALLY 3.2 BETRIEBSSPANNUNGSUBERWACHUNG 23	198	
3.3 BASISBETRIEBZUSTANDS-EINSTELLUNG 23	198	
3.4 STEUERSIGNAL-ERZEUGUNG 24	198	
ANOTHER 3.5 NOTSTEUERUNGSBAUGRUPPE 25	199	
PINHOLE 3.6 OPERATING STATES OF THE ACU 26	200	
3.7 MODULARITY OF THE ACU 26	200	
4 DEVELOPMENT ENVIRONMENT 29)	202	
4.1 HARDWARE-ENTWICKLUNGSUMGEBUNG 29	202	
4.2 SOFTWARE DEVELOPMENT ENVIRONMENT 30	203	

5 REALISATION OF THE ACU	31	204
5.1 VERSORGUNGSSPANNUNGS-STABILISIERUNG	31	204
5.1 .1 <i>circuit design for the Versorgungsspannungs-Stabilisierung</i>	32	
205		
5.2 WAITING BETRIEBSSPANNUNGSUBERWACHUNG	34	207
5.3 BASISBETRIEBZUSTANDS-EINSTELLUNG	36	208
5.3 .1 <i>circuit design for the Basisbetriebszustands-Einstellung</i>	38	
211		
5.4 STEUERSIGNAL-ERZEUGUNG	40	212
5.4 .1 <i>The hardware of the Steuersignal-Erzeugung</i>	42	
214		
5.4 .2 <i>The monitoring and the forwarding of the PWM-signals from the remote control (normal B)</i>	42	
214		
5.4 .3 <i>Notsteuersignal-Erzeugung stage</i>	149	
219		
INTERNATIONAL 5.5 NOTSTEUERUNGSBAUGRUPPE	51	222
5.5 .1 <i>circuit design for the monitoring of the control signals on the output of the ACU and the channel5-output of the Fernsteuerempfangers</i>	51	
222		
5.5 .2 <i>circuit design for the Notsteuerungsbaugruppe</i>	52	
223		
5.5 .3 <i>circuit design for the forwarding of the PWM signals from Fernsteuerungsempfänger (Luftschiff-Startphasen -Operation)</i>	53	
224		
5.6 LAYOUT DESIGN OF BOARD 1 ("VOLTAGE REGULATION", " BASISBETRIEBZUSTANDS-EINSTELLUNG " AND " NOTSTEUERUNGS-BAUGRUPPE ")	54	224
& 5.7 THE BOARDS OF THE ACU	55	225
6 EXPERIMENTAL RESULTS	56	227
CONSTRUCTION OF THE 6.1 Ø ¹ AUTHORISATION TESTPLATZES	56	227
6.2 EXPERIMENTS AND TEST	57	228
6.3 THE ORDERED TEST	57	228
6.3 .1 <i>Test articles</i>	159	
229		
6.3 .2 <i>Test</i>	260	
230		
6.3 .3 <i>Test</i>	361	
231		
6.3 .4 <i>Test</i>	461	
232		
6.3 .5 <i>Test 562 New</i>		
233		
7 SUMMARY AND OUTLOOK	64	235
LITERATURE	65 RATING	236
APPENDIX A	66 MOTORWAY	237
ANNEX B	69	240

List of figures:

Figure 1: An airship	2	181
2: Overview of different processor families	5	184
3: Block diagram of the C167he family	8	186
4: Memory organization of the C1679 tool		187
5: Layout of a Interrupt Control register of the C16712		189
6: The building of a Timer-Einheit	14	191
Fig 7: Timer T015 hand brake		192
Fig 8: Control register PWMCON017		193
Fig 9: Control register PWMCON017		193
Fig 10: The ADC circuit diagram	19	195
Figure 11: shows the Servo in 3 different states	21	196
Fig 12: The architecture of the actuator control Unit (ACU) of the "alternative Lotte"	22	197
Fig 13: States and their transitions of the ACU. remarks, see sections 3.1 to 3,526		200
Fig 14: architecture of the ACU (detailed description)	28	201
Figure 15: development environment for the hardware development; EAGLE Layout Editor	29	202
Figure 16: development environment for the software development; μ Vision2 Text Editor	30	203
17: Block diagram for the " Versorgungsspannungs-Stabilisierung "	31	204
Fig 18: circuit " Versorgungsspannungs-Stabilisierung "	32	205
19: The block diagram for Betriebsspannunguberwachung	34	207
Figure 20: for the Betriebsspannungsuberwachung Programmablaufplan on the μ C35		208
Figure 21: Block Diagram " Basisbetriebszustands-Einstellung " with Environment	36	209

Figure 22: state diagram for the manifold of the " Basisbetriebszustands-Einstellung " (see fig.13 in section 3.6 above)	37	209
Figure 23: Part 1 of the switch between Luftschiff-Startphasen -operation and normal operation (LSB/NB-switch, part1)	38	211
Fig 24: Part 2 of the switch between Luftschiff-Startphasen -operation and normal operation (LSB/NB-switch, part2)	38	211
Fig 25: Block diagram of the " Steuersignal-Erzeugung "	41	214
Fig 26: finite state machine for the " Steuersignal-Erzeugung ". (See Figure 13 in section 3.6 above)	42	214
Fig 27: for the Aktorik-Ansteuerungsprogramm Programmablaufplan on the μ C43		215
28: Capture mode	45	217
Fig 29: T0 and CC1 in Capture mode	47	218
Fig 30: PWM-production in the Mode PMX = 050		221
Fig 31: for the Blockdiagram Notsteuerungsbaugruppe	51	222
Fig 32: Monitoring of the control signals (PWM signals) at the output (CAKTIV) of the ACU and the channel5-output of the Fernsteuerempfangers (P2.5)	51 μ	222
Figure 33: Notsteuerungssignal-Erzeugung - level	252	223
Figure 34: One of the three generated PWM signals, which is created through the combination of the clocks of the two expands flip-flops (explanation, see text)	53	224
Figure 35: the layout design of printed circuit board	154	225
Fig 36: the two boards (1 and 2) The ACU	55	226
Figure 37: the receiver and the actuators (engine,servos) are on board 1 angeschlossen.	55	226
Figure 38: Testplatzaufbau	56	227
Fig 39:	59 PM	230
Fig 40: Test	260	231
Figure 41: Test	361	232
Fig 42: Test 4, CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU	62)	233

Figure 43: Test 5-1, CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU63) 234

Figure 44: Test 5-2, CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU63) 234

List of Tables

Table 1: Overview of the C166 microcontroller family 7 185

Table 2: interrupt resources of the C167) (Part 1) 12 189

Table 3: summarizes the inputs and outputs of the LSB/NB-switch together: 39 212

Table 4: shows the channel no., and the corresponding connections and Capture-Register 44
216

Table 5: shows the DCLS Einheit-Ausgange and their corresponding timer 49 (220

9.1 Introduction

9.1.1 The Solarluftschiff Lotte

Airships in the last years experience a renaissance, and at the University of Stuttgart was a Solarluftschiff (Lotte) has been established, the Institute for statics and dynamics of the air and Raumfahrtkonstruktion has in February 1992 the project "Solarluftboot" has started. objective was to use new materials and construction, to achieve a solar and new concepts of the steering, and navigation scheme to implement.

At the Institute for as aeroelasticity, flight mechanics and control engineering (IFR) is in this context with the methods of identification, as a model is only formed from measuring data, without physical knowledge of the object to be involved, for this procedure you need accurate data on the movement of the object to tailored to the IFR is a measurement and navigation structure, with the trajectory and the location of Lotte is captured and recorded.

One of the airships is replaced by the means a loss of gas its buoyancy, today usually incombustible Helium, formerly it was hydrogen, and the means a loss of gas because of its low density is "easier "than air and therefore generates a boost, a cubic meters of helium is about the capacity of a kilogram, you therefore requires a lot of volume, in order to to take a lot of weight, therefore, airships voluminous body, and the load-bearing capacity is increased with the 3 power of the long, it's perfect for this would be the shape of a ball, but as you but as much as possible with a large speed and small driver want to travel, is a form with a smaller resistance sought, from which the elongated fish or cigar will see figure 1, but the shape was like driftwood in the water in the air and bustle, if not lead plants were attached to the stabilization, which will be fitted with oars, which, as the ship or plane distract the flow to the change in direction and thus the controllability manufacture so you have three essential conditions: the impetus for a voluminous body as much as possible, minimiye the airodynamic resistance due to a streamlined form, carefully calculated to as much as possible the main plants and rowing.

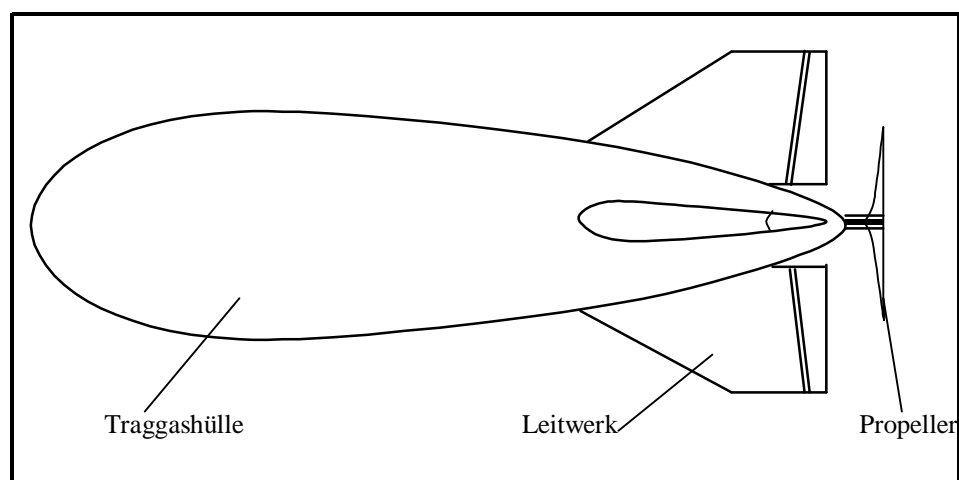


Figure 35

9.1.2 Task

For the above mentioned Solarluftschiff Lotte is the University of Stuttgart with the support of the University of Karlsruhe an alternative flight control system (flight control system - FCS) developed, which takes into account modern information technology methods.

In the present work is to Aktorik-Ansteuerung and their connection to the alternative FCS will be developed.

9.1.3 Overview

The goal of the national work it is, a Aktorikansteuerungs-Baugruppe (AABG) for the flight control system (FCS) of the airship "alternative Lotte" to realize.

First, the basics regarding real-time systems and *embedded systems* presented.

Then, the architectural design of the AABG presented.

Afterwards, the realization of the described parts, and that the draft board was carried out with the program package EAGLE. In the implementation of the Software (Microcontrollerprogramming) has been the program Vision ver, wedge used 2.03 & of the company, it is a shareware version for a limited code size of 16 Kbytes, the programming language is C.μ

Thereafter some test results of tests of the entire AABG presented.

9.2 Basics

The content of this chapter is [1], [2], [3] and [4] issued.

9.2.1 Real-time systems

For systems that are of a mechanical, electrical and data processing part, it is of crucial importance that the information processing happens so quickly, as the mechanical part of or the entire system needs, such as a plane sensors and actuators, and there are information technology units that the sensor data processing, and, according to the actuators of an implemented control loop, and it is the information processing in all your components be at least as quickly, that the actuators (Helm, etc.) get their commands in time, so that the planned rail can be flown.

This circumstance of the timeliness is called real-time, and the real-time capability of a system, or a component ensures the processing of a task within a defined time frame, and this can occur in the case of a network the guaranteed delivery of information or in the case of a operating system is the answer of a task to certain aktionsauslosende events within the defined maximum latency be.

9.2.2 Hardware

9.2.2.1 Components and classifications

The rapid technological progress causes that is nowadays an unmanageable size variety of components available on the free market is, in figure 2 is an example of a very broad overview of a small part of available given Mikroprozessorfamilien. The classification in the discretion is based on experience and have been made to be regarded rather as a grows.

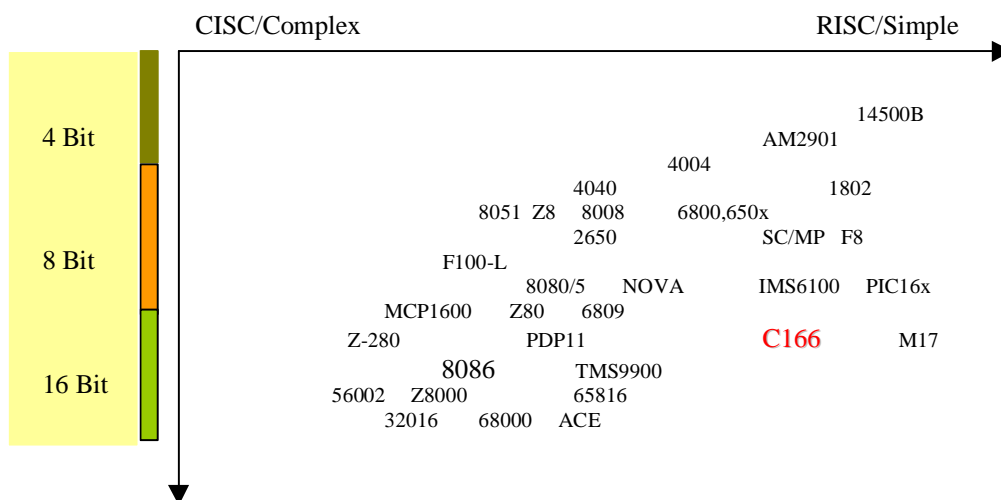


Figure 36

For the engineer in this context is often a problem of the selection and selection of an appropriate product for the task, we need to solve, for this is it I. A, necessary, and the individual products according to classify, classify and to be able to assess, and in many cases the task is mostly more difficult by the fact that many of the products of various categories merge with each other, do try to classify you different products or to compare, so you should always the target application to the fore, and try for this to find the best solution, a simple comparison with Benchmarks makes no sense, if you don't for a concrete example of the application into consideration investigated you, for example, microprocessors, these can be roughly classified according to their data bus width (16 bit, 32 Bit, ...), your scope (controller, signal processor, pure microprocessor, ...) or even after the scope of your instruction set

(RISC, CISC, ...) and of the underlying architecture, and a further, incomplete

List of criteria to be taken into consideration is the following:

- Price/piece
- Pieces, availability, compatibility
- Space, available housing types, required peripherals, etc.
- Power consumption (power supply, heat dissipation, ...)
- Development environments (programming languages, simulators, emulators,)
- Maintainability, modularity, integrity, etc.

All of these decision-making criteria are on a case-by-case basis according to be weighed and new to weights.

9.2.2.2 The microcontroller family C166

The C166he family was from the company Infineon (formerly Siemens) in the first line for time-critical control, and control tasks developed, equipped with a 16-bit wide data bus and numerous peripheral devices provides it to the user a very powerful, integrated solution is available, as typical for a controller it has a distinct Interrupt-System , many configurable I/O lines and various timer, in table 1 is an overview of the various families of the C166he given microcontroller, as is seen, are very many Members of this controller family in addition with integrated analog Digital Converters (ADC), capture Compare units, a PWM module (PWM) and serial interfaces.

C166 Family	I/O	ADC	Timer Counter	Capture Compare	PWM	Serial Interface	Watchdog	AddOn`s
C161	63-76	-/8 Ch.8 Bit	3-5	-	-	USART SSC	✓	-/I ² C
C163	77	-	5	-	-	USART SSC	✓	-
C164	59	-/8 Ch.8 Bit	5	8 Ch.	6 Outp.	USART SSC	✓	CANv2.0B
C165	77	-	5	-	-	USART SSC	✓	-
C166	76	-/10 Ch. 10 Bit	7	16 Ch.	-	2 x USART	✓	-
C167	111	-/16 Ch. 10 Bit	9	32 Ch.	6 Outp.	USART SSC	✓	CANv2.0B

Table 5

This 16-bit microcontroller family comes with only a very small number of machine instructions of the C166 has for example, only 78 asm statement, in most cases only a single machine cycle to be processed, due to the very low number of commands that the controllers can than RISC type be classified with some extensions.

The memory organization is in a *von Neumann architecture*, i.e., programs and data-sharing is a linear address space together, and each peripheral components, however, are quite modular and with a very complex networked bus system, which allows the parallel execution of internal transfers between the modules and the calculator.

9.2.2.3 The C167 microcontroller family

The C167he family is based on the architecture of the C166 family, the to 1990 by the Siemens company was newly developed in Munich, in contrast with other microcontrollers and microprocessors was in a more modern, newer concept. The focus has been on a more efficient interrupt treatment, the optimized ability, with individual bits to work quickly and efficiently, and to increased throughput between the on-chip peripherals, the memory and the CPU, put. In addition, played the expandability with other peripheral modules, a important role, in order to for various applications to obtain an optimized microcontrollers, Figure 3 shows the internal block structures of the C167he family and gives an overview of how the individual modules are linked together.

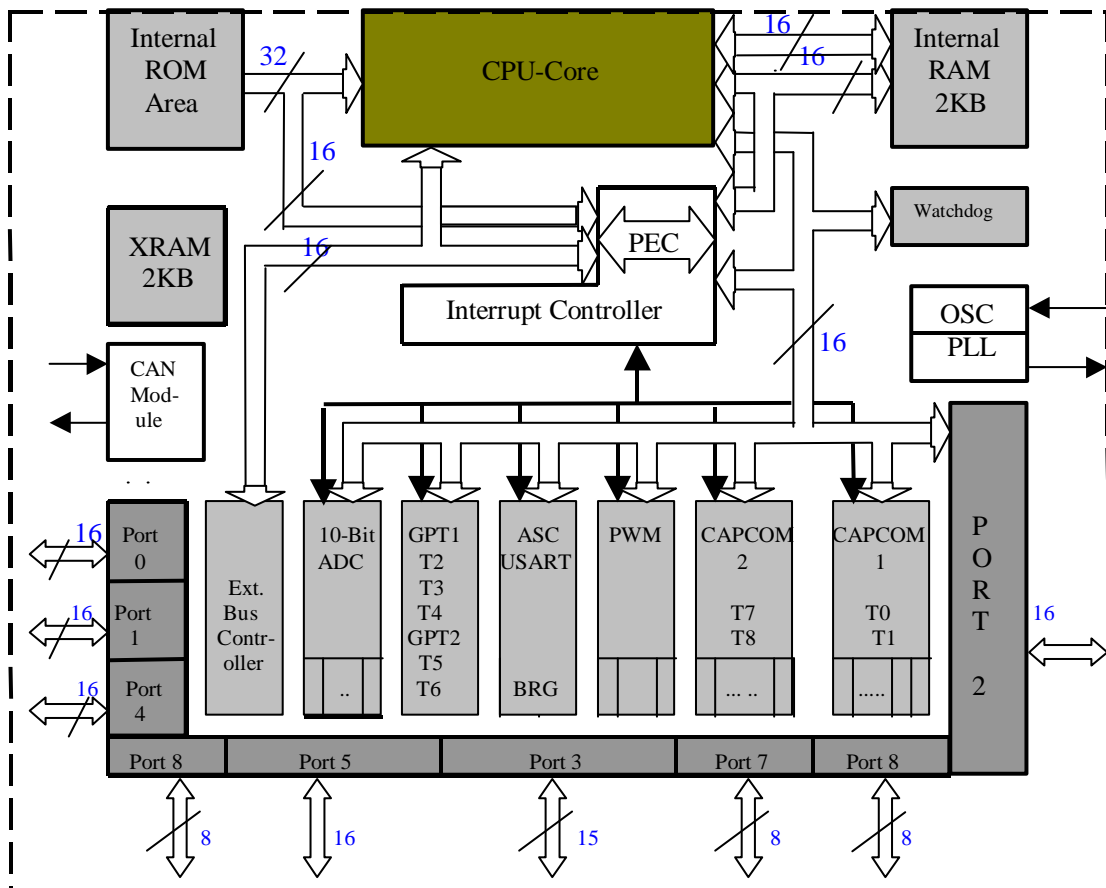


Figure 37

The most important structural elements are of the *CPU core*, compared to a slight extension of the *C166he* has experienced *CPU*, the *interrupt controller* and the *Peripherie-Einheit*. These three *Kernblöcke* determine the performance and the responsiveness of the controller to external events and in the following sections are a bit more detail.

9.2.2.4 The memory organization of the C167)

As mentioned previously, the C167) after a organized of the von Neumann architecture memory. This is through a common linear program and data memory marked of the C167) can be up to a maximum of 16 MByte addressing. The other area is divided into 256 code segments, each of which are 64 KByte large. This is another area in 1024 Data pages for each of 16 KByte organized. The code segments will be on the code segment pointer (CSP), and the data ranges of 4 data page pointer (DPP0-3) addressed. This allocation is also from the figure 4 can be seen.

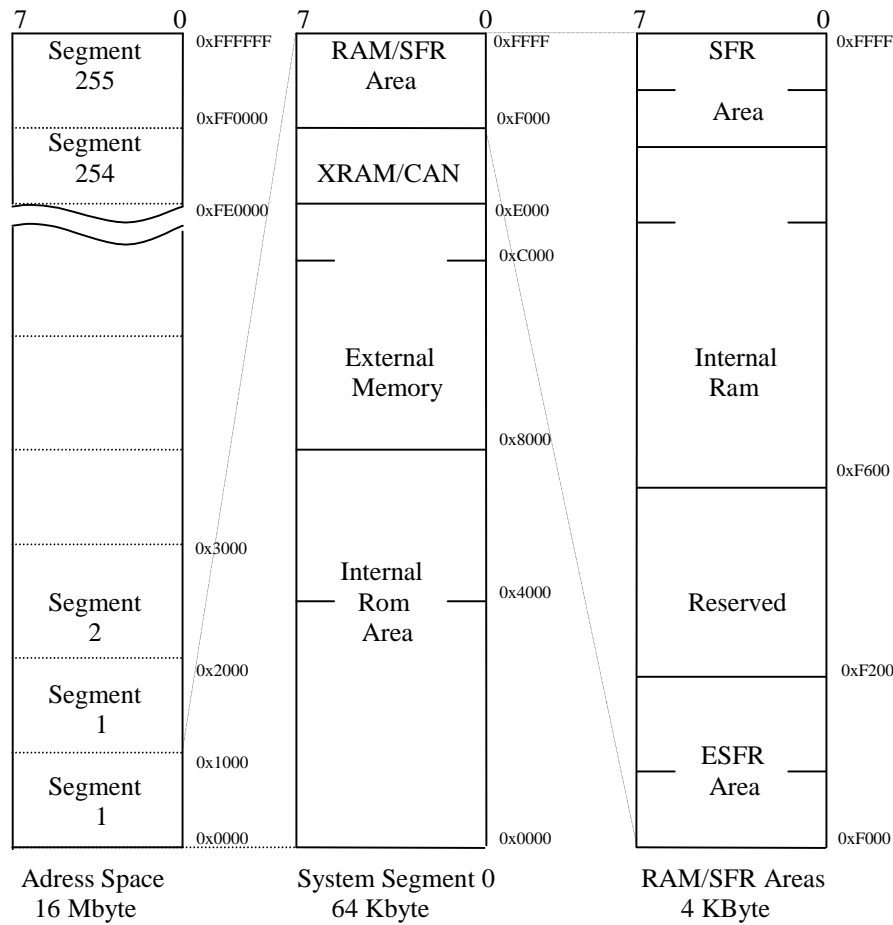


Fig 38

The segment 0 is a special case, as it registers all the peripheral modules, the General Purpose Register (GPR), the stack, the internal ROM -if any- and the internal RAM contains. From address 0x0000 in segment 0 is the internal ROM , which is also in the segment 1 can be displayed, this is during the reset process on the CPU register set SYSCON. From address 0xe000 is located in the area of the so-called XRAMs (on-chip extension RAM) or the Register of the CAN module, the XRAM behaves like a normal external memory, the required no wait states, but this will be no external bus cycles generated. In the address range between 0xf000 and 0xFFFF is located the internal RAM and the Special Function Register (SFR). This area contains all I/O register

The C167, which is remarkable, that those who register, the shaded in gray in the Adreßbereichen are (0xF100-0XF1FF, 0xFD00-0xFDFE and 0xFF00-0xffff), bit by bit-addressable, i.e. to the change of the bits is not a so-called Read-Modify -Write cycle required, as he usually when manipulating of registers by means of a bitmask is used, the internal RAM contains, the more the system stack, which, in its size in the range of 32-1024 words is programmable, and the stack grows from higher to lower addresses, addresses, and

use the Stack underflow that sometimes reduced download performance (STKUN) or stack overflow (STKOV) registers, the stack also outsourced, and thus be extended dynamically. It speaks in this case of a circular stack, and also in this area the General Purpose Register is created, the on the context pointer (CP) will be addressed, a new tab is required (for example in case of a subprogram call-up or the branch in a interrupt service time (down to -routine), it may simply by implementing of the CP on a new Registerbank be switched, this has the advantage that, in a Programmverzweigung instead all of the registers only the CP at the top of the stack Must be backed up, and also are located in this address range the Source and Destination pointer for the Peripheral Event Controller (PEC) see [1], the rest of the internal RAM is used either as a memory for variables and data, or for program code.

9.2.2.5 The interrupt system of the C167)

As the C167) microcontroller has a distinct interrupt system, which allows him to very efficiently to external and internal events react to, in principle is to distinguish between

- Normal interrupts,
- Peripheral Event Controller (PEC) interrupts,
- Trap Features (Hardware and Software traps) and
- External Interrupts.

The procedure of the expiry in the event of an interrupt will now be explained, first of all, the interrupt service routine (ISR) to create, the I. A, a subroutine is equivalent, but with the exception that for the corresponding interrupt source a defined interrupt trap number, see Table 3 , must be specified when a interrupts is then on this trap number in the Interrupt Vector table [2], with the address 0x0000 in the address space is located, called the interrupt service routine, and then the appropriate interrupt source is with a suitable interrupt priority level (ILVL) and Group Priority Level (GLVL) fitted to the C167) has 16 ILVL level, each with four group level, it can therefore a maximum of 64 different interrupt priorities be awarded multiple interrupts occur at the same time, so those with a higher wins and ILVL GLVL value.

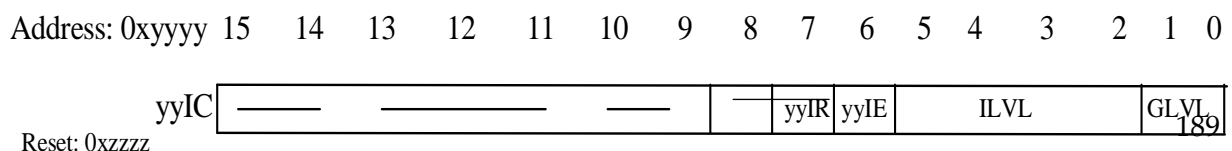
Only occurs on an interrupt, it is only the ILVL value with the CPU-ILVL value in the Program Status Word (PSW) compared, only if the value is higher, the CPU in your current valuea/valueb interrupted. After the establishment of the significance of each interrupt source

Source of Interrupt	Request	Enable	Interrupt	Vector	Trap
PEC Service	Flag	Flag	Vector	Location	Number
CAPCOM Register 0	CC0IR	CC0IE	CC0INT	0x0040	0x10
CAPCOM Register 1	CC1IR	CC1IE	CC1INT	0x0044	0x11
CAPCOM Register 2	CC2IR	CC2IE	CC2INT	0x0048	0x12
CAPCOM Register 3	CC3IR	CC3IE	CC3INT	0x004C	0x13
CAPCOM Register 4	CC4IR	CC4IE	CC4INT	0x0050	0x14
CAPCOM Register 5	CC5IR	CC5IE	CC5INT	0x0054	0x15
CAPCOM Register 28	CC28IR	CC28IE	CC28INT	0x00F0	0x3C
CAPCOM Register 29	CC29IR	CC29IE	CC29INT	0x0110	0x44
CAPCOM Register 30	CC30IR	CC30IE	C30INT	0x0114	0x45
CAPCOM Register 31	CC31IR	CC31IE	CC31INT	0x0118	0x46
CAPCOM Timer 0	T0IR	T0IE	T0INT	0x0080	0x20
CAPCOM Timer 1	T1IR	1IE	T1INT	0x0084	0x21
GPT1 Timer 2	T2IR	2IE	T2INT	0x0088	0x22
GPT1 Timer 3	T3IR	T3IE	T3INT	0x008C	0x23
GPT1 Timer 4	T4IR	T4IE	T4INT	0x0090	0x24
GPT2 Timer 5	T5IR	T5IE	T5INT	0x0094	0x25
GPT2 Timer 6	T6IR	T6IE	T6INT	0x0098	0x26
GPT2 CAPREL Reg.	CRIR	CRIE	CRINT	0x009C	0x27
A/D Conversion Comp.	ADCIR	ADCIE	ADCINT	0x00A0	0x28

Table 6 interrupt resources of the C167) (Part 1)

By the programr, this priority level (ILVL+GLVL) in the associated interrupt Control Register written (for example, T0IC = 0x0027; tells the Timer 0 Interrupt Control Register a ILVL value of 9 and a Group level from 3 to.) The general layout of an Interrupt Control register in Figure 5 is played.

Figure 39



It is this, the greater the ILVL and GLVL value the higher is the appropriate priority, after the definition of the priority is to source the appropriate interrupt enable (by setting the corresponding yyIE bits), is a PEC transfer be enabled, it is at this point the PECC Register as well as the source and destination address set. After the local interrupt release must be allowed an interrupt global. For this is in the CPU program status word register PSW the bit 'Interrupt Enable' (IEN) must be set, interrupts that occur can now be processed and approved.

The C167) only has a single *dedicated* interrupt pin - the non-maskable interrupt (NMI) - it can, however many IO pins with the associated logic be configured so that an external signal an Interrupt Request can trigger these interrupts are typically all 400ns at a speed of 20MHz is scanned, the top eight pins of Port 2 can also be used as a however *almost external interrupts* are configured, see [1] page 5-23, which can also the edge change, you want to trigger a request to be programd in this specific case, it is the scanning of the inputs all 50ns at a CPU speed of 20MHz.

In addition to the previously discussed Interrupt mechanisms there are the so-called trap functions, of which two types are distinguished in the Software trap is a trap in the valuea/valueb on function (for example: `_trap_(0x10)`) an interrupt is triggered it is processing as in the case of a normal ISR, with the exception that the Interrupt Request (IR) flag is not set and the ILVL priority value not in the Program Status Word is copied, which means that software Traps of interrupts with a lower priority are always can be interrupted, for example, the UN-ter the Hardware traps are energized because of the CPU summarized (e.g. B: mis-aligned requests, Opcode violations, etc.), these are not maskierbar and have priority over any other CPU activity within this Hardware traps there is a prioritization in different classes, depending on the severity of the error, see [1] page 5-5, and the processing of this error is the same as with a normal ISR, with the exception that there is always a ILVL value of 15 in the Program Status Word is copied.

9.2.2.6 The Timereinheiten

A timer is a hardware counter (Figure 6), although the initialized with commands, but then program runs independently and in his cause an interrupt zero-crossing can. He is used for many tasks, the software is only by program can be very costly, for example:

- As a periodic interrupt timer,
- Use as an event counter for signal edges,
- Frequency generator and frequency meter, as well as
- Pulse width modulation to control of engines (Servos).

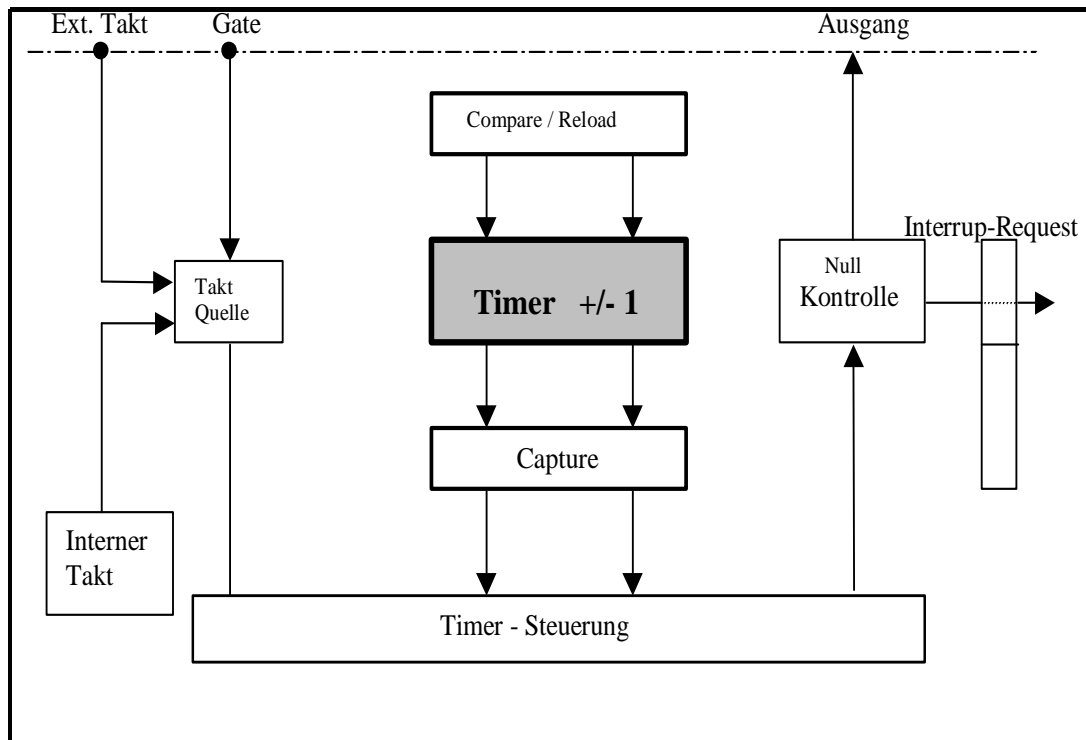


Fig 40

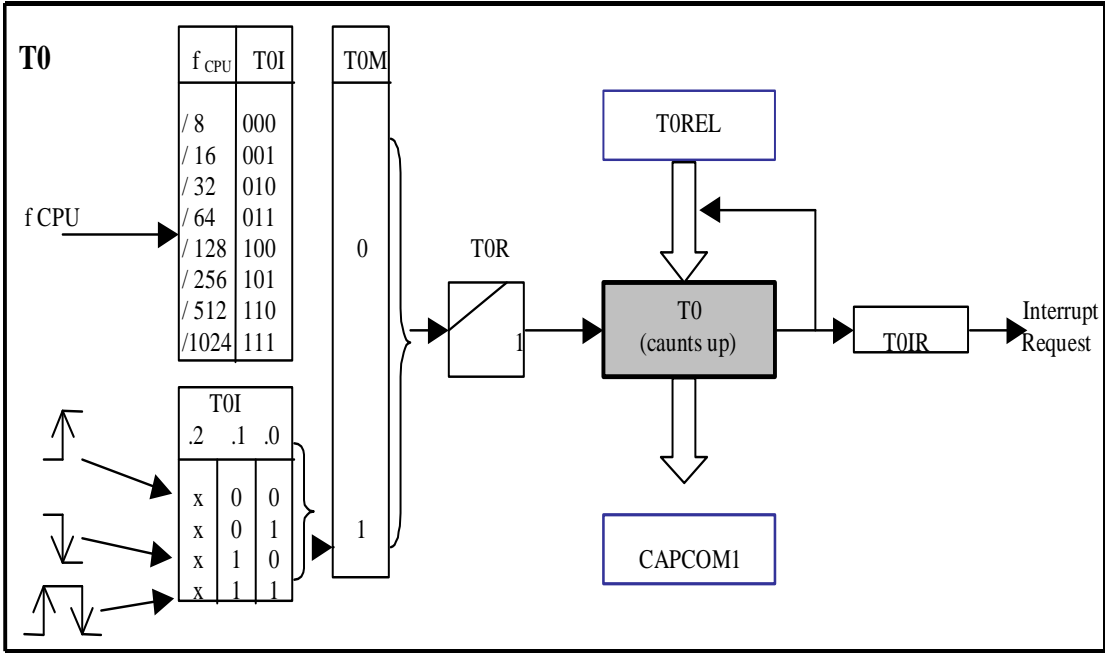
By programming the timer control the timer can be set in different modes, used as the clock source either the internal or an external signal processor, with the gate input is insulated controlled the clock is in a zero crossing of the counter can be a signal to the outside or delivered an interrupt will be triggered. Hilfsregister serve to recharge, compare, and reading of the meter reading.

9.2.2.7 Capture Compare Unit (Capcom)

The Capture Compare Unit of the C167) is used to capture and compare Signalzustanden of external digital, and can quite elegant with this module are generated pulse patterns, this module is divided into two functional groups with the CAPCOM unit 1 from the two timers T0 and T1 as well as the 16 registers CC0 to CC15 is the one of the two Timer T0 or T1 are assigned to, in the CAPCOM unit 2 work registers CC16 to CC31 with one of the two Timer T7 and T8 together in Figure 7 is a functional block diagram of the Timer T0 shown Timer T0 as a timer or counter can be used and has a separate Reload register. The timer T1 can only be used as a timer, but as well as Timer T0 on a own Reload register. The timer T7 is analogous to Timer T0 and T8 to T1 running analog.

Fig 41

In the Mode **Timer** work the timer only counts upwards with the TaktquelleIntern (getelter CPU-measure) or with the output of GPT T6, and the timer T0 and T7 have additional external flankenprogrammierbare inputs (Count-Betrieb). In each zero crossing the timer automatically Nachladeregister TxREL lengthen it from the reloaded, and it can be triggered an interrupt.



In the Mode **Capture** (field) is the current meter reading of the assigned Timer in one of the Register CCxx stored and it can be triggered an interrupt, and this is done with a programmable edge on one of the inputs CCxxIO, the external interrupt to trigger be used.

In the Mode is **Compare** (Compare) is one of the tabs CCxx on a comparator with the current count of the associated timer compared. If it matches, it will be a signal on the corresponding output CCxxIO output, and it can be triggered an interrupt.

9.2.2.8 The PWM Pulsweiten-Einheit

With this unit in the event of an emergency is constant PWM-generating signals, if the recipient for any reason no signals to Outputs further sends. This unit has 4 channels of each of the 4 channels contains its own Timer PTX, of the of the CPU-measure or a divider on the counts, the period of the signal results from a comparison of the current counter with the Periodenregister PPx. The Low-Zeit from a comparison with the Pulsweitenregister PWx.

For the setting of the measure, the use of the Interrupt mode and all 4 channels the common control register PWMCON0, PWMCON1 see figure 8 and 10, and PWMIC. For each channel there is a own Timer PTX, a own Periodenregister PX, a own Weitenregister PWX and a own output P7x.

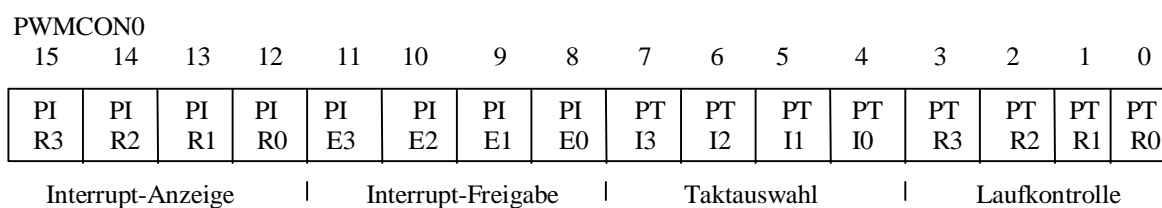


Fig 42

PTRx Timer-Laufkontrolle :0 = off, 1 = Run

PTIx Timer-Taktauswahl : 0 CPU-cycle, 1 = CPU-measure : 64

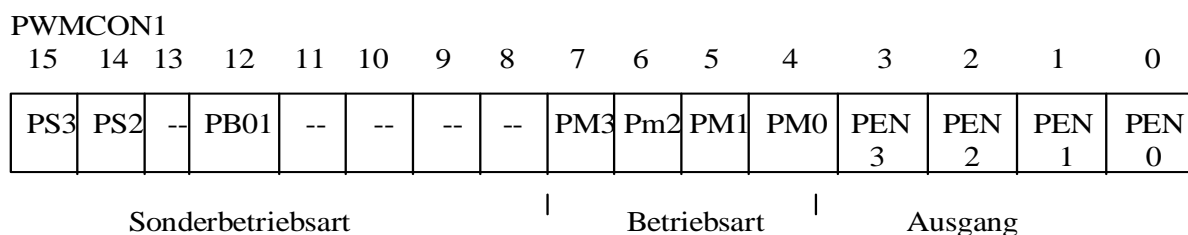


Fig 43

PENx output control : 0 output locked, 1 = output Friday

PMX Operating Mode (Mode) 0 = up counter, 1 = on-/ decremter

In the PMX = 0 Operating Mode Mode The timer counts PTX starting with the initial value of 0 only up. The counter is of the timer is equal to, or greater than the comparison value in the corresponding PWX, the output to 1 (High) Set, if the counter reaches the comparative value

of the corresponding PX, it will be in the next is kept pressed the output to 0 (low) is set and the timer with the initial value 0 started again.

9.2.2.9 Analog Digital Converter (ADC)

The C167he family has an integrated analog digital converter, see figure 11, with the following properties:

10-Bit Resolution

Wandlungsverfahren: successive approximation

Conversion time: minimum 9.6 μ s

16 Analog Input Channels

In each computer system is the so-called bit (Binary-Digit , divalent paragraph), the smallest, not more further divisible unit of information in the system, a bit can the value (logical 0), or(logical 1) Accept.

In electrical systems, these are two statuses in general shown by electrical voltage and therefore applies to, for example :

Log.1 = high = +5V

Log.0 = low level = 0V

Seen in this way it makes sense, in a first resolution step the voltage to be measured on a digital input pin of the μ Cs to connect, but unfortunately there are only two logical however it binary statuses and thus only two distinguishable also unchangeable.

Now here comes the use of the ADC, this component is nothing other than a vorschaltbaugruppe, the analog voltage signal μ C-arranges meet, i.e. in converts a numeric value of the C167-AD-converter not only converts so simply analog voltages in binary numbers, but also provides the user still 6 different modes available, see [4], with which a wide variety of measurement tasks is very easy to carry out in this work is with Auto Scan continuous conversion mode worked.

Auto Scan continuous conversion mode

In this mode, each time a whole sequence of analog channels of the series after analog-to-digital changed. In the ADCH a bitfield which is specified with the channel of the the Wandlung-Sequenz begins; it ends with channel zero, after each conversion is an interrupt request generated, indicating that, in the ADDAT Register is a current Wandlungsergebnis. In this mode, between a single and a continuous conversion mode differences.

When **Single** converts of the AD-converter will automatically all the channels from the channel down to the channel 0 and then stop and the **continuous** running analog from, only that now permanently further converts.

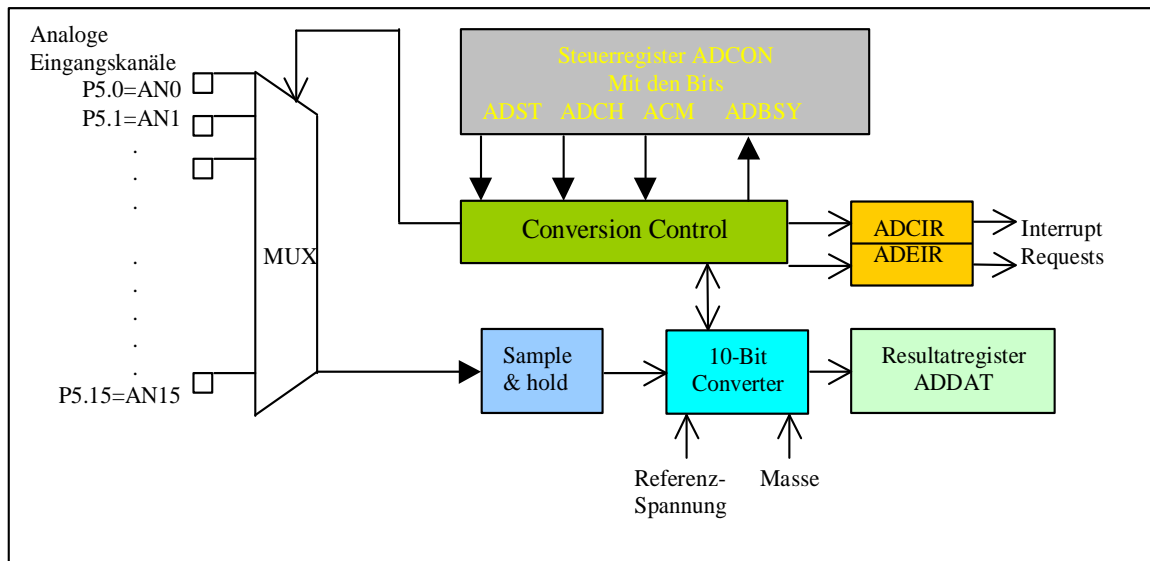


Fig 44

9.2.3 Control of servo motors

In order to with a Mikrorechnersystem as $\mu\text{C-C167}$) to also control Mechanical Systems
You can, you can rely on several concepts.

- Ωιτη στεππερ μοτορσ Αντριεβε
- Αντριεβε ωιτη ΔX μοτορσ.
- Ωιτη σερω μοτορσ Αντριεβε

Stepper motors are controlled with special Signalsequenzen to separate coils are placed these engines, so move the rotors always exactly one "step" next, mostly a few Umdrehungsgrade. About a complicated control and appropriate transmission the system may then be moved, DC motors with Mikrorechnern can once via the appropriate amplification stages be turned on and off, and on the other you can, for example, about Pulse Code Modulation set speeds.

A very different technique, the so-called servo-drives, which, inter alia, in the MODELLBAUWELT (plane, car, and model ships) are used, these are already for a few marks and have to have some impressive advantages for the amateur electronics engineers.

To work with low voltages (5V are enough), and on the other they consume little power in the hibernate. Your Ursprungszweck according to create a rotation of about 90-180 degrees to a strong reduction, i.e. very strong axis. For movements, the living with such values (e.g. , the sun tracking solar paneling, mirror tilt, steer models, etc.) are servo motors ideal, the most important ingredients are a small DC motor with gearbox, at the output is coupled mechanically a potentiometer and a Electronics, a pulslangenmoduliertes

Signal (PWM signals) evaluates, and controls the engine according to the top there is a standardized control of these servos. In the distance of 25 ms pulses are sent (in radio remote controls on the transmitter), with a length of 1.0 ms to 2.0 ms varies and so the position of the servos indicating (left and right).

Is located in the center position of the servo, it is the pulse duration 1.5 ms see figure 12. This Eingangsimpulslänge is with the pulse duration of the monostable Multivibrators compared, the of the position of the potentiometer and so that the position of the servos depends on, and only if the position of the servos the same internal pulse generated, such as the externally-scale, hears the movement to, the servo has its default position is reached, so that all of the users of a commercially available servos are still needs to do is the earth, and 5 V and connect at the entrance for a pulse sequence ensure that it is "understands". What is customary here are normal TTL-level.

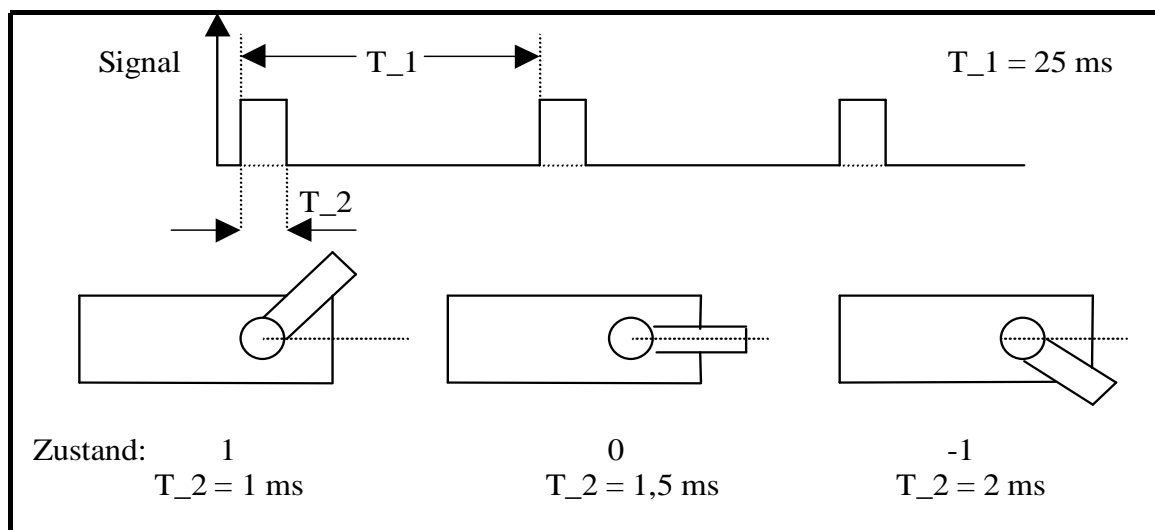


Figure 45

9.3 Architecture design of the Aktorik-Ansteuerungseinheit (engl. *Actuator Control Unit (ACU)*)

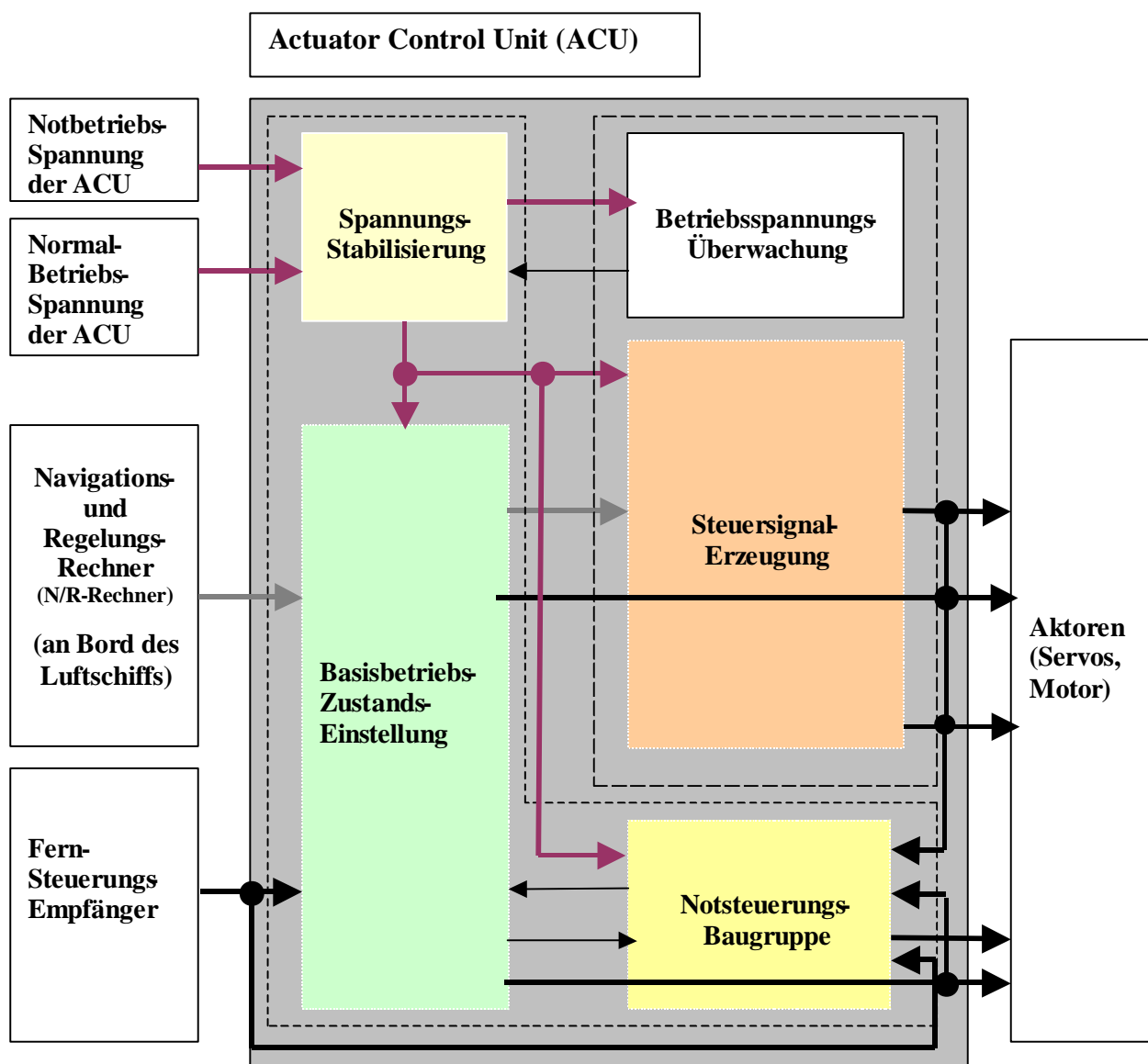


Fig 46

9.3.1 Voltage regulation

The assembly "Voltage Regulator" supplies the remaining blocks of the ACU with a constant voltage of such a level, for which the individual blocks is needed, as well as the assembly ensures that the power supply for a certain time is maintained after Normal-Betriebsspannungsversorgung not more is fully functional (by the normal operating voltage

drops below a certain level). that the normal operating voltage drops below a certain level, the "Betriebsspannungsüberwachung" found.

9.3.2 Betriebsspannungsüberwachung

The "Betriebsspannungsüberwachung" monitors, whether and when the required operating voltage falls below a particular level, in which the blocks of the ACU not more properly can be supplied. Is the block "Voltage Regulator" just in the condition "Supply of Normalbetriebsspannung" (that is from the voltage regulator to the Normalbetriebsspannung the other blocks of the ACU passed on), and then drops the Feed level requested below a certain level, there is the block "Betriebsspannungsüberwachung" a signal to the block "Voltage Regulator" that this "to be supplied with Notbetriebsspannung" switch.

9.3.3 Basisbetriebszustands-Einstellung

Using a switch of the through a separate channel of the remote control is pressed, the distributor of the " Basisbetriebszustands-Einstellung " with either **normal operation** (contains normal operation A (valid commands come from the N/R-calculator) and normal operation B (interconnection of the control signals from Fernsteuerungsempfänger)) or with Luftschiiff-Startphasen -operation be initialized when Luftschiiff-Startphasen operation, the PWM signals from Fernsteuerungsempfänger routed directly to the actuators (without the use of the microcontroller of the block " Steuersignal-Erzeugung "), during normal operation, the "Steuersignalerzeugung" from normal operation A to normal operation B toggles and vice versa, in this separate channel is constantly being sent, i.e. , the signal for normal operation or Luftschiiff-Startphasen -operation is stationary at.

During the flight can be between Luftschiiff-Startphasen -operation and normal operation be switched in both directions.

If neither control signals (PWM signals) at the output of the block "Steuersignalerzeugung" concerns, nor in the separate channel of the remote control, the between normal mode and Luftschiiff-Startphasen -operation mode, the finite state machine of the " Basisbetriebszustands-Einstellung " the system, in the Condition " Notsteuersignal-Erzeugung - level 2" (i.e. , hardware production of Notsteuersignalen) move to normal operation after emergency operation status transfer (level 2).

9.3.4 Steuersignal-Erzeugung

Of the block "Steuersignalerzeugung" generated depending on the operating condition in different ways control signals (PWM signals), the directly the actuators (engine, servos) response:

- In normal operation A receives the "Steuersignalerzeugung" control commands from the distributor of the " Basisbetriebszustands-Einstellung " and generates control signals (i.e. , PWM signals)
- In normal operation (B) are control signals (i.e. , PWM signals) from Fernsteuerungsempfänger forwarded to the actuators.

- At the system startup to the normal operation B is **initialized, if on channel 5 "Normal mode" is set, and the further operation may, at any time from normal operation B switch to normal operation** a be and vice versa, and this is done by setting from the N/R-machine from the N/R-computer receives this information on his communication system from the ground (not from the remote control).
- If in normal operation A or normal operation B due to the failure or failure of the N/R-computer and/or of the Fernsteuerungsempfangers the microcontroller no more signals to the input port of the microcontroller receives, is moved to the State "Notsteuersignal-Erzeugung - level 1 ", in which the microcontroller of the "Steuersignalerzeugung" autonomously PWM signals to the actuators generated and there is.

9.3.5 Notsteuerungsbaugruppe

In the start-up phase of the airship will be exclusively the PWM signals from Fernsteuerungsempfanger used to control the actuators, and the PWM signals but not only by the microcontroller of the "Steuersignal-Erzeugung", but by the "Notsteuerungsbaugruppe" given directly to the actuators.

If

1. No valid control signals more through the "Steuersignal-Erzeugung" be created (this may be the case, after the order of the different states in the "Steuersignal-Erzeugung" have failed, or but the "Steuersignal-Erzeugung" as a whole suddenly fails) and
2. Also no valid control signals more on the output of the Notsteuerungsbaugruppe be measured by the Direktdurchschaltung the control signals from Fernsteuerrungsempfanger Notsteuerungsbaugruppe by the stem, and
3. No PWM signals to the separate channel of the remote control concerns, the between normal mode and Luftschiff-Startphasen -operation switches to

(1., 2 and 3, in the "Notsteuerungsbaugruppe" monitors), there are the "Notsteuerungsbaugruppe" a signal to "Basisbetriebszustands-Einstellung", which automatically to the hardware Notsteuersignal-Erzeugung the "Notsteuerungsbaugruppe" mode.

9.3.6 Operating conditions of the ACU

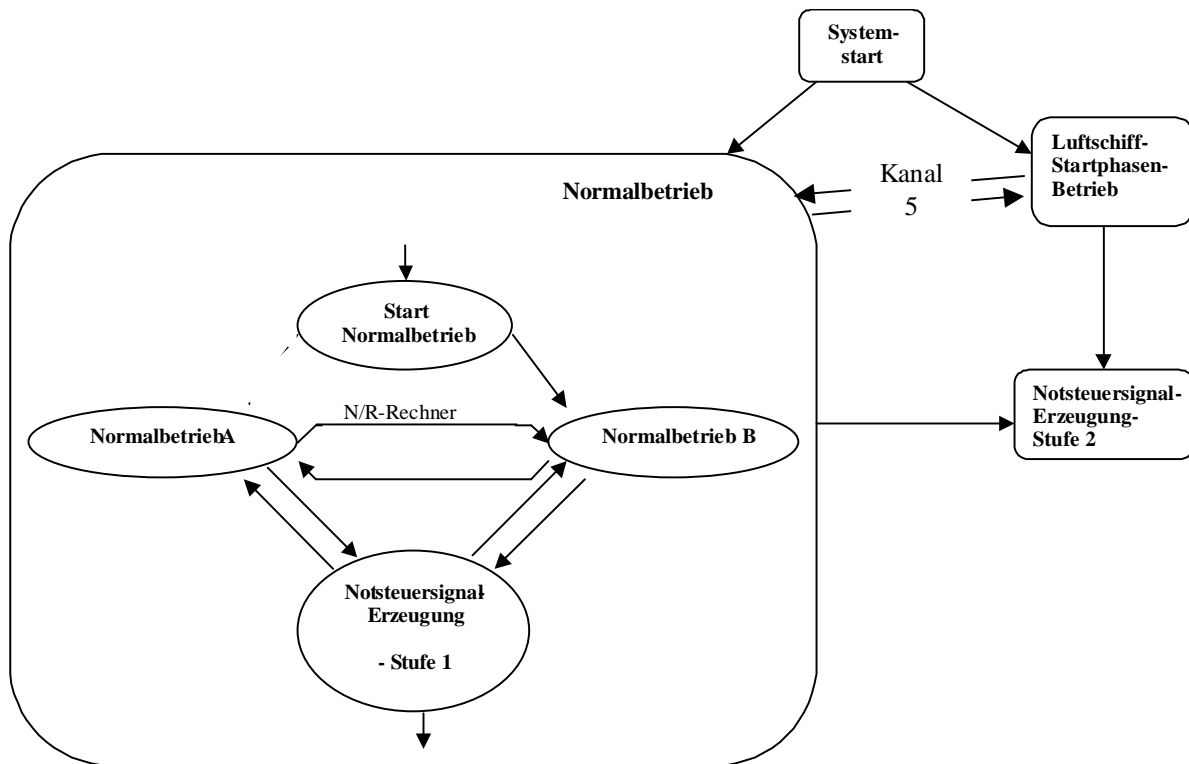


Fig 47

9.3.7 Modularity of the ACU

The blocks "Voltage regulation", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe " are realized purely hardware the blocks "Betriebsspannungsüberwachung" and "Steuersignalerzeugung" software will be with a program that runs on a microcontroller, realized.

As far as possible to the system modular design, it offers to the ACU to divide into two subsystems, each of which will be implemented on a board: a subsystem, in which the blocks "Voltage regulation", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe " are (will be realized on a printed circuit board 1), and another subsystem, in which the blocks "Betriebsspannungsüberwachung" and "Steuersignalerzeugung" are located, which has the following advantage: if the circuit board with the microcontroller fails, the ACU still work (by the Notsteuerrungsbaugruppe). The board for "Betriebsspannungsüberwachung" and "Steuersignalerzeugung" (circuit board 2) In the context of this work is not developed, already a general test board of the company PHYTEC is present, on the the two blocks are to be realized: only the software, the on the on the test board expires the microcontroller, is developed in the context of this work.

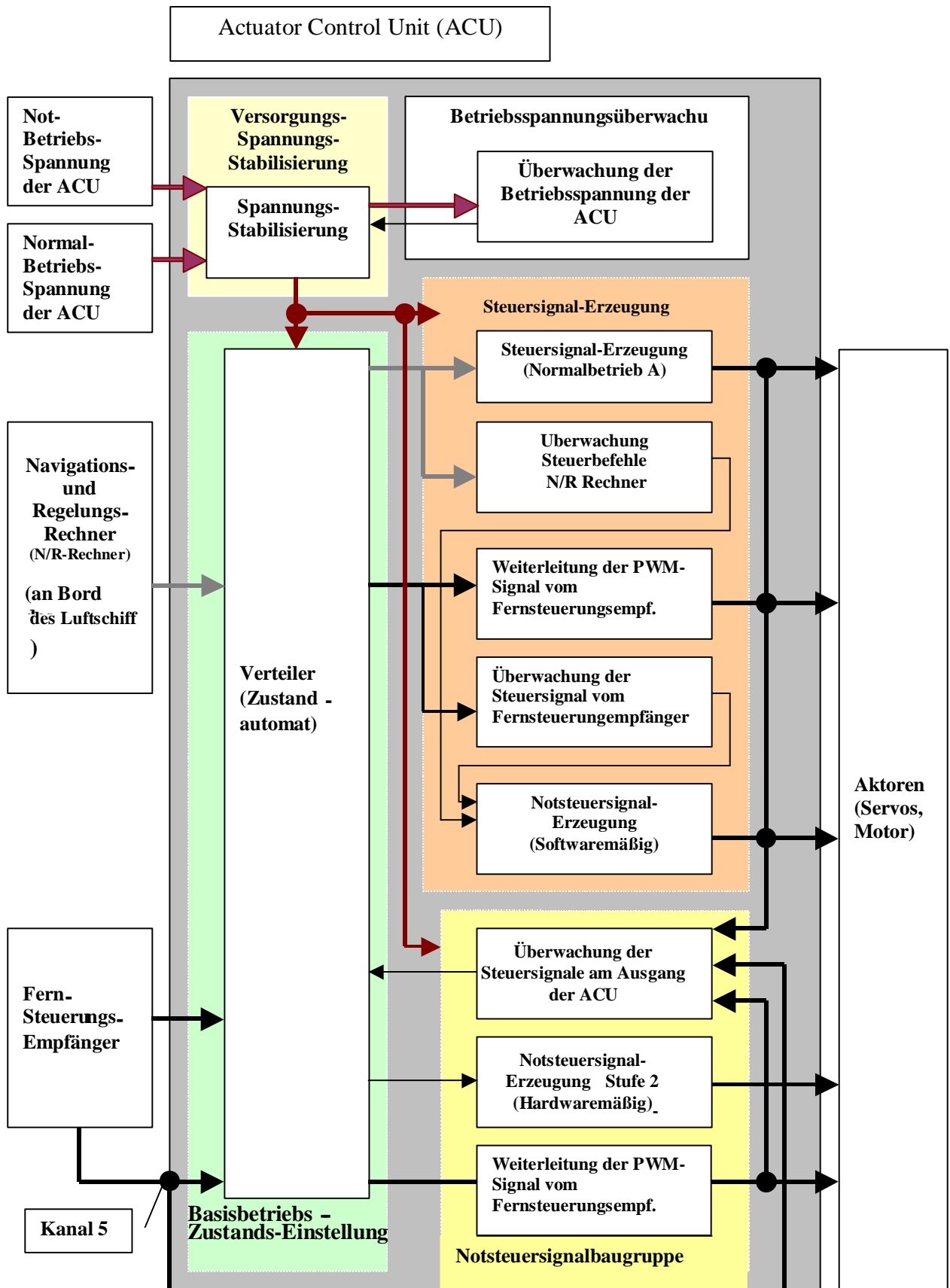


Fig 14: architecture of the ACU (detailed presentation)48

9.4 Development environment

9.4.1 Hardware-Entwicklungsumgebung

The hardware development is done with the program of the company Cadsoft EAGLE. It is the eagle-v4.0, with Windows 2000, the tool includes a **layout editor**, with which the Board for the blocks "Voltage regulation", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe " was designed, also contains a Bibliotheks-Editor EAGLE, a CAM (Computer Aided Assembly) -processor and a text editor, you can with the Bibliothekseditor housing and edit icons.

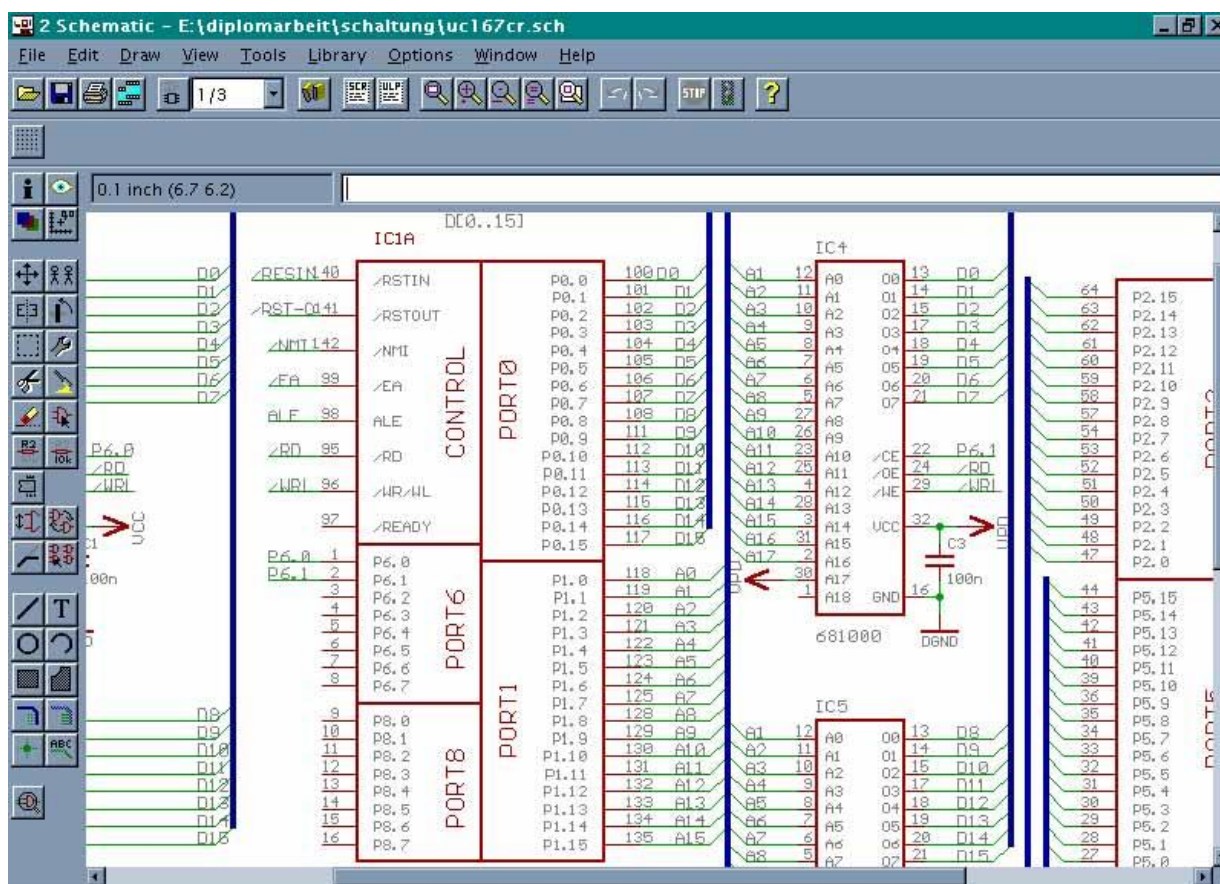


Fig 49

9.4.2 Software Development Environment

The programming of the microcontroller C166 is under development environment της χομπανψ ωεδγε μςισιον. It is a σηαρεωαρε περσιον, ωιτη ωηιχη προγραμσ υπ το 8 Κβψτε χοδε σιζε χαν βε δεπελοπεδ.

One finds in μ Vision, Version 2 the basic elements of a modern IDE:

- A specially adapted text editor.
- An ANSI-C-compiler (C166-ANSI-C-compiler),
- A Assembler (A166-Assembler),
- A left/Lokator (L166),
- A debugger (μ Vision2 Debugger)

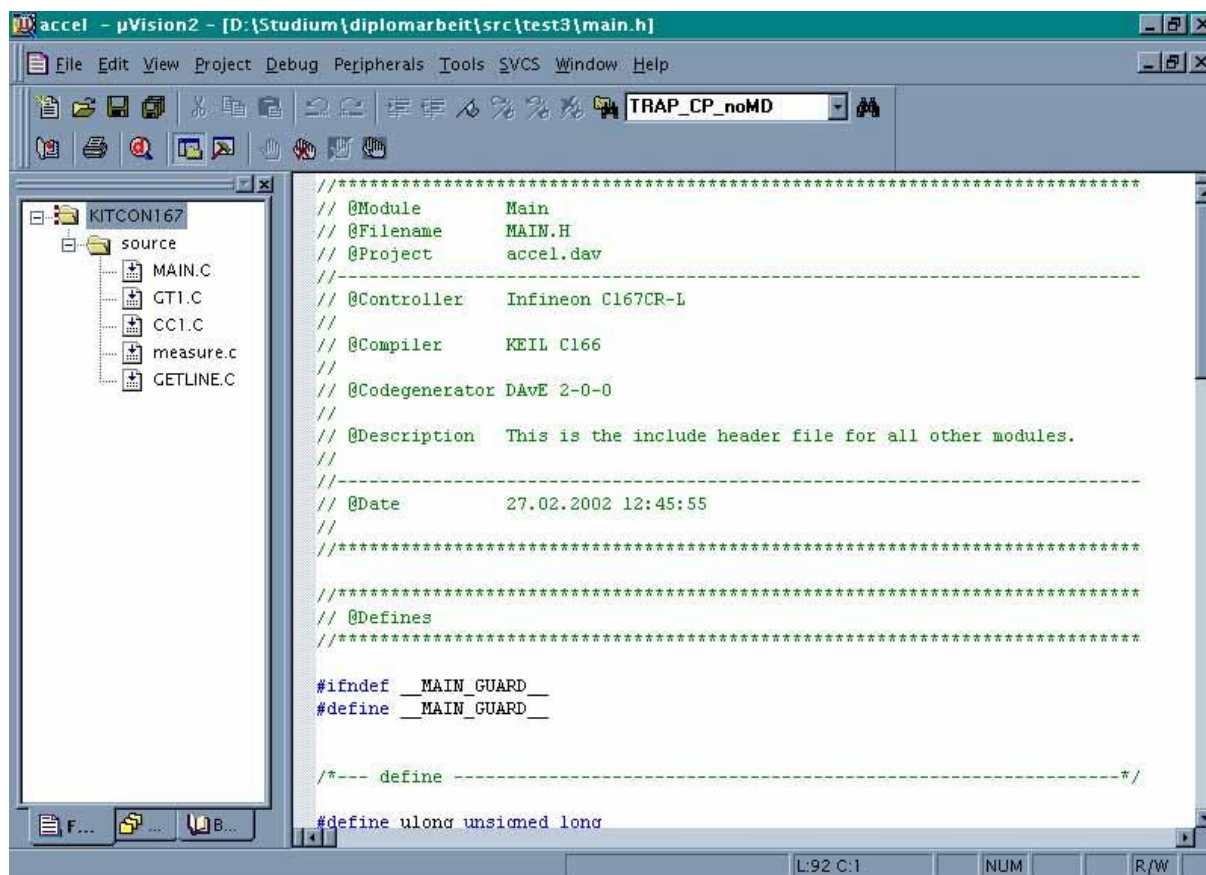


Fig 50

9.5 Realization of the ACU

In this chapter is the realization of the actuator control Unit (ACU) explains the five blocks of the ACU (see Figure 14) is as follows: First, it presented the block diagram, for the blocks "Voltage regulation", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe " is the respective circuit presented, and the individual parts of the circuit described in detail in blocks " Betriebsspannungs-Überwachung " and " Steuersignal-Erzeugung " is only described the Software on the microcontroller expires because the Board for the latter 3 blocks off-the-shelf was concerned, and only the programming of the microcontroller to an established part of the present work was.

9.5.1 Versorgungsspannungs-Stabilisierung

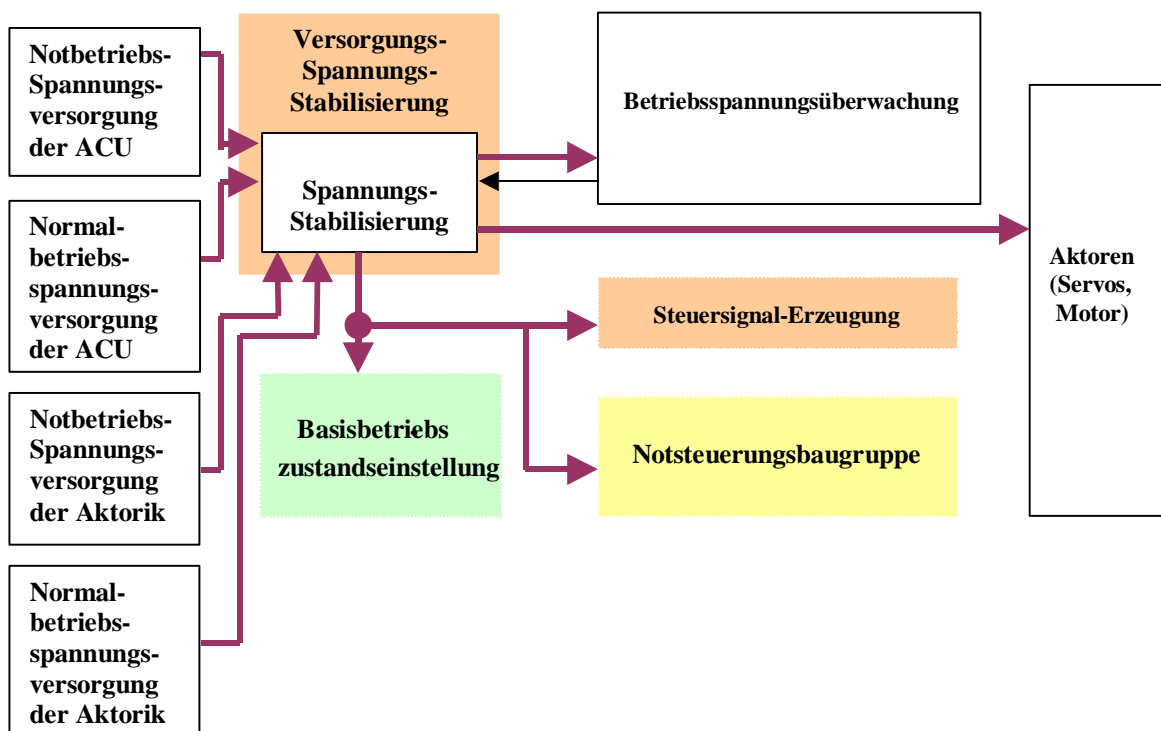


Fig 51

Figure 17 shows the assembly " Versorgungsspannungs-Stabilisierung ", which the remaining blocks of the ACU with a constant voltage of +5V supply, as well as provides this assembly to ensure that the power supply for a certain time is maintained, after the Normal-Betriebsspannungsversorgung not more is fully functional, and this is done in that the " Versorgungsspannungs-Stabilisierung " in this case, the remaining elements of the ACU to the Notbetriebsspannungsquelle connects.

9.5.1.1 Circuit Design for the Versorgungsspannungs-Stabilisierung

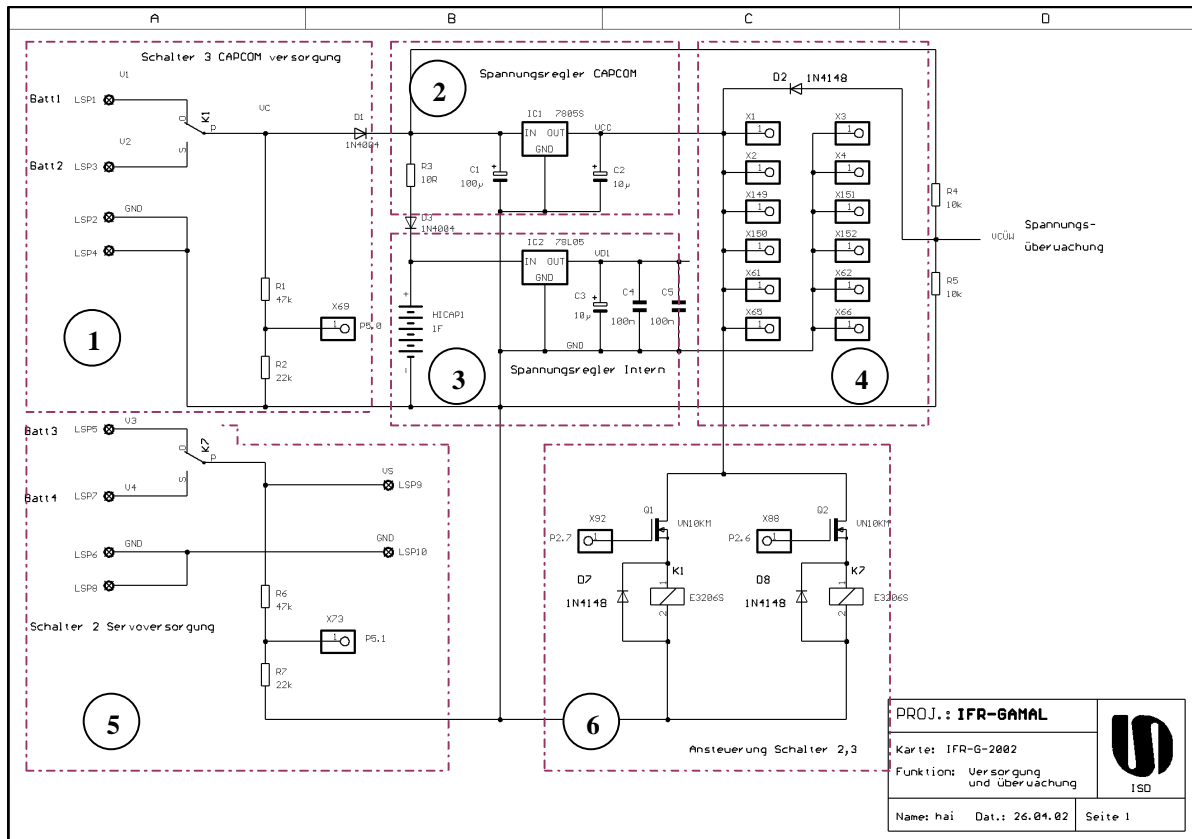


Fig 52

The circuit for the Versorgungsspannungs-Stabilisierung consists of six areas (see figure above, the ranging numbers are there circled):

1. Connection to the supply voltage for the ACU (Normal and Notbetriebsspannungsversorgung)
2. Stabilization of the supply voltage for the Blocks " Betriebsspannungs-Überwachung " and " Steuersignal-Erzeugung " (i.e. , for the microcontrollers, since these two blocks to the Microcontroller are implemented)
3. Stabilization of the supply voltage for the Blocks " Versorgungsspannungs-Stabilisierung ", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe " (i.e. , for those blocks, the hardware are implemented, and while on a common board)
4. Connection to the remaining blocks of the ACU (left series: Connections to the microcontroller, i.e. , to " Betriebsspannungs-Überwachung " and " Steuersignal-

Erzeugung "; rights series: Connections to the blocks "voltage stabilization", " Basisbetriebszustands-Einstellung " and " Notsteuerungs-Baugruppe "), to the stabilised voltage with this to supply.

5. Connection to the supply voltage for the actuators (Normal and Notbetriebsspannungsversorgung)
6. Toggle switch from normal operation to Notbetriebsspannungsversorgung

Circuit Description The Schaltungsbereiches for the supply of the Mikrokontrollers (Schaltungsbereiche 1-4, upper half of the pattern thus activating):

Relay K1 switches battery 1 (soldering tag LSP1 +Pol, and LSP2 -pol) and the battery 2 (soldering tag LSP3 +Pol, and LSP4 -pol) using Digitalport P2.7 to (' 1' = Battery 2 enabled). The charge level of the battery voltage V_c is by the voltage divider $R1 + R2$ on the analog port P5.0 measured ($V_{IN} \frac{1}{3} V_C$), with the voltage divider $R4 + R5$ can be an internal voltage activated, the supply of the Capcom is carried out by the +5V voltage regulator IC1 which are Umschaltspitzen with the capacitor C1 smoothed, the supply of the internal logic is made with the regulator IC2, the in addition to the highly-capacitive Pufferkondensator HICAP1 in total voltage drop is supplied to the servos in a defined location, see figure 18.

Circuit Description The Schaltungsbereiches for the supply of the actuators (servos and engine) (Schaltungsbereiche 5, left, the lower quarter of the pattern thus activating):

Relay K7 switches battery 3 (soldering tag LSP5 +Pol, and LSP6 -pol) and battery 4 (soldering tag LSP7 +Pol, and LSP8 -pol) using Digitalport P2.6 or higher (' 1' = Battery 4 enabled). The battery charge the battery voltage vs is by the voltage divider R6 + R7 at the analog port P5.1 measured (Vin Ca Aprx 1/3 Vs). The LSP9 (+pol) and LSP10 (-pol) is connected the supply of the governor.

9.5.2 Betriebsspannungsüberwachung

The "Betriebsspannungsüberwachung" monitors, whether and when the required operating voltage falls below a particular level, at the the blocks of the ACU not more properly can be supplied.

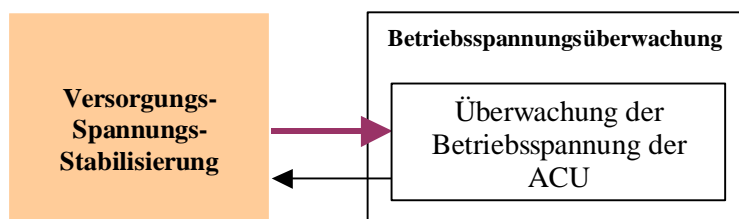


Fig 53 Block diagram for the Betriebsspannungüberwachung

The Betriebsspannungsüberwachung software is purely realized with a program, which expires on the microcontroller, the Microcontroller uses up to a A/D-converter, which is also on the PHYTEC-test board is located, to current, analog voltage level in a digital number to convert, then finally the with the help of a Mikrocontroller-Programms is compared with a threshold value, the digitized current voltage level is less than the threshold, a signal is generated on the output of the microcontroller, which is by the Block " Versorgungsspannungs-Stabilisierung " is directed, and there causes that on "Notversorgungsspannung" is switched.

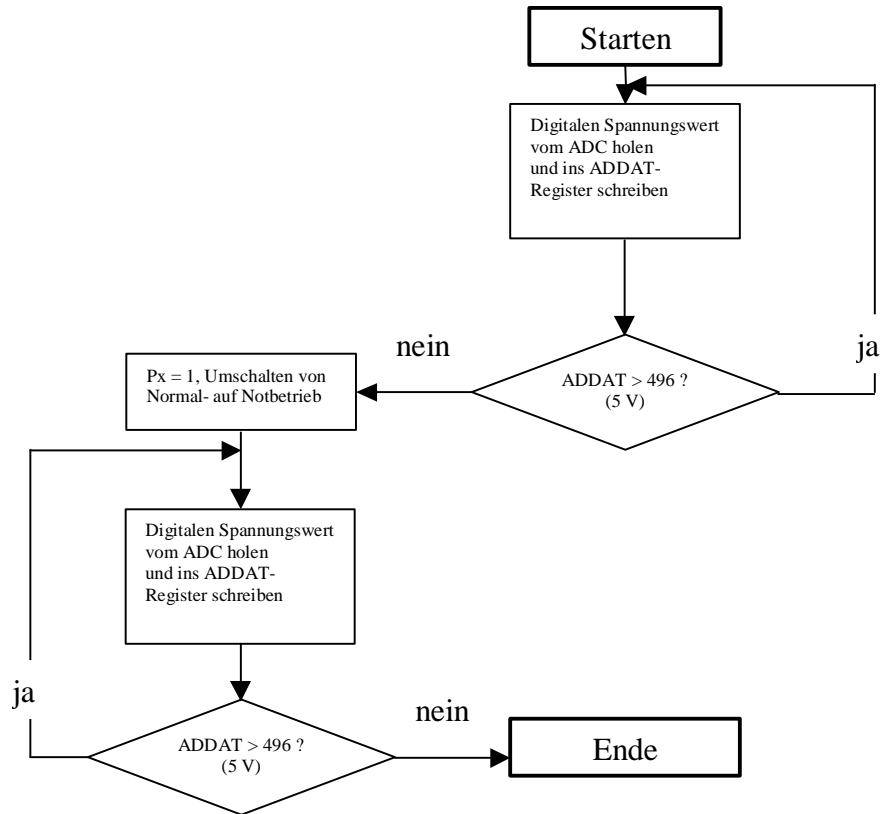


Fig 54 the μC

Figure 20 shows the μC -program for the Betriebsspannungsüberwachung. The same program is used for both the Betriebsspannungsüberwachung the Aktorik-Spannungsversorgung , as well as for the Betriebsspannungsüberwachung the ACU-power supply used. The two analog Meßeingänge for the two power supplies are available at two different channels of the ADCs and will be gradually converted by the ADC and the ADDAT in-register of the microcontrollers written (Single conversion mode of the ADC, see Ch. 2).

The microcontroller still works at a supply there must be a pause of. That is why the 4.76 V. Microcontroller are used to a voltage fell below below the 5V-border to detect and respond to it.

9.5.3 Basisbetriebszustands-Einstellung

Using a switch, the on a separate channel (channel 5) of the remote control is pressed, the distributor of the " Basisbetriebszustands-Einstellung " with either **normal operation** (contains normal operation and normal operation (A B) or with Luftschiiff-Startphasen -operation be initialized when Luftschiiff-Startphasen operation, the PWM signals from Fernsteuerungsempfänger switched directly to the actuation system (to override of the microcontroller of the block " Steuersignal-Erzeugung "), during the flight can Fernsteuerungskanal 5 between Luftschiiff-Startphasen -operation and normal operation be switched in both directions, if neither control signals (PWM signals) at the output of the block "Steuersignalerzeugung", nor on the channel5-output of the Fernsteuerempfangers concerns, the finite state machine of the " Basisbetriebszustands-Einstellung " the system, in the

Condition " Notsteuersignal-Erzeugung - level 2" (i.e. , hardware production of Notsteuersignalen) move to (state transition Notsteuersignal-Erzeugung normal operation after - level 2).

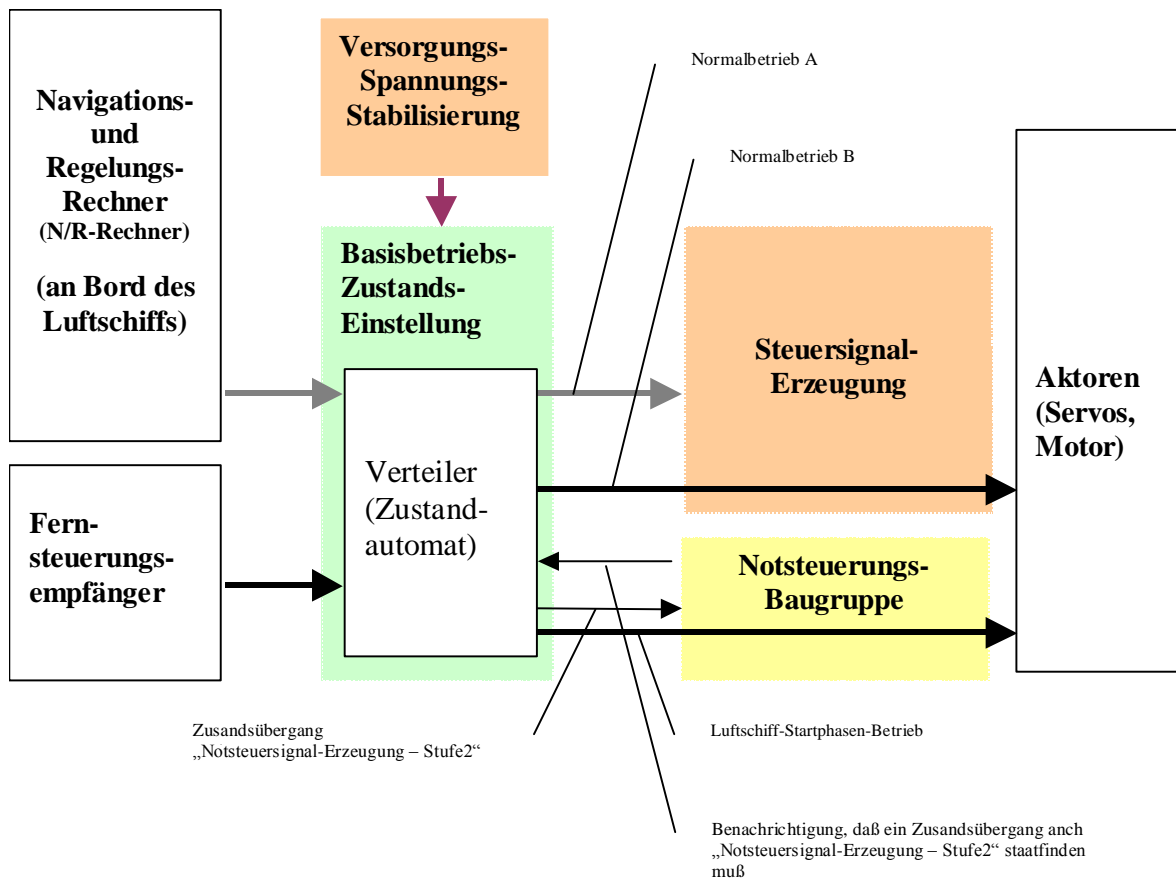


Fig 55

Figure 22 shows the state diagram for the distribution of the " Basisbetriebszustands-Einstellung ":

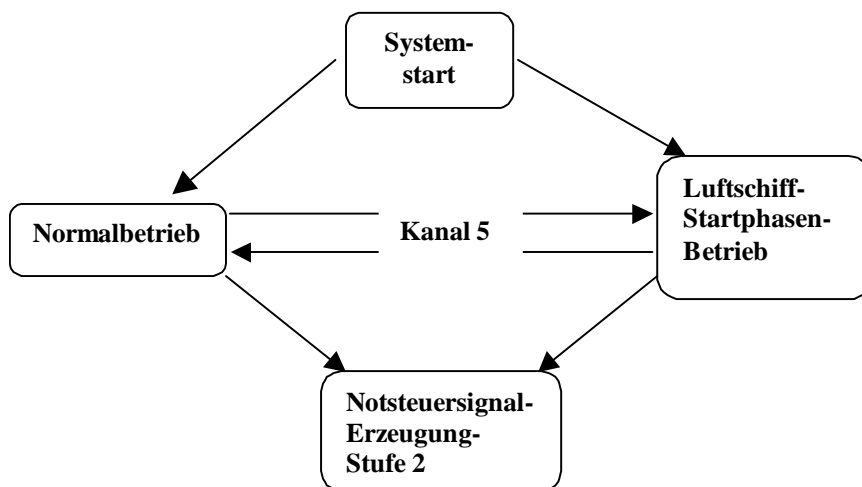


Figure 56

9.5.3.1 Circuit Design for the Basisbetriebszustands-Einstellung

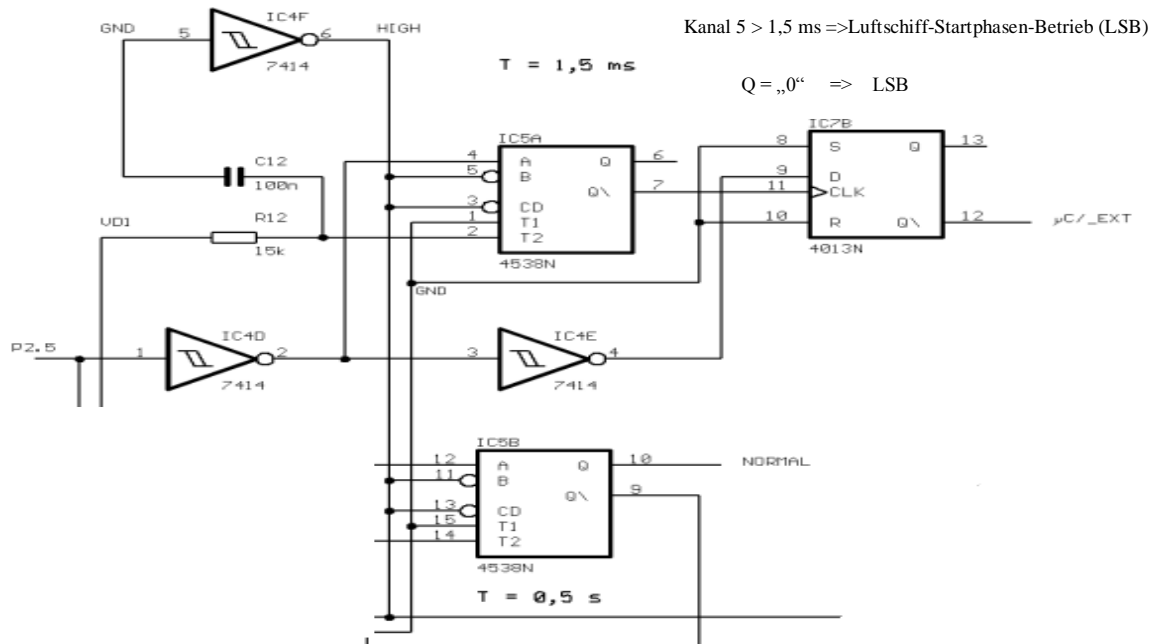


Figure 23: Part 1 of the switch between Luftschiff-Startphasen -operation and normal operation (LSB/NB-switch, part1)57

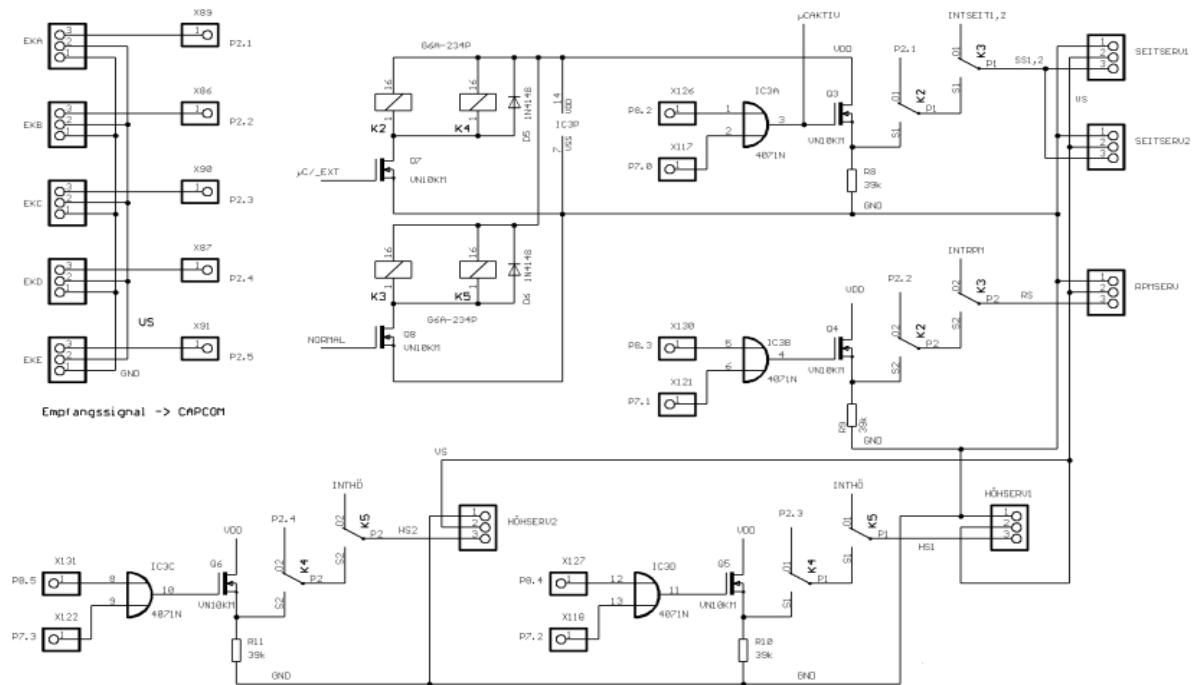


Fig 24: Part 2 of the switch between Luftschiff-Startphasen -operation and normal operation (LSB/NB-switch, part2)58

Transition from startup, after LSB or NB and between LSB and NB (see Figure 23):

The PWM-signal of Fernsteuerungskanal 5 is from the output P2.5 of the Fernsteuerungsempfangers on the entrance to the LSB/NB-switch (in Figure 24 with P2.5 refers). In the LSB/NB-switch is depending on the length of the PWM-signal greater than 1.5 ms: LSB; smaller than 1.5 ms: NB) of the output 12 (C/_Ext) of his flip-flop module IC7B on high or low (see Figure 23) is set, this line of the flip-flops, with the gate input is insulated (C/_Ext) of the Fett-Kondensators Q7 (see 24) connected. The FET controls the relay K2 and K4, and K2 can be either a interconnection of the Fernsteuerkanale 1 and 2 (P2.1 and P2.2) to the rudder and the engine or but the connection of the rudder and the engine to the outputs P8.2, P7.0, P8.3 and P7.1 of the microcontroller, K4 has the same effect as K2, but not for the rudder and the engine, but for the two Höhenrunderhalften (elevators 1 And 2).μμ

Transition from NB or LSB Notsteuersignal-Erzeugung - level 2 (see Figure 23):

If neither PWM signals at the output of the ACU concerns, nor PWM signals over Fernsteuerungskanal 5, interrupt the relay K3 and K5 the connections to the relay K2 and K4 and connect the actuators with the Block " Notsteuersignal-Erzeugung - level 2 ".

Table 7 summarizes the inputs and outputs of the LSB/NB-switch together:

Fernsteuerungs-Kanal	ACU-Eingang	ACU-Ausgang	Einstell-Potentiom.	Funktion (Servo)
1	P2.1	P8.2/P7.0	P17	Seitenrunder 1+2
2	P2.2	P8.3/P7.1	P16	Motor
3	P2.3	P8.4/P7.2	P15	Höhenrunder 1
4	P2.4	P8.5/P7.3	P14	Höhenrunder 2
5	P2.5			Umschaltung ext./μC

9.5.4 Steuersignal-Erzeugung

In Abschn.3.3 was the architecture of the " Steuersignal-Erzeugung raised before the description of the off-the-job training Shelf-Hardware and the design of the software components of the " Steuersignal-Erzeugung " is moved, here is to again on the architecture of the " Steuersignal-Erzeugung " be received, and also some of the things added:

The Block "Steuersignalerzeugung" generated depending on the operating status on various way control signals (PWM signals), the directly the actuators (engine, servos) response:

- In **normal operation A** gets the "Steuersignalerzeugung" commands from manifold of the " Basisbetriebszustands-Einstellung " and generates control signals (i.e. , PWM signals)
- In **normal operation B** are control signals (i.e. , PWM signals) from Fernsteuerungsempfanger (by the microcontroller through) activated and forwarded to the actuators, the Microcontroller is used here as Signal-Begrenzer and observers.

- At the system startup to the normal operation B is **initialized, if on channel 5 "Normal mode" is set, and the further operation may, at any time from normal operation B switch to normal operation a** be and vice versa, and this is done by setting from the N/R-machine from the N/R-computer receives this information on his **communication system** from the ground (not from the remote control).
- If in normal operation A or normal operation B due to the failure or failure of the N/R-computer and/or of the Fernsteuerungsempfangers no more signals the microcontroller on the input port of the microcontroller, will be moved to the State " Notsteuersignal- Erzeugung - level 1 ", in which the microcontroller of the "Steuersignalerzeugung" autonomously PWM signals to the actuators generated and there is, so can the time of the failure of normal operation and normal operation B with a defined PWM signals are bridged.

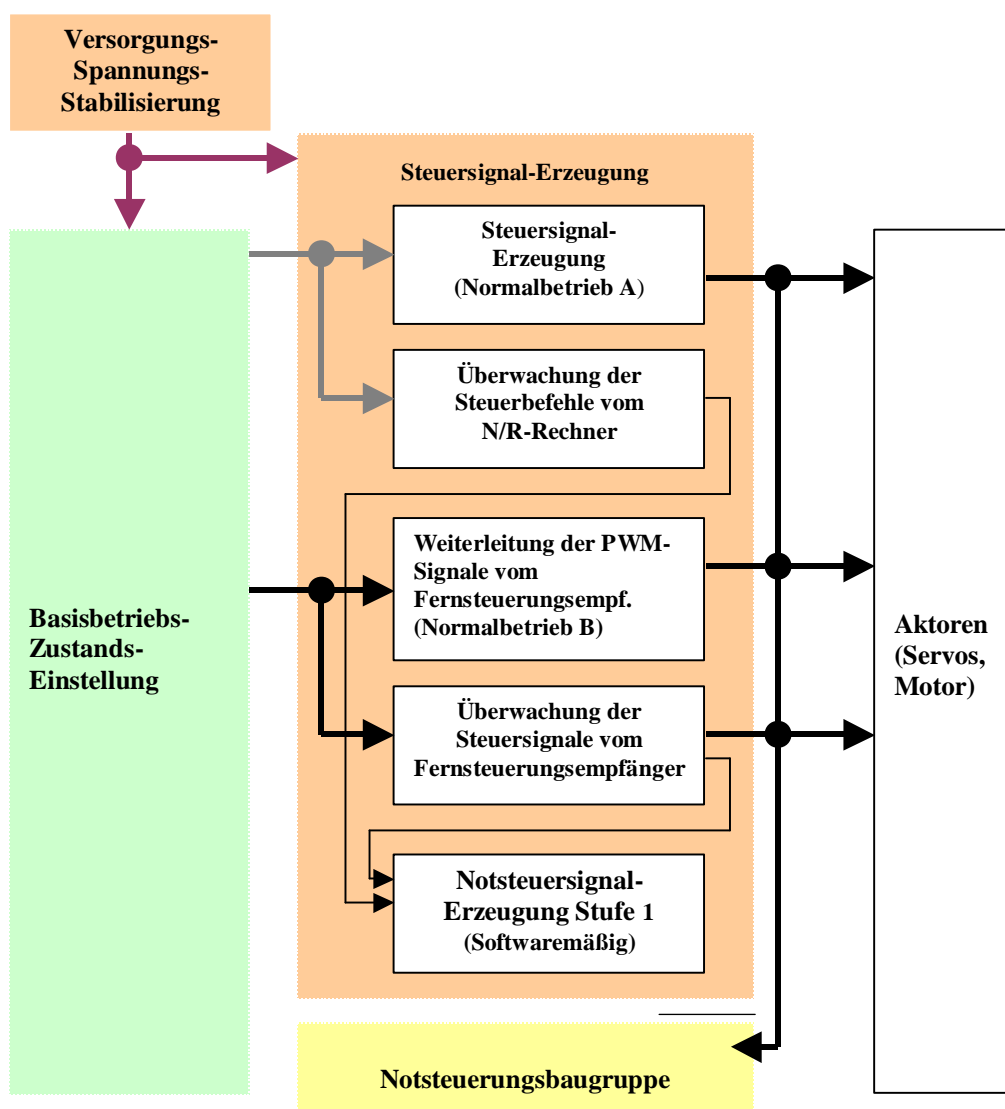


Fig 25: Block diagram of the "Steuersignal-Erzeugung" 59

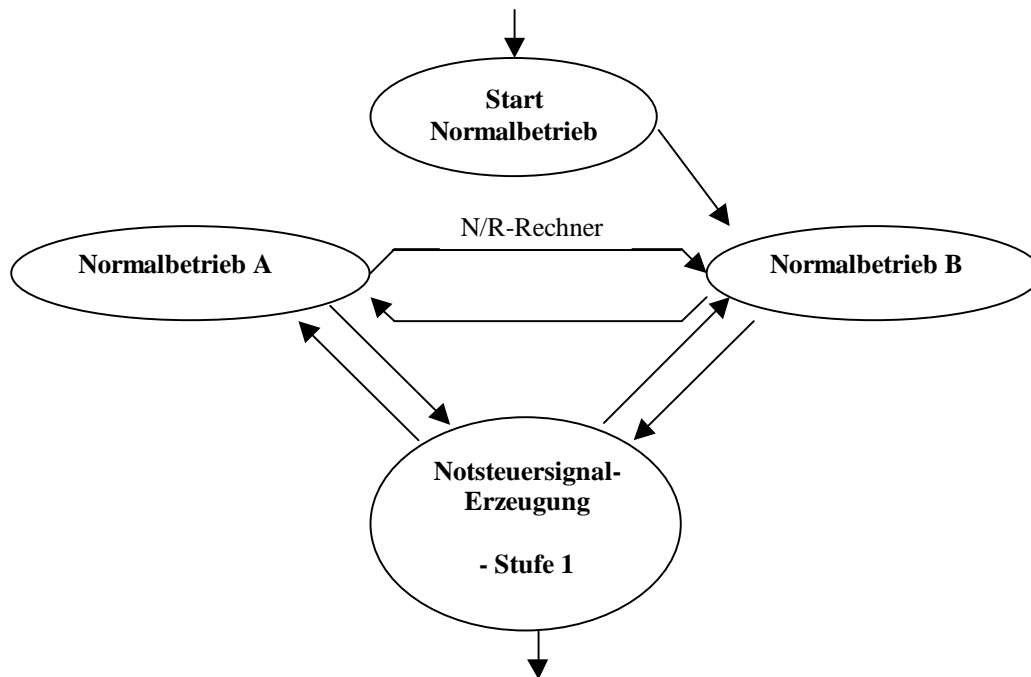


Fig 60

9.5.4.1 The hardware of the Steuersignal-Erzeugung

The Steuersignal-Erzeugung is realized by a software on the microcontroller 80C167CR-LM is running, and the microcontroller is located on a Testbord, the company was purchased PHYTEC. All the required peripherals are also located on the plate.

9.5.4.2 The monitoring and the forwarding of the PWM-signals from the remote control (normal operation (B))

The monitoring of the pulse width is particularly important for the proper function of the circuit, because of these data have the dynamic and the stationary behavior of the airship (alternative Lotte) abhanget, faults can also indirectly in the Aktorenbereich pulse produced by the be discovered.

The 4 channels of the receiver are connected to the lower 4 pins of the Port 2 connected. In Figure 27 is the program for the 4 inputs of the μ c. The pulse lengths are programd with the help of the Capcom1 unit measured

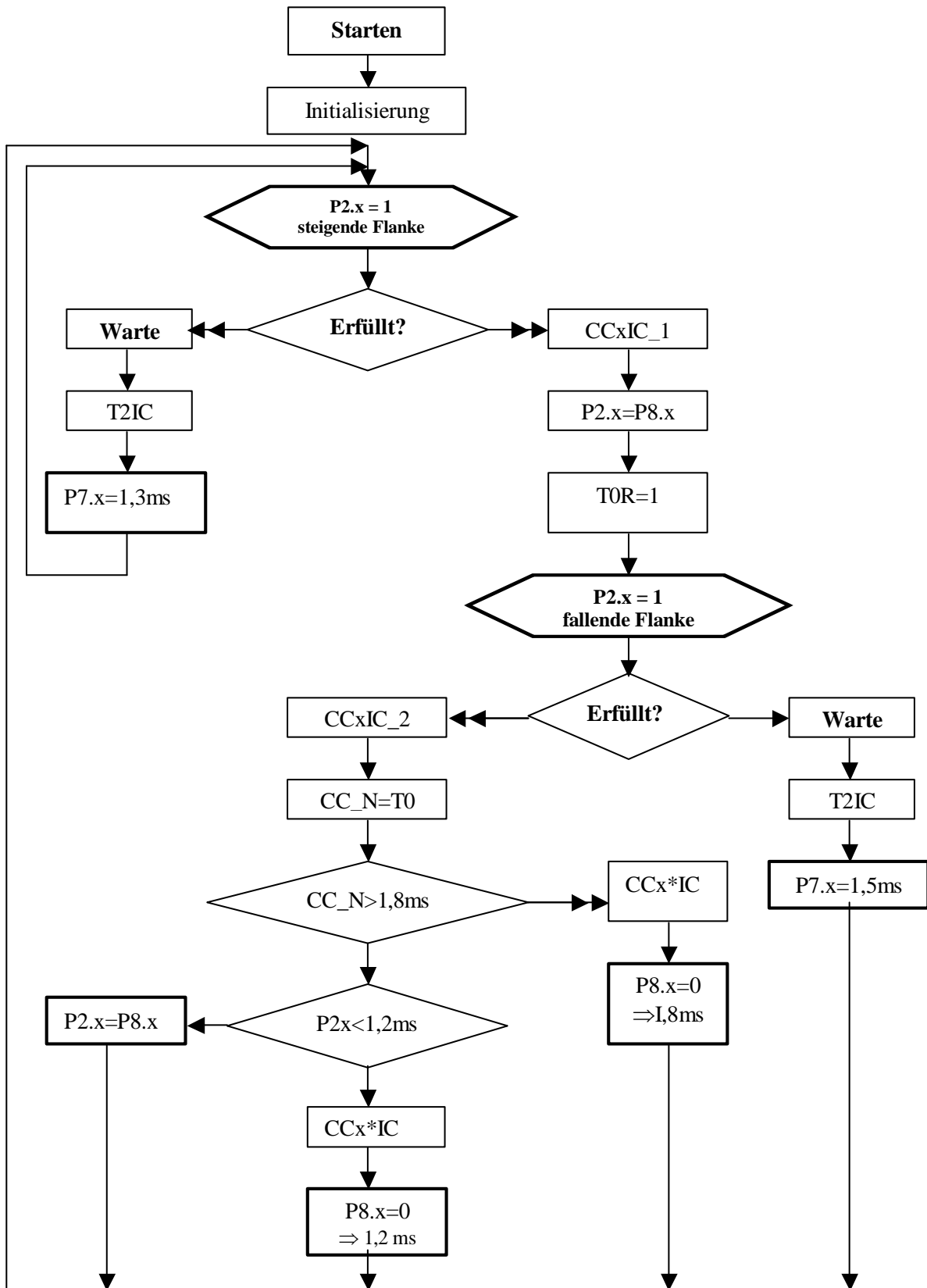


Fig 61

KanalNr.	Capture-Rg.	Mode-Register	Anschluß	Timer	Interrupt
1	CC1	CCM0 .0 - .4	CC1IO P2.1	T0	CC1IC
2	CC2	CCM0 .4 - .7	CC2IO P2.2	T0	CC2IC
3	CC3	CCM0 .8 - .11	CC3IO P2.3	T0	CC3IC
4	CC4	CCM0 .12 - .15	CC4IO P2.4	T0	CC4IC

Table 8

The subunit CAPCOM1 consists, as previously mentioned in chapter 2 was, from the two timers T0 and T1, the program was only one timer necessary and 4 registers (see Table 4) .

Be the 4 tabs in the Mode Capture (used to recover) is used, which means that the register at each rising or falling edge on these inputs corresponding the the recovered CCxx meter readings of the Timer T0 Record. Your Operating Mode (Mode) in a 4-bit-long field in a of the Betriebsartregisters CCM0.

The control register CCMx set the operating mode of each of four Captur/ compare channels each firmly.

Selection of the operating mode CCMODx

001 : Capture on the rising edge at the terminal CCxIO

010 : Capture the trailing edge at Port CCxIO

011 : Capture on both flanks at Port CCxIO

ACCx assignment the timer: 0 = T0 or "T7 ", 1 = T1 or T8

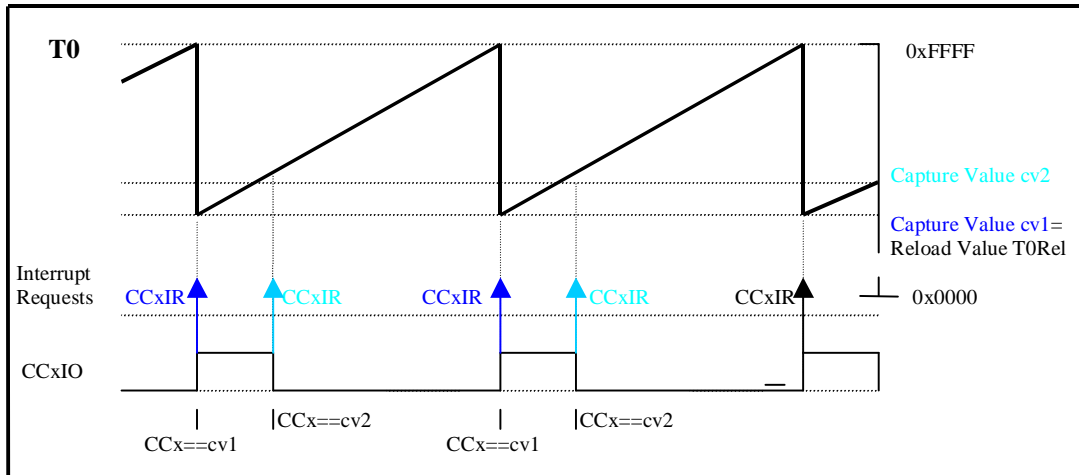
The CCM0 register is in the main function under void main (void) with CCM0 = 0x1111 set, and so is with each rising edge on these inputs a corresponding interrupt (CCxIC) is triggered, and for processing led to the first interrupt routine.

First interrupt routine:

In the **first interrupt routine** is the Timer T0 and its 3,000 Nachladeregistern the value given to the same period of 25 ms to reach the measured signals, then with TOR = 1 started, before the control register should T01CON = 0x0000 in the main function will be programd, the CCM0-Register is here with 0x2222 programd to the falling edges to capture.

And so with the falling edge on the inputs a corresponding interrupt (CCxIC) triggered and the current counter of the timer T0 to the corresponding Capture-Register CCxx saved (see Figure 28). The triggered Interrupt leads to the processing of the second interrupt routine.

Fig 62



Second Interruptroutinen:

In the **second interrupt routine** , the value of the Timer T0 in the moment in the Capture-Register saved and then to the global variable copied CC_Nx. The CCM0-Register will be programd to rising edge, and the pulse length is measured not only easy, but it must also be made in a statement, and this is why the CC_Nx values with two constant T1x_MIN and T1x_MAX compared (see program and Figure 29). The two constants are used to limit the pulse signals between 1.2 ms and 1.8 ms, and that makes it possible to get the Hohenruderausschlag limited by the software.

```

If (CC_N > T1_min )
{
    CC18 = T1_min;
    P82 = !P21;
}
Else if (CC_N < T1_max)
{
    CC18 = T1_max;
}
Else
}
P82 = P21;

```

}

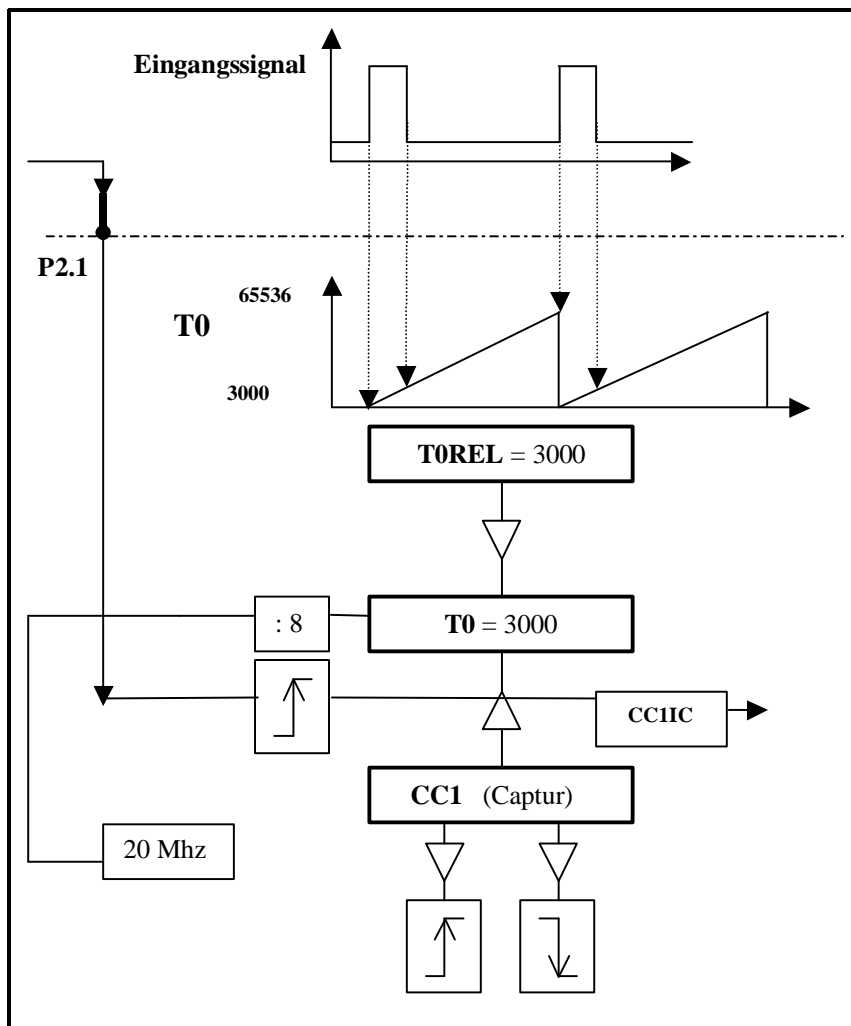


Fig 63

Third interrupt routine:

In this interrupt routine, the contents Compare-Register , of the same value of 1.8 ms is, with the Timer T1 within second interrupt routine compared. In conformity is a Interrupt CCX * IC is triggered and this will of the appropriate output port (P8X= 0) set to zero, thus the PWM-limited pulses.

At 1.2 ms, with the smaller T1_min the Compare-Registers/1.8 ms and when is with the larger value of T1_max the Compare-Registers compared.

```
CC18_isr void (void) interrupt 0x32
```

```
{
```

```

P82 = 0;
CC21 = T1_max;
}

```

Fourth interrupt routine:

For the monitoring of the input signals is the **fourth interrupt routine** with the Timer T2 initialized. The initialization of the interrupts is done after the setting of the mode of the timer T2CON = 0x0003 (period = 210 ms) and with T2R = 1, the timer is started and with a 0 the interrupt processing given freely, and the timer T2 is in the second Interruptroutinen every time on the T2 = 500 reset, which means that the third interrupt routine is not triggered, only if no rising or falling edge within the period of the timer is important, in the T2 interrupt routine, constant

PWM signals generated by the PWM unit.

Test routines

For the tests, which are described in chapter 2, test programs have been created.

Test 1 is used

For the measurement and evaluation of the Mikrocontroller-Eingangssignale (PWM signals) on board, the come from the recipient and

To test the of the PWM-unit of the microcontroller produced PWM signals.

Test 2 will test the constancy of the supply voltage of the μC , and the other components of the Aktorik-Ansteuerungsbaugruppe .

The full code is included in **Annex B** listed.

9.5.4.3 Level 1 Notsteuersignal-Erzeugung

With this unit in the event of an emergency are constant PWM signals generated (30), if the recipient for some reason no sending them signals to the outputs, this unit has 4 channels, each of the 4 channels contains its own Timer PTX, the from the CPU-measure or a divider on the counts receives(Table 4). The period of the signal results from a comparison of the current counter with the Periodenregister PX, the Low-Zeit from a comparison with the Pulsweitenregister PWX (see C program and 30).

Kanal	Timer	Periode PP _x	Weite PW _x	Port-Ausgang
0	PT0	PP0	PW0	P7.0
1	PT1	PP1	PW1	P7.1
2	PT2	PP2	PW2	P7.2
3	PT3	PP3	PW3	P7.3

Table 5: shows the DCLS Einheit-Ausgänge and their corresponding Timer

```
T2_isr void (void) interrupt 0x22
```

```
{
```

```
  P7X = 0; // P7.x is output
```

```
  PX = 7800; // Max Period
```

```
  PWX = 7372) and strong support ; // Initial value (high flank)
```

```
  PWMCON0 = 0x0044; // Timer and CPU-to-measure : Age 64 (210
```

ms)

```
  PWMCON1 = 0x0004; // outputs with mode 0 Free
```

```
}
```

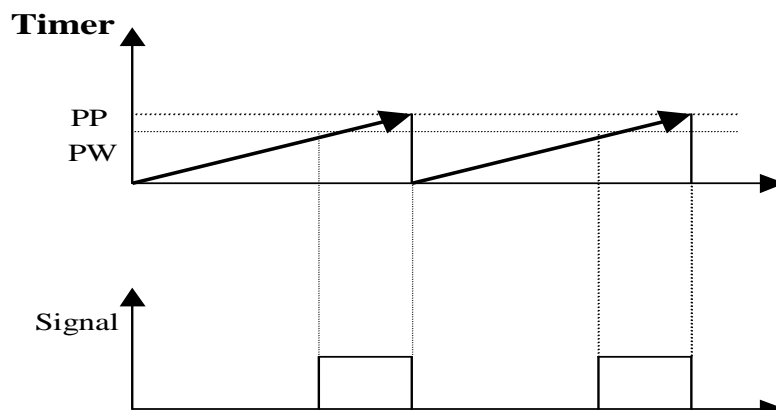


Fig 64

9.5.5 Notsteuerungsbaugruppe

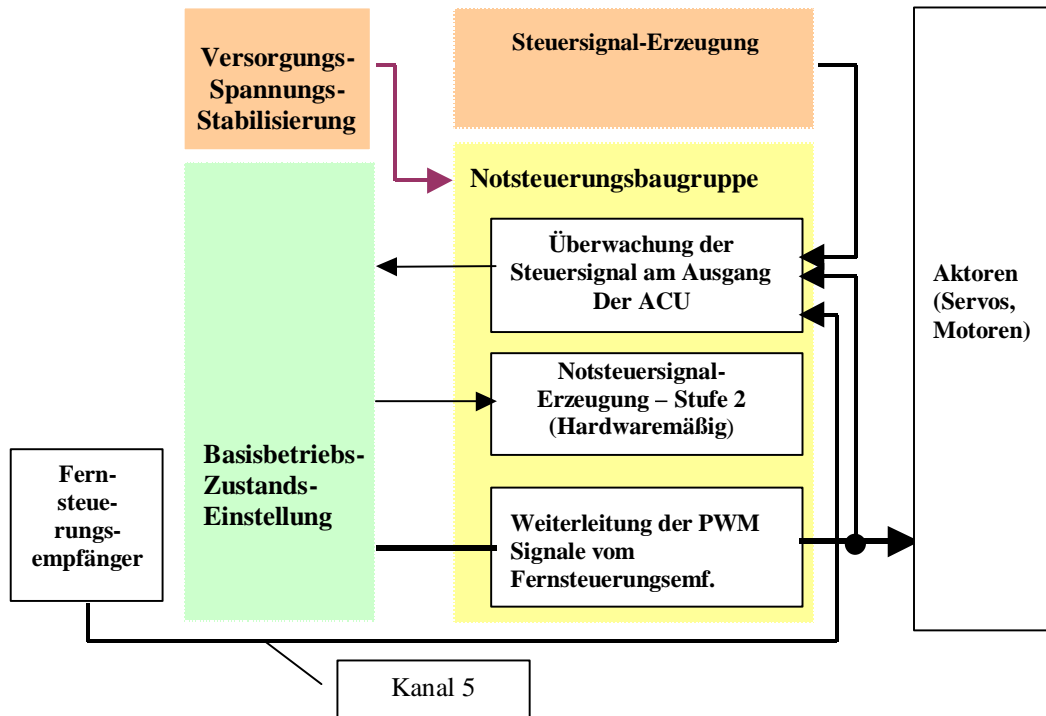


Fig 65

9.5.5.1 Circuit Design for the monitoring of the control signals on the output of the ACU and the channel5-output of the Fernsteuerempfängers

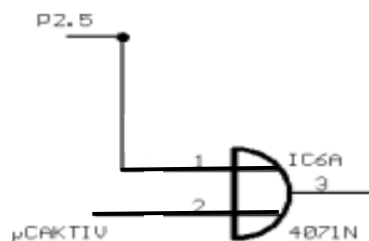


Fig 66 μ

If to P2.5) and μ CAKTIV no signals are present, the OR-link to output 3 set to "Low" and flip-flops to input 12 of the IC5B of the " Basisbetriebszustands-Einstellung " (see Figure 23). The output 9 of the IC5B is to input 11 of IC9B (see Figure 29) has been set and now led the production of Notsteuerungssignalen - level 2.

9.5.5.2 Circuit Design for the Notsteuerungsbaugruppe

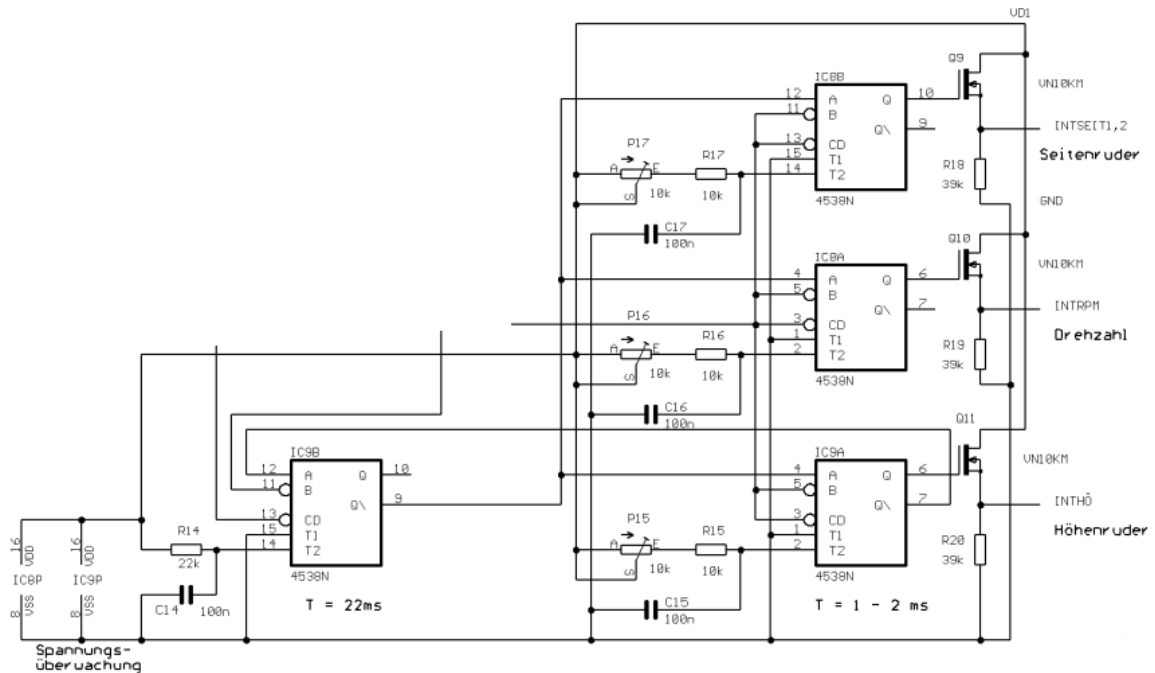


Fig 67

As already mentioned above, output 9 of the IC5B of the " Basisbetriebszustands-Einstellung " (see Figure 23) with input 11 of IC9B (see Figure 33) now connected and led the production of Notsteuerungssignalen - level 2, and this is done as follows:

Output 9 of the IC9B triggers each of the input A of the three flip-flops IC8B, IC8A and IC9A. The output of the latter three flip-flops is each with the gate of a FET, the gate signal of the reinforced forwards to the respective actuator, PWM signals are caused by the combination with the clock of the two flip-flops expands (IC9B and IC8B, IC9B and IC8A, IC9B and IC9A).

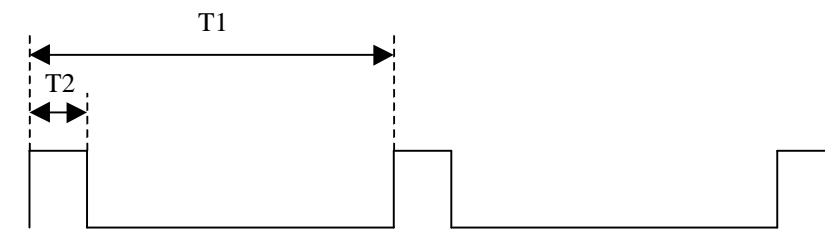


Figure 68 One of the three generated PWM signals, which is generated by the combination of the clocks of the two flip-flops expands is created (see text)

The PWM of the three generated PWM signals (in the picture as an example T2-T1 for one of the three generated PWM-signals) can before each use of the airship by the potentiometer P15 P16 and P17 be set (Figure 34). This can the Notsteuersignal-Erzeugung - level 2 of the conditions of the respective mission be adapted.

9.5.5.3 Circuit Design for the forwarding of the PWM-signals from Fernsteuerungsempfänger (Luftschiff-Startphasen -operation)

The condition "Luftschiff-Startphasen -operation" are the channels 1-4 (in Figure 24 are you with P2.1-P2.4 and higher) of the Fernsteuerempfänger placed directly to the actuators, and this is done, if K2, K3, K4 and K5 are activated accordingly (see left and right lower half of the 24), i.e. , P2.1 is directly with SEITSERV1 and SEITSERV2 connected, P2.2 is directly with RPMSEV (engine) connected, P2.3 is connected directly with HÖHSEV1 and P2.4 and higher HÖHSEV2 is directly connected with.

i.e. here is not a new circuit necessary.

9.5.6 Layout design of board 1 ("Voltage regulation", "Basisbetriebszustands-Einstellung" and "Notsteuerungs-Baugruppe")

The blocks "Voltage regulation", "Basisbetriebszustands-Einstellung" and "Notsteuerungs-Baugruppe" are realized on the circuit board 1 (see section 3.6), Figure 35 shows the layout design of Board 1.

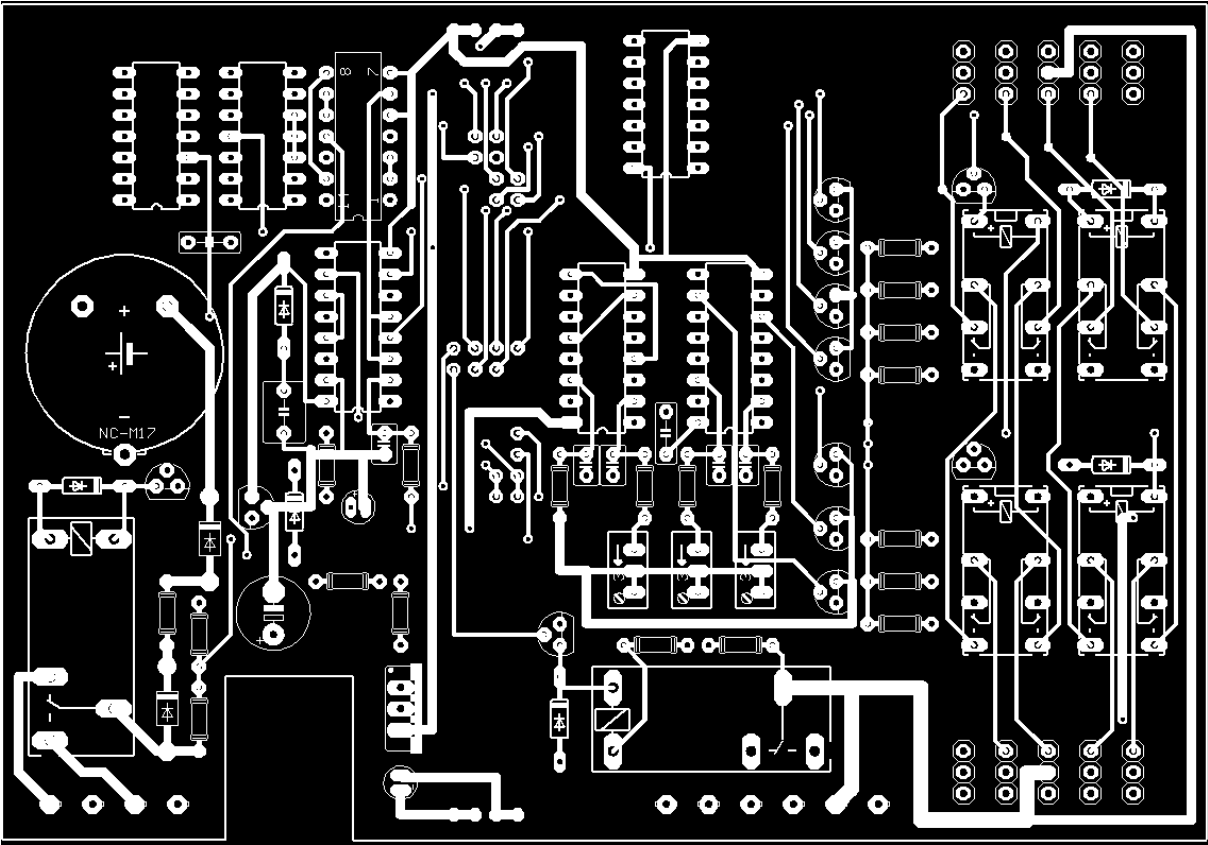


Fig 69

9.5.7 The circuit boards of the ACU

The both circuit boards of the ACU

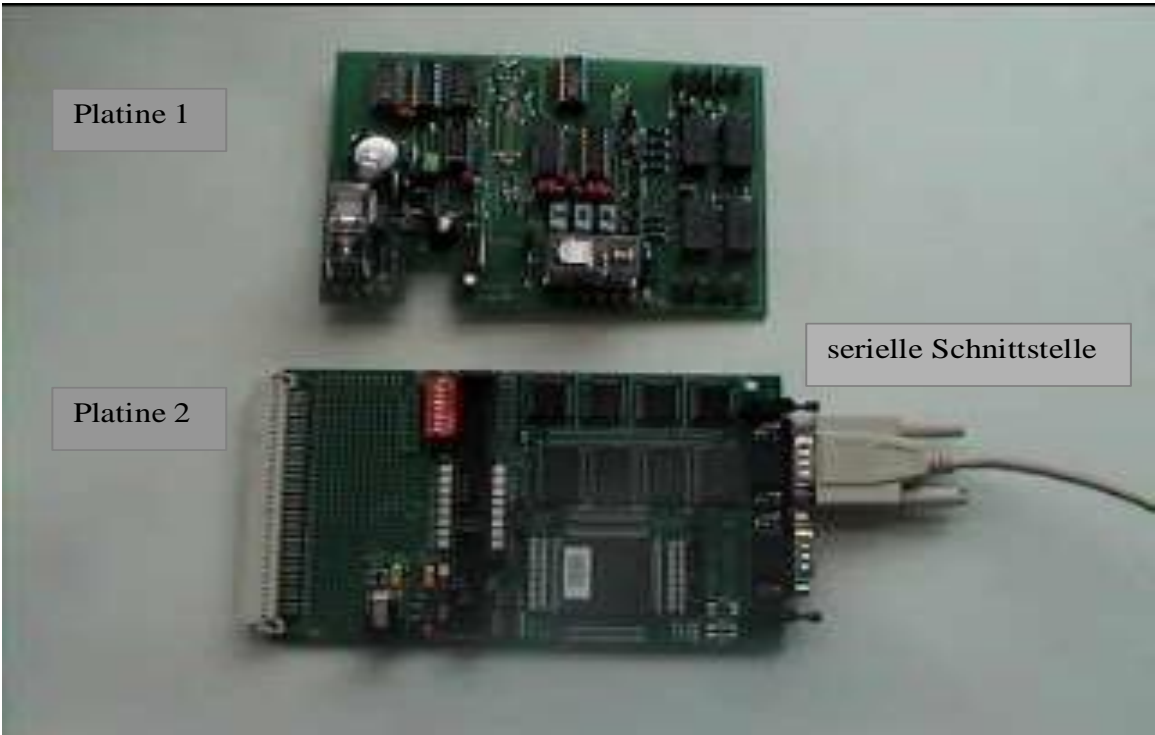


Fig 36: the two boards (1 and 2) to the ACU

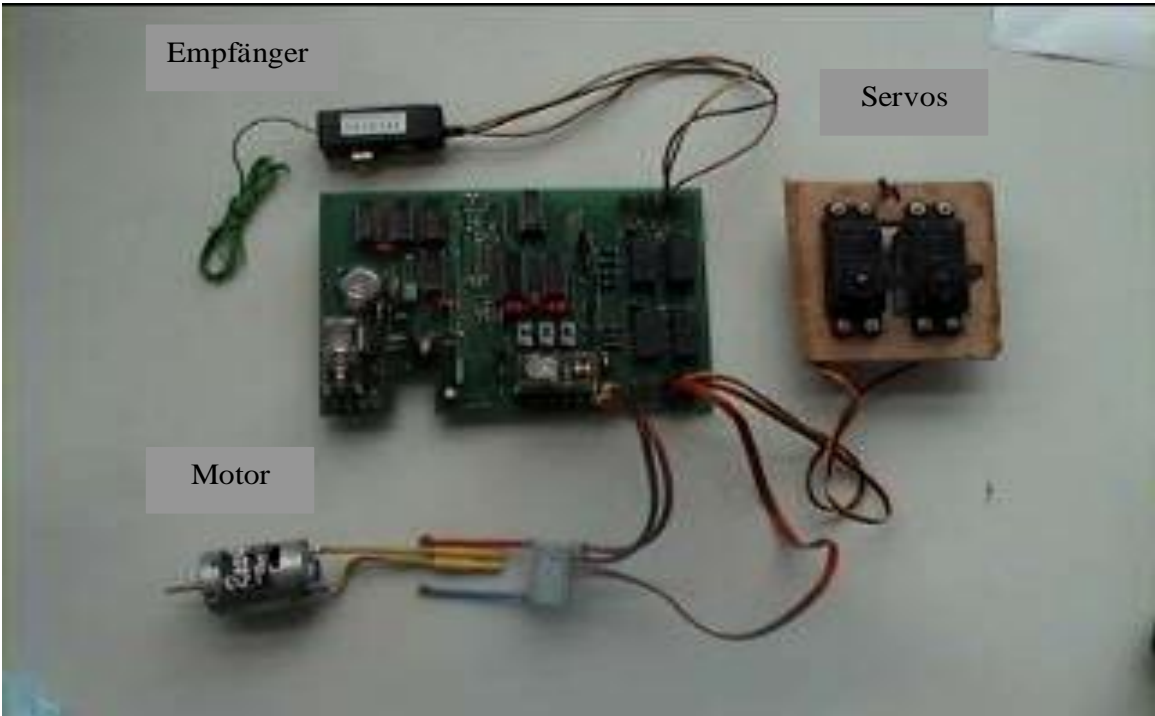


Figure 37: the receiver and the actuators (engine,servos) are connected to circuit board 1.

9.6 Experimental results

9.6.1 Structure of the Testplatzes

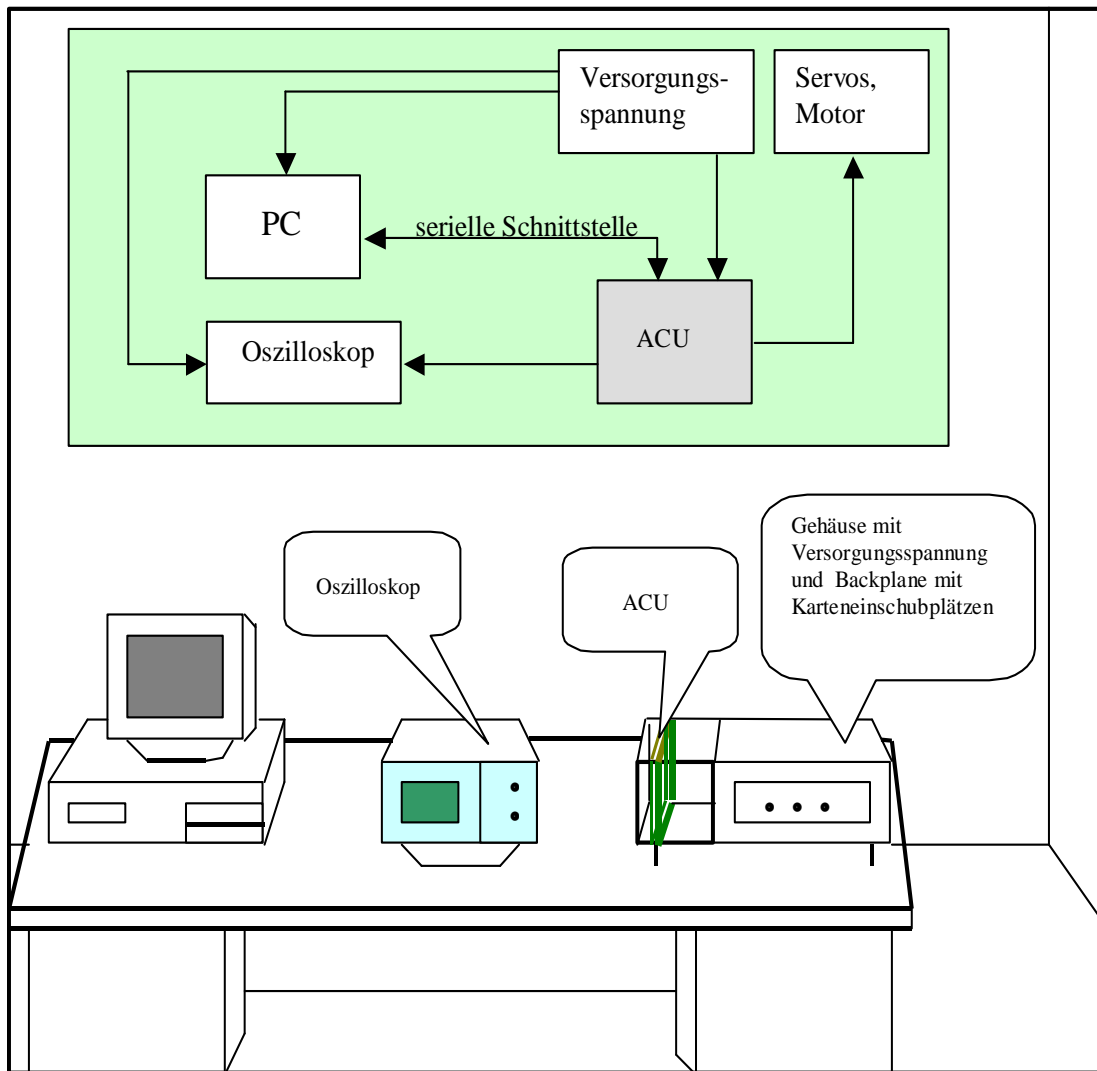


Figure 38: Testplatzaufbau

Fig 38 shows the in the experiments and the development of the program used for the ACU devices. The PC is the ACU via a serial interface (RS232) connected to the PWM signals to make visible, an oscilloscope is used.

The ACU is particularly sensitive to static electricity, which is why it is important, this assembly to a suitable place to be during the experiments, in the housing for the embedded system is a Versorgungsspannungserzeugungseinheit (right-hand side of the figure) is installed, in order to able to forego on batteries.

9.6.2 Experiments and test sequence

The output values (counter values the Compare-Einheit/swap is performed on the ACU-microcontroller for the rising and falling edges of visualized on the oscilloscope signals) over a serial interface sent to the PC, it is calculated and the pulse width on the PC-screen made visible, the three following signals are made visible at the same time on the oscilloscope.

- The ACU-input signal to channel 5,
- A selectable output signal of the ACU and
- A selectable input signal of the ACU.

On the PC-screen pulse width can be permanently on the oscilloscope with the pulse width of the respective signal shown by the user are compared, so you can while the program runs directly determine whether the program expires without fault, this data are of great help in the detection of the error with unstable program.

The two most important experimental data are visualized on the oscilloscope and the length of the period of the distance between the rising and the falling edge of the the PWM-signal.

9.6.3 The series of tests

At the time of the implementation of the tests was the ACU Mikrokontroller-Platine so faulty that input only P2.4 and higher (channel 4), exit 4 (control of elevator clattering 1) and channel 5 did, so no signals from other A and/or outputs (control of the rudder, of the engine, ...) will be tested, but are on the other a and/or outputs to the expected signals similar to those of each input 4 and Output 4.

In the following tests the signals to Input 4 and Output 4 in various operating conditions as measured. The current operational status is also measured by the signal set to channel 5, the 2 possible modes of channel 5 (1, by the recipient pulse width less than 1.5 ms: normal operation; 2, pulse width from the receiver greater than 1.5 ms: Luftschiff-Startphasen - operation) are by a switch on Fernsteuerungssender (on the ground) set by hand.

The following are the different conditions under which the different tests ran:

Test 1: pulse width of channel 5 of the receiver less than 1.5 ms: -> normal. The ACU is under the condition when you turn on "normal" to the normal mode B initialized. Under this condition runs test 1.

Test 2: pulse width of channel 5 of the recipient greater than 1.5 ms: Luftschiff-Startphasen - operation.

Test 3: there is no signal from the receiver: -> It is a signal to output generated by the microcontroller 4 (Notsteuersignal-Erzeugung - level 1)

Test 4: ACU Mikrokontroller-Platine is removed from the system: -> Notsteuersignal-Erzeugung - level 2 is activated, a signal generated by the Notsteuerungsbaugruppe is to output 4 to.

Test 5-1, Test 5-2: operation B, i.e. the microcontroller directs the PWM-signal from the remote control next (input 4 on the outcome 4)and, in addition, the pulse width of the signal

limited, i.e. , the signal on output 4 (channel 4) is in contrast to the signal at the input (up or down to) Limited, unlike the limit test 1 is indeed in the visible, because the input signals outside the limits.

Test 6: between normal operation and normal operation A B is changed by the corresponding information from the N/R-machine to the microcontroller of the ACU is given, the N/R-computer receives this information via the communication system from the ground, in test 6 this will be simulated by two different keys on the lab computer, the serial interface is connected to the ACU, and of the the N/R-simulated by computer, will be activated, if the A button is pressed, the system is intended to go to the normal mode A, in the other key is to control the system to normal operation B go. The correct transfer of information from N/R-machine to the microcontroller and the correct response is with the help of appropriate LEDs on the Mikrocontroller-Platine (board 2) simulates. These lights depending on the currently valid condition.

In the Test-Abbildungen are from top to bottom to see the following signals:

1. CH 1: channel 5 at the entrance of the ACU
2. CH 2: channel 4 at the entrance of the ACU
3. CH 3: Channel 4 on the output of the μC
4. CH 4: Channel 4 on the output of the ACU

9.6.3.1 Test 1

The channel 5- switch of the remote control on the ground, is to "pulse width less than 1.5 ms" provided.

It is the normal case B (control signals from Fernsteuerrungs-Empf . will be give to the microcontroller, the passes on this to its output) tested, with the Fernsteuerungsknuppel is a certain deflection of the commanded Hohenruders. This specification is on channel 4 sent.

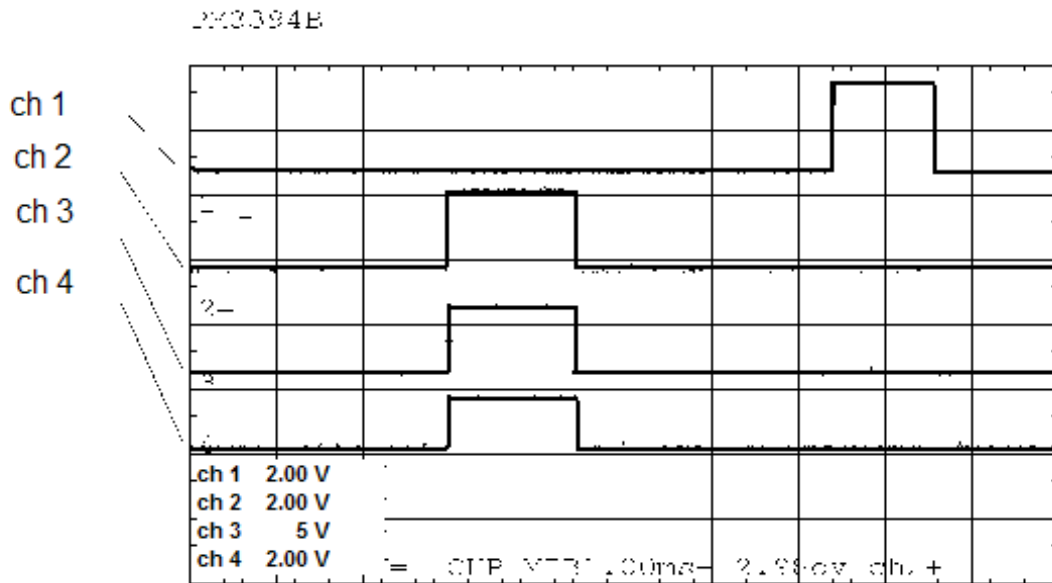


Fig 39: Test 1 CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU

Figure 39 shows that the pulse width of the channel 5 less than 1.5 ms (1.2 ms) is, this is normal operation B is set, and the PWM of the PWM signals at the input and output of the channel 4 are identical (1.5 ms), and so has the μ C the PWM signals without modification of the output of the ACU redirected. The results came out as expected.

9.6.3.2 Test 2

The channel 5- switch of the remote control on the ground, is to "pulse width greater than 1.5 ms" position, this will the Luftschiiff-Startphasen -operation (the PWM signals are Fernsteuerungsempfänger directly to the ACU-outputs passed on) set, otherwise runs Test 2 as the above test 1.

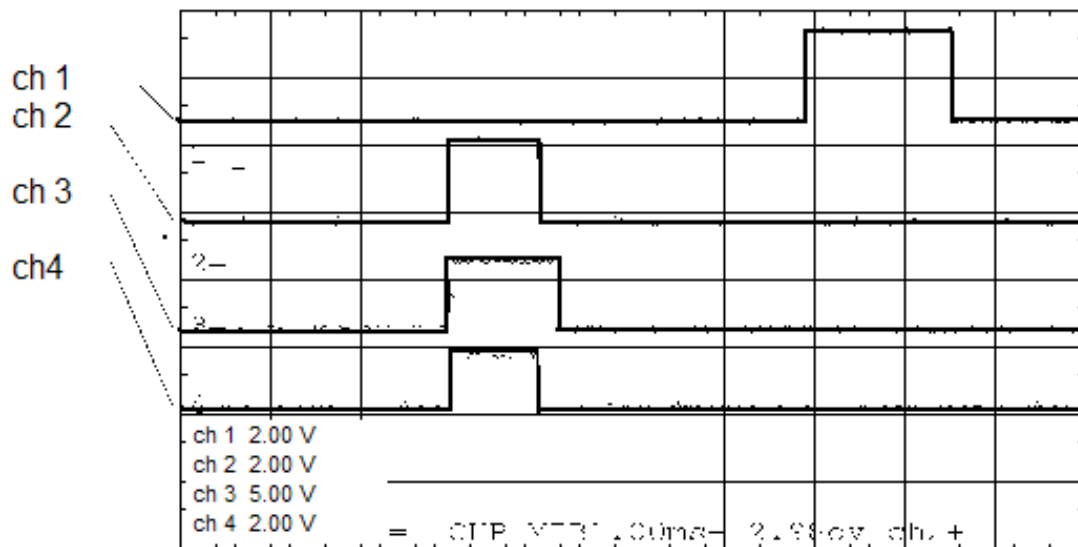


Fig 40: Test 2 CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU

In Figure 40 is the PWM-signal to channel 5 (CH 1) 2 ms long, and as expected, this Luftschiff-Startphase -operation set by the μ C, the PWM on the lower limit of 1.2 ms limited, and because the input signal (CH2) only 1.0 ms long and thus smaller than 1.2 ms, it is to 1.2 ms at the μ C output extended (see Ch 3). The Luftschiff-Startphasen -operation however the signals directly forwarded and unlimited on the ACU-output (CH4) is set, and the results in the figure are consistent with what you would expect.

9.6.3.3 TEST3.

The PWM signals from Fernsteuerungsempfänger (channel 1, channel 5) will not be more to the ACU-inputs connected (this is CH2 dead, i.e. , in the Fig is not a rash to see), thus a failure of the remote control will be simulated, and no data coming from the R/N-machine.

Thus, the level 1 Notsteuersignal-Erzeugung enabled.

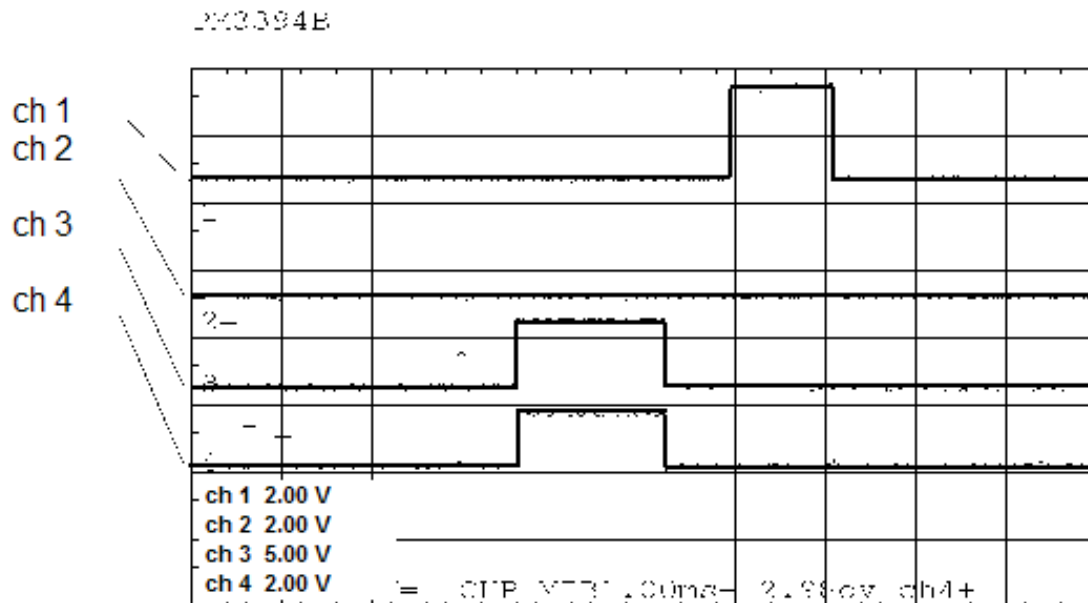


Figure 41: Test 3 CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU

As expected, the μC with the help of his internal PWM unit PWM signals with fixed pulsewidth created (CH3), the on the output of the ACU (Ch 4) concern.

9.6.3.4 Test 4

In this test, the "Notsteuersignal-Erzeugung - level 2" is activated, by the circuit board with the microcontroller (Plate 2) from the ACU is mechanically separated, this will simulate a failure of the microcontroller, in Fig. 42 one sees that neither an input signal (CH2) is present, even at the output of the microcontroller (CH3) a signal is present, even is the channel5-input signal from Fernsteuerungsempfänger (CH1) to.

The successful production of a PWM signal with the "Notsteuersignal-Erzeugung Level 2" is shown by ch 4.

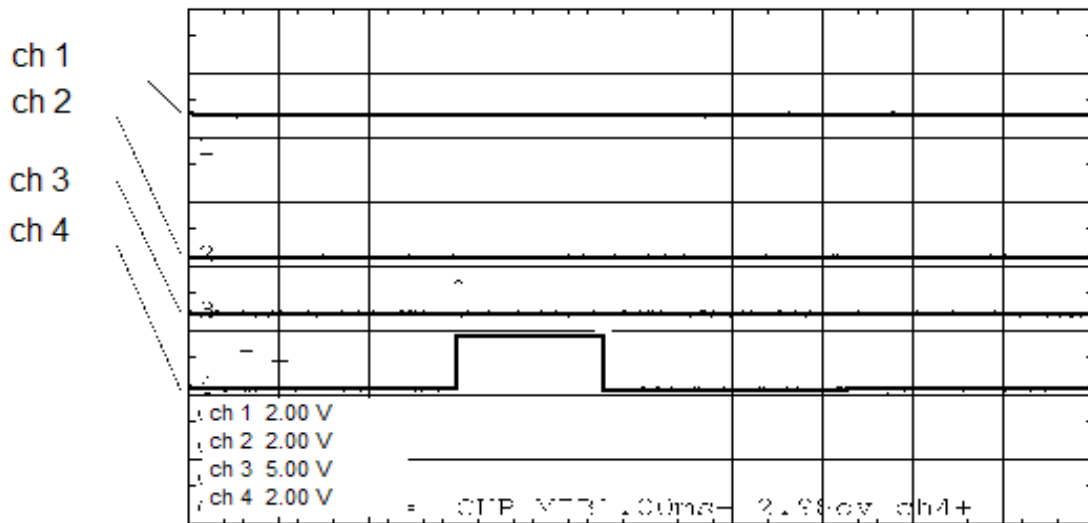


Fig 42: Test 4, CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU

9.6.3.5 Test 5

The Pulsweiten-Begrenzung the PWM signals in normal operation B at the output of the channel 4 tested.

By the Fernsteuerungsknuppel, the maximum pulse (2.0 ms) (Test 5-1) and minimum pulse width (1.0 ms) (Test 5-2.) The PWM-signal commanded. This is on ch 2 in the two illustrations 43 and 44 to see.

In Figure 43 is the limitation of the PWM-signal to the upper limit of the pulse width clearly shown: CH 2 (2.0 ms) is the unlimited signal, CH 4 (1.8 ms) is the limited signal.

In Figure 44 is the limitation of the PWM-signal to the lower border of the pulse width clearly shown: CH 2 (1.0 ms) is the unlimited signal, CH 4 (1.2 ms) is the limited signal.

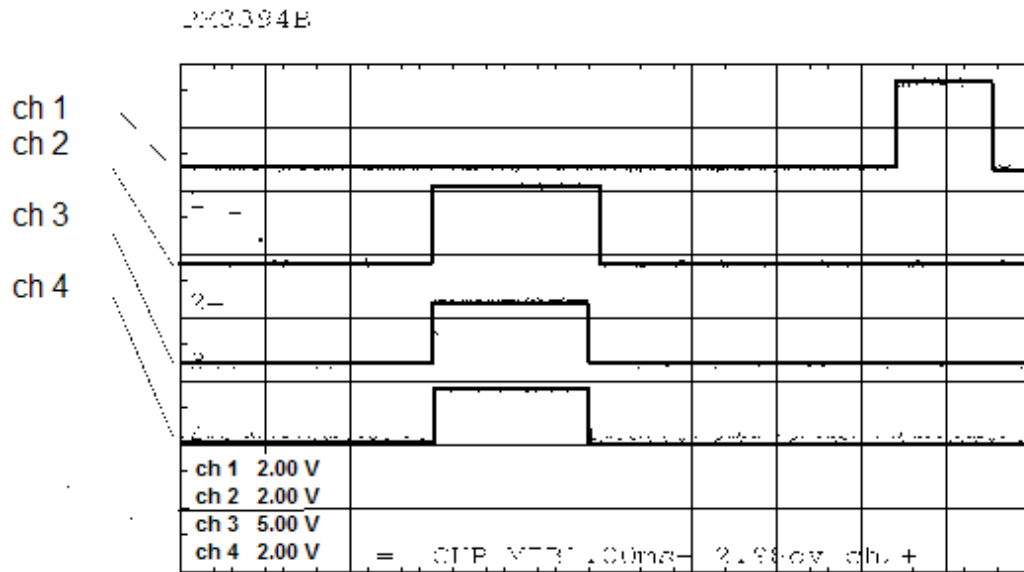


Figure 43: Test 5-1, CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU

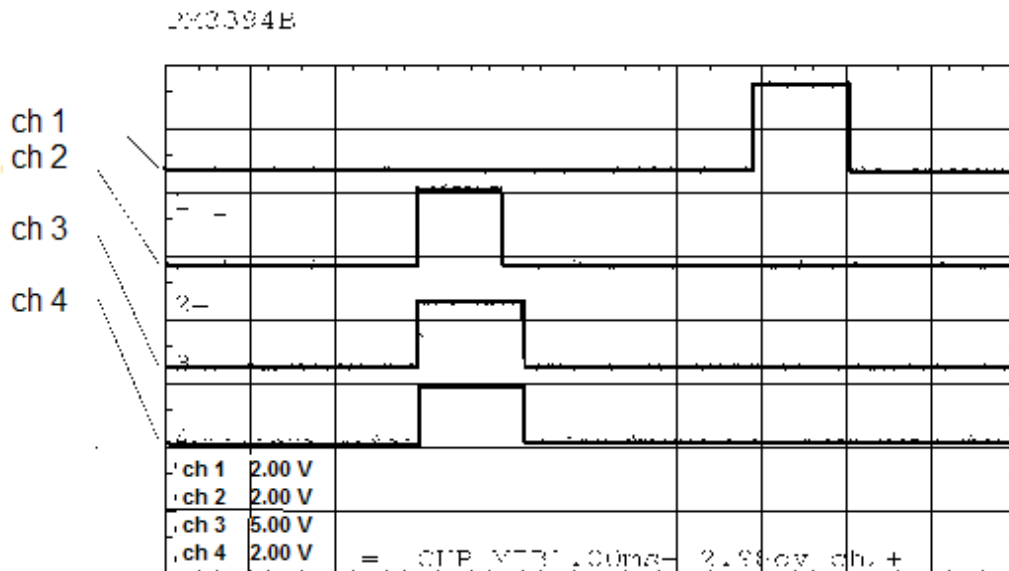


Figure 44: Test 5-2, CH1 = channel 5; CH2 = input signal from channel 4; CH3 = output from the microcontroller; CH4 = output channel 4 of the ACU

9.7 Summary and outlook

In this work, a part-functioning Aktorik-Ansteuerungs-unit (Actuator Control Unit (ACU) for the airship "alternative Lotte" was completed, and the ACU Fernsteuerungsempfänger get input signals of a or of a regulatory and the navigation computer (R/N-computer) on board, depending on the operating mode are in different ways by the ACU Antsteuerungssignale for the actuators (Helm, engine) is created, the ACU has a multi-level security system to R/N-the airship crash or Kommunikationsverbindungsausfall defined for a time as possible to control and, where necessary to bring safely to the ground.

At the time of submission of the thesis is the data stream between R/N-machine and ACU has not yet been tested on correctness, the there is a connection, but is has been verified and the data stream between R/N-machine and ACU is to be tested during the integration phase, i.e. when the ACU together with the other components of the Flugkontrollsystem (Flight Control System (FCS) are joined together.

Literature

- [1] (C167) underlying hedged transaction Users Manual, Infineon AG, Munich, 3.1 edition; March 2000.
- [2] Schmitt G, "micro-computers with the Controller C167 ", Oldenbourg Verlag, 2000.
- [3] Martin H. , "lecture notes for microcontroller C167 ", Institute of Computing Technology, Technical University of Vienna, 2000.⁵
- [4] Bernd B. / Peter G. , "The Big C167-microcontroller Operation", Gerber Verlag, 2001

⁵ The script is available for free on the Internet under <http://mc.ict.tuwien.ac.at>

9.8 Appendix A: Circuit Design for the Stabilization of Power Supply Voltage

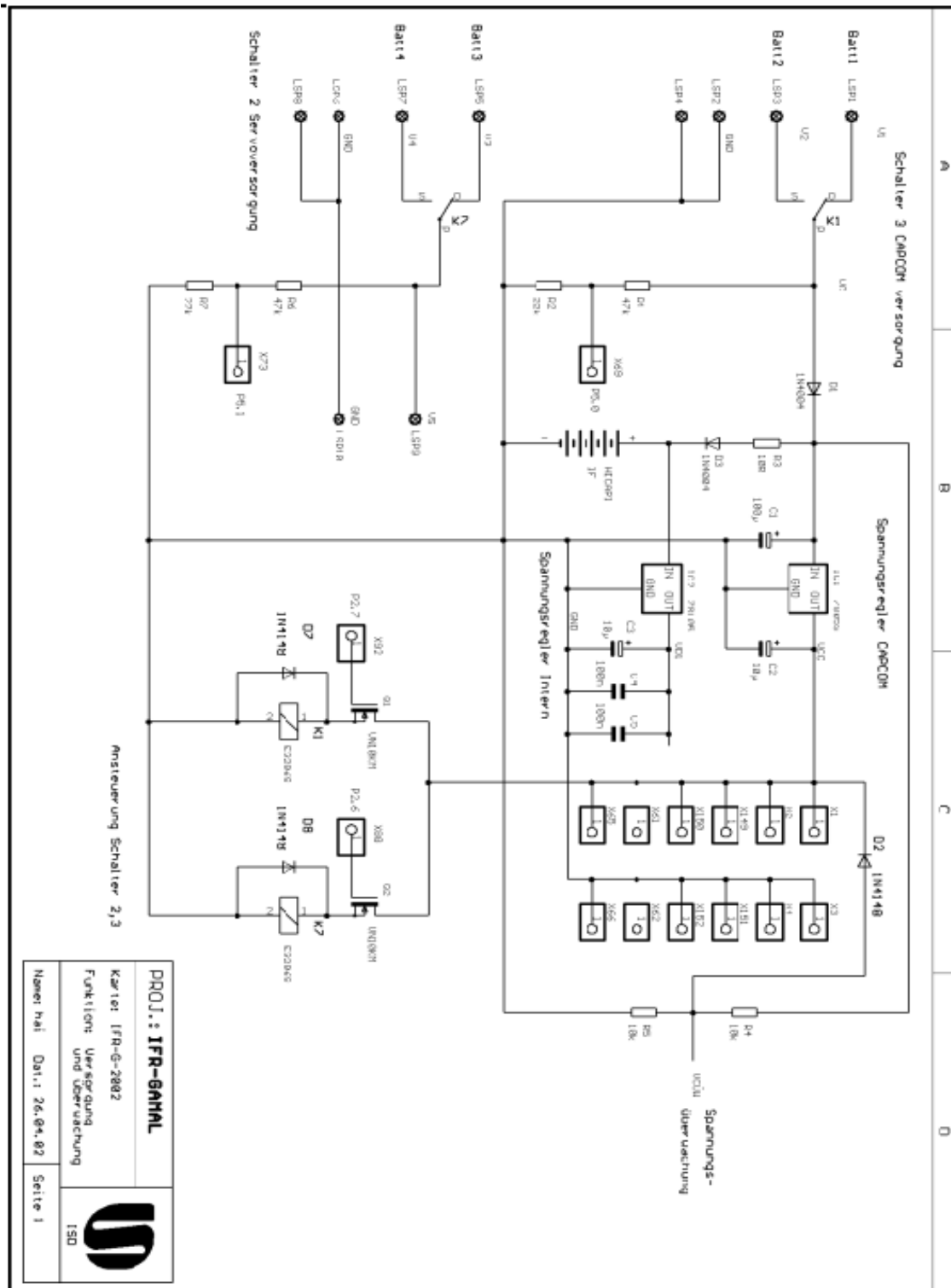


Figure 0-1: circuit design for the Stabilization of Power Supply Voltage (Versorgungsspannungs-Stabilisierung)

Circuit for switch between Luftschiff-Startphasen -operation and normal operation (LSB/NB switch, part2)

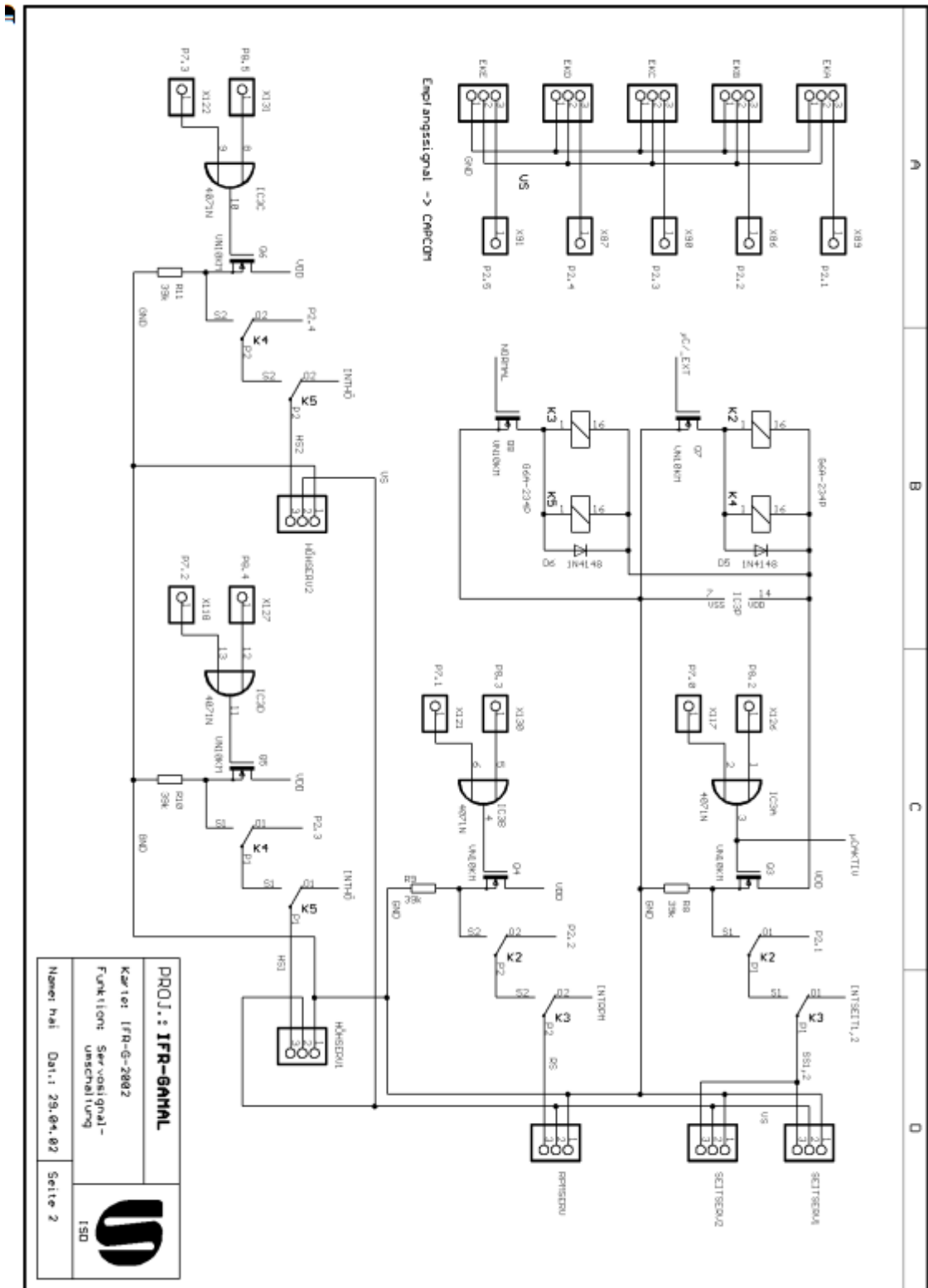


Figure 0-2: shift to switch between Luftschiff-Startphasen -operation and normal operation (LSB/NB switch, part2)

Circuit Design for the Basisbetriebszustands-Einstellung and the Basisbetriebszustands-Einstellung

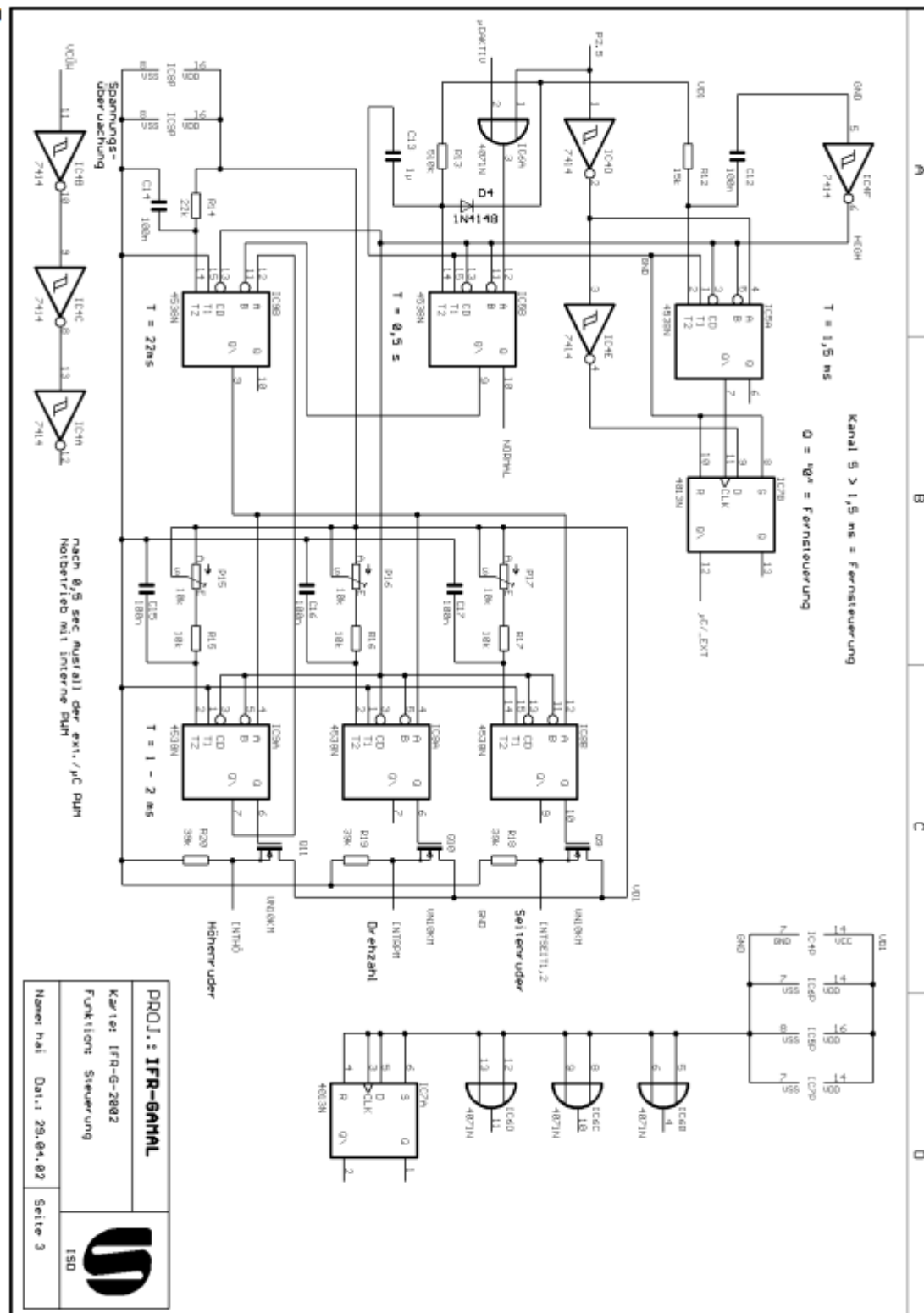


Figure 0-3: circuit design for the Basisbetriebszustands-Einstellung and the Basisbetriebszustands-Einstellung

9.9 ANNEX B: Test program 1

```
/* Measure of the length of time between successive flanks the entrance P2.3 */

#include <REG167.h>
#include <intrins.h>
#include <stdio.h>
SFR Pam Picon = 0xf1c4;
/* -----
** The direction of the ports in and out for those needing
*/
SBIT DP84) = DP8 ^ 4;
SBIT P84) = P8 ^ 4;
SBIT DP72 = DP7 ^ 2;
SBIT P72 = P7 ^ 2;
SBIT DP23 = DP2 ^ 3;
SBIT P23 = P2 ^ 3;

Unsigned int CC_N;
Unsigned int flankennr = 0; /* distinguishes between 1ter and 2nd flank */
Unsigned int T_E= 0;
Const int T1_min = 6000; /* constant to the decision on whether the timer should be stopped
*/
Const int T1_max = 7500; /* constant to the decision on whether the timer should be stopped
*/

/* -----
** Global variables for the serial interface
*/

Unsigned int t0_inhalt_start = 0;
```



```
Unsigned int t0_inhalt_stop = 0;
```

```
Unsigned int steur_reg = 0;
```

```
Unsigned int steur_zaehler = 0;
```

```
/* -----  
** Production of the PWM-signals from derPWM-unit  
* /
```

```
Void t2_isr (void) interrupt 0x22
```

```
{  
    P72 = 0;  
    PP2 = 7800;  
    PW2 = 7372) and strong support ;  
    PWMCON0 = 0x00ff;  
    PWMCON1 = 0x000F;  
  
}
```

```
/* -----  
** Limitation of the pulsewidth of the PWM signals  
* /
```

```
Void cc20ISR (void) interrupt 0x34
```

```
{  
    P84) = 0;  
    CC20 = T1_max;  
  
}
```

```
/* -----  
** Programming the PWM signals at the entrance  
* /
```

```
Void p23 (void) interrupt 0x13
```

```
{
```

```
    Charbuff_tmp [ 40];
```

```
    Int I;
```

```
    PP2 = 0;
```

```
        PW2 = 0;
```

```
        P72 = 1;
```

```
        P84) = P23;
```

```
If (flankennr= = 0)          /* First edge */
```

```
{
```

```
    T2 = 500;
```

```
    T0rel = 3000;
```

```
    T0 = 3000;
```

```
        T0R = 1;                /* starts timer T0 */
```

```
        T0_inhalt_start = T0;
```

```
        T7rel = 2950;
```

```
    T7 = 2950;
```

```
        T7R = 1;
```

```
        CCM0= CCM0 & 0x0FFF;
```

```
    CCM0= CCM0| 0x2000;
```

```
    Flankennr= 1;          /* YES -> There are still two more sides expected */
```

```
}
```

```
Else if (flankennr= = 1) /* Second flank */
```

```
{
```

```
    CC_N = CC3;
```

```
    T0_inhalt_stop = T0;
```

```
    CCM5 = 0x0005;
```

```
    CCM0= CCM0 & 0x0FFF;
```

```
    CCM0= CCM0| 0x1000;
```

```

        Flankennr = 0;

    If (CC_N < T1_min )
        {
            CC20 = T1_min;
            P84) = !P23;
        }
    Else if (CC_N > T1_max)
        {
            CC20 = T1_max;
        }
    Else
        {
            P84) = P23;
        }

/* -----
** The serial interface
*/
    If (( + +steur_reg) == 100)
    {
        For (i= 0; i< 100; i++)
        {
            sprintf(buff_tmp, "T0start: %u;Tstop: %u;imp: %u\n",
                T0_inhalt_start,t0_inhalt_stop,CC_N , steur_reg);

            Printf (buff_tmp);
        }
    }
}
}
}

```

```
Void main (void)
```

```
{
```

```
    Flankennr = 0;          /* 0.. there is still no flank occurred */
```

```
    DP72 = 1;              /* DP7.2 central as output */
```

```
    DP23 = 0;              /* sets the direction of DP2.3 to input */
```

```
    DP84 = 1; /* sets the direction of DP8.4 on output. */
```

```
    Pam Picon = Pam Picon | 0x3;
```

```
    CCM0 = 0x1000;
```

```
    CCM1 = 0x0001;
```

```
    T01CON = 0x0000; /* Timer 1 has been with period (26 ms) initialized */
```

```
/
```

```
    CC3IC = 0x0047;
```

```
    CC4IC = 0x0046;
```

```
    CC21IC = 0x0045;
```

```
    CC20IC = 0x0048;
```

```
    T2CON = 0x0003;
```

```
    T2IC = 0x0044;
```

```
    T78CON = 0x0000; /* 0 1 00 0 000 0 1 00 0 000 26 ms */
```

```
    T2R = 1;
```

```
    IEN = 1;
```

```
    While(1) {
```

```
        _idle_();
```

```
    }
```

```
}
```

Test program 2

```
#Include <reg167.h> /* Register Definitionen of the C167) */
```

```

#include <intrins.h> /* bausteinspezifische functions */
#include <stdio.h>

/* -----
** Initialization of the inputs and outputs
*/

SBIT DP26 =DP2 ^ 6;
SBIT P26 =P2 ^ 6;
SBIT DP27 =DP2 ^ 7;
SBIT P27 =P2 ^ 7;
SFR P5DIDIS = 0xFFA4;

Unsigned int adc_kanal0[200];/* buffer for the Wandlungsergebnisse */
Unsigned int adc_kanal1[200];/* buffer for the Wandlungsergebnisse */
Static unsigned int steur_reg = 0;
Static unsigned int steur_kanal0 = 0;
Static unsigned int steur_kanal1 = 0;
Static unsigned int pxt= 0;

/* In the AD-converter interrupt service routine, the Wandlungsergebnisse
Read and in the field adc_werte copied and if the value in adc_werte 16
Less than 496 (4.76 p. p. p. n V) is, is therefore in this moment by high edge to P2.6 or
higher a
HRLY RATE be switched */

Void adc_isr (void) interrupt 0x28
{
Int I;
Unsigned int test_1;
Unsigned int test_0;

```

```

Char buff_tmp[ 40];

If (ADDAT & 0x1000)
{
    TEST_0 = ADDAT & 0x03ff;
    Adc_kanal steur_kanal0+0 [ +] = ADDAT & 0x03ff; /* result of a/d process */

    If( test_0 < 800)/ * voltage small e.g. 4 V */
    {
        P26 = 1;          /* is available on the P7.8 1 (rising edge) */
    }
    Else
    {
        P26 = 0;
    }
}
Else
{
    TEST_1 = ADDAT & 0x03ff;
    Adc_kanal steur_kanal1+1 [ +] = ADDAT & 0x03ff; /* result of a/d process */

    If(test_1 < 600)/ * voltage small e.g. 4 V */
    {
        P27=0; /* is available on the P7.8 1 (rising edge) */
    }

    Else
    {
        P27 = 1;
    }
}

```

```

/* -----
From the  $\mu$ C ** data sent back to the screen
*/
If ((steur_reg) < 100)
{
    ++Steur_reg;
}
Else if ((steur_reg) == 100)
{
    For (i= 0; i< 50; i++)
    {
        sprintf(buff_tmp, "Channel1: %u, Kanal0: %u\n",
                Adc_kanal1 [i], adc_kanal0 [i] );
        Printf (buff_tmp);
    }
    ++Steur_reg;
}
}

/* -----
** With GPT1 is the conversion each time after time start bestemmte
*/
Void gpt1T3 (void) interrupt 0x23
{

    ADST = 1; /* Conversion start */
}

```

```
Void main (void) {  
  
    DP27 = 1;  
    P27 = 1;  
    DP26 = 1;  
    P26 = 1;  
    P5DIDIS |= 0x0001;  
    Adcon= 0xf221;  
    T3CON = 0x0002;  
    T3 = 0x0BDC;  
    T3IC = 0x006B;  
    T2CON = 0x0020;  
    T2 = 0x0BDC;  
    ADCIC = 0x004C;  
    IEN= 1;  
    T3R = 1;  
    While(1){  
        _idle_();  
    }  
}
```


10 Communication and User Interface

Based on the bachelor thesis of Rabih al-Farkh, "*Development of a communication interface for a measurement system*", June 2002, Lebanese University – Tripoli/Lebanon, Faculty of Engineering

Abstract

A communication system which includes a graphical user interface has been developed to control a mobile sensor platform, and to receive measurement data from this mobile sensor platform. The measurement system is used to measure alternative energy resource values like solar power and wind vector at different points.

Keywords: graphical user interface, airship, transceiver, wireless communication, V-model, structured analysis, structured design, software engineering.

10.1 Introduction

10.1.1 The LOTTE project and the "alternative Lotte"

Airships are becoming more and more important within the last years. At the "Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktion" at the University of Stuttgart a solar airship was built. In February 1992 the project "Solarluftboot" was initiated. The purpose of this project was to find new materials, new construction methods and a new concept for controlling and navigating an airship run by a solar energy engine.

The "alternative Lotte" – a cooperation project between the universities of Karlsruhe and Stuttgart and the Fachhochschule Karlsruhe - is planned to be an experimental airship which can be controlled directly from the ground (first step of implementation) or (second step of implementation) fly automatically to specified coordinates. The energy supply is going to be conventional batteries in the first step but it is planned to switch to solar energy later. With "alternative Lotte" environmental data are collected during the flight via different sensors which can be mounted on the airship. In the first step the speed of the wind, the temperature and the solar radiation are measured. Apart from that flight data such as acceleration, angular velocity and azimuth angle are measured for navigation purposes.

10.1.2 Development method

The software is developed using structured analysis / structured design (SA / SD). This method is very common in industrial development processes as well. The complete communication system is tested in a laboratory using two PCs and the transceivers. The whole development process basically follows the known V-model.

10.1.3 Working task and realization approach

The task is to develop a communication interface which is software running on a PC which is connected with the transceiver on the ground. Apart from that a program which is running on the board computer has to be developed. The purpose of this program is to establish the communication between the sensors and the transceiver on the “alternative Lotte” sensor platform.

So the diploma thesis can be divided into several tasks:

- Specification document, SA diagram (Structured Analysis) and SD diagram (Structured Design) for the User Interface.
- Programming of the User Interface using Visual Basic 6.0.
- Specification document, SA/SD for the communication protocol.
- Programming of the communication protocol with programming language C.
- Programming of the protocol for sending and receiving with an off-the-shelf transceiver cards.
- Testing the communication system with the User Interface.

10.1.4 Overview

After a short introduction some basic knowledge about the used development method SA / SD, the V-model and some general information about transceivers is provided.

In the third chapter a detailed description of the user interface and the transceiver is given. Then the system architecture of the whole system is presented.

Chapter 5 contains the SA, SD of the alternative Lotte communication system and a brief description of the software implementation. After that the development environment is explained.

The last chapter is about the tests that have been done to verify the functionality of the communication system.

Detailed figures of the hardware and the code of the software can be found in the annex.

10.2 Basics

10.2.1 The V-Model

The **V-Model** is an extensive collection of knowledge about the best practices of software development. This knowledge can be described as a process: In addition to the planned products and activities, the **V-Model** also contains information about the course the project will take. To this end, the process standard includes which output products are to be created by an activity and which successor activities need this product as input. This internal product flow allows you to derive a chronological order for the activities.

The V model is a logical product model. It describes the contents of the products which have to be created during a software project and their relationships on a very high level. However, in a real project the physical structure of these products might (and in most cases will) be quite different [1].

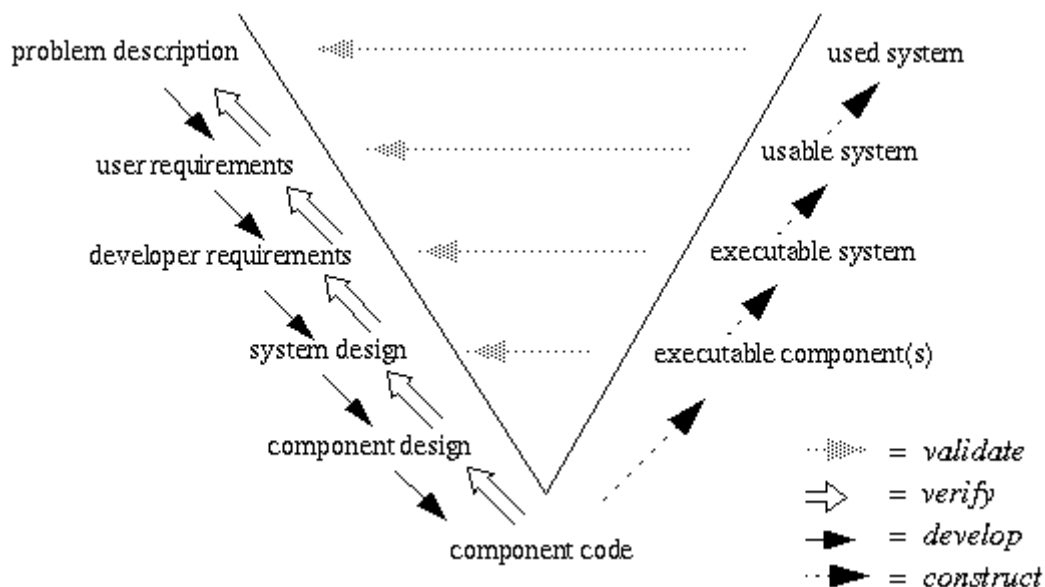


Figure 2.1: The V-Model

The left side of the V shows the products resulting from development activities: problem description, user requirements, developer requirements, system design, component design and component code. The products are not further described here.

The right side of the V documents the construction of the final system starting with individual components. The products on this side of the V are called executable components, executable system, usable system and used system.

Besides development and constructional activities the V model also contains some analytical activities: verification and validation.

Verification takes place after each development activity. The output of the activity is verified against the input in order to check if both actually match.

Validation happens after each construction step to assure that the so far constructed system behaves as it should. It is therefore tested against the corresponding product on the other side of the V model.

It should be emphasized that the V model is no physical product model nor any kind of process or life cycle model. However, due to its very general description, products used in a life cycle model can always be matched with the products contained in the V model. Therefore, it can be seen as a generic product model.

The **V-Model** is a Lifecycle Process Model, originally developed to regulate the software development process within the Federal Administration of the Federal Republic of Germany.

The **V-Model** is composed of four submodels: software development, quality assurance, configuration management, and project management.

The submodels are closely interconnected and mutually influence one another by exchange of products and results. The software development submodel develops the system or software. The quality assurance submodel submits requirements to the other submodels and test cases and criteria to assure the products and the compliance of standards. The configuration management submodel administers the generated products. The project management model plans, monitors, and informs the other submodels. Each can be further decomposed. For instance, the software development submodel can be broken down as follows (see [2]):

- System requirements analysis and design.
- Data processing requirements analysis and design.
- Software requirements analysis.
- Preliminary design.
- Detailed design.
- Implementation.
- Software integration.
- Data processing integration.
- System integration.

10.2.2 Structured Analysis (SA) / Structured Design (SD)

In the seventies and eighties of the 20th century *Structured Analysis (SA)* was described by Yourdon, Constantine, and DeMarco. These practices became very popular, and by the early eighties had a profound effect upon the definition of

analysis. **SA** was a technique in which the requirements of the customer were broken down into a hierarchy of functions. This breakdown was known as *functional decomposition*. Datasets were specified in abstract (Bacchus-Naur) terms, and their manipulation were depicted through functionally decomposed data flow diagrams.

SA was coupled to another practice known as *Structured Design (SD)*. Indeed, the two were often mentioned in the same breath as **SA/SD**. **SA** described the data sets and data transformations implied by the requirements. As such, **SA** described *what the system would do*, albeit in very technical terms. On the other hand **SD** described *the partitioning of the software into modules*, and the flow of data between those modules. Therefore an **SD** was, more or less, a **description** of *how* a system would be structured to meet the requirements. **SA/SD** contained a practice known as *The Transform Analysis*, which was used to convert the diagrams representing a **Structured Analysis** into the diagrams that represented a Structured Design. This practice, and indeed much of the documentation of the period, established the notion that the design was directly derivable from the analysis by applying some simple transformation rules. This meant that the analysis was really a preliminary **description** of the design, requiring only a mapping operation to complete. The view of **SA/SD** was a remarkable change from the systems analysis of the sixties. In the sixties no such mapping from analysis to design was implied. The analysis simply described data sets

and their transformations without implying anything about software structure. **SA/SD** also implied a temporal constraint between analysis and design that was not practiced by the Systems Analysis of the sixties. In **SA/SD** it was necessary to finish the analysis before the Transform Analysis could be applied to translate the analysis into a design. Thus, **SA/SD** strongly reinforced the waterfall model of development [3].

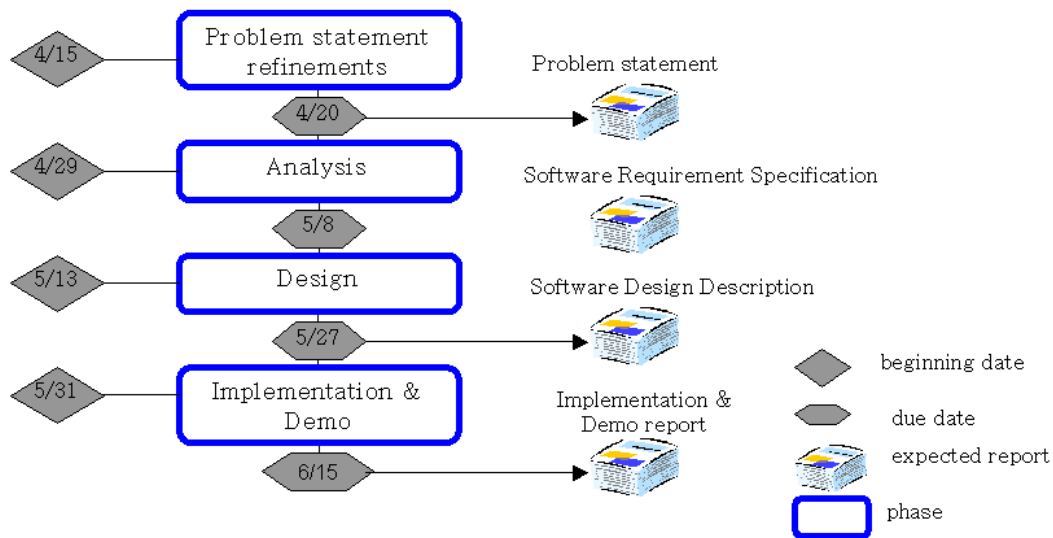


Figure 2.2: SA/SD (see [4])

10.2.3 Transceivers

There are three wireless RF modules: transmitter, receiver and transceiver. These RF modules are designed to serve as a tool for electronics design engineers, developers and students to perform wireless experiments.

The transmitter is used to transmit data and the receiver is used to receive data, so they are used in application of one-way communication. The transceiver is used to transmit and receive data, so it is used in application of two-way communication. All of the RF modules which are used in this project have 9,600 bps serial interfaces at maximum. The used modules can communicate over range up to 250 feet. Generally the range depends on the power and the frequency of the RF module.

Radio frequency (RF) refers to electromagnetic waves that have a wavelength suited for use in radio communication. Radio waves are classified by their frequencies, which are expressed in kilohertz, megahertz, or gigahertz [5].

The RF transceiver controls and modulates the radio frequencies that the antenna transmits and receives.

10.3 Requirements Specification

10.3.1 The Graphical User Interface (GUI)

The user interface is the link between the user and the alternative – LOTTE, a measurement vehicle for solar radiation and wind power. Its purpose is allowing the user to control the alternative – LOTTE from the base station, so by using the User Interface, the user can set data to the alternative – LOTTE (new position, new velocity ...) and get data from the

alternative – LOTTE (acceleration, angular velocity, azimuth, temperature, wind vector, Solar radiation ...).

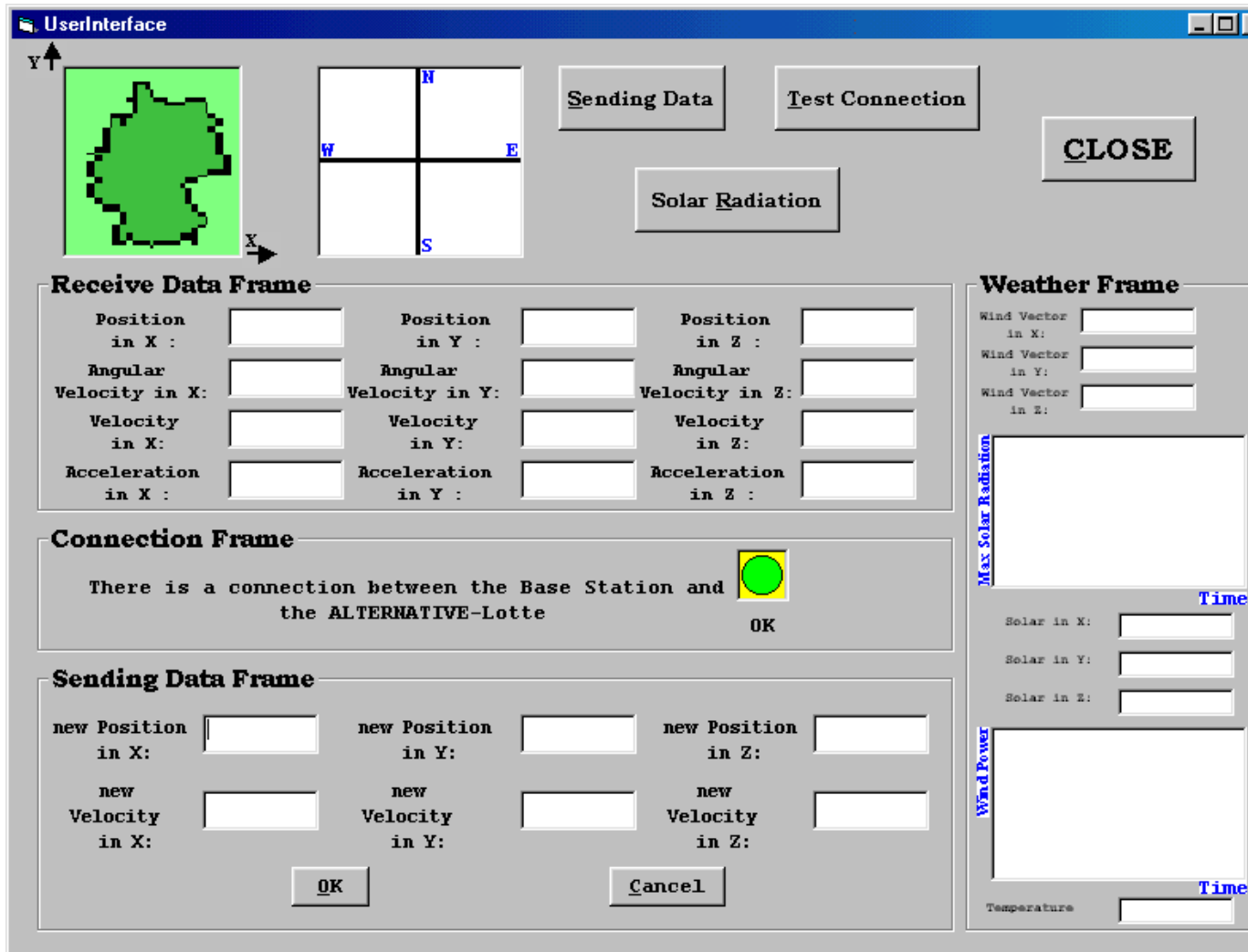


Figure 3.1: User Interface

10.3.1.1 Software Requirements Specification

- The user interface contains a frame (Receive_Data_Frame) inside which, we find several text boxes or labels.

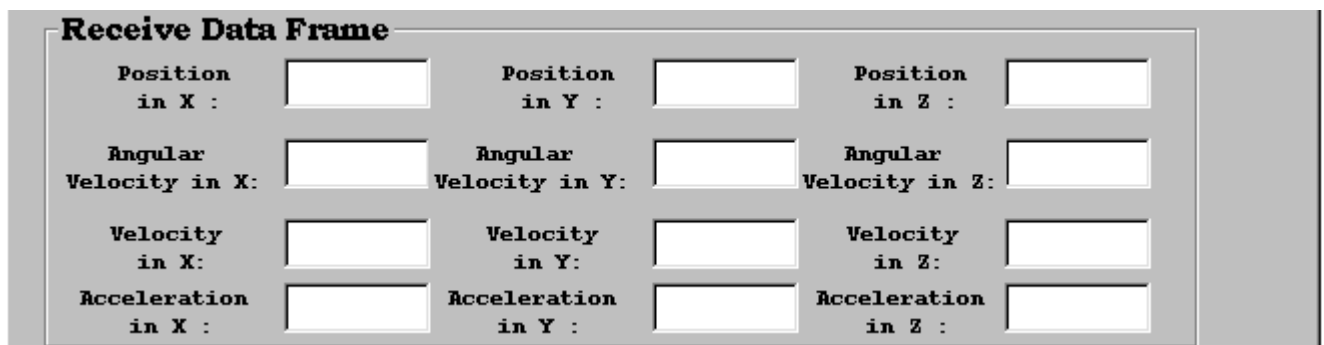


Figure 3.2: Receive Data Frame

- These several text boxes or labels are used to display the data that the base station received from the alternative - LOTTE. These data are the data sensors (acceleration in X, Y and Z, angular velocity in X, Y and Z, azimuth, temperature, solar "Solar Radiation" and wind vector). As to the azimuth parameter, it can not be seen by the user. It is used by the user interface to display the direction of the alternative - LOTTE in a picture box. These sensor data will be sent from the alternative - LOTTE to the base station. The user interface uses these data to calculate the velocity of the alternative - LOTTE in X, Y and Z and to calculate the position of the alternative - LOTTE in X, Y and Z.
- The user interface contains command button shown below, this command button is used whenever the user wants to send data (commands) to the alternative - LOTTE. When the user presses this command button, a Send_Data_Frame will be displayed, (Sending_Data_Frame). The commands that the user can send to the alternative - LOTTE are the new position in X, Y and Z, and the new velocity in X, Y and Z.



Figure 3.3: Sending Data Button

- The user interface contains Sending_Data_Frame, in this command frame, we find several text boxes that are used by the user to put the data (commands) that he will send to the alternative - LOTTE. These data (commands) are the new position in X, Y and Z, and the new velocity in X, Y and Z.

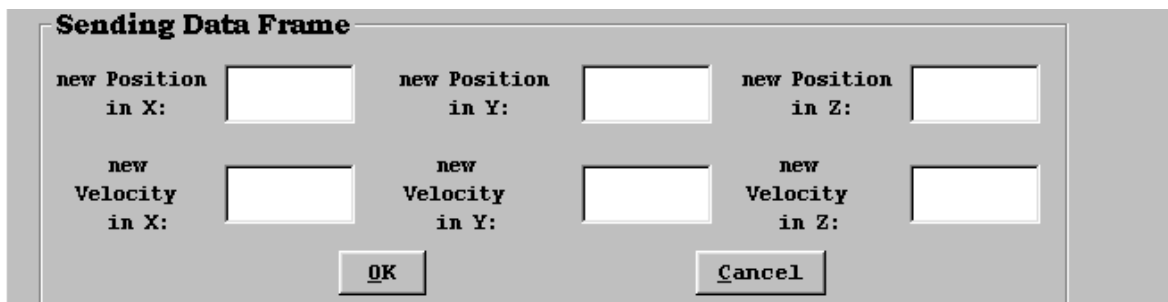
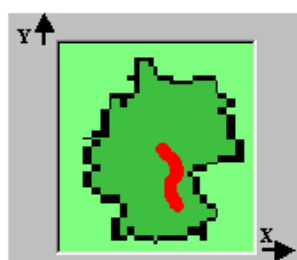


Figure 3.4: Sending_Data_Frame

- The user interface contains picture box that is used to display the position of the alternative - LOTTE on a map (two dimensions x, y). In this picture box the user will be able to see the position of the alternative - LOTTE during its flight.

Figure 3.5: PictureMap



- The user interface contains another command button. The function of this command button is to test if the connection between the base station and the alternative – LOTTE is **OK**. The result is displayed in the Connection_Frame. In fact, the connection is always put to the test whenever the user receives data from the alternative – LOTTE (which means that the connection is **OK**). If the user wants to make sure about the connection at any time all he has to do is to press the Test – Connection Button.



Figure 3.6: Test-Connection Button

- The user interface contains a Connection_Frame that is used to tell the user that the connection is OK while the alternative - LOTTE is in the air. In case there is a problem in the connection between the base station and the alternative - LOTTE, it is displayed in this frame, and the user would be able to see that there is a problem.

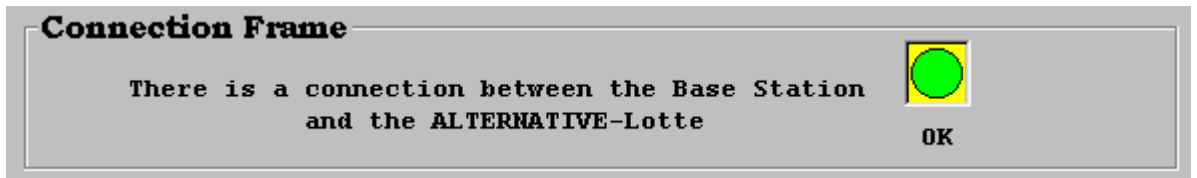


Figure 3.7: Connection Frame

- The user interface contains a picture box that is used to display the direction of the alternative - LOTTE while it is in the air. The user interface will get the azimuth from the alternative – LOTTE and use this parameter to display the direction of the alternative – LOTTE.

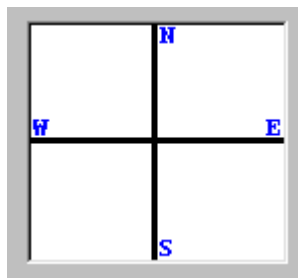


Figure 3.8: PictureDirection

- The user interface contains a weather frame that is used to tell the user the status of the weather while the alternative – LOTTE is in the air. In this frame, the user can observe the temperature, the wind vector, and the solar radiation (solar radiation in X, Y and Z). The user interface get these data from the sensors data.

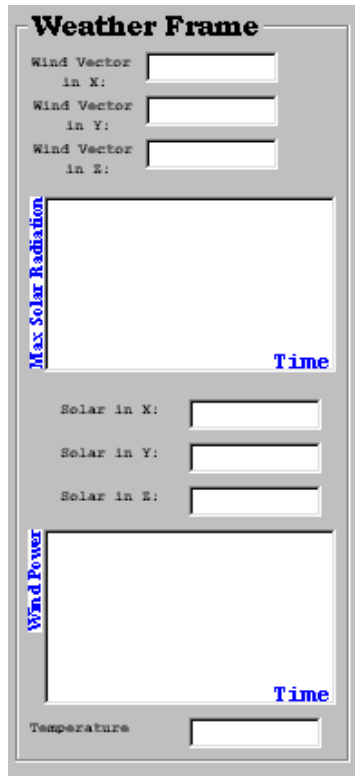


Figure 3.9: Weather Frame

- The user interface contains a third command button. The function of this command button is to display the solar radiation in front of the user in three dimensions. When the user press this button, he will see the position of the alternative – LOTTE in three dimensions with the value of the solar radiation in each position.



Figure 3.10: Solar Radiation Button

- When the user presses the Solar Radiation button, the figure shown below shall be displayed :

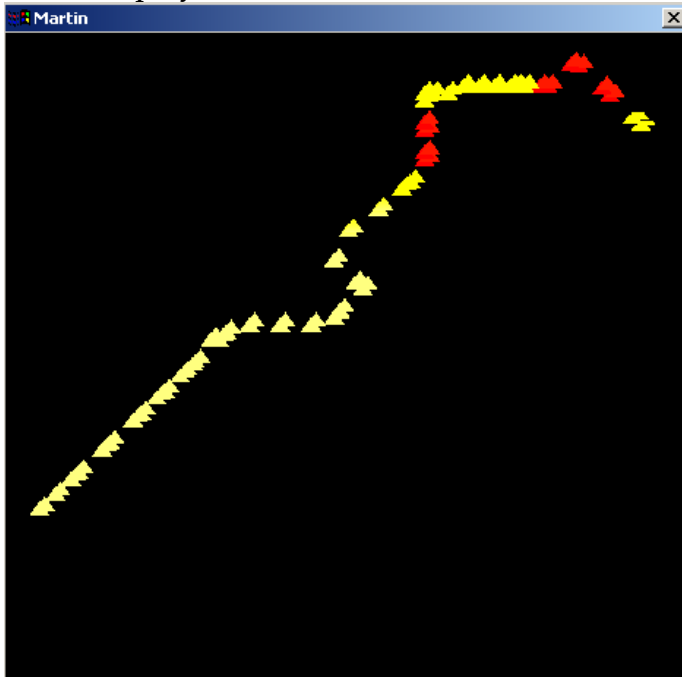


Figure 3.11: Solar Radiation show

In this figure, the position of the alternative – LOTTE is displayed with the power of the solar radiation, the color at the position shows the intensity of the solar power at this point.

10.3.2 Specifications for the transceiver

The goal of this transceiver is send and receive data between the Base station and the alternative – LOTTE. There are two transceivers, one is connected to the base station and the other is connected to alternative – LOTTE.

10.3.3 Communication Protocol for User Data

In each frame, the transceiver processes the data internally and input/output user data. User data can be specific by user. In general, data is send and receive as “packet”.

- The form of the telegram that the base station receives from the alternative – LOTTE is :

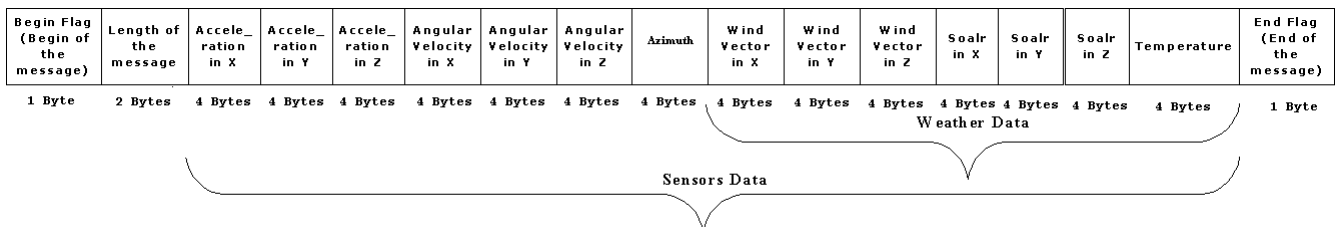


Figure 3.12: User Data Protocol _ Receive Message

- ◆ The **Begin flag** is the beginning of the telegram (message) and the length of this command is 1 byte (8 bits). It is a special byte that the user uses to tell the alternative – LOTTE and the base station that it is the beginning of the telegram (message).
- ◆ The information length of the telegram is on 2 bytes, and it contains an information about the length of the telegram (message) that the base station receives from the alternative – LOTTE, or sends to the alternative – LOTTE.
- ◆ The sensors data contains information about the acceleration in X, acceleration in Y, acceleration in Z, angular velocity in X, angular velocity in Y, angular velocity in Z, the azimuth, wind vector in X, wind vector in Y, wind vector in Z, solar in X, solar in Y, solar in Z and the temperature. Each information is in 4 bytes and the type of this data is float.
- ◆ The **End flag** is the end of the telegram and the length of this command is 1 byte. It is a special byte. So when the base station or the alternative – LOTTE receives this byte, it knows that this is the End of the telegram and it waits for another telegram from the alternative – LOTTE.

The form of the telegram (message) that the base station sends to the alternative – LOTTE is :

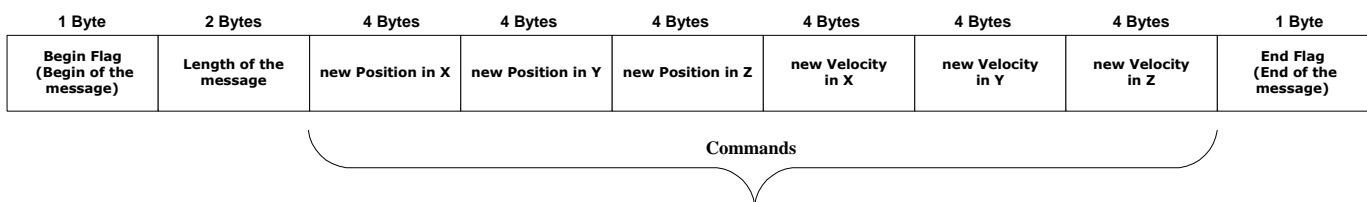


Figure 3.13: User Data Protocol _ Send Message

The sent and the received telegram have the same protocol (Begin_Flag , Length of the Telegram, Data or Commands (information) and then the End_Flag). We can send the begin flag at first, then the length of message (commands), then the commands (which gives the new position and the new velocity), and finally the end flag to tell the alternative – LOTTE that the telegram is finished.

10.3.4 Handshaking

Handshaking refers to the internal communications protocol by which data is transferred from the hardware port to the receive buffer. When a character of data arrives at the serial port, the communications device has to move it into the receive buffer. A handshaking protocol ensures that data is not lost due to a buffer overrun, where data arrives at the port too quickly for the communications device to move the data into the receiver buffer.

10.4 Architecture Design

The purpose of the user interface is to control the alternative – LOTTE from the base station and the purpose of the transceiver is to send and receive data between the base station and the alternative – LOTTE. The connection between the user interface, the transceiver and the whole system is shown in the figure below :

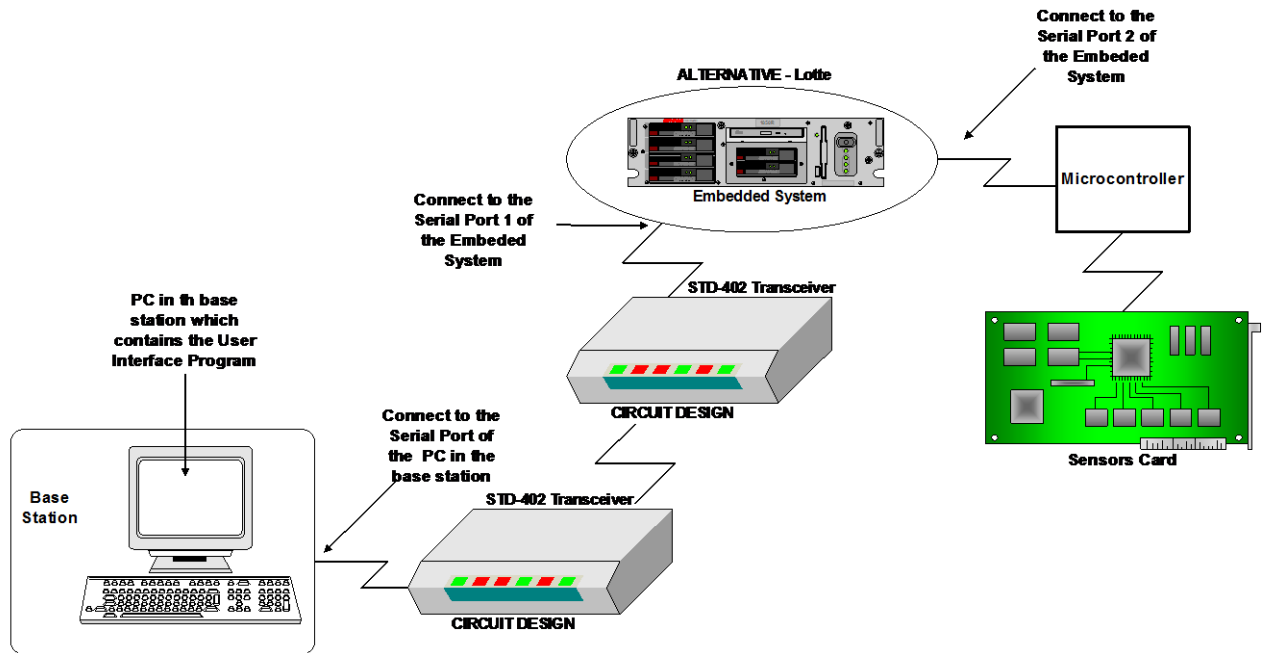


Figure 4.1: System Connection

This figure shown above describes the connection between the PC in the base station which contains the user interface program and the transceiver, and the connection between the embedded system in the alternative – LOTTE and the other transceiver. This connection is done through the serial port.

Apart the connection between the embedded system in the alternative – LOTTE, the transceiver card and the sensors card is shown in the figure above. In fact the sensors card is connected to the microcontroller which is connected to the embedded system through the serial port. The purpose of the microcontroller is to transfer the sensors data from the sensors card to the embedded system. The embedded system gets these sensors data from the serial port which the microcontroller is connected to and sends these data to the serial port to which the transceiver card is connected.

The transceiver gets these sensors data from the embedded system through the RS232C connector, then the data are stored in internal buffer. After the transceiver checks that the carrier frequency to be set is not used on air, these data will be sent to the base station using one of the channels frequencies between the 433.200 – 434.775 MHz. The same thing is done when the transceiver gets data from the base station and will sent to the alternative – LOTTE

The modulation used is the FSK modulation (Frequency Shift Keying), this type of modulation enables transmission at a maximum of 9,600bps. Also when multiple channel are to be used simultaneous or if high communication standards are required, the MSK modulation enables transmission at a maximum of 2,400bps.

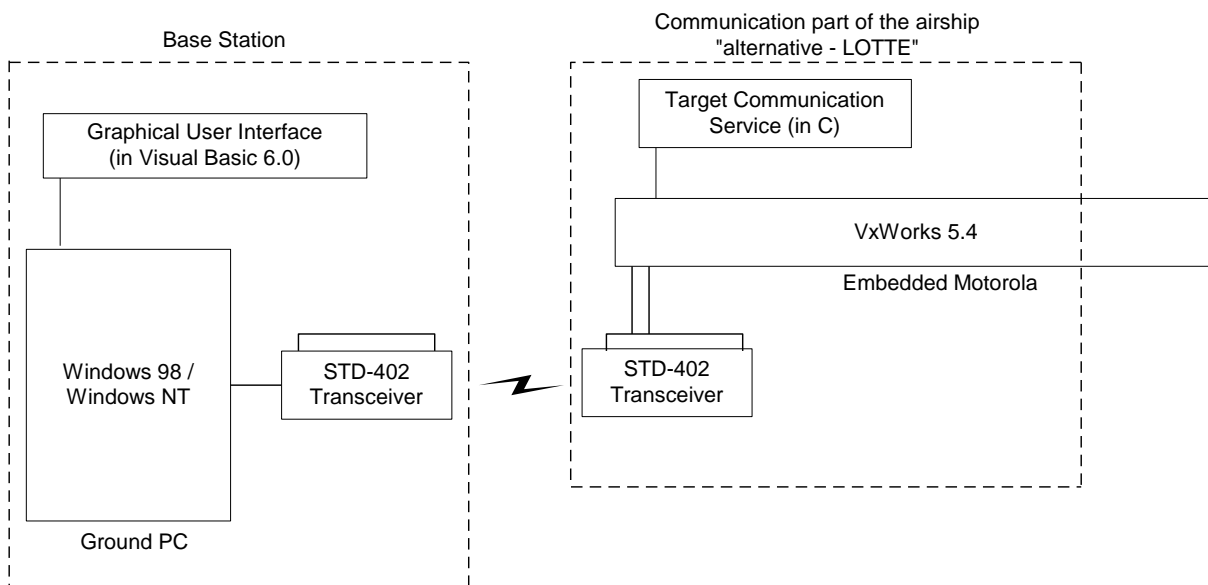


Figure 4.2: The “alternative Lotte” Communication System

10.5 Development Environments

The development has been done on a PC with AMD – k5 tm processor, AT / AT compatible, 64 MB RAM and a PC with a PENTIUM – S processor 100 MHz, 96 MB RAM.

The following software tools have been used for the development:

- MS Windows 98, MS Windows 2000, MS Office 97 and MS Office 2000.
- Windows 2000 server for workflow Management, Linux.
- MS Visual Basic 6.0 language and C language. (program C is done under VxWorks).
- Visual Basic 6.0 compiler for Visual Basic program.
- "Gcc" compiler for C program under UNIX.
- Tornado 2.0 environments for C program.
- VxWorks 5.4 compiler for C program.
- Optional : MS Visio 2000 for the graphical visualization of processes, Structured Analysis (SA) / Structured Design (SD).

10.6 Design and Implementation

10.6.1 SA / SD and Implementation for the base station program

10.6.1.1 SA (Structured Analysis)

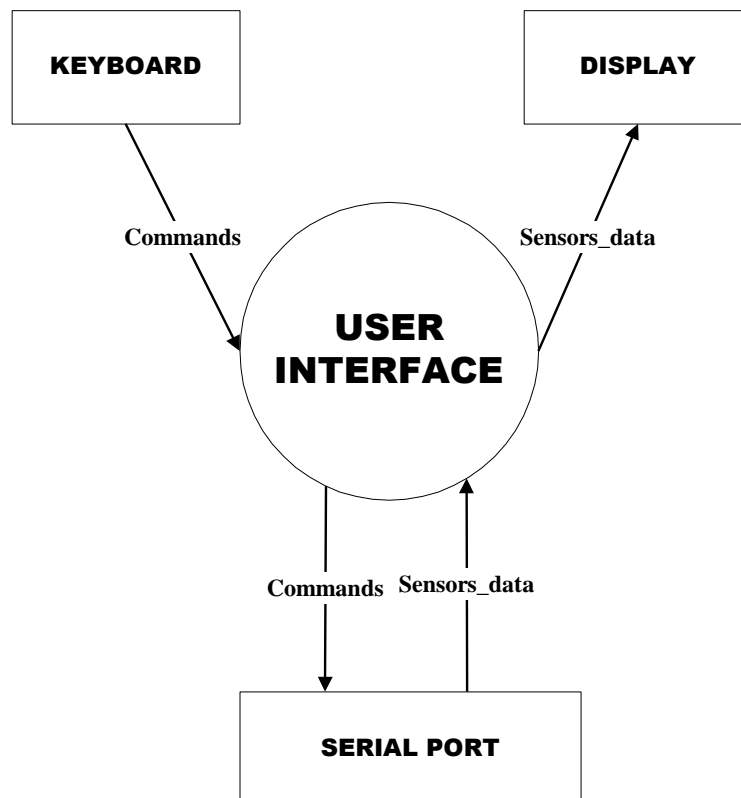


Figure 6.1: Context Diagram

- The **DISPLAY** is the screen of a PC in the base station that can be used to observe the received sensors data. These sensors data are received from the alternative – LOTTE using the serial port of the PC.
- The **KEYBOARD** is used by the user to write the commands that he wants to send to the alternative – LOTTE. These commands are sent to the alternative - LOTTE using the serial port. In fact, whenever the user wants to send commands to the alternative – LOTTE, the user interface will send them to the serial port.
- The **SERIAL PORT** is the peripheral between the user interface and the alternative – LOTTE. So by using the user interface, we can say that the user will get data (from the alternative – LOTTE) and set data or commands (to the alternative – LOTTE).

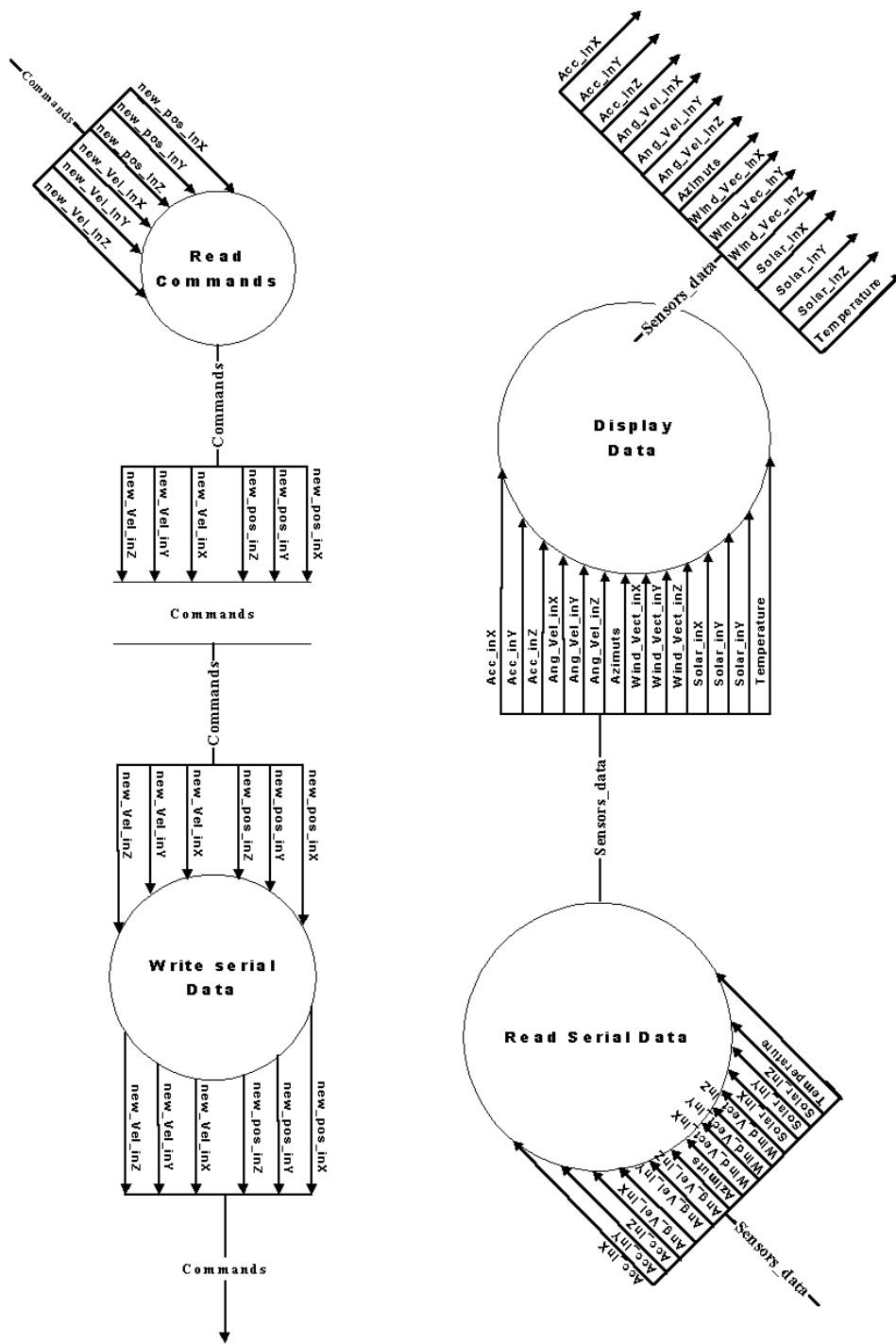
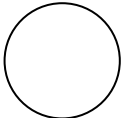


Figure 6.2: data flow diagram (DFD) of the process “User interface”

In this second diagram, we can see all the functions that the user interface can perform .

- Some remarks for structured analysis (SA) convention:

The symbol  means that we have a process and this symbol


_____ means that we have a storage.

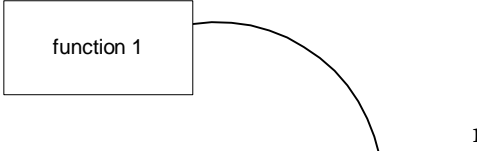
- In the second increment a security layer is added to the communication protocol. The commands that the base station sends to alternative –LOTTE will be acknowledged. If the acknowledgement fails the commands are resent.


ID	Features	Increments
01	<ul style="list-style-type: none"> ➤ Send commands to the alternative – LOTTE ➤ Receive data from the alternative – LOTTE 	1
02	<ul style="list-style-type: none"> ➤ Save Commands that the user sent to the alternative – LOTTE. ➤ Waiting for an answer from the alternative – LOTTE about what it has Received. ➤ Compare this answer with the Telegram (message) that the user sent. 	2
03	<ul style="list-style-type: none"> ➤ Send commands with saving it to compare with the answer from the alternative – LOTTE. ➤ Receive data from the alternative – LOTTE. 	All

10.6.1.2 SD (Structured Design)

Some remarks for structured Design (SD) convention:

The symbol  means that we have a function, and in this symbol we can find the same name of the function used in program.

this relation  means that the function1 calls function 2.

The symbol  describes the return value of the function


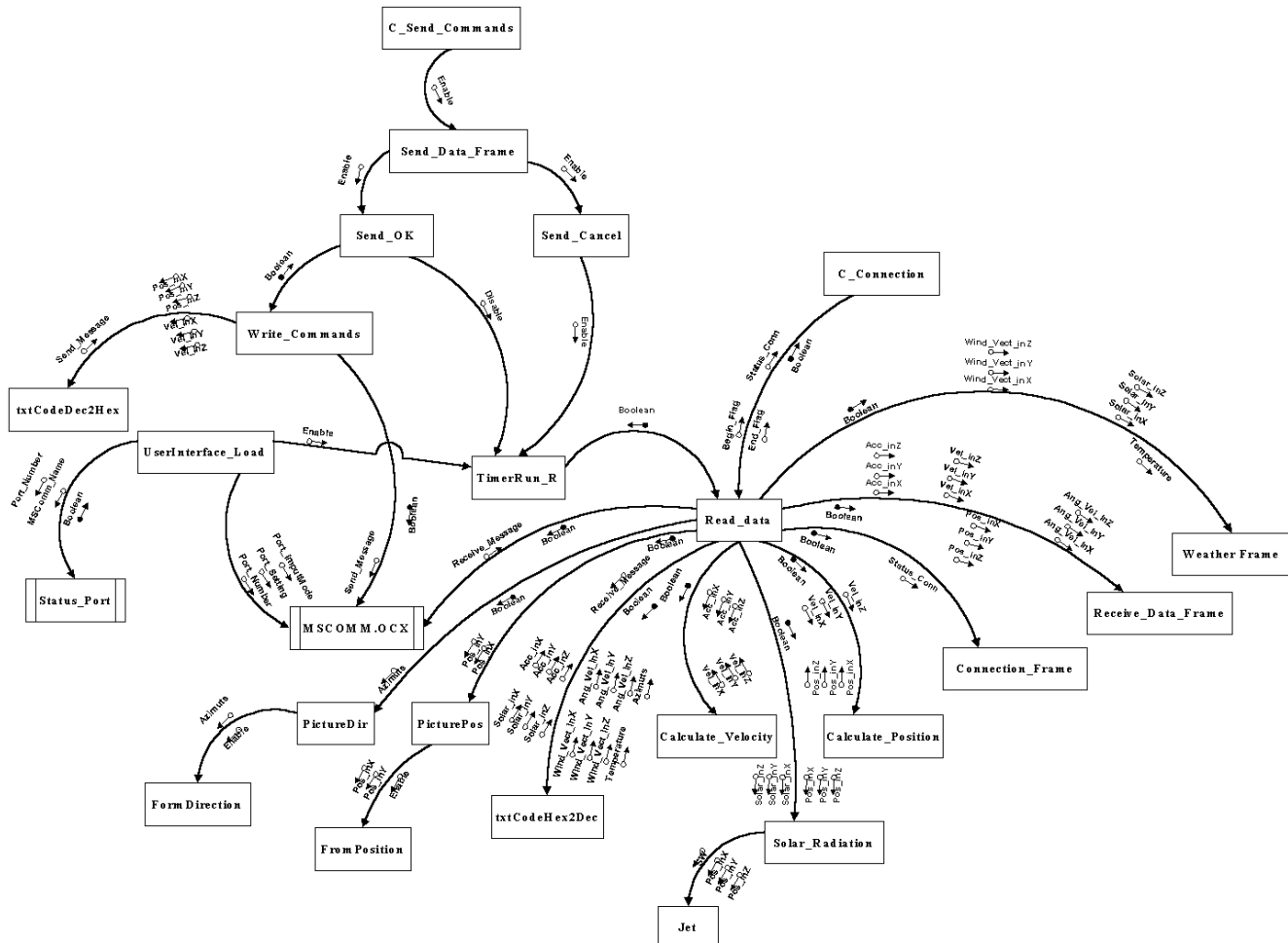
The symbol  describes the input parameters of the function.

Figure 6.3: Structured Design (SD) diagram for the user interface



Description of the functions

MSCOMM

This is an OCX (Microsoft Comm Control 6.0) which contains all functions that the programmer can use when he wants to program the serial port. Using this OCX, we can manage the serial port from Visual Basic which means that we can open the port, put the settings of the port, read from serial port and write to serial port

UserInterface Load

This function runs automatically when the user runs the Program. The purpose of this function is to give the MSCOMM the number of serial port that the user will open and the settings and the InputMode. It will also enable the TimerRun_R and testing the status of the serial port.

Timer_Run_R

This function contains the interval which the user interface uses to read data from serial port if this serial port was not busy. The serial port being busy means that it is not used at the moment. For example, if we define the interval time like one second, so every one second the program will read data from serial port. Of course these data are the sensors data, and it contains information about the acceleration of the alternative – LOTTE and the angular velocity, the azimuth, the solar, the wind vector, and the temperature.

Read_Data

The purpose of this function is to read sensors data from serial port and to use these data to calculate the velocity of the alternative – LOTTE and its position. After that it sends these data to the Receive_Data_Frame which is then displayed in front of the user. Of course the user can not change these data, he can only see it. (Of course this requires that the serial port is not busy doing any other function like sending commands to the alternative – LOTTE, or testing the connection between the base station and the alternative – LOTTE).

Write_Commands

The purpose of this functions is that the user will be able to send commands from the base station to the alternative – LOTTE, using the serial port. The user can write commands after disabling the TimerRun_R. but if the TimerRun_R is enabled, this means that the serial port is used only to Read_Data.

C_Send_Commands

The purpose of this function is to enable the `Sending_Data_Frame` that it used by the user to write the sending commands (new position in X, Y and Z, and new velocity in X, Y and Z) that he wants to send to the alternative – LOTTE.

Send_OK

This function will confirm that the user wants to send the commands that he writes in the `Sending_Data_Frame` to the alternative – LOTTE after testing the status of the serial port. This is because the serial port maybe busy doing another function at that time like receiving data from the alternative – LOTTE

Send_Cancel

This function is used by the user if he clicks the Sending Data Button and then he changes his mind and does not want to send commands to the alternative – LOTTE. So this function will disable the `Sending_Data_Frame`, and enable the `TimerRun_R` (which means that the serial port is not busy sending commands right now, but rather the program is enable to read sensors data).

C_Connection

The purpose of this function is to test the connection between the base station and the alternative – LOTTE. It refreshes the `Connection_Frame` and gets the status connection from the `Read_Data` Function. It also displayed the status of the connection on the `Connection_Frame`.

PictureBox_Position

The purpose of this function is getting data from the `Read_Data` function (position in X, position in Y) and displaying the position of the alternative – LOTTE on the map.

PictureBox_Direction

The purpose of this function is getting data from the `Read_Data` function (Azimuth) and displaying the direction of the alternative – LOTTE in the `PictureBox`.

Calculate_Velocity

The purpose of this function is to get the Acceleration from the `Read_Data` function and calculate the velocity of the alternative – LOTTE in X, Y and Z and gives this information to the `Receive_Data_Frame` that it displays.

Calculate_Position

The purpose of this function is to get the velocity from the Calculate_Velocity and calculate the position of the alternative – LOTTE in X, Y and Z and gives this information to the Receive_Data_Frame that it displays.

TxtCodeHex2Dec

The purpose of this function is to get the sensors data from the Read_Data function and to convert this data from the hexadecimal format to the decimal format.

TxtCodeDec2Hex

The purpose of this function is to get commands from the Write_Commands function and to convert this commands from the decimal format to the hexadecimal format.

FormPosition

This function run when the user press on the PicturePosition if he wants to see a large view. It contains a large picture box that inside it the user can show how the alternative – LOTTE change its position.

FormDirection

This function run when the user press on the PictureDirection if he wants to see a large view. It contains a large picture box that inside it the user can show how the alternative – LOTTE change its direction.

Solar_Radiation

This function run when the alternative – LOTTE finishes its travel. It takes the position in X, Y and Z and the Solar Radiation in X, Y and Z from the Read_Data function and uses these parameters to put the position of the alternative – LOTTE in the three dimensions with the value of the solar radiation in each position. The color of the position (point of position) changes with the value of the solar radiation.

Jet

This function run when the user press the solar radiation button. It takes the parameters (position in X, Y and Z and the solar radiation) and puts it in a three dimensional figure.

10.6.1.3 Implementation

The whole VB program code is in Annex E.

In this paragraph, the important things done in the software part will be described. At first, when the user interface program is used, the user has to care about these commands :

- `MSComm1.CommPort = 3` (*line of code: 132*)

this command is used to open the serial port number 3. Changing the number of the port is allowed. [6]

After that the initialization of the baud rate of the serial interface, the parity status, the data length and the stop bit will be done as below :

➤ `MSComm1.Settings = "9600,E,8,2"` (line of code: 133)

Then the initialization of the handshaking protocol is done by this command :

➤ `MSComm1.Handshaking = comRTSXOnXOff` (line of code: 134)

When the handshaking is set to the value shown above, the RTSEnable property will be set to TRUE as shown below :

➤ `MSComm1.RTSEnable = True` (line of code: 135)

The input mode used is input text mode, this mode is set by the command shown below :

➤ `MSComm1.InputMode = comInputModeText` (line of code: 136)

Setting the InputLen property to a number causes the communications control to read this number of bytes from the receive buffer.

➤ `MSComm1.InputLen = 116` (line of code: 137)

So by using all the commands shown above, the initialization of the serial port is completed and the serial port can be opened by this command :

➤ `MSComm1.PortOpen = True` (line of code: 138)

An important parameter used is the interval of the timer. The value putted in this parameter is used by the program to read the data from the receive buffer of the serial port.

- `TimerRun_R.Interval = 500` *(line of code: 155)*

The value used is 500 msec, so each 500 msec reading data from the receive buffer is done.

Three files are used by the user interface program. The "Receive_file.rcv" is used to save the telegram that the base station received from the alternative – LOTTE. In this file, only the correct telegram will be putted. The "Send_file.snd" is used to save the telegram that the base station sent to the alternative – LOTTE. The "Solar_file.rad" is used to save the position of the alternative – LOTTE in X, Y and Z and the solar radiation value. When the Solar Radiation button is pressed, this file will be opened to get the data from it and use these data to put the position of the alternative – LOTTE in a 3D figure with the value of the solar radiation. The command used to open a file from the visual basic is shown below :

- `Open "C:\User Interface\Data\Send_file.snd" For Binary Access Write As #2` *(line of code: 104)*

So before the user interface program putted in use, it is very important to put the correct path of the three files described above.

One program is used with the user interface. The purpose of this program is to showing the position of the alternative – LOTTE in a 3D figure with the value of the solar radiation during its flight. The command used to run this program from the user interface program is shown below :

- `Shell "C:\3D\Jet.exe", vbNormalFocus` *(line of code: 85)*

So it is very important to put the correct path of this program.

This program used with the user interface is called "Jet". This is an visual basic program. In this program, the "Solar_file.rad" is opened to get the data from it and used it to put the position of the alternative – LOTTE and the value of the solar radiation in a 3D figure as we said above. So it is very important to be care about the path of the "Solar_file.rad" file before using this program.

In the "Jet" program, the value of the solar radiation is always putted to the test whenever a value is received from the alternative – LOTTE. The purpose of this test is to changed the color of the position of the alternative – LOTTE, because the color of the point will give the user an idea about the power of the solar radiation at this point.

For the graphical show, a file is used (CURSOR20.ICO) when the user selects a region in the figure map for doing a zoom. This file must be in the same directory with the user interface program, or the path of this file should be specified.

➤ `PictureMap.MouseIcon = LoadPicture("cursor20.ico")` *(line of code: 198)*

Another program is developed using visual basic. The purpose of this program is just to build of a test environment with connected one of PC to the transceiver and connected the other PC to the other transceiver. By using this program, only sending and receiving data is allowed. This program uses a Send_Data function to send telegrams.

Note : If all the files and the programs used with the user interface program is putted in the same directory with it, so the path file will not be important, only the name of the file or of the program should be written.

10.6.2 Communication part on the board system on the airship

10.6.2.1 The transceiver

The transceiver to be used is **MB-STD-RS232**. It is a bi-directional semi-duplex radio modem having RS232 serial interface. It uses CIRCUIT DESIGN 's standard 434 MHz FM Narrow Band transceiver module **STD-402** transceiver for RF part. This transceiver was selected because of its frequency of 434 MHz. For this frequency in Germany there is no extra permission necessary. Another reason is that this transceiver is a cheap one.

The **STD-402** transceiver is an UHF Narrow Band Multi channel Transceiver.

The UHF FM-Narrow Band semi-duplex radio data module **STD-402** equipped PLL controller in its robust metal housing. Unlike other transceivers, the **STD-402** is ready to transmit RF data without complicated controller board. The compact size and low power consumption of the **STD-402** make it ideal for battery operated applications where its interference rejection and practical distance range are much better than similar RF modules based on Wide Band SAW – resonator frequency devices.

Most of RF setting are done by internal microcomputer, which allows the user to manipulate the module without professional knowledge of RF circuit.

Special for MB-STD-RS232



Figure 6.4: MB-STD-RS232 – CIRCUIT DESIGN

The RF part complies with the European radio, EMC and safety requirements and has been notified in major European countries under the R & TTE directive. The **MB-STD-RS232** provides long range data link at low/medium data rate for various industrial telemetry and data transfer applications.

Also this board can be used as a test board of the **STD-402 TR**.

Features

- CE compliance **STD-402** 434 MHz RF module on the board.
- RS232 interface with D-sub 9pins connector or Modular 6pin jack.
- Fixed frequency / Auto frequency setting selectable.
- Cross / Straight cable selection SW.

Applications

- Serial data transmission (RS232C communication)
- Telemeter (FA line, Sensor information)
- Wireless connection between PC and peripheral RS232 equipment

General Description

MB-STD-RS232 is designed to make it possible for the user to connect between RS232 equipments with the radio. **STD-402** 434 MHz narrow band radio module that complies with EN300220 is equipped on the board. 64 channels are pre-programmed in the module.

There are two frequency setting are available. **In fixed (manual) setting**, RF channel can be set on board switches. **In auto setting**, RF channel is set to vacant channel automatically.

Operation mode and communication set up (Ack, parity, data rate) can be selected by on board dip-switch. The **operation mode 1** is designed for two-way communication and the **operation mode 2** is designed for one-way communication (TX -> RX).

1:N communication is possible by using unique module ID number that designated to each RF module.

Specification

RF parameter

Communication mode	Half-Duplex
Frequency range	433.200 to 434.775 MHz
CH step	25 kHz
Number of CH	64 CH
CH setting	Fix / Auto (8Gr*8ch)
Modulation data speed	9600bps
Modulation	2FSK
Emission class	F1D
Transmission power	10 mW

Serial Interface

	Interface	RS-232C	
	Data format	Asynchronous communication (UART)	Da
speed of RS	1200/2400/4800/9600 bps		
	Flow control	RS / CS hardware control	
	Buffer	Transmission 2kB, Reception 2KB	
	Interface connector	D-Sub 9P / Modular 6P	

Other

Switches	Power, Frequency, Operation Mode, Cable (Cross/Straight)
LED indication	TX, RX, RSSI, LD, LE
Dimension	85*53*15mm
Supply Voltage	4.0 to 9V DC.

DESCRIPTION

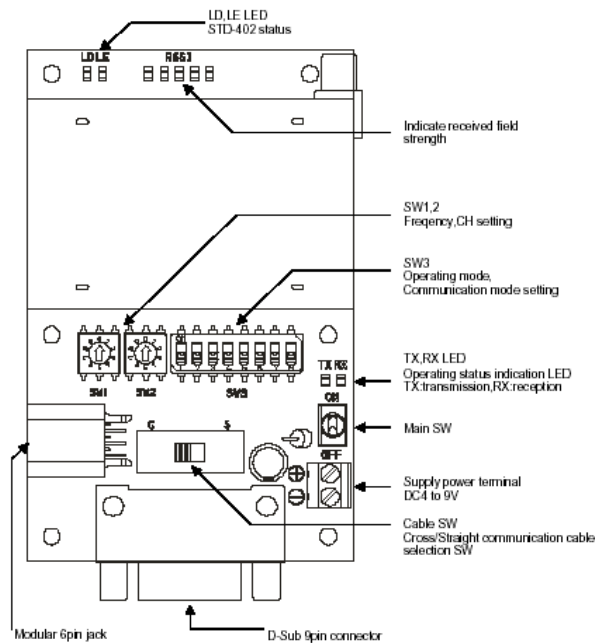


Figure 6.5: MB-STD-RS232 – CIRCUIT DESIGN

For more details about this board, refer to Annex A.

Special for STD-402 (Transceiver)



Figure 6.6: STD-402 Transceiver – CIRCUIT DESIGN

The **STD-402** transceiver is an UHF Narrow Band Multi channel Transceiver.

The UHF FM-Narrow Band semi-duplex radio data module **STD-402** equipped PLL controller in its robust metal housing. Unlike other transceiver, the **STD-402** is ready to transmit RF data without complicated controller board. The compact size and low power consumption of the **STD-402** make it ideal for battery operated applications where its

interference rejection and practical distance range are much better than similar RF modules based on Wide Band SAW – resonator frequency devices.

Most of RF setting are done by internal microcomputer, which allows the user to manipulate the module without professional knowledge of RF circuit.

Features

- European EN300 200 standard compliance.
- High technology into compact module for easy operation.
- Low voltage operation from 3.6 V DC.
- Low current consumption, ideal for battery operated applications.
- 9600bps data rate.
- Carrier sense output for Multi-Channel access operation.

Application

- Remote control system.
- Security systems.
- Bi-directional communication systems.
- Telemetry systems
- Handy terminal.

STD-402 characteristics

❖ Common

Communication form	Semi-duplex
Frequency range	433.200 MHz to 433.775 MHz.
Channel step	25 kHz.
Baud rate	9600bps max.
Supply voltage	3.6 – 12 V DC (Direct Mode).
Dimensions	53 × 35 × 12 mm.

❖ Transmitter

RF output power	9 mW ± 1mW.
Data input level	3.6 – 12V (Direct Mode).
Input signal	Digital
Spurious emission	< -60 dBm (< 1 GHz).

Supply current 36 mA.

❖ **Receiver**

Receiver type	Double superheterodyne PLL synthesizer.
Selectivity	± 4 kHz at -6 dB point.
Data output	Digital.

The **STD-402** transceiver has 3 mode operation guide :

1. Direct Mode Operation Guide (For more details about this mode, refer to Annex B)
2. Auto Mode Operation Guide. (For more details about this mode, refer to Annex C)
3. Auto Mode Operation Guide for CPU interface. (For more details about this mode, refer to Annex D)

Or the **MB-STD-RS232** equips **STD-402** transceiver module and performs packet communication using CPU interface mode of the transceiver.

10.6.2.2 The communication software on the embedded board computer

The code for this program is written in C. The program runs under the ReaL Time Operation System (RTOS) VxWorks on the embedded system in the alternative – LOTTE. The purpose of this program is to get the sensors data from one serial port and put it on another serial port which is connected to the transceiver.

The whole C program code is in Annex H.

The important thing in this program is the initialization of the serial interface communication, the following lines contain a description about this initialization :

➤ `id_com_dev = open (c_device, O_RDWR, 0);` (line of code: 17)

This command is used to open the serial port. When the serial port is not opened, this function returns -1 .

➤ `if (-1 == ioctl (id_com_dev, FIOBAUDRATE, speed))` (line of code: 24)

➤ `SerialControl = (CLOCAL | CREAD);` (line of code: 28)

➤ `Parity = PARENB ;` (line of code: 29)

➤ `CharacterSize = CS8;` (line of code: 30)

➤ `if (-1 == ioctl (id_com_dev, SIO_HW_OPTS_SET, SerialControl | Parity | CharacterSize))` (line of code: 31)

➤ `ioctlvalue = ioctl (id_com_dev, FIOSETOPTIONS, OPT_TANDEM &~ OPT_MON_TRAP);` (line of code: 36)

These commands initialize the parity, the stop bit, the data length and the handshaking mode.

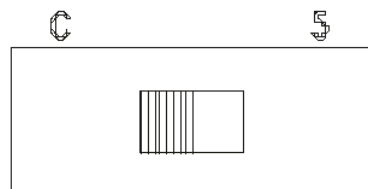
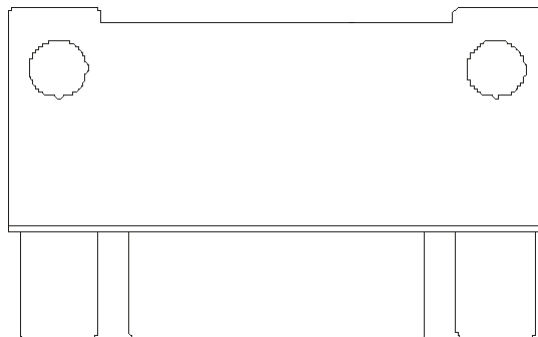
10.7 Component Tests

10.7.1 Transceiver test

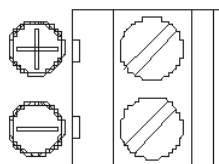
The MB-STD-RS232 board which equips the STD-402 transceiver module has stored unique module identification number in the radio module. When one unit is set up for master and other unit is for slave, slave modem unit operate with same ID as master unit.

The test of the transceiver is shown in following lines :

1. Connect the serial cable to the D-SUB 9pin connector.

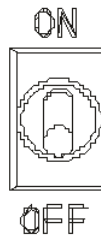
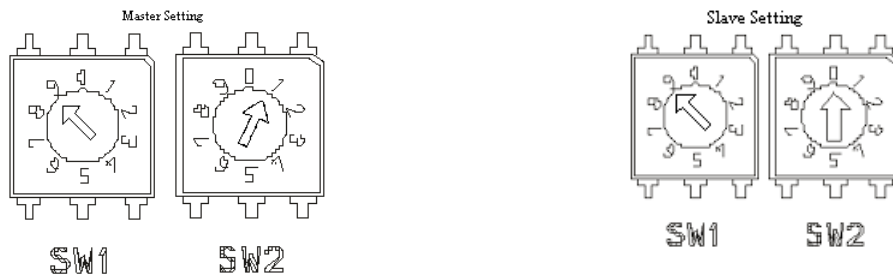


2. Set "Cable SW" (Cross / Straight) according to the cable.
3. Connect the supply voltage of the transceiver to 6V DC.



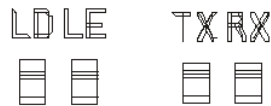
4. Select unit to be master and set SW1 to "9" and SW2 to "1". Select unit to be slave and set

SW1 to "9" and SW2 to "0".



5. Power ON the units.

6. When power of the master is turned on, TX, RX and LE LED turn ON and LD blinks.



Radio communication start and continue for about 10sec.

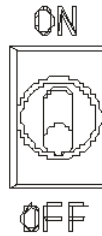
7. When power of slave unit is turned on, TX, RX LED turn ON and RSSI turn on when



signal from master is received. LD blink when group setting is completed.

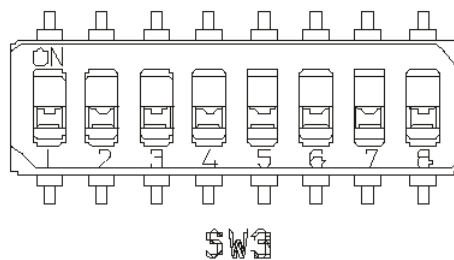
8. After slave unit receives unique module identification code stored in master units, radio communication can be performed with this code.

9. Power of the units.

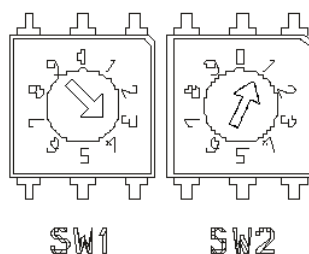


10. Setting the mode and the property of the communication by the SW switch.

- 1 : ON → Transmitter 1 : OFF → Receiver
- 2 : OFF → Mode 1 (Two way communication)
- 3 : ON → Setting prohibited.
- 4 : ON → Setting prohibited.
- 5 : ON → ACK response (Yes).
- 6 : ON → Parity Yes (Even).
- 7 : ON → Communication speed.
- 8 : ON → Communication speed. (9600bps).

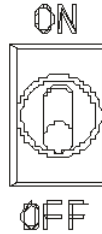


11. MB-STD-RS232 has 64 pre-programmed frequency channels. These frequencies are divided to 8 groups. Each group contains 10 frequencies. The group can be selected by SW2, and the value inside the group is selected by SW1. The frequency used to built the test is 433.975 MHz. To select this frequency, set the SW1 switch to 3 and the SW2

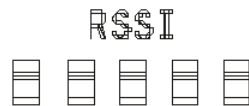


switch to 1. The master and the slave unit should be selected to the same frequency.

12. Power on the units.



13. MB-STD-RS232 is in RX mode at wait time (stand by), which means RX is turned ON at wait time. When the unit receives radio data from the other unit, the RSSI LED turn ON



and the unit will start outputting the data to RS232 port.

14. When the unit get data from PC through RS232C connector, the data is stored in internal buffer and then will be sent after the MB-STD-RS232 check that the carrier frequency to be



set is not used in air. The TX LED turned ON when the unit will transmit the data. The unit returns to RX mode when all data is gone.

10.7.2 User interface laboratory test with two PCs

As described in the implementation in chapter 6, the important things before testing the user interface program is to take care about the path of the files that the user interface used, the program run with the user interface, the initialization of the serial port (number of port, baud rate, parity, data length, stop bit, handshaking mode, input mode ...), the interval of the timer used to read data from the receive buffer. If all these commands are done in the correct way, an EXE file can be made and used.

Connect one of the transceiver to a PC that contains the user interface program and the other transceiver to an other PC that contains the program used to send data (UserInterface_SendData). Then run the two programs and click the Sending_Data button and the data will be displayed in the user interface program each 500 msec as below :

Figure 7.1: position & direction of the alternative – LOTTE

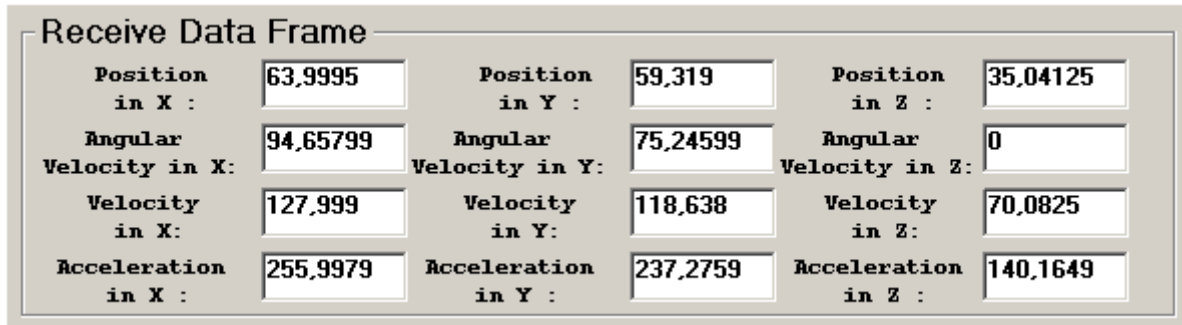


Figure 7.2: parameters of the alternative – LOTTE

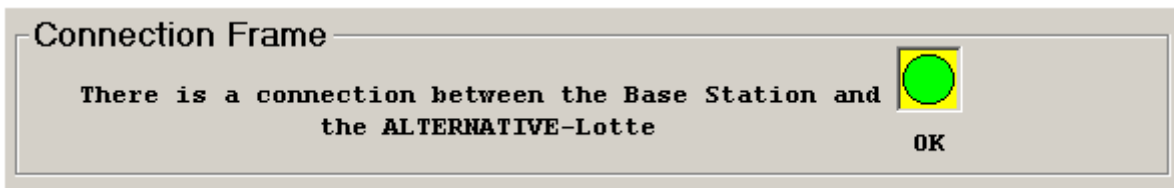


Figure 7.3: Status of the alternative – LOTTE

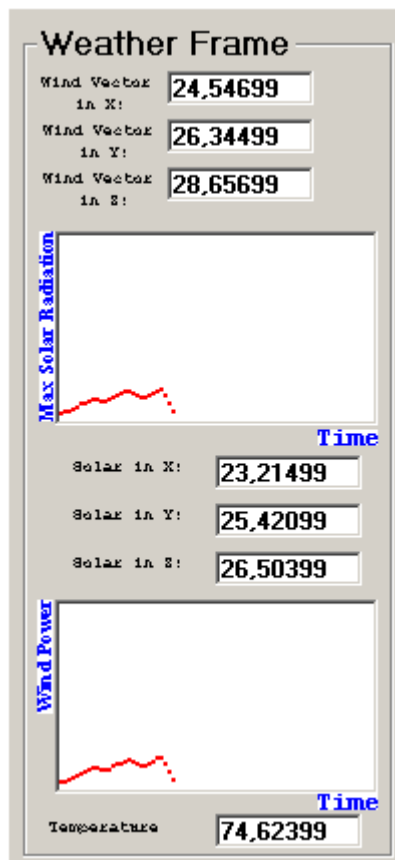
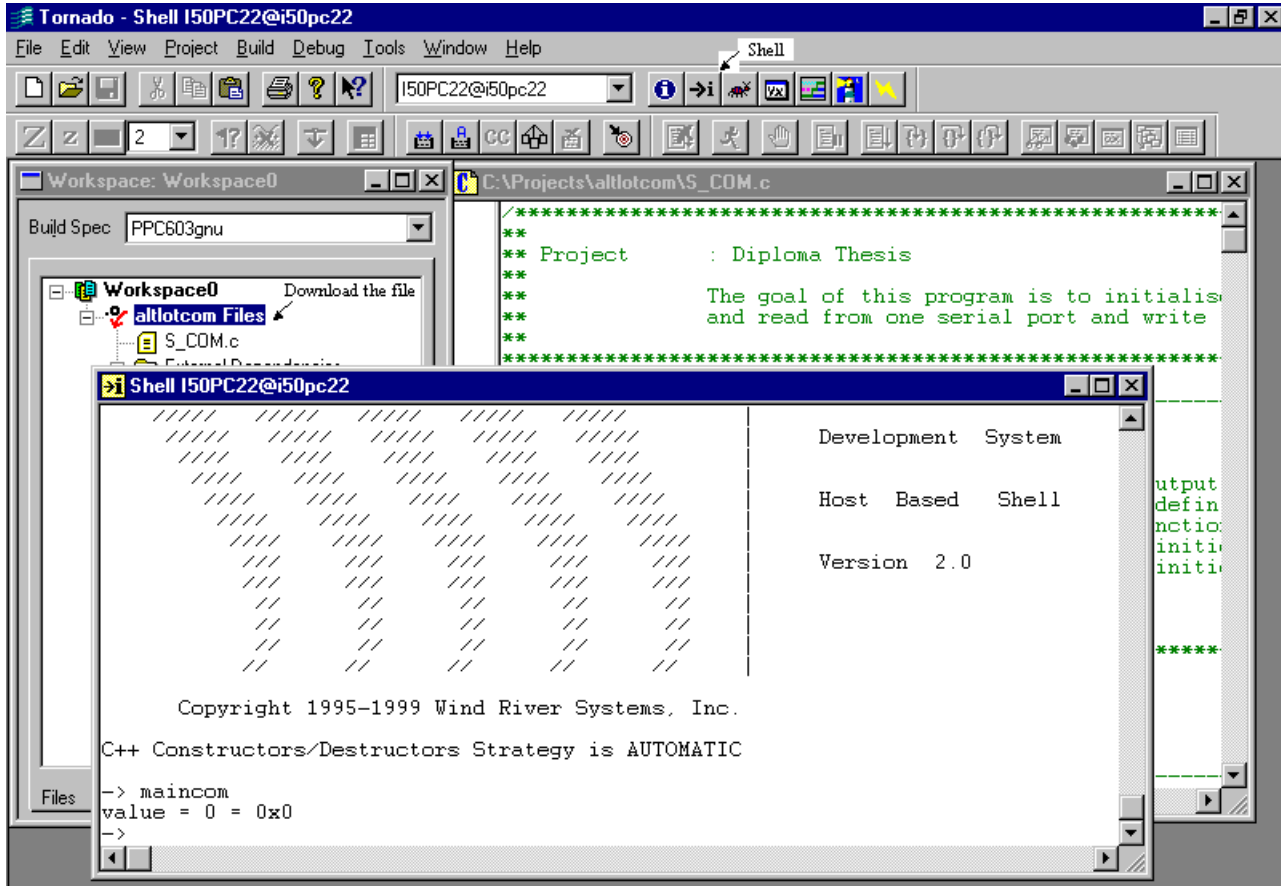


Figure 7.4: Weather parameters

As we see in the pictures above that the user interface test with the transceiver test work very well, and the data are received correctly and putted in their place.

10.7.3 Test of the user interface on the target system

To test the C program on the embedded system, run the TORNADO 2.0 then the following figure will be shown :



To open the connection between the TORNADO and the VxWorks machine, open the tools options, then select "Target server", then "PepVMP", then go to the list box to select the IP, after that the TORNADO inform you if the connection is done between it and the VxWorks machine.

So if the connection is ok, the user will be able to compile the C file, then open a shell to send commands to the VxWorks machine. Before running the C file on the VxWorks machine, you should download the C file on it, this thing can be done by select the project, right click on it and press Download "name of the project".

If all thing is done in the right way, the user will be able to type the name of the C program in the shell (In our project, the name of the C program is "maincom"), and the data is sent to the serial port of the VxWorks machine.

The whole code of the C test program used is shown in Annex G.

10.8 Annex A

Some trouble shooting

Phenomena	Cause of trouble
TX-LED blink	SW1, 2 setting error. Please check setting of SW1, 2.
RX_LED blink	Low voltage. Please check supply voltage
TX and RX LED blink by turns	Internal EEPROM is something wrong. Please contact us.
Both RX and RX LED turn ON And no transmission.	There would be interference at set frequency. Please change the frequency. Noise from PC might be the cause of the problem. Please try to leave the board from PCs and check it.

Frequency CH setting

There are two frequency settings are available.

- Fix setting (manual):
Frequency is set by switches (SW1 & SW2) on the board manually. RSSI-LED indication helps to set vacant channel.
- Auto setting:
MB-STD-RS232 search radio carrier to find vacant channel before starting radio communication and set to vacant channel automatically. For more details about the frequency table, refer to the data sheet (page 6,7)

Initial Setting (Master / Slave)

SW1	SW2	Function
9	0	STD-402 initial setting / set to Slave
9	1	STD-402 initial setting / set to Master
9	9	Setting mode (Lf, Ack, Character strings, etc..) and TEST

Operation Mode Setting SW

NO	Description		SW3 setting							
			ON				OFF			
1	Operation Mode		Transmitter				Receiver			
2			Mode 2 (One communication)				Mode 1 (Two communication)			
3	(Reserve)		Setting Prohibited				Off			
4	(Reserve)		Setting Prohibited				Off			
5	U	ACK response	Yes				No			
6	A	Parity	Yes (Even)				No			
7	R	Communication	ON	9600	OFF	4800	ON	2400	OFF	1200
8	T	Speed (bps)	ON		ON		OFF		OFF	

Stop bit at UART Communication

Normally, Stop bit is selected one of 1 / 1.5 / 2 bit when COM port of PC equipment is set-up. When continuous data are outputted, data comes after stop bit. However there is a case that there is more time space than specified stop bit length because of asynchronous communication.

The **MB-STD-RS232** receives the data from PC correctly if stop bit is 1 bit or more. Stop bit of data from the **MB-STD-RS232** to PC is fixed to 2 bit therefore communication with PC can be made regardless of stop bit (1, 1.5, 2 bit).

Jumper Setting

There are 6 jumper settings on the board. Changing J4 and J5 can change LED-working condition to reduce the consumption current.

J1, J2, J3 : Module set up **STD-402** 434 MHz setting must be Short, Open, Open.

J4: LED operation of TX, RX, RSSI LED Use: Short Non Use (off): Open

J5: LED operation of LD, LE Use: Short Non Use(off): Open

J6: CPU reset signal: this must be short

Communication Interface

The board equips D-SUB 9pin and Modular 6pin as communication interface [8].



Figure A.1: Serial Connector

Signal	I/O	Modular 6P	D-Sub 9p (*)		Remark
RD (RX)	I	1	2	3	Input terminal / from PC
RS (RTS)	O	2	7	8	Hi when the equipment become reception possible. Lo when internal Buffer is full
SD (TX)	I	3	3	2	Data output terminal / from PC
CS (CTS)	O	4	8	7	Hi when SD terminal send data Lo when SD terminal do not send data
GND (SG)		5	5		GND
DC (+)		6	-		Supply power (DC 4 to 9V) is possible From Modular 6pin.

Communication Cable

Cross cable which mainly connect between PCs and straight cable which connect PCs and peripheral RS 232 equipment are available as RS 232 standard cable. Please set “Cable

SW” (Cross / Straight) according to the cable. For more details ,refer to the Data Sheet (page 11,12).

Operation Mode

MB-STD-RS232 has two operation modes as below :

- Operation Mode 1: Two-way (Semi-duplex) communication
- Operation Mode 2: One-way communication

➤ Operation Mode 1: Two-way (Semi-duplex) communication

This mode is useful for an application of two-way communication with the Ack/Nck Response between PC and RS232 equipment, and an application that returns the data from RS232 equipment to PC according to the command from PC **MB-STD-RS232** is in RX at wait time (stand by). When the unit receives radio data from the other unit, the unit will start outputting the data to RS232 port. When **MB-STD-RS232** get the data from PC through RS232C connector, the data is stored in internal buffer and then will be sent after **the MB-STD-RS232** check that the carrier frequency to be set is not used on air. The unit returns to RX when all data in buffer is gone.

1:N communication is possible by implementing the group setting. In this case, the application software have to be programmed to consider that identification number of RS232 equipment have to be included in transmission command, and only the equipment which has requested ID number returns the data when its command is received. In this operation mode, Only Fix CH setting can be used.

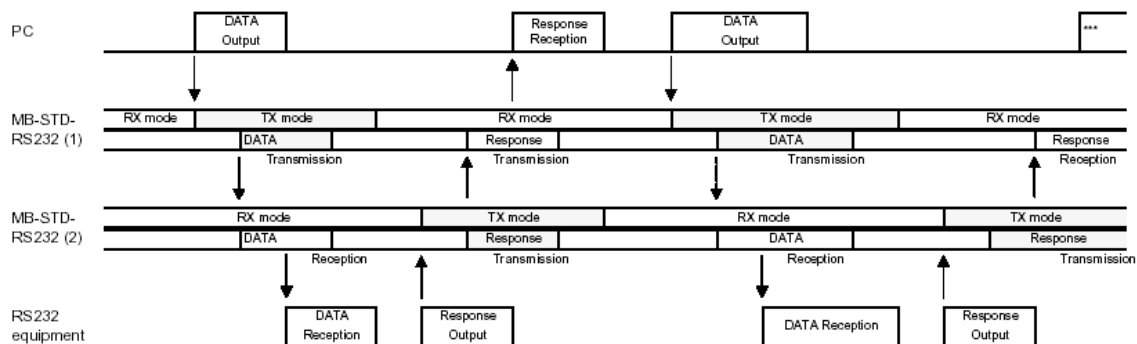


Figure A.2: Two-way, Semi-Duplex Communication

➤ **Operation Mode 2: One-way communication**

This mode is useful for an application of one-way communication from PC to measurement equipment. TX (transmitter) / RX (receiver) setting can be done with dip switch on the **MB-STD-RS232**. TX unit transmit the signal at selected channel regardless that data is in buffer or not. RX unit outputs the data to RS232 port when radio signal is received. (RX unit does not transmit even data is imputed to RS232 port).

1:N communication is possible by implementing the group setting. 1 * TX unit to N * RX units communication is possible.

In this operation mode, Either Fix channel setting or Auto channel setting is used.

In Auto setting, TX unit perform automatic RF channel search and transmit the data in the vacant RF channel. RX unit perform channel scan to detect transmission signal and set to the RF channel.

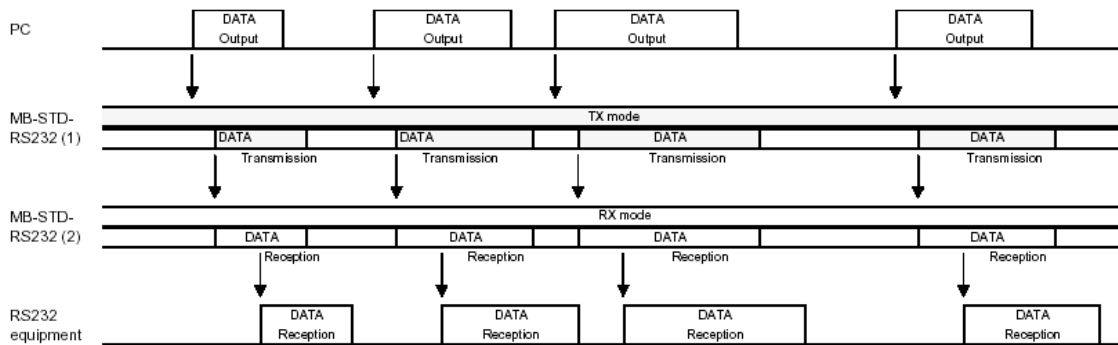


Figure A.3: One-way Communication

Restriction on data communication

The **MB-STD-RS232** equips **STD-402** transceiver module and performs packet communication using CPU interface mode (data length 63bit) of the transceiver.

Serial data or command stream from PC or other equipment connected to the **MB-STD-RS232** is pocketed and sent to a receiver. When these data are outputted from the unit, the time space appear as shown below. The time space (delay) is 300msec or more when operation mode 2 (One-way communication) is used.

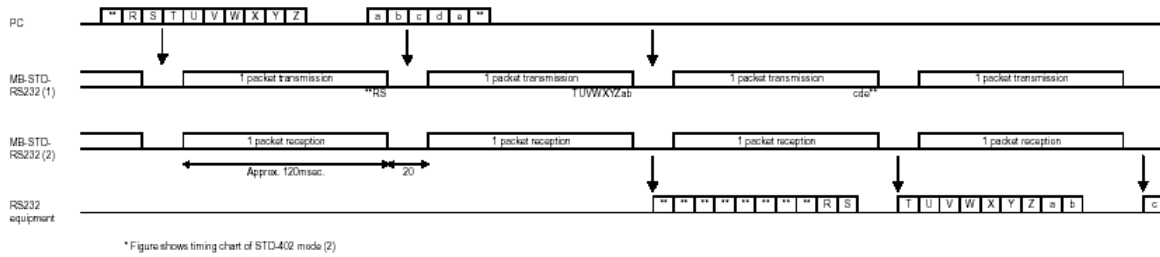


Figure A.4: Data Communication

ACK Auto Response Function

Depending on RS232 equipment, but some equipment stop or become error and repeat transmitting same data if Ack/Nck command is not returned as response signal within several 10msec.

This is not problem in wired communication because PC can return Ack/Nck command immediately after receiving data from RS232 equipment. Wireless communication with the **MB-STD-RS232** have a few second time-lag for the response. This function is given to avoid error caused by such situation. It is realized to return Ack (or specific response string) response return.

1. **MB-STD-RS232** operates as that the data from RS232 equipment is suppose to carry delimiter symbol (Etx, Lf etc...) at end.
2. When Ack/Nck character is contained in output from RS232 equipment, **MB-STD-RS232** judges that the Ack/Nck character is the response from PC and when it is not contained the output is recognized as data stream (i.e. measurement data) for request

data command from PC. The **MB-STD-RS232** sends Ack response only if this data

stream (without Ack/Nck) is recognized.

(If there is data in TX buffer, the Ack will be sent after that).

3. Delimiter, Ack/Nck and character string to recognize these operation can be changed by user.

Example

Delimiter		"Lf" (0AH)
Response detect code	1	"Ack" (06H)
	2	"Nck" (15H)

Auto response character string "Ack CR Lf" (06 0D 0AH) Max. 16 characters

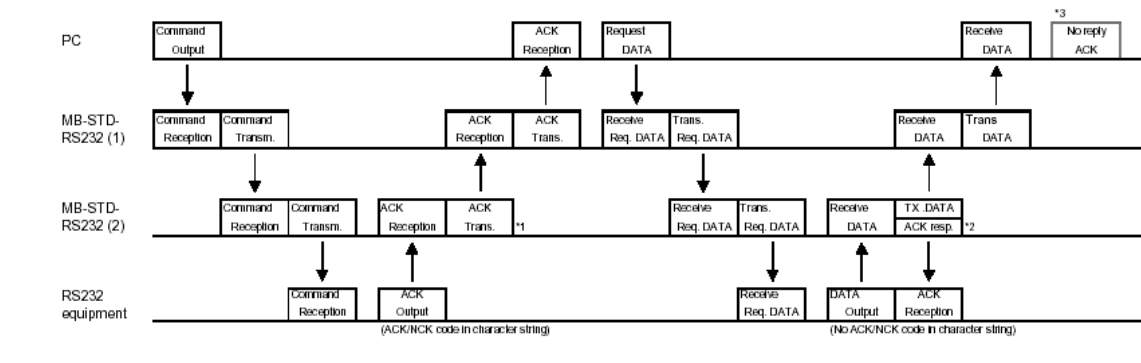


Figure A.5: ACK Auto Response

Group Setting

The **MB-STD-RS232** has stored unique module identification number in the radio module. When one unit is set up for master and other unit(s) is for slave, slave modem unit operate with same ID as master unit.

If this setting (group setting) have not finished, radio communication between the **MB-STD-RS232** is not possible even with same frequency.

➤ **Setting Procedure**

1. Select unit to be master and set SW1 to "9" and SW2 to "1". Select unit(s) to be slave and set SW1 to "9" and SW2 to "0". Power on the units.
2. When power of the master unit is turned on, TX, RX and LE LED turn ON and LD LED blinks. Radio transmission start and continue for about 10sec.
3. When power of slave unit(s) is turned on, TX, RX LED turn ON and RSSI turn ON when signal from master is received. LD blink when group setting is completed.

After slave unit receives unique module identification code stored in master units, radio communication can be performed with this code.

Note: This setting is to set identification code for radio communication. Original unique module identification code of slave unit itself remains unchanged. When the unit was used for slave is set to

master and perform communication, identification number is different from former communication. Therefore interface does not occur.

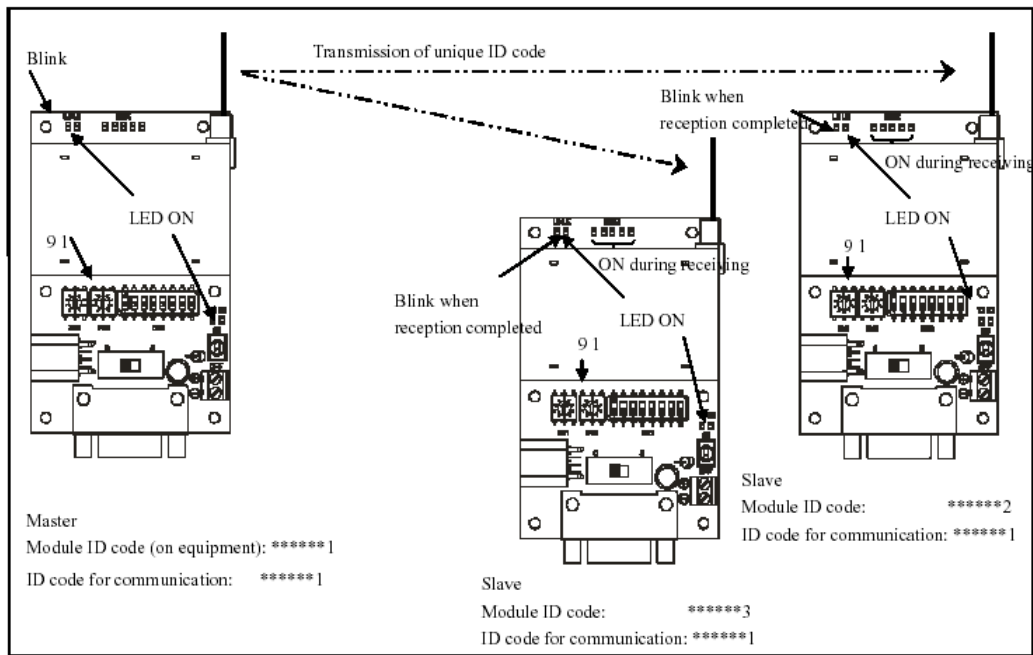


Figure A.6: Master & Slave Units

Ack Auto Response Setting

Delimiter, Ack/Nck code and auto response character string (max. 16 characters) for recognition of Ack auto response can be changed from RS232 port using software like Windows Hyper Terminal.

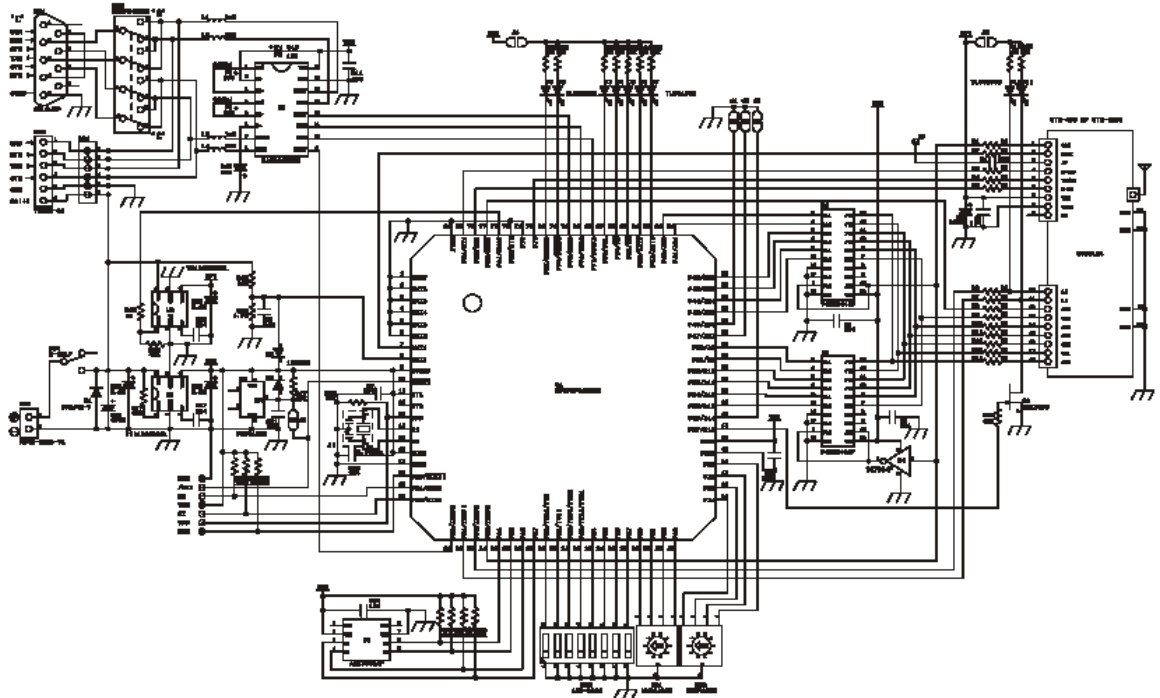
➤ Procedure

1. Connect **MB-STD-RS232** and PC with RS232 cable. Set SW1 and SW2 on the **MB-STD-RS232** to "9", "9", then power ON.
2. Start up the Hyper Terminal and set communication condition as below.
4800bps, Bit = 7, Parity = Odd, Stop Bit = 1, Flow control = Hard ware.
3. At first, type "/A CrLf" and return. Check if the **MB-STD-RS232** modern return the product name and program version.
4. Set up with reference to the following table.

Command	Transmission format	Correct Response	Description
---------	---------------------	------------------	-------------

		Data (example)	
"/A"	"/A CrLf"	"/A MB-STD-RS232, 001.000 CrLf"	Product name, Program ver.
"/I"	"/I 06, 0D, 0A CrLf"	"Ok CrLf"	Set and check of Ack response character string (HEX) Max. 12 character (prohibit to include ASCII code "00")
	"/I CrLf"	"/I 06, 0D, 0A CrLf"	
"/K"	"/K 06 CrLf"	"OK CrLf"	Set and check of Ack response detect code 1 (HEX). Example : 06H = Ack
	"/K CrLf"	"/K 06 CrLf"	
"/L"	"/L 15 CrLf"	"/L OK CrLf"	Set and check of Ack response detect code 2 (HEX). Example : 15H = Nck
	"/L CrLf"	"/L 15 CrLf"	
"/M"	"/M 0A CrLf"	"OK CrLf"	Set and check of delimiter (HEX) Example : 0AH = Lf
	"/M CrLf"	"/M 0A CrLf"	

CIRCUIT DIAGRAM



Circuit diagram for the **MB-STD-RS232**

10.9 Annex B

◆ STD-402 [Direct Mode Operation Guide]

General Description

The **STD-402** is a short range device stipulated by CEPT/ERC recommendation 70-03 and is a radio module complying with ETSI standard EN300-220-1 with a transceiver built into its compact case.

The built-in PLL synthesizer circuit enables both transmission and receipt through 64 pre-set channel frequencies between the 433 to 434 MHz bands.

The transmit mode / receive mode settings and the frequency channel settings can be made easily with dip switches or jumpers without the use of an external microcomputer.

As the system operates on low voltage and low current consumption in consideration of mobile communications, battery operation is also possible.

The FSK modulation enables transmission at a maximum of 9,600bps. Also, when multiple channels are to be used simultaneously or if high communication standards are required, the

MSK modulation enables transmission at a maximum of 2,400bps. However, it is necessary to install an MSK modem IC externally for this purpose.

The radio module has been designed with high levels of reliability cultivated through long-term results, and this provides excellent levels of selectivity receiving sensitivity and radio interference capabilities.

Features

- Comply with EN 300 220
- Compact size (53mm * 35mm * 12mm) with built-in transceiving functions (**STD-402**)
- Baud rate of maximum 9,600bps
- Simple channel setting with switches
- Continual "0" or "1" data transmission for a maximum of 20msec.
- High-level endurance capabilities against the effects of antenna reflection and surrounding equipment
- Low-voltage operations : 3.6 V to 12 V
- Low-current consumption : 36mA (transmit mode), 26mA (receive mode)

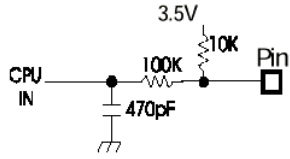
Applications

- ❖ Telecommand and Telecontrol
- ❖ Telemetry
- ❖ Alarms
- ❖ Data terminals

Pin description

Number	Pin Name	I/O	Description	Equivalent circuit
1	CAR (STATUS)	O	Carrier sense output of the receiver. The RSSI signal (receiving level) will become "H" when it exceeds the threshold.	
2	RSS	O	The receiving level output of the receiver. The strength of the level is converted to the current voltage.	
3	AF	O	The AF output of the receiver section.	
4	DO	O	The data output for the receiver section. The port uses FET output, the "H" level is Vcc.	
5	T/R	I	TX(transmit mode) / RX (receive mode) setting. The port is pulled up with resistor. TX mode: L (GND) RX mode: H (Open)	

6	DI	I	Data input for the transmission section. The port uses tri-state input, the digital "H" level is Vcc and the digital "L" is GND.	
7	VCC	-	The power supply terminal. Operates on 3.5V to 12V, but the same Vcc level as the surrounding circuits must be used.	
8	GND	-	The ground. Extend the pattern over the widest area possible on the printed circuit board.	
9	LD	O	<ul style="list-style-type: none"> The LOAD signal output for External parallel -> serial register The RESET signal is output with the receive mode. The port uses FET buffer output, the "H" level is Vcc. 	
10	LE	I/O	<ul style="list-style-type: none"> The LATCH ENABLE signal output for external serial -> Parallel register. Setting mode switch input. 	
11	RDY (CLK)	O	The READY signal output. RDY is active when at "L". The following conditions apply when RDY is "L": (1) Initial status. (2) CAR is "H" Call name being transmitted.	

12	CH5	I	<ul style="list-style-type: none"> • Set the various conditions when in the setting mode. • Sets the ID address with normal operations. 63 address types between 1 and 63 are available • Pulls up each port. 	
13	CH4			
14	CH3			
15	CH2			
16	CH1			
17	CH0			

Electrical characteristics

- *Common characteristics*

Item	Rating	Conditions/remarks
Communication form	Semi-duplex	
Modulation	F1D	FSK
Oscillation system	PLL controlled VCO	
Frequency range	433.200 – 434.775 MHz	
Channel step	25 kHz	
Number of RF channel	64 channels	
Baud rate	4800, 9600bps	FSK
Modulation polarity	Positive	
Demodulation polarity	Positive	
Antenna impedance	50 Ohm	
1st IF	21.7 MHz	
2 nd IF	450 kHz	
Range	200m or more	F1D 9600bps
Operation temperature	-10 to 55°C	
Operation power voltage	3.6 – 5V	
Supply current	36mA	TX mode

	26mA	RX mode
Dimensions	53*35*12mm	
Weight	34g	

- *Transmission section characteristics*

Item	Rating	Conditions
Transmitter type	PLL synthesizer	
RF output power	9.0mW ± 1.0mW	10mW
Frequency stability	± 4ppm	-10 to + 55°C
Spurious emission	< -60dBm	< 1GHz
	< -50dBm	> 1GHz
Deviation	±1.9 to 2.1 kHz	*1
S/N ratio	> 25dB	*1
Adjacent channel power	> 40dB	Spectrum analyzer act. *1
Carrier sense level	-107 dBm	Fixed
Transmitter start-up time	< 30msec	PLL data setting
Channel switching time	< 15msec	25KHz
	< 30msec	100KHz

*1: 9,600bps, 511 bit (Pseudo Noise)

- *Receiving section characteristics*

Item	Rating	Conditions
Receiver mode	Double super heterodyn	
Sensitivity	< -117 dBm	25°C, *2

Spurious response	> 45dB	*3
Selectivity	> 45dB	*3
Local frequency stability	± 4ppm	-10 to +55°C
Radiation from local oscillator	< -65 dBm	< 1GHz
	< -60 dBm	> 1GHz
Output level	350m Vp-p	100Kohms terminate *4
Carrier sense level	-113 dBm	Fixed
Carrier sense response time	< 30msec	PLL data setting
Channel switching time	< 15msec	25KHz
	< 25msec	100KHz
Bit error rate	1×10^{-2}	Less than -110dBm
	1×10^{-4}	Less than -107dBm

*2: AF = 1KHz, fmod = 2KHz, CCITT filter ON

*3: Jamming waves AF = 400Hz, fmod = 40%

*4: Dev:= 2Khz, AF = 1KHz

*5: 256bit / 4800bps

Modes

The **STD-402** can be controlled manually with a simple set of external switches and jumpers, or with microcomputer controller.

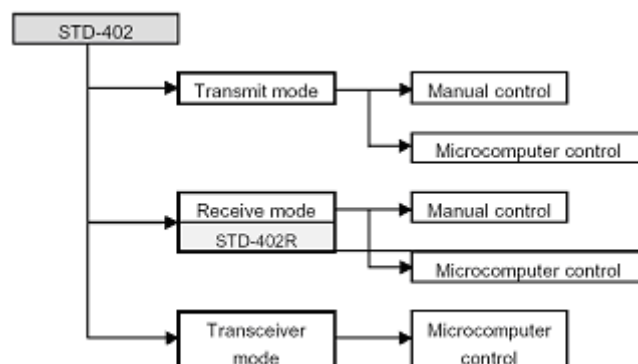


Figure B.1: Models & Modes

Modulation mode

- The maximum baud rate is 9,600bps with FSK. The data format may be decided by the user.
- The MSK modulation is recommended when stable operations are required during the use of multiple channels within the same area. However, the MSK mode requires an external MSK modem IC (MSM6882 manufactured by OKI or an equivalent model). Restrictions in the performance of this IC mean that the maximum baud rate is 2,400bps.

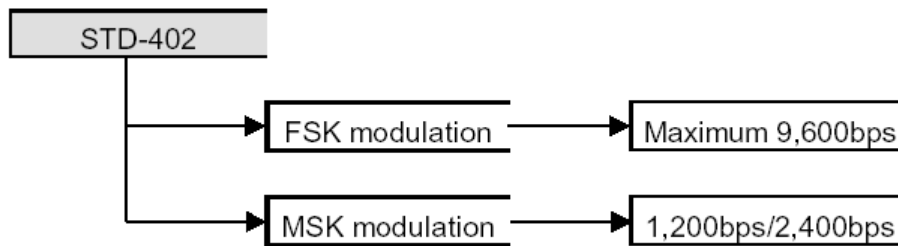


Figure B.2: Modulation Mode

For more details about the channel table and the Manual Mode, refer to Data Sheet (page 2...18).

Transceiver mode with microcomputer

- Connection method
 - More complex settings are made available when the **STD-402** modes and channels are controlled by microcomputer.
 - By developing microcomputer software that monitors the **STD-402** READY ports, it is possible to avoid channels already occupied by other radios in the same way as MCA (Multi-Channel Access) and automatically set up empty channels.

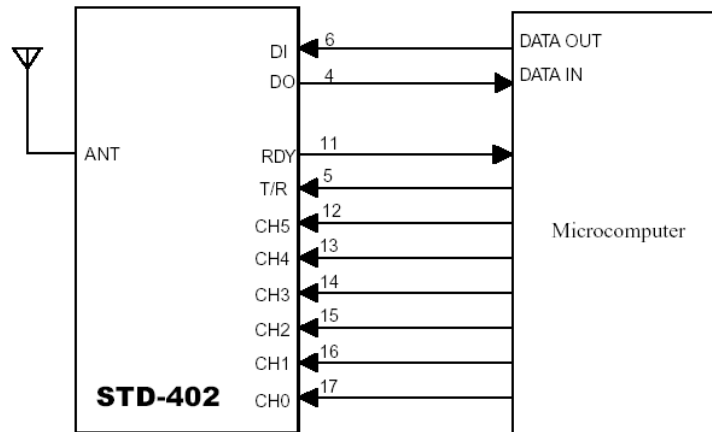


Figure B.3: Connection Method

Timing for transmit mode setting

- The flow chart and timing chart for transmit mode setting when under microcomputer control are provided below.

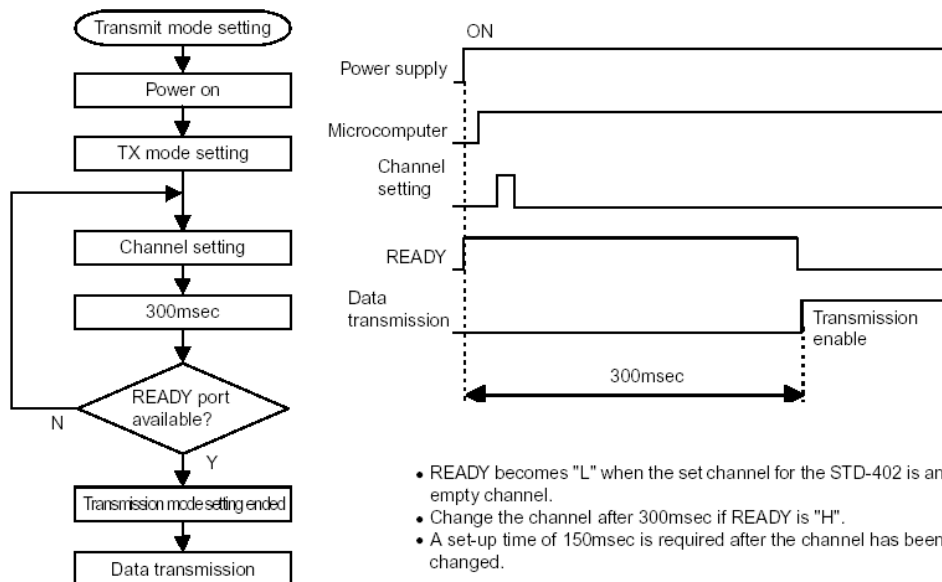


Figure B.4: Timing for transmit mode setting

Timing for receiving mode setting

- The flow chart and timing chart for receive mode setting when under microcomputer control are provided below.

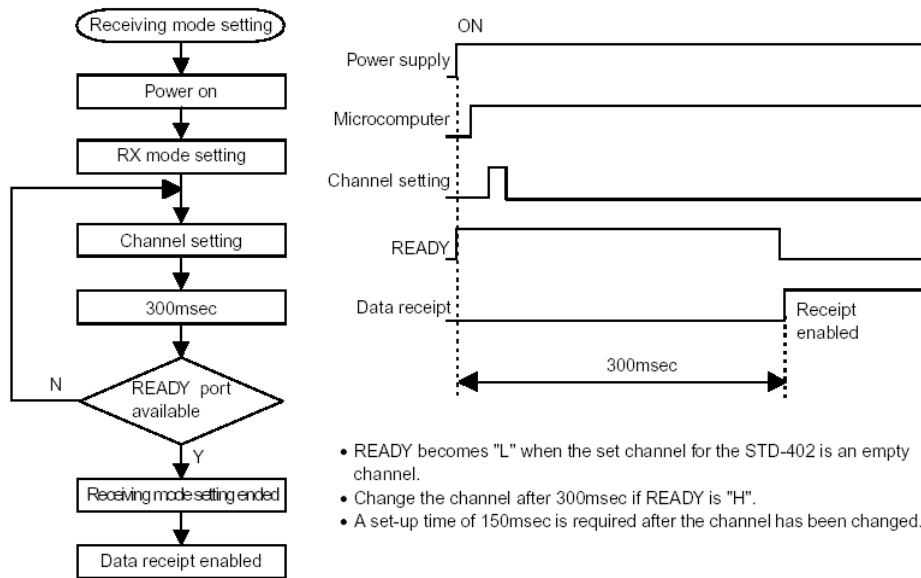


Figure B.5: Timing for receiving Mode

MSK Modulation mode

- Modem IC
 - The MSM6882 MSK modem IC manufactured by OKI is recommended for the MSK (Minimum Shift Keying) modulation mode.
 - Modulation MSK waves are emitted by loading transmission data into the encoder.
 - The decoder converts MSK waves into receiving data.
 - Examples for the application of the MSM6882 terminal function, the transmission mode and the receiving mode are provided in the Data

Sheet. For more details, refer to STD-402 Direct Mode Operation Guide (page 21, 22, 23)

Cautions

- Power supply
- The operating voltage range for the **STD-402** is between 3.6V and 12.0V. A voltage exceeding the maximum of 12.0V will result in damage to the device.
- Ensure that an open drain or open collector port is connected when the TX/RX, CH0 to CH5 and external circuits are connected together.

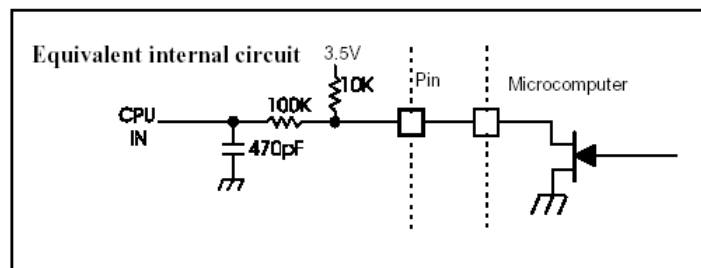


Figure B.6: Power Supply

- Reverse connection protection circuit
- ❖ When using low-voltage with battery operations

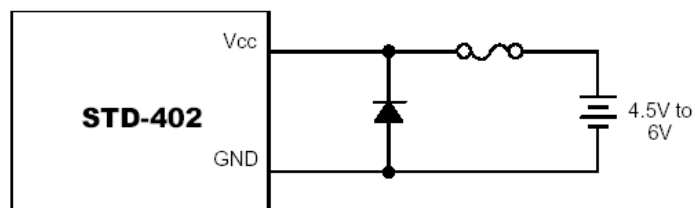


Figure B.7: Protection Circuit

- As the maximum battery supply current will flow into the diode

when the battery is connected in the reverse position, much consideration must be given to the PC (heat loss) in the diode.

- Although this method is the simplest, long-term heat emission may result in the outbreak of fire. Take extreme caution when handling this.

❖ When using high-voltage mains power for stable operations

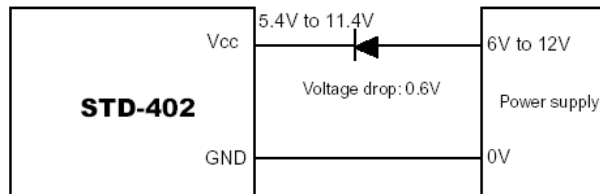


Figure B.8: High Power

- Although this method will not result in the buildup of heat in the diode, it will result in a lowering of the forward voltage in the diode and the efficiency rate of the mains power will be lower with batteries and other low voltages.
- The forward voltage will differ depending on the type of the diode, but the general rectification is approximately 0.6 to 0.7V.

Antenna

❖ Antenna and radial

- the **STD-402** antenna is approximately 17cm with a one quarter wavelength in the 434MHz band. The unit's metal case not only acts as the ground but is also known as the radial to radiate the electronic radio waves from the antenna and radial. The phase is inverted with the radial and antenna.
- Increase the module's ground pattern as much as possible. The electronic wave radiating power is strongest at the tip of the antenna, and when brought close to the unit's case, or radial, will work to cancel each other out, causing dramatic effect to the arrival distance.

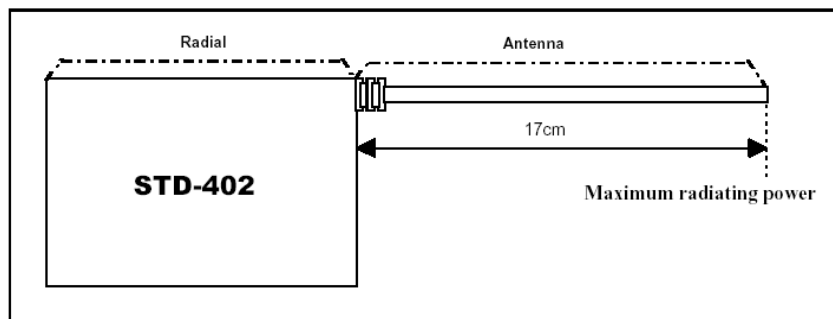
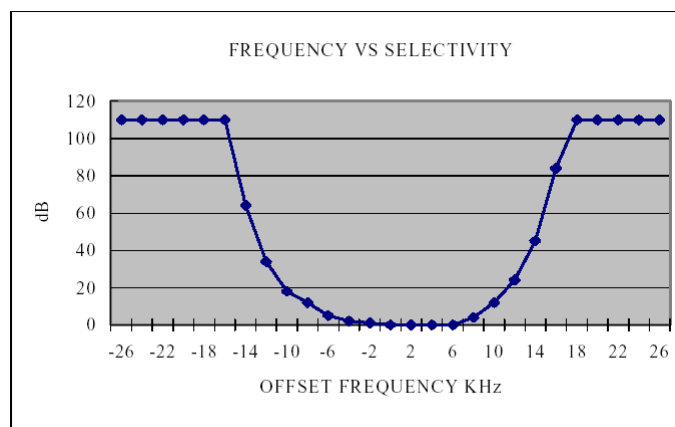
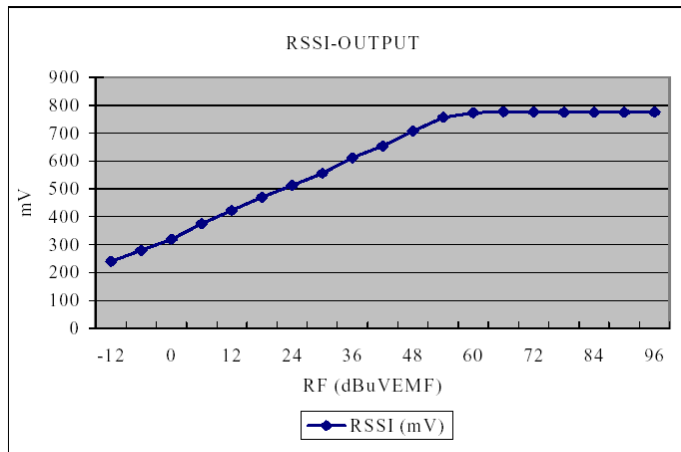
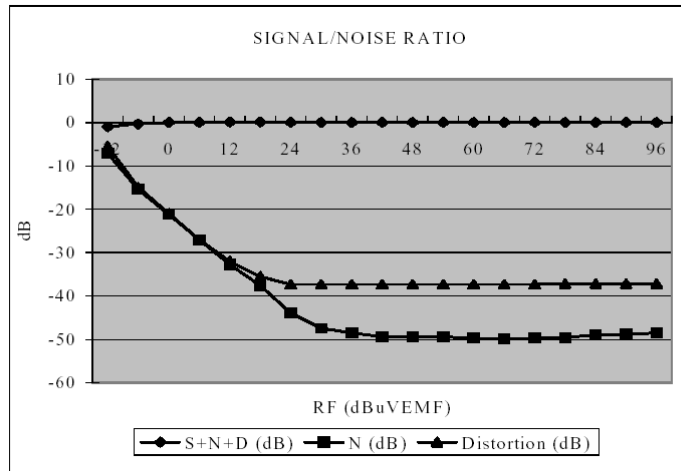


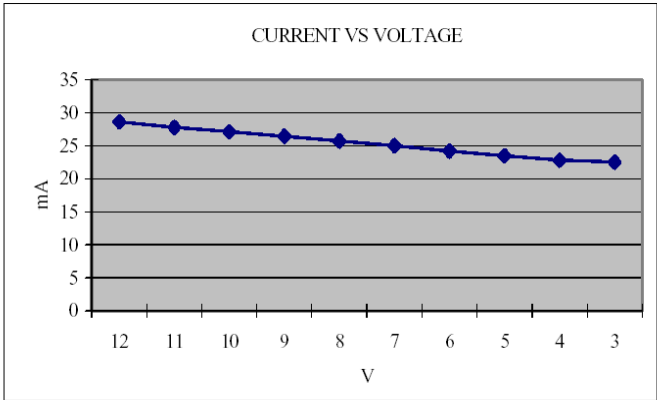
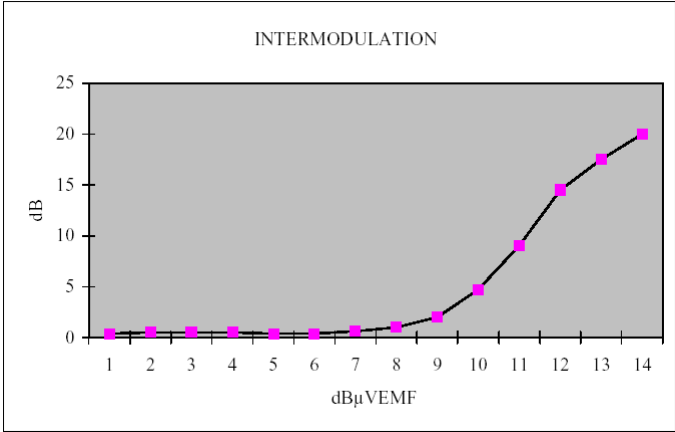
Figure B.9: Antenna

❖ Incorporating into equipment

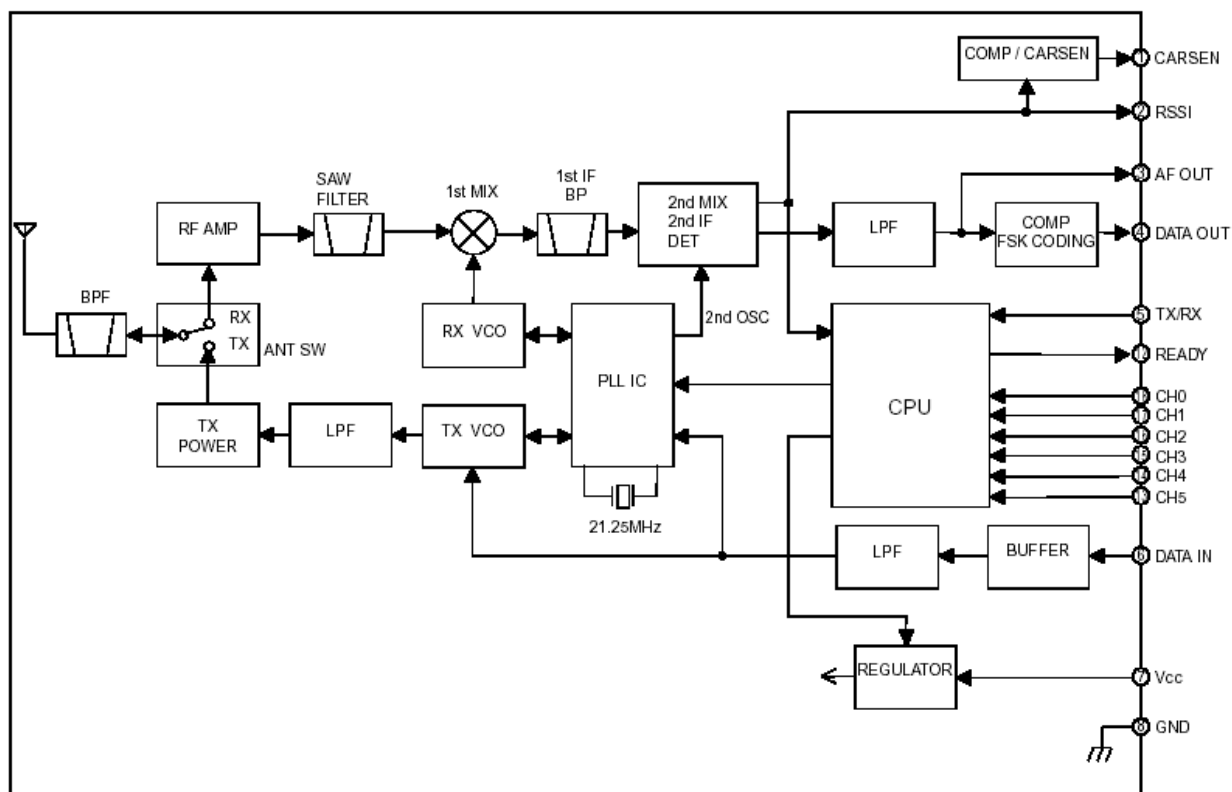
- A direct line is ideal for the antenna, but it should be separated from the case as much as possible when space for incorporating it into the equipment is limited.

Measured data





10.9.1 Block diagram of the STD-402 transceiver in Direct Mode



10.10 Annex C

◆ STD-402TR [Auto Mode Operation Guide]

10.10.1 General

The **STD-402** auto mode is a operation mode that is equipped with automatic link function and an encoding/decoding function. Automatic link searches vacant channels in order to use forty six channel frequencies without hindrance or interference. The encoding/decoding function establishes an interface between the data I/O circuit and radio assembly and executes the necessary communication protocols.

The transmitter's input circuit and the receiver's output circuit can easily be connected together with a general-purpose IC.

Note: The **STD-402** auto mode supports only one-way communication (uni-directional) transmissions (transmitter -> receiver) but support for simplex communications (bi-directional) is planned for the future.

10.10.2 Features

- The automatic link function automatically connects to channels that do not cause interference.
- Simple I/O circuits with the built-in encoder/decoder.
- The I/O connections can be expanded to a maximum of 63 bytes
- The built-in micro-computer greatly reduces the cost of new development.
- Perfect for combining with other equipment.

10.10.3 Application Examples

➤ One-way system

❖ Tele-control

- For controlling cranes, concrete pump vehicles, golf carts, remote opening/closing for various purposes, traffic lights for road works, etc...

Support for the following applications is planned for the future.

➤ Simplex systems

❖ Data transmission

- Handy terminals, bar-code readers

❖ Security

- Transmission of anti-theft alarms, immobilizes for cash delivery trucks, notification of customers entering retail shops, etc.....

❖ Telemeter

- Water level monitoring for canals and dams, etc..., monitoring of various alarms, etc... .

10.10.4 Configuration

- The auto mode automates numerous functions with the use of the built-in micro-computer.

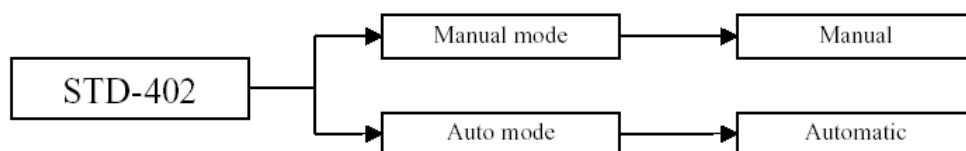


Figure C.1: Configuration

- The features of both modes are shown in the table below

MODE	Channel	Encoder/Decoder
Manual mode	Manual	External
Auto mode	Automatic	Internal

Auto mode features	(1) Automatic link (2) Programmable encoder/decoder
---------------------------	--

Figure C.2: Mode

Explanation for internal processes (1) [Automatic channel search]

- the automatic channel search is a system that detects the carrier level of the channel in use to search for vacant channels in order to avoid interference with other radios. This is controlled automatically by the built-in micro-computer.
- The channel search is executed by the module from the pre-determined starting channel.
- For example, if the channel search was executed from [0ch], the procedure would be as follows :
 1. The **STD-402** is set in the receiving mode.
 2. Set as channel 0 (433.200MHz)
 3. The carrier level is detect and compared with the standard level
 4. If the channel is occupied, the frequency is increased by two channels (50KHz).
 5. If the channel is vacant, the system is switched across to the transmission mode.
 6. The vacant channel is set.
 7. Transmission is started.

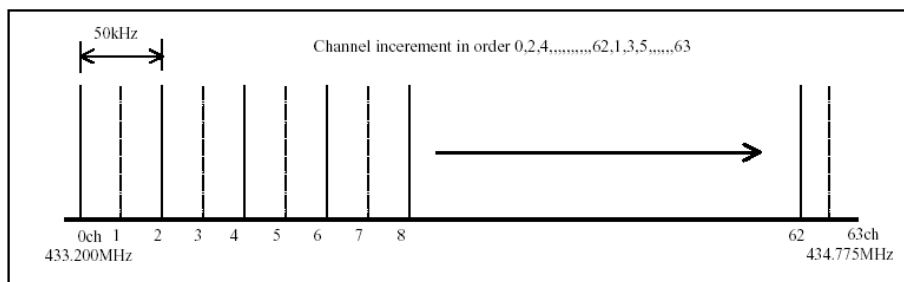
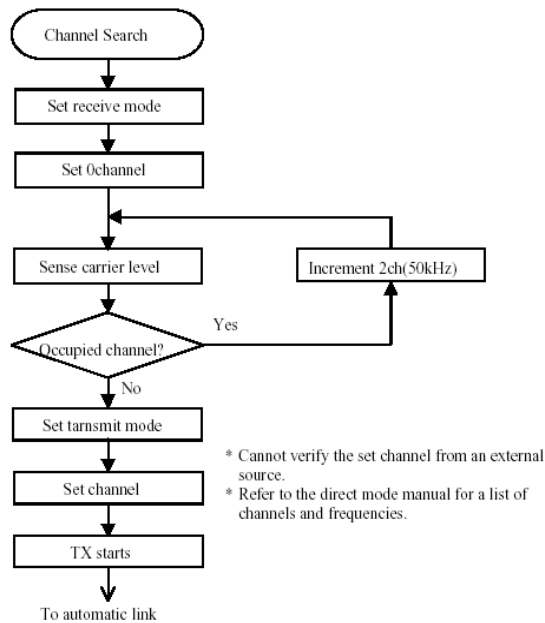


Figure C.3: Automatic Channel Search

Explanation for internal processes (2) [Automatic link]

- This is controlled automatically by the built-in micro-computer in the same way as the automatic channel search.
- All modules are started from [0ch] in the case of the automatic link.
- The channel search is executed by the module from the pre-determined starting channel.
- For example, if the channel search was executed from [0ch], the procedure would be as follows :
 1. The transmitter set for the vacant channel commences transmission.
 2. The receiver is set at channel 0 (433.200MHz).
 3. The carrier level is detected and compared with the reference level.

- * the reference level is different for the transmitter and receiver.
4. If it is below the standard level, the frequency is increased by one channel (25KHz).
 - * Increased by two channels for transmitter, but only one channel for the receiver.
 5. The ID data is received and compared with the ID register.
 6. Increased by one channel (25KHz) if the ID number matches.
 7. Switched across to the communication mode if the ID number matches.

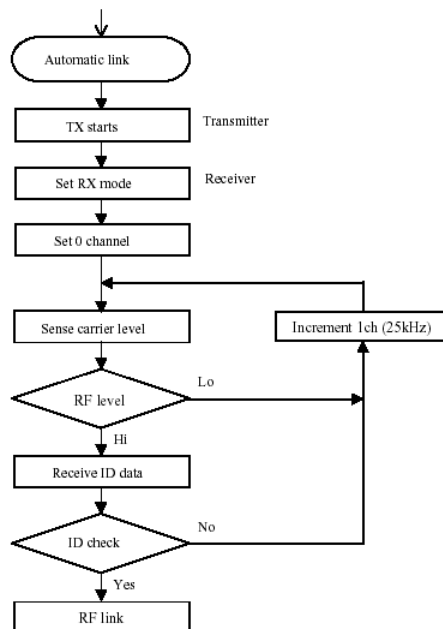


Figure C.4: Automatic Link

Explanation for internal processes (3) [ID number registration]

- It is necessary to register an ID number for the transmitter and receiver in order to enable the **STD-402** to establish an automatic link between two devices.
- If the transmission is performed on the same frequency from another radio or an **STD-402** with a different ID number, the receiver will

continue to search the channels until it finds an ID number that matches its own.

10.11 Registration of ID

- The serial number is unique number set in the factory at the time of shipment and can only be read.
- The ID register is a register for storing the ID number.

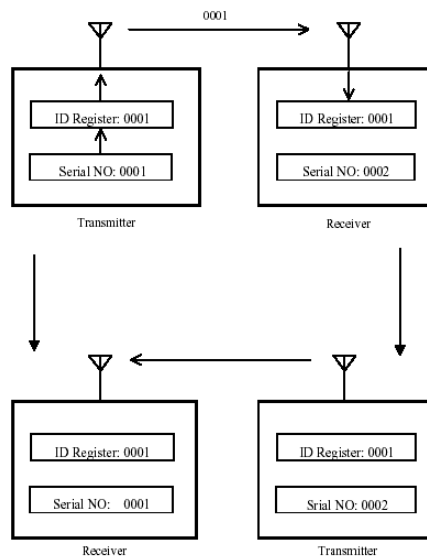


Figure C.5: ID Number Registration

- As shown above, the number registered in the ID register remains the same even when switching between the transmitter and receiver, so it is not necessary to register a new ID number.

Explanation for internal processes (4) [Encoder]

➤ Connection method

- The illustration below shows the basic circuit when the **STD-402** is used in the TX mode (transmitter).
- The 74HC165 is a general-purpose 8-bit parallel input -> serial output converter.

- The **STD-402** detects disconnection, so ensure that a DO is connected to the SI.

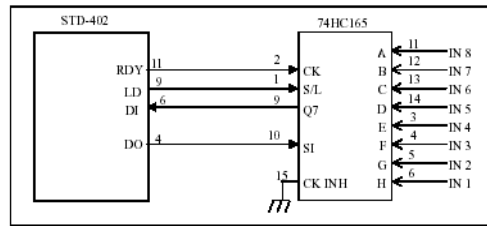


Figure C.6: Encoder

➤ **Timing chart**

- The timing for the **STD-402** TX mode is shown below.
- The micro-computer built into the **STD-402** is equipped with an encoding function and communication protocols to convert the automatic link ID numbers, the various control data and the transmitted data into FSK data.
- The entire data is called as a 'frame', and the transmission data is inserted once into a single frame. The data is transmitted continually until the power supply is switched off.
- The length of the control data is fixed, but the transmission data can be changed between 1 byte to a maximum of 63 bytes in accordance with the application.

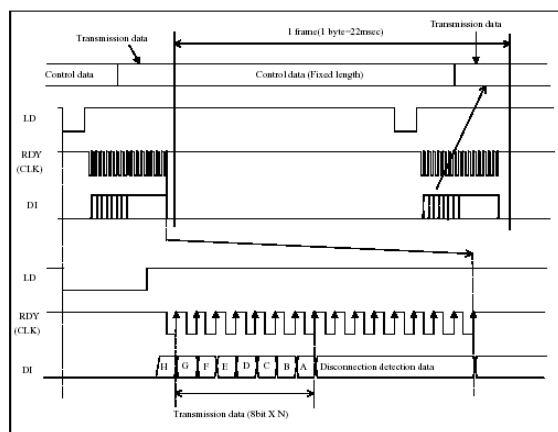


Figure C.7: Timing chart for STD-402 TX mode

- (1) The parallel data is latched onto the clock asynchronously when the LD signal is "L"
- (2) Serial data is shifted at the RDY (clock) rises when the LD signals is "H".

Explanation for internal processes (5) [Decoder]

➤ Connection method

- The illustration below shows the basic circuit when the **STD-402** is used in the RX mode (receiver).
- The 74HC595 is a general-purpose serial input -> 8-bit parallel output converter.
- The **STD-402** detects disconnection, so ensure that a QH is connected to the DI.
- The 74HC595 latch data is cleared when LD is "L"

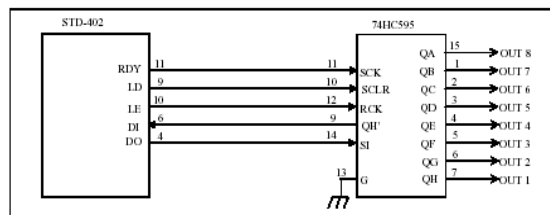


Figure C.8: Decoder

➤ Timing chart

- The timing for the **STD-402** RX mode is shown below.
- The micro-computer built into the **STD-402** is equipped with a decoding function and communication protocols to convert the received FSK data into control data and the transmitted data (serial data).
- The transmission data is output once into a single frame. The data is received continually until the power supply is switched off.

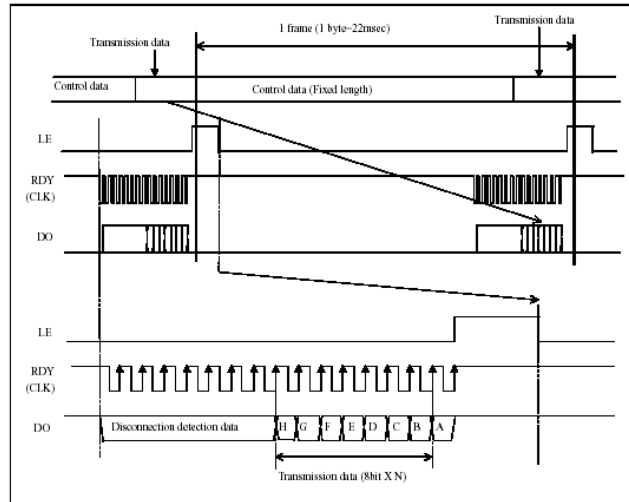


Figure C.9: Timing chart for STD-402 RX mode

- (1) The serial data is shifted when the RDY (clock) rises.
- (2) The parallel data is latched onto the clock asynchronously when the LE signal rises.

Explanation for internal processes (6) [Data format]

➤ Communication data format

- The **STD-402** control data is of fixed length and comes in a unique format that includes the ID code for radio links and special codes.
- The user data (transmission data, receiving data) is between 1 byte and 63 bytes depending on the number of I/O connections, and the transmission time will be extended in accordance with the length of the transmission data.
- The communication data format is shown below.

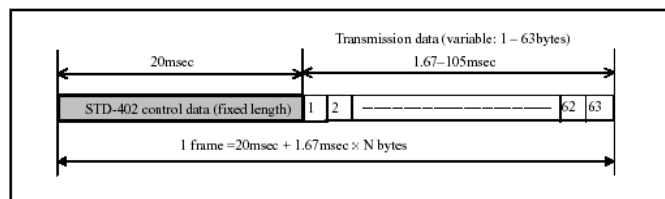


Figure C.10: Data Format

➤ **Transmitting data between Transmitter and receiver**

- The communication data is transmitted continually during transmission. However, the transmission data will not actually travel continuously at 4,800bps owing to the fact that the control data is transmitted by frame by the encoder.
- Using 1 byte of data as an example, the data volume (number of bytes) transmitted within a one-second period is as follows.

$$D = 1000\text{msec} \div \{20\text{msec} + (1.67\text{msec} \times 1)\} \approx 46 \text{ bytes.}$$

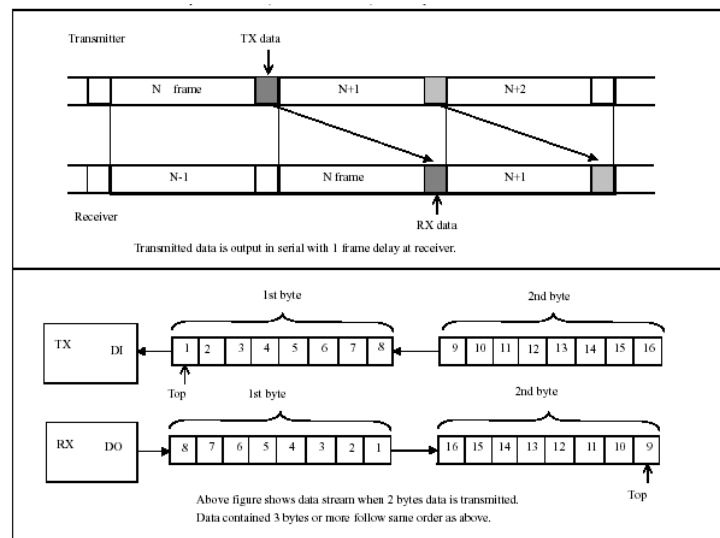


Figure C.11: Transmitting Data between TX and RX

For more details about the Encoder/Decoder and the setting mode, refer to Data Sheet (page 13...19)

Communication mode

- the communication mode enters normal operation when the power supply is switched on after all transceiver settings have been made in the setting mode.
- First of all with the communications mode, a link is established with the transmitter and receiver with the same ID number in accordance with the procedure explained for the automatic link, and data transmission is then started.
- Once a link has been established, communication will be carried out on the same channel unless the power supply is switched off.

Local ID numbers

- It is possible to transmit the data from one transmitter to a maximum of 63 different receivers with the **STD-402** auto mode.
- Local ID numbers must be set for each of the receivers for identification purposes. The ID numbers are set with the CH0 to CH5 ports.
- The (LD = L) reset signal is output and the data is cleared under normal conditions when the ID number of the transmitter and receiver do not match. [LD] is invalidated and the receiving data is output if the ID numbers do match. However, if the ID number for the data and clock matches, it is output regardless of the local ID.
- If the ID number for the transmitter is [0 (ALL)], the receiver will output the data regardless the ID number.

Transmitter	Receiver	Remarks
0 (ALL)	1	All receiver recognize transmitted data.
	2	
	⋮	
	62	
	63	
1	1	Only same ID receiver can recognize transmitted data.
2	2	
⋮	⋮	
62	62	
63	63	

Figure C.12: Local ID Numbers

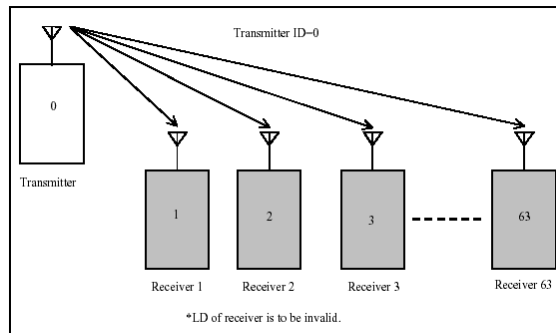


Figure C.13: Communication 1:N

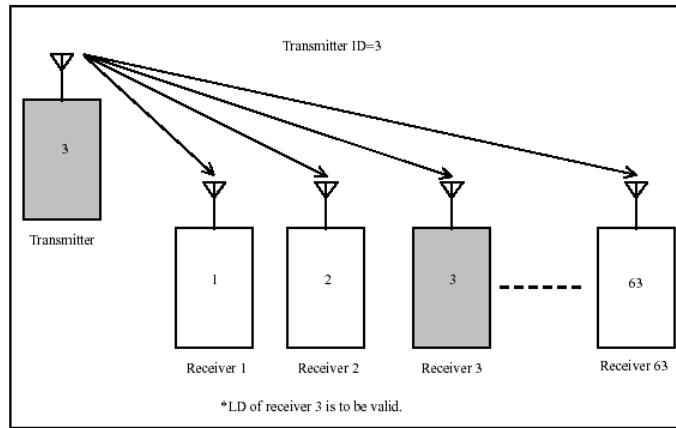
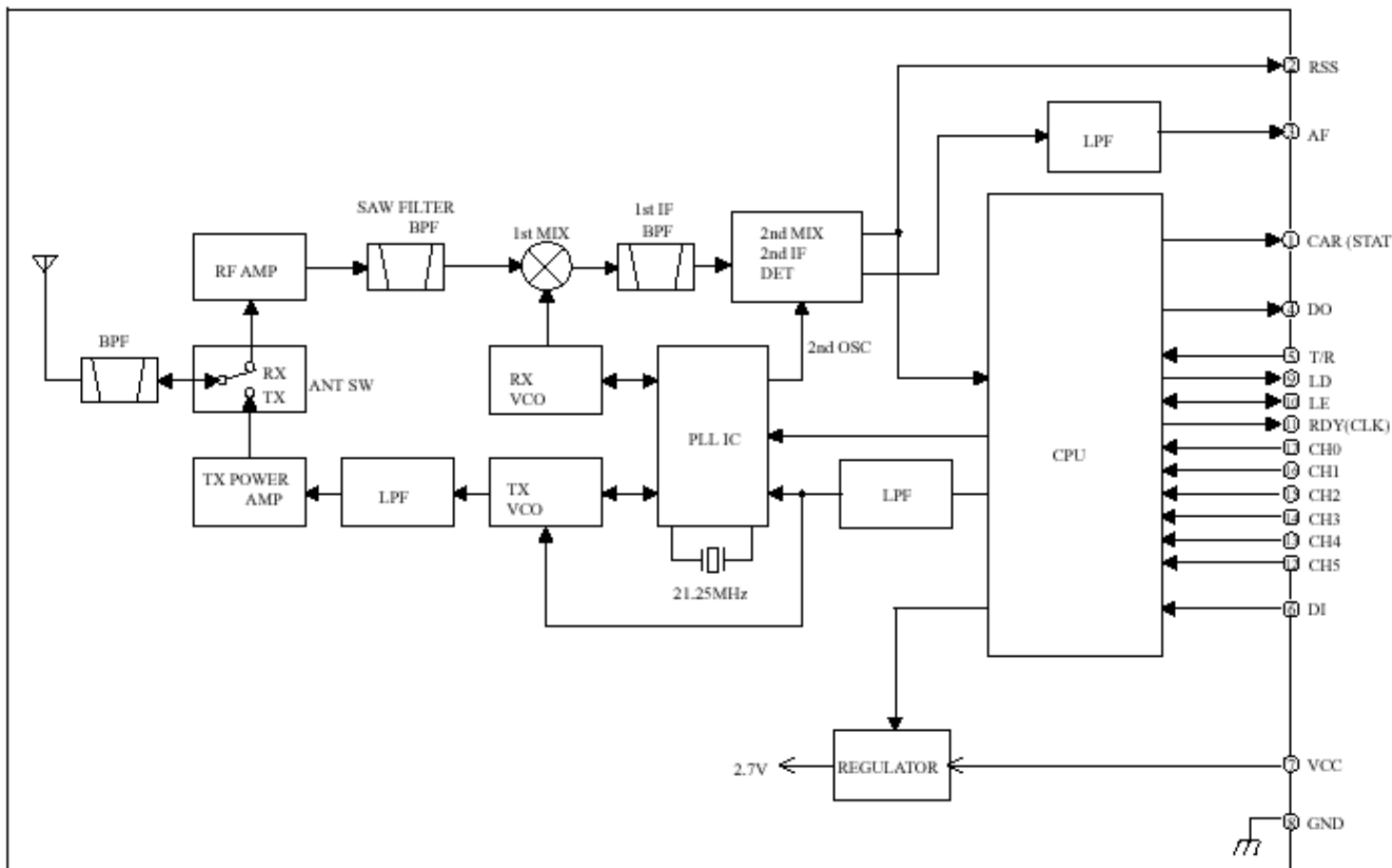


Figure C.14: Communication 1:1

To see the ID address table, refer to Data Sheet (page 22).

10.11.1 diagram of the STD-402 transceiver in Auto Mode

Block diagram



10.12 Annex D

◆ STD-402 [Auto Mode Operation Guide for CPU Interface]

Or the board used to equip the STD-402 is the MB-STD-RS232, so the this mode (Auto Mode Operation Guide for CPU Interface) is used like mode of the STD-402 transceiver.

Auto mode : CPU interface conceptual illustration

➤ Wired Communication

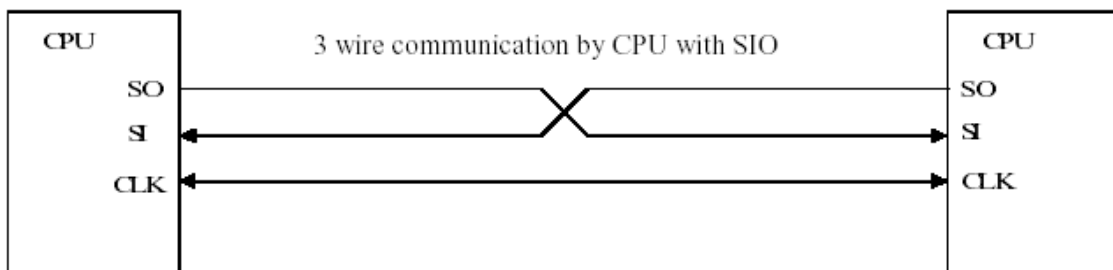


Figure D.1: Wired Communication

Above image shows basic construction of data communication with SIO (Synchronous Serial Input Output)

➤ Radio Communication through STD-402 Auto mode

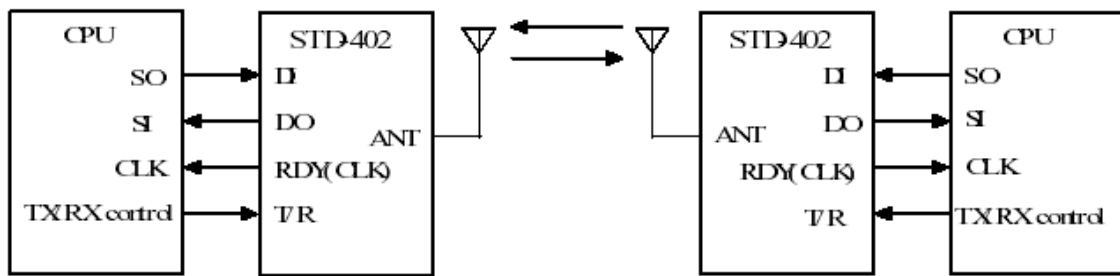


Figure D.2: Radio Communication

Above figure shows the communication that replaces the wire with the radio module (STD-402). Radio communication is not stable as wired communication.

In addition, the procedure in the beginning of radio communication link (connection between transmitter and receiver), TX/RX switching and the disconnection of radio communication are unique for radio communication therefore specific protocol is required for radio communication.

STD-402 auto mode provides such specific protocol as the function of radio link, frequency channel selection and encode/decode.

Transmission concept

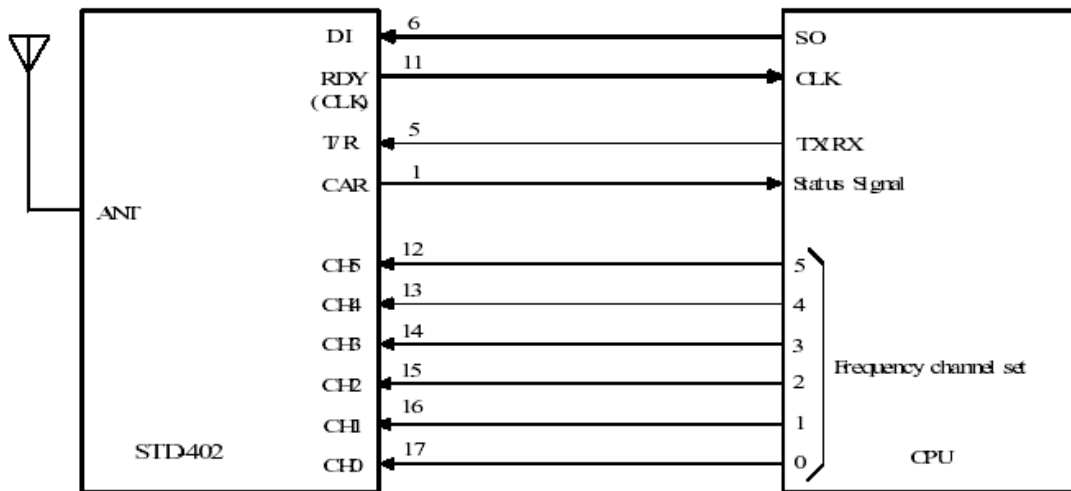


Figure D.3: Transmission Concept

Data from SO (Serial Output) of CPU, which is shifted on the raising edge of clock signal is imputed to DI (Data Input) of **STD-402**. User data can be set with any data size in byte from 1-63 byte. **STD-402** internal CPU adds identification code, error correction code and other internal process code and then transmit them as modulation data in frame.

Reception concept

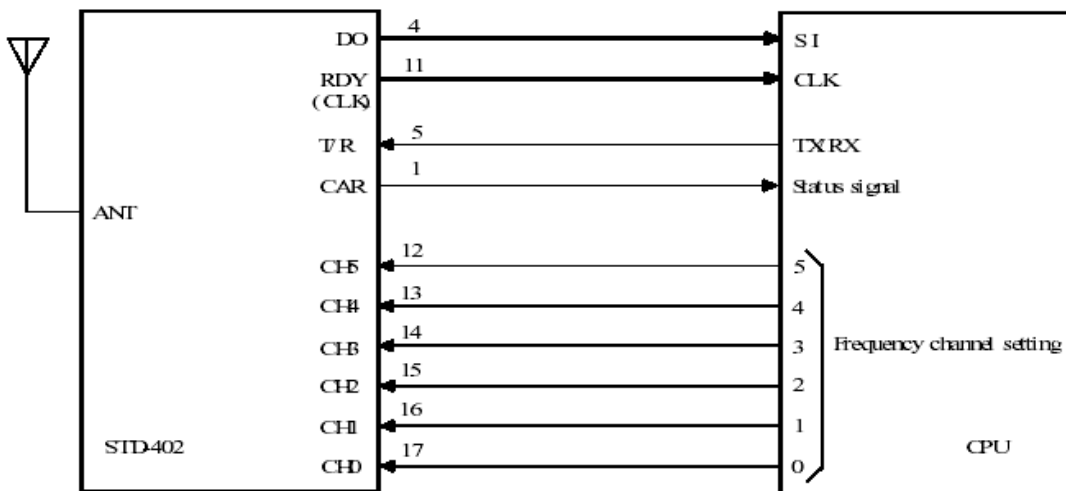


Figure D.4: Reception Concept

When the receiver receives data of which ID code, error correction and code and other internal process code are matched, only user data is outputted from DO by byte in synchronism on the rising edge of clock. If no signal is received or any of ID code, error correction or other process code is not matched, CLK and data signal does not appear.

Connection between STD-402 and CPU

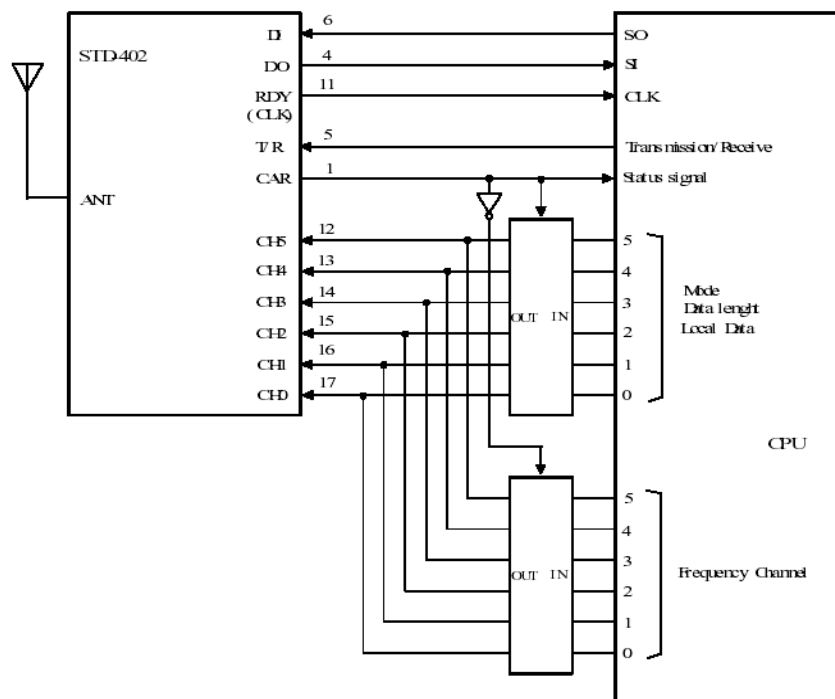


Figure D.5: Connection between STD-402 & CPU

STD-402 Terminal Description

Terminal	Description
DI	Transmission data input terminal Connect to SO (Serial Output) of CPU
DO	Receive data terminal Connect to SI (Serial Input) of CPU

RDY (CLK)	Shift clock output for SO and SI serial data
T/R	Transmitter/Receiver function select terminal L: TX, H: RX
CAR	Status signal for transmission/receive and multiplex switching signal for CH0-5. The port shows L when TX and H when RX. Multiplex switching signal comes at end of frame. It can be used for frame status signal.
CH0 – 5	Input ports for Mode & data rate, frequency channel selection mode (Auto/Fix) and data length (1 – 63 byte) for initial setting. Input port for local ID (0 – 63) and frequency channel (0 – 64 channel) during normal operation.

10.13 Communication Protocol

10.13.1 STD-402 Data Format

Space	Sync. code	Freq. CH Data	ID code	User Data (1 – 63 Byte)	Error correction data
-------	------------	------------------	---------	-------------------------	-----------------------------

Figure D.6: Communication Protocol

STD-402 communication data format is shown as above. **STD-402** internal CPU will make above data automatically except User data. Data connection is possible by feeding user data (1 – 63 bytes) to 3 wire SIO ports of CPU in synchronism with CLK. In each frame, **STD-402** process the data internally and Input/Output user data. In general, data is sent and received as “packet”.

Initial Setting

Initial setting configures **STD-402** internal setting.

1. MODE

- Direct mode
 - Auto mode: (Logic interface)
 - Auto mode: CPU interface
2. FREQUENCY CH (channel)
- Auto channel mode **STD-402** automatically search vacant channel and make radio link.
 - Fixed channel mode – Set frequency channel to 0 – 63 channel.
3. DATA LENGTH
- User data length per frame (1 to 63 byte)

Setting to transmitter (master) and receiver (slave) is different. MODE is set to both transmitter (master) / receiver (slave). FREQUENCY CH and DATA LENGTH are firstly set to transmitter (master) then these data are transmitted to receiver (slave) and then the receiver (slave) memorizes the data.

Above three setting data (MODE, FREQUENCY CH and DATA LENGTH) are inputted to CH0 – 5. MODE and FREQUENCY CH data are inputted to the ports in multiplex process, and DATA LENGTH data are inputted from the port for Mode in sequence.

Initial setting process will start with LE port “L” at power ON and complete with LE port “H”. In normal operation, power supply of the module must turn ON with LE port “H”.

It is recommended to use regulator with controller terminal for **STD-402** power supply control.

Completion of initial setting can be confirmed with LD port output.

For more details about setting parameter, refer to Data Sheet (page 9)

10.13.2 Transmitter : Initial setting flow chart

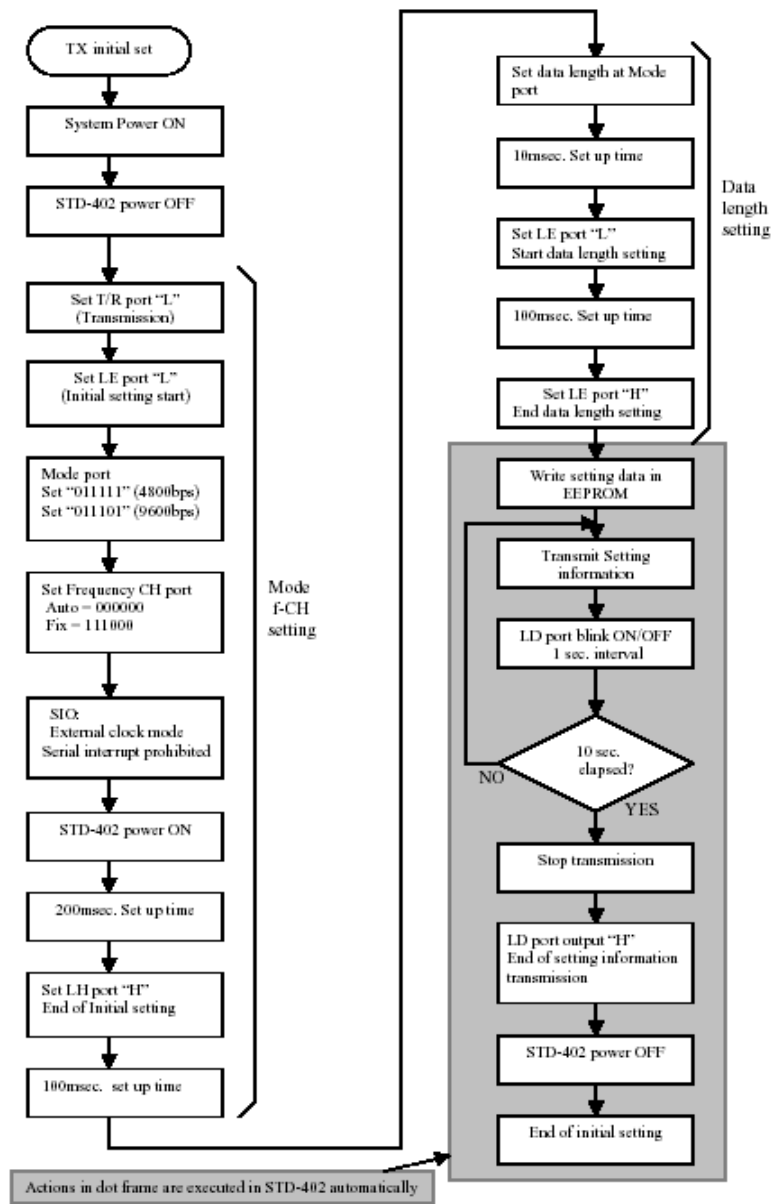


Figure D.7: TX Initial Setting Flow chart

10.13.2.1

Receiver : Initial setting flow chart

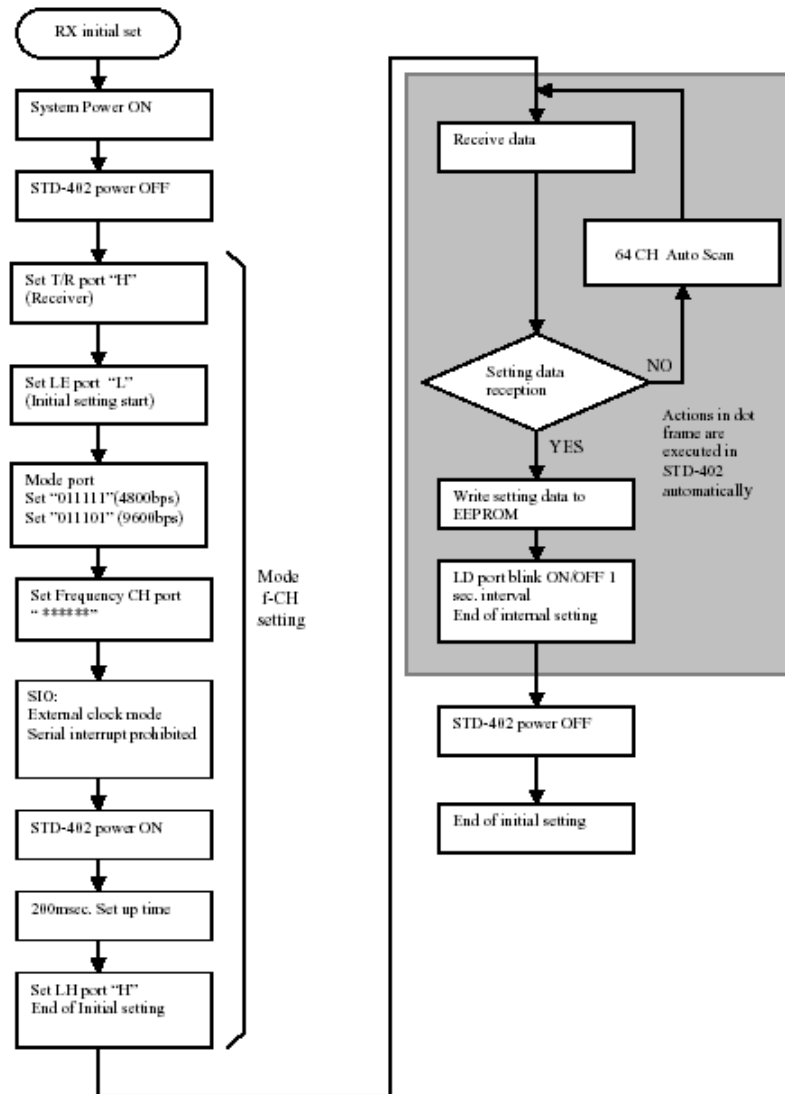


Figure D.8: RX Initial Setting Flow chart

Normal Operation

Local ID and Frequency CH can be set at CH0 – 5 in Normal operation.

1. Local ID
Set Local ID (0 to 63)
2. Frequency CH
Fixed channel mode (0 to 63 CH)

- In normal operation, CLK will be outputted when **STD-402** internal set-up is completed and data communication is ready after power ON, TX/RX switch and frequency CH change. CPU needs to check interruption.
- Set up and communication link between TX and RX. Approximately 10msec of frequency setting time is needed because **STD-402** uses a synthesizer system. Also, demodulation circuit in the receiver work in AC operation, DC drift occurs specially when power is supplied. Thus, RF communication is not stabilized during approx. First 100msec. It is therefore recommended to send dummy data (example 00H) few times before sending actual data.
- Carrier Sense
STD-402 performs carrier sense function to check the set channel is busy or vacant. When module is set fixed channel mode and the set channel is occupied, a clock is not outputted from **STD-402**. In this case, please wait for interruption or change the other frequency channel.

Transmitter: Timing at power ON

- Following shows transmitter timing chart at the time of **STD-402** power supply ON. Set to T/R port to "L" (transmit).
- Transmission cycle of data is defined as frame. Following timing figure shows one byte transmission. It can be set any byte from 1 to 63 bytes. Time of 1 frame is $(20 + \text{the number of byte} * 1.7)$ msec.
- Data transmission is not possible during module setting time (240msec). It is recommended to send 1 to 5 frame dummy data (example 00H) at the beginning of transmission to help initial radio communication link.
- T/R port, frequency CH and local ID data will be read at the end of frame.

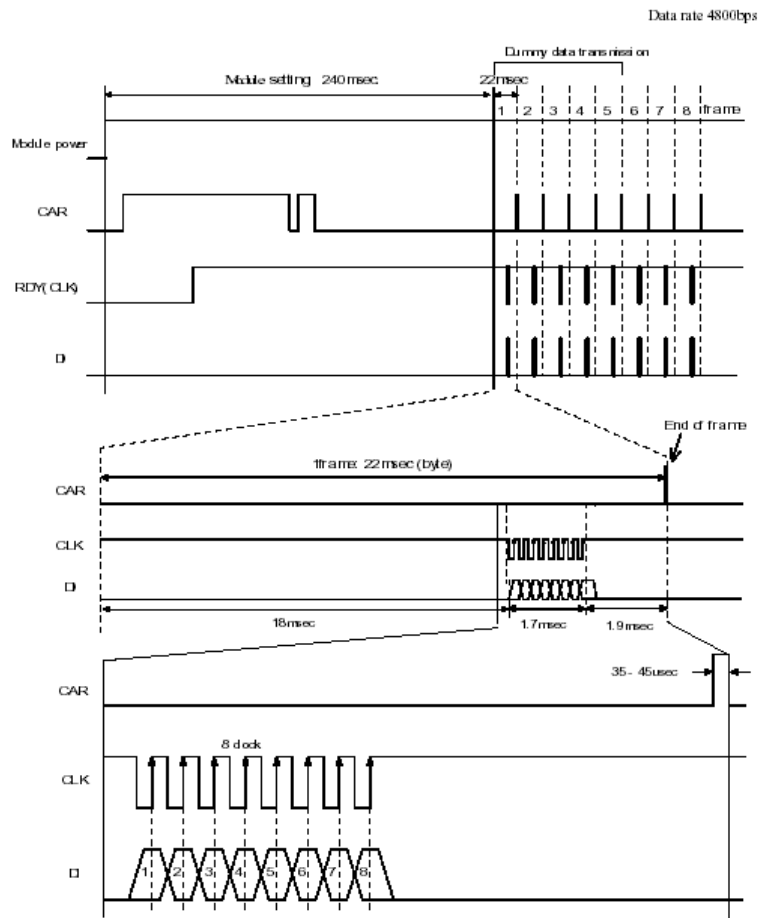


Figure D.9: TX, Timing at power ON

Transmitter: Flow chart at power ON

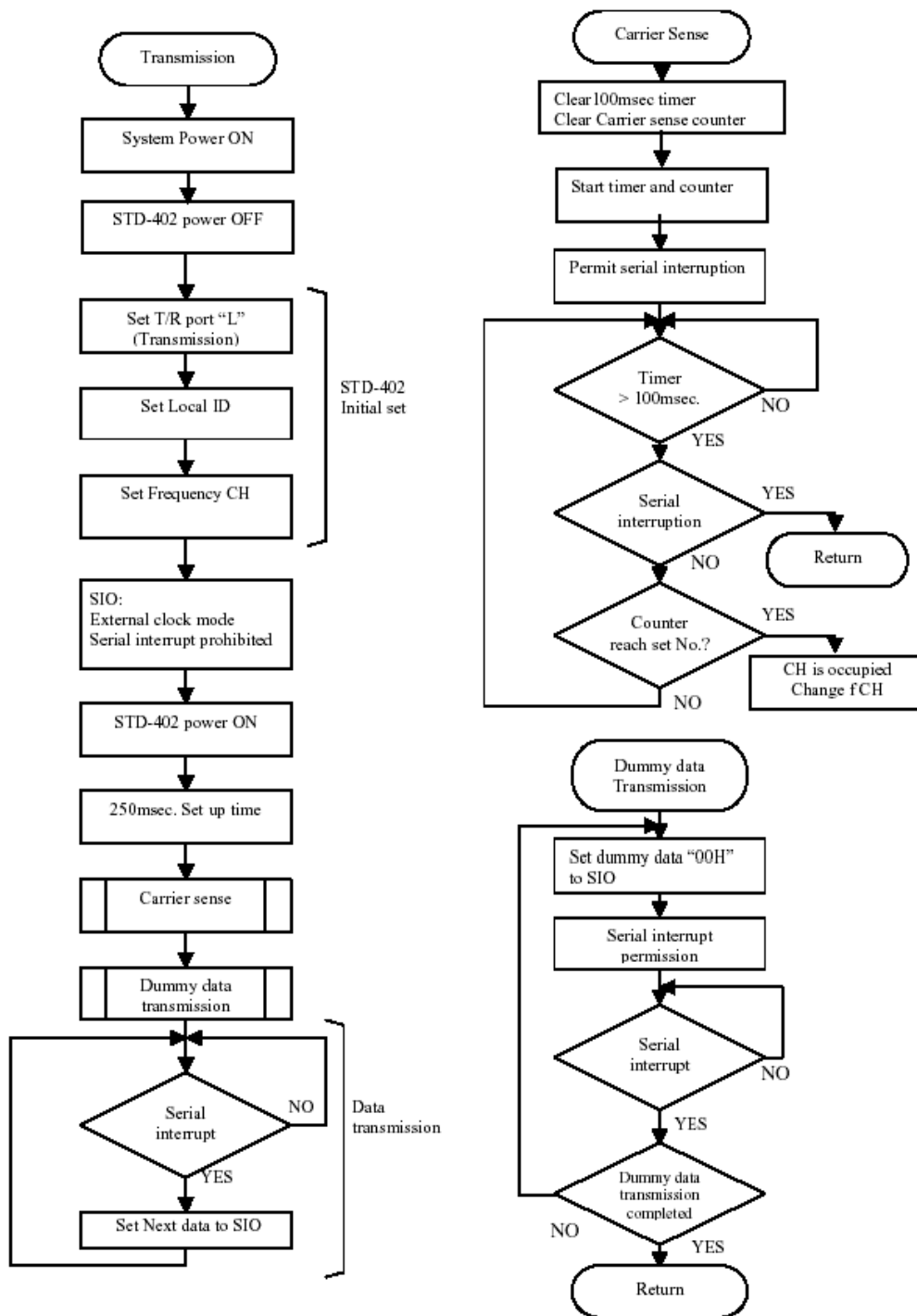


Figure D.10: TX, Flow chart at power ON

Receiver: Timing at power ON

Following shows Receiver timing chart at the time of **STD-402** power supply ON. Set T/R port to "H" (Receiver).

- Receiving cycle of data is defined as frame. Following timing figure shows one byte transmission. It can be set any byte from 1 to 63 bytes. Time of 1 frame is $(20 + \text{the number of byte} * 1.7)$ msec.
- Data can not be received during module setting (approx. 165 msec).
- T/R port, Frequency CH, and Local ID data will be read at the end of frame.
- Following shows timing that receiver turns ON when radio signal from transmitter has been transmitted.

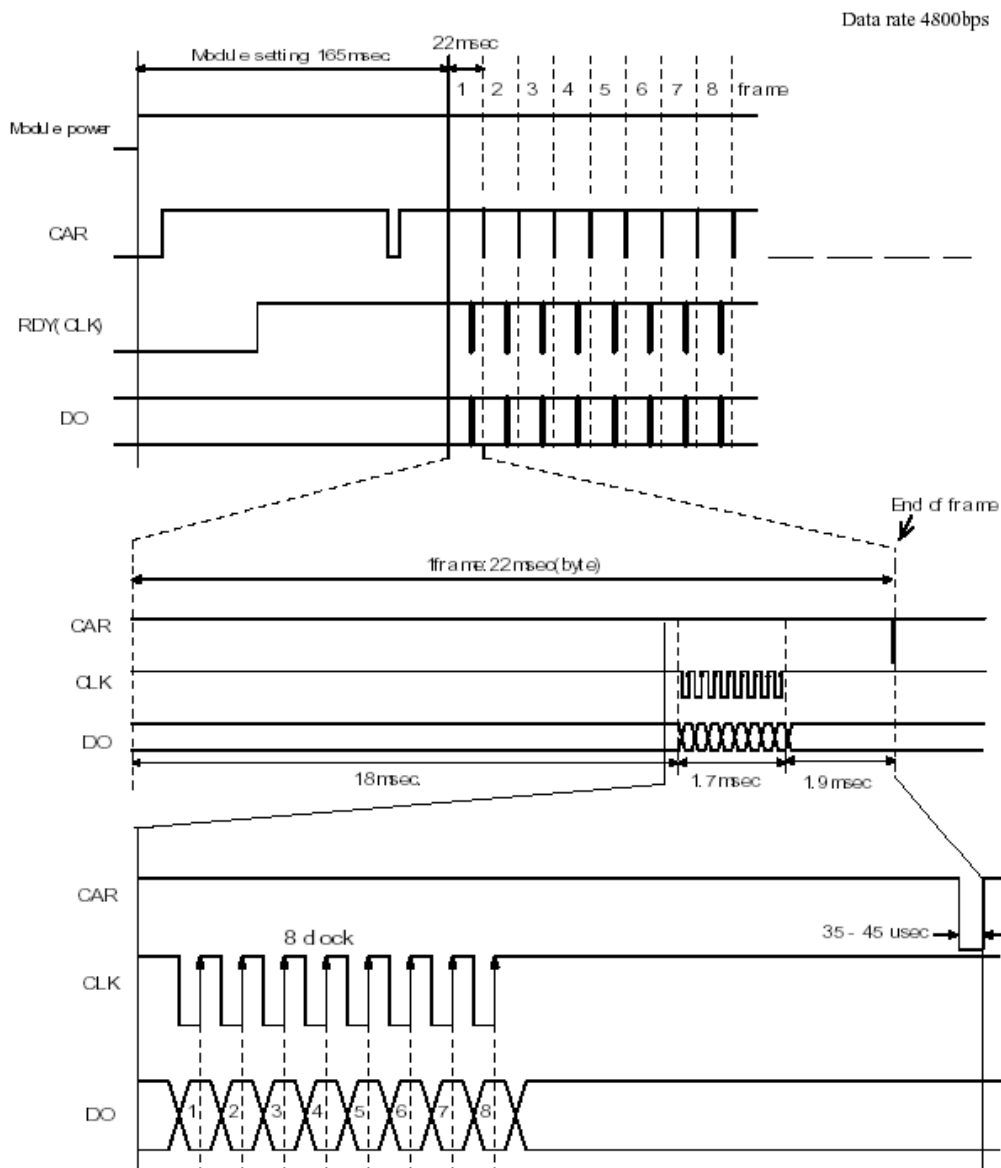


Figure D.11: RX, Timing at Power ON

10.13.3 Receiver: Flow chart at power ON

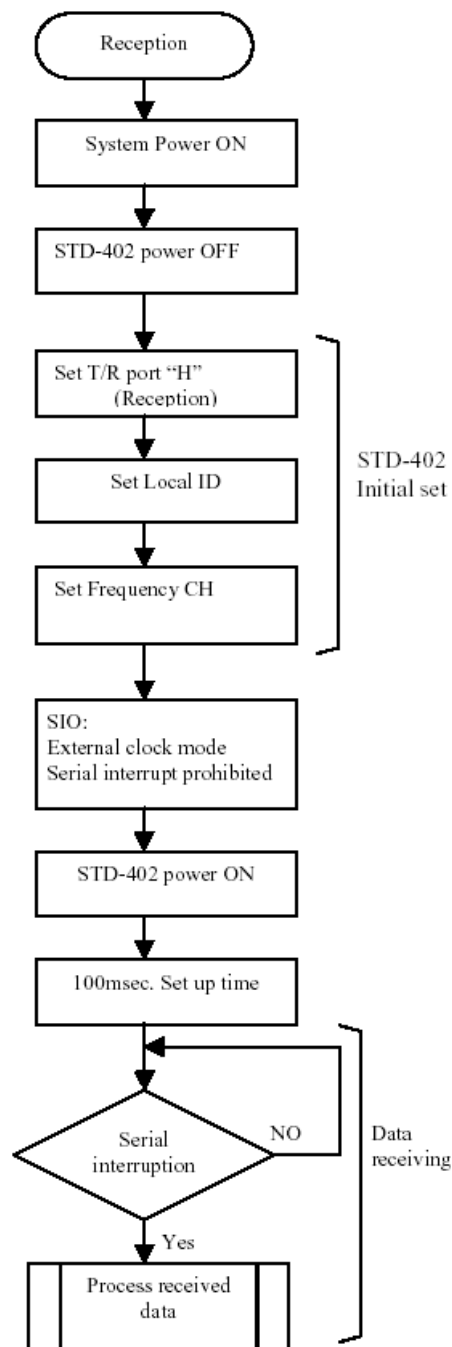


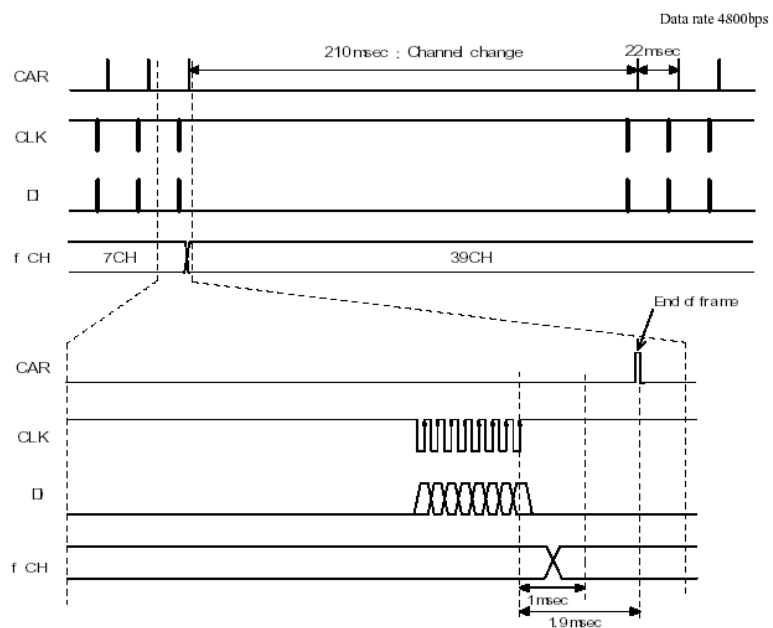
Figure D.12: RX, Flow chart at Power ON

Transmitter: Frequency channel change timing

Following shows timing chart at time when the transmitter frequency channel is changed from 7 to 39 channel.

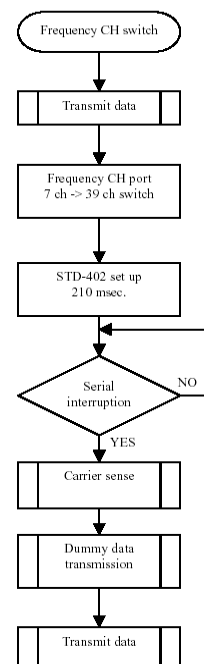
- Channel data will be read at the end of frame and the data will be set at next frame. If channel data is changed at 1 msec or later after final clock (CLK), the data will be read at next frame.
- The figure shows the example that data size is one byte. Data will be read by 22 msec span.
- Data transmission is not possible during channel switching and setting period (approx. 210 msec.). The channel switching time is same at any channel set.
- Transmission will stop and CLK will not be outputted during channel switching and setting period.
- Frequency channel change must be done at the same time for both TX and RX.

Figure D.13: TX,
Frequency CH
change timing



Transmitter: Frequency CH change flow chart

Figure D.14: TX, Frequency CH change Flow chart



- Frequency CH data is read at the end of data frame. See timing chart for detail.
- After frequency CH is switched, transmission become possible when CLK is outputted.

Receiver: Frequency channel change timing

- Following shows timing chart at time when the receiver frequency channel is changed from 15 to 32 channel.
- Channel data will be read at the end of frame and the data will be set at next frame. If channel data is changed at 1 msec or later after final clock (CLK), the data will be read at next frame.
- The figure shows the example that data size is one byte. Data will be read by 22 msec span.
- Data reception is not possible during channel switching period (approx. 210 msec.). The channel switching time is same at any channel set.
- Reception will stop and CLK will not be outputted during channel switching.
- Frequency channel change must be done at the same time for both TX and RX.

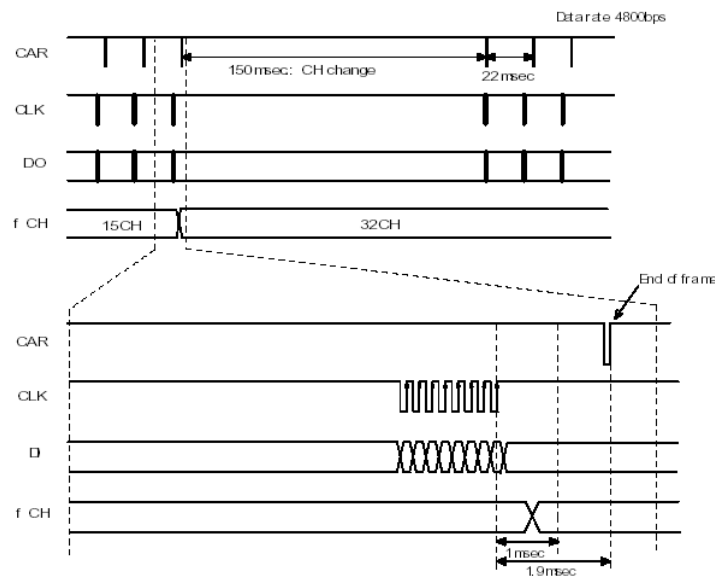


Figure D.15: RX, Frequency CH change timing

Receiver: Frequency CH change flow chart

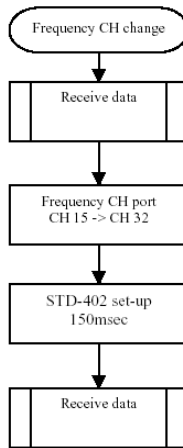


Figure D.16: RX, Frequency CH change Flow chart

- Frequency CH data is read at the end of data frame. See timing chart for detail.
- After frequency CH is switched, reception become possible when CLK is outputted.

Timing at Transmitter -> Receiver switching

- Following shows timing chart at time when the module switches from transmitter to receiver.
- T/R port will be read at the end of frame and the data will be set at next frame. If T/R port is changed at 1 msec or later after final clock (CLK), the data will be read at next frame.
- The figure shows the example with one byte data size and 4800bps data rate.
- Data reception is not possible during the receiver setting period (approx. 365 msec.).
- Reception become possible when CLK is outputted.

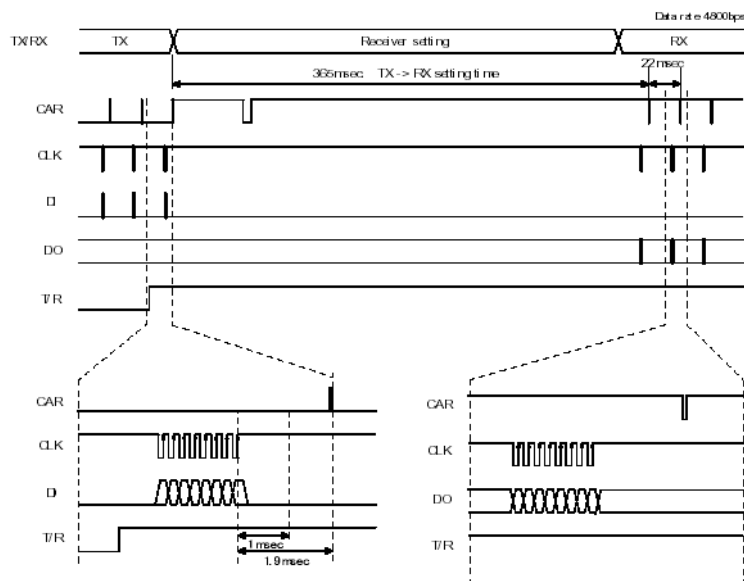


Figure D.17: Timing at TX -> RX switching

Timing at Receiver -> Transmitter switching

- Following shows timing chart at time when the module switches from receiver to transmitter.
- T/R port will be read at the end of frame and the data will be set at next frame. If T/R port is changed at 1 msec or later after final clock (CLK), the data will be read at next frame.
- The figure shows the example with one byte data size and 4800bps data rate.
- Data transmission is not possible during the receiver setting period (approx. 200 msec.).
- Transmission become possible when CLK is outputted.
- As described in timing at power ON, send dummy data "00H" for 1 – 5 frame at beginning of transmission to make radio link with receiver.

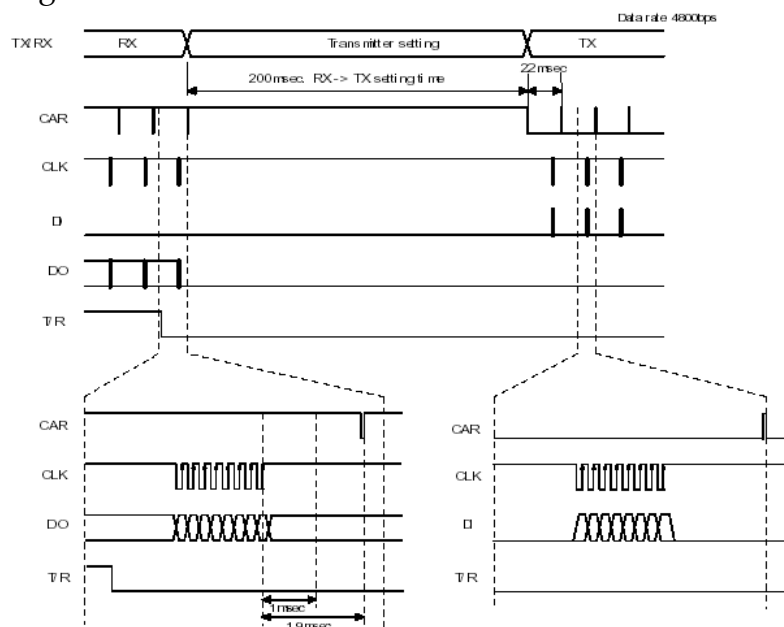


Figure D.18: Timing at RX -> TX switching

Transmitter <-> Receiver switching flow chart

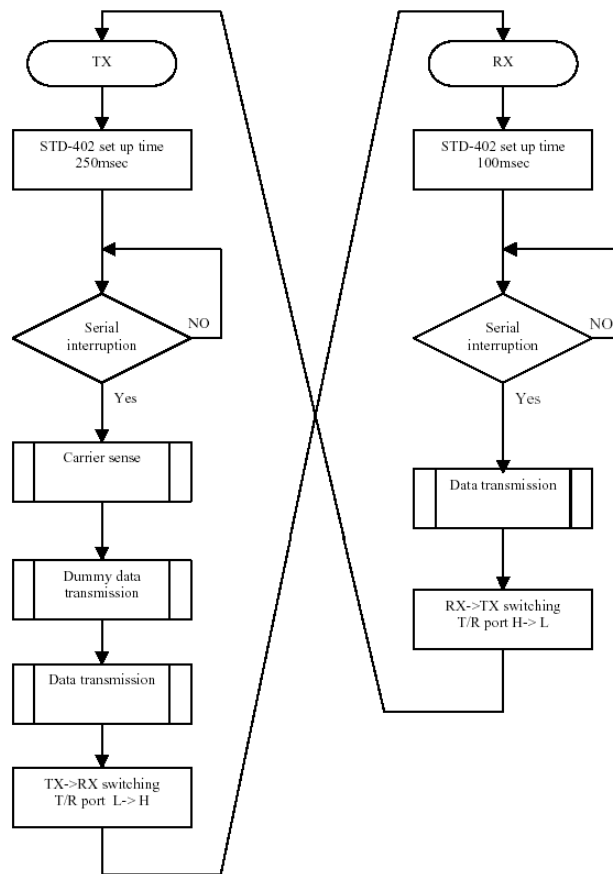
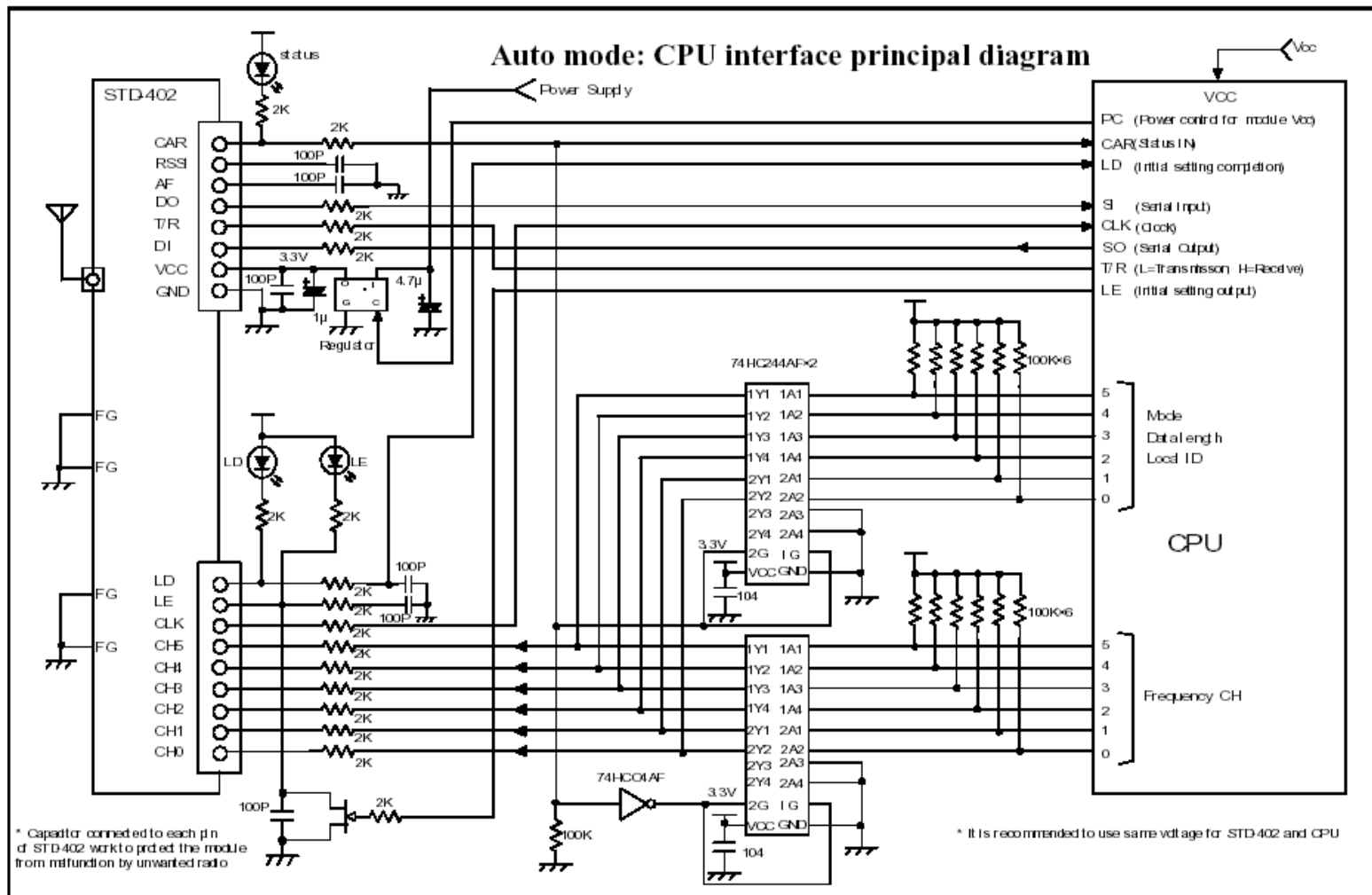


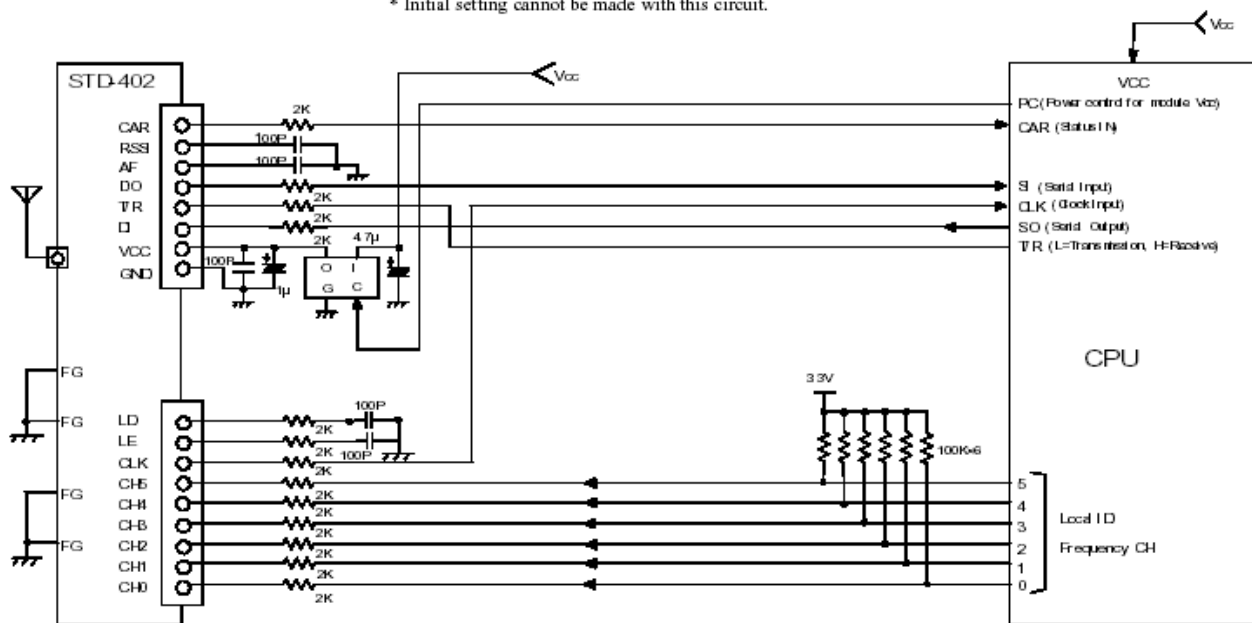
Figure D.19: TX <-> RX switching Flow chart

- TX -> RX or RX -> TX change data will be read at end of frame.
- After switching TX/RX, transmission and reception is possible when CLK is outputted.



Auto mode: CPU interface diagram without multiplex

- * Circuit diagram example simplified by deleting CH 0-5 multiplex circuit.
- * CH 0-5 operations is differed by frequency channel mode.
 Auto CH mode: Local ID input
 Fixed CH mode: Local ID and Frequency CH are same
- * Initial setting cannot be made with this circuit.



It is recommended to use same voltage for STD-402 and CPU

10.14 Annex E: Visual Basic code for the user interface program

```

1      ' Parameters using to do a Zoom on the PictureMap
2      Dim r1 As Integer, X1 As Integer, X2 As Integer, Y1 As Integer, Y2 As Integer
3      Dim ULeftX As Integer, ULeftY As Integer, Target As PictureBox

4      Private Sub C_Close_Click()
5          'Close the Serial Port
6          MSComm1.PortOpen = False

7          'Close the Receive_File
8          Close #1

9          'Close the Send_file
10         Close #2

11        'Close the Solar_file
12        Close #3
    
```

```

13      'Close the User Interface Program
14      'Close the FormMap Program
15      'Close the FormDirection Program
16      Unload FormMap
17      Unload FormDirection
18      Unload Me
19      End
20      End Sub

21      Private Sub C_Connection_Click()
22          'Wait for Data (Sensors Data) to come to the Serial Port
23          ' 116 is the length of the Receive_Message
24          Do
25              DoEvents
26          Loop Until MSCComm1.InBufferCount >= 116

28          ' Call the function used to Read Data from Serial Port ( Read_Data)
29          ' and in this function, we will test the Begin and the End Flag
30          Read_Data
31      End Sub

32      Private Sub C_Send_Cancel_Click()
33          'Enable the timer used for Receive Data form serial Port
34          TimerRun_R.Enabled = True

35          'Disable the SDF ( Sending Data Frame)
36          SDF.Visible = False

37          'clear the TextBoxes from the data
38          T_newPos_inX.Text = ""
39          T_newPos_inY.Text = ""
40          T_newPos_inZ.Text = ""
41          T_newVel_inX.Text = ""
42          T_newVel_inY.Text = ""

```

```

43         T_newVel_inZ.Text = ""
44     End Sub

45     Private Sub C_Send_Commands_Click()
46         'Enable le SDF (Send_Data_Frame)
47         SDF.Visible = True

48         'Clear the Transmit Buffer
49         MSComm1.OutBufferCount = 0

50         'Clear the TextBoxes from Data
51         T_newPos_inX.Text = ""
52         T_newPos_inY.Text = ""
53         T_newPos_inZ.Text = ""
54         T_newVel_inX.Text = ""
55         T_newVel_inY.Text = ""
56         T_newVel_inZ.Text = ""

57         'Set the Cursor in the first TextBox (new_Position inX)
58         T_newPos_inX.SetFocus
59     End Sub

60     Private Sub C_Send_OK_Click()
61         'Testing the status of the new information (TextBoxes)
62         If (UserInterface.T_newPos_inX.Text = "" Or UserInterface.T_newPos_inY.Text = "" Or 63
UserInterface.T_newPos_inZ.Text = "" _
64         Or UserInterface.T_newVel_inX.Text = "" Or UserInterface.T_newVel_inY.Text = "" Or 65
UserInterface.T_newVel_inZ = "") Then
66         MsgBox (" SORRY, but you should put a number in these TEXTBOXES !!!!!")
67         T_newPos_inX.SetFocus
68         Else
69         'Disable the Timer using for Read_Data
70         TimerRun_R.Enabled = False

71         'call the function used for writing Commands in the Transmit Buffer
72         Write_Commands

```

```

73      'Disable the SDF (Send_Data_Frame)
74      SDF.Visible = False

75      'Enable the Timer used for Reading Data
76      TimerRun_R.Enabled = True
77      End If
78      End Sub

79      Private Sub C_Solar_Radiation_Click()
80      'Close the Solar_file.rad
81      Close #3

82      'Execute the Program used to draw the Position of the
83      'ALTERNATIVE-Lotte in the 3D (three Dimensions) with
84      ' the value of the Solar Radiation
85      Shell "C:\3D\Jet.exe", vbNormalFocus
86      End Sub

87      Private Sub Form_Load()
88      'Initialisation the form of the User Interface
89      UserInterface.Width = Screen.Width
90      UserInterface.Height = Screen.Height

91      'Initialisation of the position of the ALTERNATIVE-Lotte on the MAP
92      CenterX0 = 3
93      CenterY0 = PictureMap.Height - 1702

94      'Initialisation of the Counter used for drawing
95      ' the Wind Vector and the Solar
96      CountW = 1

98      'Initialisation of the Value used to display
99      'the position and the direction in the large view

```



```

100             CLK_M = False
101             CLK_D = False

102 'Open the file (Send_file) that the user used it to put the Telegram (Message) that he will 103
send to the ALTERNATIVE-Lotte

104 Open "C:\User Interface\Data\Send_file.snd" For Binary Access Write As #2
105 SI = 0

106 ' Open the file (Receive_file) that the user used it to put the Telegram (Message) that he 107
will receive from the ALTERNATIVE-Lotte

108 Open "C:\User Interface\Data\Receive_file.rcv" For Binary Access Write As #1
109 RI = 0

110 'Open the file (Solar_File) that the user used it to put the value of the Solar Radiation
111 Open "C:\User Interface\Data\Solar_file.rad" For Binary Access Write As #3
112 SRI = 0

113             'Set the drive an path for the JPG or BMP image and the Mouse icon
114 ChDrive Left$(App.Path, 2)
115 ChDir App.Path

116 'The shape control for the ZoomBox or frame is loaded dynamically
117 'at runtime
118 Controls.Add "vb.shape", "shpZoomBox", PictureMap
119 Set Target = FormMap.PictureMap_Z
120 Target.Width = FormMap.PictureMap_Z.Width
121 Target.Height = FormMap.PictureMap_Z.Height

122             'Disable the SDF (Send_Data_Frame)
123             SDF.Visible = False

124             'Disable the Solar Radiation Command
125             C_Solar_Radiation.Enabled = False

126             'Test the Status of the serial Port
127             If (Status_Port(MSComm1, 3) = False) Then

```

```

128 MsgBox "SORRY!!!, This port is not Available (used by another application)"
129 End
130     Else
131     ' Initialisation of the serial port
132     MSComm1.CommPort = 3
133     MSComm1.Settings = "9600,E,8,2"
134     MSComm1.Handshaking = comRTSXOnXOff
135     MSComm1.RTSEnable = True
136     MSComm1.InputMode = comInputModeText
137     MSComm1.InputLen = 116
138     MSComm1.PortOpen = True

139     'Initialisation of the value of the Recevie_Message
140     RMessage.DataIn_AX = "0"
141     RMessage.DataIn_AY = "0"
142     RMessage.DataIn_AZ = "0"
143     RMessage.DataIn_AVX = "0"
144     RMessage.DataIn_AVY = "0"
145     RMessage.DataIn_AVZ = "0"
146     RMessage.DataIn_Azt = "0"
147     RMessage.DataIn_WX = "0"
148     RMessage.DataIn_WY = "0"
149     RMessage.DataIn_WZ = "0"
150     RMessage.DataIn_SX = "0"
151     RMessage.DataIn_SY = "0"
152     RMessage.DataIn_SZ = "0"
153     RMessage.DataIn_Temp = "0"

154     'Set the Interval of the timer
155     TimerRun_R.Interval = 500

156     'Enable the Timer used for Reading Data
157     TimerRun_R.Enabled = True
158     End If
159     End Sub

```

```

160         Private Sub PictureDir_Click()
161             'Enable the From_Direction
162             FormDirection.Visible = True

163             'Disable the SDF (Sending_Data_Frame)
164             SDF.Visible = False

165             FormDirection.SetFocus

166             'Close the FormMap
167             Unload FormMap

168             CLK_D = True
169             If (Azimuts < 0) Then
170                 YD_Z = 1900 - (1650 * Cos((Azimuts * PI) / 180))
171                 XD_Z = 2450 + (1650 * Sin((Azimuts * PI) / 180))
172                 FormDirection.PictureDir_Z.Line (2450, 1900)-(XD_Z, YD_Z), RGB(255, 0, 0)
173             Else
174                 YD_Z = 1900 - (1650 * Cos((Azimuts * PI) / 180))
175                 XD_Z = 2480 + (1650 * Sin((Azimuts * PI) / 180))
176                 FormDirection.PictureDir_Z.Line (2480, 1900)-(XD_Z, YD_Z), RGB(255, 0, 0)
177             End If
178         End Sub

```

```

Private Sub PictureMap_MouseDown(Button As Integer, Shift As Integer, X As
    Single, Y As Single)

```

```

180         'Get the Position of the Mouse_Down
181         X1 = X
182         Y1 = Y

183         'Get the position of the MouseDown.It is mean get
184         'the X and Y when the User press the MouseDown
185         Controls("shpZoomBox").Left = X1
186         Controls("shpZoomBox").Top = Y1

```

```

187         Controls("shpZoomBox").Width = 0
188         Controls("shpZoomBox").Height = 0
189         Controls("shpZoomBox").BorderStyle = 3
190         Controls("shpZoomBox").Visible = True

191     End Sub

Private Sub PictureMap_MouseMove(Button As Integer, Shift As Integer, X As
    Single, Y As Single)
193         'Call the function used to do a Zoom
194         ZoomBoxArea X1, X, Y1, Y

195         If Controls("shpZoomBox").Visible = True Then

196     'Change the Mouse Icon
197     PictureMap.MousePointer = vbCustom
198     PictureMap.MouseIcon = LoadPicture("cursor20.ico")

199     Controls("shpZoomBox").Width = minX(X1, X)
200     Controls("shpZoomBox").Height = minY(Y1, Y)

201         End If
202     End Sub

Private Sub PictureMap_MouseUp(Button As Integer, Shift As Integer, X As Single,
    Y As Single)
204         'Enable the FormMap
205         FormMap.Visible = True
206         FormMap.SetFocus

207         'Close the FormDirection
208         Unload FormDirection
209         FormMap.C_ZoomMap.Visible = True
210         CLK_M = True

```

```

211         Controls("shpZoomBox").Visible = False
212         PictureMap.MousePointer = vbArrow

213         'Get the position of the Mouse_Up
214         X2 = X
215         Y2 = Y

216         ZoomBoxArea X1, X2, Y1, Y2

217         'The StretchBlt API function copies a bitmap from a source
218         'rectangle into a destination rectangle.
219         r1 = StretchBlt(Target.hdc, 0, 0, Target.Width, Target.Height, PictureMap.hdc,
ULeftX, 220 ULeftY, minX(X1, X2), minY(Y1, Y2), SRCCOPY)

221         ' show the updated image
222         Target.Refresh ' show the updated image

223         'Disable the SDF (Sending_Data_Frame)
224         SDF.Visible = False
225         End Sub

226         Private Sub T_newvel_inX_KeyPress(KeyAscii As Integer)
227         'Testing the status of the TextBoxes, so if it is not clear, jump to the other
228         If (T_newVel_inX.Text <> "" And KeyAscii = 13) Then
229         T_newVel_inY.SetFocus
230         ElseIf (KeyAscii = 27) Then
231         T_newPos_inZ.SetFocus
232         T_newVel_inX.Text = ""
233         ElseIf (T_newVel_inX.Text = "") Then
234         T_newVel_inX.SetFocus
235         End If
236         End Sub

237         Private Sub T_newvel_inY_KeyPress(KeyAscii As Integer)
238         'Testing the status of the TextBoxes, so if it is not clear, jump to the other
239         If (T_newVel_inY.Text <> "" And KeyAscii = 13) Then

```

```

240 T_newVel_inZ.SetFocus
241 ElseIf (KeyAscii = 27) Then
242 T_newVel_inX.SetFocus
243 T_newVel_inY.Text = ""
244 ElseIf (T_newVel_inY.Text = "") Then
245 T_newVel_inY.SetFocus
246         End If
247         End Sub

248         Private Sub T_newvel_inZ_KeyPress(KeyAscii As Integer)
249             'Testing the status of the TextBoxes, so if it is not clear, jump to the other
250             If (T_newVel_inZ.Text <> "" And KeyAscii = 13) Then
251 C_Send_OK.SetFocus
252 ElseIf (KeyAscii = 27) Then
253 T_newVel_inY.SetFocus
254 T_newVel_inZ.Text = ""
255 ElseIf (T_newVel_inZ.Text = "") Then
256 T_newVel_inZ.SetFocus
257         End If
258         End Sub

259         Private Sub T_newPos_inX_KeyPress(KeyAscii As Integer)
260             'Testing the status of the TextBoxes, so if it is not clear, jump to the other
261             If (T_newPos_inX.Text <> "" And KeyAscii = 13) Then
262 T_newPos_inY.SetFocus
263 ElseIf (T_newPos_inX = "") Then
264 T_newPos_inX.SetFocus
265         End If
266         End Sub

267         Private Sub T_newPos_inY_KeyPress(KeyAscii As Integer)
268             'Testing the status of the TextBoxes, so if it is not clear, jump to the other
269             If (T_newPos_inY.Text <> "" And KeyAscii = 13) Then
270 T_newPos_inZ.SetFocus
271 ElseIf (KeyAscii = 27) Then

```

```

272 T_newPos_inX.SetFocus
273 T_newPos_inY.Text = ""
274 ElseIf (T_newPos_inY.Text = "") Then
275 T_newPos_inY.SetFocus
276         End If
277         End Sub

278         Private Sub T_newPos_inZ_KeyPress(KeyAscii As Integer)
279         'Testing the status of the TextBoxes, so if it is not clear, jump to the other
280         If (T_newPos_inZ.Text <> "" And KeyAscii = 13) Then
281 T_newVel_inX.SetFocus
282 ElseIf (KeyAscii = 27) Then
283 T_newPos_inY.SetFocus
284 T_newPos_inZ.Text = ""
285 ElseIf (T_newPos_inZ.Text = "") Then
286 T_newPos_inZ.SetFocus
287         End If
288         End Sub

289         Private Sub TimerRun_R_Timer()
290         'Wait for Data (Sensors Data) to come to the Serial Port
291         '116 is the length of the Receive_Message
292         Do
293             DoEvents
294         'Enable The Solar Radiation Command
295         'because when the program stop here
296         'It is mean that the travel of the
297         'ALTERNATIVE-Lotte is finished
298         C_Solar_Radiation.Enabled = True
299         Loop Until MSComm1.InBufferCount >= 116

300         'Disable the Solar Radiation Command
301         'because when the program continue
302         'It is mean that the ALTERNATIVE-Lotte
303         'continue its tavel

```

```

304         C_Solar_Radiation.Enabled = False

305     'Call the function used to Read Data
306     Read_Data
307     End Sub

308     'these 3 functions minX, minY and ZoomBoxArea are used
309     'to do a Zoom on the position of the ALTERNATIVE - Lotte
310     Public Function minX(X1, X2)

311     If X1 < X2 Then
312         minX = X2 - X1
313         Controls("shpZoomBox").Left = X1
314     Else
315         minX = X1 - X2
316         Controls("shpZoomBox").Left = X1 - minX
317     End If

318         End Function

319     Public Function minY(Y1, Y2)

320     If Y1 < Y2 Then
321         minY = Y2 - Y1
322         Controls("shpZoomBox").Top = Y1
323     Else
324         minY = Y1 - Y2
325         Controls("shpZoomBox").Top = Y1 - minY
326     End If

327         End Function

328     Public Sub ZoomBoxArea(X1, X2, Y1, Y2)

329     If X1 < X2 And Y1 > Y2 Then
330         ULeftX = X1
331         ULeftY = Y2

```



```

332 ElseIf X1 > X2 And Y1 > Y2 Then
333 ULeftX = X2
334 ULeftY = Y2
335 ElseIf X1 > X2 And Y1 < Y2 Then
336 ULeftX = X2
337 ULeftY = Y1
338 Else
339 ULeftX = X1
340 ULeftY = Y1
341 End If
342           End Sub

```

Visual Basic code for the library (Library_RW) used in the user interface program

```

343           Public Const PI = 3.141592
344           'Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

345           'these two functions are used to do a Zoom for a picture
346           Public Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal X
As
347           Long, ByVal Y As Long, ByVal nWidth As Long, ByVal nHeight As Long,
ByVal
hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As
349           Long) As Long
350 Public Declare Function StretchBlt Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, 351
ByVal Y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As 352 Long,
ByVal xSrc As Long, ByVal ySrc As Long, ByVal nSrcWidth As Long, ByVal 353 nSrcHeight As
Long, ByVal dwRop As Long) As Long

354 Public Const SRCAND = &H8800C6      ' (DWORD) dest = source AND dest
355 Public Const SRCCOPY = &HCC0020    ' (DWORD) dest = source
355 Public Const SRCERASE = &H440328   ' (DWORD) dest = source AND (NOT dest)
356 Public Const SRCINVERT = &H660046  ' (DWORD) dest = source XOR dest
357 Public Const SRCPAINT = &HEE0086   ' (DWORD) dest = source OR dest

```

```

358         'Structure of the Receive_Message
359         Public Type Struct_RMessage
360 Begin_RFlag As String      ' The really type is Byte
361 Length_RMessage As String  ' The really type is Interger
362 DataIn_AX As String        ' The really type is Single
363 DataIn_AY As String        ' The really type is Single
364 DataIn_AZ As String        ' The really type is Single
365 DataIn_AVX As String       ' The really type is Single
366 DataIn_AVY As String       ' The really type is Single
367 DataIn_AVZ As String       ' The really type is Single
368 DataIn_Azt As String       ' The really type is Single
369 DataIn_WX As String        ' The really type is Single
370 DataIn_WY As String        ' The really type is Single
371 DataIn_WZ As String        ' The really type is Single
372 DataIn_SX As String        ' The really type is Single
373 DataIn_SY As String        ' The really type is Single
374 DataIn_SZ As String        ' The really type is Single
375 DataIn_Temp As String      ' The really type is Single
376 End_RFlag As String        ' The really type is Byte
377         End Type

378         Public RMessage As Struct_RMessage

379         'String (Variable) used for Read Data from Serial Port
380         Public Receive_Message As String

381         'Structure of the Send_Message
382         Public Type Struct_SMessage
383 Begin_SFlag As String      ' The really type is Byte
384 Length_SMessage As String  ' The really type is Interger
385 DataOut_PX As String       ' The really type is Single
386 DataOut_PY As String       ' The really type is Single
387 DataOut_PZ As String       ' The really type is Single
388 DataOut_VelX As String     ' The really type is Single
389 DataOut_VelY As String     ' The really type is Single

```

```

390 DataOut_VelZ As String      ' The really type is Single
391 End_SFlag As String        ' The really type is Byte
392         End Type

393         Public SMessage As Struct_SMessage

394         'String (Variable) used for Send Data to Serial Port
395         Public Send_Message As String

396         'Parameters used to save the Old Data
397         Public Acc_inX As Single
398         Public Acc_inY As Single
399         Public Acc_inZ As Single
400         Public Ang_vel_inX As Single
401         Public Ang_vel_inY As Single
402         Public Ang_vel_inZ As Single
403         Public Azimuts As Single
404         Public Wind_inX As Single
405         Public Wind_inY As Single
406         Public Wind_inZ As Single
407         Public Solar_inX As Single
408         Public Solar_inY As Single
409         Public Solar_inZ As Single
410         Public Temp As Single

411         'Parameters used to get the Velocity and the Position
412         Public Vel_inX As Single
413         Public Vel_inY As Single
414         Public Vel_inZ As Single
415         Public Pos_inX As Single
416         Public Pos_inY As Single
417         Public Pos_inZ As Single

418         ' Parameters used for drawing the Position
419         '   of the ALTERNATIVE-Lotte on the MAP

```

```

420         Public CenterX As Single
421         Public CenterY As Single
422         Public CenterX0 As Single
423         Public CenterY0 As Single
424         Public CLK_M As Boolean

425         ' Parameters used for drawing the Direction
426         '   of the ALTERNATIVE-Lotte
427         Public XD As Single
428         Public YD As Single
429         Public XD_Z As Single
430         Public YD_Z As Single
431         Public CLK_D As Boolean

432         'Parameters used to put the Wind Vector
432         Public RW As Single      'Wind Power
433         Public CountW As Integer

434         'Parameters used to put the Solar Radiation
435         Public RS As Single      'Solar Radiation

436         'Paramters used to put the Telegram in the file
437         Public RI As Integer
438         Public SI As Integer
439         Public SRI As Integer

440         Public Function Read_Data() As Boolean
441         ' Read the Contain of the Receive Buffer
442         Receive_Message = UserInterface.MSComm1.Input

443         If (Read_Data) Then
444         'Save the old position
445         Acc_inX = Val(RMessage.DataIn_AX)
446         Acc_inY = Val(RMessage.DataIn_AY)
447         Acc_inZ = Val(RMessage.DataIn_AZ)

```

```

448  Ang_vel_inX = Val(RMessage.DataIn_AVX)
449  Ang_vel_inY = Val(RMessage.DataIn_AVY)
450  Ang_vel_inZ = Val(RMessage.DataIn_AVZ)
451  Azimuts = Val(RMessage.DataIn_Azt)
452  Wind_inX = Val(RMessage.DataIn_WX)
453  Wind_inY = Val(RMessage.DataIn_WY)
454  Wind_inZ = Val(RMessage.DataIn_WZ)
455  Solar_inX = Val(RMessage.DataIn_SX)
456  Solar_inY = Val(RMessage.DataIn_SY)
457  Solar_inZ = Val(RMessage.DataIn_SZ)
458  Temp = Val(RMessage.DataIn_Temp)
459          End If

460          ' Call the function used to convert from Hex Format to Decimal Format
461          UserInterface.SC1.Reset
462          UserInterface.SC1.AddCode UserInterface.txtCodeHex2Dec.Text

463          RMessage.Begin_RFlag = Val("&H" & Left(Receive_Message, 1))
464          RMessage.Length_RMessage = Val("&H" & Mid(Receive_Message, 2, 2))
465          RMessage.DataIn_AX          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 4, 8))
466          RMessage.DataIn_AY          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 12, 8))
467          RMessage.DataIn_AZ          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 20, 8))
468          RMessage.DataIn_AVX          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 28, 8))
469          RMessage.DataIn_AVY          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 36, 8))
470          RMessage.DataIn_AVZ          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 44, 8))
471          RMessage.DataIn_Azt          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 52, 8))
472          RMessage.DataIn_WX          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 60, 8))
473          RMessage.DataIn_WY          =          UserInterface.SC1.Run("compute",
Mid(Receive_Message, 68, 8))

```

```

474      RMessage.DataIn_WZ      =      UserInterface.SC1.Run("compute",
Mid(Receive_Message, 76, 8))
475      RMessage.DataIn_SX      =      UserInterface.SC1.Run("compute",
Mid(Receive_Message, 84, 8))
476      RMessage.DataIn_SY      =      UserInterface.SC1.Run("compute",
Mid(Receive_Message, 92, 8))
477      RMessage.DataIn_SZ      =      UserInterface.SC1.Run("compute",
Mid(Receive_Message, 100, 8))
478      RMessage.DataIn_Temp    =      UserInterface.SC1.Run("compute",
Mid(Receive_Message, 108, 8))
479      RMessage.End_RFlag = Val("&H" & Right(Receive_Message, 1))

480      ' Testing the Begin_Flag and the End_Flag for each Receive_Message
481      If (RMessage.Begin_RFlag <> 2 Or RMessage.End_RFlag <> 3) Then
482      ' In this section the Begin_Flag or the End_Flag is NOT Correct
483      Read_Data = False

484      ' Display the Old Data
485      UserInterface.L_Acc_inX.Caption = Acc_inX
486      UserInterface.L_Acc_inY.Caption = Acc_inY
487      UserInterface.L_Acc_inZ.Caption = Acc_inZ
488      UserInterface.L_Ang_inX.Caption = Ang_vel_inX
489      UserInterface.L_Ang_inY.Caption = Ang_vel_inY
490      UserInterface.L_Ang_inZ.Caption = Ang_vel_inZ
491      UserInterface.L_Wind_inX.Caption = Wind_inX
492      UserInterface.L_Wind_inY.Caption = Wind_inY
493      UserInterface.L_Wind_inZ.Caption = Wind_inZ
494      UserInterface.L_Solar_inX.Caption = Solar_inX
495      UserInterface.L_Solar_inY.Caption = Solar_inY
496      UserInterface.L_Solar_inZ.Caption = Solar_inZ
497      UserInterface.L_Temp.Caption = Temp
498      UserInterface.L_Vel_inX.Caption = Vel_inX
499      UserInterface.L_Vel_inY.Caption = Vel_inY
500      UserInterface.L_Vel_inZ.Caption = Vel_inZ
501      UserInterface.L_Pos_inX.Caption = Pos_inX
502      UserInterface.L_Pos_inY.Caption = Pos_inY

```

```

503      UserInterface.L_Pos_inZ.Caption = Pos_inZ

503      'display the status of the connection to the user
504      UserInterface.L_Connection.Caption = " TAKE CARE !!!! , there is NO
connection 505 between the Base Station and the ALTERNATIVE-Lotte"
506      UserInterface.Pic_statusOK.Visible = True
507      UserInterface.Pic_statusNO.Visible = False
508      Else
509      ' In this Section the Begin_Flag and the End_Flag are Correct
510      Read_Data = True

511      UserInterface.Text1.Text = Val(UserInterface.Text1.Text) + 1
512      ' Display the sensors data in the RDF (Receive_Data_Frame)
513      UserInterface.L_Acc_inX.Caption = Left(RMessage.DataIn_AX, 8)
514      UserInterface.L_Acc_inY.Caption = Left(RMessage.DataIn_AY, 8)
515      UserInterface.L_Acc_inZ.Caption = Left(RMessage.DataIn_AZ, 8)
516      UserInterface.L_Ang_inX.Caption = Left(RMessage.DataIn_AVX, 8)
517      UserInterface.L_Ang_inY.Caption = Left(RMessage.DataIn_AVY, 8)
518      UserInterface.L_Ang_inZ.Caption = Left(RMessage.DataIn_AVZ, 8)
519      UserInterface.L_Wind_inX.Caption = Left(RMessage.DataIn_WX, 8)
520      UserInterface.L_Wind_inY.Caption = Left(RMessage.DataIn_WY, 8)
521      UserInterface.L_Wind_inZ.Caption = Left(RMessage.DataIn_WZ, 8)
522      UserInterface.L_Solar_inX.Caption = Left(RMessage.DataIn_SX, 8)
523      UserInterface.L_Solar_inY.Caption = Left(RMessage.DataIn_SY, 8)
524      UserInterface.L_Solar_inZ.Caption = Left(RMessage.DataIn_SZ, 8)
525      UserInterface.L_Temp.Caption = Left(RMessage.DataIn_Temp, 8)

526      'Save the Data
527      Acc_inX = Left(RMessage.DataIn_AX, 8)
528      Acc_inY = Left(RMessage.DataIn_AY, 8)
529      Acc_inZ = Left(RMessage.DataIn_AZ, 8)
530      Ang_vel_inX = Left(RMessage.DataIn_AVX, 8)
531      Ang_vel_inY = Left(RMessage.DataIn_AVY, 8)
532      Ang_vel_inZ = Left(RMessage.DataIn_AVZ, 8)
533      Azimuts = Left(RMessage.DataIn_Azt, 8)
534      Wind_inX = Left(RMessage.DataIn_WX, 8)

```

```

535      Wind_inY = Left(RMessage.DataIn_WY, 8)
536      Wind_inZ = Left(RMessage.DataIn_WZ, 8)
537      Solar_inX = Left(RMessage.DataIn_SX, 8)
538      Solar_inY = Left(RMessage.DataIn_SY, 8)
539      Solar_inZ = Left(RMessage.DataIn_SZ, 8)
540      Temp = Left(RMessage.DataIn_Temp, 8)

541      'Put the Sensors_Data in the Recevie_File
542      'Put #1, (1 + 60 * RI), Begin_RFlag
543      'Put #1, (2 + 60 * RI), Length_RMessage
544      Put #1, (4 + 60 * RI), Acc_inX
545      Put #1, (8 + 60 * RI), Acc_inY
546      Put #1, (12 + 60 * RI), Acc_inZ
547      Put #1, (16 + 60 * RI), Ang_vel_inX
548      Put #1, (20 + 60 * RI), Ang_vel_inY
549      Put #1, (24 + 60 * RI), Ang_vel_inZ
550      Put #1, (28 + 60 * RI), Azimuts
551      Put #1, (32 + 60 * RI), Wind_inX
552      Put #1, (36 + 60 * RI), Wind_inY
553      Put #1, (40 + 60 * RI), Wind_inZ
554      Put #1, (44 + 60 * RI), Solar_inX
555      Put #1, (48 + 60 * RI), Solar_inY
556      Put #1, (52 + 60 * RI), Solar_inZ
557      Put #1, (56 + 60 * RI), Temp
558      'Put #1, (60 + 60 * RI), End_RFlag
559      RI = RI + 1

560      'Call the function used to Calculate the Velocity of the ALTERNATIVE-Lotte
      Calculate_Velocity

561      'Call the function used to Calculate the Position of the ALTERNATIVE-Lotte
      Calculate_Position

562      'Get the position of the Center of the Circle
563      'used to put the position of the ALTERNATIVE-Lotte

```



```

564         CenterX = CenterX0 + CSng(UserInterface.L_Pos_inX.Caption)
565         CenterY = CenterY0 - CSng(UserInterface.L_Pos_inY.Caption)

566         'Put the Position on the MAP
567         UserInterface.PictureMap.PSet (CenterX, CenterY), RGB(255, 0, 0)

568         'Clear he old direction of the ALTERNATIVE-Lotte
569         UserInterface.PictureDir.Cls
570         FormDirection.PictureDir_Z.Cls

571         'Put the Direction in the PictureDir
572         If (Azimuts < 0) Then
573             YD = 850 - (700 * Cos((Azimuts * PI) / 180))
574             XD = 920 + (700 * Sin((Azimuts * PI) / 180))
575             UserInterface.PictureDir.Line (920, 850)-(XD, YD), RGB(255, 0, 0)
576             Else
577             YD = 850 - (700 * Cos((Azimuts * PI) / 180))
578             XD = 960 + (700 * Sin((Azimuts * PI) / 180))
579             UserInterface.PictureDir.Line (960, 850)-(XD, YD), RGB(255, 0, 0)
580             End If

581         'Testing if the user click in the pictureDirection (ZOOM)
582         If (CLK_D) Then
583             'Put the Direction on the PictureDirection in large View (ZOOM)
584             If (Azimuts < 0) Then
585                 YD_Z = 1900 - (1650 * Cos((Azimuts * PI) / 180))
586                 XD_Z = 2450 + (1650 * Sin((Azimuts * PI) / 180))
587                 FormDirection.PictureDir_Z.Line (2450, 1900)-(XD_Z, YD_Z), RGB(255, 0, 0)
588             Else
589                 YD_Z = 1900 - (1650 * Cos((Azimuts * PI) / 180))
590                 XD_Z = 2480 + (1650 * Sin((Azimuts * PI) / 180))
591                 FormDirection.PictureDir_Z.Line (2480, 1900)-(XD_Z, YD_Z), RGB(255, 0, 0)
592             End If
593         End If

```

```

594          'Put the Wind Vector (Wind Power) in the PictureWind
RW = (Val(UserInterface.L_Wind_inX.Caption) ^ 2 +
      Val(UserInterface.L_Wind_inY.Caption) ^ 2 +
      Val(UserInterface.L_Wind_inZ.Caption) ^ 2) ^ (1 / 2)
UserInterface.PictureWind.PSet (CountW, UserInterface.PictureWind.Height - 1365
      - (RW / 13)), RGB(255, 0, 0)

597  'Put the Solar Vector (Solar Radiation) in the PictureSolar
RS = (Val(UserInterface.L_Solar_inX.Caption) ^ 2 +
      Val(UserInterface.L_Solar_inY.Caption) ^ 2 +
      Val(UserInterface.L_Solar_inZ.Caption) ^ 2) ^ (1 / 2)
UserInterface.PictureSolar.PSet (CountW, UserInterface.PictureSolar.Height - 1365
      - (RS / 13)), RGB(255, 0, 0)

599  CountW = CountW + 2

600          'Put the Solar Radiation Data in the Solar_file
601          Put #3, (1 + 17 * SRI), Pos_inX
602          Put #3, (5 + 17 * SRI), Pos_inY
603          Put #3, (9 + 17 * SRI), Pos_inZ
604          Put #3, (13 + 17 * SRI), RS
605          SRI = SRI + 1

606  'Display the Status of the Connection to the User
607  UserInterface.L_Connection.Caption = "There is a connection between the Base Station 608
and the ALTERNATIVE-Lotte"
609  UserInterface.Pic_statusNO.Visible = True
610  UserInterface.Pic_statusOK.Visible = False
611  End If

612          'After reading the receive message, we clean the receive buffer
613          UserInterface.MSComm1.InBufferCount = 0
614          End Function

615          Public Function Write_Commands() As Boolean

616          ' Call the function used to convert from Decimal format to Hex format

```

```

617         UserInterface.SC1.Reset
618         UserInterface.SC1.AddCode UserInterface.txtCodeDec2Hex.Text

619         'configure the Send Message
620         SMessage.Begin_SFlag = Hex$(2)

621         SMessage.Length_SMessage = Hex$(28)

622         If (Val(UserInterface.T_newPos_inX.Text) =
                CInt(Val(UserInterface.T_newPos_inX.Text))) Then
623     UserInterface.T_newPos_inX.Text = (UserInterface.T_newPos_inX.Text & ".000001")
    SMessage.DataOut_PX = UserInterface.SC1.Run("compute", UserInterface.T_newPos_inX.Text)
625 Else
626     SMessage.DataOut_PX = UserInterface.SC1.Run("compute",
        UserInterface.T_newPos_inX.Text)
627 End If

628         If (Val(UserInterface.T_newPos_inY.Text) =
                CInt(Val(UserInterface.T_newPos_inY.Text))) Then
628     UserInterface.T_newPos_inY.Text = UserInterface.T_newPos_inY.Text & ".000001"
    SMessage.DataOut_PY = UserInterface.SC1.Run("compute",
        (UserInterface.T_newPos_inY.Text))
630         Else
631     SMessage.DataOut_PY = UserInterface.SC1.Run("compute",
        (UserInterface.T_newPos_inY.Text))
632         End If

633         If (Val(UserInterface.T_newPos_inZ.Text) =
                CInt(Val(UserInterface.T_newPos_inZ.Text))) Then
634     UserInterface.T_newPos_inZ.Text = UserInterface.T_newPos_inZ & ".000001"
    SMessage.DataOut_PZ = UserInterface.SC1.Run("compute",
        (UserInterface.T_newPos_inZ.Text))
637         Else
    SMessage.DataOut_PZ = UserInterface.SC1.Run("compute",
        (UserInterface.T_newPos_inZ.Text))

```

```

639             End If

640             If (Val(UserInterface.T_newVel_inX.Text) =
                Cint(Val(UserInterface.T_newVel_inX.Text))) Then
641     UIFace.T_newVel_inX.Text = UIFace.T_newVel_inX.Text & ".000001"
        SMessage.DataOut_VelX = UIFace.SC1.Run("compute",
                (UIFace.T_newVel_inX.Text))
643             Else
644     SMessage.DataOut_VelX = UIFace.SC1.Run("compute",
                (UIFace.T_newVel_inX.Text))
645             End If

645             If (Val(UserInterface.T_newVel_inY.Text) =
                Cint(Val(UserInterface.T_newVel_inY.Text))) Then
646     UIFace.T_newVel_inY.Text = UIFace.T_newVel_inY.Text &
".000001"
        SMessage.DataOut_VelY = UIFace.SC1.Run("compute",
                (UIFace.T_newVel_inY.Text))
648             Else
649     SMessage.DataOut_VelY = UIFace.SC1.Run("compute",
                (UIFace.T_newVel_inY.Text))
650             End If

        If (Val(UserInterface.T_newVel_inZ.Text) =
            Cint(Val(UserInterface.T_newVel_inZ.Text))) Then
652     UIFace.T_newVel_inZ.Text = UIFace.T_newVel_inZ.Text & ".000001"
653     SMessage.DataOut_VelZ = UIFace.SC1.Run("compute",
            (UIFace.T_newVel_inZ.Text))
654             Else
        SMessage.DataOut_VelZ = UIFace.SC1.Run("compute",
            (UIFace.T_newVel_inZ.Text))
656             End If

657             SMessage.End_SFlag = Hex$(3)

658             'Put the Data (Commands) in the Send_Message

```

```

659   Send_Message = SMessage.Begin_SFlag & SMessage.Length_SMessage & _
      SMessage.DataOut_PX & SMessage.DataOut_PY & _
      SMessage.DataOut_PZ & SMessage.DataOut_VelX & _
      SMessage.DataOut_VelY & SMessage.DataOut_VelZ & _
      SMessage.End_SFlag

660       'Put the Telegram (Send_Message) in the Send_File
661       Put #2, (1 + 60 * SI), Send_Message
662       SI = SI + 1

663       'send the Message to the Transmit Buffer
664       UserInterface.MSComm1.Output = Send_Message

665       'Set the Status of the Write_Commands function
665       Write_Commands = True
666       End Function

667       ' the Goal of this function is to calculate the Velocity
668       ' of the ALTERNATIVE - Lotte in X, Y and Z
        ' The Velocity is the Integration of the Acceleration

670       Public Function Calculate_Velocity() As Boolean
671   Vel_inX = (RMessage.DataIn_AX) * (UserInterface.TimerRun_R.Interval / 1000)
672       Vel_inY = (RMessage.DataIn_AY) * (UserInterface.TimerRun_R.Interval / 1000)
673       Vel_inZ = (RMessage.DataIn_AZ) * (UserInterface.TimerRun_R.Interval / 1000)
674       UserInterface.L_Vel_inX.Caption = Vel_inX
675       UserInterface.L_Vel_inY.Caption = Vel_inY
676       UserInterface.L_Vel_inZ.Caption = Vel_inZ
677       Calculate_Velocity = True
678       End Function

679       ' the Goal of this function is to calculate the Positon
680       ' of the ALTERNATIVE - Lotte in X, Y and Z
681       ' The Position is the Integration of the Velocity

```

```

682     Public Function Calculate_Position() As Boolean
683     Pos_inX = Vel_inX * (UserInterface.TimerRun_R.Interval / 1000)
684     Pos_inY = Vel_inY * (UserInterface.TimerRun_R.Interval / 1000)
685     Pos_inZ = Vel_inZ * (UserInterface.TimerRun_R.Interval / 1000)
686     UserInterface.L_Pos_inX.Caption = Pos_inX
687     UserInterface.L_Pos_inY.Caption = Pos_inY
688     UserInterface.L_Pos_inZ.Caption = Pos_inZ
689     Calculate_Position = True
690     End Function

691     Public Function Status_Port(cComm As MSComm, IPort As Long) As Boolean

692     'This function Test the Status of the Serial Port
693     'NOTE : Do not try to send/receive data while executing this function...!

694     Dim oldState As Boolean
695     Dim oldPort As Long

696     ' Get current MSComm Info
697     oldState = cComm.PortOpen
698     oldPort = cComm.CommPort

699     ' Change Info
700     On Error Resume Next
701     cComm.PortOpen = False
702     cComm.CommPort = IPort

703     ' See if Port is Available
704     Err.Clear
705     cComm.PortOpen = True
706     If Err Then
707     Status_Port = False
708     Debug.Print Err.Description
709     Else
710     Status_Port = True

```

```
711             End If
712  cComm.PortOpen = False

713             ' Set Info Back To Old Data
714             cComm.CommPort = oldPort
715             cComm.PortOpen = oldState
716             On Error GoTo 0

717             End Function
```

10.15 Annex F: C program code

```
/******  
**  
** Project          : Diploma Thesis  
**  
**                The goal of this program is to initialise the serial port  
**                and read from one serial port and write to another.....  
**  
*****/  
  
/*-- Include files -----*/  
  
1 #include <stdio.h>      /* Standard input/output definitions */  
2 #include <string.h>     /* String function definitions */  
3 #include <unistd.h>     /* UNIX standard function definitions */  
4 #include <fcntl.h>     /* File Control definition */  
5 #include <errno.h>     /* Error number definitions */  
6 #include "vxWorks.h"  
7 #include "ioLib.h"  
8 #include <sioLib.h>  
  
/******  
**  
** COMM_init  
**  
** initialize COM-device  
**  
**---- Parameters -----  
**  
**  
**---- Returnvalue -----  
**  
** int : 0 if success  
** int : -1 if error  
**  
*****/
```



```

9  int COMM_init(char *c_device, int *fd)
10         {
11         int      id_com_dev; /* filename descriptor for the COM */
12         int      speed = 9600; /* speed for COM */
13         int      SerialControl;
14         int      Parity;
15         int      CharacterSize;
16         int      ioctlvalue;
17
18         /*-----
19
20         ** Open the COM-Device
21         */
22         id_com_dev = open (c_device, O_RDWR ,0);
23         if ( id_com_dev == ERROR )
24         {
25             perror ("Open_Port : Unable to open /dev/ ---");
26             return(-1);
27         }
28
29         /*
30
31         * Could not open the port
32         */
33
34         *fd = id_com_dev;
35
36         /*-----
37
38         ** setting the BAUD-rate to 9600 .....
39         */
40         if (-1 == ioctl (id_com_dev,FIOBAUDRATE , speed))
41         {
42             return(-1);
43         }
44
45         /*-----
46
47         ** Setting the Parity, the Stop bit .....
48         */

```

```

28         SerialControl = (CLOCAL | CREAD);
29         Parity = PARENB ;
30         CharacterSize = CS8;

if (-1 == ioctl (id_com_dev,SIO_HW_OPTS_SET,SerialControl | Parity |
    CharacterSize))
32     {
33         return(-1);
34     }

    /*-----
** Setting the new options of the serial port
    ** with setting the handshaking protocol
    */

35         ioctlvalue = ioctl (id_com_dev, FIOFLUSH, 0);
ioctlvalue = ioctl (id_com_dev,FIOSETOPTIONS,OPT_TANDEM &~
    OPT_MON_TRAP);
38     }

/******
**
** Main
**
** main program of RWCOM
**--- Returnvalue -----
** 0
*****/

39         int maincomRW(int argc, char *argv[])
40     {
41         char    pathserial0[] = "/tyCo/0"; /* Path of the port 2...*/
42         char    pathserial1[] = "/tyCo/1"; /* Path of the port 3...*/
43         int     ui_byte;          /* Number of bytes readed */
44         int     uo_byte;          /* Number of bytes writed */

```

```

45     char    *ur_buff; /* Buffer used to get the bytes from the serial port */
46     char    *uw_buff; /* Buffer used to set the bytes to the serial port */
47         int    fd1;    /* file descriptor for the first port */
48     int    fd2;    /* file descriptor for the second port */
49     int    len_mess; /* Length of message */

50         COMM_init(pathserial0, &fd1);
51         COMM_init(pathserial1, &fd2);
52         len_mess = 116;
53         ui_byte = 0;

54         ur_buff = (char*)malloc(116);
55         uw_buff = (char*)malloc(116);

56         while(1)
57         {
58             /* Wait until the all message arrive -----*/

59             while (ui_byte < len_mess)
60             {
61                 /*-----
62                 ** read from the COM-Device
63                 */
64                 ui_byte = read (fd1, ur_buff,len_mess);
65             }

66             /*-----
67             ** Copy the read buffer to the write buffer
68             */

69                 strcpy(uw_buff,ur_buff);

70             /*-----
71             ** send to the COM-Device
72             */

73                 uo_byte = write (fd2, uw_buff,len_mess);

```

```

65         }
66         close (fd1);
67         close (fd2);
68         return (0);
69     }

```

10.16 Annex G

```

/*****
**
** Project      : Diploma Thesis
**
**      The goal of this program is to initialise the serial port
**      and read from one serial port and write to another.....
**
*****/

/*-- Include files -----*/

#include <stdio.h>           /* Standard input/output definitions */
#include <string.h>          /* String function definitions */
#include <unistd.h>          /* UNIX standard function definitions */
#include <fcntl.h>           /* File Control definition */
#include <errno.h>           /* Error number definitions */
#include "vxWorks.h"
#include "ioLib.h"
#include <sioLib.h>

/*****
** COMM_init
**
** initialize COM-device
**
**---- Parameters -----
**---- Returnvalue -----
**
** int : 0 if success
** int : -1 if error
**
*****/

int COMM_init(char *c_device, int *fd)
{
    int      id_com_dev; /* filename descriptor for the COM */
    int      speed = 9600; /* speed for COM */
    int      SerialControl;
    int      Parity;
    int      CharacterSize;
    int      ioctlvalue;

```

```

/*-----
** Open the COM-Device
*/
id_com_dev = open ("/tyCo/1", O_RDWR ,0);
if ( id_com_dev == ERROR )
{
    perror ("Open_Port : Unable to open /dev/ ---");
    return(-1);
    /*
    * Could not open the port
    */
}

*fd = id_com_dev;

/*-----
**  setting the BAUD-rate to 9600 .....
*/
if (-1 == ioctl (id_com_dev,FIOBAUDRATE , speed))
{
    return(-1);
}

/*-----
**  Setting the Parity, the Stop bit .....
*/
SerialControl = (CLOCAL | CREAD);
Parity = PARENB ;
CharacterSize = CS8;

if (-1 == ioctl (id_com_dev,SIO_HW_OPTS_SET,SerialControl | Parity | CharacterSize))
{
    return(-1);
}

/*-----
**  Setting the new options of the serial port
**  with setting the handshaking protocol
*/

ioctlvalue = ioctl (id_com_dev, FIOFLUSH, 0);
ioctlvalue = ioctl (id_com_dev,FIOSETOPTIONS,OPT_TANDEM &~
    OPT_MON_TRAP);
}

/*****
**
** Main
**
** main program of RWCOM

```

```

**
**---- Returnvalue -----
** 0
***/
int maincom(int argc, char *argv[])
{
char pathserial0[] = "/tyCo/1"; /* Path of the port 2...*/
int us_byte; /* Number of bytes written */
int ui_byte; /* Number of bytes read */
char *us_buff0; /* Buffer used to set the bytes to the serial port */
char *us_buff1; /* Buffer used to set the bytes to the serial port */
char *us_buff2; /* Buffer used to set the bytes to the serial port */
char *us_buff3; /* Buffer used to set the bytes to the serial port */
char *us_buff4; /* Buffer used to set the bytes to the serial port */
char *us_buff5; /* Buffer used to set the bytes to the serial port */
char *us_buff6; /* Buffer used to set the bytes to the serial port */
char *us_buff7; /* Buffer used to set the bytes to the serial port */
char *us_buff8; /* Buffer used to set the bytes to the serial port */
char *us_buff9; /* Buffer used to set the bytes to the serial port */
char *ur_buff; /* Buffer used to get the bytes from the serial port */
int fd1; /* file descriptor for the serial port 2 */
int len_mess; /* Length of the send message */
int i;

COMM_init(pathserial0, &fd1);
len_mess = 116;
us_byte = 0;
ur_buff = (char*)malloc(52);

us_buff0 = (char*)malloc(116);
us_buff1 = (char*)malloc(116);
us_buff2 = (char*)malloc(116);
us_buff3 = (char*)malloc(116);
us_buff4 = (char*)malloc(116);
us_buff5 = (char*)malloc(116);
us_buff6 = (char*)malloc(116);
us_buff7 = (char*)malloc(116);
us_buff8 = (char*)malloc(116);
us_buff9 = (char*)malloc(116);

/*-----
** send to the COM-Device
*/

us_buff0="21C40C6D0E540C3EF9D40C3EF9D416418934282A5E341752F1A414420C4415251EB4179EF9D4189B
4394143DB224161C28F4173687241A1B4393";
us_buff1="21C411251EB41126E97411224DD41CA978D4278FDF3427D09374189ED91417522D0418A9BA5419A
0C494161A1CA41829994189AE1441CA978D3";
us_buff2="21C4143B22D4143374B4143DB2241C28F5C41819BA542940FDF4200CDD2418A916841999BA541AA
68724181B645419268724199ED9141F99BA53";

```

```

us_buff3="21C41740000417445A141741062422547AE427547AE41C270A342115D2F41991EB841A9B43941BC20
C44194FDF341A2BA5E41A83F7C422168723";
us_buff4="21C4192624D4192687241924BC641A845A142780A3D41C000004234000041AA916841B9ED91000000
0041A1126E41B41A9F41BB4BC6423548B43";
us_buff5="21C41AA937441AB020C41AABC6A41C5000042AAA5E34282A5E34279418941B8E35341C8000041D
9B43941B0D2F141C0EB8541CA978D48B43";
us_buff6="21C41C3020C41C3374B41C32F1A423548B44282A5E342513439425948B441CA687241DBE35341E816
8741C2916841D19BA541D9D9164280A3D73";
us_buff7="21C41DB49BA41DBB85141DB3D7041D9B64541EA916841F1B8514286A45A41DA916841E99BA541F
1B85141D453F741E1AE1441EA68724288A45A3";
us_buff8="21C41F39FBE41F3EF9D41F39DB2424148B44296A6E942BEA5E342A8764541E9F5C241F9A3D74204
DC2841E1A3D741F0F5C241FA126E428CA6663";
us_buff9="21C4205F4BC420615814205F6C842614BC6425948B4424C178D42B4000041F9A3D742068312420E9168
41F2020C4200FBE742051374429007AE3";

```

```

/* Wait until the all message arrive -----*/

```

```

if (ui_byte < len_mess)

```

```

{
    /*-----
    ** read from the COM-Device
    */
    ui_byte = read (fd1, ur_buff,len_mess);
}

```

```

us_byte = write (fd1, us_buff0,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff1,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff2,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff3,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff4,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff5,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff6,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff7,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff8,len_mess);
delayedEval(500);
us_byte = write (fd1, us_buff9,len_mess);
delayedEval(500);

```

```

close (fd1);

```

```

return (0);

```

```

}

```

11 Some repairs on the sensor card and first step integration

Inhaltsverzeichnis

I. Kurz Blick4

II .Einleitung..... 5

III.Thema der Aufgabe.....6

IV. Überblick über de bisherige gemachte Arbeit6

V. Aufgabestellung.....	7
1. Frontdeckel für die Karten.....	8
2. Fehlende Montagearbeiten an der Mikrocontrollerplatine (Meßdatenverarbeitungsplatine) und der Meßdatenerfassungsplatinen (Sensordatenplatinen).....	9
2.1 Fehlende Arbeiten bezüglich Montage an der Mikrocontrollerplatine und deren Anbindung an die Meßdatenerfassungsplatine (Sensordatenplatine).....	9
2.2 Fehlende Arbeiten bezüglich der Montage der Meßdatenerfassungsplatinen (Sensorplatinen).....	11
3. Hardwaretest der Meßdatenverarbeitungsplatine.....	15
3.1 Versorgungsspannungs-Test.....	16
3.2 Port-Tests.....	17

4. Anschluß der Phyttec-Testplatine (Ersatz für Meßdatenverarbeitungsplatine) und Sensorplatine an den Entwicklungsumgebungsrechner und Softwaretest des IMU-Gesamtsystems.....	19
4.1 Anschluß der Phyttec-Testplatine mit der Sensorplatine was die PINs bedeuten und Anschluß an die Versorgungsspannung.....	19
4.2 Anschluß der Phyttec-Testplatine (KitCON-167-Board) an den Entwicklungsumgebungsrechner.....	21
4.3 Softwaretest mit μ Vision von KEIL.....	22
5.zusammenfassung und Ausblick.....	27
ANHANG A: Spezifikation des Meßdatenaufbereitungs-Board (Mikrocontroller-platine).....	27
ANHANG B: Spezifikation des Meßdatenaufnahme-Boards (Sensorenplatine).....	28
ANHANG C: Steckerbelegung der Meßdatenaufnahme in xy-Richtung.....	29
6.Literatur.....	31

I. Kurz Blick

Das Solarluftschiff bei der Grundsteinlegung zur Cargolifter-Werfthalle. Lotte ist ein Solargetriebenes Luftschiff, das 1993 an der Universität Stuttgart entwickelt wurde. Auf der Oberseite des ferngelenkten Luftschiffes befinden sich 7 m² Solarzellen, die eine Leistung von 1123 Watt bringen. Damit schafft Lotte eine Höchstgeschwindigkeit von 46 km/h.

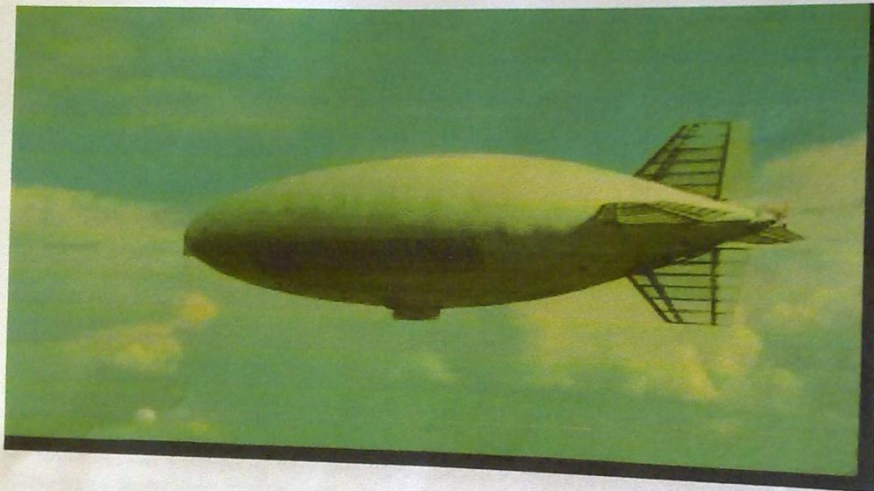


Bild 1 : Solarluftschiff " Lotte"

Im Verein für alternative Energieforschung an der Universität Karlsruhe (VaEf) wird seit dem Jahr 2000 ein Messdatenerfassungs-Luftschiff "alternative Lotte" entwickelt, mit dessen Hilfe es möglich ist, präzise Windstärke- und Solareinstrahlungsdaten in Abhängigkeit vom Ort zu erhalten. Dadurch wird es möglich, z.B. den optimalen Standort von Windrädern und Solarzellen auszuloten. Das Luftschiff soll ca. 8 m lang sein. Dieses Projekt läuft in Zusammenhang mit der Universität Karlsruhe, der Fachhochschule Karlsruhe und der Universität Stuttgart.

II. Einleitung

Das inertielle Navigationssystem ist ein autonomes System, das im Gegensatz zum Satelliten oder anderen Funknavigationssystemen keine externen Informationen benötigt. Mittels eines inertiellen Navigationssystems ist man in der Lage:

- Beschleunigungen und Drehgeschwindigkeiten zu messen
- Lagewinkel zu bestimmen
- Position und Geschwindigkeit zu berechnen - auf dynamisch bewegten Fahrzeugen.

Und ist möglich

- bei jedem Wetter,
- zu jeder Zeit,
- an jedem Ort der Erde

Aufgrund seiner Bauweise wird das vorliegende Inertialnavigationssystem den Strapdown-Systemen zugeordnet. Als Strapdown-Systeme werden inertielle Navigationssysteme bezeichnet, bei denen die Beschleunigungssensoren fest an das Fahrzeug montiert sind. Die Meßachsen der Sensoren sind hierzu in der Regel parallel zu den Hauptachsen des Fahrzeugs ausgerichtet. Die Sensorik für ein solches Strapdown-Inertialnavigationssystem besteht aus Beschleunigungsmessern, die die translatorischen Beschleunigungskomponenten erfassen, den Gyroskopen (Kreiseln), die die Drehwinkelbeschleunigungen erfassen. Diese Messgrößen werden ständig in Richtung der körperfesten Achsen des Fahrzeugs erfasst. Aus der Messung der Drehwinkelbeschleunigungen lassen sich über ein Differentialgleichungssystem die Winkelinkremente (Azimut-, Nick- und Rollwinkelinkremente) und die Lagewinkel (Azimut-, Nick- und Rollwinkel) laufend berechnen. Aus diesen Werten bestimmt ein Navigationsrechner die Lage des Fahrzeugs bezüglich des festgelegten Navigationskoordinatensystems. Die Strapdown-Technik stellt durch die direkte Kopplung der Sensoren an die Fahrzeugdynamik hohe dynamische Anforderungen an die Sensoren selbst.

III. Thema der Aufgabe

- Im Projekt "alternative Lotte" wurde ein inertiales Meßsystem IMU (Inertial Measurement Unit) entwickelt.
- Die Tests, der Aufbau eines entsprechenden mechanischen Gehäuses und die Anbindung der getesteten IMU am Bordrechner des Messdatenerfassungsluftschiffs "alternative Lotte" sind Themen dieser Arbeiten.

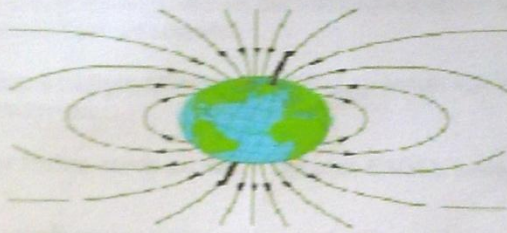


Bild 2: Magnetfeld

IV. Überblick über die bisherige gemachte Arbeit

Es wurde eine teils funktionsfähige IMU fertig gestellt. Lediglich der Mikrokontroller zur Messwertaufbereitung muss noch auf Funktionsfähigkeit getestet werden. Die ersten Testergebnisse sind gemacht worden. Bei diesem Testen handelt es sich um einfachen Tests bei ruhenden Sensoren. Es ging noch nicht um die Fehlerkorrektur im Rahmen einer temperaturänderung der Sensoren. Die Kompaßsensoren wurden noch nicht getestet.

Das Mikrocontrollerboard konnte noch nicht getestet werden, da einige Probleme bei Inbetriebnahme des Mikrocontrollers auftraten. Diese Probleme wurden zwar behoben, aber es konnten keine Softwaretests mehr durchgeführt werden. Um das Gesamtsystem zu beurteilen, müssten noch zahlreiche weitere Test unternommen werden. Mit laufender Betriebszeit würden sich jedoch die Abweichungen vor allem bei den Kreiseln akkumulieren. Es müßte also in regelmäßigen, nicht zu großen Abständen wieder ein aktualisierter, korrekter Referenzwert (z.B. durch ein GPS-System) die IMU reinitialisieren. D.h. ein nächster Schritt wäre die Weiterentwicklung zu einem Hybridsystem.

V. Aufgabenstellung

Information über die Messdatenaufbereitungsplatine

- >> Die Meßdatenaufbereitung ist auf einer vierschichtigen Euro-Platine realisiert.
 - Die erste der vier Schichten ist die Bestückungsseite.
 - Eine Zwischenschicht ist die Masse.
 - Eine zweite Zwischenschicht wird mit der Versorgungsspannung verbunden.
 - Die vierte Schicht dient als Lötseite.
- >> Die Fertigung der 4-Lagigen Mikrocontroller-Platine wurde von der Firma PCB-POOL übernommen.
- >> Der Entwurf der Platine wurde mit dem Programm EAGEL (vers. 3.55) durchgeführt.
- >> Für die Mikrocontrollerprogrammierung wurde der Programm μ -vision (ver.2.03) der Firma Keil benutzt. Es handelt sich um eine Shareware-Version für begrenzte Code-Größe von 16 KByte und wird mit dem Programm , c ' programmiert.

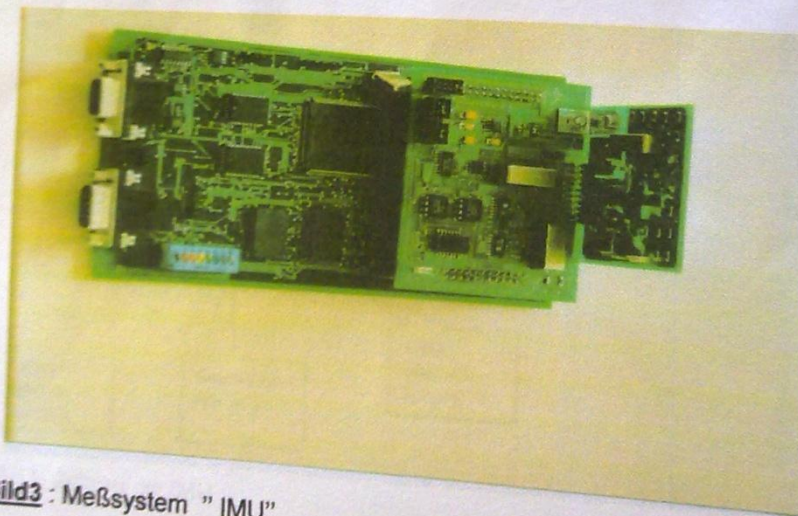


Bild3 : Meßsystem " IMU"

1. Frontdeckel für die Karten

- Frontgehäuse ist als Standart gewählt worden 19 Zoll-Gehäuse.
- Die Frontdeckel für die Meßdatenverarbeitungs-Karte + Sensorkarten-Aufsatz wurden an die Deckplatte angeschraubt.
- Die Frontdeckel hat ein Schalter für die Betriebsfunktion, eine Schnittstelle für die Versorgungsspannung, eine LED-Anzeige für die Betriebsanzeige, eine **Schalter für die Kalibrierungsmodus (optional)** und eine LED-Anzeige für die Kalibrierungsmodus-Anzeige beinhalten.

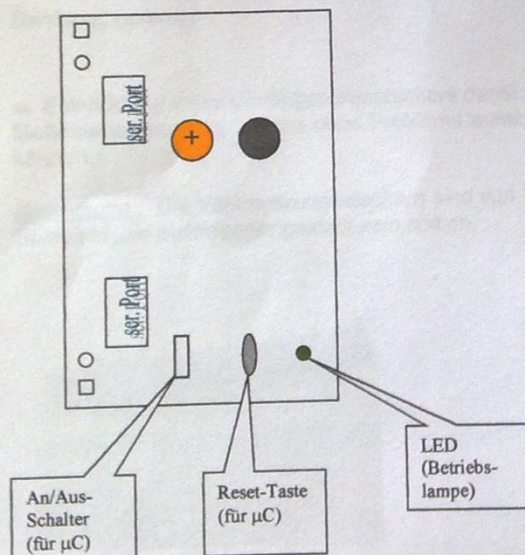


Abb.1: Gehäuse der IMU

Bild 4: Stiftleiste 2-Reihig

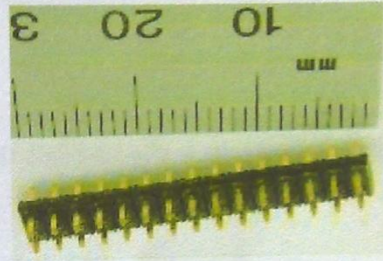
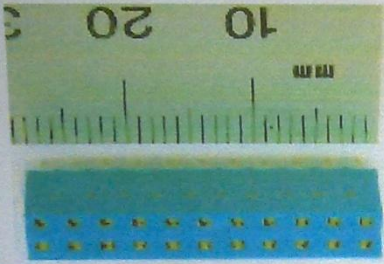


Bild 5: Dubox 2-Reihig



Bemerkung: Die Verlängerungssteckern sind von Typ : Dubox 2-Reihig und Stiftleiste , die aufeinander geklebt sein sollten.

>> Entwicklung eines Verlängerungssteckers damit die Sensorplatine und die Messdatenerfassungs-Platine ohne Probleme aufeinander zusammengesetzt werden können.

Danfors, gefertigt.

Bemerkung: Da die Steckplätze für die ROM-Bausteine falsche Format haben wurden die ROM-Bausteine auf die Steckplätze angepasst (Pins nach Innen biegen) Diese Aufgabe wurde von Herrn. Horst Kohle, Entwickler bei der Firma

SMD Löttechnik löten.


>> ROM-Bausteine (K6T1008 C2E-GB70) auf der Messdatenerfassungsplatine mit

>> Verbindung der Reset-Taste (JP 9) mit dem Frontdeckel .

>> Verbindung einer Betriebsanzeiger (LED Grün) (JP 7) mit dem Frontdeckel .

21. Fehlende Arbeiten bezüglich der Montage der Mikrocontrollerplatine und deren Anbindung an die Messdatenerfassungsplatine bzw. Sensordatenplatine:

2. Fehlende Montagearbeiten an der Mikrocontrollerplatine (Messdatenverarbeitungsplatine) und der Messdatenerfassungsplatinen (Sensordatenplatinen)

 <p>FH 1878 GEGR.</p>	<p>Fachhochschule Karlsruhe Hochschule für Technik</p>	<p>Fahrzeugtechnologie</p>
---	---	-----------------------------------

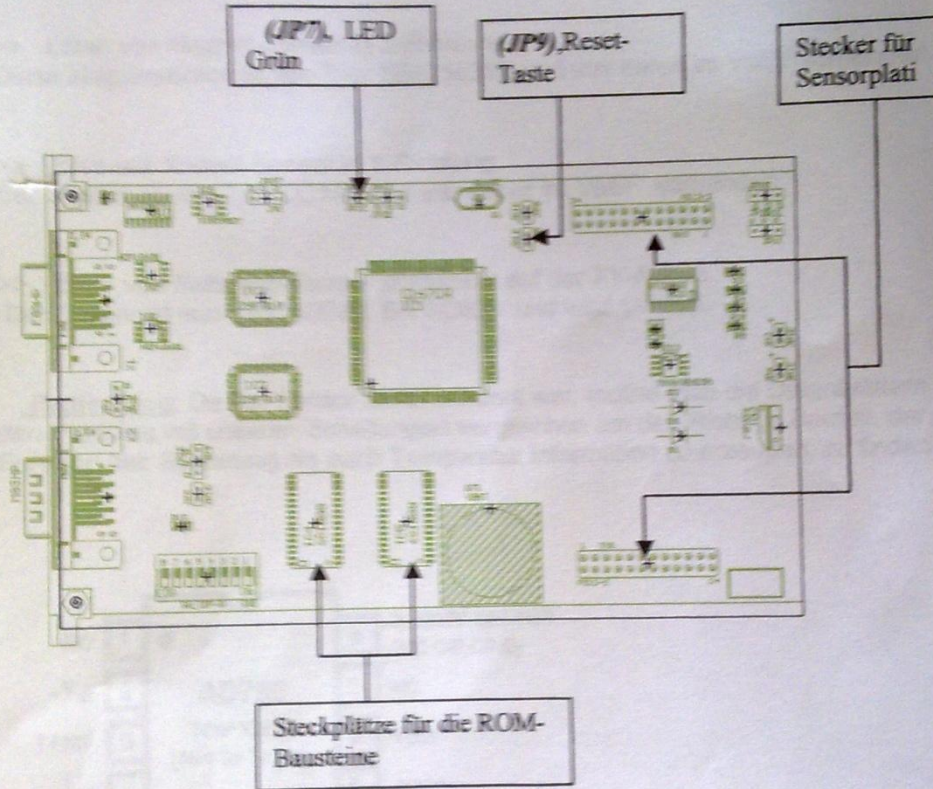


Abb.2: : Fehlende Arbeit bezüglich der Montage an der Mikrocontrollerplatine

2.2. Fehlende Arbeiten bezüglich der Montage der Maßdatenerfassungsplatinen (Sensorplatinen).

>> Löten von Magnet-Sensor in Z-Richtung.
Dieser Magnetsensor ist von Typ: **HMC1021S** und war bereit im VaEF vorhanden.

>> Löten von Kreisel-Sensor in Y-Richtung.
Der Sensor ist von Typ: **CG16D** und war bereit im VaEF vorhanden.

>> Löten von Referenz-Temperatursensor auf der XY-Achse.
Der Sensor ist von Typ: **AD780 BR SOIC8** und wird gekauft.

Bemerkung: Da der Sensor nicht bekannt war, mußte man die Datenblätter von den Produkte mit unseren Schaltungen vergleichen um den Richtige Bauteil, der die Funktion hat: Spannung als auch Temperatur Information zu erzeugen, zu finden.

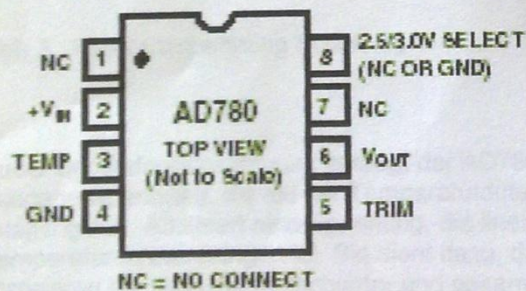


Bild 6 :Referenzspannungserzeuger

Dieser Bauteil wurde von **Spröle Elektronik GmbH** bestellt

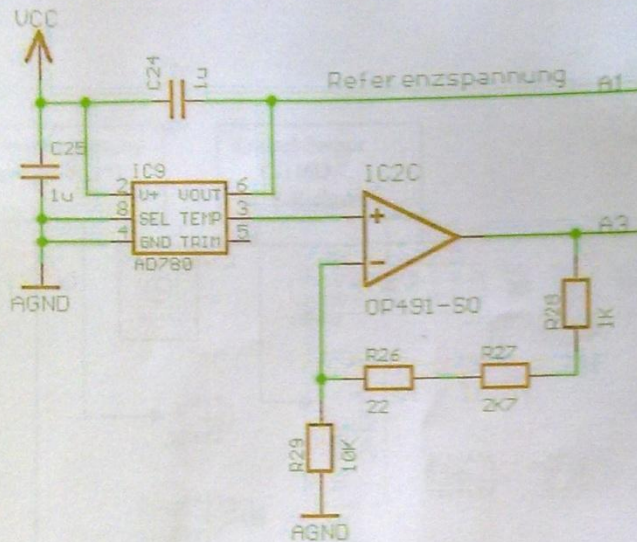


Abb.3: Referenzspannung Schaltung

Außer der Referenzspannung erzeugt der AD780 noch eine weitere Ausgangsspannung, die mit der Temperaturdifferenz korreliert (in Abb.3 ist dies der Ausgang A3). A3 liefert eine Spannung, die linear abhängig von Temperaturschwankungen ist. Sie dient dazu, die Nichtlinearität der Sensoren zu korrigieren mit Hilfe der Datenblätter und gesammelten Daten.

Bei Zimmertemperatur von 25 °C ist die Temperatureingangsspannung von 560 mV (laut Datenblatt) mit einer "temperature sensitivity" von 1,9 mV / °C. Da die Referenzspannung 2,5 Volt beträgt und der AD-Wandler des Mikrocontrollers eine Auflösung von 10 Bit hat, hat das System eine Resolution von:

$$\frac{2,5 V}{2^{10}} = 24,4 mV .$$

Um die Erfassung der Signale zu vereinfachen, wurde ein Verstärker von 2,47 gewählt (siehe Abb.3), damit sind jede Temperaturänderungen gewährleistet, daß sie erfasst werden können. Eine ± 1 °C Temperaturänderung verursacht dann am Ausgang eine Spannungsänderung von $\pm 4,7$ mV. Während der Zimmertemperatur die Ausgangsspannung 1,48 V beträgt.

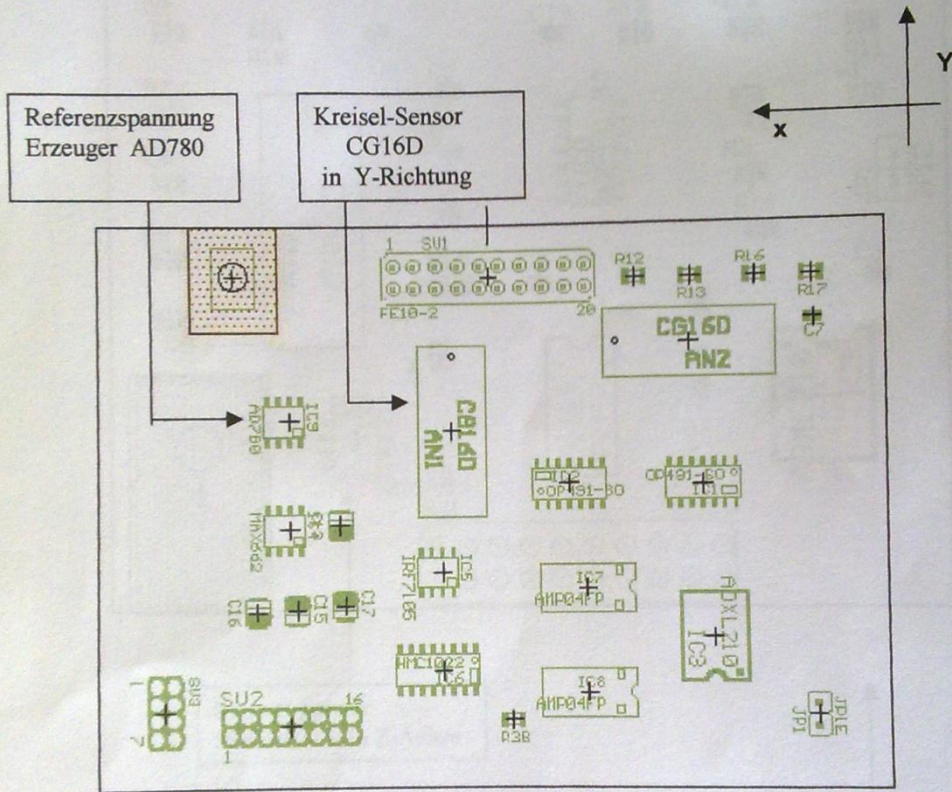


Abb.4: Fehlende Arbeit bezüglich der Sensorplatinen auf der XY-Achse

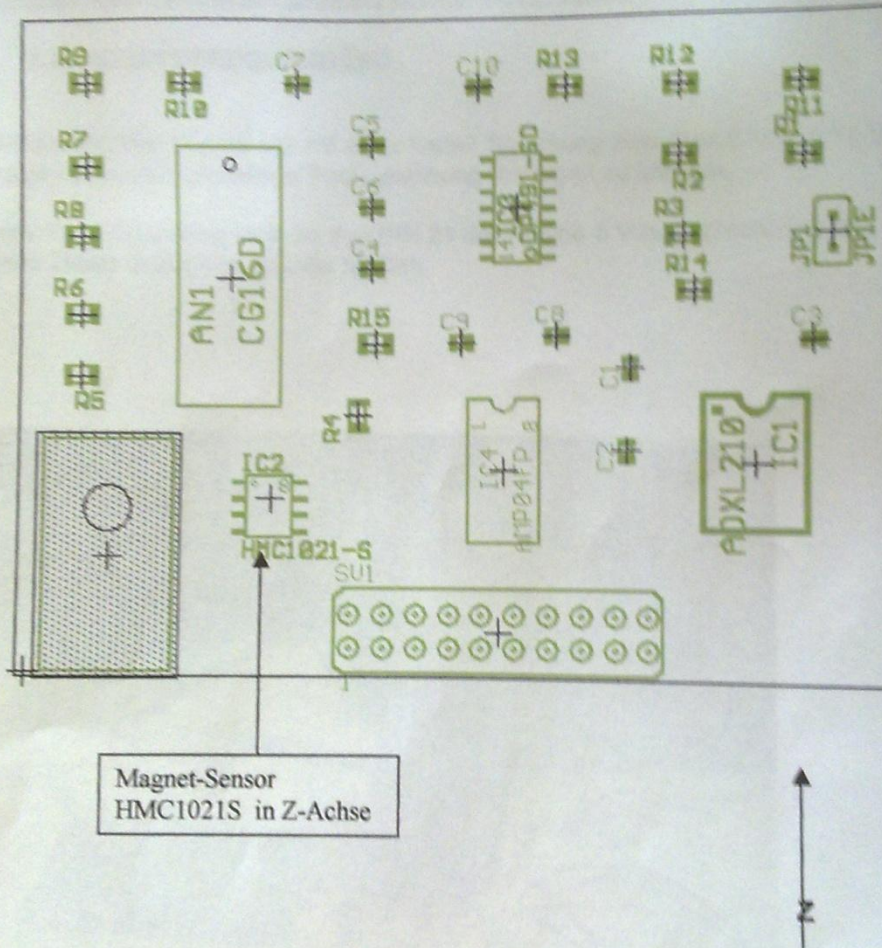


Abb.5 Fehlende Arbeit bezüglich der Sensorplatinen auf der Z-Achse

3. Hardwaretest der Meßdatenverarbeitungsplatine

3.1 Versorgungsspannungs-Test

Die Mikrocontroller-Platine soll mit einer festen Spannung zwischen 8 V und 12 V versorgt werden um die nötige Test-Spannung festlegen zu können.

Die erwartete Spannung muß an den PIN 21 der Platine 5 V betrachten und die Betriebs Diode muß eingeschaltet bleiben.

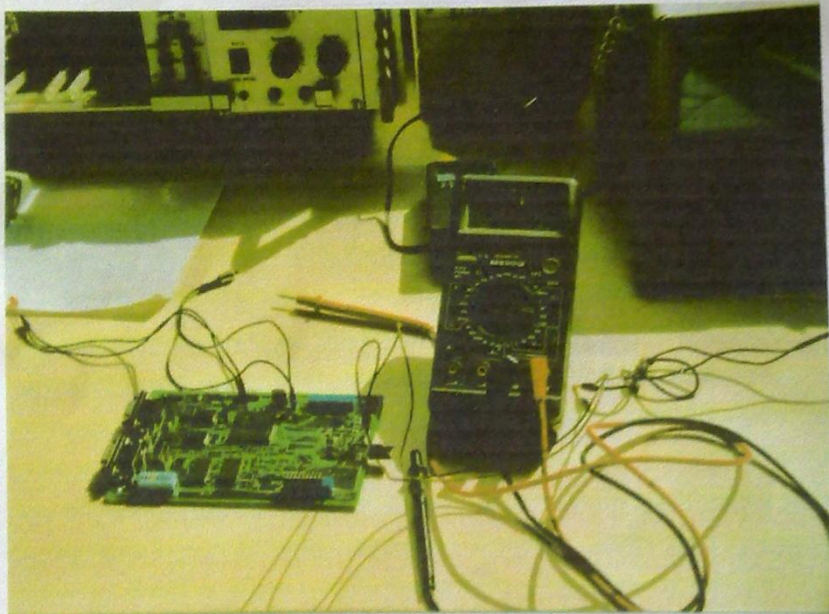


Bild 7: Versorgungsspannungs-Test

>> Test Ergebnisse

Angelegte Spannung	LED	gemessene Spannung an PIN 21 an der Stecker SV2
7.5 (V)	AN	4.1 (V)
9 (V)	AN	5,2 (V)
12 (V)	AN	5,9 (V)

>> Bemerkung

Nach dem Test hat sich festgestellt, dass die notwendige Paltinenspannung 9 Volt sein soll .da wir als Ausgang Spannung genau 5 Volt haben wollen.

3.2 Port-Tests

Bei diesem Test werden die Port-Ausgänge bzw. Port-Eingänge der Mikrocontroller-Platine getestet.

Dies ist realisiert worden, in dem man eine Testsoftware programmiert hat. Bei diesem Test sind die Portausgänge (P2 (hat 16 I/O-Kanäle) (in SV2) und P7 (hat 8 I/O-Kanäle) (in SV1), getestet.

Bemerkung: (SV1) und (SV2) sind Steckplätze für die Sensorenplatinen. Die Programmierung des Mikrocontrollers C166 erfolgt unter Entwicklungsumgebung μ -Vision der Fa. Keil. Es handelt sich um eine Shareware-Version, mit denen Programme bis zu 8 KByte Codegröße entwickelt werden können.

Man findet in μ -Vision, Version 2 die grundlegenden Elemente einer modernen IDE:

- ➔ Einen speziell angepassten Texteditor.
- ➔ Einen ANSI-C-Compiler (C166-ANSI-C-Compiler).
- ➔ Einen Assembler (A166-Assembler).
- ➔ Einen Linker/Lokator (L166).
- ➔ Einen Debugger (μ Vision2-Debugger).

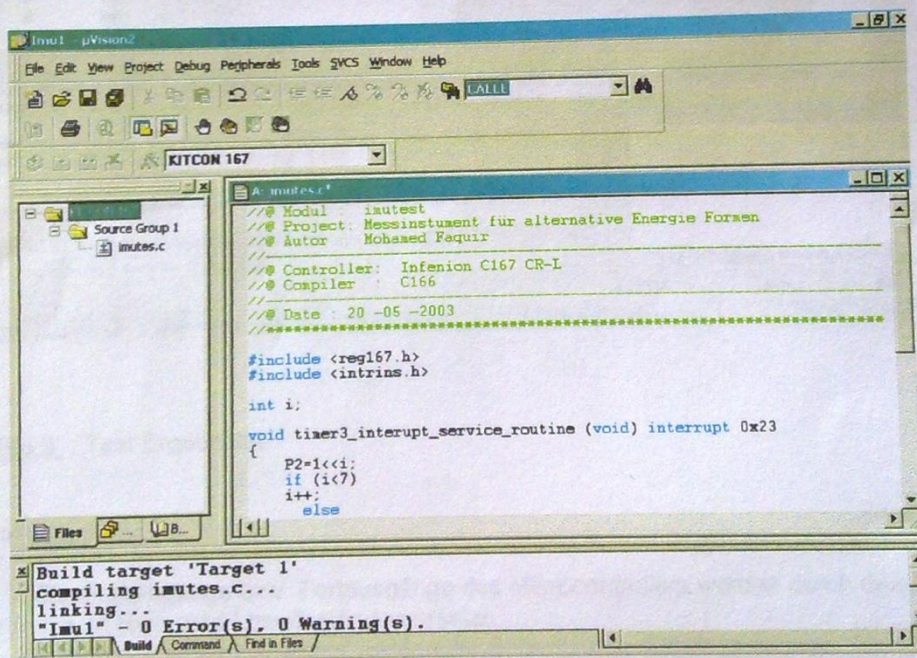


Bild 8 : Softwareentwicklung

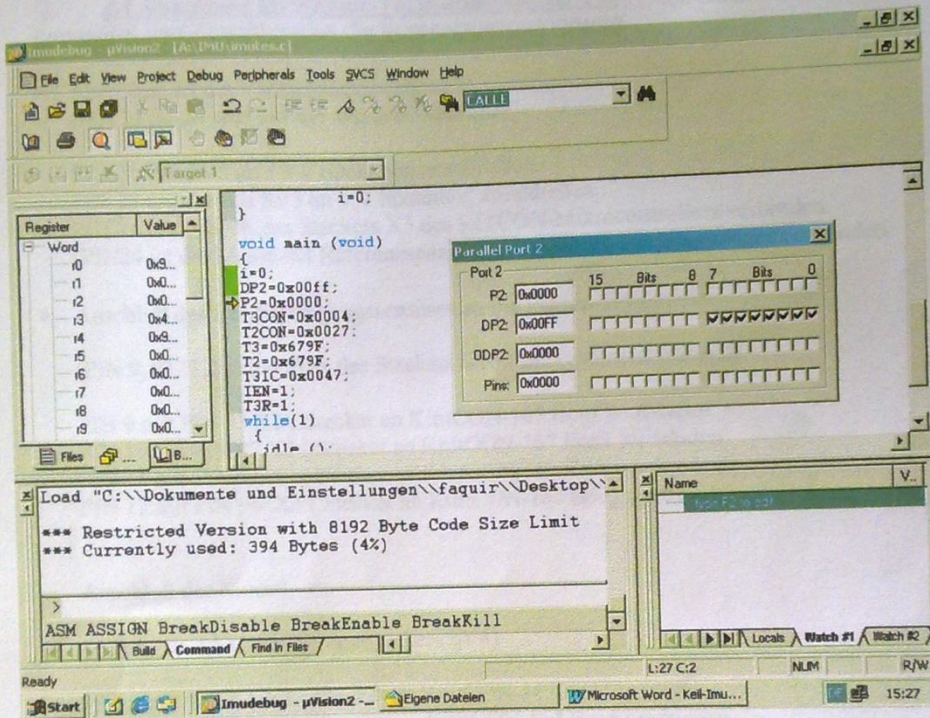


Bild 9 : Test Ergebnisse

>>Bemerkung

Die Porteingänge bzw. Portausgänge des Mikrocontrollers werden durch diese software in High- und Low-Zustände versetzt.
Mit einem Spannungsmesser werden die Spannungen dieser gemessen.

4. Anschluß der Phyttec-Testplatine (Ersatz für Meßdatenverarbeitungsplatine) und Sensorplatine an den Entwicklungsumgebungsrechner

4.1 Anschluss der Phyttec-Testplatine mit der Sensorplatine (was die PINs bedeuten) und Anschluss an die Versorgungsspannung

- Anschluss an die Versorgungsspannung:
(PIN 21,23,24 sind auf der x-y-Sensorplatine)
 - PIN 21 von SV5 an +5 V Spannung anschließen.
 - PIN 23 und 24 von SV5 an 0 V Spannung anschließen.
 - PIN 24 mit PIN 66 des Steckers X3 des KITCON-Mikrocontrollers verbinden.
(PIN24 ist die Masse der Referenzspannung, PIN66 (VGND) wird damit verbunden)

- Anschluß der Beschleunigungssensoren an die Phyttec-Testplatine:
(PIN 9,10,11,12 sind PINs des Steckers SV5 (auf x-y-Sensorplatine))
 - PIN 9 mit PIN 93 X3 (Stecker an KititCON-167 Bord)verbinden.
 - PIN 10 mit PIN 97 X3 (Stecker an KititCON-167 Bord)verbinden..
 - PIN 11 mit PIN 94 X3 (Stecker an KititCON-167 Bord)verbinden.
 - PIN 12 mit PIN 98 X3 (Stecker an KititCON-167 Bord)verbinden.

- Anschluß der Kreisel
(PIN 9, 10, 12 sind PINs des Steckers SV4)
 - PIN 9 mit PIN 69 X3 (Stecker an KititCON-167 Bord)verbinden.
 - PIN 10 mit PIN 73 X3 (Stecker an KititCON-167 Bord)verbinden.
 - PIN 12 mit PIN 70 X3 (Stecker an KititCON-167 Bord)verbinden.

- Anschluß des Kompaß
(PIN 14, 15, 16 sind PINs des Steckers SV4)
 - PIN 14 mit PIN 75 X3 (Stecker an KititCON-167 Bord)verbinden.
 - PIN 15 mit PIN 72 X3 (Stecker an KititCON-167 Bord)verbinden.
 - PIN 16 mit PIN 76 X3 (Stecker an KititCON-167 Bord)verbinden.

- Anschluss des Temperatur-Sensors
PIN 17 mit PIN 77 X3 (Stecker an KititCON-167 Borde)verbinden.

>>Bemerkung

Nachdem es festgestellt wurde, daß es noch Störungen bei der Mikrocontroller-Platine aufgetaucht sind, wurden die Datenerfassung mit dem Mikrocontroller C167 der Firma PHYTEC getestet.

Diese ist ein Standard Testboard von der Firma **PHYTEC**, der dieselben Eigenschaften hat, wie das entwickelte Mikrocontroller-Board.

Dieses Testboard wurde mit der Meßdatenerfassungsplatine (Sensorplatine) verbunden.

Die Daten wurden erfasst nach dem man die Lage und Position der Sensorplatine geändert hat.

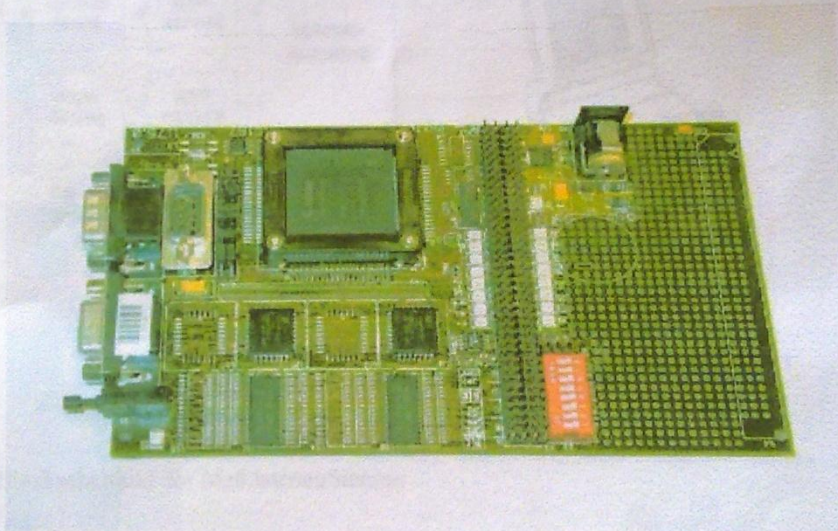


Bild 10 : Phytec-Test Platine (KitCon-167)

4.2 Anschluss der Phytec-Testplatine (KitCON-167-Bord) an den Entwicklungsumgebungsrechner

Über serielle Schnittstelle wird Port P1 der Mikrocontroller KitCon-167 und die Sensor-Platine mit dem Umgebungsrechner verbunden

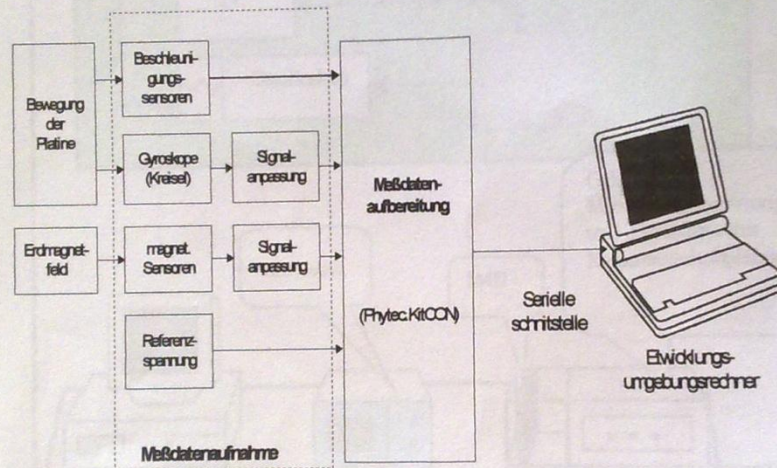


Abb. 6: Blockschaltbild der Meßdatenaufnahme

Die Platinen sind gegen statische Elektrizität besonders empfindlich. Deswegen ist es wichtig, diese Baugruppe auf einen geeigneten Platz während der Experimente zu stellen.

4.2 Softwaretest mit dem Programm μ -vision von KEIL

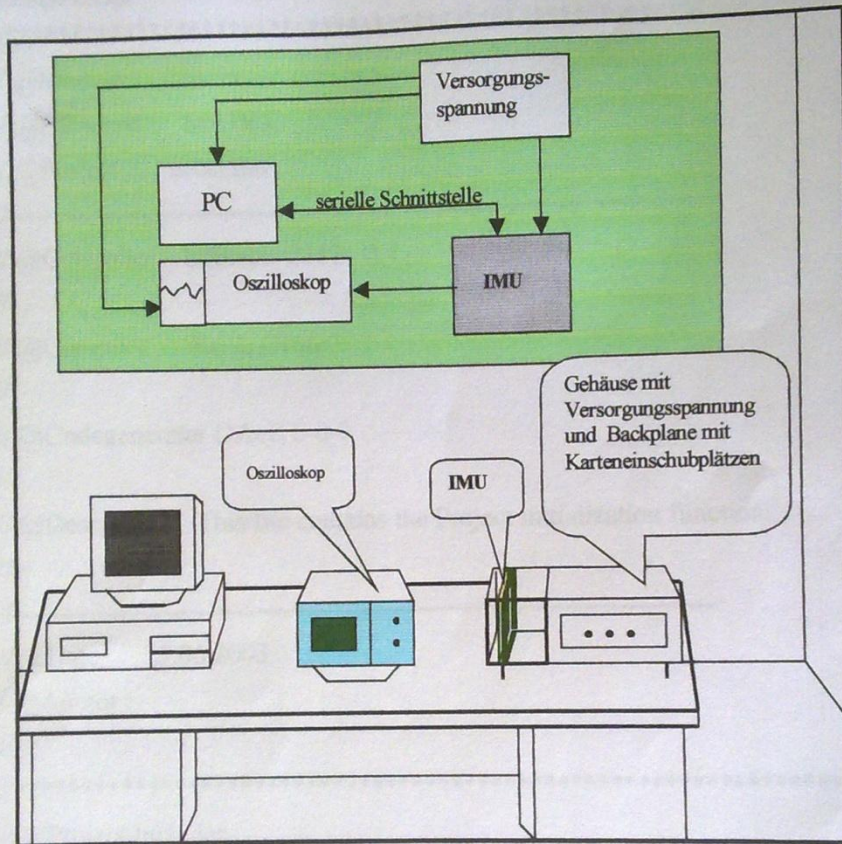


Bild 11 : Test im Prüfstandslabor

Bild 11 zeigt die bei den Experimenten und der Entwicklung des Programms für die IMU benutzten Geräte.

Der PC ist mit der IMU über eine serielle Schnittstelle (RS232) verbunden. Um die Signale sichtbar machen zu können, wird ein Oszilloskop benutzt.

Im Gehäuse für das System ist eine Versorgungsspannungserzeugungseinheit (rechte Seite der Abbildung) eingebaut, um auf Batterien verzichten zu können.

Beispiel-Test

```
*****
// @Module      Main
// @Filename     MAIN.C
// @Project      accel.dav
//-----
// @Controller   Infineon C167CR-L
//
// @Compiler     KEIL C166
//
// @Codegenerator DAVe 2-0-0
//
// @Description   This file contains the Project initialization function.
//
//-----
// @Date         25.06.2003
// @Auotor :
//
//*****
// @Project Includes
//*****
#include <reg167.h>
#include <intrins.h>
#include "MAIN.H"
//*****
// @Global Variables
```

```
/**
 *
 */
// @Function void Project_Init(void)
//-----
// @Description This function initializes the microcontroller.
//
//-----
// @Returnvalue none
//-----
// @Parameters none
//-----
// @Date 25.06.2003 12:45:56
//
/**
 *
 */
void Project_Init(void)
{
    // initializes the CAPCOM 1 peripheral
    CC1_vInit();

    // initializes the general purpose timer unit 1
    GT1_vInit();
    // globally enable interrupts

    IEN = 1;
}
/**
 *
 */
```



```
// @Function void main(void)
//-----
// @Description This is the main function.
//-----
// @Returnvalue none
//-----
// @Parameters none
//-----
// @Date 25.03.2002 12:45:56
//*****
void main(void)
{
  /*-----
  ** init all necessary function
  */
  Project_Init();
  GT1_vStartTmr(TIMER_3);
  /*-----
  ** loop forever
  */
  while (1)
  {
    _idle_();
  }
}

/*---- end file -----*/
```

```

***** REMOTE MEASUREMENT RECORDER using C167 *****
| This program is a simple Measurement Recorder. It is based
| on the SBC167 CPU and records the stat
Sensor 1 465 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 466 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 468 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64419, T1y 1148, T2 113
Sensor 1 464 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 464 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 461 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 464 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1149, T2 113
Sensor 1 464 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1149, T2 113
Sensor 1 461 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1149, T2 113
Sensor 1 464 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 463 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 466 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64420, T1y 1148, T2 113
Sensor 1 464 -> T1x: 0, T1y 0, T2 0 Sensor 1 ->T1x: 64419, T1y 1148, T2 113
|
| 210 (Accelerometer and the voltage on the Gyro.
+-----+-----+-----+
| command | syntax | function |
+-----+-----+-----+
| Show    | D      | Show current position |
| Clear   | R      | Set Position to Null  |
| Start Calib. | C      | Start the system in Calibration mode |
| Stop Calib. | F      | Stop Calibration mode |
| Start   | S      | start measurement recording |
+-----+-----+-----+
Command: |

```

Bild 12 :Test Ergebnisse

In diesem Versuch sollen die Daten erfasst nachdem man die Lage und Position der IMU Auf der X-Achse und Y-Achse geändert hat.
Die Periodendauer T1 dauert 1ms. In dem Test wurde alle 100ms abgetastet. D.h. jeder hundertste Wert des Sensors wurde erfasst.

5. Zusammenfassung und Ausblick *Anhänge*

ANHANG A

Spezifikation des Meßdatenaufbereitungs-Boards (Mikrocontroller-Board)

Type des Mikrocontrollers: SAB 80C167 CR

Quarz / Oszillator 5 MHz = Taktfrequenz 20 MHz

16 Bit-Modus, gemultiplext

256 KB RAM, durch Batterie gepuffert

256 KB FLASH RAM

Ein-/Ausgänge

16 x TTL-Ausgang

16 x AD-Wandler mit 10 Bit Resolution

1 x serielle Schnittstelle mit RS232

1 x CAN

Real Time Clock

Versorgung 8 – 12 Volt

EURO-Format (160 mm x 100 mm)

ANHANG C

Steckerbelegung der Meßdatenaufnahme in xy-Richtung

SV1

PORTNUMBER	FUNCTION	PORTNUMBER	FUNCTION
1	VCC	11	
2	AGND	12	
3		13	Z (G-OUT)
4		14	
5		15	Z (A-OUT)
6	S/R+	16	OFFS-IN
7		17	Z (C-OUT)
8	VREF	18	
9		19	OFFS-OUT
10	Z (Ay-OUT)	20	DGND

SV2

PORTNUMBER	FUNCTION	PORTNUMBER	FUNCTION
1	VCC	9	NC
2	Z(Ax-Out) = Z (A-OUT)	10	X (G-OUT)
3	VREF	11	Z (C-OUT)
4	Z(Ay-Out)	12	Y (G-OUT)
5	VTEMP	13	Y (C-OUT)
6	X (A-OUT)	14	Z (G-OUT)
7	S/R+	15	X (C-OUT)
8	Y (A-OUT)	16	DGND

SV4

PORTNUMBER	FUNCTION	PORTNUMBER	FUNCTION
1		13	
2		14	Z (C-OUT)
3		15	Y (C-OUT)
4		16	X (C-OUT)
5		17	VTEMP
6		18	
7		19	
8		20	
9	Y (G-OUT)	21	
10	X (G-OUT)	22	
11		23	
12	Z (G-OUT)	24	

SV5

PORTNUMBER	FUNCTION	PORTNUMBER	FUNCTION
1	S/R+	13	
2		14	Z (C-OUT)
3		15	Y (C-OUT)
4		16	X (C-OUT)
5		17	VTEMP
6		18	
7		19	
8		20	
9	Z (A-OUT)	21	VCC
10	Z(Ay-Out)	22	VREF
11	X (A-OUT)	23	DGND
12	Y (A-OUT)	24	RGND

6.Literatur

- [1] Martin ,Horauer : "Mikrocomputer –EU 384. 115 " Institut für Computertechnik Wien .
- [2] DOROBANTU, Raul: "Simulation des Verhaltens einer low-cost Strapdown IMU unter Laborbedingungen"; Technische Universität München; München.
- [3] MOSER, Luethi/ MOSER, Thomas: "Low Cost Inertial Navigation System"; Electronic Laboratory of Swiss Federal Institute of Technology, Zurich. 2000
- [4] KitCON-167 Hardware-Manual (Hand Buch der Phytoc Test-Platine)

-
- [1] Technische Datenblätter von Analog Device von AD 780
<http://products.analog.com/products/imfo.asp?product= AD780>
<http://www.electronic-engineering.ch/study/ins/schematics/AD780.pdf>
 - [2] Application note from Honeywell Inc. von
<http://www.ssec.honeywell.com/magnetic/datasheets/sae.pdf>
 - [3] zugekaufte Teile von:
<http://www.farnell.de>
<http://www.rs-components.de>

Fachhochschule Karlsruhe
Fachbereich Mechatronik Studiengang Fahrzeugtechnologie



Test und Integration eines inertialen
Meßsystems für eine experimentelle
Umgebung

*Praktikum
von
Mohamed Faquir*

Danksagung

An dieser Stelle möchte ich mich ganz herzlich bedanken bei Dr. Ottmar Beucher von der Fachhochschule Karlsruhe für die nette Betreuung während des Praxissemesters.

Recht herzlichen Dank an alle Assistenten des Studienganges Fahrzeugtechnologie die mir bei technischen Problemen zur Seite standen.

Mein besonderer Dank gebührt den Herrn Samir M^orad, der mein Praktikum an der VaEF ermöglicht hat, andererseits bei Herrn Jamal Khalil für seine Unterstützung.

Für die gute Zusammenarbeit und die Spannende Atmosphäre an der VaEF möchte ich mich herzlich bedanken bei meinem Betreuer Herrn ,Heddad.

Herzlichen Dank auch an all jene, die ich vergessen habe, zu erwähnen.

**1. Praxissemester
SS2003**

Von: 03.03.2003 bis: 31.07.2003

Name: Faquir

Vorname: Mohamed

Anschrift: Franz-Hirth-str 8
76316 Malsch

Matrikelnummer: 012294

Praktikumort: **Technologie Fabrik Karlsruhe**
Haid-und-Neu-str 7 D-76136 Karlsruhe

Abteilung: *Institut* **Verein für alternative Energieforschung** *an der*
(VaEF) *Univ. Karlsru.*

Geschäftsführer : Herr Dip.-Ing.-Inform. Samir Mourad

Betreuer : *Hedad*
Herr Hedad

12 Integration

Based on *Mohammed Yassir Mikou*, "Integration und Test eines Board-Computers für ein alternatives Luftschiff", Master Thesis (Diplomarbeit)

Supervisors:

Samir Mourad, Verein für alternative Energieforschung

Prof. Dr. –Ing. Habil Albrecht Zur, Fachhochschule Kiel - University of Applied Sciences, Fachbereich Informatik und Elektrotechnik

12.1 Abstract

This work deals with the development of software for controlling an airship, the weather data (solar radiation, temperature and wind) to determine. The aim of this thesis is to integrate control cards, ie To allow communication between the Sensors, Actuators and transceiver card. The integration should be carried out as part of a Linux platform. It has gained acceptance because of the embedded system and the specific processor type, the RedHat distribution.

TABLE OF CONTENTS

List of Figures	2
LIST OF TABLES	2
1. Introduction to the airship project	4
1.1 . The Lotte-project and the "Alternative LOTTE"	4
1.2 . Development method	4
1.3 . Activity and REALISIERUNGSANNAEHERUNG	5
1.4 Overview	5
2. Basics of Work	6
2.1 The V-model	6
2.2 . Structured Analysis (SA) /structured design (SD)	8
3. Development environment	10
4. Architectural design, and implementation	11
4.1 . The implementation of the BASISSTATIONSPROGRAMMS	14
4.2 . The implementation of the LUFTSCHIFFPROGRAMMS	15
Section 4.2.1 , the sensors	16
4.2.2 . The actuators	17
Point 4.2.3 .. The Transceiver	18
Point 4.2.4 .. The Board Computer	20
5. Operating System Installation	22
5.1 . Problem	22
5.2	22 implementation of the installation

5.2.1 . . configuration of the core	23
5.2.2 . . Modules creating and installing	38
5.3	39 development environment
5.4	41 Real
5.4.1 . . latency and fluctuation	42
Item 5.4.2 above	hard
and soft real-time conditions	43
5.4.3 . . embedded applications	46
5.4.4 . . Installation of Linux-Echtzeit	47
6. Description of PORTZUGRIFFSARTEN	49
6.1 . Direct Access	49
6.1.1 ,	rights awarded 49
6.1.2 . . Einlesen/Ausgeben	51
6.2 . ACCESS about GERaeTEDATEIEN	54
6.2.1) , . Ports Opened and closed	54
Item 6.2.2 , . canonical and non-canonical Input/Output	56
6.2.3 . . synchronous and asynchronous transfer	67
6.3 . THREADPROGRAMMIERBEISPIEL	72
7.	75
LITERATURREFERENZ	

List of figures:

Figure 2- 1: V-model [1]	6
Figure 2- 2: SA/SD [4] ...	9
Figure 4- 1: The connections of the system	12
Figure 4- 2: overall system of the "alternative Lotte"	13
Figure 4- 3: Diagram of the correlations	14
Figure 4- 4: airship as embedded system	15
Figure 4- 5: : The sensors through the serial interface with the Board Computer 16 connected	
Figure 4- 6: The actuation system through the serial interface to the Board Computer Angeschlossen.....	
Figure 4- 7: The transceiver through the serial interface with the Board Computer 18 connected	
Figure 4- 8: Graphical User Interface	19
Figure 4- 9: The Board Computer	20
Figure 5- 1: make config(1 Option: Code maturity level)	24
Figure 5- 2: make config(2 Option: Loadable module support)	25
Figure 5- 3: make menuconfig(1, half of the Optionen)	25
Figure 5- 4: make menuconfig(2, half of the Optionen)	26
Figure 5- 5: make xconfig(everything on Einmal)	27
Figure 5- 6: Option Processor type and features	29
Figure 5- 7: Call the assistance of the submenus Processor family	29
Figure 5- 8: sub-menu (CPU frequency scaling) from option 3	30
Figure 5- 9: General setup(1, Haelfte)	31
Figure 5- 10: submenu PCI Hotplug support	31
Figure 5- 11: submenu PCMCIA/CardBus support	32
Figure 5- 12: 42 The Ereignislatenz	
Figure 5- 13: periodic fluctuation	42
Figure 5- 14: Schadensaenderung in dependence of the response time	43
Figure 6- 1: Terminal E/A-functions in the Overview	66
Figure 6- 2: asynchronous data transfer with 7 bits Datenlaenge	68

LIST OF TABLES

Table 5- 1: Comparison of the performance of Linux with the commercial Echtzeitkernen.	
---	--

Table 6- 1: macros to access I/O-Ports	52
Table 6- 2: The five different Modi-Eigenschaften a terminal	61
Table 6- 3: Possible Steuerzeichenvarianten for the non-canonical input	63
Table 6- 4: POSIX-functions to query and change the Terminalattribute..	66

Thanksgiving

This thesis was developed at the Association for alternative energy research in Karlsruhe in cooperation with the University of Applied Sciences in Kiel. At this juncture I would like to thank for Prof. Dr. -Ing. habil Albrecht to, this thesis by the University of Applied Sciences and betreuet has had so much patience, and I would like to thank Mr Ing. Murad, Samir for the enabling of the thesis and the furnishing of the laboratory with its entire Hardwareinhalt thank you.

My special thanks to the German state, has given me the opportunity to continue my education and to acquire new perspectives.

Last I would like to thank my parents, stood side by side to me, despite the distance, and with my friends, the me morally and actually have supported.

12.2 Introduction

12.2.1 The Lotte-Projekt and the "alternative Lotte"

At the Institute of statics and dynamics of the air and Raumfahrtkonstruktion at the Universitaet Stuttgart was built a Solarluftschiff in February 1992 the project "Solarluftboot" initiated and the purpose of this project was, new materials, new methods and a new concept for the control to develop an airship, the is operated by a Solarenergiemaschine.

The "Alternative Lotte", a joint project of the VaEf e.V. , the Universities of Karlsruhe and Stuttgart and Karlsruhe University of Applied Sciences, was planned as an experimental airship, it is to be controlled directly from the ground (first step of the implementation) or automatically to specified coordinates (the second step of the implementation) fly, the energy will be the first step guaranteed by conventional batteries, but it is planned, and later to switch to solar power, with the "alternative Lotte" specific data will be during the flight through collected different sensors (sensors), the on the airship can be installed in the first step is the speed of the wind, the temperature and the solar radiation measured. In addition to such a flight data, such as angular velocity and acceleration are navigational to azimuth angle measured.

1.2 . Development Method

The complete system of the airship was in a laboratory with a Board Computer, and a sensory system-, a actuators and a Transceiverkarte provided as embedded system.

The Software is with the C-programming language developed under Linux, the complete development process follows in general the well-known V-model.

1.3 . Activity and Realisierungsannaeherung

The task is to provide a software interface for the communication to develop, the on the board computer running, which in turn with the three hood (sensors, actuators, and transceiver) is connected on the airship, the purpose of this program is, the communication between the sensors and the motors and the transceiver on the "alternative Lotte" sensor platform to produce, the sensors is used to the solar radiation, and the speed and the direction of the airship to measure and the data to send to the board computer, the actuation system is to control the direction and the acceleration (speed, position) of the airship (engines) used by the data from the Board Computer dependent on the data received from the sensors will receive the Transceiverkarte is used, to the data in the A direction by the Board computer to send to the local computer and, in the other direction to receive from the local computer. So the thesis can be divided into the following tasks:

- ☞ Preparation, configuration, and translation of the core.
- ☞ Installation of Linux and real time extension.
- ☞ Programming of the communication protocol between the cards and the Board Computer with the C-programming language.

1.4 Overview

After a brief introduction in a concise form is basic knowledge of the given development methods used.

The V-model, and some general information about (SA/SD) are in Chapter 2 gives. Chapter 3, there are a impression of what tools have been used in 4 chapter is a detailed description of the full system, Chapter 5 discusses the operating system and the real, Chapter 6 provides the necessary information on the serial interface, as well as the approach to the programming and the tests that have been carried out to the functionality of the system to check.

12.3 Development environment

The development is on a computer with GX1-AMD K6 300MHz Processor, 32MB RAM, 16 MB Flash Disk is done by the hardware (hood) to each other with the Board Computer was connected via R232 order.

The following are software tools for the development have been used:

- ④ SUSE and RedHat Linux.
- ④ KDevelop: C/C++ development environment under Linux.
- ④ "GCC" and "GDB" Compiler and Debugger for C-programs under Linux.
- ④ Microsoft Word to word processing and the representation of the design structure.
- ④ Two computers each with a RS232 connection for testing purposes.
- ④ Null Modem Cable for the connection of the RS232 serial interfaces.
- ④ Hyperterminal under Windows and minicom on Linux.

12.4 Architecture Design and Implementation

The task is to provide a software interface to develop to the integration with integration is the communication between the control cards, i.e. the sensors, the actuators, and the transceiver, this Communication is to be in the form of a C-program take place on the board computer, the then the exchange of data between the control monitors and controls, these four components are located on the "Alternate Lotte" and will be the serial interface connected to each other.

Each of these hood was for either as a thesis,

Thesis or Ingenieurpraktikum designed, developed, and tested

– up on

The Board Computer, of the company Arcom was purchased

[\[5\]. The Cards](#)

Each contain a microcontroller, the RS232 serial data for either to send provides or is ready to data from RS232 to receive.

The sensor collects data and sends it to the Board Computer go, it will be the Sonnenstrahlstaerke, speed and direction of the airship measured (sunlight in dependency of the coordinates and the season), for example by sensors be used for each coordinate: X-direction, y-direction, and z-direction (the temperature, angular velocity and azimuth angle will also be identified).

The actuation system is used to control the operation of the "alternative Lotte", i.e. the direction, speed, and angle indicate, dependent on the information from the board computer, either the then this data from the sensors or but from the transceiver receives.

For its part the Transceiverkarte to the exchange of data between the "Alternate Lotte" guarantee and the ground station and wireless, by a modem for use with an antenna, this has already been successfully tested, since the Transceiverkarte via a graphical user interface, it is possible, comman417s manually from the ground and enter corrections.

The full system connections between the hood with the airship and the ground station (Benutzerschnittstellencomputer) are shown in the following figure:

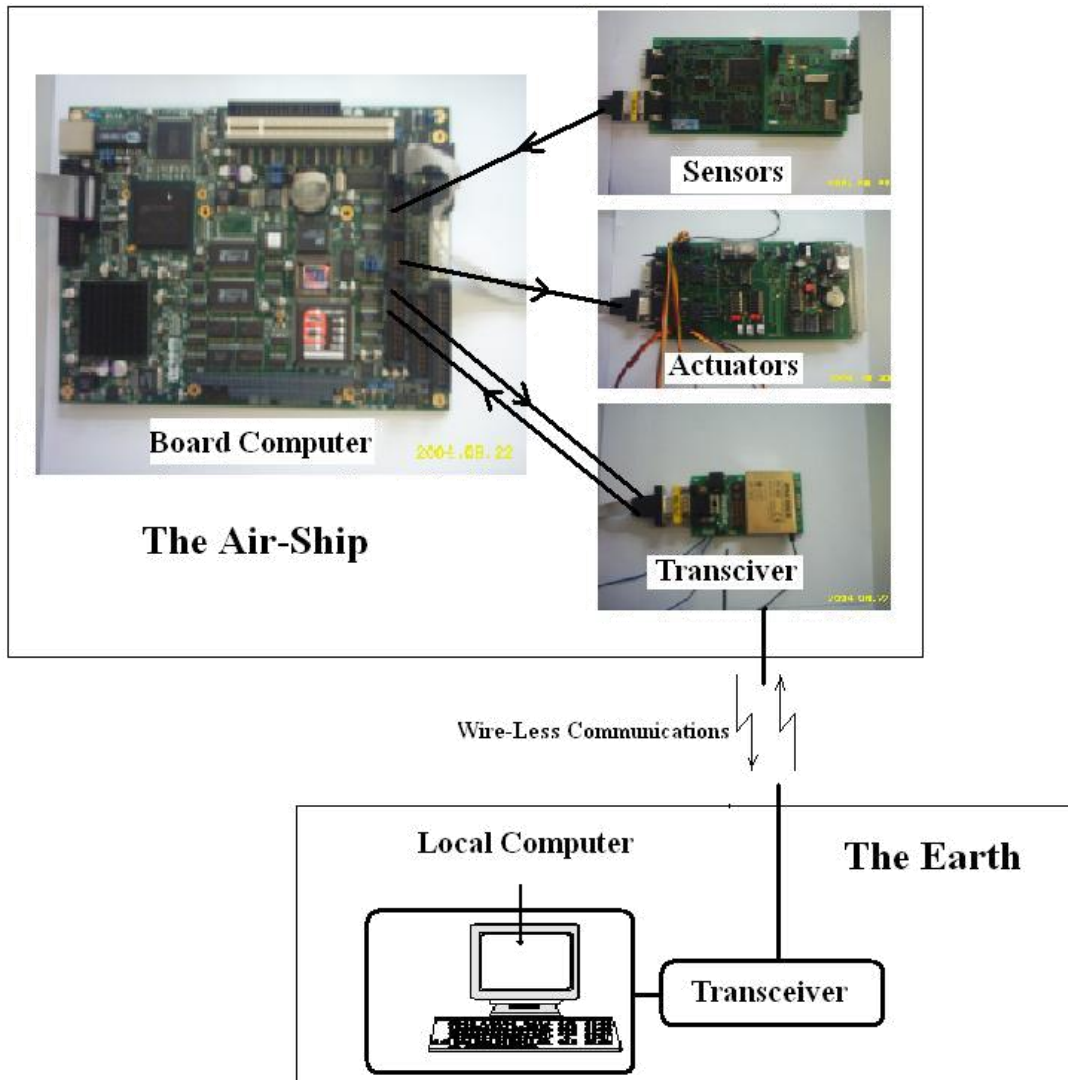


Figure 4- 1: The connections of the system.

The figure describes both the connections between the board computer and the three hood (sensors, actuators, and transceiver) in the "alternative Lotte" as well as the connection to the computer in the ground station.

Similarly, the figure shows that the overall system is divided into two parts:

1. The ground station: with a computer, of the airship from the ground and can control monitored. For this task is a graphical user interface uses. This GUI hardwaremaessig accesses to the

Transceiverkarte back. The transceiver in turn is based on a modem, the other with a wireless modem on the airship (by antennas) data in both directions may or should replace. By this graphical user interface it will be possible, wind speed, solar radiation, temperature, Luftschiffgeschwindigkeit Luftschiffkoordinaten or to receive and at the same time in the other direction wind speed, acceleration, and azimuth angle to send.

2. The airship: Here is the actual task of the present work, the integration of the three hood in the embedded computer, this is a null-modem cable by each connected with the hood, the embedded computer contains four serial interfaces, and the hood itself, are at least equipped with a serial interface. In the next section will be discussed in more detail the individual cards or presented.

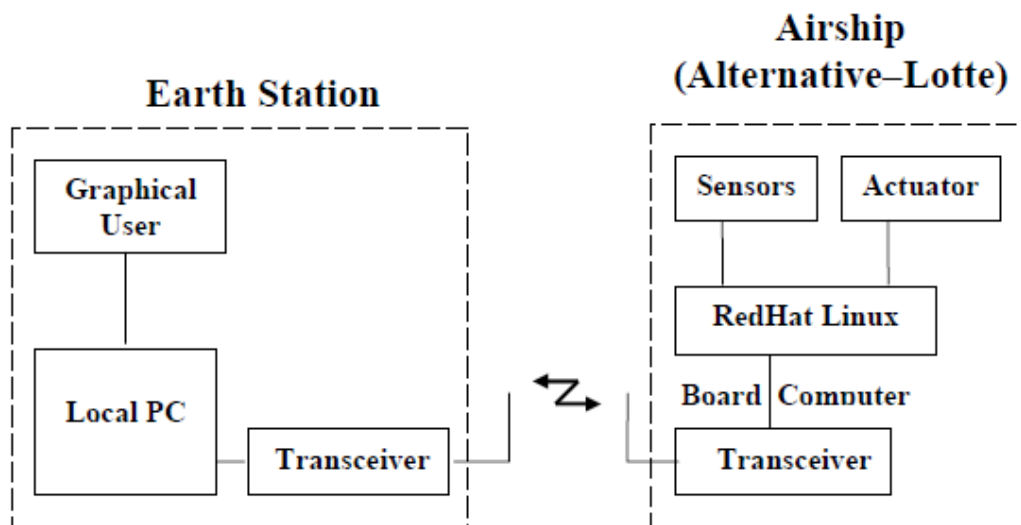
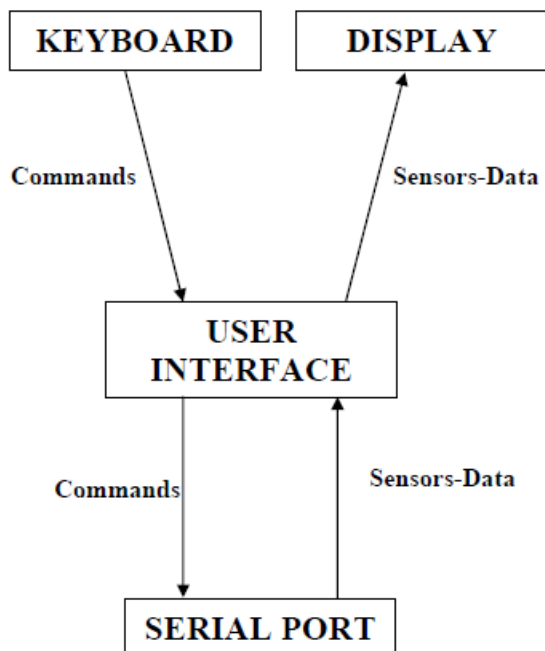


Figure 4- 2: overall system of the "alternative Lotte".



4.1 And The implementation of the Basisstationsprogramms

Figure 4- 3: Diagram of the correlations.

- ☞ The display is the screen of a computer in the ground station, it is used to to observe the received data gathered by these data gathered by the sensors are in the "alternative Lotte" detected.
- ☞ The keyboard is used by the user, to write to the commands, the he to "alternative Lotte" wants to send commands to the "alternative Lotte" sent by the serial interface, whenever the user commands to "alternative Lotte" wants to send, will be the data to the Board Computer passed from the transceiver.
- ☞ The serial port, the peripherals between the user interface and the "Alternate Lotte". By we use the user interface, we can tell, that the user receives data (from the "Alternate Lotte") and data or commands (to "alternative Lotte") sends.
- ☞ The added security layer is the communication protocol, the commands that the ground station to "alternative Lotte" send, will be confirmed, if the confirmation predictions, the commands are returned.

4.2 And The implementation of the Luftschiffprogrammms

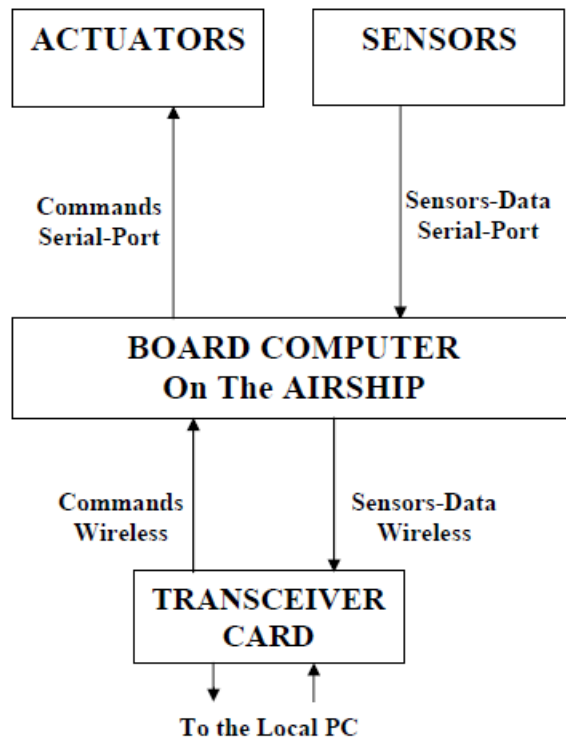


Figure 4- 4: airship as embedded system.

Section 4.2.1 The sensor system.

This card is one of the most important components of the "alternative Lotte". The sensors monitor the overall system, the fact that sensors are attached to this card, which capture the acceleration, the speed, the coordinates (the position of the airship), the angular velocity, the azimuthal angle and the direction of the then to the Board Computer via the serial interface will be transferred, this in turn, sends the data and/or information wirelessly to the ground station go. In addition provides us with the sensors other data, such as the solar radiation (solar radiation, and wind) and the temperature.

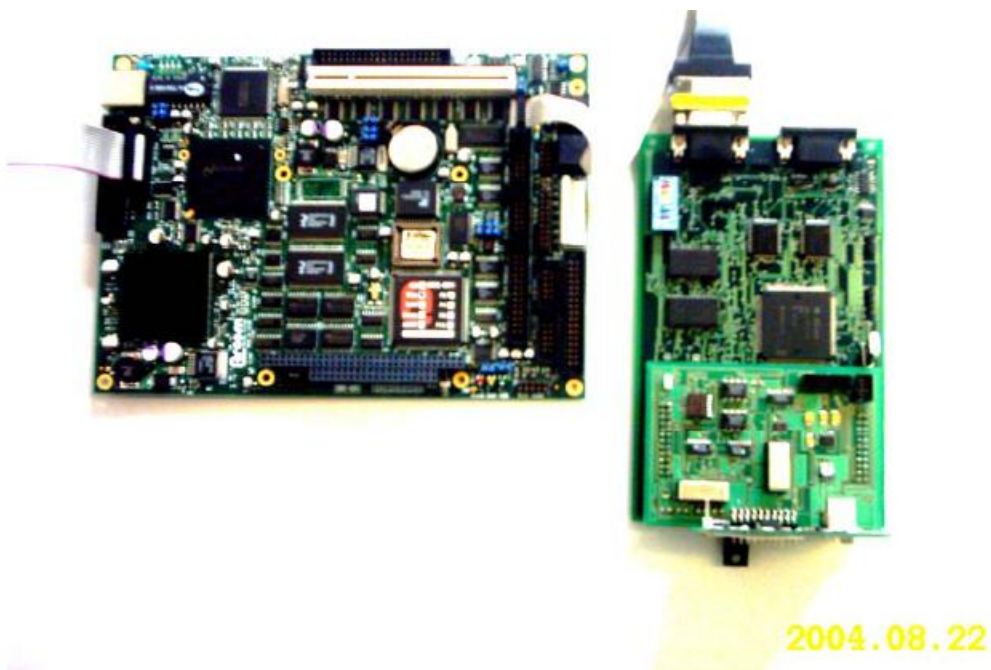


Figure 4- 5: : The sensors through the serial interface with the Board connected to your computer.

4.2.2 . The actuators

The sensors allows, flight information to collect, to bear the Veraenderlichkeit of the data to be able to be used must be the actuation system, i.e. the control card contains e.g. a stepper motor and a servo motor, in order, on the one hand, the speed of the airship and, on the other hand to control the direction, the motion and by the Board be Flugelbefehle received to your computer, these commands are either from the sensors or from the ground station and then the board computer made available to the actuators.

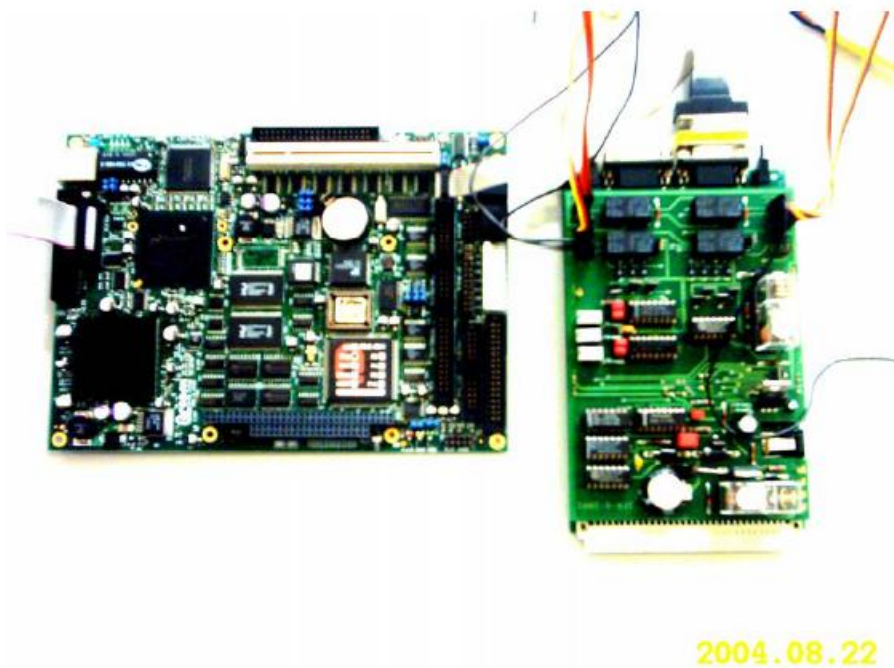


Figure 4- 6: The actuation system through the serial interface with the Board connected to your computer.

Point 4.2.3 .. the Transceiver

This map consists of both hardware and software, the hardware is in the form of a modems, the wireless communication between the local computer to the ground station and the board computer, on the "alternative Lotte" guaranteed. This communication means data exchange between the ground station and the airship, by a graphical user interface (the software p. Fig. ?) is made available, in order, for example enter the speed, this is by the Board Computer and forwarded to the actuators according to the speed increased or lowered, depending on which value has been entered or what is better.

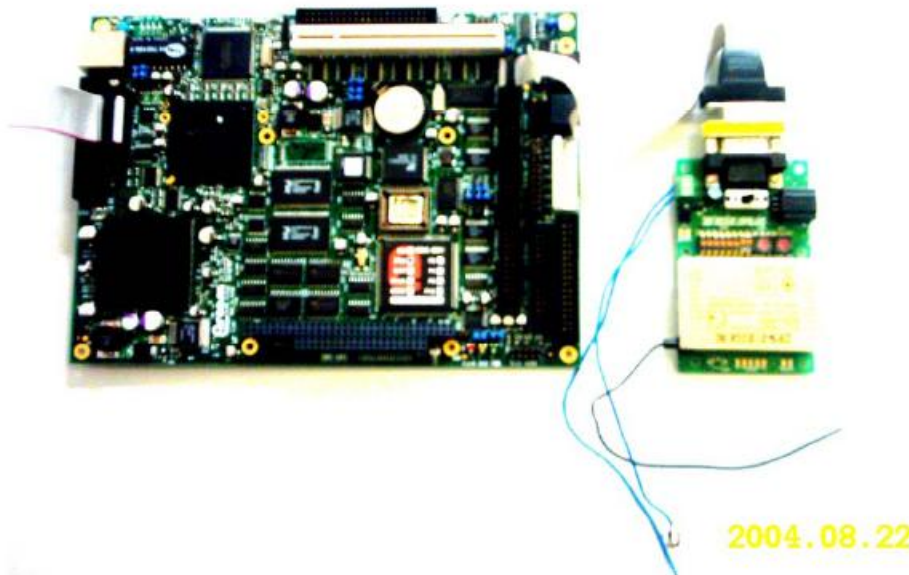


Figure 4- 7: The transceiver by the serial interface with the Board connected to your computer.

If the transceiver is receiving data gathered by the Embedded System, these data will be (in the Receive Data frame displayed) to the ground station with a frequency between 433.200 and 434.775 MHz sent, and the same happens when the transceiver data from the ground station to the "alternative Lotte" and then send the actuators wants to (in the other direction you send commands by sending the data frame is written).

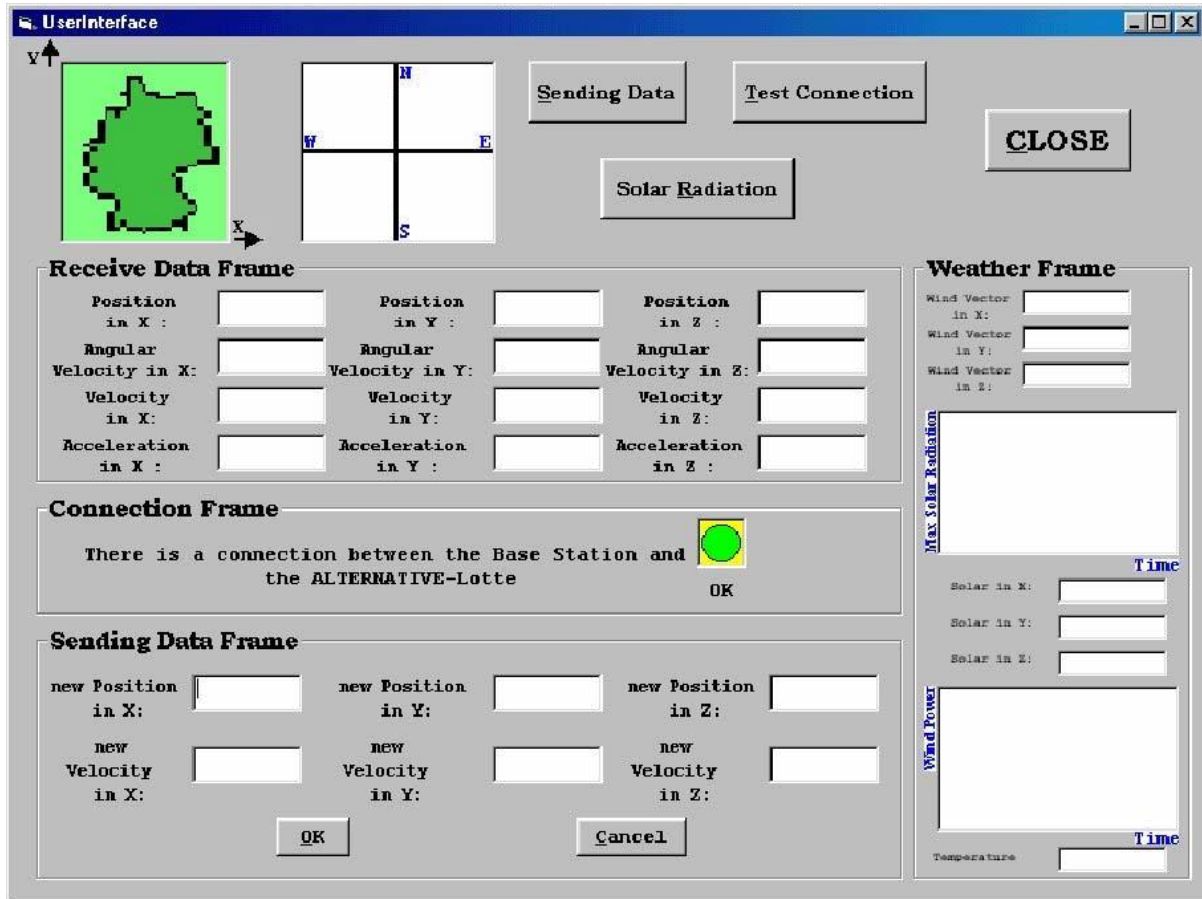


Figure 4- 8: Graphical User Interface.

Point 4.2.4 .. The Board Computer

The Board Computer is the embedded computer in the "alternative Lotte". This computer closes the three hood through the serial interfaces (RS232 together).

Specification of the embedded computer [5]: [EBX AMD Geode® GX1 Embedded Computer](#)

The SBC-GX1 is a low profile, without fan, the EBX Format board, is based on the 300MHz MMX-increased AMD Geode GX1 processor, it includes all Rechnerstandard-Schnittstellen and a full range of Multimediaeigenschaften: VGA-screen (National XpressGraphics), Ethernet 10/100BASETX (PCI 2.1 compatible), integrated 16Mbyte FLASH, double-USB, compatible interface of the sounds and four serial interfaces.



Figure 4- 9: The board computer.

Processor	Without fan, Pentium class, 300MHz AMD Geode™ GX1
Memory	SDRAM 144-pin SODIMM socket low-profile, 256Mbytes Max. Flash: 16Mbytes Intel strata Flash SRAM: 128K battery backed (factory fit option)
Cache	16k L1 write-back cache
Video	TFT flat panel & CRT XVGA, 1 - 4MB SDRAM video memory
Drive Support	FDD, HDD, Silicon Disk in Flash, Compact Flash
Networking Support	10/100BASETX Ethernet via RJ-45
USB interfaces	2 X USB ports (USB 1.1)
I/O Interfaces	4 X; 16550-compatible fast serial ports (3 x RS-232, 1 x RS 232/422/ 485)
PC peripheral	Keyboard, Mouse, Printer Port
Additional I/O	8 General purpose digital I/O signal and two user defined Jumper
Watchdog timer	Real Hardware Watchdog (2 or 8 seconds) with Reset output.
Enlargement	
Sound	Compact Flash socket, PC/104 bus, PCI slot
BIOS	16-bit SoundBlaster/Per compatible interface
MTBF	Award BIOS (in accordance with Millennium)
Format	90.000 Hrs
Power Requirement	EBX format, size 5.75 " x 8.00 " (146 mm x 203 mm) +5V only for operation (@ 1.5A Type)
Temperature Range	From: -20 to 60 °C processor fed with a low profile (no fan) temperature drop.

5. Operating System Installation

5.1 Problem .

The Board Computer hood is to manage the communication between.

He comes from the company Arcom [5]. on him is no operating system installed, since only

A 16MB Flash Disk is present (embedded computer), it is useful, to use Linux as the embedded operating system, Linux offers the advantage that the core (the core of the operating system) can be created the desired configuration of. This saves space on one hand, on the other hand, the system or the hardware be optimally set up.

5.2 Implementation of the installation.

At the beginning of the configuration options were the most disabled, because only a small core as soon as possible or a general operating system was needed, the following options were disabled: Parallel support, Plug and Play configuration, Enterprise Volume Management System, multi-device support (RAID and LVM), cryptography support (CryptoAPI), networking options, Telephony Support, ATA/IDE/MFM/RLL support, SCSI support, Fusion MPT device support, IEEE 1394 (FireWire) support (experimental), I2O device support, network device support, amateur radio support, IrDA (infrared) support, ISDN subsystem, Input core support, multimedia devices, sound, USB support, Bluetooth support, kernel hacking, library routines, plug in CPU scheduler that support resource management modules and build options.

In total, 33 options are disabled, they were divided in part, in several sub-menus, and also partly in only in a single sub-menu, and the majority of options contained two to three sub-menus, and the pictures show some screenshots, which an impression of the number of options is to communicate.

In this way although the configured core is small, did however, he is only after some effort compiling.

Due to the large number of submenus in peace had to be considered options, so that you can be more clearly understood.

The configuration steps are available everywhere on the Internet, however, the configuration of the core is not comparable with the installation of Windows, and even people, the experience with the installation of Linux, had difficulties with security with the core configuration.

The following examples are intended to illustrate Embedded operating systems, such as embedded operating systems such as PDAs, organizer, and mobile phones present, including WindowsCE or embedded Linux but also on this hardware is not, for example hard disk, but only a Flash Disk, on which the operating system is charged.

The installation of the core is actually a sequence of six commands. It is the first command the One who, of the most processing time takes or configuration in. The other need then depending on computational power and memory up to a few days processing time.

These commands are as follows:

- Make menuconfig
- Make DEP
- Make bzImage
- Make modules
- Make modules_install
- Configure the boot loader.

5.2.1 . . configuration of the Core

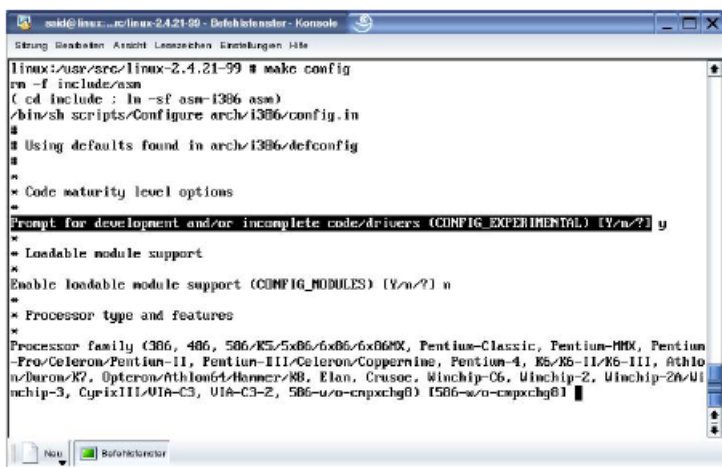
Only the administrator (root) or the user, which these rights were granted, the appropriate access rights to perform this configuration, this is done with the command `sudo "command"`, previously the administrator must the user all rights in the file `/etc/sudoers` assign.

If the Kern-Quelldateien are not yet installed, this should be done as the first, because otherwise the configuration does not can be started, the present source files were 208.19 MB in size, and also the Help files are included in the directory `/usr/src/linux-2.4.21.99 /documentation`.

There is a big help file with 1.2 MB for the core configuration, and each small help files to almost every option.

In order to now in the correct directory (/usr/src/linux-2.4.21.99 in this case) one of the three possible enter commands, there are far more than 1,000 options to choose from, with these options you can influence, which functions directly be integrated into the core, which as a module and which are not available to:

- Make config: a couple with Y(yes) /N(No) or M (modules) to answer questions one at a time (P. image. ? and ?), without the Possibility to have, reenable remote access or umzuaendern (textbased), EN few questions previously answered questions hangs up with together!



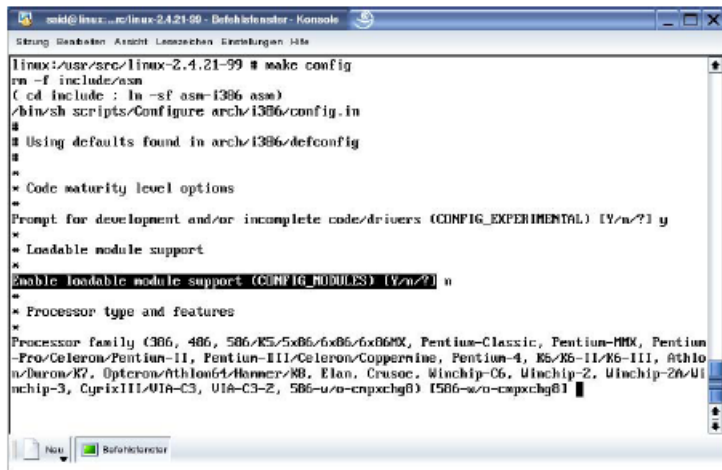
```

mid@linux...rc/linux-2.4.21.99 - Befehlsfenster - Konsole
Strung Gegeben Ansicht Lesezeichen Einstellungen Hilfe

linux:/usr/src/linux-2.4.21.99 # make config
rm -f include/asm
( cd include ; ln -sf asm-1386 asm )
/bin/sh scripts/Configure arch/i386/config.in
#
# Using defaults found in arch/i386/defconfig
#
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [Y/n/?] y
*
* Loadable module support
*
Enable loadable module support (CONFIG_MODULES) [Y/n/?] n
*
* Processor type and features
*
Processor family (386, 486, 586/485/5x86/6x86/6x86MMX, Pentium-Classic, Pentium-MMX, Pentium-Pro/Celeron/Pentium-II, Pentium-III/Celeron/Coppermine, Pentium-4, X86-11/X6-111, Athlon/Duron/K7, Opteron/Athlon64/Hammer/K8, Elan, Crusoe, Winchip-C6, Winchip-2, Winchip-2A/Winchip-3, CyrixIII/VIa-C3, VIa-C3-2, 586-u/o-cnpchg0) [586-u/o-cnpchg0]

```

Figure 5- 1: make config(1 Option: Code maturity level).



25

Figure 5- 2: make config(2 Option: Loadable module support).

- Make menuconfig: is also text-based in the process, there is also the opportunity, before and reverse to jump and to correct, i.e. , the initially to make amendments reenable remote access to edit or even new (see image, 5.3 and 5.4).

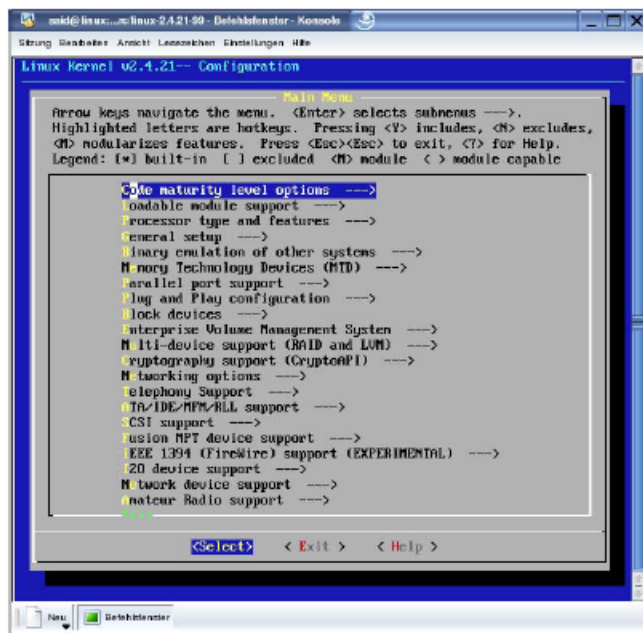


Figure 5- 3: make menuconfig(1 half of the options).

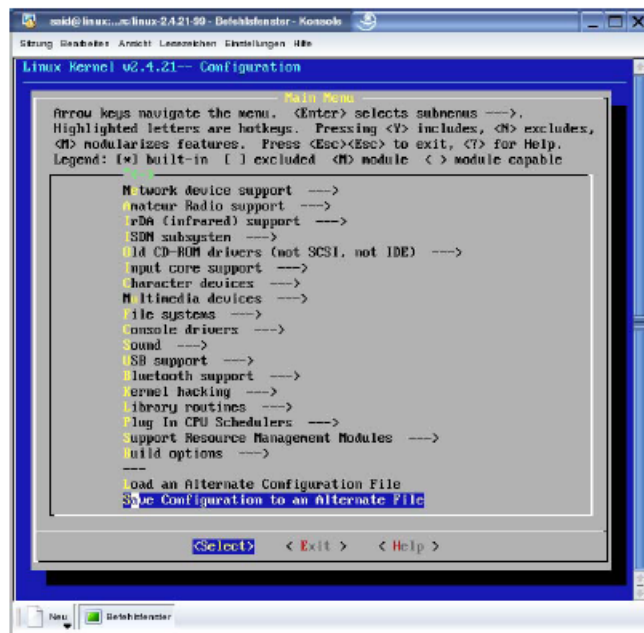


Figure 5- 4: make menuconfig(2 half of the options).

– Make xconfig: is not more text oriented, but simply can be operated with the mouse (see image, 5.5), (xconfig only if the programming language Tcl/Tk is installed), but they will need the X11-surface, i.e. the necessary libraries and the support for the Graphical User Interface, or KDE, and that is why this alternative is usually only at the beginning to test and/or used learning, one hand, because the core is compiled only for this reason, to a minimum to ensure demand for storage space, so this is not a recommended alternative, unless you want to compile the core to its Linux distribution (Suse, RedHat, ...) to its special requirements of hardware and security to adapt and optimize. without sufficient basic knowledge can play a to the configuration of the core the existing installation or even make the whole system unusable.

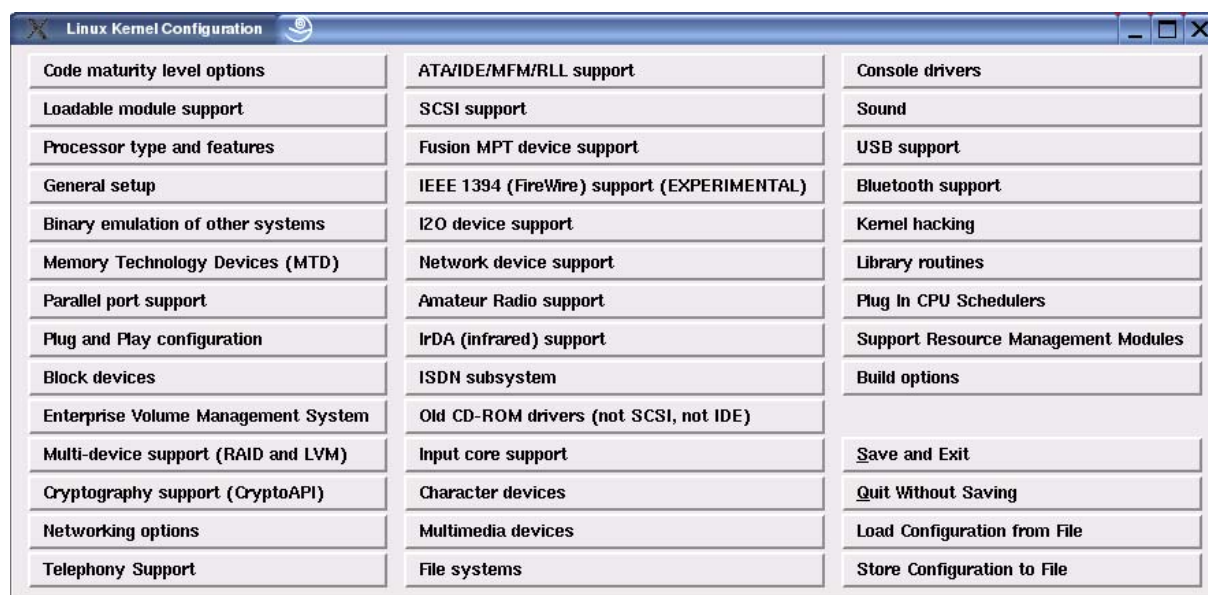


Figure 5- 5: make xconfig(everything all at once).

After menuconfig has started, there are options to the "Y" for Yes or "N" for "no" to answer Are, partly you can with "M" for module answers, and in some cases, values are specified.

First, one should the current configuration under a different name save as a precaution, the last option: Save Configuration to alternative file), as it can happen that the changes in the preconfigured file be stored, which can lead to problems, if the file does not compile due to e.g. Unvollstaendigkeit cannot be, i.e. it comes to the problems of the Kernladens when the computer is restarted.

What are modules and their benefits?

A module may be, in so far as is necessary, be integrated into the core, these modules are available but only then, when the core is loaded, loaded to the core modules can be unloaded or which modules are trying to start to find your hardware (autoprobing).

When the computer is a Basiskern loaded, the contains only those functions to Start are required, these belong firmly to the core, if in the current operation functions will be needed (e.g. , for the new hardware), is the required code as a module connected to the core, and if this

Additional functions not a while more will be needed again the module can be removed from the core, this modularized approach has many advantages:

- Kern-Module can be integrated as needed, and if a specific module only rarely is needed, can be saved on this memory, i.e. , the heart of the matter is not greater than is essential, and to the hardware of the optimally adapted user.
- For a change of the hardware (such as a new network card) must be compiled no new core, but only the new module will be involved.
 - In the development of a Kern-Moduls constantly of the computer must not be restarted in order to made to test changes to the core.

It is enough, to recompile the module, then there it can be tested while the system is running.

Under the Options, you will find for example, physical resources, such as drives, serial/parallel ports, Audio, and Videogeraete, network interfaces, keyboard, mouse, etc...

1. **Code maturity level options: here you have the possibility, even the Core**

Components that are not yet mature as apply, in the kernel config file

To be taken into account, in the other case, these options will not be active, or are not displayed. This option was chosen, as a module cannot be you are not elect.

2. **Loadable module support: you determine whether this modularized**

Concept supports, whether or not some of the options as loadable modules must be available or only with Yes/No the core are to be tied.

3. **Processor type and features** (p. image. 5.6):

Here you have the type of processor

Specify what the speed and size of the Code will affect.

Code in this case is on-, but not – as usual – abwaertskompatibel (p.

Image. 5.7), e.g. , a K6-processor can a K7-processor support

But not vice-versa, here was an Athlon/Duron/K7-processor was elected. Math emulation and symmetric-multiprocessing support were not elected,

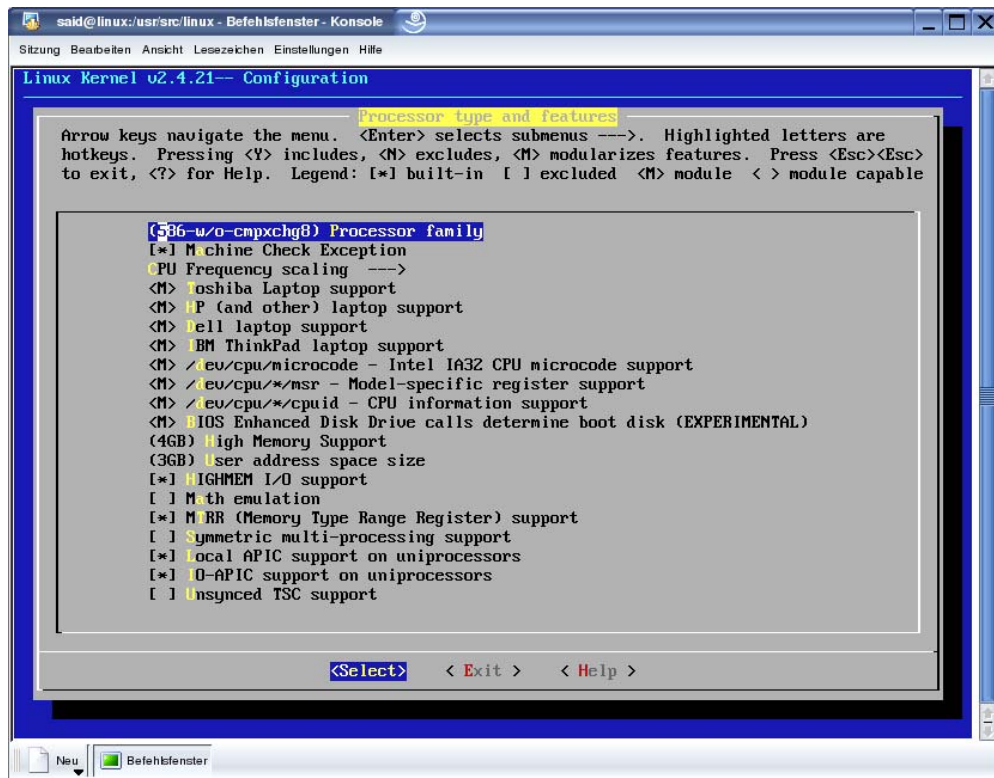


Figure 5- 6: Option Processor type and features.

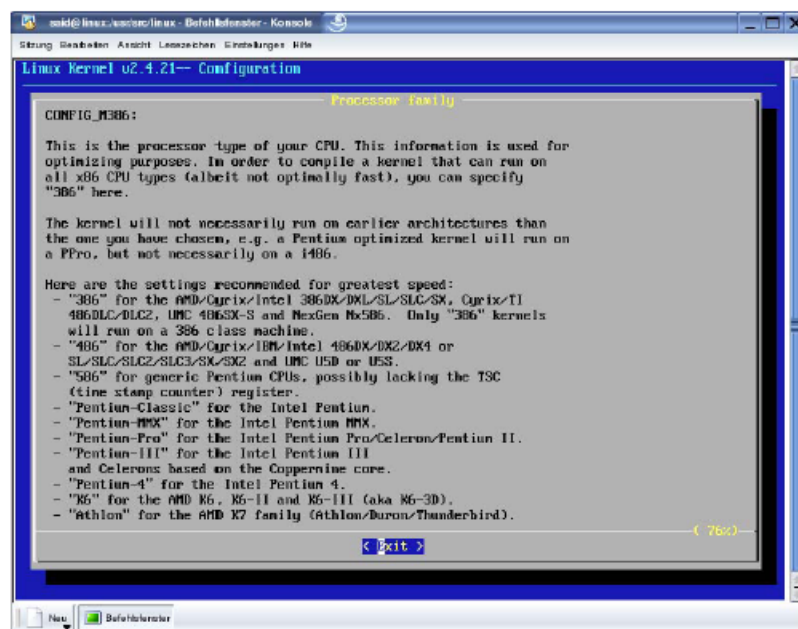


Figure 5- 7: Call the assistance of the submenu Processor family.

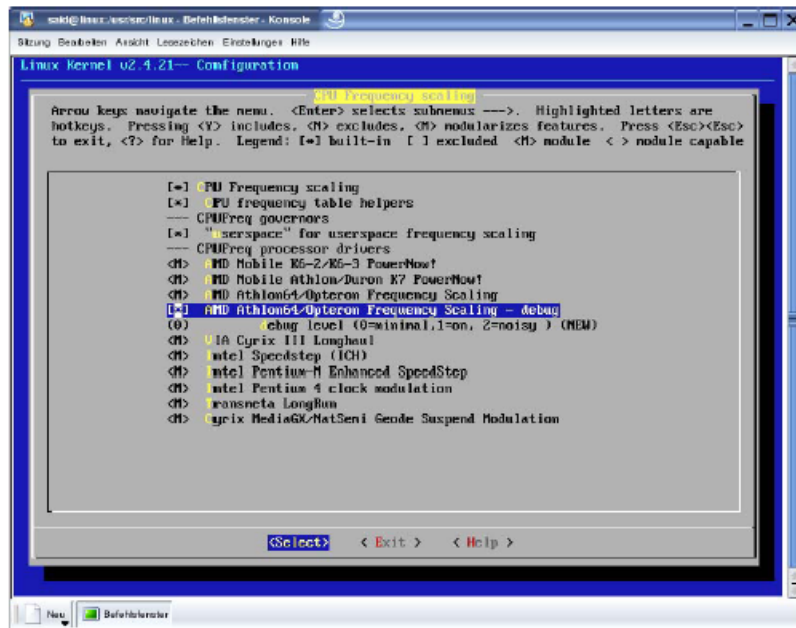


Figure 5- 8: sub-menu (CPU frequency scaling) of Option 3.

4. **General setup: Networking support** (see image, 5.9 and 5.10): This option

In any case it is needed, since some internal commands to build the network protocol, the correct networking options are only for item 13 (Networking options) discussed PCI support was chosen (p. image." field 5.11), on PCMCIA/CardBud support (see image, 5.12) has been omitted, SystemV IPC had to each case because of the needed inter-process communication are selected in the programming, Elf and a.out Kernel Core format are also set to "yes", Power Management Support for laptops with "yes" answers, but answered in ACPI support.

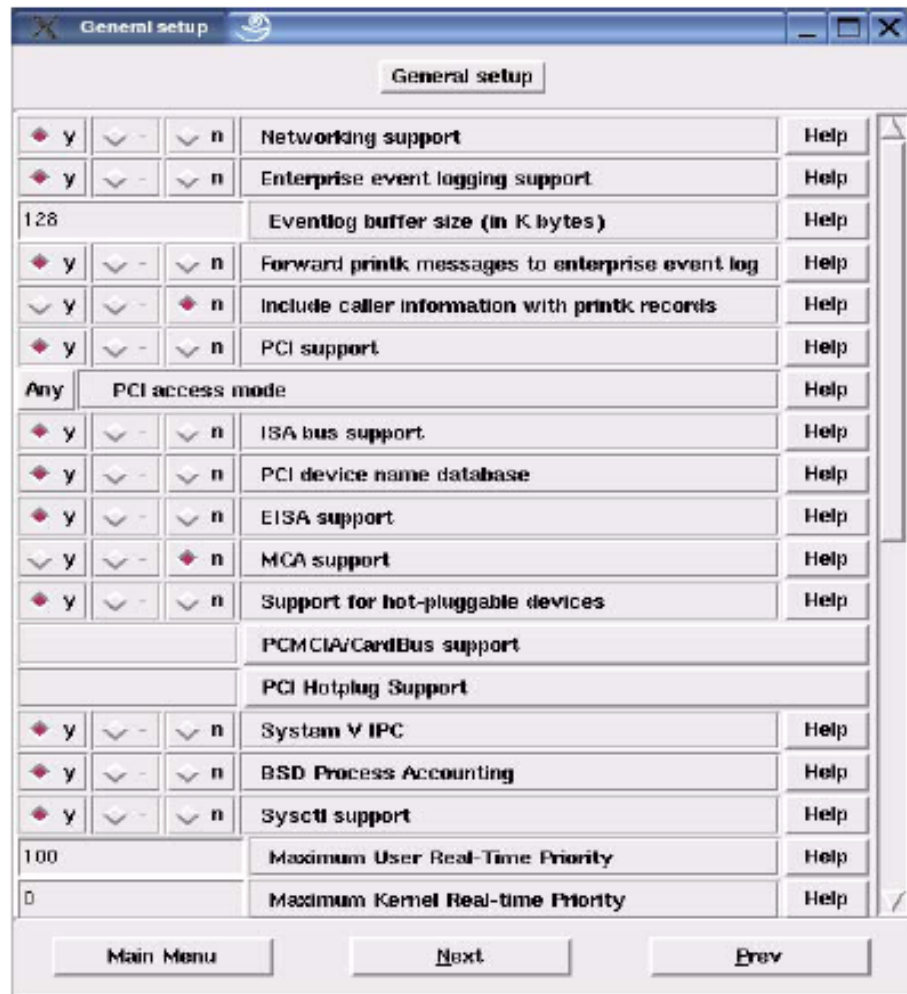


Figure 5- 9: General setup(1 half).

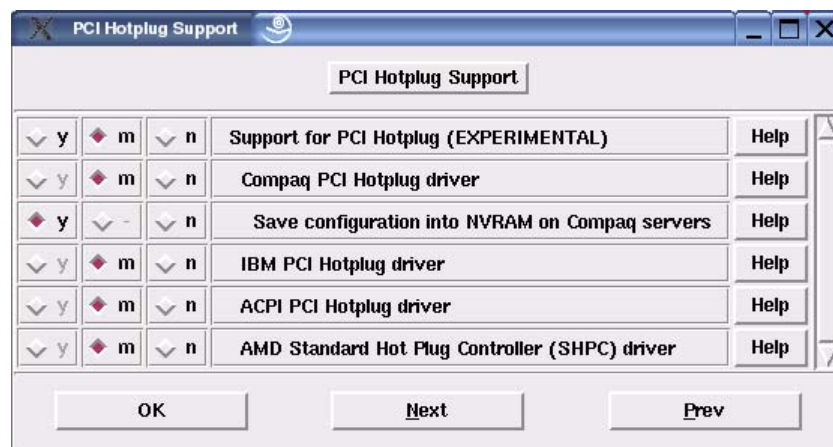


Figure 5- 10: submenu PCI Hotplug support.

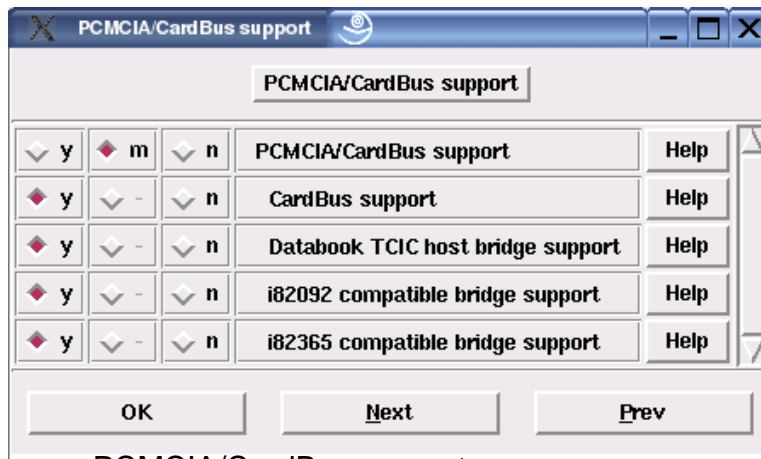


Figure 5- 11: submenu PCMCIA/CardBus support.

5. **Binary emulation of other systems:** This option is necessary, in order to e.g.

SOM and/or Solaris-binaries to load and implement directly. SOM is a binary executable format, the HP/UX is accepted, this option has been omitted.

6. **Memory Technology Devices (MTD):** so are e.g. Flash, RAM meant.

They are mostly for stable systems used in embedded systems, and that is why this option is very important in the present case and had to be answered with "yes" to any partition was selected, because only a Flash Disk is present, and even the Flash Disk of Arcom is the company to enable specified here-the driver for the SBC-GXn Board-Familie has also enabled.

7. **Parallel port support:** here it is to the parallel port to the

Normally, the printer is connected, in the present case, it is not planned to use a printer, therefore, it was "no" selected.

8. **Plug and Play configuration:** is a standard for plug-and-, the

In the Detection address decoder unit can be configured.

9. **Block devices:** a floppy drive in the present case is not

Needed, a RAID-controller also not normal (not SCSI) hard drives and CD-ROM drives (among other things) are under Linux as a block devices detected.

10. **Enterprise Volume Management System: this is a plugin-based**

Framework for Datentraegermanagement and combines the support for the partitioning, software, RAID, LVM, and much more in a single interface, user tools are necessary to the administration of EVMS logical magnetic media, according to this option was enabled.

11. **Multi-device support (RAID and LVM): RAID and logical volume**

Management will be in our Board Computer not needed, so if the "No" option was elected.

12. **Cryptography support (CryptoAPI): the authentication and guaranteed**

The encryption and probably the encrypted authentication and transfer over the network (no help), according to, a "no" selected.

13. **Networking options: Network packet filtering is responsible for the firewalls.**

In this project you are not needed, so "no" selected, select the TCP/IP networking was answered with "yes".

14. **Telephony Support: was not elected, for the case that phone**

Should be, must be a connected-associate.

15. **ATA/IDE/MFM/RLL support: here offers the possibility to appliances to**

Support to the IDE-bus be connected such as IDE-disk drives and ATAPI CD-ROMs.

16. **SCSI support: Support for the SCSI hard drive is not necessary.**

In contrast, the option SCSI CD-ROM with answers "yes". SCSI low-level drivers should not be supported, as other SCSI-hardware not is present.

17. **Fusion MPT device support: the option Message Passing Technology Offers**

As a machine name matches the possibility to activate a Systemhost

Either as a high-performance SCSI Host initiator or as a LAN (but not with parallel SCSI-media). The Fusionsarchitektur is in the location, these protocols bi-directionally in Hochgeschwindigkeitslaser (up to 2GHz * 2ports = 4GHz) and parallel SCSI (up to Ultra-320) Physical Media process. These drivers need a Systemhost installed in the MTP-compatible PCI-adapter, these adapters contain special I/O processors, and thus was the option to "no" answers.

18. IEEE 1394 (FireWire) support (experimental): Fire Wire-Geraete are not

Present, therefore the fire Wire-Anschluss are not supported.

This option would be for example inactive, if the first option(Code maturity level) answered with a "no" would have been, because the FireWire under Linux to the core 2.4 has not yet been declared for maturity.

19. I2O device support: Is a intelligent input/output architecture that it

Allowed, hardware drivers in two shares to parts: the OS module (OSM) and a hardwarespezifisches module (HDM), but it needs a I/O and in the Computer Schnittstellenadapterkarte. This card contains a special I/O processor, the high speed creates, because the CPU is no longer with the I/O needs employ, because the I/O Schnittstellenadapterkarte is not present, "no" selected.

20. Network device support: If you are in any way with another

Computer wants to communicate, then this is the first option (network device support) with "yes" to. In the track, the kind of communication and the hardware be specified, all of the following submenus were answered with a "no": ARCnet devices, Ethernet (1000 Mbps), FDDI driver support, HIPPI driver support (experimental), wireless LAN (non-hamradio), TokenRing devices, Fiber Chanel driver support, Red Hat Creek Hardware VPN (experimental), WAN interfaces. In contrast, PPP (point-to-point protocol) support, SLIP (serial line) support set to "Yes" under the Option Ethernet (10 or 100Mbit) were the information on this network card in the affirmative.

21. **Amateur Radio Support: This can cause the computer communications via**

Amateur radio operations. One of these protocols can be either for point-to-point or as a institution for TCP/IP will be used, the option has not been in the affirmative, since first, no appropriate hardware for this is present, and secondly not on the radio with other computers to communicate.

22. **IrDA (infrared) support: Here you can find drivers for the infrared port.**

This is not needed, according to the option has been denied.

23. **ISDN subsystem: here are drivers for the Linux supported hardware**

And options for various ISDN-apparatus compiled, therefore these options were answered with a "no".

24. **Input core support: Human Interface Device (HID) refers to a**

Certain Geraeteklasse the USB-standards for computer, with the description are described such devices, the directly "interact with people". meant are apparatus, with which the user directly communicate with the computer, such as mice and keyboards, but also gamepads, tablets and joysticks, HID-Geraetetreiber are usually included in the major operating systems, is a HID device (during operation) is connected, it is usually directly as a type Eingabegeraete (Human Interface Devices) recognized and then displayed in the Device Manager, support for such Eingabegeraete is not needed and therefore "no" selected.

25. **Character devices: In order to anything at all to be displayed on the Monitor**

Have the first two options (virtual terminal & Support for console on virtual terminal) be answered with "yes". The same applies for the keyboard standard/generic (8250/16550 and compatible UARTs) serial support should be on every case, "affirmative" because, in the context of the present work is the communication via serial interfaces should be guaranteed, a mouse is not used, therefore the option is "Mice" with "no" have been answered.

26. **Multimedia devices: here you have the possibility, Audio/Video, and FM-**

To support maps, different video cards and drivers are usually bound in the core, this support and also those for radio are not needed.

27. **File system: The first option (quota support) is used to the space**

Restrict individual users, according to you can be denied,

Since there is only one user, it follows a number of options, the file systems of different platforms and operating systems support, how ADFS, Amiga FFS, Apple HFS/HFS+, NTFS (read only), OS/2 HPFS file system support and DOS FAT fs support, these options were disabled, only the Linux file system was in the affirmative, so reiserfs and ext3 support file system support JBD).. Second extended fs support is the Linux file system, the Minix fs support has replaced.

The option 'ISO 9660 CD-ROM file system support is "yes" to, so that CD-ROM drives can be used, and the Microsoft Joliet CD-ROM extensions is not needed, the extension also not transparent decompression, this point needs a comment:

The option is a Linux-specific extension, to data in a compressed form on CD-ROMs and then to save transparent and these CDs to get uncompressed displayed!

Under the option Native Language Support has been everything, up to code page 437 (United States, Canada), page 850 (Europe) and NLS UTF8 was denied, so that meant Zeichensätze are to access foreign filesystems to be supported, the default option NLS option has been with "iso8859-

1. Indicated, this Standard, the from of the HTML-world is known.

Network File System: The first option (Coda file system support (advanced network fs)) allows, foreign computer via network file systems to use, which provides in comparison to NFS a number of advantages: Support for captive operations (e.g. , for Laptops), security models also by authentication and encryption, both options have been activated as modules, root file system on NFS was also activated and also CIFS support (advanced network file system for Samba, window and other CIFS compliant), SMB file system support (to mount Windows shares

Etc.) and NCP file system support (to mount NetWare Volumes) are not in the present case of use.

The last sub-menu: Partition Types was not elected, but that means if you would like to use hard drives, the under a different operating system to wählenden(e) have been partitioned.

28. Console drivers: here must be the VGA text console option set to "Yes"

In order to Text on VGA-maps to be MDA text console (dual headed) (EXPERIMENTAL) should also be answered in the affirmative, if e.g. you wants to work with two monitors, in the context of this work however, this should not be the case.

29. Sound: Here you will find both various sound cards as well as drivers for Sound support offered, but these are not needed.

30. USB support: first here to USB-names briefly defined

Will be USB stands for Universal Serial Bus, up to 127 USB devices can

To a USB-port will be connected in a tree structure, the first USB-port is the root (root) of the tree, the periphery includes the branches and the interior nodes are special USB-devices, the so-called hubs, the definition is the Help file has been removed, and the present embedded computer 1.1 contains two USB ports, but they are not used, according to the option will be denied.

31. Bluetooth support: is the current alternative to infrared or

-Transmission. Here are up to 10 meters of distance in the free possible and 5 meters distance with objects in the transmission line, since such a communication is not relevant in this case, the option has been disabled.

32. Kernel hacking: this detailed error messages for the

Persons, the Geraetetreiber letter, supports.

Kernel profiling support: Provides under /proc/profile information is available, with which can be determined, how much time the core needs for certain functions.

Profiles shift count: Determines the Adressabstand, with which the individual from the core running commands in the /proc/profile will be listed.

To quote from the README-file from Linux:

The activation of the "kernel hacking" option would normally result in a bigger or slower core (or even both), this may the core even definitely unstable because some routines then be compiled a little differently, in order to target poor routines to crash and uncover this error in the core (is complaining about kmalloc()).

The core should be used completely normal, therefore you should here with "no" answers.

The options **library routines, plug in CPU scheduler that support resource**

Management Modules, Build options in the core of this work were not present

And according to in the Help files are not represented, and consequently they were all too clear.

5.2.2 . . Modules creating and installing

This is the configuration has been completed, you now have the additions with the command "make dep" were to be established, in the track the modules created (make modules) and installed (make install_modules) will be. These commands can be run in a row separated by semicolon.

Since everything is easily walked, it was the core (make bzImage) compiled the generated bzImage and core means is located in the subdirectory arch/i386/boot/, he was then to the boot directory with the name vmlinuz moved (you can also use the root directory), from where he can by adding in the GRUB-configuration file (GRUB is available for Grand Unified Boot loader, and the other alternative is to LILO) be booted, GRUB must be called again, so that the changes be accepted, this step could be under "yast" can conveniently be processed by the entry for Linux has been copied and pasted, and then the new entry to this core has been adjusted, so the path of the root partition, and the name of the new core.

At the beginning it was planned, the embedded computer IDE hard drive to add to this should be the first time the whole be configured and installed, because this under Suse could not be reached, was switched to RedHat. Here, the Flash Disk, and the special processor supports. There is a other elegant and professional alternative. This is the core of a host computer, in the next chapter should be elaborated this alternative.

5.3 Development Environment.



The test system consists of a desktop PC as a Embedded-Rechnerersatz , on the other software or even a different operating system may be installed, of course, you want a Linux distribution on the host computer (in this case it was RedHat) be installed with all development tools, then where the core prepared, tested, and for the test system is to be prepared.

By this is the variant of the core on the DATABACKUP not physically be present of the trial system, but transferred through the network connection, using NFS, for help, and the test system is only in the test phase as a replacement for the embedded computer, because he have a keyboard, graphics card and monitor.

In addition to environment is a blank floppy disk needs, on which the boot loader is installed, the the core in the memory to load.

1. **Step: as a syslinux bootloader has been elected, three reasons for this:**
 - 1 – Syslinux should be on every Linux already exist.
 - 2 – Syslinux cannot be on DOS-formatted floppy disks with the FAT file system Install.
 - 3 – The boot loader provides a configuration file is available from the Great benefit is.

The core itself must be copied to the disk, and this is done on the host computer, and then the test system to boot from the diskette.

Blank floppy disk can with "mformat a:" be formatted, if this is not the case, and then the following command to "syslinux /dev/fd0" as root, which on the floppy disk to install syslinux, as /dev/fd0 is the name for the floppy drive under Linux is, if syslinux is not installed, simply "yast" and install syslinux with source code.

Now if we could have been the test system from the disk boot (BIOS to adjust that the computer boot from the floppy drive), then you get an error message "Could not find kernel image", this indicates that SYSLINUX easily on the floppy disk has been installed, as SYSLINUX a file called Linux (Kernel image) expected to boot, if you otherwise called the core, there is the possibility, the prompt while holding down the Shift key or Alt key during the boot a different name or to pass parameters, and a different alternative, which also has been used here, is the syslinux.cfg configuration file, then where are the boot parameters have been specified.

The following boot parameters have been specified:

The root filesystem is not written to the disk, but as mentioned before, via Network File System (NFS) loaded from the host computer. SYSLINUX was instructs, the root file system, NFS-server to get the IP-address of the test system is determined via DHCP:
append root= /dev/nfs ip=dhcpd.

2. Step: now it has the core itself be transferred to the disk, as the kernel is compiled, we have seen in the last section, and the file bzImage was the result of the core configuration, testing of the addictions, the production and installation of modules and as last of the compilation of the kernel, this I have on the disk with the name linux stored to then of the floppy disk to start, this boot process then leads to a different error message again: "kernel panic: vfs: unable to mount root", what is the purpose of that syslinux.cfg configuration file was stored on the disk.

So that the boot is made possible from the host computer, the host must be both as a NFS-server, to the root filesystem to make available as well as DHCP

Server be configured to the test system a dynamic IP-address to be handed over.

If these steps are completed, the test system should be properly booted from the floppy disk, the core of the disk loaded, rootfs of NFS-server mounted, and the IP-address of the DHCP-server can be delivered, it was no longer necessary, carry out the above-mentioned points.

The last step in the installation of the operating system is the real, this is elaborated in the following section.

5.4 Real .

Modern applications usually have the requirement that all events in the automation system time must be synchronized to each other, which is why Echtzeiterweiterungen especially find increasing popularity of Linux in the industry, as well as in the universities.

With Echtzeitfaehigkeit requirements are meant to Zeitablaeufen on a computer have to do real-time applications, regardless of the cause may not be delayed or prevented. In the context of this work must be on the events of the Transceiverkarte (Send/board) for example are triggered, in a precisely defined time a reaction on the part of the actuators, i.e. , the entire airship, are made.

The Echtzeitfaehigkeit is deactivated by default on the computers and the major operating systems does not exist, but with RTAI and RTLinux exists the possibility, to expand Linux so that a small real-time core runs is available, the guaranteed guaranteed response times and one of the advantages of these Linux Echtzeiterweiterungen is that Linux in the normal case (if the real-time core runs just has nothing to do) with the normal pipped works.

42

5.4.1 . . latency and fluctuation

The result of these changes is the Unterbrechungslatenz and fluctuation (see Figure 5-12 and 5-13) between periodic interruptions in the microseconds and the area to reduce and so faster responses to external events, and a higher Timing-Aufloesung

to allow.

The pipped of Linux shows latency (latency) and turnover (jitter) of one millisecond, the Echtzeitversionen of Linux have latency and turnover some microseconds to the processors, with several hundred MHz running.

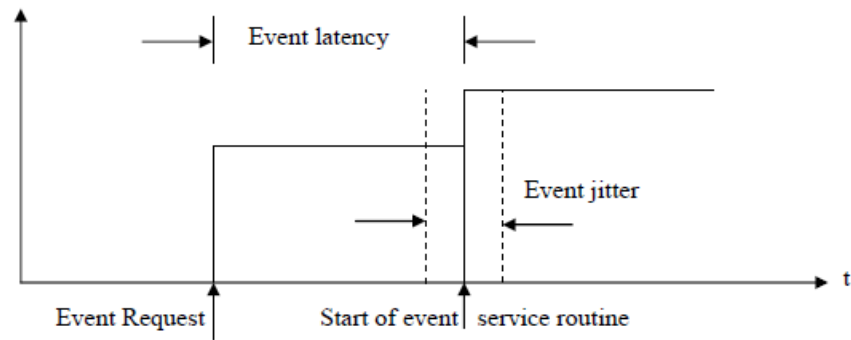


Figure 5- 12: The Ereignislatenz.

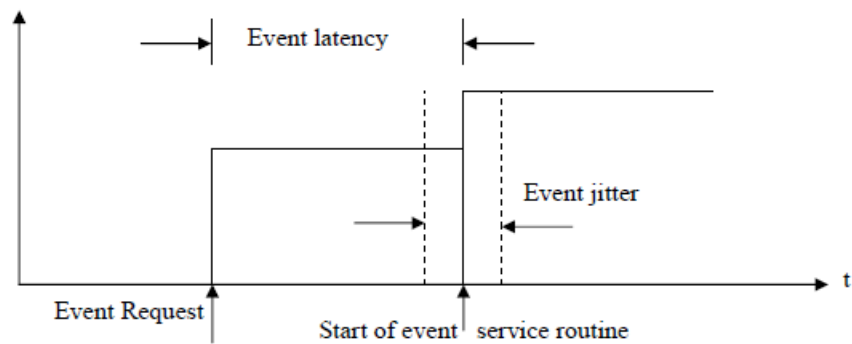


Figure 5- 13: periodic fluctuation.

The latency is the delay of an interruption to the beginning of the processing of this interruption, the difference between a specified and tatsaechlichem value is known as latency. (In practice, a desired time not always be exactly the same, also, as the system time for internal processes needs). The fluctuation is the change of the timing of the periodic cases or is the static distribution of latency.

Item 5.4.2 above hard and soft real-time conditions.

Are divided into the time-critical tasks in the following points (see Figure 5-14): soft real-time conditions: Response times must be low

His, however, a short-term exceeded certain limits no greater consequences. That means that the time is to be maintained in the funds.

Hard real-time conditions: certain response times by the system must be guaranteed under all circumstances, if they are exceeded, damage to the system on.

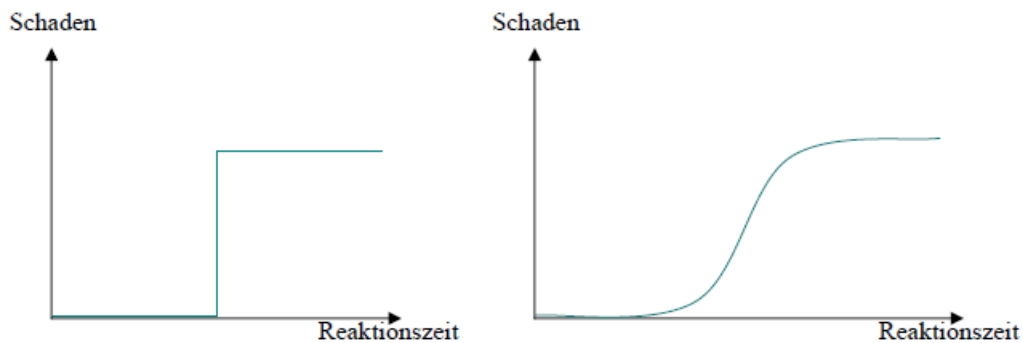


Figure 5- 14: Schadensänderung in dependence of the response time.

Linux is usually just like any other operating system, only for soft Suitable Echtzeitaufgaben (see Table 5-1).

	Application	Latency/ turnover
Standard BS.	No real time	10 μ s – 100 ms
Standard Linux	Soft real time	1 μ s
IEEE 1003.1d	Hard real time	10 – 100 μ s
Linux Microkernel	Hard real time	1 – 10 μ s
RTOS-Kernels	Hard real time	1 – 10 μ s

Table 5- 1: Comparison of the performance of Linux with the commercial Echtzeitkernen application latency/fluctuation

Standard BS. no real-time 10 μ s - 100 ms standard

Linux soft real-time 1 μ s

IEEE as 1003.1d hard real-time 10 - 100 μ s

Linux microkernel hard real-time 1 - 10 μ s

RTOS-cores hard real-time 1 - 10 μ s and that is the response time for various

reasons not deterministic:

- The core functions of the operating system are not interruptible, with the result that other requirements until the conclusion of the operation have to wait.
- The Zeitaufloesung on the x86 shall be 10ms, and is thus low e.g. for the Scheduler, the very important, even necessary for Echtzeitaufgaben is.
- The access to hard drives and on communication equipment has in principle no defined response time, can even the function to be called as long as block until all data are available [?].

It is possible, standard Linux for soft Echtzeitsteueranwendungen to

Use, in which the sampling time of approximately 10 ms commensurate long and the application is missing time schedules do allow it, this is an example using the video processing, where an occasional flare-up is often not perceived.

When data loss can be approved, then standard Linux (especially the latest cores) for soft real-time applications are used, the the Unterhaelfte (preemption logic) have.

In the piped Unterbrechungsverarbeitung work containing some is divided in two, as the Oberhaelften and be marked Unterhaelftenaufgaben. The Unterhaelfte meets the tasks, the disruption to edit, and data from the physical medium accommodate in a cache, the Oberhaelfte reads from the clipboard and sends the data to a accessible buffer of the core, in the piped (ICU) without, are all

Unsuitable interruptions, if the Unterhaelfte tasks. This means that it can be an unpredictable delay (latency), before a second interruption can be made.

If that is not the case, then the hard Echtzeitimplementierungen needs.

The Echtzeitvarianten of Linux that are usable for such applications, since 1997, and while it is true that there are mainly two implementations for the realization harder Echtzeitfaehigkeit under Linux:

- RT-Linux was in New Mexico by a working group to Professor Victor Yodaiken and Michael Barabanov developed. This variant is to a new layer of code in a highly efficient between the hardware and the pipped form the additional layer of code, called microkernel, controls the whole (total) Echtzeitfunktionalitaet, including interrupts, and

Sets high aufloesendes timing. This microkernel cannot be the Pipped
Run in the background.

- RTAI was in Milano (Politecnico di Milano) of a working group developed by Professor Paulo Maategazza. This is the second variant should Echtzeiterweiterungen to POSIX.1 within the structure of the Linux, Standardkerns imports. You adds Timer, scheduling (Scheduling) and

Unterbrechungskontrolle (Preemption) directly in a single core
Together.

These implementations will be as a supplement (patch) made available to the pipped. The microkernel starts the spends receiving (hardware interrupts) and also determines the Echtzeitaufgaben hoechstmoeglichen priority with the firmly.

Handles the real-time core runs as a direct layer of the hardware the Interruptverwaltung of the complete system from this real-time core runs the guaranteed response time for real-time programs, normally Linux worked double shifts (kernel space & User Space), but it is with the real time extension to a third layer (Realtime Space) advanced: this new layer then monitors the kernel space and thus has the possibility to interrupt the core code.

After the Echtzeitaufgaben have been processed, the realtime Space all registers again, so that nothing of the Normalkern noticed, that it was interrupted.

If the realtime space no Echtzeitaufgaben needs to do, needs no

Computing time and thus makes for the normal Linux kernel unnoticed,

comparison of the advantages and disadvantages of the two variants:

RTLinux: the development team of Victor Yodaiken will decide whether the patches of the users in the concept fit, otherwise they will be rejected patches, and while this is a consistent model achieved, but find no possibility innovative ideas in the software to be included.

RTAI: it is quite contrary to this other consistent model in which many users developments in the form of core modules have added to the Software. On the other hand this approach makes the whole system more flexible and innovative.

Although subject to both variants of the GNU-license, but at least RTAI is easy to get nowadays.

5.4.3 . . embedded applications

Real-time systems can also be embedded systems, embedded applications run usually in small hardware and/or cards, often without keyboards, Mauseen or monitors, usually with no rotating media such as hard disks or CD-ROMs, the hard working conditions could not resist and programrs can assemble Linux, so that it runs without these peripheral devices and there are several popular embedded Linux distributions, there are the pre-configured and useful tools for customer-related specialty products to include [11].

The commonly used platform for standard Linux is an Intel-based computer or a laptop computer, with the applications might not need real-time performance and the latency and the fluctuation of standard Linux no restrictions for these applications are, but there are embedded

Hardware components, the deterministic real-time performance and require for these cases must be one of the embedded application in the Linux-Echtzeitimplementierungen be used, so embedded Linux run on the hardware components from the desktop computers up to Videoarmbanduhren rich, but an embedded Linux is not necessarily Germany will present.

Since the normal Linuxkernel, real-time core runs not to be noticed, it must have the opportunity, real-time and normal processes to communicate with each other can be, and this is done with the following techniques: FIFOs, Shared Memory, mailboxes, and message queues.

5.4.4 . . Installation of Linux-Echtzeit

The steps, the of the series to be run in order to install a Germany will present, are to summarize as follows:

1. Selbstkonfigurierung of the patch:

Before you so that defended, the RTLinux to install them you have to get the corresponding Linux and RT Linux versions to download, in the present work has been working with the 2.4 kernel version, what this and the RT Linux version 3.1 have been downloaded (see download). As the C-compiler version also had to be observed, as otherwise problems can arise, and the GCC 2.7.2.3 is a minimum requirement, then the following steps must be taken:

- The downloaded files in a temporary folder must be cached, and under "`/var/tmp`".
- Prepare a working folder and empty when it is present: `md /usr/src/rtlinux` (`rm -rf /usr/src/rtlinux`), then unzip the downloaded files with the command: `tar -xzvf /var/tmp/linux-2.4.4 .tar.gz /var/tmp/rtlinux-3.1 .tar.gz`.
- The core, and the RT Linux-Ergaenzung in the subfolder linux mending: `patch -p1 < rtlinux/kernel_patch-2.4.4 . .`

- Then configure the core as follows, compile, and install, as in the last chapter: "make menuconfig; make dep; make bzImage; make modules; make modules_install;"
- The created file copy: cp arch/i386/boot/bzImage /boot/rlinux, and after the boot loader has been adapted, restart the computer.
- Now the computer with the new core restarted and simplification will be created a symbolic link: The folder

Ln -sf /usr/src/rlinux/Linux /usr/src/linux, in the then of the RTLinux with the command "make menuconfig; make dep; make; make and make install" devices configured and must be compiled.

- Now, in order to be able to run real-time programs, must be the RT Linux modules by the rlinux start command will be loaded.

2. An existing supplement (patch) use.

- In this case, it was with the downloaded file rlinux-2.4- prepached.tgz worked, so configured, dekomprimiertt, etc. otherwise not much was changed.

In both cases, had to be observed that the "hard" real-time in the Grundlagenkonfiguration set and the contrast that the APM support is disabled.

6. Description of Portzugriffsarten

There is a difference, as we will see in the following - between two I/O Portzugriffsarten: On the one side of the direct access, which is also under the name input/output falls deeper level (low level input and output), and the access of Geraetedeien on the other side.

In fact, hardware-near programming in modern, complex operating systems on modern computers unfavourable, requests to addresses outside of the assigned memory or I/O-ports are the "normal user" denied by the operating system, which previously was not the case, you could much easier access to the hardware and the maximum performance of the programs and/or reach a component.

Let us first look at the direct access type. In the following, this direct access on the example of the serial interface explained.

6.1 . Direct Access

Selfmade programs cannot readily directly on hardware access components, this is prevented by the operating system for security reasons, direct (write-) requests to the hardware of the computer to be able to system crashes and lead to data loss under the circumstances.

However, in order to allow a direct access, must be given access rights, which create the prerequisite for that then the actual requests government expenditures can be.

6.1.1 . Rights awarded

In Linux, there are in addition to the access rights read (R), write (W) and execute (x) and the right "set user ID" (S), if this bit for an executable file is set, then receives the exporting process for the duration of the execution, the user-ID of this file, i.e. if a program the user

"Root" belongs, then the exporting process root-privileges, for example, with the command line "chown root:root named myprog" the "owner" and the user group of the file and changed with "chmod a+s named myprog" set the bit of the user, and if now the program "named myprog" from the normal user (of course the user should be allowed to run the program) is started, then it is running with root privileges, and can again corresponding damage, and this is why such a program should these privileges with the help of the function call "setuid (getuid()); (#include <sys/types.h> and #include <unistd.h>)" return it as soon as you are no longer needed.

An additional safeguard against "unintended" requests to the hardware components of the computer is that the user root yourself first, the access to the hardware is denied (segmentation fault). The two functions IOPL() and ioperm() are meant for this, the I/O-Portzugriffe explicitly release. These functions (or one of the two) must be at the beginning (before that I/O-port access) of the program will be called the Zugriffsbibliotheken on the I/O-ports are from the header file /usr/include/asm/io.h provided. IOPL is available for I/O privileged level and ioperm for I/O permissions. Both of these functions return the following values: the value of 0 on success, -1 on failure, in the latter case, the Fehlerkonstante on errno is set.

Don't forget locks a setuid() is not the a normal user to do just that, by the ioperm() has been granted, but a fork() (the child receives no access, but the parent keeps him) could be used for this purpose.

The syntax of the first function is int ioperm(unsigned long from, unsigned long num, int turn_on). It is from the first port number to be accessed, and num the number of subsequent ports. For example, would call ioperm(0x3E8, 16,

1. Access to the Ports 0x3E8 to 0x3F8 grant (with 3F8 and 3E8 for the first and second serial interface). The last argument is a boolean value that is either the Program Access to the Ports (true (1)) or prohibits (false (0)). We must call ioperm() repeatedly to not-following ports to allow multiple access.

You can withdraw the administrator again, after call `ioperm()` was called, in order to gain access to the ports to be used to allow, at the end of the program you will not be asked to the Portzugangsrechte with `ioperm(... , 0)` to remove explicitly; this is done automatically, when the process is terminated.

The function call `ioperm()` can only access to the Ports 0x000 to 0x3ff give. For additional ports must be `IOPL()`, the access to all ports immediately guaranteed according to use, a limitation of the address range is unfortunately not possible, but you can the level of the expert panel on the I/O-ports from 0 (no permission) to 3 (full access) rank.

6.1.2 . . programming/Issue

Once the conditions are met, you can spend with the actual start and/or read, but are the hardware components of the computer not just like the RAM on "normal" memory accesses will be accessible, but only with the aid of special (assembler) commands accessible for C-programs are a series of macros available to these commands to C-map features, and these features are Inline-Makros : i.e. `#include <asm/io.h>` is entirely sufficient, in order to involve you, and needs no additional libraries, Table 2.1 gives an overview of the features.

In order to have a byte (8 bits) to an I/O-port to spend, you need the function `outb(Port, value)` in the correct order of the parameters (e.g. , under DOS are in the reverse order you enter) to call up to read from a port in reverse, there is the function `unsigned char inb(unsigned short int from)` to use, you are the bytes that you receive has, and the most units are designed to do just that byteweisen. All Portkommunikations way, instructions need to run at least about a microsecond. The with the suffix "_P" reasoned macros work almost identical to the other macros, but you grant an additional short delay (about a microsecond) after the do just that, you can delay of about 4 microseconds with `#define REALLY_SLOW_IO` before the `#include <asm/io.h>`

Force. The macros (unless you use `#define SLOW_I0_BY_JUMPING`, probably the less exactly is) normally use a port output is switched to 0x80 for your delay, so you must first access to the Port 0x80 with `ioperm()`. Give. outputs to port 0x80 should not affect part of the system.

macro	description
inb (Adresse) inw (Adresse) inl (Adresse) insb (Adresse) insw (Adresse) insl (Adresse)	The contents of the I / O area on the Address address is read. Here, depending on the extension of the Macros, or one byte (b), Word (w) Long-word returned (1), optionally with sign
outb (Wert, Adresse) outl (Wert, Adresse) outsb (Wert, Adresse) outsw (Wert, Adresse) outw (Wert, Adresse) outsl (Wert, Adresse)	The value of value is in the I / upper calibration address to the address written. Here, value per after ending the macro as a byte (b) Word (u) or Long-Word (1) interpreted with either too sign
inb_p(Wert, Adresse) inw_p(Wert, Adresse) inl_p(Wert, Adresse) outb_p(Wert, Adresse) outw_p(Wert, Adresse) Outl_p(Wert, Adresse)	In contrast to the macros without „_p is "for these functions, the Execution delayed so "Slower" hardware bill to wear

Table 6- 1: macros to access I/O-ports.

Due to a limitation in the GNU C-compiler (present in all versions, including the egcs), must be each (possible) source program with the `-o` option will be compiled for the optimization (`gcc -O1` or higher), as an alternative, you can `#define` external static will be inserted before `#include <asm/io.h>` is included (ultimately, the instruction `#undef` externally equal then not to be forgotten).

For debugging gcc with the command - G - 0 be called (in any case in modern versions of gcc), although the optimization the Debugger sometimes strange behavior cannot be, you can work around this by the program, the The I/O-used do just that, in a separate source file is saved, and only for the optimization is compiled with.

To begin the programming, the data via the serial interface is not correct, partially not at all, will be sent and received, and then the cable should be tested, by using Standardkommunikations programs has been attempted, data exchange with the null modem cable, and the Standardkommunikationsprogramme are to a "Hyperterminal" on Windows and to the other "minicom" under Linux, and if you two computers with Windows Operating System or Linux via the serial interface connects and Hyperterminal or MINICOM starts, then one has to be restored, i.e. baud rate, Datenlaenge, Paritaetsart, stop bit and Data Flow Control with values to, after the right COM-port was selected, and the baud rate receives values between 110 and 921,600 bits per second, the flow control can be either hardwaremaessig, address decoder unit or but not be present, in the cable was no longer in order and should be replaced, and it was that RS-232 no USB-port is, what means that the cable during operation should not be switched, so only when the computer is turned off, do we have the right to use the cable from the serial port Remove, and with the new cable could easily the Standardkommunikationsprogramme send and receive data, even "Hyperterminal" could be with "minicom" replace data, you must be careful only to initialize, e.g. , Linux does not support 1.5 bits, stop bits, and if the baud rate is set to 115200, the data will then be of Hyperterminal to minicom sent correctly, but in the other direction runs the not. In 6 bits of data are transferred correctly only numbers, the characters (letters) unfortunately, not the data are transferred by one character, which to a non-canonical and non-blocking communication indicating.

6.2 Access via Geraetedateien

The other method I/O-ports to make accessible, is the "Device" /dev/port open to - similar to what normal files either to read, write, or simultaneously to both the stdio functions `f * ()`, which usually for Blockgeraete (block device) are intended, can be used here, too, but are due to an internal buffering to avoid the Geraetedateifunktionen such as `open()` are available, as the ports under Zeichengerate (character device) can be divided.

It must also of course read/write permissions on /dev/port be present, and even if that condition is met, by the access to /dev/port is released, that means that this method does not yet beginning `ioperm()` and (even) are not "root" -access needs, but this type of port sharing in relation to system security is not recommended, because it a system breach

It is possible, it can even access to the root "surreptitiously" by putting it /dev/port used to hard drives, network cards, etc, directly accessible power.

This method is far more extensive and finds frequent use in the development of (any hardware access) of hardware-based access, it also offers much more possibilities in the handling of threads, and real-time, and it is not possible `select` (p. 2 `select`) or `poll`(you 2 `poll`) to use /dev/port to read, because the hardware no possibilities to remember of the statuses of the CPU has, if a value changes in an input port.

6.2.1), open and close

In order to provide data on a serial interface to be able send or receive, they must be opened first. This is the function `int open(const char * pathname, int openflag / * ,mode_t mode * /)` is available, this function you determine, whether from this (or to this) interface only is read or written or both at the same time, and this is done with the flags: `O_RDONLY`, `O_WRONLY` or `O_RDWR`.

In this case, you must be read once (the single-board computer, the Sonnenstrahlstaerke, the speed, the direction, the temperature, the angular velocity and the Azimutgeschwindigkeit received from the sensors, reading), once simultaneously read and write (the Transceiver passes the values to the single-board computer and reads the desired values from the sensors from). Finally will be read by the single-board computer communicated values of actuators, i.e. it is only written.

The following code example shows this step with the open() -Function:

```
#Include <sys/types.h> // Definition of primitive system data types
#include <sys/stat.h> // File Status
#include <fcntl.h> // elementary E- /A-operations
#include <errno.h> // with Konstantennummer Fehlerkonstantendefinition

#include <string.h> // macros and functions for Zeichenkettenbearbeitung int fd;

FD = open( "/dev/ttyS0 ", O_WRONLY);

If (fd < 0)
Printf( "Error ,errno , "s% ,d% ,1of one COM ffnenض
else ;((errno)perror
    Printf( "COM1 has been successfully opened. \n" );
```

This function returns the file descriptor fd back in case of success, in the other case the function returns the value -1 is returned.

It is important to note that this code the serial port COM1 to write opens. is written but it was only with the function "write". This function is explained in more detail below.

More important for this task, but optional constants are the following: O_NOCTTY:
The open terminal should not be the control terminal of the process.

O_NONBLOCK: The open serial port, on subsequent I/O operations is not blocked.

O_ASYNC: Asynchronous input/output, it generates a signal (SIGIO) O_SYNC: After each letter with the function write() is to be waiting for the write operation is completely finished. This means that each call to the function write() First of all, all data completely to the physical medium writes, before any of the other steps are possible.

Once the communication is completed, you have the serial interface to be used for other applications or users share, this is done by the serial interface with the Int function close(int fd) closes. This function returns -1 if the port does not close cannot be, otherwise it returns 0 on success back, by a small example:

```
#Include <unistd.h> /* Reduced UNIX Standard Library & symbolic constants */
#include <errno.h> // with Konstantennummer Fehlerkonstantendefinition

#include <string.h> // macros and functions for Zeichenkettenbearbeitung if

((close(fd)) < 0)
    Printf( "Could not close the port. \n %d, %s", errno, perror(errno));
```

In the context of this work was the first serial interface closed as a precaution, to then to open again, because otherwise the incoming data is not properly and completely would be received, at the end of the program, the serial interface are not closed, because the program endless to send and receive data.

Item 6.2.2 , . canonical and non-canonical Input/Output

The function ssize_t write(int fd, const void * buffer, size_t nbytes of the) is actually the number of bytes written on success back. This number is way is not necessarily equal to the variable "nbytes of the", you can be less, indicate what is to not error, but only that the time has arrived no other data are. In error The function returns -1 back. In POSIX.1 corresponds to ssize_t signed integer and unsigned integer corresponds to

size_t, what our write-function in int write(int fd, char * buffer, int nbytes of the converts.

Code sample:

```
#Include <stdio.h> // Standard Input/Output functions
#include <unistd.h> /* Reduced UNIX Standard Library & symbolic constants */

Int fd, iBuf;
Char buffer[ ] = "Now the output test";

If ((iBuf = (write(fd, buffer, sizeof(buffer))) ) < 0)
    Printf ( "Could not write on fd! " );
Else if (iBuf == sizeof(buffer))
    Printf ( "All characters were successfully written! " );
Else
    Printf( " %d characters could only write! ", iBuf);
```

A distinction can be two possibilities of communication: the one of the canonical mode, and for other of the non-canonical mode.

Of the canonical mode means that the whole line at once sent (letter) or received (read) will be, in the work with normal files one speaks of buffered input/output, and the terminals, which are used in the canonical mode are deactivated by default preconfigured. When writing to the port for the time being this will only be the characters stored in the buffer, until either the full or to the end of the line is, which means only then will the contents of the buffer was actually transferred to the physical medium, if a linefeed LF(ASCII), new line or carriage NL retrun CR occurs. When closing the ports is the rest of the Puffer-Inhalts only written automatically, then the memory for the buffer released. In this mode you have to have the flag in the local constant ICANON set c_lflag. It is called the canonical mode also blocking, because of a read.

/Write-Operation of the application returns the control only, if a line of text has been entered.

In the context of this work was written this code sample, in order to use the canonical mode:

```
#Include <stdio.h> // Standard Input/Output functions
#include <stdlib.h> / * Reduced standard C library & useful general
functions * /
#include <unistd.h> / * Reduced UNIX Standard Library & symbolic constants * /
#include <string.h> // macros and functions for
Zeichenkettenbearbeitung
#include <fcntl.h> // elementary E- /A-operations
#include <termios.h> // POSIX terminal functions and constants #include
<sys/types.h> // System Data Types Definitions of the primitive #include
<sys/stat.h> // File Status

#include <errno.h> // Fehlerkonstantendefinition with Konstantennummer

#define FALSE 0
#define TRUE 1

int STOP = FALSE;

void init_port(int

FD)
{
Struct termios tio;
```

```
Tcgetattr(fd, &tio);

Tio.c_cflag = 0;
tio.c_iflag = 0;
tio.c_oflag = 0;
tio.c_lflag = 0;

Tio.c_cflag |= B38400;
Tio.c_cflag |= CRTSCTS;
Tio.c_cflag |= CS8;
Tio.c_cflag |= (CLOCAL | CREAD);

Tio.c_iflag = IGNPAR | ICRNL;
Tio.c_oflag = 0;
Tio.c_lflag = ICANON;

Tcflush(fd, TCIFLUSH);
Tcsetattr(fd, TCSANOW, &tio);
}

Int main()
{
    Int fd, lout;
    Char buffer[] = "time a little bit send! ? ! ";

    FD = open( " /dev/ttyS0 ", O_RDWR | O_NOCTTY);
    If (fd < 0)
    {
        perror( " /dev/ttyS0 " );
        Exit(-1);
    }

    Init_port(fd);
```

```

While (STOP= =false)
{
    lout = write(fd, buffer, sizeof(buffer));
    If (lout < 0)
    {
Printf( "on COM0 can not write: %d, %s\n", errno, strerror(errno));
        Exit(-1);
    }

    Buffer[lout] = 0;

    Printf( " %d and has been written exactly %s\n", lout, buffer); if
    (buffer[ 0] == 'q') stop = TRUE;
}

Close(fd);
return 0;
}

```

A terminal has 5 different Modi-Eigenschaften : Input Mode (iflag), output mode (oflag), control mode (he says), Local Mode (lflag) and control characters (cc), each these modes manages a large number of symbolic constants, such as the following table shows.

component	description
c_iflag	input flag
BRKINT	Generate SIGINT at break
IGNBRK / IGNPAR	Ignoring break / parity errors
INPCK	Parity check allow
IGNCR	Ignorieren von Carriage Return
ICRNL	Converting CR to NL on input
INLCR	Switching on the parity check when entering
IXON / IXOFF	Flow control, enable software-
PARMARK	Mark parity errors
c_oflag	issuance flag
OPOST	Provide an implementation-defined output type.

	If this option is disabled, then all other ignores options in c_oflag.
ONLCR	NL to CR-NL Paare abbilden.
c_cflag	control flag
B0-B115200	Terminal baud rate set.
CLOCAL	Ignore the modem status symbol or terminal only operate locally.
CREAD	Character can be received.
CRTSCTS	Enable hardware flow control
CSIZE	Number of bits per character: CS5-8 for 5 to 8 bits per byte
CSTOP	Use two stop bits. By default, one stop bit.
PARENB	Parity generation for output and Enable parity checking for input.
PARODD	Odd parity enable, otherwise just default.
c_lflag	Local flags
ECHO	Each character you spend on the terminal.
ICANON	The canonical (row-oriented) mode switch.
ISIG	Generate a corresponding signal, when Terminal control characters are entered (arrived).

Table 6- 2: The five different Modi-Eigenschaften a Terminal Component

Description

C_iflag Eingabeflag

BRKINT generate interruptible with SIGINT to break

IGNBRK / IGNPAR ignore break/ Paritaetsfehlern

Allow INPCK Paritaetsprufung

IGNCR ignore of Carriage Return

Convert ICRNL of CR in the NL when entering INLCR Paritaetsprufung

switch on when you enter the IXON / IXOFF address decoder unit data

flow allow highlight of Paritaetsfehlern PARMARK

C_oflag Ausgabeflag

Allow a OPOST implementierungsdefinierten output type, if this option is locked, then all other

Options in c_oflag ignored.

ONLCR NL in CR-NL couples map.

C_cflag Kontrollflag

B0-B115200 set baud rate of the terminal.

Ignore the CLOCAL Modemstatuszeichen or the terminal only operate locally.

CREAD characters can be received.

Flow control hardwaremaessig CRTSCTS enable CSIZE bits per character:

CS5-8 for 5 to 8 bits per byte cstop use two stop bits, deactivated by default a

stop bit. PARENB Paritaetserzeugung Paritaetsprufung for the issue and allow

for the input.

Odd parity enable PARODD, otherwise it is just the default.

c_lflag Local flags

Also ECHO each character typed on the terminal output.

ICANON the canonical (CLI) mode, ISIG generate appropriate signal, if

Terminalsteuerzeichen has occurred (arrived) are.

c_cc[NCCS] Control characters

Vmin to read the minimum byte count before a read operation returns.

VTIME the minimum time that is to be serviced, up to a read operation returns.

The function `tcgetattr(int fd, struct termios * tio)` allows for it, the `tio` to determine stored Terminalattribute. The function `tcsetattr(int fd, int flag, const struct termios * tio)` in contrast, the previously set flags, where `fd` for a file descriptor is to be, for example, if the symbolic constant `TSCANOW` is used, this means that the changes are to be activated immediately. In contrast, `TCSAFLUSH` the result is that the only changes are to activate, after all pending expenditure were transferred.

The `tcflush` function(`fd, flag`) empties the A- /Output Buffer of the with `fd` open Geraetedatei. This function are, in turn different symbolic constants are available: `TCIFLUSH`, `TCOFLUSH` and `TCIOFLUSH`. You have the function that either one, off or but a and output buffers are emptied, and the still in the respective buffers the data will be deleted without processing.

The function `ssize_t read(int fd, void * buffer, size_t nbytes of the)` to read similar the function `write()`, if you the header file `<unistd.h>` slipstreams, then it is defined as follows in POSIX.1 `int read(int fd, char * buffer, int nbytes of the)` and returns the number of bytes read back on success, the value 0 if EOF, otherwise a value of -1 is returned.

When reading from a port (Zeichengeraet) with the functions `read()` or `pread()` is the number of characters to read actually often less than the `nbytes` of the specified number. This is just that at the moment no further characters are available [2].

The non-canonical mode, however, the input character immediately executed, i.e. without buffering immediately forwarded. This is either a certain number of bytes or after a certain period of time, the bytes is issued, the arrived. For this purpose the Array `c_cc[]` (see Table 6-3) available to the `termios` structure.

	MIN > 0	MIN == 0
TIME > 0	The read operation returns at least MIN bytes when TIME is not yet	The read operation returns 1 and the number of required bytes

	expired. It provides 1 or MIN Bytes if TIME has expired. The timer is the only first byte read is started. So here is an infinite Blocking can be achieved.	if TIME has not expired is. Otherwise it returns 0. The timer will start the read operation started
TIME == 0	The read operation returns MIN bytes if they exist. Blocking also possible if No MIN bytes are delivered.	The read operation returns 0 or the number of required bytes. She returns in any case without Wait any immediate return.

Table 6- 3: Possible Steuerzeichenvarianten for the non-canonical Input Min > 0
min =0

Time > 0, the read operation provides at least MIN Bytes, if time is not yet Has Expired, and provides 1 or MIN Bytes, when time has expired.	The Read operation returns 1 bytes or the number required, If time has not yet expired Is, otherwise provides you 0 back.
The timer is only started when first read bytes, therefore here can be achieved an infinite block.	The timer is started at the beginning of the read operation
Time =0 the read operation provides MIN bytes if they are present. Blocking also possible, if no MIN bytes will be delivered.	The Read operation returns 0 or the number of bytes, you will return to every case without any wait back immediately.

Non-canonical mode also means that an application can perform other tasks, as long as it is on I/O waits. For this reason, this type also non-blocking called.

The following example illustrates a part of the in the context of this work developed codes in the non-canonical mode:

```
#Include <stdio.h> // Standard Input/Output functions
#include <stdlib.h> / * Reduced standard C library & useful general
functions * /
```



```
#Include <unistd.h> / * Reduced UNIX Standard Library & symbolic constants * /  
#Include <string.h> // macros and functions for Zeichenkettenbearbeitung #include  
<fcntl.h> // elementary E- /A-operations  
#Include <termios.h> // POSIX terminal functions, and constant #include  
<sys/types.h> // Definition of primitive system data types #include  
<sys/stat.h> // File Status  
#Include <errno.h> // with Konstantennummer Fehlerkonstantendefinition
```

```
Struct coordinates {
    Short speed;
    Short direction;
    Short temperature;
};

Void init_port(int FD)
{

    Struct termios tio;

    tcgetattr(fd, &tio);
    Bzero( &tio, sizeof(tio));

    Cfsetospeed ( &tio, B38400);

    Tio.c_cflag |= PARENB;
    Tio.c_cflag &= ~PARODD;
    Tio.c_cflag &= ~cstop;
    Tio.c_cflag &= ~CSIZE;
    Tio.c_cflag |= CRTSCTS;
    Tio.c_cflag |= CS8;
    Tio.c_cflag |= (CLOCAL | CREAD);

    Tio.c_iflag = IGNPAR;
    Tio.c_oflag = 0;
    tio.c_lflag = 0;
    Tio.c_cc[VMIN] = 5;
    Tio.c_cc[VTIME] = 0;

    Tcflush(fd, TCOFLUSH);
    Tcsetattr(fd, TCSANOW, &tio);
}
```

```

Int main()
{
    Int fd, iIn;
    Struct coordinates * coordinates;
    Char buffer[ 255] ;

    FD = open( " /dev/ttyS0 ", O_RDWR | O_NOCTTY);
    If (fd < 0)
    {
        fprintf(stderr, "Can COM0 will not be opened. \n" );
        Exit(-1);
    }
    Else
    {
        Init_port(fd);

        iIn = write(fd, buffer, sizeof(buffer));
        If (iIn < 0)
        {
            Printf( "on COM0 can not write: %d, %s\n", errno, strerror(errno)); exit(-
            1);
        }

        Buffer[iIn] = 0;
        Printf( " %d, and have been read exactly %s\n", iIn, buffer);
    }

    Close(fd);
    return 0;
}

```

The function `bzero()` sets the first `n` (`sizeof()`) bytes from the beginning of the terminal `tio` to zero.

POSIX.1 offers the following summarized functions to ask and/or change the terminal properties: `tcgetattr`, `tcsetattr`, `cfgeti(o)speed`, `cfset(i)ospeed` and `tclflush`.

Table 6- 4: POSIX-functions to query and change the Terminalattribute.

`Tcgetattr` Ask for the Terminalattribute
`Tcsetattr` Set the Terminalattribute

`Cfgetispeed` Ask for the input speed
`Cfgetospeed` Ask for the output speed
`Cfsetispeed` Set the input speed
`Cfsetospeed` Set the output speed

Empty the `tclflush` A and/or output buffer

Function	description
<code>tcgetattr</code> <code>tcsetattr</code>	Check the terminal attributes Set the terminal attributes
<code>cfgetispeed</code> <code>cfgetospeed</code> <code>cfsetispeed</code> <code>cfsetospeed</code>	Call the input speed Check the output speed Set the input speed Set the output rate
<code>tclflush</code>	Empty the input and / or output buffer

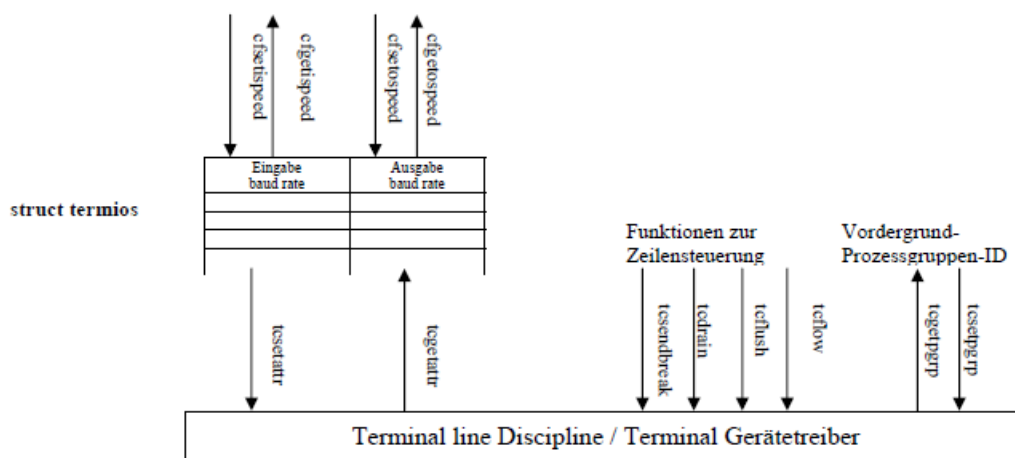


Figure 6- 1: Terminal E/A-functions in the overview.

The function `cfgeti(o)speed` obtained from the A- /output speed, but it must be called only after the `termios` structure with the function `tcgetattr()` has been determined.

The function `cfsetospeed` sets the baud rate in this case the value B38400. Here also applies that the with `cfseti(o)speed`-functions set changes not be adjusted until the function `tcsetattr()` is called.

6.2.3 . . synchronous and asynchronous transfer

The canonical or non-canonical mode can both synchronously and asynchronously be programd.

So that the computer understands the incoming serial data, he needs a reference to the place where a character ends and the following begins, in the Asynchronous Mode Seriidatenleitung remains in the high-status (1) is transferred to a character, each character is a start bit progress and will immediately of each bit in the character followed - optional of a Paritaetsbit and at least one or two stop bits.

The start bit is always a low-signal (0) and tells the computer that new series data are present, data can be sent or received at any time.

The duration of an individual bits by a clock is determined in the transmitter, by with the next clock pulse will be the next bit is output in the incoming bit pattern recipients is to the sender with a scanned asynchronous clock. If the recipients to the next character is waiting, it recognizes this scanning the digits like 1-0-edge of the start bits and knows that now is a new character, between the sender and recipients must be the number of bits per character and with the regular transfer rate and the width of a bits in the manufacture of the connection be firmly agreed.

The recipients know then, what is to be expected, and so can always be approximately in the Bitmitte palpation and the values of the individual bits of a character determine the optional Paritaetsbit is a simple sum of the bits, the view, whether the data is a even or odd number of 1-bits contain. With Straight parity is the Paritaetsbit 0, when there is an even number of ones in the characters, with odd parity is the Paritaetsbit 0, when there is an odd number of ones in

The data is there. In addition, there are still three more Paritaetsvarianten: Raumparitaet (space parity), where the Paritaetsbit is always 0, and the second type is Markierungsparitaet (mark parity) called, with her is the Paritaetsbit is always 1. The last version (no parity) sets no Paritaetsbit.

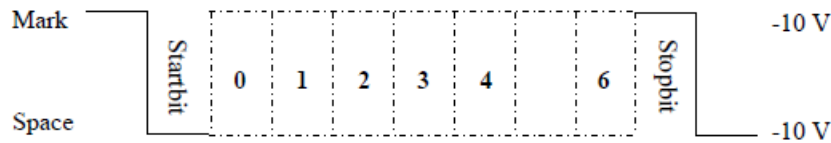


Figure 6- 2: asynchronous data transfer with 7 bits Datenlaenge.

It can be 1 or 2 stop bits between the characters to be and you always have a value of 1 (high-signal). This uebertragungsart is to remember that three bits each eight data bits are added, and that therefore, of course, the uebertragungseffizienz is reduced, and the time to transfer the "bagged" eight bits is to 3/8 longer than the time that one for "naked" would have needed eight bits, which is the time for the transfer of data will be lost, with longer telegrams is therefore offers a synchronous transfer to, when asynchronous transfer (returns) is the reading (immediately) returned immediately and sends a signal to the calling program, the following an application of the non-synchronous communication:

```
#Include <termios.h>
#include <stdlib.h>          /* Reduced standard C library & useful general functions */
#include <stdio.h> // Standard Input/Output functions
#include <unistd.h> //      reduced UNIX Standard Library & symbolic constants
#include <fcntl.h> // elementary E- /A-operations
#include <sys/signal.h> // macros and functions to intercept Signals
#include <errno.h> // with Konstantennummer Fehlerkonstantendefinition

#include <string.h> // macros and functions for Zeichenkettenbearbeitung #define

FALSE 0

#define TRUE 1

Int STOP=FALSE;
```

```
Int received=TRUE;
```

```
Void signal_handler (int s)
```

```
{
    Printf( "SIGIO receive. \n" );
    Received = FALSE;
}
```

```
Int main(int argc, char * argv[ ])
```

```
{
    Int fd, iln, i, key;
    Char CIN, message[ 90];
    Struct termios oldtio, newtio;
    Struct sigaction SIGIO;
    Char buffer[ 255];

    If (argc != 2)
    {
        Printf( "Please Port Number Ewhen ! \n" );
        Exit (-1) ;
    }
    Else
    {
        FD = open(argv[ 1], O_RDWR | O_NOCTTY
        O_NONBLOCK); if (fd < 0)
        {
            Printf( "Error ,errno ,\n\s%d % :ffnenض
            ;(1-)exit ;((errno)sterror
        }

        SIGIO.sa_handler = signal_handler;
        Sigemptyset( &SIGIO.sa_mask);
        SIGIO.sa_flags = 0;
        SIGIO.sa_restorer = NULL;
```

```
sigaction(SIGIO, &SIGIO, NULL);
```

```
FCNTL(fd, F_SETOWN, getpid( ));
```

```
FCNTL(fd, F_SETFL, FASYNC);
```

```
Tcgetattr(fd, &oldtio);
```

```
Newtio.c_cflag = B38400 | | CRTSCTS CLOCAL CS8 | | CREAD;
```

```
newtio.c_iflag = IGNPAR;
```

```
Newtio.c_oflag = 0;
```

```
newtio.c_lflag = 0;
```

```
Newtio.c_cc[VMIN] = 1;
```

```
Newtio.c_cc[VTIME] =
```

```
0;          tcflush(fd,
```

```
TCIFLUSH);
```

```
Tcsetattr(fd, TCSANOW, &newtio);
```

```
tcsetattr(1, TCSANOW, &newtio);
```

```
While (stop == false)
```

```
{
  If ((key = getc(STDIN_FILENO)) == true)
  {
    Switch (key)
    {
      Case 0x1b: /* Esc */
        STOP = TRUE;
        Break;
      Default:
        Write(fd, &key, 1);
        Break;
    }
  }
  If (received == false)
```


{

```

lin = read(fd, buffer, 255);
If (lin <= 0)
{
    Printf( "Error   ,errno   ,"n\s%d   %   :ffnenض
           ;(1-)exit ;((errno)strerror
}
Else
{
    For (i= 0; i<lin; i++) //for all chars in string
    {
        CIN = buffer[i];
        If ((CIN< 32) || (CIN>125))
        {
            sprintf(message, " %x", CIN);
            fputs(message, STDOUT_FILENO);
        }
        Else fputc ((int) CIN, STDOUT_FILENO);
    }
}

    Received = TRUE;
}
}
Tcsetattr(fd, TCSANOW, &oldtio);
Close(fd);
}
Return 0;
}

```

The transfer is deactivated by default synchronous (also known as blocking), i.e. with the reading will be maintained as long as data to be available to spend and/or read, unlike the asynchronous data will appear the synchronous data as a constant stream. to the data on the data line to read, the computer must have a common Bittaktgeber or made to provide, so that the sender and the recipients will be synchronized as synchronous protocols no character by character Synchronisierungsbit use, usually for at least a 25% improvement compared to the asynchronous communication and are for connections with more than two serial interfaces better suited.

In spite of the gains of the synchronous communication method support most RS-232 this procedure is not due to the high expense of hardware and software.

Characteristic for the synchronous data transmission is that the recipients the same clock speed as the sender uses in every telegram and his measure a synchronization with the of the transmitter performs to to be able to scan correctly.

6.3 . Threadprogrammierbeispiel

The threads-programming happens to a with the POSIX-threads, and the second possibility is the programming with the help of the common API of RTLinux, and each of these variants has advantages and disadvantages, the RTLinux API is productive as POSIX, but has the disadvantage that it is not in other operating systems cannot be.

It is still to mention that the system can crash, for example, if the collaborative process saturates the entire processor resources, i.e. no more time to run Linux (user space) has, in general, it is the real-time programming (kernel space) with a very large care to perform, as they do not offer security to Systemintegrität.

If you would like to create threads, then it is a the function

```
Int pthread_create(pthread_t * threadID, pthread_attr_t * attr, void * (* threadFct) (void * ),
```

Void * arg), this function returns 0 on success or is there a back in the other case Exxx-Wert back. When creating the threads is him a unique thread ID that is assigned to the pthread_t is one with the function pthread_self(void) can be obtained from the threads are in the Standard POSIX.1b specified and under the Herder, file pthread.h (prepared).

```
#Include <pthread.h>

Struct thread_data {
    short speed;
    Short direction;
    Short temperature;
};

Void * read(void * data);
Void * Letter(void * data);

Int main(void)
{
    pthread_t is assigned id1, ID2;
    Struct thread_data data1, data2;

    Data1.speed = " 123.4 ";
    Data1.direction = "35.64 ";
    Data1.temperature = 40;
    Pthread_create( &assigned id1, NULL, &writing, &data1);

    Pthread_create( &numcount id 2, NULL, &read; &data2);

    Pthread_join(assigned id1, NULL);
    Pthread_join(numcount id 2, NULL);

    Return 0;
}

Void * Letter(void * data)
{
    Int fd, Iin;
    Char buffer[ 255];

    FD = open( " /dev/ttyS0 ", O_RDWR | O_NONBLOCK);
    If (fd < 0) error message;
    Iin = write(fd, buffer, 255);
    if (IIN < 0) error message;

    Buffer[iIn] = 0;
    fprintf(stderr, "were written %d characters \n", IIN);
    Return 0;
}
```

In the present work, the thread parameter from the serial interface will be handed over, this is the fourth parameter arg function in the pthread_create() is available.

The pthread_join(pthread_t is Id, void ** retval) suspended the implementation of the paged thread until the thread with the identifier id is finished.

RT-Linux essentially consists of the following modules [?]:

- Rtl_sched.o: implies the System matching Echtzeitscheduler (single or dual processor system).
 - Rtl_fifo.o: API to manage the real-time FIFOs.
 - Rtl_posixio.o: Application API to Geraeteverwaltung.
 - Rtl_time.o: API for real time (accuracy in Nanosekunden). provides, the timer is available.
 - RTL.o: Echtzeit-Mikrokern , the the basic framework and provides the Interrupthandler.
-
- A real time application is in the form of a Kern-Moduls programd, what also is known as a collaborative process.
 - A collaborative process is mostly built up from various Echtzeitteilen, the quasi-run in parallel, the scheduler is the task, depending on the various computing time to assign priority.
 - This Echtzeitteile are in accordance with C-functions, the threads are called and a Echtzeitcode include, the timely, with guaranteed response time, must be performed.

How to get out of the C-programming knows, the processor looks (Praeprozessor) first, a main function; when the Echtzeitmodulen are there two functions, which are expected, and although init_module(), the loading of the module is called and is used to set up the function of the Echtzeitprozesses and cleanup_module(), which guaranteed the opposite, that is in the unloading the module is called and the clean-up is used.

12.5 Literature

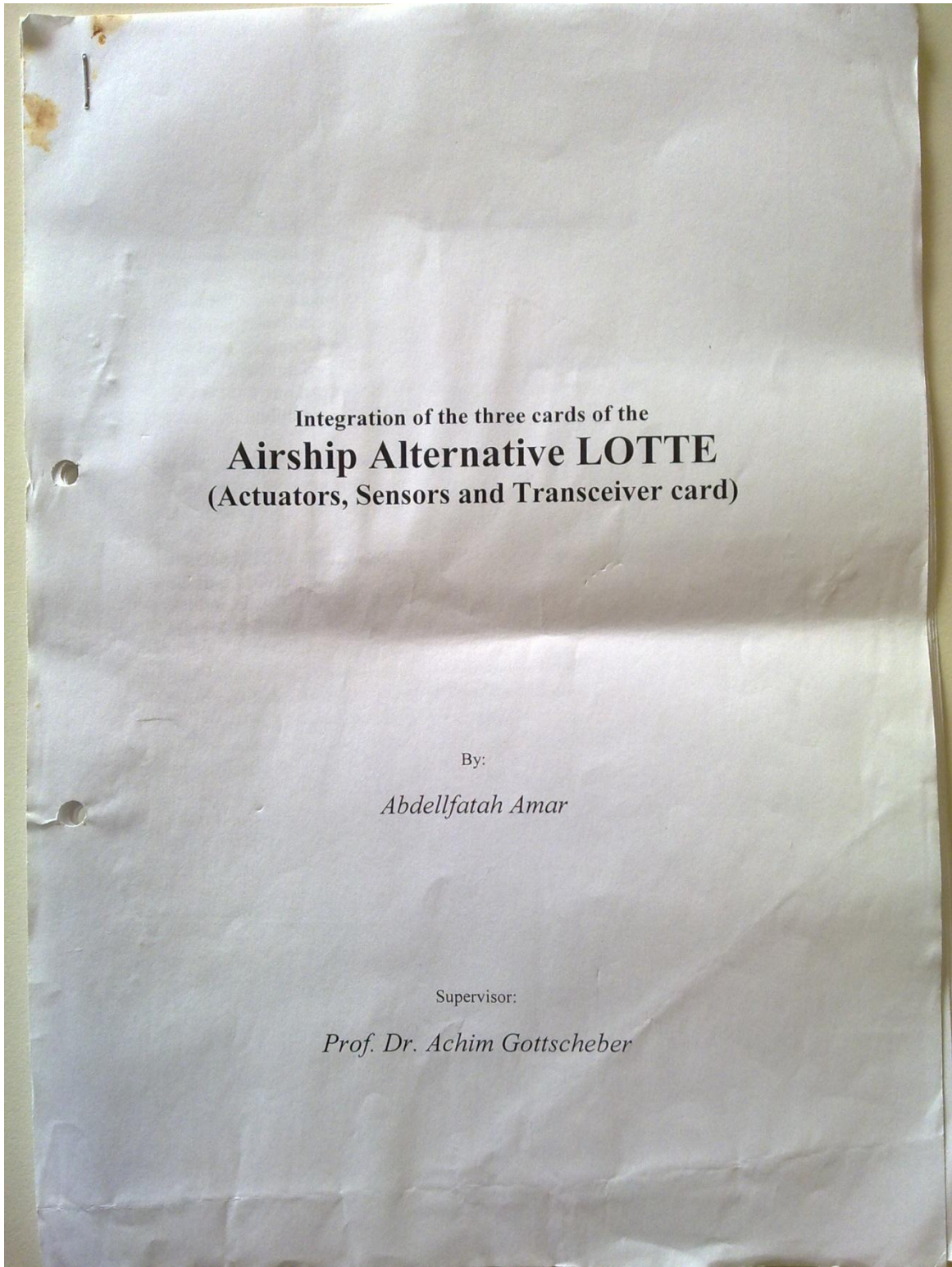
- [1] <http://www.sfb501.uni-kl.de/a1doc/glossary/v-model.html>
- [2] <http://www.stsc.hill.af.mil/crosstalk/2000/06/hirschberg.html>
- [3] <http://www.agilealliance.org/articles/articles/onAnalysis>
- [4] http://salmosa.kaist.ac.kr/~course/DrBae/cs550_2002/project.html
- [5] <http://www.arcom.com>
- [6]: <http://www.easysw.com/~mike/serial/serial.html>
- [9]: <http://en.tldp.org/HOWTO/IO-Port-Programming.html> or
[Http://ibiblio.org/pub/linux/docs/howto/io-Port-Programming](http://ibiblio.org/pub/linux/docs/howto/io-Port-Programming)
- [10]: <http://www.masoner.net/articles/async.html>
- [11]: <http://www.masoner.net/articles/async.html>
- [12]: <http://eavr.u-strassbourg.fr/library/teaching/rt/siframe.htm>
- [15]: <http://www.linuxdevices.com/articles/AT2760742655.html>
, [Http://www.linuxdevices.com/articles/at9952405558.html](http://www.linuxdevices.com/articles/at9952405558.html)
And <http://www.linuxdevices.com/articles/AT8073314981.html>
- [16]: <ftp://ftp.kernel.org/pub/linux/kernel/v2/linux-2.4.4.tar.gz> ;
<Ftp://ftp.rtlinux.com/pub/rtlinux/v3/rtlinux-3.1.tar.gz>
- [8]: Richard, Stevens (1992): Advanced Programming in the UNIX environment, USA: Addison-Wesley.

[7]: Graefe zu, Martin (2005): C and Linux) Munich: Hansen Verlag.

[14]: Embedded Linux: Manual for developers v. Robert Schwebel, 1 Edition :
with ap-Verlag, Bonn 2001.

[13]: Herold, Helmut (2004) :Linux Unix system programming, Munich:
Addison-Wesley. 7

13 Integration code in C



integration.c

```
#include <stdio.h> // für die Standardfunktionen
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <string.h> // für die Fkt strlen, strerror und strncpy
#include <errno.h> // wegen der Konstante errno
#include <stdlib.h>

static int g_fdSensor = 0;
static int g_fdActor = 0;
static int g_fdGUI = 0;

#define MODE_RECEIVE 1
#define MODE_SEND 2
#define MODE_SEND_RECEIVE 3
#define INFINIT 0

-----//
int OpenAdrPort(char* sPortNumber, int nMode;(
int WriteAdrPort(int fd, char* psOutput;(
int ReadAdrPort(int fd, char* psResponse, int iMax;(
void CloseAdrPort(int fd;(
-----//
void interpretSensorData(char* sSensorData, int nDataLen;(
void interpretGUIData(char* sGUIData, int nDataLen;(
-----//

int main(int argc, char *argv([]
}
    int ret = 0;
    fd_set fdSet;

    char* sPortSensor = "0;"
    char* sPortActor = "1;"
    char* sPortGUI = "2;"
```

```

if (argc == 4(
}
    sPortSensor = argv[1];[
    sPortActor = argv[2];[
    sPortGUI = argv[3];[
{
-----//
g_fdSensor = OpenAdrPort(sPortSensor, MODE_RECEIVE;(
g_fdActor = OpenAdrPort(sPortActor , MODE_SEND;(
g_fdGUI = OpenAdrPort(sPortGUI , MODE_SEND_RECEIVE;(
-----//

    FD_ZERO(&fdSet;(
    FD_SET (g_fdSensor, &fdSet;(
    FD_SET (g_fdGUI , &fdSet;(

    while(1(
    }
        struct timeval tv;
        tv.tv_sec = 5;
        tv.tv_usec = 0;
        ret = select(2, &fdSet, 0, 0, &tv;(

        if(ret == -1(
        }
            printf("***** select() failed\ .!n;("
        {
        else
        }

        char sData[1024];[
        int nLenData = 0;

        if(FD_ISSET(g_fdSensor, &fdSet((
        }
            printf("Sensor data are available. We read them\ .n
;("
            nLenData = ReadAdrPort(g_fdSensor, sData, 1024
;((
            if(nLenData > 0(

```

```

        }
        interpretSensorData(sData, nLenData);
    }
    {
    {
    else if(FD_ISSET(g_fdGUI, &fdSet((
    }
        nLenData = ReadAdrPort(g_fdGUI, sData, 1024);
        if(nLenData > 0(
        }
            interpretGUIData(sData, nLenData);
            {
            printf("GUI data are available. We read them\n;("
            {
            else
            }
            WriteAdrPort(g_fdGUI, "yassir & fathallah;("
            printf("Should never occure\!!!!n;("
            {
            {
            {
            -----//
            return ret;
            {
            -----//
            //interpretSensorData : Diese Funktion macht das und das
            void interpretSensorData(char* sSensorData, int nDataLen(
            }
            {
            -----//
            //interpretSensorData : Diese Funktion macht das und das
            void interpretGUIData(char* sGUIData, int nDataLen(
            }
            {
            -----//
            //interpretSensorData : Diese Funktion macht das und das
            int OpenAdrPort(char* sPortNumber, int nMode(
            }
            char sPortName[64;[
            int fd;

```

```

printf("in OpenAdrPort port#=%s\n", sPortNumber);(
sprintf(sPortName, "/dev/ttyS%s", sPortNumber);(
printf("sPortName=%s\n", sPortName);(

if(nMode == MODE_RECEIVE(
}
    //Device ffnen mit der Fct : int open(char *pathname, int flags)
    fd = open(sPortName, O_RDONLY | O_NOCTTY | O_NDELAY);
    //fd = open(sPortName, O_RDONLY);
}
else if(nMode == MODE_SEND)
{
    //fd = open(sPortName, O_WRONLY );
    fd = open(sPortName, O_WRONLY | O_NOCTTY | O_NDELAY);
}
else
{
    //fd = open(sPortName, O_RDWR);
    fd = open(sPortName, O_RDWR | O_NOCTTY | O_NDELAY);
}

if (fd < 0)
{
    printf("open error %d %s\n", errno, strerror(errno));
}
else
{
    struct termios my_termios;
    printf("fd is %d\n", fd);
    tcgetattr(fd, &my_termios);
    // NOTE: you may want to save the port attributes here so that you can
restore them later
    printf("old cflag=%08x\n", my_termios.c_cflag);
    printf("old oflag=%08x\n", my_termios.c_oflag);
    printf("old iflag=%08x\n", my_termios.c_iflag);
    printf("old lflag=%08x\n", my_termios.c_lflag);
    printf("old line=%02x\n", my_termios.c_line);

    tcflush(fd, TCIFLUSH);
}

```

```

my_termios.c_cflag = B9600 | CS8 | CREAD | CLOCAL | HUPCL;
cfsetospeed(&my_termios, B9600);
tcsetattr(fd, TCSANOW, &my_termios);

printf("new cflag=%08x\n", my_termios.c_cflag);
printf("new oflag=%08x\n", my_termios.c_oflag);
printf("new iflag=%08x\n", my_termios.c_iflag);
printf("new lflag=%08x\n", my_termios.c_lflag);
printf("new line=%02x\n", my_termios.c_line);
}
return fd;
}
//-----
// interpretSensorData : Diese Funktion macht das und das
int WriteAdrPort(int fd, char* psOutput)
{
    int iOut;

    if (fd < 1)
    {
        printf(" port is not open\n");
        return -1;
    }
    /*    Gepufferte (canonical ?) Ein/Ausgabe und ungepuffertes
(noncanonical)
        Lesen/Schreiben : int write (int fd, void *buffer, int)
        */
    iOut = write(fd, psOutput, strlen(psOutput) + 1);
    if (iOut < 0)
    {
        printf("write error %d %s\n", errno, strerror(errno));
    }
    else
    {
        printf("wrote %d chars: %s\n", iOut, psOutput);
    }
    return iOut;
}
//-----

```

```

// interpretSensorData : Diese Funktion macht das und das
int ReadAdrPort(int fd, char* psResponse, int iMax)
{
    int nRead = -1;

    if (fd < 1)
    {
        printf(" port is not open\n");
    }
    else
    {
        nRead = read(fd, psResponse, iMax-1);
        if (nRead < 0)
        {
            printf("ReadAdrPort(%d) failed! nRead(%d) errno(%d) \n", fd,
nRead, errno);
        }
        else
        {
            int zeroPos = (nRead < iMax) ? nRead : iMax;
            psResponse[zeroPos] = '\0';
        }
    }
    return nRead;
}
//-----
// interpretSensorData : Diese Funktion macht das und das
void CloseAdrPort(int fd)
{
    if (fd > 0)
    {
        close(fd);
    }
}
//-----
// interpretSensorData : Diese Funktion macht das und das

```


Development of a communication system (2013)

IAP_ECS

**Emergency Communication System to be
implemented into IAP Satellite System**

IAP_ECS Demonstration Platform

Project duration: May 2013 – Jan 2014

Authors:

Mahmoud Zohby

Samir Mourad

All rights reserved © AECENAR

January 2014

Content

ABBREVIATIONS	501
1 ABSTRACT	503
2 PROJECT MANAGEMENT	505
2.1 PROJECT DEFINITION HISTORY	505
2.2 SYSTEM BUDGET (TIME AND COST) FOR DEMO SYSTEM	505
2.3 AT 21 JAN STILL OPEN TASKS FOR IAP ECS DEMO SYSTEM WHEN USING (ONLY INTEGRATION)	505
3 BASICS.....	507
3.1 COMMUNICATION BASICS.....	507
3.1.1 <i>Transmitter design from http://en.wikibooks.org/wiki/Electronics/Transmitter_design.....</i>	<i>519</i>
3.1.1.1 Frequency synthesis and frequency multiplication	520
3.1.1.2 Frequency mixing and Modulation	521
3.1.1.3 RF power amplifiers.....	527
3.1.1.4 Linking the transmitter to the aerial.....	527
3.1.1.5 EMC matters.....	528
3.2 RECEIVER DESIGN FROM HTTP://EN.WIKIPEDIA.ORG/WIKI/TUNER_(ELECTRONICS)	533
3.3 ANTENNA	534
3.4 SOFTWARE DEFINED RADIO (SDR)	536
3.5 HDSDR (HIGH DEFINITION SOFTWARE DEFINED RADIO).....	536
3.6 EXTIO.DLL	537
3.7 HOW DO I DEVELOP AN EXTIO.DLL ?	537
3.8 VISUAL C++ 2008 EXPRESS	537
3.9 QT	537
3.10 RF HARDWARE (USB STICK).....	537
3.10.1 <i>TERRATEC ran T stick DVB-T/DAB/DAB + Stick USB 2.0</i>	<i>537</i>
3.10.2 <i>Hackrf (an-open-source-SDR-platform)</i>	<i>538</i>
3.11 RF OVERVIEW	538
3.12 RF FREQUENCIES POLICIES	539
3.13 RF MODULES.....	543
3.13.1 <i>STD-402</i>	<i>543</i>
3.13.1.1 Special for <i>MB-STD-RS232</i>	543
3.13.1.2 Special for <i>STD-402 (Transceiver)</i>	545
3.13.2 <i>RFM42B-RFM31B 433MHz</i>	547
3.13.3 <i>BOWITZ W.T.</i>	549
3.13.4 <i>Comparison between modules</i>	549
4 SPECIFICATION	551
4.1 SYSTEM REQUIREMENTS.....	551
4.2 HARDWARE REQUIREMENTS.....	551
4.3 SOFTWARE REQUIREMENTS.....	551
5 SYSTEM DESIGN	553
5.1 SYSTEM OVERVIEW.....	553
5.2 CENTRAL STATION	553
5.2.1 <i>Architecture</i>	<i>553</i>
5.2.2 <i>SDR development side</i>	<i>553</i>
5.2.3 <i>Graphical User Interface</i>	<i>554</i>

5.3	MOBILE STATIONS.....	555
6	MECHANICS.....	557
6.1	MECHANICAL DESIGN	557
6.2	PROTOTYPE WITHOUT COVER.....	557
7	SCS-SMS	559
7.1	ABSTRACT OF SCS-SMS.....	559
7.2	SYSTEM DESIGN	559
7.3	ARCHITECTURES	560
7.4	PIC SOFTWARE	566
7.5	TEST	579
8	AES ENCRYPTION.....	583
8.1	INTRODUCTION	583
8.2	AES ALGORITHM	583
8.3	CODING	586
9	HARDWARE OF ECS DEMO SYSTEM	599
9.1	REALIZATION OF RF MODULE	599
9.1.1	<i>Using STD-402.....</i>	<i>599</i>
9.1.2	<i>Realization of RF Module Using RFM42B-RFM31B – 433MHz.....</i>	<i>603</i>
9.1.2.1	Serial Periferal interface (SPI)	603
9.1.2.2	The new hardware design.....	603
9.1.2.3	MSSP module to establishing (SPI)	605
10	FURTHER WORK: SYSTEM INTEGRATION AND INTEGRATION TEST OF ECS DEMO SYSTEM	611
	APPENDIX A: ALTERNATIVE PROJECT PLANS.....	612
	APPENDIX B: ALL ABOUT HACKRF	616
	<i>B.1 HackRF overview.....</i>	<i>616</i>
	<i>B.2 Jawbreaker.....</i>	<i>618</i>
	<i>B.3 Jellybean</i>	<i>618</i>
	<i>B.4 Lemondrop.....</i>	<i>619</i>
	APPENDIX C: ALTERNATIVE SYSTEM DESIGNS.....	624
	LITERATURE	625

Abbreviations

ECS

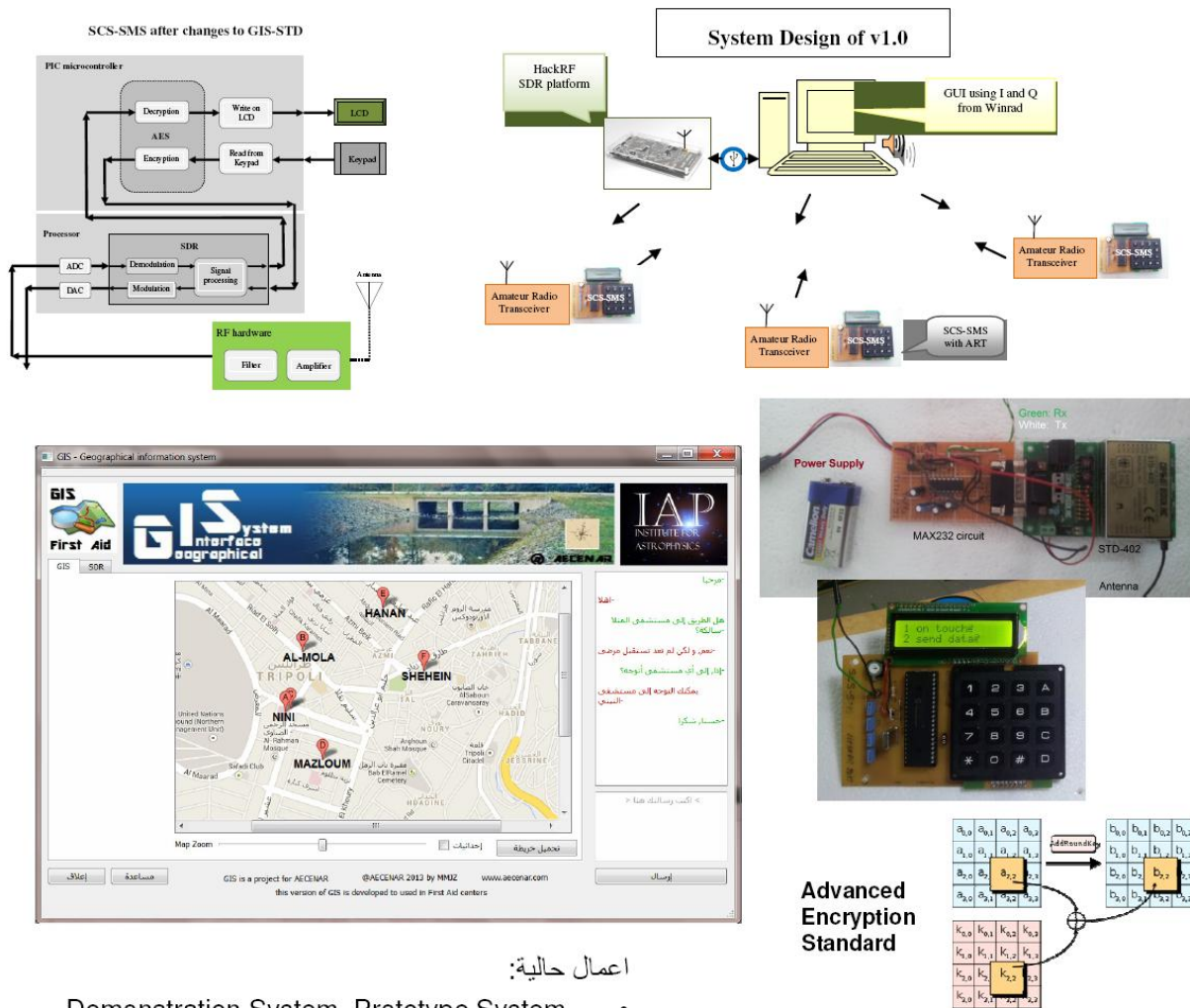
Emergency Communication System

14 Abstract



Emergency Communication System شبكة اتصالات آمنة للطوارئ و الاسعافات

Last update: 03 December 2013



اعمال حالية:

Demonstration System, Prototype System •

الحاجيات لعام 2014:

- 2 اشخاص
- \$5.000 للمواد

15 Project Management

15.1 Project Definition History

First there were developed SMS-SCS and AES. (June – August 2013). Later on there was an investigation about the possibility of using SDR (Software Designed Radio). (September, October). Later on it was decided to make a demonstration system for an Emergency Communication System. (October). IAP ECS was developed. SMS-SCS and AES were integrated into this system. (October 2013 – January 2014).

15.2 System budget (time and cost) for Demo system

Part	Task	Time (week)	Cost (USD)
Project Plan	1 engineer (Project manager)	3 weeks	E.C.
Client Side	SCS-SMS project hardware (components for 2 items)	2 week	30\$ x 2
	PIC program development (1 engineer)	3 weeks	E.C.
	RF transceiver for SCS-SMS (components for 3 sides)	2 weeks	12\$ x 6
Base Station Side	Know how of HSDR	1/2 week	E.C.
	Know how of WinRad	1/2 week	E.C.
	Know how of HackRF	1 week	E.C.
	VC++ & Qt software tutorials	2 weeks	E.C.
	Qt GUI interface	2 weeks	E.C.
	SDR platform (2 USB stick)	1 week	40\$ x 2
Overall System	Documentation and report	1 week	E.C.
	Testing system and solving problem	1 week	E.C.
Total:		19 weeks	212 \$
One engineer cost 200\$ each week. So, for 19 weeks he costs:		200\$ x 19= 3800\$	
Summation with the hardware cost:		4012\$	
4000\$ in approx. 5 months			

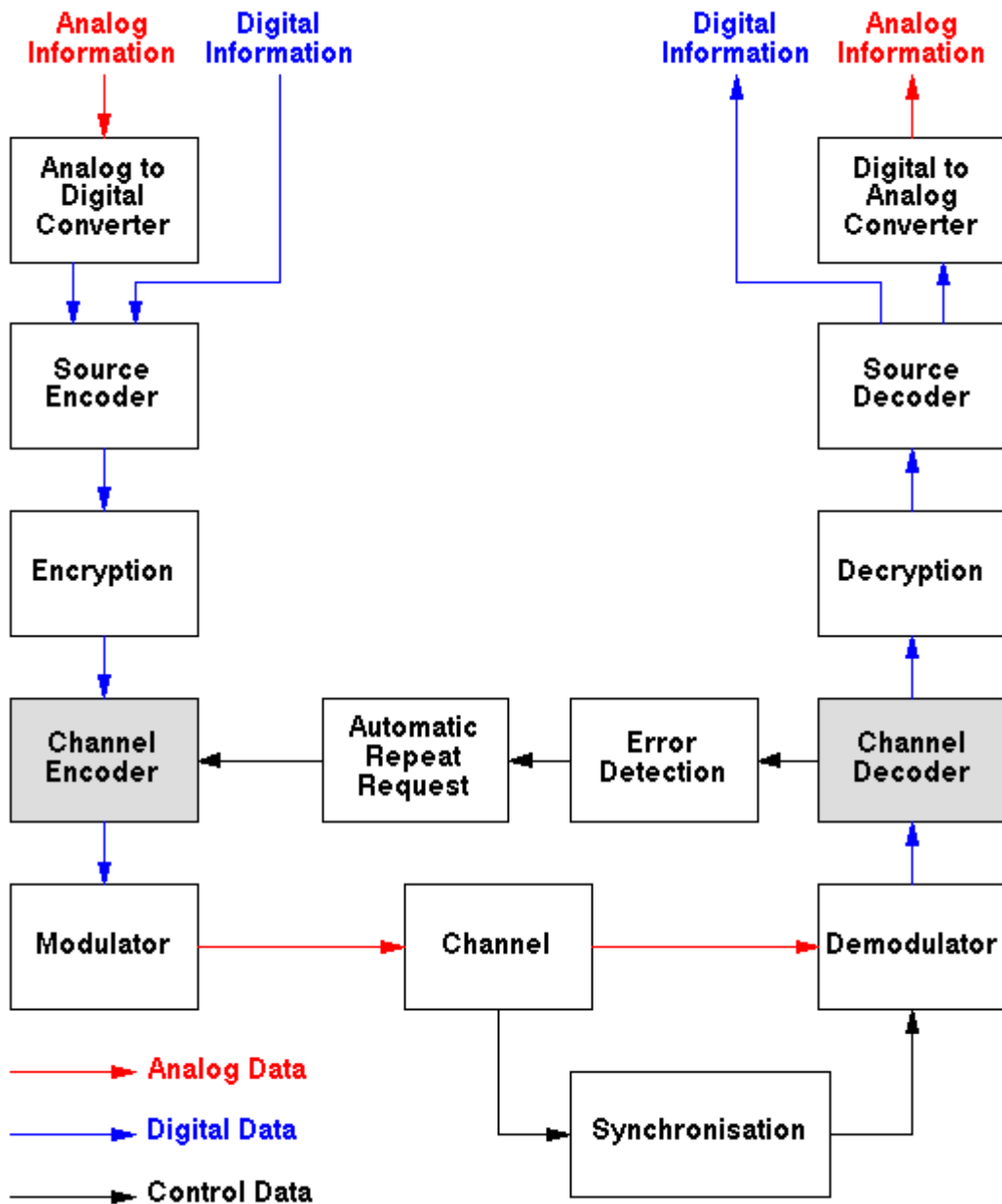
15.3 At 21 Jan still open tasks for IAP ECS Demo System when using (Only Integration)

Event	Time
Using WinRad to receive Radio wave using ran T-stick+	1 week
Complete the SCS-SMS project 1 of secured communication system	3 weeks
SW for connection SCS-SMS hardware to the RF module, testing	1 week
Take I and Q from WinRad to a file	1 week
Adapting GUI interface to read SMS from file	2 weeks
System testing	1 week

Planned time: approximately **6 weeks**

16 Basics

16.1 Communication Basics⁶



Die Aufgabe der Nachrichtentechnik besteht darin, Informationen von einem Sender zu einem Empfänger zu befördern. Die Nachrichtentechnik kann grob in zwei große Gebiete geteilt werden, in die

- Übertragungstechnik und in die
- Vermittlungstechnik.

⁶ Many is taken from: Prof. Dr.-Ing. Gerhard P. Fettweis, Technische Universität Dresden, Fakultät Elektrotechnik, Skript zur Vorlesung Einführung in die Nachrichtentechnik, Sommersemester 2012

Basics

Beispiele für nachrichtentechnische Anwendungen sind:

- Hörrundfunk und Fernsehen
 - analog: AM-Radio (Mittelwelle), FM-Radio (UKW),
 - digital: DAB (digital audio broadcasting), DVB (digital video broadcasting),
- Telefon
 - Festnetz,
 - Mobilfunk.

Nachrichtenübertragungssysteme

Man kann Nachrichtenübertragungssysteme durch das in Abb. 2.1 dargestellte Modell beschreiben.

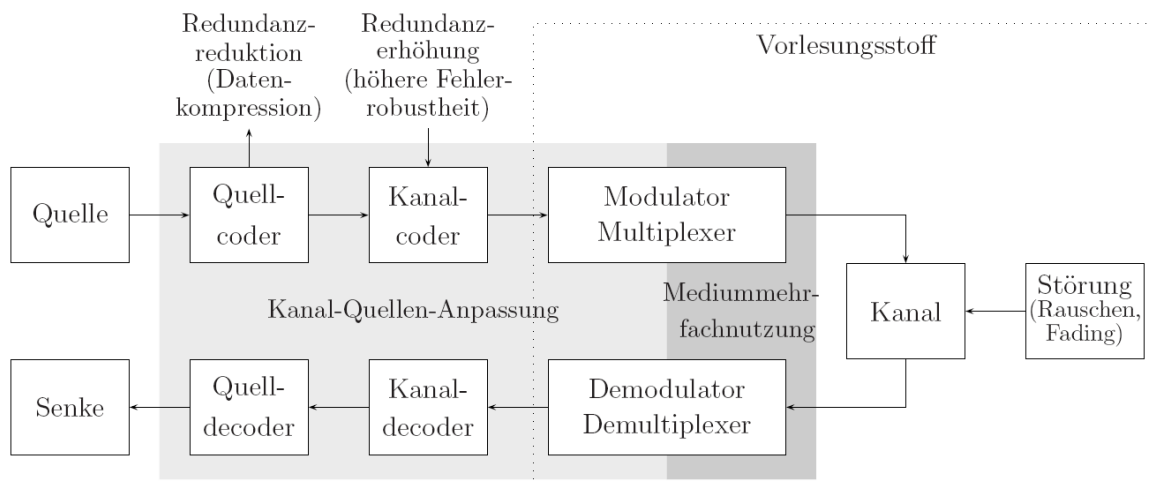
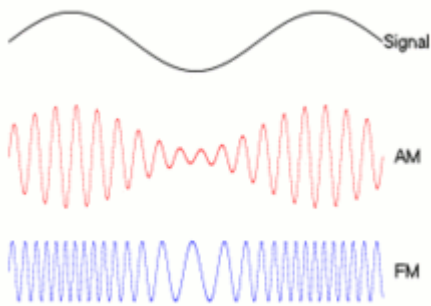


Abbildung 2.1: Allgemeines nachrichtentechnisches Übertragungssystem

Begriffe

Die Bausteine Quelle, Quellcoder, Kanalcoder, Modulator und Multiplexer werden unter dem Begriff Sender zusammengefasst. Dementsprechend gehören zu dem Empfänger die Baugruppen Demultiplexer, Demodulator, Fehlerkorrektur Elemente, Quelldecoder und eine Senke. Sender und Empfänger können sowohl stationär (z.B. Fernsehsender) als auch mobil (z.B. Handy) sein, sind aber immer leistungsbegrenzt. Der Kanal als Übertragungsmedium ist bandbreitenbegrenzt. Durch Störgeräusche, Amplitudenschwankungen (fading, verursacht durch Bewegung und Abschattung), Interferenzerscheinungen, Zeit- (delay spread, verursacht durch Mehrwegeausbreitung) und Frequenzdispersion (Doppler spread, verursacht durch Bewegung von Sender, Empfänger und/oder Streuern/Reflektoren usw.) werden die gesendeten Informationen beeinflusst.

Modulator



analoge Amplitudenmodulation (AM) und Frequenzmodulation (FM) eines niederfrequenten Signals

Übertragungsmedien

Die Wahl des Übertragungsmediums hängt sehr stark von den Anforderungen an den Übertragungskanal ab (z.B. bezüglich Frequenzbereich oder Signalbandbreite, aber auch hinsichtlich der gleichzeitigen Anzahl der Nutzer). Mögliche Übertragungsmedien sind

- "Twisted Pair" (verdrillte Kupferkabel),
 - z.B. Telefonkabel (Endgeräteanschluß)
- Koaxkabel,
 - z.B. Antennenkabel, Kabelfernsehen
- Hohlleiter,
 - z.B. Antenneneinspeisung bei hohen Frequenzen (Giga-Hertz-Bereich)
- Lichtwellenleiter,
 - z.B. Übertragung mit sehr hohen Datenraten
- Funkkanal.
 - z.B. Mobilfunk, Hörrund- und Fernsehfunk

Im Funkbereich unterscheidet man auch zwischen Indoor- und Outdoor-Anwendungen. Ein typisches Beispiel für Indoor-Anwendungen könnte die Versorgung aller Räume eines Bürogebäudes mit einem WLAN (wireless local area network) sein. Outdoor-Anwendungen sind z.B. die bundesweit verbreiteten zellularen Mobilfunknetze, derzeit GSM-900, DCS-1800 (D1-, D2-, Eplus- und E2-Netz) und zukünftig auch UMTS.

Auch die Frequenz- bzw. Wellenlängenbereiche werden unterschieden, angefangen von den bekannten MW- und UKW-Bereichen bis hin zu den Millimeterwellen-Bereichen und weiter Infrarot-Bereichen der optischen Nachrichtentechnik.

Eigenschaften

Basics

Im folgenden werden einige Eigenschaften von Nachrichtenübertragungssystemen aufgezählt. Dabei wird keinerlei Anspruch auf Vollständigkeit erhoben.

Simplex/Duplex Ein Unterscheidungskriterium ist, ob Systeme im Simplex- oder Duplexmodus betrieben werden. Simplexbetrieb bedeutet, daß Nachrichten nur in eine Richtung übertragen werden (z.B. Rundfunk), während im Duplexbetrieb die Informationen in beide Richtungen übertragen werden (z.B. Telefonie).

Single-Cast/Multi-Cast Es gibt Single-Cast-Systeme (Telefon: 1 Quelle, 1 Empfänger) und Multi-Cast-Systeme (Rundfunk: 1 Quelle mit vielen Empfängern)

Paket/Leitungs-Vermittlung Ein weiteres Merkmal ist, ob Nachrichten leitungsvermittelt (z.B. "das gute alte" Telefon) oder paketvermittelt (z.B. Datenübertragung im Internet – IP-Protokoll) übertragen werden.

Signalpegel

Oftmals sind Signale mit großen Pegelunterschieden gegeben. Typische Werte für Signalleistungen P liegen zwischen $1 \mu\text{W}$ und 1kW . Das entspricht einem Unterschied von 10^9 . Aus diesem Grund ist eine logarithmische Skala vorteilhaft. Eine solche Skala ist die dBm-Skala, bei der die Leistungspegel L_P auf $P_{\text{ref}} = 1 \text{ mW}$ normiert werden, also

$$P = \text{Leistung}(s(t)) \quad (2.1)$$

$$L_P = 10 \log_{10} \frac{P}{P_{\text{ref}}} \text{ dBm}$$

$$L_P = 10 \lg \frac{P}{P_{\text{ref}}} \text{ dBm} \quad \text{mit} \quad P_{\text{ref}} = 1 \text{ mW} \quad (2.2)$$

abs. Leistung in [mW]	0.1	0.5	1	2	4	8	10	100	1000
rel. Leistung in [dBm]	-10	-3	0	3	6	9	10	20	30

Tabelle 2.1: Absolutleistung vs. dBm-Pegel

· In der Tab. 2.1 sind einige absolute Leistungswerte und die dazugehörigen dBm-Werte angegeben.

· 2 W-Handy (D-Netz): $P_{\text{max}} = 2 \text{ W}$, äquivalente Darstellung als Pegel $L_{P_{\text{max}}} = 10 \lg \frac{2\text{W}}{1\text{mW}} \text{ dBm} = 10 \lg(2 \cdot 10^3) \text{ dBm} = (10 \lg(2) + 10 \lg(10^3)) \text{ dBm} = 33 \text{ dBm}$. Im GSM-Standard ist spezifiziert, daß der Pegel an der Basisstation mindestens -102 dBm betragen muß, d.h. es können sich Pegeldifferenzen von bis zu 135 dBm bzw. 10^{13} ergeben.

· 0.8 W-Handy (E-Netz): $\equiv 29 \text{ dBm}$. Da mit einer geringeren Leistung gesendet wird und bei 1.8 GHz wesentlich stärkere Dämpfungsverhältnisse vorliegen, ist im E-Netz eine größere Anzahl von

Basisstationen gegenüber dem D-Netz erforderlich, was sich in den Infrastrukturkosten niederschlägt.

· Auch Signalspannungen können im logarithmischen Maßstab angegeben werden. Als Bezugsspannung wird meist 0.775 V verwendet und entspricht 0 dBu. (Wahl der Referenzspannung: Welche Spannung ist notwendig, um an einem 600 Normwiderstand eine Leistung von 1 mW entstehen zu lassen? = $0.775^2 V^2 / 600 = 1mW$) Die Wahl eines anderen Normwiderstandes bzw. Referenzspannung verschiebt die dB-Skala entsprechend.

Ebenso lassen sich Verstärkungsfaktoren von Systemen äquivalent als Pegel angeben. Bezeichnen z.B. x und y den Ein- bzw. Ausgang eines Systems, so ergeben sich die Pegel zu

$$g_P = \frac{\text{Leistung}(y)}{\text{Leistung}(x)} \quad \text{bzw.} \quad g_A = \frac{\text{Amplitude}(y)}{\text{Amplitude}(x)} \quad (2.3)$$

$$\begin{aligned} & - \text{Leistungsverstärkung (gain)} & & - \text{Amplitudenverstärkung} \\ L_{g_P} = 10 \lg(g_P) \text{ dB} & & & L_{g_A} = 20 \lg(g_A) \text{ dB} \end{aligned} \quad (2.4)$$

Besonders bei passiven Systemen werden oft Dämpfungs- statt Verstärkungsfaktoren angegeben.

$$a_P = \frac{\text{Leistung}(x)}{\text{Leistung}(y)} \quad \text{bzw.} \quad a_A = \frac{\text{Amplitude}(x)}{\text{Amplitude}(y)} \quad (2.5)$$

$$\begin{aligned} a_P = \frac{1}{g_P} & & & a_A = \frac{1}{g_A} \\ & & & - \text{Amplitudendämpfung} \\ & & & - \text{Leistungs-dämpfung (attenuation)} \end{aligned} \quad (2.6)$$

$$L_{a_P} = 10 \lg(a_P) \text{ dB} \quad L_{a_A} = 20 \lg(a_A) \text{ dB} \quad (2.7)$$

$$L_{a_P} = -L_{g_P} \quad L_{a_A} = -L_{g_A} \quad (2.8)$$

Oftmals sind die Pegelverhältnisse zwischen Nutz- und Störsignalen von Interesse. Das Verhältnis

$$SNR = 10 \lg \left(\frac{\text{Nutzsignalleistung}}{\text{Störsignalleistung}} \right) \text{ dB} \quad (2.9)$$

gibt das Nutzsignal-/Störsignalleistungsverhältnis (signal-to-noise-ratio) an. Beachten Sie bitte,

- daß Pegelangaben in dB immer Verhältnisse zweier Leistungen oder Amplituden (z.B. Verstärkungsfaktor, Signal-Rausch-Abstand) bezeichnen
- daß Pegelangaben in dBm (Referenz: Leistung $P_{\text{ref}} = 1 \text{ mW}$), dBW (Referenz: Leistung $P_{\text{ref}} = 1 \text{ W}$), dBu (Referenz: Spannung $U_{\text{ref}} = 0.775 \text{ V}$) usw. immer absolute Leistungen oder Spannungen bezeichnen
- daß sich beim Rechnen mit Pegeln folgende Einheiten ergeben:

Basics

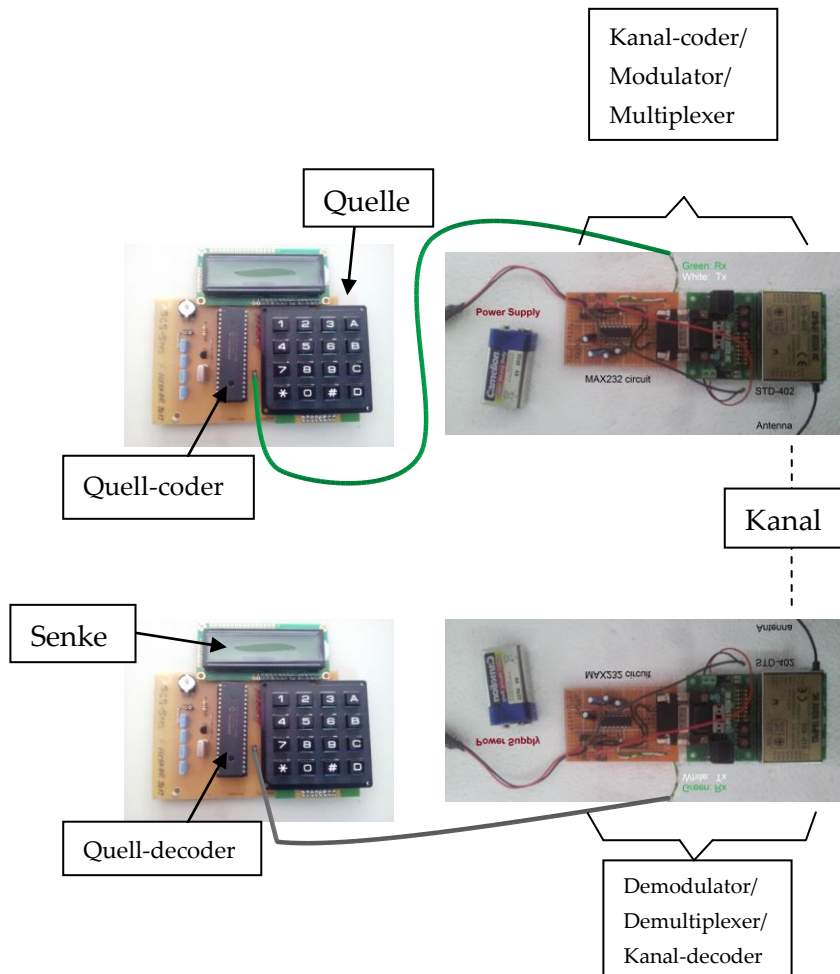
$$\text{dB} \pm \text{dB} = \text{dB} \tag{2.10}$$

$$\text{dBm} \pm \text{dB} = \text{dBm} \tag{2.11}$$

$$\text{dBm} - \text{dBm} = \text{dB} \tag{2.12}$$

$$\text{dBm} + \text{dBm} = \text{nicht definiert} \tag{2.13}$$

$$\tag{2.14}$$



From “Microwave and RF Design: A Systems Approach”, Chapter 1 (Modulation, Transmitters and Receivers)(www.ece.ucsb.edu/yuegroup/Teaching/Lectures/steer_rf_chapter1.pdf):

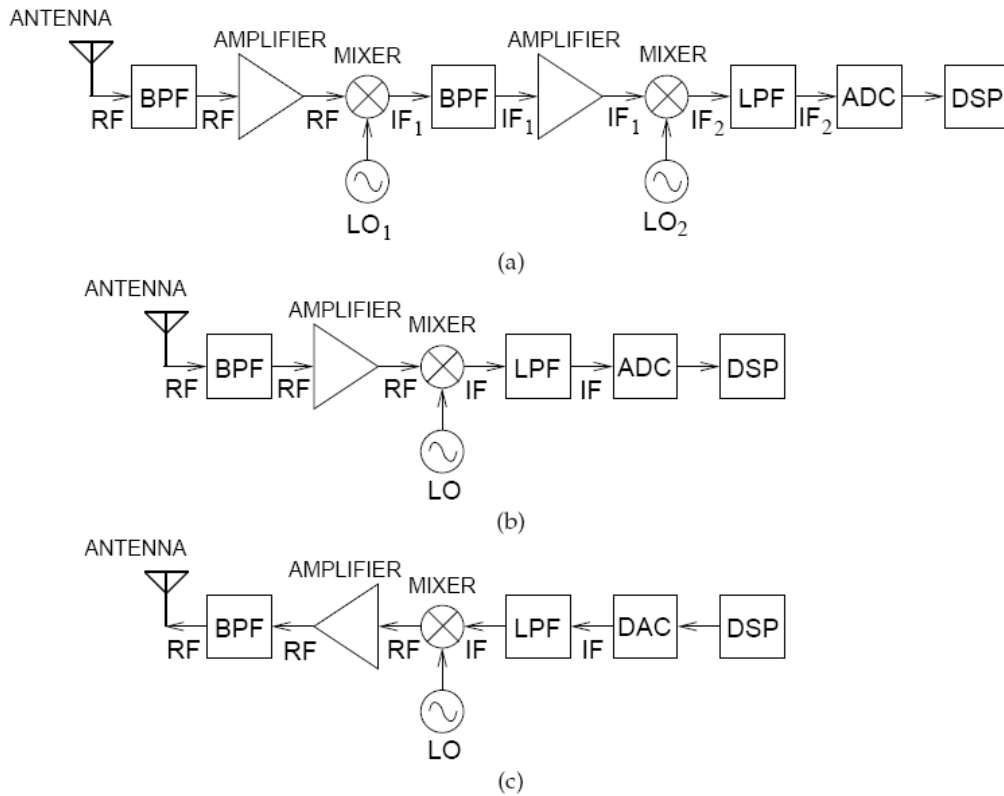


Figure 1-1 Unilateral RF frontend: (a) a receiver with two mixing stages; (b) a receiver with one heterodyne stage; and (c) a one-stage transmitter.

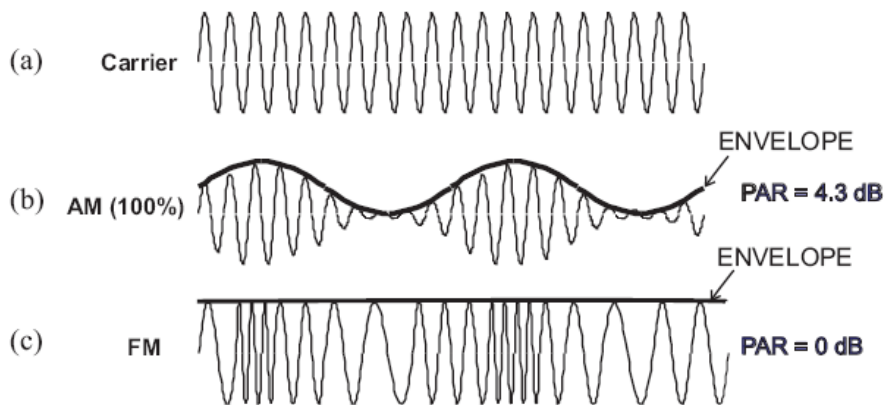


Figure 1-7 Comparison of 100% AM and FM highlighting the envelopes of both: (a) carrier; (b) AM signal with envelope; and (c) FM or PM signal with the envelope being a straight line or constant.

Basics

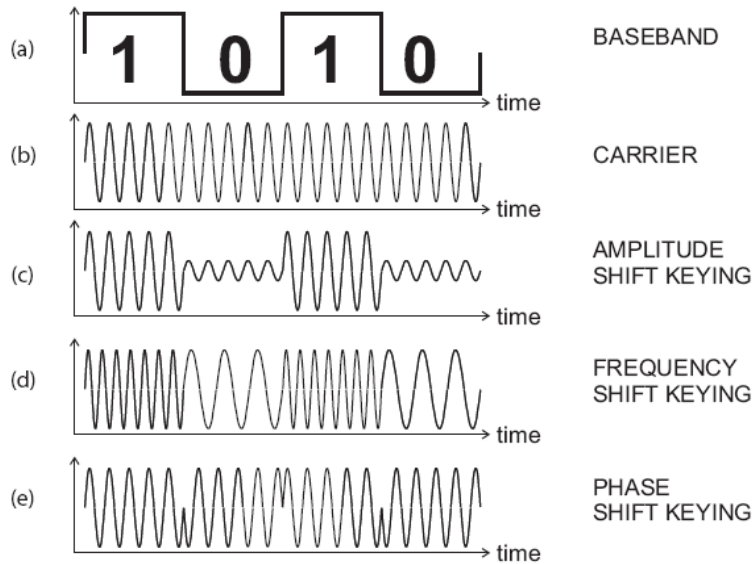


Figure 1-8 Modes of digital modulation: (a) modulating bit stream; (b) carrier; (c) carrier modulated using amplitude shift keying (ASK); (d) carrier modulated using frequency shift keying (FSK); and (e) carrier modulated using binary phase shift keying (BPSK).

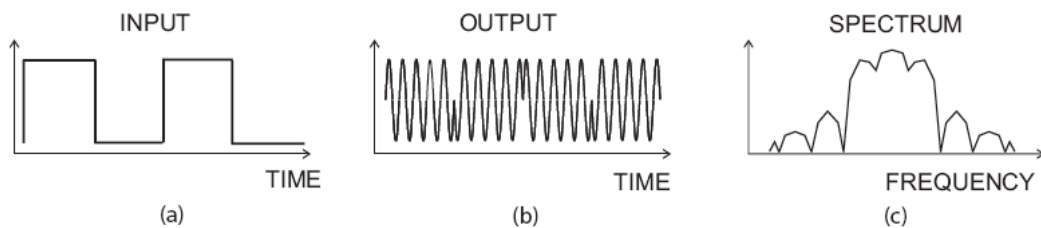


Figure 1-9 Characteristics of phase shift keying (PSK) modulation: (a) modulating bit stream; (b) the waveform of the carrier modulated using PSK with the phase determined by the 1s and 0s of the modulating bit stream; and (c) the spectrum of the modulated signal.

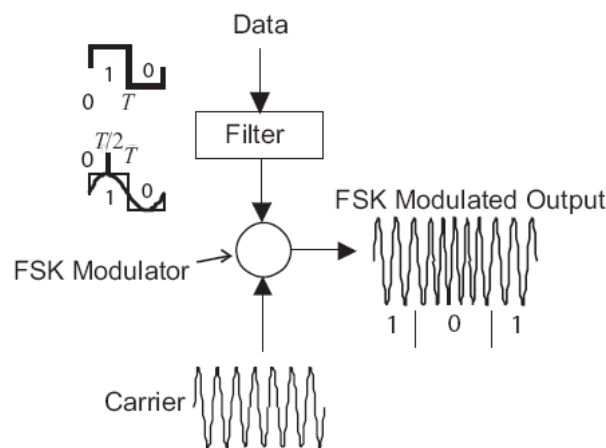


Figure 1-10 The frequency shift keying (FSK) modulation system.

Example 1.2 QPSK Modulation and Constellation

The bit sequence 110101001100 is to be transmitted using QPSK modulation. Show the transitions on a constellation diagram.

SOLUTION: The bit sequence 110101001100 must be converted to a two-bit wide parallel stream of symbols resulting in the sequence of symbols 11 01 01 00 11 00. The symbol 11 transitions to the symbol 01 and then the symbol 01 and so on. The states (or symbols) and the transitions from one symbol to the next required to send the bitstream 110101001100 are shown in Figure 1-13. QPSK modulation results in the phasor of the carrier transitioning through the origin so that the average power is lower and the PAR is high. A more significant problem is that the phasor will fall below the noise floor, making carrier recovery almost impossible.

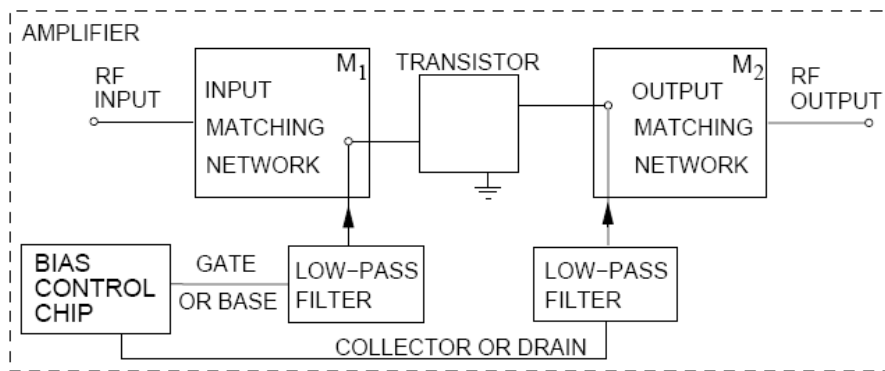


Figure 1-25 Block diagram of an RF amplifier including biasing networks.

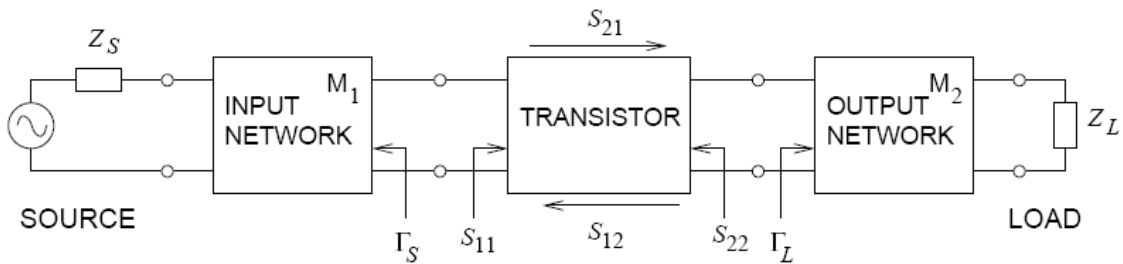


Figure 1-26 The scattering parameter (S_{nm}) and reflection coefficients (Γ) associated with a microwave transistor amplifier.

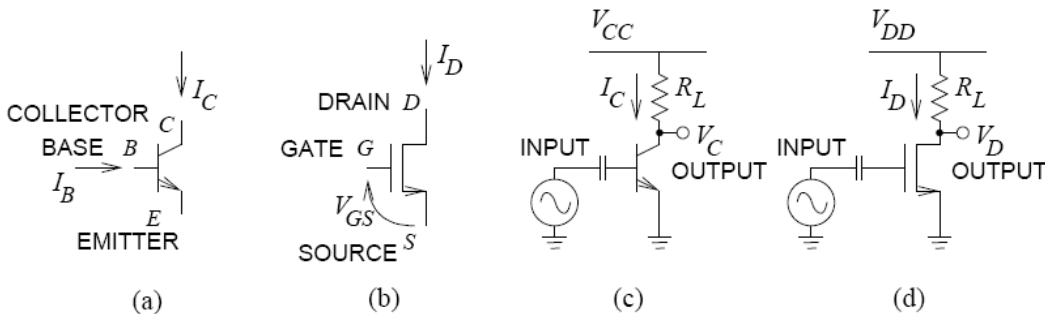


Figure 1-28 Class A single-ended amplifiers: (a) BJT transistor with B for base terminal, C for collector terminal, and E for emitter terminal; (b) MOSFET transistor with G for gate terminal, D for drain terminal, and S for source terminal; (c) single-ended BJT Class A amplifier with resistive bias; and (d) single-ended MOSFET Class A amplifier with resistive bias.

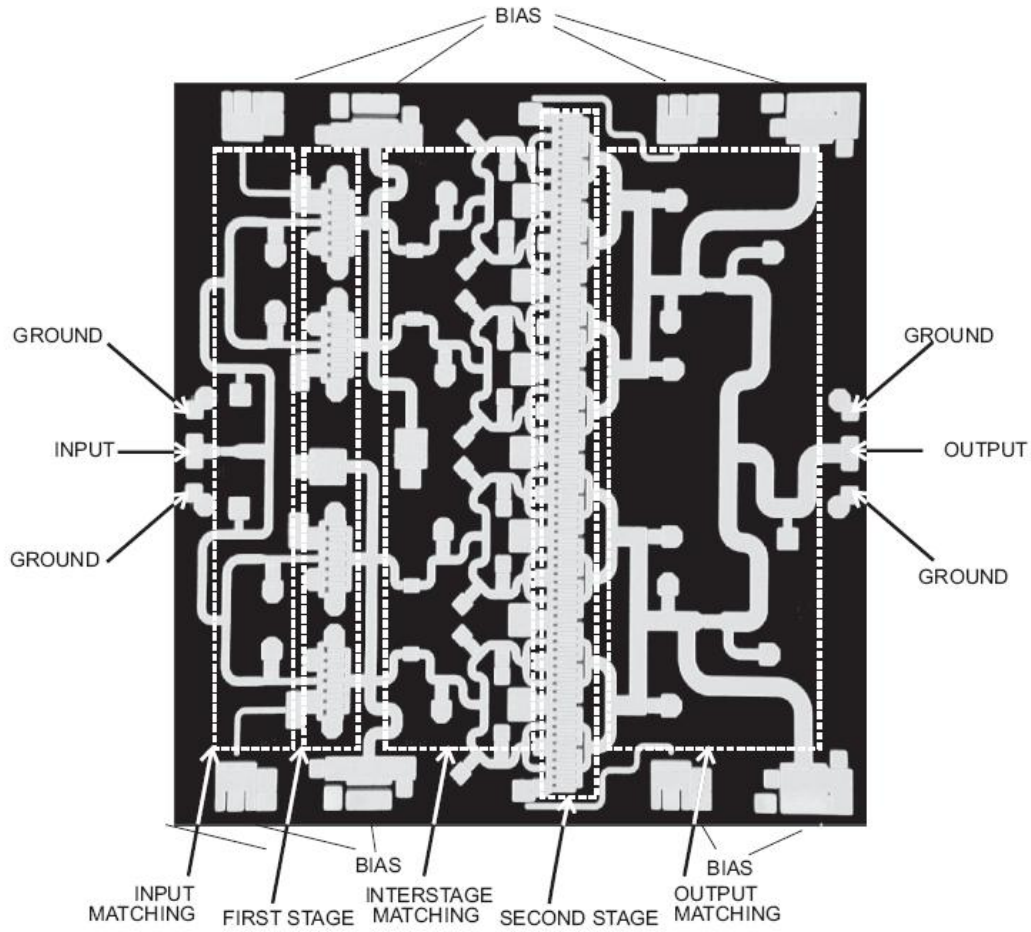


Figure 1-34 An 8–12 GHz MMIC amplifier producing approximately 1 W of output power with key networks identified. (Courtesy Filtronic, PLC, used with permission.)

Modern Transmitter Architectures

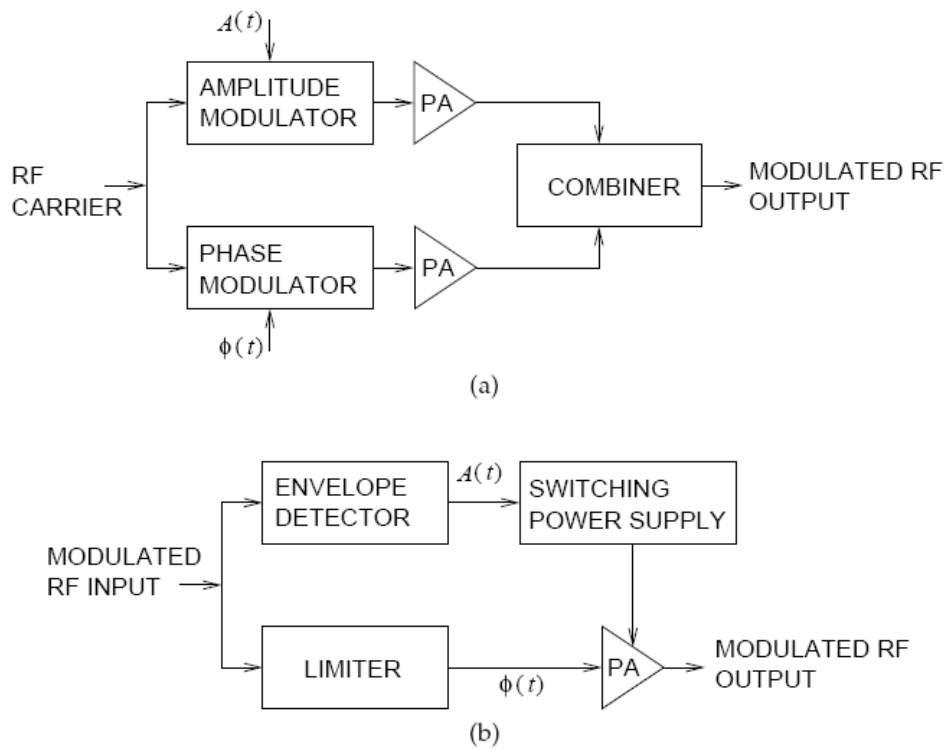


Figure 1-55 Polar modulator architectures: (a) amplitude and phase modulated components amplified separately and combined; and (b) the amplitude used to modulate a power supply driving a saturating amplifier with phase modulated input.

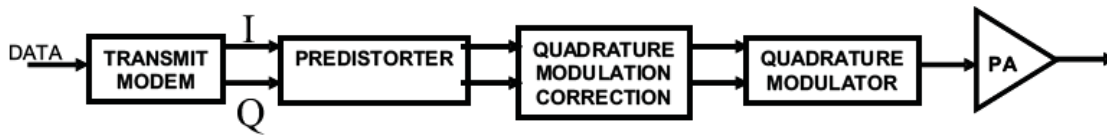


Figure 1-56 Architecture of a direct conversion transmitter.

Modern Receiver Architectures

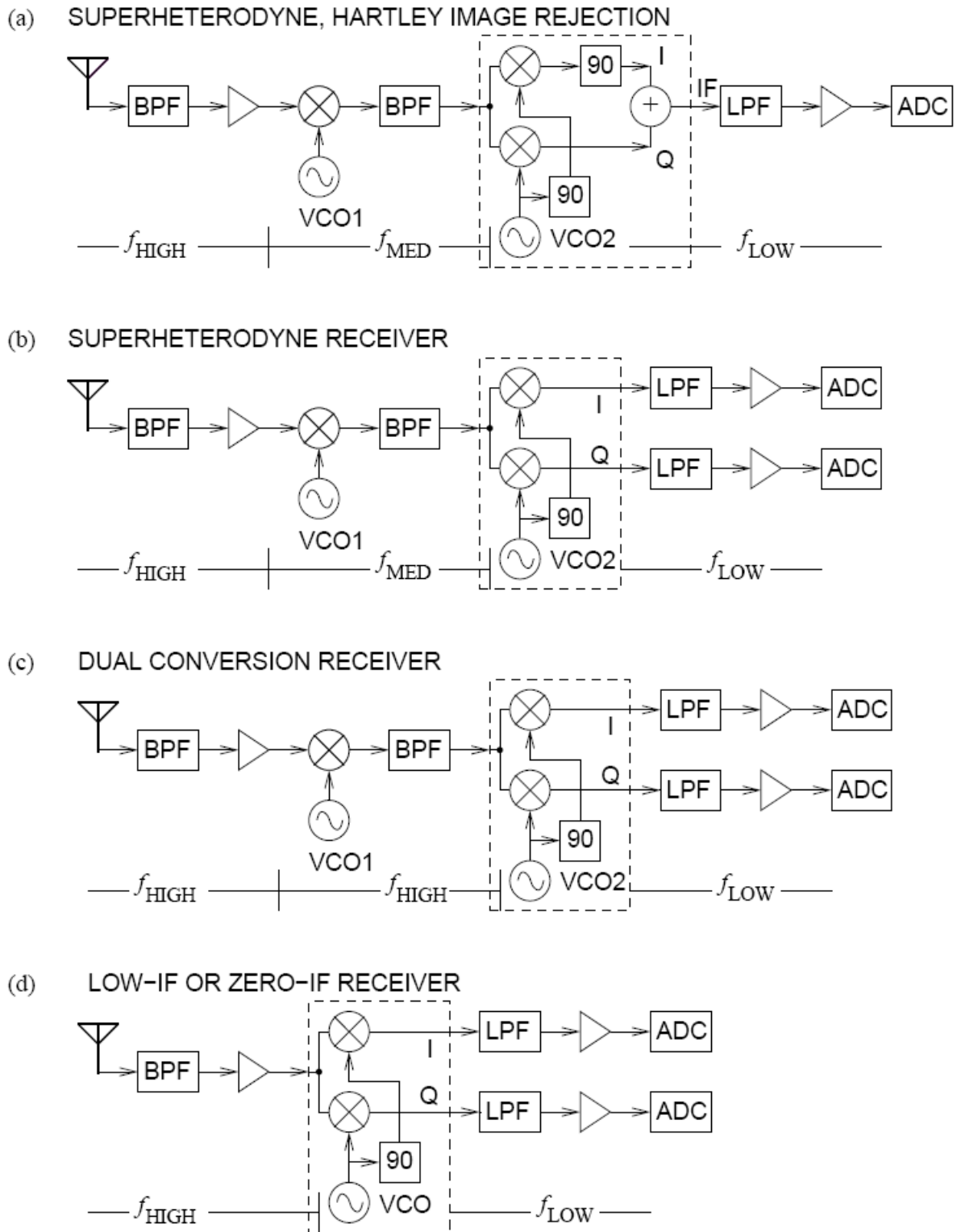


Figure 1-57 Architecture of modern receivers: (a) superheterodyne receiver using the Hartley architecture for image rejection; (b) superheterodyne receiver; (c) dual-conversion receiver; low-IF or zero-IF receiver. PBF, bandpass filter; LBF, LowPass Filter; ADC, analog to digital converter; VCO, voltage controlled oscillator; 90, 90° phase shifter; I, in-phase component, Q, Quadrature component; f_{HIGH} , f_{MED} and f_{LOW} indicate relatively high, medium and low frequencies in the corresponding section of the receiver.

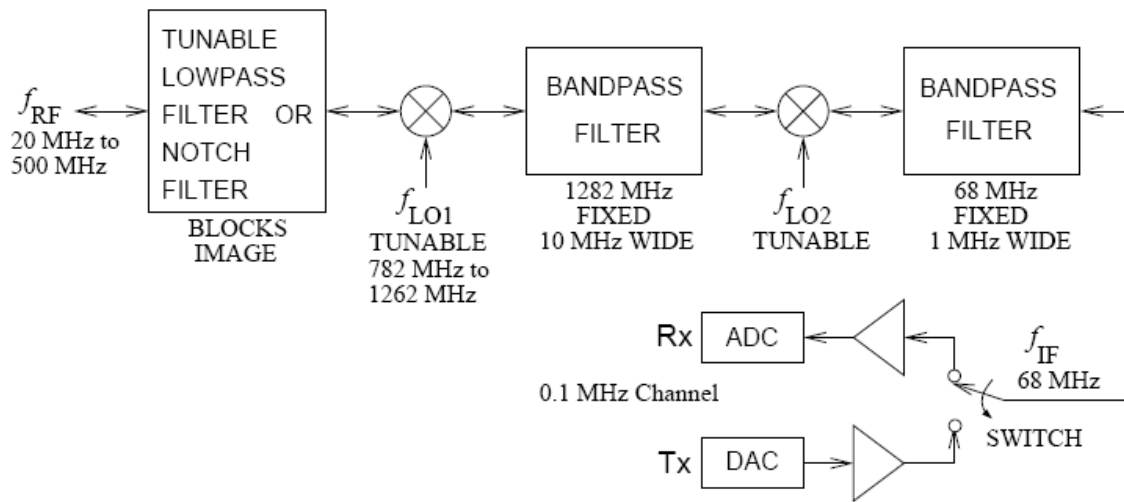


Figure 1-64 Bilateral double conversion transceiver for wideband operation.

Summary

This chapter presented the RF frontend architectures used from the beginnings of wireless communications up to those used in modern systems. Similar architectures are used in the frontends of radar and sensor systems. Wireless systems proliferate, and even in established domains such as cellphones, architectures are evolving to achieve greater efficiency, greater multifunctionality, and lower cost primarily by monolithically integrating and digitizing as much as possible of the RF frontend. Size drives the replacement of superheterodyne architecture by eliminating large intermediate filters.

16.1.1 Transmitter design from http://en.wikibooks.org/wiki/Electronics/Transmitter_design

Radio transmitter design is a complex topic which can be broken down into a series of smaller topics.

Contents

1 Frequency synthesis and frequency multiplication

1.1 Synthesis

1.1.1 Fixed frequency systems

1.1.2 Variable frequency systems

1.2 Multiplication

2 Frequency mixing and Modulation

2.1 AM modes

2.1.1 Low level and High level

2.1.1.1 Low level

2.1.1.2 High level

Basics

2.1.2 Types of AM modulators

2.1.2.1 Plate AM modulators

2.1.2.2 Screen AM modulators

2.2 Other modes which are related to AM

2.2.1 Single-sideband modulation

2.2.1.1 Filter method

2.2.1.2 Phasing method

2.2.2 Vestigial-sideband modulation

2.2.3 Morse

2.3 FM modes

2.3.1 Direct FM

2.3.2 Indirect FM

3 RF power amplifiers

3.1 Valves

3.1.1 Advantages of valves

3.1.2 Disadvantages of valves

3.2 Solid state

4 Linking the transmitter to the aerial

5 EMC matters

5.1 RF leakage (defective RF shielding)

5.2 Spurious emissions

5.2.1 Harmonics

5.2.2 Local oscillators and unwanted mixing products

5.2.3 Instability and parasitic oscillations

6 Reference

7 Further reading

16.1.1.1 Frequency synthesis and frequency multiplication

Synthesis

Fixed frequency systems

For a fixed frequency transmitter one commonly used method is to use a resonant quartz crystal in a Crystal oscillator to fix the frequency. For transmitter where the frequency has to be able to be varied then several options can be used.

Variable frequency systems

An array of crystals—This approach uses several oscillators, each tuned to a different fixed frequency.

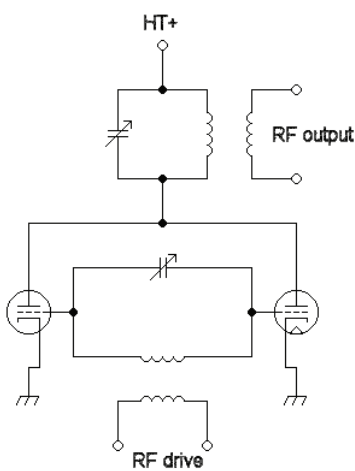
Variable frequency oscillator (VFO)

Phase locked loop (PLL) frequency synthesizer

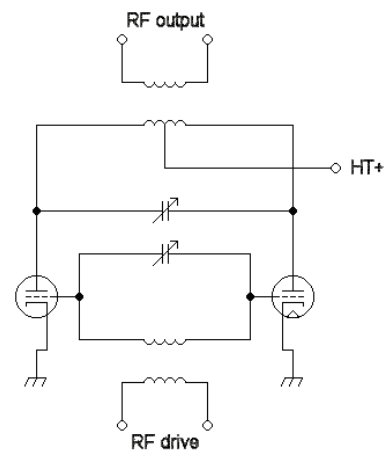
Multiplication

It is often the case for VHF transmitters that it is not possible to operate the crystal controlled or variable frequency oscillator at the frequency of the final output. Also, for reasons including frequency stability, it is better to multiply the frequency of the free running oscillator up to the final frequency which is required.

If the output of a amplifier stage is tuned to a multiple of the frequency which the stage is driven with, the stage is optimised to give a larger harmonic output than that found in a linear amplifier. In a push-push stage, the output will only contain the even harmonics. This is because the currents which would generate the fundamental and the odd harmonics in this circuit (if one valve was removed) are canceled out by the second valve. Note that in these diagrams that the bias supplies and the neutralization have been omitted for clarity. In a real system it is likely that tetrodes would be used as plate to grid capacitance in a tetrode is lower so making the stage less likely to be unstable.



Here in the push-pull stage the output will only contain the odd harmonics because of the canceling effect.



16.1.1.2 Frequency mixing and Modulation

The task of many transmitters is to transmit some form of information using a carrier wave. This process is called modulation. There are many types of RF modulation, and the choice of modulation often depends on the type of information being transmitted.

For instance, audio information is continuous in time and value, and scaling by a constant (i.e. signal inversion, volume control) is acceptable, so AM and FM transmission work. But for digital communications, the signal is discrete in time and discrete in value, and inversion of the signal is unacceptable, so AM and FM are not (on their own) satisfactory. For digital communications, a modulation such as frequency shift keying (FSK) or on-off keying (OOK) over FM would be better.

Basics

AM modes

In many cases the carrier wave is mixed with another electrical signal to impose upon it the information. This occurs in Amplitude modulation (AM).

Low level and High level

Low level

Here a small audio stage is used to modulate a low power stage, the output of this stage is then amplified using a linear RF amplifier.

Advantages

The advantage of using a linear RF amplifier is that the smaller early stages can be modulated, which only requires a small audio amplifier to drive the modulator.

Disadvantages

The great disadvantage of this system is that the amplifier chain is less efficient, because it has to be linear to preserve the modulation. Hence class C amplifiers cannot be employed.

An approach which marries the advantages of low-level modulation with the efficiency of a Class C power amplifier chain is to arrange a feedback system to compensate for the substantial distortion of the AM envelope. A simple detector at the transmitter output (which can be little more than a loosely coupled diode) recovers the audio signal, and this is used as negative feedback to the audio modulator stage. The overall chain then acts as a linear amplifier as far as the actual modulation is concerned, though the RF amplifier itself still retains the Class C efficiency. This approach is widely used in practical medium power transmitters, such as AM radiotelephones.

High level

Advantages

One advantage of using class C amplifiers in a broadcast AM transmitter is that only the final stage needs to be modulated, and that all the earlier stages can be driven at a constant level. These class C stages will be able to generate the drive for the final stage for a smaller DC power input. However in many designs in order to obtain better quality AM the penultimate RF stages will need to be subject to modulation as well as the final stage.

Disadvantages

A large audio amplifier will be needed for the modulation stage, at least equal to the power of the transmitter output itself. Traditionally the modulation is applied using an audio transformer, and this can be bulky. Direct coupling from the audio amplifier is also possible (known as a cascode arrangement), though this usually requires quite a high DC supply voltage (say 30V or more), which is not suitable for mobile units.

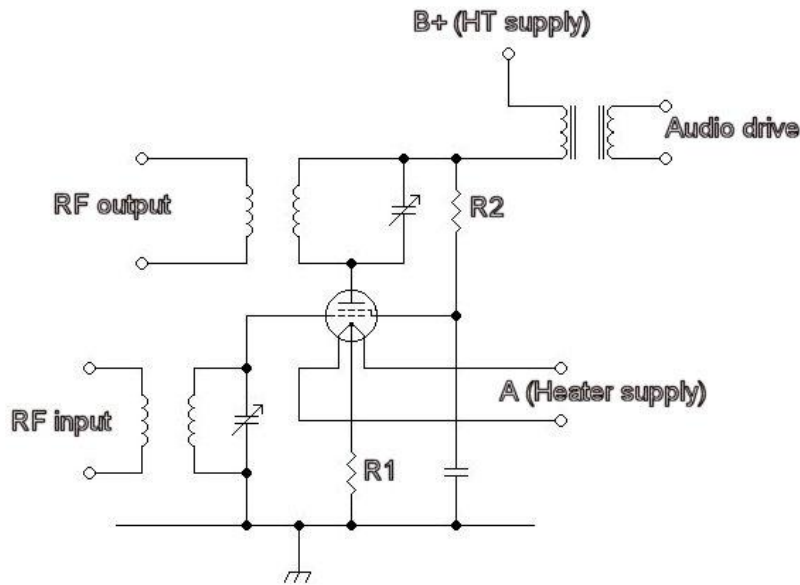
Types of AM modulators

A wide range of different circuits have been used for AM. While it is perfectly possible to create good designs using solid-state electronics, valved (tube) circuits are shown here. In general, valves

are able to easily yield RF powers far in excess of what can be achieved using solid state. Most high-power broadcast stations still use valves.

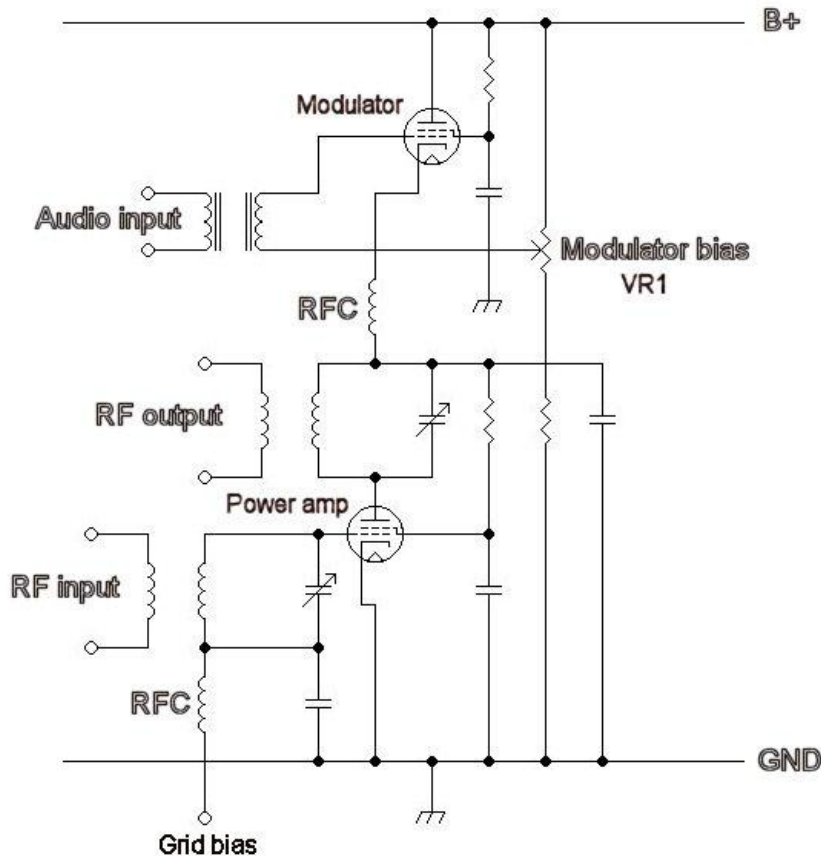
Plate AM modulators

In plate modulation systems the voltage delivered to the stage is changed. As the power output available is a function of the supply voltage, the output power is modulated. This can be done using a transformer to alter the anode (plate) voltage. The advantage of the transformer method is that the audio power can be supplied to the RF stage and converted into RF power.



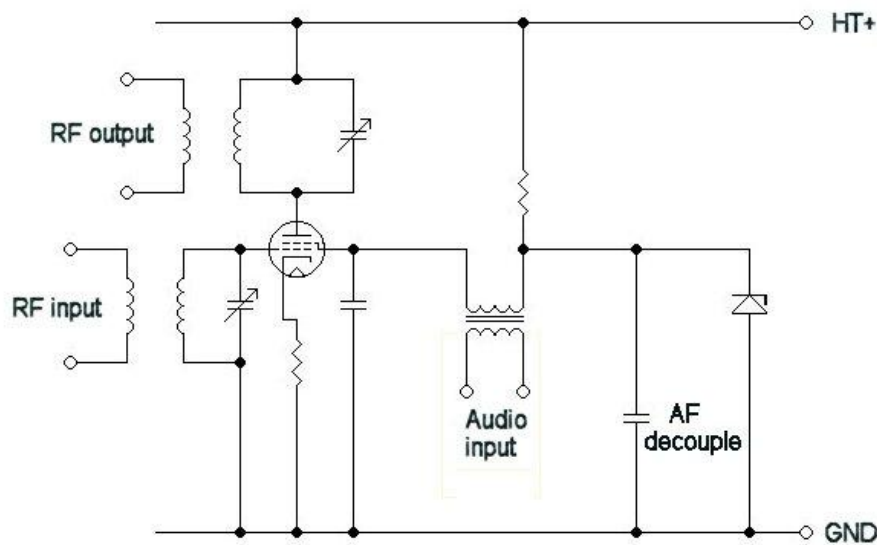
Anode modulation using a transformer. The tetrode is supplied with an anode supply (and screen grid supply) which is modulated via the transformer. The resistor R1 sets the grid bias, both the input and outputs are tuned LC circuits which are tapped into by inductive coupling.

Basics



An example of a series modulated amplitude modulation stage. The tetrode is supplied with an anode supply (and screen grid supply) which is modulated by the modulator valve. The resistor VR1 sets the grid bias for the modulator valve, both the RF input (tuned grid) and outputs are tuned LC circuits which are tapped into by inductive coupling. When the valve at the top conducts more than the potential difference between the anode and cathode of the lower valve (RF valve) will increase. The two valves can be thought of as two resistors in a potentiometer.

Screen AM modulators



Under steady state conditions (no audio driven) the stage will be a simple RF amplifier where the grid bias is set by the cathode current. When the stage is modulated the screen potential changes and so alters the gain of the stage.

Other modes which are related to AM

Several derivatives of AM are in common use. These are

Single-sideband modulation

(SSB, or SSB-AM single-sideband full carrier modulation), very similar to single-sideband suppressed carrier modulation (SSB-SC)

Filter method

Using a balanced mixer a double side band signal is generated, this is then passed through a very narrow bandpass filter to leave only one side-band. By convention it is normal to use the upper sideband (USB) in communication systems, except for HAM radio when the carrier frequency is below 10 MHz here the lower side band (LSB) is normally used.

Phasing method

The phasing method is another way to generate of single sideband signals. One of the weaknesses of this method is the need for a network which imposes a constant 90° phase shift on audio signals throughout the entire audio spectrum. By reducing the audio bandwidth the task of designing the phaseshift network can be made more easy.

Imagine that the audio is a single sine wave $E = E^{\circ} \text{ sine } (\omega t)$

The audio signal is passed through the phase shift network to give two identical signals which differ by 90°.

Basics

So as the audio input is a single sine wave the outputs will be

$$E = E^{\circ} \sin(\omega t)$$

and

$$E = E^{\circ} \cos(\omega t)$$

These audio outputs are mixed in non linear mixers with a carrier, the carrier drive for one of these mixers is shifted by 90°. The output of these mixers is combined in a linear circuit to give the SSB signal.

Vestigial-sideband modulation[]

Vestigial-sideband modulation (VSB, or VSB-AM) is a type of modulation system commonly used in TV systems, it is normal AM which has been passed through a filter which removes one of the sidebands.

Morse

Strictly speaking the commonly used 'AM' is double-sideband full carrier. Morse is often sent using on-off keying of an unmodulated carrier(Continuous wave), this can be thought of as an AM mode.

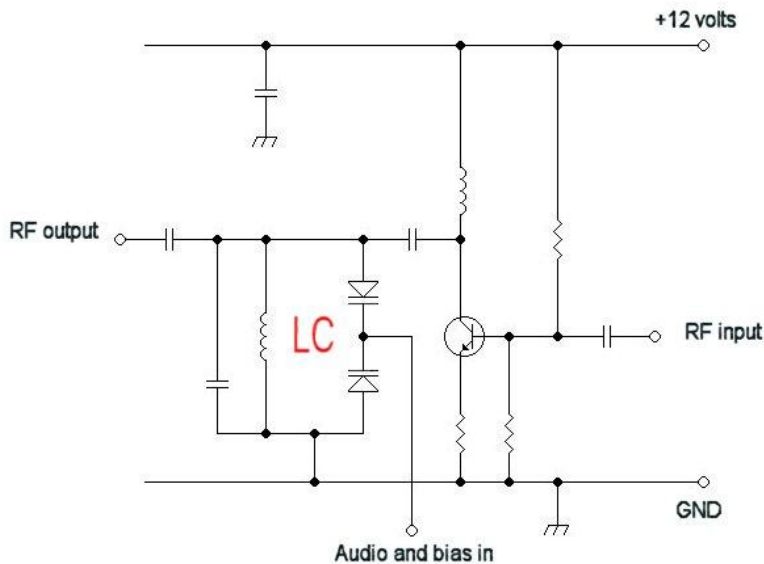
FM modes

Direct FM

Direct FM (true Frequency modulation) is where the frequency of an oscillator is altered to impose the modulation upon the carrier wave. This can be done by using a voltage controlled capacitor (Varicap diode) in a crystal controlled oscillator. The frequency of the oscillator is then multiplied up using a frequency multiplier stage, or is translated upwards using a mixing stage to the output frequency of the transmitter.

Indirect FM

Indirect FM employs varicap diode to impose a phase shift (which is voltage controlled) in a tuned circuit which is fed with a plain carrier. This is termed Phase modulation, the modulated signal from a phase modulated stage can be understood with a FM receiver but for good audio quality the audio applied to the phase modulation stage.



This is a solid state circuit, on the right a RF drive is applied to the base of the transistor, the tank circuit (LC) connected to the collector via a capacitor contains a pair of varicap diodes. As the voltage applied to the varicaps is changed the phase shift of the output will change.

Sigma-delta modulation ($\Sigma\Delta$)

16.1.1.3 RF power amplifiers

Valves

For high power systems it is normal to use valves, please see Valved RF amplifiers for details of how valved RF power stages work.

Advantages of valves

Good for high power systems

Electrically very robust, they can tolerate overloads for minutes which would destroy bipolar transistor systems in milliseconds

Disadvantages of valves

Heater supplies are required for the cathodes

High voltages (*Threat of death*) are required for the anodes

Valves have a shorter working life than solid state parts because the heaters tend to fail

Solid state

For low and medium power it is often the case that solid state power stages are used. Sadly for high power systems these cost more per Watt of output power than a valved system.

16.1.1.4 Linking the transmitter to the aerial

The vast majority of modern equipment is designed to operate with a resistive load driven via coaxial cable of one particular impedance, often 50 ohms. To connect the aerial to this coaxial cable

Basics

transmission line a matching network and/or a balun may be required. Commonly a SWR meter and/or an Antenna analyzer are used to check the goodness of the match between the aerial system and the transmission line (feeder).

See [Antenna tuner](#) and [balun](#) for details of matching networks and baluns respectively.

16.1.1.5 EMC matters

While this section was written from the point of view of a radio ham with relation to Television interference (radio transmitter interference) it applies to the construction and use of all radio transmitters, and other electronic devices which generate high RF powers with no intention of radiating these. For instance a dielectric heater might contain a 2000 Watt 27 MHz source within it, if the machine operates as intended then none of this RF power will leak out. However if the device is subject to a fault then when it operates RF will leak out and it will be now a transmitter. Also computers are RF devices, if the cases are poorly made then the computer will radiate at VHF.

For example if you attempt to tune into a weak *FM* radio station (88 to 108 MHz, band II) at your desk you may lose reception when you switch on your PC. Equipment which is not intended to generate RF, but does so through for example sparking at switch contacts is not considered here, for a consideration of such matters please see Television interference (electrical interference) for further details.

RF leakage (defective RF shielding)

All equipment using RF electronics should be inside a screened metal box, all connections in or out of the metal box should be filtered to avoid the ingress or egress of radio signals. A common and effective method of doing so for wires carrying DC supplies, 50 Hz AC connections, audio and control signals is to use a feedthrough capacitor. This is a capacitor which is mounted in a hole in the shield, one terminal of the capacitor is its metal body which touches the shielding of the box while the other two terminals of the capacitor are on either side of the shield. The feed through capacitor can be thought of as a metal rod which has a dielectric sheath which in turn has a metal coating.

In addition to the feed through capacitor, either a resistor or RF choke can be used to increase the filtering on the lead. In transmitters it is vital to prevent RF from entering the transmitter through any lead such as a power, microphone or control connection. If RF does enter a transmitter in this way then an instability known as motorboating can occur. Motorboating is an example of a self-inflicted EMC problem.

If a transmitter is suspected of being responsible for a television interference problem then it should be run into a dummy load, this is a resistor in a screened box or can which will allow the transmitter to generate radio signals without sending them to the antenna. If the transmitter does not cause interference during this test then it is safe to assume that a signal has to be radiated from the antenna to cause a problem. If the transmitter does cause interference during this test

Communication Basics

then a path exists by which RF power is leaking out of the equipment, this can be due to bad shielding. This is a rare but insidious problem and it is vital that it is tested for.

You are most likely to see this leakage on homemade equipment or equipment which has been modified. It is also possible to observe RF leaking out of microwave cookers.

Spurious emissions

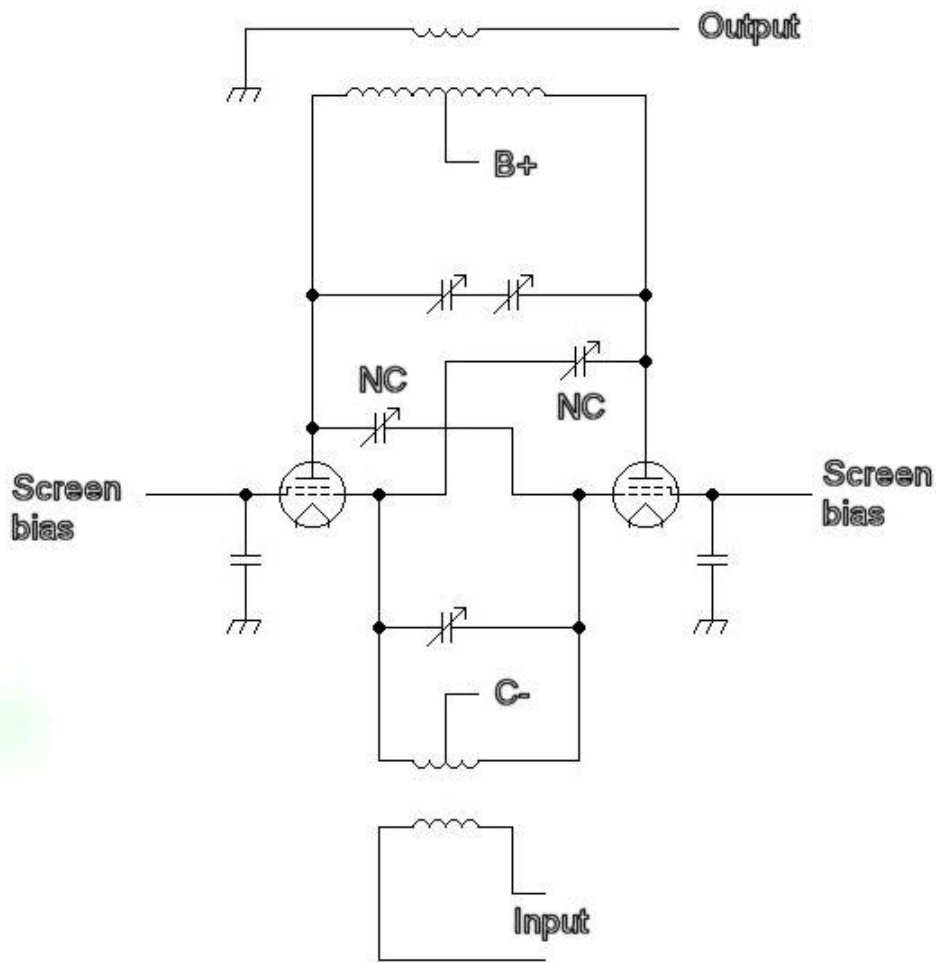
Early in the development of radio technology it was recognized that the signals emitted by transmitters had to be 'pure'. For instance Spark-gap transmitters were quickly outlawed as they give an output which is so wide in terms of frequency. In modern equipment there are three main types of spurious emissions.

The term Spurious emissions refers to any signal which comes out of a transmitter other than the wanted signal. The spurious emissions include harmonics, out of band mixer products which are not fully suppressed and leakage from the local oscillator and other systems within the transmitter.

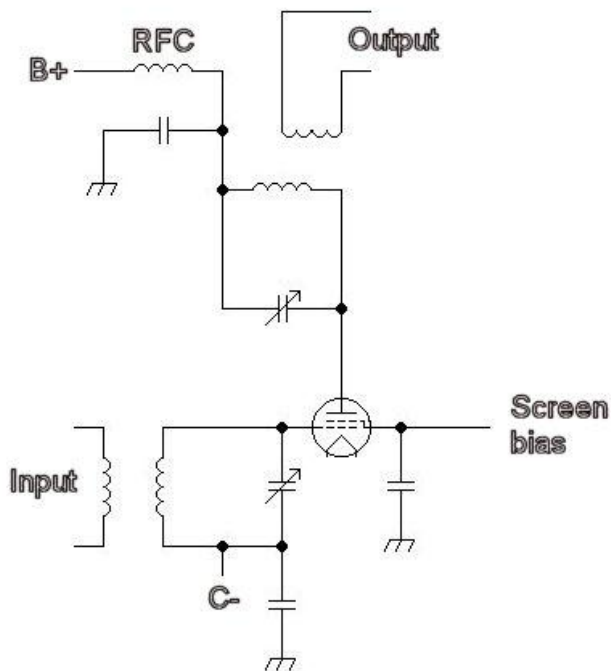
Harmonics

These are multiples of the operation frequency of the transmitter, they can be generated in a stage of the transmitter even if it is driven with a perfect sine wave because no real life amplifier is perfectly linear. It is best if these harmonics are designed out at an early stage. For instance a push-pull amplifier consisting of two tetrode valves attached to an anode tank resonant LC circuit which has a coil which is connected to the high voltage DC supply at the centre (Which is also RF ground) will only give a signal for the fundamental and the odd harmonics.

Basics



Here is a slightly worse design which only has one tetrode, while perfectly good designs have been made using this circuit it does have more potential shortcomings than the above circuit.



Communication Basics

In addition to the good design of the amplifier stages, the transmitter's output should be filtered with a low pass filter to reduce the level of the harmonics.

The harmonics can be tested for using a RF spectrum analyser (expensive) or with an absorption wavemeter (cheap). If a harmonic is found which is at the same frequency as the frequency of the signal wanted at the receiver then this spurious emission can prevent the wanted signal from being received.

Local oscillators and unwanted mixing products

Imagine a transmitter, which has an intermediate frequency (IF) of 144 MHz, which is mixed with 94 MHz to create a signal at 50 MHz, which is then amplified and transmitted. If the local oscillator signal was to enter the power amplifier and not be adequately suppressed then it could be radiated. It would then have the potential to interfere with radio signals at 94 MHz in the FM audio (band II) broadcast band. Also the unwanted mixing product at 238 MHz could in a poorly designed system be radiated. Normally with good choice of the intermediate and local oscillator frequencies this type of trouble can be avoided, but one potentially bad situation is in the construction of a 144 to 70 MHz converter, here the local oscillator is at 74 MHz which is very close to the wanted output. Good well made units have been made which use this conversion but their design and construction has been challenging. This problem can be thought of as being related to the Image response problem which exists in receivers.

One method of reducing the potential for this transmitter defect is the use of balanced and double balanced mixers. If the equation is assumed to be

$$E = E_1 \cdot E_2$$

and is driven by two simple sine waves, f_1 and f_2 then the output will be a mixture of four frequencies

$$f_1$$

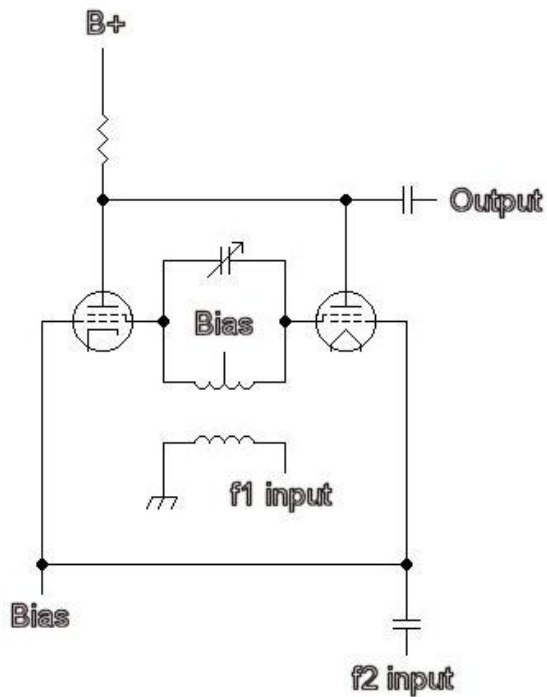
$$f_1+f_2$$

$$f_1-f_2$$

$$f_2$$

If the simple mixer is replaced with a balanced mixer then the number of possible products is reduced. Imagine that two mixers which have the equation $\{I = E_1 \cdot E_2\}$ are wired up so that the current outputs are wired to the two ends of a coil (the centre of this coil is wired to ground) then the total current flowing through the coil is the difference between the output of the two mixer stages. If the f_1 drive for one of the mixers is phase shifted by 180° then the overall system will be a balanced mixer.

Basics



$$E = K \cdot E_{f2} \cdot \Delta E_{f1}$$

So the output will now have only three frequencies

$$f_1 + f_2$$

$$f_1 - f_2$$

$$f_2$$

Now as the frequency mixer has fewer outputs the task of making sure that the final output is *clean* will be simpler.

Instability and parasitic oscillations

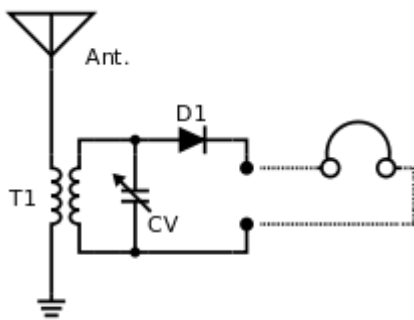
If a stage in a transmitter is unstable and is able to oscillate then it can start to generate RF at either a frequency close to the operating frequency or at a very different frequency. One good sign that it is occurring is if a RF stage has a power output even without being driven by an exciting stage. Another sign is if the output power suddenly increases wildly when the input power is increased slightly, it is noteworthy that in a class C stage that this behaviour can be seen under normal conditions. The best defence against this transmitter defect is a good design, also it is important to pay good attention to the neutralization of the valves or transistors.

Receiver Design from [http://en.wikipedia.org/wiki/Tuner_\(electronics\)](http://en.wikipedia.org/wiki/Tuner_(electronics))

Reference

Radiocommunication handbook (RSGB), ISBN 0900612584⁷

16.2 Receiver Design from [http://en.wikipedia.org/wiki/Tuner_\(electronics\)](http://en.wikipedia.org/wiki/Tuner_(electronics))



Inductively coupled [crystal radio](#) receiver

The simplest tuner consists of an [inductor](#) and [capacitor](#) connected in parallel, where the capacitor or inductor is made to be variable. This creates a [resonant circuit](#) which responds to an alternating current at one frequency. Combined with a [detector](#), also known as a [demodulator](#), (diode D-1, in the circuit), it becomes the simplest radio receiver, often called a [crystal set](#).

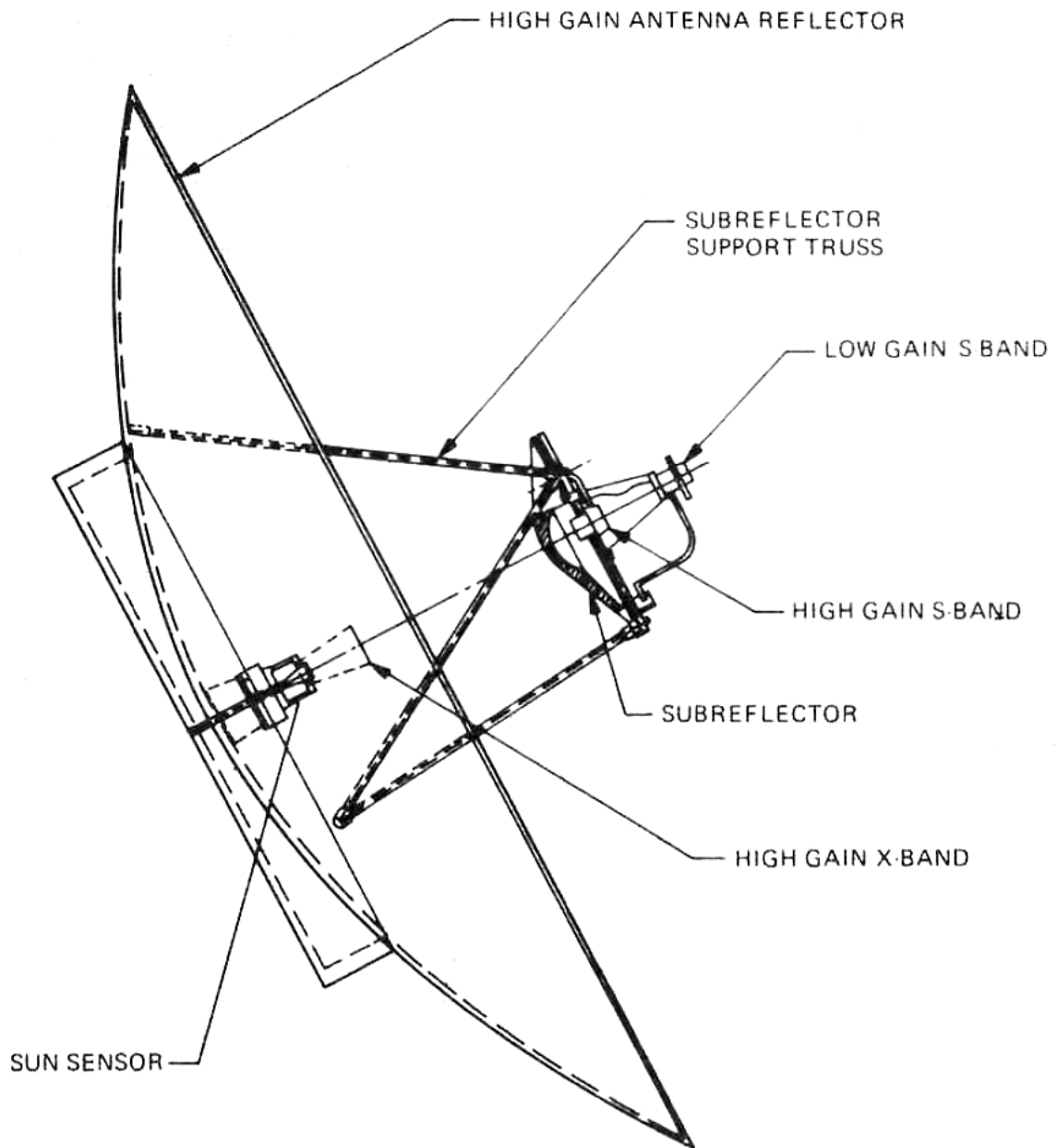
Practical radio tuners use a [superheterodyne receiver](#). Older models would realize manual tuning by means of mechanically operated ganged variable capacitors. Often several sections would be provided on a tuning capacitor, to tune several stages of the receiver in tandem, or to allow switching between different frequency bands. A later method used a [potentiometer](#) supplying a variable voltage to [varactor diodes](#) in the local oscillator and tank circuits of front end tuner, for electronic tuning. Still later, [phase locked loop](#) methods were used, with [microprocessor](#) control.

In a self-contained radio receiver for audio, the signal from the detector after the tuner is run through a volume control and to an amplifier stage. The amplifier feeds either an internal speaker or headphones. In a tuner component of an audio system (for example, a home high-fidelity system or a public address system in a building), the output of the detector is connected to a separate external system of amplifiers and speakers.

The broadcast audio FM band (88 - 108 MHz in most countries) is around 100 times higher in frequency than the AM band and provides enough space for a bandwidth of 50 kHz. This bandwidth is sufficient to transmit both stereo channels with almost the full bandwidth of the human ear. Sometimes, additional subcarriers are used for unrelated audio or data transmissions. The left and right audio signals must be combined into a single signal which is applied to the modulation input of the transmitter; this is done by the addition of an inaudible subcarrier signal to the FM broadcast signal. [FM stereo](#) allows left and right channels to be transmitted. The availability of FM stereo, a quieter VHF broadcast band, and better fidelity lead to the specialization of [FM broadcasting](#) in music, tending to leave AM broadcasting with spoken-word material.

⁷ <http://en.wikibooks.org/wiki/Special:BookSources/0900612584>

16.3 Antenna



An **antenna** (or **aerial**) is an electrical device which converts [electric power](#) into [radio waves](#), and vice versa.^[1] It is usually used with a [radio transmitter](#) or [radio receiver](#). In [transmission](#), a radio transmitter supplies an oscillating [radio frequency](#) electric current to the antenna's terminals, and the antenna radiates the energy from the current as [electromagnetic waves](#) (radio waves). In reception, an antenna intercepts some of the power of an electromagnetic wave in order to produce a tiny voltage at its terminals, that is applied to a receiver to be [amplified](#).

Antennas are essential components of all equipment that uses [radio](#). They are used in systems such as [radio broadcasting](#), [broadcast television](#), [two-way radio](#), [communications receivers](#), [radar](#), [cell phones](#), and [satellite communications](#), as well as other devices such as [garage door openers](#), [wireless](#)

Antenna

[microphones](#), [bluetooth](#) enabled devices, [wireless computer networks](#), [baby monitors](#), and [RFID tags](#) on merchandise.

Typically an antenna consists of an arrangement of metallic [conductors](#) ([elements](#)), electrically connected (often through a [transmission line](#)) to the receiver or transmitter. An oscillating current of [electrons](#) forced through the antenna by a transmitter will create an oscillating [magnetic field](#) around the antenna elements, while the [charge](#) of the electrons also creates an oscillating [electric field](#) along the elements. These time-varying fields radiate away from the antenna into space as a moving transverse electromagnetic field wave. Conversely, during reception, the oscillating electric and magnetic fields of an incoming radio wave exert force on the electrons in the antenna elements, causing them to move back and forth, creating oscillating currents in the antenna.

Antennas may also include reflective or directive elements or surfaces not connected to the transmitter or receiver, such as [parasitic elements](#), [parabolic reflectors](#) or [horns](#), which serve to direct the radio waves into a beam or other desired [radiation pattern](#). Antennas can be designed to transmit or receive radio waves in all directions equally ([omnidirectional antennas](#)), or transmit them in a beam in a particular direction, and receive from that one direction only ([directional](#) or [high gain](#) antennas).

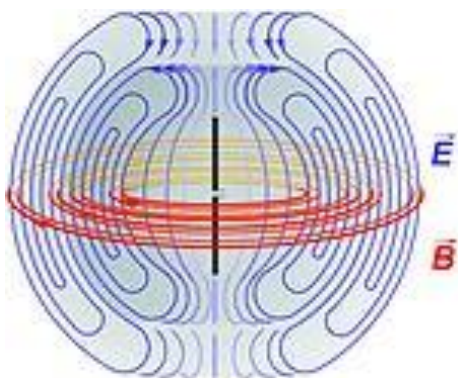


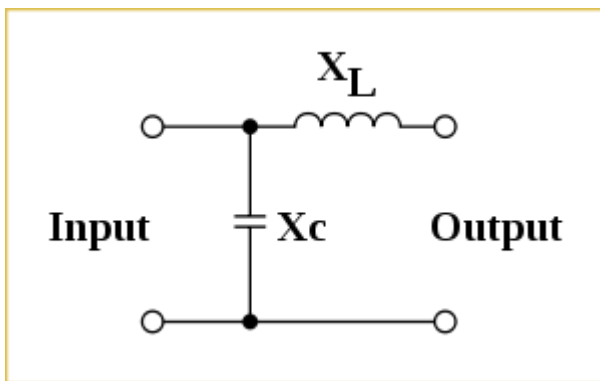
Diagram of the electric fields (blue) and magnetic fields (red) radiated by a dipole antenna (black rods) during transmission. Large parabolic antenna for communicating with spacecraft.

Antenna tuner

An **antenna tuner**, **transmatch** or **antenna tuning unit** (ATU) is a device connected between a [radio transmitter](#) or receiver and its [antenna](#) to improve power transfer between them by [matching](#) the [impedance](#) of the radio to the antenna. An antenna tuner matches a transceiver with a fixed impedance (typically 50 [ohms](#) for modern transceivers) to a load (feed line and [antenna](#)) impedance which is unknown, complex or otherwise does not match. An ATU allows the use of one antenna on a broad range of frequencies. An antenna and transmatch are not as efficient as a [resonant](#) antenna due to feedline losses due to the [SWR](#) (multiple reflections) and losses in the ATU itself. An ATU is an antenna matching unit, and cannot change the resonant frequency of the

Basics

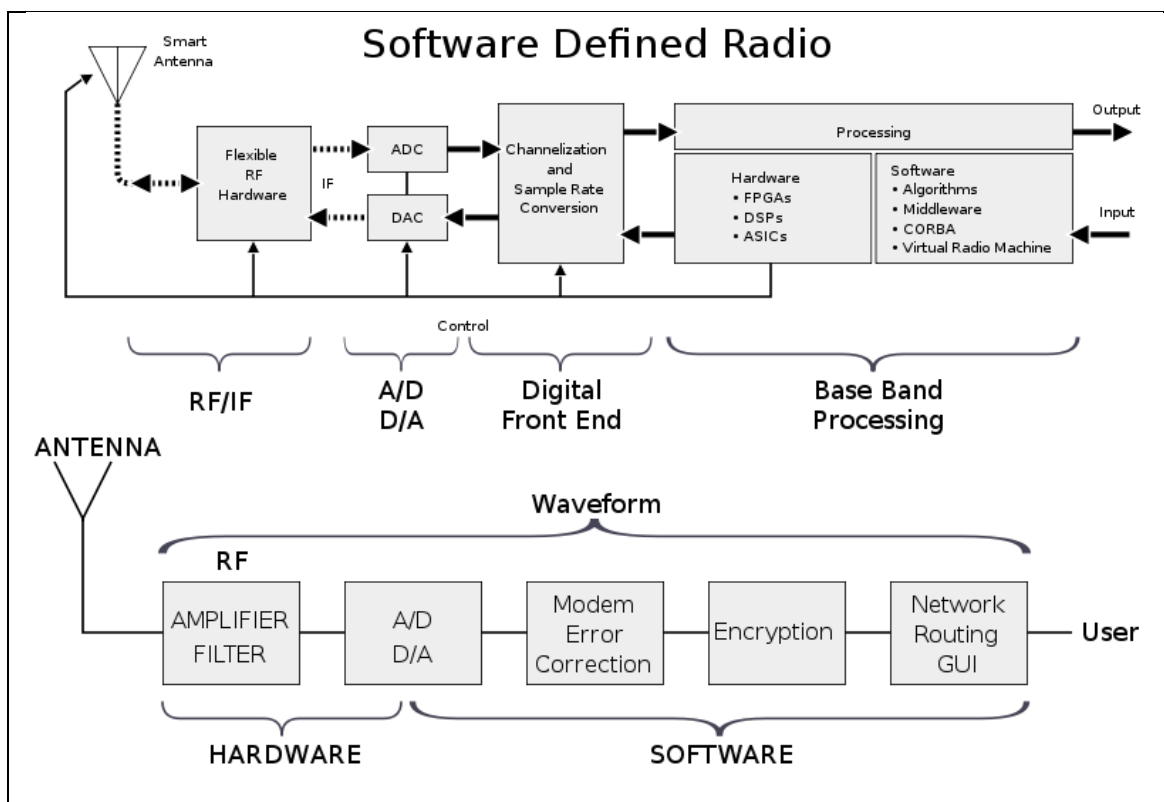
aerial. Similar matching networks are used in other equipment (such as [linear amplifiers](#)) to transform impedance.



Basic network for an antenna tuner

16.4 Software Defined Radio (SDR)

Software Defined Radio is a Wireless communication in which the transmitter modulation is generated or defined by a computer, and the receiver uses a computer to recover the signal intelligence? To select the desired modulation type, the proper programs must be run by microcomputers that control the transmitter and receiver



SDR block diagram

16.5 HSDR (High Definition Software Defined Radio)

HSDR (High Definition Software Defined Radio) is a freeware SDR program for Microsoft Windows 2000/XP/Vista/7/8. Typical applications are Radio listening, Ham Radio, SWL, Radio

ExtIO.dll

Astronomy, NDB-hunting and Spectrum analysis. HDSDR (former WinradHD) is an advanced version of Winrad, written by Alberto di Bene

16.6 ExtIO.dll

The HDSDR software doesn't communicate with the SDR hardware directly. It communicates with the SDR radio hardware through an **External Input Output Dynamic Link Library** (ExtIO-DLL) file, which is a type of plug-in. Alberto di Bene designed the DLL interface so that Winrad can operate with a wide range of SDR hardware. We extended the DLL-interface to support TX switching. Winrad and HDSDR can support new hardware radios using an ExtIO-DLL file without the need to change the HDSDR software. ExtIO DLL files are written by anyone who wishes to provide support for any particular SDR hardware. In this manner, several radios can be used with a single piece of software. The software in this case is HDSDR.

16.7 How do I develop an ExtIO.dll ?

We assume that you are a software developer familiar with C/C++ programming. Here is a "hopefully" well documented [header file](#) ⁸, which specifies the interface between HDSDR and an ExtIO-DLL. Here is also an example [ExtIO DLL](#) ⁹ with sources as public-domain, developed with Microsoft Visual C++ 2008 Express ion.

16.8 Visual C++ 2008 Express

Visual C++ is part of the *Visual Studio Programming Suite*. A light *express* version is freely available. Visual Studio is an **Integrated Development Environment** (IDE) for developing web applications, client applications, and Windows Phone mobile applications. It supports C, C++, C#, Visual Basic

16.9 Qt

Qt is designed for developing applications and user interfaces once and deploying them across several desktop and mobile operating systems. The easiest way to start application development with Qt is to download and install Qt 5. It contains Qt libraries, examples, documentation, and the necessary development tools, such as the Qt Creator integrated development environment (IDE)

16.10 RF hardware (USB Stick)

For the RF hardware there are some existing SDR USB sticks we can use it:

16.10.1 TERRATEC ran T stick DVB-T/DAB/DAB + Stick USB 2.0 ¹⁰

⁸ Guide\LC_ExtIO_Types.h - reference: http://www.hdsdr.de/download/LC_ExtIO_Types.h

⁹ Guide\ExtIO_Demo_101\.. - reference: http://hdsdr.de/download/ExtIO/ExtIO_Demo_101.zip

¹⁰ Reference: <http://www.amazon.de/Terratec-ran-T-Stick-DVB-T-schwarz/dp/B007EB995U/>

Basics



Price: EUR 29.98

The ultimate all-rounder for the digital TV and radio reception on your PC

USB stick for DVB-T (TV) and DAB / DAB + (radio). Direct recording and programming via EPG recording Software for both television and radio reception. Support for all major DVB-T Features

16.10.2 Hackrf (an-open-source-SDR-platform) ¹¹



Price: about 300\$

Transmit or receive any radio signal from 30 MHz to 6000 MHz on USB power with HackRF.

HackRF is an open source hardware project to build a Software Defined Radio (SDR) peripheral.

16.11 RF Overview

Radio Frequency (RF) is a rate of oscillation in the range of about 9 kHz to 300 GHz, which corresponds to the frequency of radio waves, and the alternating currents which carry radio signals. It is the use of radio signals to communicate real-time data from the warehouse floor to the WMS database and back to the floor.

This expes processing in the warehouse. Scanners collect the data and transmit it via radio frequency to antennas located throughout the warehouse. From the antennas, the signal proceeds to an access point that communicates with the warehouse management system. This process reduces paper, data entry time delays, cycle count processing, out of stock quantities, typing errors...

¹¹ The primary web page for HackRF is: <http://greatscottgadgets.com/hackrf/>

16.12 RF Frequencies policies

The LNFT allocates Lebanon’s radiofrequency spectrum into a number of frequency bands relevant to ITU regulations and specifies the general purposes for which the bands may be used. This process is referred to as the allocation of frequency bands to radio communication services. The primary objectives to be achieved with the radio spectrum are:

- To harmonize spectrum use with international developments. In this regard, Lebanon follows closely the work of the ITU, the CEPT, the league of Arab States and the local regional organization
- To manage the radio spectrum within Lebanon taking into account the governmental requirements and the needs of the various commercial sectors
- To stimulate technological innovation and competitiveness

The LNFT will be updated from time to time dependant on international initiatives and national decisions. The main source documentation used in the development of this version of LNFT was the ITU Radio Regulations and the Provisional Final Acts of the ITU WRC07.

Each band may be allocated to one or more services. The services printed in capitals are called “primary” services; the names which printed in small characters are called “secondary” services. Stations of a secondary service shall not cause harmful interference to stations of primary service and cannot claim protection from harmful interference from stations of a primary service.

For our purpose we should use the Amateurs service bands which are specified in the following table¹²:

Frequency Band (kHz MHz or GHz)	International Region 1 Allocation	National Allocation	Main application	Notes
135.7 – 137.8 kHz	FIXED MARITIME MOBILE Amateur 5.4C03 5.64 5.67 5.4C04	FIXED MARITIME MOBILE 5.64 5.67 5.4C04	SRD Maritime applications Ultra Low Power Active Medical Implants	ERC REC 62-01
1810-1830 kHz	AMATEUR 5.98 5.99 5.100 5.101	AMATEUR FIXED 5.98 MOBILE except aeronautical mobile 5.100	Amateur applications	
1830-1850 kHz		AMATEUR	Amateur applications	
3500-3800 kHz	AMATEUR FIXED MOBILE aeronautical 5.92	AMATEUR FIXED MOBILE aeronautical 5.92	Amateur applications	

¹² This table is a part from the LNFT document publish on 28-06-2008

Basics

7000-7100 kHz	AMATEUR AMATEURSATELLITE 5.140 5.141 5.141A	AMATEUR AMATEURSATELLITE	Amateur applications	
7100-7200 kHz	AMATEUR 5.141A 5.141B 5.141C 5.142	Amateur LBN 3 BROADCASTING 5.141C		
10100-10150 kHz	FIXED Amateur	FIXED Amateur		
14000-14250 kHz	AMATEUR AMATEURSATELLITE	AMATEUR AMATEURSATELLITE		
14250-14350 kHz	AMATEUR 5.152	AMATEUR		
18068-18168 kHz	AMATEUR AMATEURSATELLITE 5.154	AMATEUR AMATEURSATELLITE		
21000-21450 kHz	AMATEUR AMATEURSATELLITE	AMATEUR AMATEURSATELLITE		
24890-24990 kHz	AMATEUR AMATEURSATELLITE	AMATEUR AMATEURSATELLITE		
28000-29700 kHz	AMATEUR AMATEURSATELLITE	AMATEUR AMATEURSATELLITE		
50.0000 52.0000 MHz	BROADCASTING 5.164 5.162A	BROADCASTING LAND MOBILE 5.164 Amateur LBN 4, LBN 6		Geographical sharing with wind profiler radars in the range 46-68 MHz
144-146 MHz	AMATEUR AMATEURSATELLITE	AMATEUR AMATEURSATELLITE	Amateur	
430-432 MHz	AMATEUR RADIOLOCATION 5.271 5.272 5.273 5.274 5.275 5.276 5.277	FIXED 5.276 MOBILE except aeronautical mobile AMATEUR RADIOLOCATION		
432-433.05 MHz	AMATEUR RADIOLOCATION Earth Exploration Satellite (active) 5.279A 5.138 5.271 5.272 5.273 5.274 5.275 5.276 5.277 5.280 5.281 5.282	FIXED MOBILE except aeronautical mobile AMATEUR RADIOLOCATION Earth Exploration Satellite (active) 5.279A 5.276 5.277		
433.05- 434.79 MHz	AMATEUR RADIOLOCATION Earth Exploration- Satellite (active) 5.279A 5.138 5.271 5.272 5.276 5.277 5.280 5.281	FIXED MOBILE except aeronautical mobile AMATEUR RADIOLOCATION Land Mobile Earth Exploration- Satellite (active) 5.279A 5.138 5.276	ISM SRD	
434.79-435 MHz	AMATEUR RADIOLOCATION Earth Exploration- Satellite (active) 5.279A 5.138 5.271 5.272 5.276 5.277 5.280 5.281 5.282	FIXED MOBILE except Aeronautical Mobile AMATEUR AMATEURSATELLITE RADIOLOCATION Earth Exploration- Satellite (active) 5.279A 5.276		Amateur Satellite Service restricted to 435-438 MHz.
435-438 MHz		FIXED AMATEUR AMATEURSATELLITE RADIOLOCATION		

RF Frequencies policies

		5.276		
438-440 MHz	AMATEUR RADIOLOCATION 5.271 5.273 5.274 5.275 5.276 5.277 5.283	FIXED MOBILE except aeronautical mobile AMATEUR RADIOLOCATION 5.276		
1240-1300 MHz	EARTH EXPLORATION SATELLITE (active) RADIOLOCATION SPACE RESEARCH (active) RADIONAVIGATIONSATELLITE (S/E)(S/S) 5.329 5.329A 5.328B Amateur 5.282 5.330 5.331 5.335A	RADIOLOCATION EARTH EXPLORATION SATELLITE (active) SPACE RESEARCH (active) RADIONAVIGATION RADIONAVIGATIONSATELLITE 5.329 5.329A 5.328B Amateur Amateur-Satellite 5.282 5.330 5.331 5.335A	DME Radio navigation Amateur	This band 1260-1300 MHz is proposed to be protected to distance measurement equipment (DME) Wind profiler radars between 1270 MHz and 1295 MHz
2300-2450 MHz	FIXED MOBILE 5.384A Amateur Radiolocation 5.150 5.282 5.395	FIXED MOBILE Amateur Radiolocation	IMT (2300-2400 MHz) Fixed links	The band 2300-2400 MHz identified for IMT (WRC07)
		FIXED MOBILE Amateur Amateur Satellite 5.150 5.282	FIXED Links Amateur SRDs RLAN AVI RFID WLAN ISM	The band 2400-2483.5 MHz is designated for ISM applications. Radio communications must accept any interference caused by ISM apparatus in this band.
5650-5725 MHz	RADIOLOCATION MOBILE except aeronautical mobile 5.450A 5.446A Amateur Space Research (deep space) 5.282 5.451 5.453 5.454 5.455	FIXED 5.453 MOBILE RADIOLOCATION Amateur 5.282 LBN1	Defense systems Wireless Access RLANs Shipborne and VTS Radar Amateur applications	ERC REC 70-03 Amateur Satellite Service (Earth to space), 5650-5670 MHz from RR 5.282.
5725-5830 MHz	FIXED-SATELLITE (E/S) RADIOLOCATION Amateur 5.150 5.451 5.455 5.456 5.453	FIXED 5.453 MOBILE FIXED-SATELLITE (E/S) RADIOLOCATION Amateur 5.150 LBN1	Amateur applications SRD ISM Radars BFWA	ERC REC 70-03 ISM 5725-5875 MHz RTTT 5805-5815 MHz SRDs 5725-5875 MHz
5830-5850 MHz	FIXED-SATELLITE (E/S) RADIOLOCATION Amateur Amateur-Satellite (S/E) 5.150 5.451 5.455 5.456 5.453	FIXED MOBILE FIXED-SATELLITE (E/S) RADIOLOCATION Amateur Amateur-Satellite (S/E) 5.150 5.453	Fixed links Amateur applications SRD ISM Radars	Amateur Satellite 5830-5850 MHz (S/E)
10.00-10.15 GHz	FIXED MOBILE RADIOLOCATION Amateur 5.479	FIXED MOBILE RADIOLOCATION Amateur 5.479	Fixed links SAB	
10.15-10.30 GHz			Fixed links FWA	ERC REC 12-05 for fixed service ERC REC 13-04 for FWA 10.15-10.30/10.5-10.65 GHz
10.30-10.45 GHz			Fixed links SAB	
10.45-10.50 GHz	RADIOLOCATION Amateur Amateur Satellite 5.481	FIXED RADIOLOCATION MOBILE Amateur Amateur Satellite	Fixed links SAB	ERC REC 12-05 for fixed service

Basics

24.00-24.05 GHz	AMATEUR AMATEURSATELLITE 5.150	AMATEUR AMATEURSATELLITE 5.150	Amateur	ISM 24-24.5 GHz
24.05-24.25 GHz	RADIOLOCATION Amateur Earth exploration Satellite (active) 5.150	RADIOLOCATION Amateur Earth exploration Satellite (active) Fixed Mobile 5.150	Amateur ISM SAB SRD Motion sensors	ERC REC 70-03 ISM 24-24.5 GHz
47.00-47.20 GHz	AMATEUR AMATEURSATELLITE	AMATEUR AMATEURSATELLITE	Amateur applications Amateur satellite applications	
48.20-48.54 GHz	FIXED FIXED-SATELLITE (E/S) 5.552 (S/E) 5.516B 5.554A 5.555B MOBILE	FIXED FIXED-SATELLITE (E/S) 5.552 (S/E) 5.516B 5.554A 5.555B MOBILE Amateur	Fixed satellite applications SAB	ERC REC 25-10 Feeder link band for 40GHz broadcasting satellites
76.00-77.50 GHz	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur-Satellite Space Research (S/E) 5.149	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur-Satellite Space Research (S/E) 5.149	Radio astronomy applications RTTT Amateur applications Amateur satellite applications Civil radiolocation	Spectral line and wide band continuum observations Road Transport and Traffic Telematics 76- 77 GHz Radar
77.50-78.00 GHz	AMATEUR AMATEUR SATELLITE Radio Astronomy Space Research (S/E) 5.149	AMATEUR AMATEUR SATELLITE Radio Astronomy Space Research (S/E) 5.149	Radio astronomy applications	Spectral line and wide band continuum observations
78.00-79.00 GHz	RADIOLOCATION Amateur Amateur Satellite Radio astronomy Space Research (S/E) 5.149 5.560	RADIOLOCATION Amateur Amateur-Satellite Radio astronomy Space Research (S/E) 5.149 5.560	Radio astronomy applications	Spectral line and wide band continuum observations
79.00-81.00 GHz	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur Satellite Space Research (S/E) 5.149	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur Satellite Space Research (S/E) 5.149	Radio astronomy applications	Spectral line and wide band continuum observations
122.25-123 GHz	FIXED INTER-SATELLITE MOBILE 5.558 Amateur 5.138	FIXED INTER-SATELLITE MOBILE 5.558 Amateur 5.138	Amateur applications Amateur satellite applications SRD	ERC REC 70-03
134-136 GHz	AMATEUR AMATEURSATELLITE Radio Astronomy	AMATEUR AMATEURSATELLITE Radio Astronomy	Amateur applications Amateur satellite applications	
136-141 GHz	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur satellite 5.149	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur satellite 5.149	Radio astronomy applications Amateur applications Amateur satellite applications	Spectral line and wide band continuum observations
241-248 GHz	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur-Satellite 5.138 5.149	RADIO ASTRONOMY RADIOLOCATION Amateur Amateur-Satellite 5.138 5.149	Amateur applications Amateur satellite applications	Spectral line and wide band continuum observations ERC REC 70-03
248-250 GHz	AMATEUR AMATEURSATELLITE	AMATEUR AMATEURSATELLITE		

RF modules

	Radio Astronomy 5.149	Radio Astronomy 5.149		
--	--------------------------	--------------------------	--	--

The yellow rows indicate the frequency band which we are going to use it.

From 433.05 to 434.79 MHz	AMATEUR RADIOLOCATION Earth Exploration- Satellite (active) 5.279A 5.138 5.271 5.272 5.276 5.277 5.280 5.281	FIXED MOBILE except aeronautical mobile AMATEUR RADIOLOCATION Land Mobile Earth Exploration- Satellite (active) 5.279A 5.138 5.276	ISM SRD
------------------------------	---	---	------------

16.13 RF modules

An RF Module is a (usually) small electronic circuit used to transmit, receive, or transceive radio waves on one of a number of carrier frequencies. RF Modules are widely used in consumer application such as garage door openers, wireless alarm systems, industrial remote controls, smart sensor applications, and wireless home automation systems. They are often used instead of infrared remote controls as they have the advantage of not requiring line-of-sight operation.

In this project we will use an RF module to send and receive message between two programmable microcontrollers with some condition related to the frequency, range and module speed. For this purpose there are several RF modules which may do this, bellows we will take a look on some of it:

16.13.1 STD-402

The transceiver to be used is **MB-STD-RS232**. It is a bi-directional semi-duplex radio modem having RS232 serial interface. It uses CIRCUIT DESIGN's standard 434 MHz FM Narrow Band transceiver module **STD-402** transceiver for RF part. This transceiver was selected because of its frequency of 434 MHz For this frequency in Germany there is no extra permission necessary. Another reason is that this transceiver is a cheap one.

The **STD-402** transceiver is an UHF Narrow Band Multi channel Transceiver. The UHF FM-Narrow Band semi-duplex radio data module **STD-402** equipped PLL controller in its robust metal housing. Unlike other transceivers, the **STD-402** is ready to transmit RF data without complicated controller board. The compact size and low power consumption of the **STD-402** make it ideal for battery operated applications where its interference rejection and practical distance range are much better than similar RF modules based on Wide Band SAW – resonator frequency devices.

Most of RF settings are done by internal microcomputer, which allows the user to manipulate the module without professional knowledge of RF circuit.

16.13.1.1 Special for *MB-STD-RS232*



Basics

Figure 5.3.1: MB-STD-RS232 – CIRCUIT DESIGN

The RF part complies with the European radio, EMC and safety requirements and has been notified in major European countries under the R & TTE directive. The **MB-STD-RS232** provides long range data link at low/medium data rate for various industrial telemetry and data transfer applications. Also this board can be used as a test board of the **STD-402 TR**.

Features

- CE compliance **STD-402** 434 MHz RF module on the board.
- RS232 interface with D-sub 9pins connector or Modular 6pin jack.
- Fixed frequency / Auto frequency setting selectable.
- Cross / Straight cable selection SW.

Applications

- Serial data transmission (RS232C communication)
- Telemeter (FA line, Sensor information)
- Wireless connection between PC and peripheral RS232 equipment

General Description

MB-STD-RS232 is designed to make it possible for the user to connect between RS232 equipments with the radio. **STD-402** 434 MHz narrow band radio module that complies with EN300220 is equipped on the board. 64 channels are pre-programmed in the module.

There are two frequency setting are available. **In fixed (manual) setting**, RF channel can be set on board switches. **In auto setting**, RF channel is set to vacant channel automatically.

Operation mode and communication set up (Ack, parity, data rate) can be selected by on board dip-switch. The **operation mode 1** is designed for two-way communication and the **operation mode 2** is designed for one-way communication (TX -> RX).

1:N communication is possible by using unique module ID number that designated to each RF module.

Specification

RF parameter

Communication mode	Half-Duplex
Frequency range	433.200 to 434.775 MHz
CH step	25 kHz
Number of CH	64 CH
CH setting	Fix / Auto (8Gr*8ch)
Modulation data speed	9600bps
Modulation	2FSK
Emission class	F1D
Transmission power	10 mW

Serial Interface

Interface	RS-232C
-----------	---------

RF modules

Data format	Asynchronous communication (UART)
Data speed of RS	1200/2400/4800/9600 bps
Flow control	RS / CS hardware control
Buffer	Transmission 2kB, Reception 2KB
Interface connector	D-Sub 9P / Modular 6P

Other

Switches	Power, Frequency, Operation Mode, Cable (Cross/Straight)
LED indication	TX, RX, RSSI, LD, LE
Dimension	85*53*15mm
Supply Voltage	4.0 to 9V DC.

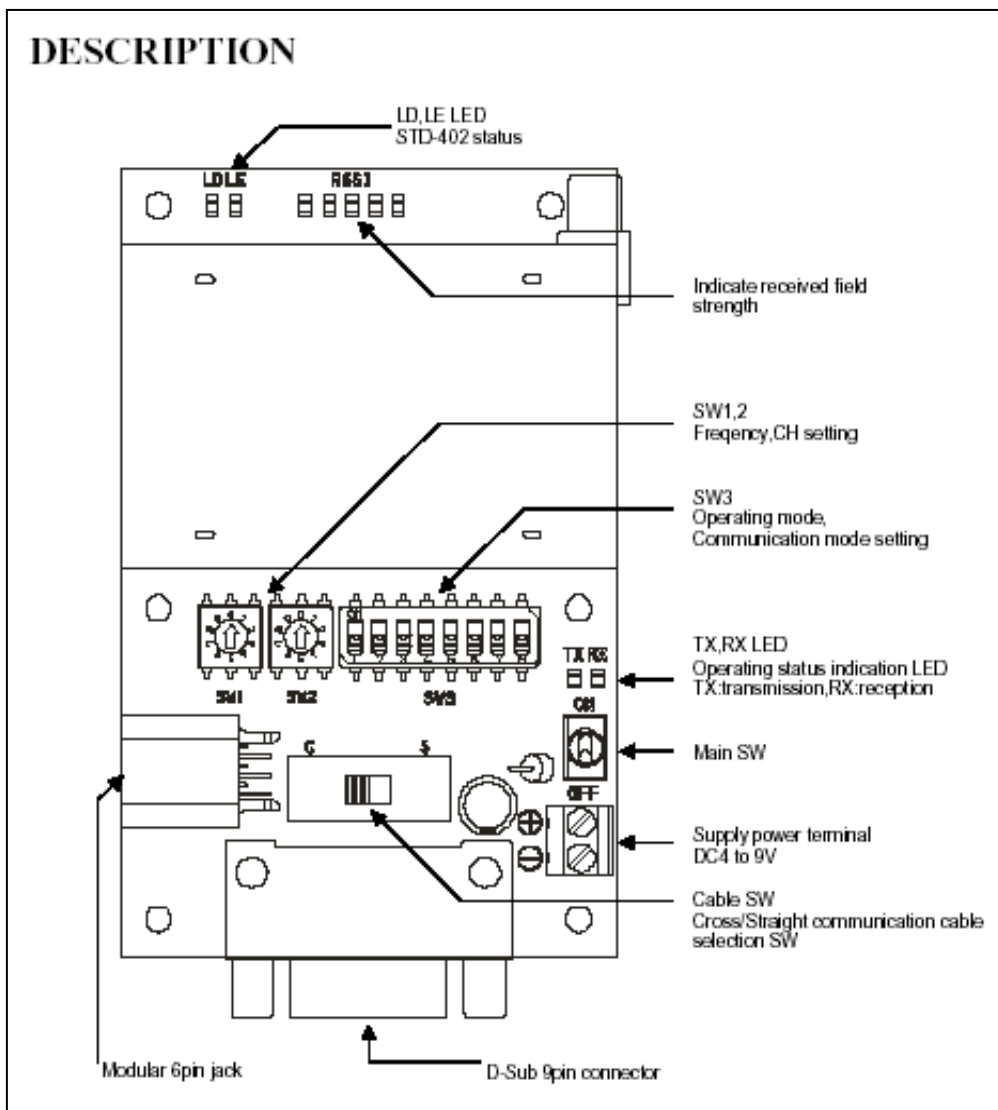


Figure 5.3.2: MB-STD-RS232 – CIRCUIT DESIGN

For more details about this board, refer to Annex A.

16.13.1.2 Special for STD-402 (Transceiver)

Basics

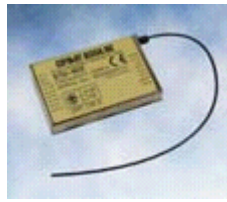


Figure 6.6: STD-402 Transceiver – *CIRCUIT DESIGN*

The **STD-402** transceiver is an UHF Narrow Band Multi channel Transceiver.

The UHF FM-Narrow Band semi-duplex radio data module **STD-402** equipped PLL controller in its robust metal housing. Unlike other transceiver, the **STD-402** is ready to transmit RF data without complicated controller board. The compact size and low power consumption of the **STD-402** make it ideal for battery operated applications where its interference rejection and practical distance range are much better than similar RF modules based on Wide Band SAW – resonator frequency devices.

Most of RF settings are done by internal microcomputer, which allows the user to manipulate the module without professional knowledge of RF circuit.

Features

- European EN300 200 standard compliance.
- High technology into compact module for easy operation.
- Low voltage operation from 3.6 V DC.
- Low current consumption, ideal for battery operated applications.
- 9600bps data rate.
- Carrier sense output for Multi-Channel access operation.

Application

- Remote control system.
- Security systems.
- Bi-directional communication systems.
- Telemetry systems
- Handy terminal.

STD-402 characteristics

❖ Common

Communication form	Semi-duplex
Frequency range	433.200 MHz to 433.775 MHz
Channel step	25 KHz.
Baud rate	9600bps max.
Supply voltage	3.6 – 12 V DC (Direct Mode).
Dimensions	53 × 35 × 12 mm.

RF modules

❖ Transmitter

RF output power	9 mW \pm 1mW.
Data input level	3.6 – 12V (Direct Mode).
Input signal	Digital
Spurious emission	< -60 dBm (< 1 GHz).
Supply current	36 mA.

❖ Receiver

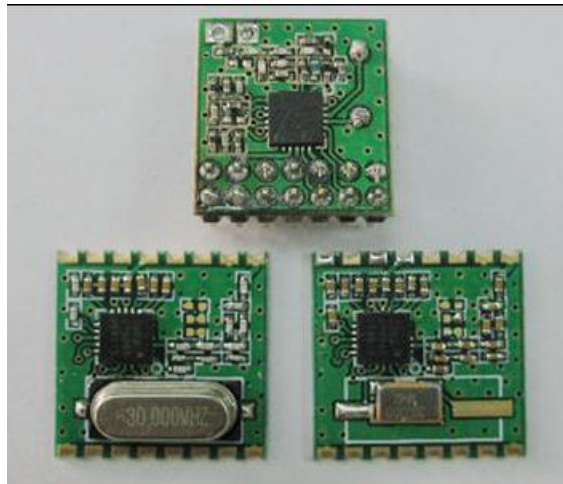
Receiver type	Double super heterodyne PLL synthesizer.
Selectivity	\pm 4 kHz at -6dB point.
Data output	Digital.

- **The STD-402 transceiver has 3 mode operation guides:**

1. **Direct Mode Operation Guide (For more details about this mode, refer to Annex B)**
2. **Auto Mode Operation Guide. (For more details about this mode, refer to Annex C)**
3. **Auto Mode Operation Guide for CPU interface. (For more details about this mode, refer to Annex D)**

Or the MB-STD-RS232 equips STD-402 transceiver module and performs packet communication using CPU interface mode of the transceiver.

16.13.2 RFM42B-RFM31B 433MHz



Features:

- 433/868/915MHz ISM bands Frequency range:
- Low Power Consumption
- Data Rate = 0.123 to 256 kbps
- FSK, GFSK, and OOK modulation
- Power Supply = 1.8 to 3.6 V
- Ultra low power shutdown mode
- Wake-up timer

Basics

- TX 64 byte FIFO
- Low battery detector
- Temperature sensor and 8-bit ADC
- -40 to +85 °C temperature range
- Integrated voltage regulators
- Frequency hopping capability
- On-chip crystal tuning
- 14-PIN DIP & 16-PIN SMD package
- Low cost
- Power-on-reset (POR)

Special for RFM42B (Transmitter):

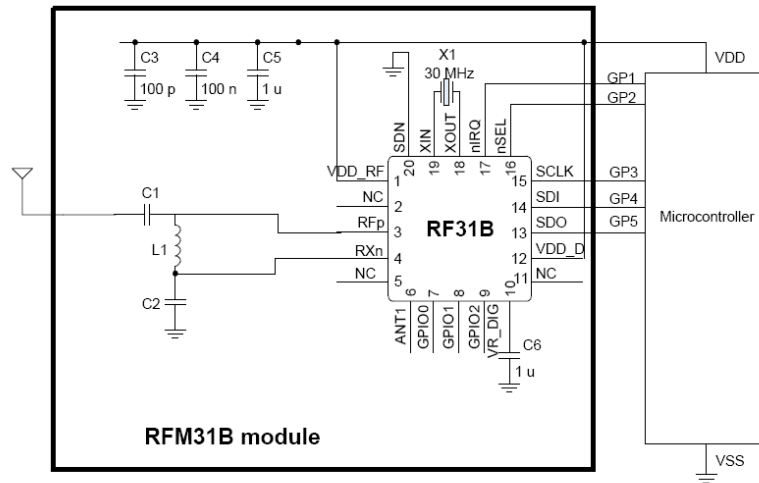
- Output Power Range
 - +1 to +20dBm (RFM42B)
 - 8 to +13dBm (RFM43B)
- Integrated 32 kHz RC or 32 kHz XTAL
- Configurable packet handler

Special for RFM31B (Receiver):

- Sensitivity = -121 dBm
- Digital RSSI
- Auto-frequency calibration (AFC)
- Clear channel RX BW 2.6–620 kHz
- Programmable assessment
- Programmable packet handler
- Programmable GPIOs
- Embedded antenna diversity algorithm
- Configurable packet handler
- Preamble detector
- RX 64 byte FIFO

Application example:

RF modules



16.13.3 BOWITZ W.T.

من كتاب الإرسال اللاسلكي و البث

الكتاب الثامن من موسوعة عالم الالكترونيات للمهندس أمين فهمي

دار الراتب الجامعية

الباب الثامن:

أجهزة الارسال و الاستقبال التجارية

إبتداء من صفحة: 165

16.13.4 Comparison between modules

module	Frequency	Range (m)	Speed (bps)	other
SHY-J6122TR	300 – 450 MHz			Made In chine Available in Lebanon
Rx Tx 315Mhy	315 MHz 433.92 MHz	> 500 m	< 10Kbps	Made In chine Available in Lebanon
RFM12 433MHz	433 MHz		> 115.2 Kbps	Programmable TxRx bandwidth SPI interface Made In chine Available in Lebanon
RFM12 915 MHz	915 MHz			
STD-402	434 MHz	500 m	9600 bps	Serial com. Need extra circuit (max232) Not available anymore

Basics

BOWITZ W.T.				Full W.T. project, We should build it by ourselves using the BOWITZ open source information
-------------	--	--	--	---

17 Specification

17.1 System Requirements

[SysReq 1] The system shall be a demonstration platform for customers. The customers can then specify their individual requirements. Afterwards the IAP ECS platform shall be migrated in every customer project to the specific needs

[SysReq 2] Our goal is to design an Emergency communication system (voice and information) for the red halfmoon, Red Cross or police to stay on touch in emergency situations.

17.2 Hardware Requirements

[HWReq1] The system shall use a SDR (software defined radio) with RF transmission technology.

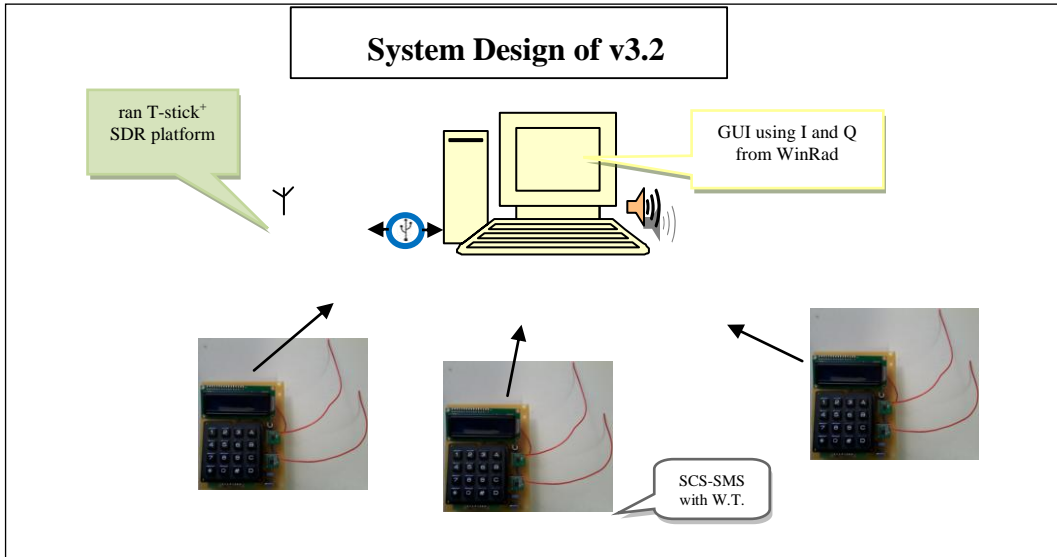
[HWReq 2] The sending and receiving HW shall be a cheap off-the-shell system so that the project can be finished in December 2013.

17.3 Software Requirements

tbd

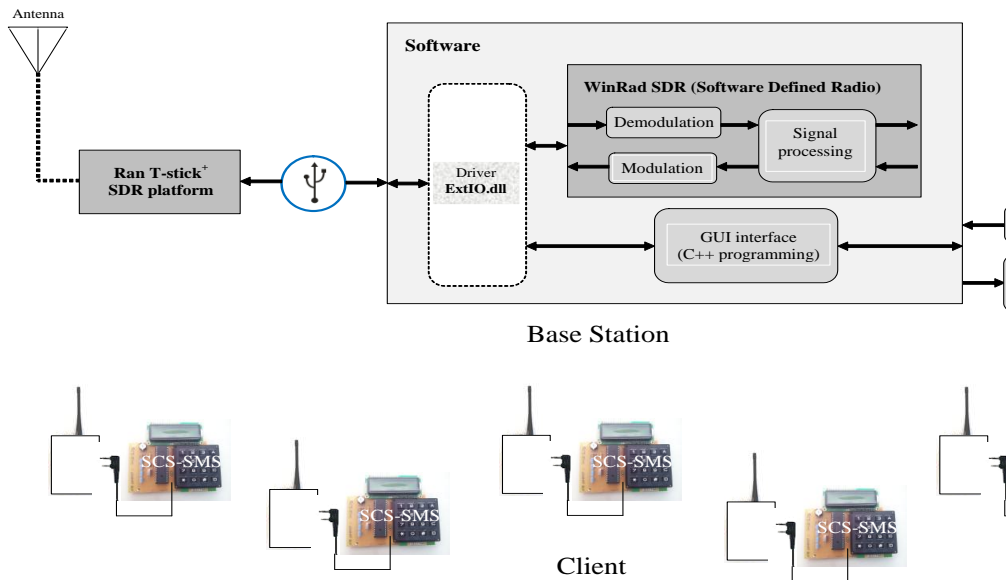
18 System Design

18.1 System Overview



18.2 Central Station

18.2.1 Architecture



18.2.2 SDR development side

In our project, we need the Software Defined Radio code which is included in **HDSDR** software. But as we know the **HDSDR** software is not open-source software while **WinRad** is. Then we have two potential choices to do this step:

System Design

For the HSDSR: we can change and develop the ExtIO.dll file of this program to input from my STD hardware source and to output on my GUI. In this case the HSDSR software will work on the background of our GUI software.

This choice means that we should use the HSDSR software in the two communication sides. This means also that we should use PC on the two sides again.

For the WinRad: as we say before WinRad is Open source then we can use its code. We should read its code to know where is the SDR code to copy it to our GUI.

This choice has the following problem: with WinRad we can only receive while we need to send and receive. This means that we should develop the code such that also sending is possible.

18.2.3 Graphical User Interface

The Graphical User Interface is the interaction interface between the user and the system. The goal of this interface is to monitor the location status: road status, problems on road, hospital status, and also it will have a messaging box to write notes for each other. Also the user interface should have the HSDSR option (change frequencies, send/receive, volume up/down) with an extra button to open the HSDSR software when user want.



Mobile Stations



18.3 Mobile Stations

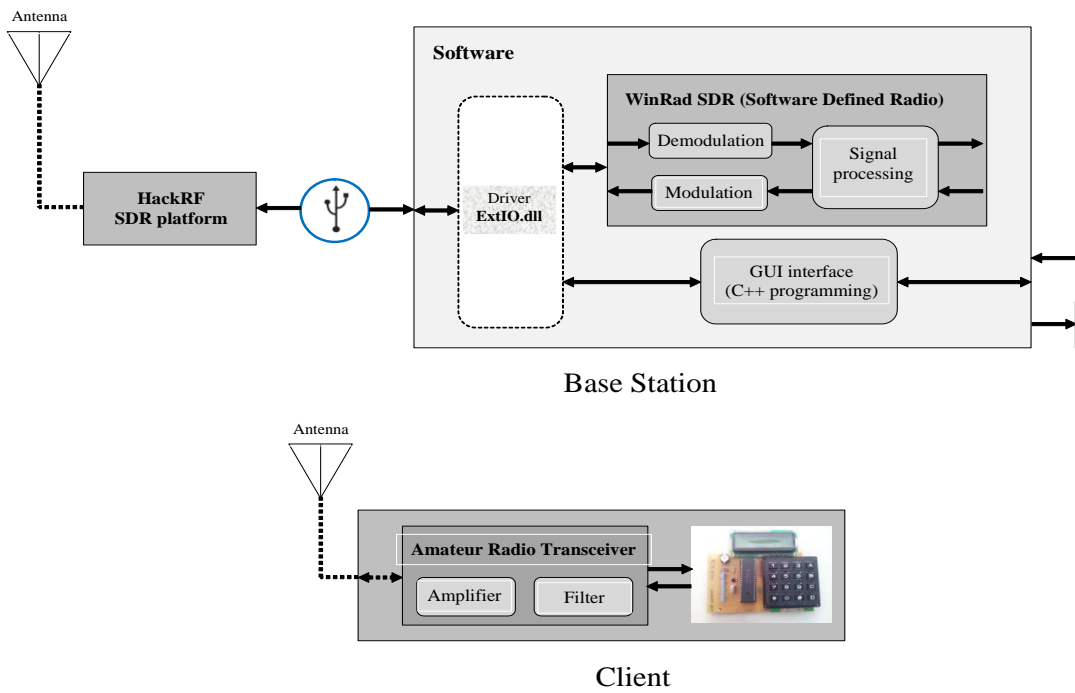
In the client side we can use the hardware of SCS-SMS project instead of PC but we should first add an RF transceiver to it with doing some modification on it



The main modifications are:

- Adding the RF hardware (amplifier, filter, antenna)
- Adding the A/D and D/A converter
- Put the SDR code on its processor

System Design

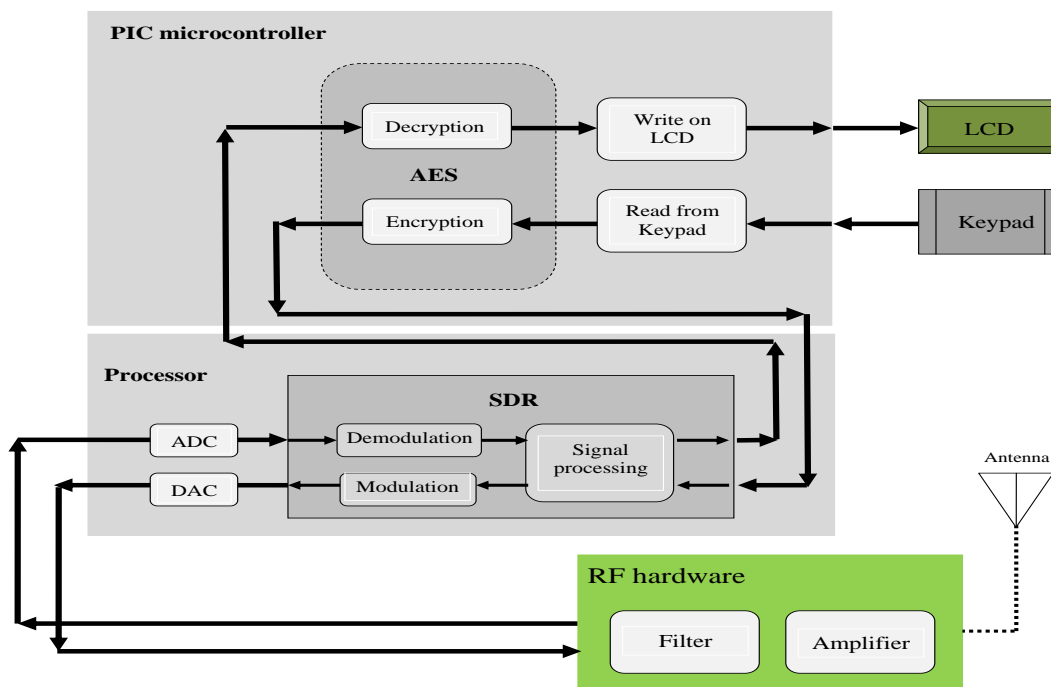


In this version we need to change on the hardware with the software of the SCS-SMS project to be able to use it in our project. The basic changes are:

- Adding the RF hardware (amplifier, filter, antenna)
- Adding the A/D and D/A converter
- Put the SDR (HSDSDR) on a second processor

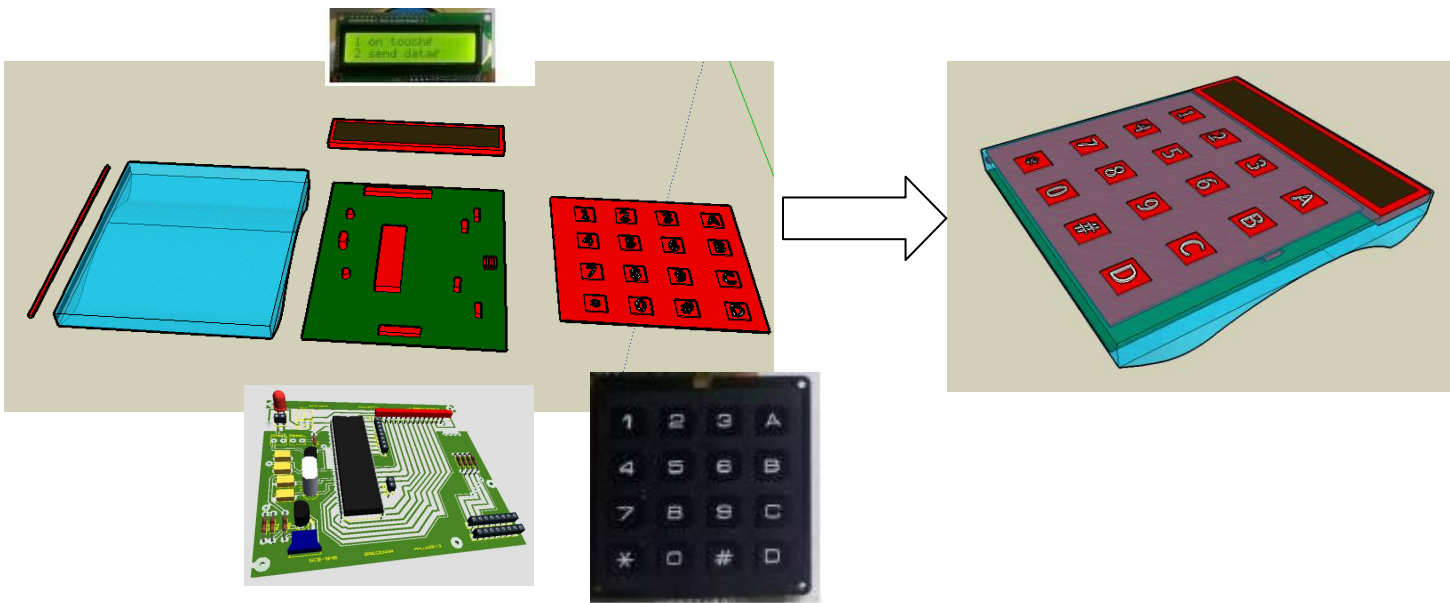
The new block diagram will be:

SCS-SMS after changes to GIS-STD

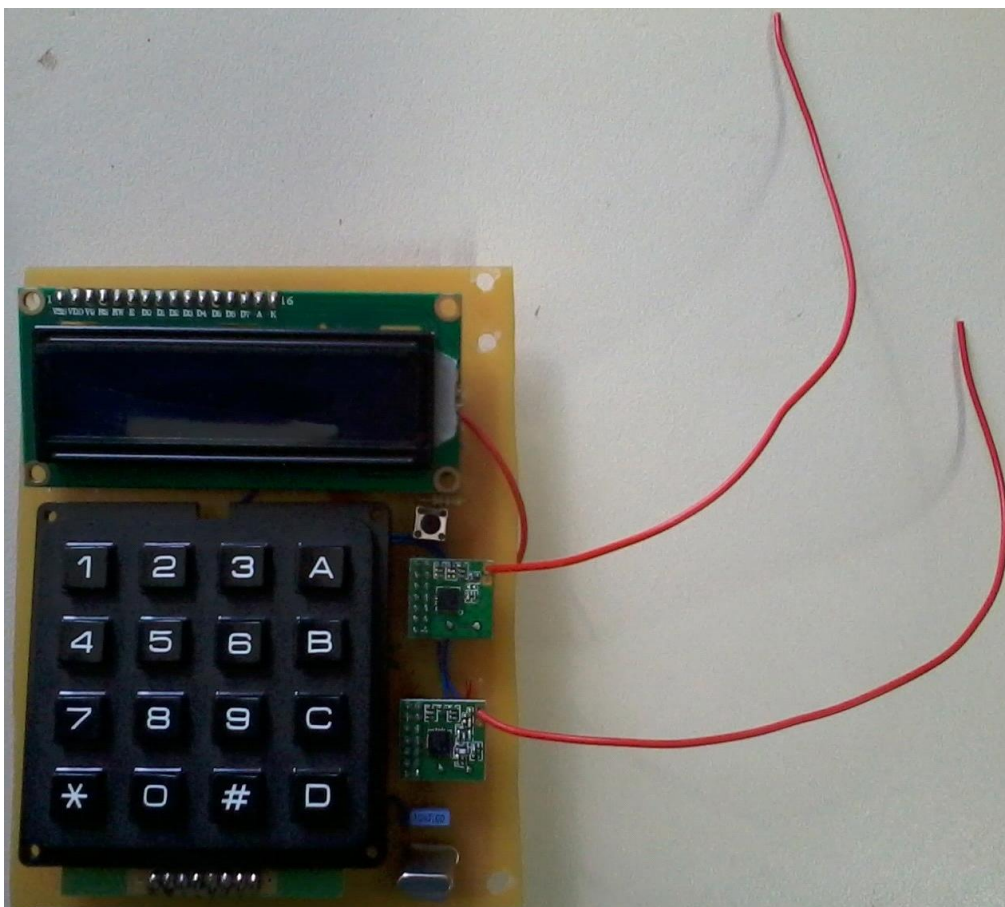


19 Mechanics

19.1 Mechanical Design



19.2 Prototype without cover



20 SCS-SMS

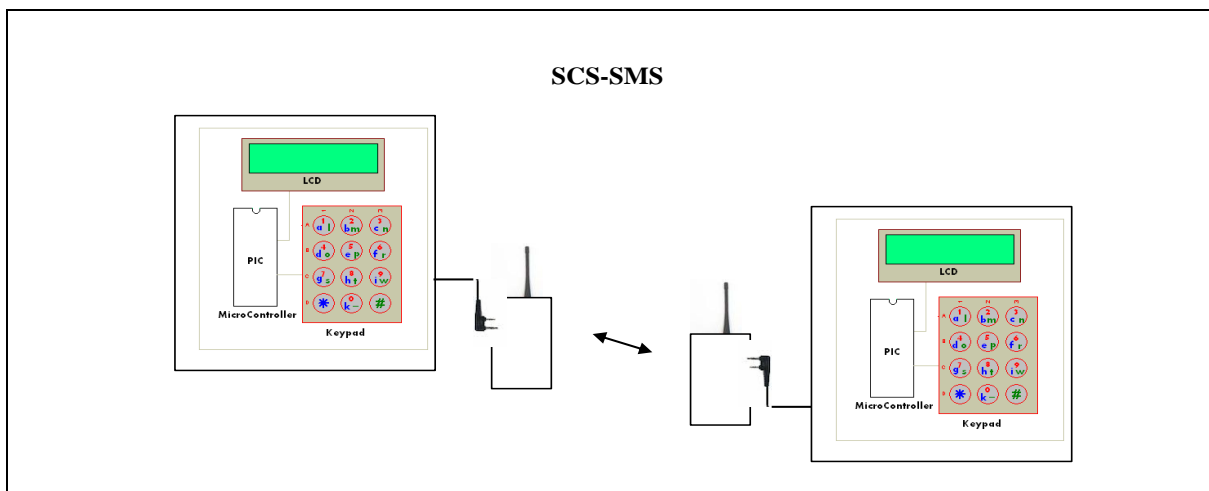
Secured communication System

20.1 Abstract of SCS-SMS

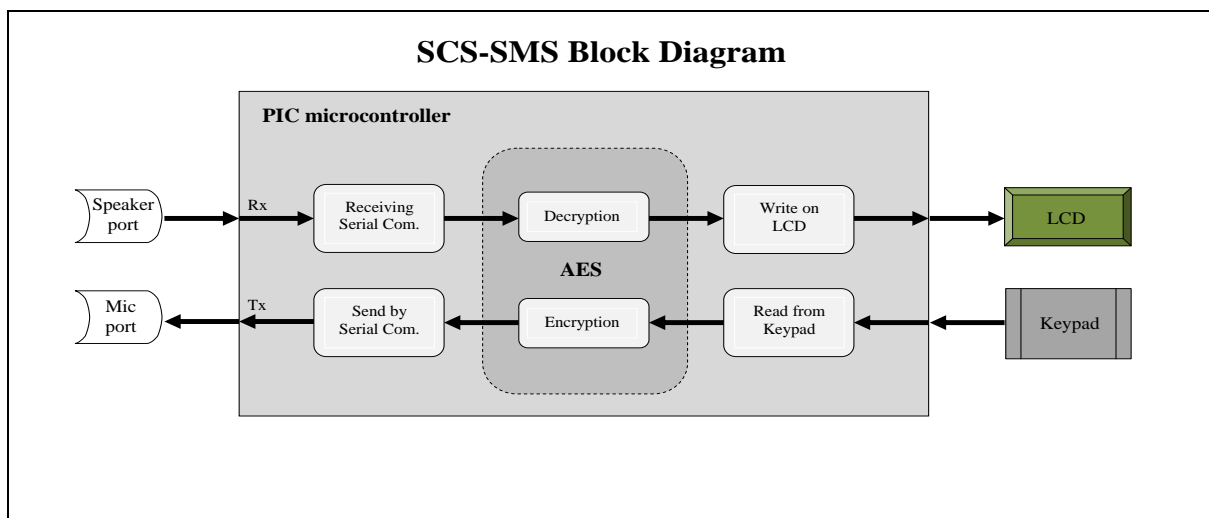
The goal of this project is to sending SMS securely on several way of communication (i. e.: telephone, mobile, RF transceiver...)

20.2 System design

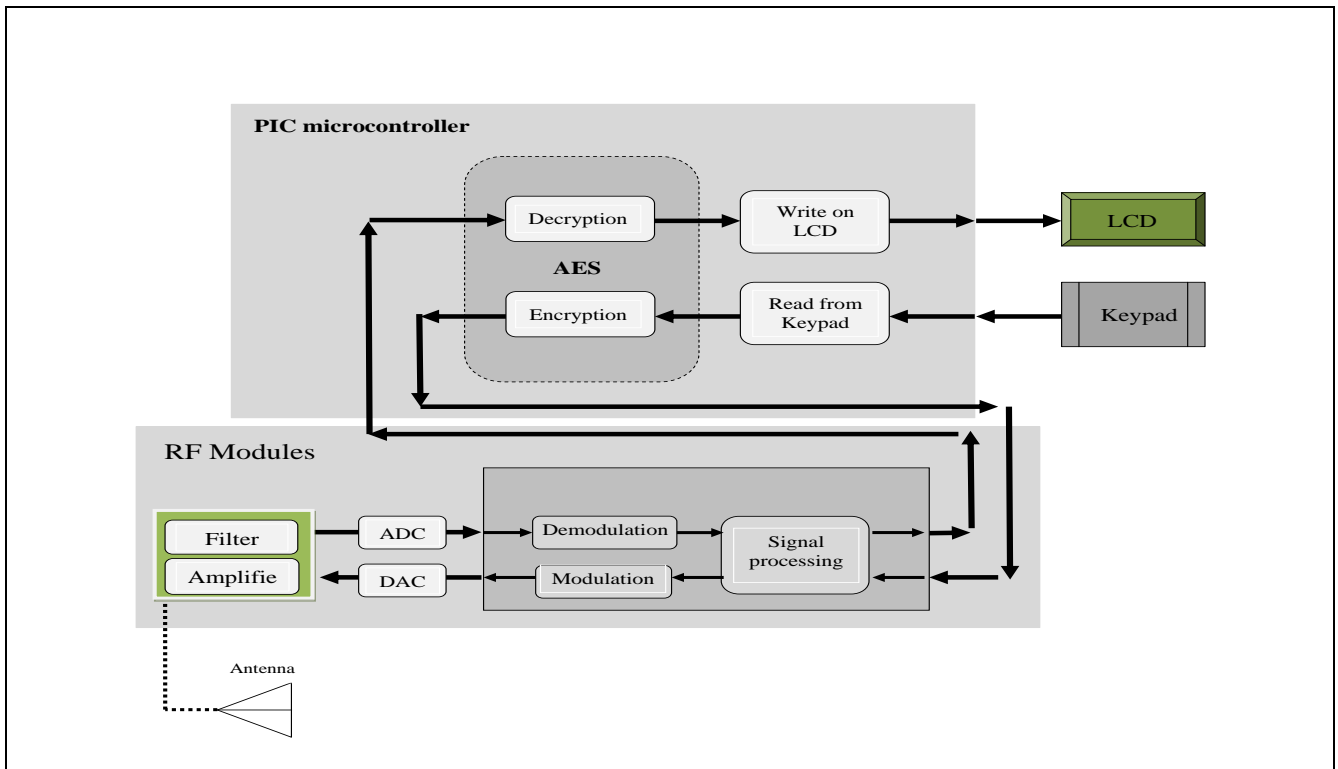
System plan:



Block diagram:



New:



20.3 Architectures

The architectures of this project is contain as a basic component a PIC microcontroller with a 4x4 keypad with a LCD and with some other electronic components

Equipment used:

PIC microcontroller	18f4550	1
LCD	2x16 (line X character)	1
Keypad	4x4	1
Resistor	4.7 kΩ	1
Pot resistor	10 kΩ max	1
Capacitor	1μF	4
Crystal	48000 MHz	1
Voltage regulator	78L05	1
Push button		1

PIC microcontroller:

The microcontroller needs a programmer with a compiler to write your program in it. The most widely compiler is the MPLAB software which we will use it in this project, the MPLAB

Architectures

already have an Assembly compiler for the PIC but if you want to write your program in C language then you need a C compiler for your PIC. On our project the compiler we use it is MCC18 which is for the 18 PIC series. We will discuss the c program in the next chapter

LCD:

2x16 LCD is used in this project to display character on 2 lines 16 characters Liquid Crystal Display. This LCD has 14 input pins to write on and to control LCD

LCD PINs description:

Pin	Symbol	Description
1	Vss	Ground
2	Vcc	+5 V power supply
3	Vee	Power supply to control contrast
4	RS	RS=0 to select command register RS=1 to select data register
5	RW	RW=0 for write RW=1 for read
6	E	Enable
7	DB0	The 8-bit data bus
8	Db1	The 8-bit data bus
9	DB2	The 8-bit data bus
10	DB3	The 8-bit data bus
11	DB4	The 8-bit data bus
12	DB5	The 8-bit data bus
13	DB6	The 8-bit data bus
14	DB7	The 8-bit data bus

On each time you need to write on LCD or to send a command to LCD you should set the Enable pin E before then disable it after sending.

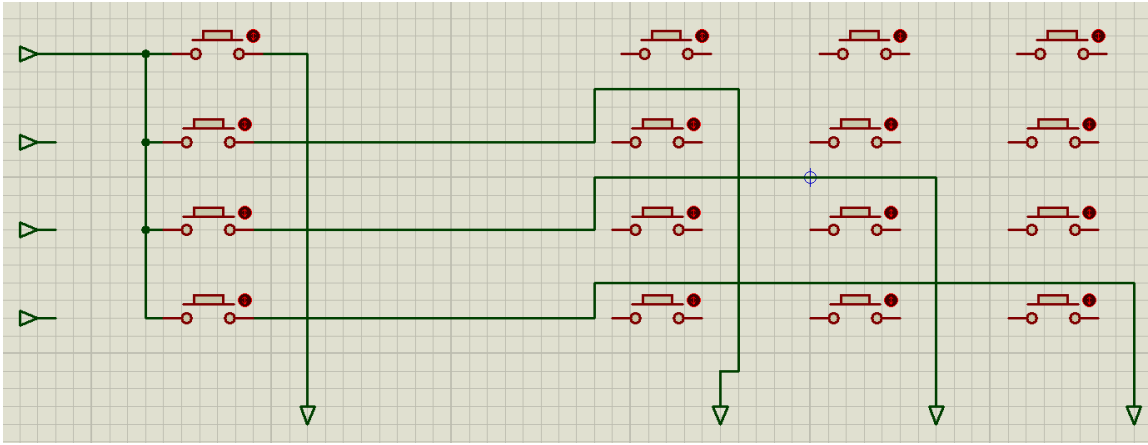
Also you should do a harmony between the speed of the LCD receiving with the speed of PIC data sending, and this will be do it by the PIC software using a **delays** function.

Finally, a Pot resistor should be connecting to the R pin of the LCD to control the LCD light brightness.

Keypad:

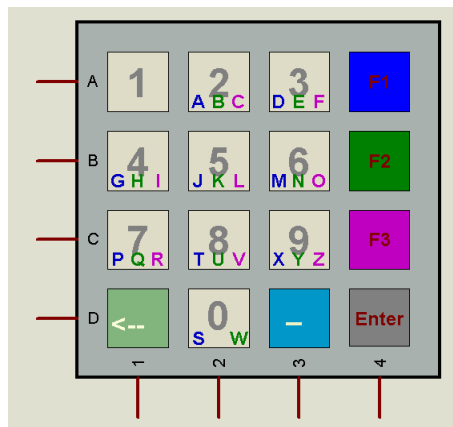
The keypad we use is 4x4 keypad with 8 I/O pins; the 4x4 keypad design show in the picture bellow:

SCS-SMS



As we see in the picture above, we have 4 input pins and 4 output pins. We should set each input one by one and in each time we should read the output on the 4 output pins to know which key was pressed.

In the software program you should specify the meaning of each key. In our project, we specify our key as follow:



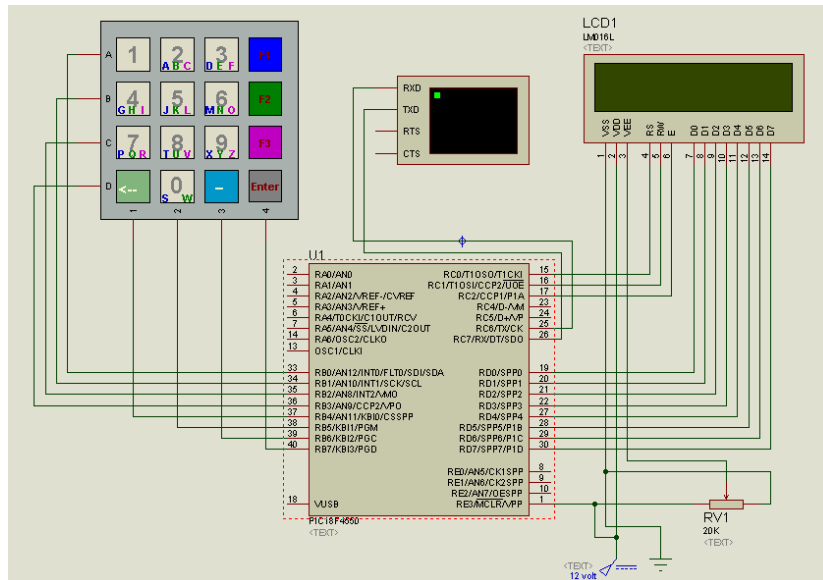
In the normal case the keypad will write numbers if F1 (Function1 key) pressed then the key will write the **Blue** character (A, D, G, J, M, P, T, X, S). IF F2 (Function2 key) pressed then the key will write the **Green** character (B, E, H, K, N, Q, U, Y, W). If the F3 (Function3 key) pressed then the key will write the **Red** character (C, F, I, L, O, R, V, Z). There are also three other key which are: (**Enter**) to send data, (**_**) to write a space, (**←**) to delete the last character.

System Design:

The design of the system was developed on Proteus to simulate and test before the built of the real system.

The figure bellow shows us the system design:

Architectures

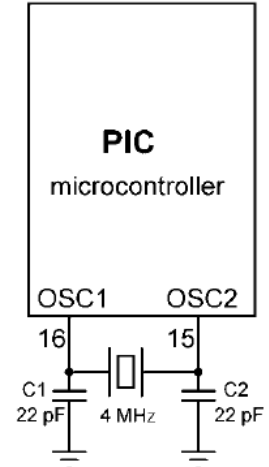


As we see, our design is too simple. It has three base components: PIC, keypad, and a LCD. The Keypad is connected to port B of the PIC, and LCD connected to the port C and D. port D connected to LCD Data register and port C (first three pins only) connected to LCD Controller. The virtual machine is added for testing purpose.

The design is enough in Proteus simulation but it is not in the real hardware while we should add some components for: timing resonator, voltage regulator, and reset button.

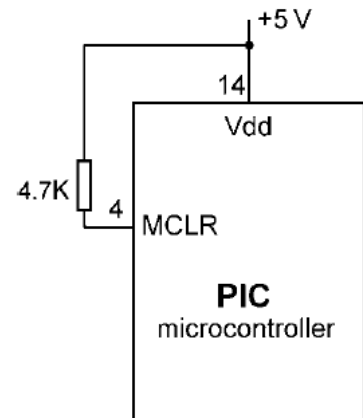
Timing components resonator:

A PIC microcontroller requires an external clock circuit (some PIC microcontrollers have built-in clock circuits) to function accurately. For accurate timing applications, the clock circuitry consists of a crystal, and two small capacitors. Figure bellow shows the circuit diagram of a PIC microcontroller with a 4-MHz crystal clock circuit. The crystal and the capacitors are connected to the OSC1 and OSC2 inputs of the microcontroller.

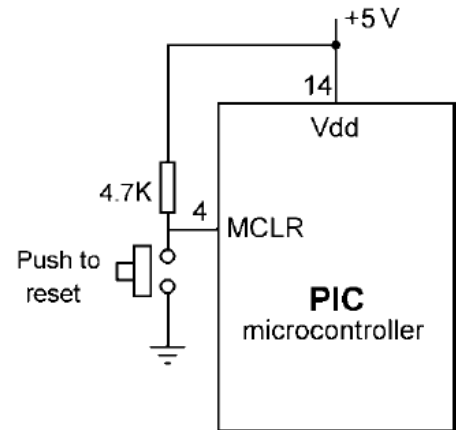


Power source and Reset circuit

A PIC microcontroller starts executing the user program from address 0 of the program memory when power is applied to the chip. As shown in Figure bellow, the reset input (MCLR) of the microcontroller is usually connected to the 5V supply voltage through a 4.7K resistor.



There are many applications where the user may want to force reset action e.g. by pressing an external button so that the program re-starts to execute from the beginning. External reset is very useful during microcontroller-based system development and testing. Figure below shows how an external reset button can be connected to a PIC microcontroller. Normally the MCLR input is at logic 1, and goes to logic 0 which resets the microcontroller when the reset button is pressed. The microcontroller goes back to the normal operating mode when the button is released.

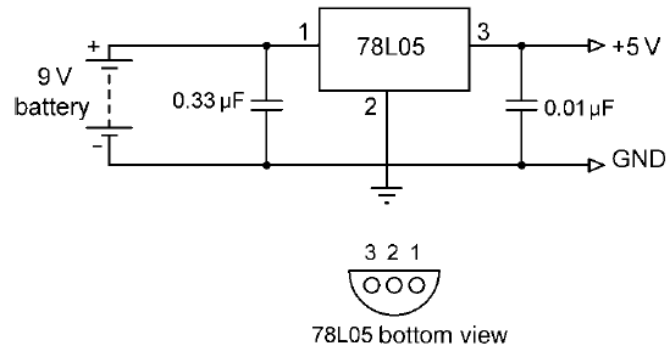


Power supply

Every electronic circuit requires a power supply to operate. The required power can either be provided from a battery, or the mains voltage can be used and then reduced to the required level before it is used in the circuit (e.g. a mains adaptor). In this section, we shall look at the design of a power supply circuit to power our PIC microcontroller circuits.

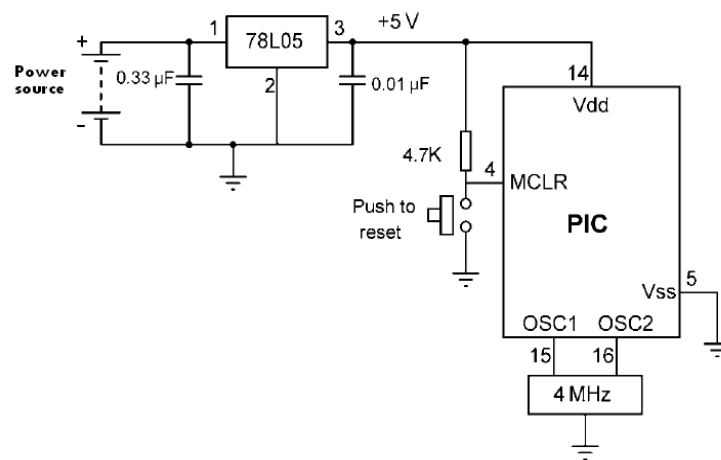
PIC microcontrollers can operate from a power supply voltage in the range 2 to 6V. The standard power supply voltage in digital electronic circuits is 5 V and this is the voltage with which the PIC microcontrollers are mostly operated. Unfortunately, it is not possible to obtain 5 V using standard alkaline batteries only. The nearest we can get is by using three batteries, which gives 4.5 V and this is not enough to power standard logic circuits. The simplest solution to drop the voltage from 9 to 5 V is by using a potential divider circuit using two resistors. Although a potential divider circuit is simple, it has the major disadvantage that the voltage at the output depends on the current drawn from the circuit. As a result of this, the output voltage will change as we add or remove components from our circuit. Also, the output voltage falls as the battery is used. A voltage regulator circuit is needed to convert the 9 V battery voltage into 5V, independent of the current drawn from the supply. A basic voltage regulator circuit consists of a regulator integrated circuit and filter capacitors. Figure below shows a low-cost voltage regulator circuit using the **78L05**-type voltage regulator IC, and two filter capacitors. **78L05** is a 3-pin IC with a maximum current capacity of 100 mA.

Architectures



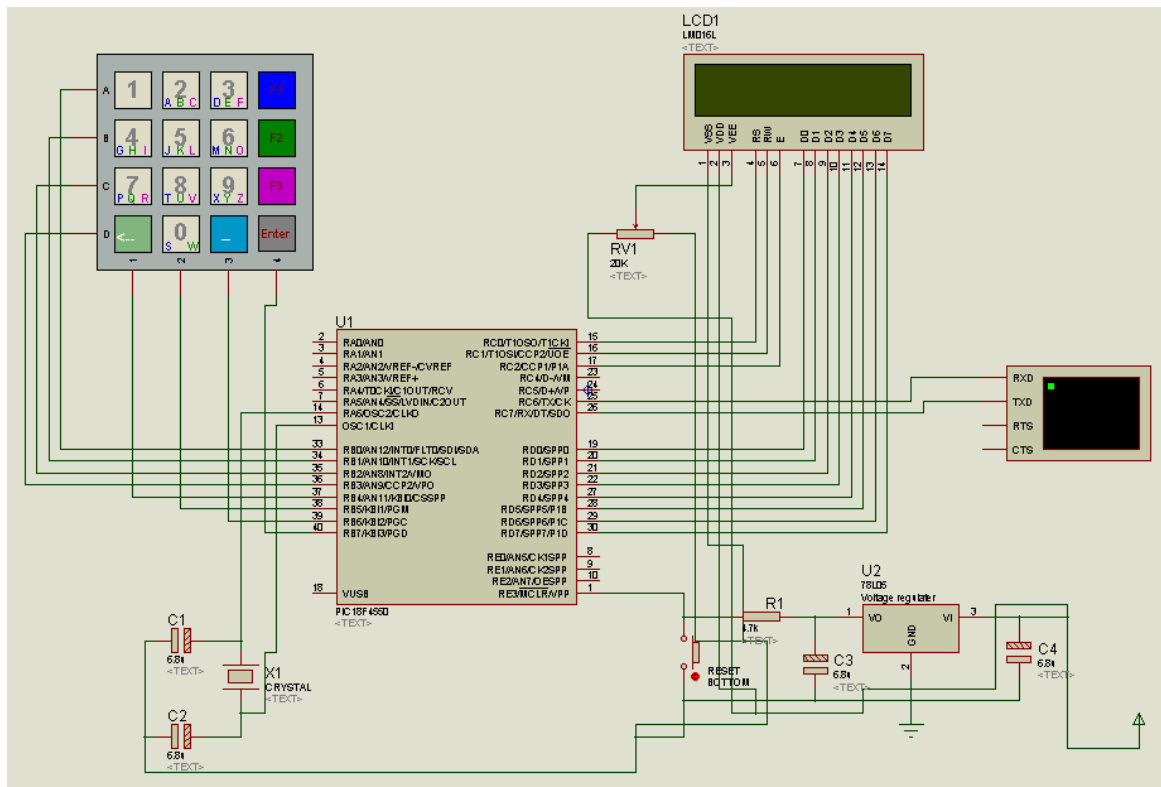
One of the pins of **78L05** is connected to the +V terminal of the battery in parallel with a 0.33-μF capacitor. One of the pins is connected to the -V terminal of the battery. The third pin provides the +5 V output and a 0.01-μF capacitor should be used in parallel with this pin. In applications where a larger current is required, the **7805** regulator IC can be used. This is pin compatible with the low-power **78L05** and it has a maximum current capacity of 1 A. **78L05** should be used with a suitable heat-sink in applications drawing more than a few hundreds of mill-amperes.

The complete circuit diagram of our PIC based basic system, together with the power supply, is shown in Figure bellow. The circuit is now fully functional, what is required now is to write our program and load it into the program memory of the microcontroller.

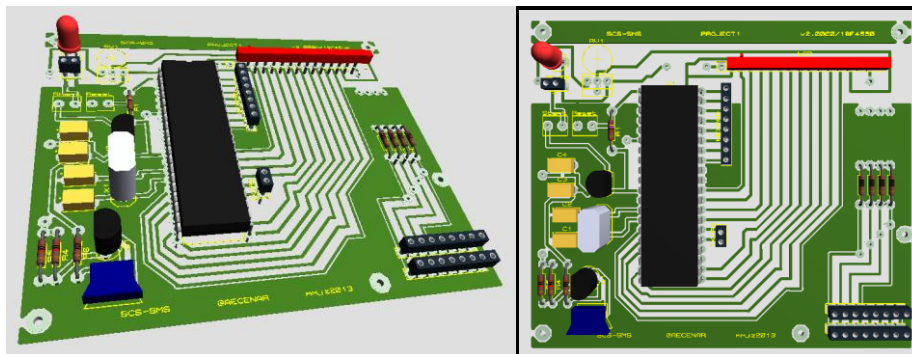


The final circuit is show in the figure bellow:

SCS-SMS



The final circuit on PCB board is shown in the figure below (see layout on Appendix B):



20.4 PIC software

As we see in the section before, we will write our program using C language. So, we should install the MCC18 C compiler for our MPLAB software.

In our program, some (.h) libraries which will be used in the program should be included at the head of our program

The (.h) libraries we include are:

P18f4550.h, stdio.h, delays.h, string.h

And this is done by this C code:

```
#include<p18f4550.h>
#include<stdio.h>
```

PIC software

```
#include<delays.h>
#include<string.h>
```

Now, let's start with our program. The program is containing some special functions a side to the main function and the initialization functions.

The special functions are:

```
void INIT_PORT(void) ;
void InitSerial(void) ;
char Encryption(char PC) ;
char Decryption(char CC) ;
void SendToSerial(char m) ;
void SendStringToSerial(char msg[]) ;
char ReceiveFromSerial(void) ;
void LCD_CMD(unsigned int value) ;
void WRITE_CHAR_LCD(unsigned char value) ;
void WRITE_STRING_LCD(char value[]) ;
char KeyPad(void) ;
void on_touch(void) ;
void send_SMS(void) ;
void send_data(void) ;
void Menu(void) ;
```

INIT_PORT function

The goal of this function is to initialize the input output port/pin for the microcontroller by setting the **TRIS** for each port or pin used. This function also clears all port from any past setting. This function has no return no calling input.

C code:

```
void INIT_PORT(void)
{
    ADCON1=0x0E; //for using analog port RA0
    TRISAbits.TRISA0 = 1; //input pin
    TRISAbits.TRISA1 = 1; //input pin
    PORTA =0x00;
    TRISB =0b11110000;
    PORTB =0x00;
    TRISCbits.TRISC0 = 0; //output pin for LCD RS
    TRISCbits.TRISC1 = 0; //output pin for LCD RW
    TRISCbits.TRISC2 = 0; //output pin for LCD E
    TRISCbits.TRISC6 = 0; //output pin
```

SCS-SMS

```
TRISbits.TRISC7 = 1; //input pin
PORTC =0x00;
TRISD =0x00; //output port 00000000
PORTD =0x00;
}
```

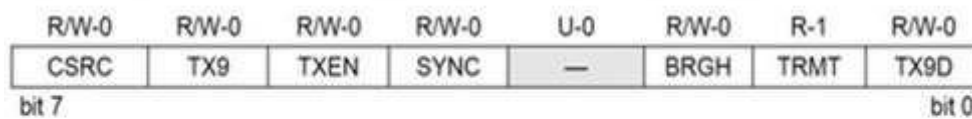
InitSerial function

To communicate with external components such as computers or microcontrollers, the PIC microcontroller uses a component called **USART - Universal Synchronous Asynchronous Receiver Transmitter**. This component can be configured as:

- A Full-Duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers
- A Half-Duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

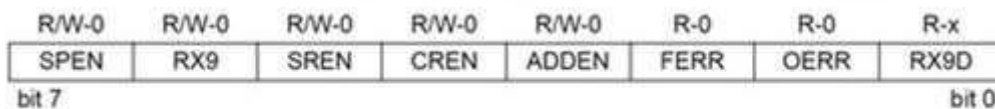
To enable the serial communication with PIC microcontroller we must set different parameters within two registers:

1. TXSTA - Transmit Status and Control Register



MicrocontrollerBoard.com

2. RCSTA - Receive Status and Control Register



MicrocontrollerBoard.com

The send information will be stored inside **TXREG** register, which acts as a temporary buffer storage of information prior to transmission. While the receive information will be store in the **RCSTA** register, which acts as a temporary buffer storage.

Each transmission is transmitted in the particular rate (BAUD). The baud rate is measured in units of **bps** (bit per second). This is done by setting the system clock to the value needed. To do so, we need to “write” a hexadecimal number to the **SPBRG** register. The value written to the **SPBRG** register set the clock cycle to the value we want for the BAUD rate.

PIC software

The size of **SPBRG** register is 8-bit. In asynchronous mode, the baud rate of transmission of the information can be set to high speed or to low speed. The rate selection, as already seen, is made by the BRGH bit in TXSTA register:

1 = High speed 0 = Low speed

For each baud rate we need to calculate the value being placed in the SPBRG differently:

SPBRG = (Fosc / (16 x Baud rate)) - 1, BRGH = 1 High Speed

SPBRG = (Fosc / (64 x Baud rate)) - 1, BRGH = 0 Low Speed

In our case, we have: Fosc=48 Mhz, Baud rate=9600

For High Speed => SPBRG = 0x137 hex

For Low Speed => SPBRG = 0x4D hex

After the calculation of this tree register we can set it by this function as follow, as we see this function also have no return no calling input

C code:

```
void InitSerial(void)
{
    SPBRG = 0x4D;            // 4D hex or 77 decimal (baud rate=9600), Low
speed: SPBRG = (Fosc / (64 x Baud rate)) - 1
    TXSTA = 0x22;            // determinng the setting for the transmitter
    RCSTA= 0x90; //determining the setting for the receiver
}
```

SendToSerial function

this function receive a character as calling input and put it into the buffer of the transmission register to be send and a while loop used with the TRMT register to wait until message was sent. No return for this function, it has only char as calling input.

C code:

```
void SendToSerial(char m)
{
    TXREG = m;
    while (TXSTAbits.TRMT==0) {}
}
```

SendStringToSerial function

SCS-SMS

This function is to send a more than one character to serial by only one command. This function has no return but it need a string array from the caller. (@) is specified for **ENTER** meaning.

C code:

```
void SendStringToSerial(char msg[])
{
    int i, lenght;
    lenght = strlen(msg);
    for(i=0; i <= lenght; i++){
        if(msg[i]=='@') break;
        SendToSerial(msg[i]);
    }
}
```

ReceiveFromSerial function

This function is to receive the data send from other and a while loop used with the **RCIF** register to wait until message was received. No calling input for this function but it return the receiving data for the caller.

C code:

```
char ReceiveFromSerial(void)
{
    PORTAbits.RA4 = 0; // reset the alert when data received
    while(PIR1bits.RCIF==0); // Wait until RCIF gets low
    return RCREG; // Retrieve data from reception register
}
```

LCD_CMD function

The goal of this function is to send a command for the LCD. To send a command for the LCD, after the enabling of **E** register you should clear the **RS** and **RW** register. When you clear this two register the LCD know that the data will be receive it is a command. After sending the hexadecimal number of the command the enable register **E** should be disabling again.

Commands Hexadecimal code:

0x01	CLEAR DISPLAY SCREEN
0x02	RETURN HOME
0x04	SHIFT CURSER TO LEFT
0x06	SHIFT CURSER TO RIGHT
0x05	SHIFT DISPLAY TO RIGHT
0x07	SHIFT DISPLAY TO LEFT
0x0C	CURSER OFF

PIC software

0x0E	CURSER BLINKING
0x10	SHIFT CURSOR POSITION TO LEFT
0x14	SHIFT CURSOR POSITION TO RIGHT
0x18	SHIFT THE ENTIRE DISPLAY TO LEF
0x1C	SHIFT THE ENTIRE DISPLAY TO RIGHT
0x80	FORCE CURSOR TO BEGINNING OF 1ST LINE Or you can start from where you want 0x8X ex: 0x83
0xC0	FORCE CURSOR TO BIGINNING OF 2ND LINE Or you can start from where you want 0cX ex: 0xc7
0x38	TO WRITE ON THE TWO LINES

A delay should be executing to harmony the speed of the LCD and PIC; PIC will wait the LCD to be ready to receive a new order. No return for this function but it receives an integer of the command value from the caller.

C code:

```
void LCD_CMD(unsigned int value)
{
    PORTCbits.RC2=1; //E
    PORTCbits.RC0=0; //RS
    PORTCbits.RC1=0; //RW
    Delay10KTCYx(15);
    PORTD=value;
    Delay10KTCYx(15);
    PORTCbits.RC2=0; //E
    Delay10KTCYx(30);
}
```

WRITE_CHAR_LCD function

As the LCD_CMD function, the goal of this function is to send a data to the LCD to write it. To write on LCD, after the enabling of E register you should set the RS register and clear the RW register. When you do that the LCD know that the data will be receive it should be write on it. Now we can send the data to be written on LCD then the enable register E should be disabling again.

Also, a delay should be executing to harmony the speed of the LCD and PIC; PIC will wait the LCD to be ready to receive a new data. No return for this function but it receives the character from the caller.

C code:

```
void WRITE_CHAR_LCD(unsigned char value)
{
    PORTCbits.RC2=1; //E
    PORTCbits.RC0=1; //RS
```

SCS-SMS

```
PORTCbits.RC1=0; //RW
Delay1KTCYx(5);
PORTD=value;
Delay1KTCYx(5);
PORTCbits.RC2=0; //E
}
```

WRITE_STRING_LCD function

This function is to send more than one character (a string) to LCD by only one command. This function has no return but it need a string array from the caller.

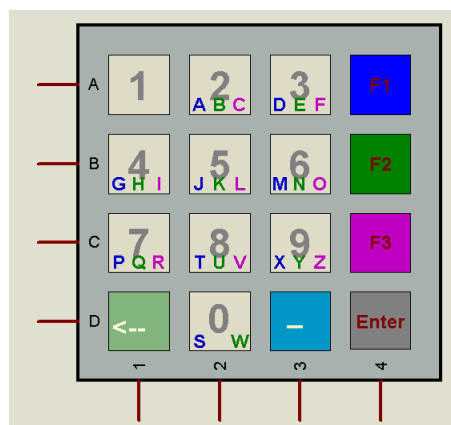
C code:

```
void WRITE_STRING_LCD(char value[])
{
    int i, lenght;
    lenght = strlen(value) - 1;
    for(i=0; i <= lenght; i++){
        WRITE_CHAR_LCD(value[i]);
    }
    //Delay10KTCYx(10);
}
```

KeyPad function

The goal of this function is to get a key pressed by the user. We discussed in the previous chapter how the keypad it works. In this function we write the IF condition which will specific the meaning of each pressed key.

In our project we specify that the normal way of the keypad is to write number with enter and space and we specify a three functional key (f1, f2, f3) to change from the normal way to one of the three functional way which will write the English alphabetic instead of numbers. Figure bellow shows the keypad key specification.



This function has no receiving input from the caller but it returns the pressed character for the caller

C code:

```

char KeyPad(void)
{
    char msg='*'; // to be sure that a key was pressed
    PORTB =0x00;

    msg = KeyPressed;
    if(msg != '*')
    {
        KeyPressed = '*';
        return msg;
    }

waiting:

    PORTBbits.RB0=1;
    if(PORTBbits.RB4==1) return ('1');
    else if(PORTBbits.RB5==1) return '4';
    else if(PORTBbits.RB6==1) return '7';
    else if(PORTBbits.RB7==1) return '<';

    PORTB =0x00;
    PORTBbits.RB1=1;
    if(PORTBbits.RB4==1) return '2';
    else if(PORTBbits.RB5==1) return '5';
    else if(PORTBbits.RB6==1) return '8';
    else if(PORTBbits.RB7==1) return '0';

    PORTB =0x00;
    PORTBbits.RB2=1;
    if(PORTBbits.RB4==1) return '3';
    else if(PORTBbits.RB5==1) return '6';
    else if(PORTBbits.RB6==1) return '9';
    else if(PORTBbits.RB7==1) return ' ';

    PORTB =0x00;
    PORTBbits.RB3=1;
    if(PORTBbits.RB4==1) // F1
    {
        PORTB =0x00;
        msg = '*'; // to be sure that a key was pressed

f1:

        PORTBbits.RB0=1;
        if(PORTBbits.RB5==1) return 'G';
        else if(PORTBbits.RB6==1) return 'P';

        PORTB =0x00;
        PORTBbits.RB1=1;
        if(PORTBbits.RB4==1) return 'A';
        else if(PORTBbits.RB5==1) return 'J';
        else if(PORTBbits.RB6==1) return 'T';
        else if(PORTBbits.RB7==1) return 'S';

        PORTB =0x00;
        PORTBbits.RB2=1;
        if(PORTBbits.RB4==1) return 'D';
        else if(PORTBbits.RB5==1) return 'M';
        else if(PORTBbits.RB6==1) return 'X';

        PORTB =0x00;
        if(msg=='*') goto f1;
    }
}

```

```

else if(PORTBbits.RB5==1)          // F2
{
    PORTB =0x00;
    msg = '*';    // to be sure that a key was pressed

f2:
    PORTBbits.RB0=1;
    if(PORTBbits.RB5==1) return 'H';
    else if(PORTBbits.RB6==1) return 'Q';

    PORTB =0x00;
    PORTBbits.RB1=1;
    if(PORTBbits.RB4==1) return 'B';
    else if(PORTBbits.RB5==1) return 'K';
    else if(PORTBbits.RB6==1) return 'U';
    else if(PORTBbits.RB7==1) return 'W';

    PORTB =0x00;
    PORTBbits.RB2=1;
    if(PORTBbits.RB4==1) return 'E';
    else if(PORTBbits.RB5==1) return 'N';
    else if(PORTBbits.RB6==1) return 'Y';

    PORTB =0x00;
    if(msg=='*') goto f2;
}
else if(PORTBbits.RB6==1)          // F3
{
    PORTB =0x00;
    msg = '*';    // to be sure that a key was pressed

f3:
    PORTBbits.RB0=1;
    if(PORTBbits.RB5==1) return 'I';
    else if(PORTBbits.RB6==1) return 'R';

    PORTB =0x00;
    PORTBbits.RB1=1;
    if(PORTBbits.RB4==1) return 'C';
    else if(PORTBbits.RB5==1) return 'L';
    else if(PORTBbits.RB6==1) return 'V';

    PORTB =0x00;
    PORTBbits.RB2=1;
    if(PORTBbits.RB4==1) return 'F';
    else if(PORTBbits.RB5==1) return 'O';
    else if(PORTBbits.RB6==1) return 'Z';
    else if(PORTBbits.RB7==1) return '^';

    PORTB =0x00;
    if(msg=='*') goto f3;
}

else if(PORTBbits.RB7==1) return '@';

PORTB =0x00;

if(msg=='*') goto waiting;

return msg;
}

```

PIC software

send_SMS function

The goal of this function is to wait the user to write a SMS to be sending via serial. (@) is specified for ENTER meaning.

C code:

```
void send_SMS(void)
{
    int i=0;
    unsigned char SMS[16];
    SMS[0]='*';

Loop:
    SMS[i]=KeyPad();

    if(SMS[i]=='<')
    {
        if(i!=0)
        {
            i--;
            LCD_CMD(0x10); //shift cursor position to
left
        }
        goto Loop;
    }
    else if(SMS[i]=='@' && i!=0)
    {
        SendCyphierToSerial(SMS);
        SMS[i]='*';
        LCD_CMD(0xC0); //Second Line
        goto Loop;
    }
    else if(SMS[i]!='^')
    {
        WRITE_CHAR_LCD(SMS[i]);

        Delay10KTCYx(20);

        if(i==15)
        {
            SendCyphierToSerial(SMS);
            LCD_CMD(0xC0); //Second Line
            i=0;
        }
        else i++;
        goto Loop;
    }
    else SMS[i]='*';
}
```

on_touch function

This function is staying on receiving mode till the user press MENU key.

C code:

```
void on_touch(void)
{
    int i=0;
```

SCS-SMS

```
        unsigned char waiting[15]="WAITING SMS...";
        char rc;
        LCD_CMD(0x01); //CLEAR SCREEN
        LCD_CMD(0x80); //FIRST LINE
        WRITE_STRING_LCD(waiting);

        LCD_CMD(0xC0); //second LINE

        while(1)
        {
/*
            LCD_CMD(0xC0); //second LINE

            ReceiveCypherText();
            Delay10KTCYx(50);

            LCD_CMD(0x01); //Clear screen
*/
            rc = ReceiveFromSerial();
            WRITE_CHAR_LCD(rc);
            i++;
            if(i==16)
            {
                LCD_CMD(0x01); //Clear screen
                LCD_CMD(0xC0); //second LINE
            }
        }
    }
```

send_data function

This function ask the user to choice how he want to send his SMS, to specific one of for all by BROADCAST.

C code:

```
void send_data(void)
{
    unsigned char broadcast[12]="4 broadcast";
    unsigned char to_one[9]="5 to one";
    unsigned char writeM[11]="write SMS:";
    unsigned char EntID[10]="Enter ID:";
    unsigned char GoOut[7]="Go Out";
    char Key_Pressed;
    int i=0;

screen2:
    LCD_CMD(0x01); //CLEAR SCREEN
    LCD_CMD(0x80); //FIRST LINE
    WRITE_STRING_LCD(broadcast);
    LCD_CMD(0xC0); //second LINE
    WRITE_STRING_LCD(to_one);

    Key_Pressed = KeyPad();
    if(Key_Pressed=='4')
    {
        // broadcast choice
        LCD_CMD(0x01); //CLEAR SCREEN
        LCD_CMD(0x80); //FIRST LINE
        WRITE_STRING_LCD(writeM);
        LCD_CMD(0xC0); //second LINE
    }
}
```



```

while(1)
{
    send_SMS();
    Key_Pressed = KeyPad();
    if(Key_Pressed == '@') break;
    else
    {
        LCD_CMD(0xC0); //Second Line
        for(i=16; i>0; i--) WRITE_CHAR_LCD('
');
        LCD_CMD(0xC0); //Second Line
        send_SMS();
    }
}
else if(Key_Pressed=='5')
{
    // to one choice
    LCD_CMD(0x01); //CLEAR SCREEN
    LCD_CMD(0x80); //FIRST LINE
    WRITE_STRING_LCD(EntID);
    LCD_CMD(0xC0); //second LINE
    Key_Pressed = KeyPad();
    if(Key_Pressed=='M')
    {
        LCD_CMD(0x01); //CLEAR SCREEN
        LCD_CMD(0x80); //FIRST LINE
        WRITE_STRING_LCD(writeM);
        LCD_CMD(0xC0); //second LINE
        while(KeyPad() != '@')
        {
            send_SMS();
        }
    }
    else
    {
        WRITE_STRING_LCD(GoOut);
        Delay10KTCYx(50);
    }
}
}

```

Menu function

This function is the MENU which the user choice in it if he want to stay on touch for any coming SMS or if he want to send SMS. This function have no input (caller input) no output (return).

C code:

```

void Menu(void)
{
    unsigned char OnTouch[11]="1 on touch";
    unsigned char SendData[12]="2 send data";
    unsigned char wrong[8]="UnValid";
    unsigned char choice[7]="choice";
    char Key_Pressed;

screen1:
    LCD_CMD(0x01); //CLEAR SCREEN

```

SCS-SMS

```
LCD_CMD(0x80); //FIRST LINE
WRITE_STRING_LCD(OnTouch);
LCD_CMD(0xC0); //second LINE
WRITE_STRING_LCD(SendData);

Key_Pressed = KeyPad();
if(Key_Pressed=='1')
{
    // on touch choice
    on_touch();
}
else if(Key_Pressed=='2')
{
    // send data choice
    send_data();
}
else
{
    LCD_CMD(0x01); //CLEAR SCREEN
    LCD_CMD(0x84); //FIRST LINE specific place
    WRITE_STRING_LCD(wrong);
    LCD_CMD(0xC4); //second LINE specific place
    WRITE_STRING_LCD(choice);
}
}
```

Encryption function

This function is to encrypt message before sending it. This will be encrypting via AES encrypt theorem in the next part.

C code:

```
char Encryption(char PC)
{
    int n=0;
    n = ((int)PC) + 1;
    return ((char)n);
}
```

Decryption function

This function is to decrypt received message before display on LCD. This will be decrypting via AES encrypt theorem in the next part.

C code:

```
char Decryption(char CC)
{
    int n=0;
    n = ((int)CC) - 1;
}
```

Test

```
return ((char) n);  
}
```

Main function

Main function of the program

C code:

```
void main()  
{  
    unsigned char welcometo[11]="welcome to";  
    unsigned char SCS[8]="GIS-ECS";  
  
    PORTAbits.RA4 = 1;  
  
    INIT_Interrupt();  
    INIT_PORT();  
    InitSerial();  
  
    LCD_CMD(0x01);  
    LCD_CMD(0x0E);  
    LCD_CMD(0x38);  
  
    Delay10KTCYx(100); // wait for 500ms  
  
    PORTAbits.RA4 = 0;  
  
    LCD_CMD(0x83); //FIRST LINE  
    WRITE_STRING_LCD(welcometo);  
    LCD_CMD(0xC4); //second LINE  
    WRITE_STRING_LCD(SCS);  
    Delay10KTCYx(10);  
  
begin:  
  
    Menu();  
  
    goto begin;  
}
```

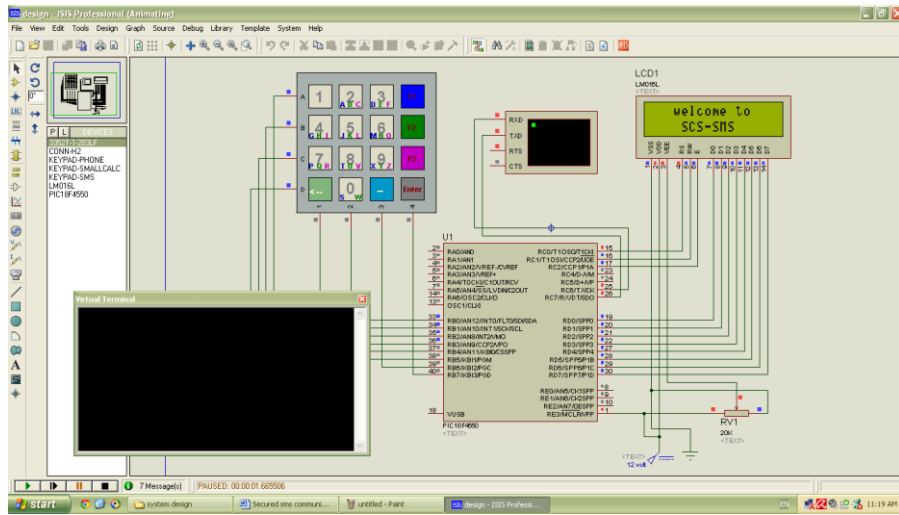
20.5 Test

Before build the hardware system we should test the system on any simulation software. We choice the Proteus ISIS simulation to do the system testing

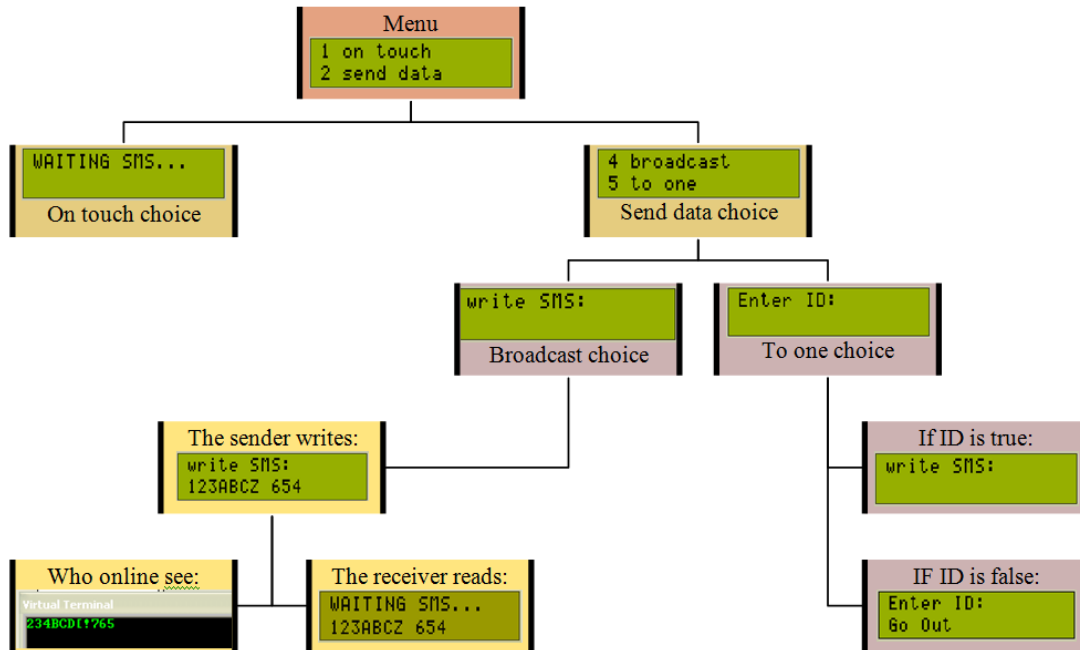
On Proteus:

In the simulation software Proteus ISIS professional the system work as follows:

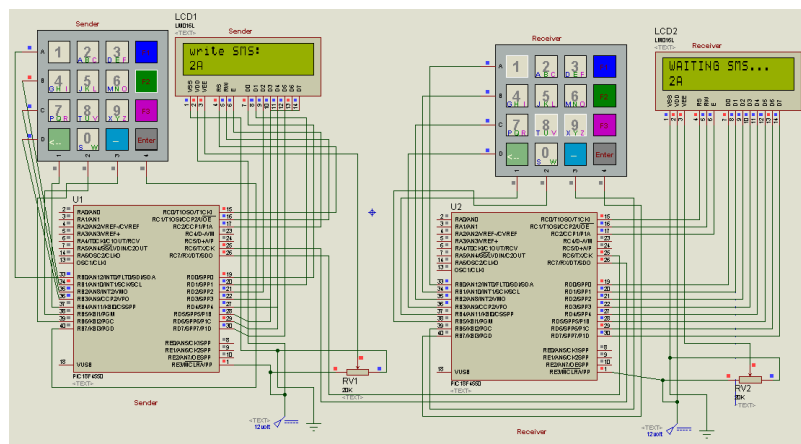
- After connecting components together a virtual terminal should be add in place of the second side to imagine the interaction between two systems.
- When you press the start button, the LCD display the welcome screen (see the figure bellow)



- Then the system work as follows:

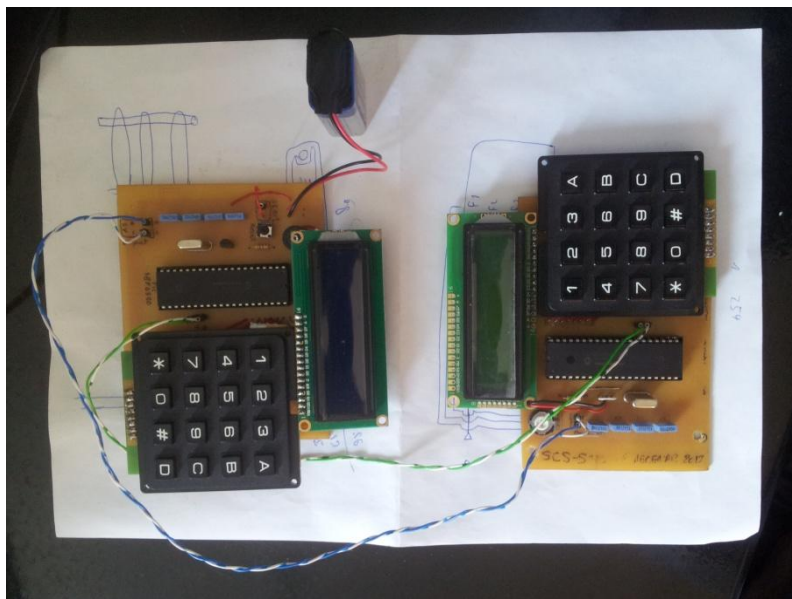
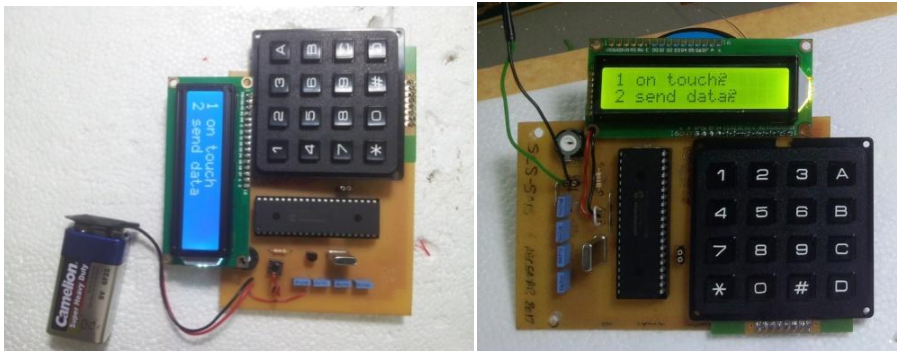


- Also, we test the sender\receiver side together with the following design:



Test

On Real Hardware:



21 AES encryption

Advanced Encryption Standard

One of the most widely used block cipher algorithms is the **Data Encryption Standard (DES)**, adopted in 1977 by the American National Standards Institute (ANSI).

After more than twenty years of use with continuous aging due to advances in cryptography, the National Institute of Standards and Technology (NIST). On 2 October 2000 the NIST announced that the new encryption technique, named **Advanced Encryption Standard (AES)**, would use the Rijndael algorithm, designed by two well-known specialists, Joan Daemen and Vincent Rijmen from Belgium.

21.1 Introduction

AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification *per se* is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4×4 column-major order matrix of bytes, termed the *state*, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the cipher-text. The number of cycles of repetition is as follows:

- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing five similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform cipher-text back into the original plaintext using the same encryption key.

21.2 AES Algorithm¹³

- i. KeyExpansion: round keys are derived from the cipher key using Rijndael's key schedule.
- ii. InitialRound
 - a. AddRoundKey—each byte of the state is combined with the round key using bitwise **xor**.
- iii. Rounds

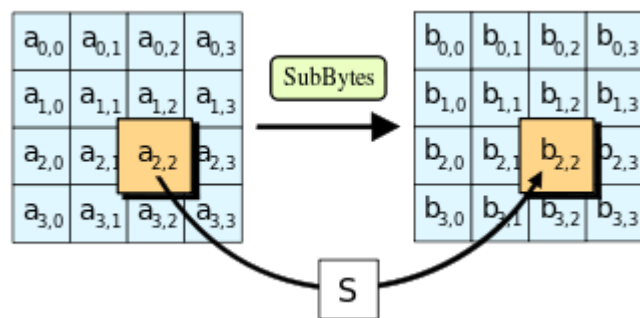
¹³ Reference: http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

AES encryption

- a. *SubBytes*—a non-linear substitution step where each byte is replaced with another according to a lookup table.
 - b. *ShiftRows*—a transposition step where each row of the state is shifted cyclically a certain number of steps.
 - c. *MixColumns*—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
 - d. *AddRoundKey*
- iv. **Final Round (no *MixColumns*)**
- a. *SubBytes*
 - b. *ShiftRows*
 - c. *AddRoundKey*

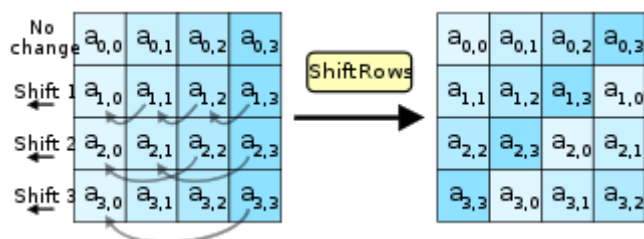
A. The *SubBytes* step

In the *SubBytes* step, each byte in the state matrix is replaced with a *SubByte* using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over $GF(28)$, known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points.



B. The *ShiftRows* step

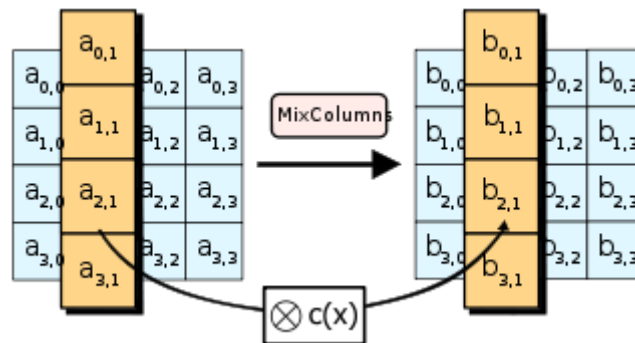
The *ShiftRows* step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same. Row n is shifted left circular by $n-1$ bytes. In this way, each column of the output state of the *ShiftRows* step is composed of bytes from each column of input state. (Rijndael variants with a larger block size have slightly different offsets). For a 256-bit block, the first row is unchanged and the shifting for the second, third and fourth row is 1 byte, 2 bytes



AES Algorithm

and 3 bytes respectively—this change only applies for the Rijndael cipher when used with a 256-bit block, as AES does not use 256-bit blocks. The importance of this step is to make columns not linear independent if so, AES becomes four independent block ciphers.

C. The MixColumns step



In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher.

During this operation, each column is multiplied by the known matrix that for the 128-bit key is:

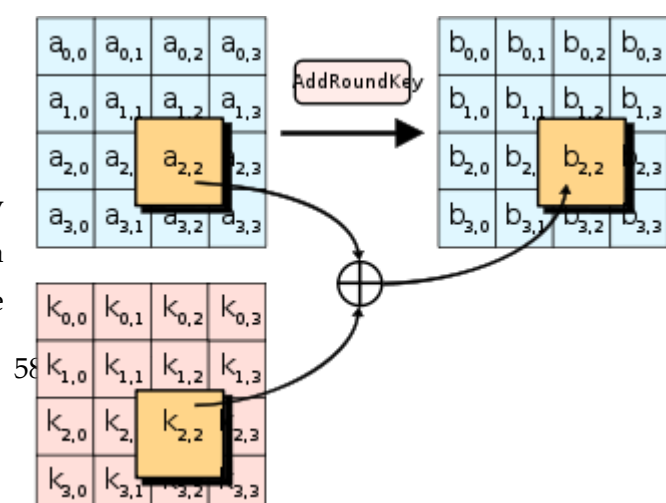
$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

The multiplication operation is defined as: multiplication by 1 means no change, multiplication by 2 means shifting to the left, and multiplication by 3 means shifting to the left and then performing XOR with the initial unshifted value. After shifting, a conditional xor with 0x1B should be performed if the shifted value is larger than 0xFF.

In more general sense, each column is treated as a polynomial over $GF(28)$ and is then multiplied modulo x^4+1 with a fixed polynomial $c(x) = 0x03 \cdot x^3 + x^2 + x + 0x02$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from $GF(2)[x]$. The MixColumns step can also be viewed as a multiplication by a particular MDS matrix in a finite field. This process is described further in the article Rijndael mix columns.

D. The AddRoundKey step

In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the



AES encryption

main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

21.3 Coding ¹⁴

AES is a symmetric key block cipher algorithm. The algorithm executes a series of rounds. The intermediate results of the rounds over the block are called states. In order to prepare for the round transformations, a "KeyExpansion" operation must be executed. This operation uses the original key to create several round keys. Each round key, including the original one, will be used in one of the rounds.

This operation is performed by this C code:

```
void KeyExpansion()
{
    unsigned char i,j;
    unsigned char temp[4],k;

    for(i=0; i<4; i++)
    {
        RoundKey[i*4]=Key[i*4];
        RoundKey[i*4+1]=Key[i*4+1];
        RoundKey[i*4+2]=Key[i*4+2];
        RoundKey[i*4+3]=Key[i*4+3];
    }

    while (i < (4 * (11)))
    {
        for(j=0;j<4;j++)
        {
            temp[j]=RoundKey[(i-1) * 4 + j];
        }
        if (i % 4 == 0)
        {
            k = temp[0];
            temp[0] = temp[1];
            temp[1] = temp[2];
            temp[2] = temp[3];
            temp[3] = k;

            temp[0]=sbox[temp[0]];
        }
        i++;
    }
}
```

¹⁴ Reference: Microchip AN821 Advanced Encryption Standard Using the PIC16XXX

Coding

```
        temp[1]=sbox[temp[1]];
        temp[2]=sbox[temp[2]];
        temp[3]=sbox[temp[3]];

    temp[0] = temp[0] ^ Rcon[i/4];
}
else if (4 > 6 && i % 4 == 4)
{
    temp[0]=sbox[temp[0]];
    temp[1]=sbox[temp[1]];
    temp[2]=sbox[temp[2]];
    temp[3]=sbox[temp[3]];
}
RoundKey[i*4+0] = RoundKey[(i-4)*4+0] ^ temp[0];
RoundKey[i*4+1] = RoundKey[(i-4)*4+1] ^ temp[1];
RoundKey[i*4+2] = RoundKey[(i-4)*4+2] ^ temp[2];
RoundKey[i*4+3] = RoundKey[(i-4)*4+3] ^ temp[3];
i++;
}
}
```

Where `Rcon` represent a vector of round constant, `RoundKey` represent the array that stores the round keys, `Key` is the encryption key, and `sbox` is the encryption substitution table and later we will see `rsbox` which is the decryption substitution table.

These variables are saved in the ROM of the PIC under this code:

```
const rom unsigned char sbox[256] = {
    //0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76, //0
    0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0, //1
    0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15, //2
    0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75, //3
    0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84, //4
    0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, //5
    0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8, //6
    0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, //7
    0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73, //8
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, //9
    0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79, //A
    0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08, //B
    0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a, //C
    0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e, //D
    0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf, //E
    0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16 }; //F

const rom unsigned char Rcon[10] = {
    0x36, 0x1B, 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};

const rom unsigned char rsbox[256] = {
    0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
    0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
    0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
    0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
    0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
    0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
    0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
```

AES encryption

```
0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,  
0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,  
0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,  
0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,  
0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,  
0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,  
0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,  
0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,  
0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d );  
  
unsigned char out[16], state[4][4];  
  
unsigned char RoundKey[160];  
  
unsigned char Key[16]="YOUR_SECURE_KEY";
```

In the encryption process, each of the ten rounds (with the exception of the last one) is composed of four stages:

- byte_sub

```
void substitution_s()  
{  
    int i,j;  
    for(i=0;i<4;i++)  
    {  
        for(j=0;j<4;j++)  
        {  
            state[i][j] = sbox[state[i][j]];  
        }  
    }  
}
```

- shift_row

```
void enc_shift_row()  
{  
    unsigned char temp;  
  
    // Rotate first row 1 columns to left  
    temp=state[1][0];  
    state[1][0]=state[1][1];  
    state[1][1]=state[1][2];  
    state[1][2]=state[1][3];  
    state[1][3]=temp;  
  
    // Rotate second row 2 columns to left
```

Coding

```
temp=state[2][0];
state[2][0]=state[2][2];
state[2][2]=temp;

temp=state[2][1];
state[2][1]=state[2][3];
state[2][3]=temp;

// Rotate third row 3 columns to left
temp=state[3][0];
state[3][0]=state[3][3];
state[3][3]=state[3][2];
state[3][2]=state[3][1];
state[3][1]=temp;
}
```

•
mix_column

```
void mix_column()
{
    int i;
    unsigned char Tmp,Mem;
    for(i=0;i<4;i++)
    {
        Mem = state[0][i];
        Tmp = state[0][i] ^ state[1][i] ^ state[2][i]
^ state[3][i];
        state[0][i] ^= Tmp ^ xtime(state[0][i] ^
state[1][i]);
        state[1][i] ^= Tmp ^ xtime(state[1][i] ^
state[2][i]);
        state[2][i] ^= Tmp ^ xtime(state[2][i] ^
state[3][i]);
        state[3][i] ^= Tmp ^ xtime(state[3][i] ^ Mem);
    }
}
```

While the `xtime` function is done under this process:

```
char xtime(char x)
{
    if(x < 0x80)
        return (x <<= 1);
    else
        return (x = (x << 1) ^ 0x1b);
}
```

AES encryption

- key_addition

```
void key_addition(unsigned char round)
{
    int i,j;
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            state[j][i] ^= RoundKey[round * 16 + i * 4 + j];
        }
    }
}
```

In the decryption process, each of the ten rounds (with the exception of the first one) is composed of four stages:

- inv_byte_sub

```
void substitution_si()
{
    unsigned char i,j;
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            state[i][j] = rsbox[state[i][j]];
        }
    }
}
```

- inv_mix_column

```
void inv_mix_column()
{
    unsigned char i;
    unsigned char Tmp0, Tmp1, Tmp2, Tmp3;

    for(i=0;i<4;i++)
    {
        Tmp0 = state[0][i] ^ state[1][i] ^ state[2][i]
        ^ state[3][i];
        Tmp1 = xtime(state[0][i] ^ state[2][i]);
```

Coding

```
        Tmp2 = xtime(state[1][i] ^ state[3][i]);
        Tmp3 = xtime( xtime( Tmp1 ^ Tmp2 ) ^ Tmp0;
    state[0][i] ^= xtime(state[0][i]^ state[1][i] ^ Tmp1)
^ Tmp3;
    state[1][i] ^= xtime(state[1][i]^ state[2][i] ^ Tmp2)
^ Tmp3;
    state[2][i] ^= xtime(state[2][i]^ state[3][i] ^ Tmp1)
^ Tmp3;
    state[3][i] = state[0][i] ^ state[1][i] ^ state[2][i]
^ Tmp0;
        }
    }
```

•
dec_shift_row

```
void dec_shift_row()
{
    unsigned char temp;

    // Rotate first row 1 columns to right
    temp=state[1][3];
    state[1][3]=state[1][2];
    state[1][2]=state[1][1];
    state[1][1]=state[1][0];
    state[1][0]=temp;

    // Rotate second row 2 columns to right
    temp=state[2][0];
    state[2][0]=state[2][2];
    state[2][2]=temp;
    temp=state[2][1];
    state[2][1]=state[2][3];
    state[2][3]=temp;

    // Rotate third row 3 columns to right
    temp=state[3][0];
    state[3][0]=state[3][1];
    state[3][1]=state[3][2];
    state[3][2]=state[3][3];
    state[3][3]=temp;
}
```

The original key schedule functions use several RAM positions, in order to save all round keys used in the encryption/decryption process.

To reduce the RAM consumption, the implementation of the round keys was done on-the-fly. To do this, three different functions were added:

AES encryption

1. `enc_key_schedule (key)`: This function takes the actual key and generates the next round key that is placed in the same RAM positions.

C code

```
void enc_key_schedule ()
{
    char Rcon = 0x01;
    Key[0] ^= sbox[13];
    Key[1] ^= sbox[14];
    Key[2] ^= sbox[15];
    Key[3] ^= sbox[12];
    Key[0] = Key[0] ^ Rcon;
    Rcon = xtime(Rcon);
    Key[4] ^= Key[0];
    Key[5] ^= Key[1];
    Key[6] ^= Key[2];
    Key[7] ^= Key[3];
    Key[8] ^= Key[4];
    Key[9] ^= Key[5];
    Key[10] ^= Key[6];
    Key[11] ^= Key[7];
    Key[12] ^= Key[8];
    Key[13] ^= Key[9];
    Key[14] ^= Key[10];
    Key[15] ^= Key[11];
}
```

2. `dec_key_schedule (key)`: This function takes the actual key and generates the previous round key that is placed in the same RAM positions.

C code

```
void dec_key_schedule ()
{
    char Rcon = 0x01;
    Key[12] ^= Key[8];
    Key[13] ^= Key[9];
    Key[14] ^= Key[10];
    Key[15] ^= Key[11];
    Key[8] ^= Key[4];
    Key[9] ^= Key[5];
    Key[10] ^= Key[6];
    Key[11] ^= Key[7];
    Key[4] ^= Key[0];
    Key[5] ^= Key[1];
}
```


Coding

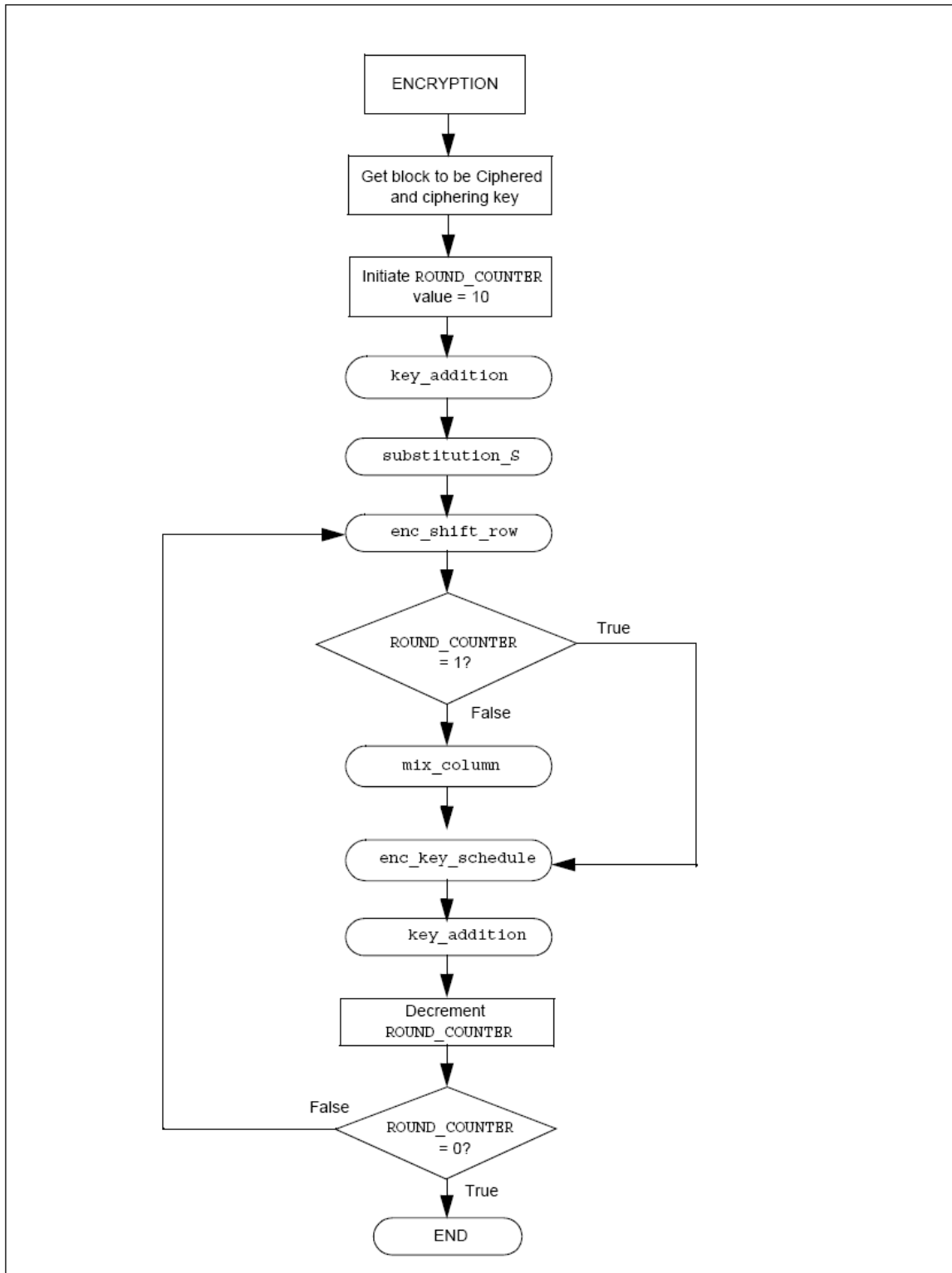
```
Key[6] ^= Key[2];
Key[7] ^= Key[3];
Key[0] ^= sbox[Key[13]];
Key[5] ^= sbox[Key[14]];
Key[6] ^= sbox[Key[15]];
Key[7] ^= sbox[Key[12]];
Key[0] = Key[0] ^ Rcon;

if(Rcon & 0x01)
    Rcon = 0x80;
else
    Rcon >>1;
}
```

AES encryption

Code flow chart

A. Encryption flow chart



Coding

The structure of the encryption program is:

```
void encrypts(char in[])
{
    unsigned char i,j,round;

        KeyExpansion();

    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            state[j][i] = in[i*4 + j];
        }
    }

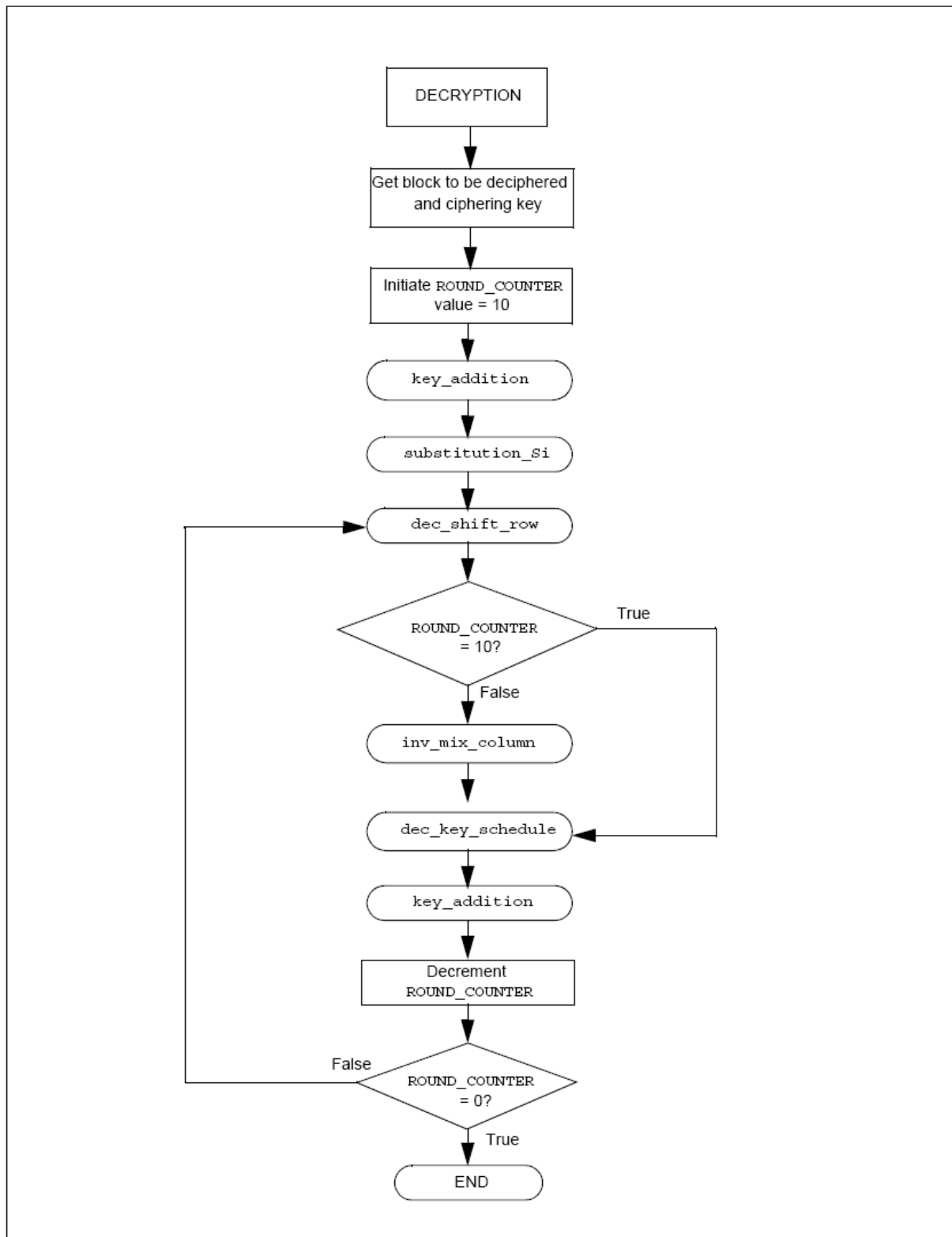
    key_addition(0);

    for(round=1;round<=10;round++)    // 10 rounds
        {
            substitution_s();
            enc_shift_row();
            if( round != 10 ) // last round is done
without mix_column
                mix_column();
            enc_key_schedule(); // direct key_schedule
executed on-the-fly
            key_addition(round);
        }

    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            out[i*4+j]=state[j][i];
        }
    }
}
```

AES encryption

B. Decryption flow chart



Coding

Then structure of the decryption program is:

```
void decrypts(char in[])
{
    unsigned char i,j,round;

    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            state[j][i] = in[i*4 + j];
        }
    }

    key_addition(10);

    round = 9;
    for(round;round>0;round--)
    {
        substitution_si();
        dec_shift_row();
        key_addition(round);
        inv_mix_column();
        dec_key_schedule(); // inverse key_schedule
executed on-the-fly
    }
    substitution_si();
    dec_shift_row();
        dec_key_schedule(); // inverse key_schedule
executed on-the-fly
    key_addition(0);

    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            out[i*4 +j]=state[j][i];
        }
    }
}
```


22 Hardware of ECS Demo System

22.1 Realization of RF Module

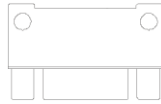
22.1.1 Using STD-402

In SCS-SMS project we use the USART theory which is included in the PCI microcontroller to send and receive data via serial. So, we can use the STD-402 to send and receive message by adding the max232 circuit. We use the max232 IC to translate the Tx/Rx of microcontroller (0V – 5V) to the RS232 Tx/Rx (-10V, +10V).

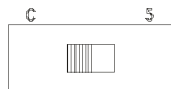
- **The test of the transceiver is shown in following lines:**

The MB-STD-RS232 board which equips the STD-402 transceiver module has stored unique module identification number in the radio module. When one unit is set up for master and other unit is for slave, slave modem unit operate with same ID as master unit.

1. Connect the serial cable to the D-SUB 9pin connector.



2. Set "Cable SW" (Cross / straight) according to the cable.

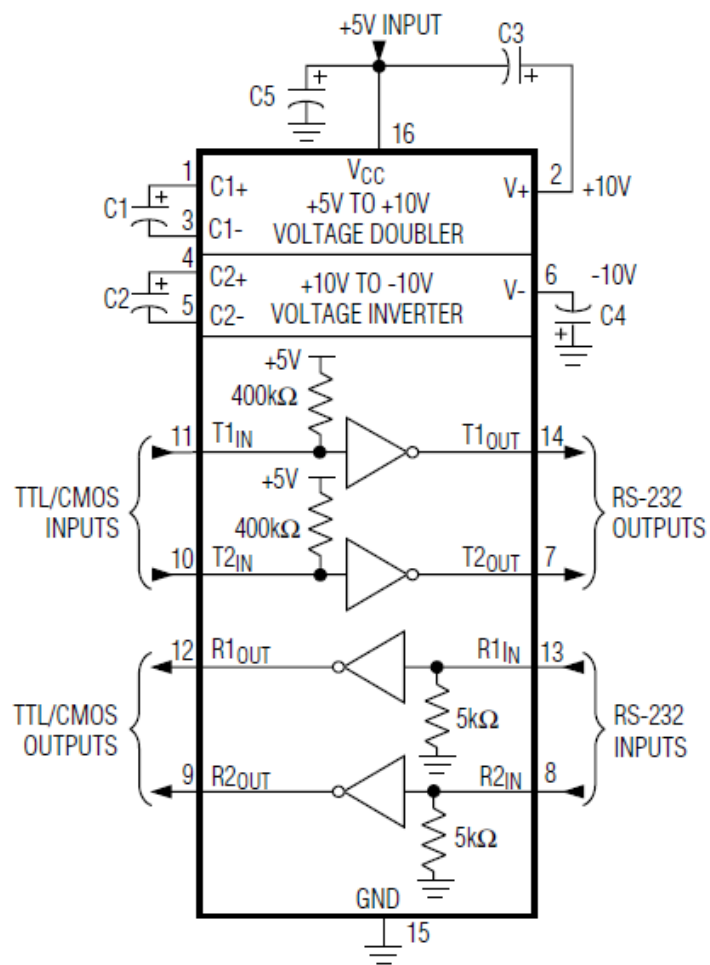
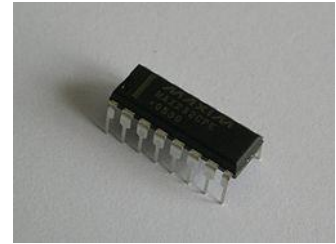


Realization of RF Module

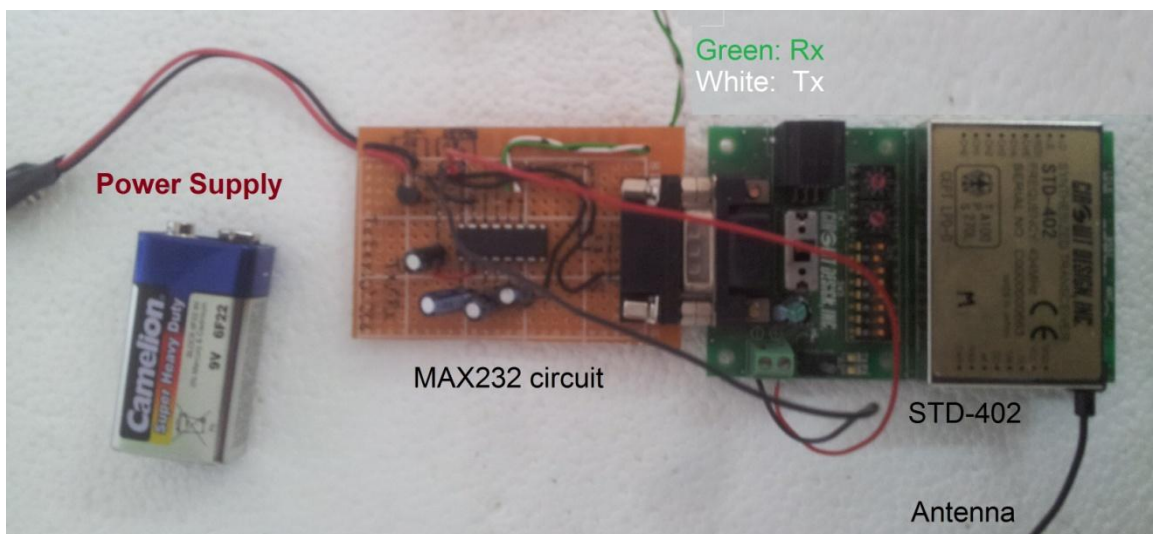
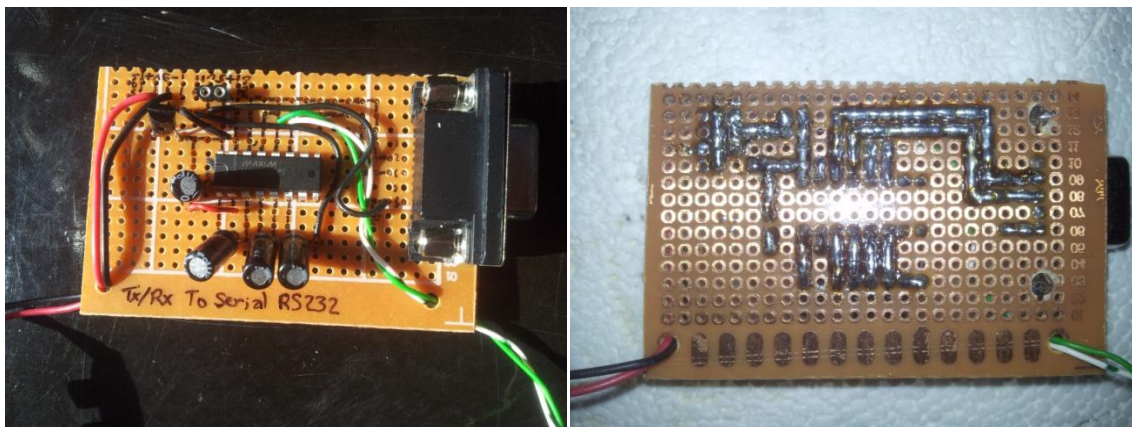
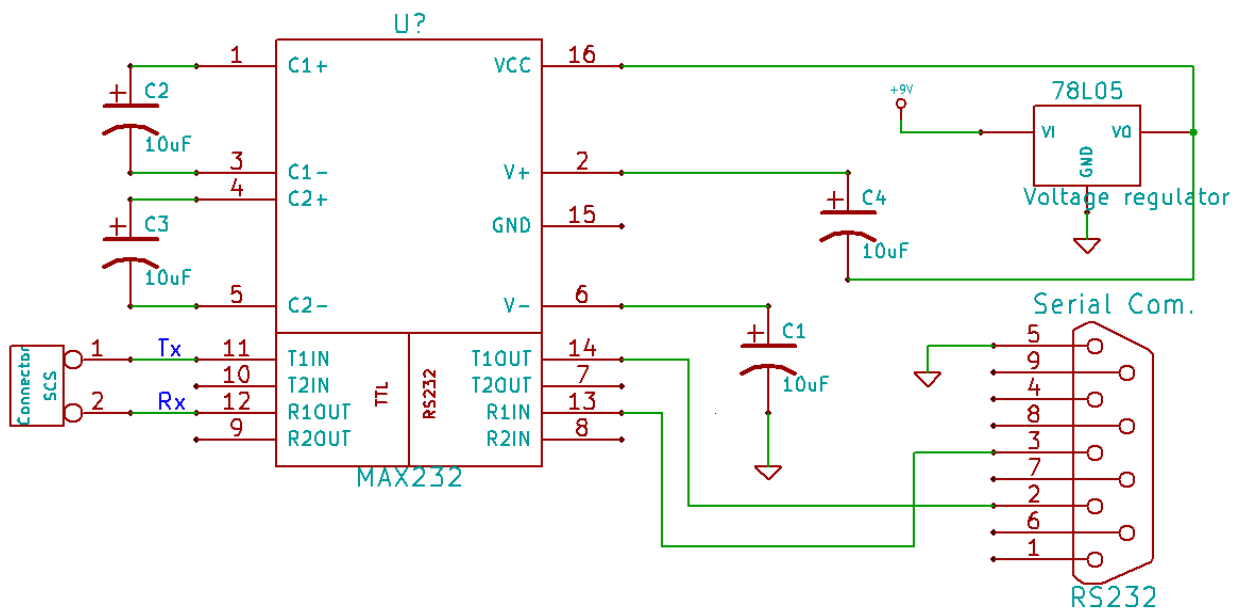


Hardware of ECS Demo System

TXRX
□ □



Realization of RF Module



22.1.2 Realization of RF Module Using RFM42B-RFM31B - 433MHz

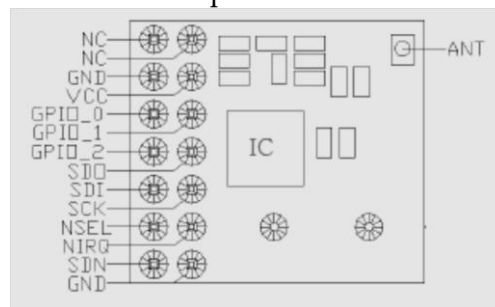
The RFM42B and the RFM31B use the SPI communication theory to send (RFM42B) or to receive (RFM31B) data. So, to use this two RF module some software and hardware change should be implement to the SCS-SMS project.

22.1.2.1 Serial Peripheral interface (SPI)

The RFM31B/42B communicates with the host MCU over a standard 3-wire SPI interface: SCLK, SDI, and nSEL. The host MCU can read data from the device on the SDO output pin. A SPI transaction is a 16-bit sequence which consists of a Read-Write (R/W) select bit, followed by a 7-bit address field (ADDR), and an 8-bit data field (DATA) as demonstrated in Figure 2. The 7-bit address field is used to select one of the 128, 8-bit control registers. The R/W select bit determines whether the SPI transaction is a read or writes transaction. If R/W = 1 it signifies a WRITE transaction, while R/W = 0 signifies a READ transaction. The contents (ADDR or DATA) are latched into the RFM31B/42B every eight clock cycles. The SCLK rate is flexible with a maximum rate of 10 MHz.

To read back data from the RFM31B/42B, the R/W bit must be set to 0 followed by the 7-bit address of the register from which to read. The 8 bit DATA field following the 7-bit ADDR field is ignored in the SDI pin when R/W = 0. The next eight negative edge transitions of the SCLK signal will clock out the contents of the selected register. The data read from the selected register will be available on the SDO output pin. The READ function is shown in Figure 3. After the READ function is completed the SDO pin will remain at either logic 1 or logic 0 state depending on the last data bit clocked out (D0). When nSEL goes high the SDO output pin will be pulled high by internal pullup.

The figure bellow shows us the PIN description:



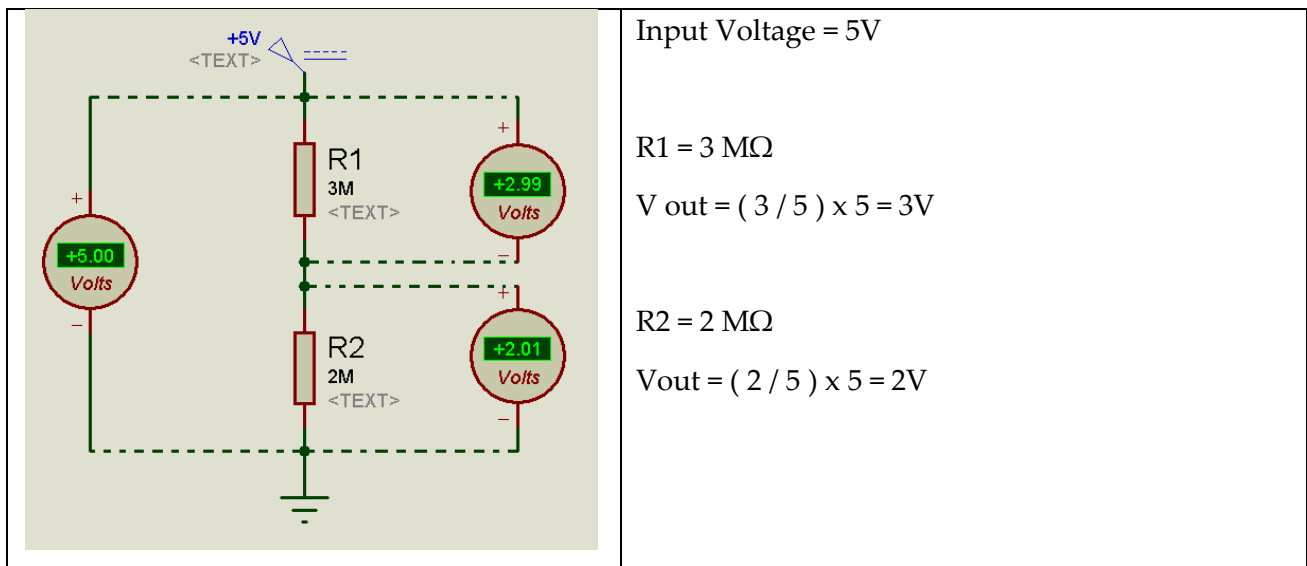
22.1.2.2 The new hardware design

The change which will be happen is caused by adding the two RFM module, the two RFM module connect to the same pins (5 pins connect to the PIC and three connect to the power supply). The RFM module is low power consumption, it's need about 85mA with supply voltage range between 1.8V and 3.6V, and you can see in its datasheet that the best is 3.0V. For this reason and because of our system supply voltage is 5V, so we need to regulate the supply voltage of the RFM to 3V. We do that by voltage divider theory by adding two high impedance (M Ω) resistors.

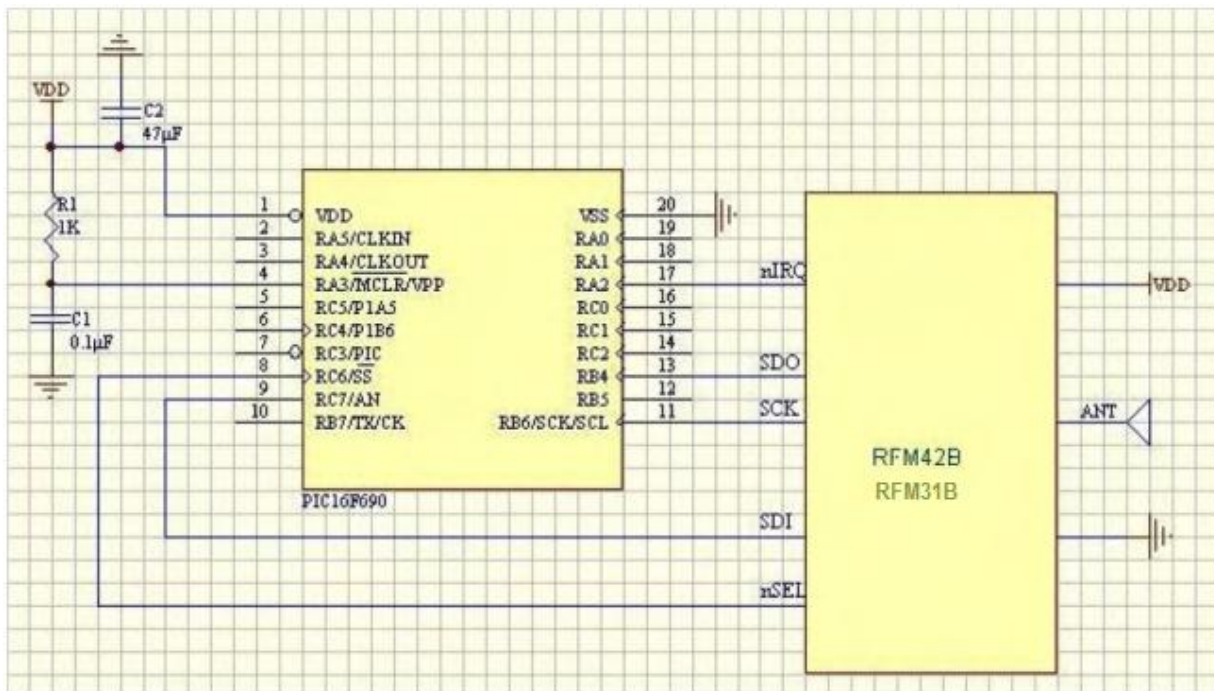
We know that our supply voltage is 5V which is regulate by 78L05 and the goal is to get 3V, so using voltage divider theory as you can see in the next figure we can get it. The cause of using

Realization of RF Module

two high impedance resistors is to eliminate the effect of the interior impedance in the system and the impedance in the RFM module.



Now, let connect the RFM modules to the PIC. The figure bellow shows us how the module connects to PIC microcontroller:



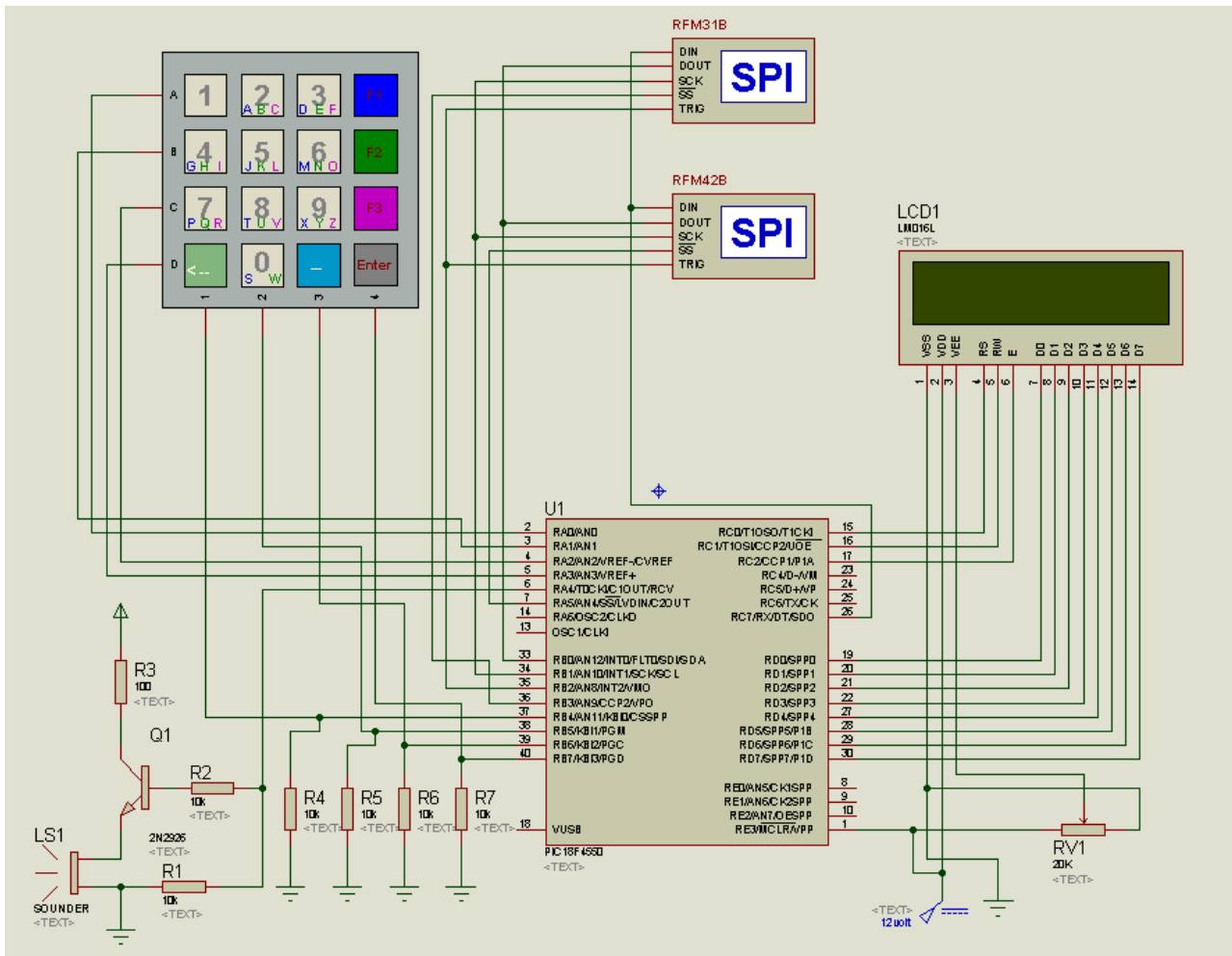
In SCS-SMS we use the PIC 18f4550, so the connection should be doing as follows:

RFM module pins	PIC pins
SDO	SDI (RB0 PIN)
SCK	SCK (RB1 PIN)
NIQR	INT2 (RB2 PIN)
SDI	SDO (RC7)

NSEL	SS (RA5) for RFM31B & (RB3) for RFM42B
------	--

While two else pins should connected to the Ground and one else should connect to +3V, and all the remaining pins doesn't connected anywhere.

As we see in the table above, we have 3 connections to the port B which already used in the SCS-SMS project for the KEYPAD. So, we need to change the connection of the KEYPAD to another pins (we take pins RA0 to RA3 for input keypad pins and RB4 to RB7 for the output keypad pins). Finally the new circuit design became as follows:



22.1.2.3 MSSP module to establishing (SPI)¹⁵

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communication with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. the MSSP module can operate in one of two methods:

- Serial Peripheral interface (SPI)
- Inter-integrated circuit (I2C)

¹⁵ PIC18F4550 datasheet, chapter 19, page 197

Realization of RF Module

▪ Control Registers:

The MSSP module has three associated control registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual Configuration bits differ significantly depending on whether the MSSP module is operated in SPI or I2C mode.

▪ SPI mode:

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of the SPI are supported. To accomplish communication, typically three pins are used:

- Serial Data Out (**SDO**) – RC7/RX/DT/SDO
- Serial Data In (**SDI**) – RB0/AN12/INT0/FLT0/SDI/SDA
- Serial Clock (**SCK**) – RB1/AN10/INT1/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select (**SS**) – RA5/AN4/SS/HLVDIN/C2OUT

▪ Registers:

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (**SSPCON1**)
- MSSP Status Register (**SSPSTAT**)
- Serial Receive/Transmit Buffer Register (**SSPBUF**)

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower six bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations; SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not doublebuffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

SSPSTAT register:

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE ⁽¹⁾	D/ \bar{A}	P	S	R/ \bar{W}	UA	BF
bit 7							bit 0

SMP: sample bit, in master mode (1: data sampled at end, 0: at middle), in slave mode (SMP=0)

CKE: SPI clock select bit (1: transmit on transition from active to Idle, 0: from Idle to active)

D/A, P, S, R/W, UA: used in I2C only

Hardware of ECS Demo System

BF: Buffer full status bit when receive (1: SSPBUF full, 0: receive not complete)

SSPCON1 register:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV ⁽¹⁾	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

WCOL: write collision Detect bit (on transmitting) (1: collision, 0: no collision)

SSPOV: receive overflow indicator bit (slave mode) (1: overflow, 0: no overflow)

SSPEN: master Synchronous Serial Port Enable bit (1: enable serial port, 0: serial port be I/O)

CKP: clock polarity select bit (1: idle state for clock is High, 0: is low)

SSPM3:SSPM0: Master Synchronous serial Port Mode Select bits

0101 = SPI Slave mode, clock=SCKpin, Sspin control disabled, SS can be used as I/Opin

0100 = SPI Slave mode, clock=SCKpin, Sspin control enabled

0011 = SPI Master mode, clock=TMR2 output/2

0010 = SPI Master mode,, clock=Fosc/64

0001 = SPI Master mode,, clock=Fosc/16

0000 = SPI Master mode,, clock=Fosc/4

In our case, we need to use the micro controller in Master mode to connect it the the 2 slave modules (RFM31B and RFM42B) we have.

▪ **Master mode:**

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This, then, would give waveforms for SPI communication. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum data rate (at 48 MHz) of 12.00 Mbps.

When used in Timer2 Output/2 mode, the bit rate can be configured using the PR2 Period register and the Timer2 prescaler. However, writing to SSPBUF does not clear the current TMR2 value in hardware. Depending upon the current value of TMR2 when the user firmware writes to SSPBUF, this can result in an unpredictable MSb bit width.

Realization of RF Module

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

▪ Implementation:

In our case, the PIC is used as SPI in master mode and initially set the two registers as follows;

SSPSTAT: (0xC0) SMT=1, CKE=1, all the remaining bits are readable not writable as we see before.

SSPCON1: (0x30) WCOL=0, SSPOV=0, SSPEN=1, CKP=1, SSPM=0000.

Below is the SPI code in C programming with comment when necessary: (SPI.c)

```
#include<p18f4550.h>
#include<string.h>
#include<delays.h>
#include"SPI.h"

void InitDAC(void)
{
    NSEL1pin = 0;
    NSEL2pin = 0;
    SDIpin = 1;
    SCKpin = 0;
    NIQRpin = 1;
    SDOpin = 0;
    ADCON1=0x0f; //Turn off A/D

    SSPSTAT=0xC0; //SMP: SPI master mode, CKE: active to idle clock state, 0, 0, 0,
0, 0, 0
    SSPCON1=0x30; // Mode 1,1 SPI Master Mode, 1/4 TOSC bit
}

void Send_char_SPI(char data)
{
    SS1 = 0; // Enable SS1 Output (low)
    SS2 = 1; // Disable SS2 Output (high)
    SSPBUF=data; // sending the upper 8 bits serially
    while(!SSPSTATbits.BF); // wait until the upper 8 bits are sent
    SS1 = 1; // Disable SS1 Output (high)
    SS2 = 0; // Enable SS2 Output (low)
}

void Send_int_SPI(int data)
{
    unsigned int c;
    unsigned int lower_bits, upper_bits;
    c = ((data+1)*16) -1; // here we obtain 12 bit data

    //first obtain the upper 8 bits
    upper_bits = c/256; // obtain the upper 4 bits
    upper_bits = (48) | upper_bits; // append 0011 to the above 4 bits

    //now obtain the lower 8 bits
    lower_bits = 255 & c; // ANDing separates the lower 8 bits

    SS1 = 1; // Disable SS1 Output (high)
    SS2 = 0; // Enable SS2 Output (low)

    PORTBbits.RB0=0;
    SSPBUF=upper_bits; // sending the upper 8 bits serially
    while(!SSPSTATbits.BF); // wait until the upper 8 bits are sent
    SSPBUF=lower_bits; // sending the lower 8 bits serially
    while(!SSPSTATbits.BF); // wait until the lower 8 bits are sent
    PORTBbits.RB0=1;
}
```


Hardware of ECS Demo System

```
    SS1 = 1;    // Disable SS1 Output (high)
    SS2 = 0;    // Enable SS2 Output (low)
}

void Send_string_SPI(char s[])
{
    int i, lenght;
    lenght = strlen(s);

    for(i=0; i <= lenght; i++)
    {
        Send_char_SPI(s[i]);
        Delay10KTCYx(17); // wait until the 8 bits char are sent
                           // Delay Subroutine: 169129 Clock cycles ~= 17(10KTCY)
                           ~= 0.014 seconds
    }
}

char Receive_data_SPI(void)
{
    while(!SSPIFbits.PIR1); // Interrupt flag set when transmission/reception is
complete
    return SSPBUF;
}
}
```

Bellowe is the Header file for this c librerly: (SPI.h)

```
#ifndef SPI_H
#define SPI_H

#define Clock_Khz 48000 //Fosc = 48Mhz

#define NSEL1pin TRISAbits.TRISA5 // connect to the NSEL pin of the Tx
#define NSEL2pin TRISBbits.TRISB3 // connect to the NSEL pin of the Rx
#define SDIpin TRISBbits.TRISB0 // connect to the SDO pin of SPI module
#define SCKpin TRISBbits.TRISB1 // connect to the SCK pin of SPI module
#define NIQRpin TRISBbits.TRISB2 // connect to the NIQR pin of SPI module
#define SDOpin TRISCbits.TRISC7 // connect to the SDI pin of SPI module

#define SS1 PORTAbits.RA5 // selection slave 1 (transmitter)
#define SS2 PORTBbits.RB3 // selection slave 2 (receiver)

void InitSPI(void);
void Send_char_SPI(char n);
void Send_int_SPI(int n);
void Send_string_SPI(char s[]);
char Receive_data_SPI(void);

#endif
```


23 Further Work: System Integration and Integration Test of ECS Demo System

tbd

Appendix A: Alternative Project Plans

A.1: Three Alternatives for Central Station / Mobile Stations

As we see before, each side of this project has multiple potential choices. And each choice has its extra tasks. In this part we will specific the tasks with the period of each choice of each side.

- **For the STD hardware.**

- a. Using existing STD hardware (ran) in the two sides (connect to computer).
No extra potential tasks for this choice (**only 1 week for testing**)
- b. Using existing STD hardware (ran) in the base side and using the new SCS-SMS hardware on the client side

Tasks for this choice:

- Change the SCS-SMS hardware to be able to use (**need among 3 weeks**)
- Change and develop the input part of the ExtIO file to be compatible with the client side hardware (**2 weeks**)

Note that, if this choice is taken you can't take the HDSDR as a choice for the SDR.

- c. Develop a hardware to be able to connect to 4 antenna on the base side and to one antenna on the client side (base on HackRF project)

Tasks for this choice:

- Develop the hardware for one antenna for the client side (**3 weeks**)
- Develop the hardware for 4 antennas for the base side (**2 extra weeks**)
- Change and develop the input part of the ExtIO file to be compatible with the client side hardware (one antenna) (**2 weeks**)
- Change and develop the input part of the ExtIO file to be compatible with the base side hardware (4 antenna) (**1 extra week**)

- **For the SDR code**

- a. Using HDSDR by develop its ExtIO.

Tasks for this choice:

- Change and develop the output part of the ExtIO file to be compatible with the GUI interface software (**4 weeks**)
- Develop our GUI interface (**4 weeks**)

- b. Using the source code of WinRad

Tasks for this choice:

- Take the SDR code from the WinRad software (3 weeks)
- Develop the SDR code to send and receive (2 weeks)
- Change and develop the output part of the ExtIO file to be compatible with the GUI interface software (4 weeks)
- Develop our GUI interface (4 weeks)

A side to this task and duration there are the testing and documentation tasks and period. Bellow is two project plans:

Project 1: choice (a) for STD hardware choice (a) for the SDR code

Using existing STD hardware (ran) in the two sides (connect to computer). With using HSDR by develop its ExtIO for the SDR code

Event	Time
Getting start with software (Qt, VC++, HSDR)	2 weeks
Using HSDR to receive and transmit Radio wave	1 week
Getting Start with DLL and see demo ExtIO	1 week
Change and develop the output part of the ExtIO file to be compatible with the GUI interface software	4 weeks
Develop our GUI interface	4 weeks
System testing (task 9)	2 weeks
Documentation and Final report (task 10)	3 weeks

Approximately 4 months and 1 week with a possibility of delay

Project 2: choice (b) for STD hardware choice (b) for the SDR code

Using existing STD hardware (ran) in the base side and using the new SCS-SMS hardware on the client side. With using of the source code of WinRad for the SDR code

Event	Time
Getting start with software (Qt, VC++, WinRad)	2 weeks
Using WinRad to receive Radio wave	1 week
Read with understanding the WinRad code	1 week
Take the SDR code from the WinRad software	2 weeks
Develop the SDR code to send and receive	2 weeks
Getting Start with DLL and see demo ExtIO	1 week
Change and develop the output part of the ExtIO file to be compatible with the GUI interface software	4 weeks
Develop our GUI interface	4 weeks
Change the SCS-SMS hardware to be able to use	3 weeks
Change and develop the input part of the ExtIO file to be compatible with the client side hardware	2 weeks
System testing (task 9)	2 weeks
Documentation and Final report (task 10)	3 weeks

Approximately 6 months and 2 weeks with a possibility of delay

Project 3: choice (c) for STD hardware choice (b) for the SDR code

Further Work: System Integration and Integration Test of ECS Demo System

Develop hardware to be able to connect to 4 antennas on the base side and to one antenna on the client side with using of source code of WinRad

Event	Time
Getting start with software (Qt, VC++, WinRad)	2 weeks
Using WinRad to receive Radio wave	1 week
Read with understanding the WinRad code	1 week
Take the SDR code from the WinRad software	2 weeks
Develop the SDR code to send and receive	2 weeks
Getting Start with DLL and see demo ExtIO	1 week
Change and develop the output part of the ExtIO file to be compatible with the GUI interface software	4 weeks
Develop our GUI interface	4 weeks
Develop the hardware for one antenna for the client side	3 weeks
Develop the hardware for 4 antennas for the base side	2 weeks
Change and develop the input part of the ExtIO file to be compatible with the client side hardware (one antenna)	2 weeks
Change and develop the input part of the ExtIO file to be compatible with the base side hardware (4 antenna)	1 week
System testing (task 9)	2 weeks
Documentation and Final report (task 10)	3 weeks

Approximately **7 months and 2 weeks** with a possibility of delay

A.2: Demo System Integration with different developers

Event	Time
Using WinRad to receive Radio wave using exist SDR platform	1 week
Introduction to HackRF SDR platform	2 weeks
Build our SDR platform	2 weeks
Using WinRad to receive Radio wave via new SDR platform	1 week
Build our Amateur Radio Transceiver (ART)	3 weeks
Connect the SCS-SMS hardware to the ART with testing	2 weeks
Take I and Q from WinRad to a file	1 week
Develop GUI interface to read SMS from file	2 weeks
System testing	2 weeks
Documentation and Final report	2 weeks

Approximately **18 weeks** with a possibility of delay

These tasks are dividing to a three work packages, which are:

1st package: building of the SDR platform

ID	Name	Start	Finish	October 2013				November 2013		
				07	14	21	28	04	11	18
1	Using WinRad to receive radio wave using exist SDR platform	10/7/2013	10/13/2013							
2	introduction to HackRF SDR platform	10/14/2013	10/27/2013							
3	Build our SDR platform	10/28/2013	11/10/2013							
4	using WinRad to receive Radio wave via new SDR platform	11/11/2013	11/17/2013							

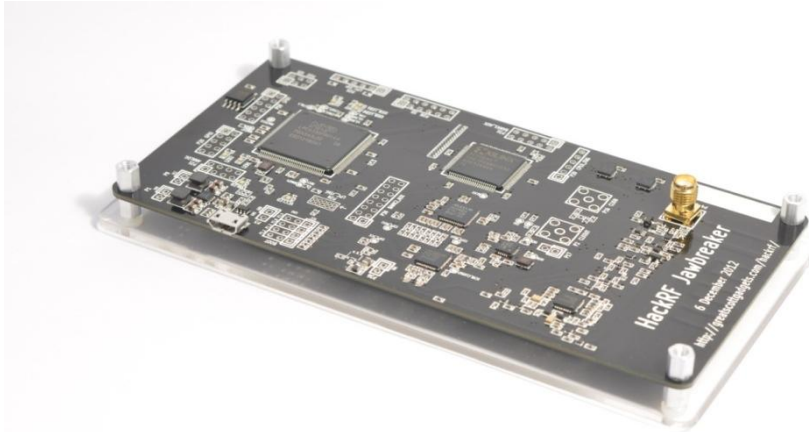
2nd package: building of the Amateur Radio Transceiver

Name	Start	Finish	October 2013				November 2013		
			07	14	21	28	04	11	18
Getting start to Amateur Radio Transceiver	10/7/2013	10/13/201	█						
build our Amateur Radio Transceiver	10/14/201	10/27/201		█	█	█			
test with SCS-SMS	10/28/201	11/10/201					█	█	█

3rd package: WinRad and interface software

ID	Name	Start	Finish	October 2013				November 2013		
				07	14	21	28	04	11	18
1	introduction to WinRad	10/7/2013	10/13/2013	█						
2	take I & Q from WinRad to a file	10/14/2013	10/20/2013		█					
3	develop user interface to read SMS	10/21/2013	11/3/2013			█	█			
4	system testing	11/4/2013	11/10/2013					█		

Appendix B: All about HackRF



HackRF is a project to produce a low cost, open source software radio platform.

Principal author: Michael Ossmann: mike@ossmann.com

Home hackRF website: <https://github.com/mossmann/hackrf>

HackRF is an open source hardware project to build a Software Defined Radio (SDR) peripheral.

B.1 HackRF overview

SDR is the application of Digital Signal Processing to radio waveforms. It is similar to the software-based digital audio techniques that became popular a couple of decades ago. Just as a sound card in a computer digitizes audio waveforms, a software radio peripheral digitizes radio waveforms. It's like a very fast sound card with the speaker and microphone replaced by an antenna. A single software radio platform can be used to implement virtually any wireless technology (Bluetooth, ZigBee, cellular technologies, FM radio, etc.).

Digital audio capabilities in general purpose computers enabled a revolution in the sound and music industries with advances such as hard disk recording and MP3 file sharing. Today's computers are fast enough to process radio waveforms in similar ways, and the radio communications industry is going through the same sorts of changes. One critical advance is finally taking place now, and that is the availability of low cost tools enabling anyone to take part in the revolution.

✓ **Wide Operating Frequency Range:**

HackRF operates from 30 MHz to 6 GHz, a wider range than any SDR peripheral available today. This range includes the frequencies used by most of the digital radio systems on Earth. It can operate at even lower frequencies in the MF and HF bands when paired with the Ham It up RF up converter.

✓ **Transceiver:**

HackRF can be used to transmit or receive radio signals. It operates in half-duplex mode: it can transmit or receive but can't do both at the same time. However, full-duplex operation is possible if you use two HackRF devices.

✓ **Low Cost:**

HackRF was designed to be the most widely useful SDR peripheral that can be manufactured at a low cost. The estimated future retail price of HackRF is \$300, but you can get one for even less by backing the Kickstarter project today.

✓ **Wideband:**

The maximum bandwidth of HackRF is 20 MHz, about 10 times the bandwidth of TV tuner dongles popular for SDR. That means that HackRF could be used for high speed digital radio applications such as LTE or 802.11g.

✓ **Open Source:**

The most important goal of the HackRF project is to produce an open source design for a widely useful SDR peripheral. All hardware designs and software source code are available under an open source license. The hardware designs are produced in KiCad, an open source electronic design automation tool. You can download the Jawbreaker (HackRF beta) design and build your own HackRF today!

✓ **Compatible:**

HackRF beta units are already being used on Linux, OS X, and Windows platforms. The device takes full advantage of USB 2.0, an interface found on almost every general purpose computer. HackRF already works with the popular GNU Radio software framework, and HackRF support can be added to other SDR software.

✓ **Tested:**

The Jawbreaker design depicted above is the fully functional HackRF beta design. Hundreds of Jawbreakers have been distributed to developers and beta testers. HackRF has already been used for Digital Audio Broadcasting (DAB), Bluetooth monitoring, spectrum sensing, wireless microphones, AIS, FM radio, and more. I plan to use feedback from beta testers to make your HackRF even better than Jawbreaker.

B.2 Jawbreaker¹⁶

Jawbreaker is the first complete HackRF platform, a wideband software radio transceiver with a USB interface.

Hardware notes:

Schematic and layout files were designed in **KiCad**, an open source electronic design automation package.

order of copper layers:

Copper 1: Front
Copper 2: Inner3
Copper 3: Inner2
Copper 4: Back

PCB description: 4 layer PCB 0.062 in

Copper 1 0.5 oz foil plated to approximately 0.0017 in
Dielectric 1-2 0.0119 in
Copper 2 1 oz foil (0.0014 in)
Dielectric 2-3 0.0280 in
Copper 3 1 oz foil (0.0014 in)
Dielectric 3-4 0.0119 in
Copper 4 0.5 oz foil plated to approximately 0.0017 in

FR4 or similar substrate with $\epsilon_r=4.5$ (+/- 0.1)

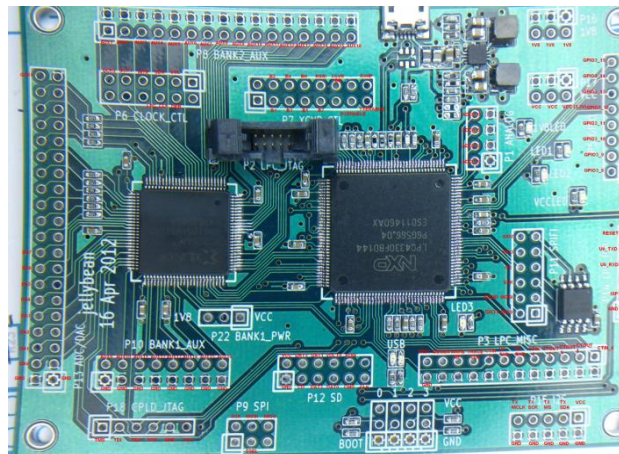
double side solder mask black

double side silkscreen white

6 mil min trace width and

6 mil min isolation

B.3 Jellybean¹⁷



¹⁶ ..\6-ECS-SDR\5-Guide\HackRF\hardware\jawbreaker

This file contain also the Hardware design file, but it should open by **KiCad** software

¹⁷ ..\6-ECS-SDR\5-Guide\HackRF\hardware\jellybean

This file contain also the PCB design as pdf, with the Hardware design file, but it should open by KiCad software

Jellybean is a microcontroller platform based on the LPC43xx. It is designed to control Lemondrop.

Hardware notes:

Schematic and layout files were designed in **KiCad**, an open source electronic design automation package.

order of copper layers:

```
Front
Inner3
Inner2
Back
```

PCB description: 4 layer PCB 1.6 mm

```
Copper      1    35 um
Dielectric  1-2  0.35 mm
Copper      2    18 um
Dielectric  2-3  0.76 mm
Copper      3    18 um
Dielectric  3-4  0.35 mm
Copper      4    35 um
```

```
DE104iML or equivalent substrate (Er=4.42@2.4GHz TanD=0.016)
double side solder mask black
double side silkscreen white
6 mil min trace width and
6 mil min isolation
```

This file contain also the PCB design as pdf, with the Hardware design file, but it should open by KiCad software

B.4 Lemondrop¹⁸

Lemondrop is a 2.3 to 2.7 GHz wireless transceiver with a 22 Msps ADC/DAC and flexible clocking for software radio applications.

Hardware notes:

Schematic and layout files were designed in **KiCad**, an open source electronic Design automation package.

order of copper layers:

```
Front
Inner3
Inner2
Back
```

PCB description: 4 layer PCB 1.6 mm

```
Copper      1    35 um
Dielectric  1-2  0.35 mm
```

¹⁸ ..\6-ECS-SDR\5-Guide\HackRF\hardware\lemondrop

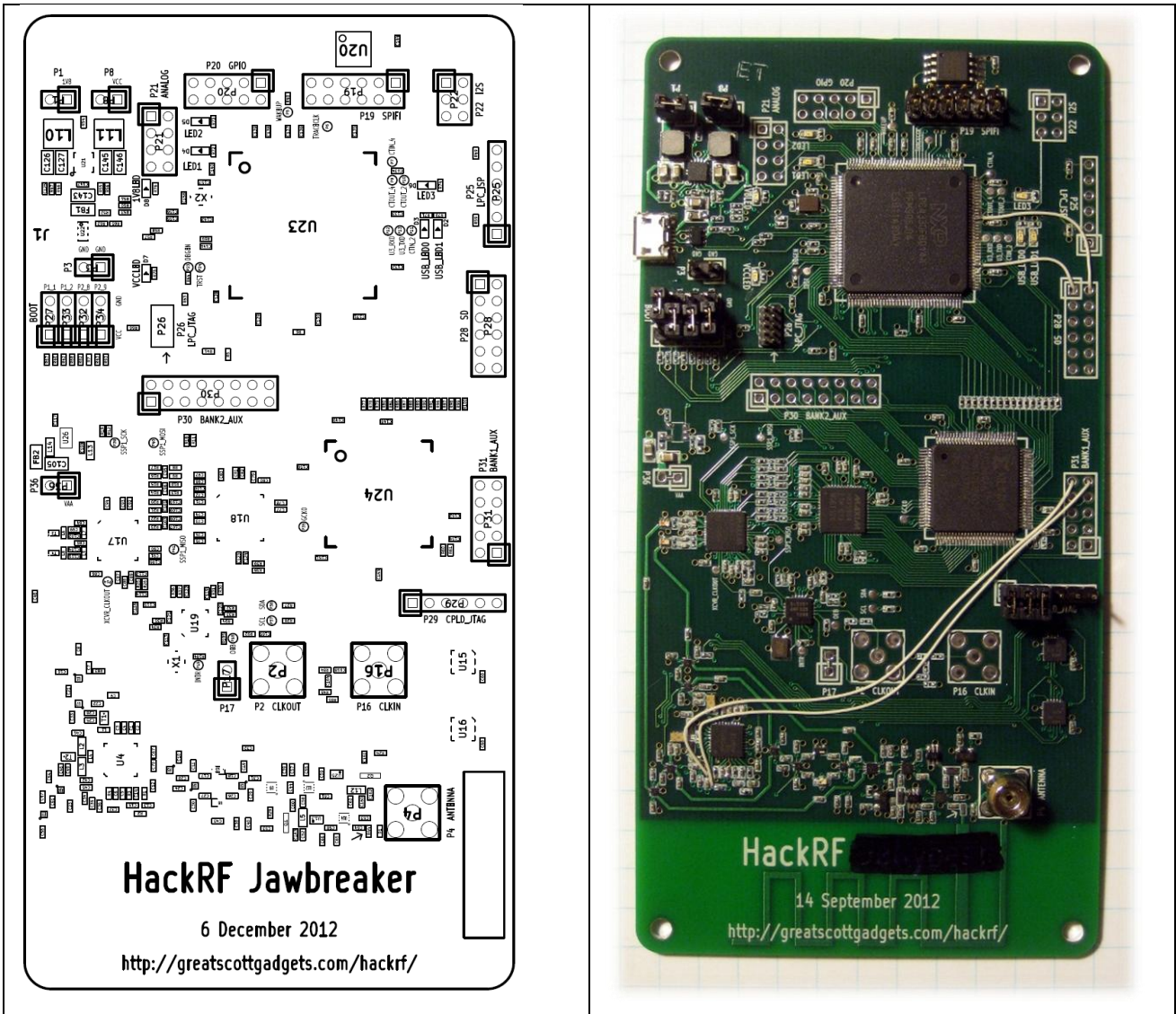
This file contain also the Hardware design file, but it should open by **KiCad** software

Further Work: System Integration and Integration Test of ECS Demo System

Copper	2	18 um
Dielectric	2-3	0.76 mm
Copper	3	18 um
Dielectric	3-4	0.35 mm
Copper	4	35 um

DE104iML or equivalent substrate (Er=4.42@2.4GHz TanD=0.016)
double side solder mask black
double side silkscreen white
6 mil min trace width and
6 mil min isolation

B.5 HackRF Hardware



- **Board IC:**

U1-U2-U5-U6-U7-U10-U11:SKY13350;Skyworks;SKY13350-385LF;0.01-6.0 GHz GaAs SPDT Switch
U3:RX_LOWPASS_FILTER;AVX;LP0603A1880ANTR;FILTER LOW PASS 1880MHZ 0603 SMD
U4:RFFC5072;RFMD;RFFC5072TR7;WIDEBAND SYNTHESIZER/VCO WITH INTEGRATED 6GHz MIXER
U8:RX_HIGHPASS_FILTER;TDK;DEA162400HT-8004B1;FILTER HIGHPASS WLAN&BLUETOOTH
U9-U12-U14:SKY13317;Skyworks;SKY13317-373LF;20 MHz-6.0 GHz pHEMT GaAs SP3T Switch
U13-U25:MGA-81563;Avago;MGA-81563-TR1G;0.1-6 GHz 3 V
U15:GSG-74HC04;Texas Instruments;SN74AHC04RGYR;IC HEX INVERTERS 14-QFN
U16:GSG-74HC08;Texas Instruments;SN74AHC08RGYR;IC QUAD 2IN POS-AND GATE 14-QFN
U17:MAX2837;Maxim;MAX2837ETM+;IC TXRX 2.3GHZ-2.7GHZ 48TQFN
U18:MAX5864;Maxim;MAX5864ETM+;IC ANLG FRONT END 22MSPS 48-TQFN
U19:SI5351C;Silicon Laboratories Inc;SI5351C-B-GM;IC CLK GENERATOR 160MHZ 20QFN
U20:W25Q80BV;Winbond;W25Q80BVSSIG;IC FLASH 8MBIT 8SOIC
U21:TPS62410;Texas Instruments;TPS62410DRCR;IC BUCK SYNC DUAL ADJ 0.8A 10SON
U22:GSG-IP4220CZ6;NXP;IP4220CZ6
U23:LPC43XXFBD144;NXP;LPC4330FBD144
U24:GSG-XC2C64A-7VQG100C;Xilinx;XC2C64A-7VQG100C;IC CR-II CPLD 64MCELL 100-VQFP
U26:RF LDO;DNP

- **Other component:**

FB1:FILTER;Murata;BLM21PG221SN1D;FERRITE CHIP 220 OHM 2000MA 0805
FB2:FILTER;Murata;BLM21PG221SN1D;FERRITE CHIP 220 OHM 2000MA 0805

Q1:MOSFET_P;Fairchild;BSS84;MOSFET P-CH 50V 130MA SOT-23
Q2:MOSFET_P;Fairchild;BSS84;MOSFET P-CH 50V 130MA SOT-23

T1:MIX_IN_BALUN;Anaren;B0310J50100AHF;Ultra Low Profile 0805 Balun 50 to 100 ohm Balanced

T2:MIX_OUT_BALUN;Anaren;B0310J50100AHF;Ultra Low Profile 0805 Balun 50 to 100 ohm Balanced

T3:RX_BALUN;Johanson Technology;2500BL14M100T;BALUN CERAMIC CHIP WIMAX 2.5GHZ

T4:TX_BALUN;Johanson Technology;2500BL14M100T;BALUN CERAMIC CHIP WIMAX 2.5GHZ

X1:GSG-XTAL4PIN;AVX;CX3225GB25000D0HEQZ1;CRYSTAL 25.000MHZ 8PF SMD

X2:MCU_XTAL;TXC;7V-12.000MAAE-T;CRYSTAL 12.000 MHZ 12PF SMD

With a lot of: capacitors, resistors, inductors, ports with jumpers

B.6 Extra file

- LPCXpresso Flash Debug Tutorial(pdf file)¹⁹

This pdf file contains the following points:

- Hardware required:
 - NXP LPC-Link board included with any LPCXPRESSO Board
 - LPC43xx board
- Software required:
 - LPCXpresso v4.2.3 build 292
- Starting LPCXpresso IDE
- Create a project
- Flashing ".bin" or ".elf" in SPIFI flash memory
- Debugger configuration

- LPCXpresso Flash Debug Tutorial(pdf file)²⁰

This file contains the following folders:

Blinky
Blinky_rom_to_ram
Common
Cpld
Cpldjtagprog
Cpldjtagprog_rom_to_ram
Hackrf_usb
Hackrf_usb_rom_to_ram
Mixertx
sgpio
sgpio_passthrough_rom_to_ram
sgpio_rx
simpletx
spiflash
startup
startup_systick
startup_systick_perfo
startup_systick_perfo_rom_to_ram

¹⁹ ..\6-ECS-SDR\5-Guide\HackRF\doc\LPCXpresso_Flash_Debug_Tutorial.pdf

²⁰ ..\6-ECS-SDR\5-Guide\HackRF\firmware

With makefile and this readme note:

The primary firmware source code for USB HackRF devices is `hackrf_usb`. Most of the other directories contain firmware source code for test and development. The common directory contains source code shared by multiple HackRF firmware projects. The `cpld` directory contains HDL source for the CPLD present on the Jawbreaker and Jellybean designs.

The firmware is set up for compilation with the GCC toolchain available [here](#):

<https://code.launchpad.net/gcc-arm-embedded>

Required dependency:

<https://github.com/mossmann/libopencm3>

Another file named `firmware-bin` contain `hackrf_usb_rom_to_ram.bin`

B.7 Host build²¹

How to build host software on Windows:

Prerequisite for `cygwin` or `mingw`:

* `cmake-2.8.10.2` or more see <http://www.cmake.org/cmake/resources/software.html>

* `libusb-1.0.14` or more see

<http://sourceforge.net/projects/libusb/files/latest/download?source=files>

* Install Windows driver for HackRF hardware or use Zadig see

<http://sourceforge.net/projects/libwdi/files/zadig>

- If you want to use Zadig select HackRF USB device and just install/replace it with WinUSB driver.

* Build `libhackrf` before to build this library, see `host/libhackrf/Readme.md`.

For Cygwin:

```
cmake -G "Unix Makefiles" -DCMAKE_LEGACY_CYGWIN_WIN32=1 -
DLIBUSB_INCLUDE_DIR=/usr/local/include/libusb-1.0/
make
make install
```

For Mingw:

```
#normal version
cmake -G "MSYS Makefiles" -DLIBUSB_INCLUDE_DIR=/usr/local/include/libusb-1.0/
#debug version
cmake -G "MSYS Makefiles" -DCMAKE_BUILD_TYPE=Debug -
DLIBUSB_INCLUDE_DIR=/usr/local/include/libusb-1.0/
make
make install
```

²¹ ..\6-ECS-SDR\5-Guide\HackRF\host\hackrf-tools

Appendix C: Alternative System Designs

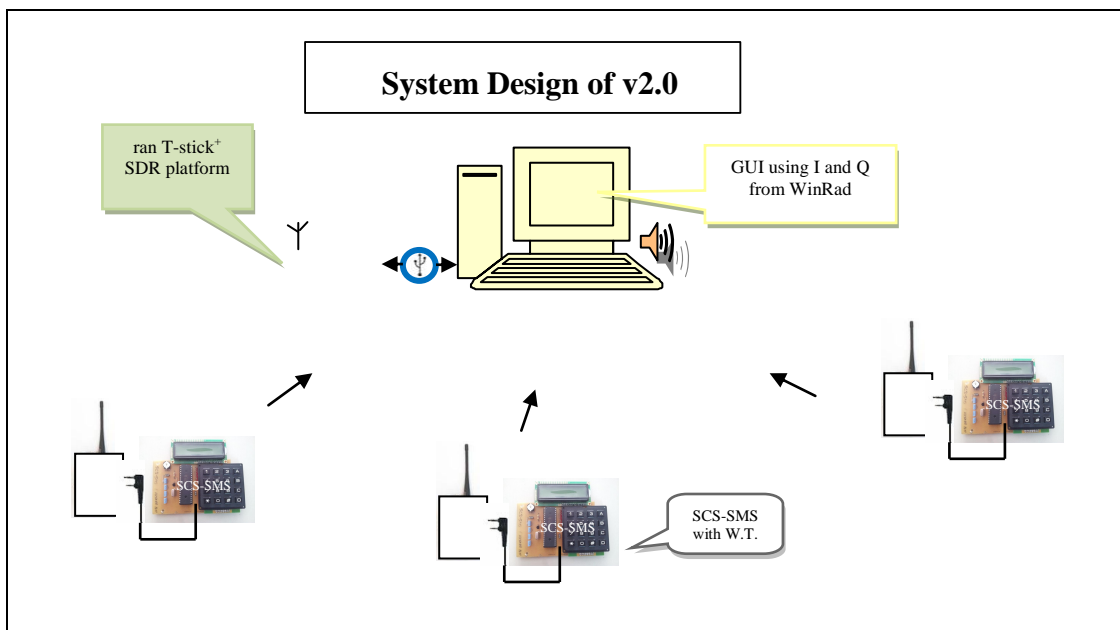
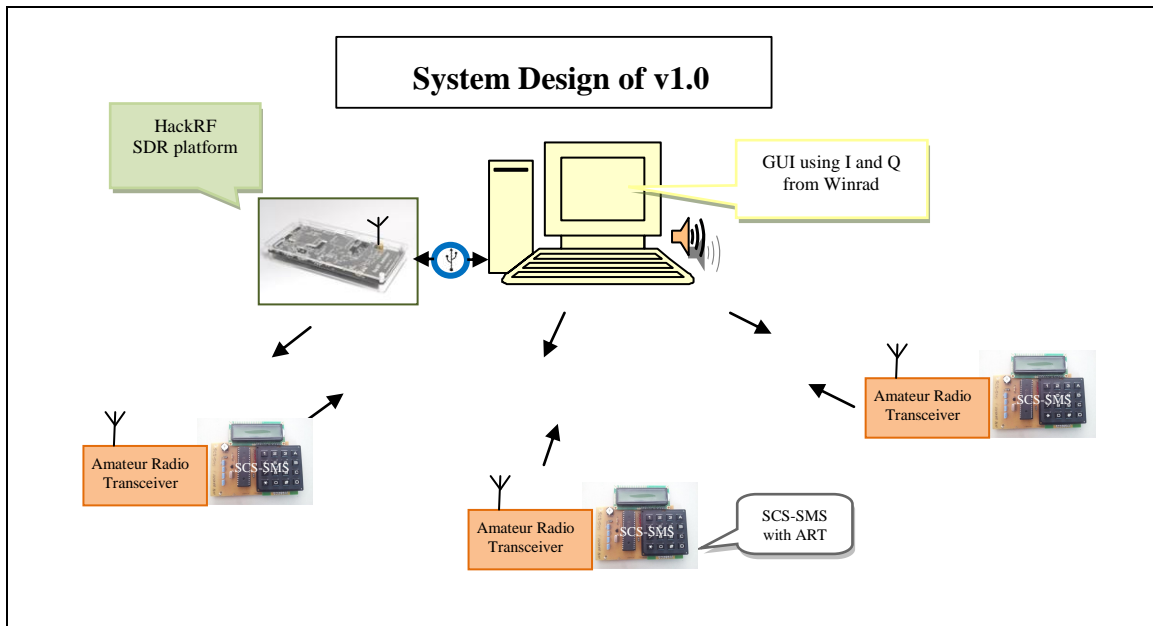


Fig. 5.1 A and B: System Overview

Literature

[HarckRF] ...

<http://en.wikibooks.org/wiki/Special:BookSources/0900612584>

...

**Communication System with HarckRF (from IAP-SAT, 6th project report)
(2020)**

Eng. MMJZ

24 Telemetry System with HarckRF

By MMJZ



24.1 Time Plan

Title	Description	Estimated Time	Status
Task 1: Introduction			
HackRF	Introduce HackRF and getting start with, a brochure summarize the main info should be prepared and established	5 days	Done
SDR (Software Defined Radio)	Introduce and understand SDR workflow and check for its available software	5 days	Done
Task 2: Getting Start			
Receiving signal	Use available SDR software to receiving signal using HackRF	2 days	Done
Transmitting signal	Use available SDR software to transmitting signal using HackRF	2 days	Done
Pentoo OS	Prepare Pentoo OS on a machine or USB image to be used with the HackRF	2 days	Done
GnuRadio	Getting start with GnuRadio compain and make some tutorials	3 days	Done

Raspberry&GnuRadio	Install GnuRadio on Raspberry Pi OS and make some tests	4 days	Done
Task 3: System 1 (Send and receive system “separately”)			
Make a Sender	Build a system to send a RF signal using a HackRF and Raspberry Pi.	4 days	In Progress
Make a Receiver	Build a system to receive a RF signal using a HackRF and developed SDR software	7 days	In Progress
Task 4: System 2 (Send \ Receive system)			
Raspberry S \ R	Update the Raspberry Pi system to make it two way communication sys (send \ receive)	3 days	
SDR software S \ R	Update the developed SDR software to make it two way communication sys. (send \ Receive)	3 days	
Task 5: System 3: Lotte System			
Arduino > Raspberry	Connect Arduino to Raspberry and send and receive data from it using USB connection	5 days	
Raspberry > HackRF	Use HackRF to send Arduino data, and to receive command for it	10 days	
HackRF > Gui	Use a Windows developed software to receive sent data from the raspberry side to be access	10 days	

Total time: 65 days,

Spent time: 25 days,

Remaining time: 40 days

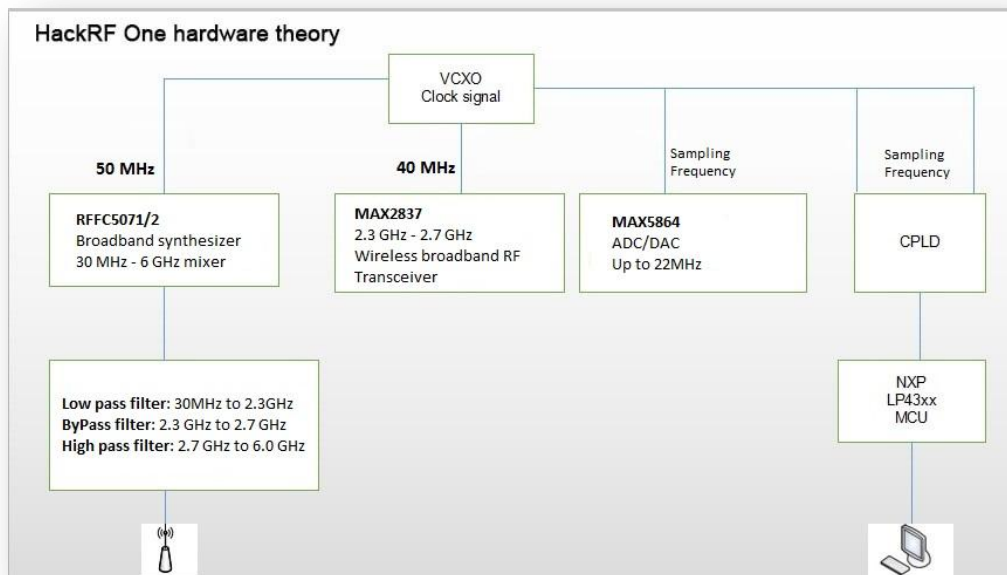
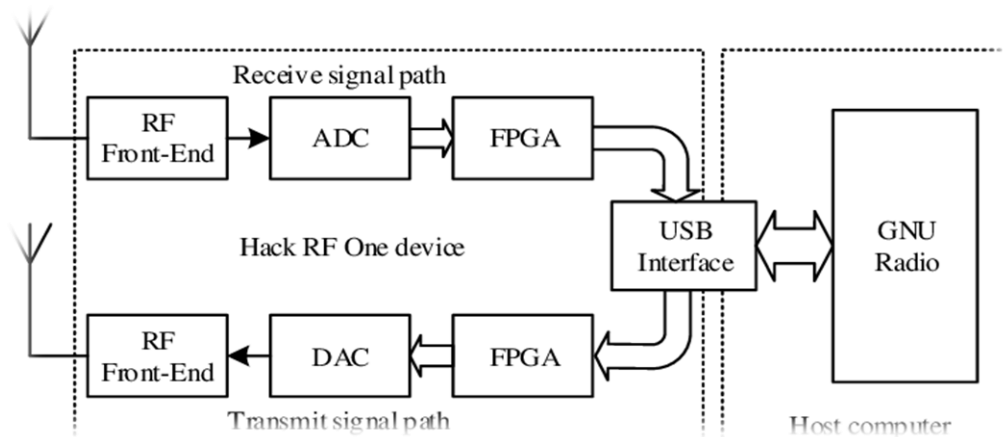
24.2 Introduction

HackRF One from **Great Scott Gadgets** is a Software Defined Radio peripheral capable of transmission or reception of radio signals from 1 MHz to 6 GHz. Designed to enable test and development of modern and next generation radio technologies, **HackRF One** is an open source hardware platform that can be used as a USB peripheral or programmed for stand-alone operation.

Specifications:

- 1 MHz to 6 GHz operating frequency
- Half-duplex transceiver
- Up to 20 million samples per second
- 8-bit quadrature samples (8-bit I and 8-bit Q)

- Compatible with GNU Radio, SDR#, and more
- Software-configurable RX and TX gain and baseband filter
- Software-controlled antenna port power (50 mA at 3.3 V)
- SMA female antenna connector
- SMA female clock input and output for synchronization
- Convenient buttons for programming
- Internal pin headers for expansion
- Hi-Speed USB 2.0
- USB-powered
- Open source hardware



Parameters:

- Frequency band: 1MHz-6Ghz
- Data bandwidth: 20MHz
- Sampling accuracy (ADC/DAC) : 8BIT
- Sampling speed (ADC/DAC): 20Mbps
- Maximum transmitting power: 10dbm

- 64QAM transmitting EVM: 1.5%
- Complex sampling bandwidth: 20Mhz

To invoke DFU mode: Press and hold the DFU button. While holding the DFU button, reset the HackRF One either by pressing and releasing the RESET button or by powering on the HackRF One. Release the DFU button.

The DFU button only invokes the bootloader during reset. This means that it can be used for other functions by custom firmware.

Hardware:

- Mixer RFFC5072: 80MHz-4200MHz
- Wireless bandwidth RF transceiver MAX2837: 2.3Ghz-2.7Ghz
- Processor LPC4330: Main frequency 204MHz
- Amplifier MGA-81563: 0.1-6Ghz, 3V, 14dbm
 - The RF switch determines whether to amplify via a 14db amplifier
 - The signal is filtered by high pass or loss pass filter
 - Signal RFFC5072 chip mixing to 2.6GHz fixed medium frequency
 - The firmware supports variable intermediate frequency options: range 2.15 GHz to 2.75 GHz
 - Signal into the MAX2837 chip mixing to the baseband, output differential IQ signal (MAX2837 chip can limit the bandwidth of the signal)
 - The MAX5864 chip digitizes the baseband signal and sends it to CPLD
 - The LPC4320/4330 processor sends the sampled data to computer via USB
 - RFFC5072 and MAX2837 are protected in a shield to prevent interference from the outside world or other chips on the board, and to prevent static electricity from penetrating some chips

24.2.1 Links and references:

Official site: <https://greatscottgadgets.com/hackrf/one/>

HackRF Lessons of Michael Ossmann: <https://greatscottgadgets.com/sdr/>

Source files on github: <https://github.com/mossmann/hackrf>

Portapack source files on github: <https://github.com/sharebrained/portapack-hackrf>

Files:



HackRF_Poster.pdf

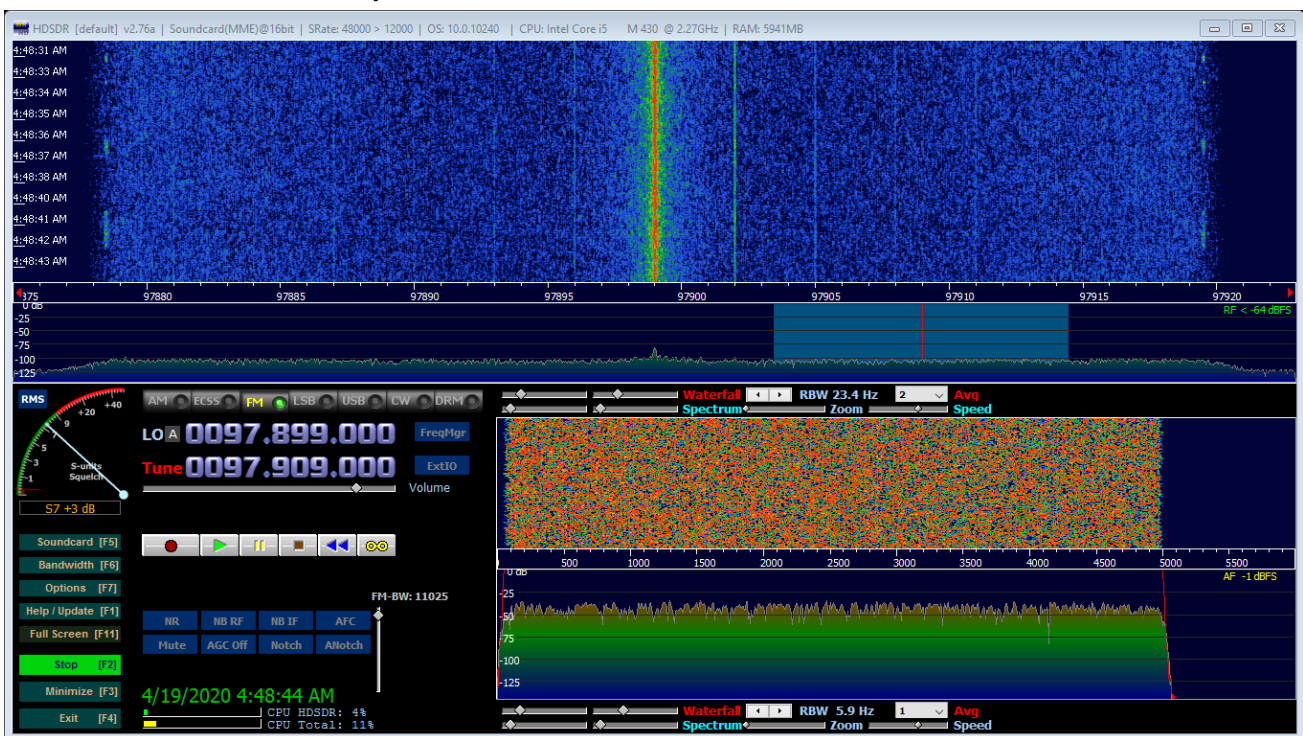
24.2.2 SDR (Software defined radio):

Software-defined radio (SDR) is a radio communication system where components that have been traditionally implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system.

There are a lot of SDR software globally like: HDSDR, SDRSharp...

24.2.3 HDSDR:

HDSDR is a freeware Software Defined Radio (SDR) program for Microsoft Windows 2000/XP/Vista/7/8/8.1/10. Typical applications are Radio listening, Ham Radio, SWL, Radio Astronomy, NDB-hunting and Spectrum analysis. HDSDR (former WinradHD) is an advanced version of Winrad, written by Alberto di Bene (I2PHD).



Main features:

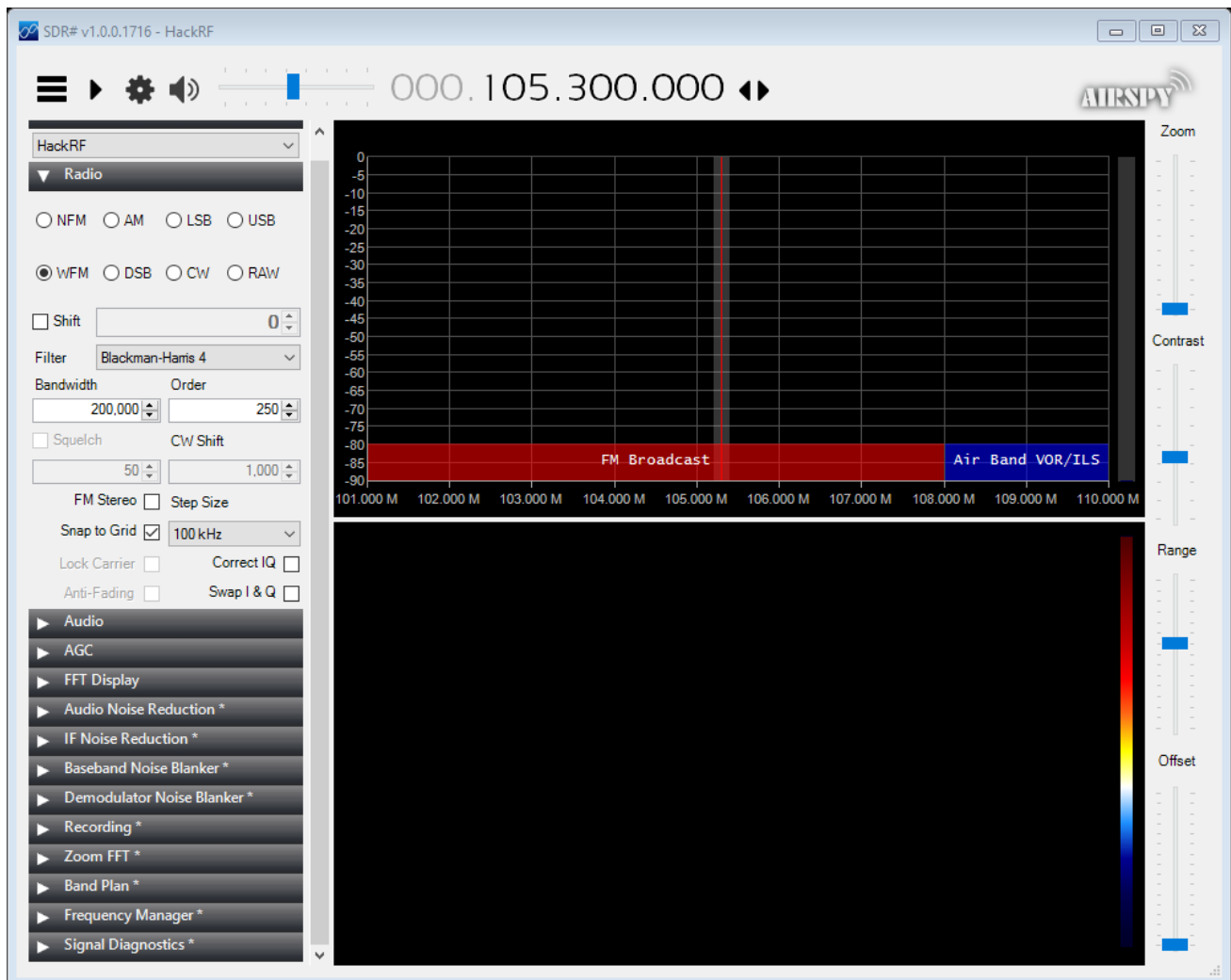
- separate large spectrum and waterfall display for input and output signals
- RF & AF spectrum and waterfall is optically zoomed to fit window width independently of FFT resolution bandwidth (RBW)
- flexible and efficient usage of the screen area from 640x480 (Netbooks) up to 8k
- extreme low-speed waterfall - helpful for pattern noise detection or short wave condition monitoring
- AM, ECSS, FM, SSB and CW demodulation
- basic transmit (TX) functionality in modes SSB, AM, FM & CW
- I/Q modulated signal pair for the TX input signal (Microphone) is produced on the TX output
- squelch, noise reduction, noise blanker, adjustable band pass filter, anti-alias filter
- automatic notch filter and up to 10 manual adjustable notch filters
- record and playback RF, IF and AF WAV files with recording scheduler
- Frequency Manager for Eibi, Ham Bands, Radio Bands, User frequency lists

- DDE client for Ham Radio Deluxe, Orbitron, WXtrack, SatPC32, Wisp and PstRotator (Howto)
- Omni-Rig support (CAT) to control additional hardware
- support for various hardware through Alberto's (I2PHD) ExtIO DLL interface
- ExtIO frequency options for IF-Adapter, Upconverter, Downconverter, Undersampling and calibration
- All HSDR program options can be stored and loaded per "profile", to ease use of different receivers
- autocorrelation and cepstrum display for demodulated audio
- some command line options with profile management

Link: <http://www.hdsdr.de/>

24.2.4 SDRSharp:

Airspy is a line of Popular Software-Defined Radio (SDR) receivers developed to achieve High Performance and Affordable Price using innovative combinations of DSP and RF techniques. The goal is to satisfy the most demanding telecommunications professionals and radio enthusiasts while being a serious alternative to both cost sensitive and higher end receivers. Airspy Radios feature world class reception quality and ease of use thanks to the tight integration with the de facto standard free SDR# software for signal acquisition, analysis and demodulation.



Link: <https://airspy.com/download/>

24.2.5 GnuRadio:

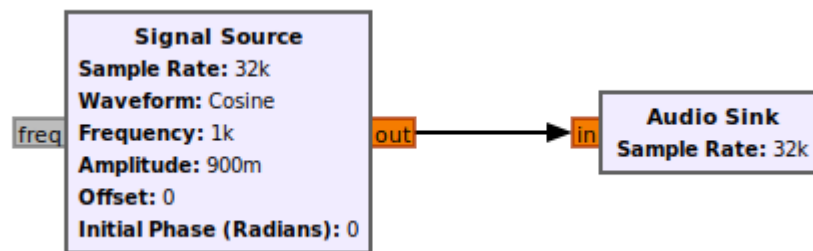
GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in research, industry, academia, government, and hobbyist environments to support both wireless communications research and real-world radio systems.

What is GNU Radio?

GNU Radio is a framework that enables users to design, simulate, and deploy highly capable real-world radio systems. It is a highly modular, "flowgraph"-oriented framework that comes with a comprehensive library of processing blocks that can be readily combined to make complex signal processing applications.

GNU Radio has been used for a huge array of real-world radio applications, including audio processing, mobile communications, tracking satellites, radar systems, GSM networks, Digital Radio Mondiale, and much more - all in computer software.

It is, by itself, not a solution to talk to any specific hardware. Nor does it provide out-of-the-box applications for specific radio communications standards (e.g., 802.11, ZigBee, LTE, etc.), but it can be (and has been) used to develop implementations of basically any band-limited communication standard.



Why would I want GNU Radio?

Formerly, when developing radio communication devices, the engineer had to develop a specific circuit for detection of a specific signal class, design a specific integrated circuit that would be able to decode or encode that particular transmission and debug these using costly equipment.

Software-Defined Radio (SDR) takes the analog signal processing and moves it, as far as physically and economically feasible, to processing the radio signal on a computer using algorithms in software.

You can, of course, use your computer-connected radio device in a program you write from scratch, concatenating algorithms as you need them and moving data in and out yourself. But this quickly becomes cumbersome: Why are you re-implementing a standard filter? Why do you have to care how data moves between different processing blocks? Wouldn't it be better to use highly optimized and peer-reviewed implementations rather than writing things yourself? And how do you get your program to scale well on a multi-core architectures but also run well on an embedded device consuming but a few watts of power? Do you really want to write all the GUIs yourself?

Enter GNU Radio: A framework dedicated to writing signal processing applications for commodity computers. GNU Radio wraps functionality in easy-to-use reusable blocks, offers excellent scalability, provides an extensive library of standard algorithms, and is heavily optimized for a large variety of common platforms. It also comes with a large set of examples to get you started.

24.2.5.1 Links:

Official site: <https://www.gnuradio.org/>

GNU Radio wiki tutorials: <https://wiki.gnuradio.org/index.php/Tutorials>

GNU Radio API manual: <https://www.gnuradio.org/doc/doxygen/>

24.3 Getting started

HSDR:

Download HSDR from here <http://www.hdsdr.de/index.html>

While you are there read the other pages, note the link to Alberto's original help and You Tube.

Execute the installation.

Later versions of Windows have security features that make altering the contents of "Program Files" difficult. A way round this might be to change the default and install HSDR somewhere different "C:\HSDR" perhaps.

Note the folder where it installs, you will usually need to find it in order to place a DLL file in there.

Look here <http://hdsdr.de/hardware.html> for DLLs and instructions for use with many radios, SDR and conventional .

Used ExtIO dll:



ExtIO_HackRF.dll



ExtIO_RTL2832.dll

Open HSDR and try to receive any local radio signal

<Add screenshot here>

SDRSharp:

Airspy is a plug-and-play device and does not require any particular driver installation on Windows Vista, 7, 8, 8.1 and 10. You just plug Airspy and Windows will download and install the right driver for you.

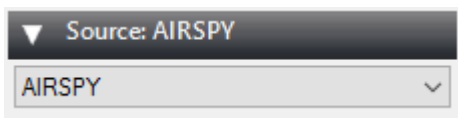
There are exceptions where the original configuration of the PC does not allow the automatic installation of this class of devices. In such case, a driver should be installed manually with the following procedure:

- Download and unzip the [WinUSB Compatibility Driver](#)

- Open the device manager and select Airspy
- Select “Update Driver” then “Browse My Computer” to the inf file

Using SDR#

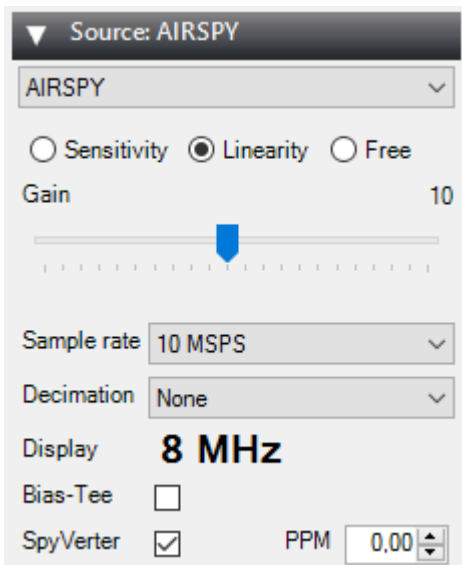
Airspy was designed by the same people who developed the SDR# software so it’s the obvious choice for running Airspy and leverage all its powerful features. First, go to the download page and get a copy. Then run SDRSharp.exe and select the “AIRSPY” front-end:



Then next step is the gain configuration. As depicted in this screen shot, there are many gain modes:

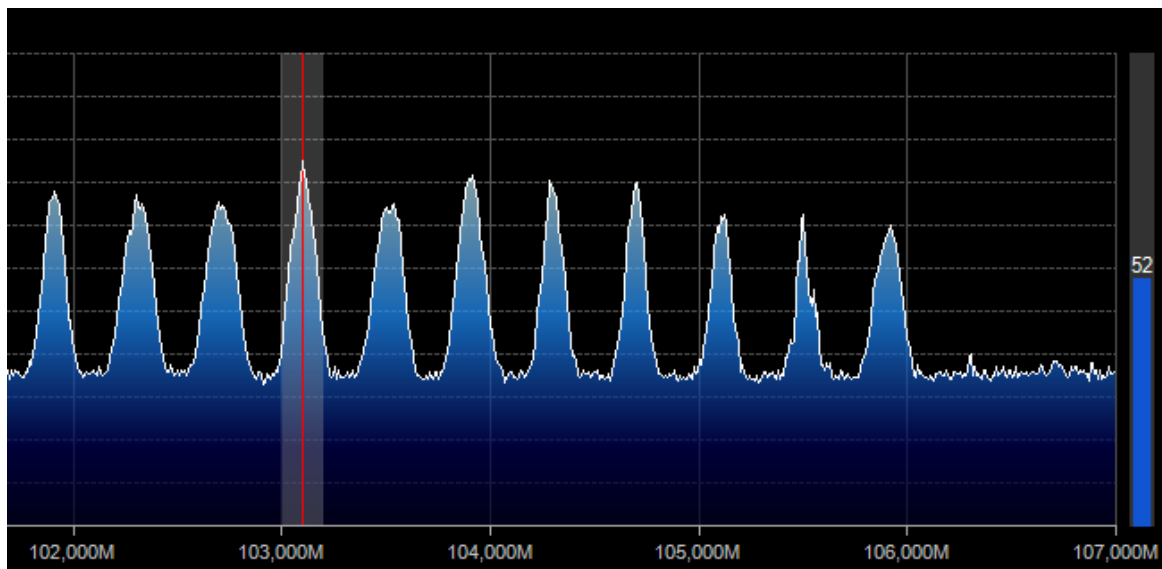
- Sensitivity
- Linearity
- Free (Custom)

The “Linearity” mode is the one you want to start with:



The following fine tuning procedure ensures you have the maximum SNR on the signal of interest while preserving the dynamic range:

- Start with the minimum gain
- Increase the gain until the noise floor rises by about 5dB
- Fine tune to maximize the SNR (the blue bar graph on the right)



In any case, you should make sure the RF noise floor just overrides the quantization noise floor of the ADC, but no more.

24.3.1 Getting Started with HackRF and GNU Radio²²

The easiest way to get started with your HackRF and ensure that it works is to use Pentoo, a Linux distribution with full support for HackRF and GNU Radio. Download the latest Pentoo .iso image from one of the mirrors listed at <http://pentoo.ch/download/>. Then burn the .iso to a DVD or use [UNetbootin](#) to install the .iso on a USB flash drive. Boot your computer using the DVD or USB flash drive to run Pentoo. Do this natively, not in a virtual machine. (Unfortunately high speed USB operation invariably fails when people try to run HackRF from a virtual machine.)

Once Pentoo is running, you can immediately use it to [update firmware](#) on your HackRF or use other HackRF command line tools. For a walkthrough, watch [SDR with HackRF, Lesson 5: HackRF One](#).

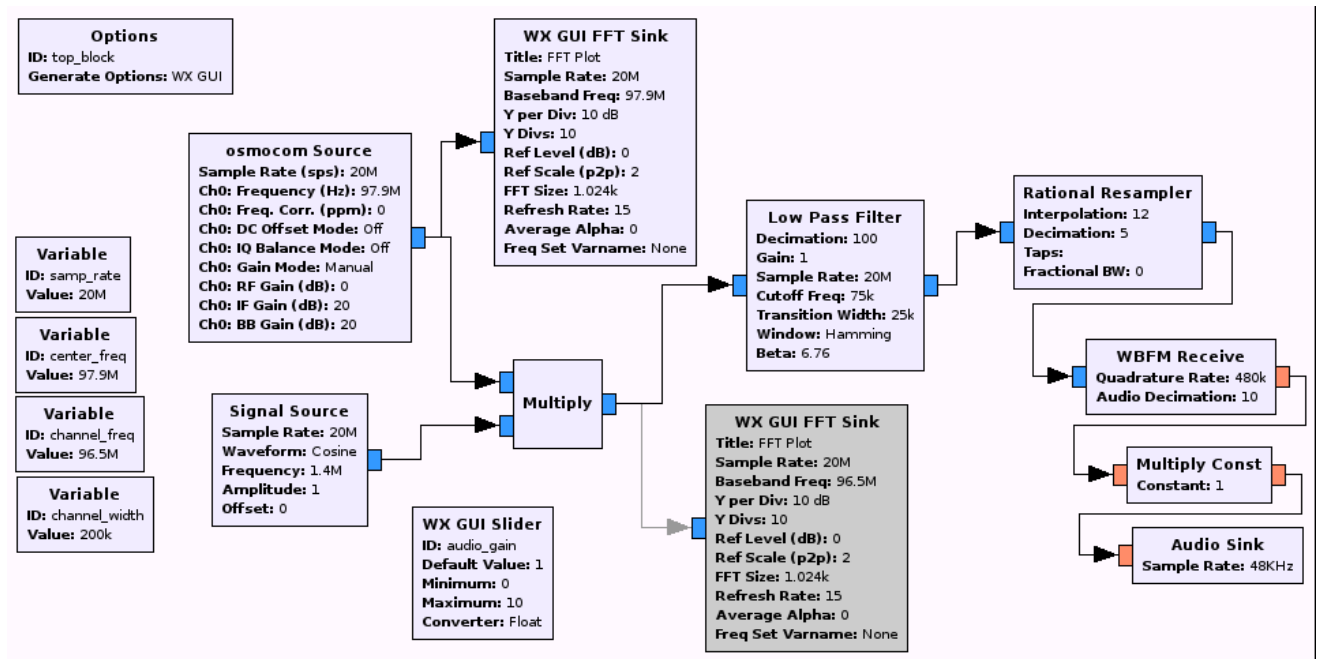
To verify that your HackRF is detected, type `hackrf_info` at the command line. It should produce a few lines of output including "Found HackRF board." The 3V3, 1V8, RF, and USB LEDs should all be illuminated and are various colors.

You can type `startx` at the command line to launch a desktop environment. Accept the "default config" in the first dialog box. The desktop environment is useful for GNU Radio Companion and other graphical applications but is not required for basic operations such as firmware updates.

Now you can use programs such as `gnuradio-companion` or `gqrx` to start experimenting with your HackRF. Try the Examples below. If you are new to GNU Radio, an excellent place to start is with the [SDR with HackRF](#) video series or with the [GNU Radio guided tutorials](#).

Try FM radio flow graph: Create a flow graph in GNU Radio Companion like the one in the video or the screenshot below. Test the flow graph by listening to a strong FM radio signal.

²² <https://github.com/mossmann/hackrf/wiki/Getting-Started-with-HackRF-and-GNU-Radio>



This flow graph is the done on the first lesson of Michael Ossmann tutorials, complete the list to be more familiar with the HackRF and gnuRadio.

<Add screenshot here for output>

24.3.2 HackRF with Raspberry Pi:

There is some files and tools should be installed on the Raspbian system of the raspberry Pi to make the ability to use HackRF. As the Raspbian is not a high performance OS, the installation will be a little bit complex.

PiSDR:

You can find on the cloud some raspbian image updated with these tools and SDR software pre-installed and ready to go such PiSDR

The PiSDR is a Raspbian based operating system for the Raspberry Pi pre-loaded with multiple Software Defined Radio software. It was created to serve as a fast and reliable bootstrap for SDR projects.

Link: <https://pisdr.luigifreitas.me/#getting-started>

Installation on RaspberryPi 3²³

What will we need

We would need Raspberry Pi 3 or Raspberry Pi Zero W 1.1, which has 64-bit ARMv7 processor. Please note that gr-gsm cannot be installed on Raspberry Pi 1 (we tested Model B), since it has ARMv6 processor. Some GNU Radio components are not supported on ARMv6 architecture.

²³ <https://github.com/ptrkrysik/gr-gsm/wiki/Installation-on-RaspberryPi-3>

On RPi we will install Debian based Linux operating system Raspbian Jessie (we used version from 2016-03-18).

Update software on Raspberry Pi

First we need to change default password with the command `passwd`. After that we need to install new updates:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Expand space on a SD card and set-up the device

Next step is to expand space on a SD card. We invoke the tool to do that with `sudo raspi-config`. In menu we select *Expand space*. We also need to set timezone, in which country we will use Wi-fi, keyboard settings (under *Internationalisation Options*). **Do not change language settings!** If system language is not English, some strange errors in Python start to appear.

Now we need to reboot the machine with `sudo reboot`.

Update firmware on Raspberry Pi

Right after that we need to update Raspberry firmware with command `sudo rpi-update`. When this is finished, we need to stop the machine with `sudo shutdown -h now` and physically plug off the power. After some seconds we power it up again.

Secure the device

Now it is good time to setup firewall, enable NTP client, change hostname... maybe we also want to set up VNC console. We assume you are able to secure the device, for some quick steps.

Do not skip this step, security is important.

Increase swap space

Sometimes you will need more memory for compiling. You can "add" new memory with increasing swap space. Open the configuration file:

```
sudo nano /etc/dphys-swapfile
```

... and look for default value in Raspbian, which is: `CONF_SWAPSIZE=100`

Change it to: `CONF_SWAPSIZE=1024` After that you need to stop and start the service that manages the swapfile on Rasbian:

```
sudo /etc/init.d/dphys-swapfile stop
```

```
sudo /etc/init.d/dphys-swapfile start
```

Now can then verify the amount of memory and swap space by issuing the following command: `free -m`

Install screen

Since compiling can take a long time, it may also be a good idea to install screen: `sudo apt-get install screen`.

Installation of software needed for gr-gsm

Install Kalibrate

First, we will install Kalibrate:

```
sudo apt-get install libtool autoconf automake libfftw3-dev librtlsdr0 librtlsdr-dev libusb-1.0-0
libusb-1.0-0-dev
```

```
git clone https://github.com/asdil12/kalibrate-rtl.git
```

```
cd kalibrate-rtl
```

```
git checkout arm_memory
```

```
./bootstrap
```

```
./configure
```

```
make
```

```
sudo make install
```

Install GNU Radio

Now we need to install GNU Radio, which is quite simple:

```
sudo apt-get install gnuradio gnuradio-dev gnu // In fact, there's no such package named "gnu", so
just install gnuradio gnuradio-dev
```

Install libosmocore

We need to compile libosmocore...

```
sudo apt-get install cmake
```

```
sudo apt-get install build-essential libtool shtool autoconf automake git-core pkg-config make gcc
```

```
sudo apt-get install libpcsclite-dev libtalloc-dev gnutls-dev libsctp-dev
```

```
git clone git://git.osmocom.org/libosmocore.git
```

```
cd libosmocore/
```

```
autoreconf -i
```

```
./configure
```

```
make
```

```
sudo make install
```

```
sudo ldconfig -i
```

```
cd
```

...and install some other things

```
sudo apt-get install swig python-docutils
```

```
sudo apt-get install gr-osmosdr rtl-sdr
```

```
sudo apt-get install libboost-dev
```



```
sudo apt-get install osmo-sdr libosmosdr-dev
```

```
sudo apt-get install libusb-1.0.0 libusb-dev
```

```
sudo apt-get install libboost-all-dev libcppunit-dev swig doxygen liblog4cpp5-dev python-scipy
```

Install gr-gsm

Now we are ready for the final step:

```
git clone https://github.com/ptrkrysik/gr-gsm.git
```

```
cd gr-gsm
```

```
mkdir build
```

```
cd build
```

```
cmake ..
```

```
make
```

```
sudo make install
```

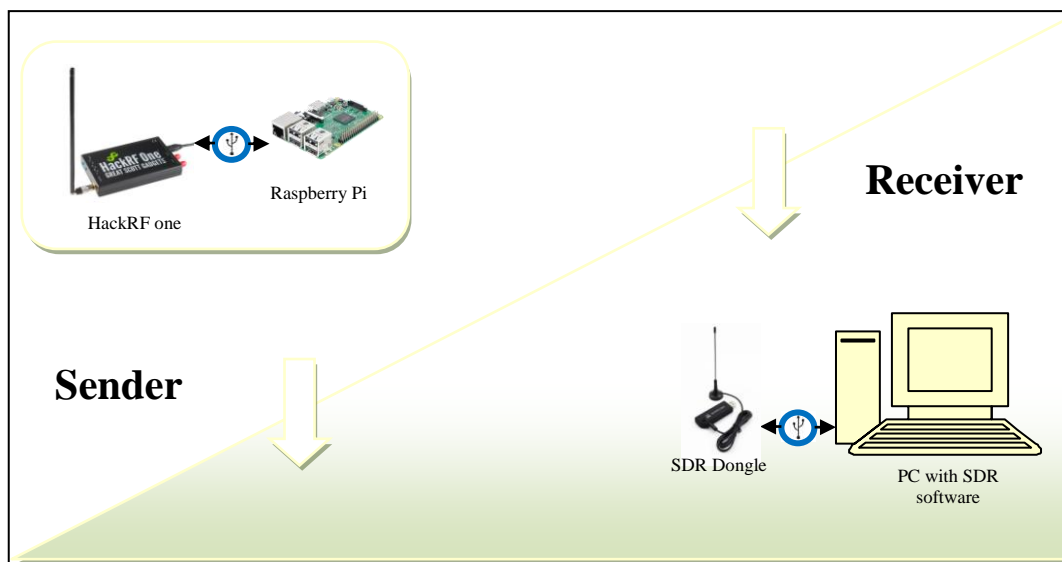
```
sudo ldconfig
```

install HackRf lib:

Step 1: sudo apt-get update -y

Step 2: sudo apt-get install -y hackrf

24.4 System 1



System 1: Send and receive system (separately)

<https://zr6aic.blogspot.com/2018/04/setting-up-my-raspberry-pi-as-bacar.html>

<https://github.com/F5OEO/rpitx>

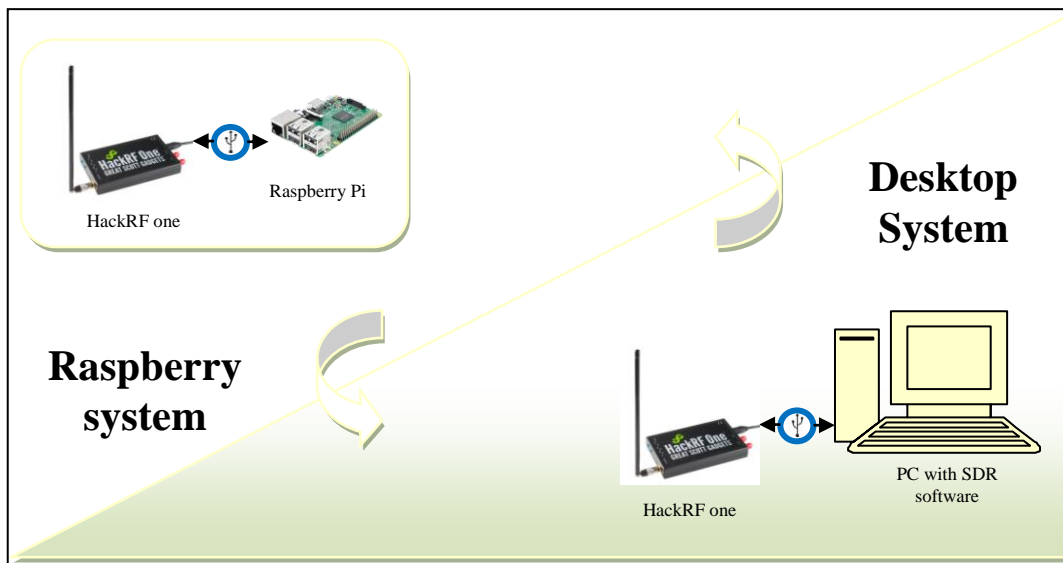
<https://www.rtl-sdr.com/transmitting-fm-am-ssb-sstv-and-fsq-with-just-a-raspberry-pi/>

<https://medium.com/@rxseger/sdr-first-project-initial-setup-node-hackrf-gnu-radio-on-linux-os-x-rpi-3-w-fm-tuner-ee16cdc8fd82>

https://www.reddit.com/r/RTLSDR/comments/dy29a3/hackrf_on_windows/

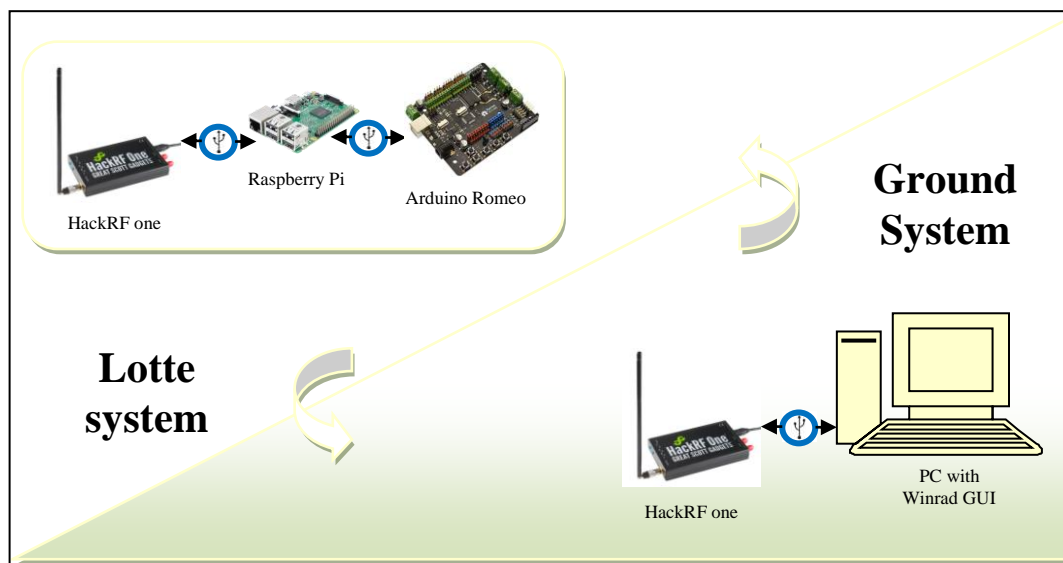
<https://github.com/mossmann/hackrf/issues/457>

24.5 System 2



System 2: Send \ Receive system

24.6 System 3

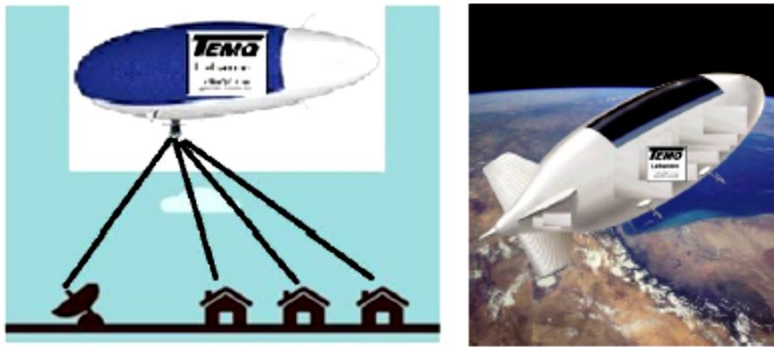


System 3: Lotte System

Aerodynamic Investigations for a High-Altitude Airship (Platform) (2017)

From:

TEMO-LEB AIRSHIP REPORT 2017



Initial Document: 18.02.2017

Last update:

Tuesday, November 28, 2017

Editor: Samir Mourad

With Contributions of

Souha Bakri (Aerodynamic Investigation),

Amena Shaker (Actuator System) and

Abdullah Mourad (Prototype Construction)

25 Basics

25.1 History of airships

In 1782, Joseph Montgolfier and his brother Etienne had the idea of using the smoke of a fire to overcome gravity. They developed the first balloon, called hot air balloon, using warm air to inflate a sphere of paper. They thus realized the first unmanned flight. Shortly afterwards, Pilâtre de Rozier and the Marquis d'Arlandes set up the first flight inhabited by man. However, the balloon remained uncontrollable in front of the winds and the ballooners therefore sought as early as 1784 to control and direct it. Some people tried to use rudders and oars, such as sailors, but without much success. The spherical shape of the balloons was soon called into question, and in the summer of 1784 the first elongated envelope was created. However, the large dimensions of aircraft remain problematic vis-a-vis the winds; the existing engines at the time were not powerful enough to counter the winds. It was thus necessary to wait until 1852 for Henri Giffard to cover a distance of 27 kilometers while controlling his aircraft (*Figure 1: The first steam dirigible, built in 1852 by Henri Giffard.*).

The feat is recognized but it cannot manage to counter the effects of the wind. Despite this unprecedented performance, Henri Giffard's aircraft is not recognized as dirigible, and it was not until 1884 to see the first airship deserve its title.

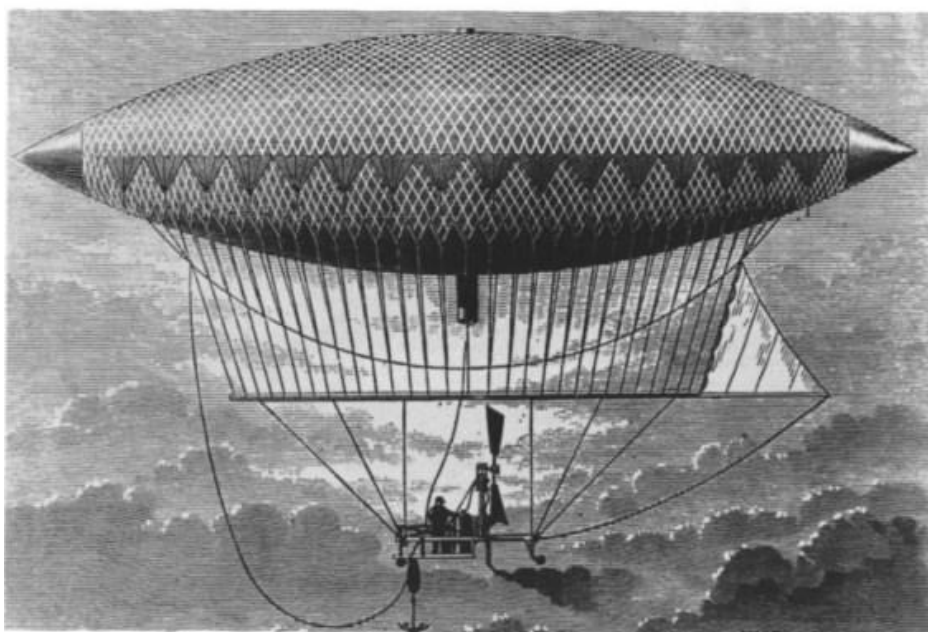


Figure 1: The first steam dirigible, built in 1852 by Henri Giffard.

After the example of Henri Giffard, many aeronauts embarked on the conquest of the perfect maneuverability of the balloons. For example, the engineer Henri Dupuy de Lôme and the Tissandier brothers who, in the years 1871 to 1874, brought many solutions to this problem.

Despite the innovations brought by these aeronauts, it was not until August 9, 1884 to see the realization of Man's dream of tame the winds. Charles Renard and Arthur Krebs carry out on board the "La France" (*Figure 2: France - First airship of the world, in Villacoublay in 1885.* aerostat what is considered the first airship flight. They make a 23-minute flight over the forest of Meudon, with the particularity of returning to their point of departure (closed circuit) thus proving the "steerability" of their aircraft.

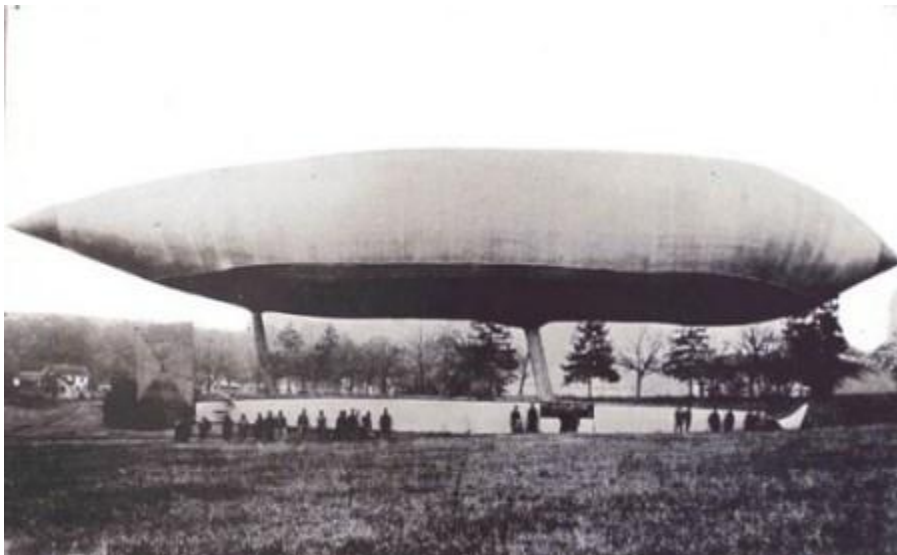


Figure 2: France - First airship of the world, in Villacoublay in 1885.

Beginning in 1898, Santos Dumont, a Brazilian by birth and French by adoption, developed many aircrafts, most of them airships. Several models stand out. From "n ° 6" in 1901, to "la Baladeuse" (*Figure 3: Santos Dumont at the controls of his dirigible No. 6: Brasil, October 19, 1901.* from the year 1903 to "No. 6" in 1901.



Figure 3: Santos Dumont at the controls of his dirigible No. 6: Brasil, October 19, 1901.

In the tradition of Santos Dumont, the Lebaudy appeared in 1902 and marked the birth of a new type of dirigibles: the "semi-rigides". The Lebaudy brothers developed a rigid hull on which the nacelle and the propulsion systems are fixed. The envelope, however, remains completely flexible.

On the German side, what became the world's largest airship company was born in 1898: Count Ferdinand Von Zeppelin deposited in 1895 a patent showing the basis of what will be the model of the Zeppelins for nearly a year century and constitutes a real break in the history of the airship: the rigid aluminum dirigible was born.

In parallel to the development of Zeppelin, the world's first airline was founded in 1909. During its five years of service, the DELAG transported some 33,000 passengers, its fleet of airships traveling 170,000 km in just over 3,000 flight hours. During those 1600 flights, no victim was to be deplored.

In 1908, Henry Deutsch de la Meurthe founded Astra and produced two flexible airships. Two years later, Leonardo Torres joins the adventure and created the series of Astra-

Torres "AT", flexible dirigibles having the peculiarity to propose for the first time a so-called "trilobée" envelope. In 1910, the Clément Bayard-II crossed the sleeve for 390 km in 6 hrs at a speed of 65 km/h.

At the announcement of the entry into the war, the European countries requisitioned their respective dirigibles. In France, the airships of companies Zodiac, Astra and Clément Bayard were used for surveillance and bombing.

At the same time, the SPIESS (*Figure 4: The Spiess, first and only rigid dirigible of French construction.*, developed in 1913 by the company Zodiac and Joseph Spiess, was produced in the Paris suburbs. It is the first and for the moment unique rigid dirigible of French construction. It is also the largest, with its 140m long and has the distinction of being composed of wooden beams.

The role of small, flexible dirigibles in maritime surveillance and anti-submarine struggle, particularly within the French and English navies, can also be mentioned.

On the German side, three companies built a large number of dirigibles: Zeppelin built nearly 67 rigid dirigibles, Parseval constructed about twenty flexible dirigibles and Schütte-Lanz also produced about 20 rigid wooden dirigibles.

During this short period, there were many developments, both in design and production and in the exploitation of airships. These intense innovations led to a race to record.



Figure 4: The Spiess, first and only rigid dirigible of French construction.

At the end of the First World War, the technologies developed made it possible to establish new standards; the interest of airships has been demonstrated and new perspectives were emerging. Each country now wanted to own the largest, and most powerful airship in support of its supremacy. Major programs were emerging:

As early as 1918, the United Kingdom began to conquer the air with its "R" series. In 1919, the R34 thus ensured the first air crossing of the Atlantic in less than 8 days of flight.

In France, the Dixmude (former Zeppelin LZ114) was ceded to the army as a war damage in 1920 but crashed in 1923.

In Italy, Umberto Nobile developed the Norge, a semi-rigid airship of 106 meters in length and departed with Amundsen to conquer the North Pole (*Figure 5: The Norge conquered the North Pole in 1926.* in 1926, reached it and flew over it on 12 May 1926. Two years later, Nobile returned towards the North Pole on the Italia dirigible, slightly longer than the Norge, but crashed on the way back. Nobile and some men survived for almost 7 weeks on the pack ice before being rescued.



Figure 5: The Norge conquered the North Pole in 1926.

In the United States, the United States, together with Zeppelin, developed several very large dirigibles, such as the first American rigid ZR-1 Shenandoah, which took off in 1923, the ZR-3 Los Angeles (1924) sisterships "USS Akron (1931-1933), and USS Macon (1933) (*Figure 6: One of the two largest US airships: the USS Macon, in 1935.* . These last two airships were the largest and most powerful American airships, true aircraft carriers flying, carrying up to 5 Sparrowhawk fighter jettisons and recoverable in flight.



Figure 6: One of the two largest US airships: the USS Macon, in 1935.

In Germany, Count Ferdinand Von Zeppelin, relayed by Dr. Eckener, still developed new airships, increasingly bigger and more efficient. Let us note the magnificent Graf Zeppelin I (LZ127) which, with its 236 meters in length, crossed the Atlantic more than 150 times from 1928 to 1940 and signed a round the world in 12 days.

The dirigibles thus dominated the heavens. This impressive aircraft became an industrial flagship, military or even a means of propaganda in some countries. It became media and famous; a real communication tool. Unfortunately, a series of accidents permanently damaged the image of the airship.

It is indeed impossible to speak of the Zeppelins without mentioning the well-known LZ129-Hindenburg, the largest of the dirigibles (245m long) which enjoyed a short career between March 1936 and May 6, 1937, crushed upon its arrival in the US, causing the death of 35 people among the 97 on board.

Although highly publicized and considered the most serious airship accident, it is not the most deadly. Nevertheless, this crash marks the spirits durably.

Indeed, like any development of an innovative product, that of the airship has not escaped a period of maturity which has resulted in a wave of accidents and more or less serious incidents. Thus, in 1920 alone, the USS-Roma crashed, causing the death of 34 people. That

of Diksmuide causes the death of 52 people. Finally the one of the R38 bore 44 families. Then, on October 1, 1929, the British rigid dirigible R101 crashed in France near Beauvais causing the death of 48 people including the British Minister of Transport. This accident marks the end of British rigid dirigibles.

On the American side, the crash of the USS Akron in 1931, causing the death of 73 people, marks the spirits and that of the Hindenburg, the largest of the airships and true flagship of the German firm Zeppelin (*Figure 7: One of the largest German airships: the Graf Zeppelin I.*, definitely signs the stop of the developments of the great dirigibles.



Figure 7: One of the largest German airships: the Graf Zeppelin I.

This series of accidents, however, is not the only cause of the decline of the dirigibles. Indeed, the development of new aircraft, such as planes, helicopters, rockets and even space conquest gradually concentrate all efforts, budgets and fantasies.

Thus, during the Second World War, airships were gradually abandoned to the profits of planes, faster and more discreet, even though small, captive, flexible airships were used to protect the battalions during landings.

Nevertheless, the US Navy continues to develop airships in large quantities to constitute an armada of patrollers-escorts for Navy ships. These dirigibles are flexible, of relatively small size, but are produced in very large quantities.

After the Second World War, priority was not given to spending on the development of new aircraft. In addition, aircraft played a key role in the fall of the Nazi regime. The

dirigibles, considered too dangerous and without real commercial or military interest, are gradually abandoned.

We note, however, the development of the American firm Goodyear, which continued to innovate and produce flexible airships: in 1954, the ZPG-2W beats the record of endurance of a flight with 11 days and more than 15,000 km on the meter. In 1958, Goodyear built the largest flexible airship (129m long!). Note that Goodyear still exists today and has just relaunched the historical collaboration that links it to Zeppelin by taking possession of the latest version of the Zeppelin NT (ZLT-101).

However, the return of airships from the 1960s remained very limited, and limited to flexible dirigibles, the aircraft then presenting no real economic interest.

Beginning in the 2000s, the airship regained favorable winds; with many projects coming to light all over the world. The renaissance of Zeppelin, with its NT07 "New Technology" (*Figure 8: The Zeppelin "New Technology", in flight for a decade.* produced in 5 copies for civil applications of tourism and scientific missions, and of which a new version has just been certified and delivered to Goodyear.



Figure 8: The Zeppelin "New Technology", in flight for a decade.

At the same time, the year 2000 marked the arrival of a new market, that of heavy hauliers, like the gigantic German CargoLifter project, abandoned in 2005, for technical reasons. On the American side, the DARPA-funded Multi-intelligence Vehicle Long-Endurance Multi-Vehicle project has created several projects: Northrop Grumman and Hybrid Air Vehicles

have teamed up to develop the Airlander, prototype made its first flight in 2012 and is expected to return soon; Aeroscraft produced and stolen its Pelican, which is now dismantled for lack of funds; and Lockheed Martin developed the P-791, a capability demonstrator, whose development now allows him to embark on a new, civilian, heavy-duty 20T load carrying program, the LMH-1.

However, no heavy-lift carrier project has yet taken off for the time being.

We will further develop the current projects in a forthcoming article devoted entirely to today's airships.

Finally, the airship enjoyed a period of splendor following a rapid development, setting it up as a true pioneer of the aeronautical world. The economic reality, the lack of technological maturity and the appearance of the heaviest air, however, have gradually erased the blimp of the collective imagination. Like its history, the airship enjoys a controversial notoriety, between fear and dreams. Today, this mythical aircraft is a great absentee of the very varied aviation sector, and it is a shame. ^[1]: <http://www.portail-aviation.com/2015/07/dossier-dirigeable-episode-1-lhistoire-des-dirigeables-pionniers-de-laeronautique.html>

25.2 Evolution of airship

Since the past ages, the high ambition of human promotes him to fly. The science of aviation has developed gradually.

25.2.1 Hot air balloons

The first attempt was based on heating of the air without a motor. The balloon consisted of the envelope, a burner and the gondola.



Figure 9: First ever of hot air balloon

The hot air balloons adopted two principles to work:

➤ **The principle of cold and of hot air**

The air is composed of the invisible particles: “molecules”. When the air is cold, the molecules approach together and take a small volume comparing to the case of hot air where the molecules move away from each other and the volume is larger.

For this reason, the mass of hot air is lighter than cold air. When the molecules of air contained in the envelope is heated, the air in the balloon becomes lighter than the air in the atmosphere, allowing the balloon to lift.

In order to descend the balloon, the air in the envelope must be cooled and it is also possible to open the valve located at the top of the balloon. In this way, the cold air replaces the hot, and the mass of interior air increases. Consequently, the balloon can descent. ^[1]

➤ **Principle of Archimedes**

When a body is immersed in a fluid (gas or liquid), it is held up by a force equal to the weight of the displaced fluid.

In order to lift an airship, the buoyant force must be larger than the weight of the displaced air.



Figure 10: Hot air balloons

The manufacturing of hot air balloon has developed and the design of his envelope, his burner and his gondola work out big success.

➤ **Development of envelope**

By time, the fabrication methods used to produce balloons have changed. The aim was to achieve the following characteristics:

- ❖ Flexibility
- ❖ Resistance
- ❖ Tightness: It is necessary to achieve this characteristic because the gas lighter than air as helium has small molecules which can leak out of the balloon. So to be able to undergo long flights, the balloon cover needs a very good seal.

a. “Baudruche”(Type of tissue)

The “Baudruche” is a membrane made from the intestine of beef and mutton. This tissue is very light and waterproof but it has low durability and high cost.

b. Cotton or Silk

Cotton or silk can provide the mechanical strength. They coated with rubber or varnish to ensure the tightness.

c. Polyamide(Nylon) and Polyester

The polyamide (Nylon) and Polyester have used a lot in this days because they have a light weight and good resistance. They are coated with polyurethane or silicone for protection to the ultraviolet rays. [2]

➤ **Development of burner**

The burner is the engine of the hot air balloons. It heats and propels the air in the envelope and contributes to lift the balloon.

In the first time, the damp straw and wood were used as fuel in the burner. After that, the coal and the oil derivatives were used.

The current balloons used the propane in the burner. In the basket, many cylinders contain propane. It is highly compressed in canisters and it is entered into the burner in liquid form.

When the burner is started up, the flame burns, the propane heats and transforms from a liquid to a gas. The gas makes for a more powerful flame and an overall more efficient fuel consumption. [3]

➤ **Development of gondola**

Originally, the gondola or the basket was made of “Wicker” (material was used at the time especially to make baskets).

Since that time, the materials have evolved. “Rattan” is also a material which used after, his advantage is more solid and less susceptible to external elements but that is heavier than “Wicker”.

Both of materials still used, they have more advantages: solidity, lightness and flexibility. [2]

The modern materials used to made baskets are from “Kooboo” and “Palambang” cane. There are many characteristics: sturdy, flexible and relativity light weight. The cane is very strong even more than aluminum or some composite plastics. [2]

25.2.2 Dirigible or Airship

The dirigible is newer than hot air balloon. It contains gas which is lighter than air, this gas is used instead of hot air. Three types of airship existed and they developed gradually:

1. Non-rigid airship(blimp):

It is composed of ballonets, rudders (for directions), elevator flaps, gondola and a small engine.



Figure 11: Blimp (Non rigid airship)

This type developed and had an amelioration, it is named:

2. Semi-rigid airship:

The improvement of this dirigible: instead of the ballonets, it used gas bags, and it used big engine and a keel.

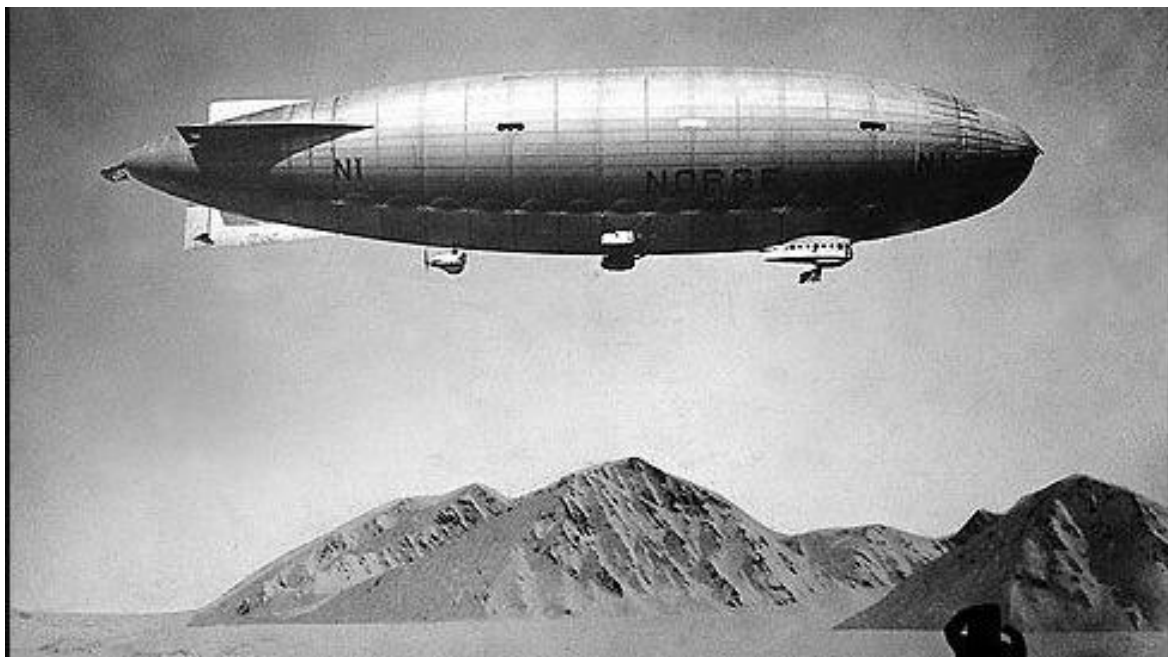


Figure 12: Semi-rigid airship

3. Rigid airship: More than the previous type, it had a rigid structure.



Figure 13: The Graf Zeppelin LZ-127 the most successful rigid airship

25.3 Evolution of the airship industry

In 1784, Jean-Baptiste Meusnier suggested a design for an airship of ellipsoid form. This design consisted from: rudder, elevator and three large aircrews without lightweight powerful engine.

In 1852, Henri Giffard succeeded to apply steam-engine technology to airships. With a single propeller driven by a three horsepower engine, his airship flew 17 miles. ^[4]

In 1872, Paul Haenlein was powered an aircraft by an internal combustion engine, the first attempt was used such an engine to power an airship. ^[5]

In 1900, the Zeppelin airships became the most famous dirigible. During World War I, the Germany army was used some of these airships as bombers.

In 1920s and 1930s, the United States and the Britain also were made airships, mostly imitating the original Zeppelin design. ^[5]

25.4 The end of airship

Series of accidents were happened and ended the golden age of airship. In particularly, the Hindenburg disaster at Lakehurst, New Jersey, 6 May 1937. The burn of this airship named " LZ 129 Hindenburg" was filled hydrogen ,killing 36 persons and becoming the most well-known and widely remembered airships disasters of all time. The public's confidence in capacities of airships was shattered.

The development and application of airplanes has declined also the use of airships. The century of airship finished and start new future of aircraft. ^[6]

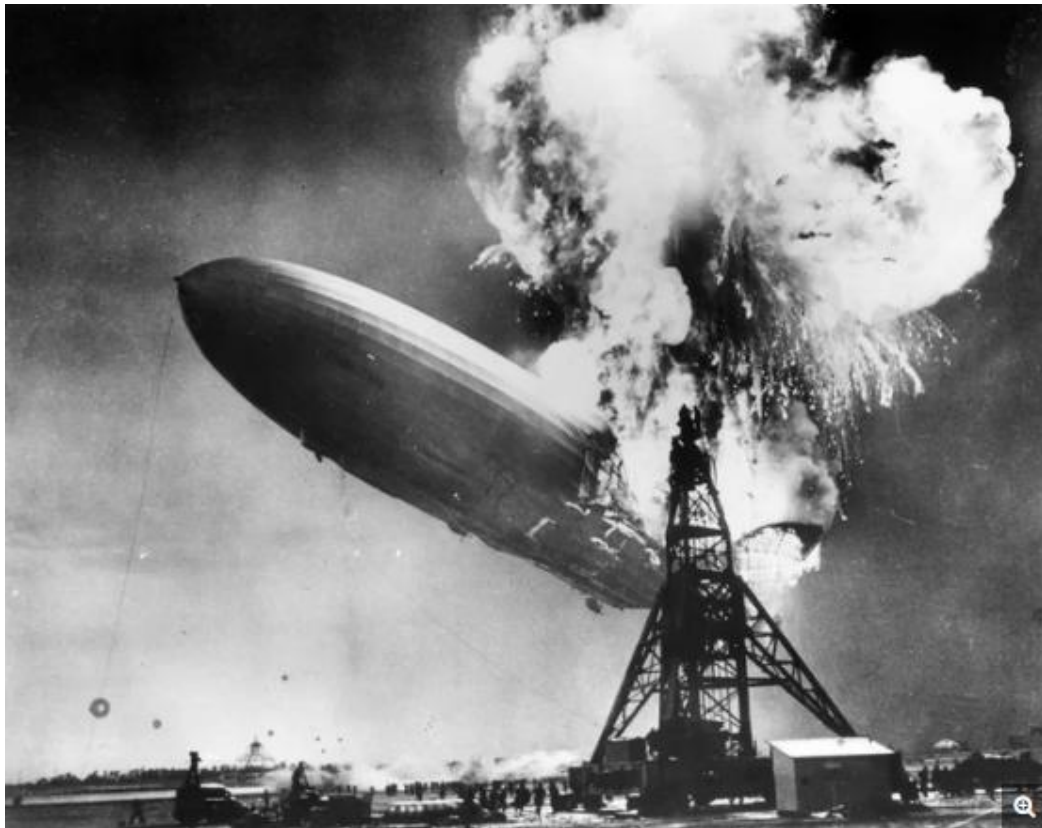


Figure 14: The Hindenburg disaster

25.5 Return of airship

After a pause of about hundred years, the airship returned to life at the end of the twentieth century. Thus the hobby of the airship has aroused the curiosity of many, and has opened up new horizons for this invention to move from sport to a means that allows man to leave the Earth and seek other uses, especially in the science field, such as surveillance and telecommunication.

If we have a traffic jam, we can use the airship as a solution to detect the main point of traffic, using the new technology people receive messages to avoid the busy roads. Consequently, people save their time.

In the far forests, if we have fire or natural disasters, we can use the airship to detect these actions as a satellite.

Also, the airships can used for tourism, they are very attractive means for transport.

Our study concerns to create a design of a rigid airship, its application will be in the field of telecommunications. Indeed, the problem of internet weakness and its high price will be solved by such a project whose objective is to distribute Internet in the far regions without having to install a complete network. Hence the objective of this thesis is to design a rigid airship which can reach a

high altitude. To appropriate this design, we need to do a study of aerostatic of airship, to draw the design on “FreeCAD” and to study the all parts of this dirigible.

25.6 Archimedes principle

The main source of lift in an airship is the bouncy force or the static lift. The bouncy force is based on Archimedes principle: if a body is immersed in a fluid, in this case the fluid is the air, it experiences a force proportional to the volume of the displaced air in the opposite direction of its weight. When the density of the gas contained in the airship is less than the air, that force is substantial.

25.7 Types of airships

25.7.1 Non-rigid Airship

The non-rigid airship named also “blimp” maintains his shape and his structural integrity using higher internal pressure from its lifting gases. ^[4]

When the airship ascends, the lifting gas expands and when it descends, the lifting gas contracts. Then, the envelope would lose shape and become unmanageable. That is happening because this type of airship doesn’t contain a framework.

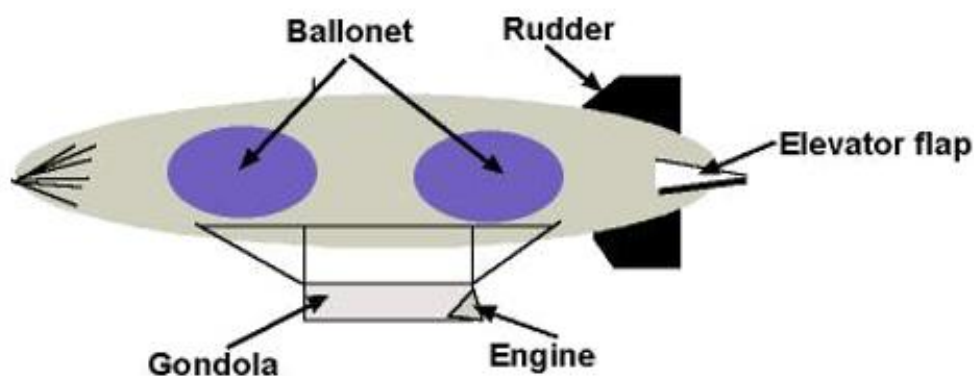


Figure 15: Parts of non-rigid airship



Figure 16: Blimp during filling gas

The development of aeronautic science was emerged a new type of airship:

25.7.2 Semi-rigid airship

Semi-rigid airships are similar to blimps in that they have no internal frame to support their envelopes. They do have, however, rigid objects on them that give them some backbone. A stiff keel runs along the length of the airship for distributing weight and attaching fins and engines. The keel also provides structural integrity during flight maneuvering. Similar to non-rigid airships, the shape of the hull is maintained largely by an overpressure of the lifting gas. Light framework at the nose and the tail may also contribute to the hull's outer shape. [4]

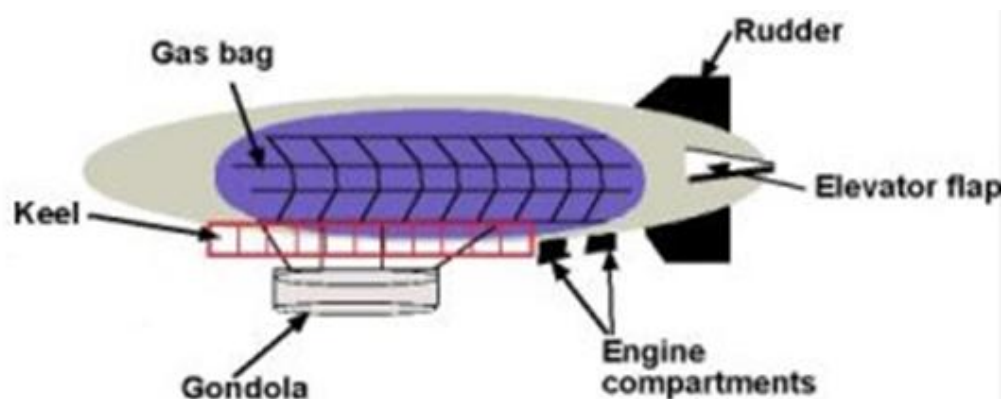


Figure 17: Parts of semi rigid airship

25.7.3 Rigid Airship

Semi-rigid and non-rigid airships maintain their shapes by the internal pressure of lifting gases. But rigid airship is different, it has an internal structure framework which it gives their shape form. It has an outer envelope to recover the framework.

Rigid airships have be built by a size larger than semi-rigid and non-rigid airships because there are no possibility of kinking in the hull due from aerodynamic forces and moments. Multiple gas bags containing the lifting gases are filled inside the internal framework of dirigible.

Splitting the gas in multiple bags instead using a single large bag is more safety and minimizes to happening a catastrophe. ^[4]

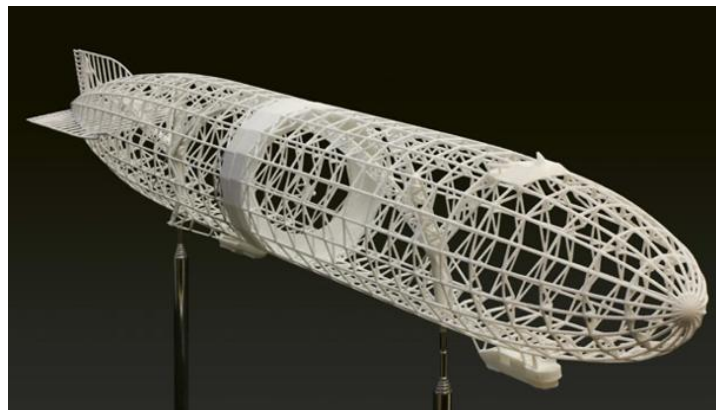


Figure 18: The internal framework of rigid airship

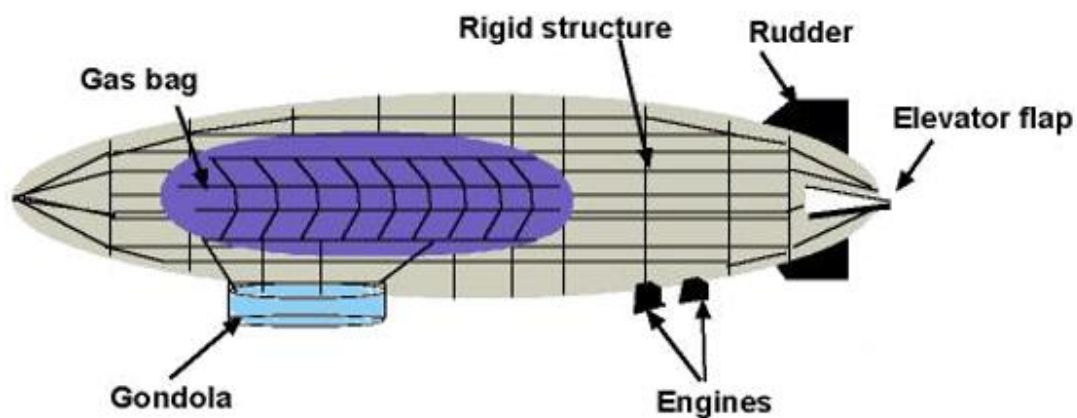


Figure 19: Parts of rigid airship

25.8 How do rigid airship work?

In order to rigid airships get off the ground, fly and descend, different gases are used by rigid airships. Today, to rise the lighter-than-air craft, the helium gas is used instead to hydrogen. Helium is more expensive than hydrogen, but it was adopted because it has more properties. It is inflammable contrary the hydrogen which causes the infamous Hindenburg accident.

The airships, filled by Helium, load ballonets (tanks of air). When the pilot opens the valves of air, a positive bouncy is created, and consequently the dirigible elevates because air is heavier than helium.

The pilot controls the airship in flight by rudders and elevators, when it becomes in the sky, like a submarine under water. The rudders are used to steer the airship and the elevators are used to ascend and descend and throttling the engine to angle it into the wind. The engine provides forward and reserve thrust.

The air pressure outside the lighter-than-air craft deceases, at higher altitudes, consequently the helium in the gasbags expands. The pilot pumps air into ballonets in order to maintain pressure.

To descend the airship, this technique is also employed, the ballonets are filled by air. The air is heavier than Helium. That is created a negatively buoyant and therefore sinks lower in the sky or bring it in to land. [7]

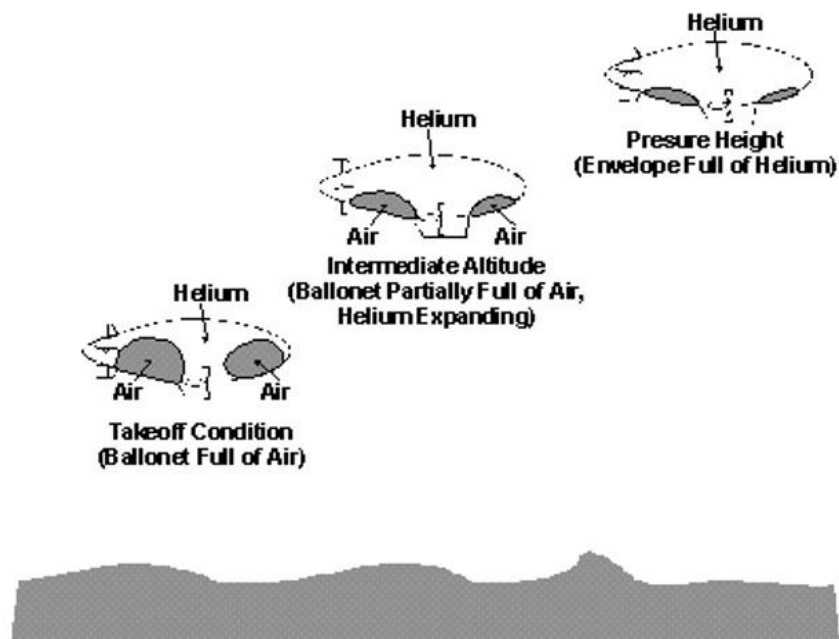


Figure 20: Airship principle of operation

25.9 Why do we use Helium not Hydrogen?

The performance of Helium in airship is better than hydrogen. In order to inflate an airship, the use of Helium instead of hydrogen reduces the cruising range from 30 to 40 % and reduces the total lift from 10 to 15 %. This is a disadvantage for Helium but it can be justified. Helium has absolute safety against the risks of fire that is expected with hydrogen. In the United States, Helium finds in large quantities. It is very expensive, but the price is significantly lowered, because the quantities of production have increased.

➤ Characteristics of hydrogen and Helium:

❖ Hydrogen

- Lightest element in the earth
- Inexpensively
- Easily to obtain
- Flammable. For this disadvantage, it is unacceptable for manned airship operations.

❖ Helium

- Scarce
- Expensive
- Non-flammable. This advantage makes it the only practical lifting gas for manned airship operations. ^[8]

25.10 Control of airships

The Control of the airship differs from the aircraft's one. It can be divided into two categories. While the control of airplanes is performed by one pilot; in airships two pilots are utilized, one for direction, one for altitude.

Both pilots must recognize the flight pattern of each and constantly alert each other to mutual assistance; to give an effective performance. The pilot should organize the route to meet the needs of the situation. There are too many coordination cases, but capable pilots have little difficulty in achieving the expected results.

Directional: as indicated in the previous paragraph, the pilot is responsible for monitoring the course of the airship in a horizontal plane. On flights throughout the country, his problem is solved with regard to the course required by the mission of the dirigible. Once the course is established, the dirigible will hold its own. But when it is affected by external powers such as gusts, the rudder must work in the opposite direction to overcome it. While flying in a very stormy air, it is impossible to prevent lace, but a good pilot can prevent the amplitude of oscillations from exceeding a few degrees. Then, as the storms also strike on both sides, the average direction of the dirigible will be the desired direction. It is essential that the pilot has a clear conception of the reaction to control the rudder of the airship at the turn. When you need to move to the right, for example, the rudder is placed to the right. Immediate effect of this rotation is to produce a force to the left acting on the right side of the rudder. This force to the left has a dual effect. In the first place, it gives a moment around the center of gravity inclined to turn the nose to the right. In the second place, it moves the whole airship to the left. When the airship moves to the left, and the nose turns to the right, both motions combine to make the air fall to the left of the envelope and to turn the nose still further to the right.

Once this has been done during a period, the pressure on the left side of the nose becomes equal to that on the right side of the rudder and the resulting total pressure is zero, but as the force is applied to the front, there is a moment of rotation that tends to continue twisting to the right.

As the motion goes further, the force on the left side of the envelope becomes larger than the force on the right side of the rudder, and there is a gravitational force to the right of the skill to begin to scroll to the right. If you leave the rudder hard or if it is rotated to neutral, this turning to the right will continue, to check the cycle, it is necessary to put the rudder on the left side of the envelope.

The shift radius is controlled by instantaneous damping on the envelope, which is more important for the rate of high-speed propagation than those with a low ratio. This should be one of the main concerns of the driver whenever he controls a new type of airship to identify himself with his turn circle. Otherwise, he may try to maneuver when the space limit is insufficient.

The first effect on the rudder mode on the right is to move the airship slightly to the left so that if the airship was stolen near the right side of the wall or any other obstruction, it would not be wise to put the rudder On the right to move to the right and away from the obstacle because the immediate action for such work would be to pay the airship in the wall.

Sometimes, when flying through foggy weather, an obstacle will suddenly loom up in front of the airship. So the pilot must first put the rudder to disperse the nose of the airship and then completely reverse the rudder in order to miss the obstacle. [2]

26 Aerodynamic investigation of a high altitude airship

In this chapter, we speak of the aerostatic of airships which is based on the buoyancy force. We also talk about the variation of the density, the pressure and the temperature as a function of the altitude. This variation is plotted on the "MATLAB" software. Then, we study the influence of these parameters on the airship.

In this chapter too, we suggested a design of "TEMO-Leb airship" drawn on "FreeCAD" (FreeCAD is a software similar to AutoCAD, but it is more complicated and needs more time to do drawing by comparing the AutoCAD. Despite this disadvantage, we obtain the same quality.)

In this chapter, the necessary calculation is made to determine whether the "TEMO-Leb airship" is able to reach the desired altitude. We talk about the problems that confront us and prevent us from applying this theoretical study.

Next, we propose a new conception which has been considered as a solution. This design which has new dimensions, will rise to a lower altitude. It calls "Low altitude test TEMO-Leb Airship". We expect its implementation.

26.1 Aerostatic of airship

To define the buoyant lift capability and to determine the maximum attainable altitude by the airship, it is necessary to calculate the volume of the hull.

Depending on aerostatic principles, we obtain some relations between mass, volume and gas densities.

26.1.1 Buoyancy force

The buoyancy force (F) generated by an airship is equal to the weight of the displaced air subtracting the weight of the lifting gas (Helium). Noting the volume of displaced air is equal to the volume occupied by the helium.

$$F = V_N \cdot (\rho_{A0} - \rho_{He0}) \cdot g \quad \text{Eq.1}$$

V_N : The net volume of displaced air.

ρ_{A0} : The density of air at sea level.

ρ_{He0} : The density of Helium at sea level.

g : The acceleration of gravity.

The amount of lift available to counteract the weight of the airship structure and payload is represented by the buoyancy force F .

26.1.2 The relation between altitude and density

The density varies with altitude. The relation is inversely proportional: when the altitude increases, in opposition, the density decreases.

The conventional state of atmosphere is defined by equations of International Standard Atmosphere (ISA). The equation below is suitable until an altitude of 11 000 m:

$$\rho_H/\rho_0 = [1-(H/44\ 300)]^{4.256} \quad \text{Eq.2} \quad [9]$$

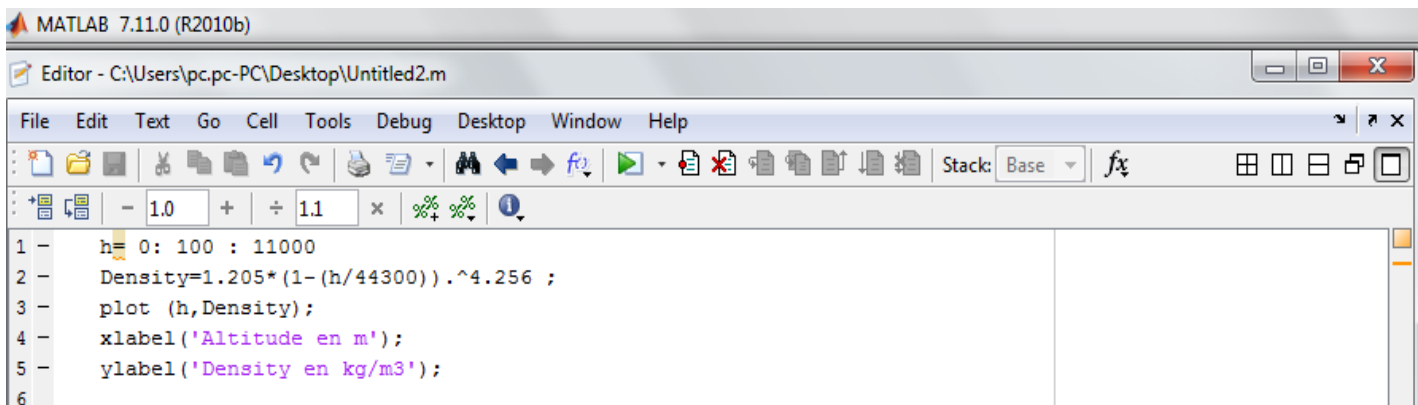
ρ_H : Density of the air as a function of altitude (kg/m^3).

ρ_0 : Density of the air as a function of altitude (kg/m^3) at sea level, $\rho_0=1.205\text{ kg/m}^3$.

H : Altitude (m).

MATLAB is used plotting the variation of density as a function of altitude using Eq.2. This program allows matrix manipulation, plotting of data and functions, creation of user interfaces, implementation of algorithms and interfacing with programs written in others languages, including “C”, “C++”, “Fortran”, “Java” and “Python”.

- ✓ Writing the code in “MATLAB”:



```

MATLAB 7.11.0 (R2010b)
Editor - C:\Users\pc.pc-PC\Desktop\Untitled2.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
fx
- 1.0 + ÷ 1.1 x %> %<
1 - h= 0: 100 : 11000
2 - Density=1.205*(1-(h/44300)).^4.256 ;
3 - plot (h,Density);
4 - xlabel('Altitude en m');
5 - ylabel('Density en kg/m3');
6
    
```

Figure 21: code of Eq.2 in MATLAB

Results and plotting of this function is shown in the next chapter.

26.1.3 The relation between altitude and pressure

The atmospheric pressure decreases rapidly with the altitude. The conventional state of atmosphere is defined by equations of International Standard Atmosphere (ISA). The equation below is available till altitude of 11 000 m:

$$P/P_0 = [1-(H/44\ 300)]^{5.256} \quad \text{Eq.3} \quad [9]$$

P : Pressure of the air as a function of altitude (Pa).

P₀: Pressure of the air as a function of altitude (Pa) at sea level, P₀=101 325 Pa.

H: Altitude (m).

MATLAB is used for plotting the variation of pressure as a function of altitude using the equation “Eq.3”.

- ✓ Writing the code in “**MATLAB**”:

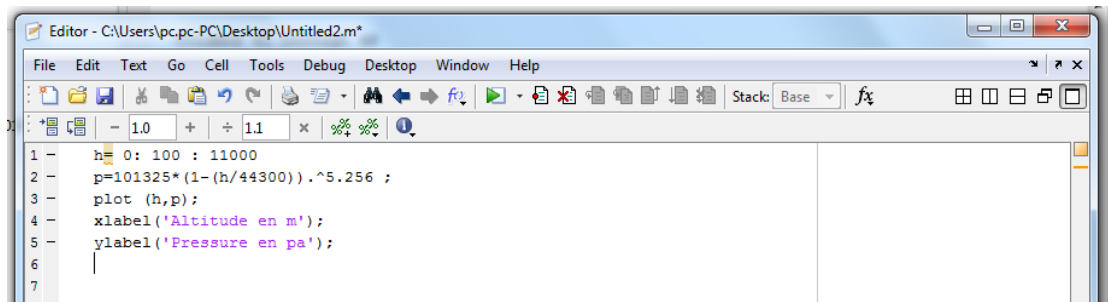


Figure 22: code of Eq.3 in MATLAB

Results and plotting of this function in the next chapter.

26.1.4 The relation between altitude and temperature

The temperature decreases with the altitude. The conventional state of atmosphere is defined by equations of International Standard Atmosphere (ISA). The equation below is available till altitude of 11 000 m:

$$T_H = T_0 - 0.0065 * H \quad \text{Eq.4} \quad [9]$$

T_H: Temperature as a function of altitude (°c).

T₀: Temperature as a function of altitude (°c) at sea level, T₀=15°c.

H: Altitude (m).

MATLAB is used for plotting the variation of pressure as a function of altitude using the equation “Eq.4”.

- ✓ Writing the code in “**MATLAB**”:

```

1 - h=0 : 100 : 11000;
2 - t=15-(0.0065*h);
3 - plot(h,t);
4 - xlabel('Altitude en m ');
5 - ylabel('Temperature en celsius ');
    
```

Figure 23: code of Eq.4 in MATLAB

The results and plotting of this function in the next chapter.

26.1.5 Pressure Altitude

It is necessary to define the term “**Pressure Altitude**”. At this point, the net volume is maximum “ V_{max} ” and no further expansion of the lifting gas volume is possible.

26.1.6 Influence of pressure, temperature, density and volume on the airship during its rise

a) Under “Pressure Altitude”

Supposing both gases, air and Helium have the same temperature and pressure, but their densities change with altitude.

A slight pressure differential is necessary, but this difference is very small. For the purposes of this analysis, it is considered null.

When the airship rises, the density of Helium decreases along with the atmospheric density. We know the relationship between density, mass and volume:

$$\rho = m/V \quad \text{Eq.5}$$

Therefore, since the mass of the Helium remains fixed, the volume V_N must increase. In order to regulate the internal pressure, two ballonets contain air are located inside the hull which expand and contract. By this way, the variation in internal lifting gas volume is achieved.

At the sea level, assume that is the airship’s launch altitude, the value of density is maximum. Also, the volume of ballonets expand to maximum while the net volume is minimum.

When the airship begins to rise, the ambient density and pressure both decrease, and air is automatically ejected from the ballonets to match the falling pressure.

At a given point, the ballonets must be empty completely. At this point, the volume of the lifting gas can't expand. The value of the net volume now is maximum " V_{max} ".

By summarizing, under the point "**Pressure Altitude**", we have:

- ❖ The differential of pressure between lifting gas and atmosphere is null: $\Delta P=0$.
- ❖ The differential of temperature between lifting gas and atmosphere is null: $\Delta T=0$.
- ❖ The density of lifting gas and atmosphere decrease, this variation affects an increase of net volume. The equation Eq.5 is available up to the "**Pressure Altitude**".

b) Above "Pressure Altitude"

When the airship continue to rise, the density of lifting gas and atmosphere also continue to decrease, but the volume of lifting gas remains constant.

The differential of pressure now isn't null, it starts to increase. An overpressure is created, and consequently the differential of pressure will increase so a rupture will be exist in the exterior structure of airship. The equation Eq.5 becomes unavailable.

26.1.7 Sizing the airship hull

To reach the Equilibrium, the bouncy force must be equal the weight of the airship structure (framework, external fabric and gasbags) and payload (propeller, control system...).

The weight is equal:

$$P = (m_s + m_p) * g \quad \text{Eq.6}$$

P: The weight of the structure and payload (N/Kg).

m_s : The mass of structure (kg).

m_p : The mass of payload (kg).

g: The acceleration of gravity (N/Kg).

When the airship rise, up to the pressure altitude, the net lift remains constant, because the atmospheric density changes at the same rate as the density of the lifting gas.

The density ratio is equal to:

$$\sigma = Q_A / Q_{A0} = Q_{He} / Q_{He0} \quad \text{Eq.7}$$

σ : The density ratio.

ρ_A : The atmospheric density at a given altitude.

ρ_{A0} : The atmospheric density at the sea level.

ρ_{He} : The density of Helium at a given altitude.

ρ_{He0} : The density of Helium at the sea level.

At a given altitude, the Eq.1 will be:

$$F = V_N \cdot \sigma \cdot (\rho_{A0} - \rho_{He0}) \cdot g \quad \text{Eq.8}$$

At the equilibrium and the point “**Pressure Altitude**” ($V_N = V_{max}$), we have:

$$\text{Eq.6} = \text{Eq.8}$$

$$(m_s + m_p) \cdot g = V_{max} \cdot \sigma \cdot (\rho_{A0} - \rho_{He0}) \cdot g$$

$$V_{max} = (m_s + m_p) / \sigma \cdot (\rho_{A0} - \rho_{He0}) \quad \text{Eq.9}$$

The equation “Eq.9” gives us the maximum volume of the airship hull which is needed to rise the airship. This volume is based upon the ratio of the density which is varied with altitude, mass of the structure and payload.

The mass of structure includes the mass of the gasbags, the mass of the airship framework and the mass of the envelope or the external skin.

26.2 TEMO-Leb Airship

In this project, we suggest an airship called “**TEMO-Leb Airship**”. This airship will distribute Internet between Turkish and Lebanon, at altitude 7 km.

The new idea in this project is that the size of “**TEMO-Leb Airship**” is smaller than other airships which take off at a high altitude.

The “**TEMO-Leb Airship**” dimensions: its length equal to twenty meters and a diameter equal to four meters.

A theoretical study is needed to calculate the volume required for “**TEMO-Leb Airship**” to reach an altitude of 7 km.

- a) First, it is important to calculate the volume of the airship based on the proposed dimensions.
- b) Secondly, it must be calculated V_{max} .
- c) Thirdly, it must be investigated whether this airship is available to reach the altitude 7 km with these dimensions. This is discussed in the next chapter.

To get V_{max} , it is necessary to calculate the mass of structure includes the mass of framework, the mass of external fabric and the mass of the gasbags. It must be know the mass of payload.

Step a) is described in section 3.2.1. Step b) is described in section 3.2.2.

The airship is like an ellipse, so the volume is equal to:

$$V_{\text{TEMOLeb Airship}} = \frac{4}{3} \pi \left(\frac{L}{2}\right) \left(\frac{D}{2}\right)^2$$

$V_{\text{TEMOLeb Airship}}$: The volume of the airship (m^3).

L: The length of the airship (m).

D: The diameter of the airship (m).

$$V_{\text{TEMOLeb Airship}} = \frac{4}{3} \pi \left(\frac{20}{2}\right) \left(\frac{4}{2}\right)^2 = 167.55 \text{ m}^3.$$

We draw this airship having this dimensions using a free 3D modeling software named “FreeCAD”. This program is oriented towards product design and mechanical engineering. It also targets towards architecture or other branches of engineering.

26.2.1 Design of “TEMO-Leb Airship”


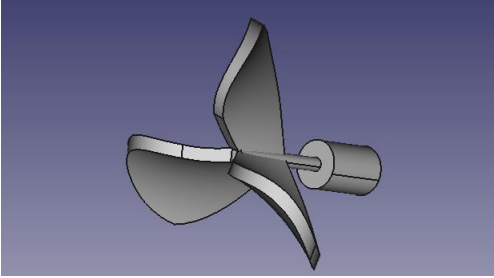

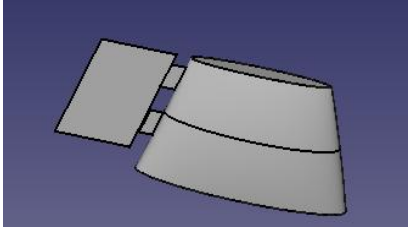

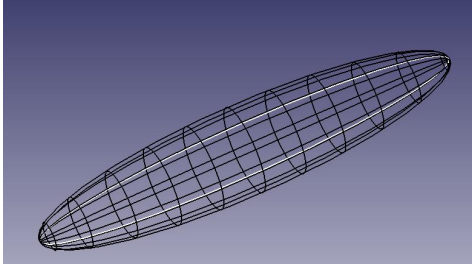

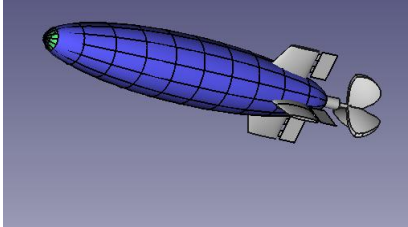

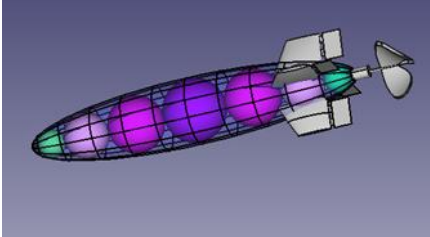

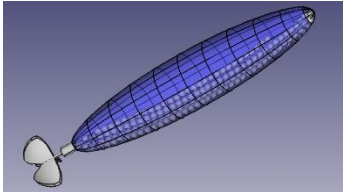
Part and details	Date and Data	Design
<p>Propeller</p>	<p>May/2017</p>  <p>Propeller.Design.FCStd</p>	
<p>Rudder and Elevator</p>	<p>May/2017</p>  <p>Rudder.Elevator.FCStd</p>	
<p>Frame of Airship L=20 m D=4m</p>	<p>April/2017</p>  <p>Frame.of.Airship.FCStd</p>	
<p>Outer Shell of the Airship</p>	<p>April/2017</p>  <p>Outer.Shell.of.airship.FCStd</p>	
<p>Airship with the internal gasbags</p>	<p>April/2017</p>  <p>Airship.with.the.internal.gasbags.FCStd</p>	
<p>Airship With Balloons</p>	<p>29/May/2017</p>  <p>070617_Airship_Gazbags(Balloons).FCStd</p>	

Table 1: All Parts and design of Airship

➤ **Propeller**

In order to displace the dirigible, we use the propeller. It transmit power by converting the rotational motion and it provides the main thrust. The propeller is also used to provide various speed.

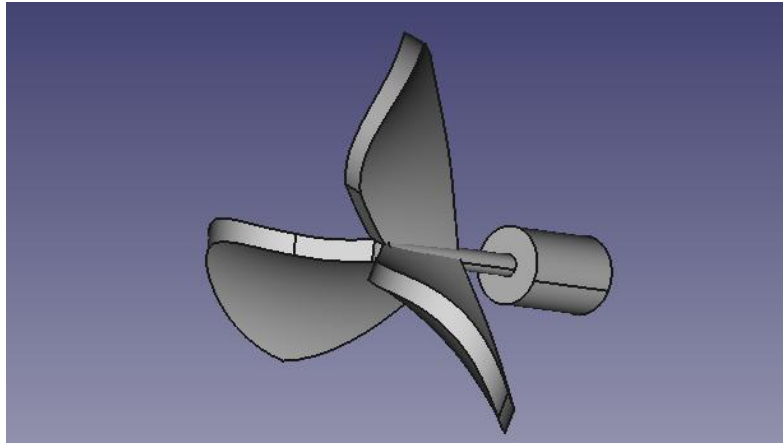


Figure 24: The propeller model drawn with FreeCAD

➤ **Rudder and Elevator**

To control an airship, we need many instruments, among them: rudders and elevators.

- The rudder is the part of airship which rotates and help to turn right or left.
- The elevator is like the rudder but it leads the dirigible to rise or descend.

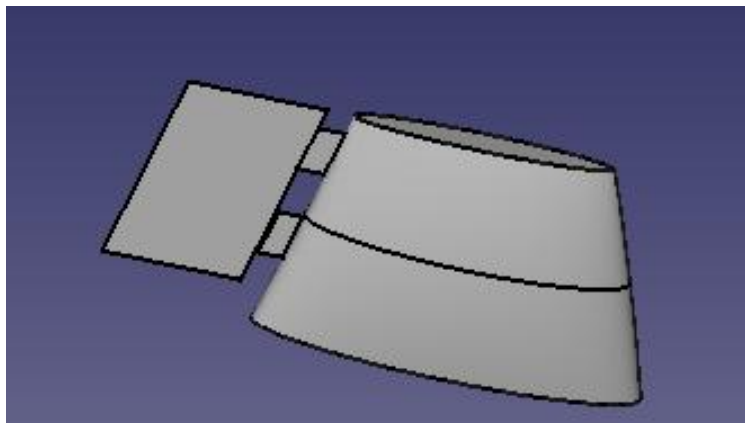


Figure 25: The rudder and the elevator models drawn with FreeCAD

➤ **Frame of the airship**

The frame is the body of airship, it gives its shapes. Which is main characteristic of rigid airship

It has formed by:

- Six ellipses: The length of the major axis is 20 m and the length of the minor axis is 4 m.

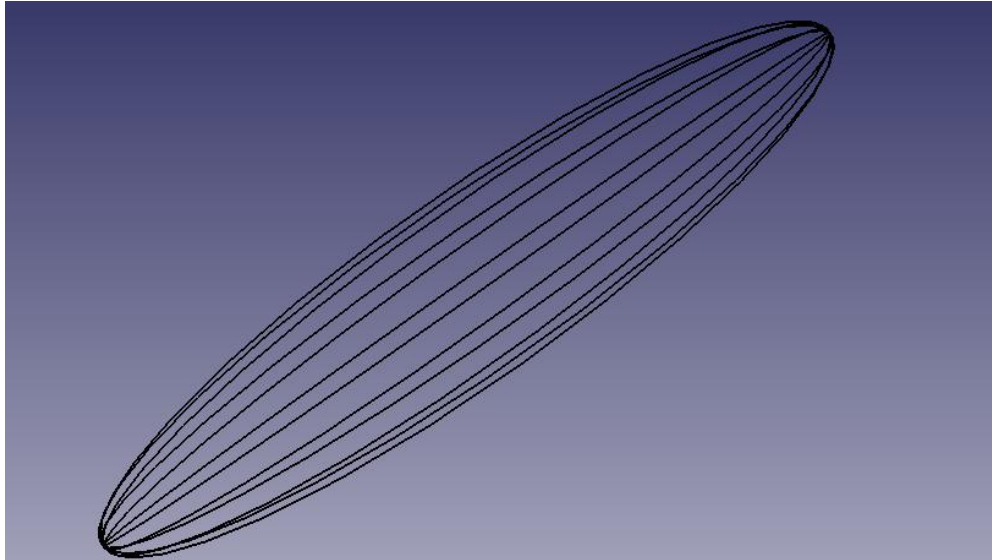


Figure 26: Ellipses of airship are drawn using FreeCAD

- 11 circles: This airship have 11 circles, four of them are symmetrical. Theirs dimensions are given in the table below.

Circle	Radius(m)
Circle 1	2
Circle 2	1.96
Circle 3	1.83
Circle 4	1.56
Circle 5	1.19
Circle 6	0.43
Circle 7	0.66

Table 2: Radius of Circles of the airship

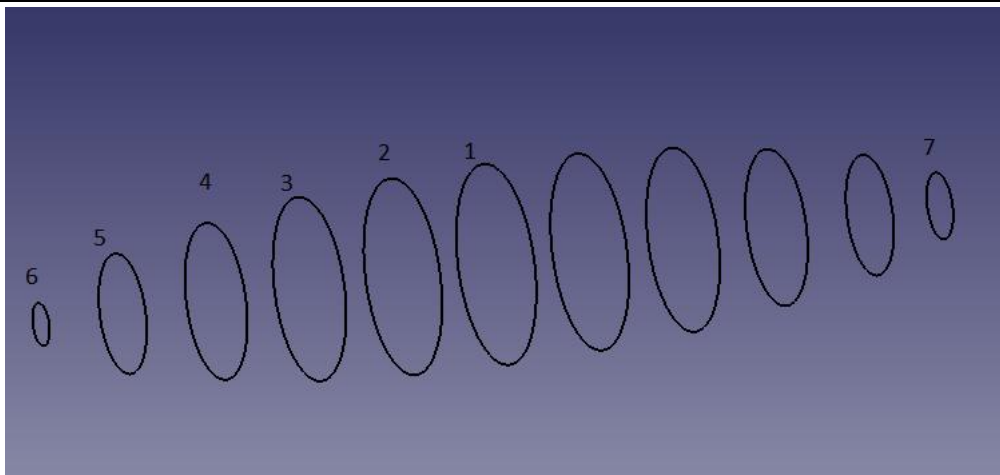


Figure 27: Circles of airship are drawn using FreeCAD

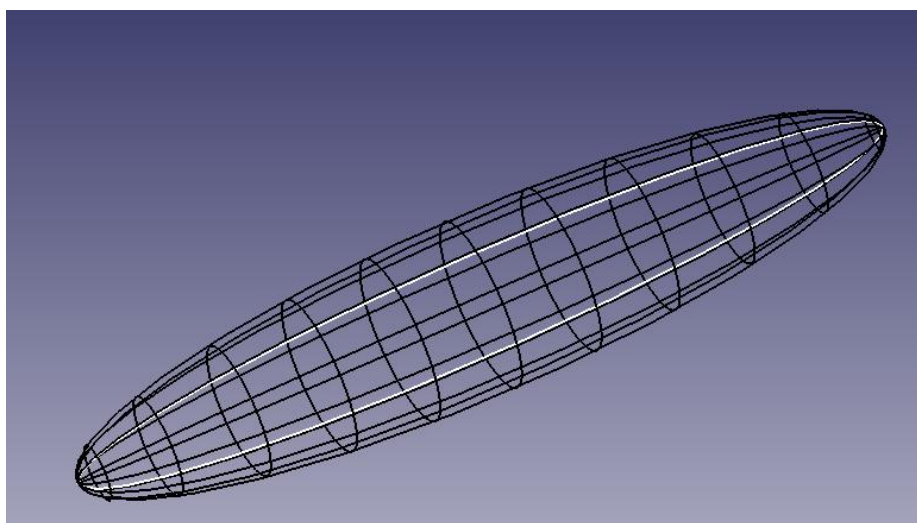


Figure 28: The framework of airship model drawn with FreeCAD

➤ **Outer shell of the airship**

The outer shell must be light. It covers the whole airship.

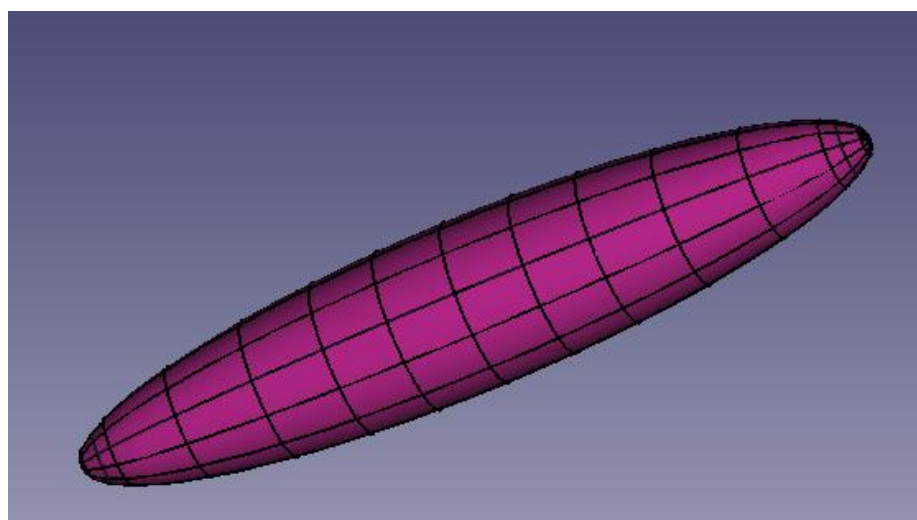


Figure 29: The hull of the airship designed with FreeCAD

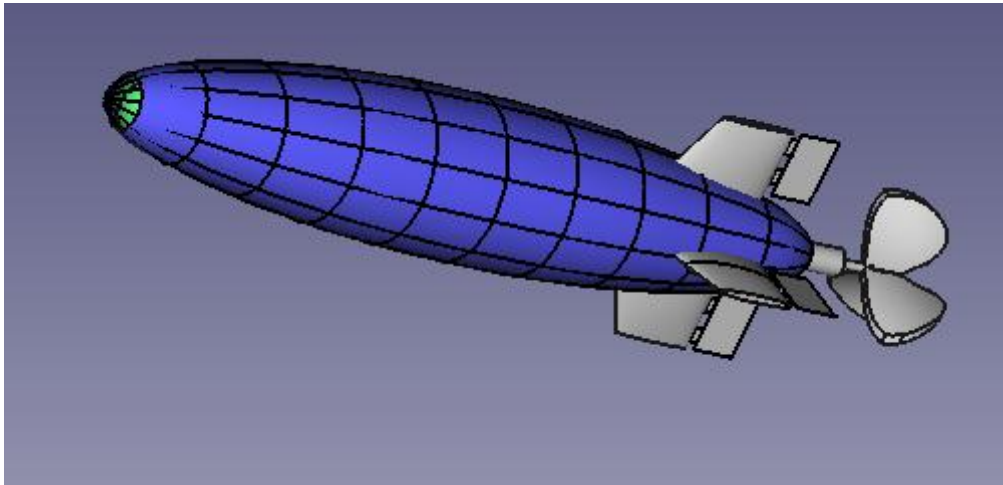


Figure 30: The envelope of airship model drawn with FreeCAD

➤ **Airship with the internal gasbags**

The internal gasbags contain the lifting gas and are designed with different dimensions.

The gas used is Helium and should choose carefully the envelope of gasbags. We know the atom of Helium is very small so it should not leak from the envelope. It gives greater durability of the airship to stay at the desired altitude.

This airship have 7 bags, three of them are symmetrical. Theirs dimensions are given in the table below.

Bag	Radius(m)
Bag 1	1.9
Bag 2	1.7
Bag 3	1.35
Bag 4	0.83

Table 3: Radius of Bags of the airship

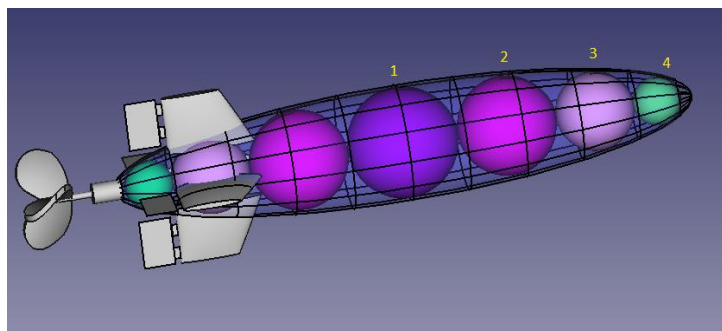


Figure 31: The internal gasbags of the airship

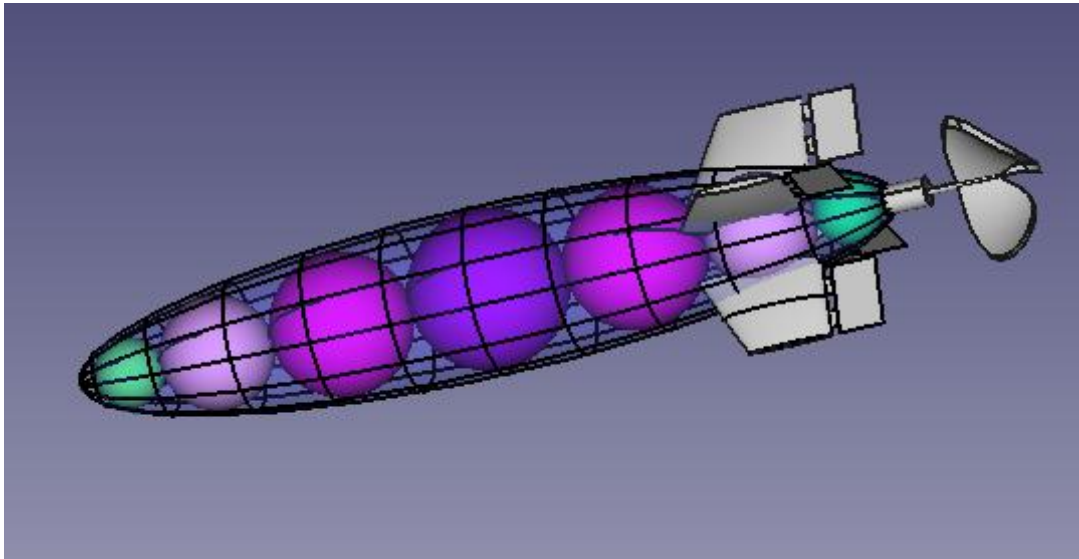


Figure 32: The internal gasbags of airship model draw with FreeCAD

➤ **Airship with balloons**

After a search in the Lebanese market done to find gasbags with the chosen dimensions, we did not find our requests. That's why we decided to use balloons having a 0.5 m of diameter instead of gasbags.

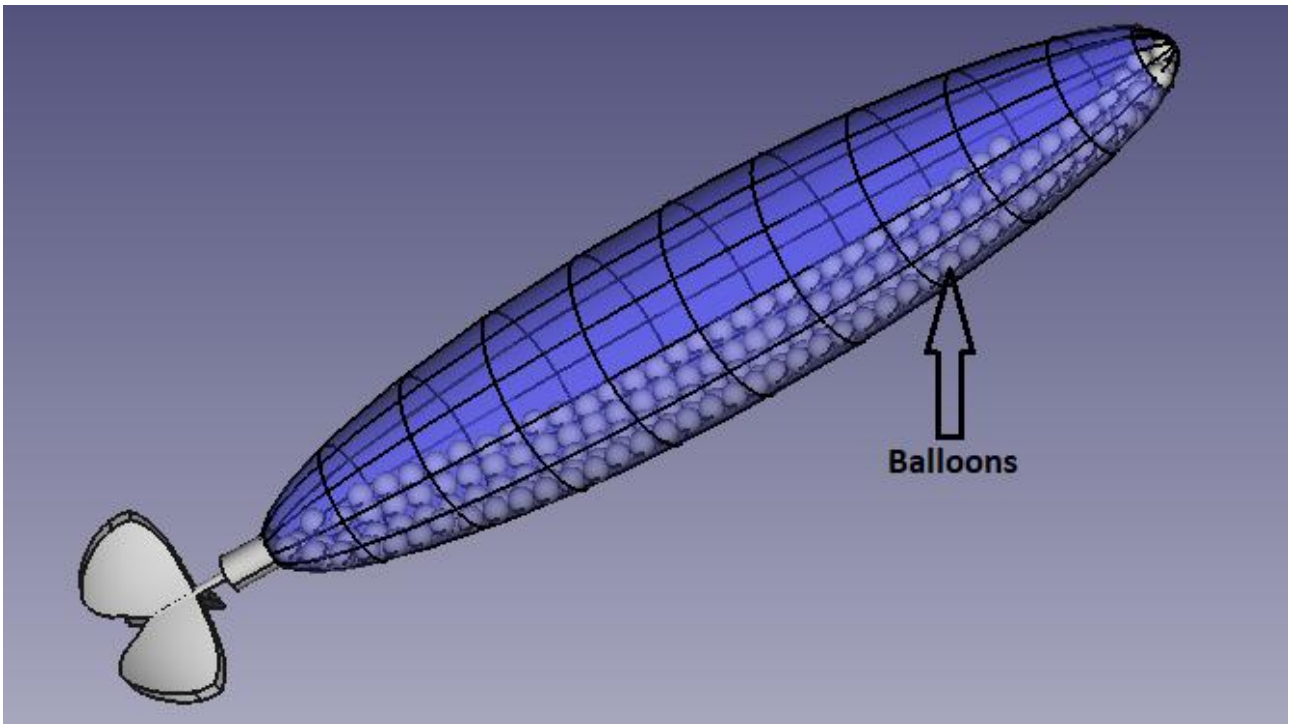


Figure 33: The airship with balloons model draws with FreeCAD

26.2.2 Mass of framework

To obtain the mass of framework, we need to choose the suitable material and to know its characteristics and its density. Then, we calculate the volume of the framework and finally, we use equation Eq.5.

Plexiglas is the suitable material to construct the airship framework. On the Internet, there are many choices. We choose Plexiglas stick, plastic, having a circular section of diameter 6 mm and a length of 1m. ^[10]

It has also many characteristics:

1. Good mechanical strength
2. Good shock resistance
3. Good dielectric properties
4. Minimal absorption of humidity
5. UV resistance
6. Density= 1180 kg/m³
7. Elongation at rupture= 4%
8. Compressive strength= 118 N/mm²
9. Elasticity module= 3140 N/ mm²
10. Specific Heat= 0.35 cal/g°C
11. Coefficient of linear expansion= $6.8 \cdot 10^{-5}$ (1/°c)
12. Resistance to cold temperature > -40°C



Figure 34: Stick of Plexiglas

Now, we know the characteristic of Plexiglas, so we can calculate the mass of framework. At the beginning, the volume of the frame must be calculated.

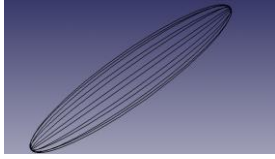
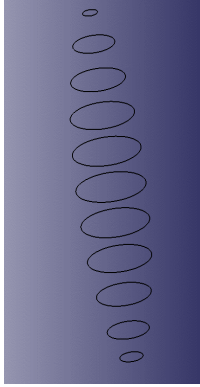
Plexiglas (Material of Structure)			
Volume of structure of ellipses (cm ³)	Ellipse 1	1281	
	Total (6 Ellipses)	7686	
Volume of structure of circles(cm ³)	Circle 1	355	
	Circle 2	76	
	Circle 3	118	
	Circle 4	213	
	Circle 5	213	
	Circle 6	277	
	Circle 7	277	
	Circle 8	326	
	Circle 9	326	
	Circle 10	348	
	Circle 11	348	
	Total	2877	

Table 4: The volume of frame made of Plexiglas in cm³

Volume of ellipse is calculated by this equation:

$$V_{\text{ellipse}} = 2 \cdot \pi \cdot \left(\left(\left(\frac{L}{2} \right)^2 + \left(\frac{D}{2} \right)^2 \right) / 2 \right)^{0.5} \cdot \pi \cdot S^2 \quad \text{Eq.A}$$

Volume of circle is calculated by this equation:

$$V_{\text{circle}} = 2 \cdot \pi \cdot \left(\frac{D}{2} \right) \cdot \pi \cdot S^2 \quad \text{Eq.B}$$

S: Demi-Section of circle and ellipse manufactured by Plexiglas.

Using Eq.5, we obtain the mass of frame made of **Plexiglas**:

Aerodynamic investigation of a high altitude airship

Total volume of frame : Volume of ellipses + Volume of circles	0.01 m ³
Density of Plexiglas	1180 kg/m ³
Mass of Plexiglas	12.47 kg

Table 5: Mass of Plexiglas

26.2.3 Mass of gasbags

We need a specific material when we use Helium. The newly developed envelope is made of high-strength, tear-resistant multi-layer laminates. Even a lighting impact cannot significantly affect the flight characteristics. The envelope has a slight overpressure of 5 mbar.

The materials used for this envelope:

- Outer Layer: "Teldar". It is a film which has a protection from UV rays.
- Intercellular Layer: "Polyester fabric". It has a snag resistant.
- Internal Layer: "Polyurethane". It has many characteristics: weldable and waterproof. ^[11]

This envelope has a density equal to 0.25 kg/m² and a force tearing equal to 285 N/cm.

Now, we know the characteristic of the envelope, so we can calculate the mass of gasbags. At the beginning, the surface of the gasbags must be calculated. Eq.5 is not suitable because when we study the tissues, we need to use the surface instead of the volume. We calculate the mass using this equation:

$$\text{Surface density } \rho' = m/S \quad \text{Eq.5'}$$

Material of Gasbags		
Surface of Gasbags(m ²)	Bag 1	45.36
	Bag 2	36.32
	Bag 3	36.32
	Bag 4	22.90
	Bag 5	22.90
	Bag 6	8.66
	Bag 7	8.66
	Total	181.12

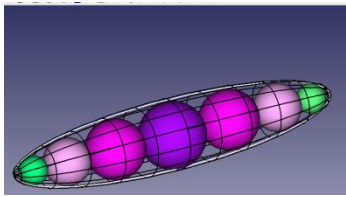


Table 6: The surface of gas bags in m²

The surface of gasbags is calculated by this equation:

$$S_{\text{sphere}} = 4 * \pi * (D/2)^2$$

Total surface of gasbags:	181.12 m ²
Surface Density	0.25 kg/m ²
Mass of gasbags	45.28 kg

Table 7: Mass of gasbags

26.2.4 Mass of external fabric

The tissue used for the external fabric is different from the one that is used in the bags. The envelope of the bags does not leak gas and this is not the status of the outer shell. So another type of tissue is used “Cotton Fabric “has a density of 0.075 kg/m².

Surface of external shell	251.33 m ²
Density	0.075 kg/m ²
Mass of external fabric	18.85 kg

Table 8: Mass of external fabric

This theoretical study is difficult to implement now, because, in this project, we want to elevate the airship at high altitude, given that we have become large-scale.

The approval of the Lebanese government must be taken, and it will take a long time. Moreover, the difficult circumstances encountered by neighboring regional states may hinder such approval.

As the aim of the project is to revive the science of airships, we decide to set up a small airship at low altitude, with some modifications to the theoretical study, due to the unavailability of airship manufacturers in the Lebanese market.

26.3 Low altitude test TEMO-Leb Airship

26.3.1 New design with new dimensions of the Low altitude test TEMO-Leb Airship

The new design of the Low altitude test TEMO-Leb Airship has the following dimensions which elevates at a low altitudes. This airship is actually under construction.

Diameter= 1.2 m;

Length= 6.4577 m

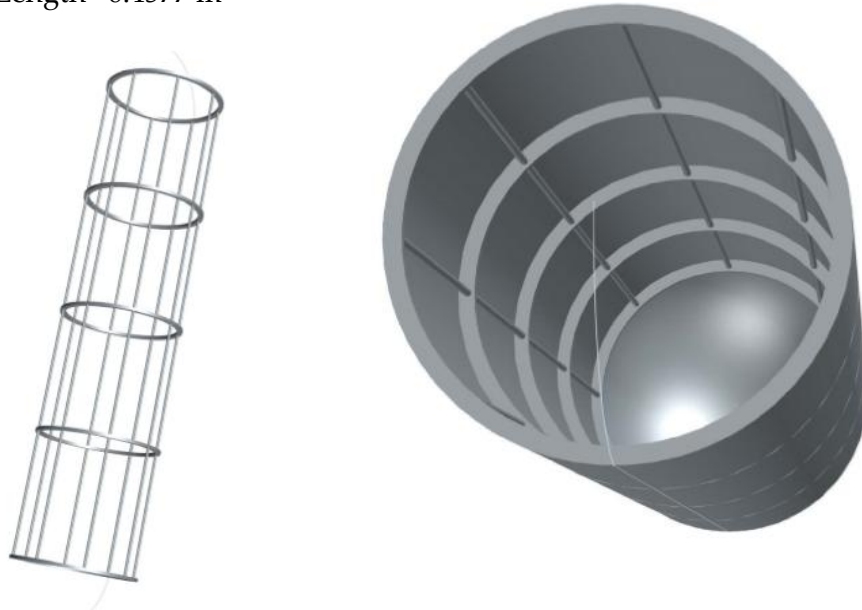


Figure 35: The framework of airship model draws with AutoCAD

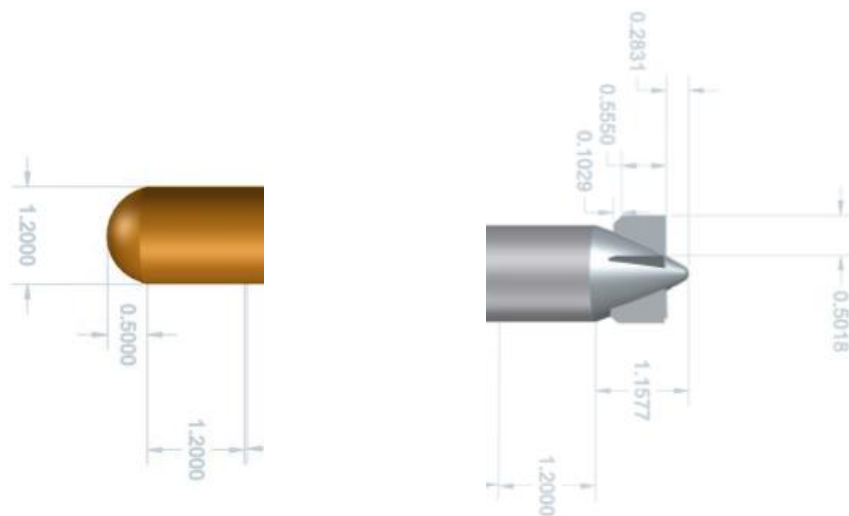


Figure 36: The new design of airship draws with AutoCAD

26.3.2 Materials of the test device for the Low altitude test TEMO-Leb Airship in Lebanese market

1) Balloons filled with Helium

The gasbags are replaced by the balloons, having 1.5 m as a diameter. It is important to know the mass hold by the balloons and the pressure of Helium contained into the balloon. To obtain these parameters, we want to perform this calculation:

Firstly, we calculate the volume of balloon:

$$V_{\text{Balloon}} = \frac{4}{3} \pi (D/2)^3 = \frac{4}{3} \pi (1.5/2)^3 = 1.77 \text{ m}^3$$

We know the density of Helium at 20°C: $\rho_{\text{He}} = 0.178 \text{ kg/m}^3$ [14]

By applying Eq.5, we obtain the mass theirs hold by the balloons:

$$m_{\text{hold by 1 balloon}} = (\rho_{\text{air}} - \rho_{\text{He}}) \cdot V_{\text{Balloon}} = ((1,2041 - 0.178) \cdot 1.77) \text{ kg} = 1.8 \text{ kg}$$

$$m_{\text{hold by balloons}} = 4 \cdot 1,8 \text{ kg} = 7.2 \text{ kg}$$

The pressure of Helium into the balloon can be calculate by using the equation:

$$P \cdot V = (m/M) \cdot R \cdot T \quad \text{Eq.10}$$

P: Pressure of Helium into the balloon (Pa)

V: Volume of balloon (m^3)

m: Mass of balloon (kg)

M: Molar mass of Helium (kg/m^3)

R: The universal gas constant = $8.314 \text{ JK}^{-1}\text{mol}^{-1}$

T: The absolute Temperature of Helium (k)

To obtain the mass, we use Eq.5:

$$\rho = m/V; m = 0.178 \cdot 1.77 = 0.3 \text{ kg/m}^3$$

The pressure is:

$$P = ((m/M) \cdot R \cdot T) / V = ((0.3/4) \cdot 8.314 \cdot 293.15) / 1.77 = 108457.6 \text{ Pa} = 1.07 \text{ atm}$$

ρ : Density of Helium at 20°C = 0.178 kg/m^3

2) Plexiglas

The framework of airship is manufactured by empty tubes of Plexiglas. The section of tube is equal: 6 mm. The mass of framework must be lower than the mass hold by the balloons, in order to rise the airship.

To ensure this calculations, we have to wait for the implementation.

3) External envelope

For the external envelope, a very light type of fabric must be used, because Plexiglas and the balloons cannot sustain an excess weight.

26.4 Results of theoretical study

26.4.1 Results of “MATLAB”

We use “MATLAB” for plotting the variation of density, pressure and temperature as a function of altitude. The results can be found below:

1) Variation of the density with altitude

- ✓ Plotting the function:

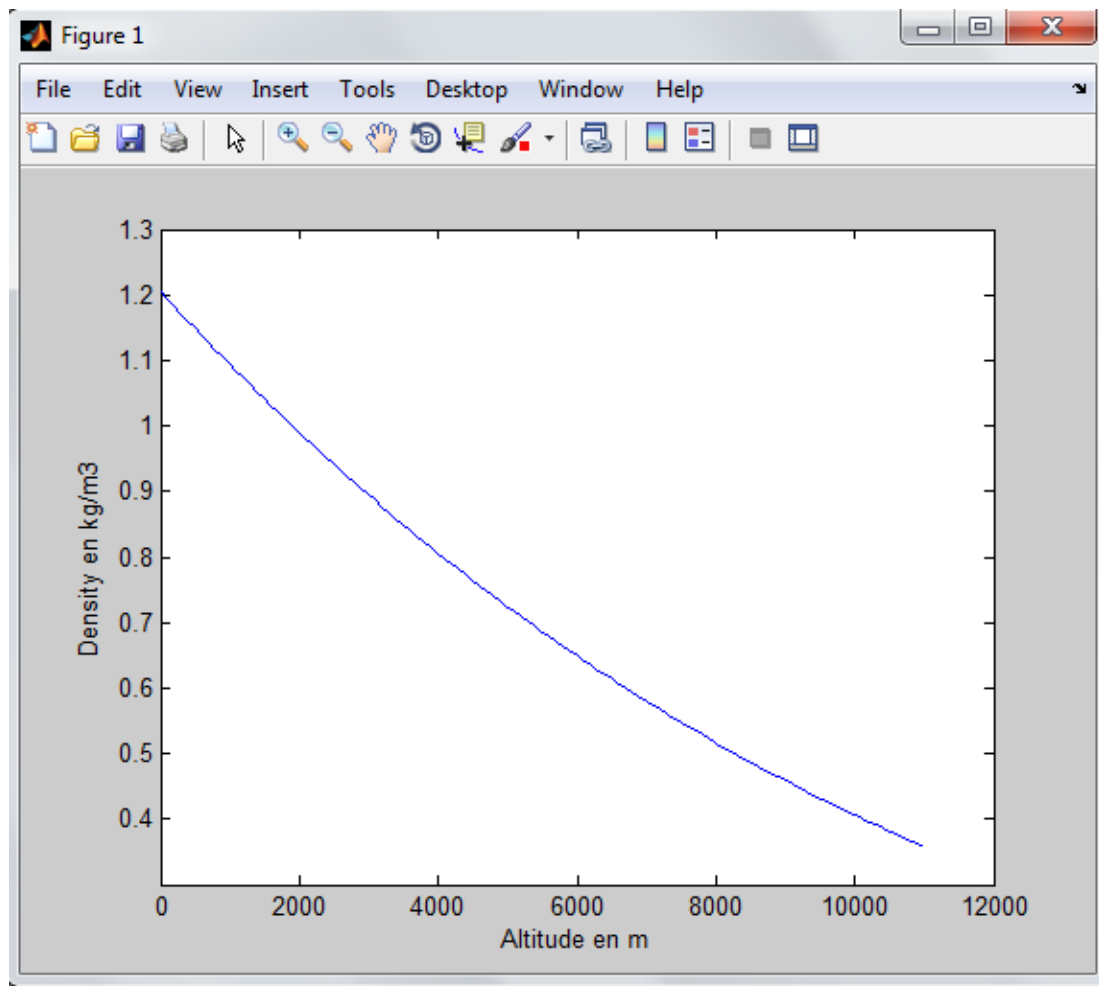


Figure 37: Variation of density (kg/m3) with altitude (m).

This graph shows the variation of the density as a function of altitude along 11 km. We notice that the density decreases with the increase of altitude.

2) Variation of pressure with altitude

- ✓ Plotting the function:

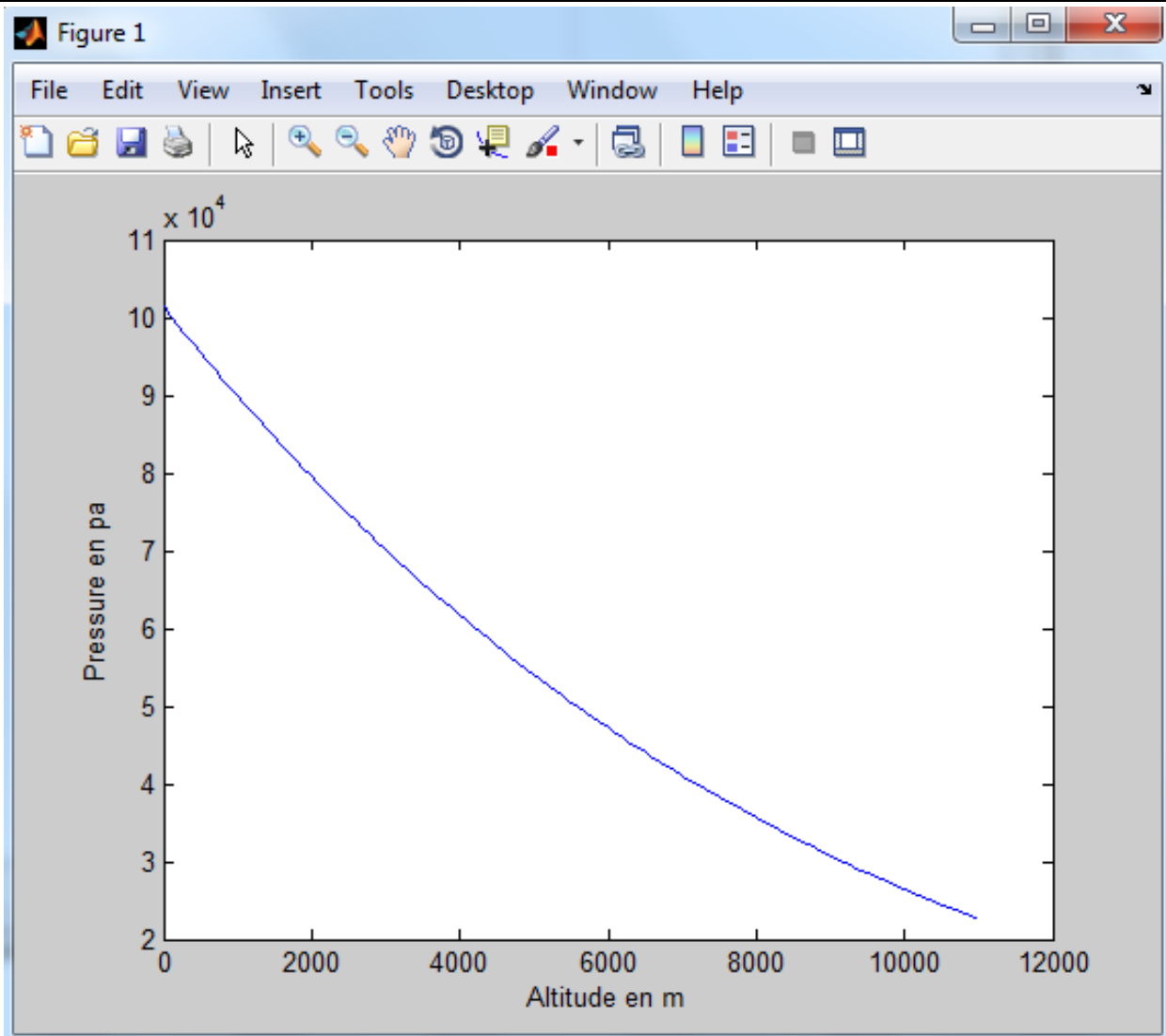


Figure 38: Variation of pressure (pa) with altitude (m) computed for 15°C and 0% relative humidity.

This graph shows the variation of the pressure as a function of altitude along 11 km. We notice that the pressure decreases with the increase of altitude.

3) Variation of temperature with altitude

- ✓ Plotting the function:

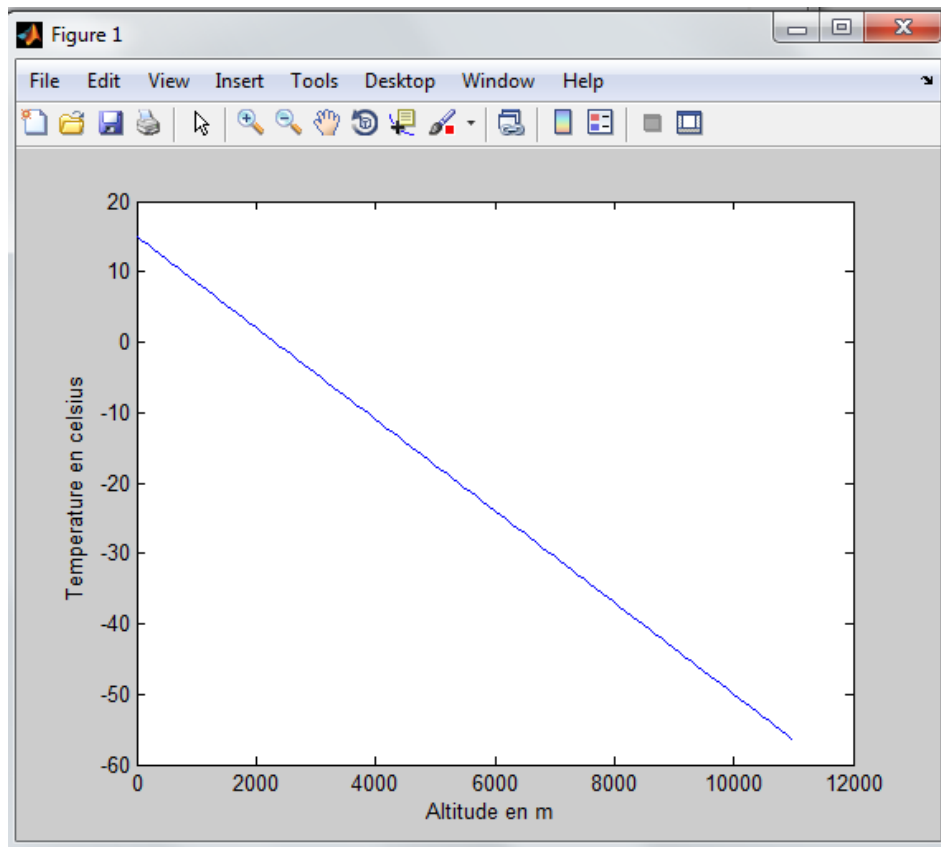


Figure 39: Variation of temperature (°C) with altitude (m).

This graph show the variation of the temperature as a function of altitude along 11 km. we notice that the temperature decreases with the increase of altitude.

4.1.2. V TEMOLeb Airship and the V_{max}

We will calculate the maximum volume of Helium contained in the gasbags, and we will check whether the airship able to reach the proposed height.

$$V_{max} = (m_s + m_p) / \sigma * (Q_{A0} - Q_{He0})$$

To obtain the mass of structure, it is necessary to add the mass of framework, the mass of gasbags and the mass of external fabric.

$$m_s = 76.6 \text{ kg}$$

It is assumed that the mass of payload:

$$m_p = 2 \text{ kg}$$

We have the density of air and Helium at 15°c and at sea level:

$$Q_{A0} = 1.225 \text{ kg/m}^3; Q_{He0} = 0.169 \text{ kg/m}^3$$

The density ratio “ σ ” is varied with altitude, depending on the below table, we calculate the ratio:

$$\sigma = Q_A / Q_{A0}$$

	0	100	200	300	400
-500	1.285	1.273	1.261	1.249	1.237
0	1.225	1.213	1.202	1.190	1.179
500	1.167	1.156	1.145	1.134	1.123
1000	1.112	1.101	1.090	1.079	1.069
1500	1.058	1.048	1.037	1.027	1.017
2000	1.007	0.996	0.986	0.977	0.967
2500	0.957	0.947	0.938	0.928	0.919
3000	0.909	0.900	0.891	0.881	0.872
3500	0.863	0.854	0.845	0.837	0.828
4000	0.819	0.811	0.802	0.794	0.785
4500	0.777	0.769	0.760	0.752	0.744
5000	0.736	0.728	0.720	0.713	0.705
5500	0.697	0.690	0.682	0.675	0.667
6000	0.660	0.652	0.645	0.638	0.631
6500	0.624	0.617	0.610	0.603	0.596
7000	0.590	0.583	0.576	0.570	0.563
7500	0.557	0.550	0.544	0.538	0.531
8000	0.525	0.519	0.513	0.507	0.501
8500	0.495	0.489	0.484	0.478	0.472
9000	0.466	0.461	0.455	0.450	0.444
9500	0.439	0.434	0.428	0.423	0.418
10000	0.413	0.408	0.403	0.398	0.393
10500	0.388	0.383	0.378	0.373	0.369
11000	0.364	0.359	0.355	0.350	0.346
11500	0.341	0.337	0.333	0.328	0.324
12000	0.320	0.316	0.311	0.307	0.303

Table 9: Variation of density of air with altitude

To obtain the density of atmosphere, we use table 9, when we want to obtain the density of 600 km, we see the column in the left: 500 and the line above 100 and we read the density 1.156 kg/m³.

Altitude (km)	$\sigma^*(Q_{A0}-Q_{He0})$	$V_{max}(m^3)$
H=0	1.056	74
H=1	0.959	82
H=2	0.868	91
H=3	0.784	100
H=4	0.706	111
H=5	0.634	124
H=6	0.569	138
H=7	0.508	155
H=8	0.461	170

Table 10: Variation of volume Helium as function of altitude

4.1.3. Positive results

From 0 km to 7 km, we are under the point “Pressure Altitude” (explain in Chapter 3).

Under this point, we have:

- ❖ The differential of pressure between lifting gas and atmosphere is null: $\Delta P=0$.
- ❖ The differential of temperature between lifting gas and atmosphere is null: $\Delta T=0$.
- ❖ The density of lifting gas and atmosphere decrease, this variation affects an increase of net volume.
The equation Eq.5 is available up to the **“Pressure Altitude”**.

The variation of density and pressure is composed by the volume. When the density and the pressure decrease, the volume of gasbags increases.

At 7 km, this is the proposed altitude to elevate **TEMO-Leb airship**:

$$V_{\text{TEMOLeb Airship}} > V_{\text{max}}$$

This is a **positive result**, it hasn't any danger to a rupture of airship. The volume of Helium in the gasbags is enough to rise **TEMO-Leb airship**, at **7 km**.

The airship cannot to rise at 8 km, because:

$$V_{\text{TEMOLeb Airship}} < V_{\text{max}}$$

We are now above the point **“Pressure Altitude”** (explain in Chapter 3).

When the airship continue to rise, the density of lifting gas and atmosphere also continue to decrease, but the volume of lifting gas remains constant.

The differential of pressure now isn't null, it starts to increase. The overpressure is created, and consequently, a rupture of the exterior structure of airship can be result if the differential becomes too great.

27 References

27.1 References for Chapters 1

[FatimaAlChaar 2015] Fatima Al Chaar, "Simulation of the meteorological satellite IAP-SAT", Master Thesis, AECENAR/LU, 2015, see www.aecenar.com/publications

27.2 References for Sections 2.1, 2.10-2.15 and Chapter 4

- [1]: <http://www.portail-aviation.com/2015/07/dossier-dirigeable-episode-1-lhistoire-des-dirigeables-pionniers-de-laeronautique.html>
- [2]: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/media/airship_aerodynamics.pdf
- [3]: <https://en.wikipedia.org/wiki/Actuator>
- [4]: <https://www.quora.com/What-is-an-actuator>
- [5]: <http://www.baldor.com/Shared/manuals/1205-394.pdf>
- [6]: https://en.wikipedia.org/wiki/Pneumatic_motor
- [7]: http://www.bluetools.com/Air-Tools-Motors/c88_50/index.html
- [8]: <http://www.machinedesign.com/archive/basics-electromagnetic-clutches-and-brakes>
- [9]: <http://www.warnerelectric.com>
- [10]: <https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor>
- [11]: https://en.wikipedia.org/wiki/Induction_motor
- [12]: <http://electronicsforu.com/buyers-guides/selecting-electric-motor-drive-system>
- [13]: https://en.wikipedia.org/wiki/Hydraulic_motor
- [14]: <http://www.directindustry.com/prod/hydro-leduc/product-7677-1287099.html>
- [15]: <https://circuitdigest.com/article/servo-motor-basics>
- [16]: <https://www.sparkfun.com/products/11965>
- [17]: https://www.iei.liu.se/flumes/tmhp51/filearchive/coursematerial/1.105708/HydServoSystems_part1.pdf
- [18]: <https://books.google.com.lb/books?id=cF7YBAAAQBAJ>
- [19]: www.mobilehydraulictips.com/hydraulic-motors/
- [20]: <http://www.supinfo.com/articles/single/296-qu-est-ce-qu-servomoteur>
- [21]: https://www.teamrobobox.fr/documentation/02_Le_moteur.pdf
- [22]: <http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html>
- [23]: <https://www.servocity.com/what-is-a-servo>
- [24]: <http://www.pobot.org/+servomoteur+.html?lang=fr>
- [25]: <https://www.scribd.com/document/99583469/Introduction-to-Servo-Motors-Arduino>
- [26]: <https://makezine.com/2016/05/13/understanding-types-of-servo-motors-and-how-they-work/>

27.3 References for Sections 2.2-2.9 and Chapter 3

^[1] Joseph Louis Lecornu, *La navigation aérienne: histoire documentaire et anecdotique*

^[2] **La technique du ballon, G. Espitallier - 1907**

^[3] **www.eballoon.org**

^[4] Frederick, Arthur, et al., *Airship saga: The history of airships seen through the eyes of the men who designed, built, and flew them*, 1982

^[5] Griebel, Manfred and Joachim Dressel, *Zeppelin, The German Airship Story*, 1990

^[6] Archbold, Rich and Ken Marshall, *Hindenburg, an Illustrated History*

^[7] Althoff, William F., ***USS Los Angeles: The Navy's Venerable Airship and Aviation Technology***

^[8] Lutz, T. and Wagner, S., "Drag Reduction and Shape Optimization of Airship Bodies," Institute for Aerodynamics and Gas Dynamics, University of Stuttgart, AIAA, Germany, 1997.

^[9] **Rehmet, M. A., Krplin, B., Epperlein, F., R.Kornmann, and Schubert, R., "Recent Developments on High Altitude Platforms**

^[10] **www.plastiquesurmesure.com/materiaux-plastiques.html**

^[11] www.carnetdevol.org/zeppelin/technique.html

Appendix A: Contact data of specialists (معلم), workers, ...

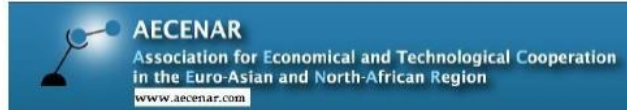
Specialist for / price	Name	Address	Phone
Aluminium, 80\$/qm	عمر	بعبة - عكار	70 140828
Electricity 25 USD/day	Abdullah (from Syria), brother of Ibrahim (Mustafa knows him)		
Sanitary 25 USD/day	Abdullah (from Syria), brother of Ibrahim (Mustafa knows him)		
Painting 25 USD/day	Abdullah and Ibrahim (from Syria) (Mustafa knows them)		
Bilat	Mustafa (from Halab)	Ras Nhache	76 493901
Welder / Metal working	Muhammad Qammah	Mina	70 339875
	Muhammad Akkumi	Biddawi	71669613
	Said Hussein, 25- 45.000LL/day	Biddawi	06/383728 or 03/793802
Stainlessschweißer	Bilal Naouchi	bilalnaoushi@ho tmail.com	03 446027
Wärme u. Kälte technik u.s.w.	Khidr Balita	Mina	03 232088

Appendix B: for Aerodynamic Investigation

Initial Working packages



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



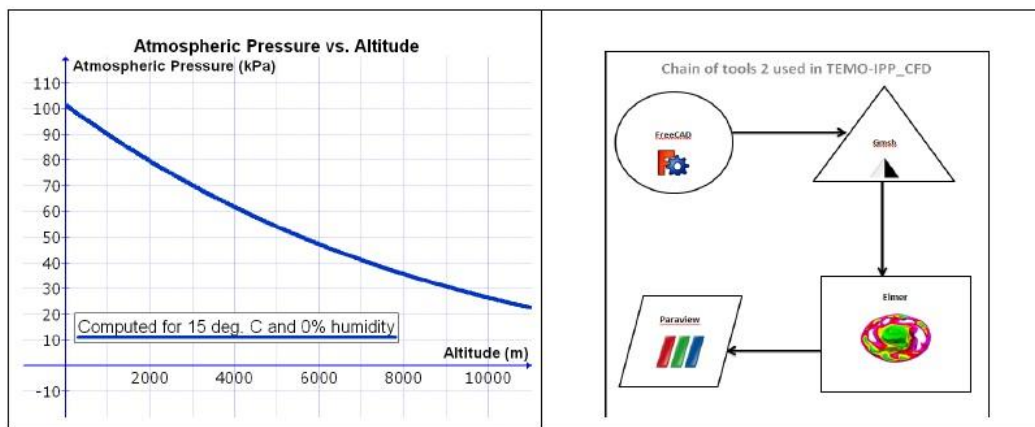
Ras Nhache, Batroun/Tripoli, 27.02.2017

There TEMOLEbanon-Mintad project aims to offer internet access via a airship platform. There are the following Milestones for 2017/2018:

- Sep 2017: Low-altitude flying airship (1-3 km) for internet supply is in operation (above Qalamoun/Ras Nhache) (prototype is developed with some master thesis')
- 2018 High-altitude platform (20-22 km) for internet supply in cooperation with Turkey



Master Thesis: aerodynamic investigation of a solar powered high altitude airship



Tasks:

- Rough modeling of all parts of platform including internet communication payload
- Computing of total weight to required helium gas volume of lifting cells relation
- Specification of rough design parameters of airship (length, volume of lifting cells) and altitude 1 (about 1-2 km) and altitude 2 (about 20 km)
- Animation (Film) of flights in the two operational altitudes

Contact:

Eng. Samir Mourad, Mob. +961 76 341526 (Lebanon), WhatsApp +49 178 7285578 (Mob. Germany)

Outworked by Souha Bakri

Estimation for costs of logging sensors during flight



Haykalyeh Str Harba Bld Ground
 Floor
 Ras-Maska
 Tripoli, Lebanon, 0000
 Phone # 06412895
 Web Site www.cnclablb.com

Estimate

Date	Estimate #
8/15/2017	E17-54

Name / Address
Souha Bakri

Ship To		
Customer Phone	P.O. No.	Price Level

Item	Description	Qty	Cost	Total
WRL0057	HR GY-NEO 6M V2 GPS module	1	27.95	27.95
INT0089	Micro SD Storage Board TF Card Reader Memory Shield	1	1.94	1.94
PWR0024	Polymer Lithium Ion Battery - 1000mah 7.4v	2	12.95	25.90
DEV0042	HR UNO R3 + USB Cable	1	12.95	12.95
PRT0005	Jumper Wires - Connected 6" (M/F, 20 pack)	1	2.50	2.50
SEN0131	HR 3pin Button Key Switch Sensor Module	1	1.95	1.95
INT0030	Digital Red LED Light Module	1	2.99	2.99
Total				\$76.18

This quotation is valid for 30 days since issued

Customer Signature _____

**Building of the frame including actuators for a small prototype of an airship
(2017)**

28 Basics concerning actuators used in aerospace

28.1 What's an actuator?

The actuator is a component of the device responsible for the transmission or control of a mechanism or system, for example by running (opening or closing) a valve; in simple terms, it is an "engine".

The control signal and the source of energy are required for the operator (Figure 1) principal of work of actuator.). Relatively, it is a low power same as electrical voltage, or pneumatic or hydraulic push, or maybe human force. The principal source of power furnished can be electrical current, hydraulic liquid press, or aerial compression. After receiving the control signal, a mechanical movement appear from the operator by converting her energy. ^[1]: <http://www.portail-aviation.com/2015/07/dossier-dirigeable-episode-1-lhistoire-des-dirigeables-pionniers-de-laeronautique.html>

[2]: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/media/airship_aerodynamics.pdf

[3]: <https://en.wikipedia.org/wiki/Actuator>

Actuators are usually consumed by factories or industrial usage and can be utilized in devices like engines, pumps, switches and vans.

This motion can be in almost any form, such as blocking, clamping or output. Actuators are usually used in manufacturing or industrial applications and can be used in devices such as motors, pumps, switches and valves. ^[4]: <https://www.quora.com/What-is-an-actuator>

A simple scheme (*Figure 40: principal of work of actuator.*^[4]: <https://www.quora.com/What-is-an-actuator>) can explain the principal of work of actuator:

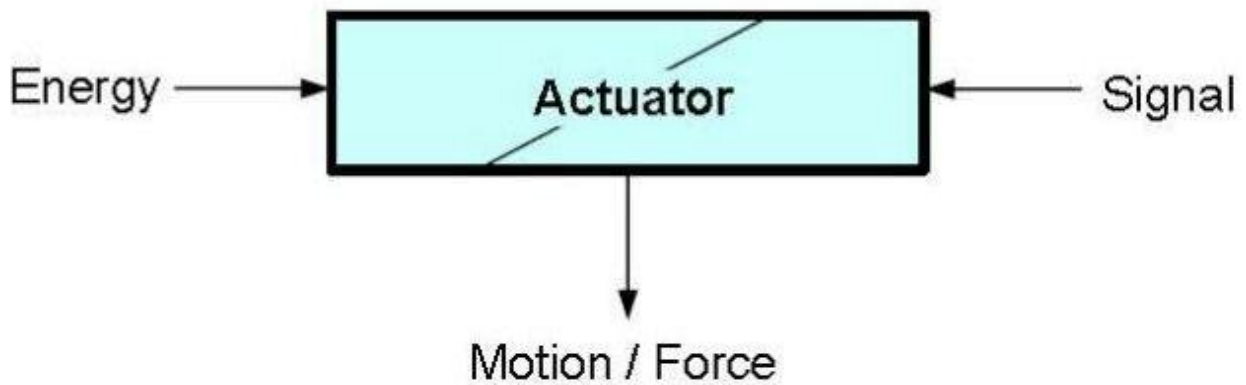


Figure 40: *principal of work of actuator.*^[4]: <https://www.quora.com/What-is-an-actuator>

28.2 Types of actuators

Many types of actuators for controlling motion such as speed control, torque or positional accuracy exist:

- **Air Motors (pneumatic)**
- **Hydraulic Motors**
- **Clutch/Brake**
- **Stepper Motors**
- **AC Induction Motors**
- **Servomotors** ^[5]: <http://www.baldor.com/Shared/manuals/1205-394.pdf>

28.2.1 Air Motors

A pneumatic engine is a kind of mechanical motor that use pressed air to make motion. His principal based on transforming pressed air power into mechanical action either over straight or rotary movement. Many kinds (**Figure 41: axial piston, radial piston, and rotary vane air motors.** ^[7]:http://www.bluetools.com/Air-Tools-Motors/c88_50/index.html) of air engine exists like axial piston, radial piston or rotary vane motor. A diaphragm or piston motor, can create the linear motion while the rotary motion is provided by a type air engine, air piston engine, and wind turbine or gear type motor.

The manufacture of hand tools are using this motor vastly and successfully, also it was utilized in a constant area of industrial usages. Continuous efforts are being made to develop their apply in the transport industry. ^[6]: https://en.wikipedia.org/wiki/Pneumatic_motor



Figure 41: axial piston, radial piston, and rotary vane air motors. ^[7]: http://www.bluetools.com/Air-Tools-Motors/c88_50/index.html

28.2.2 Clutch/Brake:

Electromagnetic clutch or brake (Figure 42: electromagnetic clutch/brakes.^[9]: <http://www.warnerelectric.com> acts electrically but it transfer the torque mechanically. Defined by a device coupling a rotating shaft and a load. The separation of the pregnancy leads to stopping the movement of the shaft.^[8]: <http://www.machinedesign.com/archive/basics-electromagnetic-clutches-and-brakes>



Figure 42: electromagnetic clutch/brakes.^[9]: <http://www.warnerelectric.com>

28.2.3 Stepping Motors:

Stepper motors existing in a variety shape and size (Figure 43: variety shape of stepper motor. ^[10]: <https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor> are a special type of DC motors that move in discrete steps. It consist of many coils arranged into groups called

"phases". Activate each stage in sequence, the engine will spin, one step at a time. The principal of this electromechanical device is to convert one digital pulse into a specific rotational movement or displacement.^[10]: <https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor>



Figure 43: variety shape of stepper motor. ^[10]: <https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor>

28.2.4 AC Induction Motors

The AC induction motor is the electric engine in which the electric current in the rotor is needed for the production of torque obtained by the electromagnetic induction of the magnetic domain of the static coil. The induction motor can be made accordingly without electrical connections to the rotor.

This kind of engine widely utilized for constant speed requirements. Three-phase induction motors (Fehler! Verweisquelle konnte nicht gefunden werden. are vastly utilized in industrial drives because they are rugged, reliable and economical. Single-phase induction motors are widely used for small loads, such as household appliances like fans.^[11]: https://en.wikipedia.org/wiki/Induction_motor



Figure 44: *three phase AC motor.* Fehler! Verweisquelle konnte nicht gefunden werden.

28.2.5 Hydraulic motors:

Hydraulic motor principal based on pressurized oil. A mechanical actuator that converts hydraulic pressure and flow into torque and angular displacement (rotation). Higher pressure results in higher torque (i.e. brute force).

winches and crane drives, wheel motors for military vehicles, self-driven cranes, excavators, conveyor and feeder drives, mixer and agitator drives, roll mills, drum drives for digesters, trommels and kilns, shredders for cars, tires, cable and general garbage, drilling rigs, trench cutters, high-powered lawn trimmers, and plastic injection machines utilize hydraulic motor presently.

We count many kinds of hydraulics like gear motor, vane motor, and axial piston motor (Fehler! Verweisquelle konnte nicht gefunden werden. ... ^[13];

https://en.wikipedia.org/wiki/Hydraulic_motor



Figure 45: Axial piston hydraulic motor. ^[9]: <http://www.warnerelectric.com>

[10]: <https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor>

[11]: https://en.wikipedia.org/wiki/Induction_motor

[12]: <http://electronicsforu.com/buyers-guides/selecting-electric-motor-drive-system>

[13]: https://en.wikipedia.org/wiki/Hydraulic_motor

[14]: <http://www.directindustry.com/prod/hydro-leduc/product-7677-1287099.html>

28.2.6 Servomotors

Servo motor is a standalone electric device with feedback, which can push or rotate parts of a machine with great accuracy. It can rotate an object at certain specific angles or distance. It is made only from a simple engine that operates through a servo mechanism. Servo motor can be DC or AC powered. In a small and light weight packages we can find a very high torque servo motor (Fehler! Verweisquelle konnte nicht gefunden werden..Because of these features they are used to control movement in a variety of electromechanical industries, such as robots, CNC, toy cars, and in space. ^[15]: <https://circuitdigest.com/article/servo-motor-basics>



Figure 46: generic high torque servo.^[16]: <https://www.sparkfun.com/products/11965>

28.3 Technology comparisons

Our study was based on two important types that can be used in aerospace. Hydraulic and servo motor.

The effectiveness of electro-hydraulic motor surround aerospace industry usage. As indicated in (Figure 47: comparison between electro-hydraulic; electro-mechanical; and electro-pneumatic actuator. ; the performance of electro-hydraulic actuator is higher than electro-mechanical and electro pneumatic. That's because electro-hydraulic systems have been designed and sophisticated to achieve every manifested mission. ^[17]: https://www.iei.liu.se/flumes/tmhp51/filearchive/coursematerial/1.105708/HydServoSystems_part1.pdf

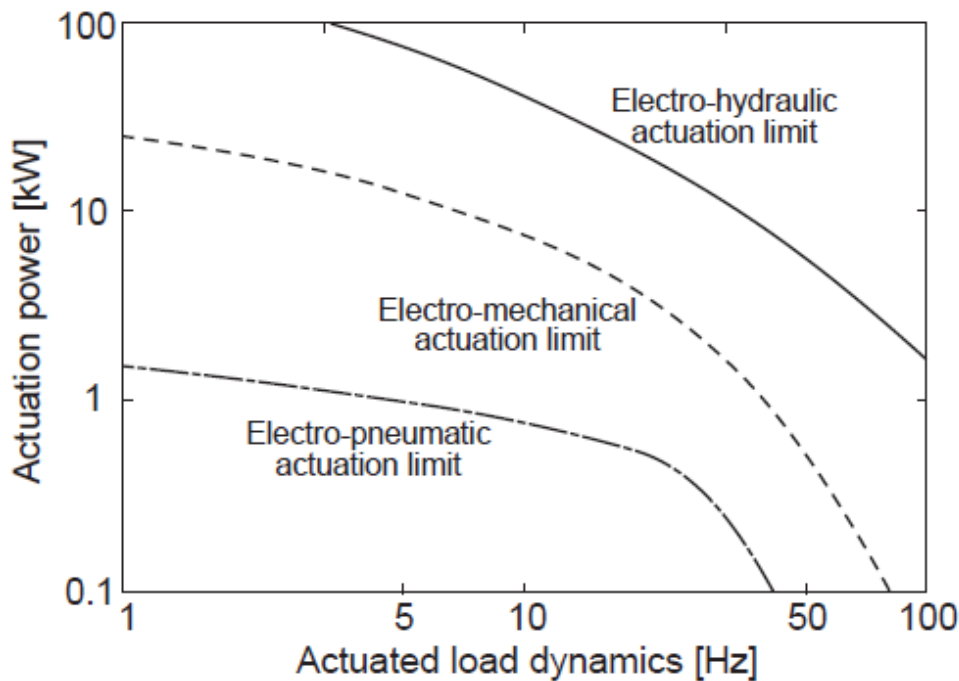


Figure 47: comparison between electro-hydraulic; electro-mechanical; and electro-pneumatic actuator. [17]: https://www.iei.liu.se/flumes/tmhp51/filearchive/coursematerial/1.105708/HydServoSystems_part1.pdf

28.4 First actuator chosen: hydraulic actuator

28.4.1 Applications of Hydraulic Motors

Operation of wing trailing edge flaps and leading edge slats are the most common application of hydraulic motors in aerospace vehicles. In these applications, a hydraulic motor push the flutter or slice via a torque tube that work over the trailing edge or leading edge. Gearboxes (90 degree, bevel and offset gear arrangements) connect the torque tube along the trailing edge or leading edge. Other applications for hydraulic motors are folding wing control, cargo doors and ramps and landing gear. Motors are ordinarily high speed with low torque that are geared down to provide a lower speed and higher torque.

This is why we decided to use it in the airship to control its direction, especially since the outer shape of the airship is very similar to aircraft's one. Fehler! Verweisquelle konnte nicht gefunden werden.

28.4.2 Types of hydraulic motors

We distinguish three kinds of hydraulic engines that are presently utilized; gear, vane and piston motors—with a variety of styles available among them. In addition, several other varieties exist that are less commonly used, gerotor or gerolor are including (orbital or roller star) motors.

Hydraulic motors can be either fixed- or variable-displacement, and work unidirectionally or bi-directionally. While a constant input flow is provided, Fixed-displacement motors drive a load at a constant speed. Variable-displacement engines can offer varying flow rates by changing the displacement. Fixed-displacement motors provide constant torque; variable-displacement designs provide variable torque and speed.

The three different kinds of engines have different characteristics. Gear motors work best at medium pressures and flows, and are usually the lowest cost. Medium pressure ratings and high flows, with a mid-range cost Vane motors, were offered on the other hand. At the most expensive end, piston motors offer the highest flow, pressure and efficiency ratings. In the following; a detailed explanation about the three common types already cited.

28.4.2.1 Gear motor

Gear engines(*Figure 48: external gear motor.* focus on two gears, one being the driven gear—which is joined to the output shaft—and the idler gear. Their work is simple: High-pressure oil is ported into one side of the gears, where it flows around the gears and housing, to the outlet port and pressed out of the motor. Meshing of the gears is a bi-product of high-pressure inlet flow acting on the gear teeth. What actually prevents fluid from leaking from the low pressure (outlet) side to high pressure (inlet) side is the pressure differential. With gear motors, you must be concerned with leakage from the inlet to outlet, which decrease engine capacity and fabricate temperature as well.

In addition to their low cost, gear engines do not drop out as fast or simply as other methods, because the gears wear down the covering and bushings before a tragic deficiency can happen.



Figure 48: *external gear motor*. ^[19] www.mobilehydraulictips.com/hydraulic-motors/

28.4.2.2 Vane Motor

Vanes motor (**Figure 49**: *vane motor*. slide in and out, run by the eccentric bore at the medium-pressure and cost range. The movement of the compressed fluid makes an unbalanced force, which in turn forces the rotor to turn in one direction.



Figure 49: *vane motor*. ^[19] www.mobilehydraulictips.com/hydraulic-motors/

28.4.2.3 Piston type motor

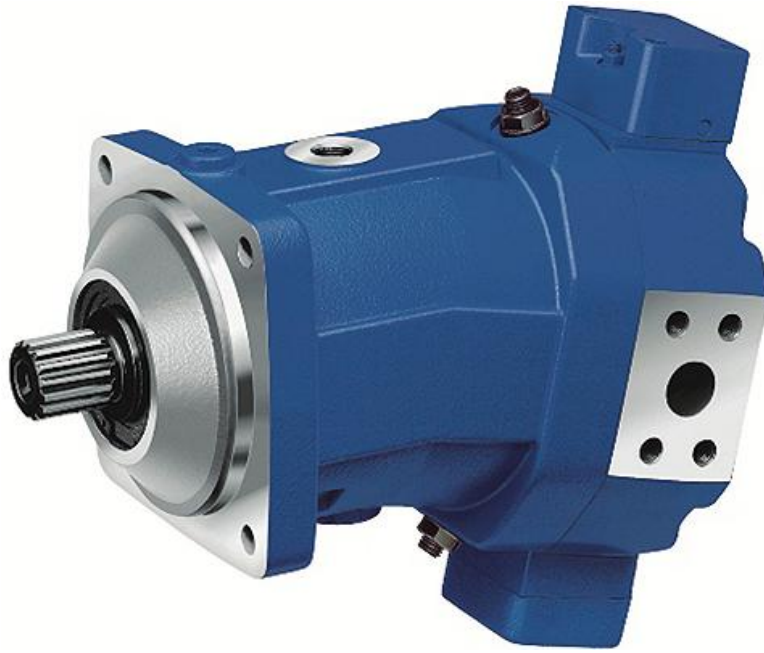


Figure 50: Variable, axial piston motor, with the bent-axis design. ^[19]:
www.mobilehydraulictips.com/hydraulic-motors/

Many sorts of piston motor exists such as including radial (*Figure 51: Radial piston motor.*), axial (*Figure 50: Variable, axial piston motor, with the bent-axis design.* , and others. In the first type; pistons are characterized by its perpendicular configuration to the crankshaft's axis. when the latter enter in rotation motion; pressurized fluid makes pistons in a linear action. The second type; is characterized by his number of pistons coordinated in an orbicular model inside a cover; which in rotation around its axis by a shaft that is in line with the pumping pistons. We identify two designs of axial piston motors ; swashplate and bent axis types. In Swashplate designs the pistons and drive shaft are arranged in parallel. In the next one , the pistons are sloped to the main drive shaft.

Among the lower models used two designs, roller star motors is characterized by a low friction, high efficiency and high start-up torque than gyrator designs. They also provide, low-speed operation and offer longer life with less wear on the rollers. Running and present extended life through low endurance on the pulleys. Gyrators furnish continued fluid-tight sealing over their soft operation. ^[19]: www.mobilehydraulictips.com/hydraulic-motors/



Figure 51: *Radial piston motor.* ^[19]: www.mobilehydraulictips.com/hydraulic-motors/

28.4.3 Details about axial piston hydraulic motor:

A pivotal piston engine is like to an axial piston pump and is the most functional common engine frequently utilized in aerospace, consequent to high power to weight ratio. A schematic of an axial piston engine is shown in Figure 8. As can be seen in the figure, the axial piston pump is similar to a piston pump, excluding that the swashplate (plateau oscillant) corner is presently constant (i.e., there is no compensator and control piston). The entry part of the engine is the high pressure side and the way out is low pressure. The push distinction causes the pump to rotate. Since the swashplate is constant, quickness of this engine is controlled by either controlling entry pressure (Δp across the motor) or the flow rate. Also, such as pumps, hydraulic engines resort to own 9 pistons, or probably 7 (more pistons raise extraction and for this reason increase output torque). Piston motors extend the top sealing for high input pressures and work best in high torque, low speed purposes. They have the best sealing and will be the most effective. An axial piston engine with a constant swashplate is unidirectional (rotate in 1 direction only). To be bi-directional, the swashplate would require to be changeable status. Finally, piston motors will own a case drain line to allow piston drain to spout to return. ^[20]:

28.4.4 Motor Installation Considerations:

The most important parameters for the installation of any motor are listed below.

Dimensions – global dimensions for the motor are intended to establish the desired setup volume.

Interfaces – interfaces contain piping connections and places on the motor, as well as assembly handle and output shaft site.

Weight – weight of the motor, which is commonly provided as a dried weight. When full of with fluid, weight will be higher.

Noise – Motors turn on at high velocity and produce racket. A specialization for extreme noise level should be considered for motor installations.

Attachments – way of connexion of the motor to the airframe affects constructional stiffness of the motor and also noise transition into the airframe.

Motor/Shaft Alignment – put the motor and shaft on one line; needs to hold to tight tolerances. Considerations are tolerance stickups, relative motion between motor and shaft, possible angular displacements on installations, spline teeth dimensions, etc. incorrect alignment can cause exaggerated vibration (leading to premature failure), or failure of the motor seals.

Splines – Usually some grease is applied to the splines to reduce wear. Selection of lubrication should include temperature, corrosion inhibiting and acceptable life of the grease before collapse take place.

Torque Requirements – we should consider start-up and running torque. First one is higher than second. Start-up torque accelerates the mass of the motor and load, leading to temporary high stresses within the mounting hardware and motor. Obviously, the speed of the motor must match the manufacturer's recommended speed for actuation device. A gear arrangement may be used, if in demand.

Axial and Radial Shaft Load Capability – Be sure that the motor shaft and splines are designed specifically to sustain to the loads that motor will confront during its all life. Inlet and outlet fittings must be matched to the motor.

Direction of Rotation – We distinguish two types of motors about direction: uni-directional or bi-directional. So a bi-directional pump is to be used in aerospace application. Clearly, the control valve position must be matched to the suit direction of rotation. ^[21]:

28.5 Second actuator chosen: servo actuator

28.5.1 Reason for choose of servo actuator:

Servomotors are not so old; they can therefore be integrated into various applications. Despite their small dimensions; they offer a hard rap and a very good energy yield. With these specifications; their uses extend into the field of toy automobiles; robots, radio control aircraft; as well as in industry such as robotics; pharmacy; etc. ^[22]:
<http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html>

They are also used in powerful heavy sailing boats. The servos are rated for speed and torque. Although these motors exist in different sizes but their patterns are similar. These small motors are extremely powerful compared to their size; and their feed for a load is suitable. For this; generally a servo loaded by a small element is a well valued energy. ^[23]

So that all we came to mention leads to replace the hydraulics with the servo because "it has some characteristics of the aircraft and some other of robots. Also, the lightness of its weight and strength that is good, in addition to the light exchange of energy make it overcome the hydraulic and replace it in this Type of applications.

28.5.2 General information about servo:

A servomotor is a type of motor that incorporates in the same housing the mechanics (DC motor) and the simplified control electronics, generally servo-controlled in position with a limit of 180 degree angle travel, but also available in continuous rotation .

Its advantage is the ease of control by an external digital signal and the high gearing that the gears integrated in its housing allow.

these motors are usually used to move pieces (sails, capstans, drifts, control surfaces, flaps) of models, boats or planes, and are recognizable thanks to their standardized box, rectangular and black.

They are therefore used as actuators or for the motorization of small robots, possibly by modifying their mechanics so that they rotate continuously, thus eliminating the servo-control in position. ^[24]: <http://www.pobot.org/+servomoteur+.html?lang=fr>

Usually; servo motors comes with arms (metals or plastic) that is connected to the object required to move (see figure below to the right).

Servo have 3 wires:

Black wire: GND (ground); RED wire: +5v; Colored wire: control signal (*Figure 52: scheme for the three wire of servo.*).

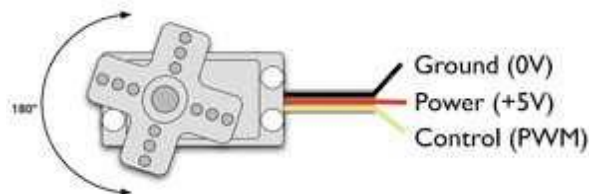


Figure 52: scheme for the three wire of servo. ^[25]: <https://www.scribd.com/document/99583469/Introduction-to-Servo-Motors-Arduino>

The third pin accepts the control signal which is a pulse-width modulation (PWM) signal. It can be easily produced by all micro- controllers and an Arduino board. This accepts the signal from your controller that tells it what angle to turn to. The control signal is fairly simple compared to that of a stepper motor. It is just a pulse of varying lengths. The length of the pulse corresponds to the angle the motor turns to.

The pulse width sent to servo ranges as follows:

Minimum: 1 millisecond ---> Corresponds to 0 rotation angle.

Maximum: 2 millisecond ---> Corresponds to 180 rotation angle.

Any length of pulse in between will rotate the servo shaft to its corresponding angle. For example, 1.5 ms pulse corresponds to rotation angle of 90 degree (*Figure 53: scheme explaining the pulse width and its corresponding angle.*).

This is will explained in figure below. ^[25]:

<https://www.scribd.com/document/99583469/Introduction-to-Servo-Motors-Arduino>

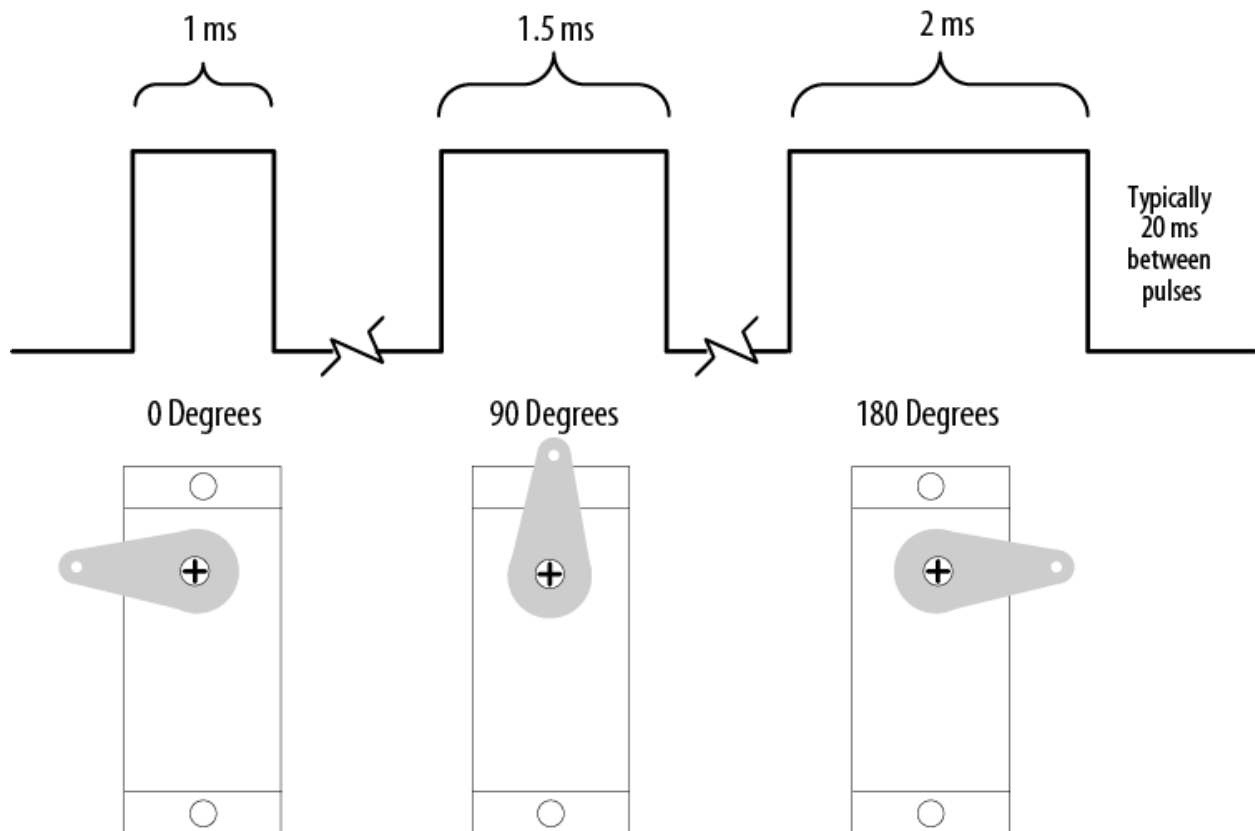


Figure 53: scheme explaining the pulse width and its corresponding angle. ^[25]:

<https://www.scribd.com/document/99583469/Introduction-to-Servo-Motors-Arduino>

28.5.3 Types of servo motors

There are three Main Types of servo motors which are:

Positional Rotation Servo: (*Figure 54: positional rotation servo.*

This variety rotates within a 180° range. It's not designed to turn beyond its preset limits.

Useful for limited-range applications like moving levers or steering linkages.



Figure 54: *positional rotation servo.* ^[26]:

Continuous Rotation Servo: (Figure 55: *continuous rotation servo.*

While levers are often used on standard servos, wheels and gears become more useful with this style, which can turn in any direction independently and continuously.

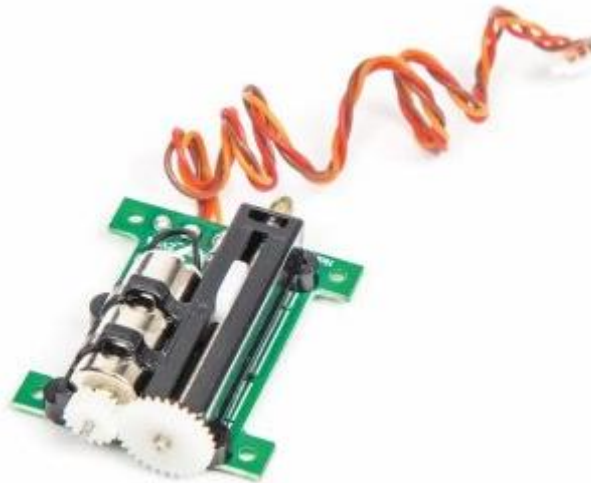


Figure 55: *continuous rotation servo.* ^[26]:

Linear Servo: Fehler! Verweisquelle konnte nicht gefunden werden.(Figure 56: *Spectrum Mini servo linear servo Gear box material.*

This type offers more gears than the positional rotation servo, but is otherwise very similar. It uses a rack and pinion mechanism to change the output back and forth instead of circularly. This servo is rare, but can be found in larger hobby planes and robots. ^[26]:



Figure 56: *Spectrum Mini servo linear servo Gear box material.* [26];

29 Actuator System of TEMOLeb-Mintad

29.1 Hydraulic actuator system

29.1.1 Work principal for hydraulic motor

In the figure below (**Figure 57: simple work principal of hydraulic actuator.**; two cylinders are used; to understand the principle of hydraulic actuator operation. In fact; the section of the left cylinder is 1 square centimeter; and that of the right is 10 square centimeters. The two cylinders are filled by an incompressible fluid. If a pressure unit is applied to the left cylinder, which is used to push the pump; of 10 centimeters; the effect of this force on the right cylinder is to push the piston one centimeter with a unit force. Which implies that a unit of force applied on one side of the cylinder provides 10 units of this force on the other side. This relation results from the Pascal principle which shows us the proportionality between force and pressure:

$$P = \frac{F}{S}$$

This type of cylinder (or called a linear hydraulic motor) is useful in systems that require high strength but not much accuracy. For example, if you need a linear force to move a robot. The action of the hydraulic pressure on these actuators provides a reciprocating movement of the piston included in the cylinder. And since the load is attached to the piston, it will move in parallel with its motion. Even a rotating action can be created by transforming the hydraulic pressure.

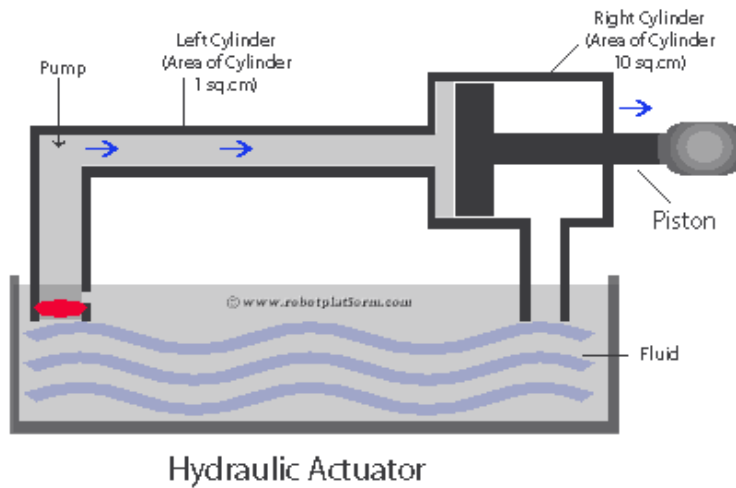

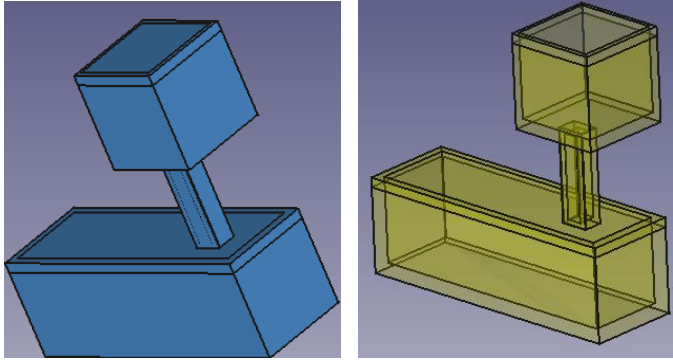

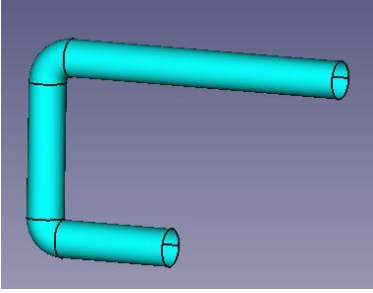

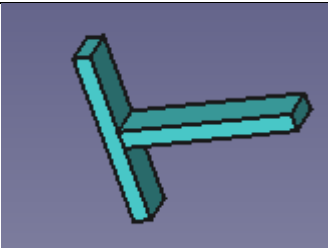


Figure 57: simple work principal of hydraulic actuator.

29.1.2 FreeCAD design proposal

Parts and details	Date and FreeCAD file	Screenshot
Oil tank	April /2017  oil_tank.FCStd	
Pipes	April/2017  pipes.FCStd	
Piston	April/2017  piston.FCStd	


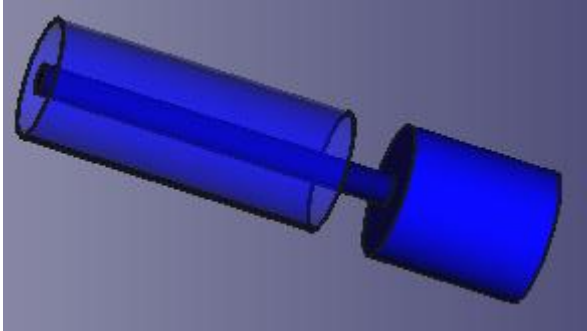
Pump Electric Motor	May/2017  pump_22 5 2017.FCStd	
---------------------------	---	--

Table 11: freeCAD design for each part of the proposal hydraulic actuator system.

The whole motor is drawn below:


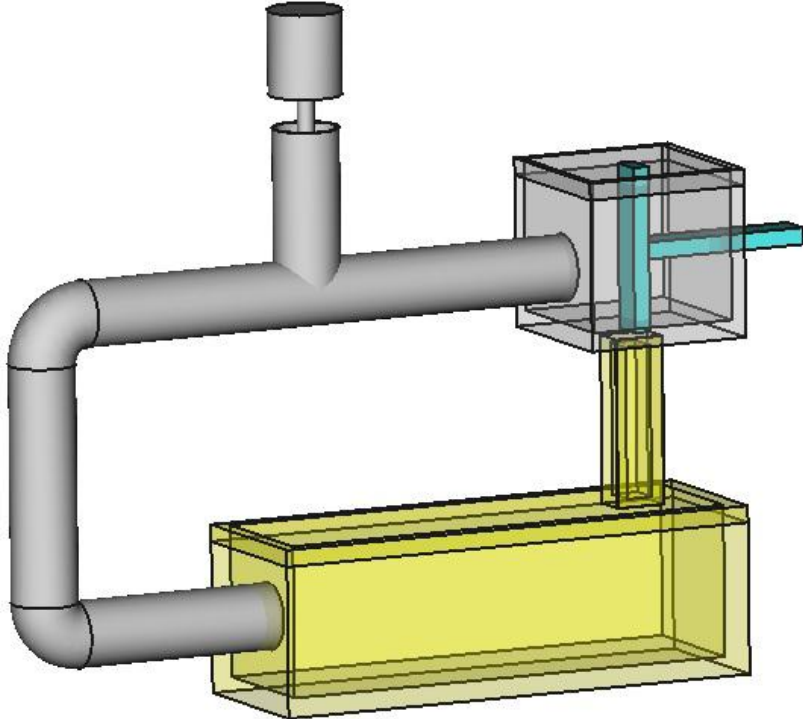
 whole_hydraulic_mot or_22_5_2017.FCStd	
---	---

Table 12: freeCAD model of whole actuator.

We later discovered that this type of actuator who drives the piston in one direction is not enough to move the rudder and elevator in two different directions, when the airships control system gives the command. And to do that, we need a system acting in both directions such as the “double acting cylinder”. This last isn’t a very easy system; a lot of problems we face if we are to manufacture this type... Even if we want to buy it.

29.2 Servo actuator system

29.2.1 Adopted Motor (*Figure 58: the servo motor which we buy.*)



Figure 58: the servo motor which we buy.

29.2.2 Basic parts of the servo (*Figure 59: picture showing all interior parts for any servo (motor, gearbox, potentiometer, and control electronics).*)

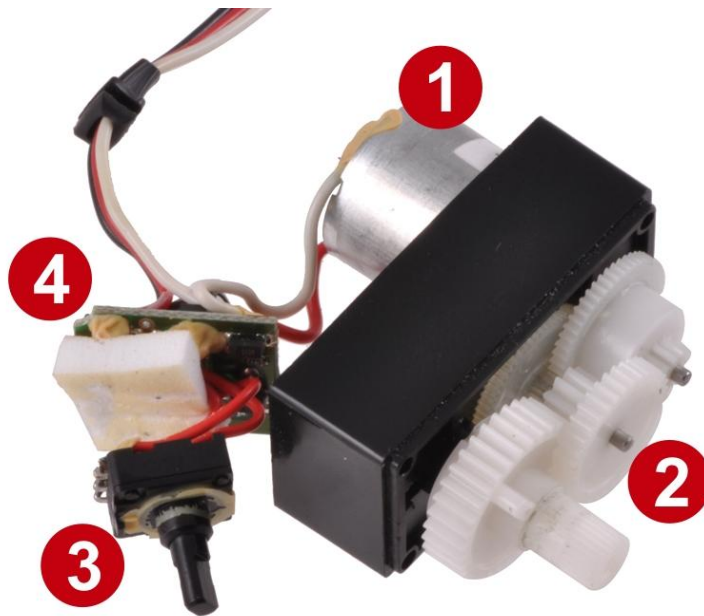


Figure 59: picture showing all interior parts for any servo (motor, gearbox, potentiometer, and control electronics).

1. Motor
2. Gearbox
3. Position sensor
4. Motor control electronics

29.2.3 Servo motor block diagram (Figure 60: servo motor block diagram.

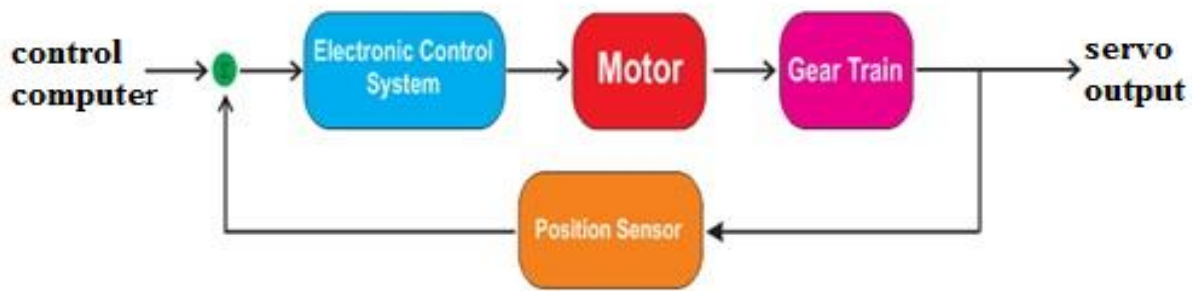


Figure 60: servo motor block diagram.

29.2.3.1 Control computer:

The mobile application used “BlunoBasicDemo”; send an electronic control signal containing the angle of each servo and the on or off for propeller.

29.2.3.2 Electronic control system:

It is an interconnection between 2 signals: input and output. A signal is transformed to another one using a process; in the objective to create motion, change a speed, etc. specifically, in our project we use the type called closed loop electronic control system. By correcting the error occurred; it helped us to main the system more stable and enhance its control. In the following a scheme (Figure 61: the concept of electronic control system. is showed to explain this concept.

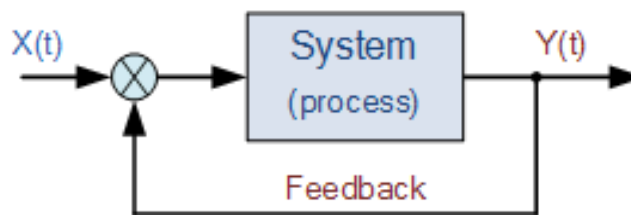


Figure 61: the concept of electronic control system.

29.2.3.3 Motor:

Inside the servo is a DC motor. This one will turn when receiving an electric signal from the control system.

29.2.3.4 Gear train:

An assembly of many gears; receive the rotation force from DC motor; and transform it into torque.

29.2.3.5 Position sensor:

Such as a potentiometer; which has to learn continuously the mechanical position of the shaft by changing the resistance of an interior resistor.

29.2.3.6 Servo output:

A PWM signal was sent to the motor, turn the shaft in the desired position. This can describe the output of a servo.

29.2.4 Design of servo actuator system (FreeCAD)


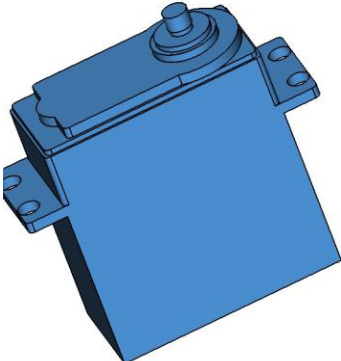

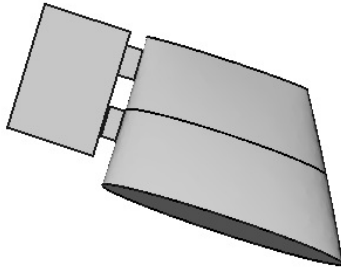
Parts and details	Date and FreeCAD file	Screenshot
Servo Motor	 servomotor.FCStd	
Rudder	 rudder.FCStd	

Table 13: freeCAD model of servo motor and the airship rudder.

29.2.5 Motor Controller and Interfaces

The work which is described in this section was done by CNCLab (see quotation in Appendix).



The mobile application named “BlunoBasicDemo” sends a command to the Bluetooth module taking place on the Arduino board. It passes through the motor drive then arrive to the servo actuator in the form of PWM signal which is responsible for rotating the servo in different angles.

29.2.5.1 Mobile App:



Figure 62: *BlunoBasicDemo* logo.

BlunoBasicDemo (Figure 62: *BlunoBasicDemo* logo. is a basic Demo for Bluno including all the code and executable on Android, IOS and Android. You can easily develop your own with this Demo.

29.2.5.2 Bluetooth Module: (BLE Link -A Bluetooth 4.0 module for Arduino)

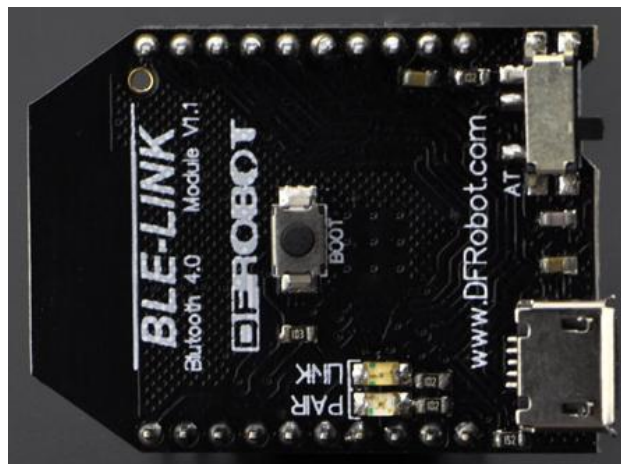


Figure 63: *BLE Link -A Bluetooth 4.0 module for Arduino.*

Description:

It is a peripheral that quickly connects the device to the mobile via BLE, using the XBEE model suitable for all XBEE screens. Many of the applications for Android and IOS systems are developed to be at the service of users who must use it to communicate between BLE and Arduino.

Specification:

Price: 23.95\$

Mark: DFRobot

Serial number: WRL0024

Chip: TI CC2540>

Working voltage: +3.3DC

Power consumption: working 10.6 mA average, ready mode: 8.7 mA

Pin Layout: Compatible With XBEE pinout

Frequency: 2.4 GHz

Transfer rate: 1 Mbps

Modulation: GFSK, Bluetooth low power, V4.0

Sensitivity: -93dB

Operating temperature: - 40 → +85

Transmission distance: 60 m in free space

Size: 32mm * 22mm (1.26 * 0.87")

2.2.5.3. Arduino controller: Romeo V2-All in one Controller-motor drive built in

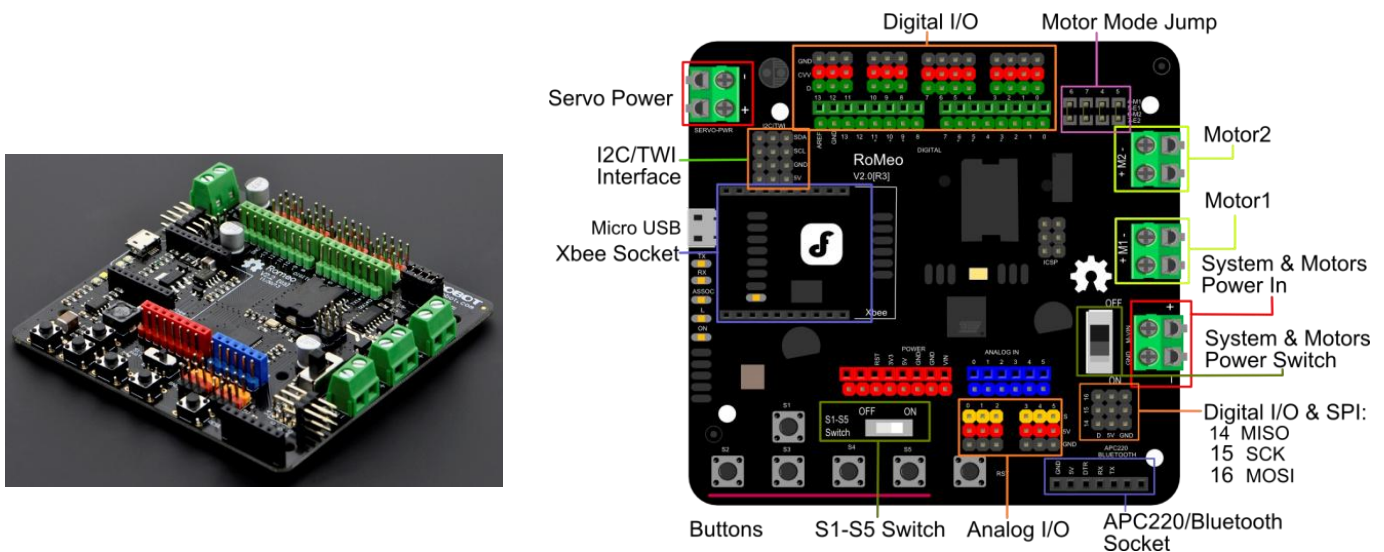


Figure 64: Romeo V2-All in one Controller-motor drive built in.

Description:

Romeo V2 [R3] (Figure 64: Romeo V2-All in one Controller-motor drive built in.) is an All-in-One Arduino, microcontroller, made specifically for robotics applications. This type of Arduino based on the ATmega32u4 chip, can be programmed fast via the Arduino IDE. Thanks to ATmega32u4 chip, the ease and the simplicity of RoMéo V2. Another application of Romeo V2 is that it can control a stepper motor.

Specification:

Price: 46.95 \$

Mark: DFRobot

Serial number: DEV0005

DC Supply: USB Powered or External 6V ~ 23V DC

DC Output: 5V (2A) / 3.3V DC

Motor driver Continuous Output Current: 2A

Microcontroller: ATmega32u4

Bootloader: Arduino Leonardo

Compatible with the Arduino R3 pin mapping

Analog Inputs: A0-A5, A6 - A11 (on digital pins 4, 6, 8, 9, 10, and 12)

PWM: 3, 5, 6, 9, 10, 11, and 13. Provide 8-bit PWM output

5 key inputs for testing

Auto sensing/switching external power input

Serial Interface

TTL Level

USB

Support Male and Female Pin Header

Built-in XBEE socket

Integrated sockets for APC220 RF Module and DF-Bluetooth Module

Three I2C/TWI Interface Pin Sets (two 90° pin headers)

Two way Motor Driver with 2A maximum current

One Stepper Motor Drive with 2A maximum current

Size: 89 x 84 x 14mm

2.2.5.4. Motor drive: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N

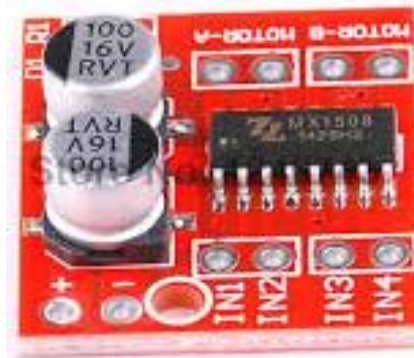


Figure 65: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N.

Description:

HR 2 motordrive (*Figure 65: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N.*) can be used in a device need a voltage between 2 and 10 v. with his 2 pin each to 1.5 A DC current ; it can manage the position and speed control.

Note that in this project motor drive is used only for DC motor which's responsible of propeller motion and speed (i.e. servo motor don't need motor drive, it will be directly connected to arduino).

Specification:

Price: 3.95\$

Serial number: DRV0023

H bridge motor dual drive, and can drive two DC motor or 1 line 4 phase stepper motor;

The voltage of the power supply module 2V-10V;

The signal input voltage 1.8-7V;

Single channel current of 1.5A, peak current up to 2.5A, low standby current (less than 0.1uA);

The built-in common conduction circuit, the input end is suspended, the motor will not malfunction;

The built-in overheat protection circuit with hysteresis effect (TSD), there is no need to worry about motor stall;

Product size: 24.7*21*5mm (LxWxH), ultra-small size, suitable for assembly and vehicle;

Mounting hole diameter: 2 mm.

Weight: 5g

29.2.6 Power Management: Polymer Lithium Ion Battery - 1000mah 7.4v



Figure 66: Polymer Lithium Ion Battery - 1000mah 7.4v.

Description:

This LiPo (*Figure 66: Polymer Lithium Ion Battery - 1000mah 7.4v.*) is an excellent battery that can be used in any application; who need a little power supply has several punch as in robotics. Its low voltage and sufficient flow allows it to acquire many electronic and some small motors

The battery has two cells and produces 7.4 V to store 1000 mAh of charge. This type of batteries requires a specific charger.

Specification:

Price: 13.95\$

Mark: SparkFun Electronics®

Serial number: PWR0024

7.4V 2-cell pack

1000mAh of charge

Discharge rate: 25C continuous

Charge plug: JST-XH

Discharge plug: JST-RCY

Dimensions: 70mm x 35mm x 18mm

Weight: 85g (2.99oz)

29.3 Software Development on Arduino side with the Arduino integrated development environment (IDE)

29.3.1 Architecture of the program on the Arduino

Arduino IDE is a software which can be used to program an Arduino board; with C or C++ languages; referring to its special programming rules. It supplies a software library from the Wiring project, which provides many common input and output procedures.

To well writing the code; two functions are required:

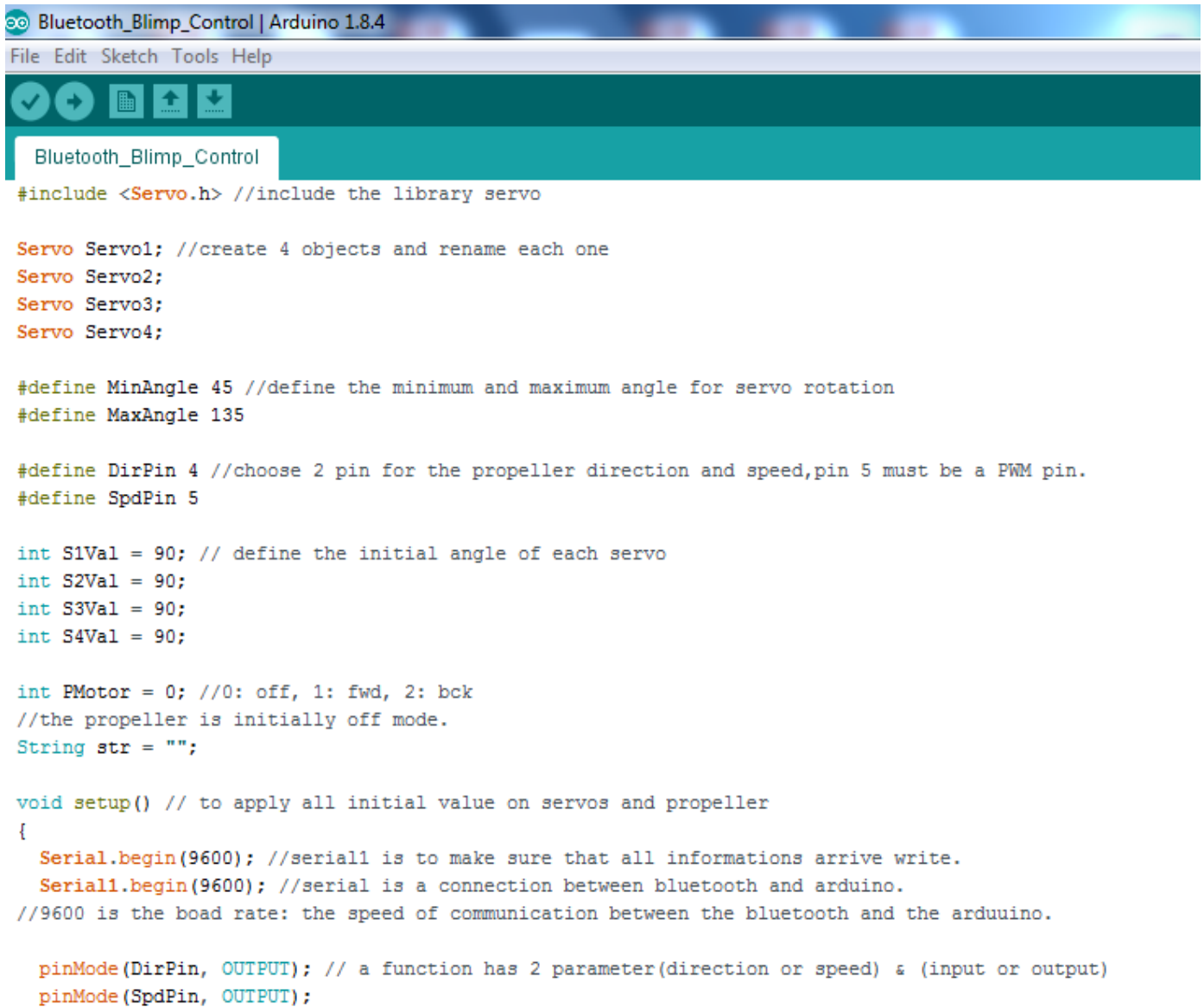
Setup (): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

Loop (): is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

Another functions furnished by the internal libraries; are also used in this program: `pinMode ()`, `digitalWrite ()`, and `delay ()`.

The program code is then converted into a text file; in hexadecimal encoding using a specific program; that is loaded into the Arduino board by a loader program in the board's firmware. So you will give short strokes at a click; and the program becomes uploaded.

29.3.2 Bluetooth_Blimp_Control: Code and explication



```
Bluetooth_Blimp_Control | Arduino 1.8.4
File Edit Sketch Tools Help

Bluetooth_Blimp_Control
#include <Servo.h> //include the library servo

Servo Servo1; //create 4 objects and rename each one
Servo Servo2;
Servo Servo3;
Servo Servo4;

#define MinAngle 45 //define the minimum and maximum angle for servo rotation
#define MaxAngle 135

#define DirPin 4 //choose 2 pin for the propeller direction and speed,pin 5 must be a PWM pin.
#define SpdPin 5

int S1Val = 90; // define the initial angle of each servo
int S2Val = 90;
int S3Val = 90;
int S4Val = 90;

int PMotor = 0; //0: off, 1: fwd, 2: bck
//the propeller is initially off mode.
String str = "";

void setup() // to apply all initial value on servos and propeller
{
  Serial.begin(9600); //serial1 is to make sure that all informations arrive write.
  Serial1.begin(9600); //serial is a connection between bluetooth and arduino.
  //9600 is the boad rate: the speed of communication between the bluetooth and the arduino.

  pinMode(DirPin, OUTPUT); // a function has 2 parameter(direction or speed) & (input or output)
  pinMode(SpdPin, OUTPUT);
```


Building of the frame including actuators for a small prototype of an airship (2017)

```
Bluetooth_Blimp_Control | Arduino 1.8.4
File Edit Sketch Tools Help
Bluetooth_Blimp_Control

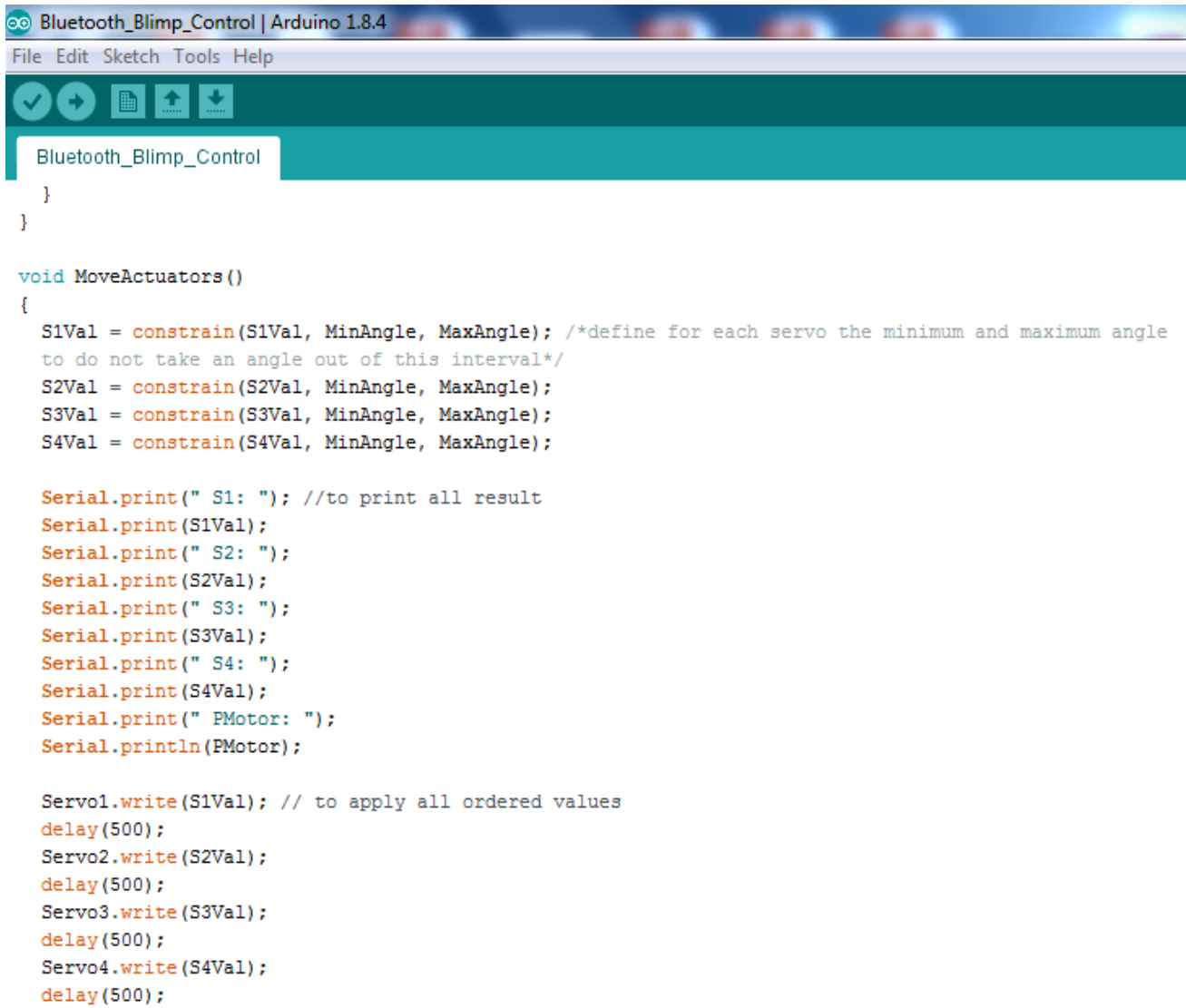
// digitalWrite(DirPin, LOW);
analogWrite(SpdPin, 0); // to be sure that the initial speed is zero.

Servo1.attach(6); //to choose the servos pins between 14 pins in the arduino board; which must be PWM pin.
Servo2.attach(9);
Servo3.attach(10);
Servo4.attach(11);

Servo1.write(S1Val); //to give each servo his initial value.
delay(100); /*a pause between the motion of each 2 servos(so as not be exposed to a shortage
of electricity if the 4 servos start together)*/
Servo2.write(S2Val);
delay(100);
Servo3.write(S3Val);
delay(100);
Servo4.write(S4Val);
}

void loop()
{
  if (Serial1.available())/*when an ordedr arrived to the bluetooth port this function become valid
so the processor enter into it and start to performs all steps inside */
  {
    S1Val = Serial1.parseInt(); //parsint: whatever the value was it will be taked as an integer
    S2Val = Serial1.parseInt(); /*to check the value arrived from the bluetooth to serial1
and send it to sival(value for servol)*/
    S3Val = Serial1.parseInt();
    S4Val = Serial1.parseInt();
    PMotor = Serial1.parseInt();
    MoveActuators();
  }
}
```

Building of the frame including actuators for a small prototype of an airship (2017)

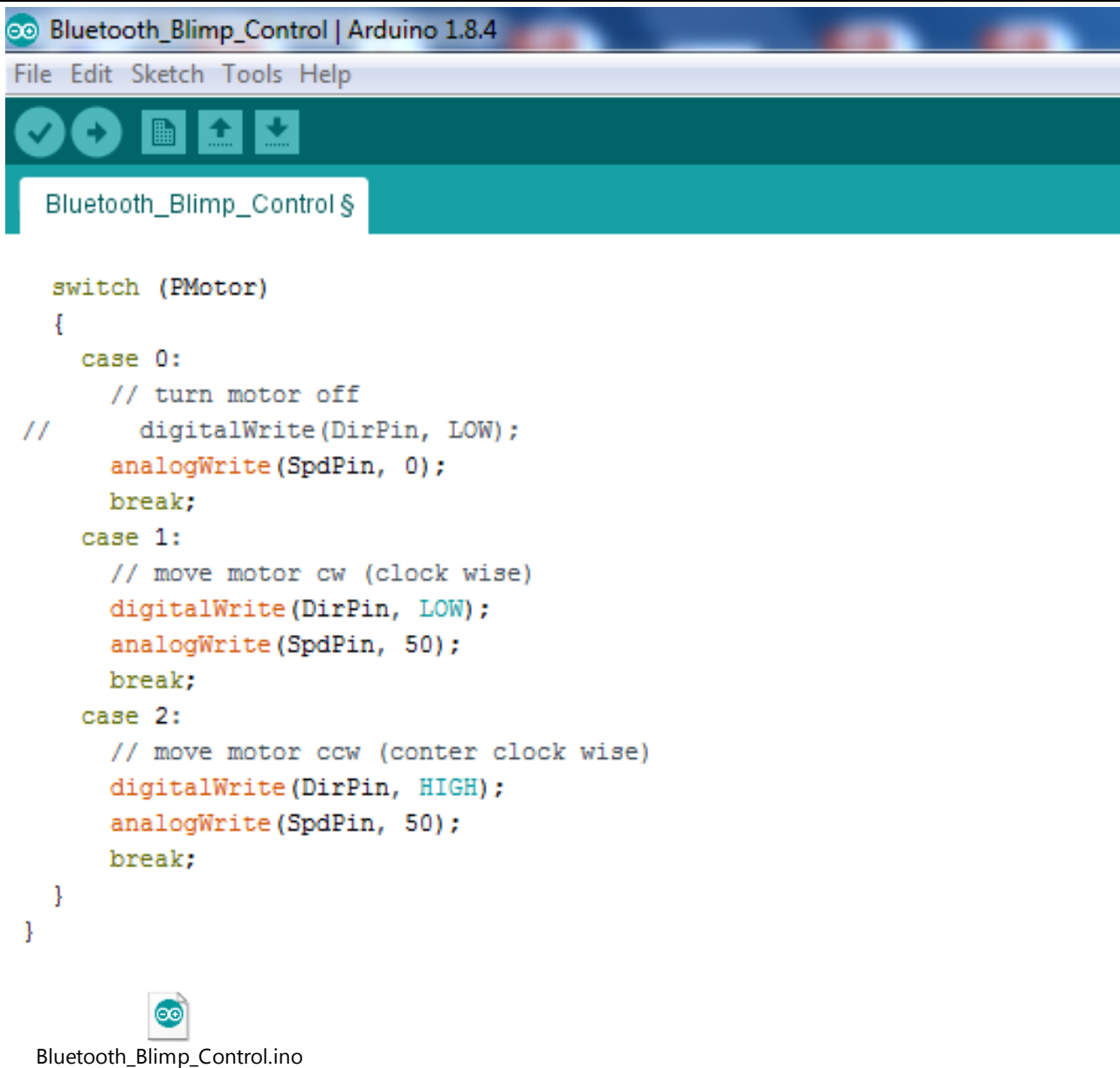


```
Bluetooth_Blimp_Control | Arduino 1.8.4
File Edit Sketch Tools Help
Bluetooth_Blimp_Control
}
}

void MoveActuators()
{
  S1Val = constrain(S1Val, MinAngle, MaxAngle); /*define for each servo the minimum and maximum angle
  to do not take an angle out of this interval*/
  S2Val = constrain(S2Val, MinAngle, MaxAngle);
  S3Val = constrain(S3Val, MinAngle, MaxAngle);
  S4Val = constrain(S4Val, MinAngle, MaxAngle);

  Serial.print(" S1: "); //to print all result
  Serial.print(S1Val);
  Serial.print(" S2: ");
  Serial.print(S2Val);
  Serial.print(" S3: ");
  Serial.print(S3Val);
  Serial.print(" S4: ");
  Serial.print(S4Val);
  Serial.print(" PMotor: ");
  Serial.println(PMotor);

  Servo1.write(S1Val); // to apply all ordered values
  delay(500);
  Servo2.write(S2Val);
  delay(500);
  Servo3.write(S3Val);
  delay(500);
  Servo4.write(S4Val);
  delay(500);
}
```



29.4 Final assembly

Each part which was described and discussed in the previous chapter will be seen in this chapter in the final assembly; which contains: the programmed arduino,4 servos; bluetooth module;DC Motor; and the battery. All that we have mentioned appears in the following photo (*Figure 67: the final assembly of servo actuator system and its interfaces.*).

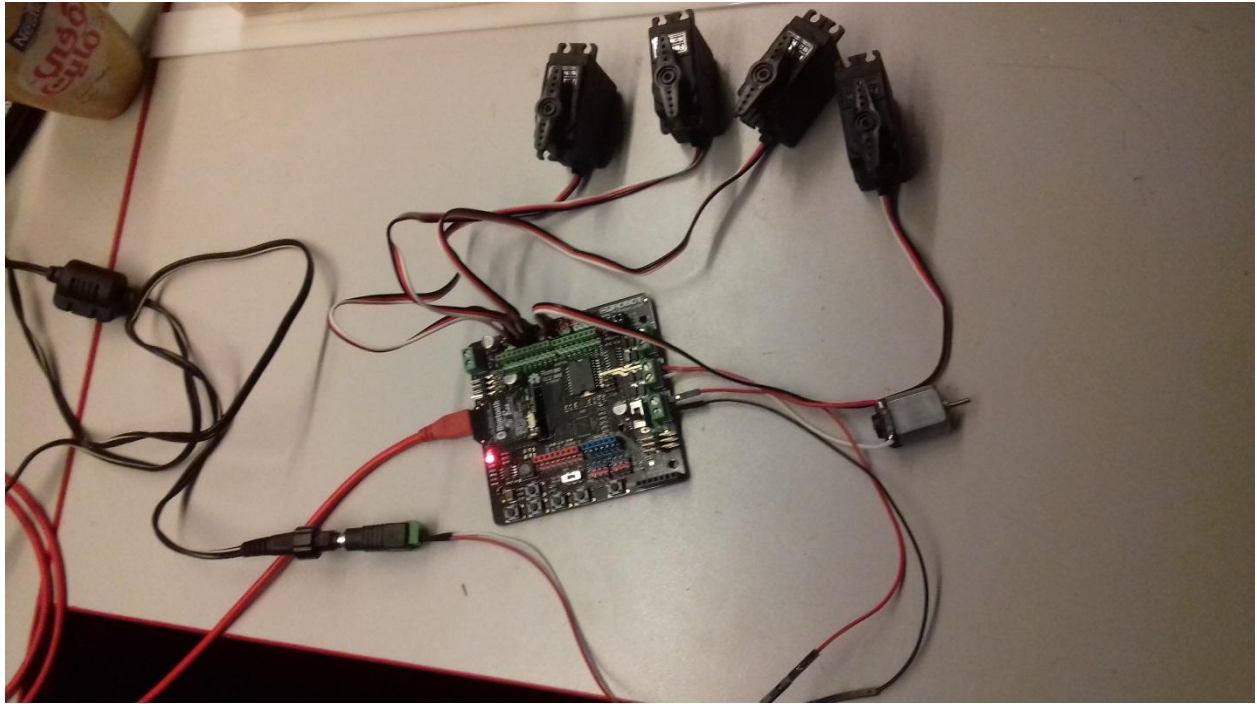
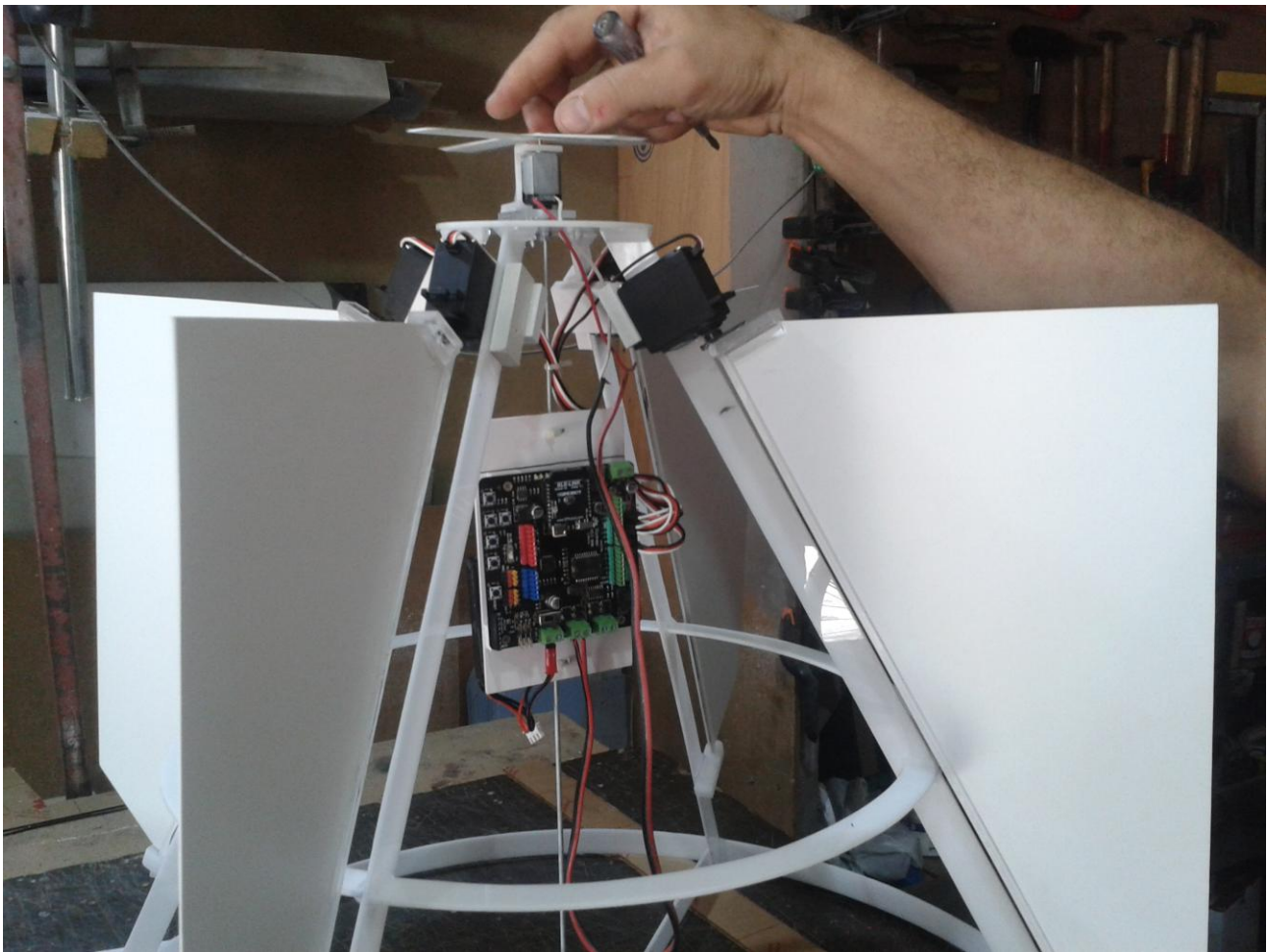


Figure 67: the final assembly of servo actuator system and its interfaces.

In effect the program already written will then be uploaded to the Arduino board. Once the upload is finished; and the Arduino is supplied by the compatible power; all servos will turn to their initial position (90°) as defined in the code.



29.5 Testing the integrated actuator

To test this assembly there are several steps to follow. First; make sure that all peripheral devices are well connected to the Arduino board; and the board is supplied by the compatible battery. The light seen on the Arduino confirms the arrival of current into the board. Then; verify the position of all servos on PWM pins (6, 9, 10, and 11) and the DC motor position. Later; start to connect your mobile Bluetooth to BLE link Bluetooth.

And now we start the control using the mobile application “BlunoBasicDemo”. To do that you must enter 5 number between each 2 numbers a comma. The 4 first numbers will be the rotation angle of all 4 servos; which must be between the minimum (45°) and the maximum (135°) angles. And the last number can be 0 (to turn off the propeller’s motor) and 1 (to turn it on).

This process is illustrated in the following figure: *(Figure 68: showing how to use BunoBasicDemo to give order to our system.*



Figure 68: showing how to use BunoBasicDemo to give order to our system.

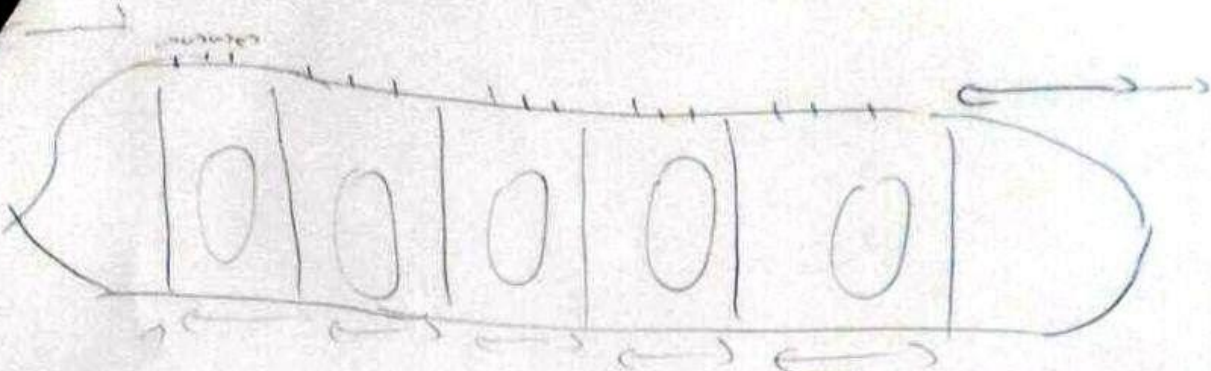
Each servo will turns in the required angle.



30 Prototype Construction



Aluminium Pipes for skeleton



$\bigcirc \times 6 = (8 + 2\pi r) \cdot 6 = \overset{123,398 \text{ m}}{323,0682 \text{ m}}$
 $+ \bigcirc \times 15 = (2\pi r) \cdot 15 = 188,495 \text{ m}$
 $+ \text{---} \times 10 = 25 \cdot 10 = 250 \text{ m}$
 Total = 561,893 m.

$\left| \left(\frac{m}{L} \right)_{\frac{70}{561,893}} = 0,125 \text{ kg/m} \right|$

$V \left(\text{---} \xrightarrow{1 \text{ m}} \bigcirc \right) = \pi r^2 \cdot l =$

$M \left(\text{---} \xrightarrow{\quad} \bigcirc \right) = V \cdot \rho_{Al} = \pi r^2 \cdot \rho_{Al}$

$\Rightarrow r = \sqrt[3]{\frac{M}{\rho_{Al} \cdot \pi}} = 0,242 \text{ kg m}$

$d = 8,412 \text{ mm}$ | Al.
 $d = 1,156 \text{ cm}$ | Plexi.

$\rho_{N_2O_4} = 0,4 \text{ kg/m}^3$

Oberfläche einer Kugel = $A_0 = 4\pi r^2$

Volumen einer Kugel = $V = \frac{4}{3}\pi r^3$

$\rho_{He} = 0,1785 \text{ kg/m}^3$

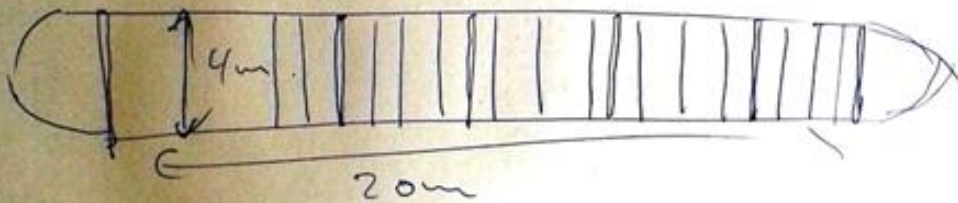
$\rho_{Luft} = 1,184 \text{ kg/m}^3$

Archimedes Satz $\rightarrow F_{\text{Auftrieb}} = V_{\text{verdrängt}} \cdot \rho_{\text{Luft}} \cdot g$

Gewicht eines Ballons = $m_{N_2O_4} + m_{He} = A_0 \cdot m/A + \rho V =$

Gewicht des Gerüsts $\approx 64,76 \text{ kg} \cdot \frac{917,64 \text{ kg}}{142,74 \text{ kg}}$

Tragkraft aller Ballons = $5 \cdot 13,57516 = 67,8758 \text{ kg}$



$12 \times 25 + 20 \times 2\pi \cdot 2 + 6 \cdot 2 \cdot 4 =$

$300 + 263,76 + 48 = 611,76 \text{ m}$

$A_0 = 4\pi \cdot 4 = 50,24 \text{ m}^2$

$V = \frac{4}{3}\pi \cdot 8 = 33,49 \text{ m}^3$

$m_{N_2O_4} = A_0 \cdot m/A(N_2O_4) = 50,24 \cdot 0,4 = 20,096 \text{ kg}$

$m_{He} = \rho V = 1,184 \cdot 33,49 = 39,71 \text{ kg}$



$$V = \frac{4}{3} \pi r^3$$

$$\text{Mantelfläche} = A = 4 \pi r^2$$

$$\text{Archimedes} = F_{\text{Auftrieb}} = V_{\text{verdrängt}} \cdot \rho_{\text{Fluid}} \cdot g$$

$$\rho_{\text{plexi.}} = 1,19 \text{ g/cm}^3$$

$$\rho_{\text{H}} = 0,0899 \text{ kg/m}^3$$

$$\rho_{\text{He}} = 0,1785 \text{ kg/m}^3$$

$$\rho_{\text{Alu}} = 2,6989 \text{ g/cm}^3 = 2,6989 \frac{10^3 \text{ kg}}{10^3 \text{ m}^3}$$

$$\rho_{\text{Luft}} = 1,293 \text{ kg/m}^3$$

$$\left(\frac{m}{A}\right)_{\text{Nylon}} = 400 \text{ g/m}^2$$

$$V_{\text{Ballon}} = \frac{4}{3} \pi r^3 = \frac{4}{3} \pi (2)^3 = 33,510 \text{ m}^3$$

$$(\text{Mantelfläche})_{\text{Ballon}} = 4 \pi r^2 = 4 \pi (2)^2 = 50,265 \text{ m}^2$$

$$F_{\text{Auftrieb}} = V_{\text{Ballon}} \rho_{\text{Luft}} \cdot g = 425,052 \text{ N}$$

$$m_{\text{Ballon}} = m_{\text{He}} + m_{\text{Ballonhülle}} =$$

$$= V_{\text{Ballon}} \cdot \rho_{\text{He}} + \left(\frac{m}{A}\right)_{\text{Nylon}} \cdot (\text{Mantelfläche})_{\text{Ballon}}$$

$$= 33,510 \cdot 0,1785 + 0,4 \cdot 50,265$$

$$= 26,088 \text{ Kg}$$

$$(\text{mögliche getragene Masse})_{\text{Ballon}} = \frac{F_{\text{Auftrieb}}}{g} - m_{\text{Ballon}}$$

$$= 43,32843 - 26,088$$

$$= 17,240 \text{ Kg}$$

$$(\text{mögliche getragene Masse})_{\text{Ballon}} = 17,240 \cdot 0,95$$

$$= 16,378 \text{ Kg}$$

$$(\text{Mögliche Nutzlast}) \approx 16,378 \text{ Kg}$$

Appendix C: for Actuator System

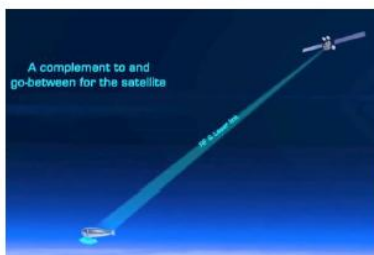
Initial Working packages



Ras Nhache, Batroun/Tripoli, 10.03.2017

The TEMOLEbanon-Mintad project aims to offer internet access via a airship platform. There are the following Milestones for 2017/2018:

- Sep 2017: Low-altitude flying airship (1-3 km) for internet supply is in operation (above Qalamoun/Ras Nhache) (prototype is developed with some master thesis')
- 2018 High-altitude platform (20-22 km) for internet supply in cooperation with Turkey



Master Thesis: Solar-power based actuator system for an airship

	<p>Energy Supply Generation System: Solar plates -> Electrolyzer -> Hydrogene Gas tank -> Fuel Cells</p>
	<p>Actuator System: Paddles, Propeller</p>
<p>Literature: https://partners.ni.com/directory/solution/details.aspx?id=63&tab=overview http://www.ideal-aerosmith.com/motion/model-hla-hp-hydraulic-linear-actuator-high-performance https://partners.ni.com/directory/solution/details.aspx?id=63&tab=overview</p>	

Tasks:

- Detailed Modeling of elements Energy Supply Generation System and Actuator System with FreeCAD
- Contruction of a test system for Energy Supply Generation and Actuator System

Contact: Eng. Samir Mourad, Mob. +961 76 341526 (Lebanon), WhatsApp +49 178 7285578 (Mob. Germany)

Quotation for Arduino based actuator system

All material and interfaces from: www.cnclablb.com

Invoice

Date	Invoice #
9/6/2017	117-53

Bill To
Dr. Samir Mourad

Ship To

P.O. Number	Terms	Ship	Via	Customer Price Level
		9/6/2017	Aramex	

Item Code	Description	Quantity	Price Each	Amount
ACT0057	HR Servo S3003	4	7.95	31.80
DEV0005	DFRduino Romeo V2 - motor drive built in	1	44.73	44.73
WRL0024	BLE Link Bluetooth Bee	1	23.95	23.95
PWR0024	Polymer Lithium Ion Battery - 1000mah 7.4v	1	13.95	13.95
ACT0059	HR DC toy motor 3V-6V	1	1.95	1.95
Consultancy	Consultancy Fees Hourly Rate	1	100.00	100.00
Total				\$216.38

Appendix D: For Prototype Construction

Initial Working Packages



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



Ras Nhache, Batroun/Tripoli, 27.02.2017

There TEMOLEbanon-Mintad project aims to offer internet access via a airship platform.

There are the following Milestones for 2017/2018:

<ul style="list-style-type: none"> • Sep 2017: Low-altitude flying airship (1-3 km) for internet supply is in operation (above Qalamoun/Ras Nhache) (prototype is developed with some master thesis') • 2018 High-altitude platform (20-22 km) for internet supply in cooperation with Turkey 	
---	--

Master Thesis: Construction, manufacturing and testing of a solar powered high altitude rigid airship

	<p>A rigid airship with a carrier frame formed of cross-ribs interconnected by long beams. A plurality of cross-ribs is interconnected by long beam sections interposed between neighboring cross-ribs. The so-formed carrier frame supports the lift producing carrier or lift gas cells and any other structural groups needed for the operation of the airship.</p>
<p>1-Automatic valve (gas) 18-Gas cell net-cord netting, between gas cell and wire netting 35-Maneuvering valve head 2-Axial cable-continuous through gas cells from bow to stern 19-Gas shaft 36-Mooring cone 3-Axial cone 20-Gas shaft hood 37-Mooring cone outrigger 4-Balloon surface 21-Gasoline tank 38-Observation platform 5-Balloon bag (water) 22-Handing lines 39-Outer cover 6-Balloon bag (water) emergency 23-Hand rail 40-Pneumatic bumper 7a-Bow nose 24-Hull 41-Quadrant 42-Rireta crum</p> <p>http://naca.central.cranfield.ac.uk/reports/1927/naca-report-240.pdf, p.73</p>	

Tasks:

- Construction of an airship based on patent, see <https://www.google.com/patents/US5110070?hl=de&cl=en> (
- Mounting a solar system
- Mounting actuator system (from other master thesis)
- Testing of airship at altitude of about 1 km

Contact:

Eng. Samir Mourad, Mob. +961 76 341526 (Lebanon), WhatsApp +49 178 7285578 (Mob. Germany)

Development of a GCS (Ground Control System) (2018)

From:

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



GCS PFD

of TEMOLeb-Mintad

Last update: Thursday, November 29, 2018



Designed on: Windows form application and programed using: C# coding language by: MMJZ

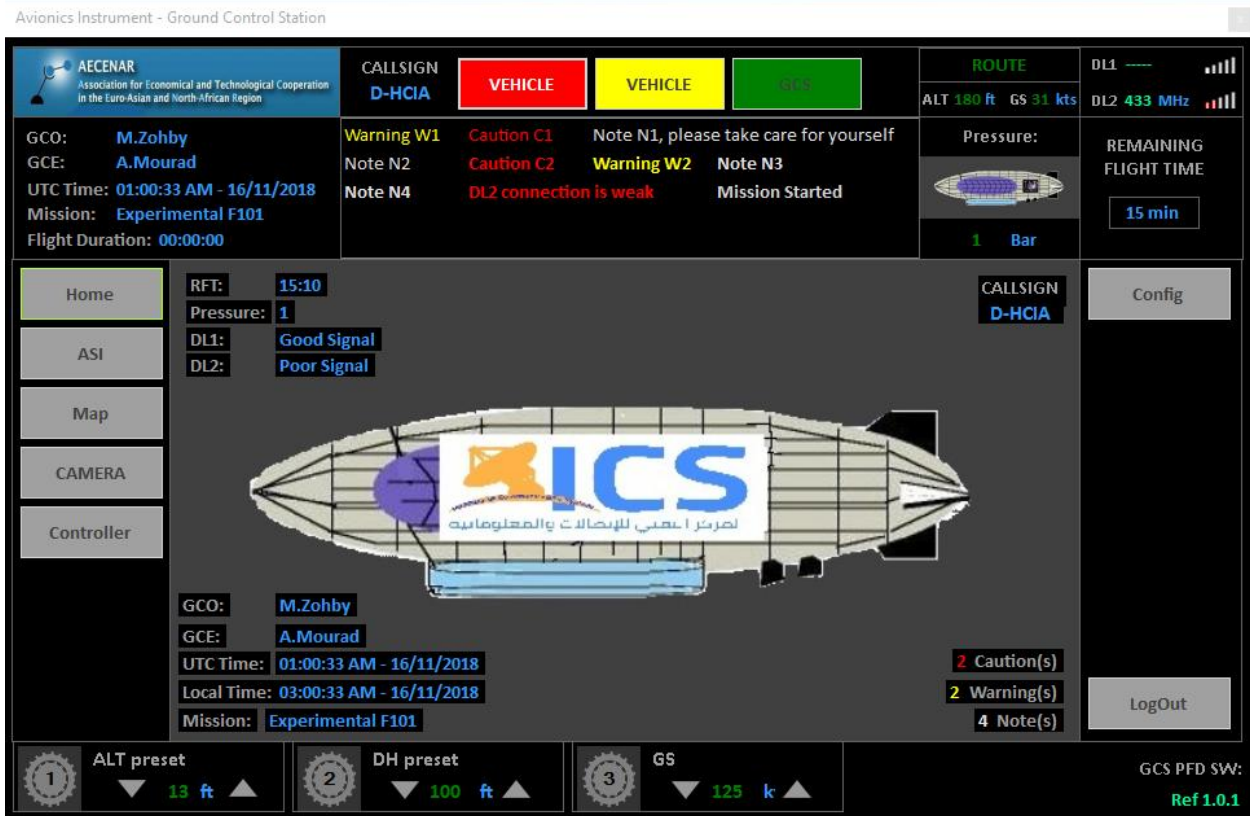
AECENAR @ 2018

Contents

CONTENTS	746
1 APP STRUCTURE:	747
1.1 HEADLINE WIDGET CONTAIN:	747
1.2 MAIN AREA:	748
1.3 LEFT NAVIGATION PANEL CONTAIN:	748
1.4 RIGHT NAVIGATION PANEL CONTAIN:	749
1.5 FOOTER WIDGET CONTAIN:	749
2 WIDGET:	750
2.1 HOME WIDGET:	750
2.2 ASI WIDGET:	750
2.3 MAP WIDGET:	751
2.4 CAMERA WIDGET:	751
2.5 CONTROLLER WIDGET:	752
3 CONFIGURATION PAGE:	753
3.1 GENERAL INFO CONFIGURATION:	753
3.2 CONNECTION CONFIGURATION:	754
3.3 KNOBS CONFIGURATION:	754
3.4 PFD CONFIGURATION:	755
3.5 MAP CONFIGURATION:	755
3.6 ROUTE CONFIGURATION:	756
3.7 TASK CONFIGURATION:	757
4 CONTROLLERS	758
4.1 SIGNAL STRENGTH	758
4.2 KNOBS CONTROLLER	758
5 USER CONTROL CODE SUMMARY:	759
5.1 CONFIGURATION:	759
5.2 CONTROLLER:	759
5.3 FOOTERWIDGET:	759
5.4 HEADLINEWIDGET:	760
5.5 HOMEWIDGET:	760
5.6 MAPWIDGET:	760
5.7 PFDWIDGET:	760
5.8 MAINMENU:	761
5.9 APPCONFIG:	762
5.10 LIBRARIES:	763
6 REFERENCES:	764

Headline Widget contain:

31 App Structure:

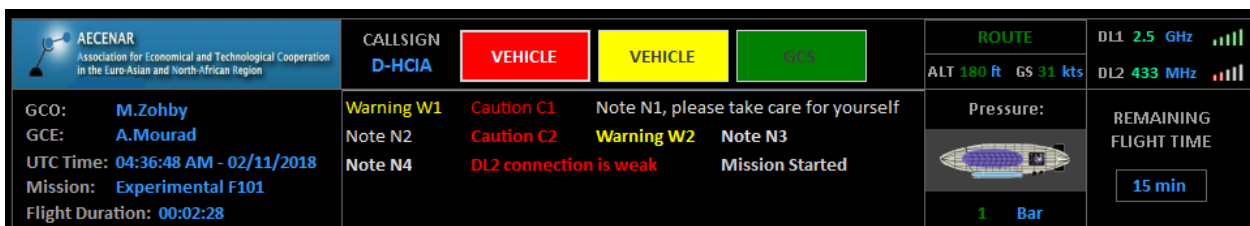


App structure consist of one main page, divided to 5 panels

- Headline Widget
- Main area
- Right navigation panel
- Left navigation panel
- Footer widget

Each panel contain the following:

31.1 Headline Widget contain:



- LOGO
- General info
- Caution-Warning indicator
- Caution\Warning\Notes main list
Caution shall be shown in Red, Warning shall be shown in Yellow, Notes shall be shown in White
- Route Altitude\Speed info
- Data Link 1&2 signal strength indicator
- Gas Pressure Info

App Structure:

- Remaining flight time Count down
Show the remaining Time of the lotto battery in Minutes than in Seconds, it shall be colored Red when it decrease to seconds

31.2 Main Area:

The area where the widget will be shown after selection and navigation from the Left and right navigation panels

31.3 Left navigation panel contain:



- Button to navigate to Home page
- Button to navigate to ASI widget
- Button to navigate to Map widget
- Button to navigate to Camera widget
- Button to navigate to Controller widget

31.4 Right navigation Panel Contain:



- Button to access the App configuration
- Button to Log-Out and close the App

31.5 Footer Widget Contain:



- 4 Quick access knobs to access important entrance
Knobs can be enabled or disabled from app configuration
- App sign and version

Widget:

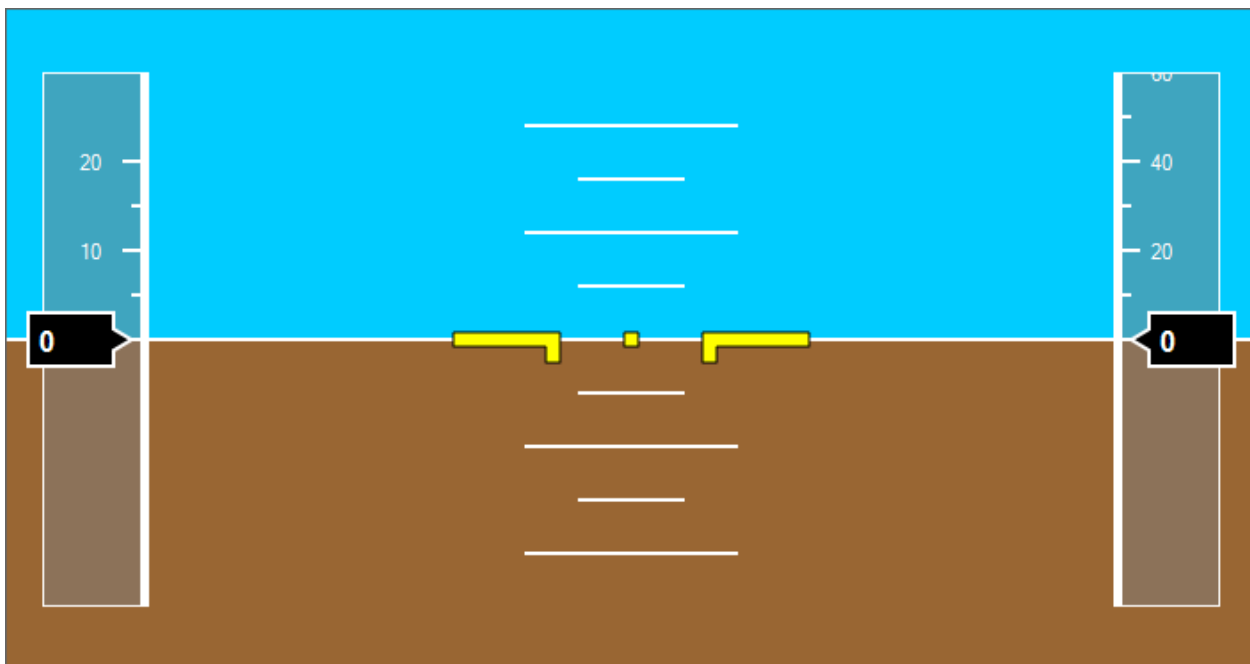
32 Widget:

32.1 Home Widget:



Show the general info and states of the vehicle, and the flight mission

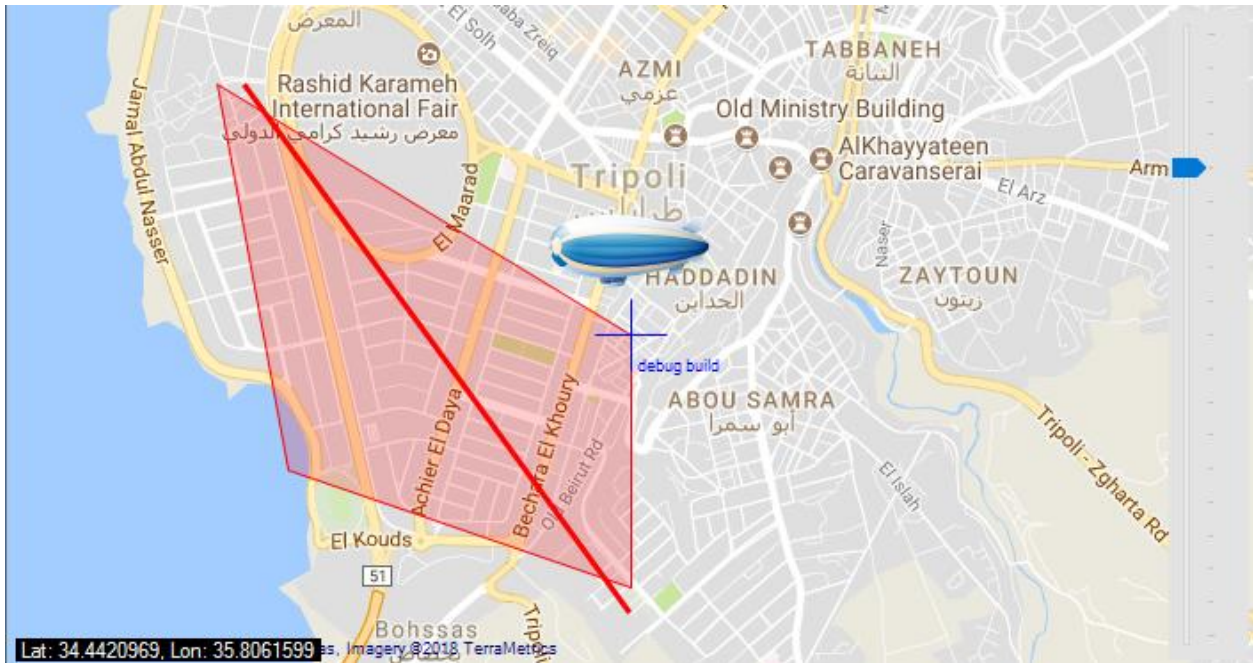
32.2 ASI Widget:



Show the Altitude and the speed of the lotto, with the orientation.

Map Widget:

32.3 Map Widget:



Map Widget is a GMap controller to show map with the lotto on it with the route and zone it should be pass, the route and zone set from the Route Configuration page

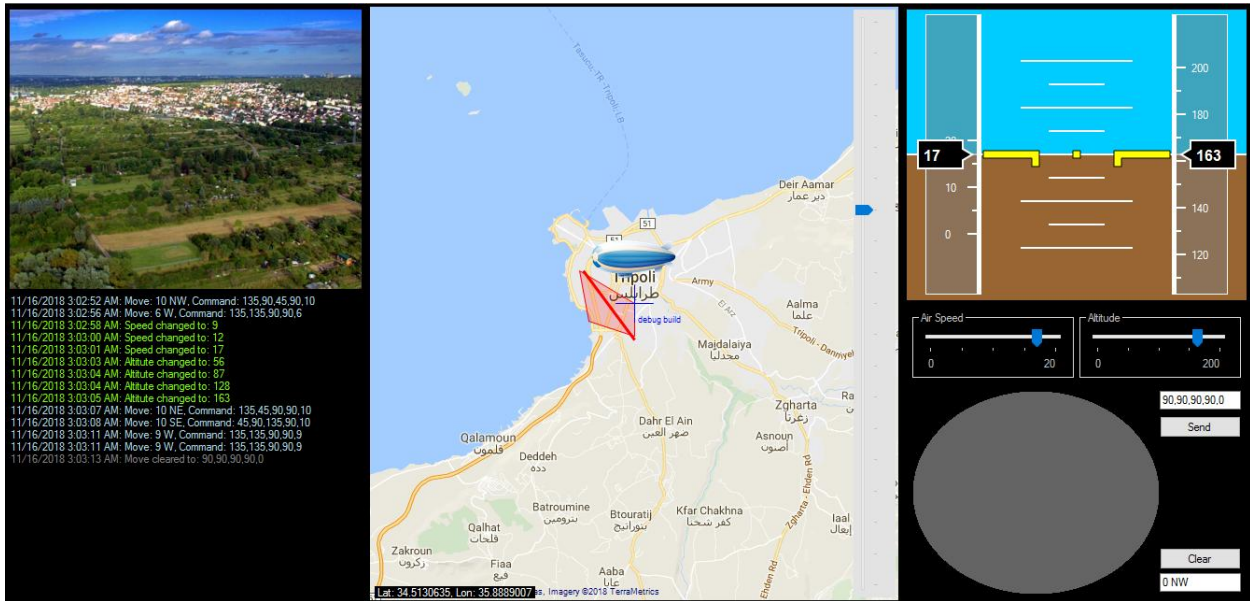
32.4 Camera Widget:



Show view of the lotto camera

Widget:

32.5 Controller Widget:



Contain the three monitoring widget with the controller of the lotto to fully access, log file of each flight mission will be saved as a txt file in "My Documents\AvionicsMissions"

33 Configuration Page:

Configuration page divided to tabs as follows:

- General
 - o General Info
 - o Connection
 - o Knobs
- Widgets
 - o PFD (ASI)
 - o Map
- Mission
 - o Route
 - o Task

33.1 General Info Configuration:

General	General Info
General Info	GCO name M.Zohby
Connection	GCE name A.Mourad
Knobs	Mission Experimental F101
Widgets	
PFD	
Map	
Mission	
Route	
Task	

Update

Press Update to save the update or it will be ignored

33.2 Connection Configuration:

General	Connection	
General Info		
Connection		
Knobs		
Widgets		
PFD		
Map		
Mission		
Route		
Task		

DL1		Serial
IP address	<input type="text"/>	Port: <input type="text"/>
Port	<input type="text"/>	Baud Rate: 115200
Subnet	<input type="text"/>	Parity: None
<input type="button" value="Connect"/>		Databits: 8
		Stopbits: One
		<input type="button" value="Connect"/>

DL2		
IP address	<input type="text"/>	
Port	<input type="text"/>	
Subnet	<input type="text"/>	
<input type="button" value="Connect"/>		

<input type="button" value="Update"/>

33.3 Knobs Configuration:

General	Knobs	
General Info		
Connection		
Knobs		
Widgets		
PFD		
Map		
Mission		
Route		
Task		

Knob 1	ALT preset
Knob 2	DH preset
Knob 3	GS
Knob 4	none

<input type="button" value="Update"/>

None disable the knobs

33.4 PFD Configuration:

General	PFD
General Info	Minimum Altitude 0
Connection	Maximum Altitude 200
Knobs	Minimum Speed 0
Widgets	Maximum Speed 20
PFD	ALT preset 13
Map	DH 100
Mission	
Route	
Task	

Update

This values will be configure the PFD view of the ASI widget

33.5 Map Configuration:

General	Map
General Info	Map Provider Google
Connection	Initial Zoom 10
Knobs	Initial Origin Lat 34.4289699 Lon 35.836247
Widgets	<input type="checkbox"/> Show center
PFD	
Map	
Mission	
Route	
Task	

Update

Map Widget view can be configure from here, map provider can be google or bing, knowing that the bing map is faster, while the google contain more info. Map is work online and offline if it already loaded before to the cache.

33.6 Route Configuration:

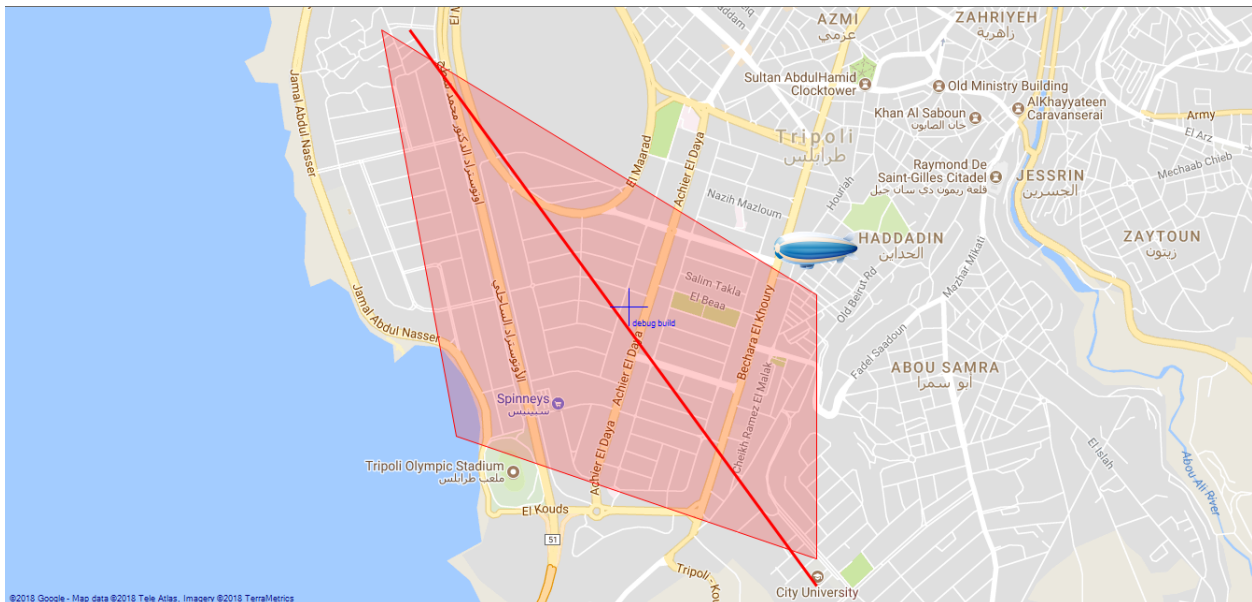


Zone: is the safe zone of the flight mission, a caution will be pop-up when the lotto get out from its safe zone, zone shall be shown as a red zone on the map.

Route: Is the route which the lotto should trace, a warning it will be pop-up in each time the lotto go away from the route, Route shall be shown as a red trace on the map

Each point in the zone or route list shall consist of a Lat. and Long. separate by coma ',' and each line while represent a point.

Map with Zone and route routed:

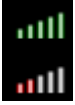


33.7 Task Configuration:



34 Controllers

34.1 Signal Strength



Use levels and colors to indicate signal strength, color and level will be change as follows:

- Till 10%: one Light Gray level
- Till 30%: 2 Red levels
- Till 40%: 3 Red levels
- Till 50%: 3 Yellow levels
- Till 70%: 4 Yellow levels
- Then, 5 Green levels

34.2 Knobs controller



Using this controller, user can quick access the field of the knobs without leaving widgets, each controller contain an up and a down arrow to increase and decrease value of the knobs.

Configuration:

35 User Control Code Summary:

35.1 Configuration:

LoadSettings	Load all settings from AppConfig to the user control fields
btn_Menu_Click	Callback for the Configuration menu tab choosing
btn_Connect_Serial_Click	Prepare the parameters for the connect to serial function
serialport_connect	Make a connection to a serial port
btn_Update_Click	Update the settings from the user control fields to the AppConfig
Sport	Public static element of the serial port

35.2 Controller:

Controller_Load	Load Controller widget settings from the AppConfig
btn_Send_Click	Send Command via serial port
trackAltitudeValue_Scroll	Callback to change the altitude value
trackAirSpeedValue_Scroll	Callback to change the speed value
UpdateAltitudeGaugeParams	Update the PFD view
MouseStick_MouseStickMoved	Callback on move to set the move speed and direction
MouseStick_MouseDoubleClick	Callback on double click to specify the move and set command
InsertLogLine	Function to write log line on log display and file
ParseMoveInfo	Parse move info to move command
trackAirSpeedValue_MouseUp	Callback on Mouse up to write log for the new speed value
trackAltitudeValue_MouseUp	Callback on Mouse up to write log for the new altitude
btn_Clear_Click	Callback to clear move command

35.3 FooterWidget:

FooterWidget_Load	Call UpdateKnobs function to load Footer knobs
UpdateKnobs	Load Knobs into footer
pb_Down_Click	Move one step down from the knobs value
pb_Up_Click	Move one step up to the knobs value

35.4 HeadlineWidget:

HeadlineWidget_Load	Load Headline info and settings, and start the clock and stopwatch timer
LoadGeneralInfo	Load General info from the AppConfig to the Headline widget
AddWarningCautionNote	Public function used to add warning\Caution\Note to the list
AnimateItem	Used to animate the new item add to the list
The headline widget use the SignalStrength control from the Libraries folder to indicate the Datalink signal strength	

35.5 HomeWidget:

LoadGeneralInfo	Load general info to the home screen
-----------------	--------------------------------------

35.6 MapWidget:

LoadMap	Load widget map under the settings set in the AppConfig
trackBar1_Scroll	Callback to change map zoom on track bar scroll
Addmarker	Add the Airship icon to the map
gmap_OnMarkerClick	Event called when the airship marker click
AddPolygon	Add the Zone polygon set in the AppConfig
gmap_OnPolygonClick	Event called when the zone polygon click
AddRoute	Add the route of the flight set in the AppConfig mission to the map
gmap_MouseMove	Callback on mouse move to display the coordinate on the mouse point
GetMouseCoordinate	Public function return the mouse coordinate

35.7 PFDWidget:

InitializeGauges	Initialize the gauges of the PFD widget, and set the callback of the sport receiver
sport_DataReceived	Callback of the sport receiver
SetValue	Update PFD values under the data received from the sport receiver

MainMenu:

SetText	Update the value of the PFD info label
Redraw	Redraw the user control
GraphicUserControl_Resize	Callback to redraw the PFD widget when the user control resized
GraphicUserControl_Paint	Callback to paint the Widget
The PFD Widget use the classes and interfaces and controllers in the Libraries\PFDLib folder	

35.8 MainMenu:

MainWindow_Load	Callback on page load to load the home widget
ResetButtons	Function to reset buttons selection and update headline content
btn_Home_Click	Callback on Home button click to load Home Widget
btn_ASI_Click	Callback on ASI button click to load PFD Widget
btn_Map_Click	Callback on Map button click to load Map Widget
btn_Camera_Click	Callback on Camera button click to load Camera Widget
btn_Controller_Click	Callback on Controller button click to load Controller Widget
btn_Config_Click	Callback on Config button click to load Config Widget
btn_LogOut_Click	Callback on logout button click to logout and close the app

35.9 AppConfig:

Setting	Type	Description	Default Value
GCO	String		
GCE	String		
Mission	String		
MinAlt	Int	Minimum value of the altitude roller	0
MaxAlt	Int	Maximum value of the altitude roller	200
MinSpeed	Int	Minimum value of the speed roller	0
MaxSpeed	Int	Maximum value of the speed roller	20
Knob1	String	Parameter of the footer quick access knob 1 (choose from: ALT Preset, DH Preset, GS, none)	ALT Preset
Knob2	String	Parameter of the footer quick access knob 2 (choose from: ALT Preset, DH Preset, GS, none)	DH preset
Knob3	String	Parameter of the footer quick access knob 3 (choose from: ALT Preset, DH Preset, GS, none)	GS
Knob4	String	Parameter of the footer quick access knob 4 (choose from: ALT Preset, DH Preset, GS, none)	None
AltPreset	Int	The altitude preset value	13
DHPreset	Int	The DH preset value	100
GS	Int	The ground speed value	125
MapProvider	String	The Map provider	Google
InitialMapZoom	Int	The initial map zoom value (choose value from 0 to 16)	10
OrigLat	String	The map origin initial latitude	34.4289699
OrigLon	String	The map origin Initial Longitude	35.836247
Zone	String	The safe zone of the flight mission	
Route	String	The route of the flight mission	
Task	String	The task of the mission	
ShowMapCenter	Bool	Add map center cross to the map or not	False
LogFilePath	String	Path of the log file on the local computer	
MissionStart	Bool	Mission is start or not	False

Libraries:

35.10 Libraries:

GMap library (DLL)	Used in Map widget
PrimayFlighDisplay (DLL)	Used with its PFDLib file contents as it is in the PFD widget
MouseStick1 (DLL)	Used on the mouse stick in the controller widget
SignalStrength (user Control)	Used on the Headline widget to show signal strength

36 References:

Setup zip file:



setup.zip

Development of a FCS (Flight Control System) (2018)



FCS

Of TEMOLeb-Mintad



Last update: 16.4.2019

Author: MMJZ

AECENAR @ 2018/19

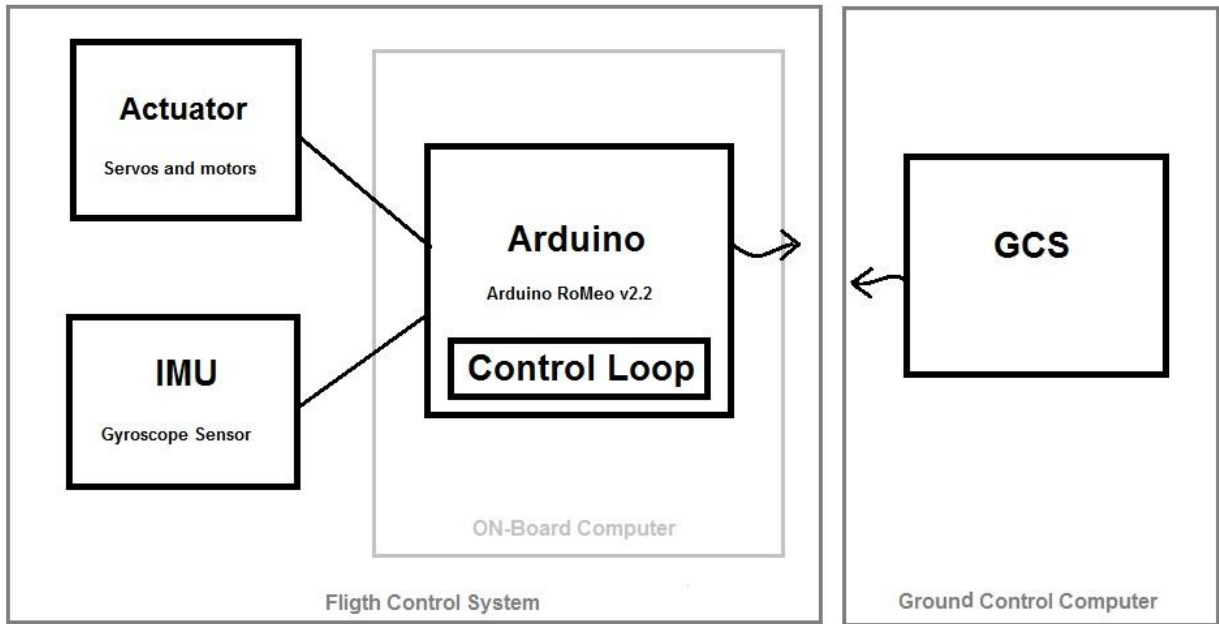
Contents

1	SYSTEM ARCHITECTURE	768
2	SERVO ACTUATOR SYSTEM	769
2.1	ADOPTED MOTOR	769
2.2	BASIC PARTS OF THE SERVO	769
2.3	SERVO MOTOR BLOCK DIAGRAM	770
2.3.1	<i>Control computer:</i>	770
2.3.2	<i>Electronic control system:</i>	770
2.3.3	<i>Motor:</i>	770
2.3.4	<i>Gear train:</i>	771
2.3.5	<i>Position sensor:</i>	771
2.3.6	<i>Servo output:</i>	771
2.4	DESIGN OF SERVO ACTUATOR SYSTEM (FREECAD)	771
2.5	MOTOR CONTROLLER AND INTERFACES	771
2.5.1	<i>Mobile App:</i>	772
2.5.2	<i>Bluetooth Module: (BLE Link -A Bluetooth 4.0 module for Arduino)</i>	772
2.5.3	<i>Arduino controller: Romeo V2-All in one Controller-motor drive built in</i>	773
2.5.4	<i>Motor drive: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N</i>	775
2.6	POWER MANAGEMENT: POLYMER LITHIUM ION BATTERY - 1000MAH 7.4V	776
3	REALIZATION OF INERTIAL MEASUREMENT UNIT (IMU)	778
3.1	IMU-SENSOR: LSM9DS0	779
3.2	LSM9DS0 HOOKUP GUIDE	780
3.2.1	<i>Covered In This Tutorial</i>	780
3.2.2	<i>About the LSM9DS0</i>	780
3.2.3	<i>Choose Your Own Adventure: SPI or I²C</i>	781
3.2.4	<i>Breakout Overview</i>	782
3.2.5	<i>Basic Arduino Example</i>	785
3.2.6	<i>Resources and Going Further</i>	787
4	WIFI COMMUNICATION WITH NRF24L01	788
4.1	ARDUINO LIBRARY	789
5	SOFTWARE DEVELOPMENT ON ARDUINO SIDE WITH THE ARDUINO INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)	792
5.1	ARCHITECTURE OF THE PROGRAM ON THE ARDUINO	792
5.2	BLUETOOTH_BLIMP_CONTROL CODE	792
5.3	IMU SENSOR LSM9DS0 CODE	795
5.4	CONTROL LOOP	797
5.5	WIFI COMMUNICATION CODE	797
5.6	FULL CODE	800
6	ASSEMBLY	808
6.1	ACTUATOR ASSEMBLY	808

Development of a FCS (Flight Control System) (2018)

6.2	IMU ASSEMBLY	814
6.3	WIFI MODULE ASSEMBLY	814
6.4	FULL ASSEMBLY	815
7	FINAL IMPLEMENTATION	816

37 System Architecture



38 Servo actuator system

38.1 Adopted Motor



Figure 69: the servo motor which we buy.

38.2 Basic parts of the servo

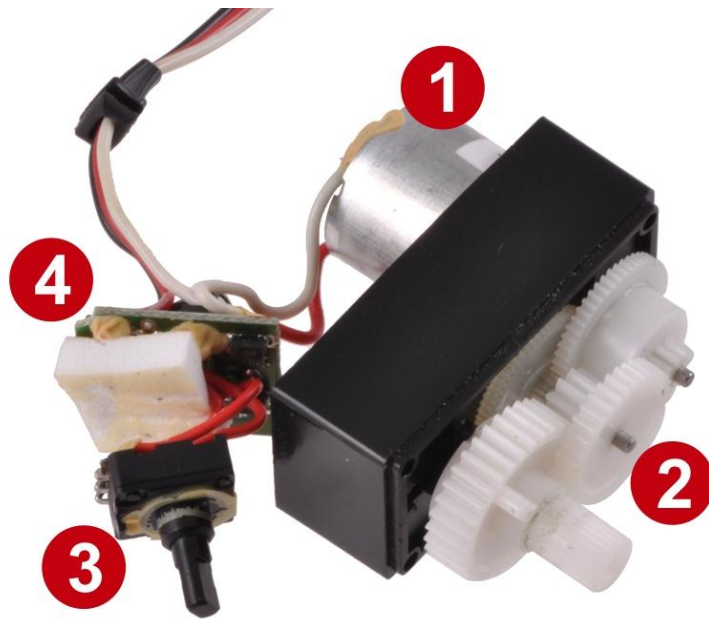


Figure 70: picture showing all interior parts for any servo (motor, gearbox, potentiometer, and control electronics).

- | | |
|--------------------|------------------------------|
| 1. Motor | 2. Gearbox |
| 3. Position sensor | 4. Motor control electronics |

38.3 Servo motor block diagram

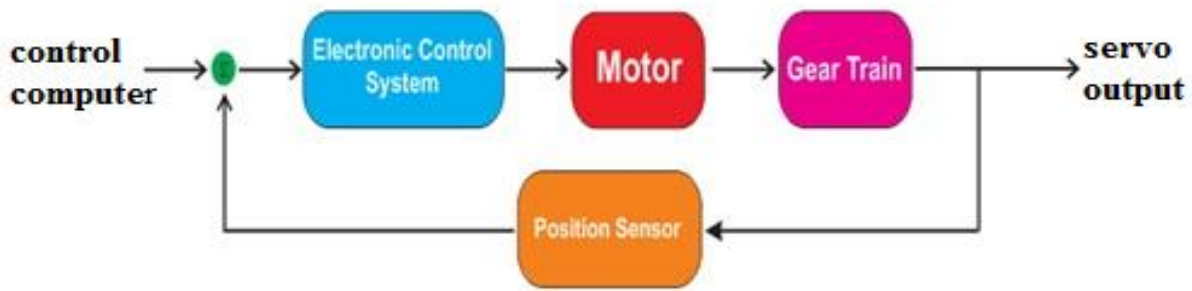


Figure 71: servo motor block diagram.

38.3.1 Control computer:

The mobile application used “BlunoBasicDemo” for testing to send an electronic control signal containing the angle of each servo and the on or off for propeller.

Then it should be controlled by the GCS and the Lotte_Regler.

38.3.2 Electronic control system:

It is an interconnection between 2 signals: input and output. A signal is transformed to another one using a process; in the objective to create motion, change a speed, etc. specifically, in our project we use the type called closed loop electronic control system. By correcting the error occurred; it helped us to main the system more stable and enhance its control. In the following a scheme (Figure 61: the concept of electronic control system. is showed to explain this concept.

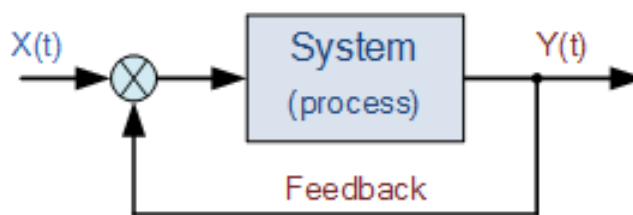


Figure 72: the concept of electronic control system.

38.3.3 Motor:

Inside the servo is a DC motor. This one will turn when receiving an electric signal from the control system.

38.3.4 Gear train:

An assembly of many gears; receive the rotation force from DC motor; and transform it into torque.

38.3.5 Position sensor:

Such as a potentiometer; which has to learn continuously the mechanical position of the shaft by changing the resistance of an interior resistor.

38.3.6 Servo output:

A PWM signal was sent to the motor, turn the shaft in the desired position. This can describe the output of a servo.

38.4 Design of servo actuator system (FreeCAD)


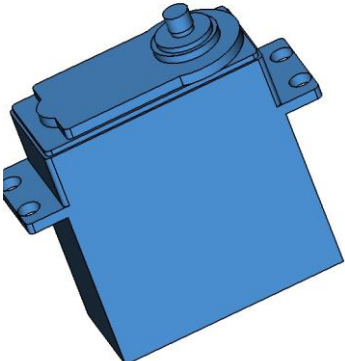

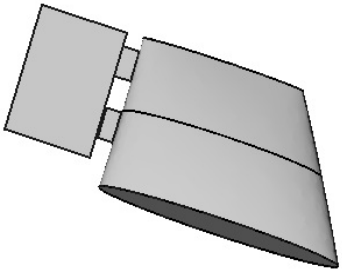
Parts and details	Date and FreeCAD file	Screenshot
Servo Motor	 servomotor.FCStd	
Rudder	 rudder.FCStd	

Table 14: freeCAD model of servo motor and the airship rudder.

38.5 Motor Controller and Interfaces

The work which is described in this section was done by CNCLab.



The mobile application named “**BlunoBasicDemo**” sends a command to the Bluetooth

module taking place on the Arduino board. It passes through the motor drive then arrive to the servo actuator in the form of PWM signal which is responsible for rotating the servo in different angles.

38.5.1 Mobile App:



Figure 73: BlunoBasicDemo logo.

BlunoBasicDemo (Figure 62: BlunoBasicDemo logo. is a basic Demo for Bluno including all the code and executable on Android, IOS and Android. You can easily develop your own with this Demo.

38.5.2 Bluetooth Module: (BLE Link -A Bluetooth 4.0 module for Arduino)

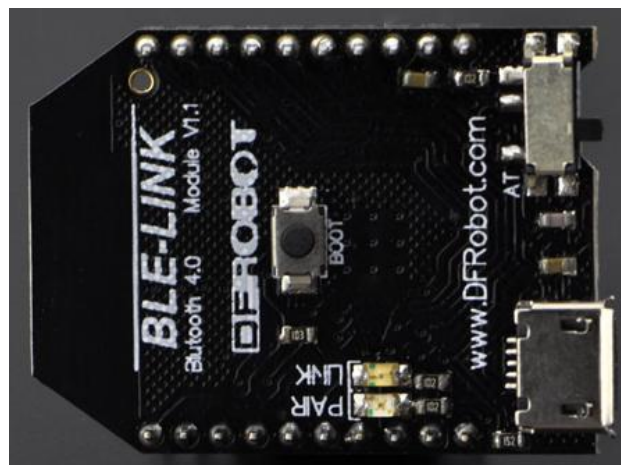


Figure 74: BLE Link -A Bluetooth 4.0 module for Arduino.

Description:

It is a peripheral that quickly connects the device to the mobile via BLE, using the XBEE model suitable for all XBEE screens. Many of the applications for Android and IOS systems are developed to be at the service of users who must use it to communicate between BLE and Arduino.

Specification:

Price: 23.95\$

Mark: DFRobot

Serial number: WRL0024

Chip: TI CC2540>

Working voltage: +3.3DC

Power consumption: working 10.6 mA average, ready mode: 8.7 mA

Pin Layout: Compatible With XBEE pinout

Frequency: 2.4 GHz

Transfer rate: 1 Mbps

Modulation: GFSK, Bluetooth low power, V4.0

Sensitivity: -93dB

Operating temperature: - 40 → +85

Transmission distance: 60 m in free space

Size: 32mm * 22mm (1.26 * 0.87")

38.5.3 Arduino controller: Romeo V2-All in one Controller-motor drive built in

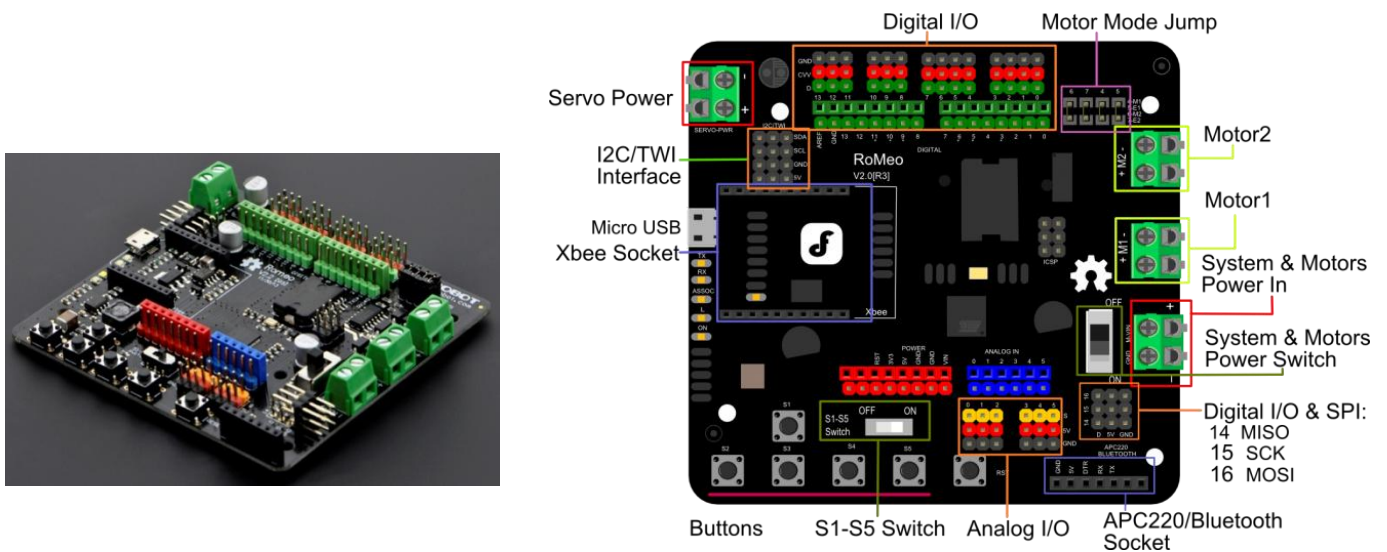


Figure 75: Romeo V2-All in one Controller-motor drive built in.

Description:

Romeo V2 [R3] (Figure 64: Romeo V2-All in one Controller-motor drive built in.) is an All-in-One Arduino, microcontroller, made specifically for robotics applications. This type of Arduino based on the ATmega32u4 chip, can be programed fast via the Arduino IDE. Thanks to ATmega32u4 chip, the ease and the simplicity of RoMeo V2. Another application of Romeo V2 is that it can control a stepper motor.

Specification:

Price: 46.95 \$

Mark: DFRobot

Serial number: DEV0005

DC Supply: USB Powered or External 6V ~ 23V DC

DC Output: 5V (2A) / 3.3V DC

Motor driver Continuous Output Current: 2A

Microcontroller: ATmega32u4

Bootloader: Arduino Leonardo

Compatible with the Arduino R3 pin mapping

Analog Inputs: A0-A5, A6 - A11 (on digital pins 4, 6, 8, 9, 10, and 12)

PWM: 3, 5, 6, 9, 10, 11, and 13. Provide 8-bit PWM output

5 key inputs for testing

Auto sensing/switching external power input

Serial Interface

TTL Level

USB

Support Male and Female Pin Header

Built-in XBEE socket

Integrated sockets for APC220 RF Module and DF-Bluetooth Module

Three I2C/TWI Interface Pin Sets (two 90° pin headers)

Two way Motor Driver with 2A maximum current

One Stepper Motor Drive with 2A maximum current

Size: 89 x 84 x 14mm

38.5.4 Motor drive: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N

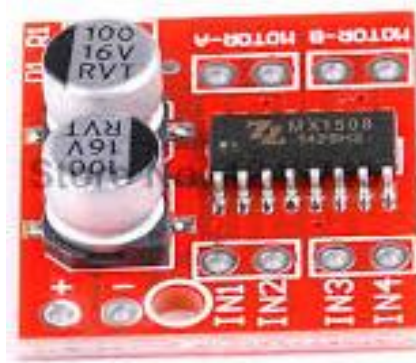


Figure 76: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N.

Description:

HR 2 motordrive (*Figure 65*: HR 2 channel dc motordrive dual h bridge stepmotor reversing PWM speed control mini L298N. can be used in a device need a voltage between 2 and 10 v. with his 2 pin each to 1.5 A DC current ; it can manage the position and speed control.

Note that in this project motor drive is used only for DC motor which's responsible of propeller motion and speed (i.e. servo motor don't need motor drive, it will be directly connected to arduino).

Specification:

Price: 3.95\$

Serial number: DRV0023

H bridge motor dual drive, and can drive two DC motor or 1 line 4 phase stepper motor;

The voltage of the power supply module 2V-10V;

The signal input voltage 1.8-7V;

Single channel current of 1.5A, peak current up to 2.5A, low standby current (less than 0.1uA);

The built-in common conduction circuit, the input end is suspended, the motor will not malfunction;

The built-in overheat protection circuit with hysteresis effect (TSD), there is no need to worry about motor stall;

Product size: 24.7*21*5mm (LxWxH), ultra-small size, suitable for assembly and vehicle;

Mounting hole diameter: 2 mm.

Weight: 5g

38.6 Power Management: Polymer Lithium Ion Battery - 1000mah 7.4v



Figure 77: Polymer Lithium Ion Battery - 1000mah 7.4v.

Description:

This LiPo (Figure 66: Polymer Lithium Ion Battery - 1000mah 7.4v. is an excellent battery that can be used in any application; who need a little power supply has several punch as in robotics. Its low voltage and sufficient flow allows it to acquire many electronic and some small motors

The battery has two cells and produces 7.4 V to store 1000 mAh of charge. This type of batteries requires a specific charger.

Specification:

Price: 13.95\$

Mark: SparkFun Electronics®

Serial number: PWR0024

7.4V 2-cell pack

1000mAh of charge

Discharge rate: 25C continuous

Charge plug: JST-XH

Discharge plug: JST-RCY

Dimensions: 70mm x 35mm x 18mm

Weight: 85g (2.99oz)

39 Realization of Inertial Measurement Unit (IMU)

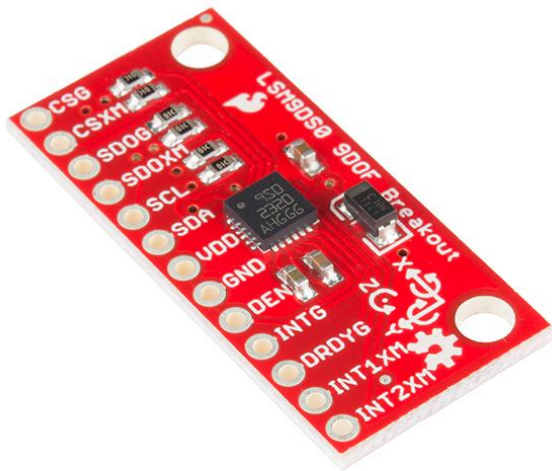
The IMU has the following parts:

- Gyro
- Accelerometer
- Magnetometer

The software runs on a raspberry pi.



39.1 IMU-Sensor: LSM9DS0



LSM9DS0 IMU Breakout - 9DoF

Price : 39.95\$

Mark : SparkFun Electronics®

Serial number : SEN0080

Description:

This is the LSM9DS0, a versatile motion-sensing system-in-a-chip that houses a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer. That's right, 9 degrees of freedom (9dof) from a single IC!

Each sensor in the LSM9DS0 supports a wide range of, well, ranges: the accelerometer's scale can be set to ± 2 , 4, 6, 8, or 16 g, the gyroscope supports ± 245 , 500, and 2000 $^{\circ}/s$, and the magnetometer has full-scale ranges of ± 2 , 4, 8, or 12 gauss. Additionally, the LSM9DS0 includes an I2C serial bus interface supporting standard and fast mode (100 kHz and 400 kHz) and an SPI serial standard interface.

FEATURES

3 acceleration channels, 3 angular rate channels, 3 magnetic field channels

$\pm 2/\pm 4/\pm 6/\pm 8/\pm 16$ g linear acceleration full scale

$\pm 2/\pm 4/\pm 8/\pm 12$ gauss magnetic full scale

$\pm 245/\pm 500/\pm 2000$ dps angular rate full scale

16-bit data output

SPI / I2C serial interfaces

Analog supply voltage 2.4 V to 3.6 V

Programmable interrupt generators

Embedded self-test

Embedded temperature sensor

Embedded FIFO

39.2 LSM9DS0 Hookup Guide²⁴

39.2.1 Covered In This Tutorial

This tutorial is devoted to all things LSM9DS0. We'll introduce you to the chip itself, then the breakout board. Then we'll switch over to example code, and show you how to interface with the board using an Arduino and our [SFE LSM9DS0 Arduino library](#).

- The tutorial is split into the following pages:
- [About the LSM9DS0](#) – An overview of the LSM9DS0, examining its features and capabilities.
- [Breakout Overview](#) – This page covers the LSM9DS0 Breakout Board – topics like the pinout, jumpers, and schematic.
- [Hardware Assembly](#) – Assembly tips and tricks, plus some information about the breakout's dimensions.
- [Basic Arduino Example](#) – How to install the **Arduino library**, and use a simple example sketch.
- [Advanced Arduino Example](#) – A more advanced Arduino sketch – using the library – showing off features like switch the sensors' scales and data rates.
- [Using the Arduino Library](#) – An overview of the SFE_LSM9DS0 Arduino library.

39.2.2 About the LSM9DS0

The LSM9DS0 is one of only a handful of IC's that can measure three key properties of movement – angular velocity, acceleration, and heading – in a single IC.

The [gyroscope](#) can measure **angular velocity** – that is “how fast, and along which axis, am I rotating?” Angular velocities are measured in **degrees per second** – usually abbreviated to DPS or °/s. The LSM9DS0 can measure up to ± 2000 DPS, though that scale can also be set to either 245 or 500 DPS to get a finer resolution.

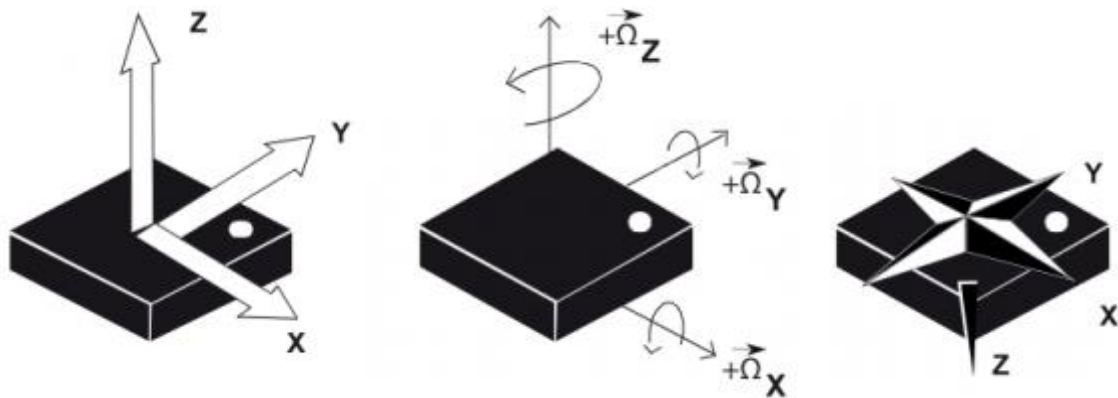
An [accelerometer](#) measures **acceleration**, which indicates how fast velocity is changing – “how fast am I speeding up or slowing down?” Acceleration is usually either measured in m/s² (meters per second per second) or g's (gravities [about 9.8 m/s²]). If an object is sitting motionless it feels about 1 g of acceleration towards the ground (assuming that ground is on earth, and the object is near

²⁴ https://learn.sparkfun.com/tutorials/lsm9ds0-hookup-guide?_ga=2.72099031.592506407.1511257598-1712679554.1505480322

sea-level). The LSM9DS0 measures its acceleration in g 's, and its scale can be set to either ± 2 , 4, 6, 8, or 16 g .

Finally, there's the [magnetometer](#), which measures the power and direction of **magnetic fields**. Though they're not easily visible, magnetic fields exist all around us – whether you're holding a tiny ferromagnet or feeling an attraction to [Earth's magnetic field](#). The LSM9DS0 measures magnetic fields in units of **gauss (Gs)**, and can set its measurement scale to either ± 2 , 4, 8, or 12 Gs.

By measuring these three properties, you can gain a great deal of knowledge about an object's movement. 9DOF's have tons and tons of applications. Measuring the force and direction of Earth's magnetic field with a magnetometer, you can approximate your **heading**. An accelerometer in your phone can measure the direction of the force of gravity, and estimate **orientation** (portrait, landscape, flat, etc.). Quadcopters with built-in gyroscopes can look out for sudden rolls or pitches, and correct their momentum before things get out of hand.



The LSM9DS0 measures each of these movement properties in three dimensions. That means it produces **nine pieces of data**: acceleration in $x/y/z$, angular rotation in $x/y/z$, and magnetic force in $x/y/z$. On the breakout board, the z -axis runs normal to the PCB, the y -axis runs parallel to the short edge, and the x -axis is parallel to the long edge. Each axis has a positive and negative direction as well, noted by the direction of the arrow on the label.

The LSM9DS0 is, in a sense, two IC's smashed into one package – like if you combined an [L3G4200D gyro](#) with an [LSM303DLMTR accel/mag](#). One half of the device takes care of all-things gyroscope, and the other half manages both the accelerometer and magnetometer. In fact, a few of the control pins are dedicated to a single sensor – there are **two chip select pins** (CSG for the gyro and CSXM for the accel/mag) and **two serial data out pins** (SDOG and SDOXM).

39.2.3 Choose Your Own Adventure: SPI or I²C

In addition to being able to measure a wide variety of movement vectors, the LSM9DS0 is also multi-featured on the hardware end. It supports both [SPI](#) and [I²C](#), so you should have no difficulty finding a microcontroller that can talk to it.

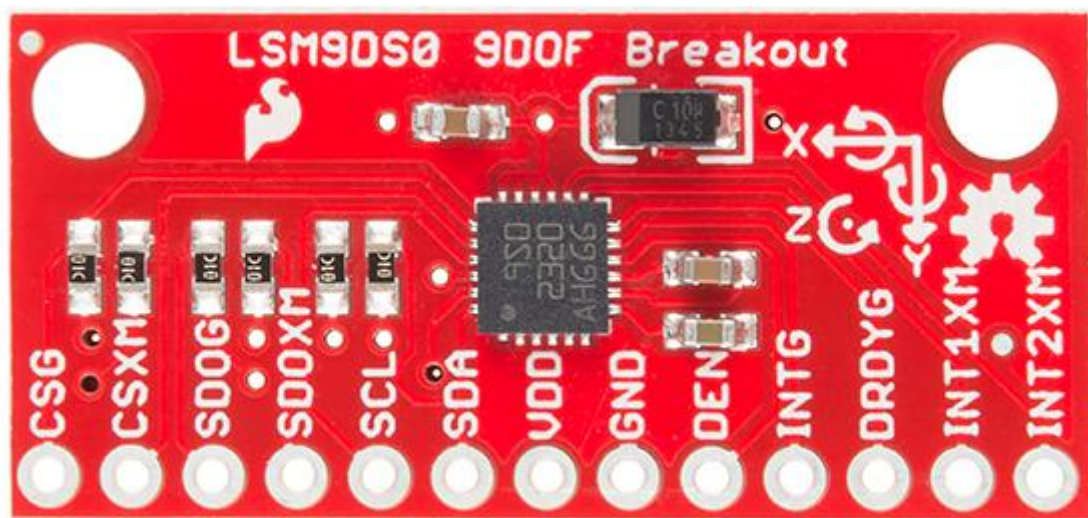
For much more detailed information about the IC, we encourage you to [check out the datasheet](#)²⁵

39.2.4 Breakout Overview

Now that you know everything you need to about the LSM9DS0 IC, let's talk a bit about the breakout board it's resting on. On this page we'll discuss the pins that are broken out, and some of the other features on the board.

39.2.4.1 The Pinout

In total, the LSM9DS0 Breakout breaks out 13 pins.



Here's an overview of each of the pin functions:

Pin Label	Pin Function	Notes
CSG	Chip Select Gyro	This pin selects between I ² C and SPI on the gyro. Keep it HIGH for I ² C, or use it as an (active-low) chip select for SPI. HIGH (1): SPI idle mode / I²C enabled LOW (0): SPI enabled / I²C disabled.
CSXM	Chip Select Accel/Mag (XM)	This pin selects between I ² C and SPI on the XM. Keep it HIGH for I ² C, or use it as an (active-low) chip select for SPI. HIGH (1): SPI idle mode / I²C enabled LOW (0): SPI enabled / I²C disabled.
SDOG	SPI: Gyroscope MISO	In SPI mode, this is the gyroscope data output (SDO_G).

²⁵ <https://cdn.sparkfun.com/assets/f/6/1/f/0/LSM9DS0.pdf>

Development of a FCS (Flight Control System) (2018)

	I ² C: Gyro address select	In I ² C mode, this selects the LSb of the I ² C address (SA0_G)
SDOXM	SPI: Accel/Mag MISO I ² C: XM address select	In SPI mode, this is the XM data output (SDO_XM). In I ² C mode, this selects the LSb of the I ² C address (SA0_XM)
SCL	Serial Clock	I ² C and SPI serial clock.
SDA	SPI: MOSI I ² C: Serial Data	SPI: Device data in (MOSI) I ² C: Serial data (bi-directional)
VDD	Power Supply	Supply voltage to the chip. Should be regulated between 2.4V and 3.6V .
GND	Ground	0V voltage supply
DEN	Gyroscope Data Enable	Mostly unknown. The LSM9DS0 datasheet doesn't have much to say about this pin.
INTG	Gyro Programmable Interrupt	An interrupt that can be programmed as active high/low, push-pull, or open drain. It can trigger on over/under rotation speeds.
DRDYG	Gyroscope data ready	An interrupt that can indicate new gyro data is ready or buffer overrun.
INT1XM	Accel/Mag Interrupt 1	A programmable interrupt that can trigger on data ready, over-acceleration or "taps".
INT2XM	Accel/Mag Interrupt 2	A programmable interrupt that can trigger on data ready, over-acceleration or "taps".

These pins can all be classified into one of three categories: communication, interrupts, or power.

Power Supply

The VDD and GND pins are where you'll supply a voltage and 0V reference to the IC. The breakout board does not regulate this voltage, so make sure it falls within the allowed supply voltage range of the LSM9DS0: **2.4V to 3.6V**. Below is the electrical characteristics table from the datasheet.

Symbol	Parameter	Test conditions	Min.	Typ. ⁽¹⁾	Max.	Unit
Vdd	Supply voltage		2.4		3.6	V
Vdd_IO	Module power supply for I/O		1.71	1.8	Vdd+0.1	
Idd_XM	Current consumption of the accelerometer and magnetic sensor in normal mode ⁽²⁾			350		μA
Idd_G	Gyroscope current consumption in normal mode ⁽³⁾			6.1		mA
Idd_G_LP	Gyroscope supply current in sleep mode ⁽⁴⁾			2		mA
Idd_Pdn	Current consumption in power-down mode ⁽⁵⁾			6		μA
VIH	Digital high-level input voltage		0.8*Vdd_IO			V
VIL	Digital low-level input voltage				0.2*Vdd_IO	V
VOH	High-level output voltage		0.9*Vdd_IO			V
VOL	Low-level output voltage				0.1*Vdd_IO	V
Top	Operating temperature range		-40		+85	°C

The communication pins are not 5V tolerant, so they'll need to be regulated to within a few mV of VDD.

Another very cool thing about this sensor is how **low-power** it is. In normal operation – with every sensor turned on – it'll pull around **6.5mA**.

Communication

CSG, CSXM, SDOG, SDOXM, SCL, and SDA are all used for the I²C and SPI interfaces. The function of these pins depends upon which of the two interfaces you're using.

If you're using using I²C here's how you might configure these pins:

- Pull CSG and CSXM HIGH. This will set both the gyro and accel/mag to I²C mode.
- Set SDOG and SDOXM either HIGH or LOW. These pins set the I²C address of the gyro and accel/mag sensors.
- Connect SCL to your microcontroller's SCL pin.
- Connect SDA to your microcontroller's SDA pin.
- The board has a built-in 10kΩ pull-up resistor on both SDA and SCL lines. If that value is too high, you can add a second 10kΩ resistor in parallel to divide the pull-up resistance to about 5kΩ.

Or, if you're using SPI:

- Connect CSG and CSXM to two individually controllable pins on your microcontroller. These chip-selects are active-low – when the pin goes LOW, SPI communication with either the gyro (CSG) or accel/mag (CSXM) is enabled.
- SDOG and SDOXM are the serial data out pins. In many cases you'll want to connect them together, and wire them to your microcontroller's **MISO** (master-in, slave-out) pin.

- Connect SCL to your microcontroller's SCLK (serial clock) pin.
- Connect SDA to your microcontroller's **MOSI** (master-out, slave-in) pin.

Interrupts

There are a variety of interrupts on the LSM9DS0. While connecting up to these is not as critical as the communication or power supply pins, using them will help you get the most out of the chip.

The accelerometer- and magnetometer-specific interrupts are **INT1XM** and **INT2XM**. These can both be programmed to interrupt as either active-high or active-low, triggering on events like data ready, tap-detection, or when an acceleration or magnetic field passes a set threshold.

DRDY and **INTG** are devoted gyroscope interrupts. DRDY can be programmed to go high or low when new gyroscope readings are ready to read. INTG is a little more customizable, it can be used to trigger whenever angular rotation exceeds a threshold on any axis.

1) Basic Arduino Example

This example will show you how to download and install the SFE_LSM9DS0 library, and use it in it's most basic form. We'll use I²C and ignore the interrupts, which means we'll be using as few wires and Arduino pins as possible.

2) Download and Install the Library

We've written a full-featured Arduino library to help make interfacing with the LSM9DS0's gyro and accelerometer/magnetometer as easy-as-possible. Visit the [GitHub repository](#) to download the most recent version of the library, or click the link below:

[DOWNLOAD THE SFE LSM9DS0 ARDUINO LIBRARY](#)

For help installing the library, check out our [How To Install An Arduino Library tutorial](#). You'll need to move the *SFE_LSM9DS0* folder into a *libraries* folder within your Arduino sketchbook.

3) Simple Hardware Hookup (I²C)

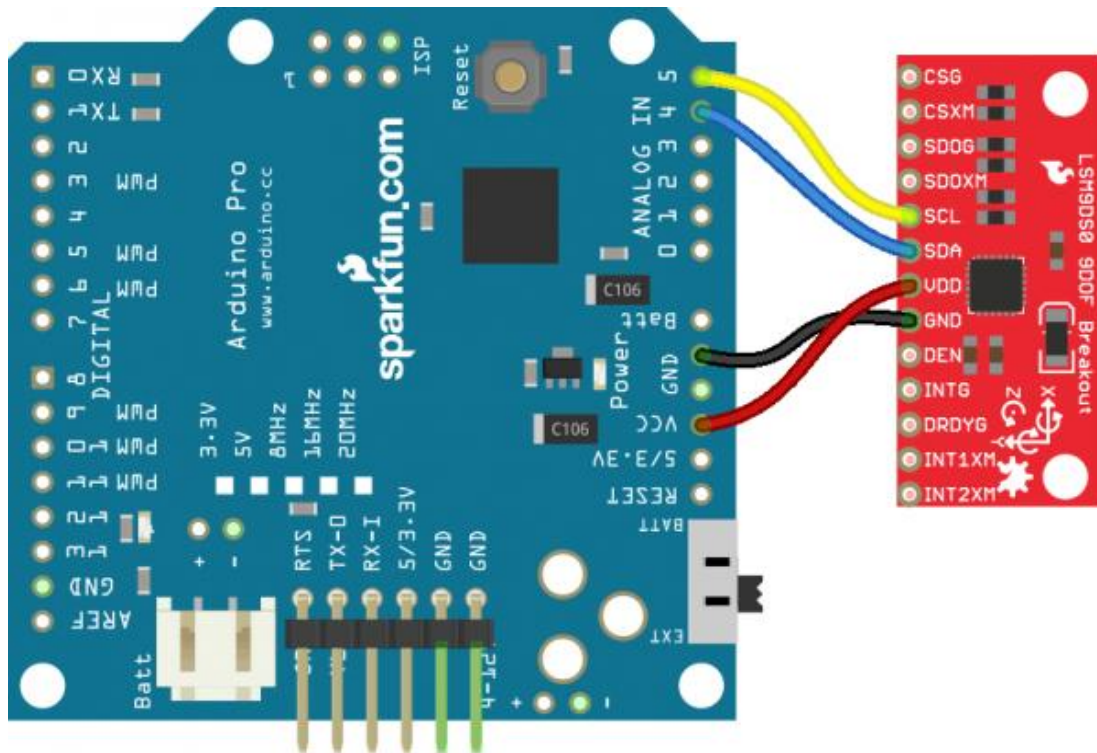
The library will work with either I²C or SPI. Since we're trying to be as frugal with our Arduino pins as possible, I²C it is! Here's a fritzing diagram for this example:

Connecting the LSM9DS0 to a RedBoard via a [Bi-Directional Logic Level Converter](#).

This hookup relies on all of the **jumpers** on the back of the board being set (as they should be, unless they've been sliced). If the jumpers have been disconnected, connect all four CS and SDO pins to 3.3V.

Since we're using I²C all we have to do is connect SDA to SDA and SCL to SCL. Unfortunately, since the LSM9DS0's **maximum operating voltage is 3.6V**, we need to use a [level shifting board](#) to switch between 3.3V and 5V.

Alternatively, if you have a 3.3V-operating Arduino – like the [3.3V/8MHz Pro](#) – you can connect SDA and SCL directly from microcontroller to sensor.



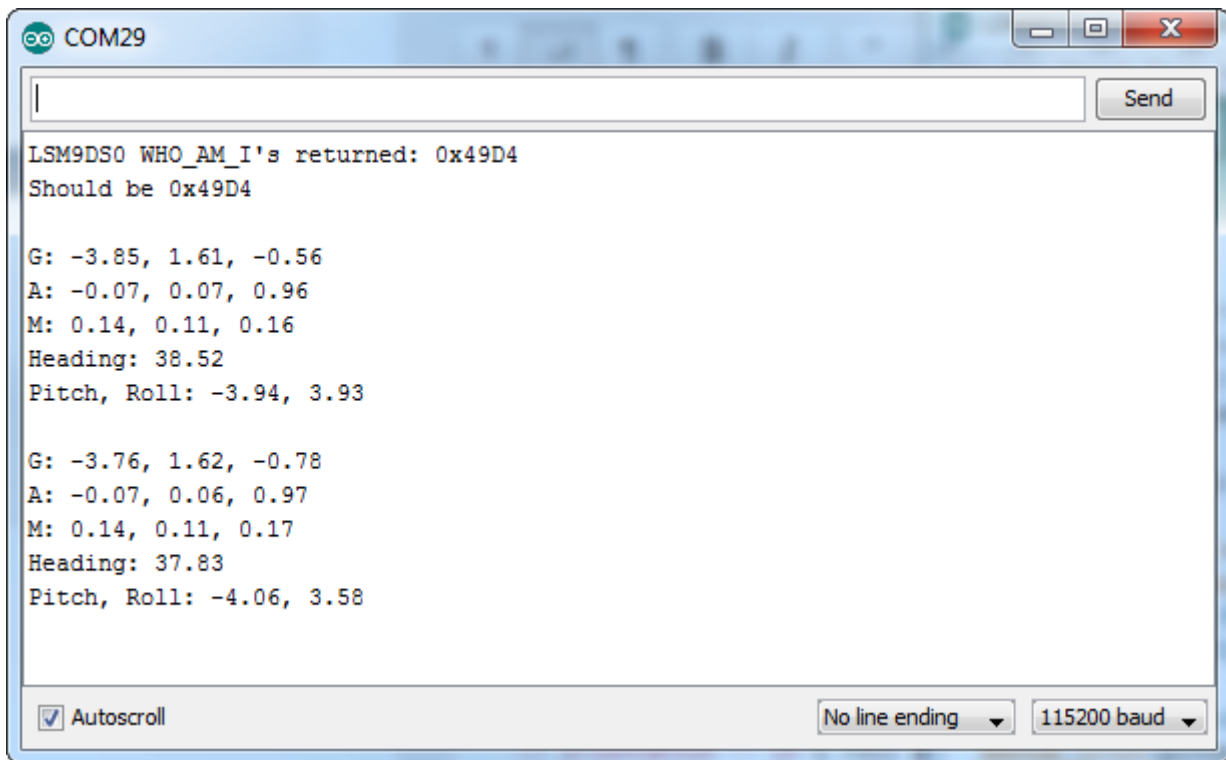
Heck, you can even mount the breakout board on top of the Arduino Pro. If you do this, you'll need to set A3 HIGH and A2 LOW. The sensor pulls little enough current that the Arduino's I/O pins can power it!

The wireless hookup: mounting an LSM9DS0 on top of an Arduino Pro. Pull A2 LOW and A3 HIGH to power the breakout.

4) Open the LSM9DS0_Simple Example

Once you've installed the library, open Arduino (or restart it if it was already open). You'll find this first example under the **File > Examples > SFE_LSM9DS0 > LSM9DS0_Simple**:

After uploading the code, open up your **serial monitor** and **set the baud rate to 115200 bps**. You should see something like this begin to stream by:



Each serial output blurb spits out the readings from all nine dimensions of movement. First the gyroscope readings (“G: x, y, z”) in **degrees per second** (DPS). Then come three degrees of acceleration in **g’s** (“A: x, y, z”), followed by the magnetic field readings (“M: x, y, z”) in **gauss** (Gs).

Try moving your breadboard around (carefully, don’t disconnect any wires!). Are the numbers changing? Check out the acceleration values – the axis normal to gravity should feel about 1 g of acceleration on it.

Does the **heading** output what you’d expect? If north seems a few degrees off, you may need to adjust for your **declination**. That means adding or subtracting a constant number that correlates to your location on [this map](#).

That’s all there is to it! If you want to get more out of the LSM9DS0 by using the interrupt outputs, check out the next page! Or check out the [Using the Arduino Library Page](#) for help using the library.

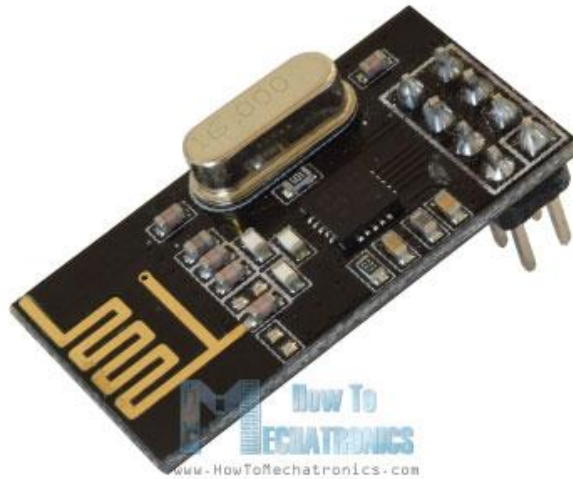
39.2.5 Resources and Going Further

Hopefully that info dump was enough to get you rolling with the LSM9DS0. If you need any more information, here are some more resources:

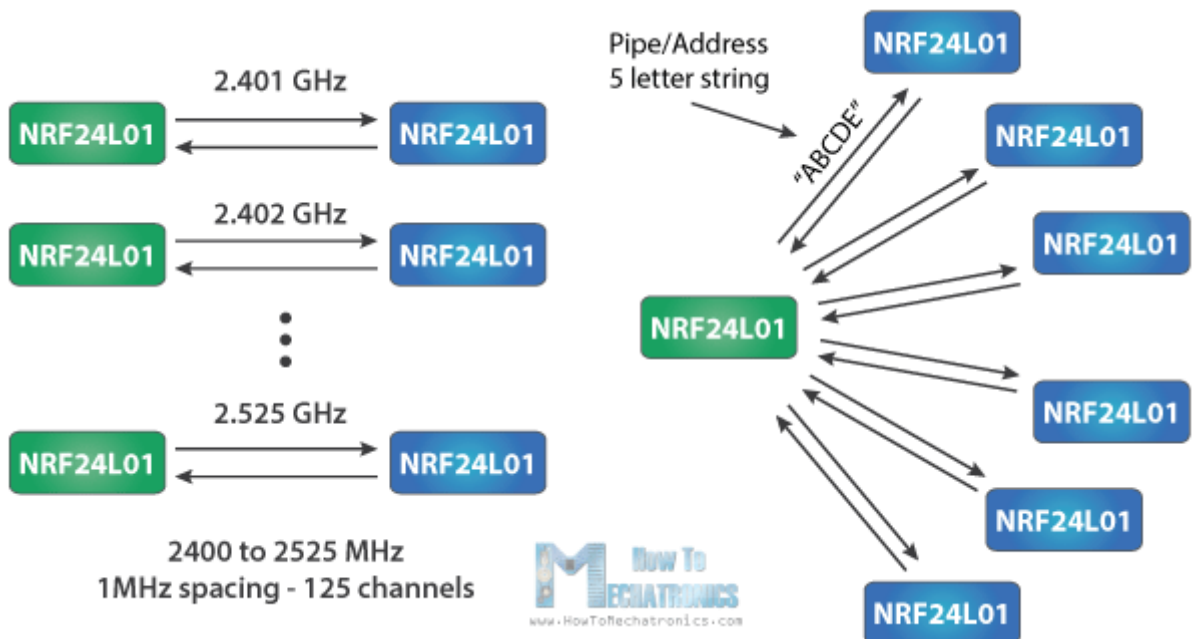
- [LSM9DS0 Datasheet](#) – This datasheet covers everything from the hardware and pinout of the IC, to the register mapping of the gyroscope and accelerometer/magnetometer.
- [LSM9DS0 Breakout Schematic](#)
- [LSM9DS0 Breakout EAGLE Files](#)

40 Wifi Communication with nRF24L01

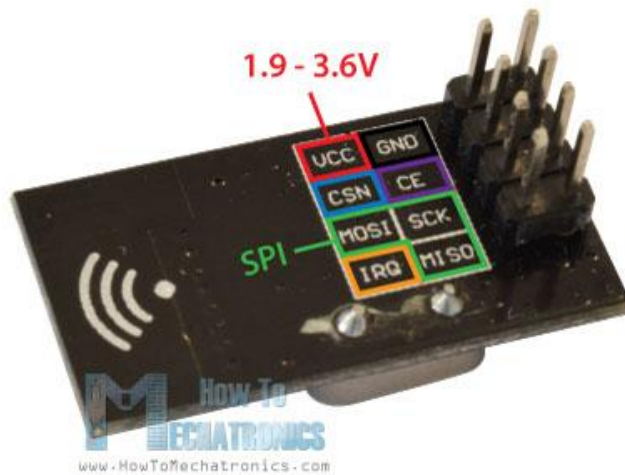
Let's take a closer look at the NRF24L01 transceiver module. It uses the 2.4 GHz band and it can operate with baud rates from 250 kbps up to 2 Mbps. If used in open space and with lower baud rate its range can reach up to 100 meters.



The module can use 125 different channels which gives a possibility to have a network of 125 independently working modems in one place. Each channel can have up to 6 addresses, or each unit can communicate with up to 6 other units at the same time.



The power consumption of this module is just around 12mA during transmission, which is even lower than a single LED. The operating voltage of the module is from 1.9 to 3.6V, but the good thing is that the other pins tolerate 5V logic, so we can easily connect it to an Arduino without using any logic level converters.

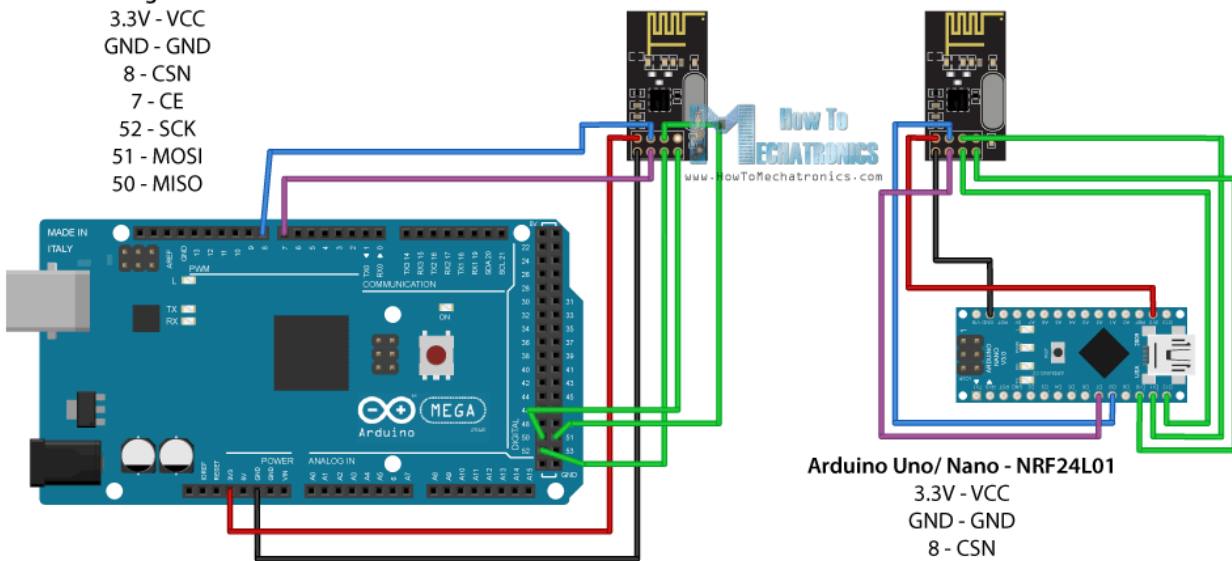


Three of these pins are for the SPI communication and they need to be connected to the SPI pins of the Arduino, but note that each Arduino board have different SPI pins. The pins CSN and CE can be connected to any digital pin of the Arduino board and they are used for setting the module in standby or active mode, as well as for switching between transmit or command mode. The last pin is an interrupt pin which doesn't have to be used.

So once we connect the NRF24L01 modules to the Arduino boards we are ready to make the codes for both the transmitter and the receiver.

Arduino Mega - NRF24L01

- 3.3V - VCC
- GND - GND
- 8 - CSN
- 7 - CE
- 52 - SCK
- 51 - MOSI
- 50 - MISO



Arduino Uno/ Nano - NRF24L01

- 3.3V - VCC
- GND - GND
- 8 - CSN
- 7 - CE
- 13 - SCK
- 11 - MOSI
- 12 - MISO

40.1 Arduino Library

First we need to download and install the [RF24 library](#) which makes the programming less difficult.



RF24-master.zip

Here are the two codes for the wireless communication and below is the description of them.

So we need to include the basic SPI and the newly installed RF24 libraries and create an RF24 object. The two arguments here are the CSN and CE pins.

```
1. RF24 radio(7, 8); // CE, CSN
```

Next we need to create a byte array which will represent the address, or the so called pipe through which the two modules will communicate.

```
1. const byte address[6] = "00001";
```

We can change the value of this address to any 5 letter string and this enables to choose to which receiver we will talk, so in our case we will have the same address at both the receiver and the transmitter.

In the setup section we need to initialize the radio object and using the `radio.openWritingPipe()` function we set the address of the receiver to which we will send data, the 5 letter string we previously set.

```
1. radio.openWritingPipe(address);
```

On the other side, at the receiver, using the `radio.setReadingPipe()` function we set the same address and in that way we enable the communication between the two modules.

```
1. radio.openReadingPipe(0, address);
```

Then using the `radio.setPALevel()` function we set the Power Amplifier level, in our case I will set it to minimum as my modules are very close to each other.

```
1. radio.setPALevel(RF24_PA_MIN);
```

Note that if using a higher level it is recommended to use a bypass capacitors across GND and 3.3V of the modules so that they have more stable voltage while operating.

Next we have the `radio.stopListening()` function which sets module as transmitter, and on the other side, we have the `radio.startListening()` function which sets the module as receiver.

```
1. // at the Transmitter
2. radio.stopListening();

1. // at the Receiver
2. radio.startListening();
```

In the loop section, at the transmitter, we create an array of characters to which we assign the message "Hello World". Using the `radio.write()` function we will send that message to the receiver. The first argument here is the variable that we want to be sent.

```
1. void loop() {
2.   const char text[] = "Hello World";
3.   radio.write(&text, sizeof(text));
4.   delay(1000);
5. }
```

By using the "&" before the variable name we actually set an indicating of the variable that stores the data that we want to be sent and using the second argument we set the number

of bytes that we want to take from that variable. In this case the `sizeof()` function gets all bytes of the strings “text”. At the end of the program we will add 1 second delay.

On the other side, at the receiver, in the loop section using the `radio.available()` function we check whether there is data to be received. If that’s true, first we create an array of 32 elements, called “text”, in which we will save the incoming data.

```
1. void loop() {
2.   if (radio.available()) {
3.     char text[32] = "";
4.     radio.read(&text, sizeof(text));
5.     Serial.println(text);
6.   }
7. }
```

Using the `radio.read()` function we read and store the data into the “text” variable. At the end we just print text on the serial monitor. So once we upload both programs, we can run the serial monitor at the receiver and we will notice the message “Hello World” gets printed each second.

41 Software Development on Arduino side with the Arduino integrated development environment (IDE)

41.1 Architecture of the program on the Arduino

Arduino IDE is a software which can be used to program an Arduino board; with C or C++ languages; referring to its special programming rules. It supplies a software library from the Wiring project, which provides many common input and output procedures.

To well writing the code; two functions are required:

Setup (): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

Loop (): is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.

Another functions furnished by the internal libraries; are also used in this program: pinMode (), digitalWrite (), and delay ().

The program code is then converted into a text file; in hexadecimal encoding using a specific program; that is loaded into the Arduino board by a loader program in the board's firmware. So you will give short strokes at a click; and the program becomes uploaded.

41.2 Bluetooth_Blimp_Control Code

```
#include <Servo.h> //include the library servo

Servo Servo1; //create 4 objects and rename each one
Servo Servo2;
Servo Servo3;
Servo Servo4;

#define MinAngle 45 //define the minimum and maximum angle for servo rotation
#define MaxAngle 135

#define DirPin 4 //choose 2 pin for the propeller direction and speed, pin 5 must be a PWM
pin.
#define SpdPin 5
```



```
int S1Val = 90; // define the initial angle of each servo
int S2Val = 90;
int S3Val = 90;
int S4Val = 90;

int PMotor = 0; //0: off, 1: fwd, 2: bck
//the propeller is initially off mode.
String str = "";

void setup() // to apply all initial value on servos and propeller
{
  Serial.begin(9600); //serial1 is to make sure that all informations arrive write.
  Serial1.begin(9600); //serial is a connection between bluetooth and arduino.
  //9600 is the board rate: the speed of communication between the bluetooth and the arduino.

  pinMode(DirPin, OUTPUT); // a function has 2 parameter(direction or speed) & (input or
output)
  pinMode(SpdPin, OUTPUT);

  // digitalWrite(DirPin, LOW);
  analogWrite(SpdPin, 0); // to be sure that the initial speed is zero.

  Servo1.attach(6); //to choose the servos pins between 14 pins in the arduino board; which
must be PWM pin.
  Servo2.attach(9);
  Servo3.attach(10);
  Servo4.attach(11);

  Servo1.write(S1Val); //to give each servo his initial value.
  delay(100); /*a pause between the motion of each 2 servos(so as not be exposed to a shortage
of electricity if the 4 servos start together)*/
  Servo2.write(S2Val);
  delay(100);
  Servo3.write(S3Val);
  delay(100);
  Servo4.write(S4Val);
}

void loop()
```

```

{
  if (Serial1.available())/*when an ordedr arrived to the bluetooth port this function become
valid
  so the processor enter into it and start to performs all steps inside */
  {
    S1Val = Serial1.parseInt(); //parsint: whatever the value was it will be taked as an integer
    S2Val = Serial1.parseInt(); /*to check the value arrived from the bluetooth to serial1
and send it to s1val(value for servo1)*/
    S3Val = Serial1.parseInt();
    S4Val = Serial1.parseInt();
    PMotor = Serial1.parseInt();
    MoveActuators();
  }
}

void MoveActuators()
{
  S1Val = constrain(S1Val, MinAngle, MaxAngle); /*define for each servo the minimum and
maximum angle
  to do not take an angle out of this interval*/
  S2Val = constrain(S2Val, MinAngle, MaxAngle);
  S3Val = constrain(S3Val, MinAngle, MaxAngle);
  S4Val = constrain(S4Val, MinAngle, MaxAngle);

  Serial.print(" S1: "); //to print all result
  Serial.print(S1Val);
  Serial.print(" S2: ");
  Serial.print(S2Val);
  Serial.print(" S3: ");
  Serial.print(S3Val);
  Serial.print(" S4: ");
  Serial.print(S4Val);
  Serial.print(" PMotor: ");
  Serial.println(PMotor);

  Servo1.write(S1Val); // to apply all ordered values
  delay(500);
  Servo2.write(S2Val);
  delay(500);
  Servo3.write(S3Val);

```

```
delay(500);
Servo4.write(S4Val);
delay(500);

switch (PMotor)
{
case 0:
    // turn motor off
//    digitalWrite(DirPin, LOW);
    analogWrite(SpdPin, 0);
    break;
case 1:
    // move motor cw (clock wise)
    digitalWrite(DirPin, LOW);
    analogWrite(SpdPin, 50);
    break;
case 2:
    // move motor ccw (conter clock wise)
    digitalWrite(DirPin, HIGH);
    analogWrite(SpdPin, 50);
    break;
}
}
```



Bluetooth_Blimp_Control.ino

41.3 IMU Sensor LSM9DS0 Code

```
#include <SPI.h> // Included for SFE_LSM9DS0 library
#include <Wire.h>
#include <SFE_LSM9DS0.h>

#define LSM9DS0_XM 0x1D // Would be 0x1E if SDO_XM is LOW
#define LSM9DS0_G 0x6B // Would be 0x6A if SDO_G is LOW
LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);

#define PRINT_SPEED 100 // 100 ms between prints
```

```
void setup()
{
  Serial.begin(9600); // Start serial at 115200 bps
  uint16_t status = dof.begin();

  Serial.print("LSM9DS0 WHO_AM_I's returned: 0x");
  Serial.println(status, HEX);
  Serial.println("Should be 0x49D4");
  Serial.println();
}

void loop()
{
  printOr();
  delay(PRINT_SPEED);
}

void printOr()
{
  dof.readGyro();
  dof.readAccel();
  dof.readMag();

  float pitch, roll;
  float x = dof.calcAccel(dof.ax), y = dof.calcAccel(dof.ay), z = dof.calcAccel(dof.az);

  pitch = atan2(x, sqrt(y * y) + (z * z));
  roll = atan2(y, sqrt(x * x) + (z * z));
  pitch *= 180.0 / PI;
  roll *= 180.0 / PI;

  Serial.print("#P: ");
  Serial.print(pitch, 2);
  Serial.print(", #R: ");
  Serial.println(roll, 2);
}
```



AccModule_Spark.i
no



41.4 Control Loop

```
void ReglerCheck(float pitch, float roll)
{
  S1Val = S3Val = 90 + roll;
  S2Val = S4Val = 90 + pitch;

  MoveActuators();
}
```



Avionics_Control_WAvionics_Control_W
ith_Regler_RoMeo.iirith_Regler_UNO.ino

41.5 Wifi Communication code

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

RF24 radio(9,10);

const uint64_t pipes[2] = { 0xDEDEDEDEE7LL, 0xDEDEDEDEE9LL };

boolean stringComplete = false; // whether the string is complete
static int dataBufferIndex = 0;
boolean stringOverflow = false;
char charOverflow = 0;

char SendPayload[31] = "";
char RecvPayload[31] = "";
char serialBuffer[31] = "";

void setup(void) {

  Serial.begin(9600);
```

```
printf_begin();
radio.begin();

radio.setDataRate(RF24_250KBPS);
radio.setPALevel(RF24_PA_MAX);
radio.setChannel(70);

radio.enableDynamicPayloads();
radio.setRetries(15,15);
radio.setCRCLength(RF24_CRC_16);

radio.openWritingPipe(pipes[0]);
radio.openReadingPipe(1,pipes[1]);

radio.startListening();
radio.printDetails();

Serial.println();
Serial.println("RF Chat V01.0");
delay(500);
}

void loop(void) {
  nRF_receive();
  serial_receive();
} // end loop()

void serialEvent() {
  while (Serial.available() > 0 ) {
    char incomingByte = Serial.read();

    if (stringOverflow) {
      serialBuffer[dataBufferIndex++] = charOverflow; // Place saved overflow byte into
buffer
      serialBuffer[dataBufferIndex++] = incomingByte; // saved next byte into next buffer
      stringOverflow = false; // turn overflow flag off
    } else if (dataBufferIndex > 31) {
      stringComplete = true; // Send this buffer out to radio
      stringOverflow = true; // trigger the overflow flag
    }
  }
}
```

```
charOverflow = incomingByte; // Saved the overflow byte for next loop
dataBufferIndex = 0; // reset the bufferindex
break;
}
else if(incomingByte=='\n'){
    serialBuffer[dataBufferIndex] = 0;
    stringComplete = true;
} else {
    serialBuffer[dataBufferIndex++] = incomingByte;
    serialBuffer[dataBufferIndex] = 0;
}
} // end while()
} // end serialEvent()

void nRF_receive(void) {
    int len = 0;
    if ( radio.available() ) {
        len = radio.getDynamicPayloadSize();
        radio.read(&RecvPayload,len);
        delay(5);

        RecvPayload[len] = 0; // null terminate string

        Serial.println("R:");
        Serial.print(RecvPayload);
        Serial.println();
        RecvPayload[0] = 0; // Clear the buffers
    }

} // end nRF_receive()

void serial_receive(void){

    if (stringComplete) {
        strcat(SendPayload,serialBuffer);
        // swap TX & Rx addr for writing
        radio.openWritingPipe(pipes[1]);
        radio.openReadingPipe(0,pipes[0]);
        radio.stopListening();
        radio.write(&SendPayload,strlen(SendPayload));
```

```

Serial.println("S:");
Serial.print(SendPayload);
Serial.println();
stringComplete = false;

// restore TX & Rx addr for reading
radio.openWritingPipe(pipes[0]);
radio.openReadingPipe(1,pipes[1]);
radio.startListening();
SendPayload[0] = 0;
dataBufferIndex = 0;
} // endif
} // end serial_receive()

```



RF24_config.h



RF24.h



printf.h



nRF24L01.h



Serial_Chat_Tested.i
no

41.6 Full code

```

#include <Servo.h> //include the library servo
#include <SPI.h> // Included for SFE_LSM9DS0 library
#include <Wire.h>
#include <SFE_LSM9DS0.h>

#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

// Comment out this section if you're using SPI
// SDO_XM and SDO_G are both grounded, so our addresses are:
#define LSM9DS0_XM 0x1D // Would be 0x1E if SDO_XM is LOW
#define LSM9DS0_G 0x6B // Would be 0x6A if SDO_G is LOW
// Create an instance of the LSM9DS0 library called `dof` the
// parameters for this constructor are:
// [SPI or I2C Mode declaration],[gyro I2C address],[xm I2C add.]
LSM9DS0 dof(MODE_I2C, LSM9DS0_G, LSM9DS0_XM);

RF24 radio(7,8);

```



```
Servo Servo1; //create 4 objects and rename each one
Servo Servo2;
Servo Servo3;
Servo Servo4;

// Do you want to print calculated values or raw ADC ticks read
// from the sensor? Comment out ONE of the two #defines below
// to pick:
#define PRINT_CALCULATED
//#define PRINT_RAW

#define PRINT_SPEED 50 // 500 ms between prints

#define MinAngle 45 //define the minimum and maximum angle for servo rotation
#define MaxAngle 135

#define DirPin 4 //choose 2 pin for the propeller direction and speed, pin 5 must be a PWM
pin.
#define SpdPin 5

int S1Val = 90; // define the initial angle of each servo
int S2Val = 90;
int S3Val = 90;
int S4Val = 90;

int PMotor = 0; //0: off, 1: fwd, 2: bck
//the propeller is initially off mode.
String str = "";

bool ReglerON = true;

const uint64_t pipes[2] = { 0xDEDEDEDEE7LL, 0xDEDEDEDEE9LL };

boolean stringComplete = false; // whether the string is complete
static int dataBufferIndex = 0;
boolean stringOverflow = false;
char charOverflow = 0;

char SendPayload[31] = "";
char RecvPayload[31] = "";
```

```
char sendBuffer[31] = "";

void setup() // to apply all initial value on servos and propeller
{
  Serial.begin(9600); //serial1 is to make sure that all informations arrive write.

  //while (!Serial);

  radio.begin();

  uint16_t status = dof.begin();

  Serial.print("LSM9DS0 WHO_AM_I's returned: 0x");
  Serial.println(status, HEX);
  Serial.println("Should be 0x49D4");
  Serial.println();

  pinMode(DirPin, OUTPUT); // a function has 2 parameter(direction or speed) & (input or
output)
  pinMode(SpdPin, OUTPUT);

  // digitalWrite(DirPin, LOW);
  analogWrite(SpdPin, 0); // to be sure that the initial speed is zero.

  Servo1.attach(6); //to choose the servos pins between 14 pins in the arduino board; which
must be PWM pin.
  Servo2.attach(9);
  Servo3.attach(10);
  Servo4.attach(11);

  Servo1.write(S1Val); //to give each servo his initial value.
  delay(100); /*a pause between the motion of each 2 servos(so as not be exposed to a shortage
of electricity if the 4 servos start together)*/
  Servo2.write(S2Val);
  delay(100);
  Servo3.write(S3Val);
  delay(100);
  Servo4.write(S4Val);

  radio.setDataRate(RF24_250KBPS);
```

```
radio.setPALevel(RF24_PA_MAX);
radio.setChannel(70);

radio.enableDynamicPayloads();
radio.setRetries(15,15);
radio.setCRCLength(RF24_CRC_16);

radio.openWritingPipe(pipes[0]);
radio.openReadingPipe(1,pipes[1]);

radio.startListening();
delay(500);
}

void loop()
{
  nRF_receive();
  printOr();
}

void printOr()
{
  dof.readGyro();
  dof.readAccel();
  dof.readMag();

  float pitch, roll;
  float x = dof.calcAccel(dof.ax), y = dof.calcAccel(dof.ay), z = dof.calcAccel(dof.az);

  pitch = atan2(x, sqrt(y * y) + (z * z));
  roll = atan2(y, sqrt(x * x) + (z * z));
  pitch *= 180.0 / PI;
  roll *= 180.0 / PI;

  //Serial1.print("#P: ");
  //Serial1.print(pitch, 2);
  //Serial1.print(",#R: ");
  //Serial1.println(roll, 2);

  String data = "#P: ";
```

```
data = data + pitch;
data = data + ",#R: ";
data = data + roll;
data.toCharArray(sendBuffer, 31);
nRF_send();

if(ReglerON)
  ReglerCheck(pitch, roll);
else
  delay(PRINT_SPEED);
}

void ReglerCheck(float pitch, float roll)
{
  S1Val = S3Val = 90 + roll;
  S2Val = S4Val = 90 + pitch;

  MoveActuators();
}

void MoveActuators()
{
  S1Val = constrain(S1Val, MinAngle, MaxAngle); /*define for each servo the minimum and
maximum angle
to do not take an angle out of this interval*/
  S2Val = constrain(S2Val, MinAngle, MaxAngle);
  S3Val = constrain(S3Val, MinAngle, MaxAngle);
  S4Val = constrain(S4Val, MinAngle, MaxAngle);

  Serial.print(" S1: "); //to print all result
  Serial.print(S1Val);
  Serial.print(" S2: ");
  Serial.print(S2Val);
  Serial.print(" S3: ");
  Serial.print(S3Val);
  Serial.print(" S4: ");
  Serial.print(S4Val);
  Serial.print(" PMotor: ");
  Serial.println(PMotor);
```

```
Servo1.write(S1Val); // to apply all ordered values
delay(50);
Servo2.write(S2Val);
delay(50);
Servo3.write(S3Val);
delay(50);
Servo4.write(S4Val);
delay(50);

switch (PMotor)
{
  case 0:
    // turn motor off
    //digitalWrite(DirPin, LOW);
    analogWrite(SpdPin, 0);
    break;
  case 1:
    // move motor cw (clock wise)
    digitalWrite(DirPin, LOW);
    analogWrite(SpdPin, 50);
    break;
  case 2:
    // move motor ccw (conter clock wise)
    digitalWrite(DirPin, HIGH);
    analogWrite(SpdPin, 50);
    break;
}
}

void nRF_receive(void) {
  int len = 0;
  if ( radio.available() ) {
    len = radio.getDynamicPayloadSize();
    radio.read(&RecvPayload,len);
    delay(5);

    RecvPayload[len] = 0; // null terminate string

    Serial.print("R:");
    Serial.print(RecvPayload);
```

```
Serial.println();

char* command = strtok(RecvPayload, ",");
while(command != 0) {
    *command = 0;
    S1Val = atoi(command);
    ++command;
    S2Val = atoi(command);
    ++command;
    S3Val = atoi(command);
    ++command;
    S4Val = atoi(command);
    ++command;
    PMotor = atoi(command);

    MoveActuators();
}

if(S1Val == 90 && S2Val == 90 && S3Val == 90 && S4Val == 90)
    ReglerON = true;
else
    ReglerON = false;

RecvPayload[0] = 0; // Clear the buffers
}
}

void nRF_send(){
    strcat(SendPayload,sendBuffer);
    // swap TX & Rx addr for writing
    radio.openWritingPipe(pipes[1]);
    radio.openReadingPipe(0,pipes[0]);
    radio.stopListening();
    radio.write(&SendPayload,strlen(SendPayload));

    Serial.print("S:");
    Serial.print(SendPayload);
    Serial.println();

    // restore TX & Rx addr for reading
```

```
radio.openWritingPipe(pipes[0]);  
radio.openReadingPipe(1,pipes[1]);  
radio.startListening();  
SendPayload[0] = 0;  
dataBufferIndex = 0;  
}
```



Avionics_Control_W
ith_Regler_RoMeo_\\

42 Assembly

42.1 Actuator Assembly

Each part which was described and discussed in the previous chapter will be seen in this chapter in the final assembly; which contains: the programmed arduino, 4 servos; bluetooth module; DC Motor; and the battery. All that we have mentioned appears in the following photo (*Figure 67: the final assembly of servo actuator system and its interfaces.*).

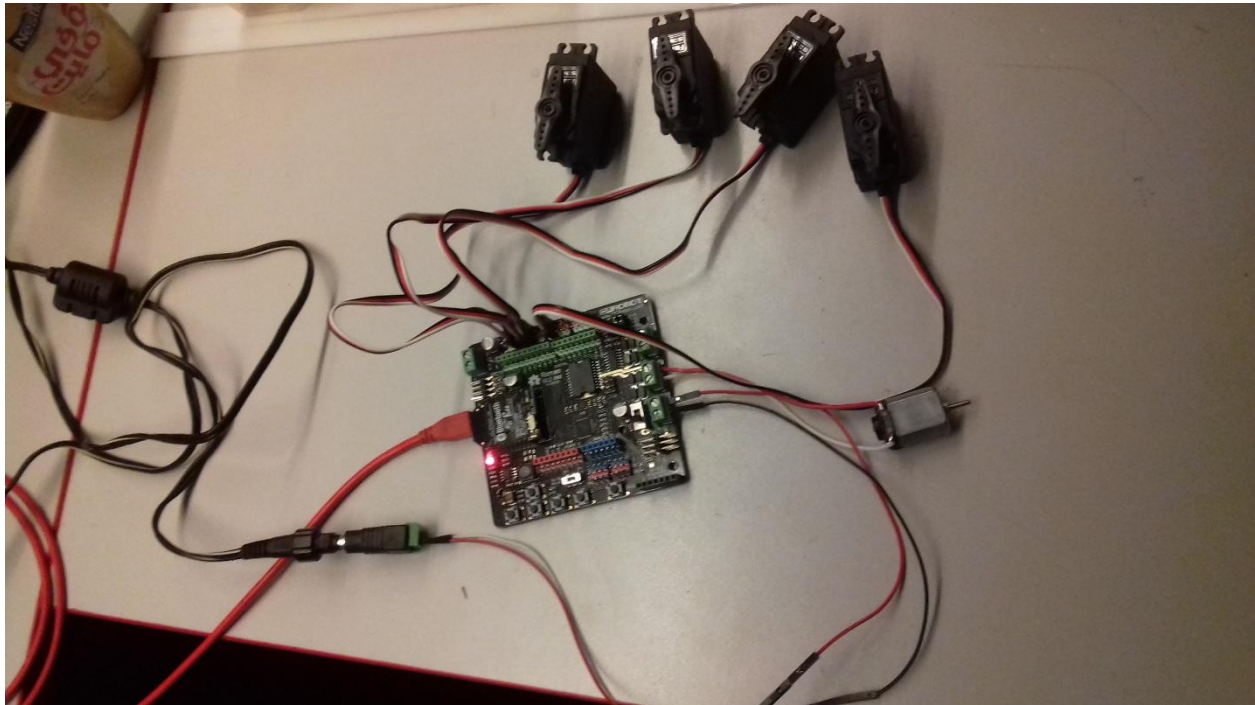
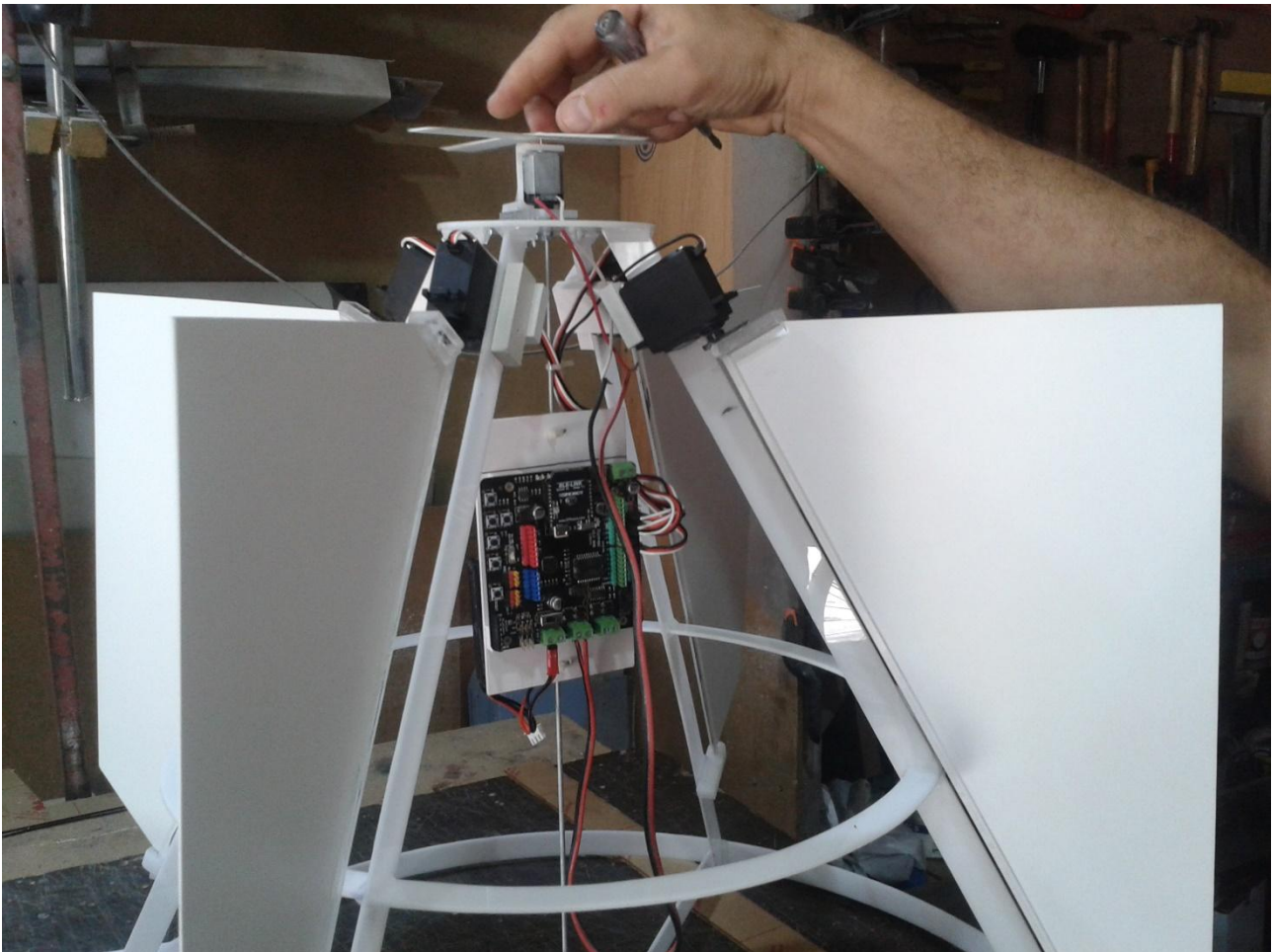


Figure 78: the final assembly of servo actuator system and its interfaces.

In effect the program already written will then uploaded to the arduino board. Once the upload finished; and the arduino supplied by the compatible power ; all servos turned to their intial position (90^0) as defined in the code.



To test this assembly there are several steps to follow. First; make sure that all peripheral devices are well connected to the Arduino board; and the board is supplied by the compatible battery. The light seen on the Arduino confirms the arrival of current into the board. Then; verify the position of all servos on PWM pins (6, 9, 10, and 11) and the DC motor position. Later; start to connect your mobile Bluetooth to BLE link Bluetooth.

And now we start the control using the mobile application "BlunoBasicDemo". To do that you must enter 5 number between each 2 numbers a comma. The 4 first numbers will be the rotation angle of all 4 servos; which must be between the minimum (45°) and the maximum (135°) angles. And the last number can be 0 (to turn off the propeller's motor) and 1 (to turn it on).

This process is illustrated in the following figure: (*Figure 68*: showing how to use BunoBasicDemo to give order to our system.

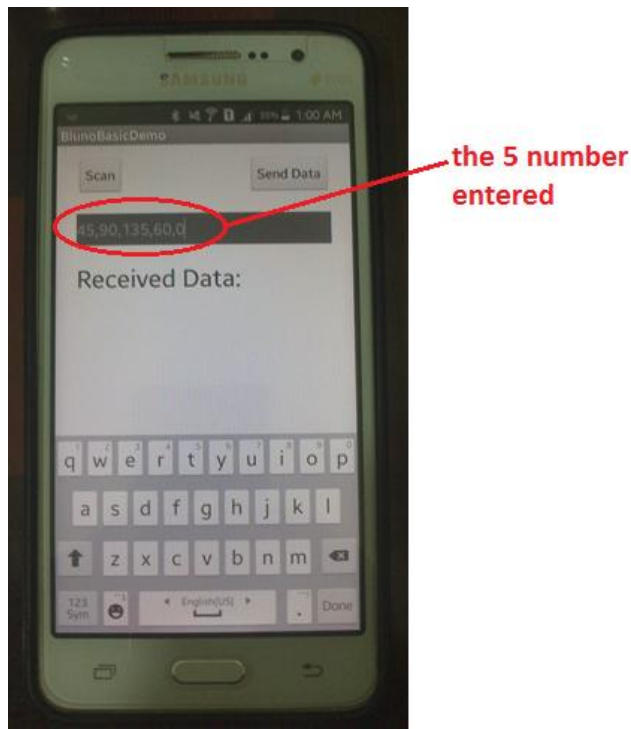
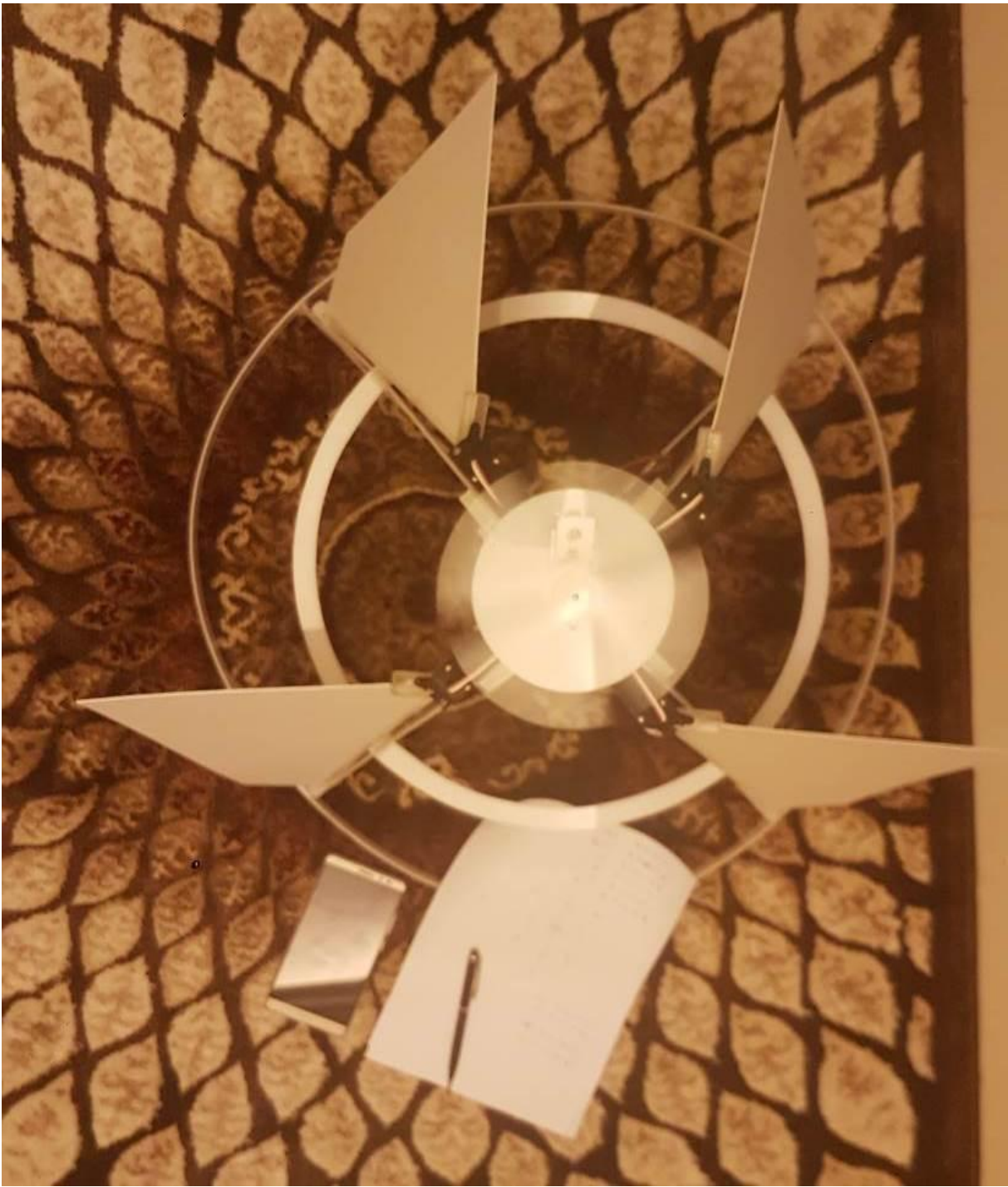
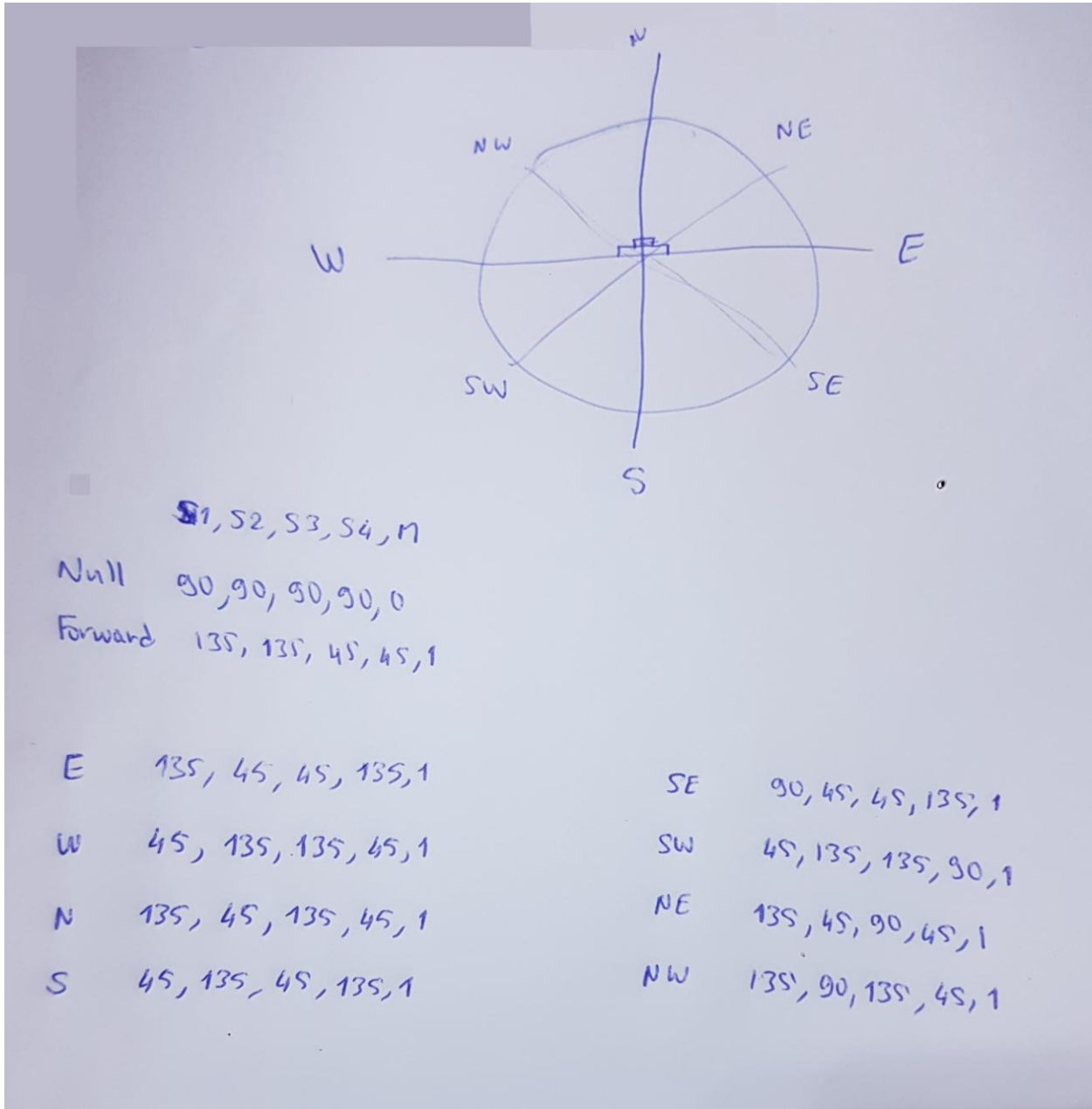


Figure 79: showing how to use BunoBasicDemo to give order to our system.
Each servo will turns in the required angle.

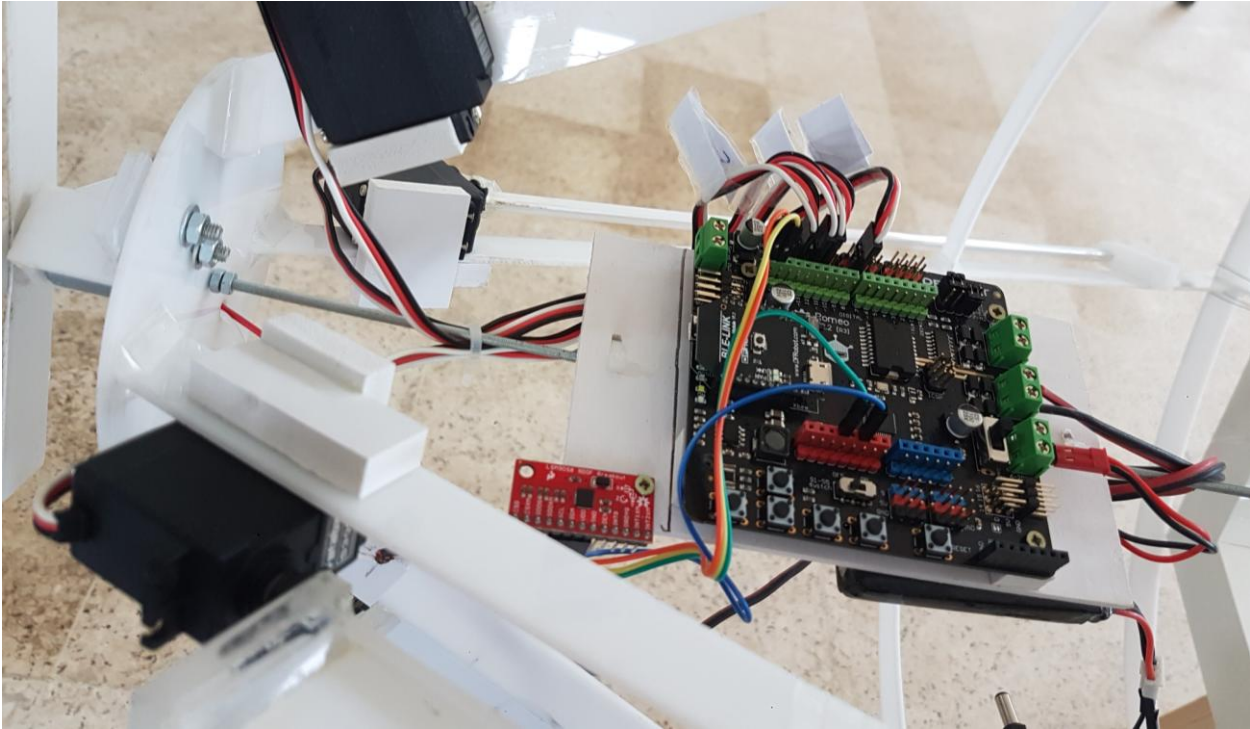




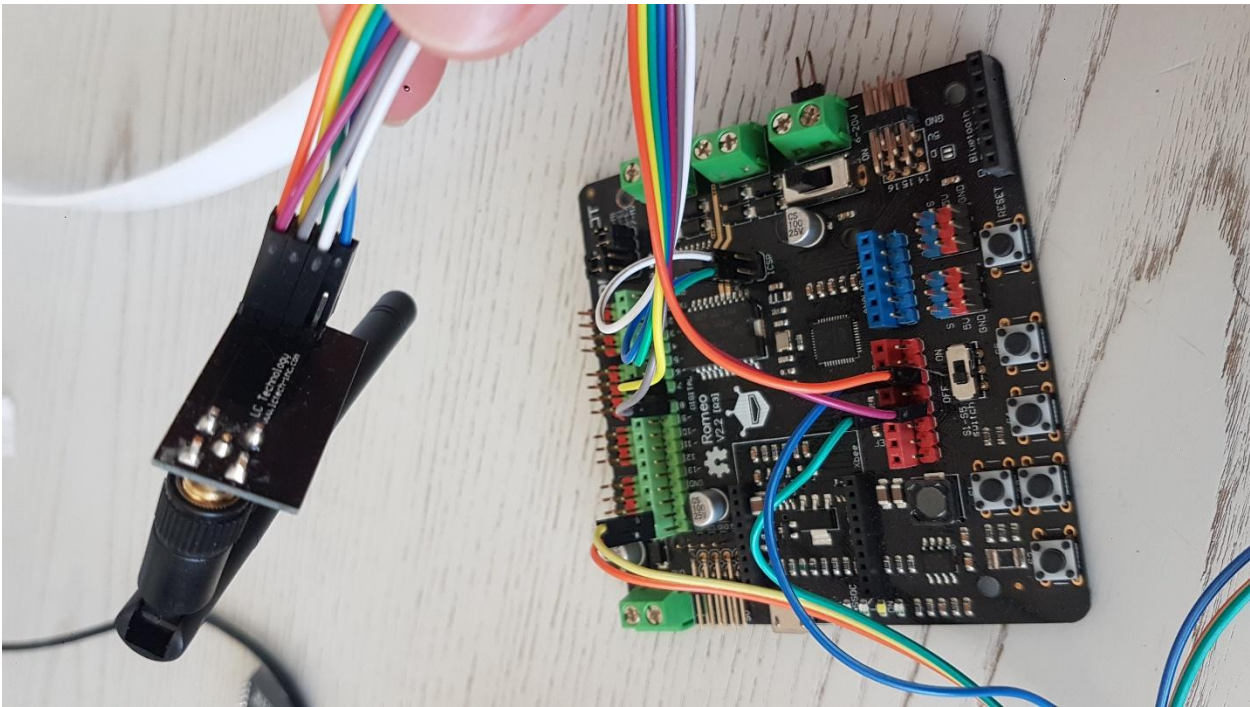
Servos angle:



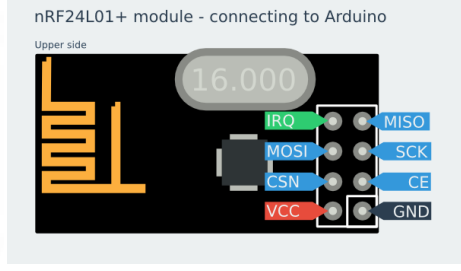
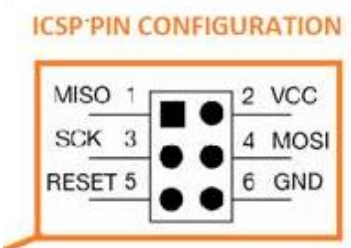
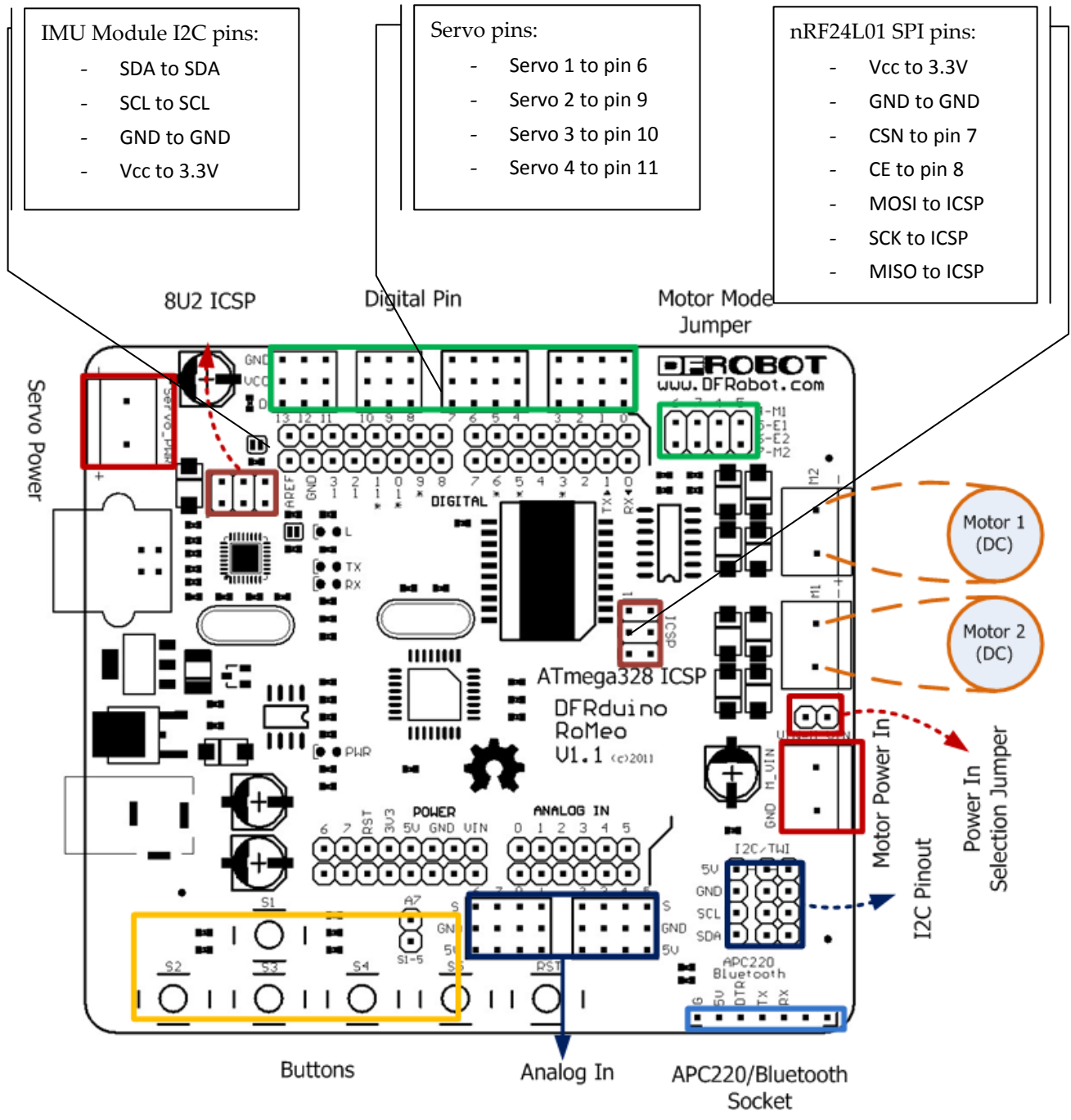
42.2 IMU Assembly



42.3 Wifi Module Assembly



42.4 Full Assembly



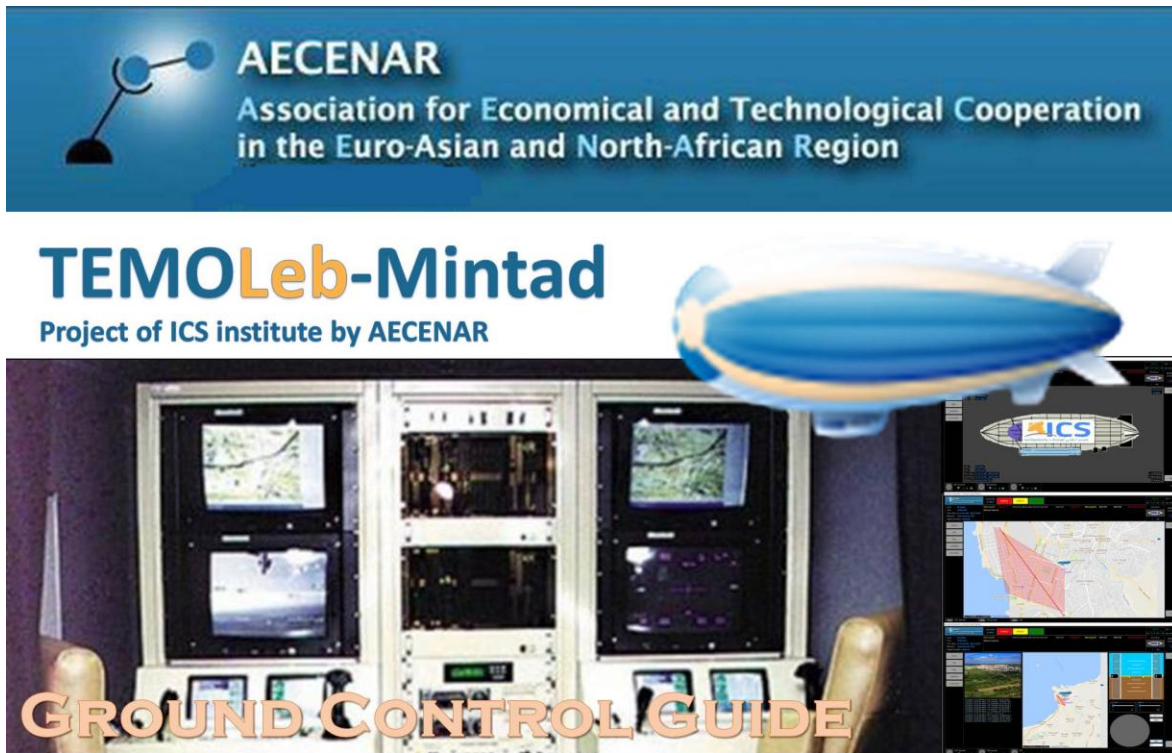
43 Final Implementation



43.1 Migrated to small prototype



44 Mintad Prototype Test Rig Quick Guide



When the **Mintad** in the idle status (float status), it will regulates itself to keep straight forwarded



To tracking and control **Mintad** Connect the communication unit to a computer containing the GCS software by USB.



Open the GSC software and accept the privileges he ask for.



Go to: Config > Connection > then choose the serial port of the communication unit, all other field should keep as default.

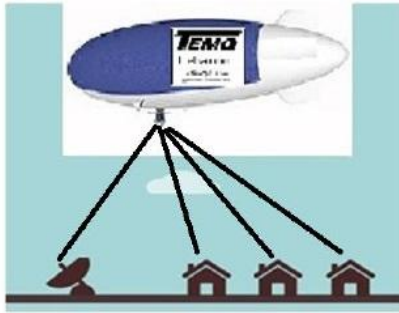


Go to the ASI widget to check the **Mintad** orientation, states will be getting lively from the FCS.



Go to the Control widget to send command to the FCS to move.

Marketing Concept (2017-2019)



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Qalamoun-COM_Mintad 2017

نظام لتوزيع انترنت للقلمون والجوار



90.000\$	كلفة المنظومة (بالون واحد زائد ايليكترونيك التحكم ومنظومة الاتصالات)
	كلفة التشغيل سنوياً
\$36,000	كلفة التشغيل (للمراقبة شخص في كل دوام، 3 دوامات = 3 اشخاص)، \$1200 للشخص الواحد شهريا
\$14,000	تكاليف تصليح (سنويا)
	الربح سنوا (مدخول ناقص تكاليف التشغيل والتصليح)
	سعر الانترنت للمستخدم: \$20 في الشهر، 3000 مستخدم -> المدخول سنويا \$240,000
	الربح سنويا \$190,000

Calculation: The concept is to take a fast internet bundel from OGERO and to distribute it.

من OGERO	عدد المستهلكين الاجمالي	معدل الاستهلاك الشهري [GB]
الرسم الشهري لـ 100 GB (ل.ل.)	3000	20
LBP 100.000	ADSL More than 8Mbps	
	عدد المستهلكين تسعهم الـ package من OGERO	
	5	

الربح السنوي (\$) \$240.000

الربح هنا هو الدخل السنوي لمشروع المنطاد

بيع الخدمة للمستهلك (ل.ل.)

LBP 30.000

الربح علي الـ package

LBP 50.000

الربح الاجمالي في الشهر

LBP 30.000.000



Registration in Germany

Unternehmen in der Anmeldung (to be filled in)	Abteilung in Deutschland, die den Antrag stellt (to be filled in)	Formular der Firma Nr. (to be filled in)
Seite 1 von 1		
1. Name des Antragstellers (Antragsteller)		
a. Name		
b. Name (vollständiger Name)		
c. Name (Abkürzung, vollständiger Name, Abkürzung)		
d. Name		
2. Organisation des Antragstellers		
a. Name (Abkürzung, vollständiger Name, Abkürzung)		
b. Name		
c. Name (Abkürzung, vollständiger Name, Abkürzung)		
d. Name		
3. Name des Antragstellers (Antragsteller)		
a. Name		
b. Name (vollständiger Name)		
c. Name (Abkürzung, vollständiger Name, Abkürzung)		
d. Name		
4. Name des Antragstellers (Antragsteller)		
a. Name		
b. Name (vollständiger Name)		
c. Name (Abkürzung, vollständiger Name, Abkürzung)		
d. Name		

German Address:
Im Klingenberg 2a,
69123 Heidelberg, Germany

Lebanese Address:
Bahsas, Haykalieh Str., Harba Bld.,
Ground Flr., Tripoli, Lebanon

Satellite Based Communication Systems

مشروع TEMOLeb_SatInterface

من خلال المشروع الحالي نهدف الى انشاء محطات ابراج لتسهيل التواصل بالاقيمار الصناعية لتسهيل وتسريع التواصل، ومن اجل هذا الهدف لا بد من انشاء عدة تقنيات من بينها برنامج التحكم بالابرارج وايضا لتتمكن المستخدمين من الوصول الى خدماتنا من خلال واجهة مستخدم من عدة منصات كالحاسب والهاتف .

البرنامج منقسم الى جزئين

واجهة المدراء (Operators Interface)
من خلال هذه الواجهة يقوم للمدراء كل حسب تخصصه برؤية الاقمار الاستيعابية والابرارج المنتشرة لمعرفة كمية سر عملها واكتشاف الاحتماء والمشاكل الفورية للسعي الى حلها

واجهة المستخدم (User Interface)
هذه الواجهة مخصصة للمستخدم النهائي او العميل التي من خلالها يستطيع التسجيل والاشتراك بالإضافة الى استخدام الخدمات التوفرة له، يجب ان تتوفر هذه الواجهة على عدة منصات كالهاتف والحاسب وأنظمة تشغيل ليتمكن كل شخص مهما كان نوع جهازه من الاشتراك كما يجب ان تكون بسيطة وسهلة الاستخدام بعيدة عن التعقيد وبمعية قدر الامكان لتتناسب جميع الاجهزة .

Balloon Based Communication Systems

Applications:

- Offshore internet supply (on sea)

Actual Project: Internet supply for Gas exploration facilities in front of the Lebanese coast

- High Altitude Communication Platform

Literature

Diplomarbeit (DA) Samir Mourad 2000

DA Jamal Ebeidieh 2002

DA Muhammad Subhan 2002

DA Abdelfattah Ammar 2005

...