بسم الله الرحمن الرحيم

MEGBI Vaccine Pilot Plant (MEGBI-VPP) - 3rd Project Report

(Jan 2014 – Dec 2015)

- Rough Business Plans and Commercializing Market Strategy
- Specification and System Design of Downstream Process of Hepatitis B DNA Vaccine Pilot Plant
- Detailed Mechanical Design of Devices
- Automation System (Graphical User Interface, Adaptation to Sensors/Actuators, Functional operation algorithm for full automation)
- Manufatoring of a minimal system (without disc stack centrifuge, homogenizer, 2 columns)

Authors:

Samir Mourad, Jihad Samarji, Zaher Chendeb, Ibrahim Zaaroura, Haitham Hindy

Initial Document: 10 Dec 2014

Last update/آخر تعديل: 7 January 2016

مركز أبحاث الشرق الأوسط للجينات والتقنية البيولوجية

رأسنحاش – قضاء البترون– لبنان

Middle East Genetics and Biotechnology Institute (MEGBI)

A Member Institute of AECENAR

Main Road, Ras-Nhache, Batroun, Lebanon

http://www.aecenar.com/institutes/megbi

www.temo-ek.de

TEMO Biotechnology

TEMO e.K., Im Klingenbühl 2a, 69123 Heidelberg, Germany

المضمون

Project Status at beginning of this project phase

مركز ابحاث الشرق الاوسط
للجينات والتقنية البيولوجية
http://aecenar.com/institutes/megbi
ابحاث عن ألقحة

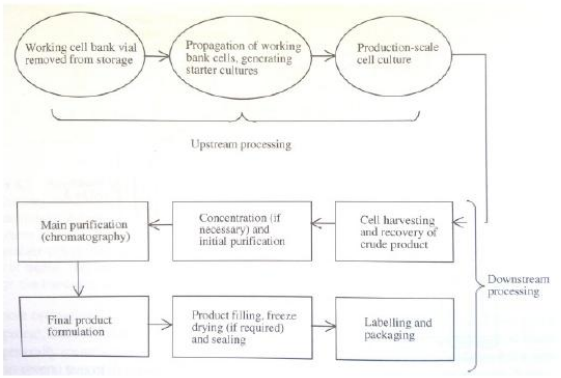
AECENAR
Association for Economical and Technological Cooperation in the Euro-Asian and North-African Region
www.aecenar.com

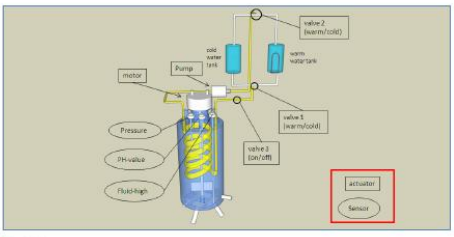
Recombinant Vaccine Technology / Biotechnological Upstream & Downstream Processing

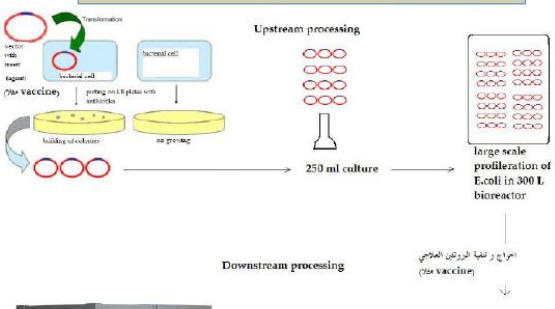
Hepatitis B DNA Vaccine Pilot Plant

صناعة القحة عن طريق البيوتكنولوجيا

Last update: 03 December 2013







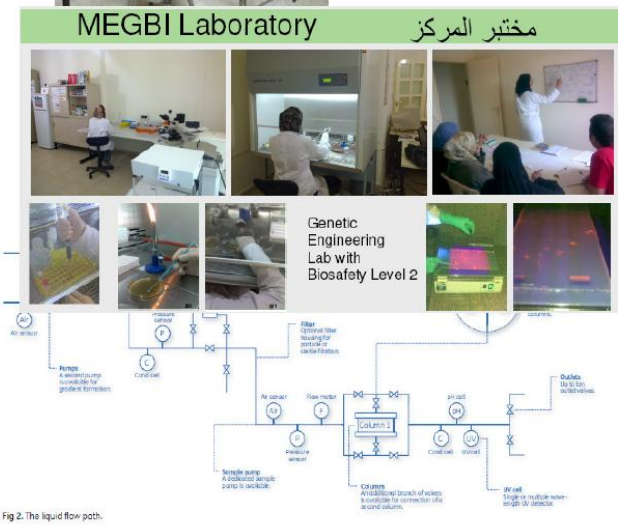


Fig 2. The liquid flow path.

اعمال حالية:

- Prototype for a Chromatographic Purification Machine for Production Scale

الحاجيات لعام 2014:

- 2-1 اشخاص (مثلا طلاب ماستر في الهندسة الادواة الطبية او في الAutomation Engineering)
- $34.000 تكلفة مواد

Contact:
Samir Mourad
Mobile: +96176341526
samir.mourad@aecenar.com

1. Project Management / إدارة المشروع

1.1 Project goal /هدف العمل

The goal is to install a DNA vaccine production plant.

1. HBSAg vaccine production based on S.cerevisae
2. later on migration to MAB production in E.coli

1.2 Budget Planning

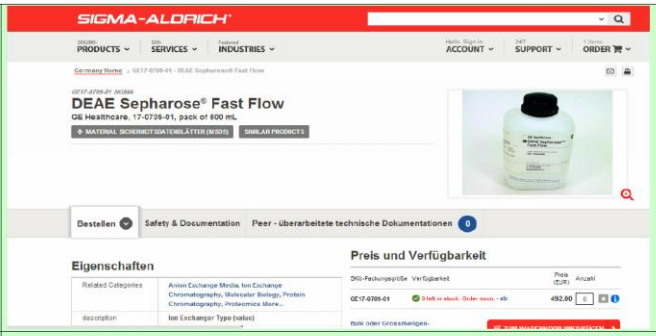
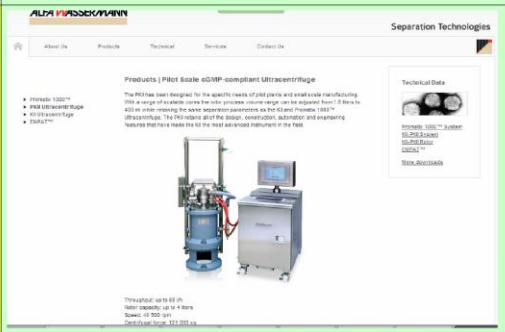
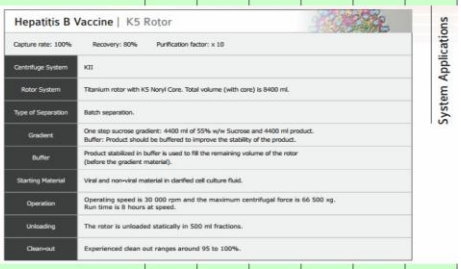
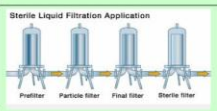
1.2.1 *Akta Based Plant*

Smaller Scale	Prices 2014	Prices 2015 including transport and douane in Lebanon
Bioreactor (offer from 2013)	130.000 €	175.760 €
Uniflux (Filtration) 10	172.000 €	232.544 €
Aktaready	140.000 €	189.280 €
Ultracentrifuge	50.000 €	67.600 €
DNA Laboratory		130.000 €
Total		795.184 €

Medium Scale	Prices 2014	Prices 2015 including transport and douane in Lebanon
Bioreactor Xcell	240.000 €	324.480 €
Uniflux (Filtration) 30	305.000 €	412.360 €
AktaProcess	170.000 €	229.840 €
Ultracentrifuge	50.000 €	67.600 €
DNA Laboratory		130.000 €
Total		1.164.280 €

1.2.2 *Patent Based semi-own manufactoring*

Last update: 24.12.2014

Device No.	Device Name	Price including transport & customs	Supplier/image
1	Disc Stack Centrifuge 1	$25.000	
2	Homogenizer	$15.000	 
3	Storage Tank/ PEG Precipitation mixing tank 100L	$2.000	
4	Disc Stack Centrifuge 2	$25.000	
5	Storage Tank/ CaCl2 Precipitation mixing tank 100L	$2.000	
6	Ultrafiltration Device	$5.000	
7	Storage Tank 100L for UF	$2.000	
8	20 L Chromatography column	$3.000	
9	20 L Sepharose 4B	$18.000	
10	Storage Tank 100L	$2.000	
11	20 L Chromatography column	$3.000	
12	10 L DEAE matrix DFF100 DEAE-Sepharose® Fast Flow (Sigma Aldrich)	$17.000	
13	Storage Tank 100L	$2.000	
14	Ultracentrifugation CsCl	$150.000	 
15	20 L Chromatography column	$3.000	
16	20 L Sepharose 4BCLFF	$18.000	
17	Storage Tank 100L	$2.000	
18	sterile filtration 0.22 Mikrometer	$2.500	
19	Chemicals	5000	
20	Engineering 6 man months	18000	
21	Project Management 6 man months	24000	
	Total	**$343.500**	

1.2.3 *Budget for MEGBI Vaccine Production Pilot Plant (MEGBI-VPP)*

MEGBI-VPP
DNA Vaccine Pilot Plant for production of Hepatitis B virus
2015

Last Update: 04. Feb 15

DNA Transfer Laboratory

device	price
Safety Cabinet B2	8.000 €
centrifuge	5.000 €
Shake incubator	2.500 €
Thermocycler	2.500 €
blood freezer	3.500 €
Gel rack with voltage	1.500 €
Precision Balance	800 €
Minus 80 °C freezer	10.000 €
UV Machine	1.100 €
vortex	300 €
precision pipettes	500 €
molecular biological reagences	4.000 €
chemicals	2.000 €
laboratory materials	1.500 €
laboratory tables	1.500 €
IT	1.000 €
fluorescence microscope	3.000 €

Transportation and customs	6.000 €
Building Lab 2009-11	25.000 €
Total Laboratory	**79.700 €**

Upstream Processing (Bioreactor)

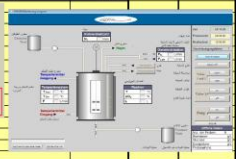
Material Costs

System	Piece	#	Piece price	Total
Behälter (1301 Stainless)		1	600 €	600 €
Temperiersystem	Behälter	2	50 €	100 €
	Abdeckwolle	1	100 €	100 €
	Blech	1	120 €	120 €
	Pumpe	1	80 €	80 €
	Beheizungsrad	1	60 €	60 €
Aut. Valve		2	200 €	400 €
Temp.Sensor		1	20 €	20 €
PH Sensor		1	100 €	100 €
PO_2 Sensor		1	1.200 €	1.200 €
Ausfluß	Aut.Valve	1	200 €	200 €
Medium Einfluß	Aut.Valve	1	200 €	200 €
	Behälter	1	50 €	50 €
PH Regulierung	Behälter	2	50 €	100 €
	Aut.Valve	2	200 €	400 €
Beimpfungsreinigung	Behälter	1	50 €	50 €
	Aut.Valve	1	200 €	200 €
Control System	Siemens S7 PLC	1	1.500 €	1.500 €
	PC	1	500 €	500 €
			Sum Material	5.980 €

Personnel Costs

Task	duration (months)	Qualification	price / month	Total
Integration Mechanics	1	Eng.	2.000 €	2.000 €
Integration Automation System	1	Eng.	2.000 €	2.000 €
Programming Control AECENAR	1	Eng.	2.000 €	2.000 €
Project Managing	3	Eng.	2.000 €	6.000 €
			Sum Personnel Costs	**12.000 €**

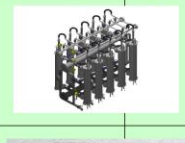
Personal Specific Costs	cost per month
Student	0 €
Specialized Worker	1.000 €
Engineer	2.000 €

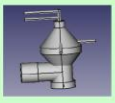


Total Upstream Processing Unit (Bioreactor) 17.980 €

Downstream Processing

	price per piece	pieces needed	Costs	Needed Staff
Homogenizer	1.000 €	1	1.000 €	
Prototype Chromatogr. Process Device (mech)	13.000 €	3	39.000 €	
Prototype Chromatogr. Process Device(autom)			0 €	
Ultrafiltration Unit	1.500 €	1	1.500 €	
Ultracentrifuge (used)			0 €	
Disc Stack Centrifuge	10.500 €	1	10.500 €	
Mixing/Storing tanks	1.000 €	4	4.000 €	
Pump	700 €	6	4.200 €	
Automation (PLC system, GUI)			10.000 €	
Total Devices	73.200 €	Integration (piping, ...)	3.000 €	**Total Downstream Processing Unit 76.200 €**





Project Management

		#months	monthly rate for manage-ment			
Program Managing over 9 months		9	2.000,00 €		18.000 €	
Documentation					4.000 €	Total Project Management 22.000 €
Total MEGBI-VPP	**195.880 €**	planned project duration: Dez. 2014 - August 2015				

MEGBI-VPP
DNA Vaccine Pilot Plant for production of Hepatitis B virus
2015

Last Update: 04. Feb 15

DNA Transfer Laboratory

device	price
Safety Cabinet B2	8.000 €
centrifuge	5.000 €
Shake incubator	2.500 €
Thermocycler	2.500 €
blood freezer	3.500 €
Gel rack with voltage	1.500 €
Precision Balance	800 €
Minus 80 °C freezer	10.000 €
UV Machine	1.100 €
vortex	300 €
precision pipettes	500 €
molecular biological reagences	4.000 €
chemicals	2.000 €
laboratory materials	1.500 €
laboratory tables	1.500 €
IT	1.000 €
fluorescence microscope	3.000 €

Transportation and customs	6.000 €
Building Lab 2009-11	25.000 €
Total Laboratory	**79.700 €**

Upstream Processing (Bioreactor)

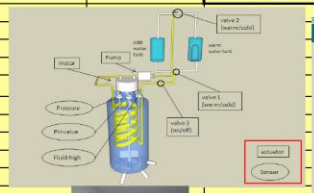
Material Costs

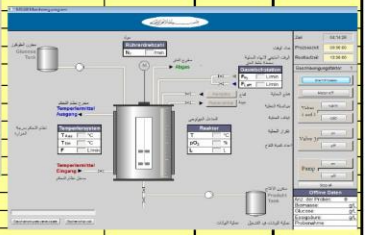
System	Piece	#	Piece price	Total
Behälter (130l Stainless)		1	600 €	600 €
Temperiersystem	Behälter	2	50 €	100 €
	Abdeckwolle	1	100 €	100 €
	Blech	1	120 €	120 €
	Pumpe	1	80 €	80 €
	Beheizungsrad	1	60 €	60 €
Aut. Valve		2	200 €	400 €
Temp.Sensor		1	20 €	20 €
PH Sensor		1	100 €	100 €
PO_2 Sensor		1	1.200 €	1.200 €
Ausfluß	Aut.Valve	1	200 €	200 €
Medium Einfluß	Aut.Valve	1	200 €	200 €
	Behälter	1	50 €	50 €
PH Regulierung	Behälter	2	50 €	100 €
	Aut.Valve	2	200 €	400 €
Beimpfungsreinigung	Behälter	1	50 €	50 €
	Aut.Valve	1	200 €	200 €
Control System	Siemens S7 PLC	1	1.500 €	1.500 €
	PC	1	500 €	500 €
			Sum Material	5.980 €

Personnel Costs

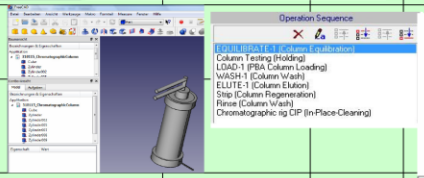
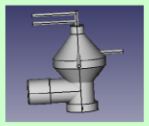
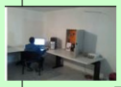
Task	duration (months)	Qualification	price / month	Total
Integration Mechanics	1	Eng.	2.000 €	2.000 €
Integration Automation System	1	Eng.	2.000 €	2.000 €
Programming Control	1	Eng.	2.000 €	2.000 €
AECENAR Project Managing	3	Eng.	2.000 €	6.000 €
			Sum Personnel Costs	**12.000 €**

Personal Specific Costs	cost per month
Student	0 €
Specialized Worker	1.000 €
Engineer	2.000 €





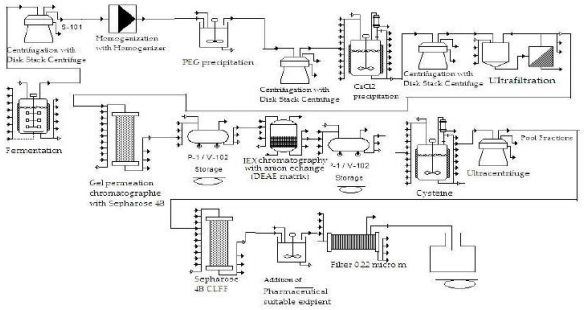
Total Upstream Processing Unit (Bioreactor)	**17.980 €**

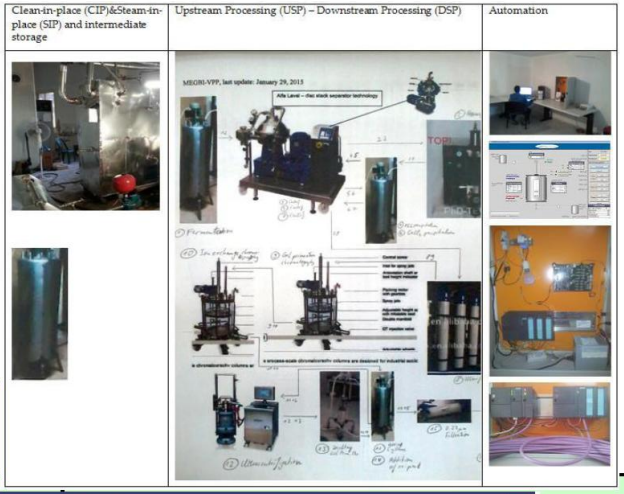
Downstream Processing					price per piece	pieces needed	Costs	Needed Staff		
Homogenizer					1.000 €	1	1.000 €			
Prototype Chromatogr. Process Device (mech)					13.000 €	3	39.000 €			
Prototype Chromatogr. Process Device(autom)							0 €			
Ultrafiltration Unit					1.500 €	1	1.500 €			
Ultracentrifuge (used)							0 €			
Disc Stack Centrifuge	 				10.500 €	1	10.500 €			
Mixing/Storing tanks					1.000 €	4	4.000 €			
Pump					700 €	6	4.200 €			
Automation (PLC system, GUI)	  						10.000 €			
				Total Devices	73.200 €	Integration (piping, ...)	3.000 €	Total Downstream Processing Unit		76.200 €
Project Management			#months	monthly rate for manage-ment						
	Program Managing over 9 months		9	2.000,00 €		18.000 €				
	Documentation					4.000 €		Total Project Management		**22.000 €**
	Total MEGBI-VPP	**195.880 €**		planned project duration: Dez. 2014 - August 2015						

This calculation was presented to LASeR (Prof Moustafa Jazar, Dr Ammar Assoum, Dr Bachar al-Hasan) and discussed at LU Doctoral School in Tripoli (Dr Ammar Assoum, Dr Bachar al-Hasan, Dr Mohammad Khalil). Result: LASeR waits for the opinion of Prof Monzer Hamze (Status 17 March 2015), one phone call with Dr Monzer in March 15, then no further anwers)

Missing Budget to complete a low cost MEGBI-VPP prototype

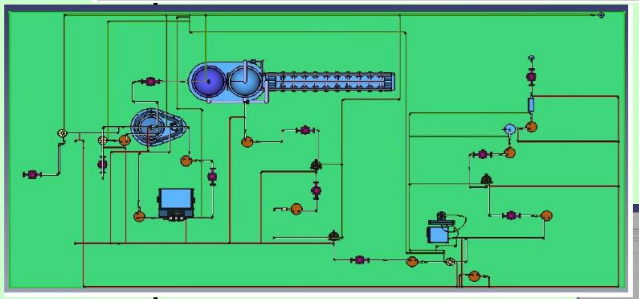
29.11.2015

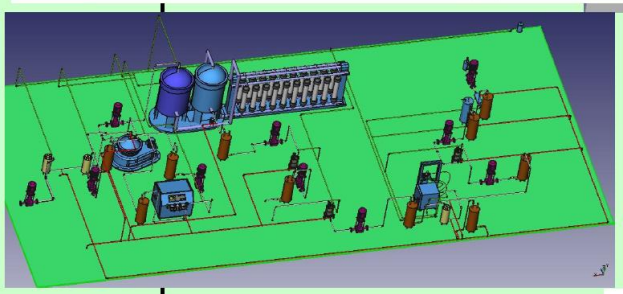




Open Issues

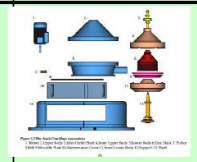
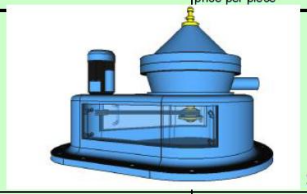
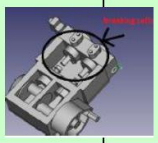
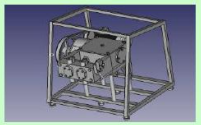
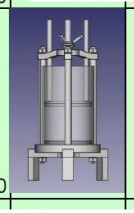
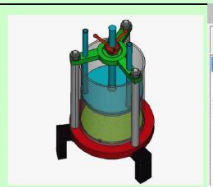
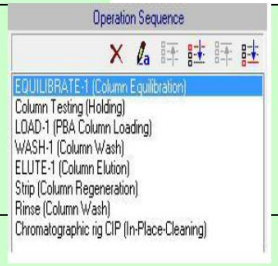
aus Deutschland zu holen ca. Dez. 2015



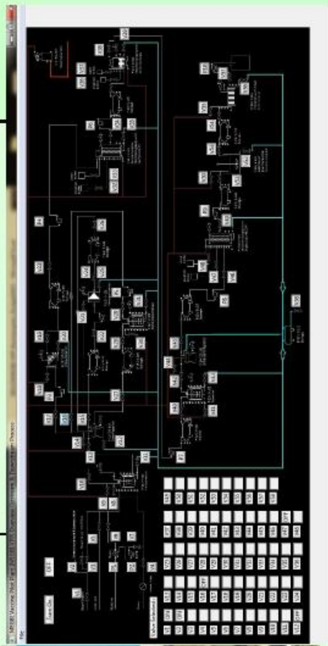


Upstream Processing Mech.Parts

System	Piece	# of open pieces	Piece price	Total
Behälter (130l Stainless)		0	$500	$0
Temperiersystem	Behälter	2	$50	$100
	Abdeckwolle	1	$100	$100
	Blech	1	$120	$120
	Pumpe	1	$80	$80
	Beheizungsrad	1	$60	$60
Aut. Valve		2	$200	$400
Temp.Sensor		1	$20	$20
PH Sensor		1	$100	$100
~~PO_2 Sensor~~		0	$1.200	$0
Ausfluß	Aut.Valve	1	$150	$150
Medium Einfluß	Aut.Valve	1	$150	$150
	Behälter	1	$50	$50
PH Regulierung	Behälter	2	$50	$100
	Aut.Valve	2	$200	$400
Beimpfungsreinigung	Behälter	1	$50	$50
	Aut.Valve	1	$200	$200
		0	$1.500	$0
	PC	0	$500	$0
			Sum Material	~~$2.080~~

Downstream Processing	# of mechnical pieces	average complexity/cost of part			price per piece	pieces needed	Costs	Needed Staff			
Disc Stack Centrifuge	20	$70			$1.400	1	$1.400				
Homogenizer	10	$120			$1.200	1	$1.200				
Chromatogr. Column (mech)	8	$50			$400	3	$1.200				
Ultrafiltration Unit					$200	1	$200				
Ultracentrifuge (used)	20	$70			$1.400	1	$1.400				

Mixing/Storing tanks			$60	14
			$840	
Total Devices Integration (piping, ...) 1/2 inch			$1.500	
			$6.240	
Total DSP			$7.740	



Automation System

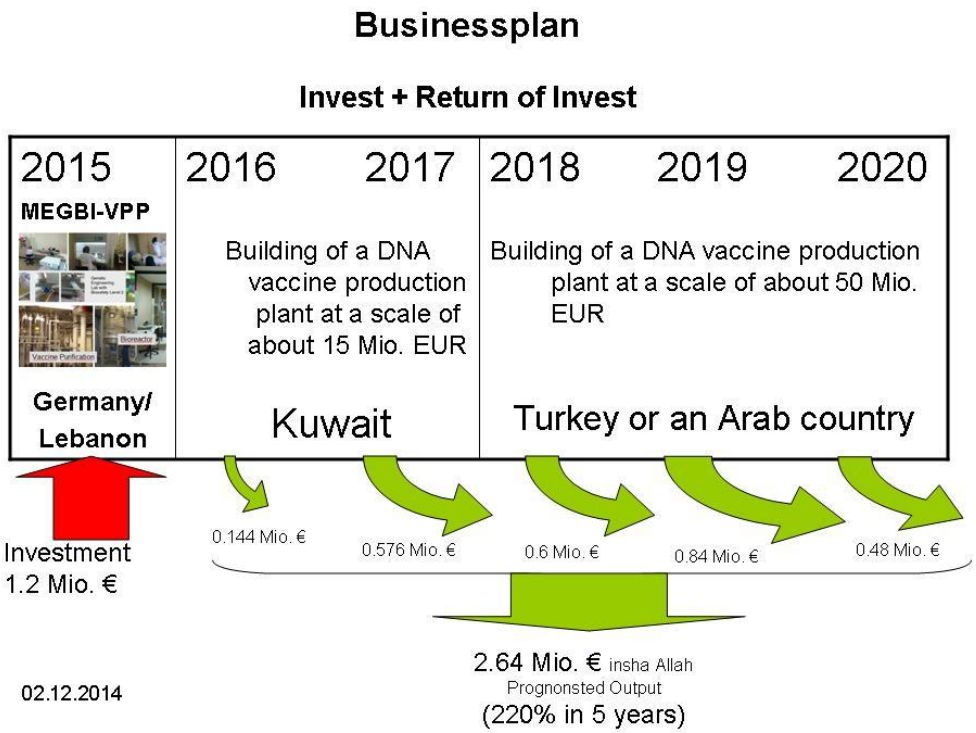
Item	price per piece	# of pieces
pump	$60	10
electronic valve	$50	58
Total price		$3.500

Mechanical Parts USP	$0
Mechanical Parts DSP	$7.740
Automation System	$3.500
Total MEGBI-VPP Rest Budget Requirement	**$11.240**

This version includes:

- smaller vessels
- simple disc stack centrifuge
- cost reduction option: vessel with chemical opening of cells instead of homogenizer
- disc stack centrifuge instead of ultracentifuge
- main new manufactoring: chromatographic columns (requires fine mechanical manufactoring facility)

1.3 Business Plan 1 (presented Dec 2014)



1.3.1 *Investors for Business Plan 1*

Übersicht der Investoren am MEGBI-VPP

Stand: **31.12.2014**

Investoren

Gesamtentwicklungswert **1.200.000 €**

Investor	Höhe des Investitionswertes	Anteile am Gewinn (Entwicklung) bis April 2011	Bemerkung/Datum der Investition
Amine Bouafif	100,25 €	0,0083542%	Investion bezahlt (Überweisung ca. 11.12.14)
Nasser Al Araimi	1.200 €	0,1000000%	Investition bezahlt (Überweisung 27.12.14)
David Yildiz	600 €	0,0500000%	Investion bezahlt (bar ca. 8.12.14)
AECENAR	133.000€	11,0833333%	DNA Labor 130TEUR, Miete Jan-Jun 15 3TEUR
Summe:	134.900€	11,2416875%	
Restentwicklungsanteile TEMO	1.065.100 €	88,76%	

derzeit ist der größte Teil der Projektdokumente öffentlich zugänglich und hier einsehbar:

http://temo-ek.de/8.html

1.3.2 *Location Concept of Business Plan 1*

Cooperation between Europe and the Middle East

By the cooperation with Turkish and Arab partners the MEGBI-VPP project creates working possibilities for the young educated people in the Middle East region and helps to stabilize the region.

And with the help of God, the Almighty, this will be a big effort for a better and peaceful future for the two neighbour regions Europe and North-Africa/Middle East.

The genetic engineering laboratory of the pilot plant (for step 1) is based in North Lebanon at the nice village Ras Nhache. Threre were also undergone the prestudies of the project.

Bioreactor (step 2) and purification (step 3) will be implemented at TEMO Biotechnology site at Heidelberg, Germany

The administration of the project will be done by TEMO Biotechnology in Germany.

TEMO Biotechnology
TEMO e.K.
Im Klingenbühl 2a
69123 Heidelberg, Germany

Email: info@temo-ek.de
Website: www.temo-ek.de

1.4 Business Plan 2 (Presented Feb 2015 to Azm Association)

Bismillah

MEGBI-VPP Businessplan and Feasibility Study

18.2.2015

1. The Goal: a Facility able to produce vaccines and antibodies based on fermentation and purification (biotechnological upstream and downstream process)

The Goal is a company like this

Facility History

- **1988 -** Cetus Corporation, Emeryville, CA, US establishes its European headquarters, EuroCetus B.V. in Amsterdam, the Netherlands.
- **1990-1991 -** EuroCetus facility licensed for production and release of ProLeukin (IL-2) for Europe and rest of world, except US and Japan.
- **1992 –** Chiron Corporation acquires Cetus Corporation. Eurocetus B.V. becomes Chiron B.V.
- **1996 –** Facility licensed for commercial production of Pertussis vaccine.
- **1997 –** Facility licensed by WHO for commercial production of Meningitis A polysaccharides.
- **1997-1999 –** Facility licensed by UK, MCA and Italian Health Branch for Meningitis C polysaccharide and CRM197 carrier protein for conjugated vaccine Menjugate®.
- **2000 –** Facility acquired by Dr William J. Rutter. SynCo Bio Partners B.V. established.
- **2001 –** Facility licensed by EMA and Health Canada for Meningitis C polysaccharides and CRM197 carrier protein for Menjugate® production.
- **2004 –** Installation of new air handling systems completed for all of its facilities.
- **2005 –** Installation of new aseptic filling machine in its Class A zone, allowing larger batches sizes and expansion of filling ranges.
- **2005 –** Expansion of GMP facilities to allow production of a wider range of protein, vaccine and live biotherapeutic products.
- **2006 –** Facility licensed by Korean FDA for commercial production.

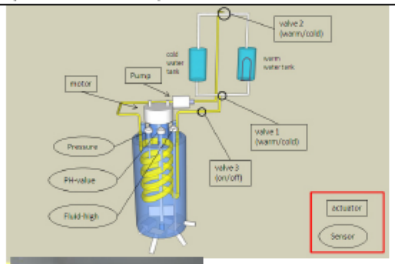
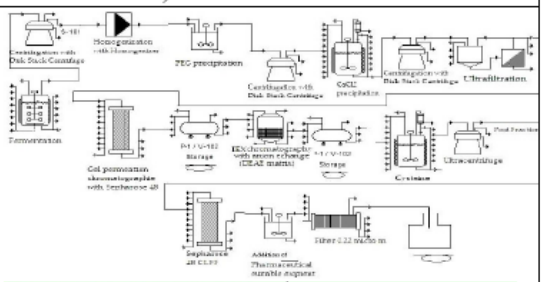
- **2008 –** Expansion of process development capabilities to enable a greater range of projects to be completed.
- **2009 –** Facility licensed by ANVISA for commercial production.
- **2011 –** Expansion of the Class A zone of its aseptic filling facility successfully completed, allowing SynCo to support new product launches in the US and the rest of the world.

2. Business Plan for our plant

Year	2009-2015	2016	2017	2018
	MEGBI Vaccine Production Pilot Plant (MEGBI-VPP)		Licenced for production of diagnostic antibodies (diagnosis of blood groups) in Lebanon/Jordan/Turkey and other Arab Countries	Licenced for production of Hepatitis B antigen vaccine (similar to Engerix) in Lebanon and other Arab Countries
Investment Costs	**120,000 €** **(still open to complete: 65.000 EUR)**	**4 Mio. EUR**		
Return of Invest			About 20 x 250 blood laboratories&blood banks: 2017-2019 5000x40x12 EUR = 2.4 Mio. EUR income per year **7.2 Mio. EUR**	

2.1. 2009-2015

MEGBI-VPP pilot plant shall convince an investor to invest in the next steps

Genetic Engineering Laboratory	Upstream Processing (Bioreactor)	Downstream Processing Unit (Protein Purification)
	 	 

2.2. 2016

A plant like this:

Azm Association (Faisal Maulawi, Dr Dani Saaduddin, Dr Kifah Tout) visited AECENAR Center at Ras Nhache on 6th March 2015 and Business Plan 2 was discussed. Result (Status 17th March 2015): Azm wants a more detailed business plan with detailed market strategy.

1.5 Time Schedule / الجدول الزمني

Nov/Dec 14: Financement and Concept Phase

Jan – June 15: Finishing of Development of MEGBI Vaccine Production Pilot Plant (MEGBI-VPP)

	Planned	Staff
System Design with SuperPro	15.12.-24.12.2014	
Automation System (GUI, veleman board)	1.3.-15.10.2015	Haitham Hindi (Master Student)
Design of mechanical parts with FreeCAD (Vessels, Storage Tanks, columns, Disc Stack Centrifuge, Ultracentrifuge, Homogenizer)	Mai - July 2015	Practicants from BAU (Jihad, Zaher, Ibrahim, Fadi)
Materials List (sensors, actuators, chemicals to be purchased)		
Manufactoring a low cost version of MEGBI-VPP (	16.3.-30.7.15	

2. Basics

2.1 Recovery of recombinant S.cerevisae cells

From The Elements of Immunology , https://books.google.com.lb/books?isbn=8131711587, Fahim Halim Khan:

use is hepatitis B vaccine. A single gene for the major surface antigen of *hepatitis B virus* (HbsAg) is cloned in yeast cells. The recombinant yeast cells are grown in fermenters. HbsAg, the surface antigen, is expressed and accumulates inside the yeast cells. The yeast cells are then harvested and then burst open by high pressure releasing the recombinant HbsAg (among other proteins). HbsAg is then purified by a standard biochemical technique such as affinity chromatography. The purified antigen has been shown to induce humoral immunity. This approach has been used to make several

From Catherine Charcosset, Membrane Processes in Biotechnology and Pharmaceutics:

3.3.2 Concentration and clarification of cells

Another common application of MF is the concentration and washing of cultures of single-cell organisms where the product is intracellular or cell associated [7]. Common cases include recombinant yeast cultures producing proteins and antigen particles, and recombinant *E. coli* producing proteins in the form of solid inclusion bodies. The next step is usually cell lysis and recovery of the product from cell debris. Separation of product from cell debris can often be performed by MF, as detailed in the next section.

Russotti et al. [107] have used cross-flow MF for the harvest of recombinant yeast in a short period of time to minimize the risk for product degradation. MF studies with flat sheet membranes showed high throughput with initial fluxes on the order of water fluxes (>1000 l m^{-2} h^{-1}, regime I, <2 min), followed by a rapid decay towards a low pseudo-steady state flux (20 l m^{-2} h^{-1}, regime II, >2 min). Large pore membranes (0.65 μm) were found to be more suitable for harvesting yeast (10 μm size) without cell leakage than smaller pore ones (0.22 μm and 0.45 μm). Among operating parameters, feed flow rate (i.e. shear rate) had a significant impact on average flux, whereas change in *TMP* afforded little improvement. In another recent example, Lee [108] concentrated recombinant yeast cells using a cross-flow MF unit containing a 0.2 μm membrane device. The concentrated cells were homogenized by several passes through a high-pressure homogenizer. The homogenate was then clarified using cross-flow MF. The clarified material was concentrated by UF and buffer-exchanged before delivering the material to down-stream for further purification.

3.3.3 Separation of products from fermentation broth

Microorganisms are sources of valuable enzymes, proteins and other bio-products. They produce two basic types of biological molecules: extracellular, which are excreted into a growth medium, and intracellular, which are retained inside the cytoplasm of the cells [9]. A variety of host microorganisms have been studied. The most often used organisms are *E. coli*, *S. cerevisiae* and *Bacillus subtilis*. Several other microbial strains have been used for production of microbial enzymes, such as *Aspergillus niger* and *Kluyveromyces fragilis* (for production of catalase), *Saccharomyces lactis* and *Kluyveromyces lactis* (β-galactosidase), *Bacillus coagulans* and

From Process Scale Bioseparations for the Biopharmaceutical Industry, edited by Abhinav A. Shukla, Mark R. Etzel, Shishir Gadam:

fouling due to media components, and influence of osmotic pressure. Patel et al. [21] have compared the different filter formats: pleated-sheet microfilter, tubular microfilter, and hollow fiber ultrafiltration (UF), in terms of flux and cell yields obtained with CFF of yeast cell suspensions. They found that the UF module had much lower fouling rate than with the pleated-sheet microfilter that had rapid plugging and significant cleaning issues. Bailey and Meagher [27] performed a similar comparison between the hollow fiber and plate and frame formats for microfiltration of recombinant *Escherichia coli* lysate and found both options to be comparable in performance under optimized conditions. Sheehan et al. [22] performed a comparison of the centrifugation vs. membrane-based separations of extracellular bacterial protease and found the membrane process to be twice as cost effective as the centrifuge and equivalent to a precoated filter, on the basis of unit cost of enzyme product recovered. Industrial studies demonstrating robust operation of tangential flow filtration (TFF) for harvest of mammalian cell culture [23] and CFF for harvest of recombinant yeast cell product [26] have also been reported. More fundamental studies investigating the various aspects of filtration processes such as membrane fouling, mathematical modeling, and critical flux determination have also been published [24,28,29].

Pilot-scale harvest of recombinant yeast employing microfiltration: a case study, Gregory Russotti[a], A.Edward Osawa[a], Robert D. Sitrin[a], Barry C. Buckland[a], William R. Adams[b], Steven S. Lee

Abstract

In order to develop a cost-effective recovery process for an intracellular product, crossflow microfiltration was studied for the harvest of a recombinant yeast under severe time constraint. It was required to process yeast broth in a short period of time to minimize the risk for product degradation. Preliminary microfiltration studies employing flat sheet membranes showed high throughout with initial fluxes on the order of water fluxes (> 1000 LMH, regime I, $2 min), followed by a rapid decay towards a low pseudo-steady state flux (20 LMH, regime II, > 2 min). Exploitation of these high fluxes and control of their eventual decline were crucial in establishing a rapid crossflow filtration process. The effect of several parameters, such as initial cell concentration, shear rate, transmembrane pressure, membrane pore size and medium composition on filtration performance were investigated to better understand the flux decline mechanisms. We found that the major contributor to flux decay was reversible fouling by the cake formation on the membrane surface. Within the operating boundaries of our microfiltration system, large-pore membrane (0.65 μm) was much more desirable for harvesting our yeast (10 μ size) without cell leakage than smaller pore ones (0.22 μm and 0.45 μm). Among adjustable operating parameters, feed flow rate (i.e., shear rate) exerted significant impact on average flux, whereas manipulation of transmembrane pressure afforded little improvement. Although initial cell concentration affected adversely the permeation rates, growth medium components, especially soy-peptone, was deemed pivotal in determining the characteristics of cell cake, thus controlling yeast microfiltration.

Keywords: Crossflow filtration; Microfiltration; *Saccharomyces cerevisiae*; Yeast; Recombinant

Biotechnology Techniques Vol 4 No 5 329-334 (1990)
Received 12th June

DISRUPTION OF *Saccharomyces cerevisiae* USING ENZYMATIC LYSIS COMBINED WITH HIGH-PRESSURE HOMOGENIZATION

Clark Baldwin and Campbell W. Robinson†
Department of Chemical Engineering, University of Waterloo,
Waterloo, Ontario, Canada N2L 3G1

ABSTRACT

The disruption of commercially-available pressed Bakers' yeast (*Saccharomyces cerevisiae*) was studied using a relatively new high-pressure homogenizer (the Microfluidizer). Initial experiments using only mechanical disruption generally gave low disruption yields (i.e., less than 40 % disruption in 5 passes). Consequently combinations of two disruption methods, namely enzymatic lysis and subsequent homogenization, were tested to identify achievable levels of disruption. The enzyme preparation employed was Zymolyase, which has been shown to effectively lyse the walls of viable yeast. Yeast cell suspensions ranging in concentration from 0.6 to 15 gDW/L were disrupted with and without enzymatic pre-treatment. Final total disruption obtained using the combined protocol approached 100 % with 4 passes at a pressure of 95 MPa, as compared to only 32 % disruption with 4 passes at 95 MPa using only homogenization. A model is presented to predict the fraction disrupted while employing this novel enzymatic pre-treatment.

INTRODUCTION

An abundance of valuable biochemicals can be produced by microorganisms. Some of these products can be made to be secreted by the microorganism; however in some cases this is not possible to achieve and the cells must be disrupted to release their contents. There are many different methods for cell disintegration. Some of these methods can be used in association with one another, in order to take advantage of their combined benefits [1].

The Microfluidizer, a high-pressure homogenizer, has recently been tested with native and recombinant strains of *E. coli* [2]. The cell suspension is forced through two parallel channels (2x100 µm) and the separate high-velocity streams are directed at each other in front of a wall.

Preliminary disruption experiments with *Saccharomyces cerevisiae* using Microfluidization alone indicated the need for an improved disruption methodology. A combination of methods was suggested and an enzymatic pre-treatment using Zymolyase to "soften up" the cells was employed.

Zymolyase-20T is commercially available from Seikagaku Kogyo Co., Ltd. It is produced by a submerged culture of *Oerskovia xanthineolytica* (previously classified as *Arthrobacter luteus*). An essential enzyme responsible for lysis of viable yeast cells in this preparation is β-1,3-glucan laminaripentaohydrolase [3-6].

Using Zymolyase as a pre-treatment to Microfluidization, a study of various homogenization operating conditions (i.e., disruption pressure and number of passes) has been undertaken in order to determine the acceptability of using the Microfluidizer as a method for mechanical disruption of yeast. An empirical model is presented to account for the introduction of this novel enzymatic pre-treatment regime.

MATERIALS AND METHODS

ENZYME EMPLOYED FOR PRE-TREATMENT

Zymolyase-20T, produced by a submerged culture of *Oerskovia xanthineolytica*, is a relatively new enzyme preparation which effectively lyses the walls of viable yeast cells [7,8]. Zymolyase-20T has 20,000 units/g of lytic activity, defined below, toward Brewers' yeast (*Saccharomyces uvarum*, resting stage).

One unit of lytic activity is defined as that amount which results in a 30 % decrease in absorbance at 800 nm (A 800) of the reaction mixture under the following conditions:

Reaction Mixture	
Enzyme.....0.1 mg/mL solution	1 mL
Substrate.....Yeast Cell Suspension (2 mgDW/mL)	3 mL
Buffer.....M/15 Phosphate Buffer pH 7.5	5 mL
Deionized water	1 mL
Total volume	10 mL

After incubation for 2 h at 25 °C with gentle shaking, A 800 of the mixture is determined. As a reference, 1 mL of deionized water is used instead of the enzyme solution.

$$\% \text{ decrease in A 800} = \frac{(\text{A 800 ref.} - \text{A 800 react. mixture})}{\text{initial A 800 ref.}} \times 100\%$$

When an A 800 decrease of 60 %, equivalent to 2 units of activity, is observed in the reaction system, the yeast cells are completely lysed, namely 1 unit of Zymolyase-20T lyses 3 mg dry weight of Brewers' yeast [9].

DISRUPTION EQUIPMENT

The disruption device used in this work was a Microfluidizer high-pressure homogenizer (model M110T with extra heavy duty pump; Microfluidics Corp., Newton, MA., U.S.). The disruptor consisted of an air-driven, high-pressure pump (ratio 1:250; required air pressure 0.6-1 MPa) and a special disruption chamber with an additional back pressure unit. A minimum sample size of 20 mL is required for processing. Further details are given in [2].

STRAINS AND ANALYTICAL

Commercially-available pressed Bakers' yeast (*Saccharomyces cerevisiae*) was resuspended in deionized water to a cell concentration of 2 mgDW/mL. The resulting reaction mixture thus has a yeast cell concentration of 0.6 gDW/L. The above enzymatic pre-treatment (prior to homogenization) was carried out for 2 h, with the percent decrease in A 800 found at the end of the 2 h incubation (the course of the enzymatic lysis was followed by sampling the reaction and reference mixtures every 15 min.).

The buffered and partially lysed yeast suspension after enzymatic pre-treatment was cooled to 4 °C and was subsequently homogenized at various operating pressures (30-95 MPa) and at up to five passes through the homogenizer. The resulting homogenized yeast suspensions were cooled using ice packs, as was the disruption chamber. The Microfluidizer

2.1.1 ***How to Use Avestin Emulsiflex C3 Homogenizer to Disrupt Cells***

Cell Biology > Cell viability > Cell lysis

Fungi > Saccharomyces > Saccharomyces cerevisiae > Other compound

Author: Zongtian Tong
1/5/2011, 7257 views, 1 Q&A

[Abstract] The EmulsiFlex-C3 homogenizer is powered by an electric motor. The pump does not require a compressor for it to run. This equipment can be used to disrupt cells at a large scale. The EmulsiFlex-C3 has a fixed flow-through capacity of 3L/h. It has the ability to process samples as small as 10 ml. The homogenizing pressure is adjustable between 500 and 30,000 psi. In this protocol, we describe the use of the Avestin Emulsiflex C3 Homogenizer to disrupt *S. pombe* and *S. cerevisiae* cells.

Equipment

Figure 1. Avestin Emulsiflex C3 homogenizer

Procedure

1. Switch on the homogenizer at the back.
2. Turn on nitrogen. Pressure reads 80 psi.
3. Unscrew the funnel cap. Check if the funnel cap is on to make sure ethanol does not evaporate.
4. Turn red stop knob close wise and push green knob to start.
5. Pump residual ethanol out of the tubing.
6. Pour DI water into the funnel to wash ethanol out. Keep air pressure on occasionally to make sure no cell debris is left from the last user.
7. Before load your samples, take the funnel off and roll it on ice to keep it cool. Install the funnel back to the top. Put the steel coil heat exchanger into ice to cool down the samples.
8. Load your samples into the funnel. Turn on the homogenizer. Let the samples run through the tubing back to the funnel before air pressure is on.
9. Turn on air pressure. Air pressure at 40 psi, gauge pressure ≥ 20,000 psi and < 25,000 psi. The maximum pressure is 30,000 psi. Leave the tubing in a sample collection tube chilled on ice.
10. *S. Pombe* samples need to be passed through 5~6 times to reach 80~90% efficiency. *S. cerevisiae* samples need to be passed through 8~9 times to reach 80~90% lysis efficiency. Check samples under a microscope.
11. If the homogenizer is clogged by the samples, cap the funnel and blow with nitrogen tube.
12. After samples are done, take off the funnel and rinse it with DI water.
13. Run more water to flush out cell debris. Keep the air pressure on occasionally.

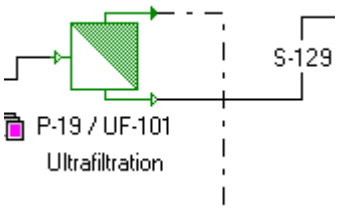
14. Run ethanol and leave 1/3 of a funnel volume of ethanol in the funnel.

References

1. http://www.avestin.com/English/c3page.html

How to cite this protocol: (2011). How to Use Avestin Emulsiflex C3 Homogenizer to Disrupt Cells. *Bio-protocol* Bio101: e11. http://www.bio-protocol.org/e11

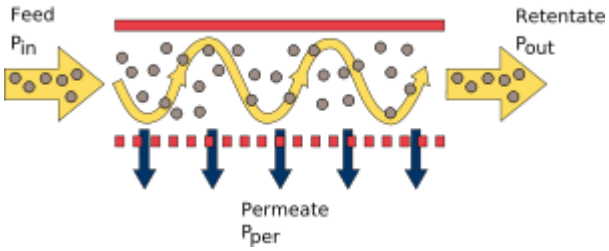
2.2 Crossflow filtration



The whey proteins are concentrated using crossflow ultrafilters (UF-101).

From Wikipedia, the free encyclopedia

Jump to: navigation, search



In chemical engineering, biochemical engineering and protein purification, crossflow filtration[1] (also known as tangential flow filtration[2]) is a type of filtration (a particular unit operation). Crossflow filtration is different from dead-end filtration in which the feed is passed through a membrane or bed, the solids being trapped in the filter and the filtrate being released at the other end. Cross-flow filtration gets its name because the majority of the feed flow travels tangentially across the surface of the filter, rather than into the filter.[1] The principal advantage of this is that the filter cake (which can blind the filter) is substantially washed away during the filtration process, increasing the length of time that a filter unit can be operational. It can be a continuous process, unlike batch-wise dead-end filtration.

Besonders dafür geeignet sind Hohlfasern (Kapillarmembran oder auch Hohlfäden genannt), deren Leistungsfähigkeit noch durch den sogenannten Pinch-Effekt verstärkt wird. Eine übliche Hohlfaser hat einen Innendurchmesser von circa 1,5 mm (3,0 mm bis 0,1 µm möglich) und eine Porengröße von 200 bis 5 nm (2 µm bis 1,0 nm möglich). Je nach Anwendung werden hunderte bis tausende Kapillaren in Modulen zusammengefasst und vergossen (Hohlfasermodule). Mit Hilfe einer Zirkulationspumpe wird das unfiltrierte Produkt solange durch die Kapillaren zirkuliert, bis die Trubstoffe im Retentat so konzentriert sind, dass eine Entleerung und Reinigung erforderlich wird.

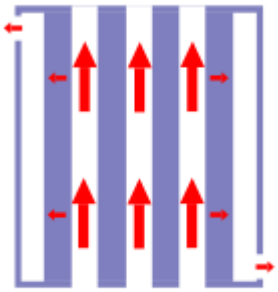


Diagram of cross-flow filtration

This type of filtration is typically selected for feeds containing a high proportion of small particle size solids (where the permeate is of most value) because solid material can quickly block (blind) the filter surface with dead-end filtration. Industrial examples of this include the extraction of soluble antibiotics from fermentation liquors.

Filtration unit for industrial cross-flow filtration

Charakteristische Merkmale der Cross-Flow-Filtrationstechnologie sind die weitgehende Eliminierung von Filterhilfsmitteln, d. h. deren Beschaffung, Lagerung, Handhabung und Entsorgung, die rasche, arbeitsextensive sowie die qualitätsschonende Verarbeitung.

Die Cross-Flow-Filtration ist sehr energieaufwändig. Ein großer Teil der in die Förderung des Feeds investierten Energie geht durch das Retentat verloren. Deshalb wird an den Stellen, wo darauf verzichtet werden kann, immer stärker auf die Dead-End-Filtration zurückgegriffen.

2.3 Dead-End Filtration

To be described

2.4 Anion exchange chromatography[1]

Anion-exchange chromatography is a process that separates substances based on their charges using an ion-exchange resin containing positively charged groups, such as diethyl-aminoethyl groups (DEAE).[2] In solution, the resin is coated with positively charged counter-ions (cations). Anion exchange resins will bind to negatively charged molecules, displacing the counter-ion. Anion exchange chromatography is commonly used to purify proteins, amino acids, sugars/carbohydrates and other acidic substances [3] with a negative charge at higher pH levels. The tightness of the binding between the substance and the resin is based on the strength of the negative charge of the substance.

General technique for protein purification

A slurry of resin, such as DEAE-Sephadex is poured into the column. After settling, the column is pre-equilibrated in buffer before the protein mixture is applied. Unbound proteins are collected in the flow-through and/or in subsequent buffer washes. Proteins that bind to the resin are retained and can be eluted one of two ways. First, the salt concentration in the elution buffer is gradually increased. The negative ions in the salt solution (e.g. Cl-) compete with protein in binding to the resin. Second, the pH of the solution can be gradually decreased which results in a more positive charge on the protein, releasing it from the resin. As buffer elutes from the column, the samples are collected using a fraction collector.

2.5 Final product formulation[2]

High-resolution chromatography normally yields a protein that is 98-99 per cent pure.

Final product formulation bings the product in the final format. This envolves:

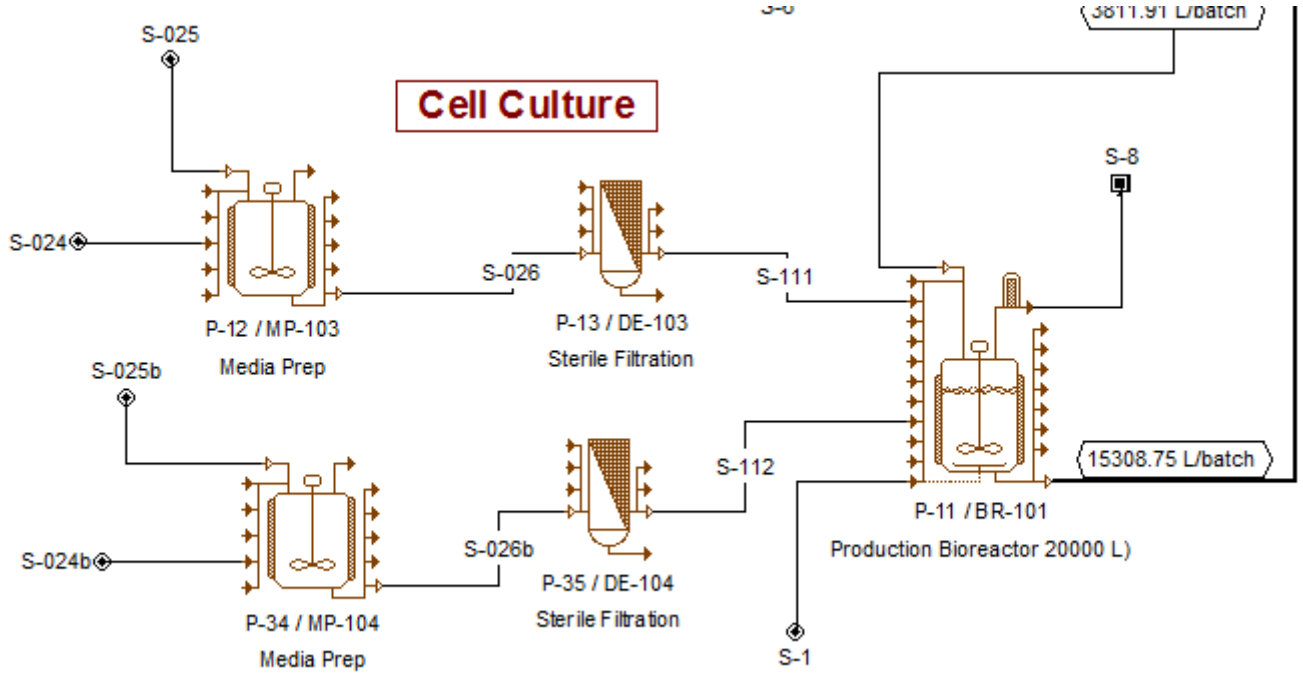
- Addition of various excipients
- Filtration of the final product through a 0.22 μm absoulute filter, then aseptic filling into final containers.
- Freeze-drying if the product is to be marketed in a powedered format.

Fig.: 0.22 μm filter

[1] http://en.wikipedia.org/wiki/Anion-exchange_chromatography

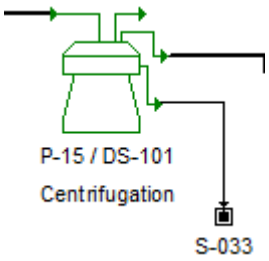
[2] [Walsh 2007], ch.6.9

2.6 **Bioreactor/Tank**

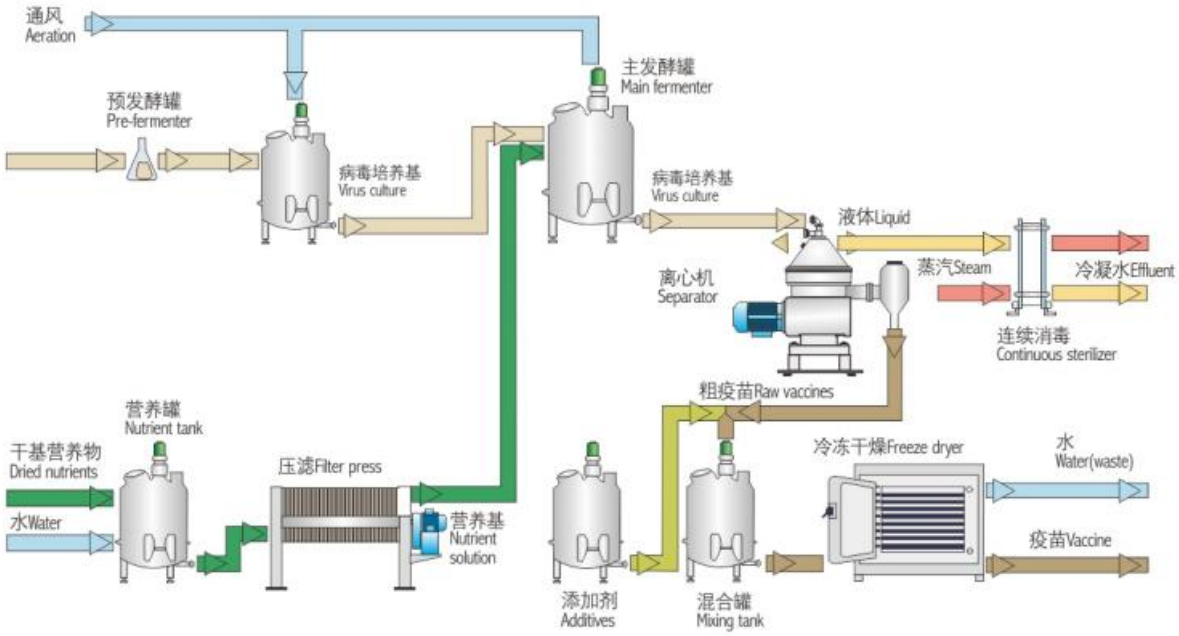


S.cerevisiae needs as nutrition only sugar.

2.7 **Disc-Stack Centrifuge**



2.7.1 *Example for Supplier: Huanding China*



型号 Model	处理量 Capacity(L/h)	重量 Weight(Kg)	外型尺寸(长*宽*高) Dimension(L*W*H)
BTSX15	400~1000	1100	950×950×1250
BTSX35	1000~3000	1500	1400×1400×1450
BTSX85	3000~70000	1800	1750×1450×1850
BTSX150	14000-30000	3700	1850×1500×1850
BTSX200	20000-40000	4800	2100×1700×2200

2014-12-26 13:57 GMT+02:00 Chouse Hu <chousehu@huading-separator.com>:
Dear Samir Mourad

do you think $250,000 dollar in your budget ?

Best Regards

Chouse Hu

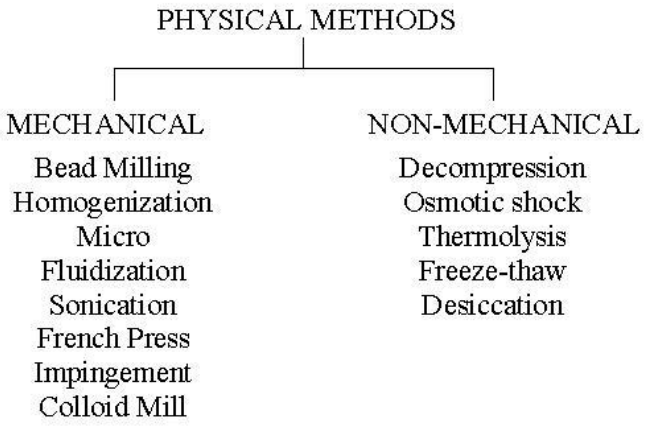
On Dec 27, 2014 8:50 AM, "Chouse Hu" <chousehu@huading-separator.com> wrote:
our cheapest centrifuge can be used in this case at all, that means nothing for you.

please refer to Westalia Separator, and learn what a particular centrifuge can meet your demand.

then, you could try to find out an alternative from China, to reduce your cost.

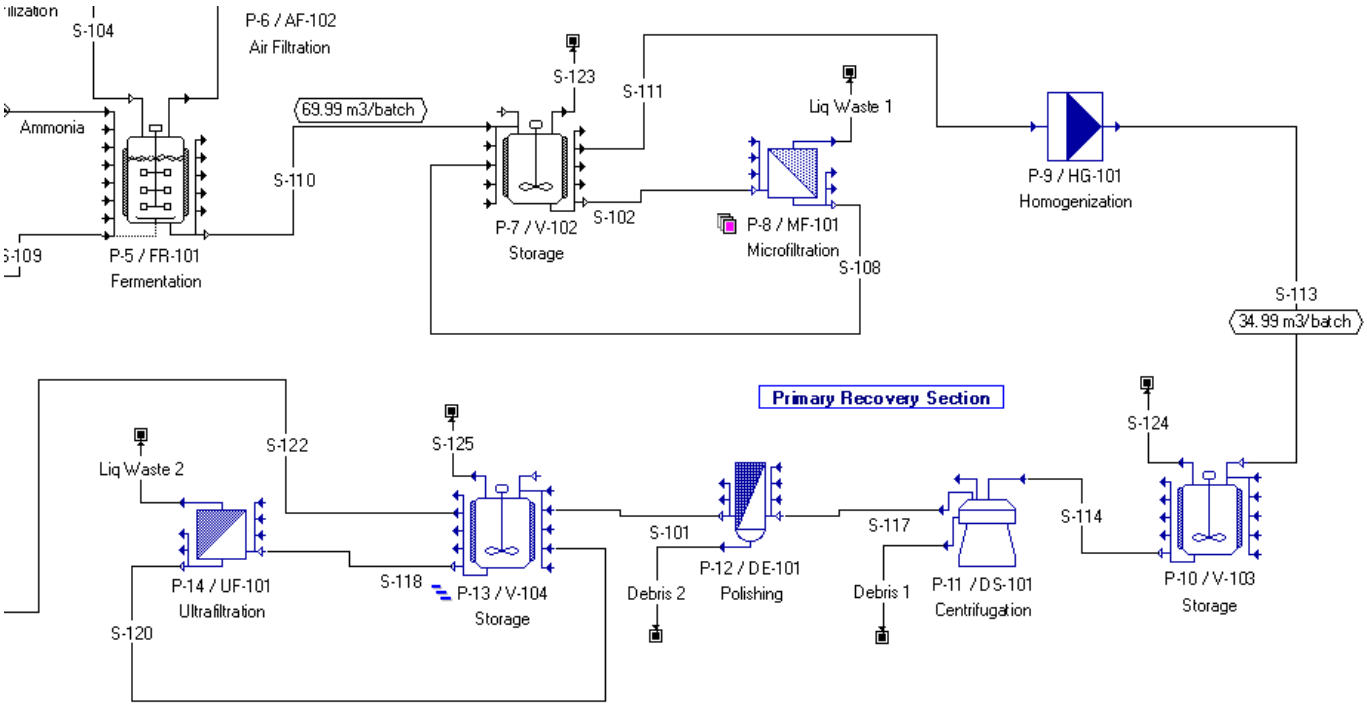
http://www.westfalia-separator.com/no_cache/contact/locations-worldwide/view.html

2.8 Homogenization



Method of cell lysis	disadvantages
freeze-thaw	very slow
chemical lysis	can cause changes in protein structure, difficulties in purification, expensive detergents
enzymatic lysis	often not reproducible, enzyme stability, long incubation time, necessity of removing the lysis enzymes, expensive scale-up, often combination with other method necessary
cell bomb	only applicable to specific cell types
high pressure homogenizer (e.g. french press)	expensive equipment, high maintenance due prone of orifices' clogging
centrifugation	only for very weak cell walls
ball mill / bead mill	uneven processing = incomplete lysis, protein denaturation, low efficiency whilst relatively high energy consumption, complex separation of milling medium and product, time-consuming cleaning

2.8.1 *Principle*



2.8.2 ***Suppliers***

2.8.2.1 Ningbo Scientz company

High Pressure Homogenizer JG-IA Series

Acting on high-pressure principle, the instrument is used for extruding cells, especially suitable for smashing thick-wall cells, germs and denser solution samples. Having no noise, less temperature rise and no metal ion contamination, it has wide application of such research fields as protein study, nucleic acid extraction, cell disruption in genetic engineering labs of colleges, scientific research institutes and pharmaceutical factories.

● Specifications:

Model	JG－IA	Capacity	50ml/times Continuous increase samples for scientific research
Voltage	380 VAC	Maximum operating pressure	256 Mpa (37120 psi)
Pressure device	Hydraulic System	Max Pressure stroke	170 mm
Sample tube	Φ25mm Stainless	Pressure plate speed	6.8 mm/s
Dimensions	555×600×1170 mm	Largest volume of sample	50 ml/times

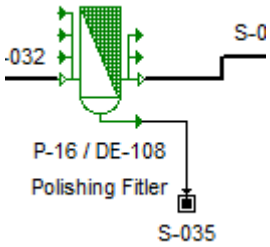
2.8.2.2 Zhangjiagang Beyond Machinery Co., Ltd.

Model Number: GJB, Shipping Terms: FOB Port: Shanghai

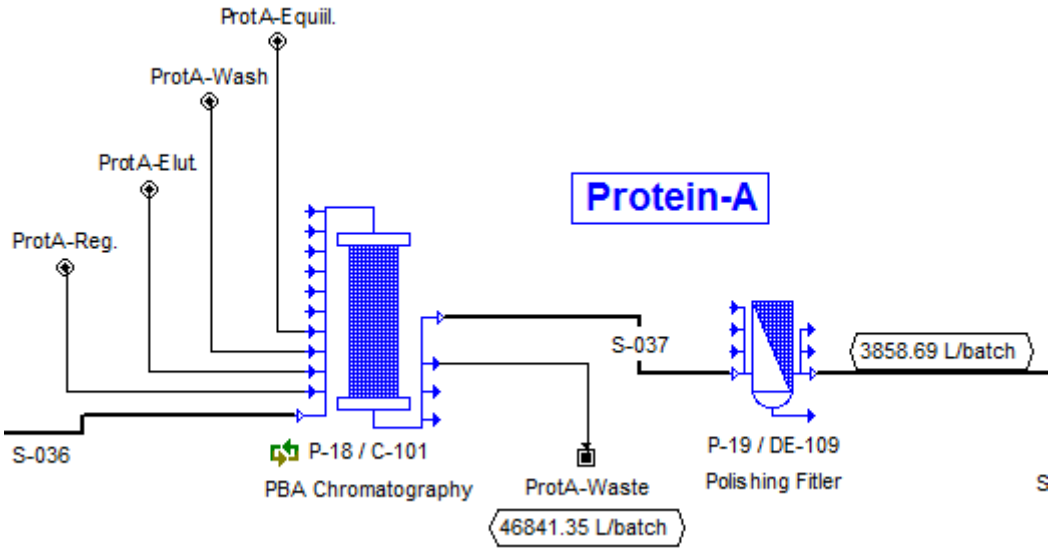
Unit Price: USD 10,000/Sets quantity: 1 Sets

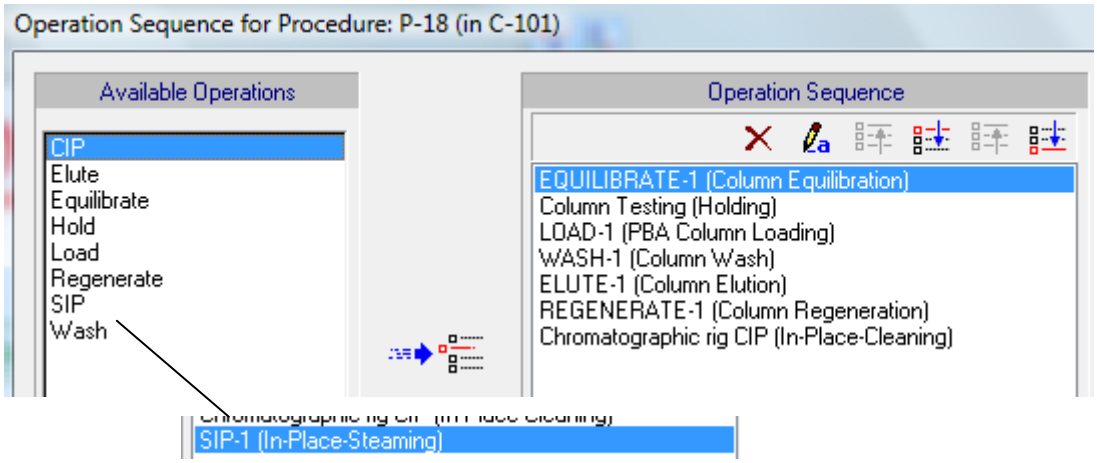
Lead Time: 30days Payment Terms: T/T Quotation Valid Till: 2015-1-28

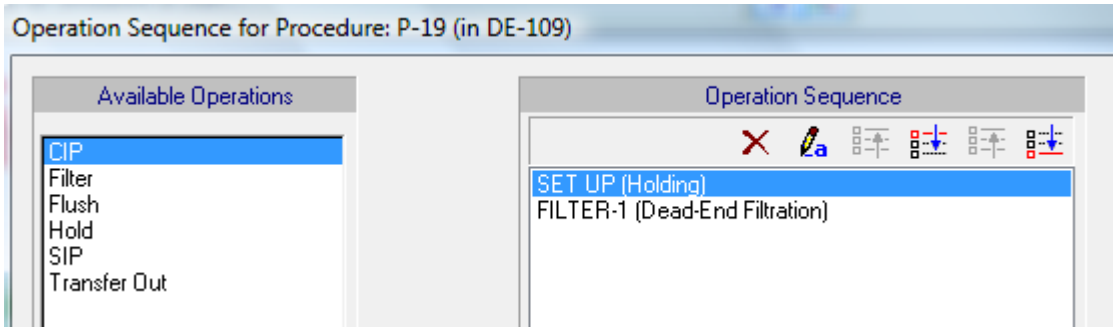
2.9 Dead-End Filter



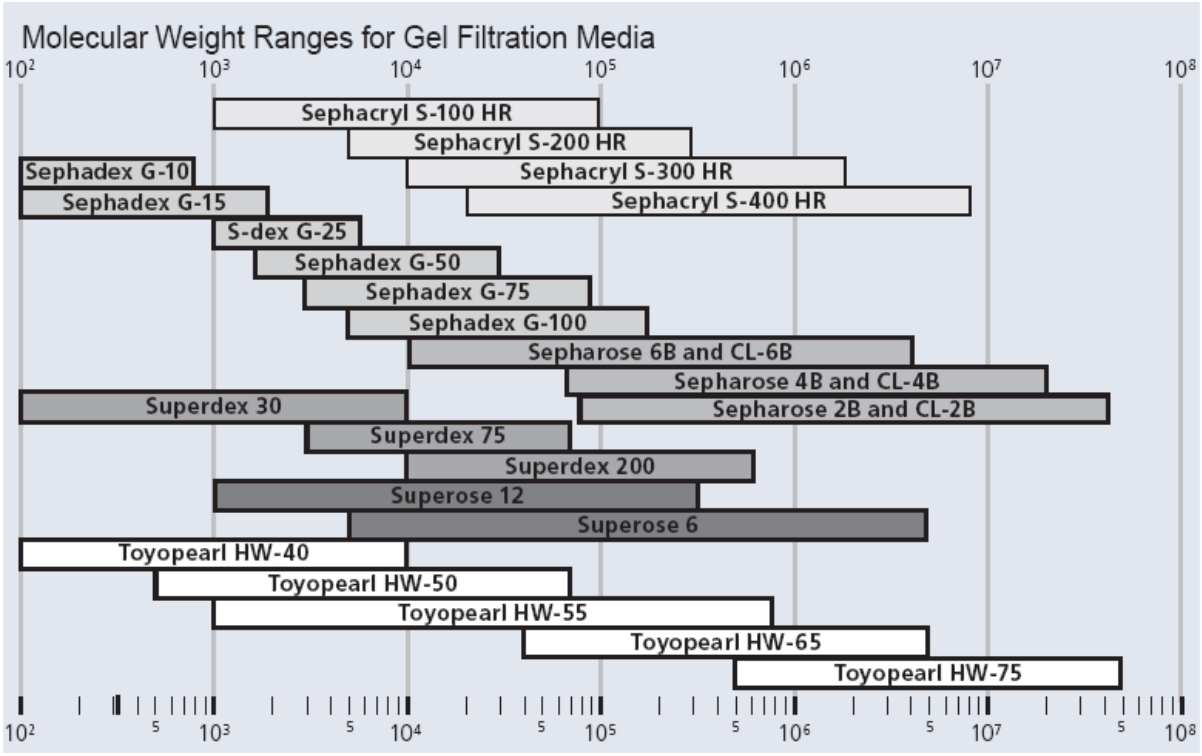
2.10**Protein-A Chromatography**



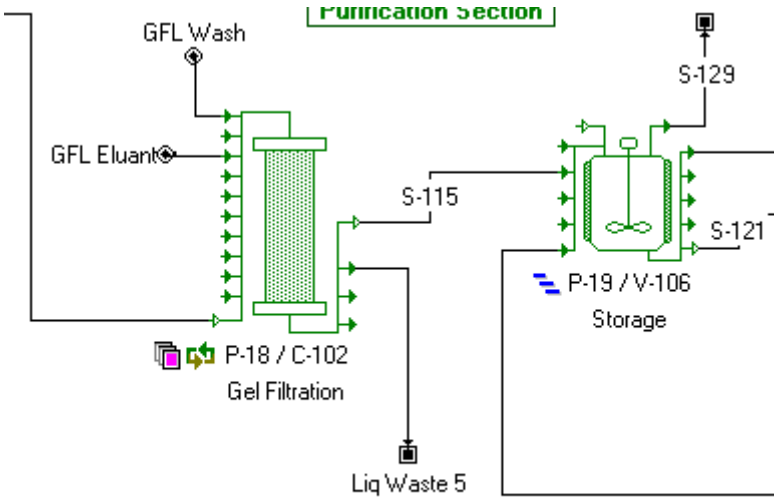




2.11Gel filtration chromatography



4B Sepharose



2.11.1 *sensor/actuator list*

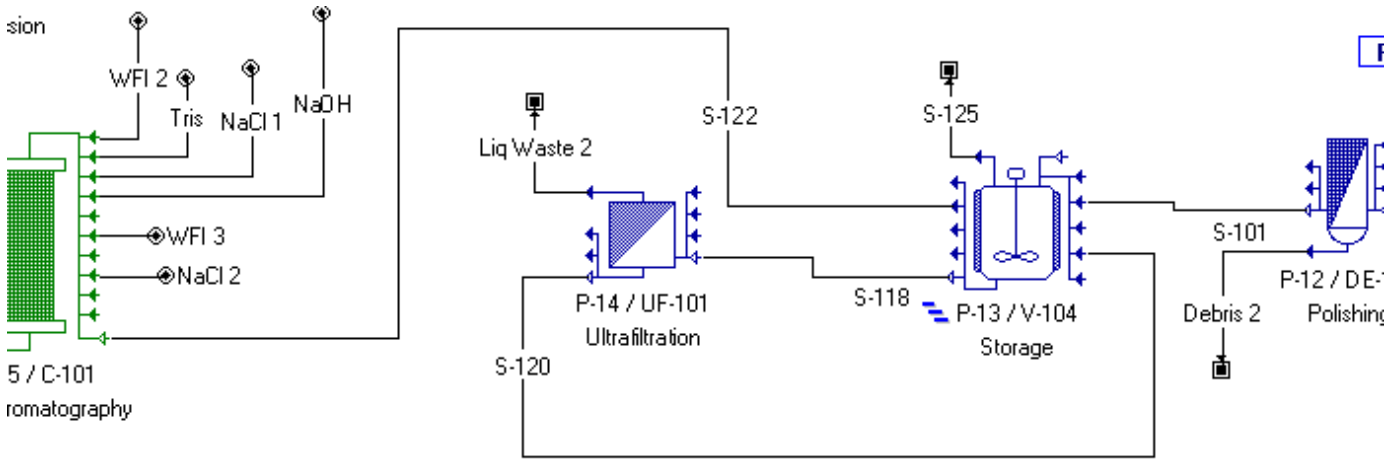
similiar to AKTA process

Akta process Sensors and actuators 13.12.13

Teil	**Anzahl**	Item Price
Air trap	1	
Filter	1	
Filter vent valve	1	
Capsule filter bottom manual valve	1	
Capsule filter top manual valve	1	
System pump	2	
Sample pump	1	

Pressure control valve	2
Buffer A inlet valves	10
Buffer B inlet valves	6
Sample connection valve	1
Sample inlets valves	2
Air trap inlet valve	1
Air trap bypass valve	1
Air trap vent valve	1
Air trap outlet valve	1
Filter inlet valve	1
Filter bypass valve	1
Filter outlet valve	1
System connection valve	1
Column 1 top inlet valve	1
Column 1 bottom inlet valve	1
Column 1 top valve	1
Column 1 bottom valve	1
Column 1 top outlet valve	1
Column 1 bottom outlet valve	1
Column 2 top inlet valve	1
Column 2 bottom inlet valve	1
Column 2 top valve	1
Column 2 bottom valve	1
Column 2 top outlet valve	1
Column 2 bottom outlet valve	1
Outlet valves	9
Air trap drain valve	1
Filter drain valve	1
CIP / AxiChrom manifold	1
Buffer inlet air sensor	1
Pre-column air sensor	1
Post-column pH-meter	1
Post-column UV-meter	1
Pre-column conductivity meter	1
Post-column conductivity	1
System flow meter	1
Air trap high level meter	1
Air trap low level meter	1
Pre-filter pressure meter	1
Pre-column pressure meter	1
Sample pump pressure meter	1
PCV pressure meter, A inlets	1
PCV pressure meter, B inlets	1

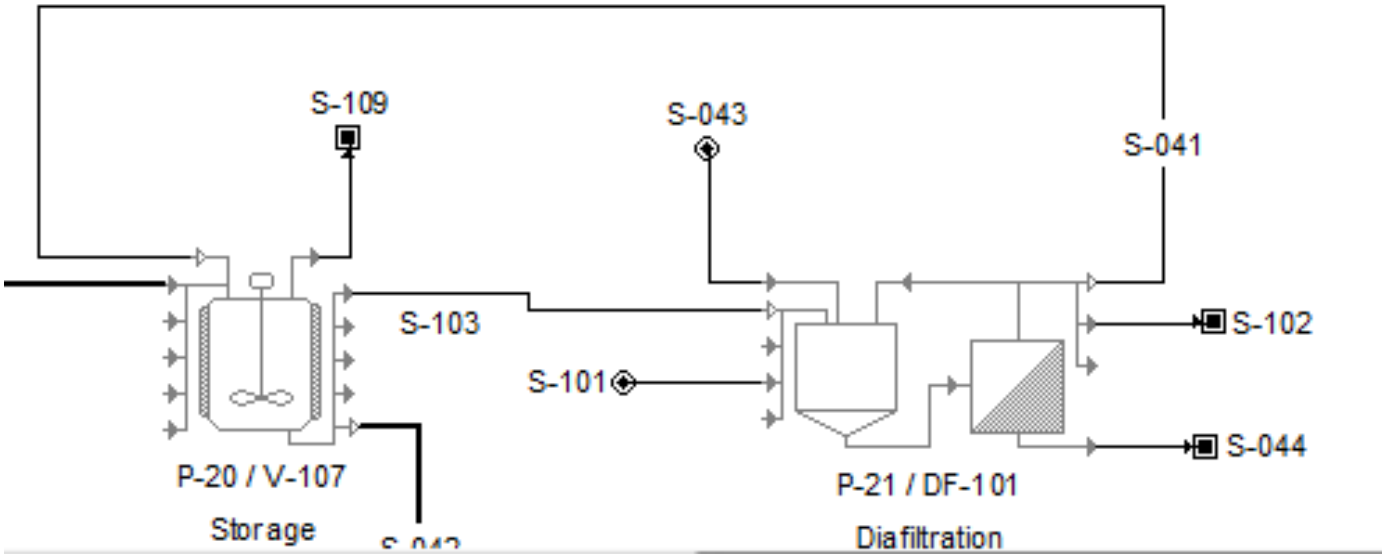
2.12**Ultrafiltration**

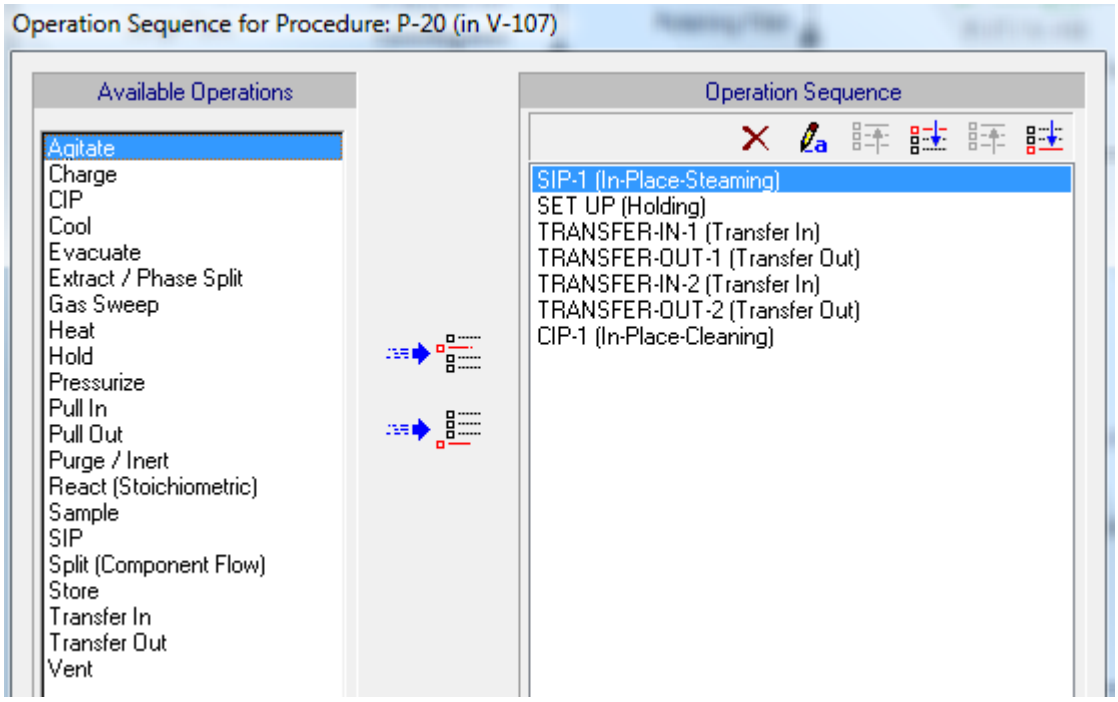


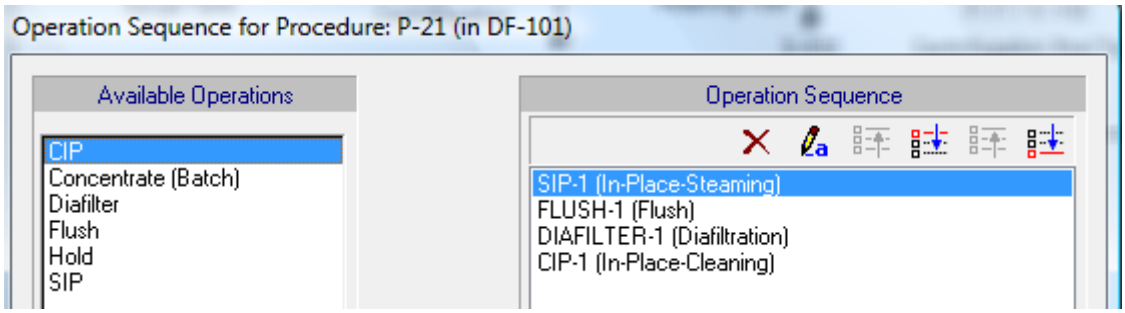
From SuperPro Example Bgal

2.13**Diafiltration**

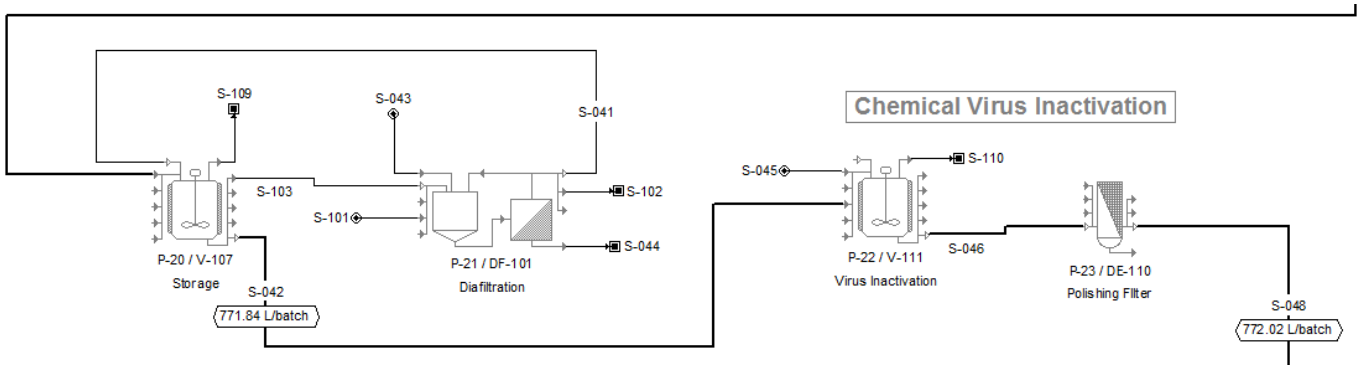
Same equipment as ultrafiltration

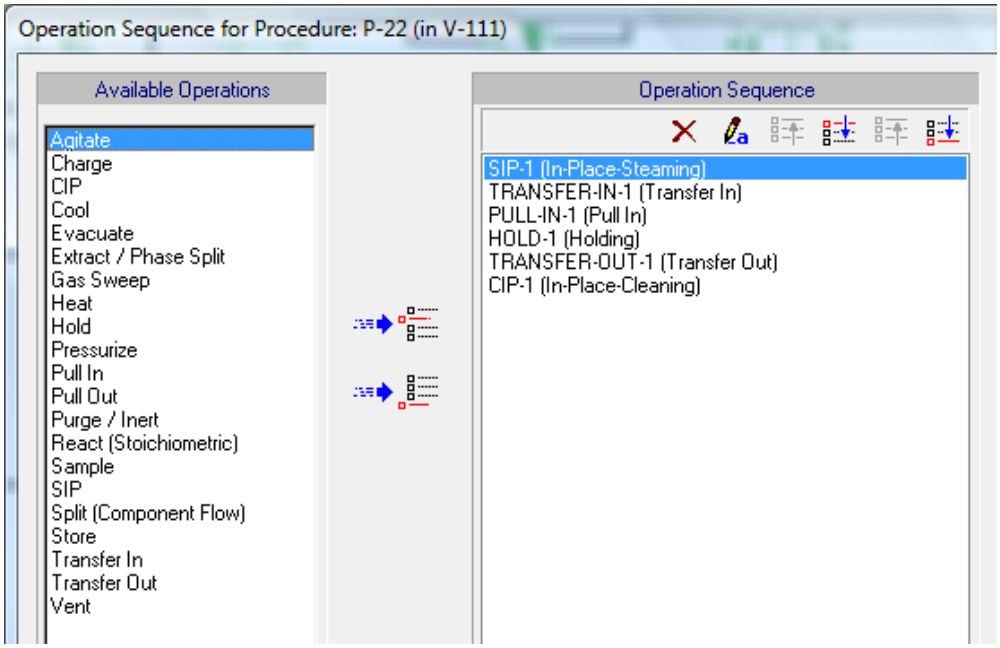






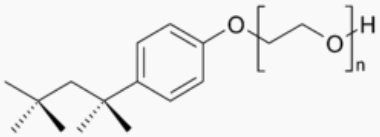
2.14 Chemical Virus Inactivation





2.14.1 ***Solvent/detergent (S/D) inactivation***[3]

This process, developed by the New York Blood Center,[5] is the most widely used viral inactivation method to date. It is predominantly used in the blood plasma industry, by over 50 organizations worldwide and by the American Red Cross [1]. This process is only effective for viruses enveloped in a lipid coat, however. The detergents used in this method interrupt the interactions between the molecules in the virus's lipid coating. Most enveloped viruses cannot exist without their lipid coating so are destroyed when exposed to these detergents. Other viruses may not be destroyed but they are unable to reproduce rendering them non-infective. The solvent creates an environment in which the aggregation reaction between the lipid coat and the detergent happen more rapidly. The detergent typically used is Triton-X 100.



Chemical Structure of Triton X-100 (n = 9-10).

This process has many of the advantages of the "traditional" removal techniques. This process does not denature proteins, because the detergents only affect lipids and lipid derivatives. There is a 100% viral death achieved by this process and the equipment is relatively simple and easy to use. Equipment designed to purify post-virus inactivated material would be necessary to guard against contamination of subsequent process streams.

S/D treatment utilizes readily available and relatively inexpensive reagents, but these reagents must be removed from the product prior to distribution which would require extra process steps. Because this process removes/inactivates the lipid coating of a virus, viruses without any sort of lipid envelope will be unaffected. There is also no inactivation effect by the buffers used in this process.

---

3 http://en.wikipedia.org/wiki/Virus_processing#Viral_inactivation

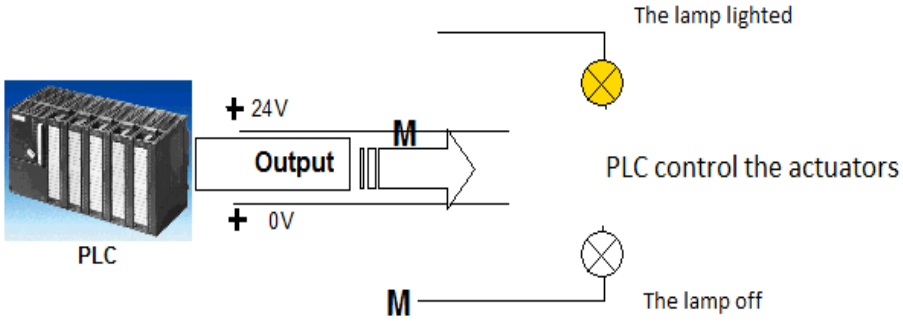
2.15Introduction to Siemens S7 PLC system

2.15.1 *Definition of PLC:*

PLC is an acronym for Programmable Logic Control (logic controllers programmable). This description of the system that controls the process (exp machine to print newspapers, facility for packing cement, piston to cut plastic ...). This process is carried out in accordance with the instructions of the program in the device memory.

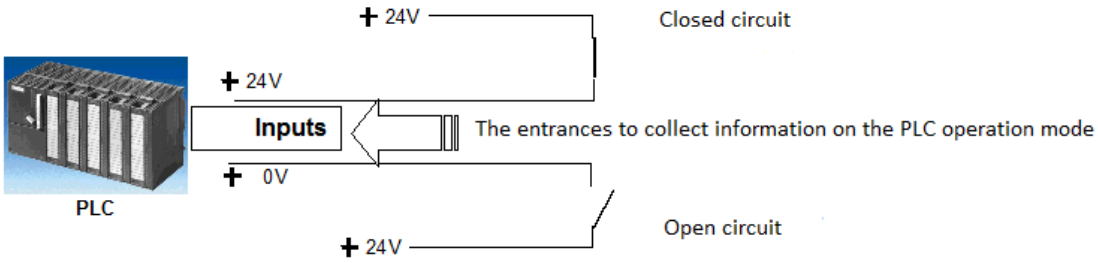
2.15.2 *Function of PLC:*

PLC controls the process in which the actuators are connecting to links feed (for example, 24 volts) specific to the PLC exits Outputs. Through these links we can run and turn off the engine, open and close the valves, or turn on and turn off the lights.

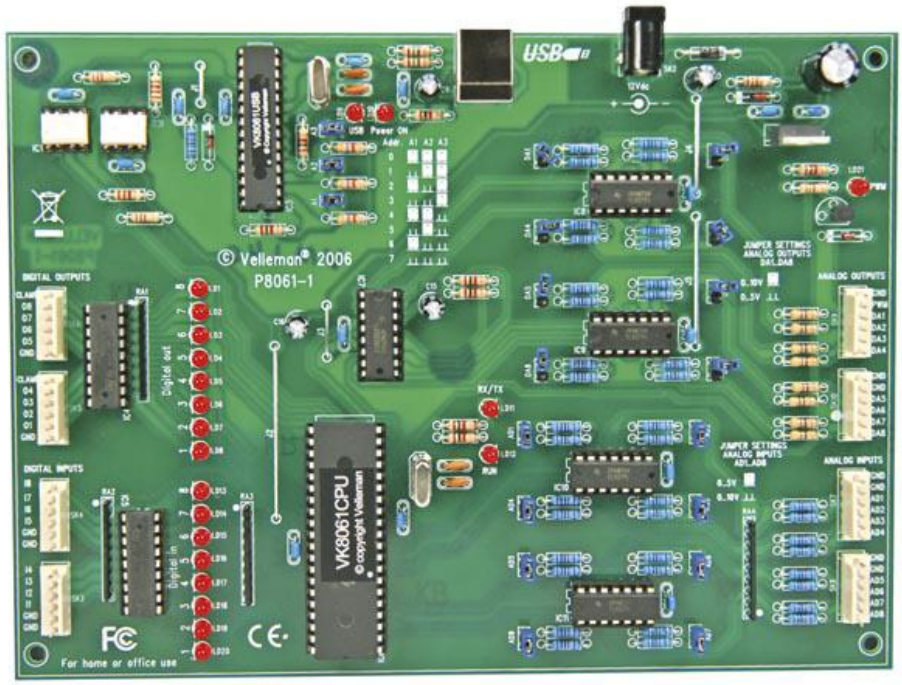


2.15.3 *Where to get the information about the PLC operation mode:*

PLC receives the information about the operation from the signal - generator connected to the entrances of the PLC. These signal generators can be, for example, sensors recognize the status of the working parts, keys or buttons. This specific situation can be open or closed. The difference between Note normally open NC: Normally Closed is ineffective when they are closed and petitions NO: Normally Open Normally open that are not effective when they are open:



2.16 Introduction to the DLL for the USB Interface Board K8061:

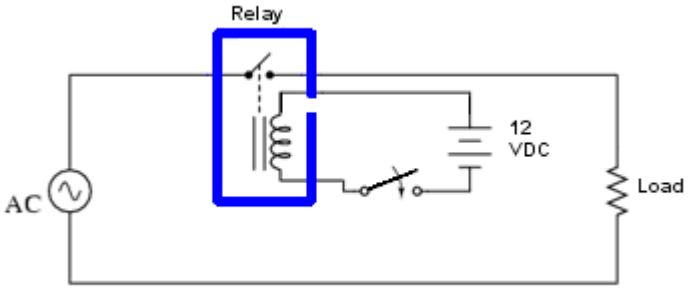


The K8061 interface board has 8 digital input channels and 8 digital output channels. In addition, there are 8 analogue inputs, 8 analogue outputs and one PWM output. The number of inputs/outputs can be further expanded by connecting more (up to a maximum of eight) cards to the PC's USB connectors. Each card is given its own identification number by means of three jumpers, A1, A2 and A3 (see table 1 below for card numbering).

Connection to the computer is optically isolated, so that damage to the computer from the card is not possible.

2.17**Definition of a relay**

A relay is usually an electromechanical device that is actuated by an electrical current. The current flowing in one circuit causes the opening or closing of another circuit. Relays are like remote control switcher and are used in many applications because of their relative simplicity, long life, and proven high reliability. Relays are used in a wide variety of applications throughout industry, such as in telephone exchanges, digital computers and automation systems. Highly sophisticated relays are utilized to protect electric power systems against trouble and power blackouts as well as to regulate and control the generation and distribution of power.



How do relays work?

All relays contain a sensing unit, the electric coil, which is powered by AC or DC current. When the applied current or voltage exceeds a threshold value, the coil activates the armature, which operates either to close the open contacts or to open the closed contacts. When a power is supplied to the coil, it generates a magnetic force that actuates the switch mechanism. The magnetic force is, in effect, relaying the action from one circuit to another. The first circuit is called the control circuit; the second is called the load circuit.
There are three basic functions of a relay: On/Off Control, Limit Control and Logic Operation.

3. Concept

3.1 Mechanical structure

The concept is to install a simplified DNA vaccine production line based on devices of GE Health and other suppliers.

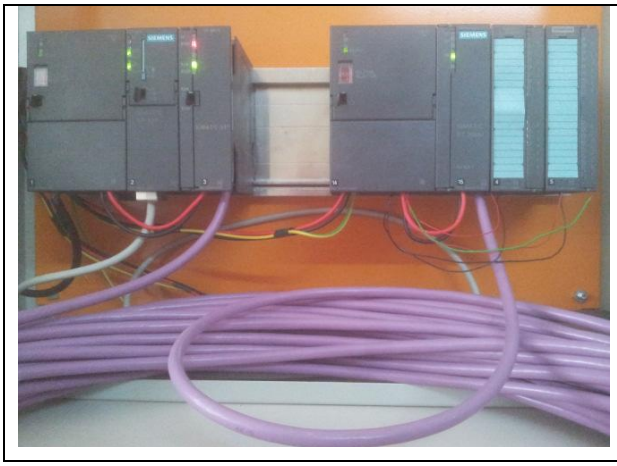
(As increment: simplified monoclonal antibodies production line using the same equipment)

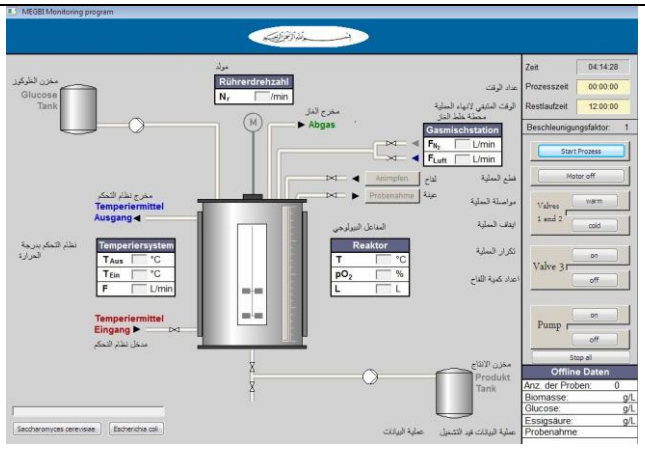
The stainless structure shall be built by TEMO. The instrumentation and special pipes etc. shall be buyed for GE Health or others.

Fig.: Solaris downstream line. The MEGBI-VPP downstream line shall be similar to this.

3.2 Automation System

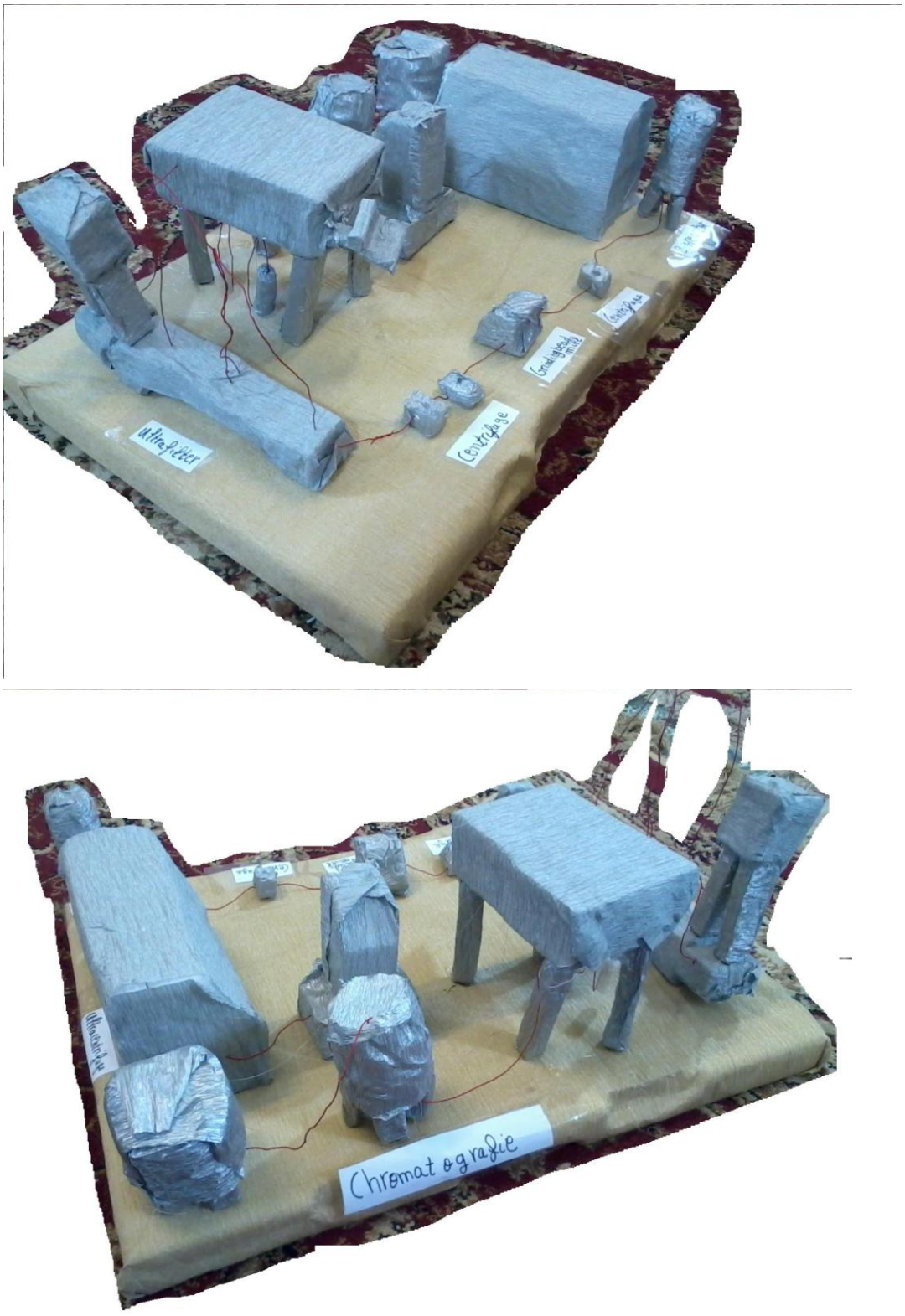
The automation system shall have a C++/phython user interface and a Simatic S7 interface to the sensors/actuators.





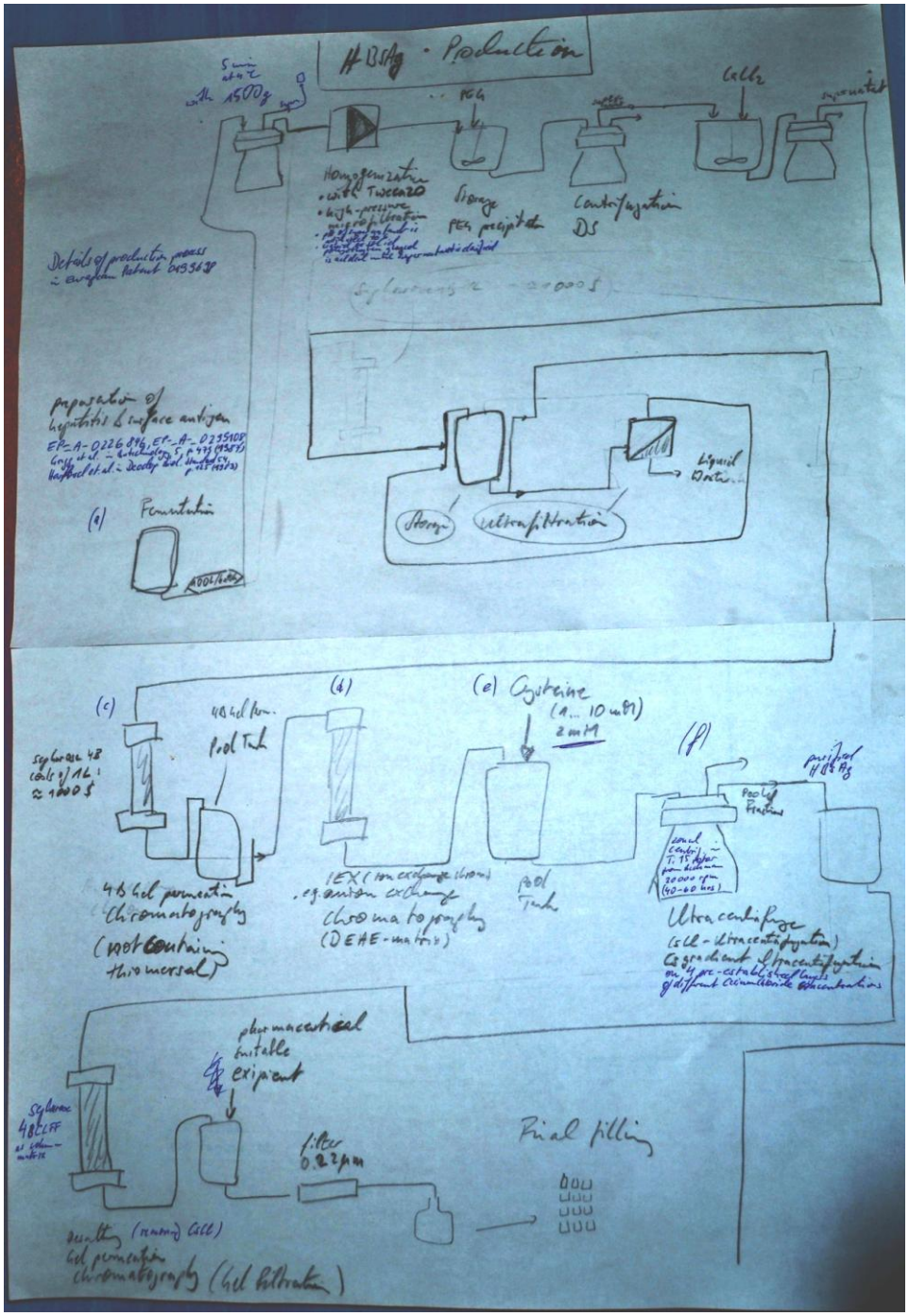
4. System Design

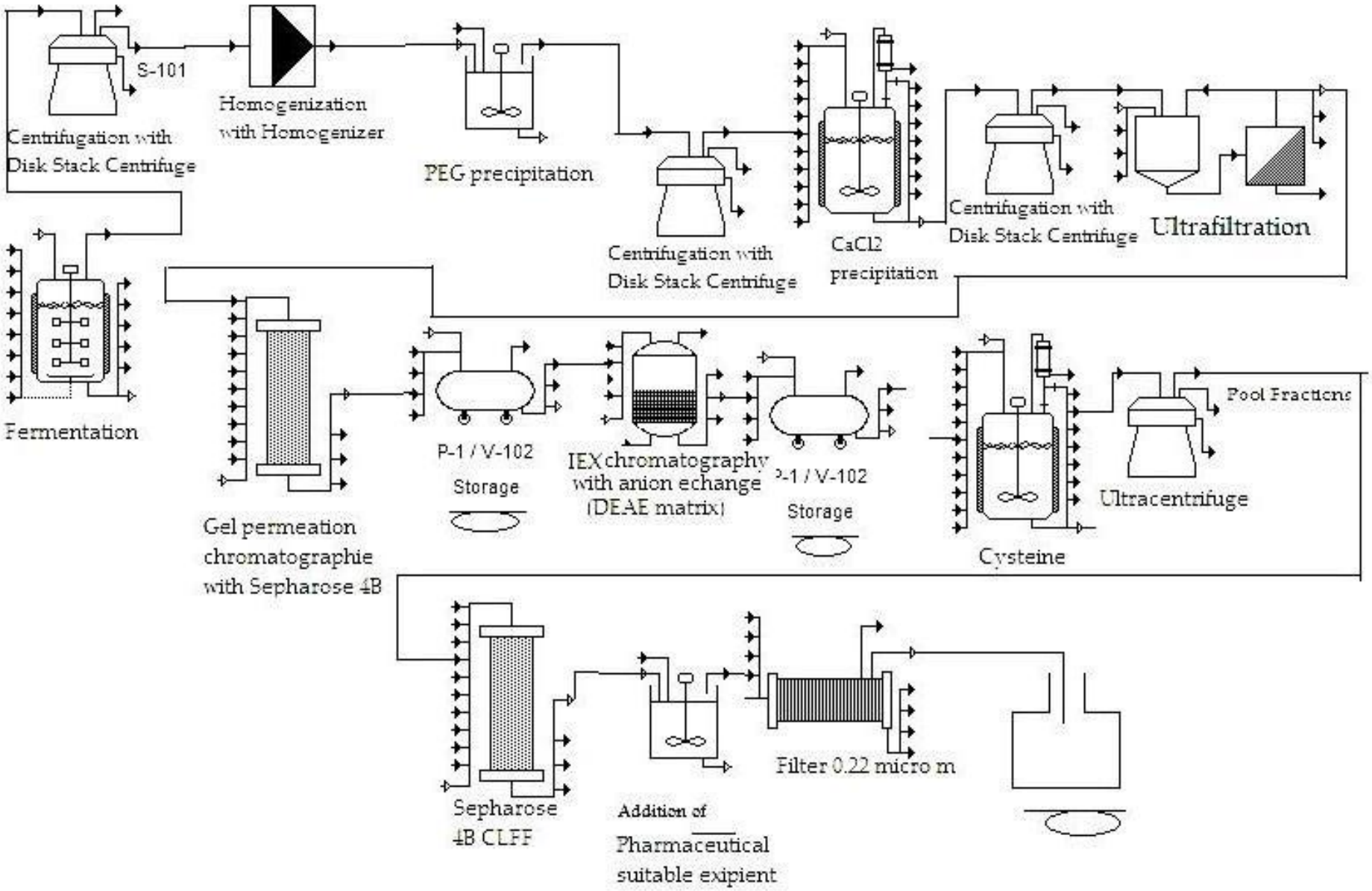
4.1 Mock-up model for the MEBGI vaccine pilot plant



4.2 Integration Overview

Based on the EngerixB patent and the films Bioprocessing Part 2_ Separation _ Recovery and Bioprocessing Part 3_Purification which describe in details the process of Fluorecence Protein Production in E.coli there shall be designed a simplified system for HBSAg vaccine. Using SuperPro SW for describing the design.

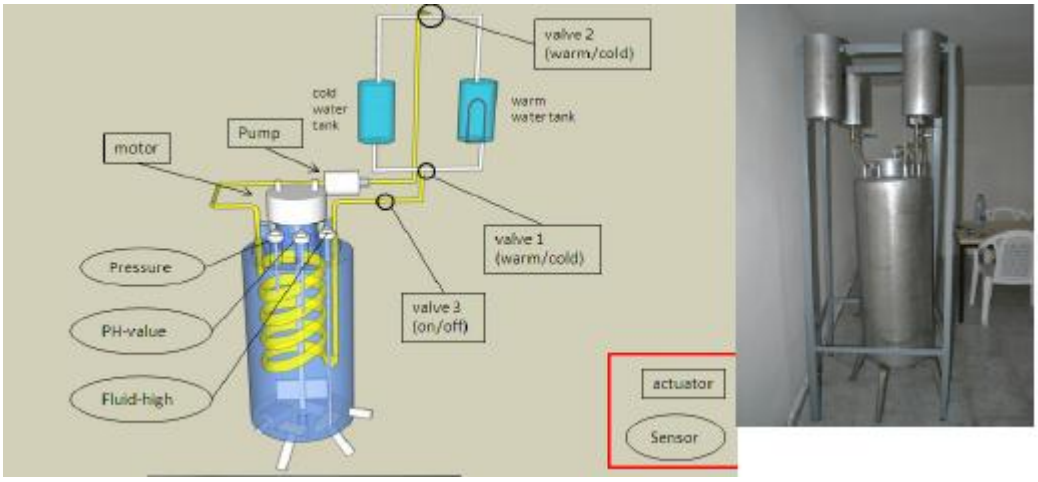



S-101
Centrifugation with
Disk Stack Centrifuge
Homogenization
with Homogenizer
PEG precipitation
Centrifugation with
Disk Stack Centrifuge
CaCl2
precipitation
Centrifugation with
Disk Stack Centrifuge
Ultrafiltration
Fermentation
Gel permeation
chromatographie
with Sepharose 4B
P-1 / V-102
Storage
IEX chromatography
with anion echange
(DEAE matrix)
P-1 / V-102
Storage
Ultracentrifuge
Pool Fractions
Cysteine
Sepharose
4B CLFF
Addition of
Pharmaceutical
suitable exipient
Filter 0.22 micro m

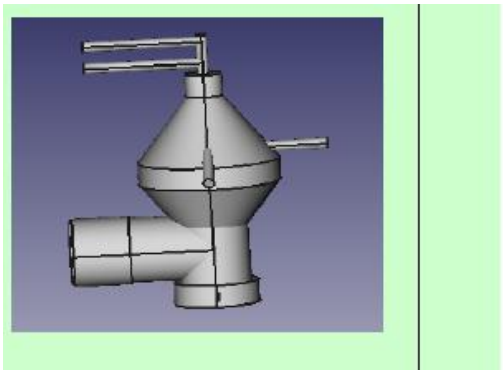
Clean-in-place (CIP)&Steam-in-place (SIP) and intermediate storage	Upstream Processing (USP) – Downstream Processing (DSP)	Automation
 		 

4.3 Fermentation

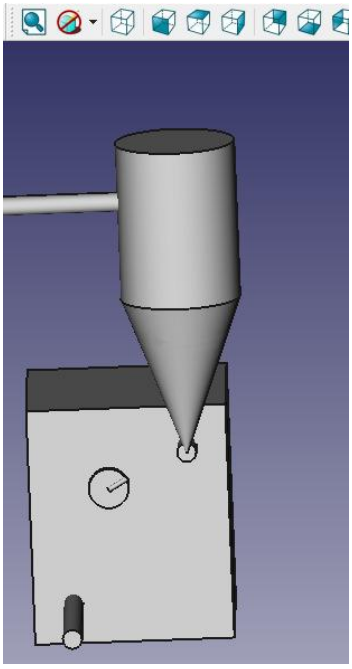
75 L/batch



4.4 Centrifugation with Disc Stack Centrifuge (1)

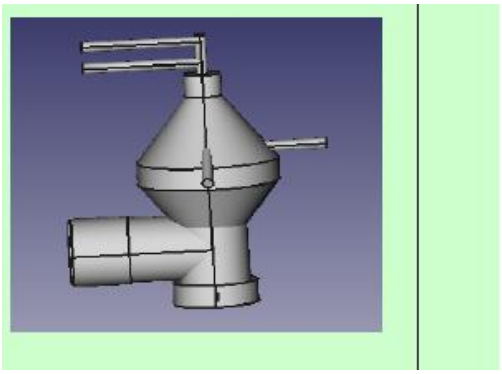


4.5 Homogenization with Homogenizer



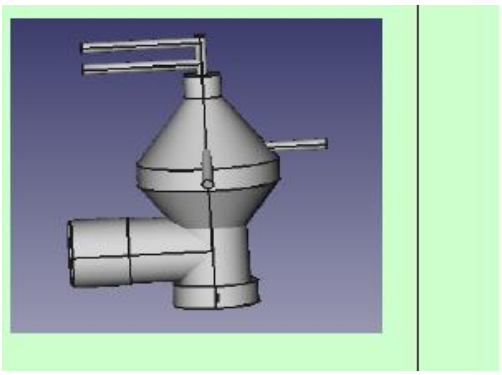
4.6 PEG precipitation

4.7 Centrifugation with Disc Stack Centrifuge (2)



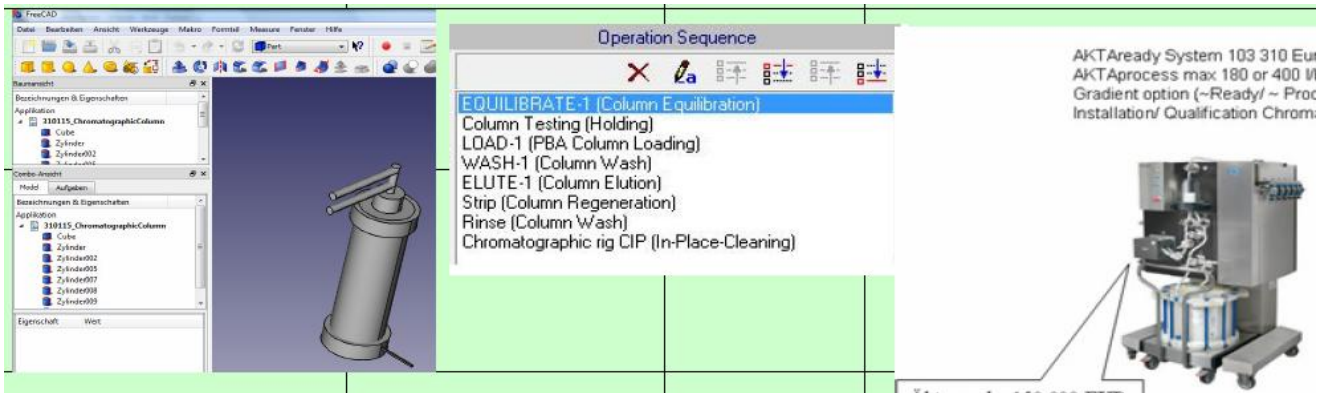
4.8 CaCl2 precipitation

4.9 Centrifugation with Disc Stack Centrifuge (3)

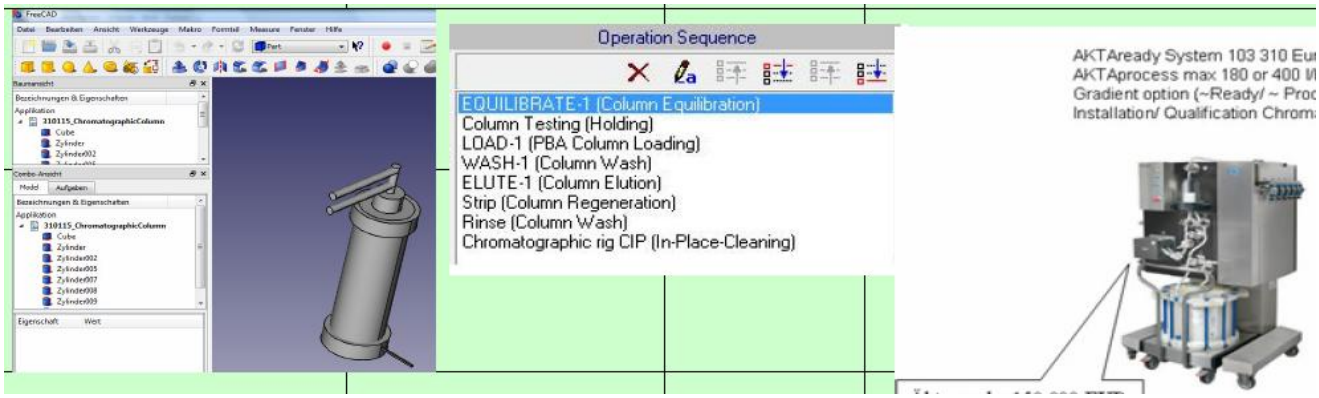


4.10Ultrafiltration

4.11Gel permeation chromatography with Sepharose 4B



4.12IEX chromatography with anion echange (DEAE matrix)



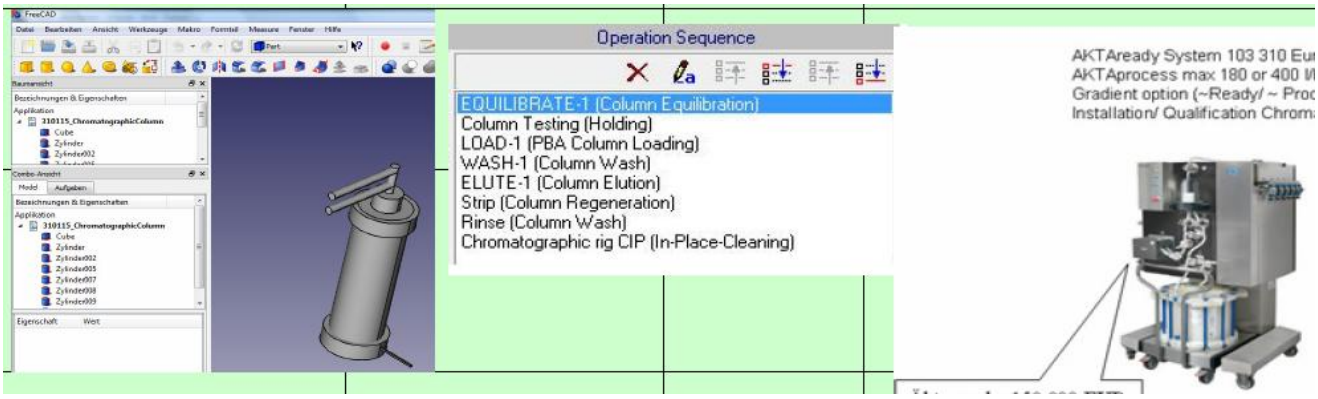
4.13Cystein addition

4.14Ultracentrifugation

Pooled fractions contain HBSAg

Figure: Alfa Wassermann Ultracentrifuges

4.15 Desalting Gel permeation chromatography with Sepharose 4BCLFF



4.16 Filtering with 0.22 µm filter

5. Detailed Devices Mechanical Design

Done as 6 weeks practical work for 4 students (former planned as master thesis)

بسم الله الرحمن الرحيم

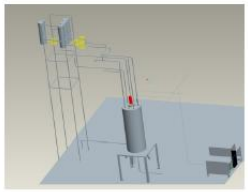
AECENAR
Association for Economical and Technological Cooperation in the Euro-Asian and North-African Region
www.aecenar.com

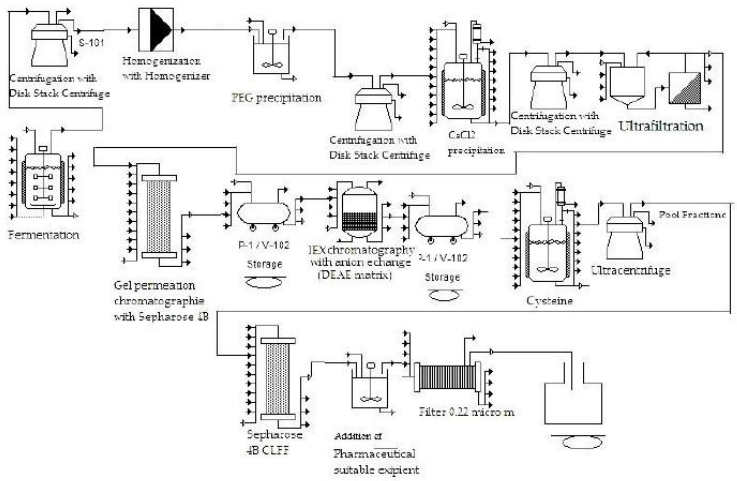
LASeR
Lebanese Association for Scientific Research
الجمعية اللبنانية لدعم البحث العلمي

TEMO
Hardware & Consulting e.K. - Software

Ras Nhache/Batroun - Tripoli, 12th Jan 2015

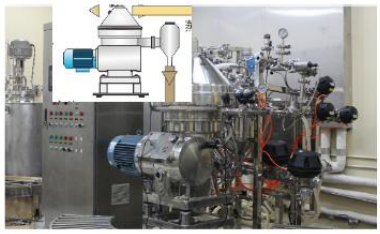
MEGBI Bioreactor





MEGBI-VPP system design:

Downstream System (Purification) from Solaris Group (in this way MEGBI-VPP downstream processing shall be developed)



Disc stack centrifuge from Huading/China

Master Thesis

Modeling and Integration of devices for the vaccine production plant MEGBI-VPP

- Modeling of Homogenizer (3 weeks)
- Modeling of Disc Stack Centrifuge in including CIP/SIP functional elements (mechanics) (3 weeks)
- Modeling Process Scale Gel Permeation and ion exchange chromatographic devices (3 weeks)
- Modeling of Process Scale Ultrafiltration Device (3 weeks)
- Integration of Devices to MEGBI-VPP downstream processing (DSP) unit (2 weeks)
- Documentation (3 weeks)

Keywords: CAD, Tool ProEngineer, Process Scale Homogenizer, Process Scale Centrifugation, Process Scale Protein Chromatography, Biotechnology

AECENAR

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

بسم الله الرحمن الرحيم

Project: MEGBI-VPP (Detailed Devices Mechanical Design for 3rd MEGBI-VPP Report)

Report of Practical Work Jun/Jul 2015 (6 Weeks), Final version: 19 Aug 2015

Authors: Jihad Samarji, ZaherChendeb, Ibrahim Zaaroura, FadiYahya

OBJECTIVE AND PURPOSE OF THE ASSOCIATION:

The association is committed to the promotion of international cooperation in the economic and scientific fields in order to achieve the idea of international understanding and a closer relationship between institutions the Middle East, in Europe and its neighbors. To Download the AECENAR flyer as pdf please click here: AECENAR Brochure 2014

INSTITUTE:

MEGBI - Middle East Genetics and Biotechnology Institute

PROJECT:

MEGBI-VPP / DNA Vaccine Pilot Plant, Recombinant Vaccine Technology / Biotechnological Upstream & Downstream Processing Hepatitis B DNA Vaccine Pilot Plant

PROJET CURRENT TASKS:

Modelling of homogenizer

Modelling of Disc Stack Centrifuge in including CIP/SIP functional elements (mechanics)

Modelling Process Scale Gel Permeation and Ion Exchange Chromatography Devices

Modelling of Process Scale Ultrafiltration Device

Integration of Devices to MEGBI-VPP Downstream processing (DSP) unit

Documentation

SKILLS NEEDED:

CAD, Process Scale Homogenizer, Process Scale Centrifugation, Process Scale Protein Chromatography

SOFTWARE USED:

FREECAD (+ Python MACROS), Microsoft OFFICE, Adobe Suite, Windows OS / Linux OS





5.1 Demonstration and Modelling of the Ultracentrifuge.

"To Separate the High Density from the Low Density Molecules, We Use The Ultracentrifuge." –Jihad Samarji

Figure 1.1 PKII Ultra Centrifuge

5.1.1 *DEVICE DETAILS & SPECIFICATIONS*

The **ultracentrifuge** is a centrifuge optimized for spinning a rotor at very high speeds, capable of generating acceleration as high as 2 000 000 g (approx. 19 600 km/s²). There are two kinds of ultracentrifuges, the preparative and the analytical ultracentrifuge. Both classes of instruments find important uses in molecular biology, biochemistry, and polymer science.

In our Case we're using the Preparative Ultracentrifuge

Preparative ultracentrifuges are available with a wide variety of rotors suitable for a great range of experiments. Most rotors are designed to hold tubes that contain the samples. *Swinging bucket rotors* allow the tubes to hang on hinges so the tubes reorient to the horizontal as the rotor initially accelerates. *Fixed angle rotors* are made of a single block of material and hold the tubes in cavities bored at a predetermined angle.

In our Case we're using the Zonal Rotor

Zonal rotors are designed to contain a large volume of sample in a single central cavity rather than in tubes. Some zonal rotors are capable of dynamic loading and unloading of samples while the rotor is spinning at high speed.

Preparative rotors are used in biology for pelleting of fine particulate fractions, such as cellular organelles (mitochondria, microsomes, and ribosomes) and viruses. They can also be used for gradient separations, in which the tubes are filled from top to bottom with an increasing concentration of a dense substance in solution. Sucrose gradients are typically used for separation of cellular organelles. Gradients of caesium salts are used for separation of nucleic acids. After the sample has spun at high speed for sufficient time to produce the separation, the rotor is allowed to come to a

smooth stop and the gradient is gently pumped out of each tube to isolate the separated components. In our Case we're using it to produce anti-bodies

We Need To Design a Density Gradient Ultracentrifugation System for large scale and pilot scale downstream processing of viral vaccines and virus like particles.

	Air Drive
Models Available	KII and PKII
Process Air	2.83 m^3/min
Process Cooling	4.5 °C, 4 l/min
Electrical Supply	15A, 1ph 230V
Environmental Conditions	0-40°C, RH 85%
Clean Room Classifications	Class B,C,D
Bio-containment	Up to BL3
System Footprint	2140 x 1330 (W x D)
Height Requirements	295 cm (PKII 220 cm)
System Weight	1270 kg (PKII 1204 kg)

Table 1.1 Ultracentrifuge KII&PKII specifications

So we're going to design something close to the ALFA WASSERMAN KII & PKII Continuous Flow Ultracentrifuge model. See figure 1.1 and 1.2

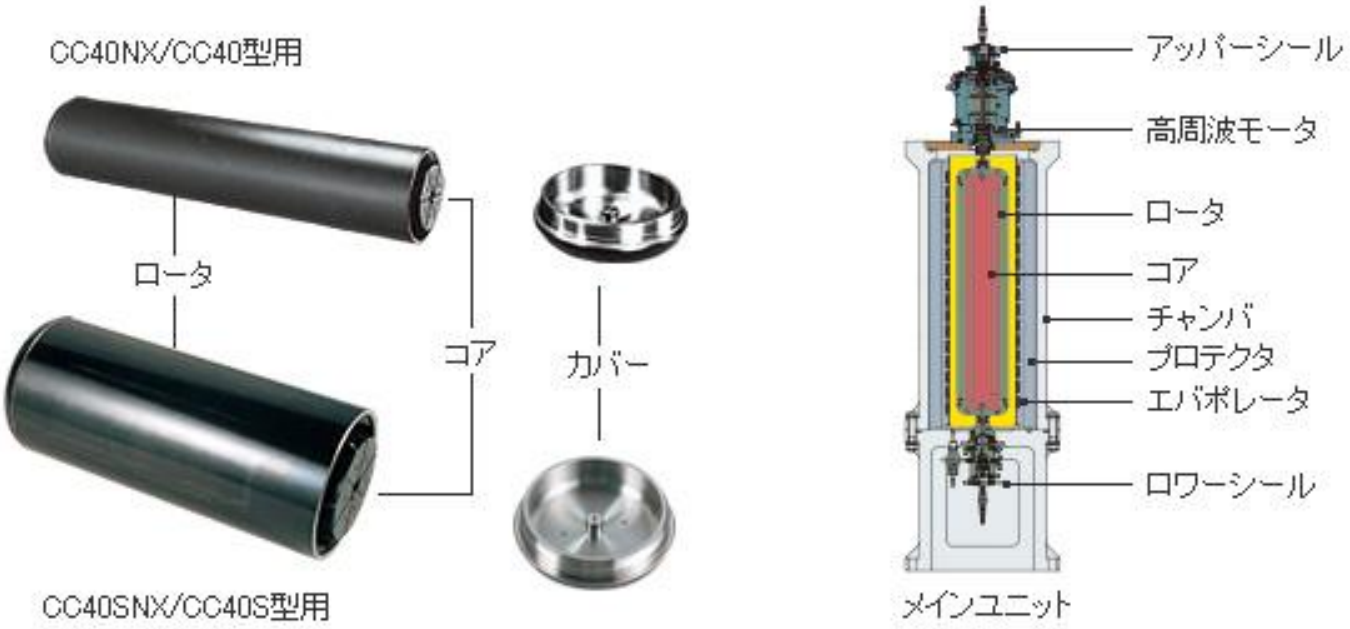
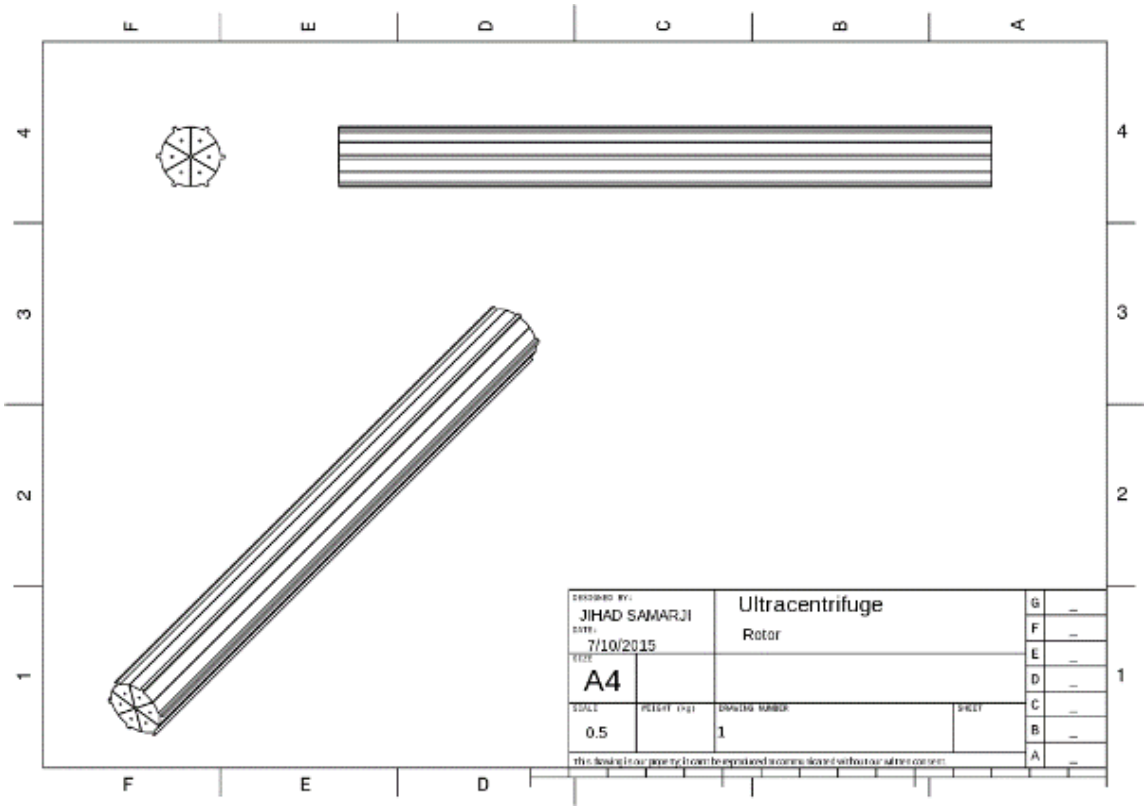


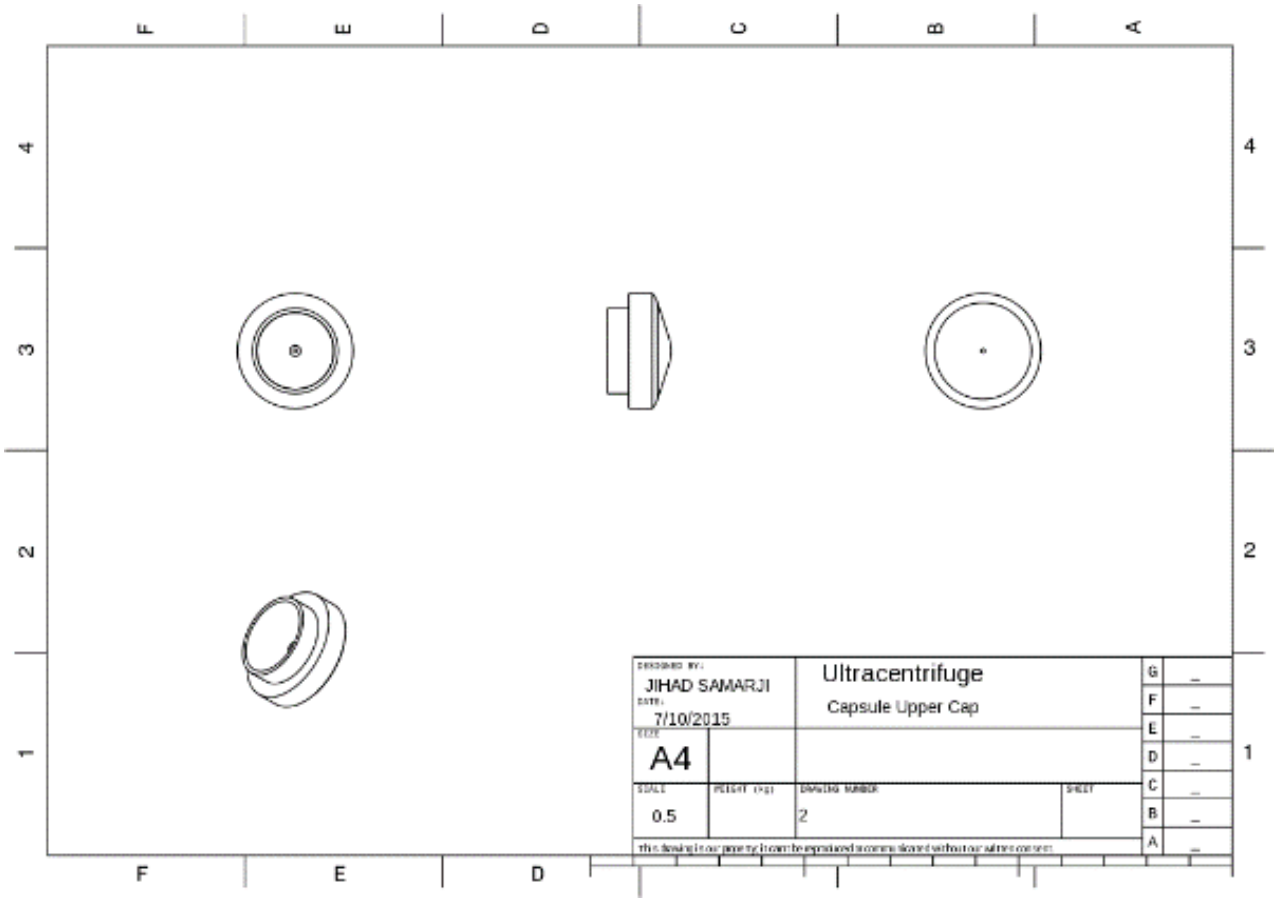
Figure 1.2 Ultracentrifuge annotation in Chinese

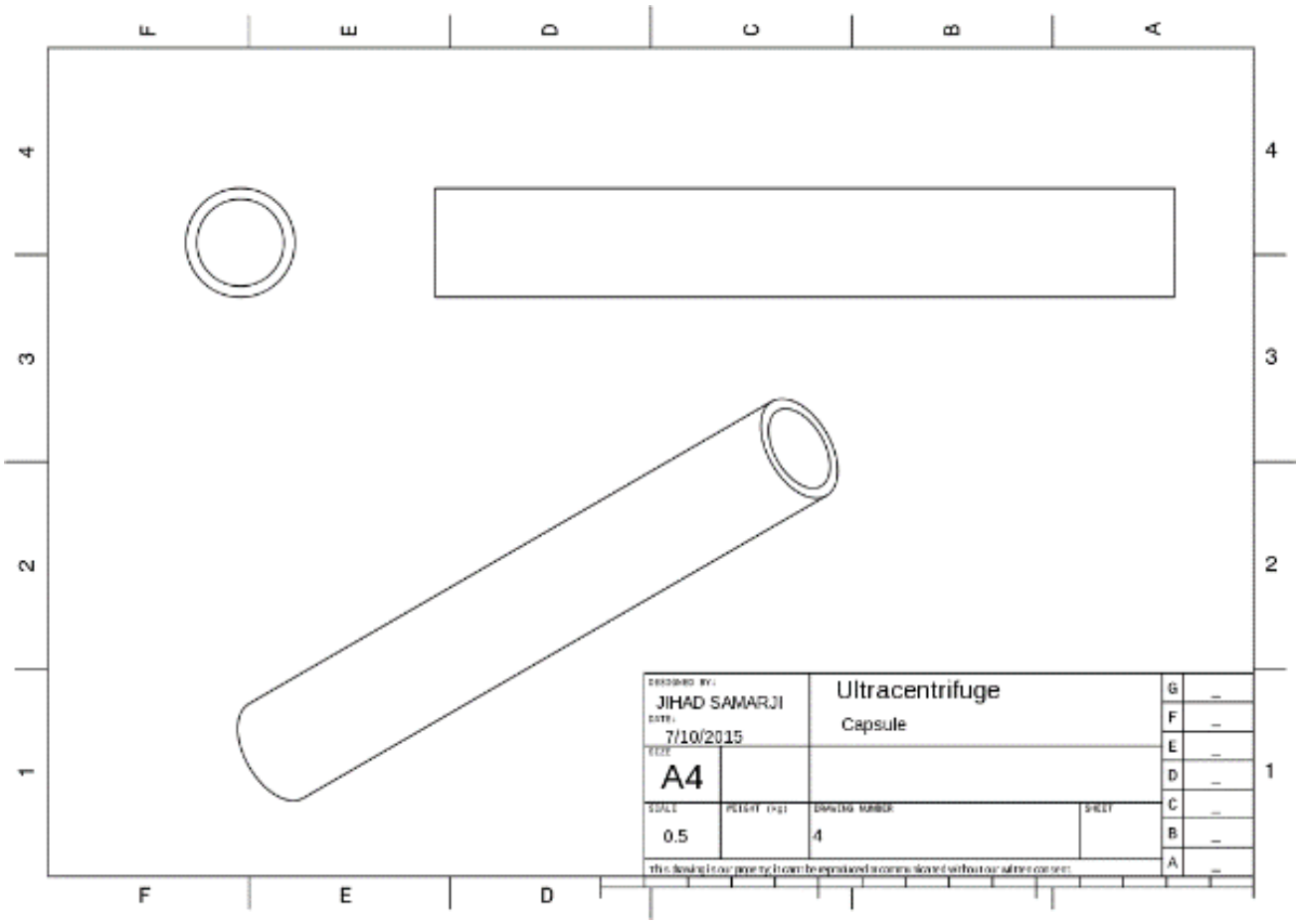
The Ultracentrifuge can be used For Uranium Enrichment which makes it very dangerous and Complicated, Therefore gaining information about this particular machine is very limited. This is by far the clearest picture describing the Ultracentrifuge on the internet.

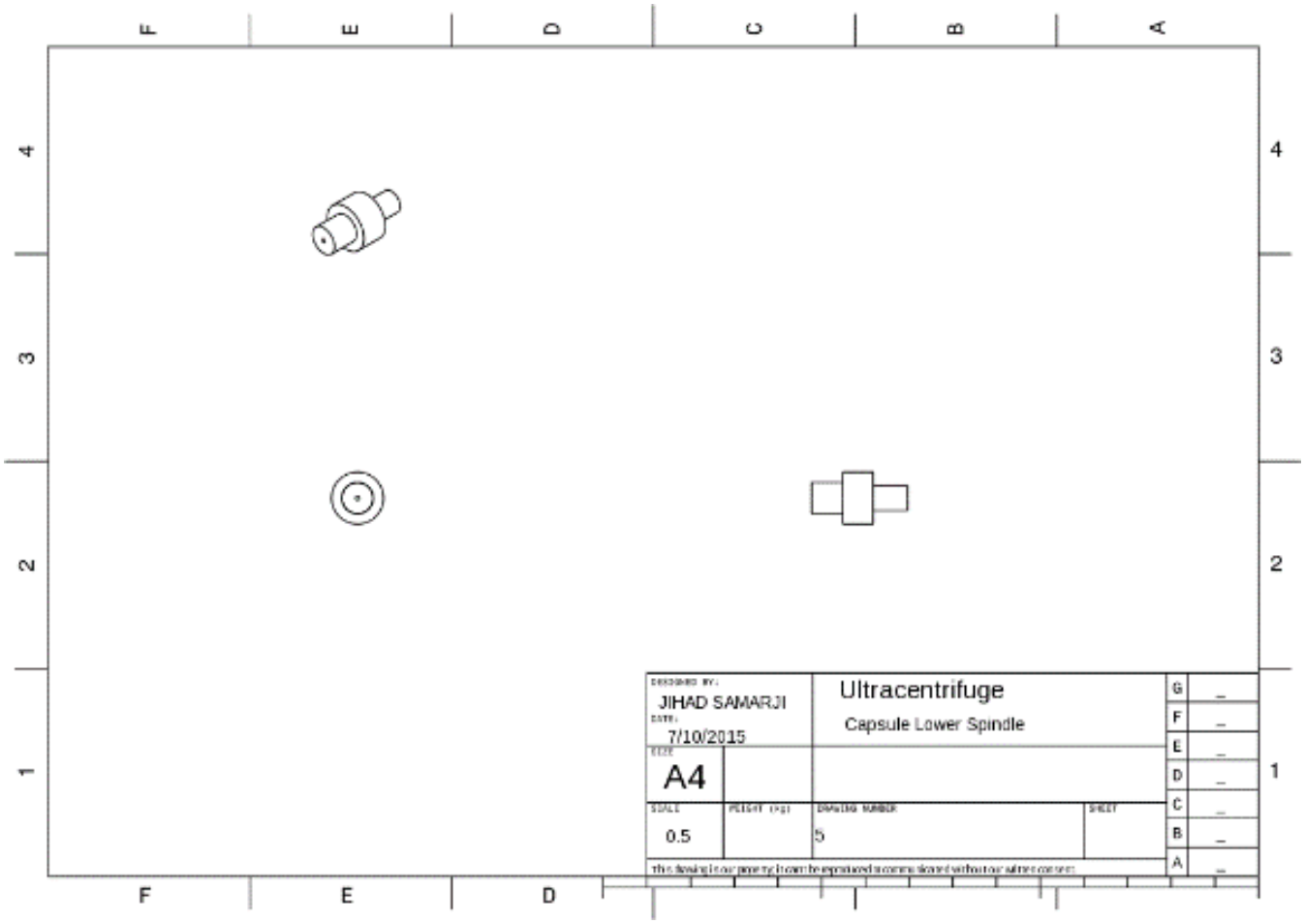
In fact, searching for the Keyword “Ultracentrifuge” on Google makes you suspicious for Global Terrorism. BE CAUTIOUS

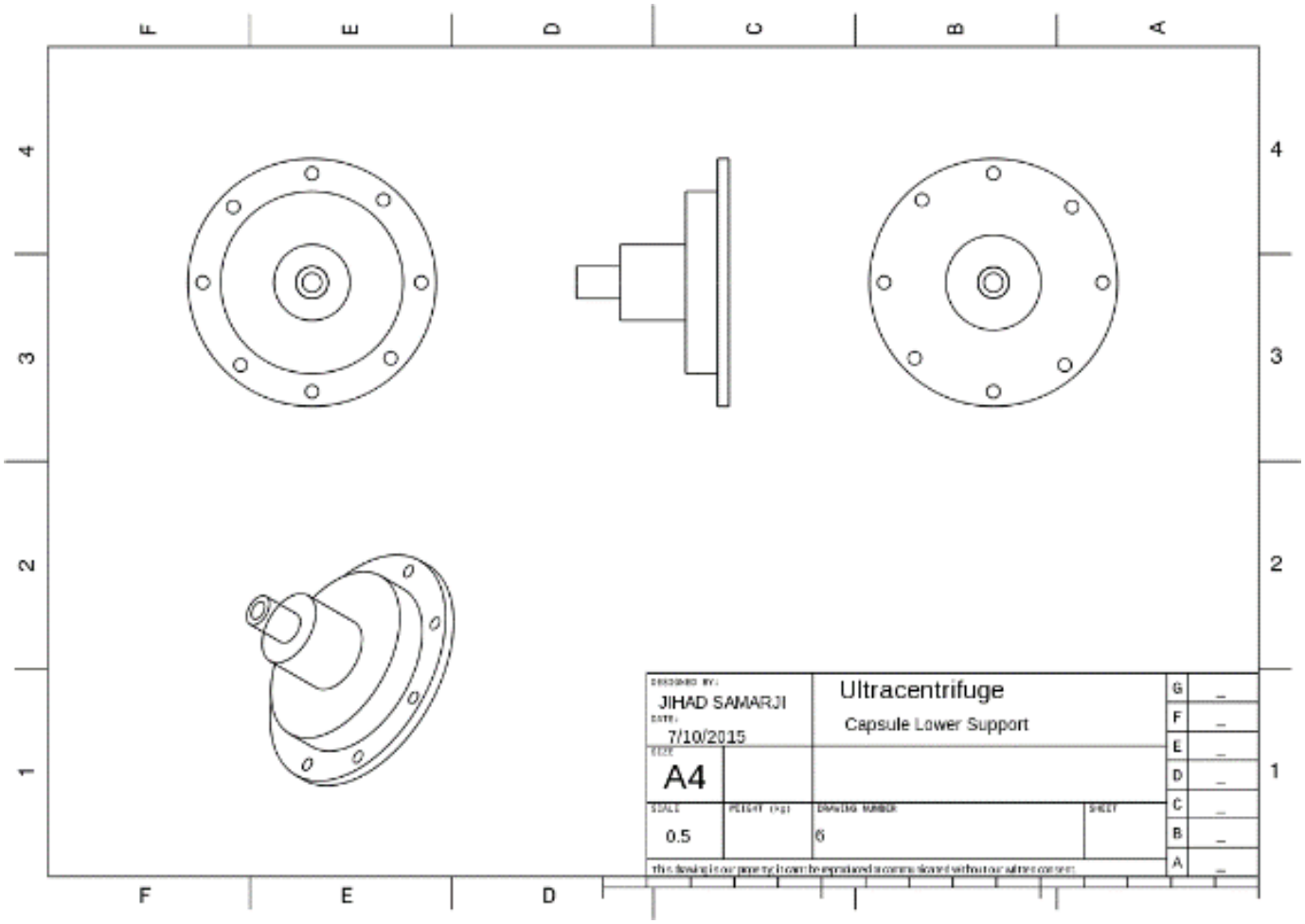
5.1.2 *PART DIMENSIONS & DESIGN*

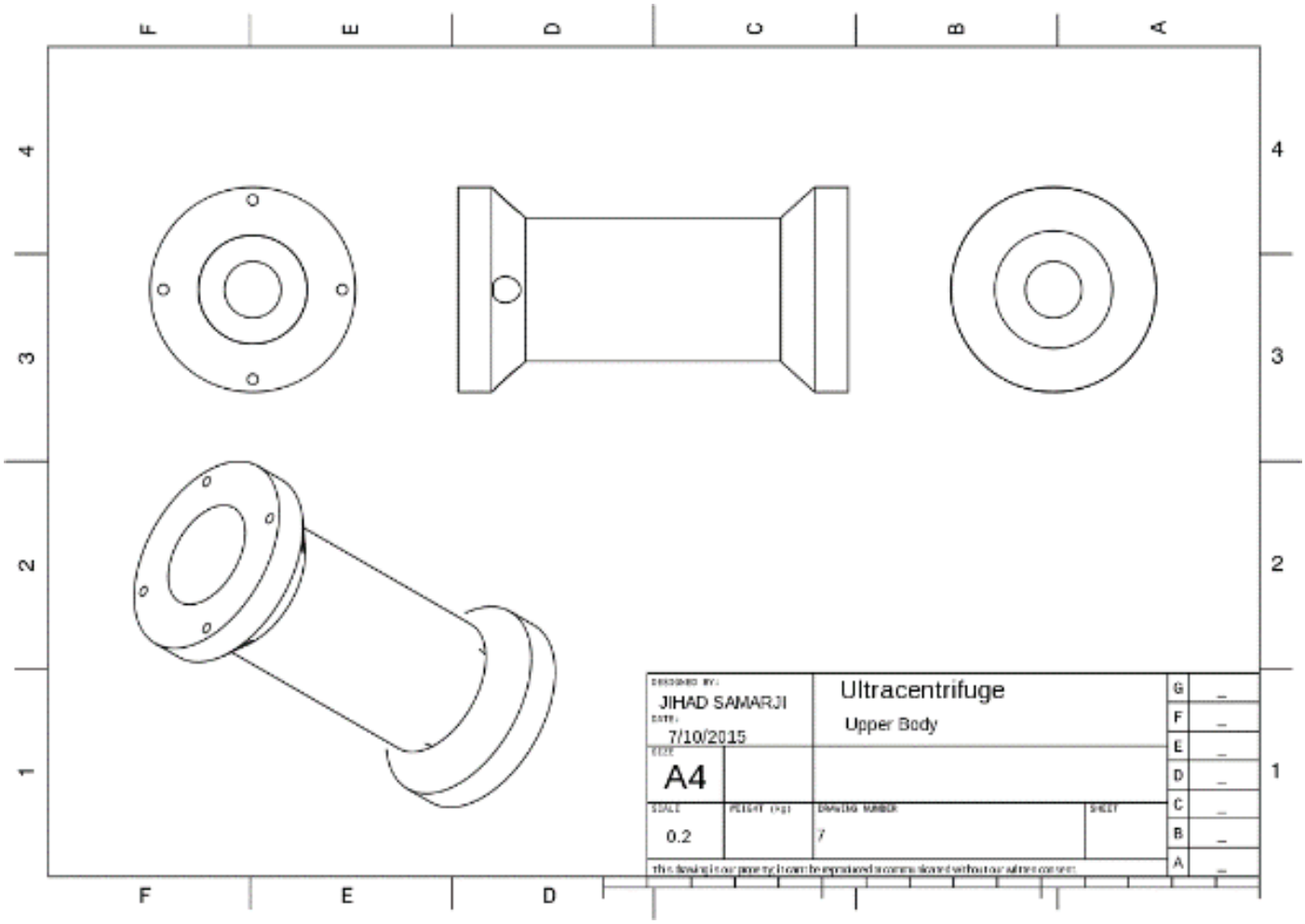


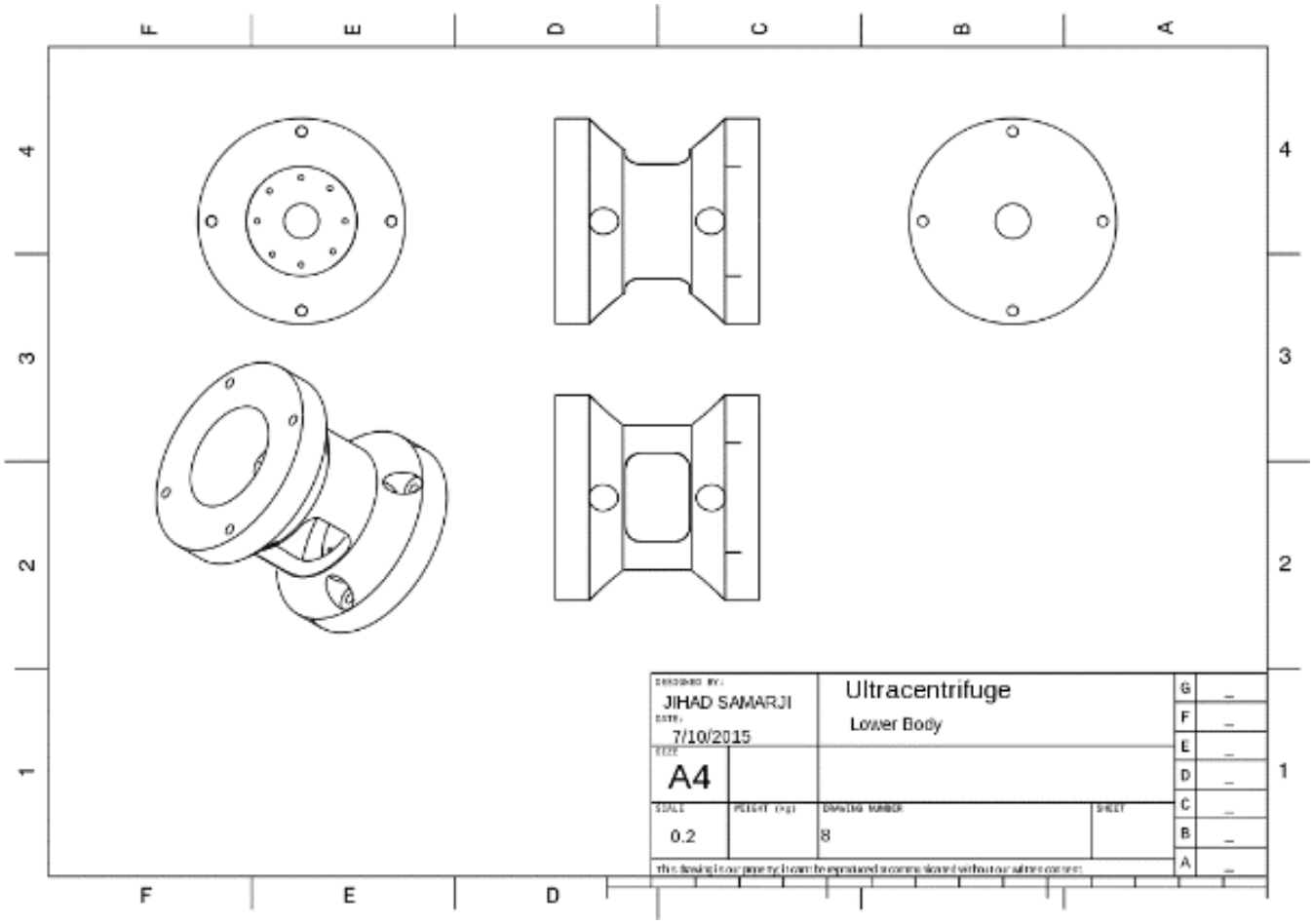


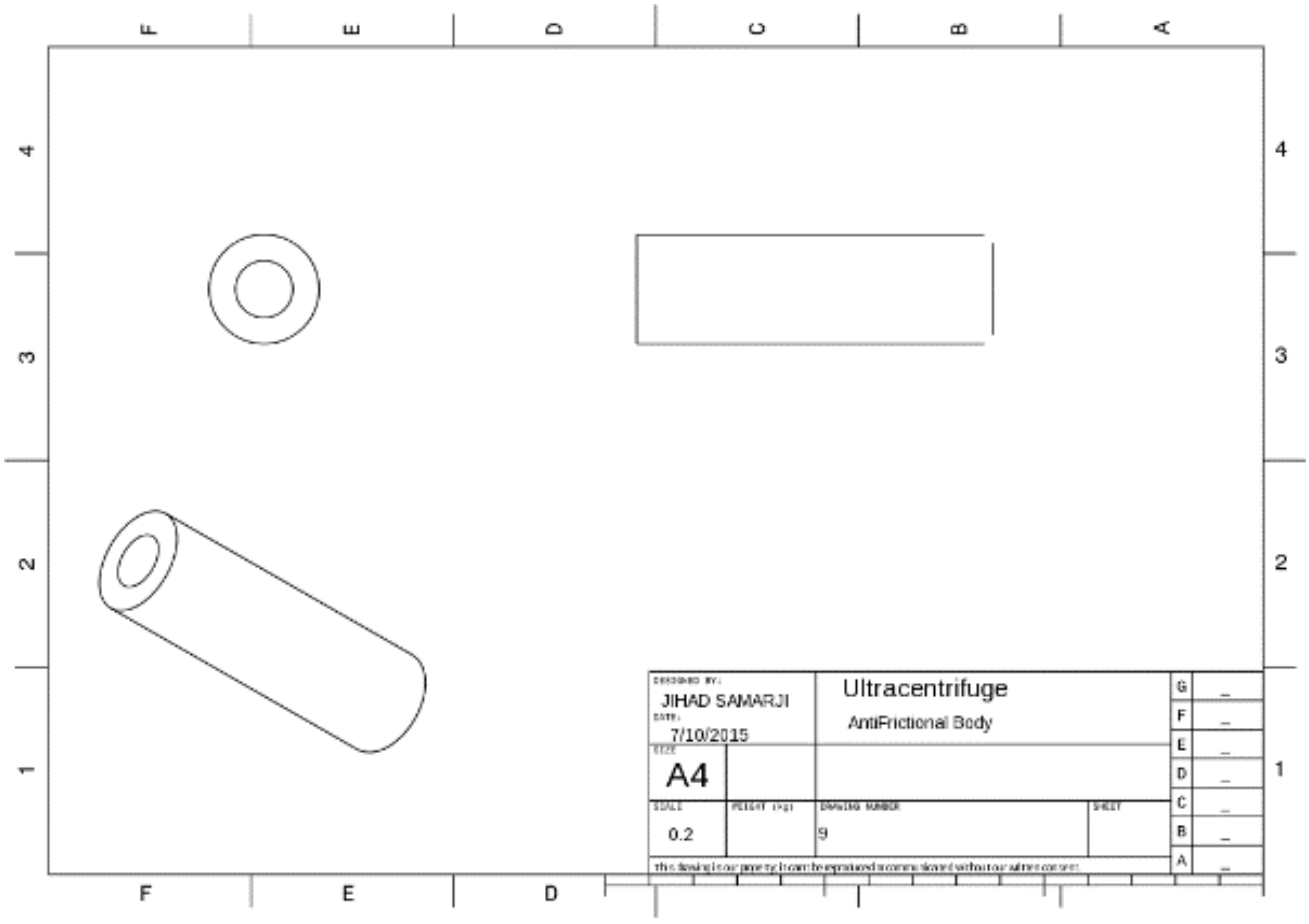

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.5
WEIGHT (kg)
DRAWING NUMBER
4
SHEET
Ultracentrifuge
Capsule

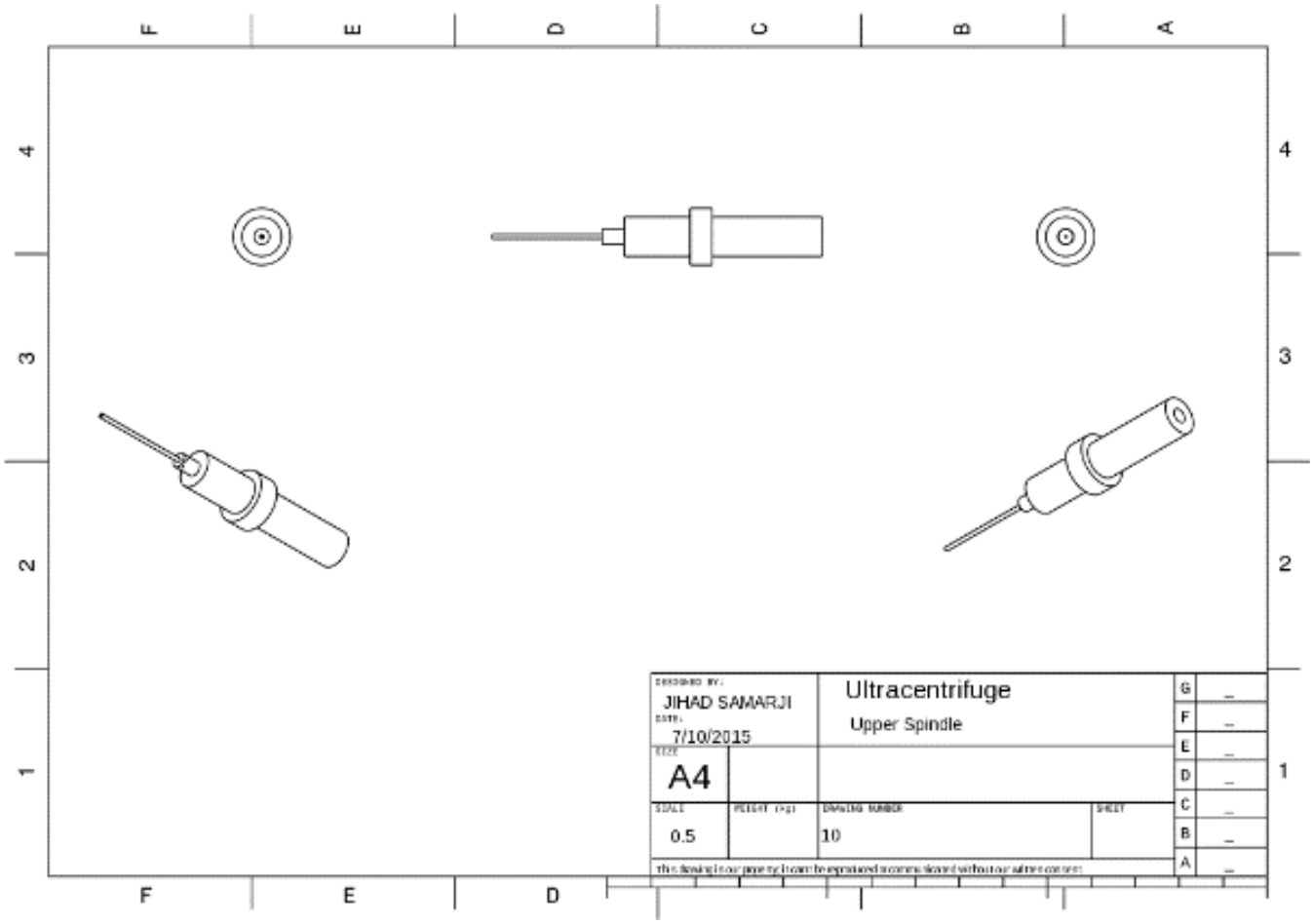

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.5
WEIGHT (kg)
DRAWING NUMBER
5
SHEET
Ultracentrifuge
Capsule Lower Spindle

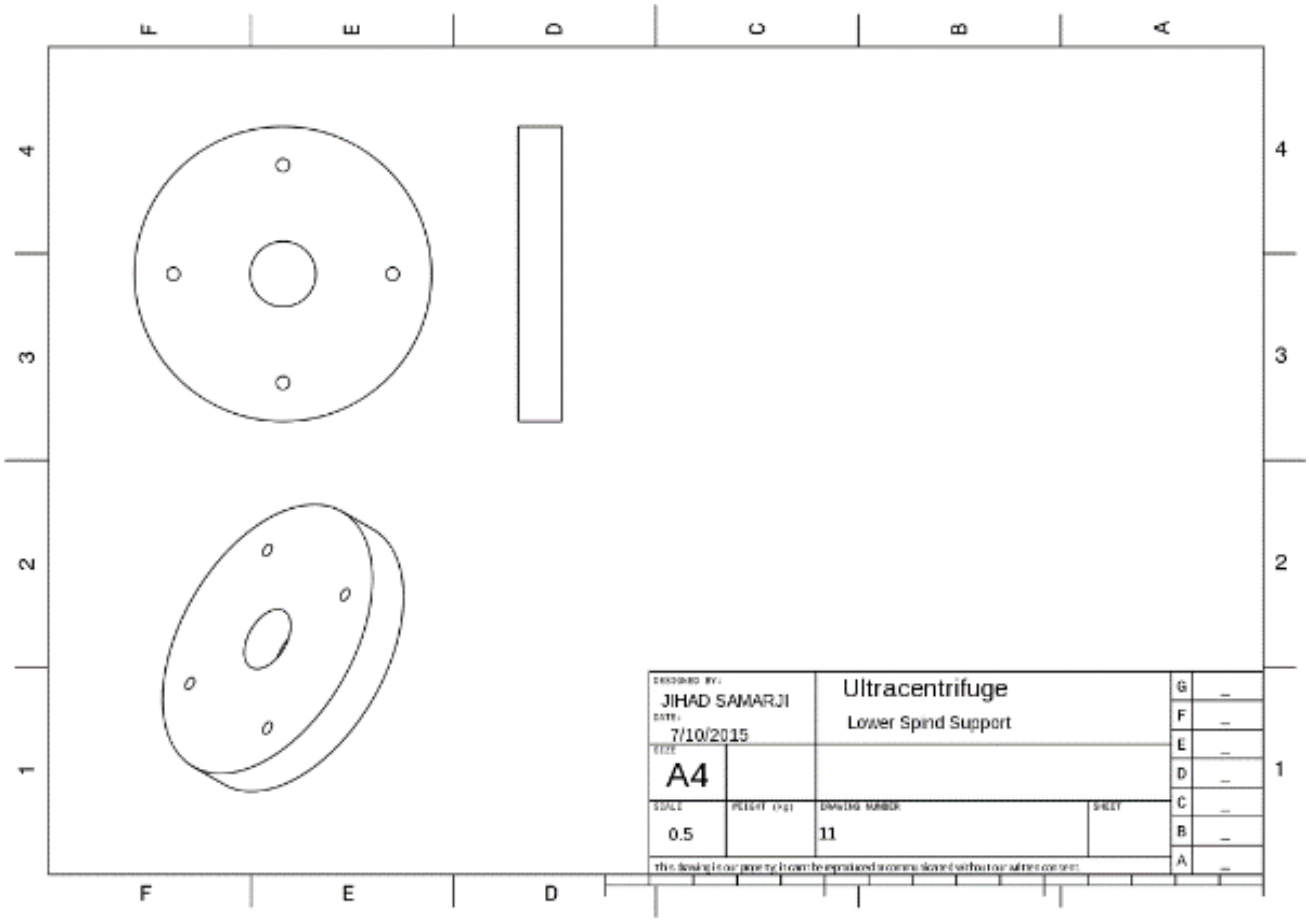

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.5
WEIGHT (kg)
DRAWING NUMBER
6
SHEET
Ultracentrifuge
Capsule Lower Support

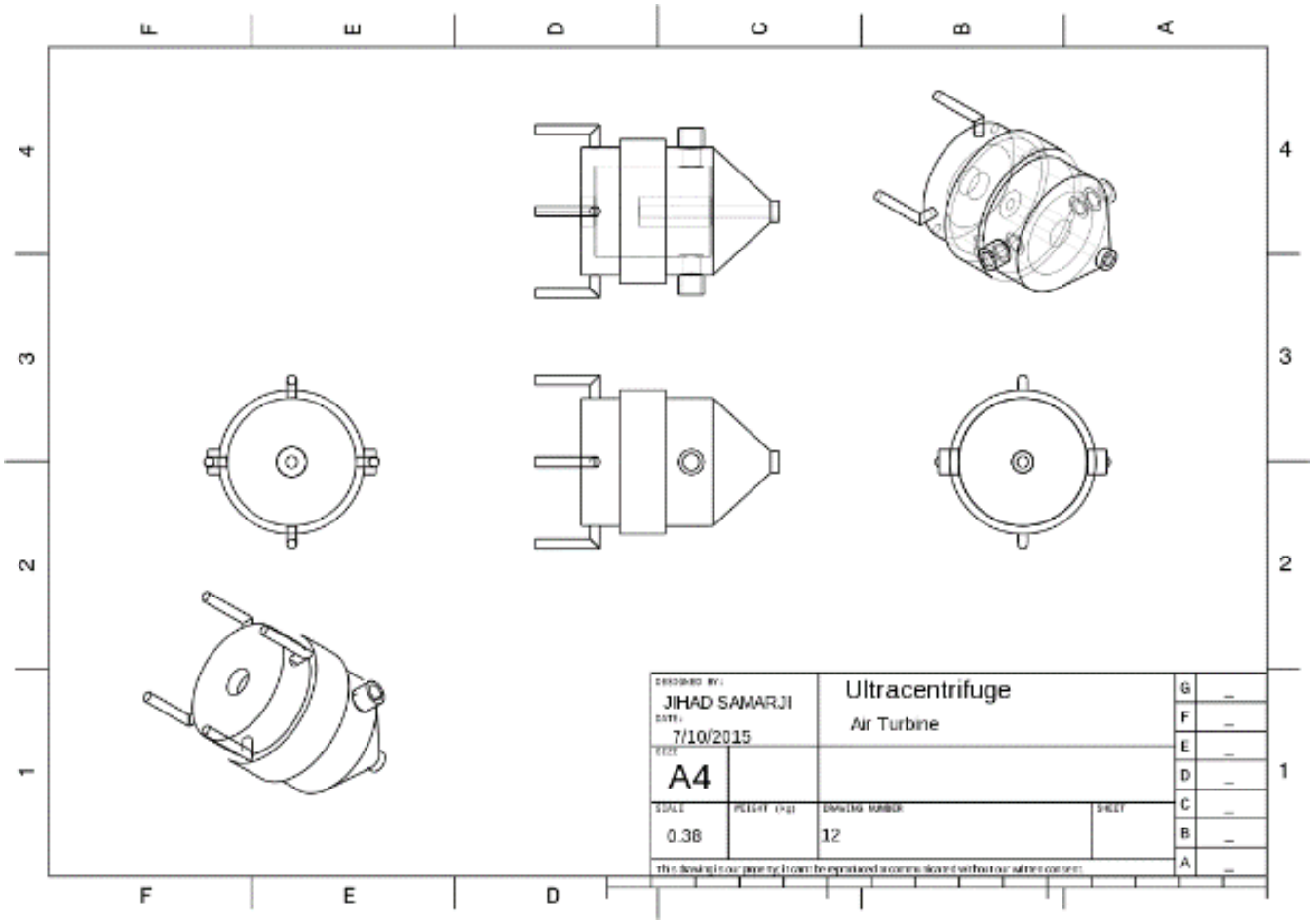

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.2
WEIGHT (kg)
DRAWING NUMBER
7
SHEET
Ultracentrifuge
Upper Body

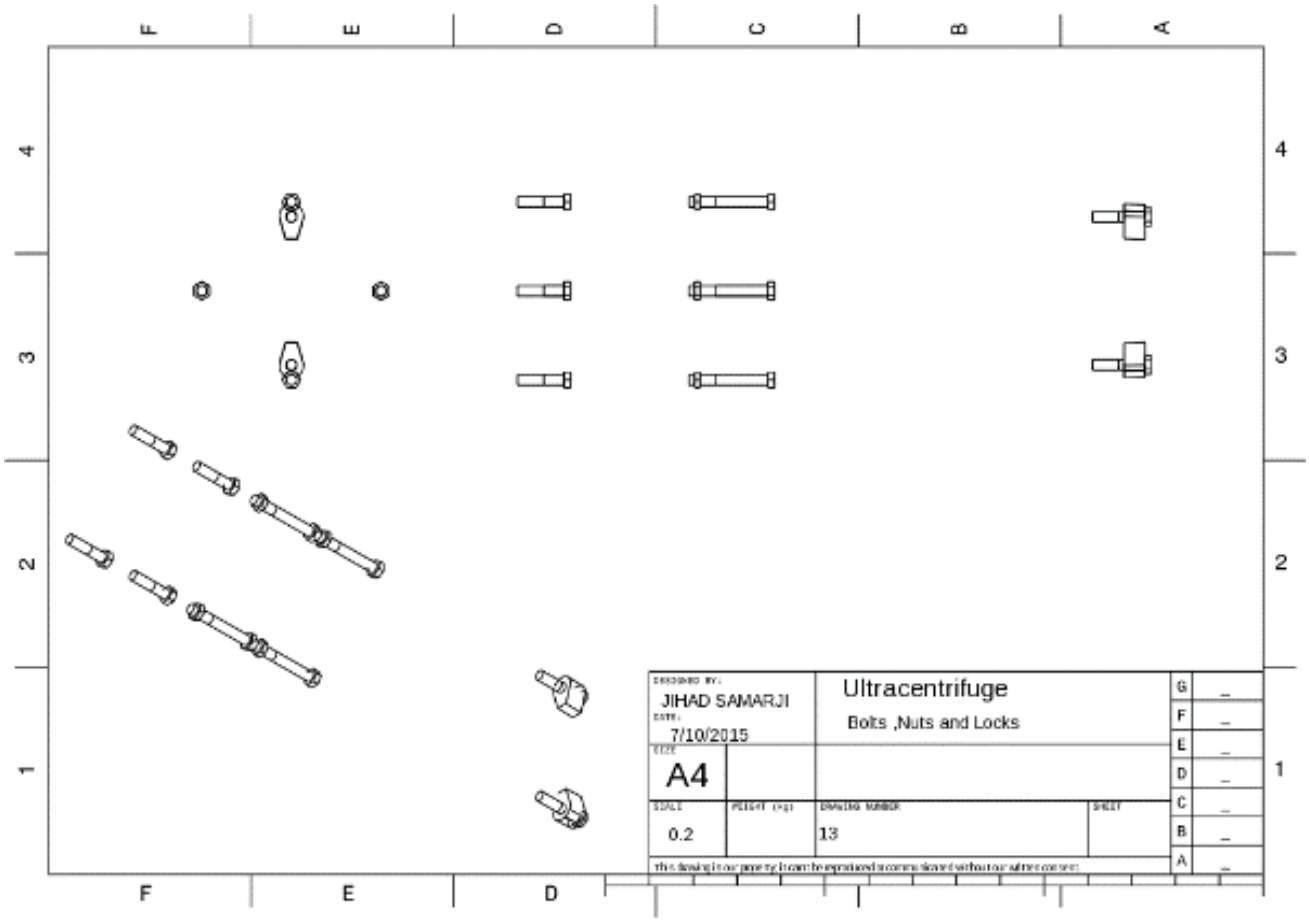

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.2
WEIGHT (kg)
DRAWING NUMBER
8
SHEET
Ultracentrifuge
Lower Body

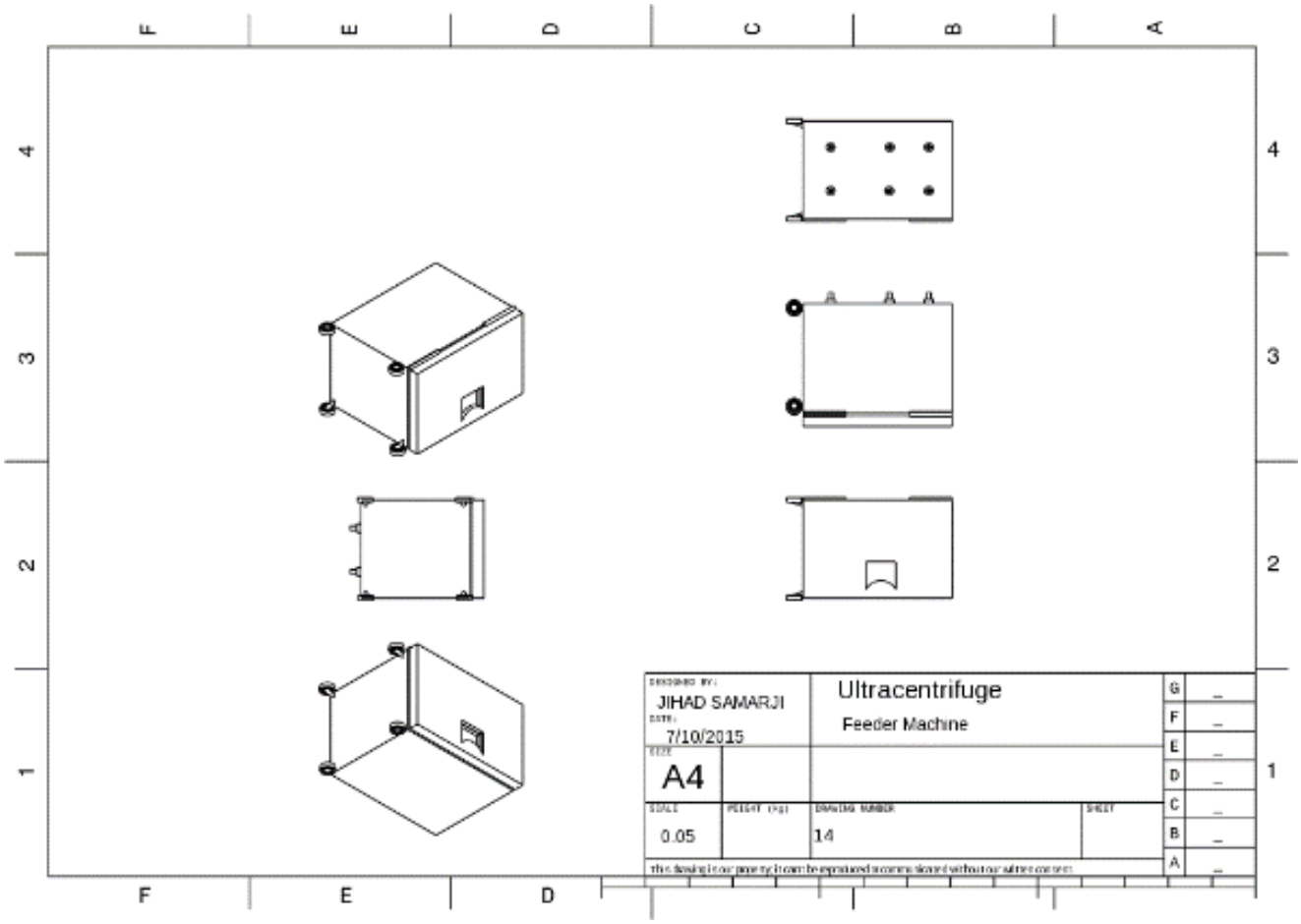

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.2
WEIGHT (kg)
DRAWING NUMBER
9
SHEET
Ultracentrifuge
AntiFrictional Body

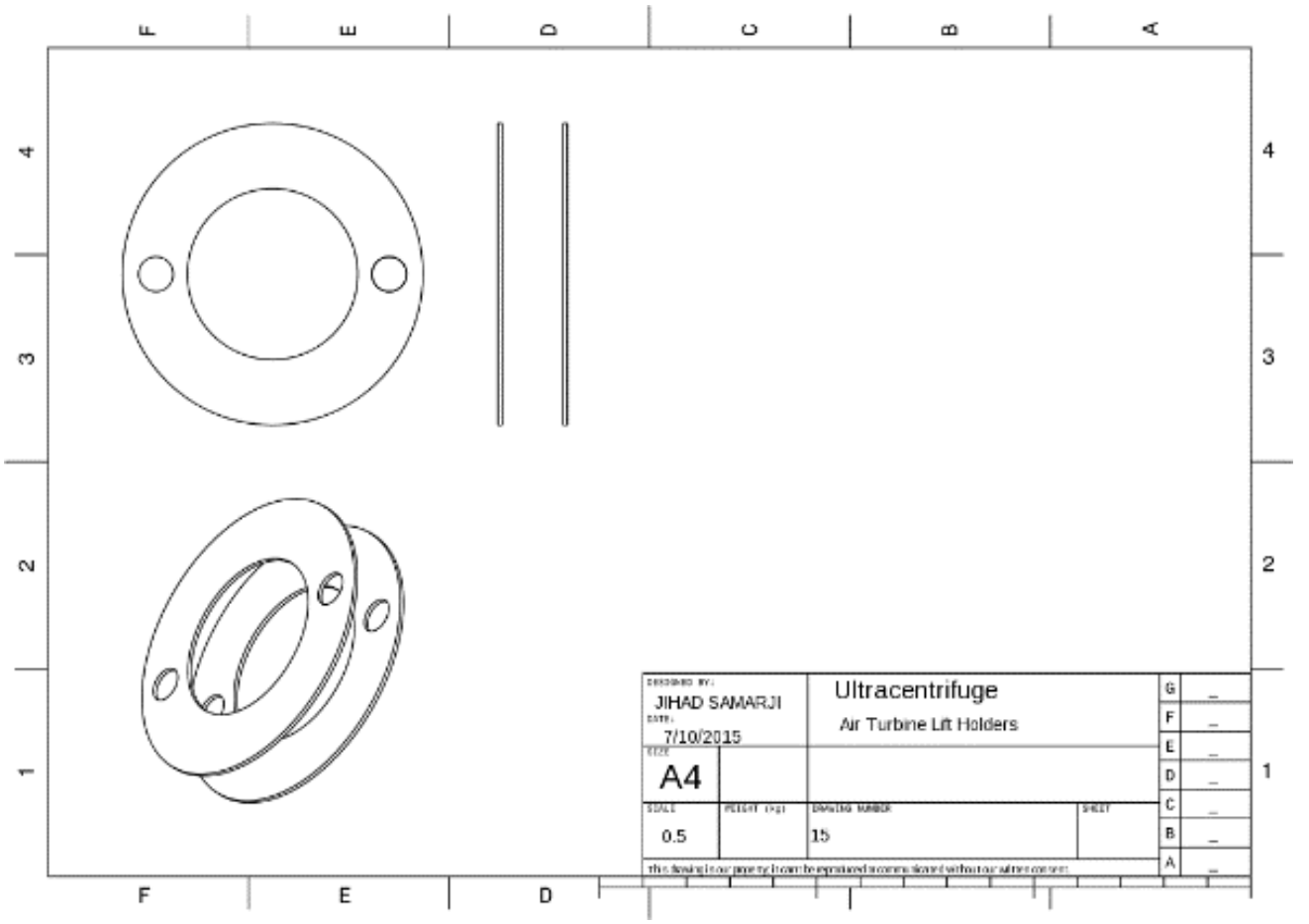

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.5
WEIGHT (kg)
DRAWING NUMBER
10
SHEET
Ultracentrifuge
Upper Spindle

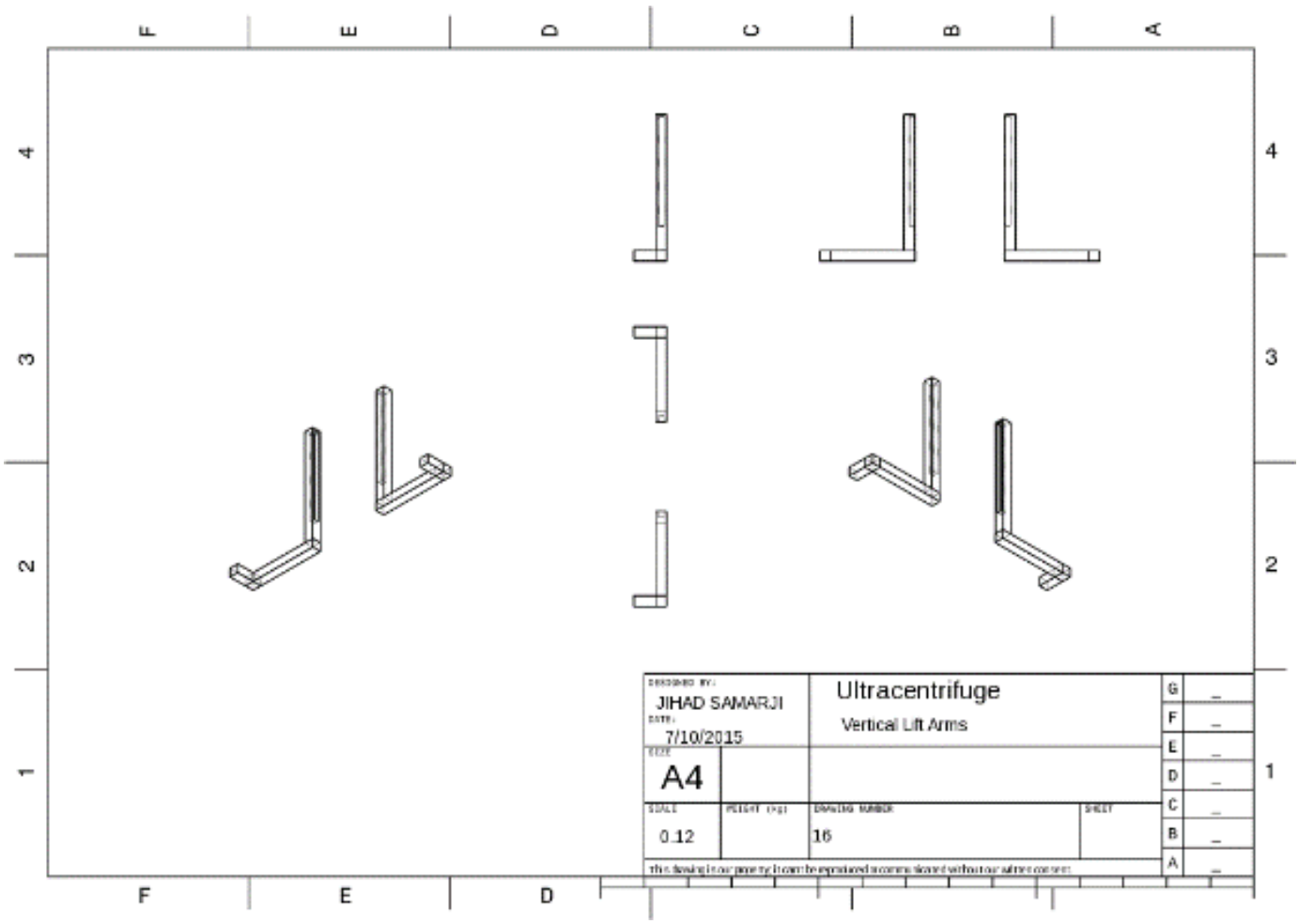

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.5
WEIGHT (kg)
DRAWING NUMBER
11
SHEET
Ultracentrifuge
Lower Spind Support

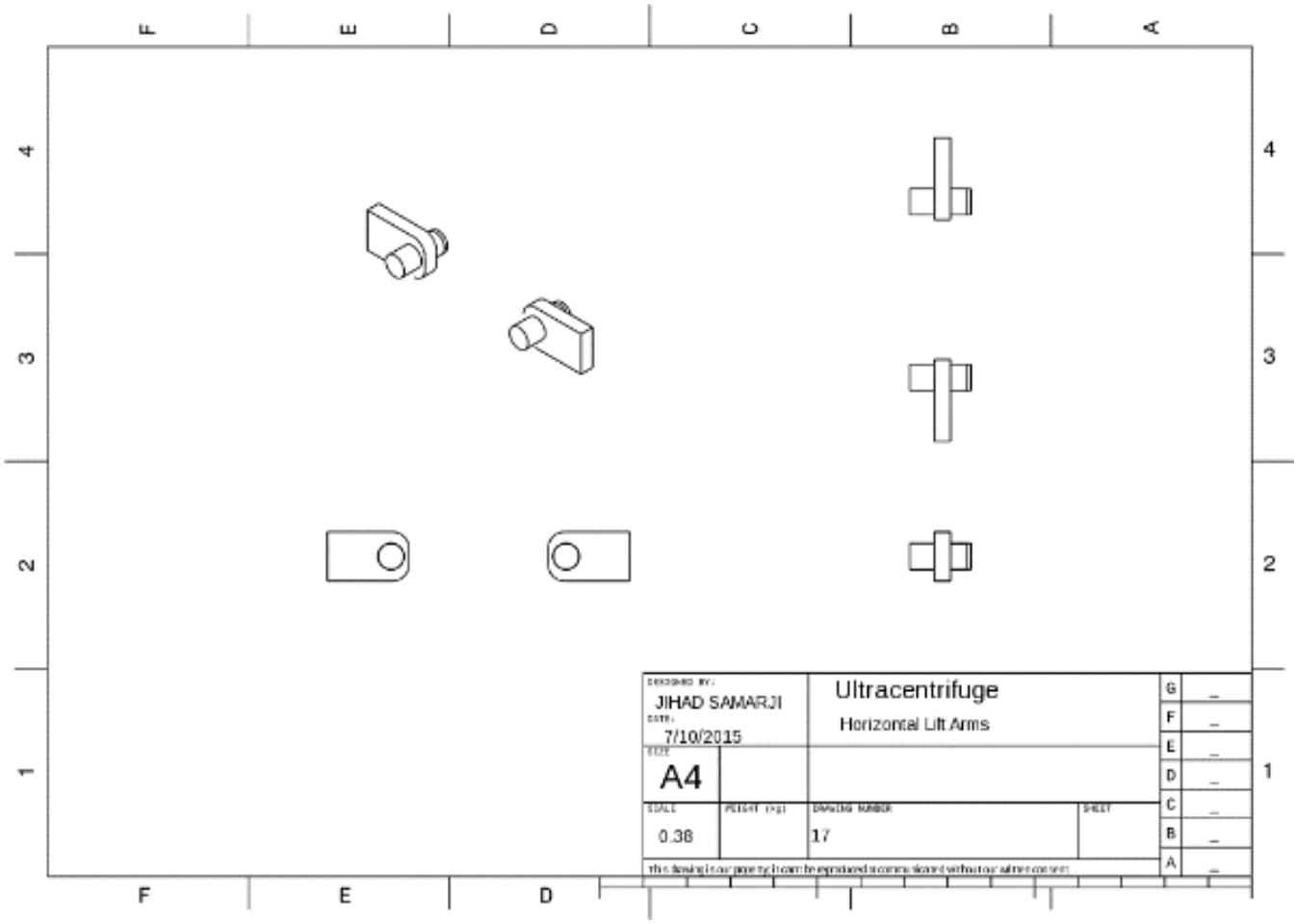

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.38
WEIGHT (kg)
DRAWING NUMBER
12
SHEET
Ultracentrifuge
Air Turbine

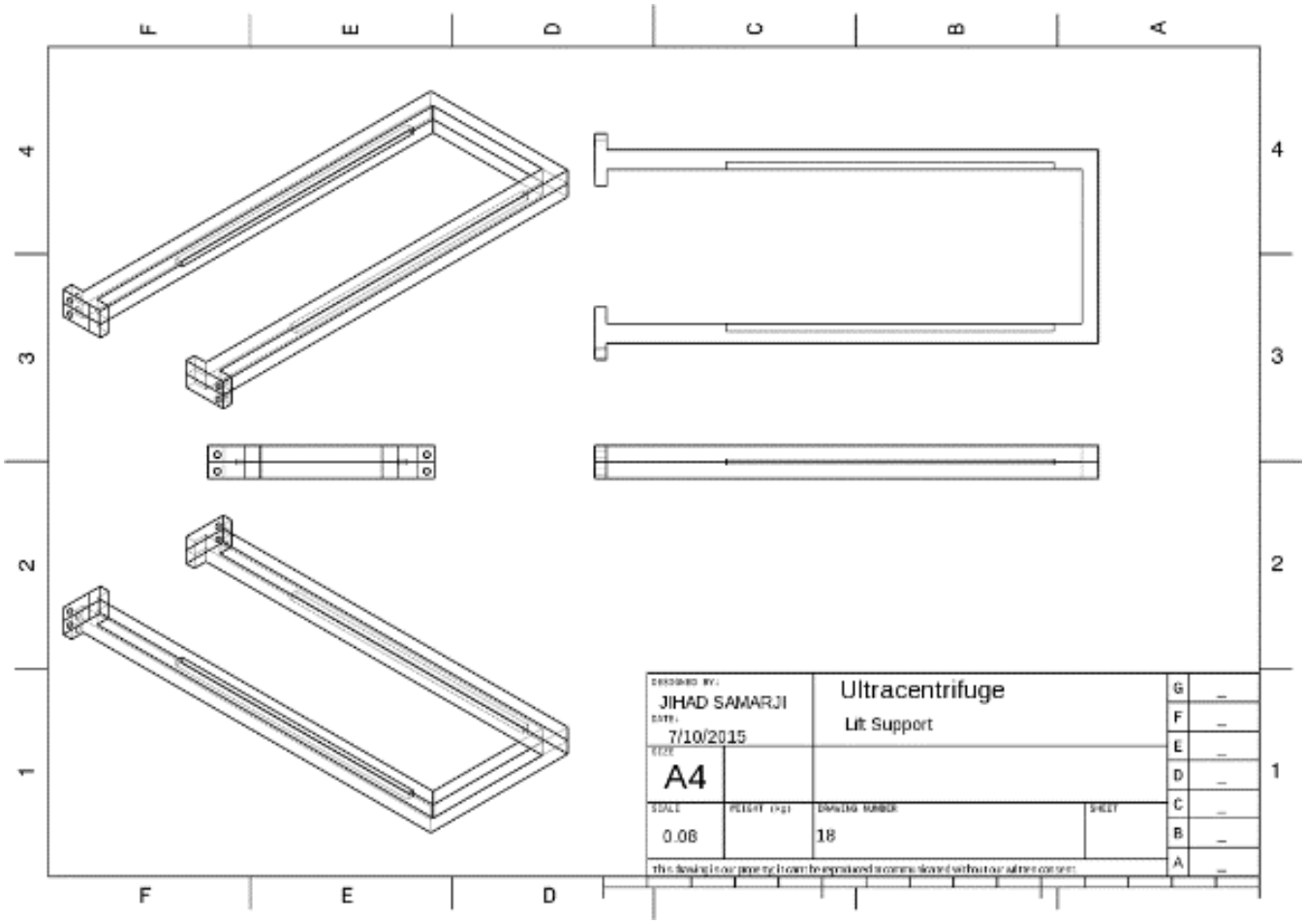

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.2
WEIGHT (kg)
DRAWING NUMBER
13
SHEET
Ultracentrifuge
Bolts ,Nuts and Locks

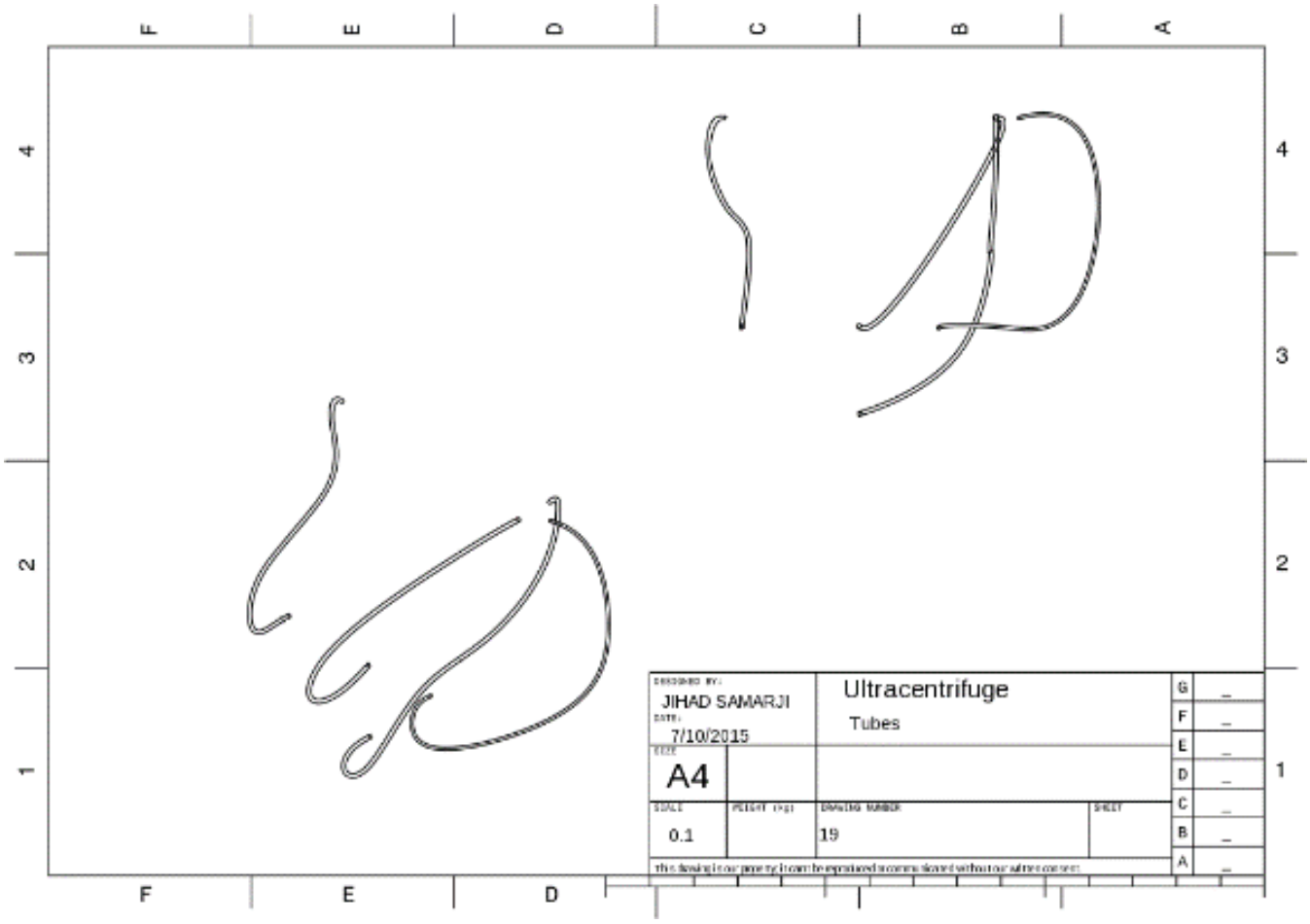

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.05
WEIGHT (kg)
DRAWING NUMBER
14
SHEET
Ultracentrifuge
Feeder Machine


DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.5
WEIGHT (kg)
DRAWING NUMBER
15
SHEET
Ultracentrifuge
Air Turbine Lift Holders


DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.12
WEIGHT (kg)
DRAWING NUMBER
16
SHEET
Ultracentrifuge
Vertical Lift Arms
this drawing is our property it can be reproduced in commercial use without our written consent


DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.38
WEIGHT (kg)
DRAWING NUMBER
17
SHEET
Ultracentrifuge
Horizontal Lift Arms
this drawing is our property it can be reproduced in commercial use without our written consent


DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.08
WEIGHT (kg)
DRAWING NUMBER
18
SHEET
Ultracentrifuge
Lift Support


DESIGNED BY:
JIHAD SAMARJI
DATE:
7/10/2015
SIZE
A4
SCALE
0.1
WEIGHT (kg)
DRAWING NUMBER
19
SHEET
Ultracentrifuge
Tubes

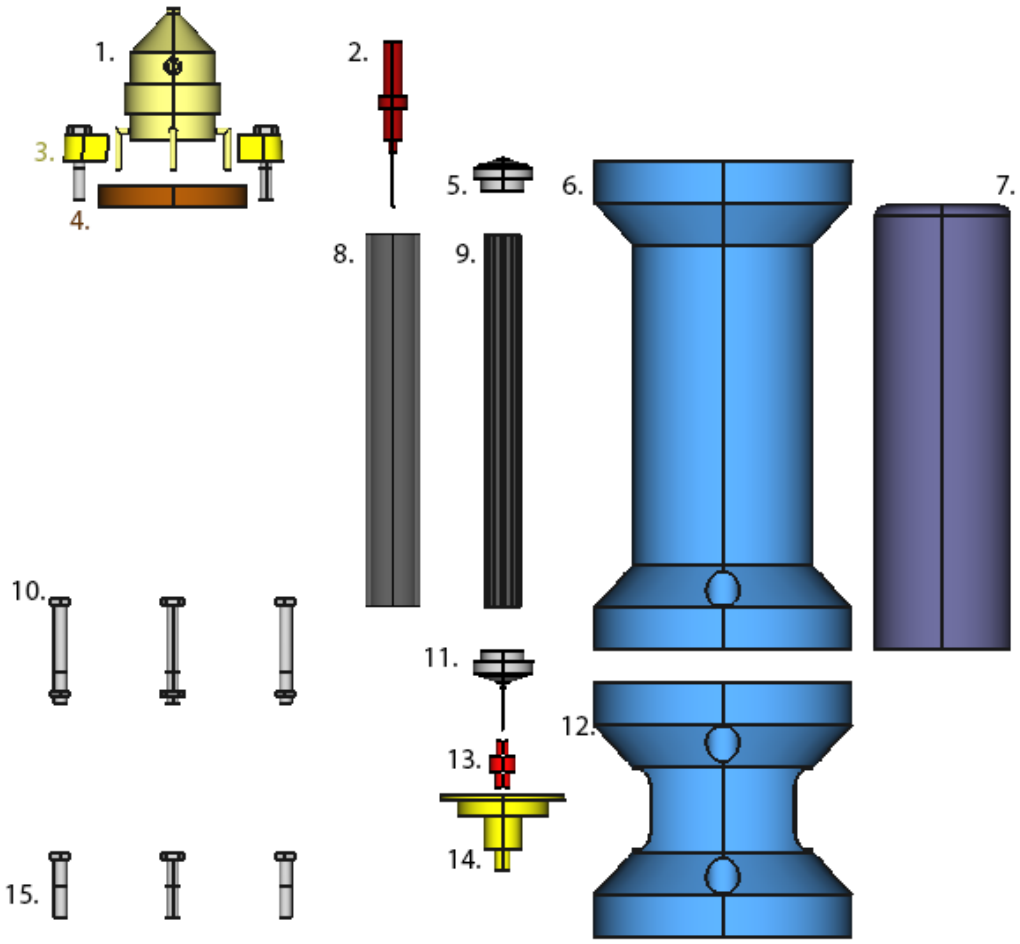


Figure 1.3 Ultracentrifuge First Group Parts Annotations

1. Air Turbine 2.Upper spindle 3.Locks 4.Air Turbine Support 5.Upper Capsule Cap 6.Upper Body 7. Anti-frictional body 8.Capsule 9.Rotor 10.Bolts & Nuts 11.Lower Capsule Cap 12.Lower Body 13. Lower Spindle 14.Lower Spindle Support 15.Bolts

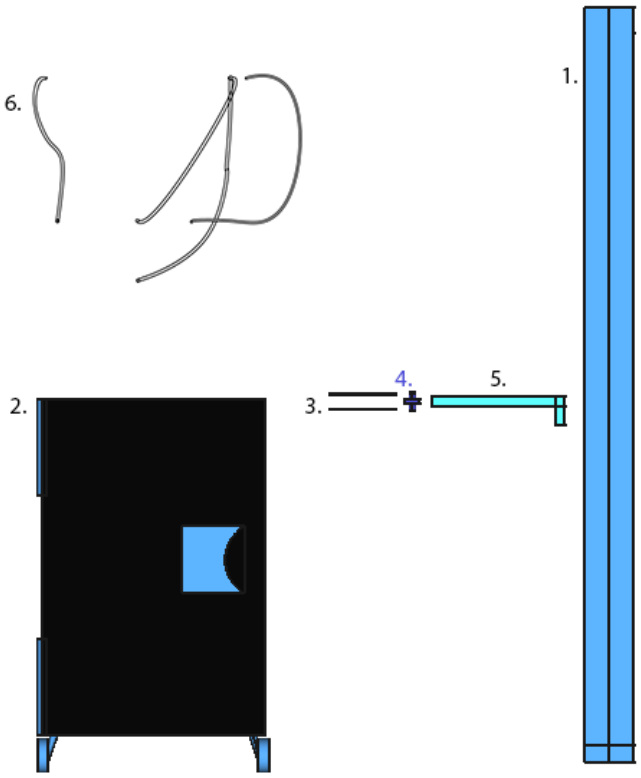
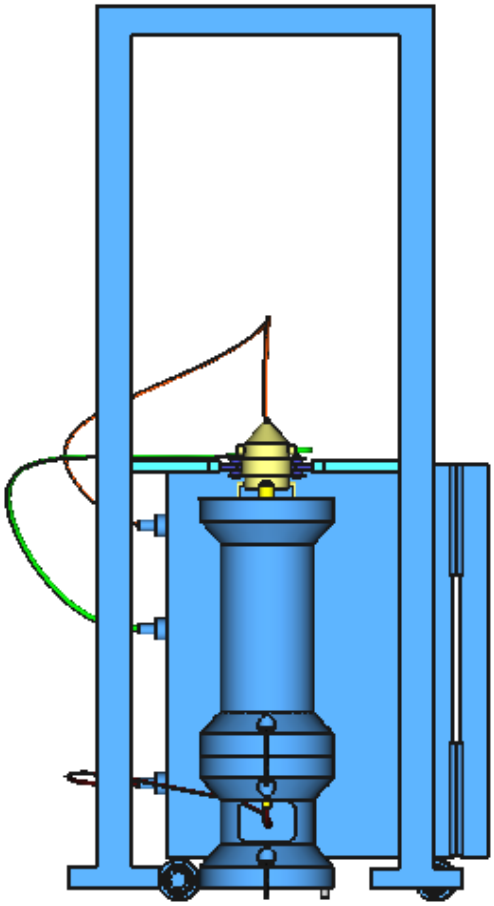
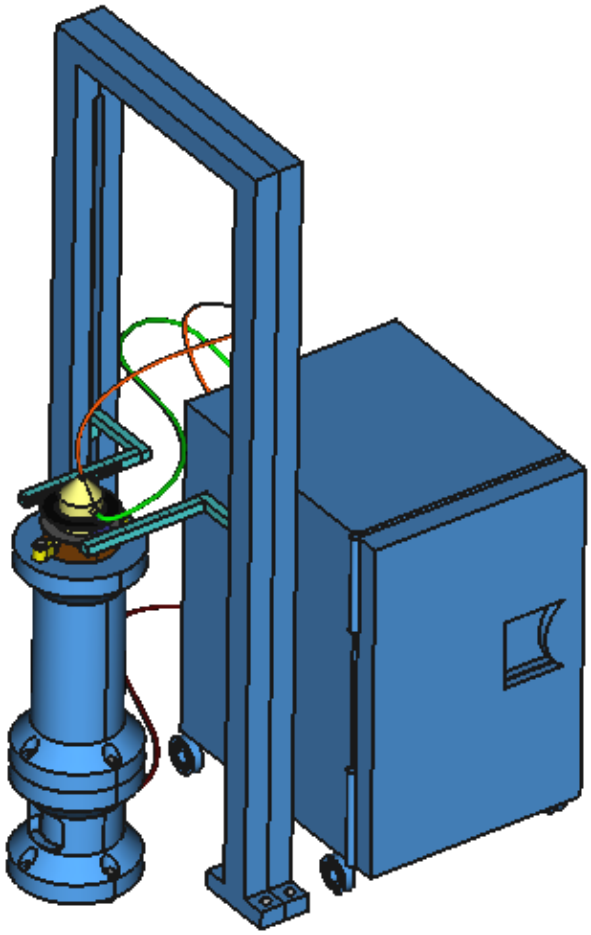


Figure 1.4 ULTRACENTRIFUGE SECOND GROUP PART ANNOTATIONS

1. Lift Support 2.Feeding Machine 3.Air Turbine Lift Holders 4.Horizontal Lift Arm 5.Vertical Lift Arm 6. Tubes

Remark: This Design Is for Educational Purposes. This is just a Prototype Concept Which will Not Work Because Some Parts Are Missing.





5.2 Demonstration and Modelling of the Disc Stack Centrifuge

"To Separate a Liquid from Solid or Liquid from Liquid, We Use the Decanter or the Disc Stack or a Filter Depending On the Amount ofSolids"-Jihad Samarji

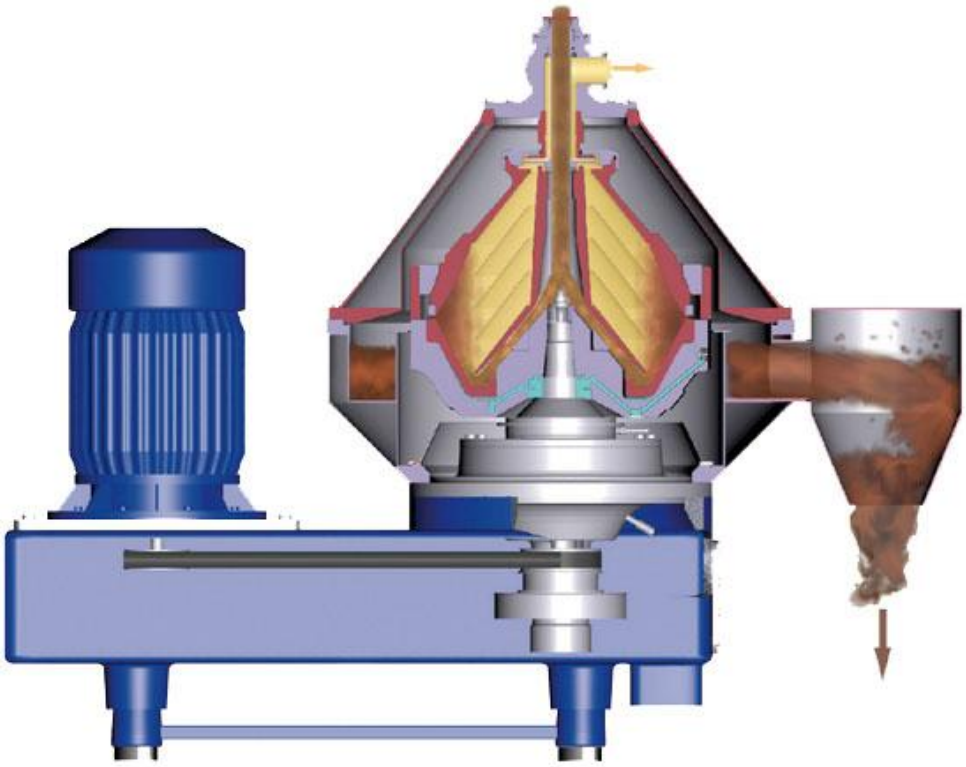


Figure 2.1 Disc Stack Centrifuge

5.2.1 *DEVICE DETAILS & SPECIFICATIONS*

Conical plate centrifuge (also known as disc bowl centrifuge or disc stack separator) is a type of centrifuge that has a series of conical discs which provides a parallel configuration of centrifugation spaces.

The conical plate centrifuge is used to remove solids (usually impurities) from liquids or to separate two liquid phases from each other by means of an enormously high centrifugal force. The denser solids or liquids which are subjected to these forces move outwards towards the rotating bowl wall while the less dense fluids moves towards the center. The special plates (known as disc stacks) increase the surface settling area which speeds up the separation process. Different stack designs, arrangements and shapes are used for different processes depending on the type of feed present. The concentrated denser solid or liquid is then removed continuously, manually or intermittently, depending on the design of the conical plate centrifuge. This centrifuge is very suitable for clarifying liquids that have small proportion of suspended solids

There is 3 Designs for the disc Stack Centrifuge

In our case, we need to make mass production of substance in big scale and without stopping to clean the disc stack centrifuge from solids so we're going to use the Self-Cleaning Centrifuge.

The Self Cleaning Centrifuge has a Movable plate that operates by the pressure of water underneath it,

When the water is sank, the movable plate goes down to open small nozzles for the solids to pass outside. When all the solids are ejected, water is re-flooded beneath the movable plate to make it goes up and block the nozzles.

We should come up with something similar to this:

Utilities consumption	
Electric power	max. 46 kW
Operating liquid during discharge	10 l/h
Cooling water, jacket	300 l/h
Cooling water, oil	80 l/h
Sealing liquid	100 l/h
Flushing liquid, per discharge	25–30 l
ATEX design codes	
BD 95X: EX II 2 G T4 X for zone 1 & 2	Inert gas design
BD 95Y: EX II 3 G T4 X for zone 2	Electrically protected
Material data	
Bowl body, hood and lock ring	s.s. 1.4418
Frame top part and hood	s.s. 1.4401 UNS 31600
Frame bottom part	Cast iron
Gaskets and O-rings	Fluorocarbon rubber

Technical specifications	
Throughput capacity	max. 52 m³/h
Bowl speed	4,300 rpm
Bowl volume	66 l
Sludge space	17 l
Motor speed synchron. 60.7 Hz	1,821 rpm
Motor power installed	52 kW
Starting time	6–8 min
Stopping time without brake	80 min
Inlet pressure at 46 m³/h	300 kPa
Outlet pressure, methyl ester phase	min. 200 kPa
Outlet pressure, heavy phase	800 kPa
Sound pressure	78 dB(A)
Overhead hoist lifting capacity	min. 1,200 kg

Table2.1 Disc Stack Centrifuge Specifications. See also figure 2.1 and 2.2

1. Feed valve
2. Centrate out
3. Pneumatic butterfly valve
4. Cooling jacket
5. Bowl
6. Cooling water
7. Axial seal
8. Check valve
9. Spindle cartridge
10. Oil filler and gauge
11. Flat driving belt
12. Cyclone with recirculation tube
13. Solids pockets
14. Pneumatically operated slide
15. Motor
16. Unibase frame
17. Foundation

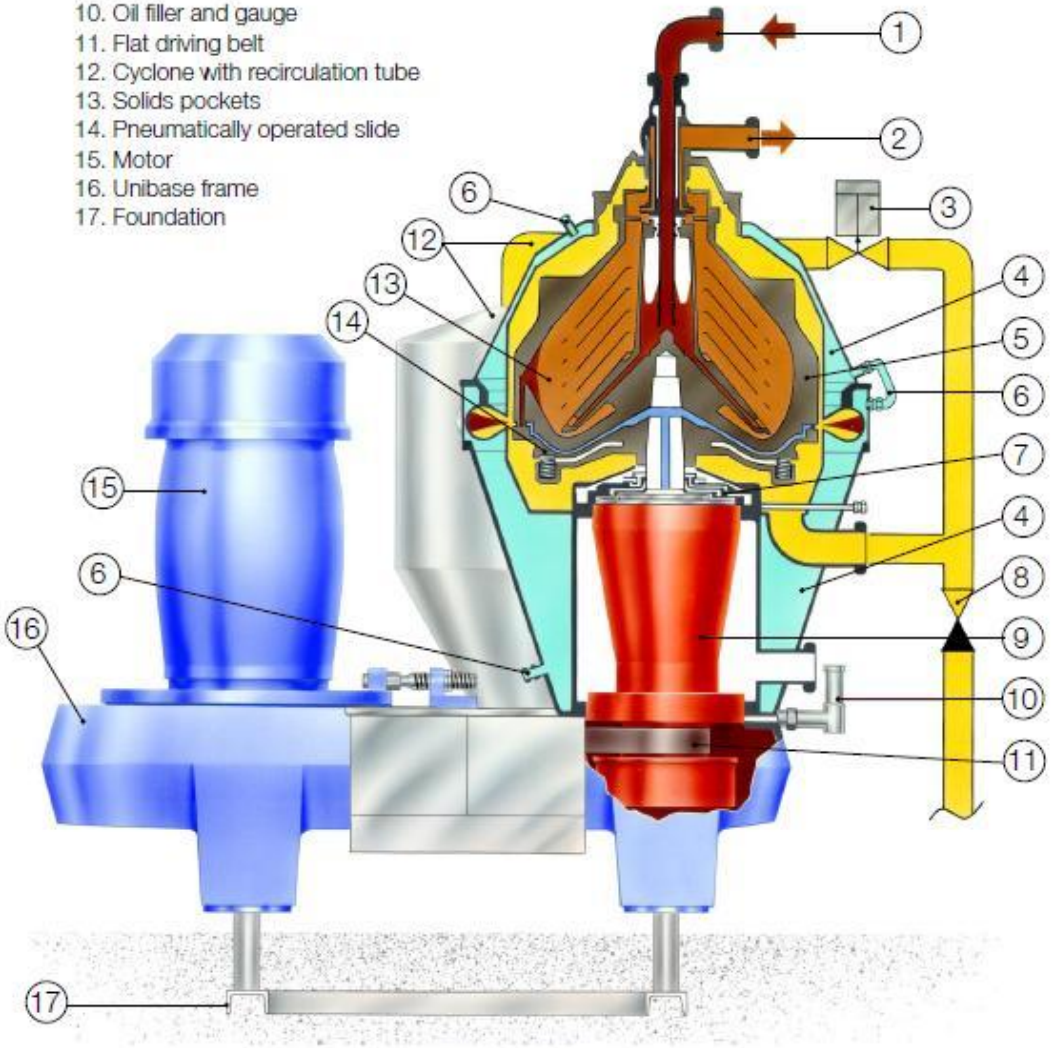
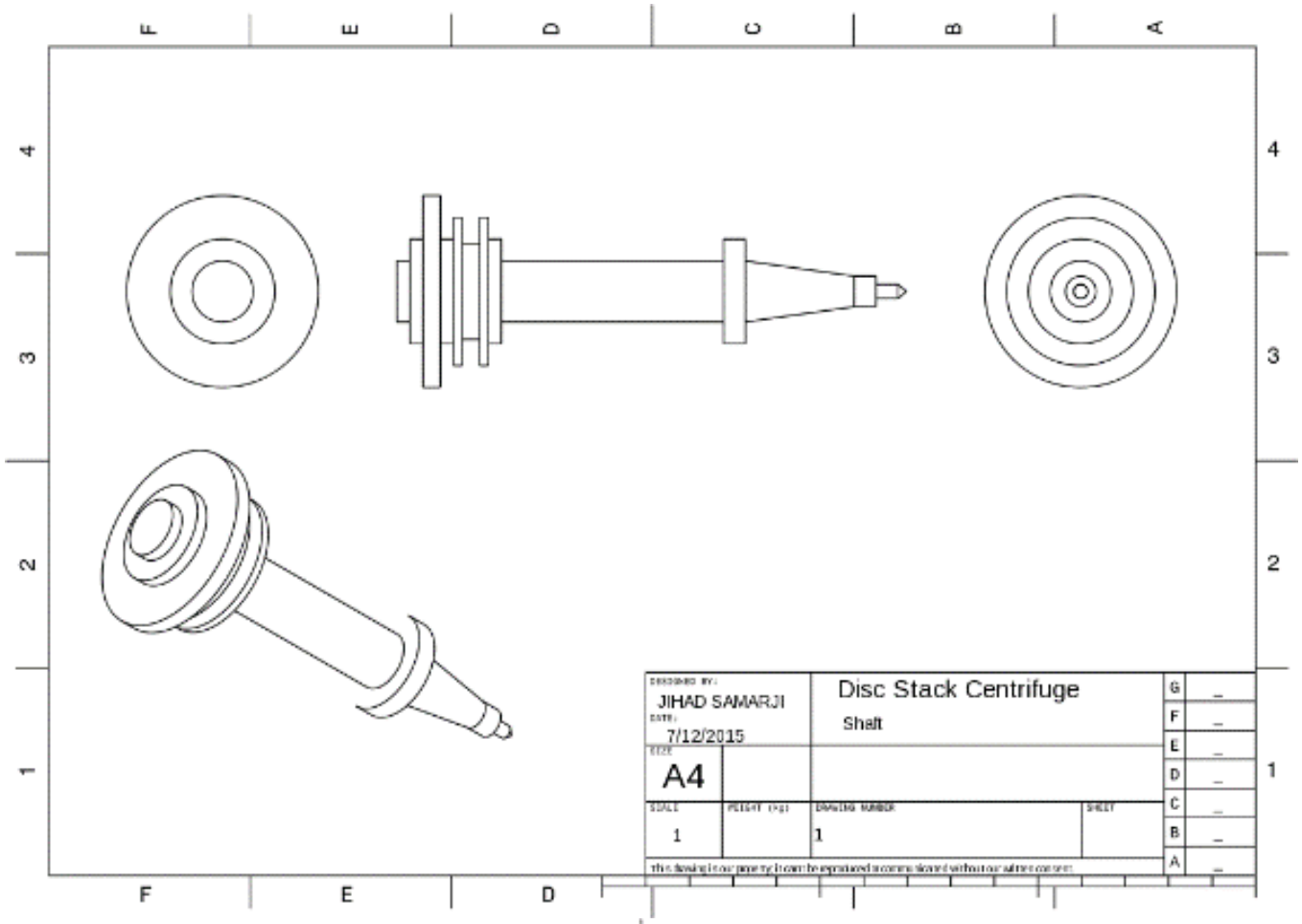
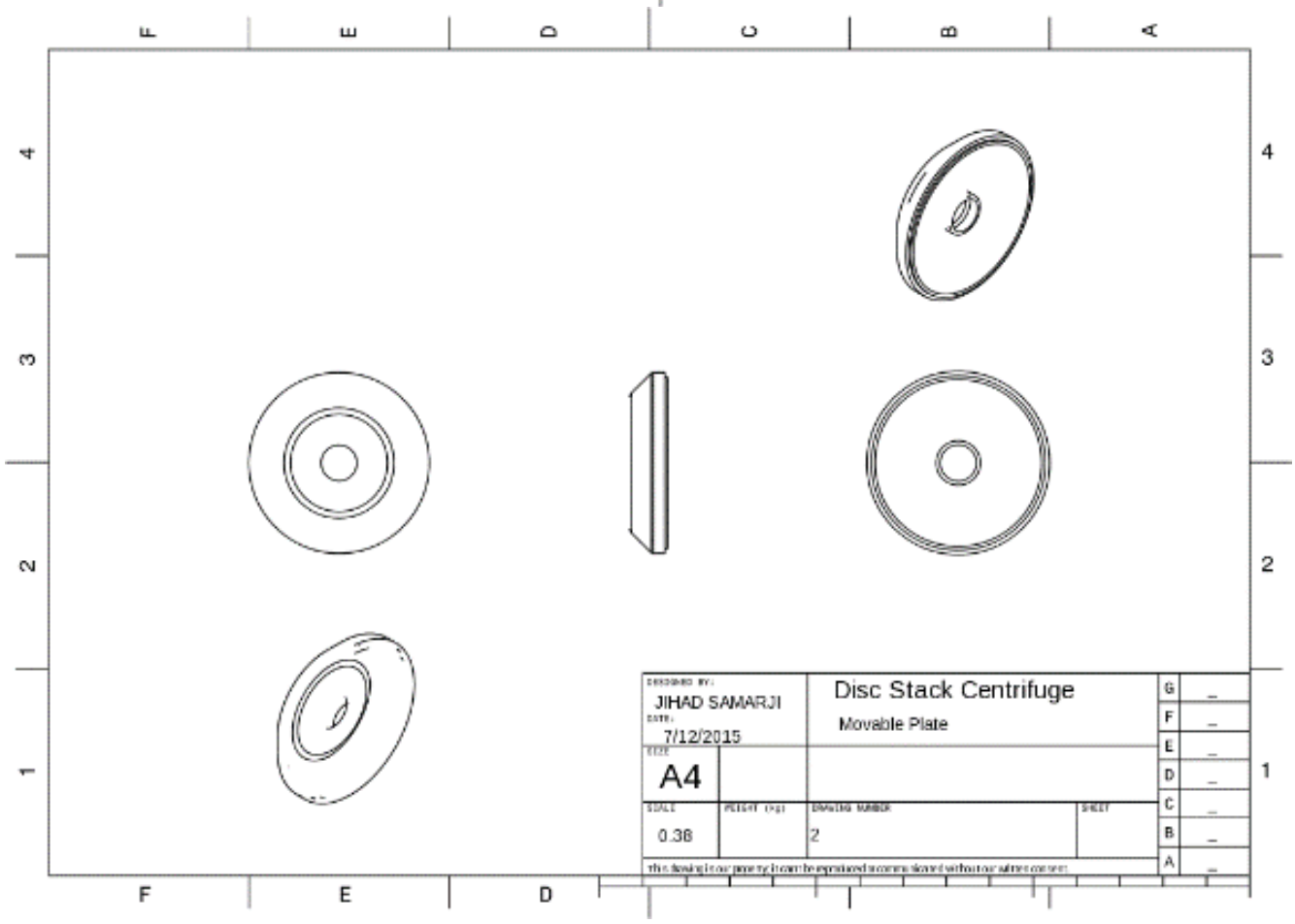
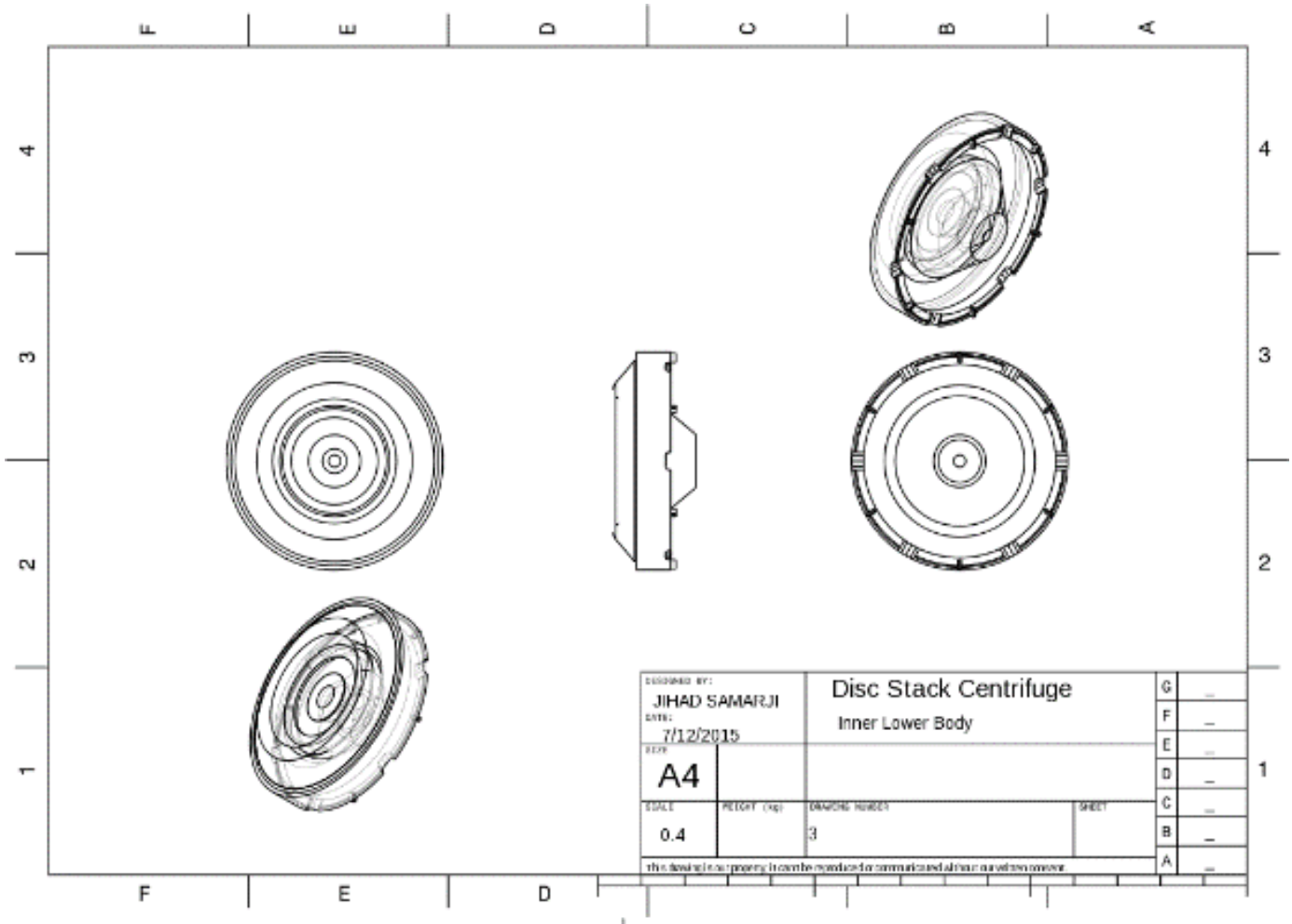


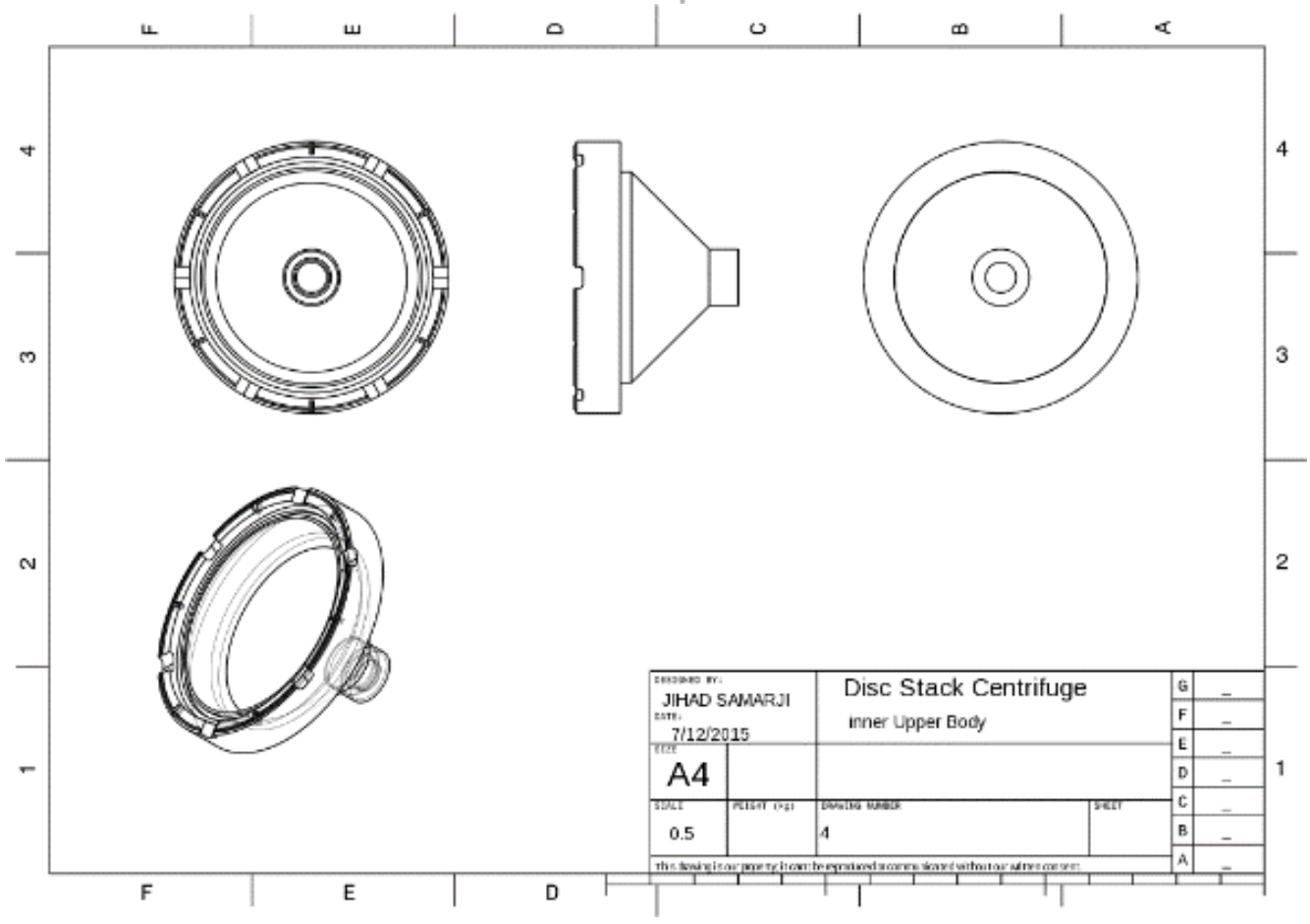
Figure 2.2 Disc Stack Centrifuge Annotation

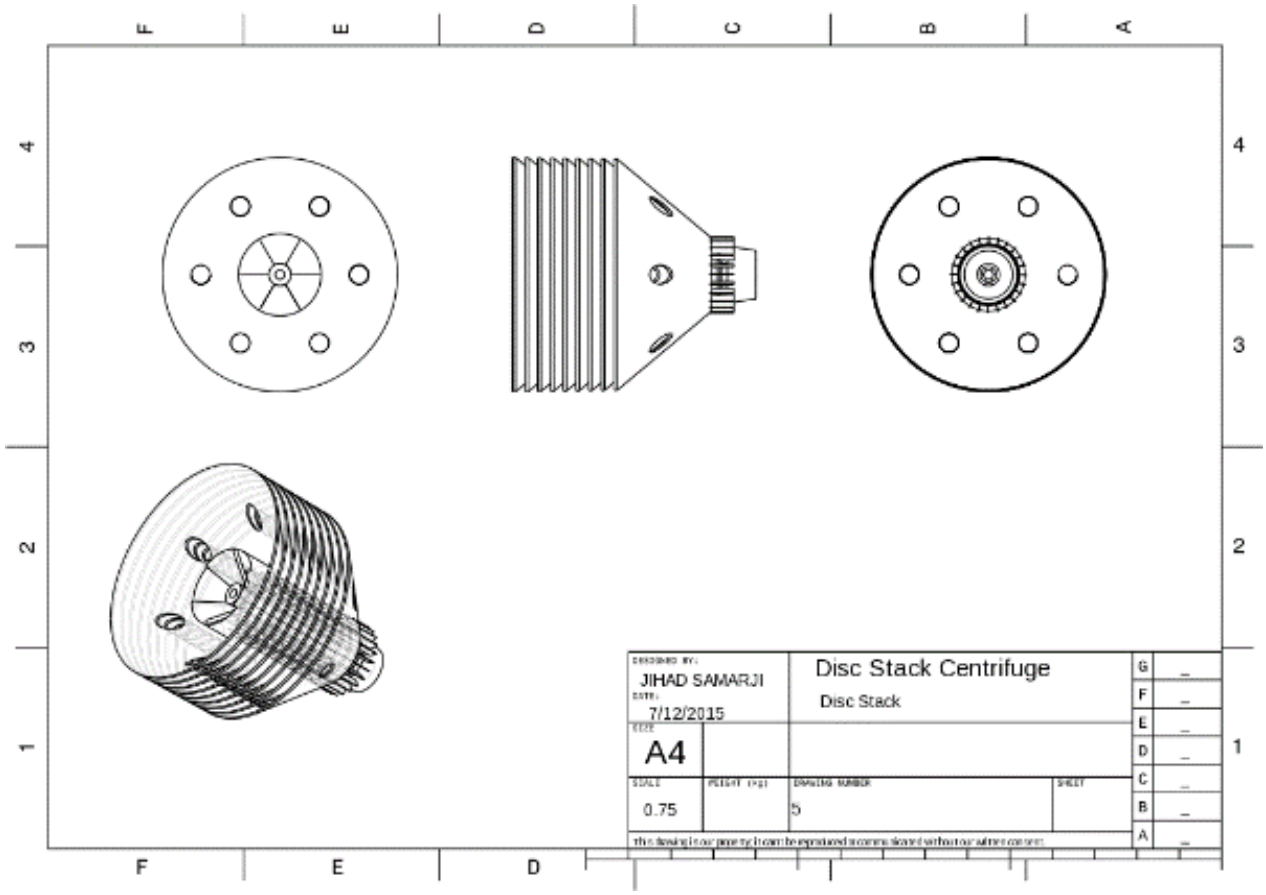
5.2.2 ***PART DIMENSIONS & DESIGN***

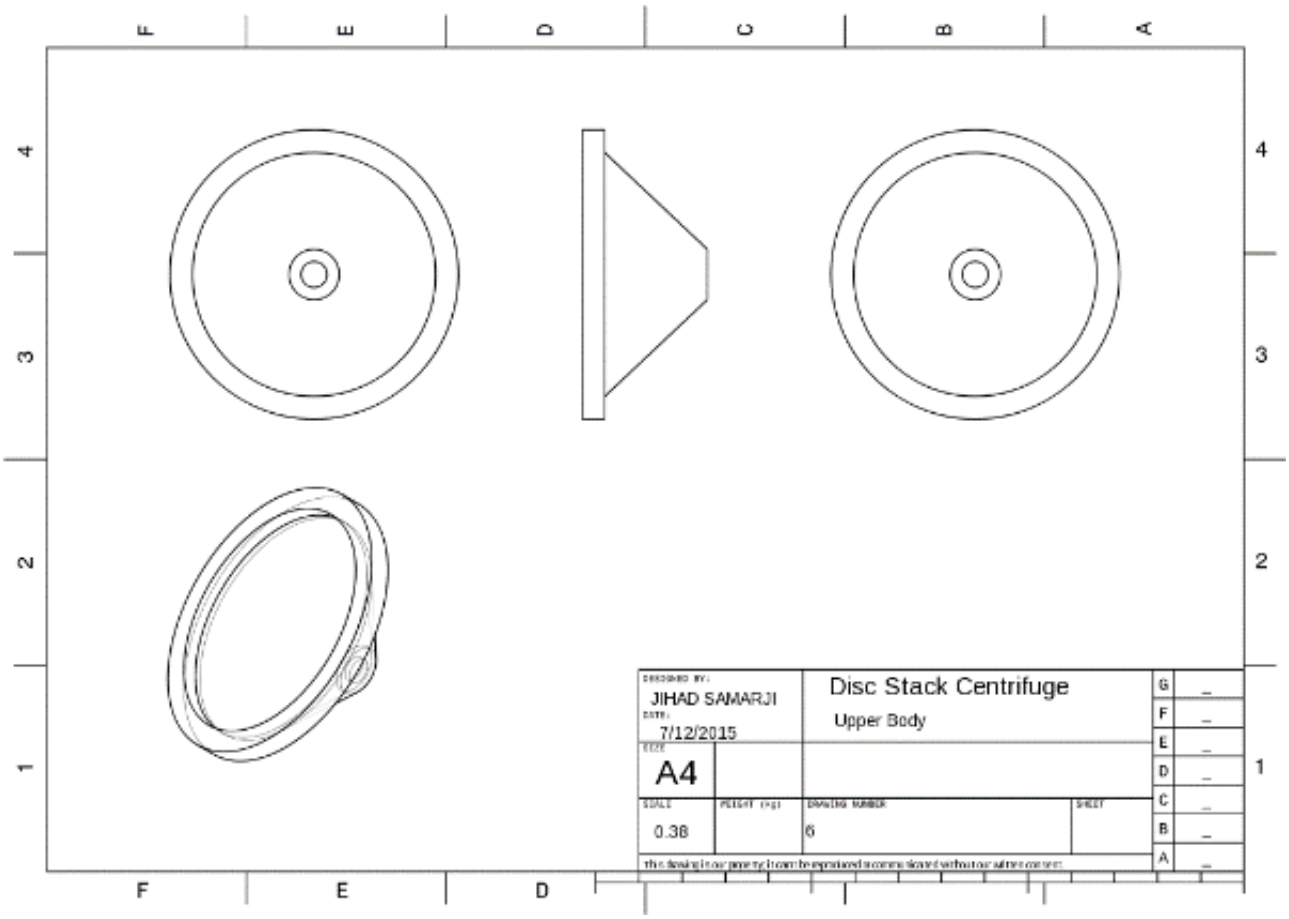


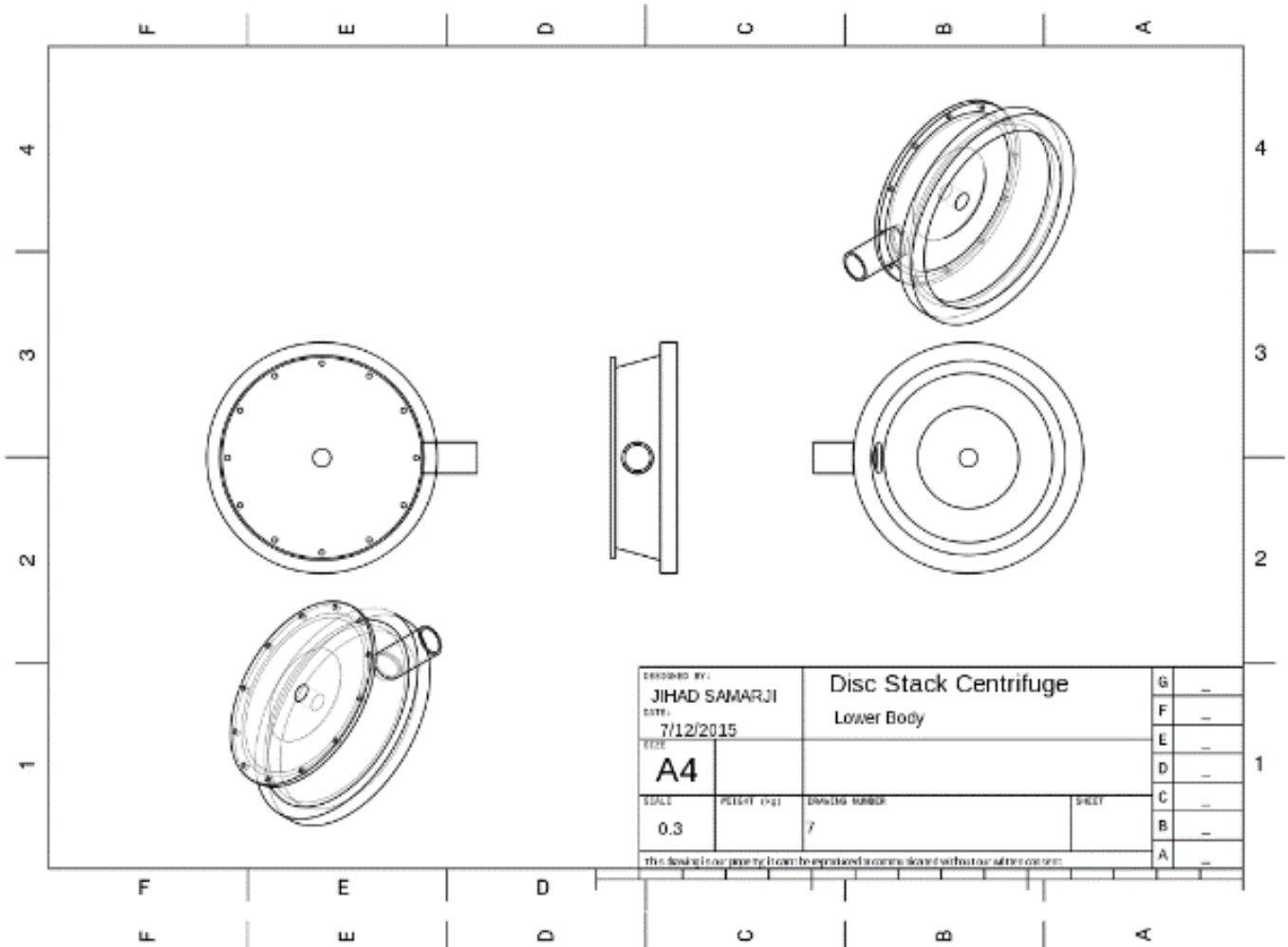


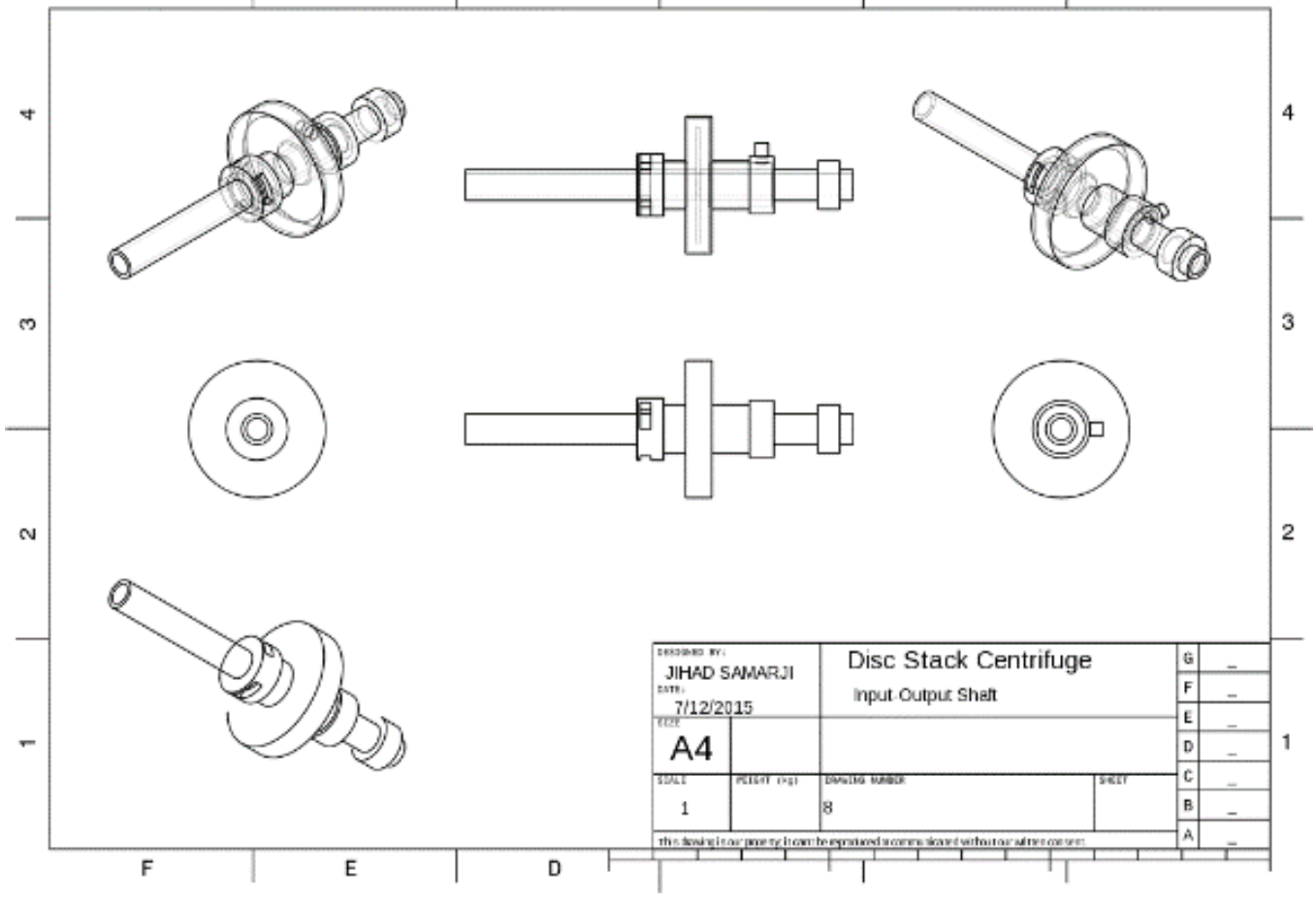

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/12/2015
SIZE
A4
SCALE
0.4
WEIGHT (kg)
DRAWING NUMBER
3
SHEET
Disc Stack Centrifuge
Inner Lower Body

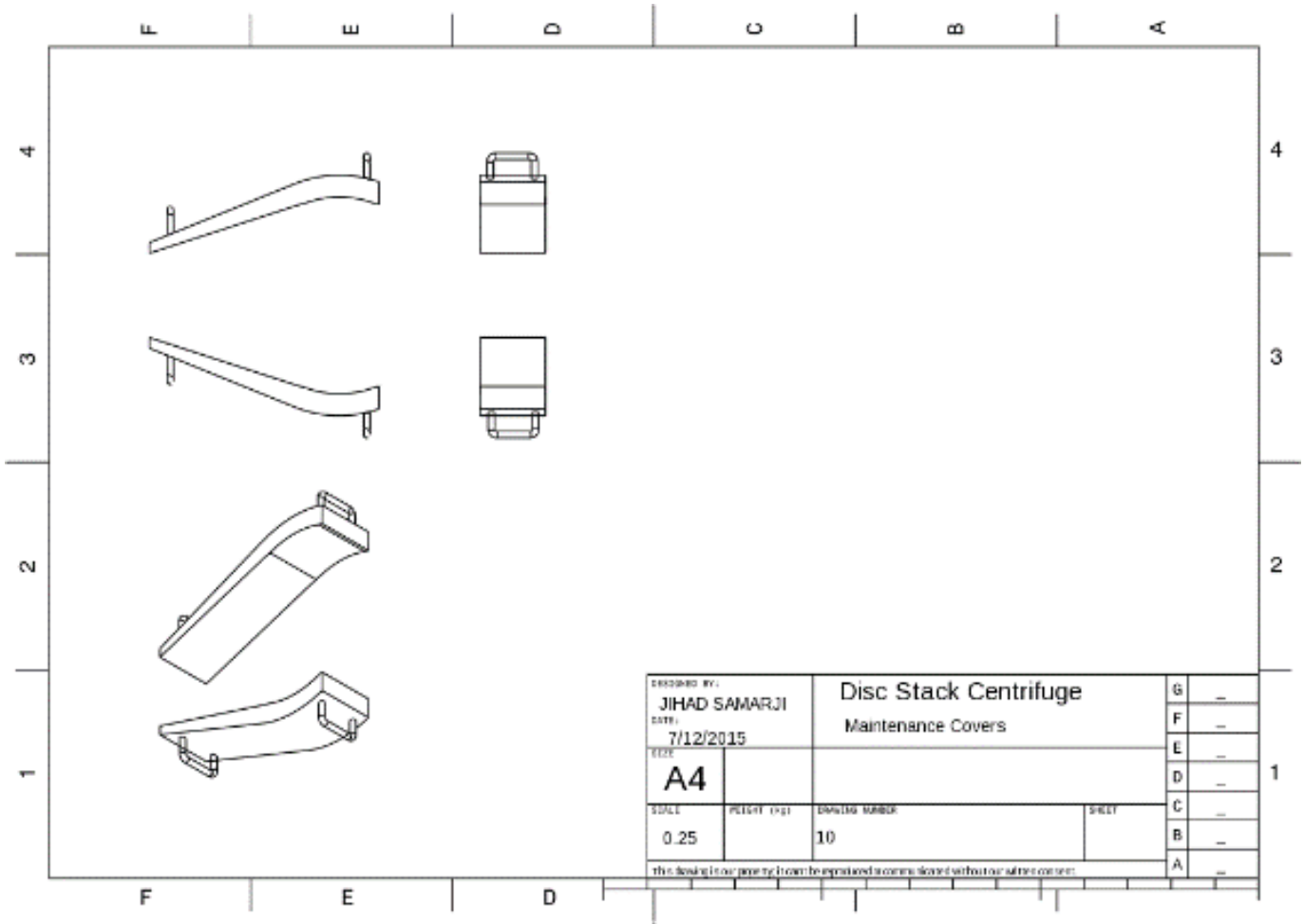

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/12/2015
SIZE
A4
SCALE
0.5
WEIGHT (kg)
DRAWING NUMBER
4
SHEET
Disc Stack Centrifuge
inner Upper Body


DESIGNED BY:
JIHAD SAMARJI
DATE:
7/12/2015
SIZE
A4
SCALE
0.75
WEIGHT (kg)
DRAWING NUMBER
5
SHEET
Disc Stack Centrifuge
Disc Stack

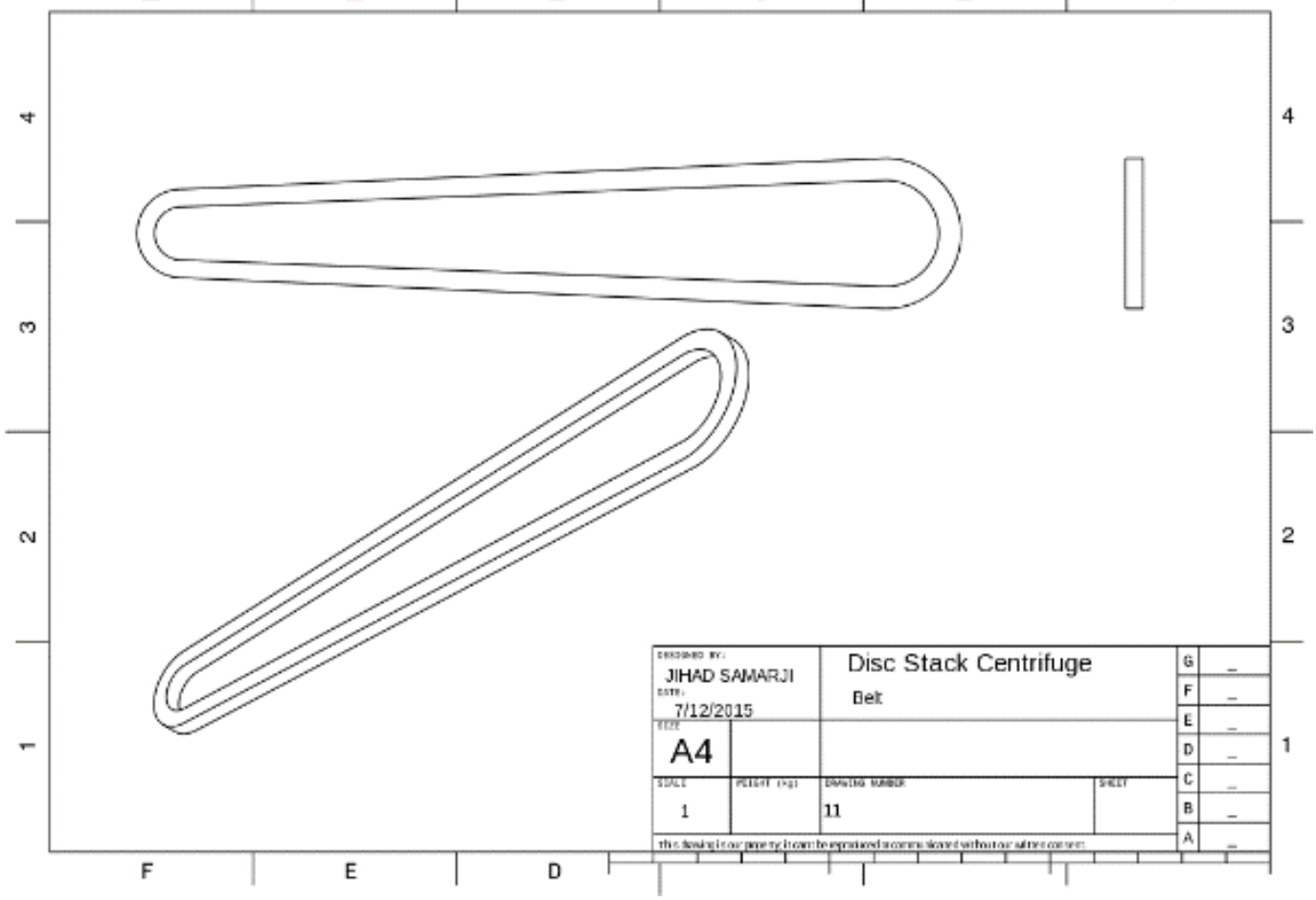

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/12/2015
SIZE
A4
SCALE
0.38
WEIGHT (kg)
DRAWING NUMBER
6
SHEET
Disc Stack Centrifuge
Upper Body


DESIGNED BY:
JIHAD SAMARJI
DATE:
7/12/2015
SIZE
A4
SCALE
0.3
WEIGHT (kg)
DRAWING NUMBER
7
SHEET
Disc Stack Centrifuge
Lower Body

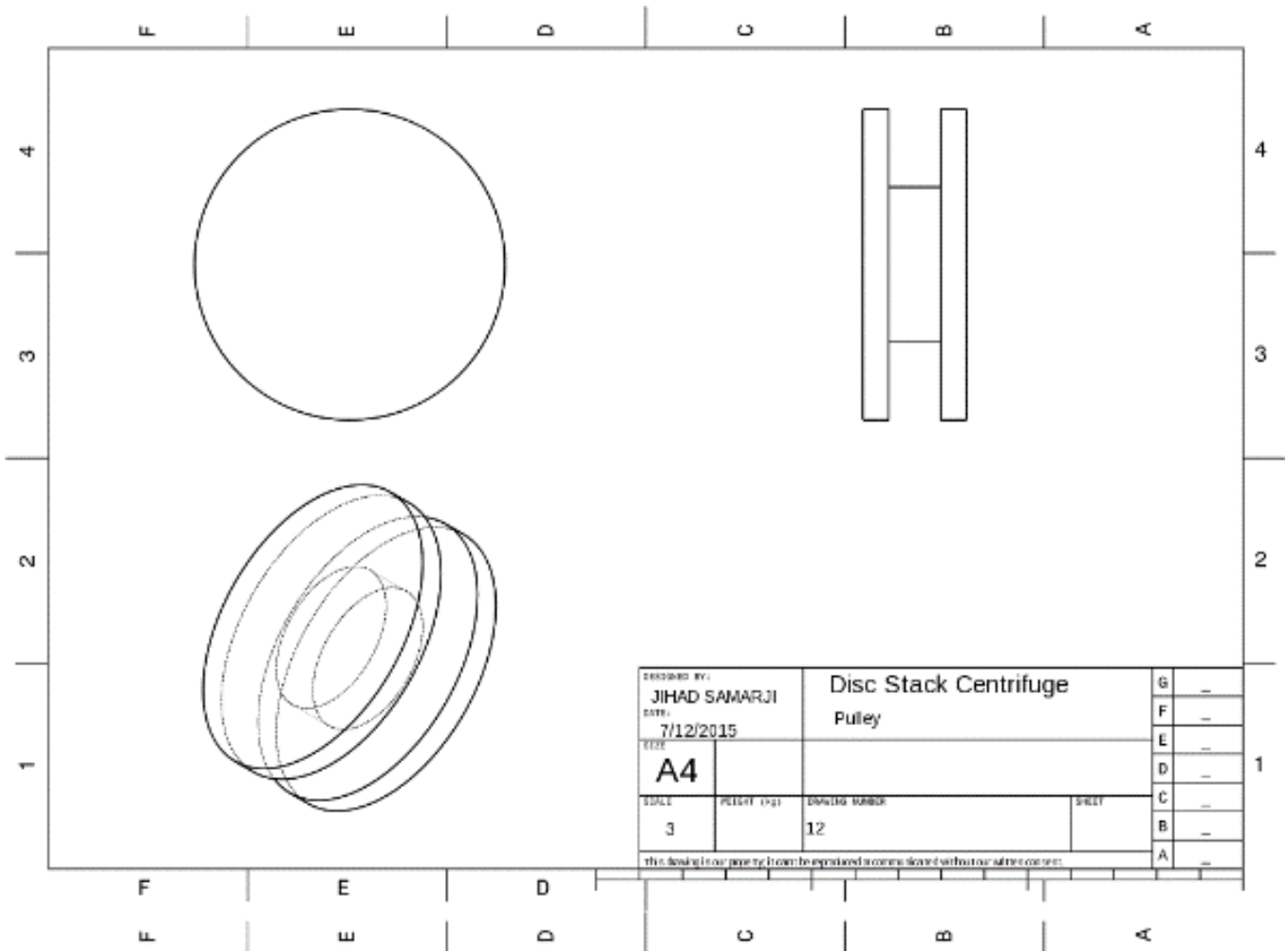

DESIGNED BY:
JIHAD SAMARJI
DATE:
7/12/2015
SIZE
A4
SCALE
1
WEIGHT (kg)
DRAWING NUMBER
8
SHEET
Disc Stack Centrifuge
Input Output Shaft



DESIGNED BY: JIHAD SAMARJI	Disc Stack Centrifuge		
DATE: 7/12/2015	Maintenance Covers		
SIZE: A4			
SCALE: 0.25	WEIGHT (kg):	DRAWING NUMBER: 10	SHEET:

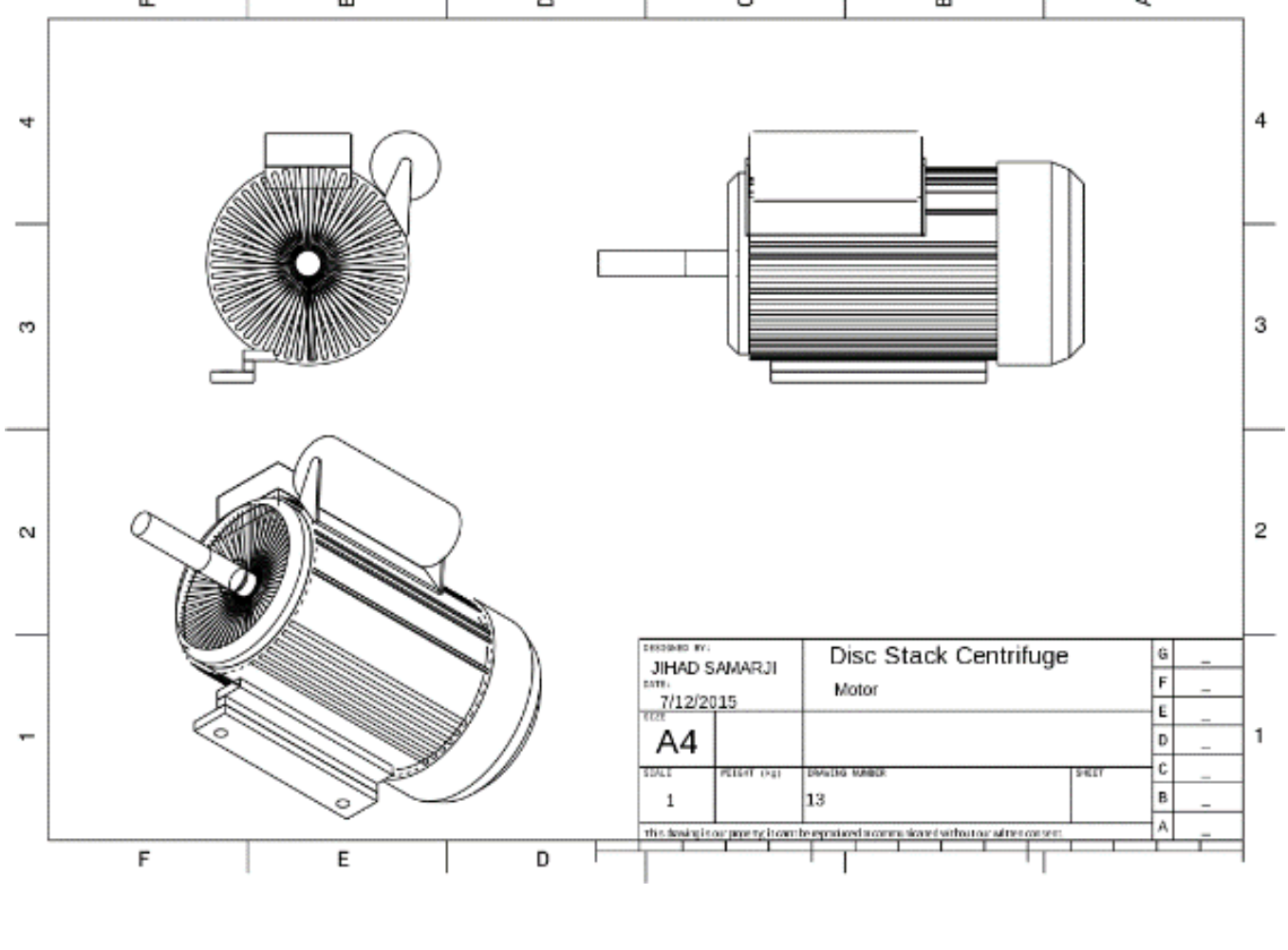


DESIGNED BY: JIHAD SAMARJI	Disc Stack Centrifuge		
DATE: 7/12/2015	Belt		
SIZE: A4			
SCALE: 1	WEIGHT (kg):	DRAWING NUMBER: 11	SHEET:



DESIGNED BY: JIHAD SAMARJI	Disc Stack Centrifuge		
DATE: 7/12/2015	Pulley		
SIZE A4			
SCALE 3	WEIGHT (kg)	DRAWING NUMBER 12	SHEET

this drawing is our property it can't be reproduced in any manner without our written consent



DESIGNED BY: JIHAD SAMARJI	Disc Stack Centrifuge		
DATE: 7/12/2015	Motor		
SIZE A4			
SCALE 1	WEIGHT (kg)	DRAWING NUMBER 13	SHEET

this drawing is our property it can't be reproduced in any manner without our written consent

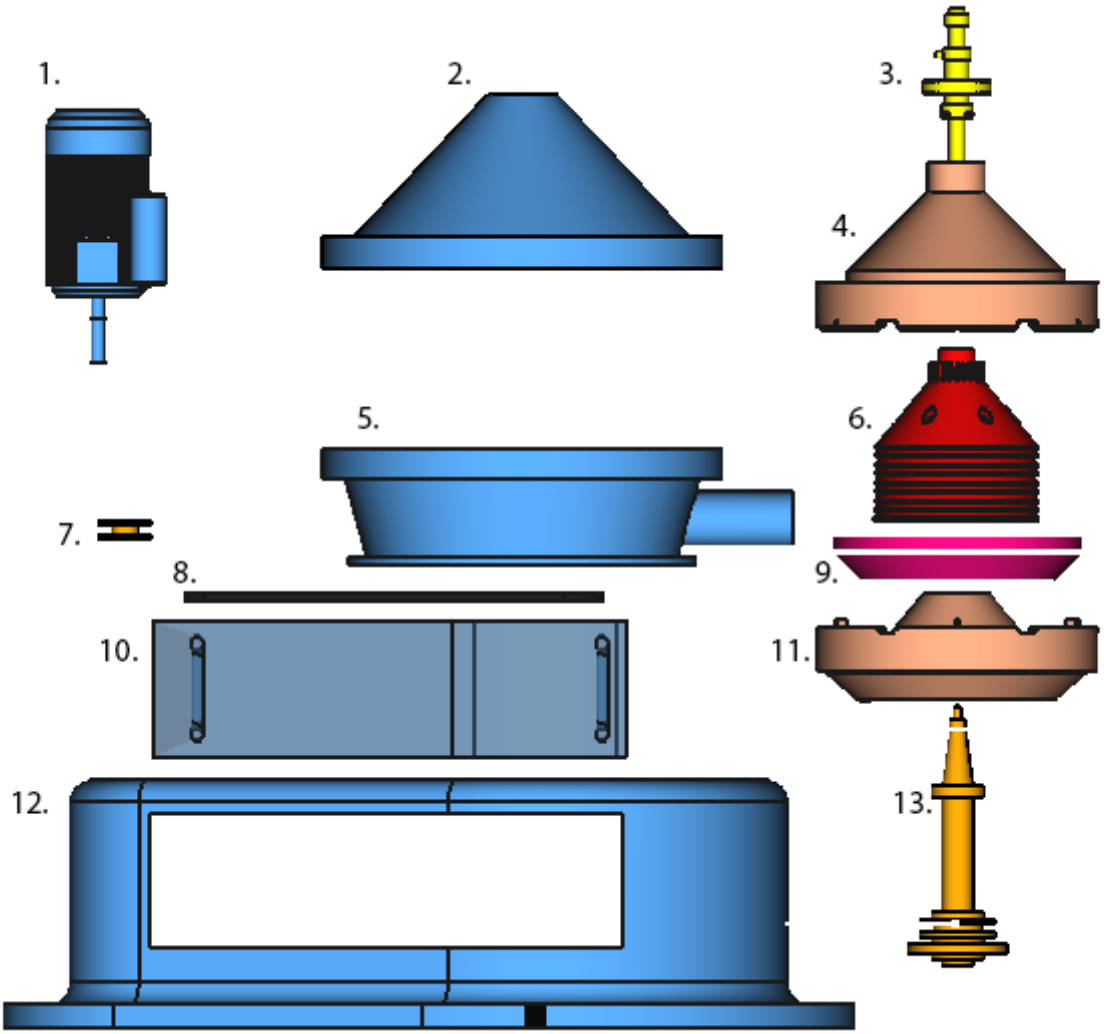
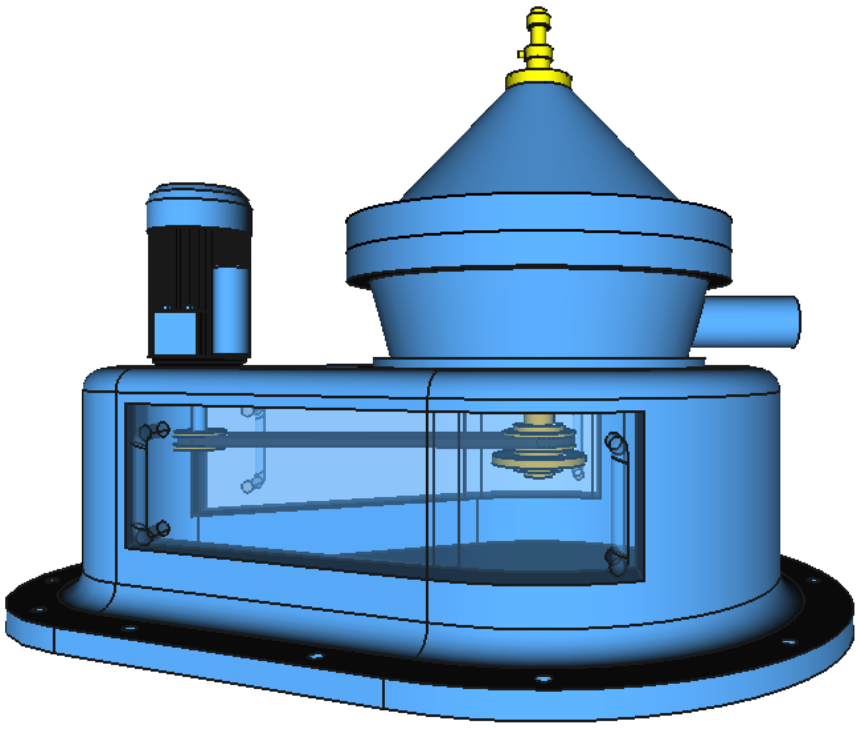


Figure 2.3 Disc Stack Centrifuge Annotations

1. Motor 2.Upper Body 3.Inlet-Outlet Shaft 4.Inner Upper Body 5.Lower Body 6.Disc Stack 7. Pulley 8.Belt 9.Movable Plate 10.Maintenance Cover 11.Inner Lower Body 12.Support 13. Shaft



5.3 Chapter 3: Demonstration and Modelling of the Ultrafiltration.

"To Filtrate a Huge Supply of Substance or Water we need an Ultrafiltration Device"-Jihad Samarji

Figure 3.1 Ultrafiltration Machine

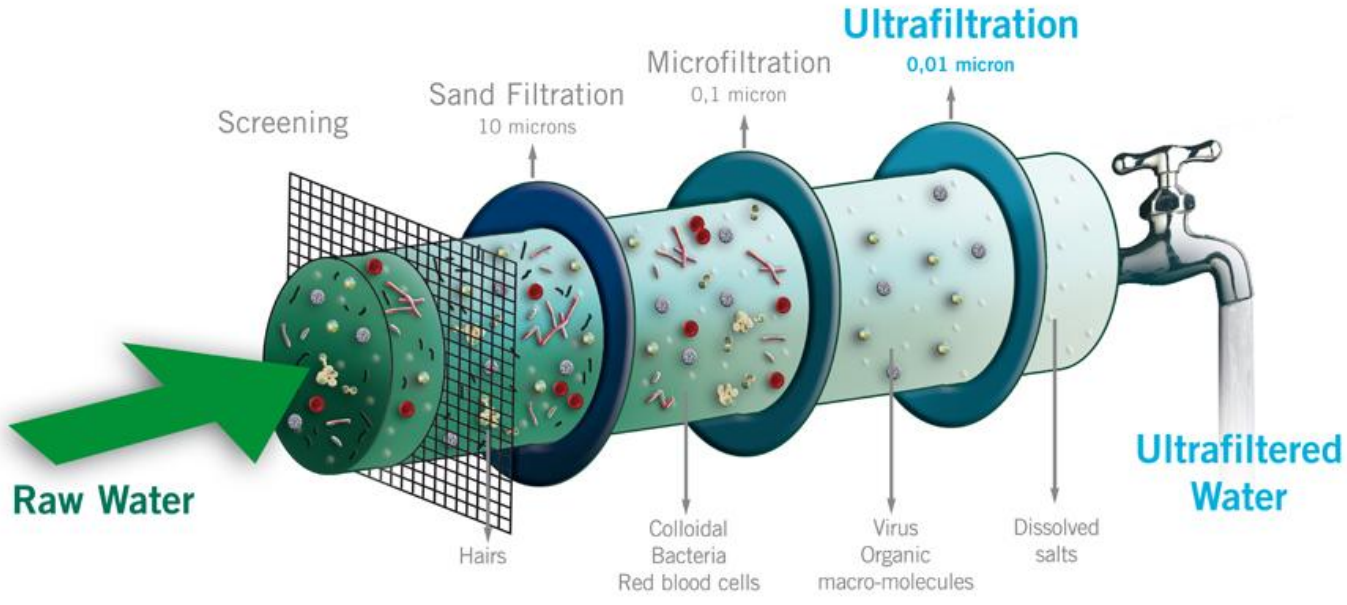


Figure 3.2 a small demonstration about the Ultrafiltration Process

5.3.1 *DEVICE DETAILS & SPECIFICATIONS*

Ultrafiltration (UF) is a variety of membrane filtration in which forces like pressure or concentration gradients lead to a separation through a membrane. Suspended and solutes of high molecular weight are retained in the so-called retentive, while water and low molecular weight solutes pass through the membrane in thepermeate. This separation process is used in industry and research for purifying and concentrating macromolecular (103 - 106 Da) solutions, especially protein solutions. Ultrafiltration is not fundamentally different from microfiltration. Both of these separate based on size exclusion or particle capture. It is fundamentally different from membrane gas separation, which separate based on different amounts of absorption and different rates of diffusion. Ultrafiltration membranes are defined by the molecular weight cut-off(MWCO) of the membrane used. Ultrafiltration is applied in cross-flow or dead-end mode.

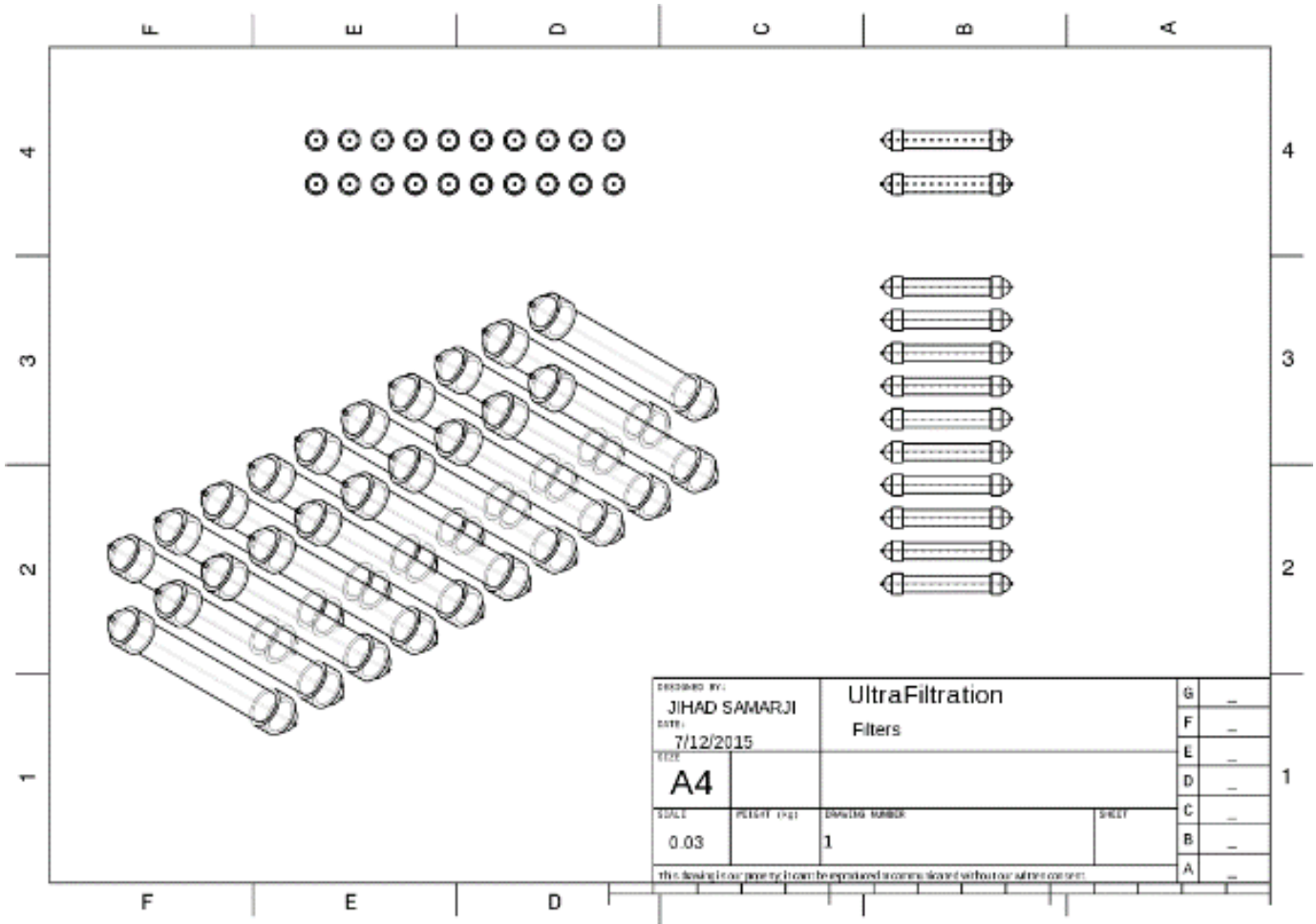
Ultrafiltration is used in Industries such as chemical and pharmaceutical manufacturing, food and beverage processing, and waste water treatment, employ ultrafiltration in order to recycle flow or add value to later products. Blood dialysis also utilizes ultrafiltration.

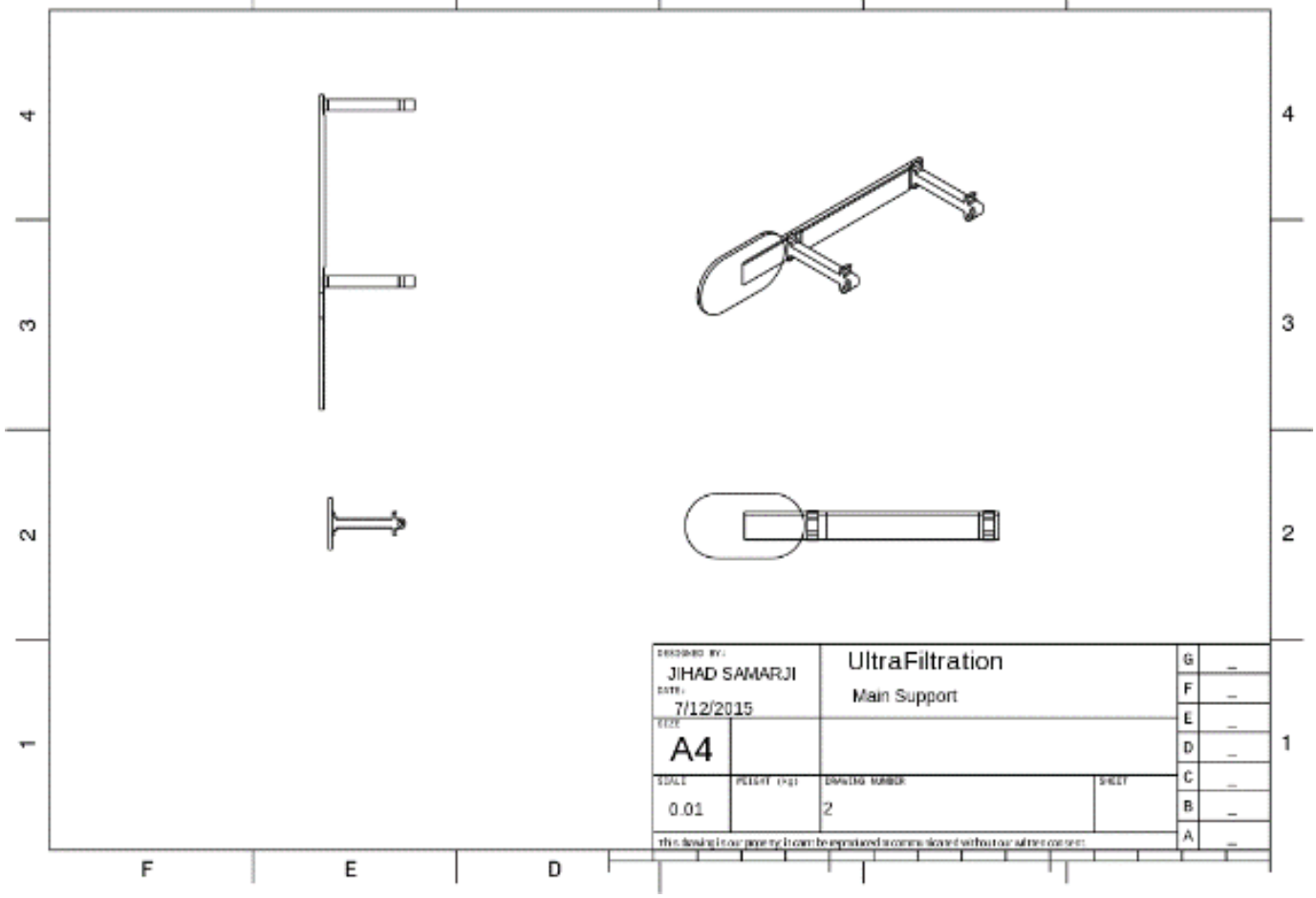
In our case we're going to use the ultrafiltration device for Protein concentration. See Figure.3.1 and 3.2

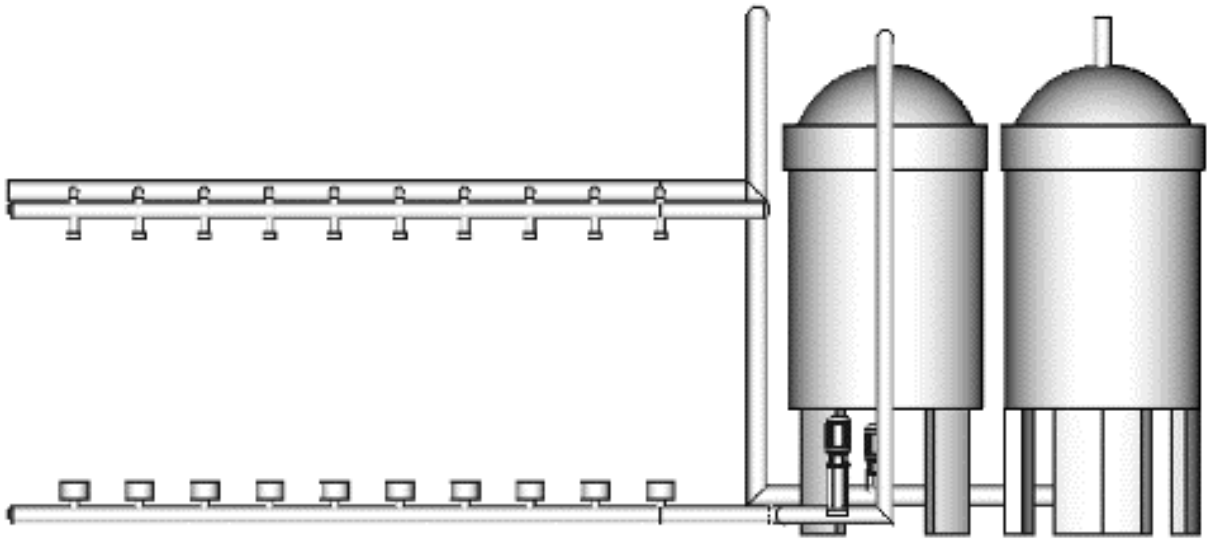
Foulant	Reagent	Time and Temperature	Mode of Action
Fats and oils, proteins, polysaccharides, bacteria	0.5M NaOH with 200 ppm Cl2	30-60 min 25-55 °C	Hydrolysis and oxidation
DNA, mineral salts	0.1M – 0.5M acid (acetic, citric, nitric)	30-60 min 25-35 °C	Solubilization
Fats, oils, biopolymers, proteins	0.1% SDS, 0.1% Triton X-100	30 min – overnight 25-55 °C	Wetting, emulsifying, suspending, dispersing
Cell fragments, fats, oils, proteins	Enzyme detergents	30 min – overnight 30 – 40 °C	Catalytic breakdown
DNA	0.5% DNAase	30 min – overnight 20 – 40 °C	Enzyme hydrolysis

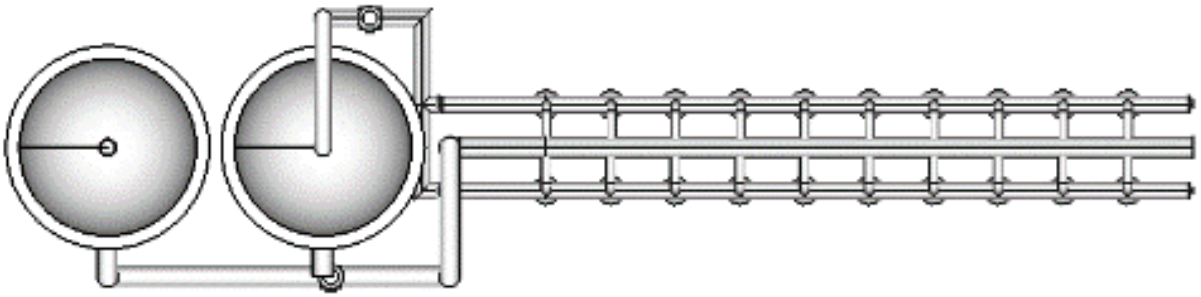
Table 3.2 SUMMARY OF COMMON TYPES OF FOULING AND THEIR RESPECTIVE CHEMICAL TREATMENTS

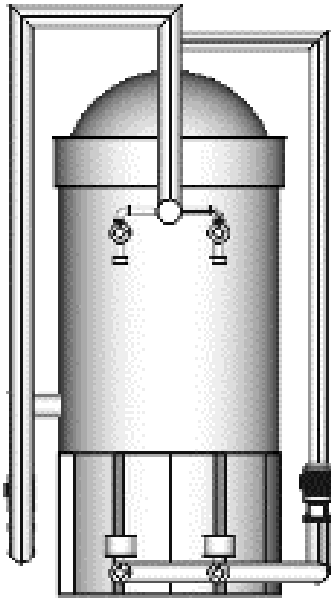
5.3.2 *PART DIMENSIONS & DESIGN*

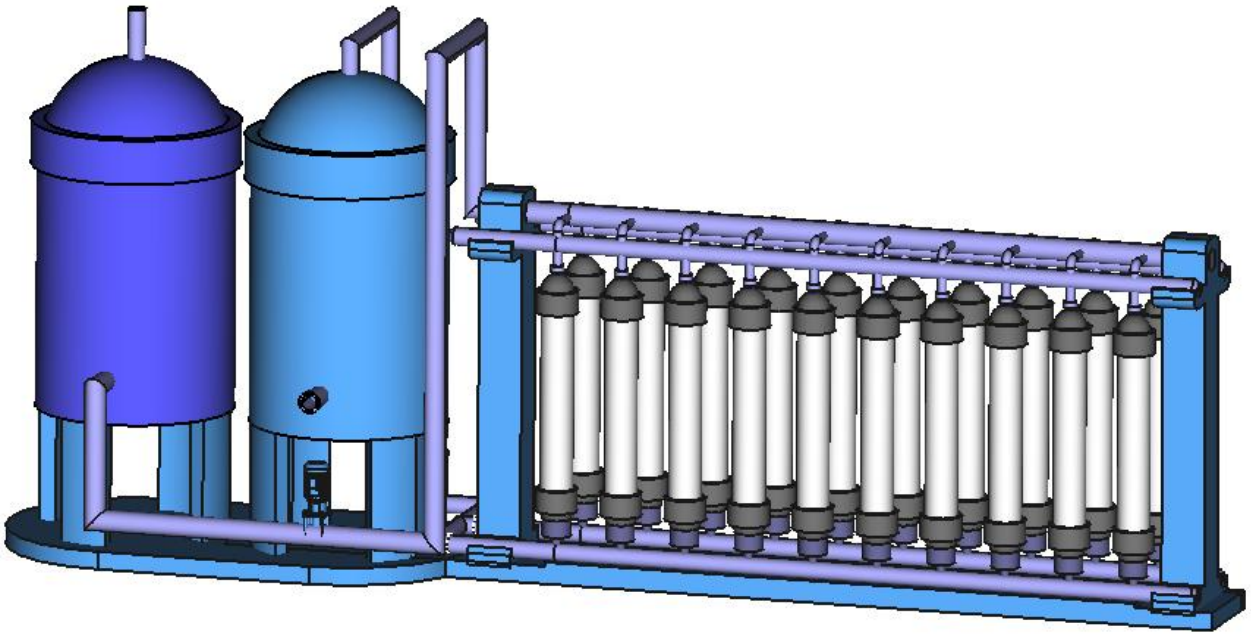












The Ultrafiltration is Simple, no need for further dimensions nor annotations. It contains 2 Tanks, one for filtrated substance and the other for non-filtrated substance which is connected to a set of filters. The device also contains 2 pumps to boost the substance into the pipes. For more information about the ultrafiltration process, see figure.5

5.4 **Demonstration and Modelling of the Chromatography Device**

-ZaherChendeb-

5.4.1 ***DEVICE DETAILS & SPECIFICATIONS***

AxiChrom<Chromatography>

The AxiChrom column platform is a revolutionary concept in column chromatography that simplifies column handling at all scales from process development to full-scale production. AxiChrom columns introduce three key features – Intelligent Packing, Intuitive handling, and Predictable scale-up – that together make process chromatography easier, safer, and more efficient.

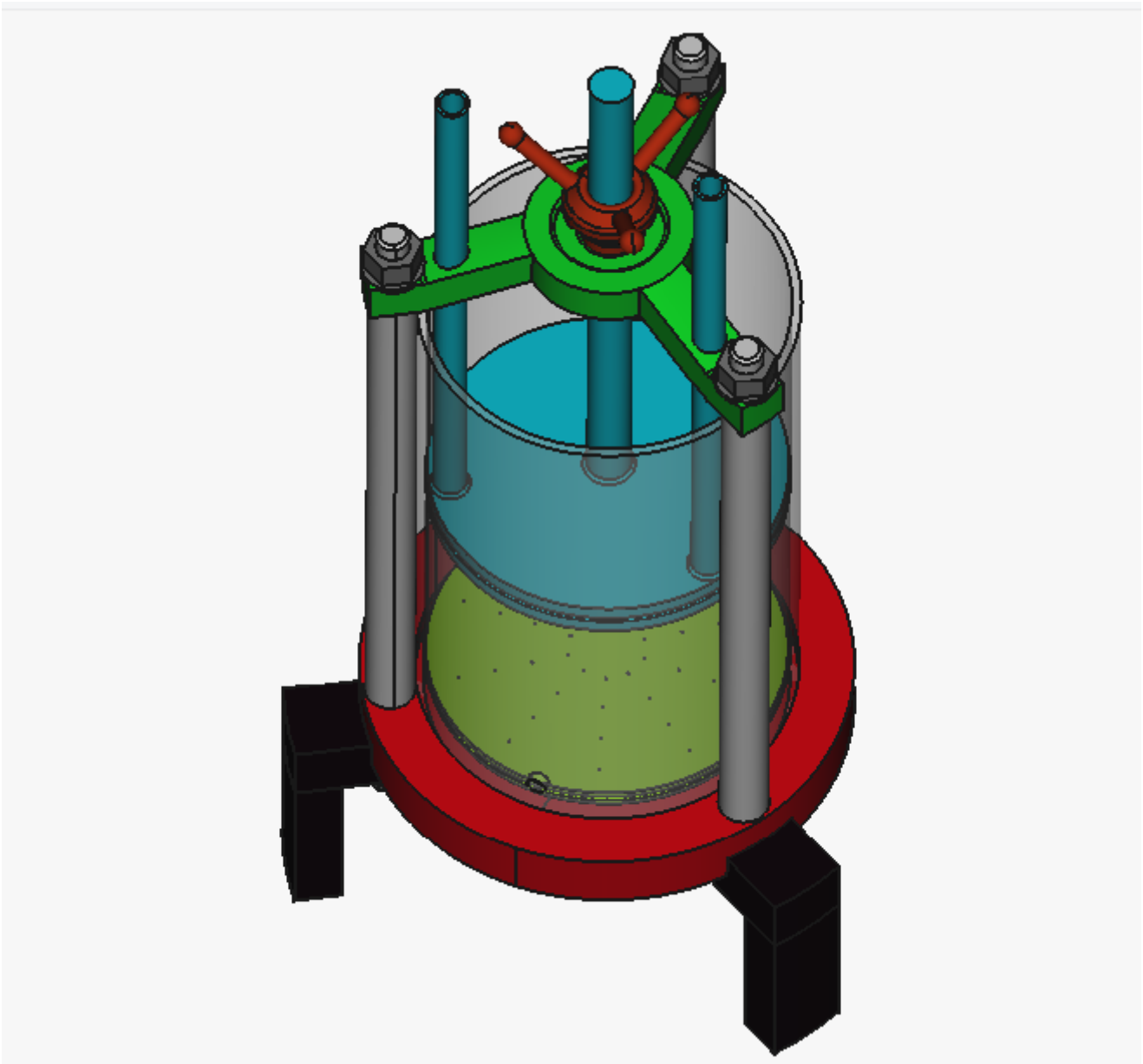
The AxiChrom process column family has been designed to deliver reproducible results from process development to production scales. This is facilitated by the innovative Intelligent Packing where UNICORN™ software, ÄKTA™ systems and AxiChrom columns work together to facilitate a convenient operation for packing of the bed via axial compression.

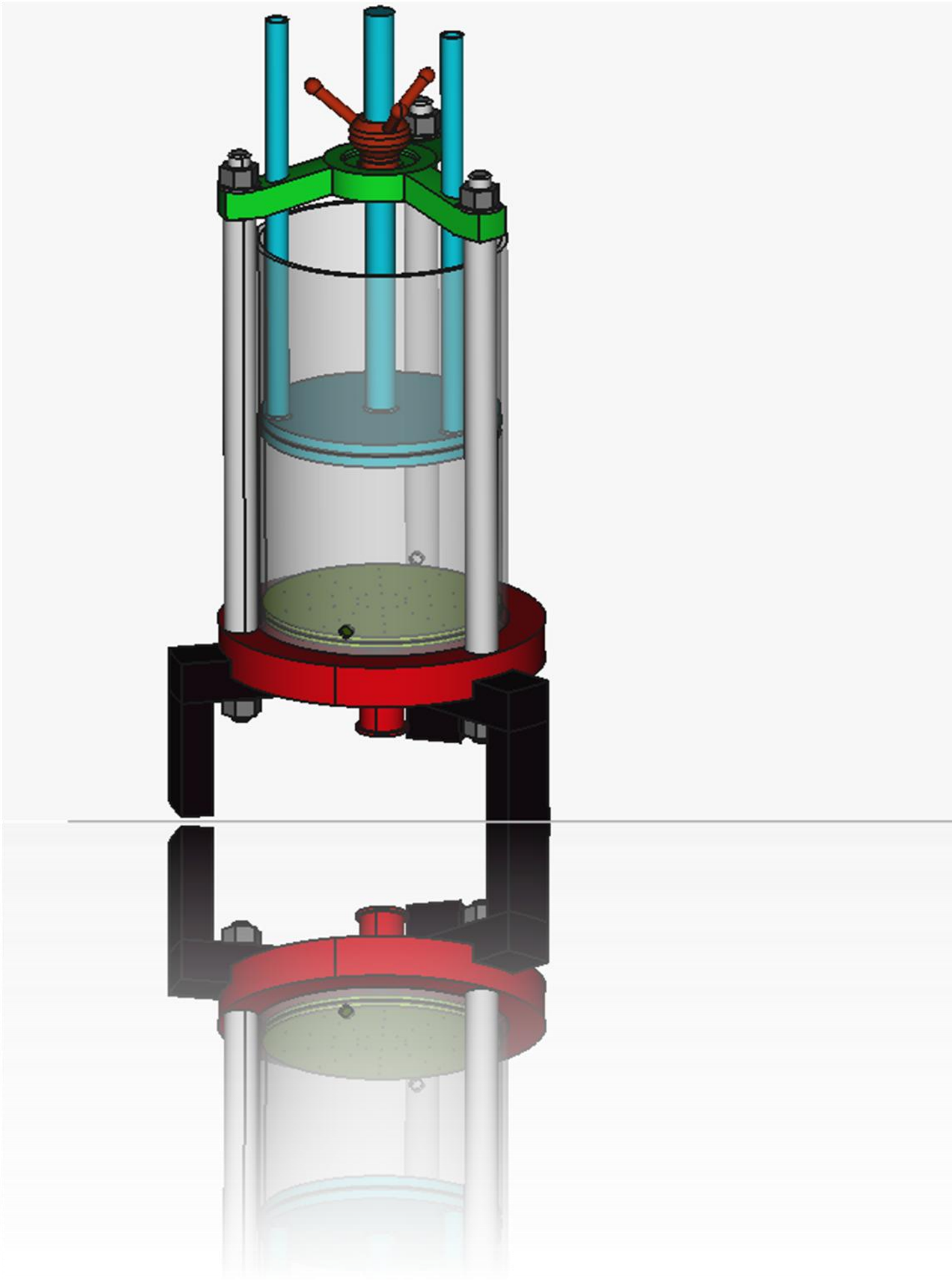
The Intended use of The AxiChrom family of process columns has been designed for low pressure chromatographic separation of biomolecules such as proteins, peptides and oligonucleotides in GMP-regulated environments.

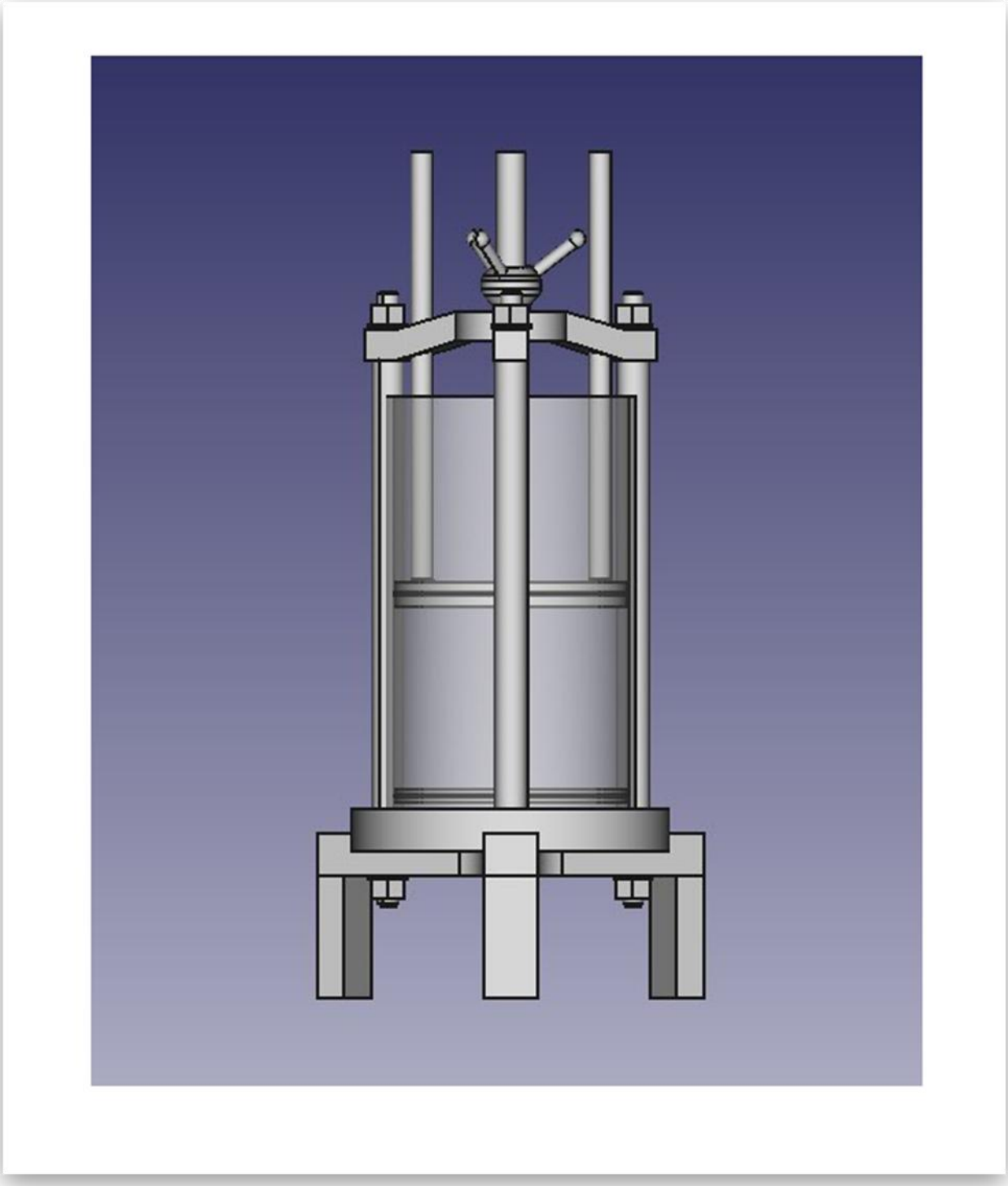
The AxiChrom columns are intended for production use only and should not be used for diagnostic purposes in any clinical or in vitro procedures. The columns are not suitable for operation in a

potentially explosive atmosphere or for handling flammable liquids. If the columns are used for purposes other than those specified in the user documentation, safe operation and the protection provided by the system may be impaired.

5.4.2 *PART DIMENSIONS & DESIGN*



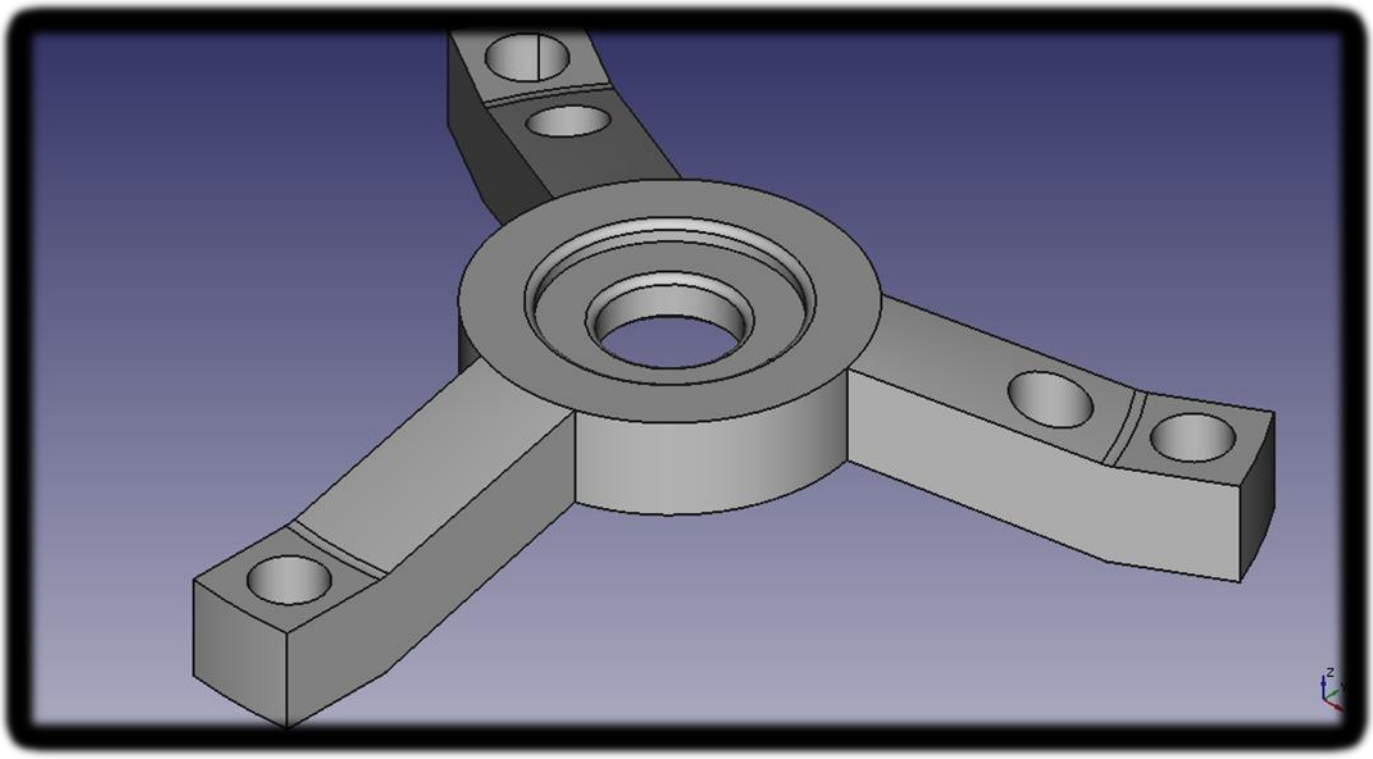


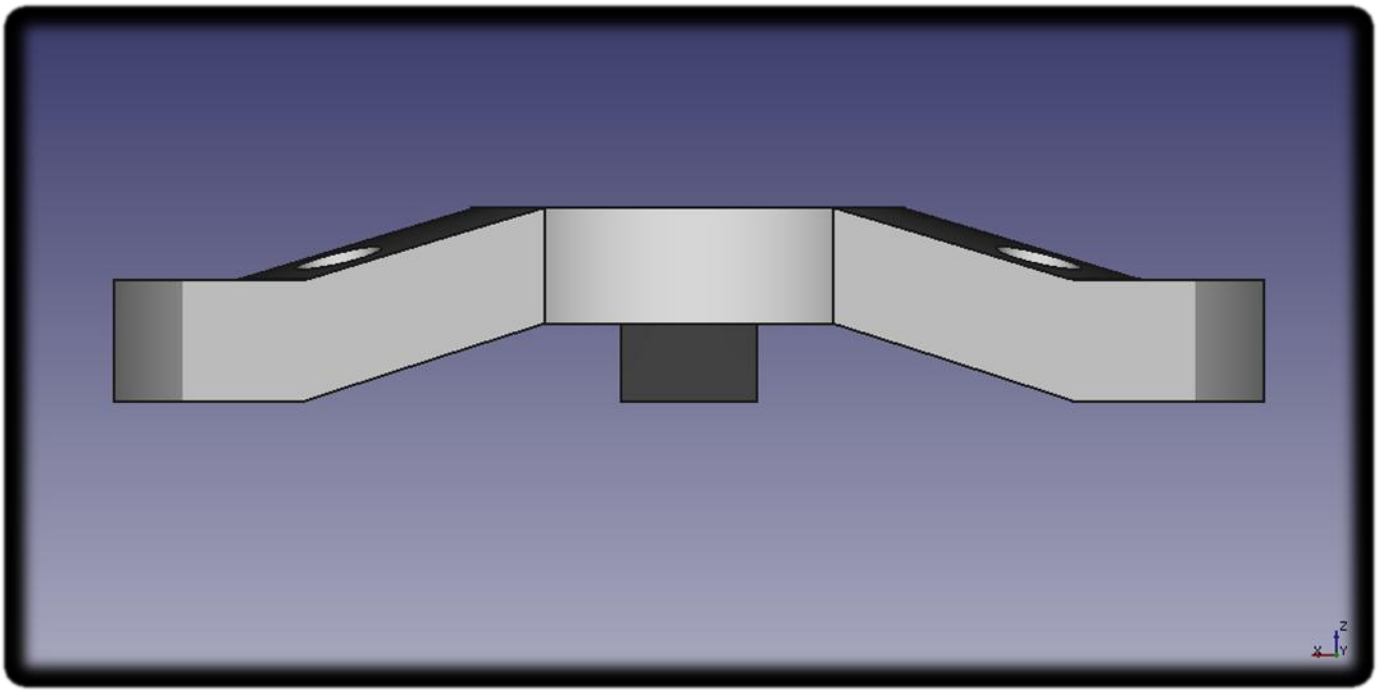


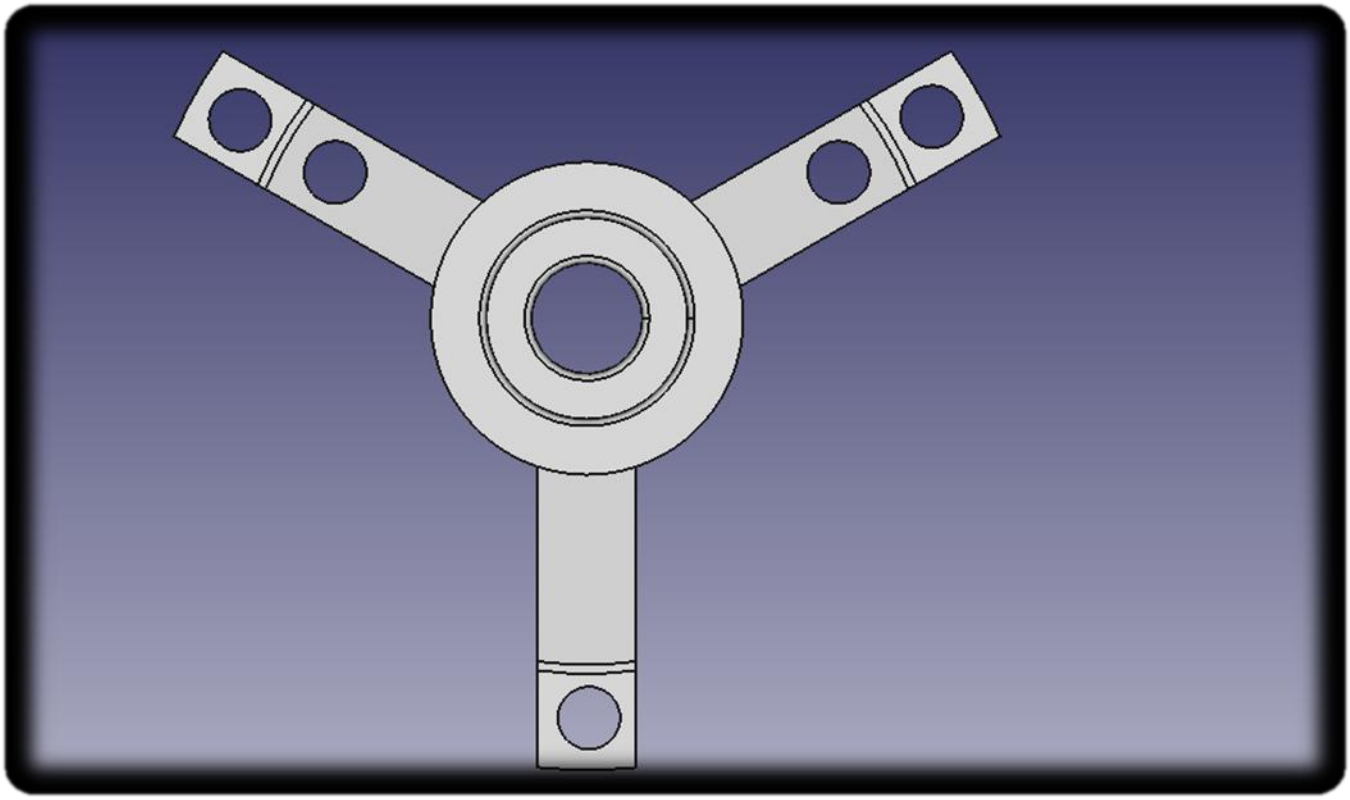
The Chromatography column contains eight main parts: top, handle, cylinder, piston, disc, bottom, base, and column.

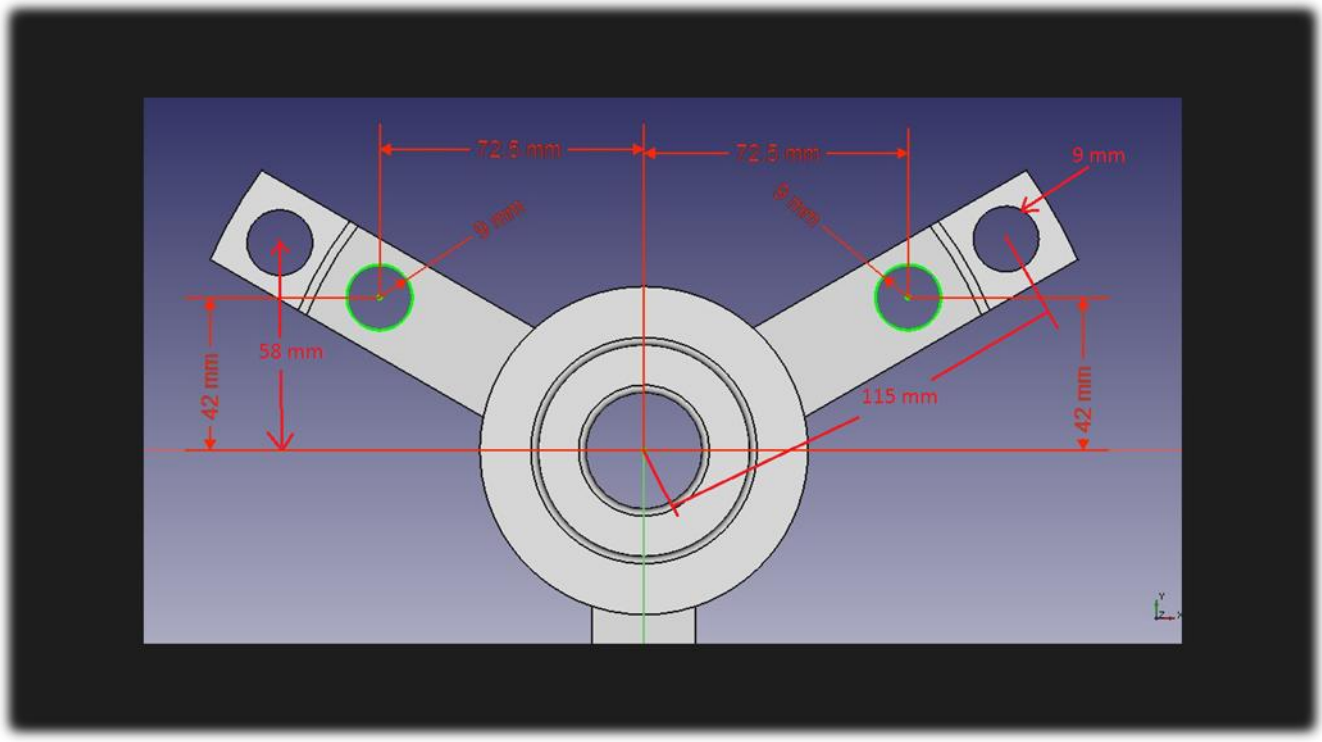
5.4.2.1 Top

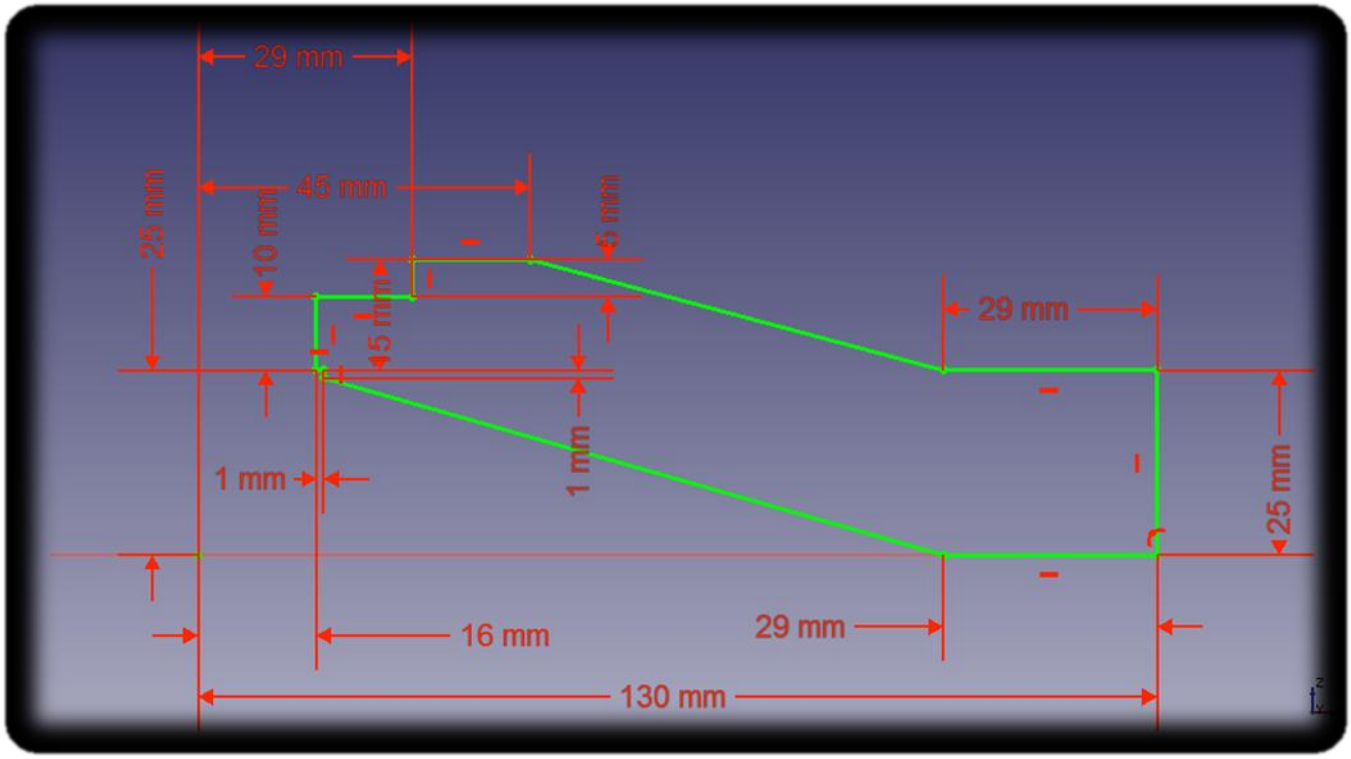
The top is fixed by nuts with three columns that pass through it, and it contains two other holes to let the tube of the piston pass through it







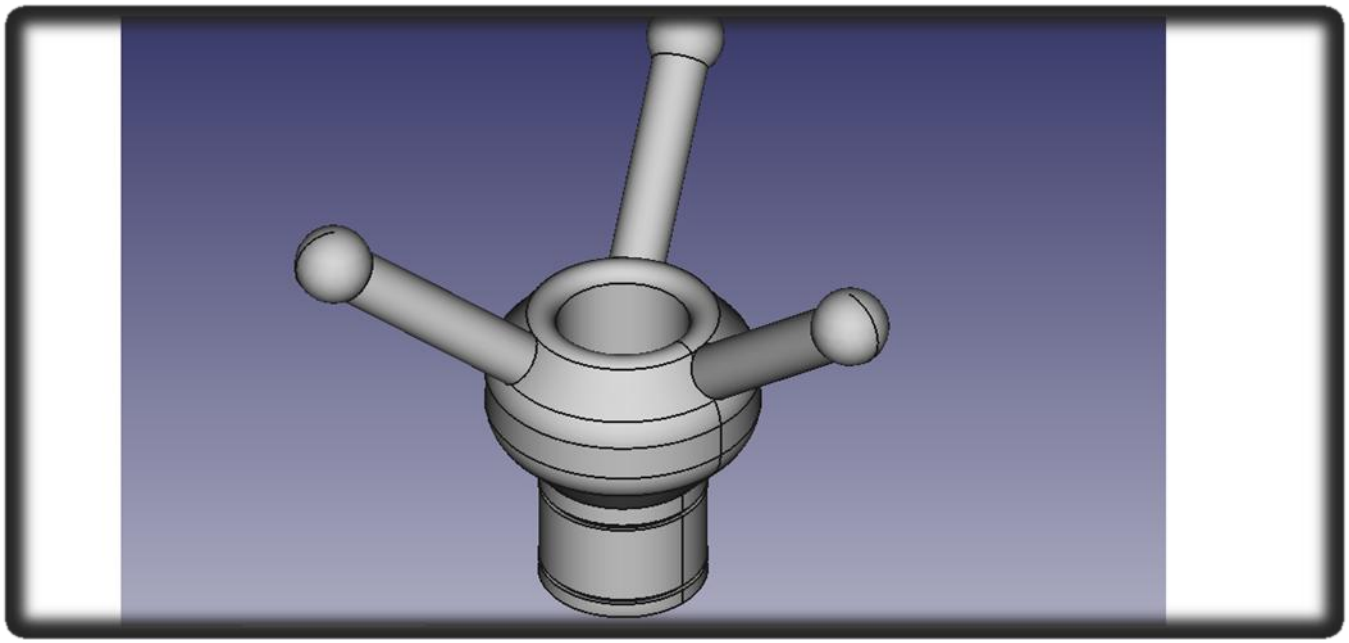

72.5 mm
72.5 mm
9 mm
9 mm
9 mm
58 mm
42 mm
115 mm
42 mm

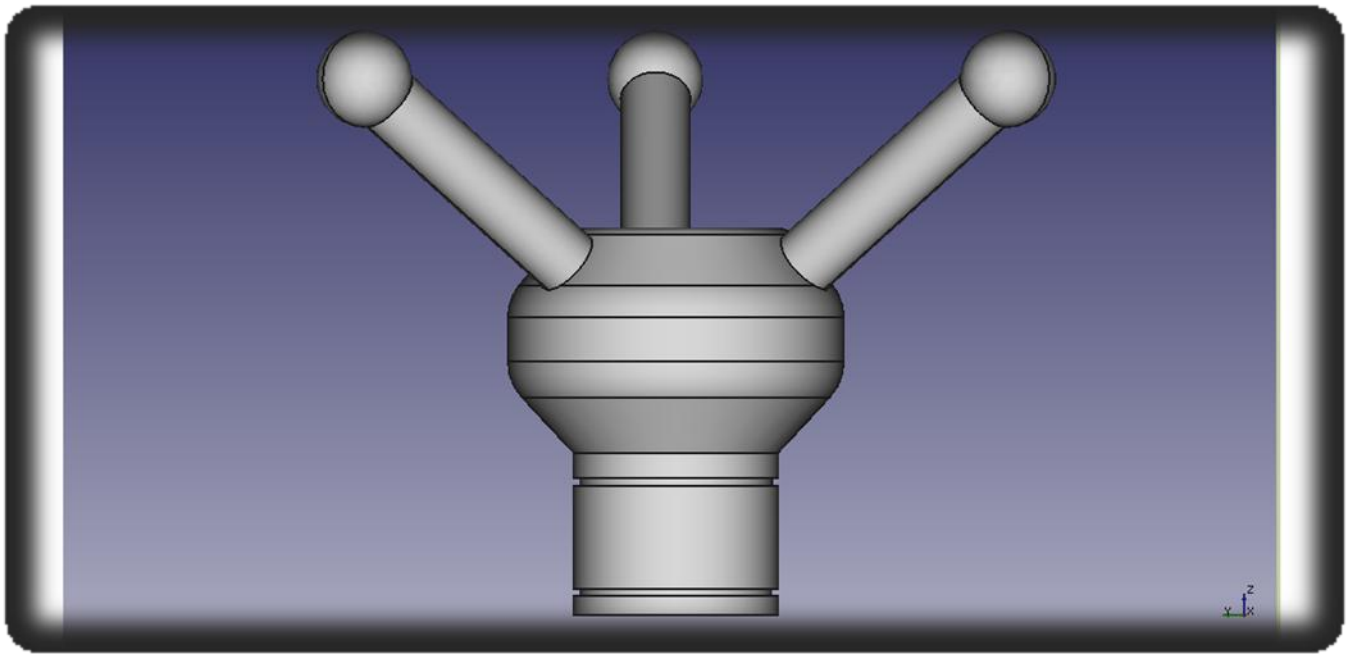


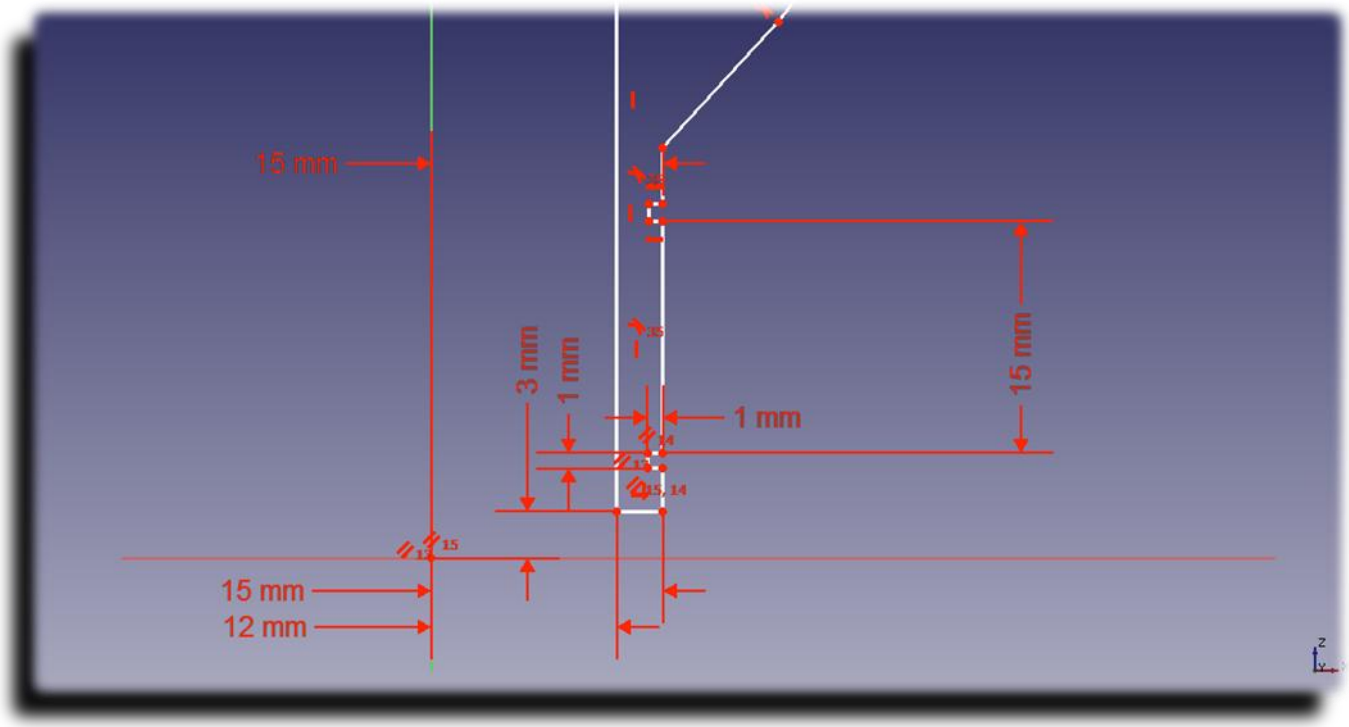
5.4.2.2 Handle

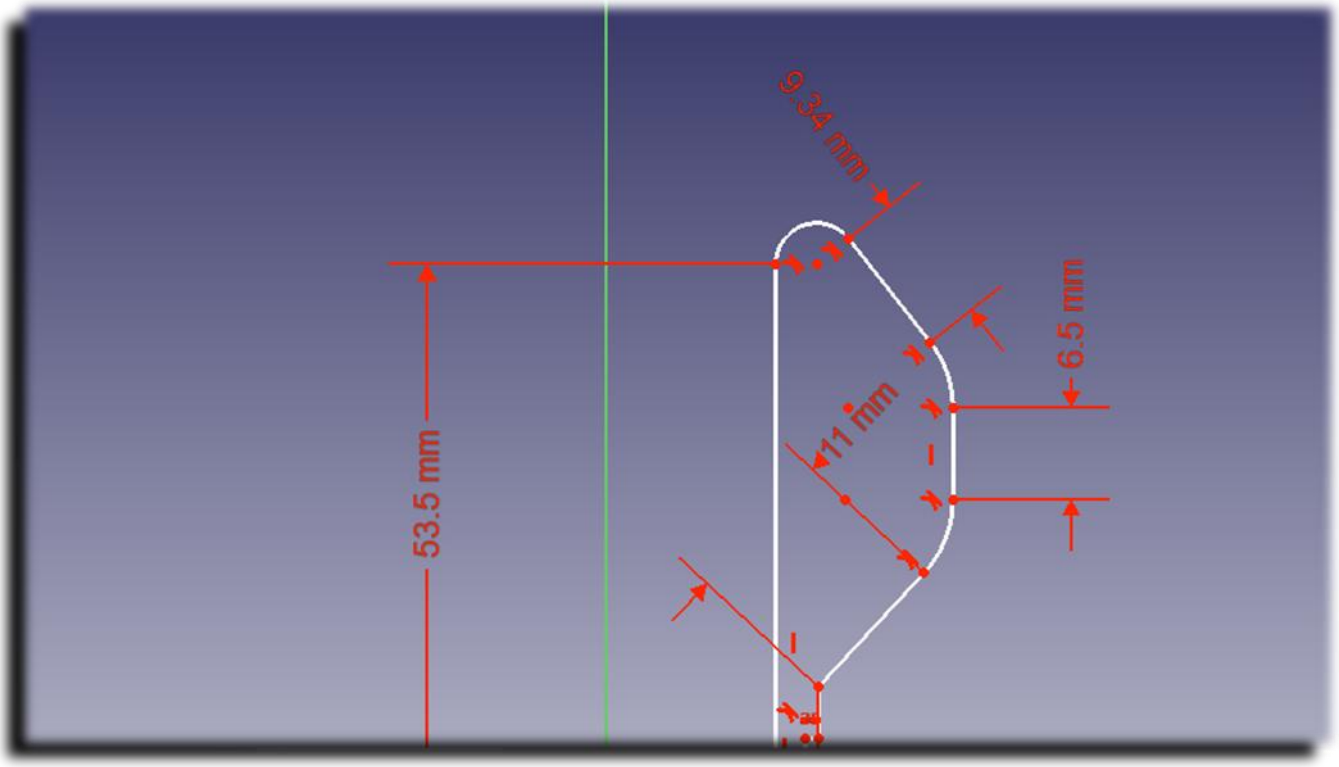
I used a handle instead of a motor, because it doesn't need a high power to rotate it and it is cheaper

Bearing is placed on the bottom between two rings



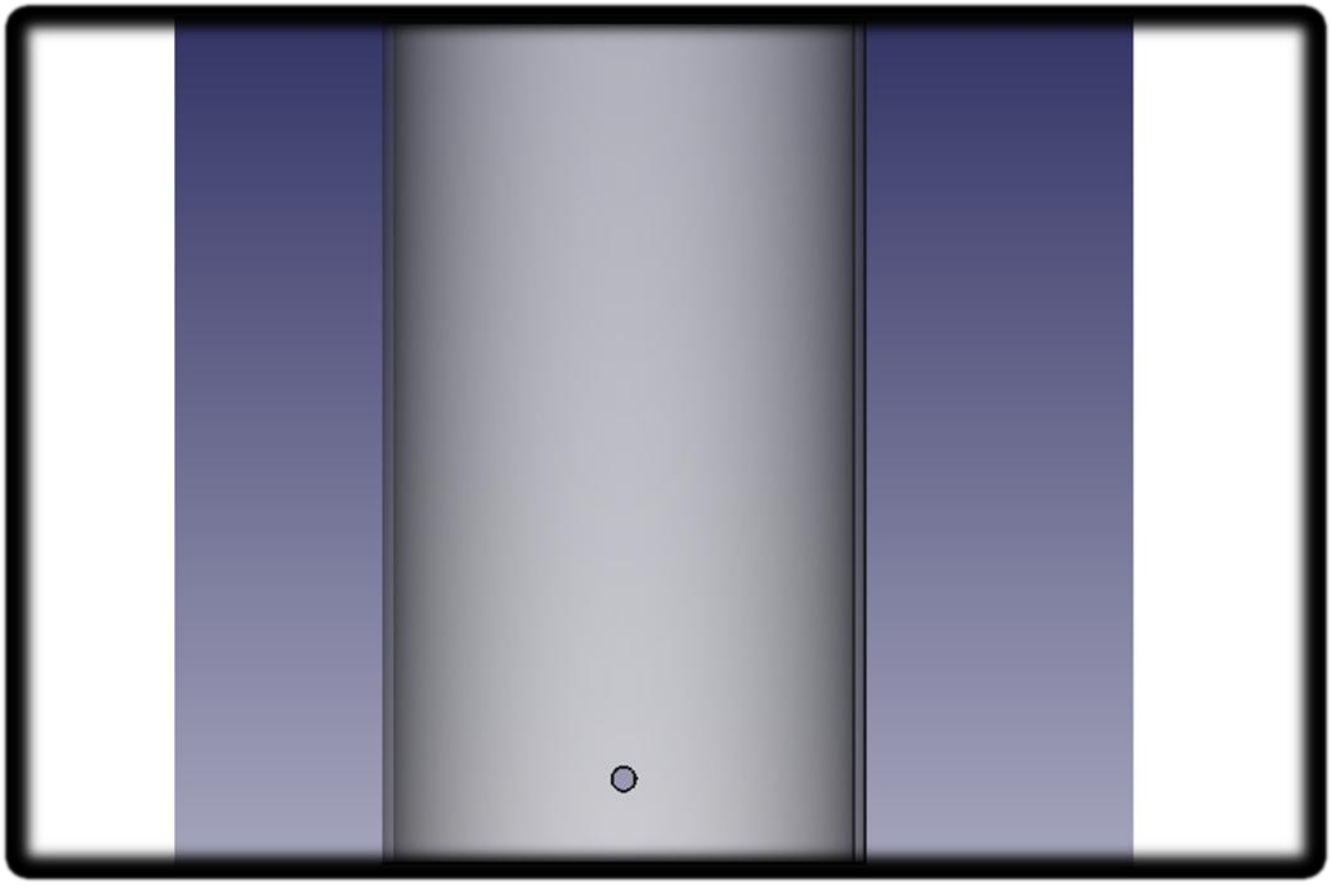


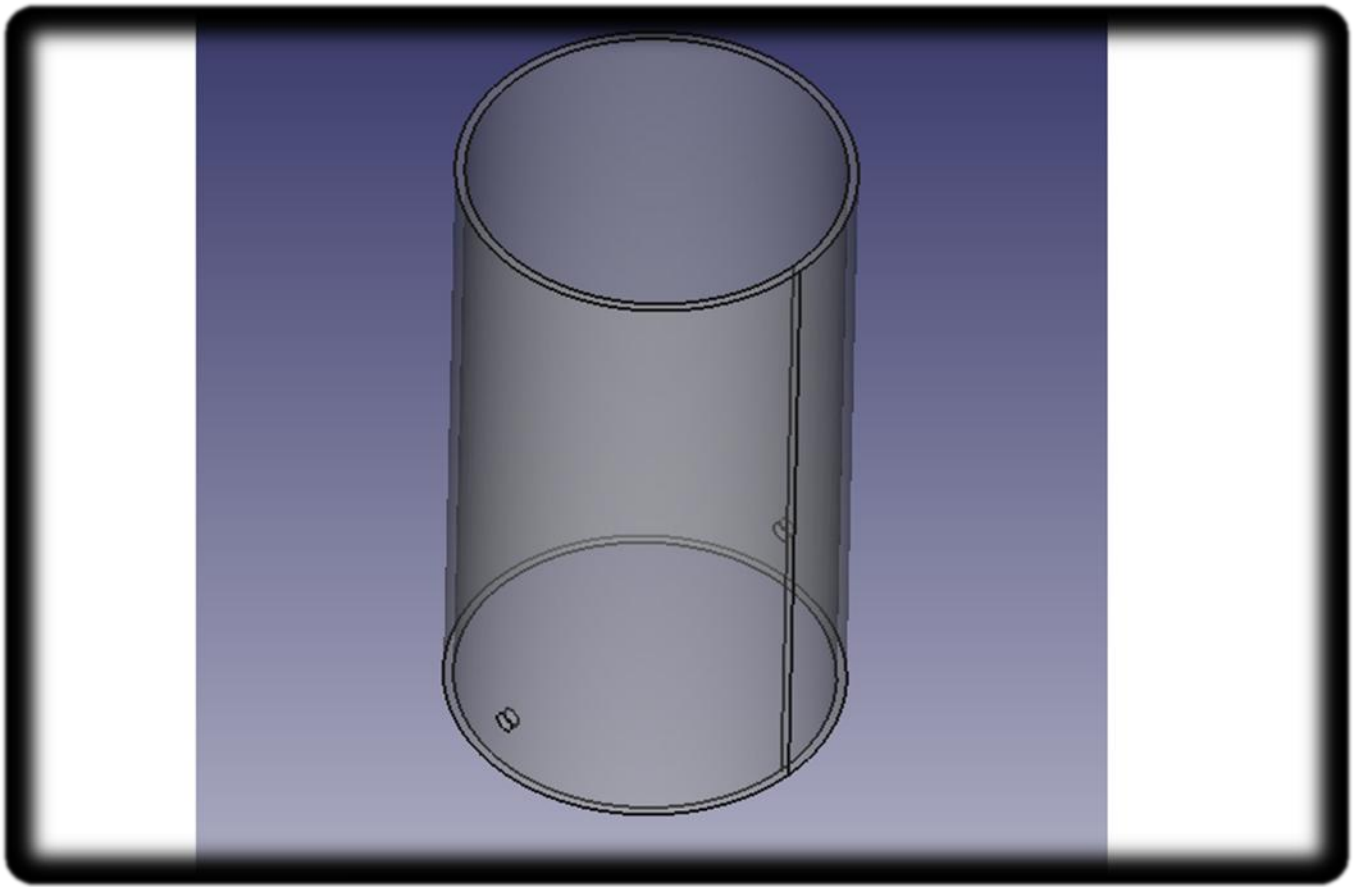

15 mm
3 mm
1 mm
1 mm
15 mm
15 mm
12 mm

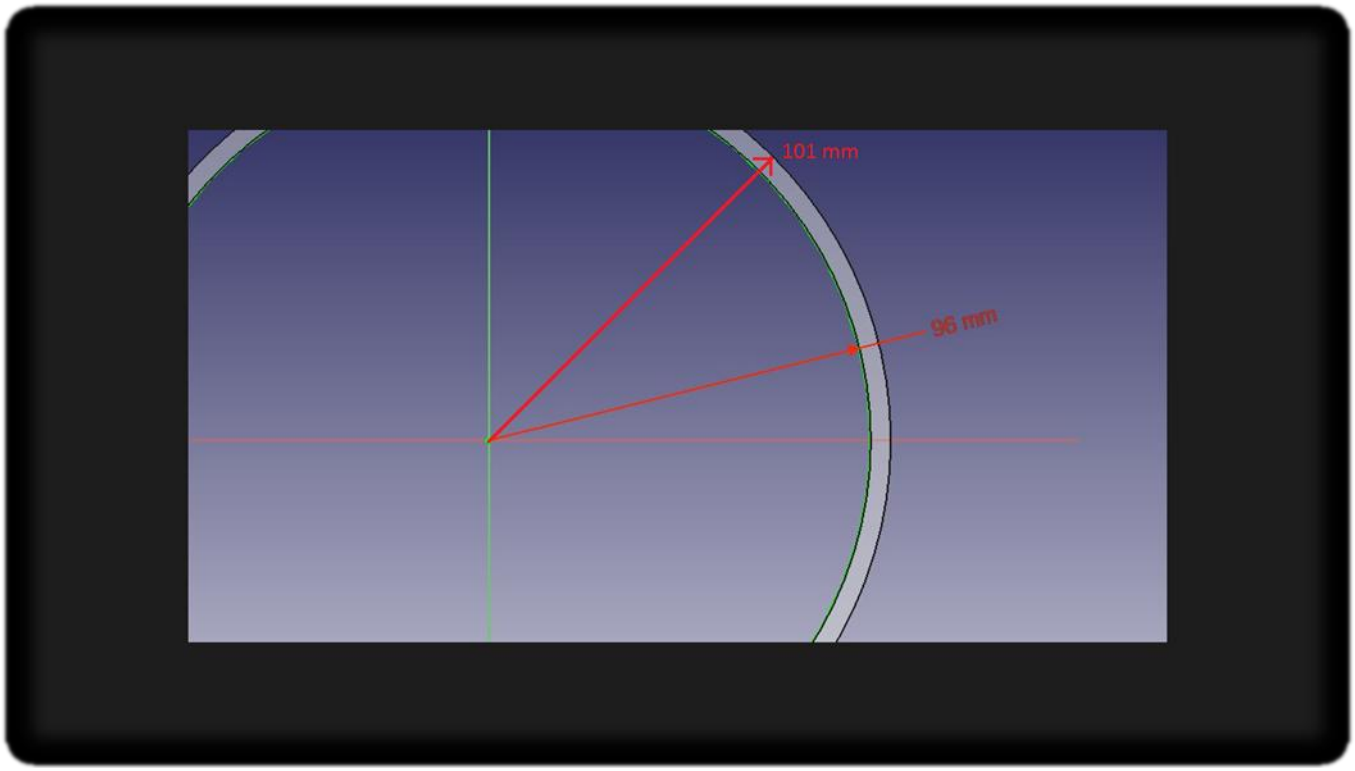


5.4.2.3 *Cylinder*

Is made of Plexiglas, it's where we put the solid particles

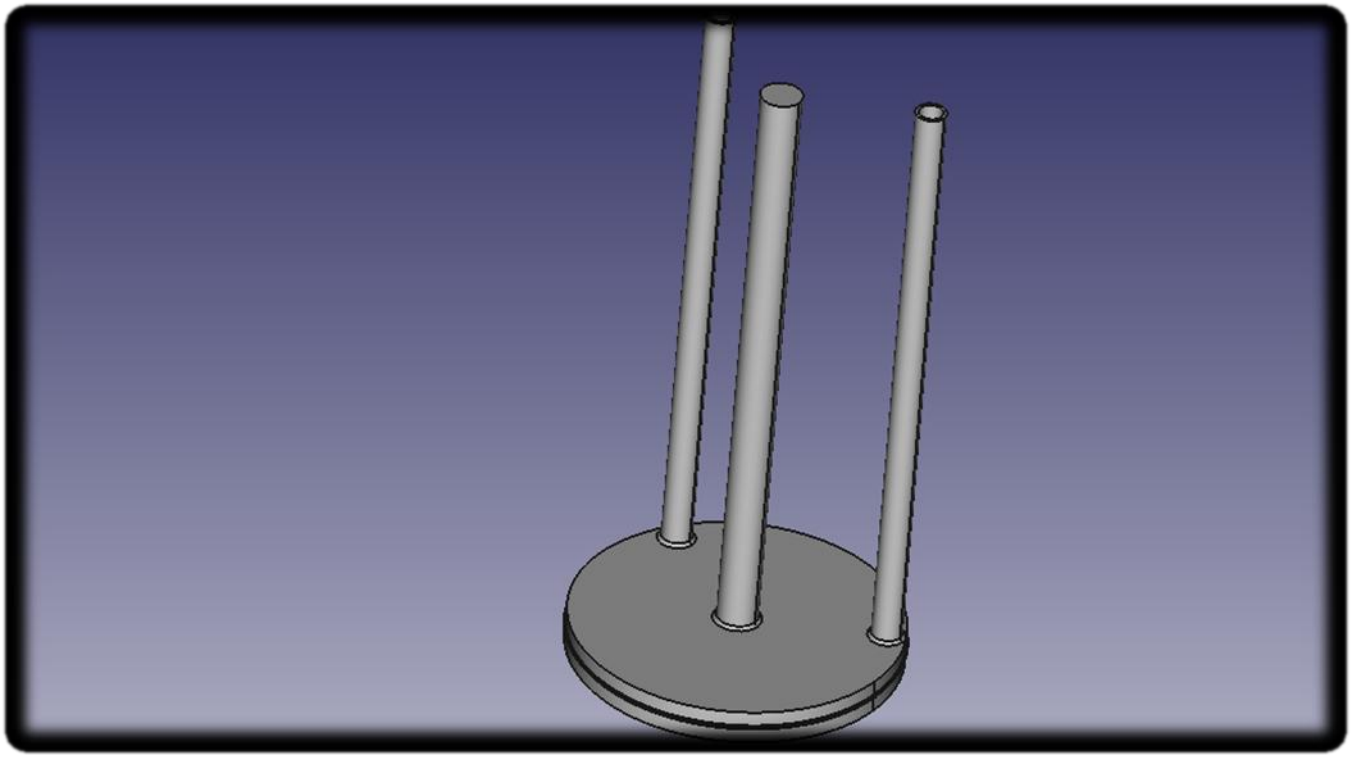


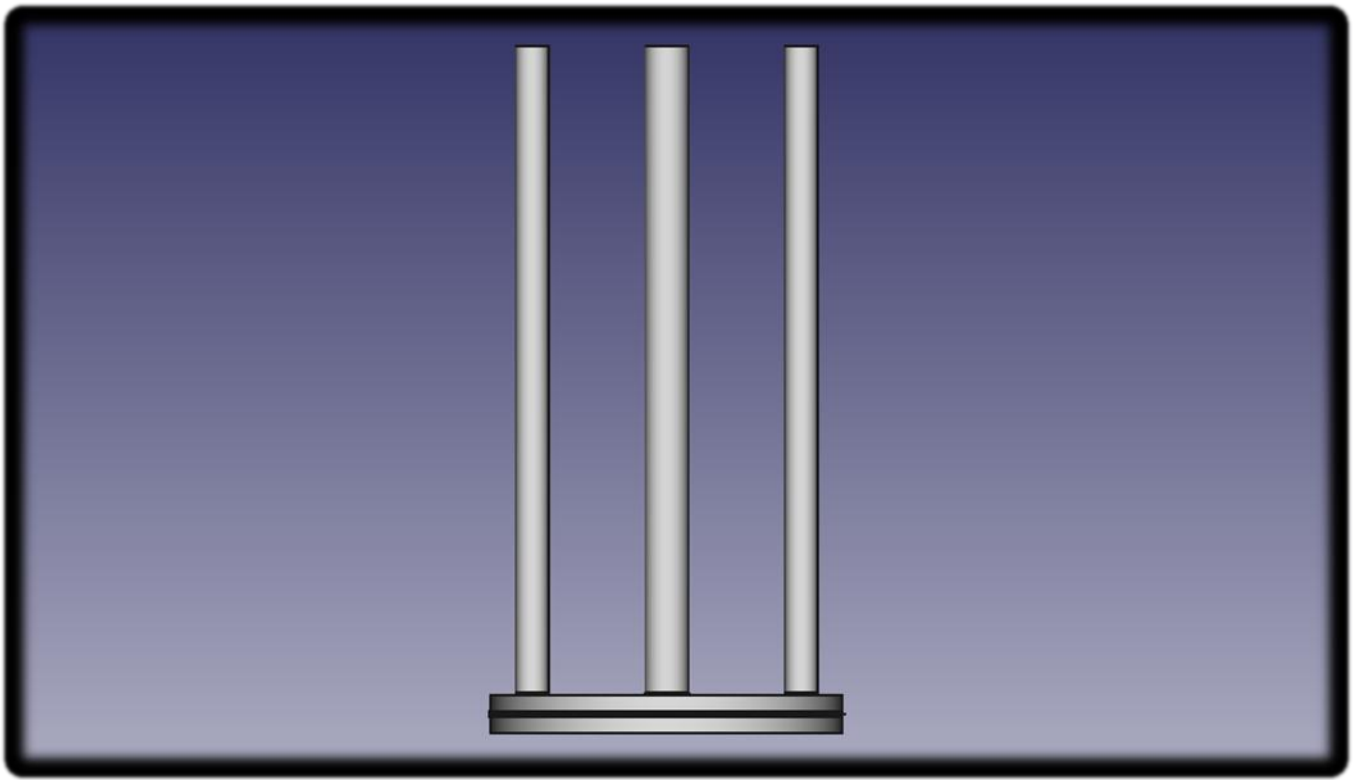


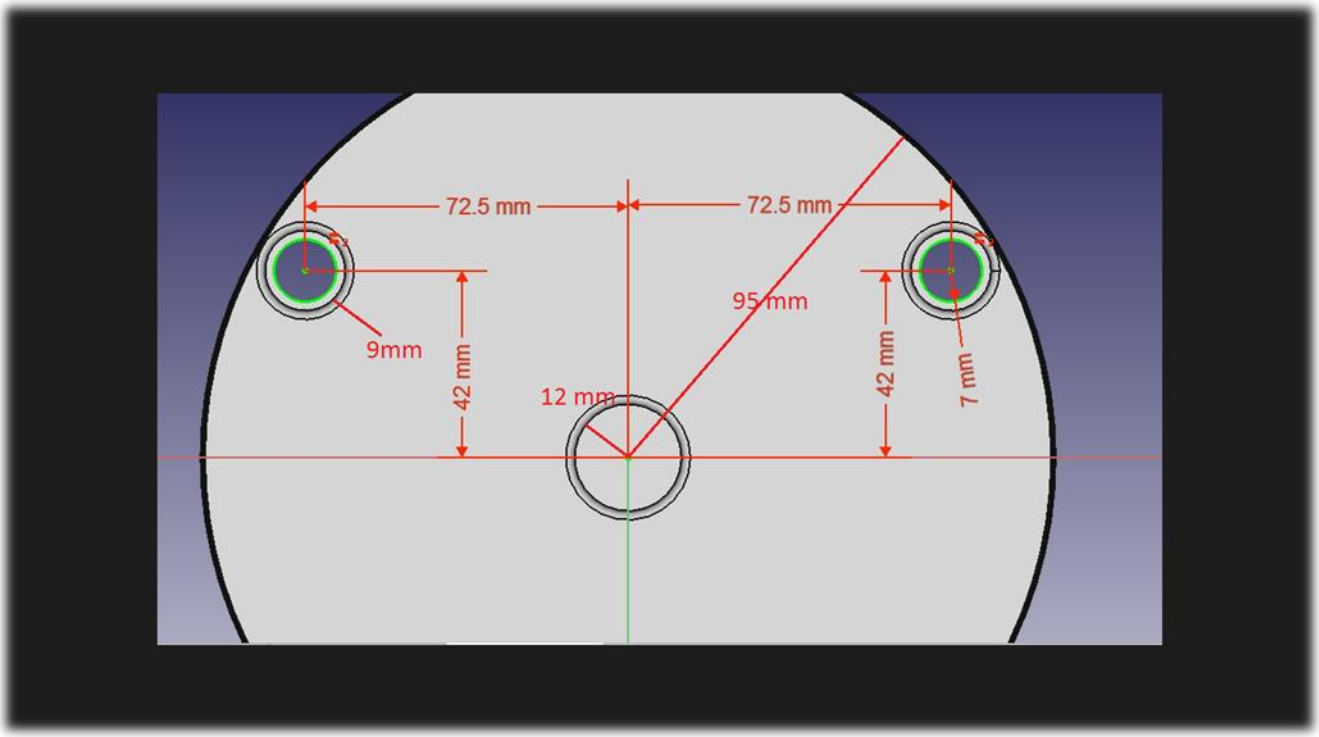


5.4.2.4 Piston

Contains a ring to avoid fluid leaks and to let the piston move easily and two tubes where fluid passes through into the piston

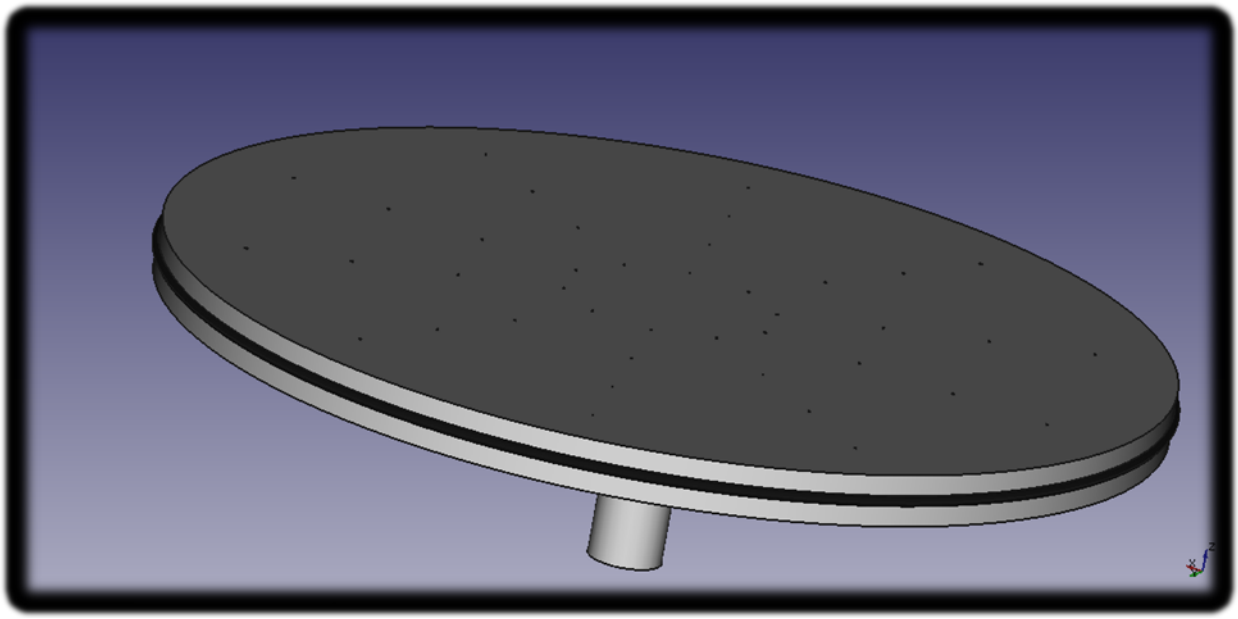


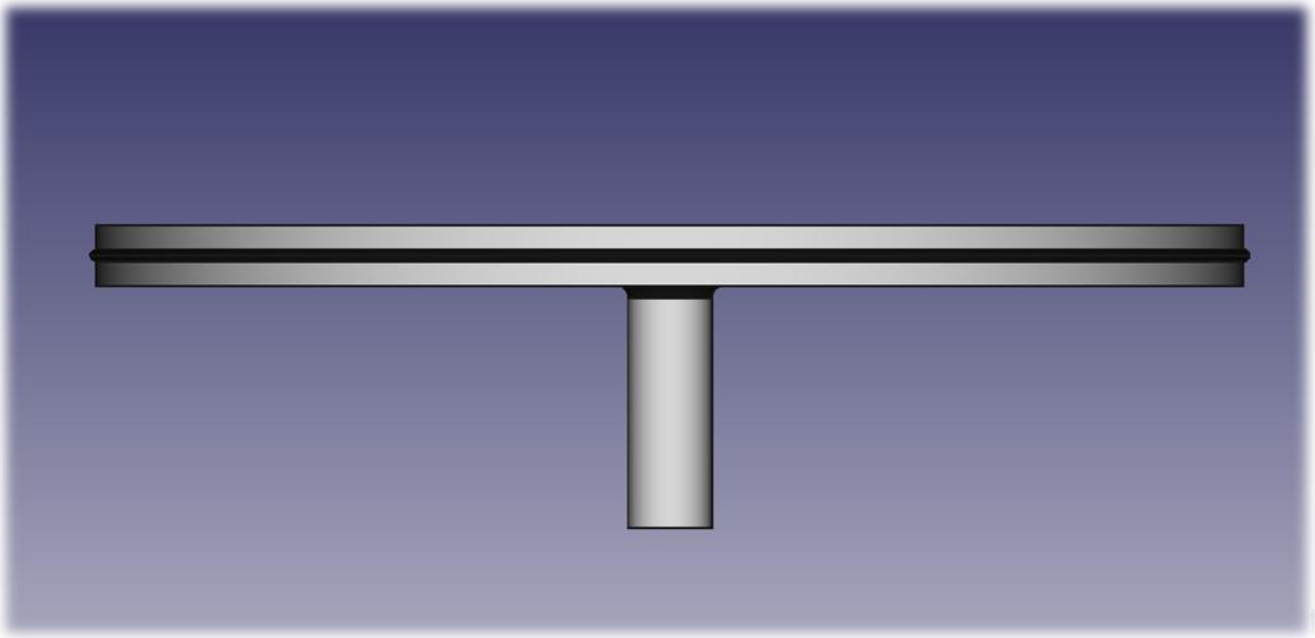


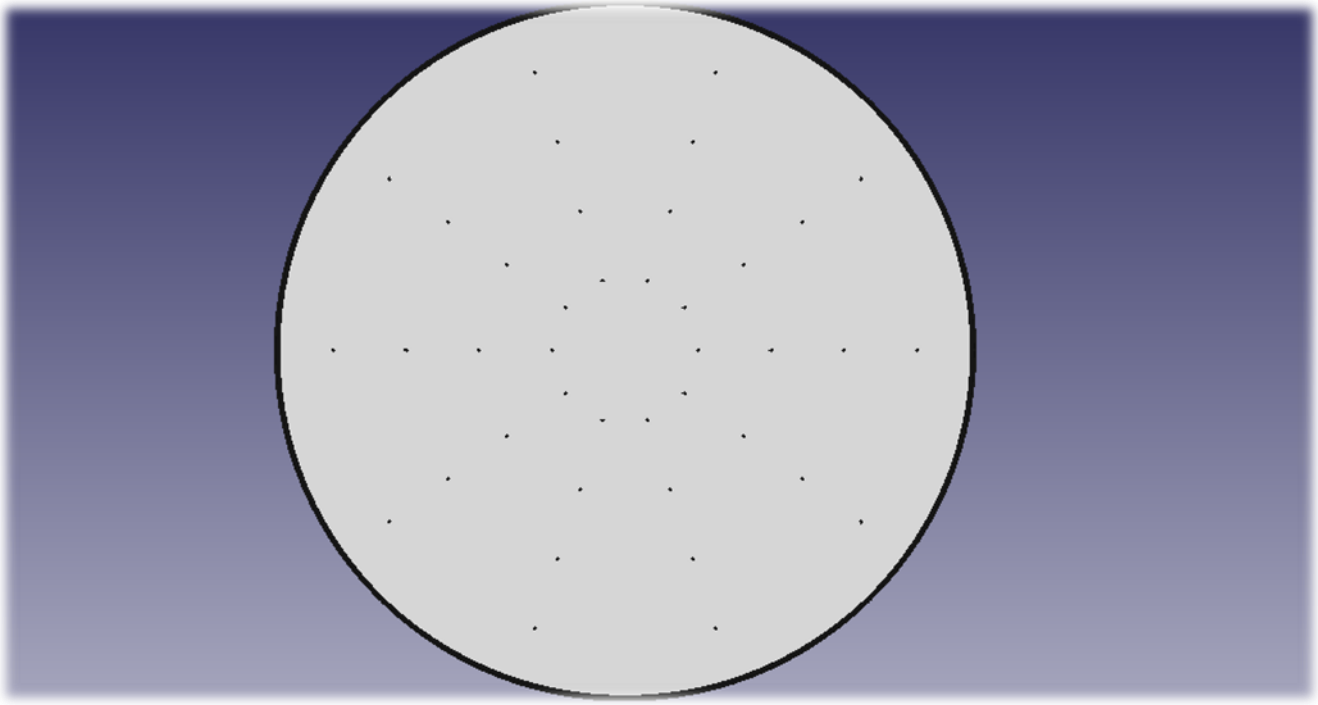


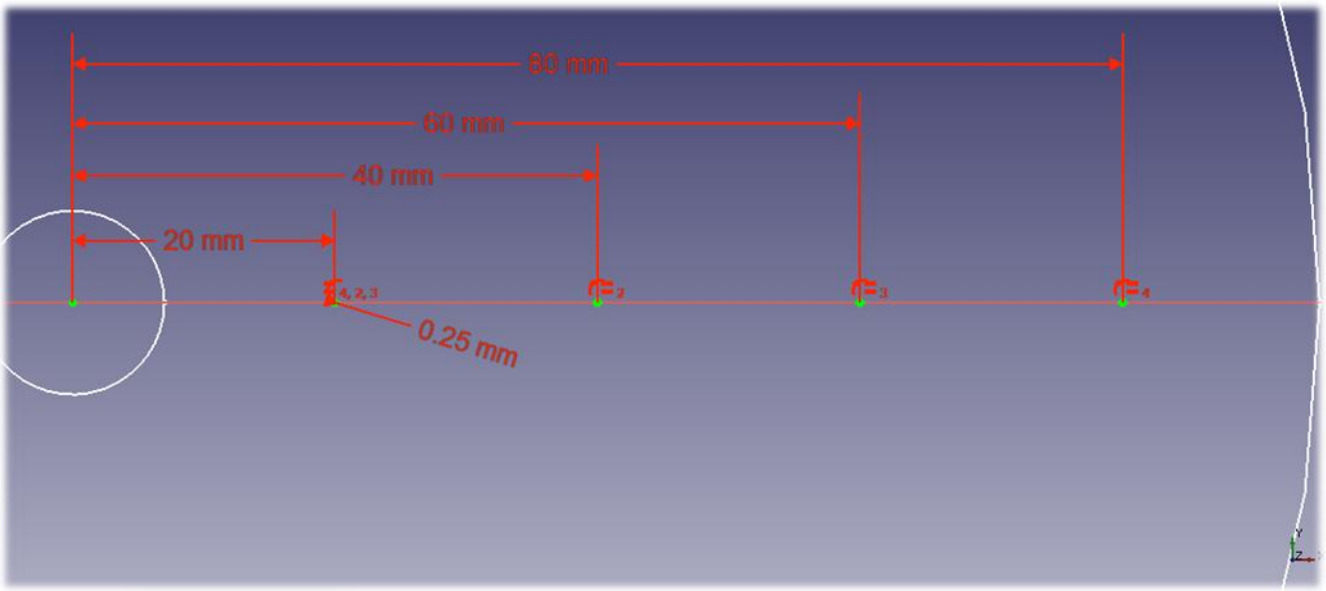
5.4.2.5 *Disc*

Is placed on the bottom of the cylinder. It contains holes of diameter 0.25 mm to separate big molecules from small ones and it contains a ring to avoid leakage



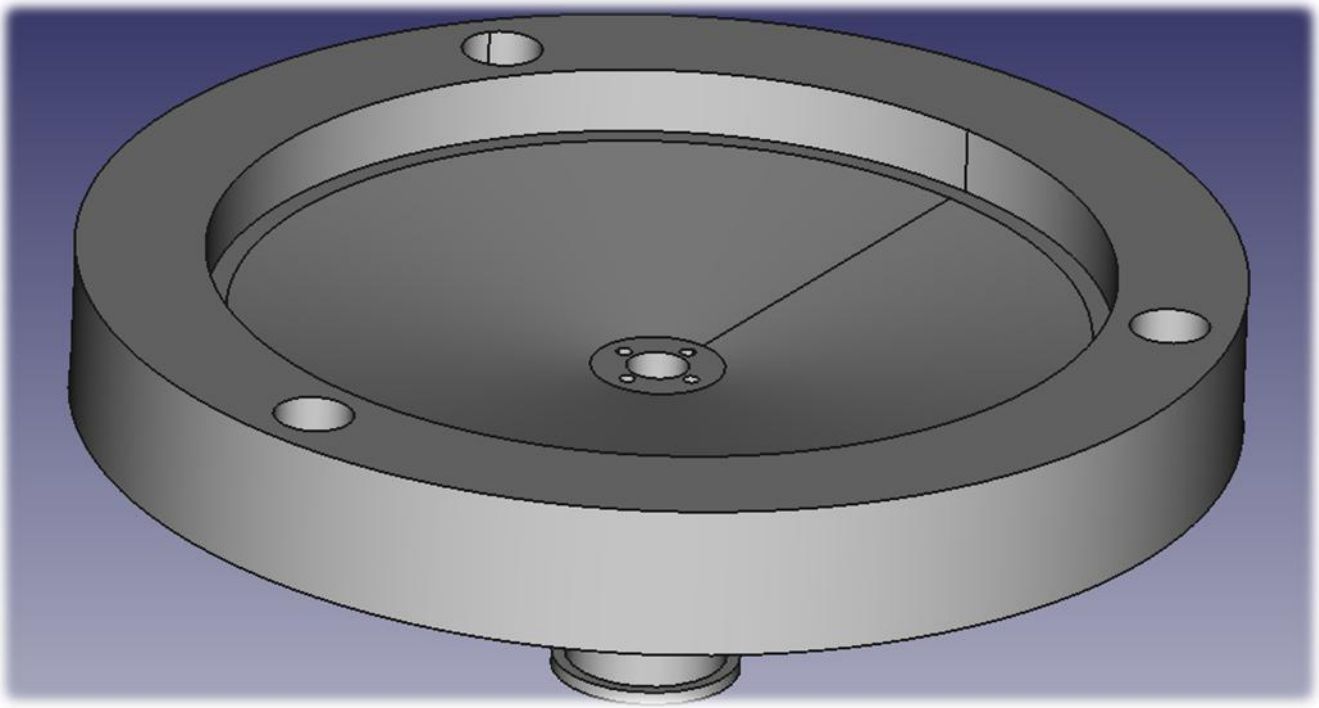


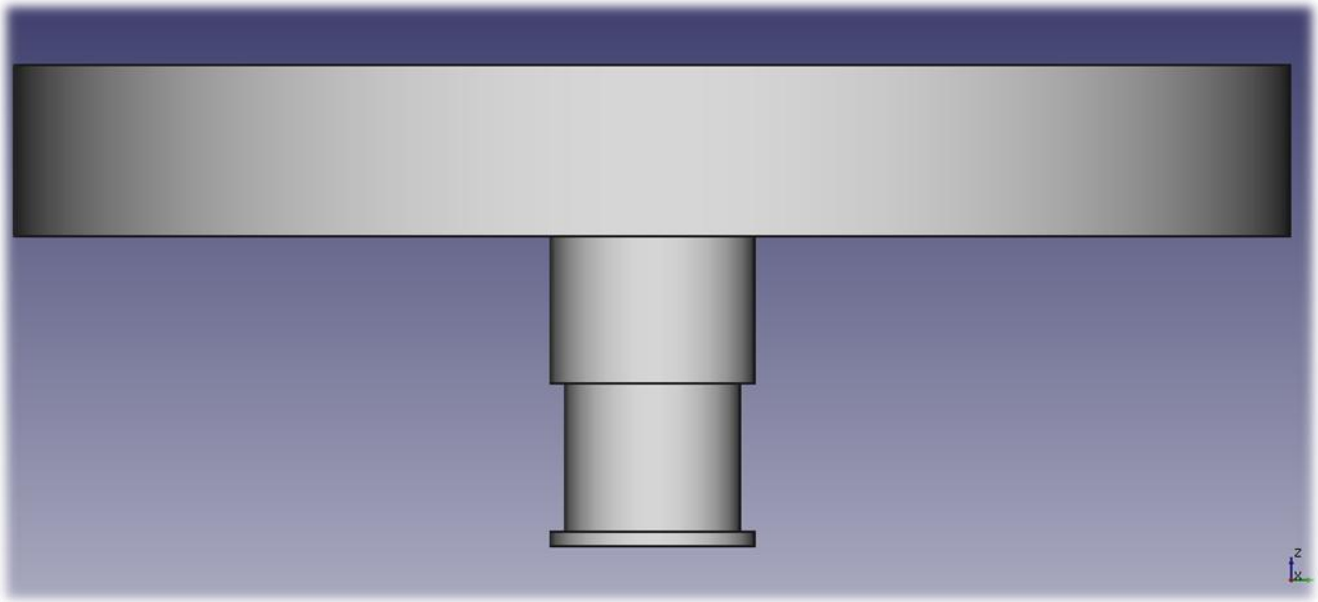


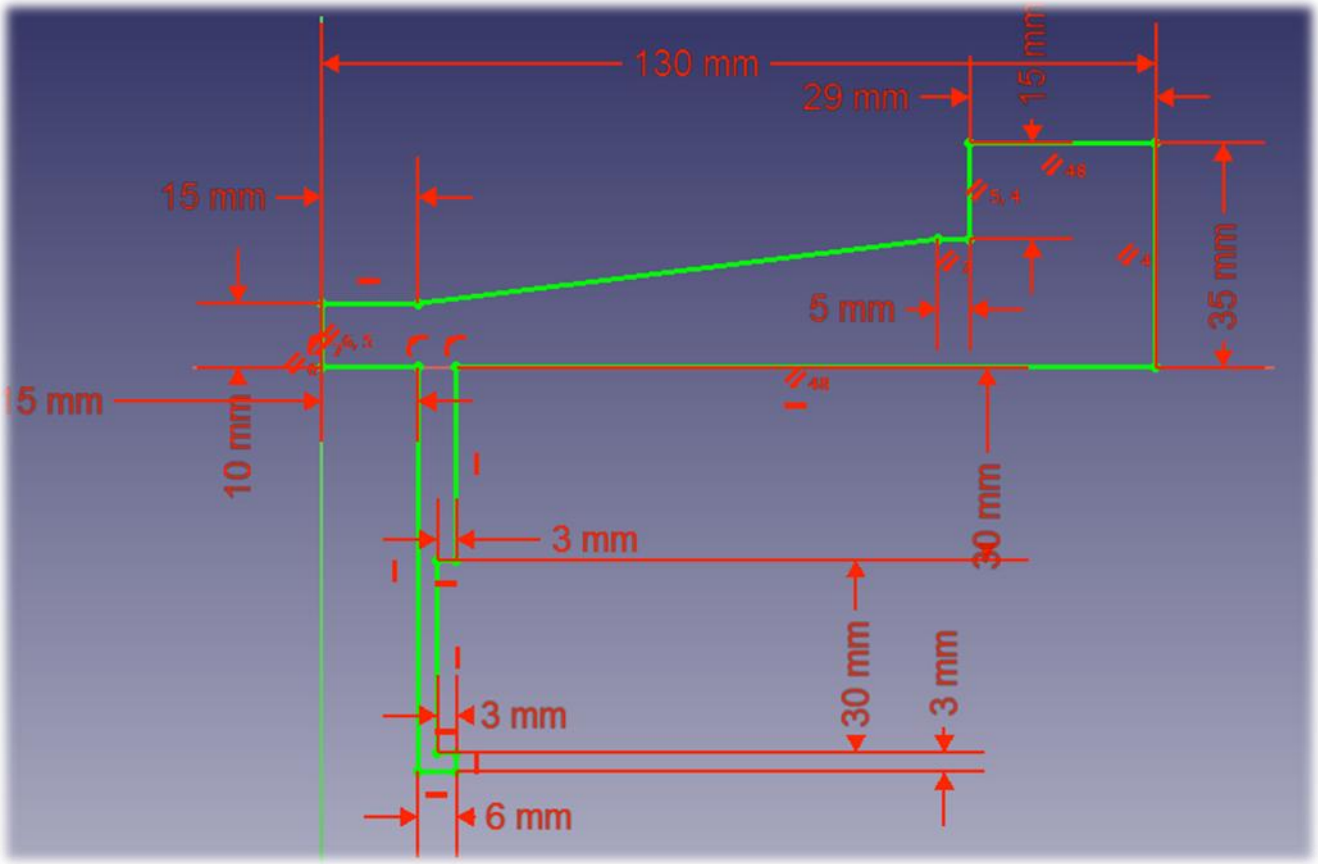


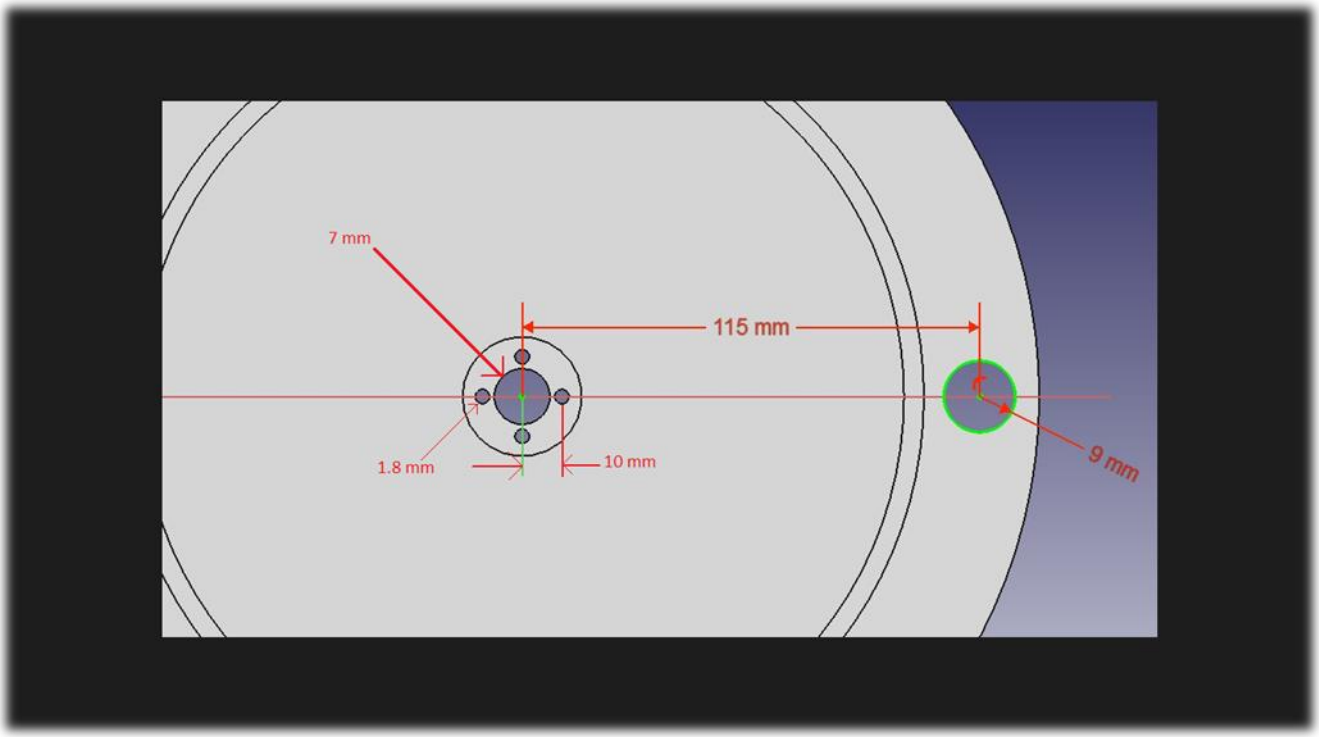
5.4.2.6 *Bottom*

Contains three external holes fixed to the three columns by bolts.it has a conical shape to let the fluid reach to the center



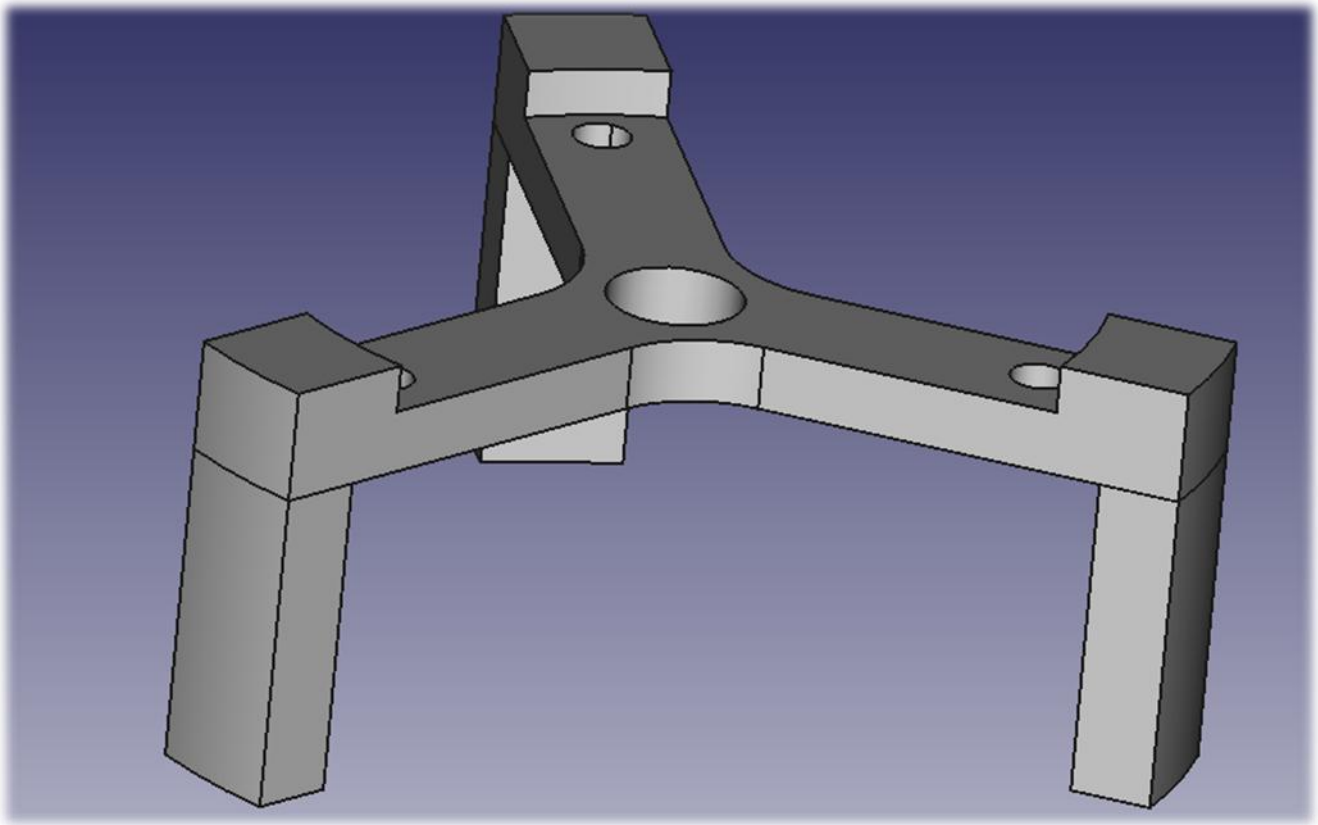


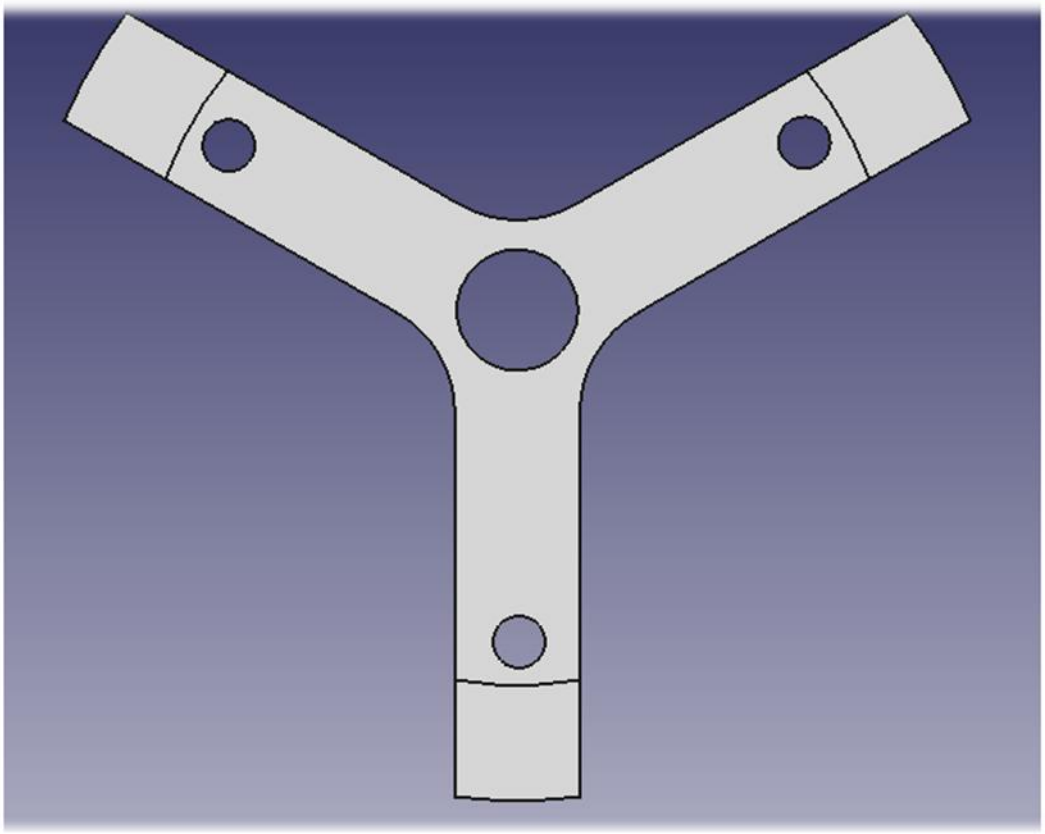

130 mm
29 mm
15 mm
15 mm
35 mm
5 mm
15 mm
10 mm
30 mm
3 mm
30 mm
3 mm
3 mm
6 mm

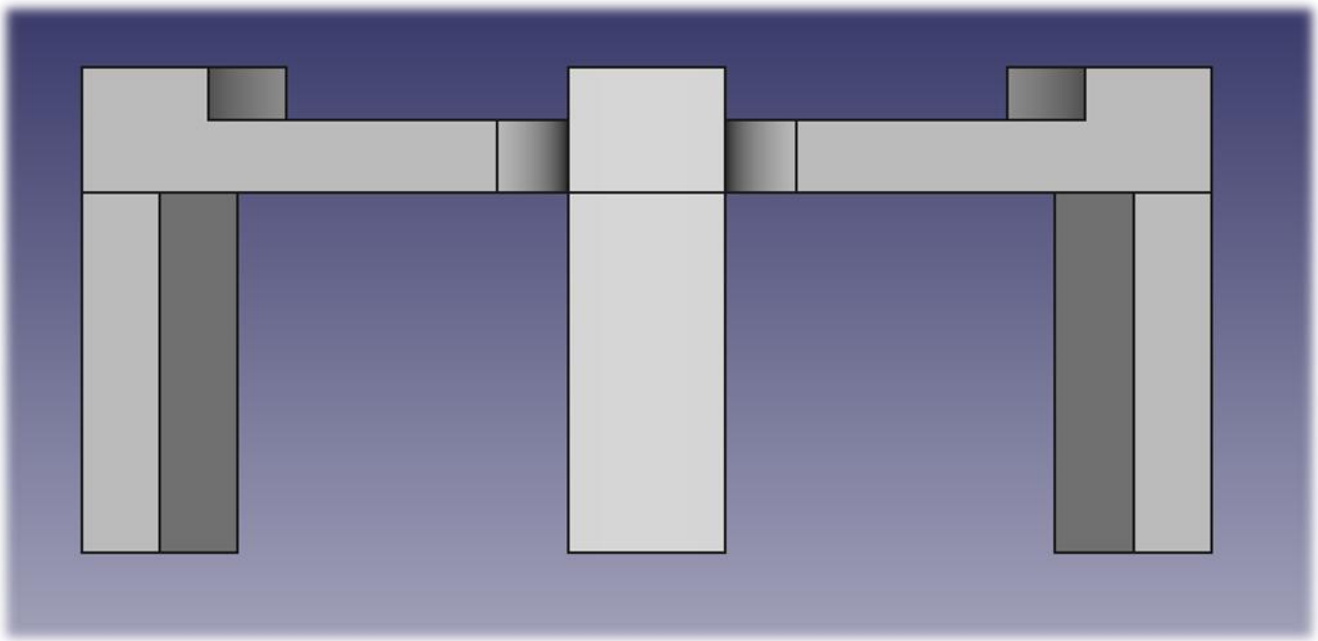


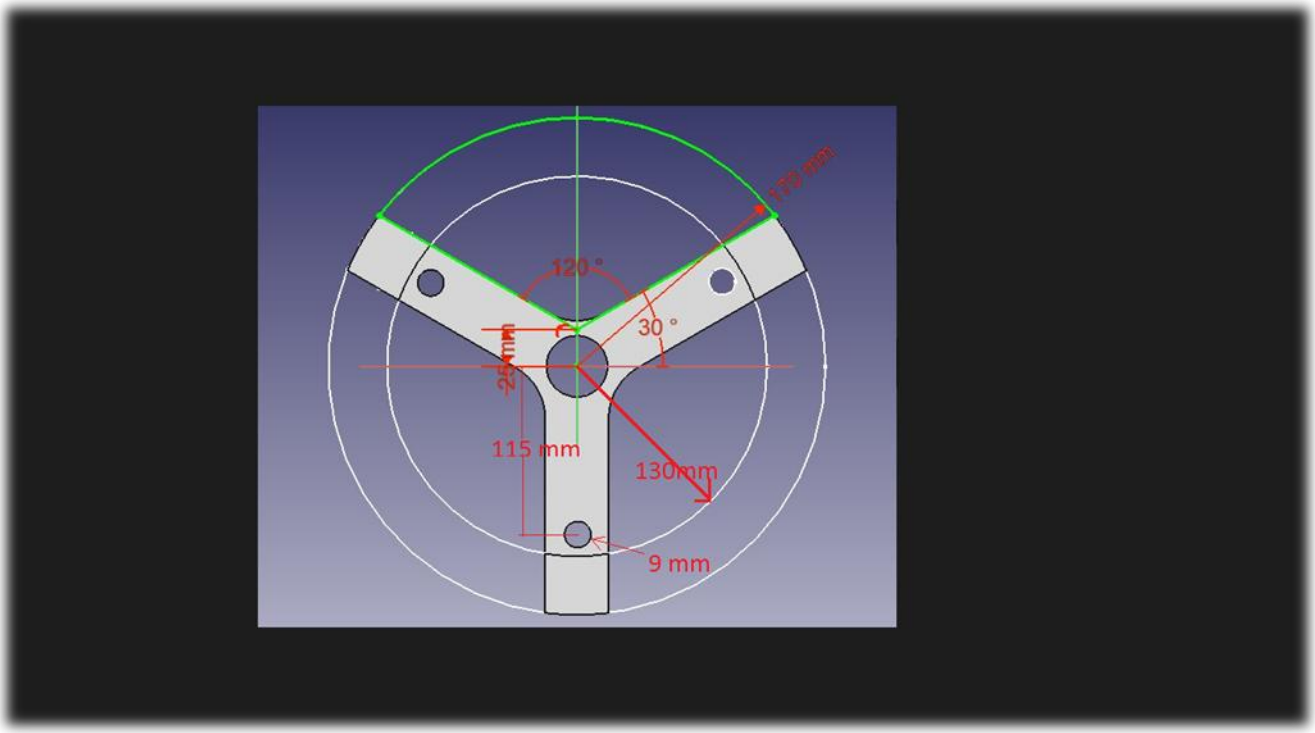
5.4.2.7 Base

Contains three legs and three external holes to fix the columns by bolts



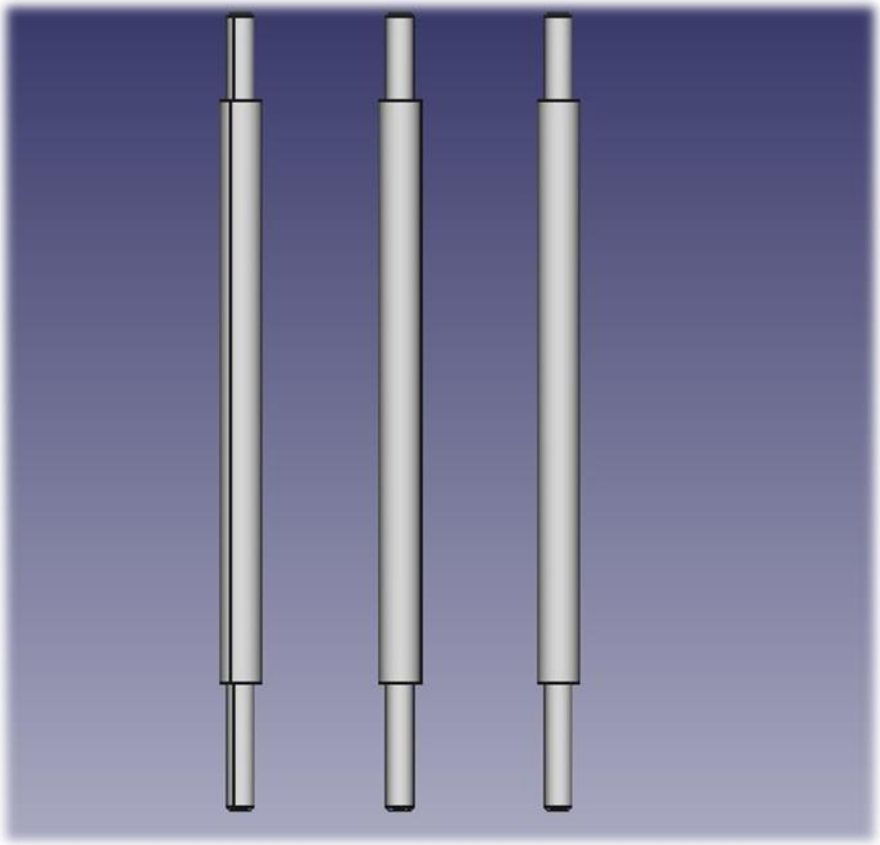


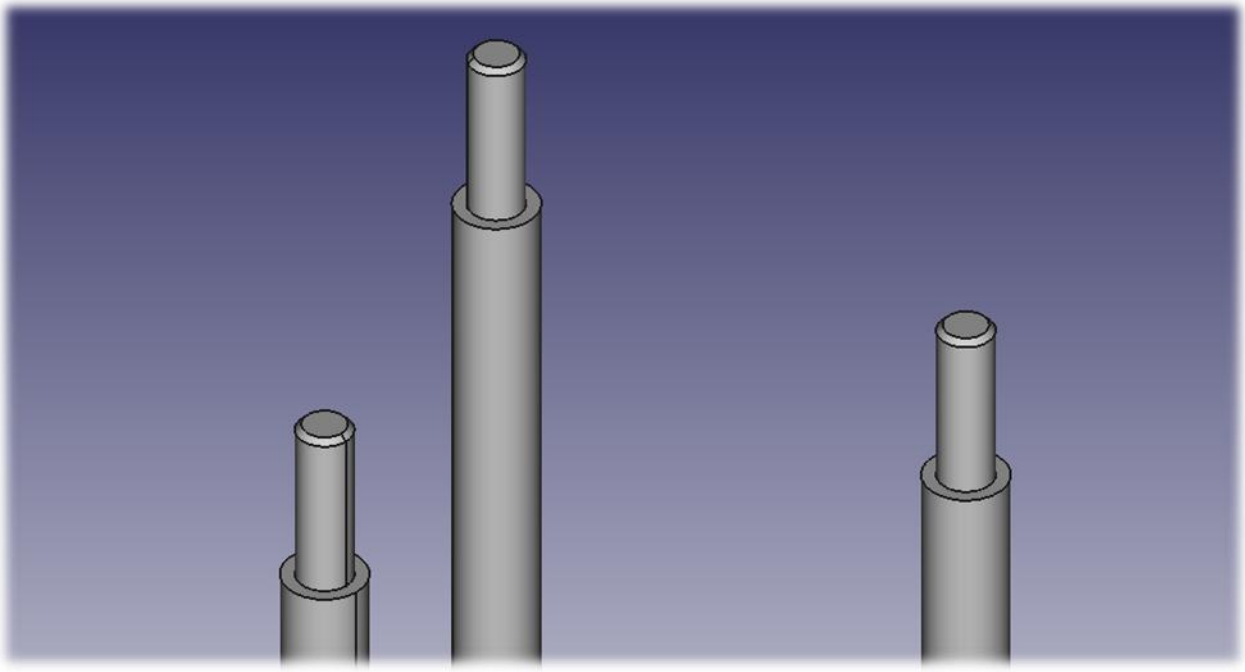


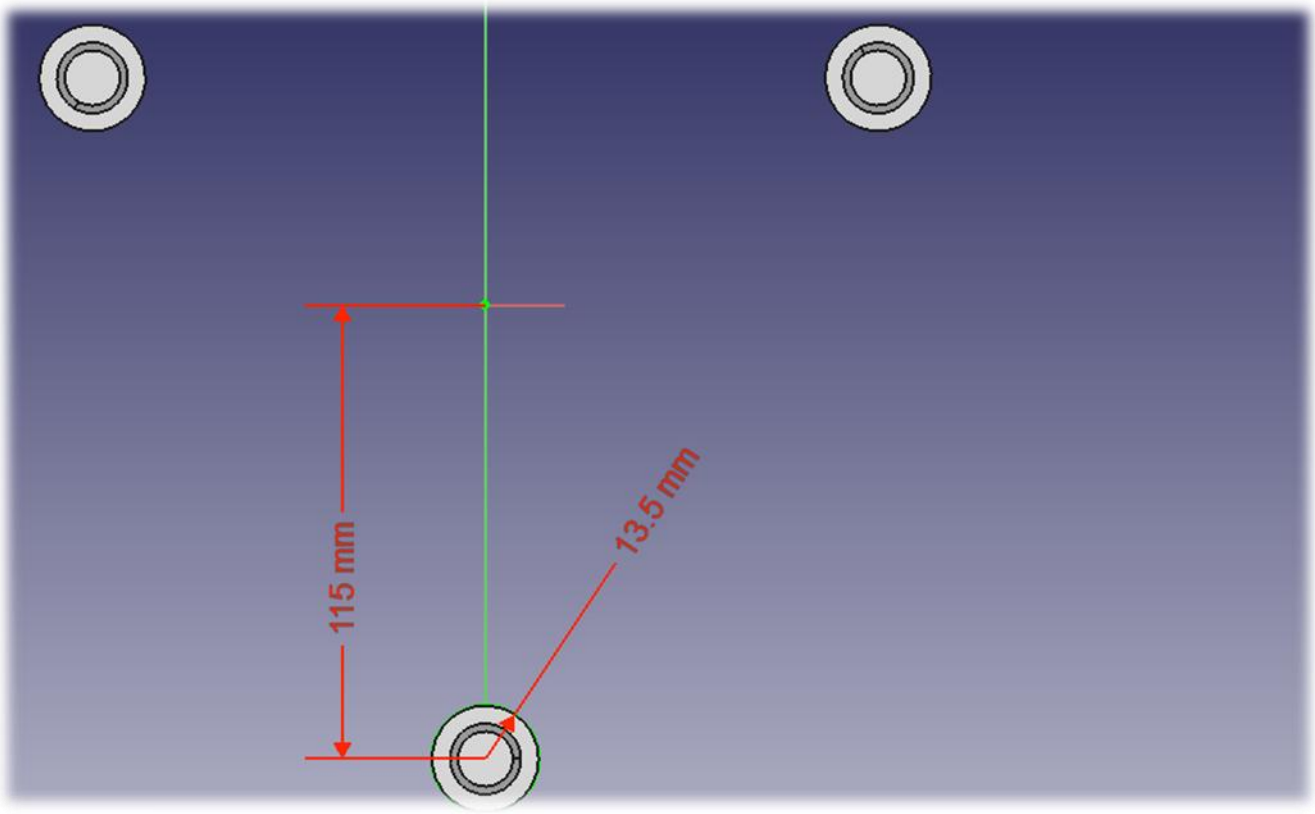


5.4.2.8 *Column*

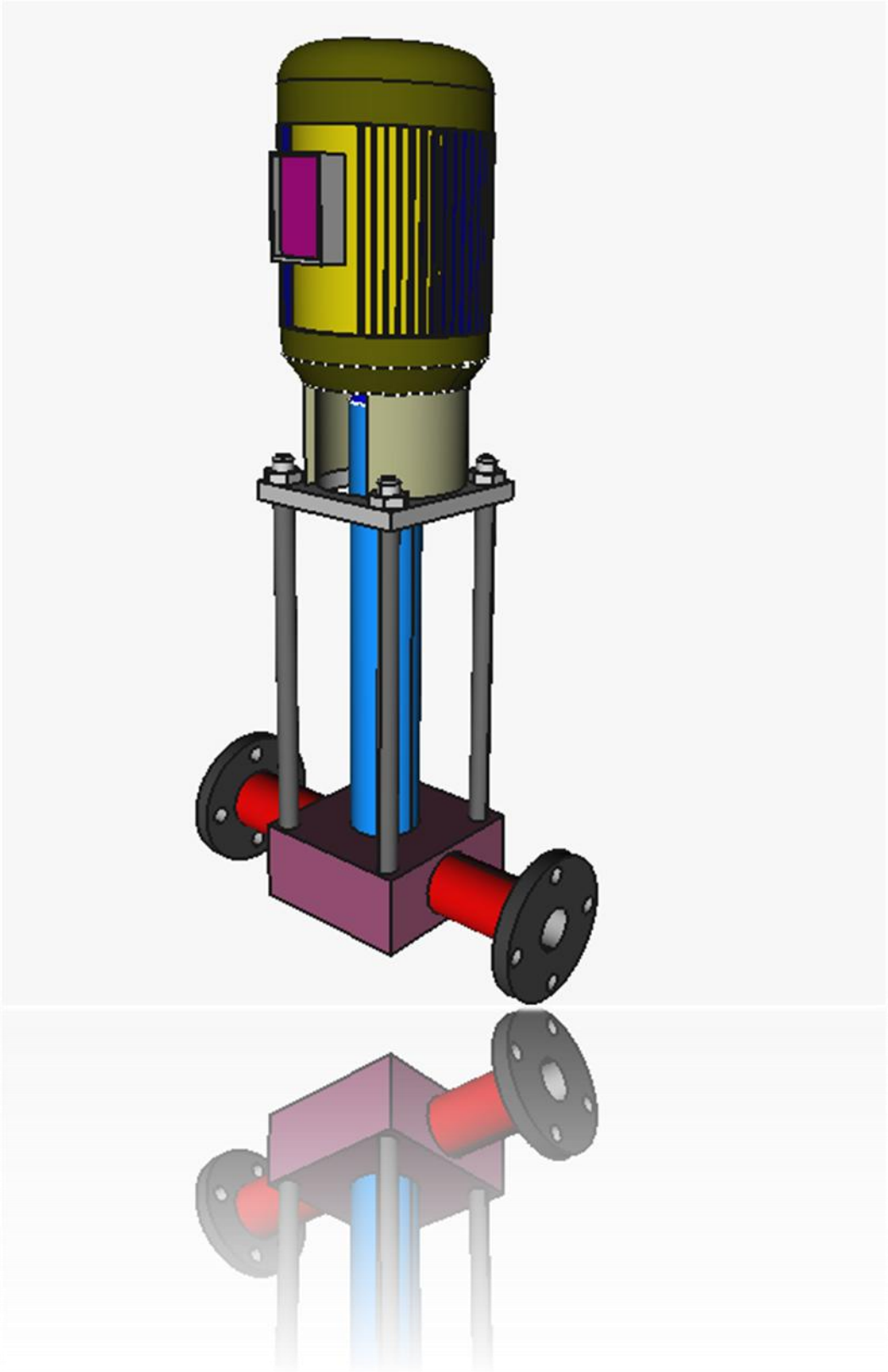
It is the part that connects the top, bottom and base parts together.

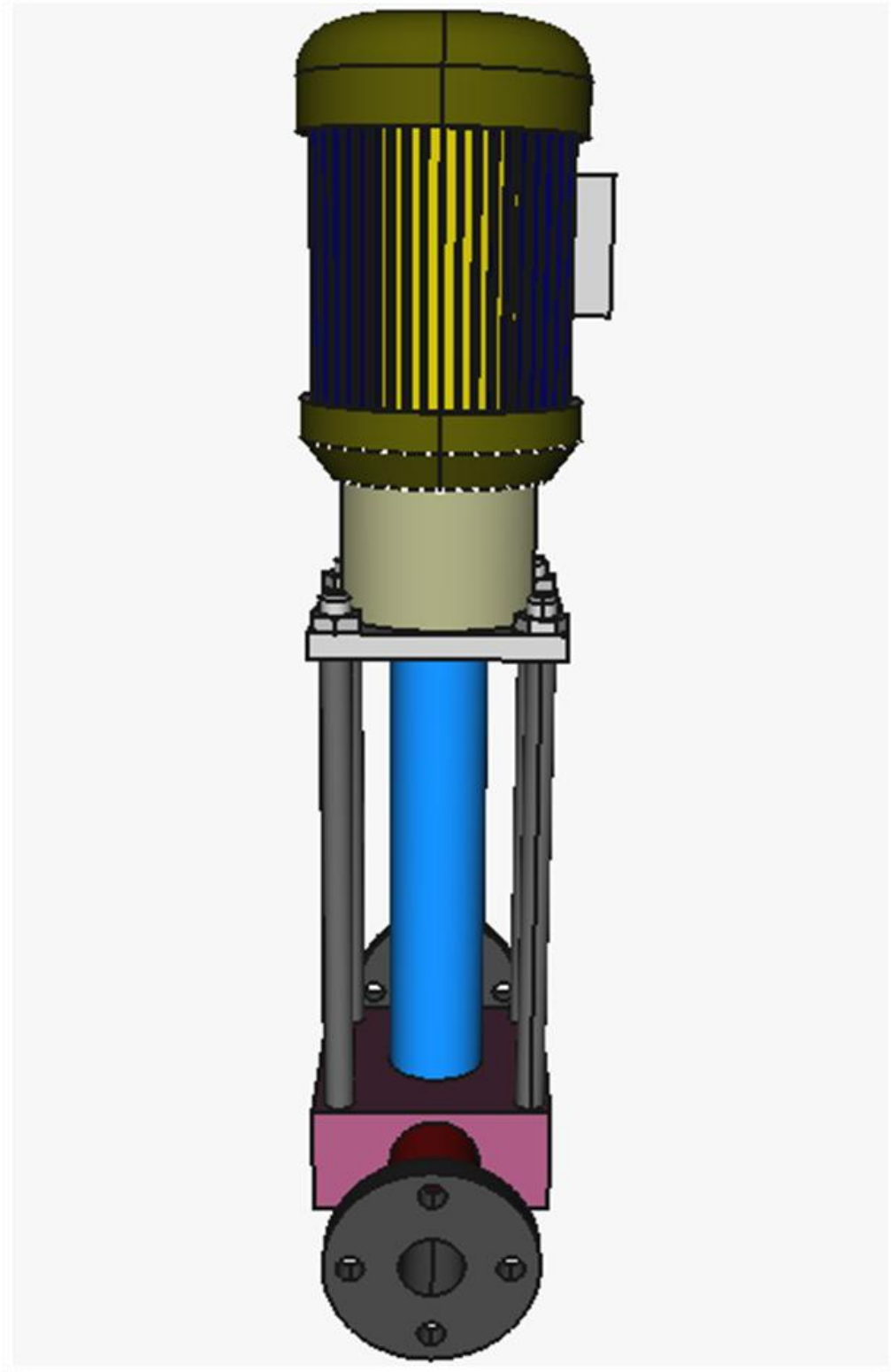


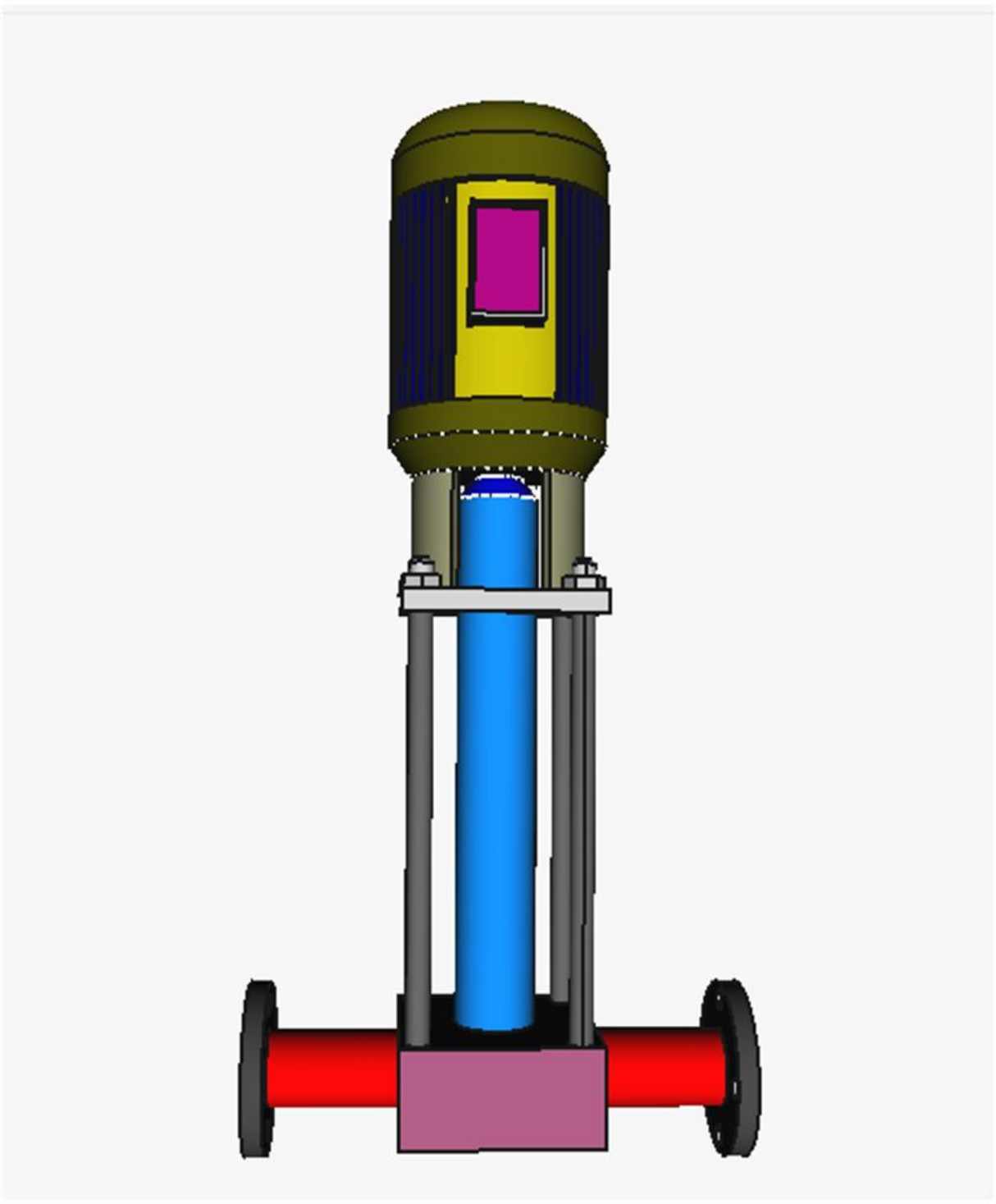



115 mm
13.5 mm

5.5 Demonstration and Modelling of the Pump







5.6 Demonstration and Modelling of the Homogenizer

-Ibrahim Zaaroura-

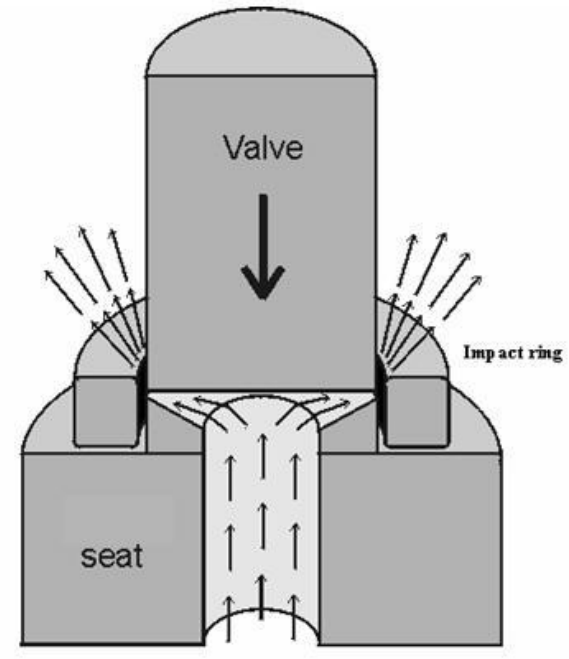
5.6.1 *DEVICE DETAILS & SPECIFICATIONS*

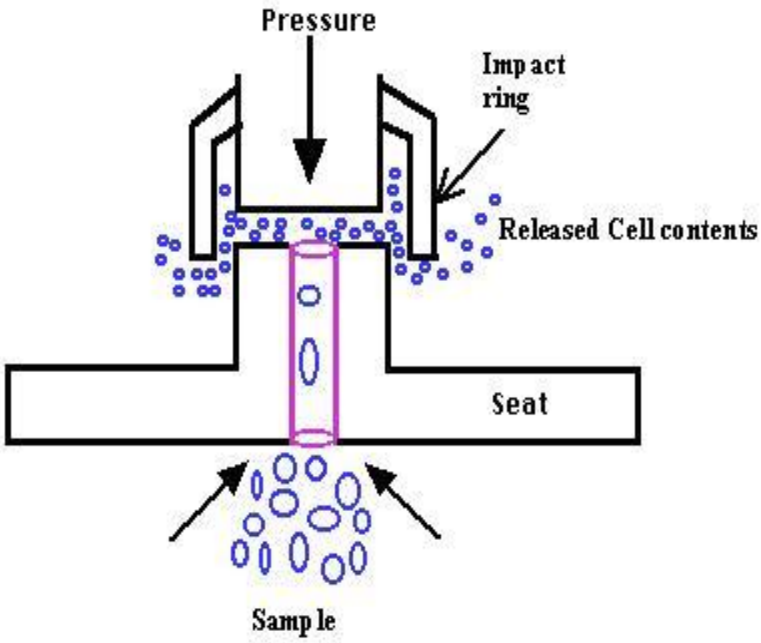
A homogenizer is a piece of laboratory or industrial equipment used for the homogenization of various types of material, such as tissue, plant, food, soil, and many others. Many different models have been developed using various physical technologies for disruption.

Homogenization is a very common sample preparation step prior to the analysis of nucleic acids, proteins, cells, metabolism, pathogens, and many other targets.

5.6.1.1 Working principle

High Pressure Homogenization is a process of increasing the consistency of a product by means of dispersions. The product is displaced under the generation of high pressure and is forced through homogenizing valve gap. Cavitation's turbulence and sheer force break the product into particles of size less than 1 microns.





Acting on high-pressure principle, the instrument is used for extruding cells, especially suitable for smashing thick-wall cells, germs and denser solution samples. Having no noise, less temperature rise and no 28 metal ion contamination, it has wide application of such research fields as protein study, nucleic acid extraction, cell disruption in genetic engineering labs of colleges, scientific research institutes and pharmaceutical factories.

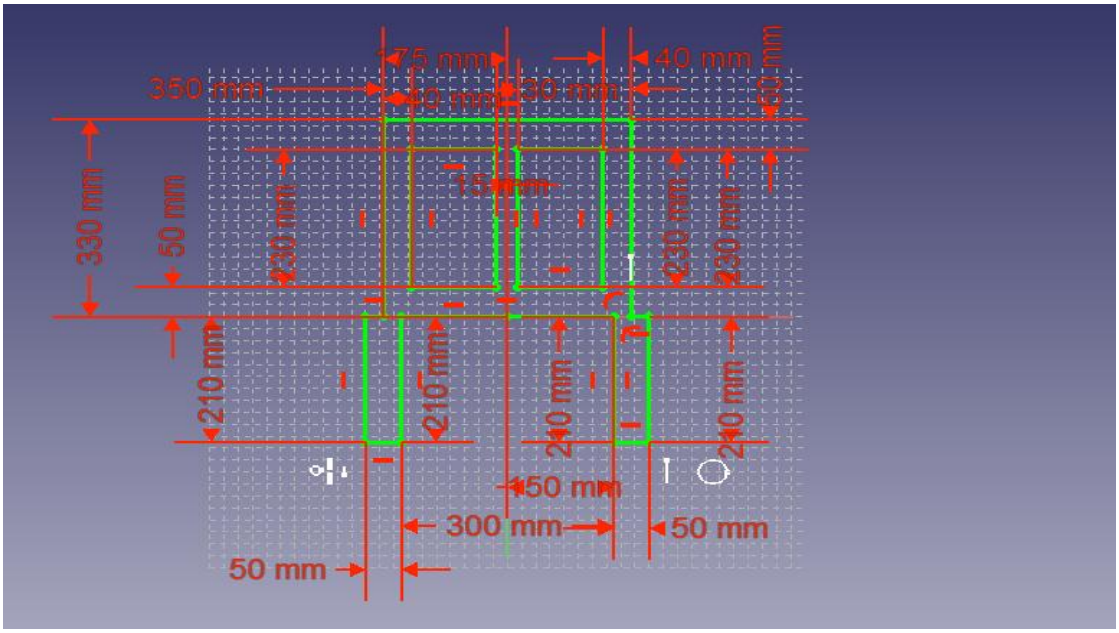
5.6.1.2 Specifications

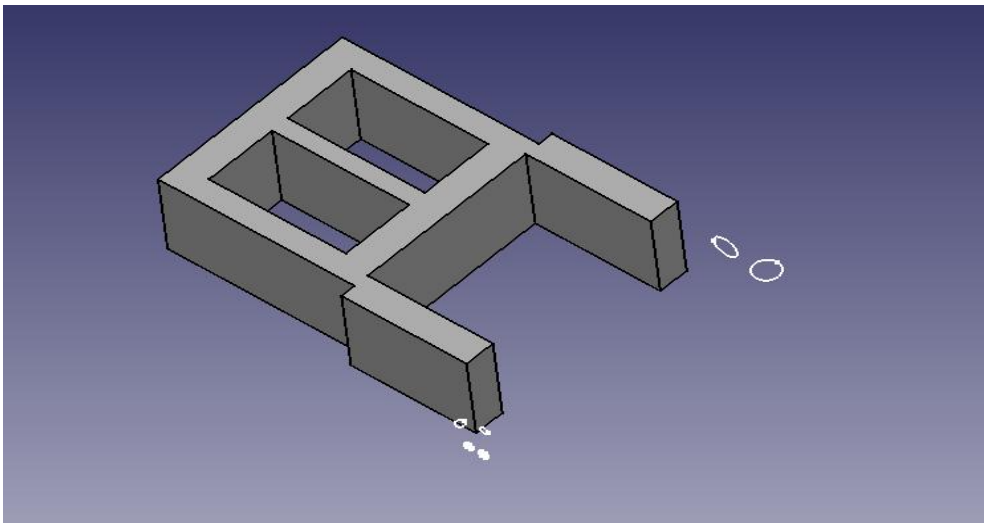
MODEL	JG-IA	CAPACITY	50/ML SCIENTIFIC RESEARCH
VOLTAGE	380 VAC	MAX OPERATING PRESSURE	256 MPA
PRESSURE DEVICE	HYDRAULIC SYSYTEM	MAX PRESSURE STROKE	170 MM
SAMPLE TUBE	Ø25 MM STAINLESS	PRESSURE PLATE SPEED	6.8 MM/S
DIMENSIONS	800×910×810	LARGEST VOLUME SAMPLE	50ML/TIMES

5.6.2 *PART DIMENSIONS & DESIGN*

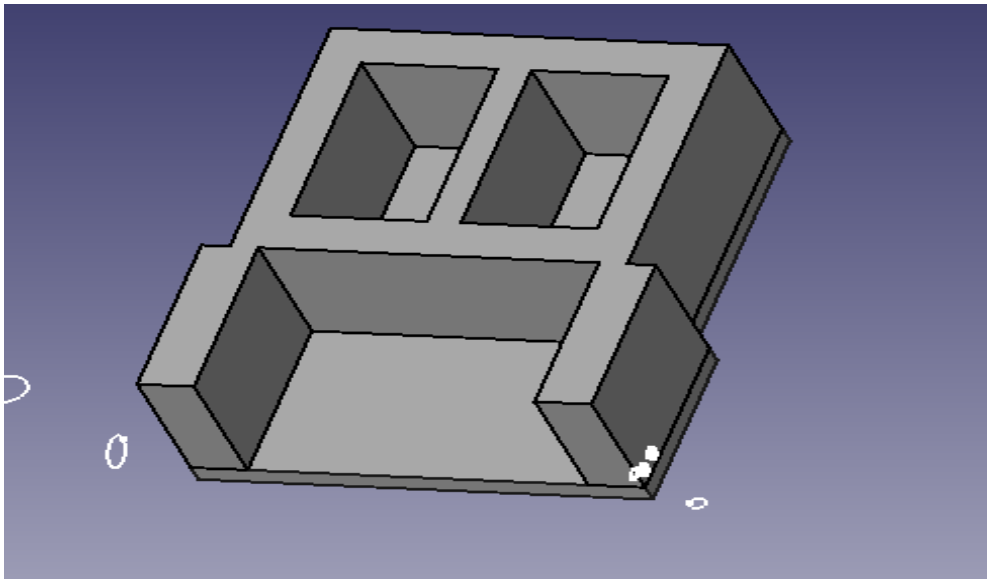
5.6.2.1 BASE PART

-box (sketch) with thickness =175 mm that contains the crankshaft and pistons to operate the process (breaking cells).



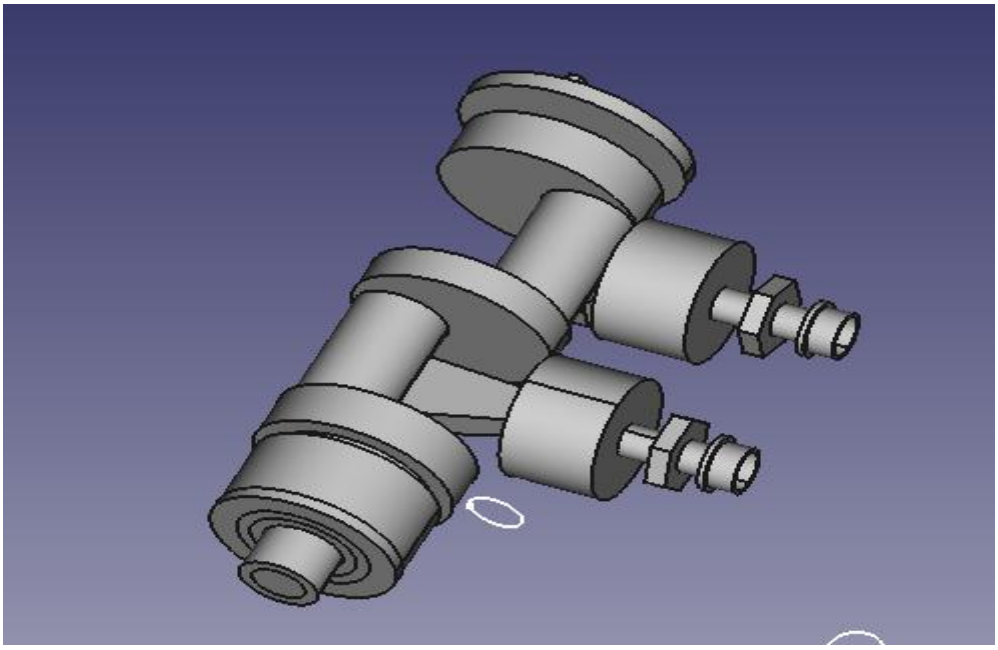


-Cover for Base with thickness =25mm

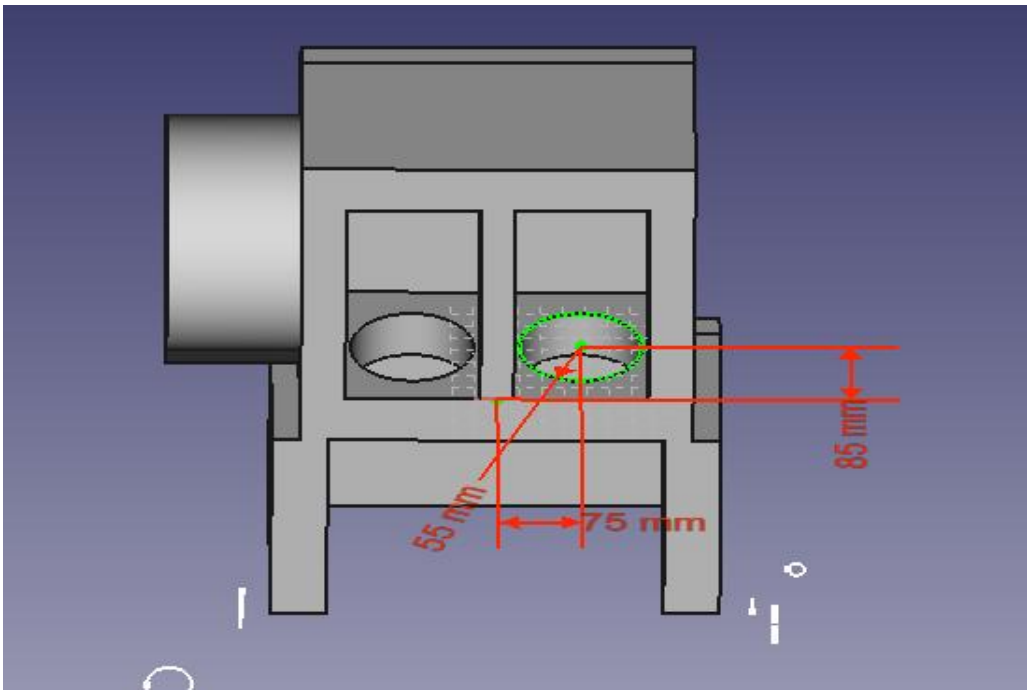


5.6.2.2 CRANK SHAFT AND PISTONS

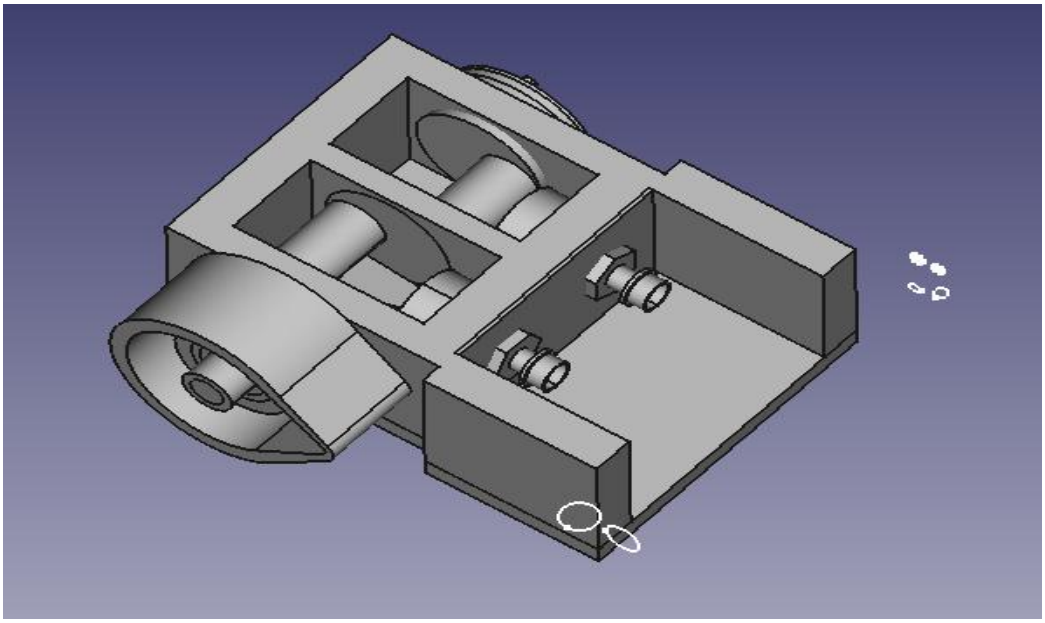
-Taking into account the dimensions of the shaft and pistons according to the box dimensions.



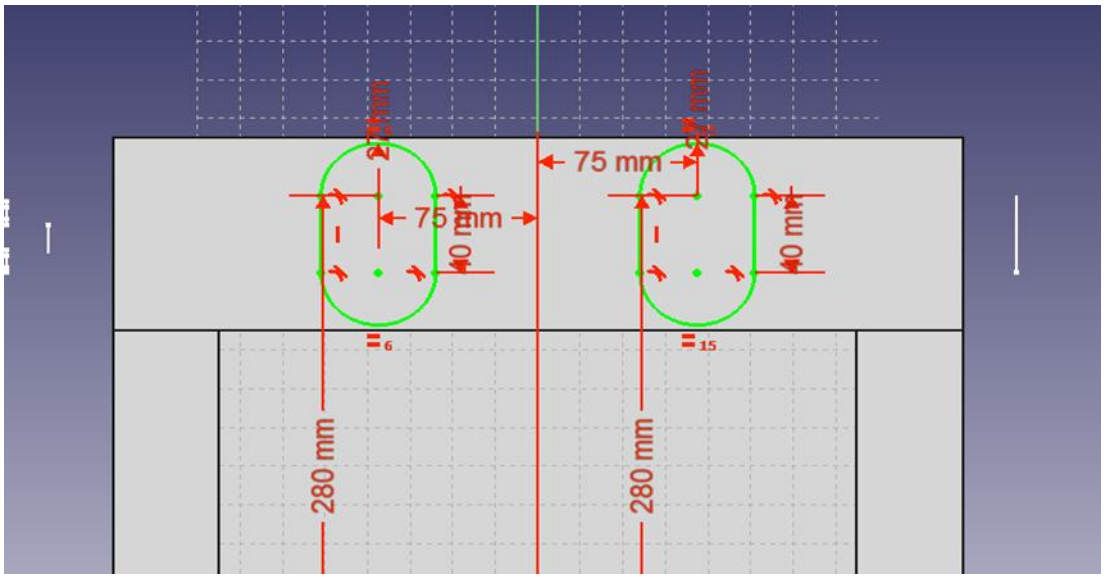
5.6.2.3 Pocket circle with R= 55mm to pass the cylinder of the pistons

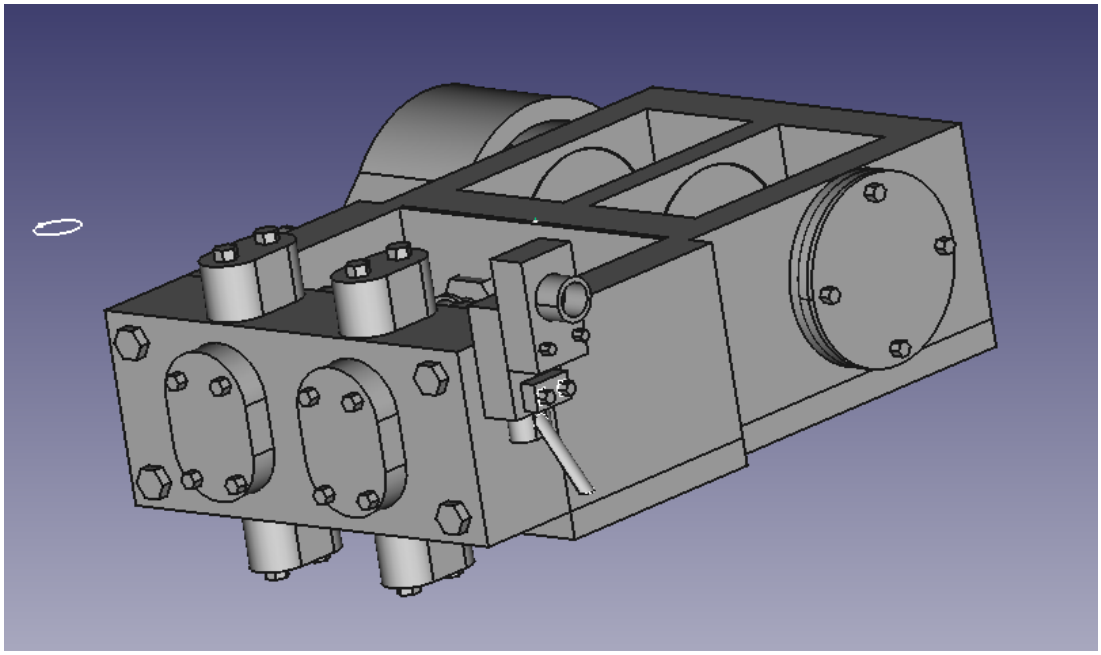


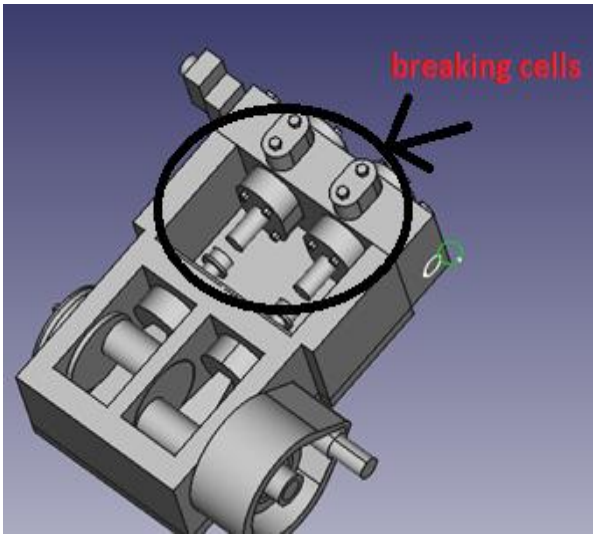
5.6.2.4 With same sketch for pistons and box bind the two parts together.

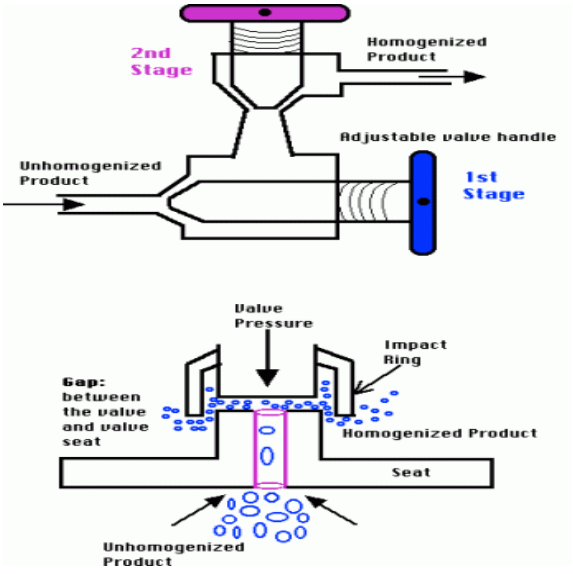


5.6.2.5 Draw the head of the machine where the cell will break due to the pistons and valve.

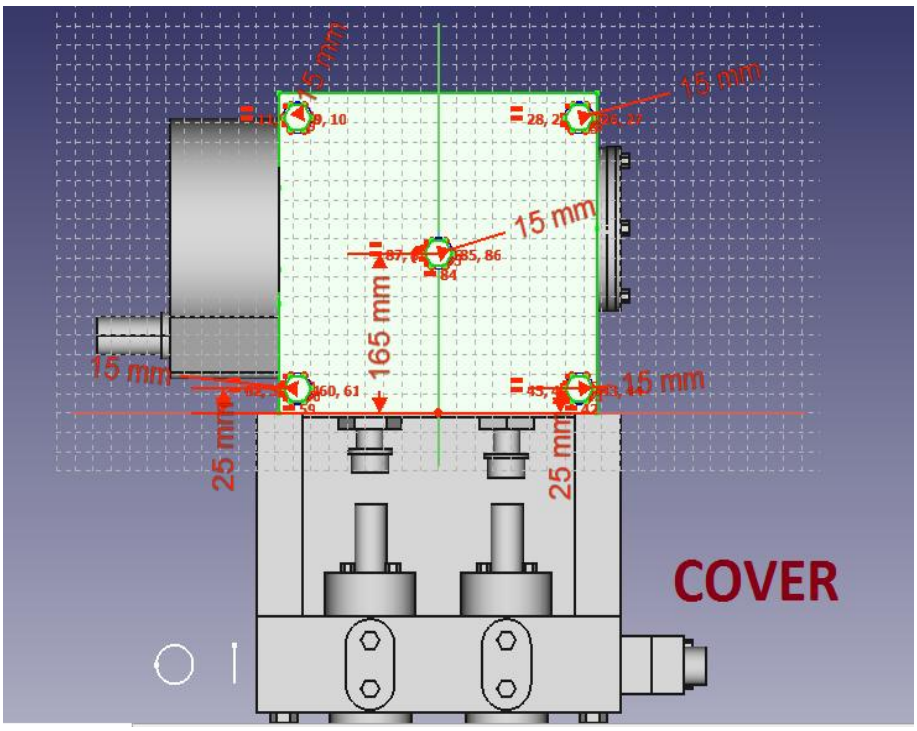


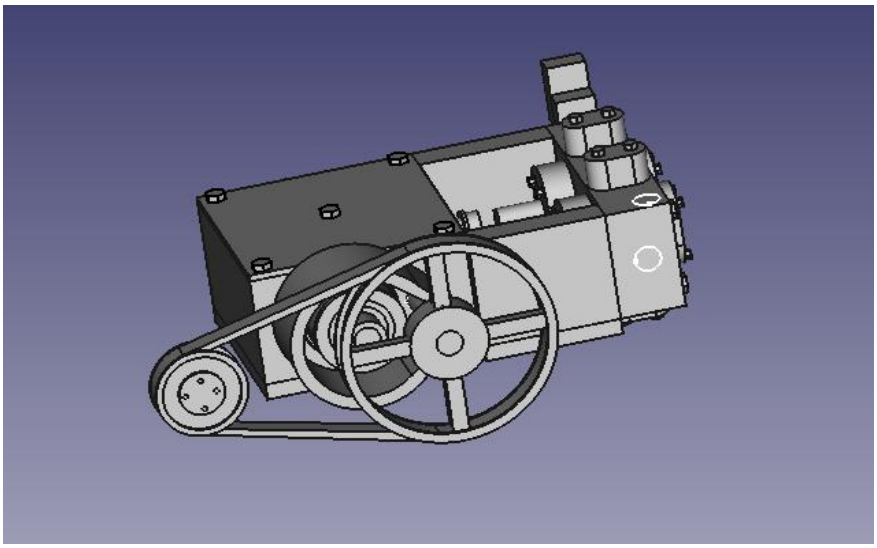






5.6.2.6 Putting the cover over the box and fixed it with bolts and added the shaft comes out from crank shaft and connecting to pulley, this pulley also connected to other one with belt.

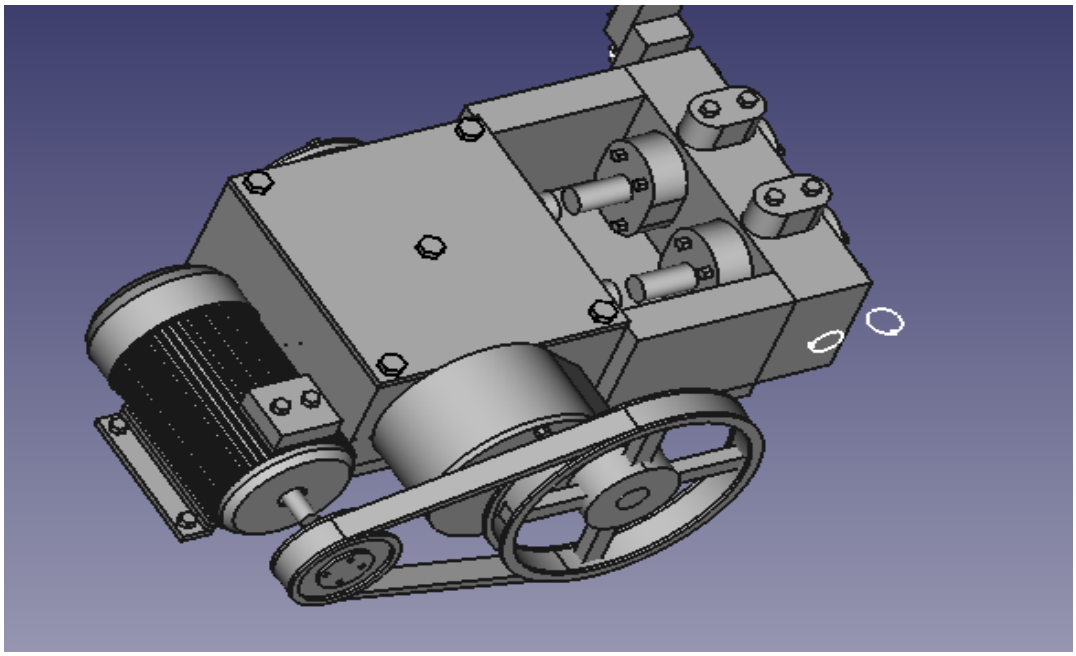




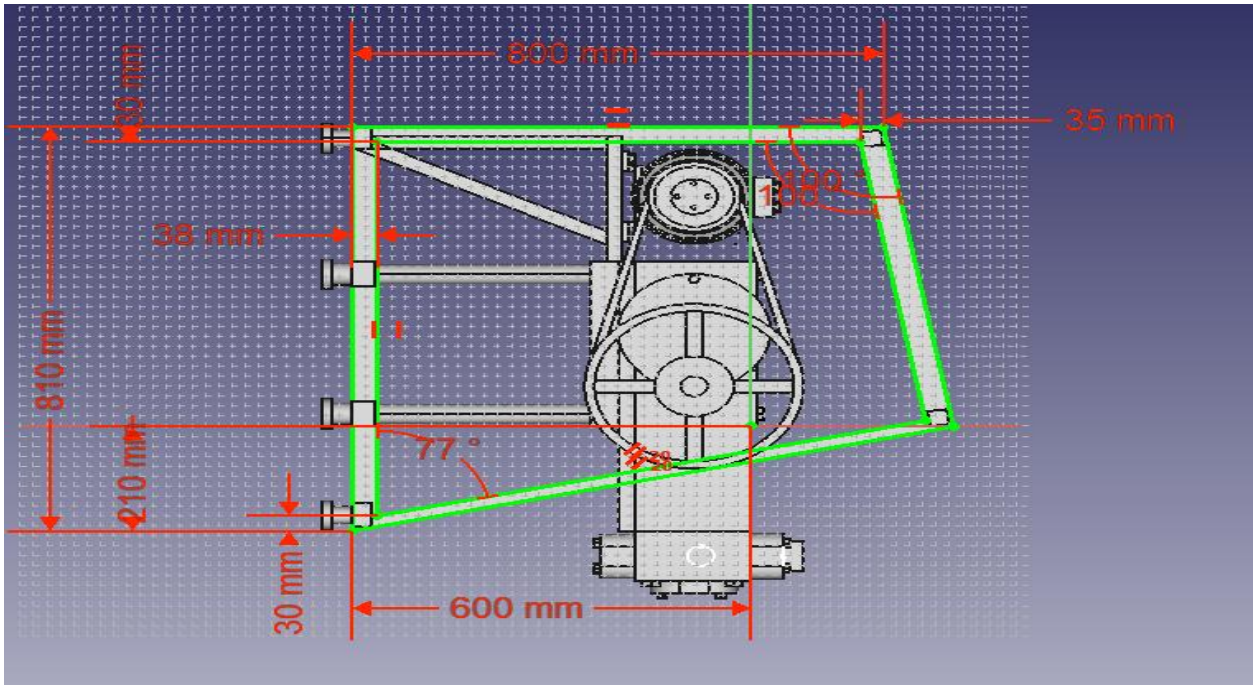
-belt thickness =10 mm

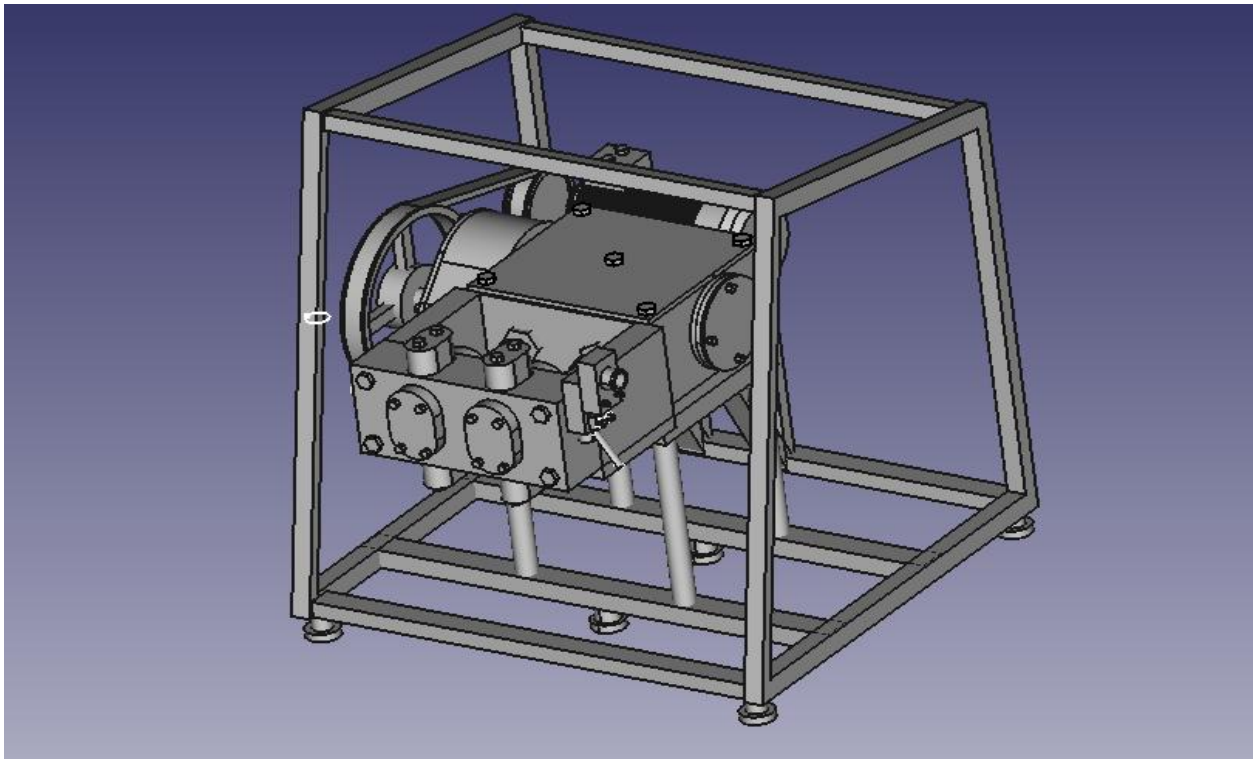
-Large circle with R=165 mm and small R=70 mm

5.6.2.7 Select a motor drive and connected to the small pulley to drive the second and the pistons start work

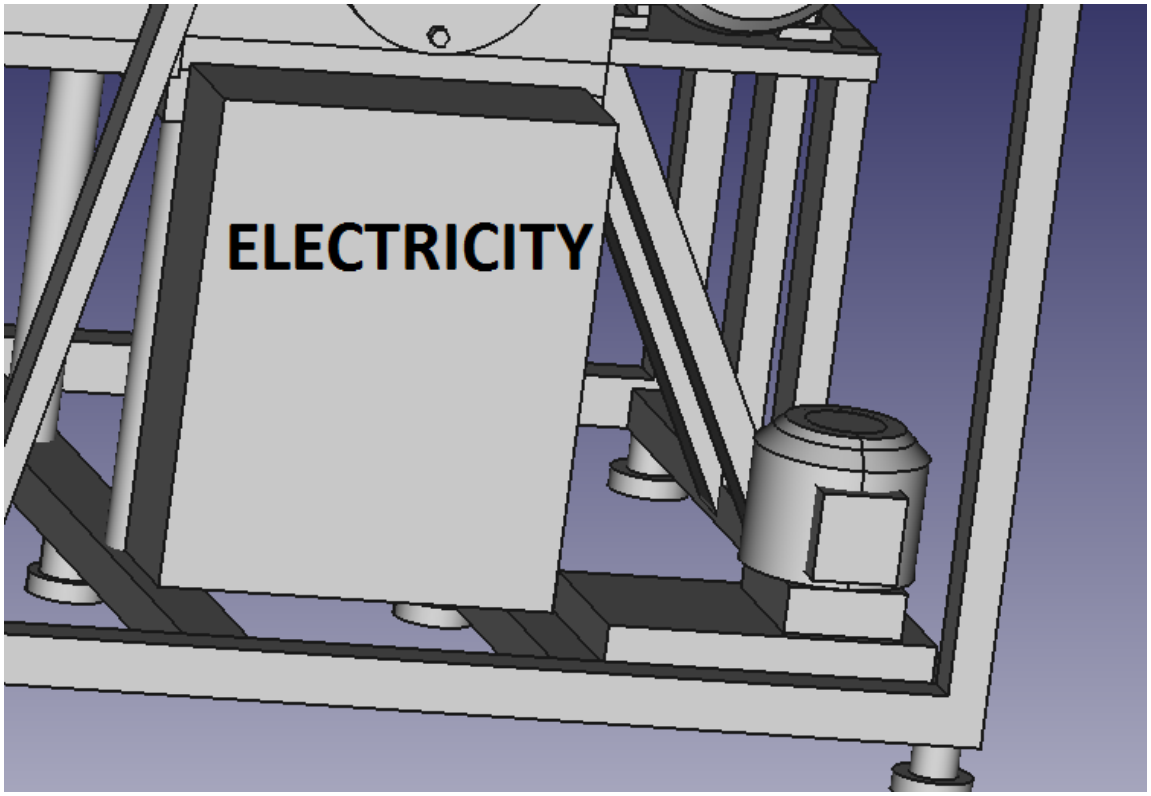


5.6.2.8 Drawing the boundaries of the machine and the basement for the box

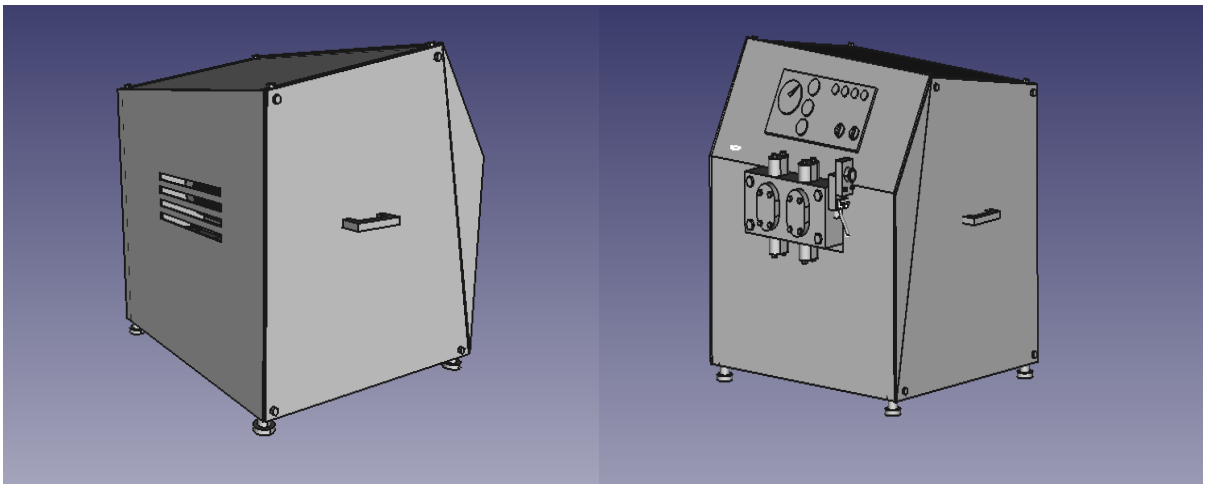




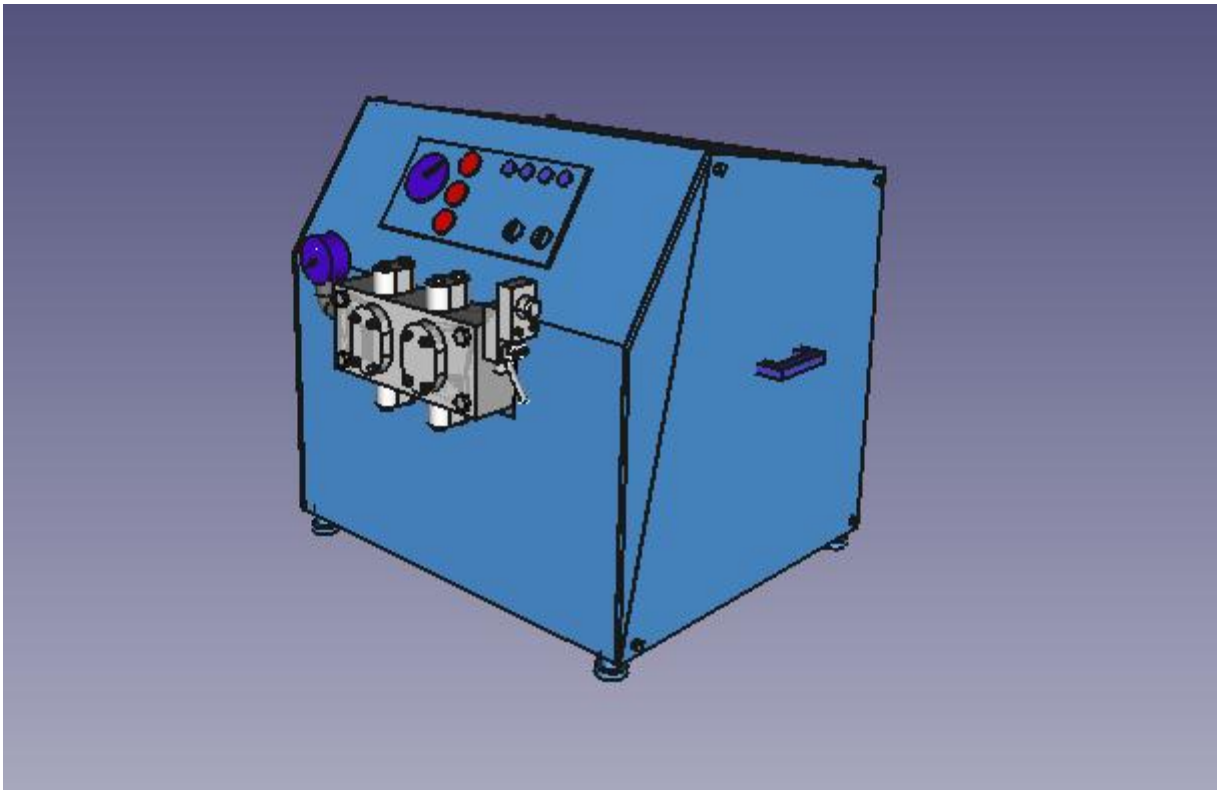
-Small motor and electricity box



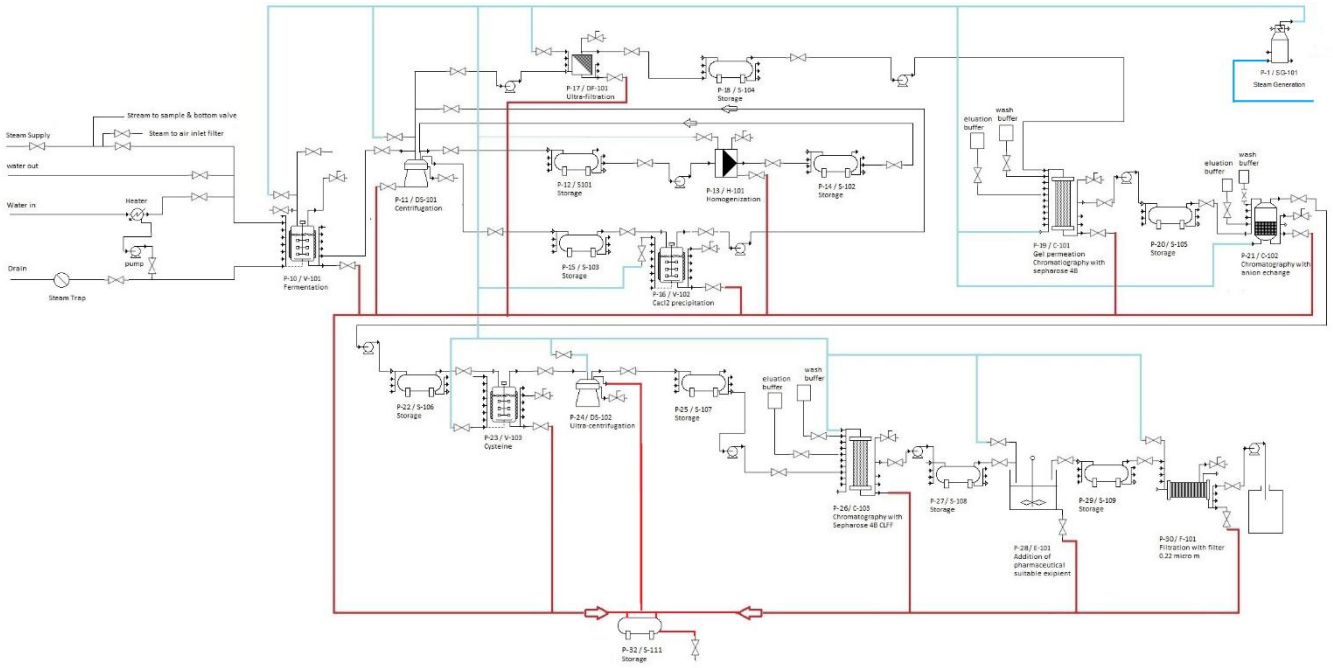
5.6.2.9 Finally close the machine with stainless wall



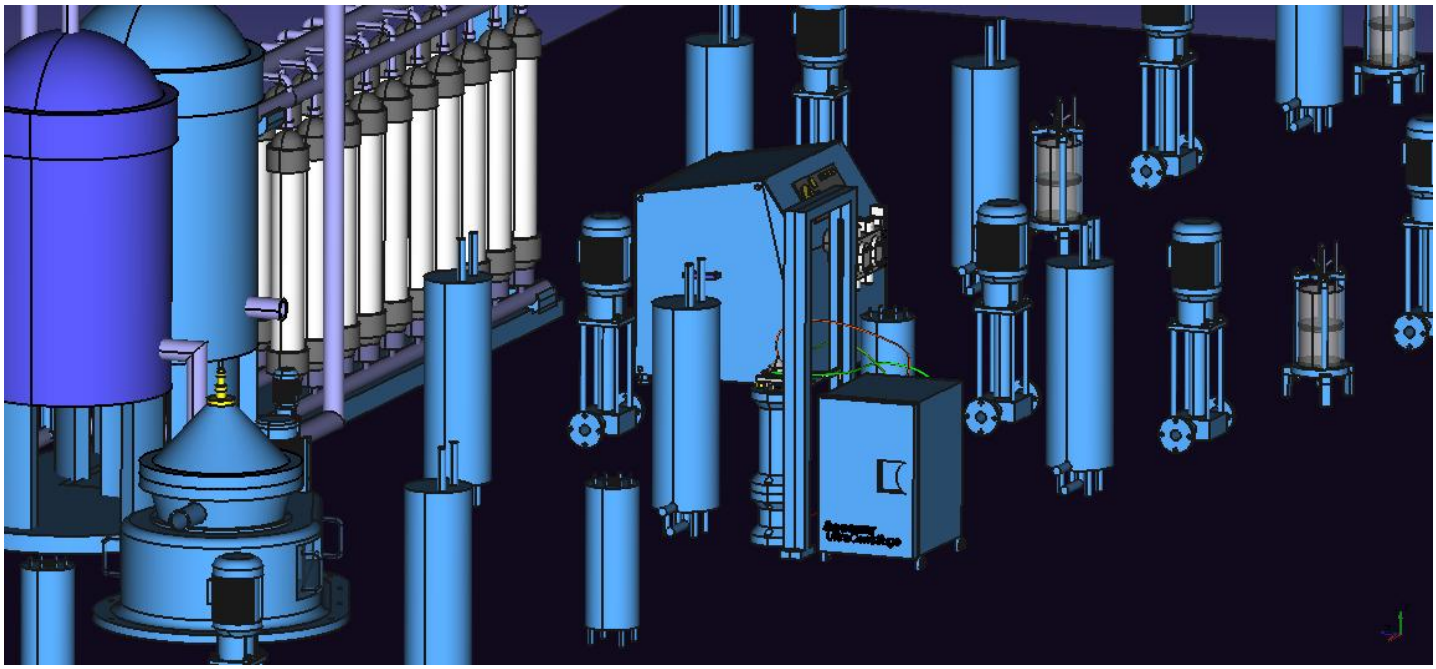
FINAL IMAGE: HOMOGENIZER

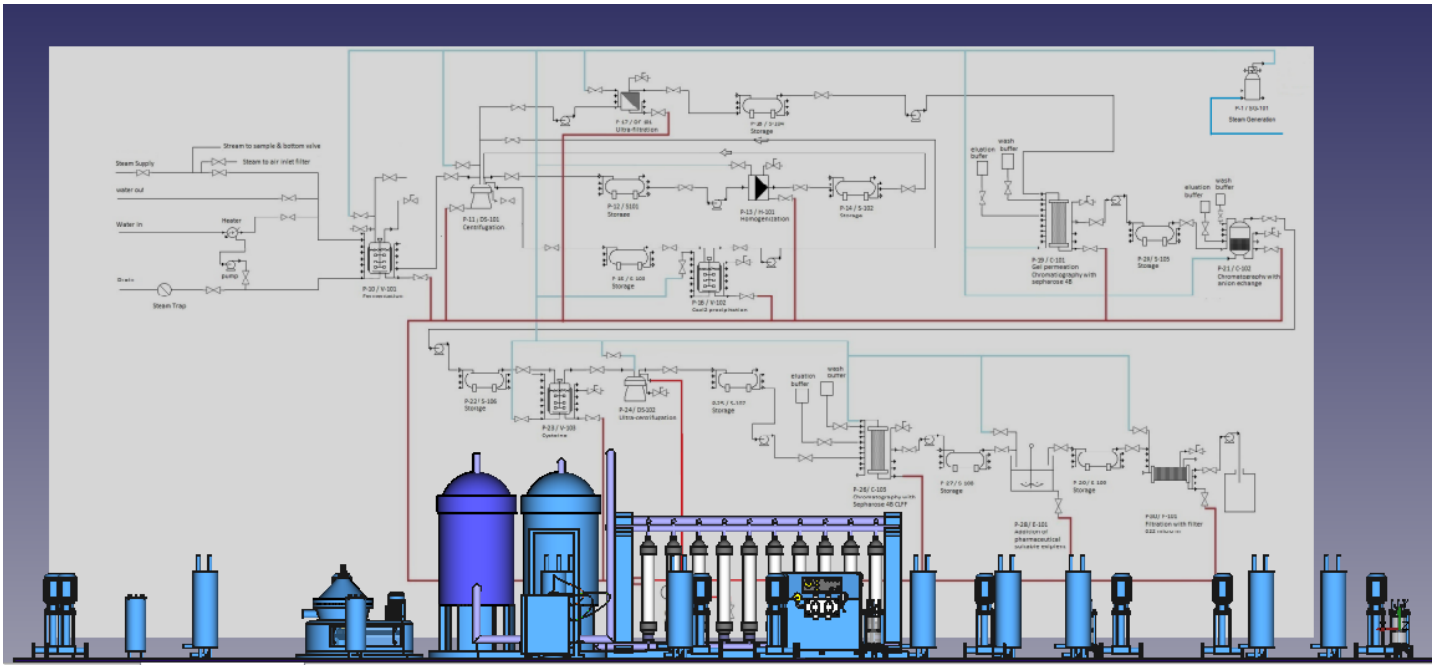


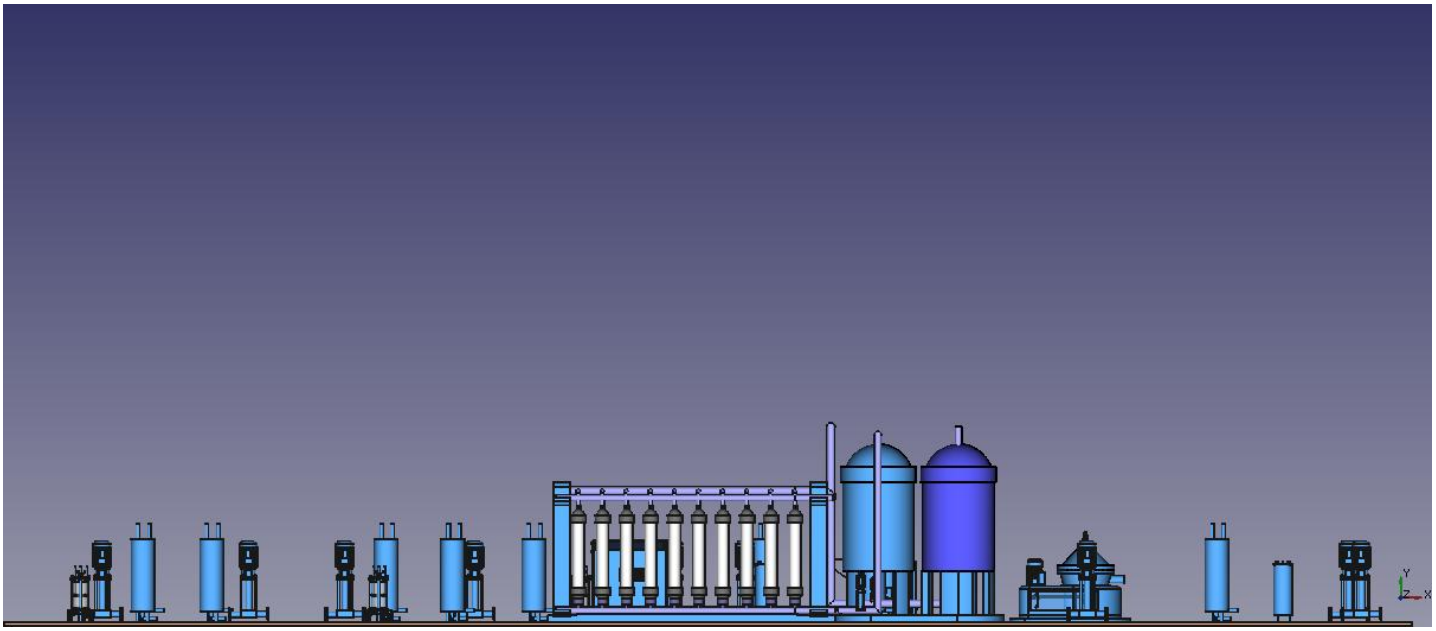
5.7 **Final Assembly of the MEGBI Plant**

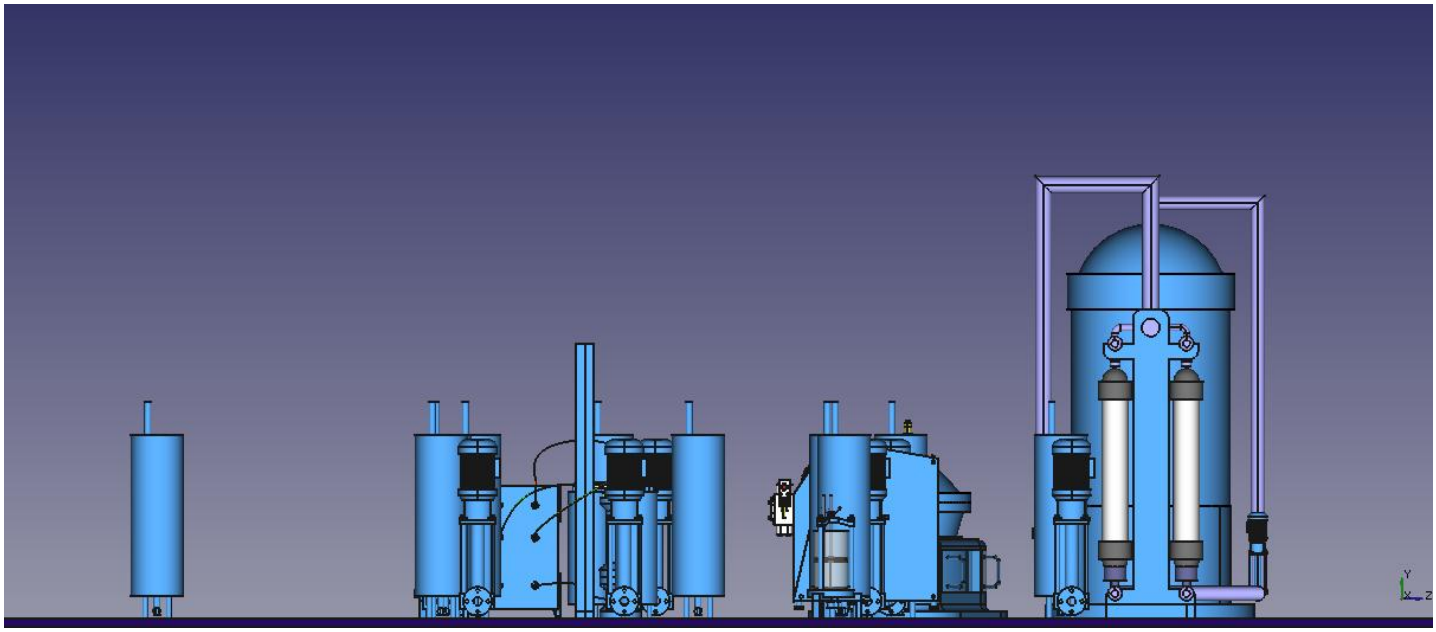


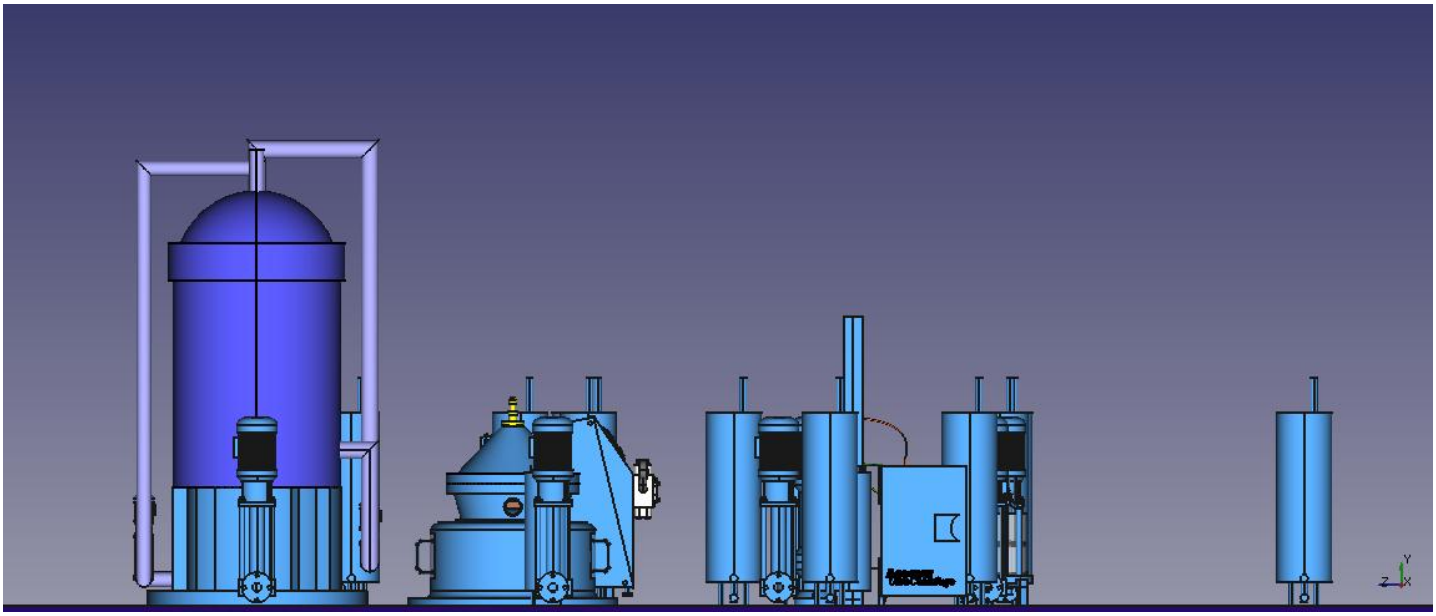
"Due to the lack of Assembly Features in FREECAD, Assembling These Parts Together. Was harder than Modelling Itself "– Jihad Samarji

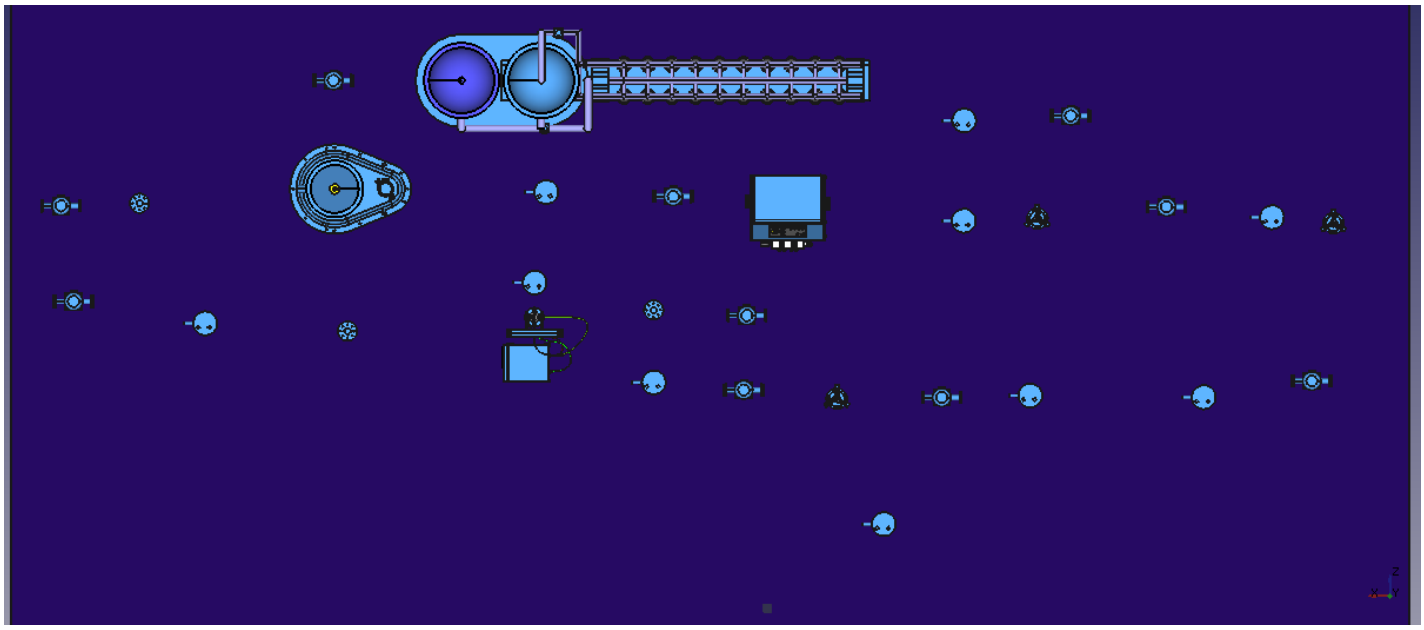


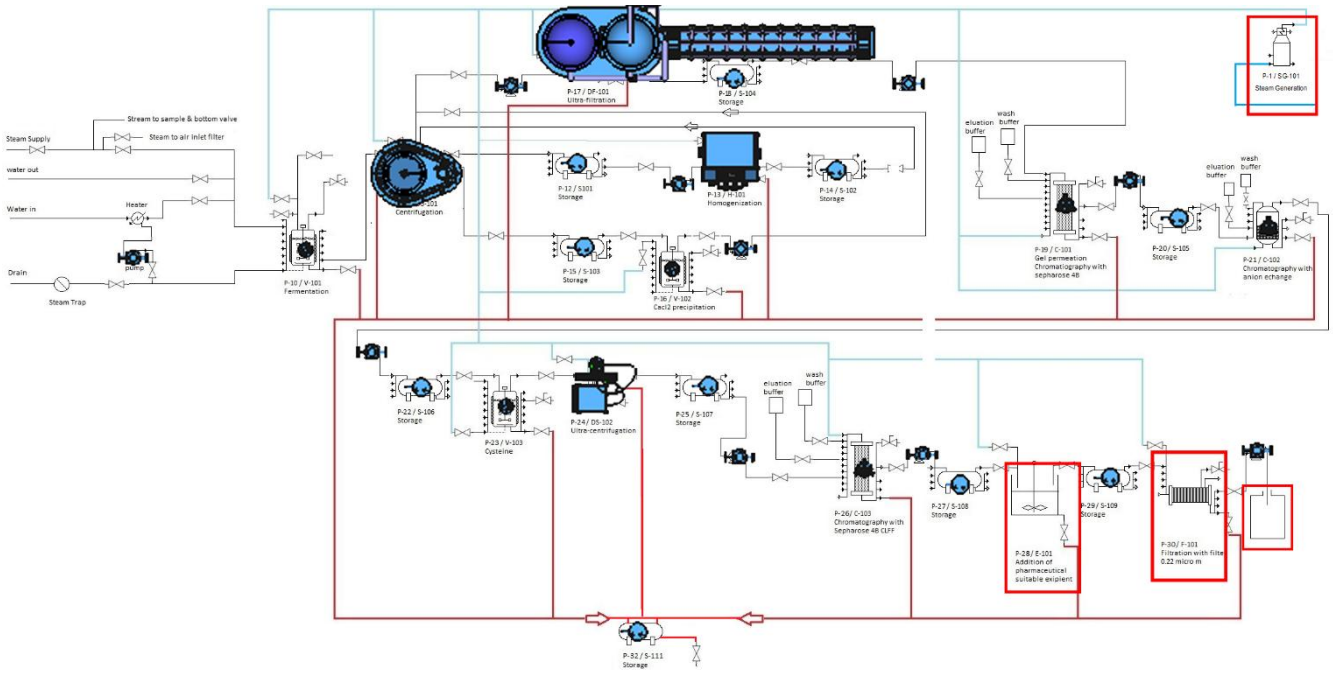

Steam to sample & bottom valve
Steam to air inlet filter
Steam Supply
water out
Water in
Heater
pump
Drain
Steam Trap
P-11 / DS-101
Centrifugation
P-12 / S101
Storage
P-13 / H-101
Homogenization
P-14 / S-102
Storage
P-17 / DF-101
Ultra-filtration
P-18 / V-104
Storage
P-18 / E-103
Storage
P-19 / V-102
P-20 / V-101
P-21 / S-106
Storage
P-22 / S-106
Storage
P-23 / V-103
Centrifuge
P-24 / DS-102
Ultra-centrifugation
P-25 / E-107
Storage
P-26 / C-103
Chromatography with Sepharose 4B-CL6
P-27 / S-108
Storage
P-28 / E-101
P-29 / C-101
Gel permeation Chromatography with sepharose 4B
P-30 / E-104
Storage
P-31 / S-105
Storage
P-32 / C-102
Chromatography with anion exchange
P-33 / F-101
Filtration with filter
P-1 / SG-101
Steam Generation
elution buffer
wash buffer

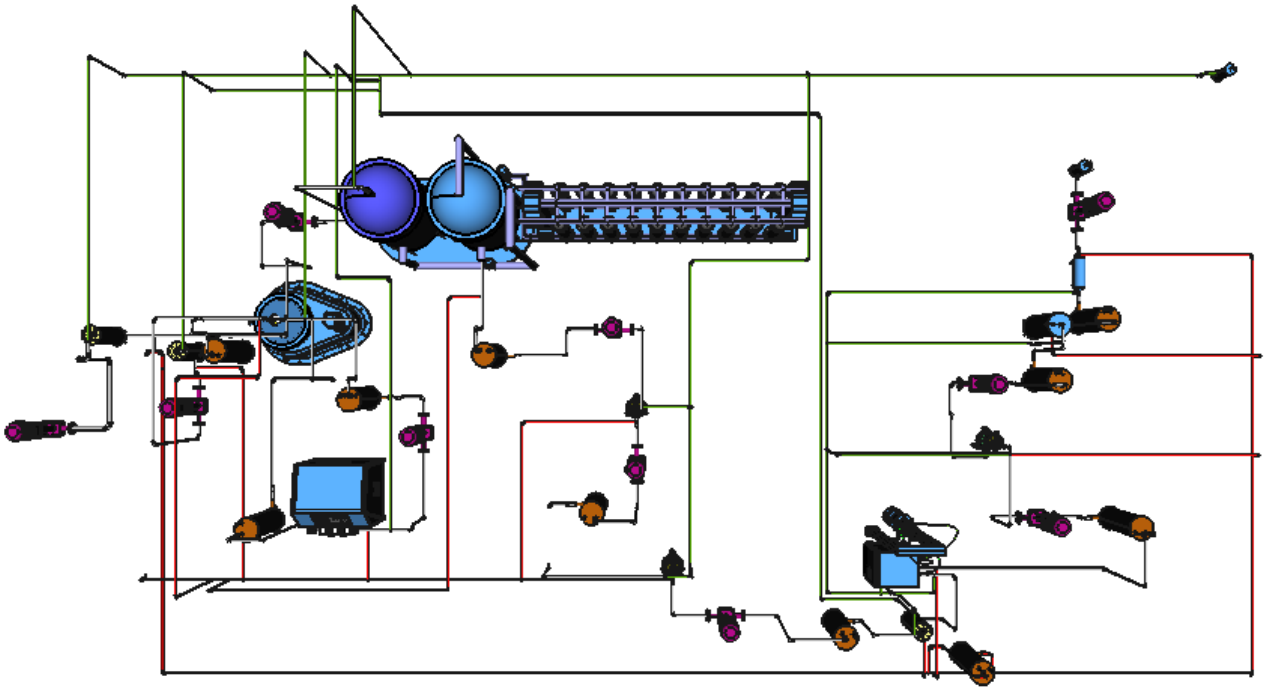


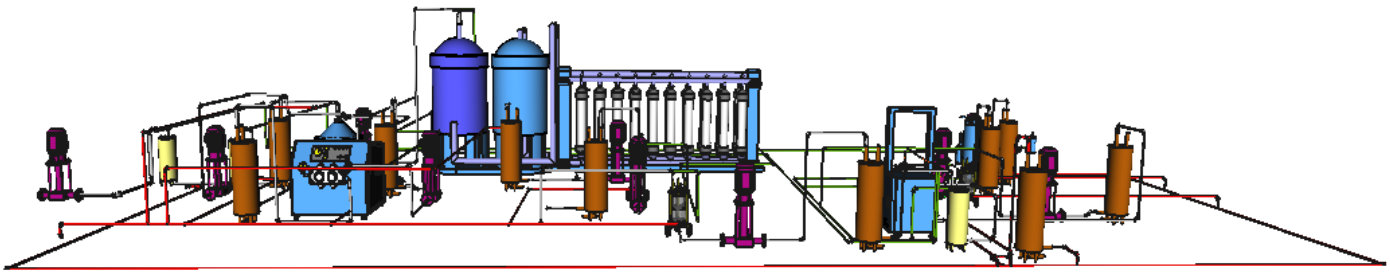







Stream to sample & bottom valve
Steam to air inlet filter
Steam Supply
water out
Water in
Heater
pump
Drain
Steam Trap
P-10 / V-101
Fermentation
P-17 / DF-101
Ultra-filtration
P-18 / S-104
Storage
P-1 / SG-101
Steam Generation
P-12 / S101
Storage
P-13 / H-101
Homogenization
P-14 / S-102
Storage
P-15 / S-103
Storage
P-35 / V-102
CaCl2 precipitation
eluation buffer
wash buffer
P-39 / C-101
Gel permeation Chromatography with sepharose 4B
P-20 / S-105
Storage
P-21 / C-102
Chromatography with anion exchange
P-22 / S-106
Storage
P-23 / V-103
Cysteine
P-24 / DS-102
Ultra-centrifugation
P-25 / S-107
Storage
P-26/ C-103
Chromatography with Sepharose 4B CLFF
P-27 / S-108
Storage
P-28 / E-101
Addition of pharmaceutical suitable exipient
P-29 / S-109
Storage
P-30/ F-101
Filtration with filte 0.22 micro m
P-32 / S-111
Storage





MODELLING AND ASSEMBLING MEGBI-VPP IS DONE

NEXT STEP: IS TO MAKE AN AUTOMATION SYSTEM TO SIMULATE THE WORK OF THIS PLANT

THIS CHAPTER CONTAINS INFORMATION GATHERED FROM PERSONAL EXPERIENCE AND THE INTERNET.

6. Automation System Design and GUI

This task was done as master thesis from Haitham Hindy, Lebanese University, Tripoli

Master Thesis

Automation of measurement of temperature, pressure and pH data and automation of fluid flow of a biotechnological production plant

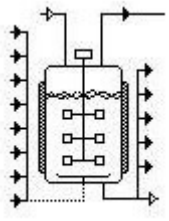
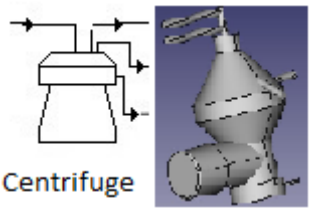
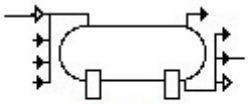
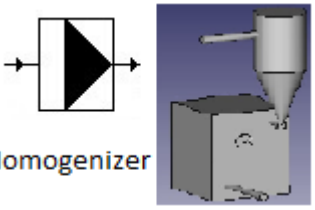
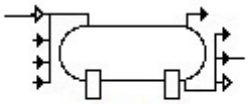
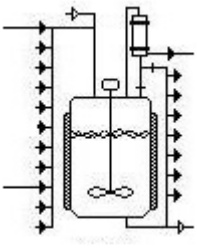
- Design of Software (State machines) for Homogenizer, Disc Stack Centrifuge in including CIP/SIP functional elements, Process Scale Gel Permeation and ion exchange chromatographic devices, Process Scale Ultrafiltration Device (6 weeks)
- Graphical User Interface for the automation of MEGBI-VPP downstream processing (DSP) unit (4 weeks)
- Adaptation of a Graphical User Interface to a Siemens S7 PLC system (6 weeks)
- Documentation (3 weeks)

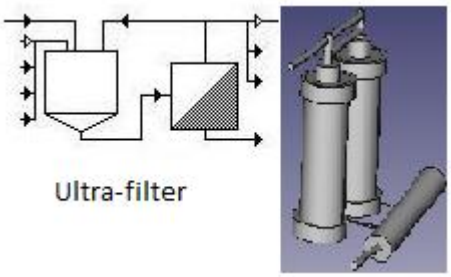
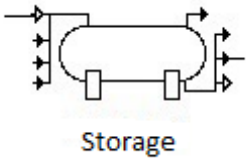
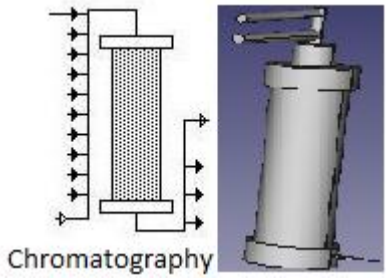
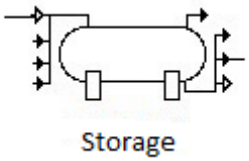
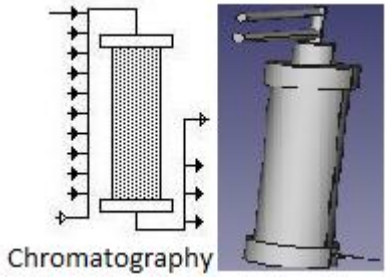
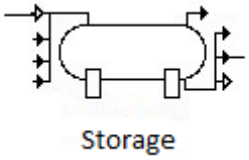
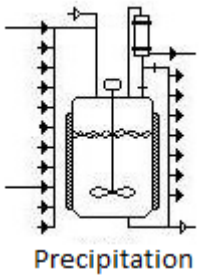
Keywords: measurement of temperature, pressure and pH data, Automation of fluid flow, PLC, Siemens S7, Programming, User Interface, C++/Java, Biotechnology

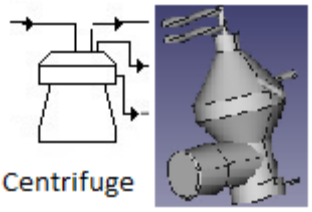
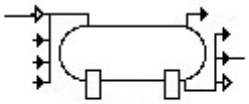
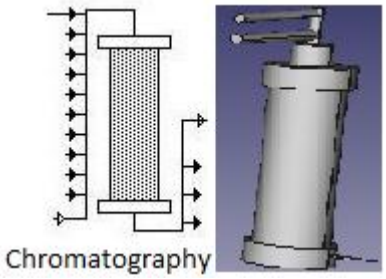
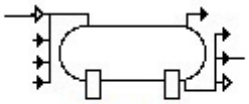
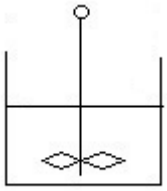
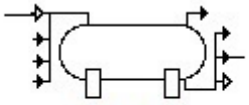
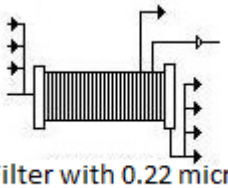
For automation system basics please refer to chapter 2, sections 15-17

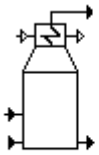
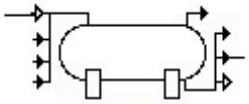
6.1 Automation System Specification

*Table 3: List of Process **devices** with machine, PI diagram symbol and instruments needed.*

PROCESS NUMBERS	DEVICES NUMBERS	PI DIAGRAM SYMBOL/CAD MODEL	VALVES / PUMPS	SENSORS
P-10	V-101	 Bioreactor	V1 V2 V3 V4 V5	T1 PH1 MF1 PR1 PO1
P-11	DS-101	 Centrifuge	V1 V2 V3 V4 V5 V6	T2 MF2 PR2
P-12	S-103	 Storage	V1	---
P-13	H-101	 Homogenizer	V1 V2 V3 PU1	T3 MF3 PR3
P-14	S-102	 Storage	V1	---
P-16	V-102	 Precipitation	V1 V2 V3 V4 V5	T4 MF4 PR4

P-16	DF-101		V1 V2 V3 V4 V5 PU2	T5 MF5 PR5
P-17	S-103		V1	
P-18	C-101		V1 V2 V3 V4 V5 V6 PU3	T6 MF6 PR6
P-19	S-104		V1 PU4	
P-20	C-102		V1 V2 V3 V4 V5 V6	T7 MF7 PR7
P-21	S-105		V1 PU4	
P-22	V-103		V1 V2 V3 V4 V5	T8 MF8 PR8

P-23	DS-102		V1 V2 V3 V4	T9 MF9 PR9
P-24	S-106		V1	
P-25	C-103		V1 V2 V3 V4 V5 V6 PU5	T10 MF10 PR10
P-26	S-107		V1	
P-27	E-101		V1 V2 V3 V4	
P-28	S-108		V1	
P-29	F-101		V1 V2 V3 V4 V5	T11 PR11 MF11

P-30	S-110	 Steam generater	V1	P1
P-31	S-109	 Storage	V1	---

With this list can make the screen, we have painted a miniature in a paper A4 link the instruments together in order to process the next process and we have introduced the valves to get this:

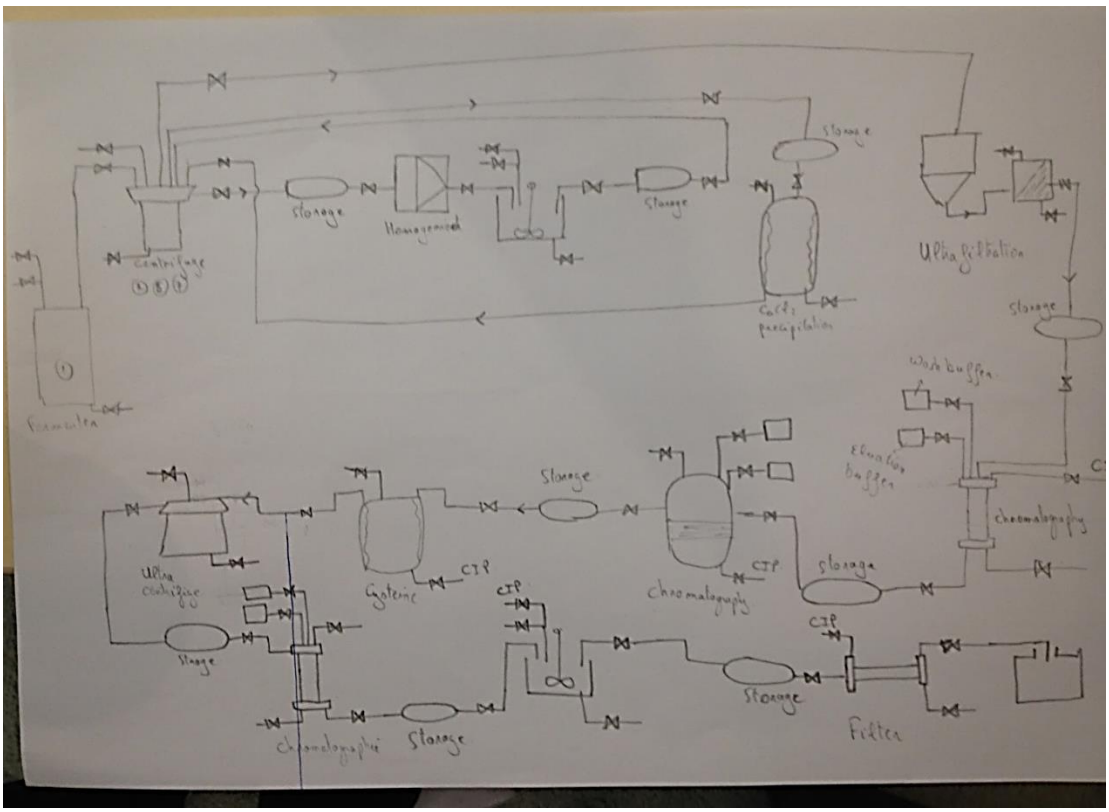
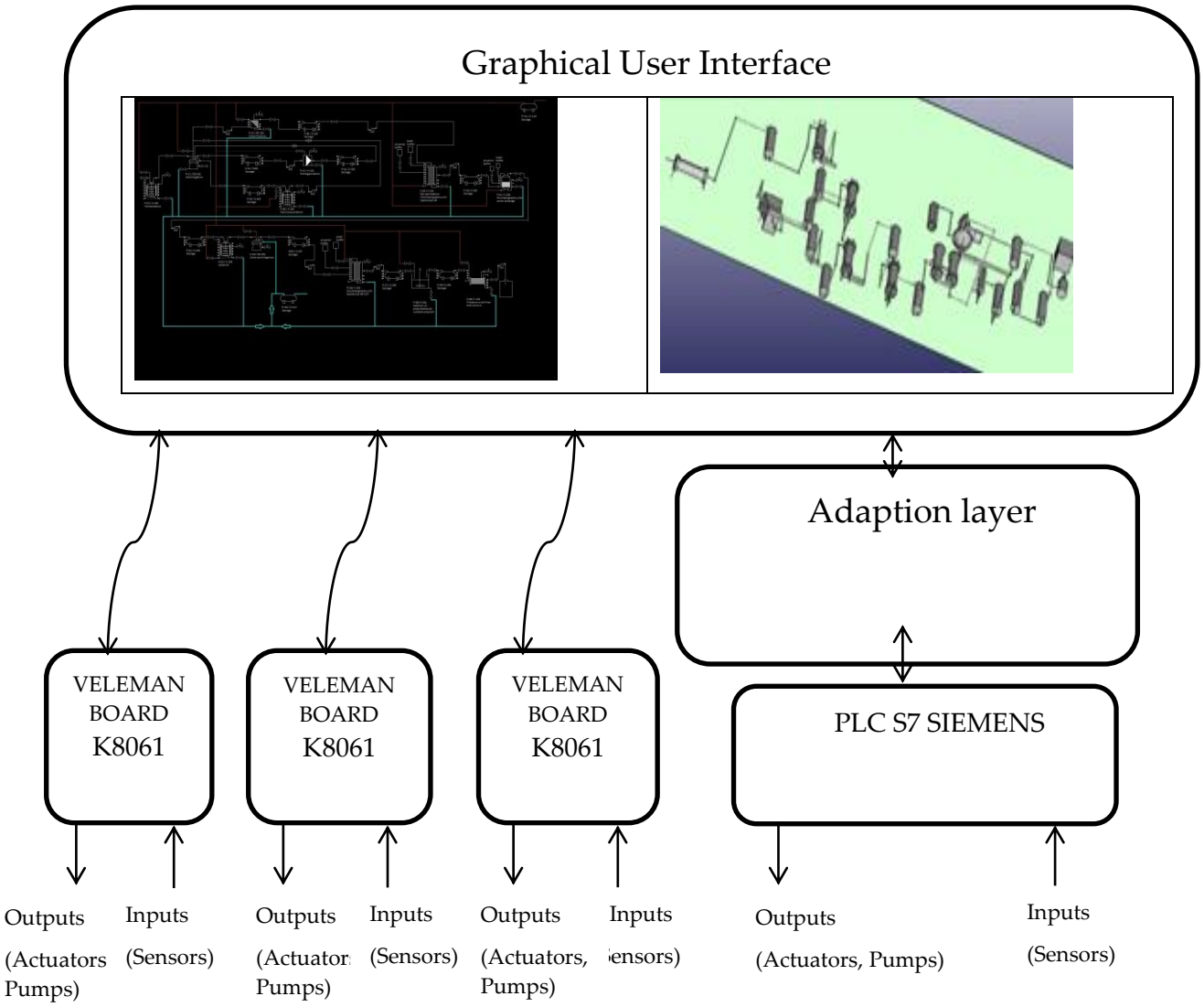


Figure 3: *Painting the Downstream Processing.*

Next step we have used this image to create the graphical user interface, we have collected the SuperPro with paint to make all instruments and process together and get the image:


Graphical User Interface
Adaption layer
VELEMAN BOARD K8061
VELEMAN BOARD K8061
VELEMAN BOARD K8061
PLC S7 SIEMENS
Outputs (Actuators Pumps)
Inputs (Sensors)
Outputs (Actuator Pumps)
Inputs (Sensors)
Outputs (Actuators, Pumps)
Inputs ensors)
Outputs (Actuators, Pumps)
Inputs (Sensors)

6.2 Graphical User Interface

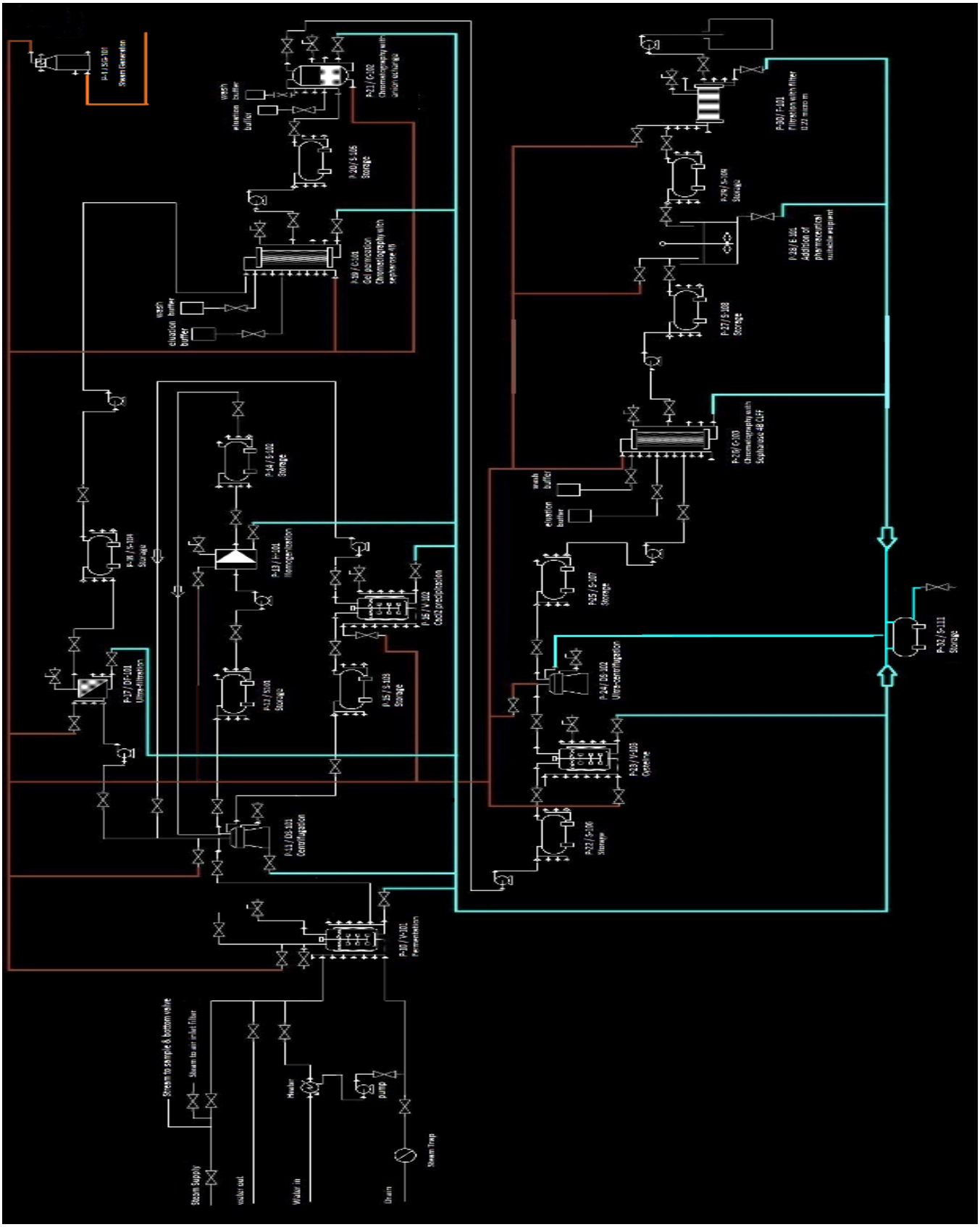


Figure 2.4: *PI Diagram which represents the MEGBI-VPP Processing.*

So by using the concepts and the specification of the production for the vaccine, we have organized the PI diagram for the MEGBI-VPP Processing and finally we get the PI diagram full needed to continue our project and to be the first step that has done and now this PI diagram (figure 2.1) represents the MEGBI-VPP Processing. On this PI diagram we have putted 3 pipelines white represents the road which will pass by the vaccine and some materials adding in many machines, we thought about cleaning the machines, everything need clean after use to get a perfect production, for that we have created the 2 next pipelines red and blue, the red one is the road of the steam water coming from a steam generator and the other one is the road of the dirty water ,it's the dirt coming from the production of the vaccine hepatitis b and beside the bioreactor we have added the temperature control system specific for it ,we have putted the valves necessary on the pipes to controls the in and out of fluid for the machine , and we have putted the pumps but we make look about the position that we have to use pump , so as we know if there is a elevation between the machine we need a pump so that is the condition that we have based on it.

Next have to use the PI diagram on Python to create the control screen, we have passed by many problems on python start by how to install, how to run, and how to use, we have searched on internet sites to answer on this questions, and we get the answer needed by the Python book[4] and internet sites[6] then we have used this reference for knowing the codes needed on Python to create this screen and how we should use it, first the idea of the control screen is controlling the fluid flow by open and close the valves, turning on and off the pumps, watching the status of each instruments on this screen for that we have needed buttons, labels texts list box background and window…

The hard job was on installing the Python, run it and how to choose the specific code for each thing we have to create on the screen so we have putted a plan about the things that we will show on the control screen, so as we talk before we need buttons by a specific code on Python we have created 68 buttons, 58 for valves and 10 for pumps , and we have organized it like each valve beside it the button specific for it , so we have identified the position of the valve to know where we will put this button and we did this for the 68 buttons. Before button we have used the PI diagram created with a specific code on Python to the background of the control screen

[4]Book name: Wx Python in Action

and with a specific code too we have created a window for this screen and we have created a bar status for this screen with 2 buttons 'exit' 'about', plus we have written on 'about' information about this screen and what it's represents. Next we have created two list box (one for valves, other for pumps) concerning the list name of the actuators (valve or pump) beside it we have putted a label for each actuators to shows on it the status of the actuators, this status we will use it to controls the actuators by open/close or turn on/off by a 'ON' or 'OFF' on label, so we have created on Python a condition between the label and button which we can use the click on button to get 'ON' for first click 'OFF' for other for a manual work, and for the full automation we have putted a "Turn ON" button that we will click on it to begin the production process. Now we have the window with the PI diagram background, buttons, and two list box and if we click on button we get the status of it on label in list box this for a manual work and we have the "Turn On" button, So the control screen is done and ready to use (figure2.2). About all specific codes used you can see the full program in Annex2 and see the codes.

Some difficulties during installation

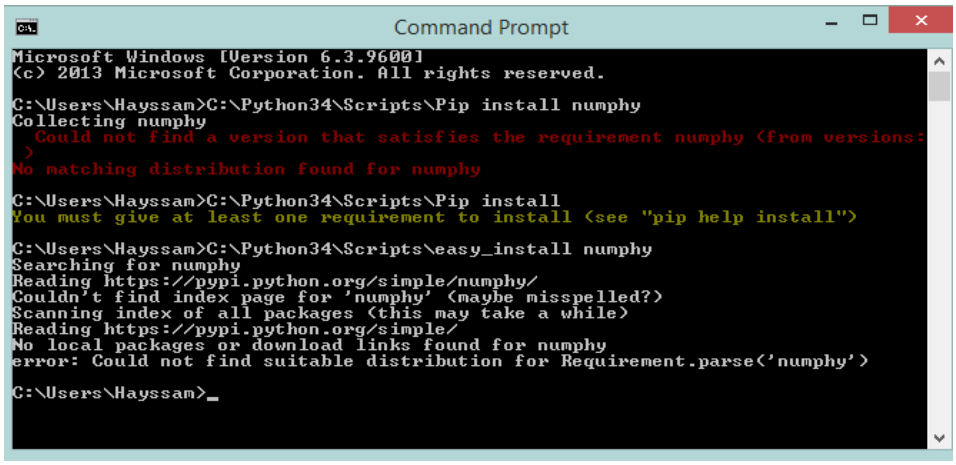


Figure 5 : **Error Command 2**

To repair this error, must upgrade the pip package:

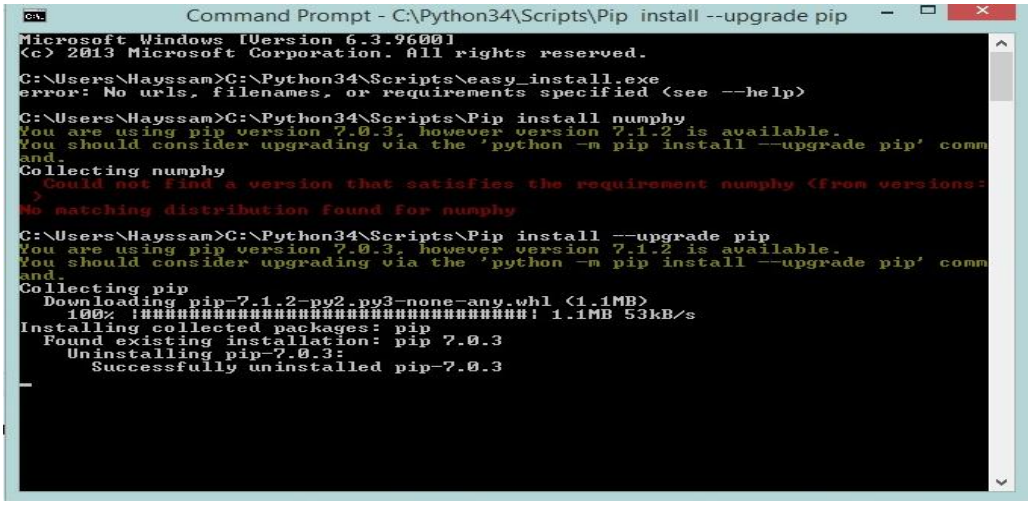


Figure 6: ***First step of upgrading.***

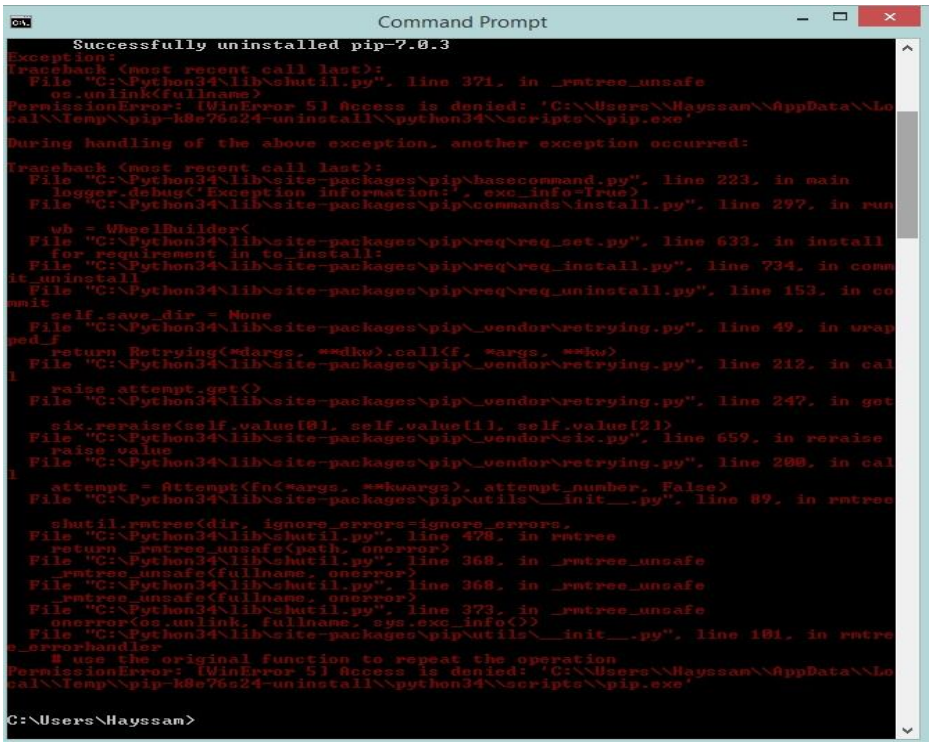


Figure 7 : *Done step of upgrading.*

For complete the installing must define the entire package like that example and run it:

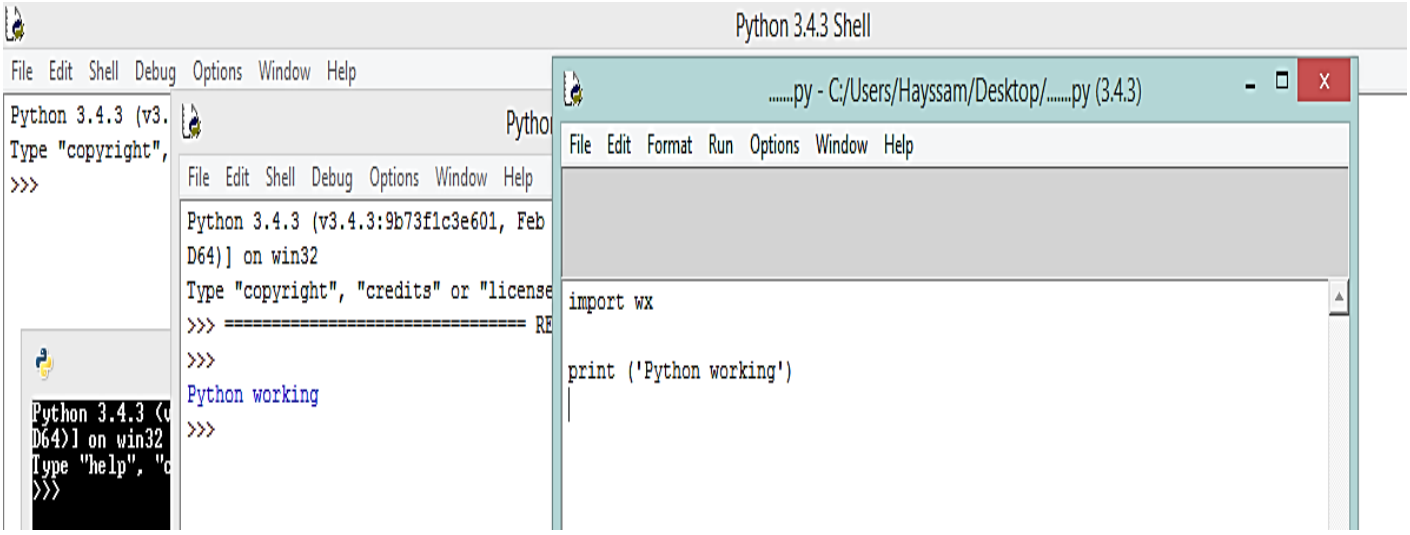


Figure 8 : *Python working.*

After installing and running the Python, must continue to written:

First how to insert a background, that background should be the screen, the previous image was painted must be that background, the Python had a code to use an image like a background we used the code to my image.

This is the code:

```
#**********************                    panel frame                    **********************
#*****************************************************************************************

BACKGROUND_IMAGENAME =\"C:\Users\Hayssam\Desktop\GUI 220415MEGPI-VPP_PI_DIAGRAM.PNG"


class MyBackgroundPanel(wx.Panel):

    def __init__(self, parent):
        wx.Panel.__init__(self, parent)
        self.bmp = wx.Bitmap(BACKGROUND_IMAGENAME)
        self.SetSize(self.bmp.GetSize())
        self.Bind(wx.EVT_PAINT, self.on_paint)


    def on_paint(self, event = None):
        dc = wx.BufferedPaintDC(self, self.bmp)
```

Figure 9 : *Python 1*

After this we had created a window from a code across written a new class, use the wx.Frame function, put name for this window and some developed.This is the code:

```
class MyFrame(wx.Frame):

    def __init__(self, parent = None, title = "MEGBI Vaccine Pilot Plant (MEGBI-VPP) Overview Upstream & Downstream Process"):


        wx.Frame.__init__(self, parent, -1, title)

        panel = MyBackgroundPanel(self)

        LABELSTYLE = wx.BORDER_SUNKEN | wx.ST_NO_AUTORESIZE | wx.ALIGN_CENTER_HORIZONTAL


        menuFile = wx.Menu()
        menuFile.Append(1, "&About...")
        menuFile.AppendSeparator()
        menuFile.Append(2, "Exit")
        menuBar = wx.MenuBar()
        menuBar.Append(menuFile, "File")
        self.SetMenuBar(menuBar)
        self.CreateStatusBar()
        self.SetStatusText("Welcome to MEGPI Project!")
        self.Bind(wx.EVT_MENU, self.OnAbout, id=1)
        self.Bind(wx.EVT_MENU, self.OnQuit, id=2)


    def OnAbout(self, event):
        wx.MessageBox("This is a Screen controller of MEGPI Project","Welcome to my python", wx.OK | wx.ICON_INFORMATION, self)
    def OnQuit(self, event):
        self.Close()
```

Figure 10 : *Python 2*

After the frame class, must put the finale class for the main loop, this is for show the background and the frame without it nothing done.

This is the code:

```
#**********************                main definition and loop              ********************
#*****************************************************************************************************

def main():
    """Testing"""
    MyBackgroundPanel = wx.PySimpleApp(wx.Panel)
    f = MyFrame()
    d=ValveListBox()
    d.Show()
    f.Center()
    f.Show()
    MyBackgroundPanel.MainLoop()

if __name__ == "__main__":

    main()
```

Figure 11 : *Python 3*

Next increased buttons for valves and pumps cause needed to click on it to open and close the valve, turn on or to turn off the pump, for that we used the code in python to create the buttons but we must specify the position of the valve or pump to put the button near of it, to know that button to that valve or pump.

The code in python given much of details like we can name the button and position for it in screen so that easy to create in my screen . I have 67 valves and 10 pumps.

This is the code:

```
            # Create Valves

               #V1

wx.StaticText(panel,-1," V1 ",(18,300))


self.button_V_1 = wx.Button(panel, -1, "V1", pos=(95,100),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V1, self.button_V_1)


def V1(self, event):
        wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
```

Figure 12 : *Python 4*

Getting this screen in first try, if we click on V1 will be given “V1 is Open”:

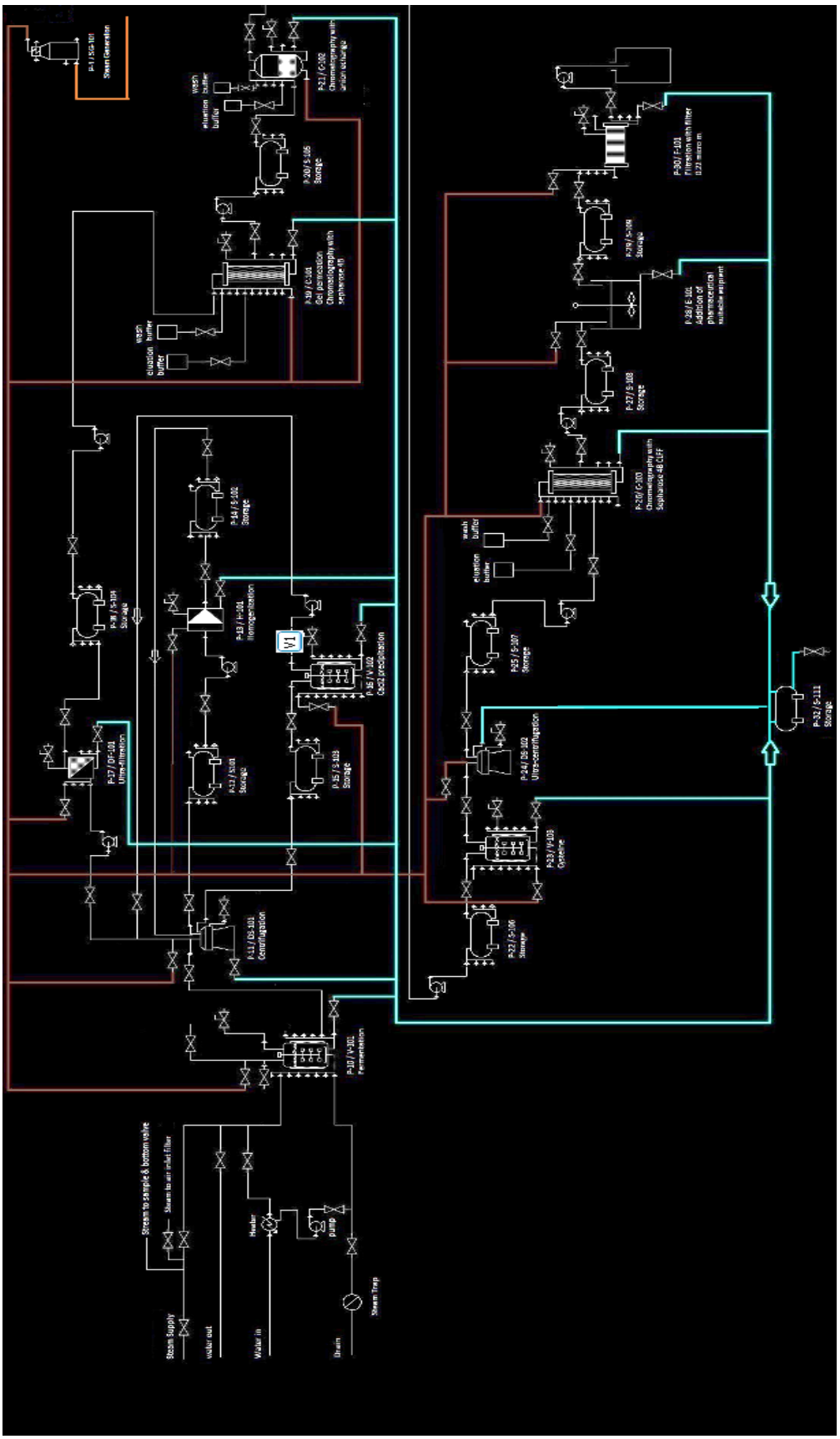


Figure 13 : *Button V1 Created*

Next step is creating label and collected it with buttons to show if we click the valve button will give “ON” or “OFF”

We need a code through which we can enter a condition and we can access to open-close the valve plus give in the label “ON”-“OFF”, and turn off-on the pump plus give “ON”-“OFF”.

This is the code:

```
          # Create Valves

             #V1
wx.StaticText(panel,-1," V1 ",(18,300))

self.button_V_1 = wx.Button(panel, -1, "V1", pos=(95,100),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V1, self.button_V_1)


def V1(self, event):
    valve_status_ON = 'ON'
    valve_status_OFF = 'OFF'

    if  self.Valve1.GetLabel()==valve_status_ON:
        self.Valve1.SetLabel(valve_status_OFF)
        wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
    else:
        self.Valve1.SetLabel(valve_status_ON)
        wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)
```

Figure 14 : *Python 5*

With this code we have continued the Screen Controller to get it, this all buttons and labels represent the actuators, for the sensors we have developed some labels because the data of the sensors are taked from Velleman board.

We passed by some problems with function, how we will join the button with label and how to connect the label with condition ‘if …else’ to given “ON” or “OFF”, and the order of steps. With the help from www.Python-forum.org, there we asked about the problem, and how to fix it.

They have given us help and repaired the code just in exception case.

This was the wrong code:

CODE: SELECT ALL

```
            #V1
      self.button_V_1 = wx.Button(panel, -1, "V1", pos=(95,100),size=(20,20))
      self.Bind(wx.EVT_BUTTON, self.V1, self.button_V_1)

def V1(self, event):

           valve_status_ON = 'ON'
           valve_status_OFF = 'OFF'
           if self.V1.Bind("<Button-1>", callback):
           #self.V1 == True:
           #valve_status = valve_status_ON
            self.Valve1.SetLabel(valve_status_ON)
            wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
```

Figure 15: *Shot of the wrong code in forum.*

And this was the right code after the help from the forum python site:

CODE: SELECT ALL

```
def V1(self, event):
           valve_status_ON = 'ON'
           valve_status_OFF = 'OFF'

           if self.Valve1.GetLabel()==valve_status_ON:
               self.Valve1.SetLabel(valve_status_OFF)
               wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
           else: :
               self.Valve1.SetLabel(valve_status_ON)
               wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)
```

Figure 16 : *Shot of the right code in forum.*

After writing the code for all the buttons, labels, text, we go the Controller Screen:

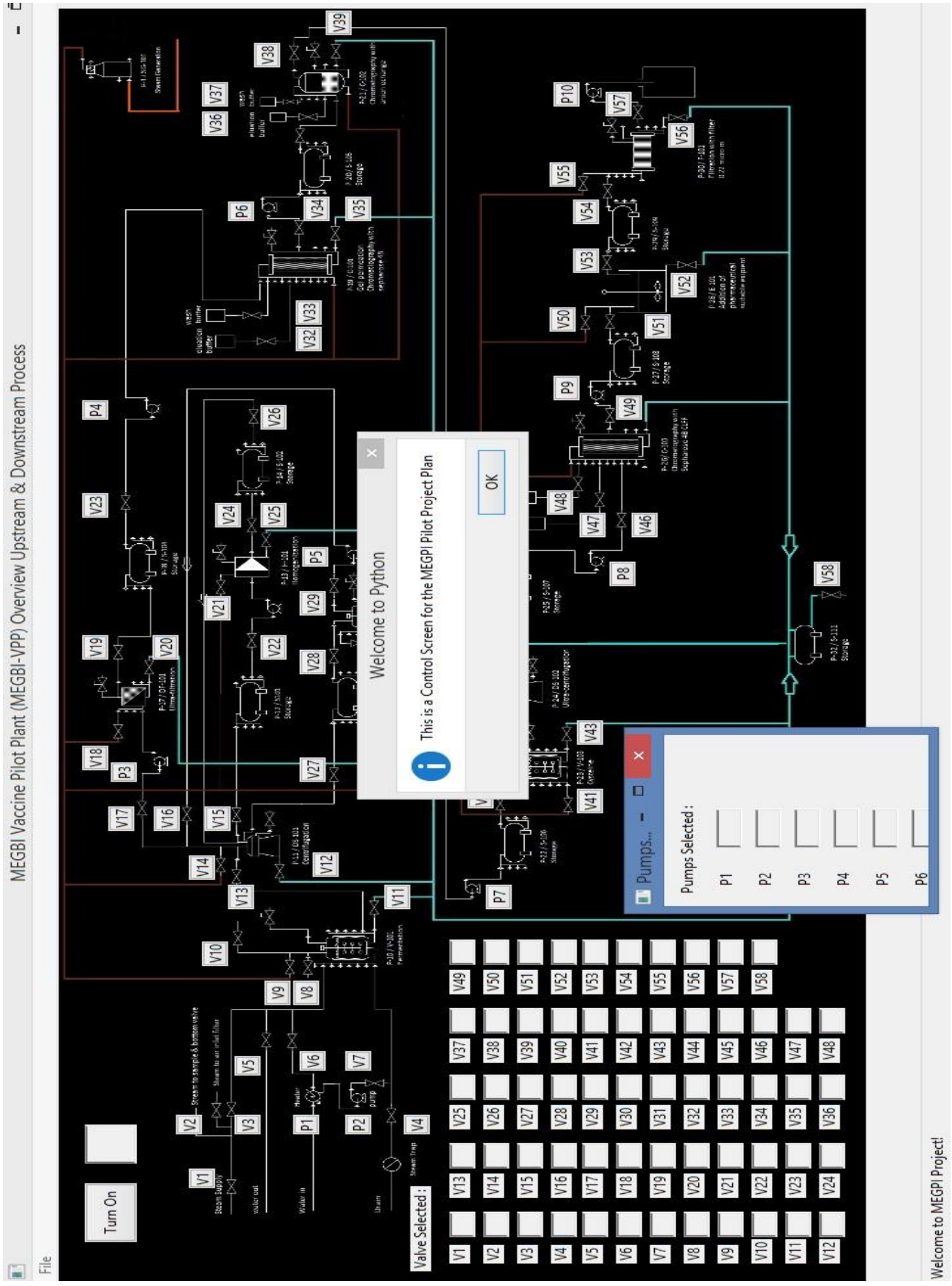


Figure 17 : *The control screen full of buttons ,labels ,list box.*

So as we talk before, the next step is connecting the control screen to USB Velleman K8061 board, we have used a specific code on Python to can the screen read the board and run it, before that we have defined the device of the USB board to pc by 2 dll files named "K8061.dll" " mpusbapi.dll" we downloading it from a site [7], when the board was connected to the screen we get a command shows the message between it (figure 2.3).

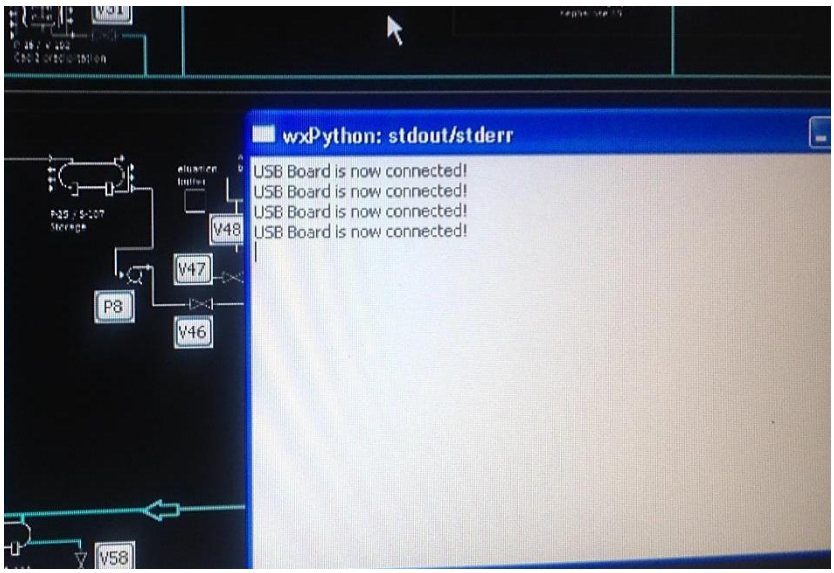


Figure 2.18 : *Command between the USB board and the control screen .*

This is the code:

```
#**********************                Run USB System                      ***********************
#*****************************************************************************************************
    def OpenUSBBoardThread(self):
        self.dll = windll.K8061
        i = self.counterUSBBoards
        for doit in range(0,i+1):
            try:
                self.dll.OpenDevice()
                self.USBOpened = True
# debug info
                print ('USB Board is now connected!')
#end debug info
            except:
                txt = ('Please Check USB Board connection')
                print ('txt')
                return
```

Figure 19 : *Python 6*

Next of this code, must know how to connect the screen to the Velleman board, for that we must read about Velleman board, how to connect it ,how many inputs outputs ,the channels it has, and the actuators sensors which represents the outputs inputs .

And to connect the temperature sensors and get value, we must use a function to make the Velleman board read the value of the sensors and represents it in the label, to read the value should we use self.dll.ReadAnalogChannel(Adress,Channel) function ,the address of the board used is 0 and channel was 1. , next is the code:

```
        # temperature sensors value

wx.StaticText(panel,-1," Temperature Value ",(100,12))
self.temp_Vaporizer_out = wx.StaticText(panel, size = (26, -1), pos = (200, 10), style = LABELSTYLE )



new_value = str(self.dll.ReadAnalogChannel(0,1))
self.temp_Vaporizer_out.SetLabel(new_value)
self.temp_Vaporizer_out.Refresh()
```

Figure 20 : *Python 7*

After knowing how to run, we must collect the circuit of the temperature sensor:

6.3 Adaptation of Device Sensors and Actuators

6.3.1 *Actuators (Valves)*

Next we will talk about the actuators and sensors, and how we have connected it to the USB board to control them by the control screen. But when we makes the connection between the temperature sensors USB board we get wrong results that make a question about temperature so we have canceled it from the project and we continue about automation system by control the fluid flow from the actuators. To connect the actuators to USB board we need a relay, this relay will control the valve by open/close when they takes the order from the board same to pump by turn on/off, so we have built this relay by using a schema (figure 2.4).

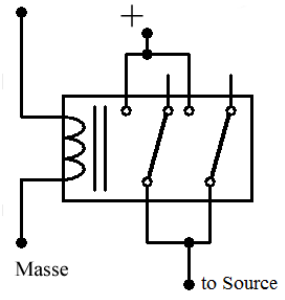


Figure 2.21:*Schema represents the construction of relay.*

So with this schema (figure 2.4), we build it by: relay-small board-tapes-welding

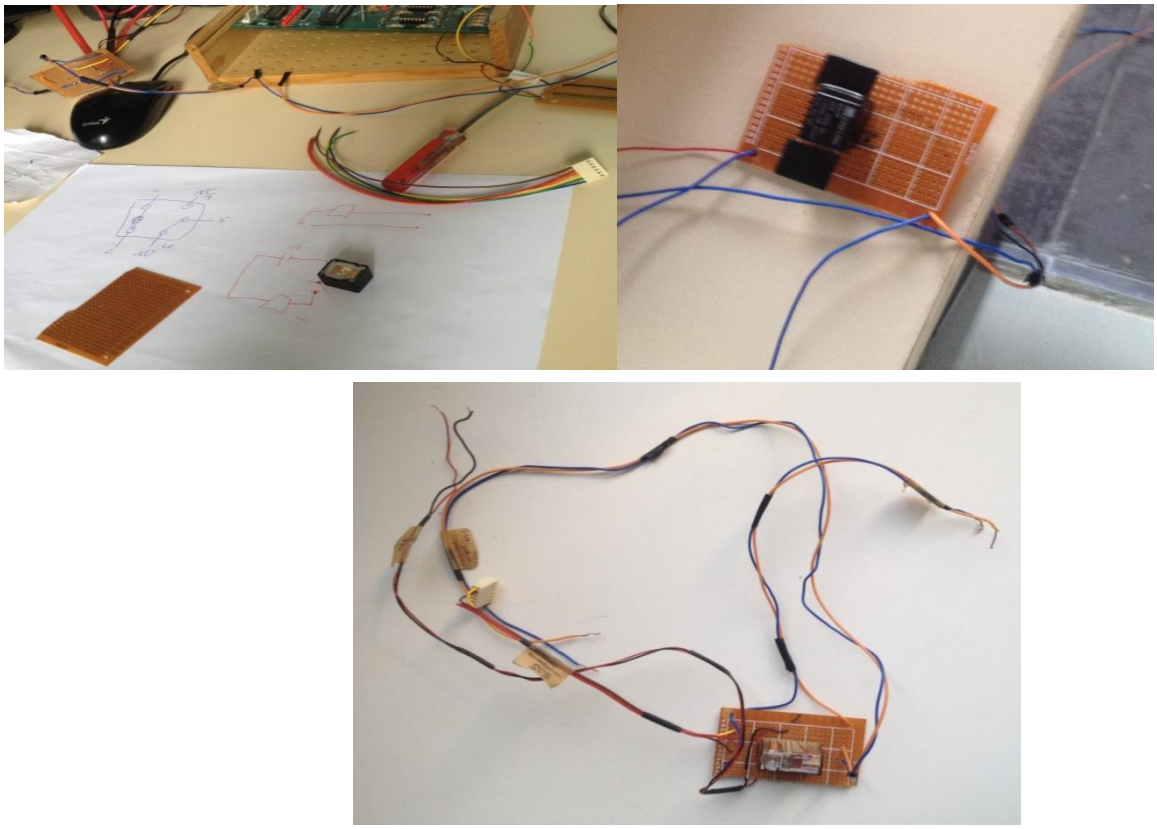
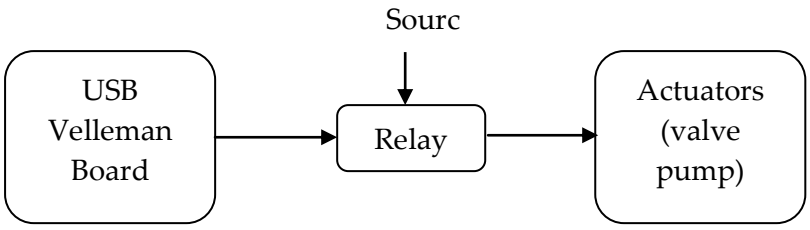


Figure 2.22: *installation a relay.*

And we use it to connect the actuators so as we can see on figure it's like a switch but with order this order come from the board, plus this actuators have connected to outputs (digital or analog).



4 steps to link the relay and make the connection between the board and the actuators:

- Connecting the relay to source (220V)
- Connecting the relay to actuators (220V)
- Connecting the relay to outputs channel(5V-12V)
- Connecting the relay to power of USB board(12V)

After knowing how to connect the actuators, we take a valve and we will try to connect it, so as we talk before, we have connected the valve to relay, then we have linked the relay and we get the circuit between the board, valve, relay, source (figure 2.6).

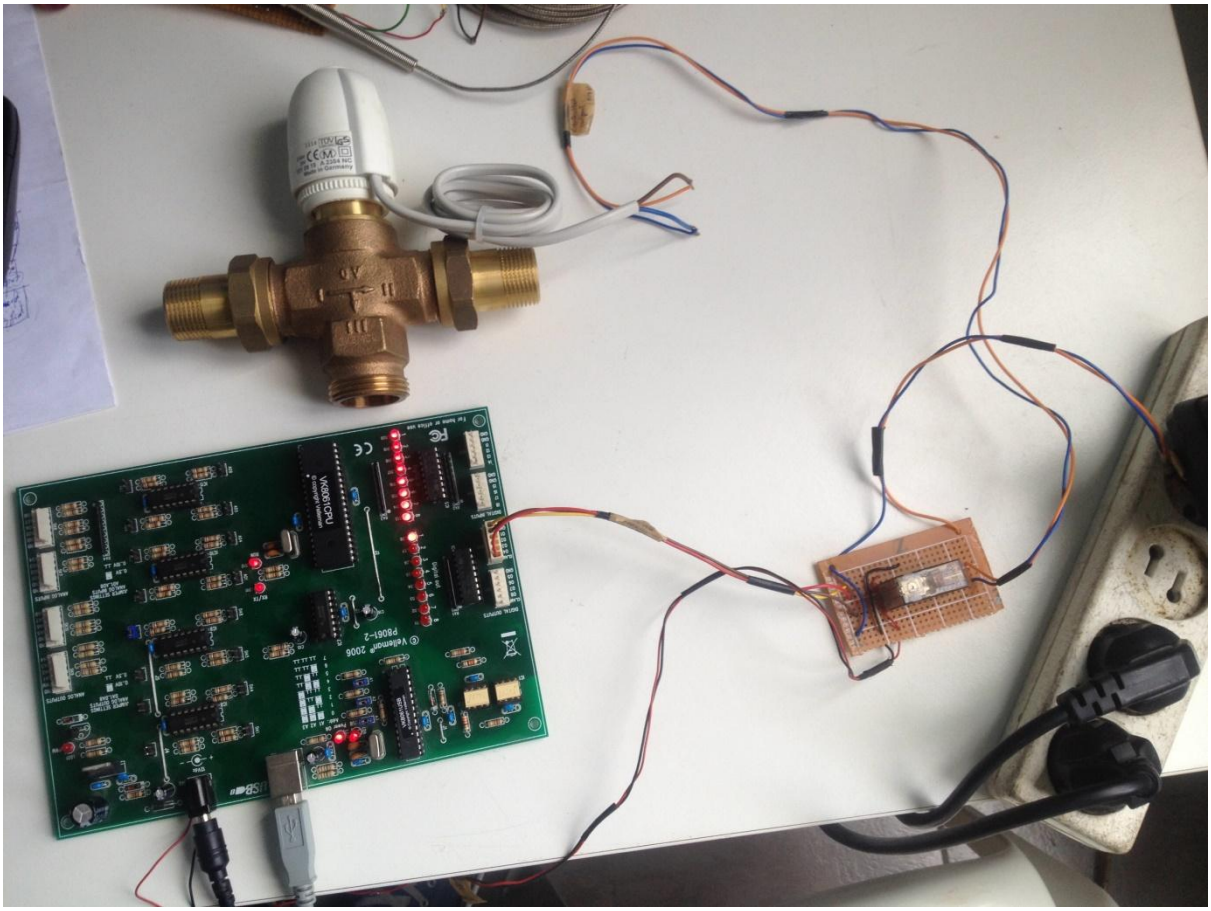


Figure 2.23 : *The circuit between the board, valve, relay, source.*

And we will test it to make sure the valve the circuit is working good and ready to use it in the test stand, so we will try to open and close it from the screen by clicking on button and we take many pictures showings the test, as we see the blue color mean the valve is getting open so the order from the board has been arrived to valve to open and the red light on the board in the outputs channel confirm the same thing (figure 2.7) ,and vice versa the red light disappeared when we close the valve so that confirm the same thing that the board gives the order to open or close(figure 2.8), so the control of the valve is working .

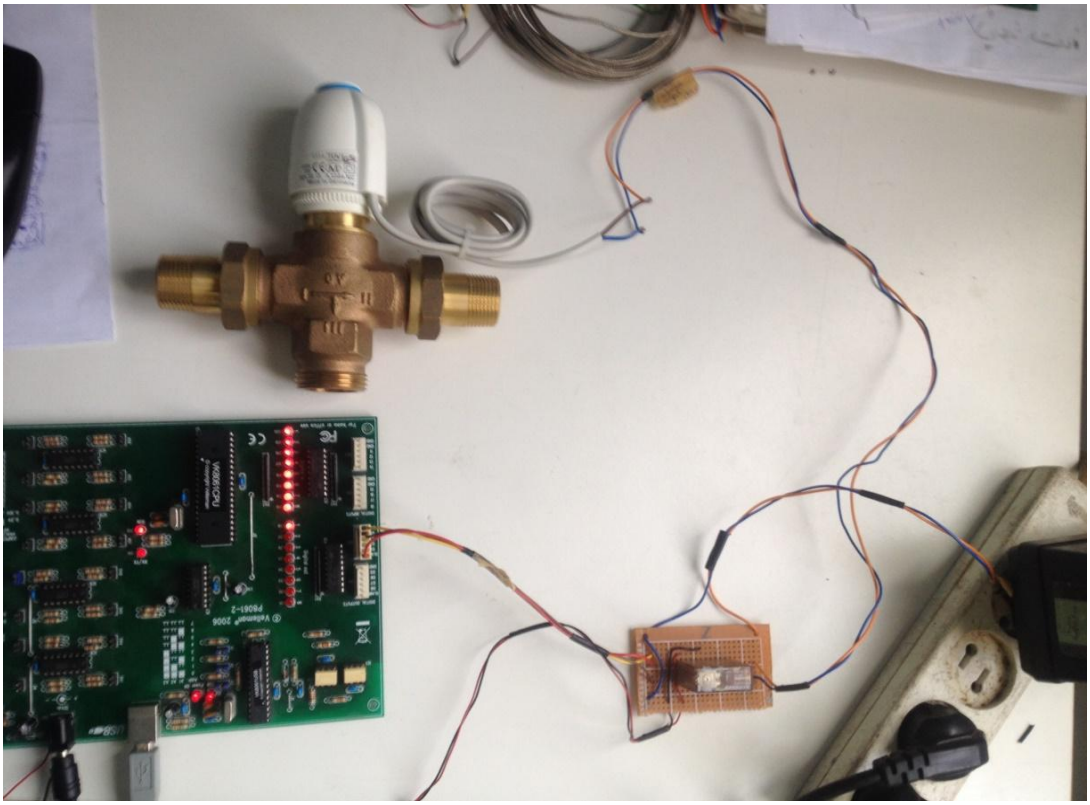


Figure 2.24 : *The circuit when the valve gets opened.*

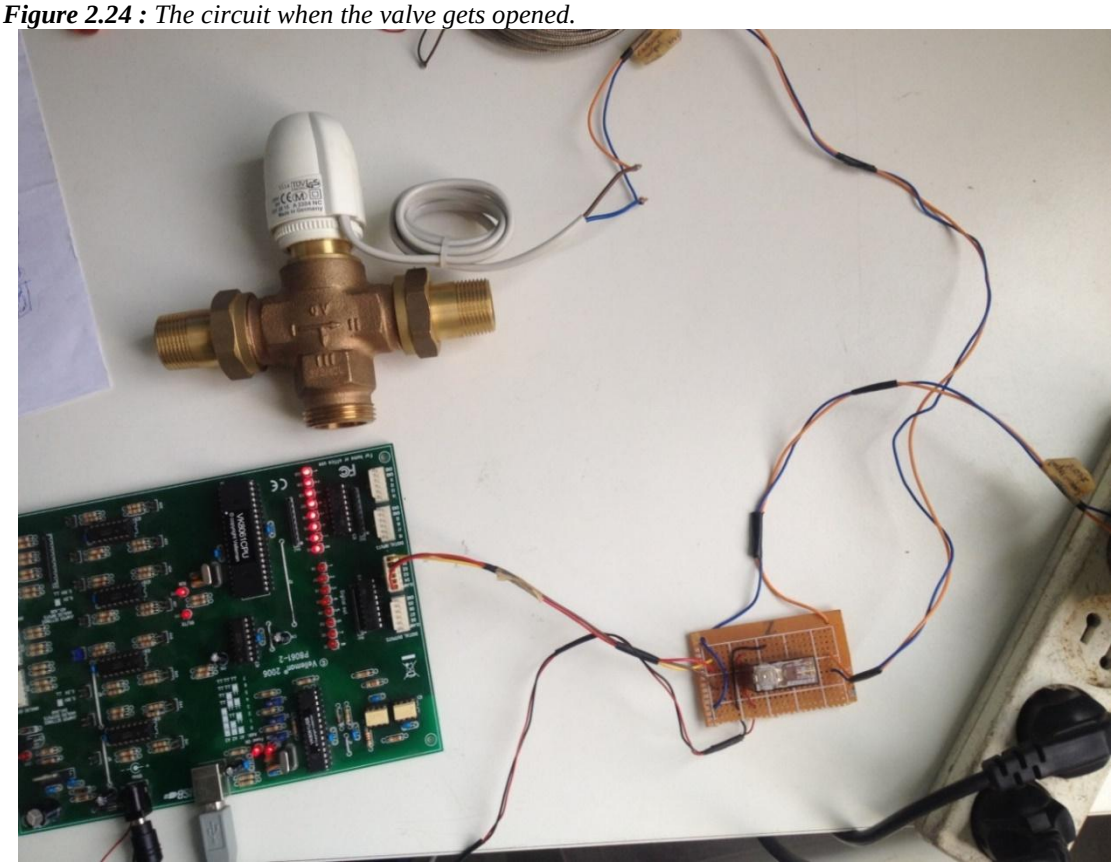


Figure 2.25 : *The circuit when the valve get closed.*

So we have 58 valves and 10 pumps concerning the production,and the valves , the pumps have used on temperature control system for bioreactor and the cleaning way,and we will make a connection on the PI diagram between the all actuators and the boards ,PLC.so we have connected each valve to a output channel on board or PLC(figure 2.9),16 outputs (digital + analoge) on board ,16 on PLC.

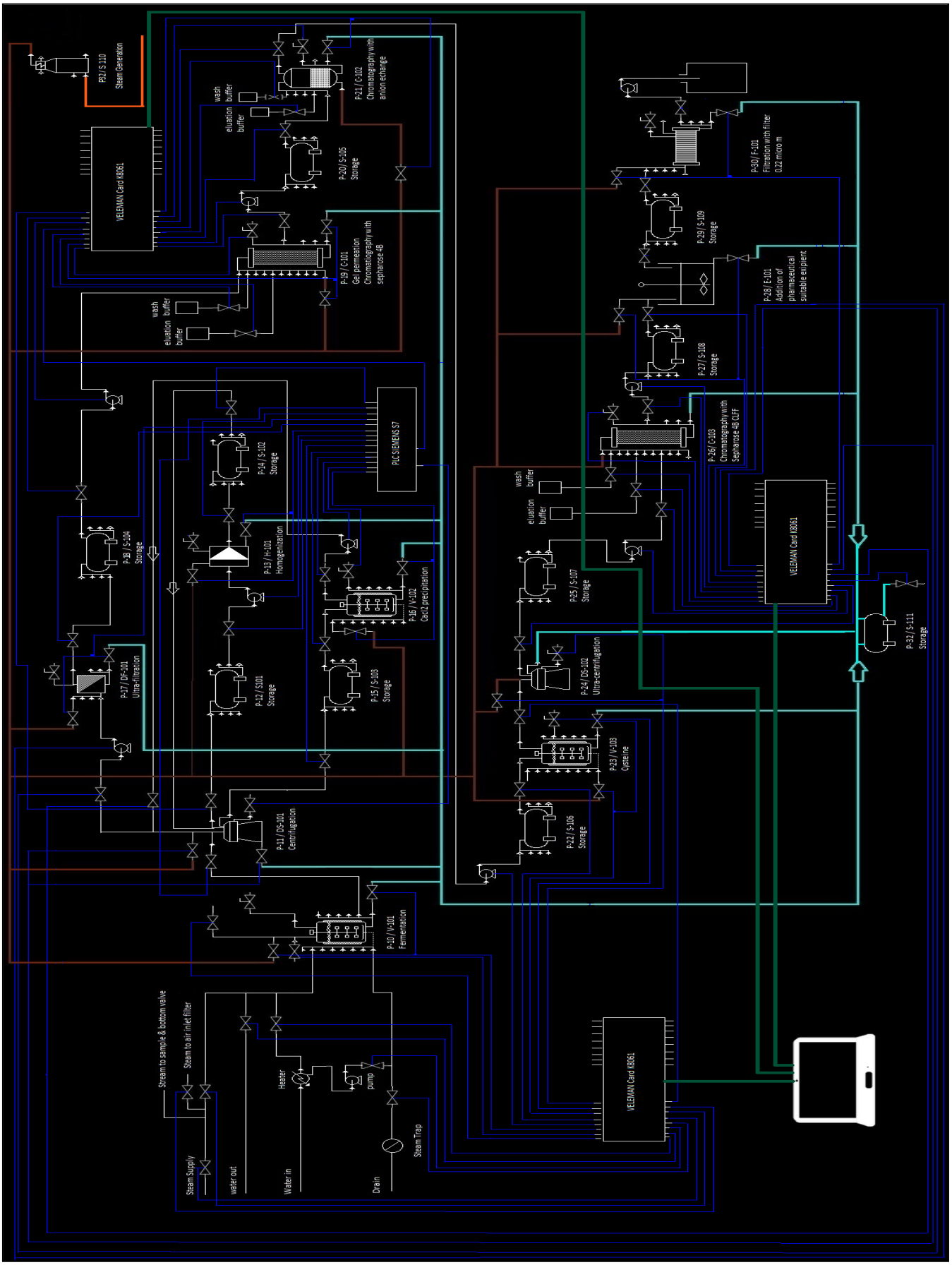


Figure 2.26 : Network between the actuators and the boards,PLC.

6.3.2 *Sensors*

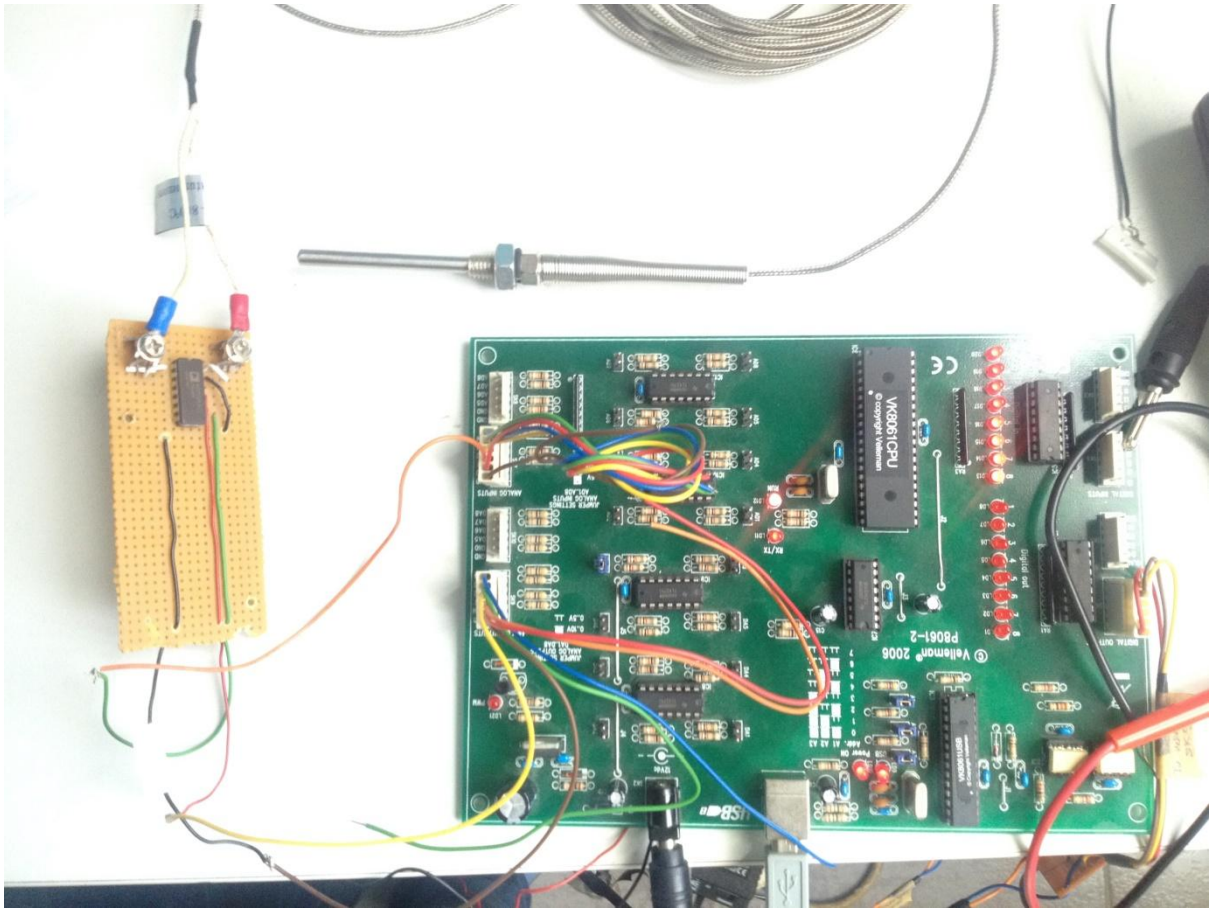


Figure 27 : *The circuit of Temperature sensor with Velleman board.*

This is the circuit, in which we have connected the temperature sensor to Velleman board across a connection, we have put it on inputs channel one but we have to connect it to output at same time, for even take electricity from that output it just the only way to filled.

So we need 2 connections, one on inputs channel 1 position 1, and other in outputs channel 2 position 1 plus ground and it's ready to work.

After the scan and test shows that there is the temperature sensor malfunction and it should be removed from the project for a reason of time constraints.

We take 3 shots to show how the sensor didn't give a constant value this shows that there malfunction, plus all the connection was right.

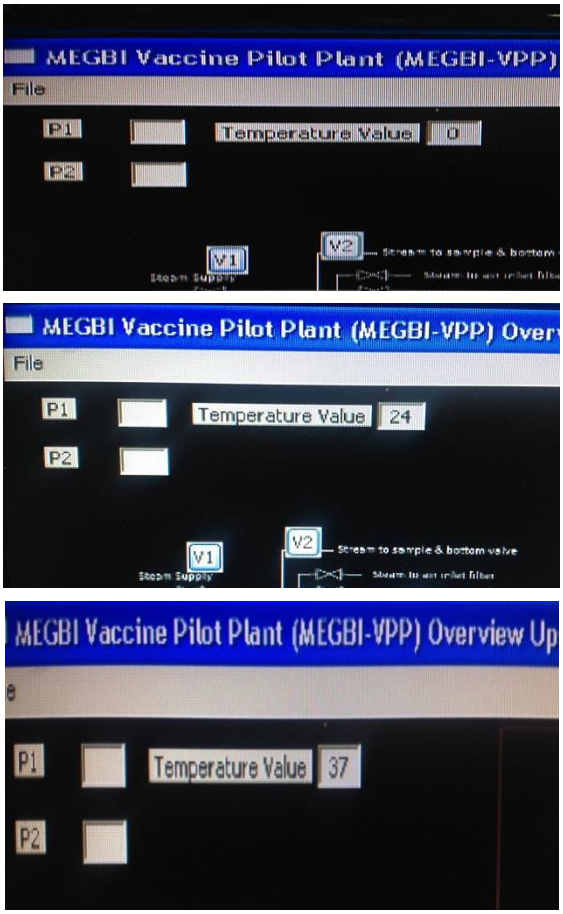


Figure 28 : *3 Shots show the malfunction.*

Now, how to connect the outputs represents across valves and pumps.

For example we worked on valve, must click on V1, that button give order to Velleman board to open or close the valve. So we have used a code to connect the python code valve to the board.

This is the code:

```
    def V1(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'

# open the USB board

        if  self.Valve1.GetLabel()==valve_status_OFF:
            self.Valve1.SetLabel(valve_status_ON)


            self.OpenUSBBoardThread()
            self.dll.SetDigitalChannel(0,1)
            wx.MessageBox("V1 is open ", "Open", wx.OK|wx.ICON_INFORMATION)
            self.timer = wx.Timer()
            self.timer.Bind(wx.EVT_TIMER, self.on_timer)
            self.timer.Start(0)
            wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve1.SetLabel(valve_status_OFF)
            time.sleep(2)
            self.dll.ClearDigitalChannel(0,1)
            print 'Digital Channel Cleared, V1 turn off'
            wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)
```

Figure 29 : Python 8

We have used timer to get order fast if we click it opens directly and same to close it.

Now if I click on valve V1 they give V1 is open and the board take the order to open it, same to close it.

How to connect the valve to the Velleman board:

There should be a relay, like we see:

Must make a relay, we start with collect the contents

Needed:

- relay
- Small board
- Tapes
- Welding

the relay play the role of the brood , but with order.

For example , if a request from the board to open the actuators ,this brood will hatched and give the power required to actuators.

So when we click on button at screen,the board get order to open or close the actuators,and the relay is helping to do this order,across a signal from the board represent by 0.5V.

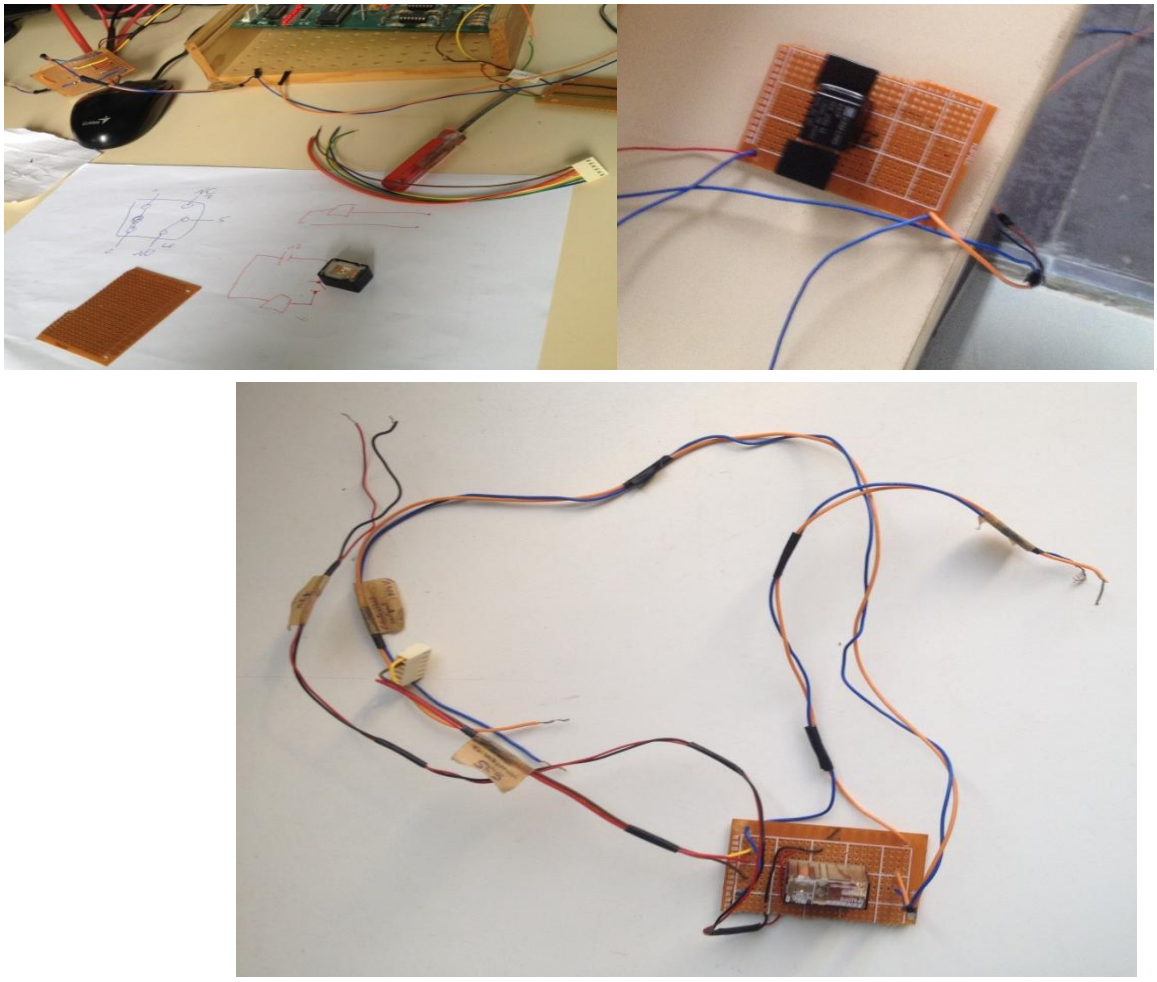


Figure 30: *Building a relay.*

This work like an electric transformer:

4 tapes:

- First one is getting 220V from source.
- the other is 220V linked to Valve(because the valve working on 220V)
- the other is connected to the source of Velleman board that mean take 12V
- The last one is connected to channel Analog outputs.

And we take a picture of the circuit:

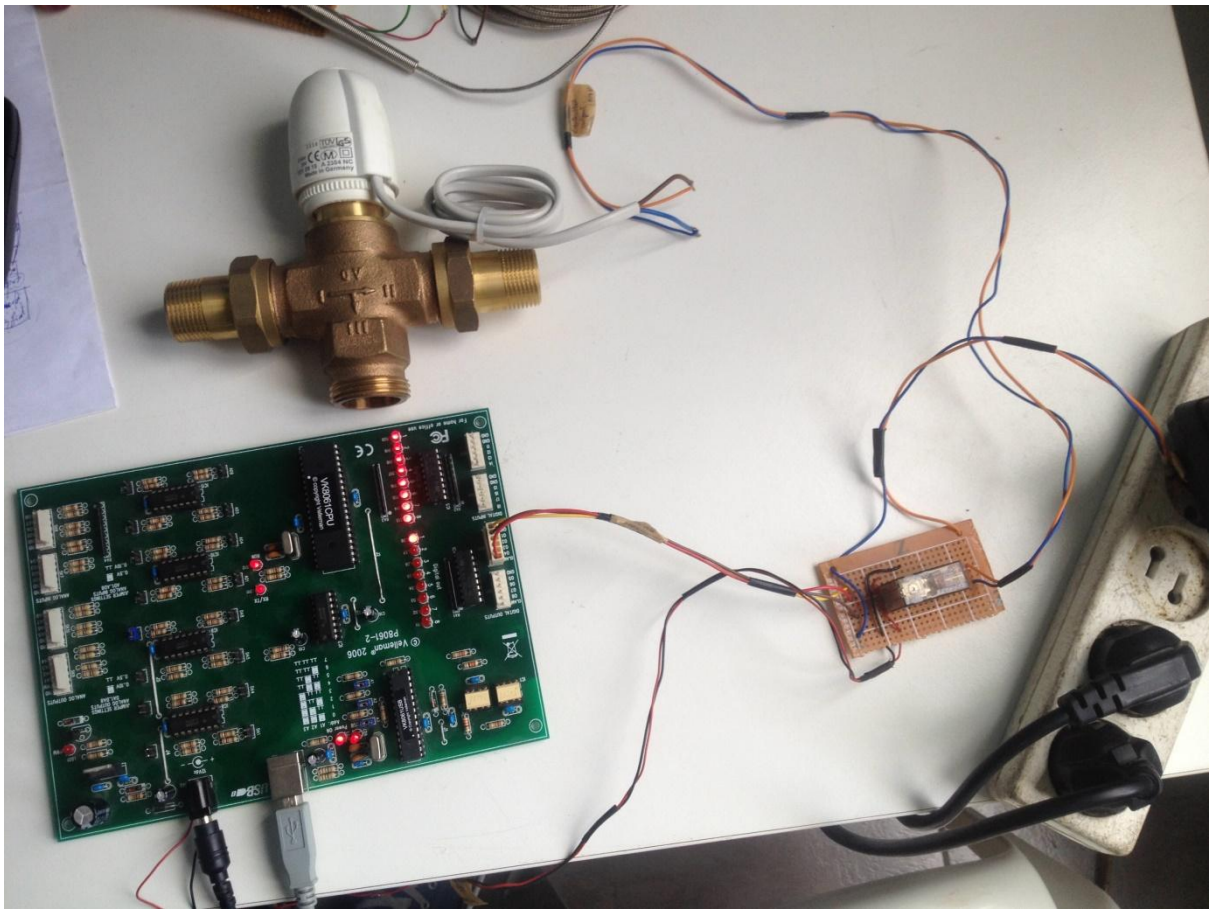


Figure 31 : *The circuit of Valve with Velleman board*

Now must test it:

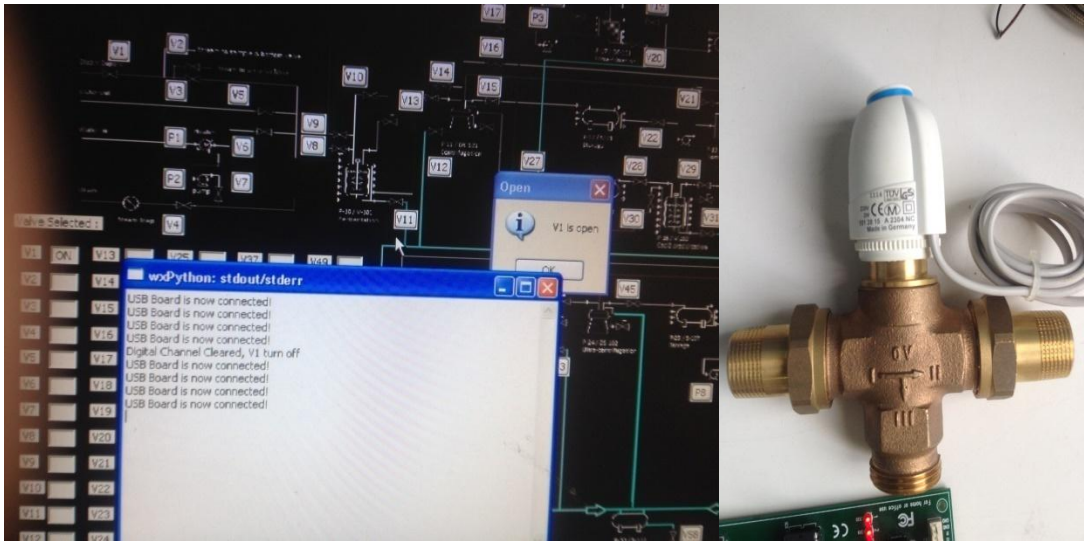


Figure 32 : *Testing the process.*

The valve is open, that blue color on valve means that the valve was open.

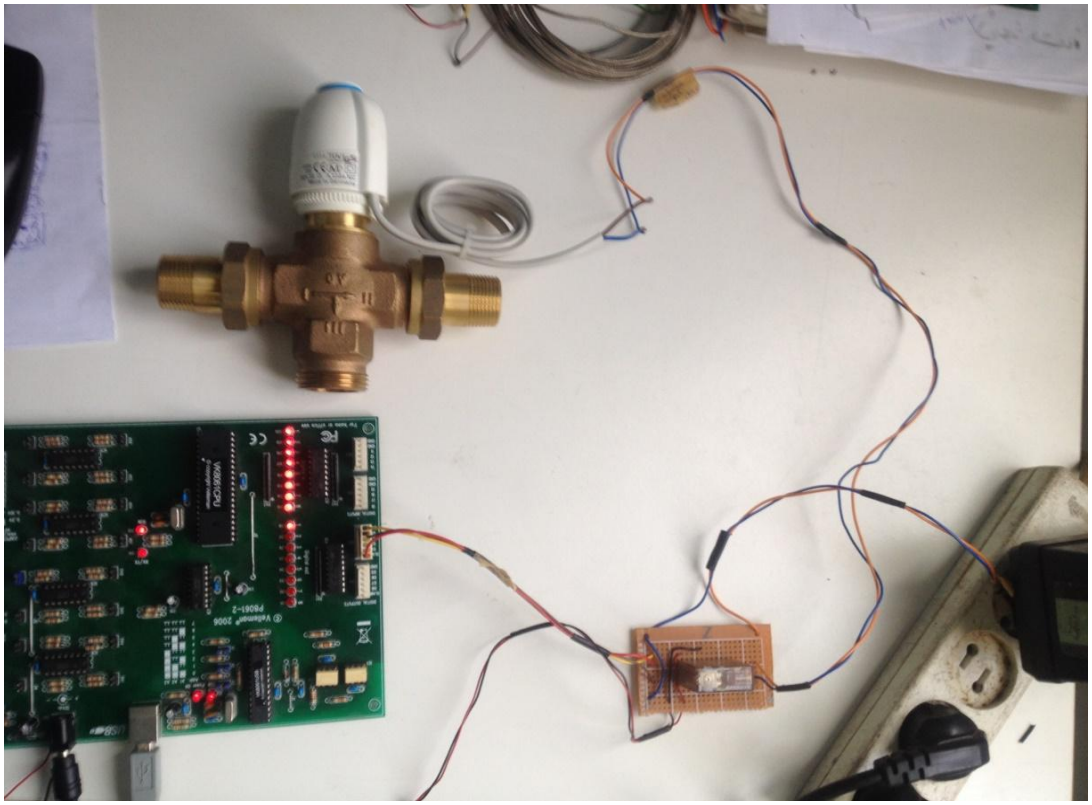


Figure 33 : *The circuit when the valve gets opened.*

They is red light on near the series of light, that light meaning the valve is getting order and open that is working when The button was clicked, so the valve was open.

Next click to close it, and we should showing it too:

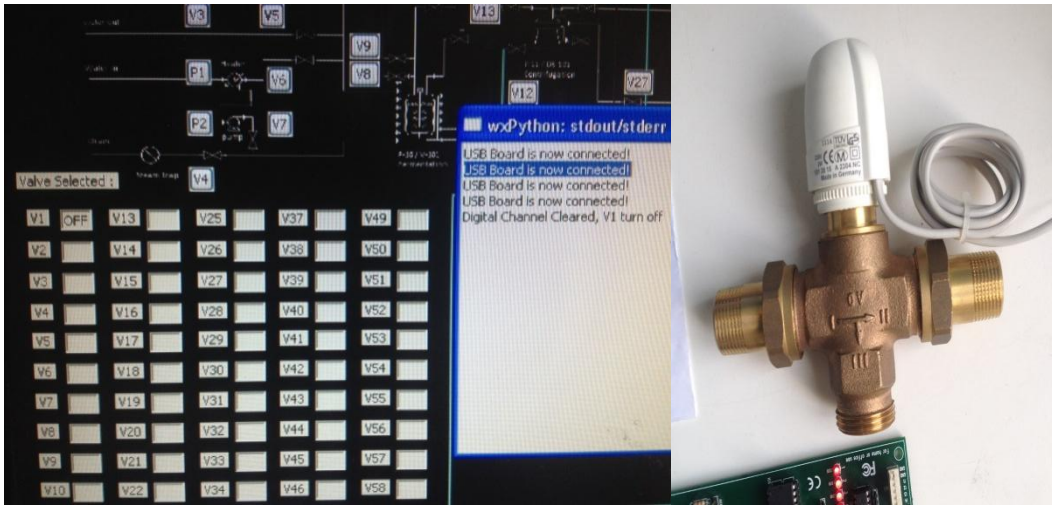


Figure 34 : *Testing Process.*

The valve is closed; we didn't saw the blue color.

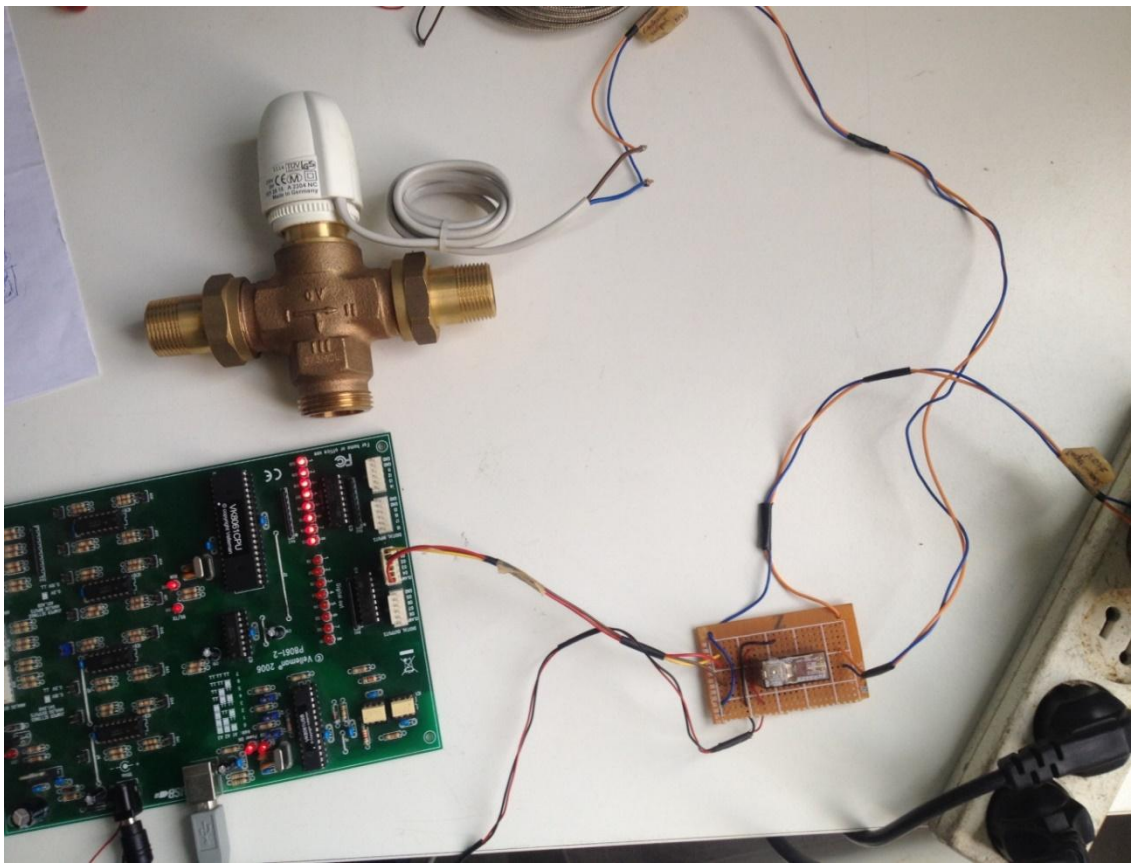


Figure 35 : *The circuit when the valve get closed.*

We saw in this picture the red light is off, that means the board gives order to valve to turn off, and is sawing in screen V1 "OFF".

Next, we have moved to the test stand, we have painted the circuit needed, across a website [2].

We have taken part of The Downstream processing, and we will test it across the board, valve, pump, and the control screen.

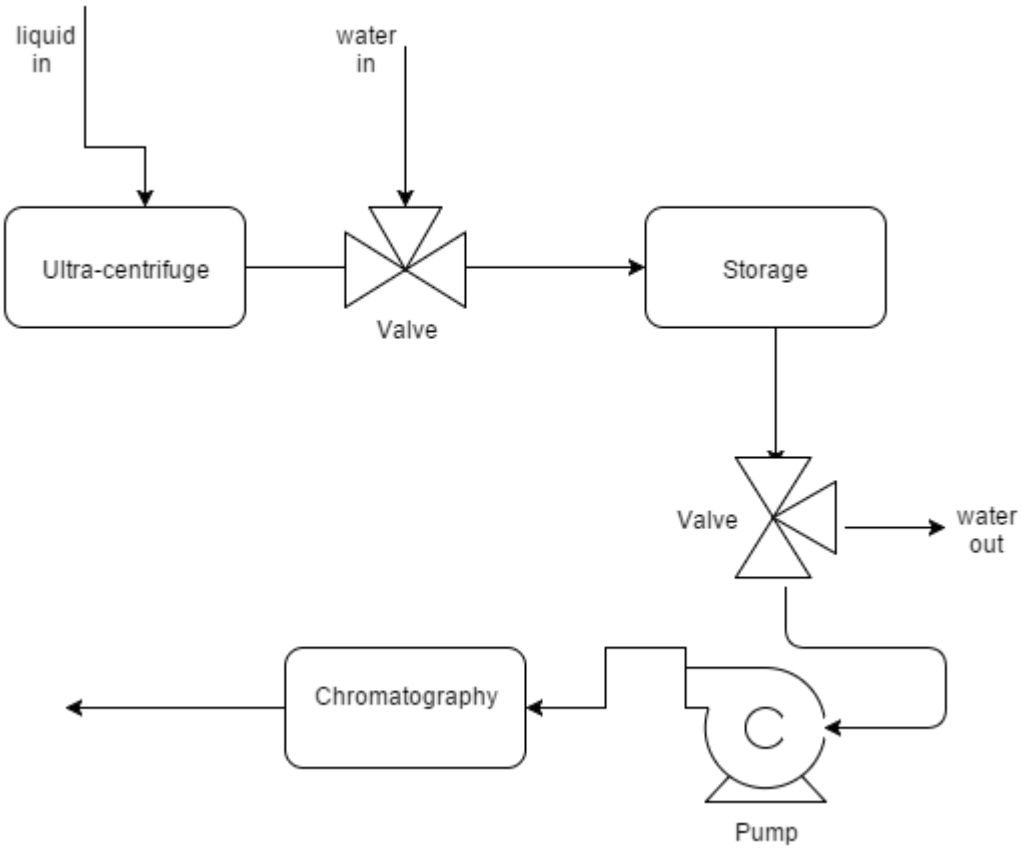


Figure 36 : *Diagram for the part of downstream processing used.*

We saw liquid enter to the ultra-centrifuge and continue across the valve to Storage.

After the storage, the liquid continue to Chromatography across valve and pump.

We saw in the way a line to water-in, that line is used to clean the storage, and more the line of water-out is used to get rid of the dirty water.

Across the diagram we will create a schema to the process that will test it.

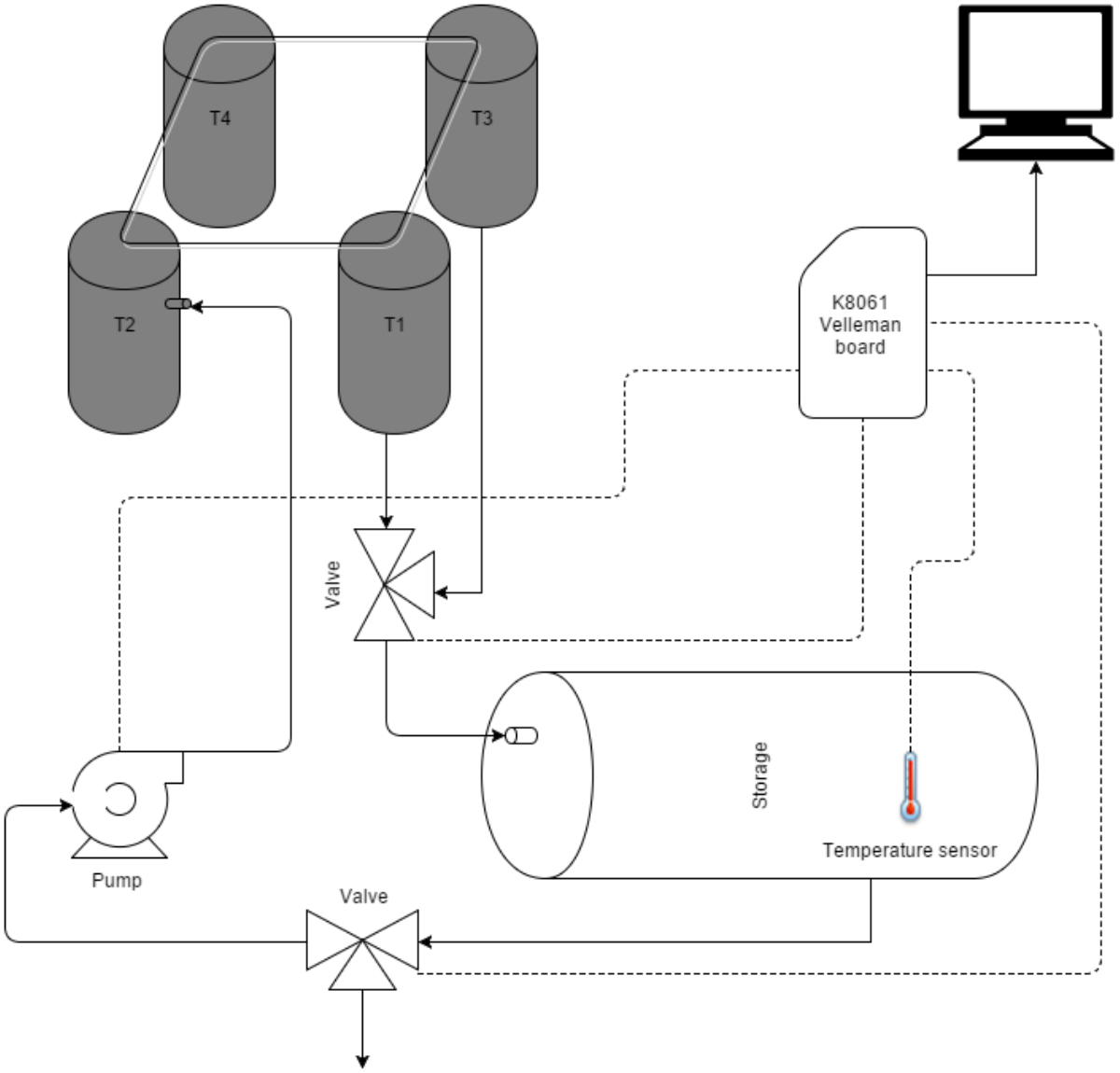


Figure 37 : *Schema of the part of downstream processing used.*

The contents of this schema:

- T1 represent the ultra-centrifuge.
- T2 represent the Chromatography.
- T3 represent the Storage of water used for cleaning.
- USB Velleman K8061 board connected to laptop.

This schema is a miniature of the really circuit.

Figure 38 : *Test stand.*

Like we saw in schema we have built the test stand:

The board is connected to pc, and the control screen was working and ready to test,

With using of plastic pipes, we have linked T1 to valve and the valve to the storage, plus T3 to the valve too.

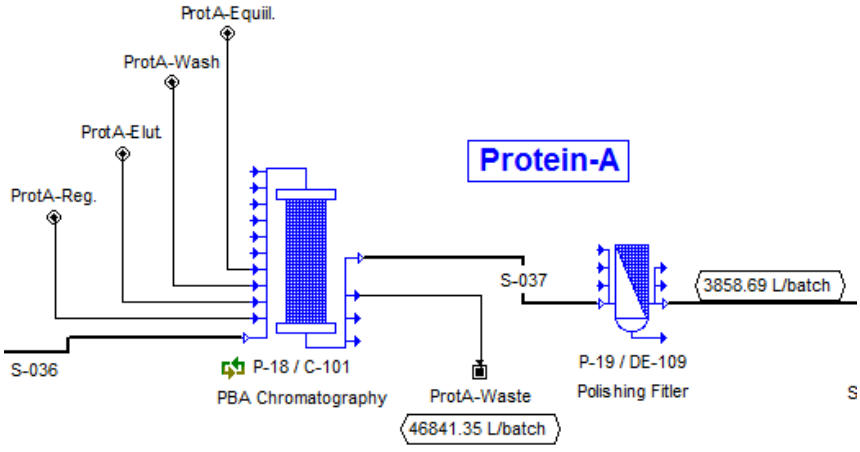
On the other we have linked the storage to valve, the valve to pump and pump to T2.

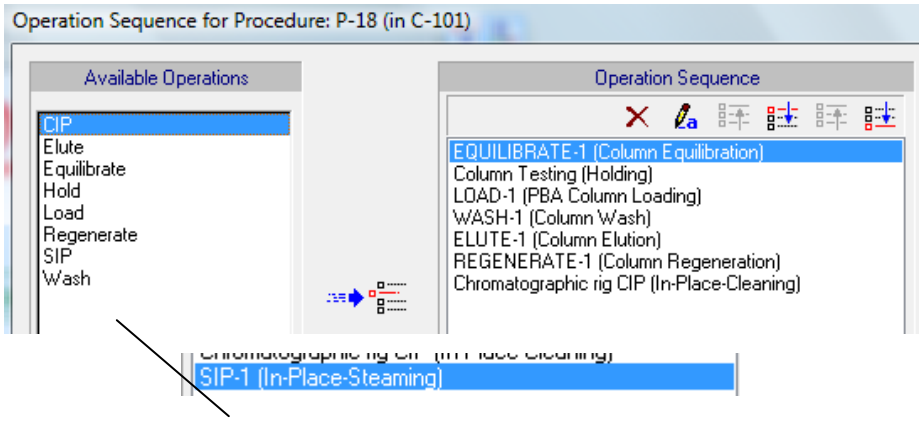
We have worked hard to build it, like we saw, we used plastic pipes, the mechanics tools, cash cables …

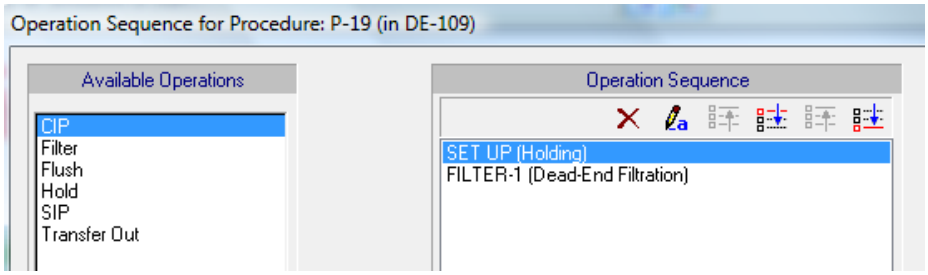
Finally getting the test stand:

Figure 39 : *The test stand ready for work.*

6.4 Functional Operation Algorithm of Automation System







Now we have created a algorithm represents the production of the vaccine process by process plus the valve necessary and the order of open/close turn on/off for the process and we have based on this algorithm to made the full automation of the system of production the vaccine by control the fluid flow, so we have based on this algorithm to collect on Python[8] a code can with it "Turn On' the system to start the production, so with the algorithm we have created the Python code that we have need to the automation process and it's done ,ready to test , just click on "Turn On".

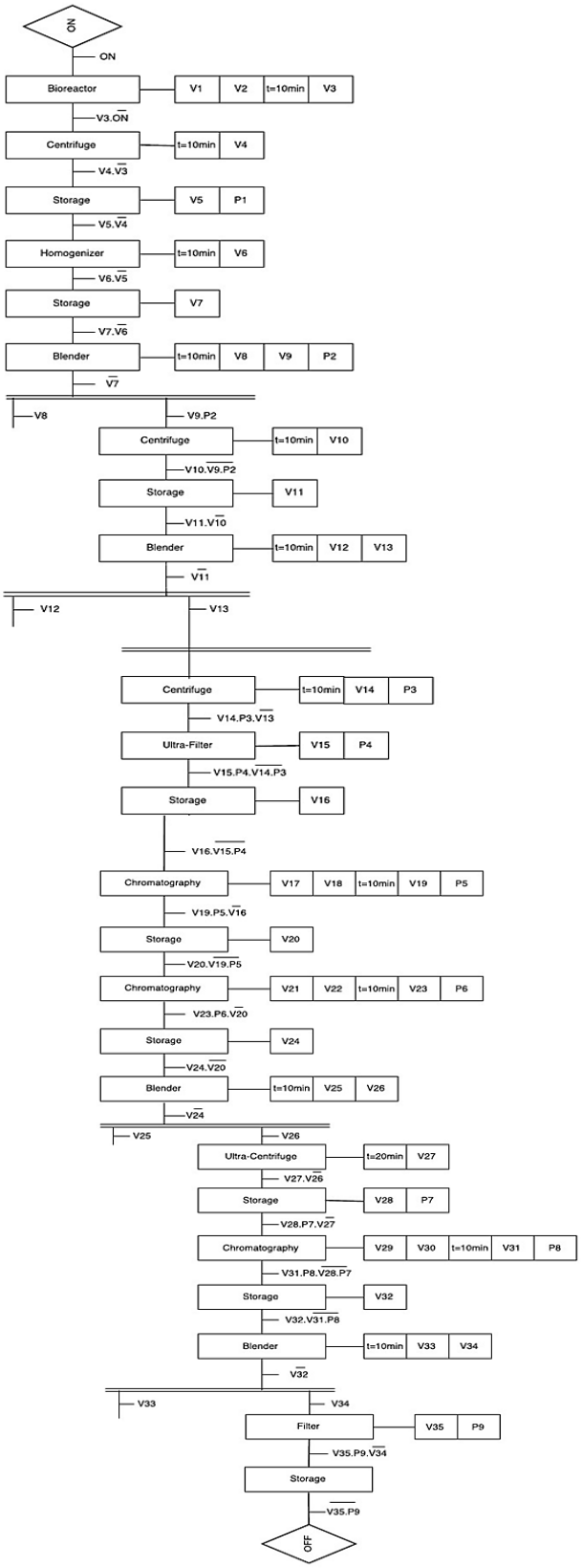
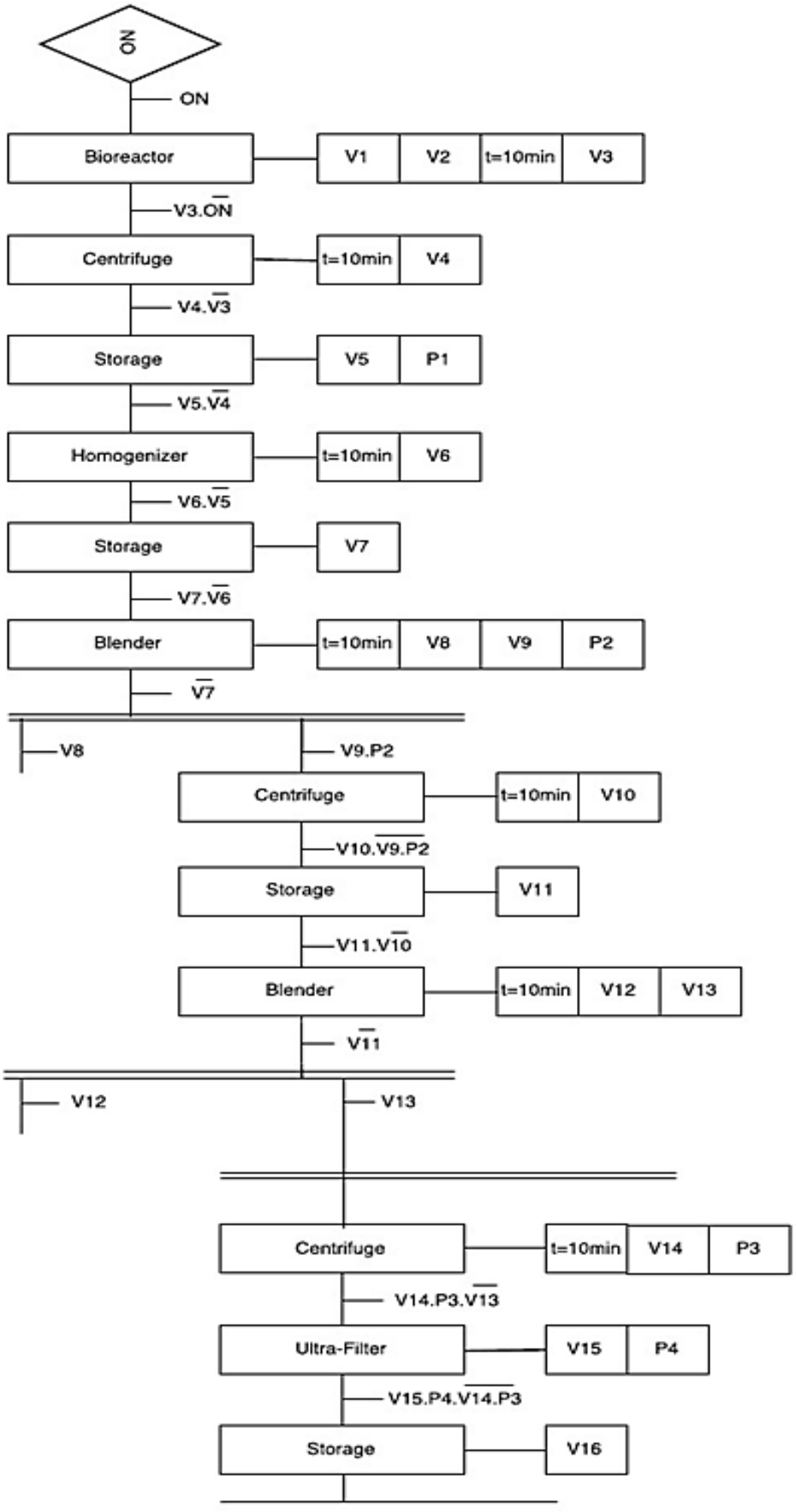


Figure 2.40a : *Algorithm represents the automation of the production (Overview)*


ON
ON
Bioreactor
V1
V2
t=10min
V3
V3.$\overline{ON}$
Centrifuge
t=10min
V4
V4.$\overline{V3}$
Storage
V5
P1
V5.$\overline{V4}$
Homogenizer
t=10min
V6
V6.$\overline{V5}$
Storage
V7
V7.$\overline{V6}$
Blender
t=10min
V8
V9
P2
$\overline{V7}$
V8
V9.P2
Centrifuge
t=10min
V10
V10.$\overline{V9.P2}$
Storage
V11
V11.$\overline{V10}$
Blender
t=10min
V12
V13
$\overline{V11}$
V12
V13
Centrifuge
t=10min
V14
P3
V14.P3.$\overline{V13}$
Ultra-Filter
V15
P4
V15.P4.$\overline{V14.P3}$
Storage
V16

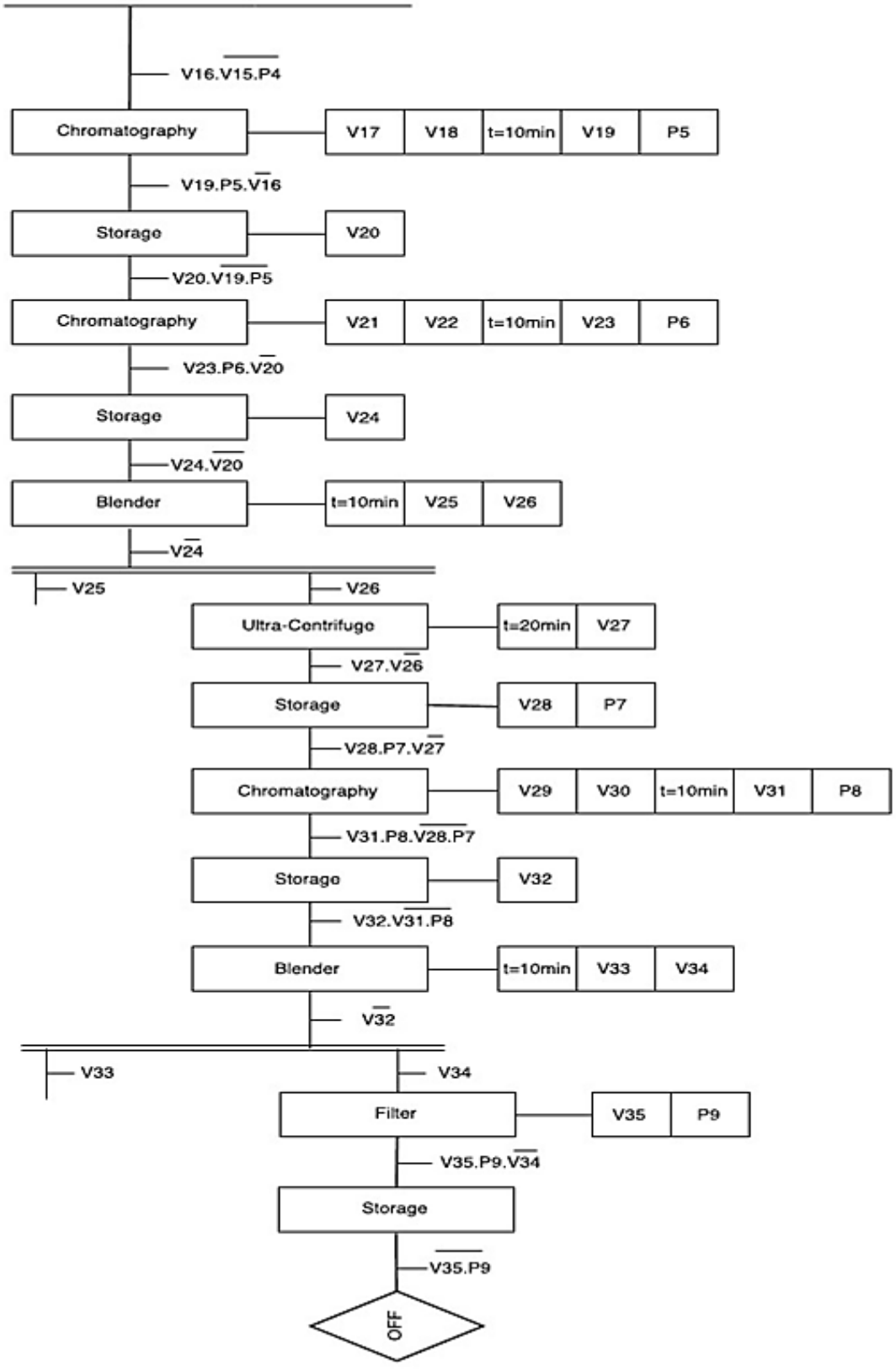


Figure 2.41b : *Algorithm represents the automation of the production (as in overview, but two pictures)*

After creating that algorithm, we have moved to the test stand, the idea of this step is to build a part of the PI diagram of the MEGBI-VPP Processing and testing it ,so we have to choose the circuit that we will based on it build the test stand. By a website [9] we have created the diagram that represents this part. We will test it by the board, valve, pump, and the control screen.

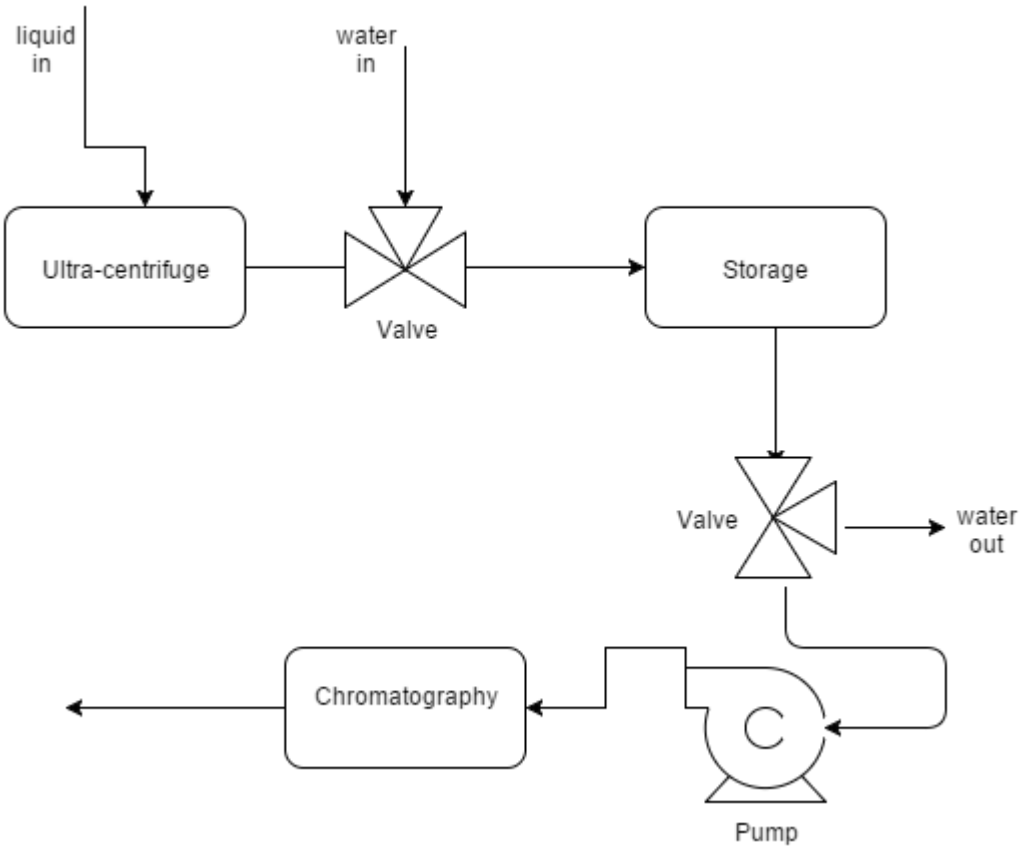


Figure 2.42 : *Diagram for the part of MEGBI-VPP Processing used.*

The liquid enter to ultra-centrifuge and continue by a valve to Storage, then to Chromatography by valve and pump (figure 2.11). This liquid represents the production and this is the part that we have chosen to base on it to build our test stand. We saw in the way a line to water-in that line is to clean the storage, and more the line of water-out is to get rid of the dirty water. With this Diagram we have created a Schema represents the test stand that we will build and each machine, instruments have to use (figure 2.12).

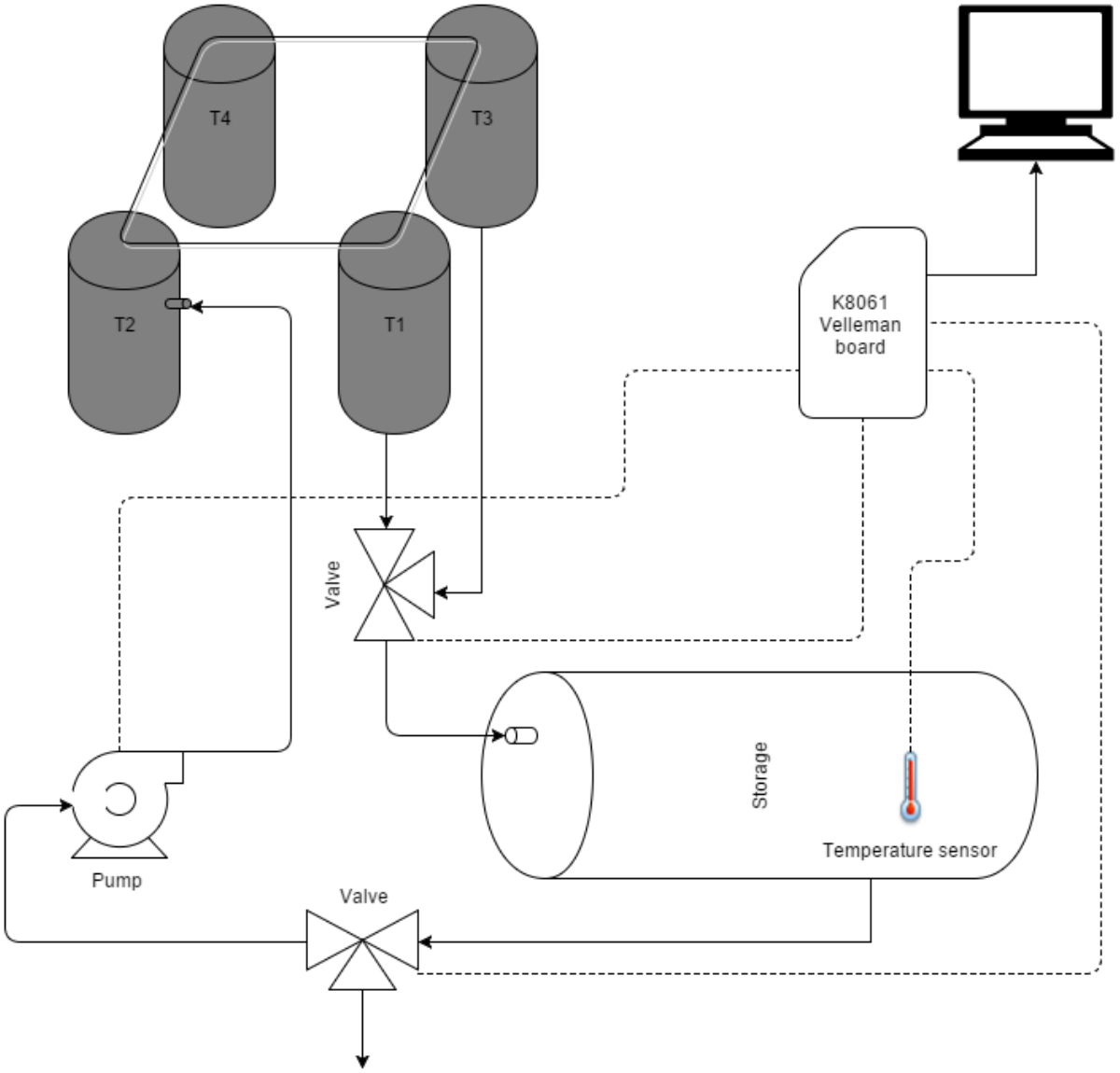


Figure 2.43 : *Schema for the part of the MEGBI-VPP Processing used.*

The contents of this schema:

- T1 represent the ultra-centrifuge.
- T2 represent the Chromatography.
- T3 represent the Storage of water used for cleaning.
- T4 is not used.
- USB Velleman K8061 board connected to laptop.

This schema is a miniature of the really circuit that we will build a really one like it.

Figure 2.44 : *Test stand.*

As we saw in schema we have built the test stand (figure 2.13). The board is connected to pc, and the control screen was working and ready to test, with using of plastic pipes, we have linked T1 to valve and the valve to the storage, plus T3 to valve too, T4 is not used. On the other we have linked the storage to valve, the valve to pump and pump to T2, here we have used a pump cause we want to send the production to T3 and T3 is located at a height of 2 meters We have worked hard to build it, like we saw we use plastic pipes, the mechanics tools, instruments, cash cable …

Figure 6.14 a: *The test stand after build it.*

Figure 6.14b: Installed *Electrical Control Cabinet*

7. Testing the automation system

We have divided our work to tow part of results, the first one will examine all the actuators in the test stand to see if there is any problem or to see if everything is all right, it has been working on three tests by trying to control the actuators on the test stand from the control screen, then the other will examine the full automation system of the production of the vaccine by testing the Python code that based on algorithm to do this automation and control all the process of the production.

7.1 Testing the Test Stand

We start with the test of the test stand we made three tests:

7.1.1 *Test 1*

On this test ,will examine the valve by sending water from the ultra-centrifuge to the storage and will exercise control over them and see if we can introduce water into the storage when we open the valve by pressing the button of valve on control screen and the usb board will give the order to open.In short,will click the button of valve on screen to open it and other click to close it.

Figure 3.1 : *The Test 1 concerning open and close a valve.*

PRECONDITION	TEST ACTIVITY	POST CONDITION	TEST SUCCEED/FAILED
PIPES WITHOUT WATER	OPENING AND CLOSING THE VALVE.	FIRST CLICK VALVE OPEN OTHER CLICK VALVE CLOSE	SUCCEED

7.1.2 ***Test 2:***

On this test ,will examine the valve with the pump by sending water from Storage to the Chromatography and will exercise control over them and see if we can introduce water into the Chromatography when we open the valve and turn on the pump,used pump here because the slot was from under so the liquid haved a low pressure,plus the Chromatography exist at high so must use pump to increase pressure.By pressing the button of valve with pump on the control screen and the usb board will give the order to open and turn on the pump.In short,will click the button of valve with pump on screen to open it ,turn on the pump and other click to close it and turn off the pump.

Figure 3.2: *The test 2 concerning open/close valve turn on/off pump.*

PRECONDITION	TEST ACTIVITY	POST CONDITION	TEST SUCCEED/FAILED
PIPES WITHOUT WATER	OPENING AND CLOSING THE VALVE WITH	FIRST CLICK VALVE OPEN AND PUMP TURN ON OTHER CLICK	SUCCEED

	TURNING ON AND OFF THE PUMP.	VALVE CLOSE AND PUMP TURN OFF	

7.1.3 ***Test 3:***

In this test ,will examine the two previous two tests together.

Figure 3.3: *The Test 3 concerning open/close 2 valves and turn on/off pump.*

PRECONDITION	TEST ACTIVITY	POST CONDITION	TEST SUCCEED/FAILED
PIPES WITHOUT WATER	OPENING AND CLOSING THE VALVE(V1) WITH TURNING ON AND OFF THE PUMP.	VALVE1 OPEN PUMP TURN ON VALVE 2 OPEN	SUCCEED
	OPENING AND CLOSING THE VALVE (V2) AT SAME TIME.	VALVE 1 CLOSE PUMP TURN OFF VALVE 2 CLOSE	SUCCEED

7.2 Testing the full automation of the MEGBI-VPP Processing:

On this part we will test the automation system that we have created before and get the results about it, this automation is for the production only so they haves the valves and pumps

necessary for it , we not including the valves necessary to cleaning.so as we see on algorithm before we have 35 valves 9 pumps so we needed 42 outputs ,divided on 3 boards each one have 16 outputs(digital analogue),but we will test the automation on board and screen cause we haven't the 35 valves 9 pumps so test the automation system by the board and the screen is enough cause that mean we will test the outputs and it's enough. Now we will shows all the results on list of pictures started by turning on the system by click on "Turn on":

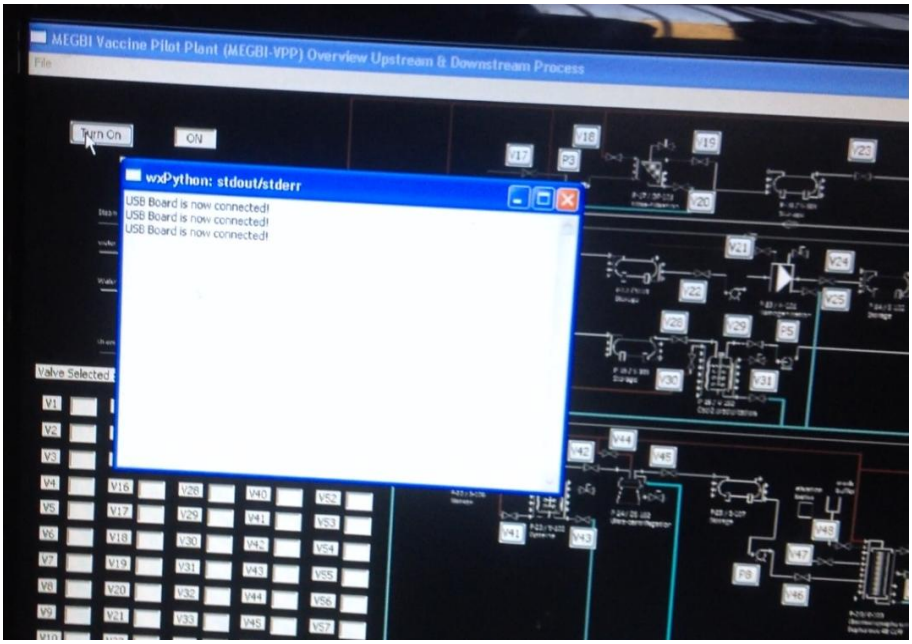


Figure 3.4: *The test of the automation system by Turn on.*

With clicking on "Turn On", the system is start (figure 3.4), and the full automation is started to test, this full automation can gives to us the controls of the fluid flow.

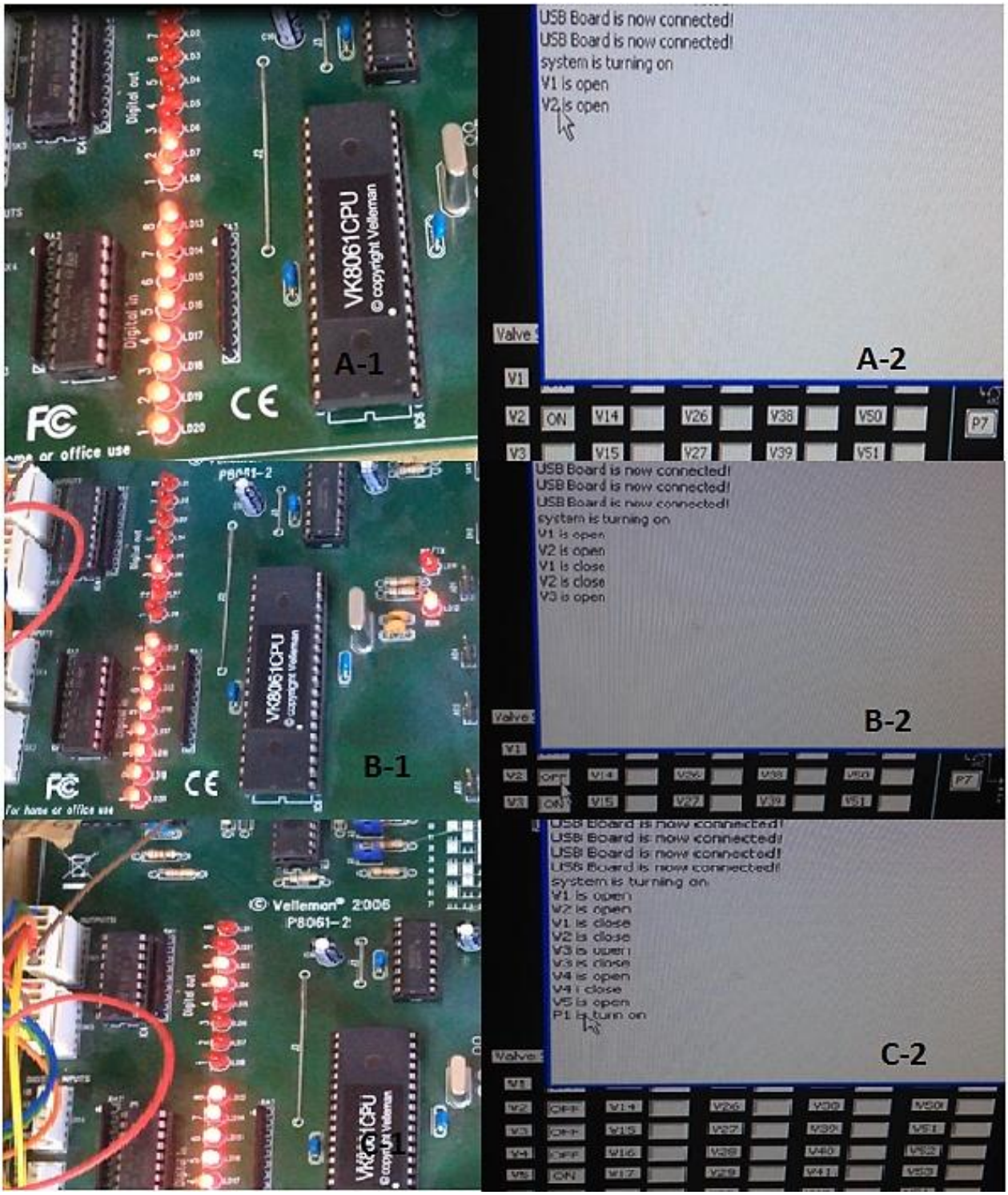


Figure 3.5: *The test of the full automation system step A,B and C.*

The automation is working good, as we see (figure 3.5) the valve V1 is open and the red light on board mean that , so when it close this red light will disappeared and that what we shows in this picture

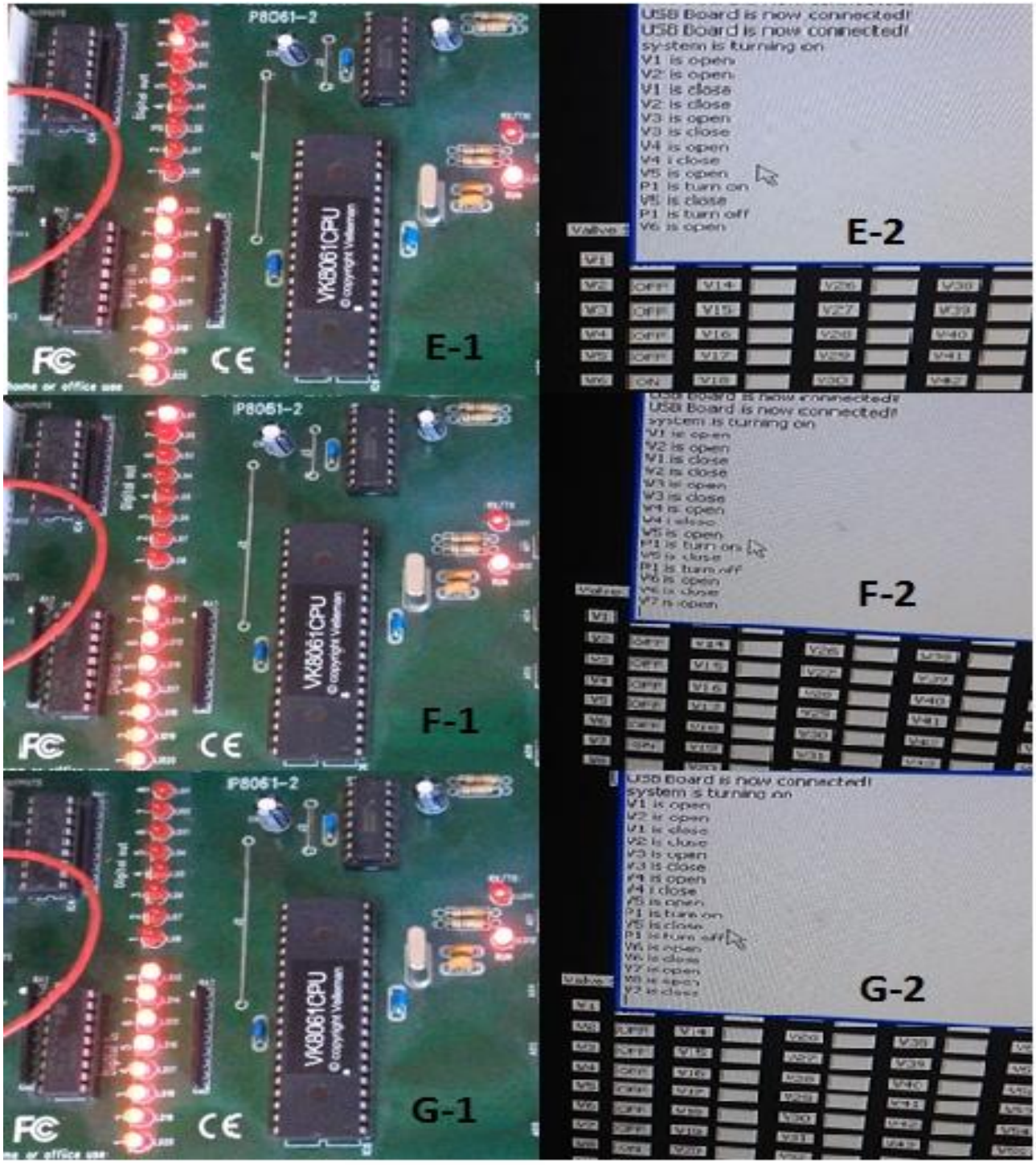


Figure 3.6: *The test of the full automation system step E,F and G.*

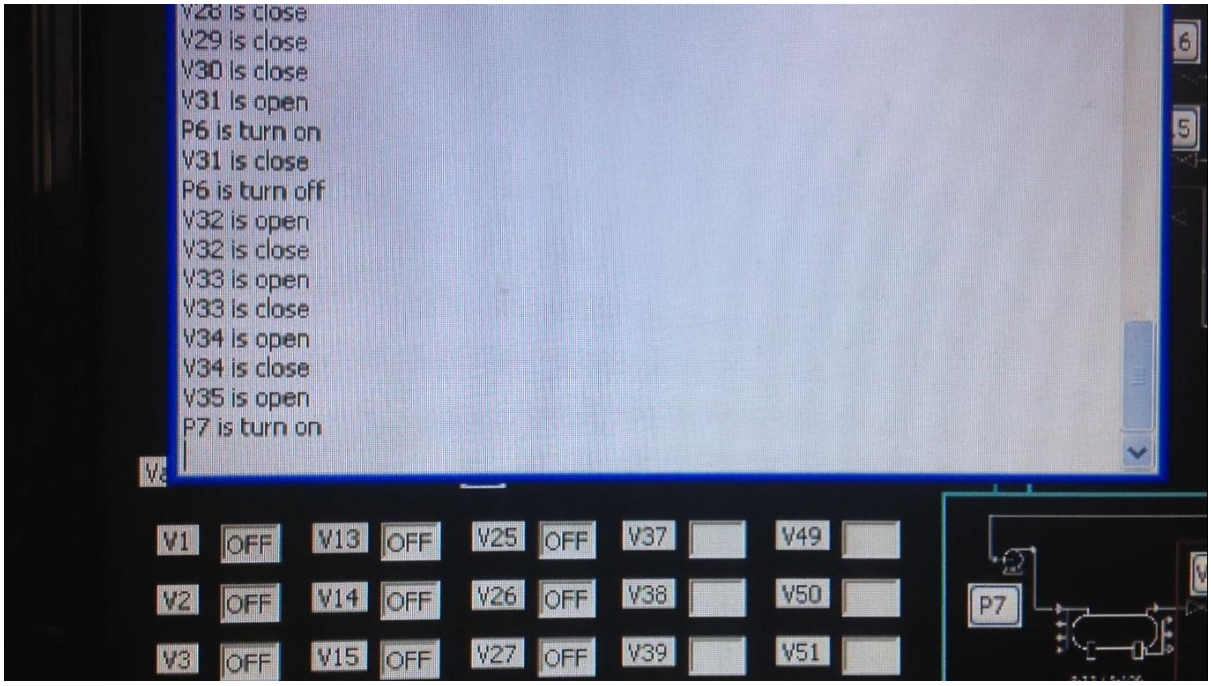
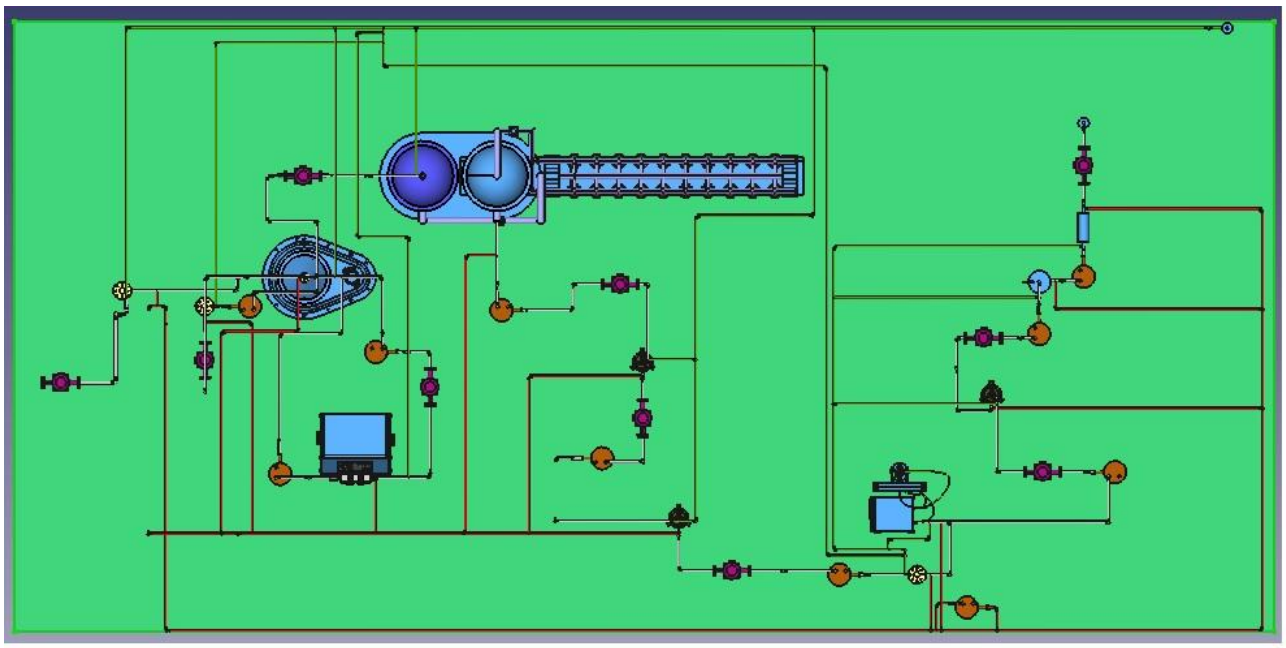


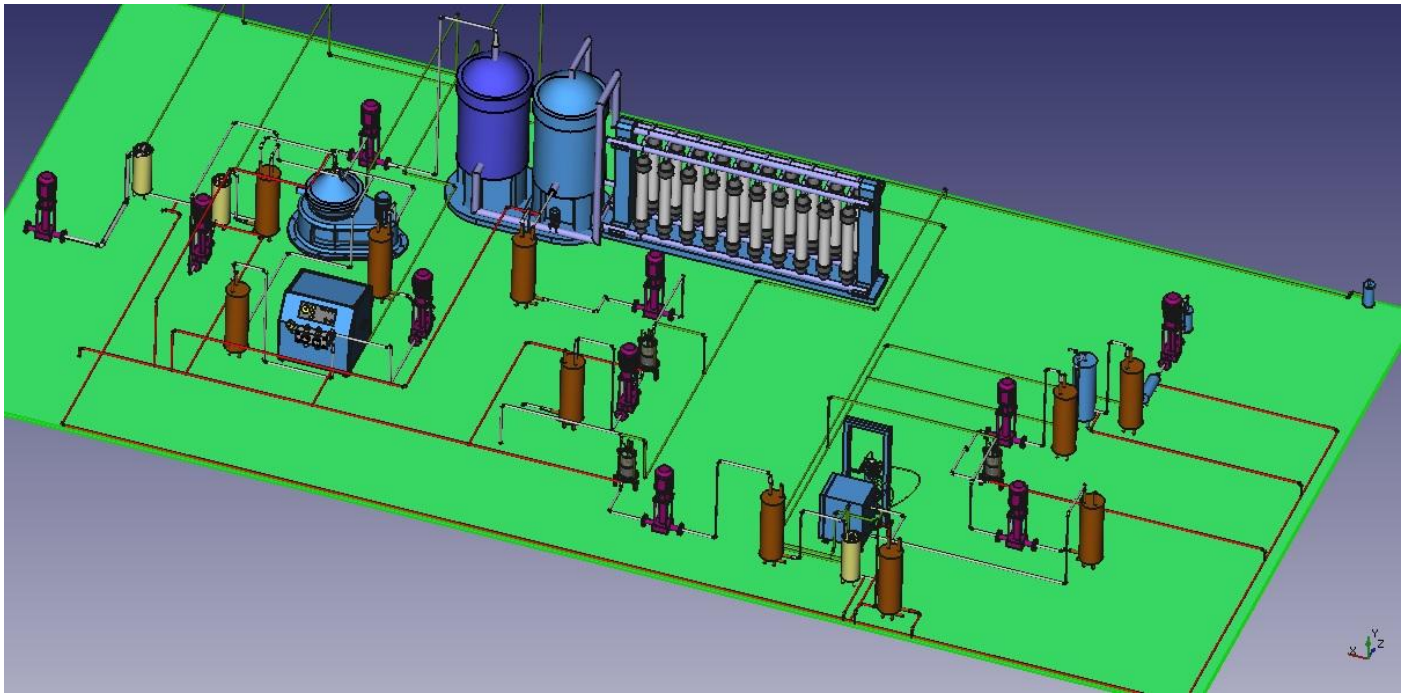
Figure 3.7: The test of the automation system by last step.

The system continues the step to finish at all the step of production and they automatically turned off, so the full automation is working and the test is succeed.

8. Integration of MEGBI–VPP based on Budget Plan for completing a minimal Prototype Plant

8.1 Overview

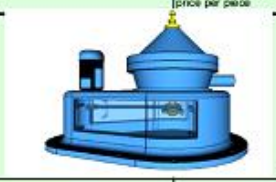




This version (minimal Prototype Plant) includes:

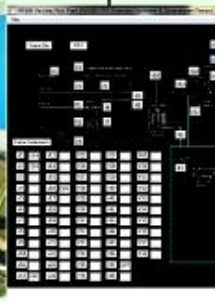
- smaller vessels
- simple disc stack centrifuge
- cost reduction option: vessel with chemical opening of cells instead of homogenizer
- disc stack centrifuge instead of ultracentifuge
- main new manufactoring: chromatographic columns (requires fine mechanical manufactoring facility)

8.2 **Cost Planning November 15**

Downstream Processing	# of mechnical pieces	average complexity/cost of part			price per piece	pieces needed	Costs
Disc Stack Centrifuge	20	$70			$1.400	1	$1.400
Homogenizer	10	$120			$1.200	1	$1.200
Chromatogr. Column (mech)	8	$50		 	$400	3	$1.200
Ultrafiltration Unit					$200	1	$200
Ultracentrifuge (used)	20	$70			$1.400	1	$1.400
Mixing/Storing tanks					$60	14	$840
						Total Devices	$6.240
						Integration (piping, ...) 1/2 inch	$1.500

Automation System		
Item	price per piece	# of pieces
pump	$60	10
electronic valve	$50	58
	Total price	$3.500





	Mechanical Parts USP	$0
	Mechanical Parts DSP	$7.740
	Automation System	$3.500
	Total MEGBI-VPP Rest Budget Requirement	**$11.240**

Alternative was:

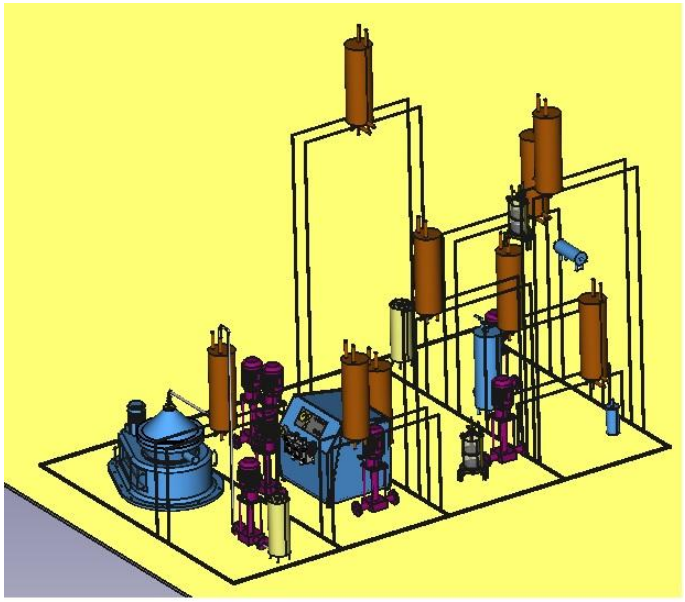
- Manufactoring of Devices and Integration. **Budget: 20.000 USD** (Material Costs: 10.000 USD, Personnel Cost: 4 months welder: 4000 USD, 4 months Eng.: 6000 USD)

Planned for Nov 15 - Feb 2016

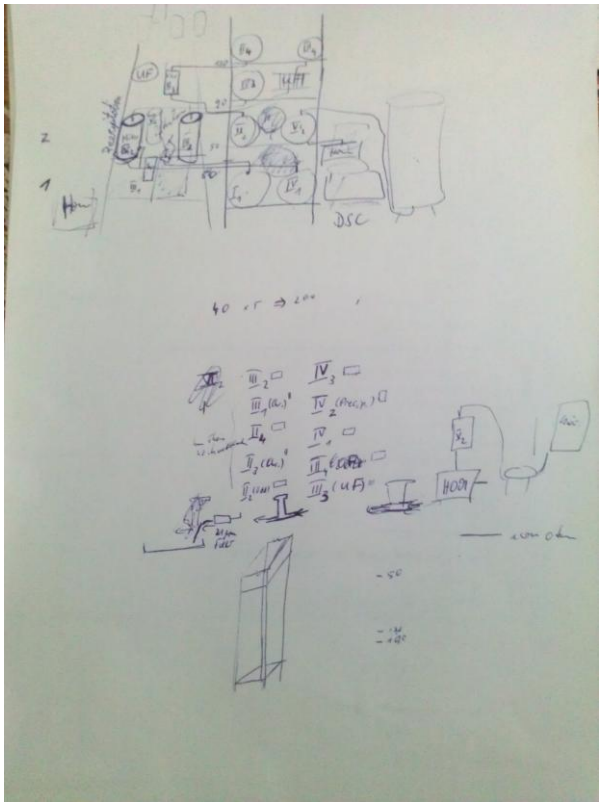
- Integration of Automation System to the whole plant (**Material Budget: 10 000 USD**),

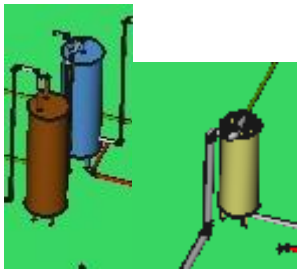
Planned for Mar-Aug 16 (as Master Thesis)

8.3 Manufactoring Planning of minimal system



8.3.1 *Package 1: vessels for storing and mixing*





Placing of storages based on flowdiagram

Costs: 1500$

8.3.2 *Package 2: Chromatographic Columns, Disc Stack Centrifuge, Homogenizer*

Costs: 1400$

8.3.3 *Package 3: Pumps & Valves*

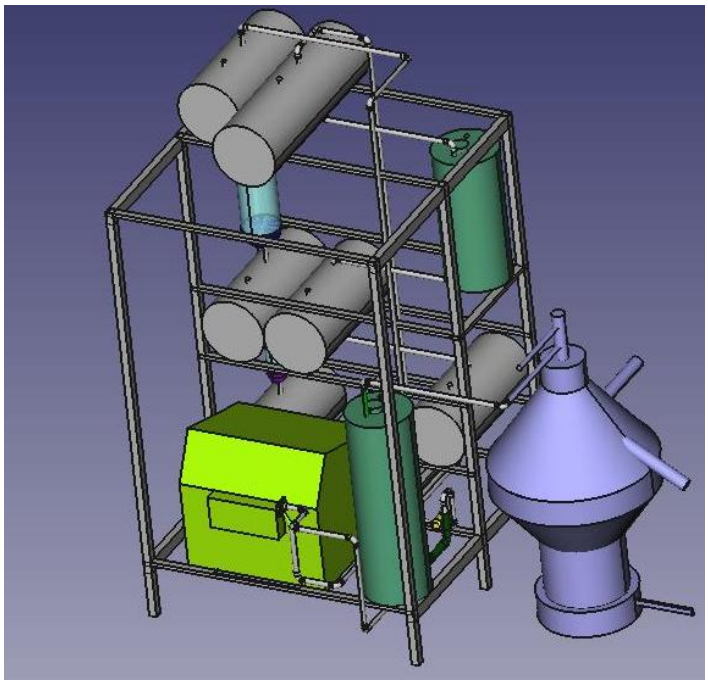
Costs: 3500$

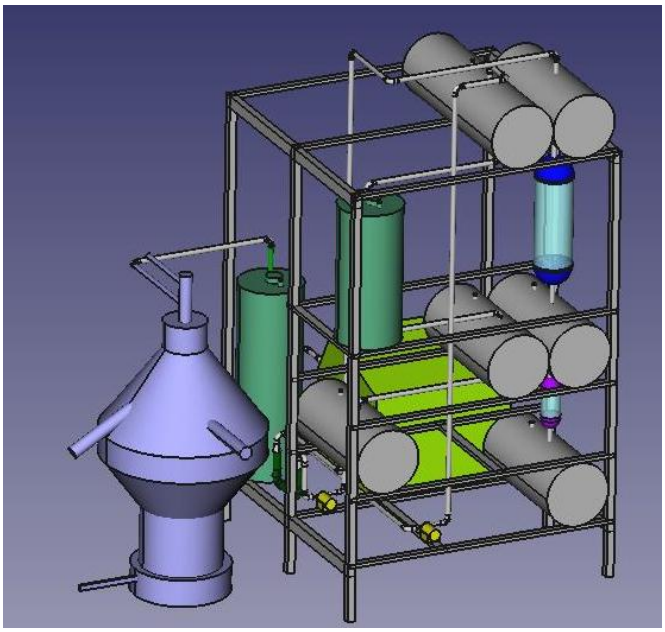
8.3.1 *Package 4: Piping*

Costs: 1500$

8.4 Manufactored 24.12.-30.12.2015 (based on minimal system)

8.4.1 *Design*





8.4.2 *Costs*

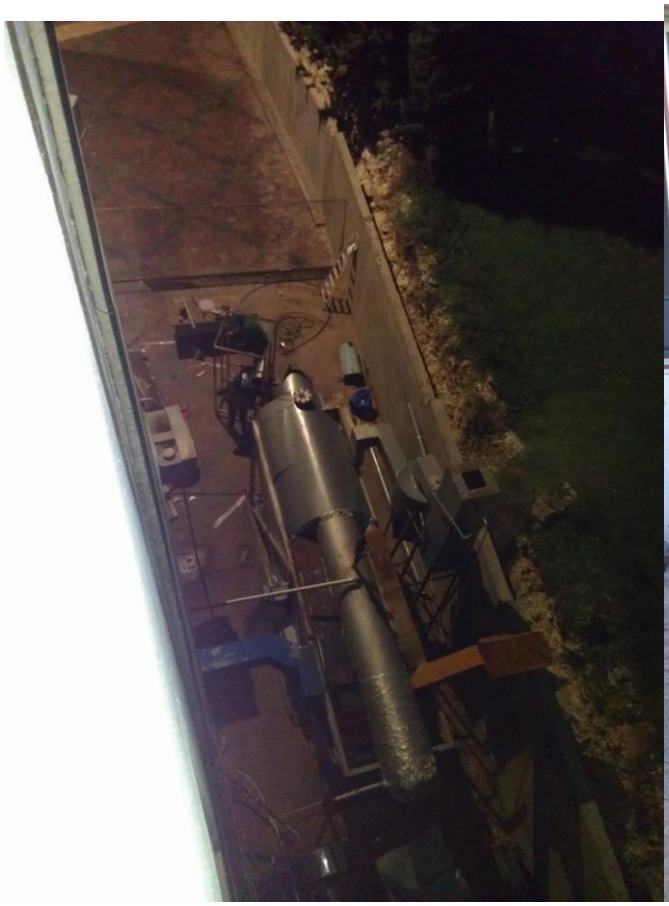
1 sheet stainless steel 316: 1m x2m x 2mm -> about 150$

1 sheet stainless steel 304: 1m x2m x 2mm -> about 100$

stage material: 6m x 38 mm x 38 mm -> 35$

Material	Staff
storage & mixing tanks (stainless steel 316) and metal stage (stainless steel) -> about 850 $ 4 rolls (15$) 1 pump stainless steel 1,5 bar (55$)	Welder (Mohammad Qammah) for 7 days (all in 315 $)

There were total costs of 1230 $.



8.5 Still missing (to be manufactored/buyed in 2016 insha Allah)

piping

disc stack centrifuge

homogenizer

2 columns

Adaption of automation system

For manufactoring disc stack centrifuge and homoginizer a CNC machine is needed.

9. Annex

9.1 Annex 1a: How to install Python

- download python from www.python.org , we downloading python 3.5 version for windows 8

-install Python by run the program and continue the steps.

Needed a series of packages:

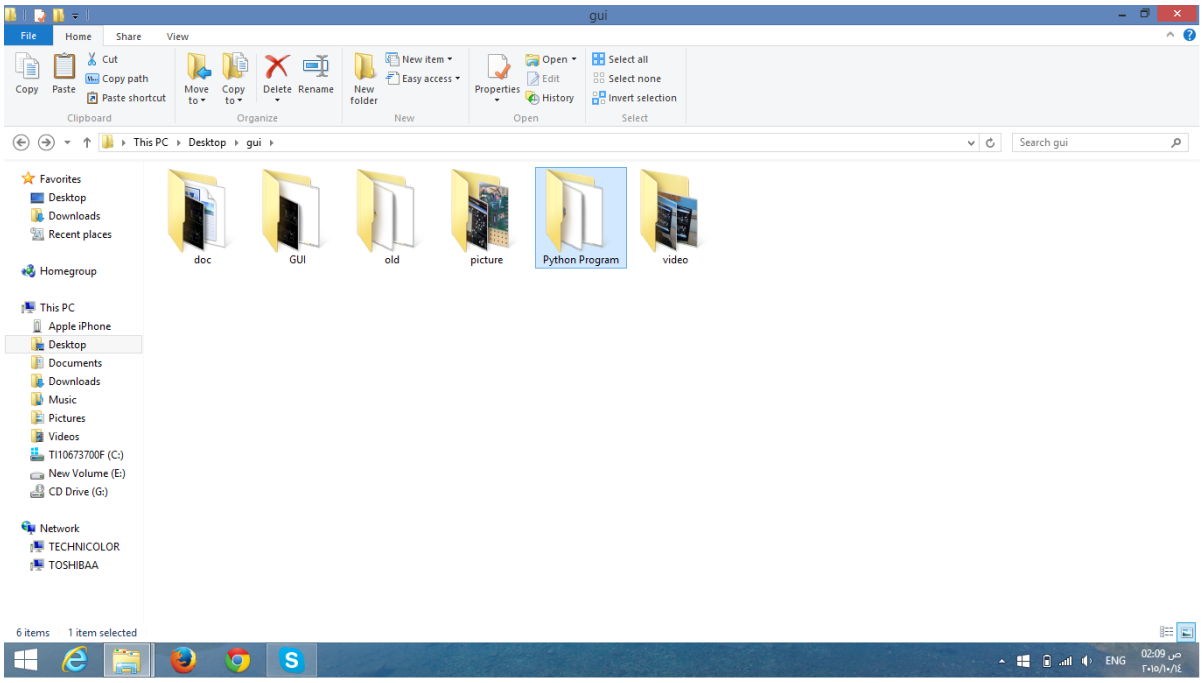
- Numpy-1.9.2
- Py2exe
- PyDDE
- wxPython-3.0.2.0

To install each package u should to open a command and write:

“C:\location of your python\Scripts\Pip install package name.”

We have putted a CD with this document

concerning:



9.2 Annex 1b: Connecting Veleman Board, Drivers

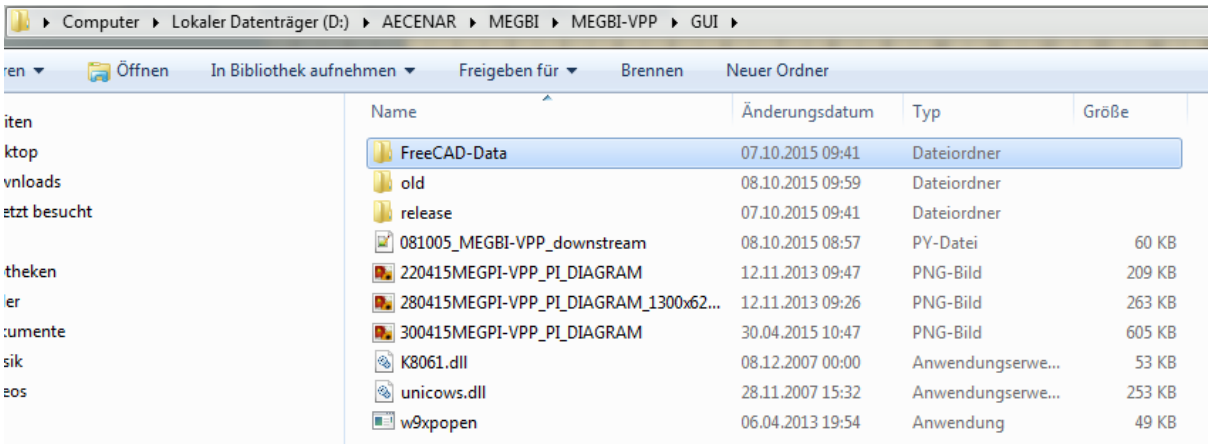
For the veleman board there are needed drivers. These drivers are only compatible to Windows XP, not to Windows 7. Following files are needed:

K8061.dll

unicows.dll

w9xpopen

The drivers must be in the same folder as the .py file.



9.3 Annex 2a: phython automation code only in maintance mode (without full automation)

```
##!/usr/bin/env python
## -*- coding: iso-8859-15 -*-
import sys
from tkinter import *
import random
import time
import wx
import thread



#**********************                    panel frame                      **********************
#*****************************************************************************************************

BACKGROUND_IMAGENAME = "220415MEGPI-VPP_PI_DIAGRAM.PNG"


class MyBackgroundPanel(wx.Panel):

    def __init__(self, parent):
        wx.Panel.__init__(self, parent)
        self.bmp = wx.Bitmap(BACKGROUND_IMAGENAME)
        self.SetSize(self.bmp.GetSize())
        self.Bind(wx.EVT_PAINT, self.on_paint)


    def on_paint(self, event = None):
        dc = wx.BufferedPaintDC(self, self.bmp)

class MyFrame(wx.Frame):

    def __init__(self, parent = None, title = "MEGBI Vaccine Pilot Plant (MEGBI-VPP) Overview Upstream & Downstream Process"):

        self.testUSB                = True
        self.dll                    = None
        self.USBAdr0                = 0
        self.USBOpened              = False
        self.counterUSBBoards       = 3

        wx.Frame.__init__(self, parent, -1, title)

        panel = MyBackgroundPanel(self)

        LABELSTYLE = wx.BORDER_SUNKEN | wx.ST_NO_AUTORESIZE | wx.ALIGN_CENTER_HORIZONTAL


        menuFile = wx.Menu()
        menuFile.Append(1, "&About...")
        menuFile.AppendSeparator()
        menuFile.Append(2, "Exit")
        menuBar = wx.MenuBar()
        menuBar.Append(menuFile, "File")
        self.SetMenuBar(menuBar)
        self.CreateStatusBar()
        self.SetStatusText("Welcome to MEGPI Project!")
        self.Bind(wx.EVT_MENU, self.OnAbout, id=1)
        self.Bind(wx.EVT_MENU, self.OnQuit, id=2)
```

```
        # temperature sensors value

wx.StaticText(panel,-1," Temperature Value ",(100,12))
self.temp_Vaporizer_out = wx.StaticText(panel, size = (26, -1), pos = (200, 10), style = LABELSTYLE )



new_value = str(self.dll.ReadAnalogChannel(0,1))
self.temp_Vaporizer_out.SetLabel(new_value)
self.temp_Vaporizer_out.Refresh()


             #Valves

                #V1
self.button_V_1 = wx.Button(panel, -1, "V1", pos=(95,100),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V1, self.button_V_1)

                #V2
self.button_V_2 = wx.Button(panel, -1, "V2", pos=(150,90),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V2, self.button_V_2)
                #V3
self.button_V_3 = wx.Button(panel, -1, "V3", pos=(150,135),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V3, self.button_V_3)


                #V4
self.button_V_4 = wx.Button(panel, -1, "V4", pos=(150,265),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V4, self.button_V_4)

                #V5
self.button_V_5 = wx.Button(panel, -1, "V5", pos=(210,135),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V5, self.button_V_5)


                #V6
self.button_V_6 = wx.Button(panel, -1, "V6", pos=(215,185),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V6, self.button_V_6)

                #V7
self.button_V_7 = wx.Button(panel, -1, "V7", pos=(215,220),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V7, self.button_V_7)

                #V8
self.button_V_8 = wx.Button(panel, -1, "V8", pos=(282,180),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V8, self.button_V_8)

                #V9
self.button_V_9 = wx.Button(panel, -1, "V9", pos=(282,158),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V9, self.button_V_9)

                #V10
self.button_V_10 = wx.Button(panel, -1, "V10", pos=(322,110),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V10, self.button_V_10)

                #V11
self.button_V_11 = wx.Button(panel, -1, "V11", pos=(380,250),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V11, self.button_V_11)

                #V12
self.button_V_12 = wx.Button(panel, -1, "V12", pos=(413,195),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V12, self.button_V_12)

                #V13
self.button_V_13 = wx.Button(panel, -1, "V13", pos=(380,130),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V13, self.button_V_13)

                #V14
self.button_V_14 = wx.Button(panel, -1, "V14", pos=(410,100),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V14, self.button_V_14)
```

```
                #V15
self.button_V_15 = wx.Button(panel, -1, "V15", pos=(460,112),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V15, self.button_V_15)

                #V16
self.button_V_16 = wx.Button(panel, -1, "V16", pos=(460,73),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V16, self.button_V_16)

                #V17
self.button_V_17 = wx.Button(panel, -1, "V17", pos=(460,38),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V17, self.button_V_17)



                #V18
self.button_V_18 = wx.Button(panel, -1, "V18", pos=(522,18),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V18, self.button_V_18)

                #V19
self.button_V_19 = wx.Button(panel, -1, "V19", pos=(635,18),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V19, self.button_V_19)

                #V20
self.button_V_20 = wx.Button(panel, -1, "V20", pos=(635,72),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V20, self.button_V_20)

                #V21
self.button_V_21 = wx.Button(panel, -1, "V21", pos=(675,112),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V21, self.button_V_21)

                #V22
self.button_V_22 = wx.Button(panel, -1, "V22", pos=(635,155),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V22, self.button_V_22)

                #V23
self.button_V_23 = wx.Button(panel, -1, "V23", pos=(780,18),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V23, self.button_V_23)

                #V24
self.button_V_24 = wx.Button(panel, -1, "V24", pos=(770,120),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V24, self.button_V_24)

                #V25
self.button_V_25 = wx.Button(panel, -1, "V25", pos=(770,155),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V25, self.button_V_25)

                #V26
self.button_V_26 = wx.Button(panel, -1, "V26", pos=(870,155),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V26, self.button_V_26)


                #V27
self.button_V_27 = wx.Button(panel, -1, "V27", pos=(510,185),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V27, self.button_V_27)

                #V28
self.button_V_28 = wx.Button(panel, -1, "V28", pos=(620,185),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V28, self.button_V_28)

                #V29
self.button_V_29 = wx.Button(panel, -1, "V29", pos=(680,185),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V29, self.button_V_29)

                #V30
self.button_V_30 = wx.Button(panel, -1, "V30", pos=(620,240),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V30, self.button_V_30)

                #V31
self.button_V_31 = wx.Button(panel, -1, "V31", pos=(710,235),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V31, self.button_V_31)
```

```
                #V32
self.button_V_32 = wx.Button(panel, -1, "V32", pos=(950,182),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V32, self.button_V_32)

                #V33
self.button_V_33 = wx.Button(panel, -1, "V33", pos=(977,182),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V33, self.button_V_33)

                #V34
self.button_V_34 = wx.Button(panel, -1, "V34", pos=(1083,188),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V34, self.button_V_34)

                #V35
self.button_V_35 = wx.Button(panel, -1, "V35", pos=(1083,220),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V35, self.button_V_35)

                #V36
self.button_V_36 = wx.Button(panel, -1, "V36", pos=(1170,110),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V36, self.button_V_36)

                #V37
self.button_V_37 = wx.Button(panel, -1, "V37", pos=(1200,110),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V37, self.button_V_37)

                #V38
self.button_V_38 = wx.Button(panel, -1, "V38", pos=(1240,150),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V38, self.button_V_38)

                #V39
self.button_V_39 = wx.Button(panel, -1, "V39", pos=(1273,205),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V39, self.button_V_39)

                #V40
self.button_V_40 = wx.Button(panel, -1, "V40", pos=(483,314),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V40, self.button_V_40)

               #V41
self.button_V_41 = wx.Button(panel, -1, "V41", pos=(478,398),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V41, self.button_V_41)
               #V42
self.button_V_42 = wx.Button(panel, -1, "V42", pos=(540,314),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V42, self.button_V_42)
               #V43
self.button_V_43 = wx.Button(panel, -1, "V43", pos=(548,398),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V43, self.button_V_43)
               #V44
self.button_V_44 = wx.Button(panel, -1, "V44", pos=(580,300),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V44, self.button_V_44)
               #V45
self.button_V_45 = wx.Button(panel, -1, "V45", pos=(620,314),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V45, self.button_V_45)
               #V46
self.button_V_46 = wx.Button(panel, -1, "V46", pos=(760,440),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V46, self.button_V_46)
               #V47
self.button_V_47 = wx.Button(panel, -1, "V47", pos=(760,400),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V47, self.button_V_47)
               #V48
self.button_V_48 = wx.Button(panel, -1, "V48", pos=(783,375),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V48, self.button_V_48)
               #V49
self.button_V_49 = wx.Button(panel, -1, "V49", pos=(880,428),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V49, self.button_V_49)
               #V50
self.button_V_50 = wx.Button(panel, -1, "V50", pos=(970,377),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V50, self.button_V_50)
               #V51
self.button_V_51 = wx.Button(panel, -1, "V51", pos=(963,450),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V51, self.button_V_51)
               #V52
self.button_V_52 = wx.Button(panel, -1, "V52", pos=(1005,470),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V52, self.button_V_52)
```

```
                #V52
self.button_V_52 = wx.Button(panel, -1, "V52", pos=(1005,470),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V52, self.button_V_52)
                #V53
self.button_V_53 = wx.Button(panel, -1, "V53", pos=(1030,395),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V53, self.button_V_53)
                #V54
self.button_V_54 = wx.Button(panel, -1, "V54", pos=(1080,395),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V54, self.button_V_54)
                #V55
self.button_V_55 = wx.Button(panel, -1, "V55", pos=(1118,377),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V55, self.button_V_55)
                #V56
self.button_V_56 = wx.Button(panel, -1, "V56", pos=(1158,468),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V56, self.button_V_56)
                #V57
self.button_V_57 = wx.Button(panel, -1, "V57", pos=(1190,420),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V57, self.button_V_57)
                #V58
self.button_V_58 = wx.Button(panel, -1, "V58", pos=(710,580),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V58, self.button_V_58)


            #Pumps

                #P1
self.button_P_1 = wx.Button(panel, -1, "P1", pos=(150,180),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P1, self.button_P_1)
                #P2
self.button_P_2 = wx.Button(panel, -1, "P2", pos=(150,220),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P2, self.button_P_2)
                #P3
self.button_P_3 = wx.Button(panel, -1, "P3", pos=(510,40),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P3, self.button_P_3)
                #P4
self.button_P_4 = wx.Button(panel, -1, "P4", pos=(880,18),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P4, self.button_P_4)

                #P5
self.button_P_5 = wx.Button(panel, -1, "P5", pos=(730,187),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P5, self.button_P_5)

                #P6
self.button_P_6 = wx.Button(panel, -1, "P6", pos=(1083,130),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P6, self.button_P_6)

                #P7
self.button_P_7 = wx.Button(panel, -1, "P7", pos=(380,328),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P7, self.button_P_7)

                #P8
self.button_P_8 = wx.Button(panel, -1, "P8", pos=(710,423),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P8, self.button_P_8)

                #P9
self.button_P_9 = wx.Button(panel, -1, "P9", pos=(900,380),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P9, self.button_P_9)

                #P10
self.button_P_10 = wx.Button(panel, -1, "P10", pos=(1200,380),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P10, self.button_P_10)
```

```
monitoring valve status


    #Text column 1
 wx.StaticText(panel,20," Valve Selected : ",(10,270))
 wx.StaticText(panel,-1," V1 ",(18,300))
 wx.StaticText(panel,-1," V2 ",(18,326))
 wx.StaticText(panel,-1," V3 ",(18,352))
 wx.StaticText(panel,-1," V4 ",(18,378))
 wx.StaticText(panel,-1," V5 ",(18,402))
 wx.StaticText(panel,-1," V6 ",(18,428))
 wx.StaticText(panel,-1," V7 ",(18,454))
 wx.StaticText(panel,-1," V8 ",(18,480))
 wx.StaticText(panel,-1," V9 ",(18,506))
 wx.StaticText(panel,-1," V10 ",(18,532))
 wx.StaticText(panel,-1," V11 ",(18,558))
 wx.StaticText(panel,-1," V12 ",(18,584))

    #Text column 2
 wx.StaticText(panel,-1," V13 ",(85,300))
 wx.StaticText(panel,-1," V14 ",(85,326))
 wx.StaticText(panel,-1," V15 ",(85,352))
 wx.StaticText(panel,-1," V16 ",(85,378))
 wx.StaticText(panel,-1," V17 ",(85,402))
 wx.StaticText(panel,-1," V18 ",(85,428))
 wx.StaticText(panel,-1," V19 ",(85,454))
 wx.StaticText(panel,-1," V20 ",(85,480))
 wx.StaticText(panel,-1," V21 ",(85,506))
 wx.StaticText(panel,-1," V22 ",(85,532))
 wx.StaticText(panel,-1," V23 ",(85,558))
 wx.StaticText(panel,-1," V24 ",(85,584))

  #Text column 3
 wx.StaticText(panel,-1," V25 ",(155,300))
 wx.StaticText(panel,-1," V26 ",(155,326))
 wx.StaticText(panel,-1," V27 ",(155,352))
 wx.StaticText(panel,-1," V28 ",(155,378))
 wx.StaticText(panel,-1," V29 ",(155,402))
 wx.StaticText(panel,-1," V30 ",(155,428))
 wx.StaticText(panel,-1," V31 ",(155,454))
 wx.StaticText(panel,-1," V32 ",(155,480))
 wx.StaticText(panel,-1," V33 ",(155,506))
 wx.StaticText(panel,-1," V34 ",(155,532))
 wx.StaticText(panel,-1," V35 ",(155,558))
 wx.StaticText(panel,-1," V36 ",(155,584))

 #Text column 4
 wx.StaticText(panel,-1," V37 ",(223,300))
 wx.StaticText(panel,-1," V38 ",(223,326))
 wx.StaticText(panel,-1," V39 ",(223,352))
 wx.StaticText(panel,-1," V40 ",(223,378))
 wx.StaticText(panel,-1," V41 ",(223,402))
 wx.StaticText(panel,-1," V42 ",(223,428))
 wx.StaticText(panel,-1," V43 ",(223,454))
 wx.StaticText(panel,-1," V44 ",(223,480))
 wx.StaticText(panel,-1," V45 ",(223,506))
 wx.StaticText(panel,-1," V46 ",(223,532))
 wx.StaticText(panel,-1," V47 ",(223,558))
 wx.StaticText(panel,-1," V48 ",(223,584))

#Text column 5
wx.StaticText(panel,-1," V49 ",(293,300))
wx.StaticText(panel,-1," V50 ",(293,326))
wx.StaticText(panel,-1," V51 ",(293,352))
wx.StaticText(panel,-1," V52 ",(293,378))
wx.StaticText(panel,-1," V53 ",(293,402))
wx.StaticText(panel,-1," V54 ",(293,428))
wx.StaticText(panel,-1," V55 ",(293,454))
wx.StaticText(panel,-1," V56 ",(293,480))
wx.StaticText(panel,-1," V57 ",(293,506))
wx.StaticText(panel,-1," V58 ",(293,532))
```

```
    #Label  Column 1
self.Valve1 = wx.StaticText(panel, size = (26, -1), pos = (45, 300), style = LABELSTYLE)
self.Valve2 = wx.StaticText(panel, size = (26, -1), pos = (45, 326), style = LABELSTYLE)
self.Valve3 = wx.StaticText(panel, size = (26, -1), pos = (45, 352), style = LABELSTYLE)
self.Valve4 = wx.StaticText(panel, size = (26, -1), pos = (45, 378), style = LABELSTYLE)
self.Valve5 = wx.StaticText(panel, size = (26, -1), pos = (45, 402), style = LABELSTYLE)
self.Valve6 = wx.StaticText(panel, size = (26, -1), pos = (45, 428), style = LABELSTYLE)
self.Valve7 = wx.StaticText(panel, size = (26, -1), pos = (45, 454), style = LABELSTYLE)
self.Valve8 = wx.StaticText(panel, size = (26, -1), pos = (45, 480), style = LABELSTYLE)
self.Valve9 = wx.StaticText(panel, size = (26, -1), pos = (45, 506), style = LABELSTYLE)
self.Valve10 = wx.StaticText(panel, size = (26, -1), pos = (45, 532), style = LABELSTYLE)
self.Valve11 = wx.StaticText(panel, size = (26, -1), pos = (45, 558), style = LABELSTYLE)
self.Valve12 = wx.StaticText(panel, size = (26, -1), pos = (45, 584), style = LABELSTYLE)



    #Label column 2
self.Valve13 = wx.StaticText(panel, size = (26, -1), pos = (115, 300), style = LABELSTYLE)
self.Valve14 = wx.StaticText(panel, size = (26, -1), pos = (115, 326), style = LABELSTYLE)
self.Valve15 = wx.StaticText(panel, size = (26, -1), pos = (115, 352), style = LABELSTYLE)
self.Valve16 = wx.StaticText(panel, size = (26, -1), pos = (115, 378), style = LABELSTYLE)
self.Valve17 = wx.StaticText(panel, size = (26, -1), pos = (115, 402), style = LABELSTYLE)
self.Valve18 = wx.StaticText(panel, size = (26, -1), pos = (115, 428), style = LABELSTYLE)
self.Valve19 = wx.StaticText(panel, size = (26, -1), pos = (115, 454), style = LABELSTYLE)
self.Valve20 = wx.StaticText(panel, size = (26, -1), pos = (115, 480), style = LABELSTYLE)
self.Valve21 = wx.StaticText(panel, size = (26, -1), pos = (115, 506), style = LABELSTYLE)
self.Valve22 = wx.StaticText(panel, size = (26, -1), pos = (115, 532), style = LABELSTYLE)
self.Valve23 = wx.StaticText(panel, size = (26, -1), pos = (115, 558), style = LABELSTYLE)
self.Valve24 = wx.StaticText(panel, size = (26, -1), pos = (115, 584), style = LABELSTYLE)

 #Label column 3

self.Valve25 = wx.StaticText(panel, size = (26, -1), pos = (185, 300), style = LABELSTYLE)
self.Valve26 = wx.StaticText(panel, size = (26, -1), pos = (185, 326), style = LABELSTYLE)
self.Valve27 = wx.StaticText(panel, size = (26, -1), pos = (185, 352), style = LABELSTYLE)
self.Valve28 = wx.StaticText(panel, size = (26, -1), pos = (185, 378), style = LABELSTYLE)
self.Valve29 = wx.StaticText(panel, size = (26, -1), pos = (185, 402), style = LABELSTYLE)
self.Valve30 = wx.StaticText(panel, size = (26, -1), pos = (185, 428), style = LABELSTYLE)
self.Valve31 = wx.StaticText(panel, size = (26, -1), pos = (185, 454), style = LABELSTYLE)
self.Valve32 = wx.StaticText(panel, size = (26, -1), pos = (185, 480), style = LABELSTYLE)
self.Valve33 = wx.StaticText(panel, size = (26, -1), pos = (185, 506), style = LABELSTYLE)
self.Valve34 = wx.StaticText(panel, size = (26, -1), pos = (185, 532), style = LABELSTYLE)
self.Valve35 = wx.StaticText(panel, size = (26, -1), pos = (185, 558), style = LABELSTYLE)
self.Valve36 = wx.StaticText(panel, size = (26, -1), pos = (185, 584), style = LABELSTYLE)

#Label column 4

self.Valve37 = wx.StaticText(panel, size = (26, -1), pos = (253, 300), style = LABELSTYLE)
self.Valve38 = wx.StaticText(panel, size = (26, -1), pos = (253, 326), style = LABELSTYLE)
self.Valve39 = wx.StaticText(panel, size = (26, -1), pos = (253, 352), style = LABELSTYLE)
self.Valve40 = wx.StaticText(panel, size = (26, -1), pos = (253, 378), style = LABELSTYLE)
self.Valve41 = wx.StaticText(panel, size = (26, -1), pos = (253, 402), style = LABELSTYLE)
self.Valve42 = wx.StaticText(panel, size = (26, -1), pos = (253, 428), style = LABELSTYLE)
self.Valve43 = wx.StaticText(panel, size = (26, -1), pos = (253, 454), style = LABELSTYLE)
self.Valve44 = wx.StaticText(panel, size = (26, -1), pos = (253, 480), style = LABELSTYLE)
self.Valve45 = wx.StaticText(panel, size = (26, -1), pos = (253, 506), style = LABELSTYLE)
self.Valve46 = wx.StaticText(panel, size = (26, -1), pos = (253, 532), style = LABELSTYLE)
self.Valve47 = wx.StaticText(panel, size = (26, -1), pos = (253, 558), style = LABELSTYLE)
self.Valve48 = wx.StaticText(panel, size = (26, -1), pos = (253, 584), style = LABELSTYLE)
#Label column 5

self.Valve49 = wx.StaticText(panel, size = (26, -1), pos = (323, 300), style = LABELSTYLE)
self.Valve50 = wx.StaticText(panel, size = (26, -1), pos = (323, 326), style = LABELSTYLE)
self.Valve51 = wx.StaticText(panel, size = (26, -1), pos = (323, 352), style = LABELSTYLE)
self.Valve52 = wx.StaticText(panel, size = (26, -1), pos = (323, 378), style = LABELSTYLE)
self.Valve53 = wx.StaticText(panel, size = (26, -1), pos = (323, 402), style = LABELSTYLE)
self.Valve54 = wx.StaticText(panel, size = (26, -1), pos = (323, 428), style = LABELSTYLE)
self.Valve55 = wx.StaticText(panel, size = (26, -1), pos = (323, 454), style = LABELSTYLE)
self.Valve56 = wx.StaticText(panel, size = (26, -1), pos = (323, 480), style = LABELSTYLE)
self.Valve57 = wx.StaticText(panel, size = (26, -1), pos = (323, 506), style = LABELSTYLE)
self.Valve58 = wx.StaticText(panel, size = (26, -1), pos = (323, 532), style = LABELSTYLE)


self.P1 = wx.StaticText(panel, size = (26, -1), pos = (60, 10), style = LABELSTYLE)
self.P2 = wx.StaticText(panel, size = (26, -1), pos = (60, 40), style = LABELSTYLE)
wx.StaticText(panel,-1," P1 ",(20,10))
wx.StaticText(panel,-1," P2 ",(20,40))
```

```
#**********************                Run USB System                        **********************
#*****************************************************************************************************
    def OpenUSBBoardThread(self):
        self.dll = windll.K8061
        i = self.counterUSBBoards
        for doit in range(0,i+1):
            try:
                self.dll.OpenDevice()
                self.USBOpened = True
# debug info
                print ('USB Board is now connected!')
#end debug info
            except:
                txt = ('Please Check USB Board connection')
                print ('txt')
                return


############################       Menu Bar              ##########################
#*****************************************************************************
    def OnAbout(self, event):
        wx.MessageBox("This is a Screen controller of MEGPI Project",
                      "Welcome to my python", wx.OK | wx.ICON_INFORMATION, self)
    def OnQuit(self, event):
        self.Close()
    def on_timer(self, event = None):
        new_value = str(windll.K8061.ReadAnalogChannel(0,1))

#**********************                      Define Valvue                     ********************
#*****************************************************************************************************
    def V1(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'

# open the USB board

            if  self.Valve1.GetLabel()==valve_status_OFF:
                self.Valve1.SetLabel(valve_status_ON)
                self.OpenUSBBoardThread()
                self.dll.SetDigitalChannel(0,1)
                wx.MessageBox("V1 is open ", "Open", wx.OK|wx.ICON_INFORMATION)
                self.timer = wx.Timer()
                self.timer.Bind(wx.EVT_TIMER, self.on_timer)
                self.timer.Start(0)

            else:
                self.Valve1.SetLabel(valve_status_OFF)
                time.sleep(2)
                self.dll.ClearDigitalChannel(0,1)
                print 'Digital Channel Cleared, V1 turn off'

    def V2(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'

            if self.Valve2.GetLabel()==valve_status_ON:
                self.Valve2.SetLabel(valve_status_OFF)

            else:
                self.Valve2.SetLabel(valve_status_ON)
```

```
    def V3(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve3.GetLabel()==valve_status_ON:
                self.Valve3.SetLabel(valve_status_OFF)
            else:
                self.Valve3.SetLabel(valve_status_ON)

    def V4(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve4.GetLabel()==valve_status_ON:
                self.Valve4.SetLabel(valve_status_OFF)
            else:
               self.Valve4.SetLabel(valve_status_ON)


    def V5(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve5.GetLabel()==valve_status_ON:
                self.Valve5.SetLabel(valve_status_OFF)
            else:
                self.Valve5.SetLabel(valve_status_ON)

    def V6(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve6.GetLabel()==valve_status_ON:
                self.Valve6.SetLabel(valve_status_OFF)
            else:
                self.Valve6.SetLabel(valve_status_ON)

def V7(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve7.GetLabel()==valve_status_ON:
            self.Valve7.SetLabel(valve_status_OFF)
        else:
            self.Valve7.SetLabel(valve_status_ON)
```

```
def V8(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve8.GetLabel()==valve_status_ON:
            self.Valve8.SetLabel(valve_status_OFF)
        else:
            self.Valve8.SetLabel(valve_status_ON)

def V9(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve9.GetLabel()==valve_status_ON:
            self.Valve9.SetLabel(valve_status_OFF)
        else:
            self.Valve9.SetLabel(valve_status_ON)

def V10(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve10.GetLabel()==valve_status_ON:
            self.Valve10.SetLabel(valve_status_OFF)
        else:
            self.Valve10.SetLabel(valve_status_ON)

def V11(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve11.GetLabel()==valve_status_ON:
            self.Valve11.SetLabel(valve_status_OFF)

        else:
            self.Valve11.SetLabel(valve_status_ON)

 def V12(self, event):
         valve_status_ON = 'ON'
         valve_status_OFF = 'OFF'
         if self.Valve12.GetLabel()==valve_status_ON:
             self.Valve12.SetLabel(valve_status_OFF)
         else:
             self.Valve12.SetLabel(valve_status_ON)

 def V13(self, event):
         valve_status_ON = 'ON'
         valve_status_OFF = 'OFF'
         if self.Valve13.GetLabel()==valve_status_ON:
             self.Valve13.SetLabel(valve_status_OFF)
         else:
             self.Valve13.SetLabel(valve_status_ON)

 def V14(self, event):
         valve_status_ON = 'ON'
         valve_status_OFF = 'OFF'
         if self.Valve14.GetLabel()==valve_status_ON:
             self.Valve14.SetLabel(valve_status_OFF)
         else:
             self.Valve14.SetLabel(valve_status_ON)

 def V15(self, event):
         valve_status_ON = 'ON'
         valve_status_OFF = 'OFF'
         if self.Valve15.GetLabel()==valve_status_ON:
             self.Valve15.SetLabel(valve_status_OFF)
         else:
             self.Valve15.SetLabel(valve_status_ON)

def V16(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve16.GetLabel()==valve_status_ON:
            self.Valve16.SetLabel(valve_status_OFF)
        else:
            self.Valve16.SetLabel(valve_status_ON)
```

```
    def V17(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve17.GetLabel()==valve_status_ON:
                self.Valve17.SetLabel(valve_status_OFF)
            else:
                self.Valve17.SetLabel(valve_status_ON)

    def V18(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve18.GetLabel()==valve_status_ON:
                self.Valve18.SetLabel(valve_status_OFF)
            else:
                self.Valve18.SetLabel(valve_status_ON)

    def V19(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve19.GetLabel()==valve_status_ON:
                self.Valve19.SetLabel(valve_status_OFF)
            else:
                self.Valve19.SetLabel(valve_status_ON)

    def V20(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve20.GetLabel()==valve_status_ON:
                self.Valve20.SetLabel(valve_status_OFF)
            else:
                self.Valve20.SetLabel(valve_status_ON)

    def V21(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve21.GetLabel()==valve_status_ON:
                self.Valve21.SetLabel(valve_status_OFF)
            else:
                self.Valve21.SetLabel(valve_status_ON)

    def V22(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve22.GetLabel()==valve_status_ON:
                self.Valve22.SetLabel(valve_status_OFF)
            else:
                self.Valve22.SetLabel(valve_status_ON)

    def V23(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve23.GetLabel()==valve_status_ON:
                self.Valve23.SetLabel(valve_status_OFF)
            else:
                self.Valve23.SetLabel(valve_status_ON)

    def V24(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve24.GetLabel()==valve_status_ON:
                self.Valve24.SetLabel(valve_status_OFF)
            else:
                self.Valve24.SetLabel(valve_status_ON)

    def V25(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve25.GetLabel()==valve_status_ON:
                self.Valve25.SetLabel(valve_status_OFF)
            else:
                self.Valve25.SetLabel(valve_status_ON)
```

```
def V26(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve26.GetLabel()==valve_status_ON:
            self.Valve26.SetLabel(valve_status_OFF)
        else:
            self.Valve26.SetLabel(valve_status_ON)

def V27(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve27.GetLabel()==valve_status_ON:
            self.Valve27.SetLabel(valve_status_OFF)
        else:
            self.Valve27.SetLabel(valve_status_ON)

def V28(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve28.GetLabel()==valve_status_ON:
            self.Valve28.SetLabel(valve_status_OFF)
        else:
            self.Valve28.SetLabel(valve_status_ON)

def V29(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve29.GetLabel()==valve_status_ON:
            self.Valve29.SetLabel(valve_status_OFF)
        else:
            self.Valve29.SetLabel(valve_status_ON)
def V30(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve30.GetLabel()==valve_status_ON:
            self.Valve30.SetLabel(valve_status_OFF)
        else:
            self.Valve30.SetLabel(valve_status_ON)

def V31(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve31.GetLabel()==valve_status_ON:
            self.Valve31.SetLabel(valve_status_OFF)
        else:
            self.Valve31.SetLabel(valve_status_ON)

def V32(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve32.GetLabel()==valve_status_ON:
            self.Valve32.SetLabel(valve_status_OFF)
        else:
            self.Valve32.SetLabel(valve_status_ON)

def V33(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve33.GetLabel()==valve_status_ON:
            self.Valve33.SetLabel(valve_status_OFF)
        else:
            self.Valve33.SetLabel(valve_status_ON)

def V34(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve34.GetLabel()==valve_status_ON:
            self.Valve34.SetLabel(valve_status_OFF)

        else:
            self.Valve34.SetLabel(valve_status_ON)
```

```
def V35(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve35.GetLabel()==valve_status_ON:
            self.Valve35.SetLabel(valve_status_OFF)
        else:
            self.Valve35.SetLabel(valve_status_ON)

def V36(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve36.GetLabel()==valve_status_ON:
            self.Valve36.SetLabel(valve_status_OFF)
        else:
            self.Valve36.SetLabel(valve_status_ON)

def V37(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve37.GetLabel()==valve_status_ON:
            self.Valve37.SetLabel(valve_status_OFF)
        else:
            self.Valve37.SetLabel(valve_status_ON)

def V38(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve38.GetLabel()==valve_status_ON:
            self.Valve38.SetLabel(valve_status_OFF)
        else:
            self.Valve38.SetLabel(valve_status_ON)

def V39(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve39.GetLabel()==valve_status_ON:
            self.Valve39.SetLabel(valve_status_OFF)
        else:
            self.Valve39.SetLabel(valve_status_ON)

def V40(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve40.GetLabel()==valve_status_ON:
            self.Valve40.SetLabel(valve_status_OFF)
        else:
            self.Valve40.SetLabel(valve_status_ON)

def V41(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve41.GetLabel()==valve_status_ON:
            self.Valve41.SetLabel(valve_status_OFF)
        else:
            self.Valve41.SetLabel(valve_status_ON)

def V42(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve42.GetLabel()==valve_status_ON:
            self.Valve42.SetLabel(valve_status_OFF)
        else:
            self.Valve42.SetLabel(valve_status_ON)

def V43(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve43.GetLabel()==valve_status_ON:
            self.Valve43.SetLabel(valve_status_OFF)

        else:
            self.Valve43.SetLabel(valve_status_ON)
```

```
def V44(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve44.GetLabel()==valve_status_ON:
            self.Valve44.SetLabel(valve_status_OFF)
        else:
            self.Valve44.SetLabel(valve_status_ON)

def V45(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve45.GetLabel()==valve_status_ON:
            self.Valve45.SetLabel(valve_status_OFF)
        else:
            self.Valve45.SetLabel(valve_status_ON)

def V46(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve46.GetLabel()==valve_status_ON:
            self.Valve46.SetLabel(valve_status_OFF)
        else:
            self.Valve46.SetLabel(valve_status_ON)

def V47(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve47.GetLabel()==valve_status_ON:
            self.Valve47.SetLabel(valve_status_OFF)
        else:
            self.Valve47.SetLabel(valve_status_ON)

def V48(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve48.GetLabel()==valve_status_ON:
            self.Valve48.SetLabel(valve_status_OFF)
        else:
            self.Valve48.SetLabel(valve_status_ON)

def V49(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve49.GetLabel()==valve_status_ON:
            self.Valve49.SetLabel(valve_status_OFF)
        else:
            self.Valve49.SetLabel(valve_status_ON)

def V50(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve50.GetLabel()==valve_status_ON:
            self.Valve50.SetLabel(valve_status_OFF)
        else:
            self.Valve50.SetLabel(valve_status_ON)

def V51(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
        if self.Valve51.GetLabel()==valve_status_ON:
            self.Valve51.SetLabel(valve_status_OFF)
        else:
            self.Valve51.SetLabel(valve_status_ON)

 def V52(self, event):
         valve_status_ON = 'ON'
         valve_status_OFF = 'OFF'

         if self.Valve52.GetLabel()==valve_status_ON:
             self.Valve52.SetLabel(valve_status_OFF)

         else:
             self.Valve52.SetLabel(valve_status_ON)
```

```
    def V53(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve53.GetLabel()==valve_status_ON:
                self.Valve53.SetLabel(valve_status_OFF)
            else:
                self.Valve53.SetLabel(valve_status_ON)

    def V54(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve54.GetLabel()==valve_status_ON:
                self.Valve54.SetLabel(valve_status_OFF)
            else:
                self.Valve54.SetLabel(valve_status_ON)

    def V55(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve55.GetLabel()==valve_status_ON:
                self.Valve55.SetLabel(valve_status_OFF)
            else:
                self.Valve55.SetLabel(valve_status_ON)

    def V56(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve56.GetLabel()==valve_status_ON:
                self.Valve56.SetLabel(valve_status_OFF)
            else:
                self.Valve56.SetLabel(valve_status_ON)

    def V57(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve57.GetLabel()==valve_status_ON:
                self.Valve57.SetLabel(valve_status_OFF)
            else:
                self.Valve57.SetLabel(valve_status_ON)

    def V58(self, event):
            valve_status_ON = 'ON'
            valve_status_OFF = 'OFF'
            if self.Valve58.GetLabel()==valve_status_ON:
                self.Valve58.SetLabel(valve_status_OFF)
            else:
                self.Valve58.SetLabel(valve_status_ON)

********************                    Define Pumps                    ********************
*************************************************************************************************

    def P1 (self,event):
        wx.MessageBox("P3 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)
    def P2 (self,event):
        wx.MessageBox("P3 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)
    def P3 (self,event):
        wx.MessageBox("P3 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)
    def P4 (self,event):
        wx.MessageBox("P4 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P5 (self,event):
        wx.MessageBox("P5 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P6 (self,event):
        wx.MessageBox("P6 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P7 (self,event):
        wx.MessageBox("P7 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P8 (self,event):
        wx.MessageBox("P8 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P9 (self,event):
        wx.MessageBox("P9 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P10 (self,event):
        wx.MessageBox("P10 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)
```

```
#**********************              main definition and loop              ********************
#*****************************************************************************************************

def main():
    """Testing"""
    MyBackgroundPanel = wx.PySimpleApp(wx.Panel)
    f = MyFrame()
    d=ValveListBox()
    d.Show()
    f.Center()
    f.Show()
    MyBackgroundPanel.MainLoop()

if __name__ == "__main__":

    main()
```

9.4 Annex 2b: phython automation code including full automation

D:\AECENAR\MEGBI\MEGBI-VPP\AUT\CDMasterThesisHayssamHindy\GUI\101015_14MEGBI-VPP_automation.py

```
#-------------------------------------------------------------------------------
# Name:        MEGBI monitoring software
# Purpose:
#
# Author:      abd el rahman, Samir, hayssam
#
# Created:     09/05/2013
# Copyright:   (c) aecenar 2013, 2015
# Licence:     <AECENAR>
#-------------------------------------------------------------------------------

##!/usr/bin/env python
## -*- coding: iso-8859-15 -*-
import sys

#from tkinter import *
import threading
#from PIL import Image, ImageTk
import random

#import tkinter
```

```
import time
#from pymouse import PyMouse

#import ctypes
from ctypes import *

import wx
#import thread



#*********************              panel frame              **********************
#*****************************************************************************************
#wx.SetDefaultPyEncoding("iso-8859-15")
BACKGROUND_IMAGENAME = "220415MEGPI-VPP_PI_DIAGRAM.PNG"
#BACKGROUND_IMAGENAME =\"C:\Users\Hayssam\Desktop\GUI 220415MEGPI-VPP_PI_DIAGRAM.PNG"
##"hintergrundbild.jpg"



class MyBackgroundPanel(wx.Panel):

   def __init__(self, parent):
      wx.Panel.__init__(self, parent)
      self.bmp = wx.Bitmap(BACKGROUND_IMAGENAME)
      self.SetSize(self.bmp.GetSize())
      self.Bind(wx.EVT_PAINT, self.on_paint)


   def on_paint(self, event = None):
      dc = wx.BufferedPaintDC(self, self.bmp)


class MyFrame(wx.Frame):
```

```
    def __init__(self, parent = None, title = "MEGBI Vaccine Pilot Plant (MEGBI-VPP) Overview Upstream
& Downstream Process"):

        self.testUSB              = True
        self.dll                  = None
        self.USBAdr0              = 0
        self.USBAdr1              = 1
        self.USBAdr2              = 2
        self.USBOpened            = False
        self.counterUSBBoards     = 3

        wx.Frame.__init__(self, parent, -1, title)

        panel = MyBackgroundPanel(self)

        LABELSTYLE = wx.BORDER_SUNKEN | wx.ST_NO_AUTORESIZE |
wx.ALIGN_CENTER_HORIZONTAL



        menuFile = wx.Menu()
        menuFile.Append(1, "&About...")
        menuFile.AppendSeparator()
        menuFile.Append(2, "Exit")
        menuBar = wx.MenuBar()
        menuBar.Append(menuFile, "File")
        self.SetMenuBar(menuBar)
        self.CreateStatusBar()
        self.SetStatusText("Welcome to MEGPI Project!")
        self.Bind(wx.EVT_MENU, self.OnAbout, id=1)
        self.Bind(wx.EVT_MENU, self.OnQuit, id=2)
```

```
        #Valves

            #V1
self.button_V_1 = wx.Button(panel, -1, "V1", pos=(95,100),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V1, self.button_V_1)

            #V2
self.button_V_2 = wx.Button(panel, -1, "V2", pos=(150,90),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V2, self.button_V_2)
            #V3
self.button_V_3 = wx.Button(panel, -1, "V3", pos=(150,135),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V3, self.button_V_3)



            #V4
self.button_V_4 = wx.Button(panel, -1, "V4", pos=(150,265),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V4, self.button_V_4)

            #V5
self.button_V_5 = wx.Button(panel, -1, "V5", pos=(210,135),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V5, self.button_V_5)

            #V6
self.button_V_6 = wx.Button(panel, -1, "V6", pos=(215,185),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V6, self.button_V_6)

            #V7
self.button_V_7 = wx.Button(panel, -1, "V7", pos=(215,220),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.V7, self.button_V_7)

            #V8
self.button_V_8 = wx.Button(panel, -1, "V8", pos=(282,180),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V8, self.button_V_8)
```

```
        #V9
self.button_V_9 = wx.Button(panel, -1, "V9", pos=(282,158),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V9, self.button_V_9)

        #V10
self.button_V_10 = wx.Button(panel, -1, "V10", pos=(322,110),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V10, self.button_V_10)

        #V11
self.button_V_11 = wx.Button(panel, -1, "V11", pos=(380,250),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V11, self.button_V_11)

        #V12
self.button_V_12 = wx.Button(panel, -1, "V12", pos=(413,195),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V12, self.button_V_12)

        #V13
self.button_V_13 = wx.Button(panel, -1, "V13", pos=(380,130),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V13, self.button_V_13)

        #V14
self.button_V_14 = wx.Button(panel, -1, "V14", pos=(410,100),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V14, self.button_V_14)

        #V15
self.button_V_15 = wx.Button(panel, -1, "V15", pos=(460,112),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V15, self.button_V_15)

        #V16
self.button_V_16 = wx.Button(panel, -1, "V16", pos=(460,73),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V16, self.button_V_16)

        #V17
self.button_V_17 = wx.Button(panel, -1, "V17", pos=(460,38),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V17, self.button_V_17)
```

```
        #V18
self.button_V_18 = wx.Button(panel, -1, "V18", pos=(522,18),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V18, self.button_V_18)

        #V19
self.button_V_19 = wx.Button(panel, -1, "V19", pos=(635,18),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V19, self.button_V_19)

        #V20
self.button_V_20 = wx.Button(panel, -1, "V20", pos=(635,72),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V20, self.button_V_20)

        #V21
self.button_V_21 = wx.Button(panel, -1, "V21", pos=(675,112),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V21, self.button_V_21)

        #V22
self.button_V_22 = wx.Button(panel, -1, "V22", pos=(635,155),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V22, self.button_V_22)

        #V23
self.button_V_23 = wx.Button(panel, -1, "V23", pos=(780,18),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V23, self.button_V_23)

        #V24
self.button_V_24 = wx.Button(panel, -1, "V24", pos=(770,120),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V24, self.button_V_24)

        #V25
self.button_V_25 = wx.Button(panel, -1, "V25", pos=(770,155),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V25, self.button_V_25)

        #V26
self.button_V_26 = wx.Button(panel, -1, "V26", pos=(870,155),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V26, self.button_V_26)


        #V27
```

```
self.button_V_27 = wx.Button(panel, -1, "V27", pos=(510,185),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V27, self.button_V_27)

        #V28
self.button_V_28 = wx.Button(panel, -1, "V28", pos=(620,185),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V28, self.button_V_28)

        #V29
self.button_V_29 = wx.Button(panel, -1, "V29", pos=(680,185),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V29, self.button_V_29)



        #V30
self.button_V_30 = wx.Button(panel, -1, "V30", pos=(620,240),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V30, self.button_V_30)

        #V31
self.button_V_31 = wx.Button(panel, -1, "V31", pos=(710,235),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V31, self.button_V_31)

        #V32
self.button_V_32 = wx.Button(panel, -1, "V32", pos=(950,182),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V32, self.button_V_32)

        #V33
self.button_V_33 = wx.Button(panel, -1, "V33", pos=(977,182),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V33, self.button_V_33)



        #V34
self.button_V_34 = wx.Button(panel, -1, "V34", pos=(1083,188),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V34, self.button_V_34)

        #V35
self.button_V_35 = wx.Button(panel, -1, "V35", pos=(1083,220),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V35, self.button_V_35)
```

```
        #V36
self.button_V_36 = wx.Button(panel, -1, "V36", pos=(1170,110),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V36, self.button_V_36)

        #V37
self.button_V_37 = wx.Button(panel, -1, "V37", pos=(1200,110),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V37, self.button_V_37)

        #V38
self.button_V_38 = wx.Button(panel, -1, "V38", pos=(1240,150),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V38, self.button_V_38)

        #V39
self.button_V_39 = wx.Button(panel, -1, "V39", pos=(1273,205),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V39, self.button_V_39)



        #V40
self.button_V_40 = wx.Button(panel, -1, "V40", pos=(483,314),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V40, self.button_V_40)

        #V41
self.button_V_41 = wx.Button(panel, -1, "V41", pos=(478,398),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V41, self.button_V_41)

        #V42
self.button_V_42 = wx.Button(panel, -1, "V42", pos=(540,314),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V42, self.button_V_42)

        #V43
self.button_V_43 = wx.Button(panel, -1, "V43", pos=(548,398),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V43, self.button_V_43)


        #V44
self.button_V_44 = wx.Button(panel, -1, "V44", pos=(580,300),size=(25,20))
```

```
self.Bind(wx.EVT_BUTTON, self.V44, self.button_V_44)

        #V45
self.button_V_45 = wx.Button(panel, -1, "V45", pos=(620,314),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V45, self.button_V_45)



        #V46
self.button_V_46 = wx.Button(panel, -1, "V46", pos=(760,440),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V46, self.button_V_46)

        #V47
self.button_V_47 = wx.Button(panel, -1, "V47", pos=(760,400),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V47, self.button_V_47)

        #V48
self.button_V_48 = wx.Button(panel, -1, "V48", pos=(783,375),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V48, self.button_V_48)

        #V49
self.button_V_49 = wx.Button(panel, -1, "V49", pos=(880,428),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V49, self.button_V_49)



        #V50
self.button_V_50 = wx.Button(panel, -1, "V50", pos=(970,377),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V50, self.button_V_50)

        #V51
self.button_V_51 = wx.Button(panel, -1, "V51", pos=(963,450),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V51, self.button_V_51)

        #V52
self.button_V_52 = wx.Button(panel, -1, "V52", pos=(1005,470),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V52, self.button_V_52)

        #V53
```

```
self.button_V_53 = wx.Button(panel, -1, "V53", pos=(1030,395),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V53, self.button_V_53)

        #V54
self.button_V_54 = wx.Button(panel, -1, "V54", pos=(1080,395),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V54, self.button_V_54)
        #V55
self.button_V_55 = wx.Button(panel, -1, "V55", pos=(1118,377),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V55, self.button_V_55)

        #V56
self.button_V_56 = wx.Button(panel, -1, "V56", pos=(1158,468),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V56, self.button_V_56)

        #V57
self.button_V_57 = wx.Button(panel, -1, "V57", pos=(1190,420),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V57, self.button_V_57)



        #V58
self.button_V_58 = wx.Button(panel, -1, "V58", pos=(710,580),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.V58, self.button_V_58)



    #Pumps

        #P1
self.button_P_1 = wx.Button(panel, -1, "P1", pos=(150,180),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P1, self.button_P_1)

        #P2
self.button_P_2 = wx.Button(panel, -1, "P2", pos=(150,220),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P2, self.button_P_2)

        #P3
self.button_P_3 = wx.Button(panel, -1, "P3", pos=(510,40),size=(20,20))
```

```
self.Bind(wx.EVT_BUTTON, self.P3, self.button_P_3)

        #P4
self.button_P_4 = wx.Button(panel, -1, "P4", pos=(880,18),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P4, self.button_P_4)

        #P5
self.button_P_5 = wx.Button(panel, -1, "P5", pos=(730,187),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P5, self.button_P_5)

        #P6
self.button_P_6 = wx.Button(panel, -1, "P6", pos=(1083,130),size=(20,20))
self.Bind(wx.EVT_BUTTON, self.P6, self.button_P_6)

        #P7
self.button_P_7 = wx.Button(panel, -1, "P7", pos=(380,328),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P7, self.button_P_7)

        #P8
self.button_P_8 = wx.Button(panel, -1, "P8", pos=(710,423),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P8, self.button_P_8)

        #P9
self.button_P_9 = wx.Button(panel, -1, "P9", pos=(900,380),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P9, self.button_P_9)

        #P10
self.button_P_10 = wx.Button(panel, -1, "P10", pos=(1200,380),size=(25,20))
self.Bind(wx.EVT_BUTTON, self.P10, self.button_P_10)

    #control buttons

            #Turn ON button
self.button_Turn_ON = wx.Button(panel, -1, "Turn On", pos=(40,40),size=(60,20))
self.Bind(wx.EVT_BUTTON, self.MEGBIVPPFullAutomationStart, self.button_Turn_ON)
self.ON = wx.StaticText(panel, size = (40, -1), pos = (140, 40), style = LABELSTYLE)
```

```
    #Text column 1
wx.StaticText(panel,20," Valve Selected : ",(10,270))
wx.StaticText(panel,-1," V1 ",(18,300))
wx.StaticText(panel,-1," V2 ",(18,326))
wx.StaticText(panel,-1," V3 ",(18,352))
wx.StaticText(panel,-1," V4 ",(18,378))
wx.StaticText(panel,-1," V5 ",(18,402))
wx.StaticText(panel,-1," V6 ",(18,428))
wx.StaticText(panel,-1," V7 ",(18,454))
wx.StaticText(panel,-1," V8 ",(18,480))
wx.StaticText(panel,-1," V9 ",(18,506))
wx.StaticText(panel,-1," V10 ",(18,532))
wx.StaticText(panel,-1," V11 ",(18,558))
wx.StaticText(panel,-1," V12 ",(18,584))

    #Text column 2
wx.StaticText(panel,-1," V13 ",(85,300))
wx.StaticText(panel,-1," V14 ",(85,326))
wx.StaticText(panel,-1," V15 ",(85,352))
wx.StaticText(panel,-1," V16 ",(85,378))
wx.StaticText(panel,-1," V17 ",(85,402))
wx.StaticText(panel,-1," V18 ",(85,428))
wx.StaticText(panel,-1," V19 ",(85,454))
wx.StaticText(panel,-1," V20 ",(85,480))
wx.StaticText(panel,-1," V21 ",(85,506))
wx.StaticText(panel,-1," V22 ",(85,532))
wx.StaticText(panel,-1," V23 ",(85,558))
wx.StaticText(panel,-1," V24 ",(85,584))

 #Text column 3
wx.StaticText(panel,-1," V25 ",(155,300))
wx.StaticText(panel,-1," V26 ",(155,326))
wx.StaticText(panel,-1," V27 ",(155,352))
wx.StaticText(panel,-1," V28 ",(155,378))
wx.StaticText(panel,-1," V29 ",(155,402))
wx.StaticText(panel,-1," V30 ",(155,428))
```

```
wx.StaticText(panel,-1," V31 ",(155,454))
wx.StaticText(panel,-1," V32 ",(155,480))
wx.StaticText(panel,-1," V33 ",(155,506))
wx.StaticText(panel,-1," V34 ",(155,532))
wx.StaticText(panel,-1," V35 ",(155,558))
wx.StaticText(panel,-1," V36 ",(155,584))

#Text column 4
wx.StaticText(panel,-1," V37 ",(223,300))
wx.StaticText(panel,-1," V38 ",(223,326))
wx.StaticText(panel,-1," V39 ",(223,352))
wx.StaticText(panel,-1," V40 ",(223,378))
wx.StaticText(panel,-1," V41 ",(223,402))
wx.StaticText(panel,-1," V42 ",(223,428))
wx.StaticText(panel,-1," V43 ",(223,454))
wx.StaticText(panel,-1," V44 ",(223,480))
wx.StaticText(panel,-1," V45 ",(223,506))
wx.StaticText(panel,-1," V46 ",(223,532))
wx.StaticText(panel,-1," V47 ",(223,558))
wx.StaticText(panel,-1," V48 ",(223,584))

#Text column 5
wx.StaticText(panel,-1," V49 ",(293,300))
wx.StaticText(panel,-1," V50 ",(293,326))
wx.StaticText(panel,-1," V51 ",(293,352))
wx.StaticText(panel,-1," V52 ",(293,378))
wx.StaticText(panel,-1," V53 ",(293,402))
wx.StaticText(panel,-1," V54 ",(293,428))
wx.StaticText(panel,-1," V55 ",(293,454))
wx.StaticText(panel,-1," V56 ",(293,480))
wx.StaticText(panel,-1," V57 ",(293,506))
wx.StaticText(panel,-1," V58 ",(293,532))
#wx.StaticText(panel,-1," V59 ",(293,558))
# wx.StaticText(panel,-1," V60 ",(293,584))
```

```
    #Label  Column 1
self.Valve1 = wx.StaticText(panel, size = (26, -1), pos = (45, 300), style = LABELSTYLE)
self.Valve2 = wx.StaticText(panel, size = (26, -1), pos = (45, 326), style = LABELSTYLE)
self.Valve3 = wx.StaticText(panel, size = (26, -1), pos = (45, 352), style = LABELSTYLE)
self.Valve4 = wx.StaticText(panel, size = (26, -1), pos = (45, 378), style = LABELSTYLE)
self.Valve5 = wx.StaticText(panel, size = (26, -1), pos = (45, 402), style = LABELSTYLE)
self.Valve6 = wx.StaticText(panel, size = (26, -1), pos = (45, 428), style = LABELSTYLE)
self.Valve7 = wx.StaticText(panel, size = (26, -1), pos = (45, 454), style = LABELSTYLE)
self.Valve8 = wx.StaticText(panel, size = (26, -1), pos = (45, 480), style = LABELSTYLE)
self.Valve9 = wx.StaticText(panel, size = (26, -1), pos = (45, 506), style = LABELSTYLE)
self.Valve10 = wx.StaticText(panel, size = (26, -1), pos = (45, 532), style = LABELSTYLE)
self.Valve11 = wx.StaticText(panel, size = (26, -1), pos = (45, 558), style = LABELSTYLE)
self.Valve12 = wx.StaticText(panel, size = (26, -1), pos = (45, 584), style = LABELSTYLE)



    #Label column 2
self.Valve13 = wx.StaticText(panel, size = (26, -1), pos = (115, 300), style = LABELSTYLE)
self.Valve14 = wx.StaticText(panel, size = (26, -1), pos = (115, 326), style = LABELSTYLE)
self.Valve15 = wx.StaticText(panel, size = (26, -1), pos = (115, 352), style = LABELSTYLE)
self.Valve16 = wx.StaticText(panel, size = (26, -1), pos = (115, 378), style = LABELSTYLE)
self.Valve17 = wx.StaticText(panel, size = (26, -1), pos = (115, 402), style = LABELSTYLE)
self.Valve18 = wx.StaticText(panel, size = (26, -1), pos = (115, 428), style = LABELSTYLE)
self.Valve19 = wx.StaticText(panel, size = (26, -1), pos = (115, 454), style = LABELSTYLE)
self.Valve20 = wx.StaticText(panel, size = (26, -1), pos = (115, 480), style = LABELSTYLE)
self.Valve21 = wx.StaticText(panel, size = (26, -1), pos = (115, 506), style = LABELSTYLE)
self.Valve22 = wx.StaticText(panel, size = (26, -1), pos = (115, 532), style = LABELSTYLE)
self.Valve23 = wx.StaticText(panel, size = (26, -1), pos = (115, 558), style = LABELSTYLE)
self.Valve24 = wx.StaticText(panel, size = (26, -1), pos = (115, 584), style = LABELSTYLE)

 #Label column 3
```

```
self.Valve25 = wx.StaticText(panel, size = (26, -1), pos = (185, 300), style = LABELSTYLE)
self.Valve26 = wx.StaticText(panel, size = (26, -1), pos = (185, 326), style = LABELSTYLE)
self.Valve27 = wx.StaticText(panel, size = (26, -1), pos = (185, 352), style = LABELSTYLE)
self.Valve28 = wx.StaticText(panel, size = (26, -1), pos = (185, 378), style = LABELSTYLE)
self.Valve29 = wx.StaticText(panel, size = (26, -1), pos = (185, 402), style = LABELSTYLE)
self.Valve30 = wx.StaticText(panel, size = (26, -1), pos = (185, 428), style = LABELSTYLE)
self.Valve31 = wx.StaticText(panel, size = (26, -1), pos = (185, 454), style = LABELSTYLE)
self.Valve32 = wx.StaticText(panel, size = (26, -1), pos = (185, 480), style = LABELSTYLE)
self.Valve33 = wx.StaticText(panel, size = (26, -1), pos = (185, 506), style = LABELSTYLE)
self.Valve34 = wx.StaticText(panel, size = (26, -1), pos = (185, 532), style = LABELSTYLE)
self.Valve35 = wx.StaticText(panel, size = (26, -1), pos = (185, 558), style = LABELSTYLE)
self.Valve36 = wx.StaticText(panel, size = (26, -1), pos = (185, 584), style = LABELSTYLE)

#Label column 4

self.Valve37 = wx.StaticText(panel, size = (26, -1), pos = (253, 300), style = LABELSTYLE)
self.Valve38 = wx.StaticText(panel, size = (26, -1), pos = (253, 326), style = LABELSTYLE)
self.Valve39 = wx.StaticText(panel, size = (26, -1), pos = (253, 352), style = LABELSTYLE)
self.Valve40 = wx.StaticText(panel, size = (26, -1), pos = (253, 378), style = LABELSTYLE)
self.Valve41 = wx.StaticText(panel, size = (26, -1), pos = (253, 402), style = LABELSTYLE)
self.Valve42 = wx.StaticText(panel, size = (26, -1), pos = (253, 428), style = LABELSTYLE)
self.Valve43 = wx.StaticText(panel, size = (26, -1), pos = (253, 454), style = LABELSTYLE)
self.Valve44 = wx.StaticText(panel, size = (26, -1), pos = (253, 480), style = LABELSTYLE)
self.Valve45 = wx.StaticText(panel, size = (26, -1), pos = (253, 506), style = LABELSTYLE)
self.Valve46 = wx.StaticText(panel, size = (26, -1), pos = (253, 532), style = LABELSTYLE)
self.Valve47 = wx.StaticText(panel, size = (26, -1), pos = (253, 558), style = LABELSTYLE)
self.Valve48 = wx.StaticText(panel, size = (26, -1), pos = (253, 584), style = LABELSTYLE)

#Label column 5

self.Valve49 = wx.StaticText(panel, size = (26, -1), pos = (323, 300), style = LABELSTYLE)
self.Valve50 = wx.StaticText(panel, size = (26, -1), pos = (323, 326), style = LABELSTYLE)
self.Valve51 = wx.StaticText(panel, size = (26, -1), pos = (323, 352), style = LABELSTYLE)
self.Valve52 = wx.StaticText(panel, size = (26, -1), pos = (323, 378), style = LABELSTYLE)
self.Valve53 = wx.StaticText(panel, size = (26, -1), pos = (323, 402), style = LABELSTYLE)
self.Valve54 = wx.StaticText(panel, size = (26, -1), pos = (323, 428), style = LABELSTYLE)
```

```
        self.Valve55 = wx.StaticText(panel, size = (26, -1), pos = (323, 454), style = LABELSTYLE)
        self.Valve56 = wx.StaticText(panel, size = (26, -1), pos = (323, 480), style = LABELSTYLE)
        self.Valve57 = wx.StaticText(panel, size = (26, -1), pos = (323, 506), style = LABELSTYLE)
        self.Valve58 = wx.StaticText(panel, size = (26, -1), pos = (323, 532), style = LABELSTYLE)
        #self.Valve59 = wx.StaticText(panel, size = (26, -1), pos = (323, 558), style = LABELSTYLE)

######temperature sensors
        wx.StaticText(panel,-1," Temperature : ",(1100,592))
        self.temp = wx.StaticText(panel, size = (40, -1), pos = (1200, 590), style = LABELSTYLE)









#**********************          Run USB System          **********************
#*****************************************************************************************************
    def OpenUSBBoardThread(self):
        self.dll = windll.K8061
        i = self.counterUSBBoards
        for doit in range(0,i+1):
            try:
                self.dll.OpenDevice()
                self.USBOpened = True
# debug info
                print ('USB Board is now connected!')
#end debug info
            except:
                txt = ('Please Check USB Board connection')
                print ('txt')
                return
```

```
#####################################################

   def OnAbout(self, event):
      wx.MessageBox("This is a Screen controller of MEGPI Project","Welcome to my python", wx.OK |
wx.ICON_INFORMATION, self)
   def OnQuit(self, event):
      self.Close()
   def on_timer(self, event = None):
      self.Close()


   def Sensors(self,event):
      answer = self.dll.ReadAnalogChannel(0,1)
      new_value = str(answer)
      self.temp.SetLabel(answer)
      #power for the temperature sensor (255 >>>> 5V power)
      self.dll.OutputAnalogChannel(0,6,255)

#*********************          Define Valvue          ********************
#*************************************************************************************************
   def MEGBIVPPFullAutomationStart (self, event):


      valve_status_ON = 'ON'
      valve_status_OFF = 'OFF'




      if self.ON.GetLabel()==valve_status_OFF:
         self.ON.SetLabel(valve_status_ON)
         self.OpenUSBBoardThread()
         self.timer = wx.Timer()
         self.timer.Bind(wx.EVT_TIMER, self.on_timer)
         self.timer.Start(100)
         print'system is turning on'
         self.dll.SetDigitalChannel(0,1)
         print'V1 is open'
         self.Valve1.SetLabel(valve_status_ON)
         time.sleep(5)
         self.dll.SetDigitalChannel(0,2)
```

```
print'V2 is open'
self.Valve2.SetLabel(valve_status_ON)
self.timer.Start(40)
time.sleep(20)
self.dll.ClearDigitalChannel(0,1)
self.dll.ClearDigitalChannel(0,2)
print'V1 is close'
self.Valve1.SetLabel(valve_status_OFF)
print'V2 is close'
self.Valve2.SetLabel(valve_status_OFF)
self.dll.SetDigitalChannel(0,3)
print'V3 is open'
self.Valve3.SetLabel(valve_status_ON)
self.timer.Start(40)
time.sleep(20)
self.dll.SetDigitalChannel(0,4)
self.dll.ClearDigitalChannel(0,3)
self.Valve3.SetLabel(valve_status_OFF)
print'V3 is close'
print'V4 is open'
self.Valve4.SetLabel(valve_status_ON)
#self.timer.Start(30)
#time.sleep(20)
self.dll.SetDigitalChannel(0,5)
self.dll.ClearDigitalChannel(0,4)
print'V4 i close'

self.Valve4.SetLabel(valve_status_OFF)
print'V5 is open'
self.Valve5.SetLabel(valve_status_ON)
self.dll.SetDigitalChannel(0,6)
print'P1 is turn on'
#self.timer.Start(400)
time.sleep(20)
self.dll.ClearDigitalChannel(0,5)
self.dll.ClearDigitalChannel(0,6)
print'V5 is close'
self.Valve5.SetLabel(valve_status_OFF)
```

```
print'P1 is turn off'
self.dll.SetDigitalChannel(0,7)
print'V6 is open'
self.Valve6.SetLabel(valve_status_ON)
#self.timer.Start(40)
time.sleep(5)
self.dll.ClearDigitalChannel(0,7)
print'V6 is close'
self.Valve6.SetLabel(valve_status_OFF)
self.dll.SetDigitalChannel(0,8)
print'V7 is open'
self.Valve7.SetLabel(valve_status_ON)
#self.timer.Start(400)
time.sleep(20)
self.dll.OutputAnalogChannel(0,2,255)
print'V8 is open'
self.Valve8.SetLabel(valve_status_ON)
self.dll.ClearDigitalChannel(0,8)
print'V7 is close'
self.Valve7.SetLabel(valve_status_OFF)
#self.timer.Start(40)
time.sleep(5)
self.dll.OutputAnalogChannel(0,2,0)
print'V8 is close'
self.Valve8.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(0,3,255)
print'V9 is open'
self.Valve9.SetLabel(valve_status_ON)
self.dll.OutputAnalogChannel(0,4,255)
print'P2 is turn on'
#self.timer.Start(400)
time.sleep(20)
self.dll.OutputAnalogChannel(0,3,0)
self.dll.OutputAnalogChannel(0,4,0)
self.dll.OutputAnalogChannel(0,5,255)
print'V9 is close'
self.Valve9.SetLabel(valve_status_OFF)
print'P2 is turn off'
```

```
print'V10 is open'
self.Valve10.SetLabel(valve_status_ON)
#self.timer.Start(40)
time.sleep(5)
self.dll.OutputAnalogChannel(0,5,0)
print'V10 is close'
self.Valve10.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(0,1,255)
print'V11 is open'
self.Valve11.SetLabel(valve_status_ON)
self.timer.Start(400)
time.sleep(20)
self.dll.OutputAnalogChannel(0,1,0)
self.dll.OutputAnalogChannel(0,7,255)
print'V11 is close'
self.Valve11.SetLabel(valve_status_OFF)
print'V12 is open'
self.Valve12.SetLabel(valve_status_ON)
#self.timer.Start(40)
time.sleep(5)
self.dll.OutputAnalogChannel(0,7,0)
print'V12 is close'
self.Valve12.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(0,8,255)
print'V13 is open'
self.Valve13.SetLabel(valve_status_ON)
#self.timer.Start(400)
time.sleep(20)
self.dll.OutputAnalogChannel(0,8,0)
self.dll.SetDigitalChannel(1,1)
print'V13 is close'
self.Valve13.SetLabel(valve_status_OFF)
print'V14 is open'
self.Valve14.SetLabel(valve_status_ON)
self.dll.SetDigitalChannel(1,2)
print'P3 is turn on'
#self.timer.Start(40)
time.sleep(5)
```

```
self.dll.ClearDigitalChannel(1,1)
self.dll.ClearDigitalChannel(1,2)
print'V14 is close'
self.Valve14.SetLabel(valve_status_OFF)
print'P3 is turn on'
self.dll.SetDigitalChannel(1,3)
print'V15 is open'
self.Valve15.SetLabel(valve_status_ON)
#self.timer.Start(50)
time.sleep(5)
self.dll.ClearDigitalChannel(1,3)
print'V15 is close'
self.Valve15.SetLabel(valve_status_OFF)
self.timer.Start(100)
self.dll.SetDigitalChannel(1,4)
print'V16 is open'
self.Valve16.SetLabel(valve_status_ON)
#self.timer.Start(20)
time.sleep(10)
self.dll.ClearDigitalChannel(1,4)
self.dll.SetDigitalChannel(1,5)
self.dll.SetDigitalChannel(1,6)
print'V16 is close'
self.Valve16.SetLabel(valve_status_OFF)
print'V17 is open'
self.Valve17.SetLabel(valve_status_ON)
print'V18 is open'
self.Valve18.SetLabel(valve_status_ON)
#self.timer.Start(500)
time.sleep(20)
self.dll.ClearDigitalChannel(1,5)
self.dll.ClearDigitalChannel(1,6)
print'V17 is close'
self.Valve17.SetLabel(valve_status_OFF)
print'V18 is close'
self.Valve18.SetLabel(valve_status_OFF)
self.dll.SetDigitalChannel(1,7)
print'V19 is open'
```

```
self.Valve19.SetLabel(valve_status_ON)
self.dll.SetDigitalChannel(1,8)
print'P4 is turn on'
#self.timer.Start(50)
time.sleep(10)
self.dll.ClearDigitalChannel(1,7)
print'V19 is close'
self.Valve19.SetLabel(valve_status_OFF)
self.dll.ClearDigitalChannel(1,8)
print'P4 is turn off'
self.dll.OutputAnalogChannel(1,1,255)
print'V20 is open'
self.Valve20.SetLabel(valve_status_ON)
#self.timer.Start(50)
time.sleep(5)
self.dll.OutputAnalogChannel(1,1,0)
print'V20 is close'
self.Valve20.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(1,2,255)
self.dll.OutputAnalogChannel(1,3,255)
print'V21 is open'
self.Valve21.SetLabel(valve_status_ON)
print'V22 is open'
self.Valve22.SetLabel(valve_status_ON)
#self.timer.Start(500)
time.sleep(20)
self.dll.OutputAnalogChannel(1,2,0)
self.dll.OutputAnalogChannel(1,3,0)
print'V21 is close'
self.Valve21.SetLabel(valve_status_OFF)
print'V22 is close'
self.Valve22.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(1,4,255)
self.dll.OutputAnalogChannel(1,5,255)
print'V23 is open'
self.Valve23.SetLabel(valve_status_ON)
print'P5 is turn On'
#self.timer.Start(40)
```

```
time.sleep(5)
self.dll.OutputAnalogChannel(1,4,0)
self.dll.OutputAnalogChannel(1,5,0)
print'V23 is close'
self.Valve23.SetLabel(valve_status_OFF)
print'P5 is turn off'
self.dll.OutputAnalogChannel(1,6,255)
print'V24 is open'
self.Valve24.SetLabel(valve_status_ON)
#self.timer.Start(500)
time.sleep(20)
self.dll.OutputAnalogChannel(1,6,0)
print'V24 is close'
self.Valve24.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(1,7,255)
print'V25 is open'
self.Valve25.SetLabel(valve_status_ON)
#self.timer.Start(50)
time.sleep(5)
self.dll.OutputAnalogChannel(1,7,0)
print'V25 is close'
self.Valve25.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(1,8,255)
print'V26 is open'
self.Valve26.SetLabel(valve_status_ON)
#self.timer.Start(600)
time.sleep(20)
self.dll.OutputAnalogChannel(1,8,0)
print'V26 is close'
self.Valve26.SetLabel(valve_status_OFF)
self.dll.SetDigitalChannel(2,1)
print'V27 is open'
self.Valve27.SetLabel(valve_status_ON)
#self.timer.Start(40)
time.sleep(5)
self.dll.ClearDigitalChannel(2,1)
print'V27 is close'
self.Valve27.SetLabel(valve_status_OFF)
```

```
self.dll.SetDigitalChannel(2,2)
print'V28 is open'
self.Valve28.SetLabel(valve_status_ON)
#self.timer.Start(60)
time.sleep(10)
self.dll.SetDigitalChannel(2,3)
self.dll.SetDigitalChannel(2,4)
print'V29 is open'
self.Valve29.SetLabel(valve_status_ON)
print'V30 is open'
self.Valve30.SetLabel(valve_status_ON)
self.dll.ClearDigitalChannel(2,2)
print'V28 is close'
self.Valve28.SetLabel(valve_status_OFF)
#self.timer.Start(40)
time.sleep(5)
self.dll.ClearDigitalChannel(2,3)
self.dll.ClearDigitalChannel(2,4)
print'V29 is close'
self.Valve29.SetLabel(valve_status_OFF)
print'V30 is close'
self.Valve30.SetLabel(valve_status_OFF)
#self.timer.Start(600)
time.sleep(20)
self.dll.SetDigitalChannel(2,5)
print'V31 is open'
self.Valve31.SetLabel(valve_status_ON)
self.dll.SetDigitalChannel(2,6)
print'P6 is turn on'
#self.timer.Start(50)
time.sleep(5)
self.dll.ClearDigitalChannel(2,5)
self.dll.ClearDigitalChannel(2,6)
print'V31 is close'
self.Valve31.SetLabel(valve_status_OFF)
print'P6 is turn off'
#self.timer.Start(40)
time.sleep(5)
```

```
self.dll.SetDigitalChannel(2,7)
print'V32 is open'
self.Valve32.SetLabel(valve_status_ON)
#self.timer.Start(600)
time.sleep(20)
print'V32 is close'
self.Valve32.SetLabel(valve_status_OFF)
self.dll.ClearDigitalChannel(2,7)
self.dll.SetDigitalChannel(2,8)
print'V33 is open'
self.Valve33.SetLabel(valve_status_ON)
#self.timer.Start(50)
time.sleep(5)
self.dll.ClearDigitalChannel(2,8)
print'V33 is close'
self.Valve33.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(2,1,255)
print'V34 is open'
self.Valve34.SetLabel(valve_status_ON)
#self.timer.Start(50)
time.sleep(5)
self.dll.OutputAnalogChannel(2,1,0)
print'V34 is close'
self.Valve34.SetLabel(valve_status_OFF)
self.dll.OutputAnalogChannel(2,2,255)

self.dll.OutputAnalogChannel(2,3,255)
print'V35 is open'
self.Valve35.SetLabel(valve_status_ON)
print'P7 is turn on'
#self.timer.Start(50)
time.sleep(5)
self.dll.OutputAnalogChannel(2,2,0)
self.dll.OutputAnalogChannel(2,3,0)
print'V35 is close'
print'P7 is turn off'
print'System is turning off'
```

```
    else:
        self.ON.SetLabel(valve_status_OFF)
        wx.MessageBox("Turning OFF ","System", wx.OK|wx.ICON_INFORMATION)









        #self.dll.OutputAnalogChannel(3,8,255)
        #
        #def V1(self,event):
         #   valve_status_ON = 'ON'
          #  valve_status_OFF = 'OFF'

           # self.OpenUSBBoardThread()
            #time.sleep(0.5)
            #self.dll.SetDigitalChannel(0,1)
            #self.timer = wx.Timer()
            #print'V1 is open'
            #self.V1.Open()
            #self.timer.Bind(wx.EVT_TIMER, self.on_timer)
            #self.timer.Start(120)
        #def V2 (self,event):
         #   self.OpenUSBBoardThread()
          #  self.dll.SetDigitalChannel(0,2)
           # self.Valve1.SetLabel(valve_status_ON)


    #else:
        #self.ON.SetLabel(valve_status_ON)
        #wx.MessageBox("Turning ON ","System", wx.OK|wx.ICON_INFORMATION)



def V1(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
```

```
        if self.Valve1.GetLabel()==valve_status_ON:
            self.Valve1.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve1.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)




def V2(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve2.GetLabel()==valve_status_ON:
            self.Valve2.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve2.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

def V3(self, event):

        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve3.GetLabel()==valve_status_ON:
            self.Valve3.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve3.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

def V4(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
```

```
        if self.Valve4.GetLabel()==valve_status_ON:
            self.Valve4.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
           self.Valve4.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

    def V5(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve5.GetLabel()==valve_status_ON:
            self.Valve5.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve5.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

    def V6(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve6.GetLabel()==valve_status_ON:
            self.Valve6.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve6.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

    def V7(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve7.GetLabel()==valve_status_ON:
            self.Valve7.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
```

```
            self.Valve7.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

    def V8(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve8.GetLabel()==valve_status_ON:
            self.Valve8.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve8.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

    def V9(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve9.GetLabel()==valve_status_ON:
            self.Valve9.SetLabel(valve_status_OFF)
           # wx.MessageBox("V1 is Open ","Open", wx.OK|wx.ICON_INFORMATION)
        else:
            self.Valve9.SetLabel(valve_status_ON)
            #wx.MessageBox("V1 is Close","Closed", wx.OK|wx.ICON_INFORMATION)

    def V10(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve10.GetLabel()==valve_status_ON:
            self.Valve10.SetLabel(valve_status_OFF)

        else:
            self.Valve10.SetLabel(valve_status_ON)



    def V11(self, event):
        valve_status_ON = 'ON'
```

```
        valve_status_OFF = 'OFF'

        if self.Valve11.GetLabel()==valve_status_ON:
            self.Valve11.SetLabel(valve_status_OFF)

        else:
            self.Valve11.SetLabel(valve_status_ON)

    def V12(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve12.GetLabel()==valve_status_ON:
            self.Valve12.SetLabel(valve_status_OFF)

        else:
            self.Valve12.SetLabel(valve_status_ON)

    def V13(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve13.GetLabel()==valve_status_ON:
            self.Valve13.SetLabel(valve_status_OFF)

        else:
            self.Valve13.SetLabel(valve_status_ON)

    def V14(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve14.GetLabel()==valve_status_ON:
            self.Valve14.SetLabel(valve_status_OFF)

        else:
            self.Valve14.SetLabel(valve_status_ON)
```

```
def V15(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve15.GetLabel()==valve_status_ON:
            self.Valve15.SetLabel(valve_status_OFF)

        else:
            self.Valve15.SetLabel(valve_status_ON)

def V16(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve16.GetLabel()==valve_status_ON:
            self.Valve16.SetLabel(valve_status_OFF)

        else:
            self.Valve16.SetLabel(valve_status_ON)
def V17(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve17.GetLabel()==valve_status_ON:
            self.Valve17.SetLabel(valve_status_OFF)

        else:
            self.Valve17.SetLabel(valve_status_ON)

def V18(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve18.GetLabel()==valve_status_ON:
            self.Valve18.SetLabel(valve_status_OFF)

        else:
            self.Valve18.SetLabel(valve_status_ON)
```

```
def V19(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve19.GetLabel()==valve_status_ON:
            self.Valve19.SetLabel(valve_status_OFF)

        else:
            self.Valve19.SetLabel(valve_status_ON)

def V20(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve20.GetLabel()==valve_status_ON:
            self.Valve20.SetLabel(valve_status_OFF)

        else:
            self.Valve20.SetLabel(valve_status_ON)

def V21(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve21.GetLabel()==valve_status_ON:
            self.Valve21.SetLabel(valve_status_OFF)

        else:
            self.Valve21.SetLabel(valve_status_ON)

def V22(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve22.GetLabel()==valve_status_ON:
            self.Valve22.SetLabel(valve_status_OFF)
```

```
        else:
            self.Valve22.SetLabel(valve_status_ON)

    def V23(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve23.GetLabel()==valve_status_ON:
            self.Valve23.SetLabel(valve_status_OFF)

        else:
            self.Valve23.SetLabel(valve_status_ON)

    def V24(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve24.GetLabel()==valve_status_ON:
            self.Valve24.SetLabel(valve_status_OFF)

        else:
            self.Valve24.SetLabel(valve_status_ON)

    def V25(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve25.GetLabel()==valve_status_ON:
            self.Valve25.SetLabel(valve_status_OFF)

        else:
            self.Valve25.SetLabel(valve_status_ON)

    def V26(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve26.GetLabel()==valve_status_ON:
```

```
            self.Valve26.SetLabel(valve_status_OFF)

        else:
            self.Valve26.SetLabel(valve_status_ON)

    def V27(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve27.GetLabel()==valve_status_ON:
            self.Valve27.SetLabel(valve_status_OFF)

        else:
            self.Valve27.SetLabel(valve_status_ON)

    def V28(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve28.GetLabel()==valve_status_ON:
            self.Valve28.SetLabel(valve_status_OFF)

        else:
            self.Valve28.SetLabel(valve_status_ON)

    def V29(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve29.GetLabel()==valve_status_ON:
            self.Valve29.SetLabel(valve_status_OFF)

        else:
            self.Valve29.SetLabel(valve_status_ON)

    def V30(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
```

```
        if self.Valve30.GetLabel()==valve_status_ON:
            self.Valve30.SetLabel(valve_status_OFF)

        else:
            self.Valve30.SetLabel(valve_status_ON)

    def V31(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve31.GetLabel()==valve_status_ON:
            self.Valve31.SetLabel(valve_status_OFF)

        else:
            self.Valve31.SetLabel(valve_status_ON)



    def V32(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve32.GetLabel()==valve_status_ON:
            self.Valve32.SetLabel(valve_status_OFF)

        else:
            self.Valve32.SetLabel(valve_status_ON)

    def V33(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve33.GetLabel()==valve_status_ON:
            self.Valve33.SetLabel(valve_status_OFF)

        else:
            self.Valve33.SetLabel(valve_status_ON)
```

```
def V34(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve34.GetLabel()==valve_status_ON:
            self.Valve34.SetLabel(valve_status_OFF)

        else:
            self.Valve34.SetLabel(valve_status_ON)

def V35(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve35.GetLabel()==valve_status_ON:
            self.Valve35.SetLabel(valve_status_OFF)

        else:
            self.Valve35.SetLabel(valve_status_ON)

def V36(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve36.GetLabel()==valve_status_ON:
            self.Valve36.SetLabel(valve_status_OFF)

        else:
            self.Valve36.SetLabel(valve_status_ON)

def V37(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve37.GetLabel()==valve_status_ON:
            self.Valve37.SetLabel(valve_status_OFF)

        else:
```

```
            self.Valve37.SetLabel(valve_status_ON)

    def V38(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve38.GetLabel()==valve_status_ON:
            self.Valve38.SetLabel(valve_status_OFF)

        else:
            self.Valve38.SetLabel(valve_status_ON)

    def V39(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve39.GetLabel()==valve_status_ON:
            self.Valve39.SetLabel(valve_status_OFF)

        else:
            self.Valve39.SetLabel(valve_status_ON)

    def V40(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve40.GetLabel()==valve_status_ON:
            self.Valve40.SetLabel(valve_status_OFF)

        else:
            self.Valve40.SetLabel(valve_status_ON)

    def V41(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve41.GetLabel()==valve_status_ON:
            self.Valve41.SetLabel(valve_status_OFF)
```

```
        else:
            self.Valve41.SetLabel(valve_status_ON)

    def V42(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve42.GetLabel()==valve_status_ON:
            self.Valve42.SetLabel(valve_status_OFF)

        else:
            self.Valve42.SetLabel(valve_status_ON)

    def V43(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve43.GetLabel()==valve_status_ON:
            self.Valve43.SetLabel(valve_status_OFF)

        else:
            self.Valve43.SetLabel(valve_status_ON)

    def V44(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve44.GetLabel()==valve_status_ON:
            self.Valve44.SetLabel(valve_status_OFF)

        else:
            self.Valve44.SetLabel(valve_status_ON)

    def V45(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'
```

```
        if self.Valve45.GetLabel()==valve_status_ON:
            self.Valve45.SetLabel(valve_status_OFF)

        else:
            self.Valve45.SetLabel(valve_status_ON)

    def V46(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve46.GetLabel()==valve_status_ON:
            self.Valve46.SetLabel(valve_status_OFF)

        else:
            self.Valve46.SetLabel(valve_status_ON)

    def V47(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve47.GetLabel()==valve_status_ON:
            self.Valve47.SetLabel(valve_status_OFF)

        else:
            self.Valve47.SetLabel(valve_status_ON)

    def V48(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve48.GetLabel()==valve_status_ON:
            self.Valve48.SetLabel(valve_status_OFF)

        else:
            self.Valve48.SetLabel(valve_status_ON)

    def V49(self, event):
        valve_status_ON = 'ON'
```

```
        valve_status_OFF = 'OFF'

        if self.Valve49.GetLabel()==valve_status_ON:
            self.Valve49.SetLabel(valve_status_OFF)

        else:
            self.Valve49.SetLabel(valve_status_ON)

    def V50(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve50.GetLabel()==valve_status_ON:
            self.Valve50.SetLabel(valve_status_OFF)

        else:
            self.Valve50.SetLabel(valve_status_ON)

    def V51(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve51.GetLabel()==valve_status_ON:
            self.Valve51.SetLabel(valve_status_OFF)

        else:
            self.Valve51.SetLabel(valve_status_ON)

    def V52(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve52.GetLabel()==valve_status_ON:
            self.Valve52.SetLabel(valve_status_OFF)

        else:
            self.Valve52.SetLabel(valve_status_ON)
```

```
def V53(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve53.GetLabel()==valve_status_ON:
            self.Valve53.SetLabel(valve_status_OFF)

        else:
            self.Valve53.SetLabel(valve_status_ON)

def V54(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve54.GetLabel()==valve_status_ON:
            self.Valve54.SetLabel(valve_status_OFF)

        else:
            self.Valve54.SetLabel(valve_status_ON)

def V55(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve55.GetLabel()==valve_status_ON:
            self.Valve55.SetLabel(valve_status_OFF)

        else:
            self.Valve55.SetLabel(valve_status_ON)

def V56(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve56.GetLabel()==valve_status_ON:
            self.Valve56.SetLabel(valve_status_OFF)

        else:
```

```
            self.Valve56.SetLabel(valve_status_ON)

    def V57(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve57.GetLabel()==valve_status_ON:
            self.Valve57.SetLabel(valve_status_OFF)

        else:
            self.Valve57.SetLabel(valve_status_ON)

    def V58(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve58.GetLabel()==valve_status_ON:
            self.Valve58.SetLabel(valve_status_OFF)

        else:
            self.Valve58.SetLabel(valve_status_ON)

    def V59(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve59.GetLabel()==valve_status_ON:
            self.Valve59.SetLabel(valve_status_OFF)

        else:
            self.Valve59.SetLabel(valve_status_ON)

    def V60(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve60.GetLabel()==valve_status_ON:
            self.Valve60.SetLabel(valve_status_OFF)
```

```
        else:
            self.Valve60.SetLabel(valve_status_ON)

    def V61(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve60.GetLabel()==valve_status_ON:
            self.Valve60.SetLabel(valve_status_OFF)

        else:
            self.Valve60.SetLabel(valve_status_ON)

    def V62(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve60.GetLabel()==valve_status_ON:
            self.Valve60.SetLabel(valve_status_OFF)

        else:
            self.Valve60.SetLabel(valve_status_ON)

    def V63(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve60.GetLabel()==valve_status_ON:
            self.Valve60.SetLabel(valve_status_OFF)

        else:
            self.Valve60.SetLabel(valve_status_ON)



    def V64(self, event):
        valve_status_ON = 'ON'
```

```
        valve_status_OFF = 'OFF'

        if self.Valve60.GetLabel()==valve_status_ON:
            self.Valve60.SetLabel(valve_status_OFF)

        else:
            self.Valve60.SetLabel(valve_status_ON)



    def V65(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve60.GetLabel()==valve_status_ON:
            self.Valve60.SetLabel(valve_status_OFF)

        else:
            self.Valve60.SetLabel(valve_status_ON)

    def V66(self, event):
        valve_status_ON = 'ON'
        valve_status_OFF = 'OFF'

        if self.Valve60.GetLabel()==valve_status_ON:
            self.Valve60.SetLabel(valve_status_OFF)

        else:
            self.Valve60.SetLabel(valve_status_ON)

#*********************          Define Pumps          ********************
#*****************************************************************************************

    def P1 (self,event):

        wx.MessageBox("P1 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P2 (self,event):
```

```
    wx.MessageBox("P2 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P3 (self,event):
    wx.MessageBox("P3 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P4 (self,event):
    wx.MessageBox("P4 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P5 (self,event):
    wx.MessageBox("P5 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P6 (self,event):
    wx.MessageBox("P6 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P7 (self,event):
    wx.MessageBox("P7 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P8 (self,event):
    wx.MessageBox("P8 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P9 (self,event):
    wx.MessageBox("P9 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

def P10 (self,event):
    wx.MessageBox("P10 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)




##############################################################################
```

#********** listbox **********

#**********

class ValveListBox(wx.Frame):

```
 def __init__(self):

    wx.Frame.__init__(self, None, -1,'Pumps list', size=(250,300))

    panel = wx.Panel(self,-1)

    #List1=Listbox()
    #List1.insert(1,'P1')
    #List1.insert(2,'P2')
    #List1.insert(3,'P3')
    #List1.insert(4,'P4')
    #List1.insert(5,'P5')
    #List1.insert(6,'P6')
    #List1.insert(7,'P7')
    #List1.insert(8,'P8')
    #List1.insert(9,'P9')
   # List1.insert(10,'P10')
    LABELSTYLE = wx.BORDER_SUNKEN | wx.ST_NO_AUTORESIZE |
wx.ALIGN_CENTER_HORIZONTAL



    wx.StaticText(panel,20," Pumps Selected : ",(10,10))
    wx.StaticText(panel,-1," P1 ",(18,40))
    wx.StaticText(panel,-1," P2 ",(18,70))
    wx.StaticText(panel,-1," P3 ",(18,100))
    wx.StaticText(panel,-1," P4 ",(18,130))
    wx.StaticText(panel,-1," P5 ",(18,160))
    wx.StaticText(panel,-1," P6 ",(18,190))
    wx.StaticText(panel,-1," P7 ",(18,220))
    wx.StaticText(panel,-1," P8 ",(18,250))

    wx.StaticText(panel,-1," P9 ",(18,280))
    wx.StaticText(panel,-1," P10 ",(18,310))

    self.Pump1 = wx.StaticText(panel, size = (40, -1), pos = (60, 40), style = LABELSTYLE)
    self.Pump2 = wx.StaticText(panel, size = (40, -1), pos = (60, 70), style = LABELSTYLE)
    self.Pump3 = wx.StaticText(panel, size = (40, -1), pos = (60, 100), style = LABELSTYLE)
```

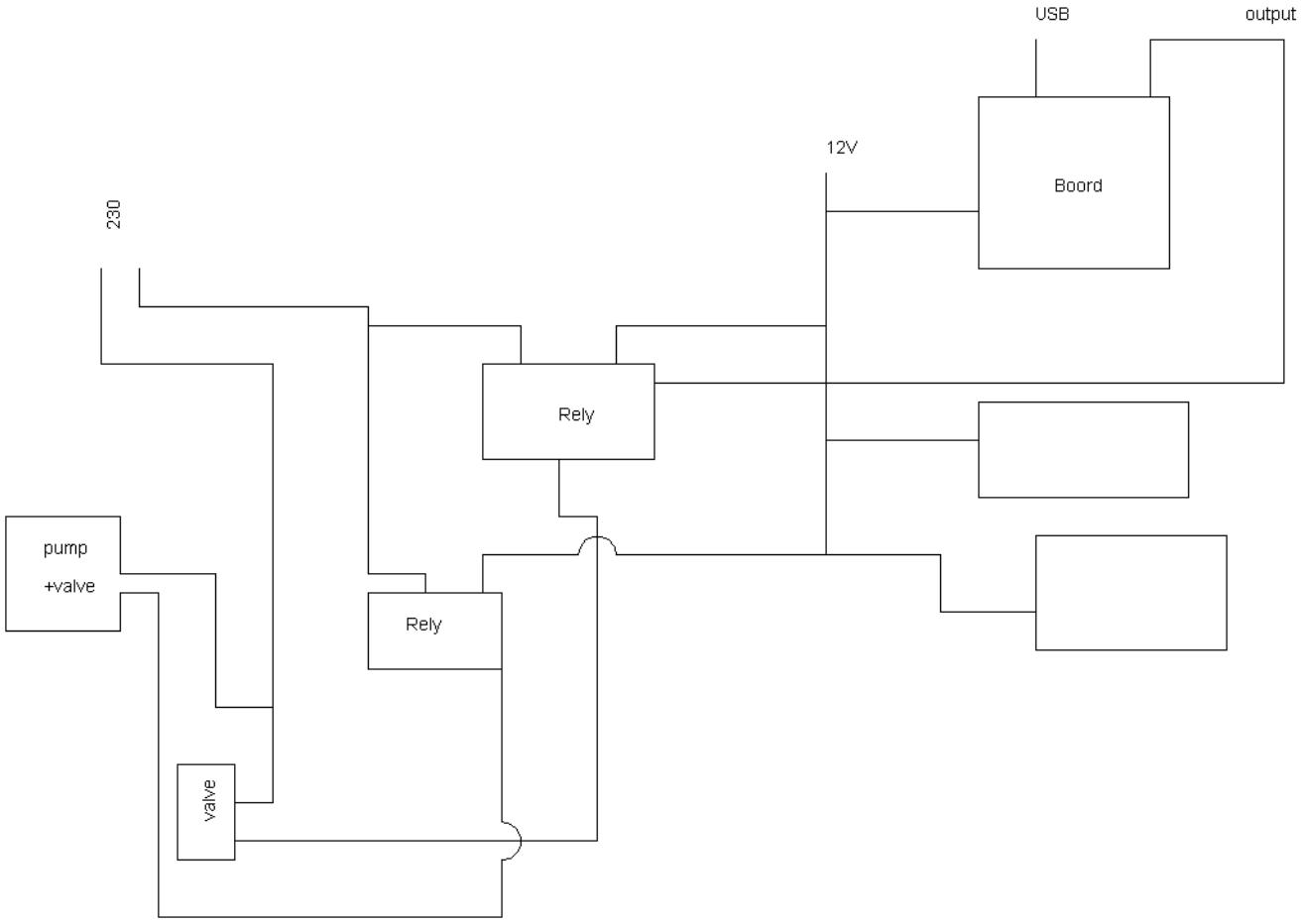
```
self.Pump4 = wx.StaticText(panel, size = (40, -1), pos = (60, 130), style = LABELSTYLE)
self.Pump5 = wx.StaticText(panel, size = (40, -1), pos = (60, 160), style = LABELSTYLE)
self.Pump6 = wx.StaticText(panel, size = (40, -1), pos = (60, 190), style = LABELSTYLE)
self.Pump7 = wx.StaticText(panel, size = (40, -1), pos = (60, 220), style = LABELSTYLE)
self.Pump8 = wx.StaticText(panel, size = (40, -1), pos = (60, 250), style = LABELSTYLE)
self.Pump9 = wx.StaticText(panel, size = (40, -1), pos = (60, 280), style = LABELSTYLE)
self.Pump10 = wx.StaticText(panel, size = (40, -1), pos = (60, 310), style = LABELSTYLE)
```

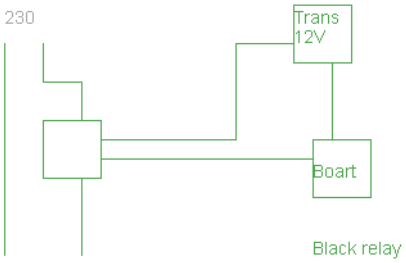
```
#sampleList= ['P1','P2','P3','P4','P5','P6','P7','P8','P9','P10']
 #
'V21','V22','V23','V24','V25','V26','V27','V28','V29','V30','V31','V32','V33','V34','V35','V36','V37','V38','V39','V
40',
  #        'V41','V42','V43','V44','V45','V46','V47','V48']
   #wx.StaticText(panel,-1,"Pumps",(20,5))
   #wx.ComboBox(panel,-1," valve ",(20,30),wx.DefaultSize,sampleList,wx.CB_DROPDOWN)
  #wx.ComboBox(panel,-1," valve ",(150,30),wx.DefaultSize,sampleList,wx.CB_SIMPLE)

# listBox = wx.ListBox(panel, -1, (0, 0), (250, 300), sampleList,wx.LB_EXTENDED)
# listBox.SetSelection(3)

# listbox = wx.CheckListBox(panel,-1,(20,20),(100,300),sampleList, wx.LB_SINGLE)
def P1 (self,event):

    pump_status_ON = 'ON'
    pump_status_OFF = 'OFF'

    if self.Pump1.GetLabel()==pump_status_ON:
        self.Pump1.SetLabel(pump_status_OFF)

    else:
        self.Pump1.SetLabel(pump_status_ON)
```

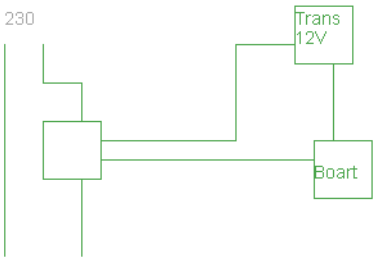
```
            wx.MessageBox("P1 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P2 (self,event):
        wx.MessageBox("P2 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P3 (self,event):
        wx.MessageBox("P3 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P4 (self,event):
        wx.MessageBox("P4 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P5 (self,event):
        wx.MessageBox("P5 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P6 (self,event):
        wx.MessageBox("P6 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P7 (self,event):
        wx.MessageBox("P7 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P8 (self,event):
        wx.MessageBox("P8 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P9 (self,event):
        wx.MessageBox("P9 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)

    def P10 (self,event):
        wx.MessageBox("P10 is Open ","Pump", wx.OK|wx.ICON_INFORMATION)




#**********************          main definition and loop          ********************
#*****************************************************************************************

def main():
    """Testing"""
```

```
    MyBackgroundPanel = wx.PySimpleApp(wx.Panel)
    f = MyFrame()

#in D:\AECENAR\MEGBI\MEGBI-VPP\AUT\CDMasterThesisHayssamHindy\GUI\101005_MEGBI-VPP_downstream.py the follwowing two lines are not in comment
    #d=ValveListBox()
    #d.Show()
    f.Center()
    f.Show()
    MyBackgroundPanel.MainLoop()

if __name__ == "__main__":
    main()
```

9.5 Annex 3: Electrical circuits for connection between veleman board, relais and actuators (pump. valve)




230
Trans
12V
Boart
Black relay


230
Trans
12V
Boart

10. References / مراجع

[1]http://www.aecenar.com/publications

[2] http://www.aecenar.com/downloads/cat_view/7-megbi-institute

[3] http://www.aecenar.com/downloads/cat_view/3-meae-institute?start=10

[4]http://www.eio.com/p-1603-velleman-k8061-extended-usb-interface-board.aspx

[5]http://www.apogeekits.com/images/usb_io_interface.jpg

[6] www.pc-control.co.uk/relays.htm

[7] http://www.velleman.eu/products/view/?id=364910

[8]http://www.tutorialspoint.com/python/

[9] https://www.draw.io/

Most Devices manufactor contacts from www.alibaba.com

SOFTWARES AND VIDEO DOCUMENTATION ARE FOUND ON THE DISC SUPPLIED WITH THIS DOCUMENT IN THE LAST PAGE