LEBANESE INTERNATIONAL UNIVERSITY

Autonomous Quadcopter Swarm for Early FIre detection system using UAV and AI

- Project Report 1 (Oct 2020 – June 2021) -

Based on Master Thesises Raja M. Murad, Nour O. Karim, Ali A. Assaad, Mohammad M. Mourad, submitted to the School of Engineering of the Lebanese International University (LIU), Tripoli, Lebanon, in partial fulfillment of the requirements for the degree of master of Science in Electrical Engineering
Spring 2020-2021, Supervisor Dr. Abdelrazzak Merheb

DEDICATION

This thesis is dedicated to everybody who helped in succeeding this project, our family, and friends. The thanks go also to the Lebanese International University and its entire staff especially Dr. Abdelrazzak Merheb and Dr. Samir Mourad for their support, guidance, and efforts.

ACKNOWLEDGEMENTS

We take this opportunity to express our profound gratitude and deep regards to our supervisor Dr. Abdelrazzak Merheb for his useful help and continuous support, monitoring, and encouragement throughout our thesis. Dr. Abdelrazzak Merheb gave us the idea for this project and helped us step by step in order to get this project done perfectly and gave us the moral and practical support. We cannot disregard the effort of Dr. Samir Mourad who have always been very responsive in providing necessary information and giving us access to his lab in order to help us proceed in our project.

Furthermore, we would like to thank our friends for their encouragement without which this project could not be possible to be accomplished.

ABSTRACT

In the last decade, autonomous quadcopters have gained a huge attention in both academic and industrial fields. They have been developed and improved to solve complex problems that have a direct risk on humans. In this thesis, an autonomous, triangular shaped, swarm of quadcopters based on leader-follower scheme based on GPS position will be designed and implemented for the purpose of forest fire and smoke detection that will be implemented based on machine learning. The system will receive live video stream from a flying quadcopter over a forest region and detects the fire and smoke in the received frame. The system uses Artificial Neural Networks (ANNs) based on Darknet-53 and YOLO V3 pretrained network with 1200 train images and 200 test images for model validation. The swarm is to be controlled via a ground station controller which gives the high-level commands and will be flying over 120 meters from ground. Also, a Kalman filter will be added to estimate the quadcopters' position in case of GPS shortage. For the fire detection system, the base station is composed of a laptop PC which will receive a live video-stream from a quadcopter flying over a certain region of forests and then run the detection algorithm on CPU. For testing purposes, the swarm system will be designed and tested in 2 stages: standalone control system for each drone and a swarm controller and to test the fire detection system accuracy, virtual fires and smokes are simulated in a recorded drone live footage. The system effectively detected fire and smoke in a video-stream with accuracies of 98% and 96% respectively. Whereas, for the swarm system, unsatisfactory results were found in terms of cooperative control but with successful individual control.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION 17

1.1. Background 17

1.2. Problem Statement 22

1.3. Thesis Overview 24

1.4. literature survey 26

1.5. Thesis Outline 28

CHAPTER 2. SYSTEM DESIGN 29

2.1. Introduction 29

2.2. Preliminaries and Assumptions 31

2.3. Kinematic Model 32

2.4. Dynamical Model 34

2.5. Linear Model 37

2.6. Controllers 38

2.6.1. PID Controller 39

2.6.2. Position Controller 41

2.6.3. Obstacle Avoidance 43

2.6.4. Attitude Controller 45

2.6.5. Formation Controller 48

2.7. Conclusion 53

CHAPTER 3. PROJECT SPECIFICATIONS 54

3.1. Introduction 54

3.2. System and Equipment 55

3.3. MULTIWII Flight Controller 56
3.4. NAZA Flight Controller 58
3.5. Parrot AR DRONE 59
3.6. Quadcopter Frames 61
3.7. Brushless DC Motors 62
3.8. Propellers 63
3.9. Electronic Speed Controller (ESC) 63
3.10. Battery 64
3.11. GPS Sensor 66
3.12. Power Distribution Board 67
3.13. Arduino Mega 2560 68
3.14. Gimbal 68
3.15. Vision 69
3.16. Communication 70
3.17. Central PC 71
3.17.1. Hardware Requirements 71
3.17.2. Software Requirements 72
3.17.2.1. Python 73
3.17.2.2. LabelImg 73
3.18. Conclusion 74
CHAPTER 4. PROJECT DESIGN 75
4.1. Introduction 75
4.2. Methodology 75
4.3. Naza Controller-based Quadcopter (Leader) 76
4.3.1. RC Transmitter Signal Decoding 79

4.3.2. Naza GPS/Compass Decoding 80
4.4. Multiwii Controller-based Quadcopter (Follower) 81
4.4.1. Altitude Control 84
4.4.1.1. Moving Average and Complimentary Filter 85
4.4.1.2. Adaptive PID Controller 87
4.4.1.3. Battery Compensation 88
4.5. Kalman Filter 90
4.6. Position Controller 92
4.7. convolutional Neural networks 95
4.8. Yolo V3 96
4.9. Training and Testing 98
4.10. Gimbal Control 99
4.11. Conclusion 102
CHAPTER 5. NON-TECHNICAL ASPECTS 103
5.1. Introduction 103
5.2. Economical/Financial 103
5.3. Project Management 105
5.4. Ethical and Social 106
5.4.1. Quadcopter 106
5.4.2. Camera 107
5.4.3. Neural Networks 107
5.5. Environmental and Sustainability 108
5.6. Standards 109
5.6.1. Quadcopter Standards 109
5.6.2. Neural Networks Standards 110

5.7. Conclusion 111

CHAPTER 6. RESULTS 112

6.1. Introduction 112

6.2. Kalman Filter 112

6.3. Naza-based Quadcopter 113

6.4. AR-Drone 114

6.5. Swarming 115

6.6. neural networks 116

6.7. Gimbal 118

6.8. conclusion 120

CHAPTER 7. CONCLUSION 121

7.1. General Conclusion 121

7.2. Future Work 123

LIST OF FIGURES

Figure 1.1. Breguet Richet Gyroplane No.1 17

Figure 1.2. Quadcopter maneuvers 18

Figure 1.3. AUB Statistics: Hourly Temperature Compared with Maximum and Minimum Historical Temperatures in 2019 23

Figure 2.1. Leader-Follower high-level controller block diagram 30

Figure 2.2: Quadcopter high-and low-level controllers 30

Figure 2.3. Quadcopter rotation angles 31

Figure 2.4. Quadcopter rotation angles and coordinate frames 32

Figure 2.5. Translational dynamics Simulink model 36

Figure 2.6. Rotational dynamics Simulink model 36

Figure 2.7. Single quadcopter control system block diagram 39

Figure 2.8. PID controller block diagram 40

Figure 2.9. Position controller Simulink model 42

Figure 2.10. Position controller with obstacle avoidance algorithm 43

Figure 2.11. Repulsive field 44

Figure 2.12. Root locus plot for roll, pitch, and yaw systems 46

Figure 2.13. Attitude controller implemented in Simulink 47

Figure 2.14. Roll and pitch response to step angle of 25 degrees 48

Figure 2.15. Yaw response to step angle of 90 degrees 48

Figure 2.16. Formation in 2D 49

Figure 2.17. L-F formation control Simulink model 49

Figure 2.18. 3D and 2D path-tracking without obstacles 51

Figure 2.19. X and Y leader's position with respect to time 51

Figure 2.20. 2D Position tracking with obstacle avoidance 52

Figure 3.1. Project main parts.....54
Figure 3.2. MULTIWII SE V2.0 56
Figure 3.3. Data exchange between MULTIWII and Arduino mega.....58
Figure 3.4. NAZA_M V2 flight controller 58
Figure 3.5. Data exchange between NAZA-M V2 and Arduino mega.....59
Figure 3.6. Parrot AR Drone.....60
Figure 3.7. Quadcopter frames chosen for the project 61
Figure 3.8. PROPDRIVEV2 and A2212/13T motors.....62
Figure 3.9. 1045 Propellers 63
Figure 3.10. ECSs chosen for the project.....64
Figure 3.11. Batteries chosen for the project.....65
Figure 3.12. Ublox Neo 6m GPS 66
Figure 3.13. Power distribution board.....67
Figure 3.14. Gidy camera gimbal.....68
Figure 3.15. AKK video transmission system 70
Figure 3.16. HP laptop 15-da1xx.....71
Figure 3.17. Python logo.....73
Figure 3.18. LabelImg logo.....73
Figure 4.1. Autonomous State Diagram.....76
Figure 4.2. Arduino-Naza Physical Connections 78
Figure 4.3. Arduino-Naza Circuit Diagram.....78
Figure 4.4. RC Transmitter Signal 79
Figure 4.5. PCINT Subroutine 80
Figure 4.6. Naza GPS Pinout 81
Figure 4.7. Arduino-Multiwii Circuit Diagram.....82

Figure 4.8. Multiwii GUI 83
Figure 4.9. Multiwii-Arduino Physical Connections 83
Figure 4.10. Altitude Control Block Diagram 85
Figure 4.11. Altitude Measurement of the Multiwii-based Follower. Complimentary Filtering in Green, Moving Average in Blue and the Raw Measuremnt From The Ultrasonic Sensor in Red 87
Figure 4.12. Voltage Divider Circuit 89
Figure 4.13. Battery Compensation PWM and The Required Hovering PWM vs. Battery Voltage 90
Figure 4.14. ECEF and Inertial Frames 93
Figure 4.15. Latitude and Longitude of Earth 93
Figure 4.16. Convolutional Neural Network architecture 95
Figure 4.17. Yolo v3 Architecture 97
Figure 4.18. Sigmoid Activation Function 98
Figure 4.19. LabelImg Annotation Tool 99
Figure 4.20. Gimbal Control Strategy 100
Figure 4.21. Gimbal Designed 3D and Physical Models 101
Figure 4.22. Servo Motor Control Circuit Connections 102
Figure 5.1. Thesis Gantt chart 105
Figure 5.2. Thesis Gantt Chart 106
Figure 5.3. GWP and particulates of 1 Kilometer delivery by electric motorcycle, gasolian motorcylce, and drone. **(a)** GWP of 1 Km delivery; **(b)** $PM_{2.5}$ of 1 Km delivery 108
Figure 6.1. Kalman Filter Algorithm. Stationary Quadcopter (Left), a Small Quadcopter Tour (Right) 113

Figure 6.2. Estimated Quadcopter Speed.....113

Figure 6.3. Calculated Heading (Left), Actual Path (Right).....114

Figure 6.4. Quadcopter Swarm Following a Desired Setpoint.....115

Figure 6.5. Training Losses (a), Detection Accuracy (b), Training Precision Percentage (c), Average Intersection Over Union (d), F1 Score (e), and Mean Training Precision (f).....117

Figure 6.6. Detected Fires and Smokes in a Video Frame.....118

Figure 6.7. Gimbal Input Controlled from PC.....120

LIST OF TABLES

Table 2.1. Physical parameters 37

Table 2.2. PID position constants 50

Table 3.1. Hardware equipment used in the system 55

Table 3.2. MULTIWII SE V2.0 features and specifications 57

Table 3.3. NAZA-M V2 Specifications 58

Table 3.4. AR Drone specifications 60

Table 3.5. Turnigy Heavy Aerial Lift and DJI F450 Specifications 61

Table 3.6. BLDC motors specifications 62

Table 3.7. ESCs Specifications 64

Table 3.8. HRB and AR Drone battery specifications 66

Table 3.9. Ublox Neo 6m GPS specifications 67

Table 3.10. Power distribution board (type A) Specifications 67

Table 3.11. Gidy camera gimbal specs 69

Table 3.12. Camera specs 69

Table 3.13. Communication system specs 70

Table 3.14. HP laptop specs 72

Table 3.15. Google Collabs hardware specs 72

Table 4.1. Naza Features 77

Table 4.2. Multiwii Attitude PID Tuned Gains 84

Table 5.1. Material Cost 104

Table 5.2. Engineering Staff Cost 105

Table 5.3. Environmental Impact of Selected Categories 108

Table 5.4. FAA’s Model Aircraft Rules 109

Table 5.5. IEEE Standards Related to Neural Networks 110

Table 6.1. Gimbal Roll and Yaw Data Associated to the frame coordinates 119

LIST OF SYMBOLS AND ABBREVIATIONS

APF: Artificial Potential Field

BBR: Behavior-Based Robotics

EGA: Enhanced Genetic Algorithm

GPS: Global Positioning System

IMU: Inertial Measurement Unit

KF: Kalman Filter

L-F: Leader-Follower

NIFC: National Interagency Fire Center

OA: Obstacle Avoidance

PID: Proportional, Derivative and Integral

SAR: Search and Rescue

SMC: Sliding Mode Control

UAVs: Unmanned Aerial Vehicles

VTOL: Vertical Takeoff and Landing

ANN: Artificial Neural Networks

CPU: Central Processing Unit

EFFIS: European Forest Fire Information System

ESC: Electronic Speed Controller

FFDI: Forest Fire Detection Index

FPS: frames per second

GPU: Graphical Processing Unit

Li-ion: Lithium ion

LiPo: Lithium Polymer

ML: Machine Learning

NN: Neural Networks

RCNN: Region-based Convolutional Neural Network

CHAPTER 1. INTRODUCTION

1.1. BACKGROUND

Unmanned Aerial Vehicles (UAVs) have grabbed a great attention in both academic and practical fields through the past decades. They are used in various fields such as military reconnaissance, like gathering sensitive data during and after military missions to aid in security and decision-making. In addition, UAVs are used in Search and Rescue (SR) missions, where 3-D mapping of catastrophic regions can help rescue teams for better estimation and preparation before entering hazardous situations. In addition, UAVs can be used in agriculture field, taking advantage of them in a variety of farming needs such as spraying fertilizers and insecticides, identifying weed infestations, and monitoring crop health. Other applications such as live entertainment, inspection, weather forecasting and maritime operations can be found in [1] [2]. Because of their simple structure, small size, strong mobility and low cost, UAVs were chosen to complete hazardous tasks. In 1907, the Breguet Brothers built the first flyable quadcopter called “gyroplane No.1 shown in Figure 1.1 [3] [4].

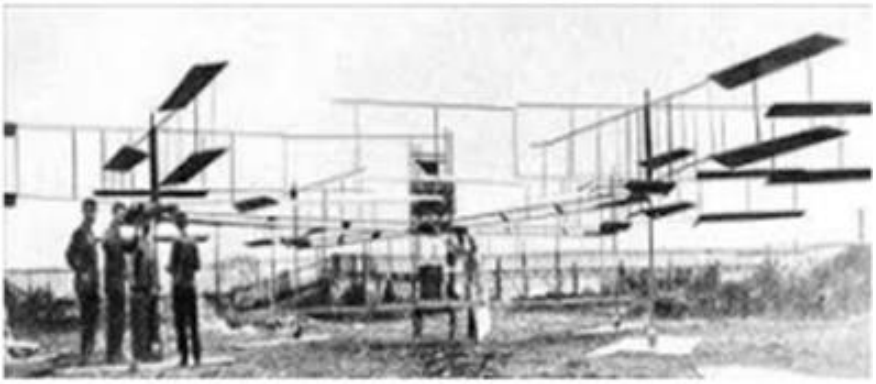

Figure 1.1. Breguet Richet Gyroplane No.1

So far, engineers have developed quadcopter UAVs to solve the problem of vertical flights which pilots had with conventional winged UAVs [5]. Compared with the latter, quadcopters do not require a tail rotor to stabilize their heading, they can cancel out net torques naturally because of their rotors' mount. quadcopters can fit in more sophisticated flight environments with their ability to travel in narrow spaces, apply roll and ultra-soft flight and hovering, take-off and land vertically, (VTOL), and move in a flexible manner [6]. Quadcopter maneuvering can be achieved by varying rotors' speeds as shown in Figure 1.2. With the evolution of quadcopter UAVs, they became fully autonomous. They can track specific trajectory, hold their altitude, and maneuver in narrow areas on their own. Scientists and control system engineers developed various controllers to achieve quadcopter's full autonomy such as conventional PID controllers, backstepping, robust controllers (such as Sliding Mode Controllers (SMC)), and fuzzy logic controllers. Each of these is characterized by its flexibility and boundaries. For example, conventional PID controllers do not take into consideration the unknown disturbances acting on the system, while robust controllers do.

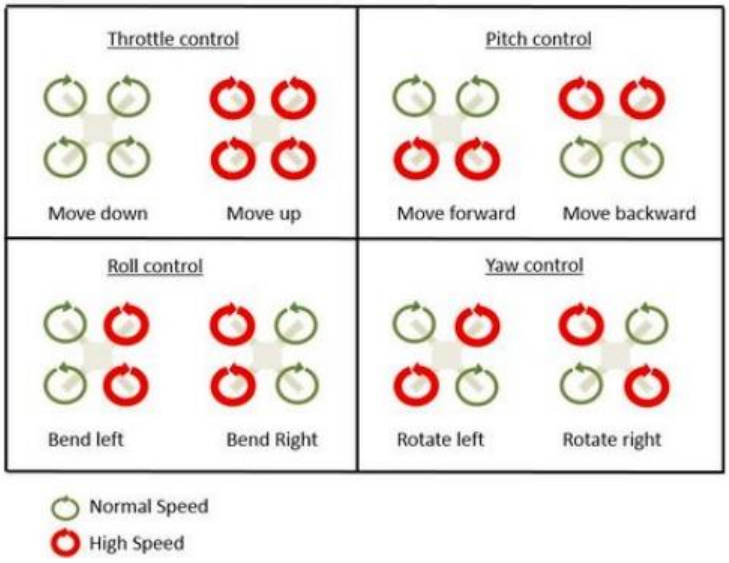

Figure 1.2. Quadcopter maneuvers

Nowadays, world becomes more complex and more connected than ever before which led to more complicated objectives, and the need to deal with complex tasks requires new methodologies, such as imitating swarm of robots to survey unknown environments. Compared with a single quadcopter, using a swarm of quadrotors in missions can increase the efficiency as well as it increases the probability of mission success. In addition, using quadcopter swarm can increase the surveillance area, scan a whole damaged building quickly, and reduce the expense of military missions. Quadcopter team is a popular research topic as it can solve real-world tasks efficiently [7] [8] [9].

For instance, in 2019, heat and drought wave different areas all around the world. Consequently, many countries suffered from wildfires. Around 50,447 wildfires occurred, which burned around 4.7 million acres of green lands according to National Interagency Fire Center (NIFC) [10]. In 2019, series of 100 fires have broken out within 24 hours in Lebanon as officials said. This catastrophe started in Lebanon's western mountains, and spread to other areas, because of a heatwave and strong winds which the country faced October. Furthermore, Lebanon's ability to face this catastrophe was almost non-existent. Lebanon's firefighting helicopters could not be used because of the lack of maintenance [11] [12].

In order to respond to this disastrous event, emergency responders around the world started using next-generation technologies to help avoid wildfires, prevent spread wildfires, and extinguish them when they occur [13]. For instance, the fire department in State of Michigan in USA in coordination with the University of Michigan College of Engineering used a swarm of quadcopters to combat wildfire effectively with the goal of finding, mapping wildfires and reporting the estimated fire boundary. The swarm flies above the fire boundary and then reports accurate and real-time fire estimates. Thus, responders can know where they

can go to be safe, where they cannot go, and how they can help certain people who are in need [14].

There are plenty of high-level controllers in which engineers trying to improve by time to establish a fully-autonomous formation control such as virtual structure [15], Behavior-Based Robotics (BBR) [16], and leader-follower technique [17]. Where the leader moves along with a predefined trajectory while the followers are controlled to maintain a desired position (orientation and distance) with respect to the leader [18]. This approach is highly effective and can be reconfigured easily [19]. These algorithms should be optimally combined with path-planners in order to achieve a safe and efficient mission.

Path-planning algorithms, in such missions, should generate an obstacle-free path to the quadcopter team taking into consideration the anti-collision of agents. There are many path-planning and obstacle avoiding algorithms used by scientists such as backstepping obstacle-avoiding techniques, genetic, and Neural Network UAV path-planning. Artificial Potential Fields (APF) is also another path-planning algorithm and is widely used in robotics field. It first started by Oussama Khatib in 1980's for path-planning of mobile robots and manipulators [20]. APF can briefly be described as a goal attracting and obstacles repelling the robot through artificial forces. The sum of forces applied determines the speed and direction the robot should take in order to reach its goal and repel from obstacles.

Moreover, the advancement of industry in the last century increased the environmental pollution and climate change in many regions of the world. This led to rising global temperatures which is one of the reasons for the outbreak of fires in large areas of forests [1] [2]. In 2013, the US had lost 104,131 Hectares of forest due to fires [3]. According to the European Forest Fire Information System (EFFIS) report, the Middle East and North Africa have lost at least 176,116 Hectares of Forest in 2014 [4].

In 2019, a series of fires broke out in the forests of Lebanon and nearby countries with nearly 100 fires on the Lebanese territories, according to the Lebanese Civil Defense [5]. In 2020, and based on the data and estimates of the municipalities in the villages, the area burned in this incident has reached 12 million square meters [6]. Unfortunately, this happens every year in Lebanon. Meteorological experts stated that the fires were caused by high temperatures, which reached 38 degrees Celsius (nearly 10 degrees above the average), and dry winds that contributed to forest fires [7].

Problems caused by the lack of technological aspects in the field of fire detection are obviously disastrous. There is still no way to know when the early flames are that caused the fire ignited. Because there are no methods used in Lebanon that keep on checking and identifying the forests' status, it is always too late to prevent forest fires that happen every year. These fires contribute to a great physical, climate, and economic losses. Thus, it is a must to develop monitoring systems that detect fires, especially early flames, and show their status (level, direction, speed). This will help firefighters in accessing the stage safely and treating fire quickly and efficiently which will definitely save more lives and green spaces. Unfortunately, traditional fire detecting sensors, such as ionization smoke sensors and flame detectors frequently lack efficiency when stationed in nature, and often have false alarms. This raises an urgent need for a better and more accurate technology.

The fire detection system using drones has gained a huge turnout especially in the past two decades. This system has been gradually developed throughout the years, starting from remote-controlled drones with smoke sensors, to installing cameras on autonomous drones that take pictures of the fire and transmit live video stream of the fire and how it is moving. This technology helped NASA to detect effectively the California wildfire in 2008 and helped preventing it from spreading in an uncontrolled manner [8]. On one hand, using drones to detect fires has positive aspects. They help firefighters to specify the state of the

fire like its direction and extension. Besides, their convenient size helps them penetrate through areas unreachable by pilots. Drones also cost less than helicopters and decrease human losses by being their substitute. On the other hand, there are drawbacks. Being always available and hovering above forests are constrained by their low flying time and some technical risks.

Firstly, continuous use of quadcopters results in less efficient motors, this is called motor aging. Motors will lose their power as they "grow up" which may cause undesirable behavior of the drone and maybe crashing. Moreover, quadcopters' communication channels maybe interfered with other signals. This risk causes loss of information between sender and receiver and may end up with a disastrous situation. In addition, and one of the most common risks, sudden power death due to battery deficiency may occur and crashing will defiantly occur. These risks should be taken into consideration and never be ignored in the design stage of a quadcopter.

1.2. PROBLEM STATEMENT

Lebanon has faced many events which tested its readiness and capabilities to face fires throughout the last decade. According to the American University of Beirut [21], fires in Lebanon were classified as a recurring disaster with no efficient intervention. In 2010, 320 fires ignited with 4661 hectares burned. In 2014, 156 fires ignited which resulted in 1852 hectares burned. In 2016, 260 fires ignited with 1871 burned hectares. And, in October 2019, more than 120 fires ignited simultaneously within 48 hours only. Fire density was approximated to be 0.16 fires/km which is 10 times larger than the Amazon's famous fire in 2019. As a result, significant losses in private and public properties appeared, more than 2000 hectares of green land were burned and 1 person died because of asphyxiation. These catastrophic events have several factors. Weather conditions are one of the factors. During

October 2019, Lebanon faced a heatwave with maximum and minimum temperatures exceeding historical averages, low humidity, and wind speed of 10 m/s (normally 3 m/s in Beirut). Figure 1.3 shows the hourly temperature in October 2019 compared with historical average temperatures. Moreover, weak response is also a main factor. It includes slow early response, lack in firefighters, and absence of necessary equipment to face fires. In addition, one of the most important factors is the lack of prevention measures. Lebanon fire prevention system lacks early fire detection. For this reason, deploying a monitoring system is a must and autonomous swarm of quadcopters could be one of these systems as it provides an "eye in the sky" for stakeholders and since multiple quadcopters can increase coverage area and survey larger areas with at same time.

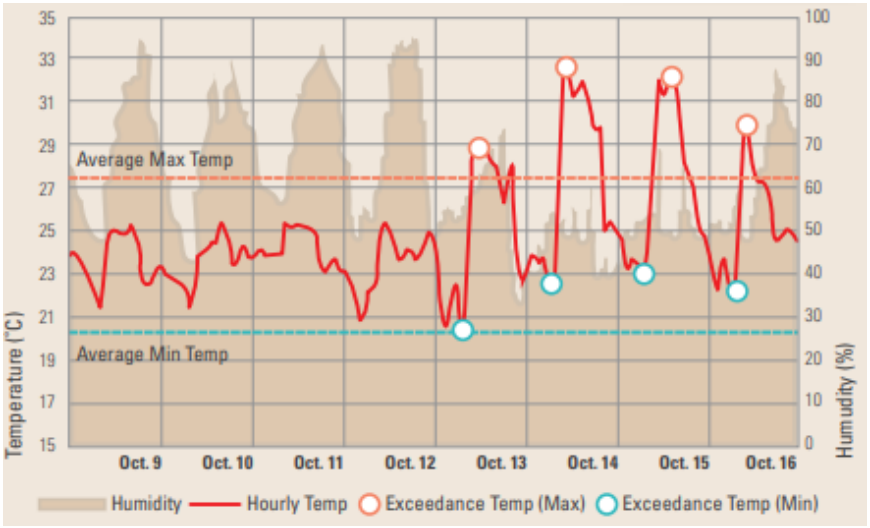

Figure 1.3. AUB Statistics: Hourly Temperature Compared with Maximum and Minimum Historical Temperatures in 2019

However, deploying swarm of quadcopters is a challenging task. Quadcopters should behave as a coherent team to complete their surveillance mission in the most efficient way and with no crashing. In a perfect world, a quadcopter team can follow its predefined trajectory infinitely without colliding because of its perfect sensors and communication time almost negligible. Unfortunately, in real world applications, perfect sensors and ultimate coordination do not exist and have a lot of constraints.

A Quadcopter swarm is susceptible to sensor noises which weakens the localization and quality of the formation and may result in formation loss or crashing. Moreover, formation control may encounter sudden unknown type of disturbances including strong wind gust, flying birds, or sudden communication loss with positioning system satellites. These disturbances have high occurrence probability and cannot be predicted. Furthermore, formation path-planner, which is responsible to plan an obstacle-free path, is vulnerable to the dynamics of moving obstacles in the surrounding environment where fast-moving or nonpredictable objects result in unstable decisions. In addition, black smoke released from fires can temporarily negatively affect the behavior of onboard sensors and cameras for each agent in the formation. These mentioned constraints limit the deployment process of formation controllers and are extremely challenging to deal with.

Early fire detection is very necessary. Take the wildfire that broke in 2019 in the Lebanese forests for instance. This fire caused dangerous injuries for more than 88 civilians [9] and 5 firefighters and burnt at least 4 houses in the surrounding area [10]. 3700 acres of green land became ashes in 48 hours, which means the loss of thousands of olive trees and other fruity trees that their owners depended on for their income [9]. Usually, the detection of wildfires is late due to either false alarms or the complete absence of alarms in some forests. This partially eliminates the possibility to access the terrain which allows the fire to nurture by the various fuel sources in the forest. Thus, the need for an efficient way to detect early fires is extremely necessary to avoid further losses.

1.3. THESIS OVERVIEW

In this thesis, a decentralized, autonomous quadcopter team consisting of 3 quadcopters will be designed and implemented which is characterized by:

- Fly in a triangular-shaped formation with 5 meters safe distance between agents.

- Track a predefined obstacle-free path, with a longitudinal speed of $\leq 5\frac{m}{s}$ over regions with high probability of fires.
- Send attitude, altitude, position and live-stream data to a Ground Control Station (GCS) in real-time.
- Land after 20 minutes of continuous flight.

The swarm is to be formed using Leader-Follower (L-F) structure, where one quadcopter sends its position information to all followers which by turn, maintain a specific distance from it without losing the formation. For attitude, altitude and position tracking, a PID controller will be implemented. Artificial Potential Field (APF) is chosen as a path-planning algorithm for its simplicity and high efficiency because it takes into consideration static and dynamic obstacles located around. In addition, and to deal with uncertainties in sensors and unknown disturbances, a Kalman Filter (KF) is to be implemented onboard of every single quadcopter as an optimal state estimator. The behavior of proposed controllers, path-planner and estimation algorithm is to be tested on MATLAB simulation environment.

As for the fire detection system, this thesis is dedicated to design and implement an early fire detection system, with the aid of quadcopter, based on Machine Learning (ML)/Neural Networks (NN).

The system is responsible to detect fires in video frames received from quadcopters flying over a region of forest. For this reason, a “Yolov3” neural network model will be used to detect fire in a frame. This model has been chosen for its high training accuracy and the ability to work with low-cost computers. The model will be trained using 1000 of positive images (images with fires). The labeling of images will be done using “labelImg” software. Model training will be handled over “Google Collabs” that offers a free GPU. In addition, a simple proportional controller will be used to calculate the necessary actions for a 2-D gimbal in order to keep the camera focused on fire region.

1.4. LITERATURE SURVEY

There are a lot of work relating path-planning and formation control of quadcopters have been published. For example, Nguyen and Sung [22] proposed a robust adaptive formation controller to keep a team of quadcopters, based upon leader-follower scheme, in a specific formation with the presence of unknown uncertainties. Promising results obtained by simulations. However, they did not test it over real quadcopters. Another method presented by Falin Wu et al. [23] where they used a linear PID controller to control each individual quadcopter along with a Sliding Mode Controller (SMC) to solve the formation problem. They acquired effective performance upon simulations, ignoring practical noises and communication delays. Moreover, similar work presented by Mu et al. [24], Khaled and Youmin [25] and Mercado et al. [26] where they also used SMC to solve the leader-follower tracking formation control problem. The SMC was very responsive to keep the formation in shape but its main disadvantage is the chattering of control input which results in oscillating followers and decreases the lifetime of the system.

The proposed formation control methods are designed to operate in an obstacle-free environment. In most cases, the environment is full of obstacles. For this reason, Reagan et al. [27] implemented a real-time obstacle avoiding algorithm using APF to control multiple quadcopters in order to reach a specific goal location with following the most feasible obstacle-free path. They tested this algorithm using simulations and on real quadcopters indoor with achieving very effective results in keeping the formation as well as reaching the goal. However, the main disadvantage of using APF is the necessity preliminary knowledge of robots' position, obstacles' position and target position where it also needs a lot of fine-tuning to achieve the best performance. Milad Nazarahari et al. [28] updated the use of APF. They used APF to find all feasible paths between the starting position to goal position, then they developed an Enhanced Genetic Algorithm (EGA) to improve these initial paths and

find the optimal path between the robot and goal. They combined path length, smoothness, and safety in order to achieve a multi-objective path-planning. Furthermore, Derek J. Bennet and Colin R. McInnes [29] used bifurcating potential fields to establish formation reconfigurability while maintaining robustness and to failures. They proved how various formation patterns can be autonomously established by simple free parameter change. They tested the proposed algorithm using simulations only.

There are a lot of fire detection algorithms and techniques used by scientists and engineers to help in early fire detection and monitoring processes using UAVs. For example, Chi Yuan et al. [11] did effectively extract and track fire pixels in an infrared video sequence received from a UAV. They used brightness and motion clues along with image processing histogram segmentation to extract hot object regions. They also used optical flow sensors to calculate motion vectors of these hot candidate regions. Another work done by Casbeer et al. [12], where they explored the feasibility of a short term, low altitude UAV team to cooperatively track and monitor forest fires' propagation. They simulated a full 6-DOF dynamic model of the UAVs and some numerical models for forest fire propagation. They did not test it in real fire situations. In addition, Zhou et al. [13] gathered video streams from a UAV and applied orthorectification method of these received images to monitor forest fires. They pointed out some specific problems which should be treated in case of forest fire detection and presented some primary solutions to these problems. However, there were no results presented. Moreover, Mubarak Mahmoud et al. [14] collected 6 videos available online and used image processing algorithms to detect fires. They first applied background subtraction to capture movements within the region detected. Then they converted the moving regions from RGB color space into YCbCr which helped them to apply 5 different fire detection rules in order to separate fire pixels.

And finally, they used temporal variation method to distinguish between fire and fire color objects. They achieved 96.63% accuracy detection rate. Another method proposed by Henry Cruz et al. [15] developed a method, which can be used on UAVs, called Forest Fire Detection Index (FFDI) based on using a new color index. This index is based on vegetation methods to detect flames and smoke. They tested this method upon a database imagery with acquiring very good results of about 96.82% precision accuracy over 960 x 540 pixels samples along with 0.0447 seconds processing speed.

1.5. THESIS OUTLINE

The rest of the report is divided as follows; Chapter 2 will include simulation and control of 3 quadcopters tracking a goal position with avoiding obstacles using APF repulsive forces. In addition to the mathematical modeling and control of a quadcopter drone. Chapter 3 will illustrate the components to be used. Chapter 4 will combine all parts together and will show the practical design procedure (technical aspect) of 3 quadcopters flying as a team with implementing Kalman Filter (KF) for better position estimation and will show the procedure followed to train a Neural Network (NN) in order to detect fire in camera frames and the design of th simple control system to drive a camera gimbal to keep tracking fire region. Chapter 5 will show the economical, ethical, and environmental impact of this project. Chapter 6 will show the results and chapter 7 will conclude the project.

CHAPTER 2. SYSTEM DESIGN

2.1. INTRODUCTION

In this chapter, a mathematical model, including the kinematics and dynamics, of a quadcopter UAV and the design of its PID-based attitude, altitude, and position controllers will be presented. Then, the simulation will be integrated to control 3 UAVs following leader-follower scheme. The simulation is done using MATLAB/Simulink environment in order to ensure the viability of the proposed controllers in real world application. Figure 2.1 illustrates the high-level control diagram of L-F scheme. The trajectory planner generates the desired leader's trajectory $\boldsymbol{r_d} = [x_d, y_d, z_d]^T$ and the desired course angle for every quadcopter in the team (ψ_d). Each quadcopter in the team is equipped with a local planner (position, altitude, attitude and obstacle avoidance controllers) that can achieve partial undependability from leader's decisions. However, the desired trajectory of each follower is directly dependent upon leader's position. The desired trajectory will be tracked by the leader quadcopter and the formation controller is responsible for maintaining the followers' relative desired distance from the leader (Δ_x, Δ_y). A main advantage of this method is the ease of implementation and expendability (i.e., "*n*" followers can be added easily) however it lacks full independence.

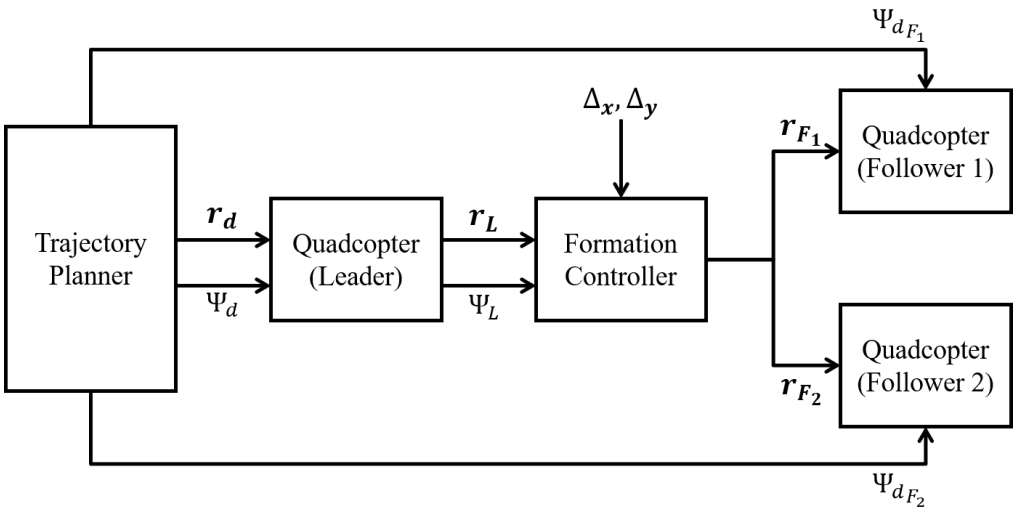

Figure 2.1. Leader-Follower high-level controller block diagram

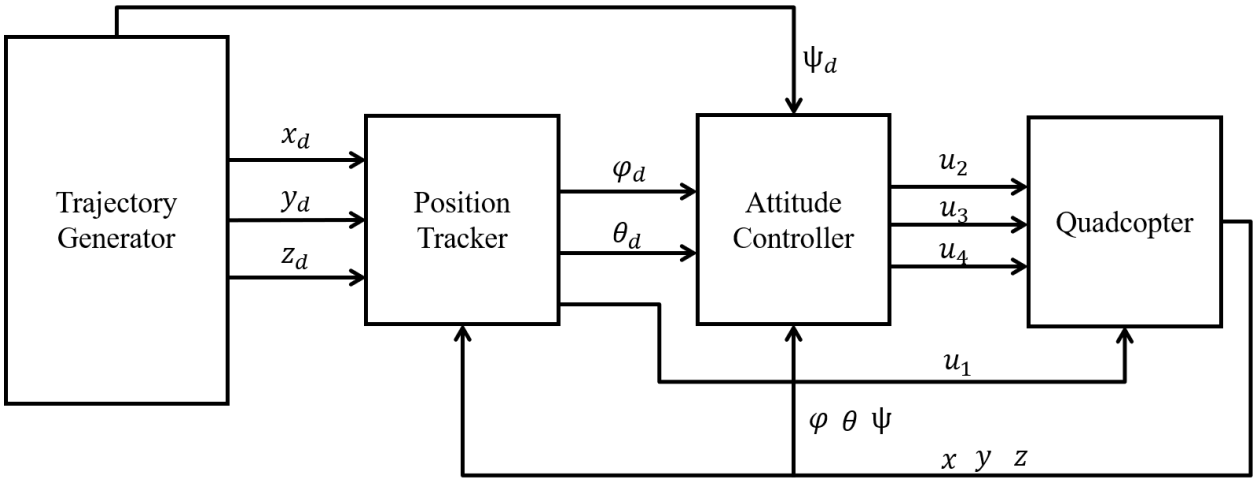

Figure 2.2: Quadcopter high-and low-level controllers

Figure 2.2 shows the overall control system structure. The quadcopter is equipped with a high-level (position) controller responsible for tracking a desired trajectory in (x_d, y_d, z_d), generated by the trajectory generator, and a low-level (attitude controller) controller responsible for tracking the rotational setpoints $(\varphi_d, \theta_d, \psi_d)$. The quadcopter then receives 4 control inputs (u_1, u_2, u_3, u_4) which control the throttle force, rolling, pitching, and yawing torques respectively.

The quadcopter produces rotational torques in order for it to navigate in space. These rotation angles can be described as Euler angles in 3D. The rotations about x-axis, y-axis and

z-axis are represented as rolling (φ), pitching (θ), and yawing (ψ) respectively as shown in Figure 2.3.

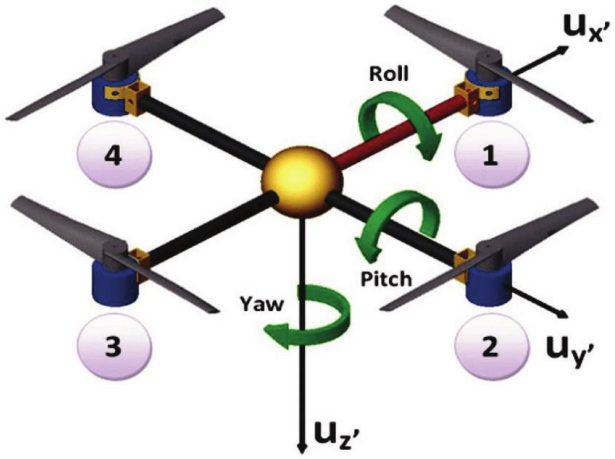

Figure 2.3. Quadcopter rotation angles

2.2. PRELIMINARIES AND ASSUMPTIONS

To achieve the desired coordination between several quadcopters, some basic information and constraints should be known.

- The rotation angles about $[x, y, z]$ axes are represented by Euler angles, roll, pitch and yaw $[\varphi, \theta, \psi]$ respectively with constraining $-25 < \varphi < 25^o$, $-25^o < \theta < 25^o$ (for linear region) and $-180^o < \psi < 180^o$ as shown in Figure 2.4.
- The quadcopter is assumed to be a rigid body of mass "*m*" and an inertia matrix "***J***". Also, the propellers are also assumed to be rigid with neglecting flapping effect.
- The quadcopter is symmetrical and its center of mass is exactly located at the center of the rigid body.
- Motor inertia is assumed to be neglected.
- Flat earth is assumed.

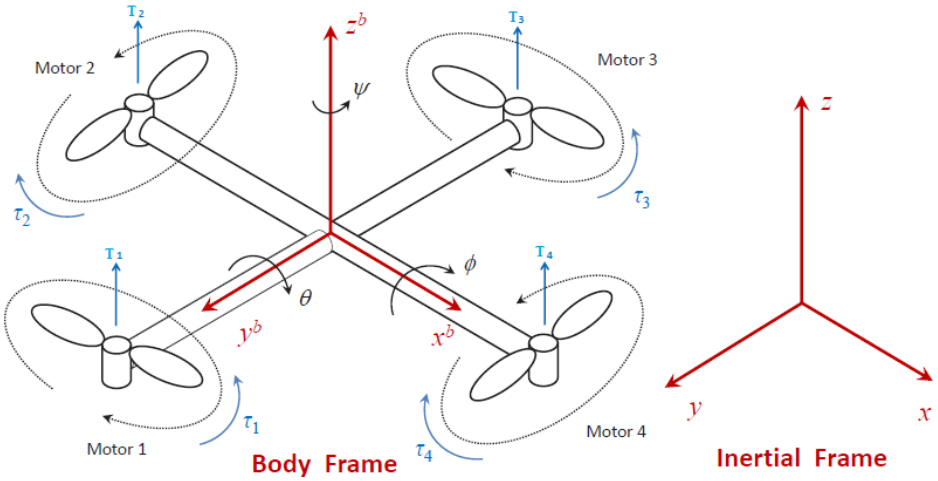

Figure 2.4. Quadcopter rotation angles and coordinate frames

2.3. KINEMATIC MODEL

Translations and rotations of a quadcopter are represented in 2 different frames as in Figure 2.4; Body fixed frame B and inertial frame I. Generally, a quadcopter is equipped with an Inertial Measurement Unit (IMU) which senses quantity relative to its body frame but on the other hand a Global Positioning System (GPS) returns the quadcopter's position in inertial frame. Also, the system actuator inputs are relative to body frame. However, and because most of quadcopter missions are handled in the fixed inertial frame, it is necessary to establish a conversion concept that translates rotations and translations from body to inertial frames and vice versa.

As mentioned before, the attitude of the quadcopter is represented by Euler angles, with phi, theta and psi, defined by $\boldsymbol{\phi} = [\varphi, \theta, \psi]^T$, being the rotations about the inertial x-axis, y-axis and z-axis respectively. The position of the quadcopter in the inertial frame is represented by $\boldsymbol{r} = \left[x^i, y^i, z^i\right]^T$, the angular velocities with respect to body frame $\boldsymbol{\nu} = [p, q, r]^T$ and $\boldsymbol{V_B} = \left[v_x^b, v_y^b, v_z^b\right]^T$ represents the linear velocity components in the body frame.

There are several ways to calculate the direction cosine matrix which establishes the conversion between these two frames. The rotation matrix from inertial frame to body frame using Euler angles can be seen in equation (2.1). Derivation process can be found here [30].

$$R_{inertial}^{body} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\psi) \\ \sin(\varphi)\sin(\theta)\cos(\psi) - \cos(\varphi)\sin(\psi) & \sin(\varphi)\sin(\theta)\sin(\psi) + \cos(\varphi)\cos(\psi) & \sin(\varphi)\cos(\theta) \\ \cos(\varphi)\sin(\theta)\cos(\psi) + \sin(\varphi)\sin(\psi) & \cos(\varphi)\sin(\theta)\sin(\psi) - \sin(\varphi)\cos(\psi) & \cos(\varphi)\cos(\theta) \end{bmatrix} \quad (2.1)$$

The transformation from body frame to inertial frame can be obtained by inverting the matrix in equation (2.1), and because it is orthonormal matrix, its inverse is simply the transpose of it.

$$R_{body}^{inertial} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\varphi)\sin(\theta)\cos(\psi) - \cos(\varphi)\sin(\psi) & \cos(\varphi)\sin(\theta)\cos(\psi) + \sin(\varphi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\varphi)\sin(\theta)\sin(\psi) + \cos(\varphi)\cos(\psi) & \cos(\varphi)\sin(\theta)\sin(\psi) - \sin(\varphi)\cos(\psi) \\ -\sin(\psi) & \sin(\varphi)\cos(\theta) & \cos(\varphi)\cos(\theta) \end{bmatrix} \quad (2.2)$$

Therefore,

$$\begin{bmatrix} x^i \\ y^i \\ z^i \end{bmatrix} = R_{body}^{inertial} \begin{bmatrix} x^b \\ y^b \\ z^b \end{bmatrix} \quad (2.3)$$

Similarly, Euler rates are used to determine the attitude of the quadcopter in the inertial frame. Thus, the relation between Euler rates and body angular rates is calculated as follows [31]:

$$\boldsymbol{T}_{inertial}^{body} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\varphi) & \sin(\varphi)\cos(\theta) \\ 0 & -\sin(\varphi) & \cos(\varphi)\cos(\theta) \end{bmatrix} \quad (2.4)$$

As a result, the Euler rates in the inertial frame can be obtained as:

$$\dot{\boldsymbol{\phi}} = \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T^{-1}{}_{inertial}^{body} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.5}$$

With,

$$\boldsymbol{T}^{-1}{}_{inertial}^{body} = \boldsymbol{T}_{body}^{inertial} = \begin{bmatrix} 1 & \tan(\theta)\sin(\varphi) & \tan(\theta)\cos(\varphi) \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi)\sec(\theta) & \cos(\varphi)\sec(\theta) \end{bmatrix} \tag{2.6}$$

2.4. DYNAMICAL MODEL

The dynamical study studies the effect of the external forces applied on the quadcopter and how these forces affect its behavior (translational and rotational). It is worth of mentioning that the quadcopter is an under-actuated system which has 4 input parameters with 6 DOF, in other words, it has 6 outputs with only 4 inputs. The dynamical model of the quadcopter can be calculated from Newton's second law where the summation of all external forces acting on the quadcopter equals to its mass multiplied by its linear acceleration. Also, the summation of all torques equals moment of inertia multiplied by angular acceleration. The translational and rotational models, given in the inertial frame, are represented as [32] [33] [34] [35] [36]:

$$\begin{aligned} \ddot{x} &= \frac{1}{m}[(\cos(\varphi)\sin(\theta)\cos(\psi) + \sin(\varphi)\sin(\psi))u_1 - k_{fx}\dot{x}] \\ \ddot{y} &= \frac{1}{m}[(\cos(\varphi)\sin(\theta)\sin(\psi) - \sin(\varphi)\cos(\psi))u_1 - k_{fy}\dot{y}] \\ \ddot{z} &= \frac{1}{m}\left[(\cos(\varphi)\cos(\theta))u_1 - k_{fz}\dot{z}\right] - g \end{aligned} \tag{2.7}$$

$$\ddot{\varphi} = \frac{1}{I_{xx}}\left[\dot{\theta}\dot{\psi}\left(I_{yy} - I_{zz}\right) - J_{tp}\bar{\Omega}\dot{\theta} + u_2\right] \tag{2.8}$$

$$\ddot{\theta} = \frac{1}{I_{yy}}\left[\dot{\varphi}\dot{\psi}(I_{zz} - I_{xx}) + J_{tp}\bar{\Omega}\dot{\varphi} + u_3\right]$$

$$\ddot{\Psi} = \frac{1}{I_{zz}}\left[\dot{\theta}\dot{\varphi}\left(I_{xx} - I_{yy}\right) + u_4\right]$$

With, *m* is the total mass of the quadcopter and *g* is the gravity force in the negative z-direction.$[I_{xx}, I_{yy}, I_{zz}]$ are the moment of inertia about each axis. $[k_{fx}, k_{fy}, k_{fz}]$ are drag constants resulted from aerodynamical effects. J_{tp} is the total rotational moment of inertia around the propeller axis and $\bar{\Omega} = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$ is the total gyroscopic torque affecting the quadcopter.

The inputs of the quadcopter (u_1, u_2, u_3, u_4) can be written as:

$$\begin{bmatrix} Throttle \\ Rolling \\ Pitching \\ Yawing \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ -lb & 0 & lb & 0 \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{2.9}$$

Where, *b* is a thrust coefficient characterized by the physical properties of the propellers, *l* is the arm length, *d* is a drag constant and Ω_i (for i = 1,2,3,4) is the rotor speed. Equation (2.9) is used to map the desired motors' speeds with the desired control actions. Figure 2.5 shows the translational dynamics, shown in equation (2.7), and Figure 2.6 shows the rotational dynamics, shown in equation (2.8), implemented in Simulink.

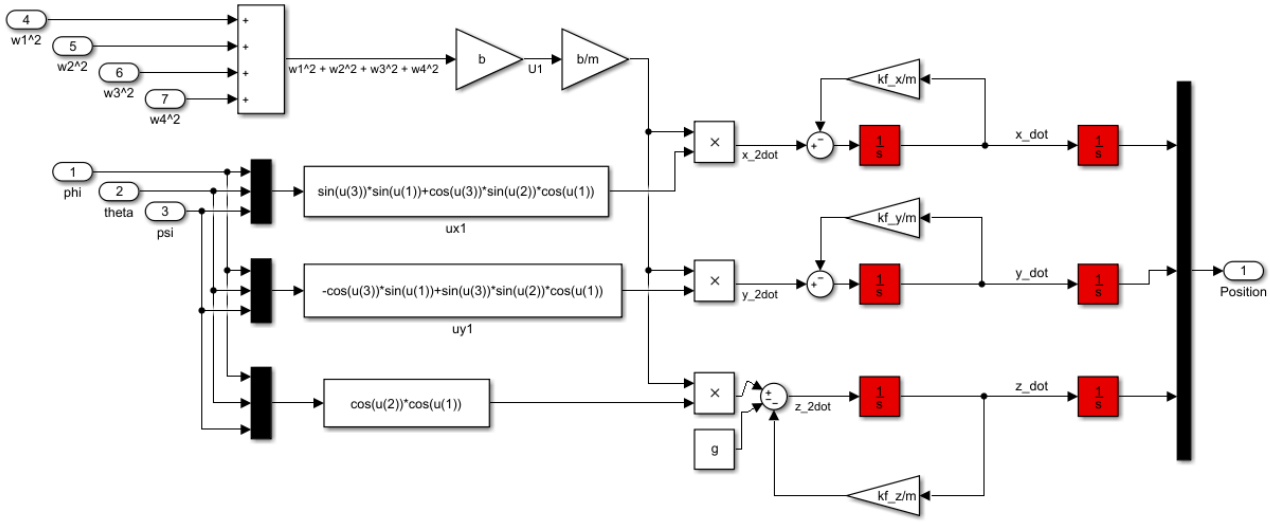

Figure 2.5. Translational dynamics Simulink model

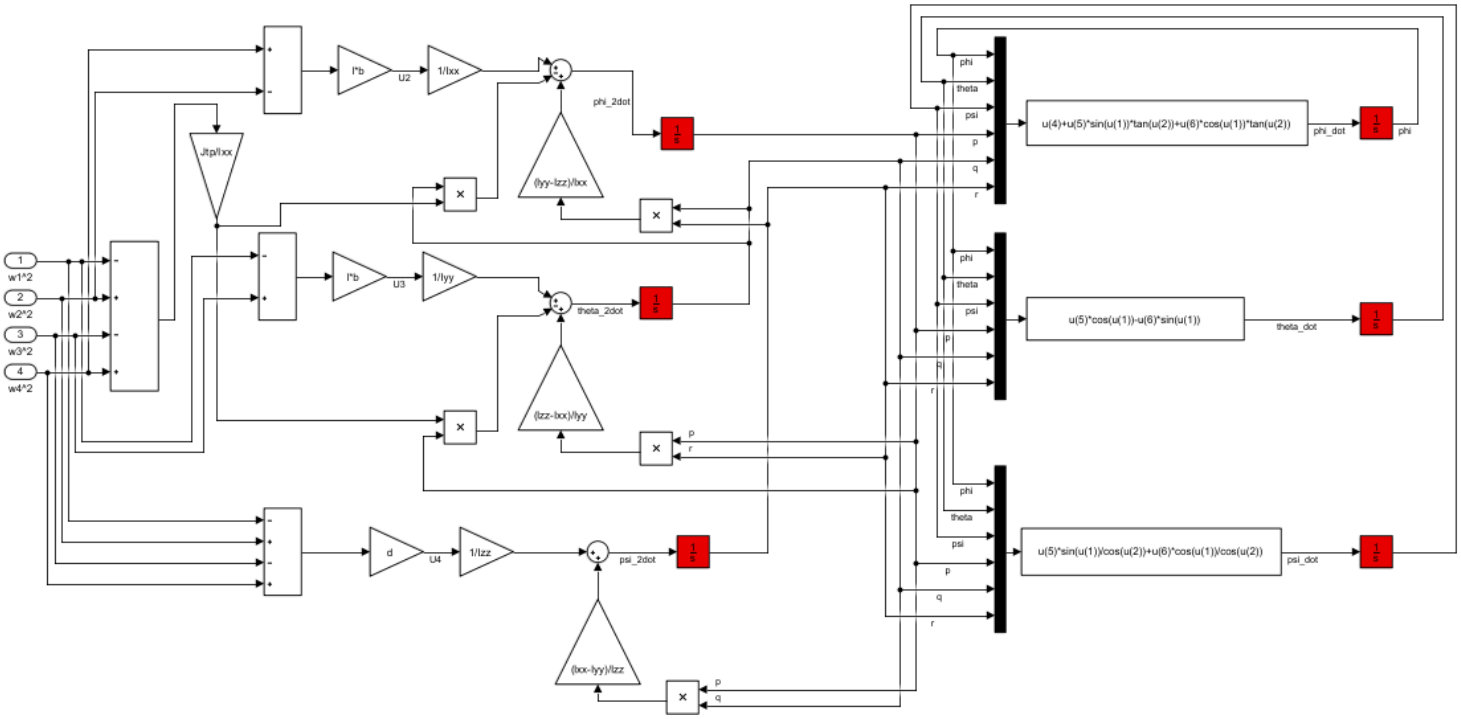

Figure 2.6. Rotational dynamics Simulink model

The parameters for the dynamical model are summarized in Table 2.1. Equations for these physical quantities are clearly presented in Appendix A (A.1).

Table 2.1. Physical parameters

Parameter	Description	Value	Unit
$\boldsymbol{m}$	Mass of the quadcopter	2	kg
$\boldsymbol{I_{xx}}$	Moment of inertia about x-axis	0.0641	$kg.m^2$
$\boldsymbol{I_{yy}}$	Moment of inertia about y-axis	0.0641	$kg.m^2$
$\boldsymbol{I_{zz}}$	Moment of inertia about z-axis	0.1148	$kg.m^2$
$\boldsymbol{g}$	Gravity force	9.81	$m.s^{2^{-1}}$
$\boldsymbol{b}$	Thrust constant	1.02×10^{-6}	$N.m.s^2$
$\boldsymbol{d}$	Drag constant	1.3×10^{-7}	$N.m^{-1}$
$\boldsymbol{l}$	Quadcopter arm length	0.275	m
$\boldsymbol{J_{tp}}$	Total rotational moment of inertia around the propeller axis	104×10^{-6}	$kg.m^2$
$\boldsymbol{k_{fx}}$	x-axis drag constant	0.00215	$N.m^{-1}$
$\boldsymbol{k_{fy}}$	y-axis drag constant	0.00215	$N.m^{-1}$
$\boldsymbol{k_{fz}}$	z-axis drag constant	0.00215	$N.m^{-1}$

2.5. LINEAR MODEL

Linearizing the quadcopter equations of motion is crucial in order to design linear controllers like PID. Linearizing a system can help designers better predict a system's state and control it using easy linear controller tools. The general form of a non-linear system can be written as:

$$\dot{x} = f(x, u) \tag{2.10}$$

Since the quadcopter is a highly non-linear system with strong coupling between states, it is very hard to find a predictable state solution at time "*t*". Thus, linearization is made about an equilibrium point such that:

$$\hat{f}(\bar{x}, \bar{u}) = 0 \tag{2.11}$$

Where $\bar{x}$ is the state at equilibrium point and $\bar{u}$ is the equilibrium input in which when applied, the system stays at equilibrium with all of its state derivatives equal to 0. The

equilibrium point chosen is the hover point (small oscillations model) which has the following characteristics:

- $\varphi \approx \theta \approx 0$
- $\cos(\varphi) = \cos(\theta) = 1$
- $\sin(\varphi) = \varphi \; and \sin(\theta) = \theta$
- $\dot{\boldsymbol{\phi}} = \boldsymbol{v}$
- $[u_1, u_2, u_3, u_4] = [mg, 0,0,0]$

Therefore, equations (2.7) and (2.8), which represent the non-linear dynamics of the quadcopter, are realized to become:

$$\begin{aligned} \ddot{x} &= g\,[(\theta\cos(\psi) + \varphi\sin(\psi))] \\ \ddot{y} &= g\,[(\theta sin(\psi) - \varphi\cos(\psi))] \\ \ddot{z} &= \frac{1}{m}u_1 \end{aligned} \tag{2.12}$$

$$\begin{aligned} \ddot{\varphi} &= \frac{u_2}{I_{xx}} \\ \ddot{\theta} &= \frac{u_3}{I_{yy}} \\ \ddot{\Psi} &= \frac{u_4}{I_{zz}} \end{aligned} \tag{2.13}$$

2.6. CONTROLLERS

The objective of this thesis is to form a cooperative flight between 3 quadcopters tracking a predefined trajectory without colliding. So first, and to achieve full autonomy, the control strategy is to design robust and independent altitude, attitude, and position PID-based controllers for each quadcopter and then implement a formation control algorithm to achieve swarming (see Figure 2.1). The detailed closed loop control diagram for a single quadcopter is shown in Figure 2.7.

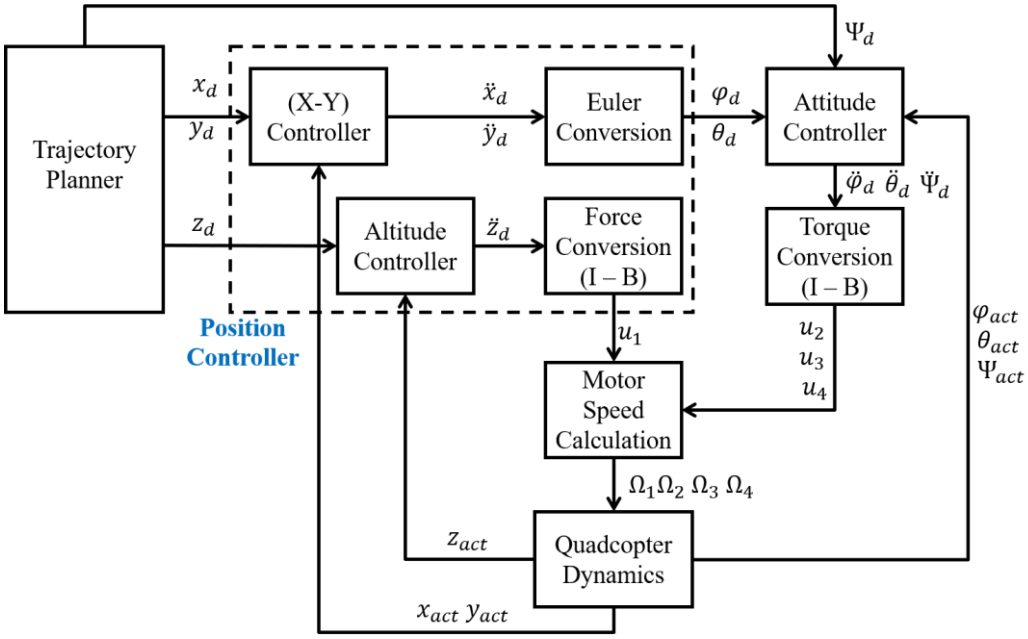

Figure 2.7. Single quadcopter control system block diagram

2.6.1. PID Controller

The Proportional, Derivative and Integral (PID) controller is a linear-type controller used to stabilize a system with respect to a given state setpoint. This controller is used for its efficiency, simplicity, and ease of implementation in real-world applications. However, PID is limited in linear regions only and can withstand only very narrow range of unknown disturbances acting on the system. Consider a controlled variable (distance, velocity, angle or temperature etc.), state error (difference between desired state and actual state) defined in time domain $e(t) = x_d - x_{actual}$, the PID controller output is calculated as:

$$u(t) = K_{p_x} e(t) + K_{i_x} \int e(t)dt + K_{d_x} \frac{de(t)}{dt} \tag{2.14}$$

and in frequency domain (Laplace):

$$U(s) = K_{p_x} E(s) + \frac{K_{i_x}}{s} E(s) + sK_{d_x} E(s) \tag{2.15}$$

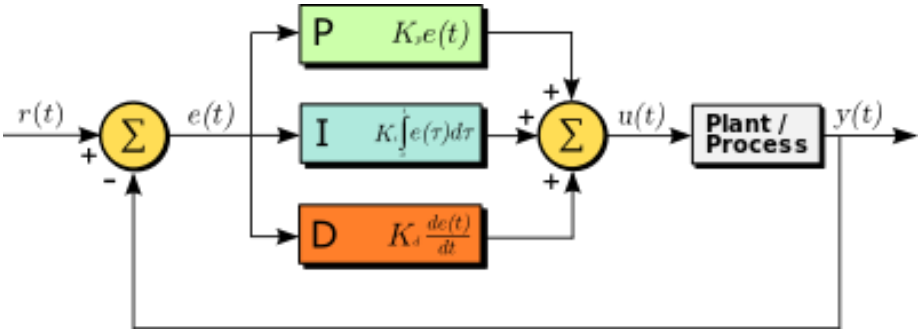

Figure 2.8. PID controller block diagram

Where, $[K_{p_x}, K_{i_x}, K_{d_x}]$ are positive the proportional, integral and derivative gains respectively. x is the controlled variable. Changing these gains will result in completely different system behavior. Thus, these gains should be tuned according to the desired system behavior chosen by the designer. Tuning these gains in real systems is difficult and has no direct rule. However, the following guidelines for tuning these values can be beneficial most of the times:

- Proportional action reacts proportionally with the error. Increasing K_p speeds up the system response but decreases stability.
- Integral action accumulates the error signal over time in order to remove the steady state error. Increasing K_i removes steady state error but it does also increase the oscillations and decreases stability accordingly.
- Derivative action damps the system when approaching the setpoint. Increasing K_d will surely increase the stability but it slows down the response.
- Set $K_i = K_d = 0$ and keep increasing K_p until the system starts oscillating. At that point, multiply K_p by 0.6 and stick to that value.
- Similarly, with fixed K_p, start increasing K_d until the system starts oscillating, then divide K_d by 2 and stick to that value.

- For K_i, and because it is very sensitive, increase it by a step of "0.001" each time and observe the behavior of the system. When it starts oscillating, divide K_i by 2 and stick to that value.

2.6.2. Position Controller

The goal of the position controller is to track a predefined trajectory or reach a fixed coordinate in the inertial frame (x^i, y^i, z^i). To do this, this PID-based controller is required to transform the error signals into roll and pitch commands since translational movements are directly coupled with rotational movements (equation (2.12)). For example, to reach a 2D goal point located at [10,0] with fixed altitude and 0 degree heading angle (ψ = 0), the quadcopter should produce pitching torque only but if the goal point is at [0,10] the quadcopter should roll. This conversion function is held by "Euler Conversion" block which transforms the desired 2D accelerations with respect to inertial frame into desired attitude. From the linear model presented in equation ((2.12), the desired roll and pitch angles can be calculated as:

$$\begin{aligned} \varphi_d &= \frac{1}{g}[\ddot{x}_d \sin(\psi_{des}) - \ddot{y}_d \cos(\psi_{des})] \\ \theta_d &= \frac{1}{g}[\ddot{x}_d \cos(\psi_{des}) + \ddot{y}_d \sin(\psi_{des})] \end{aligned} \tag{2.16}$$

With $\ddot{x}_d$ and $\ddot{y}_d$ are the (X-Y) PID position controller output and are given by:

$$\begin{aligned} e_x &= x_d - x_{actual} \\ e_y &= y_d - y_{actual} \end{aligned} \tag{2.17}$$

$$\ddot{x}_d = k_{p_{position}} e_x + k_{i_{position}} \int e_x \, dt + k_{d_{position}} \dot{e}_x \tag{2.18}$$

$$\ddot{y}_d = k_{p_{position}} e_y + k_{i_{position}} \int e_y \, dt + k_{d_{position}} \dot{e}_y$$

Similarly, the altitude controller needs to transform its desired vertical acceleration from inertial frame to body frame (using rotation matrix in equation ((2.1) in order to calculate the desired throttling input (u_1) in body frame. This is done using "Force Conversion (I – B)" block. From equation (2.7), the desired throttle input can be calculated as:

$$u_1^d = \frac{m}{\cos(\varphi)\cos(\theta)} \ddot{z}_d \tag{2.19}$$

With $\ddot{z}_d$ being the output of altitude PID controller and is given by equation (2.20). It is worth of mentioning that when the altitude error is 0, the controller should keep a bias of 1g force in order to maintain hovering. The position controller implemented in Simulink is shown in Figure 2.9.

$$\ddot{z}_d = k_p e_z + K_i \int e_z \, dt + K_d \dot{e}_z + g \tag{2.20}$$

Where, $e_z = z_d - z_{actual}$ is the altitude error.

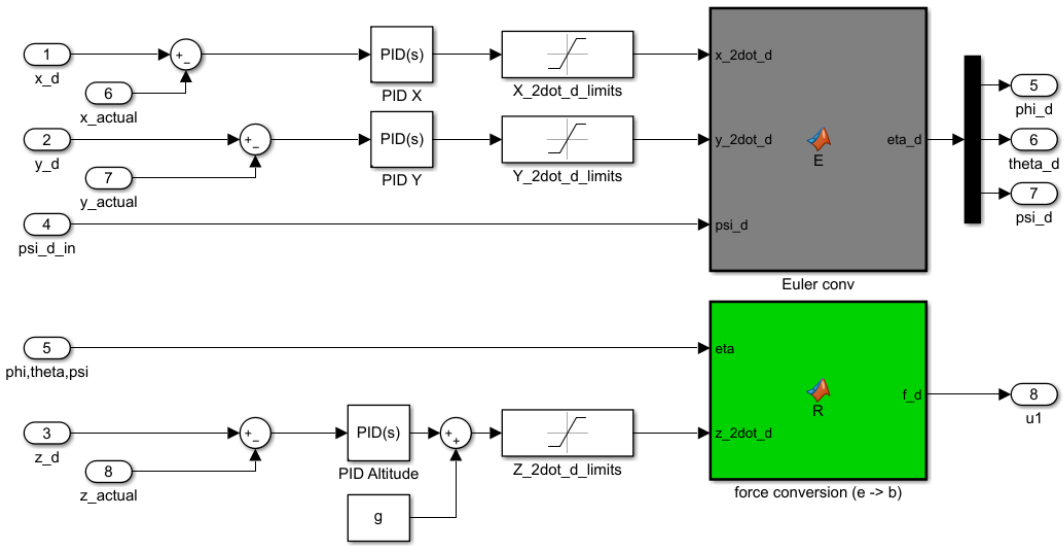

Figure 2.9. Position controller Simulink model

2.6.3. Obstacle Avoidance

For the safety of each quadcopter, an obstacle avoidance algorithm should be implemented. This algorithm will directly affect the high-level commands of the position controller ($\ddot{x}_d, \ddot{y}_d$). Figure 2.10 shows the position controller equipped with Obstacle Avoidance (OA) algorithm which is based on APF repulsive forces.

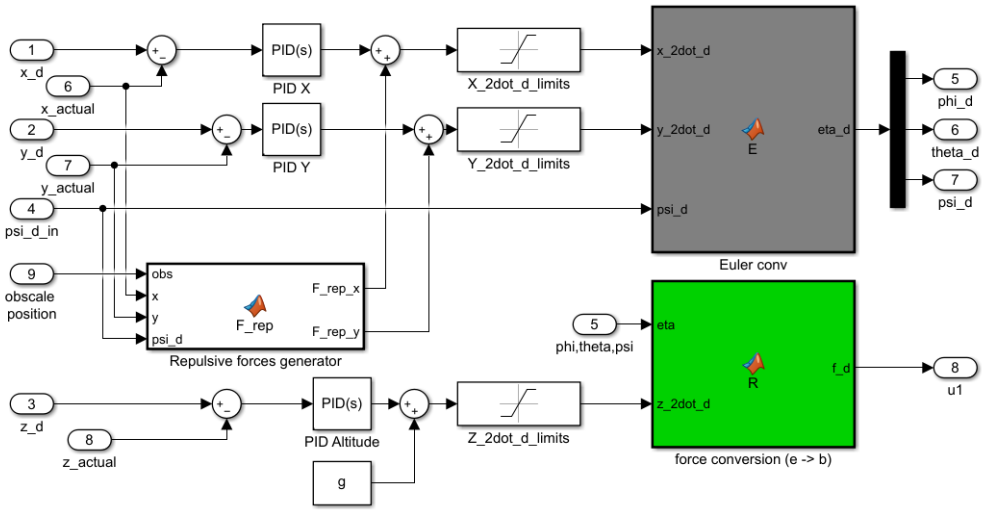

Figure 2.10. Position controller with obstacle avoidance algorithm

These repulsive forces are generated in a way to keep each quadcopter away from obstacles. Also, these forces are characterized by intense strength when the quadcopter is near any obstacle and have decreasing influence when it is far from obstacles. There are many forms of repulsive force equations, one possible repulsive force generated from an obstacle "i" is as [37]:

$$U_{rep_i}(q_0) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{d_{obst_i}(q_0)} - \frac{1}{d_0} \right)^2 & , if\ d_{obst_i}(q_0) \leq d_0 \\ 0 & , if\ d_{obst_i}(q_0) > d_0 \end{cases} \qquad (2.21)$$

With $d_{obst_i}(q_0)$ is the Euclidean distance between the UAV and obstacle "i". k_{rep} is a scaling factor used to scale the repulsive force intensity according to designer's choice and d_0 is a safe distance chosen by the designer, the repulsive force has no effect if the distance to

the obstacle is greater than this value. Figure 2.11 illustrates the effect of repulsion when a quadcopter's 2D position varies from $(x, y) = (-20{:}\,100, -20{:}\,100)$ and obstacle positions of $(20,20)$, $(15,50)$ and $(60,80)$. Repulsive forces can be simply by considering the quadcopter as a ball rolling in space. When this quadcopter reaches an obstacle, the slope of the environment will prevent the ball from approaching the obstacle. It can be seen clearly that when a quadcopter approaches an obstacle, the repulsive force goes very intense, while it has completely no effect when the quadcopter is far away.

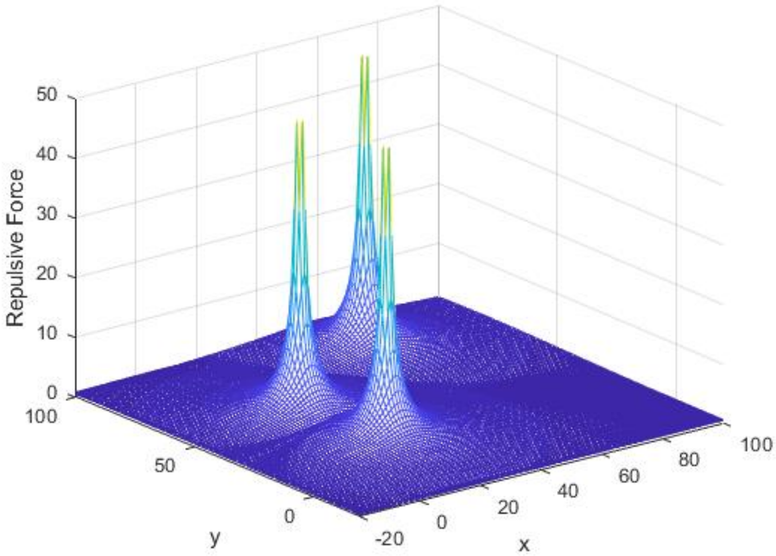

Figure 2.11. Repulsive field

Hence, the gradient of this repulsive field will attract the quadcopter towards the obstacle. So, to repel the quadcopter from obstacle, it is crucial to apply the negative gradient upon the robot. The negative gradient of this repulsive field, $F_{rep_i}(q_0) = -\nabla U_{rep_i}(q_0)$, is given by:

$$F_{rep_i}(q_0) = \begin{cases} k_{rep}\left(\dfrac{1}{d_{obst_i}(q_0)} - \dfrac{1}{d_0}\right)\dfrac{1}{d^2_{obst_i}(q_0)}\hat{e}_i \,, if\ d_{obst_i}(q_0) \le d_0 \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad , if\ d_{obst_i}(q_0) > d_0 \end{cases} \qquad (2.22)$$

Where, $\hat{e}_i = \frac{\partial d_{obst_i}(q_0)}{\partial(q_0)}$ is said to be a unit vector to indicate the direction of the repulsive force. This repulsive force vector can be considered as an external force acting on the quadcopter, thus it can be considered proportional to the acceleration of the quadcopter in its body frame, the desired acceleration in the inertial frame can then be calculated as:

$$\begin{aligned} \ddot{x}_d{}^{Rep} &= F_{rep_x} \sin(\psi_d) - F_{rep_y} \cos(\psi_d) \\ \ddot{y}_d{}^{Rep} &= F_{rep_x} \cos(\psi_d) + F_{rep_y} \sin(\psi_d) \end{aligned} \tag{2.23}$$

And therefore, the total acceleration acting on the quadcopter is (see Figure 2.10):

$$\begin{aligned} \ddot{x}_d &= \ddot{x}_d{}^{PID} + \ddot{x}_d{}^{Rep} \\ \ddot{y}_d &= \ddot{y}_d{}^{PID} + \ddot{y}_d{}^{Rep} \end{aligned} \tag{2.24}$$

From equation (2.24), it can be seen that the position PID controller is acting as an attractive force guiding the quadcopter towards the goal position, and the APF repulsive forces guide the quadcopter away from obstacles. The total desired acceleration determines the required roll and pitch angles.

2.6.4. Attitude Controller

Attitude controller is responsible to track the desired $(\varphi_d, \theta_d, \psi_d)$ generated from the position controller in order to achieve the desired translation in space (inertial frame). Since the attitude or rotational system is responsible for all translational and rotational movements, its stability is crucial to ensure the whole system's stability. From equation ((2.13), and from their Laplace transform, it can be seen that all 3 angular acceleration linear dynamics are marginally stable with 2 poles at the *(0,0)* of the s-plane.

$$\frac{\varphi(s)}{U_2(s)} = \frac{1}{s^2 I_{xx}} \tag{2.25}$$

$$\frac{\theta(s)}{U_2(s)} = \frac{1}{s^2 I_{yy}}$$

$$\frac{\psi(s)}{U_2(s)} = \frac{1}{s^2 I_{zz}}$$

From equation (2.25), the root locus plot can be illustrated as Figure 2.12. Root locus plot varies a proportional gain (*K*) from 0 to infinity and shows all probabilities of closed-loop poles. However, using only a proportional gain will not stabilize these systems because this type of controllers does not shift the location of poles. Instead, it is required to use a derivative controller (D-controller) to increase the stability of the system by adding a "zero" at the numerator of each system and a very small (I-controller) action to remove steady state errors. For this project, tuning PID gains was done experimentally until achieving non-oscillatory response with acceptable rise time.

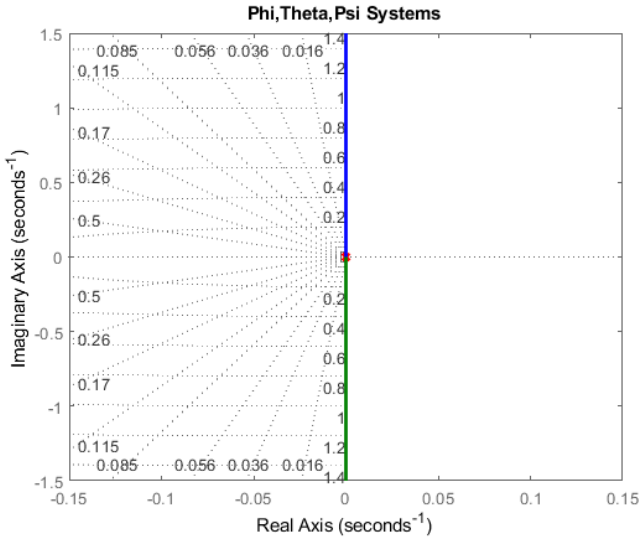

Figure 2.12. Root locus plot for roll, pitch, and yaw systems

Similar to translational system, from equations (2.13) and (2.4), the desired rolling, pitching and yawing torques in the body frame can be calculated as:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \boldsymbol{T}_{inertial}^{body} \begin{bmatrix} \ddot{\varphi}_d \\ \ddot{\theta}_d \\ \ddot{\Psi}_d \end{bmatrix} \tag{2.26}$$

With the desired angular accelerations being the output of the attitude PID controllers and are calculated using equations (2.27) and (2.28).

$$\begin{aligned} e_\varphi &= \varphi_d - \varphi_{actual} \\ e_\theta &= \theta_d - \theta_{actual} \\ e_\psi &= \psi_d - \psi_{actual} \end{aligned} \tag{2.27}$$

$$\begin{bmatrix} \ddot{\varphi}_d \\ \ddot{\theta}_d \\ \ddot{\Psi}_d \end{bmatrix} = k_{p_{angle}} \begin{bmatrix} e_\varphi \\ e_\theta \\ e_\psi \end{bmatrix} + k_{i_{angle}} \begin{bmatrix} \int e_\varphi dt \\ \int e_\theta \, dt \\ \int e_\psi \, dt \end{bmatrix} + k_{d_{angle}} \begin{bmatrix} \dot{e_\varphi} \\ \dot{e_\theta} \\ \dot{e_\psi} \end{bmatrix} \tag{2.28}$$

Figure 2.13 shows the attitude PID controller with “torque conversion” block implemented in Simulink.

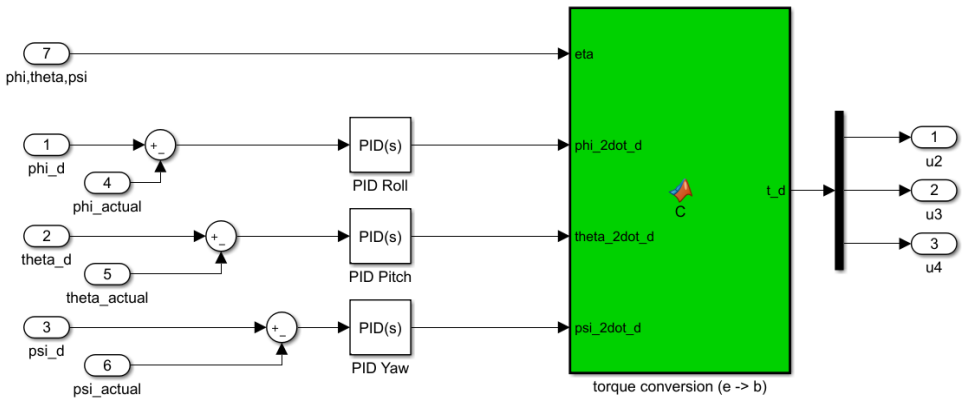

Figure 2.13. Attitude controller implemented in Simulink

The responses for roll, pitch and yaw systems subject to step input are shown in Figure 2.14 and Figure 2.15. The tuned PID parameters were $\left[k_{p_{angle}}, k_{i_{angle}}, k_{d_{angle}}\right] = [5.3, 0.8, 28]$. It is clear that these systems are stable with rise time of approximately 0.05 seconds and 0% overshoot with very small and acceptable steady state error.

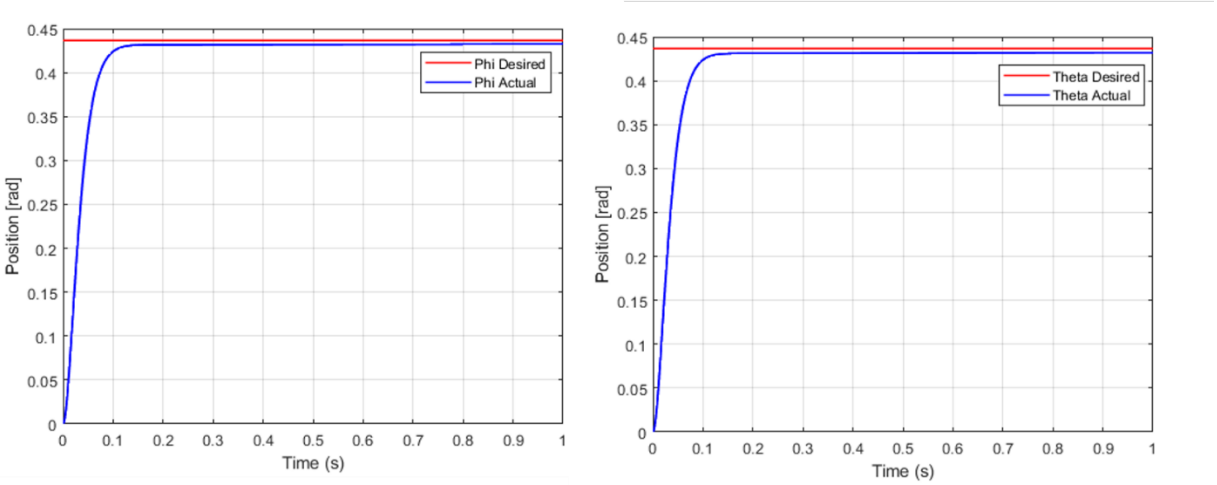

Figure 2.14. Roll and pitch response to step angle of 25 degrees

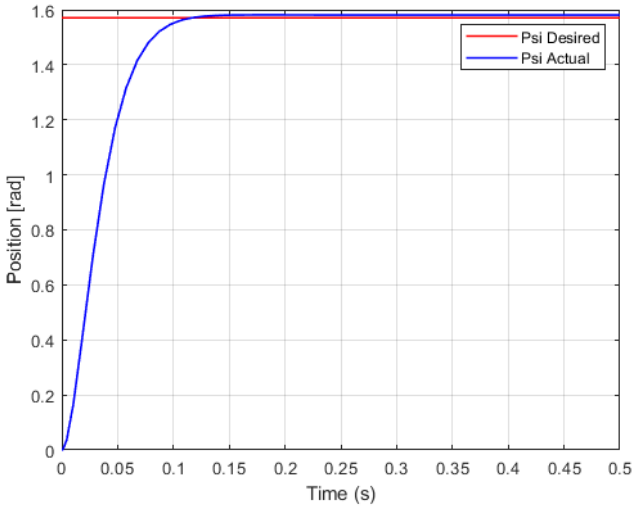

Figure 2.15. Yaw response to step angle of 90 degrees

2.6.5. Formation Controller

After designing independent controllers for each quadcopter, a formation algorithm, which keeps the formation of the swarm in shape, is presented. From Figure 2.16, $(\lambda_x \; or \; \Delta_x)$

and $(\lambda_y \; or \; \Delta_y)$ represent the x and y components of the desired separation distance λ in the leader's body frame and can be calculated as:

$$\Delta_x = -(x_L - x_F)\cos(\psi_L) - (y_L - y_F)\sin(\psi_L)$$
$$\Delta_y = (x_L - x_F)\sin(\psi_L) - (y_L - y_F)\cos(\psi_L) \tag{2.29}$$

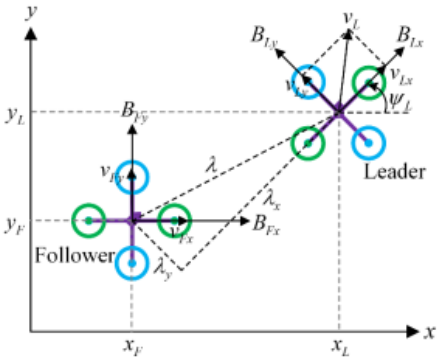

Figure 2.16. Formation in 2D

Where (x_L, y_L) and (x_F, y_F) represent the leader's and followers' position in inertial frame respectively. From equation (2.29), the desired followers' position in inertial frame can be calculated as:

$$x_F = x_L + \Delta_x \cos(\psi_L) - \Delta_y \sin(\psi_L)$$
$$y_F = y_L + \Delta_x \sin(\psi_L) + \Delta_y \cos(\psi_L) \tag{2.30}$$

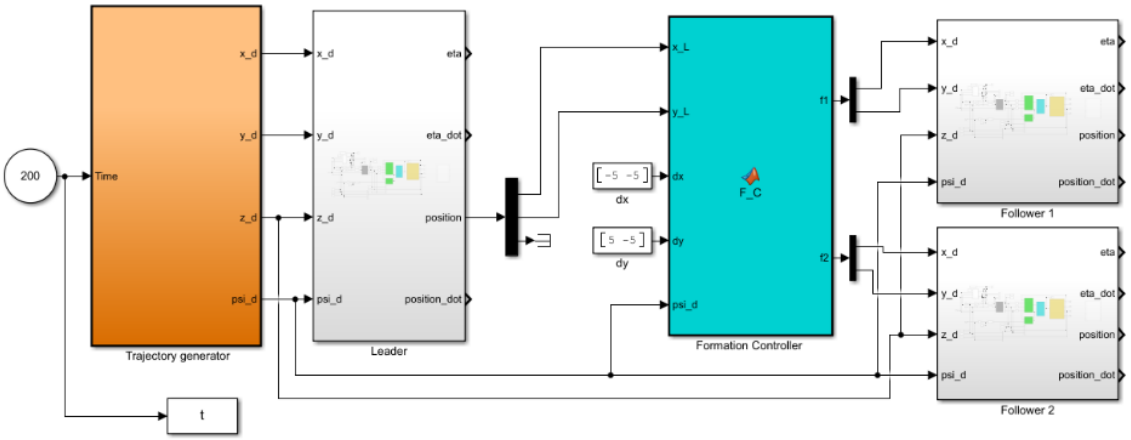

Figure 2.17. L-F formation control Simulink model

The simulation was set to 200 seconds. Position initial conditions are set to $[0,0,0]$ for leader quadcopter and $[-5,0,0]$, $[5,0,0]$ for followers 1 and 2 respectively. The trajectory generator was set to produce the following setpoints:

$$\boldsymbol{r_d} = \begin{cases} x_d = 0,\ y_d = 0,\ z_d = 10, & if\ t < 30 \\ x_d = 40,\ y_d = 0,\ z_d = 10, & if\ 30 < t < 120 \\ x_d = 80,\ y_d = 80,\ z_d = 10, & if\ t > 120 \end{cases} \tag{2.31}$$

Since these quadcopters are responsible for flying over forest regions, the required tracking speed is chosen to be low ($\leq 5\frac{m}{s}$). So, the best tuned PID position constants, repulsive force constant, and distance of influence d_0 for obstacle avoidance are found to be:

Table 2.2. PID position constants

	Leader			**Followers**		
	$\boldsymbol{k_p}$	$\boldsymbol{k_i}$	$\boldsymbol{k_d}$	$\boldsymbol{k_p}$	$\boldsymbol{k_i}$	$\boldsymbol{k_d}$
	5	0	20	15	0	20
$\boldsymbol{k_{rep}}$	20					
$\boldsymbol{d_0}$	$2\ m$					

This difference between leader's and followers' k_p gains has been established by trial and error. It is required that the followers follow the leader instantaneously when leader's position changes quickly from one point to another. For this reason, tracking speed for followers should be at least 2 times the tracking speed of leader.

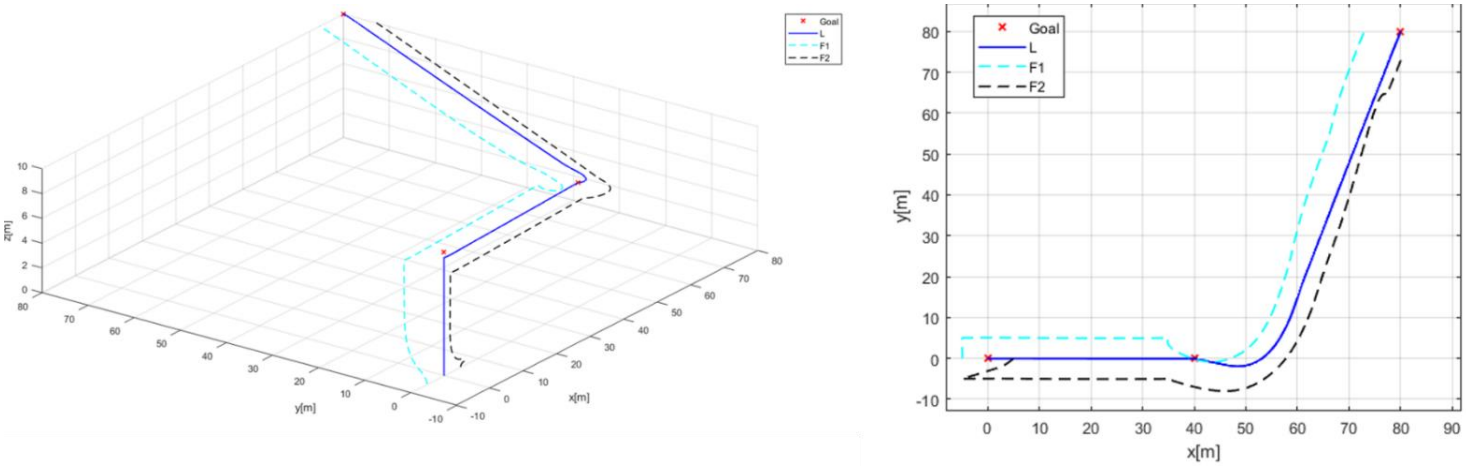

Figure 2.18. 3D and 2D path-tracking without obstacles

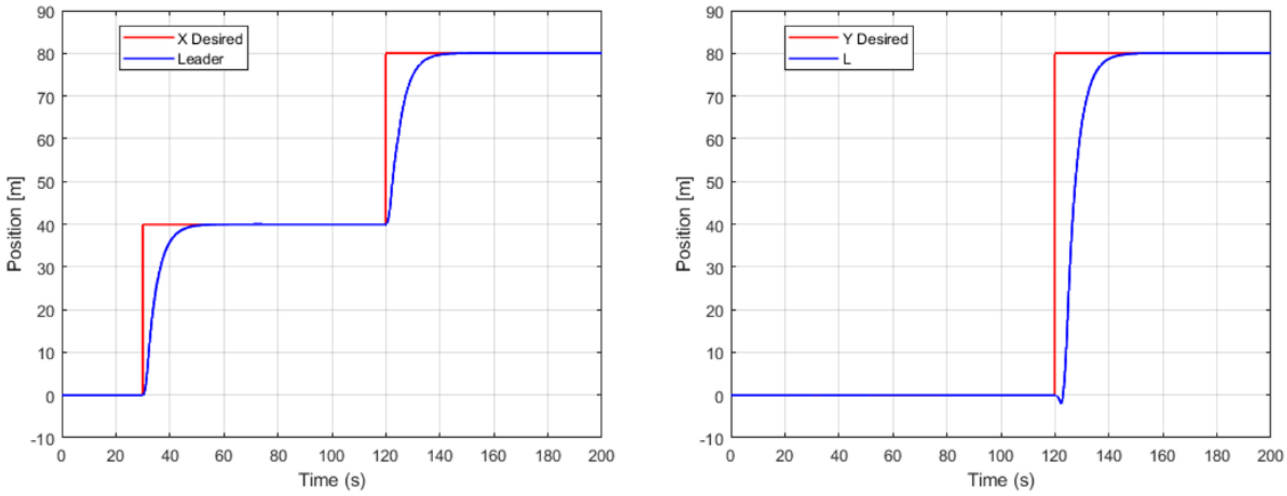

Figure 2.19. X and Y leader's position with respect to time

From Figure 2.19, it can be seen that the position tracking speed is approximately $4\frac{m}{s}$ which is in the limit of the desired speed for this thesis. Other scenarios with an obstacle located at $(10,0,10)$, $(10,5,10)$, and $(10,-5,10)$, and a goal location at $(30,0,10)$ are presented in Figure 2.20.

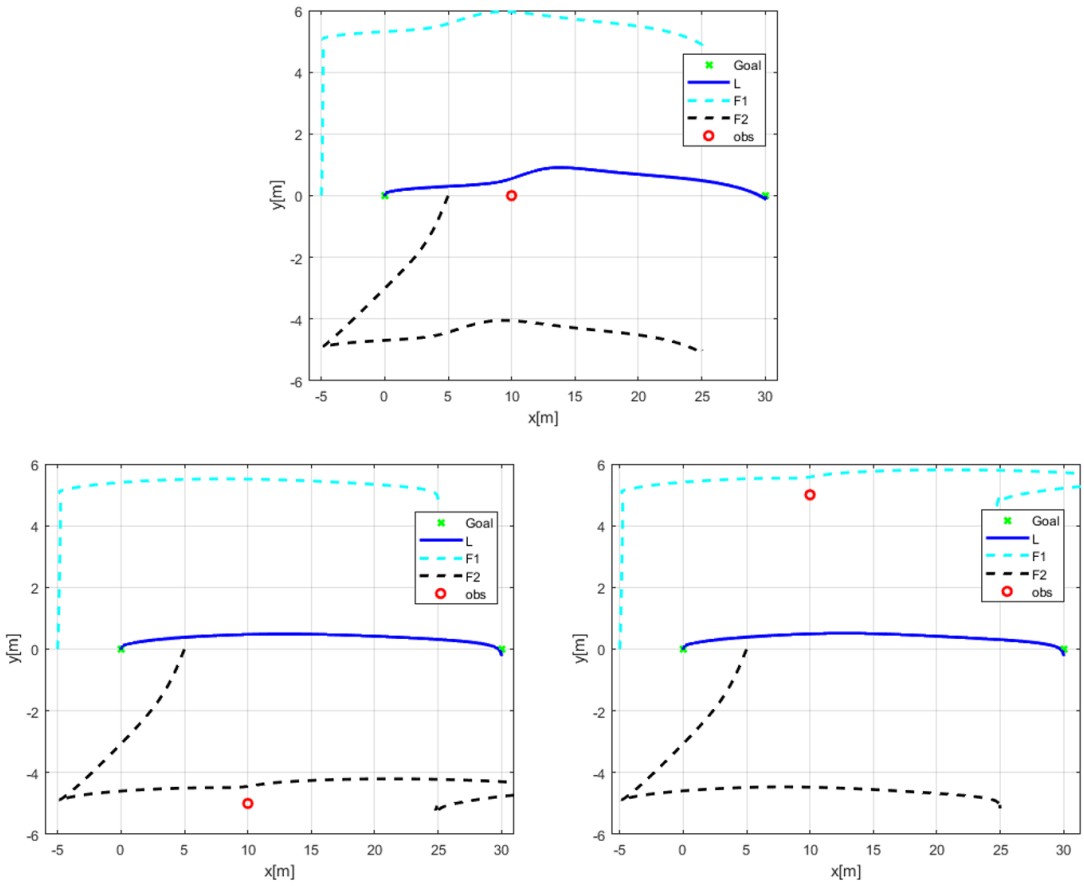

Figure 2.20. 2D Position tracking with obstacle avoidance

As a result, it can be seen that every quadcopter in the team can avoid obstacles interrupting its way. However, after multiple experiments, the repulsive forces generated from APF did not act directly and resulting in an instantaneous crash. Thus, solving this required to give the repulsive forces more weight than attractive ones using complementary filter, and therefore equation (2.24) can be updated to:

$$\ddot{x}_d = c\,\ddot{x}_d{}^{PID} + (1-c)\,\ddot{x}_d{}^{Rep}$$
$$\ddot{y}_d = c\,\ddot{y}_d{}^{PID} + (1-c)\,\ddot{y}_d{}^{Rep} \qquad (2.32)$$

Where c is scaling factor weighing one quantity over other. If $c < 0.5$, repulsive accelerations have more weight than attractive and it was tuned to $c = 0.3$ in this thesis.

2.7. CONCLUSION

This chapter presented the modeling, simulation, and control of a quadcopter. A PID-based attitude, altitude, and position controllers with APF repulsive forces as obstacle avoidance controller were proposed. Then, a formation controller is used to coordinate between a leader quadcopter and 2 followers. The simulation gave satisfactory results when using L-F scheme, however, this method lacks independence (i.e., in case of leader's failure, followers will have no position setpoints). On the other hand, the obstacle avoidance algorithm, APF repulsive forces, was able to guide the quadcopters away from obstacles efficiently. However, a main drawback of APF repulsive fields is the necessity of obstacles location pre-knowledge.

CHAPTER 3. PROJECT SPECIFICATIONS

3.1. INTRODUCTION

In This chapter, all quadcopters' components used in this thesis and their specifications will be presented. In addition, the functionality, role, and the reason of choosing of each component will be presented. Additionally, for the part of fire detection, hardware and software requirements will be demonstrated. Figure 3.1 illustrates the main parts used in this project. The system components can be divided into 4 main parts: quadcopter, vision, communication and central PC.

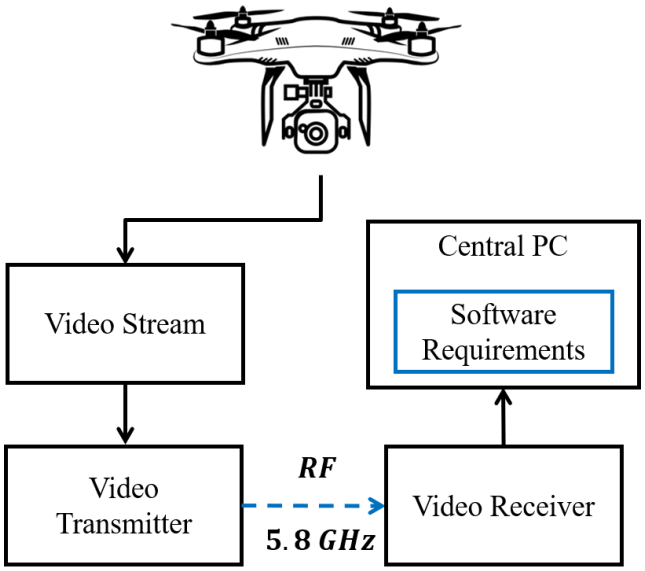

Figure 3.1. Project main parts

3.2. SYSTEM AND EQUIPMENT

The proposed swarm system is composed of three quadcopters: one leader and two followers. The hardware equipment that used to achieve this swarm system is presented in Table 3.1.

Table 3.1. Hardware equipment used in the system

Component	**Quantity**
Quadcopter frame	3
Brushless DC motor	12
Propeller	12
Electronic Speed Controller	12
Flight controller with Inertial Measurement Unit	3
Power Distribution Board	3
Battery	3
GPS sensor	2
Onboard computer	1
Onboard camera sensor	1
Gimbal	1

In this work, three different quadcopters will be selected to maintain a predefined swarm shape during flight. The first quadcopter (the leader) is composed of MULTIWII

flight controller, Arduino Mega microcontroller, and sensors fixed on the frame. The first follower quadcopter uses NAZA-M V2 as flight controller, in addition to a different type of quadcopter frame. The second follower quadcopter is a Parrot AR Drone with 1280×720 HD camera that supports live stream and image capturing. This quadcopter will be controlled by MATLAB software on personal computer (PC).

3.3. MULTIWII FLIGHT CONTROLLER

The MULTIWII SE V2.0 flight controller, shown in Figure 3.2, is an open-source circuit board developed by Oscar Liang in 2013. The flight controller aims to control the RPM of motors of any multi-rotor aircraft, in response to inputs such as desired altitude, and desired position. Moreover, these inputs will be transformed to 6 degrees of freedom needed to control any quadrotor. MULTIWII is based on Arduino board and uses gyro / accelerometer sensors to sense quantities relative to quadcopter body frame. Moreover, this controller is totally open source and very flexible system. Table 3.2 presents the specifications of MULTIWII SE V2.0. Moreover, this flight controller supports scalable options with full programmability. The selection of MULTIWII was according to its ability to tackle tasks assigned to it, and in accordance to market availability and low-cost benefits.

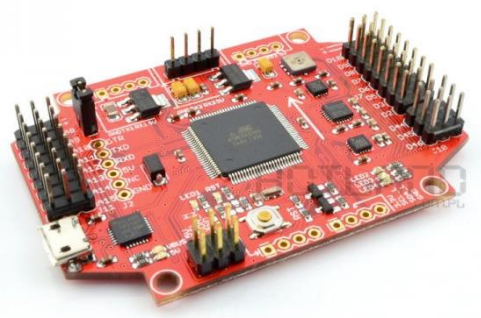

Figure 3.2. MULTIWII SE V2.0

Table 3.2. MULTIWII SE V2.0 features and specifications

Features/Specifications	**Description**
Motor output	Up to 8-axis
Servos output	2 for PITCH and ROLL gimbal system
Dimension	40x12x40mm
Weight	9.6g
Serial cables	FTDI/UART TTL
Voltage regulator	3.3V and 5V LDO
Microcontroller	ATMega 328P
MPU6050	Gyro Full Scale Range: ± 250 ± 500 ± 1000 ± 2000 Gyro Sensitivity: 131 / 65.5 / 32.8 / 16.4 Gyro Rate Noise: 0.005 / 0.005 / 0.005 / 0.005 Accel Full Scale Range: ±2 ±4 ±8 ±16 Accel Sensitivity: 16384 / 8192 / 4096 / 2048
HMC5883L	3-axis digital magnetometer Full scale range: ± 8 gauss Sensitivity: 230~ 1370 LSb/gauss Cross-Axis Sensitivity: ± 0.2 %FS/gauss
BMP085 (barometric pressure sensor)	Pressure range: 300~1100 hPa Absolute accuracy pressure: ± 2.5 hPa
Display	I2C LCD/OLED or CRIUS I2c-GPS NAV board

The block diagram presented by Figure 3.3 shows how MULTIWII and Arduino Mega communicate with each other. The autonomous tasks are handled by a high-level controller (Arduino Mega) that is responsible for position and altitude control and communicates with the low-level controller (MULTIWII) that is responsible for attitude control. These two communicate with each other serially through Tx and Rx pins. Arduino Mega takes data form MULTIWII like IMU and GPS data, etc. then, provides control commands based on a given navigation algorithm in the form of PWM signals for MULTIWII.

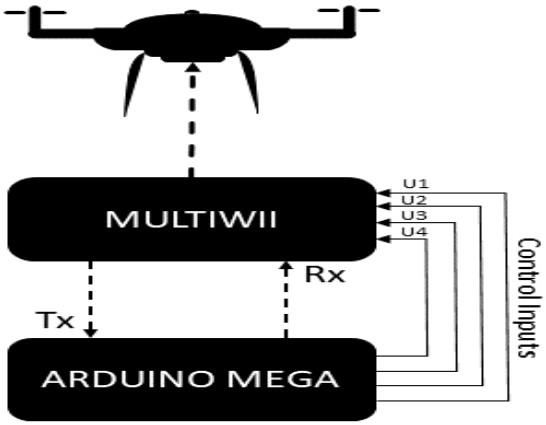

Figure 3.3. Data exchange between MULTIWII and Arduino mega

3.4. NAZA FLIGHT CONTROLLER

NAZA-M V2 shown in Figure 3.4 is a closed source powerful flight controller. This flight controller is selected as a second flight controller according its market availability and its specifications, shown in Table 3.3, that meets the tasks assigned to it. NAZA-M V2 receives the flight commands from a higher-level position controller presented by Arduino Mega.

Figure 3.4. NAZA_M V2 flight controller

Table 3.3. NAZA-M V2 Specifications

Specifications	**Value**
Weight	95g

Dimensions	$45.5 \times 32.5 \times 18.5mm$
Power Consumption	Max: 1.5W (0.3A@5V) Normal: 0.6W (0.12A@5V)
Working Voltage Range	MC:4.8V~5.5V VU input: 7.4V~16.0V Output: 3A@5V
Max Yaw Angular Velocity	200 ◦/s
Max Tilt Angle	35◦

The block diagram presented in Figure 3.5 shows NAZA-M V2 receiving the position commands from a higher-level position controller. The autonomous tasks are handled by the high-level controller (Arduino Mega) that is responsible for position and altitude control, and communicates via PWM signals with the low-level controller (NAZA-M V2) that is responsible for the attitude control.

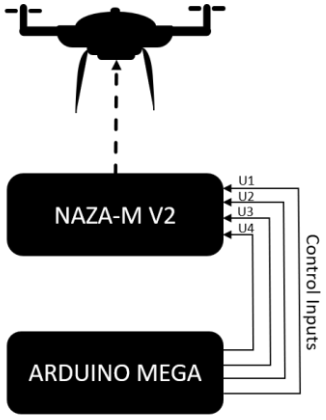

Figure 3.5. Data exchange between NAZA-M V2 and Arduino mega

3.5. PARROT AR DRONE

The AR Drone shown in Figure 3.6 is a quadcopter which combines numerous of the new and advanced technologies in radio-controlled flight that receives position information via WIFI. AR Drone can be controlled through MATLAB Simulink via computer. MATLAB expresses the whole control system for AR Drone (high- and low-level

control systems), and through flight commands of MATLAB, AR Drone moves. AR Drone is selected based on its specifications presented by Table 3.4 and its market availability.

Figure 3.6. Parrot AR Drone

Table 3.4. AR Drone specifications

Specifications	**Value**
Dimensions	23x23x5 inches
weight	4.62 pounds
Battery	Lithium Metal Batteries
RAM	1024 MB
Wireless communication technologies	WIFI
Camera	720 HD camera records video at 30 fps
Processor	32-bit ARM Cortex A8 1GHz

3.6. QUADCOPTER FRAMES

All components of the first quadcopter (using MULTIWII) will be fixed together on a “Turnigy Heavy Aerial Lift” frame shown in Figure 3.7 (type A). On the other hand, all components of the NAZA-M V2 controlled quadcopter will be fixed together on “DJI F450” frame presented in Figure 3.7 (type B). These two frames are selected based on their specifications, shown in Table 3.5, which are low prices, and market availability.

Figure 3.7. Quadcopter frames chosen for the project

Table 3.5. Turnigy Heavy Aerial Lift and DJI F450 Specifications

	Turnigy Heavy Aerial Lift	**DJI F450**
Specifications	**Value**	
Weight	614 g	282 g
Width	585mm	450 mm
Stator Size	28 x 35 mm	23 x 12 mm

3.7. BRUSHLESS DC MOTORS

A Brushless DC motor (BLDC) transforms electrical energy into mechanical rotational energy. The BLDC is a good choice for quadcopter applications because of its efficiency that rises up to 90% and its long operational life. In addition, its large speed range that vary according to the Pulse Width Modulation (PWM) signals that is also an appealing feature.

The brushless motors selected, are the PROPDRIVE V2 2826 1200KV (type A) for the leader and A2212/13T (type B) for the follower, shown in Figure 3.8. The motors selection was based on their specifications presented by Table 3.6, market availability, and low-cost benefits.

Figure 3.8. PROPDRIVEV2 and A2212/13T motors

Table 3.6. BLDC motors specifications

	PROPDRIVE V2 2826	**A2212/13T**
Specification	**Value**	
Model	PROPDRIVE v2 2826 1200kv	A2212 1400KV
KV ($\frac{RPM}{V}$) ratio	1200 KV	1000KV
Max current	15 A	20 A
ESC	20~30 A	20~30A

Cell Count	3s~4s Lipoly	2s~4s
Pole Count	12	14
Max Power	215W @12v (3S) /286W@15v(4s)	220/3 W
Shaft	3.175mm	3.17 mm
Weight	59 g	62 g

3.8. PROPELLERS

The Propellers are used to convert the rotational speed generated by the BLDC into a lift force. Propellers are chosen by their pitch angle that can describe the travel distance of one propellers rotation, and their length that are the diameter of a circle the propellers makes when it is spinning. The Propellers selected (shown in Figure 3.9Figure 3.9) have 10 inches as length and 4.5 degrees as pitch angle, two CW rotation, and two CCW rotating propellers. This selection is according to market availability and project needs.

Figure 3.9. 1045 Propellers

3.9. ELECTRONIC SPEED CONTROLLER (ESC)

The ESC is an electronic device used to control the speed of a BLDC motor by activating and disactivating the appropriate MOSFETS. The ESCs used in this thesis, shown

in Figure 3.10, are characterized by their maximum current rating (A) and their duty cycle and frequency range. ECSs (type A) are used into the leader, and the others (type B) are used into the first follower. ECSs should be selected upon the maximum current needed by motor and other specifications mentioned in Table 3.7.

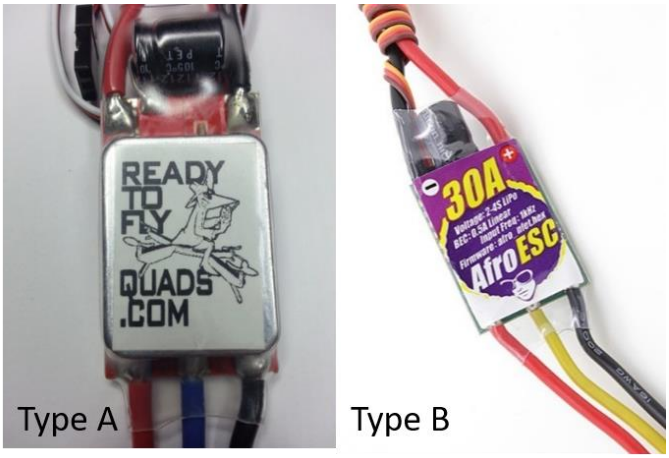

Figure 3.10. ECSs chosen for the project

Table 3.7. ESCs Specifications

	AFRO ESC 30A	**Ready To Fly 30A**
Specifications	**Value**	
Current	30 A (continuous)	~25A (continuous)
Voltage Range	2s~4s (1s = 4.2V)	3s~4s
Frequency	Up to 1 KHZ	Up to 600 Hz
weight	26.5g	28g (with bullets)

3.10. BATTERY

Leader (used MULTIWII) and follower 1 (used NAZA-M) quadcopters are connected to a DC power source presented by a Lithium Polymer (LiPo) “HRB 5000mAh

11.1v 50C" battery shown in Figure 3.11 (Type A). It can be calculated using the following formula:

$$flight\ time\ (in\ minutes) = \frac{Capacity\ (Ah) \times 0.8}{Total\ load\ (A)} \times 60 \qquad (3.1)$$

Where the battery capacity is expressed in Ampere hours (Ah), 0.8 is the efficiency of the LiPo battery (80%) and the total load consumption is expressed in Amperes (A) which can be obtained by adding all current consumption of each component and is approximated to be 60A. This total load cannot be checked during flight, however it is assumed to be constant. For this project, a 5000 mAh LiPo battery was chosen. In the other hand. AR Drone is connected to another type of battery, as shown in Figure 3.11 (Type B), called "Parrot AR Drone 2.0 1500mAh High Density". These batteries have many advantages such as quick recharging, high efficiency, etc. that are selected referring to them, and to its specifications presented in Table 3.8.

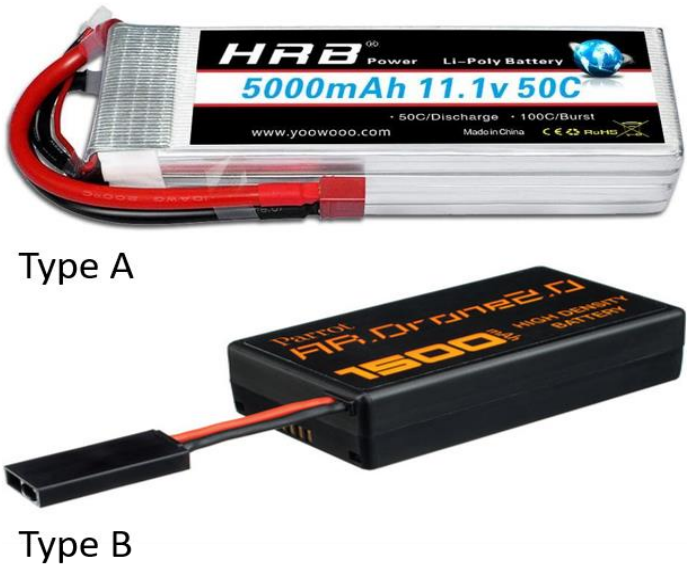

Figure 3.11. Batteries chosen for the project

Table 3.8. HRB and AR Drone battery specifications

	HRB 5000mAh 11.1v 50C	**Parrot AR Drone 2.0 1500mAh**
Specifications	**Value**	
Voltage	11.1V	11.1V
Capacity	5000 mAh	1500mAh
Weight	376g	100g
Battery Cell Type	Lithium Polymer	Lithium Polymer
Dimensions $(L \times W \times H)$	$155 \times 48 \times 24mm$	$104.1 \times 68.5 \times 25.4mm$
Balancer Connector Type	JST-XHR	JST-XH

3.11. GPS SENSOR

The Global Positioning System (GPS), shown in Figure 3.12, is used to determine the geographical location of the quadcopter. The GPS is a device that communicates with 30+ navigation satellites around the Earth. The satellites' locations are known through the continuous signals sent out of them. Then, GPS receives these signals and calculates its distance from several GPS satellites. Hence, the GPS receiver can figure out its latitude and longitude positions. The GPS selected is the Ublox Neo 6m based on market availability and its specifications are shown in Table 3.9.

Figure 3.12. Ublox Neo 6m GPS

Table 3.9. Ublox Neo 6m GPS specifications

Specification	**Value**
Supply	2.7-3.6V
interface	UART/USB/SPI/DDC(I^2C compliant)
Navigation	Up to 5 Hz
Accuracy	Position: 2.5 m CEP SBAS: 2.0 m CEP
Tracking Sensitivity	161 dBm

3.12. POWER DISTRIBUTION BOARD

The Power distribution board or panelboard shown in Figure 3.13 (Type A), is used in the leader to distribute the voltage over the 4 motors, plus, the whole system fixed on the quadcopter. In the other hand, type B shown in Figure 3.13 is built-in power distribution board in frame "DJI F450" that used for the follower 1. Power distribution boards (type A) are selected based on their specifications presented in Table 3.10.

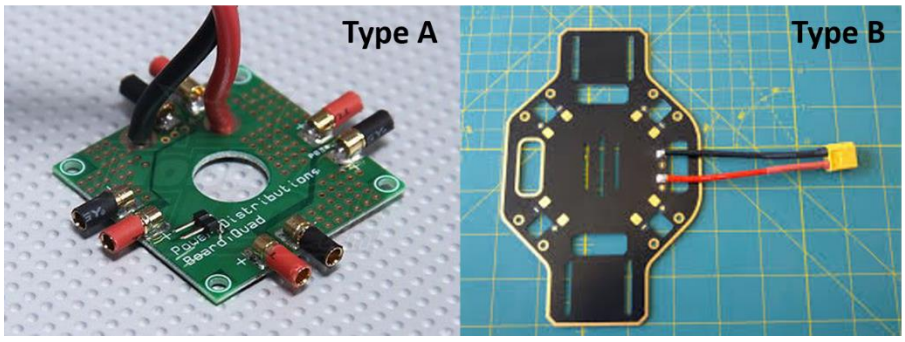

Figure 3.13. Power distribution board

Table 3.10. Power distribution board (type A) Specifications

Specifications	**Value**
Current	$4 \times 20A$ (MAX)
Power input	XT60 with 12AWG wire

Motor output	$4 \times 3.5mm$ Female bullet plug
Aux output	2 pin JST compatible
Weight	27.3g (including wires)

3.13. ARDUINO MEGA 2560

The Arduino Mega 2560 is a development board based on ATmega2560 microcontroller. The main role of Arduino mega in this project is to tackle the high-level control such as position and altitude control for the leader and follower 1. In the other hand, AR Drone will be used as navigation controller.

3.14. GIMBAL

The camera gimbal is a device used to control the movement smoothly without producing vibrations. It is mounted on the quadcopter to provide smooth video output from the camera (vibration-free) and is driven by 3 brushless motors to stabilize the camera's position in 3 directions (roll, pitch and yaw). The selected gimbal is shown in Figure 3.14 and has the specs listed in Table 3.11.

Figure 3.14. Gidy camera gimbal

Table 3.11. Gidy camera gimbal specs

Parameter	**Value**
Pitch	-90° to+30°
Roll	0° or 90° (horizontally and vertically)
Stabilization	3 axes (pitch, roll, yaw)
Weight	158.757 g

Now, going specifically to the fire detection implememtation requirements, it needs the following devices to be implemented and completed.

3.15. VISION

The appropriate selection of vision system, or camera is very crucial for NN application. There are several camera factors that will affect the performance of fire detection algorithm which are frame rate, camera resolution, field of view, and ISO range. Fps drop depends on the PC used, filters applied, and number of objects detected in a single frame. The camera selected for this project is built-in with the gimbal device shown in Figure 3.14 and its specs are listed in Table 3.12.

Table 3.12. Camera specs

Parameter	**Value**
Sensor	1/2.3'' (CMOS), 12.35 MP
ISO range	Video: 100-3200 Photo: 100-1600
Image size	4K – 4000×3000
Video recording modes	**C4K**: 4096×2160 24p **4K**: 3840×2160 24/25/30p 2.7K: 2720x1530 24/25/30p **FHD**: 1920×1080 24/25/30/48/50/60/96p

	HD: 1280×720 24/25/30/48/50/60/120p

3.16. COMMUNICATION

The communication system includes the video transmitting and receiving units. This unit is responsible to transmit live video streams from quadcopter and receive them on the central PC in order for these frames to be processed. These devices are chosen according to their Radio Frequency (RF) bandwidth, power consumption, sending rate, and sending range. For this project, it is desired to have a video-stream transmission within 1.5 to 2 km. Thus, the selected communication system is shown in Figure 3.15 and its specs are listed in Table 3.13.

Figure 3.15. AKK video transmission system

Table 3.13. Communication system specs

Parameter	**Value**
Sending range	2000 m and ≥3000 m in open areas
Number of channels	40 covering bands A, b, E, F, r
Operating voltage	7-16V
Power consumption	0.22/0.65A: non-transmitting/transmitting @12V
Video format	NTSC/PAL
Weight	85 g

3.17. CENTRAL PC

This is the most important component in this thesis. The central PC is responsible for all fire detection process. It will receive raw images from the quadcopter flying over a forest region, then it will process these incoming frames and detect fires using Artificial Neural Networks (ANN). Fire detection was done on a central computer since ANN are power-greedy and will noticeably reduce the flight time if calculated onboard. Choosing the right PC for this application is a bit expensive.

3.17.1. Hardware Requirements

The chosen PC is based on CPU not GPU and all fire detection scripts will be run by, which causes significant frames per second (fps) drop when detecting fires in frame. GPU has very small fps drop when running fire detection scripts on, since it can handle more graphical information than CPU but it is very expensive. However, CPU-based PC was selected upon its availability, cheapness and is shown in Figure 3.16 with its specs listed in Table 3.14.

Figure 3.16. HP laptop 15-da1xx

Table 3.14. HP laptop specs

Parameter	Value
Processor	Intel® core™ i5-8265U
Ram	8 GB
System type	64-bit Operating System (OS)
Windows	10

As mentioned before, the training will be held over "*Google Collabs",* which is a python development environment that runs in any browser, because it offers a GPU rather than CPU which is way better to train a network over. Training a NN over a CPU would take approximately 4-5 times more time than training it over a GPU, according to Buber et al. [38]. "*Google Collabs"* uses the hardware specs listed in Table 3.15.

Table 3.15. Google Collabs hardware specs

Parameter	**Value**
GPU	Nividia k80/T4
GPU memory	12GB/16GB
GPU memory clock	0.82GHz/1.59GHz
Performance	4.1 TFlops/8.1 TFlops
Number of CPU cores	2
Available RAM	12GB (upgradeable to 26.75GB)
Disk space	358GB

3.17.2. Software Requirements

This section will show the necessary software applications to be installed on the central laptop in order to setup and run all required files to detect fires in video frames. There are mainly 2 required software applications that must be installed (all installation procedure is shown in Appendix section).

3.17.2.1. Python

Python is a high-level and general-purpose programming language which is getting more popular over time. It is widely used in Machine Learning (ML) applications because it offers very easy and ready-to-use libraries. It includes a very famous computer vision library (OpenCV) that provides a very large image processing functions. For this project, the required libraries to run the fire detection code are: *"OpenCV"* and *"Numpy"*.

Figure 3.17. Python logo

3.17.2.2. LabelImg

LabelImg is an interactive image annotation tool written in python. It will be used to label fire regions in the training data set (1000 images). LabelImg is one of many image annotations tools but it is selected for its simplicity and *"YOLOv3"* network friendly. The procedure of using this software application and setting up training set will be explained in chapter 4.

Figure 3.18. LabelImg logo

3.18. CONCLUSION

As we mentioned before, the equipment used in our project is defined by three quadcopters, the leader use MULTIWII and two followers which are AR drone and quadcopter that used NAZA-M. All data, role, and specifications for the equipment was presented in this chapter. Additionally, all hardware and software requirements to build an early fire detection system using NN were shown. Python scripts for detecting fires will be handled by a CPU-based PC but training NN will be held by a GPU-based hardware offered from "Google Collabs".

CHAPTER 4. PROJECT DESIGN

4.1. INTRODUCTION

In this chapter, hardware implementation process of 3 quadcopters flying in formation having different platforms will be presented. The implementation phase is divided into 3 subcategories; unit testing, integration testing and project validation. Unit testing focuses on testing each individual component to ensure its functionality. Integration testing aims to combine multiple platforms and test their functionality together. In addition, a Kalman Filter (KF) will be implemented to better estimate position. Also, the procedure to train a neural network using YoloV3 models to detect forest fires will be demonstrated. A brief introduction for Neural Networks (NN) will be presented and the necessary parameters that have to be tuned during training will be highlighted.

4.2. METHODOLOGY

In order to coordinate between 3 quadcopters (1 leader and 2 followers), the leader is responsible for sending the high-level commands along with its position. These commands are sent from a RC controller held by the operator and received by the leader which is valid up to 2 Km of direct distance. Communication between the leader and the followers is obtained by radio frequency modules (2.4 GHz). The high-level commands as states can be summarized by the state diagram shown below.

Taking off: All quadcopters will take off to approximately 2 meters above ground.

- Ascending/Descending: All quadcopters will ascend/descend to a specific altitude determined by the leader.

- Position holding: All quadcopters will hold their current position.
- Navigating: Followers will follow the leader with a specified separation distance.

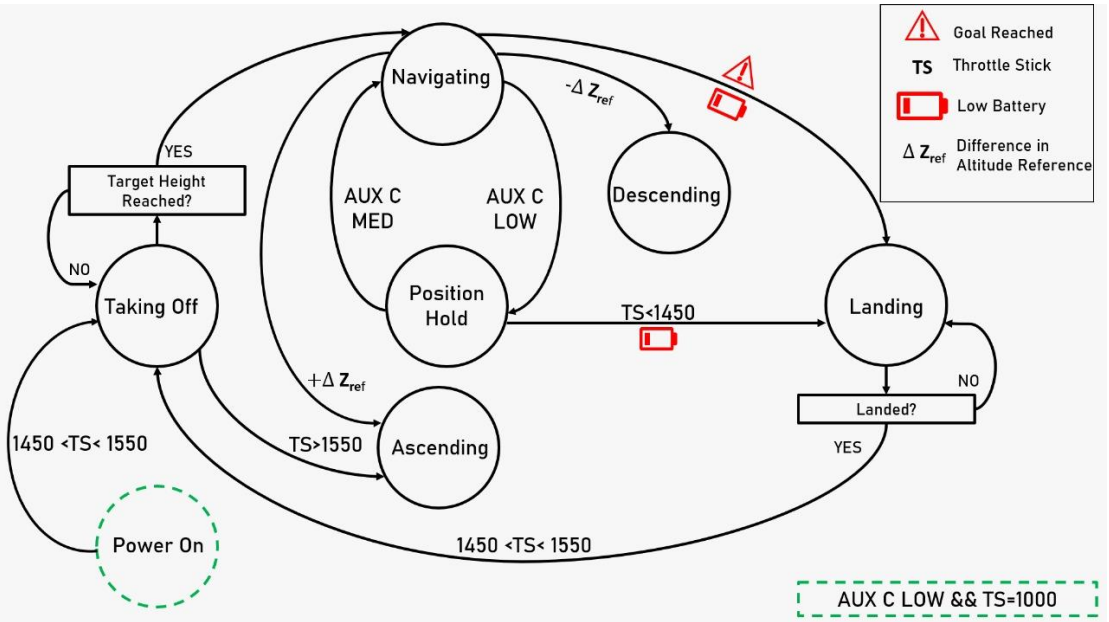

Figure 4.1. Autonomous State Diagram

4.3. NAZA CONTROLLER-BASED QUADCOPTER (LEADER)

The Naza controller, as mentioned earlier, is a standalone autopilot system consisting of several features. The quadcopter which is equipped with the Naza controller is configured as the leader. The leader is responsible for sending the high order commands (via radio frequency module: nrf24) received from a ground operator via a remote control. To access these signals from the RC operator, an Arduino uno is mounted over the Naza to handle the high order commands; reading and decoding RC and GPS signals, sending the leader's position and commands to the followers and calculate the desired attitude in order for the leader to follow the specified waypoints. Figure 4.2 shows the physical connections between Arduino uno and Naza flight controller and Figure 4.3 illustrates the circuit diagram. The Arduino will control the Naza via the TX inputs (as if it is the RC transmitter). Naza flight controller has several features that can help during flight, and are summarized in the below table [39].

Table 4.1. Naza Features

Feature	**Description**	**Command**
Automatic takeoff	The drone will takeoff and hold its altitude at approximately 2.5 meters	Throttle stick at middle position (> 1500 us).
Manual Mode	This mode is for acrobatics or racing (not used in this project)	Control mode switch at upper position (1100 us)
Attitude Mode	Will keep the drone leveled when roll and pitch sticks are at their center position.	Control mode switch at middle position (1500 us)
Attitude GPS Mode	Will hold the quadcopter in place.	Control mode switch at lower position (1800 us)

Fortunately, Naza flight controller is easy to deal with. It is just required to know when and where to give a certain command in order for the Naza to operate properly. However, the complex part is configuring the Arduino to operate exactly as a remote-control transmitter and decode Naza's GPS/Magnetometer signals. The following subsections will discuss the decoding processes in details.

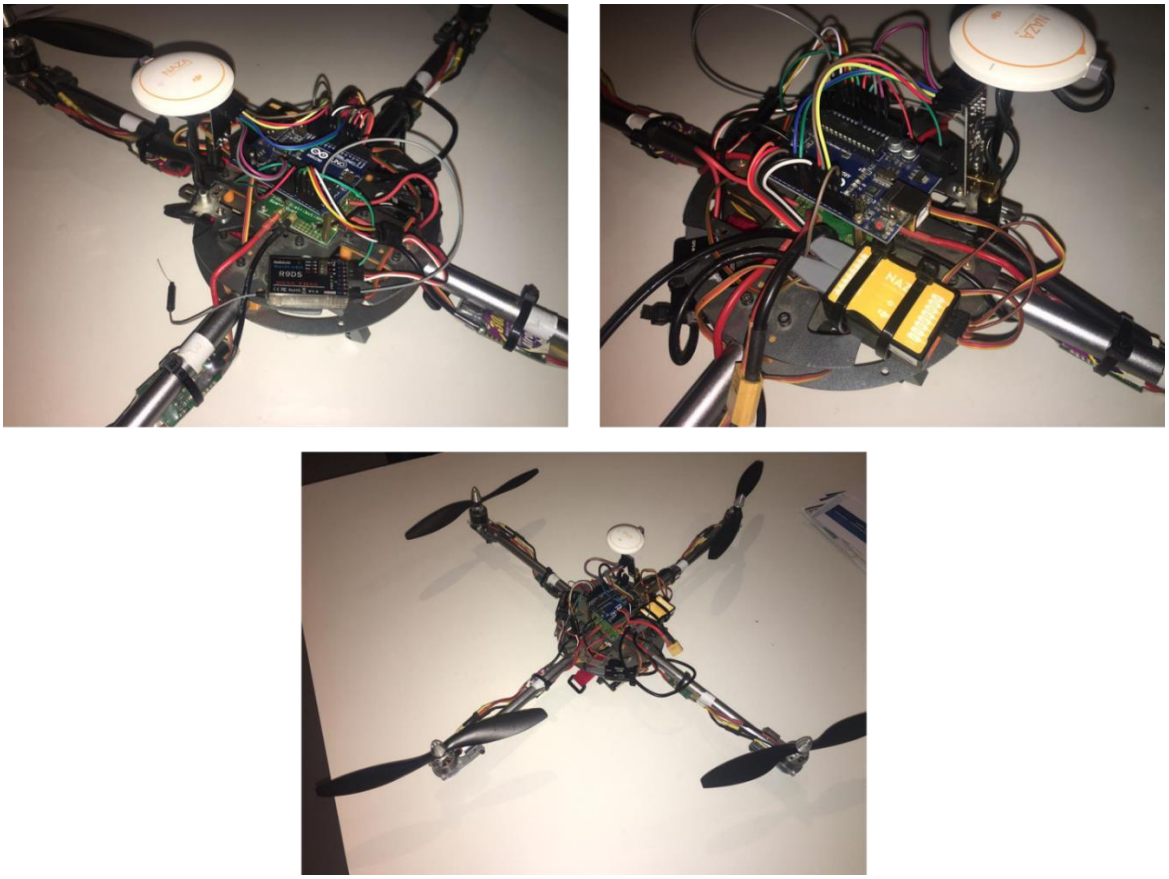

Figure 4.2. Arduino-Naza Physical Connections

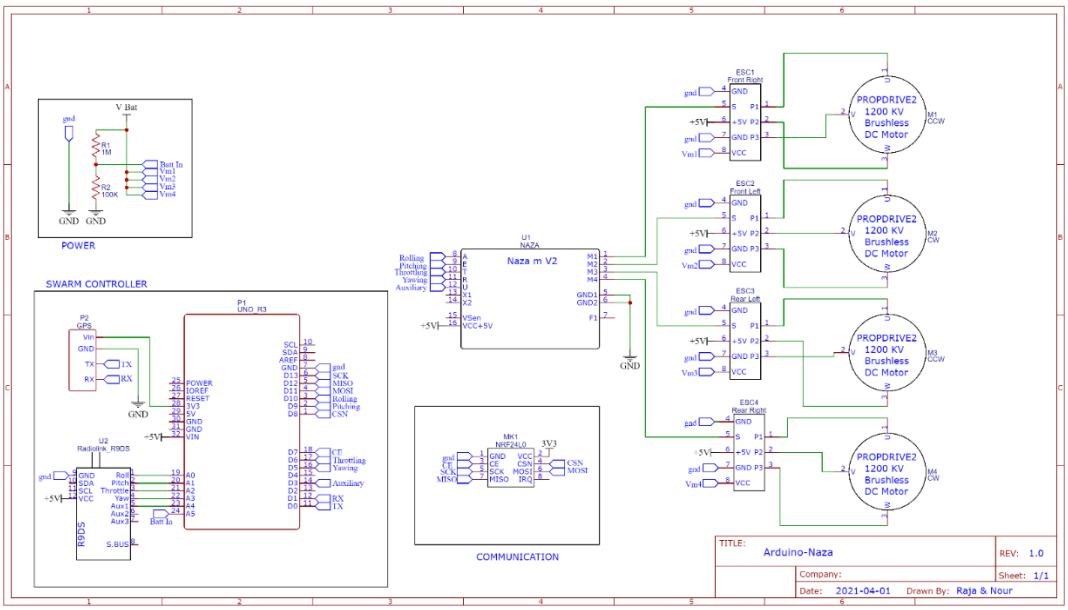

Figure 4.3. Arduino-Naza Circuit Diagram

4.3.1. RC Transmitter Signal Decoding

Traditional RC transmitters work with Pulse Width Modulation (PWM) signals. The amount of on-time (T-ON) of the signal or the duty cycle determines the required action. Generally, the refresh rate of a RC transmitter signal is 50 Hz which means 20 ms period which is a standard communication signal between RC transmitters and flight controllers. The duty cycle is varied between 1 and 2 ms which are the low-most and up-most commands respectively. As mentioned before, the Arduino is responsible to read and decode these signals from the transmitter which are 5 channels (throttling, rolling, pitching, yawing and the auxiliary switch). Each channel of these has a varying duty cycle between 1000 and 2000 us over a period of 20 ms and 5 V amplitude as shown below in Figure 4.4.

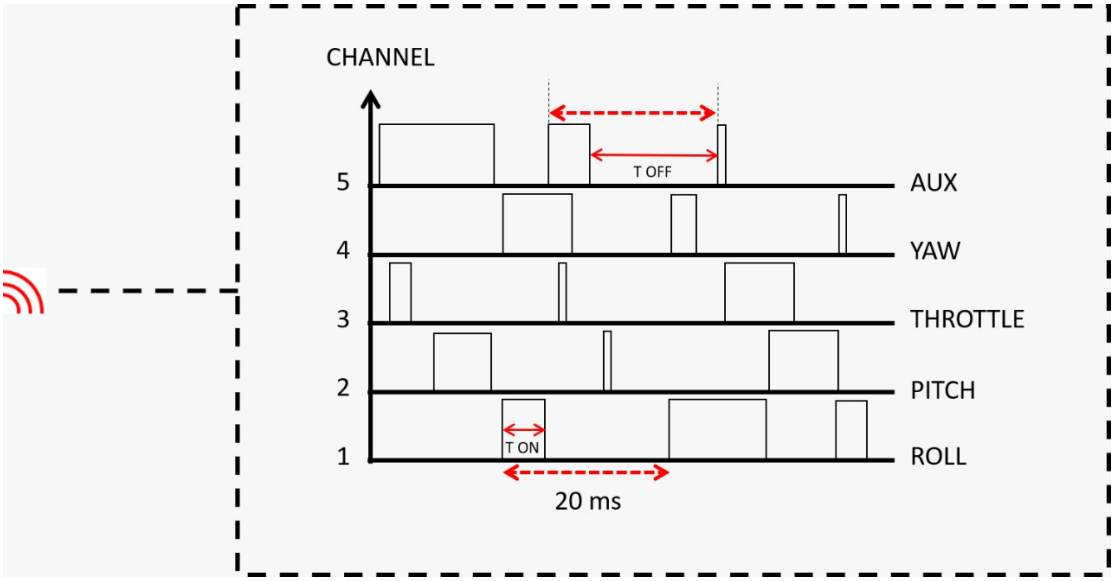

Figure 4.4. RC Transmitter Signal

The algorithm to read these signals must be to verify 5 digital pins as inputs and continuously check if one of these inputs has changed its state (1 to 0 or 0 to1) then calculate the time between these transitions. Fortunately, Arduino has Pin-Change Interrupt (PCINT) feature over three groups of pins (in this project, PCINT group 1 is used: Pins A0 till A5) which automatically stimulates an interrupt subroutine if one of these inputs has changed its

state. By doing this, the loop time of the code would take a much fewer time (several microseconds) than continuously checking input pins (hundreds of milliseconds). The below flowchart (Figure 4.5) describes how PCINT group 1 behaves if one of its inputs is changed.

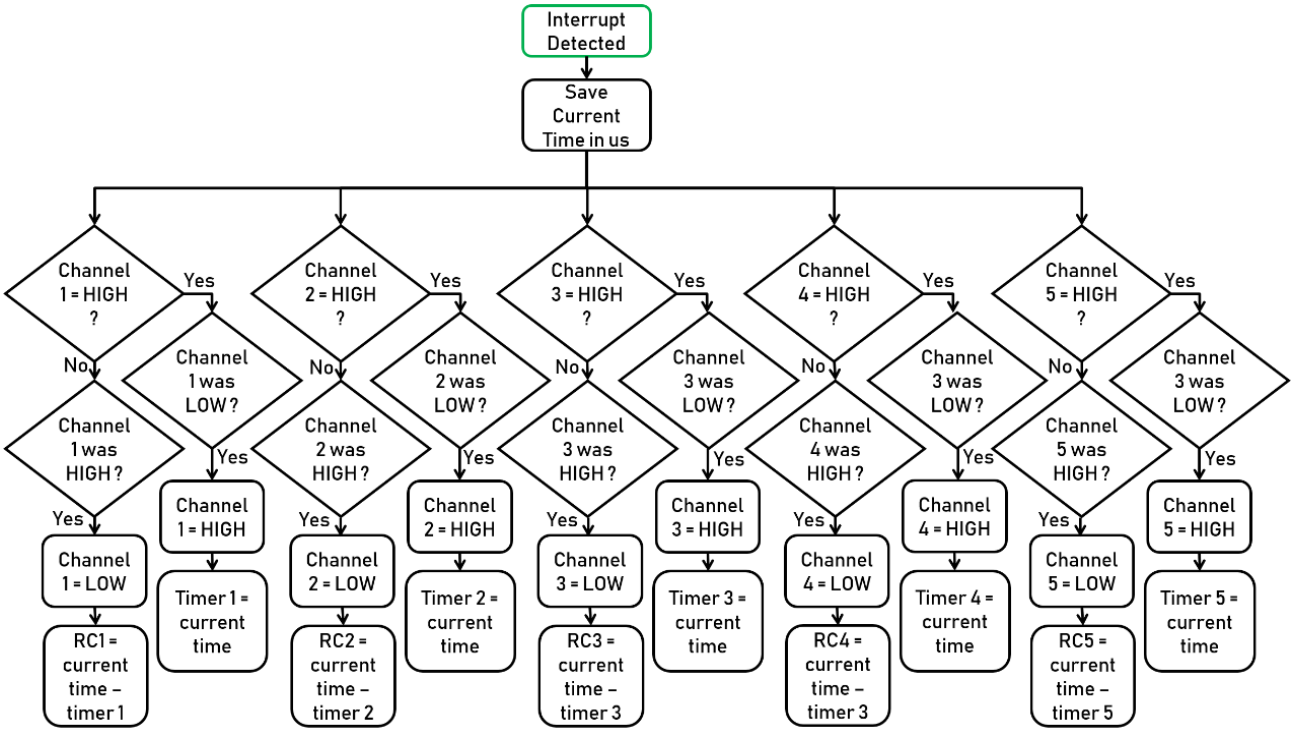

Figure 4.5. PCINT Subroutine

The subroutine, when called, will save the current time in microseconds and start checking the pins one by one (pins A0 till A4 refer to channel 1 till 5) to verify which pin has changed its state. RC1:5 refer to the final measured time in us of the desired 5 channels which will be used later to decide what commands to send and execute. The Arduino will write the desired PWM signal in us to the RC receiver pins of the Naza. The code is attached in the appendix and for further information about PCINT, the reader is directed to [40].

4.3.2. Naza GPS/Compass Decoding

In order for executing autonomous tasks, it is a must to know the current location of the quadcopter. Most GPS devices in the market use the NMEA protocol when sending

position and time information [41] over normal serial communication bus. Naza GPS uses CAN bus communication to transmit data over a serial bus. Figure 4.6 shows the Naza GPS pinout diagram where only one serial pin is taken (TX pin) to receive its data.

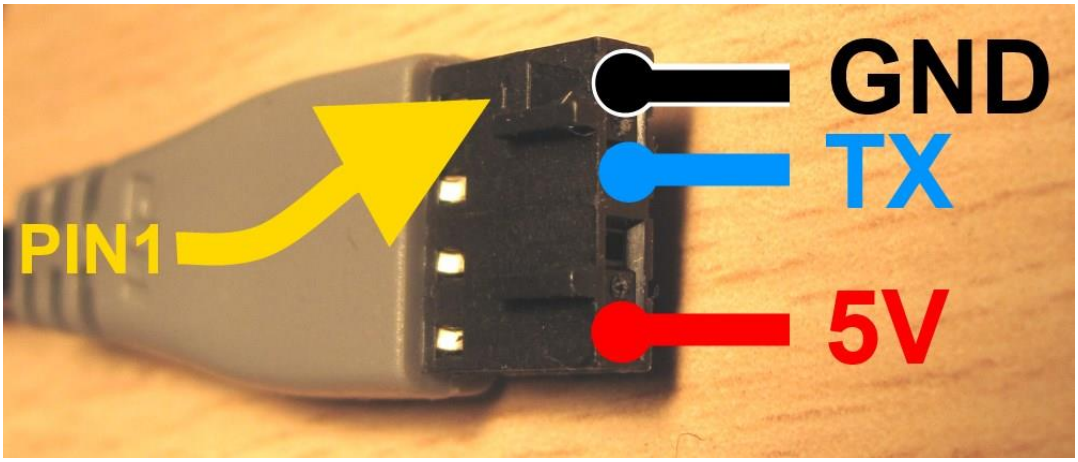

Figure 4.6. Naza GPS Pinout

When connecting the GPS's TX pin to Arduino's RX pin, the following message is shown using 115200 baud-rate [42]:

$$55\ AA\ XX\ YY < Payload > ZZ\ ZZ \tag{4.1}$$

Where, XX is the length of the message, YY is the message ID, $Payload$ is the message content, and $ZZ\ ZZ$ is the checksum. Where ID 10 contains GPS data like 3D position, number of satellites, fix type, time, horizontal and vertical accuracy and are sent every 250 ms. ID 20 contains raw compass data and are sent every 30 ms. And ID 30 contains module version numbers which is sent every 2 seconds. For the best accuracy, it is required to have at least 4 satellites seeing the GPS and a 3D fix type.

4.4. MULTIWII CONTROLLER-BASED QUADCOPTER (FOLLOWER)

Multiwii is also a standalone flight controller but with more complex tuning parameters than Naza flight controller. However, Multiwii is an open source and can be

adjusted. As mentioned before, Multiwii consists of 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer, and an altimeter. It uses a complementary filter to calculate the attitude variables $(\varphi \; and \; \theta)$ and the heading angle relative to north pole ψ is obtained from the magnetometer corrected with angular rate measurements. In the same manner of Naza controller, an Arduino mega is attached as a high-level controller responsible for receiving attitude and position information from Multiwii, applying the swarm navigation algorithms, and controlling the behavior of the quadcopter over the RC receiver pins. Figure 4.7 shows the connections between Arduino mega and Multiwii flight controller. The Multiwii sends attitude data at 100 Hz refresh rate and position was configured via CFG messages [43] to send data at 10 Hz.

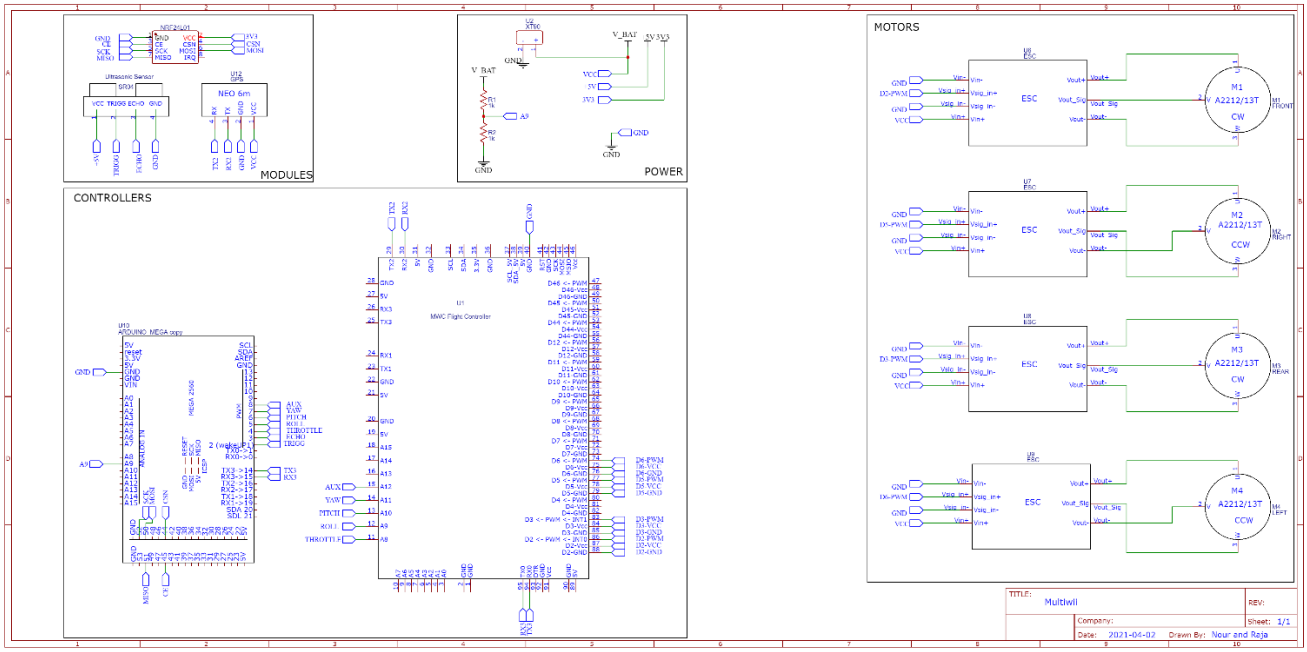

Figure 4.7. Arduino-Multiwii Circuit Diagram

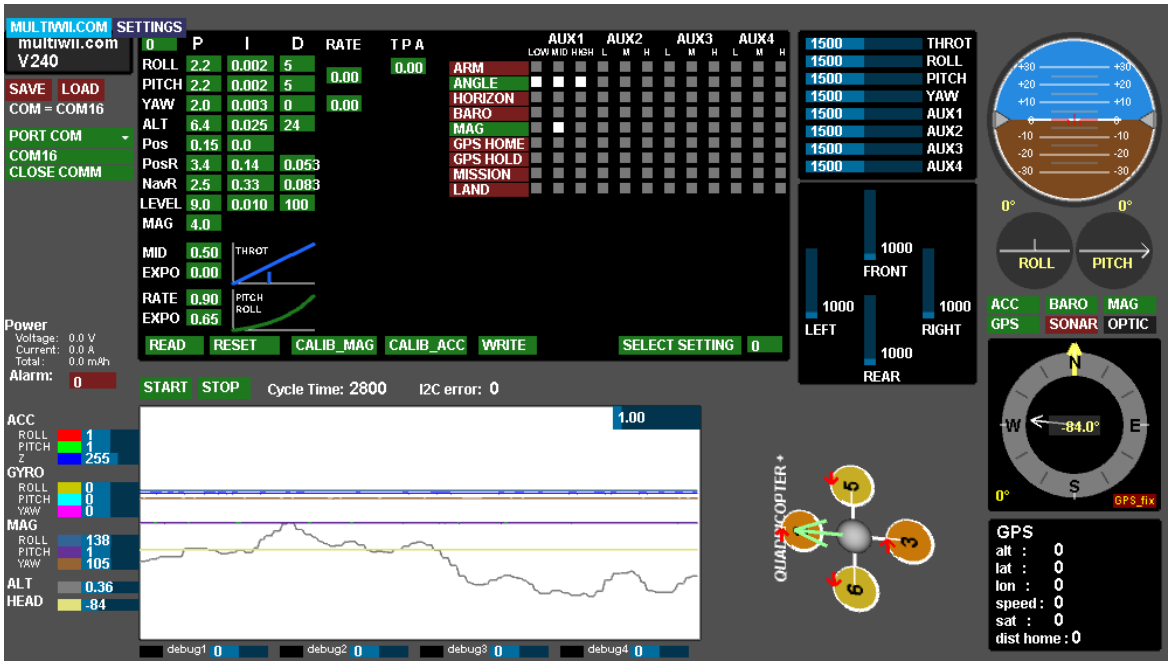

Figure 4.8. Multiwii GUI

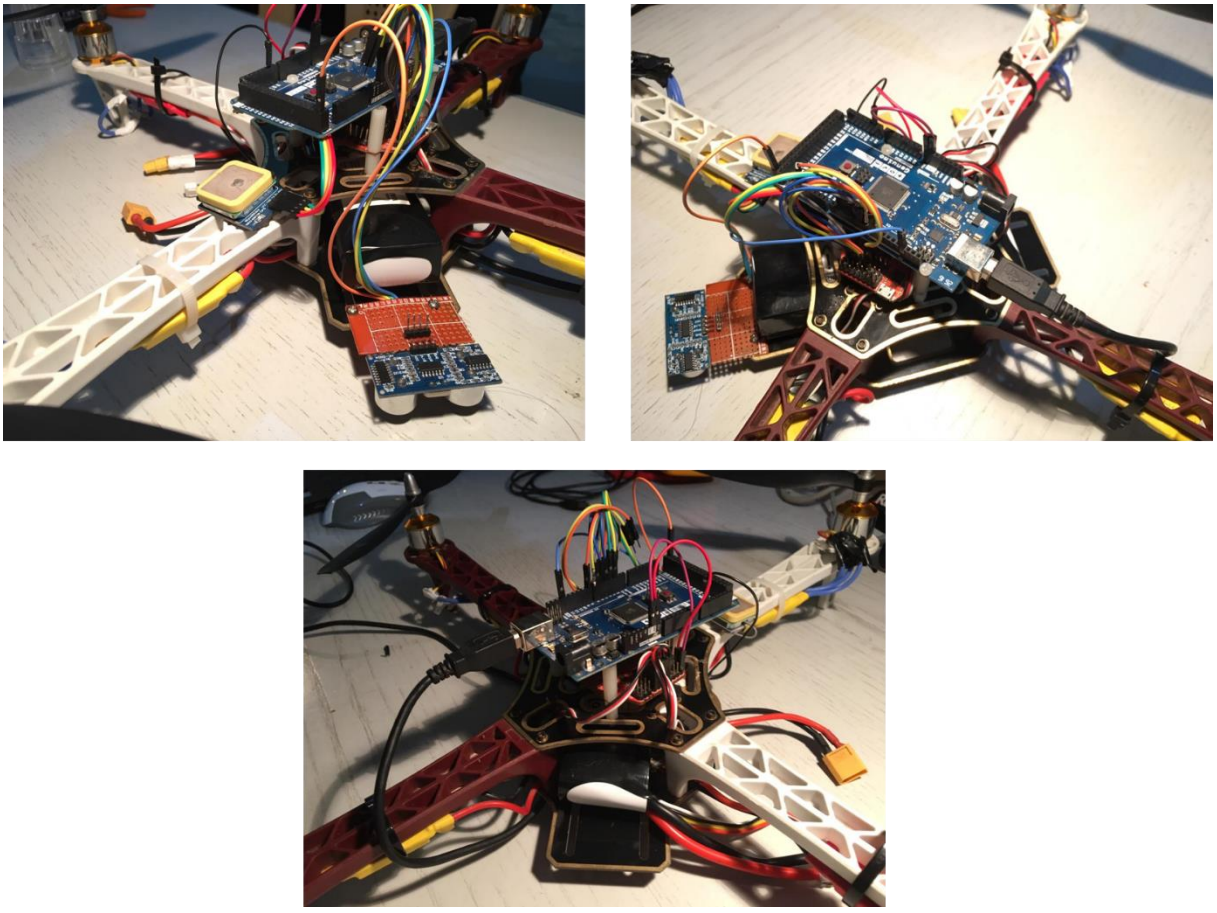

Figure 4.9. Multiwii-Arduino Physical Connections

The Arduino mega will receive the leader's position and high order commands from the leader (Naza-controlled quadcopter) via radio frequency modules (nrf24) and will operate accordingly. The procedure discussed in section 2.6.1 was followed to tune attitude PID gains in Multiwii GUI (Figure 4.8) and the best attitude parameters are summarized in the Table 4.2.

Table 4.2. Multiwii Attitude PID Tuned Gains

	k_p	k_i	k_d
Roll	2.2	0.002	5
Pitch	2.2	0.001	5
Yaw	2.0	0.003	0

Altitude and position control were a bit challenging due to the malfunctioning altimeter and GPS commands in Multiwii respectively. Thus, a standalone altitude and position controllers is to be redesigned using the high-level Arduino mega controller.

4.4.1. Altitude Control

To bypass the problem of the malfunctioning altimeter, an ultrasonic sensor was used to obtain the altitude of the quadcopter. This sensor will work well only up to 4 meters. The ultrasonic sensor transmits an ultrasound wave (via its trigger pin) at the speed of light and receive it back (via its echo pin) when it hits an obstacle. The measured time between the emitting and receiving is to be measured knowing the traveling speed of the wave then the estimated distance can be calculated. Coding wise, when transmitting a signal from an ultrasonic sensor, it is required to wait until the reflected wave is received. This will result in high waiting time (up to 10 ms) which will affect the control negatively. To mitigate this, the ultrasonic sensor was connected to an external interrupt pin on Arduino mega (Pin 2) which will execute an interrupt subroutine if the echo pin has changed from low to high or from high to low (same concept of PCINT). By doing this, there is no wasted time when waiting

for ultrasonic sensor received wave and data is requested from the ultrasonic sensor on 50 Hz. Figure 4.10 illustrates the practical block diagram of an altitude control PID algorithm.

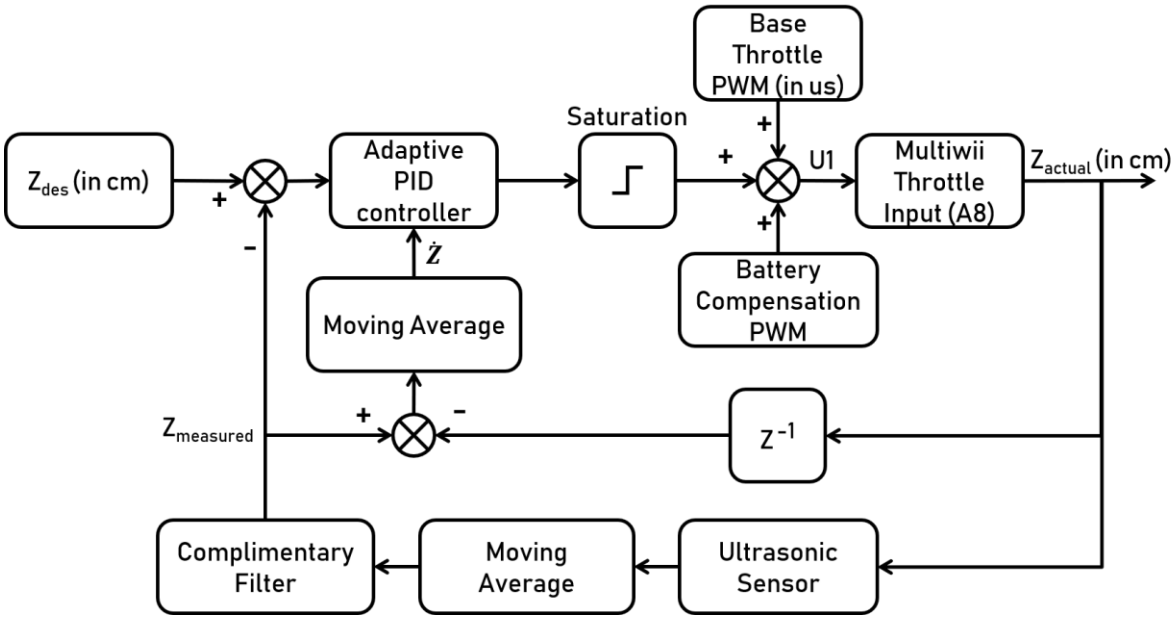

Figure 4.10. Altitude Control Block Diagram

The goal of this PID loop is to calculate the best value of U_1 in order for the quadcopter to reach its desired altitude.

4.4.1.1. Moving Average and Complimentary Filter

The feedback sensor (ultrasonic) is noisy and it is required to filter out its readings to make more sense of the altitude measurements. Moving average and complimentary filter are 2 types of digital filters that act as low-pass filters. The moving average, from its name, takes *n* number of measurements, add them together, and divide them by *n* and it has the following equation:

$$Z_k = \frac{1}{n}(Z_k + Z_{k-1}) \tag{4.2}$$

Where, Z_k is the altitude measurement at time *k*, Z_{k-1} is the altitude measurement at time *k-1*, and *n* is the window size. The window size, *n*, determines the lag between the

measured and filtered data. The higher it is, the higher the lag is. The window size in this project was selected 30 so that there is no big lag between filtered and measured data.

The complimentary filter is used as a second-step filtration algorithm for its robustness. A complimentary filter takes a portion of a certain variable and add it to the complement portion of another variable. It is mainly used to compensate a more accurate measurement with noticeably high uncertainty with a less-accurate measurement but with very low uncertainty. A major drawback of complimentary filter is its lag between filtered and measured quantities. It has the following discrete domain equation:

$$Z_k = \alpha . Z_k + (1 - \alpha) . Z_{k-1} \tag{4.3}$$

Where, α is a weighing factor between 0 and 1.

$$Z_{CF} = \alpha . Z_{CF} + (1 - \alpha) . Z_{MA} \tag{4.4}$$

Where, Z_{CF} and Z_{MA} are the altitude measured from complementary filter and moving average respectively. Here, the complimentary filter is used in an iterative manner, where its reading from the current time step will be the previous readings of the next time step. Increasing α means adding reliability to the history data of the complimentary filter and decreasing reliability over moving average readings. For the best accuracy and lowest lag,α was chosen 0.8 (or 80%). Figure 4.11 below shows the raw altitude measurements (when moving the quadcopter randomly up and down) taken from the ultrasonic sensor which have integer data type, the moving average, and complimentary filter algorithms. It can be clearly seen that the complimentary filter behaves very well with the smoothest form and lowest lag.

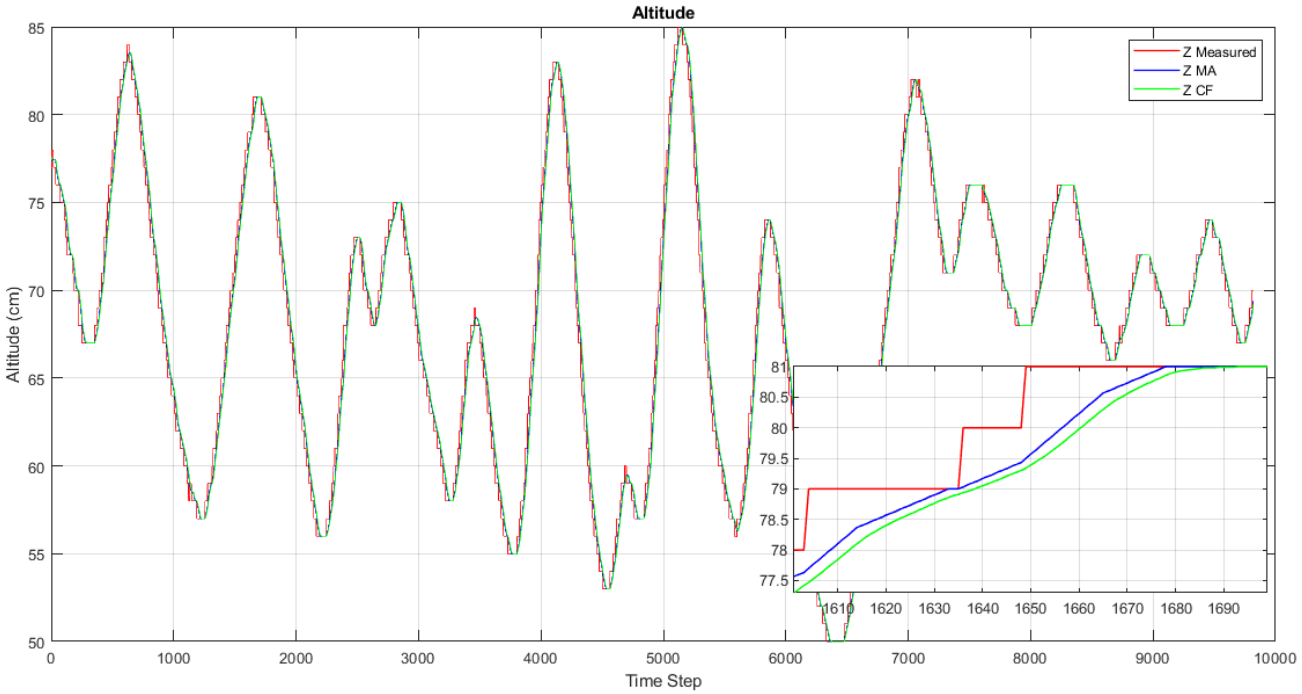

Figure 4.11. Altitude Measurement of the Multiwii-based Follower. Complimentary Filtering in Green, Moving Average in Blue and the Raw Measuremnt From The Ultrasonic Sensor in Red

4.4.1.2. Adaptive PID Controller

The adaptive PID control has the same structure of an ordinary PID controller but with varying proportional gain. As explained before, the throttling input on the Multiwii accepts PWM signal with duty cycle varying between 1000 and 2000 us with 1000 us being no thrust at all and 2000 us being the maximum throttle. After some experiments, a 50 us-change in the throttle input causes a noticeable motors' thrust. So, an adaptive P controller was implemented to scale the proportional gain proportionally to the error when far from the setpoint and a constant value within a defined range of the setpoint. It is also noteworthy that the adaptive PID controller block uses the improved version of D-controller where the derivative of the state is fed into the equation and not the error. The adaptive PID block receives the error signal between the desired and measured altitudes and also receives the

estimated vertical velocity and outputs the desired throttling PWM which will be added to the base PWM (PWM at which the quadcopter is hovering). The adaptive controller implemented has the following form:

$$e_z = Z_{desired} - Z_{measured}$$
$$Z_{pid} = k_p(e_z).e_z + k_i \int e_z\, dt - k_d.\dot{Z}_{measured} \tag{4.5}$$

Where, e_z is the error signal, $k_p, k_i\ and\ k_d$ are the PID gains. The derivative term $k_d.\dot{Z}_{measured}$ acts as a damping system, if the speed increases (quadcopter is accelerating up), this term increases negatively and slows down the vertical acceleration and vice versa. It is also noticed that k_p is as a function of the error signal. The k_p was chosen in a way to increase linearly with the error if the error is outside a certain range from the setpoint and has the following form:

$$k_p(e_z) = \begin{cases} 1.4 & if\ |e_z| < 10 \\ 1.4 + \dfrac{|e_z| - 10}{20} & elsewhere \end{cases} \tag{4.6}$$

Where, 1.4 is the base *kp* which was tuned by trial and error, and $\frac{|e_z|-10}{20}$ is the adaptive *kp* factor.

4.4.1.3. Battery Compensation

Another factor to take into consideration when designing an altitude hold controller is the battery voltage. As the voltage goes down, motors' thrust goes down too. Imagine the quadcopter is required to maintain an altitude of 1.5 meters. If an error exists, the PID calculates the required PWM which will be added to the hovering base throttle (scientifically known as *mg* which is the weight of the quadcopter) in order to maintain the quadcopter at the desired altitude. However, after a short period, an oscillatory behavior will definitely

appear no matter how good the PID is. This is caused by the voltage drop of the battery which affects the hovering throttle and the urge to compensate for this drop is a must. The used LiPo battery used has voltage of 12.6 volts when fully-charged and 11 volts when empty while the Arduino mega's analog pins can withstand maximum voltage up to 5 volts. So, a simple voltage divider circuit was designed to minimize the voltage level. Figure 4.12 shows the designed voltage divider network. The values of R1 and R2 was chosen by fixing R1 to *10 kOhms* and selecting R2 via equation (4.7).

$$V_{analog} = \frac{R_1}{R_1 + R2} . V_{Battery} \tag{4.7}$$

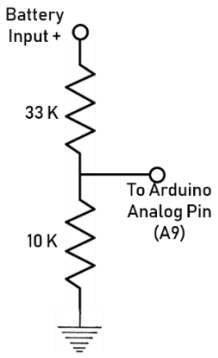

Figure 4.12. Voltage Divider Circuit

Where, V_{analog} is the input to Arduino's analog pin. By substituting 5 V to V_{analog} and 12.6 V to $V_{Battery}$, one can compute the value of R2. But by doing this, there is no safe margin considered to protect the analog pin. If the battery voltage gets a bit higher than 12.6 V, then the analog pin will read voltage higher than 5 V and gets damaged. So, $V_{Battery}$ is taken to be 21 V and R2 value is selected accordingly.

The second step was to determine experimentally the required amount of hovering PWM for different battery voltages. Figure 4.13 below shows the required hovering throttle (in blue) for different battery voltages from complete charge to complete discharge. When the battery is full, the required hovering throttle was 1480 us while when fully discharged the

required hovering throttle goes up to 1520 us. The battery compensation was assumed to be a linear function (Figure 4.13: red line) and has the form of equation (4.8). When the voltage drops to 11 V, the compensation PWM goes up to 40 and is added to the base hovering throttle of 1480 us.

$$Compensation_{us} = \left(12.6 - V_{battery}\right).25 \tag{4.8}$$

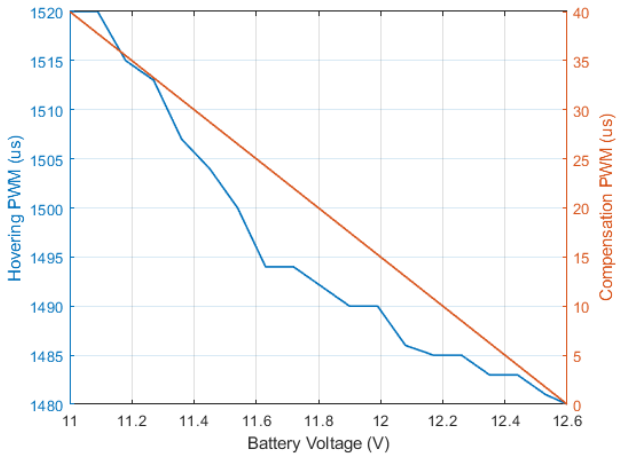

Figure 4.13. Battery Compensation PWM and The Required Hovering PWM vs. Battery Voltage

The final form of the throttling control input U_1 is then realized to be:

$$U_1 = Throttle_{Base} + PID_Z + Compensation_{Vbatt} \tag{4.9}$$

4.5. KALMAN FILTER

Kalman Filter (KF) is an optimal estimation algorithm which is intended to estimate a state *x* at time *k* by using linear stochastic difference equation with the assumption of the state *x* at time *k* is evolved from the previous state x^- at *k-1* and can be written as:

$$\boldsymbol{x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}} \quad (4.10)$$

Where, $\boldsymbol{A}$ is the state transition matrix (aka state evolution matrix), $\boldsymbol{B}$ is the input matrix relating the input to the state and $\boldsymbol{w}$ is known as process noise and is considered normally distributed with 0 mean and Q covariance. The state equation has always to be coupled with a measurement model which clarifies the relation between the state and the measured quantity. The measurement model, declared by the matrix $\boldsymbol{z}$, can be written as:

$$\boldsymbol{z_k = Hx_k + v_k} \quad (4.11)$$

Where, $\boldsymbol{H}$ is a transformation matrix which transforms the state into measurement domain, and $\boldsymbol{v_k}$ is the measurement noise vector and is also considered Gaussian with 0 mean and covariance R.

Kalman filtering has many advantages in autonomous systems design. They can be summarized as:

- It filters out noisy measurements and makes sensor data clean and more understandable.
- It updates the state based on a system kinematic model so it can predict states without directly measuring them by a sensor.
- It increases the refresh rate of some control-critical states (e.g., position).

The Kalman filter equations are divided into 2 groups; time update (aka predictor) and measurement update (aka corrector). The time update equations are responsible for projecting forward the current state and error covariance matrices to give a priori knowledge over the states for the next step. This step answers the question of where should the system be

next? The measurement update is responsible to correct the prediction phase by a reliable measurement from a sensor (e.g., a GPS). Kalman filter predictor equations can be written as:

$$\begin{aligned} \widehat{\boldsymbol{x}}_{\boldsymbol{k+1}} &= \boldsymbol{A}\widehat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{B}\boldsymbol{u}_{\boldsymbol{k-1}} \\ \widehat{\boldsymbol{P}}_{\boldsymbol{k+1}} &= \boldsymbol{A}\widehat{\boldsymbol{P}}_{\boldsymbol{k}}\boldsymbol{A}^{\boldsymbol{T}} + \boldsymbol{Q} \end{aligned} \tag{4.12}$$

Where, the *hat* symbol denotes that the variable is estimated and $\widehat{\boldsymbol{P}}_{\boldsymbol{k+1}}$ is the estimated error covariance matrix. The corrector equations can be written as:

$$\begin{aligned} \boldsymbol{K}_{\boldsymbol{k}} &= \widehat{\boldsymbol{P}}_{\boldsymbol{k+1}}\boldsymbol{H}^{\boldsymbol{T}}\left(\boldsymbol{H}\widehat{\boldsymbol{P}}_{\boldsymbol{k+1}}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R}\right)^{-\boldsymbol{1}} \\ \widehat{\boldsymbol{x}}_{\boldsymbol{k}} &= \widehat{\boldsymbol{x}}_{\boldsymbol{k+1}} + \boldsymbol{K}_{\boldsymbol{k}}(\boldsymbol{z}_{\boldsymbol{k}} - \boldsymbol{H}\widehat{\boldsymbol{x}}_{\boldsymbol{k+1}}) \\ \boldsymbol{P}_{\boldsymbol{k}} &= (\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{k}}\boldsymbol{H})\widehat{\boldsymbol{P}}_{\boldsymbol{k+1}} \end{aligned} \tag{4.13}$$

Where, $\boldsymbol{K}_{\boldsymbol{k}}$ is the Kalman gain. It is well noticed that when the estimated error covariance P is large, the Kalman gain becomes less and the updated state relies more on the previous estimated state. The Kalman gain is optimally tuned by itself and will settle to a constant value after a short period (or maybe more depending on the states' and error covariance's initial values) of time since the error covariances (R and Q) are considered to be constants.

4.6. POSITION CONTROLLER

Initially, it is necessary to understand the frame in which the quadcopter will fly in. In chapter 2, 2 frames were presented (body-fixed and inertial frames) where the body-fixed frame follows a North-East-Down (NED) (where North refers to positive x-axis, East to positive y-axis and down to positive z-axis) configuration and is attached to each quadcopter's center of mass while the inertial frame also has a NED configuration but it is fixed and tangent to the earth's curvature as shown in Figure 4.14 with the z-axis pointing towards the center of the earth.

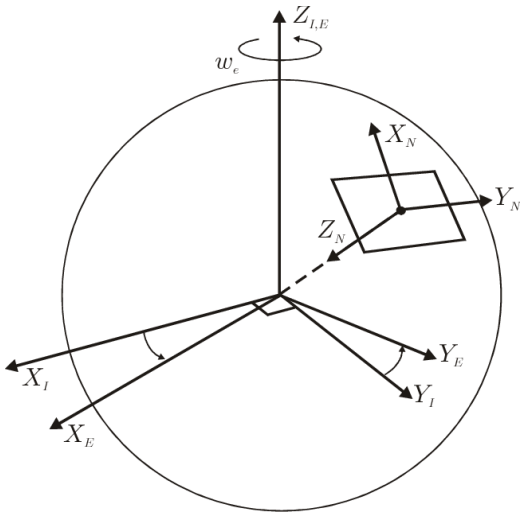

Figure 4.14. ECEF and Inertial Frames

Quadcopters will navigate in the inertial frame, so it is necessary to transform attitude actions from body to inertial frame. The GPS sends its position data in inertial frame at slow refresh rates (up to 10 Hz). The latitude and longitude information received from GPS are expressed in degrees as shown below in Figure 4.15.

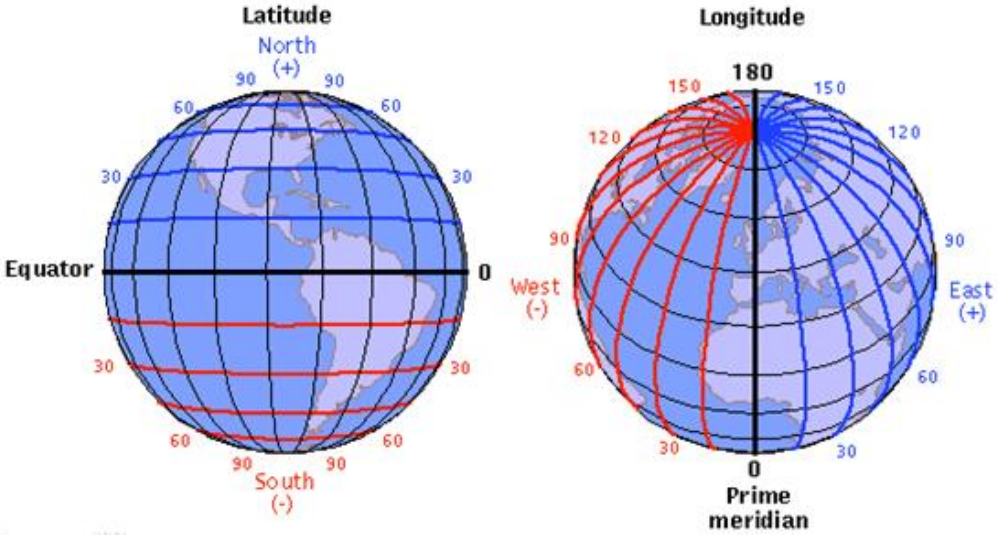

Figure 4.15. Latitude and Longitude of Earth

To transform these coordinates into a local navigation frame, it can be estimated that each latitude (*x-axis* in local navigation frame) and longitude (*y-axis* in local navigation

frame) degree is equal to 111 Km. Another factor to take into consideration is the longitude shrinking where it can be clearly seen from the image above that the distance between 2 consecutive longitude lines shrinks at the poles. The transformation from inertial frame to a local navigation frame relative to a start position can be written as equation (4.14):

$$\begin{aligned} x(m) &= (lat_0 - lat) \times \left(\frac{\pi}{180}\right) \times \left(\frac{R_{earth}}{10000000}\right) \\ y(m) &= (lon_0 - lon) \times \left(\frac{\pi}{180}\right) \times \left(\frac{R_{earth} \times cosLat_0}{10000000}\right) \end{aligned} \tag{4.14}$$

Where, lat_0 and lon_0 are the initial latitude and longitude degrees or the coordinates at where the quadcopter started, lat and lon are the current coordinates in degrees, R_{earth} is the radius of the earth and it is approximated to be *6378137 Km*, and $cosLat_0$ is the longitude shrinking factor which depends on the current latitude position. This shrinking factor is required to be calculated only 1 time since quadcopters' battery life do not allow them to travel 1 latitude or longitude degree (or 111 km). To control the position of the quadcopters, with translating the body-frame motion to inertial frame, an improved version of PD controller is used which has the form of equation (4.15).

$$\begin{aligned} x_e &= x_{ref} - x_{actual} \\ y_e &= y_{ref} - y_{actual} \\ \ddot{x}_d &= k_p.x_e - k_d\dot{x} \\ \ddot{y}_d &= k_p.y_e - k_d\dot{y} \\ roll_{adjust} &= \ddot{y}_d \cos(\psi) - \ddot{x}_d \sin(\psi) \\ pitch_{adjust} &= \ddot{x}_d \cos(\psi) + \ddot{y}_d \sin(\psi) \end{aligned} \tag{4.15}$$

And finally, the roll and pitch control inputs, respectively, can be written as:

$$U_2 = U_{2_{base}} + roll_{adjust} + U_{2_{manual}}$$
$$U_3 = U_{3_{base}} + pitch_{adjust} + U_{3_{manual}} \quad (4.16)$$

4.7. CONVOLUTIONAL NEURAL NETWORKS

A Convolutional Neural Network (CNN), which is also known as Multi-Layer Perception (MLP), is a class of deep neural networks for learning frame works. The first known CNN was introduced by LeCun in 1990 and is called LeNet [44]. CNNs, unlike feedforward Networks, are used for image recognition and classification. Image classification based on CNNs can optimally and automatically learn to extract image features effectively. Figure 4.16 illustrates the flow of CNN-based fire detection algorithms. The detection CNN has region proposals, feature extraction, and image classification functions. The first step consists of the CNN accepting an image as an input and outputs region-based proposals by 2 main layers; convolution and pooling. Then, it is the turn of region-based fire detection CNN to decide whether there is fire or not in proposal regions through convolutional, pooling, fully-connected layers.

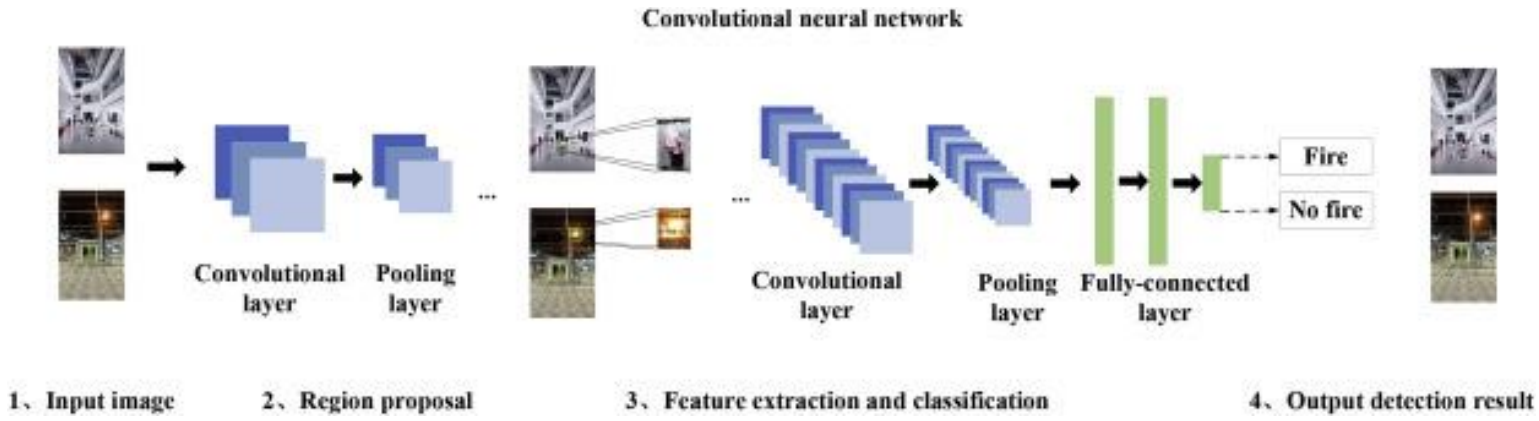

Figure 4.16. Convolutional Neural Network architecture

The convolutional layer is the most essential part in any CNN. Ordinary Neural Networks (NN) use connection weights, biases and weighted sums while convolutional layer is equipped with image transform filters also known as convolutional kernel in order to generate feature map of the original image. So, a convolutional layer is nothing but a set of convolutional kernels. The kernel slides over the whole image and computes a new pixel by some sort of weighted sum of the pixel which is floating over, in order to generate a full feature map. Equation (4.17) shows the main calculation formula of the convolutional layer.

$$y = \sum_{j=0}^{J-1} \sum_{i=0}^{I-1} w_{ij} \cdot x_{m+1,n+j} . + b, \qquad (0 \leq m \leq M, 0 \leq n \leq N) \qquad (4.17)$$

Where x is the input image of size $W \times H$, w is defined as a convolutional kernel of size $J \times I$, b is a bias value and y is the output of the feature maps. Practically speaking, w and b values are determined optimally through training process. Pooling layer samples the feature map acquired from the convolutional layer attempting to significantly reduce the overfitting, the number of parameters, and the computation in a CNN. Lastly, the fully-connected layer produces the final classification vector. It is connected to every single neuron in the layer before it decides the existence of possible matching between combination of features found and class labels.

4.8. YOLO V3

You Only Look Once Version 3 (YOLO V3) is an object detection network which is used after a feature extraction network to detect classify images with fires and smoke created by Joseph Redmon and Ali Farhadi in 2018 [45]. The feature extraction layer uses Darknet-53 network. YOLO V3 refers the idea of residual network to improve the accuracy of object detection. In addition, this network performs perfectly on detection speed for it uses a one -

stage strategy. Figure 4.17 shows the detailed architecture of YOLO V3 network. The feature extraction network, or Darknet-53, generates a small-scale feature map which 32 times smaller than the original sampled images. Typically, YOLO V3 network accepts input images with dimensions of *416 ×416* so the size of the feature map extracted from the Darknet-53 becomes roughly *13 × 13.* The goal of this small feature map is to detect large objects. Then, YOLO V3 network generates a large-scale feature map by enlarging the small-scale feature map got from the feature extraction network (Darknet-53) and concatenating with an earlier layer-feature map.

The large-scale feature map includes information of previous layers and other complex features from deeper layers which are used to detect small objects. Practically, there are 3 scales of feature maps; 8, 12 and 32 time smaller from the original image.

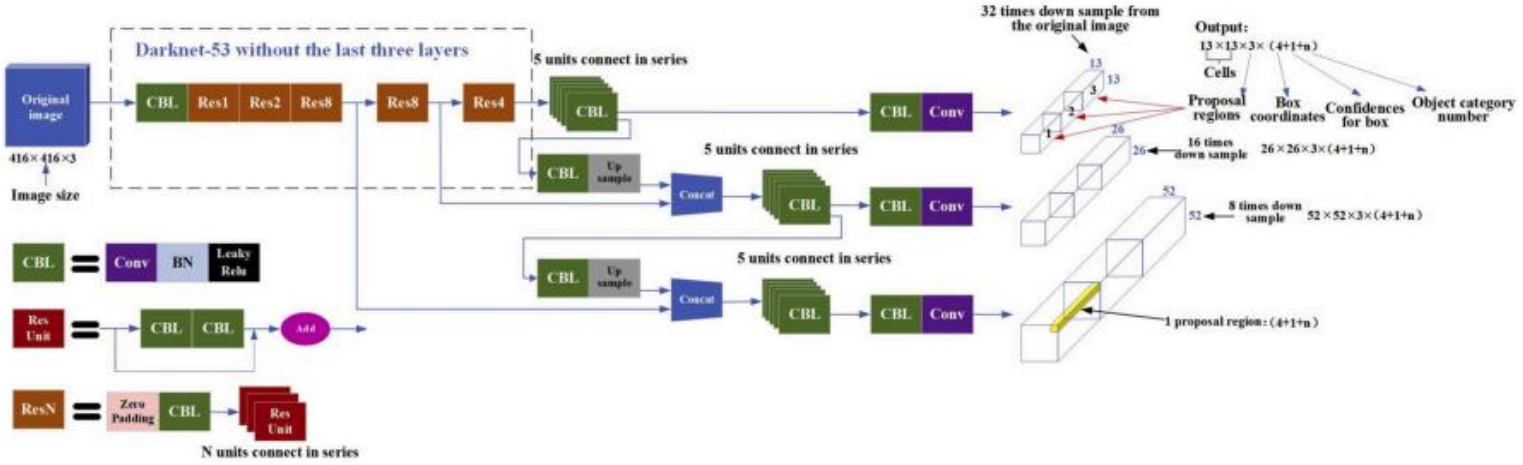

Figure 4.17. Yolo v3 Architecture

In the above figure (Figure 4.17), *N* in *ResN* clarifies that there are *N* number of *Res* units connected in series. Whereas, Concat refers to the concatenation operation which expands the dimension of the feature maps. It noteworthy that concatenation process is different than an ordinary addition operation, the normal addition does not change the dimension of the feature maps. YOLO V3 uses a sigmoid activation function to predict and

detect multilabel classifications per one bounding box. The sigmoid function has the form as in equation (4.18) and Figure 4.18.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.18}$$

The main advantage of sigmoid function is that it is a smooth version of an ordinary step function. In other words, it has a derivative everywhere. This is very important in neural networks since the fully-connected layer in a CNN computes the gradients through backpropagation to update the weights.

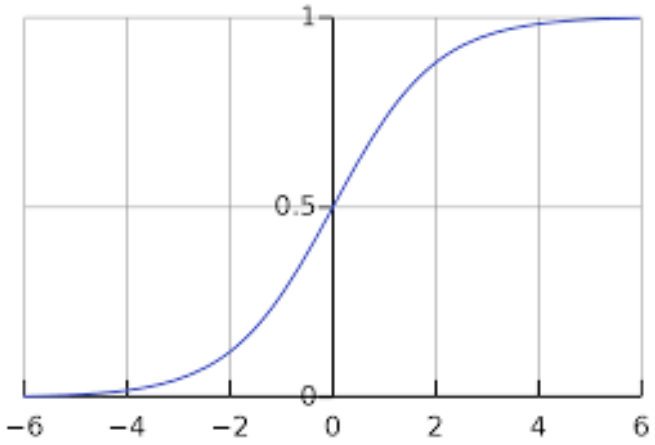

Figure 4.18. Sigmoid Activation Function

So, a YOLO V3 network can be trained to detect multiple objects in on frame. For this thesis, it was trained to detect fires and smoke in a single frame image.

4.9. TRAINING AND TESTING

Training the CNN-based algorithm requires a huge amount of data. Therefore, in this thesis, *1200* fire and smoke images were collected from different internet sources for training and *200* for testing. Using LabelImg image annotation tool, each image in the dataset is annotated with a bounding box around fire and smoke. Figure 4.19 shows the training

images setup procedure. Around *4500* fires were annotated in these *1000* images and *3265* smokes.

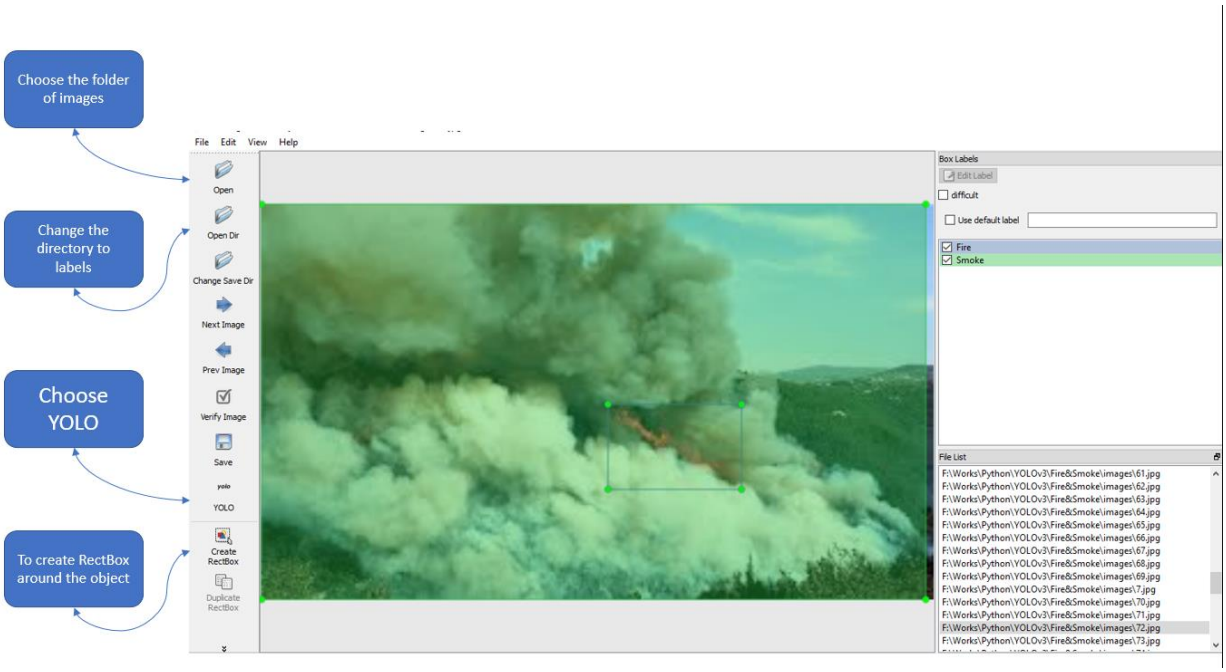

Figure 4.19. LabelImg Annotation Tool

4.10. GIMBAL CONTROL

This part will deal with designing an appropriate control method to drive the gimbal, holding the camera, towards the fire and smoke region. In other words, the camera mounted on the quadcopter will always concentrate on the center of the fire and smoke. The gimbal is driven by 2 servo motors; to control the yaw and the roll angles. Fortunately, a built-in feedback control system is already built inside each servo motor, thus it is enough to feed through the desired rotation angle to the servos. The control methodology is illustrated in Figure 4.20, gimbal design is shown in Figure 4.21, and the circuit diagram in Figure 4.22. In short, the central PC will detect the presence of fire in video frames received from the quadcopter and create bounding boxes around all fires and smokes in the frame. The center of each bounding box has 2D coordinates (*x or w: width, y or h: height)* in the image frame expressed in Pixels (px).

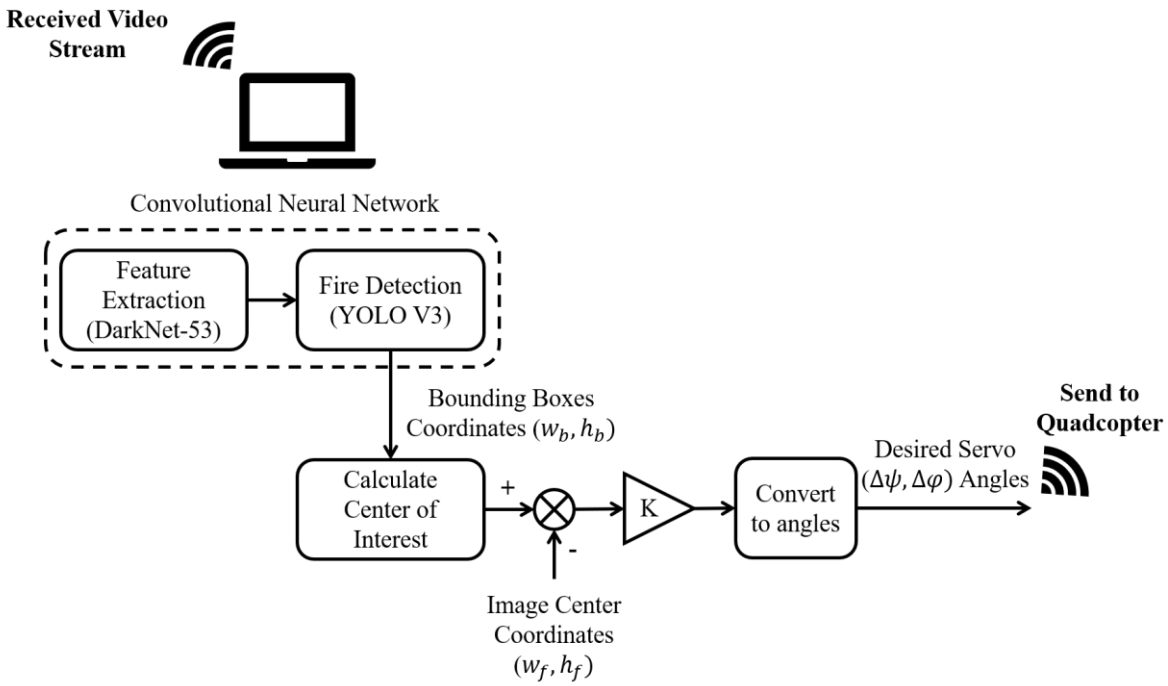

Figure 4.20. Gimbal Control Strategy

If there are multiple bounding boxes in the frame, it is required to calculate a central position between all these bounding boxes. Thus, center of interest 2D coordinates are calculated by equation (4.19):

$$w_c = \frac{1}{n}\sum_{i=1}^{n} w_i$$
$$h_c = \frac{1}{n}\sum_{i=1}^{n} h_i \qquad (4.19)$$

Where, w_i and h_i are respectively the width and height of bounding box center *i* and *n* is the number of bounding boxes in the frame. In a control system manner, and as mentioned before, it is required to keep the camera focusing on the center of interest so this variable is the setpoint of the system which will continuously be compare with the center coordinates of the frame *(w_f and h_f)*. The error signal undergoes a simple P-controller (gain *K*) to scale it and finally a mapping function is used to convert Pixels (px) into the desired

servo angle deviations to be sent back to the gimbal controller (Arduino Uno) mounted on the quadcopter. Servo deviations calculation, restrained between 0^o and 180^o in yawing and 0^o and 65^o in rolling, are shown in equation (4.20).

$$\Delta\psi = map(K \times (w_c - w_f), 0, 180)$$
$$\Delta\varphi = map(K \times (h_c - h_f), 0, 65) \quad (4.20)$$

And on the Arduino side, the required yawing and rolling is calculated through equation (4.21):

$$\psi = \psi_{current} + \Delta\psi$$
$$\varphi = \varphi_{current} + \Delta\varphi \quad (4.21)$$

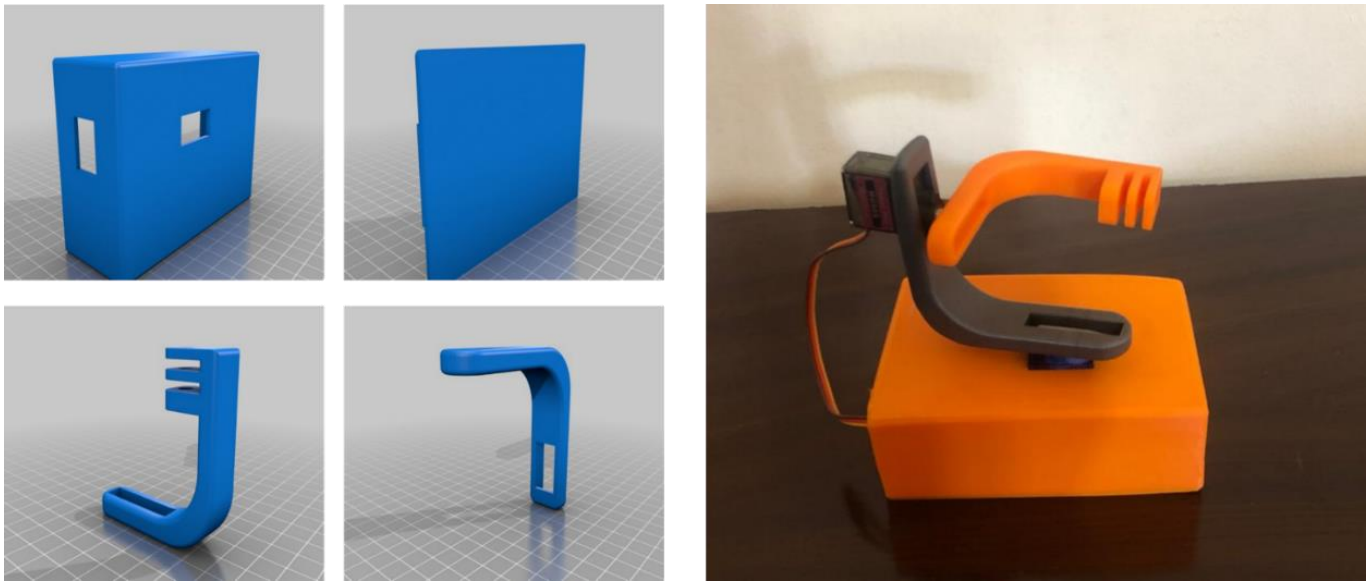

Figure 4.21. Gimbal Designed 3D and Physical Models

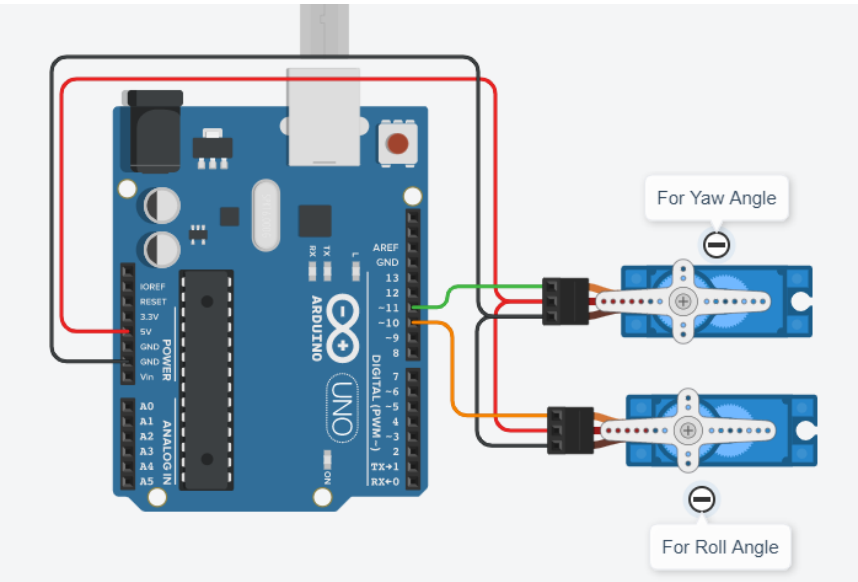

Figure 4.22. Servo Motor Control Circuit Connections

4.11. CONCLUSION

This chapter presented the hardware design of the leader and followers quadcopters. In addition, it briefly discussed the Kalman filter and position control implementation. Also, training a neural network model of a fire detection system via a feature extraction deep network (Darknet-53) and image recognition (YOLO V3) models was explained. And, a P-control-driven gimbal controller was designed

CHAPTER 5. NON-TECHNICAL ASPECTS

5.1. INTRODUCTION

In this chapter, all components of each quadcopter will be presented with their cost in Local/Global markets in addition to the approved method of management in this work. The last part will cover the ethical, social, and environmental effect on society, as well as the sustainability will be covered.

5.2. ECONOMICAL/FINANCIAL

The thesis is composed of 3 quadcopters (Leader, follower 1, follower 2), as mentioned before. Table 5.1 lists the chosen items for each quadcopter and their cost according to Local/Global markets. On the other hand, Table 5.2 shows the Engineering staff costs.

Table 5.1. Material Cost

item	price/ piece	LEADER	follower 1	follower2	
Turnigy Heavy Aerial Lift frame	$40.84	1	0	0	
DJI F450 Frame	$69.99	0	1	0	
PROPDRIVE V2 2826 motor	$15.44	0	4	0	
A2212/13T motor	$8.86	4	0	0	
Propeller	$2.99	2	2	0	
Ready To Fly 30A	$10.00	0	4	0	
AFRO ESC 30A	$13.50	4	0	0	
MULTIWII Flight controller	$22.95	1	0	0	
NAZA Flight Controller	$209	0	1	0	
PARROT AR Drone	$180.00	0	0	1	
Power Distribution Board	$5.91	1	0	0	
Parrot AR Drone 2.0 1500mAh	$21.98	0	0	1	
HRB 5000mAh 11.1v 50C	$69.99	1	0	0	
Ublox Neo 6m GPS	$15.00	1	1	1	
Ultrasonic Sensor HC-SR04	$0.66	1	0	0	Costs all in all $1601.99
Arduino Mega 2560	$40.30	1	0	0	X 1.3(with overhead) $2082.587
Arduino Uno	$33.24	0	0	1	
FCONEGY 5500mAh	$39.97	0	1	0	
Tools	$10.00	1	0	0	
Camera with Gimbal	$609.00	1	0	0	
Total ($)		**910.07**	**441.7**	**250.22**	

Table 5.2. Engineering Staff Cost

Task	MM	Qualification	Salary/MM	Total Salary
Assembling quadcopters	0.25	Eng.	$1000	$250
Control system (Hardware and Software development)	1	Eng.	$1000	$1000
Testing quadcopters	0.5	Eng.	$1000	$500
Integration quadcopters process	0.5	Eng.	$1000	$500
Swarm Quadcopter Testing	0.5	Eng.	$1000	$500
Project Management	1	Eng.	$1000	$1000
Programming (training)	5	Eng.	$500	$2500
Testing	1	Eng.	$500	$500
Total Man Power Costs				$6750

5.3. PROJECT MANAGEMENT

The thesis is divided into 3 main phases: Gathering Data, Simulation, and implementation. Thesis started by gathering data and literature surveys. This phase started in July 27 until Aug 24, in other words, it took 20 days. After that, the simulation phase started by modeling and designing the High/Low level control of quadcopters. The Simulation took 60 days, from Aug 24 to Nov 13. Finally, the implementation phase covers all hardware, testing, and system validation processes. The period of this phase extended from Feb 28 to Aug 24, in a total of 86 days. All phases and tasks are presented in Figure 5.1 below.

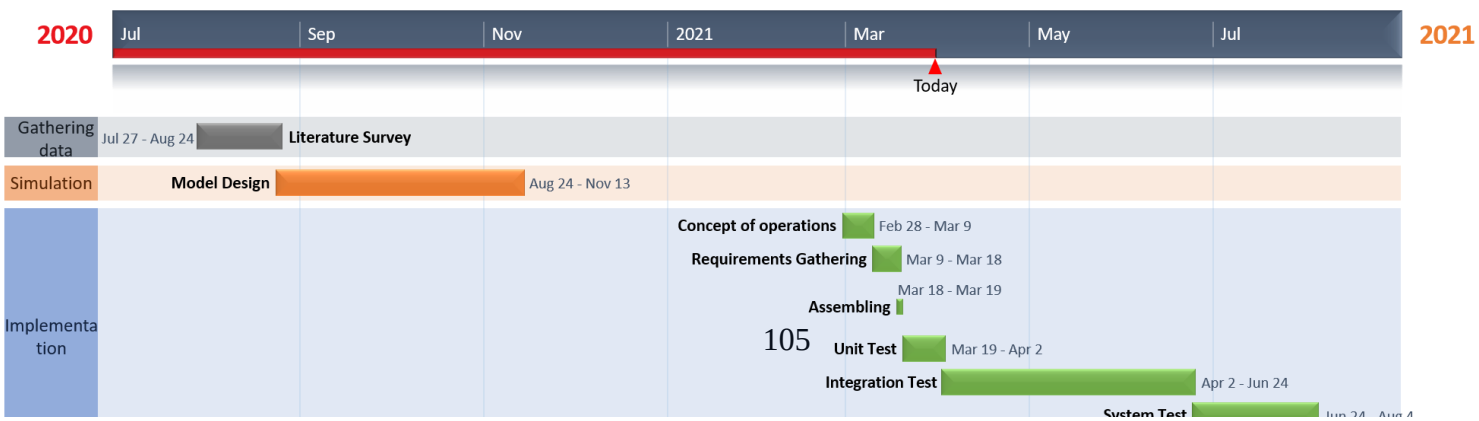

Figure 5.1. Thesis Gantt chart

Whereas the fire detection system is also divided into 3 main phases: Gathering Data, Simulation, and Software. Thesis started by gathering data and literature surveys. This phase started in July 27 until 16 Oct, in other words, it took 60 days. Then, the simulation phase started by modeling and designing the High/Low level control of quadcopters. The Simulation took 20 days, from Aug 24 to Sep 18. At the end, the Software phase covers programming, Training, and testing processes. The period of this phase extended from Sep 18 to Jun 2, in a total of 184 days. All phases and tasks are presented in Figure 5.2.

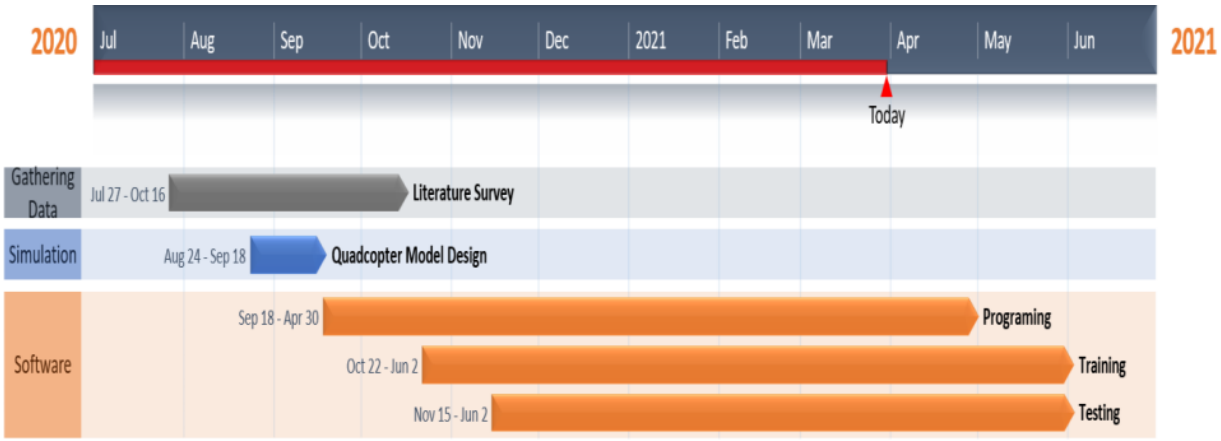

Figure 5.2. Thesis Gantt Chart

5.4. ETHICAL AND SOCIAL

Ethics on definition are the moral values that guide the performance of an action or administer a person's behavior.

5.4.1. Quadcopter

All types of drones face different ethical issues that vary depending on the drone's category (Recreational, Civilian and Commercial, and Military) [46]. This thesis' quadcopters are categorized as civil and commercial aiming for fire foreign surveillance. The use of drones

for this aim raises ethical problems related to the collection of citizen's data located inside or nearby the forest. This issue leads to an invasion of privacy. This adds to the noise of the operating drone problem. Also, having too many quadcopters used for surveillance purpose in a limited space might be challenging.

5.4.2. Camera

Taking photographs or live stream via camera mounted on a drone for a diverse purpose, may not have an ethical issue related to them. However, if a person accidentally takes a picture of another person, it can be an invasion of privacy. Hence, photography can only be evaluated based upon the action, intention, and the consequence of the actions intended by the photographer [47].

5.4.3. Neural Networks

Due to the hidden and complex implementation of algorithm elements or the intermediate layer of statistically trained 'neurons', there is a topic that proposes a cognitive dilemma and brings related ethical issues for deep learning (neural networks) [48]. Otherwise, computer scientists and software engineer themselves are increasingly worried about the lack of transparency of AI and deep learning [49]. This opacity of algorithmic black boxes poses a direct and long-term challenge to lawmakers and policymakers. At the end, in Thilo Hagendorf's (2020) review of twenty-two recommendations on the ethical guidelines of AI by governments and NGO's, transparency (in general, AI systems) is second only to privacy, fairness, and accountability. In other word, it is the fourth important theme of 22 themes [50].

5.5. ENVIRONMENTAL AND SUSTAINABILITY

Quadcopter is an environmentally friendly electric machine, that is widely used as an alternative of polluting ones. For instance, a study conducted in 2018, compared ten environmental impact categories (listed in Table 5.3) as well as the number of particulates produced during 1 Kilometer of delivery between Drone (used as delivery), and motorcycle (gasoline and electric motorcycle) in Korea [51]. The study showed that the traditional motorcycle (gasoline motorcycle) had the highest Global Warming Potential (GWP) followed by the electric motorcycle, while, drones had the lowest GWP. Therefore, electric motorcycles produced more particulates than gasoline and drone, as shown in Figure 5.3 below:

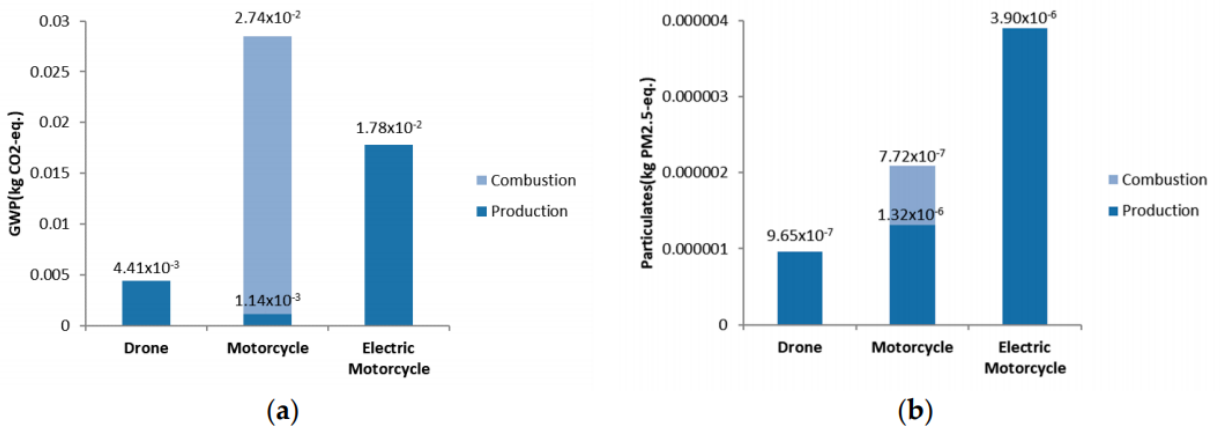

Figure 5.3. GWP and particulates of 1 Kilometer delivery by electric motorcycle, gasolian motorcylce, and drone. **(a)** GWP of 1 Km delivery; **(b)** $PM_{2.5}$ of 1 Km delivery

Table 5.3. Environmental Impact of Selected Categories

Impact Category		Unit
ADP	Abiotic Depletion Potential	1/year
AP	Acidification Potential	kg SO_2-eq
EP	Eutrophication Potential	kg PO_4^{3-}-eq

FAETP	Freshwater Aquatic Ecotoxicity Potential	kg 1,4 DCB-eq.
GWP	Global Warming Potential	kg CO_2-eq.
HTP	Human Toxicity Potential	kg 1,4 DCB-eq.
MAETP	Marine Aquatic Ecotoxicity Potential	kg 1,4 DCB-eq.
ODP	Ozone Depletion Potential	kg CFC_{11}-eq.
POCP	Photochemical Oxidants Creation Potential	kg ethylene-eq.
TETP	Terrestrial Ecotoxicity Potential	kg 1,4 DCB-eq.
$PM_{2.5}$	Particulates Matter	kg $PM_{2.5}$-eq.

Finally, the results show that the environmental improvement effect would be higher with the use of drones.

5.6. STANDARDS

5.6.1. Quadcopter Standards

According to Lebanese Regulations and Laws, there is no drone laws issued. However, it does not mean that you can take flight wherever you like. In fact, it is possible that legislators will oppose the use of drones in general. To avoid any problem or have the drone taken away, contacting the Lebanese Directorate General of Civil Aviation (DGCA) will be recommended. In the other hand, the U.S.'s Federal Aviation Administration's has some rules for drone usage. We list here some of them in Table 5.4 below:

Table 5.4. FAA's Model Aircraft Rules

Nb.	Rules
1	Avoid flying within five miles of an airport
2	Keep the drone within visual line-of-sight
3	Fly at or below 400 feet

4	Fly during daylight or civil twilight
5	Fly at or under 100 mph
6	Yield right of way to manned aircraft
7	Do not fly directly over people
8	Do not fly from a moving vehicle, unless in a sparsely populated area

Particularly, UAVs are regulated by several standards and laws around the world. According to them, quadcopters are dispersed into classes, categories, and labels. One of the most famous regulations are Eu's number 2019/947 and 2019/945 standards from European Union Aviation Agency (EASA) [52] [53]. According to them, quadcopters used in this thesis are classified as "Open-A3".

Plus, there are several developed UAVs standards (i.e., ISO 21384-1, ISO 21384-2, ISO 21384-3, ISO 21384-4), and other incomplete so far (i.e., ISO/IEC AWI 22460-2, ISO/IEC AWI 4005-1, etc.) [54].

5.6.2. Neural Networks Standards

In general, there are many standards that define and classify Artificial intelligence, machine learning, deep learning, and Neural network. Some of them are approved and other still under study. Table 5.5 lists some of IEEE Standards related to Neural Networks (completed and incomplete) [55].

Table 5.5. IEEE Standards Related to Neural Networks

ID	Title
IEEE 3333.1.1-2015	Standard for Operator Interfaces of Artificial Intelligence
IEEE 3333.1.2-2017	Standard for the Perceptual Quality Assessment of Three-Dimensional (3D) and Ultra-High-Definition (UHD) Contents

P3333.1.3	Standard for the Deep Learning-Based Assessment of Visual Experience Based on Human Factors
P3333.1.1	Standard for Quality of Experience (QoE) and Visual-Comfort Assessments of Three-Dimensional (3D) Contents Based on Psychophysical Studies
P2941.1	Standard for Operator Interfaces of Artificial Intelligence

5.7. CONCLUSION

As mentioned before, all components used in each quadcopter were presented with their cost in local/global markets in addition to the approved method of management in this work. Finally, the ethical, social, and environmental effects on society, and the sustainability were presented in this chapter.

CHAPTER 6. RESULTS

6.1. INTRODUCTION

This chapter will present the results associated with hardware design. Multiple scenarios were done to test and validate the efficiency of individual and swarm controllers. Also, the NN training and testing results will be shown as well.
Training accuracy, training losses and training precision percentages will be plotted. In addition, the servo motor-controlled angles will be plotted vs. fires and smoke frames in a video stream.

6.2. KALMAN FILTER

The Kalman filter was able to significantly reduce the noise of the GPS readings with also estimating the velocity of the quadcopter. The first test was done to figure out the values of the process noise Q and measurement noise R. In fact, these variables are tuned via trial and error with small reference to the GPS's datasheet. Figure 6.1 below shows 2 scenarios, stationary and moving quadcopter. It can be seen that when the quadcopter is stationary, the GPS has, in its best cases, a 3.5 meters deviation in the latitude direction and approximately 2 meters deviation in the longitude direction. According to the GPS's datasheet, the standard deviation of the used module is 5 meters. Thus, the measurement covariance is taken for *3.5* meters in both directions. The process noise was chosen in a way that the estimated position follows the measured one quickly with the least noise. Increasing Q increases the lag between the estimated and measured variables. This, Q was chosen *0.001* for latitude and longitude.

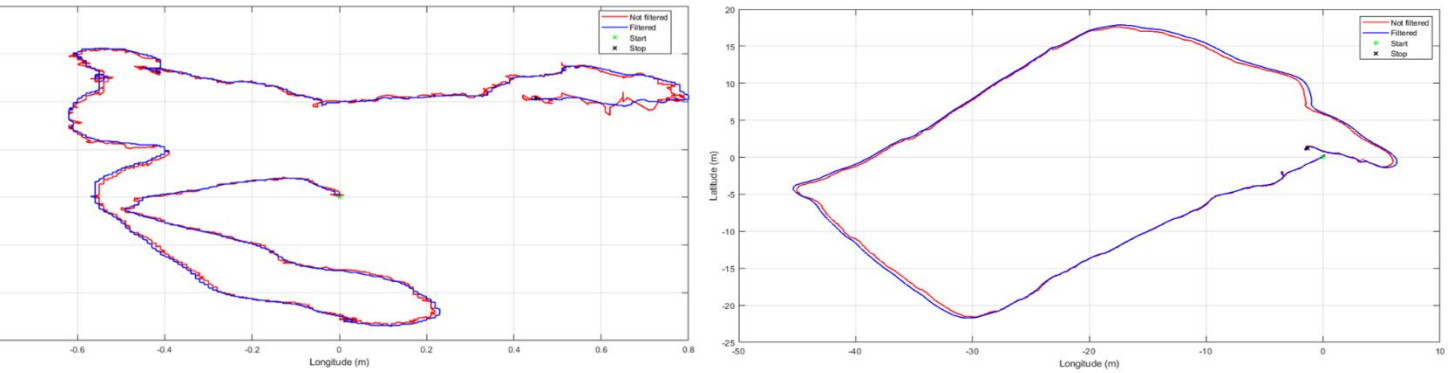

Figure 6.1. Kalman Filter Algorithm. Stationary Quadcopter (Left), a Small Quadcopter Tour (Right)

Another test was done to validate the estimated speed. The quadcopter was put in a car which is moving at known speeds between 0 and 40 Km/h or 0 to 11 m/s with several accelerations and decelerations. Figure 6.2 below shows the estimated speed of the quadcopter when it was moving at an average speed of 20 Km/h or 5.5 m/s.

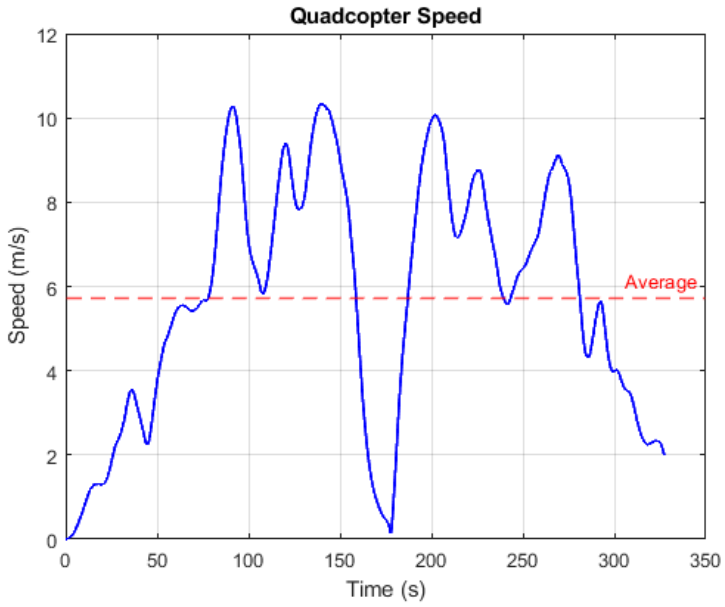

Figure 6.2. Estimated Quadcopter Speed

6.3. NAZA-BASED QUADCOPTER

As mentioned before, the leader quadcopter is equipped with a Naza flight controller and GPS/compass modules. The scenario here was to test if the leader quadcopter can

individually track a setpoint with adjusting its bearing. The setpoint was set, relative to its first position, to (-18 m, -18 m) or approximately 25 m of direct distance. The leader was able to adjust its heading and reach the desired waypoint in around 12 seconds. However, a slight deviation from the desired goal position can be noticed and it can be neglected.

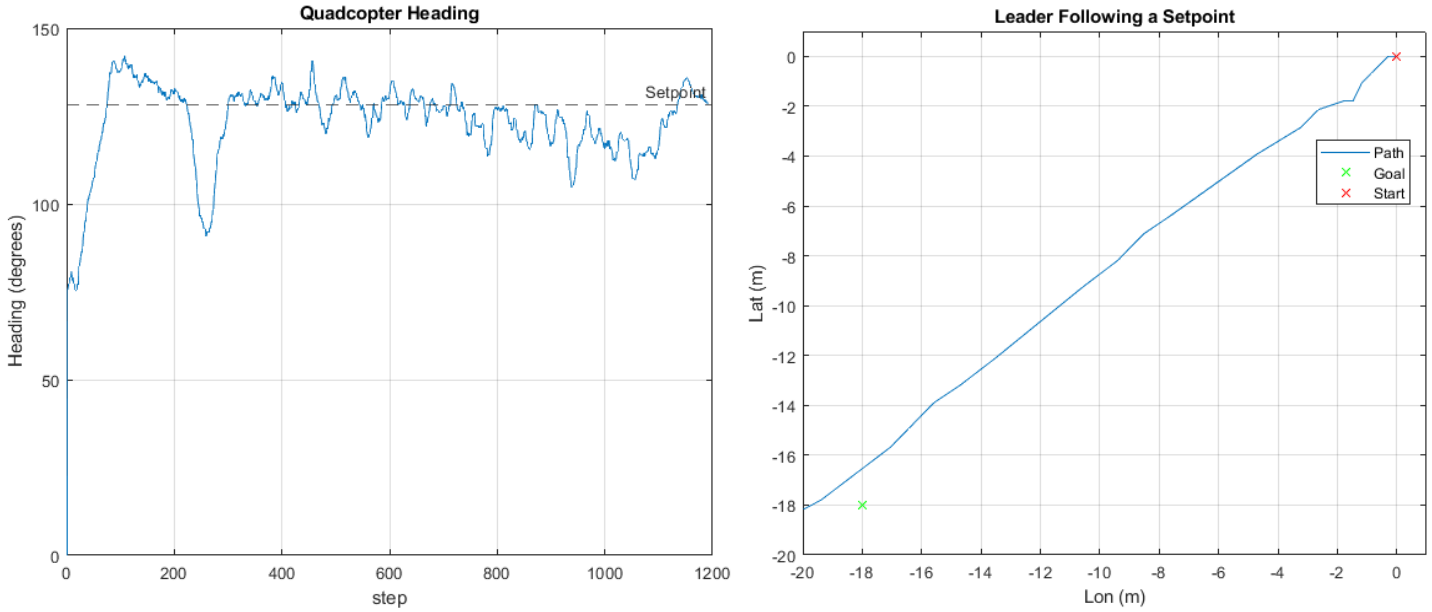

Figure 6.3. Calculated Heading (Left), Actual Path (Right)

6.4. AR-DRONE

The goal was to control the AR Drone using AR Drone Simulink Development-Kit V1.1 (by David Escobar Sanabria) on MATLAB (version 2014). AR Drone has been able to receive just two high-level commands (takeoff and land), in addition to waypoint tracking via the serial monitor of Arduino UNO. But many problems have been encountered, such as unexpected information loss and malfunctioned software (caused by limited capability of available PC). The AR drone kit did not meet the need for video extraction to detect fire through its camera. To solve this problem, Python is adopted instead of MATLAB. Via Python, full control was achieved with accessing the AR Drone Camera, thus, real video stream and image capturing. But the libraries do not support the extraction of sensors' data (e.g., accelerometer, magnetometer, etc.), thus, no full access to data for extraction. Finally,

to solve this problem, NodeJS was adopted instead of Python. Via NodeJS, full control and data extraction were achieved. However, flying according to leader data was not achieved yet. All this work has been done with ruined battery (maximum flight time does exceed 1 min).

To eliminate this issue, AR Drone was connected directly to a power supply with a cable of 1.5 mm^2 thickness and length of 10 m. But this method failed due to completely unknown reasons, even if the length of the cable was decreased or the voltage was increased to avoid the voltage drop and other consequences.

6.5. SWARMING

After a lot of unsuccessful trials to coordinate between the leader and the followers because of hardware issues, the expected trajectory of the 2 followers was simulated and is shown in Figure 6.4.

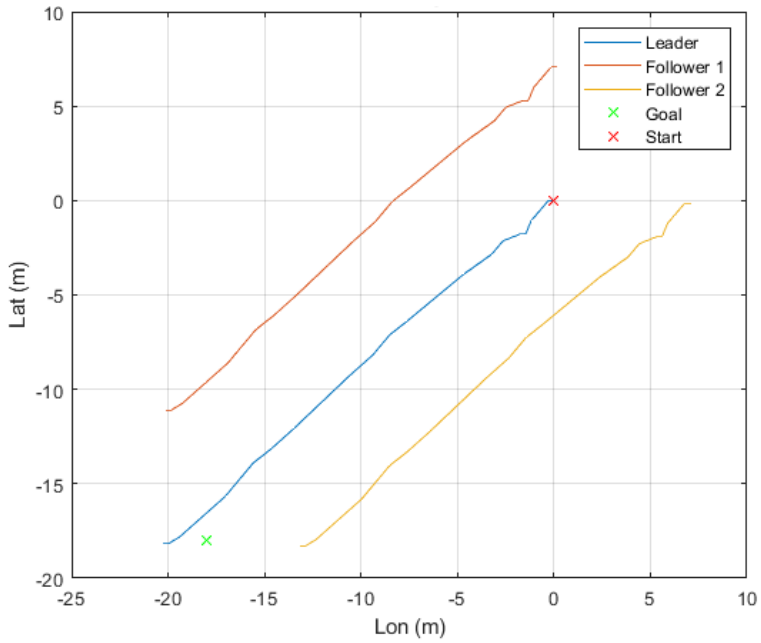

Figure 6.4. Quadcopter Swarm Following a Desired Setpoint

6.6. NEURAL NETWORKS

Training the Neural Network model to detect fires and smoke in a live video stream was done by iterating 30000 times. Figure 6.5 shows the essential training parameters. Training losses and average training losses should be as low as possible and if the average training losses becomes less than 0.0607, the training can be stopped and best results are acquired. The fire and smoke detection accuracy graphs shows that the detection accuracy after approximately 1000 iterations becomes 20% and after 30000 iterations, it settles at approximately 98% for fire and 95% for smoke. The precision percentage is the ratio of true positive images (images containing fire and smoke) to the total number of positive predictions. Intersection over Union (IoU) percentage is the accuracy of the detection algorithm given a dataset. The F1-score is the model's accuracy over a give dataset. It is a combination between the precision percentage and recall percentage of the model. And finally, the mean average precision percentage from its name, is the average of the training precision percentage.

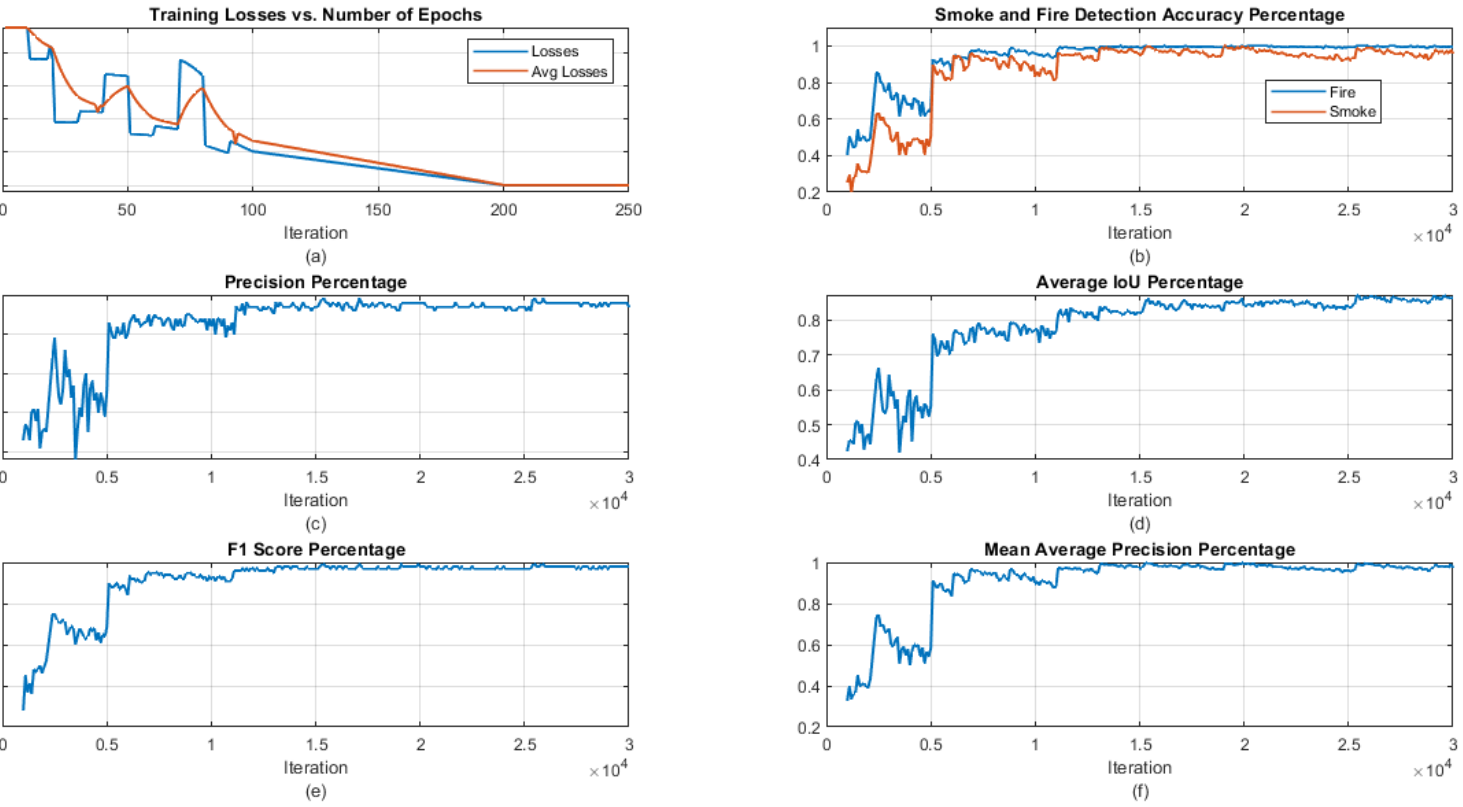

Figure 6.5. Training Losses (a), Detection Accuracy (b), Training Precision Percentage (c), Average Intersection Over Union (d), F1 Score (e), and Mean Training Precision (f)

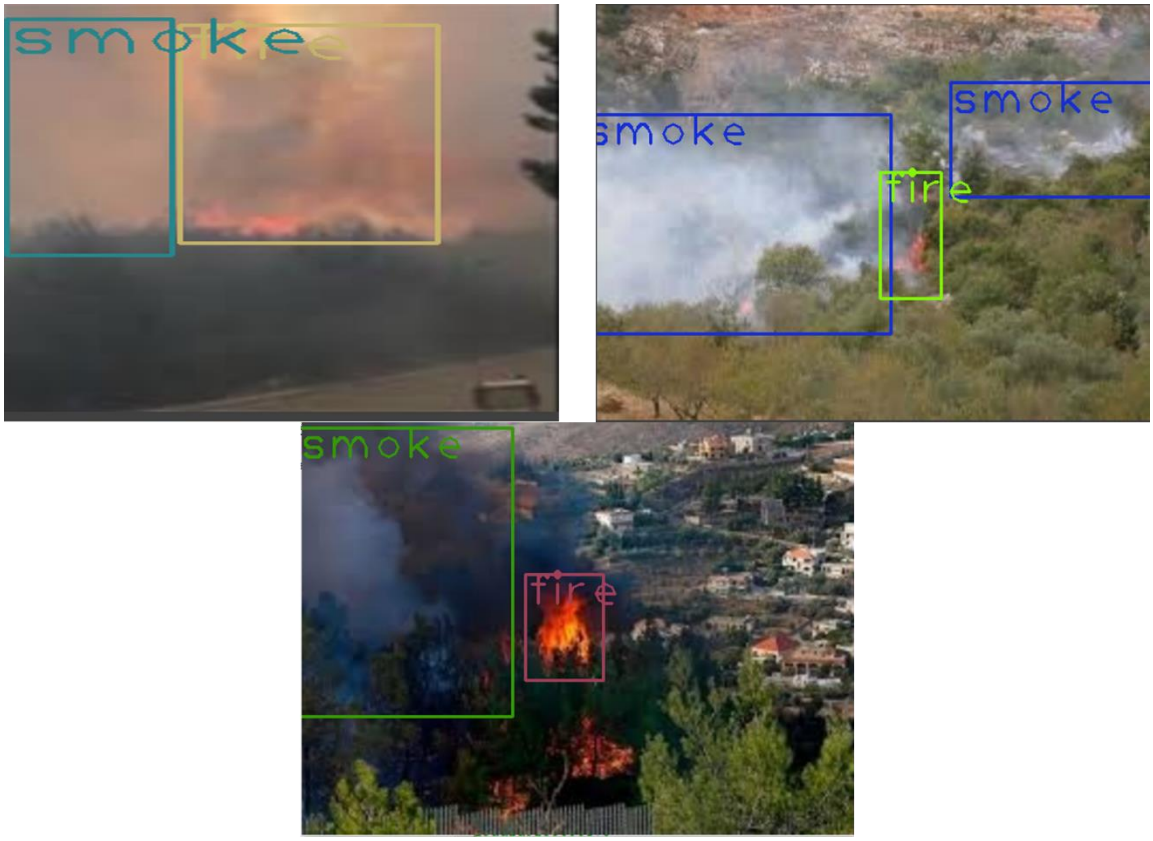

Figure 6.6. Detected Fires and Smokes in a Video Frame

6.7. GIMBAL

The gimbal is receiving its commands from python which calculates the desired servo deviation angles based on the fires and smoke center coordinates. The gimbal's center position is at (roll = 30º, yaw = 30º). Table 6.1 below shows the desired rolling and yawing gimbal angles calculated from the central x and y coordinates and Figure 6.7 shows the gimbal desired angles received from python.

Table 6.1. Gimbal Roll and Yaw Data Associated to the frame coordinates

X	Y	Roll	Yaw
0	0	30	30
42	47	32	44
50	50	32	44
47	52	30	49
42	32	42	63
23	50	42	111
54	46	45	103
61	59	38	83
53	70	24	77
49	37	33	79
39	45	36	99
48	54	33	102
50	62	24	102
50	50	24	102
47	52	22	107
61	59	15	87
51	74	70	13
33	43	75	43
70	35	85	7
60	47	87	-11
61	50	87	-30
48	40	94	-26
38	48	95	-4
32	47	97	28
53	70	80	22
42	43	85	36
38	44	89	57
50	39	96	57

62 45	49 53	96 94	35 44

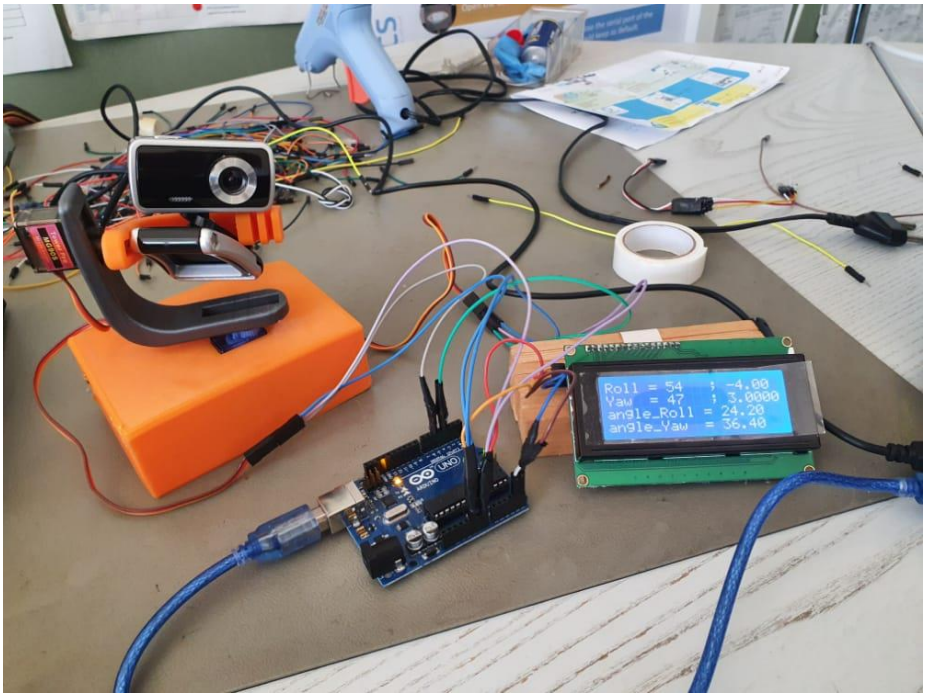

Figure 6.7. Gimbal Input Controlled from PC

6.8. CONCLUSION

This chapter presented the hardware results of individual and swarming tests done. The leader was expected to calculate its bearing and follow a predefined setpoint then the followers were expected to follow it with keeping a safe distance between them. The training showed satisfactory results in terms of fire and smoke detection with accuracy up to 98%.

CHAPTER 7. CONCLUSION

7.1. GENERAL CONCLUSION

This thesis presented the design and implementation of an autonomous swarm of 3 quadcopters. The first stage covered the simulation via Simulink of the quadcopters flying with triangular shape and following a predefined path along with an offline obstacle avoidance algorithm based on artificial potential fields. The second part covered the hardware and software implementation of 3 system-different quadcopters where each has its own functions. However, this project gave unsatisfactory results in terms of swarming but with good enough individual controllers. Dealing with different systems was the main interrupting factor that led to coordination errors. Hence, it was more feasible to fully deploy the system on 1 drone only and then generalize it on 2 other drones. All the work done in this thesis can be summed by:

- A successful coordination between the leader (Naza-based) and 1 follower (MultiWii-based) quadcopters was done. The 2 drones were able to arm together and receive high order commands such as takeoff or land. Hence, the first part (communication between drones) to achieving a functional swarm of quadcopters is done.
- A sudden ESC failure in the MultiWii-based follower occurred which prevented the continuity of the cooperative control. Where, all the used parts in this project were borrowed and any replacement part cannot be bought because of the economic crisis in Lebanon.

- Sudden sensor failures occur in the MultiWii board and a calibration is needed every 2 consecutive power ONs. This can be because the MultiWii flight controller has been settled unused for approximately 4 years which resulted in sensor malfunctions.
- The plan from the beginning was to use a ready-to-use flight controllers and focus only on the swarming part. However, the position and altitude controller functions in the MultiWii flight controller were malfunctioned which was a very time-consuming to redesign a new altitude and position controllers.
- Besides the very good results obtained from Kalman filter when filtering GPS data only, it was very difficult to increase the position refresh rate by fusing the accelerometer measurements with GPS ones. This depends on the accelerometer quality which was not that good at all.

For the fire detection system, the project implementation of a forest fire and smoke detection system based on Darknet-53 and YOLO v3 object detection networks. A camera was mounted on a quadcopter, which was not shown intact, that will transmit back a live video stream of the target forest in order for the detection system to calculate the desired gimbal deviations to maintain the video stream back at the middle of the fires and smoke. Training the neural network model was held on Google Colab's GPU while the detection was performed on a CPU-based processor. The system was able to detect fires and smoke in a video stream successfully. Through analyzing and validating the output results, it can be concluded that:

- Running the fire and smoke detection system on a GPU is 7 times faster than running it on a CPU. A significant fps drop resulted from running the system on CPU.
- Training YOLO v3 models is much flexible compared to other types of models such as RCNN, faster RCNN etc.

- Training the model on 5000 iterations gave significantly unsatisfactory detection accuracy of 55% for smoke and 65% for fires. The model needed to be trained at a minimum number of 11,000 iterations to exceed the 90% accuracy line.

7.2. FUTURE WORK

From the valuable experience gained from this project, a complete drone-based fire detection system will be designed focusing only on 1 drone and by adding the work from the fire detection algorithms using neural networks group. In addition, by providing accurate drone position and heading in case of fire presence, it is possible to coordinate with a survey engineering team unit to precisely locate the fire on a map and take the necessary actions.

And the fire detection system can be further developed in many ways. The propagation speed of the fires can be estimated for extra monitoring information that can help locals to better take decisions. In addition, the system can be trained to detect humans in video frames and send special kind of alert to the stakeholders.

APPENDIX A.

A.1. PARAMETERS ESTIMATION

In this section, physical parameters estimation is handled.

A.1.1. Thrust and Drag Coefficients

Thrust and drag constants are calculated according to the propellers used. UIUC university provides a datasheet for different types of propellers. In this work, GWS 1045 propellers are used (10 inches in length and 4.5 inches pitch).

$$thrust\ constant = b = \frac{C_T \times \rho_{air} \times R^4}{4\pi^2}$$

$$drag\ constant = d = \frac{C_H \times \rho_{air} \times R^5}{4\pi^2}$$

With:

$C\_T: Thrust\ coefficient\ =\ 0.137$

$C\_H: Drag\ coefficient\ =\ 0.0092$

$\rho_{air}: Air\ Density = 1.135 \frac{kg}{m^3}$

$$R: Rotor\ Radius = 5\ inches = 0.127\ m$$

A.1.2. Drag Force

$k_{fx} = k_{fy} = k_{fz} = \frac{1}{2} C_D \rho_{air} S$

$C_D: Drag\ coefficient = 0.63$

$$S: Reference\ area, known\ as\ orthographic\ projection\ of\ surface = 0.006\ m^2$$

A.1.3. Moment of Inertia

The quadcopter is assumed to be symmetrical. It can be seen as a main sphere with its center coincides with the center of mass connected into 4 smaller spheres (motors) through rectangle rods. The inertia around each axis can be calculated as:

$$I_{xx} = I_{yy} = \frac{2}{5} \times M \times R^2 + 2 \times m_{motor} \times l^2$$

$$I_{zz} = \frac{2}{5} \times M \times R^2 + 4 \times m_{motor} \times l^2$$

Where:

$R: Radius\ of\ the\ main\ sphere\ (m)$

$l: Quadcopter\ arm\ length$

$M: Mass\ of\ the\ main\ sphere\ (kg)$

$m_{motor}: Mass\ of\ the\ motors\ (kg)$

REFERENCES

[1] Y.R. Tang, Y. Li, "Dynamic modeling for high-performance controller design of a UAV quadrotor," *IEEE Int. Conf. Informat. Automation* , pp. 3112-3117, 2015.

[2] M. H. a. H. R. K. X. Huo, ", Attitude stabilization control of a quadrotor UAV by using backstepping approach, Mathematical Problems in Engineering," , 2014, pp. 1-9..

[3] .. J. G. Leishman, " "Principles of Helicopter Aerodynamics with CD Extra,"," Cambridge University Press, London, , 2006.

[4] .. Wikipedia, "“Curtiss-Wright Vz-7,Wikipedia, The Free Encyclopedia,” 2012,," Available: http://en.wikipedia.org/wiki/Curtiss-Wright_VZ-7, Accessed on 01/10/2012.

[5] "Quadcopter Arena," 12 December 2018. [Online]. Available: https://quadcopterarena.com/the-history-of-drones-and-quadcopters.

[6] A. T. a. S. McGilvray, ", Attitude stabilization of a VTOL quadrotor aircraft, IEEE Trans. Control Syst. Technol. vol. 14, pp. 562-571,," 2006.

[7] K. F, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Field Robot* , no. 29, pp. 78-315, 2012.

[8] Yun B, Chen BM, Kai YL and Tong HL, "Design and implementation of a leader-follower cooperative control system for unmanned helicopters," *Control Theory Appl,* vol. 8, pp. 8-61, 2010.

[9] Nathan PT, Almurib HA et al, " A review of autonomous multi-agent quadrotor control techniques and applications," *4th Int. Conf. on Mechatronics (Kuala Lumpur),* pp. 1-7, 2011.

[10] Insurance Information Institute, "Facts + Statistics: Wildfires," 2020.

[11] ""Lebanon forest fires,"," 13 October 2019. . [Online]. Available: [Online]. Available: Wikipedia: https://en.wikipedia.org/wiki/2019_Lebanon_forest_fires#cite_note-4..

[12] "BBC," 15 October 2019. [Online]. Available: https://www.bbc.com/news/av/world-middle-east-50063429.

[13] A. NOLTE, "Swarming Behavior: Drone Swarms to Survey Unknown Environments," *Northrop Grumman* , Nov 9th 2020.

[1 "University of Michigan Engineering," 15 Nov 2019. [Online]. Available:

4] https://www.youtube.com/watch?v=XNF_Sddlgy4&ab_channel=UniversityofMichiganEngineering.

[15] Lewis MA and Tan K-H, "High precision formation control of mobile robots using virtual structure Autonomous Robot," vol. 4, pp. 387-403, 1997.

[16] Balch T and Arkin RC, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Automa,* vol. 14, 1998.

[17] Roldão V, Cunha R, Cabecinhas D, Silvestre C and Oliveira P, "A leader-following trajectory generator with application to quadrotor formation flight Robot," *Auton. Syst,* 2014.

[18] P. K. C. Wang, "“Navigation strategies for multiple autonomous mobile robots moving in formation,”," (1991) J.Robot.Syst...8,177-195,.

[19] Weitz LA, Hurtado JE and Sinclair AJ, "Decentralized cooperative-control design for multivehicle formations," *Guidance, Control and Dynamics,* p. 31, 2008.

[20] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal Robotics Research,* vol. 5, pp. 90-99, 1986.

[21] I. F. I. f. P. P. a. I. Affairs, "FOREST FIRES IN LEBANON: A RECURRING DISASTER," AUB, Beirut, 2019.

[22] Nguyen Xuan-Mung and Sung Kyung Hong, "Robust adaptive formation control of quadcopters based on a leader–follower approach," *International Journal of Advanced Robotic Systems,* pp. 1 - 11, 2019.

[23] Falin Wu , Jiemin Chen and Yuan Liang, "Leader-Follower Formation Control for Quadrotors," in *IOP Conf. Ser.: Mater. Sci.,* 2017.

[24] Mu B, Zhang K, and Shi Y, "Integral sliding mode flight controller design for a quadrotor and the application in a heterogeneous multi-agent system," *IEEE Trans Ind Electron,* vol. 64, no. 2, p. 9389–9398, 2017.

[25] Khaled AG and Youmin Z, "Formation control of multiple quadrotors based on leader-follower method," in *International conference on unmanned aircraft systems (ICUAS)*, Denver, CO, 2015.

[26] Mercado D A, Castro R, and Lozano R, "Quadrotors flight formation control using a leader follower approach," in *European control conference (ECC)*, Zurich, 2013.

[27] Reagan L. Galvez, Gerard Ely U. Faelden, Jose Martin Z. Maningo, Reiichiro Christian S. Nakano, Elmer P. Dadios, Argel A. Bandala, Ryan Rhay P. Vicerra and Arvin H. Fernando, "Obstacle Avoidance Algorithm for Swarm of Quadrotor Unmanned Aerial

Vehicle Using Artificial Potential Fields," in *IEEE Region 10 Conference (TENCON)*, Malaysia, 2017.

[28] Milad Nazarahari, Esmaeel Khanmirza, and Samira Doostie, "Mult-iobjective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Systems with Applications*, vol. 115, pp. 106-120, 2019.

[29] Derek J Bennet and Colin R McInnes, "Distributed control of multi-robot systems using bifurcating potential fields," *Robotics and Autonomous Systems* , vol. 58, no. 3, pp. 256-264, 2010.

[30] A. Rodriguez, "Analysis and Design of Multivariable Feedback Control Systems," in *CONTROL3D*, 2002.

[31] Gonzalez-Sanchez, M., L. Amezquita-Brooks, E. Liceaga-Castro and P. d. C. Zambrano-Robledo, "Simplifying quadrotor controllers by using simplified design models," *Decision and Control (CDC)* , pp. 4236-4241, 2013.

[32] T. Bresciani, "Modelling, Identification and Control of a Quadrotor Helicopter," Department of Automatic Control Lund University, Lund, Sweeden, 2008.

[33] A. Das, K. Subbarao and F. Lewis, "Dynamic inversion with zero-dynamics stabilisation for quadrotor control," *IET Control Theory and Applications*, vol. 3, no. 3, p. 303– 314, 2009.

[34] T. Dierks and S. Jagannathan, "Output Feedback Control of a Quadrotor UAV Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 50-66, 2010.

[35] Z. Zuo, "Trajectory tracking control design with command-filtered compensation for a quadrotor," *IET Control Theory & Applications*, vol. 4, no. 11, pp. 2343-2355, 2010.

[36] A. Benallegue, A. Mokhtari and L. Fridman, "High-order sliding-mode observer for a quadrotor UAV," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 4-5, pp. 427-440, 2008.

[37] Sivaranjini Srikanthakumar, Cunjia Liu and Wen-Hua Chen, "Optimization-based Safety Analysis of Obstacle Avoidance Systems for Unmanned Aerial Vehicles," Department of Aeronautical and Automotive Engineering, Loughborough University, 2015.

[38] E. BUBER and B. DIRI, "Performance Analysis and CPU vs GPU Comparison for Deep Learning," in *International Conference on Control Engineering & Information Technology (CEIT)*, Istanbul, 2018.

[39] DJI, *Naza-M V2 Quick Start Guide*, DJI, 2015.

9]

[40] K. Eyeone, "Arduino Project Hub," Arduino, 29 July 2019. [Online]. Available: https://create.arduino.cc/projecthub/kelvineyeone/read-pwm-decode-rc-receiver-input-and-apply-fail-safe-6b90eb.

[41] E. Gakstatter, "What Exactly Is GPS NMEA Data?," GPS World, 4 February 2015. [Online]. Available: https://www.gpsworld.com/what-exactly-is-gps-nmea-data/.

[42] pawelsky, "DJI NAZA GPS communication protocol - NazaDecoder Arduino library," RC Groups, 11 September 2013. [Online]. Available: https://www.rcgroups.com/forums/showthread.php?1995704-DJI-NAZA-GPS-communication-protocol-NazaDecoder-Arduino-library.

[43] U-Blox, *u-blox 6 Receiver Description Including Protocol Specificaions,* Zuercherstrasse: U-Blox, 2018.

[44] Y. LeCun et al, "Handwritten digit recognition with a back-propagationnetwork," *inProc. Adv. Neural Inf. Process. Syst.,* p. 396–404, 1990.

[45] Synced, "Medium," Medium, 27 March 2018. [Online]. Available: https://medium.com/syncedreview/the-yolov3-object-detection-network-is-fast-fcceae0ab650.

[46] D. M. Marshall, Introduction to Unmanned Aircraft Systems, Second ed., CRC Press, Taylor and Francis Group, 2012, pp. 29-49.

[47] "Photography ethicscenter," 25 April 2018. [Online]. Available: https://www.photoethics.org/content/2018/5/31/photography-ethics-and-why-they-matter.

[48] B. Long, "Blog of the APA," 13 August 2020. [Online]. Available: https://blog.apaonline.org/2020/08/13/the-ethics-of-deep-learning-ai-and-the-epistemic-opacity-dilemma/.

[49] P. Voosen, "ScienceMag," 6 Jul 2017. [Online]. Available: https://www.sciencemag.org/news/2017/07/how-ai-detectives-are-cracking-open-black-box-deep-learning.

[50] T. Hagendorff, "The Ethics of AI Ethics: An Evaluation of Guidelines," pp. 99-120, 28 july 2020.

[51] Jiyoon Park, Solhee Kim, Kyo Suh, "A Comparative Analysis of the Environmental Benefits of Drone-Based Delivery Services in Urban and Rural Areas," p. 7, 20 March 2018.

[5 E. U. A. Agency, "EASA," 13 jan 2021. [Online]. Available:

2] https://www.easa.europa.eu/document-library/easy-access-rules/easy-access-rules-unmanned-aircraft-systems-regulation-eu.

[53] "DRONERULES," july 2020. [Online]. Available: https://dronerules.eu/en/recreational/eu_regulations_updates.

[54] "ISO," [Online]. Available: https://www.iso.org/search.html?q=drone&hPP=10&idx=all_en&p=0&hFR%5Bcategory%5D%5B0%5D=standard.

[55] I. SA, "IEEE STANDARDS ASSOCIATION," [Online]. Available: https://standards.ieee.org/search-results.html?q=neural+network&facetValue=4294967245.