ESTABLISHING AUTONOMOUS AUS-QUADROTOR

A THESIS IN MECHATRONICS

Presented to the faculty of the American University of Sharjah
College of Engineering
in partial fulfillment of
the requirement of the degree

MASTER OF SCIENCE

by
YOUNES AL-YOUNES
B.S. 2006

Sharjah, U.A.E
October 2009

We approve the thesis of Younes Al-Younes

Date of signature

Dr. Mohammad Al-Jarrah, Professor, Department of Mechanical Engineering Thesis Advisor	
Dr. Ali Jhemi, Assistant Professor, Department of Mechanical Engineering Thesis Co-advisor	
Dr. Khaled Assaleh, Associate Professor, Department of Electrical Engineering Graduate Committee	
Dr. Mamoun Abdel-Hafez, Assistant Professor, Department of Mechanical Engineering Graduate Committee	
Dr. Aydin yesildirek, Associate Professor, Department of Electrical Engineering Graduate Committee	
Dr. Khalifa Harib, Associate Professor, Department of Mechanical Engineering Graduate Committee (External Examiner)	
Dr. Rached Dhaouadi, Coordinator Mechatronics Engineering Graduate Program	
Dr. Hany El Kadi, Associate Dean, College of Engineering	
Dr. Yousef Al Assaf, Dean, College of Engineering	
Mr. Kevin Mitchell, Director, Graduate & Undergraduate Programs	

ESTABLISHING AUTONOMOUS AUS-QUADROTOR

Younes Al-Younes, Candidate for the Master of Science Degree

American University of Sharjah, 2009

ABSTRACT

Vertical takeoff and landing Unmanned Aerial Vehicles (VTOL-UAVs) are superior to their counterparts fixed wing UAVs for urban applications. The objective of this thesis is to establish a VTOL UAV platform to complement the ongoing research activities in the area of Autonomous cooperative multi-agent system at AUS. The chosen platform for this research is a commercial remotely controlled quadrotor.

First, the mathematical model will be developed and flight simulator will be designed using Matlab/Simulink environment. The simulator will be used to develop attitude stabilization flight control laws taking into account simulated noisy inertial measurements. Then hover autopilots will be designed using classical PID and LQR controllers. These autopilots will be used to simulate basic trajectory tracking flights. These autopilots and subsequent trajectory generation and tracking will be used as a benchmark for developing nonlinear autopilots.

The proposed nonlinear autopilots will be based on Adaptive Integral Backstepping Controller (AIBC) for controlling the quadrotor. The recursive Lyapunov methodology in the backstepping technique will ensure the system stability, the integral action will increase the system robustness against disturbances and model uncertainties, and the adaptation law will estimate the modeling errors caused by assumptions in simplifying the complexity of the quadrotor model. In addition, a Lyapunov-based Velocity Controller (LVC) is introduced to work side by side with the AIBC for hover and like-hover control.

Fuzzy logic methodology will be investigated with the objective of boosting the performance of the AIBC by scheduling its parameters based on the state of the vehicle. Since the quadrotor is electrically powered, minimizing the control effort while keeping track of the trajectory will be investigated using least mean square algorithm. Furthermore, for path following the Tangent Heading Algorithm (THA) is proposed.

To achieve the goals of this research one needs to develop in the process a Ground Control Station (GCS) for data logging, monitoring, and controlling the AUS quadrotor while performing its planned tasks.

The system will be developed using hardware in the loop simulation to test and verify the development of the flight controls and the trajectory tracking ability. These simulations will validated through subsequent flight experiments.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENT

At the beginning, I would like to thank Allah for giving me the opportunity to exist in this world between a wonderful family.

Sincere thanks go to the founder of Mechatronics center Dr. Jarrah and our program coordinator Dr. Dhaouadi and to the friendliest doctor I met ever Dr.Ali, for their direction, support, enthusiasm and confidence.

I will not forget Mohammed saleh and his motivation to join this university. Long will I remember the days of Hussien Sadjadi, one of the smartest guys I met ever. I would like to thank him for his support during my stay at AUS. I would like to tell him " ya tanaket zitoon keep going I would like to see you a professor one day"

I will not forget the best and strangest guy in this world, amir jafari. We made a lot of fun together and with Ibrahim, the new comer, we watched the movies until midnight, thank you amir for spreading the smile everywhere.

I am grateful to many dear friend Wassim jaber, and to the moments we spent together, especially the last summer at AUS, you are really a brother, thank you so much.

To my parents, you have worked day and night in the hope that your son will make you proud one day. Thank you for your love, care, sacrifice and wisdom that you gave me throughout my life, the words cannot express my feeling toward what you did for me. I will never forget my sisters and brothers for their aid and support to achieve my goals.

And to Neda … get ready for the new life!

DEDICATION

It is an honor to dedicate this work to my lovely parents and to my sweet heart, Neda.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Recently, unmanned aerial vehicles (UAVs) gained increased attention for their civil and scientific research applications. Progress in sensing elements, control theories, and data processing techniques lead to the rapid development of the UAV applications in different fields. Important class of UAVs is the Vertical Take-Off and Landing (VTOL) rotorcraft that has four lift-generating propellers known as the quadrotor. [1]

Two of the propellers spin clock wise and the other two counter-clockwise. Controlling the vehicle motion can be achieved by varying the relative speed of the propellers. Figure 1-1 illustrates the possible motions caused by this variation.

The symmetry of the design allows for a centralization of control systems and payload. The four rotors of a quadrotor provide a larger amount of thrust than a typical helicopter which allows for larger payloads and computing platforms, especially important in UAV applications.

The quadrotor has six degrees of freedom (6DOF), yaw, pitch, roll, x (longitudinal motion), y (lateral motion), and z (altitude). These 6DOF must be controlled using only four actuators. This provides an interesting area of study into how to decouple control to allow for stable flight and control of all 6DOF. [2]

Due to the complexity of the dynamic model, and the non-linearity of the quadrotor system, its control becomes quite a challenge. The stability issue in hover is the main objective for most studies conducted in this field. Sliding mode control, basic PID, and LQR control are some of the control methods that have been applied to the quadrotor platform.

In most of the studies the control algorithm is based on a linearized version of the model. The linearization will restrict the control to be valid for a certain conditions like the hover flight condition. The drifting and fluctuating of the quadrotor in hover flight were some of the problems that faced the researcher in their work. So, a nonlinear controller could be suitable to overcome these challenges in the hover and non-hover flights.

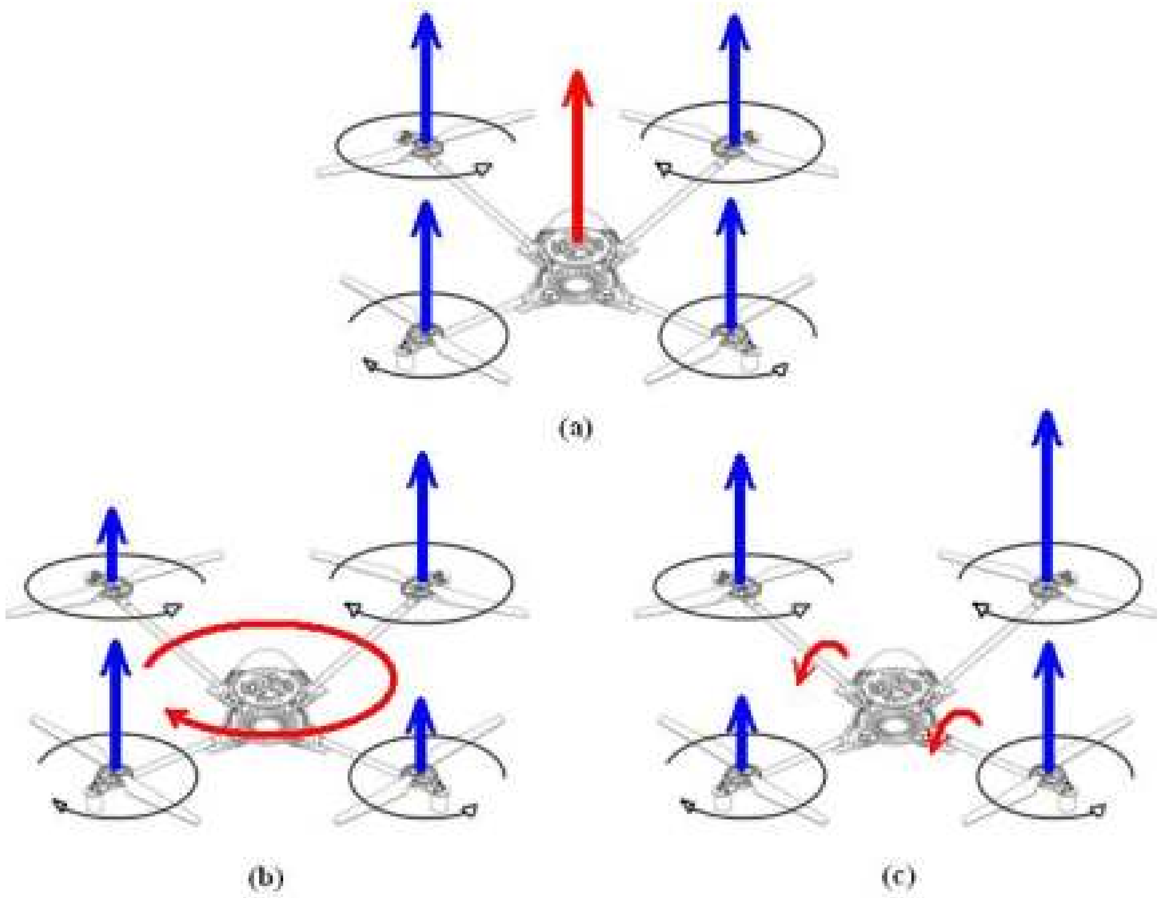


Figure 1-1: (a) Total Thrust, (b) Yaw Motion, and (c) Roll/Pitch Motion

1.2 Quadrotor History [3]

1.2.1 Gyroplane No.1

The first rotary-wing aircraft rose vertically from the ground in the late summer of 1907, the Gyroplane No.1 built by Louis and Jacques Breguet in association with Professor Charles Richet.

Rotorcraft Structure:

Basically, the Gyroplane No.1 constructed from a rectangular central chassis of steel tubes supporting the powerplant and the pilot; from each corner of this chassis there extended arm, at the extremity of extended arm a fabric-covered 4-blade biplane rotor, making a total of 32 small lifting surfaces. One pair of diagonally opposed rotors rotated in a clockwise direction, the other pair moving anti-clockwise. The structure of the Gyroplane No.1 depicted on Figure 1-2

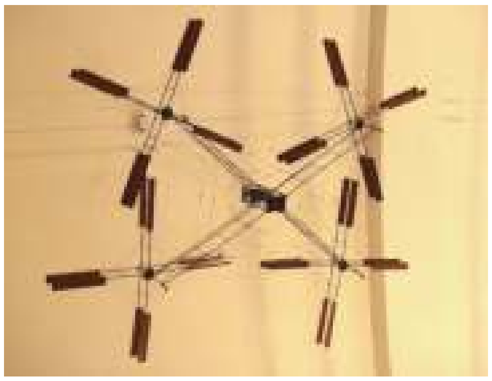


Figure 1-2: Gyroplane No.1

Flying Tests:

The first flight of the Breguet machine was in Douai, 24 August and 19 September 1907; on this occasion the aircraft rose to about 0.60m. Take-off to some 1.50m was achieved during a test on 29 September, and similar heights were reached in several subsequent tests.

Evaluation:

The Breguet-Richet aircraft was neither controllable nor steerable in a horizontal plane.

1.2.2 Oemichen No. 2

In France, 1920, Etienne Oemichen began rotary-wing experiments by building a total of six different machines. His second machine flew unassisted on 11 November 1922.

Rotorcraft Structure:

The Oemichen No. 2 had an "X"- shaped, tubular frame with a wide two-bladed rotor at the end of each arm, as shown in Figure 1-3. For control and lateral movement, eight small propellers were used: five horizontal propellers with variable and reversible pitch for lateral stability, another propeller at the nose for steering, and another pair of pushers for forward motion.

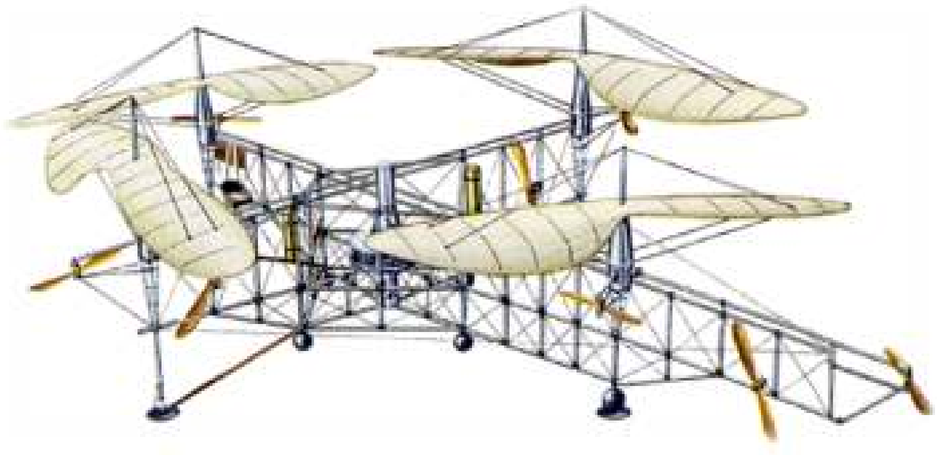


Figure 1-3: Oemichen No. 2

Flying Tests:

In 1923, the Oemichen No. 2 was able to remain airborne for several minutes, and on 14 April 1924, it established the first rotary wing distance record which is 360m. On 4 May, it completed the first 1km closed circuit flight by a rotary wing vehicle in 7 minutes 40 seconds. Maximum endurance was 14 minutes.

Evaluation:

Despite the fact that it was able to demonstrate sufficient controllability and power in ground effect for this historic flight, it was not a practical flying machine.

1.2.3 de Bothezat

In January 1921, Dr. George de Bothezat and Ivan Jerome developed their quadrotor flight machine.

Rotorcraft Structure:

The 1678kg "X"-shaped structure supported an 8.1m diameter six-blade rotor at each end of the 9m arms. At the ends of the lateral arms, two small propellers with variable pitch were used for thrusting and yaw control. A small lifting rotor was also mounted above the 180hp Le Rhone radial engine at the junction of the frames. Each rotor had individual collective pitch control to produce differential thrust through vehicle inclination for translation. Figure 1-4 shows the structure of de Bothezat vehicle.

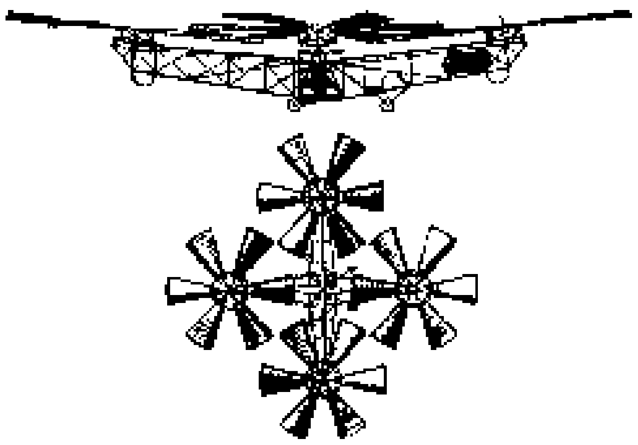


Figure 1-4: de Bothezat

Flying Tests:

About 100 flights were made by the end of 1923 at what would eventually be known as Wright Field near Dayton, Ohio, including one with three "passengers" hanging onto the airframe. Although the contract called for a 100m hover, the highest it ever reached was about 5m.

Evaluation:

After expending $200,000, de Bothezat demonstrated that his vehicle could be quite stable and that the practical helicopter was theoretically possible. It was, however, underpowered, unresponsive, mechanically complex and susceptible to reliability problems. Pilot workload was too high during hover to attempt lateral motion.

1.2.4 Convertawings Model "A" Quadrotor

Convertawings has revived the concept tried out on the previous works. A first prototype was completed in 1955 and it has since flown successfully.

Rotorcraft Structure:

The four rotors of this helicopter are mounted laterally on outriggers in two tandem pairs. The control mechanism is extremely simplified and obtained by differential change of thrust between the rotors. There is no cyclic control, but only collective control. In the experimental prototype the fuselage was of tubular steel, the booms supporting the rotors in aluminium alloy. Power is provided by two engines connected to the rotor drive system by multiple vee belts. Shafting and transmission cases ensure interconnection between the four rotors, so that at need either engine can drive all of them. There is a tricycle undercarriage with two wheels in the rear section and a nose wheel which can swivel.

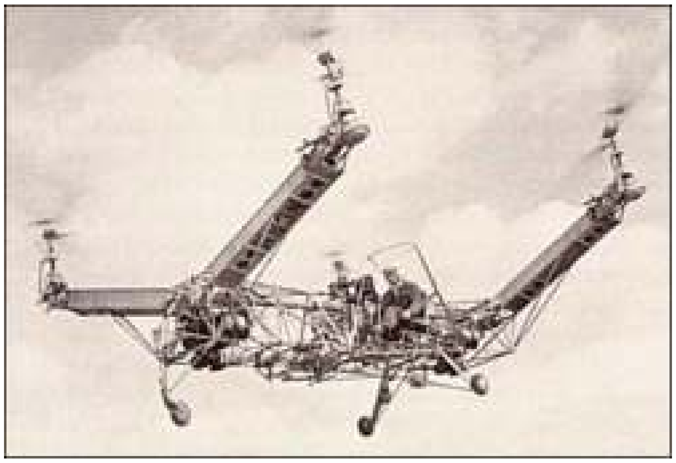


Figure 1-5: Convertawings Model "A" Quadrotor

Tests and Evaluation:

The first flights took place in March 1956. Despite successful testing and development, military support for the Quadrotor ceased after cutbacks in defense spending. However, the design - particularly its control system - was a precursor of current experimental vertical-rising aircraft designs that incorporate tandem wings or a square configuration of four fans, ducts, or jets.

1.3 LITERATURE REVIEW

1.3.1 Stanford University Quadrotor **[2]**

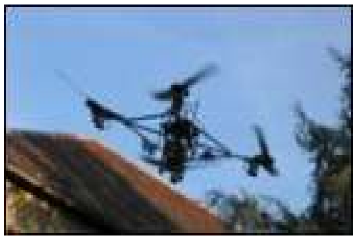
Goals:

The STARMAC project at Stanford University focused on multi-agent control of the quadrotors.

Platforms:

STARMAC-I was a DraganFlyer IV from RC Toys. This platform yielded 1kg of thrust and could maintain hover for up to 10 minutes at full throttle. It used a 3-axis gyro for attitude, sonar for altitude and a GPS receiver for position information.

Then the STARMAC-II was developed based on the X-4 platform to obtain 4kg of thrust and a much longer ight time. STARMAC-II continued to use GPS units for location information outdoors and sonar for altitude. In flying indoors an overhead web cam was used for positioning. Collaboration between quad-rotors and ground-station control was initially done with bluetooth, but later switched to WiFi.


(a)

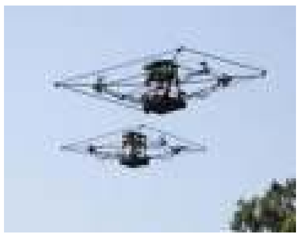

(b)

Figure 1-6: (a) STARMAC-I , (b) STARMAC-II

Control Theory:

Attitude control was achieved using integral LQR techniques. Position control was achieved using integral LQR techniques with information from GPS sensors or an overhead web cam. It was mentioned that stable altitude control was unachievable with LQR because of the down wash effect of the four rotors. To overcome this ultrasonic sensor was used with outlier rejection, Kalman filtering, and an integral sliding mode control and model-based reinforcement learning.

Achievements:

The initial goal was to maintain hover for 2 minutes within a 3m circle (the large area due to the inaccuracies of GPS) and this result was obtained with the STARMAC system. The STARMAC quadrotors have been shown to be very stable and well-controlled. Flight is limited however, to outdoor environments or special in-door rooms with external cameras and processing systems. No on-board processing of vision sensor information is performed for controlling the quad-rotor.

1.3.2 Brigham Young University Quadrotor [2]

Goals:

The initial goal of the Helio-copter project was to stabilize the attitude and altitude in the quad-rotor.

Hover within a limited area for at least 30 seconds without human intervention.

The final goal is to obtain basic target tracking by shifting the basis of position stabilization to a small target placed on the floor.

Platform:

Based on draganflyer quadrotor package, the BYU Robotic Vision Lab designed the Helio-copter quadrotor vision platform. It includes 3-axis rate gyros and accelerometers plus 3 temperature sensors for measurement calibration, and barometric sensors for altitude and velocity measurements.

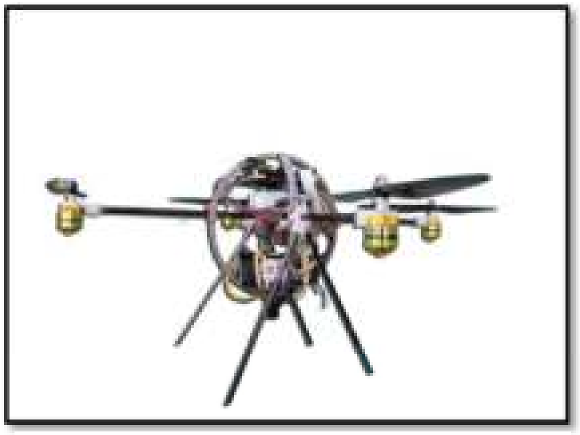


Figure 1-7: Helio-copter quadrotor

During normal operation it consumes .77W of power. It has four serial ports for communication with other systems and connections for four pulse-width-modulated servo connections.

Control Theory:

Using PID control structures the quad-rotor has been stabilized in two of its six degrees of freedom using a commercially available inertial measurement unit. An on-board FPGA system has been developed to obtain and process image information to stabilize the other four degrees of freedom required for stable, controllable flight. This vision system has been coupled with the IMU system through a communication scheme that allows sensor data to be exchanged between embedded systems. The vision sensor information and IMU information has been filtered and improved through an artificial neural network, outlier rejection, and Kalman filtering.

Achievements:

Yaw control was found to be extremely stable and maintain the quad-rotor at the same heading for an entire flight. The Helio-copter found to be able to maintain a constant desired altitude for an entire flight.

Translational drift proved more difficult. Through in-rig testing it was discovered that the translational correction system would require an integrator-only style control loop. Proportional gains on the translation controller caused oscillations even at very low gain values. However, the PID structures in the autopilot were not designed to handle an integrator-only approach and without changing the structure, good translational correction was unobtainable.

1.3.3 Aalborg University Quadrotor [4]

Goals:

The goal of this project was to attain autonomous fight with the X-Pro in hover state.

Platform:

The X-Pro dragonfly quadrotor platform was modeled in order to create a model-based controller.

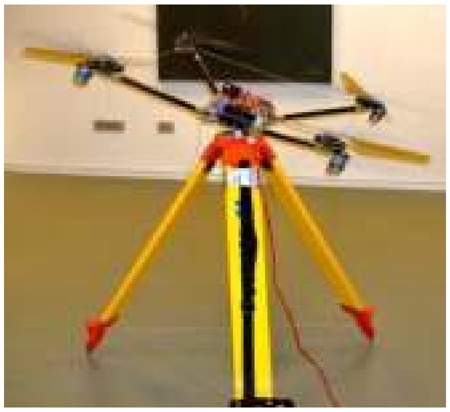


Figure 1-8: X-Pro on a Stand

The model is a non-linear model, and it’s linearized in order to use a linear controller. This was done using the hover position and hover attitudes as operating point. The linearized model was compared to the non-linearized model, and the linearized model provided a satisfactory response.

Control Theory:

Two controllers were designed to control the X-Pro. First, an ordinary PID controller and then a model based LQR controller.

The PID controller was made using trial and error to fine tune the parameters necessary to make the X-Pro hover. The PID controller has an inner attitude control loop and an outer position control loop.

The LQR controller was designed based on the state space representation of the X-Pro, containing an observer and a controller.

Achievements:

The PID controller was capable of keeping the X-Pro in hover, while the heading and the altitude are within the boundaries specified in the requirement specification, both the x and y position are not. The X-Pro floats outside of the desired area, while the controller attempts to bring it back, but the response was slow, resulting in steady movement around the reference point. Comparing the flight data obtained by manual flight, with flight data using the PID controller, both data sets representing attempts to stay in hover, the PID controller is better at keeping the X-Pro in hover.

The LQR controller does not work as well as the PID controller when attempting to keep the X-Pro in hover position. The positions are fluctuating a lot around the hover point in all 3 directions and it never settles. The LQR controller showed more fluctuations than the PID controller. Thus, they added an integral part to counter the obvious steady state errors. Therefore, to counter the fluctuations in position, the Q matrix needs to be weighted differently.

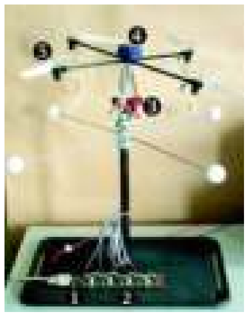
1.3.4 University of Aboubekr Belkaid Quadrotor [5]

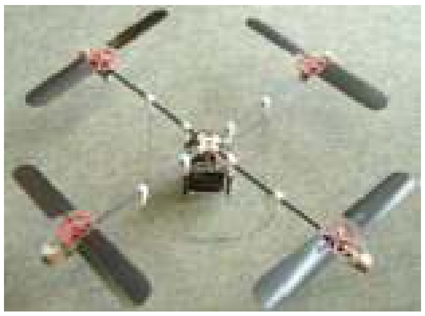
Goals:

The objective of the project is to model, design and control a micro miniature quadrotor.

Platform:

Two platforms were developed; the first one is a quadrotor like test-bench with off-board data processing and power supply. It was used to safely and easily test control strategies. The second one, OS4, is a highly integrated quadrotor with on-board data processing and power supply. It has all the necessary sensors for autonomous operation.





(b)

Figure 1-9: (a) Quadrotor like Test-Bench, and (b) OS4 Quadrotor

Control Theory & Results:

Five control techniques were used for the attitude control of a quadrotor.

The first technique is based on Lyapunov theory; it proved to be very reactive, especially for the yaw angle control. The stabilization in the direct neighborhood of the equilibrium point was not rigid enough to permit hover flight.

The second one is a PID controller; it proved to be well adapted to the quadrotor when flying near hover. It was possible using this technique to successfully perform the first autonomous flight. The PID controller was only able to control the quadrotor in near hover and absence of large disturbances.

The third one is an LQ controller, it displayed average stabilization results. It showed to be less dynamic than the PID.

The fourth control technique is the Backstepping; its ability to control the orientation angles in presence of relatively high perturbations is very interesting.

The sliding-mode technique is the fifth approach; it did not provide excellent results. The switching nature of the controller seems to be ill adapted to the dynamics of the quadrotor. The results of all these control approaches conducted to a combination of PID and Backstepping into the so-called Integral Backstepping. This was proposed as a single tool to design attitude, altitude and position controllers. The experiment had shown that OS4 is able to take-off, hover, land and avoid collisions automatically. As far as we know, OS4 is the first quadrotor practically capable of a collision avoidance maneuver.

1.3.5 Summary of the previous work

Table 1-1: Summary of the previous quadrotor projects

	Goals	Platform	Control Theory	Results	challenges
Stanford University	Multi-agent quadrotor control	STARMAC-I Draganflyer IV & STARMAC-II X-4 platform	Integral LQR	• Maintain hover for 2 minutes within a 3m circle. • Indoor Trajectory Tracking with accuracy of 10cm	Need overhead webcam for position control (to simulate the GPS readings)
BYU	• Attitude and altitude stabilization • Hover flight • Trajectory tracking	Helio-copter Draganflyer	PID	• Altitude and yaw stabilization • Drifting while hovering	Translational correction was unobtainable because of the Integral part
Aalborg University	Hover flight	The X-Pro Draganfly	• PID • LQR	Suffering from Fluctuation while hovering.	Suffering from Fluctuation while hovering.
University Aboubekr Belkaid	• Hover flight • Collusion avoidance	Quadrotor like test-bench & OS4	• Lyapunov • PID • LQ • Backstepping • Sliding-Mode	The OS4 is able to take-off, hover, land and avoid collisions automatically by using the Backstepping controller.	The controller designed based on a simplified model for hover flight and for small attitude perturbation.

1.4 Problem Statement

In this work, the AUS quadrotor project is established to be a new platform in the American University of Sharjah – Unmanned Autonomous Vehicle Multi-agent project (AUS-UAVM). This work includes designing, modeling and controlling the AUS quadrotor with novel nonlinear control algorithms (AIB-FL/LMS and LV algorithms) for hover flights.

To explain more details about the above statement, next chapters will clarify the proposed fields of the AUS quadrotor project.

1.5 Contribution

The contributions of this thesis are mainly points those addressed on the problem statement: Establishing, designing, modeling, and controlling of the AUS quadrotor. Listed below are the detailed contributions:

Establishing the AUS quadrotor project:

As a new platform in the AUS-UAVM project, this thesis is the bedrock of the quadrotor project in AUS.

Designing the platform:

The X-Pro Draganfly from rctoys is the platform that is used in this project. The structure of the X-Pro quadrotor has been modified to meet the project requirements. The contributions are in following modifications:

1. Two level boards are built to carry the microcontroller, the electrical power drive circuit, and the sensor devices.
2. Design electrical boards for the microcontroller and the electrical drive circuit.
3. Design a ground system to have the full functionality on the quadrotor.
4. For Safety:
    a. Assemble the training set on the quadrotor.
    b. Design a Manual Override System and a Failure Detector System to prevent the system failure.

Nonlinear dynamics model of the quadrotor:

The contribution in this field is to obtain the quadrotor parameters of a nonlinear dynamic model that represents the actual dynamic of the quadrotor.

Control Algorithms:

The most important contributions are in control. The contributions scope covers the field of rotorcraft control, and in particular the AUS quadrotor control. This is done by introducing the following novel control algorithms:

1. Enhance the backstepping control technique by including:
    a. The Integral action to resist the disturbances.
    b. The Adaptation scheme to estimate the modeling errors and variations.
2. Use the AIBC in attitude and altitude stabilization control.
3. Improve the performance of the system by merging the AIBC with:
    a. The Fuzzy Logic control technique.
    b. The Least Mean Square algorithm.
4. Control the velocities using Lyapunov-based Controller (LVC).
5. Compare the proposed algorithms with the PID and LQR controllers.
6. Propose the Tangent Heading Algorithm (THA) for trajectory tracking.
7. Design and implement an autopilots system, based on the proposed algorithms, which is capable of commanding the quadrotor and make it autonomous.
8. Test and validate the developed algorithms by :
    a. *Software simulation:* by showing extensive simulations using Matlab/SImulink Environment.
    b. *Real time simulation:* by implementing the Hardware in the loop Simulation, HILS.
    c. *Real time Implementation:* by performing test to validate our simulation results.

1.6 THESIS OUTLINE

After introducing the quadrotor in this chapter, this thesis will going to represent five chapters more about the AUS quadrotor. A brief on these chapters are represented in following:

Chapter 2: Quadrotor Platform

This chapter concerns the development of the quadrotor platform. The quadrotor structure, the high-level architecture, and finally the safety precautions will be represented in this chapter.

Chapter 3: Quadrotor Model

This chapter derives the mathematical non-linear model of the X-Pro quadrotor. Furthermore, two ways of deriving the quadrotor model will be represented, and a validation of the chosen model will be simulated.

Chapter 4: Quadrotor Control

Novel control algorithms will be represented in this chapter. As well as, a comparison between the proposed algorithm with the PID and the LQR controllers will be expressed and evaluated.

Chapter 5: Test and Validation

Mainly, this chapter consists of three sections: the software simulation, the Hardware-In-The-Loop-Simulation (HILS), and the real implementation test. An intensive simulation and test results will be depicted on this chapter to validate the proposed control algorithms in chapter 4.

Chapter 6: Closure

This chapter concludes upon the work carried out in this master's thesis.

Appendices

This part contains Appendices A trough D, each of which individually elaborates on various subjects throughout the thesis.

CHAPTER 2

QUADROTOR PLATFORM

2.1 INTRODUCTION

An overview about the X-Pro quadrotor platform architecture will be represented in this chapter. The Draganfly X-Pro is a commercial quadrotor that is:

- ➢ Capable of carrying payloads of 0.5 kg.
- ➢ Features brushless dc motors and belt drives for all four rotors.
- ➢ Offers high maneuverability at a reasonable cost that can be affordable to be used as a research platform to minimize development cost.

The structure of the X-Pro quadrotor has been modified to meet the project requirements. The four arms with actuator and the bottom carbon fiber sheet are reused. In addition, two level boards are built to carry the power drive circuit and the sensors.

This chapter will describe the quadrotor parts and their connection chart. First, a specific description of the hardware platform will be represented. Later, the high-level architecture will be depicted in a connection chart, followed by a description of the Ground Control Station (GCS) software. Finally, the quadrotor is a harmful vehicle, where it needs a special care while test operations, therefore a safety kit with test stands are constructed for safety issues.

2.2 DESCRIPTION OF THE HARDWARE PLATFORM

The main parts of the X-Pro quadrotor platform are shown in Figure 2-1. Next, the description of each part will be represented.

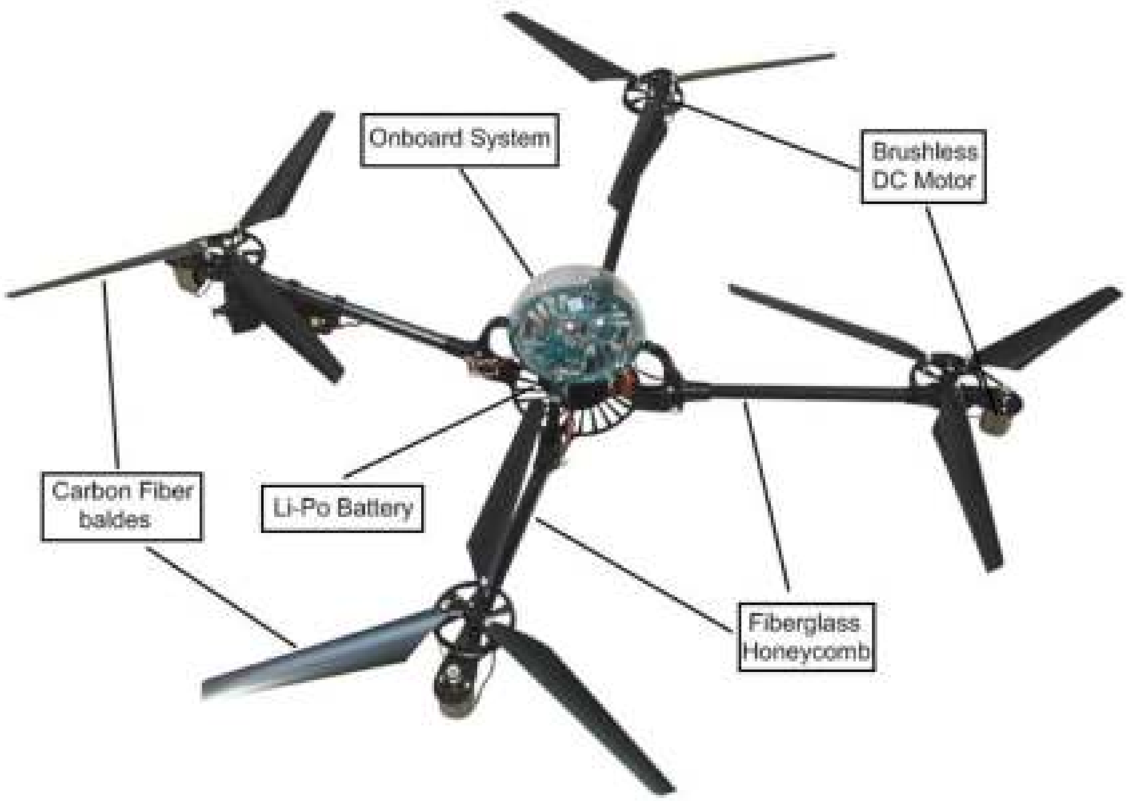


Figure 2-1: X-Pro quadrotor Platform Structure [6]

2.2.1 Quadrotor Frame Structure

The quadrotor frame is consisting of three parts:

Cabin:

Contains the onboard system (microcontroller, electrical driving circuit, sensors, wireless antenna … etc). It's designed to hold the microcontroller with the electrical driving circuits in the lower level board. While the second level board is carrying the wireless and sensors devices. Figure 2-2 depicts the design architecture of the cabin.

Base Plate:

Formed from carbon fiber material. It's strong enough to fix the four arm tubes and to carry the heavy weighted batteries. The base plate is shown in Figure 2-3.

Arm Tubes:

Four arms tubes are used to connect the actuators with the main body of the quadrotor. It's formed from the fiberglass honeycomb material. The honeycomb tubes are laminated with the fiberglass on the inside and outside, the lamination produces an incredibly light and strong body tube [7]. A fitting is designed on each end of the arm tube to fix the actuator from one side and the rizer from the other side. Where the rizer is the connection part between the main body of the quadrotor and the arm tubes. The arm tubes and the rizers are shown in Figure 2-4 and Figure 2-5 respectively.

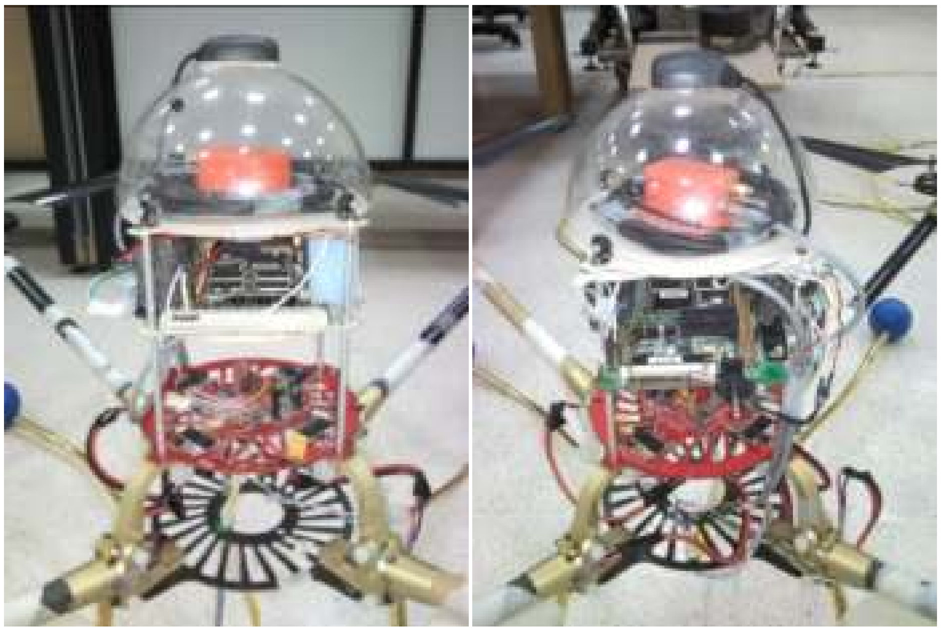


Figure 2-2: Cabin

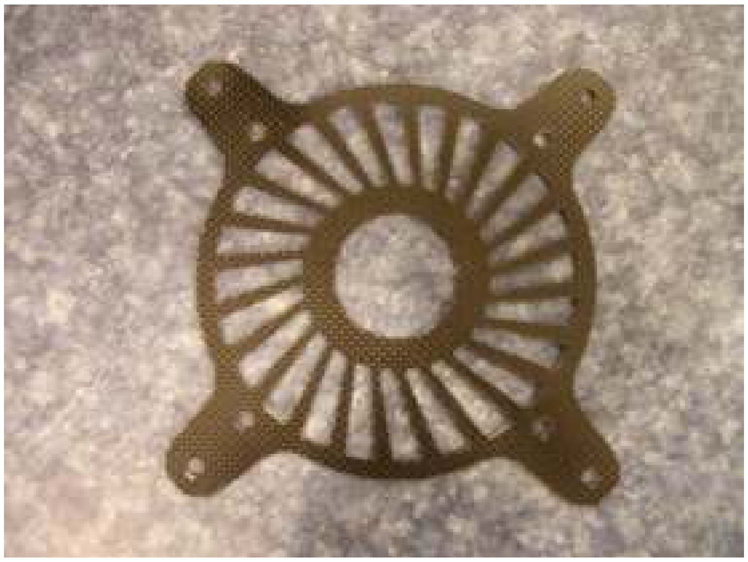


Figure 2-3: Base Plate.

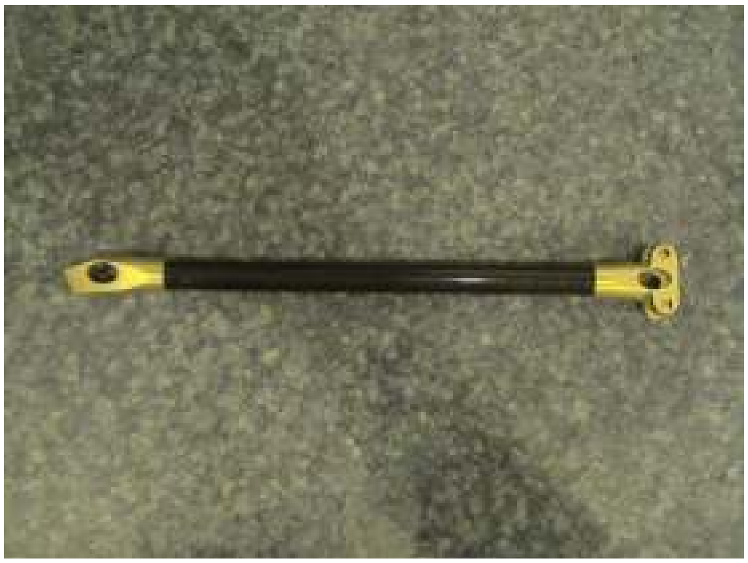


Figure 2-4: Arm Tube with fittings.

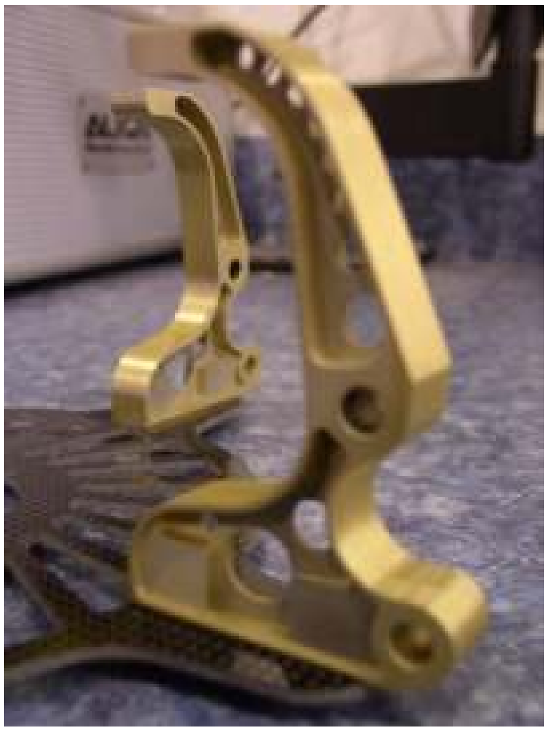
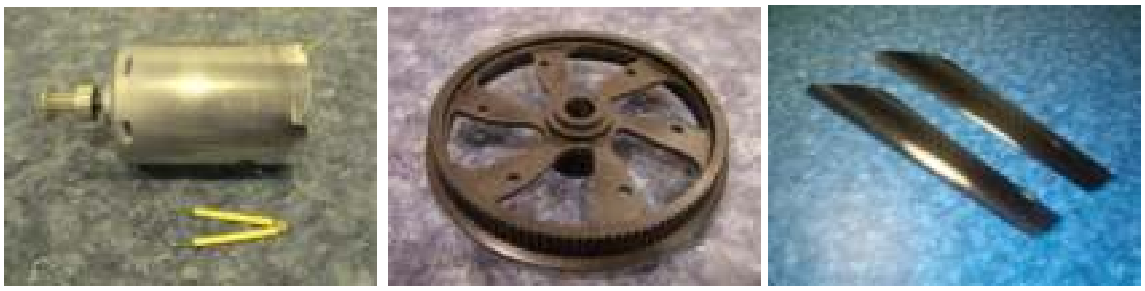


Figure 2-5: Rizers fixed on the base plate.

2.2.2 Actuators

Each actuator in the quadrotor consists of three parts: motor, gear and rotor (propeller). Figure 2-6 shows the actuator components.


(a) (b) (c)
Figure 2-6: Actuator Parts: (a) Brushless Motor, (b) Gear, and (c) propellers (rotor blades)

The X-Pro quadrotor is using Mabuchi RS-545SH-5018 motors. This motor is a high RPM brushed DC motor. The motor specifications can be found in Appendix B.1.

The power is provided to the motors by an electrical driving circuit, where each motor draws a maximum current of 8A, i.e. 32A as a total. The motors are attached to the gears using belts. The gears are used to reduce the rotational speed and to increase the output torque of the motors. To produce the downward thrust, the propellers are mounted on the gears as shown in Figure 2-7. This figure illustrates the way that the three parts are assembled to form the actuator.

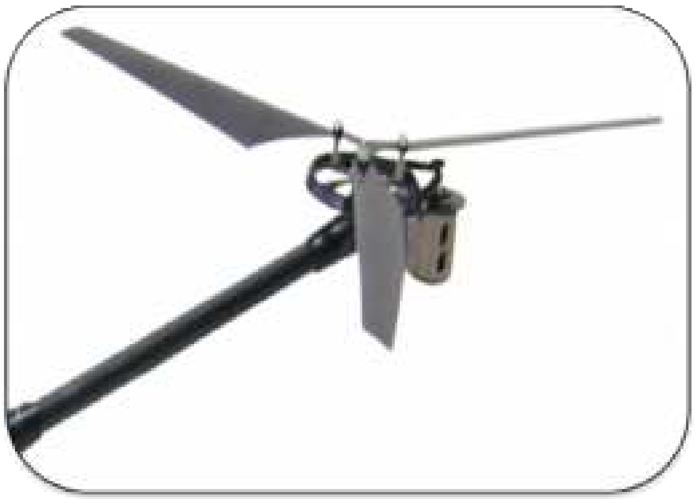


Figure 2-7: X-pro Quadrotor Actuator

2.2.3 Power Sources

The quadrotor systems consist of the ground and the onboard systems. The way that the power is supplied to each system is different. This is due to the requirements and the availability of the power source type. In the ground system, the power is provided by a power supply. On the other hand, the batteries are used to provide the onboard system and the actuators with the necessary power.

Power Supply:

The EX354D power supply is used for the ground system; it has two independent and isolated outputs each with a 0 to 35V, 0 to 4A capability. The outputs can be wired in either series or parallel to provide voltages up to 70 volts or currents up to 8 amps. [8] And this is more than enough for the ground system power requirements (MPC565 and ATMEL board’s microcontrollers). Where, it requires a 9V and 1.5A as a total.

Figure 2-8: Ground System Power Supply

Batteries:

The onboard system acquires the power from two batteries. The first battery supplies the power to the electronic devices (such as microcontrollers, sensors, wireless communication device …) and the other battery is used for the actuators.

For electronic devices, the battery voltage is regulated to provide us with different voltage sources according to the voltage requirements of each electronic device. The battery used for the electronic devices is 4-cell Li-Po battery from Thunderpower (14.8V & 4400mAh). For actuators, a 5-cell Li-Po battery (18.5V & 5300mAh) is used as well. [9]

Figure 2-9 shows the batteries that are used in the onboard system. Furthermore, the datasheets and the procedure of charging the batteries are provided in Appendix B.2.

Figure 2-9: Li-Po Batteries that are used in the Onboard System

2.2.4 *RC Transmitter*

The radio control transmitter sends a radio signal over a frequency to the receiver in the X-pro quadrotor. The transmitter has a 9V Ni-Cd battery, which provides the power for the controls and transmission of the signal.

The RC transmitter used in this project is 9-channel Futuba transmitter. The transmitter provides us with two kinds of communication modes, the Pulse Code Modulation (PCM) and the Pulse Position Modulation (PPM) mode. The PCM mode encodes a radio signal for communication which is wirelessly transmitted to the receiver. The PPM mode generates digital signal with specific sequence. The PPM signal can be decoded to indicate the Input values of the channels.

In this project, the radio receiver is removed and the PPM signal is decoded to switch between the different flight modes. The switching system is called by the Manual Override System (MOS). The MOS will be explored more in detail in section 2.3.3.3.

Figure 2-10: Futuba RC Transmitter

2.3 High-level Architecture

From the original X-Pro quadrotor, the mechanical parts are reused, which are the four arms, the actuators, and the base plate. The top boards of the body are all new design which is contains all the electronics needed to build the AUS autonomous quadrotor vehicle.

In this section, the high-level architecture diagram will be represented in a connection chart. Mainly, the architecture depicts the connections between the ground and the onboard systems. Later each part in the connection chart will be described in detail.

2.3.1 Connection Chart

The quadrotor features three microcontroller units designed to share the processing load of the system. Two microcontroller units in the ground level are responsible for data monitoring and collection from the sensors. The collected data is shared with the onboard microcontroller, which is responsible for vehicle stabilization and navigation, as well as transmitting data wirelessly to the ground system. Figure 2-11 depicts the connection chart of the AUS quadrotor project

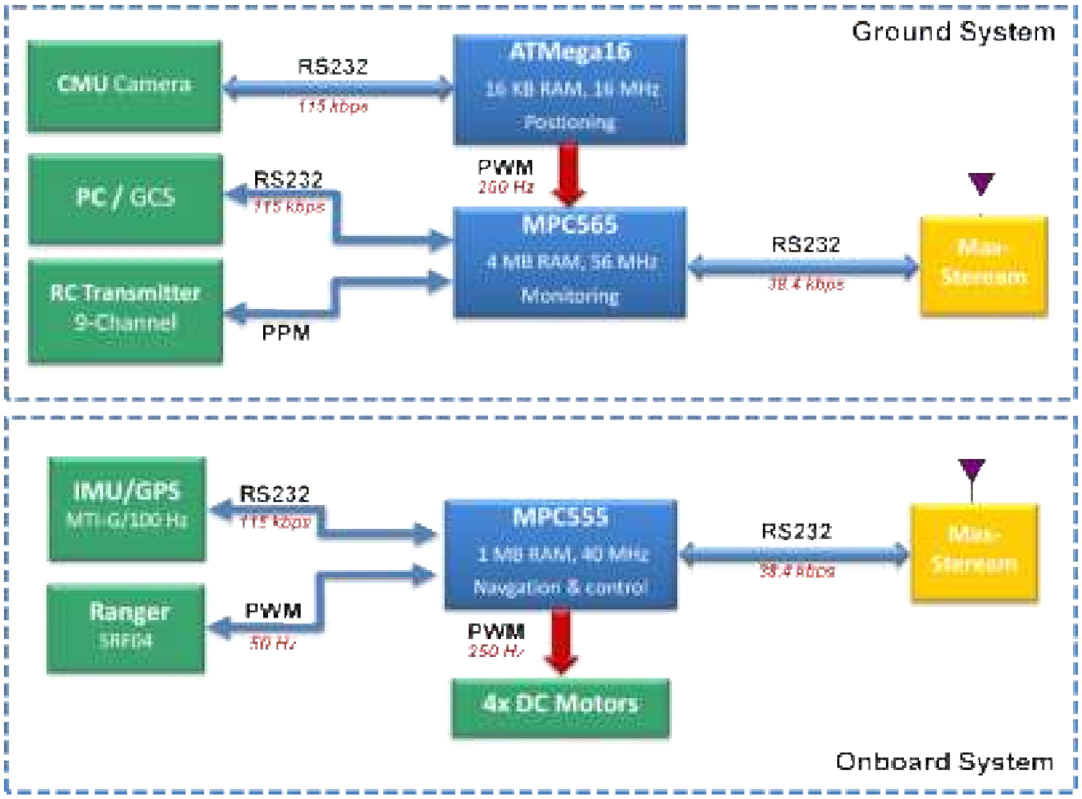


Figure 2-11: High-Level Architecture Diagram

2.3.2 Onboard System

The onboard system is responsible of controlling the quadrotor autonomously. As shown in Figure 2-11, The MPC555 microcontroller is the brain of the onboard system. The MPC555 Microcontroller collects data from the sensor devices (IMU/GPS, Ultrasonic sensors and wireless device) to command the thrust of the four actuators.

The MPC555 is programmed with an algorithm process to perform the stabilization and the navigation of the AUS quadrotor. In this project two control algorithms are represented, the classical PID and the AIBC algorithms. The two controllers are explored more in detail in chapter-4 and chapter-5.

Next, general overview about the onboard system will be represented by giving a brief explanation on each part, and showing the interaction between the parts.

2.3.2.1 MPC555 Microcontroller

The MPC555 Motorola Microcontroller is a high-speed 32-bit Central Processing Unit that contains a floating point unit designed to accelerate the advanced algorithms necessary to support complex applications. [10]

High-performance data manipulation capabilities and large on-chip Flash memory with powerful peripheral subsystems makes from the MPC555 a powerful microcontroller that is capable to process the control algorithms while interacting with many sensor devices instantly.

The MPC555 contains three Time Processing Units (TPUs). Each TPU is considered as a sub-microcontroller that is used to share the processing loads with the main CPU. Reading and sending PWM signals with different devices are performed using the TPUs. This is will distribute the interrupt processing load and it will give the CPU less load to process the control algorithm.

The MPC555 used in this project is a phyCORE-MPC555 from phytec. The clock frequency of the MPC555 is 40MHz. It has 1 MB Flash memory. The Flash serves as Boot and code memory and supports In-System programming. An additional 26 KB on-chip SRAM and 448 KB on-chip Flash memory is available on the MPC555. The 272-pin BGA controller also boasts a 64-bit Floating Point Unit, SPI, dual on-chip Full 2.0B

CAN, dual-channel UART, and dual 16-channel A/D converters with 10-bit resolution. [11]

Figure 2-12 shows the phyCORE-MPC555 board that is used in the onboard system.

Figure 2-12: phyCORE-MPC555 board

The peripherals used in this project are:

- UART-1 to collect the IMU/GPS data from the MTi-G unit.
- UART-2 to communicate with the ground system through the wireless device.
- Four pulse width generation for the actuators.
- Input capture (pulse measurement) and pulse width generation with two SRF04 ultrasonic sensors.

The controlling process requires sensoring information from the navigation sensors. The information provided by the navigation sensors will be represented by describing them individually.

One of the advantages of using the MPC555 microcontroller is the Matlab/Simulink provide us with the Target Support Package FM5. The package let the user to deploy a production code generated from Real-Time Workshop Embedded Coder directly onto the MPC555 microcontroller. The code could be executed in real time for on-target rapid prototyping, production deployment of embedded applications, or validation and performance analysis.

The I/O device Simulink driver library in Target Support Package FM5 includes driver blocks for digital I/O, pulse width modulation (PWM), waveform measurement, CAN, serial, analog output, and TPU functions such as quadrature decode or programmable time accumulator. The MPC555 device driver blocks are combined with control system algorithms for real-time execution. [12]

In contrast, the package has many disadvantages compared with the direct programming using any programming languages, like C/C++. The limitation of the sampling time provided from the package, the synchronization problems with other devices, and the lacking support of some peripherals (I2C and SPI), raise many programming issues especially for complicated system such as our system.

Inspite of that, the synchronization problems had been solved, and the package is used to program the stabilization and navigation control algorithms.

The control algorithms are represented and described thoroughly in the following chapter-4.

2.3.2.2 MTi-G IMU/GPS ***[13]***

The MTi-G is an integrated GPS and Inertial Measurement Unit (IMU) with a Navigation and Attitude and Heading Reference System (AHRS) processor from xsens. The MTi-G is a small size and low weight 6 DOF measurement unit that is used for control, navigation and analysis of the quadrotor.

The MTi-G is the heart of the AUS quadrotor vehicle. It has many features that make from it a suitable device for controlling purposes. Some of the important features are listed below:

- real-time computation of inertial enhanced position/velocity and GPS enhanced attitude/heading on embedded DSP
- built-in 16 channel Global Position System (GPS) receiver
- -158 dBm tracking sensitivity
- accurate full 360 degrees 3D orientation output (Attitude and Heading)
- 3D acceleration, 3D rate of turn and 3D earth-magnetic field data
- static pressure sensor (barometer)
- high update rate (100 Hz on embedded DSP, 512 Hz inertial data only)
- compact design and low weight device

- ultra-low power consumption
- individually calibrated for temperature, 3D misalignment and sensor cross-sensitivity
- antenna fault detection
- external active antenna status detection circuit

Figure 2-13 shows the MTi-G sensor device used in this project.

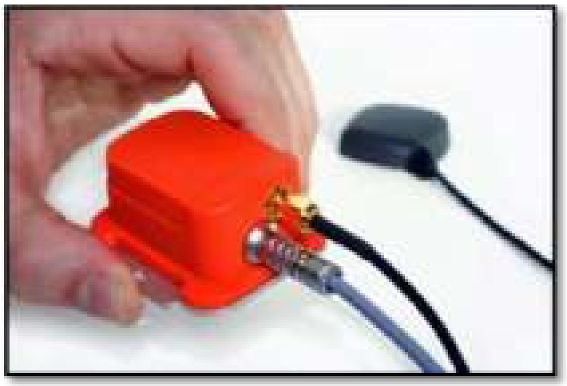


Figure 2-13: MTi-G sensor

The components and the hardware architecture of the MTi-G are depicted on the following Figure 2-14:

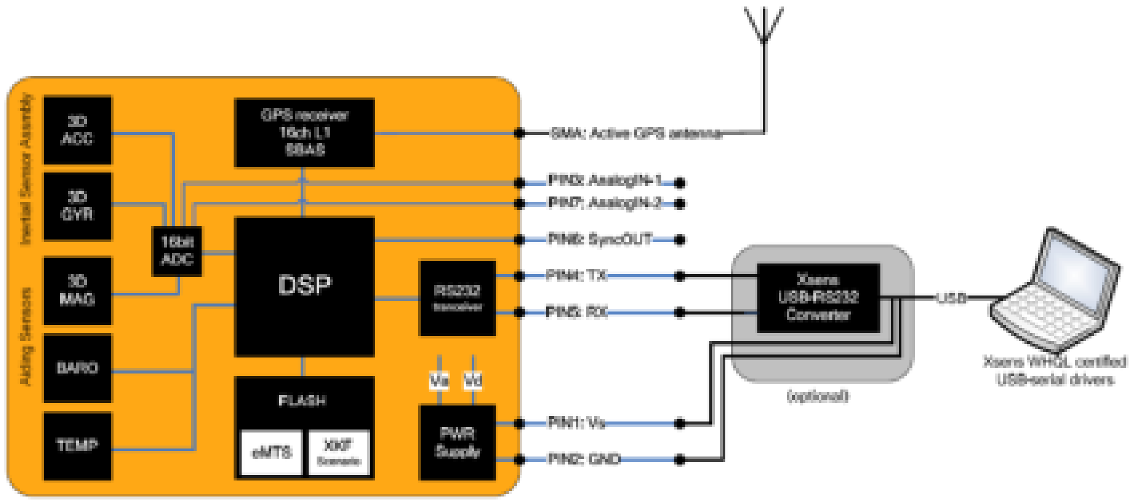
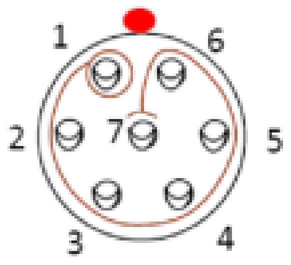


Figure 2-14: MTi-G Hardware Architecture

The orientation and position of the MTi-G is estimated using an extended Kalman filter called by Xsens Kalman Filter 6DOF GPS (XKF-6G). A Kalman filter is divided into two steps, a prediction step and a correction step. In the prediction step, the inertial sensors are integrated over time to obtain a position and orientation estimates. In the correction step, the drift of the measurement of inertial sensors integration is corrected using the GPS receiver data and the static pressure sensor signal (barometer).

The MPC555 connected to the MTi-G unit through the RS232 serial connector. The USB connector provided with the kit is replaced with the RS232 serial connecter. Figure 2-15 gives an idea about the pins layout of the MTi-G output socket/plug:



Signal	ODU pin
VCC	Pin 1
GND	Pin 2
Analog IN1	Pin 3
TX (sensor)	Pin 4
RX (sensor)	Pin 5
SyncOut	Pin 6
Analog IN2	Pin 7

Figure 2-15: MTi-G pins socket layout

The communication with the MTi-G is done by messages which are built according to a specific communication protocol. The communication protocol contains 0 to 254 bytes of data and the total length is 5 to 259 bytes.

The protocol contains the following fields:

PRE	BID	MID	LEN	DATA	CS

Table 2-1 describes the fields of the communication protocol:

Table 2-1: MTi-G Communication Protocol Structure

Field	Field width	Description
PRE	1 byte	Preamble, indicator of start of packet → 250 (0xFA)
BID	1 byte	Bus identifier / address → 255 (0xFF)
MID	1 byte	Message identifier
LEN	1 byte	Value equals number of bytes in DATA field Maximum value is 254 (0xFE). Value 255 (0xFF) is reserved.
DATA	0 – 254 bytes	Data bytes (optional)
CS	1 byte	Checksum of message

The Message Identifiers used in this project are represented in Appendix C.1.

2.3.2.3 SRF04 Ultrasonic Sensors

This sonar ranger from Devantech used to measure the altitude of the quadrotor in the range of 3cm to 3m. The SRF04 is designed to measure the distance by acquiring a short trigger pulse and providing an echo pulse. Which in turn, the MPC555 microcontroller sends a trigger pulse and waits the echo to measure the range. The connections and schematic diagrams of the SRF04 sensor can be found in Appendix B.3. Figure 2-16 depicts the SRF04 ultrasonic ranger that is used in this project.

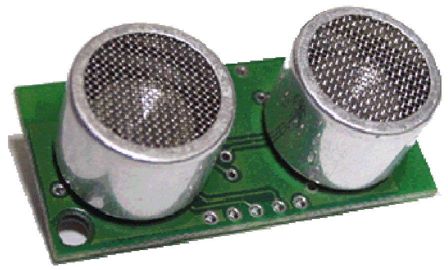


Figure 2-16: SRF04 Ultrasonic Sensor

The SRF04 sends out 8 cycles burst of ultrasound at 40 KHz and makes the echo line high. As soon as it detects the echo signal it lowers the echo line again. The echo line is a pulse whose width is proportional to the distance of the object. By timing the pulse it's possible to calculate the range. If nothing is detected then the SRF04 will lower its echo line anyway after about 36ms. [14]

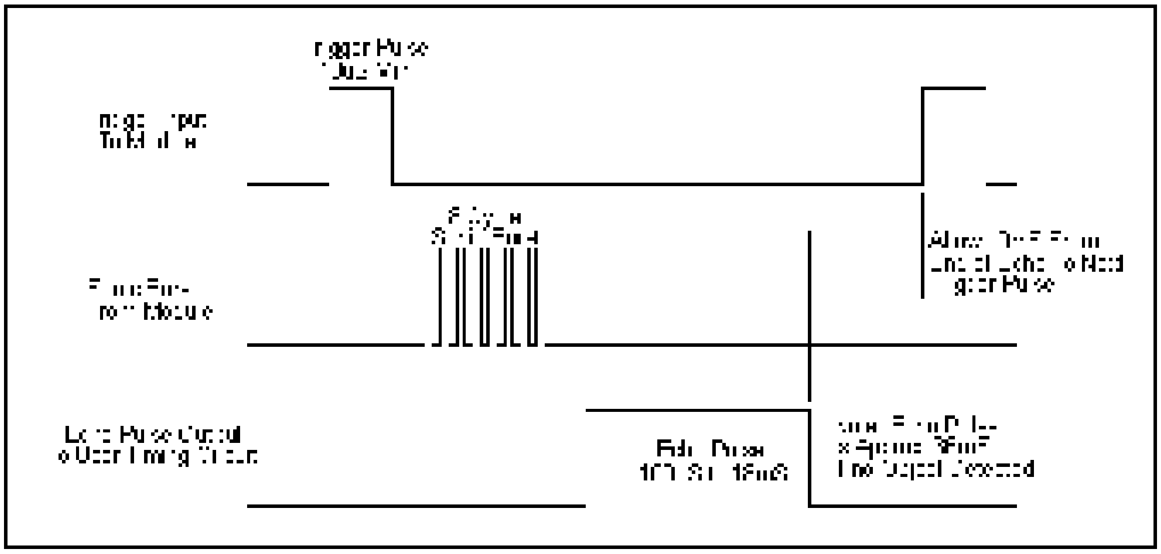


Figure 2-17: SRF04 Timing Diagram

If the width of the pulse is measured in μs, then dividing by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches.

$$Distance(cm) = \frac{\mu s\ width}{58}$$

$$Distance(inches) = \frac{\mu s\ width}{148}$$

In the AUS quadrotor, two SRF04 are mounted on the y-axis as shown in Figure 2- 18. This configuration gives accurate readings of the altitude by finding the mean of the two heights.

In addition to the IMU pitch readings, it is used to estimate the pitch angle when the ground is leveled.

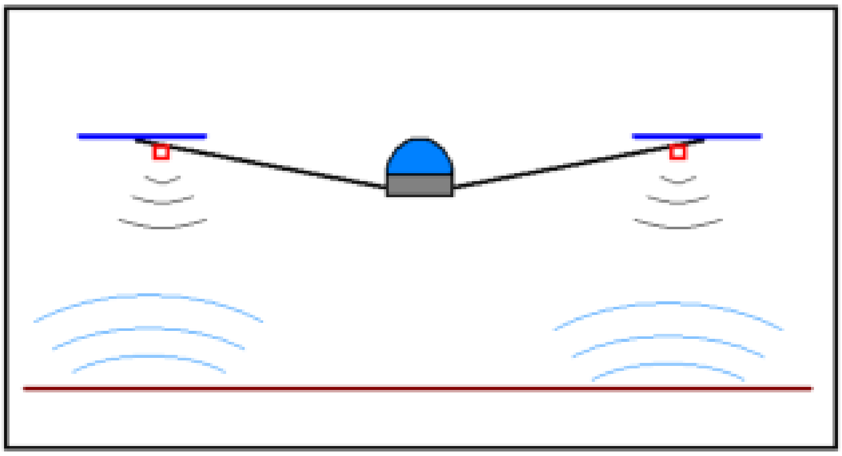


Figure 2-18: Location of SRF04 Ultrasonic Sensors on the AUS Quadrotor

The average distance is found according to this equation:

$$Average\ Distance = \frac{D_f + D_r}{2}$$

On the other hand, the pitch angle can be estimated according to the following equation:

$$Pitch\ Angle = \operatorname{asin}\left(\frac{D_f - D_r}{96}\right)$$

Where; D_f is the distance in cm of the front ultrasonic sensor, and D_r is the distance in cm of the rear ultrasonic sensor.

Figure 2-16 illustrate how the pitch angle equation is extracted.

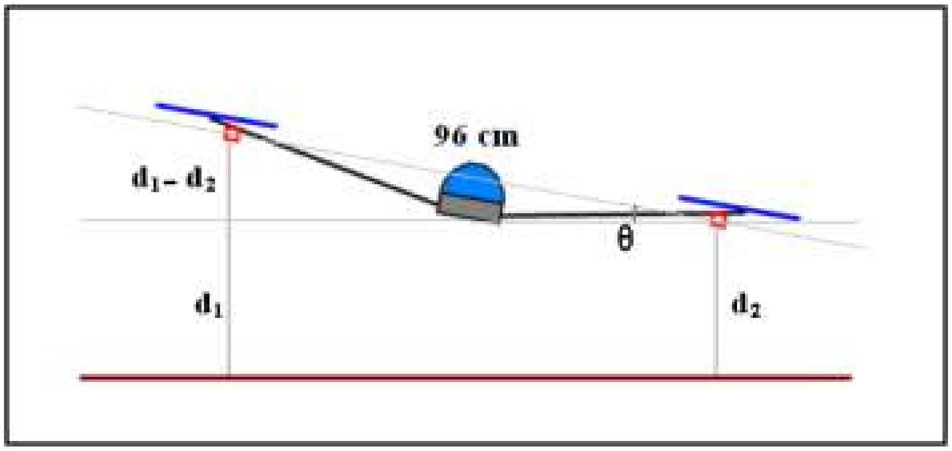


Figure 2-19: Pitch angle estimation from Ultrasonic Sensors Measurements

2.3.3 Ground System

The ground system is a crucial part in this project. It gives the functionality to monitor and command the AUS quadrotor during the flight operations. Mainly, the ground system is responsible of:

- Providing the AUS quadrotor with the navigation data (position) from a fixed camera on the ground.
- Changing the flight modes using the MOS.
- Controlling the AUS quadrotor manually or semi-automatically.
- Setting the controller parameters during the flight.
- Monitoring the AUS quadrotor data (e.g. navigation data).

Mainly, the MPC565 is a central part of the ground system. It is considered to be the bridge between the different parts on the ground system as Figure 2-11 shows.

In addition to the MPC565, another microcontroller from ATMEL (ATmega16) is used to gather the position data from the CMUcam3. The ATmega16 passes the position data to the MPC565 through a PWM line as depicted in Figure 2-11.

Next, general overview about the ground system will be represented by giving a brief explanation of each part, and showing the interaction between the parts.

2.3.3.1 MPC565 Microcontroller

The MPC565 is manufactured by phytec, which is the same company of MPC555. Figure 2-20 depicts the MPC565 board on the development board.

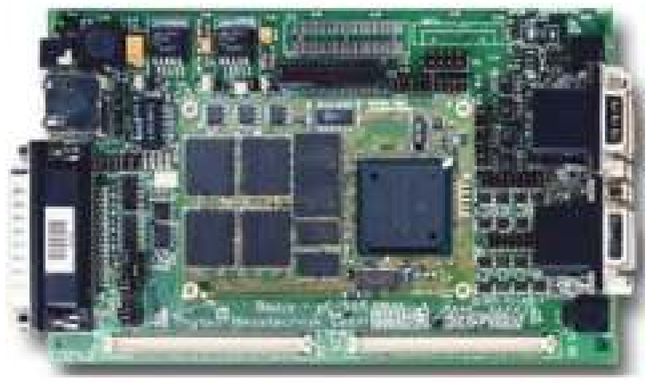


Figure 2-20: phyCORE-MPC565 Development board

The MPC565 is an enhanced version of the MPC555. Most functional features of the MPC555 are unchanged on the MPC565. Inspite of the MPC565 has higher capabilities and faster speed than the MPC555, but the performances of the two microcontrollers are close. The difference could be noticeable when the processing load is heavy and the algorithm is very complicated, and this is not the case in this project.

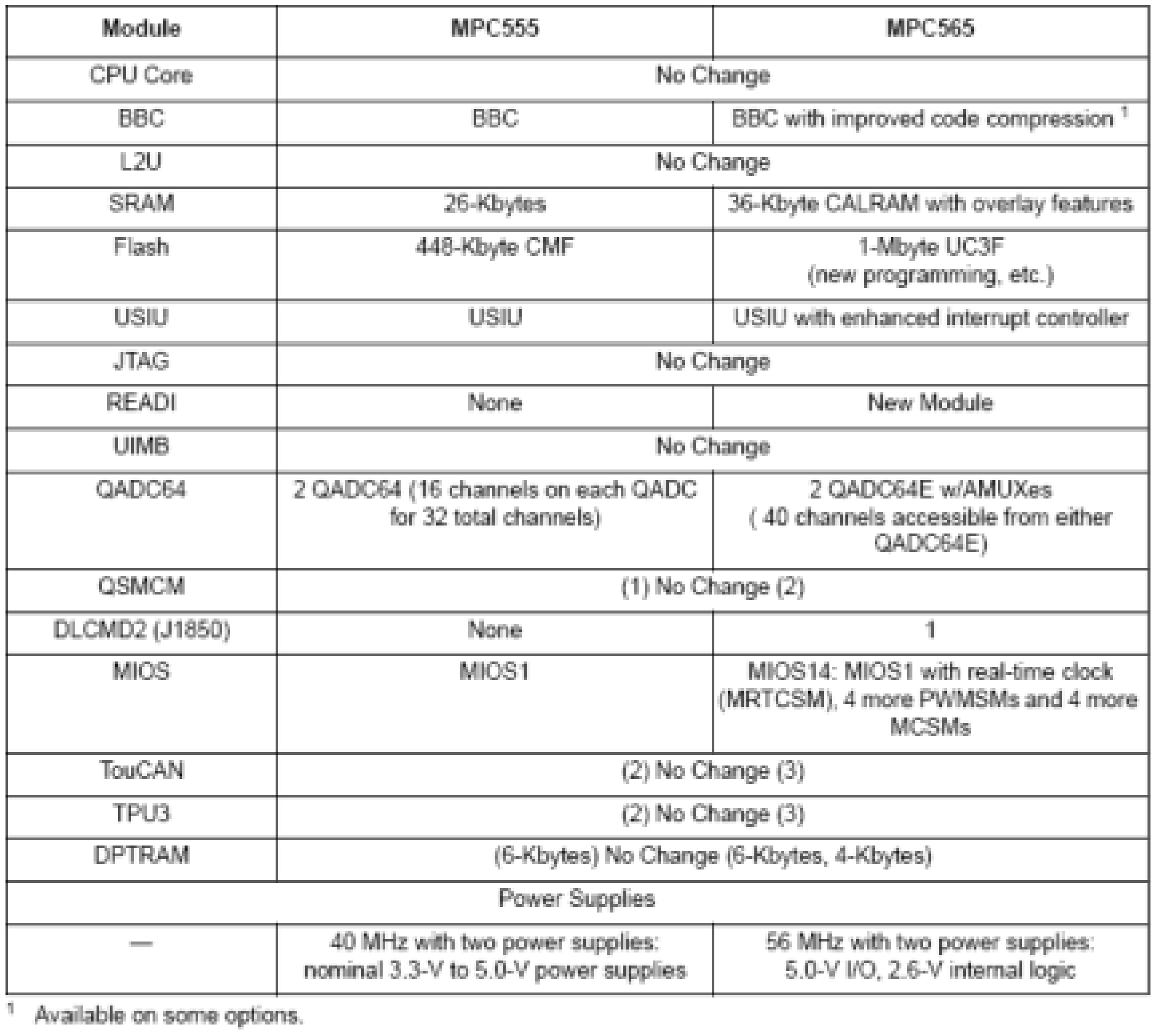
The reason of using MPC555 instead of using MPC565 in the onboard system is that the AUS-UAV team designed a replacement of the MPC555 development board. Developing a replacement (PCB) comes for the following reasons:

- To get rid of the heavy weight of the development board
- To be commonly used in all UAV platforms

Because the weight and the space are not an issue in the ground system, and the development board provides us with both digital and analog peripherals, we take the advantage of using the MPC565 development board in the ground system.

A comparison between MPC555 & MPC565 modules are illustrated in Table 2-2.

Table 2-2: Differences Between Modules of the MPC555 and the MPC565 [11]



Module	MPC555	MPC565
CPU Core	No Change	
BBC	BBC	BBC with improved code compression [1]
L2U	No Change	
SRAM	26-Kbytes	36-Kbyte CALRAM with overlay features
Flash	448-Kbyte CMF	1-Mbyte UC3F (new programming, etc.)
USIU	USIU	USIU with enhanced interrupt controller
JTAG	No Change	
READI	None	New Module
UIMB	No Change	
QADC64	2 QADC64 (16 channels on each QADC for 32 total channels)	2 QADC64E w/AMUXes (40 channels accessible from either QADC64E)
QSMCM	(1) No Change (2)	
DLCMD2 (J1850)	None	1
MIOS	MIOS1	MIOS14: MIOS1 with real-time clock (MRTCSM), 4 more PWMSMs and 4 more MCSMs
TouCAN	(2) No Change (3)	
TPU3	(2) No Change (3)	
DPTRAM	(6-Kbytes) No Change (6-Kbytes, 4-Kbytes)	
Power Supplies		
—	40 MHz with two power supplies: nominal 3.3-V to 5.0-V power supplies	56 MHz with two power supplies: 5.0-V I/O, 2.6-V internal logic

[1] Available on some options.

The library provided from the third party (Matlab/Simulink) is used to program both microcontrollers (MPC5xx). On the other hand, after solving the synchronization issues between the microcontroller and the system modules, programming the microcontrollers using the Target Support Package FM5 becomes a possible task.

As mentioned previously in section (2.3.3), the MPC565 is an essential part of the ground system. The work done by the MPC565 can be represented as follows:

- Collect the data from the onboard system through the wireless device using UART-1 peripheral.
- Gather the position data from the ATmega16 microcontroller using the Input capture (pulse measurements). The pulse generated from the Atmega16 will be described more in detail in the following section.

- Transmit the control parameters and the position data through the wireless device to the onboard system.
- Monitor the data by transmitting it through the UART-2 to the GCS installed on a PC.

2.3.3.2 ATmega16 Microcontroller

The ATmega16 from ATMEL is a low-power; 8-bit microcontroller based on the AVR enhanced RISC architecture. The ATmega16 has the following features:

- High-performance, Low-power AVR 8-bit Microcontroller
- Advanced RISC Architecture
    - 131 Powerful Instructions – Most Single-clock Cycle Execution
    - 32 x 8 General Purpose Working Registers
    - Fully Static Operation
    - Up to 16 MIPS Throughput at 16 MHz
- High Endurance Non-volatile Memory segments
    - 16K Bytes of In-System Self-programmable Flash program memory
    - 512 Bytes EEPROM
    - 1K Bytes Internal SRAM
    - In-System Programming by On-chip Boot Program
- JTAG (IEEE std. 1149.1 Compliant) Interface
- Peripheral Features
    - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
    - Two 16-bit Timer/Counters with Separate Prescalers, Compare Modes, and Capture Modes
    - Real Time Counter with Separate Oscillator
    - Six PWM Channels
    - Dual Programmable Serial USARTs
    - Master/Slave SPI Serial Interface
- Special Microcontroller Features
    - Power-on Reset and Programmable Brown-out Detection
    - Internal Calibrated RC Oscillator

- I/O and Packages
    - 35 Programmable I/O Lines

More details about Atmega16 microcontrollers are available in the datasheets available on Atmel website. [15] Figure 2-21 shows the ATmega16 on the Atmel STK500 development board.

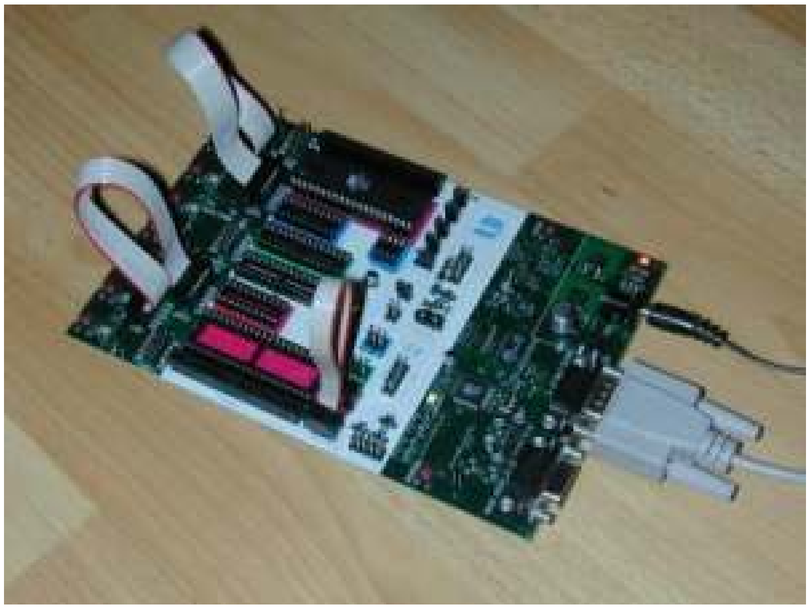


Figure 2-21: Atmel STK500 development board

The Atmega16 is considered as a helpmate microcontroller of the MPC565. The need of using more than two RS232 serial ports in the MPC565 (the only two ports that are connected to the wireless device and the PC) prevents the direct communication with the CMUcam3, where the camera provides us with the position data thorough the RS232 serial port.

Therefore, the ATmega16 is the link between the CMUcam3 and the MPC565. This done by acquiring the data from the CMUcam3 through the serial port, and converting the data to PWM signal transmitted through a digital line between the microcontrollers.

In the next section, the conversion process in the ATmega16 microcontroller will be explained after getting an idea about the communication protocol of the CMUcam3.

2.3.3.3 CMUcam3 Camera ***[16]***

The CMUcam3 from Carnegie Mellon University is a camera that provides simple vision capabilities to small embedded systems in the form of an intelligent sensor.

The CMUcam3 board communicates via a RS-232 or a TTL serial port and has the following functionality:

- Track user defined color blobs at up to 50 Frames Per Second
- Track motion using frame differencing at 26 Frames Per Second
- Find the centroid of any tracking data
- Gather mean color and variance data
- Gather a 28 bin histogram of each color channel
- Manipulate Horizontally Pixel Differenced Images
- Transfer a real-time binary bitmap of the tracked pixels in an image
- Arbitrary image windowing
- Adjust the camera's image properties
- Dump a raw image (single or multiple channels)
- Up to 352x288 Resolution
- Supports Multiple Baudrates: 115,200 57,600 38,400 19,200 9,600 4,800 2,400 1,200
- Control 5 servo outputs
- Slave parallel image processing mode off of a single camera bus
- Automatically use servos to do two axis color tracking
- B/W Analog video output (PAL or NTSC)
- Flexible output packet customization
- Multiple pass image processing on a buffered image

The hardware connections are provided in Appendix B.4. More details about CMUcam3 camera are available in cmucam website. [16]

One of the primary uses of the CMUcam3 in our project is to track or monitor specific color. The best performance can be achieved when the color contrast is high and intense. For instance, it can easily track a green dome of the AUS quadrotor on a white background, as shown in the Figure 2-22.

The CMUcam3 installed on a fixed fixture, as depicted in Figure 2-23. The camera provides us with the position data by tracking the green dome of the AUS quadrotor. In general, the readings of the position are limited to a small portion of area, and the sampling frequency is eight times less than the processing sampling frequency of the microcontroller.

These cons prevent us to control the position accurately, where the AUS quadrotor suffered from oscillation due to the slow sampling frequency. The small portion of area covered by the camera makes the controlling process more difficult as well.

The simulation results are shown in chapter 5. The results were over the expectation, where the quadrotor showed stabilization around the reference point with small drift.

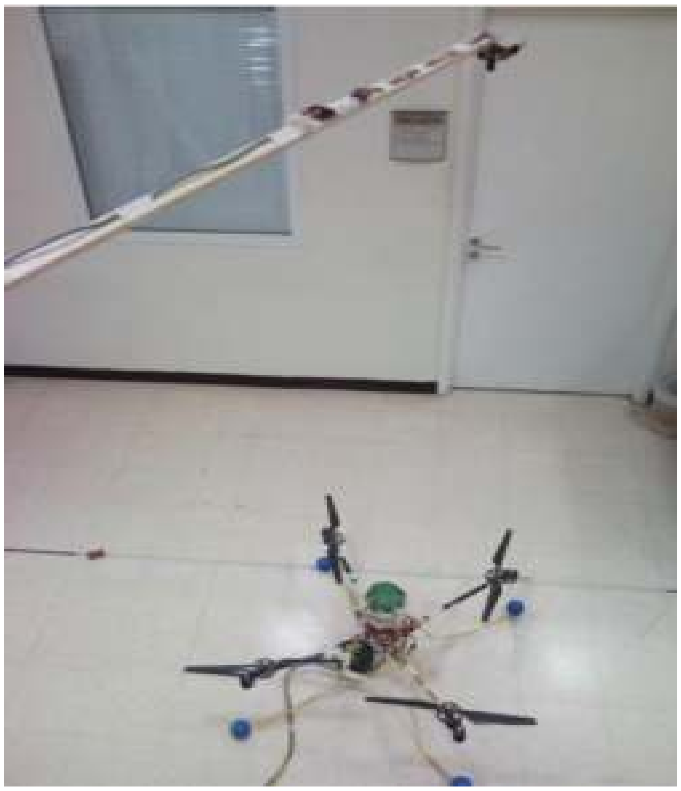


Figure 2-22: CMUcam3 setup to track the green dome of the AUS quadrotor

Figure 2-23 depicts the setup of the CMUcam3. Where, the camera is fixed on the roof.

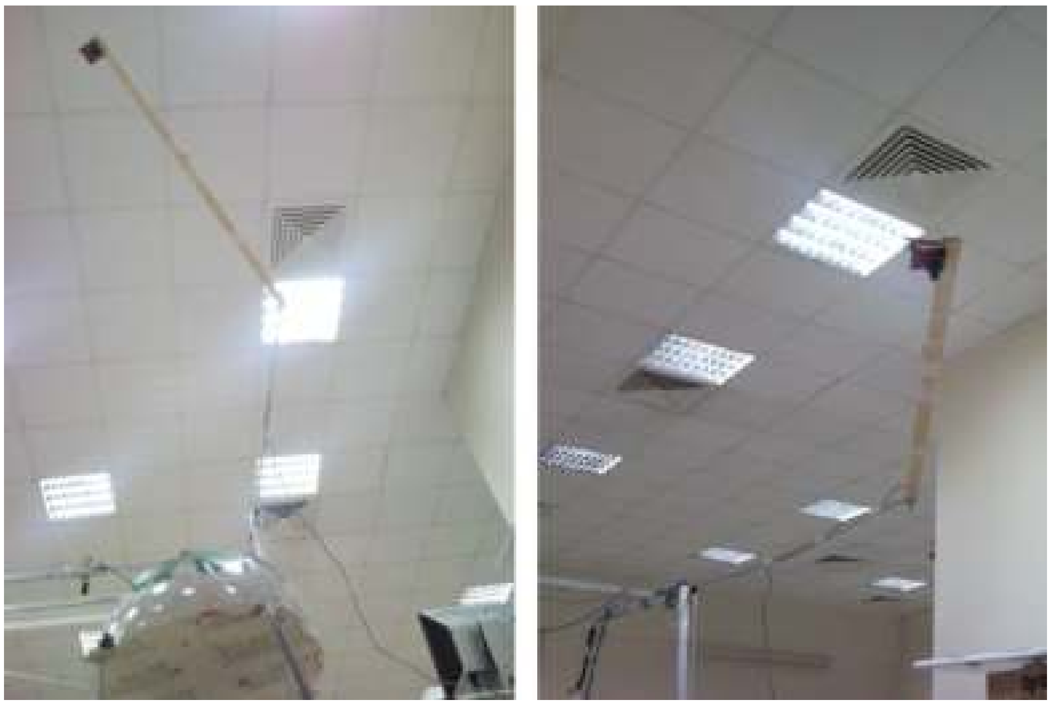


Figure 2-23: CMUcam3 Fixture

The process of getting the position is done by sending the range of the RGB color (the range of color that want to be detected) from the ATmega16 to the CMUcam3 through the RS232 serial port. The requested range can be defined by the following message:

TC		lowest Red		highest Red		lowest Green		highest Green		lowest Blue		highest Blue	CR

The CMUcam3 camera replies to the request by sending the following message:

TC		x-axis 0-255		y-axis 0-255		. . .

The position readings on the axes (x, y) are in the range of (0-255). Thus, the area covered by the camera depends on the camera angle of view; consequently it depends on the camera height from the ground. And because the height is fixed, the viewed area is fixed too. Figure 2-24 shows the relation between the height and the area covered by the camera.

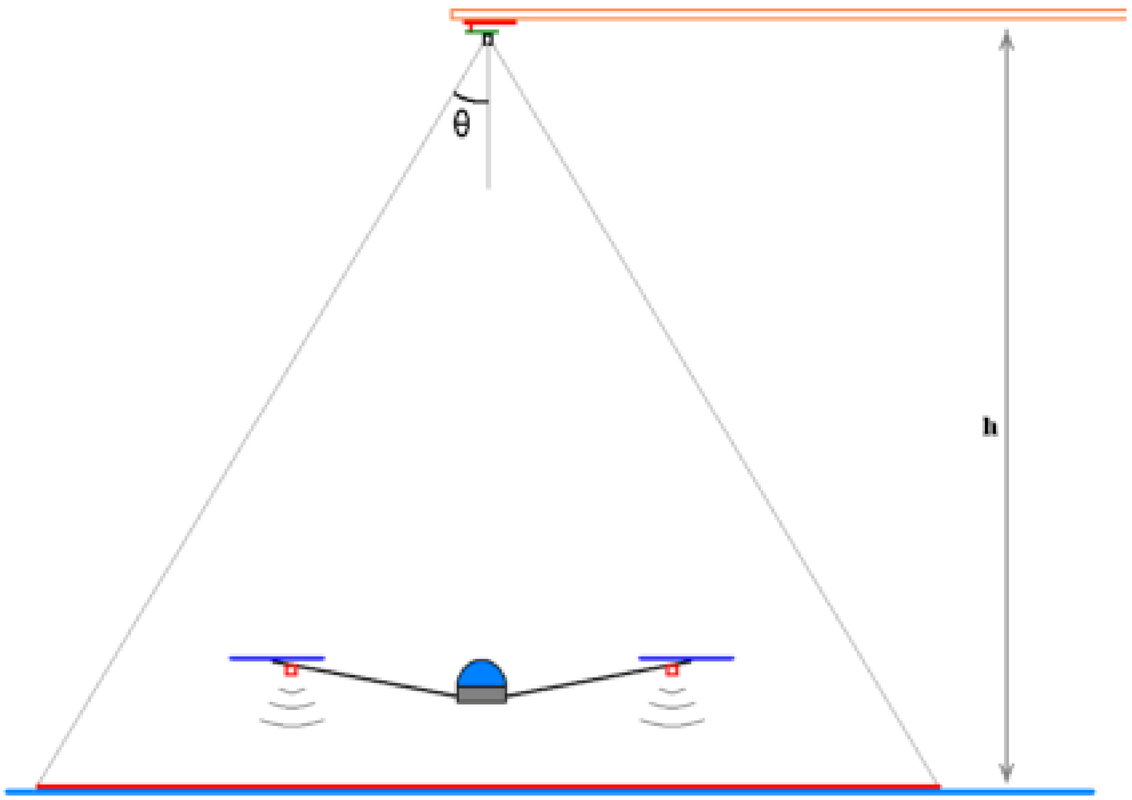


Figure 2-24: CMUcam3 angle of view

The relations between the height and the coordinates founded to be as follows:

□ □ 0.8 □□ , □ □ 0.6 □□

At the begging, we tried to fix the camera on the quadrotor to track a marked area on the ground as shown in Figure 2-25. But the position reading was so limited, because of the low height at take-off. Consequently, the area of view is so small, and the marked area is usually getting out the camera coverage.

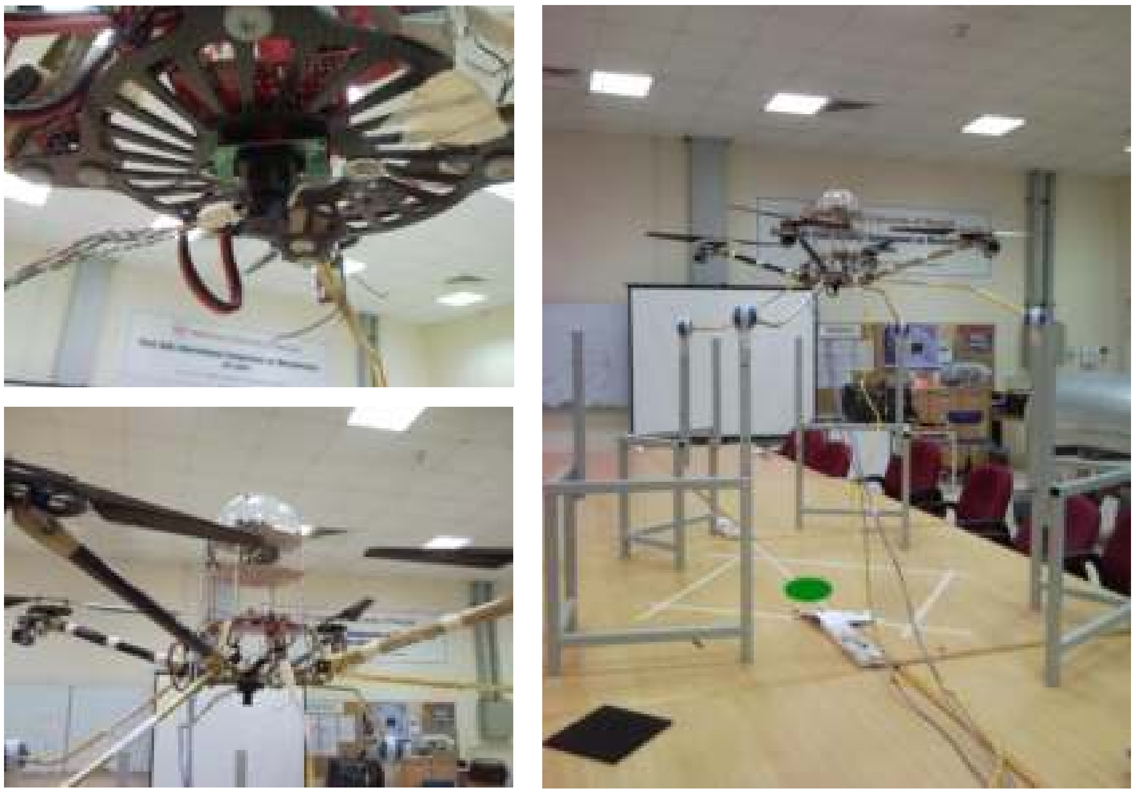


Figure 2-25: CMUcam3 fixture on the AUS quadrotor

Finally, The CMUcam3 is useful to demonstrate the position control of the AUS quadrotor in a closed enclosure. But for outdoor application (which has more priority than the indoor one) the IMU/GPS solution (Kalman filter) in the MTi-G is used to estimate the position and the velocity as mentioned in section 2.3.2.2.

Next, we will discuss the second information provided from the ground system to the onboard system, which is the MOS.

2.3.3.4 Manual Override System, MOS

The PPM signal provided from RC transmitter (as mentioned in section 2.2.4) is used to switch between the different flights modes on the quadrotor.

Before going to the procedure of commanding the quadrotor using the MOS, we should first understand the PPM signal and the way to read it from the RC transmitter using the MPC565 microcontroller.

The PPM signal is a sequence of the pulses that represent the channel readings. The sequence repeat itself every 20 ms approximately, which is staying fixed despite the change of the pulses width.

In the pulses sequence, the high pulses width are changing (from 0.6 to 1.6 ms) according to the channels reading, while the low pulses width are fixed (about 0.4 ms). The accumulative width of the sequence should remain fixed (20ms), therefore a synchronization pulse will substitute the remaining of the pulses widths. In another word, the summation of all pulses including the synchronization should remain 20ms.

Figure 2-26 is describing the PPM signal of 8-channel RC transmitter. From the previous discussion, the range of the synchronization pulse period can be calculated as follows:

[illegible]. [illegible] 20 [illegible]

Therefore, the minimum and maximum pulse width are:

[illegible]. [illegible] 20 [illegible] 8 [illegible] 1 [illegible] 12 [illegible]

[illegible]. [illegible] 20 [illegible] 8 [illegible] 2 [illegible] 4 [illegible]

By comparing the pulses period of the channels, we can find that each pulse does not exceed 2ms (from 1 to 2ms). Therefore, the synchronization pulse period is higher than the channel pulses, and it's easier to be captured from the microcontroller.

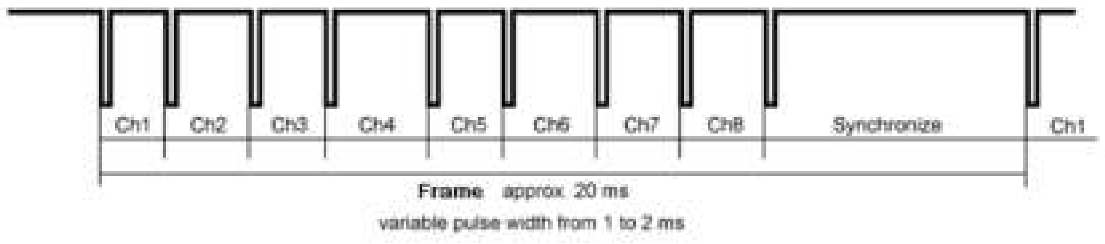


Figure 2-26: PPM Signal Description for 8-Channel RC Transmitter

The procedure of decoding the PPM signal is described in the following steps:

1. The MPC565 will capture the pulses width of the sequence continuously, using the Input capture pin.
2. While capturing the pulses, a small routine in the program will check if the pulse width is greater than 2ms, to indicate the synchronization pulse.
3. After indicating the synchronization pulse, the channels readings will be stored respectively in an array that has the size of the channels number.
4. The channel readings will be transmitted from the MPC565 to MPC555.

Before the real implementation on the AUS quadrotor, decoding of PPM Signal had been tested on the FMS simulator, as follows:

- The PPM signal is decoded using MPC555
- The readings of channels are converted to a serial protocol which commands the inputs in the FMS simulator. More in detail about the serial protocol can be found in [17].

Figure 2-27 shows the decoding process:

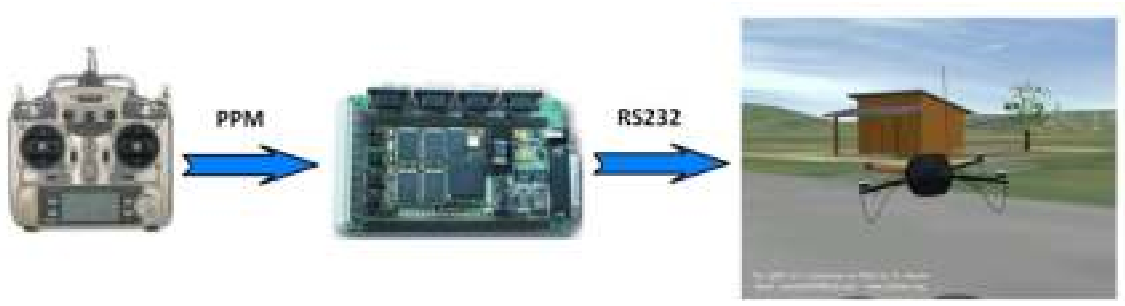


Figure 2-27: Decoding Process of the PPM Signal

In our project, our RC transmitter has 9-channel. Table 2-3 shows the functionality of each channel in our application:

Table 2-3: RC Transmitter channel readings functionality on the AUS quadrotor

Channel #	Function
Channel 1	Roll
Channel 2	Pitch
Channel 3	Thrust
Channel 4	Yaw
Channel 5	Different flight modes: Manual Autopilot
Channel 6	In Autopilot mode, different stabilization mechanism can be chosen: Attitude and Altitude stabilization (origin reference).

	Attitude and Altitude stabilization (changeable reference). (semi-automatic control using channel (1 to 4)) Position and velocity control (path following control)
Channel 7	Reserved
Channel 8	Reserved
Channel 9	Reserved

From Table 2-3, the pilot has the full functionality to change between the flights modes from the ground system. Any failure in the wireless transmission can cause catastrophic failure. Therefore, a Failure Detector System (FDS) in the onboard system is implemented. If any failure happens during the flight, the onboard system will command the quadrotor to hover and to land gradually. More about the FDS will be represented in safety section 2.4.

2.3.3.5 Ground Control Station, GCS

The pilot in the ground system needs to monitor the AUS quadrotor during the flight time. For that reason, GCS software should be installed in a PC to monitor the data and to tune the controller parameters during the flight. Figure 2-11 shows the bidirectional transmission line (RS232 serial) between the PC and the MPC565.

The GCS that is used in this project is a freeware from Microdrons called by mdCockpit [18]. It's used to retrieve data at flight time like:

- Attitude and Altitude readings
- GPS position and accuracy
- Speed
- Temperature
- Battery state
- Direction of the compass
- 3D recording of the flight path
- Real-time data export to Google Earth

Screenshots of the mdCockpit GCS are depicted in Figure 2-28.

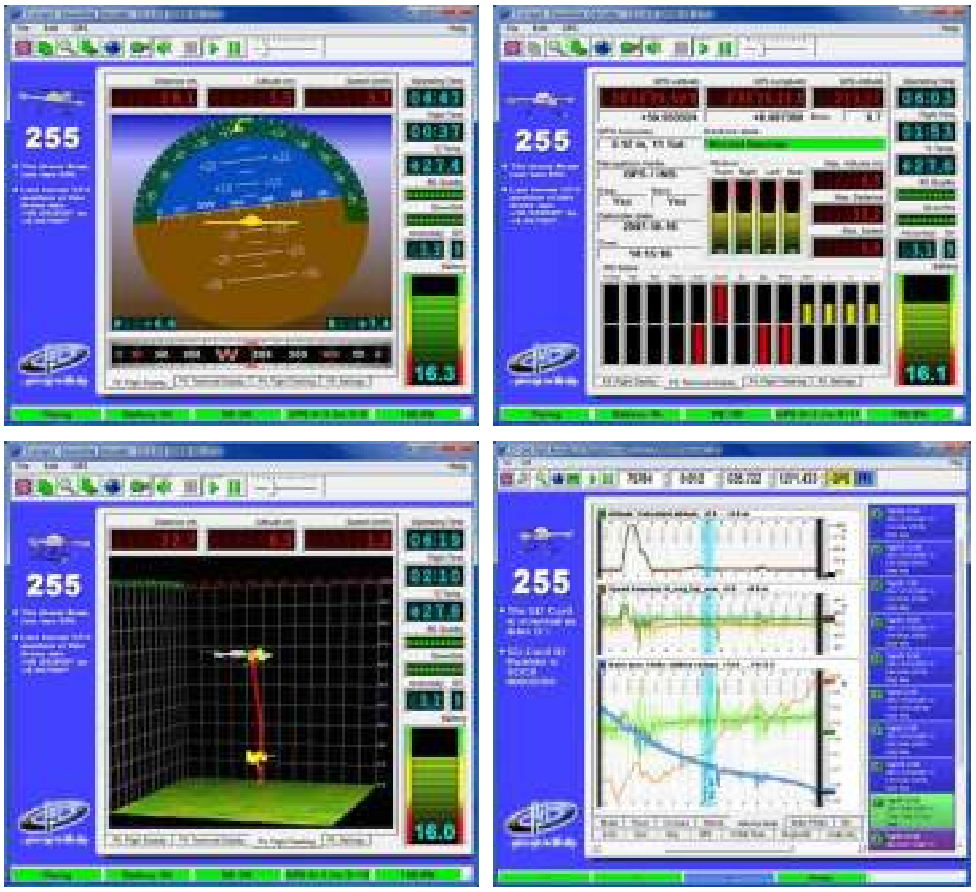


Figure 2-28: mdCockpit GCS

2.4 SAFETY

A professional work is always evaluated by its safety precautions. The importance of the safety issues is represented to reduce the failures incidence possibilities on the user and quadrotor.

In this project, the safety issues are categorized into mechanical and electrical safety parts. In the mechanical safety part, at the first stage of this project, two test stands are constructed with safety barriers are installed as well. In the later stage, the training set is installed on the quadrotor, and is tethered to restrict the motion to certain area.

On the other hand, the electrical safety part is dealing with the electronic and electrical failures. The MOS, FDS, batteries status are the components of the electrical safety part.

2.4.1 Mechanical Safety

2.4.1.1 Test Stands

The reason of using the test stands is to aid the development of a model and controllers for the X-Pro, without the risk of flying in indoor or outdoor environments.

Two test stands are constructed to test the quadrotor stabilizations, at the early stage of this project. The first test stand is made to perform pitching and rolling stabilizations individually, where the vehicle is allowed to freely move around one axis (1DOF), as shown in Figure 2-29.

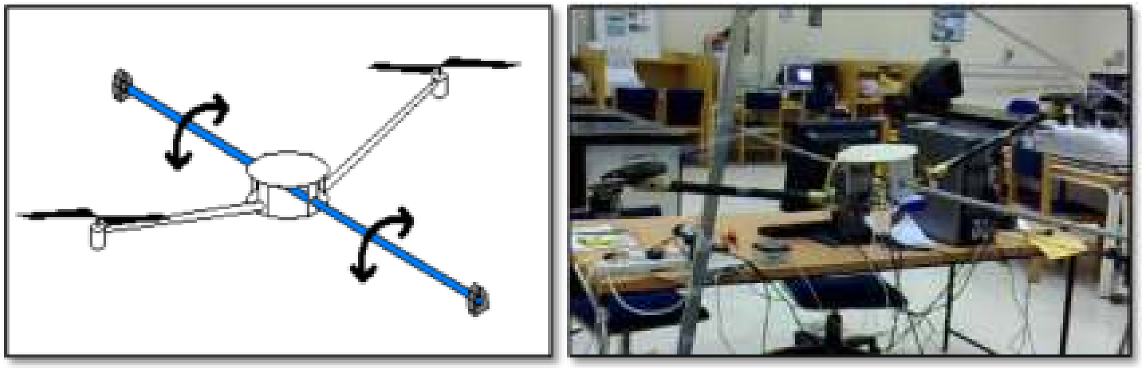


Figure 2-29: 1-DOF Test Stand

The 1DOF test stands constructed from long rod fixed on the opposite axis of rotation of the AUS quadrotor, as shown in Figure 2-29.

The second test stand is made to give the mobility to the AUS quadrotor to move in 4DOF. It allows the vehicle to stabilize around the attitude and altitude axes. Screenshots of the 4DOF test stand are depicted in Figure 2-30.

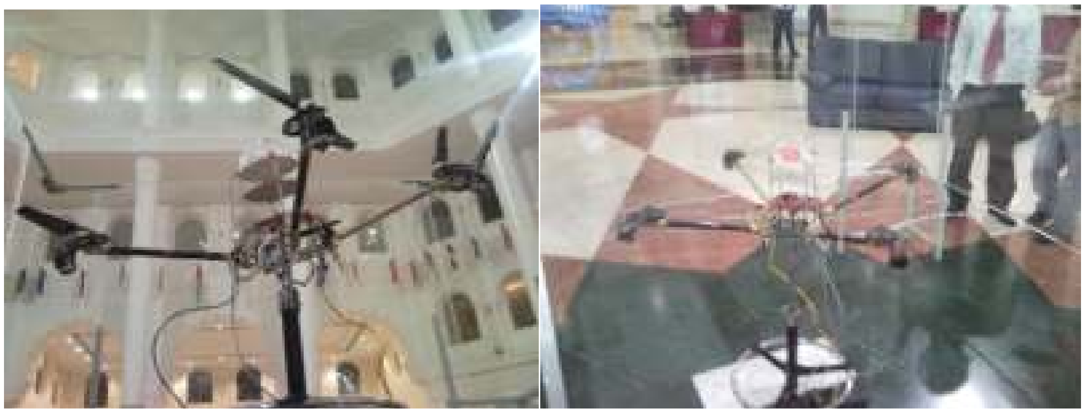


Figure 2-30: Different Views of the 4DOF Test Stand

The test stand manufactured in a way to adjust the attitude and the altitude of the vehicle at the same instant.

For attitude stabilization, a ball joint assembled at the COG of the quadrotor. The ball joint allows the rotorcraft to tilt within ±45^{o} around the *x* and *y* axes (pitch and roll motions), and to rotate freely around the z-axis (yaw motion).

On the other hand, the quadrotor (with the ball joint fixture) are mounted on a cylinder rod that slides inside the basement cylinder. This configuration allows the quadrotor to adjust the height within half a meter. Figure 2-31 depicts the setup of the 4DOF test stand.

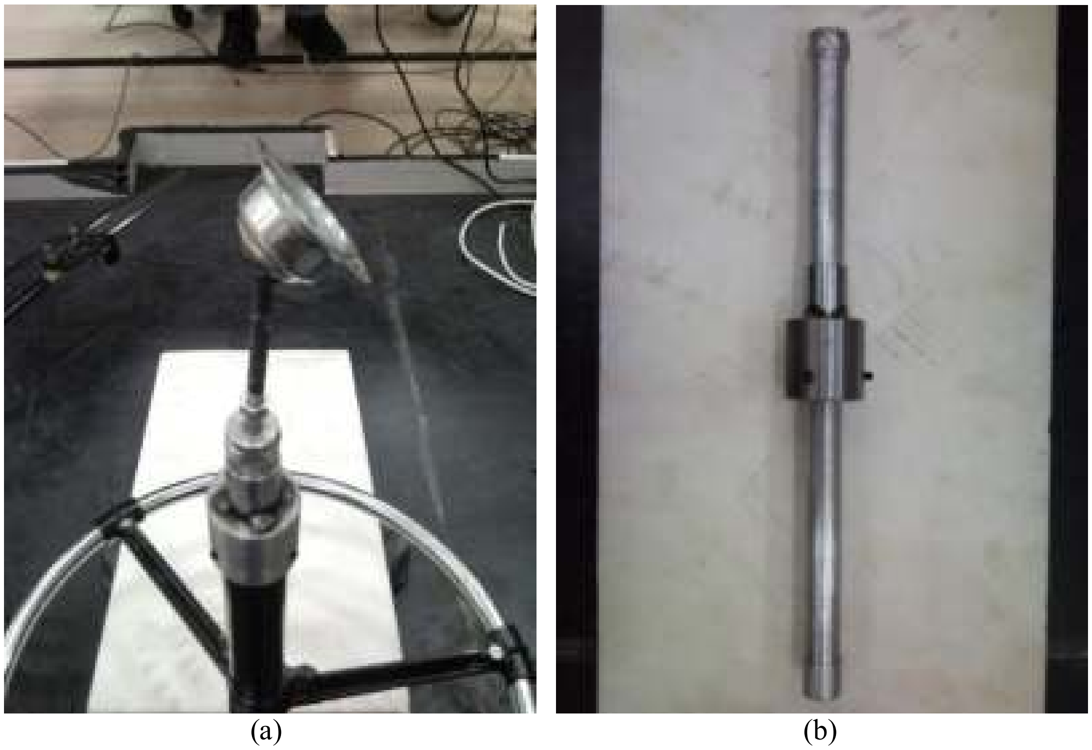

(a) (b)

Figure 2-31: (a) ball joint fixed on the cylinder rod, (b) cylinder rod

2.4.1.2 Plexiglass Barrier shield

The probability of a system failure is increased during the test flight even with using the test stands. Where the carbon fiber blades are rotating in a high speed and it could be causing an injury or some time it could kill the person who is standing beside the quadrotor while operating. Therefore, a barrier shield is manufactured using four plexiglass sheets. The plexiglass sheets are mounted on a basement made from square pipes. The airflow has drastically effect on the quadrotor stabilization; therefore the basement raises the plexiglass sheets half a meter from the ground to let the air flows smoothly during the flight. Figure 2-32 shows the plexiglass barrier shield.

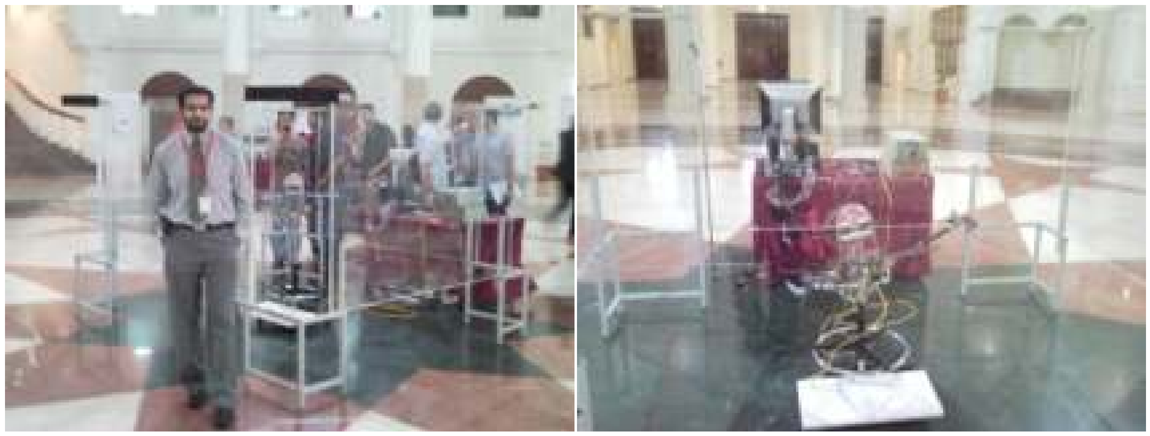


Figure 2-32: Plexiglass Barrier Shield

2.4.1.3 Training Set

The training set is a handy kit that helps the newbie RC pilot in his flight test. The training set is helping to prevent the quadrotor from flipping over by keeping the blades far away from the ground. As well as, it reduces the impact forces when the quadrotor hit the ground. Figure 2-33 depicts the installed training set on the quadrotor.

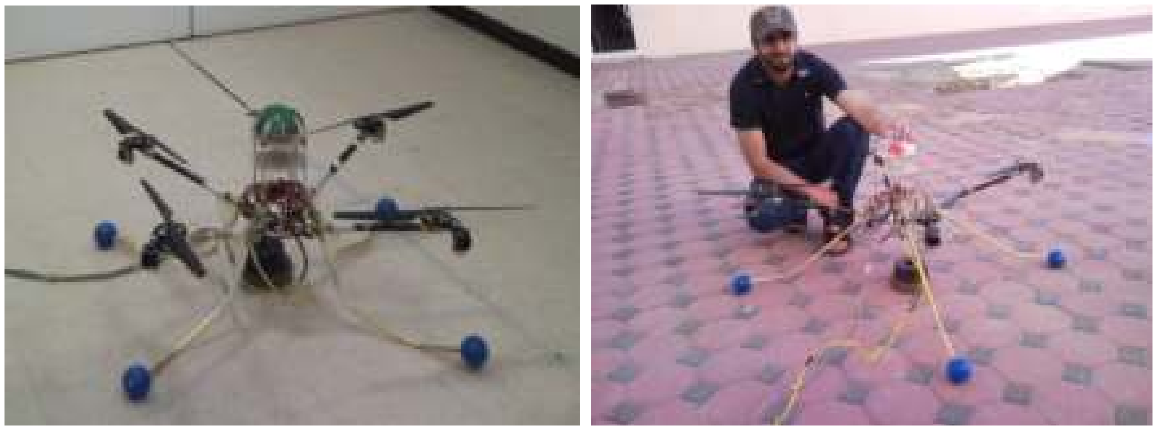


Figure 2-33: The Training Set installed on the quadrotor

The mechanical safety is not enough to minimize the probability of failure occurrence. In addition to the mechanical safety, the electrical safety will try to keep both, the human and the quadrotor safe as much as possible. Next, the electrical safety will be represented.

2.4.2 Electrical Safety

2.4.2.1 Manual Override System, MOS

The MOS helps to switch between the different flights modes. But the pilot should be aware to avoid any failure while switching and to choose the appropriate switching time as well. More details about the MOS were represented in section 2.3.3.4.

2.4.2.2 Failure Detector System, FDS

The FDS is a failure detector algorithm in the onboard system microcontroller, MPC555. This algorithm switches the quadrotor to the hover mode as soon as any failure to the wireless communication is occurred. The algorithm is illustrated as following:

1. The MPC555 always checks the presence and the checksum of the message protocol that received wirelessly from the ground system.
2. If the MPC does not receive the message or the error in the checksum is occurring for more than 50 ms, then the MPC555 will switch between the current flight mode to the hover mode.
3. The thrust will be decreased gradually until the four rotors completely stop.

2.4.2.2 *Batteries Status*

From the discharging chart of the Li-Po cell batteries, as shown in Figure 2-34. The voltage of the cell suffers from a sudden drop. The sudden drop found to be equal to a 90% of the rated voltage. The voltage status is tracked using an A/D converter on the onboard system by the MPC555. This is will prevent the sudden drop of the battries voltage, and it will keep the quadrotor safe from any shortage of the power supply during the flight.

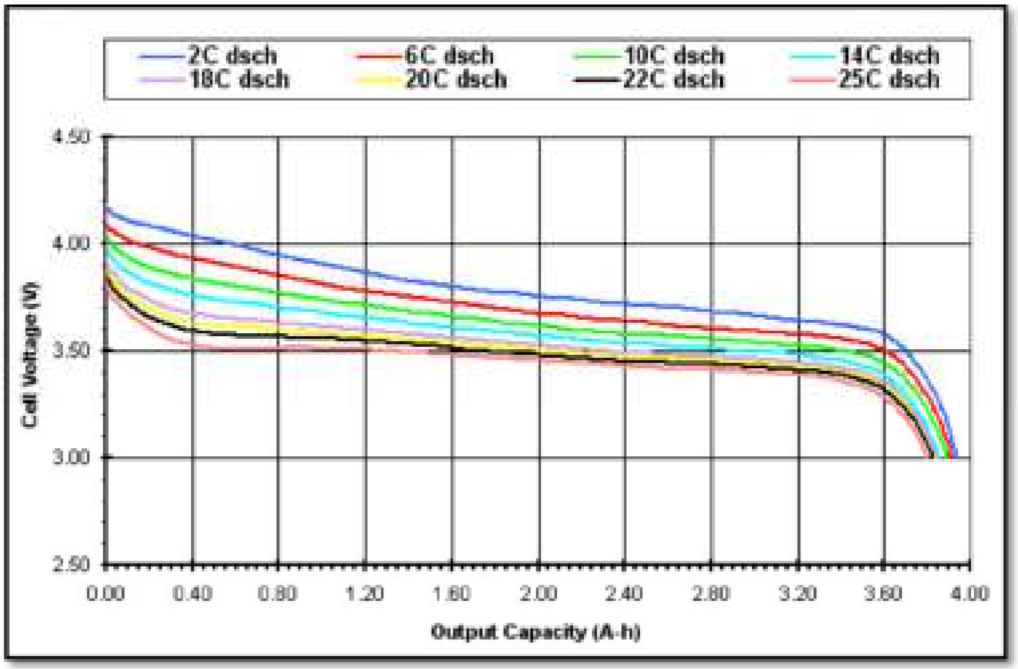


Figure 2-34: Typical Thunder Power Li-Po Discharge Cycle

In addition to that what we mentioned about the precautions that should be taken in the mechanical and electrical safety sections, the pilot should be aware while testing the quadrotor, especially in the development stage of the quadrotor. Where, the probability of occurring a failure is higher than the final stage.

The pilot should keep safe distance between him and the quadrotor, and he should wear a hat in the sunny environments. Also he should prevent the people from getting close to the quadrotor during the flight time.

CHAPTER 3

QUADROTOR MODEL

3.1 INTRODUCTION

In this chapter, the mathematical model of the X-pro quadrotor will be derived. The result is a set of nonlinear equations representing the behavior of X-pro in flight. There are two approaches to obtain the model: First-Principle Modeling (FPM) and System Identification (SID).

FPM is based on physical principles and laws to represent the system dynamic behavior. It is a bottom-up procedure where all components of the system are described individually, and then combined to represent the complete system. In the SID approach, the system is treated as a black box, where the focus is shifted from understanding the physical system to the input/output response behavior.

Obtaining and validating the model using FPM is a complex task. It requires great deal of mathematical derivation and thorough analysis. On the other hand, the SID is an effective method of obtaining a model from experimental data and it has been preferred in most rotor UAV projects.

Aalborg University quadrotor team represents in [4] the SID for the four actuators (motor, gear, and rotor mechanism) in the X-pro. The actual correlation between the input signals and the force generated by each actuator is successfully predicted by the SID. This is advantageous method since a reliable model is obtained without going into the complexity of deriving the mathematical equations; especially that is related to the aerodynamics.

A shortfall of SID is that the model is only valid at the operating point from where the data is collected, or in the vicinity of this operating point. Generally it can be said that FPM and SID complement each other.

Primarily, the FPM will be used in deriving the rigid body dynamics and kinematics, while the force generating process from the actuators will be modeled using the SID. A similar X-pro quadrotor to the Aalborg team [4] is used in this project. Therefore, the work done by the team using the SID will be utilized in obtaining the actuator model.

The model will be divided up into two main parts which are shown in Figure 3-1.

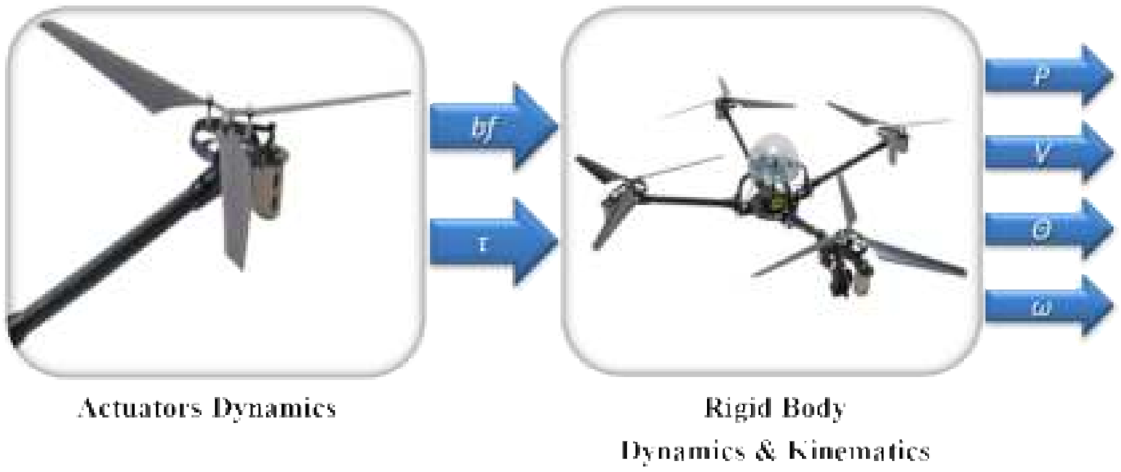


Figure 3-1: Block diagram of the components in the quadrotor model.

The first block describes the forces (lift/drag) and the moments generated by the actuators. The dynamics in the second block renders the angular and linear accelerations from the total forces and moments, and the kinematics translates the dynamics of the quadrotor to Euler angles, position, and their rates.

The equation of motions will be derived by combining the two parts. The complete mathematical description will be the joint to simulate and design a controller for the X-pro quadrotor.

To precisely describe the position and the orientation of the X-pro quadrotor, the following is needed:

- a description of the various reference frames and coordinate systems
- a transformations between the coordinate systems

Hence, an overview of the coordinate systems and their transformations will be represented next.

3.2 REFERENCE FRAMES

To describe the equation of motions of a quadrotor, a set of basic frames and notations must be defined:

- A frame with constant inertia of the quadrotor will be designated as a Body Frame (BF). The basis vector of BF will be denoted by ▯▯▯, ▯▯, ▯▯▯ The BF will identify the orientation of the quadrotor according to the right-hand rule at the Centre of Gravity (CG).
- A fixed frame (not accelerated frame) is required to describe the position and the translational motion of the quadrotor body. This inertial frame will be referred to the earth frame (EF) with the basis vector ▯▯▯, ▯▯, ▯▯▯ The EF could be freely positioned on the earth surface, which is non-accelerated compared to the quadrotor.

Figure 3-2, shows the reference coordinate systems.

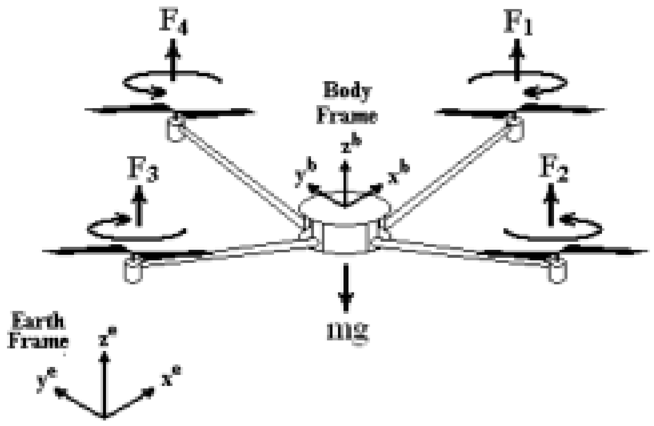


Figure 3-2: Reference Coordinate Systems

The purpose of describing various reference frames is necessary for the following reasons [19]:

- Aerodynamic forces and torques are applied in the BF.
- Equations of motion are derived based on the BF.
- On-board sensors (rate gyros and accelerometers) measure the data with respect to the BF.
- GPS and Magnometer measurements are related to the EF.
- Flight trajectories and map information are given in the EF.

The transformation between the coordinate systems is Important for many applications. e.g. data filtering, controlling … etc.

Next, a description of the transformation between the coordinate system will be represented.

3.3 TRANSFORMATION

3.3.1 Rotation Matrix

Forces and moments generated by the actuators are represented in the BF; nevertheless, the control of the quadrotor is preferred to be with respect to the EF. Therefore, any arbitrary vector in BF can be expressed in the EF by using the rotation matrix R_B^E.

The rotation matrix can be found using Euler angles with rotation arranged in 3-2-1 order. In this sequence, the order of rotation begins with z-axis then y-axis and lastly x-axis, using the right-handed rotation of the coordinate system. [20]

The right-handed rotation of the coordinate system about the z-axis gives:

$$R_z^E(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

Proceeding in a similar way, a right-handed rotation of the coordinate system about the y-axis gives:

$$R_y^E(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.2}$$

Finally, the right-hand rotation of the coordinate system about the x-axis:

$$R_x^E(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{3.3}$$

Due to the orthonormality transposing principle, the resulting transformation matrix R_B^E is:

$$R_B^E = R_z^E(\psi)\,R_y^E(\theta)\,R_x^E(\phi) =$$

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \cos(\theta)\sin(\psi) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{3.4}$$

The functionality of the rotation matrix can be utilized by expressing the position vector X^B in BF to its equivalent vector X^E in EF:

$$X^E = R_B^E \, . \, X^B \tag{3.5}$$

3.3.2 Euler Rates

$\omega^B = [p \;\; q \;\; r]^T$, are the body anglular rates of quadrotor about the BF axes. The relationship between the body angular rates and the Euler rates $\dot{\Theta} = [\dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi}]^T$ is complicated by the fact that they are from different coordinate systems.

The body angular rates are expressed in different frames. In order to add them, the first Euler angle has no additional rotations due to the rotational order, second angle has one additional rotation and the third has two additional rotations [19]:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}^{-1} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{3.6}$$

Therefore;

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.7}$$

Inverting the transformation matrix results in:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan(\theta)\sin(\phi) & \tan(\theta)\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sec(\theta)\sin(\phi) & \sec(\theta)\cos(\phi) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.8}$$

3.4 Actuator Dynamics

3.4.1 Introduction

A tremendous work by the Aalborg university team in [4] and [21] had been done to derive the mathematical representation of the actuator dynamics using FDM and SID techniques.

In this project, the derived actuator model in their work will be used for the following reasons:

1. Similar X-pro quadrotor is being used in this project. Where, the parameters of the quadrotor are the same.
2. Both FPM and SID had been used to describe the actuator dynamics, followed by a validation of the model.
3. This work is focused in the control side.

In [21], the FPM is used to model the actuator dynamics. The complete model is a combination of electrical (motor) and mechanical (gear and rotor) parts. On the other hand, the SID is used in [4], where a transfer function that describes the input/output relationship of the actuators was extracted.

In this project, a direct input/output relationship that represents the actuator dynamics in [4] will be adopted. Thus, the complexity of mathematical derivations will be avoided on each part of the actuator components.

3.4.2 Actuator Model

Each actuator of the X-pro quadrotor consists of: motor, gear, and rotor. The motor/gear mechanism will be excluded from the rotor model, due to the limitation of the forces measurements.

The process of obtaining the mathematical description of the actuator dynamics is divided into two steps:

1. The SID will be used to get the relationship between the Input signal (PWM) and the output angular rate of the motor/gear mechanism.
2. The correlation between the angular velocity and the generated thrust of the rotor is obtained manually by using a set of measurement points.

The block diagram of the motor/gear and rotor is illustrated in Figure 3-3:

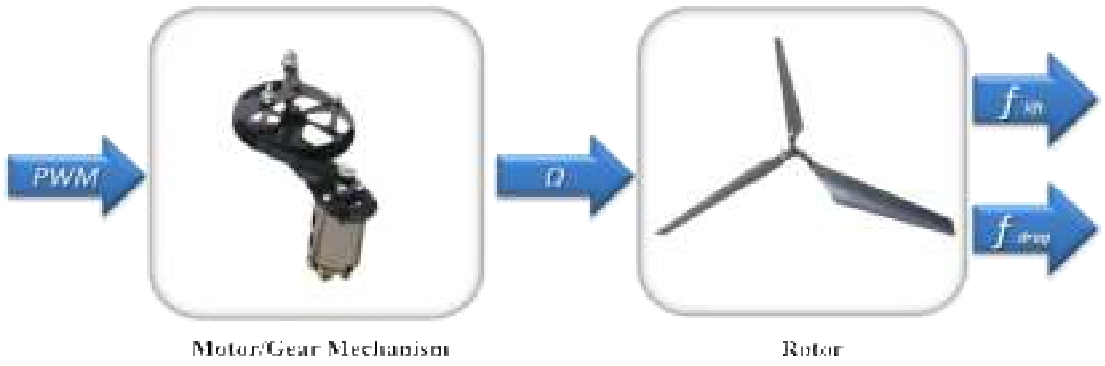


Figure 3-3: Block diagram of the components in the Actuator model.

3.4.2.1 System Identification of Motor/Gear Mechanism

SID is an effective method of representing a system by extracting a transfer function that relates the outputs to the inputs. Figure x.3 shows the Input/Output signals of the motor/gear mechanism.

The output signal is the angular velocity of the gear measured in [rad/sec]. And the input signal is the PWM, which is supplied to the motor driving circuit (e.g. h-bridge, power amplifier …). The driving circuit will provide the motor a voltage that is proportional to the PWM duty cycle.

The transfer function of the Input/Output signals can be expressed as:

[illegible] (3.9)

According to the work done by Aalborg university team in [2], the estimation of the transfer function was performed as follows:

1. SID model structure
2. Collecting the data
3. Parameter estimation
4. Validation

The model structure is chosen based on the knowledge of the noise aspects and the physical behavior of the system. In [4] ARX model had been chosen, because the white Gaussian noise is the only expected disturbance on the system.

After collecting the data, the parameters were estimated using Matlab software, and the spectral analysis was performed to specify the order of the transfer function. The result was a second order linear system model.

The discrete transfer function is found to be:

$$G(z) = \frac{0.0433z + 0.0227}{z^2 - 1.6306z + 0.64960} \qquad (3.10)$$

Finally, the output of the estimated model was compared with the measured output signal. Figure 3-4, shows good matching between the outputs:

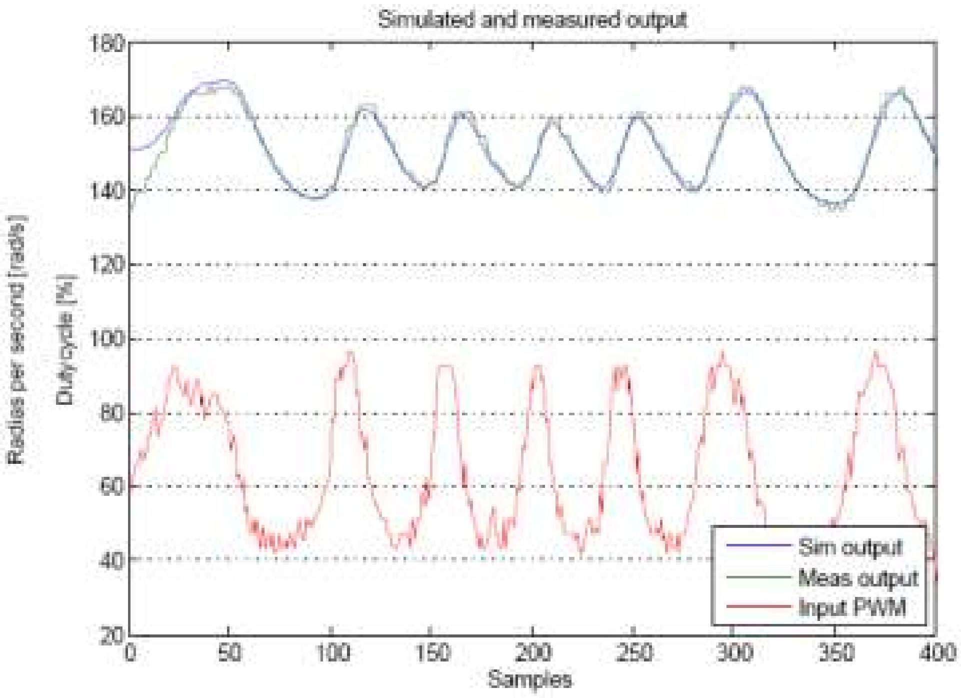


Figure 3-4: Validation of the estimated model

Therefore, the motor/gear dynamics is represented by a transfer function that relates the PWM signal to the rotor angular velocity. Because of the direct connection between the rotor and gear, the angular velocity is the same.

Next, the relationship between the angular velocity and the generated forces of the rotor will be represented.

3.4.2.2 Rotor dynamics

The rotor is a set of tilted blades that pushing the air downwards while rotating. The power provided by the motor/gear mechanism to the rotor, produces lift and drag forces. The aerodynamics forces of the rotating blades are affected by the air, the rotor, the position of quadrotor, and the pitch angle of the blades. Consequently, the derivation of a mathematical model is a complex task.

Restraining the model into hover flight simplifies the model considerably. The resulting model can be described by a second-order polynomial, taking the rotational speed as input and providing lift and drag as output.

Lift Force Measurements:

To measure the lift force of each actuator, a test bed that allowed the quadrotor to only pitch was constructed by [4].

The setup of the test bed is illustrated in Figure x.5:

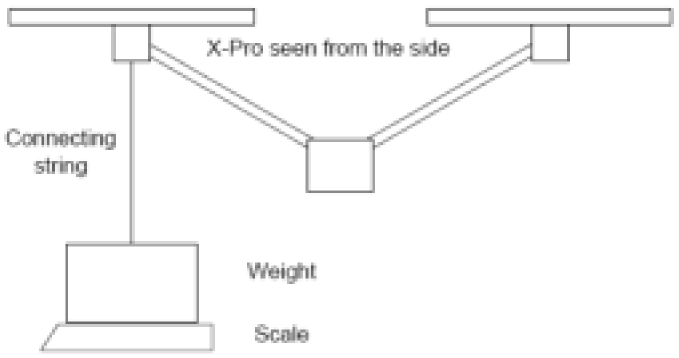


Figure 3-5: The test bed for measuring the lift force [4]

The X-pro is restrained by strings to prevent the rotation in the yaw and the roll directions, and then the actuator is attached to a weight by a string. By increasing the rotational speed of the rotor, the lifting force will be increased and the scale reading will be decreased. The lift provided by the rotor can be directly derived from the readings on the scale knowing that:

[illegible] (3.11)

The angular velocity of the rotor was controlled manually by using an RC transmitter in [4]. After collecting the data, the data fitted using Matlab and a mathematical equation was extracted, which relates the angular velocity to the lift force that generated from each rotor, as follows:

$$[illegible]\Omega[illegible]\ [illegible]\ 0.2263\text{x}10^{[illegible]}.\Omega^{[illegible]}\ [illegible]\ 2.488\ \text{x}10^{[illegible]}.\Omega\ [illegible]\ 11.97\ \text{x}10^{[illegible]}\ [illegible] \qquad (3.12)$$

Drag Force Measurements:

To measure the drag force, a Newtonmeter is fixed horizontally as shown in Figure 3-6. The quadrotor is only allowed to move in the yaw direction.

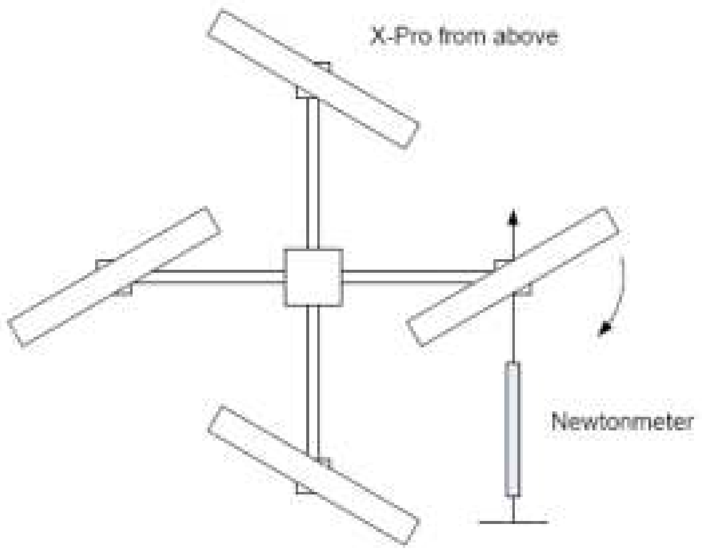


Figure 3-6: The test bed for measuring the drag force [4]

The Newtonmeter is fixed perpendicular to the rotor in the horizontal plane. Where, it’s parallel to the drag force. Because the quadrotor is rotating around the z-axes (yaw), the drag force will not be more parallel to the Newtonmeter . Due to the short arm distance, the readings are considered to be accepted.

Again the angular speed of the rotor controlled manually and the speed calculated using Halleffect sensor.

By fitting the collected data using Matlab software, a mathematical equation that relates the drag force to the angular speed was extracted and found to be, as follows:

$$[illegible]\Omega[illegible] \; [illegible] \; 26.57\text{x}10^{[illegible]}.\Omega^{[illegible]} \; [illegible] \; 1.933\,\text{x}10^{[illegible]}.\Omega \; [illegible] \; 18.84\,\text{x}10^{[illegible]} \quad [illegible] \tag{3.13}$$

Finally, the full model of the actuator dynamics was obtained by combining the transfer function of the motor/gear mechanism with the rotor forces equations. In total, the resultant model is relating the PWM input signal with the output forces generated from each actuator.

3.5 Rigid Body Dynamics and Kinematics

The quadrotor is a complex mechanical system. Understanding the aerodynamics and all mechanics domains will help to derive a reliable mathematical model of the quadrotor.

Basic assumptions are presented to develop the model of quadrotor [5], the following assumptions are considered:

- Quadrotor body and propellers are rigid
- The quadrotor has symmetrical structure
- The CG and the BF are coinciding.

The rigid body model of the quadrotor will be derived based on Newton-Euler formalism. Accordingly, a superior representation of the internal and external forces on the quadrotor structure makes the model more realistic, especially in the forward motion.

Next, the forces and moments that affect the quadrotor movements will be described thoroughly.

3.5.1 Force Dynamics and Kinematics

The forces regarded on the X-Pro quadrotor are:

- *Rotor Lift Force:*

Each actuator in the quadrotor generates a lift force (thrust) according to the rotational speed of the rotor. The lift force is the internal force that controlling the quadrotor motion. The direction of the forces is given in the z-axis of the BF.

The relationship between the thrust and the rotational speed of the rotor is presented in equation (3.12). The relation is valid on hover when the quadrotor is considerably far from the ground, where the lifting force will be larger at the ground level due to the increasing of the pressure below the blades [21].

➢ *Gravity:*

The gravity force has a large influence on the quadrotor dynamics. The direction of the gravity is the z-axis of the EF, passes through the CG and points toward the ground.

➢ *Ground Effect:*

The rotor experience thrust augmentation due to the better efficiency when the rotor is operating near to the ground. It's related to the reduction of the induced airflow velocity. By capturing the variation of the induced inflow velocity, the thrust coefficient will be varied accordingly.

In this thesis, the ground effect is neglected. The mathematical description of the forces in section (3.4) was taken in hover mode flight.

In this model we will include the thrust and the gravity forces. Hence, the total force on the BF:

[illegible]

(3.14)

[illegible] , representing the EF relative to the BF transformation. Where;

[illegible] (3.15)

Consequently, the forces on the EF are:

[illegible]

(3.16)

On the other hand, and according to the Newton's second law:

[illegible] (3.17)

Where; [illegible]

From (3.16) and (3.17), the translational equations of motion on the EF:

$$[illegible] = \frac{1}{[illegible]} [illegible] . [illegible] [illegible] [illegible] \qquad (3.18)$$

Knowing that:

$$[illegible] = \begin{bmatrix} 0 \\ 0 \\ 9.82 \end{bmatrix} [illegible]$$

$$[illegible] = \begin{bmatrix} 0 \\ 0 \\ [illegible] \end{bmatrix} [illegible]$$

(3.19)

Where, [illegible] is the magnitude of the lift force from the *i* th rotor.

Equation (3.18) can be rewritten in the three axes of the EF. Therefore, the dynamics of the translational motions are:

$$[illegible] = \frac{1}{[illegible]} \begin{bmatrix} [illegible] \sin[illegible] \cos[illegible] [illegible] \sin[illegible] \sin[illegible] . [illegible] [illegible] \\ [illegible] \sin[illegible] \sin[illegible] \cos[illegible] [illegible] \cos[illegible] \sin[illegible] . [illegible] [illegible] \\ [illegible] \cos[illegible] \cos[illegible] [illegible] \end{bmatrix} \qquad (3.20)$$

From the kinematics of the translational equations of motion, the velocity and the position can be described by integrating the linear acceleration in (3.18) twice, as follows:

[illegible]

[illegible]

(3.21)

Where;

[illegible] & [illegible]

For controlling purposes, the translational equations of motion are derived with respect to the EF frame. Where, the navigation sensory devices provide the measurements in the inertial frame, which is here the EF.

3.5.2 Moment Dynamics and Kinematics

The moments that act on the quadrotor are:

- *Drag Moment:*

Defined as, the moment generated about the rotor shaft due to the horizontal aerodynamic forces acting on the blades (drag forces). The drag forces are described in equation (3.13). The resultant moment produced from the cross product will be perpendicular to the force and the moment arm P, and it causes the yaw motion, as this equation shows:

$$\vec{M}^{b}_{drag} = \sum_{i=1}^{4} \left(\vec{P}_i \times \vec{F}^{b}_{drag,i} \right) \tag{3.22}$$

- *Lift Moment :*

The lift moment is the moment created by the rotor lift around the CG; this is equal to the lift force times the distance between the shaft of the rotor and the CG. The lift moments cause the pitch and roll motions.

$$\vec{M}^{b}_{lift} = \sum_{i=1}^{4} \left(\vec{P}_i \times \vec{F}^{b}_{lift,i} \right) \tag{3.23}$$

Figure 3.7 illustrates the drag and the lift moment directions:

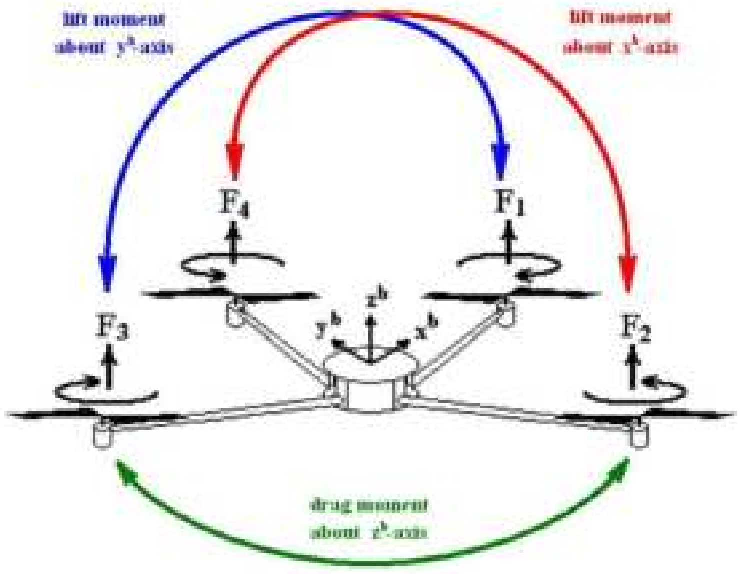

lift moment about y^b-axis
lift moment about x^b-axis
F4
F1
F3
F2
z^b
y^b
x^b
drag moment about z^b-axis

Figure 3-7: Drag and lift moment directions

The derivation of the angular momentum equation of a rigid body about its CG is given as:

[illegible] (3.24)

Where H is the angular momentum vector of the rigid body about its CG, and [illegible] is the external moment acting on the quadrotor about its CG. Angular momentum vector can be defined as:

[illegible] (3.25)

Where, *I* denotes the inertia matrix of the quadrotor.

[illegible] (3.26)

Hence, equation (3.24) becomes:

[illegible]

[illegible]

(3.27)

The mass distribution of the quadrotor is constant; therefore, [illegible]/[illegible] [illegible] 0. Consequently, the angular acceleration can be derived in the following way:

$$\text{[illegible]}$$

$$\text{[illegible]} \qquad (3.28)$$

Because of the symmetry in the quadrotor structure [4], the inertia matrix in (3.26) can be written as follows:

$$\text{[illegible]} \begin{bmatrix} \text{[illegible]} & 0 & 0 \\ 0 & \text{[illegible]} & 0 \\ 0 & 0 & \text{[illegible]} \end{bmatrix} \qquad (3.29)$$

Therefore, the rotational equations of motion can be written as follows:

$$\text{[illegible]}$$

$$\text{[illegible]}$$

$$\text{[illegible]} \qquad (3.30)$$

Where, [illegible]. By integrating the angular acceleration, the angular velocities in body frame are:

$$\text{[illegible]} \qquad (3.31)$$

The Euler rates in EF can be found by using the transformation matrix *T*, in (3.31)

$$\Theta \; \text{[illegible]} \; T.\text{[illegible]} \qquad (3.32)$$

Consequently, the Euler angles can be found by integrating the Euler rates :

$$\Theta \; \text{[illegible]} \; \Theta \qquad (3.33)$$

3.6 Equations of Motion

The equations of motion for the translational and rotational subsystems are derived in equations (3.20) and (3.30). The state space representation of the nonlinear model could be written as follows:

$$X\text{[illegible]}, \text{[illegible]}$$

$$\begin{bmatrix} \text{[illegible]} \end{bmatrix} = \begin{bmatrix} \frac{1}{\text{[illegible]}}\left(\text{[illegible]}\sin\text{[illegible]}\cos\text{[illegible]} + \sin\text{[illegible]}\sin\text{[illegible]}\right)\sum_{\text{[illegible]}}^{\text{[illegible]}}\text{[illegible]} \\ \frac{1}{\text{[illegible]}}\left(\sin\text{[illegible]}\sin\text{[illegible]}\cos\text{[illegible]} - \cos\text{[illegible]}\sin\text{[illegible]}\right)\sum_{\text{[illegible]}}^{\text{[illegible]}}\text{[illegible]} \\ \sum_{\text{[illegible]}}^{\text{[illegible]}}\text{[illegible]}\cos\text{[illegible]}\cos\text{[illegible]} - \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]}/\text{[illegible]} \quad \text{[illegible]}/\text{[illegible]} \\ \text{[illegible]}/\text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]}/\text{[illegible]} \end{bmatrix} \tag{3.34}$$

By defining the control inputs as follows:

$$\begin{aligned} \text{[illegible]} &= \sum_{\text{[illegible]}}^{\text{[illegible]}}\text{[illegible]} \\ \text{[illegible]} &= \text{[illegible]} \\ \text{[illegible]} &= \text{[illegible]} \\ \text{[illegible]} &= \text{[illegible]} \end{aligned} \tag{3.35}$$

Then the state space representation will be as follows:

$$\begin{bmatrix} \text{[illegible]} \end{bmatrix} = \begin{bmatrix} \frac{1}{\text{[illegible]}}\left(\text{[illegible]}\sin\text{[illegible]}\cos\text{[illegible]} + \sin\text{[illegible]}\sin\text{[illegible]}\right).\text{[illegible]} \\ \frac{1}{\text{[illegible]}}\left(\sin\text{[illegible]}\sin\text{[illegible]}\cos\text{[illegible]} - \cos\text{[illegible]}\sin\text{[illegible]}\right).\text{[illegible]} \\ \frac{1}{\text{[illegible]}}\left(\cos\text{[illegible]}\cos\text{[illegible]}\right).\text{[illegible]} - \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]}/\text{[illegible]} \quad \text{[illegible]}/\text{[illegible]} \\ \text{[illegible]}/\text{[illegible]} \\ \text{[illegible]}/\text{[illegible]} \\ \text{[illegible]}/\text{[illegible]} \end{bmatrix} \tag{3.36}$$

CHAPTER 4

QUADROTOR CONTROL

4.1 CONTROL ARCHITECTURE

Mainly in this chapter, three control approaches will be represented: classical PID, LQR, and our proposed control algorithms. Figure 4-1 depicts the top down levels of the proposed control algorithms, which are essentially our contributions in this thesis.

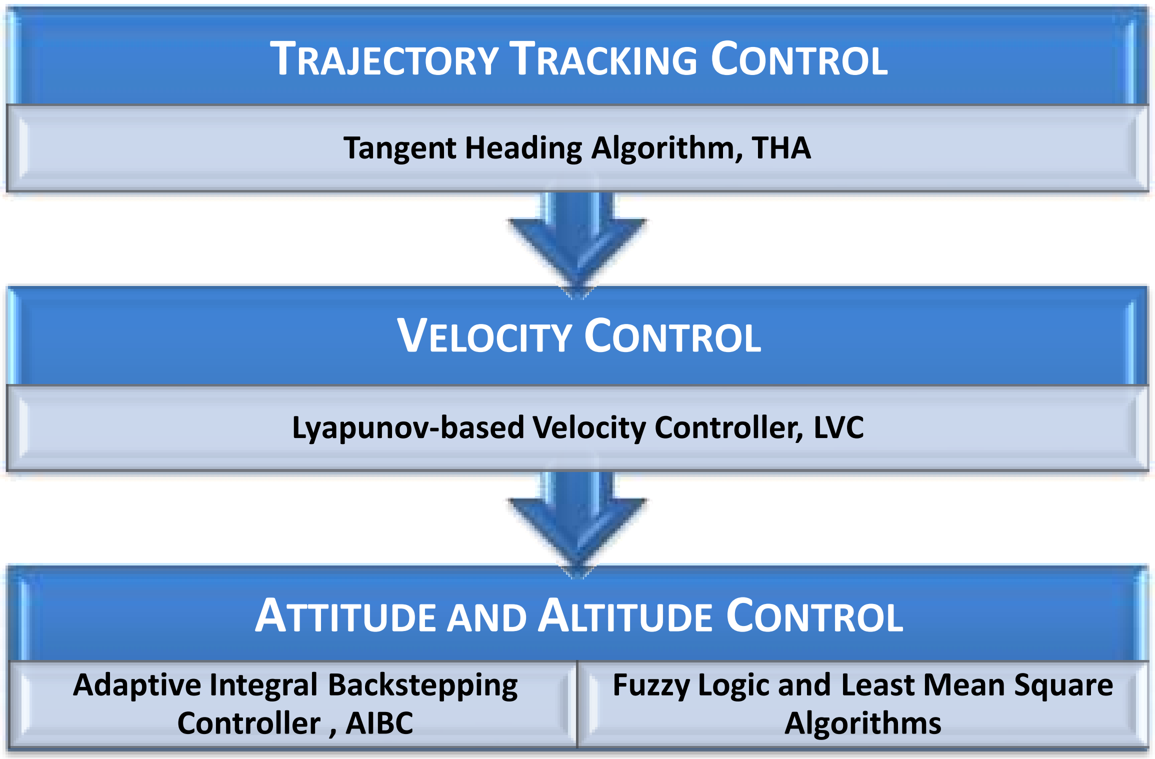

TRAJECTORY TRACKING CONTROL
Tangent Heading Algorithm, THA
VELOCITY CONTROL
Lyapunov-based Velocity Controller, LVC
ATTITUDE AND ALTITUDE CONTROL
Adaptive Integral Backstepping Controller , AIBC
Fuzzy Logic and Least Mean Square Algorithms

Figure 4-1: Control Architecture

4.2 PID CONTROL

4.2.1 Introduction

A proportional–integral–derivative controller (PID controller) is a generic loop feedback control that is widely used in industrial control systems. In process control today, more than 95% of the control loops are of PID type, most loops are actually PI control.

Nowadays, the microprocessor has had a dramatic influence on the PID controller. This has given the ability to provide additional features like automatic tuning, gain scheduling, and continuous adaptation. [22]

A general form of the PID controller is described by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}.$$

Where; y = the measured process variable,

r = the reference variable,

u = the control signal,

e = the control error ($e = y_{sp} - y$)

The controller parameters are proportional gain K_p, integral gain K_i, and derivative gain K_d. In linear systems, the process of controlling the system to follow certain specification is done using the rootlocus. But this is not the case in the nonlinear systems, where the system should be linearized first at certain points.

The PID controller is considered to be the most basic controller that is commonly used in different applications. Accordingly, a PID controller will be used to compare the performance of the quadrotor with respect to the proposed algorithm in this thesis.

The controller does not take the model into account and as such can only be seen as a preliminary controller. Hence, finding the PID gains is performed using one of the heuristic tuning methods, which is Ziegler-Nicholas technique.

4.2.2 Attitude and Altitude Control

The PID controller is designed for attitude and altitude stabilization. Thrust, roll, pitch, and yaw are controlled independently by using different PID controllers. To test and validate the 4-DOF motion, a test-stand is constructed in Figure 3-20 that will:

- Restrict the attitude and the altitude for a certain limits.
- Prevent the quadrotor from crashing due to unexpected malfunctions or bad tuning of the controller.
- Provide us with a set of data to obtain the controller gains using Ziegler Nicholas technique.
- Make the tuning steps much easier by performing online tuning of the PID parameters.

The control can be established on one degree of freedom at a time. The roll and pitch degrees require similar control gains because of the symmetry. After that, the yaw control can be implemented to prevent the quadrotor from rotating, and the altitude control will maintain the quadrotor on a desired elevation.

Figure 4-2 shows the feedback loops of the attitude and altitude PID controllers:

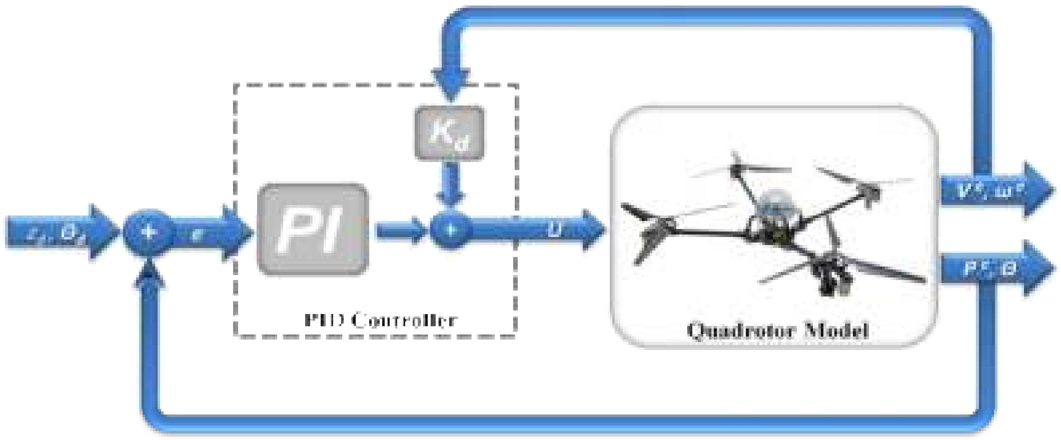


Figure 4-2: Block diagram of the Attitude and Altitude PID controllers.

The proportional and integral parts of the PID controller are obtained using the error feedback of each state with respect to a reference value [illegible] , [illegible] On the other hand, the derivative part is provided by the sensors directly and not from the error derivative, as shown in Figure 4-2. This is due to:

- The desired values are constant in attitude and altitude control. Therefore, their derivatives are equal to zero.
- The sampling processing time is set equal to the sensors sampling time.
- In the sensoring devices:
  - The states [illegible], [illegible] are extracted and filtered from the rates [illegible], [illegible] Therefore, the process of getting back the rates from the states will take more computational time and lagging behind.
  - Also, the error will diverge and get larger by going through non-necessary computational analysis (integration then derivation) rather than obtaining the reading of the rates directly from the sensors.

In this project, the PID controller is considered to be as the preliminary control algorithm for comparison purposes with other control algorithms. Tuning the PID gains is not an easy task for a nonlinear system such as the quadrotor system. The Ziegler-Nicholas method is a reliable technique that is commonly used for tuning the gains of the PID controllers. Next, the procedure of obtaining the gains will be described using the Ziegler-Nicholas method.

4.2.3 Ziegler-Nicholas Frequency Response Method

The Zigler-Nicholas method is based on a simple characterization of the system dynamics. The designed is based on knowledge of the point on the Nyquist curve of the system transfer function, where the Nyquist curve intersects the negative real axis. For historical reasons this point is characterized by the *ultimate gain* K_u, and the *ultimate period* T_u. the procedure of determining these parameters as follow:

- Using a closed loop PID controller, the gains are set so that the control action is proportional and ($K_i = K_d = 0$).
- The proportional gain will be increased slowly until the system starts to oscillate.
- The gain when the system is oscillating is the *ultimate gain* K_u and the period of oscillation is the *ultimate period* T_u.
- Table 4-1 shows simple formulas that are used to estimate the PID controller gains.

Table 4-1: PID controller obtained from the Ziegler-Nicholas frequency response method

Controller	K_p	K_i	K_d
P	0.5 K_u		
PI	0.4 K_u	0.5 (K_u / T_u)	
PID	0.6 K_u	1.2 (K_u / T_u)	0.075 $K_u T_u$

The Ziegler-Nicholas tuning method is designed to provide a good system response to the load disturbances. The design criterion was quarter the amplitude of the decay ratio, where the decay ratio is the ratio of the amplitudes of two successive oscillations. The decay ratio is related to the damping ratio according to this equation:

$$□ □ □^{□□□□/□\overline{□□□□}}$$

The decay ratio in the classical Ziegler-Nicholas method is equal to □ □ 1/4, and that gives □ □ 0.22 according to the equation (), which is a small damping for the quadrotor system. Since the primary design was to reduce the load disturbances, it is necessary to choose a set of gains that gives satisfactory results for our application.

Tyreus & Luyblen retuned the classical Ziegler-Nicholas method by introducing new rules for determining the PI and PID controller gains, which are called the TLC tuning rules. These values tend to reduce oscillatory effects and improve robustness. Table 4-2 shows the TLC retuning method:

Table 4-2: the Tyreus & Luyblen method

Controller	K_p	K_i	K_d
PI	0.3125 K_u	0.142 (K_u / T_u)	
PID	0.4545 K_u	0.206 (K_u / T_u)	0.075 $K_u T_u$

Next, a validation by simulation using matlab/simulink will show the response of the controller using Ziegler-Nicholas and TLC tuning methods. Consequently, the appropriate tuning method will be selected. Then the real implementation will be held to get the ultimate period and gain. In return, the gains of the PID controller will be specified.

From that, the PID controller will be the benchmark for comparing the various control algorithms that are used in this thesis.

4.3 LINEAR-QUADRATIC REGULATOR CONTROL, LQR

4.3.1 Introduction

The LQR control algorithm is an optimal control that is concerned with operating a dynamic system at minimum cost. The cost of the system that represented by a linear differential equations is described by a quadratic functional.

For a continuous-time linear system described by

$$\dot{x} = Ax + Bu \tag{4.1}$$

With a quadratic cost function defined as

$$J = \int_0^\infty \left(x^T Q x + u^T R u \right) dt \tag{4.2}$$

The feedback control law that minimizes the value of the cost is

$$u = -Kx \tag{4.3}$$

where K is given by

$$K = R^{-1} B^T P \tag{4.4}$$

And P is found by solving the continuous time differential Riccati equation.

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{4.5}$$

The LQR control can be applied as a trajectory follower to minimize the small errors between the reference state ref x_d and the measured state x, such that the applied control Input becomes:

$$u = -K(x - x_d) \tag{4.6}$$

Bryson's rule : matrices Q and R is given by the Bryson's rule. Where Q and R selected to be diagonal matrices:

$$Q_{ii} = \frac{1}{\text{maximum acceptable value of } x_i^2}, \quad i \in \{1,2,\dots,\ell\}$$

$$R_{jj} = \frac{1}{\text{maximum acceptable value of } u_j^2}, \quad j \in \{1,2,\dots,k\} \tag{4.7}$$

This corresponds to the following criteria

$$J = \int_0^\infty \left(\sum_{i=1}^{\ell} Q_{ii} x_i^2 + \rho \sum_{j=1}^{k} R_{jj} u_j^2 \right) dt \tag{4.8}$$

In LQR one seeks a controller that minimizes both energies. However, decreasing the energy of the controlled output will require a large control signal and a small control signal will lead to large controlled outputs. The role of the constant [illegible] is to establish a tradeoff between these conflicting goals:

1. If [illegible] is chosen to be very large, the most effective way to decrease the cost function is to use little control input, at the expense of a large controlled output.
2. When we chose [illegible] very small, the most effective way to decrease the cost function is to obtain a very small controlled output, even if this is achieved at the expense of a large controlled Input.

The Bryson's rule is just the starting point to a trial-and-error iterative design procedure aimed to obtain desirable properties for the closed-loop system.

4.3.2 LQR Attitude Control

The linearization version of the equations of motion in (3.36) of the rotational subsystem is:

$$\text{[illegible]}\ \text{[illegible]}/\text{[illegible]}$$

$$\text{[illegible]}\ \text{[illegible]}/\text{[illegible]}$$

$$\text{[illegible]}\ \text{[illegible]}/\text{[illegible]} \tag{4.9}$$

Thus, the state space representation of the following state variables is:

$$\text{[illegible]}$$

$$\text{[illegible]}$$

$$\begin{bmatrix} \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \end{bmatrix} \ \text{[illegible]} \ \begin{bmatrix} \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \end{bmatrix} \ \text{[illegible]} \ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \end{bmatrix} \ \text{[illegible]} \ \begin{bmatrix} 0 & 0 & 0 \\ 1/\text{[illegible]} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1/\text{[illegible]} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1/\text{[illegible]} \end{bmatrix} \begin{bmatrix} \text{[illegible]} \\ \text{[illegible]} \\ \text{[illegible]} \end{bmatrix} \tag{4.10}$$

Bryson's rule specifies the weight of the Q and R matrices. The gain matrix K is obtained by choosing the Q and R matrices, so that the system performance are chosen appropriately (e.g. <0.5sec settling time, and 10% overshoot). This is done by using the LQR Matlab toolbox.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \tag{4.11}$$

By choosing the Q and R matrices in (4.10), the gain matrix K is calculated using the Matlab toolbox as follows:

```
[K,S,E] = lqr(A,B,Q,R);
```

$$K = \begin{bmatrix} 10 & 2.4698 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 2.4698 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 2.3874 \end{bmatrix} \tag{4.12}$$

The substitution of the gain matrix K on the equation (4.6) leads us to conclude that the LQR controller is a PD controller. In addition, the integral part is added to the LQR control to reject the disturbances and the model uncertainties. Consequently, the integral LQR control was another choice to find the PID control gains.

The Integral LQR control algorithm will be tested and validated in chapter-5.

4.4 Adaptive Integral Backstepping Controller, AIBC

The main contribution in this thesis is achieved by proposing a new control approach using Adaptive Integral Backstepping Controller (AIBC) to control the AUS quadrotor. The recursive Lyapunov methodology in the backstepping technique will ensure the system stability, the integral action will increase the system robustness against disturbances and model uncertainties, and the adaptation law will estimate the modeling errors caused by assumptions in simplifying the complexity of the quadrotor model.

This methodology has been proposed to overcome these challenges:

1. Stabilize the attitude and the altitude of the quadrotor

2. Control the nonlinear model of the quadrotor by ensuring the stability based on Lyapunov criterion.
3. Use the adaptation scheme to estimate the translation matrix between the Euler angles rates and the inertial orientation angles rates.
4. Enhance the stabilization performance and compare it with the PID and LQR controllers.

Backstepping Control Algorithm:

The starting point is to analyze the algorithm on a second order system, which has the following form:

[illegible] (4.13)

In AUS quadrotor model, [illegible] is the state variable, [illegible] represents the body gyroscopic effect, [illegible] is constant and represents the inertia, [illegible] is the control input, and [illegible] is an estimator for modeling errors and variations.

The design starts with the definition of the position tracking error and its derivative as follows:

[illegible] (4.14)

[illegible] (4.15)

The subscripts in [illegible] (into [illegible]) is for differentiating between the derivative of the desired value [illegible], and the desired virtual input [illegible] as will be shown later.

Explicitly, this definition specifies our control objective, where the recursive methodology will systematically drive the tracking error to zero.

Now, let us consider the Lyapunov function [illegible], which is positive definite around the desired position:

[illegible] (4.16)

[illegible] (4.17)

If the velocity [illegible] was our control input, it would be straightforward to choose [illegible] to have exponential convergence for the system. One example is:

[illegible] (4.18)

Where [illegible] is positive number that determines the convergence speed of the error. Thus the derivative of the Lyapunov function will be semi-negative definite and consequently the error will converge exponentially to zero ([illegible] 0).

Since the velocity [illegible] is only a system variable and not a control input, we can't specify its value as we did in (4.17). However, we can still choose the desired behavior for [illegible] and consider it as our "virtual" control input. In the backstepping design, this desirable dynamic behavior is called the stabilizing function.

Furthermore, the integral action will be introduced by choosing the virtual input as follows:

[illegible] (4.19)

[illegible] (4.20)

The integral action in the backstepping design is to ensure the convergence of the tracking error to zero at the steady state, despite the presence of the disturbances and model uncertainties.

Since, [illegible] is not our control input, there exists a dynamic error between it and its desired behavior [illegible]. Therefore, we will compensate this dynamic error through our next design step by defining the velocity tracking error and its derivative as follows:

[illegible] (4.21)

[illegible] (4.22)

We can rewrite the derivative of the position error:

[illegible] (4.23)

Therefore,

[illegible] (4.24)

The Integral Action:

By adding the integral action, the augmented Lyapunov function will be:

[illegible] (4.25)

The derivative will satisfy [illegible] 0[illegible], when the control input is:

$$[illegible] \qquad (4.26)$$

Adaptation Scheme:

Since we do not know the real value of [illegible], we will replace it with the estimated value [illegible] and the adaptation law will be designed:

$$[illegible] \qquad (4.27)$$

The design is not complete since we still need to derive the update law for the estimated parameter [illegible]. For this purpose, we will define the parameter estimation error signal in (4.27) as follows:

$$[illegible] \qquad (4.28)$$

After this definition, by substituting (4.27) into (4.24) the derivative of the velocity tracking error is:

$$[illegible] \qquad (4.29)$$

The Lyapunov function is used a second time to augment the estimated parameter error:

$$[illegible] \qquad (4.30)$$

$$[illegible] \qquad (4.31)$$

$$[illegible] \qquad (4.32)$$

Where [illegible] is a positive design constant that determines the convergence speed of the estimate.

In order to render the non-positivity of the Lyapunov derivative in the above equation, we will choose the adaptation law for the estimated parameter [illegible] as:

$$[illegible] \qquad (4.33)$$

$$[illegible] \qquad (4.34)$$

Where;

$$[illegible] \tag{4.35}$$

Thus,

$$[illegible] 0 \tag{4.36}$$

Finally, the control input that contains the adaptation law and the integral action is derived as follows:

$$[illegible] \tag{4.37}$$

By the definition of Lyapunov function and its non-positive derivative, the position tracking error [illegible], the velocity tracking error [illegible], the integral action [illegible], and the estimated parameter error [illegible] are bounded signals. Thus, we have concluded the boundedness of all the internal signals in the closed-loop control system. Thus the derivatives of the error signals, [illegible] and [illegible], are bounded as well.

By applying the AIB algorithm on each equation of motion of the AUS quadrotor, the control inputs are going to be:

$$[illegible]$$

$$[illegible]$$

$$[illegible]$$

$$[illegible] \tag{4.38}$$

4.5 Lyapunov-based Velocity Controller, LVC

The velocity of the quadrotor is controlled based on the Lyapunov criterion to approve the system stability. The controller is commanding the desired angles of the AIBC, where the LVC is cascaded with the AIBC to do the velocity control.

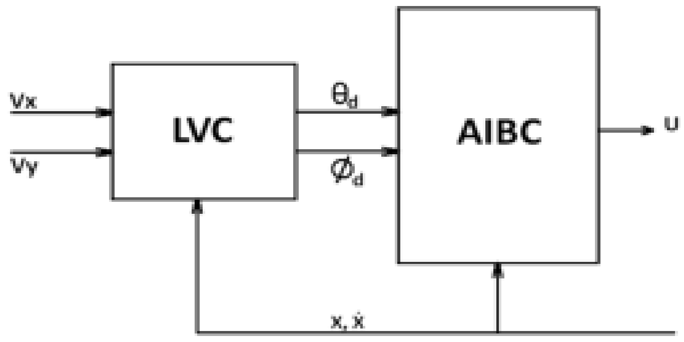


Figure 4-3: Cascading of LVC and AIBC

This methodology has been proposed to overcome these challenges:

a. Apply the hover flight on the quadrotor by cascading the LVC with the AIBC.
b. Compare the proposed algorithms in the hover with other control algorithm e.g. sliding-mode, PID an LQR controllers, where they suffered from the drift and the fluctuation in previous works.
c. Extend the work on the hover flight to test the trajectory tracking flight, where the proposed algorithms represent the nonlinear model efficiently.

Control Algorithm:

The translation subsystem shows dependency on rotation. So, by coupling the two systems, we will control the remaining states based on the Lyapunov approach.

Assume the position tracking error for u as:

[illegible] (4.39)

The Lyapunov function and its derivative are:

[illegible] (4.40)

[illegible] (4.41)

The derivative will be semi-negative definite ([illegible]) if:

[illegible] (4.42)

But,

[illegible] (4.43)

The pitch angle [illegible] is the desired input to the AIBC:

[illegible] (4.44)

Similarly, for the lateral direction:

[illegible] (4.45)

For this Lyapunov based Velocity Controller (LVC) the Lyapunov approach will ensure the stability as well as the boundedness of the errors and their derivatives, and as a result the control inputs will be bounded as well.

4.6 Fuzzy Logic and Least Mean Square Techniques

The FL technique utilizes the advantage of fuzzy logic to schedule the backstepping controller parameters. This resulted in improved performance compared with that achieved by backstepping controller alone. On the other hand, the LMS technique is used to minimize the control efforts by finding the parameters of backstepping controller which provide minimum control efforts to the system.

Those methodologies have been proposed to achieve those goals:

a. Improve the system performance of our control algorithms by adjusting the backstepping controller parameters.
b. Distribute the control efforts by using FL to avoid the saturation of the control efforts.
c. Minimize the control efforts by using LMS to detect the minimum efforts that could be able to stabilize the quadrotor.

Backstepping Least Mean Square, BLMS Controller:

The purpose of this controller is to minimize the input efforts to the system by minimizing the values of the control inputs U_x, using the least mean squares algorithm.

First, the square of the roll control input is obtained as follows:

[illegible]

[illegible]

[illegible] (4.46)

According to this algorithm, the squared control input is differentiated with respect to the parameters [illegible]& [illegible] of Backstepping controller.

Therefore,

[illegible] 2 [illegible] 0

[illegible] 2 [illegible] 0

(4.47)

And,

[illegible] (4.48)

By controlling [illegible]& [illegible] , the values of the backstepping parameters will give the minimum control efforts.

Backstepping Fuzzy Logic, BFL Controller:

BFL controller aims to tune the parameters backstepping parameters to improve the performance of the system. It behaves similar to a gain scheduling controller that results in better performance.

The linearization around different angles leads to different linear systems. Depending on operating point (angle and angle rate), the behavior of the system, in terms of response time and control efforts, is affected. e.g. for the pitch angle ranged between (-15^o, 15^o) the parameter of the linearized system is obtained each 5 degree using the Matlab linearization toolbox. Therefore, the controlling gains should be adjusted to give better response at each instant.

In addition, the BFL controller should assign reasonable values to the backstepping parameters, in order to compensate for the limitation in the supplied power drawn by the quadrotor. Therefore, the optimal parameter values are constrained by desired time response and the available control effort. Since, the scheduled parameters are tuned as function of the attitude by the BFL controller; the stability of the overall system will be significantly enhanced.

BFL controller consists of three fuzzy based rules for pitch, roll and yaw, respectively. Each attitude controller has two inputs (angle and angle rate) and two outputs to specify the backstepping parameters $\square_{\square}$.

Next, the pitch control using fuzzy logic theory will be demonstrated, while the roll and yaw control will be tackled in similar procedures as that of the pitch control.

Pitch Control in BFL controller:

The pitch angle $\square$, and the pitch angle rate $\square$,$^{\square}$are the inputs to the BFL controller and the outputs are $\square_{\square}$ and $\square_{\square}$. Fuzzy-logic controller system consist of three processes:

Fuzzification is a process of transforming crisp values into grades of membership for linguistic terms of fuzzy sets. The membership function is used to associate a grade to each linguistic term. Here we will assign three membership functions for each input and output, where nine rules are enough to perform relatively smooth change in $\square_{\square}$.

Following the fuzzyfication stage, $\square$, $\square^{\square}\square\square\square$ $\square_{\square}$, $\square_{\square}$ are represented each with three equally spaced membership functions as shown in Figure 4-4.

The next stage is the *Fuzzy inference* system which is the process of formulating the mapping from a given input to an output using fuzzy logic. Mamdani's fuzzy inference method (Minimum of Maximum (MOM)) is used to map the inputs to the outputs.

Nine rules are needed to complete this mapping. Finally, the *Defuzzification* phase is applied to get a crisp output using Center of Area technique COA.

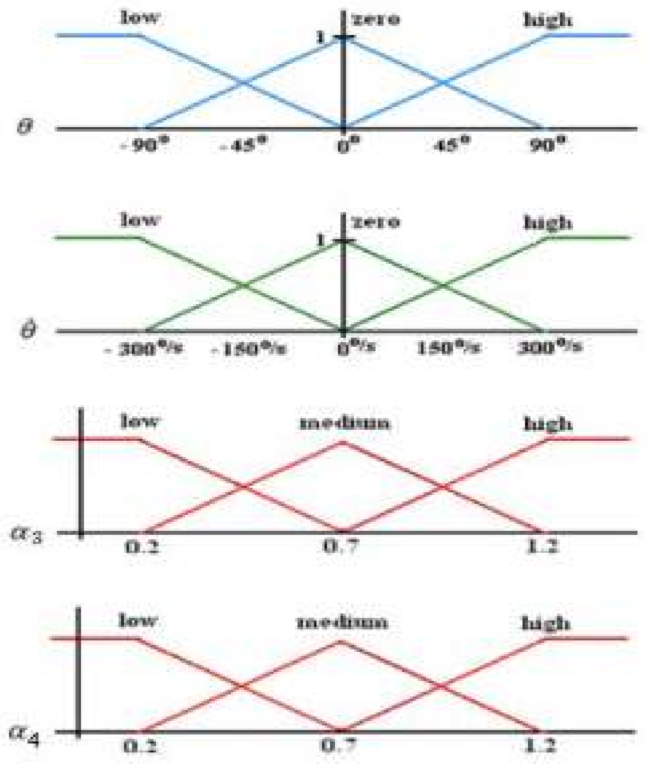
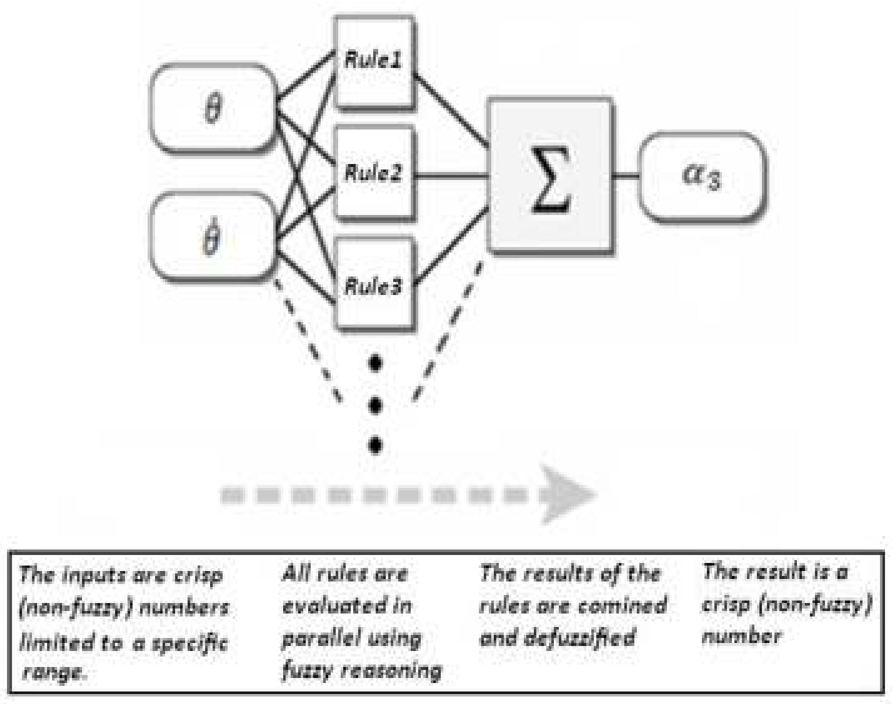


Figure 4-4: Fuzzy logic pitch membership functions

The whole process of the fuzzy logic system is shown in the following Figure 4-5.



The inputs are crisp (non-fuzzy) numbers limited to a specific range.

All rules are evaluated in parallel using fuzzy reasoning

The results of the rules are comined and defuzzified

The result is a crisp (non-fuzzy) number

Figure 4-5: Fuzzification, inferring & defuzzification Processes

4.7 Tangent Heading Algorithm, THA

The purpose of proposing an algorithm for trajectory tracking is to simulate the proposed controllers in hover and non-hover flights.

Trajectory Tracking Algorithm

After generating the path using any method (e.g. b-spline method (Appendix C.2)) for a given waypoints, we will move to follow this path using Tangent- Heading Algorithm:

a. The curve will represented many points to describe the desired path.
b. This algorithm keeps the quadrotor heading with the tangent of the curve in each point.

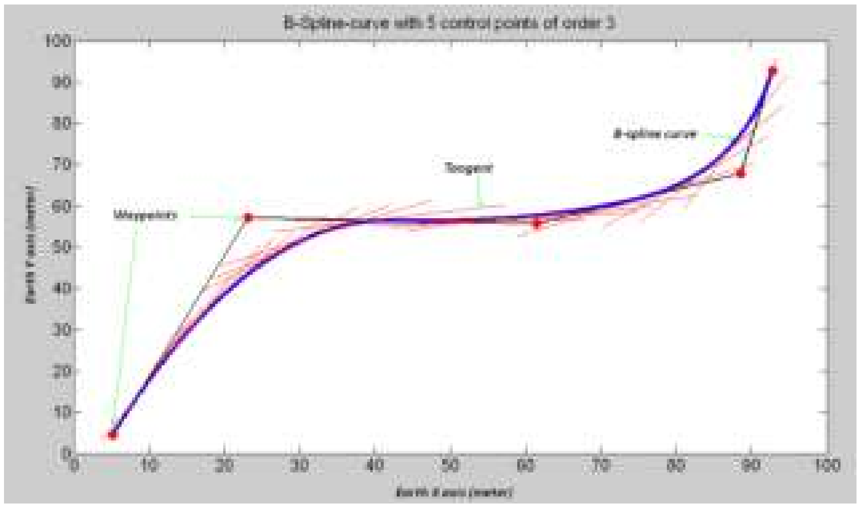


Figure 4-6: Description of the THA path

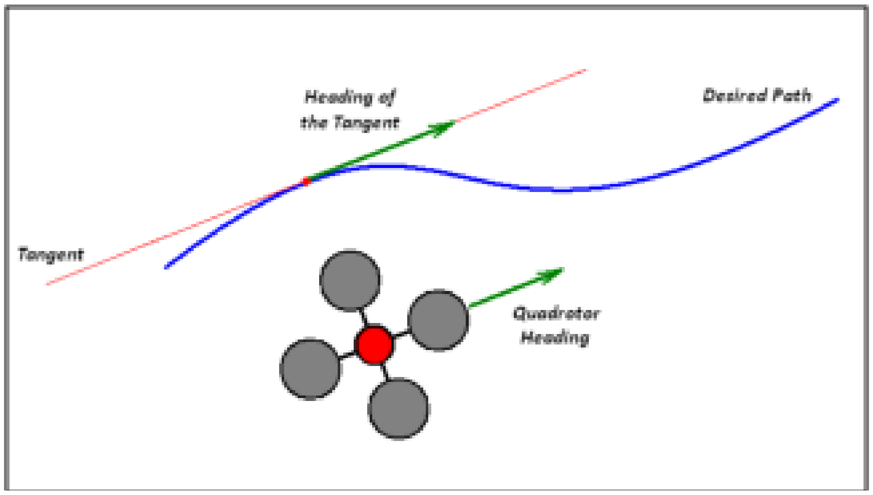


Figure 4-7: The Quadrotor and the Tangent Headings

c. Because the heading depends on the tangent of the path. We should find a way to keep track the position with respect to the curve by finding the nearest distance between the curve and the current position of the quadrotor. It's very hard to check the distance between the current position and all the points that represent the path. Therefore, we will check the minimum distance of ten consequent future points, this distance will be the perpendicular line to the tangent of the point that has the minimum distance.

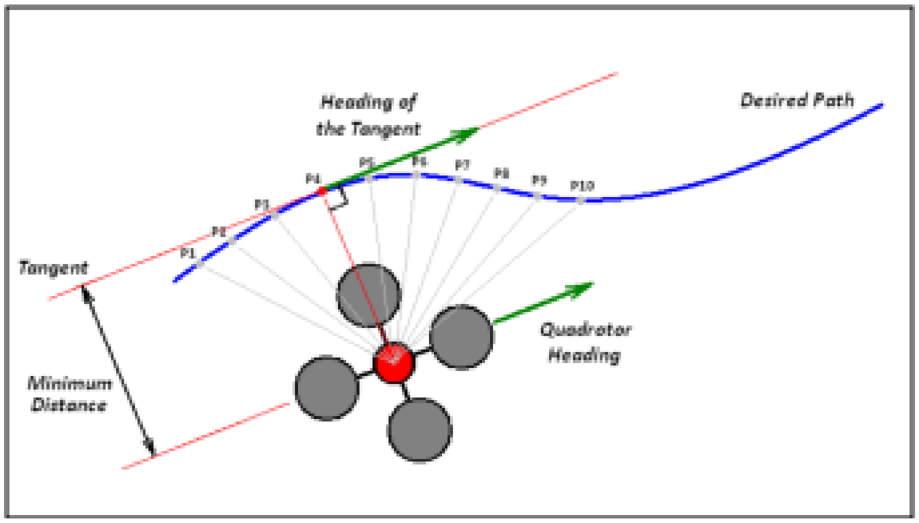


Figure 4-8: Minimum Distance between the Quadrotor and the Path

This figure shows the nearest point to the quadrotor. Where, the line between the current position and P4 is perpendicular to the tangent at P4. And the quadrotor heading parallel to the tangent. Next, P4 will be the reference point for another ten future points on the next calculation.

***Algorithm 1*:** Nearest_Distance_to_the_Quadrotor (reference_point , x_Q, y_Q , x_i, y_i)

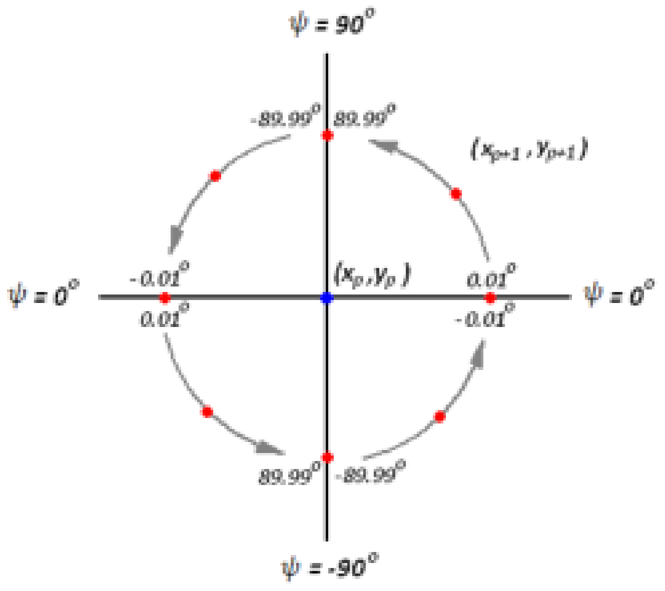
```
[illegible]
[illegible]
[illegible]
[illegible]
    [illegible]
    [illegible]
        [illegible]
        [illegible]
    [illegible]
[illegible]
[illegible]
```

d. To find the tangent of the point P_i on the curve, we will find the slope between this point P_i and the next point P_{i+1} . this assumption is valid because:

   a. The curve will be described by many points. So, the distance between two consecutive points is small. Therefore, the slope almost represents the tangent for that point P_i, and exactly represents the tangent of the point $P_{i+1/2}$.

   b. The tangent of point $P_{i+1/2}$ will predict the future heading, where the system needs time to change the heading, and this assumption will give smoothness to follow the desired path.

e. To find the right heading, while tracking a trajectory, a specific coordinate system should be defined to deal with the specified heading, as mentioned in point (4). Figure 4-9 shows all the possibilities of the heading angle □ on the earth frame coordinate system. To define the full round (2π) that start and end on a certain axis without any jumping, we should transform the coordinate system as in Figure (-b). Here we are using the (0^o or 360^o) axis as our reference axis. The *If condition functions* in this figure will make this transformation is possible.



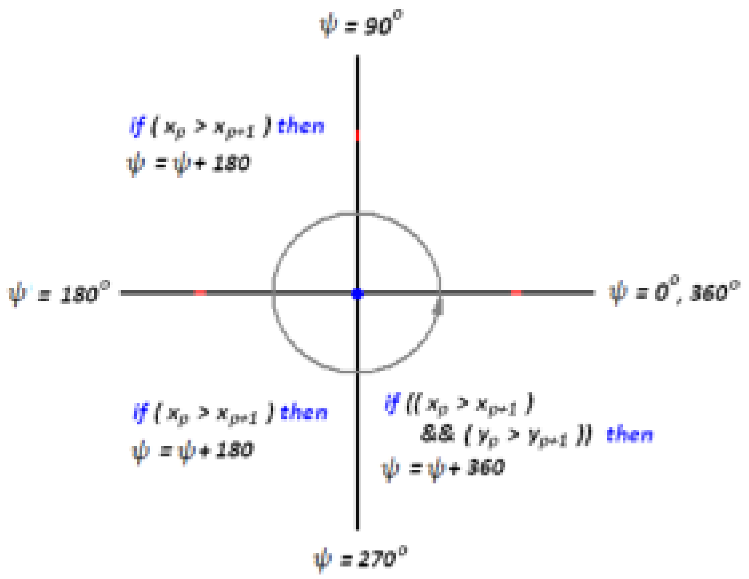


Figure 4-9: Coordinates Transformation

f. After referring the desired heading to the new coordinate system (0 to 2π), we should refer the current heading of the quadrotor to that coordinate too. Because the current heading is cumulative and it's possible to be more than 2π if the quadrotor turns more than full round counterclockwise, or less than 0 if it's rotating clockwise.

The second algorithm shows how to refer the current heading to the desired coordinate system.

Algorithm 2**:** Current_Heading_Transformation ([illegible])

```
[illegible]
[illegible]

  [illegible]
    [illegible]
  [illegible]
    [illegible]
  [illegible]

[illegible]

[illegible]
```

g. We will follow new strategy to avoid the sharp turning of the heading, if the difference between the new and the current headings is greater than π:

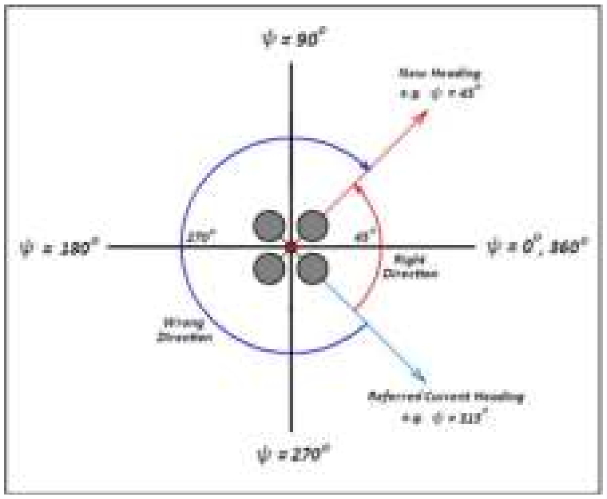


Figure 4-10: Sharp turning Avoiding Strategy

Figure 4-10 shows that the quadrotor has two possibilities to turn to the new heading. The difference between the new heading and the referred current heading is greater than π. So we should follow the second way to do the right turning.

This situation has many possibilities, where the difference between the two headings could be greater or less than zero, also the right direction could cross the reference axis, and maybe we will have many problems that could be faced to find the right direction and difference between the two headings.

To solve these problems we will follow a strategy which includes all the possibilities, as follows:

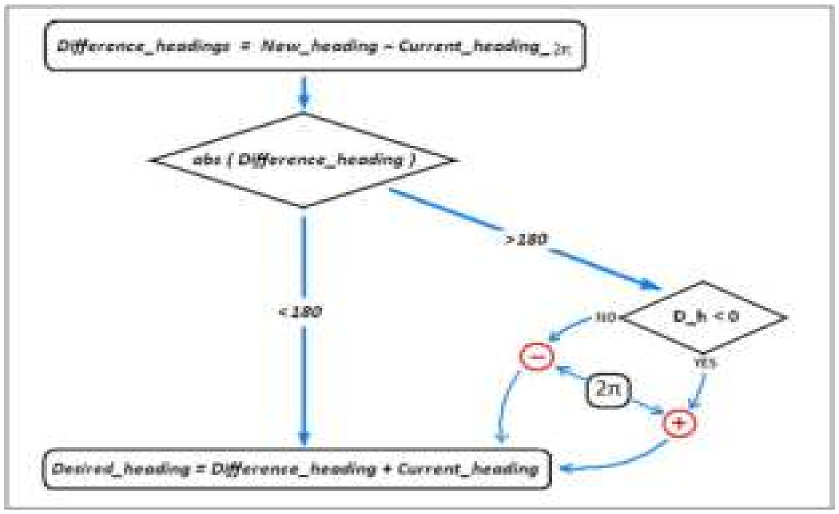

Difference_headings = New_heading – Current_heading_2π
abs (Difference_heading)
>180
<180
D_h < 0
NO
YES
−
2π
+
Desired_heading = Difference_heading + Current_heading

Figure 4-11: Desired Heading Measurement Flowchart

This flowchart shows the process of getting the desired heading to be our current heading for the next iteration.

The new heading from Figure 4-11 and the referred current heading from algorithm (2) are between (0^o or 360^o) as we did before in the previous steps. This operation will make the difference less than 360^o, but not necessarily less than 180^o. In this case our strategy will find the right difference that should be accumulated to the current heading to give the right desired heading.

The example in Figure 4-10 shows that the new heading is 45^o and the referred current heading is 315^o (e.g. the current heading is -45^o before the referring process). According to our algorithm, the absolute of the difference is 270^o and this value is greater than 180^o. But the difference is negative ($-270\ ^o$) so; we will add 360^o to the difference and the new difference will be equal to (90^o). Finally the desired heading will be equal to the summation of the new difference and the current heading, which is (45^o).

This algorithm is valid for all the possibilities of the current and new headings.

CHAPTER 5

TESTS AND VALIDATION

5.1 CONTROL STRATEGY

This chapter aims to simulate and implement the proposed nonlinear control algorithms. Furthermore, the PID and the Integral-LQR controllers are simulated and implemented to be the benchmark for the nonlinear control algorithms.

The tests procedure is passing through three consecutive steps:

- *Software Simulation:* the Matlab/Simulink is the simulator of the AUS quadrotor model. The control algorithms (from chapter 4) are tested and validated based on the represented nonlinear model in chapter 3.
- *Hardware-In-The-Loop-Simulation (HILS):* is the intermediate stage between the software simulation and the real time implementation, to test and validate the control algorithms on a microcontroller before the real tests.
- *Real Time Implementation:* the main objective of this thesis is to build and test the AUS quadrotor helicopter. In this chapter, we are going to test the quadrotor to perform the attitude and altitude stabilization, then the indoor and the outdoor hover control.

Next, for each step we will elaborate more by showing extensive simulation and flight test results.

5.2 SOFTWARE SIMULATION

5.2.1 Tuning the PID Controllers

Validating the controller performance is necessary before the real test implementation. For that reason, the nonlinear model from chapter-3 is simulated and controlled by using the Matlab/Simulink software. Figure 5-1 represents the Simulink block diagram of the quadrotor control system.

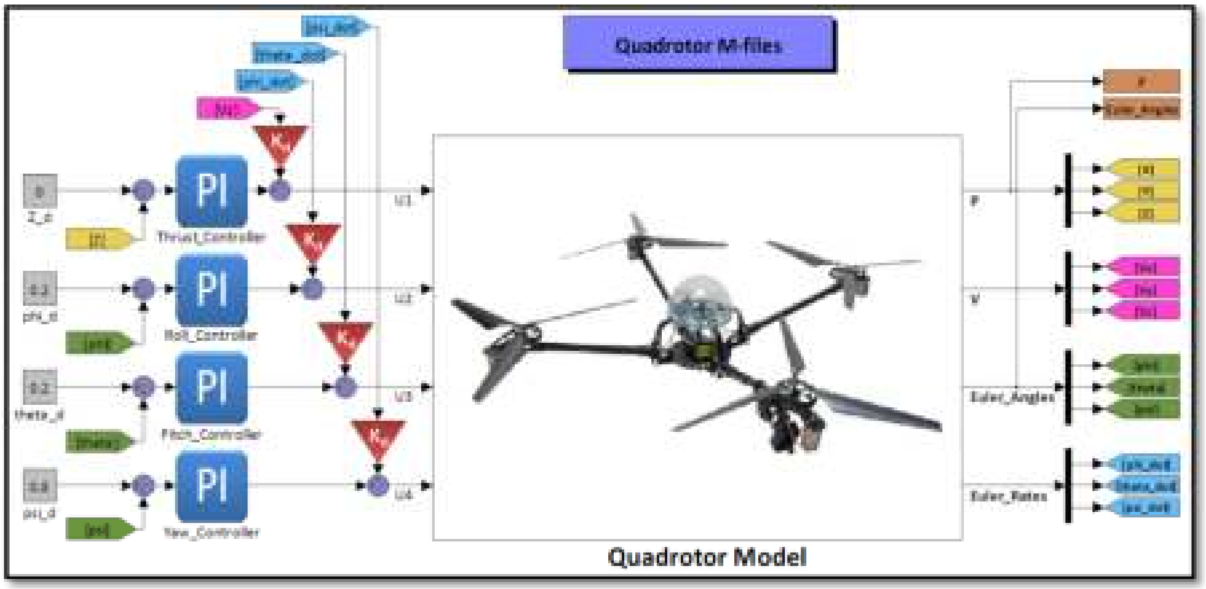


Figure 5-1: Block diagram of the quadrotor control system.

Mainly, the Simulink model is consisting of: the quadrotor model block and the controller’s blocks. The inputs to the controller’s blocks are the attitude and the altitude error signals, where:

□□□ □□□□□ □□□□□ □ □□□ □□□□□□□ □□□□□ □ □□□ □□□□□□□□□ □□□□□

On the one hand, the desired inputs are provided by the user or another controller (cascading of multi-stage controllers). On the other hand, the feedback signals are provided by the output states of the quadrotor model block. According to the error signal, the controller will provide the quadrotor model by an appropriate control inputs (efforts).

Using Ziegler-Nicholas and TLC tuning methods, the PID controller’s gains are tuned, and different simulation results are executed using Simulink Matlab toolbox. By comparing the different tuning method, the better that gives higher performance is selected.

5.2.1.1 Ziegler-Nicholas Simulation Results

From the closed loop system, the ultimate gain and the ultimate period are obtained for each state (roll, pitch, yaw and thrust). The symmetry around the z-axis serves the user to tune the pitch and roll controllers with the same parameters.

The ultimate gains and periods are obtained in Figure 5-2 for attitude and altitude controllers.

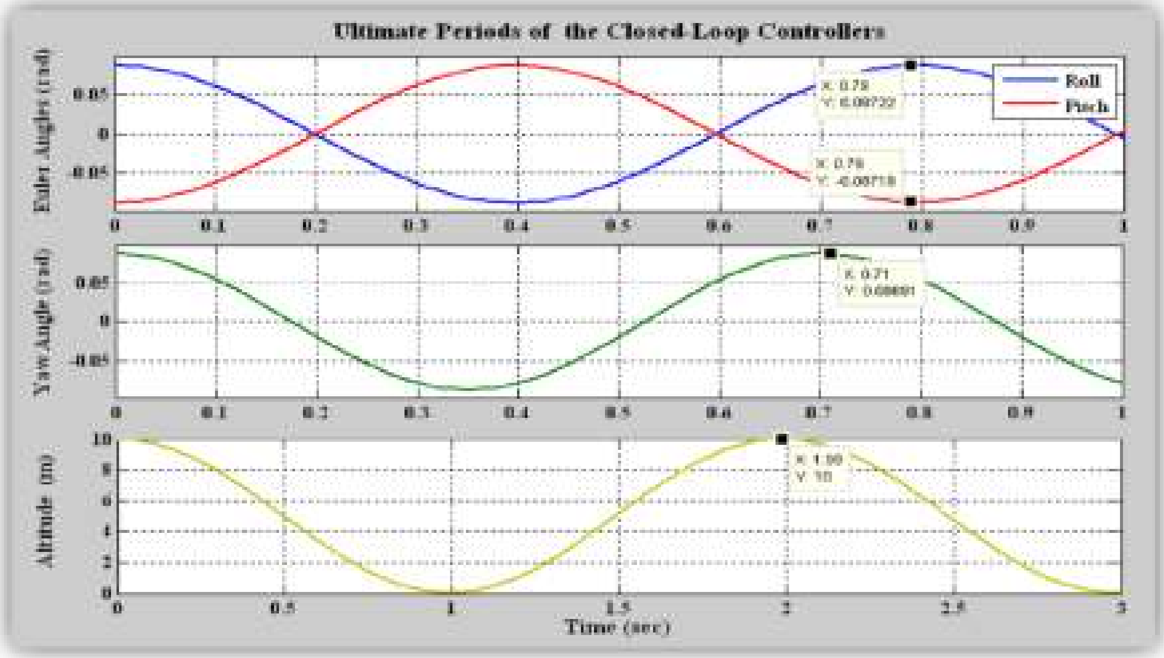


Figure 5-2: Simulation results to obtain the Ultimate periods

The system will star to oscillate by providing enough proportional gain (ultimate gain). Then the period of oscillation is measured for each variable state. The system in our simulations is disturbed around the origin for attitude control and around 5m for altitude control. The ultimate gains and periods for each controller are obtained and represented in Table 5-1, and the PID control gains are calculated as well.

Table 5-1: PID Controllers parameters for attitude and altitude control

Controller Parameters	*Roll/Pitch Controller*	*Yaw Controller*	*Altitude Controller*
K_u	22	12	20
T_u	0.79	0.71	1.99
K_p	13.2	7.2	12
K_i	33.42	20.28	12.06
K_d	1.304	0.639	2.985

The system responses of the PID controllers are shown in Figure 5-3.

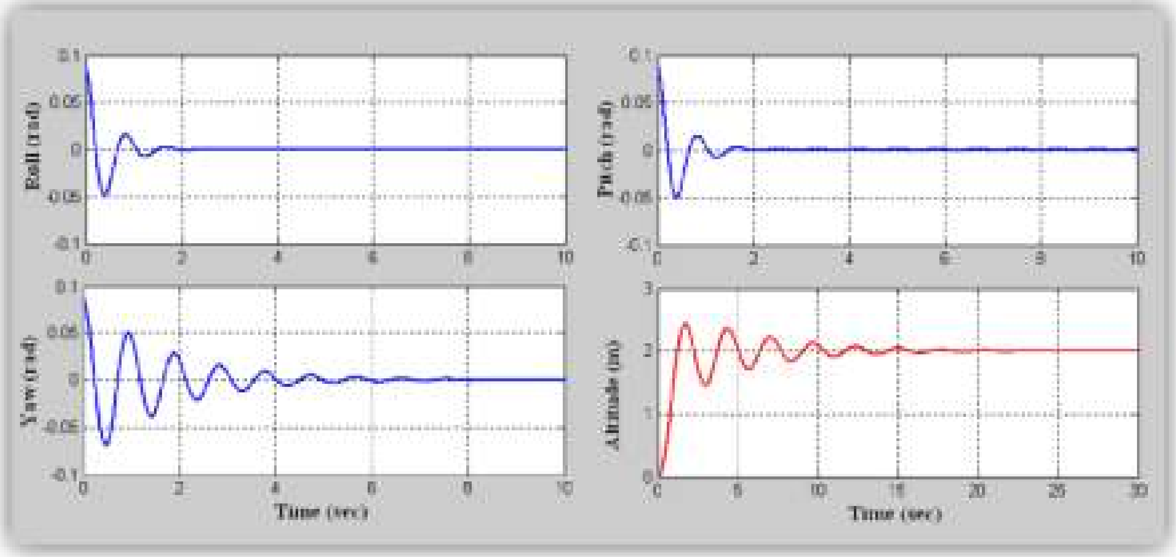


Figure 5-3: System responses using Ziegler-Nichols tuning method

As mentioned in section (4.4), the Ziegler-Nicholas tuning method is designed to provide a good system response to the load disturbances. But it shows oscillatory responses. To reduce oscillatory effects and improve robustness, the TLC retuning method will be simulated.

5.2.1.2 TLC Simulation Results

In Table 5-2, the same procedure of the Ziegler-Nichols method is followed to calculate the PID controller gains.

Table 5-2: Controllers parameters for attitude and altitude control

Controller Parameters	*Roll/Pitch Controller*	*Yaw Controller*	*Altitude Controller*
K_u	22	12	20
T_u	0.79	0.71	1.99
K_p	9.999	5.454	9.090
K_i	5.737	3.482	2.070
K_d	1.304	0.639	2.985

Figure 5-4 shows the system responses of the tuned PID controllers.

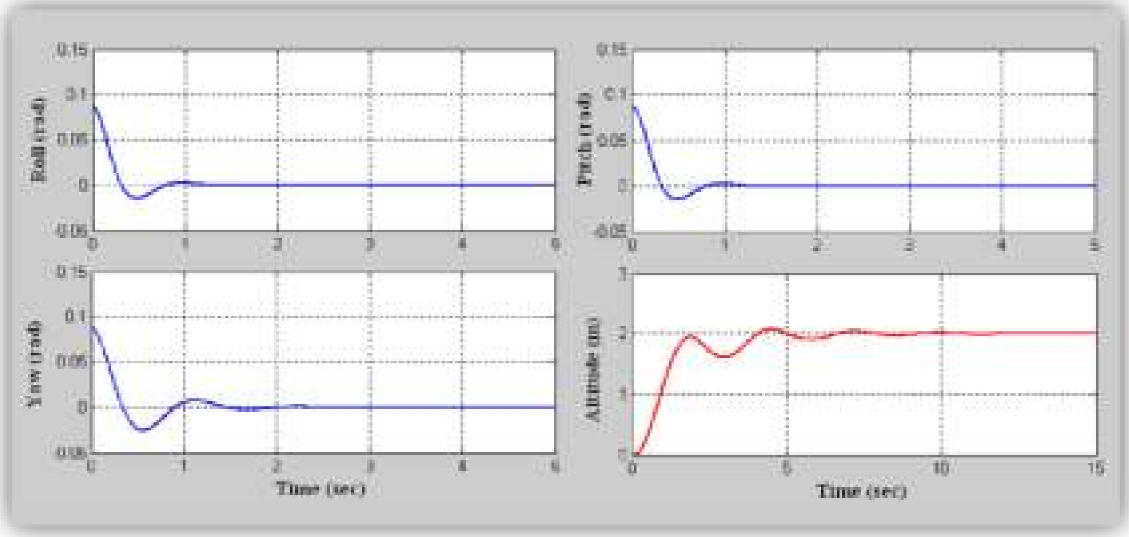


Figure 5-4: System responses using TLC retuning method

As a result, the effect of the oscillatory response has been removed and the system behaves with faster time response.

5.2.1.3 *Integral LQR Controller*

As mentioned in section 4.3.2, the integral LQR controller is performing as a PID controller. The simulation result is depicted on the following Figure 5-5, where the system response is relaxed compared to the TLC tuning method. The reason behind this is the optimization of the control inputs at the expense of the system response as mention in section 4.3.1.

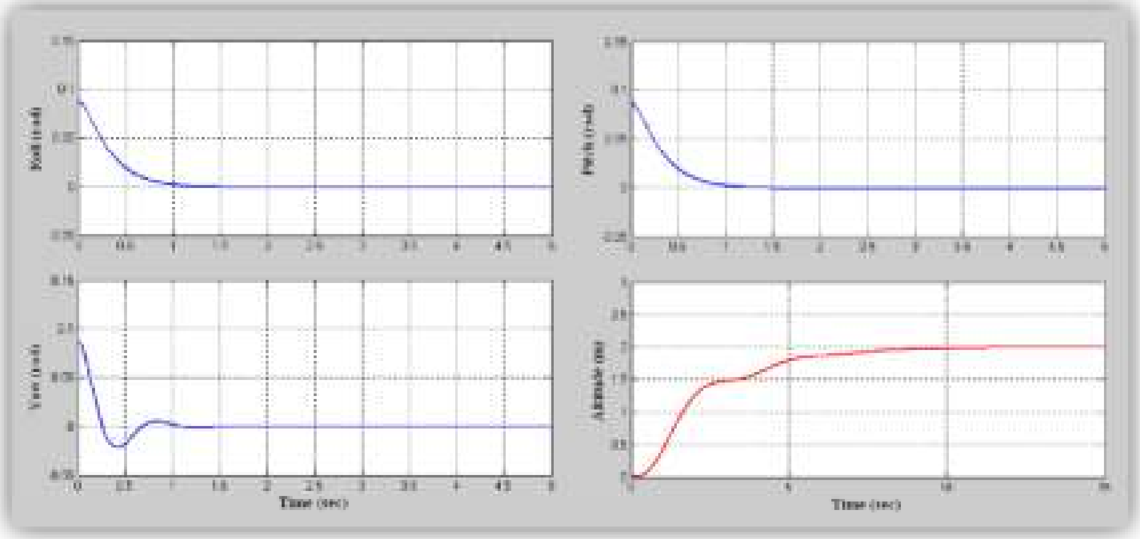


Figure 5-5: System responses using LQR Control method

5.2.2 Nonlinear Control Algorithms

5.2.2.1 Control Architecture

The block diagram in Figure 5-5 depicts the nonlinear control algorithms that are used in this thesis.

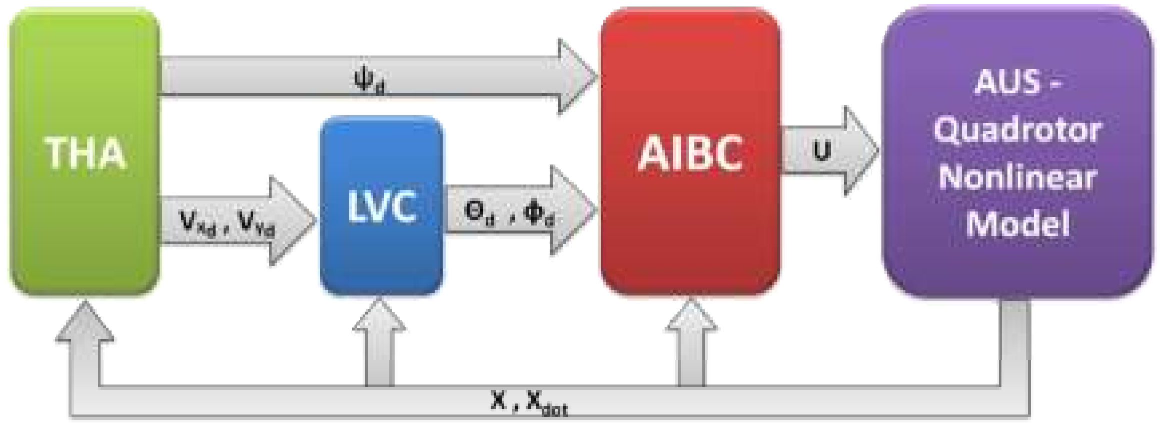


Figure 5-6: Nonlinear Control Architecture

According to the defined path in the THA, the algorithm generates desired velocities (longitudinal and lateral velocities) that are going to be the desired input to the LVC. This in turn produces the desired angles to the AIBC.

The AIBC controller acquires the desired angles from the LVC and THA, to produce the required control efforts to the quadrotor model. More details about the algorithms of each block were represented on chapter 4.

Based on Matlab/Simulink environment, the control architecture is depicted on Figure 5-6. The desired path is generated based on the b-spilne curves of pre-defined way points. More details about generating the path based on the b-spline method are illustrated in Appendix C.2.

The top-down level in Figure 4-1 elucidates the job of each control algorithm. From the top level, the THA is the trajectory tracking algorithm, the LVC is proposed to control the velocities in the linear motion, and the AIBC is stabilizing the attitude and the altitude of the AUS quadrotor. To validate the algorithms, extensive simulation results are represented and comparisons with other work are done. The results are encouraging and promising.

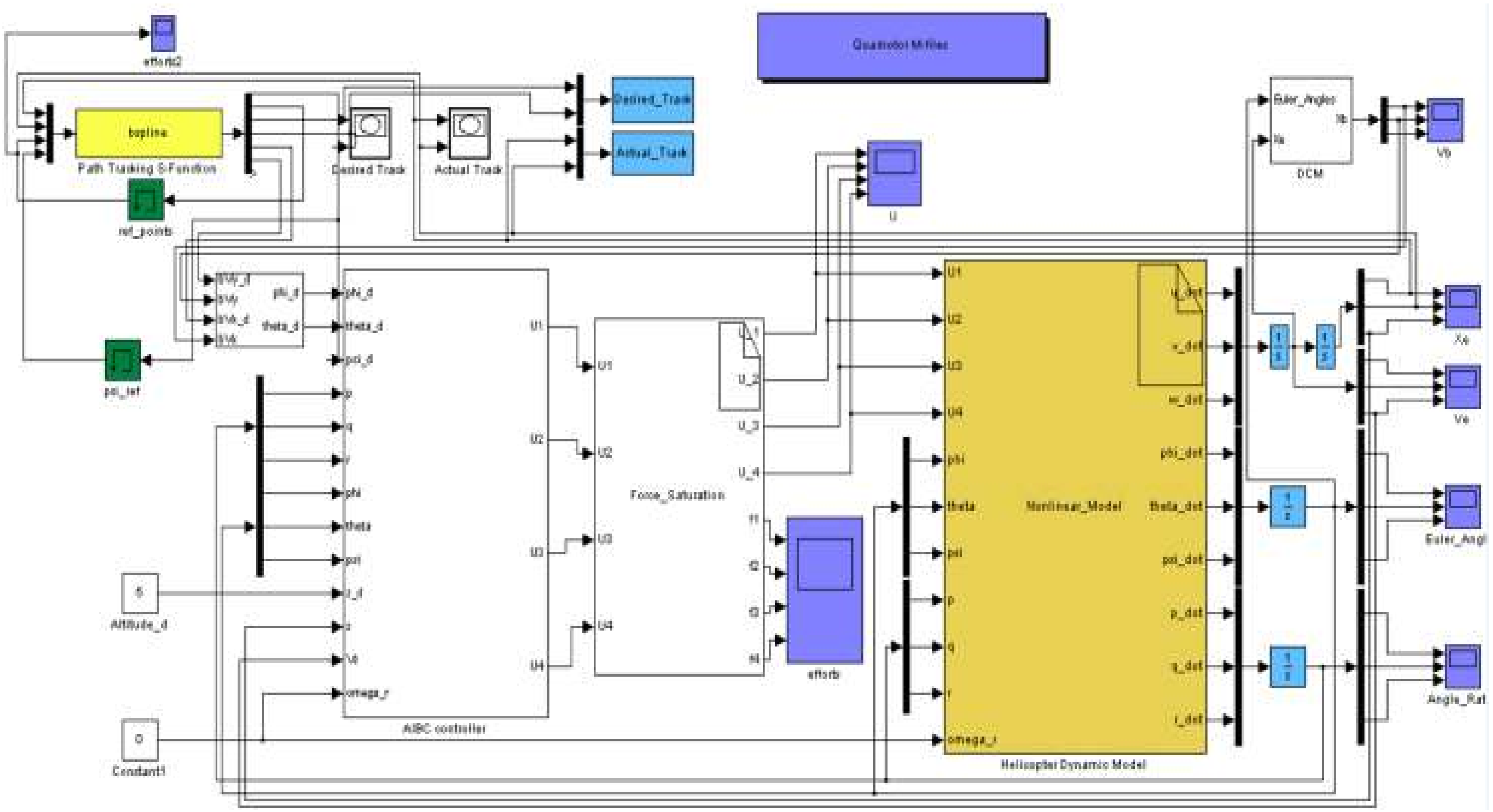


Figure 5-7: Matlab/Simulink Simulator AUS Quadrotor Controller

5.2.2.2 Adaptive Integral Backstepping Controller, AIBC

Equations 4-26 express the relationship between the variable states and the control inputs. The control inputs (*U*$_2$*, U*$_3$ and *U*$_4$) are controlling the attitude of the quadrotor, while *U*$_1$ is for altitude stabilization. The input\output of the AIBC block is clearly shown in Figure 5-6.

The AIBC parameters are defined based on the PID controller gains. This process is done by removing the nonlinear terms in the equations of motion 4-26, then equating the result with the PID controller terms. The relationship is described in the following steps:

$$U_2 = I_{xx}\left[(1 - c_1^2 + \lambda_1)e_1 + (c_1 + c_2)e_2 - c_1\lambda_1\chi_1 + \gamma_1\chi_2 + \ddot{x}_{1d} - (\frac{I_{yy} - I_{zz}}{I_{xx}})\, x_4 x_6\right]$$

The linearized version of the control efforts is:

[illegible]

[illegible]

[illegible] (5.1)

The equivalent PID controller can be expressed as following:

[illegible] (5.2)

Therefore;

[illegible]

[illegible]

[illegible] (5.3)

The equations in (5.3) describe the indirect relationship between the PID gains and the AIBC parameters of the linearized control efforts. Based on this relationship, the comparison between the PID and AIBC will be applied.

Due to the fast response of the controller, the duration time of the simulation is set to be 5sec (2sec for attitude is viewed). The initial conditions are 45^o for the Euler angles, and 2m step input for the elevation, as shown in Figure 5-7:

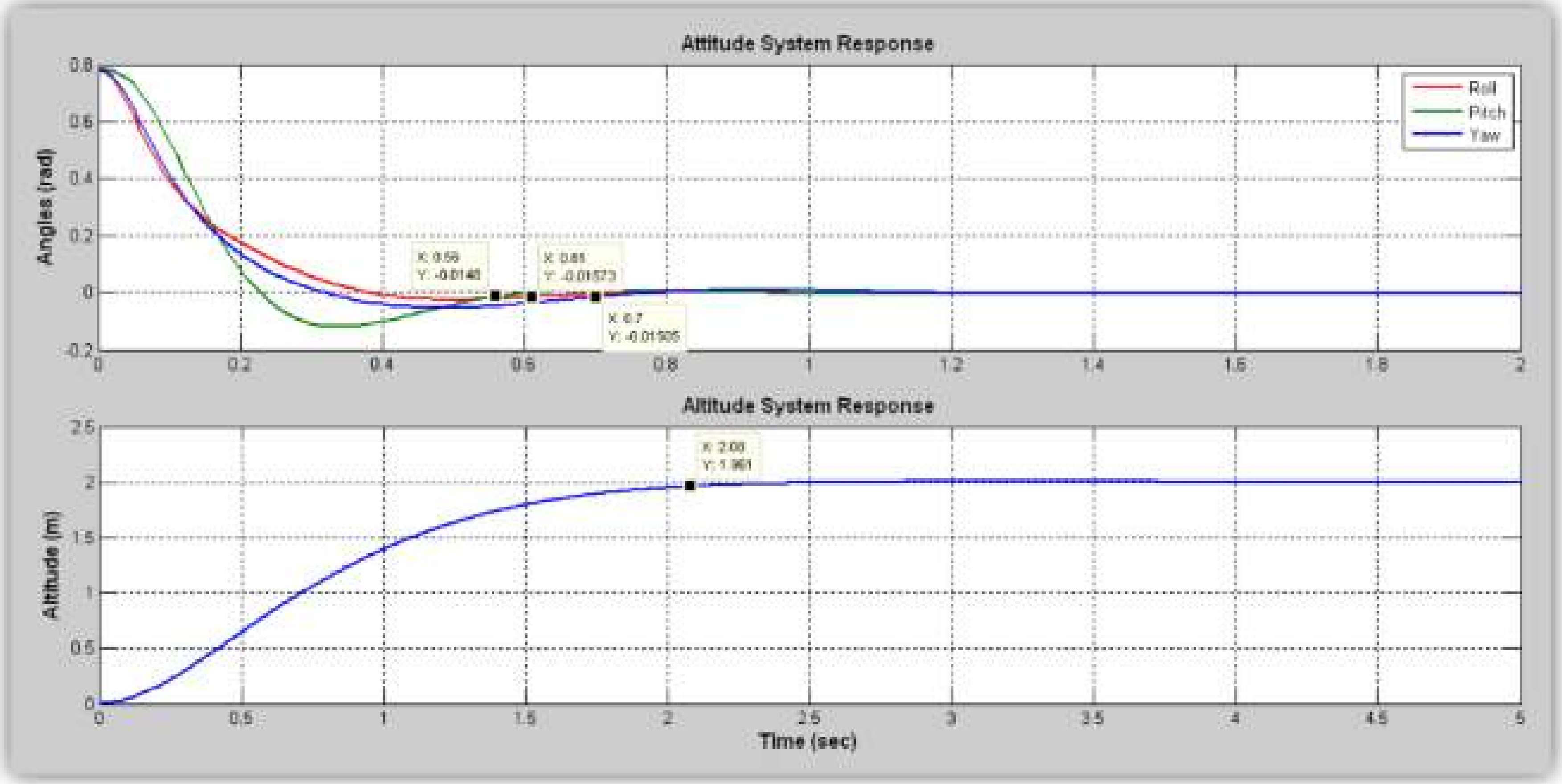


Figure 5-8: Attitude and Altitude Stabilization of the AIBC

The simulation results will be discussed in the comparison section 5.2.3.

5.2.2.3 Lyapunov-Based Velocity Controller, LVC

The LVC is simulated for 5sec with (1m/s) step inputs for the lateral and longitudinal velocities. Figure 5-8 depicts the simulation results of the BF velocities and their attitude commands:

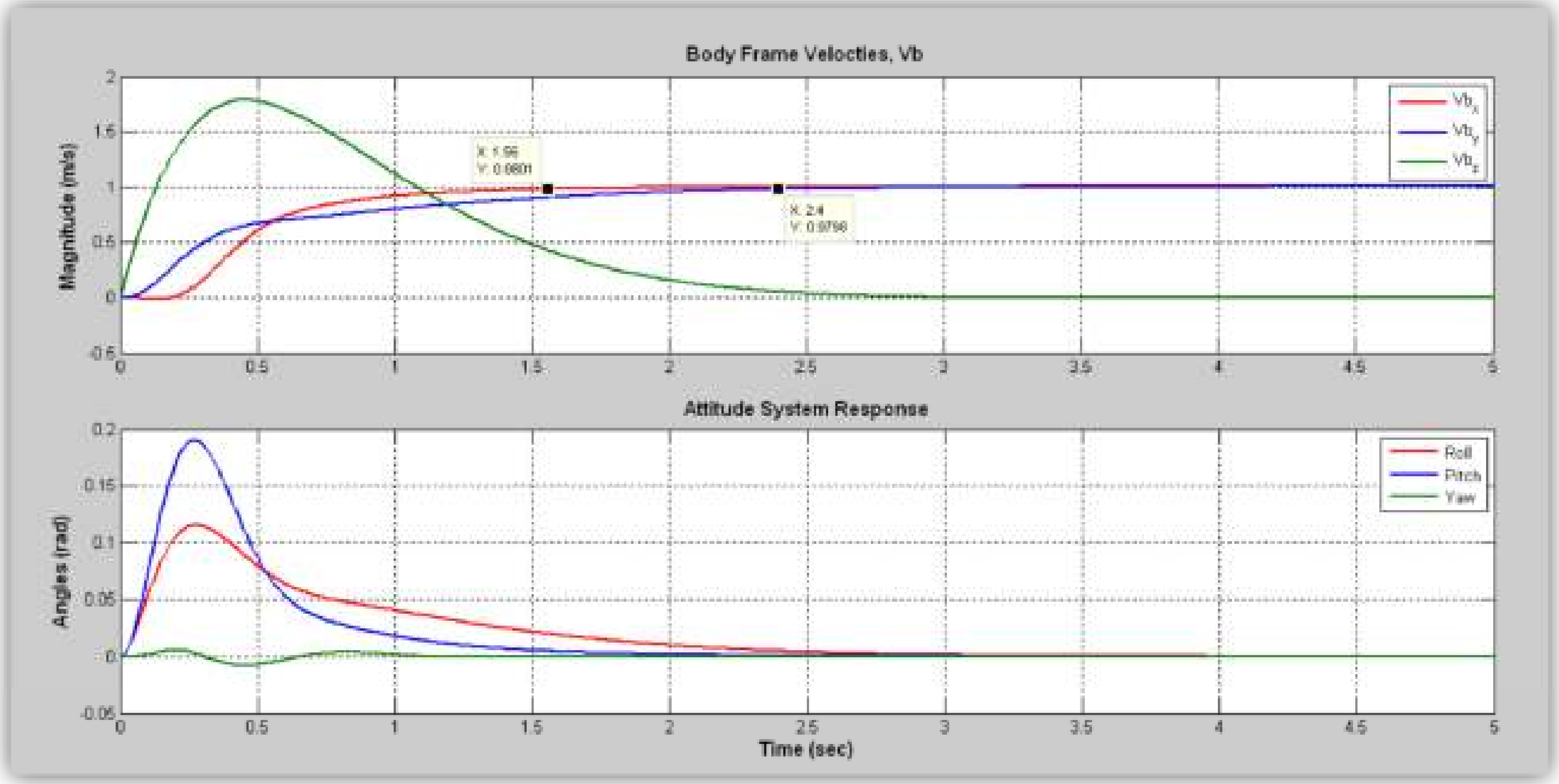


Figure 5-9: BF Velocities with the Attitude Response of the LVC and AIBC controllers

The parameters of LVC have been chosen according to the maximum tilting angles in pitch and roll that the user is defining. In our thesis, we are commanding the two angles for not more than 10^o, which is enough to do the linear motion, and to avoid the flipping over of the quadrotor.

Figure 5-8 shows the BF horizontal velocities and the vertical velocity as well. The altitude control is controlling the quadrotor on a certain elevation (2m in our simulation); therefore, the vertical velocity is converging to zero.

By controlling the velocities we will be able to go for higher control level, which is the path following level in next.

5.2.2.4 Tangent Heading Algorithm, THA

By using b-spline method to generate a track from pre-defined waypoints; the THA is simulated to keep the quadrotor in the track as shown in Figure 5-9. More details about b-spline method and the generated path are illustrated in Appendix C.2.

The simulation in Figure 5-9 shows the 2D-horizontal tracking for an arbitrary path that is generated using Matlab software. The simulation shows good matching between the actual and desired paths with an error less than 20cm. where, a guest wind and a white noise are simulated as well, to give a realistic control while trajectory the track.

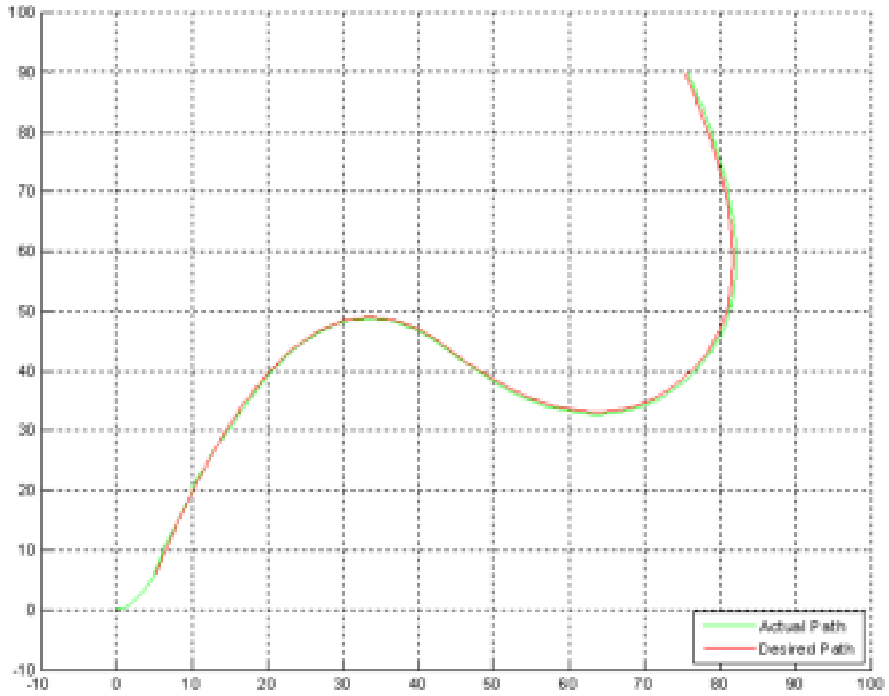


Figure 5-10: THA for path following control

5.2.3 Comparison with other Algorithms

(AIBC & LVC) vs CFLC

At the early stage of developing the AIBC, the simulation is done on a helicopter model, which is developed by Aalborg university helicopter team in [20]. The team controls the helicopter attitude and velocities using the Cascade Feedback Linearization Controller (CFLC). For comparison purposes, the AIBC is used to control the attitude and the LVC to control the velocities. The author of this thesis elaborates more about this comparison in his conference paper in [23]. Furthermore, the simulation results of this comparison are shown in Figure 5-7 and Figure 5-6.

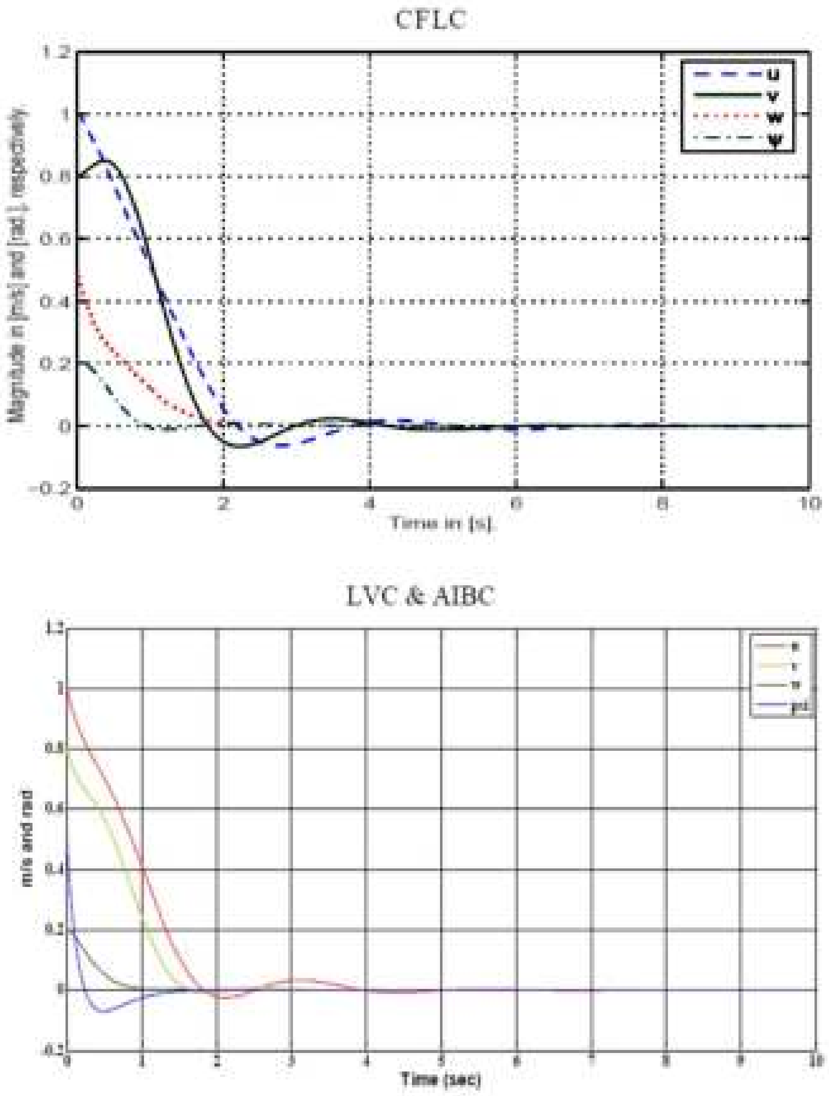


Figure 5-11: Velocities and Heading Comparison

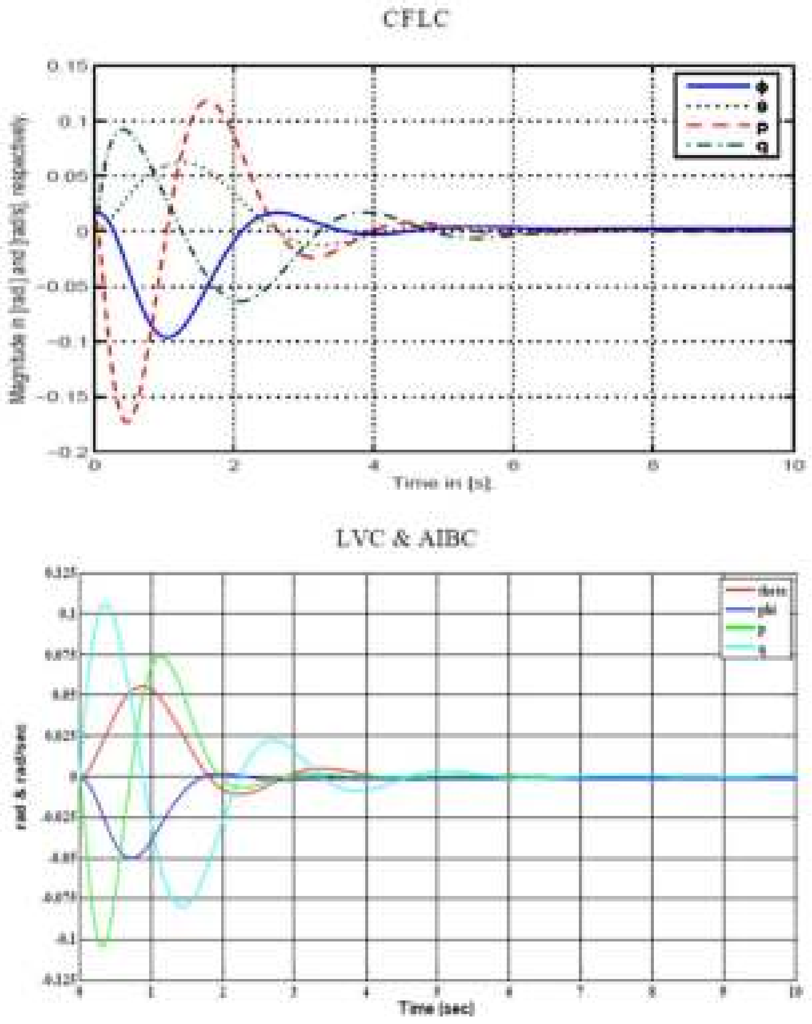
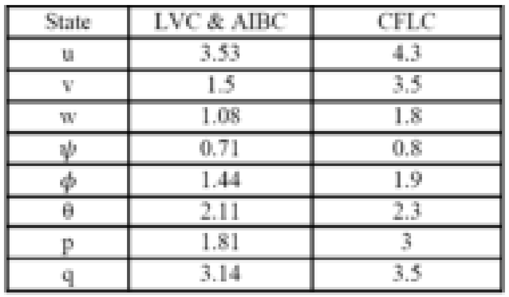


Figure 5-12: Angles and Angles Rate Comparison

In addition, a statistical comparison has been done to give more about the differences between the two approaches. Table 5-3 shows the time response comparison in seconds:

Table 5-3: Statistical Comparison



State	LVC & AIBC	CFLC
u	3.53	4.3
v	1.5	3.5
w	1.08	1.8
ψ	0.71	0.8
ϕ	1.44	1.9
θ	2.11	2.3
p	1.81	3
q	3.14	3.5

From that, the proposed controller shows reasonable differences in the time response of the system.

At this stage, the controllers were implemented on the helicopter model not on the quadrotor. Next, by implementing the proposed algorithms on the AUS quadrotor model, a comparison with a PID controller will be done, not only for simulation but it will be extended to the real time implementation too.

AIBC vs PID vs LQR Controllers:

From Figure (5-3, 5-4, & 5-5), the PID controller gains are tuned after going through two tuning methods: Ziegler-Nichols and TLC methods. In addition, the Integral LQR algorithm considered as another PID tuning method. A reasonable controlling performance is performed using the TLC retuning method and the LQR Controller.

Equations (5.3) relate the values of AIBC parameters and the PID controller gains. The reasons of getting the AIBC parameters from this relationship are:

- ➢ To find a base relation between the two control algorithms, for fair comparison.
- ➢ To check the effect of the nonlinear terms in the control input formulas

Figure 5-14 compares the AIBC, PID, and Integral-LQR algorithms for the attitude and altitude stabilization. From this figure, the AIBC shows better performance than the PID and the integral LQR controllers. The time response is faster in the attitude stabilization, and it's not comparable in the altitude control. Table 5-4 shows the 2% settling time of the two controllers in seconds:

Table 5-4: 2% Settling Time Comparison between AIBC, PID, and LQR Controllers

Variable State	AIBC	PID	Integral-LQR
Roll	0.61	0.76	1.08
Pitch	0.56	1.08	0.95
Yaw	0.7	1.33	0.9
Altitude	2.07	16.5	8.2

The noise rejection and the model error estimations are considered as well in our comparisons. This comparison will be explored in the real time Implementation tests in section (5.4). The difference between the three control algorithms for disturbance rejection is going to be depicted in details.

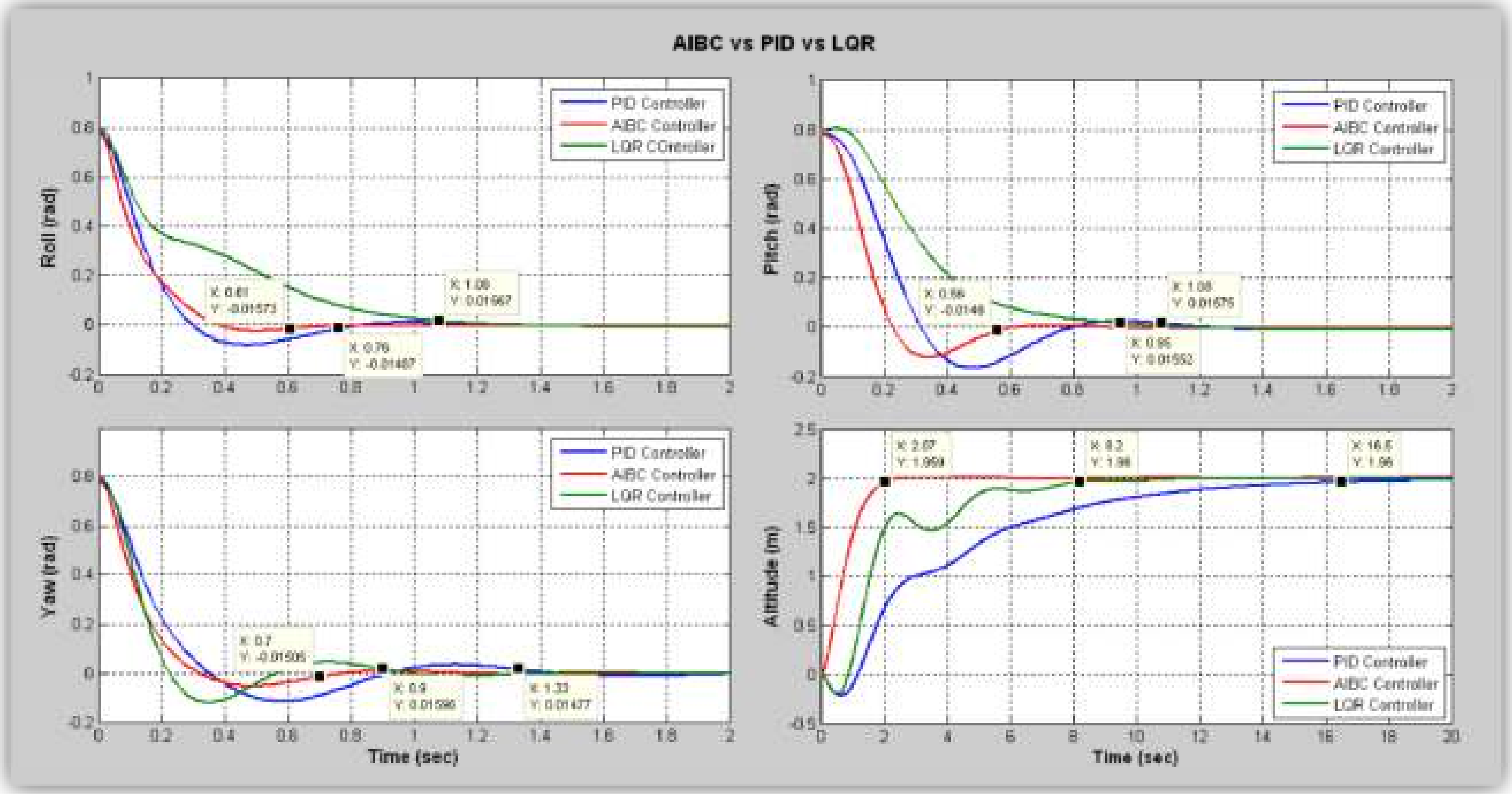


Figure 5-13: AIBC vs PIC Controllers

Backstepping, BFL, and BLMS controllers:

As mentioned in chapter 4, the Fuzzy logic and the Least Mean Square Algorithms are found to enhance the work of the backstepping controller. Figure 5-12 shows the comparison between the three controllers according to the system time response and the control efforts.

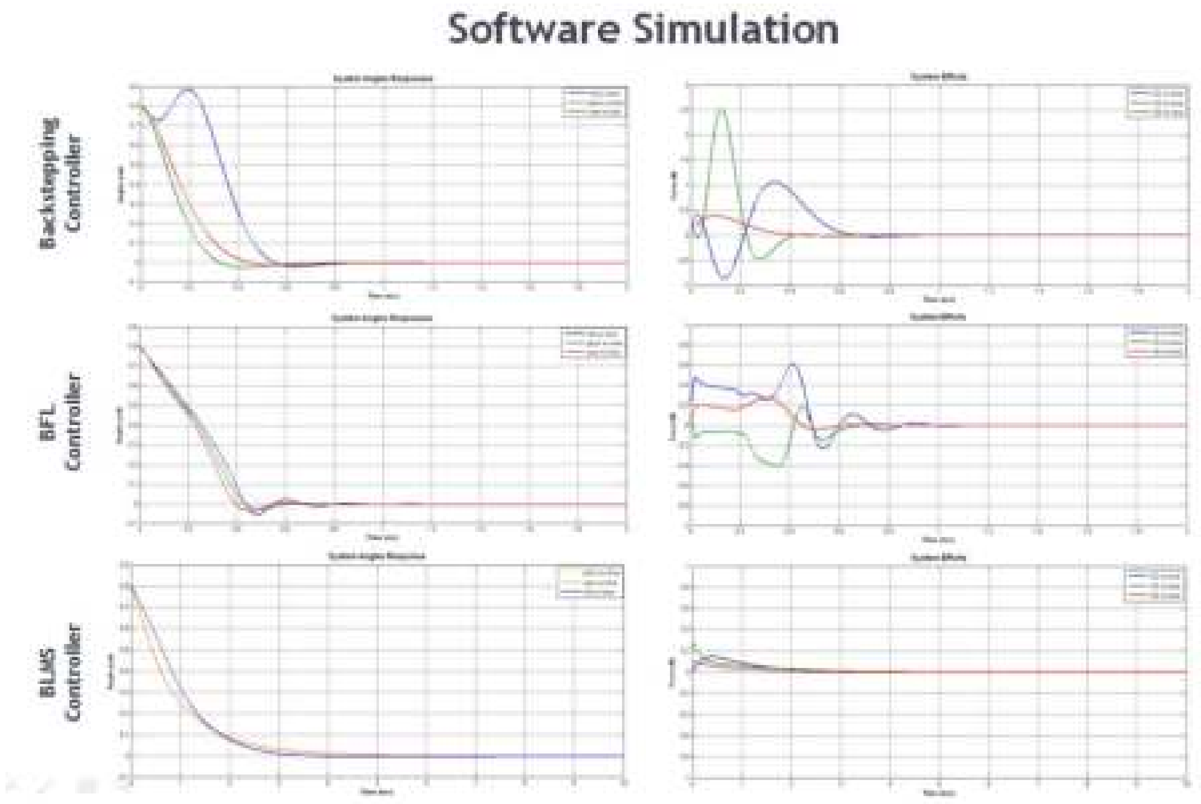


Figure 5-14: Backstepping, BFL, and BLMS Simulation Results

From that, the BFL controller gives the best controllability among the three controllers, where a compromised system response in terms of settling time and efforts was achieved, while the backstepping controller showed increased demand on the controller efforts.

BLMS controller which is designed to minimize the system efforts, but it suffers from prolonged (6 to 7 times greater) response time when compared with the other controllers.

5.3 HARDWARE-IN-THE-LOOP-SIMULATION, HILS

As an intermediate stage before going to the practical tests, the HILS is used to validate the control algorithms that are downloaded on the microcontrollers. Mainly, the HILS is consisting of the dSPACE and the MPC555 controllers.

The structure of the HILS is depicted on the following Figure 5-:

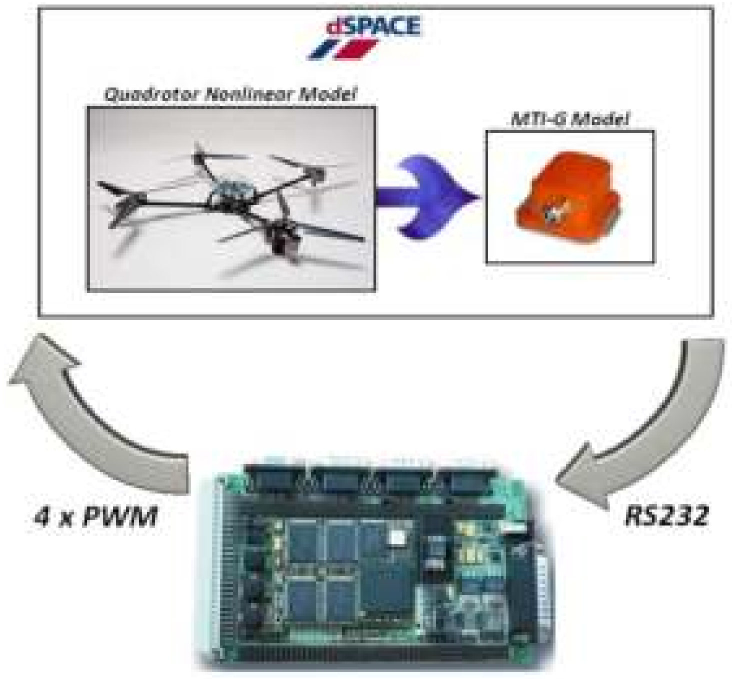


Figure 5-15: HILS Structure

In the dSPACE controller, the quadrotor and its sensoring devices are modeled as follows:

- ➢ The 6-DOF nonlinear equations of motion (from chapter 3) are programmed on the dSPACE to mimic the real quadrotor dynamics.
- ➢ The MTi-G IMU/GPS device will be modeled to send the same message of the real one.
- ➢ The variable states will be sent in a certain protocol (MTi-G protocol). This is done according to the following steps:
    - o Convert each variable state from a decimal number to a single point floating number (IEEE754 representation) which is a 4 bytes number.

- Send the variable states in a certain sequence which similar to the MTi-G communication protocol.

➢ The dSPACE will read the four PWM signals from the microcontroller.

➢ The PWM signals will be converted to the control Inputs (Thrusts).

Mainly, the dSPACE side will contain:

➢ *The Quadrotor model* : represents the nonlinear equations of motion

➢ *The MTi-G IMU model:* mimic the real IMU communication protocol

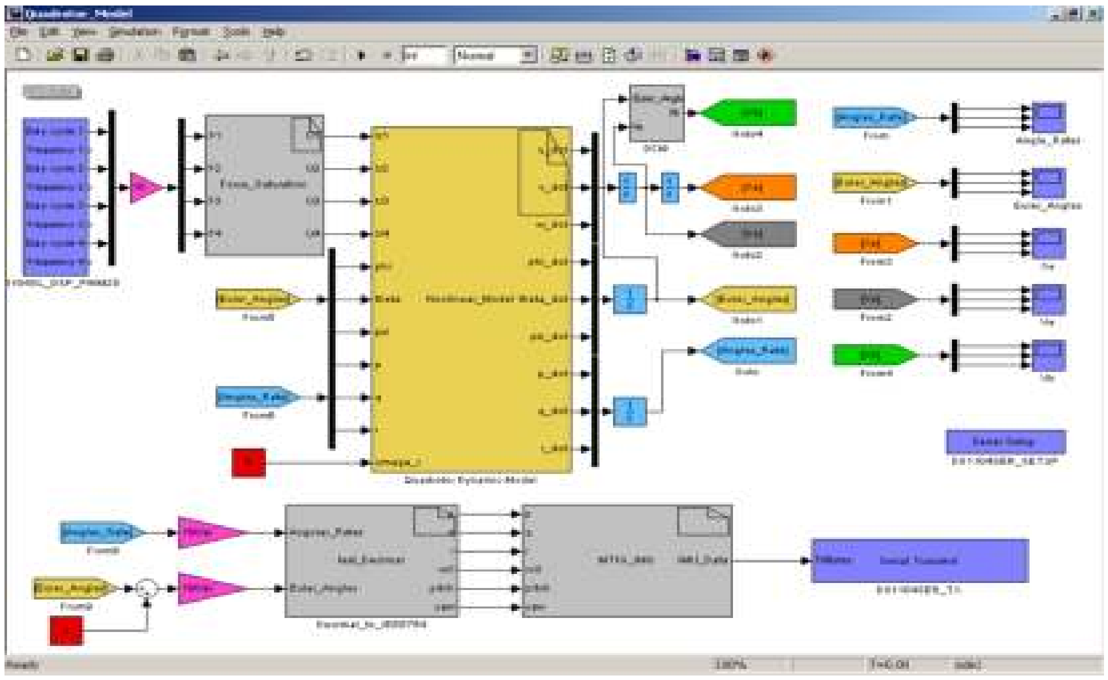


Figure 5-16: dSPACE Simulink Model

On the other hand, the control algorithms are programmed in the microcontroller for testing and validation before going to the test field. The Simulink programs will be represented in the real time implementation section 5.4.

A successful real time simulation of the HILS was achieved and the results are similar to the simulation results:

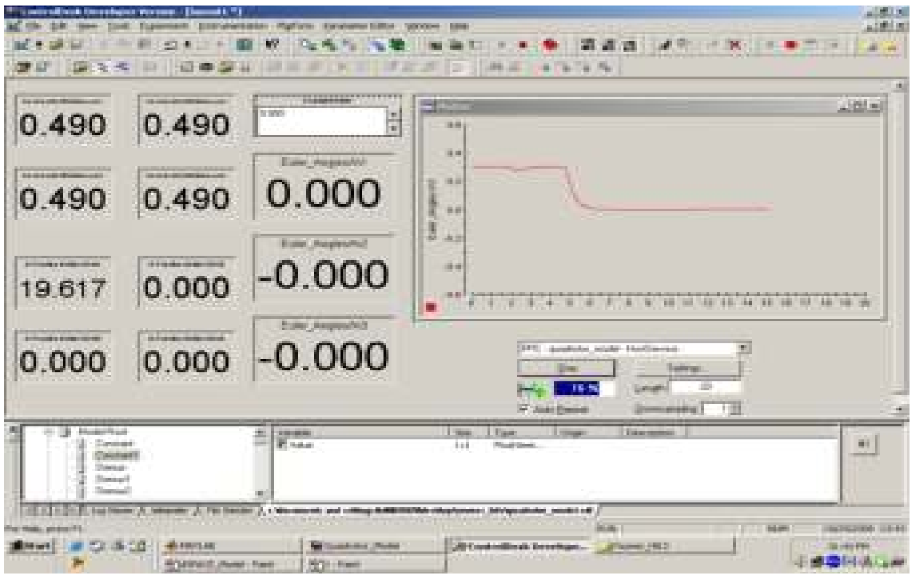


Figure 5-17: HILS Results

5.4 REAL TIME IMPLEMENTATION

This section intends to show that the proposed AIBC controller is able to stabilize the AUS quadrotor in attitude, altitude and hover conditions. The verification is carried out in section (5.2) in Simulink software. Next, the real implementation for the flight tests is presented to validate the proposed algorithms.

5.4.1 Attitude Stabilization

In chapter 2, the high level architecture is depicted in Figure 2-11. The onboard system contains the controlling processing unit of the AUS quadrotor, which is the MPC555 microcontroller. The proposed control algorithms in this thesis are programmed on this microcontroller using Target Support Package FM5 in Simulink.

For test and validation the AIBC, PID, and Integral-LQR Controller are implemented on the AUS quadrotor. The flight tests were done in non-windy outdoor environment. Mainly, the comparison highlights the stabilization of the attitude angles in different initial conditions. Figure 5-19 & 5-20 show the AIBC and PID Simulink diagrams, respectively.

Each Simulink diagram contains many C-MEX blocks for C programming in the Simulink environment. The C-language aid to develop the control algorithms rather than dealing with the blocks, especially with the mathematical representations. The C-MEX codes are represented in Appendix A.

Figure 5-21 compares the three controllers while maintaining the attitude angles around the origin. For roll and pitch angles, the AIBC shows better stabilization response than the other two controllers. The AIBC stabilizes the angles in a range of less than $\pm 1^o$, compared to $\pm 2^o$ in the PID controller, and $\pm 1.5^o$ in the Integral-LQR controller. On the other hand, the noise and disturbance rejection are enhanced in the AIBC, where less fluctuation around the origin is depicted in Figure 5-21.

For the yaw angle, Figure 5-22 shows the stabilization comparison as well. Where, the AIBC is performing better than the PID and Integral-LQR in maintaining the yaw angle around the north direction.

On the other hand, a test using 10^o step input is performed. The performance of the time response and the stabilization around the desired value are validated. As shown in Figure 5-23, the AIBC shows faster response and better stabilization among the other controllers. Furthermore, the optimization of the control inputs of the Integral-LQR Controller explains the slower settling time and the smoother stabilization compared to the PID controller.

The effects of considering the nonlinearities in the AIBC control inputs (efforts) are clearly seen from these experiments. The adaptation and integration parts are performing better stabilization results as well.

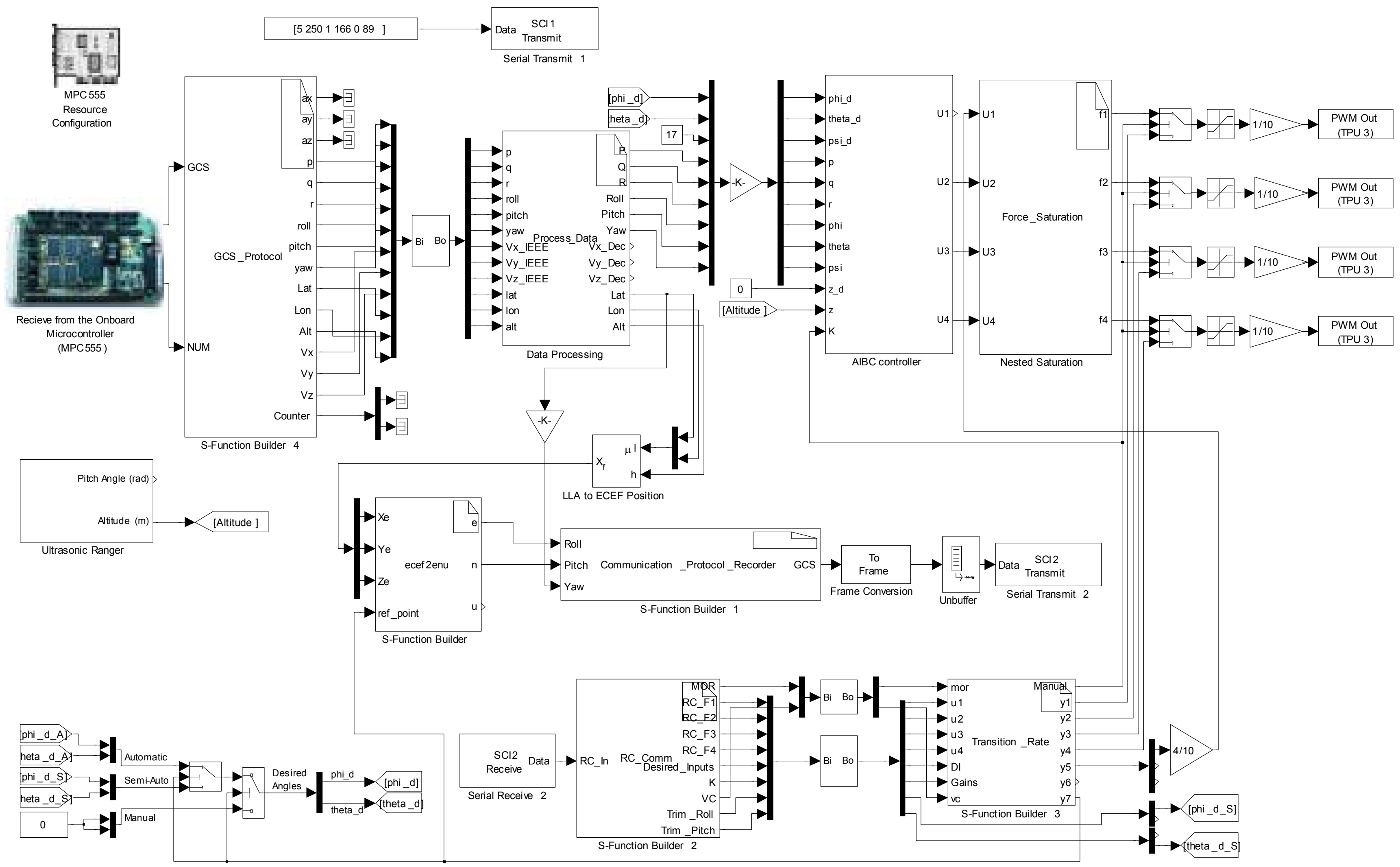


Figure 5-18: Simulink model of the AIBC Controller

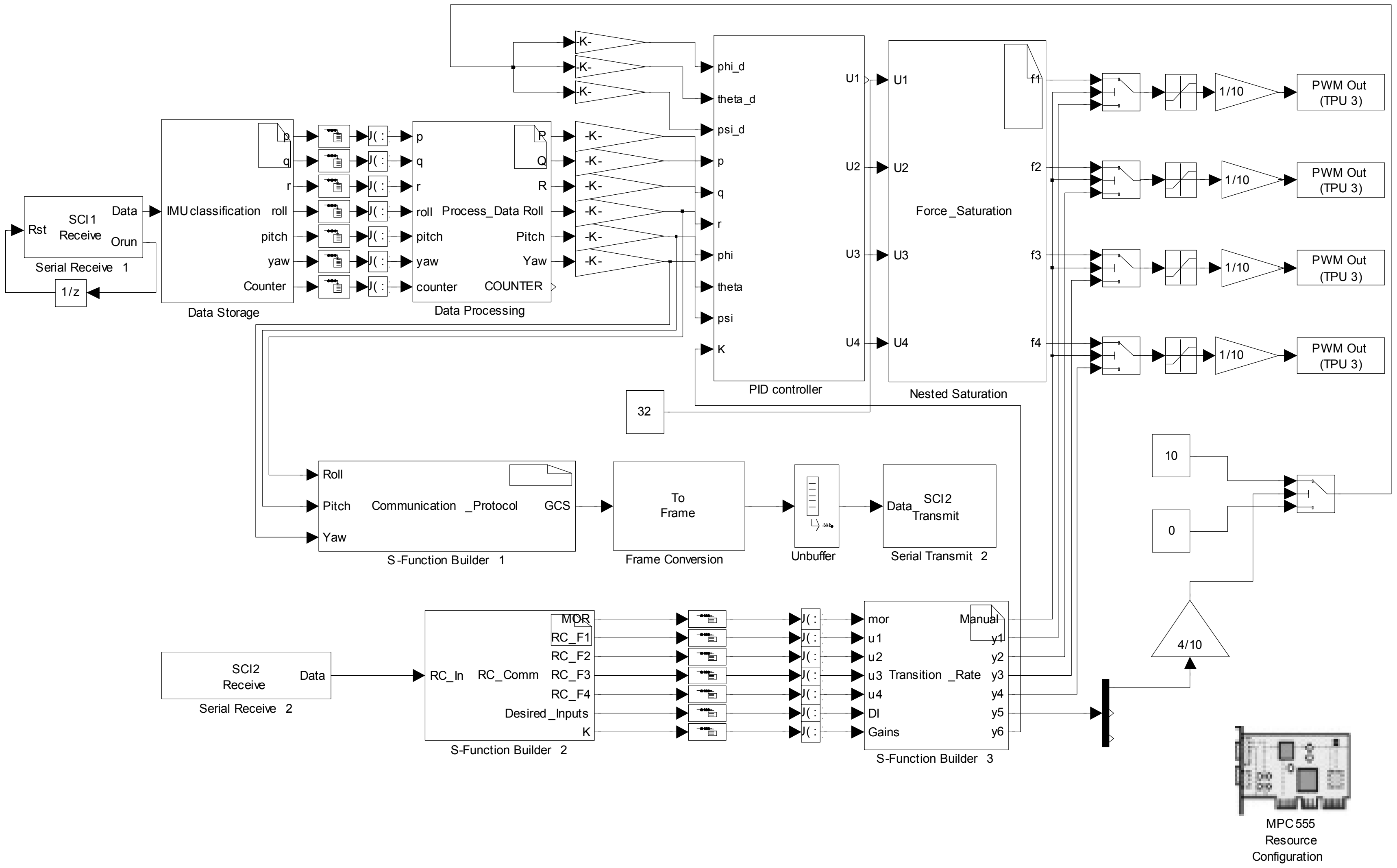


Figure 5-19: Simulink Model of the PID controller

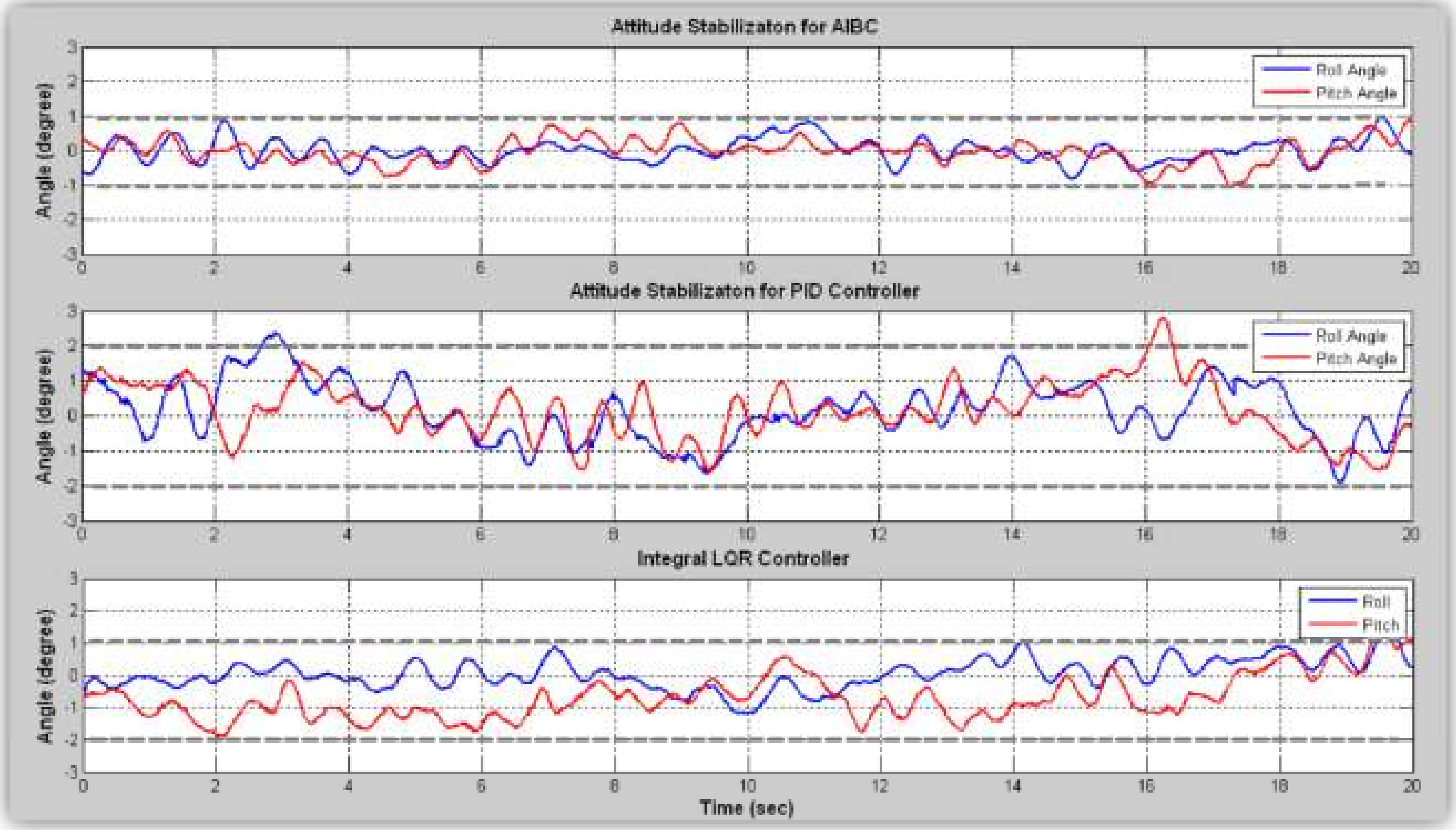


Figure 5-20: AIBC vs PID in Attitude Stabilization (Roll and Pitch Angles)

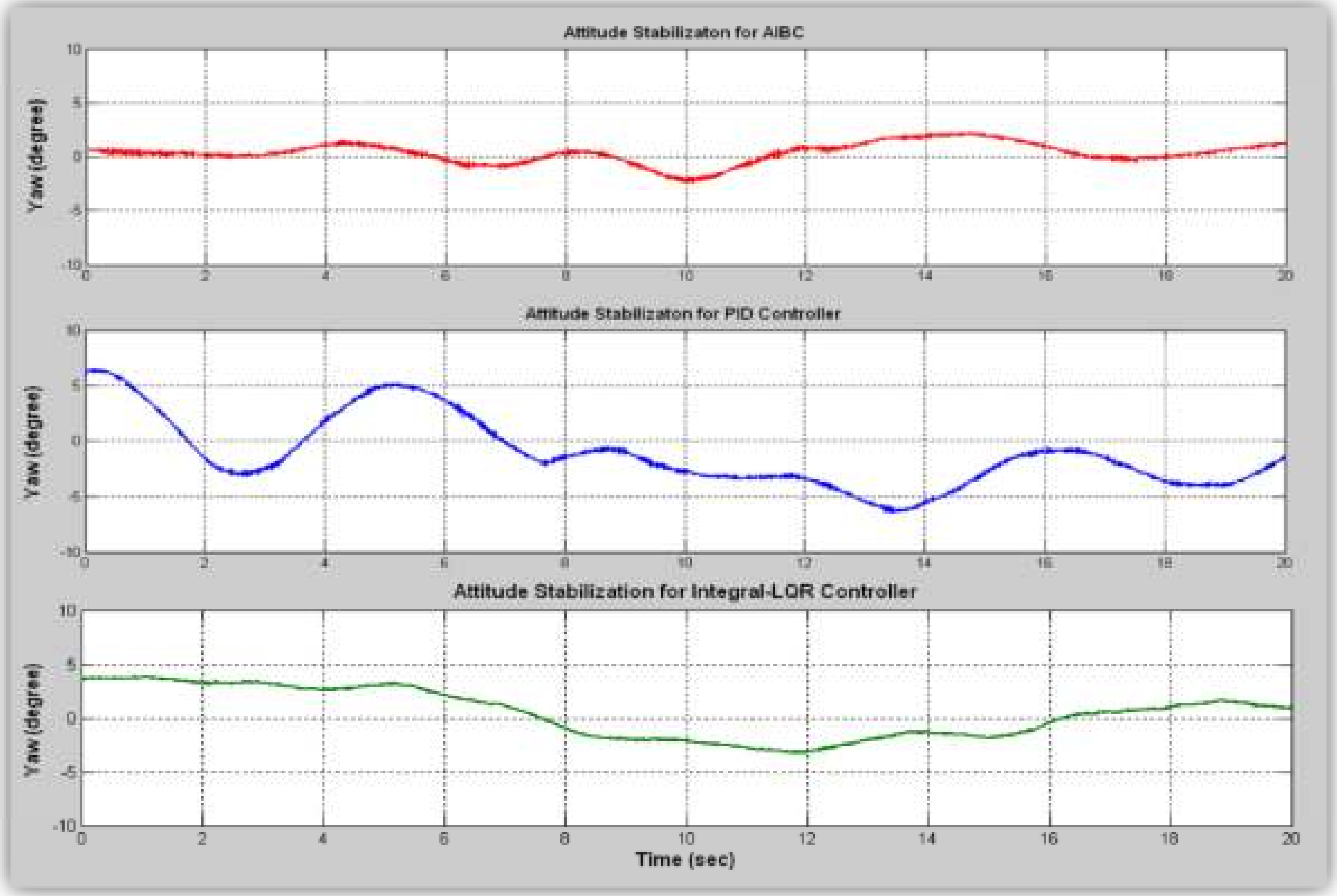


Figure 5-21: AIBC vs PID in Attitude Stabilization (Yaw Angle)

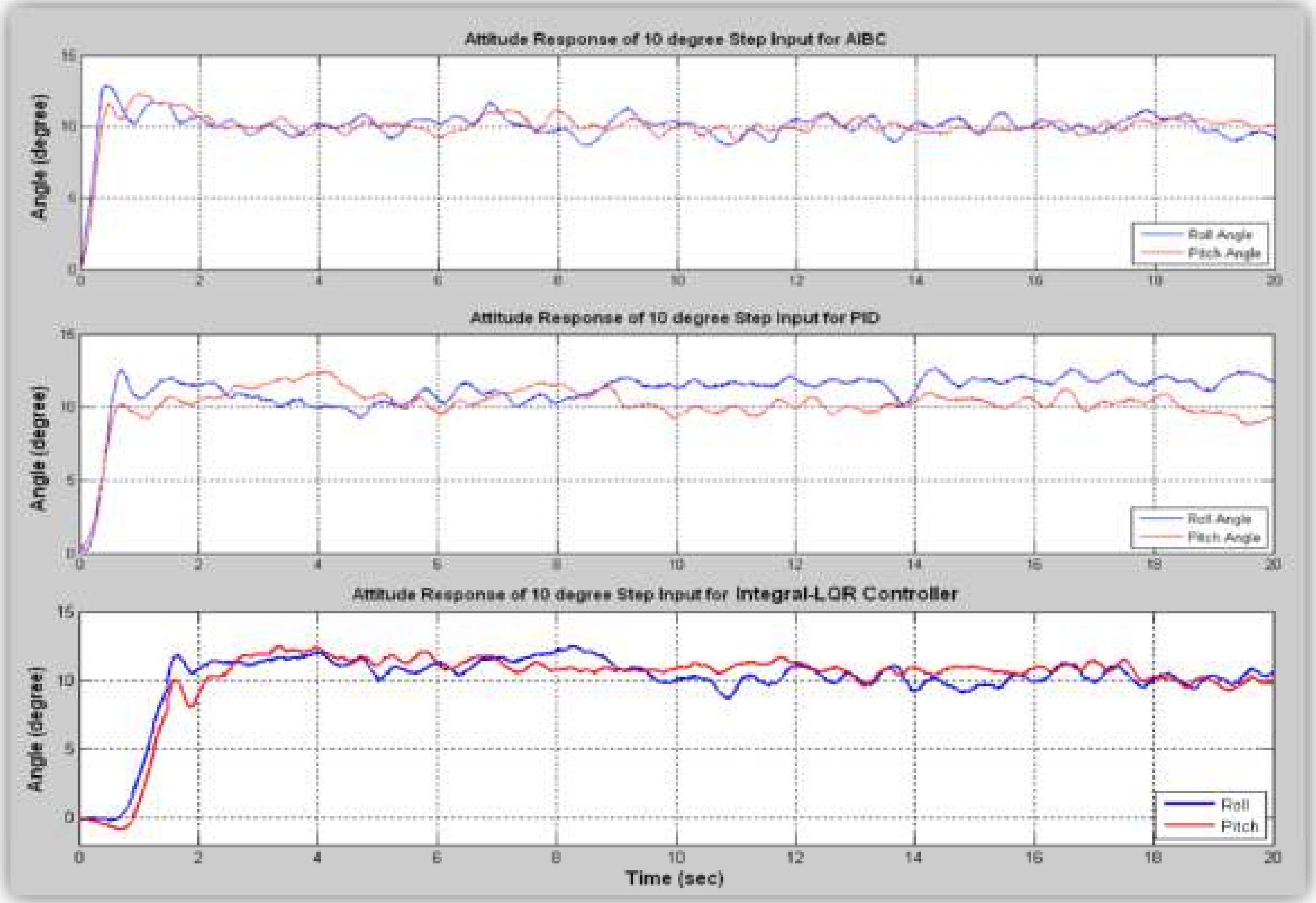


Figure 5-22: AIBC vs PID in Attitude stabilization (10^{o} step input)

5.4.2 Hover Control

The AIBC algorithm is considered in the hover flight conditions, after it shows a good performance compared to the PID and Integral-LQR controllers. This section includes the hover flight tests. The position and velocity data are collected using different devices according to the flight environment. The CMUcam is for the indoor environment and the MTi-G GPS for the outdoor one. This section will represent and discuss the results of the hover test flights.

5.4.2.1 Indoor Hover Control

At the early stage, the hover is done using the CMUcam. The implementation of the camera and the quadrotor are discussed in chapter 2. The position data is measured according to the location of the quadrotor green dome. The uncertainty of position readings makes the estimation of the velocity reading quite hard. Therefore, the position control was a proportional controller that is depending on the position readings only.

The AUS quadrotor tethered by a rope of 1.5m length, to keep the quadrotor within the camera detection area. The camera fixture, the quadrotor preparation, and the test location are discussed in chapter 2, section 2.3.3.3.

The position of the quadrotor is depicted in a 2D-plot (Figure 5-24) and 3D-plot (Figure 5-25) while hovering for 15 seconds around the origin of the camera window. The hovering data is attached with this thesis on a CD “Hover_CMUcam.avi”.

As expected, the quadrotor is oscillating around the origin due to the lack of the velocity readings. Figure 5-24 shows that the vehicle is passing from the origin many times, but still oscillating around it. On the other hand, the quadrotor was controlled to perform 50cm altitude height, as shown in Figure 5-26

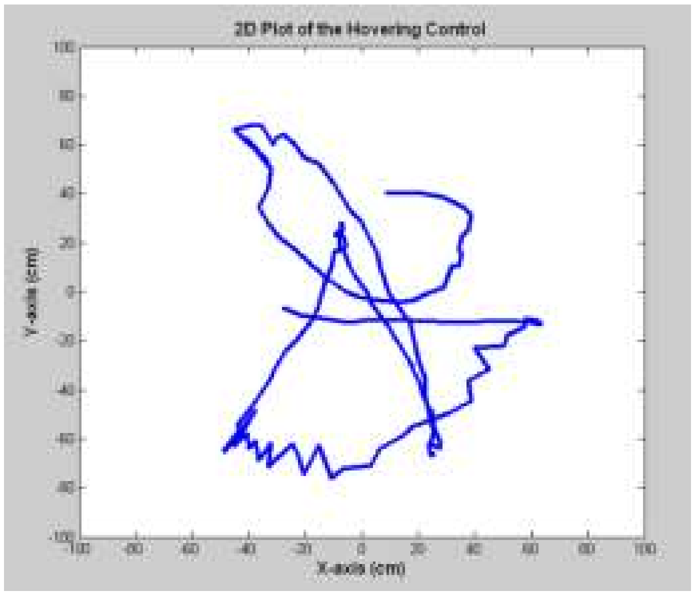


Figure 5-23: 2D-Plot of the Indoor Hover flight

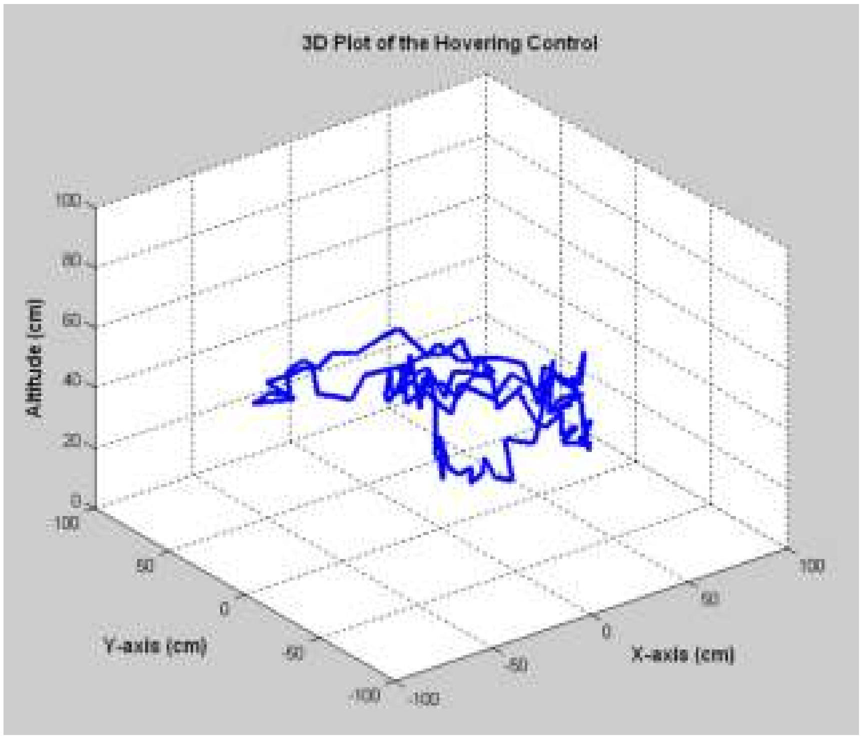


Figure 5-24: 3D-Plot of the Indoor Hover flight

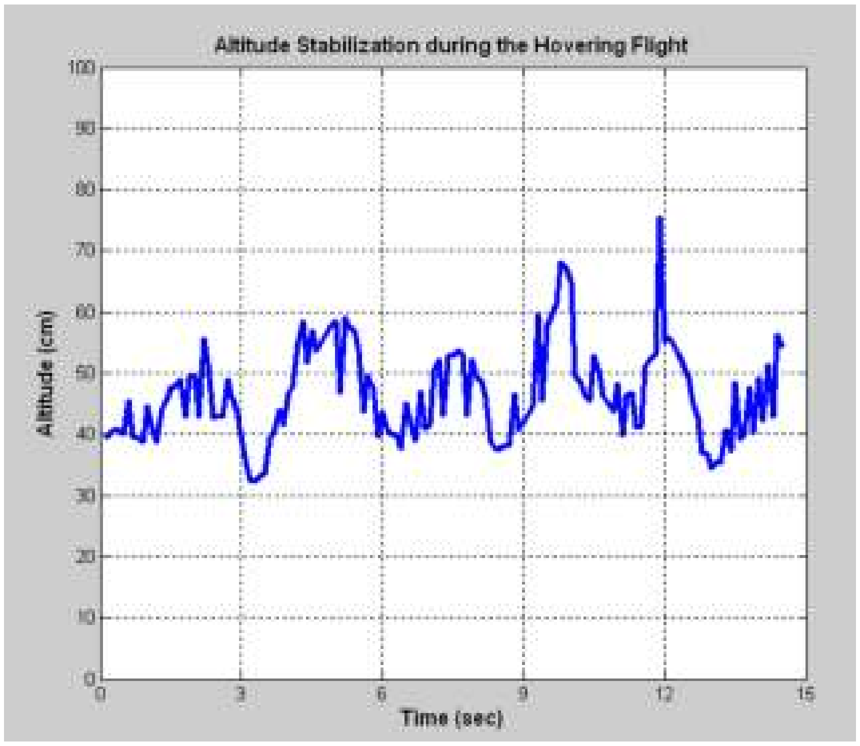


Figure 5-25: The Altitude Control during the Hovering Flight

5.4.2.1 Outdoor Hover Control

The quadrotor hover controller is the AIBC algorithm cascade with a Lyapunov-based position controller. The position controller is a combination of the LVC and the THA algorithms. More details about the proposed control algorithms is illustrated in chapter 2.

In this experiment, our controller is relying on the GPS reading provided from the MTi-G unit. The unit is filtering the GPS and the IMU raw data using Extended Kalman filter, and provides the filtered position and velocity data to the MPC555. More details about the MTi-G unit and the Extended Kalman filter can be found in chapter 2, section 2.3.2.2.

For outdoor environment, a movable setup is constructed to do the test in different locations (depends on the GPS coverage). Figure 5-27 depicts different views of the movable GCS setup.

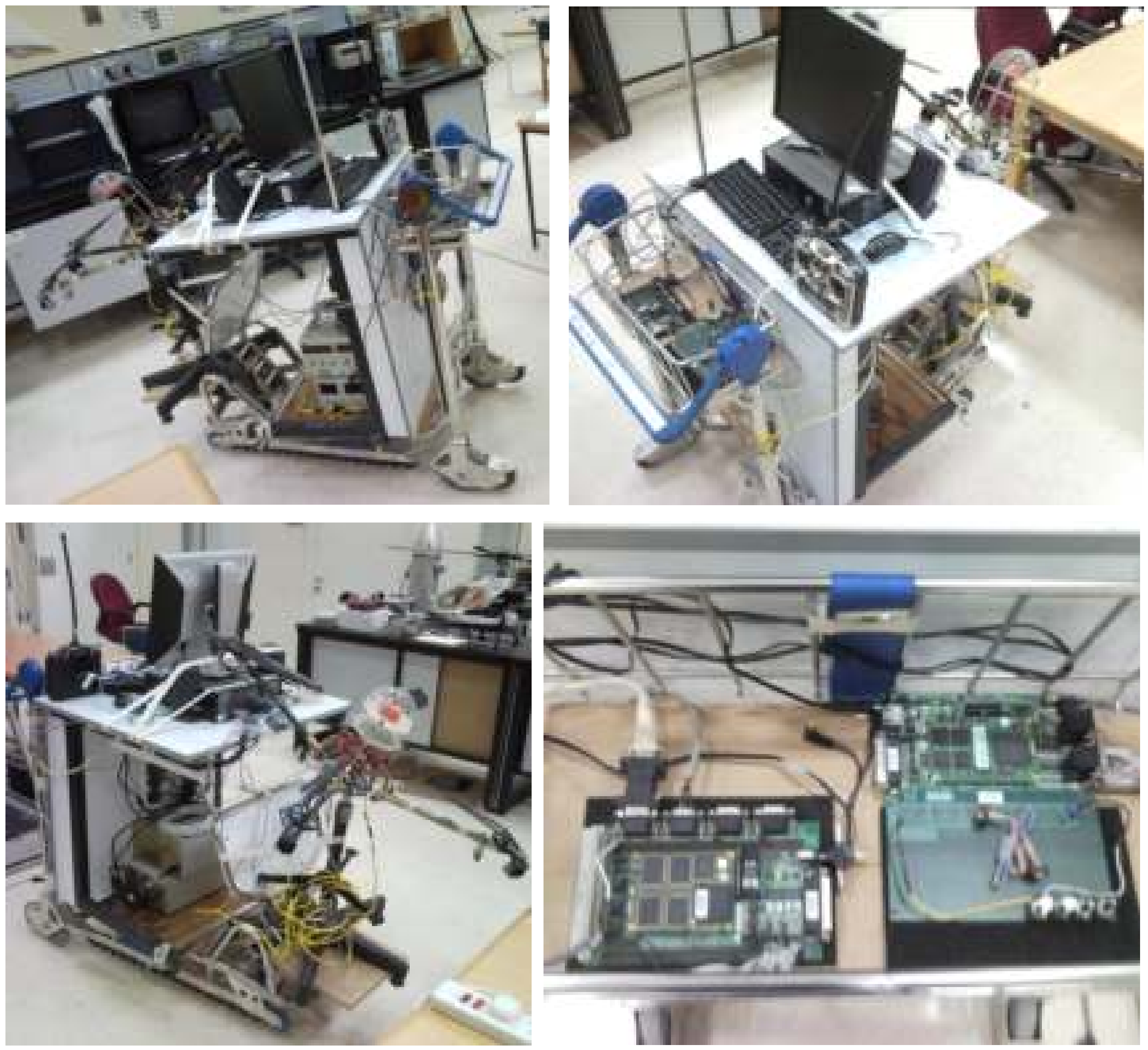


Figure 5-26*:* Movable GCS Setup

All safety precautions are taken into accounts for the final tests. Inspite of that, the quadrotor flipped over two times due to a malfunction of the microcontroller side. Later on, we find that the microcontroller was resetting during the flight test, the reason behind that is a loose connections.

Furthermore, the MOS is used during the flight tests to change between the different flights modes. More details about the MOS is illustrated in chapter 2, section 2.3.3.4.

Many experiments are conducted to check the accuracy of the filtered GPS/IMU data. The best results of the position readings achieved within a circle of 2m radius. While the velocity readings obtained with a maximum drift of 0.3m/s.

The experiment results were encouraging more than the indoor hovering control. Where, the velocity data reduces the oscillation. On the other hand, the uncertainty, the low frequency, and the jumping of the GPS readings makes the quadrotor moving around within a circle of 4m radius.

Next, a successful hover flight test will be depicted. In this flight test, the quadrotor keeps flying for 30 sec in a windy environment. Figure 5-28 and 5-29, are demonstrating the hover flight experiment readings.

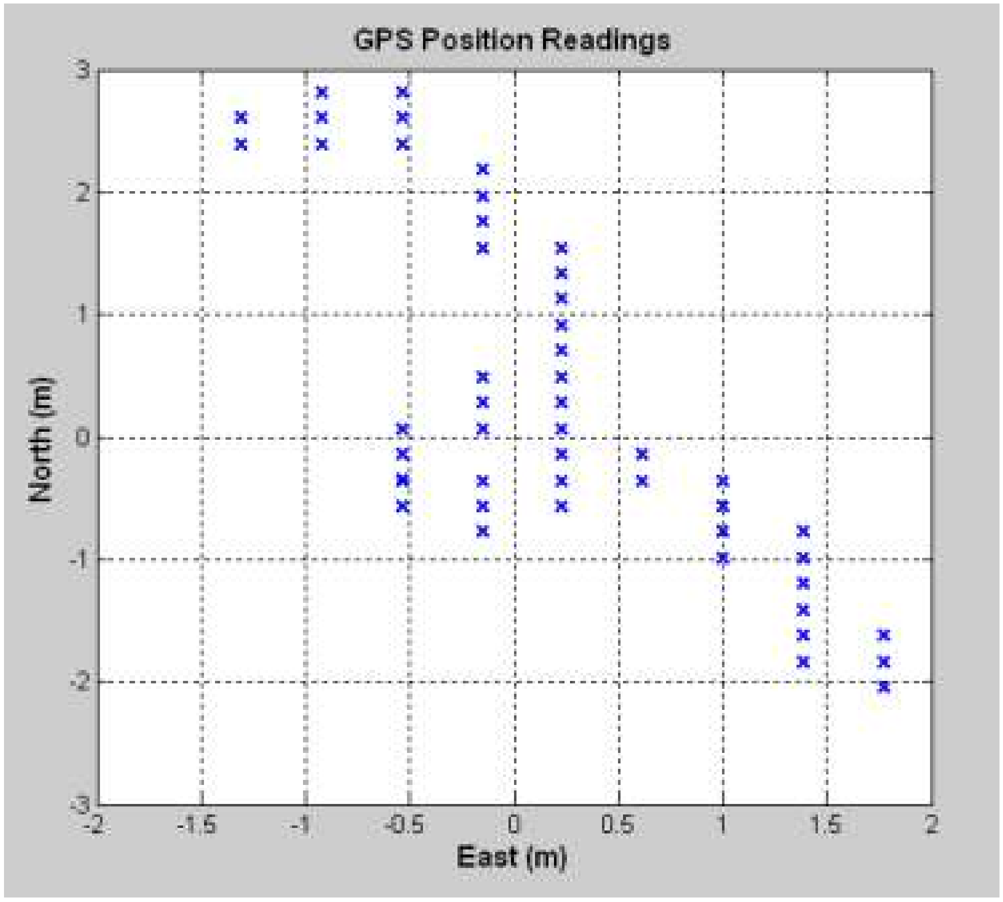


Figure 5-27: MTi-G IMU/GPS Position Reading

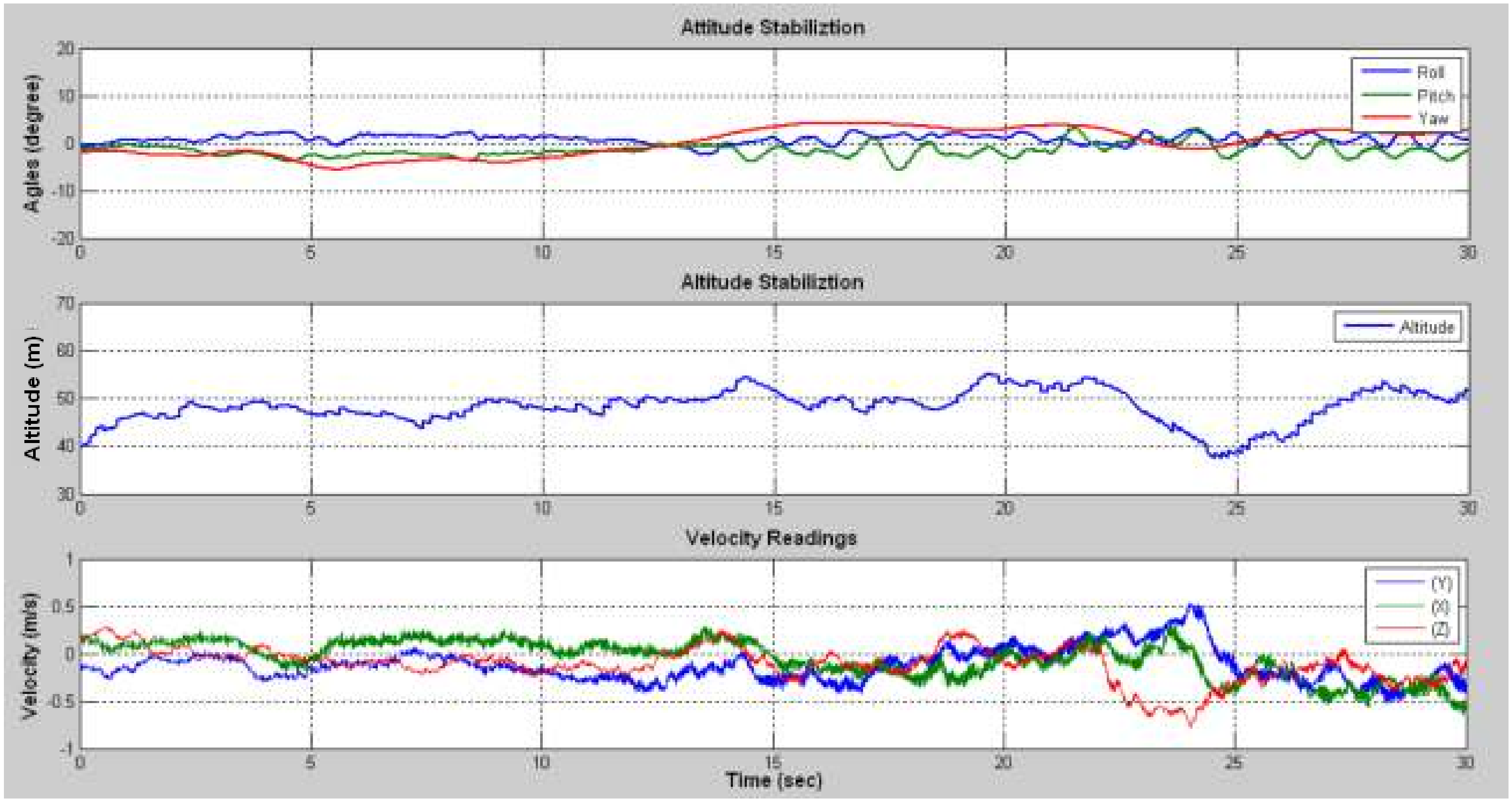


Figure 5-28: Attitude, Altitude and Velocity Responses during the Outdoor Hover Flight

Because of the windy environment, the attitude angles were fluctuating more than the indoor flight tests. Besides that, the position control is commanding the desired angles within a range of $\pm 2^{o}$ to do the linear motion. The small commanding range will ensure the smooth correction of the position, and it will ensure the stability of the quadrotor too.

Figure 5-28 draws the position data of the GPS, and not the actual quadrotor position data. But it gives an indication about the area that the quadrotor was moving in. In contrast, the quadrotor was moving diagonally during the flight test like what is depicting in Figure 5-28. And this is validating the work of the position controller. More details about the outdoor hover test is included on the attached CD “Hover_GPS.avi”.

In Figure 5-29, the quadrotor was stabilizing around 50 cm height. This in turn gives more room space to the quadrotor to move in. where, the quadrotor is tethered with a rope of 4m length.

From that, the quadrotor was able to hover in the windy outdoor environment, and it was following the position command from the GPS. The high uncertainty in the position reading, which is provided from the commercial GPS, is resulting to the quadrotor drift during the hover flight. On the other hand, the adaptation scheme and the integral action in the backstepping algorithm provide us with a stable and smooth hover flight.

CHAPTER 6

CLOSURE

In this thesis, the AUS quadrotor project was established in the AUS Mechatronics Center as a new platform in the AUS-UAVM project. Hopefully this project would continue and develop in the field of unmanned autonomous robot systems.

6.1 SUMMARY OF WORK

The followings are a summary of work done in this project:

- Introduce and establish the AUS quadrotor project in both hardware and software aspects.
- Review the relevant literature along with the most successful platforms and algorithms.
- Demonstrate the AUS quadrotor in terms of model, design, and implementation.
- Provide a comprehensive overview about the proposed control algorithms and explain in details how the algorithms were developed.
- Finally, test and evaluate the AUS quadrotor through extensive simulation results and flight tests, and discuss the results.

6.2 CONCLUSIONS

The quadrotor design is different from the conventional helicopter design. It couldn't adjust anything but the rotational velocity of the rotors. Despite of the quadrotor design, the quadrotor is freely moving in 6DOF, like the helicopter.

The goal of this project was to attain autonomous flight with the AUS quadrotor in hover state. To achieve this, the quadrotor was modeled in order to develop and simulate the controller. The model of the quadrotor is a nonlinear model. Therefore, nonlinear control algorithms are proposed to deal with the nonlinearities and the model uncertainties as well.

In this thesis, a novel nonlinear control approach is proposed based on a recursive Lyapunov methodology using the backstepping technique. The new approach employs the adaptation scheme to estimate the ignored parts in the dynamic model, and the integral action is used to overcome any disturbances and model uncertainties. Altogether, the Adaptive Integral Backstepping Controller (AIBC) is introduced to control the attitude and the altitude of the autonomous AUS quadrotor. Furthermore, the Lyapunov algorithm is used to include the velocity control and the tangent heading algorithms for position control.

Simulation results and test flights had been conducted to validate the control design schemes, and to show that the proposed Adaptation and integral action of the backstepping controller shows better performance in dynamic response. Moreover, AIBC, LVC and THA together show robustness in hover and like-hover situations. Comparisons with previous work and with the conventional PID and Integral-LQR controllers show that the proposed algorithm dealing with the nonlinear dynamics batter than the linear controllers (PID and Integral-LQR).

On the other hand, the AIBC performance is enhanced by adjusting the positive parameters of the controller. The Fuzzy Logic Control utilizes the advantage of fuzzy logic to schedule the backstepping controller parameters. This resulted in improved performance compared with that achieved by backstepping controller alone. On the other hand, the Least Mean Square algorithm is used to minimize the control efforts by finding the parameters of backstepping controller which provide minimum control efforts to the system.

Finally, all developed algorithms had been tested and validated. Theoretically by showing extensive simulations using Matlab/Simulink Environment, and practically by performing the HILS, then the flight tests to validate our simulation results.

6.3 Recommendations for Future work

This work was established to pave the way for others to continue and contribute more to this project.

The physical platform can be improved in the following ways:

1. Reduce the weight of the quadrotor by:
    a. Designing a lighter protection structure than the training set
    b. Replacing the rapid development board of the MPC555 with a Printed Circuit Board (PCB).
2. Increase the thrust generated from the actuators, by adding a third blade on each rotor.
3. Remove the attached cables to the quadrotor during the flight state, by:
    a. Implementing the wireless transceiver.
    b. Providing the power by using Li-Po batteries to the actuators and the onboard system.
4. Implement an indoor positioning system to provide accurate readings of the position and velocity.

The control algorithms can also be developed in the following ways:

1. Adjust the parameters of the AIBC algorithm by linearizing the model in different set points. Then schedule the appropriate control parameters on each set point, and cluster them in membership functions using the fuzzy logic control.
2. Move from the hover flight to non-hover flight states, by implementing the THA algorithm to follow the desired paths.
3. Work on a path planning techniques, and improve the path following algorithm to track 3D trajectories.
4. Develop GCS that has the functionality to adjust the parameters of the control algorithms besides the monitoring purposes.
5. Investigate the use of the adaptation law in greater details to estimate the variation of the quadrotor dynamics, due to the drag forces, and the modeling errors.

BIBLIOGRAPHY

[1] Y. M. Alyounes and M. A. Jarrah, "Attitude Stabilization Of Quadrotor UAV Using Backstepping Fuzzy Logic & Backstepping Least-Mean-Square Controllers," in *proceeding of the 5th International Conference on Mechatronics and its Applications, ISMA'08*, Amman, Jordan, 2008.

[2] S. G. Fowers, "Stabilization and Control of a Quad-rotor Micro-UAV Using Vision Sensors," M.S. thesis, Brigham Young University, Provo, UT, 2008.

[3] Aviastar, "Quadrotor Helicopter," All The World's Rotorcraft, Oct. 1997. [Online]. Available: http://www.aviastar.org. [Accessed: Dec. 13, 2008].

[4] C. F. Petersen, H. Hansen, S. Larsson, L. T. Madsen, and M. Rimestad, "Autonomous Hovering with a Quadrotor Helicopter," M. S. thesis, Aalborg university, Aalborg, Denmark, 2008.

[5] S. Bouabbdallah, "Design and Control of Quadrotors with Application to Autonomous Flying," M. S. thesis, Abubaker Belkad University, Algeria, 2007.

[6] DraganFLY Innovation Inc., "R/C Helicopter," X-Pro Quadrotor, 2003. [Online]. Available: http://www.rctoys.com. [Accessed: Sep. 2, 2007]

[7] J. Coker, "Honycomb Tubes," 1998. [Online]. Available: http://www.jcrocket.com/honeycomb.shtml. [Accessed: March 10, 2008]

[8] " TTi EX354D Power Supply," 2003. [Online]. Available: http://www.4gte.com/EquipmentPages/EX354Dpowersupply.htm. [Accessed: Jan. 6, 2009]

[9] Thunder Power RC, "High Performance Li-Po Batteries," 2003. [Online]. Available: http://www.thunderpowerrc.com/. [Accessed: Sep. 15, 2008]

[10] Freescale semiconductor, "MPC-5xx Microcontroller," Codewarrior Compiler, 2004.

[Online]. Available: http://www.freescale.com/. [Accessed: Oct. 22, 2008]

[11] PHYTEC America, "PhyCORE-MPC555 and PhyCORE-MPC565," 1996. [Online]. Available: http://www.phytec.com/. [Accessed: June. 15, 2008]

[12] The Mathworks, "FM5 Target Support Package," MPC5xx Target Support Package, 1998. [Online]. Available: http://www.mathworks.com. [Accessed: Feb. 1, 2008]

[13] Xsens, "MTi-G," Xsens," 3DOF orientation sensors, 2000. [Online]. Available: http://www.xsens.com/. [Accessed: July 10, 2008]

[14] D. L. Heiserman, "SRF04 Ultrasonic Ranger," Robot Electronics, Jan. 2001. [Online]. Available: www.robot-electronics.co.uk/htm/srf04tech.htm. [Accessed: Nov 25, 2007]

[15] ATMEL, "Atmega16 Microcontroller," ATMEL Microcontrollers, March, 2002. [Online]. Available: http://www.atmel.com. [Accessed: Aug. 1, 2007]

[16] The CMUcam Vision Sensors, "CMUcam3," Carnegie Mellon University, May 2002. [Online]. Available: http://www.cs.cmu.edu/. [Accessed: Feb. 10, 2009]

[17] A. manries, "Simulation RC," 2001. [Online]. Available: http://simulrc.am.free.fr. [Accessed: April 28, 2008]

[18] Microdrones GmbH, "Cockpit GCS," Jan. 2005. [Online]. Available: http://www.microdrones.com. [Accessed: July 20, 2008]

[19] R. W. Beard, "Quadrotor Dynamics and Control," M. S. thesis, Brigham Young University, Provo, UT, 2008.

[20] M. F. Pettersen, "Nonlinear Control Approach to Helicopter Autonomy," M. S. thesis, Aalborg university, Aalborg, Denmark, 2005.

[21] K. S. Bjørn, "Autonomous hover flight for a quadrotor helicopter," M. S. thesis, Aalborg University, Aalborg, Denmark, 2007.

[22] K. J. Åström, *Control System Design*, 3rd ed. California, United States of America: Prentice Hall, 2002.

[23] Y. M. Alyounes and M. A. Jarrah, "Adaptive Integral Backstepping Controller for an Autonomous Rotorcraft," in *proceeding of the 6th International Conference on Mechatronics and its Applications, ISMA'09*, Sharjah, UAE, 2009.

[24] S. M. Deb, "Three-Dimensional offline Path Planning for UAVs Using Multiobjective Evolutionary Algorithms," *IEEE Congress on Evolutionary Computation*, pp. 3195-3202, 2007.

[25] D. H. Baker, *Computer Graphics*, 3rd ed. Upper Saddle River, NJ, United States of America: Prentice-Hall, 1996.

APPENDICES

APPENDIX A

MICROCONTROLLER CODES

A.1 *Onboard System Microcontrollers*

Using C-MEX s-function builder in Simulink, the following codes are generated.

A.1.1 MPC555 Microcontroller Codes

MTi-G IMU/GPS communication protocol

```
/* this code is generated to read the MTi -G IMU/GPS communication protocol and store it
in an array, then send it for conversion */

// Declaration
unsigned char i,j;
unsigned int checksum=0;

Protocol[0]=250;

// Initialization
if(Indx>=79) {
     Indx=0;
     FLG1=0;
     FLG2=0;

     // Cehcksum
     for(i=1;i<79;i++) {
         checksum = checksum + Protocol[i];
     }
     checksum = checksum & 0x00FF;

if(checksum == 0) {
    // Store the data in a protocol array
    ax[0] = Protocol[4];
    ax[1] = Protocol[5];
    ax[2] = Protocol[6];
    ax[3] = Protocol[7];
    ay[0] = Protocol[8];
    ay[1] = Protocol[9];
    ay[2] = Protocol[10];
    ay[3] = Protocol[11];
    az[0] = Protocol[12];
    az[1] = Protocol[13];
    az[2] = Protocol[14];
    az[3] = Protocol[15];

    p[0] = Protocol[16];
    p[1] = Protocol[17];
    p[2] = Protocol[18];
    p[3] = Protocol[19];
    q[0] = Protocol[20];
    q[1] = Protocol[21];
    q[2] = Protocol[22];
    q[3] = Protocol[23];
    r[0] = Protocol[24];
    r[1] = Protocol[25];
    r[2] = Protocol[26];
    r[3] = Protocol[27];

    roll[0]  = Protocol[40];
    roll[1]  = Protocol[41];
    roll[2]  = Protocol[42];
    roll[3]  = Protocol[43];
    pitch[0] = Protocol[44];
    pitch[1] = Protocol[45];
    pitch[2] = Protocol[46];
    pitch[3] = Protocol[47];
    yaw[0]   = Protocol[48];
    yaw[1]   = Protocol[49];
    yaw[2]   = Protocol[50];
    yaw[3]   = Protocol[51];

    Lat[0] = Protocol[52];
    Lat[1] = Protocol[53];
    Lat[2] = Protocol[54];
    Lat[3] = Protocol[55];
    Lon[0] = Protocol[56];
    Lon[1] = Protocol[57];
    Lon[2] = Protocol[58];
    Lon[3] = Protocol[59];
    Alt[0] = Protocol[60];
    Alt[1] = Protocol[61];
    Alt[2] = Protocol[62];
    Alt[3] = Protocol[63];

    Vx[0] = Protocol[64];
    Vx[1] = Protocol[65];
    Vx[2] = Protocol[66];
    Vx[3] = Protocol[67];
    Vy[0] = Protocol[68];
    Vy[1] = Protocol[69];
    Vy[2] = Protocol[70];
    Vy[3] = Protocol[71];
    Vz[0] = Protocol[72];
    Vz[1] = Protocol[73];
    Vz[2] = Protocol[74];
    Vz[3] = Protocol[75];

    Counter[0] = Protocol[76];
    Counter[1] = Protocol[77];
   }
}

// Cehck the starting bit of the message to start the storing process

if(GCS[0]==250) FLG1=1;

if((GCS[0]==255)&&(FLG1=1)) FLG2=1;

if((NUM[0]==1)&&(FLG1==1)&&(FLG2==1)) {
     Protocol[Indx+1]=GCS[0];
     Indx++;
}
```

Data Conversion

```
/* this code is responsible of converting the IEEE754 single floating point format to d e-
cimal number*/

#include "IEEE_754.h"
#include "IEEE_754.c"

// Angular Rate Conversion
P[0] = IEEE_to_Decimal(p[50], p[150], p[250],p[350]) ;
Q[0] = IEEE_to_Decimal(q[50], q[150], q[250],q[350]) ;
R[0] = IEEE_to_Decimal(r[50], r[150], r[250],r[350]) ;

// Angle Conversion
Roll[0]  = IEEE_to_Decimal(roll[50], roll[150], roll[250],roll[350]) ;
Pitch[0] = IEEE_to_Decimal(pitch[50], pitch[150], pitch[250],pitch[350]) ;
Yaw[0]   = IEEE_to_Decimal(yaw[50], yaw[150], yaw[250],yaw[350]);

// Velocity Conversion
Vx_Dec[0] = IEEE_to_Decimal(Vx_IEEE[50],Vx_IEEE[150], Vx_IEEE[250], Vx_IEEE[350]) ;
Vy_Dec[0] = IEEE_to_Decimal(Vy_IEEE[50],Vy_IEEE[150], Vy_IEEE[250], Vy_IEEE[350]) ;
Vz_Dec[0] = IEEE_to_Decimal(Vz_IEEE[50],Vz_IEEE[150], Vz_IEEE[250]) ;

// Position Conversion
Lat[0]  = IEEE_to_Decimal(lat[50], lat[150], lat[250],lat[350]) ;
Lon[0]  = IEEE_to_Decimal(lon[50], lon[150], lon[250],lon[350]) ;
Alt[0]  = IEEE_to_Decimal(alt[50], alt[150], alt[250],alt[350]) ;
```

IEEE_754.h

```
double IEEE_to_Decimal(short ,short ,short ,short);
```

IEEE_754.c

```
double IEEE_to_Decimal(short A,short B,short C,short D) {

    // Declaration
    long N;
    short Vir_sign,sign;
    long Vir_exp;
    double exponent;
    long Vir_mant;
    double mantissa;
    double IEEE_D;

    //Check the sign
    Vir_sign = A & 0x80;
    if(Vir_sign == 0) sign=1;
    else sign=-1;

    //Find the exponent
    Vir_exp = A*0x100 + B;
    Vir_exp = Vir_exp >> 7;
    Vir_exp = Vir_exp & 0xFF;
    exponent = (double)Vir_exp;
    exponent = pow(2,(exponent-127));

    //Calculate the Fraction
    Vir_mant = B*0x10000 + C*0x100 + D;
    Vir_mant = Vir_mant & 0x7FFFFF;
    Vir_mant = Vir_mant << 1;
    mantissa = (double)Vir_mant;
    mantissa = (mantissa/0x1000000)+1;

    //Find IEEE754 Decimal number
    IEEE_D = sign * exponent * mantissa;

    return IEEE_D;

}
```

Adaptive Integral Backstepping Controller

```
/* this code is responsible of controlling the quadrotor using AIBC*/

//////////////////////////////////////////////////////////////////////////
// Declaration
//////////////////////////////////////////////////////////////////////////

double e1,e2,c1=0.049,c2=1333.28,lammda1=0.000513,gamma1=0.01,limit1=2.0,KI1;
double e3,e4,c3=0.049,c4=1333.28,lammda2=0.000513,gamma2=0.01,limit2=2.0,KI2;
double e5,e6,c5=0.049,c6=1000.00,lammda3=0.000000,gamma3=0.01,limit3=0.5,KI3;
double e7,e8,c7=1.5,c8=5,lammda4=0,gamma4=0,limit4=2,u1,KI4;


//////////////////////////////////////////////////////////////////////////
//Altitude, Z  //////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////
e7 = z_d[0] - z[0];
if((e7>1)||(e7<-1)) e7=0;

KI4 = c7*lammda4*(e7 + int_e7);
if((KI4<=limit4)&&(KI4>=-limit4)) int_e7 = e7 + int_e7 ;
else if(KI4>0) KI4=limit4;
else KI4=-limit4;

// at start
if(K[0]==0) int_e7=0;

e8 = c7*e7 + + (KI4/c7)+ z_dot_d[0]- Vz[0];

u1 =(-m/(cos(phi[0])*cos(theta[0])))*((1-c7*c7+lammda4)*e7 - KI4 - g + (c7+c8)+
    c8*e8+ gamma4*e8_int[0] + z_ddot_d[0]);

if(u1<20) u1=25;
if(u1>30) u1=37;
U1[0]=u1;


//////////////////////////////////////////////////////////////////////////
//ROLL
//////////////////////////////////////////////////////////////////////////
```

```
e1 = phi_d[0] - phi[0];

//check the occurance of the error jumping for more than 70 degree
if((e1>1.2)||(e1<-1.2)) e1=0;

// integral part thresholding
KI1 = c1*lammda1*(e1 + int_e1);
if((KI1<=limit1)&&(KI1>=-limit1)) int_e1 = e1 + int_e1 ;
else if(KI1>0) KI1=limit1;
else KI1=-limit1;

// at start
if(K[0]==0) int_e1=0;

e2 = c1*e1 + phi_dot_d[0] + (KI1/c1) - p[0];

U2[0]=(Ix/l)*((1-c1*c1+lammda1)*e1+(c1+c2)*e2 - KI1 + gamma1*e2_int[0]
       +phi_ddot_d[0]+ q[0]*(Jr/Ix)*omega_r[0]-q[0]*r[0]*(Iy-Iz)/Ix);


///////////////////////////////////////////////////////////////////////////
//PITCH:
///////////////////////////////////////////////////////////////////////////

e3 = theta_d[0] - theta[0];

//check the occurance of the error jumping for more than 70 degree
if((e3>1.2)||(e3<-1.2)) e3=0;

// integral part thresholding
KI2 = c3*lammda2*(e3 + int_e3);
if((KI2<=limit2)&&(KI2>=-limit2)) int_e3 = e3 + int_e3 ;
else if(KI2>0) KI2=limit2;
else KI2=-limit2;

// at start
if(K[0]==0) int_e3=0;

e4 = c3*e3 + theta_dot_d[0] + (KI2/c3) - q[0];

U3[0]=(Iy/l)*((1-c3*c3+lammda2)*e3+(c3+c4)*e4- KI2 +  gamma2*e4_int[0]
       +theta_ddot_d[0] + p[0]*(Jr/Iy)*omega_r[0]-p[0]*r[0]*(Iz-Ix)/Iy);


///////////////////////////////////////////////////////////////////////////
//YAW
///////////////////////////////////////////////////////////////////////////
e5 = psi_d[0] - psi[0];

//check the occurance of the error jumping for more than 70 degree
 if((e5>3.14)||(e3<-3.14)) e3=0;

// integral part thresholding
KI3 = c5*lammda3*(e5 + int_e5);
if((KI3<=limit3)&&(KI3>=-limit3)) int_e5 = e5 + int_e5 ;
else if(KI3>0) KI3=limit3;
else KI3=-limit3;

// at start
if(K[0]==0) int_e5=0;

e6 = c5*e5 + psi_dot_d[0] + (KI3/c5) - r[0];
```

```
U4[0]=Iz*((1-c5*c5+lammda3)*e5+(c5+c6)*e6 - KI3
+/*gamma3*e6_int[0]+psi_ddot_d[0]*/-p[0]*q[0]*(Ix-Iy)/Iz);

///////////////////////////////////////////////////////////////////////////
```

Control Inputs to Forces

```
/* this code is converting the control inputs of the AIBC to the appropriate forces of
the rotors using nested saturation process*/

// Declaration
double F1,F2,F3,F4;
double k=1; //assume k=1 so we will not devide by k in each formula.

F1=(U1[0]/4)+(U3[0]/2)-(U4[0]/4);
f1[0]=F1;

F2=(U1[0]/4)-(U2[0]/2)+(U4[0]/4);
f2[0]=F2;

F3=(U1[0]/4)-(U3[0]/2)-(U4[0]/4);
f3[0]=F3;

F4=(U1[0]/4)+(U2[0]/2)+(U4[0]/4);
f4[0]=F4;
```

ECEF to ENU coordinates

```
 /* this code is converting the position data from ecef coordinate to enu coordinate for
position controller */

// Declaration
double phiP,lambda;

// Reference Point
if(ref_point[0] <= 1.5) {
    Xr = Xe[0];
    Yr = Ye[0];
    Zr = Ze[0];
}

// Conversion Process
phiP = atan2(Zr,sqrt((Xr*Xr)+(Yr*Yr)));
lambda = atan2(Yr,Xr);

e[0] = -sin(lambda)*(Xe[0]-Xr) + cos(lambda)*(Ye[0]-Yr);
n[0] = -sin(phiP)*cos(lambda)*(Xe[0]-Xr) - sin(phiP)*sin(lambda)*(Ye[0]-Yr) +
       cos(phiP)*(Ze[0]-Zr);
u[0] = cos(phiP)*cos(lambda)*(Xe[0]-Xr) + cos(phiP)*sin(lambda)*(Ye[0]-Yr) +
       sin(phiP)*(Ze[0]-Zr);
```

Ground Control Station Communication Protocol

```
/* this code is designed to transmit the communication protocol of the GCS by converting
the decimal number to a special ASCII code format . */

// Declaration
```

```
double V_Roll,V_Pitch,V_Yaw,CS;
int INT,i;
unsigned char Checksum = 0;


// Initialization

if(Tx==0) {
    V_Roll  = Roll[0];
    V_Pitch = Pitch[0];
    V_Yaw   = Yaw[0];

//Communication Protocol Format
    Seven[0] = 35;
    Seven[1] = 55;
    Seven[2] = 44; Seven[9] = 44; Seven[16] = 44; Seven[24] = 44;
    Seven[5] = 46; Seven[12] = 46; Seven[20] = 46;

    //Roll [3,4,6, 7 and 8]
    //Roll_Sign
    if(V_Roll<0) {
        Seven[3] = 45;
        V_Roll = V_Roll*(-1);
    }
    else Seven[3] = 48;

    //R1
    INT = (int)V_Roll;
    Seven[4] = INT + 0x30;

    //R2
    V_Roll = (V_Roll - INT)*10;
    INT = (int)V_Roll;
    Seven[6] = INT + 0x30;

    //R3
    V_Roll = (V_Roll - INT)*10;
    INT = (int)V_Roll;
    Seven[7] = INT + 0x30;

    //R4
    V_Roll = (V_Roll - INT)*10;
    INT = (int)V_Roll;
    Seven[8] = INT + 0x30;

    //Pitch [10,11,13,14 and 15]
    //Pitch_Sign
    if(V_Pitch<0) {
        Seven[10] = 45;
        V_Pitch = V_Pitch*(-1);
    }
    else Seven[10] = 48;

    //P1
    INT = (int)V_Pitch;
    Seven[11] = INT + 0x30;

    //P2
    V_Pitch = (V_Pitch - INT)*10;
    INT = (int)V_Pitch;
    Seven[13] = INT + 0x30;

    //P3
```

```
V_Pitch = (V_Pitch - INT)*10;
INT = (int)V_Pitch;
Seven[14] = INT + 0x30;

//P4
V_Pitch = (V_Pitch - INT)*10;
INT = (int)V_Pitch;
Seven[15] = INT + 0x30;


//Yaw [17-23]
//Yaw_Sign
if(V_Yaw<0) {
    Seven[17] = 45;
    V_Yaw = V_Yaw * (-1);
}
else Seven[17] = 48;

//Y1
Seven[18] = 48;

//Y2
INT = (int)V_Yaw;
Seven[19] = INT + 0x30;

//Y3
V_Yaw = (V_Yaw - INT)*10;
INT = (int)V_Yaw;
Seven[21] = INT + 0x30;

//Y4
V_Yaw = (V_Yaw - INT)*10;
INT = (int)V_Yaw;
Seven[22] = INT + 0x30;

//Y5
V_Yaw = (V_Yaw - INT)*10;
INT = (int)V_Yaw;
Seven[23] = INT + 0x30;

//CheckSum
for(i=0;i<25;i++)
    Checksum = Checksum + Seven[i];

Checksum = ~Checksum;

CS = Checksum/100;
INT = (int)CS;
Seven[25] = INT + 0x30;

CS = Checksum/10;
CS = CS - (INT*10);
INT = (int)CS;
Seven[26] = INT + 0x30;

CS = (Checksum%10);
INT = (int)CS;
Seven[27] = INT + 0x30;

//CR+LF
Seven[28]=13;
Seven[29]=10;
```

```
    Tx = 1;
}

else {

    //2-byte Output format
    if(flag < 15) {
        GCS[0] = Seven[2*flag];
        GCS[1] = Seven[2*flag+1];
        flag++;
    }
    else {
        flag=0;
        GCS[0] = ' ';
        GCS[1] = ' ';
        Tx = 0;
    }
}
```

Receiving Data from the MOS

```
/* this code is responsible of analyzing the communication protocol coming from the
ground system through the MOS*/

// Declaration

char data;
double Thrust,Yaw,f1,f2,f3,f4;
double Thrust_d,Roll_d,Pitch_d,Yaw_d;
double Ver;

data = RC_In[0];

// Number of Received Channels

if(data == 1)
    RC_Flag = 1;
else if(data == 2)
    RC_Flag = 2;
else if(data == 3)
    RC_Flag = 3;
else if(data == 4)
    RC_Flag = 4;
else if(data == 5)
    RC_Flag = 5;
else if(data == 6)
    RC_Flag = 6;
else if(data == 7)
    RC_Flag = 7;
else if(data == 8)
    RC_Flag = 8;
else if(data == 9)
    RC_Flag = 9;
else
    RC_Array[RC_Flag-1] = data;
```

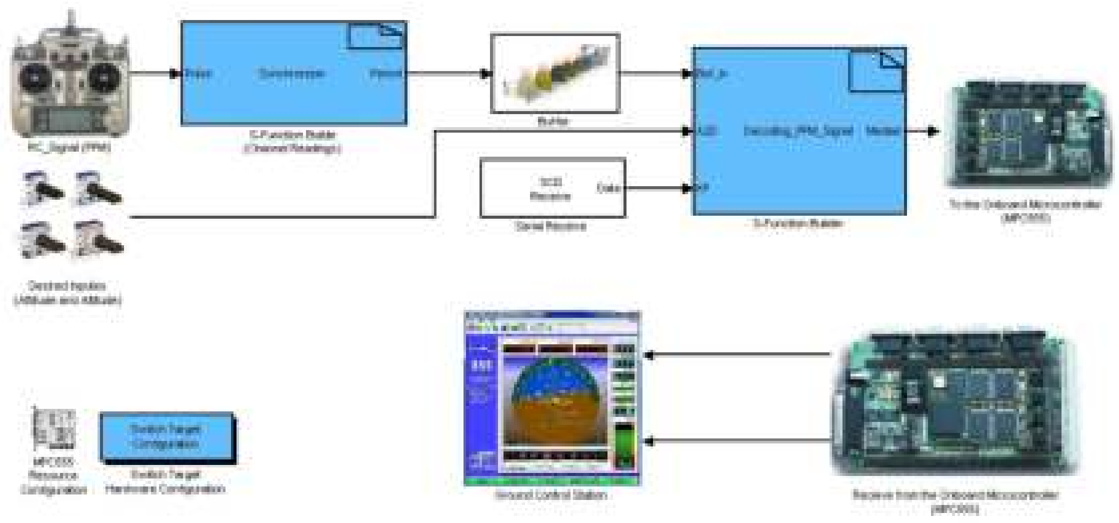
```
//Manual Override System data

MOR[0] =  ((RC_Array[4]-10)/240);
VC[0] =  ((RC_Array[5]-10)/120);
Trim_Roll[0]  = 15-((RC_Array[0]-10)/8);
Trim_Pitch[0] = ((RC_Array[1]-10)/8)-15;

// //Forces
Ver = (double) RC_Array[2];
Thrust = ((Ver-10)/240);

Ver = (double) RC_Array[3];
Yaw = ((Ver-10)/120);

Ver = (double) RC_Array[0];
f2 = ((Ver-10)/24);
Ver = f2*Thrust*Yaw;
RC_F2[0] = Ver;
f4 = 10 - f2;
Ver = f4*Thrust*Yaw;
RC_F4[0] = Ver;

Ver = (double) RC_Array[1];
f1 = ((Ver-10)/24);
Ver = f1*Thrust*(2-Yaw);
RC_F1[0] = Ver;
f3 = 10 - f1;
Ver = f3*Thrust*(2-Yaw);
RC_F3[0] = Ver;


// Desired Inputs

Roll_d  = (double) (((RC_Array[6]-10)*90)/240);
Desired_Inputs[0] = Roll_d;

Pitch_d = (double) (((RC_Array[7]-10)*90)/240);
Desired_Inputs[1] = Pitch_d;

Yaw_d   = (double) (((RC_Array[8]-10)*90)/240);
Desired_Inputs[2] = Yaw_d;
```

A.1.2 ATmega16 Microcontroller Codes

```c
// this code is written to capture the position data from the CMUcam3

// ICC-AVR application builder : 5/5/2009 12:01:58 PM
// Target : M16
// Crystal: 1.0000Mhz

#include <iom16v.h>
#include <macros.h>

void port_init(void)
{
 DDRA  = 0xFF;
 DDRB  = 0xFF;
 DDRC  = 0xFF;
 DDRD  = 0xFF;
}

//TIMER1 initialize - prescale:1
// WGM: 14) PWM fast, TOP=ICRn
// desired value: 3906Hz
// actual value: 3906.250Hz (0.0%)
void timer1_init(void)
{
 TCCR1B = 0x00; //stop
 TCNT1H = 0xFF; //setup
 TCNT1L = 0xEE;
 OCR1AH = 0x00;
 OCR1AL = 0x12;
 OCR1BH = 0x00;
 OCR1BL = 0x12;
 ICR1H  = 0x00;
 ICR1L  = 0xFF;
 TCCR1A = 0xA2;
 TCCR1B = 0x19; //start Timer
}

//UART0 initialize
// desired baud rate: 4800
// actual: baud rate:4808 (0.2%)
void uart0_init(void)
{
 UCSRB = 0x00; //disable while setting baud rate
 UCSRA = 0x00;
 UCSRC = BIT(URSEL) | 0x06;
 UBRRL = 0x0C; //set baud rate lo
 UBRRH = 0x00; //set baud rate hi
 UCSRB = 0x18;
}

// Transmit
void TransmitByte( unsigned char data )
{while ( !(UCSRA & (1<<UDRE)) );UDR = data;}
// Receive
unsigned char ReceiveByte(void)
```

```
{while ( !(UCSRA & (1<<RXC)) ) {;} return UDR;}

//call this routine to initialize all peripherals
void init_devices(void)
{
 //stop errant interrupts until set up
 CLI(); //disable all interrupts
 port_init();
 timer1_init();
 uart0_init();

 MCUCR = 0x00;
 GICR  = 0x00;
 TIMSK = 0x00; //timer interrupt sources
 SEI(); //re-enable interrupts
 //all peripherals are now initialized
}

void main(void)
{
    unsigned char R,MSG[8];
    int i,s1,s2,x,y;
    init_devices();

    TransmitByte('T');TransmitByte('C');TransmitByte(' ');
    TransmitByte('0');TransmitByte('0');TransmitByte('
    ');TransmitByte('3');TransmitByte('5');TransmitByte(' ');
    TransmitByte('2');TransmitByte('0');TransmitByte('
    ');TransmitByte('5');TransmitByte('5');TransmitByte(' ');
    TransmitByte('0');TransmitByte('0');TransmitByte('
    ');TransmitByte('4');TransmitByte('0');TransmitByte(13);
    while (1)
    {
          s1=0;s2=0;x=0;y=0;
          while (ReceiveByte()!='T');R=ReceiveByte();
          for (i=0;i<8;i++) {MSG[i]=ReceiveByte();}

          for (i=0;i<8;i++)
              {
               if ( (MSG[i]==' ') && (s1!=0)  && (s2==0)) {s2=i;}
               if ( (MSG[i]==' ') && (s1==0) ) {s1=i;}
               MSG[i]=MSG[i]-48;
              }

          if (s1==3) {x=MSG[0]*100+MSG[1]*10+MSG[2];}
          if (s1==2) {x=MSG[0]*10+MSG[1];}
          if (s1==1) {x=MSG[0];}

          s2=s2-s1-1;
          if (s2==3) {y=MSG[s1+1]*100+MSG[s1+2]*10+MSG[s1+3];}
          if (s2==2) {y=MSG[s1+1]*10+MSG[s1+2];}
          if (s2==1) {y=MSG[s1+1];}
          OCR1A=x;OCR1B=y;
    }
}
```

A.2 Ground System Microcontroller (MPC565)



Decoding the PPM signal

```
// In this code, the PPM signal is decoded from the RC transceiver

// Declaration
unsigned long Ch1,Ch2,Ch3,Ch4,Ch5,Ch6,Ch7,Ch8;

Ch1 = Buf_In[50];
Ch2 = Buf_In[150];
Ch3 = Buf_In[250];
Ch4 = Buf_In[350];
Ch5 = Buf_In[450];
Ch6 = Buf_In[550];
Ch7 = Buf_In[650];
Ch8 = Buf_In[750];

// Thresholding
if(Ch1 < 885) Ch1 = 885;
else if(Ch1 > 1908) Ch1 = 1908;
CH[0]=Ch1;

if(Ch2 < 885) Ch2 = 885;
else if(Ch2 > 1908) Ch2 = 1908;
CH[1]=Ch2;

if(Ch3 < 885) Ch3 = 885;
else if(Ch3 > 1908) Ch3 = 1908;
CH[2]=Ch3;

if(Ch4 < 885) Ch4 = 885;
else if(Ch4 > 1908) Ch4 = 1908;
CH[3]=Ch4;

if(Ch5 < 885) Ch5 = 885;
else if(Ch5 > 1908) Ch5 = 1908;
```

```
CH[4]=Ch5;

if(Ch6 < 885) Ch6 = 885;
else if(Ch6 > 1908) Ch6 = 1908;
CH[5]=Ch6;

if(Ch7 < 885) Ch7 = 885;
else if(Ch7 > 1908) Ch7 = 1908;

if(Ch8 < 885) Ch8 = 885;
else if(Ch8 > 1908) Ch8 = 1908;


// Output the decoded signal using certain Protocol

if(Indx1==Indx2) {
    Indx2++;
    Median[0] = Indx2;
}
else {
    if(Indx2<7)
        Median[0] = (unsigned char)(((CH[Indx1]-885)/4.2625)+10);
    else if(Indx2==9)
        Median[0] = KP[0];
    else
        Median[0] = A2D[Indx1-6];
    Indx1++;
}

if(Indx1>=9) {
    Indx1 = 0;
    Indx2 = 0;
}
```

Detecting the Channels sequence on the PPM signal (Synchronizer)

```
// this code is for detecting the channels sequence in the PPM signal, this is
// done by capturing the synchronization pulse as discussed in chapter 2.

unsigned long PW,N;
char ver;

//save the current value of pulse width
PW = Pulse[0];

//check the synchronize width and put its value in Channel[0]
// if(PW >= 3750) {
if(PW >= 2750) {
    Flag = 1;
    Index = 0;
}

if(Flag == 1) {
    Channel[Index] = PW;
    if(Index != 0)        //check the repeated value
        if(Channel[Index] == Channel[Index-1])
            Index--;
    Index++;
}

// Channels Readings
if(Index > 8) {
    Flag = 0;

    Period[0] = Channel[1] ;
    Period[1] = Channel[2] ;
    Period[2] = Channel[3] ;
    Period[3] = Channel[4] ;
    Period[4] = Channel[5] ;
    Period[5] = Channel[6] ;
    Period[6] = Channel[7] ;
    Period[7] = Channel[8] ;
}
```

APPENDIX B

MANUAL AND DATASHEETS

B.1 Motor Datasheet

RS-545SH

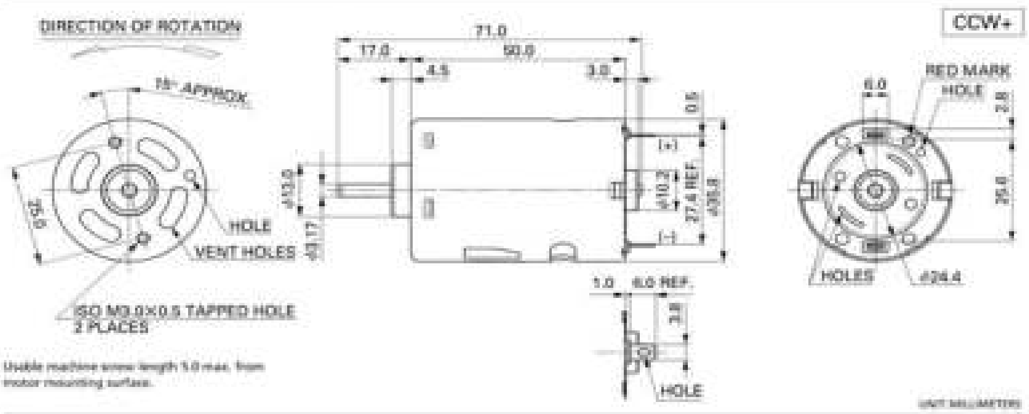
MABUCHI MOTOR
Carbon-brush motors

OUTPUT : 5.0W~75W (APPROX) WEIGHT : 156g (APPROX)

Typical Applications **Automotive Products :** Automatic Cruise Control
Home Appliances : Hair Dryer / Massager / Vibrator / Vacuum Cleaner / Bilge Pump

MODEL		VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
		OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
				r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
RS-545SH-5018	(*1)	4.5~12.0	12V CONSTANT	24000	1.30	20510	7.65	31.1	317	66.7	214	2181	45.0
RS-545SH-18150	(*1)	12~32	30V CONSTANT	7000	0.13	5720	0.58	18.2	185	10.9	99.2	1011	2.59

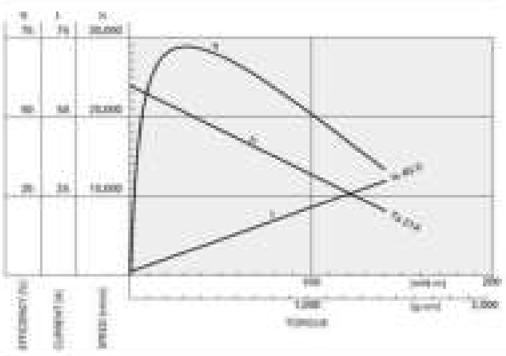
(*1) CCW shifted commutation (CCW+)
The terminal position against the tapped holes varies depending on CW+NEUTRAL.



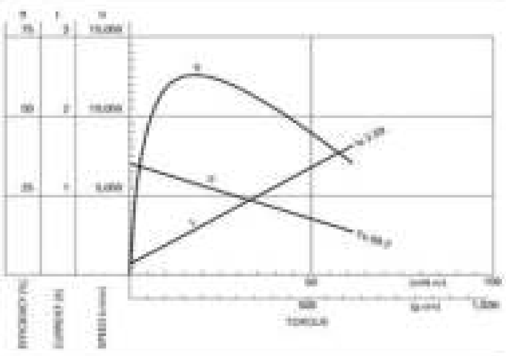
Usable machine screw length 5.0 max. from motor mounting surface.

UNIT: MILLIMETERS

RS-545SH-5018 12.0V



RS-545SH-18150 30.0V



B.2 Charging of Li-Po Batteries Manual

This material copied from Thunder Power website [9]

General Guidelines and Warnings

1. Thunder Power batteries are NOT charged as you receive them. They contain approximately 50% of a full charge.
2. Use Lithium Polymer specific chargers only. Do not use a NiCd or NiMh charger - Failure to do so may cause a fire, which may result in personal injury and property damage.
3. Never charge batteries unattended. When charging LiPo batteries you should always remain in constant observation to monitor the charging process and react to potential problems that may occur.
4. Some LiPo chargers on the market may have technica l deficiencies that may cause them to charge LiPo batteries incorrectly. It is solely the responsibility of the user to assure that the charger used works properly. Thunder Power only recommends chargers and balancers made by Thunder Power, other brands may work but are out of Thunder Power's control.
5. If at any time you witness a battery starting to balloon or swell up, discontinue the charging process immediately. Disconnect the battery and place it in a safe observation area for approximately 15 minutes. Continuing to charge a battery that has begun to swell will result in fire.
6. Battery observation should occur in a safe area outside of any building or vehicle and away from any combustible material. The middle of a cement driveway is a good example of a safe observation area.
7. Shorts can cause fires! If you accidentally short the wires, the battery must be placed in a safe area for observation for approximately 15 minutes. Additionally, be mindful of the burn danger that may occur due to a short across jewelry (such as rings on your fingers).
8. Chemical reactions are not instantaneous; a battery that has been shorted may not ignite for 10 minutes.

9. All crash batteries, even if not deformed, should be placed in a safe area for observation for at least 15 minutes
10. If for any reason you need to cut the terminal wires, cut each wire separately, ensuring the wires do not become shorted across the cutting tool.
11. When soldering connectors, first place a short length of heat shrink tubing over each wire. Then remove the insulating tape from the red wire and strip a short length of the insulation off, exposing the conductor approximately ¼". Tin the exposed wire as well as the connector terminals. Place the wire in contact with the positive connector terminal and re -flow the solder of both together. Once cool, slide the heat shrink tubing down to cover the joint and shrink. Repeat the process for the black wire. If you accidentally short the battery wires, place the battery in a sa fe area and observe it for approximately 15 minutes.
12. Never store or charge a battery pack inside your car if the internal temperature will exceed 120 degrees

Before the First Charge

1. Make a visual inspection of the pack. Checking for any damaged leads, connectors, broken/cracked shrink covering, puffiness or other irregularities.
2. Before installing or changing the connector, check the voltage of the pack using a digital voltmeter (not your charger). All new packs ship at approximately 3.80V to 3.9V per cell.

   For example: A 2S pack should read approximately 7.60V to 7.8V, A 3S pack should read approximately 11.40V to 11.7V.
3. If any damage to the pack or leads is found, or the voltage is significantly less for your pack than specified above, do not attempt to charge or fly the pack; contact Thunder Power directly as soon as possible.

Charging Process

1. Never charge batteries unattended.
2. Charge in an isolated area, away from flammable materials.
3. Let the battery cool down to ambient temperature before charging.

4. Do not charge battery packs in series except as outlined in step 8. Charge each battery pack individually. Overcharging of one or the other battery may occur resulting in fire.

   In order to discharge packs in series, the charged voltage of each cell in both packs must be within 0.01V

5. When selecting the cell count or voltage for charging purposes, select the cell count and voltage as it appears on the battery label. Selecting a cell count or voltage other than the one printed on the lab el may result in overcharging and fire. As a safety precaution, please confirm that the information printed on the battery is correct.

   For example: If a battery label indicates that it is a 3 cell battery (3S), its voltage should read between 11.4 and 11.7 volts. This battery must be charged as a 3 cell battery (peak of 12.6V).

6. You must check the pack voltage after each flight before re-charging. Do not attempt to charge any pack if the unloaded individual cell voltages are less than 3.3V.

   For example: Do not charge a 2-cell pack if below 6.6V, Do not charge a 3 cell pack if below 9.9V

7. NORMAL CHARGING: The charge rate should not exceed 1C (one times the capacity of the battery, unless otherwise noted*). Higher setting may cause problems which can result in fire.

   For example: Charge a 730mAh battery at or below 0.73Amps. Charge a 5000mAh battery at or below 5Amps.

   Thunder Power packs with balancing connectors can be used with TP balancers for safer charging. For more information, please visit: www.thunderpowerrc.com

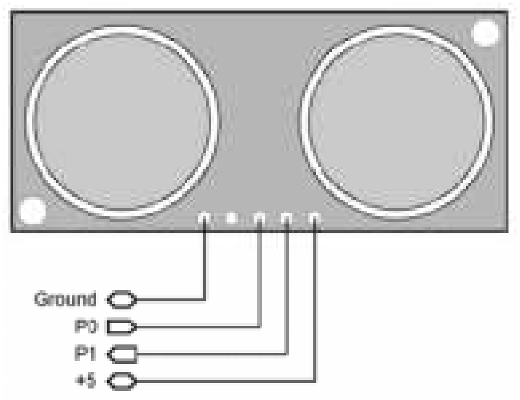
   *To charge at greater than 1C (no more than 3C): You must use a Thunder Power 1010C charger in conjunction with a Thunder Power Balancer (205 or 210) and data cable. Only ProLite 910, 1320, 2000 and 2100 cells qualify for charging up to 3C.

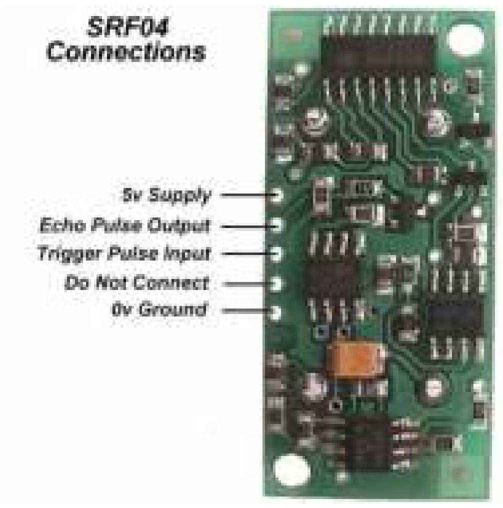
8. To charge two packs in series: The packs need to first be charged individually (using a 1010C, 210V balancer and associated data cable), and flown in series for a couple of cycles. Then, having flown both packs together in series, using a good quality DVM, check the individual cell voltages at the balancing connector. If all the voltages are within 0.01V of each other, series charging should be safe. Please note that

this requires a “Y” cable be made to electrically attach the packs together in series and that the battery on the negative most side of this cable (the lead that goes to the negative terminal of the charger) be attached to “group A” of the balancer. Please see 1010C/210V instructions.

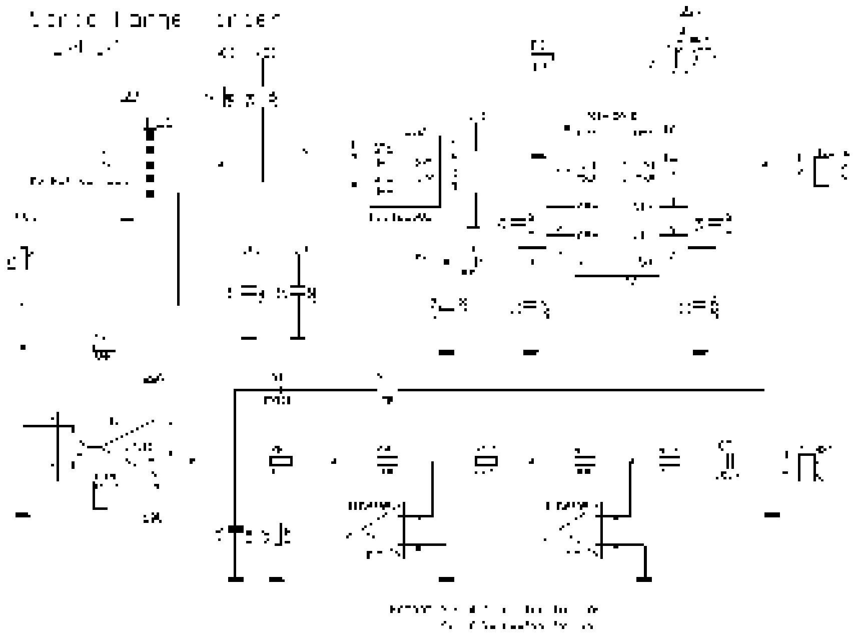
B.3 SRF04 Ultrasonic Sensor Datasheet

Connection Pinout

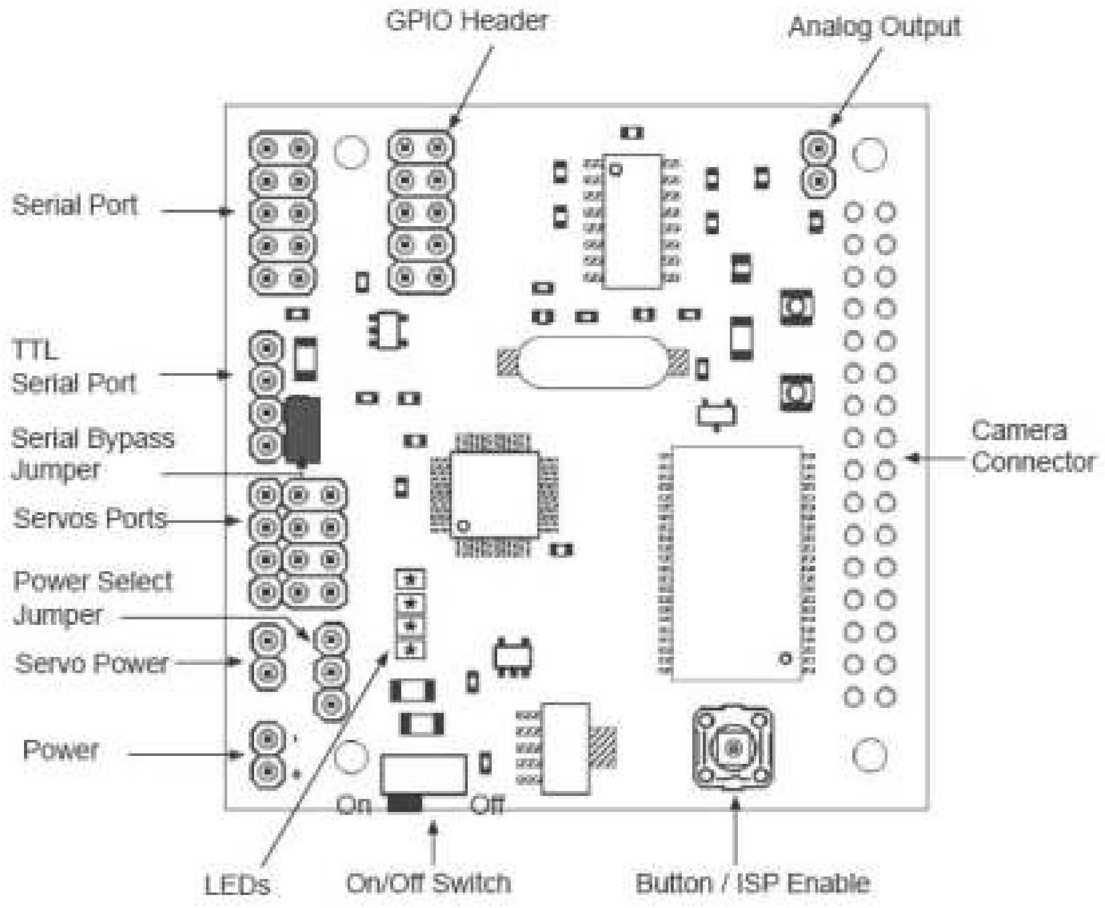




Schematic Diagram



B.4 CMUcam3 Hardware Connections



APPENDIX C

ALGORITHMS AND PROTOCOLS

C.1 MTi-G Message Identifier

Calibrated data output mode (36 bytes)
Contains the calibrated data output of the accelerations, rate of turn and magnetic field in X, Y and Z axes in floats.

accX	accY	accZ	gyrX	gyrY	gyrZ	magX	magY	magZ

Orientation data output mode – Euler angles (12 bytes)
Contains the three Euler angles, in floats, that represent the orientation of the MT

roll	pitch	yaw

Position data output mode – LLA (12 bytes)
Contains the latitude, longitude and altitude, in floats, that represent the position of the MTi-G.

Lat	Lon	Alt

Velocity data output mode – VelXYZ (12 bytes)
Contains the velocity X, Y and Z, in, that represent the velocity of the MTi-G. Note that velocity in North East Down can be obtained by changing bit 31 in the `SetOutputSettings` DATA field.

Vel_X	Vel_Y	Vel_Z

C.2 Path Generation Using B-Spline Curves Method ***[24]***

These are the most widely used class of approximating splines. B-splines have two advantages over Bézier splines:

1. The degree of a B-spline polynomial can be set independently of the number of control points (with certain limitations), and
2. B-splines allow local control over the shape of a spline curve or surface the trade-off is that B-splines are more complex than Bézier splines.

The coordinates of the free-to-move control points of the B-Spline curve are the variables used in our algorithm. Each control point in 2-D space is represented using two values, one each along the *(X , Y)* axis. Thus, if there are *n* free-to-move control points, then a solution will involve *2n* real-parameter variables, as a two-tuple for each control point in a successive manner, as shown below:

$$(\underbrace{(X_1, Y_1)}_{P_1}, \underbrace{(X_2, Y_2)}_{P_2}, \ldots, \underbrace{(X_n, Y_n)}_{P_n})$$

While computing the entire path from start to end, the start and end points are added in the above list at the beginning and at the end, respectively, and the B-Spline formulation is used to get a mathematical formulation of the entire path for the UAV.

Computing B-Spline Curves from Control Points

B-Spline curves are parametric curves, and their constructions are based on blending functions [25]. Suppose the number of control points of the curve is (n + 1), the coordinates being (x_0 , y_0), . . . , (x_n , y_n), then the coordinates of the B-Spline curves are given by:

$$X(t) = \sum_{i=0}^{n} x_i B_{i,k}(t),$$

$$Y(t) = \sum_{i=0}^{n} y_i B_{i,k}(t),$$

Where, $B_{i,K}(t)$ is a blending function of the curve and *K* is the order of the curve. Higher the order of the curve, smoother the curve is. For example, see Figure (1), (2) and (3), shown for different order *K* of the curve. The parameter *t* varies between zero and *(n - K + 2)* with a constant step, providing the discrete points of the B-spline curve.

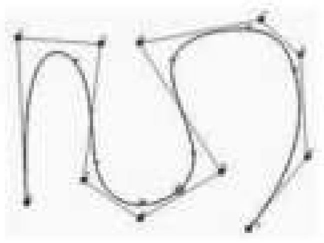


Figure 1 : B-Spline curves with 11 control points and order 3

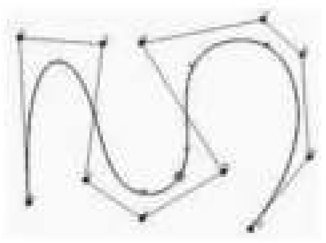


Figure 2 : B-Spline curves with the same set of 11 control points and order 5

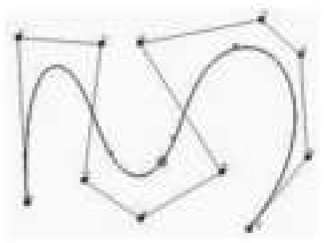


Figure 3 : B-Spline curves with the same set of 11 control points and order 7

For our problem, the blending functions have been defined recursively in terms of a set of *Knot* values, which in this case is a *uniform non-periodic one*, defined as:

$$Knot(i) = \begin{cases} 0, & \text{if } i < K \\ i - K + 1, & \text{if } K \leq i \leq n \\ n - K + 2, & \text{if } n < i \end{cases}$$

The blending function $B_{i,K}$ is defined by the Cox-deBoor recursion formulas, using the *Knot* values given above:

$$B_{i,1}(t) = \begin{cases} 1, & \text{if } Knot(i) \leq t < Knot(i+1) \\ 1, & \text{if } \begin{cases} Knot(i) \leq t \leq Knot(i+1) \\ \text{and} \\ t = n - K + 2 \end{cases} \\ 0, & \text{otherwise} \end{cases}$$

$$B_{i,K}(t) = \frac{(t - Knot(i)) \times B_{i,K-1}(t)}{Knot(i+K-1) - Knot(i)} + \frac{(Knot(i+K) - t) \times B_{i+1,K-1}(t)}{Knot(i+K) - Knot(i+1)}$$

If the denominator of either of the fractions is zero, that fraction is defined to be zero.

Properties of the B-spline curves:

- The polynomial curve has degree k - 1 and C^{k-2} continuity over the range of t.
- For $n+1$ control points, the curve is described with $n+1$ blending functions.
- Each blending function $B_{i,K}$ is defined over k subintervals of the total range of t, starting at *knot* value t_i.
- The range of parameter t is divided into $n + k$ subintervals by the $n+k+1$ values specified in the *knot* vector.
- With *knot* values labeled as $\{t_0, t_1, \ldots, t_{n+k}\}$, the resulting B-spline curve is defined only in the interval from *knot* value t_{k-1} up to *knot* value t_{n+1}.
- Each section of the spline curve (between two successive *knot* values) is influenced by k control points.
- Any one control point can affect the shape of at most k curve sections.

In addition, a B-spline curve lies within the convex hull of at most $k + 1$ control points, so that B-splines are tightly bound to the input positions. For any value of t in the interval from *knot* value t_{k-1} to *knot* value t_{n+1} the sum over all basis functions is 1:

$$\sum_{i=0}^{n} B_{i,k}(t) = 1$$

Given the control-point positions and the value of parameter k, we then need to specify the *knot* values to obtain the blending functions using the Cox-deBoor recursion formulas.

MATLAB CODE

b-spline.m

```
function spline(n,order)

% function spline(n,order)
%
% Plots the B-slpine-curve of n control-points.The control points can be
% chosen by clicking with the mouse on the figure.
%
% COMMAND:  spline(n,order)
% INPUT:    n       Number of Control-Points
%           order   Order of B-Splines
%                   Argnument is arbitrary
%                   default: order = 4

n=5;
close all;
if (nargin ~= 2)
    order = 4;      % you can change the order of b-spline. Consequently the
                    % the curve will be smoother .
end
nplot = 100;

if (n < order)
    display([' !!! Error: Choose n >= order=',num2str(order),' !!!']);
    return;
end

figure(1);
axis([0 100 0 100]);
hold on; box on;
set(gca,'Fontsize',16);


t = linspace(0,1,nplot);

for i = 1:n
    title(['Choose ',num2str(i),' th. control point']);
    p(i,:) = ginput(1);
    hold off;
    plot(p(:,1),p(:,2),'k-','LineWidth',2);
    axis([0 100 0 100]);
    hold on; box on;
    if (i  >= order)
        T = linspace(0,1,i-order+2);
        y = linspace(0,1,1000);
        p_spl = DEBOOR(T,p,y,order);
        plot(p_spl(:,1),p_spl(:,2),'b-','LineWidth',4);
    end
    plot(p(:,1),p(:,2),'ro','MarkerSize',10,'MarkerFaceColor','r');
end
title(['B-Spline-curve  with  ',num2str(n),'  control  points  of  order
',num2str(order)]);

%save the B-spline curve
S = struct('field1',p_spl)
save BSPLINE -struct S

end
```

DEBOOR.m

```
function val = DEBOOR(T,p,y,order)

% function val = DEBOOR(T,p,y,order)
%
% INPUT:  T         Knot
%         p         Control Inputs (nx2-Matrix)
%         y         Evaluated points
%         order     Order of B-Splines

p
m = size(p,1);
n = length(y)
X = zeros(order,order);
Y = zeros(order,order);
a = T(1);
b = T(end);
T = [ones(1,order-1)*a,T,ones(1,order-1)*b];


for l = 1:n
    t0 = y(l);
    id = find(t0 >= T);
    k = id(end);
        if (k > m)
            return;
        end
    X(:,1) = p(k-order+1:k,1);
    Y(:,1) = p(k-order+1:k,2);

    for i = 2:order
        for j = i:order
            num = t0-T(k-order+j);
            if num == 0
                weight = 0;
            else
                                s = T(k+j-i+1)-T(k-order+j);
                weight = num/s;
            end
            X(j,i) = (1-weight)*X(j-1,i-1) + weight*X(j,i-1);
            Y(j,i) = (1-weight)*Y(j-1,i-1) + weight*Y(j,i-1);
        end
    end
    val(l,1) = X(order,order);
    val(l,2) = Y(order,order);
end
```

Execution of the m-files:

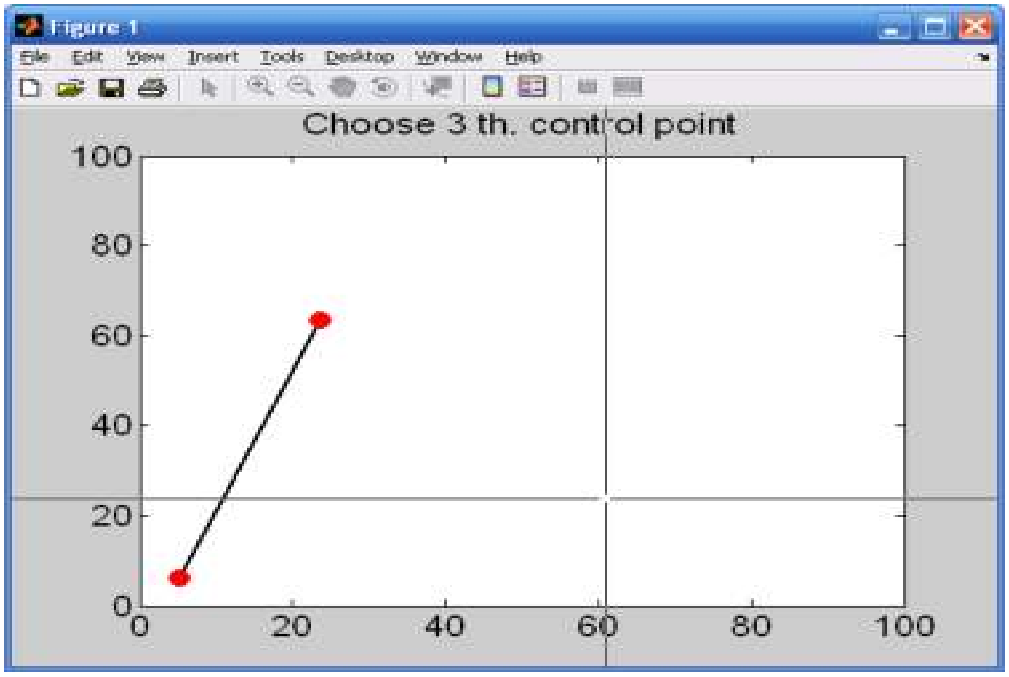


Figure 1
File Edit View Insert Tools Desktop Window Help
Choose 3 th. control point
100
80
60
40
20
0
0
20
40
60
80
100

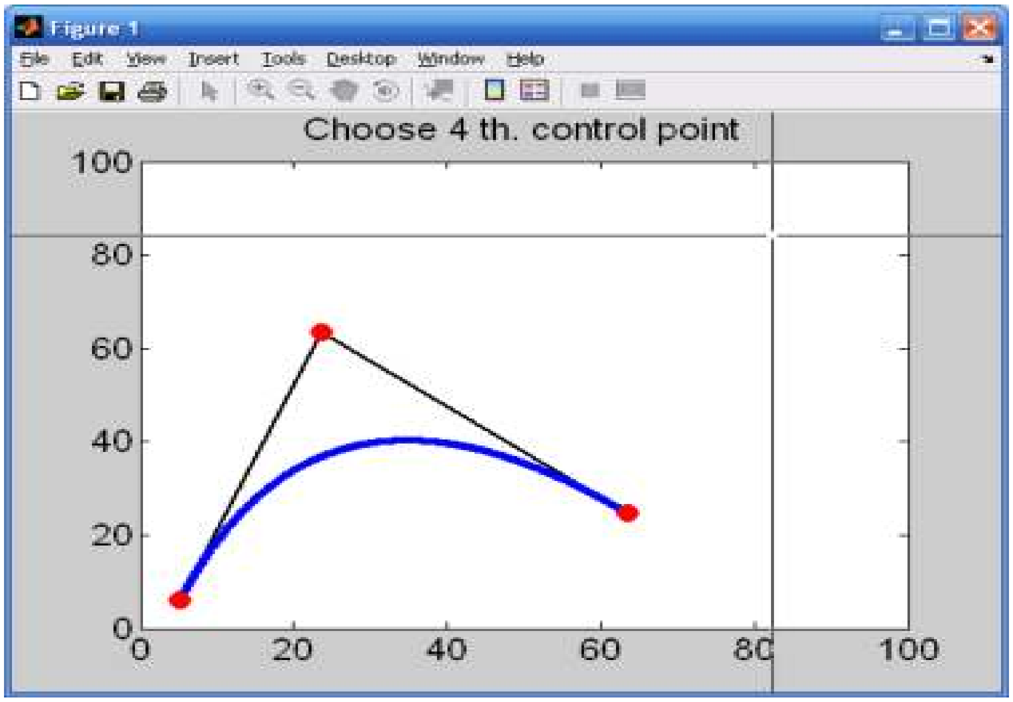


Figure 1
File Edit View Insert Tools Desktop Window Help
Choose 4 th. control point
100
80
60
40
20
0
0
20
40
60
80
100

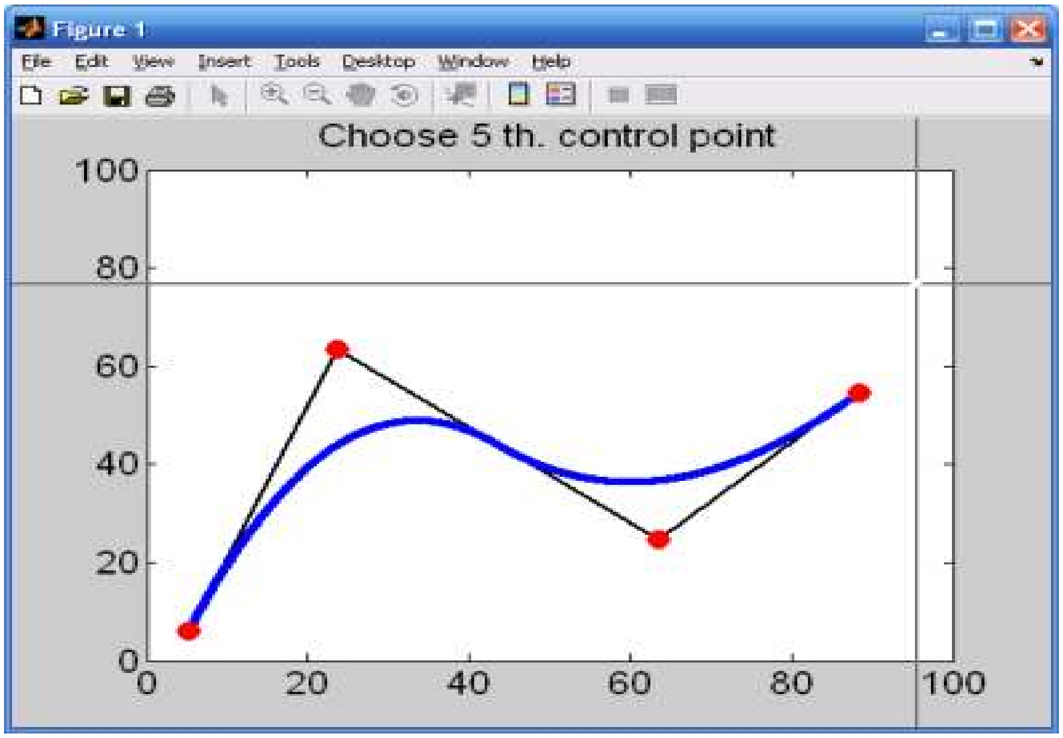


Figure 1
File Edit View Insert Tools Desktop Window Help
Choose 5 th. control point
100
80
60
40
20
0
0
20
40
60
80
100

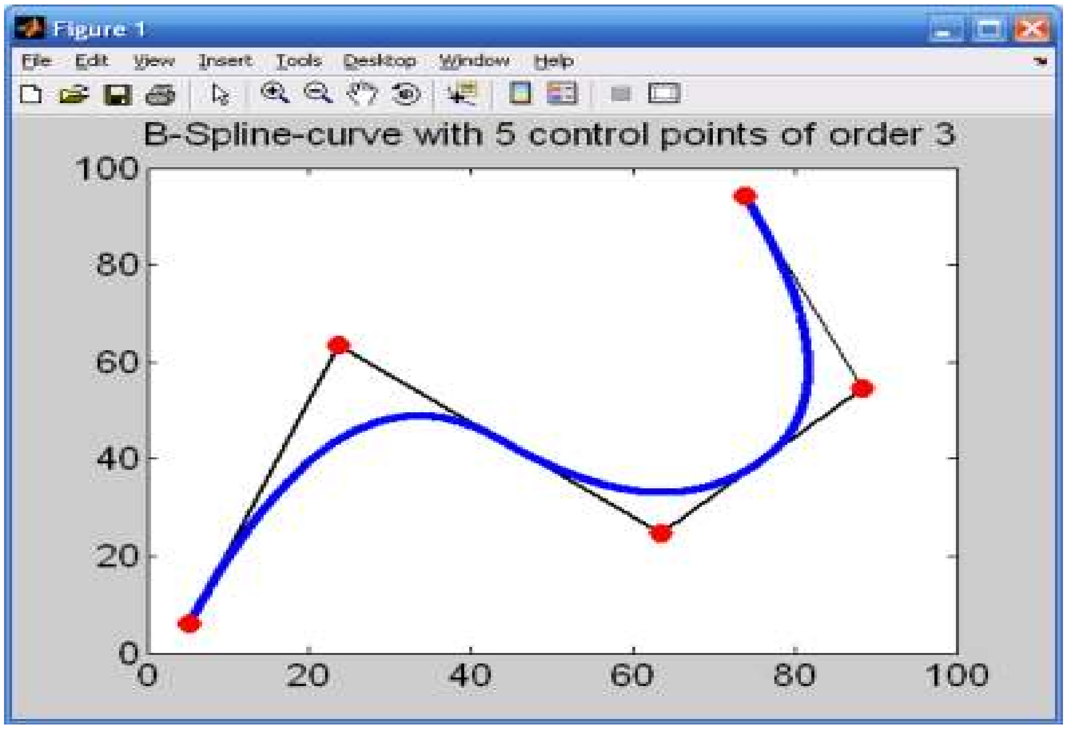


Figure 1
File Edit View Insert Tools Desktop Window Help
B-Spline-curve with 5 control points of order 3
100
80
60
40
20
0
0
20
40
60
80
100

These figures show the process of generating a b-spline curve from a given way-points (control inputs).

VITA

Younes Alyounes was born on September 9, 1983, in Dammam, Kingdom of Saudi Arabia, KSA. At the age of thirteen, he moved to Jordan and finished his secondary school there, after that he got his Bachelor of Science degree in Mechanical/Mechatronics at Jordan University of science and Technology (JUST). Mr. Alyounes was ranked the first in the class among one-hundred and two students with an honor and excellent percentage average (87.6%), beside that, he ranked the sixth on the Jordan kingdom in the achievement exam for the mechanical engineering students. After getting his bachelor degree, he joined Consolidated Contractor Company, CCC where he acted as a Mechanical engineer specialist in welding for one year.

Mr. Alyounes began a Master of Scienece degree in Mechatronics at AUS in 2007 and awarded the Master of Science degree in 2009 with a 3.71 GPA. Since July, 2007 he has been working as a Teaching Assistant at AUS for two years, and now he joined the higher college of technology (HCT) as a lab engineer.