A PROJECT REPORT
ON

"CONTROL SYSTEM DESIGN OF A QUADROTOR USING PIXHAWK4 CONTROLLER"

Submitted to

COLLEGE OF ENGINEERING PUNE

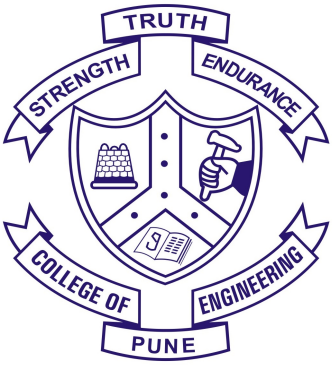
In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN ELECTRICAL ENGINEERING

BY

ONKAR MURKEWAR	**111505030**
NACHIKET PANSE	**111505031**
AJINKYA PHANSE	**111505035**
TEJAS RAJOPADHYE	**111505039**

UNDER THE GUIDANCE OF
Dr. Shailaja Kurode
D.R.D.O. Guide :- Mr. Jayawant Kolhe



DEPARTMENT OF ELECTRICAL ENGINEERING
COLLEGE OF ENGINEERING PUNE
SHIVAJINAGAR, PUNE - 411005

2018-2019

College Of Engineering Pune

Department of Electrical Engineering
Wellesley Road, Shivajinagar, Pune 411005

CERTIFICATE

This is certify that the project entitled

"Control System Design of Quadrotor using Pixhawk 4 Controller"

submitted by

Onkar Murkewar	**111505030**
Nachiket Panse	**111505031**
Ajinkya Phanse	**111505035**
Tejas Rajopadhye	**111505039**

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Electrical Dept.) at College Of Engineering, Pune under the University of Pune. This work is done during year 2018-2019, under our guidance.

Date: / /

(Dr. Shailaja Kurode)
Project Guide

(Mr. Jaywant Kolhe)
Project Guide, D.R.D.O.

(Dr. Sanjay Dambare)
HOD, Electrical Department

Acknowledgements

We are profoundly grateful to **Dr. Shailaja Kurode** for her expert guidance and continuous encouragement throughout to see that this project succeeds in its target since its commencement to its completion.

We would like to express deepest appreciation towards **Dr. Sanjay Dambare**, Head of Department of Electrical Department, COEP and **Mr. Jaywant Kolhe**, Project Guide, D.R.D.O whose invaluable guidance supported us in completing this project.

Onkar Murkewar
Nachiket Panse
Ajinkya Phanse
Tejas Rajopadhye

ABSTRACT

Quadrotors are UAVs - Unmanned aerial vehicles - which can develop complex motions because of its six degrees of freedom. Recently, they are used in many fields like - transportation, military,etc. It is not easy to control a quadrotor due to its highly nonlinear dynamics, variable coupling and model uncertainties. When the motors of the quadrotors are controlled properly various different maneuvers are possible.

This report presents the development of mathematical modeling, control techniques, simulation and real-time testing on a developed quadrotor as an unmanned aerial vehicle. Modeling of the dynamic system of a quadrotor including the motor dynamics is carried out using Newton-Euler Method. Using this model a Proportional,integral, Derivative (PID) control is developed. a first-order Sliding Mode Control (SMC) is also developed as a nonlinear robust control technique. Using the developed mathematical model and the two control techniques viz- the Proportional-Integral-Derivative (PID), and SM Control, simulations are run on MATLAB with and without the presence of external disturbances. Multiple simulations testing are performed to ensure the adequacy of the proposed control techniques using MATLAB. A novel feature of this project is the integration of PX4 toolchain with MATLAB 2018b. Using this toolchain, a MATLAB Embedded Coder Support For PX4 Autopilot simulation model can be converted into Pixhawk based target files giving the control system designer freedom in developing a customized control law for the quadrotor.

Finally, hardware design of a quadrotor has been developed and implemented. During flight, on-board control optimization and actuation based on real time data from various on-board sensors is provided by an embedded processor, Pixhawk controller. Pixhwak software supports multi-threading which gives it an edge over other controllers at performing various tasks. Various sensors like gyroscope, acclerometer are inbuilt and available in varied configurations with the hardware to give accurate angles and directions. Results of real-time flight tests using the PID controller are presented and a model is proposed for trajectory tracking. At the end, recommendations and possible future work on this subject is given.

Keywords: Newton-Euler, PID, SMC, Pixhawk

Contents

List of Figures

List of Tables

List of Abbreviations

BEC	Battery Eliminating Circuit
BLDC	Brushless Direct Current
CAN	Controller Area Network
CCW	Counter Clock Wise
CW	Clock Wise
DSM2/DSMx	Digital System Multiplexer
ESC	Electronic Speed Controller
FMU	Flight Management Unit
FOSM	First Order Sliding Mode
FPV	First Person View
GPS	Global Positioning System
I2C	Inter Integrated Circuit
LiPo	Lithium Polymer
mAh	milli-Ampere-hour
MAV-link	Micro Air Vehicle Communications Protocol
PCB	Printed Cicuit Board
PCM	Pulse Code Modulation
PID	Proportional Integral Derivative

PPM	Pulse Position Modulation
PSP	Pilot Support Package
PWM	Pulse Width Modulation
RC	Remote Controller
RPM	Revolutions Per Minute
RSSI	Receiver Signal Strength Indicator
RX	Receiver
S.Bus	Serial Bus
SMC	Sliding Mode Control
SPI	Serial Peripheral Interface
TX	Transmitter
UART	Universal Asynchronous Receiver Transmitter
UAV	Unmanned Ariel Vehicle
VTOL	Vertical Take-Off and Landing
WSL	Wondows System for Linux

Nomenclature

ϕ	Roll angle
ψ	Yaw angle
θ	Pitch angle
C_M	Torque Coefficient
C_T	Thrust Coefficient
c_θ	cosθ

g	Gravitational acceleration
m	Quadcopter's mass
s_θ	sinθ
T_ϕ	Torque provided by motors in roll direction
T_ψ	Torque provided by motors in yaw direction
T_θ	Torque provided by motors in pitch direction
U_i	Control inputs of motor,i= 1,2,3,4
w_i	Speeds of motor,i= 1,2,3,4
I_{xx}	Moment of Inertia of quadcopter around X axis
I_{yy}	Moment of Inertia of quadcopter around Y axis
I_{zz}	Moment of Inertia of quadcopter around Z axis

Chapter 1

Introduction

Vertical take-off and landing (VTOL) flying machines have evolved greatly in the past century and have attracted great attention from researchers in different disciplines due to their capabilities and vast variety of missions they can do as compared to fixed-wing Unmanned Aerial Vehicles (UAV). Out of many reasons that attracted researchers and engineers towards VTOL planes, some of them are:

1. Very little space is needed for take off and landing
2. Increased portability as compared to a fixed wing aircraft
3. VTOL drones, especially smaller drones are more suitable for aerial photography than fixed wing aircrafts

It is not easy to control a quadrotor due to its highly nonlinear dynamics, variable coupling and model uncertainties. The under actuation property of the quadrotor also adds another degree of complexity to the model due to limited availability of control techniques that can be applied to under actuated systems. This thesis presents the development of mathematical modeling, control techniques, simulation and real-time testing on a developed quadrotor.

Quadrotors are one-of-a-kind UAVs (Unmanned aerial vehicles) capable of executing complex maneuvers owing to its six degrees of freedom. Because of their low noise and quick movement they find a lot of applications in military, transportation and surveillance. In order to get quick movement, control algorithms with small settling times and higher accuracy are required. This thesis explores robust control for the quadrotor, simulations of the control algorithms and practical validation using an embedded processor. The control algorithm is proof tested with the actual quadcopter,

interfaced with MATLABs Embdded Coder Support For PX4 Autopilots which provides various Simulink actuator blocks to do real time testing. After the algorithm is tested, it is used as a firmware for the quadrotor and actual flight is performed with Pixhawk hardware at its core.

1.1 Introduction to Quadcoptor

A quadrotor consists of four motors which are symmetrically placed in the shape of a X. Four motors having four command inputs, which enable the quadrotor to have six degrees of freedom. The adjacent motors of the quadrotor spin in opposite fashion. As shown in Figure 1.1, the left and front motors rotate in opposite direction to maintain stability. This helps the quadrotor to maintain stability about Z axis. The shape of the blades attached to the quadrotor are designed such that rotational motion provided by the motor produces downward thrust which counteracts the effect of gravity. Thus, with the help of rotating blades, quadrotor is able to maintain its position and altitude. There are various states in which the quadrotor manoeuvres like hovering, throttle, roll, pitch and yaw motion. The conventional movement in X, Y and Z co-ordinates is a result of these manoeuvres. The assumptions made here are :-

1. On the quadrotor, the motors are symmetrically placed.
2. Thrust produced by the quadcopter is directly proportional to the motor's speed square.
3. The quadcopter is a rigid structure and is placed in a perfect X configuration.

Following these assumptions the working for various states are:

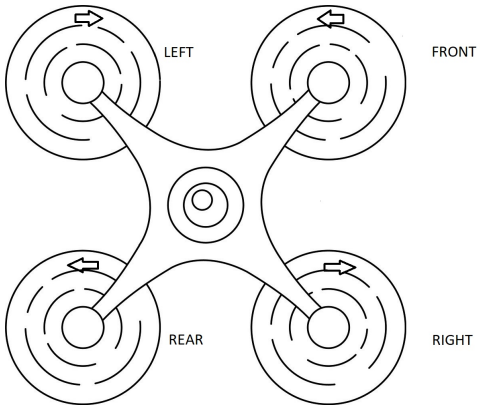


Figure 1.1: Quadcopter

Hovering- When all the motors of the quadrotor rotate at the same speed such that the thrust produced as a result counterbalances gravity, there is no motion about the Z

axis.This is called hovering.

Throttle- With an increase or decrease in the speed of the motors, such that thrust produced is more than gravitational force , thus producing acceleration , a motion in Z direction is achieved. By increasing/decreasing more or less thrust is produced. This causes movement in Z direction called throttle.

Roll Motion- When speed of left rotors is increased and speed of right rotors is decreased, roll angle changes. This is called a roll maneuver. The speeds of rotors need to be adjusted such that the overall thrust remains same, thus, maintaining the altitude constant.

Pitch Motion- When speed of front rotors is increased and speed of rear rotors is decreased equally or vice versa, then pitch angle changes leading to a pitch maneuver. The rotors altitude doesnt decrease as the net downward thrust is maintained constant.

Yaw Motion- When speed of left rotor is decreased and speed of front rotor is increased or vice versa , it causes yaw motion. This also needs to be accompanied with an exact same increase/decrease in diagonally opposite rotors so that the pitch or roll angles do not change.

The forward and backward movement is caused by the adjustments in the speed of front and rear rotors which causes movement along X- direction. In all of the above discussion, quadcopter 's size, symmetry and weight plays a major role in determining its on-flight time because, lesser weight accounts for lesser gravitational pull that needs to be countered in steady state and hence increases the battery life and, thus, the overall efficiency.

1.2 Literature Review

The state of the art in UAV with VTOL has changed drastically with immense number of contributions in this field. The first work on quadrotor was performed by Breguet and Richet in the year 1907.

Recently, quadrotors have attracted the interest of scientists and researchers because of a wide area of applications and its many advantages over other types of unmanned vehicles. Quadrotor VTOL machines have hovering capability, high maneuverability

and agility. If compared with standard helicopters, quadrotors have several advantages such as small size, efficiency, and safety. These advantages made quadrotors eligible for applications like military services, surveillance, rescue, research area, remote inspection, and photography. [1]

Most of the researchers focus on proposing a control algorithm or comparing the control laws of a few techniques. Some works have also been done in the design and modeling of a quadrotor [2]. There are different methods to model the dynamics of quadrotor. Methods used for modelling are Newton-Euler 's method and Lagrangian method. Most of the quadcopter designers use Newton-Euler equations to write the basic equations of motion and use Lagrangian method to derive the second order non-linear equation. The method described in [3] is Newton-Euler followed by Lagrangian equation to derive the final non-linear dynamics equations.

There are many control approaches that are considered.Linear Quadratic Regulator (LQR) is a method of control that has been proposed by a few, but a lot of computations are involved to solve Ricatti equation and moreover, it is not robust [4] .A control strategy using PID controller was proposed and used. However, strong disturbances, like presence of wind gusts, were poorly rejected [5]. Some papers have proposed a backstepping control strategy. In case of backstepping control though internal convergence of state was assured, a lot more computation was required and it might lead to explosion of states. Backstepping method was able to control the Euler angles but for relatively critical initial conditions. [6]

Improvement to backstepping control is integral backstepping which is a combination of PID and backstepping . This method guarantees asymptotic stability and has robustness to some uncertainties, while the steady state error is cancelled by integral action. In the paper, attitude, altitude and position control of a quadrotor was achieved. [7]. Sliding mode control strategy to control a 4-Y octarotor, which is an extension of a quadrotor, has been proposed in [8]. The mathematical model of the octarotor is developed based on which a sliding Mode Control law is synthesised and results of numerical simulations have been presented.

1.3 Motivation

Quadcopters are simple and robust VTOL - Vertical Take-Off and Landing- UAVs which are compact in size. In the past few years, quadcopters have gained popularity and have been used widely in different applications. The literature reveals the wide spectrum of applications and the necessity of a maneuverable vehicle. They can carry different payloads for varied tasks. Because of these things they find a lot of applications in military and transportation. In military domain, accurate sensing and actuation is required with fault tolerant capability. In transportation , the quadcopter is required to carry heavy weight and yet maintain stability. Another demanding application is surveillance, which requires the quadrotor to manoeuvre continuously in a particular fashion and quickly respond to any threats. In all these applications, robust control of quadcopter becomes crucial. This highlights the need for a robust and reliable controller. The dynamics of a quadcopter i.e. motions in roll, pitch and yaw are non-linear and coupled which makes the problem of control convoluted. This makes the design of control strategies challenging. This motivated us to perform analysis of PID as well as Sliding Mode Control to increase the accuracy and robustness.

1.4 Problem description

The quadrotor dynamics being non-linear, under-actuated and tightly coupled require us to build a compensation system that can handle unwanted disturbances and also control the dynamics. Various steps taken to control the attitude and the altitude of the quadrotor UAV, are :

1. Modeling of various motions like roll, pitch, yaw and altitude.
2. Design of control strategy for stabilisation and dynamics.
3. Validation using simulations.
4. Testing of control system with PSP and actual flight with quadcopter.

1.5 Report Layout

Chapter 1 gives a brief introduction to the thesis and to the quadrotor system. It then gives an idea related to the work done by various researchers in this field. This is followed by the motivation for taking up the project and problem description.

Chapter 2 presents the modeling of the quadcopter dynamics based on Newton-Euler formalism. In this chapter, a detailed derivation of equations balancing forces and moments is presented while clearly stating the assumptions.

Chapter 3 gives the details of the controllers developed. The first controller that is implemented is PID. Due to the short comings faced with PID controller, SMC is examined. the simulation results of both the controllers are shown. Disturbance is later fed into the model to account for outdoor phenomenon like wind in the z-direction, simulation is carried out and results are displayed.

Chapter 4 presents the hardware used in development of the vehicle. The flight controller and specifications are clearly stated along with other components and peripherals such as battery, motors, telemetry, Radio Control (RC) and Electronic Speed Controllers (ESC)

Chapter 5 explains about pixhawk 4 controller and matlab's pixhawk pilot support package and it's inplementation in the project

Finally, in chapter 6 the conclusion of the thesis work is presented with potential future work.

Chapter 2

Mathematical Modeling of Quadrotor

2.1 Introduction

To design the control algorithm to control the six degrees of freedom of motion of quadrotor, it is necessary to have mathematical equations containing speeds of four motors and all the forces acting on the quadrotor including gyroscopic effects.

For modeling of quadrotor following assumptions are made

1. The frame structure and propellers are considered rigid
2. Structure is assumed to be perfectly symmetrical

There are different modelling methods to model the quadrotor dynamics into mathematical equations like Newton-Euler's method and Lagrange's method. Method described in this thesis is Newton-Euler's method. Forces acting on the quadrotor body can be represented either by taking earth or quadcopter body centre as the origin, leading to two different frames called Earth fixed frame and body fixed frame respectively. So understanding of co-ordinate system is necessary.

2.2 Co-ordinate frames

2.2.1 Earth fixed frame

Earth fixed frame is the frame in which co-ordinate axes are fixed with respect to earth surface and its origin is home location. It is also called as inertial frame.[9]

2.2.2 Body fixed frame

Body fixed frame is fixed with respect to body and its origin is the body centre in the case of symmetrical bodies.

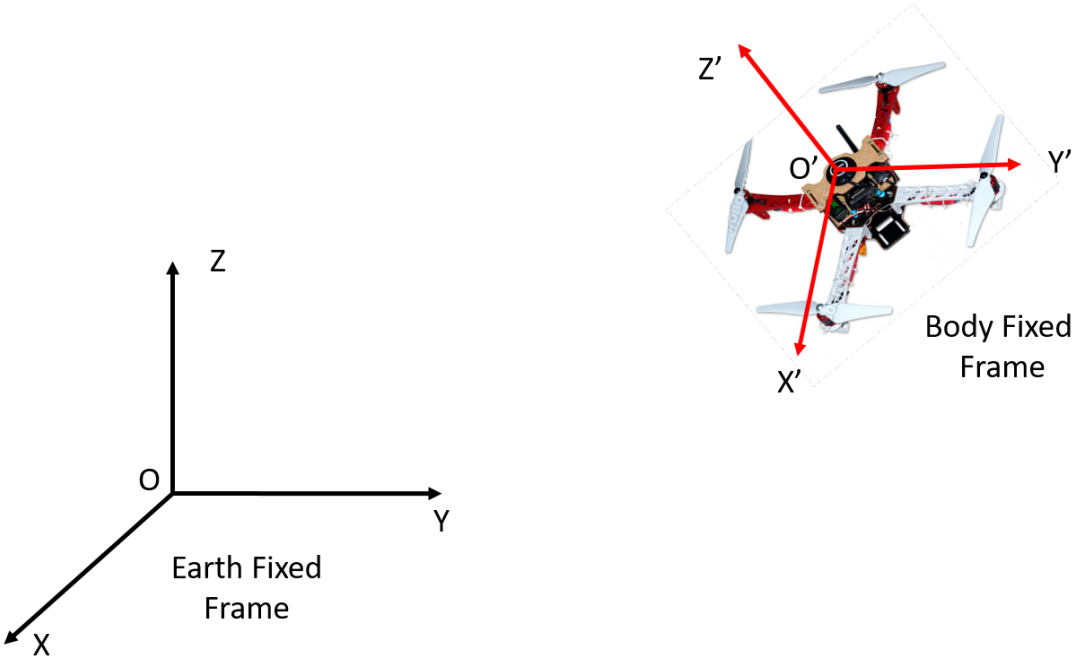


Figure 2.1: Frame

2.3 Rotation matrices

To convert the physical parameters of quadcopter from one frame to other frame following method is used.
Consider two dimensional plane having frame X-Y as an inertial frame and X'-Y'as a body fixed frame. Coordinates of point P in inertial frame are P(x,y) and in body fixed frame are P(x',y'). For the purpose of derivation lets assume origins of both the frames coincide.

As shown in figure (2.2) α is angle between inertial frame and body fixed frame and $\alpha + \beta$ is angle made by point P with X axis of inertial frame.

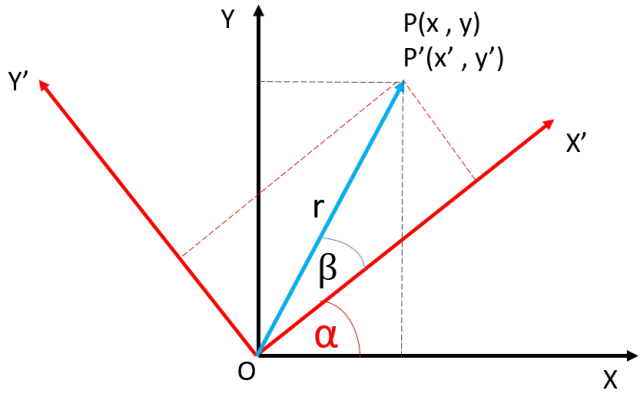


Figure 2.2: Rotation in 2D

r is the magnitude of vector OP, which will be same for both the coordinate frame.

From the figure (2.2), we can write

$$x' = r\cos\beta \tag{2.1}$$

$$y' = r\sin\beta \tag{2.2}$$

$$x = r\cos(\alpha+\beta) \tag{2.3}$$

$$y = r\sin(\alpha+\beta) \tag{2.4}$$

Solving above equations we get,

$$x = x'\cos\alpha - y'\sin\alpha \tag{2.5}$$

$$y = x'\sin\alpha + y'\cos\alpha \tag{2.6}$$

In matrix form it can be written as,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{2.7}$$

It can be written as,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.8}$$

Solving above equations we get,

$$x = x'\cos\alpha - y'\sin\alpha \tag{2.9}$$

$$y = x'\sin\alpha + y'\cos\alpha \tag{2.10}$$

In matrix form it can be written as,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{2.11}$$

It can also be written by taking inverse of rotation matrix as,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.12}$$

Considering three dimensonal co-ordinate system, three single orientations about Z,X,Y axis are described by:

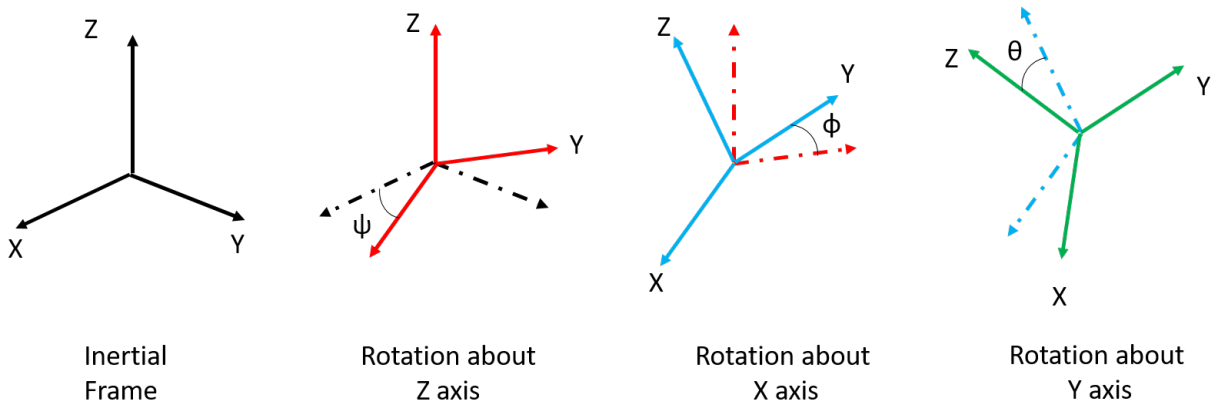


Figure 2.3: Rotation about x, y and z axes

$R(Z,\psi)$ =rotation about Z axis
$R(X,\phi)$ =rotation about X axis
$R(Y,\theta)$ =rotation about Y axis

They are represented by:

$$R(Z,\psi) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

$$R(X,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \tag{2.14}$$

$$R(Y,\theta) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \tag{2.15}$$

The complete rotation matrix is the product of the previous three successive rotations:

$$R(\psi,\phi,\theta) = R(Z,\psi)R(X,\phi)R(Y,\theta) \tag{2.16}$$

After matrix multiplication

$$R = \begin{bmatrix} c_\psi.s_\theta + s_\psi.s_\phi.s_\theta & -s_\psi.c_\phi & c_\psi.s_\theta - s_\psi.s_\phi.c_\theta \\ s_\psi.c_\theta + c_\psi.s_\phi.s_\theta & c_\psi.c_\phi & s_\psi.s_\theta - c_\psi.s_\phi.c_\theta \\ -c_\phi.s_\theta & s_\phi & c_\phi.c_\theta \end{bmatrix} \tag{2.17}$$

where $s_x = \sin x$ and $c_x = \cos x$

2.4 Newton-Euler equations of motion :

As rate of change of linear momentum is equal to the net force acting on the body,

$$F = \sum_{i=1}^{N} F_i = m\frac{\mathrm{d}}{\mathrm{d}t}({}^{A}V^{C}) \tag{2.18}$$

A represents quantities in inertial frame.[10]
${}^{A}V_C$ = Velocity of centre of mass in inertial frame.

$$F = \frac{\mathrm{d}}{\mathrm{d}t}({}^{A}L) \tag{2.19}$$

For Rotational motion, rate of change of angular momentum is equal to the net moment,

$${}^{A}\frac{\mathrm{d}}{\mathrm{d}t}({}^{A}H_C^B) = M_C^B \tag{2.20}$$

$${}^{A}H_C^B = I_C.{}^{A}\omega^B \tag{2.21}$$

${}^{A}H_C^B$ = Angular Momentum
I_C = Inertial tensor with centre of mass as origin
${}^{A}\omega^B$ = angular velocity of body in inertial frame in A
M_C^B = Net moment from all external forces and torques about the centre of mass
Net moment = rate of change of angular momentum.

$$M_c = \frac{\mathrm{d}}{\mathrm{d}t}({}^{A}H_c) \tag{2.22}$$

$${}^{A}\omega^B = pb_1 + qb_2 + rb_3 \tag{2.23}$$

To write the equation of motion in body fixed frame correction factor should be added

in equation (2.22)

$$\frac{\mathrm{d}}{\mathrm{d}t}({}^{B}H_C) + {}^{A}\omega^{B}XH_C = M_C \tag{2.24}$$

$\frac{\mathrm{d}}{\mathrm{d}t}(H_C^B)$ = differentiation in body fixed frame
${}^{A}\omega^{B}XH_C$ = correction factor

$$\frac{\mathrm{d}}{\mathrm{d}t}({}^{B}H_C) = I_{11}\dot{\omega}_1 b_1 + I_{22}\dot{\omega}_2 b_2 + I_{33}\dot{\omega}_3 b_3 \tag{2.25}$$

equation (2.24) can be written as,

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} + \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & -\omega_1 & 0 \end{bmatrix} \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} M_{C1} \\ M_{C2} \\ M_{C3} \end{bmatrix} \tag{2.26}$$

If $\ddot{r}$ is linear acceleration and F_1, F_2, F_3, F_4 are thrust produced by each motor of quadcopter, then by Newtons-Euler law, equation of motion can be written as

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \tag{2.27}$$

$m\ddot{r}$ = Components in the inertial frame
R = Rotational matrix from body to inertial frame
$\begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$ = body fixed frame forces

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \tag{2.28}$$

where R = Rotation matrix. On multiplying the rotational matrix and separating the equations we get ,

$$m\ddot{x} = (c_\psi . s_\theta - s_\psi . s_\phi . c_\theta)T \tag{2.29}$$

$$m\ddot{y} = (s_\psi . s_\theta - c_\psi . s_\phi . c_\theta)T \tag{2.30}$$

$$m\ddot{z} = c_\phi . c_\theta . T - mg \tag{2.31}$$

where T is total thrust provided by all rotors

equation (2.24) can be written as,

$$I\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}L(F_4-F_2)\\ L(F_3-F_1)\\ M_2+M_4-M_1-M_3\end{bmatrix} - \begin{bmatrix}p\\ q\\ r\end{bmatrix} \times I \begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{2.32}$$

L = Length m = Mass

Transformation matrix from $\dot{\phi},\dot{\theta},\dot{\psi}$ to body angular rates *p,q,r* is

$$\begin{bmatrix}p\\ q\\ r\end{bmatrix} = \begin{bmatrix}\cos\theta & 0 & -\cos\phi\sin\theta\\ 0 & 1 & \sin\phi\\ \sin\theta & 0 & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{2.33}$$

After matrix multiplication

$$\begin{bmatrix}p\\ q\\ r\end{bmatrix} = \begin{bmatrix}c_\theta.\dot{\phi}-\dot{\psi}.c_\phi.s_\theta\\ \dot{\theta}+\dot{\psi}.s_\phi\\ \dot{\phi}.s_\theta+\dot{\psi}.c_\phi.c_\theta\end{bmatrix} \tag{2.34}$$

Taking differentiation

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}-\dot{\theta}.\dot{\phi}.s_\theta+\ddot{\phi}.c_\theta-\ddot{\psi}.c_\phi.s_\theta+\dot{\psi}.\dot{\phi}.s_\phi.s_\theta-\dot{\psi}.\dot{\theta}.c_\phi.c_\theta\\ \ddot{\theta}+\ddot{\psi}.s_\phi+\dot{\psi}\dot{\phi}.c_\phi\\ \ddot{\phi}.s_\theta+\dot{\phi}\dot{\theta}.c_\theta+\ddot{\psi}.c_\phi.c_\theta-\dot{\psi}\dot{\phi}.s_\phi.c_\theta-\dot{\psi}\dot{\theta}.c_\phi.s_\theta\end{bmatrix} \tag{2.35}$$

Cross product of two matrices U and V can be written as

$$U\times V = \hat{U}V \tag{2.36}$$

Therefore

$$I\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}L(F_4-F_2)\\ L(F_3-F_1)\\ M_2+M_4-M_1-M_3\end{bmatrix} - \begin{bmatrix}\hat{p}\\ q\\ r\end{bmatrix} I \begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{2.37}$$

where,

$$\begin{bmatrix}\hat{p}\\ q\\ r\end{bmatrix} = \begin{bmatrix}0 & -r & q\\ r & 0 & -p\\ -q & p & 0\end{bmatrix} \tag{2.38}$$

$$\begin{bmatrix} \hat{p} \\ q \\ r \end{bmatrix} = \begin{bmatrix} 0 & -(\dot{\phi}.s_\theta + \dot{\psi}.c_\phi.c_\theta) & (\dot{\theta} + \dot{\psi}.s_\phi) \\ (\dot{\phi}.s_\theta + \dot{\psi}.c_\phi.c_\theta) & 0 & -(\dot{\phi}.c_\theta - \dot{\psi}.c_\phi.s_\theta) \\ -(\dot{\theta} + \dot{\psi}.s_\phi) & (\dot{\phi}.c_\theta - \dot{\psi}.c_\phi.s_\theta) & 0 \end{bmatrix} \tag{2.39}$$

putting equations 2.39, 2.34 , 2.35 in 2.37

$$\begin{bmatrix} I_{xx}(-\dot{\theta}.\dot{\phi}.s_\theta + \ddot{\phi}.c_\theta - \ddot{\psi}.c_\phi.s_\theta + \dot{\psi}.\dot{\phi}.s_\phi.s_\theta - \dot{\psi}.\dot{\theta}.c_\phi.c_\theta) \\ I_{yy}(\ddot{\theta} + \ddot{\psi}.s_\phi + \dot{\psi}\dot{\phi}.c_\phi) \\ I_{zz}(\ddot{\phi}.s_\theta + \dot{\phi}\dot{\theta}.c_\theta + \ddot{\psi}.c_\phi.c_\theta - \dot{\psi}\dot{\phi}.s_\phi.c_\theta - \dot{\psi}\dot{\theta}.c_\phi.s_\theta) \end{bmatrix} =$$

$$\begin{bmatrix} L(F_4 - F_2) \\ L(F_3 - F_1) \\ (M_2 + M_4 - M_1 - M_3) \end{bmatrix} - \begin{bmatrix} 0 & -(\dot{\phi}.s_\theta + \dot{\psi}.c_\phi.c_\theta) & (\dot{\theta} + \dot{\psi}.s_\phi) \\ (\dot{\phi}.s_\theta + \dot{\psi}.c_\phi.c_\theta) & 0 & -(\dot{\phi}.c_\theta - \dot{\psi}.c_\phi.s_\theta) \\ -(\dot{\theta} + \dot{\psi}.s_\phi) & (\dot{\phi}.c_\theta - \dot{\psi}.c_\phi.s_\theta) & 0 \end{bmatrix}$$
$$\text{x} \begin{bmatrix} I_{xx}(c_\theta.\dot{\phi} - \dot{\psi}.c_\phi.s_\theta) \\ I_{yy}(\dot{\theta} + \dot{\psi}.s_\phi) \\ I_{zz}(\dot{\phi}.s_\theta + \dot{\psi}.c_\phi.c_\theta) \end{bmatrix} \tag{2.40}$$

On equating row terms we get,

$$I_{xx}(-\dot{\theta}.\dot{\phi}.s_\theta + \ddot{\phi}.c_\theta - \ddot{\psi}.c_\phi.s_\theta + \dot{\psi}.\dot{\phi}.s_\phi.s_\theta - \dot{\psi}.\dot{\theta}.c_\phi.c_\theta)$$
$$= L(F_4 - F_2) + (I_{zz} - I_{yy})(\dot{\theta} + \dot{\psi}.s_\phi)(\dot{\phi}.s_\theta + \dot{\psi}.c_\phi.c_\theta)$$

$$I_{yy}(\ddot{\theta} + \ddot{\psi}.s_\phi + \dot{\psi}\dot{\phi}.c_\phi) = L(F_3 - F_1) + (I_{xx} - I_{zz})(c_\theta.\dot{\phi} - \dot{\psi}.c_\phi.s_\theta)(\dot{\phi}.s_\theta + \dot{\psi}.c_\phi.c_\theta)$$

$$I_{zz}(\ddot{\phi}.s_\theta + \dot{\phi}\dot{\theta}.c_\theta + \ddot{\psi}.c_\phi.c_\theta - \dot{\psi}\dot{\phi}.s_\phi.c_\theta - \dot{\psi}\dot{\theta}.c_\phi.s_\theta)$$
$$= (M_2 + M_4 - M_1 - M_3) + (I_{yy} - I_{xx})(c_\theta.\dot{\phi} - \dot{\psi}.c_\phi.s_\theta)(\dot{\theta} + \dot{\psi}.s_\phi)$$

Using small angle approximation

$$\ddot{\phi} = \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\psi}.\dot{\theta} + \frac{L(F_4 - F_2)}{I_{xx}} \tag{2.41}$$

$$\ddot{\theta} = \frac{I_{xx} - I_{zz}}{I_{yy}}\dot{\psi}.\dot{\phi} + \frac{L(F_3 - F_1)}{I_{yy}} \tag{2.42}$$

$$\ddot{\psi} = \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\phi}.\dot{\theta} + \frac{(M_2 + M_4 - M_1 - M_3)}{I_{zz}} \tag{2.43}$$

Considering gyroscopic effect caused by rotor inertia(J_r) and relative speed($\omega_r = \omega_2 + \omega_4 - \omega_1 - \omega_3$) between the motors, equation can be written as,

$$\ddot{\phi} = \frac{I_{yy} - I_{zz}}{I_{xx}} \dot{\psi}.\dot{\theta} + \frac{L(F_4 - F_2)}{I_{xx}} - \frac{J_r}{I_{xx}} \dot{\theta}\omega_r \tag{2.44}$$

$$\ddot{\theta} = \frac{I_{xx} - I_{zz}}{I_{yy}} \dot{\psi}.\dot{\phi} + \frac{L(F_3 - F_1)}{I_{yy}} + \frac{J_r}{I_{yy}} \dot{\phi}\omega_r \tag{2.45}$$

$$\ddot{\psi} = \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\phi}.\dot{\theta} + \frac{(M_2 + M_4 - M_1 - M_3)}{I_{zz}} \tag{2.46}$$

2.5 Propulsion System Modelling

2.5.1 Propeller Modelling

Propeller performance depends on its thrust T(unit:N) and torque M(unit: N.m).They are expressed as

$$T = C'_T \rho \left(\frac{N}{60}\right)^2 D_P^4 \tag{2.47}$$

$$M = C'_M \rho \left(\frac{N}{60}\right)^2 D_P^5 \tag{2.48}$$

where
N : propeller speed(RPM)
D_P : diameter of propeller(m)
C'_T : thrust coefficient
C'_M : torque coefficient
ρ : Air density(kg/m^3)
D_P : Diameter of propeller(m) [9]

Finding values of C'_T and C'_M

we have,

1. propeller diameter(D_P) = 0.254 m
2. propeller pitch(H_P) = 0.1143 m

Blade angle

$$(\Theta_{\mathrm{P}}) = arctan\left(\frac{H_p}{\pi D_P}\right) = 0.1422rad \tag{2.49}$$

Angle of attack

$$\alpha = \varepsilon.\Theta_{\mathrm{P}} = 0.1209rad. \tag{2.50}$$

where ε is a correction factor due to downwash , ε = 0.85

Lift and Drag coefficient: The lift coefcient C_1 of the blade airfoil section is related to the angle of attack α, which is given by:

$$C_1 = \frac{K_0\alpha}{1+\frac{K_0}{\pi A}} = \frac{6.11\times 0.1209}{1+\frac{6.11}{3.14\times 5}} = 0.5322 \tag{2.51}$$

where, K_0 is chosen as K_0 = 6.11

A = $\frac{D_{\mathrm{P}}}{C_{\mathrm{P}}}$ is the aspect ratio, where C_P is the average blade chord length. Considering the downwash, A is chosen as 5.

Drag coefficient C_d is given by :

$$C_{\mathrm{d}} = C_{\mathrm{fd}} + \frac{1}{\pi A e}\times {C_1}^2 = 0.03672 \tag{2.52}$$

where, C_{fd} is the zero-lift drag coefcient which depends on the thickness of the blade, the Reynolds number, the angle of attack, etc. C_{fd} = 0.015

The Oswald factor e is selected between 0.7 and 0.9.

C'_T is found using relation:

$$C'_T = 0.25\pi^3\lambda\zeta^3 B_p K_0 \frac{\alpha}{\pi A + K_0} = \mathbf{0.0984} \tag{2.53}$$

C'_M is found using relation:

$$C'_M = \frac{1}{8A}\pi^2 C_d \zeta^2 \lambda B_p^2 = \mathbf{0.00679} \tag{2.54}$$

By considering ρ= $1.293Kg/m^3$, D_P= 0.225m thrust and torque can be written as,

$$T = 1.471\times 10^{-7}N^2 \tag{2.55}$$

$$M = 2.582 \times 10^{-9} N^2 \tag{2.56}$$

where N is rotor speed in RPM.
Let,

$$C_T = 1.471 \times 10^{-7} \tag{2.57}$$

$$C_M = 2.582 \times 10^{-9} \tag{2.58}$$

2.6 Inertia Tensor Matrix

Inertia tensor matrix gives inertias in X, Y, Z directions. As momentum is calculated along principal axes, only diagonal elements are present.

We have,

1. Frame width = 450 mm. hence, x = y = 450/2 = **225 mm**

2. Total Weight(M) = Weight of frame + motors and ESC + battery + Pixhawk + GPS module

= 270 + 52*4 + 215 + 237 = **930 gm**

therefore,

$$I_{xx} = M(y^2 + z^2) = 0.93(0.225^2 + 0) = 0.0470 kgm^2 \tag{2.59}$$

$$I_{yy} = M(x^2 + z^2) = 0.93(0.225^2 + 0) = 0.0470 kgm^2 \tag{2.60}$$

$$I_{zz} = M(x^2 + y^2) = 0.93(0.225^2 + 0.225^2) = 0.0941 kgm^2 \tag{2.61}$$

Inertia tensor matrix = $\begin{bmatrix} 0.0470 & 0 & 0 \\ 0 & 0.0470 & \\ 0 & 0 & 0.0941 \end{bmatrix}$

2.7 Summary

This Chapter has described the rotation matrices that relate a vector in two co-ordinate frames. The different co-ordinate frames related to the study of the quadrotor model have been discussed and the transformation matrix between the inertial frame and the body frame has been derived. A mathematical model of the quadrotor system has been developed using Newton-Euler approach. It could be seen from the dynamics of the quadrotor motion that the quadrotor model is complex, coupled and highly non-linear. Because of the coupled dynamics, roll motion gets affected by pitch and yaw, pitch motion gets affected by roll and yaw and yaw motion is affected by roll and pitch. Along with the modelling of the quadrotor, chapter also explains about propulsion system modelling and calculations of various constants which will be used in further simulations.The control of attitude and altitude of the quadrotor is achieved by controlling the speed of the four propellers. The complete schematic of the attitude control of the quadrotor and different control strategies has been explained in next Chapter.

Chapter 3

Control System Design

3.1 Introduction

As can be seen from the equations (2.29, 2.30, 2.31, 2.44, 2.45 and 2.46), the quadrotor dynamics are highly non-linear and coupled. It means that a change in roll affects pitch and yaw, a change in pitch affects roll and yaw and a change in yaw affects roll and pitch. Quadrotor is also prone to external disturbances like wind gust and uncertainties in the mathematical model and performance of the hardware. Hence, the performance of the controller in the presence of disturbances is an important criteria for selection of the control strategy to be used.

Different control strategies have been examined in this chapter for controlling the attitude of the quadrotor. In this chapter PID and FOSM controllers have been studied and the simulation results have been presented for the same. A modification in the traditional PID controller is presented so that the resultant controller has better disturbance rejection properties. Because the quadrotor dynamics are coupled, four independent controllers are used; one each for roll, pitch, yaw and altitude. The controller for roll motion acts on the errors generated from the difference between actual and desired roll angles. The controllers for pitch and yaw motion work similarly. The controller for altitude acts on the errors generated from the difference between actual and desired altitude. The four controllers generate the required thrust needed to achieve the desired attitude and altitude.

This Chapter examines conventional PID controller for the attitude and altitude control of the quadrotor. Sliding mode control strategy is explained briefly and first order sliding mode control has been investigated for the control of quadrotor. The simulation results have also been presented.

The figure (3.1) illustrated the different stages in implementation of control for the quadrotor.

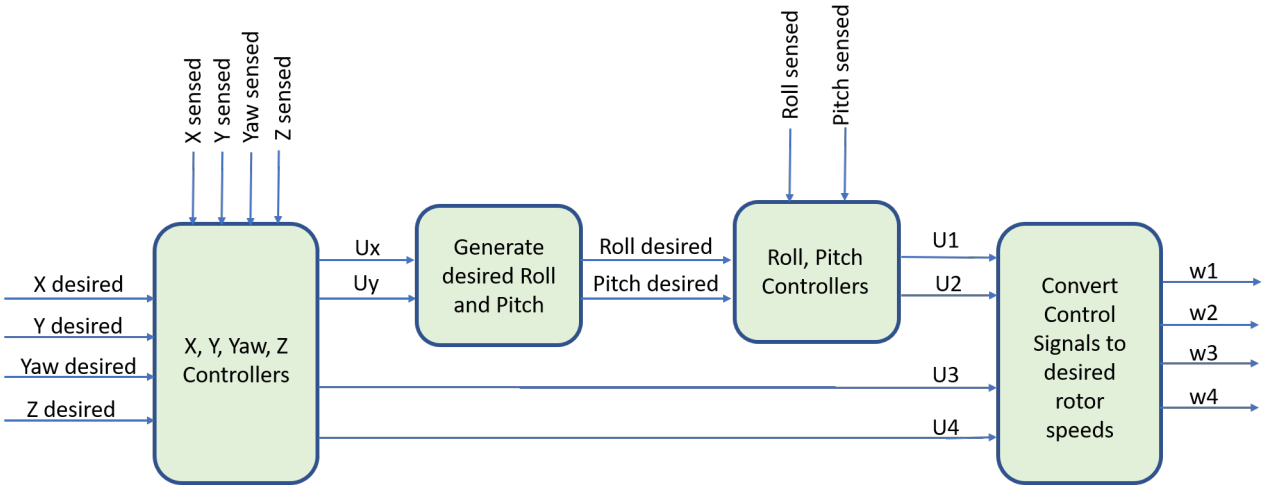


Figure 3.1: Illustrative Block Diagram

3.2 System Equations in terms of Control Inputs

Upon rewriting the thrusts generated in terms of control inputs, (U_1, U_2, U_3 and U_4), we get the following equations.

$$T_\phi = LC_T(\omega_4^2 - \omega_2^2) = U_1 \tag{3.1}$$

$$T_\theta = LC_T(\omega_3^2 - \omega_1^2) = U_2 \tag{3.2}$$

$$T_\psi = C_M(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2) = U_3 \tag{3.3}$$

$$T = C_T(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) = U_4 \tag{3.4}$$

Upon solving the above equations, rotor speeds can be written in terms of the control inputs as,

$$\omega_1^2 = -\frac{U_2}{2LC_T} - \frac{U_3}{4C_M} + \frac{U_4}{4C_T} \tag{3.5}$$

$$\omega_2^2 = -\frac{U_1}{2LC_T} + \frac{U_3}{4C_M} + \frac{U_4}{4C_T} \tag{3.6}$$

$$\omega_3^2 = \frac{U_2}{2LC_T} - \frac{U_3}{4C_M} + \frac{U_4}{4C_T} \tag{3.7}$$

$$\omega_4^2 = \frac{U_1}{2LC_T} + \frac{U_3}{4C_M} + \frac{U_4}{4C_T} \tag{3.8}$$

where C_T and C_M are thrust and torque coefficients.

As will be seen later from the control equations, the thrusts U_1 to U_4 provide the necessary thrust for trajectory control. Figure (3.2) shows the block diagram of the model that was simulated. From Figure (3.2) it can be seen that U_4 will provide the control for altitude while the x-y position controller will generate u_x and u_y. u_x and u_y can be seen as virtual commands which will be used to generate the desired ϕ_d and θ_d using the following equations. [8]
u_x and u_y virtual commands can be considered from equation (2.29 and 2.30)-

$$\ddot{x} = u_x.U4 \tag{3.9}$$

$$\ddot{y} = u_y.U4 \tag{3.10}$$

where u_x and u_y cane be considered commands given by terms of PID or SMC.

$$\phi_d = \arcsin\left(\sin\psi.u_x - \cos\psi.u_y\right) \tag{3.11}$$

$$\theta_d = \arcsin\frac{\left(\cos\psi.u_x + \sin\psi.u_y\right)}{\cos\phi_d} \tag{3.12}$$

These angles along with the desired yaw angle will be used to generate the thrusts U_1 to U_3.

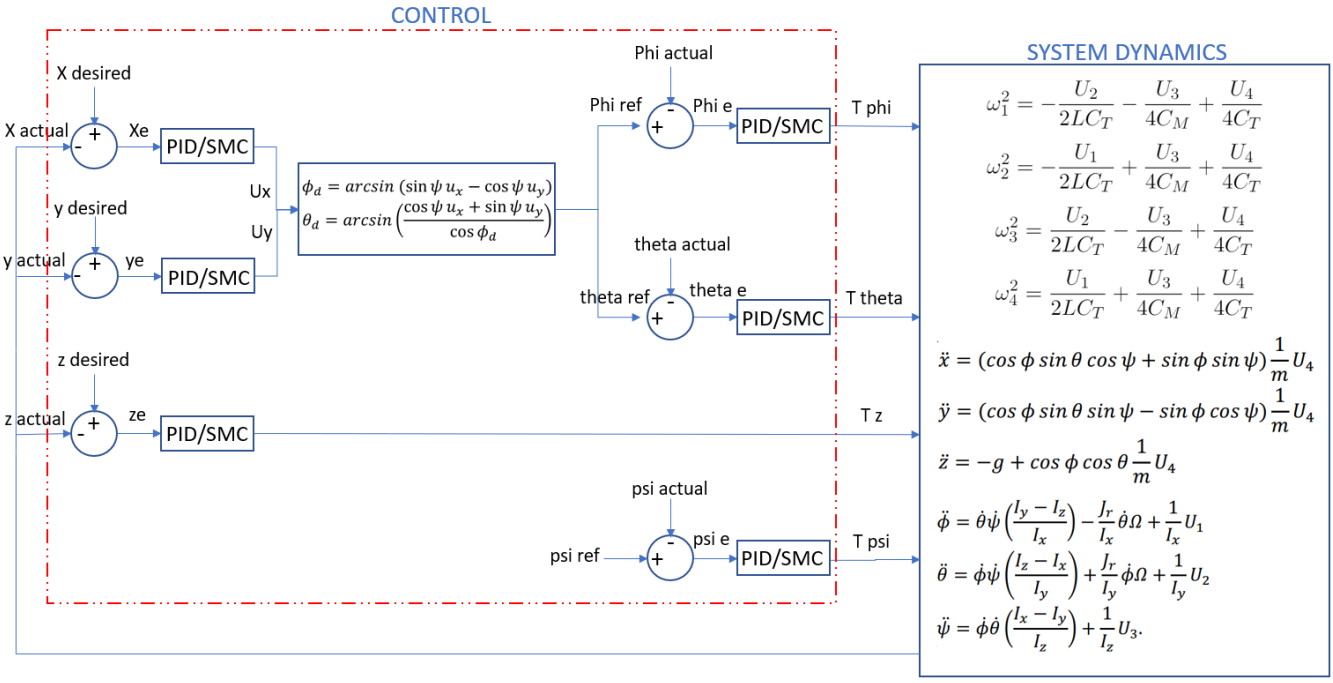


Figure 3.2: Control Block Diagram

3.3 Proportional Integral Derivative Controller

A PID controller iteratively calculates an error value e(t) as the difference between the desired set-point and the measured process variable and applies a correction based on proportional, integral, and derivative terms as shown in equation (3.13). [11]

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{3.13}$$

where,
K_p is the proportional gain
K_i is the integral gain
K_d is the derivative gain

K_p can be increased to speed up the transient response. However, with increase in the proportional gain, the gain cross over frequency is pushed further to the right, deteriorating the phase margin. The integral term is used to eliminate the steady state error. Derivative term damps any oscillations in the system. It is, however, susceptible to noise, thus requiring proper filtering. [12]

3.3.1 Simulation Results with PID controller

Four PID controllers were used to control the roll, pitch, yaw and altitude of the quadrotor. The reference signals for the roll and pitch angles are generated using the output signals of the x-trajectory controller and the y-trajectory controller according to the equations (3.11) and (3.12). The desired time response specifications for all control laws were : Rise-time $<$10 sec and acceptable limit of overshoot was 20%. It can be seen that the desired response time is easily acheived by the tuned gains of PID.

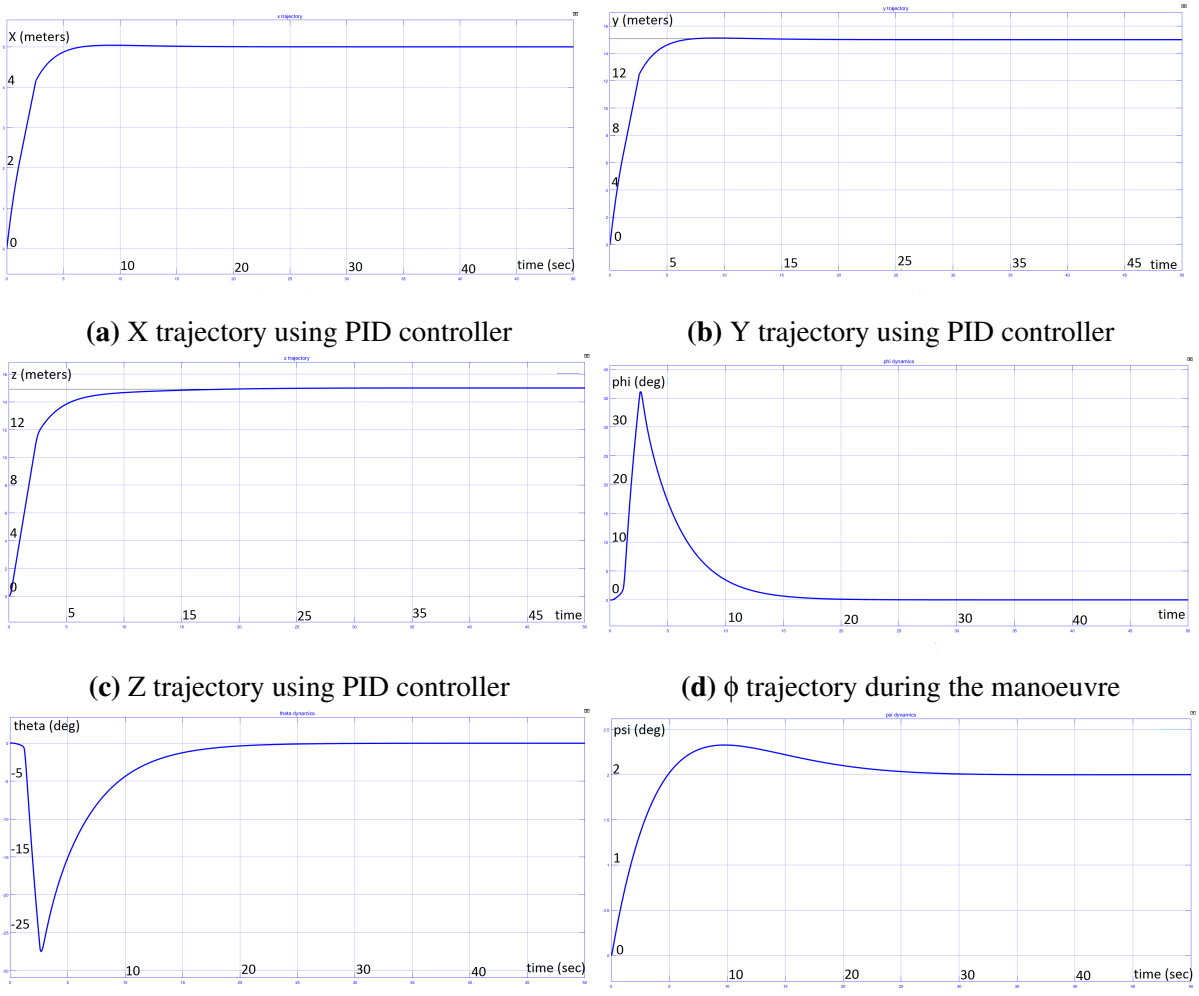
Table 3.1: Gains of the PID Controllers

	X controller	**Y controller**	**Z controller**	**ϕ controller**	**θ controller**	**ψ controller**
K_P	2	2	0.6	2	2	1
K_d	2	2	2.5	2	2	3
K_i	0	0	0.1	0.002	0.02	0.005

Without Disturbance

PID controllers with proportional, integral and derivative gains shown in the above table were used to control the response of the system when a desired location is given

as input to the x-y-z trajectory controllers. The response is shown in Figure (3.3). Figure (3.4) shows the trajectory in three dimensions. Figure (3.5) shows the angular speed of the four rotors of the quadcopter obtained during the simulations.



(a) X trajectory using PID controller

(b) Y trajectory using PID controller

(c) Z trajectory using PID controller

(d) ϕ trajectory during the manoeuvre

(e) θ trajectory during the manoeuvre

(f) ψ trajectory during the manoeuvre

Figure 3.3: Trajectory results using PID controller

With Disturbance

The figures(3.6a, 3.6c, 3.6e and 3.6g)show the performance of the PID controllers in the presence of a matched sinusoidal disturbance.It can be clearly seen that the conventional PID controller fails in the presence of such disturbance. Because the quadrotor is subject to multiple external disturbances, an improvement in the PID controller becomes necessary.

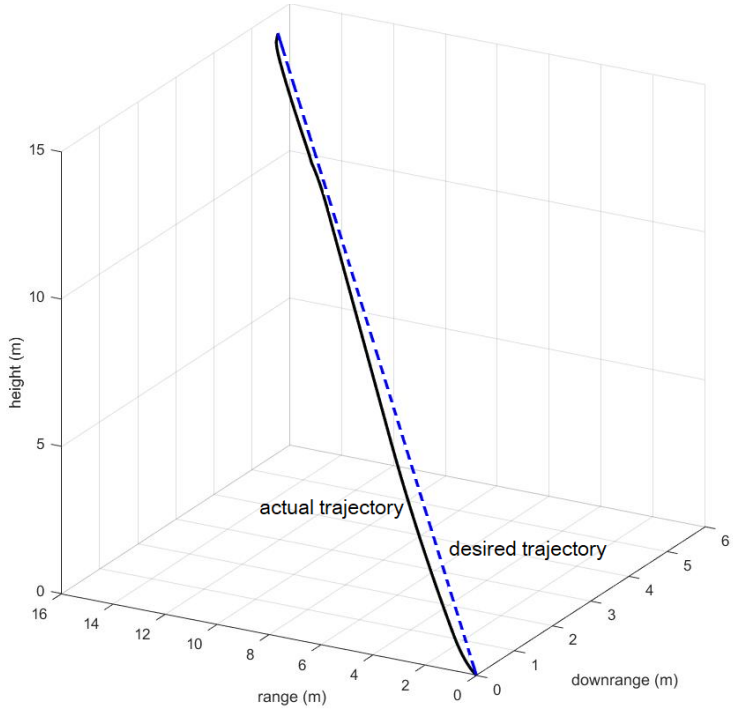


Figure 3.4: 3D trajectory

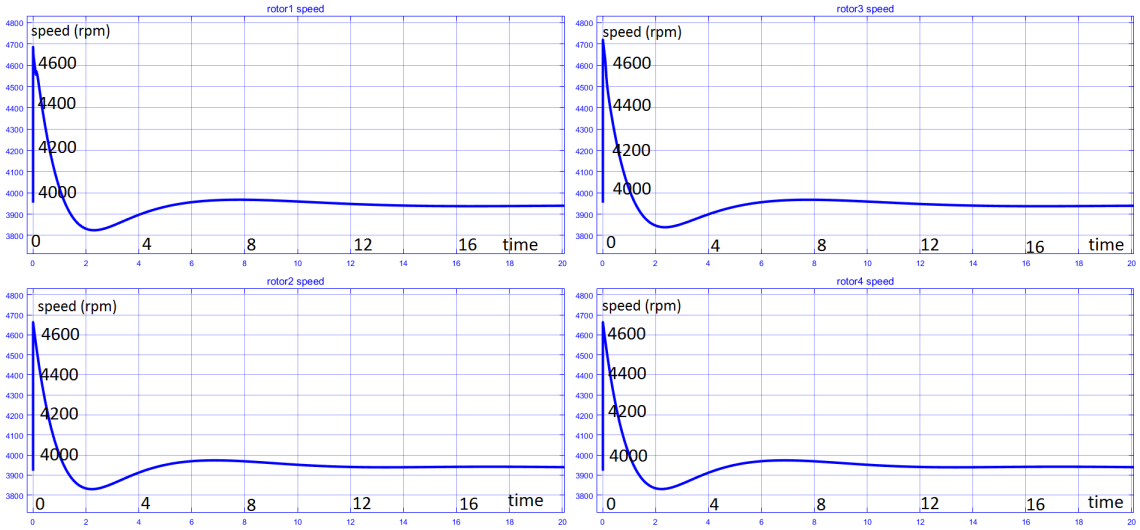


Figure 3.5: Angular speed of rotors

3.4 Robust PID Controller

One of the drawbacks of the PID controller is that it fares poorly in the presence of an external disturbance.The figures (3.6a, 3.6c, 3.6e and 3.6g)show the response of the system employing a traditional PID controller when a sinusoidal disturbance is applied to the quadrotor.

The PID control law can be modified to include Variable Structure Compensation as in Sliding Mode Control. The control law can be written as shown in the equation

(3.14). A certain class of disturbances in the system are rejected by employing this control law and a robust controller is obtained. [13]

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{\mathrm{d}}{\mathrm{d}t} e(t) - K sign(s) \tag{3.14}$$

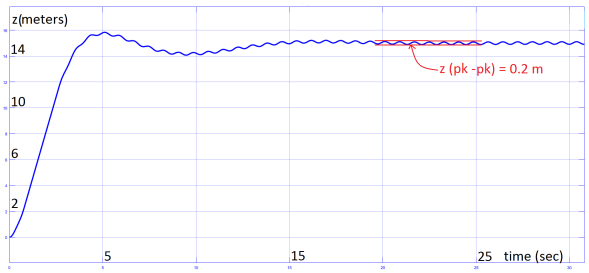
where, s is given as

$$s = \dot{e} + \alpha e, \;\; where\, \alpha > 0 \tag{3.15}$$

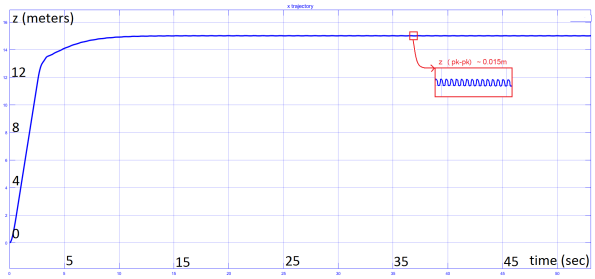
The $sign(s)$ is a discontinuous function defined as follows,

$$sign(s) = \begin{cases} +1, \; s > 0 \\ 0, \; s = 0 \\ -1, \; s < 0 \end{cases} \tag{3.16}$$

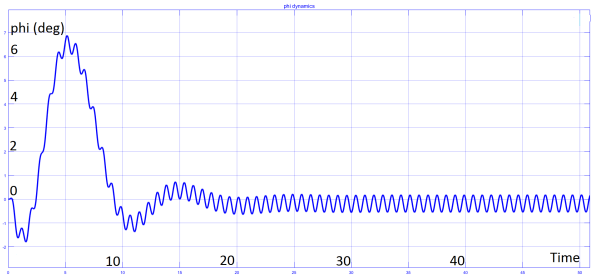
The constant K is determined based on the maximum expected disturbance in the system.



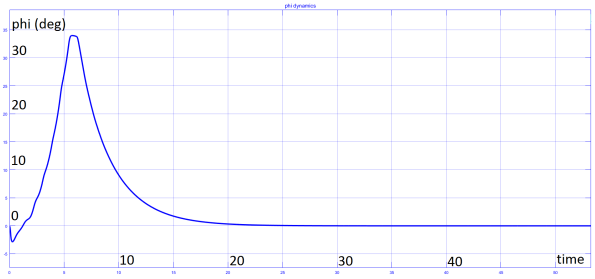
(a) z trajectory using PID controller in the presence of a sinusoidal disturbance



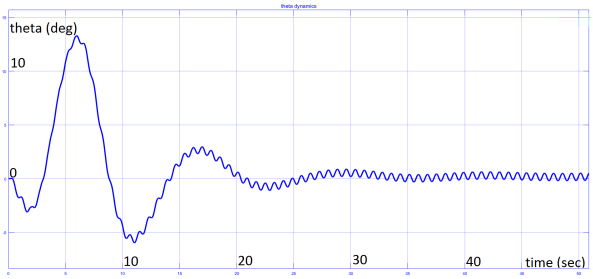
(b) z trajectory using a robust PID controller in the presence of a sinusoidal disturbance



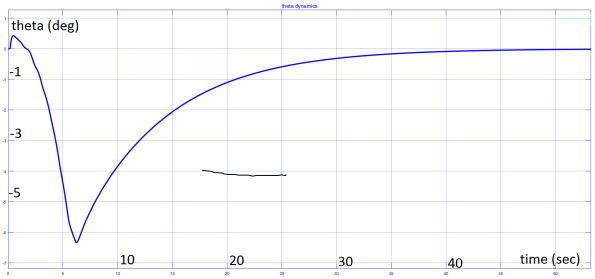
(c) ϕ trajectory using PID controller in presence of a sinusoidal disturbance



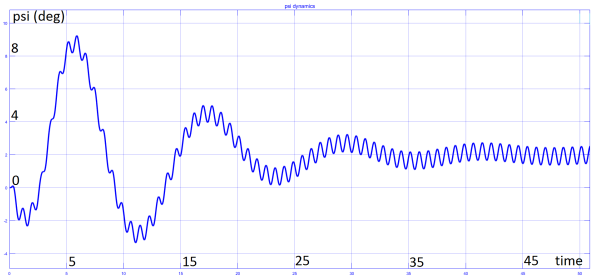
(d) ϕ trajectory using a robust PID controller in presence of a sinusoidal disturbance



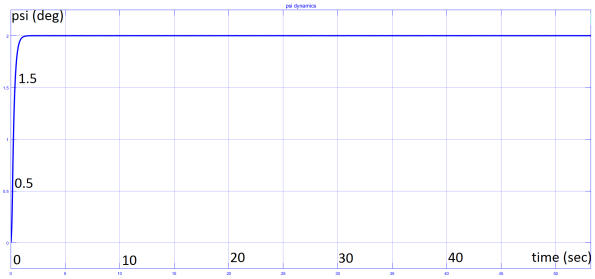
(e) θ trajectory using PID controller in presence of a sinusoidal disturbance



(f) θ trajectory using a robust PID controller in presence of a sinusoidal disturbance



(g) ψ trajectory using PID controller in the presence of a sinusoidal disturbance



(h) ψ trajectory using a robust PID controller in the presence of a sinusoidal disturbance

Figure 3.6: Trajectory results using Robust PID controller

3.5 Sliding Mode Controller

3.5.1 Introduction to Sliding Mode Control

Because of its disturbance rejection and robustness properties, the sliding mode control is suitable in a non-linear and coupled system like a quadrotor. The main idea behind the classical sliding mode control is maintaining the system state trajectory on a chosen

surface, called the sliding surface, by using discontinuous control.

As explained in [14], the idea is to replace an n^{th} order control problem with a first order problem by creating an intermediate variable "s", called sliding variable. s = 0 is called the sliding surface.
Consider a typical second order system

$$\ddot{x} = f(x,\dot{x}) + g(x,\dot{x})u + d \tag{3.17}$$

where 'd' is lumped uncertainties and external disturbance in the system.
which is defined as

$$s = \dot{e} + \alpha e, where\, \alpha > 0 \tag{3.18}$$

where,

$$e = x - x_d \tag{3.19}$$

where x is the actual value and x_d is the desired value
Upon differentiating the sliding variable 's', we get

$$\dot{s} = \ddot{e} + \alpha\dot{e} \tag{3.20}$$

$$\dot{s} = \ddot{x} - \ddot{x}_d + \alpha(\dot{x} - \dot{x}_d) \tag{3.21}$$

Since the desired value is constant, $\dot{x_d} = 0$ and $\ddot{x_d} = 0$
Therefore,

$$\dot{s} = \ddot{x} + \alpha\dot{x} \tag{3.22}$$

Substituting the value of $\ddot{x}$ from (3.17), we get,

$$\dot{s} = f(x,\dot{x}) + g(x,\dot{x})u + d + \alpha\dot{x} \tag{3.23}$$

If $\dot{s}$ is made discontinuous then, s and $\dot{s}$ become zero is finite time. Using,

$$\dot{s} = -qs - ksign(s) \tag{3.24}$$

From (3.23 and 3.24), we get the control expression as below,

$$-ksign(s) = f(x,\dot{x}) + g(x,\dot{x})u + d + \alpha\dot{x} \tag{3.25}$$

Since the disturbance is not known, we choose the value of u as below,

$$u = g(x,\dot{x})^{-1}(-f(x,\dot{x}) - ksign(s) - qs - \alpha\dot{x}) \tag{3.26}$$

(3.26) is the necessary sliding mode controller.
k is determined such that the sliding condition s = 0 is ensured and reached in a finite time even in the presence of any disturbance 'd'.
Defining a Lyapunov function,

$$V = \frac{1}{2}s^2 \tag{3.27}$$

The time derivative of (3.27) is

$$\dot{V} = \dot{s}s \tag{3.28}$$

Substituting for $\dot{s}$ from (3.23)

$$\dot{V}(s) = s(f(x,\dot{x}) + g(x,\dot{x})u + d + \alpha\dot{x}) \tag{3.29}$$

Substituting u from (3.26)

$$\dot{V}(s) = s(f(x,\dot{x}) + g(x,\dot{x})\ g(x,\dot{x})^{(-1)}(-\alpha\dot{x} - f(x,\dot{x}) - ksign(s) - qs) + d + \alpha\dot{x}) \tag{3.30}$$

$$\dot{V}(s) = s(-ksign(s) - qs + d) \tag{3.31}$$

$$\dot{V}(s) = -kssign(s) - qs^2 + ds \tag{3.32}$$

$$\dot{V}(s) = -k|s| - qs^2 + ds \tag{3.33}$$

According to reachability condition, which is, $s\dot{s} = -\varepsilon|s|$, where ε is some positive constant, it is clear that if $k > d_{max}$, finite time reaching condition is guaranteed.

The control will be stabilising since $\dot{V}(s) < 0$

3.5.2 Control Equations for a Quadrotor

The following control laws for Z and X-Y translation are obtained [8]

$$U_4 = \frac{m}{cos\phi cos\theta}[g + \ddot{z_d} - \alpha_3(\dot{z} - \dot{z}_d) - k_3 sign(s) - l_3 s] \tag{3.34}$$

$$u_x = \frac{m}{U_4}[\ddot{x}_d - \alpha_1(\dot{x} - \dot{x}_d) - k_1 sign(s) - l_1 s] \tag{3.35}$$

$$u_y = \frac{m}{U_4}[\ddot{y}_d - \alpha_2(\dot{y} - \dot{y}_d) - k_2 sign(s) - l_2 s] \tag{3.36}$$

$$U_1 = I_x\left[-\dot{\theta}\psi\left(\frac{I_y - I_z}{I_x}\right) + \frac{J_r}{I_x}\dot{\theta}\Omega + \ddot{\phi} - \alpha_4(\dot{\phi} - \dot{\phi}_d) - k_4 sign(s) - l_4 s\right] \tag{3.37}$$

$$U_2 = I_y\left[-\dot{\phi}\psi\left(\frac{I_z - I_x}{I_y}\right) + \frac{J_r}{I_y}\dot{\phi}\Omega + \ddot{\theta} - \alpha_5(\dot{\theta} - \dot{\theta}_d) - k_5 sign(s) - l_5 s\right] \tag{3.38}$$

$$U_3 = I_z\left[-\dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \ddot{\psi} - \alpha_6(\dot{\psi} - \dot{\psi}_d) - k_6 sign(s) - l_6 s\right] \tag{3.39}$$

A problem with the sliding mode control is the chattering phenomenon due to the $sign(s)$ function in the above expressions. To avoid this drawback, which can affect the overall performance, this discontinuous function is replaced by a saturation function defined as follows:

$$sat(s_k) = \begin{cases} sign(s_k), & \mid s_k \mid \leq \rho_k \\ \frac{s_k}{\rho_k}, & \mid s_k \mid < \rho_k \end{cases} \tag{3.40}$$

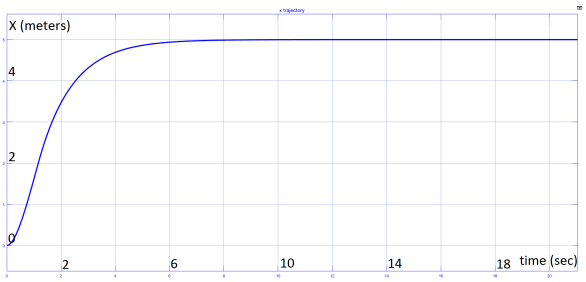
Table 3.2 shows the values of constants used in the simulations.

Table 3.2: Constants of SMC

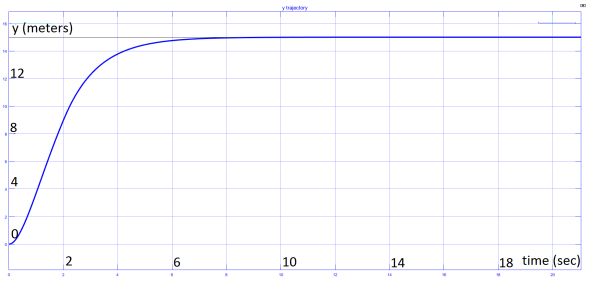
	X-Y translation		**Altitude**	**Attitude**		
	X controller	**Y controller**	**Z controller**	**ϕ controller**	**θ controller**	**ψ controller**
α_i	0.8	0.8	0.9	7	7	5
$\mathbf{k_i}$	1.98	1.98	2.78	5.25	5.25	2.75
$\mathbf{l_i}$	0.9	0.9	1.25	3.5	3.5	1.8

3.5.3 Simulation Results with SMC

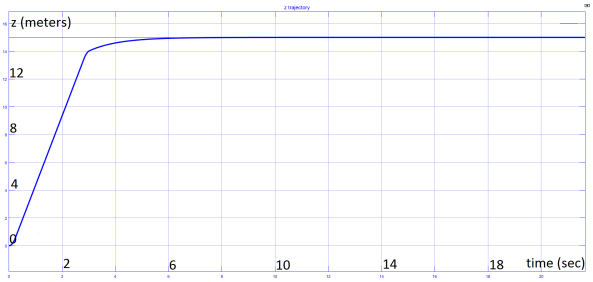
Figure (3.7) shows the simulation results of the X-Y-Z trajectories and variations in the ϕ, θ and ψ angles to execute the trajectory manoeuvre when a desired (5, 15, 15) set point is given as an input to the model of the quadrotor. The desired rise time is $t_r < 10sec$ and the overshoot should not be more than 20 percent.



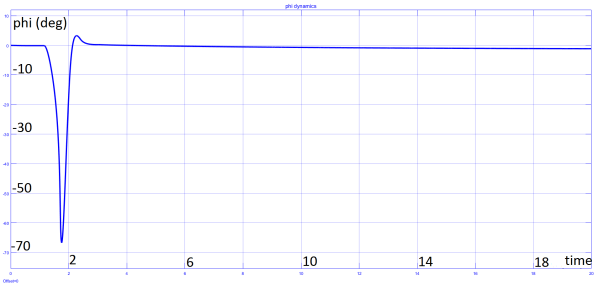
(a) X trajectory using SMC



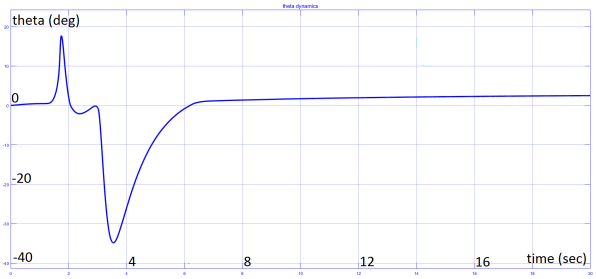
(b) Y trajectory using SMC



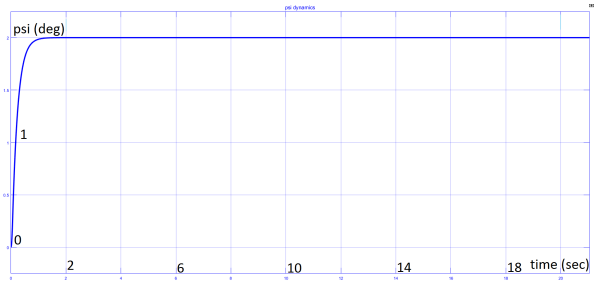
(c) Z trajectory using SMC



(d) ϕ trajectory during the manoeuvre

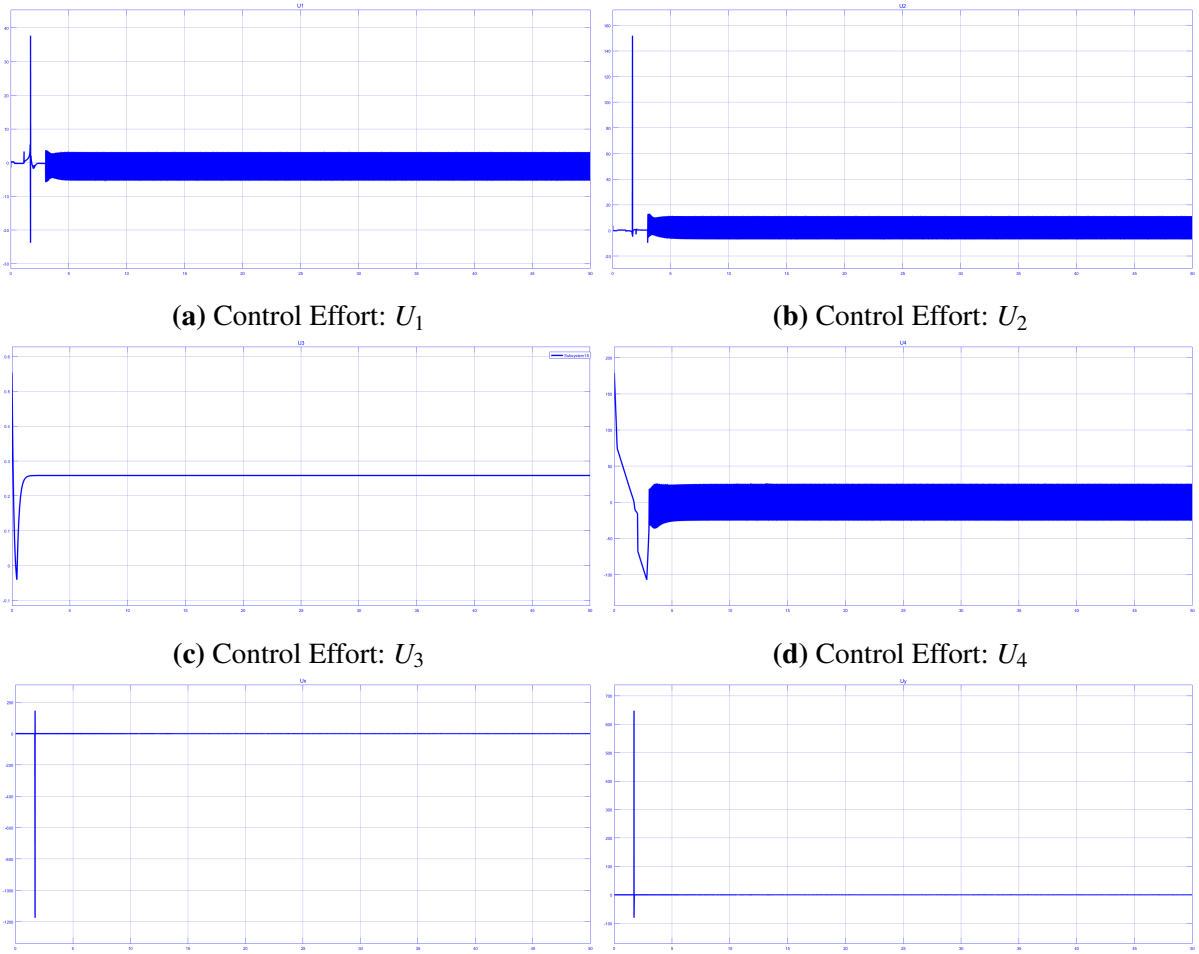


(e) θ trajectory during the manoeuvre



(f) ψ trajectory during the manoeuvre

Figure 3.7: Trajectory results using Sliding Mode Controllers



(a) Control Effort: U_1 **(b)** Control Effort: U_2

(c) Control Effort: U_3 **(d)** Control Effort: U_4

(e) Control Effort: U_x **(f)** Control Effort: U_y

Figure 3.8: Control Efforts vs time of Sliding Mode Controllers

3.5.4 Performance of PID and Sliding Mode Controllers with disturbance

The quadrotor is subjected to various disturbances due to the external environment like wind or collision with any particle, un-modelled dynamics or uncertainties in the system parameters. These disturbances have a destabilising effect of the quadrotor. Failure of the control algorithm in the middle of the flight can leave the quadrotor damaged and the mission of the drone will remain incomplete. Thus, disturbance rejection is an important criteria when designing systems like quadrotors which carry out critical missions.

In the simulations, the quadrotor is subjected to a disturbance $d(t) = 0.1sin(\omega t)$. The altitude tracking results are shown in the Figure (3.9). As observed from the Figure (3.9), the PID controller fails to give the desired results in the presence of disturbances. The Sliding Mode Controller is able to reject the disturbance and gives a superior performance as compared to the PID controller. The modifications to the PID controller proposed in the section (3.4), improve its disturbance rejection properties, thus making the controller better suited for outdoor applications of quadrotors as compared to

traditional PID controllers.

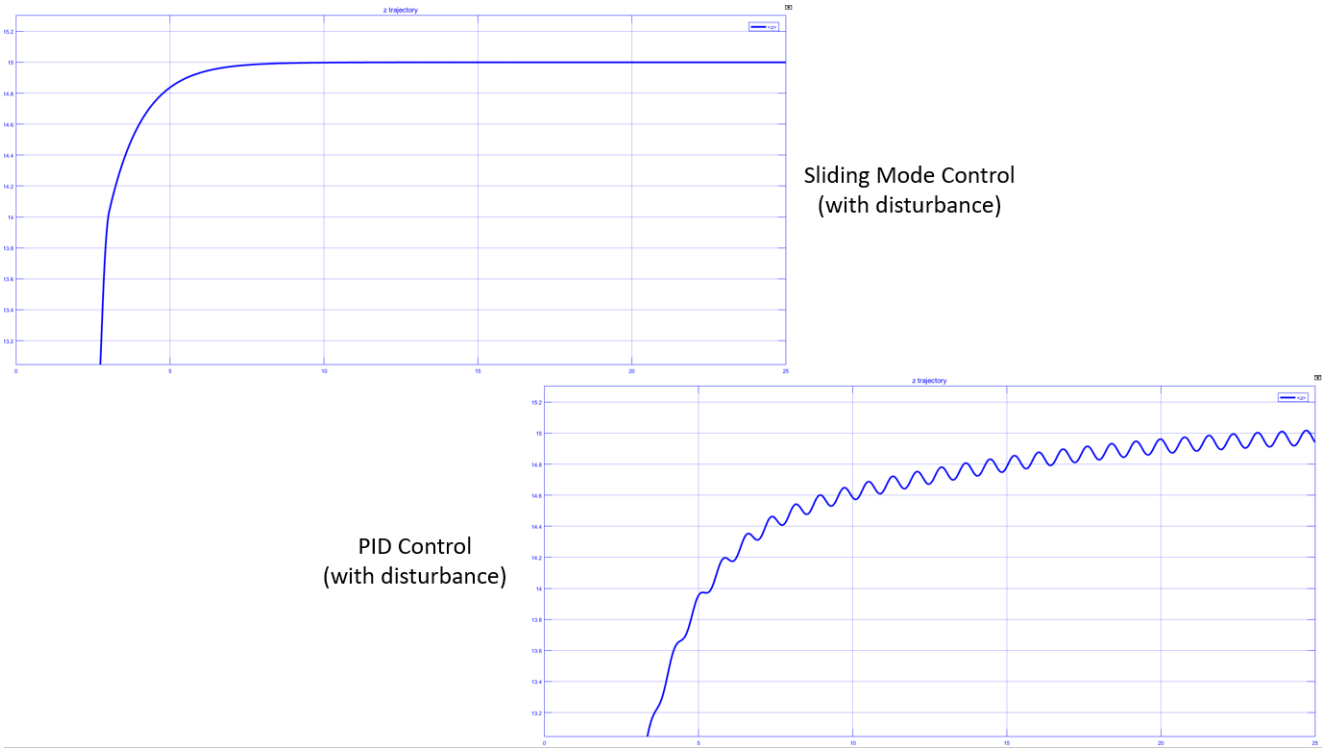


Figure 3.9: Response of PID and Sliding Mode Controllers in the presence of a disturbance $d(t) = 0.1sin(\omega t)$

The table (3.3) shows the values of control energy utilised by the four decoupled PID controllers required to achieve the desired location in the presence of matched disturbance. It can be clearly seen that the control energy required by Sliding Mode Controller is smaller as compared to that required by the conventional PID controller. The norm of error in z, ϕ, θ and ψ is also shown. The Sliding Mode Controller has better performance in the presence of disturbance as compared to the PID controllers, however, SMC requires calculation of multiple derivatives which is computationally difficult as well as prone to noise.

Table 3.3: Control Energy

	$\|\mathbf{U_1}\|$	$\|\mathbf{U_2}\|$	$\|\mathbf{U_3}\|$	$\|\mathbf{U_4}\|$	$\|\mathbf{e_z}\|$	$\|\mathbf{e_\phi}\|$	$\|\mathbf{e_\theta}\|$	$\|\mathbf{e_\psi}\|$
PID Control	314.74	236.36	974.93	61978	2436.0	112.15	125.55	182.59
Robust PID	230.41	187.98	270.5	42623	3450	79.79	51.46	56.03
Sliding Mode Control	173.91	173.14	180.31	25302	1591.6	22.82	14.77	17.74

3.6 Summary

In this chapter, attitude and altitude controllers have been developed using Proportional Integral Derivative Control strategy and Sliding Mode Control strategy. The simulation results of all the three methods have been presented with and without external disturbances. The gains of PID controller have been chosen to give a satisfactory response. Sliding Mode Control strategy has been explained briefly and a conventional first order sliding mode controller has been developed. From the simulation results it is evident that both the PID and FOSM controllers give the desired response in the absence of any external disturbance, however, FOSM is a robust control strategy which is evident from the rejection of disturbance as seen in Figure (3.9). The requirement of computation of double derivatives of system parameters adds to the complexity in implementing the FOSM controller. Modified PID controller presented in the section (3.4) provides a performance superior to conventional PID controller in the presence of disturbances and is computationally less demanding as compared to FOSM. Details about hardware implementation are presented in the next chapter.

Chapter 4

Quadcopter Hardware Development

4.1 Introduction

When given a quadcopter, parameters of hardware components define the overall performance of the system. To design a quadcopter, designers have to select proper components to meet system performance requirements. System parameters like hover endurance, system efficiency, maximum pitch angle, maximum payload, maximum flight distance, all depend on the choice of hardware components.

Quadcopter performance is mainly determined by the

1. Propellers
2. Brushless Direct Current motors
3. Electronic Speed Controllers (ESCs)
4. Batteries

Other hardware components used in the quadrotor system are - frame to accomodate all components, Pixhawk Controller, transmitter and receiver, GPS module for accurate detection of quadrotor in 3D space and remote controller to operate the quadrotor remotely. Specifications of each components are given in this chapter.

4.2 Frame

Quadcopter frame is the basic structure that carries all the components of the quadrotor like controller, GPS module, receiver, batteries, etc. It should have enough strength and space to accommodate all the components on it and should also allow installation of propellers that will produce the required thrust for the quadrotor.

The frame used is Q450 [15] - Quadcopter frame from Robokits India - a cross shaped airframe. The Q450 is 450mm quad frame.The main frame is glass fiber while the arms are constructed from ultra-durable polyamide nylon.

Q450 frame features integrated PCB connections for direct soldering of ESCs. This eliminates the need for a power distribution board or messy multi-connectors keeping the electronics layout very tidy. Q450 also comes with stronger molded arms, so arm breakage at the motor mount on a hard landing is avoided.

A great feature of this frame is the large mounting tabs at the front and rear of the main frame bottom plate for mounting cameras or other accessories. This allows one to take aerial video or fly FPV(First person View) without the need to add any additional mounting brackets.

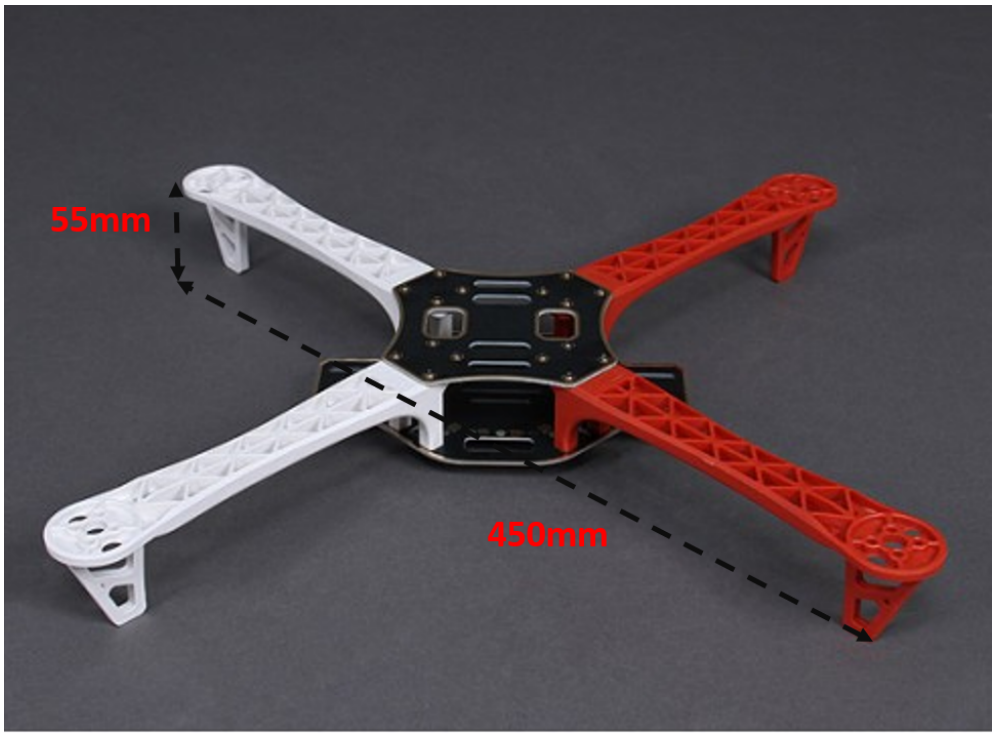


Figure 4.1: Q450 Frame

Table 4.1: Q450 Frame Specifications

Width	450mm
Height	55mm
Weight	270gm
Motor Mount Bolt Holes	16/19mm

4.3 Motors

The quadrotor propulsion system consists of Brushless Direct Current (BLDC) electric motors, electronic speed controller and propellers. BLDC motors have proven payload efficiency, high power and reliable performance. They replace the field windings with permanent magnets located on the rotor and move the armature windings to the stator. In this manner, the need for mechanical commutation is eliminated reducing noise and electromagnetic interference. The elimination of mechanical commutation allowed greater rotational speeds and eased maintenance.

BLDC motors are synchronous motors which mean that both the magnetic fields generated by the stator and the rotor rotate at the same frequency. BLDC motors do not experience the slip that is normally seen in induction motors.

A common type of BLDC motors is the outrunner, which has a spinning outer case as part of the rotor. This type of BLDC motor can provide a larger amount of torque at low speeds without the need for mechanical gearing. BLDC motors exist in different configurations, but the most common configuration is the three phase motor due to its efficiency, low torque and good precision in control. Motor used in this project is Make Robokits, Model No 2212, 1000KV[16]. It is a powerful Brushless motor in small size for remote control applications and is specialised for multi-rotors. It includes prop adaptor to suite variety of props, supports pusher or puller props (counter rotating props). It comes with motor mount and Bullet male connectors.

Table 4.2: Specifications of BLDC Motor

Weight	52 gm
KV	1000RPM/V
Maximum Current	13A
No load Current	0.5A at 10V
Maximum Surge Watts	180W
Number of poles	14
Internal Resistance	0.090 ohms
Shaft Diameter	3.2mm

Figure 4.2: Robokits BLDC motor A2212/13T 1000KV

4.4 Electronic Speed Controller

Brushless DC motors entails the use of Electronic Speed Controllers (ESC) to perform electronic commutation. ESCs are very important as they control the speed of BLDC motor.
A standard commercial ESC should tell the number of amps it supplies to your motor (this is the 'size' of the ESC) along with additional information such as if it has a Battery Eliminating Circuit (BEC) or not, and what batteries it supports.

Normally the size and weight of an ESC is proportional to the amp rating. BEC equipped ESC means that they are able to output a constant voltage, and so power some equipment like the fight controller. BEC spares the developer of the quadcopter platform the headache of providing diferent power sources for both the motors and other components.
ESC used in this project is SimonK ESC 30A [17] as shown in the figure (4.3).

4.4.1 Features of ESC

Some of the important featurs of SimonK ESC 30A are

- High-quality MOSFETs for BLDC motor drive
- High-performance microcontroller for best compatibility with all types of motors at greater efficiency
- Fully programmable with any standard RC remote control
- Heat sink with high-performance heat transmission membrane for better thermal management

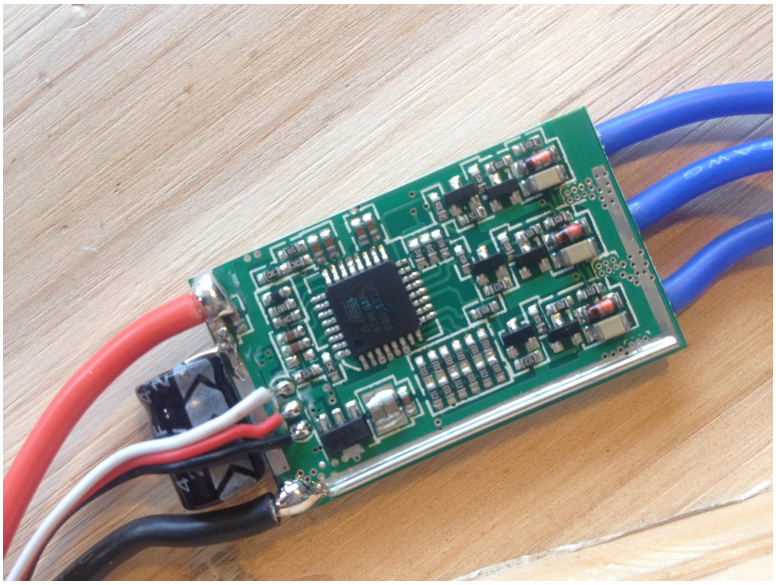


Figure 4.3: SimonK ESC 30A

- 3 start modes: Normal / Soft / Super-Soft, compatible with fixed-wing aircraft and helicopters
- Throttle range can be configured to be compatible with any remote control available in the market

Table 4.3: ESC Specifications

Type	SimonK 30A
Maximum Current	40A for 10 sec
Constant Current	30A
Inpput Voltage	6-12V
BEC	5V, 3A
Weight	26gm
Dimensions	56mmX24mmX8.5mm

4.5 Propeller

In rotorcraft, the propellers are the sole generators of aerodynamic forces. They generate lift and drag forces. Its rotary motion through the air creates a difference in air pressure between the front and back surfaces of its blades. Propellers exist in different length and pitch :

- The length of a propeller is the diameter of a disc the propeller makes when it is spinning
- The pitch is the turning of the angle of the blades of a propeller to control the production or absorption of power

Figure 4.4: 1045 Propllers

Larger propeller or higher pitch will increase aircraft's speed but also use more power, the larger the propeller (either increasing diameter, pitch or both) the more energy it takes to spin.

Normally, a propeller for a quadrotor is characterized by its material, weight and rigidity of its structure. Most durable propellers are made of composite plastic or carbon fiber. Composite plastic is ubiquitously used, however carbon fiber is lighter in weight and has a more rigid frame.

Lighter weight means lower drag and less moment of inertia, this enhances the performance and increases controllability.The more rigid the propeller frame is, the better performance at high RPMs. Weak propellers may bend or deform at high RPM, this would make the propeller lose thrust and destabilize the vehicle.

In this work, 1045 L/R CW CCW Nylon Propeller Pair 10" x 4.5"[18]. Diameter and Pitch angle of the selected propeller are 10 inch and 4.5 inch respectively.

4.6 Battery

The most common type of batteries used in UAV applications have Lithium polymer cell chemistry, more commonly known as LiPo. This could be attributed to their high energy density, high discharge rate and light weight which make them a great candidate.

Figure 4.5: Orange 3000mAh 30C Battery

LiPo batteries are made up of individual battery cells connected in series.Each cell has a nominal voltage of 3.7V. Motor RPM is directly related to battery voltage.The output RPM from the BLDC motor depends on both the motor's KV value and supplied voltage. However, the higher the voltage the bigger the battery.

Characterization of LiPo batteries does not depend solely on voltage, the capacity of battery is measured in Milli-amp hour (mAh) which indicates the amount of current the motor can draw from the battery for an hour until it is empty. For example, an 2200 mAh would take an hour to get empty if the motors are constantly drawing 2.2 amp. Also, battery discharge capacity (C-rating) is another factor that can affect motor performance. The battery discharge capacity should be adequate enough to supply motors maximum current draw.

Battery used in this project is Orange 3000mAh 3S 40C/80C Lithium polymer battery Pack (LiPo)[19].Battery is equipped with 3 cells in series with 40C discharge rate at constant level and 80C dischare rate at burst conditions.Detailed specifications of the battery are mentioned in the table 4.4

Table 4.4: Specifications of Orange 3000mAh Battery

Capacity	3000mAh
Configuration	3S1P / 11.1V /3 Cell
Constant Discharge	40C
Peak Discharge	80C
Weight	215gm

4.7 Remote Controller

A Radio Control (RC) is a device that allows the control of the aircraft wirelessly through the wireless transmitter (TX). The signal/commands are then received by a radio receiver (RX) which is connected to a flight controller. Normally, an RC is being selected based on the number of channels, modes, frequency technology and type of receiver. The number of channels in an RC determines the number of individual actions by which the aircraft can be controlled. For example, a throttle control needs one channel and so does roll, pitch and yaw control. So a minimum of 4 channels are required for controlling the 6 motions of quadcopter. However, normal control of a quadrotor requires more channels to allow for extra functions and control of the vehicle and keep space for changing mode or trigger a certain function of the multirotor (i.e. emergency landing, fail safe mode, etc). The new standard for RC controller uses 2.4Ghz frequency. There are other frequencies that have been used among RC communities like 1.3ghz, 900mhz or 433MHz but they are less common. Finally, the criteria for using the RC is based on the protocol receiver protocol .Some of the protocols are

1. PWM Pulse Width Modulation: The old fashioned receivers, they use one servo wire for each channel. They are the cheapest option, however and due to the messy wiring they require , they have been replaced recently.

2. PPM Pulse Position Modulation: The advantage of PPM is that only one signal wire is needed for several channels, instead of a number of individual wires. Technically, they send multiple PWM signals down a single wire in succession. The PPM signal is just like the PWM signal, both are analog signals.

3. PCM Pulse Code Modulation: which is similar to PPM in its data types, however, PCM sends a digital signal while PPM is analog.

4. SBUS- Serial BUS: it is an inverted UART communication signal and a type of serial communication protocols, used by Futaba and FrSky RCs. It supports up to 18 channels using only one signal cable.

5. DSM2/DSMx: an interference-resistant transmitting protocol. It always hops between frequencies to maintain the best connection to your radio. It has two advantages, first, the signal is spread out over a wider frequency band and each transmitter/receiver pair uses its own coding scheme to scramble the signal.

By considering various receiver , SBUS capable receiver is used as its protocol supports a large number of channels and uses a single cable for giving PWM signals.

Frsky's X8R receiver [20] is used because of capability of supporting upto 16 channels and SBUS protocol. Table 4.5[20] lists down some of the features about the receiver.

Table 4.5: Frsky's X8R Receiver

Number Of Channels	1-16ch
Operating Voltage	4.0 - 10.0V
Operating Current	100mA@5V
Operating Range	full range (greater than 1.5km)
Weight	16.6gm
Compatibility	FrSky Taranis X9D Plus/XJT in D16 mode

Figure 4.6: Frsky's Receiver X8R

For this project Frsky's Taranis X9D Plus radio controller [21]. It supports upto 1-16 channels and RSSI (Receiver Signal Strength Indicator) gives alert if the signal strength falls below certain limit. Its is equipped with ACCST (Advanced Continuous Channel Shifting Technology). Also it provides an LCD screen to configure various features. This LCD panel can also be used for telemetry purposes. It uses an open source firmware called OpenTX. Being open source it can be modified by the user. Some notable features about the RC are as follows given in table 4.6[22] -

Table 4.6: Frsky's Taranis X9D Specifications

Number Of Channels	1-16ch
Operating Voltage	6-15V(2S, 3S Lipo's acceptable)
Operating Current	260mA maximum
Operating Temperature	10-60 degrees celsius
Compatibility	Frsky X Series,D series, V8-II

Figure 4.7: TARANIS PLUS Remote controller

4.8 GPS Module

The GPS module provides the geolocation to the flight controller, it is specifically useful for providing coordinates of the vehicle that the flight controller can use. A GPS module is used in this project to perform trajectory tracking by getting the latitude and longitude.

For this project 'Holybro's Pixhawk 4 GPS module [23]' which has Ublox Neo-M8N GPS module[24] is used. The NEO-M8 series of standalone concurrent GNSS modules is built on the exceptional performance of the U-blox M8 GNSS (GPS, GLONASS, BeiDou, QZSS, SBAS and Galileo-ready1) engine in the industry proven NEO form factor. This module has a staggering 167db navigation sensitivity which is currently industry leading sensitivity level. This module also contains IST8310 compass and a tri-colored LED indicator. Some of the specification about the GPS module are -

Table 4.7: GPS Module Specifications

GPS Module	Ublox Neo-M8N module
Sensitivity	167 dBm navigation sensitivity
Antenna	25 x 25 x 4 mm ceramic patch antenna
Compass	IST8310
Operating Voltage	4 - 5V

Figure 4.8: Holybro's GPS Module

4.9 Telemetry

Radio Telemetry allows a user to connect ground station computer with the flight controller to sense and update parameters about the flight controller. Telemetry allows a user to wirelessly connect to the flight controller and monitor data. Telemetry can be used to collect certain data at various points and send it to a computer or ground station for analyzing the same. Telemetry helps user to control flight parameters -for e.g. a user can change the PID contants in the flight controller while the quadcopter is flying to fine tune the controller. For this project mRo SiK Telemetry Radio V2 433Mhz is used. A SiK Telemetry Radio is a small, light and inexpensive open source radio platform that typically allows ranges of better than 300m. The range can be extended to several kilometers by use of patch antenna on the ground. The radio uses open source firmware which has been specially designed to work well with MAVLink, thus it is compatible with flight manager softwares like Qgroundcontrol, Mission planner, etc. In this project telemetry is used for transmission of sensor data from matlab models when in external mode. Some of the feature of this telemetry device are given in table 4.9

Table 4.8: Details on Telemetry

Very Light Weight (Under 4 grams without antenna)
Frequency of 433Mhz (for v2)
Transmit Power upto 20dBm
MAVlink Enabled
Receiver sensitivity to -121 dBm
Open source firmware

Figure 4.9: mRo Sik Telemetry Radio v2

4.10 Quadcopter Assembly

Figure 4.10: Quadcopter Assembly

Quadcopter assembly is done by using the above mentioned components. Figure 4.10 shows the assembled quadcopter which is used for flight testing. Total weight of the quadrotor is 990gm and this can be calculated by addition of mass of each component.

4.11 Summary

This chapter gives detailed explanation about each component used for the assembly of quadcopter. To meet the system requirements with good efficiency proper components selection is necessary. This chapter explains the impetus behind selection of components used in quadcopter. It gives detailed explanation on each components specification - maximum-minumum driving points, dimensions, features,etc. This chapter expounds on the use of various components/modules in this project. Detailed information about Pixhawk Auto -Pilot which is major hardware component in designing quadcopter is explained in next chapter.

Chapter 5

Pixhawk Flight Controller

5.1 Introduction to Pixhawk Controller

After completing the simulations using the designed control law, the algorithm has to be tested in environmental conditions to validate the model. A number of actuators and sensors will have to be used on the quadcopter to complete this task. Sensors such as gyroscopefor vehicles attitude in roll, pitch, yaw, accelerometer for finding instantaneous accelerations in X, Y, Z and barometer-for measuring relative height will be used. Assembling a micro-controller with various sensors will not satisfy our requirement of speed and robustness- as it will not be able to interact and take decisions rapidly like a flight controller. Along with faster processing speed there is a need for asynchronous processing i.e. multithreaded processes.With that said, interacting systems i.e. sensors and actuators have to be robust and should have faster response. All these requirementsrobustness, high speed, durability- are satisfied by Pixhawk 4 controller.

Pixhawk 4 [25] is advanced autopilot controller. It is based on Pixhawk-project FMUv5 open hardware design and runs PX4 on Nuttx OS [25]. Nuttx is a open source RTOS Real Time Operating System- which uses multithreading for optimised performance of the CPU. Nuttx is a POSIXPortable operating system interface and, being based on linux it is easier and familiar to use.

Pixhawk 4 controller provides support for various communication devices like remote control receiver , GPS, telemetry, and protocols like SPI ,USART/UART ,etc .It has two separate pins for powering I/O and FMU output pins. Dedicated debug ports can be used to analyse any process running on Pixhawk 4 O.S. and debug issues related with various sensors and actuators.

Pixhawk 4 controller houses two independent processors for handling I/O and FMU-required for functions other than camera, gimbal, etc. The one used for I/O is STM32F100 (32 Bit Arm Cortex-M3, 24MHz, 8KB SRAM) and FMU uses STM32F765 (32 Bit Arm Cortex-M7, 216MHz, 2MB memory, 512KB RAM). List of sensors and actuators have been given in the table below [25]

Table 5.1: Pixhawk 4 Specifications

Component	**Model**
Main FMU Processor	STM32F765
IO Processor	STM32F100
Primary Accel/Gyro	ICM-20689
Secondary Accel/Gyro	BMI055
Magnetometer	IST8310
Barometer	MS5611
Power module output	4.9-5.5V
USB Power Input	4.75-5.25V
Servo Rail Input	0-36V
Weight	15.8gm
Dimensions	44x84x12mm

Apart from onboard sensors on the Pixhawk 4 controller , some more noble features which makes it a easy-to-use flight controller are -

1. Dedicated GPS connection.
2. 16 PWM pins (8 - I/O , 8 - FMU).
3. Support for I2C , SPI, UART/USART and CAN protocols.
4. Dedicated interface for various R/C receivers like CPPM, Spektrum / DSM and S.Bus with analog / PWM RSSI input.
5. Two seperate power connectors for I/O and FMU processors.

Figure 5.1: Pixhawk 4 Controller

5.2 Pixhawk Power board

The Power distribution board is used for delivering power to the ESC's and also to the servo motors. PWM signals from the I/O out pins and FMU out pins are connected to the power distribution board which are then given to the corresponding ESC's. The Power board contains pin outputs for FMU, capture and ADC input .See figure 5.2 for connection of FMU, I/O, Capture ADC-input, etc. The PWR1 and PWR2 provide power pins for powering the Pixhawk via the battery for powering the Pixhawk 4. Power Distribution board by Holybro [26] is shown in the figure 5.2

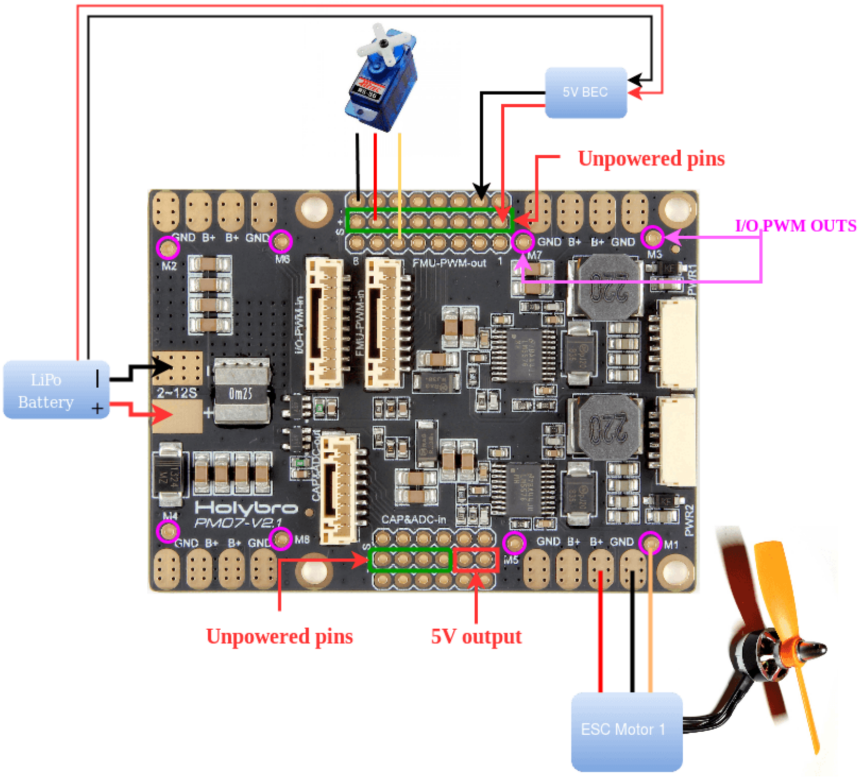


Figure 5.2: Power Distribution Board by holybro

5.3 Modelling Using Embedded Coder Support Package For PX4 Autopilots

For testing the control algorithm, it has to be compiled and burnt/loaded onto Pixhawk 4 controller. MATLABs Embedded Coder Support For PX4 Autopilots [27] helps in building and uploading the model to the Pixhawk 4 controller. Embedded Coder Support For PX4 Autopilots provides ready-made blocks for interacting with the controller. With these blocks a user can get data about the real time attitude and altitude of the quadrotor. Processing this data and using the control algorithm function made with matlab functions , actuator commands - PWM in case of ESCs- can be sent. The model used for testing control algorithm will have an RC controller , which will give commands about roll, pitch, yaw, height- and then these commands will go through some processing and then finally will give commands to the actuators.

Apart from Embedded Coder Support For PX4 Autopilots being easy-to-use and helps in saving time, the prominent reason for using it is that it has inbuilt embedded coder which has the ability to convert the model from Simulink model to source files in ANSI/ISO C files which can then be incorporated by the Pixhawk toolchain to built required firmware (here built for FMU-v5).

Pixhawk Simulink blocks allows user to access sensor data in real time using external mode option. With this capability a user can analyse the sensor data and improve algorithm in real time.

For this project , Embedded Coder Support For PX4 Autopilots (issued in April 2019) is used. The firmware used for building targets is based of the forked version of official Pixhawk Firmware available online on Github .

Note :- Earlier Pixhawk Hardware used Pixhawk Pilot support (PSP). Pixhawk PSP v2.1 -released for MATLAB 2016- is used for targets having firmware FMUv2. Later releases i.e. Pixhawk PSP v3.0.4 offer support for Pixhawk FMUv2 and FMUv3 but does not officially support FMUv5. This project initially explored Pixhawk PSP v3.0.4 for flight testing but couldn't build and upload complete model. Embedded Coder Support Package For PX4 Autopilots officially supports Pixhawk 4 which hosts FMUv5.

5.4 Establishing the toolchain and building Firmware

Toolchain is set of distinct software development tools that are linked (or chained) together specific stages such as GCC, etc. Mostly, the toolchain used for embedded development is cross toolchain or more commonly known as a cross compiler. All the programs (like GCC) run on a host system of a specific architecture (such as x86, x64), but they produce binary code (executables) to run on a different architecture (for example, ARM). This is called cross compilation and is the typical way of building embedded software.Various Toolchains for Pixhawk are given in [28]

The toolchain used for compiling and building using matlab is Pixhawk v3.0.4. The toolchain is installed as a integral step while installing Embedded Coder Support Package for PX4 Autopilots. Steps involving installation of support package are .

1. Install the Embedded Coder Support For PX4 Autopilots Add-on using Matlab Package installer.
2. Enable Windows subsystem for linux (WSL) and setup Ubuntu 16.04 environment.
3. Once the WSL is setup , execute the `windows_bash_shell` provided in `https://github.com/mathworks/PX4-Setup`.
4. Once the installation is complete, toolchain is established and then firmware can be downloaded using the Ubuntu Shell.
5. After downloading and validating the Firmware, bash , python using the UIs validate command, the user has to select the Cmake configuration for corresponding Pixhawk hardware and then build the Firmware for the first time.The firmware appears in '`Firmware/build`' folder or '`Firmware/built_px4fmu-v*_default`', where * firmware version.

Note :- Cmake is used for managing the build process of a software-here simulink app-using a compiler independent method. The Cmake Configuration used for this project is `px4fmu-v5_default` [5.3]

Embedded Coder Support Package For PX4 Autopilots uses the support for Embedded coder to convert the Simulink model to ISO/ANSI C files which can then be incorporated by MathWorks Build Tool Integration (BTI) to allow matlab ARM-GCC compiler to build `px4_simulink_app`. The code generation steps can be given as

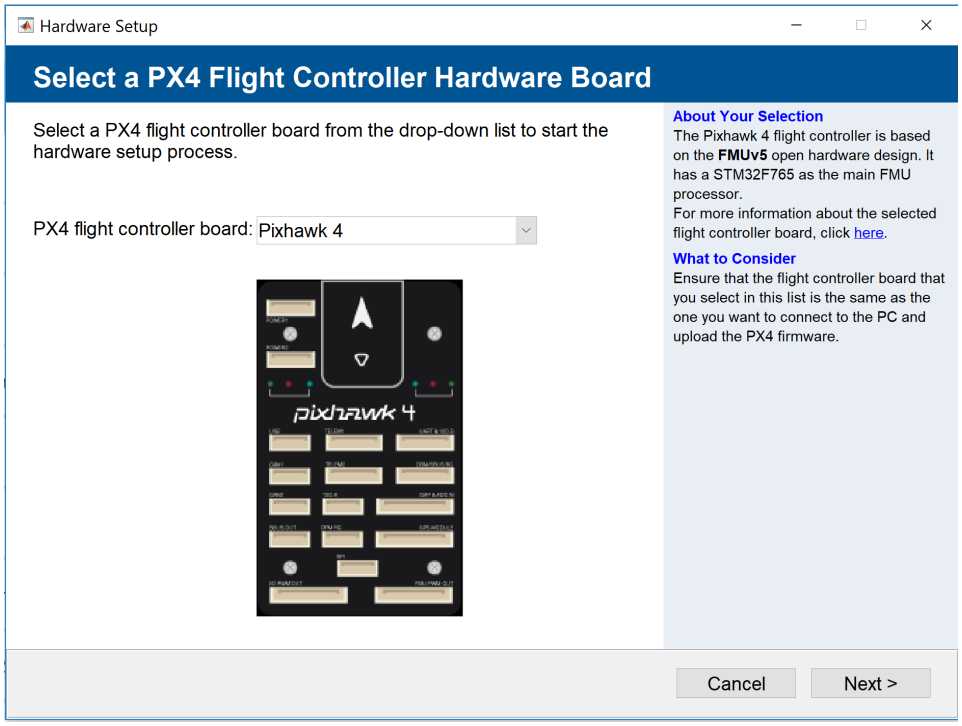


Figure 5.3: Pixhawk PSP UI

1. Code generation of Simulink Model.
2. Transfer generated code to `\px4\Firmware\src\modules\px4_simulink_app` along with a CMakelist.txt which describes the necessary source files, include paths and compiler options inherited from Build Tool Integration.
3. Invoke CMake commands to build the entire firmware. Since we already built most of the Firmware this process should advance more quickly than building it for the first time. The only difference being that CMake will now integrate our newly added Simulink generated code for `px4_simulink_app`. If you are on a Windows 10 machine this part is done within a Ubuntu bash terminal.
4. The firmware image (*.px4 file) will be compiled and placed here `'\px4\Firmware\build_<firmwaretype>\'`.
5. If the build, load and run option was also selected, user will be prompted to plug in the Pixhawk FMU to upload the firmware.

5.5 Qgroundcontrol Planner

QGroundControl [29] provides full flight control and mission planning for any MAV-Link enabled drone. Its primary goal is ease of use for professional users and developers. All the code is open-source source, and anyone can contribute and evolve it.

Qgroundcontrol provides a way of calibrating various sensors on the Pixhawk 4 controller. It provides step by step guide to calibrate each sensor. It can also be used to calibrate ESCs all at a time and store the calibration data into the parameter list. Qgroundcontrol has a unique feature of analysing sensor values in real time using the widget/analyse tab, where a user can analyse various real-time parameters on flight controller.

Another feature which is help full to the developer/professional using Pixhawk hardware is its facility to log data. This can be used to get information on any sensor malfunction and possibly do diagnosis on that. Qgroundcontrol is also be able to connect to the Pixhawk console using the MAVlink - MAVLink or Micro Air Vehicle Link is a protocol for communicating with small unmanned vehicle . Pixhawk console provides an in-depth information on the file-system, testing routines and functions. For more details on calibration refer appendix.

5.6 Models used for testing sensor data

This section contains information about sensor testing using Simulink models. Matlabs Embedded Coder Support Package For PX4 Autopilots provides Simulink blocks[30] which can be used to get data from sensors and give commands to actuator blocks .Some of the blocks are given in figure 5.4

After sensor calibration (see appendix), binding the remote controller and ESC calibration via Qgroundcontrol or RC, it is prudent to test the sensor data, PWM-outputs and BLDC's by making simple models in Simulink and then uploading on Pixhawk 4 hardware. In this section various simple models have been made and tested using the external mode facility in matlab. With this a user can output real-time sensor data on a simple sim blocks like display, scope, etc. External mode can also used to change certain parameters/constants in the model and get its impact on the output. One such model where parameters are dynamically changed is pwm testing model.
For uploading any model, necessary changes in the bootloader script have to be made.

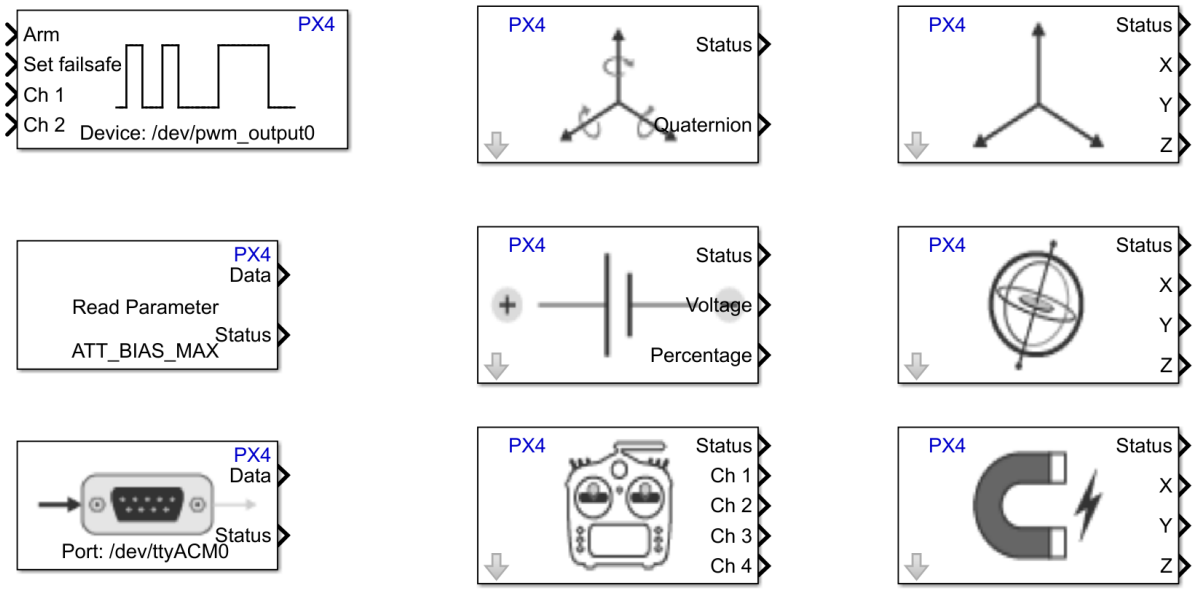


Figure 5.4: Pixhawk Simulink blocks

The inbuilt or default bootloader script calls factory setup flight controllers and various sensor and actuator topics along with topics related to estimators e.g. Extended Kalmam Filter (EKF2 Estimators). It also defines the MAVlink connection and its setting like baud-rate, etc along with setting up of certain peripherals like Telemetry, GPS, PWM blocks (I/O Pwm-out, FMU Pwm-out).

The bootloader can be changed by writing a rc.txt file and placing it in the '/etc' folder on the microsd card. This will change the bootloader setting and make it possible for Simulink to establish a connection with Pixhawk 4 hardware while uploading the script and also connect it serially when in external mode. For more details on rc.txt refer appendix.

5.6.1 Model for testing attitude

Simulink model for testing attitude readings using Embedded Coder blocks is given in figure 5.5

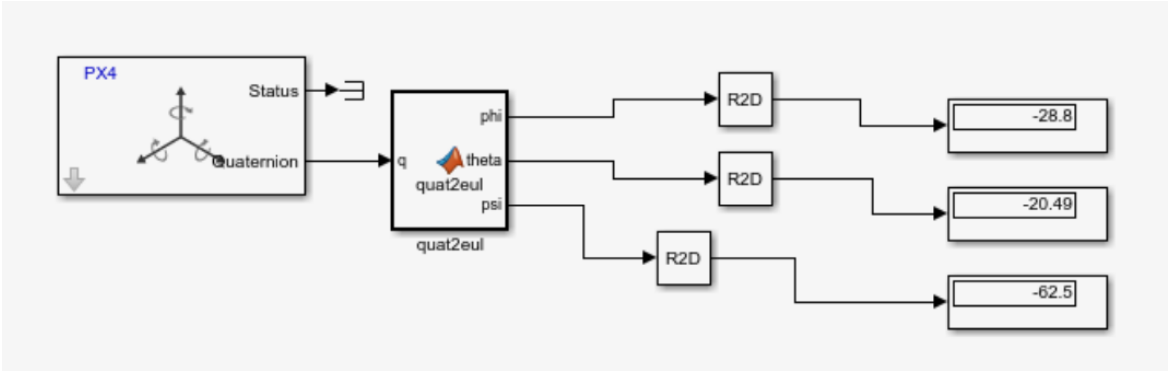


Figure 5.5: Attitude Testing Simulink Model

This model uses the Vehicle attitude blocks and gives values of sensors in quaternion format. This quaternion format is converted into roll (phi), pitch (theta), yaw (psi) by the matlab function '`quad2eul`'.As seen in the model, sensor values are displayed on the display boxes. To check if the sensor values are correct the quadcopter can be tilted 90 degrees and measurements can be checked.

5.6.2 Model for testing PWM-outputs

Once the ESC calibration is done - either via Qgroundcontrol or by remote controller - it is imperative to test the BLDC's for a trial run to see if the interaction of Simulink blocks - PWM block- with the Pixhawk 4 hardware is proper. For this testing, Embedded Coder's Support for PX4 autopilot's PWM block is been used along with a 'uORB read' block which reads the input on the 'I/O PWM out' pins of Pixhawk 4 hardware.

In this model a pwm with duty cycle of 1000 usec at 50hz (showm in figures 5.7 and 5.8) wave is set - which is lowest duty cycle for lowest motor speed as set by the calibration. A boolean 'true' is provided by constant block from simulink to arm the device .When armed, the PWM values provided by the Pixhawk are sent to the I/O pwm pin. When disarmed by setting to boolean 'false' or '0' a duty cycle of 900 usec is set to avoid timeout of the ESC's. 'uORB read' which subscribes to 'Actuator.msg' topic outputs the value of PWM available on 'I/O PWM out' pins. Figure 5.6 shows the complete model for testing PWM. Figure 5.9 shows a pwm with ON-time of 1200usec for 50hz wave . The value of pwm was changed in real-time- one of the advantages of external mode.

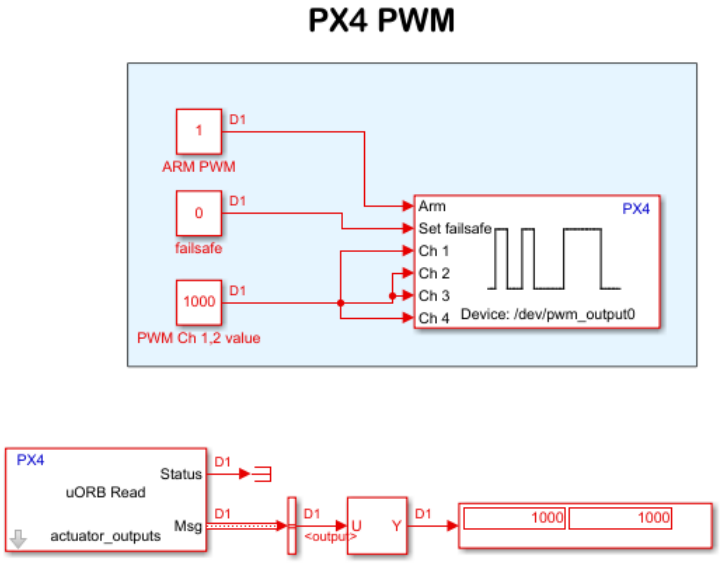


Figure 5.6: PWM Output for model

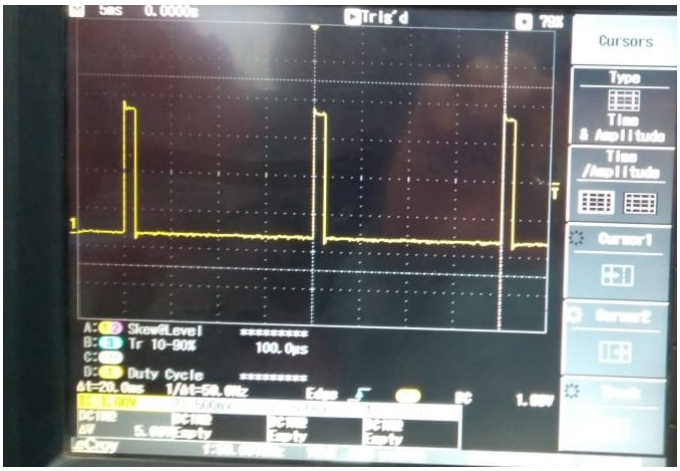


Figure 5.7: PWM Frequency 50Hz

Figure 5.8: PWM with ON-time of 1000usec

Figure 5.9: PWM with ON-time of 1200usec

5.6.3 Model for testing GPS values and measuring relative height using barometer

'uORB read' block from embedded coder helps in getting data from various topics running on pixhawk 4 controller. This models uses Uorb read block to get data on GPS topic.This block is then passes onto bus selector to select what outputs need to get propogated. Finally, display boxes are used to log latitude, longitude, and altitude. The GPS latitude and longgitude have to me multiplied with 1e-7 and altitude by 1e-3 to get the data in degrees and meters respectively.

Note : Initially, GPS output might be zero .This is because GPS module is not locked to some satellite. Usually , a 30 sec time is enough for it to lock on to some satellite. Figure 5.10 shows values GPS testing model.

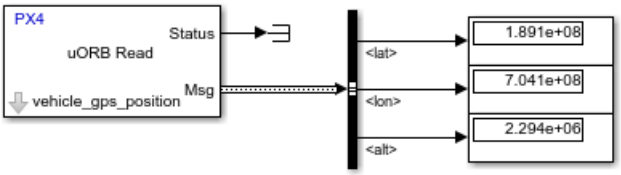


Figure 5.10: GPS showing lat, lon

The values of altitude from GPS, do not have that accuracy and sensitivity to changes in altitude of quadcopter. In order to get accurate a barometer is equipped with pix-

hawk 4. Values from barometer can similarly be taken using uorb read block. Barometer readings are in pressure, temperature and error-counts which have to be converted into height (in meters). This is done by using the Hypsometric equation described in appendix. The model in the figure 5.11 shows usage of barometer and matlab function which is used to convert it to height in meters.

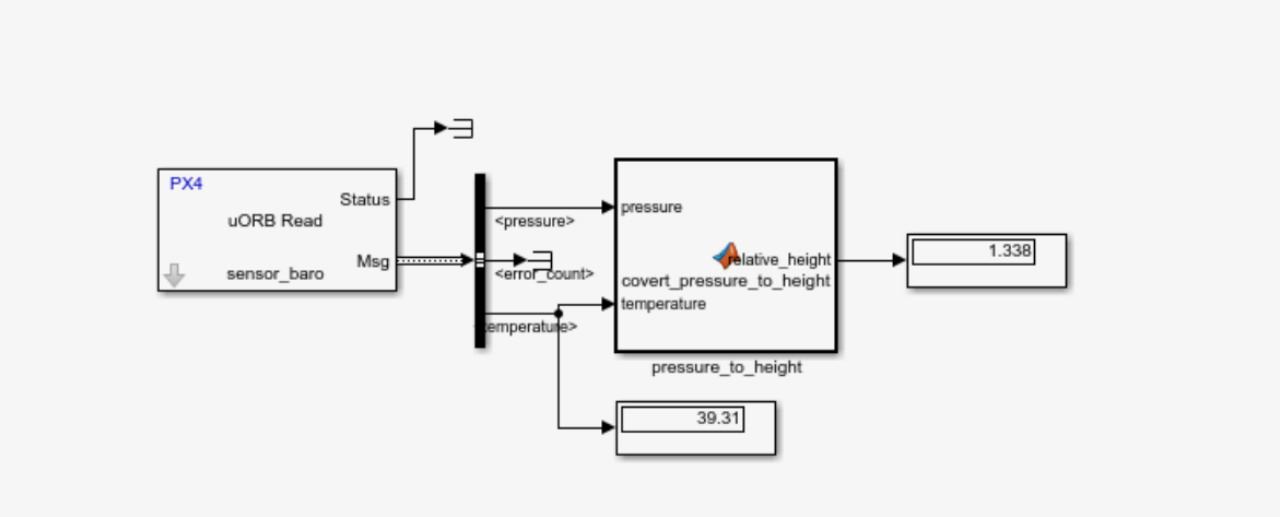


Figure 5.11: Barometer Testing

5.6.4 Model for Validating actuator commands

Taranis X9D plus remote controller will be used to give setpoint inputs to the controller. This remote controller has to be first binded with compatible receiver to transfer data. Once the binding is done and basic four channels- channels 1, 2, 3, 4 for corresponding Thrust, Roll, Pitch, Yaw -are setup then Simulink model for RC is tested. This model consists of 'RC block' which is set to four channels and four display are connected. This model is run in external mode. If the remote controller is successfully binded with receiver and compatible with pixhawk hardware, it will display commanded values from RC. These values vary from 1000 to 2000 (datatype - uint16). If RC values exceed these bounds then trims can be used on RC to correct this error. Figure 5.12 shows simulink model for testing RC

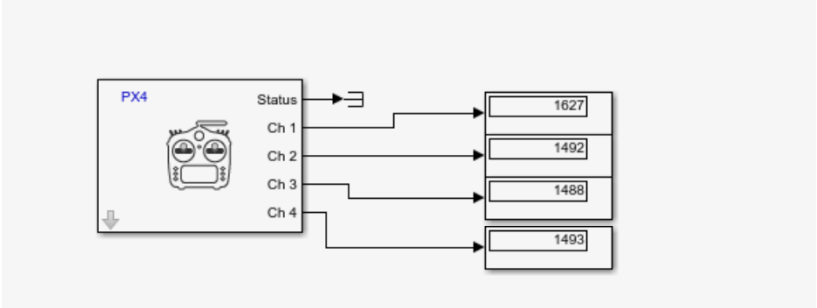


Figure 5.12: Remote Controller Testing

5.7 Models for Validating the control algorithm

5.7.1 Proposed Model For Testing Control Law

After testing the sensor data using the Simulink models mentioned in aforementioned section, They are interconnected to perform the actual flight test. A model containing the sensor blocks along with actuator blocks has been made. In the first model shown in figure 5.13 , an RC-remote controller= has been used to give the following commands i.e. height from the ground, roll, pitch, yaw. The remote controls inputs are such mapped that the maxmimum output of RC does corresponds to 8m and minimum 0m- the transition from 0 to 8 m being linear- in the case of height and -10 to 10 degrees in the case of roll, pitch, yaw angles. These input commands are then feed to the PID or the SMC blocks ,where the error is then feed to the matlab functions which associates it with the control law to be tested. The control law then with the help of mathematical model evaluates the PWM signal to be given to the ESCs. An arming (boolean) value has to be provided to PWM block for driving the PWM output to the MAIN Pwm out i.e. I/O-Out. This arming signal is been given by remote controller. Vehicle Attitude block is used for getting the real time sensor values for roll, pitch, yaw motions. Similarly GPS module or barometer- using 'Sensor_combined' block from Embedded Coder's blocks- can be used to get the relative height. These all sensor values are then provided to the subtractor to get the error and then passed to the PID/SMC block. Refer figure 5.13 for complete model for stabilization . A similar model can be made for SMC.

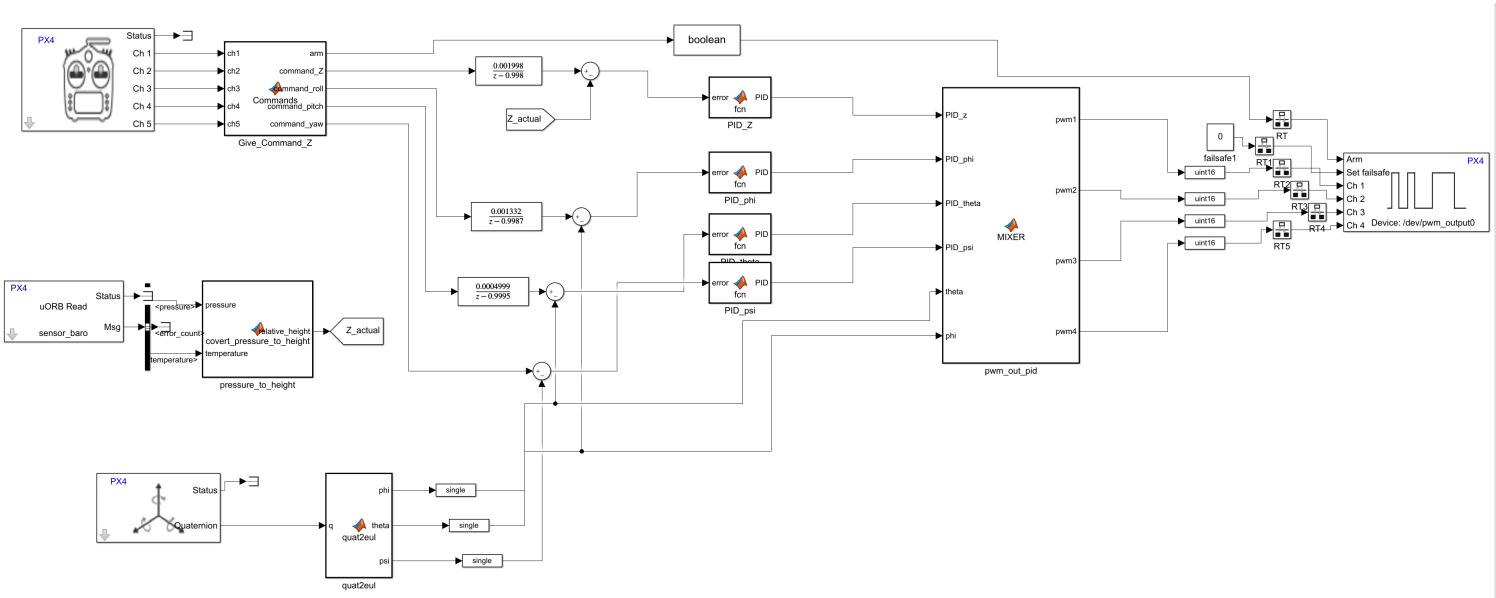


Figure 5.13: Model For Testing Control Law

5.7.2 Proposed Model For Testing Trajectory Control

In order to perform trajectory tracking , data about the position of the quad-copter in space is required. This requirement is satisfied by the GPS mounted onto the quad-copter and connected to Pixhawk 4. The model used in this case uses the Simulinkś GPS block provided by Embedded Coder Support and gets information about the Latitude and Longitude of the quadcopter. In the initialization step , i.e. just when the Simulink model is run on the pixhawk, the matlab function altitude_adjust is so coded as to get and store information regarding the initial co-ordinates (in latitude and longitude) and hence in X-Y. These X-Y initial values are used as reference for the rest of the flight. Thus, when any command for the X and Y trajectory to the quad-copter is given it is done relative to X-Y initial values. Figure 5.14 shows the complete model implemented for trajectory control .

Note : Transition blocks have been used in this model as the sampling for Data acquisition on GPS is different from Sensors . So in order to make it synchronized, rate transition block is used.

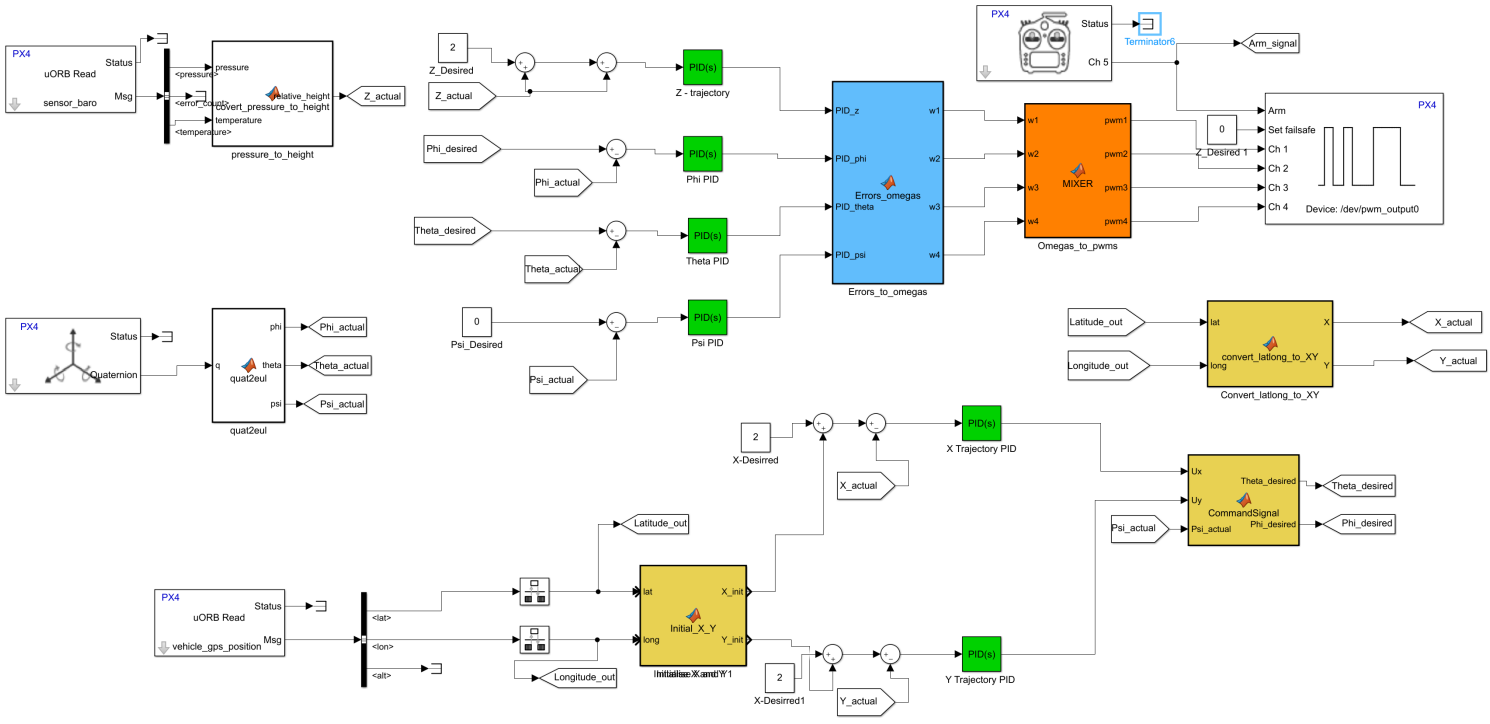


Figure 5.14: Model For Testing Trajectory Control

The control stage for this model is similar with that of the earlier model. The only difference in this model is incorporation of GPS and barometer . The barometer gives values output in terms of pressure . These values have been converted to height and then used in the model. Matlab function 'CommandSignal' takes in the error for the

X-Y motions and then converts into desired theta and phi angles as desribed in 'Control for Quadcopter' section.

5.8 Experimental Results

5.8.1 Flight Review Online Tool

Log files generated by logger on pixhawk can be analyzed using a Flight Log reviewing tool. The log file generated is in .Ulog format and can be opened by certain online tools like Flight Review and offline tools like pyulog, pyFlightAnalysis, FlightPlot, PX4Tools, etc [31]. In this thesis, online reviewing tool- Flight Review -is used. Flight Review is the successor of Log Muncher. It can be used in combination with the new ULog logging format. Some of the key features of Flight Review are -

1. Web-based tool.
2. A User can upload, share and download anywhere.
3. Gives a plethora of plots for flight parameter analysis.

With the help of Flight Review, Flight and controller analysis can be done. Flight Review not only provides information regarding the dynamics of quadcopter, but also GPS locations, Actuator commands, CPU/RAM usage, Parameters set inside Pixhawk Hardware, etc. Such a wide range of logs can help the user to analyse a Flight completely and take decisions regarding the controller performance and sensor data -in case if calibrated wrongly.

5.8.2 Experimental results

One such analysis for PID control using the model described above in Pixhawk Flight Controller Section and Qgroundcontrol's PID is given below. The experiment where carried out at DRDO R&DE facility , Pune with zero to moderate wind conditions. The Graphs shown were taken from Flight Review online tool. Figure 5.15 shows Z trajectory by giving setpoint on RC controller. The gitter in the graph can be attributed to slight changes in RC values and wind gust.

Figure 5.16 shows Roll graph, where the setpoint was zero degrees. With PID, whenever there was a change in the value of roll angle, the PID brought it back.

Figure 5.19 shows plot regarding the CPU/RAM usage. The scale is set as fractions of full capacity. The CPU usage can be seen to be spiking when the controller starts its flight and at stages where correction measure need to be taken.

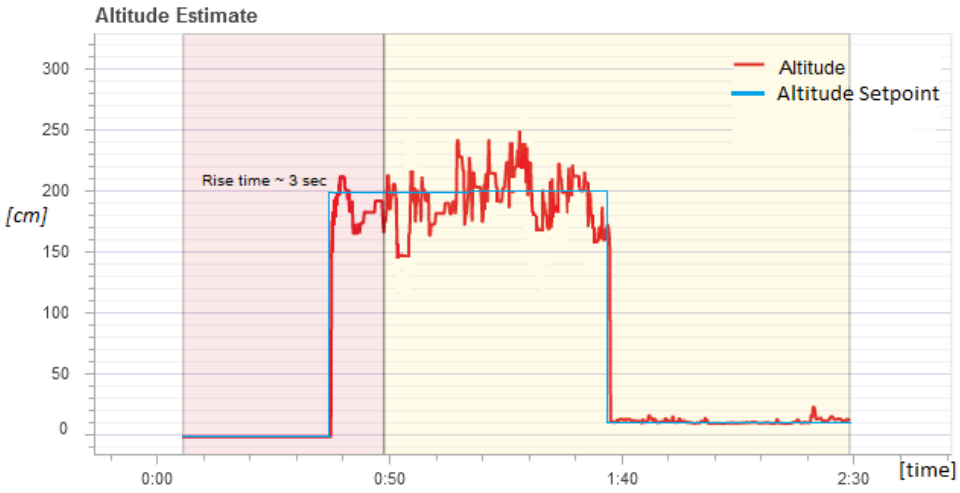


Figure 5.15: Altitude - Z Trajectory

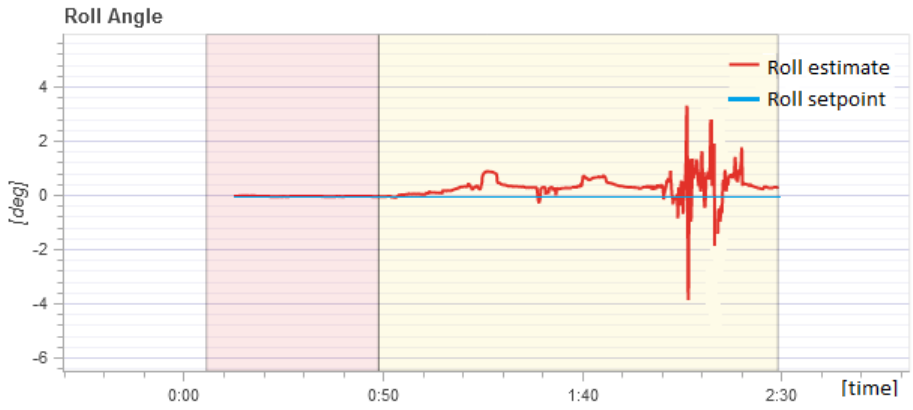


Figure 5.16: Roll angle

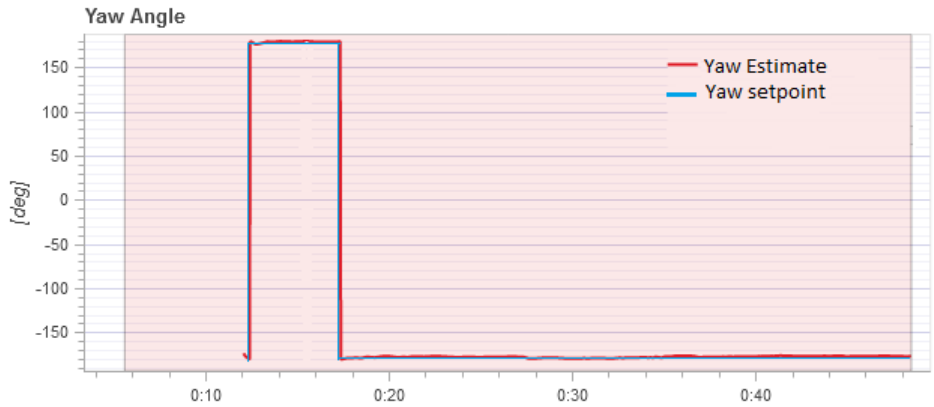


Figure 5.17: Yaw Graph

Similarly figure 5.18 was found to have similar results. Table 5.2 shows experimental and simulation results for altitude setpoint - 2m and deviations in roll pitch to be 2 to 4

degrees.. Figure 5.17 shows the setpoint and actual plot for yaw motion.

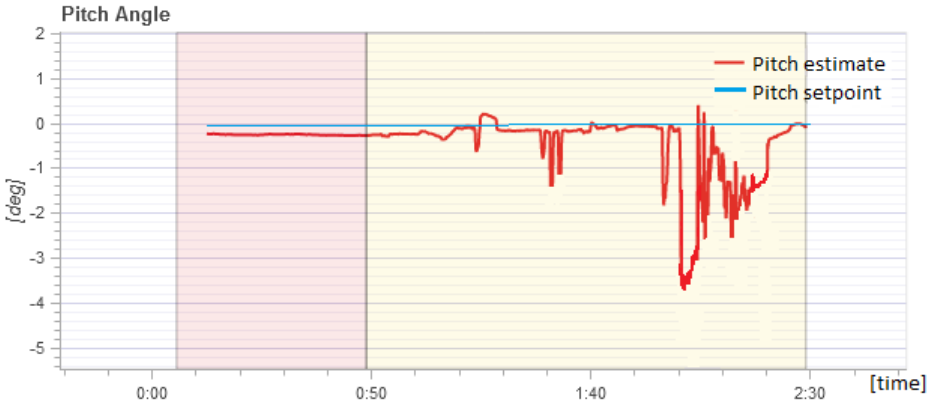


Figure 5.18: Pitch graph

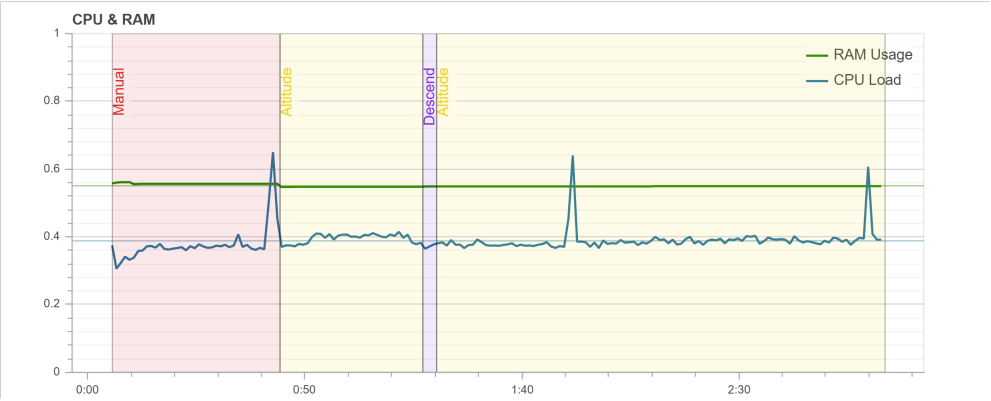


Figure 5.19: CPU/RAM usage while performing flight

Table 5.2: Experimental and Simulation reults

Motion	Experimental	Simulation
Altitude setpoint - 2m	≈ 3sec	≈ 4sec
Roll and pitch deviations for 2-4 degrees	≈ 4.5sec	≈ 3.5sec

5.9 Summary

This chapter explains about Pixhawk 4 hardware and its features . It then goes on explaining various MATLAB components that have been used for modeling the control law. A brief account on establishment of toolchain and its basic working is presented, describing its usefulness in this project. A bit information about Qgrouncontrol planner is given which is used for calibrating sensors and for diagnostic purposes. Various different test models have been discussed and the results for the same have been given in the experimental results section. At the end, the two proposed models for implementing control algorithm and trajectory tracking have been discussed. An idea about implementation of this models with the help of Embedded Coder Support For PX4 Autopilots blocks is briefly given in this model. Later experimental results on test with PID controller are presented. In the conclusion, problems faced while implementing SMC on Pixhawk using Embbedded Coder Support and the future scope is presented.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The attitude control and trajectory tracking of quadrotor using PID controller and FOSM controller was simulated in the MATLAB-Simulink environment and practical tests were carried out using a PID controller.

1. The mathematical model of the quadrotor has been developed in Chapter 2, but practically there are many unmodelled dynamics and disturbances that affect the system dynamics. Thus, a controller that can compensate for unmodelled dynamics is required.
2. Initially, classical PID control has been examined. The PID was tuned to give the desired performance in the absence of disturbance. But, it can be seen that with addition of disturbance, PID fails to give the desired response. Thus, there arises a need to examine robust controllers. A modification to PID has been proposed which makes the system insensitive to external disturbances. A First Order Sliding Mode Controller has been developed and simulated.
3. SMC requires significantly more amount of numerical processing as it involves double derivatives. Such numerical processing is intensive task and gets difficult in computing as the control law becomes complex.

6.2 Future Scope

The following section gives information regarding the future scope for this project :

1. SMC control can be analysed for numerical processing. A less numerically intensive algorithm can be developed or certain numerical methods can be used to reduce the processing required by the Pixhawk.
2. Sensor are susceptible to failure because of vibrations produced by motors, wind, etc and changes in the environment. To tackle this, fault tolerant systems needs to be designed.
3. Double PID loops- inner consisting of angles and outer consisting of rates- can be developed to get quicker and more stable response.
4. The following control algorithm is designed for autonomous flying. Collision detection and avoidance methods can be implemented.

Chapter 7

Publications

Title : - Control System Design Of a Quadcopter Using Pixhawk 4 controller

The above paper is being communicated to 5^{th} Indian Control Conference to be held at Indian Institute Of Technology Hyderabad, Hyderabad, India.

Appendix A

Appendix

A.1 Qgroundcontrol Planner for Calibration

Pixhawk 4 comes with a lot of sensor like gyroscope, accelerometer, barometer, magnetometer, etc. It has two compasses namely primary compass and secondary compass. Primary compass is refered to compass on GPS module and secondary compass is refered to compass in magnetometer on Pixhawk 4 board. Appropriate calibration of these sensors is needed for correct functioning. Qgroundcontrol is an open-source software which is able to connect to PX4 targets using MAVlink connection. Once the connection is established a PX4 Firmware has to be flashed onto it. This is done in Firmware tab in Qgroundcontrol. Once the hardware is setup its time for the sensor calibration. Brief steps to calibrate sensor are : -

1. Open the 'Sensors' Tab.
2. Click on individual sensor. Set the orientation. First on compass then rest of the sensors.
3. Follow instructions regarding the tilting of Pixhawk 4 in various direction and click 'OK' once the calibration is done.
4. Repeat the steps for various sensors.

One way to check if sensor calibration is done is by tilting the quadcopter to various known and measured angle or use a Simulink Matlab model for testing attitude data and check if the value matches with measured value.
Qgroundcontrol is also used to calibrate all ESC's at once. In Qgroundcontrol `'MIN_PWM'` is set to 1000 usec (duty cycle) and `'MAX_PWM'` is set to 1950 usec. For calibration of ESC's initially all the props have to be removed . For precaution, check the power board connection using a multimeter for any short circuit. After that click

on 'Calibrate' button and then connect the battery when prompted by the Qgroundcontrol. A successful ESC calibration message is given by Qgroundcontrol on the window. Once the ESC is successfully calibrated test each BLDC by connecting the Servo pin to I/O pwm-out pins and run the Simulink model for testing PWM provided in the Model testing in Pixhawk Flight Controller section. A sucessfully calibrated Pixhawk 4 hardware screen looks like A.1.

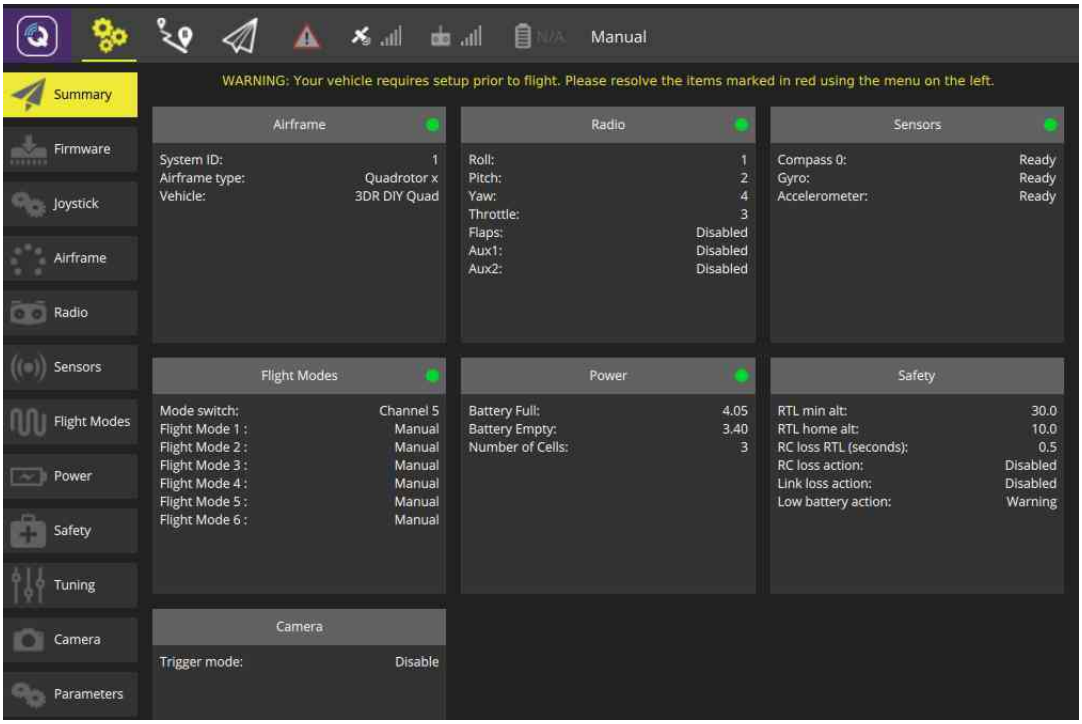


Figure A.1: Qgrouncontrol -All sensors and actuators calibrated

A.2 Bootloader Script

The PX4 startup is controlled by shell scripts. On NuttX it reside in the `'ROMFS/px4fmu_common/init.d'`. When the system starts the bootloader initially calls the rcS file which then calls subsequent files to initial Pixhawk 4 to factory settings for that. rcS maps various ports to there serial port reference in POSIX, sets up various functionalities like logging, etc. It also calls on various sensor and actuator topics and sets up the serial port configuration for various peripherals like GPS, Telemetry, etc. In order for Matlab to communicate with Pixhawk hardware certain processes have to shutdown. This is done by the text file rc.txt . A sample of the text file is shown below.

```
usleep 1000
uorb start
usleep 1000
nshterm /dev/ttyACM0 &
usleep 1000
px4io start
usleep 1000
#commander start
#usleep 1000
#mavlink start -d /dev/ttyS1 -b 115200
#usleep 5000
#dataman start
#usleep 1000
#navigator start
#usleep 1000
sh /etc/init.d/rc.sensors
usleep 1000
#sh /etc/init.d/rc.logging
#usleep 1000
#gps start
attitude_estimator_q start
position_estimator_inav start
usleep 1000
mtd start
usleep 1000
param load /fs/mtd_params
usleep 1000
rgbled start
usleep 1000
px4_simulink_app start
exit
```

A.3 Hypsometric Equation for relative height

Barometer gives pressure and temperature values which have to be converted to height in meters. This transformation is done by hypsometric equation [32]. The hypsometric equation, also known as the thickness equation, relates an atmospheric pressure ratio to the equivalent thickness of an atmospheric layer under the assumptions of constant temperature and gravity. It is derived from the hydrostatic equation and the ideal gas law.

The hypsometric equation is expressed as : -

$$h = z_2 - z_1 = \frac{R.T}{g} \ln\left(\frac{p_2}{p_1}\right) \tag{A.1}$$

where,
h = thickness of the layer [m],
z = geometric height [m],
R = specific gas constant for dry air,
T = mean temperature in kelvin [K],
g = gravitational acceleration [m/s^2],
p = pressure [Pa].

Diagramatically it call be described as A.2 [33]

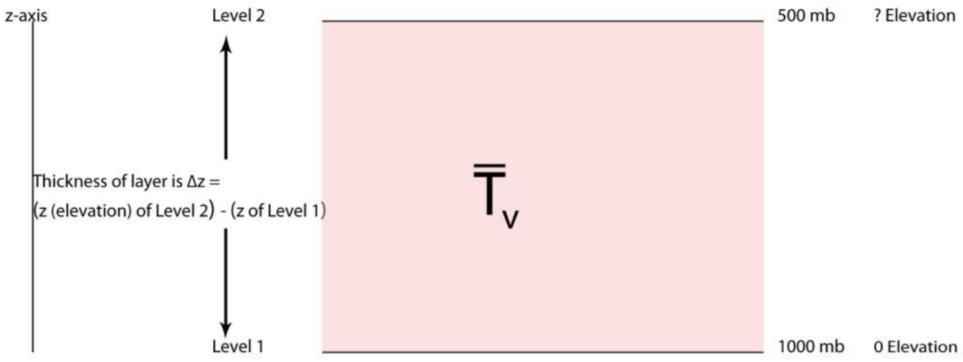


Figure A.2: Hypometric Equation Diagram

Matlab script used for achieving this result is given below,

```
function relative_height = covert_pressure_to_height(pressure, temperature)
R = single(287.058);
g = single(9.81);
temperature = temperature + single(273.15); %convert to kelvin from degrees

persistent itr;
if isempty(itr)
    itr = 0;
end

persistent init_pressure;
if isempty(init_pressure)
    init_pressure = single(944.77);
end

persistent initial_temp;
if isempty(initial_temp)
    initial_temp = temperature;
end

if (itr == 0)
    initial_pressure = pressure;
```

```
23      itr = itr + 1;
24  else
25      initial_pressure = single(init_pressure);
26  end
27
28  if (isreal(log(initial_pressure/pressure)))
29      log_value = log(initial_pressure/pressure);
30  else
31      log_value = 0;
32  end
33
34  init_pressure = single(initial_pressure);
35
36  relative_height = single(((R*initial_temp)/g)*log_value);
```

Listing A.1: Conversion to relative height

In the following matlab script persitent variable are used in matlab to get the initial pressure, initial temperature . The value of gas constant 'R' and 'g' are taken to be variables of datatype (single) 32bit floating type to make it less memory consuming. Temperature is converted from degrees (float32 or single) to kelvin. Condition for some sporadic and undesired values of log are taken care of in the 'if' statement.

References

[1] Y.-C. Choi and H.-S. Ahn, "Nonlinear control of quadrotor for point tracking: Actual implementation and experimental tests," *IEEE/ASME transactions on mechatronics*, vol. 20, no. 3, pp. 1179–1192, 2015.

[2] H. Fernando, A. De Silva, M. De Zoysa, K. Dilshan, and S. Munasinghe, "Modelling, simulation and implementation of a quadrotor uav," in *2013 IEEE 8th International Conference on Industrial and Information Systems*. IEEE, 2013, pp. 207–212.

[3] H. Okazaki, K. Isogai, and H. Nakano, "Modeling and simulation of motion of a quadcopter in a light wind," in *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2016, pp. 1–4.

[4] B. Samir, N. Andr, and S. Roland, "P1d vs lq con-trol techniques applied to an indoor micro quadrotor," in *Proceedings of the 2004 1EEE/RSJ International Conference on In-telligent Robots and Systems. Sendal, Japan: IEEE*, 2004, pp. 2451–2436.

[5] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 4393–4398.

[6] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2247–2252.

[7] ——, "Full control of a quadrotor," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ieee, 2007, pp. 153–158.

[8] V. G. Adir, A. M. Stoica, and J. F. Whidborne, "Sliding mode control of a 4y octorotor," *UPB Sci. Bull., Series D*, vol. 74, no. 4, pp. 37–51, 2012.

[9] Q. Quan, *Introduction to multicopter design and control*. Springer, 2017.

[10] "Aerial robotics," https://www.youtube.com/watch?v=nLlLEw_hiNc&list=PLblGgzWkqSqM7IWsgjDetdzZDS1NbkTnd.

[11] I. Nagrath and M. Gopal, *Textbook of control systems engineering (Vtu)*. New Age International, 2008.

[12] L. Umanand, "Switched mode power conversion."

[13] D. Liu, Z. Tang, and Z. Pei, "Variable structure compensation pid control of asymmetrical hydraulic cylinder trajectory tracking," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[14] M. Leclerc, "Nonlinear control system design - mit-2.152."

[15] "Q450 quadrotor frame with integrated pcb for easy wiring," https://robokits.co.in/drones-quad-hexa-octa-fpv/frames-misc.-parts/r450-quadrotor-frame-with-integrated-pcb-for-easy-wiring.

[16] "Brushless motor datasheet," https://www.rhydolabz.com/documents/26/BLDC_A2212_13T.pdf.

[17] "Brushless motor speed controller esc 30a," https://robokits.co.in/motor-drives-drivers/dc-motor-driver/brushless-motor-speed-controller-esc-30a.

[18] "Counter rotating propeller 1045/1045r for quadrotor-multirotor," https://robokits.co.in/drones-quad-hexa-octa-fpv/brushless-motor-propeller-esc/propeller/5-to-10-inch/counter-rotating-propeller-1045-1045r-for-quadrotor-multirotor.

[19] "Orange 3000mah 3s 40c/80c lithium polymer battery pack (lipo)," https://robu.in/product/orange-3000mah-3s-40c80c-lithium-polymer-battery-pack-lipo/.

[20] "Frsky 2.4ghz accst x8r manual," https://www.frsky-rc.com/wp-content/uploads/2017/07/Manual/X8R.pdf.

[21] "Frsky 2.4ghz accst taranis x9d manual," https://www.frsky-rc.com/wp-content/uploads/Downloads/Manual/X9DP/X9D%20PLUS-manual.pdf.

[22] "Taranis x9d plus," https://www.frsky-rc.com/product/taranis-x9d-plus-2/.

[23] "Pixhawk4 gps module," http://www.holybro.com/manual/Pixhawk4-GPS-Quick-Start-Guide.pdf.

[24] "Neo8," https://www.u-blox.com/sites/default/files/NEO-M8_DataSheet_%28UBX-13003366%29.pdf.

[25] "Pixhawk 4," https://docs.px4.io/en/flight_controller/pixhawk4.html.

[26] "Pixhawk 4 wiring quick start," https://docs.px4.io/en/assembly/quick_start_pixhawk4.html.

[27] "Installing files and code," https://in.mathworks.com/help/supportpkg/px4/index.html?s_tid=CRUX_lftnav.

[28] "Installing files and code," https://dev.px4.io/en/setup/dev_env.html.

[29] "Qgroundcontrol dev guide," https://dev.qgroundcontrol.com/en/.

[30] "Simulink blocks from embedded coder support," https://in.mathworks.com/help/supportpkg/px4/modeling-px4.html.

[31] "Flight log data review tools," https://docs.px4.io/en/log/flight_log_analysis.html.

[32] "Hypsometric equation," https://en.wikipedia.org/wiki/Hypsometric_equation.

[33] "Hypsometric equation more details," http://tornado.sfsu.edu/Geosciences/classes/e260/Hypsometric/Hypsometric%20Equation.pdf.