DESIGN OF EXTENDED KALMAN FILTER OPTIMIZED FUZZY PID CONTROLLER FOR A QUADCOPTER IN THE EVENT OF ONE ROTOR FAILURE

JACKSON OCHIENG OLOO

MASTERS OF SCIENCE
(Electrical Engineering)

JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

2021

Design of Extended Kalman Filter Optimized Fuzzy PID Controller for a Quadcopter in the Event of One Rotor Failure

Jackson Ochieng Oloo

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science of the Jomo Kenyatta University of Agriculture and Technology

2021

DECLARATION

This thesis is my original work and has not been presented for a degree in any other University.

Signature……. Date……..

Jackson Ochieng Oloo

This thesis has been submitted for examination with our approval as University Supervisors.

Signature……. Date……..

Prof. S. Kamau, Dr.-Ing.

JKUAT, Kenya

Signature……. Date……..

Prof. S. Kang'ethe, PhD

JKUAT, Kenya

DEDICATION

To my wife and daughter,

ACKNOWLEDGEMENT

I wish to thank the Almighty God for the grace in completing this work. I also like to thank my supervisors for guiding and encouraging me through this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

BLDC	Brushless Direct Current
COG	Center of Gravity
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FBL	Feedback Linearization
FDD	Fault Detection and Diagnosis
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Controller
FPID	Fuzzy Proportional Integral Derivative
FTC	Fault Tolerant Controller
GPS	Global Positioning System
GS-PID	Gain Scheduled Proportional Integral Derivative controller
IMU	Inertial Measurement Unit
LQR	Linear Quadratic Regulator
MEM	Microelectromechanical Systems
MFs	Membership Functions
MPC	Model Predictive Control
MRAC	Model reference adaptive controller

PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing

ABSTRACT

A quadcopter is an Unmanned Aerial Vehicle (UAV) that uses four rotors in a cross or plus configuration for lifting and propulsion. Its basic motion is generated by varying the speeds of all the four rotors. It is an under actuated vehicle with unstable dynamics since it is a 6 degree of Freedom (DOF) device with only four actuators. Moreover, these quadcopters have hardware redundancy limitations thereby calling for design of reliable control systems for efficient performance. It is therefore a challenge to maintain quadcopter stability when one of the rotors is faulty, this is as a result of further under actuation. Previous research in literature assumes the quadcopter is equipped with a fault diagnosis, detection and isolation unit, that continuously assesses the state of the quadcopter or that the fault has already been detected and isolated. Switching is therefore expected between the healthy state controller and a faulty state controller. This procedure is complex and therefore prolong the stabilization time. Most of the available literature also investigated partial rotor failures of up to 20 % only.In this work, an Extended Kalman Filter Fuzzy Proportional, Integration and Derivative (PID) controller is applied for control of a quadcopter in the presence of 100 % fault on a single rotor. The system does not use a fault diagnosis and detection system and only employs a single controller. A fuzzy system is used to tune the PID controller gains while the tuning of the fuzzy Membership Functions (MF) is performed using an Extended Kalman Filter (EKF). The optimized Fuzzy PID controller constrains the system to its set point, returning the system to its stable hovering position. This is accomplished by varying simultaneously the velocity of the three fault free rotors. By computing the adaptive PID gains in real time, the effects of rotor failure are compensated. The controller stabilizes the attitude and altitude of the quadcopter and enables it to make a safe landing without causing further damages to the quadcopter structure.

CHAPTER ONE

INTRODUCTION

1.1 Background

Quadcopter Unmanned Aerial Vehicles (UAVs) consist of four separately controlled rotors which work together to provide actuation. The rotors are set up such that two rotors assume clockwise rotation and the other two counterclockwise (Rich & Elia, 2012).

Figure 1.1: A typical quadcopter configuration

Recently, the use of small UAVs for various civilian and military applications has been of a particular interest (Girard et al., 2004). These applications include package delivery, aerial imagery, surveillance, and structural inspection; a common aspect is that these tasks are either in remotely inaccessible locations and require dangerous maneuverability or are in unfriendly environments in case of military operations. Several different UAV platforms exist that have the potential to solve these problems such as fixed-wing airplanes and multirotor aircrafts. A quadcopter has advantages over the fixed wing UAVs in that it has Vertical Take-off and Landing (VTOL) and can perform maneuvers. Its advantage over other rotary UAVs, such as a helicopter, is that it is mechanically simple; a quadcopter does not need a complex set of mechanical

linkages to alter rotor blade angles. Quadcopter UAVs do not require a tail rotor allowing it to direct all UAV power to producing thrust. The quadcopter is therefore able to support payload capacity in relation to its weight. The recent technological advancements in the field of Microelectromechanical Systems (MEMs) sensors have allowed for these sensors to become smaller, cheaper, and lighter, with increased sensing resolution thereby reducing the cost and complexity of quadcopter UAV systems.

However, a quadcopter, being a 6 DoF device, has only four actuators making it under actuated as well as nonlinear and unstable system (Wu & Liu, 2018). Therefore, for actuation of lateral and longitudinal motion by the rotor thrusts, the entire vehicle must angle in one direction or another. This action however, is potentially disadvantageous since it confines the dynamics of the UAV as it cannot move back and forth or side to side acceleration and maintain a given orientation at the same time. It might flip over.

An Extended Kalman Filter (EKF) optimized Fuzzy PID controller is used to stabilize the altitude and the attitude of the quadcopter when one rotor fails to enable it land safely without further structural damages.

1.2 Problem Statement

With autonomous quadcopters, the system gets unstable for a faulty actuator thereby crashing the UAV immediately. Military and commercial application quadcopter UAVs around the world are costly in terms of construction, a fault recovery controller system needs to be installed on these UAVs so that the vehicle executes a landing maneuver without further damages (Jihad, 2013). Moreover, when a quadcopter UAV under mission crashes in an unfriendly territory, data collected might be compromised or even the delivery package lost.

It is therefore important that a quadcopter fault tolerant flight control system is designed to enable the UAV reach a point in space in the event that one of the rotors is lost.

1.3 Justification

The failure rate of a commercial engine is about $1/10^5$ flight hours while quadcopter UAV systems a rotor failure rate of about $1/10^2$ flight hours. These quadcopter UAVs

have therefore a rotor failure rate of about 30% (Ballenger, 2013; Clough, 2005; De Francesco & De Francesco, 2015; Paggi et al., 2015; Schmidt & Parker, 1995). Today, quadcopters are used for surveillance and gathering of information ranging from the search and rescue of disaster victims to monitoring the condition of the power lines. A fault tolerant quadcopter is capable of crash-free landing thereby preventing further damage and repair cost. Quadcopter UAVs can be designed to hold light payloads such as medical supplies and lightweight food materials delivering them to remote locations unreachable by normal planes. The proposed technique is cheaper and simpler to implement. It mitigates the fault effect on the UAV by appropriately adjusting the controller gains thereby restoring the system to its hovering position. With this fault tolerant controller, a quadcopter will still make a safe landing with the deployed package or data collected without loss.

1.4 Objectives

1.4.1 Main objective

To design an Extended Kalman Filter optimized Fuzzy PID Controller for stabilization and safe landing of a quadcopter in the event of rotor failure.

1.4.2 Specific Objectives

i. Develop a procedure for computation of Roll, Pitch and Yaw angles using signal fusion.
ii. Design an EKF optimized Fuzzy-PID controller for optimal PID gains to enable stabilization of the quadcopter.
iii. Simulate the closed loop system using MATLAB/SIMULINK and compare conventional PID, Fuzzy-PID and EKF optimized Fuzzy-PID simulations.
iv. Implement a prototype of the designed controller into a quadcopter control board.

1.5 Scope

The research is focused on developing a controller that enables a quadcopter under actuator fault to stabilize and perform safe landing using the remaining three healthy rotors. To achieve this, orientation sensors are used to give feedback signals which are then converted into angles.

These angles are then fed into a microcontroller for referencing to obtain tracking errors. An EKF algorithm is used to optimize FLC parameters based on the errors obtained. The output of the FLC is then used to optimize the PID gains in real time to ensure adaptability of the system during maneuvers.

1.6 Contribution of Thesis

Most of the research work in literature approach the issue of quadcopter stabilization during rotor fault by application of a Fault Diagnosis and Isolation (FDI) system. This necessitates the use of more than one controller which then necessitates switching between the controllers depending on the rotor's effectiveness. This prolongs the fault recovery time and it is also expensive. (Liu et al., 2016) proposed EKF-FLC system that regulates the gains of a designed state-feedback tracking controller for partial loss of effectiveness, up to 20 % actuator failure. The controller required a precise mathematical model of the quadcopter under actuator fault and did not address the issue of noise in practical systems. Figure 1.2, 1.3 and 1.4 show the quadcopter recovery duration under only 25% rotor failure.

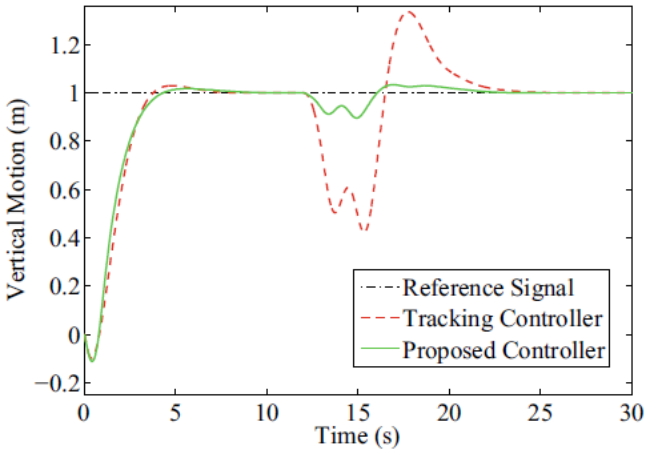

Figure 1.2*:* Stabilization time of approximately 11s

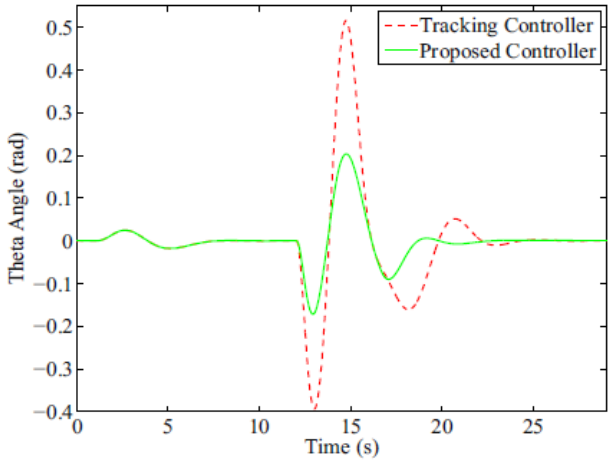

Figure 1.3: Desired Pitch angle achieved after 10s

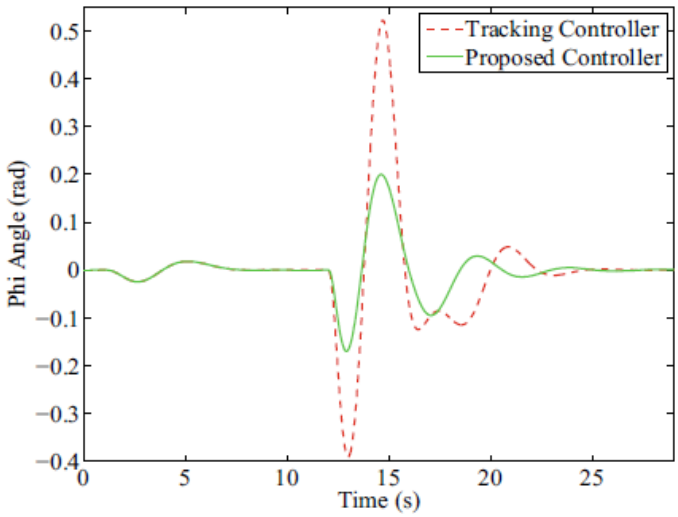

Figure 1.4: Desired Roll angle achieved after 10s

This Thesis presents an EKF optimized FLC that regulates the gains of a PID controller, giving it dynamic response under 100 % rotor failure with a minimum fault recovery time of about 2 $seconds$. The PID is easy to implement due to the fact that it does not rely on an accurate mathematical model of the system under control. In practical applications, motor vibrations introduce noise in accelerometer readings for feedback signals as shown in Figure 1.5. In this work, the noise is filtered by fusing signals from an accelerometer and gyroscope using Kalman filter.

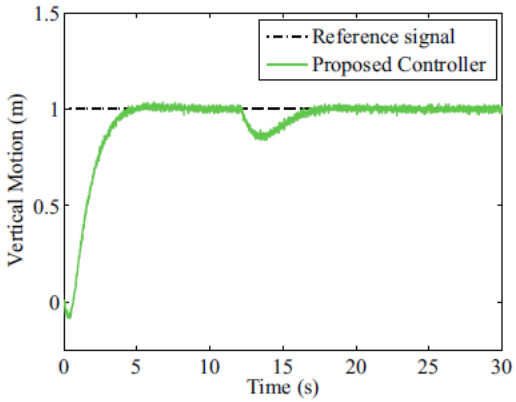

Figure 1.5: Altitude response with noise

The EKF technique tunes the parameters of the MFs of the fuzzy controller depending on the working condition of the system .Then based on the auto tuned MF parameters, PID gains are readjusted to properly compensate for the errors incurred due rotor failure. A Fuzzy Logic System (FLS) automatically tunes the PID gains, where the offset from the target (error) and the rate of change of the tracking error are used in the FLS to make the system return to the expected working condition (desired target) in the event that one of the rotor fails. In this way, the need for fault identification and diagnosis is eliminated.

1.7 Organization of Thesis

This thesis is organized into five chapters as follows: Chapter One entails the general introduction consisting of background information on what quadcopters are, problem statement for the study and justification, research objectives and the scope of work. Chapter Two deals with the Literature Review where general discussion of PID, Fuzzy and Extended Kalman Filter are covered. Related works are also reviewed in this chapter. Chapter Three deals with quadcopter dynamic system modelling and design of PID, Fuzzy PID and optimized Fuzzy PID controllers. It also covers the implementation of the control algorithms into a developed quadcopter UAV prototype. Chapter Four presents the results obtained from simulations and performance of the developed prototype. Chapter Five summarizes the research with conclusions and also presents recommendations from the problems under study.

CHAPTER TWO

LITERATURE REVIEW

This chapter discusses general quadcopter dynamics and the available control techniques. Theory on angle estimation using signal fusion is also described. Summary of research gaps is also covered.

2.1 Quadcopter Motion Overview

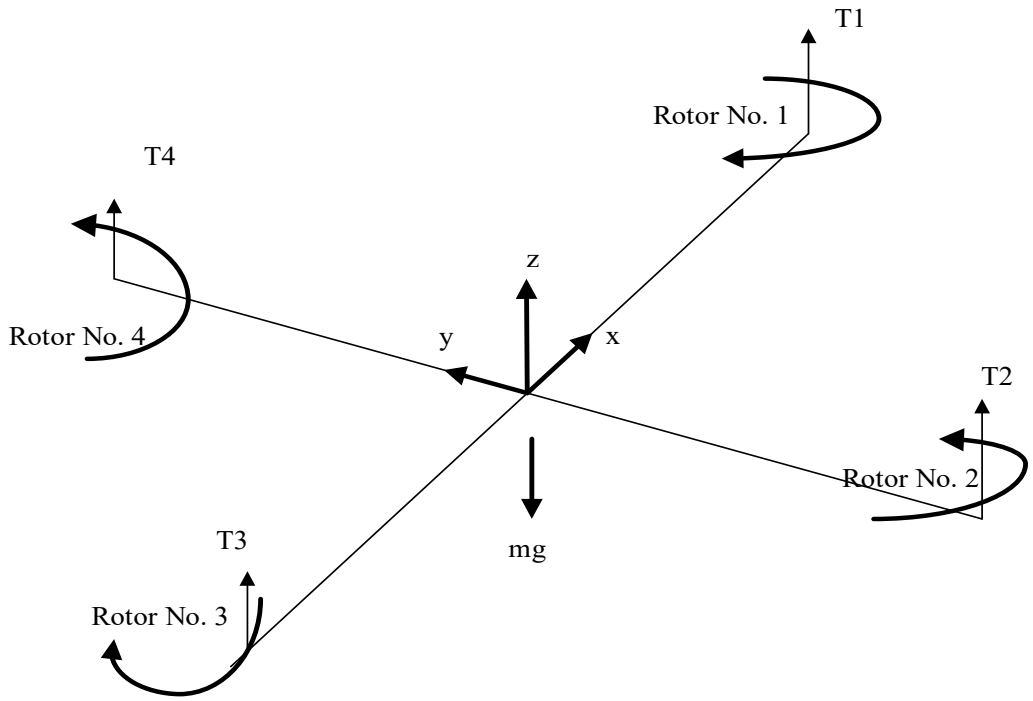

Figure 2.1: Quadrotor free body diagram.

Quadcopters UAVs consist of four independently controlled rotors as shown in Figure 2.1. Rotors (1, 3) rotate clockwise and the other two (2, 4) counterclockwise. Control and stability of the UAV system is attained through differential control of the force generated by each rotor (Ivan, 2012).

2.1.1 Roll Motion

The roll axis travels down the center of the quadcopter from the front to the rear. The roll motion is accomplished by speeding up the left rotors and at the same time lowering the right rotors' speed, or vice versa (Ivan, 2012). Roll angle φ, is controlled by slowing down motor 4 while speeding up motor 2 or vice versa

2.1.2 Pitch Motion

The pitch axis travels in the right to left fashion. This motion is achieved by increasing or decreasing the front rotors' speed and simultaneously decreasing or speeding up the back rotors. This results in pitching up or down of the nose due to the differential on the pitch axis. Pitch angle,θ, is controlled by speeding up motor 3 while slowing down motor 1 or vice versa

2.1.3 Yaw Motion

This motion is achieved by speeding up or decreasing the speed of the front-rear rotor and simultaneously decreasing or speeding up the left-right rotors. Speeding up or decreasing the speed of two opposite motors gives the desired rotation, clockwise or anticlockwise. A net torque is always induced on the quadcopter UAV leading to a yaw angle change for decreasing the speed of the clockwise spinning actuators or speeding up the counter clockwise actuators. Since the quadcopter does not have a tail rotor to control the yaw movement, it becomes an under actuated system. This implies that forward/backward and left/right movements are coupled with pitch/roll motions respectively and are therefore controlled through them (Razinkova et al., 2014). Yaw angle,ψ, is controlled by increasing speeds to motors 1,3 while slowing down motors 2,4 or vice versa.

2.1.4 Hovering

This is also the altitude motion and is controlled by simultaneously speeding up or decreasing all the rotors' speed by the same amount resulting in a vertical force. The quadcopter UAV is raised or lowered accordingly (Ivan, 2012). When an equal thrust is achieved for all actuators, the quadcopter UAV either holds in steady hover or increases/decreases altitude depending on actual thrust value.

2.2 Quadcopter Kinematics

The relationship between the vehicle's position and velocity can be described using kinematic equations. The quadcopter UAV has 6 DoF that can be described using into two motions (Gibiansky, n.d.):

- Translational motion which occurs in $x, y,$ and z directions.
- Rotational motion named as roll (about x axis) $(\emptyset)$, pitch (about y axis) (θ) and yaw (about z axis)(φ).

The quadcopter UAV in 6 DoF has location and attitude which are identified using the reference frames. For the evaluation of quadcopter equations of motion, there are three main frames of reference (Raza & Gueiaeb, 2010):

i. The Inertial frame is an earth-fixed coordinate system such as the base station where the origin is located on the ground. Conventionally, the $x-$axis points towards the north, the $y-$ axis points towards the east, and the $z-$axis points towards the center of the earth as shown in Figure 2.2.

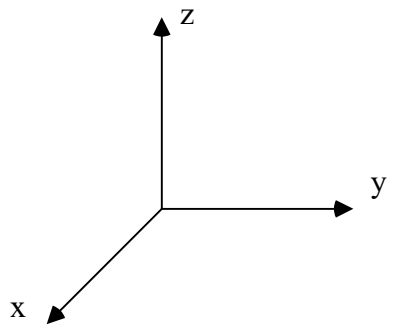

Figure 2.2: Inertial frame of reference

ii. The Body frame has its origin located at the center of gravity of the quadcopter UAV with its axes aligned such that the $x-$axis is along the arm with front motor, the $y-$axis is along the arm with right motor, and the $z-$axis is a cross product of the $x-$axis vector and $y-$axis vector.

iii. The vehicle frame is similar to the inertial frame except that the origin is located at the center of gravity of the craft.

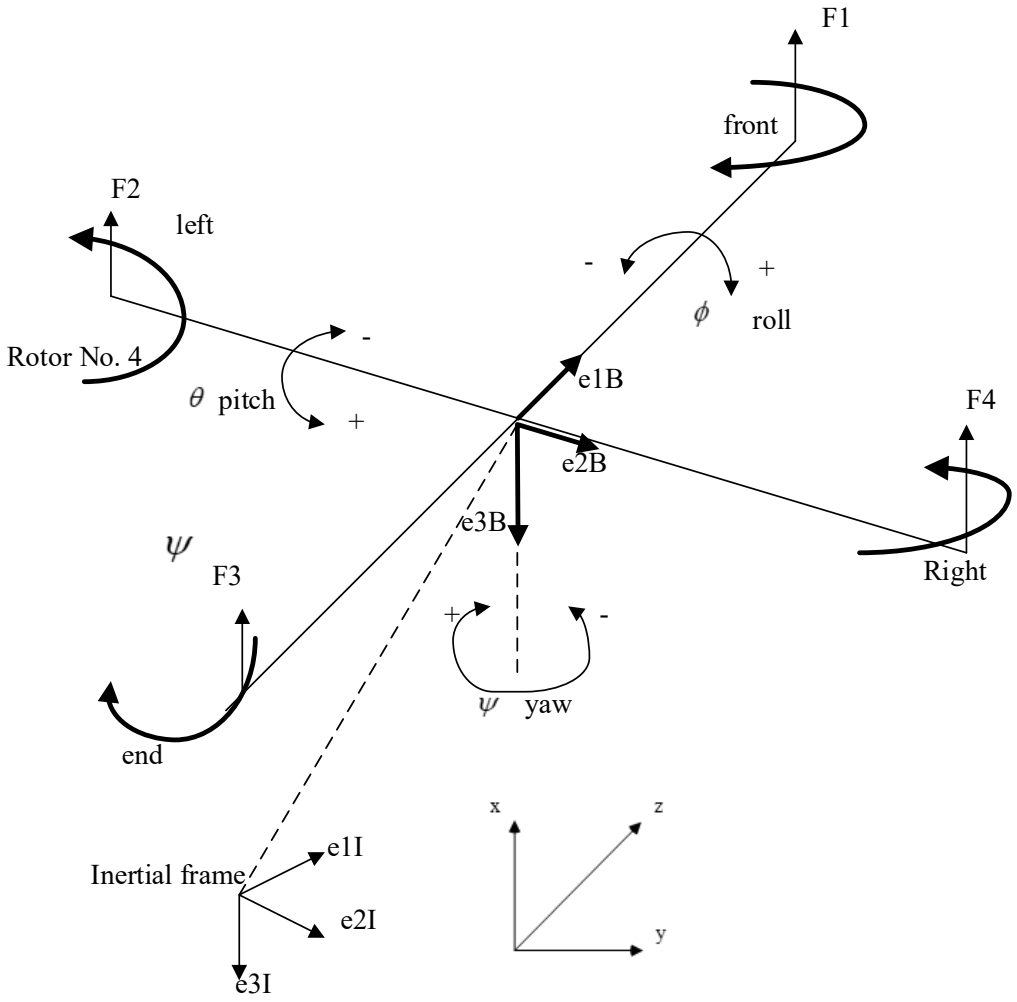

Figure 2.3: Quadcopter relative coordinate system

From Figure 2.3, the earth fixed inertial reference frame is $E_I(e_{1I}, e_{2I}, e_{3I})$ and the body fixed reference frame is $EB(e_{1B}, e_{2B}, e_{3B})$. $X = [x, y, z]^T$ and Euler angles $[\emptyset, \theta, \psi]^T$ which are earth fixed coordinates, describe the positon and rotational motion of the quadcopter UAV respectively. Where, φ is the yaw angle, θ, the pitch angle and $\emptyset$ the roll angle and x, y, z are the positions in space. $V = [u, v, w]^T$ and $\Omega = [p, q, r]^T$ which are body fixed coordinates, describes the linear velocity vector and the angular velocity vector of the air frame respectively, where u, v, w and p, q, r are the linear and angular velocities respectively.

Due to the existence of two different coordinate frames, there is need for transformation matrix between these two systems. The consequent rotations around $z \rightarrow y \rightarrow x$ axes is described using the rotation matrix (Kivrak, 2006):

$$R = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \sin\psi\cos\phi & \sin\theta\sin\phi\sin\psi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \sin\theta\cos\phi\cos\psi + \sin\psi\sin\phi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\theta\cos\phi \end{bmatrix} \quad (2.1)$$

For transformations from Earth-fixed coordinates to body fixed coordinates, matrix R is used.

The rate of change of position according to the craft velocity in the body frame is given by (Kivrak, 2006).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R^{-1} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (2.2)$$

The general description for the rate of change of rotational motion in relation to the quadcopter's rotation in the body frame is given as (Kivrak, 2006):

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = E^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.3)$$

$$\text{where } E = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

2.3 Dynamics of the Quadcopter UAV System

2.3.1 Quadcopter Equations of motion

There are two methods are used to describe the dynamics of the UAV, Newton-Euler and Langragian approach. Newton-Euler approach is used since it considers the forces of constraints and geometrical nature of the quadcopter system. Consider the general dynamics of a rigid body with consideration of external forces and moments in the body reference as in equations (2.4) and (2.5) (P. Wang et al., 2016).

Force Equation:

$$m\frac{dV^b}{dt} + \omega^b \times mV^b = \vec{F} \tag{2.4}$$

Moment Equation:

$$\dot{i}\frac{d\omega^b}{dt} + \omega^b + \dot{i}\omega^b = \overrightarrow{M} \tag{2.5}$$

where

$V^b = [u \quad v \quad w]^T$ is the relative linear velocity of the center of mass of the rigid body with respect to an inertial frame, $\omega^b = [p \quad q \quad r]^T$ the angular velocity in the body frame with respect to an inertial frame, $\dot{i}$ the inertia tensor of the rigid body and $\overrightarrow{M}$ is the external sum of all the moments in the body reference frame.

Expanding equation (2.4) and applying cross product of vectors yields (P. Wang et al., 2016)

$$\begin{bmatrix} \sum F_x \\ \sum F_y \\ \sum F_z \end{bmatrix} = m\left(\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}\right) = m\begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{2.6}$$

Assuming the aerodynamic forces, then the external forces acting on the quadcopter's body is the thrust *T* with the weight force only acting on the *z* axis.

Therefore, $F_x = 0, F_y = 0, F_z = -T$, and incorporating gravitational force, equation (2.6) can be converted to body fixed frame by multiplying it with the rotation matrix R from equation (2.1) (P. Wang et al., 2016).

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = -\begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} + \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \begin{bmatrix} -g\sin\theta \\ g\sin\phi\cos\theta \\ g\cos\phi\cos\theta \end{bmatrix} \tag{2.7}$$

where

m is mass, θ ,pitch angle and ϕ roll angle respectively.

Newton's second law states that [13],

$$\frac{dh^b}{dt_i} = \tau \tag{2.8}$$

where

$h =$ Angular momentum

$\tau =$ Applied torque.

Equation 2.8 implies that the angular momentum will change with application of torque. For changes in angular momentum vector direction, total derivative of h becomes (P. Wang et al., 2016):

$$\frac{dh}{dt_i} = \frac{dh}{dt_b} + \omega_{b/i} \times h = \tau \tag{2.9}$$

where i and b indicate inertial frame and body fixed reference frames respectively.

Equation (2.8) can be resolved in body coordinates where $h^b = I\omega^b_{b/i}$, I being the constant inertia matrix (moment of inertia of the quadcopter) given as [13]:

$$I = \begin{pmatrix} \int (y^2 + z^2)dm & -\int xydm & -\int xzdm \\ -\int xydm & \int (x^2 + z^2)dm & -\int yzdm \\ -\int xzdm & -\int yzdm & \int (x^2 + y^2)dm \end{pmatrix} = \begin{pmatrix} I_x & -I_{xy} & -I_{xx} \\ -I_{xy} & I_y & -I_{yz} \\ I_{xz} & -I_{yz} & I_z \end{pmatrix} \tag{2.10}$$

Due to quadcopter symmetry about all the three axes, $I_{xy} = I_{yz} = I_{xz} = 0$, the moment of inertia tensor is given as (P. Wang et al., 2016):

$$I = \begin{pmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{pmatrix} \tag{2.11}$$

where

I_x, I_y, I_z represent the symmetrical moments of inertia of the quadcopter. With ω as the change of rotational motion and I the moment of inertia, the applied torque can then be expressed as

$$\tau = I\dot{\omega} + \omega \times I\omega \tag{2.12a}$$

which can then be expanded as (P. Wang et al., 2016):

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}^{-1} \left\{ - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_x p \\ I_y q \\ I_z r \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \right\} \tag{2.12b}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = - \begin{bmatrix} qr(I_z - I_y)/I_x \\ pr(I_x - I_z)/I_z \\ pq(I_y - I_x)/I_z \end{bmatrix} + \begin{bmatrix} M_x/I_x \\ M_y/I_y \\ M_z/I_z \end{bmatrix} \tag{2.12c}$$

According to (Sidi, 1997), the rate of change of Euler angles and angular rates vector in the body reference frame are related as follows:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = R_\theta R_\phi R_\psi \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R_\theta R_\phi \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_\phi \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

Where $R_\theta R_\phi R_\psi$ are expressed as

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \quad R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \quad R_\psi = \begin{bmatrix} \cos\phi & \sin\psi & 0 \\ -\sin\psi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The moments equation described in equation (2.4) can thus be expressed as follows after all matrix multiplication (Sidi, 1997).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.14}$$

Euler angle rates can be determined by taking the inverse of the matrix in equation (2.12) (Selby, n.d.):

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan\theta\sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.15}$$

From equation (2.1), the translational equation of motion $[\dot{x} \quad \dot{y} \quad \dot{z}]^T = R^{-1}[u \quad v \quad w]^T$ can be expressed as (Selby, n.d.):

$$\ddot{x} = -\frac{F_z}{m}(\sin\psi * \sin\phi + \sin\theta * \cos\phi * \cos\psi) \tag{2.16}$$

$$\ddot{y} = -\frac{F_z}{m}(\sin\theta * \sin\psi * \cos\phi - \cos\phi * \sin\psi) \tag{2.17}$$

$$\ddot{z} = g - \frac{F_z}{m}(\cos\theta * \cos\phi) \tag{2.18}$$

Equations described in (2.7), (2.10), (2.13), (2.14), (2.15) and (2.16) are used in this research to model and simulate the quadcopter dynamics in MATLAB/Simulink environment.

2.3.2 Motor Dynamics

A quadcopter is controlled by varying the speed of the four rotors independently. These motors have their own dynamics. The equations of motion here are similar to the equations of motion of DC motor(Bolandi et al., 2013; Jirinec, 2011) since the quadcopter rotors are DC motors. The conventional permanent magnet DC motor is described using the basic physical principles which can also be applied to the brushless DC motor. Equations (2.19) and (2.20) are the Kirchhoff's Law and Newton's Second

Law respectively. The latter describes the dynamic equation for the motor coupled to a load (Selby, n.d.)

$$PWM_{app}(t) = Ri(t) + L\frac{di(t)}{dt} + K_e\omega(t) \tag{2.19}$$

$$J\frac{d\omega}{dt} = K_\tau i(t) - K_e\omega(t) \tag{2.20}$$

where i = motor current, R = Resistance of the coils and windings, L = Inductance of coils, R = Resistance of the coils and windings, J = Moment of inertia of rotor (Kg/m^2), K_e = EMF constant $(volts\ per\ rad)$, K_τ = Armature constant (N - m/amp), K_f = Motor friction constant (N - m - s/rad), ω = Motor speed (rad/s)

A transfer function is determined for the purposes of modelling the system dynamics of the motor with propeller speed as the output and motor voltage as the input. In frequency domain (Selby, n.d.),

$$PWM_{app}(s) = (sL + R)I(s) + K_e\omega(s) \tag{2.21a}$$

$$K_\tau I(s) = (Js + K_f)\omega(s) \tag{2.21b}$$

$$G(s) = \frac{\omega(s)}{PWM_{app}(s)} = \frac{K_\tau}{(sL + R)(Js + K_f) + K_eK_f} \tag{2.21c}$$

$$G(s) = \frac{K_\tau}{JLs^2 + (K_fL + RJ)s + RK_f + K_eK_f} \tag{2.21d}$$

Since inductance can be assumed to be low i.e. $L \approx 0$ and the motor is small thus the motor friction constant K_f is very small, it can be assumed that:

$RJ >> K_fL$
$K_eK_\tau >> RK_f$

then equation (2.21d) simplifies to (Selby, n.d.):

$$G(s) = \frac{K_\tau}{JLs^2 + RJs + K_eK_f} \tag{2.22}$$

Since the motor is small, inductance can be assumed to be low i.e. $L \approx 0$ reducing the transfer function to a first order system:

$$G(s) = \frac{K_\tau}{RJs + K_e K_f} \tag{2.23}$$

The total upward thrust equals the sum of the rotor angular speeds squared as in equation (2.24) (L. Wang & Chen, 2016)

$$T = b(\omega^2_1 + \omega^2_2 + \omega^2_3 + \omega^2_4) \tag{2.24}$$

where b is the thrust coefficient of the rotor. Pairwise differences in rotor angular speed ω_i with $i = 1; 2; 3; 4$, referring to the rotor number, results in the torques $\tau_\emptyset; \tau_\theta \; and \; \tau_\Psi$ in the body frame and causes the aerial vehicle to rotate about the $x, y \; or \; z$ −axis (L. Wang & Chen, 2016).

$$\tau_m = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lb(\omega_4{}^2 - \omega_2{}^2) \\ lb(\omega_1{}^2 - \omega_3{}^2) \\ d(\omega_1{}^2 - \omega_2{}^2 + \omega_3{}^2 - \omega_4{}^2) \end{bmatrix} \tag{2.25}$$

τ_m = body torque from motors, $\tau_\phi, \tau_\theta, \tau_\psi$ = torques along the body axes, b = thrust coefficient of the rotor, d = drag torque proportionality constant, l = arm length from the centre of mass

2.3.3 Control input

The Electronic Speed Controller (ESC) generates corresponding PWM signal for motor speed control (Selby, n.d.) by converting the control constants $[U_{roll}, U_{pitch}, U_{yaw}]$ to motor control signals$[u_1, u_2, u_3, u_4]$. A control vector U defines four inputs as follows (Selby, n.d.):

$$U = [u_1 \quad u_2 \quad u_3 \quad u_4] \tag{2.26}$$

where u_1 = Total thrust of all motors, u_2 = Control input to the roll, u_3 = Control input to the pitch, u_4 = Control input to the pitch.

The above inputs are then changed into individual motor speeds. The achieved speeds are relayed to the ESCs and finally to the motors.

Using equations (2.24) and (2.25), the equations for the above inputs to the quadrotor UAV system are expressed as (Selby, n.d.):

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ lb & 0 & -lb & 0 \\ d & -d & d & -d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{2.27}$$

Inverting equation (2.27) gives the desired motor speeds (Selby, n.d.):

$$\omega_1^2 = \frac{u_1}{4b} + \frac{u_3}{2lb} + \frac{u_4}{4d}$$

$$\omega_2^2 = \frac{u_1}{4b} - \frac{u_4}{4d} - \frac{u_2}{2lb}$$

$$\omega_3^2 = \frac{u_1}{4b} - \frac{u_3}{2lb} + \frac{u_4}{4d}$$

$$\omega_4^2 = \frac{u_1}{4b} + \frac{u_3}{2lb} - \frac{u_4}{4d}$$

2.4 Quadcopter System model under actuator faults

In the presence of actuator fault, the control signal can be written as (Sadeghzadeh et al., 2014)

$$u_{fi}(t) = l_{fi} u_i(t) \tag{2.28}$$

where l_{fi} is the effectiveness of a particular actuator with l_{fi}=1 as a faultless actuator and $l_{fi} = 0$ as the complete actuator failure.

For the quadcopter UAV, the effectiveness of four actuators can be expressed as (Liu et al., 2016):

$$u_f(t) = L_f u(t) \tag{2.28}$$

where

$L_f = diag\{l_{f1},....,l_{f4}\}$ is the effectiveness factors written in a diagonal matrix, $u_f(t) = [u_{f1}(t),...,u_{f4}(t)]^T$ is the faulty control input vector and

$u(t) = [u_1(t),...,u_4(t)]^T$ the control actions on fault free quadcopter UAV.

Equations (2.7), (2.10), (2.13), (2.14), (2.15), (2.16) and (2.27) can be expressed in linearized state space representation (Liu et al., 2016):

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y &= Cx(t) \end{aligned} \tag{2.29}$$

where $\dot{x}(t)$ represents the states, A the transition matrix, B the input matrix, $u(t)$ the input variable and y the system output.

From (Sadeghzadeh et al., 2014), the system with actuator fault is expressed as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_f u(t) \\ y &= Cx(t) \end{aligned} \tag{2.30}$$

where $B_f = BL_f$ is the control input matrix after fault, the state vector and control input are taken as $x(t) = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \theta & \dot{\theta} & \phi & \dot{\phi} & \psi & \dot{\psi} \end{bmatrix}^T$ and $u(t) = \begin{bmatrix} u_z & u_\theta & u_\phi & u_\psi \end{bmatrix}^T$ respectively.

2.5 General Control Strategy

In this research, the translational and the lateral movements of the quadcopter are measured by the onboard sensors. The values are then compared with the desired values. The resulting error is fed into the PID controller.

2.5.1 Quadcopter Proportional, Integration and Derivative control

The composition of the PID controller in time domain is given by (Sadeghzadeh et al., 2014)

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \qquad (2.31)$$

where

K_p, K_i, and K_d are the PID gains. $u(t)$ is the control output, $e(t)$ is the error between the actual state and the desired state (Kobayashi et al., 1995a).

The PID controller functions in three parts as shown in Figure 2.3.

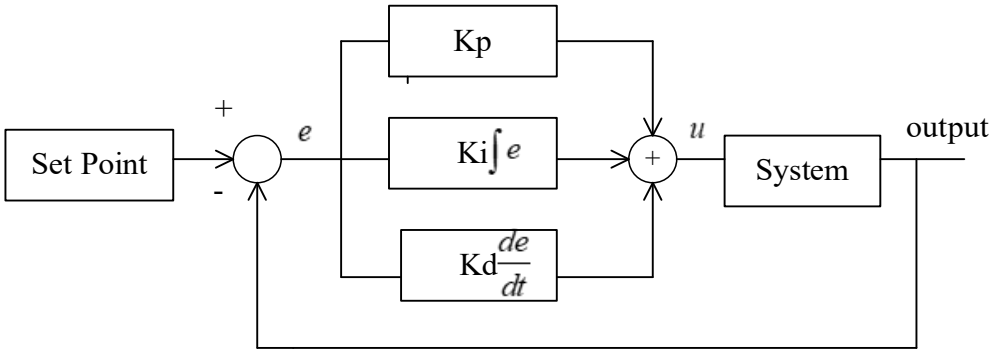

Figure 2.4: Conventional PID controller

The PID controller acts on the error creating a rapid response to control output. It also acts in proportion to the error integrator eliminating the steady-state error (Kobayashi et al., 1995a).

Some of the algorithms used in various research to tune the PID gains are the Ziegler-Nichols, Cohen-Coon Method, Tyreus-Luyben and Damped Oscillation Method among others (Kobayashi et al., 1995a). In this work, the trial and error is used to adjust the parameters based on the experts experience and gain adjustments can be done based on observations.

The PID corrects the errors as follows:

Proportional, K_p

The farther away the desired position, the harder the controller tries to get it back to the desired position. If the quadcopter is supposed to be horizontal but one side is slanted downwards, the proportional controller pushes that side back up as quickly as

it can. The value of P dictates how hard it tries to get back to the desired position (Amoozgar et al., 2012).

Signs of P being too high:

- The quadcopter tries too hard to get back to the desired point and will actually overshoot and then try to correct the error again. This results in more overshoot and even undershoots. The response goes past the desired point and the error has to be fixed in turn.
- The model oscillates for left to right maneuvers

Integral, K_i

Provides controls for sustained deviation from the desired orientation. It maintains a particular attitude. For instance, if flying forward but the nose is rising all the time, it is because the sustained change, which is letting the nose drift up, is not being taken care of by the Integral part of the PID loop. The quadcopter should maintain a forward attitude for a correct integral value but if it tilts back up, the value of I should be raised (Amoozgar et al., 2012).

Derivative

It dictates the speed at which K_p and K_i work out. If proportional gain is how hard the system tries to get back to where it needs to be and I corrects for sustained differences from the desired orientation, then D is how quickly the PID loop is trying to get system back to the desired position.

PID controllers are frequently and widely used in various number of industrial applications since they are simple and easy to use (Amoozgar et al., 2012). However, one of their main shortcomings is the lack of a systematic way of choosing the optimum control parameters that ensure acceptable performance. Structural deviations affect the performance of these controllers. It is therefore important to have an algorithm hybridized with the PID controller that automatically adjusts the above mentioned parameters to compensate for unforeseen structural and environmental changes. In this research, a Fuzzy Logic Controller (FLC) is used since it is simple to

design and implement and can also provide acceptable control performance under structural uncertainties in nonlinear systems.

2.5.2 Quadcopter Fuzzy PID control

Fuzzy Logic Control (FLC), developed by Lofti Zadeh in 1965 (Tamilselvan & Aarthy, 2017), is an intelligent control method that imitates the logical thinking of human. The FLC can compensate for some of the PID shortcomings.

Rather than fixed reasoning, FLC being multi-valued logic deals with approximate reasoning. The variables varies in truth value that ranges between 0 and 1 as opposed to conventional binary sets. The FLC comprises of Fuzzification, rule base, inference mechanism and defuzzification as shown in Figure 2.4 (Sivanandam et al., 2007).

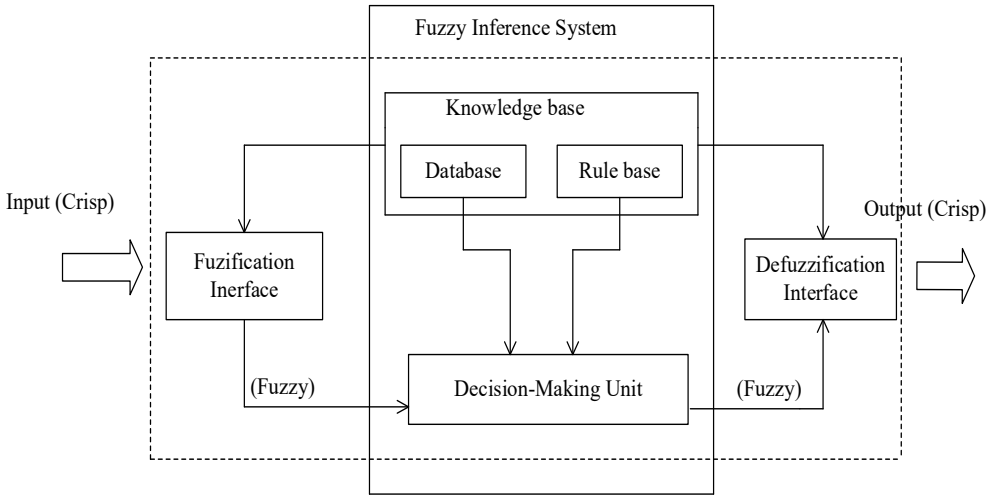

Figure 2.5: Block diagram of Fuzzy logic model

The input of the fuzzy logic model is crisp value i.e. a precise variable that assigns a value of either 0 or 1 to each element of the universe. This is then converted into fuzzy linguistic variables through the process of fuzzification using Membership Functions (MFs) which can be Triangular, Trapezoidal, Bell, and Gaussian. Fuzzification involves conversion of crisp values of the control inputs into values which are imprecise rather than exact. The fuzzified output is fed into the Fuzzy Inference System (FIS) which initiates the decision making process by using fuzzy set theory or fuzzy reasoning to map nonlinear input data set into an output scalar data using rule base. Rule base have control rules obtained from an expert knowledge expressed as IF-THEN rules and the associated database (Lee, 1990)

The commonly used fuzzy inference system are the Mamdani and Takagi Sugeno methods. In essence, the operational procedure of Mamdani inference engine is similar to that of Takagi Sugeno method except that for the latter, all membership functions are designed using singleton spikes or the output of each fuzzy rule is constant (Ofosu et al., 2016). They are similar in that the crisp values are mapped onto the MFs according to rule evaluation. Mamdani FIS is used in this work because it is easy to implement since it follows linguistic fuzzy modelling as opposed to Takagi Sugeno which follows precise numerical modelling.

AND, OR and NOT are used as fuzzy logic operators in rule evaluation of the antecedent and consequence MFs of fuzzy IF-THEN rules. Aggregation involves summation of all the outputs of fuzzy rules by taking the MFs previously scaled and combining them into a single fuzzy set. The outcome of the FIS is transformed to crisp values that can be used as a control signal through the defuzzification process.

In this research, triangular MFs are used for the FLC because of its computational efficiency and simplicity (Lee, 1990). It is also commonly used. The error derivative $\left|\dot{e}(t)\right|$ and error $|e(t)|$ are taken as two inputs. The degree of membership of a crisp input is expressed as:(Ahn & Truong, 2009).

$$f_{ij}(zj) = \begin{cases} 1 + {(z_j - c_{ij})}/{b^-_{ij}} \text{ if } -b_{ij} \le (z_j - c_{ij}) \le 0 \\ 1 - {(z_j - c_{ij})}/{b^+_{ij}} \text{ if } 0 \le (z_j - c_{ij}) \le b^+_{ij} \\ 0 \quad \text{otherwise, j} = 1,2,..\text{N} \end{cases} \tag{2.32}$$

z_j is the j^{th} input, b^-_{ij} lower half-width, c_{ij} the centroid and b^+_{ij} the upper half-width of the i^{th} triangle membership function of the j^{th} input. N is the number of MF triangles.

For the output, the centroids and half widths are denoted by γ_{ij} and β^-_{ij}, β^+_{ij} respectively.

The quadcopter altitude and attitude deviation e from the reference are sampled and calculated. The fuzzy matrix table which is based on the fuzzy rules and reasoning determines ΔKP, ΔKI and ΔKD which are the fuzzy output variables. The initial PID values are then adjusted using ΔKP, ΔKI and ΔKD depending on the altitude and rotational motion references. The Fuzzy-PID structure is as shown in Figure 2.6 (Bousbaine et al., 2016).

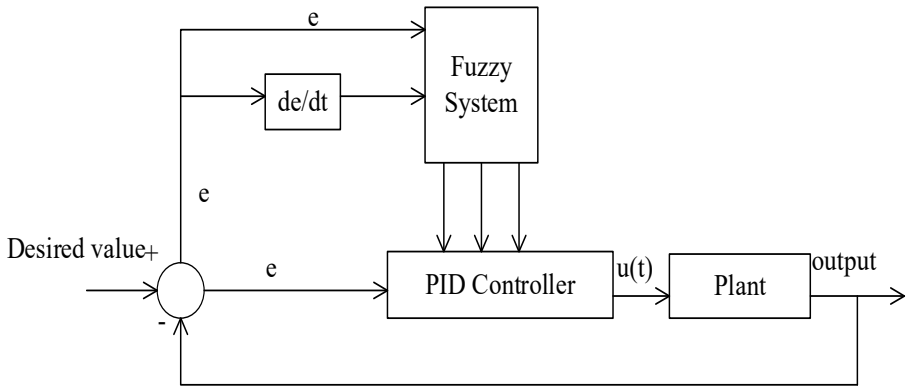

Figure 2.6: Basic Structure of Fuzzy PID controller with plant

The proportional, derivative and integral gains K_p, K_d and K_i respectively, are adjusted in the controlled process of the system according to the following different scenarios based on expert experience (L.-X. Wang, 2003):

When the error is large, K_p and K_i should be increased to eliminate the steady state error. K_d should also be chosen properly to avoid system response oscillation near the set point.

When the error derivative is small, K_d should be relatively bigger and when the error derivative change is large, K_d should be smaller; usually. This necessitates that K_d should be of middle size i.e., neither big nor small.

When the error $|e(t)|$ and error derivative $\left|\dot{e}(t)\right|$ is of middle size, K_p should be small to reduce the overshoot of system response, and assure certain response speed.

The Fuzzy PID controllers proposed in (Amoozgar et al., 2012)and (Bousbaine et al., 2016)are based on experimental design depending on operating conditions of the quadcopter. Therefore, the design of fuzzy rules depend on the experience of experts since there is no systematic method to design and examine the number of rules and membership functions (Ahn et al., 2007; Truong et al., 2007).

2.6 Extended Kalman filter

The typical Fuzzy PID controllers cannot adapt for structural and environmental variations since the rules and initial MFs are based on expert experience (Ahn et al., 2008). Therefore, there is need for a control technique that has learning capabilities to combine with the Fuzzy PID to overcome such shortcomings. The control technique selected should be able to adjust the MF parameters of the Fuzzy controller which in turn adjusts the PID gains automatically to minimize the control error.

Kalman Filter is an efficient mathematical technique for controller training purpose. It is also able to estimate sensor measurements from a noisy environment by making an approximation of the system states, called the priori estimate (Ahn & Truong, 2009). This estimate is used to predict the measurement that is about to arrive. The current estimate is recursively conditioned on all of the past measurements, and generally converges in a few iterations.

However, Kalman filter operates efficiently with linear systems while most physical systems contain nonlinearities and the noise in the measurements generally do not have a Gaussian distribution. This necessitates linearization of processes in real life before estimation using Kalman Filter.

This is where Extended Kalman Filter (EKF) comes since it is able to estimate nonlinear functions (Anderson & Moore, 1979). It is a derivative-based method with fast convergence (Simon, 2002) suitable for optimization of the MF parameters. Triangular MFs are used in this research. An error function is given by (Anderson & Moore, 1979):

$$E = \frac{1}{2N}\sum_{q=1}^{N}(\tilde{y}_q - y_q) \tag{2.33}$$

y_q is the target value of the fuzzy system, N is the number of training samples and $\tilde{y}_q$ is the output of the fuzzy system. E can be optimized by using the partial derivatives of E with respect to centroids and halfwidths of the input and output fuzzy MFs

2.7 Attitude Estimation using Sensor fusion

Attitude here refers to roll, pitch and yaw motions. Quadcopters have a set of sensors known as Inertial Measurement Unit (IMU) that provide the information needed by the attitude control systems. The IMU of a quadcopter contains an accelerometer, a gyroscope and a magnetometer.

A gyroscope based IMU is not affected by motor vibrations. However, drifting of the angles is a problem. With accumulation of errors with time due to gyroscope bias, it is practically impossible to rely on gyroscope data alone. In static or slow movement, the accelerometer measures roll and pitch by leveling to correct the gyro-unbounded error. This is due to the trustworthiness of the gravitational measurement. While the accelerometer gives absolute measurement of the quadcopter attitude, the motors on the quadcopter produce a lot of vibrations introducing significant noise into the accelerometer reading (De Marina et al., 2012). Therefore, a proper fusion of IMU data is needed to overcome the shortcomings of each sensor. Fusion involves combining different sensor measurements using a suitable filter to achieve better estimates, redundancy and drift compensation(Kubelka & Reinstein, 2012).

In this work, gyroscope and accelerometer measurements are fused using Kalman Filter and Complimentary filter to estimate the orientation.

2.7.1 Accelerometer

Accelerometer measures total acceleration relative to free fall, also called specific force $\bar{f}^b$ (Jirinec, 2011). When an accelerometer is part of a system like UAVs and robots, it not only measures acceleration due to gravity but also translational and rotational accelerations. Therefore, an ideal accelerometer aligned with the body measures specific force as shown in equation (2.34). A detailed derivation is given in (Kaba et al., 2017).

$$\overline{f}^{b} = \begin{bmatrix} f_{x,accel} \\ f_{y,accel} \\ f_{z,accel} \end{bmatrix} = -g \begin{bmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{bmatrix} \tag{2.34}$$

where, $\bar{g}^{b}$is gravity in body coordinates, ϕ and θ represent roll and pitch in radian respectively.

2.7.2 Gyroscope

Gyroscope sensors measure angular velocity in x, y, z directions although its measurements include biases. It is modeled as follows (Kaba et al., 2017):

$$\overline{\Omega}^{b}_{gyro} = \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} = \overline{\Omega}^{b} + \begin{bmatrix} n_{\Omega_x} + b_{\Omega_x} \\ n_{\Omega_y} + b_{\Omega_y} \\ n_{\Omega_z} + b_{\Omega_z} \end{bmatrix} \tag{2.35}$$

b_{Ω} is the gyroscope bias and n_{Ω} represent the associated noise. Gyroscope measurements and Euler angle rate are related as shown (Kaba et al., 2017):

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} \tag{2.36}$$

Where θ and $\emptyset$ represent pitch and roll respectively and Ω is the propeller angular velocity.

2.7.3 Attitude Estimation using Complimentary Filter

When measuring the body angle with the accelerometer, it is affected by translation and vibrations of the motors, but the errors are not accumulated. When measuring with the gyro sensor, the errors are accumulated, but vibrations do not affect its operation. These two sensors measure the same physical quantities, and the properties are complementary, so the weaknesses of each sensor can be supplemented through convergence.

The complementary filter, Figure 2.7, consists of high pass filter and low pass filter. Low pass filter is needed for correction of accelerometer data because as the small

forces create disturbance in measurement, long-term measurement is reliable. High pass filter is used for gyroscopic data correction since its values start to drift in the long term.(Cao et al., 2009; Higgins, 1975).

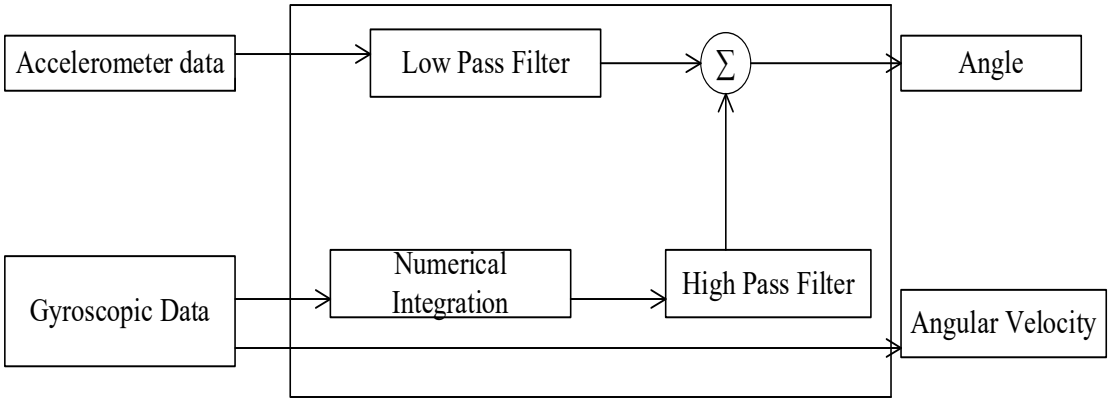

Figure 2.7: Block diagram of the composition of the Complimentary filter

The complimentary filter is a unity filter (Vasconcelos et al., 2009). The weighted portions of the gyroscope angle is added to weighted portions of the accelerometer angle to give the complimentary filter angle (Vasconcelos et al., 2009).

$$\theta_k = G_g * (\theta_{k-1} + G_d * dt) + (0.020) * (A_d) \qquad (2.37)$$

where: θ_k is the complimentary filter angle, G_g is the gyroscope gain, θ_{k-1} is the previous complimentary filter angle, G_d is the gyroscope data, A_d is the accelerometer data.

2.8 Review of Quadcopter Control Algorithms

In this section, applicable algorithms for quadcopter control under rotor fault are analyzed highlighting their advantages and disadvantages.

2.8.1 PID Technique

PIDs are control loop feedback mechanisms that directly adjust control values with a closed-form formula based on derivative, integral, and proportional gains (Numan, 2017).

PID formulation is employed in (Numan, 2017), beginning with an open loop system to show the advantages of PID control and the closed loop response is as shown in Figure 2.8. The scheme then describes the implementation of a PID controller without

rotor failure with the gain values set at $K_p = 10, K_i = 0$ and $K_d = 11$. The resulting overshoot of 0% and settling time of 3.6043 seconds was observed.

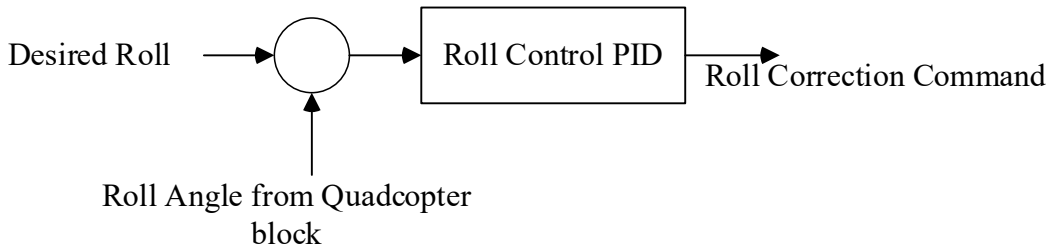

Figure 2.8: Roll Control PID Block

Similar arrangement applies to PID Pitch control

PID control was implemented in design of Quadrotor Controller for stabilization after failure of one of the rotors in (Cho et al., 2015). Motor 2 is switched off after 37 seconds and the simulation results showed that the quadcopter lost stability and altitude, crashing right away.

In conclusion, the classical PID controller parameter gains are easy to tune, simple to design and is robust. However, the quadcopter being under actuated has a nonlinear mathematical model. PID linear controller has been found to struggle with aggressive maneuvers (Balas, 2007) especially when one of the rotors is faulty.

Gain Scheduled Proportional Integral Derivative controller (GS-PID)

GS-PID is applied in (Sadeghzadeh et al., 2014) to control a quadcopter for an 18% loss in power for all the motors. The fault was injected at 20s and acceptable tracking deviation was obtained as shown in Figure 2.9.

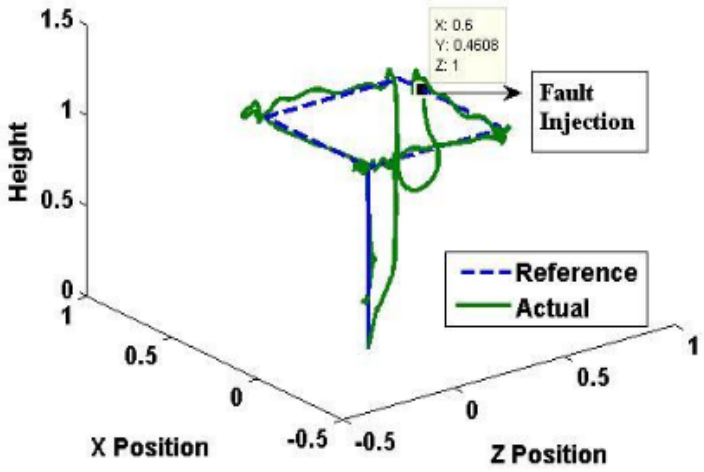

Figure 2.9: GS-PID for a faulty condition

GS-PID is simple to design and implement in MATLAB/ Simulink, however, tuning GS-PID controller gains is time consuming and the switching time (more than 1 second) between the gains could cause the quadcopter to crash for a faulty situation.

2.8.2 Model reference adaptive controller (MRAC)

In MRAC, the dynamic response is forced to asymptotically approach a reference system even in the presence of faults. In (Zhang, 2011), the flight was tested for both hovering control and square trajectory tracking controls with fault injection. The flight results from the experiment are shown in Figure 2.9 and 2.10. In Figure 2.10, the Quadcopter is fault free and the MRAC tracks the trajectory close to the desired trajectory. Two motors (the left and back) are subjected to an 18% power loss at 20 seconds during flight. Figure 2.10, shows the Quadcopter tracking the desired trajectory and performing a safe landing.

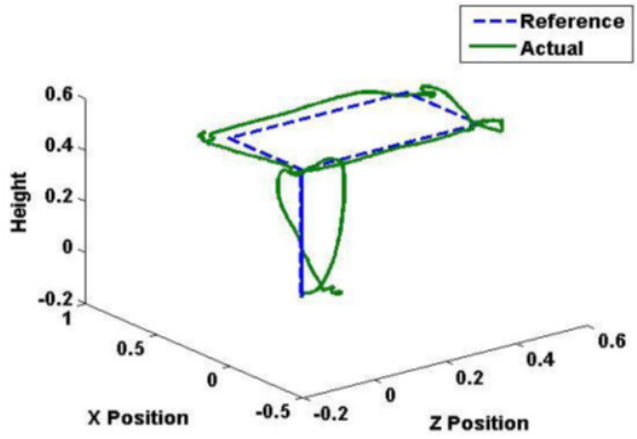

Figure 2.10: MRAC in fault-free condition

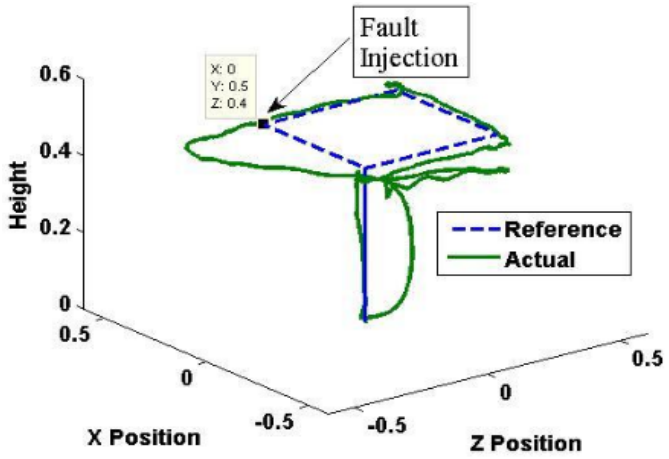

Figure 2.11: MRAC in faulty condition

The MRAC used in (Sadeghzadeh et al., 2014) proved to be more robust and reliable to structural changes. However, involving mathematical model derivations are needed to design and implement the controller.

2.8.3 Linear Quadratic Regulator

This method derives the feedback gain for a system. Applying the Linear Quadratic (LQ) control requires the quadcopter equations of motion to be linearized.

In (Castillo et al., 2005), LQR controller was implemented and the controller performed satisfactorily during simulation. However, the controller, due to its linearity, could not stabilize the system when perturbations were introduced. Moreover, the controller could not stabilize the physical model under these conditions. Simulation results in literature have shown that LQR controllers applied in fault free

quadcopters have performed better than classical PID controllers in terms of transient response (Lan et al., 2012) In (Bouabdallah et al., 2004), LQR algorithm and PID performance are compared with the PID being applied on a simplified quadcopter dynamics and the LQR on the complete model. The performance of LQR was found to be superior to that of the PID controller

2.8.4 Feedback Linearization (FBL)

In FBL a nonlinear system is transformed into a linear system by designing a suitable control law such that the nonlinear term is cancelled to result in a controllable linear system. Some of the disadvantages of using FBL is that it needs a Fault Diagnosis and Identification system (FDIs) and the loss of precision due to linearization (Zulu & John, 2014).

In (Freddi et al., 2011b), a controller was developed having inner, faster controller which regulates the attitude angles and the vehicle attitude. This is because the roll, pitch and altitude have the highest priority during flight and therefore the inner controller should work much faster than an outer controller. The slower outer controller supplies a proper input to the remaining fault free rotors. The outer control law ensures the quadcopter reaches a desired position in space by supplying proper input to the fault free rotors. When one of the rotor fails in one of the axis, the inner controller ensures the velocity of the fault free rotor on the same axis as the faulty rotor is modulated until the value of the angle controlled by that axis is zero. The vehicle then spins around the vertical axis while at the same time varying simultaneously the rotational velocity of the two rotors of the fault-free axis. This makes it possible to set a desired altitude for the vehicle.

FBL for fault tolerant control is described in (Freddi et al., 2011a) where the quadcopter continues to fly with only three functional actuators. A simulation is done with only the inner controller activated. Figure 2.11 shows a smooth profile of angles $\emptyset$ and θ go to zero in less than $15s$ and the altitude z regulated to the desired value of $10m$ even with the sustained rotor fault.

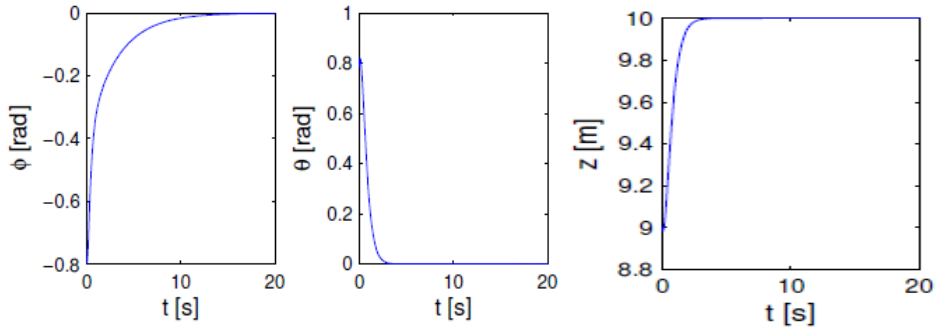

Figure 2.12: Response with only inner controller applied

A second simulation was performed with the desired attitude and altitude set to 0 and 10m respectively and both the control laws incorporated. A landing procedure is initiated if the quadcopter does not reach the desired lateral and longitudinal position as shown in Figure 2.13. FBL was able to stabilize the attitude angles $\emptyset$ and θ, but with oscillations due to the presence of the outer controller.

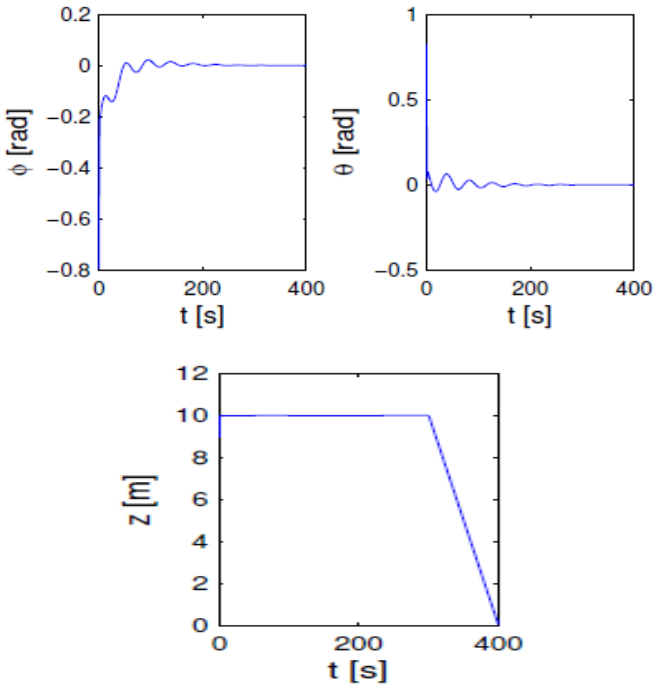

Figure 2.13: Angle stabilization.

From the simulation results, it was observed that the outer controller forces attitude angles to differ from zero as long as the target position is not reached whereas the inner

controller tries to constrain those values to zero. This results in more time needed for the target set points to be attained thus the 400s simulation time

2.8.5 Sliding Mode Controller (SMC)

A state feedback control law is designed in such a way that the current state is mapped to an input u so that the system is stable around the origin i.e. a system started away from origin will always return to it. SMC forces the system trajectories into a constrained subspace then holds them there so that they slide along with it.

(Sharifi et al., 2010) described the fault tolerance property of SMC and uses it in a Fault Tolerant Controller. The objective of their work was to land the quadcopter for conditions ($\emptyset = 0 \ \ and \ \theta = 0$) when a rotor fault occurs.

A faulty condition (partially loses its thrust) was introduced after 7 seconds to rotor 1 and the performance of the quadcopter was observed without and with the SMC. The controller was able to achieve both the desired attitude angles and the height as shown in Figure 2.14 and Figure 2.15.

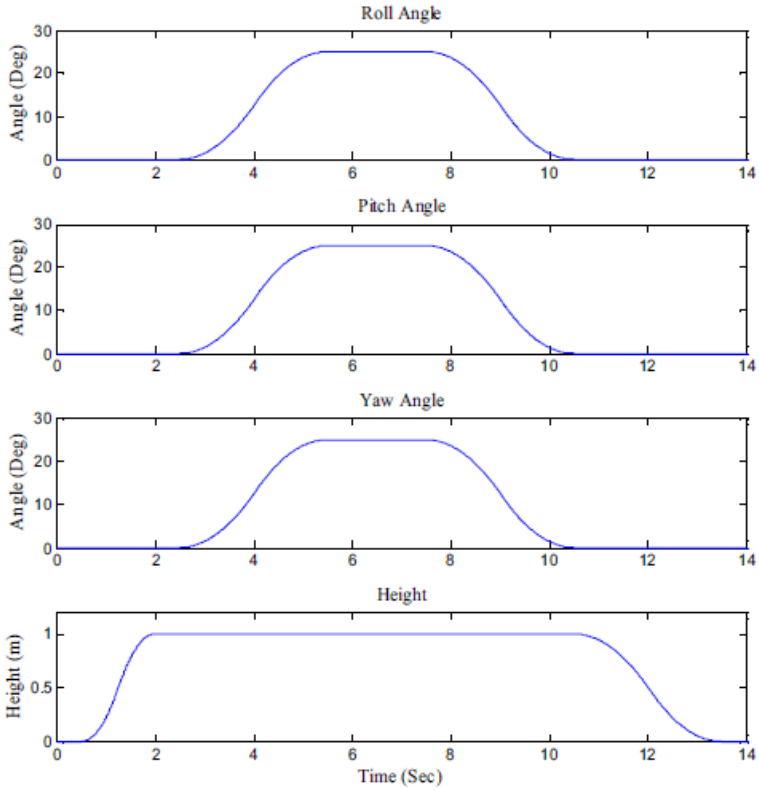

Figure 2.14: Desired quadcopter attitude and altitude

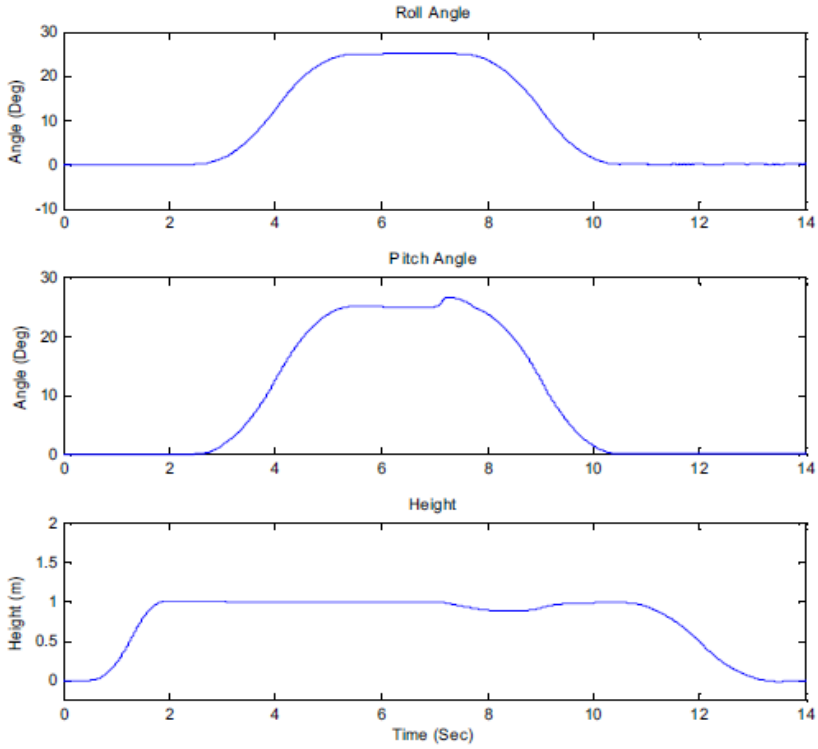

Figure 2.15: Quadcopter performance under faulty condition

However the proposed SMC was not able to perform without the assistance of a state estimator i.e. a FDD unit, making the whole unit expensive. The performance of the SMC without the FDD is as shown in Figure 2.16. It can be observed that the quadcopter could not be stabilized.

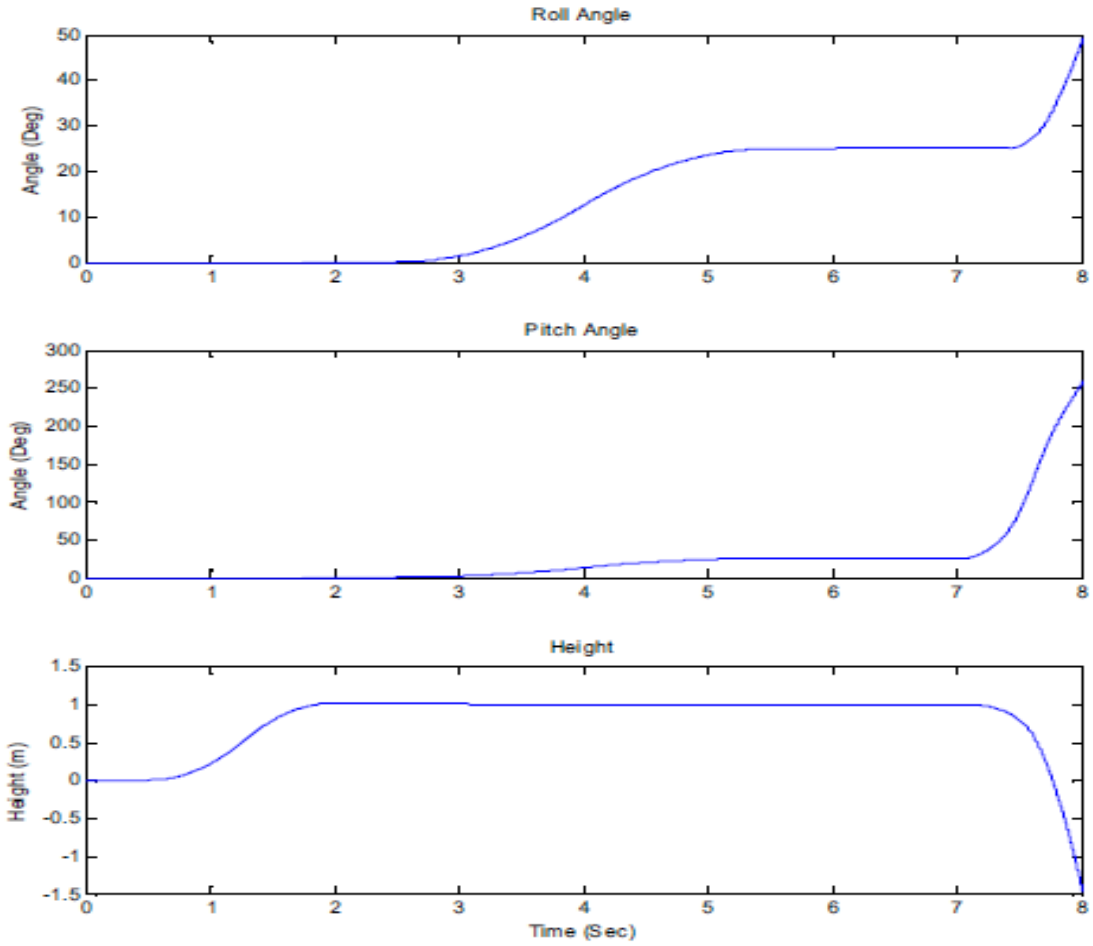

Figure 2.16: Quadcopter performance without the FDD unit

Furthermore, experimental results show that sliding mode controller is more robust with better tracking performance in the presence of faults but the SMC has not been tested for severe actuator failures and the simulation results show that there is chattering due to the switching from the normal operation mode and the faulty mode (Li et al., 2013).

2.8.6 Model predictive control (MPC)

In MPC, the control signal is recalculated at each sampling time with any deviations reflected in signal calculation (Maciejowski & Jones, 2003). The drawbacks of MPC include:

(1)An FDD is required as an observer to allow the MPC to consider faults (Izadi et al., 2011).

(2) An explicit model of the quadcopter is needed to calculate a stabilizing control signal.

The problem becomes more critical where the system dynamics is described by a nonlinear model (L. Wang & Beumer, 2016).

An MPC strategy was proposed in (L. Wang & Beumer, 2016), sacrificing the control on yaw. According to the simulations, the MPC method is able to get the quadcopter UAV in hover position. The presented simulations showed that the roll and pitch angles were stabilized at the desired angles. However, the controller cannot be implemented on hardware for experimental results since the angular velocities were very high and may cause problems therefore there is need to physically validate these simulation results.

2.8.7 Backstepping control

A higher order nonlinear system is broken down into a number of lower order subsystems that are cascaded with each other to form the general closed loop system. The stabilization of each subsystem is recursive, i.e. stabilization by a successive lower order subsystem generating a control input. The recursive algorithm is used to achieve feedback control. This feedback control law is converted, through parameter estimation, into dynamic control law to accommodate the dynamic perturbations in parameters (Khebbache, 2012).

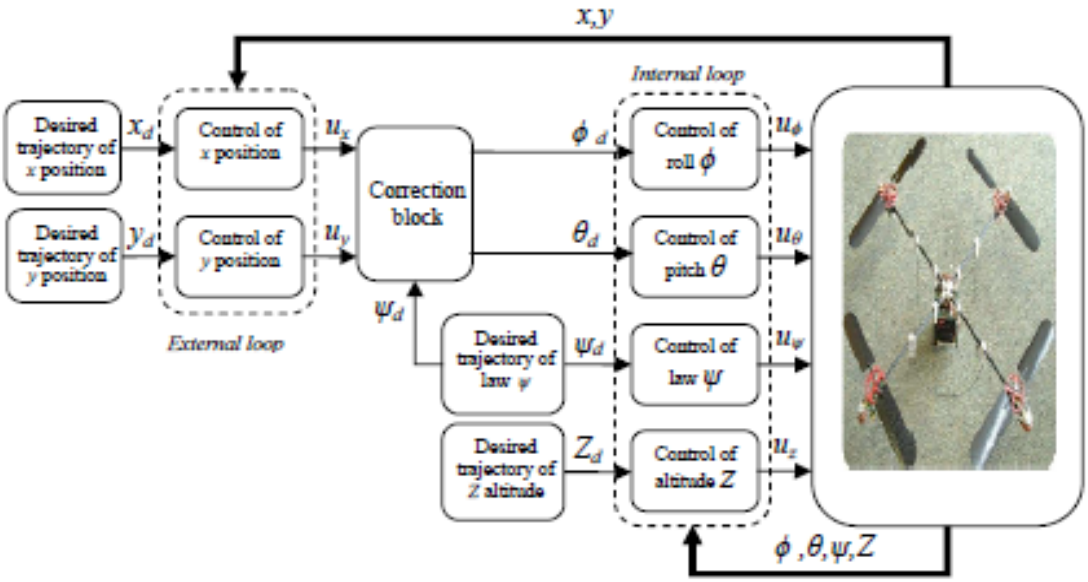

Figure 2.17: Backstepping Control Strategy

The adopted control strategy in (Khebbache, 2012) describes a control strategy based on an internal loop and external loop. The internal loop controls the roll, pitch, yaw and altitude. The external loop controls positions x and$y$. The external control loop generates a desired roll (ϕ_d) and pitch (θ_d) as follows:

$$\phi_d = \arcsin(U_x Sin(\varphi_d) - U_y \cos(\varphi\_d)) \tag{2.38}$$

$$\theta_d = \arcsin(\frac{U_x \cos(\varphi_d) + U_y Sin(\varphi_d)}{\cos(\phi_d)}) \tag{2.39}$$

The control laws constraining the system to the target trajectory in the event of rotor failure is synthesized using a recursive algorithm.

The simulation results in (Khebbache, 2012) showed acceptable performance towards tracking even after the occurrence of actuator faults explaining the efficiency of the control strategy developed. However, Backstepping approach is affected by chattering effects on the inputs. There is therefore need to include other control strategies for elimination the chattering effects on inputs control u_1, u_2, u_3, u_4, while maintaining the stability and the performances of this system, with implementation on a real system.

Mueller and D'Andrea (Mueller & D'Andrea, 2014) presented solutions for a quadcopter experiencing up to three complete rotor failures with the quadcopter UAV maintaining a position in space. This was achieved by having the UAV rotate freely about an axis, fixed with respect to the body. The UAV was also assumed to already have a Fault Detection, Diagnosis and Isolation (FDI) module. The hover stabilized after 12s and the UAV lands at 21.5s.

100% propeller loss for a quadcopter UAV is described in (Freddi et al., 2011a). To achieve a horizontal spin using the remaining propellers. This paper successfully developed a control law using feedback linearization (FBL) approach for stabilization of the quadcopter UAV in event of rotor failure and enable it make a safe landing. This scheme assumes that the vehicle is already equipped with an observer to switch between the fault free controller and the faulty one.

Partial failure of a quadcopter actuator is investigated in (Ranjbaran & Khorasani, 2010). Here, a parameter estimation algorithm which acts as a FDI system is proposed to provide an estimate of the severity of the faulty rotor. The behavior of the quadcopter system is investigated in case of a more severe loss of effectiveness fault. 25% and 50% LOE were considered faults in the first actuator for evaluation.

Multiple partial loss of effectiveness was considered in (Ranjbaran & Khorasani, 2012). First, a partial fault of 20% in rotor 1 $t = 20sec$ and a second 35% loss of effectiveness in rotor number 4 at t = 35sec are considered. The same control strategy as in (Amoozgar et al., 2012)is used but with a Fault detection and diagnosis system in place.

Mohammad et al. (Amoozgar et al., 2012) proposed an FTC using adaptive PID controller for a quadcopter UAV in 15% and 20% faults in all the four motors. A FIS is used for real-time tuning of the PID gains. Tracking errors and rate of change of tracking errors are used to make the system return to desired working conditions in the event of rotor failure. The control scheme proposed here was similar to that of (Ranjbaran & Khorasani, 2012) in that FPID was employed for control but differed in that the designer (Amoozgar et al., 2012) only considers restoring the vehicle to its hover position.

According to (Lippiello & Ruggiero, 2014), there is an assumption that the loss of a propeller has been already detected. The system switches to emergency landing modalities with an already planned trajectory. The quadcopter is modelled as a birotor by proposing the turning off of the motor on the same quadcopter axis of the broken propeller.

(Sadeghzadeh et al., 2012) implemented a GS-PID with Fault Detection and Diagnosis (FDD) system which provided the time, location and magnitude of fault occurrence. Here the PID gains are tuned for both the normal and faulty rotor conditions.

2.9 Summary of Research Gaps

It can be concluded that most of the research work in literature approach the issue of quadcopter stabilization during rotor fault by application of a Fault Diagnosis and Isolation system (FDI). This necessitates the use of more than one controller which then necessitates switching between the controllers depending on the rotor's effectiveness. This prolongs the fault recovery time and it is also expensive.

An EKF was used to optimize a Takagi Sugeno FIS for PID base controller in (Gautam & Ha, 2013) for rotational motion and position control a quadcopter. The proposed controller showed good performance against wind disturbances bringing the quadcopter back to stability quickly because of smart selection technique and tuning of active fuzzy MF parameters. The proposed controller was however not tested for a quadcopter under rotor failure.

(Liu et al., 2016) used EKF to optimize a Mamdani FIS for a state feedback base controller for a quadcopter subjected to 20% rotor failure. Performance in vertical direction and angular action was investigated for 20% loss of effectiveness of the rotor. A fault recovery time of about 6 seconds for vertical motion and altitude loss of 0.2m were recorded. 9 seconds of recovery time for angular control was also recorded.

Motivated by the work done in (Liu et al., 2016) and (Gautam & Ha, 2013) and also by the need to reduce the system fault recovery time and the loss of altitude from hover position, an Extended Kalman filter optimized Fuzzy PID system is proposed for rotational motion and altitude control of a quadcopter UAV system subjected to 100% rotor fault.

CHAPTER THREE

METHODOLOGY

3.1 General design overview

In this chapter, a fault tolerant controller is designed and implemented on a quadcopter flight control board as follows:

i. Raw data is obtained from two orientation sensors, accelerometer and gyroscope. The Kalman Filter is used fuse data from these two sensors to cancel out errors from each. The noise free data obtained is then used to calculate the orientation tracking error. Raw data and filtered data will be graphically compared to show the effectiveness of the fusing the two sensor signals.

ii. Determination of optimal PID gains to serve as reference gains for the generation of PWM control signals. The PWM signals properly adjusts the thrust of motors to compensate for the actuator faults with less time delay. For Conventional PID controllers, fixed gains are used. This does not give a better performance for an unpredictable and wide range of operating conditions. In this scheme, FLC is used to tune the gains of the base controller. However, designing and examining the number of rules and Membership Functions (MFs) depends on expert experience. There is no systematic method. Moreover, for systems that are nonlinear with large uncertainties, the Fuzzy Inference System (FIS) has no learning and adaptive capabilities. Hence, the MFs and fuzzy rules are initially designed and then updated in real time using EKF technique. The EKF is used to adjust the MF parameters depending on the operating condition of the quadcopter UAV to minimize the incurred errors.

iii. The closed loop system is simulated using MATLAB/SIMULINK. For purposes of performance comparison, a conventional PID, Fuzzy-PID and EKF optimized Fuzzy-PID will be simulated.

iv. Finally, the prototype of the designed controller was implemented into a quadcopter control board.

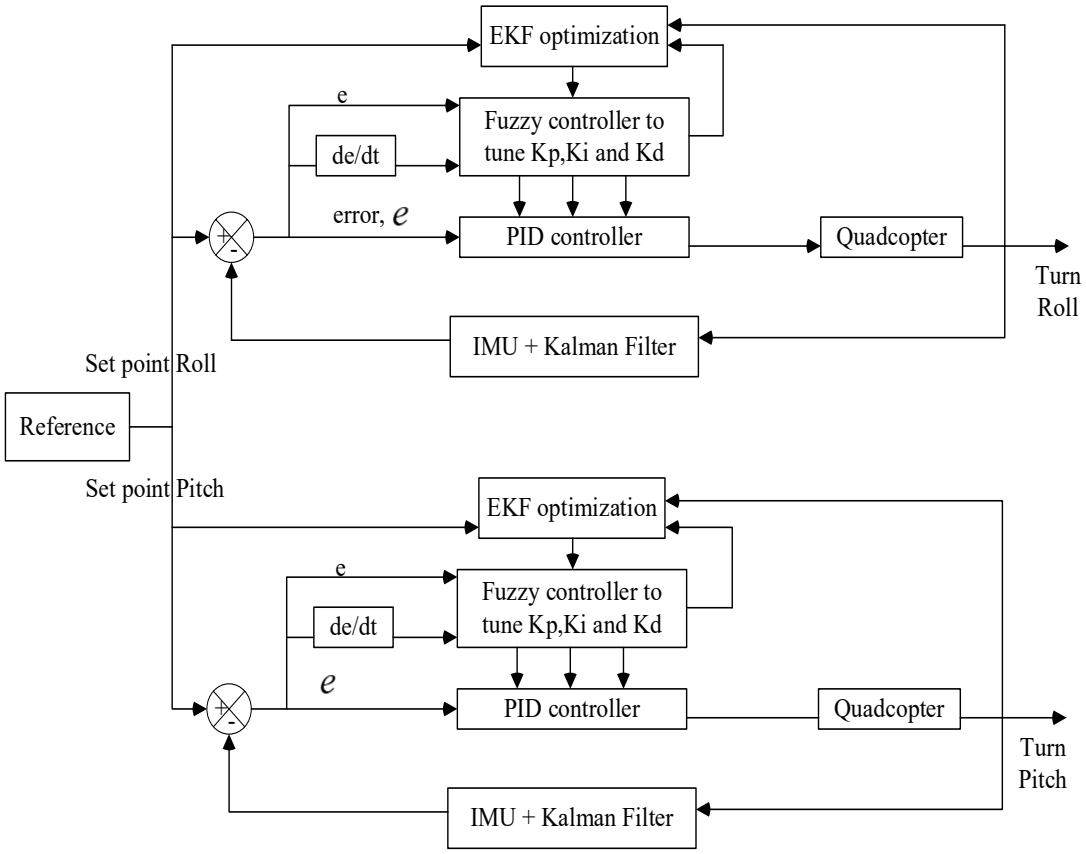

Figure 3.1: General control strategy

3.2 Determination of Attitude angles

To determine the attitude angles, the Inertial Measurement Unit (IMU) is used. It is composed of gyroscope and accelerometer. Whereas an accelerometer gives absolute measurement of the quadcopter attitude, the motors on the quadcopter produce a lot of vibrations introducing significant noise into the accelerometer reading. The gyroscope is much less affected by the vibrations but its readings are prone to drifting over time from the actual angle. Kalman filter is therefore proposed to merge the two sensor measurements to achieve better estimates, redundancy and drift compensation.

The sensor used in this research for determination of these angles is MPU6050 IMU sensor in Figure 3.2 (Sanjeev, 2018).

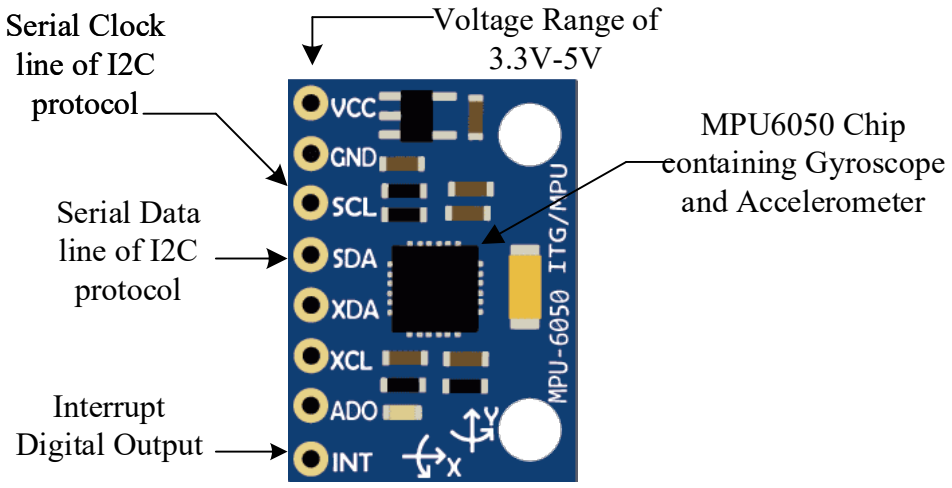

Figure 3.2: MPU 6050 Pinout on a GY-521 board

The MPU6050 chip is powered by 3.3V supply but a voltage regulator on the GY-521 board allows up to 5V supply.

The gyroscope and accelerometer are both embedded inside the MPU6050. Six values are obtained as output i.e. Three from the gyroscope and three from the accelerometer. MPU6050 uses I2C (inter-integrated circuit) protocol for communication with the Arduino microcontroller (Sanjeev, 2018).

3.2.1 Interfacing the MPU 6050 and the Arduino Uno

The Arduino and the MPU 6050 communicates through the I2C (inter-integrated circuit) protocol. This protocol has to be added in the Arduino library to establish communication between the two devices. Pin connection between the MPU 6050 and Arduino is as shown in Figure 3.3 (Sanjeev, 2018).

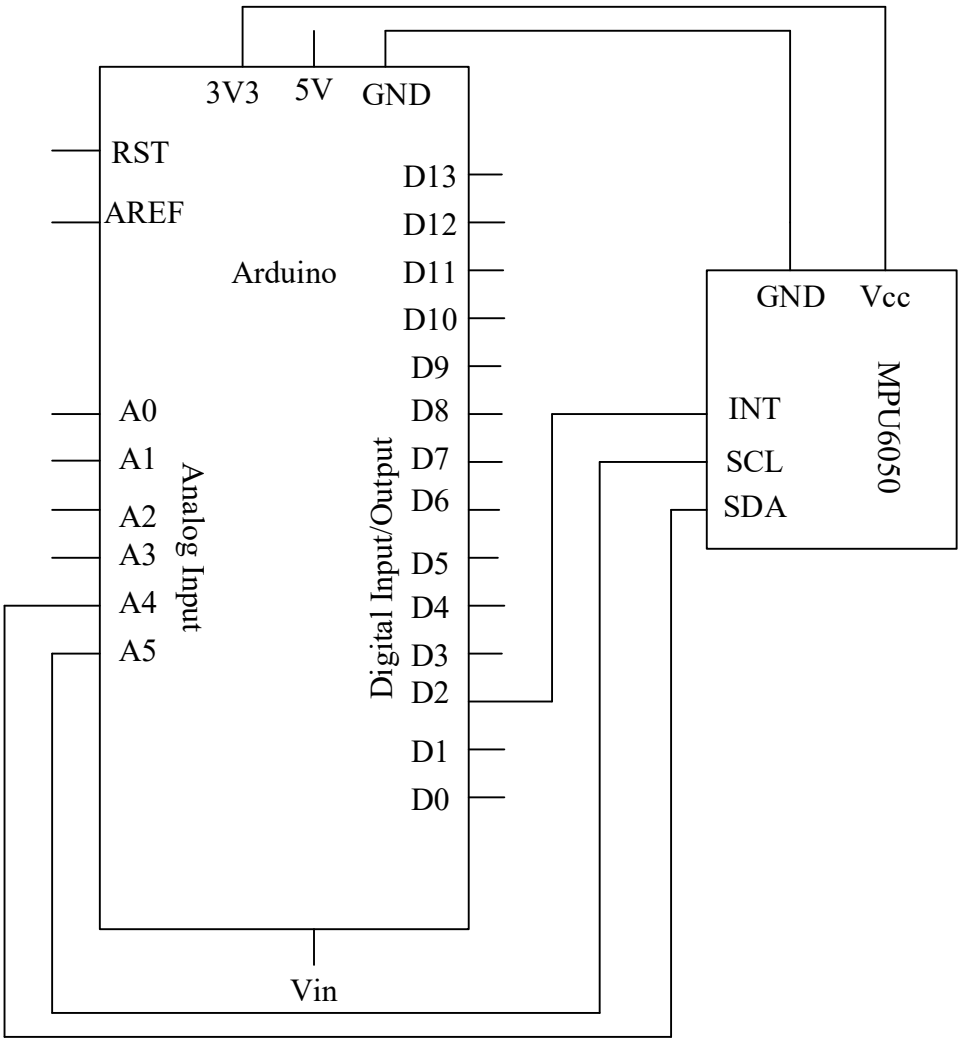

Figure 3.3: Arduino board and MPU6050 connection diagram

Setting up the I2C line involves connecting Arduino pins 4 and 5 to pins SDA and SCL of the MPU6050 respectively.

3.2.2 Conversion of signals to degrees

While the accelerometer measures acceleration (g's) in three dimensions, the gyroscope measures degrees per second both giving analog signals as output measurements.

3.2.2.1 The gyroscope

Here, the bits are translated into angles using the MPU6050 data sheet.

Equations (3.1) to (3.8) are derived based on work done in (Sparkfun, n.d.) and (Haoyu Electronics, n.d.). The sensitivity is given as 3.33mV/deg/s, but for a 10 bit resolution, the sensitivity has to be scaled between 0-1023 as shown:

$$\text{scaled sensitivity} = (\text{datasheet sensitivity})/1023 \times \text{supply voltage} \tag{3.1}$$

Given the data sheet sensitivity was 3.33mV/deg/s (Sparkfun, n.d.), then the scaled sensitivity was obtained as 1.023.

So the resultant degrees per second becomes

$$\theta / s = (G_r - G_z) / \text{sensitivity} \tag{3.2}$$

where, G_r are the read values from the sensor, G_z is the value when it is stationary.

$$\theta / s = (G_r - G_z) / 1.023 \tag{3.3}$$

The results are in degrees per second and are therefore converted to degrees in code by *gyroAngle += gyroRate*dtime/1000* where gyroRate = (gyroAdc-gyroZero)/sensitivity.

The gyroscope readings however, drift overtime (Haoyu Electronics, n.d.). There is therefore need to fuse these readings with those of the accelerometer.

3.2.2.2 The Accelerometer

The accelerometer data does not have any drift but is unfortunately affected by motor vibrations and are therefore not reliable for a short period. The analog readings from the accelerometer are converted into degrees by:

$$a_x = (x_a - y_0) / \text{sensitivity} \tag{3.4}$$

Where a_x is the g read by the accelerometer, x_a the analogue reading and y_0 the value when the reading is $0g$ i.e. when it is horizontal and during calibration. Sensitivity is obtained from the datasheet (Sparkfun, n.d.).

From the datasheet (Sparkfun, n.d.), the zero value can also be obtained. The zero voltage at $0g$ is approximately $1.5V$. This can be translated into bits by:

$$\text{scaled sensitivity} = \text{zeroVoltage}/3.3 \times 1023 \tag{3.5}$$

Similarly, with sensitivity converted to bits as was done earlier with the gyro, the final equation for accelerometer becomes:

$$a_x = (x_a - y_0) / 465 \tag{3.6}$$

These accelerometer readings are now in $g's$ defining the inertial force vector and therefore have to be converted to degrees by using the $arccos$ () function.

The accelerometer angle calculations in three dimension are computed as follows:

$$accXangle = \arccos\left(a_x / R\right)$$

$$accYangle = \arccos\left(a_y / R\right) \tag{3.7}$$

$$accZangle = \arccos\left(a_z / R\right)$$

R is the force vector derived in (Starlino, n.d.) and in this context implemented as:

$$R = \sqrt{\left(a_x^2 + a_y^2 + a_z^2\right)} \tag{3.8}$$

The output is in radians and can be transformed into degrees by multiplying the result with $\frac{180}{\pi}$.

3.2.3 Signal fusion using Kalman and Complimentary filter

The Kalman filter is a two-step process: the system acts as a Predictor; i.e. it uses the model of the system, the current state and the input vector to predict the future state considering the covariance error. In application, the filter takes the gyroscope measurements and calculates attitude estimations based on the gyroscope rates, and makes a prediction estimate of the error covariance (Boutayeb et al., 1997).

The measurement update phase, which is the second phase, corrects the predicted state and the estimated covariance error according to the measurements and its noise covariance. These are then used to calculate the Kalman gain. The accelerometer data is incorporated to aid the gyroscope measurement. These two values are multiplied by the Kalman gain taking a percentage of each measurement based on their noise

characteristics. The process is described as follows (Boutayeb et al., 1997; Kobayashi et al., 1995b):

System state

Equations (3.9) through (3.20) are based on literature derived from (Boutayeb et al., 1997; Kobayashi et al., 1995b):

The state of the system is given by:

$$x_k = \mathbf{F}x_{k-1} + \mathbf{B}u_k + \omega_k \tag{3.9}$$

In this case, equation (2.43) can be written as:

$$x_k = \mathbf{F}x_{k-1} + \mathbf{B}\dot{\theta}_k + \omega_k \tag{3.10}$$

where

- x_{k-1} is the previous state
- $x_k = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}$ is the state matrix showing that the output of the filter will be the angle θ and the bias $\dot{\theta}_b$. The bias is the amount the gyroscope has drifted. The true gyroRate is therefore calculated by subtracting the bias from the gyroscope measurements.
- $\mathbf{F} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$ is the transition model.
- u_k is the control input which is also known as the rate $\dot{\theta}$ and here it represents the gyroscope measurement in degrees per second.
- $\mathbf{B} = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$ is the control input model
- $\omega_k \sim N(0, Q_k)$ is the process noise. It is a Gaussian distribution with a zero mean and process noise covariance matrix Q_k. Q_k is considered here as the covariance matrix of the state estimate of the accelerometer and bias as shown:

$$\mathbf{Q}_k = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t \tag{3.11}$$

Since Q_k depends on current time k, the accelerometer variance Q_θ and the variance

of the bias $Q_{\dot{\theta}_b}$ is multiplied by the delta time Δt. The constants are set based on trial and error i.e. if the estimated angle starts to drift, $Q_{\dot{\theta}_b}$ has to be increased and if the estimate tends to be slow, the value of Q_θ has to be decreased to make it more responsive.

The measurement z_k is determined as:

$$z_k = \mathbf{H}x_k + v_k \tag{3.12}$$

$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ is the observation model and maps the state space into observed space. Measurement is the accelerometer data.

$v_k \sim N(0, R)$ is the measurement noise of Gaussian distribution with a zero mean and R as the covariance. The measurement noise is equal to the variance of measurement since the covariance of the same variable is equal to the variance.

$$R = \text{var}(v_k) \tag{3.13}$$

The noise measurement variance $\text{var}(v_k)$ is also determined by trial and error i.e. if too high, the filter responds slowly as it trusts new measurements less and if too small, the accelerometer measurements are trusted too much which might result in overshoot and noisy measurements.

As a predictor, the filter tries to estimate the current state based on all the previous states and the gyroscope measurements.

$$\hat{x}_{k|k-1} = \mathbf{F}\hat{x}_{k-1|k-1} + \mathbf{B}\dot{\theta}_k \tag{3.14}$$

The a priori error covariance matrix $P_{k|k-1}$ can be estimated based on the previous error covariance matrix $\mathbf{P}_{k|k-1}$ as:

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}_k \tag{3.15}$$

This matrix estimates how much the current values of the estimated state can be trusted. The smaller it is, the more the current estimated state is trusted.

In the measurement update phase, a priori state $x_{k|k-1}$ is the estimate of the state matrix at the current time k based on the previous state of the system and the estimates before it. Therefore, the innovation is determined by the difference between the measurement z_k and the a priori state as:

$$\tilde{\mathbf{y}}_{\mathbf{k}} = z_k - \mathbf{H}\hat{x}_{k|k-1} \tag{3.16}$$

$\mathbf{H}$ maps the a priori state $x_{k|k-1}$ into the observed measurements i.e. the measurement from the accelerometer. The innovation covariance is given by:

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^{\mathbf{T}} + \mathbf{R} \tag{3.17}$$

$\mathbf{S}_k$ predicts how much the measurement based on the a priori error covariance matrix $\mathbf{P}_{k|k-1}$ and the measurement covariance $\mathbf{R}$ should be trusted. If the value of the measurement noise is bigger, then the value of $\mathbf{S}$ becomes larger, meaning that the incoming measurements cannot be trusted that much. This concept is much understood with introduction of the Kalman gain.

The Kalman gain is calculated as:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1} \tag{3.18}$$

Equation (3.18) indicates that for a small Kalman gain, *S* will be high meaning that the innovation is not trusted that much. This also means that the estimate of the state is trusted with the error covariance matrix ***P*** being small.

The a posteriori estimate (the estimate of the state at time k given observations up to and including at time k) of the current state can be updated by:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{3.19}$$

Equation (3.19) means that the estimate of the a priori state (calculated using previous state and the gyroscope measurements) is corrected with the accelerometer measurement. The a posteriori error covariance matrix is updated as follows:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1} \tag{3.20}$$

where $\mathbf{I}$ is an Identity matrix.

The implementation of equation (3.9) to (3.20) for Kalman filtering is done in C code in Arduino IDE as detailed in the appendix.

The complimentary filter is easy to implement since it has just one equation expressed as [68]:

$$\theta_k = G_g * (\theta_{k-1} + G_d * dt) + (0.020) * (A_d) \tag{3.21}$$

where θ_k is the complimentary filter angle, G_g is the gyroscope gain, θ_{k-1} is the previous complimentary filter angle, G_d is the gyroscope data, A_d is the accelerometer data

It sums weighted fractions of the accelerometer and the gyro angles creating more accurate orientation angles.

3.2.4 Implementation

Equations (3.1) to (3.21) are coded in Arduino IDE and uploaded into Arduino board in the set-up of Figure 3.4. The serial monitor is opened up with the baud rate set at 115200. The roll, pitch and yaw attitude angles should start streaming in from the IMU sensor. The test was performed as follows:

-First the IMU GY-521breakoutboard was tilted smoothly.

-Next, the board was then continually tilted with some vibrations, i.e. by tapping and shaking the board quickly.

The data was received on the Arduino platform serial monitor as in Figure 3.4:

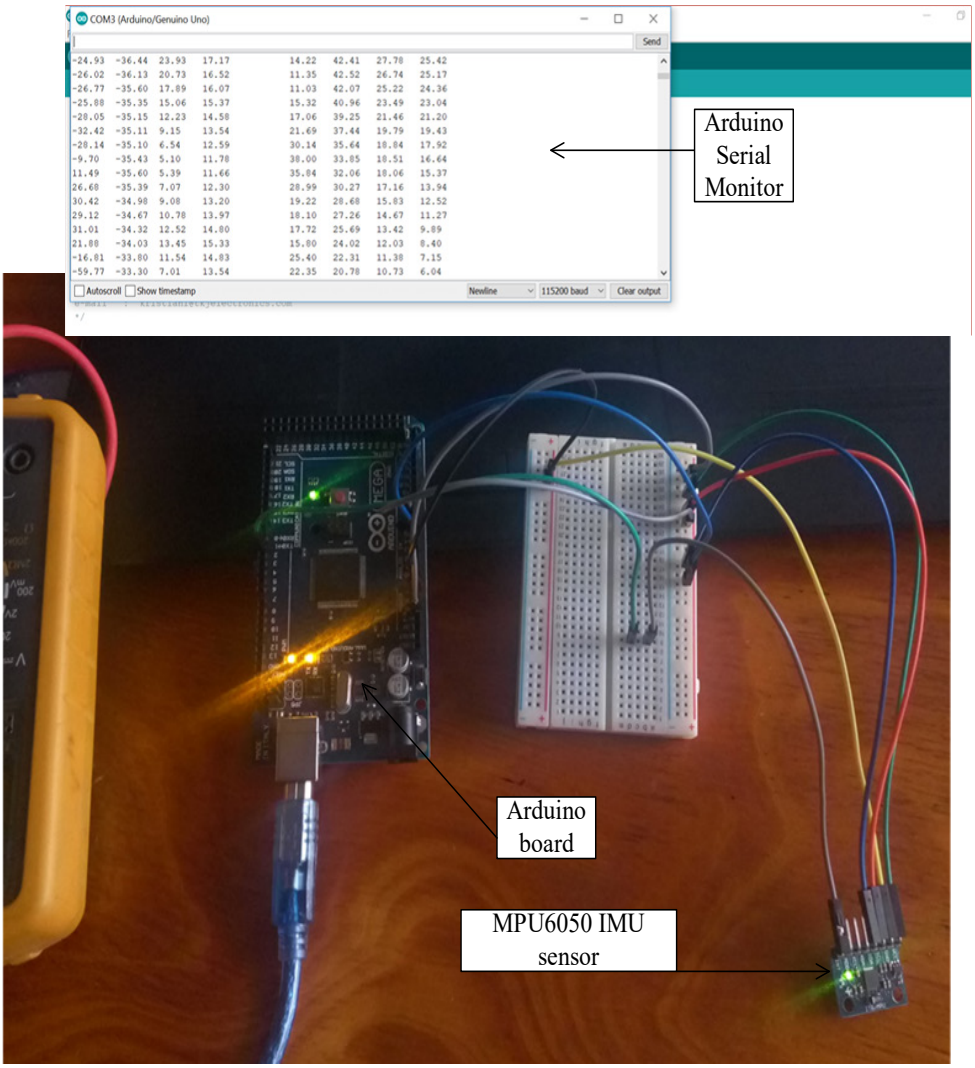

Figure 3.4: Generation of orientation angles

From Figure 3.4, the data from the orientation sensors was received in the Arduino serial monitor and then analyzed in MATLAB. The data was filtered and the performance of Kalman filter and the complimentary filter compared.

3.3 Controller Design

The equations used for modelling the quadcopter dynamics are listed in this section.

3.3.1 Summary of the necessary Equations needed for the Modelling of Quadcopter Dynamics in MATLAB/SIMULINK

a) Inertia tensor moment matrix J:

$$J = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix} \tag{3.22}$$

I_{xx}, I_{yy}, I_{zz} are the symmetrical Moments of Inertia of the quadcopter. The off-diagonal elements are zero because the quadcopter is symmetrical.

Pairwise differences in rotor angular speed ω_i with $i = 1; 2; 3; 4$, referring to the rotor number, results in the torques $\tau_{\emptyset}; \tau_{\theta}\ and\ \tau_{\Psi}$ in the body frame and causes the aerial vehicle to rotate about the $x, y\ or\ z -$ axis. The body torque from motors τ_m is given by:

$$\tau_m = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lb(\omega_4^{\,2} - \omega_2^{\,2}) \\ lb(\omega_1^{\,2} - \omega_3^{\,2}) \\ d(\omega_1^{\,2} - \omega_2^{\,2} + \omega_3^{\,2} - \omega_4^{\,2}) \end{bmatrix} \tag{3.23}$$

where τ_m = body torque from motors, $\tau_\phi, \tau_\theta, \tau_\psi$ = torque along the body axes x, y, z, b = thrust coefficient, d = drag proportionality consonant, l = arm length from the center mass

b) The control input vector

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -lb & 0 & lb \\ lb & 0 & -lb & 0 \\ d & -d & d & -d \end{bmatrix} \begin{bmatrix} \omega_1^{\,2} \\ \omega_2^{\,2} \\ \omega_3^{\,2} \\ \omega_4^{\,2} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{3.24}$$

where T is the total thrust.

c) Angular accelerations comprising of Roll, Pitch and Yaw rates

$$\begin{aligned}
\ddot{\phi} &= \frac{l}{I_{xx}}U_2 - \frac{J_r}{I_{xx}}\dot{\theta}\Omega_r + \frac{I_{yy}}{I_{xx}}\dot{\psi}\dot{\theta} - \frac{I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} \\
\ddot{\theta} &= \frac{l}{I_{yy}}U_3 - \frac{J_r}{I_{yy}}\dot{\phi}\Omega_r + \frac{I_{zz}}{I_{yy}}\dot{\psi}\dot{\phi} - \frac{I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} \\
\ddot{\psi} &= \frac{l}{I_{zz}}U_4 + \frac{I_{xx}}{I_{zz}}\dot{\theta}\dot{\phi} - \frac{I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi}
\end{aligned} \tag{3.25}$$

where J_r is the Rotor's inertial and Ω_r is the rotor's relative speed i.e. $\left(-\omega_1 + \omega_2 - \omega_3 + \omega_4\right)$

d) Linear Accelerations obtained as:

$$\begin{aligned}
\ddot{x} &= -\frac{U_1}{m}\left(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi\right) \\
\ddot{y} &= -\frac{U_1}{m}\left(\cos\phi\sin\theta\sin\psi - \cos\phi\sin\psi\right) \\
\ddot{z} &= g - \frac{U_1}{m}\left(\cos\phi\cos\theta\right)
\end{aligned} \tag{3.26}$$

3.3.2 PID Control

To stabilize the nonlinear quadcopter system, a base PID controller was first designed. Desired angles (input angles) are taken as setpoints and are then compared with feedback angles from the fused accelerometer and gyroscope measurements for the generation of the control error.

The controller regulates the angular velocity by computing the difference between the target rates and the rates read by the gyroscope. The three moment control inputs, $\tau_\phi, \tau_\theta, \tau_\psi$ are then determined using the obtained error and are then combined with the

lift control input as in equation (3.24) from the altitude controller. By converting these values to desired motor speeds, commands to the rotors are obtained and sent through the electronic speed controllers.

In this work, trial and error, as shown in Figure 3.5, is used to adjust the PID parameters based on the experts experience and gain adjustments can be done based on observations. An initial value of K_p which is less than 1 is chosen and increased guided by observation of the output. K_i, and K_d are initially set at zero and then adjusted appropriately depending on the overshoot and transient state.

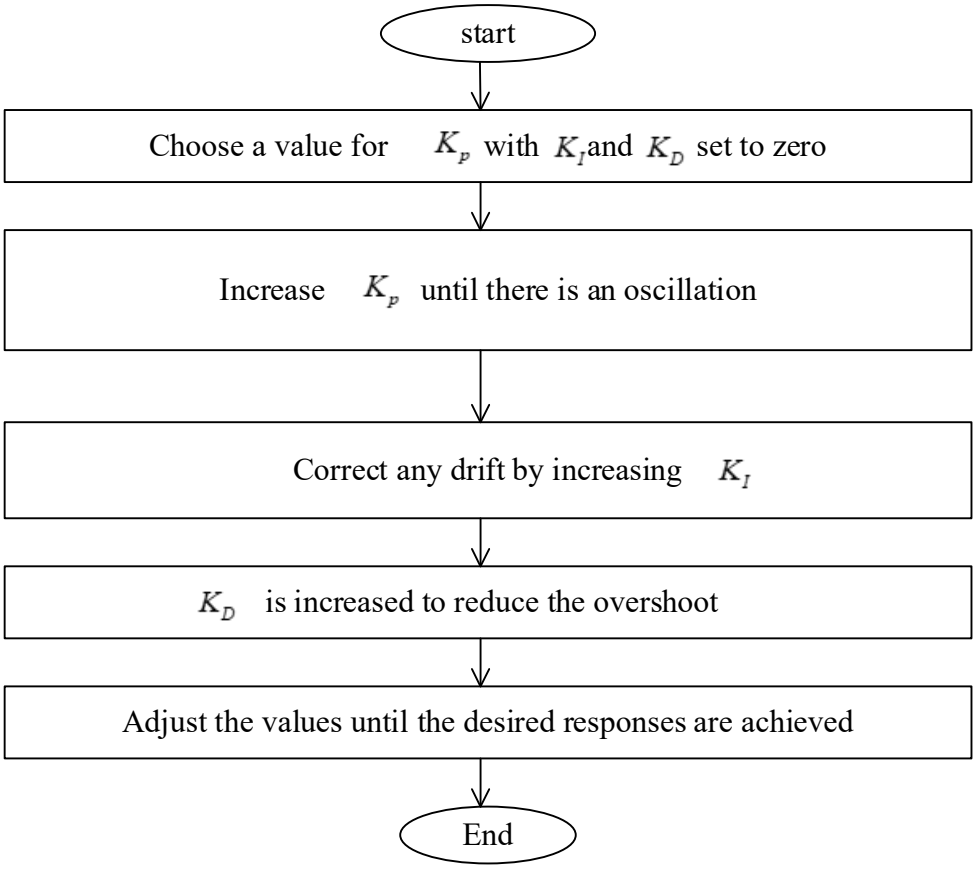

Figure 3.5: Flowchart for tuning PID gains.

The procedure described in Figure 3.5 is applied separately for Roll, Pitch, Yaw and altitude controllers. The composition of the PID controller in time domain is as shown in Figure 3.6 where K_p, K_i, and K_d are the PID gains. $u(t)$ is the control output, $e(t)$ is the error between the actual state and the desired state. The PID controller acts on

the error creating a rapid response to control output. It also acts in proportion to the error integrator eliminating the steady-state error (Kobayashi et al., 1995b).

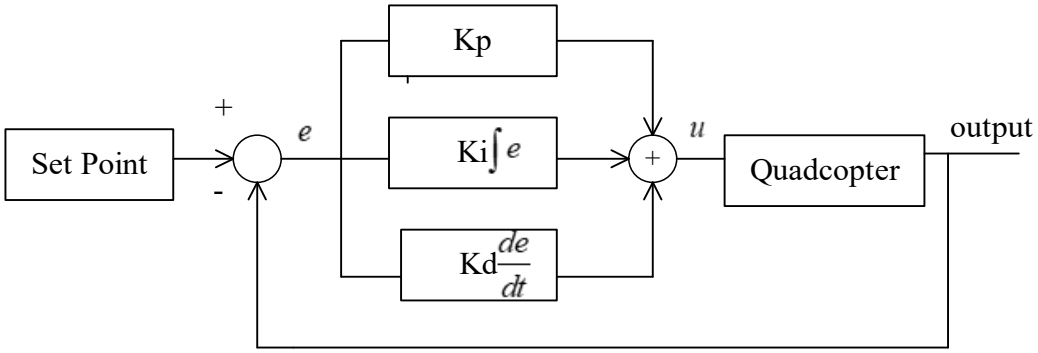

Figure 3.6: Block diagram of PID controller arrangement

Figure 3.7 is the MATLAB model for Figure 3.6. A step input was used with the feedback represented as *In 1* and *out 1* as the control signal. By regulating the gains, the desired output roll, pitch, yaw and altitude are obtained.

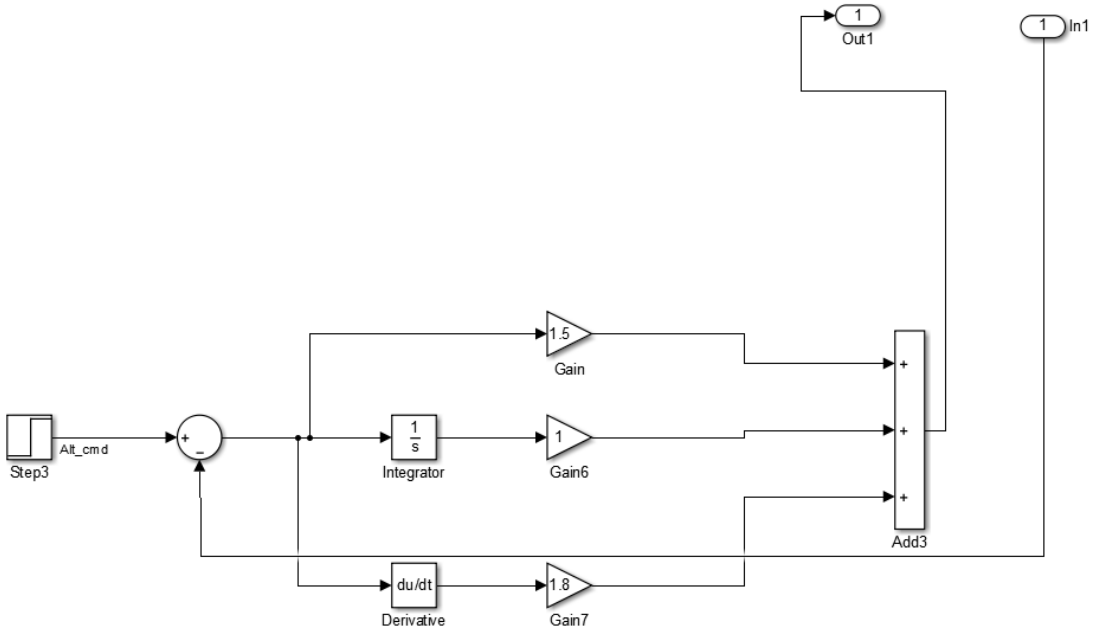

Figure 3.7: Altitude PID controller

Equations (3.22) to (3.26) were modeled in MATLAB/Simulink for the overall system control using PID strategy. Figure 3.8 shows the overall quadcopter PID control. This layout is used to construct the MATLAB model in Figure 3.9 of the equations

described in Equations (3.22) to (3.26). U is the control input to the quadcopter plant after comparison of the feedback information and the set point Roll, Yaw or Altitude.

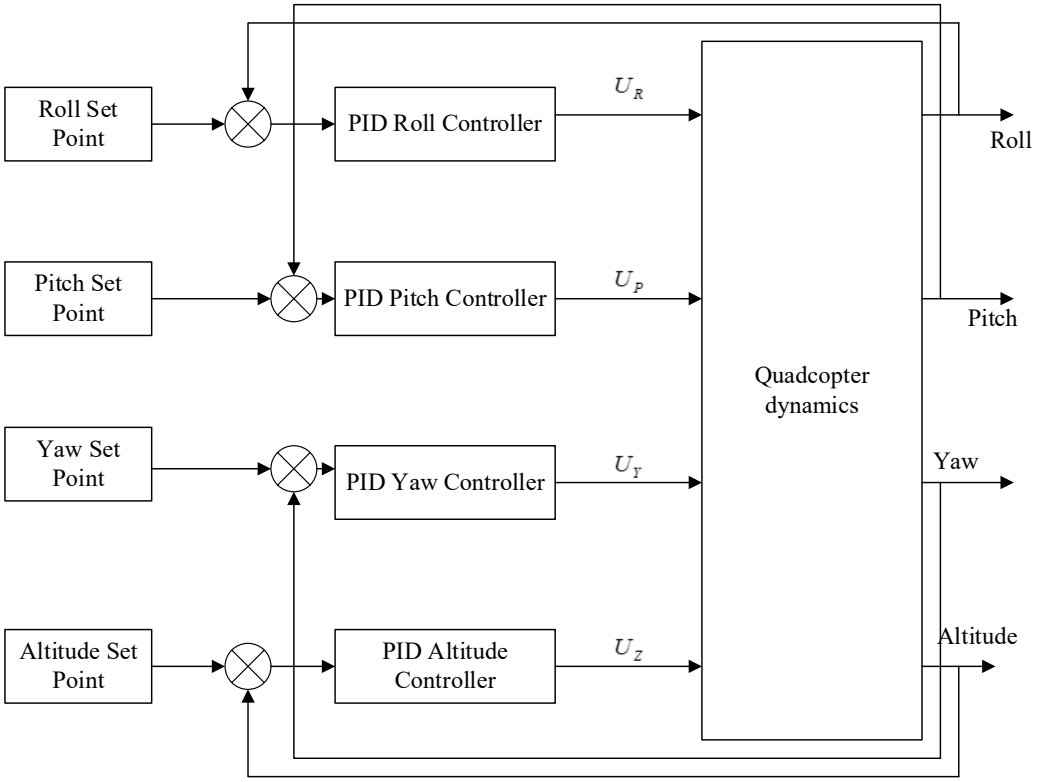

Figure 3.8: Block diagram of the overall Quadcopter PID control

Figure 3.9 is the MATLAB model of Figure 3.8 designed using Equations (3.22) to (3.26). The overall outputs are exported to workspace for plotting of the data for comparisons of PID performance with FuzzyPID and EKF-FuzzyPID.

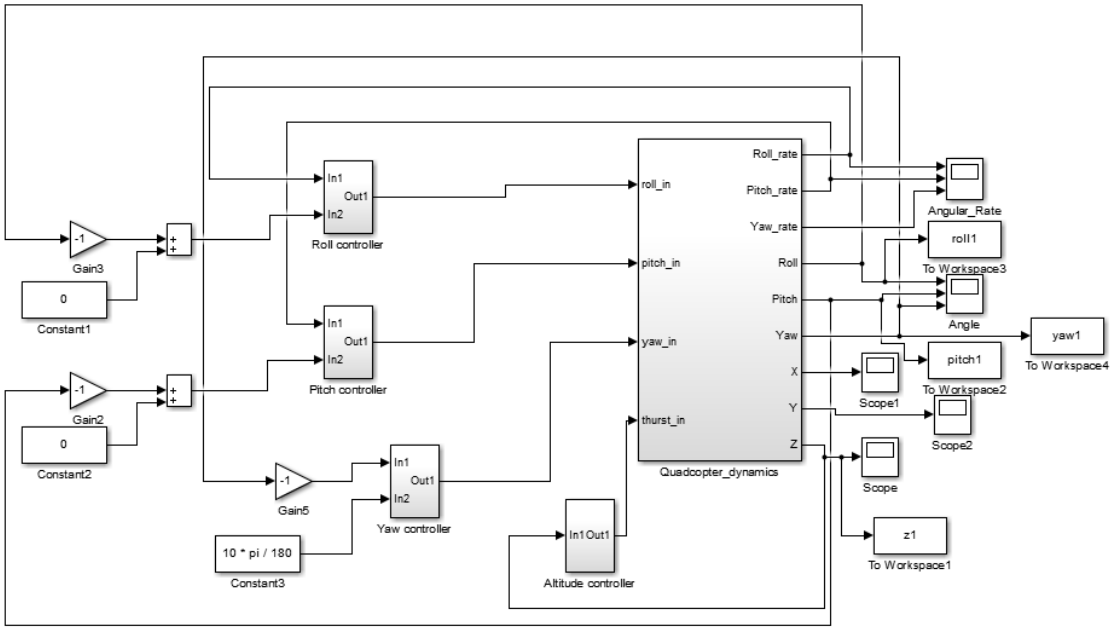

Figure 3.9: Quadcopter Simulink diagram

3.3.3 Fuzzy Membership Function (MF) Design

The triangular MFs are chosen for this work with the initial rule base constructed based on trial and error. The rule matrix is as shown in Tables 3.1 to 3.3. The inputs to the fuzzy system are the error e and the derivative of error, $\dot{e}$. PID values are the single output. The output is the K_p, K_i and K_d crisp value. The error is the reference angle or altitude minus their measured values.

Table 3-1: Fuzzy Matrix for K_p

				e			
$\dot{e}$	NB	NM	NS	Z	PS	PM	PB
NB	M	S	VS	VVS	VS	S	M
NM	B	M	S	VS	S	M	B
NS	VB	B	M	S	M	B	VB
Z	VVB	VB	B	M	B	VB	VVB
PS	VB	B	M	S	M	B	VB
PM	B	M	S	VS	S	M	B
PB	M	S	VS	VVS	VS	S	M

For the rule matrix: VVB = Very Very big, Z = Zero, PB = Positive Big, NB = Negative Big, PM Positive Medium, VVS = Very Very Small, NM = Negative Medium, PS = Positive Small, NS = Negative Small

Table 3-2: Fuzzy Matrix for K_i

$\dot{e}$ \ e	NB	NM	NS	Z	PS	PM	PB
NB	M	S	VS	VVS	VS	S	M
NM	B	M	S	VS	S	M	B
NS	VB	B	M	S	M	B	VB
Z	VVB	VB	B	M	B	VB	VVB
PS	VB	B	M	S	M	B	VB
PM	B	M	S	VS	S	M	B
PB	M	S	VS	VVS	VS	S	M

Table 3-3: Fuzzy matrix table for K_d

$\dot{e}$ \ e	NB	NM	NS	Z	PS	PM	PB
NB	M	B	VB	VVB	VB	B	M
NM	S	M	B	VB	B	M	S
NS	VS	S	M	B	M	S	VS
Z	VVS	VS	S	M	S	VS	VVS
PS	VS	S	M	B	M	S	VS
PM	S	M	B	VB	B	M	S
PB	M	B	VB	VVB	VB	B	M

The IF THEN rules are implemented as follows: "If the $error$ is Positive Small (PS), and the $change\ in\ error$ is Zero (Z), then K_p value is changed by Small (S) amount" as shown in Figure 3.10.

3.4 Extended Kalman Filter (EKF) Model for the Fuzzy System

Application of EKF to a general nonlinear system has been discussed in Chapter 2 Section 2.6.1. To apply the EKF for the fuzzy parameter tuning, firstly, the MF parameters are taken to form the state vector as follows:

1. μ fuzzy sets for the first input
2. ν fuzzy sets for the second input
3. κ fuzzy sets for the output

In this work, μ, ν and κ are totaling to 63 parameter values as

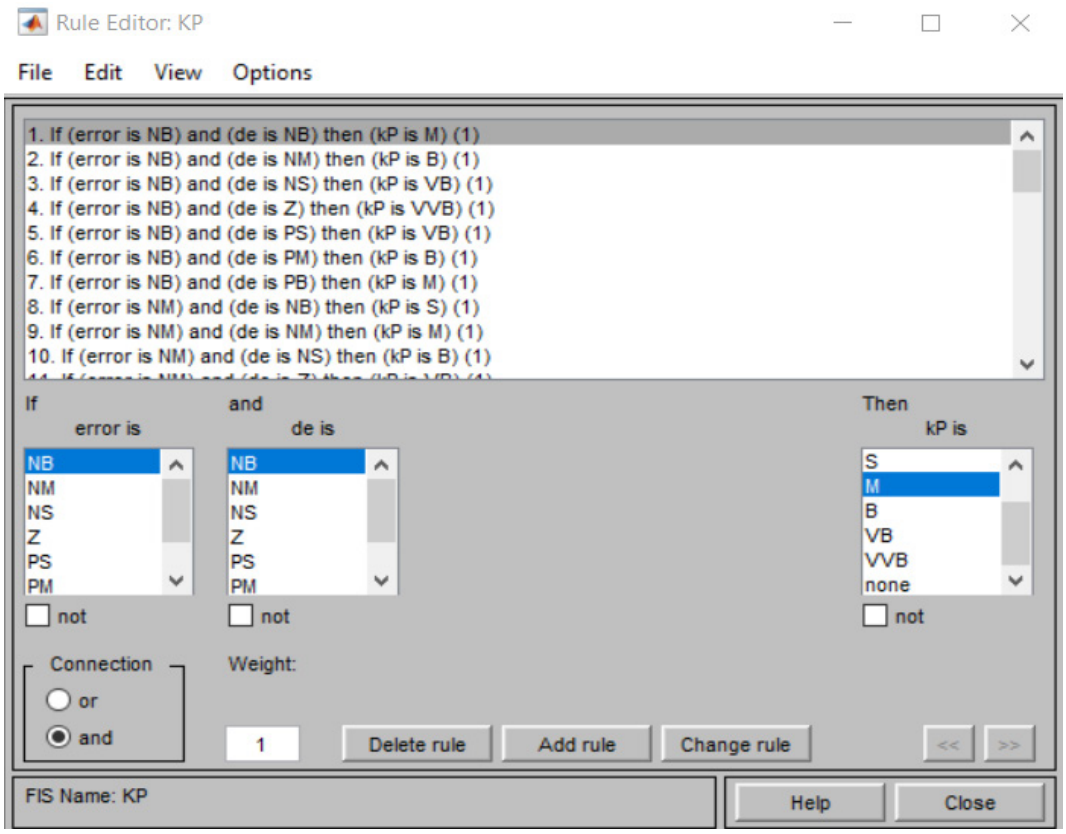

Figure 3.10: Fuzzy Rule base implementation in Matlab.

Figure 3.11 shows 7 triangular MFs each for both fuzzy inputs and the outputs. The error and error derivative,de are the inputs while the regulating value K_p is the fuzzy output. These MFs were constructed based on expert knowledge and experience.

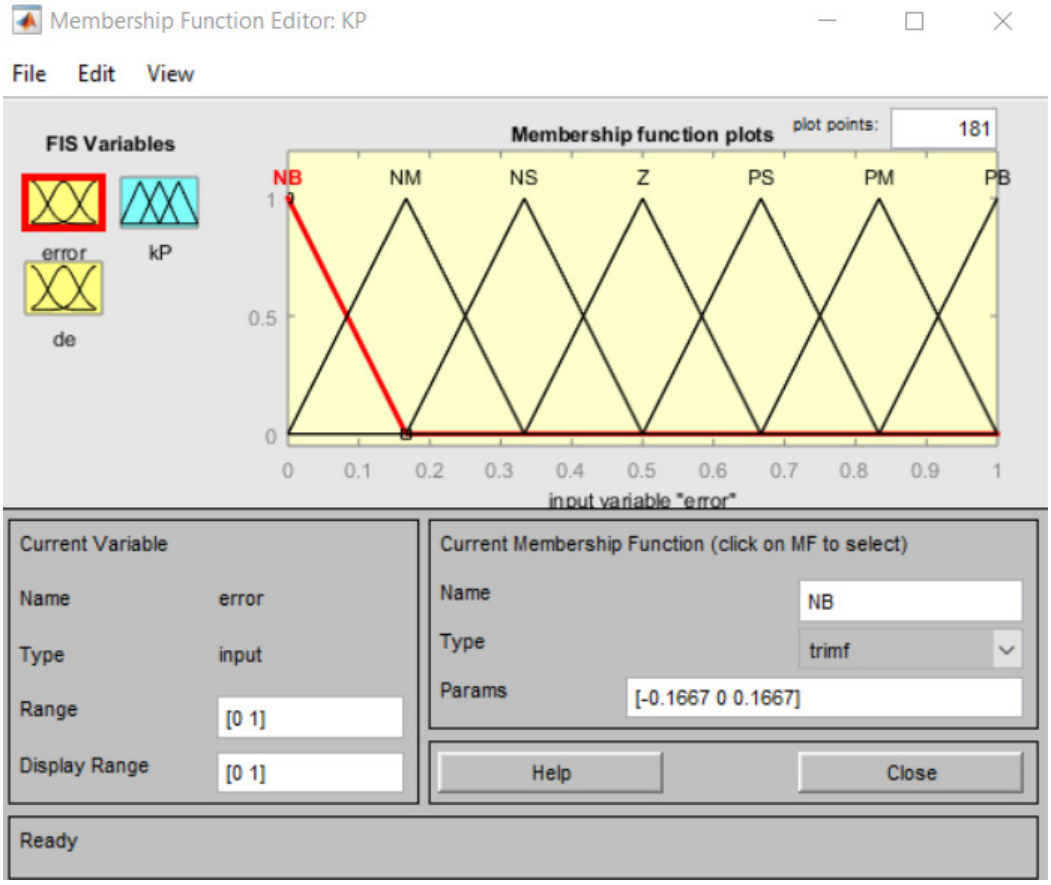

Figure 3.11: Membership function parameters

The state of the nonlinear system, derived from the MF parameters, can therefore be denoted as a column vector:

$$x = [a_{11}\ b_{11}\ c_{11}\ .....a_{\mu 1}\ b_{\mu 1}\ c_{\mu 1}\ a_{12}\ b_{12}\ c_{12}.....a_{v2}\ b_{v2}\ c_{v2}\ p_1\ q_1\ r_1.....p_k\ q_k\ r_k]^T \quad (3.27)$$

where

a_{ij}, b_{ij} – lower and upper halfwidths of the MFs respectively

c_{ij} – centroid of the i^{th} MF of the j^{th} input

μ – fuzzy sets for the first input

v – fuzzy sets for the second input

κ – fuzzy sets for the output

p,q,r – halfwidths and centroid of the output MF

The parameters defined in equation (3.27) form the state of the nonlinear system resulting in a form of MF optimization problem suitable for Kalman Filtering. The desired output of the fuzzy system is z_k and the actual output is denoted as $h(x_k)$ then (Anderson & Moore, 1979; Grewal & Andrews, 1993):

$$x_{(k+1)} = f(x_k) + w_k$$
$$z_k = h(x_k) + v_k \tag{3.28}$$

$w_k \sim N(0, Q_k)$ is random variable for process noise, $v_k \sim N(0, R_k)$ the measurement noise, x_k is the state of the system at time k, z_k is the measurement vector, R_k and Q_k is the measurement noise and process noise covariance respectively, $f(.)$ and $h(.)$ are nonlinear vector functions of the state. The tuning parameters, Q *and* R are the covariance matrices of the artificial noise process ω_k and v_k.

The problem addressed by the extended Kalman filter is to find an estimate $\hat{x}_{(k+1)}$ of x_k. The nonlinearities in (3.28) then can be expanded around the state estimate $\hat{x}$ using Taylor series to obtain (Simon, 2002):

$$f(x_k) = f(\hat{x}_k) + F_k \times (x_k - \hat{x}_k) + \text{higher order terms},$$
$$h(x_k) = h(\hat{x}_k) + H_k^T \times (x_k - \hat{x}_k) + \text{higher order terms},$$
(3.29)

Where $F_k = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}_k}$ and $H_k{}^T = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}_k}$

Neglecting the higher order terms, the system in equation (3.28) can be approximated as (Simon, 2002).

$$x_{k+1} = F_k x_k + w_k + \phi_k \tag{3.29}$$
$$z_k = H_k{}^T x_k + v_k + \varphi_k \tag{3.30}$$

where

$$\phi_k = f(\hat{x}_k) - F_k \hat{x}_k \text{ and } \quad \varphi_k = h(\hat{x}_k) - H_k{}^T \hat{x}_k$$

The desired estimate can be obtained by the recursion (Simon, 2002):

$$\begin{aligned} \hat{x}_k &= f(\hat{x}_{(k-1)}) + K_k[z_k - h(\hat{x}_{(k-1)})] \\ K_k &= P_k H_k (R_k + H_k^T P_k H_k)^{-1} \\ P_{(k+1)} &= F_k (P_{(k-1)} - K_k H_k^T P_{(k-1)}) F_k^T + Q_k \end{aligned} \tag{3.31}$$

where P_k and K_k is the state estimation error covariance matrix and the Kalman gain respectively.

3.5 Quadcopter System Implementation

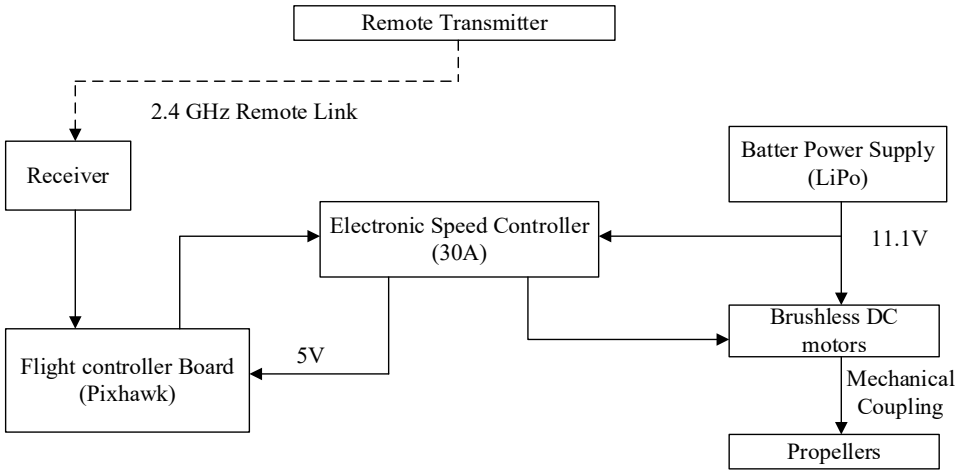

Figure 3.12: Block diagram of Quadcopter Hardware connection

A physical quadcopter was implemented using locally available components as shown in Figure 3.12.

3.5.1 Battery (LiPo)

The LiPo battery powers the quadcopter motors and sensors are all powered by using a battery pack. It should provide input voltage as per the microcontroller requirement providing enough power for a flight time of at least 20 minutes. For this project, there is need for a battery that is low cost, lightweight and rechargeable. The three main types of rechargeable batteries are the Lithium Polymer (LiPo) batteries, Nickel-Cadmium (NiCad) and Nickel-Metal Hydride (NiMH). These batteries have a low internal resistance allowing for high-power output and can operate at a large

temperature range. The NiCad battery suffers from decrease of the overall capacity over time. However, NiMH batteries can hold 30% more capacity compared to NiCad, but are affected by larger memory loss where the term memory refers to the amount of capacity the battery can store after each discharge (Pure Energy Solutions Inc., 2008).

LiPo batteries can also hold 30% more capacity but are much lighter than a NiMH battery. They are commonly used for powering remote controlled UAVs because of their longer flight time, lightweight, energy density and rechargeability.

A 2200mAh LiPo battery was selected with specifications shown in Figure 3.13(Pure Energy Solutions Inc., 2008):

Figure 3.13: A 2200mAh LiPo battery

LiPo batteries are specified in terms of their voltage, miliamperes-hour (mAh) and discharge rate. The battery used has the following specifications;

- Amp. Hour – 2200mAh
- Discharge rate – 25C
- Voltage – 11.1V

The current supplied by a LiPo battery is calculated as (Starlino, n.d.);

$$currentsupplied=Ah\ (Amperehour)*C\ (discharge\ rate) \quad (3.32)$$

$$current=2200/1000*25=55\ amperes \quad (3.33)$$

The motors draw currents equally. Therefore, the current going to each ESC and motor is:

$$current\ \ to\ \ each\ motor=55/4=15\ A. \quad (3.34)$$

This current is suitable as it can be handled by the 30A ESC and 15-25A motors.

3.5.2 DC Brushless Motors

DC brushless motors form the main component of the quadcopter rotor. The motors should be reliable with fast response for stabilization and control the vehicle. Moreover, the motors should be powerful enough to lift the quadcopter UAV and its payload. They should also be able to perform various maneuvers. A typical DJI 2212/920 motor used in this work is as shown in Figure 3.14 (Numan, 2017).

Figure 3.14: BLDC motor from DJI 2 4

Readytosky 2212 920KV brushless motors are used. The 2212 refers to the motor size that supports a frame size of up to 450 mm. The KV is theoretical increment in rotor's revolution per minute when the voltage would increase by 1V without any load. Therefore for a battery of 11.1V tends to revolve at 10212rpm without any load.

The BLDC motor model 2212 has a total weight of 38g with dimensions of 28(diameter)*22mm (long) and shaft size of 3.17(diameter)*33mm (long). The RPM/v is 920 KV and a maximum power of 102W. It works with a 2S – 3S Li-Po battery.

The motors chosen provide a maximum thrust of 1200g each, with 4 motors providing a total a total of 4800 g thrust.A quadcopter should be able to carry half of the thrust provided by the motors. In this case $1/2 * 4800g = 2400g$

In this work, the quadcopter weighs $2200\,g$ which is just below the maximum payload.The propellers are symmetrically pitched propellers as shown in Figure 3.16. A propeller generates aerodynamic lift force. A pair of clockwise and anticlockwise rotating propellers nullify the gyroscopic effect of each individual motor. With light propellers (commonly carbon fiber), the quadcopter is able to perform lift and hover at less than 50 percent motor rating.

3.5.3 Sizing of Propellers

Sizing of propeller was done based on Thrust-RPM mathematical model discussed in (Jirinec, 2011). The propeller variations based on quadcopter frame size and motor characteristics are as shown in Table 3.4.

Table 3.4: Propeller Sizing and Selection

Frame size	Prop Size	Motor size	KV
150mm	3”	1105-1306 or smaller	3000KV or higher
180mm	4”	1806	2600KV-3000KV
210mm	5”	2204-2208,2306	2300KV-2600KV
250mm	6”	2204-2208,2306	2000KV-2300KV
350mm	7”	2208	1600KV
450mm	8”,9”,10”	2212 or larger	1000KV or Lower

A large propeller of 8×3.5 with 8 inches in diameter and 3.5 inches in pitch was chosen as they are the most suitable for motors with low KV ratings. The large diameter and a lower pitch generates more torque (less turbulence) for lifting. The pitch is the travel distance of a single propeller's rotation.

3.5.4 Testing of Propeller symmetry

The propellers used were tested for symmetry as shown in Figures 3.15. A propeller with a broken tip was mounted on a symmetry tester allowed to settle. The unsymmetrical propellers are not able to maintain a horizontal orientation on the symmetry tester as shown in Figure 3.16. It is worth checking for symmetry of propellers (even for unbroken tips) since unsymmetrical propellers will not provide the desired thrust.

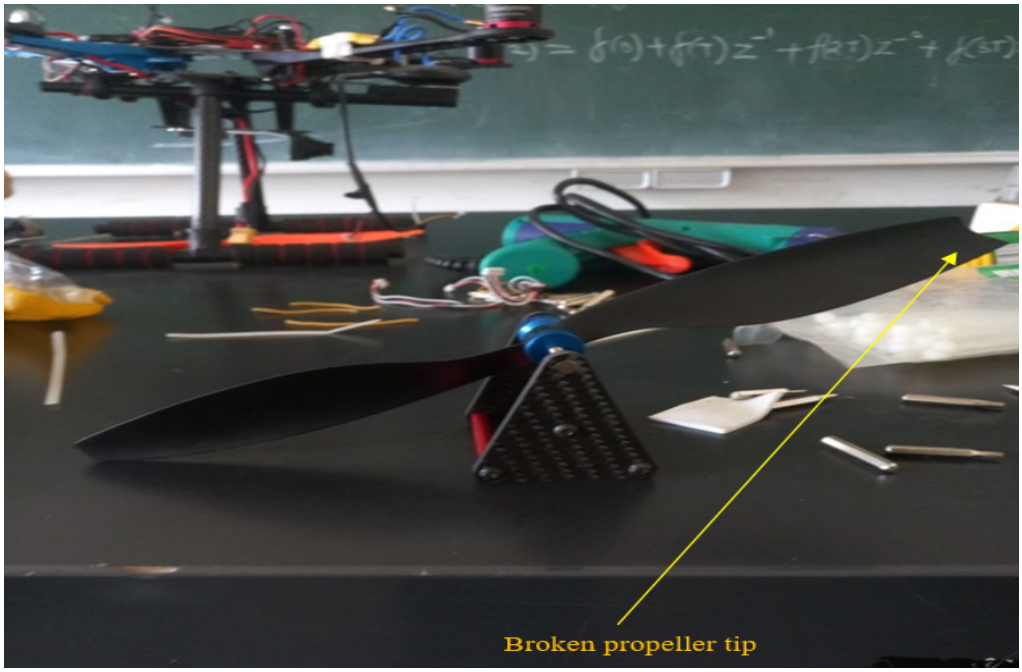

Figure 3.15: Unsymmetrical propeller due to broken tip

A suitable propeller should maintain an orientation shown in Figure 3.17.

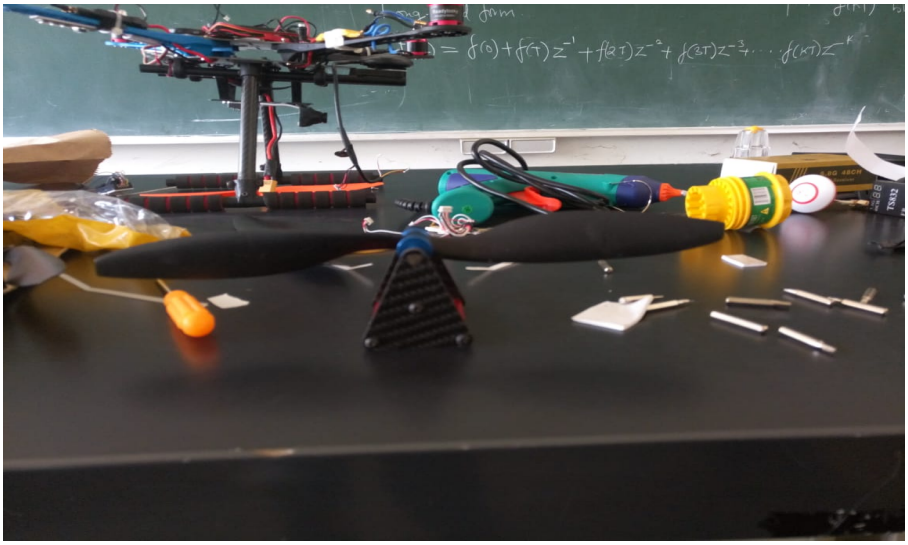

Figure 3.16: A symmetrically pitched Propeller

3.5.5 Electronic Speed Controllers (ESCs)

The motor speeds are varied using an electronic circuit known as Electronic Speed Controllers (ESCs) (Wil Selby, n.d.). Apart from providing for dynamic braking for the motors, the ESCs also convert the supplied battery DC voltage into 3 phased PWM. The ESC selected for this work has the following features:

- Low-voltage protection compatible with 2S – 4S LiPO battery,
- Over-heat protection
- Throttle signal loss protection
- Current rating of 30 A and 40A burst current
- Dimensions: 52*26*7 millimetres with a weight of 28 grams

3.5.6 PixHawk Controller Board

The quadcopter requires a microcontroller for flight controller. There are several fight controllers like Arducopter,APM, APM, APM 2.8 or HobbyKing KK2.1. A PixHawk flight controller is used in this scheme because it is highly recommended for general use. In addition to being cheap, it is easy to integrate with a series of analog and digital sensors that provide feedback states. The board uses a 168MHz Cortex M$F CPU

(256KB RAM). It runs a Real Time Operating System (RTOS) with 14 PWM outputs and other connectivity options for additional peripherals (Herrera, 2017).

Figure 3.17: PixHawk Board

Special features: Compatible with a PPM receiver, dimensions: 81.5 mm x 50 mm x 15.5 mm with a weight of 49 grams pounds, allows for automatic and manual mode and has micro SD slot for image storages.

The Pixhawk flight controller gives out signals to the ESC in PWM form. The ESC converts the PWM signal to the equivalent level of electrical power then feeds it to the motor.

CHAPTER FOUR

RESULTS AND DISCUSSION

This chapter presents analysis and discussion of the results obtained from the study. The developed EKF-FuzzyPID has been tested by simulation in MATLAB-SIMULINK and also in experiments. The EKF-FPID as well as the PID controller were separately applied to a physical quadcopter. The FLC-PID performance was tested in MATLAB/SIMULINK only. The performance of the quadcopter under PID, FuzzyPID and EKF-PID were compared to validate the effectiveness of the developed controller.

4.1 Roll and Pitch Estimation

A section of experimental values in the determination of attitude angles is as shown in Table 4.1. This set of data was obtained directly from the physical quadcopter setup while mimicking its maneuvers during flight. The significance of this experiment was to show that the actual feedback signals were noise free and that there was no bias. The columns Acc_r, Com_r, Kal_r and Gyr_r are the roll angles determined by accelerometer, complimentary filter, Kalman filter and Gyroscope respectively whereas Acc_p, Com_p, Kal_p and Gyr_p are the pitch angles. The data from the accelerometer are combined using estimation techniques Kalman filter and Complimentary filter.

Table 4-1: Attitude angles in degrees

A section of Feedback values obtained from orientation sensors in degrees								
Time(s)	**Acc_r**	**Com_r**	**Kal_r**	**Gyr_r**	**Acc_p**	**Com_p**	**Kal_p**	**Gyr_p**
1	-16.63	13.19	15.19	-39.84	-4.72	-6.96	-9.03	33.34
2	-16.85	11.3	14.61	-39.62	-6.61	-8.16	-10.27	32.03
3	-16.44	9.53	14	-39.44	-14.23	-9.7	-11.55	30.82
4	-12.8	8.04	13.4	-39.36	-26.14	-11.76	-12.88	29.85
5	-0.6	7.44	13.04	-39.35	-35.71	-14.05	-14.12	29.19
6	4.66	7.18	12.75	-39.42	-41.26	-16.32	-15.21	28.8
7	3.42	6.72	12.29	-39.64	-41.56	-18.2	-16.03	28.67
8	-4.79	5.73	11.65	-39.83	-37.64	-19.5	-16.54	28.74
9	-16.35	4.06	10.8	-39.96	-31.77	-20.13	-16.72	28.98
10	-24.38	1.96	9.77	-40.08	-24.78	-20.07	-16.55	29.4
11	-25.93	-0.07	8.75	-40.16	-16.98	-19.28	-16.01	30.01
12	-23.87	-1.8	7.82	-40.23	-12.76	-18.14	-15.24	30.76
13	-20.27	-3.18	6.98	-40.32	-9.34	-16.78	-14.36	31.55
14	-16.35	-4.18	6.26	-40.41	-8.27	-15.51	-13.53	32.28
15	-12.16	-4.79	5.68	-40.47	-8.37	-14.48	-12.88	32.85
16	-8.96	-5.11	5.23	-40.49	-8.95	-13.75	-12.46	33.22
17	-5.54	-5.16	4.87	-40.52	-9.31	-13.34	-12.31	33.32
18	-3.25	-5.01	4.62	-40.5	-10.91	-13.29	-12.44	33.19
19	-1.98	-4.77	4.42	-40.47	-13.09	-13.56	-12.78	32.89
20	0.29	-4.4	4.27	-40.45	-15.14	-14.02	-13.24	32.51
21	1.58	-3.93	4.19	-40.4	-16.49	-14.54	-13.73	32.14
22	-0.35	-3.59	4.1	-40.3	-19.22	-15.15	-14.2	31.83
23	-4.13	-3.56	3.9	-40.23	-21.16	-15.75	-14.59	31.65

Signal fusion is accomplished through Kalman and the Complimentary filtering. The values in Table 4.1 are plotted in MATLAB as in Figure 4.1 and 4.2.

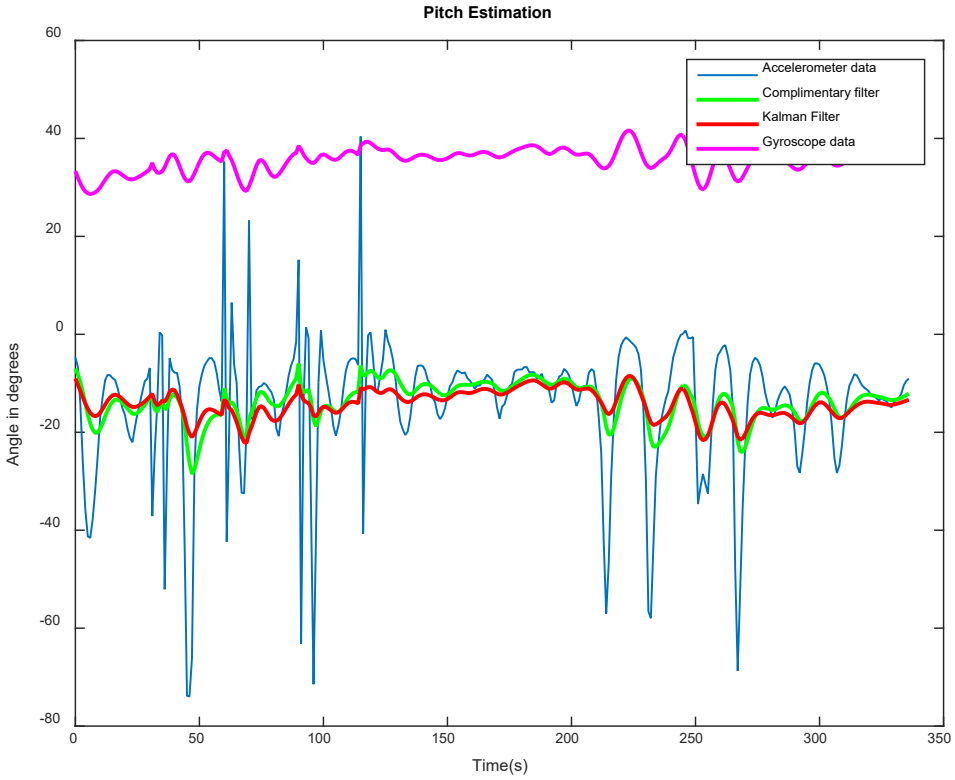

Figure 4.1: Pitch angle determination using different methods

Here, the reference point is zero. From Figure 4.1 and 4.2, it can be observed that the gyroscope data has drifted from zero and even continues to drift from its initial value. This sensor therefore needs recalibration every time it is used to determine the feedback angles. For longer period of use, it would be very inaccurate. The accelerometer gives very accurate feedback angles with time. It therefore performs better than the gyroscope for longer periods. However, its performance is affected by vibrations, from motors in this case, and the feedback signal is therefore very noisy as can be observed. The two sensor values can be complimented to combine their advantages and eliminate their shortcomings using the Complimentary filter and the Kalman filter. The two filters are able to eliminate the drifting and the noise through signal fusion. Complimentary filter is simple and easy to implement as compared to Kalman filter. It can however be observed that Kalman slightly performs better than the complimentary filter when filtering the vibration noise.

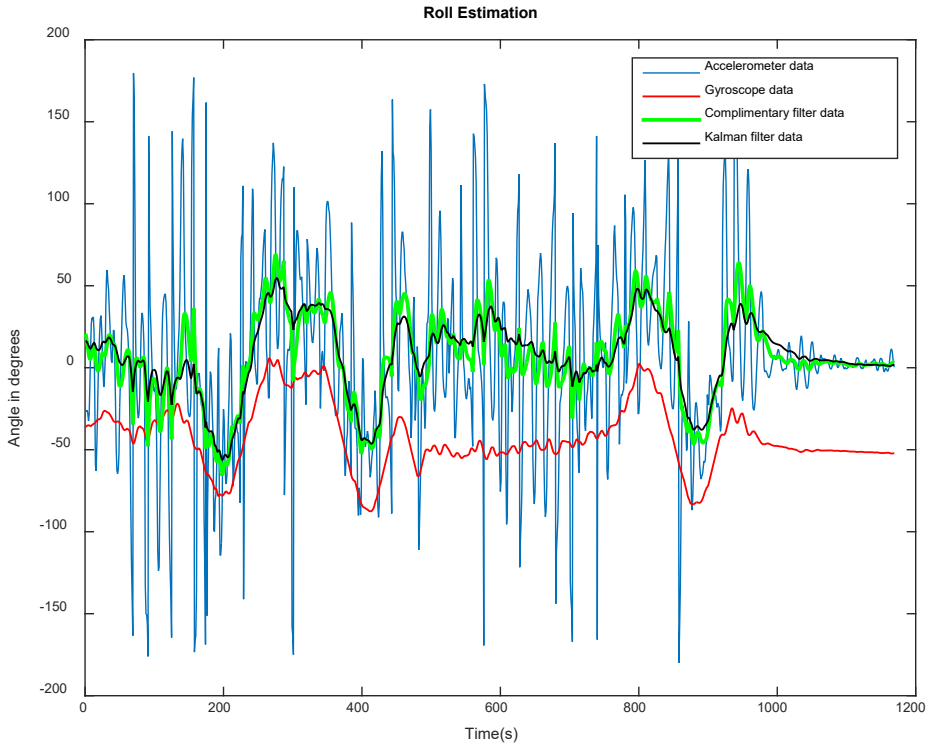

Figure 4.2: Pitch angle determination using different methods

4.2 PID control

Here, the PID gains used are initially guessed and then subsequently tuned depending on the observed system performance. The controller was simulated for a faultless system and faulty system with complete rotor failure introduced at 12th second for rotor number 1. The PID gains used and as a result the overshoot sustained are as shown in Table 4.2.

Table 4-2: PID values for Pitch Angle

	Pitch angle without fault	**Pitch angle with fault at 12th second**
K_P	1	1
K_I	0.05	0.05
K_D	1	1
% Overshoot	9%	12

Figure 4.3 shows pitch response for PID control. It can be observed that the angle completely deviates further from the set point 0 when a complete loss of rotor 1 occurs at $t = 12\ seconds$. The PID controller is unable to return the quadcopter to its setpoint orientation as far as the pitch is concerned.

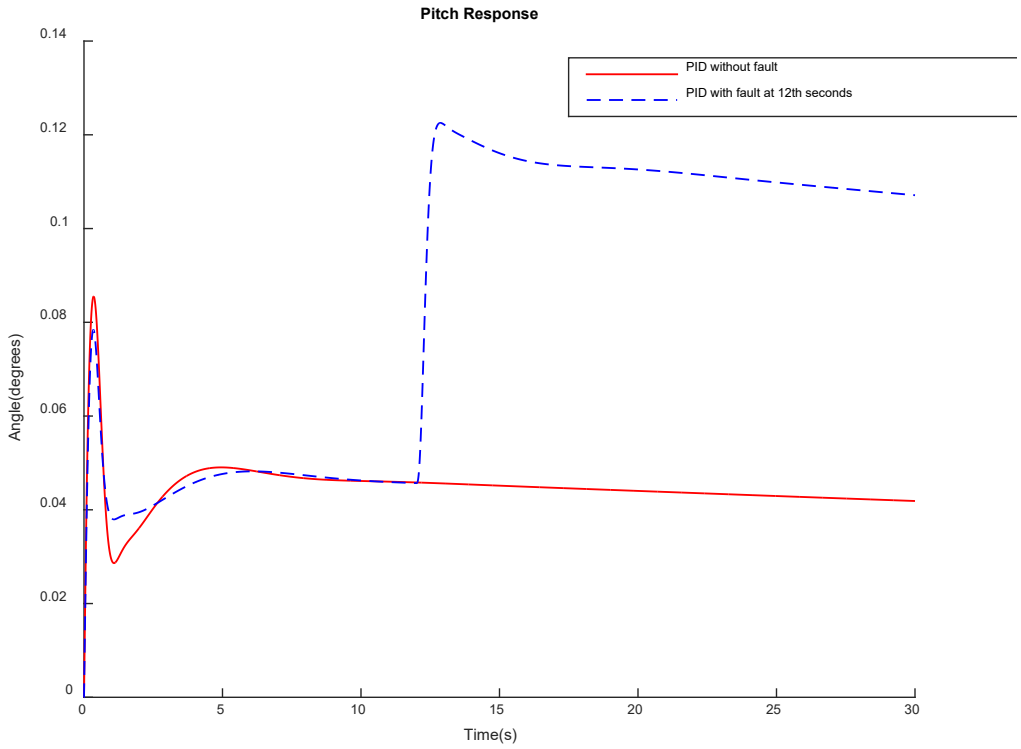

Figure 4.3: Pitch Response with PID control only

Table 4-3: PID values for Roll control

	Roll angle without fault	**Roll angle with fault at 12th second**
K_P	1	1
K_I	0.05	0.05
K_D	1	1
% Overshoot	35%	40%
% Undershoot	8%	9%

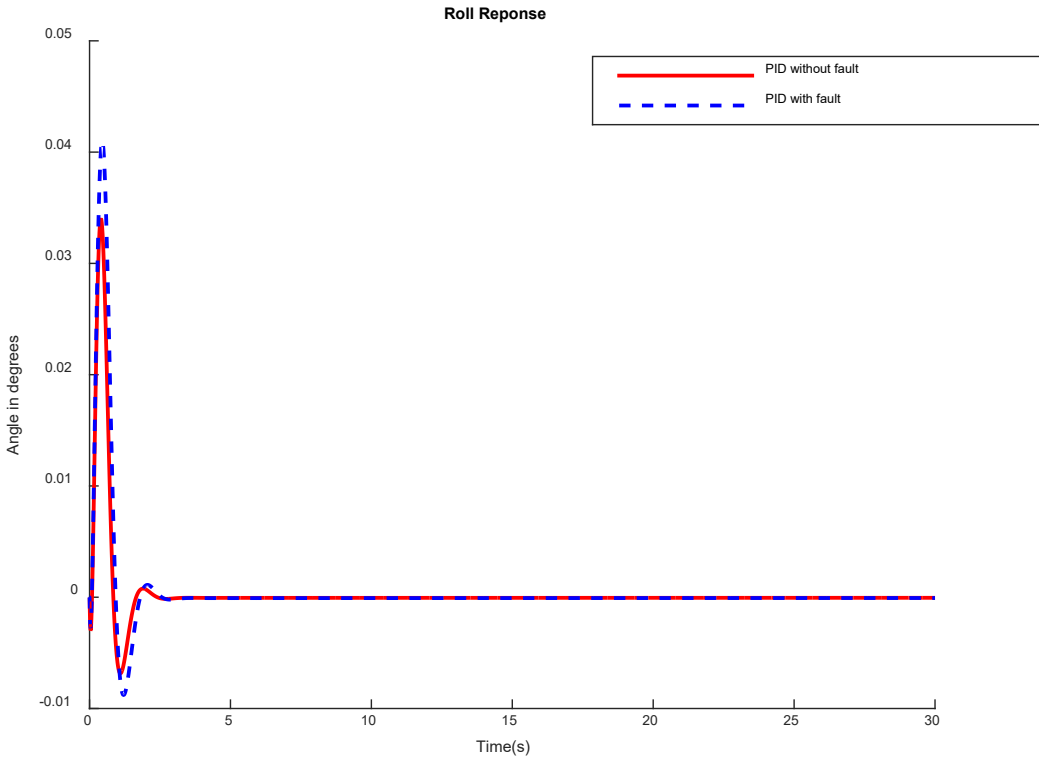

Figure 4.4: Roll Response with PID control only

Figure 4.4 shows that the complete loss of rotor 1 at 12th second does not affect the roll orientation, the quadcopter is able to maintain the set point roll angle.

Table 4-4: PID values for Yaw control

	Yaw angle without fault	**Yaw angle with fault at 12th second**
K_P	1	1
K_I	0	0
K_D	0.25	0.25
% Overshoot	6.5	6.5

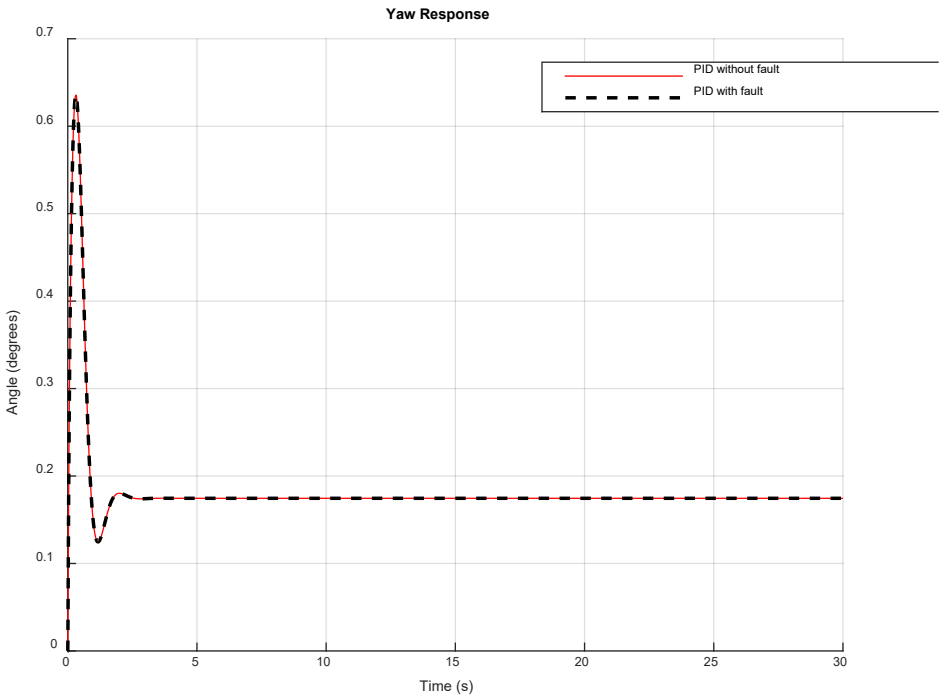

Figure 4.5: Yaw Response with PID control only

Figure 4.5 shows that the complete loss of rotor 1 at 12th second does not affect the yaw orientation, the quadcopter is able to maintain the set point angle (of $10\pi/180$ as set during the simulation).

Table 4-5: PID values for Angle control

	Altitude without fault	Altitude with fault at 12th second
K_P	1.6	1.5
K_I	1.1	1
K_D	1.5	1.8
% Overshoot	23	23
Settling time	12s	20s

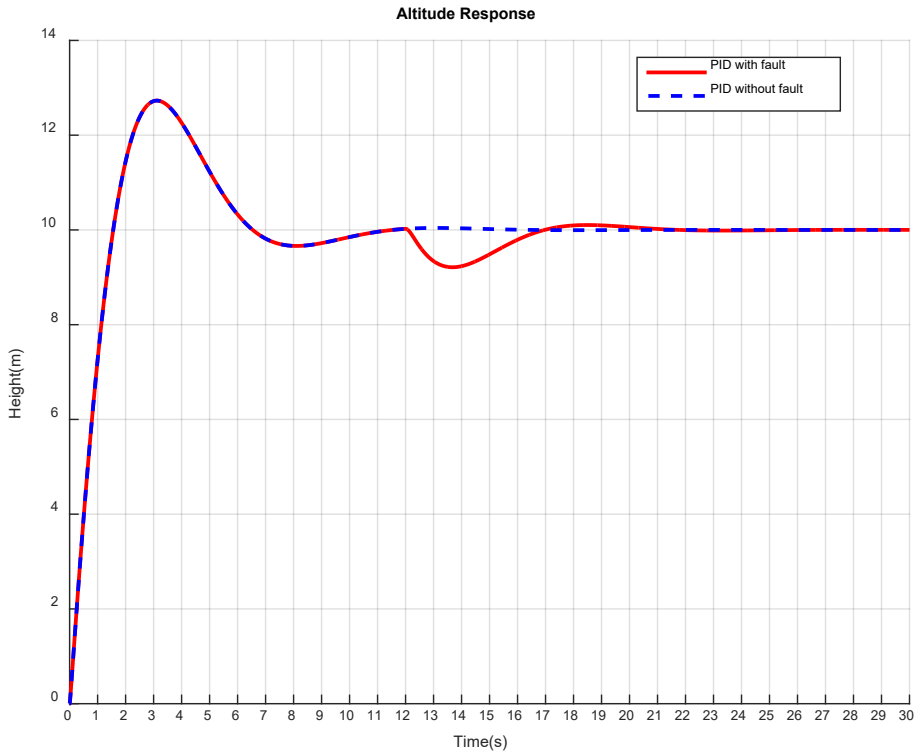

Figure 4.6: Altitude Response with PID control only

With an overshoot of 23%, the PID system is able to constrain the system around the set point height of $10m$ as depicted in Figure 4.6. The system is subjected to a complete failure of rotor 1 at 12th second and a loss in height is observed for $8\ seconds$. The quadcopter therefore becomes unstable for $8\ seconds$ and then control is regained.

4.3 Fuzzy PID control

The inputs to Fuzzy system are the error and the derivative of error from the quadcopter's altitude and attitude parameters. The fuzzy rules are used to adjust the PID gains automatically. The fuzzy output variables $\Delta K_p, \Delta K_i\ and\ \Delta K_d$ are extracted from the fuzzy matrix table discussed in section 3.3.2. The PID control parameters are adjusted using $\Delta K_p, \Delta K_i\ and\ \Delta K_d$ for real time attitude and altitude control. Figure 4.7 is the 3D representation of K_p as function of error, e and error derivative, $\frac{de}{dt}$. It shows how smooth or rough the system transitions as the gains are regulated.

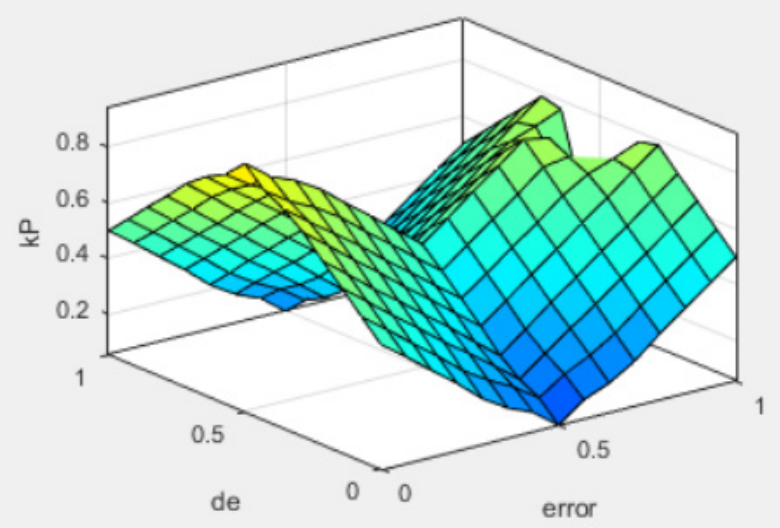

Figure 4.7: Variation of K_p

Figure 4.8 shows Pitch response when a rotor failure occurs at 12th second. It is evident that after the fault, the pitch response has shifted to 0.12 from the desired set point of 0. Here, the error is not corrected even after a period of time. The Fuzzy PID controller was able to return the quadcopter to the target set point even after the error is corrected. The time taken to return the system to desired set point angle is 5 seconds as can be observed from the figure.

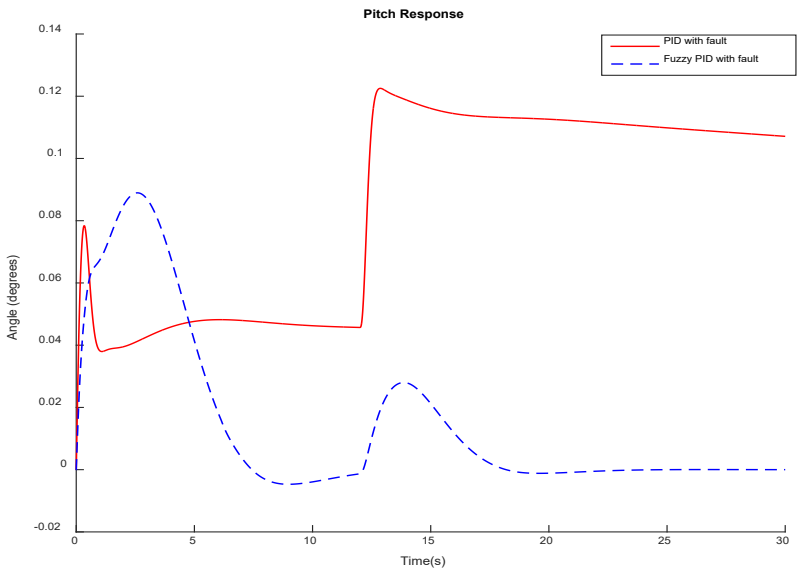

Figure 4.8: Pitch Response under actuator fault

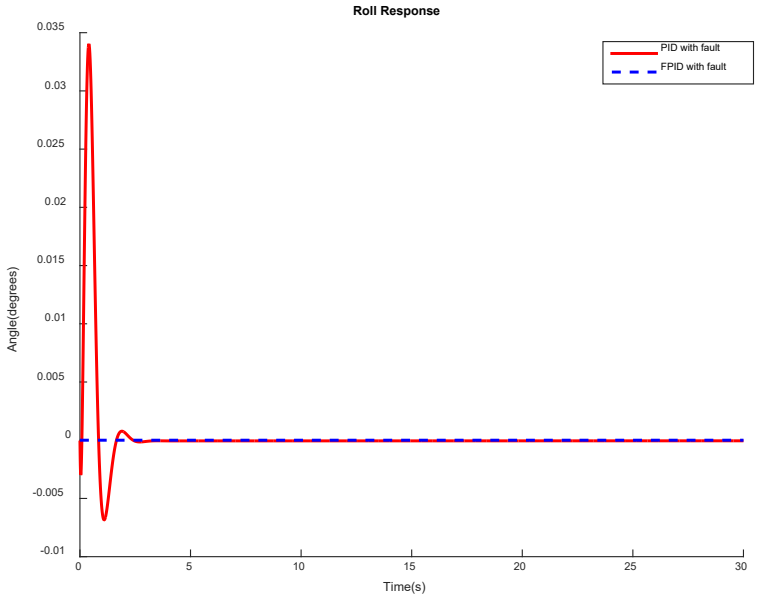

Figure 4.9: Roll Response under actuator fault

Figure 4.9 shows that whereas PID controller has an overshoot of 3%, the Fuzzy PID was able to completely eliminate the overshoot.

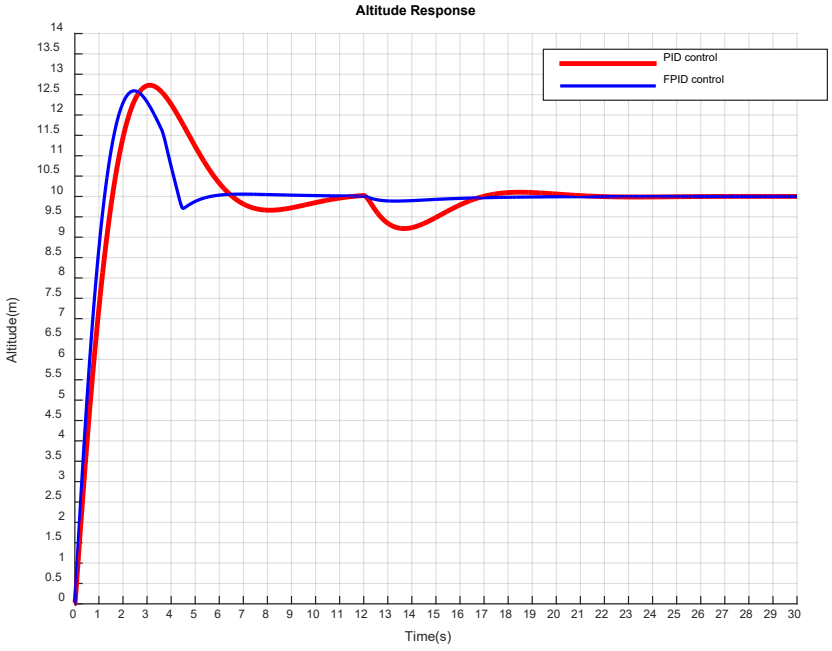

Figure 4.10: Altitude Response under actuator fault

The set point height in Figure 4.10 is $10m$ and the PID controller attains this height in 11 seconds according to Figure 4.6 with an overshoot of 28% while Fuzzy PID attains the target height in 5 seconds with an overshoot of 25%. A complete rotor failure is introduced in 12th second and it can be observed that PID controller registers a loss in height of $1m$ with a recovery time of $8\,seconds$ after the fault. The Fuzzy PID controller registers a slight loss in height of about $0.05m$ with a recovery time of $2.5\,seconds$ after the fault as captured in Table 4.5.

Table 4 6 shows the comparison between fuzzy PID and PID performance. Fault recovery is defined in this work as the time taken for the quadcopter system to come back to the desired hover height and maintain the desired orientation angles. FuzzyPID performed better than the PID controller as presented in Table 4.6

Table 4-6: Comparison between fuzzy PID and PID performance

	Fuzzy PID	**PID**
Overshoot	25%	28%
Settling time	5seconds	12 seconds
Fault Recovery time	2.5seconds	9 seconds

4.4 EKF Optimized Fuzzy PID controller

The membership function parameters used as EKF state vector were extracted from the initially designed membership functions. These are then adjusted according to section 3.4.

In this work, there are two fuzzy inputs and one output totaling to three variables with each variable having seven membership functions. This fuzzy system is therefore made of 21 membership functions. Each triangular MF used has a modal point and two half-widths totaling up to 3 parameters.

Tables 4.7, 4.8 and 4.9 are used to present the data set for the Membership Function parameters of the Fuzzy inputs. Table 4.10 presents the MF parameters of the Fuzzy output. These values were used as the initial estimates of the Extended Kalman Filter recursion algorithm for tuning of the initial Fuzzy Inference system. The values were

obtained from the left half widths, right half widths and the centroids of the triangular membership functions.

Table 4.7 shows the data for the MF parameters from the first input, e, here, b^-, b^+ are the left and right half widths and c is the centroid of the triangular MFs

Table 4-7: Membership function Parameters: left, right half widths and the centroid

K_P Error MF Parameters	b^-	b^+	c
1	-0.1667	0	0.1667
2	0	0.1667	0.3333
3	0.1667	0.3333	0.5
4	0.3333	0.5	0.6667
5	0.5	0.6667	0.8333
6	0.6667	0.8333	1
7	0.8333	1	1.167

Table 4.8 shows the data for the MF parameters from the second input error derivative, de. These values are derived from the centroid and half widths of the triangular membership functions used.

Table 4-8: Membership function Parameters: left, right half widths and the centroid

K_P Error dot MF Parameters	b^-	b^+	c
1	-0.1667	0	0.1667
2	0	0.1667	0.3333
3	0.1667	0.3333	0.5
4	0.3333	0.5	0.6667
5	0.5	0.6667	0.8333
6	0.6667	0.8333	1
7	0.8333	1	1.167

Table 4.9 shows the data for the MF parameters from the output MFs, K_p while Table 4.10 shows the data set obtained from the MF parameters of the outputs K_d and K_i.

Where,

b^-, b^+ are the left and right half widths and c is the centroid.

Table 4-9: Membership function Parameters: left, right half widths and the centroid

K_P Output MF Parameters	b^-	b^+	c
1	-0.1667	0	0.1667
2	0	0.1667	0.3333
3	0.1746	0.3412	0.5079
4	0.3333	0.5	0.6667
5	0.5	0.6667	0.8333
6	0.6667	0.8333	1
7	0.8333	1	1.167

Table 4-10: Membership function Parameters: left, right half widths and the centroid

K_I and K_D Output MF Parameters	b^-	b^+	c
1	-0.1667	0	0.1667
2	0	0.1667	0.3333
3	0.1667	0.3333	0.5
4	0.3333	0.5	0.6667
5	0.5	0.6667	0.8333
6	0.6667	0.8333	1
7	0.8333	1	1.167

The reference altitude was set at 10m so the goal of the controller is to maintain a 10m height even after a sudden complete loss of one rotor.

For 10 iterations, Figure 4.11 depicts the progress of training with Kalman filter indicating that its converging to better solutions.

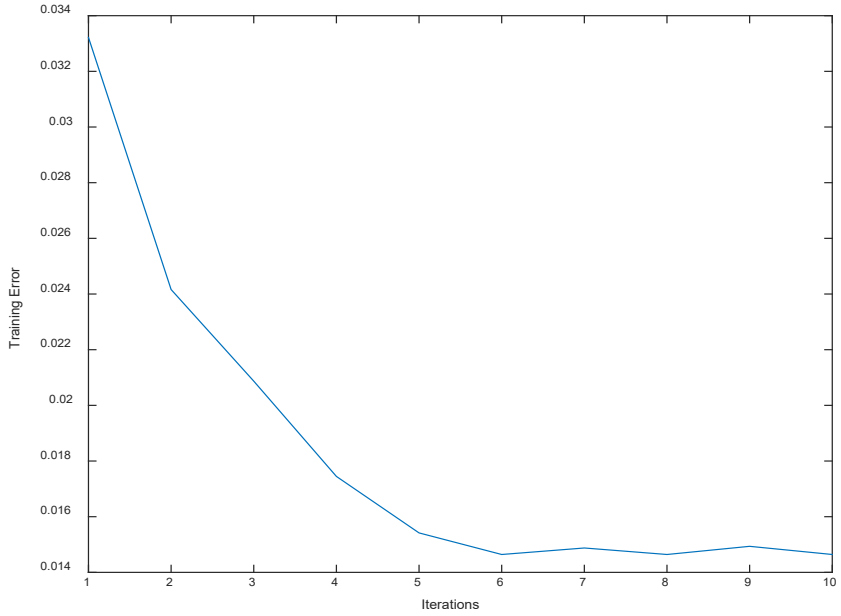

Figure 4.11: Training Progress

Altitude response under EKF optimized Fuzzy PID is as shown in Figure 4.12. There is no loss in height even after rotor failure is introduced at 12th second. The response takes 5 seconds to reach the target height as compared to 6 and 9 seconds for Fuzzy PID controller and PID controller respectively. The overshoot is compared in Table 4.11.

Table 4-11: Overshoot comparison

	EKF optimized FPID	**Fuzzy PID**	**PID**
Overshoot	18%	27%	30%
Fault Recovery time	1 seconds	2.5seconds	9 seconds

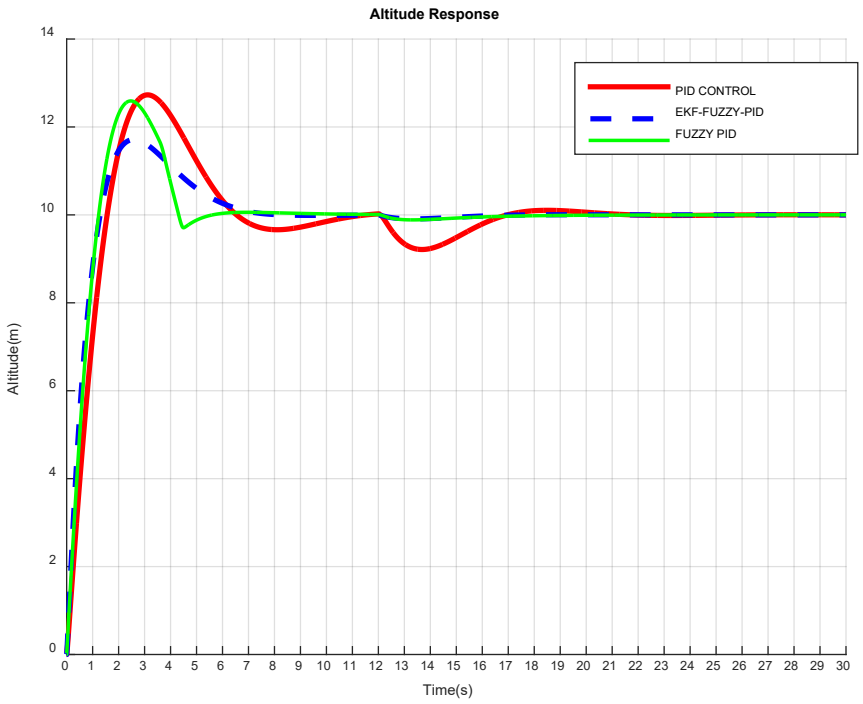

Figure 4.12: Performance of different controllers for the altitude

From Figure 4.12, the developed controller registered a negligible loss of altitude after rotor failure as compared to FuzzyPID which registered an approximate loss of about 0.05m. Table 4.11 shows the system recovered quickly from the effects of rotor failure for the developed controller as compared to FPID and PID controllers.

The Pitch response due to the various controllers is depicted in Table 4.12. FPID is used here as the tracking controller and from the overshoot values and fault recovery time, the optimized registered a quicker correction back to the desired set point orientation.

Table 4-12: Overshoot and recovery time comparison for Pitch Response

	EKF optimized FPID	**Fuzzy PID**
Overshoot	4%	9%
Recovery time after fault	4 seconds	10 seconds

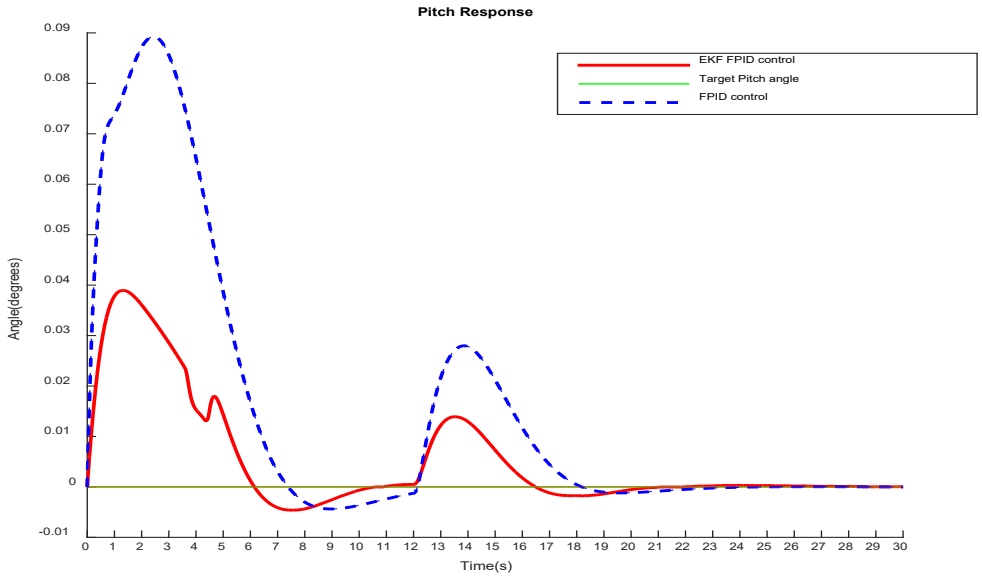

Figure 4.13: Pitch response

From Figure 4.13, the EKF FPID controller has an overshoot of 4 % compared to 9% for FPID. The rotor failure is introduced at 12th second with EKF FPID having an offset from the target by 0.01 and a recovery time of 4 seconds as compared to 0.03 offset and 10 seconds recovery time for FPID. The EKF FPID performs better than the FPID.

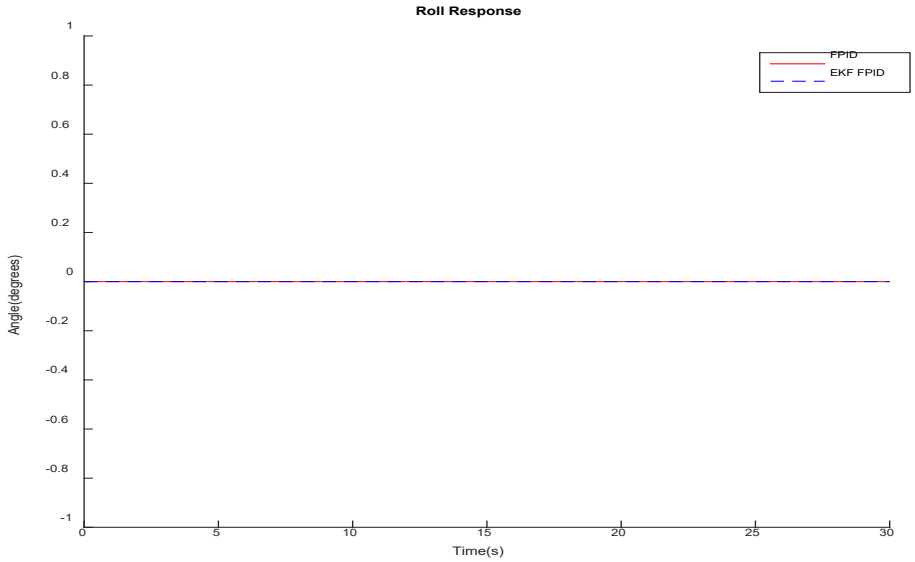

Figure 4.14: Roll response

The roll angle is not affected by the introduction of complete failure on rotor 1 at 12th second

4.5 The Implemented Quadcopter and its Performance Analysis

The complete design and fabricated quadcopter UAV is as shown in Figure 4.15

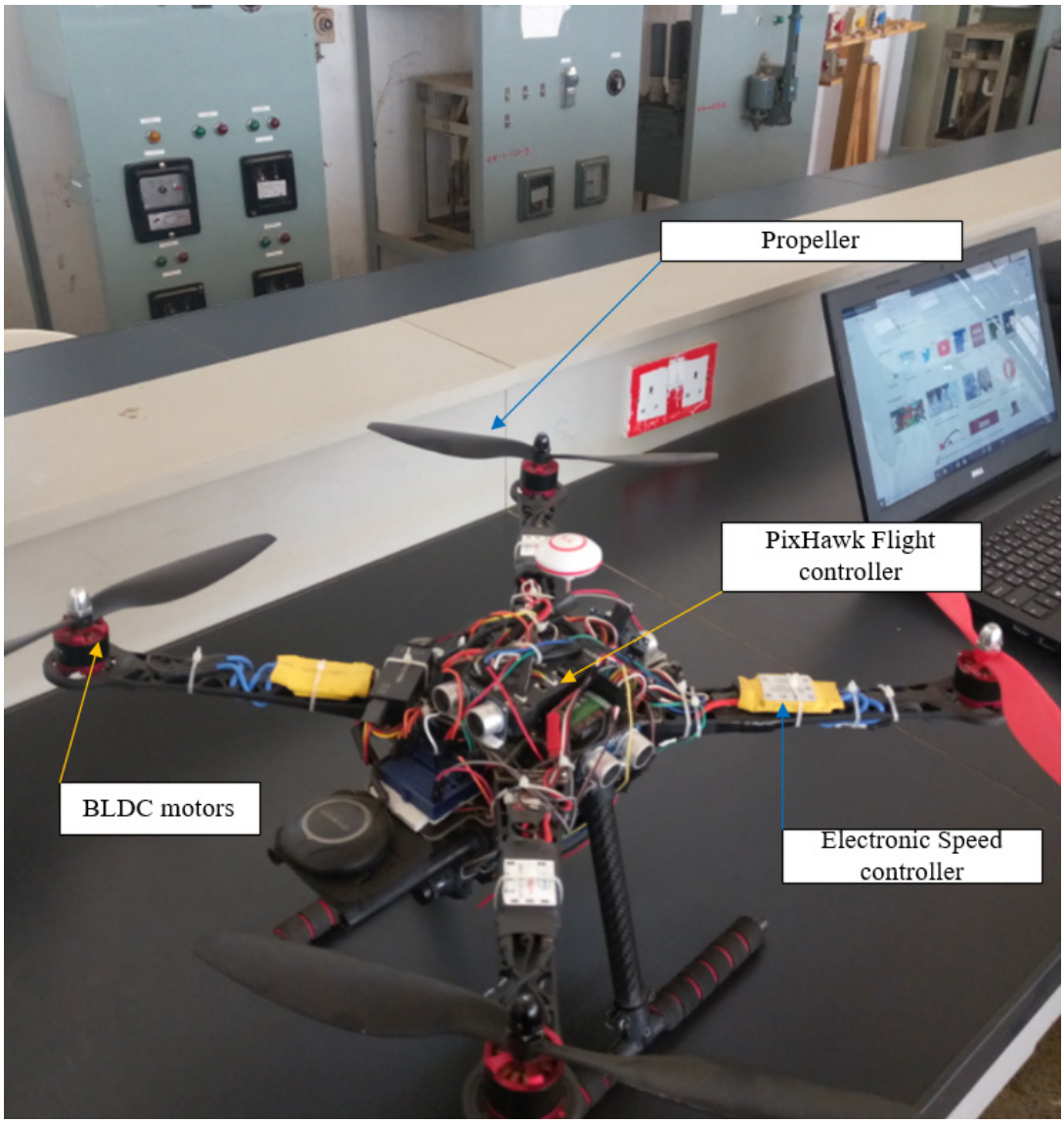

Figure 4-15: The designed Quadcopter

4.5.1 Quadcopter Flight Test Results with PID control

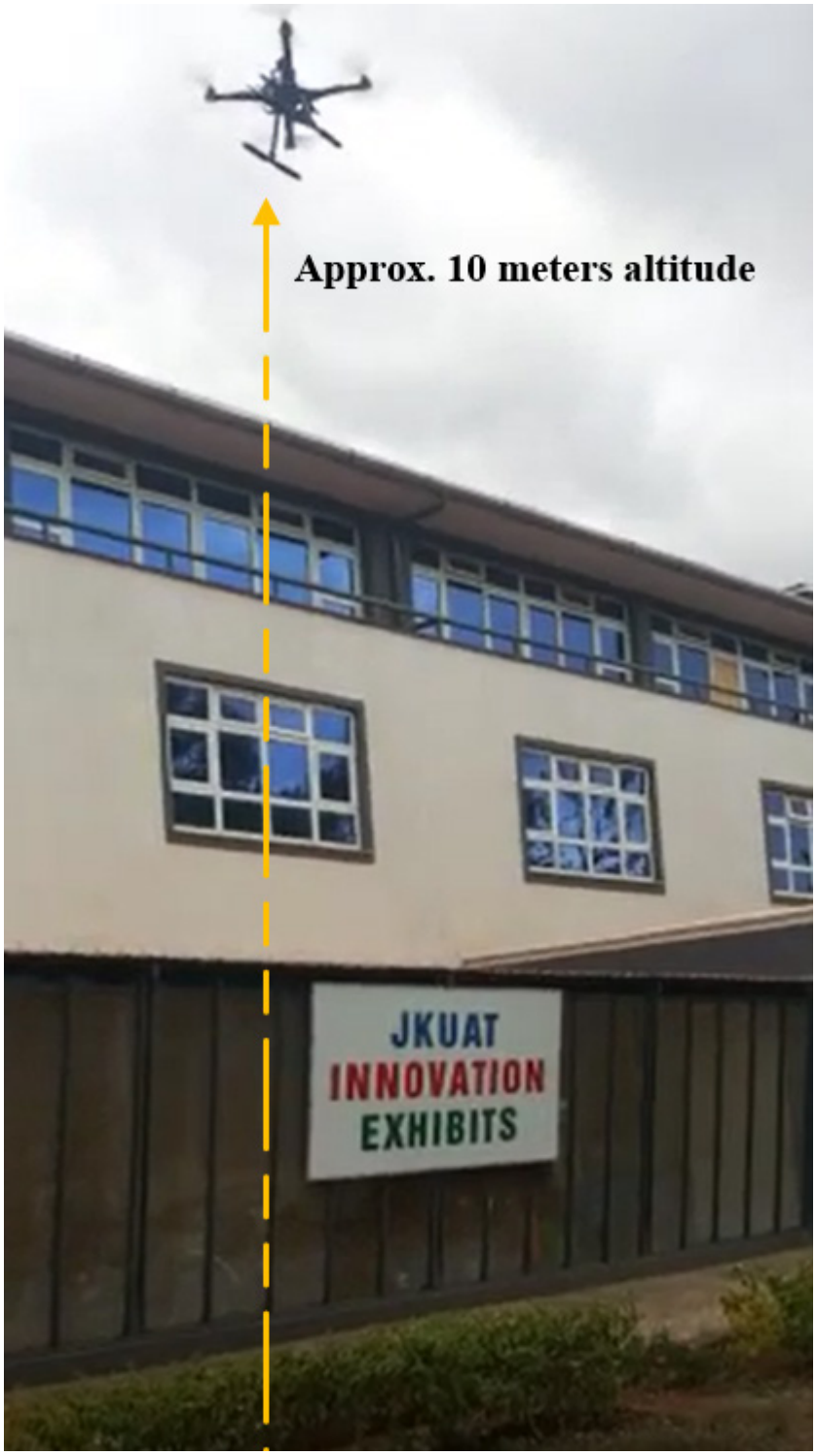

Figure 4.15: Quadcopter UAV flight with PID control algorithm

The quadcopter UAV was able to attain a height of approximately 10 meters above the ground without fault as shown in Figure 4.15. Rotor fault was introduced midflight and the quadcopter UAV lost angular control and crashed against the wall shown in Figure 4.16

The quadcopter UAV crashed after fault occurrence since the control gains are fixed and could not adjust to sustain the structural changes. From simulations in Figure 4.6 and 4.8, it can be deduced that the quadcopter becomes uncontrollable in angular control although maintains the hover position after fault. The experimental quadcopter crashes against the wall because of loss control of orientation angles.

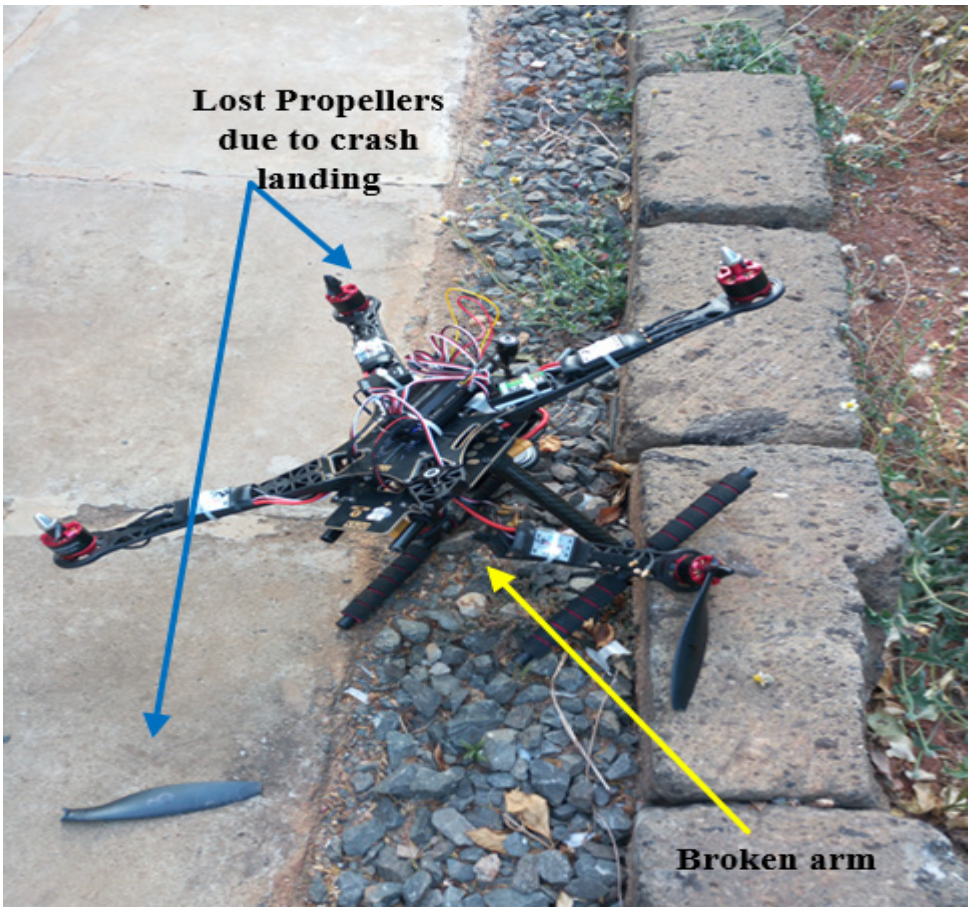

Figure 4.16: Crashed quadcopter UAV with PID control algorithm

4.5.2 Quadcopter Flight Test Results with the EKF-FPID controller

It should be noted that experimental tests for FuzzyPID were not done due extra cost of losing another quadcopter prototype. Figure 4.17 shows quadcopter UAV with the EKF-FuzzyPID controller performance without fault. It attained an altitude of about 4 meters for test flight without rotor fault as can be observed from Figure 4.17.

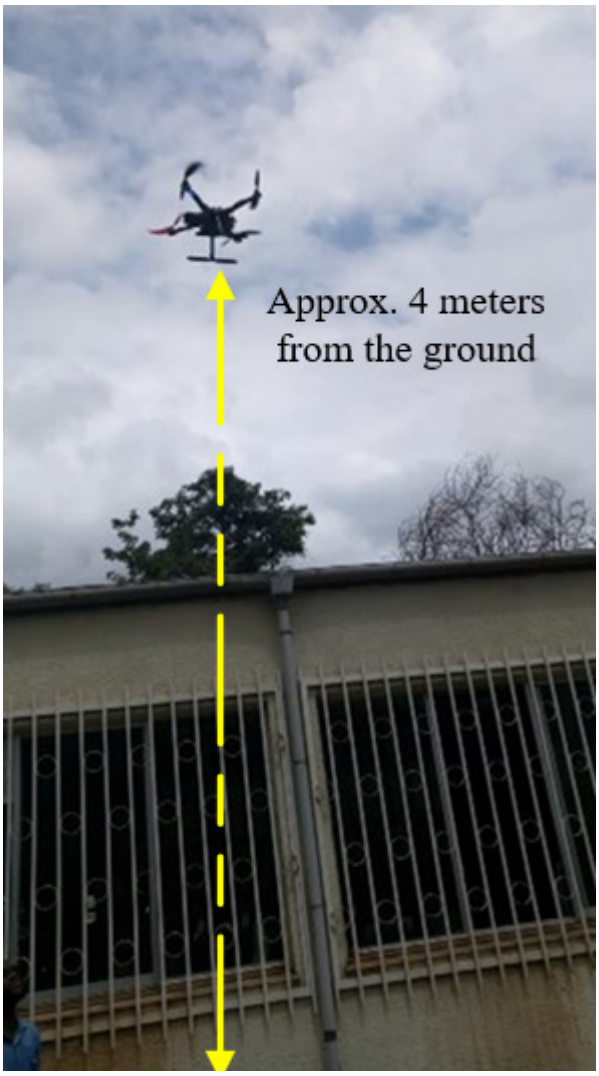

Figure 4.17: Quadcopter UAV flight with the EKF-FPID control algorithm without fault

A fault was introduced at 2meters (for observation purposes since a barometer was not used for altitude measurement. This is due to additional costs and interference of the desired payload. Additional components also overloads the battery which means

reduced flight time for better experimental tests) and the quadcopter UAV maintained that hover position (of about 1.99 meters from observation and recovered immediately in less than 2 seconds) from the ground as shown in Figure 4.18.

Figure 4.18: Flight hover under rotor fault with the proposed control algorithm

The controller was able to automatically regulate the gains thereby compensating for the structural changes. The quadcopter was able to execute a safe landing despite the rotor loss.

4.6 Discussion and Validation of Results

It should be noted that *Fault recovery time* is defined in this work as the time taken for the quadcopter system to come back to the desired altitude and maintain the desired orientation angles after a rotor fault is introduced.

The Experimental and Simulation results comparison was made as in Table 4.13. The Simulated system is the work presented in this thesis with 100% loss of effectiveness on rotor 1. A fault recovery time of 1second back to the desired height of a step input of 10m was recorded as compared to one of the best works in (Liu et al., 2016). The performance of the quadcopter prototype with the developed controller subjected to rotor failure is also compared with the simulated version. The performance of Kalman filter in estimation and filtering of orientation angles was validated by comparing its performance with that of the commonly used complimentary filter for the same set of accelerometer and gyroscopic data.

Table 4.13: Comparison of Performance for the developed controller in MATLAB/SIMULINK and in a physical quadcopter prototype

	Simulated system (100% LOE)	**Existing simulations in literature (20% LOE)**	**Quadcopter prototype**
Fault Recovery time	1 second	5 seconds	2 seconds
Lost altitude from setpoint	0.01 meters	0.08 meters	0.02 meters

PID, FPID and EKF-FuzzyPID controller performances were compared in MATLAB/SIMULINK whereas experimental tests were only performed for PID and EKF-FuzzyPID due high costs of losing another quadcopter and components in general. The experimental tests for PID was performed at approximately 10m (Similar to simulations) since it was so obvious it would crash. PID controller operated the quadcopter back to the desired height in simulation but angle control was lost, similarly PID control in experiments could not bring the quadcopter back to the desired orientation thereby crashing on a wall although at the desired altitude.

The performance of the EKF-FPID controller in simulation was done at 10m whereas experimentally it was done at 2m for timing and observation. Physically it is impossible to observe the behavior of the quadcopter at an altitude of 10m. The quadcopter was not fitted with a barometer for measurement of height variations due to avoidance of extra costs and limitations on payload and battery usage. The variations

in simulation and experimental results can be explained taking into consideration that the simulation were done under the following assumptions:

- Environmental disturbances such wind variations are ignored for MATLAB/SIMULINK simulations
- Aerodynamic, gyroscopic and Coriolis effects are neglected in Matlab Simulations
- The components used for the quadcopter implementation were chosen based on industrial standards and available literature since designing and sizing of individual components is beyond the scope of this work
- Smaller angles are used for simulations

However, the variations between simulations and experimental tests for the EKF-FuzzyPID controller were very minimal as shown in Table 4.13 above.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The objective of this thesis was to design an optimized Fuzzy PID Controller using extended Kalman filter for stabilization and safe landing of a prototype quadcopter in the event that rotor failure occured. This was verified through simulated output responses of the system to step input signals. Moreover, the experimental results observed from the quadcopter prototype showed that the developed controller and its algorithms worked.

Noise free and bias free signals for angle estimations were obtained successfully by combining the benefits of accelerometer and gyroscope through Kalman filter and validating the results by comparing the performance of the used filter and the Complimentary filter

From simulations, PID, FPID and EKF-FuzzyPID can operate the quadcopter back to the desired height, the performance of the EKF-FPID controller is significantly superior than that of the baseline controller after fault occurrence with PID having a recovery time of 9s whereas the EKF optimized FPID having a recovery time of 1s.The quadcopter dynamic system with PID control algorithm was simulated in MATLAB/Simulink for rotor loss at $t = 12$ seconds. PID controller guarantees quadcopter stability in absence of any structural disturbance. When a rotor failure is introduced, 27% of overshoot were distinctly observable in altitude response and even shifts from target angle in attitude control. Loss of height of about 1 meter was also observed when actuator fault occurred at 12^{th} second with a fault recovery time of 8 seconds. The overall settling time after fault simulation is 20%

To reduce the error, Fuzzy PID was used and the fault recovery time significantly improved to 2.5 seconds with altitude loss of 0.02 meters. Conclusively, the fuzzy PID controller showed better performance as compared to the conventional PID controller because of the fixed PID gains. Fuzzy controller was able to automatically adjust the

gains. Similarly, to further reduce the error, the quadcopter dynamic system was simulated in MATLAB/ Simulink with the EKF optimized fuzzy PID. For a rotor fault at $t = 12\ seconds$, there was a significant improvement in overall settling time of 14 seconds for EKF optimized fuzzy PID from 20seconds for the PID controller and overshoot from 27% to 18%. The fault recovery time was recorded at 1 second with an insignificant loss of altitude.

The designed control algorithm was also tested on a physical flight controller. A rotor fault resulted in negligible loss of altitude from 2 meters to a hover position of about 1.98 meters above the ground. The quadcopter UAV was able to execute a safe landing thereafter. The proposed objectives were therefore successfully achieved.

5.2 Recommendations

a) Future research in this area may focus on the optimization of the fuzzy systems using non-triangular membership functions.
b) The quadcopter performance should also be investigated for loss of rotor 2, 3 or 4.
c) Moreover, performance of a quadcopter UAV should be investigated for loss of 2 rotors.
d) In this work, test flights on the quadcopter UAV were done without any safety cage around it. This led to breakages of parts and loss of components whenever a crash was encountered. This increased the cost of implementation. It is therefore necessary that in any future work, a safety cage is designed around the quadcopter UAV to guarantee safety of components and parts.

REFERENCES

Ahn, K. K., & Truong, D. Q. (2009). Online tuning fuzzy PID controller using robust extended Kalman filter. *Journal of Process Control*, *19*(6), 1011–1023. https://doi.org/10.1016/j.jprocont.2009.01.005

Ahn, K. K., Truong, D. Q., & Soo, Y. H. (2007). Self-tuning Fuzzy PID Control for Hydraulic Load Simulator. *Proceedings of the IEEE International Conference on Control, Automation and Systems*, 345–349.

Ahn, K. K., Truong, D. Q., Thanh, T. Q., & Lee, B. R. (2008). Online self-tuning fuzzy proportional-integral-derivative control for hydraulic load simulator. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, *222*(2), 81–95. https://doi.org/10.1243/09596518JSCE484

Amoozgar, M. H., Chamseddine, A., & Zhang, Y. (2012). Fault-tolerant fuzzy gain-scheduled PID for a quadrotor helicopter testbed in the presence of actuator faults. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *2*(PART 1), 282–287. https://doi.org/10.3182/20120328-3-it-3014.00048

Anderson, B., & Moore, J. (1979). Optimal Filtering. In *Optimal Filtering*. Prentice Hall: Englewood Cliffs.

Balas, C. (2007). Modelling and Linear Control of a Quadrotor. In *Interface*. Cranfield University, Cranfield.

Ballenger, K. (2013). Unmanned Aircraft Systems: General Overview (Presentation). *American Institute of Aeronautics and Astronautics*.

Bolandi, H., Rezaei, M., Mohsenipour, R., Nemati, H., & Smailzadeh, S. M. (2013). Attitude Control of a Quadrotor with Optimized PID Controller. *Intelligent Control and Automation*, *04*(03), 335–342. https://doi.org/10.4236/ica.2013.43039

Bouabdallah, S., Noth, A., & Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro Quadrotor. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, *3*, 2451–2456. https://doi.org/10.1109/iros.2004.1389776

Bousbaine, A., Bamgbose, A., Poyi, G., & Joseph, A. (2016). Design of Self-tuning PID Controller Parameters Using Fuzzy Logic Controller for Quad-rotor Helicopter. *International Journal of Trend in Research and Development*.

Boutayeb, M., Rafaralahy, H., & Darouach, M. (1997). Convergence analysis of the extended Kalman filter used as an observer for nonlinear deterministic discrete-time systems. *IEEE Transactions on Automatic Control*, *42*(4), 581–586. https://doi.org/10.1109/9.566674

Cao, D., Qu, Q., Li, C., & He, C. (2009). Research of attitude estimation of UAV based on information fusion of complementary filter. *ICCIT 2009 - 4th International Conference on Computer Sciences and Convergence Information Technology*, 1290–1293. https://doi.org/10.1109/ICCIT.2009.61

Castillo, P., Lozano, R., & Dzul, A. (2005). Stabilization of a Mini Rotorcraft with Four Rotors. *IEEE Control Systems*, *25*(6), 45–55. https://doi.org/10.1109/MCS.2005.1550152

Cho, K., Shin, J., & Kuc, T. (2015). Design of quadrotor controller for emergency situation using Xplane. *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2015*, 311–314. https://doi.org/10.1109/URAI.2015.7358960

Clough, B. T. (2005). Unmanned aerial vehicles: Autonomous control challenges, a researcher's perspective. *Journal of Aerospace Computing, Information and Communication*, *2*(8), 327–347. https://doi.org/10.2514/1.5588

De Francesco, R., & De Francesco, E. (2015). The CoDeF structure: A way to evaluate Ai including failures caused by multiple minor degradations. *2nd IEEE International Workshop on Metrology for Aerospace, MetroAeroSpace 2015 - Proceedings*, 33–37. https://doi.org/10.1109/MetroAeroSpace.2015.7180622

De Marina, H. G., Pereda, F. J., Giron-Sierra, J. M., & Espinosa, F. (2012). UAV attitude estimation using unscented Kalman filter and TRIAD. *IEEE Transactions on Industrial Electronics*, *59*(11), 4465–4474. https://doi.org/10.1109/TIE.2011.2163913

Freddi, A., Lanzon, A., & Longhi, S. (2011a). A Feedback Linearization Approach to Fault Tolerance in Quadrotor Vehicles. *IFAC Proceedings Volumes*, *44*(1), 5413–5418. https://doi.org/10.3182/20110828-6-IT-1002.02016

Freddi, A., Lanzon, A., & Longhi, S. (2011b). A feedback linearization approach to fault tolerance in quadrotor vehicles. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *44*(1 PART 1), 5413–5418. https://doi.org/10.3182/20110828-6-IT-1002.02016

Gautam, D., & Ha, C. (2013). Control of a quadrotor using a smart self-tuning fuzzy PID controller. *International Journal of Advanced Robotic Systems*, *10*. https://doi.org/10.5772/56911

Gibiansky, A. (n.d.). *Quadcopter Dynamics*. Andrew Gibiansky. http://andrew.gibiansky.com/ physics/quadcopter-dynamics/

Girard, A. R., Howell, A. S., & Hedrick, J. K. (2004). Border patrol and surveillance missions using multiple unmanned air vehicles. *Proceedings of the IEEE Conference on Decision and Control*, *1*, 620–625. https://doi.org/10.1109/cdc.2004.1428713

Grewal, M., & Andrews, P. (1993). Kalman Filtering: Theory and Practice. In *Theory and Practice* (Vol. 5).

Haoyu Electronics. (n.d.). *GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module*. Haoyu Electronics. https://www.hotmcu.com/gy521-mpu6050-3axis-acceleration-gyroscope-6dof-module-p-83.html

Herrera, D. F. G. (2017). *Design, Development and Implementation of Intelligent Algorithms to Increase Autonomy of Quadrotor Unmanned Missions*. https://commons.erau.edu/edt/330

Higgins, W. T. (1975). A Comparison of Complementary and Kalman Filtering. *IEEE Transactions on Aerospace and Electronic Systems*, *AES-11*(3), 321–325. https://doi.org/10.1109/TAES.1975.308081

Ivan, G. (2012). Attitude stabilization of a Quad-rotor UAV based on rotor speed sensing with accelerometer data estimation via Kalman filtering. *Chinese Control Conference, CCC*, 5123–5128.

Izadi, H. A., Zhang, Y., & Gordon, B. W. (2011). Fault tolerant model predictive control of quad-rotor helicopters with actuator fault estimation. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *44*(1 PART 1), 6343–6348. https://doi.org/10.3182/20110828-6-IT-1002.03709

Jihad, G. (2013). *Fault Tolerant Flight Control Techniques with Application to a Quadrotor UAV Testbed*. Lebanese university.

Jirinec, T. (2011). Stabilization and control of unmanned quadcopter. In *Master Thesis, Czech Technical University in Prague*.

Kaba, A., Ermeydan, A., & Kiyak, E. (2017). Model derivation, attitude control and Kalman filter estimation of a quadcopter. *2017 4th International Conference on Electrical and Electronics Engineering, ICEEE 2017*, 210–214. https://doi.org/10.1109/ICEEE2.2017.7935821

Khebbache, H. (2012). Robust Stabilization of A Quadrotor Aerial Vehicle in Presence of Actuator Faults. *International Journal of Information Technology, Control and Automation*, *2*(2), 1–13. https://doi.org/10.5121/ijitca.2012.2201

Kivrak, A. (2006). Design of Control Systems for a Quadrotor. In *Thesis* (Issue December).

Kobayashi, K., Cheok, K. C., & Watanabe, K. (1995a). Estimation of absolute vehicle speed using fuzzy logic rule-based Kalman filter. *Proceedings of the American Control Conference*, *5*, 3086–3090. https://doi.org/10.1109/acc.1995.532084

Kobayashi, K., Cheok, K. C., & Watanabe, K. (1995b). Estimation of absolute vehicle speed using fuzzy logic rule-based Kalman filter. *Proceedings of the American Control Conference*, *5*, 3086–3090. https://doi.org/10.1109/acc.1995.532084

Kubelka, V., & Reinstein, M. (2012). Complementary filtering approach to orientation estimation using inertial sensors only. *Proceedings - IEEE International Conference on Robotics and Automation*, 599–605. https://doi.org/10.1109/ICRA.2012.6224564

Lan, T., Fei, Q., Geng, Q., & Hu, Q. (2012). Control algorithm research of quadrotor system based on 3-DOF simulation platform. *IET Conference Publications*, *2012*(598 CP), 697–700. https://doi.org/10.1049/cp.2012.1073

Lee, C.-C. (1990). Fuzzy Logic in control systems: fuzzy logic controller-parts 1 and 2. *IEEE Transactions on Systems, Man, and Cybernetics*, *20*(2), 404–435.

Li, T., Zhang, Y., & Gordon, B. W. (2013). Passive and active nonlinear fault-tolerant control of a quadrotor unmanned aerial vehicle based on the sliding mode control technique. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, *227*(1), 12–23. https://doi.org/10.1177/0959651812455293

Lippiello, L., & Ruggiero, F. (2014). Emergency Landing for a Quadrotor in case of Propeller Fault: A PID approach. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Liu, Z., Yuan, C., Zhang, Y., & Luo, J. (2016). A Learning-Based Fault Tolerant Tracking Control of an Unmanned Quadrotor Helicopter. *Journal of Intelligent & Robotic Systems*, *84*(1–4), 145–162. https://doi.org/10.1007/s10846-015-0293-0

Maciejowski, J. M., & Jones, C. N. (2003). MPC fault-tolerant flight control case study: Flight 1862. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *36*(5), 119–124. https://doi.org/10.1016/S1474-6670(17)36480-7

Mueller, M. W., & D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 45–52. https://doi.org/10.1109/ICRA.2014.6906588

Numan, M. (2017). *Controller Design for Attitude and Position Control of Quadrotor*. Capital University of Science and Technology, Pakistan.

Ofosu, R. A., Kaberere, K. K., Nderu, J. N., & Kamau, S. I. (2016). Design of BFA-optimized fuzzy electronic load controller for micro hydro power plants. In *Energy for Sustainable Development* (Vol. 51). JKUAT.

Paggi, R., Mariotti, G. L., Paggi, A., & Leccese, F. (2015). Optimization of availability operation via simulated prognostics. *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, 44–48. https://doi.org/10.1109/MetroAeroSpace.2015.7180624

Pure Energy Solutions Inc. (2008). *Battery Types: Which Batteries to Use? 1*, 1–5. http://www.pureenergybattery.com/pdf/batterytypes.pdf

Ranjbaran, M., & Khorasani, K. (2010). Fault recovery of an under-actuated quadrotor aerial vehicle. *Proceedings of the IEEE Conference on Decision and Control*, 4385–4392. https://doi.org/10.1109/CDC.2010.5718140

Ranjbaran, M., & Khorasani, K. (2012). Generalized fault recovery of an under-actuated quadrotor aerial vehicle. *2012 American Control Conference (ACC)*, 2515–2520. https://doi.org/10.1109/ACC.2012.6315526

Raza, S. A., & Gueiaeb, W. (2010). Intelligent Flight Control of an Autonomous Quadrotor. *Motion Control*, *1*, 245–264. http://www.intechopen.com/books/motion-control

Razinkova, A., Gaponov, I., & Cho, H. C. (2014). Adaptive control over quadcopter UAV under disturbances. *International Conference on Control, Automation and Systems*, 386–390. https://doi.org/10.1109/ICCAS.2014.6988027

Rich, M., & Elia, N. (2012). *Model development, system identification, and control of a quadrotor helicopter*. Iowa State University Ames.

Sadeghzadeh, I., Mehta, A., Chamseddine, A., & Youmin Zhang. (2012). Active Fault Tolerant Control of a quadrotor UAV based on gainscheduled PID control. *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–4. https://doi.org/10.1109/CCECE.2012.6335037

Sadeghzadeh, I., Mehta, A., Zhang, Y., & Rabbath, C. A. (2014). Fault-tolerant trajectory tracking control of a quadrotor helicopter using gain-scheduled PID and model reference adaptive control. *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2011, PHM 2011*, 247–256.

Sanjeev, A. (2018). *How to Interface Arduino and the MPU 6050 Sensor*. Maker Pro. https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor

Schmidt, J., & Parker, R. (1995). Development of A UAV Mishap Human Factors Database. *Unmanned Systems 1995 Proceedings*.

Selby, W. (n.d.). *Model Verification*. Wil Selby. https://www.wilselby.com/research/arducopter/model-verification/

Sharifi, F., Mirzaei, M., Gordon, B. W., & Zhang, Y. (2010). Fault tolerant control of a quadrotor UAV using sliding mode control. *Conference on Control and Fault-Tolerant Systems, SysTol'10 - Final Program and Book of Abstracts*, 239–244. https://doi.org/10.1109/SYSTOL.2010.5675979

Sidi, M. J. (1997). Spacecraft dynamics and control: A practical engineering approach. In *Spacecraft Dynamics and Control: A Practical Engineering Approach*. Cambridge University Press. https://doi.org/10.1017/CBO9780511815652

Simon, D. (2002). Training fuzzy systems with the extended Kalman filter. *Fuzzy Sets and Systems*, *132*(2), 189–199. https://doi.org/10.1016/S0165-0114(01)00241-X

Sivanandam, S. N., Sumathi, S., & Deepa, S. N. (2007). Introduction to fuzzy logic using MATLAB. In *Introduction to Fuzzy Logic using MATLAB*. https://doi.org/10.1007/978-3-540-35781-0

Sparkfun. (n.d.). *IMU*. Sparkfun. https://www.sparkfun.com/

Starlino. (n.d.). *A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications.* Starlino Electronics. http://www.starlino.com/imu_guide.html

Tamilselvan, G. M., & Aarthy, P. (2017). Online tuning of fuzzy logic controller using Kalman algorithm for conical tank system. *Journal of Applied Research and Technology*, *15*(5), 492–503. https://doi.org/10.1016/j.jart.2017.05.004

Truong, D. Q., Ahn, K. K., Soo, K. J., & Soo, Y. H. (2007). Application of fuzzy-PID controller in hydraulic load simulator. *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, ICMA 2007*, 3338–3343. https://doi.org/10.1109/ICMA.2007.4304098

Vasconcelos, J. F., Silvestre, C., Oliveira, P., Batista, P., & Cardeira, B. (2009). Discrete time-varying attitude complementary filter. *Proceedings of the American Control Conference*, 4056–4061. https://doi.org/10.1109/ACC.2009.5159879

Wang, L.-X. (2003). *A course in fuzzy systems*. Tsinghua Press. http://books.google.com.br/books?id=wbJQAAAAMAAJ%5Cnhttp://books.google.com.br/books/about/A_Course_in_Fuzzy_Systems_and_Control.html?id=wbJQAAAAMAAJ&pgis=1

Wang, L., & Beumer, T. (2016). *Fault Tolerant Control of a Quadrotor Unmanned Aerial Vehicles* (Issue March) [RMIT University, Eindhoven]. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Lyapunov-based+fault+tolerant+control+of+quadrotor+unmanned+aerial+vehicles#1

Wang, L., & Chen, X. (2016). *PID control of quadrotor*.

Wang, P., Man, Z., Cao, Z., Zheng, J., & Zhao, Y. (2016). Dynamics modelling and linear control of quadcopter. *International Conference on Advanced Mechatronic Systems, ICAMechS*, 498–503. https://doi.org/10.1109/ICAMechS.2016.7813499

Wil Selby. (n.d.). *Model Verification*. Wil Selby. https://www.wilselby.com/research/arducopter/model-verification/

Wu, X., & Liu, Y. (2018). Trajectory tracking control of quadrotor UAV. *Chinese Control Conference, CCC*, *2018-July*, 10020–10025. https://doi.org/10.23919/ChiCC.2018.8482939

Zhang, Y. (2011). Concordia University Fault-Tolerant Trajectory Tracking Control of a Quadrotor Helicopter Using Gain-Scheduled PID and Model Reference Adaptive. *Annual Conference of the Prognostics and Health Management Society*.

Zulu, A., & John, S. (2014). A Review of Control Algorithms for Autonomous Quadrotors. *Open Journal of Applied Sciences*, *04*(14), 547–556. https://doi.org/10.4236/ojapps.2014.414053

APPENDICES

Appendix I: Implementation of the Kalman Filter

The equations in 3.2.3 was implemented into a simple C code as follows:

Step 1:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + B\dot{\theta}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k$$

$$= \begin{bmatrix} \theta - \dot{\theta}_b \Delta t \\ \dot{\theta}_b \end{bmatrix}_{k-1|k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \dot{\theta}_k$$

$$= \begin{bmatrix} \theta - \dot{\theta}_b \Delta t + \dot{\theta} \Delta t \\ \dot{\theta}_b \end{bmatrix}$$

$$= \begin{bmatrix} \theta + \Delta t(\dot{\theta}_b - \dot{\theta}) \\ \dot{\theta}_b \end{bmatrix}$$

In C, this is written as:

$$rate = newRate - bias;$$
$$angle += dt * rate;$$

Step 2:

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_k$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} & P_{01} - \Delta t P_{11} \\ P_{10} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} \end{bmatrix}_{k-1|k-1} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t$$

$$= \begin{bmatrix} P_{00} - \Delta t P_{10} - \Delta t(P_{01} - \Delta t P_{11}) + Q_\theta \Delta t & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}$$

$$= \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{10} - P_{01} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b} \Delta t \end{bmatrix}$$

These equations are implemented into C as:

```
P[0][0]+ = dt * (dt * P[1][1] - P[0][1] - P[1][0] + Q_angle);
P[0][1]- = dt * P[1][1];
P[1][0]- = dt * P[1][1];
P[1][1]+ = Q_gyroBias * dt;
```

Step 3:

$$\tilde{y}_k = z_k - H\hat{x}_{k|k-1}$$

$$= \mathrm{z_k} - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1}$$

Implemented as:

```
y = newAngle - angle;
```

Step 4:

$$
\begin{aligned}
S_k &= HP_{k|k-1}H^T + R \\
&= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + R \\
&= P_{00k|k-1} + R \\
&= P_{00k|k-1} + \text{var}(v)
\end{aligned}
$$

Implemented as:

```
S = P[0][0] + R_measure;
```

Step 5:

$$K_k = P_{k|k-1}H^T S_k^{-1}$$

$$
\begin{aligned}
\begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k &= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} S_k^{-1} \\
&= \begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1} S_k^{-1} \\
&= \frac{\begin{bmatrix} P_{00} \\ P_{10} \end{bmatrix}_{k|k-1}}{S_k}
\end{aligned}
$$

Which is implemented in C as:

```
K[0] = P[0]P[0] / S;
K[1] = P[1]P[0] / S;
```

Step 6:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k} = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} \tilde{y}_k$$
$$= \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \tilde{y} \\ K_1 \tilde{y} \end{bmatrix}_k$$

Implemented as:

```
angle += K[0] * y;
bias  += K[1] * y;
```

Step 7:

$$P_{k|k} = (I - K_k H) P_{k|k-1}$$

$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$
$$= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ K_1 & 0 \end{bmatrix}_k \right) \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1}$$
$$= \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix}$$

This is implemented as:

```
float P00_temp = P[0][0];
float P01_temp = P[0][1];
P[0][0] -= K[0] * P00_temp;
P[0][1] -= K[0] * P01_temp;
P[1][0] -= K[1] * P00_temp;
P[1][1] -= K[1] * P01_temp;
```

From literature, the following variances work well with most Inertial Measurement Units.

```
float Q_angle     = 0.001;
float  Q_gyroBias = 0.003;
float R_measure   = 0.03;
```

Appendix II: List of Journal Publications and Conference papers

Oloo, J., Kamau, S., & Kang'ethe, S. (2019). Fuzzy PID Control of a Quadcopter Altitude, Roll and Pitch in the Event of Rotor loss. *European Journal of Advances in Engineering and Technology*, *6*(1).

Oloo, J. O., & Kamau, S. I. (2018a). *Quadcopter Attitude Estimation using filters for Sensor Fusion in 6D Inertial Measurement Unit*. *1*, 2–5.

Oloo, J. O., & Kamau, S. I. (2018b). *Quadcopter Control Algorithms in the event of Loss of One of the Actuators : A Review*. 45–53.