People's Democratic Republic of Algeria Ministry of Higher Education and Scientific Research

University M'hamed Bougara - Boumerdes

Institute of Electrical and Electronic Engineering

Project Report Presented in Partial Fulfillment of the Requirements of the Degree of:

'Master'

In Computer Engineering

Control and Implementation of Quadrotor using a Raspberry PI as Ground Station

Presented by :

Nesrine ATTOUCHE

Malak BELKACEMI

Supervisor :

Dr.H. BELAIDI

Co-Supervisor :

Pr.A. NEMRA (EMP)

Promotion: 2022/2023

Abstract

As the development and utilization of Unmanned Aerial Vehicles (UAVs) continue to grow, it becomes increasingly important to address the challenges related to their control and performance. This master's project aims to build a stable quadrotor's flight controller using the ESP32 microcontroller, enabling wireless guidance through a Graphical User Interface (GUI) on Raspberry Pi. In this regard, this project provides quadrotor modeling and simulating the Backstepping controller (BSC) approach with various simulation scenarios. Additionally, two control approaches were practically implemented . The first approach utilizes the classical Proportional Integral Derivative (PID) technique, while the second approach employs the modern Backstepping technique. The results obtained from the project demonstrate that both control approaches exhibit satisfactory performance and responsiveness in terms of stailizing the quadrotor when implemented in real-world scenarios. However, it has been noticed that the Backstepping Controller can achieve higher performance, whereas, PID performance is limited by tuning and model.

Keywords : UAVs, Quadrotor, Backstepping controllers, ESP32 microcontroller, Raspberry Pi, GUI, Attitude controllers, PID, Implementation, prototype, wireless guidance.

Dedication

“

We dedicate this project to our loving family, whose unwavering support, encouragement, and sacrifices have been the foundation of our journey. Their constant motivation has propelled us forward. This project is a testament to their love and belief in us,

We gratefully dedicate this work to our advisors and mentors, whose guidance has profoundly influenced our intellectual growth. We deeply appreciate their commitment to our academic and personal development.,

To our friends and colleagues, thank you for being our source of motivation, that made this academic endeavor enjoyable.

Lastly, we dedicate this achievement to all those who strive to advance knowledge and make a positive impact in their respective fields.

”

Acknowledgement

First and foremost, we thank Allah the Almighty for granting us the courage and patience necessary to complete this work.

We would like to express our heartfelt gratitude to our supervisor, **Dr.H.BELAIDI**, for her competent assistance, patience, and encouragement. Her critical eye has been invaluable in structuring the work and improving the quality of the various sections.

We would also like to thank our mentor, **Pr.A.NEMRA**, for his immense help, and quality guidance, and for providing us with invaluable advice and information with unmatched patience and professionalism.

We would also like to express our heartfelt gratitude to the **Military Polytechnic School EMP (Ex-ENITA)** where we completed our internship. Our deepest thanks go to the all school members who supported us during our time there.

We would like to thank **Mr. A.SAIBI**, for their valuable assistance, and encouragement.

To the members of the jury, we sincerely thank you for the honor of taking the time to read and evaluate this work.

We would also like to thank the pedagogical and administrative teams of **IGEE** for their efforts in providing us with excellent education.

Finally, we would like to thank everyone who has contributed directly or indirectly to the completion of this work.

Contents

List of Figures

List of Tables

List of Abbreviations

6DOF	Six-degrees of Freedom
BLDC	Brush-Less Direct Current
BSC	Back-Stepping Controller
ESC	Electronic Speed Controller
ESP	Espressif System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
I^2C	Inter-integrated circuit
IMU	Inertial Measurement Unit
IP	Internet Protocol
LiPo	Lithium Polymer
PCB	Printed Circuit Board
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
SoC	System on Chip
UAV	Unmanned Areal Vehicle
VTOL	Vertical Take Off and Landing

List of Symbols

$\boldsymbol{F}$	Net force
$\boldsymbol{I}$	Quadrotor's inertia matrix
$\boldsymbol{K_f}$	Aerodynamic force constant
$\boldsymbol{K_m}$	Aerodynamic moment constant
$\boldsymbol{M_B}$	All the moments acting on the quadrotor in the body frame
$\boldsymbol{M_G}$	Gyroscopic moment
$\boldsymbol{\Omega_i}$	Angular velocity of rotor i
$\boldsymbol{R}$	Rotation matrix
$\boldsymbol{R_r}$	Transformation matrix
$\boldsymbol{m}$	Quadrotor's mass
$\boldsymbol{r_e}$	Linear position
$\boldsymbol{v_b}$	Linear velocity
$\boldsymbol{w_b}$	Angular velocity
$\boldsymbol{y_e}$	Angular position
$\boldsymbol{\varphi}$	Roll angle
$\boldsymbol{\psi}$	Yaw angle
$\boldsymbol{\theta}$	Pitch angle

General Introduction

Context

Unmanned Aerial Vehicles (UAVs), popularly known as drones, have become very popular in recent years. This remote control has caught the attention of people, businesses, and governments around the world. Quadrotors, especially, have attracted attention with their agility, stability, and maneuverability. The quadrotor design allows vertical take-off and landing (VTOL) operations, making the suitable for confined spaces and rapid deployment. A quadrotor can move through space, fly in any direction and perform precise maneuvers, making it versatile for a variety of applications.

Motivation

The fast improvement and increasing popularity of quadrotor systems have led to various advancements in their design, control, and applications. However, there are still challenges and unresolved troubles that should to be addressed to fully exploit the potential of quadrotors.

One of the primary challenges is accomplishing stable flight and precise maneuverability. Quadrotors are highly dynamic systems that require sophisticated control algorithms to maintain stability. Thus, developing robust and efficient control strategies that can handle uncertainties, disturbances, and varying operating conditions is crucial.

Objectives

The goal of our project is to construct a stable quadrotor based on the ESP32 as a flight controller and guided wirelessly via a Graphical User Interface (GUI) on Raspberry PI, and develop a dynamic model used for two control approaches. First, a classical approach by Proportional Integral Derivative (PID), then a modern approach by the Backstepping technique.

Report Organization

This report is organized into four chapters:

The first chapter, **Overview of UAVs**, provides an overview of UAVs, starting with a brief history and discussing their applications. It explores the principles of flight that govern quadrotor systems, highlighting their challenges. This chapter serves as a foundation for understanding the fundamental concepts and terminology related to UAVs and quadrotors.

The second chapter, **Modeling a Quadrotor**, delves into the modeling of quadrotors. It presents the mathematical models and equations that describe the dynamics and behavior of quadrotor systems. Various factors such as aerodynamics and kinematics are considered in developing accurate and reliable models. This chapter provides a comprehensive understanding of the quadrotor's kinematics, dynamics characteristics, and behavior.

The third chapter, **System Control**, focuses on simulations using MATLAB Simulink to analyze and test the quadrotor system before actual implementation. This chapter discusses the simulation results to validate the quadrotor model and evaluate its performance. It covers topics such as controller design using Backstepping Controller and stability analysis through simulation experiments.

The fourth chapter, **Building and Testing a Quadrotor Prototype**, is dedicated to the implementation of the quadrotor using the ESP32 microcontroller as a Flight controller and Raspberry Pi as the guidance card. It explores the hardware and software required for building a functional quadrotor system. It also addresses the challenges and considerations in real-world implementation.

Chapter 1

Overview of UAVs

UAVs (Unmanned Aerial Vehicles), also known as Drones, are aerial vehicles that do not require a human operator on a board. They can be either piloted remotely or fly autonomously using pre-programmed flight plans [1]. In this chapter, we will discuss the historical development of UAVs, their different categories, and their configurations. We will also examine the various areas of application, as well as their advantages and limitations.

1.1 Brief History of UAVs

The media has recently brought attention to civil drones, leading many to believe that they are a new technology. However, drones have been around for almost a century, with the key difference being that today's drones are smaller, less expensive, and more readily accessible.

- In 1849, the earliest unmanned aerial vehicle in the history of drones was seen when Austrian soldiers attacked the city of Venice in world war II with unmanned balloons filled with explosives [2].

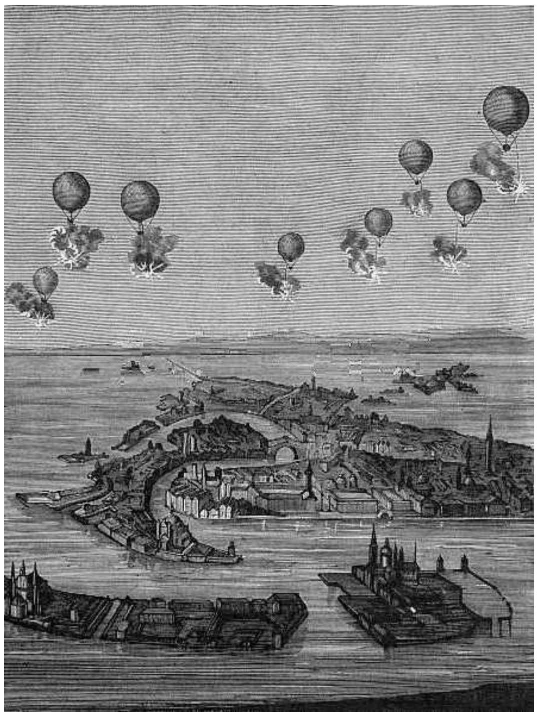


Figure 1.1: Aerial bombardment of Venice in 1849[3] .

- In 1917, “Charles Kettering” invented the unmanned “Kettering Aerial Torpedo”, commonly known as the “Bug” in Ohio [4]. When the Bug reached a pre-determined distance, the engine would stop, the wings would detach, and the Bug would fall from the sky.

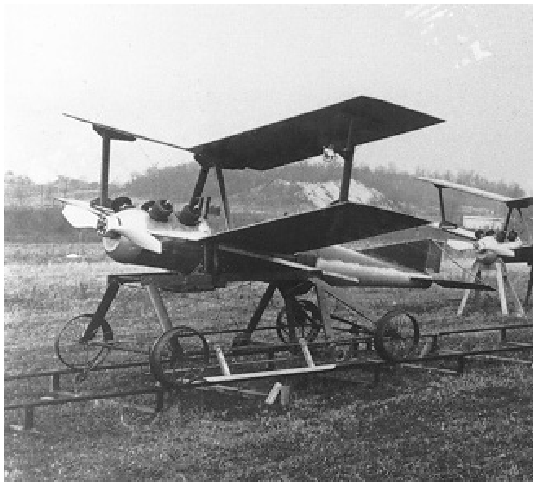


Figure 1.2: Kettering Bug [5].

- In 1935, the “De Havilland DH.82B Queen Bee” aircraft was used as a low-cost radio-controlled drone developed for aerial target practice. It is considered by many to be the first modern drone[4].

Figure 1.3: Queen Bee seaplane with Prime Minister Winston Churchill [6].

- In 1943, “Boeing” and the “U.S. Airforce” developed the “BQ-7”, which operated on a crude FPV (First-Person View) system. A human pilot would fly the aircraft toward the target. Once the target was in view, the autopilot was engaged, and the pilot bailed out of the plane. The “BQ-7” would then fly to the target on its own.

- In 2010, The French drone manufacturer “Parrot” unveiled its “AR Drone”, the first of its kind to be controllable directly from a Smartphone.
- In 2013, Amazon CEO Jeff Bezos announced that the company was considering using drones as a delivery method. The iconic DJI Phantom 1 is released and began the modern drone craze.
- In 2016, a Chinese Tech Company called “Ehang” develops the first fully functional Autonomous Passenger Drone Car with a futuristic design.
- In 2020-2021, Drones have been a staple during the coronavirus outbreak, helping with medical supply deliveries, police work in social distancing, and quarantine [7].

1.2 Applications of the UAVs

a) Firefighting: Forest fire management is the most advanced and extensively demonstrated UAV application. The earliest detection plays a crucial role in mitigating the damages caused by fires and enables prompt response measures to be taken.

b) Security and Surveillance: The concept of surveillance drones arises to aid the problem of tracking humans quickly without risking more human lives in disasters or terrorist attacks.

c) Aerial Photography and Video: Drones are transforming film and television, news, and real estate photography. It provides numerous opportunities to capture high-quality images and videos from angles that were previously impossible [8].

d) Surveying and Mapping: Topographic maps are essential for planning large-scale and complex construction projects, but their production is often expensive and time-consuming. The use of drones is highly effective in such cases as they can capture large amounts of data in a relatively short time, resulting in significant cost [9].

e) Agriculture: Farmers and agriculturists are always looking for cheap and effective methods to regularly monitor their crops. The infrared sensors in drones can be tuned to detect crop health, enabling farmers to react and improve crop conditions locally, with inputs of fertilizer or insecticides. It also improves management and effectuates better yield of the crops [10].

f) Military: UAVs in military applications provide critical capabilities, such as surveillance, reconnaissance, target acquisition, combat support, offensive operations, force protection, and search and rescue.

Figure 1.4: Applications of UAVs [11].

1.3 UAVs Limitations

The use of drones offered advantages on so many levels, from commercial to personal. However, drone systems suffer from different security, safety, and privacy issues. The breaches of security and privacy led by drones should be addressed at the highest national level. Moreover, there should exist a very strict approach to limit the drone's ability to gather images and record videos of people and properties without authorized permission [12].

1.4 Classification of UAVs

There are different ways to classify UAVs, either according to their range of action, Aerodynamic configuration, size, and payload or according to their levels of autonomy...etc

1.4.1 Range of Action Classification

UAVs can be classified into 7 different categories based on their maximum altitude, endurance, and sizes ranging from larger drones to micro drones as follows:

a) High-Altitude Long-Endurance (HALE): These drones are capable of flying at high altitudes (above 60,000 feet) for extended periods (over 24 hours) and are typically used for surveillance and reconnaissance missions [13].

b) Medium-Altitude Long-Endurance (MALE): These drones are similar to HALE UAVs but can operate at lower altitudes (between 10,000 and 45,000 feet) for shorter periods (around 24 hours). They are also used for surveillance and reconnaissance missions[14].

c) Medium-Range or Tactical UAV (TUAV): These drones are used for military operations and can operate at medium to low altitudes (between 50 and 18,000 feet) for shorter periods (up to 12 hours). They are designed for reconnaissance, surveillance, and target acquisition.

d) Close-Range UAVs: These drones are lightweight and typically used for civilian and commercial applications such as photography, mapping, and inspections. They can fly at low to medium altitudes (between 50 and 500 feet) for short periods (up to 1 hour).

e) Mini UAV (MUAV): These can carry out their missions with a low endurance (30mn) within a range of up to 10km. They are intended primarily for intelligence gathering in close combat. Their vertical take-off and landing capability makes their application considerable in congested environments.

f) Micro UAV (MAV): These are the smallest drones and can be used for indoor and outdoor operations. They are typically used for close-range surveillance, inspection, and reconnaissance missions.

g) Nano-Air Vehicles (NAV): they have a small size of about 10 mm. They are mainly used in swarms for applications such as radar confusion in hard-to-reach places [15].

1.4.2 Aerodynamic Configuration Classification

UAVs can be classified into two main categories based on their aerodynamic configuration as follows:

a) Fixed-wing UAVs: require a runway to take off and land. They can fly for a long time and at high cruising speeds. They are mainly used in scientific applications such as meteorological reconnaissance and environmental monitoring.

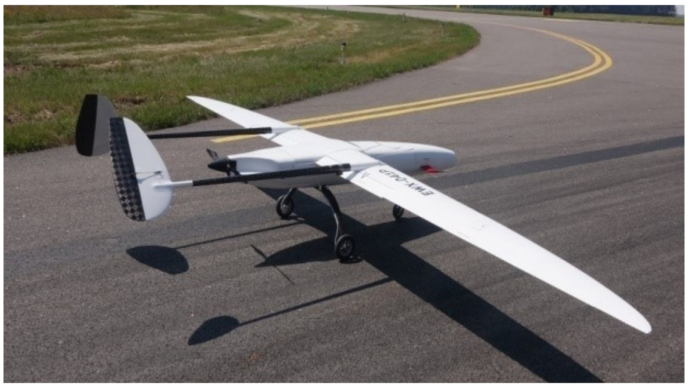


Figure 1.5: Fixed wing UAV [16].

b) Rotary-wing UAVs: an unmanned aerial vehicle that utilizes rotating blades or rotors to generate lift and control their flight. Unlike fixed-wing UAVs, rotary-wing UAVs can achieve vertical takeoff and landing (VTOL) and hover in a stationary position.

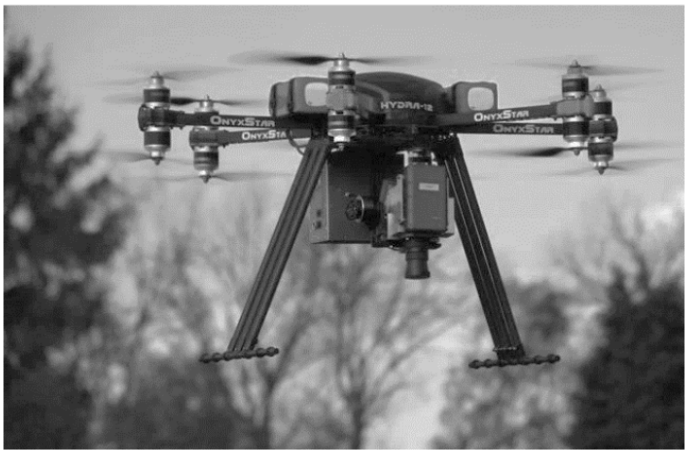


Figure 1.6: Multirotor UAV [17].

1.5 Quadrotor

In our project, we are particularly interested in Quadrotors which have been increasingly used as a preferred unmanned aerial vehicle (UAV) platform for indoor applications due to their ease of maintenance, high maneuverability, vertical takeoff, and landing capabilities.

A Quadrotor, also known as a quadcopter, is a type of multirotor that is powered by four rotors, each rotor consists of a motor and a propeller that generate vertical lift and control the vehicle's movement.

Figure 1.7: Quadrotor UAV [18] .

1.5.1 Quadrotor's Movements

Quadrotors consist of four actuators that are individually controlled to produce a relative thrust. Two of the rotors rotate clockwise, and the other two rotors rotate in a counter-clockwise direction.

By varying the speed and direction of rotation of these rotors, the quadrotor can generate thrust in different directions, enabling it to move in various ways. The quadrotor's movement can be described in four basic movements:

a) Throttle Movement :

This command is provided by increasing (or decreasing) all the propeller speeds by the same amount. It leads to a vertical force that raises or lowers the quadrotor.

b) Roll Movement (ϕ) :

This command is provided by increasing (or decreasing) the left propeller speed and by decreasing (or increasing) the right one. The motion in the 'x' simply describes the quadrotor moving sideways.

c) Pitch Movement (θ) :

This command is very similar to the roll and is provided by increasing (or decreasing) the rear propeller speed and by decreasing (or increasing) the front one. The motion in the 'y' describes the quadrotor moving forward/backward.

d) Yaw Movement (ψ) :

This command is provided by increasing (or decreasing) the front-rear propellers' speed and by decreasing (or increasing) that of the left-right couple. The motion in the 'z' direction describes the quadrotor turning left/right.

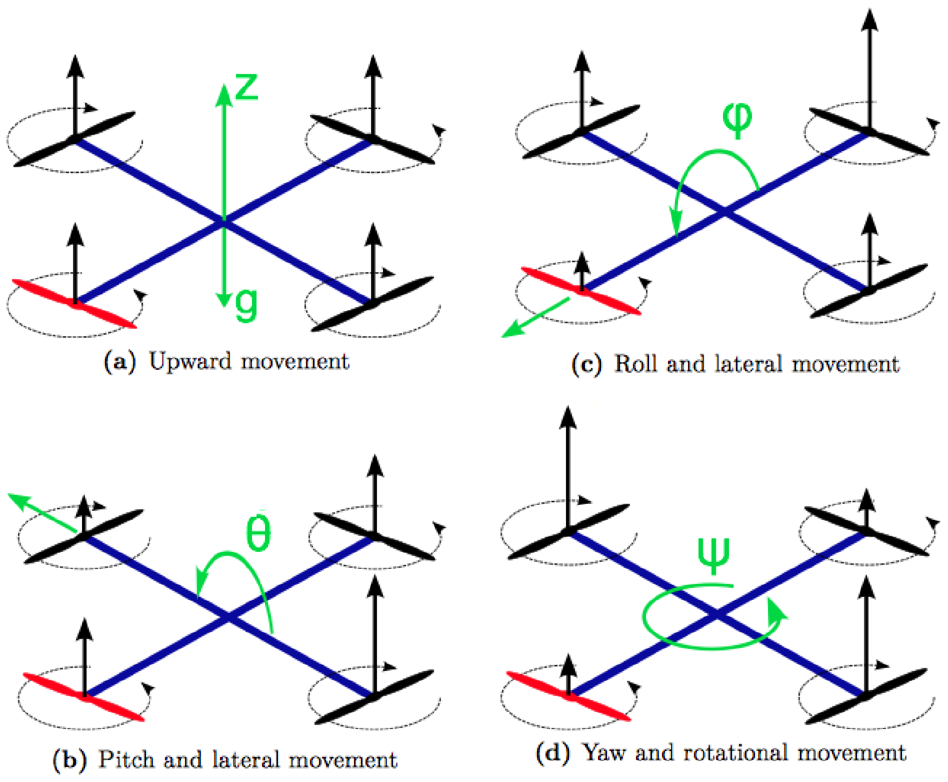

z
g
(a) Upward movement
φ
(c) Roll and lateral movement
θ
(b) Pitch and lateral movement
ψ
(d) Yaw and rotational movement

Figure 1.8: Quadrotor Mouvements [19] .

1.5.2 Dealing with Six-Degrees of Freedom (6 DOF)

Controlling a quadrotor can be challenging due to its six degrees of freedom and the fact that it relies on only four independent rotor commands, which means there is an inherent coupling between the different movements. This coupling creates complexities in controlling the quadrotor precisely. When a command is given to change one specific movement, it can affect other movements as well.

1.6 Conclusion

Drone technology is an important part of the future and is set to become a big commercial industry and an exciting field in the world of aviation. Drones have the potential to become a vital part of society. Quadrotors are the most stable and easy to be designed and implemented kind of drone. Thus, the mathematical and dynamic modeling of the quadrotor is discussed in the next chapter.

Chapter 2

Modeling a Quadroter

In order to design a flight controller, we must understand the different motions of a quadrotor and its dynamic equations to ensure that simulations of quadrotor behavior are closer than possible to reality.

In this chapter, we focus on mathematically modeling the dynamics of a quadrotor using Newton's equations of motion. We start by introducing the reference points that are needed to express the orientation and position of the quadrotor, as well as the physical quantities required for modeling. Finally, we examine all the forces and moments that affect the quadrotor model.

2.1 Kinematic and Dynamic Modeling

A mathematical model has been developed by using the kinematics and dynamics of a quadrotor.

Before jumping into the mathematical model equations, certain assumptions should be defined in order to go further in the theory:

- Quadrotor has a rigid body with no moving parts except rotor blades;
- Quadrotor structure is symmetric about the x and y-axis;
- Rotors are placed at an equal distance from the center of mass;
- Fixed propellers have been used;
- The drag forces and drag moments are neglected ;

Another important aspect of the mathematical model is the coordinate system used which will vary whether a plus "+" or "X" configuration is presented. In our modeling, we opt for the "X" configuration as illustrated in figure 2.1.

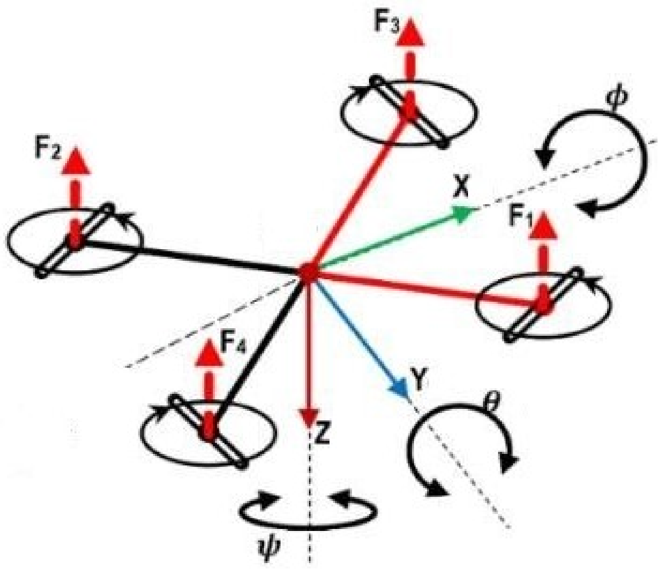


Figure 2.1: Quadrotor Configuration [20].

2.1.1 Quadrotor Kinematic Model

To describe the motion of a 6 DOF rigid body it is usual to define two reference frames :

- The Earth inertial frame (E-Frame);
- The Body-fixed frame (B-Frame).

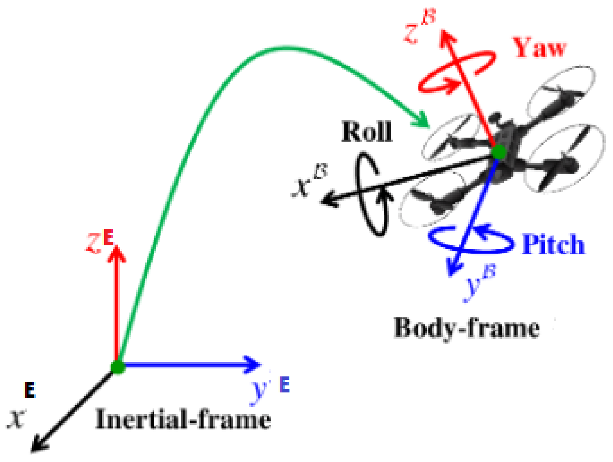


Figure 2.2: The relative orientation between body-frame and inertial-frame [21].

The linear position r_e of the quadrotor is determined by the coordinates of the vector between the origin of the B-frame and the origin of the E-frame with respect to the E-frame according to the equation:

$$r_e = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.1}$$

The angular position (or attitude) y_e of the quadrotor is defined by the orientation of the B-frame with respect to the E-frame.

$$y_e = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{2.2}$$

The body-fixed frame is fixed with a quadrotor. The linear velocity v_b, the angular velocity w_b $[rad/s]$, are defined in this frame according to those equations:

$$v_b = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.3}$$

$$w_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.4}$$

To relate the position of the quadcopter in the E-frame to its velocity in B-frame, we use a rotation matrix R. This rotation matrix, denoted as R, is obtained by multiplying the three fundamental rotation matrices together.

$$R(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{bmatrix} \quad R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad R(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The multiplication of the three matrices gives the complete rotational matrix:

$$R = \begin{bmatrix} c(\theta)c(\psi) & s(\theta)s(\psi) & -s(\theta) \\ -c(\varphi)s(\psi) + s(\varphi)s(\theta)c(\psi) & c(\varphi)c(\psi) + s(\varphi)s(\theta)s(\psi) & c(\theta)s(\varphi) \\ s(\psi)s(\varphi) + c(\psi)s(\varphi)s(\theta) & -s(\varphi)s(\psi) + c(\varphi)s(\psi)s(\theta) & c(\theta)c(\varphi) \end{bmatrix} \tag{2.5}$$

Where : $c(x) = \cos(x), s(x) = \sin(x)$.

To link between y'_e that is measured in the inertial frame and angular body rates w_b. The following transformation is needed:

$$w_b = R_r y'_e \tag{2.6}$$

$$R_r = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\varphi) & sin(\varphi)cos(\theta) \\ 0 & -sin(\varphi) & cos(\varphi)cos(\theta) \end{bmatrix} \tag{2.7}$$

Around the hover position, we can simplify the equation by assuming a small angle, such that: $cos(\varphi) \approx cos(\theta) \approx 1$, and $sin(\varphi) \approx sin(\theta) \approx 0$. Thus R_r can be simplified to an identity matrix I [22].

2.1.2 Quadrotor Dynamic Model

Dynamics is a branch of mechanics that studies the effects of forces and torques on the motion of a body or system of bodies. There are several techniques that can be used to derive the equations of a rigid body, but in general, there are two methods for determining the equations of motion of a quadrotor:

- Newton-Euler formulation;
- Euler-Lagrange formulation.

The Newton-Euler dynamic model is the preferable choice for quad-copters with many degrees of freedom [23]. The advantage of this model is that it produces a model recursively which is generally faster in calculation and command. To simplify the modeling, the Newton-Euler approach is used.

The Newton-Euler equation can be expressed as [24] :

$$\begin{bmatrix} f \\ T \end{bmatrix} = \begin{bmatrix} mI & O_3 \\ O_3 & I_3 \end{bmatrix} \begin{bmatrix} a \\ \alpha \end{bmatrix} + \begin{bmatrix} 0 \\ w + I_3 w \end{bmatrix} \tag{2.8}$$

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.9}$$

Where:

f: the net force acting on the quadrotor

T: the net torque

I: 3×3 identity matrix

m: the quadrotor mass.

a: the linear acceleration of the center of mass.

α : the angular acceleration.

I_{xx}, I_{yy},I_{zz}: the area moments of inertia about the principal axes in the b-frame.

The motion of the quadrotor can be split into two subsystems: the rotational subsystem, which includes (roll, pitch, and yaw) , and the translational subsystem, which involves (altitude, x position, and y position). The rotational subsystem is fully actuated whereas the translational subsystem is underactuated.

1. **Rotational Equation of Motion**

The rotational equation of the quadrotor, based on the Newton-Euler method, is derived from the body frame of the quadcopter using the following formalism:

$$M_B = M_G + Iw'_b + w_b \times Iw_b \tag{2.10}$$

Where:

I: Quadrotor inertia Matrix.

w_b: Angular velocity.

M_G: Gyroscopic moments due to rotors' inertia.

M_B: All the moments acting on the quadcopter in its body frame.

Moments Acting on the Quadrotor (M_B)

Each rotor in the quadrotor creates an upward thrust force and generates a moment with the direction opposite to the direction of rotation of the corresponding rotor i.

$$f_i = k_f \cdot \Omega_i^2 \tag{2.11}$$

$$M_i = k_m \cdot \Omega_i^2 \tag{2.12}$$

Where:

k_f and k_m are the aerodynamic force and moment constants respectively.

Ω_i is the angular velocity of rotor i.

we will express the moment about each axis as follows:

The total moment about the x-axis can be expressed as:

$$M_x = f_1 l - f_2 l - f_3 l + f_4 l = (k_f \Omega_1^2) l - (k_f \Omega_2^2) l - (k_f \Omega_3^2) l + (k_f \Omega_4^2) l \tag{2.13}$$

$$M_x = l k_f (\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{2.14}$$

The total moment about the y-axis can be expressed as:

$$M_y = f_1 l - f_2 l + f_3 l - f_4 l = (k_f \Omega_1^2) l - (k_f \Omega_2^2) l + (k_f \Omega_3^2) l - (k_f \Omega_4^2) l \tag{2.15}$$

$$M_y = l k_f (\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \tag{2.16}$$

The total moment about the z-axis can be written as:

$$M_z = M_1 + M_2 - M_3 - M_4 \tag{2.17}$$

$$M_z = k_m (\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \tag{2.18}$$

By combining the equations (2.14),(2.16) and (2.18) we can get:

$$M_B = \begin{bmatrix} l k_f (\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ l k_f (\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \\ k_m (\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{2.19}$$

Where:

l: The length from the center of the quadrotor to any of the propellers.

Diagonal Inertia Matrix (I)

The quadrotor's inertia matrix is structured as diagonal matrix, its diagonal elements are I_{xx},I_{yy} and I_{zz}, which represents the mass moments of inertia about the principal axes in the body frame.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.20}$$

Gyroscopic Moment (M_G)

The gyroscopic effect is exhibited by a rotor when an external force is applied to change its orientation. The gyroscopic moments are defined as follows:

$$M_G = w_b \times \begin{bmatrix} 0 \\ 0 \\ J_r\Omega_r \end{bmatrix} \tag{2.21}$$

Where:

J_r: Inertia of rotors

Ω_r: rotors' relative Speed

$$\Omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$$

2. **Translational Equations of Motion**

The quadrotor's translation equation of motion by using Earth's frame and Newton's second rule is :

$$mr_e = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_b \tag{2.22}$$

where:

m: quadrotor's mass.

g: gravitational acceleration $g = 9.81m/s^2$.

R: Rotational matrix.

r_e: linear position of quadrotor.

F_b: non-gravitational forces acting on the quadrotor.

Non-gravitational Force (F_b)

The non-gravitational force is the thrust force acting on the quadcopter in the vertical direction (there is no rolling or pitching).It can be expressed as follows:

$$F_b = \begin{bmatrix} 0 \\ 0 \\ -k_f(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{2.23}$$

2.2 State Space Model

This section focuses on discussing the mathematical equations of motion presented in a state space representation, and this will help to control the quadrotor.

2.2.1 State Vector X

Due to the quadrotor's six degrees of freedom, a state vector is required to represent its position in space, as well as its angular and linear velocities. It will include twelve elements given in those equations:

$$X = \begin{bmatrix} \phi \\ \phi' \\ \theta \\ \theta' \\ \psi \\ \psi' \\ z \\ z' \\ x \\ x' \\ y \\ y' \end{bmatrix} \tag{2.24}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} \tag{2.25}$$

2.2.2 Control Input Vector U

The control vector U consisting of the control inputs U_1 through U_4 is defined to be:

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{2.26}$$

Where:

$$U_1 = k_f(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{2.27}$$

$$U_2 = k_f(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{2.28}$$

$$U_3 = k_f(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \tag{2.29}$$

$$U_4 = k_m(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \tag{2.30}$$

The equations (2.27) to (2.30) can be expressed in matrix form as:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ K_f & -K_f & -K_f & K_f \\ K_f & -K_f & K_f & -K_f \\ K_m & K_m & -K_m & -K_m \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{2.31}$$

The control input can be used to calculate the velocity of the rotor using the following equation:

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4K_f} & \frac{1}{2K_f} & \frac{1}{2K_f} & \frac{1}{4K_m} \\ \frac{1}{4K_f} & -\frac{1}{2K_f} & -\frac{1}{2K_f} & \frac{1}{4K_m} \\ \frac{1}{4K_f} & -\frac{1}{2K_f} & \frac{1}{2K_f} & -\frac{1}{4K_m} \\ \frac{1}{4K_f} & \frac{1}{2K_f} & -\frac{1}{2K_f} & -\frac{1}{4K_m} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{2.32}$$

The calculation of the rotors' velocities from the control inputs can be expressed as :

$$\Omega_1 = \sqrt{\frac{1}{4K_f}U_1 + \frac{1}{2K_f}U_2 + \frac{1}{2K_f}U_3 + \frac{1}{4K_m}U_4} \tag{2.33}$$

$$\Omega_2 = \sqrt{\frac{1}{4K_f}U_1 - \frac{1}{2K_f}U_2 - \frac{1}{2K_f}U_3 + \frac{1}{4K_m}U_4} \tag{2.34}$$

$$\Omega_3 = \sqrt{\frac{1}{4K_f}U_1 - \frac{1}{2K_f}U_2 + \frac{1}{2K_f}U_3 - \frac{1}{4K_m}U_4} \tag{2.35}$$

$$\Omega_4 = \sqrt{\frac{1}{4K_f}U_1 + \frac{1}{2K_f}U_2 - \frac{1}{2K_f}U_3 - \frac{1}{4K_m}U_4} \tag{2.36}$$

2.2.3 Rotational Equation of motion in state space form

The equation describing the total moments acting on the quadcopter body becomes:

$$M_B = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} \tag{2.37}$$

By utilizing equations (2.10) and (2.37), the angular accelerations can be expressed as follows:

$$\phi" = \frac{l}{I_{xx}}U_2 - \frac{J_r}{I_{xx}}\theta'\Omega_r + \frac{I_{yy} - I_{zz}}{I_{xx}}\theta'\psi' \tag{2.38}$$

$$\theta" = \frac{l}{I_{yy}}U_3 - \frac{J_r}{I_{yy}}\phi'\Omega_r + \frac{I_{zz} - I_{xx}}{I_{yy}}\phi'\psi' \tag{2.39}$$

$$\psi" = \frac{1}{I_{zz}}U_4 + \frac{I_{xx} - I_{yy}}{I_{zz}}\phi'\theta' \tag{2.40}$$

To simplify equations (2.38) to (2.40), the inertia terms are written as:

$$a_1 = \frac{I_{yy} - I_{zz}}{I_{xx}} \quad a_2 = \frac{J_r}{I_{xx}} \quad a_3 = \frac{I_{zz} - I_{xx}}{I_{yy}}$$

$$a_4 = \frac{J_r}{I_{yy}} \quad a_5 = \frac{I_{xx} - I_{yy}}{I_{zz}}$$

$$b_1 = \frac{l}{I_{xx}} \quad b_2 = \frac{l}{I_{yy}} \quad b_3 = \frac{1}{I_{zz}}$$

Equations (2.38) through (2.40) can be rewritten in a simple form in terms of the system states as:

$$\phi" = b_1 U_2 - a_2 x_4 \Omega_r + a_1 x_4 x_6 \tag{2.41}$$

$$\theta" = b_2 U_3 - a_4 x_2 \Omega_r + a_3 x_2 x_6 \tag{2.42}$$

$$\psi" = b_3 U_4 + a_5 x_2 x_4 \tag{2.43}$$

2.2.4 Translational Equations of motion in state space form

The equation of the upward force acting on the quadrotor becames:

$$F_b = \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} \tag{2.44}$$

By expanding equations (2.22),(2.23), and (2.44), the accelerations can be expressed as follows [22]:

$$x'' = -\frac{U_1}{m}(\sin\phi \sin\psi + \cos\phi \cos\psi \sin\theta) \tag{2.45}$$

$$y'' = -\frac{U_1}{m}(\sin\theta \sin\psi \cos\phi - \sin\phi \cos\psi) \tag{2.46}$$

$$z'' = -g + \frac{U_1}{m}(\cos\phi \cos\theta) \tag{2.47}$$

Rewriting in terms of the state variable X:

$$x'' = -\frac{U_1}{m}(\sin x_1 \sin x_5 + \cos x_1 \cos x_5 \sin x_3) \tag{2.48}$$

$$y'' = -\frac{U_1}{m}(\sin x_3 \sin x_5 \cos x_1 - \sin x_1 \cos x_5) \tag{2.49}$$

$$z'' = -g + \frac{U_1}{m}(\cos x_1 \cos x_3) \tag{2.50}$$

2.2.5 State Space Representation

By incorporating the rotational angular acceleration equations (2.40) to (2.42), and the translation equations (2.44) to (2.46), the complete mathematical model of the quadrotor can be represented in a state space form as follows:

$$\begin{cases} x_1' = \phi' = x_2 \\ x_2' = \phi" = b_1 U_2 - a_2 x_4 \Omega_r + a_1 x_4 x_6 \\ x_3' = \theta' = x_4 \\ x_4' = \theta" = b_2 U_3 - a_4 x_2 \Omega_r + a_3 x_2 x_6 \\ x_5' = \psi' = x_6 \\ x_6' = \psi" = b_3 U_4 + a_5 x_2 x_4 \\ x_7' = z' = x_8 \\ x_8' = z" = g - \frac{U_1}{m}(\cos x_1 \cos x_3) \\ x_9' = x' = x_{10} \\ x_{10}' = x" = -\frac{U_1}{m}(\sin x_1 \sin x_5 + \cos x_1 \cos x_5 \sin x_3) \\ x_{11}' = y' = x_{12} \\ x_{12}' = y" = -\frac{U_1}{m}(\sin x_3 \sin x_5 \cos x_1 - \sin x_1 \cos x_5) \end{cases} \tag{2.51}$$

Whereas:

$$\begin{cases} U_x = \sin x_1 \sin x_5 + \cos x_1 \cos x_5 \sin x_3 \\ U_y = \sin x_3 \sin x_5 \cos x_1 - \sin x_1 \cos x_5 \end{cases}$$

2.3 Conclusion

In this chapter, the main focus was on the modeling of a quadrotor. We delved into the dynamics of a quadrotor system and discussed the mathematical models used to represent its motion and behavior. Through the exploration of fundamental concepts such as kinematics, dynamics, and control, we have established a strong basis in quadrotor technology.

Chapter 3

System Control

In this chapter, we are interesting in the controller design for the modeled Quadrotor which is an underactuated, and highly nonlinear system. To address the control challenges, we intend to utilize the backstepping control technique as our chosen controller for the quadrotor system.

3.1 Back-stepping Control Algorithm

Backstepping is a control methodology used to design control laws for systems with nonlinear dynamics. It is particularly useful for systems with high levels of uncertainty or complexity. The backstepping approach involves dividing the system into subsystems and designing controllers for each subsystem. This design is based on the derived state vector equations and relies on Lyapunov's theory (particularly Lyapunov's second method).

In the context of quadrotor control, backstepping can be used to develop control laws for both the orientation (ϕ, θ, ψ) and position (x, y, z) subsystems.

3.2 The Adopted Control Strategy

In this section, we have proposed a control strategy for the Quadrotor based on two loops (outer and inner loops). The inner loop contains attitude control laws for the roll, pitch, and yaw control. The outer loop contains controllers for the x,y, and z positions. The output of the outer loop is injected into a block called « Correction block » which is introduced to generate the desired roll and pitch angles. Figure 3.1 presents the scheme that illustrates the adopted strategy.

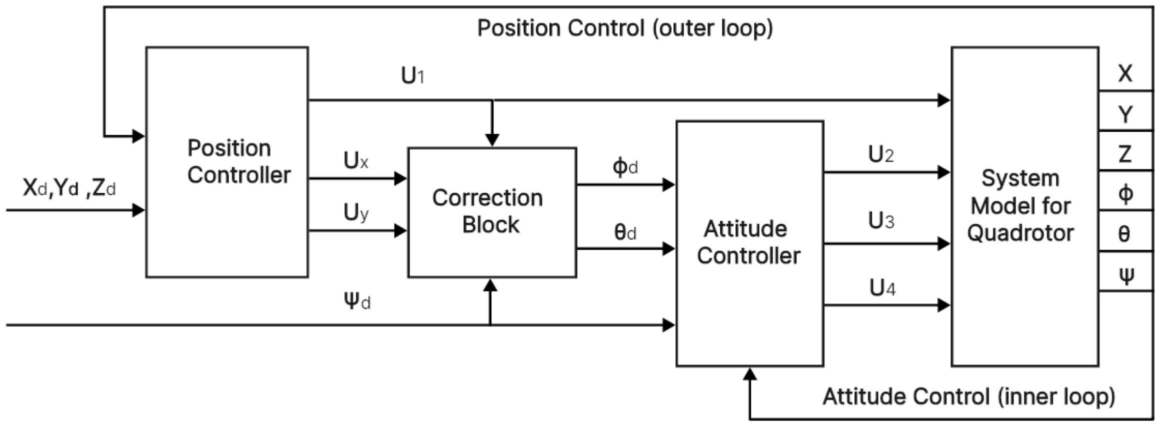


Figure 3.1: Quadrotor control strategy scheme.

3.2.1 Attitude and Position control

We start by using the first two variables of the state vector which are the roll angle and its derivative, we get:

$$\begin{cases} x_1' = x_2 \\ x_2' = x_4 x_6 a_1 - a_2 \Omega_r x_4 + b_1 U_2 \end{cases} \tag{3.1}$$

The synthesis of the roll command U_2 is done in two stages :

Step 1 :

We take the first equation:

$$x_1' = x_2 \tag{3.2}$$

We define the error ϵ_1 between the desired and the measured roll angle.

$$\epsilon_1 = x_1^d - x_1 \tag{3.3}$$

$$\epsilon_1' = (x_1^d)' - x_2 \tag{3.4}$$

The error ϵ_1 must asymptotically converge towards zero, this definition explicitly indicates our command objective. We choose the first candidate Lyapunov function of the following form:

$$V_1 = \frac{1}{2}\epsilon_1^2 \tag{3.5}$$

Where V_1 is computed as follows:

$$V_1' = \epsilon_1(\epsilon_1)' = \epsilon_1((x_1^d)' - x_2) \tag{3.6}$$

To ensure stability, it is necessary that $V_1' \leq 0$; for that we take as a virtual command x_2^d with:

$$x_2^d = (x_1^d)' + k_1\epsilon_1 \quad \text{where } k_1 > 0 \tag{3.7}$$

Step 2 :

As the virtual control cannot instantly take its desired value, we seek in what follows to stabilize the error between the virtual control and the stabilizing function.

$$\epsilon_2 = x_2^d - x_2 \tag{3.8}$$

$$\epsilon_2 = (x_1^d)' + k_1\epsilon_1 - x_2 \tag{3.9}$$

so :

$$V_2 = V_1 + \frac{1}{2}\epsilon_2^2 \tag{3.10}$$

$$(V_2)' = (V_1)' + \epsilon_2(\epsilon_2)' \;= -\epsilon_1\epsilon_2 - k_1\epsilon_1^2 + \epsilon_2(x_2)' - (x_1^d)'' - k_1(\epsilon_1)' \tag{3.11}$$

The first Lyapunov function is added to a quadratic term of the second error variable ϵ_2 to get a second Lyapunov function V_2.

$$V_2 = k_1\epsilon_1^2 + k_2\epsilon_2^2 \quad \text{where } k_2 > 0 \tag{3.12}$$

$$V_2' = V_1' + \epsilon_2\epsilon_2' = -\epsilon_1\epsilon_2 - k_1\epsilon_1^2 + \epsilon_2(x_2' - x_1^{d\,\text{"}} - k_1\epsilon_1') \tag{3.13}$$

The control input U_2 that ensures V_2'=-k$_1\epsilon_1^2 - k_2\epsilon_2^2$ is given by :

$$U_2 = \frac{1}{b_1}((x_1^d)'' - x_4x_6a_1 - a_2\Omega_r x_4 + b_1U_2 + k_1(\epsilon_2 - k_1\epsilon_1) + \epsilon_1 + k_2\epsilon_2) \tag{3.14}$$

The term $k_2\epsilon_2$ with $k_2 > 0$ is added to stabilize ϵ_2.

U_3, U_4, U_1, U_x, and U_y are calculated in the same way.

$$U_3 = \frac{1}{b_2}((x_3^d)'' - x_2x_6a_3 - a_4\Omega_r x_2 + k_3(\epsilon_4 - k_3\epsilon_3) + \epsilon_3 + k_4\epsilon_4) \tag{3.15}$$

$$U_4 = \frac{1}{b_3}((x_5^d)'' - x_2x_4a_5 + k_5(\epsilon_6 - k_5\epsilon_5) + \epsilon_5 + k_6\epsilon_6) \tag{3.16}$$

$$U_1 = \frac{m}{\cos(x_1)\cos(x_3)}(-\epsilon_7 + g - k_7x_7 + k_8\epsilon_8) \tag{3.17}$$

$$U_x = \frac{m}{U_1}((x_9^d)'' + k_9(\epsilon_{10} - k_9\epsilon_9) + \epsilon_9 + k_{10}\epsilon_{10}) \tag{3.18}$$

$$U_y = \frac{m}{U_1}((x_{11}^d)'' + k_{11}(\epsilon_{12} - k_{11}\epsilon_{11}) + \epsilon_{11} + k_{12}\epsilon_{12}) \tag{3.19}$$

3.3 Simulation of Control System

In this section, the overall block diagram of the backstepping approach using Matlab Simulink is presented in figure 3.2, to be used for control algorithm development and verification, before working with a real experimental system.

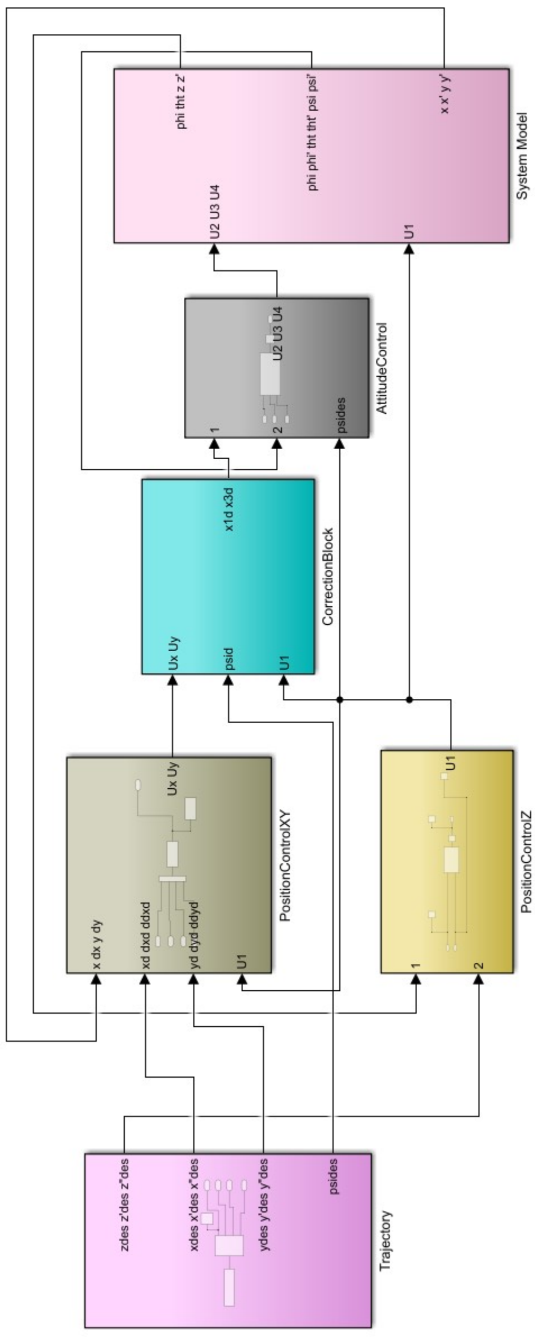


Figure 3.2: The overall Simulink model of the Quadrotor using Backstepping.

Table 3.1: Parameters of the Quadrotor

Parameter	Value	Units
g	9.81	$\mathrm{m/s^2}$
m	0.5	kg
l	0.2	m
I_{xx}	2.510^{-4}	$\mathrm{kg \cdot m^2}$
I_{yy}	2.3210^{-4}	$\mathrm{kg \cdot m^2}$
I_{zz}	3.73810^{-4}	$\mathrm{kg \cdot m^2}$

3.3.1 Description of Simulation Blocks

In this section, we will cover the simulation blocks used to control a quadrotor's behavior. We will explain each block's function, equations, and inputs/outputs, how they contribute to the overall control of the quadrotor, and how these blocks work together to achieve stable and precise control.

a) **Trajectory Block:** It takes as inputs x, y, z and ψ angle to generate the desired trajectory for the quadrotor to follow, which can be specified using mathematical functions or a set of waypoints. The output of this block shown in figure 3.3, is the desired position and yaw orientation of the quadrotor, which are then used as inputs for the "Position Controller" block.

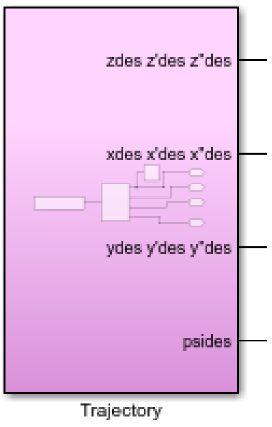


Figure 3.3: Simulink Trajectory block.

b) **Position Controller Block**: Responsible for implementing the control algorithm to track the desired position of the quadrotor generated by the "Trajectory Desired" block. In our model shown in figure 3.4, two separate position controllers are used, one for the x and y directions and another for the z direction. The equations implemented on those blocks are eqs (3.18), and (3.19) for x and y to provide the virtual commands U_x and U_y respectively, and eq (3.17) for U_1.

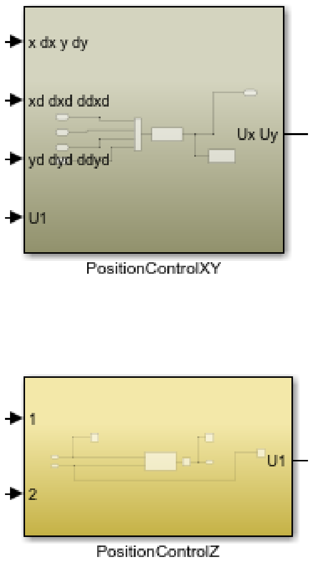


Figure 3.4: Simulink block of position controllers.

c) **Correction Block:** Uses the commands generated from the position controllers (U_x,U_y, and U_1) and ψ_d to compute the desired orientation of the quadrotor (see figure 3.5). The outputs are then used as inputs for the "Attitude Controller" block. This approach allows the desired angles to be calculated from the equations below [25] without directly specifying them, which can simplify the control design.

$$\begin{cases} \phi_d = \arcsin\left(m \cdot \dfrac{U_x \cdot \sin(\psi_d) - U_y \cdot \cos(\psi_d)}{\sqrt{U_x^2 + U_y^2 + U_1^2}}\right) \\ \theta_d = \arctan\left(\dfrac{U_x \cdot \cos(\psi_d) + U_y \cdot \sin(\psi_d)}{U_1 + g}\right) \end{cases}$$

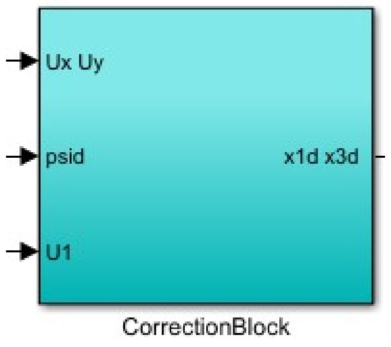


Figure 3.5: Simulink block of Correction block.

d) **Attitude Controller:** This block given in figure 3.6 implements a control algorithm to track the desired orientation. The outputs generated by this block are the U_2, U_3, and U_4 which are implemented using the eqs (3.14), (3.15), and (3.16) respectively.

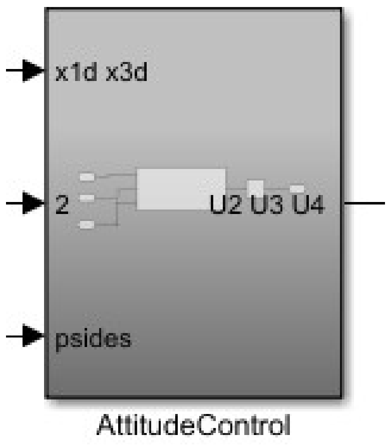


Figure 3.6: Simulink block of Attitude controller.

e) **Quadrotor System Model:** This block describes the equations of motion for our quadrotor. The equations implemented in this block are based on Newton's laws and the principles of rigid body dynamics. The output of this block is the current position and orientation as illustrated in figure 3.7, which is fed back to the "Position Controller" and "Attitude Controller" blocks.

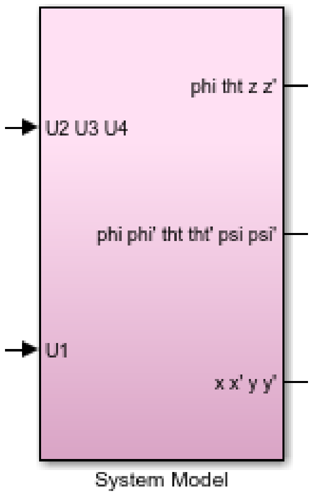


Figure 3.7: Simulink block of Quadrotor System model.

3.3.2 Simulation Results

Altitude /Attitude Stabilization

The control objectives were to achieve and maintain a desired altitude/attitude, allowing the quadrotor to hover steadily at a fixed point. The desired altitude/attitude was specified as $[z_d, \phi_d, \theta_d, \psi_d] = [1, 0, 0, 0]^T$.The parameters given in Table 3.2 is used.

Table 3.2: Parameter Values

Parameter	k1	k2	k3	k4	k5	k6
Value	18	14	15	19	16	17

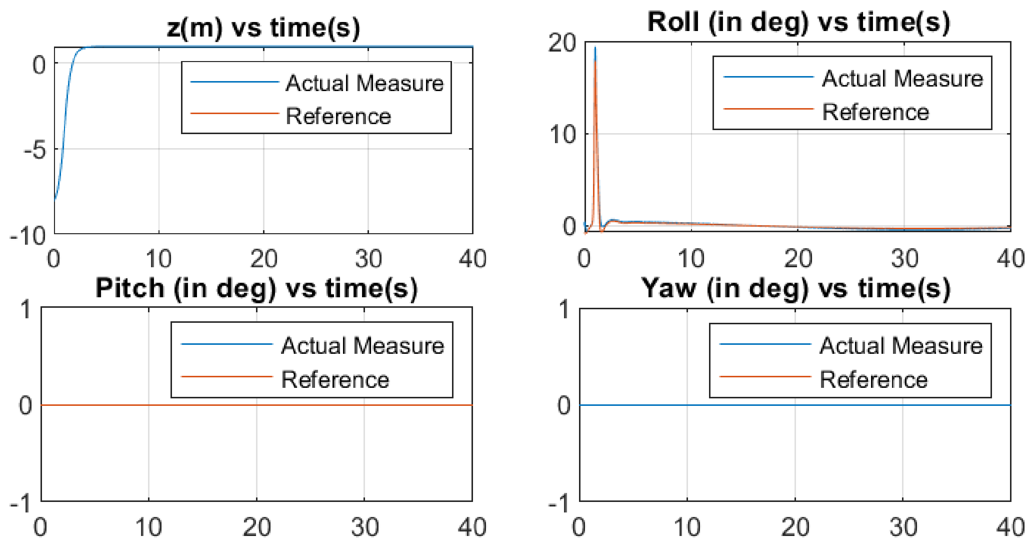


Figure 3.8: Altitude/attitude of the hovering quadrotor.

Trajectory Tracking

In this section, we present the simulation results for trajectory tracking by the quadrotor system, using the parameter of tables 3.2 and 3.3. The final simulation time is set to $t_{\text{final}} = 100\text{s}$. The desired trajectory is chosen to be a helix shape given by:

$$\begin{cases} x_d(t) = 8\cos(\frac{1}{2}t) \\ y_d(t) = 8\sin(\frac{1}{2}t) \\ z_d(t) = 0.2t \\ \Psi_d = 0 \end{cases} \tag{3.20}$$

Table 3.3: Parameter Values

Parameter	k7	k8	k9	k10	k11	k12
Value	4	3	10	10	9	11

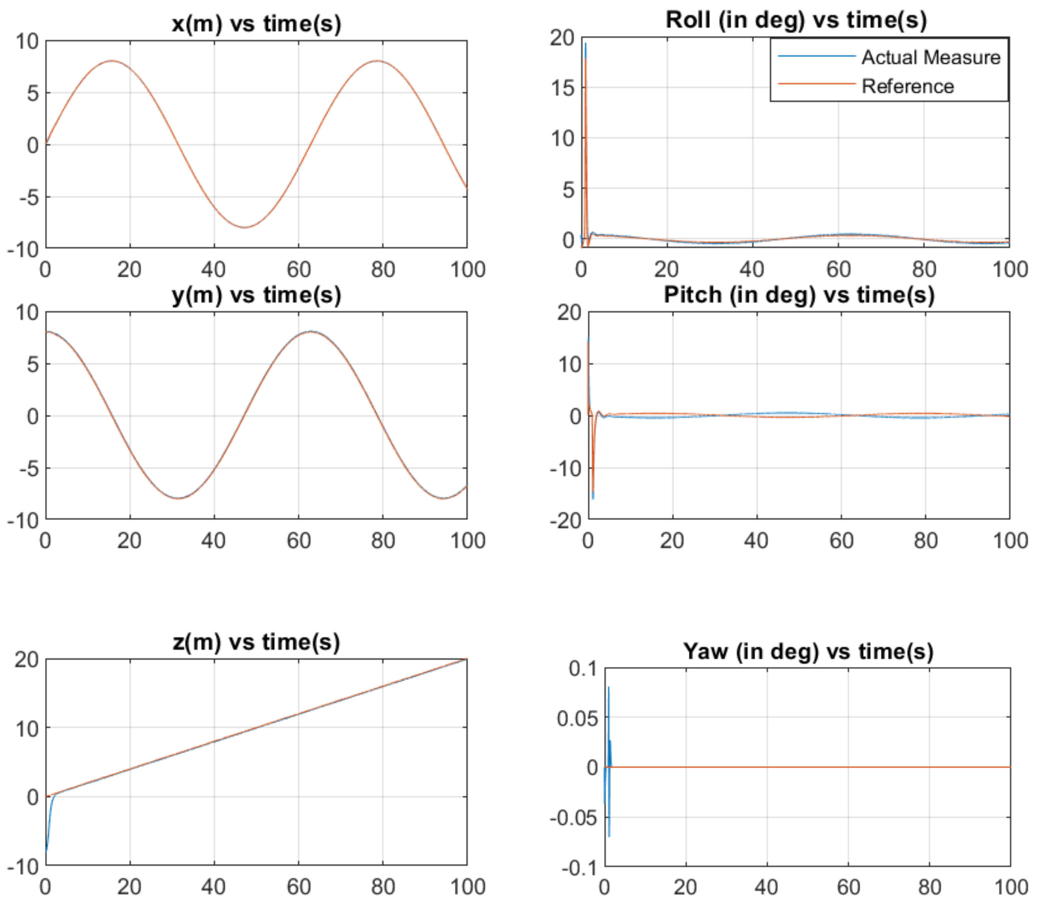


Figure 3.9: Trajectory Tracking Results.

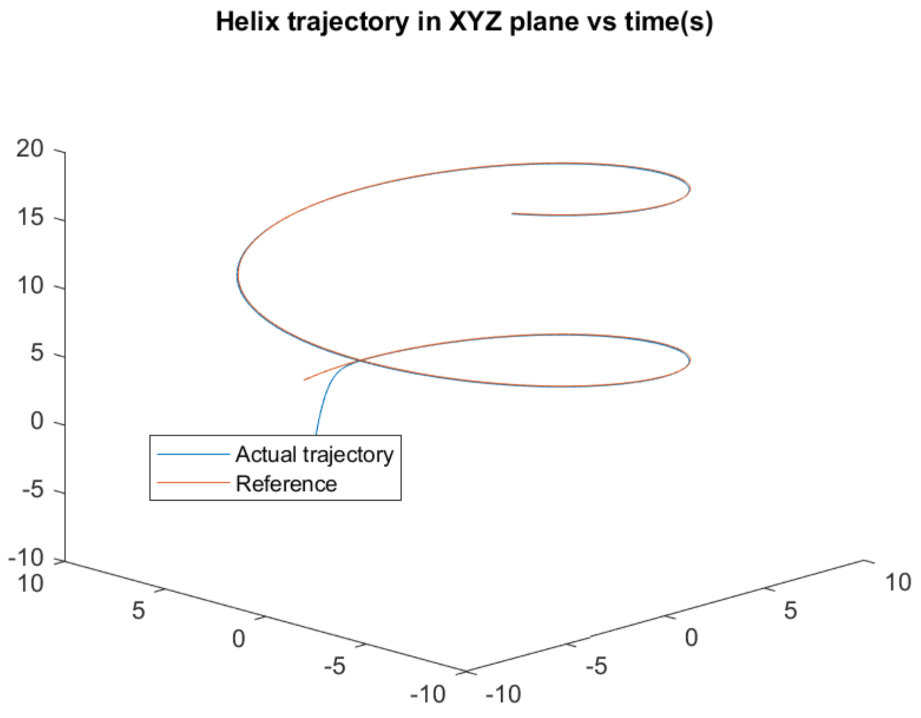


Figure 3.10: Helix trajectory in XYZ plane.

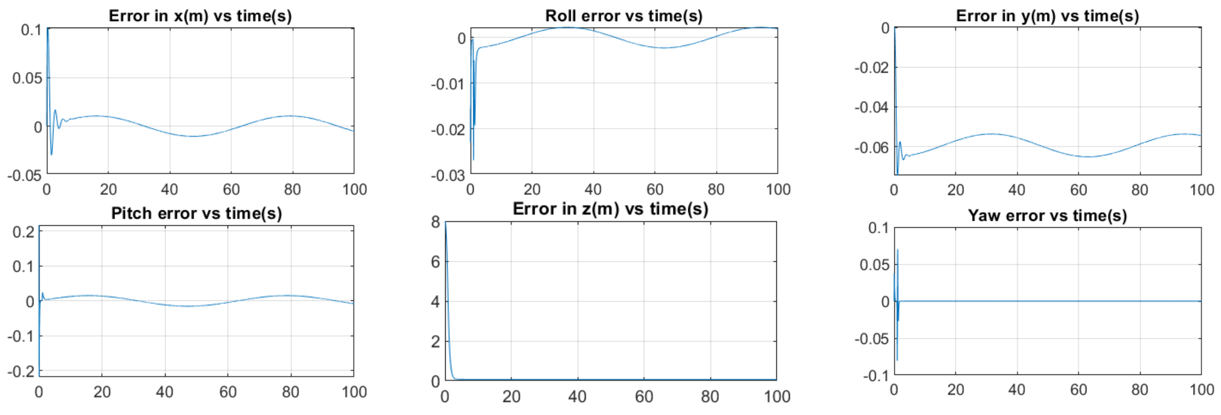


Figure 3.11: Error results of positions and orientations simulation.

3.4 Results and Discussion

The results of the simulations are shown in figures 3.8 to 3.11. From figure 3.8 it can be observed that the altitude/attitude of the quadrotor can be maintained at the desired altitude/attitude, that is, the hovering flight is stable. figure 3.9 reveals the maximum overshoots obtained for roll, pitch, and yaw angles. The values of 18.40°, -17.91°, and 0.08° respectively, indicate that there is a slight deviation from the desired angles during the start time of simulations. From figure 3.10 the graph demonstrates the quadrotor's ability to track a predefined trajectory (Helix) over time. Initially, the quadrotor exhibits some deviations from the desired trajectory. However, as the simulation progresses, we observe a gradual reduction in the deviation, indicating an improvement in trajectory tracking. Overall, the simulation results using the back-stepping algorithm demonstrate the quadrotor system's ability to maintain stable hovering flight, achieve desired positions and angles with acceptable overshoots, and effectively track predefined trajectories. Furthermore, the errors graph in figure 3.11 showcases a consistent convergence toward zero as the simulation progresses. The quadrotor starts at certain positions and angles, but over time, it gradually approaches zero.

3.5 Conclusion

In this chapter, we focused on the stabilization of a quadrotor using the backstepping control strategy. The objective was to ensure system stability and demonstrate the effectiveness of this approach. The simulation results demonstrated a high level of reliability, with a very short and precise response time. All these advantages prove the robustness of the backstepping command.

Chapter 4

Building and Testing a Quadrotor Prototype

The implementation focuses on the practical aspects of building and operating a quadrotor system. In this chapter, we delve into the details of assembling the hardware components, configuring the software, and conducting experiments to validate the functionality and performance of the quadrotor.

4.1 Quadrotor's Hardware Components

We will group the components into three categories: flight control system, powertrain, and wireless ground station.

1. **The Flight Control System:** including the ESP32 microcontroller and IMU sensor.
2. **Wireless Ground Station:** we will use Raspberry Pi to send commands and instructions to the ESP32 and recieve data using a Graphical User interface (GUI).
3. **The Power-Train:** is the high-current part of the quadrotor. It is powered by a battery, which supplies electrical current to the electronic speed controllers (ESCs). Each ESC converts the current into pulses, with pulse length determined by the microcontroller's motor commands. The characteristics of these modules are summarized in Table 4.1. These modules are connected as illustrated in figure 4.1,figure 4.2 and figure 4.3.

Table 4.1: The used Quadrotor components and their characteristics

Powertrain	**Flight Controller**
(×4) BLDC Motors 1000Kv	(×1) ESP32 Microcontroller
(×4) Propellers 1045	(×1) MPU6050 GY-521
(×4) Electronic Speed Controller (ESC) 30A	
(×1) Lipo Battery 2200mAH	
Wireless Ground Station	
Raspberry PI 4 Model b 8GB	

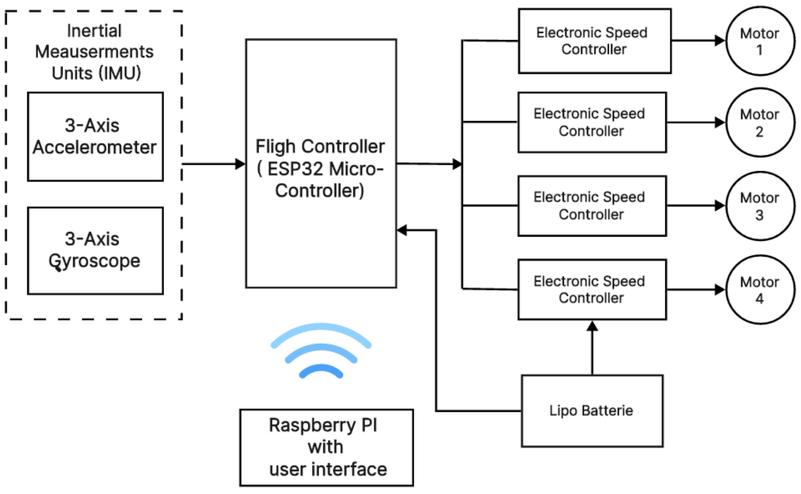

Inertial Meauserments Units (IMU)
3-Axis Accelerometer
3-Axis Gyroscope
Fligh Controller (ESP32 Micro-Controller)
Electronic Speed Controller
Electronic Speed Controller
Electronic Speed Controller
Electronic Speed Controller
Motor 1
Motor 2
Motor 3
Motor 4
Lipo Batterie
Raspberry PI with user interface

Figure 4.1: Block diagram of Quadrotor prototype.

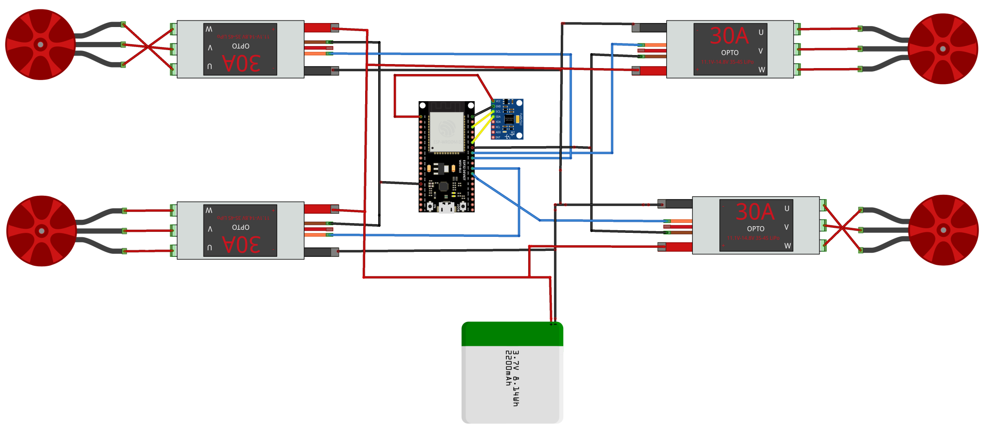


Figure 4.2: Circuit diagram of Quadrotor prototype.

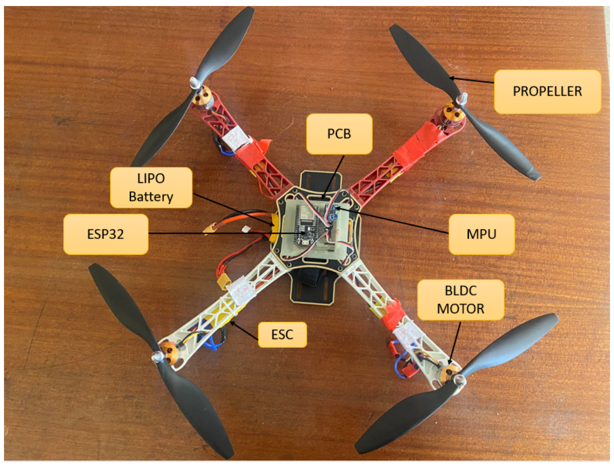


Figure 4.3: Quadrotor component.

4.1.1 ESP32 Discription

The ESP32 is a highly integrated system-on-chip (SoC) that combines a microcontroller unit (MCU) with a Wi-Fi and Bluetooth radio. It features a dual-core processor, ample memory, various peripherals, and built-in connectivity options, making it suitable for a wide range of applications [26]. The board used in our project is ESP32 Development Kit (ESP32 DevKitC) shown in figure 4.4.

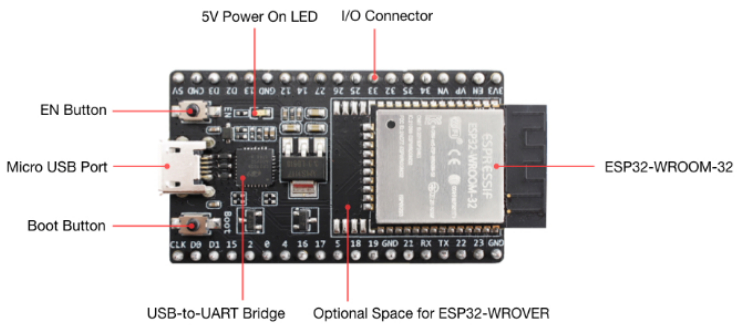


Figure 4.4: ESP32-DevKitC with ESP32-WROOM-32 module soldered [27].

4.1.2 Configuring the IMU (MPU6050)

For our project, the flight controller Board was paired with an MPU6050 IMU breakout board. The communication between them is done through standard I2C protocol. The MPU6050 GY-521 will be used, which incorporates both a gyroscope and an accelerometer, Where the gyroscope provides precise angular rate measurements, allowing us to accurately determine rotational movements.

On the other hand, the accelerometer measures linear acceleration, enabling to detect changes in orientations. we used those equations tocalculates the angles in degrees from the accelerometer :

$$\text{AngleRoll} = \arctan\left(\frac{\text{AccY}}{\sqrt{\text{AccX}^2 + \text{AccZ}^2}}\right)\frac{1}{\pi} \tag{4.1}$$

$$\text{AnglePitch} = -\arctan\left(\frac{\text{AccX}}{\sqrt{\text{AccY}^2 + \text{AccZ}^2}}\right)\frac{1}{\pi} \tag{4.2}$$

where AccX,AccY and AccZ are the accelerations readings around x, y and z axis respectively.

Despite their advantages, both the gyroscope and accelerometer have some limitations. Gyroscopes are susceptible to drift over time, causing small errors to accumulate and leading to inaccurate orientation estimation. Accelerometers can be sensitive to external vibrations and noise, affecting their ability to measure true acceleration as illustrated in figure 4.5.

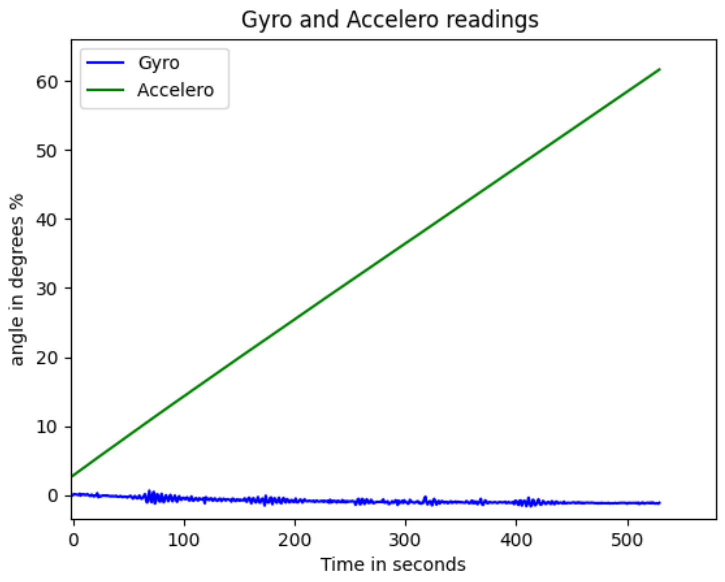


Figure 4.5: Comparison between Accelero and Gyro readings for pitch angle .

From figure 4.5, we can see that accelerometer provides instantaneous measurements of the pitch angle. The sharp peaks in the accelerometer signal suggest rapid variations in acceleration. In contrast, gyroscope readings are prone to drift, meaning that their measurement tends to gradually deviate from the true values over time. As a result, continuous re-calibration or correction is often required to maintain accurate measurements.

Gyro calibration involves determining and compensating for any biases or drifts in the sensor readings. The calibration process is explained in the flowchart given in figure 4.6.

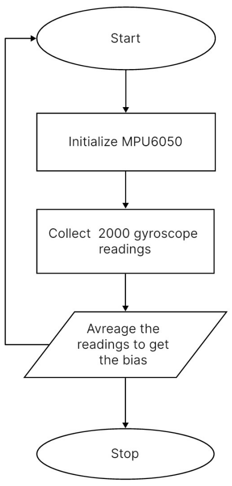

Start
Initialize MPU6050
Collect 2000 gyroscope readings
Avreage the readings to get the bias
Stop

Figure 4.6: Flowchart of Gyro calibration.

Complementary Filter

To further enhance the accuracy and stability of our sensor data, a complementary filter is used. The complementary filter combines the outputs of the gyroscope and accelerometer, leveraging their respective strengths while compensating for their weaknesses. This filtering algorithm allows us to obtain a reliable and robust estimation of the device's orientation.

$$\text{OutputAngle} = \alpha \cdot \text{GyroAngle} + (1 - \alpha) \cdot \text{AcceleroAngle}$$

For our quadrotor, a factor of 0.8 was assigned to the gyroscope data, while a factor of 0.2 was assigned to the accelerometer data. Some comparison results are illustrated in figure 4.7.

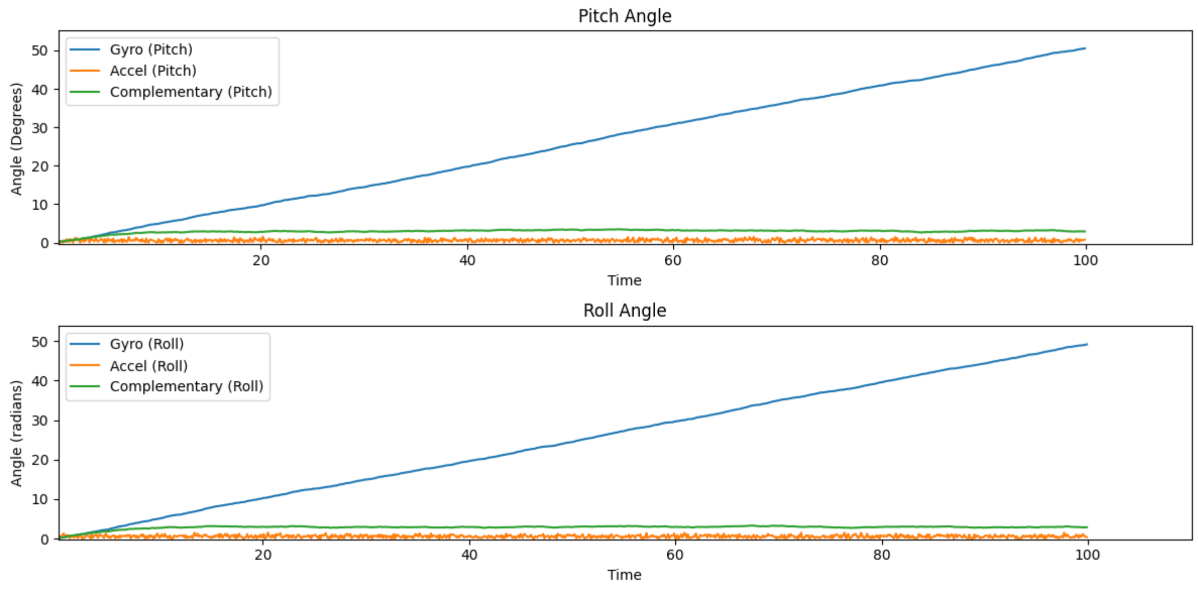


Figure 4.7: Comparison between accelerator, Gyro, and complementary filter for roll and pitch angles.

4.1.3 Electronic Speed Controller

The ESC acts as an intermediary between the flight controller and the motor. It receives commands from the control source, typically in the form of a Pulse width Modulation (PWM) signal, and translates them into appropriate power levels for the motor. The standard PWM range for controlling the motor speed of an ESC is commonly defined as 1000 to 2000 microseconds (µs) as shown in figure 4.8.

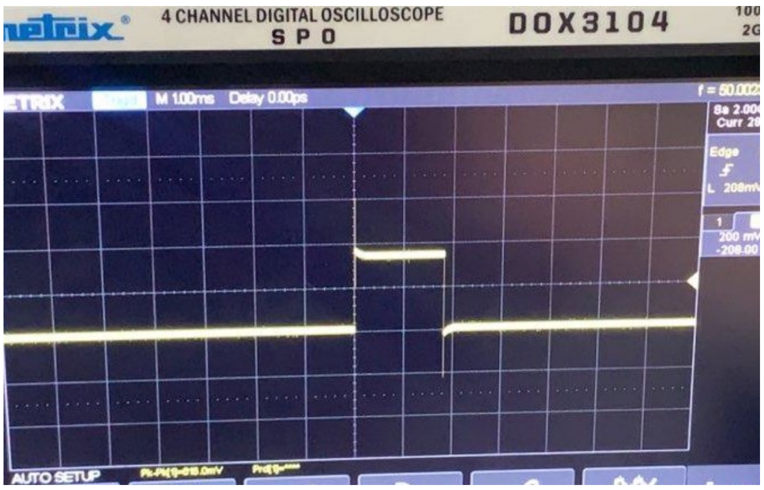


Figure 4.8: PWM signal of the ESC.

4.1.4 Motors Control

According to our MPU position as shown in figure 4.9, motor control signals are generated. This process ensures the quadrotor's movements based on the information provided by the sensor. We describe all movements as a linear combination of each other, for all motor outputs as shown in figure 4.10.

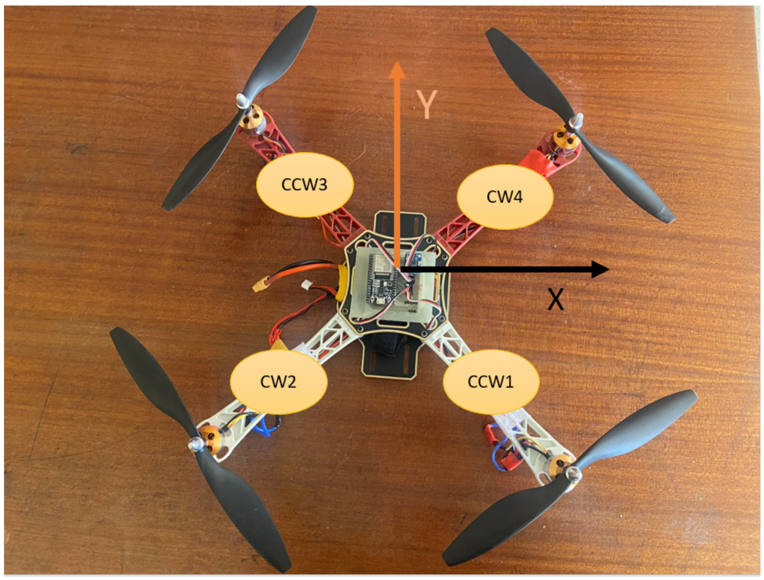
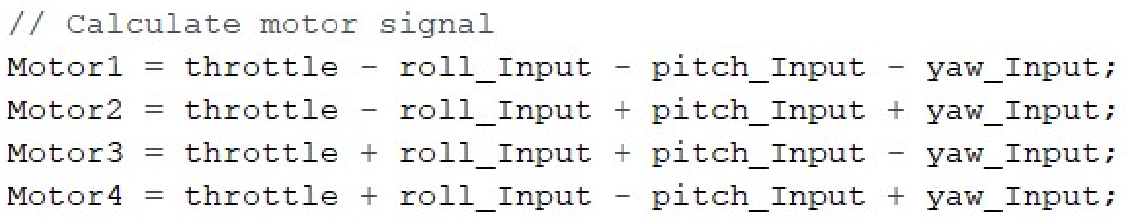


Figure 4.9: Quadrotor axis.



```
// Calculate motor signal
Motor1 = throttle - roll_Input - pitch_Input - yaw_Input;
Motor2 = throttle - roll_Input + pitch_Input + yaw_Input;
Motor3 = throttle + roll_Input + pitch_Input - yaw_Input;
Motor4 = throttle + roll_Input - pitch_Input + yaw_Input;
```

Figure 4.10: Generated motor signal .

4.1.5 PCB Design

Recognizing the need to simplify the connections between the components, specifically the ESP32, MPU 6050, and motors, we decided to leverage the power of Printed Circuit Boards (PCBs) shown in figure 4.11.

By designing and incorporating a PCB interface, we have significantly reduced the complexity of our wiring setup while enhancing overall system efficiency. We utilized the software called Eagle for designing our board. Eagle is a powerful and versatile tool that offers comprehensive features for creating professional-grade printed circuit boards. The overall circuit of the PCB design is shown in figure 4.12.

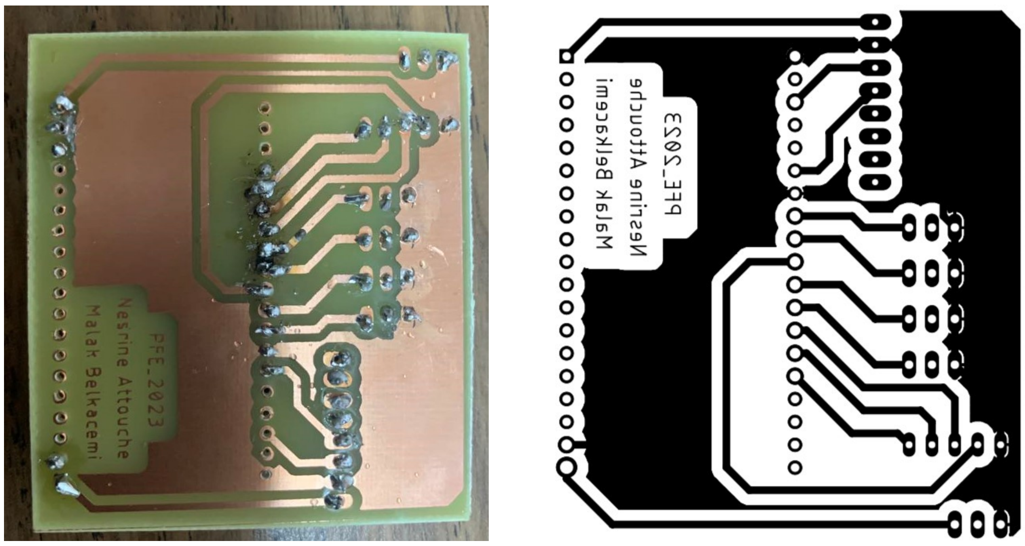


Figure 4.11: PCB Design using EAGLE software.

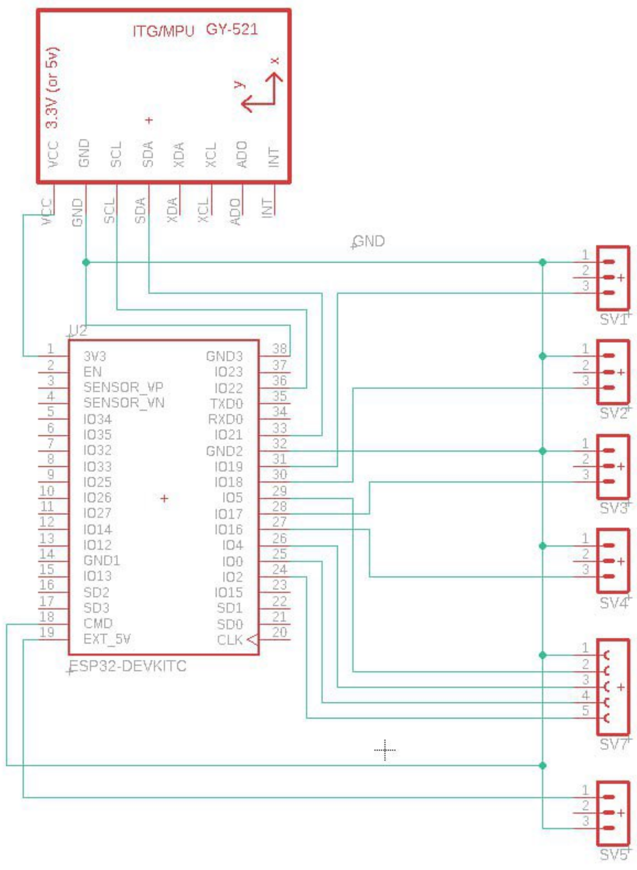


Figure 4.12: PCB Design using EAGLE software.

4.1.6 Raspberry PI

The Raspberry Pi is a series of small, single-board computers developed by the Raspberry Pi Foundation. It features a low-power ARM-based processor, memory, storage, and a range of connectivity options, all integrated onto a single board. It runs on a Linux-based operating system and supports a wide range of software tools and libraries [28]. In our project, we will use the Raspberry Pi 4 Model B as shown in figure 4.13.

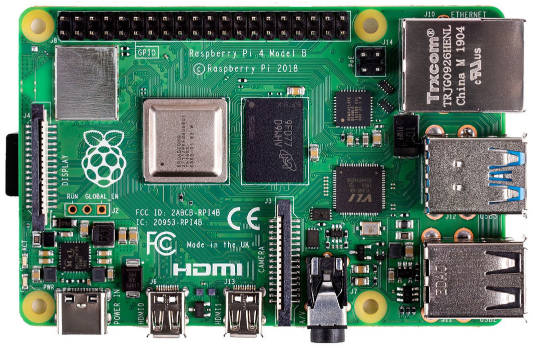


Figure 4.13: Raspberry Pi 4 Model B [29].

4.1.7 Wireless Graphical User interface using Raspberry PI

To take advantage of the WiFi capability of the ESP32, we aim to establish a wireless connection between the ESP32 and a Raspberry Pi. This connection allows us to transmit commands from the Raspberry Pi to the ESP32 wirelessly and to recieve sensor readings. To facilitate this communication, we intend to develop a graphical user interface (GUI) on the Raspberry Pi, as illustrated in figure 4.14.

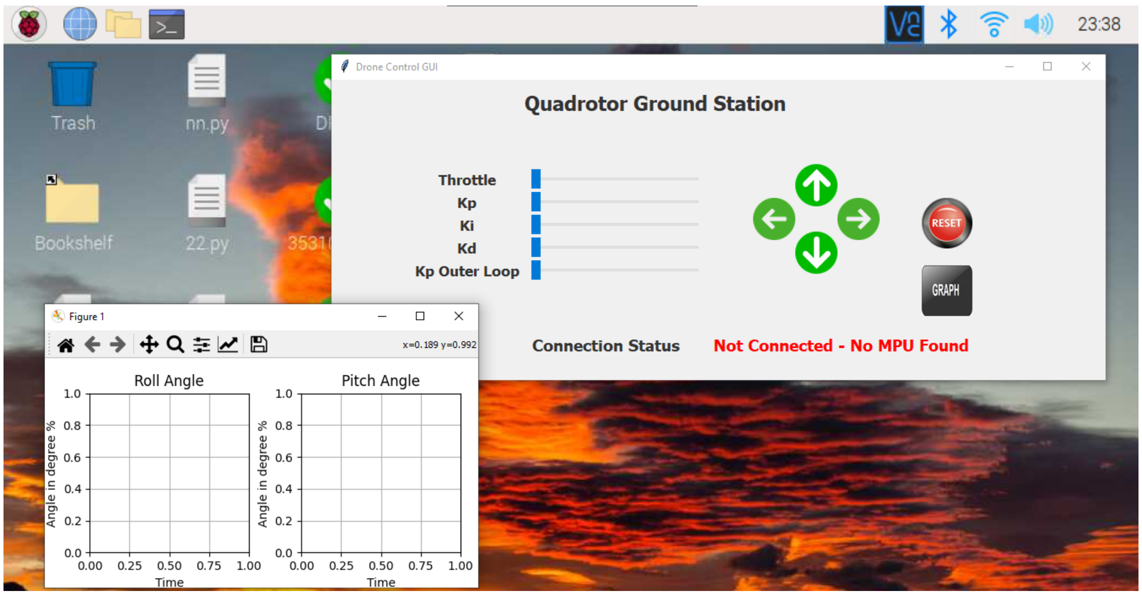
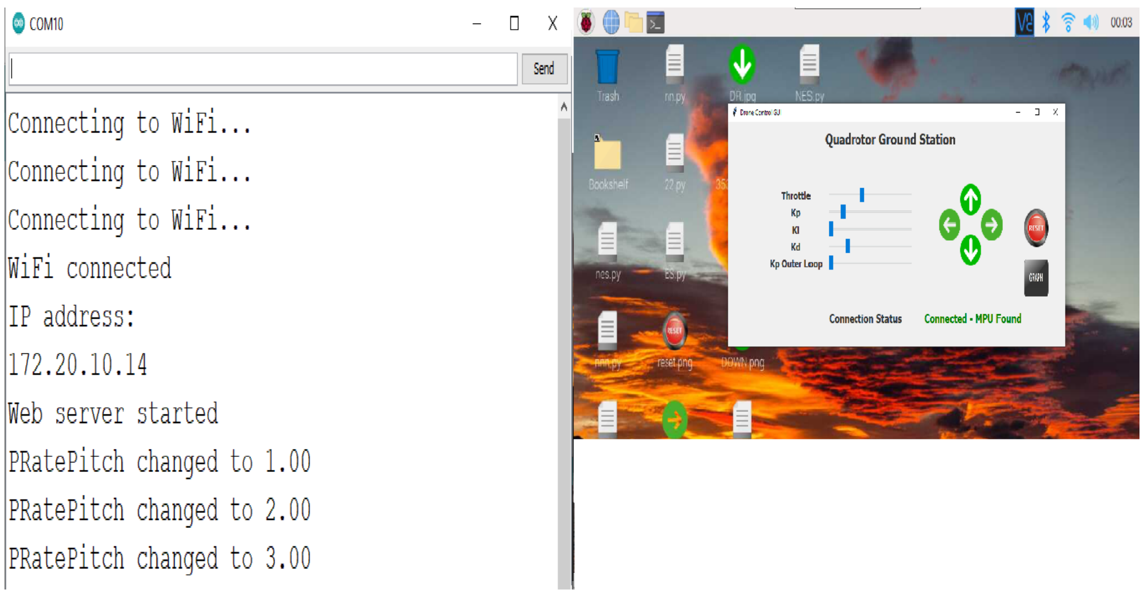


Figure 4.14: Graphical User Interface (GUI) on the Raspberry Pi.

The GUI shown in figure 4.14 of the Raspberry Pi, was programmed using a Python script within the Tonny IDE, which provided a conducive environment for our coding. The GUI includes sliders to adjust the throttle, desired roll, pitch angles, and controller coefficients. Additionally, it includes a button for resetting the flight controller and an interactive button to display the graphs with a simple click to easily analyze the quadrotor's response and behavior.

Communication Protocol

For the communication between the ESP32 and Raspberry Pi, the chosen protocol is HTTP (Hypertext Transfer Protocol). HTTP is a widely used protocol for transmitting data over the internet or a local network [24]. It is based on a client-server architecture, as illustrated in figure 4.15.



```
Connecting to WiFi...
Connecting to WiFi...
Connecting to WiFi...
WiFi connected
IP address:
172.20.10.14
Web server started
PRatePitch changed to 1.00
PRatePitch changed to 2.00
PRatePitch changed to 3.00
```

Figure 4.15: Wireless communication between the ESP32 and Raspberry Pi .

figure 4.15, shows the successful communication between the Raspberry Pi and the ESP32. The HTTP protocol facilitates this communication using the IP address of the flight controller. When modifications are made in the GUI, such as input values, or controller coefficients, those changes are immediately reflected on the serial monitor within the Arduino IDE.

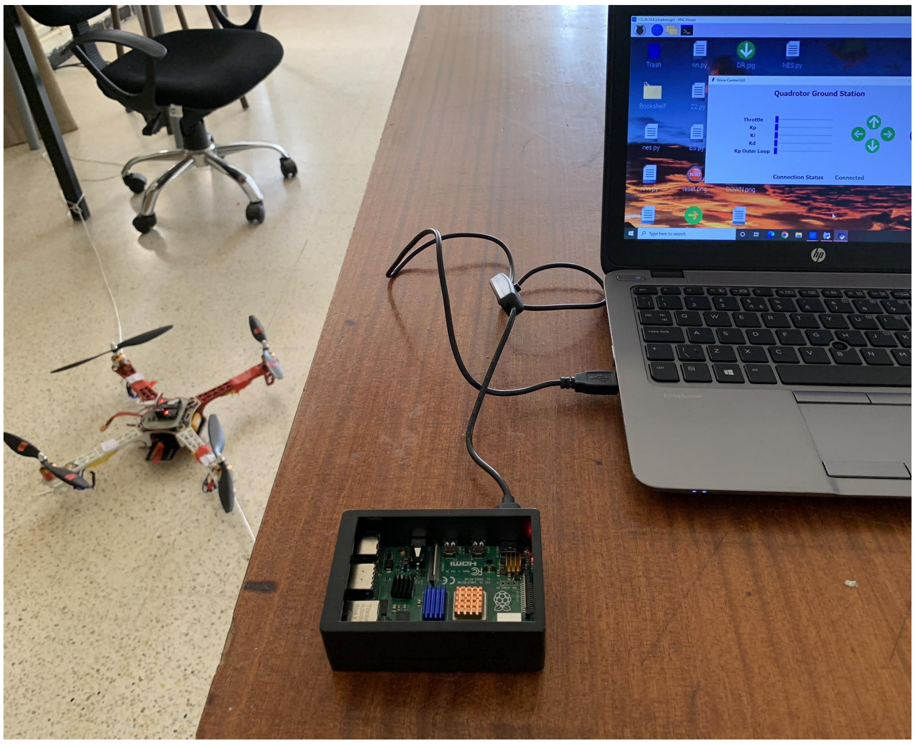


Figure 4.16: Quadrotor flight controller controlled wirelessly with Raspberry Pi GUI.

4.2 Attitude Controller

The attitude controller is a critical component in controlling the orientation of a quadrotor. In this section, we will explore two types of controllers: the PID controller and the backstepping controller. We will discuss the principles behind each controller and provide a comparison of their strengths and weaknesses.

4.2.1 Proportional-Integral-Derivative Controller

PID controller is a linear control system method that consists of three terms controller type which are proportional, integral, and derivative. Each term has a different task in improving the dynamic response of the controlled plant. Generally, the structure of the PID controller is shown in figure 4.17.

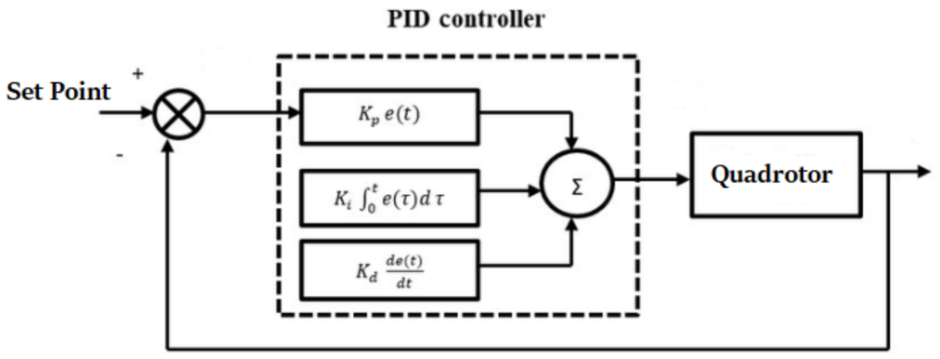


Figure 4.17: The block diagram of the PID controller .

The proportional term provides immediate response, the integral term eliminates steady-state error, and the derivative term enhances stability. Proper selection and tuning of the Kp, Ki, and Kd constants are essential to achieve the desired control performance.

Designing a Dual-Loop Attitude Controller

To design our controller, we need to implement two separate loops: one for controlling the rate and another for controlling the angle.

1. **The Rate Control loop (inner loop):** focuses on regulating the angular velocity. It measures the rate and compares it to the desired rate, generating a control signal to adjust the rate accordingly. This loop helps stabilize the system's dynamics and manage fast changes in angular velocity.

2. **The Angle Control loop (outer loop):** deals with maintaining the desired angles of the system. It measures the current angle and compares it to the desired angle, producing a control signal to steer the system toward the desired angle. This loop ensures the system maintains changes in the desired angle using only 'P' controller. The two loops are shown in 4.18.

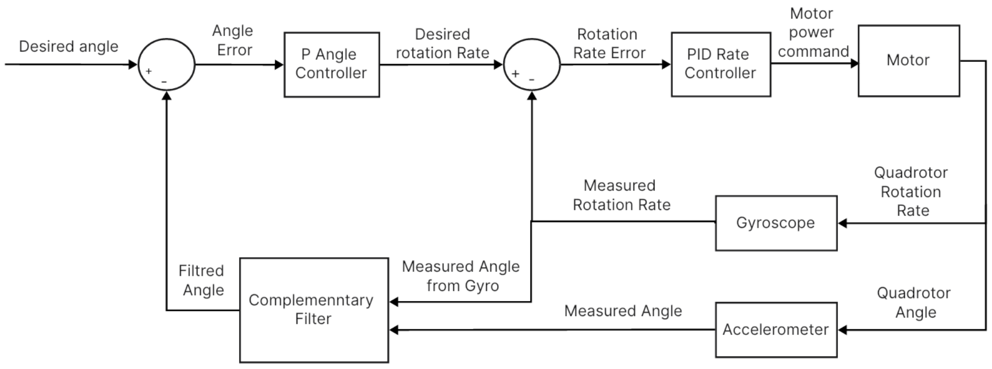


Figure 4.18: Control strategy using PID.

Programming Algorithm

While implementing our quadrotor, we need to design a software algorithm that is formed in the programming language and embedded in the ESP32 microcontroller. In this study, Arduino IDE is used as a compiler with C/C++ programming language. The overall programming structure includes the PID algorithm shown in figure 4.19.

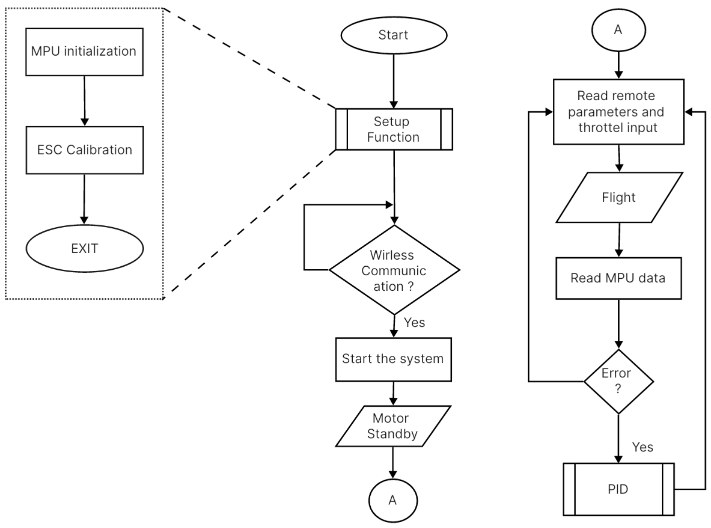


Figure 4.19: Programming structure of attitude controller using PID.

4.2.2 Hardware Test

In this section, we will conduct hardware tests to evaluate its capabilities and its performance.

Test Without propellers

Testing without propellers provides an opportunity to verify and validate the functionality of the quadrotor's software to focus solely on the flight control algorithms, sensor integration, and communication protocols without the influence of physical forces generated by the propellers as illustrated in figure 4.20.

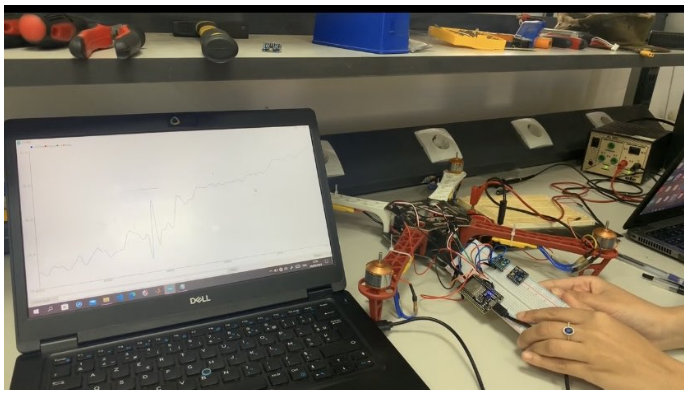


Figure 4.20: Test without propellers.

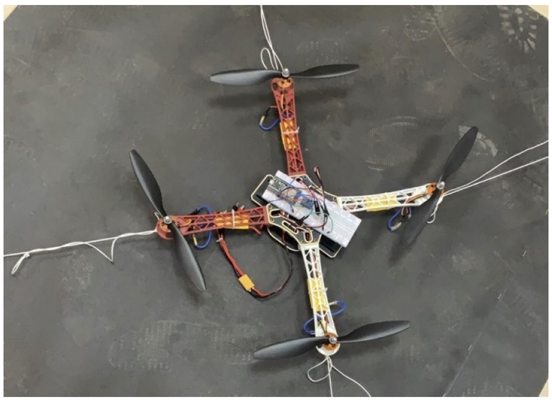
Test With propellers

Testing a quadrotor with propellers is essential to evaluate its actual flight behavior, performance, and stability as illustrated in figure 4.21.

PID Tuning

After numerous attempts and diligent tuning of the PID parameters, we have successfully achieved stabilization of the quadrotor system. The process involved iterative adjustments and fine-tuning of the proportional, integral, and derivative gains to optimize the control response as illustrated in figure 4.22.

Due to the symmetry of quadrotors, it is often possible to set the same gain values for the pitch and roll axes. Since the dynamics and control requirements are similar for these two axes. On the other hand, the yaw axis, may not require as precise control as the pitch and roll axes.


(a) Grounded Quadrotor

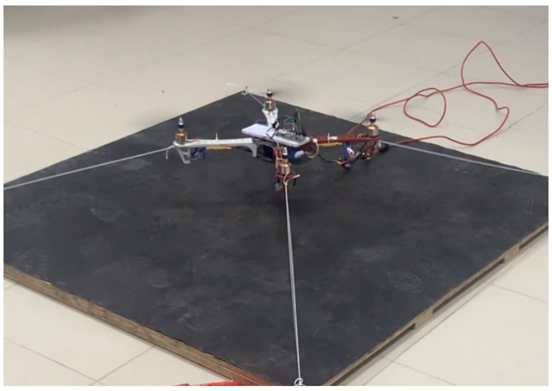
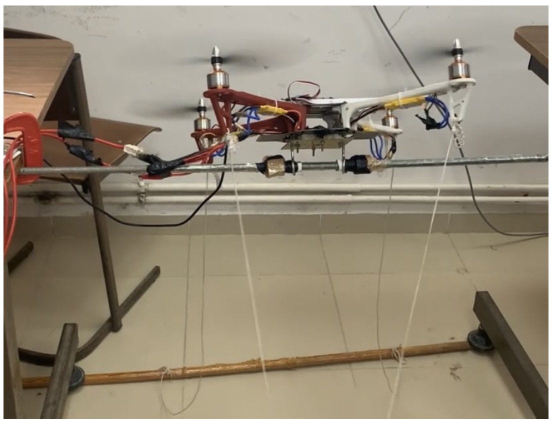

(b) Quadrotor in flight mode

Figure 4.21: Test with propellers.


(a) Not stable behavior of the quadrotor

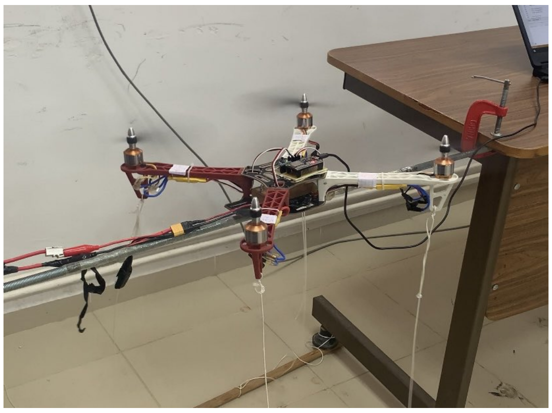

(b) Stable behavior of the quadrotor

Figure 4.22: PID Tunning tests for pitch mouvement.

Table 4.2: PID Parameters for Quadrotor Control unstable state (inner loop).

Axis	**Proportional (Kp)**	**Integral (Ki)**	**Derivative (Kd)**
Roll/Pitch	17	1	2
Yaw	0.4	0.02	0.1

Table 4.3: PID Parameters for Quadrotor Control in a stable state (inner loop).

Axis	**Proportional (Kp)**	**Integral (Ki)**	**Derivative (Kd)**
Roll/Pitch	7.5	1	5
Yaw	0.4	0.02	0.1

Therefore, a Kp coefficient of 3 is chosen for the outer loop, which is applied to all angles. figure 4.23 illustrates the response of the angle pitch with respect to the implementation of a PID controller.

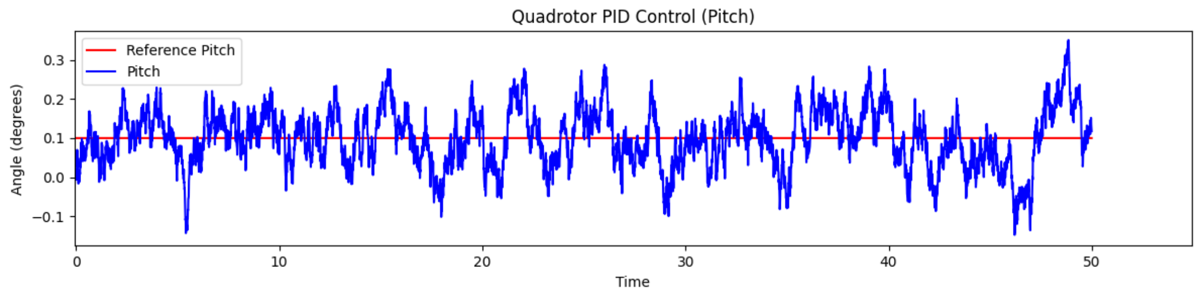


Figure 4.23: Quadrotor angle pitch response with PID controller.

4.2.3 Results and Discussion

During the initial tuning process illustrated in Table 4.2, we observe overshooting and oscillations in the quadrotor's response, which resulted in instability as shown in figure 4.22(a). However, after fine-tuning these parameters demonstrated in Table 4.3, the behavior of the quadrotor became stable as shown in figure 4.22 (b).

By decreasing the proportional gain (Kp), we effectively reduce oscillations in the system which helps to result in smoother movements and behavior. By increasing the derivative gain (Kd), the controller responds faster to changes in the error rate which improves stability.

From figure 4.23, the PID controller effectively stabilizes the angle to the desired 0 degrees. However, it is noticeable that there is some initial overshoot and also transient overshoot during the stabilization process. Thus, Fine-tuning the PID gains required careful adjustment to achieve the desired balance between stability and agility and yielded positive results.

4.3 Back-Stepping Controller

In this section, the objective is to demonstrate the implementation of the attitude controller to our quadrotor prototype using the backstepping technique, which we previously discussed in the System Control chapter. We will put the theoretical concepts into practice by demonstrating how to implement the controller and validate its performance through tests.

4.3.1 Designing Single-Loop Controller

Due to the coupled nature of the equations governing the system's dynamics, such as angular velocities and angles, makes it challenging to separate the controllers into distinct inner and outer loops. Therefore, to effectively handle the interdependencies between these variables, we have opted for a single-loop controller approach. As illustrated in figure 4.24.

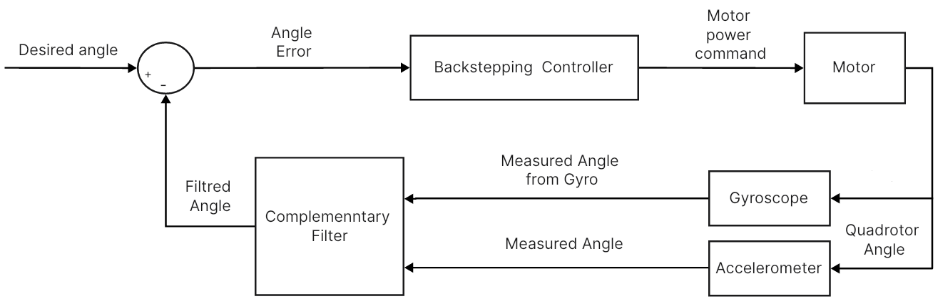


Figure 4.24: Attitude controller using the Backstepping technique.

Programming Algorithm

The flowchart of the overall programming structure including the backstepping Controller algorithm on the ESP32 is shown in figure 4.25.

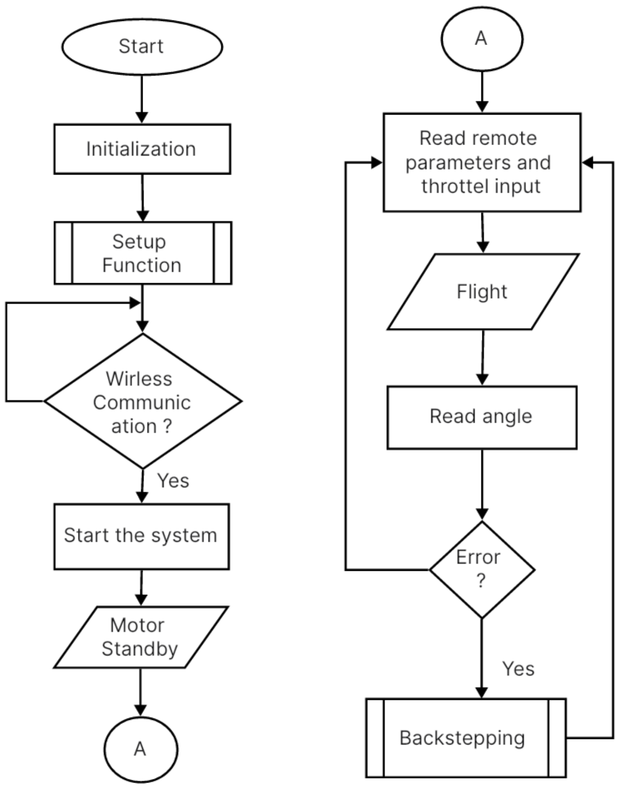


Figure 4.25: Attitude controller using the backstepping technique.

4.3.2 Hardware Test

The implementation of the backstepping controller on the actual hardware platform resulted in notable instability issues. The quadrotor exhibited erratic behavior, including oscillations and difficulty in maintaining stable flight as illustrated in figure 4.26.

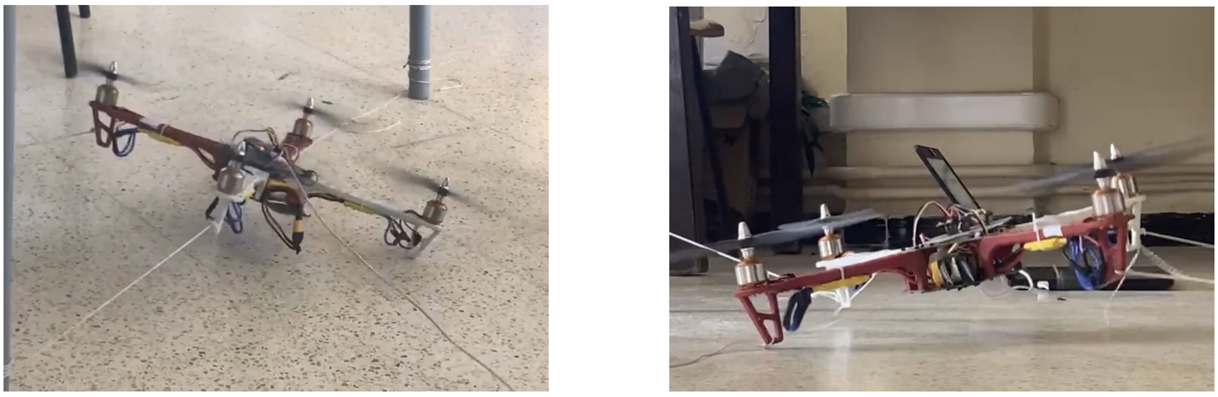


Figure 4.26: Quadrotor Hardware Tests using Backstepping controller.

figure 4.27 illustrates the unstable response of the angle pitch with respect to the implementation of a backstepping controller.

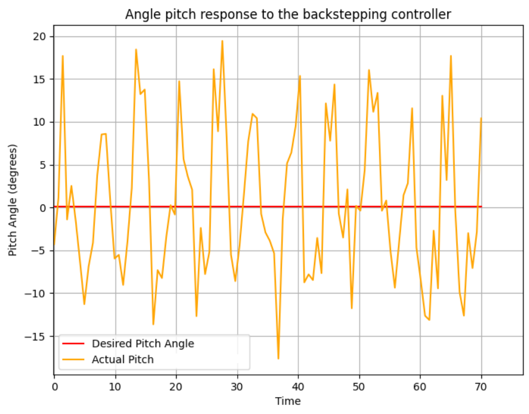


Figure 4.27: Unstable response of pitch angle with Backstepping .

After making numerous adjustments to the gains in the backstepping controller, we have significantly improved the behavior of the quadrotor. It demonstrates enhanced stability, responsiveness, and smooth transitions, as shown in figure 4.28.

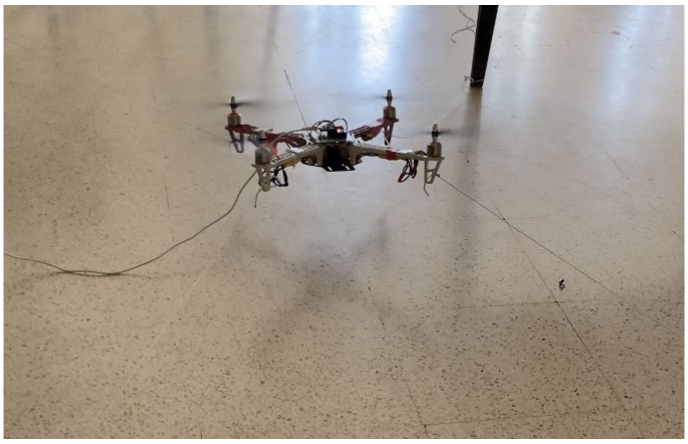


Figure 4.28: Stable behavior using Backstepping controller.

figure 4.29 illustrates the stable response of the angle pitch with respect to the implementation of a backstepping controller.

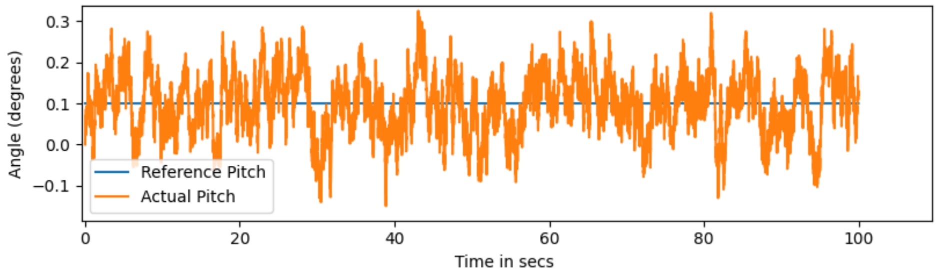


Figure 4.29: Stable response of pitch angle with Backstepping.

4.3.3 Results and Discussion

Based on the results shown in figure 4.26 and figure 4.27, we observe that the quadrotor exhibited unstable behavior. The instability was manifested through oscillations, erratic movements, and an inability to maintain a steady flight.

Insufficient adjustment of the control parameters (improper tuning) was identified as the primary factor contributing to the observed instability in the quadrotor. However, after conducting extensive parameter tuning, we have successfully improved its behavior, as shown in figure 4.28 and figure 4.29. This fine-tuning process entailed adjusting the coefficients of control laws of each subsystem within the backstepping controller.

4.4 Comparision Between PID and Back-Stepping

In this section,the two control approaches: Proportional-Integral-Derivative (PID) and backstepping controllers are compared. Various criteria are examined to evaluate their performance and effectiveness. The following table 4.4 provides a detailed comparison between PID and backstepping.

Table 4.4: Comparison between Backstepping and PID Controllers in Quadrotor

Aspect	**Backstepping Controller**	**PID Controller**
Design Complexity	Requires mathematical model of the system	Requires tuning of controller parameters
Stability	Can guarantee asymptotic stability	May have stability issues if not tuned well
Adaptability	Can handle changes in system dynamics	May require retuning for changes
Implementation	More complex to implement	Relatively easier to implement
Performance Limit	Can achieve higher performance	Performance limited by tuning and model

4.5 Problem Description

During our hardware implementation, several problems were encountered which made reaching our objective harder; some of these problems are discussed in the coming subsections. Nevertheless these obstacles, we have got satisfactory results as was proved previously.

Battery Consumption

The total voltage of the 3-cell is 12.6 volts, indicating a fully charged battery. Therefore, with a 2200mAh battery and constant current consumption of approximately 30A, the estimated flight time is around 4.4 minutes.

During the flight test, we observed a poorer performance from the quadrotor. However, it is worthwhile to utilize a power supply with a range of 30-72A as shown in figure 4.30 for further test experiments.

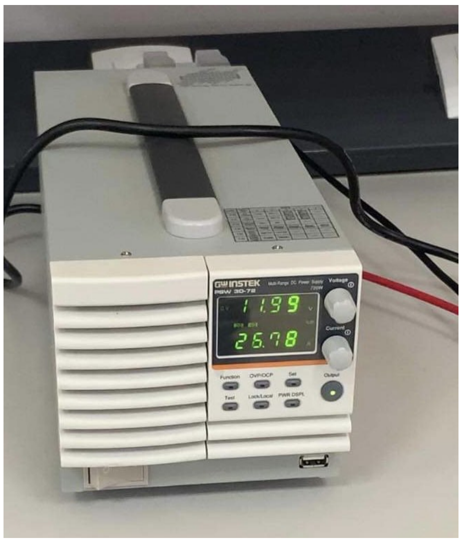
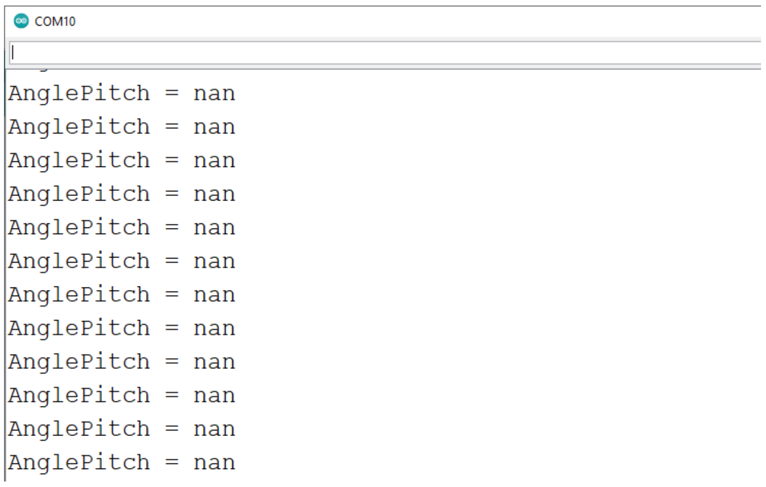


Figure 4.30: 30A-72A Power Supply.

Loss of sensor connection

The loss of the MPU 6050 connection comes from loose wires, a breach in the cable, or a malfunction in the IMU itself and that causes a lot of time to exhibit unpredictable behavior, so to track this problem we printed the outputs of the angles on the serial monitor. figure 4.31 illustrated an example of losing sensor data.



```
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
AnglePitch = nan
```

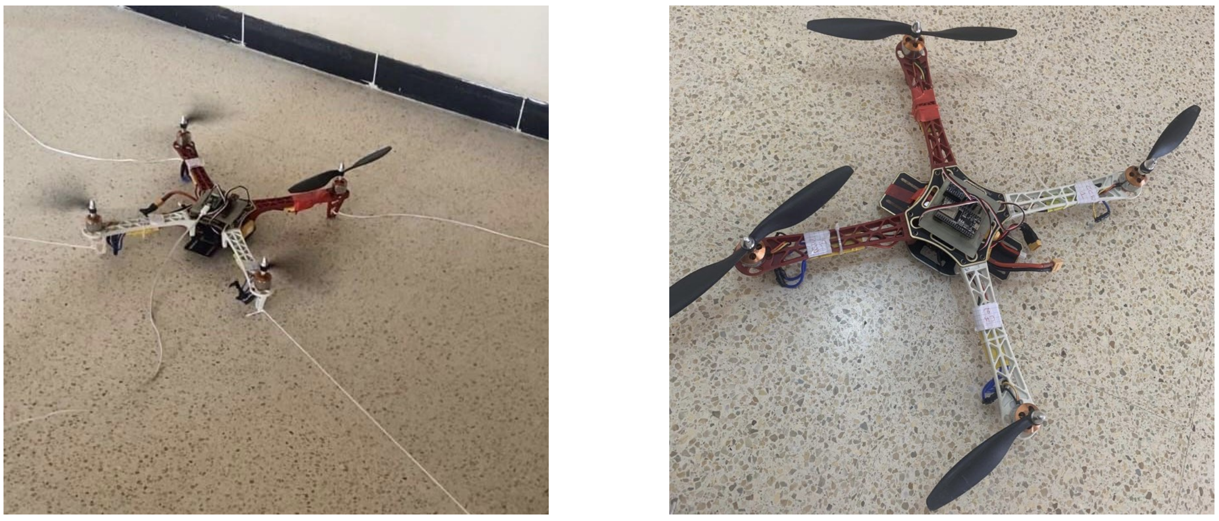
Figure 4.31: Losing Data from MPU6050.

Motor Failures

Overheating and excessive load, cause the motors to experience a decrease in performance or lead to damage, as shown in figure 4.32 (a).

Propeller damage

The propeller damage as demonstrated in figure 4.32 (b), resulted in a delay in our testing schedule.



(a) Motor damage (b) Propeller broken

Figure 4.32: Hardware problems.

4.6 Conclusion

In this chapter, we focused on the hardware implementation of the prototype quadrotor, covering various aspects such as component connections, algorithm development, and software integration. Furthermore, we conducted tests with the prototype to visualize the attitude controller's performance, utilizing a PID and backstepping controller. Our experiments showcased the effectiveness of both controllers in stabilizing the quadrotor in real-world scenarios. Unfortunately, due to safety concerns and potential risks associated with testing, we are unable to test the quadrotor prototype without implementing threads.

General Conclusion

In this project, we explored Unmanned Aerial Vehicles (UAVs) with a specific focus on quadrotors. Our research primarily revolved around three key areas: the utilization of a mathematical model and the simulation of backstepping control algorithms. Moreover, the PID (Proportional-Integral-Derivative) and backstepping controllers are implemented on a quadrotor platform, employing a Raspberry Pi as a ground station.

A mathematical model was used to capture the dynamics of the quadrotor. By understanding the mathematical representation of the system, we were able to develop control strategies that could effectively align with these mathematical models. During the simulations, we observed that the backstepping controller performed exceptionally well. It led to stable flight and precise tracking of the desired trajectory.

Regarding hardware implementation, we successfully implemented a quadrotor prototype to visualize both the BSC and PID control algorithms. Despite the achieved stability through those control approaches, the quadrotor system may still experience some instability under specific conditions. This instability can be attributed to various factors, including suboptimal tuning of the control parameters, inaccuracies in the sensor measurements, or limitations in the hardware components.

Furthermore, the theory and simulation of the backstepping controller have demonstrated exceptional performance in achieving precise orientation. However, The implementation can be time-consuming due to its intricacies and complexities.

From the project, we gained a profound understanding of quadrotors, from theoretical modeling to simulation and practical implementation. We witnessed the translation of theoretical concepts into tangible outcomes and evaluated the strengths and limitations of different control approaches. This project enhanced our knowledge of UAV dynamics and control and also equipped us with the practical skills required to develop UAV systems.

Future Work and Perspectives

The quadrotor possesses countless features when it comes to functional capabilities. Here are some potential expansions that could be considered in the future:

- Sensor Fusion such as Kalman filter can provide more accurate and drift-free attitude estimation.
- Autonomous Navigation and trajectory tracking by using GPS, cameras, lidar, and ultrasonic sensors to enable the quadrotor to perceive its environment, plan its trajectory, and execute its flight path.
- Object Detection by detecting and tracking specific objects of interest within an image or video using computer vision.

References

[1] Anuj Puri. “A survey of unmanned aerial vehicles (UAV) for traffic surveillance”. In: *Department of computer science and Engineering, University of South Florida* (2005), pp. 1–29.

[2] Kennedy Martinez. “The History of Drones (Drone History Timeline from 1849 To 2019)”. In: *Drone Ethusiast* (2018).

[3] de Nansouty-M De Nansouty-M. “Aérostation, Aviation Savoirs Et Traditions”. In: *HACHETTE LIVRE* (1914), p. 774.

[4] Erik Haywood Stoer. *Flying bombs, aerial torpedoes, and Kettering bugs: America’s first cruise missiles.* The Florida State University, 2001.

[5] Astronomy Encyclopedia of Astrobiology and Space Flight. “http://www.daviddarling.info/encyclopedia”. In: (25 APRIL 2023).

[6] https://commons.wikimedia.org/. “Winston Churchill and the Secretary of State for War waiting to see the launch of a de Havilland Queen Bee radio-controlled target drone”. In: (6 June 1941).

[7] Paul Posea. “The Complete History of Drones (1898-2023) - INFOGRAPHIC”. In: (2023).

[8] L Zitzman. “Drones in Construction: How They’re Transforming the Industry. Retrieved August 2, 2019”. In: (2018).

[9] Matúš Tkáč and Peter Mésároš. “Utilizing drone technology in civil engineering”. In: *Selected Scientific Papers-Journal of Civil Engineering* 14.1 (2019), pp. 27–37.

[10] Mohamed Amine Kafi et al. “A study of wireless sensor networks for urban traffic monitoring: applications and architectures”. In: *Procedia computer science* 19 (2013), pp. 617–626.

[11] Jack Brown. *https://www.mydronelab.com/blog/drone-uses.html*. 2021.

[12] Ann Cavoukian. *Privacy and drones: Unmanned aerial vehicles*. Information and Privacy Commissioner of Ontario, Canada Ontario, 2012.

[13] Saad Nazarudeen. “Conceptual Design of High Altitude Long Endurance Solar Powered UAV”. In: (2018).

[14] Sharon L Conwell et al. “Evolution of Human Systems Integration for Remotely Piloted Aircraft Systems”. In: *Remotely Piloted Aircraft Systems: A Human Systems Integration Perspective: A Human Systems Integration Perspective* (2016), pp. 15–39.

[15] Kasim Biber. “Aerodynamic analysis of a medium-altitude, long-endurance unmanned aircraft”. In: AIAC-2015-062, 8th Ankara International Aerospace Conference, 10–12 … 2015.

[16] K Raja Sekar et al. “Aerodynamic design and structural optimization of a wing for an Unmanned Aerial Vehicle (UAV)”. In: *IOP Conference Series: Materials Science and Engineering*. Vol. 764. 1. IOP Publishing. 2020, p. 012058.

[17] Pimonrat Tiansawat and Stephen Elliott. *Unmanned aerial vehicles for automated forest restoration*. 2020.

[18] “https://www.directindustry.com/prod/uav-america/product-182005-2113873.html”. In: (21/05/2023).

[19] Hakan Üçgün et al. “Graphical Interface Design and Implementation for Flight Tests of the Rotary Wing Unmanned Aerial Vehicles”. In: (2016).

[20] Aminurrashid Noordin, Mohd Ariffanan Mohd Basri, and Zaharuddin Mohamed. “Sensor fusion for attitude estimation and PID control of quadrotor UAV”. In: *International Journal of Electrical and Electronic Engineering and Telecommunications* 7.4 (2018), pp. 183–189.

[21] Hashim A Hashim, Lyndon J Brown, and Kenneth McIsaac. “Nonlinear explicit stochastic attitude filter on SO (3)”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 1210–1216.

[22] Hatem M Kandeel, Ebrahim A Abdelmaksod, and Abdelrady Okasha Elnady. “Modeling and Control of X-Shape Quadcopter”. In: ().

[23] Herman Høifødt. "Dynamic Modeling and Simulation of Robot Manipulators: The Newton-Euler Formulation". In: 2011.

[24] Amrouche Hafid;Recham Zine Eddine. "Adaptive control for disturbance rejection in quadrotors". In: 2021.

[25] Zongyu Zuo. "Trajectory tracking control design with command-filtered compensation for a quadrotor". In: *IET control theory & applications* 4.11 (2010), pp. 2343–2355.

[26] Eduardo Luís Neto Santos. "Sleep Bruxism Detection Device". PhD thesis. 2020.

[27] *ESP32 DevKitC Core Board ESP32 Development*. Alibaba. Accessed on June 10, 2023. URL: `%7Bhttps://arabic.alibaba.com/product-detail/ESP32-DevKitC-core-board-ESP32-development-62183009476.html%7D`.

[28] Basit Qureshi and Anis Koubâa. "On energy efficiency and performance evaluation of single board computer based clusters: A hadoop case study". In: *Electronics* 8.2 (2019), p. 182.

[29] *Raspberry Pi 4 Model B 2GB (BCM2711)*. e-komponent. Accessed on June 10, 2023. URL: `%7Bhttps://www.e-komponent.com/raspberry-pi-4-model-b-2gb-bcm2711%7D`.

[30] InvenSense, Inc. *MPU-6000 Register Map*. InvenSense. 2015. URL: `https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf`.

Appendices

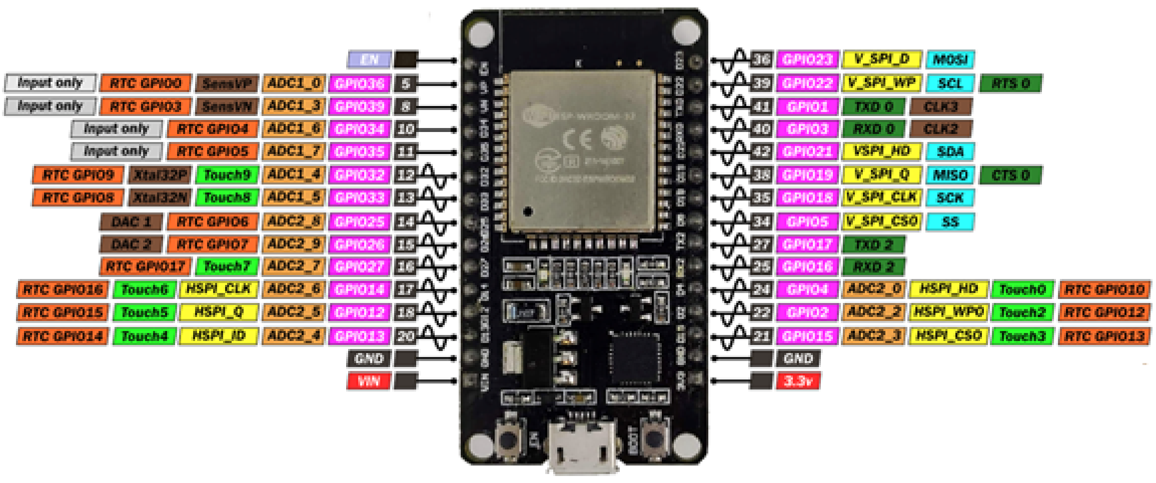
Project Cost

In Table 0.5, we present the cost breakdown for our quadrotor project. Some of the materials had to be repurchased due to damage and breakage.

Table 0.5: Cost of Quadrotor Components.

Component	**Cost (DZD)**
Frame	6000.00
Motors	2200.00 (each)
Electronic Speed Controllers (ESCs)	2200.00 (each)
ESP32	2950.00
Propellers	800.00 (set)
LIPO Battery	8000.00
MPU6050	500.00
Battery charger	8500.00
Raspberry PI 4	40000.00
Total Cost	9 3850.00

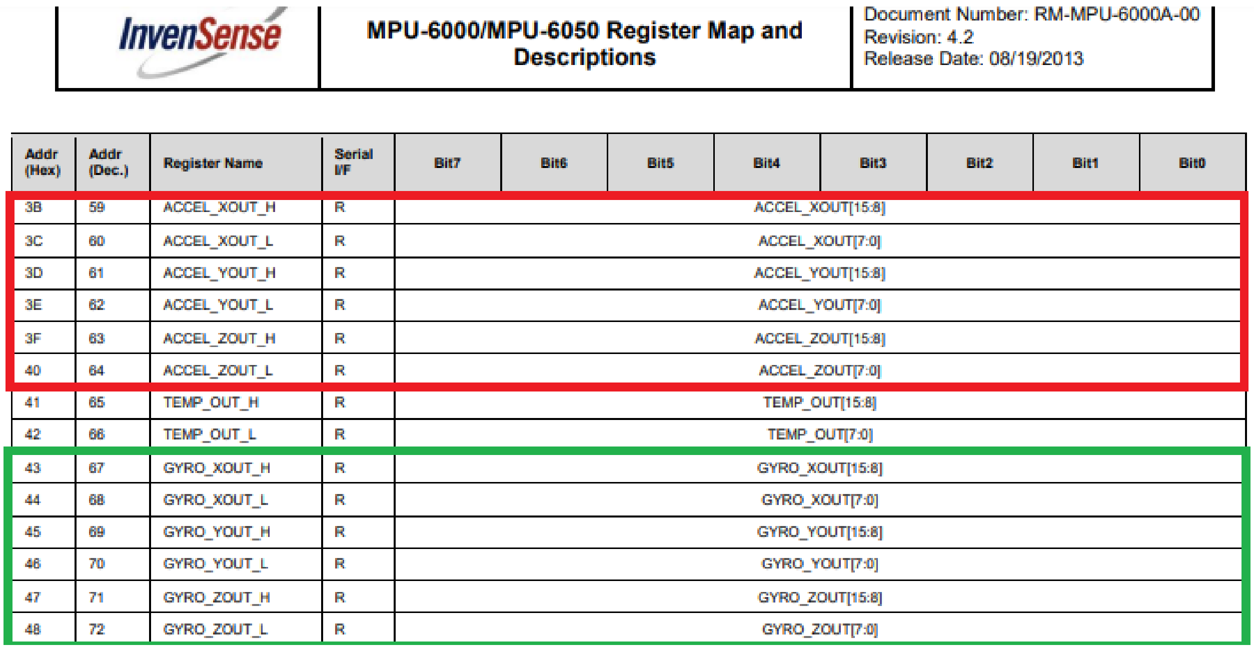
ESP32 Pins chart



EN
Input only RTC GPIO0 SensVP ADC1_0 GPIO36 5
Input only RTC GPIO3 SensVN ADC1_3 GPIO39 8
Input only RTC GPIO4 ADC1_6 GPIO34 10
Input only RTC GPIO5 ADC1_7 GPIO35 11
RTC GPIO9 Xtal32P Touch9 ADC1_4 GPIO32 12
RTC GPIO8 Xtal32N Touch8 ADC1_5 GPIO33 13
DAC 1 RTC GPIO6 ADC2_8 GPIO25 14
DAC 2 RTC GPIO7 ADC2_9 GPIO26 15
RTC GPIO17 Touch7 ADC2_7 GPIO27 16
RTC GPIO16 Touch6 HSPI_CLK ADC2_6 GPIO14 17
RTC GPIO15 Touch5 HSPI_Q ADC2_5 GPIO12 18
RTC GPIO14 Touch4 HSPI_ID ADC2_4 GPIO13 20
GND
VIN
36 GPIO23 V_SPI_D MOSI
39 GPIO22 V_SPI_WP SCL RTS 0
41 GPIO1 TXD 0 CLK3
40 GPIO3 RXD 0 CLK2
42 GPIO21 VSPI_HD SDA
38 GPIO19 V_SPI_Q MISO CTS 0
35 GPIO18 V_SPI_CLK SCK
34 GPIO5 V_SPI_CS0 SS
27 GPIO17 TXD 2
25 GPIO16 RXD 2
24 GPIO4 ADC2_0 HSPI_HD Touch0 RTC GPIO10
22 GPIO2 ADC2_2 HSPI_WP0 Touch2 RTC GPIO12
21 GPIO15 ADC2_3 HSPI_CS0 Touch3 RTC GPIO13
GND
3.3v

Figure 0.33: ESP32 Pins chart [27].

MPU Registers



InvenSense

MPU-6000/MPU-6050 Register Map and Descriptions

Document Number: RM-MPU-6000A-00
Revision: 4.2
Release Date: 08/19/2013

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							

Figure 0.34: Mpu registers [30].