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

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering
Department of fundamental teaching

Project Report Presented in Partial Fulfilment of

the Requirements of the Degree of

'Bachelor

In Electrical and Electronic Engineering

Title:

Quadcopter design using PID Controller

Presented By:

- **Mr. DAHMANE Mounir**

Supervisor:

Dr. Touzout W

Spring 2024

Abstract

Drones and *Quadcopters* have revolutionised flight. They help humans to take to the air in new, profound ways. Today they come with mind-blowing capabilities, flight being the least of these.

A quadcopter is an *under-actuated system* because it uses four actuators to control *six* DoF (degrees of freedom), it is also considered as highly coupled and nonlinear system. Therefore, the design of a controller for the stabilization of the quad-copter during vertical flight represents a real challenge.

This Project aim to design and specifically implement The **PID** Control to stabilize and control the quadcopter drone since it is the most common control approach used in industrial and commercial mechatronics products. In addition we investigates its efficiency during the tuning process and how its parameters affect the system stability performance.

In this work a custom PID controller was implemented using C/C++, with **ESP32** Micro-controller as the main control unit, using its built-in **WIFI** module (ie.**TCP**) to communicate with the user's device. The MPU6050 senor represents the IMU, to get the 3 different angles in the 3D axis. The manual tuning was performed based on theoretical background to get the suitable values.

The final results were quite good, because the last setup of the Hardware and Software stabilizes the system, where we can conclude that the reputation of this controller approach is well earned, but i would like to mention that it is preferable to use a powerfully micro-controller in terms of processing and computing like : **STM32**, since the esp32 gave some throwbacks issues.

Acknowledgments

Alhamdulillah for the strength and the faith he spreads in my heart to achieve and accomplish this humble work. I would like to express my gratitude for my supervisor Dr.TOUZOUT Walid for accepting my report proposal as well as my project, in addition for guiding me throughout this journey with his valuable insights, support, and mentorship.

A special thanks to my family and both of my friends Hamidi Sidahmed and Dahmani Abdelmadjid, because of their encouragement and belief in me have been instrumental in my accomplishment, and i will not forget Wameedh Scientific Club for providing me the non-available material, Thank you.

Contents

List of Figures

List of Tables

Symbols and Abbreviations

PID Proportional-Derivative-Integral

TCP Transmission Control Protocol

SoC System on chip

IoT Internet of things

GPIO General Purpose input output

IDE integrated developement envirement

DOF Degrees of Freedom

TCP Transmission Control Protocol

Val Value

UAV Unmanned aerial vehicle

IMU Inertial Measurement Unit

ESC Electronic speed controller

LiPo Lithium Polymer

CW Clock-wise

CCW Counter clock-wise

$S_{\mathbf{x}}$ $Sin(x)$

$C_{\mathbf{x}}$ $cos(x)$

CoM Center of mass

RPM Revolutions per minute

ϕ, θ, ψ Roll, pitch and yaw

DMP Digital Motion Processor

PWM Pulse Width Modulation

µC Microcontroller

SDA Serial Data

SCL Serail Clock

BLDC Brushless DC motors

Introduction

Recently, UAVs have gained increasing attention from researchers; most notably the quadcopters configuration due to their low cost, mechanical simplicity, maneuverability and the ability to handle multiple tasks such as : film-making, search and rescue, surveillance, agriculture, ...etc. Unlike other aerial vehicles, quadcopters generate lift using a set of four rotors and variate their RPM to achieve vehicle control in three dimensions. However, the downside is that quadcopters are highly non-linear unstable systems that possess six DOF and four input variables, which classifies them as under-actuated systems. Also, another difficulty is developing the mathematical model, since many non-linear variables should be taken into account. Thus, developing a controller for quadcopter is very interesting research topics.

Quadcopter control can be divided into attitude, altitude and position control. The purpose of attitude control is to command the three angles of the quadcopter to guarantee horizontal leveling and a stable vertical flight. Altitude control aims to keep the vehicle hovering at the desired attitude. Both of the above are part of the "inner loop". Position control, also referred to as the "outer loop" is about controlling the coordinates of the quadcopter in three dimensions. Since the rotational and transnational dynamics are coupled, a change in attitude is needed to reach the desired position. Thus, this controller outputs the desired angles which are then fed back to the inner loop.

The aim of this project is to implement a suitable controller for the quadcopter design to ensure a stable vertical flight using ESP32 micro-controller by Espressif systems (https://www.espressif.com/en/products/socs/esp32). For that purpose, a PID controller was

chosen; despite its simplicity, it provides satisfactory results in practical implementations.

Chapter one of this thesis starts off by presenting an introduction to quadcopters. Then, it describes the mathematical model of the quadcopter based on Newton-Euler approach. This chapter also provides a basic overview of control systems in addition to an introduction to the used protocol of communications. Chapter two provides a detailed description of the hardware components used as well as the electronic circuit schematic of the quadcopter control system. The software implementation, containing all libraries and functions used for controlling the quadcopter, is covered in chapter three. Finally, chapter four discusses the results concerning the flight tests conducted on the quadcopter platform, and presents ideas for future work.

Chapter 1

Theoretical Background

In this chapter, the theory behind this project will be explored, starting from the Drone theory and modeling, control systems and the PID controller, finally the protocol of communications used in the project's enviroment. A quadcopter is an under-actuated aircraft with fixed pitch angle, four rotors. A typical quadcopter have four rotors with fixed angles and they represents the four input forces, which are basically the thrust provided by each propellers. There are two possible configurations for most of quadcopter designs + and **X** as shown in figure 1.1, the difference between them is which rotors to control in order to change the orientation of the quadcopter.

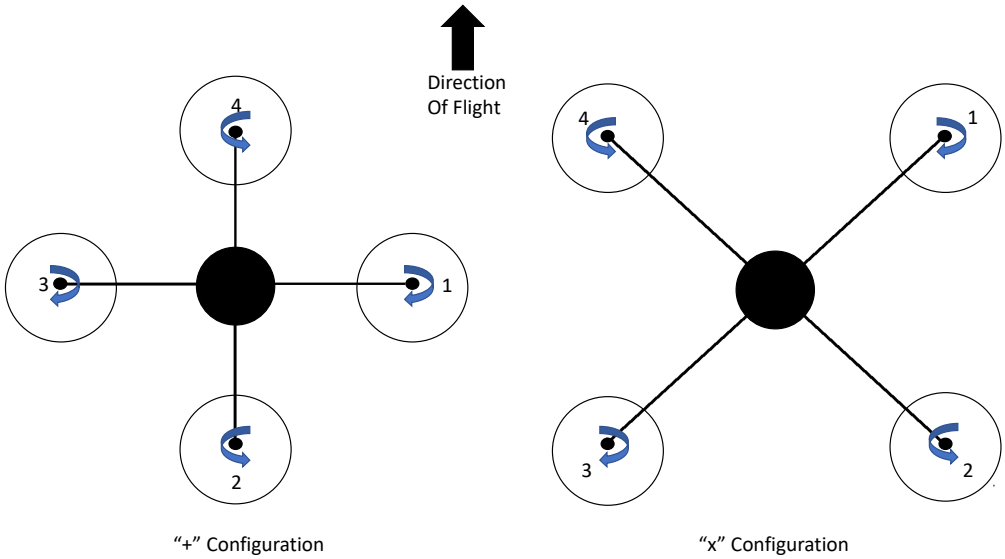


Figure 1.1: + and X Configurations

The X-configuration quadcopter is considered to be more stable compared to + configuration, which is a more acrobatic configuration, for which it was the configuration considered to be used in this project [1].

Propellers 1 and 3 rotates CW, and 2 and 4 rotates CCW , to cancel out the torque and maintain forward/backward motion by controlling the speed of front/rear rotors speed. This process is required to compensate the action/reaction effect (Third Newton's Law).

There are two reference systems that have to be defined as a reference which are **Inertial** reference system (Earth frame- XE, YE, ZE) and **quadrotor** reference system (Body frame- XB, YB, ZB). The reference system frames are shown in Figure 1.2. The dynamics of quad-copter can be describe in many different ways such as quaternion, Euler angle and direction matrix. However, in designing attitude stabilization control reference in axis angle is needed, so the designed controller can achieve a stable flight. In attitude stabilization control, all angle references in each axis must be approximately zero especially when take-off, landing or hover. It ensures that, the quadcopter body always is in horizontal state, when external forces are applied on it. The quadcopter orientation can be defined by three Euler angles which are roll angle (ϕ), pitch angle (θ) and yaw angle (ψ). The position of the quadcopter is defined in the inertial frame x, y, z axes with ξ.

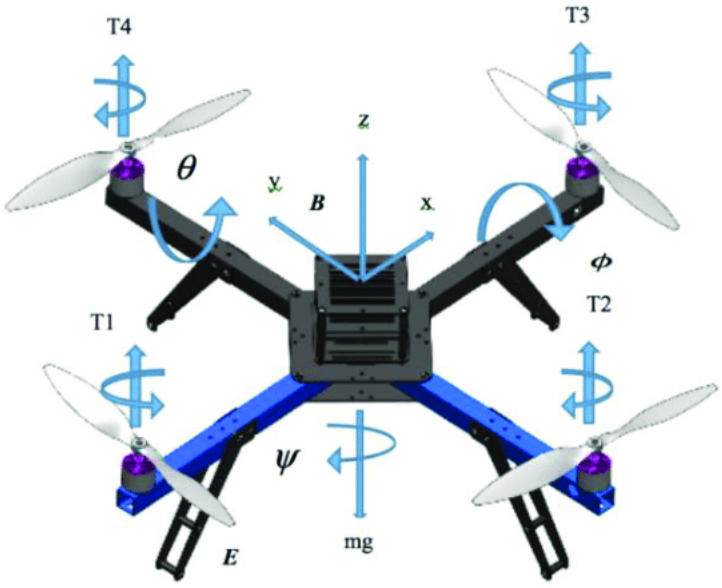


Figure 1.2: Forces, moments and reference systems of a quadcopter.

On Figure 1.2, $\omega 1$, $\omega 2$, $\omega 3$, $\omega 4$ rotation speeds (angular velocity) of the propellers $T1$, $T2$, $T3$, $T4$: forces generated by the propellers; F_i α ω_i^2 on the basis of propeller shape, air density, etc.; m : the mass of the quad-copter; mg: the weight of the quad-copter; ϕ, θ, ψ : roll, pitch and yaw angels. As seen in Figure 1.2, one pair of diagonally opposing rotors (T1, T3) spins in the counterclockwise direction, while the other pair (T2, T4) spins in the clockwise direction. This setup allows the manipulation of the thrust and the roll, pitch and yaw angles of the quadcopters independently of each other. Each of the four rotors produces a lift force in the positive z direction. If all rotors spin at the same speed, they would produce the same torques, which will cancel out due to the setup of the rotors. Since the net torque produced is zero, according to Newton's third law the anti-torque is also zero, which means the quadcopter would not rotate around the z-axis ie. zero yaw angle. In case the net lift force is greater than gravity, the quadcopter would rise. If the rotors generate a net lift force less than gravity, the body would experience a vertical descent. If the net lift force is equal to the gravity, the body would hover.

A rotation around the x-axis is determined by the roll angle ϕ and is achieved by increasing (or decreasing) the speed of the right rotors (T3, T4) and decreasing (or increasing) that of the left rotors (T1, T2). A rotation around the y-axis is determined by the pitch angle θ and is achieved by increasing (or decreasing) the speed of the front rotors (T1, T4) and decreasing (or increasing) that of the rear rotors (M3, M2). A clockwise rotation around the z-axis is determined by the yaw angle ψ and is achieved by increasing the speed of the pair of opposing rotors (T2, T4) and decreasing that of the other pair (T1, T3). The CCW rotation is achieved by reversing this process [2]. The process is summarized in Figure 1.3.

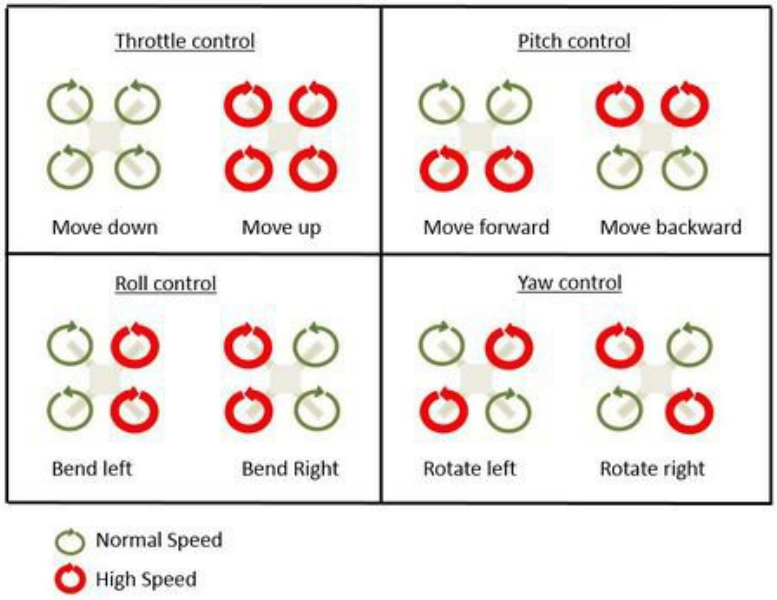


Figure 1.3: Quadcopter movements

1.1 Drone and their applications

The applications of drones cover a wide range of civil and military applications. Drones can perform both outdoor and indoor missions in very challenging environments. Also Drones can be equipped with various sensors and cameras for doing intelligence, surveillance, and reconnaissance missions. The applications of drones can be categorized in different ways. It can be based on the type of missions (military/civil), type of the flight zones (outdoor/indoor), and type of the environments [3].

Figure 1.4: Quadcopter civil application

Figure 1.5: Quadcopter military application

1.2 Quad-copter Kinematics

The two reference systems defined, are to be used to analyze the kinematics of the quadcopter. The quadcopter is a 6 DOF, it is described by twelve states. The first six states are the attitude (angular position) and its change. These are ϕ, θ, ψ (in inertial frame) and p, q, r (in body frame). The remaining six states are position x, y, z (in inertial frame) and the speeds u, v, w (in body frame).

Pitch angle θ determines the rotation of the quadcopter around the y-axis. Roll angle ϕ determines the rotation around the x-axis and yaw angle ψ around the z axis.

The following quantities are defined :

$$\text{The linear position vector : } \xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1.1}$$

$$\text{The angular position vector : } \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{1.2}$$

$$\text{The linear and angular position vectors : } q = \begin{bmatrix} \xi \\ \eta \end{bmatrix} \tag{1.3}$$

$$\text{The linear velocity vector : } v = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{1.4}$$

The origin of the body reference (body frame) is in the center of mass of the quadcopter. In the body frame, the linear velocities are determined by JB and the angular velocities by Ω :

$$JB = \begin{bmatrix} Jx & B \\ Jy & B \\ Jz & B \end{bmatrix}, \Omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{1.5}$$

Due to the existence of two reference systems, a rotation matrix R is defined for body frame to inertial frame transformation, which by

matrix multiplication transfers the velocity vector from one reference frame to another. The rotation over x, y and z axes are defined respectively as with S_x and C_x representing $sin(x)$ and $cos(x)$ respectively :

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix}, R_y = \begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & C_\theta \end{bmatrix}, R_z = \begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.6}$$

Then the consequent rotations around $Z \to Y \to X$ axes can be described by the multiplication of the above matrices, which results the bellow rotation matrix :

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + S_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \tag{1.7}$$

and the following matrices shows the relation between the different quantities :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R * \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & S_\phi C_\theta \\ 0 & -S_\phi & C_\phi C_\theta \end{bmatrix} * \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.8}$$

When ψ and θ are close to 0, which corresponds to the quadcopter being close to hover, The second matrix is approximately a unit matrix. In this case the relation between the angles and angular rates can be approximated as linear :

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.9}$$

1.3 Quad-copter Dynamics

- The quadcopter has a rigid physical structure and its center of gravity coincides with the body fxed frame origin.
- The Earth gravitational feld, mass of the quadcopter and body inertia matrix of the quadcopter are constants.

- Lift factor and drag factor of the motor are constants.
- The propellers are supposed to be rigid and aerodynamics of blade flapping are ignored as well as ground effects.
- The gyroscopic effect of the propellers has been neglected. The lift force generated by each rotor is proportional to the square of its angular rate :

$$F_i = K * w_i^2 \tag{1.10}$$

Where K is the lift coefficient. The total Thrust given by the rotors is given by :

$$F_t = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ K * (w_1)^2 + (w_2)^2 + (w_3)^2 + (w_4)^2 \end{bmatrix} \tag{1.11}$$

The moments acting on the quadcopter cause roll, pitch and yaw movements:

- The rolling moment is the result of the difference in thrust between rotors 3,4 and rotors 1,2 as :

$$M_\phi = L * (-F_1 - F_2 + F_3 + F_4) \tag{1.12}$$

- The pitching moment is the result of the difference in thrust between rotors 2,3 and rotors 1,4 as :

$$M_\theta = L * (F_1 - F_2 - F_3 + F_4) \tag{1.13}$$

- The yawing moment is the result of the difference in thrust between the diagonal pairs of rotors as :

$$M_\psi = D * (w_1^2 - w_2^2 + w_3^2 - w_4^2) \tag{1.14}$$

Where D is the drag coefficient, L is the length between the rotors and the CoM. Thus, the total moments are given by:

$$M_t = \begin{bmatrix} M_\phi \\ M_\theta \\ M_\psi \end{bmatrix} = \begin{bmatrix} L * K * (-w_1^2 - w_2^2 + w_3^2 + w_4^2) \\ L * K * (w_1^2 - w_2^2 - w_3^2 + w_4^2) \\ D * (w_1^2 - w_2^2 + w_3^2 - w_4^2) \end{bmatrix} \tag{1.15}$$

As discussed previously, any movement of the quadcopter can be achieved by combining the four basic movements, thrust, roll, pitch,

and yaw. Therefore, the input vector can be defined as :

$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K * (w_1^2 + w_2^2 + w_3^2 + w_4^2) \\ L * K * (-w_1^2 - w_2^2 + w_3^2 + w_4^2) \\ L * K * (w_1^2 - w_2^2 - w_3^2 + w_4^2) \\ D * (w_1^2 - w_2^2 + w_3^2 - w_4^2) \end{bmatrix} \tag{1.16}$$

Where :

-u_1 is the throttle.

-u_2 is the rolling moment.

-u_3 is the pitching moment.

-u_4 is the yawing moment.

-The Newton-Euler equations are defined as :

$$m * \dddot{\xi} = R * F_t * F_g \tag{1.17}$$

$$M_t = \dot{\Omega} + \Omega * I + \Omega \tag{1.18}$$

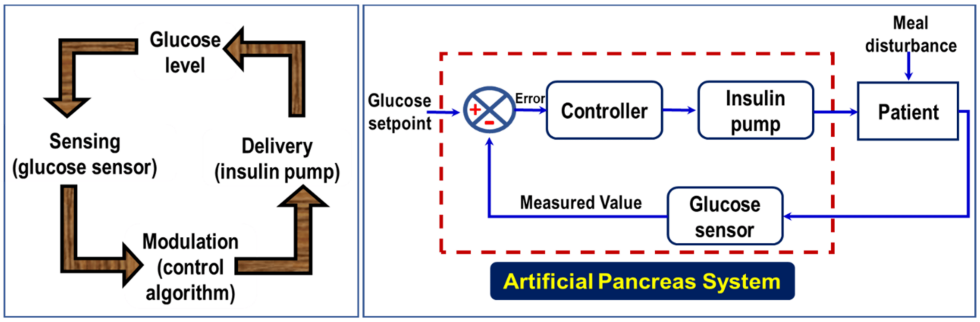
-Where I is the moment of inertia tensor, it consists of moments of inertia Ixx, Iyy, Izz on the diagonal, and products of inertia on the off-diagonal. However, since the quadcopter is symmetrical, the off-diagonal elements are zero and the moment of inertia is written as :

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & Iyy & 0 \\ 0 & 0 & Izz \end{bmatrix} \tag{1.19}$$

1.4 Control Systems and PID controller

A control system is a set of mechanical or electronic devices that regulates other devices or systems by way of control loops. Typically, control systems are computerized.

Control systems exists everywhere, nature industries, and even a human body, for instance the pancreas as shown as Figure 1.6; also they are a central part of production and distribution in many industries. Automation technology plays a big role in these systems. The types of control loops that regulate these processes include industrial control systems, such as supervisory control and data acquisition, systems and distributed control systems.



(a) (b)

Figure 1.6: The control system of the pancreas

A typical control system like the one used for this project, as shown in Figure 1.7 consists of plant or the system we want to control ,or the system whose behavior we want to affect, the input into the plant is the actuated signal and the output is the controlled variable which different industries may refer to these signals as the plant input/output, beside the commanded variable or the setpoint which is what we want the system to do. In feedback control, the output of the system is fed back and compared to the command to see how far off the system is from where we want it to be, and the difference between the two is the error term, if the output is exactly what we commanded it to be then the error would go to zero and that is what we want, zero error, and we take this error term and convert it into suitable actuator, so that over time the error is driven to zero, and that using a controller.

In theory, control systems are further divided into linear and non-linear systems. However, all real-life systems are non-linear. The quad-copter is a prime example of a highly non-linear and unstable system

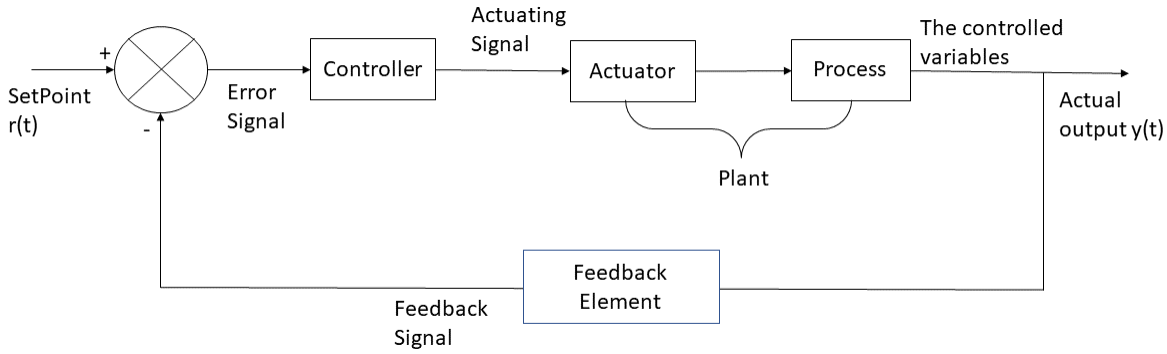


Figure 1.7: closed loop configuration of a control system

that must be stabilized using an adequate controller, for which we introduce the PID controller.

The PID controller which is still wildly used in the industry due to its simplicity, applicability and functionality. That is why it was chosen for this project.

PID stands for Proportional-Integral-Derivative, and it is the most common form of controllers. It is an equation used by the controller to assess the controlled variables based on the value of the error between the feedback signal and the input signal. The error value passes through one or more of the PID paths and the controller issues a command to correct the error.

1.4.1 Proportional path

The proportional path generates a command that is proportional to the error value, so it can be defined as :

$$P = K_P * e(t) \tag{1.20}$$

-K_P is the proportional gain

-$e(t)$ is the error value : $e(t)$ = setpoint - actual output

It can be seen from the above equation that the larger the error value, the larger the command issued by the controller, which results in a faster response; this leads to the first problem of the proportional path : it runs continuously close to the set-point, but not quite there, in addition it only responds to the magnitude of the error not its rate

of change , which causes the plant to overshoot. The second problem is that it will always suffer from steady state error since it can't compensate for small errors. These problems are solved by the two remaining paths.

1.4.2 Integral path

The integral path generates an output signal proportional to the duration and magnitude to the error signal, the longer the error and the greater the amount, the larger the integral output :

$$I = K_I * \int_0^t e(t)dt \tag{1.21}$$

-K_I is the integral gain

As long as an error exists, integral action will continue. It needs to be noted that the integral path suffers from a phenomenon called "windup". Since the error will keep being integrated, the integral term might become very large and cause the actuator to saturate. The solution for windup is to include a way to limit the correction introduced by the controller, which will be discussed in Chapter 3.

1.4.3 Derivative paths

The derivative path creates an output signal proportional to the rate of change of the error signal, so the faster the error changes the larger the derivative output as shown in its definition :

$$D = K_D * \frac{de(t)}{dt} \tag{1.22}$$

-K_D is the derivative gain

Derivative control looks ahead to see what the error will be in the future, and contributes to the controller output accordingly. Its tradeoff is that it reduces overshoot but slows down the response. Also, overshoot and instability will arise from a K_D that is too large due to the amplification of the noise signal.

Now that the three paths are covered, they can be combined to form the PID controller defined as :

$$u = P + I + D = K_P * e(t) + K_I * \int_0^t e(t)dt + K_D * \frac{de(t)}{dt} \quad (1.23)$$

Where "u" is the command issued by the controller. Since the PID controller is implemented in a µC, Equation **??** needs to be discretized, it becomes:

$$u[t_K] = \underbrace{K_P * e[t_K]}_{\text{P term}} + \underbrace{I + K_I * e[t_K]}_{\text{I term}} + \underbrace{K_D * (e[t_K] - e[t_{K-1}])}_{\text{D term}} \quad (1.24)$$

A basic block diagram of a control system with a PID controller is shown in Figure **??** :

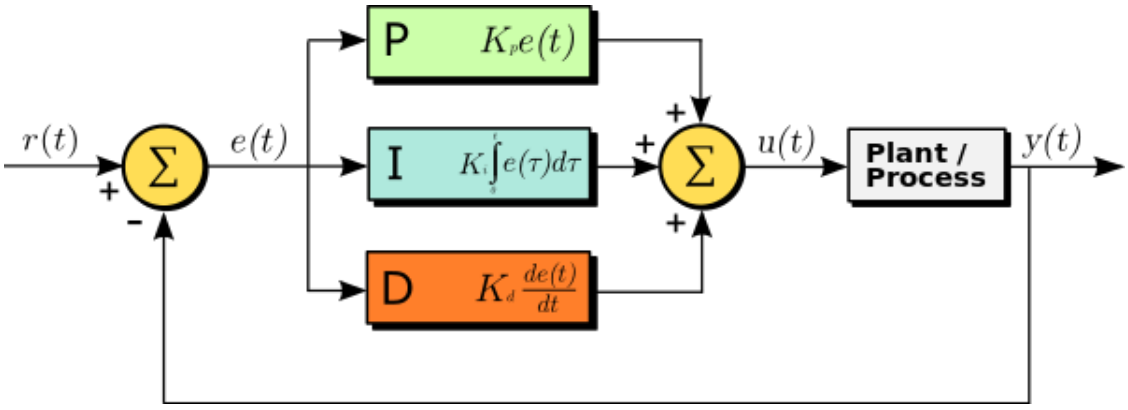


Figure 1.8: The PID controller

In this project, three variables need to be controlled in order to stabilize the quadcopter; they are the roll (ϕ), pitch (θ) and yaw (psi) angles. Thus, three controllers are needed to control each and keep them as close as possible to the desired set-point. The resulting quadcopter control system's block diagram is shown in Figure 1.9:

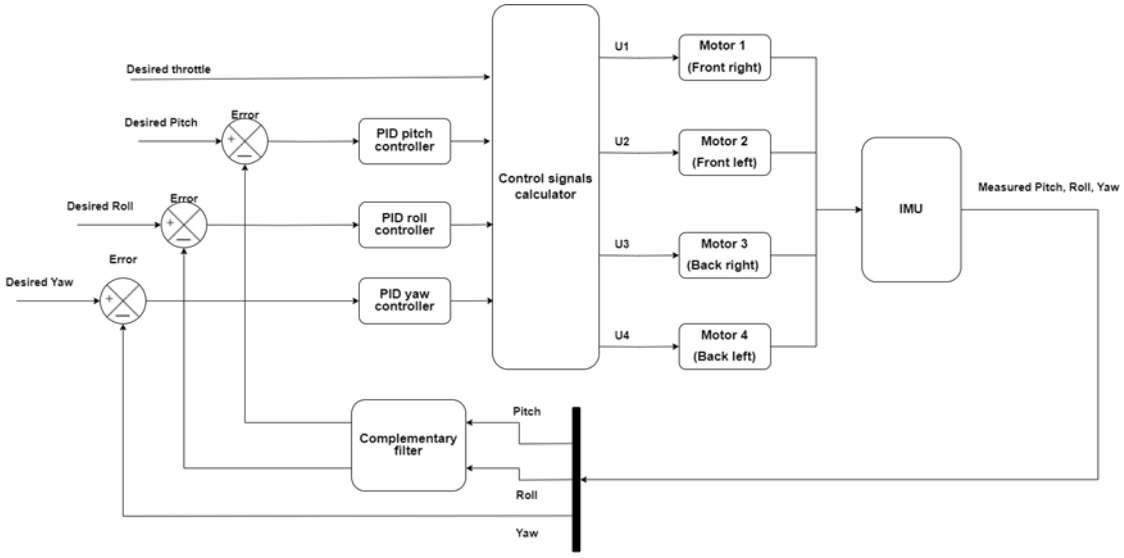


Figure 1.9: The Quadcopter control system block diagram

1.4.4 Tuning PID parameters

Every process responds differently, and the PID controller determines how much and how quickly correction is applied, by adjusting proportional, integral and derivative action. Controller tuning involves correctly setting the controller K_P, K_I, K_D values, for specific process requirements

The next step after setting up the PID controller is to tune its parameters. The aforementioned effects of changing each gain should be kept in mind during the tuning process. Multiple offline and online tuning methods have been developed over the decades, some of the most common are : Ziegler-Nichols method, Cohen-Coon, and software based methods such as genetic algorithm (GA) that might require knowledge about the mathematical model of the plant.

In this project, the tuning process consisted of trying multiple initial values of quadcopters of similar physical and electrical properties found in literature, then fine-tuning those values according to the behavior of the quadcopter until satisfactory performance was reached, so its the manual tuning based on the background theory accumulated.

1.5 Protocol of communicatios

In this section we will introduce the communication protocols used in this project brifely :

1.5.1 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification defines the protocol referred to as "HTTP/1.1" [4].

1.5.2 Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet switched computer communication networks, and in interconnected systems of such networks.

TCP, goes beyond just establishing a connection. It meticulously ensures data arrives correctly. Imagine a three-way handshake where both parties agree on the communication rules. Then, TCP assigns unique sequence numbers to every data packet, like a mailman numbering each letter. The receiver sends acknowledgments ("got it!") for each packet, allowing the sender to resend any missing or corrupted ones.

TCP uses a sliding window mechanism to control the flow of data, ensuring the receiver can handle the incoming packets without getting overwhelmed. This meticulous approach, with its sequence numbers, acknowledgments, and flow control, guarantees reliable data deliv-

ery, making TCP the perfect choice for applications like web browsing, email, and file transfers where accuracy is crucial[5].

Chapter 2

Hardware Description

In this chapter, the hardware components used in the implementation of the quadcopter will be presented in detail.

2.1 The Control Unit

The SOC µC used for this purpose is the ESP32 shown in Figure 2.1, developed by Espressif Systems as an upgraded version of the ESP8266; it is destined for embedded systems and IoT applications. It is wildly known for its low cost, low power consumption and many other features. It was mainly chosen for this project due to the abundance of GPIOs and its integrated Bluetooth and Wi-Fi modules.

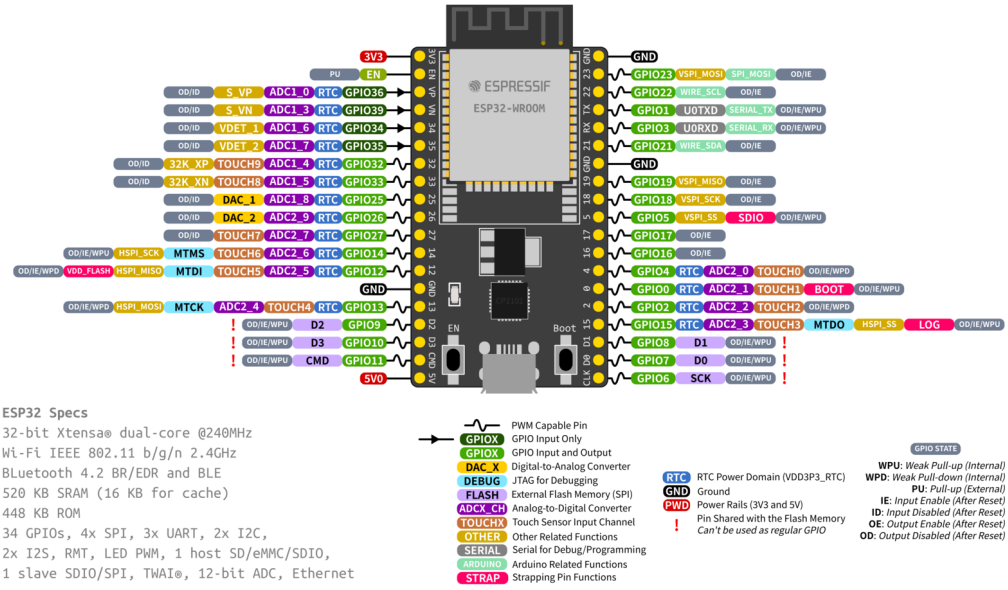


Figure 2.1: ESP32 Development Board By Espressif Systems

Features	ESP8266	ESP32
MCU	Xtensa Single-Core 32-bit L106	Xtensa Dual-Core 32-bit LX6 600 DMIPS
Frequency	80 MHz	80-240 MHz
WI-FI	802.11 b/g/n	Same
Bluetooth	NO	BL v4.2, BLE
SRAM	160 kB	512 kB
Flash	SPI Flash, up to 16 MB	Same
GPIO	17	36
HW/SW PWM	NO/8 channels	1/16 channels
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2
ADC	10 bit	12 bit
CAN	NO	1
Ethernet Mac interfdace	NO	1
Touch sensor	NO	YES
Temperature sensor	NO	YES

Table 2.1: comparison between EPS32 and ESP8266

2.2 Inertial Measurement Unit

An Inertial Measurement Unit or IMU is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body; it is generally composed of an accelerometer and a gyroscope. The 6-DOF MPU-6050 which made by **invensense tdk** is the ideal chip for the job;https://invensense.tdk.com.

The MPU-6050 devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die, together with an onboard DMP™, which processes complex 6-axis MotionFusion algorithms. The device can access external magnetometers or other sensors through an auxiliary master I^2C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor.

In this project; the goal is to apply the necessary manipulation on raw data to obtain the Roll, Pitch and Yaw angles of the quadcopter to sense its orientation and to make sure it stays horizontally leveled.

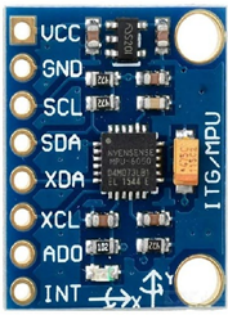


Figure 2.2: MPU6050

- The accelerometer operates on the principle of a mass on a spring, when an acceleration occurs, the string is either stretched or compressed. It is composed of a movable mass placed between stationary plates, when the former is subjected to an accelerating motion, it experiences a displacement and thus the capacitance of the stationary plates is varied. The value of the acceleration is related to the change in capacitance that results from it.

If the sensor is at rest with 0° tilt angle, the acceleration along both x and y axes are $a_x = a_y = 0\ m/s^2$, While the acceleration on the z-axis would be $a_z = 9.8\ m/s^2$.

- The gyroscope measures the angular velocity when a rotation occurs. At the occurrence of the former, the gyroscope measures the angular velocity. This is accomplished by utilizing the Coriolis effect, which states that if a body moves at a constant angular rate in a specific direction and a rotation is applied, a force perpendicular to the velocity and rotation axes is formed. When a gyroscope rotates, a Coriolis force is generated and measured using displacement and capacitance change; the angular rate is then determined as it is proportional to the Coriolis force.

- However, at rest and at 0° tilt angle, slightly unexpected results are found, which is due to offset errors. Offset errors can be fixed by calibrating the MPU-6050 chip. The calibration and data processing will be explored in detail in the next chapter.

2.3 Interfacing

The raw measurement data is transferred to the µC using the I^2C communication protocol, thus the sensor's SDA pin is connected to the µC's SDA pin : GPIO 21, and it is read using a built-in "wire.h" I^2C library. Serial Data (SDA) is the line used to exchange data between the master (the µC) and slave (MPU-6050) devices.

- The SCL pin of the MPU-6050 is connected to the SCL pin of the µC : GPIO 22. Signal Clock (SCL) is a signal that synchronizes the communication between the master and slave devices.

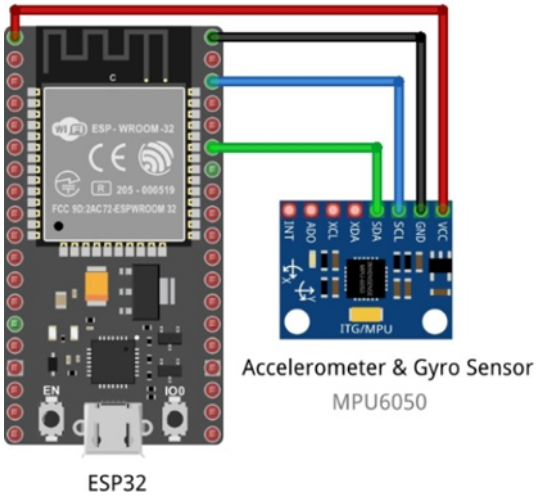


Figure 2.3: MPU-6050 connections

2.4 Actuators

2.4.1 Motors

DC motors are wildly used in robotics applications, there are two main types : Brushed DC motors (BDC) and Brushless DC motors (BLDC). Brushed motors spin the coil inside a case with fixed magnets mounted around the outside of the casing. Brushless motors do the opposite, the coils are fixed either to the outer casing (in-runner) or inside the casing (out-runner) while the magnets are spun.

Brushless motors are preferred because they spin much faster and are easier to maintain. In addition, out-runner motors are used instead

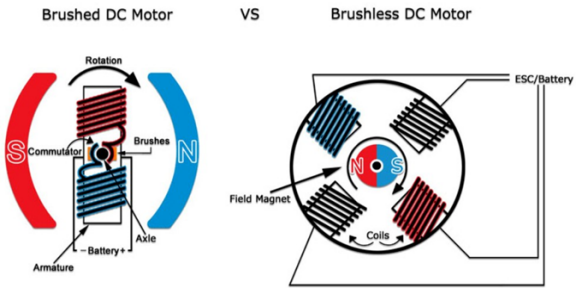


Figure 2.4: Brushed and Brushless DC motors

of in-runners because even-though they spin relatively slower, they can generate more torque which allows them to drive larger propellers. Kv is the number of RPM that the motor will spin at when supplied with 1V without a load attached to the motor. It also allows an approximation on the torque expected from the motor. Torque is influenced by the strength of the magnets and the number of windings on the armature. A low Kv motor has more winds of thinner wire, it will carry more volts at fewer amps, produce higher torque, and swing a bigger propeller, while a high Kv motor has fewer winds of thicker wire that carry more amps at fewer volts and spin a smaller propeller at high revolutions.

In our project, we chose the A2212 1400KV Brushless Motor seen in Figure 2.5, it weighs around 60g and can generate up to 1Kg lift force (depending on the properties of the used propellers), which satisfies the rule of thumb that the motors should generate a lift force that is twice the weight of the quadcopter. Thus, in the case of a quadcopter that weighs between 1Kg and 2Kg, the four motors can generate up to 4Kg lift force which guarantees enough power and a smooth motion.

It would be likely to mention that the motors should be calibrated before using them, at least once, and that was performed of course.

Figure 2.5: A2212 1400KV Brushless Motor

2.4.2 Electronic speed controller

An electronic speed control (ESC) is an electronic circuit that controls and regulates the speed of an electric motor. It may also provide reversing of the motor and dynamic braking. Miniature electronic speed controls are used in electrically powered radio controlled models. Full-size electric vehicles also have systems to control the speed of their drive motors.

the 30A rating model was the one chosen. It means this ESC in particular handles continuous 30A, and it can also handle bursts to up to 40A for a small period of time.

Figure 2.6: Readytosky 30A ESC

The ESCs are comprised of 3 sets of wires, 3 bullet connector wires (Blue) that connect to the motor, a pair of thick wires (red and black) that go back to the battery, and a 3-pin servo connector that goes into the flight controller (white : signal, red : 5V, black : ground). ESCs contain a number of components, the most notable are the microcontroller and MOSFETs. The microcontroller houses the firmware that interprets the PWM (PulseWidth-Modulation) signals sent by the flight controller, manipulates it while taking into consideration the current position of the motor, then sends the appropriate power to the motor by managing the operation of the FETs that are arranged in half bridge configuration to control the commutation sequence; a wider PWM signal implies a faster commutation sequence which dictates faster switching of the FETs, thus resulting in a faster motor speed. The position and speed of the motor can be sensed either using sensors such as “Hall

Effect" sensors inside the motor or by relying on the "Back Electromotive Force" (Back EMF) generated by the poles, the latter is used in quadcopters because the Back EMF is reliable at high speeds and is relatively cheaper. This information is crucial in determining the appropriate switching of the FETs.

2.5 Power

2.5.1 Battery

Lithium Polymer batteries, also known as LiPo batteries, have high energy density, high discharge rate and light weight which make them a great candidate in Radio-Controlled applications.

They consist of cells that are connected in series, each cell can supply a nominal voltage of 3.7V. The "S" number defines how many cells a battery possesses. For example, a 3S battery has 3 cells connected in series which provides a nominal voltage of 11.1V.

The capacity of a LiPo battery is measured in "mAh", which refers to how many continuous amps can be drawn from it for an hour until it's empty.

The "C" rating (discharge rate) refers to the maximum continuous current that can be safely drawn.

$$MaxCurrent = C_{rating} * Capacity \tag{2.1}$$

In this project, the moseworth™ 3s 2200 mah XT60 battery is chosen due to its relatively light weight and reliability.

Figure 2.7: Caption

2.5.2 Power distribution board

A power distribution board (PDB) is a component that distributes the power from the battery to the ESCs. The chosen XT60 PDB is a high performance $36 * 50$mm PCB that can also provide a synchronized and regulated 5V DC that is used to power the flight controller and other on board electronics.

Figure 2.8: XT60 PDB

2.6 Mechanical parts

2.6.1 Frame

In this project, the "X" configuration $F450$ frame is used. The mainframe is constructed using glass fiber and the arms using poly amide nylon, which makes it a very sturdy frame while keeping it relatively light. It also features integrated PCB connections for direct soldering of the ESCs. This eliminates the need for messy multi connectors, which helps keep the layout very tidy.

Figure 2.9: F450 Frame

2.6.2 Propellers

Propellers are the workhorses that convert the motor's power into thrust, propelling the quadcopter through the air. They come in various sizes and materials, each with its own advantages and trade-offs. Larger propellers, often used with lower Kv motors, which can generate more thrust but spin slower, leading to higher efficiency for longer flight times. Conversely, smaller propellers paired with high Kv motors spin much faster, creating a more agile and maneuverable quadcopter but sacrificing some flight time. The chosen propeller material also plays a role. Common materials include plastic, wood, and carbon fiber.

Its specifications are defined in terms of diameter and pitch (DxP). The diameter is the end to end length (in inches) of the propeller. The pitch can be defined as the distance (in inches) travelled per revolution. Larger propellers with lower pitch ratings are more efficient than smaller propellers with higher pitch ratings.

In this project, the 10-inch diameter 45-inch pitch (1045) propellers were chosen mainly due to their compatibility with the frame size and their availability.

Figure 2.10: 1045 Propellers

2.7 Hardware implementation

In this section, the necessary connections between all the sensors, actuators and µC are made as shown in Figure 2.11

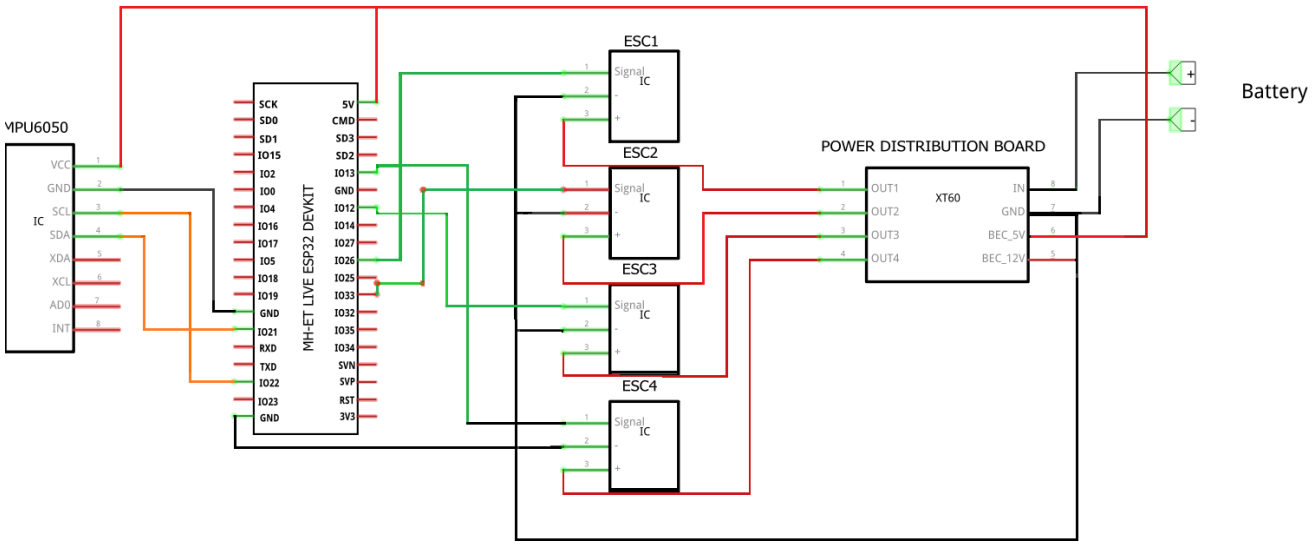


Figure 2.11: System schematic

Chapter 3

Software Implementation

In this chapter, the software implementation of the quadcopter will be discussed. Firstly, reading and manipulating the angular data from the IMU will be collected, then the communication with the control unit using the application and the rest of the components are covered leaving the PID controller for last. The purpose of this project is to ensure that the quadcopter stays leveled, which translates to minimal error between the desired pitch and roll angles (0°) and the actual angles read by the IMU. Minimizing these errors will be ensured by the PID controllers that will manipulate the motors to reach the desired states.

3.1 IMU Manipulation

For that i used two methods, and both performed well.

The first one using the MPU6050_light library, created by **rfetick**, which does all the manipulation of this sensor's data, you can find it on GitHub https://github.com/rfetick/MPU6050_light.

The second method is to manually manipulate the registers and its behavior and extract the raw data based on the data-sheet from the manufacturer TDK https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf.

Both methods are available in my GitHub [6], but we will discuss about the second one, since this approach provide a deep understanding to the environment.

3.1.1 I2C serial communication protocol

The MPU-6050 chip uses the I^2C protocol to communicate with the µC.

This protocol has two lines : the SDA line used to send and receive data bit by bit, and the SCL line used for the clock signal. Data is transferred in messages, Each message has an address frame that contains the binary address of the slave, which have a 7 or 10-bit slave address frame that contain the data being transmitted, one or two 8-bit data frames, a start bit, a stop bit, a read/write bit and an acknowledgment bit.

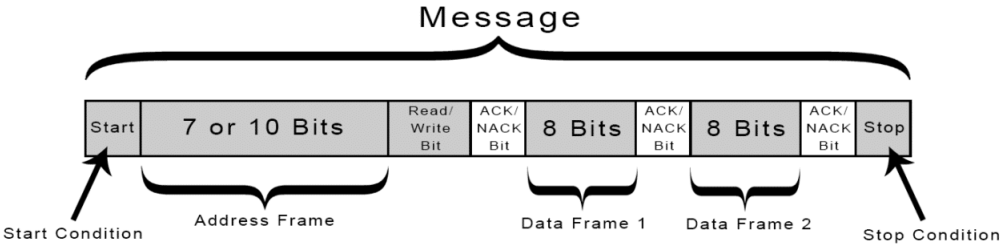


Figure 3.1: I^2C message format

In esp32 the I^2C pins are GPIO21 and GPIO22 for SDA and SCL lines respectively. Multiple master devices can have multiple slave devices. In this case, the master is the µC while the MPU-6050 is the slave. As long as the addresses of multiple slaves are distinct, no risk of conflict will be present when communicating with them.

The communication is handled by the " Wire.h" library, but further manipulation is needed.

3.1.2 Calibration

At the rest state, the IMU reading are not 0° tilt, due to offset reading error, so this formula is used :

$$val_{offset} = \frac{\sum_{i=1}^{N} Val_{measured}}{N} \tag{3.1}$$

then the true value will be:

$$Val_{true} = Val_{measured} - Val_{offset} \tag{3.2}$$

This operation is used in every reading iteration to make sure the values are corerct.

3.1.3 Extracting the Roll, Pitch and Yaw angles

From Accelerometer :

Since gravitational acceleration is always acting on and being measured by the accelerometer, this vector can be decomposed into its components in the X, Y and Z axes.

The pitch θ and roll ϕ angles, and those values are calculated by Acc_data function, then using inverse trigonometric functions, which known as the Euler angle formulas, the final values are found. The derived formulas are as follows :

$$\theta = \arctan \frac{-a_x}{\sqrt{a_y^2 + a_z^2}} * \frac{180}{\pi}, \phi = \arctan \frac{a_y}{\sqrt{a_x^2 + a_z^2}} * \frac{180}{\pi} \tag{3.3}$$

Since the formula rely on the acceleration values measured by the accelerometer, any sudden acceleration in any direction will disconcert the estimation of the angles. So even though the accelerometer is reliable in the long term since gravity is always present and constant, it leaves a lot to be desired when dealing with short term quick measurements.

From Gyroscope :

The gyroscope measures the rotational motion of the Quadcopter. Specifically, it measures the angular rate "ω" along three axes on the X, Y and Z axes, and this is done by the Gyro_data function. The angles traveled in a single time step is simply found by integrating the angular rate measured by the gyroscope.

Complementary filter

The complementary filter is one of many sensor fusion algorithms. They combine the data of multiple sensors in order to reach a better

performance than a single one . As is the case in this project, one of the sensor's strengths can compensate for the weaknesses of the others. Among many algorithms, the complementary filter was chosen due to its simplicity and good performance. It combines the roll and pitch angles extracted from both the accelerometer and gyroscope to approximate an estimation of the angles as follows :

$$\begin{bmatrix}\theta\\ \phi\end{bmatrix} = 0.9996 * \begin{bmatrix}\theta_{gyro}\\ \phi_{gyro}\end{bmatrix} + 0.0004 * \begin{bmatrix}\theta_{acc}\\ \phi_{acc}\end{bmatrix} \tag{3.4}$$

The complementary filter takes the best of the two sensors : the low frequency components of the accelerometer, and the high frequency components of the gyroscope.

All of the above equations are implemented in a header file with the name "IMU.h" with the functions :

-"mpu_init()" that initiates the Wire communication with the MPU-6050 by sending the slave address which is 0x68, then reset the 6B register.

- "IMU_calibration()" that reads the measurements of both the gyroscope and accelerometer and applies the calibration formula 3.1, to discard the offset errors.

- "get_angle()" it begins the Wire communication then it reads the measurements from the I^2C bus and applies the formula 3.3 to obtain the correct values. It uses the latter to calculate the angles, then applies the complementary filter equation 3.4 to acquire the final pitch, roll and yaw angles.

The entirety of the header file can be accessed in the project's GitHub repository[6].

3.2 Communication

Communication between the µC and an interface of some kind is necessary in the tuning process and in controlling the setpoint. In this project, thanks to the ESP32's integrated Wi-Fi module, it is connected to a custom graphical user interface (GUI) application.

The ESP32 can either connect to an existing Wi-Fi access point (Station mode) or act as an access point itself so other devices can connect to it (Access Point mode). All of the above is easily provided by the built-in library “wifi.h”.

In this project, the µC is set to access point mode and a mobile is connected to it, once the connection is established, the IP address assigned to the µC (the server) is retrieved. The IP address will be used by the client to connect and send data to the ESP32.

3.2.1 Application / Client side

The application is used to send throttle values and the roll and pitch values desired, but PID coefficients were changed manually in the code of the µC because with this method more stability were gained.

- The throttle is controlled using the joystick inside the application to listen to the events. The ESCs are controlled using a 50Hz PWM signal, a pulse width of 1000µs is equivalent to zero RPM, while one of 2000µs is equivalent to maximum RPM.

-The pitch and roll values were controlled with the second joystick, by limiting the range of the desired angle, but that comes after the testing phase, as a final step which take place after assuring that the throttle input works just fine.

-A security flag named ”armed/desarmed”, stopping any movement or calculation when its equivalent to zero, using the RST() function, that reset everything, in the other hand, if the flag is armed, the calculation are well performed.

-The files that provide the information about the GUI are WifiServer.h and Controller.h.

3.2.2 Server side

Once the mobile is successfully connected to the ESP32 hotspot, it listens for any client requests. The three-way handshake is performed and each time the client sends data, the “server-¿handleClient()” function receives the packets, unpacks them, converts them and assigns each value to a variable.

3.3 Controller

The controller is responsible for calculating errors and issuing commands in order to compensate for the calculated errors.

3.3.1 Calculate_errors function

In this function, the set-points are read and compared to the measured angles, which allows the calculation of the errors. However, this function is best visualized using the flowchart in Figure below. In order to prevent windup of the integral path, the sum of the errors is limited. The function used for this purpose is called "minmax()" or just a couple of if statements because they are identical.

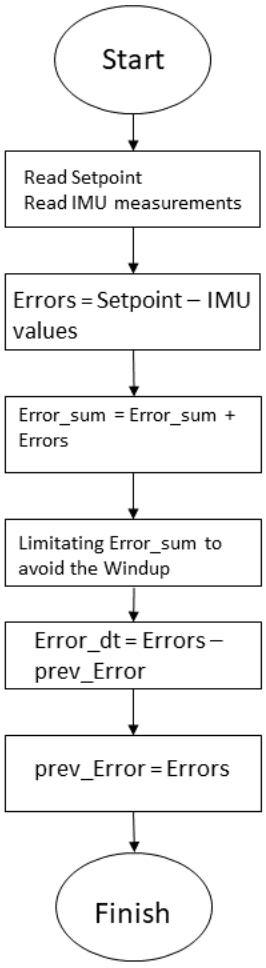


Figure 3.2: calculate_error function flowchart

3.3.2 PID function

This function takes the values of the three controllers, and calculate the overall result using the equation 1.24, then adjust the speed of each motor by the following formula :

$$\begin{aligned} EscFrontRight_{Speed} &= throttle_{command} - roll_{PID} + pitch_{PID} + yaw_{PID} \\ EscBackRight_{Speed} &= throttle_{command} + roll_{PID} + pitch_{PID} - yaw_{PID} \\ EscFrontLeftSpeed &= throttle_{command} - roll_{PID} - pitch_{PID} - yaw_{PID} \\ EscBackLeft_{Speed} &= throttle_{command} - roll_{PID} + pitch_{PID} - yaw_{PID} \end{aligned} \tag{3.5}$$

3.3.3 brushless_speed function

In order to control the ESCs, the library called "ESP32Servo.h" is used, which is made by Kevin Harrington, John K. Bennett , with its documentation https://madhephaestus.github.io/ESP32Servo/annotated.html. It was developed based of the Arduino "Servo.h" library. Like it was stated in the client side section, this library enables the user to select the duty cycle of the PWM signal. At first, the "motor_init" function is used to initialize the ESCs by attaching them to their respective PWM pins.

Secondly, the ESCs are set to the maximum value (2000µs) for 2 seconds then they're set back to the minimum value (1000µs); this operation is used for calibrating the ESCs, the calibration is handled using the "esc_calibration()".

Note that this operation is only performed after changes in the configuration. After the needed command is calculated and passed through the "minmax()" function to ensure no saturation occurred, it is sent to the ESCs through the function "brushless_speed()" which send the Final values of the PID() finction as speeds of each motor respectively as seen before the eqautions 3.5, but after limitating it.

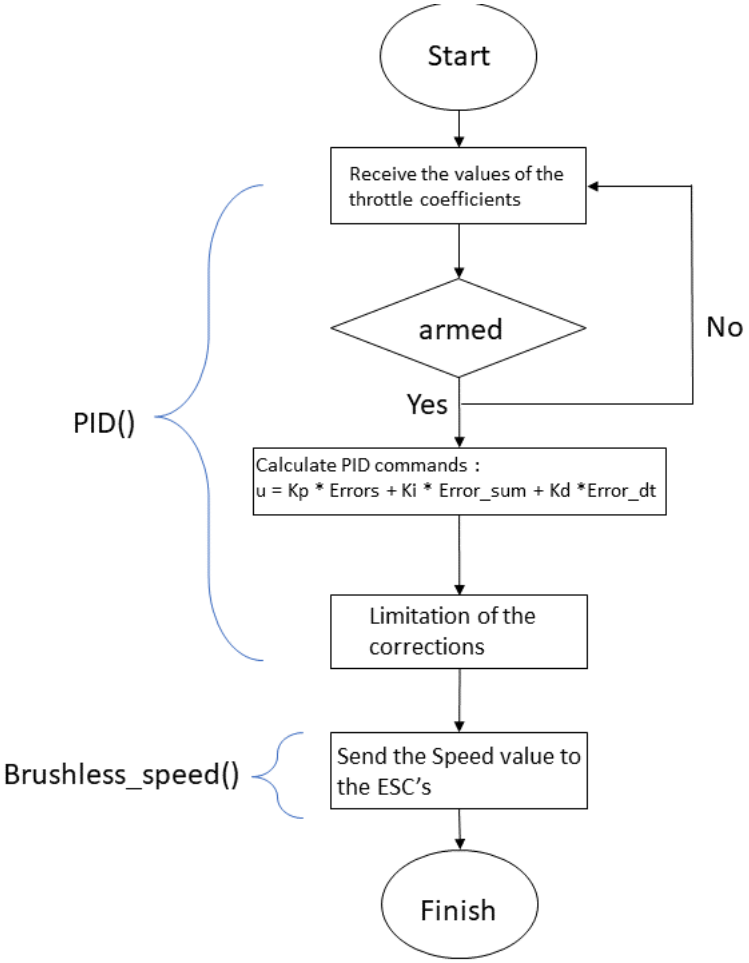


Figure 3.3: Compensation Flowchart

And Finally after all the main calculation, this is the overall flowchart :

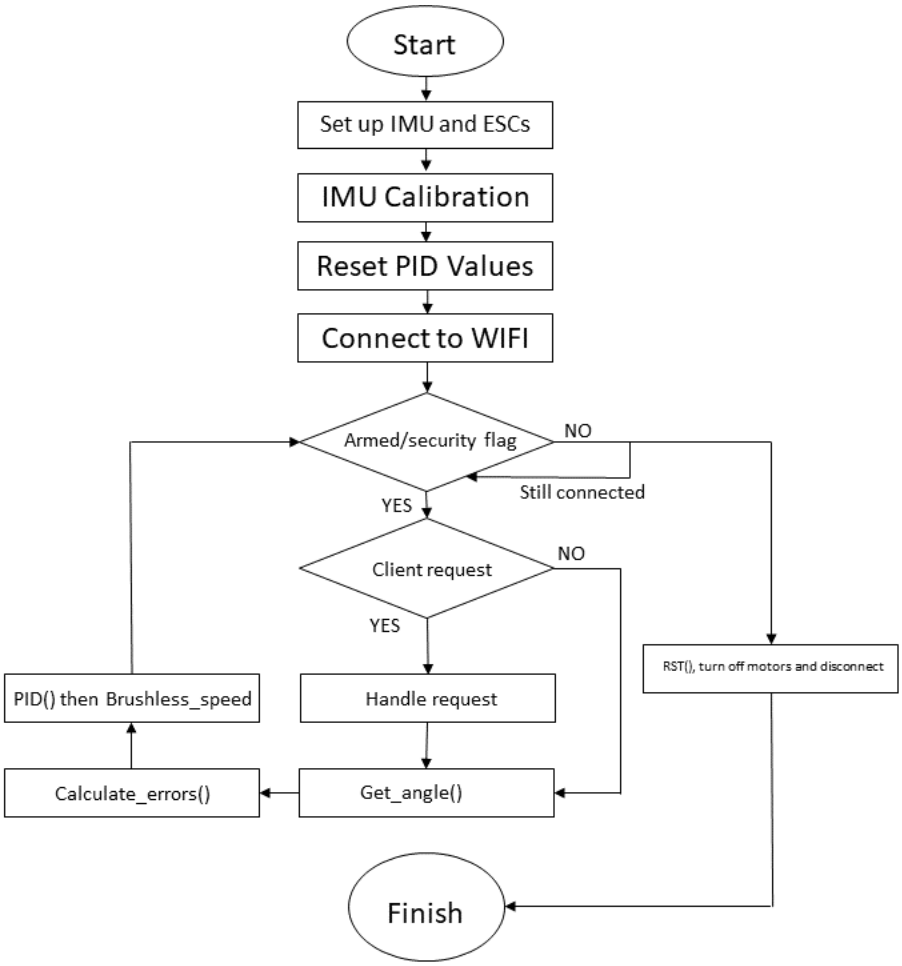


Figure 3.4: Quadcopter stabilization flowchart

Chapter 4

Result and Discussion

In this chapter, a general discussion concerning the hardware/software implementation and the obtained results is provided, in addition, It presents practical constraints encountered during the different stages of the quadcopter system implementation and testing. Also, light will be shed on suggestions to improve this work in the future.

During the implementation of this project, multiple problems were faced. Most notably, vibrations caused by the revolutions of the motors that affects the IMU and affected its values in a minor manner, adding a piece of foam under the IMU proved to be a simple yet effective solution.

Concerning vibrations, unbalanced propellers may result in a very unsteady performance. They cause oscillations that will prevent reaching a satisfactory performance. Thus, balancing propellers is a very important process to ensure a stable flight. All the aforementioned problems were faced during the tuning process. However, all were solved and satisfactory results were reached after many redesigns, small tweaks and tuning sessions.

The tuning process consisted of using new values based on the theoretical background, and then notice there effect on the quadcopter which determine the next change. The flight tests were done outdoors in a spacious area with minimal presence of obstacles to avoid damage to both property and the quadcopter itself.

Also, windy days were avoided since they may cause fatal crashes to the vehicle. The flight tests consisted of powering the quadcopter, increasing the throttle slowly until it lifts off, keeping it at hover and

fine-tuning the PID coefficients through the ide while observing the stability until a satisfactory flight was reached.

The obtained final PID coefficients for pitch and roll axes are: $k_p = 1.93$, $k_i = 0.003$, $k_d = 65$ For the yaw axis : $k_p = 1.4$, $k_pi = 0$, $k_d = 0$, but there is room for further improvement, and this is the next step.

Concerning the limitations and problems encountered during the realization of this project :

- Power limitations: the quadcopter manages to reach small time of flight time on a fully charged battery. Naturally, the flight time can be extended by using a larger battery. However, since that is the most expensive component in this project, it would raise the cost by up to 50%, and will cause a change in the coefficients because of the additional weight.

- Communication range : the communication range is a very important parameter because it allows controlling the quadcopter from far distances. In our project, Wi-Fi communication was used. However, it allows communication in a range of 100 m only, thus the quadcopter can be operated only within that range.

- Manual altitude control : due to the absence of a barometer, altitude control couldn't be implemented. This leaves the altitude control as a manual task to be performed by the user that has to command the throttle values.

The vision behind undertaking this project is to take a step towards completing a fully autonomous quadcopter for the purpose of implementing projects in swarm technology. For this purpose, we propose the following ideas and improvements :

- Increasing the range of communication is highly recommended; it can be done using NRF24 modules that use radio frequency which are known to reach a range of up to 800m and provide powerful communication.

- Implementing altitude and position control using a barometer and a global position system module (GPS). However, it is important to address the inherent large error associated with the GPS module. To enhance the accuracy of the estimated position, it is highly recommended to incorporate the Kalman filter that performs data fusion by

combining the measurements of both the GPS module and the IMU. This does indeed mitigate the limitation of the GPS module and provides a very precise altitude and position control.

- Trajectory planning is also a critical function of a fully autonomous quadcopter that relies on the latter recommendation, it might be done using multiple approaches.

- Equipping the quadcopter with a camera is an intriguing aspect to explore, as it enables the execution of a wide variety of advanced operations. Since Artificial Intelligence is at an all-time high, quadcopters can utilize AI applications such as depth estimation, object detection and much more. Adding a camera to a UAV gives it the ability to record visual data from its surrounding enabling it to analyze its environment and make intelligent decisions such as obstacle avoidance. The integration of a camera and AI technology presents interesting opportunities to enhance the capabilities and autonomy of quadcopters, opening new doors for applications in areas like surveillance, inspection, mapping and beyond.

Finally, i would like to thank you for reading my report, i hope the best for you in your next step.

Conclusion

In this work, a quadcopter stabilization system during vertical flight was successfully implemented using a PID controller. Through careful assembly and multiple sessions of meticulous tuning of the controller gains, a satisfying performance was achieved.

The ESP32 µ**C** proved to be a solid choice for implementing this project; it provided seamless integration with the sensors and actuators and handled the communication well.

The constraints faced in this project were hard to overcome specially manual tuning, but through trial and error satisfactory, results were reached and the quadcopter performed a stable vertical flight. Thus, it can be said that the purpose of this project was achieved. Through this project, a deep knowledge of the fundamental technologies related to quadcopter systems was acquired, mainly quadcopter modelling, microcontroller programming, how to establish Wifi communication, basic operation of IMU, programming in c/c++, interact with multiple environments and APIs, and finally PID controller practical tuning. It is worthwhile to mention that huge efforts were made for both hardware and software implementation of the project, in order to guarantee the success of this work.

This project was a real practical challenge that put one's skills to the test and further polished them and made acquiring new ones possible. Future work consists of moving forward in order to realize the abovementioned vision and contribute to research in the field.

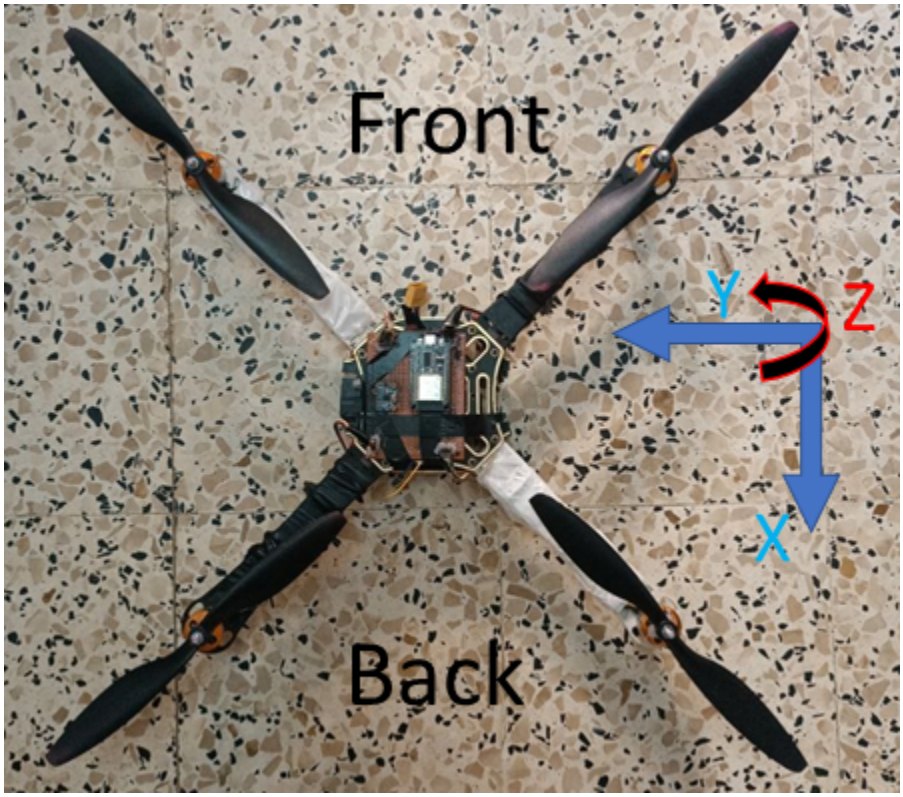


Figure 1: Body of the Quadcopter

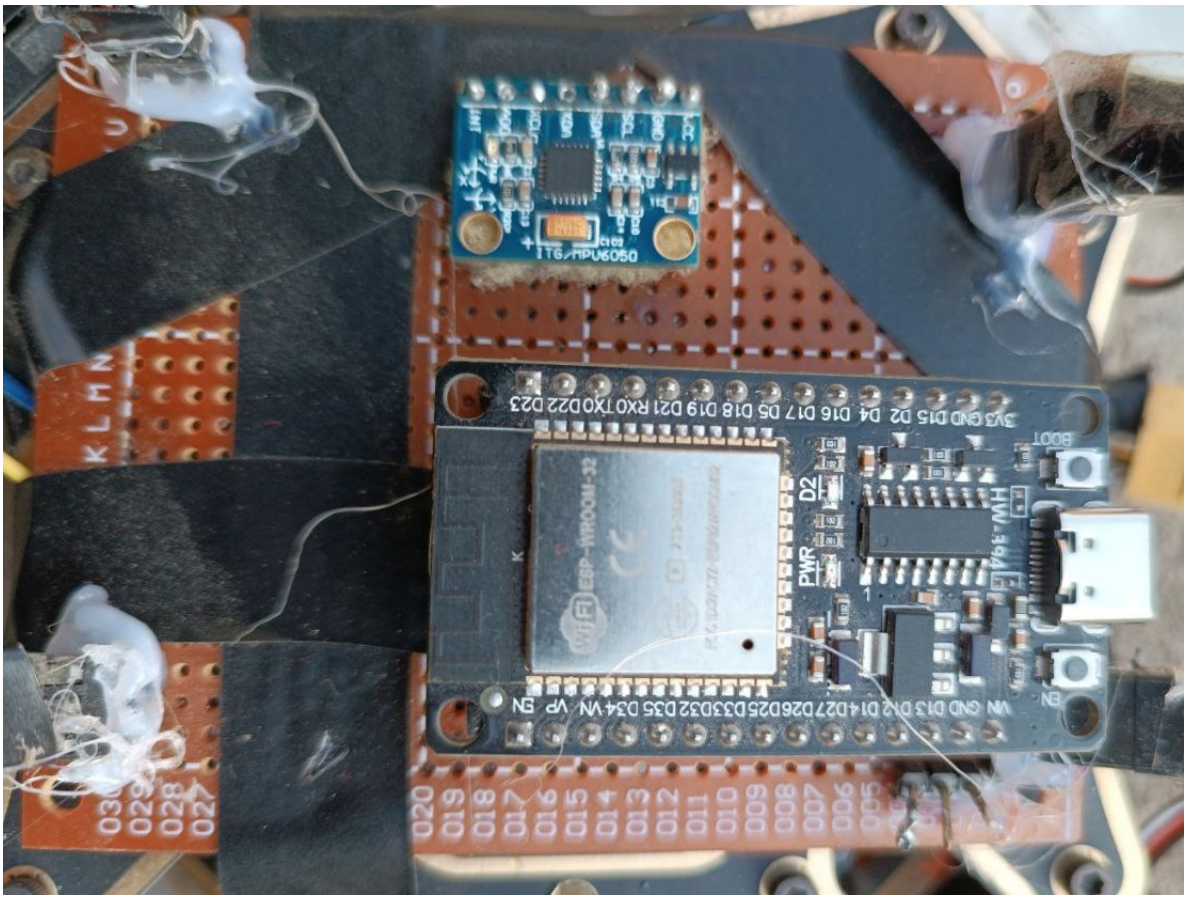


Figure 2: Flight controller

Bibliography

[1] R. Niemiec and F. Gandhi (2016), URL https://api.semanticscholar.org/CorpusID:53062402.

[2] K. M. Thu and A. Gavrilov, Procedia Computer Science **103**, 528 (2017), ISSN 1877-0509, xII International Symposium Intelligent Systems 2016, INTELS 2016, 5-7 October 2016, Moscow, Russia, URL https://www.sciencedirect.com/science/article/pii/S1877050917300479.

[3] K. B. S. G. b. Apekshit Goel, Vibhore Bindra (2016), URL http://ir.juit.ac.in:8080/jspui/jspui/handle/123456789/7542.

[4] H. Nielsen, J. Mogul, L. M. Masinter, R. T. Fielding, J. Gettys, P. J. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (1999), URL https://www.rfc-editor.org/info/rfc2616.

[5] W. Eddy, *Transmission Control Protocol (TCP)*, RFC 9293 (2022), URL https://www.rfc-editor.org/info/rfc9293.

[6] MounirDahmane, *MounirDahmane/Quadcopter-design-using-PID- Controller: Init* (2024), URL https://github.com/MounirDahmane/Quad-copter-design-using-PID-Controller.