الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and scientific research
المدرسة الوطنية العليا للتكنولوجيات المتقدمة
National Higher School of Advanced Technologies
Department of Electrical Engineering and Industrial Computing

Final Year Project to Obtain the Diploma of

Engineering

Specialty:

Embedded Systems

-Subject-

Design of a Multi-rotor Drone Equipped with a Vision-based Automatic Piloting Mode

Realized by

Mohamed Ben Chikh El Hocine

Members of The Jury:

Chair	Tarek Cherifi	MCA	ENSTA
Supervisor	Bouchachi Islem	MCB	ENSTA
Co-Supervisor	Abbadi Belkacem	MCB	ENSTA
Examiner	Nouredine Cherabit	MCB	ENSTA
Examiner	Seloua Chouaf	MAA	ENSTA

Algiers, The 25/06/2024

Academic year 2023 - 2024

Table of Contents

List of Figures

List of Tables

Dedication

This thesis is dedicated to several special people in my life whose support, and encouragement have made this journey possible.

I dedicate my graduation to the one who was the light of my path, and the one who taught me without waiting for my dear father, and to my hope in life, to the one whose prayers were the secret of my success, my beloved mother.

To my siblings, ou have been my support during my journey, and for that, I am eternally grateful. May Allah bless you.

To my friends, ***Khalil, Toufik, Anis, and Tarek*** *Thank you for the laughter, fun, and memories we share. Your friendship brings joy to my life, and I am grateful for each of you.*

The last day of my academic life, and I thank God for completing 17 a year of study.

Lastly, to everyone who believed in me and contributed to my success, I am grateful. This thesis would not have been possible without each of you.Thank you all for being part of this journey with me.

Acknowledgments

Above all, we are thankful to Allah for enabling us with the strength and courage to accomplish this work. Alhamdulillah for His guidance and blessings.

I would like to express my gratitude to my supervisors, **Mr. Bouchachi** and **Mr. Abbadi**, *for their continuous support, patience, and motivation. Their guidance was invaluable throughout the research and writing of this thesis. I am grateful to have* **Mr. Bouchachi** *as an excellent advisor in the field of UAVs, his extensive experience and generosity with his knowledge have been invaluable. Additionally, i cannot forget* **Mr. Abbadi**, *who taught me embedded system architecture and image processing, which greatly helped me with my project.*

I would also like to thank the members of my thesis committee: **Mr. Cherifi**, **Mr. Cherabit**, and **Ms. Chouaf**, *for their time and consideration in discussing and evaluating my work.*

I am also grateful to the online communities and forums dedicated to UAV technologies for providing resources and sharing their knowledge and experiences, which were especially helpful for troubleshooting.

To my friends, thank you for your support and understanding during my project. Your companionship made this journey more enjoyable and less stressful.
This accomplishment would not have been possible without Allah's blessings. Alhamdulillah for everything.

Symbols and Abbreviations

Symbols

x, y, z: Longitudinal, lateral, and vertical motions [m].
$\dot{x}, \dot{y}, \dot{z}$: Velocity components along x, y, and z axes [m/s].
ϕ, θ, ψ: Euler angles representing roll, pitch, and yaw [rad].
$\dot{\phi}, \dot{\theta}, \dot{\psi}$: Angular velocities corresponding to roll, pitch, and yaw[rad/s].
u, v, w: Velocity components of the quadcopter in the body frame [m/s].
p, q, r: Angular velocities of the quadcopter in the body frame [rad/s].
m: Mass [kg].
g: gravity acceleration, 9.81 [m/s^2].
K: Thrust coefficient.
ω: Angular velocity of the rotor[rad/s].
l: distance [m]
τ_x, τ_y, τ_z: Roll, pitch, and yaw moments [N·m].
I_{xx}, I_{yy}, I_{zz}: Roll, pitch, and yaw moments of inertia [kg·m²].

Abbreviations

CCW	Counter-Clockwise
CIA	Central Intelligence Agency
CW	Clockwise
DRAM	Dynamic Random-Access Memory
GPS	Global Positioning System
HTTP	HyperText Transfer Protocol
Li-Po	Lithium Polymer
OS	Operating System
OSI	Open Systems Interconnection
PDB	Power Distribution Board
PID	Proportional, Integral, Derivative
PC	Personal Computer
PSRAM	Pseudo Static Random-Access Memory
PWM	Pulse Width Modulation
S_BUS	Serial Bus
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
WiFi	Wireless Fidelity

Abstract

This work focuses on the development of a multirotor drone with a vision-based automatic piloting mode. The project integrates various components, including the Pixhawk 2.4.8 flight controller and the ESP32-CAM module for real-time visual data transmission. By utilizing vision-based navigation, the drone's security is improved, instead of GPS systems that communicate with satelite by electromagentic waves that can be easily hacked.

This project aims to demonstrate the efficacy of vision-based systems in ensuring reliable and secure drone operations. Camera can replace traditional GPS systems and the drone relies on its estimatation location.

KeyWords:

Multirotor Drone, Vision-Based Automatic mode, Pixhawk 2.4.8, ESP32-CAM Module, Drone Security, GPS Systems.

Résumé

Ce travail porte sur le développement d'un drone multirotor doté d'un mode de pilotage automatique basé sur la vision. Le projet intègre divers composants, notamment le contrôleur de vol Pixhawk 2.4.8 et le module ESP32-CAM pour la transmission de données visuelles en temps réel. En utilisant la navigation basée sur la vision, la sécurité du drone est améliorée, au lieu des systèmes GPS qui communiquent avec le satellite par des ondes électromagnétiques qui peuvent être facilement piratées.

Ce projet vise à démontrer l'efficacité des systèmes basés sur la vision pour garantir la fiabilité et la sécurité des opérations des drones. La caméra peut remplacer les systèmes GPS traditionnels et le drone se fie à l'estimation de sa position.

Mots clés:

Multirotor Drone, Mode automatique basé sur la vision, Pixhawk 2.4.8, ESP32-CAM Module, Sécurité des drones, GPS.

ملخص

يركّز هذا العمل على تطوير طائرة بدون طيار مزودة بوضع التوجيه التلقائي القائم على الرؤية. يدمج المشروع مكونات مختلفة، بما في ذلك وحدة التحكم في الطيران Pixhawk 8.4.2 ووحدة ESP32-CAM لنقل البيانات المرئية في الوقت الحقيقي. ومن خلال استخدام الملاحة القائمة على الرؤية، يتم تحسين أمان الطائرة بدون طيار، بدلاً من أنظمة تحديد المواقع التي تتواصل مع الأقمار الصناعية عن طريق الموجات الكهرومغناطيسية التي يمكن اختراقها بسهولة.

يهدف هذا المشروع إلى إثبات فعالية الأنظمة القائمة على الرؤية في ضمان عمليات موثوقة وآمنة للطائرات بدون طيار. يمكن للكاميرا أن تحل محل أنظمة GPS التقليدية وتقوم الطائرة بدون طيار على تحديد موقعها.

General Introduction

Drones, also known as unmanned aerial vehicles (UAVs), have become very popular in the last decade as a result of their effectiveness in many various applications. At the beginning, they were mainly used for military purposes, but now they are used in different industries such as agriculture, and cinematography. Drones are very useful because they can go to places that are hard for people to reach out and do things by themselves, which is extremely helpful in today's world.

Nowadays, researchers are increasingly interested in the security of drones, especially in military applications. This interest expanded after an incident where Iran guided a lost CIA (Central Intelligence Agency) stealth drone to land in their territory. They did this by exploiting a known weakness in the drone's navigation system. Iranian electronic warfare experts cut off the drone's communication links and reconfigured its GPS, making it believe it was landing at its home base in Afghanistan when it was actually in Iran. This event shows the importance of improving drone security [29].

The primary objective of this project is to develop a quadcopter drone equipped with a vision-based automatic piloting mode. Unlike other approaches such GPS, which can be easily hacked, the use of vision based approach allows to enhance safety.

This report is organized as follows:

The first chapter covers the fundamentals of quadcopter drones, including kinematics, dynamics, communication protocols, and PID controllers.

The second chapter focuses on brushless DC motors and their crucial role in enabling drones to fly, while also exploring drone components and operational principles. This chapter is essential for successfully building and flying a drone.

In the third chapter explains each step of the development process in straightforward terms, ensuring comprehensive coverage from start to finish, including results and obstacles encountered during this project.

Chapter 1

Theoretical Background

1.1 Introduction

This chapter covers the underlying principles of the project. We describe quadcopter modeling, the implemented communication protocols, then we explore the PID controllers.

A quadcopter has four motors, and each one has its own propeller. The important thing about these motors is their configuration and rotation, they come in two configurations which are the **X configuration** and **plus (+) configuration** as shown in **Figure 1.1** below.

The only difference between the two is which motors we send commands to change the direction of the quadcopter. Also, the X configuration provides better stability and maneuverability, and it will be used throughout this project [1].

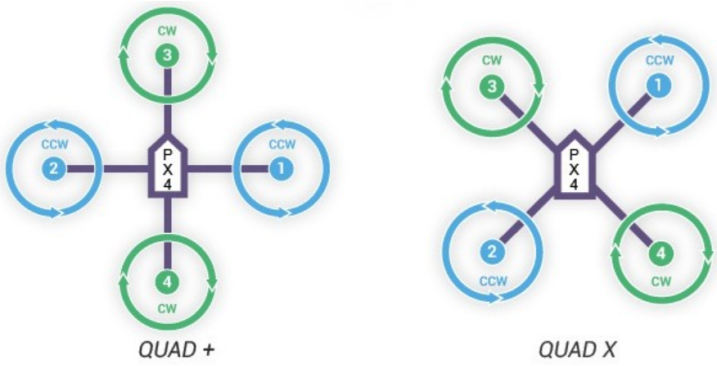


Figure 1.1: The two configurations of quadcopter's motors

1.2 How does a quadcopter fly

Before we start building our quadcopter, it is important to understand its operation principles. A quadcopter is capable of being maneuvered in six degrees of freedom, allowing movement along three axes and rotation around three points. Specifically, two propellers of the quadcopter turn in a clockwise direction while the two others rotate counterclockwise **(Figure 1.2)**.

When all the motors spin at the same speed and at the same time, they create lift force which is greater than the weight of the drone, causing its lift into the air. Otherwise, the drone descent due to its weight. To move the drone to the left, we decrease the speed of the left motor and increase the speed of the right one. Conversely, to move it to the right, we decrease the speed of the right motor and increase the speed of the left motor. By reducing the speed of the front motors, the drone moves forward. To rotate the drone clockwise, we reduce the speed of the motor that makes it move clockwise. If we want to rotate it counterclockwise, we adjust the motor speed accordingly [2].

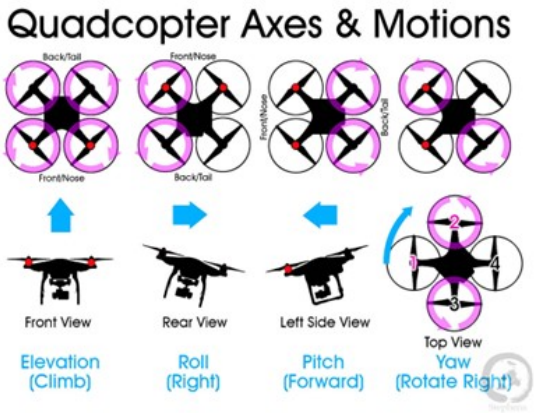


Figure 1.2: Quadcopter movements

1.3 Reference frames

In order to describe physical events occurring in space and time, such as the motion of bodies, we first need to consider two reference frames, as shown in **Figure 1.3**. It is important to understand that both position and orientation in space are relative. The inertial frame is defined by the ground, with gravity pointing in the negative z direction. Meanwhile, the body frame is defined by the orientation of the quadcopter, with the rotor axes pointing in the positive 'z' direction and the arms pointing in the x and y directions [3].

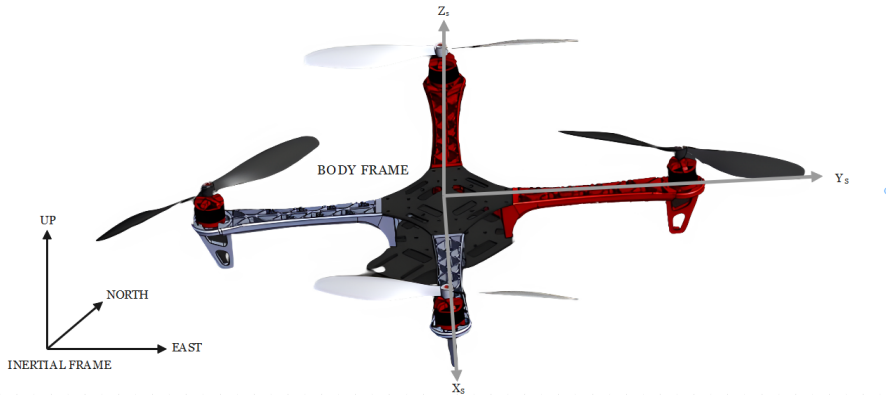


Figure 1.3: The inertial and body frames of a quadcopter

1.4 Quadcopter Kinematics

Before jumping into the physics of a quadcopter's motion, we first need to establish the Kinematics within both the body and inertial frames. The quadcopter's location and velocity are respectively defined in the inertial frame as $\xi = (x,y,z)^T$, and $\dot{\xi} = (\dot{x},\dot{y},\dot{y})^T$. Similarly, we define the roll, pitch, and Yaw angles in the inertial frame as $\eta = (\phi,\theta,\psi)^T$, with corresponding angular velocities equal to $\dot{\eta} = (\dot{\phi},\dot{\theta},\dot{\psi})^T$ [4].

As we explore rotational concepts, we begin by rotating a body frame by a given angle. This seamlessly transitions us into discussing Euler angles, which offer a structured method for representing three-dimensional rotations through sequential elemental rotations about fixed axes [5].

1. **Roll motion**

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{1.1}$$

2. **Pitch motion**

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{1.2}$$

3. **Yaw motion**

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

By multiplying the three previous matrices together, we obtain the rotation matrix which goes from the fixed inertial frame to the mobile frame (**equation 1.4**).

$$R_z(\psi)\cdot R_y(\theta)\cdot R_x(\phi)$$

$$R = \begin{bmatrix} cos(\phi)cos(\psi) & -sin(\psi)cos(\phi)+sin(\phi)cos(\psi)sin(\theta) & sin(\psi)cos(\theta)+cos(\phi)cos(\psi)sin(\theta) \\ cos(\theta)sin(\psi) & cos(\psi)cos(\theta)+sin(\theta)sin(\psi)sin(\phi) & -sin(\theta)cos(\psi)+sin(\theta)sin(\psi)cos(\phi) \\ -sin(\theta) & sin(\phi)cos(\theta) & cos(\theta)cos(\phi) \end{bmatrix} \tag{1.4}$$

A rotation matrix is in the special orthogonal:

$$\begin{cases} R\cdot R^T = R^T\cdot R = I \\ \det(R) = 1 \end{cases} \tag{1.5}$$

Where R^T represents the transpose matrix of R, and I is the identity matrix. Usually, the rate of change of angular positions should be given in angular velocities. However, since the angular

positions and velocities are in a different frame, we need a transformation matrix to switch between reference frames. Let:

- $\dot{\xi} = (\dot{x}, \dot{y}, \dot{z})^\top$ $\dot{\eta} = (\dot{\phi}, \dot{\theta}, \dot{\psi})^\top$: The velocity, and angular velocity of the quadcopter in the inertial frame.
- $\mathbf{V} = (u, v, w)^\top$ $\Omega = (p, q, r)^\top$: The velocity, and angular velocity of the quadcopter in the body frame.

Equation 1.6 gives the relationship between the linear speeds in both the inertial and body frames.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{1.6}$$

Equation 1.7 provides the conversion between the angular rates in the inertial frame and those in the body frame.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sin(\theta) \\ 0 & cos(\phi) & cos(\theta)sin(\phi) \\ 0 & -sin(\phi) & cos(\theta)cos(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{1.7}$$

As discussed previously, we conclude that the velocities observed from different frames are indeed different, but their magnitudes remain the same.

1.5 Forces and torques applied to the quadcopter

In the operation of a quadcopter, various forces and torques act on the system to control its motion and stability.

1.5.1 The forces

A force is a push or pull that can cause an object to accelerate, decelerate, deform, or change direction. It is typically characterized by its magnitude, direction, and the point at which it is applied. One of the most important forces that affects a quadcopter is gravity and thrust force.

1. **Gravity**

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{1.8}$$

   Where m is the total mass, and g is the acceleration due to gravity.

2. **Thrust force**
   Each motor generates an upward force (or thrust along the z-axis), and this force is directly

proportional to ω^2.

$$Thrust = K\omega^2 \tag{1.9}$$

With:

- K: the thrust coefficient.
- ω: the angular velocity of the rotor.

So, the total thrust can be written as:

$$F_t = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ K \cdot (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{bmatrix} \tag{1.10}$$

1.5.2 The Torques

The torque in a quadcopter refers to the rotational force generated by its motors to control its movement and orientation in the air, enabling the quadcopter to rotate around its axes. Torque plays a crucial role in stabilizing and maneuvering the quadcopter during flight. By adjusting the speed of individual motors, pilots can control the torque distribution and achieve desired movements such as pitch, roll, and yaw. Torque is measured in Newton meters $(N \cdot m)$.

- **Roll Torque**
  The roll torque, also known as the yawing moment, is noted as τ_x. It is given by the expression:
  $$\tau_x = l \cdot k \cdot (\omega_4^2 - \omega_2^2) \tag{1.11}$$

- **Pitch Torque**
  The pitch torque is the result of the difference in thrust between rotors 3 and 1, its expression is given by:
  $$\tau_y = l \cdot k \cdot (\omega_3^2 - \omega_1^2) \tag{1.12}$$
  where,
  - l represents the distance between the center of mass and each rotor.
  - k is a lift constant.
  - ω_i denotes the angular velocity of each rotor.

- **Yaw Torque**
  The yaw torque is generated by the difference in thrust between diagonally opposite rotors. The expression for yaw torque, denoted as τ_z, can be mathematically represented as:
  $$\tau_z = b \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{1.13}$$
  Here, b is a drag constant.

The overall moments are described as:

$$\tau_t = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l \cdot K \cdot (\omega_4^2 - \omega_2^2) \\ l \cdot K \cdot (\omega_3^2 - \omega_1^2) \\ b \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{1.14}$$

1.5.3 Moment of Inertia

Moment of inertia, often referred to as inertia tensor or rotational inertia, is a measure of an object's resistance to changes in rotational motion about a particular axis. In simpler terms, it indicates how difficult it is to change the rotational motion of an object. We refer to **Figure1.4** to determine the moment of inertia along the x, y, and z axes.

The inertia can be estimated by considering a dense spherical core with mass M and radius R, along with individual point mass of the motor m located at a distance l from the center [6].

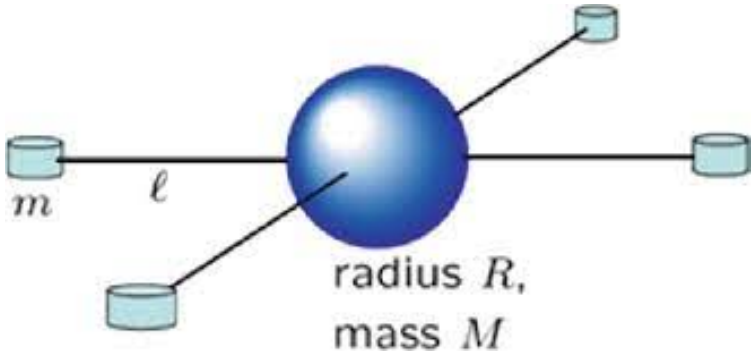


Figure 1.4: Moment inertia

$$I_{xx} = \frac{2MR^2}{5} + 2ml^2 \tag{1.15}$$

$$I_{yy} = \frac{2MR^2}{5} + 2ml^2 \tag{1.16}$$

$$I_{zz} = \frac{2MR^2}{5} + 4ml^2 \tag{1.17}$$

Given that I_{xx} is equal to I_{yy}, the inertia matrix is considered perfectly symmetrical.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{1.18}$$

1.6 The dynamic model

Dynamic modeling involves obtaining the differential equations that describe how a quantity changes over time. It includes factors such as the mass, inertia, and applied forces and torques that cause motion. These equations are derived in mechanics to understand how objects move and rotate. Dynamics explores the causes of motion and how these forces influence the motion of objects. The study of the causes of motion involves principles such as **Euler-Lagrange** or **Newton-Euler** equations [7].

The Lagrange formalism describes the behavior of a dynamic system in terms of energy. The Lagrange equations are typically written in the form:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \Gamma \tag{1.19}$$

Where L is the Lagrangian of the system. It is defined as the difference between the kinetic energy T and the potential energy U.

$$L = T - U \tag{1.20}$$

To find the overall dynamic equation for the quadcopter, we start by identifying its kinetic and potential energies, as well as the lagrangian. Then, we insert these values into the Lagrange equation. We use the symbol q to represent the generalized coordinates, which include both the Euler angle $\eta = (\phi, \theta, \psi)^T$ and linear position $\xi = (x, y, z)^T$.

$$\mathbf{q} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \tag{1.21}$$

- **The kinetic energy**
  The kinetic energy consists of two components, the first is the kinetic energy associated with translation, while the second corresponds to the kinetic energy related to rotation (**equation 1.22**).

$$T = T_{\text{trans}} + T_{\text{rot}} \tag{1.22}$$

1-The kinetic energy of Translation

$$T_{\text{trans}} = \frac{1}{2} m \dot{\xi}^T \dot{\xi} \tag{1.23}$$

$$T_{\text{trans}} = \frac{1}{2} m \dot{x}^2 + \frac{1}{2} m \dot{y}^2 + \frac{1}{2} m \dot{z}^2 \tag{1.24}$$

Here, m represents the mass.

2-The kinetic energy of Rotation

$$T_{\text{rot}} = \frac{1}{2} I \dot{\eta}^T \dot{\eta} \tag{1.25}$$

with, I represent the inertia matrix.

$$T_{\text{rot}} = \frac{1}{2} I_{xx} \dot{\phi}^2 + \frac{1}{2} I_{yy} \dot{\theta}^2 + \frac{1}{2} I_{zz} \dot{\psi}^2 \tag{1.26}$$

- **The potential energy**

$$U = m \cdot g \cdot z \tag{1.27}$$

Where m stands for the mass of the quadcopter, z is the altitude, and g is the acceleration.

- **The external force**
The external forces are applied to the quadcopter so, we need to multiply the thrust force by the rotation matrix R.

$$F_{ext} = R \cdot \begin{bmatrix} 0 \\ 0 \\ K \cdot (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{bmatrix} \tag{1.28}$$

- **External forces and torque vectors**

$$\Gamma = \begin{bmatrix} F_{\text{ext}} \\ \tau_{\text{ext}} \end{bmatrix} = \begin{bmatrix} F_z (\sin(\psi)\cos(\theta) + \cos(\phi)\cos(\psi)\sin(\theta)) \\ F_z (-\sin(\theta)\cos(\psi) + \sin(\theta)\sin(\psi)\cos(\phi)) \\ F_z (\cos(\theta)\cos(\phi)) \\ l \cdot K \cdot (\omega_4^2 - \omega_2^2) \\ l \cdot K \cdot (\omega_3^2 - \omega_1^2) \\ b \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{1.29}$$

With F_z, the thrust force is along the z-axis.

So, by integrating the expressions of T and U into equation (**1.20**), we obtain:

$$L = \frac{1}{2} m \dot{x}^2 + \frac{1}{2} m \dot{y}^2 + \frac{1}{2} m \dot{z}^2 + \frac{1}{2} I_{xx} \dot{\phi}^2 + \frac{1}{2} I_{yy} \dot{\theta}^2 + \frac{1}{2} I_{zz} \dot{\psi}^2 - mgz \tag{1.30}$$

1. **Translation equations**

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = m\ddot{x} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}}\right) - \frac{\partial L}{\partial y} = m\ddot{y} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{z}}\right) - \frac{\partial L}{\partial y} = m\ddot{z} - mg \end{cases} \tag{1.31}$$

2. **Rotation equations**

$$\begin{cases} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \frac{\partial L}{\partial \phi} = I_{xx}\ddot{\phi} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = I_{yy}\ddot{\theta} \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \frac{\partial L}{\partial \psi} = I_{zz}\ddot{\psi} \end{cases} \tag{1.32}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \begin{bmatrix} m\ddot{x} & m\ddot{y} & m\ddot{z} - mg & I_{xx}\ddot{\phi} & I_{yy}\ddot{\theta} & I_{zz}\ddot{\psi} \end{bmatrix}^T \tag{1.33}$$

The quadcopter model, using the Euler-Lagrange formalism, is:

$$\begin{cases} \ddot{x} = \frac{F_z}{m}\left(\sin(\psi)\cos(\theta) + \cos(\phi)\cos(\psi)\sin(\theta)\right) \\ \ddot{y} = \frac{F_z}{m}\left(-\sin(\theta)\cos(\psi) + \sin(\theta)\sin(\psi)\cos(\phi)\right) \\ \ddot{z} = \frac{F_z}{m}\left(\cos(\theta)\cos(\phi)\right) + g \\ \ddot{\phi} = \frac{l \cdot K}{I_{xx}}(\omega_4^2 - \omega_2^2) \\ \ddot{\theta} = \frac{l \cdot K}{I_{yy}}(\omega_3^2 - \omega_1^2) \\ \ddot{\psi} = \frac{b}{I_{zz}}(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{cases} \tag{1.34}$$

1.7 TCP and UDP Protocols

Both OSI (Open Systems Interconnection) and TCP/IP (Transmission Control Protocol/Internet Protocol) models are used to understand how the various components of a computer network communicate with each other. The OSI model has seven layers, each with its specific functionality, while the TCP/IP model combined several layers into one, as shown in **Figure 1.5**.

Inside both models is the transport layer, which is responsible for reliable communication between devices in the network. It sits above the network layer, handling routing and addressing, and below the session layer, managing the establishment and termination of connections.

The transport layer plays a crucial role in network communication by providing two fundamental protocols, TCP and UDP [8].

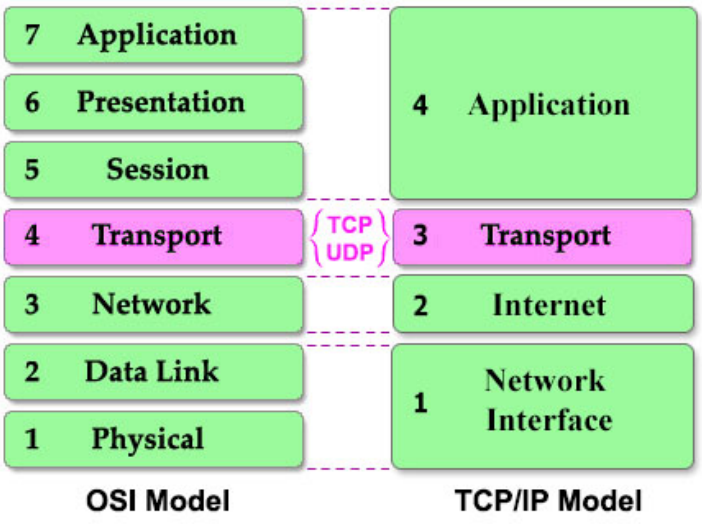


Figure 1.5: OSI and TCP/IP layers

1.7.1 Transmission Control Protocol(TCP)

TCP is a connection-oriented protocol, this basically means that the session must first be acknowledged between two communicating terminals. Therefore the terminal checks the connection before communication occurs. This is done via a three-way handshake.
Here is how TCP works:

1. A computer will send a serial number (SYN).
2. The receiving computer will send back a serial number as confirmation, Often called (SYN-ACK), telling the sender that it a confirmation message is received.
3. Finally, the sender sends another confirmation message to the receiver.

Once this process is completed, data can be sent as shown in **Figure 1.6** below. Another important aspect of TCP to keep in mind is that every time a data packet is sent, it requires a confirmation from the receiver. Therefore, if no confirmation is received, the data is sent again [9] [10].

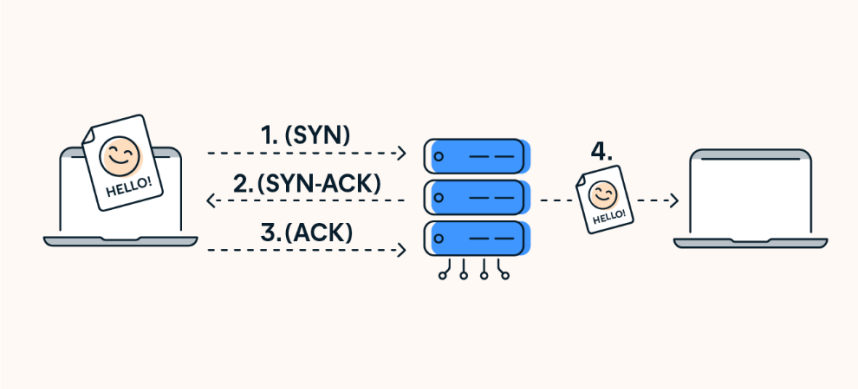


Figure 1.6: Three-way handshake

1.7.2 User Datagram Protocol (UDP)

UDP is quite similar to TCP, it is used for sending and receiving data as well. However, the main difference is that UDP is connectionless, which means it does not create a session and does not confirm data delivery. This is why UDP is sometimes referred to as the **Fire and Forget** protocol [10].

1.7.3 Key Differences between TCP and UDP

Let us highlight the differences between TCP and UDP

TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
1. Connection-oriented protocol. 2. Provides a good and reliable data transmission. 3. Uses a three-way handshake. 4. It requires confirmation that data arrives in the correct order. 5. Supports only unicast (one-to-one) 6. Ideal for applications where data is important, such as web browsing, Email or texting, and file transfer.	1. Connectionless protocol. 2. Does not guarantee data delivery. 3. Does not create a session before sending data. 4. Does not confirm whether the data arrives 5. Supports broadcast and multicast. 6. Ideal for applications that prioritize speed over reliability, like Video chat.

Table 1.1: Comparison between TCP and UDP

In this project, UDP was chosen over TCP for live streaming telemetry data using ESP8622 with Pixhawk board. This choice was made because UDP is faster and well suited for real-time applications compared to TCP. This decision aimed to make sure that the live streaming of telemetry data between the drone and a laptop was smooth and responsive.

1.8 PID controller

In the world of control and automation, systems are categorized as either linear or nonlinear. In everyday applications, most systems show nonlinear behavior. A quadcopter is a prime example of a nonlinear system due to its complex dynamics involving motors, propellers, and sensors. The main idea in control systems is to find the right signal to control a specific variable effectively. The most widely used method for this purpose is the PID controller, chosen for our project because it's straightforward and widely used across different industries. PID stands for Proportional-Integral-Derivative and operates within a closed-loop system, as shown in **Figure 1.7**. This controller uses a formula to continuously compare the actual feedback with the desired setpoint over time. It then adjusts to minimize any differences and aims to reduce them to zero [11] [12].

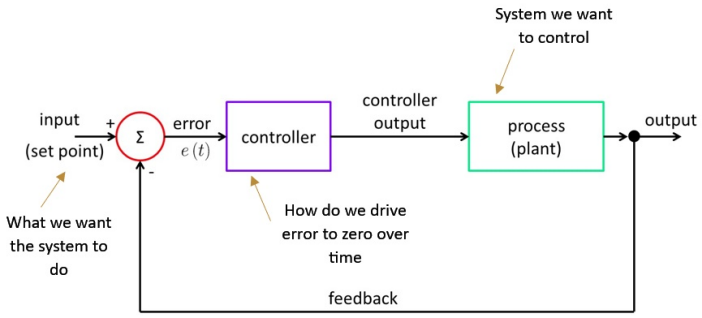


Figure 1.7: closed-loop control system

1.8.1 Proportional

The proportional term is defined as shown in **equation 1.35**, where it multiplies the current error $e(t)$ by the gain K_p.

$$P = K_p \cdot e(t) \tag{1.35}$$

Since the proportional term amplifies the error, it creates a constant error known as a steady-state error. Once the system reaches the desired response, the proportional term becomes inactive, which can potentially lead the drone to crash. This error can be eliminated and such events avoided by adding an integral term to our system, which enables the controller to use past information.

1.8.2 Integral

The integrator term adds up the input signals, so it remembers what happened in the past.

$$I = K_i \int e(t)\, dt \tag{1.36}$$

where K_i is the integral constant.

If the integral part keeps accumulating, it may cause the system to go higher than our goal, which is called overshot. To solve this we need to predict the future and respond to how we are closing in on our goal by letting our controller system use derivative term.

1.8.3 Derivative

The derivative term is used to measure the rate of change of the target error, so it helps manage the speed at which the target error changes.

$$D = K_d \cdot \frac{de(t)}{dt} \tag{1.37}$$

Now that we have covered the three terms of the PID controller, the output of the PID controller is expressed as:

$$U(t) = K_p \cdot e(t) + K_i \int e(t)\,dt + K_d \cdot \frac{de(t)}{dt} \tag{1.38}$$

The corresponding diagram of the control system is shown in **Figure 1.8**.

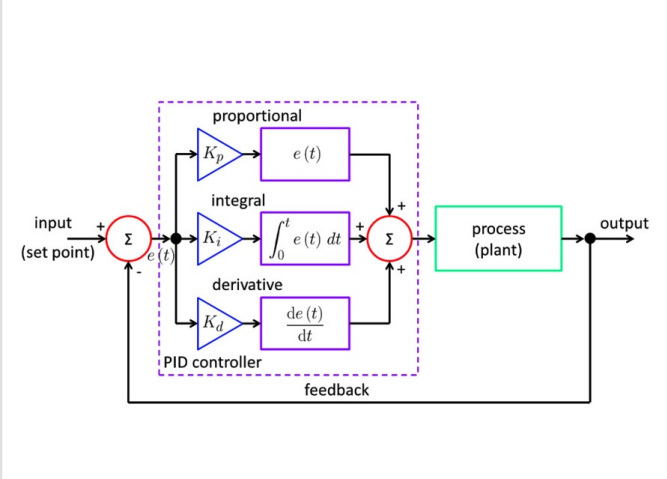


Figure 1.8: Block diagram of PID controller

1.9 Conclusion

In this chapter we have covered the fundamental concepts related to our project, namely: modeling the quadcopter using subsystems (Kinematics and dynamics), communication protocols, and PID controllers. We have presented the two configuration modes of a quadcopter and the difference between them. The X configuration have been chosen due to its stability and maneuverability. Now as we have understood the required basics, We are ready to move on to the next chapters, where we will explore the practical implementation steps. This will cover not only the components of the drone but also other tools used in computer vision.

Chapter 2

Practical Experience

2.1 Introduction

When building a quadcopter, one of the primary steps to consider is choosing the appropriate motor. To make the drone fly better and longer, it is important to try multiple motors and pick the most compliant with our specifications. This chapter describes the components of our quadcopter with more focus on motors, their operation mechanism, and the associated control electronics. Thus, we will see the role of Electronic Speed Controllers.

2.2 Components of Drone Hardware

A drone, specifically a quadcopter, is composed of several essential components that work together to enable flight and various functionalities. These components include the frame, brushless DC motors, electronic speed controllers (ESCs), propellers, a power distribution board (PDB), a flight controller, a transmitter and receiver system, a camera, and a battery. Each of these parts plays a critical role in the operation and performance of the drone. In this chapter, we will provide a description of each component, detailing their function, construction, and significance in ensuring the drone operates efficiently and effectively. The frame provides the structural foundation, while the motors and ESCs work in tandem to control the propellers and generate thrust. The PDB ensures power is appropriately distributed to all components, and the flight controller acts as the brain, processing inputs and stabilizing the drone. The transmitter and receiver system facilitates remote control, the camera captures visual data, and the battery supplies the necessary energy to keep the drone in flight.

2.3 Frame

A drone's frame is a structure made from lightweight materials like carbon fiber, aluminum, or plastic, designed to hold all components together. The shape and size depend on the number of propellers, with quadcopters (four propellers), hexacopters (six), and octocopters (eight)

being common types. Design considerations include weight, strength, and aerodynamics to enhance flight efficiency and durability. Frames often have mounting points for motors, propellers, and other components, and can be customized for specific purposes like racing or photography.

2.4 Brushless DC Motors

A Brushless DC motor is a kind of electric motor which can be either 'inrunner' or 'outrunner' as shown in **Figure2.1**. This motor is chosen for its reliability, minimal noise, and low maintenance compared to alternatives.

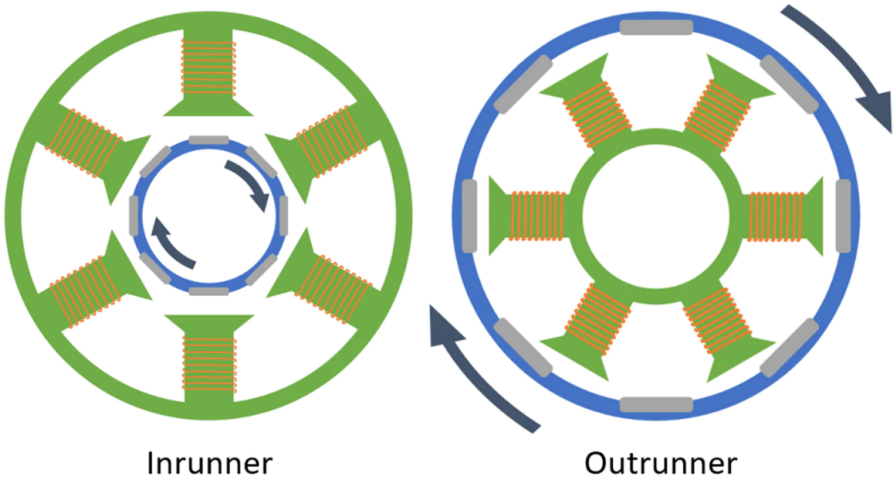


Figure 2.1: Inrunner and Outrunner brushless motor

2.4.1 Brushless Motor Components

Brushless motors are made of many key components, each plays an important role in the motor's operation and reliability. By understanding its components we can choose the right motor for our applications. Here is an overview of the main components.

2.4.1.1 Electromagnets and Permanent Magnets

Both electromagnets and permanent magnets generate magnetic fields. They have north and south poles and interact with external sources that have magnetic fields. Electromagnets, as shown in **Figure 2.2**, do not induce magnetic fields unless there is an electric current flowing through them, unlike permanent magnets that do not require an electric current [13].

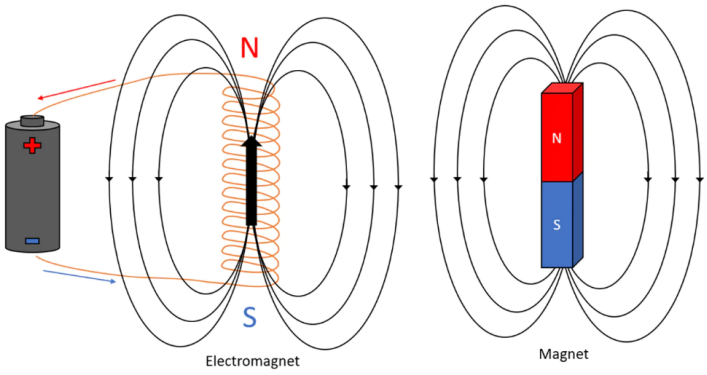


Figure 2.2: Electromagnets and permanent magnets

The stator is the fixed component of the motor which generate a dynamic magnetic field. It consists of a magnetic circuit comprised of windings or electromagnets. **Figure 2.3** shows the internal organization of a brushless motor, the stator is in green color.

The rotor is the rotating part of the motor and it typically consists of permanent magnets, which are attached to its shaft. These magnets create a magnetic field that works together with the stator's magnetic field to make the motor turn **(Figure 2.3)**.

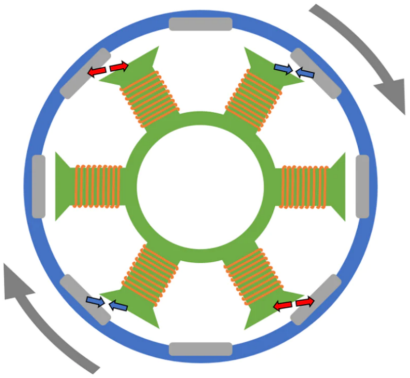


Figure 2.3: Brushless motor Diagram

As seen above, the stator and rotor are separated by a gap. This air gap allows for the efficient transfer of magnetic flux between the stator and rotor. The smaller the gap, the better the transformation of magnetic flux.

2.4.2 Brushless Motor Mechanism

A brushless motor typically has multiple phases, often three. When electricity is applied to the coils, the stator generates a magnetic field that attracts the permanent magnets, which makes it spin. As the motor rotates, it also generates its own voltage known as back electromotive force (EMF). This generated voltage opposes the voltage we supply to the motor.

In brushless motors, back electromotive force is important for controlling the motor's speed and position, which are essential for the synchronous commutation of the motor's phases.

In a three-phase Brushless Motor, there is a precise six-step excitation sequence of the electromagnets. Each step of this commutation sequence is achieved by supplying a positive current to one of the windings, a negative current to the second, and leaving the third winding open [14].

Table 2.1 shows a simplified schematic of a six-stage commutation. This cyclic sequence continues until the motor is powered off.

Step	1	2	3	4	5	6
High	A	B	B	C	C	A
Low	C	C	A	A	B	B

Table 2.1: Commutation sequence for a three-phase Brushless Motor

Figure 2.4 Shows a simplified schematic of a three-phase Brushless Motor and also shows the situation where phase A is supplied with a positive current, while phase C is supplied with a negative current.

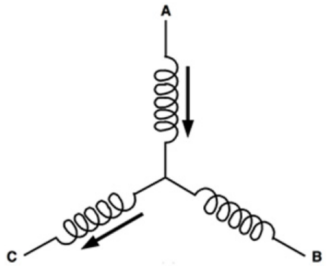


Figure 2.4: The initial stage of a six-step electrical cycle for a Brushless motor

2.4.3 Performance and characteristics of a brushless motor

Understanding the performance and characteristics of brushless DC motors is important for selecting the most suitable motor for our project. It helps us determine some key parameters such as flight time and payload capacity, which are crucial for building a drone that can lift with the required weight and for the desired duration.

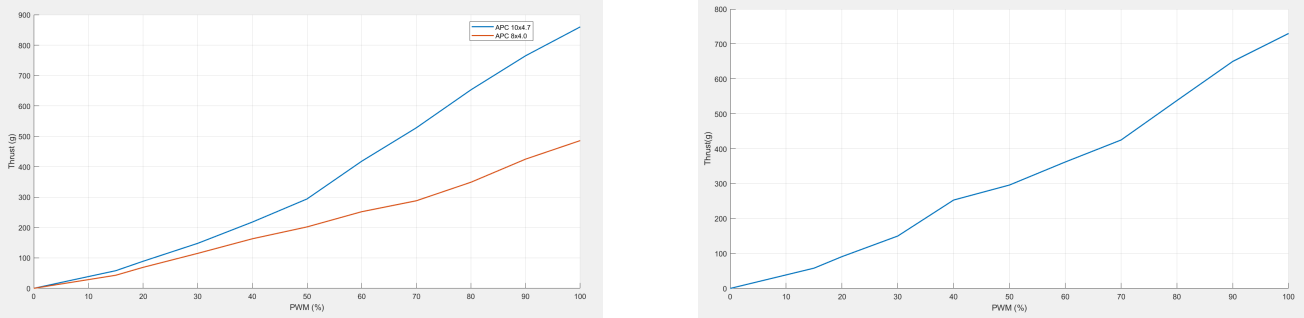
The **Figure 2.5** shows a setup for testing a brushless motor with a propeller, commonly used in RC models like drones or airplanes. It includes a Taranis Plus transmitter for controlling the motor, a receiver connected to the transmitter, and a yellow component with blue wires connected to the motor, which is an electronic speed controller (ESC) that regulates power to the motor. The brushless motor with an attached propeller is mounted on a wooden block. Additionally, there is a magnet connected to a DC motor to assist a Hall effect sensor in detecting the magnetic field. A power meter with a black cover is also integrated into the setup to monitor electrical consumption. An Arduino board is connected to a laptop via cable, used for calculating the RPM (Revolutions Per Minute) value. The setup is powered by a Li-Po battery. The only difference between **Figure 2.5(a)** and **Figure 2.5(b)** is the type of motor: the first figure uses a brushless DC motor with 980KV, while the second figure uses one with 1400KV.

(a) 980KV DC motor (b) 1400KV DC motor

Figure 2.5: Different types of brushless motors

To measure the thrust value, we need to consider two factors:

1. the propeller pushing air downward.
2. The motor rotating in the same direction as the propeller.



(a) 980KV DC motor (b) 1400KV DC motor

Figure 2.6: Thrust Comparison of 980KV Motor with Different Propeller Sizes, and 1400KV motor

The **Figure 2.6 (a)** shows the importance of propeller size in determining the thrust produced by a 980KV motor. A 10×4.7 propeller gives better results, where the motor can lift 860g at maximum speed. On the other hand, **Figure 2.6 (b)** indicates that the 1400KV motor produces 730g of thrust with its own propeller (8×4.0) [15].

To measure RPM (Revolutions Per Minute), we use a Hall effect sensor **(Figure 2.7)**. This sensor operates by sensing the magnetic field created by magnets that we can attach to the rotor of the motor. As the motor rotates, the magnets pass by the sensor field, so it generate a digital signal each time a magnet passes. By counting these pulses over a specific period, we can accurately determine the motor's RPM using **(equation 2.1)**.

$$\text{RPM} = \frac{\text{Number of pulses}}{t} \times 60 \tag{2.1}$$

Where the multiplication by 60 converts the frequency of pulses per second (pulses per second) into RPM.

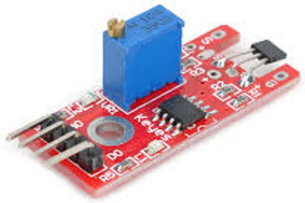


Figure 2.7: Hall Effect Sensor

As shown by **Figure 2.8** the RPM varies linearly with the duty cycle of the applied PWM signal (Pulse Width Modulation). Additionally, the motor with a 1400KV spins faster than the one with a 980KV [15].

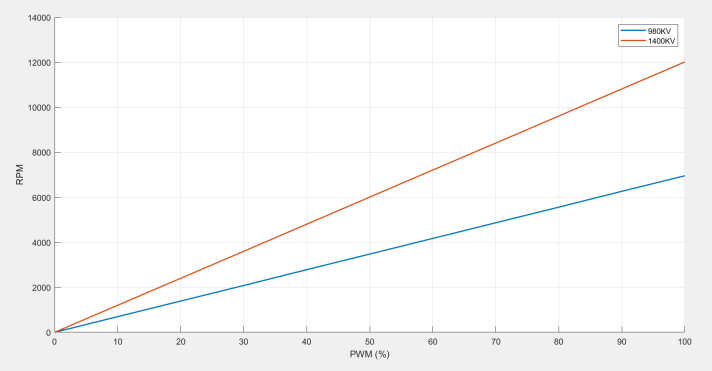


Figure 2.8: RPM for Different Motor Configurations

Figure 2.9 shows the power consumption for each motor. As the motor speed increases, power consumption also increases. The motor with a 980KV consumes less power compared to the other motor [15].

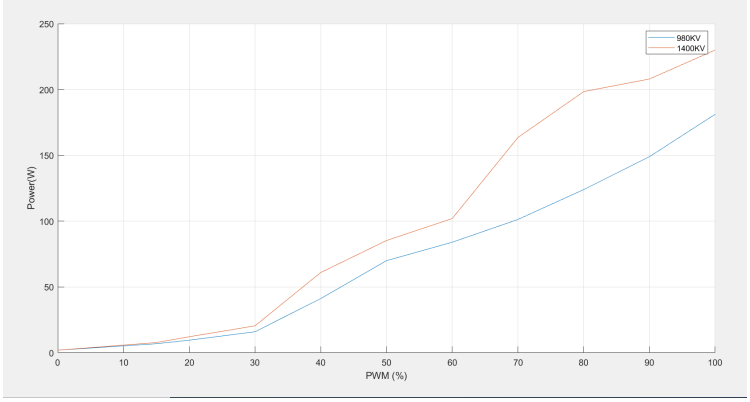


Figure 2.9: Power consumption for each motor.

2.5 Role of Electronic Speed Controllers(ESC)

An ESC, short for Electronic Speed Controller, uses MOSFET transistors to control the speed and direction of a brushless motor. Six MOSFETs are typically used for driving three-phase Brushless motors, with two MOSFETs allocated per phase. To control the motor effectively, the ESC needs to know the position of the motor's rotor. This is necessary because brushless motors do not have mechanical brushes to change the direction of current. Instead, this task is done by monitoring the back-EMF generated in the motor's coils. The **Figure 2.10** below shows how an ESC switches between different stages.

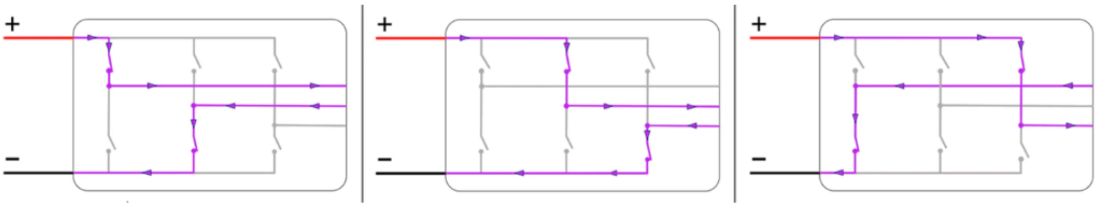


Figure 2.10: Example of different stages controlled by the ESC

2.6 Power Distribution Board (PDB)

A PDB (Power Distribution Board) functions similarly to a circuit board, specifically designed to efficiently distribute power to different components within a system, such as a quadcopter or other electronic devices. One of its primary roles is to manage and deliver power to essential components like the Electric Speed Controller (ESC), which controls the speed of the motors. The PDB typically features multiple input and output points where power can be supplied from a battery or another power source. It ensures that each component receives the appropriate voltage and current, minimizing power fluctuations and optimizing the overall performance and stability of the system.

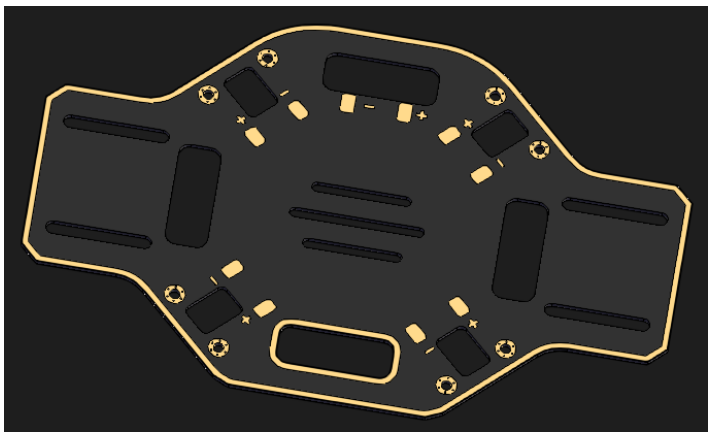


Figure 2.11: Power Distribution Board

2.7 Propellers

Propellers are a basic component of any quadcopter, as they directly influence the drone's ability to lift and navigate during flight. Propellers can be made from different materials, including plastic, carbon fiber, and wood. Plastic propellers are lightweight and cheap, while carbon fiber propellers are stronger and provide better performance but cost more.

The size of the propeller, usually measured in inches, affects thrust and efficiency. Larger propellers create more lift, which is good for carrying heavier loads, but they also require more power from the motors. Smaller propellers are more efficient at higher speeds.

Most quadcopters use two-blade propellers, but three-blade and even four-blade propellers are also available. More blades can increase thrust and stability but also create more drag and reduce efficiency.

Propellers are designed to rotate either clockwise (CW) or counterclockwise (CCW), and quadcopters need both types to stay balanced and controlled, as shown in **Figure 2.11** below.

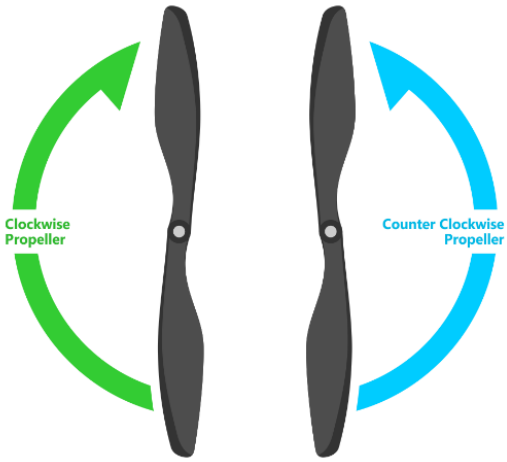


Figure 2.12: Propellers direction

2.8 Flight Controller

The flight controller or 'Autopilot' is the brain of our drone. There are different types available on the market, but for our quadcopter, we chose a Pixhawk **(Figure 2.13)**. It works along with a 'Ground Control Station' software named Mission Planner which is open source. Pixhawk is one of the most popular choices due to its stability and reliability. In addition, it provides a precise control over various types of vehicles such as planes, rovers, quadcopters, hexacopters, and octocopters.

Figure 2.13: Pixhawk 2.4.8

2.8.1 Pixhawk Specification

In what follows, we summarize the main features of the Pixhawk autopilot system. The last is basically a processor board which integrate the required interfaces and sensors to manage the drone's operation [16].

Processor

- 32-bit ARM Cortex M4 core with FPU.
- 168 MHz/256 KB RAM/2 MB Flash.
- 32-bit failsafe co-processor.

Sensors

- MPU6000 as main accelerometer and gyroscope.
- ST Micro 16-bit gyroscope.
- ST Micro 14-bit accelerometer/compass (magnetometer).
- MEAS barometer.

Interfaces

- 5x UART serial ports.
- Futaba S.BUS input.
- PPM sum signal.
- RSSI (PWM or voltage) input.

- I2C, SPI, 2x CAN, USB.
- 3.3V and 6.6V ADC inputs.

Dimensions

- Weight: 38 g.
- Width: 50 mm.
- Height: 15.5 mm.
- Length: 81.5 mm.

Additionally, Pixhawk provides various flight modes, including commonly used ones like stabilized flight, altitude hold, loiter mode, and others. More details about it are provided within the Master's report.

2.8.2 Sensors

As we discussed previously, Pixhawk is equipped with a set of sensors which are mainly used to determine vehicle's state. This is important for drone's stabilization and allows it to be fully autonomous.

1. **Gyroscope:**
   A gyroscope is a device used to measure or maintain orientation and angular velocity, allowing the controller to make adjustments in order to keep the drone stable during flight. It operates based on the principle of angular momentum.

2. **Accelerometer:**
   Accelerometers measure the drone's linear acceleration along x, y, and z axes. They help in detecting changes in velocity and acceleration, which are essential for maintaining stability, especially during maneuvers or windy conditions.

3. **Magnetometer:**
   Magnetometers help to determine the drone's heading or direction by detecting the Earth's magnetic field.

4. **Barometer (Barometric Pressure Sensor):**
   Barometers measure the atmospheric pressure, which can be useful to estimate the drone's altitude above sea level.

5. **GPS (Global Positioning System):**
   GPS sensors are devices which receive latitude and longitude data from GPS satellites by mean of electromagnetic signals. This helps to control autonomous flight and extends drone's functionalities, such as waypoint navigation, and return-to-home.

2.9 Transimtter and Receiver

A radio command is a handheld device used to wirelessly control a vehicle. The last should be equipped with a compatible radio receiver which acquires commands that manage actuating components operation, such as motors and servos. Creating a connection between the radio command and the receiver, known as binding, is a necessary process that needs to be done first.

A typical radio command consists of two joysticks equipped with a radio module, and sometimes a display screen as shown in **Figure 2.14**. Various types of radio command systems are available, such as Futaba, FlySky, and FrSky. An important factor to consider when choosing a system is the number of channels and the supported protocols, such as PPM, S-BUS, and I-BUS.

1. **Roll and Pitch Stick**
   The roll and pitch sticks control the angle of the drone around the horizontal axes. These sticks allow the operator to tilt the drone forward, backward (pitch), or sideways (roll), influencing its direction of movement and orientation during flight.

2. **Yaw and Throttle Stick**
   The yaw stick on a drone controls the rate of rotation around the vertical axis, enabling the drone to turn left or right. It adjusts the direction the drone faces without changing its position in the air. Conversely, the throttle stick regulates both the altitude and speed of the drone. Increasing throttle lifts the drone higher and speeds up its movement, while reducing throttle lowers its altitude and decreases speed, affecting both ascent and descent during flight.

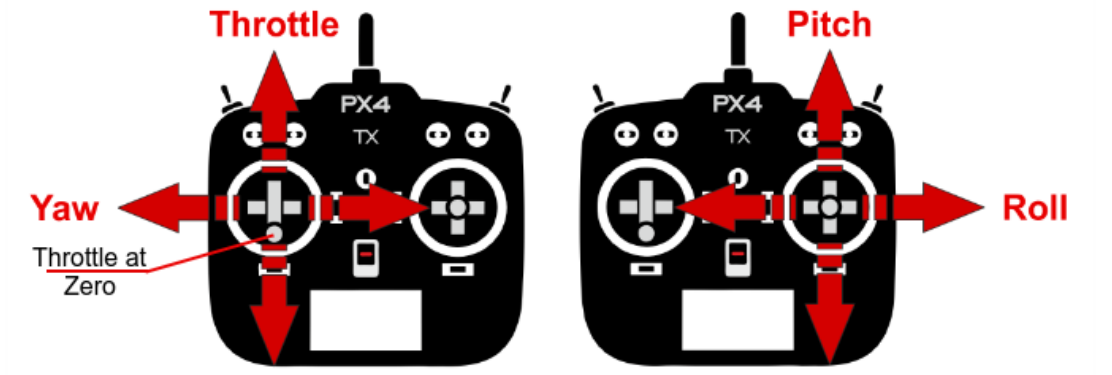


Figure 2.14: Remote Control

2.9.1 S-BUS signal

The S-BUS protocol was developed by Futaba for hobby remote control applications. It is derived from the RS232 protocol, but the voltage levels are inverted. This protocol allows to transmit 8 bits data words along with a parity bit and two stop bits with a rate of 100K bauds. The S-BUS packet consists of 25 bytes which are described as follows [17].

- **Byte[0] = 0x0F**: SBUS header.
- **Byte[1 - 22]**: 16 servo channels, 11 bits each.
- **Byte[23]**: Two digital channels (17, 18), frame lost, and failsafe.
- **Byte[24] = 0x00**: SBUS footer (the end of the transmission).

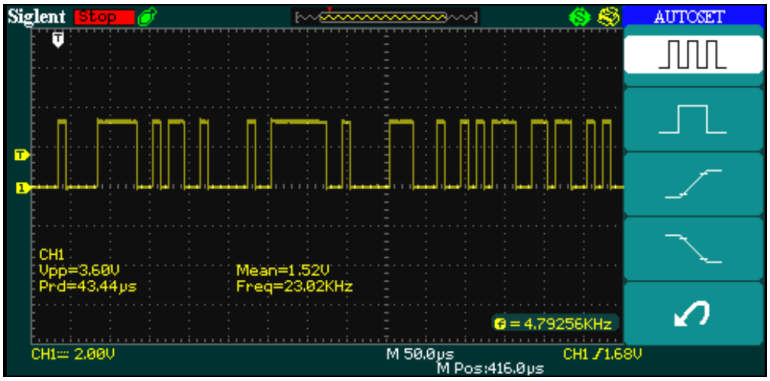


Figure 2.15: S-BUS signal

We visualized a S-BUS signal using an oscilloscope, as shown in the **Figure2.15**. The oscilloscope allowed us to observe the characteristics of the signal, displaying its waveform in detail.

2.10 Camera

A camera is an electronic device, also known as a black box, that captures and transforms the three-dimensional world into a two-dimensional image by letting light pass through a lens (**Figure 2.16**). A camera's properties and capabilities are determined by its intrinsic and extrinsic parameters.

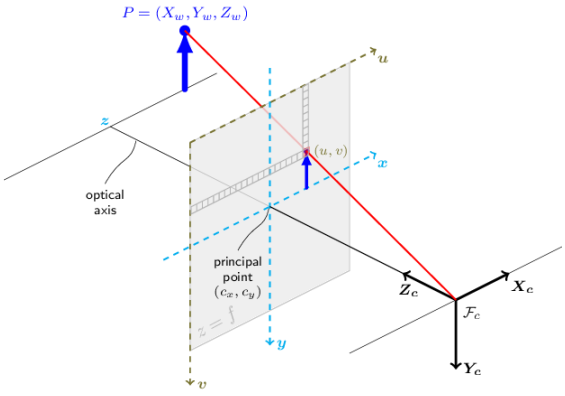


Figure 2.16: Pinhole camera model

2.10.1 Intrinsics

Intrinsic parameters refer to the internal characteristics of the camera, which include the focal length, optical center, and lens distortion coefficients. The following matrix enables the conversion of 3D world coordinates to 2D pixel coordinates using the pinhole camera model.

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Where:
- f_x and f_y are the focal lengths of the camera in the x and y directions, respectively.
- c_x and c_y are the coordinates of the principal point (optical center) of the image, typically located at the center of the image.

This matrix is used for various computer vision tasks such as camera calibration which helps to correct distortions to ensure image accuracy and pose estimation [18].

2.10.2 Extrinsics

Extrinsic parameters describe the camera's pose (position and orientation) in the 3D world. In other words, they determine where the camera is located and how it is oriented relative to the scene being captured. This involves translation along the X, Y, and Z axes, as well as rotations around these axes, commonly known as pitch, roll, and yaw.

2.11 Battery

A LiPo (Lithium Polymer) battery, known for its lightweight and high energy density, is widely utilized in electronic projects such as drones and RC vehicles. These batteries pack more energy into a smaller and lighter package compared to traditional battery types, making them ideal for applications where weight and space are critical factors. However, LiPo batteries require careful handling during charging and use to ensure safety and longevity. One critical component used in charging LiPo batteries is a balancer. A balancer ensures that each cell within the battery pack is charged evenly, preventing overcharging or undercharging that could lead to performance degradation or safety hazards. This balancing process is essential for maximizing the lifespan and reliability of LiPo batteries, ensuring they deliver optimal performance in demanding electronic applications.

The **Figure 2.17** represents that the numbers 3300mAh and 25C refer to two important specifications:

- 3300mAh (milliampere-hour): This indicates the battery's capacity to store electrical energy. Specifically, it means the battery can provide a current of 3300 milliamperes for one hour before it is fully discharged.

Figure 2.17: LI-PO Battery

- 25C: This indicates the discharge rate capability of the battery. It represents how quickly the battery can discharge its stored energy. In this case, 25C means the battery can discharge at a rate that is 25 times its capacity (in this case, 3300mAh). So, the maximum discharge current would be 25×3300 mA, which equals 82,500 mA or 82.5 amps.

Regarding flight time, our drone operates for between 6 and 8 minutes. This duration depends on factors such as the weight of the drone, its speed, and the efficiency of components, including the battery.

2.12 Computer Vision

In the context of technology, computer vision refers to the ability of machines to capture and interpret visual information from the environment. This is done using cameras and computer algorithms. Vision systems are useful in many fields, such as pattern recognition, motion analysis, and more.

In our project, we use computer vision to get information about drone's location and orientation relative to a reference object that can be detected by the camera. For that, we exploit ArUco markers which are small, black-and-white squares that can be easily detected and identified by their unique identifiers (IDs). This allows the drone to perform tasks like hovering, landing, and reaching target locations [19].

To implement such process we make use a very common library for computer vision: OpenCV, which includes built-in support for detecting ArUco markers, as shown in **Figure 2.18** below.

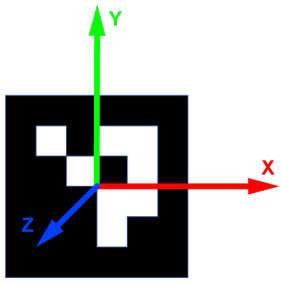


Figure 2.18: Aruco marker with 3D axis

2.12.1 OpenCV

OpenCV, which stands for Open Source Computer Vision, is a free library designed for computer vision and machine learning. Its purpose is to help in the development of computer vision tools and to simplify their application in various fields [20]. Currently, OpenCV supports multiple programming languages such as C++ and Python and is compatible with several platforms, including Windows and Linux. In what follows, we cite some of development facilities provided by OpenCV.

1. **Image and Video Processing:**
   OpenCV is a powerful library for handling images and videos in computer vision applications. It provides fundamental functions such as resizing images to specific dimensions, converting between different color spaces (like RGB and grayscale), reading images from files or cameras, saving processed images back to files, and displaying images or video streams on screens or graphical user interfaces (GUIs). These capabilities are essential for tasks ranging from basic photo editing and format conversion to real-time video analysis and computer vision algorithms.

2. **Geometric Transformations:**
   OpenCV allows performing geometric transformations on images, such as scaling, rotating, and translating. These transformations are essential for tasks like image registration, correcting image alignment, or adjusting perspectives.

3. **Image Filtering:**
   OpenCV offers various image filtering techniques, including blurring, sharpening, noise reduction, and edge detection. Image filters help in enhancing image quality, reducing noise interference, and extracting important features from images for further analysis.

4. **Camera Calibration:**
   OpenCV includes functionalities for calibrating cameras to extract parameters such as focal length and camera center. Camera calibration is crucial in robotics, drone applications, and computer vision tasks where accurate measurements and spatial understanding are required.

5. **Object Detection:**
   OpenCV provides algorithms for recognizing objects within images or real-time video streams. These algorithms are used extensively in applications such as object tracking, surveillance systems, and autonomous vehicles to identify and locate specific objects of interest.

2.13 Conclusion

In this chapter, a comprehensive overview of our drone's components is presented. Understanding brushless DC motors operation principle, electronic speed control and propellers, flight principles, and other hardware elements is an important step for designing and optimizing our drone's performance. Additionally, cameras, various sensors, and vision technologies are discussed as they play a significant role in helping drones to perform tasks efficiently.

Chapter 3

Implementation

3.1 Introduction

In this chapter, we present roughly and step by step the building process of our drone. First, we begin with the assembly of the drone. Since we have already discussed the required components in the previous chapter, we will now see how to put these parts together. Next, we proceed to software implementation using OpenCV library, a powerful tool that enables us to estimate the drone's position by mean of ArUco markers. We will finally show the various experimentations in both manual and automatic modes and discuss obtained results and confronted difficulties.

3.2 Step-by-Step Assembly

To begin the process of building the quadcopter, we will first justify our choices about various components that we use. Each component was selected based on criteria such as performance, compatibility, reliability, and cost.

3.2.1 Frame

For our project we chose a lightweight and durable plastic material with X-shaped frame as show in **Figure 3.1**.

Figure 3.1: Drone Frame

3.2.2 Motors

According to the previous study, we selected the 980KV EMAX brushless DC motors (**Figure 3.2**) because they offer high efficiency and provide the necessary thrust for the drone to lift and maneuver, compared to the 1400KV motor.

Figure 3.2: Emax Brushless Motor 980KV

The 980KV EMAX motor presents the following characteristics:

- 980 RPM/V (Revolutions per minute per volt), which means that for every 1 volt supplied, the motor spins at 980 RPM.
- A2212 includes two parameters:
  - A22: shows the rotor diameter in millimeters.
  - 12: Represents the stator height.
- Can be used with 2S or 3S (number of cells in series).
- Propeller size: 9x4.7 or 10x4.7 (inches).

3.2.3 Electric Speed Controller (ESC)

An Electronic Speed Controller (ESC) is a device used to control a brushless motor. It receives a PWM signal either from a receiver or any microcontroller. The ESC typically has three wires: red for 5 volts, black for ground, and white for the signal input.

In addition to these wires, there are three more wires (red, blue, and black) that should be connected directly to the motor. The direction of the motor can be reversed by simply switching any two of these motor wires, as shown in **Figure 3.3** below.

Figure 3.3: Electric Speed Controller

3.2.4 Flight Controller

For the flight controller, we chose the Pixhawk 2.4.8, as mentioned in the previous chapter, because it offers advanced features, good stability and flexibility, and supports multiple flight modes. Additionally, the Pixhawk 2.4.8 is open-source, providing a large community and extensive documentation. This makes it easier to troubleshoot and customize. **Figure 3.4** below shows how to connect various components to the Pixhawk.

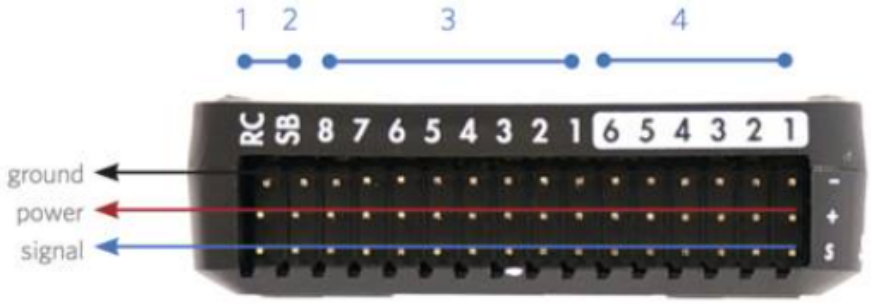


Figure 3.4: Pixhawk Pinout

1. PPM or SBUS signal input.
2. SBUS signal output.
3. PWM signal output.
4. Auxiliary output.

Figure 3.5 shows the Pixhawk wiring with various components, including a Buzzer, Switch, and GPS module with an external compass.

1. **Buzzer**
   The buzzer is connected to the BUZZER port. It provides audible alerts for various system states, such as arming and disarming, battery warnings, and errors.

2. **Switch**
   The safety switch is connected to the SWITCH port. It allows to arm or disarm the vehicle safely. It is crucial for pre-flight checks and ensures the system is ready for operation.

3. **GPS/Compass Module**
   It is highly recommended to use an external Compass/GPS module mounted as far away from the motor and ESC power lines as possible due to potential magnetic interference. The GPS module typically has six wires: four for the GPS module and two for the compass.
   The GPS module connects to the GPS port, while the compass connects to the I2C port.

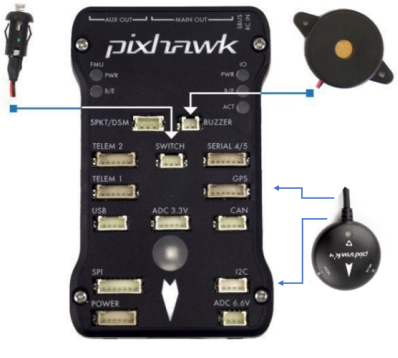


Figure 3.5: Pixhawk wiring diagram

3.2.5 ESP8266

The ESP8266 WiFi module is an inexpensive and programmable device used for WiFi telemetry. We chose to use the ESP8266 instead of the SiK radio telemetry (**Figure 3.7**) due to its lower price and lighter weight. The idea is to replace the SiK radio with the ESP8266.

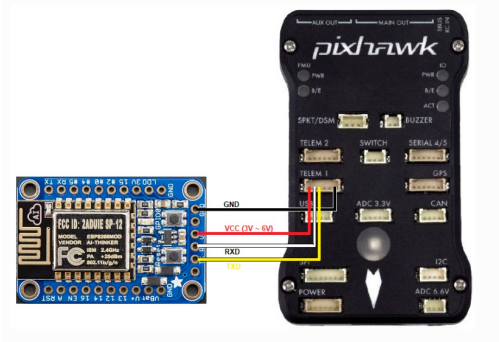


Figure 3.6: ESP8266 Connecting to an autopilot

Figure 3.6 above shows how to connect the ESP8266 to the Pixhawk. The two boards communicate with each other via UART port. The ESP8266 sends the data, received from the autopilot, to the Ground Control Station (GCS) via WiFi using the UDP protocol.

Figure 3.7: Sik Telemetry Radio

3.2.6 Radio Control

For this project, we have chosen the Taranis X9D Plus radio transmitter (**Figure 3.8**). The Taranis X9D Plus is an excellent choice because it supports the S-Bus signal, which is faster. Additionally, Pixhawk can only accept S-BUS or PPM signals.

Figure 3.8: Taranis X9D Plus and its receiver

Figure 3.9 shows the integration of various components with the Pixhawk flight controller, demonstrating the connectivity setup for a comprehensive flight control system. Key components include:

- An ESP8266 WiFi telemetry module enables real-time wireless data transmission for monitoring and control in drone applications.
- Four motors are connected to MAIN OUT, which supplies PWM output signals: two CCW (Counter-Clockwise) motors are connected to pins 1 and 2, and two CW (Clockwise) motors are connected to pins 3 and 4.
- The receiver connects to an ESP32CAM module via UART.
- The ESP32CAM module is then connected to the Pixhawk via UART, interfacing with the Pixhawk's RC IN port.

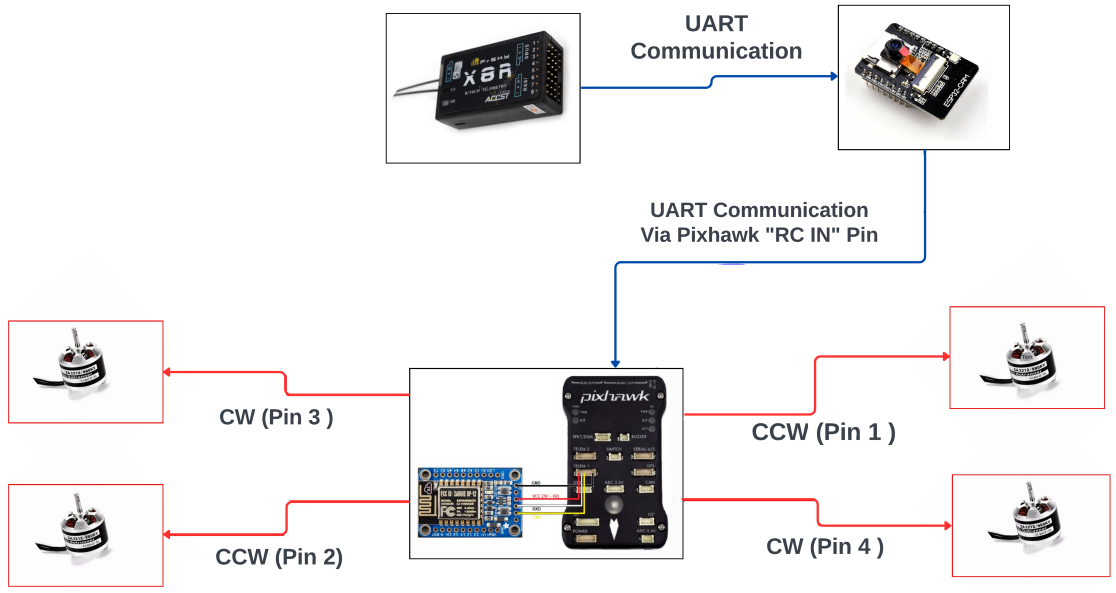


Figure 3.9: Integration of Components with Pixhawk Flight Controller

After understanding the components of a drone, their importance, and the reasons for choosing them, it was time to test and discuss the different operation modes. We began with autotuning to optimize the flight control parameters, where the drone executed predefined maneuvers to adjust its PID settings for improved stability and responsiveness with minimal overshoot. Initially, the drone flew in Alt Hold mode, allowing it to move and rotate freely around the x and y axes. Then we attempted to use AutoTune. However, AutoTune is not always able to determine a good tune for the vehicle due to issues like strong wind, high levels of gyroscope noise, and other factors [21], so we ensured our drone was autotuned in good conditions. Once autotuning was complete, we transitioned to manual mode to assess the drone's handling and performance with the new settings.

3.3 Manual Mode

Manual mode is a control mode where the user has direct control over the drone. In this mode, the user send commands through a remote device, which directly affects the drone's movements. The manual mode allows for precise and responsive control, enabling the user to navigate the drone exactly as desired.

The manual mode has been highly successful. We visualized the changes in PWM signals using telemetry, as shown in **Figure 3.10**. Telemetry data helps in understanding how the PWM signals change in response to the user's commands, providing valuable insights into the drone's behavior and performance.

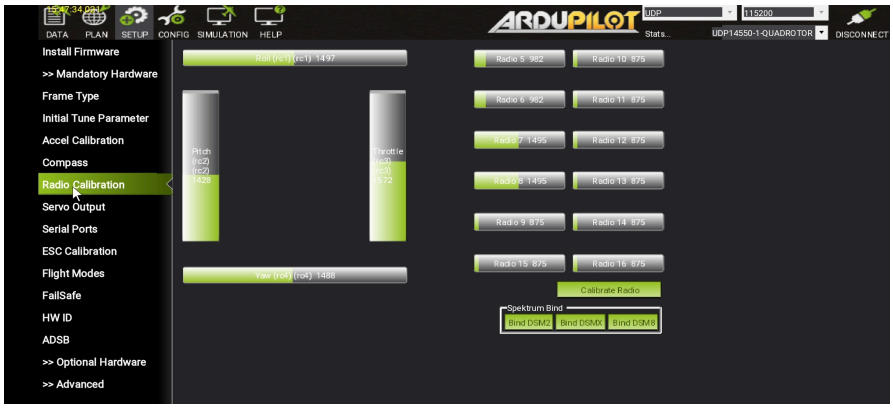


Figure 3.10: Visualization of PWM signals on Mission Planner

The first test was done outdoors to make sure the drone was working properly and could avoid crashes. Testing the drone in an open space allowed us to see how it responded to commands as seen in **Figure 3.11**. This helped to ensure the drone's performance and safety in real-world conditions and gave us an idea of how the drone operates in auto mode.

Figure 3.11: Manual Mode

3.4 Auto mode

In Auto mode, the vehicle follows a pre-set mission plan stored in its autopilot system. This plan includes navigation instructions like waypoints for guiding the vehicle's path. It also includes commands for actions such as taking off, landing, and hovering in place (loiter). These instructions allow the vehicle to perform tasks autonomously based on the programmed sequence, reducing the need for constant manual control.

3.4.1 Waypoints

Waypoints are specific geographic locations defined by coordinates (longitude and latitude) as well as altitude. Additionally, they can include parameters like airspeed that a vehicle follows during a flight.

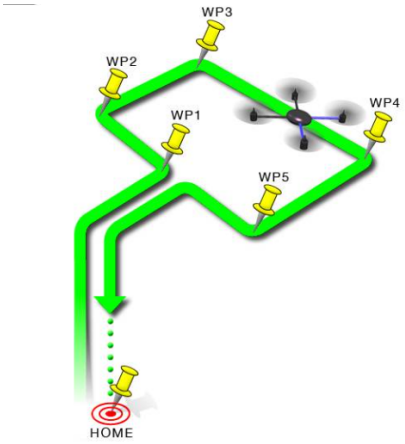


Figure 3.12: Waypoints

3.4.2 Loiter mode

In Loiter mode, the drone maintains its current position along the x and y directions and its altitude along the z-axis. This mode typically requires GPS, but we are particularly interested in developing a loiter mode based on vision. Using ArUco markers offers a reliable alternative in situations where GPS data is unavailable. For this purpose, we chose the ESP32-CAM module (**Figure 3.13**), which combines the ESP32 microcontroller with either an OV2640 or OV7670 camera module. It features a 2-megapixel resolution, onboard Wi-Fi and Bluetooth transceiver. What distinguishes the ESP32-CAM from other modules is its ability to capture frames and transmit them wirelessly via Wi-Fi.

Figure 3.13: ESP32-CAM

It is essential to note that OpenCV requires an operating system (OS). For that, the ArUco markers detection and analysis are performed on a PC platform on which we installed OpenCV library. The last send back just x, y, and z coordinates over WiFi to the ESP32-CAM, which then generate, accordingly, the necessary PWM signals to control the drone. Of course, this require to establish full-duplex communication between the two devices. We opted in addition to implement a WebSocket server, which take care to manage dataexchange, since we have find some trouble to make them communicate directly (this issue will be discussed later).

3.4.2.1 WebSocket

A WebSocket is a technology that enables two-way communication between two different terminals over a long period of time. Unlike traditional HTTP requests, where the client sends a request, waits for a response, and then closes the session, WebSockets allow both the client and server to send messages to each other at any time until one of them decides to close the connection. This makes WebSockets especially useful for real-time applications like chat apps and online games [22]. **Figure 3.14** shows the difference between WebSocket and HTTP connections. WebSocket is a full-duplex communication protocol, while HTTP is a half-duplex communication protocol. In HTTP, the client sends a request, the server responds, and the connection closes.

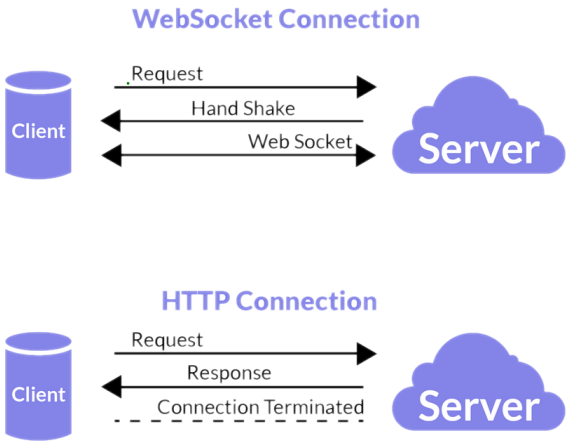


Figure 3.14: Websocket vs HTTP connection

In our project, the ESP32-CAM waits for a client, which is a computer, to connect to the server. Once the connection is established, the ESP32-CAM starts sending captured frames to the computer in binary format. The computer takes care to process frames and extract the x, y coordinates, as well as altitude information relative to the marker. These parameters are sent back to the ESP32- CAM, which then executes additional code to generate PWM signals for each parameter. These PWM values are encoded into binary format to generate the S-BUS signal.

In **Figure 3.15** below, we visualize the S-BUS signal before and after the ESP32-CAM processes it. Initially, the ESP32-CAM receives the signal from the radio control via the receiver.

The ESP32-CAM decodes and re-generates this signal, then transmits the same signal to the Pixhawk. In other words, the ESP32-CAM acts as a bridge, allowing the signal to pass through in manual mode. When switched to auto mode, it starts generating its own signal based on the current location of the drone

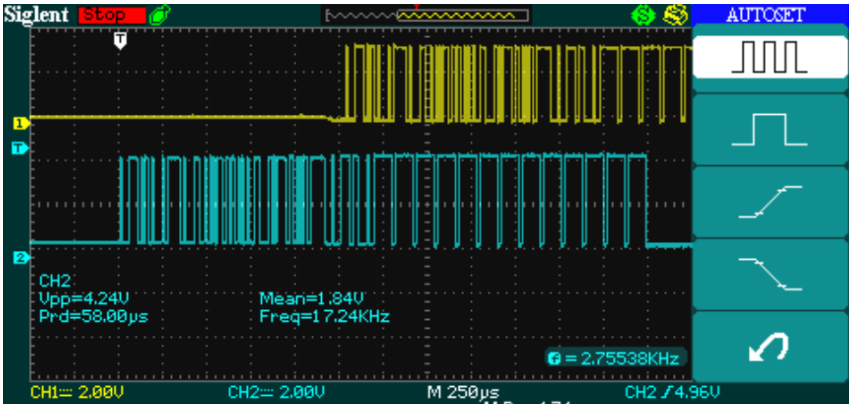
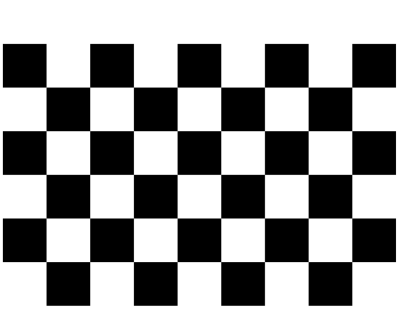


Figure 3.15: S-BUS signal before and after processing by ESP32-CAM

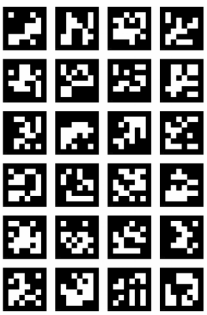
As shown in the **Figure 3.15**, two waveforms are visualized on the oscilloscope. The yellow one represents the signal before processing by the ESP32-CAM (coming from the radio command), and the blue one represents the signal after processing (regenerated signal). It is clear that the processing time is very short (approximately 1.3 milliseconds), and so has no effect on the overall drone's operation.

3.4.2.2 Camera Calibration

Camera calibration is the process of determining the intrinsic and extrinsic parameters of a camera. Calibration ensures accurate measurement by using a chessboard or an ArUco marker grid (**Figure 3.16**).



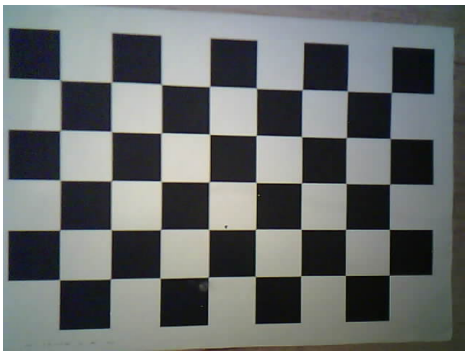
(a) 6x9 Chessboard



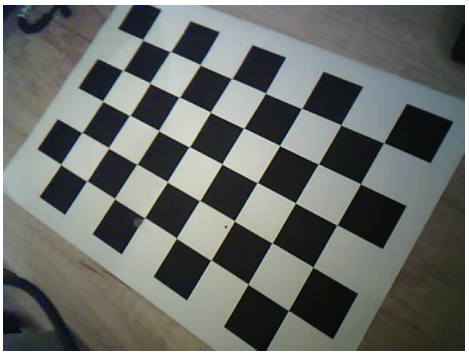
(b) ArUco marker grid

Figure 3.16: Different boards for calibrating the camera

In our case, we used the chessboard by following several steps [23]. We first captured several images of the chessboard from different angles and distances, as shown in **Figure 3.17**.



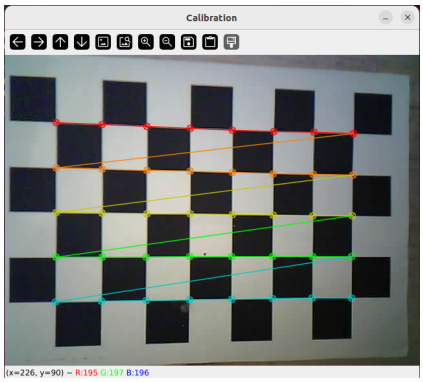
(a) Chessboard image 1



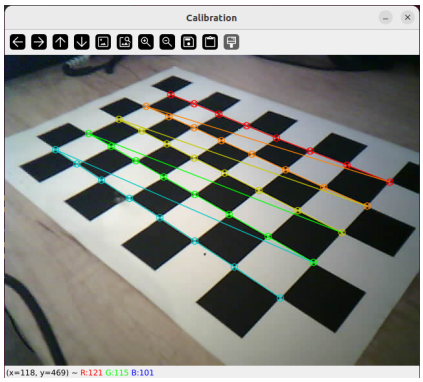
(b) Chessboard image 2

Figure 3.17: Chessboard images captured from different angles and distances

We saved the images for the calibration process, and run another program that detect, in each image, the corners of the chessboard squares, then use them to calculate intrinsic and extrinsic parameters of the camera, as shown in **Figure 3.18**:



(a) Detected corners in image 1



(b) Detected corners in image 2

Figure 3.18: Detected corners in the chessboard images

After calibrating the camera and determining its parameters, particularly the focal length, estimating altitude becomes feasible. This process involves identifying the optical center and measuring its distance from the center of the Aruco marker to determine the x and y coordinates. Perspective projection (**Figure 3.19**) matrices play a crucial role in computer graphics and vision tasks such as 3D rendering and camera calibration. One commonly used form is the OpenGL style matrix, which is essential for transforming 3D points into 2D image coordinates, ensuring accurate representation and measurement in applications like augmented reality and robotics [24].

In OpenGL, the perspective projection matrix is typically defined as:

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{3.1}$$

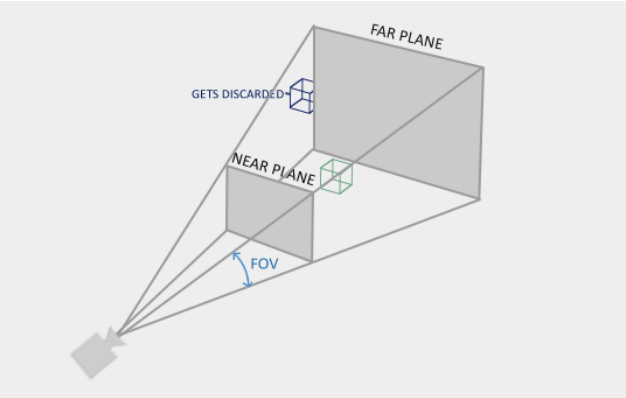


Figure 3.19: perspective projection

where:

- n is the distance to the near clipping plane.
- f is the distance to the far clipping plane.
- r, l are the right and left coordinates at the near clipping plane.
- t, b are the top and bottom coordinates at the near clipping plane.

3.4.2.3 PID tuning

The choice of PID coefficients for the various command variables is challenging, as it requires finding a compromise between the different parameters of all the six degrees of freedom. However, our focus has been limited to the three translational parameters along the x, y, and z axes. We first began by varying the proportional term (K_p), starting with the value 5, where the drone encountered issues and crashed, which led us to implement a semi-automatic mode to avoid such situation. This mode allows the user to take control if significant deviations occur, eliminating the need to manually engage the switch button.

We continued adjusting until we achieved satisfactory performance with a K_p gain equal to 2. However, the drone immediately descended because the proportional term did not affect the system once it reached the desired response. For this reason, we added the integrator term (K_i). The choice of K_i is crucial: a greater K_i results in a shorter response time, but can cause the drone to overshoot the target, while a smaller K_i results in a longer response time. In this phase, we paid attention to select an appropriate Ki value to balance response time and target accuracy. Finally, we found that a K_i value of 0.35 provided the best results. The **Figure 3.20** shows the drone hovering at the desired altitude and position.

(a) (b)

Figure 3.20: Loiter Mode

Figure 3.21 shows how the ESP32-CAM communicates with the Pixhawk flight controller. The ESP32-CAM first checks if it is connected to WiFi before both manual and automatic modes. If not connected, it is not ready to arm. In manual mode, the ESP32-CAM simply passes the received S-BUS signal to the Pixhawk. In automatic mode, if the PC (client) connects to the server (ESP32-CAM), the ESP32-CAM sends images to the PC. The PC processes these images and sends back coordinates (x, y, and z) to the ESP32-CAM. The ESP32-CAM uses these coordinates to generate PWM signals using a PID controller, which are then converted into S-BUS signals for the Pixhawk. The Pixhawk uses these signals to control the motor speeds. This workflow is important because it ensures the drone can operate both manually and automatically, with steps to handle errors and ensure reliable control.

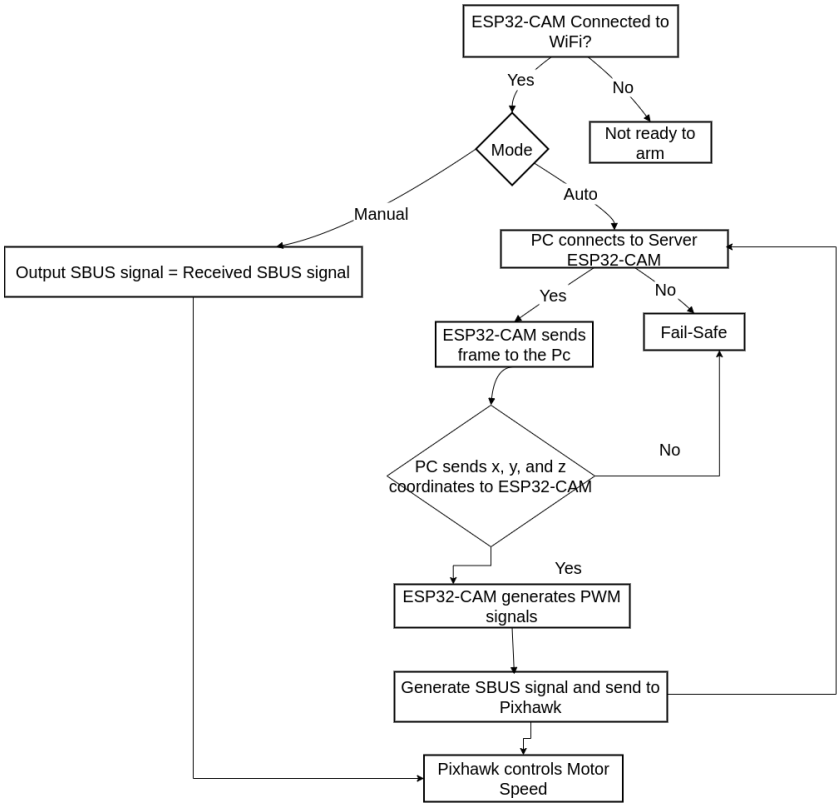


Figure 3.21: Workflow for ESP32-CAM Communication with Pixhawk

3.5 Challenges and solutions

In this section, we would like to discuss the differents problems we faced while developing our project.

Starting with transferring telemetry data using the radio transceiver NRF24L01. The last can't communicate directly with the auto-pilot board, but rather require an intermediary device such as microcontroller. This setup made it difficult to stream real-time data directly into the Mission Planner software. This constraint is overcome after switching to ESP8266 module, which is perfectly optimized to WiFi communication.

Furthermore, several challenges were encountered with the ESP32-CAM board. First, this one can handle only one client at a time, although it supports Websocket communications. In other words, when a second client tries to get access to the server, the ESP32-CAM stops sending frames to the first one, which is not convenient for our project. Additionally, the ESP32- CAM could not be configured as an access point, so we had to connect it to an existing network.

While we were working on S-BUS signal on the ESP32-CAM, everything was functioning perfectly until we started using Camera module. We discovered that the ESP32-CAM's UART0 is occupied by the camera module and is primarily used for debugging purposes (**Figure 3.22**). However, our project requires receiving and transmitting S-BUS signals via UART pins.

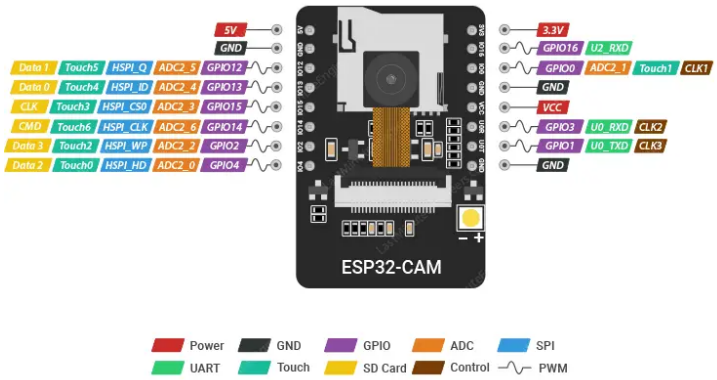


Figure 3.22: ESP32-CAM Pinout

We attempted to use SoftSerial to transmit the S-BUS signal, but it did not work, and we encountered the error shown in **Figure 3.23**. We also tried receiving the signal via RX2 and transmitting it through any GPIO pins, but we could not achieve the same baud rate. It is challenging to make a software-based solution work as effectively as a hardware-based one, which presented a significant obstacle during development. On a positive side, the ESP32-CAM does provide flexibility to allocate a serial interface to any free pins.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
E (474) gpio: gpio_install_isr_service(449): GPIO isr service already installed
```

Figure 3.23: Interrupt Service routine issue

The third encountered issue related the ESP32-CAM is that it began rebooting and displaying an error about pushing the content of the frame buffer to the PSRAM (Pseudo Static RAM) as shows the **Figure 3.24**, which is typically used for higher quality frame sizes. However, our setup does not utilize PSRAM. We thought about a configuration variable which saves, by default, frame buffers content to the PSRAM, then the problem was solved by reconfiguring the camera to save frame buffers' content to the internal DRAM (Dynamic RAM) instead.

```
Guru Meditation Error: Core  0 panic'ed (StoreProhibited). Exception was unhandled.

Core  0 register dump:
PC      : 0x4008a628  PS      : 0x00060f30  A0      : 0x800869c5  A1      : 0x3ffdfb00
A2      : 0x00000400  A3      : 0x3ffd73e8  A4      : 0x3ffd83e8  A5      : 0x00060123
A6      : 0x00060120  A7      : 0x00000001  A8      : 0xffffffff  A9      : 0x000000ff
A10     : 0x00000003  A11     : 0x00060123  A12     : 0x00060120  A13     : 0x3ffdfb68
A14     : 0x00000000  A15     : 0x00000000  SAR     : 0x0000000c  EXCCAUSE: 0x0000001d
EXCVADDR: 0xffffffff  LBEG    : 0x4008b3d4  LEND    : 0x4008b3f0  LCOUNT  : 0xffffffff


Backtrace: 0x4008a625:0x3ffdfb00 0x400869c2:0x3ffdfb20 0x40104c05:0x3ffdfb40



ELF file SHA256: 18644b5c00688df9

Rebooting...
ets Jul 29 2019 12:21:46

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
Connecting.......
```

Figure 3.24: ESP32-CAM is rebooting

After mounting the ESP32-CAM on our drone, we first attempted to send commands to the flight controller and see the reactions in Mission Planner software. We noticed that the response time was significantly important (about six seconds), so we had to drop down the frame rate dropped 3 FPS instead of 12, which is the value that we used previously. A good solution was to use an OS (operating system) in order to run the various tasks (including frames acquisition and transmission) in multitasking fashion. Fortunately, the ESP32 family supports the so called FreeRTOS, which is a free and open source real time OS. By using it, one more benefit is the ability to meet time constraints associated to each task, thing which is, actually, highly demanded in drones development. However, when we mounted it again on our drone and placed it on the floor, the frame rate remained at 2 or 3 FPS, but the response to radio control signals was faster.

During testing, we also measured the voltage output of the ESP32-CAM, which was stable at 4.2 volts. So we though powering it from an ESC since it was not in use.

The final issue with the ESP32-CAM is that we could not transmit more than 8 frames per second, even on a short distance. This is likely due to a weak connection between the PC and the ESP32-CAM. To improve signal transmission, we considered add a dedicated antenna to the ESP32-CAM, which finally allowed to resolve this issue.

3.6 Conclusion

In this chapter, we detailed the step-by-step process of assembling and implantation of our drone, all being based on what we have acquired from the previous chapters. We started by assembling the frame and installing the propellers along with their electronic speed controllers, and finally the flight controller. Each component was carefully selected and studied to ensure their compatibility and required performance. Next, we integrated the ESP8266 module to enable wireless communication of telemetry data, followed by configuring the radio command system for manual mode. We also explained the configuration process the drone's auto mode, which we used in case of loiter mode for stable hovering.

Additionally, we highlighted the importance of camera calibration, in ensuring an accurate data for the automatic piloting system. Throughout the test phase, we conducted multiple flight tests in various modes including manual and the different auto modes, such as loiter, altHold, and auto landing. Due to time constraints, our primary focus was on fine-tuning the performance of the vision-based loiter mode.

General Conclusion

In this report we presented the design and implementation of a multi-rotor drone equipped with a vision-based automatic piloting mode. The multimode operation has been taken into account in the context of this project, all taking the advantages of the Pixhawk autopilot board. By using vision based approach for control and navigation, as an alternative to the GPS based system, our drone is safe and can operate indoors or outdoors.

In the theoretical part, we explored the fundamentals of quadcopter drones flight, including the kinematics and dynamics models. We also discussed the control technics, with focusing on PID controllers. We presented the communication protocols which are essential for data and commands transmission. Finally, we devoted a good part to discuss the computer vision related aspects.

The second chapter introduce the different components of our drone with more focus on electrical motors and their characterization. Detailed calculations were presented to match the motor specifications with the design requirements, ensuring that the motors provide sufficient lift and control.

The practical implementation involved assembling the hardware components, such as the frame, motors, electronic speed controllers (ESCs), flight controller, and camera. We successfully integrated these components and configured the necessary software to enable both manual and automatic control modes. The vision-based piloting system was calibrated and tested, demonstrating the feasibility of autonomous navigation by relying on visual data.

Throughout the project, we encountered and resolved serious challenges, including hardware compatibility issues of the ESP32-CAM and calibration complexities. These experiences provided valuable insights and contributed to the overall success of the project.

Finally, the developed quadcopter showed the effectiveness of combining embedded systems with computer vision to achieve autonomous flight. At the end of this experience, we look forward for more improvements in terms of our drone's performance and functionalities, but also for future research achievements in the area of applied computer vision.

As a perspective, we consider::

- Exploiting a high-definition satellite image stored in the drone's memory as a reference for autonomous navigation.
- Optimizing physical features of our drone by using quality materials such as carbon fiber.
- The integration of all electronic systems onto a single board, sufficiently powerful and versatile.

Bibliography

[1] R. Niemiec and F. Gandhi, "A comparison between quadrotor flight configurations," 2016.

[2] A. Sigalos, M. Papoutsidakis, A. Chatzopoulos, and D. Piromalis, "Design of a flight controller and peripherals for a quadcopter," *International Journel of Engineering Applied Sciences and Technology*, vol. 4, no. 5, pp. 463–470, 2019.

[3] I.-R. Morar and I. Nascu, "Model simplification of an unmanned aerial vehicle," in *Proceedings of 2012 IEEE International Conference on Automation, Quality and Testing, Robotics*, pp. 591–596, 2012.

[4] Eduardo M. Bucio-Gallardo, Ricardo Zavala-Yoé, and Ricardo A. Ramírez-Mendoza, "Mathematical Model and Intelligent Control of a Quadcopter, with Non-conventional Membership Functions," *Journal of Energy and Power Engineering*, vol. 10, Oct. 2016.

[5] R. Beard, "Uav coordinate frames and rigid body dynamics," 2004.

[6] A. Sanca, P. Alsina, and J. Cerqueira, "Dynamic Modelling of a Quadrotor Aerial Vehicle with Nonlinear Inputs," *Latin American Robotics Symposium and Intelligent Robotics Meeting*, vol. 0, pp. 143–148, Oct. 2008.

[7] B. Ebrahimi, "Modelling and control of quadcopter School of Science Mat-2.4108 Independent research project in applied mathematics A,"

[8] Y. Li, D. Li, W. Cui, and R. Zhang, "Research based on osi model," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, pp. 554–557, 2011.

[9] S. Narayan, "Improving network performance: an evaluation of tcp/udp on networks," 2014.

[10] A. Das, "Tcp/ip thesis." `https://www.academia.edu/37672498/TCP_IP_thesis`. ND.

[11] N. S.Nise., "Control systems engineering.," *6th ed. 111 River Street, Hoboken:J.Wiley Sons, Inc*, 1994.

[12] R. V. A. Jayachitra., "Genetic algorithm based pid controller tuning approach for continuous stirred tank reactor," *In: Advances in Artificial Intelligence*, 2014.

[13] A. Hughes and B. Drury, "Electric motors and drives: Fundamentals, types and applications," vol. 486, pp. 1–13, 2006.

[14] P. Yedamale, "Brushless dc (bldc) motor fundamentals," *Microchip Technology Inc*, vol. 20, 2003.

[15] Robu.in, "Emax xa2212-980kv outrunner brushless dc motor." https://robu.in/wp-content/uploads/2017/11/EMAX-XA2212-980KV-Outrunner-Brushless-DC-Motor-ROBU.IN_.pdf, 2017.

[16] ArduPilot, "Pixhawk overview." https://ardupilot.org/copter/docs/common-pixhawk-overview.html.

[17] TheDIYGuy999, "S-bus communication protocol." https://https://github.com/TheDIYGuy999/SBUS.

[18] E. Gottleben, "Vision based control of an autonomous uav," *Master of Science Thesis in Reglerteknik*.

[19] Sergio Garrido, Alexander Panov, "Detection of aruco markers." https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html.

[20] OpnCV, "Opencv website." https://opencv.org/.

[21] ArduPilot, "Autotune." https://ardupilot.org/copter/docs/autotune.html, 2023.

[22] J.-P. Erkkilä, "Websocket security analysis," *Aalto University School of Science*, 2012.

[23] "Camera calibration." https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.

[24] S. H. Ahn, "Opengl projection matrix." https://www.songho.ca/opengl/gl_projectionmatrix.html.

[25] arupilot, "Ardupilot documentation." https://ardupilot.org/plane/docs/common-pixhawk-overview.html.

[26] px4, "Autopilot user guide." https://docs.px4.io/v1.12/en/.

[27] px4 community, "Autopilot community." https://discuss.px4.io/.

[28] ardupilot community, "Ardupilot community." https://discuss.ardupilot.org/.

[29] NBC News, "Iran hijacked us drone, claims iranian engineer." https://www.nbcnews.com/id/wbna45685870, 2011.