الجمهـــــــــــورية الجزائـــــــرية الديمقـــــراطية الشـــــعبية
Democratic and Popular Republic of Algeria
وزارة التعليـــــــم العـــــــالي والبحـــــــث العلمـــــي
Ministry of Higher Education and Scientific Research

Larbi Ben M'hidi University O.E.B
Institute of Applied Sciences and Technologies.
Department of Networks and Telecommunications

جامعة العربي بن مهيدي أم البواقي
معهد العلـوم و التقنيات التطبيقيـة
قسـم شبكات و اتصالات سلكية ولا سلكية

Thesis

Presented for the purpose of obtaining the Professional Master Degree's in Systems Networks and Telecommunications.

Designing and Implementation of a multi-purpose UAV

Submitted by**: DERGHAL Safa**
DJABALLAH Badr

Under the Supervision of: Dr. **MOULAHCENE Fateh**

2023/2024

Dedication:

I would like to express my deepest gratitude to my family for their love, support, and sacrifices throughout my academic journey. Their constant encouragement and belief in me gave me the strength to persevere and complete this thesis.

I dedicate this work to my mother and father

To my brothers and sisters

To my friends

To my thesis Co-worker

Thank you,

Derghal Safa,

Praise be to God who guided us and helped us along the way to complete this work.
Praise be to God, praise worthy of His majesty and greatness.

First of all, I want to dedicate this work for my family, who have been my greatest support and who tolerated my absence for five years. Thank you very much.

Secondly, i want to thank our supervisor, Dr. Molahcene Fateh, for guiding us and providing everything we need to complete this project.

I also want to thank my mate who endured the trouble and hardship of this project with me. It was an honor for me to work with you. I wish you all the best in your life.

Also I want to thank my friends and specially Zain Garadi for his guidance and assistance and everyone I knew during this period.

Finally, and as Snoop Dogg said: «Last but not least, I wanna thank me
I wanna thank me for believing in me
I wanna thank me for doing all this hard work
I wanna thank me for having no days off
I wanna thank me for... for never quitting
I wanna thank me for always being a giver
And tryna give more than I receive
I wanna thank me for tryna do more right than wrong
I wanna thank me for just being me at all times»

A journey has ended, and another has just began

Djaballah Badr,

Acknowledgements:

First, we would like to express our sincere gratitude to all those who have supported and guided us throughout the course of this thesis.

Also we would like to thank our supervisor Dr Moulahcene Fateh whose expertise, understanding and patience added considerably to our graduate experience.

Lastly, we would like to extend our sincere appreciation to our colleagues and friends for their encouragement, valuable discussions, and moral support throughout this journey. Their camaraderie has made this academic pursuit an enriching and fulfilling experience.

Thank you all.

Abstract

This thesis presents the design and implementation of an Unmanned Aerial Vehicle (UAV) quadcopter, developed to perform multiple tasks including delivery, fires detection and firefighting, as well as plant disease detection and treatment. It introduces a full approach to the design and implementation of control system for six degrees of freedom, four rotors unmanned aerial vehicle known as quadcopter. Initially, different forces acting on the system and aerodynamic effects are analyzed. Then, the mathematical model for simulation and control of such systems is detailed as well as the design methodology for the Quadcopter. Based on the mathematical model, linear and nonlinear control techniques are used to design and simulate various controllers along this work. Next, the model is developed and tested in Matlab\Simulink environment. Lastly, it discusses the step-by-step construction of the Arduino-based quadcopter, and the upgrade to an Ardupilot system. The UAV's practical applications are demonstrated through deployment scenarios in delivery, fire management, and agriculture. Object detection for forest fire prevention and image classification algorithms for plant disease identification are implemented and evaluated. This research contributes to the field by providing a robust framework for the design and application of versatile UAVs, addressing both theoretical and practical challenges, and proposing enhancements for future development.

ملخص

تعرض هذه الأطروحة تصميم وتنفيذ مركبة جوية بدون طيار (UAV) كوادكوبتر، تم تطويرها لأداء مهام متعددة بما في ذلك التسليم والكشف عن الحرائق ومكافحة الحرائق، وكذلك الكشف عن أمراض النبات وعلاجها. إنه يقدم منهجًا كاملاً لتصميم وتنفيذ نظام التحكم لستة درجات من الحرية، وأربعة دوارات للمركبات الجوية بدون طيار المعروفة باسم كوادكوبتر. في البداية، يتم تحليل القوى المختلفة المؤثرة على النظام والتأثيرات الديناميكية الهوائية. ثم يتم تفصيل النموذج الرياضي للمحاكاة والتحكم في مثل هذه الأنظمة بالإضافة إلى منهجية تصميم الطائرة الرباعية. استنادا إلى النموذج الرياضي، تم استخدام تقنيات التحكم الخطية وغير الخطية لتصميم ومحاكاة وحدات التحكم المختلفة على طول هذا العمل. بعد ذلك، تم تطوير النموذج واختباره في بيئة Matlab\Simulink. وأخيرًا، يناقش البناء خطوة بخطوة للطائرة الرباعية المبنية على Arduino، والترقية إلى نظام Ardupilot. يتم عرض التطبيقات العملية للطائرات بدون طيار من خلال سيناريوهات النشر في التسليم وإدارة الحرائق والزراعة. يتم تنفيذ وتقييم اكتشاف الكائنات للوقاية من حرائق الغابات وخوارزميات تصنيف الصور لتحديد أمراض النبات. يساهم هذا البحث في هذا المجال من خلال توفير إطار قوي لتصميم وتطبيق الطائرات بدون طيار متعددة الاستخدامات، ومعالجة التحديات النظرية والعملية، واقتراح تحسينات للتطوير المستقبلي.

Résumé

Ce mémoire présente la conception et la mise en œuvre d'un quadricoptère de véhicule aérien sans pilote (UAV), développé pour effectuer de multiples tâches, notamment la livraison, la détection et la lutte contre les incendies, ainsi que la détection et le traitement des maladies des plantes. Il présente une approche complète de la conception et de la mise en œuvre d'un système de contrôle pour un véhicule aérien sans pilote à six degrés de liberté et à quatre rotors connu sous le nom de quadricoptère. Dans un premier temps, différentes forces agissant sur le système et les effets aérodynamiques sont analysés. Ensuite, le modèle mathématique de simulation et de contrôle de tels systèmes est détaillé ainsi que la méthodologie de conception du Quadcopter. Sur la base du modèle mathématique, des techniques de contrôle linéaire et non linéaire sont utilisées pour concevoir et simuler divers contrôleurs tout au long de ce travail. Ensuite, le modèle est développé et testé dans l'environnement Matlab\Simulink. Enfin, il aborde la construction étape par étape du quadricoptère basé sur Arduino et la mise à niveau vers un système Ardupilot. Les applications pratiques du drone sont démontrées à travers des scénarios de déploiement dans les domaines de la livraison, de la gestion des incendies et de l'agriculture. Des algorithmes de détection d'objets pour la prévention des incendies de forêt et de classification d'images pour l'identification des maladies des plantes sont mis en œuvre et évalués. Cette recherche contribue au domaine en fournissant un cadre robuste pour la conception et l'application de drones polyvalents, en relevant à la fois les défis théoriques et pratiques et en proposant des améliorations pour le développement futur.

Table of content

List of figures:

General Introduction

General introduction:

Drones, more technically known as Unmanned Aerial Vehicles are highly versatile platforms used for an array of applications from aerial photography to surveillance, agriculture, search, and rescue. A thorough engineering of drones, from specification through modeling and control to implementation and testing, requires a sturdy theoretical background in a multitude of areas, such as aerodynamics, flight dynamics, control, mathematical modeling.

The first chapter covers the theoretical underpinnings of UAV operation, providing a full account of the fundamental concepts and principles required for the engineering of UAV design, analysis, and inspection. We derived the kinematics and dynamics models of the quadcopter based on Newton-Euler or Euler-Lagrange formalisms, these formalisms provide assumption about structure and operation of the quadcopter.

In the second chapter, we give an outline of software and hardware used in our work, describing the component used in construction of the quadcopter. We discuss the engineering and architectural decisions we made when constructing the quadcopter to provide the best performance. Further, we identify the main uses and proposed deployment of the quadcopter to encompass delivery services, fire fighting and detection, plant disease identification and eradication.

For the third chapter, our attention has been drawn to the potential of object detection in forest fire prevention through flame detection, as well as the efficacy of live plant disease classification utilizing image classification algorithms.

While the last chapter discusses the installation steps building the arduino based quadcopter, the limitation faced then the upgrade to the ardupilot, and it also discusses the system simulation using simulink MATLAB.

Chapter I:

Theory, fundamentals and modeling of UAV

I.1 Introduction:

This first chapter explores the fundamental concepts and principles essential for UAV engineering, focusing on quadcopters, the kinematics and dynamics models of quadcopters using the Newton-Euler and Euler-Lagrange formalisms. These frameworks accurately describe the motion and forces acting on a quadcopter, based on assumptions about its structure and operation. This theoretical foundation equips readers with the necessary insights for advanced UAV design and analysis.

Modeling the quadcopter requires paying attention to the design configuration. Researchers have already studied the dynamics of both +- or X-configuration. The +-configuration is simpler and easier to explain the quadcopter flight mechanisms according to Alexander Lebedev (1). However, real applications favor the use of X-configuration due to intensive stability (2) (Our quad is designed according to X-Configuration).

I.2 UAVs classification:

The growing interest in UAVs in recent years has led to the strong emergence of various types of aircraft with varying configurations and components in terms of shape and size. UAVs are divided into four types: single-rotor, multi-rotor, fixed-wing, and hybrid (3) , as shown in figure 1.

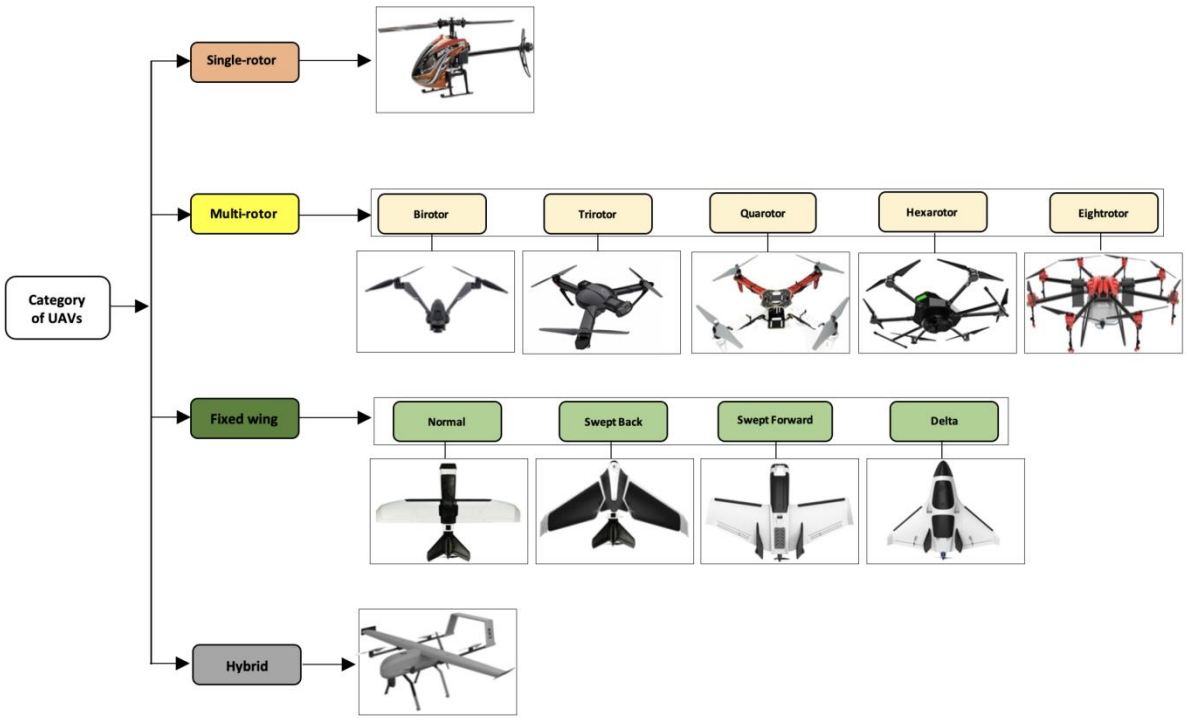


Figure 1 UAVs Categories

Single rotor(4) (or helicopter): This type of UAVs – namely the VTOLs – allows you to take off and land vertically. These helicopters are designed with a main rotor for pitch, roll, and yaw control and a tail rotor to manage the direction. Its outstanding characteristic is its capacity to bear heavy loads for an extended period of the time. On the one hand; the complexity of their mechanical system, as well as the massive size and the high cost of rotors constitute a danger to the unmanned versions on the other hand (5).

Multi-rotor (or multicopter): thus is it a VTOL having more than two wings. This UAV category is subdivided into 5 sub-categories, which are birotor, trirotor, quadrotor, hexarotor, and eight-rotor (octocopter) (6) (7). Multi-rotor drones are as well as single-rotor drones enable vertical takeoff and landing. They in general have great speed and ability to proclam their aerial maneuvers in the sky or even in small mandatory space. But still, the problem of short estimated time of flight is the main shortcoming of these kinds of aircraft.

Fixed-wing: the basic principle of aerodynamics for this class of UAVs includes an extremely simple structure consisting of a single rigid wing. These drones are not assigned a class only due to the kind of wings, but also in accordance with things like body and power system (Li-ion, Li-Po batteries, and gas-powered). They are subdivided into four subcategories: normal, swept back, swept forward, and delta. In addition, they have the capability to load heavier plots than the multi-rotors (8). The problem with these UAVs is that they do not have the agility to carry out complex maneuvers and also, they require a runway for landing and takeoff (9).

Hybrid: this last category is still under development. It is an enhanced version that uses the advantages of multi-rotor and fixed wing UAVs. In this regard, they are good for quick movements and are used for long-distance flights. They feature large capacity and do not need a runway. The main disadvantages are the high price, the complicated mechanics, and the lower performance in terms of flight stability and the restrictive speed ranges (10).

I.3 Quad-rotors:

Quadrotors, or quadcopters, are rotorcraft with four vertical rotors arranged in a square or x configuration. All rotors contribute lift and thrust, giving quadrotor the ability to change its spatial orientation by controlling the rotor’s speed individually. They offer exciting flight features through their acrobatic abilities, high level of adaptability, and the

breadth of their potential usage areas. Propellers 1 and 3 (Figure 2) rotate in the same direction while propellers 2 and 4 rotate in an opposite direction. This rotation style leads to balancing the total system torque and cancelling the aerodynamics torques in stationary flights (11).

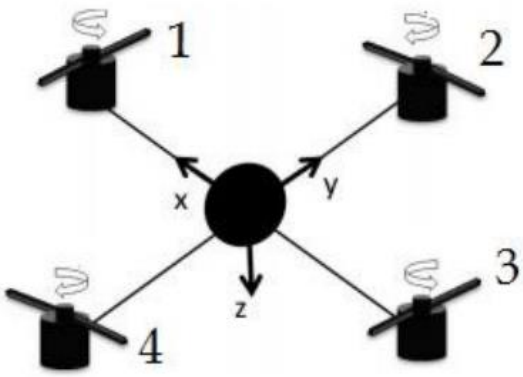


Figure 2: Structure of quadcopter

I.3.1 Quad-rotor configurations:

The (+) configuration:

In a + configuration, the four rotors are located at the corners of the square or rectangular pattern rotately, with one rotor at each corner of the square. This + structure is easy to build and control because the rotors are aligned along the cardinal axes: front, back, left to right, and stable in both pitch and roll motions what makes it easier to control, which is used by beginners and in applications where stability is critical.

The centralized layout of the rotors enables more balanced distribution of the payload weight that is good for the carrying of the external equipment or sensors.

The x Configuration:

In an X configuration, the four rotors are positioned in the form of an X, with two rotors at the front and two at the back. It presents the advantage of greater agility and maneuverability than the + configuration. The X configuration is characterized by a differential torque generated by offset placement of front and rear rotors, which improves yaw control through rotating the UAV around its vertical axis. it commonly leads to more compact and aerodynamic designs that are beneficial in many applications, such as indoor navigation or search and rescue operations where size and flexibility are the key components.

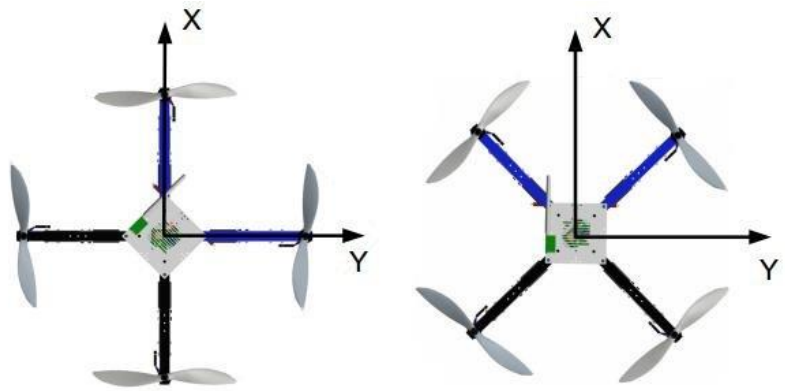


Figure 3: + and x Quadrotor's configurations

For both configurations, vertical upward motion is generated by increasing the rotors speed whereas vertical downward motion is generated by decreasing rotors speed. For +-configuration, pitch rotation is realized by varying the speed of the front and back rotors while roll rotation is realized by varying the speed of left and right rotors. For X-configurations, the speed of four rotors is varied according to the desired rotation.

I.4 UAVs Dynamic Fundamental Principles:

I.4.1 Newton's third law:

When a quadcopter's propellers spin, they push air downward. Using Newton's third law, this represents the action. For Newton's law to be true, there must be an equal and opposite reaction. This reaction is an upward force pushing on the quadcopter. Once this force exceeds the force of gravity pulling the quadcopter downward, the quadcopter begins to move up. (12)

Figure 4 Upward motion of a quadcopter (Newton's third law)

I.4.2 Forces acting on a Quadcopter:

There are three forces acting on a quadcopter:

F_W is the total weight of the drone with a payload, which pulls down the drone due to the force of gravity, whereas **F_{DH}** and **F_{DV}** are drag forces in horizontal and vertical direction, respectively, that are caused by the disruption of airflow. Drag opposes a movement of the drone in horizontal and vertical directions. **F_T** is the thrust produced by the rotating propellers of the drone; it opposes the weight and drag to sustain the height and speed of the drone. **Figure 5(a)** and **Figure 5(b)** show the overall forces when a drone

moves vertically at a constant speed $\mathbf{v_v}$, and flies horizontally at a constant speed $\mathbf{v_h}$, respectively. The sum of the weight and drag equals to the thrust in both cases. (13)

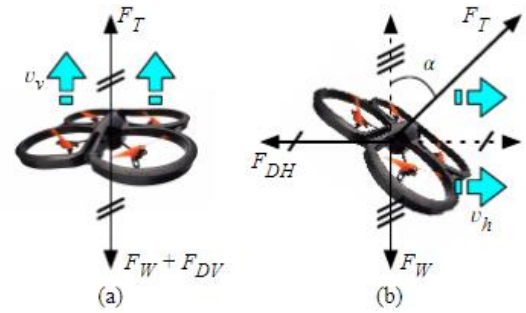


Figure 5: Forces acting on a quadcopter

I.4.3 Principal Axes:

The main axes for a quadcopter are three orthogonal axes that simulate the rotational direction and flight maneuver of the UAV.

Roll Axis: This axis is oriented along the flight axis of the quadcopter itself from one side to the other. Rotating about the roll axis causes a side tilt that gives a rolling motion.

Pitch Axis: This axis is located from the front to back of the quadcopter. The rotation around the axis of the pitch makes the quadcopter to move forwards or backwards, thus becoming a pitching movement.

Yaw Axis: Here, the y-axis is the vertical cross section through the quadcopter's center and is perpendicular to the roll and pitch axes. The turning motion on the yaw axis creates the yawing effect, which makes the quadcopter tilt right or left.

These axes play a key role in streamlining the direction and behavior of the quadcopter in flight. Through controlling the speeds of spinning propellers that spin in different angles, this quadcopter can obtain the various types of rotations around the axes which make the machine capable of agile and coordinated flights.

Figure 6: Quadcopter axes

I.4.4 Movement of a quadcopter:

Principal motions of a quad are (Climbing, right and left translations, forward and backward translations, right and left rotations).

Climbing:

In order to go up vertically, the quadcopter significantly increases the total thrust of all four rotors concurrently, thus inducing a vertical ascent.

Right and Left Translations:

In order to fly right, the quadcopter implements higher speed rotation on its left hand rotor and slightly lower speed on its right hand rotor. So, the drone begins to tilt towards the right and travels in this direction. On the other hand, to move to the left the speed of the rotors on the right side increases while the speed of the rotors on the left side slightly decreases.

Forward and Backward Translations:

In order to go ahead, the quadcopter raises the speed of the rear rotors while at the same time lowers the speed of the front rotors. This pushes the drone forward as it tilts forward. To go in reverse direction, the quadcopter accelerates the speed of its front rotors more than its rear rotors.

Right and Left Rotations:

In order to turn to the right, the quadcopter raises the speed of the left rotors spinning clockwise (the ones shown in figure 7) slightly slower as compared with the speed by which it speeds up the right rotors that spin counterclockwise. This makes the rocket to instead rotate to the right with a tendency towards a yaw motion. To rotate to the left, the quadcopter increases the rate at which its rotors rotating clockwise (viewed from above) while slightly decreasing the speed of its counterclockwise rotors.

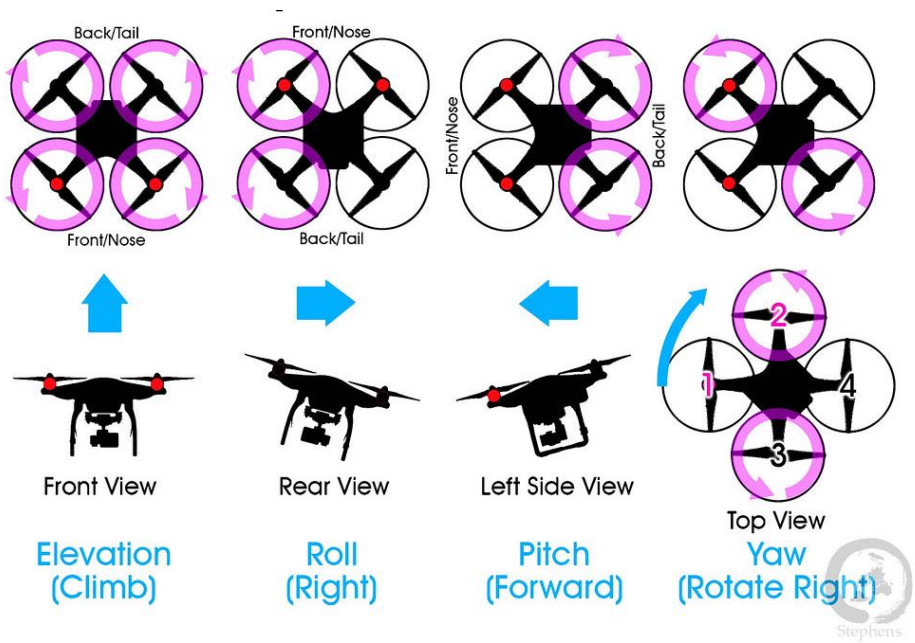


Figure 7: Quadcopter axes and motions

I.5 Kinematics and Dynamics:

In order to discus modeling of the quadrotor, we need first to define the reference frames will be used.

I.5.1 Reference frames:

The quadrotor is equipped with numerous sensors and variety of mission requirements. Each mission and a sensor result is derived from different coordinate frames. So, a single frame of reference is not enough to represent the motion of the quadrotor.

For example, an accelerometer and gyroscope provide output with respect to body frame, whereas a magnetometer gives output with respect to the Earth frame. Moreover, the output of GPS depends on the geodetic frame.

All equations must be expressed in the same frame in the mathematical model, so transformations must be performed between the coordinate frames (14).

- **Body Frame (Mobile frame):**

The origin of the body fixed frame is at the center of mass of the quadrotor. It is previously mentioned that the quadrotor is designed according to X configuration. Therefore, the X axis of body frame extends to forward direction of quadrotor, the Z axis points downward, which is oriented to gravity. Y axis is formed within a right handed orthogonal set, right side of the system.

- **Inertial frame or Earth frame (NED):**

The Earth frame is the North-East-Down coordinate system, which is the most often applied in aviation. The choice of the reference frame can be made arbitrary, it can be centered on any point on the earth, the landing site of the vehicle is most generally used as the origin point. The Z axis is oriented in the direction of the gravitational force. The X axis is directed to north, Y axis goes out to east (14). Because of low speed of the system, the Earth is supposed to be fixed (it doesn't rotate) and flat.

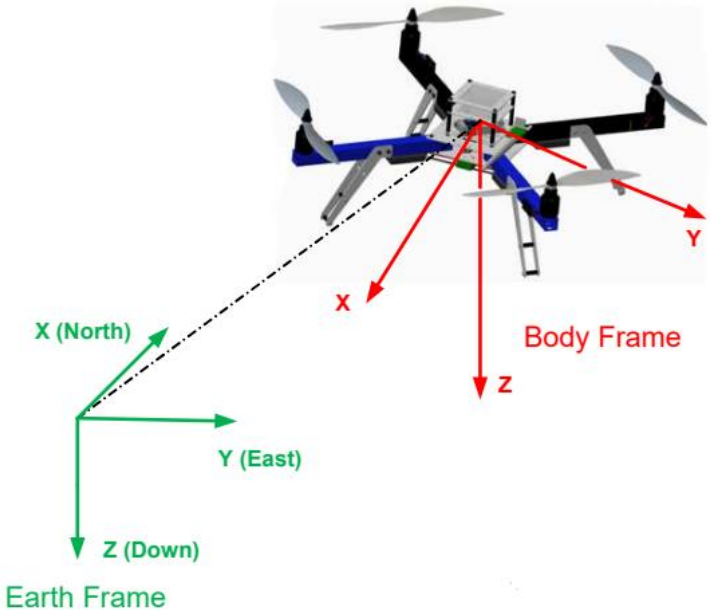


Figure 8: Earth and Body frame of quadcopter

- **Geodetic and ECEF Coordinate System:**

The Geodetic coordinate system is used on GPS-based navigation system. More specifically, this is not an ordinary Cartesian system that has no orthogonal components; instead, longitude, latitude and height (altitude) are used as coordinates.

In the Earth Centered Earth Fixed (ECEF) coordinate system, the system's origin is at the center of the earth, the Z axis points to the North Pole, the X axis intersects at the equator and 0° longitude. The frame rotates together with the earth.

ECEF coordinates are used in the conversion of geodetic coordinates to an Earth-frame. The outputs of GPS are in the geodetic coordinate system, so they are transformed into ECEF frame as provided with the world shape model based on WGS 84 (World Geodetic System 84). Afer the ECEF is computed, the result is converted to an earth-frame

coordinates. It is an essential feature of the autopilot system's tracking algorithm.

I.5.2 Kinematic modeling:

The kinematics concern is the study of motion of the objects without considering the forces and moments acting upon these objects. Transformation between coordinates systems is a function of kinematics.

In order to derive the dynamics equations of the quadcopter, the quadcopter's states should be introduced as well as its transformations between different coordinate frames. These states describe how the quadcopter is going to move in free space. This motion can be **translational**; expressed by linear position and velocity, or **rotational**; expresses by the angular rates and Euler angles of the quadcopter.

- The position of the quadcopter according to the inertial frame is $[\ x\ \ y\ \ z]^T$.
- The velocity of the quadcopter with respect to the body frame $[u\ \ v\ \ w]^T$.
- The angular velocity in the body frame $[p\ \ q\ \ r]^T$.
- The Euler angles (roll, pitch and yaw) $[\varphi\ \ \theta\ \ \psi]^T$.

To represent the body frame vector $\overrightarrow{V^b}$ in the inertial frame $\overrightarrow{V^i}$ three rotational matrices are required.

First, the body frame rotates about the X axis by amount of φ(roll angle). The resulting frame is called $a1$ frame which has the same X axis as the body frame but different Y and Z axes. Secondly, the $a1$ frame is rotated about Y axis by amount of θ(pitch angle). The resulting frame is called $a2$. Finally, $a2$ frame is rotated about Z axis by amount of ψ(yaw angle) to give the inertial frame.

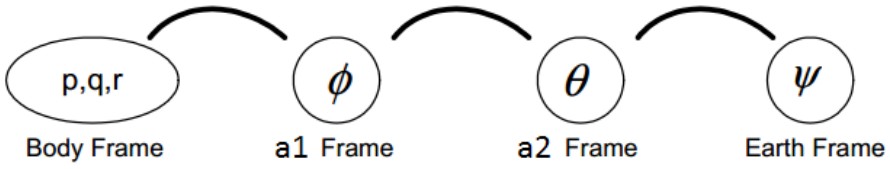


Figure 9: Transformation frames

Each rotation between those four frames has its own primary rotation matrix. Those rotation matrices are expressed as follows:

$$R_b^{a1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{bmatrix}$$

$$R_{a1}^{a2} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_{a2}^{i} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The final rotational matrix is obtained by multiplying the three primary matrices(s and c represent sin() and cos() respectively):

$$R_b^i = R_b^{a1} \times R_{a1}^{a2} \times R_{a2}^i$$

$$R_b^i = \begin{bmatrix} \mathrm{c}(\theta)\,\mathrm{c}(\psi) & \mathrm{s}(\varphi)\mathrm{s}(\theta)\mathrm{c}(\psi) - \mathrm{c}(\varphi)\mathrm{s}(\psi) & \mathrm{c}(\varphi)\mathrm{s}(\theta)\mathrm{c}(\psi) + \mathrm{s}(\varphi)\mathrm{s}(\psi) \\ \mathrm{c}(\theta)\mathrm{s}(\psi) & \mathrm{s}(\varphi)\mathrm{s}(\theta)\mathrm{s}(\psi) + \mathrm{c}(\varphi)\mathrm{c}(\psi) & \mathrm{c}(\varphi)\mathrm{s}(\theta)\mathrm{s}(\psi) - \mathrm{s}(\varphi)\mathrm{c}(\psi) \\ -\mathrm{s}(\theta) & \mathrm{s}(\varphi)\mathrm{c}(\theta) & \mathrm{c}(\varphi)\mathrm{c}(\theta) \end{bmatrix}$$

The representation of the body frame vector in the inertial frame is as follows:

$$\overrightarrow{V^i} = R_b^i \, \overrightarrow{V^b}$$

Euler angles are defined in different frames which are a1, a2 and earth frame. The T(φ, θ, ψ) matrix gives us a way to relate the angular rate vector $[\mathrm{p\ q\ r}]^{\mathrm{T}}$ and Euler angles (15). Angular rate defined in body frame could be written in Earth frame by multiplying it with the inverse of the T(φ, θ, ψ).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = T^{-1}(\varphi, \theta, \psi) \frac{\mathrm{d}}{\mathrm{dt}} \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}$$

The body rotation is achieved by only one rotation for the roll and for the pitch, it requires two successive rotations, for the yaw, it requires three successive rotations (as explained previously). Therefore, each angle is defined in a different coordinate frame; the transformation is expressed as:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = R_b^{a1} \frac{d}{dt} \begin{bmatrix} \varphi \\ 0 \\ 0 \end{bmatrix} + R_b^{a1} R_{a1}^{a2} \frac{d}{dt} \begin{bmatrix} 0 \\ \theta \\ 0 \end{bmatrix} + R_b^{a1} R_{a1}^{a2} R_{a2}^{i} \frac{d}{dt} \begin{bmatrix} 0 \\ 0 \\ \psi \end{bmatrix}$$

$T^{-1}(\varphi, \theta, \psi)$ will be:

$$T^{-1}(\varphi,\theta,\psi) = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\varphi) & \sin(\varphi)\cos(\theta) \\ 0 & -\sin(\varphi) & \cos(\varphi)\cos(\theta) \end{bmatrix}$$

$$T(\varphi,\theta,\psi) = \begin{bmatrix} 1 & \sin(\varphi)\tan(\theta) & \cos(\varphi)\tan(\theta) \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi)\sec(\theta) & \cos(\varphi)\sec(\theta) \end{bmatrix}$$

After obtaining rotational matrix and the transformation matrix, kinematic equations are expressed as:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} R_b^i(\varphi,\theta,\psi) & 0_{3\times3} \\ 0_{3\times3} & T(\varphi,\theta,\psi) \end{bmatrix}\begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}$$

I.5.3 Dynamic modeling:

Dynamics equations mathematically represent a body motion as a function of time and consider all the forces and torques. These equations are constructed writing with Newton-Euler method techniques. All dynamic equations are written with respect to the body frame. The approach is based on the following assumptions (16):

- The quadcopter structure is assumed to be rigid.
- The quadcopter structure is assumed to be symmetrical.
- The quadcopter center of mass and the body frame origin are assumed to coincide with each other.
- The propellers are supposed as rigid.
- The thrust force and hub moment are proportional to square propellers speed.
- The inertial matrix is time invariant.

As pointed out previously, the quadcopter has four vectors (12 states) to define its movement. Motion description of the quadcopter with respect to inertial frame depends on two vectors; position and orientation vectors. The computation of both demands

knowledge of all initial conditions for these vectors. Furthermore, knowledge of net force $\vec{F}$ and net torque $\boldsymbol{\tau}^{\rightarrow}$ that are acting on the drone is required.

In the Newton-Euler formulation, the net external force acting on the body $\vec{F}$ equals the rate of change of the linear momentum $\frac{d\vec{p}}{dt}$, Similarly, the net external torque applied to the body $\vec{\tau}$ is equivalent to the rate of change of the angular momentum $\frac{d\vec{h}}{dt}$.

$$\frac{\mathrm{d}\,\vec{\mathrm{p}}}{\mathrm{dt}} = \vec{F}\,,\ \frac{\mathrm{d}\,\vec{\mathrm{h}}}{\mathrm{dt}} = \vec{\tau}$$

I.5.3.1 Forces affect the Quadcopter:

- **Gravity Effect:**

The gravitational force has a key role in quadcopter flight dynamics by exerting a downward force on the quad (along the positive z axis with respect to the inertial frame). In order for the quadcopter to sustain flight and a balanced hover, the rotors must produce a thrust equivalent to the weight of the quadcopter. Moreover, the precise maneuvering is a result of speed adjustment of the rotors which resists the gravitational force pulling the quadcopter in its trajectory and makes it stable. The force with respect to the body frame ($\overrightarrow{G_b}$) is found by multiplying with matrixR_i^b.

$$R_i^b = R_b^{i\ T}$$

The gravitation force on the quadrotor $\overrightarrow{G^e}$ which is expressed in the inertial frame as [0 0 mg]$^{\mathrm{T}}$. It is transformed into the body frame as follows:

$$\overrightarrow{G^b} = R_i^b\ \overrightarrow{G^e}$$

$$\overrightarrow{G^b} = \begin{bmatrix} -mg\sin(\theta) \\ mg\cos(\theta)\sin(\varphi) \\ mg\cos(\theta)\cos(\varphi) \end{bmatrix}$$

- **Thrust force:**

The thrust force in quadcopter is the up force generated by the rotors. It is essential for overcoming the downward pull of gravity, enabling lift-off, and sustaining flight. While rotating, the propellers push the air downwards, thus causing an equal and opposite reaction that lifts the quadcopter upwards, as per the third law of Newton.

The thrust generated by a propeller is influenced by factors such as the propeller's diameter, rotational speed, and the density of the air in which it operates. Here's an overview of how you can estimate or calculate the thrust produced by a propeller with the help of the blade element theory (17):

$$T = C_T \rho \pi \omega^2 R^4$$

Where ρ is the air density, ω is the rotor speed, R is the rotor radius. They are considered as time invariant. C_T is the thrust coefficient, a time variant parameter.

- **Roll and Pitch moments:**

The quadrotor is a system that is naturally underactuated and provides interaction among the degrees of freedom. At the hover, the thrust vector is constantly being aimed upward, so the angle of tilt must be adjusted to ensure a straight movement of the craft as the pushers cannot move side to side. Roll and pitch moments are a result of variances in the propeller speed. Moreover, as the quadrotor makes rotations along the x-axis, propellers p1 and p4 decrease their speed while p2 and p3 accelerate, giving rise to a thrust imbalance between the left and right sides of the drone. Proportioned by the moment arm, this causes a roll moment (U2). Likewise, changing the speeds of propellers p1 and p3 in conjunction with those of p2 and p4 produces a pitch moment (U3) along the Y-axis.

The quadrotor's arms are perpendicular to each other and there is 45◦ between the body frame axes and the arms. Therefore, the moment arm is equal to $1/\sqrt{2}$ times of the quadrotor's arm (2).

Roll and pitch moments are computed as follows;

$$U_2 = b \frac{1}{\sqrt{2}}(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2)$$

$$U_3 = b \frac{1}{\sqrt{2}}(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$$

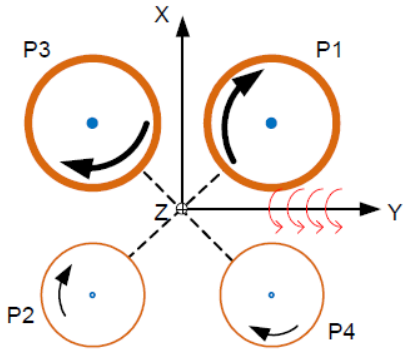


Figure 10: Pitch moment generation

- **Yaw moment:**

The yawing moment is created by the dragging moment of the propeller. The drag moment is the horizontal moment acting about the rotor shaft of the rotating rotor and the moment direction is opposite to the rotation spin. Each propeller produces a drag moment with spinning speed and rotation direction.

Drag moment is given as (18):

$$D = C_Q \rho A(\omega R)^2 R$$

The expression is simplified as (C_Q and ρ are constant, $d \triangleq C_Q \rho \pi R^5$):

$$D = d\omega_i^2$$

The neighboring rotors are required to revolve at the same speed but opposite directions. In that similar way, drag moment of a propeller is counterbalanced by the neighboring propellers and directional moment reduces to zero along the Z axis. The change in the rotor speeds induces an imbalance in the yaw moment which enables the changing of the yaw (heading) angle.

The total yaw moment (U_4) is calculated as;

$$U_4 = \sum_{i=1}^{4} D_i = d(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2)$$

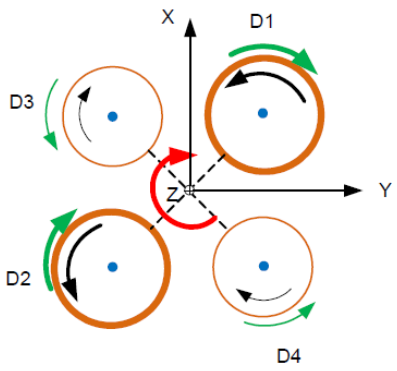


Figure 11: Yaw moment

Further details can be found in (19)

I.6 PID Controller:

PID controller is used to attenuate the error produced from the difference between the desired state and the actual state (20). It is a fundamental component in the control system of quadcopters, playing a critical role in ensuring stable and responsive flight.

The PID controller algorithm involves three separate constant parameters; Proportional (P), Integral (I), Derivative (D). These values can be interpreted in terms of

time: 'P' denotes the current error, 'I' stands for the sum of past errors, and 'D' is the prediction of future errors, which is achieved by seeing the changing rate, at the moment.

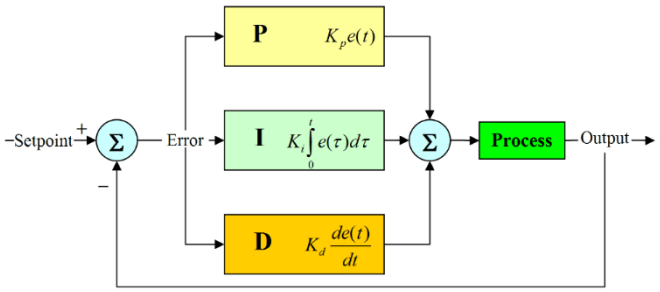


Figure 12: PID Controller structure.

- **Proportional term:**

The proportional term creates an output value that is proportional to the current error value. In the simplest way, the more the error the more the output value and therefore the closer the setpoint is achieved (but beware of overshoot and instability). Proportional response can be adjusted by multiplying the error by a constant Kp, which is known as the proportional gain (21).

The proportional term is given by:

$$P_{Out} = K_P\, e(t)$$

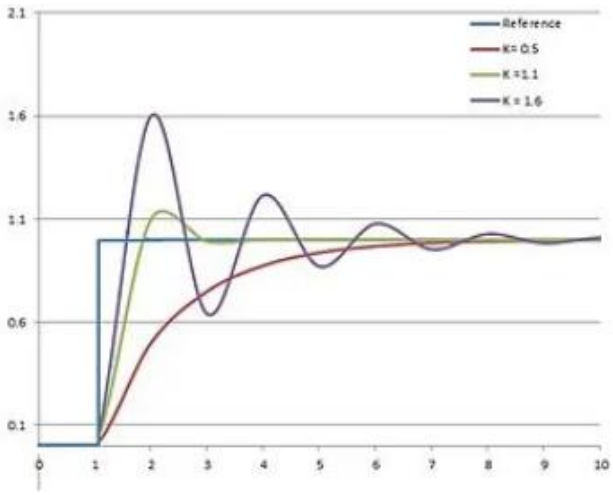


Figure 13: Kp influence in a PID control system (Kd and Ki constants)

- **Integral term:**

The contribution from the integral term is then inversely proportional to both the magnitude of the error and the error duration. PID controller has an integral part, which is the sum of the instantaneous error over the time and it gives the accumulated offset (steady-state error) that should have been corrected previously. The accumulated error is then multiplied by the integral gain (Ki) and added to the controller output. This parameter

removes offset as it rises the nature and order of the system by 1. This parameter also rises the system reaction speed, but at the cost of continued oscillations. (21).

$$I_{Out} = K_i \int_0^t e(t)\, dt$$

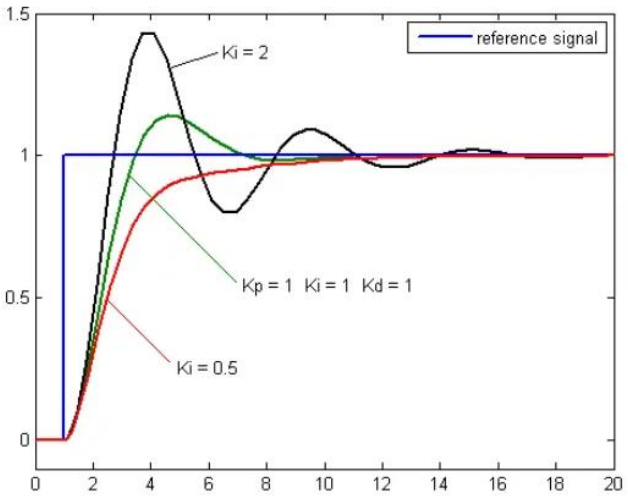


Figure 14: K_i Influence in a PID control system (K_P and K_D constants)

- **Derivative term:**

In essence, this setting reduces the system's oscillatory response. It has no influence on the offset and it has no influence on the system's nature and order. It examines how quickly the error signal changes. The derivative is the cause of the larger system reaction to the increasing speed of change. Using too much derivative term may give the system a highly oscillatory behavior or even a control overshoot (21).

$$D_{Out} = K_D \frac{d\, e(t)}{dt}$$

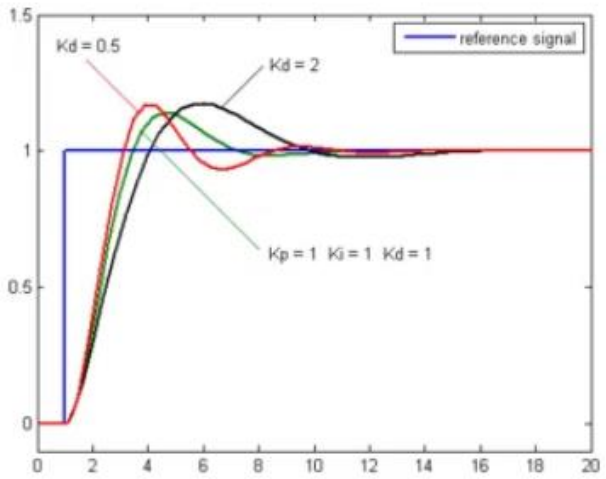


Figure 15: K_D influence in a PID control system (K_i and K_P are constants)

A PID controller is a control loop mechanism that continuously calculates an error value $e(t)$ as the difference between a desired setpoint and a measured process variable, and applies a correction based on proportional, integral, and derivative terms. Both altitude and attitude controller use PID controller to give out control inputs U1, U2, U3 and U4. All of these control inputs are the fundamental in producing the outputs z, θ, φ, ψ.

The system contains four PID controllers, so the tuning process will begin with the altitude controller and disconnecting the other three control inputs. The equations below show the PID controller equations to obtain all control inputs U1, U2, U3, and U4 which are used to produce the relative speed Xr. The control inputs of U2, U3 and U4 come from the use of angular error whereas U1 comes from the use of altitude error.

$$U_1(t) = K_p e_z(t) + K_I \int_0^t e_z(t)dt + K_D \frac{e_z(t) - e_z(t-1)}{dt}$$

$$U_2(t) = K_p e_\varphi(t) + K_I \int_0^t e_\varphi(t)dt + K_D \frac{e_\varphi(t) - e_\varphi(t-1)}{dt}$$

$$U_3(t) = K_p e_\theta(t) + K_I \int_0^t e_\theta(t)dt + K_D \frac{e_\theta(t) - e_\theta(t-1)}{dt}$$

$$U_4(t) = K_p e_\psi(t) + K_I \int_0^t e_\psi(t)dt + K_D \frac{e_\psi(t) - e_\psi(t-1)}{dt}$$

I.7 Conclusion:

This chapter has provided an essential overview of the theory and fundamentals of Unmanned Aerial Vehicles (UAVs), with a focus on quadcopters. We started with an introduction to UAVs and their classification, followed by an in-depth look at quadrotor configurations, specifically the X and + configurations.

We explored the dynamic principles of UAVs, highlighting Newton's third law and the forces acting on a quadcopter, including lift, weight, thrust, and drag. Additionally, we examined the principal axes, which are crucial for understanding the quadcopter's orientation and movements governed by pitch, roll, and yaw.

Overall, this chapter lays the groundwork for understanding the basic principles that will be applied in the modeling, simulation, and practical implementation of quadcopter systems in subsequent chapters.

Chapter II:

Hardware\Software and Design Implementation

II.1 Introduction:

This chapter deals with the software and the hardware used in the development of the quadcopter. And explaining the quadcopters design, and its applications; delivery, Fire detection and fighting, and the Plants diseases detection and treatment.

II.2 Hardware Components:

II.2.1 Arduino Mega 2560:

Arduino Mega is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started (22).

Figure 16: Arduino Mega 2560

Specifications:

- Microcontroller : ATmega2560.
- Operating Voltage : 5V.
- Input Voltage (recommended) : 7-12V
- Input Voltage (limit) : 6-20V
- Digital I/O Pins : 54 (of which 15 provide PWM output)
- Analog Input Pins : 16.
- DC Current per I/O Pin: 20mA
- DC Current for 3.3V Pin : 50 mA
- Flash Memory : 256 KB of which 8 KB used by bootloader
- SRAM : 8 KB
- EEPROM : 4 KB
- Clock Speed : 16 MHz.

- Weight : 37g

II.2.2 MPU6050 (GY521):

The MPU-6050 sensor module contains an accelerometer and a gyro in a single chip. It is very accurate, as it contains 16-bits analog to digital conversion hardware for each channel. Therefore it captures the x, y, and z channel at the same time. The sensor uses the I2C-bus to interface with the Arduino.

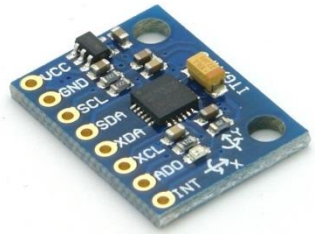


Figure 17:MPU6050(GY521)

Features:

- Power supply: 3-5v (internal low dropout regulator).
- Communication modes: standard I2C communications protocol.
- Chip built-in 16bit AD converter, 16-bit data output.
- Immersion Gold PCB machine welding process to ensure quality.
- Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a fullscale range of ±250, ±500, ±1000, and ±2000dps
- Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g

II.2.3 GPS Module (GY-NEO6MV2):

GY-NEO6MV2 module series is a family of stand-alone GPS receivers featuring the high-performance ublox 6 positioning engine. The module simply checks its location on earth and provides output data which is the longitude and latitude of its position. GY-NEO6MV2 comes with a small battery for hot-start, along with a built-in EEPROM. To offer better signal reception, there is an external ceramic antenna that connects to the board via a U.FL connector

GY-NEO6MV2 GPS module can track up to 22 satellites over 50 channels and achieve the industry's highest level of tracking sensitivity i.e. -161 dB while consuming only 45 mA current. Its compact architecture, modular design, power, and memory options

make NEO6MV2 modules ideal for battery-operated mobile devices with very strict cost and space constraints (23).

Figure 18: GY-NEO6MV2 GPS Module

II.2.4 HC-05 Bluetooth Module:

The HC-05 zs-040 Bluetooth module is an integral component of the quadcopter's communication system, facilitating wireless data transmission between the quadcopter and a remote control unit. This module supports Bluetooth 2.0 and provides a reliable serial communication link, making it ideal for real-time control and telemetry (24).

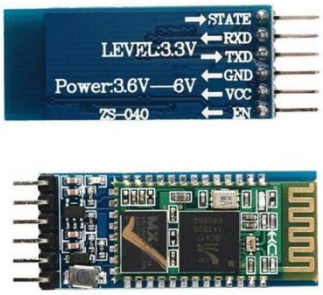


Figure 19: zs-040 HC-05 Bluetooth module

Key Features:

- **Bluetooth Version:** 2.0+EDR (Enhanced Data Rate)
- **Frequency:** 2.4GHz ISM band
- **Modulation:** GFSK (Gaussian Frequency Shift Keying)
- **Baud Rate:** Configurable from 9600 to 115200 bps (default 9600 bps)
- **Operating Voltage:** 3.3V (compatible with 5V systems using a voltage divider or level shifter)
- **Communication Range:** Up to 10 meters in open space
- **Profiles Supported:** Serial Port Profile (SPP)
- **Interface:** UART with TTL logic level

II.2.5 DJI F450 frame:

This quadcopter carbon fiber frame is one of the simplest, most common, and most durable frames. The frame comes with a prebuilt power distribution board that is included in the interior part of the lower core plate which eliminates the use of DIY wiring harnesses. This frame makes the building process easier with the possibility of additional accessories to the quadcopter design. It is very light in weight, strong and as an appropriate fitting for the other components.

Figure 20:DJI F450 Quadcopter frame

II.2.6 EMAX XA2212/980KV Brushless Outrunner:

The Emax 980KV Brushless Motor is an Emax XA2212 Outrunner Brushless Motor Module having power of 980 KV. It is composed of various features like: - High power and efficiency, High qualitative bearings, Low price with rich fittings, Silicon steel and magnets that ensures its functionality and durability and many more. It is mainly used in 4 rotors helicopters, fixed wing aircraft and 3D planes or puller prop trainers with a wing-span of 1m to 1.2 m etc (25).

Figure 21: EMAX XA2212 980KV Outrunner

Specifications:

Operating Voltage Range: 7.2 to 13 V

Current : Maximum of 15 Amps for 30 Seconds

Motor KV (RPM/V) : 980

Maximum Thrust (gm): 880

Compatible LiPO Batteries: 2S ~ 3S

Shaft Diameter (mm) : 3

Framework: 12N14P

Weight (g): 55

II.2.7 Electronic speed controller(Skywalker):

Hobbywing Skywalker 30A brushless electronic speed controller with 2A 5V BEC supply suitable for use with most RC Arcraft including helis, fix wing, and multicopters. The ESC also includes 3x 3.5mm gold plated bullet connectors for connection to a brushless motor and one Deans connector for connection to a battery (26).

Figure 22: SKYWALKER 30A electronic speed controller

Specifications:

- Output power: constant current 30A, 40A Burst (not more than 10 seconds)
- Power input: 2-3S lithium battery or section 5-9 cells ni-mh/ni-cd battery pack.
- BEC output: 5 v, 2A (linear voltage regulation mode).
- Maximum speed: 2 pole motor 210000 r/min, 6 pole motor 70000 r/min BLM, 12 pole 35000 r/min BLM.
- Size: 68 mm (length) * 25 mm (width) * 8 mm (high).
- Weight: 37 g.

II.2.8 ArduPilot Mega(APM):

APM is a professional quality IMU autopilot that is based on the Arduino Mega platform. This autopilot can control fixed-wing aircraft, multi-rotor helicopters, as well as traditional helicopters. It is a full autopilot capable for autonomous stabilization, way-

point based navigation. Supporting 8 RC channels with 4 serial ports. ArduPilot Mega consists of the main processor board and the IMU shield which fits above or below it (27).

Figure 23: ArduPilot Mega 2.8

APM Features:

- Free open source autopilot firmware that supports planes, multicopters (tri, quad, hex, oct, etc), traditional helicopters and ground rovers!
- Simple setup process and firmware loading via a point-and-click utility. No programming required! (But if you do want to fiddle with the code, you can with the easiest embedded programming toolkit available: Arduino).
- Full mission scripting with point-and-click desktop utilities.
- Can support hundreds of 3D waypoints.
- Two-way telemetry and in-flight command using the powerful MAVLink protocol.
- Choice of free Ground Stations, including the state-of-the-art HK GCS, which includes mission planning, in-air parameter setting, on-board video display, voice synthesis, and full datalogging with replay.
- Autonomous takeoff, landing and special action commands such as video and camera controls.
- Supports full "hardware-in-the-loop" simulation with Xplane and Flight Gear.
- 4MB of onboard data-logging memory. Missions are automatically datalogged and can be exported to KML.
- Built-in hardware failsafe processor, can return-to-launch on radio loss.

II.2.9 LiPo battery :

The battery gives energy to the system. For the quadcopter, the weight and capacity of the battery bound the flight time and agility. The lithium polymer (LiPo) batteries are the most common power supply for the quadcopter because they have very good power to

weight ratio and they are lighter than other types of batteries (28).

Figure 24: 3s LiPo Battery

Specifications:

- Capacity: 2200mAh
- Voltage: 11.1V / 3-Cell / 3S1P
- Discharge Rate: 70C Continual / 140C Burst
- Charge Rate: 5C Max
- Size(1-5mm difference): 31X35X106mm
- Approx Weight(±5g) : 225g
- Output Connector: XT60
- Balance Connector: JST / XH

II.2.10 FlySky RC Controller:

The FlySky RC controller is a popular choice among hobbyists and enthusiasts in the radio control (RC) community due to its affordability and reliability. FlySky offers a range of transmitters and receivers that are compatible with various types of RC vehicles, including drones, airplanes, helicopters, and cars (29).

Figure 25: FlySky Transmitter and Receiver

Specifications:

Channels:	6
RF range:	2.408 - 2.475GHz
Bandwidth:	500 KHz
RF power:	Less than 20 dBm
Protocol:	AFHDS 2A
Modulation type:	GFSK
Stick resolution:	4096
Low voltage alarm:	Yes (lower than 4.2V)
PS2/USB Port:	Yes (Micro-USB)
Receiver output:	PWM, PPM, iBus
Power input:	6V DC 1.4AA*4
Weight:	392g
Receiver weight:	6,4 g
Size:	174*89*190 mm

II.2.11 Raspberry Pi B4:

Raspberry Pi 4 Model B features a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro HDMI ports, hardware video decode at up to 4Kp60, up to 8GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on). For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems (30).

Figure 26: Raspberry pi model B4

Specifications :

Processor:	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory:	1GB, 2GB, 4GB or 8GB LPDDR4 (depending on model) with on-die ECC
Connectivity:	• 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE • Gigabit Ethernet • 2 × USB 3.0 ports • 2 × USB 2.0 ports.
GPIO:	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & sound:	• 2 × micro HDMI ports (up to 4Kp60 supported) • 2-lane MIPI DSI display port • 2-lane MIPI CSI camera port • 4-pole stereo audio and composite video port
Multimedia:	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD card support:	Micro SD card slot for loading operating system and data storage
Input power:	• 5V DC via USB-C connector (minimum 3A1) • 5V DC via GPIO header (minimum 3A1) • Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature 0–50ºC

II.3 Software:

II.3.1 Arduino IDE:

Arduino Integrated Development Environment or Arduino IDE is an application software which is used to upload the developed programs into the Arduino-boards. It is received with its interface and libraries as the most important aspects to help in the creation as well as deployment of the software. It offers a basic command prompt through which one can write code and experiment with the microcontroller, which can be enhanced with

numerous libraries to help in interfacing of various sensors and actors. It can be used on Windows, Mac and Linux operating systems to afford access to as many users as possible.

For code development, Arduino programs are referred to as sketches and consist of two main functions: This is done by using two methods, namely; setup() and loop().

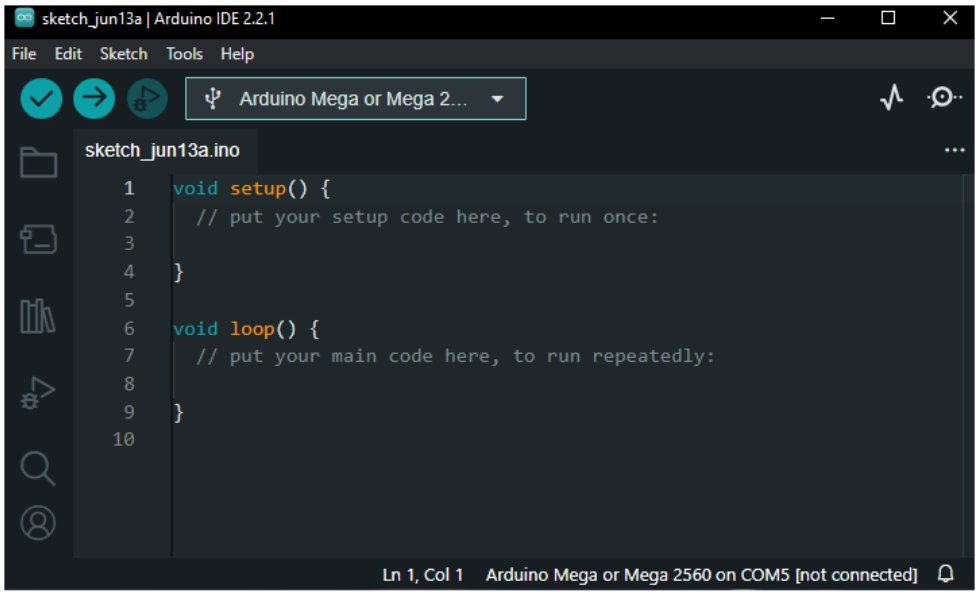


Figure 27: Arduino IDE interface

II.3.2 MATLAB Simulink:

Simulink, a component of MATLAB, provides a block diagram environment for multidomain simulation and model-based design. It is particularly useful for the design and simulation of control systems, which are integral to the operation of Unmanned Aerial Vehicles (UAVs).

Simulink provides a graphical interface that simplifies the design and understanding of complex systems. Seamless integration with MATLAB allows for powerful script-based analysis and data processing.

It enables rapid prototyping and testing through simulation, reducing the need for physical prototypes during the initial stages of development.

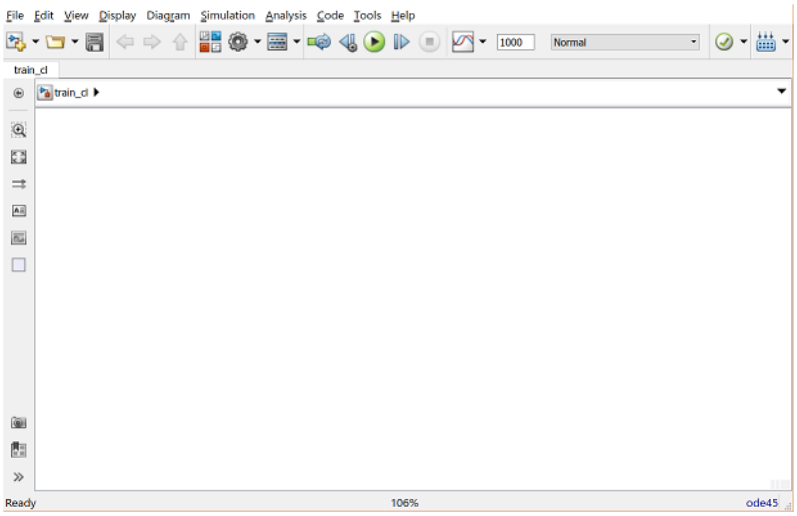


Figure 28: MATLAB Simulink interface

II.3.3 FreeCAD :

FreeCAD is a parametric 3D computer-aided design (CAD) modeling application. Parametric design is a method of designing objects based on defining parameters and relationships between them. In this approach, changes to parameters automatically propagate throughout the associated elements of the design, allowing for rapid iterations and adaptations.

Commonly used in fields such as architecture, engineering, and product design, parametric design helps create complex shapes and forms efficiently, with the ability to adjust and fine-tune various design aspects through the modification of input values. (31).

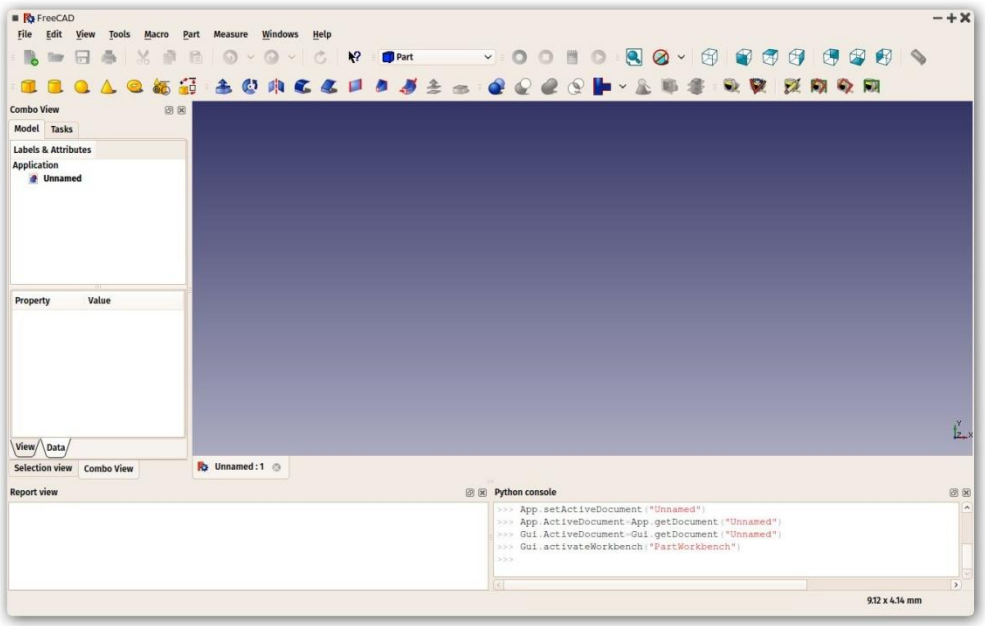


Figure 29: FreeCAD interface

II.4 Drone design:

The proposed drone design is intended to serve multiple purposes, encompassing delivery, fire detection and fighting, and plant disease detection and treatment.

The structural foundation of our drone is the DJI F450 frame, renowned for its durability and lightweight carbon fiber construction. This X-frame configuration provides a balanced and stable platform ideal for diverse applications. The drone is equipped with four EMAX 980KV brushless motors, paired with 10-inch carbon fiber propellers. This propulsion system is optimized for efficiency and performance, delivering a high thrust-to-weight ratio essential for carrying additional payloads and equipment. A 3S LiPo battery powers the system, ensuring sufficient energy for extended flight durations required by each application.

II.4.1 Delivery:

For delivery purposes, the drone is designed to carry small to medium-sized packages efficiently and reliably. The lightweight and sturdy DJI F450 frame supports payload mounts, while the powerful EMAX motors provide the necessary lift and stability. The drone's flight controller can be programmed for precise navigation and autonomous delivery routes, ensuring timely and accurate drop-offs. The integration of GPS modules aids in real-time tracking and navigation

II.4.2 Fire detection and fighting:

In fire detection and fighting, the drone utilizes advanced image processing capabilities facilitated by a Raspberry Pi. The drone can detect heat signatures and identify fire hotspots. The image data is processed in real-time by the Raspberry Pi, enabling early detection of fires and providing critical information to firefighting teams. Additionally, the drone can be equipped with payloads to drop fire retardants or deliver supplies to firefighting personnel in hard-to-reach areas.

II.4.3 Plant disease detection and treatment:

Similarly, the drone utilizes image processing for plant disease detection and treatment. The Raspberry Pi processes multispectral images to identify signs of disease in crops, such as discoloration or unusual patterns in foliage. Once a disease is detected, the farmer lances the order to apply targeted treatments using attached spraying mechanisms.

II.5 System Design:

II.5.1 Block Diagram of the Control System:

Our block diagram of quadcopter illustrates the arrangement of the linking components which carry out a total control of the aircraft with the main flight controller in the center. This one has been assigned to the receiver role which catches data from the transmitter. It also sends signals to the electronic speed controllers (ESCs) which make variations in the speed of Brushless DC motors to regulate the flight path of the Drone. Moreover, a highly accurate GPS module adjacent to the flight controller allows for real-time navigation with accurate location information. All these parts are supplied by a LiPo battery that is connected through a power distribution device.

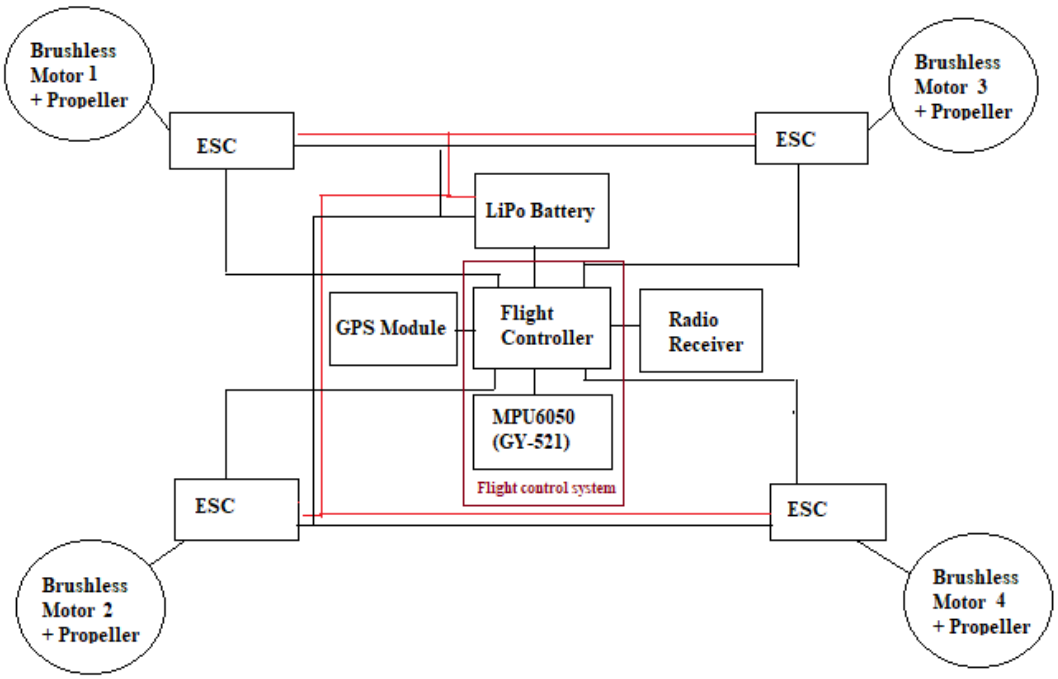


Figure 30: Block diagramme of the QuadCopter

II.5.2 Control system architecture:

For the control system we are developing, our quadcopter has 4 inputs which spin the propellers generating thrust forces and torques. The output that we want is to hover the drone in a fixed altitude, to make this possible we should command the Throttle, Pitch, Roll and yaw independently using the following motor algorithm:

U1 = Throttle + Yaw +Pitch + Roll

U2 = Throttle + Yaw – Pitch – Roll

U3 = Throttle – Yaw + Pitch – Roll

U4 = Throttle – Yaw – Pitch + Roll

We command the thrust by making all the motors in the same speed (Throttle), and we can create the Yaw by increasing two motors that spin in the same direction and decreasing the others, pitch is created by increasing and decreasing the front motor pair and then command the back pair in the opposite direction, and the same for the roll but with the right and left motors.

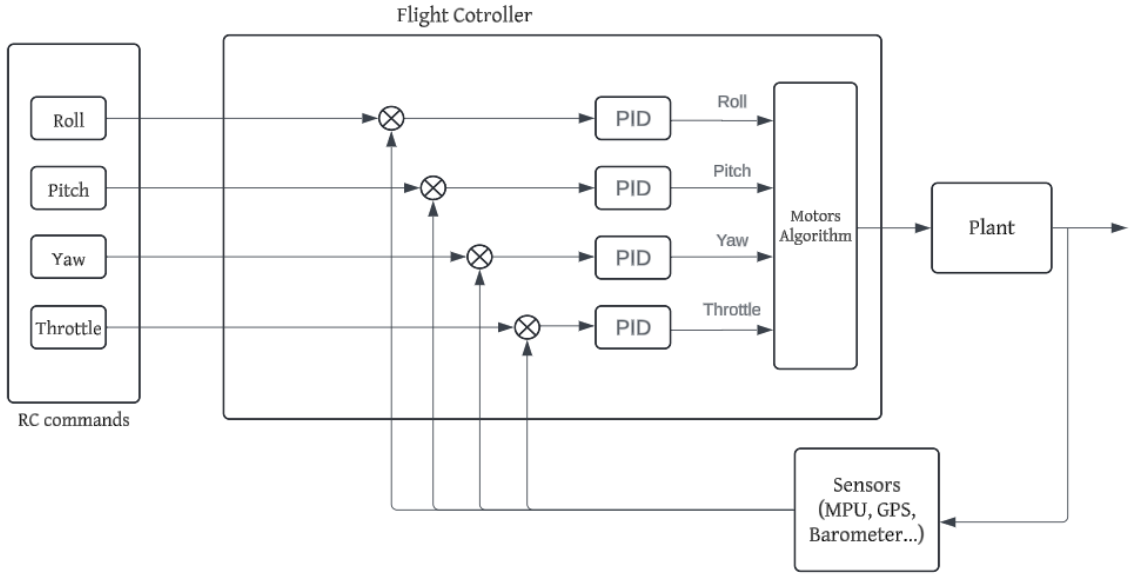


Figure 31: Control system architecture

This control system maintains the altitude and keeps the quadcopter leveled with ground, but to keep the drone in the same position while hovering even with the wind drug and gusts that affects it, we add a position controller to our flight control system.

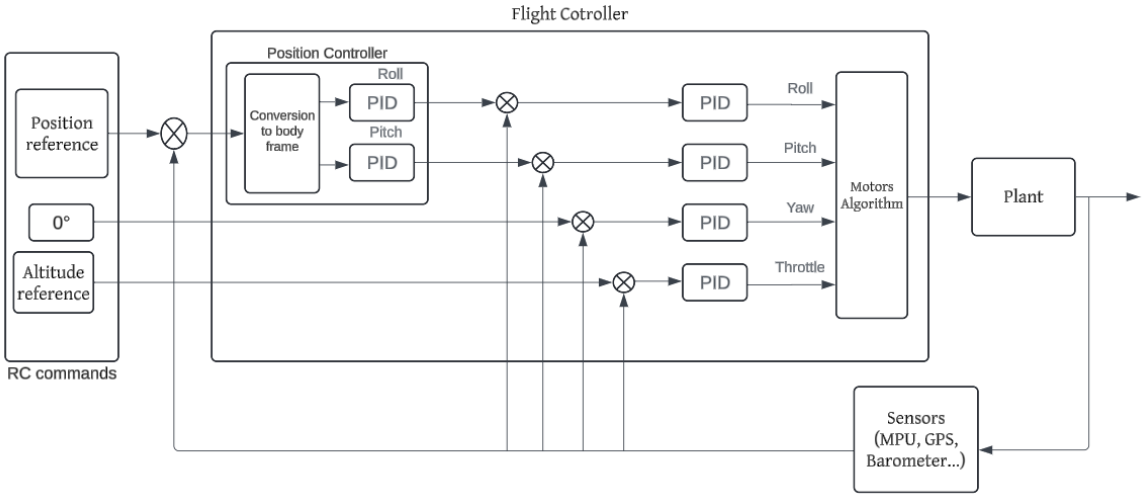


Figure 32: Altitude holds Control system architecture

The position controller uses the position reference and the yaw estimation to determine the roll, pitch and yaw or a combination to achieve the desired position. The error between the actual position of the drone and the reference position is the input of the

position controller, which determines the right roll and pitch degrees to reach the right position.

II.6 Conclusion:

In this chapter, we have provided an overall picture of the quadcopter project activities that will require the hardware and software tools, control system and drone design and architecture to be executed. Physical components such as the microcontroller, motors, propellers, sensors, and power system offer a stable and dependable framework for the quadcopter.

Chapter III:

Image Processing.

III.1 Introduction:

Deep learning has had a tremendous impact on various fields of technology in the last few years. One of the hottest topics buzzing in this industry is computer vision, the ability for computers to understand images and videos on their own. Self-driving cars, biometrics and facial recognition all rely on computer vision to work. At the core of computer vision is image processing.

In this chapter we will go dive into Image Processing, what it is and how we can use it in our lives; also we will see some real applications that have been done by us.

III.2 Types of images:

III.2.1 Binary Image:

Binary images are those that have just two possible pixel intensity values: 0 for black and 1 for white. These pictures are usually used to draw attention to a specific area of a colorful picture. As demonstrated below, image segmentation is one popular application for it.

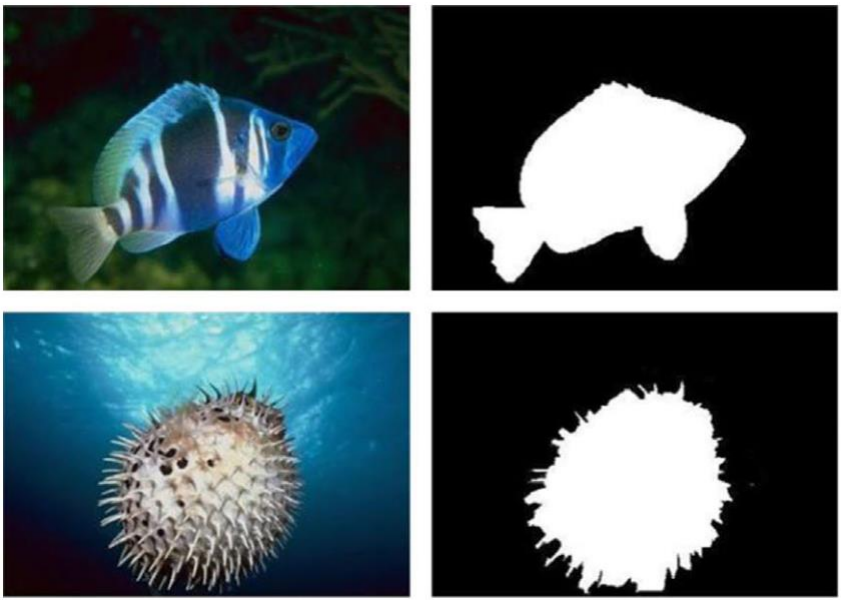


Figure 33: Binary Image

III.2.2 Grayscale Image:

A pixel intensity of 0 denotes black, and a pixel intensity of 255 denotes white in grayscale or 8-bit images, which are made up of 256 distinct colors. The many shades of gray comprise all 254 additional values that fall in between.

Below is an example of an RGB image that has been converted to grayscale. Observe how the RGB and grayscale photos' histograms have the same form.

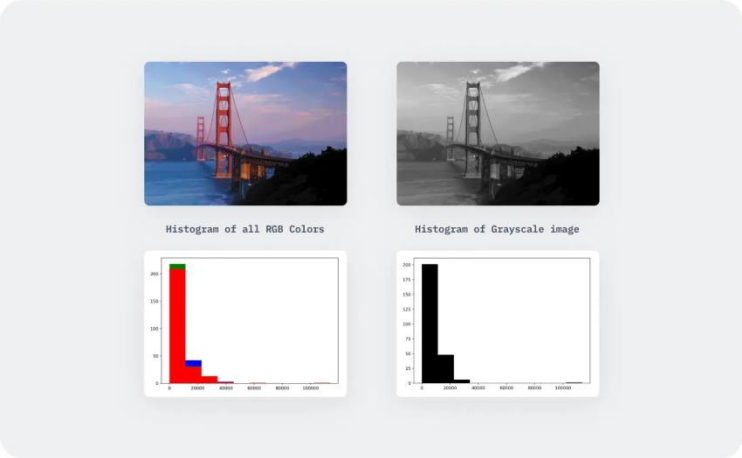


Figure 34: Grayscale Image

III.2.3 RGB Color:

The bright visuals that humans are used to seeing in the modern world are actually 16-bit matrices to computers. This implies that there are 65,536 possible color combinations for each pixel. An image's Red, Green, and Blue "channels" are denoted by "RGB". (32)

A pixel with a value of (0, 0, 0) will be black in an RGB image, and a pixel with a value of (255, 255, 255) will be white. Any combination of numbers in between can produce any of the many colors found in nature. For example, the color red is (255, 0, 0) since only the red channel is active for this pixel. Likewise, (0, 255, 0) is red, while (0, 0, 255) is blue. In a similar vein, (0, 0, 255) is blue while (0, 255, 0) is green.

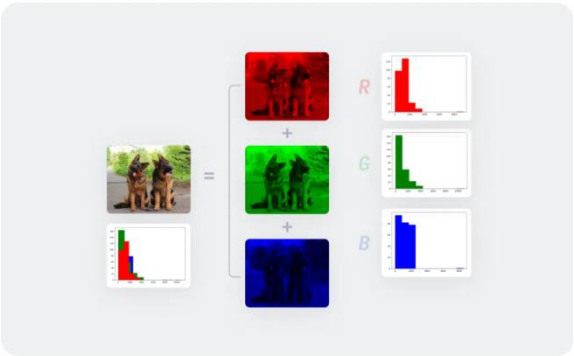


Figure 35: RGB Color Image

III.2.4 RGBA Image:

RGBA photos are RGB images, with an additional channel called "alpha" that shows how opacity is applied to the RGB image. Opacity is basically a "see-through" quality, with a value ranging from 0% to 100%.

In physics, opacity describes how much light can flow through an item. For example, wood is opaque, frosted glass is translucent, and cellophane paper is transparent (100 percent opacity). In RGBA photos, the alpha channel attempts to replicate this characteristic. This is demonstrated in the example below.

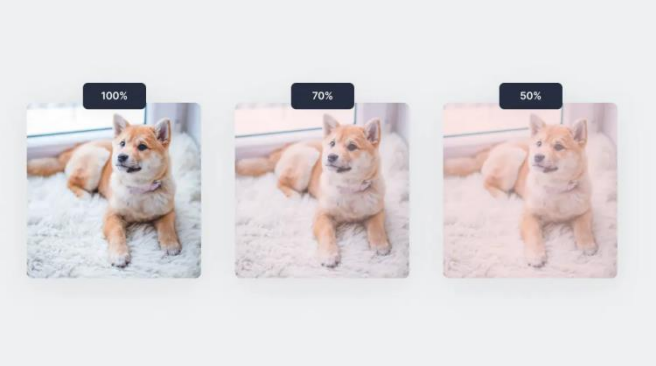


Figure 36: RGBA Image

III.3 Image processing techniques:

III.3.1 Image enhancement:

One of the most common image processing operations is improving an image, or increasing its quality. It is necessary for tasks involving surveillance, remote sensing, and computer vision. A common technique is to adjust the image's contrast and brightness.

Contrast is the brightness difference between the lightest and darkest areas of an image. Increasing contrast will increase an image's overall brightness and make it easier to view. Brightness is the total amount of lightness or darkness in an image. Increasing an image's brightness can make it lighter and easier to view. In addition to having the ability to change contrast and brightness automatically, the majority of image editing programs also let you adjust them manually.

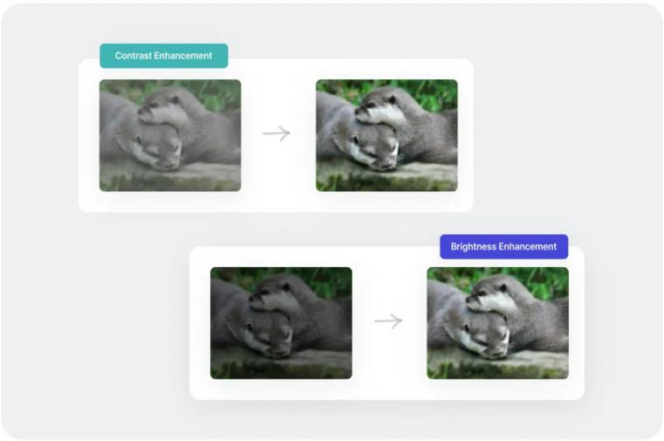


Figure 37: Image enhancement

III.3.2 Image restoration:

The quality of images could degrade for several reasons, especially photos from the era when cloud storage was not so commonplace. For example, images scanned from hard copies taken with old instant cameras often acquire scratches on them.

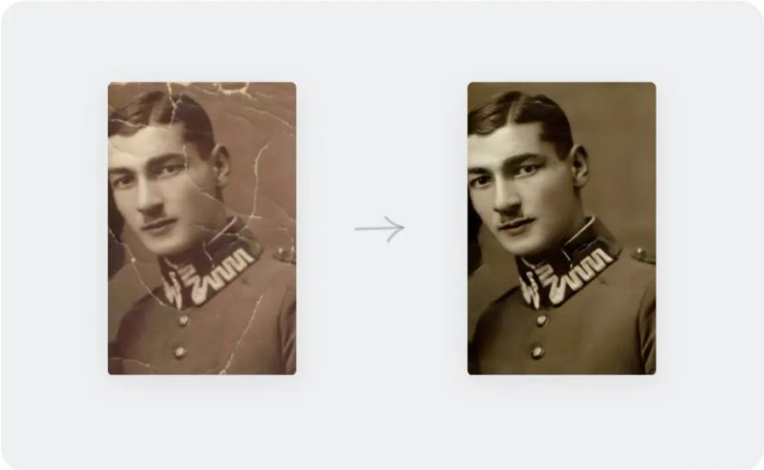


Figure 38: Image restoration

III.3.3 Object Detection:

Object detection is the process of recognizing things in a picture, and it is widely employed in security and surveillance systems. Many alternative techniques can be used for object detection, however the most prevalent method is to employ Deep Learning models, notably Convolutional Neural Networks (CNNs).

Figure 39: Object detection

III.3.4 Image compression:

The technique of image compression is used to reduce an image's byte size without sacrificing its quality below a certain point. More photos can be saved in a given amount of disk or memory space by lowering the file size. Additionally, the image uses less bandwidth when it is downloaded from a webpage or transferred over the internet, which clears network congestion and expedites the delivery of content.

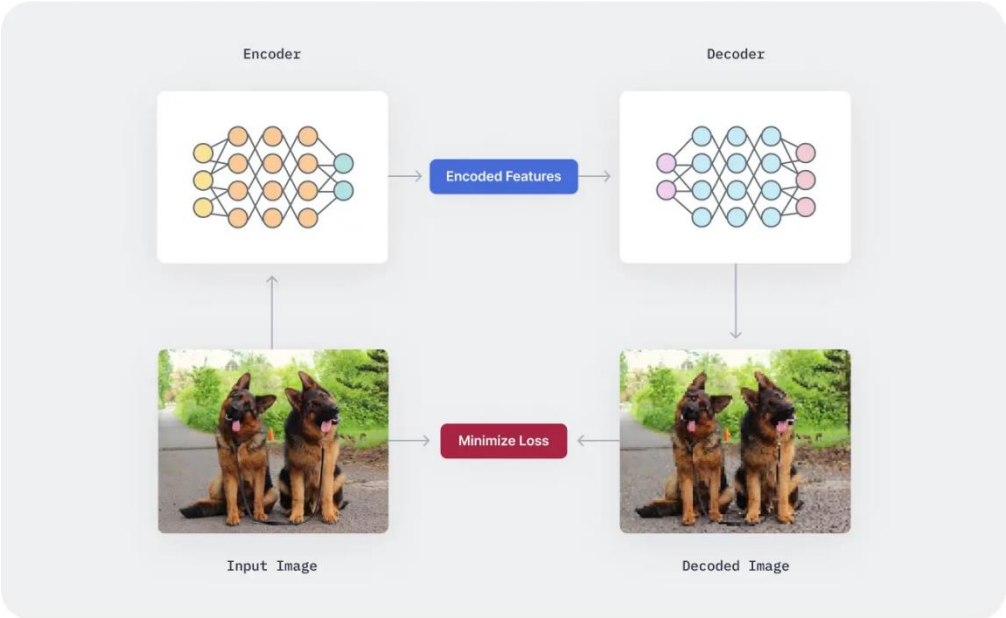


Figure 40: Image compression

III.3.5 Image manipulation :

Fundamentally, image manipulation is the art and science of transforming an image in order to accomplish a specific goal. It's a tradition that dates back to the first images to be created, progressing from physical photo editing to the modern computer manipulations.

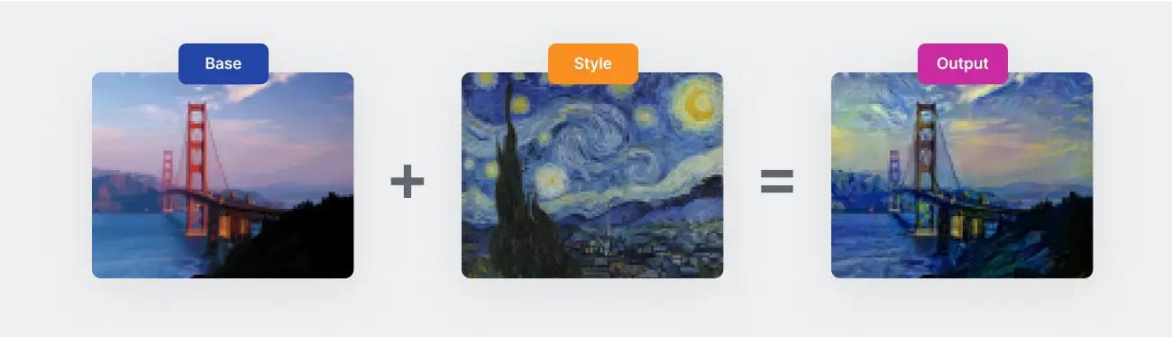


Figure 41: Image manipulation

III.3.6 Image generation:

Another crucial duty in image processing is the synthesis of fresh images, particularly for Deep Learning algorithms that need a lot of labeled data to learn. Another novel neural network design, Generative Adversarial Networks (GANs), are commonly used in image generating techniques.

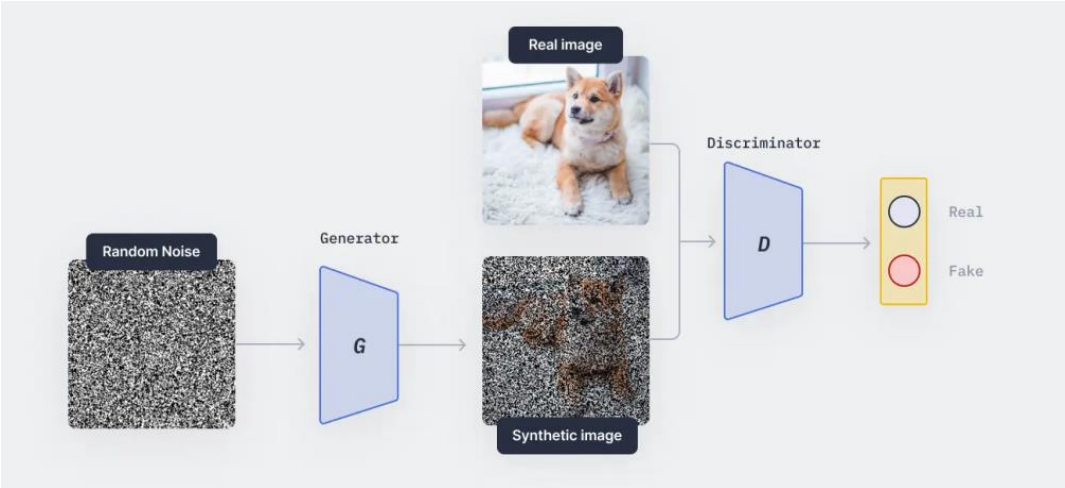


Figure 42: Image generation

III.3.7 Image to Image translation :

Image-to-Image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. For example, a free-hand sketch can be drawn as an input to get a realistic image of the object depicted in the sketch as the output, as shown below.

Figure 43: Image to Image translation

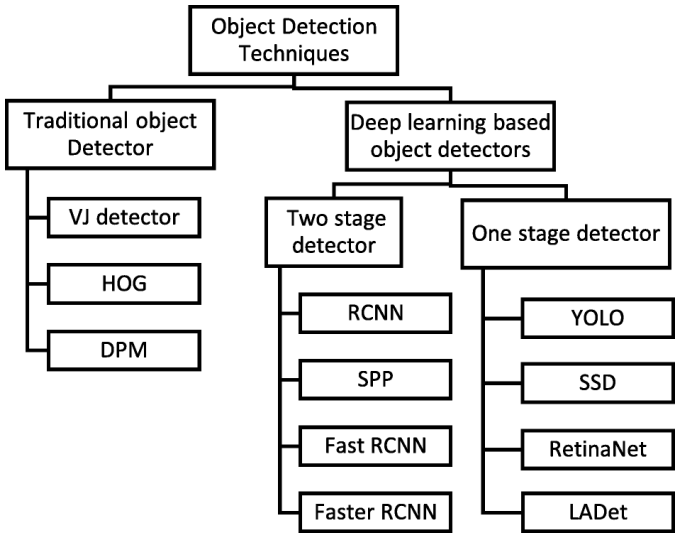
III.4 Object detection:

The fundamental task of computer vision is object detection, which involves identifying and localizing objects in images or video streams. Object detection differentiates itself from image classification by detecting multiple objects and delineating their precise boundaries instead of assigning a single label to an entire image.

III.4.1 Object detection techniques:

Object detection is a powerful aspect of computer vision, and its primary objective

is to locate objects and categorize them in a given image. There are two categories in the object detection framework. This figure illustrates how different object detection methods are organized into two groups: traditional detectors and deep learning-based detectors. The deep learning detectors can be further subdivided into sub-parts such as 'Two-stage detector' and 'One-stage detector'.



III.4.1.1 Traditional detectors:

Traditional object detection techniques favored the use of manually crafted features. Complex feature vectors were the only method used by most researchers due to the limited computational resources and incomplete development of advanced image representation methods at that time. Traditional detectors use methods such as the VJ (Viola-Jones) detector, HOG, and SIFT to identify the object.. Traditional detectors use methods such the SIFT, HOG, and VJ (Viola-Jones) detectors to identify the item. Because feature descriptors were meticulously crafted to accomplish the regions of interest in the image, traditional object detection techniques were successful. The impressive outcome was achieved with a classifier and feature representation on the Pascal VOC dataset.

- **Viola-Jones algorithm (VJ):**

It is a machine-learning technique for object detection. It was proposed in 2001 by Paul Viola and Michael Jones in their paper titled "Rapid object detection using a boosted cascade of simple features." Initially conceived for face detection, it efficiently analyzes windows of different sizes and positions in a grayscale image to detect specific objects by looking for specific image features. (33)

- **Histograms of Oriented Gradients:**

It is a description of features that are used in computer vision and image processing. In the specific parts of an image, it records occurrence of gradients orientation. HOG operates on a very large grid of uniformly spaced cells, which uses overlapping local contrast adjustment for better accuracy than edge orientation histograms. (34)

- **DPM:**

It is an object detection model that combines a Dalal-Triggs detector (based on HOG features) with a star-structured part-based model. It enriches the original model by incorporating parts filters and deformation models. DPMs operate on a scale-space pyramid of gradient orientation feature maps. (35)

III.4.1.2 Deep Learning based object detectors:

- **Two stage detectors:**
  - **R-CNN :**

Region-based Convolutional Neural Network (R-CNN) is a type of deep learning architecture used for object detection in computer vision tasks. RCNN was one of the pioneering models that helped advance the object detection field by combining the power of convolutional neural networks and region-based approaches. (36)

  - **Spatial Pyramid pooling (SPP) :**

It is a pooling layer that removes the fixed-size constraint of the network, i.e. a CNN does not require a fixed-size input image. Specifically, we add an SPP layer on top of the last convolutional layer. The SPP layer pools the features and generates fixed-length outputs, which are then fed into the fully connected layers (or other classifiers). In other words, we perform some information aggregation at a deeper stage of the network hierarchy (between convolutional layers and fully connected layers) to avoid the need for cropping or warping at the beginning. (36)

  - **Fast R-CNN:**

It is an object detection model that improves its predecessor **R-CNN** in a number of ways. Instead of extracting CNN features independently for each region of interest, Fast R-CNN aggregates them into a single forward pass over the image; i.e. regions of interest

from the same image share computation and memory in the forward and backward passes. (37)

- **Faster R-CNN**

It is an object detection architecture presented by Ross Girshick, Shaoqing Ren, Kaiming He and Jian Sun in 2015, and is one of the famous object detection architectures that uses convolution neural networks like YOLO (You Look Only Once) and SSD (Single Shot Detector). (38)

- **One stage detectors:**

- **YOLO (You Only Look Once):**

Proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection. (39)

- **The single shot multibox detector (SSD):**

It is an algorithm for identifying objects in images that uses a feed-forward convolutional neural network (CNN).

- **RetinaNet:**

It is a one-stage object detection model that utilizes a focal loss function to address class imbalance during training. Focal loss applies a modulating term to the cross entropy loss in order to focus learning on hard negative examples. RetinaNet is a single, unified network composed of a backbone network and two task-specific subnetworks. (40)

- **LADet**

Lightweight and Adaptive Network for Multi-Scale Object Detection is a novel one-stage object detector that consists of two core modules:

Adaptive Feature Pyramid Module (AFPM): Unlike the top-down approach, AFPM generates complementary semantic information for each level of feature maps by jointly utilizing multi-level feature maps from the backbone network.

Light-weight Classification Function Module (LCFM): LCFM allows for the exploitation of more types of anchor boxes without significantly increasing the model's

parameters. It achieves this through interleaved group convolution.

III.4.2 Implementing Object detection model using deep learning:

- **Dataset collection:**

The quality of datasets is the sole factor that determines the outcome of computer vision algorithms based on data. Considering object categories like images, videos, etc. is the initial step in data collection. Using resources like online websites and manual image capturing using the camera is a good method to collect data. Images obtained from the Flickr photo-sharing website comprise the majority of the dataset. Because specific phrases are not used as often in that search engine, simple query strings only return a small number of photos (showing only a few hundreds to thousand). Consequently, by enlarging the query set or translating the searches into additional languages like Italian, Dutch, Spanish, etc., a sizable collection of photos can be obtained. A few variables are carefully taken into account when creating the dataset, including lighting, orientation, stance, and the type of environment (sunny, cloudy, daytime, or nighttime) when selecting which samples to include and exclude. After collecting an adequate amount of data, a data cleansing process is necessary. The accuracy and performance of the model depend on the quality of the data. For large models, perfect data collecting is a difficult task. Therefore, collecting data from different sensors or cameras is not directly used for training. Some pre-processing tasks are required to clean or improve the data's quality. (41)

- **Dataset annotation:**

The database can be maintained and scaled easily with images stored in the dataset and additional formal representation. Metadata is the basis of formal representation which includes information like date, location, physical properties, and symbolic descriptions, and it can be presented in either a free or constrained format. The field of object detection and recognition research requires a significant amount of data with ground-truth labels, and additional data is utilized for learning and evaluation. The image's labeling of an object provides information about its shape, location, identity, and pose. Due to the annotations of many objects in images, building a large dataset can be costly and lengthy. As a result, the existing dataset consists of a restricted category of data. When annotating, the bounding box is drawn around the identified objects and holds their properties. Computer vision projects can only succeed if the annotated data is of high quality. The machine learns through input samples in the same way. The annotated labels fed during the training phase

are the basis for the object detection model's result. If inaccurate data is fed into the model while training, the trained model's results will also be incorrect.

Annotations on images are made using Bounding Box, semantic segmentation, polygon segmentation, lines, splines, and other annotations. Define the boundary or location of objects in an image with Bounding Box, which is the most frequently used annotation. The majority of objects are not rectangular, so a polygon shape is employed to pinpoint those types of objects. Every pixel in the image is classified using semantic segmentation, which is a pixel-wise annotation method. Detecting objects, such as lane detection, can be achieved using line and spline annotations. As a result, numerous annotation tools are utilized to label objects in images based on their characteristics. COCO, PASCAL VOC, and YOLO are just some of the techniques that use various annotation formats, like JSON, XML, and plain text. Annotators, inspectors, and examiners perform all annotation processes in the annotation pipeline, which has been used by certain authors. Annotation tools that are either free or require a premium license have been developed in recent years. The following section briefly shows some of the tools:

- **MakeSense:**

It is an open-source online annotation tool designed for labeling photos. It allows users to annotate images easily for various purposes, such as object detection, image recognition, and more. (42)

- **LabelImg:**

It is a free, open source tool for graphically labeling images. It’s written in Python and uses QT for its graphical interface. It’s an easy, free way to label a few hundred images. (43)

- **LabelMe:**

It is a database and digital image annotation tool that helped in many research areas of computer vision. It is a free, open annotation tool that helps to create a large image database. LabelMe is developed at MIT CSAIL (Computer Science and Artificial Intelligence Laboratory. The annotation files are exported in XML file formats. (41)

- **Preprocessing the dataset:**

After annotating the dataset, the next stage involves image preprocessing, which encompasses resizing to a standard size, pixel normalization, and employing data

augmentation methods like rotation, flipping, or zooming to prevent overfitting.

- **Training:**

In order to reduce the error between predicted and actual outputs, backpropagation involves iterative adjustment of the model's weights. The most commonly used loss functions are the mean error squared MSE for the bounding box regression and the cross entropy for the classification. The weights are updated by optimizers such as Adam or SGD. It can be time consuming to train, particularly for large datasets, but the power of GPUs is accelerating it.

- **Evaluation:**

Assess the trained model with a validation set using metrics such as mean average precision (mAP). Adjust hyperparameters or architecture as needed based on the evaluation outcomes.

- **Deployment:**

After training and evaluating the model, the model will be deployed to the raspberry pi 4, where it can be used for object detection by importing images and creating bounding boxes around objects in the image based on the output.

III.4.3 Benefits and limitations:

- **Benefits:**

Biometrics is crucial for security because of the increasing technological development of technology. Individual identity can be identified more reliably through biometric authentication. Various biological characteristics like fingerprints, DNA, retinal scans, and ear lobes are used to carry out authentication (44). In previous research, biometric analysis has been carried out using various types of object detection techniques

Technical video surveillance systems have replaced human monitoring of video clips in living areas like metros, parks, schools, and shopping centers. Object detection plays an important role in video surveillance to identify and track instances of a particular object in a scene at once, e.g., tracking the suspected person or vehicle from the video (45).

In recent years, autonomous robots have been proven to be one of the most interesting research areas. Object detection is the primary task that the robot performs to identify nearby objects and perform some operations such provide information, open-close

the door, alarm, etc. (46).

Human detection is also challenging in computer vision because people as objects have an issue with various appearances and adopt a wide range of poses. Pedestrian detection and other object detection algorithms have been proposed to identify human beings in images or videos. Crowd counting in densely populated areas like parks, malls, etc. has been done very quickly using object detection. (47).

- **Limitations:**

Multiclass detection is necessary for various applications that require the detection of more than one object class simultaneously. The importance of detection processing speed is heightened by the need for multiple classifications without sacrificing accuracy.

Detection of single class objects with a specific view is limited by some object detection methods or trained models. Different views, poses, or angle variations cannot affect their ability to detect the same object.

Different aspect ratios and spatial locations are shown for the object in the image. Those smaller objects will not be detected by the system if they are smaller than 5% of the image. Detecting objects that are arranged very closely together, like plates, can be difficult at times.

Every object detection system needs to measure efficiency as an important parameter. While some object detection systems have been developed to be robust and fast in processing, they still need high-efficiency resources such as CPU and GPU. Due to their limited processing resources, micro-systems pose a challenge.

Only objects for which they have been trained can be detected by object detection models. The ball detection model recognizes a ball, but it can also recognize orange as a ball due to its close-in shape. The ball object's absence during training is the main reason for the detection failure.

III.4.4 Our Application:

Our research focused on creating an object detection system to detect and manage forest fires. We used a deep learning approach to create a Yolov5-based model to detect flames and human figures in forest environments. This paper describes the development

process, from data preparation to model training, deployment, and evaluation.

Prior to data preprocessing, we carefully adjusted the image size and normalized the pixel values to make sure that the input to the model was consistent. We trained the model using the robust architecture of Yolov5, which is well-known for its ability to capture complex features. Python was used as the language of programming, and Roboflow was used for image labeling.

Our first task was to create a custom model for detecting objects. Significant computational demands were revealed by the training process, such as a large dataset and long training epochs. The requirement for uninterrupted internet connection was compounded by the ephemeral nature of Google Colab sessions, usually limited to less than six hours. It was possible to use a high-performance computing environment inside a local Visual Studio Code (VSCode) instance, but the resources were not that good. due to these constraints, we strategically changed our approach to be more efficient . We were able to achieve a confidence level of 96% by using a pre-trained YOLOv8 model. Our application has a robust foundation while training time and computational resource requirements were significantly reduced by this strategic shift.

Our success has allowed us to develop two practical applications. The first code does is for detecting flames in a pre-recorded video. The second code is designed to detect live flames, which enables real-time flame identification.

Deep learning, environmental science, and hardware deployment are interconnected in the development of technological solutions for protecting natural ecosystems, as demonstrated in this research.

III.5 Image Classification:

With so many uses, image classification is an essential part of computer vision tasks. Conventional techniques for classifying images entail the extraction of features and subsequent classification within feature space. The state-of-the-art techniques used today involve feature extraction and classification integrated into the model through end-to-end learning with deep neural networks. It is crucial to comprehend traditional image classification because many of its design principles are similar to those found in neural network components. This understanding can aid in clarifying the sometimes cryptic behavior of these networks.

III.5.1 Image Classification Techniques:

III.5.1.1 Supervised Learning Techniques:

- Support Vector Machine: are supervised algorithms used for both regression and classification The classes are separated by a linear boundary when used for classification purposes.

- Decision Trees: it's a supervised machine learning algorithm that relies only on the tree data structure and a few if/else statements depending on the features that have been chosen.

- Random Forests: A collection of decision trees that minimizes overfitting and increases accuracy.

III.5.1.2 Unsupervised Learning Techniques:

- K-Means Clustering: An unsupervised machine learning technique divides the unlabeled dataset is grouped into different clusters based on their similarity.

- Hierarchical Clustering: is a connectivity-based clustering model that groups the data points together that are close to each other based on the measure of similarity or distance.

- Gaussian Mixture Models (GMM): are a type of probabilistic model used for clustering data.

III.5.1.3 Deep Learning Architectures:

- Convolutional Neural Networks (CNN):is a type of deep learning neural network architecture commonly used in computer vision.

- Transfer Learning: is a machine learning technique in which a model created for one task is applied to another task as the foundation for a new model.

- Recurrent Neural Networks (RNNs): is a special type of artificial neural network designed to handle time-series data or data involving sequential events.

III.5.1.4 Feature Extraction Architectures:

– Histogram of Oriented Gradients (HOG): is a feature descriptor used in computer vision and image processing for object detection

– Local Binary Patterns (LBP): are a powerful visual descriptor used for classification in computer vision. It is useful for texture analysis.

– Color Histograms:are graphical representations of the distribution of colors within an image.

III.5.2 Implementing Image Classification model using deep learning:

- **Data collection :**

Make sure to collect a diverse collection of labeled images as the quality and variety of the data have a significant impact on the performance of the model.

- **Data Processing :**

Make sure that images are resized to a uniform size, standardize pixel values, and consider expanding the dataset. Once that is finished, split the data into training, validation, and test sets.

- **Model Architecture:**

Provide a description of the architecture of your neural network. Convolutional Neural Networks (CNNs) are commonly employed for image classification. Specify the layers, activation functions, and any hyper parameters that need to be identified.

- **Training the Model:**

Train the model with the training data, utilizing an appropriate loss function (such as cross-entropy) and an optimizer (like Adam). Monitor the training progress and adjust hyper parameters.

- **Model Evaluation:**

Evaluate the model's performance with the validation set, taking into account metrics like accuracy, precision, recall, and F1-score. Make adjustments to your model according to the evaluation results.

- **Model deployment:**

After training and evaluating the model, the model will be deployed to the raspberry pi 4,where it can be used for image classification by importing images and creating bounding boxes around objects in the image based on the output.

III.5.3 Benefits and Limitations:

- **Benefits:**

Automation And Efficiency, Image classification simplifies the categorization of images, reducing the requirement for manual analysis. Automating this task for large datasets saves time and effort, improving efficiency in areas like object identification, anomaly detection, and content.

Scalability and Consistence Performance, After receiving training, an image classification model is capable of handling new images quickly and without the need for human intervention. Scalability is essential for applications such as content moderation on social media platforms, where millions of images are uploaded daily. Reliable results across diverse dataset can be assured through consistent performance.

Transfer Learning and Pre-trained models, Transfer learning allows the use of pre-trained models to apply knowledge from one domain, such as natural images, to another domain, like medical images. This approach accelerates model development and decreases the need for labeled data by leveraging.

- **Limitations:**

Data Dependency and Labeled Data Availability, In order to train accurate models, it is necessary to have a lot of labeled data. It can be time-consuming and expensive to acquire diverse and high-quality annotations. Models may have difficulty generalizing if there is not enough data.

Complexity of Image Features, Images contain abundant information, including textures, colors, shapes, and spatial relationships. It can be challenging to accurately classify them by capturing all relevant features. Deep learning models are being trained to understand them.

III.5.4 Our Applications:

The focus of our research project was to create a system that categorizes images and manages plant diseases. We created a custom model using deep learning techniques to categorize plant images based on disease symptoms. The development process is detailed in this paper, from data preparation to model training, deployment, and evaluation.

Compulsory data preprocessing was crucial before tackling model training. To maintain consistency in input for the model, we adjusted image sizes and pixel values.

Capturing intricate features was achieved through our choice of architecture, YOLOv5. Google Colab provided ample computational resources for us to train the model. In addition, we adjusted hyperparameters at a local level through Visual Studio Code (VSCode).

After training the model, the intended method of deployment was to transfer it directly onto a Raspberry Pi 4 that has a camera module. The objective of this configuration is to make it easier to obtain real-time images and monitor plant health on-site. Because of library installation issues on the Raspberry Pi that we didn't anticipate, we had to look into alternative methods, like containerization via Docker and testing different Python versions. despite these significant efforts, The model deployment on the Raspberry Pi 4 was still a challenge .

We created a website that enables users to upload images of plants, aside from the basic model. Results are instantly provided by the app, indicating if the plant is healthy or affected by disease. It is easy for users to assess the health status of their plants and take necessary measures.

In addition, we have put in place live detection capabilities. Our live video streams are processed using the same model, alerting farmers or gardeners to potential disease outbreaks in real time. Timely intervention and crop damage reduction can be achieved through this proactive approach.

We refined hyperparameters iteratively throughout the training process. The model's accuracy and reliability were assessed through regular evaluation using a validation set. We created a separate test set to validate the performance in real life. This test set further validated the model's effectiveness by achieving a good accuracy rate of 87%.

We want to mention that we have trained another model using the CNN technique, and we achieved an accuracy of 94%. Unfortunately, we could not deploy this model on the Raspberry pi 4 because of some errors in the code.

The fields of deep learning, environmental science, and hardware deployment are united in our work. Our contribution to protecting natural ecosystems is through the protection of crops from diseases. Using technology to address critical environmental challenges is demonstrated by the intersection of these fields.

III.6 Challenges:

Several challenges were encountered during the development process that had an impact on project execution. These challenges are discussed in this section and potential solutions to address them are explored.

- **Library Installation :**

Installing essential libraries for image processing on the Raspberry Pi 4 presented challenges for this section of the project. Notably, attempts to install TensorFlow, a popular deep learning framework, resulted in errors such as "couldn't find a version that satisfies the requirement tensorflow" and "No matching distribution found for tensorflow." The errors persist despite experimenting with various Python versions.

- **Training Time Constraints :**

Significant time constraints hindered the training process for the image processing model. The main reason for this was:

Large Dataset Size: The model needed a significant amount of processing time to learn effectively due to the exceptionally large training dataset used.

Insufficient Hardware Resources: Even though the PC has potential, it may not have the necessary processing power and memory for efficient training of large datasets. This resulted in extended training times, hindering project progress.

In order to overcome these limitations, Google Colab, a platform that is hosted on the cloud and provides high-performance computing resources, was implemented. The training process was significantly accelerated by Google Colab, proving it is a viable alternative. Recognizing the limitations of Google Colab is crucial. To be able to use it successfully, it is crucial to have a stable and high-speed internet connection. Google Colab requires session time limits, which could result in frequent reconnections during extended training sessions.

III.7 Conclusion:

Through our examination of image processing in this chapter, we have identified its essential importance as the foundation for advancement in computer vision. Our research has explored the fundamental principles that govern this field, exploring its practical implementations across various sectors, and demonstrating real-life instances that demonstrate its transformative impact. Our attention has been drawn to the potential of object detection in forest fire prevention through flame detection, as well as the efficacy of live plant disease classification utilizing image classification algorithms.

Nonetheless, there have been numerous obstacles to fully utilizing the capabilities of image processing. As evidenced during our project development, obstacles such as compatibility issues with library installations on specific hardware platforms (e.g., Raspberry Pi 4) and inadequate hardware resources for training extensive datasets can significantly impede progress.

Although there are challenges, the progress in cloud-based computing platforms and continuous optimization techniques provide promising solutions. Efficiencies and effectiveness can be increased in image processing applications by recognizing these constraints and exploring alternative strategies, like reducing dataset sizes or leveraging cloud resources. The ultimate goal is to unlock the full potential of this powerful technology.

Chapter IV:
Tests and implementation

IV.1 Introduction:

In this chapter we explore the prototypes made for the delivery and plants diseases treatment. Next we begin with the project relied on an Arduino microcontroller, which was chosen for its ease of programming and its wide use in educational and prototyping purposes. We begin with the tests of the components. Next the initial setup explaining how we combined Arduino with motors, sensors, and electronic speed controllers, and how we managed to overcome the challenges in making a basic flight control using custom-developed software algorithms.

As our project's needs grew to a larger stability and more autonomy, we switched to the ArduPilot Mega (APM). This flight controller came with significant advancements including sensor integration, and capability of doing complex missions autonomously. The transition process, from Arduino to APM, proved to be quite challenging, with new technical issues cropping up in some areas such as software configuration and system calibration.

IV.2 Drone attachments:

IV.2.1 Medical Box Prototype:

The concept of this prototype was to create a 3D printed medical delivery box that could be integrated with a drone to transport critical medical supplies. The delivery of medical aid in remote or disaster-stricken areas could be transformed by this system.

Figure 44: Deleuvry box prototype

IV.2.2 Agriculture Prototype

The objective of this prototype is to develop a 3D printed attachment for a drone that can be used to apply pesticides to plants. Optimizing pesticide application can be achieved through precision agriculture techniques that utilize drones and image processing. To identify specific types of plants or areas that require pesticide treatment, image

processing could be utilized. By equipping the drone with the 3D printed attachment, it can target pesticide application accordingly, minimizing waste and environmental impact.

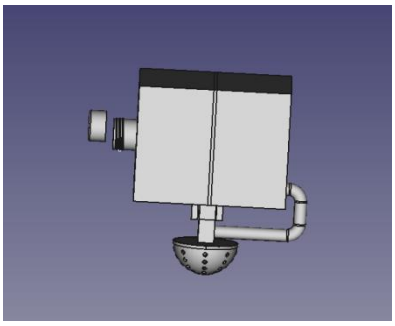


Figure 45: Agriculture prototype

IV.3 Quad Implementation:

IV.3.1 Org chart of the system:

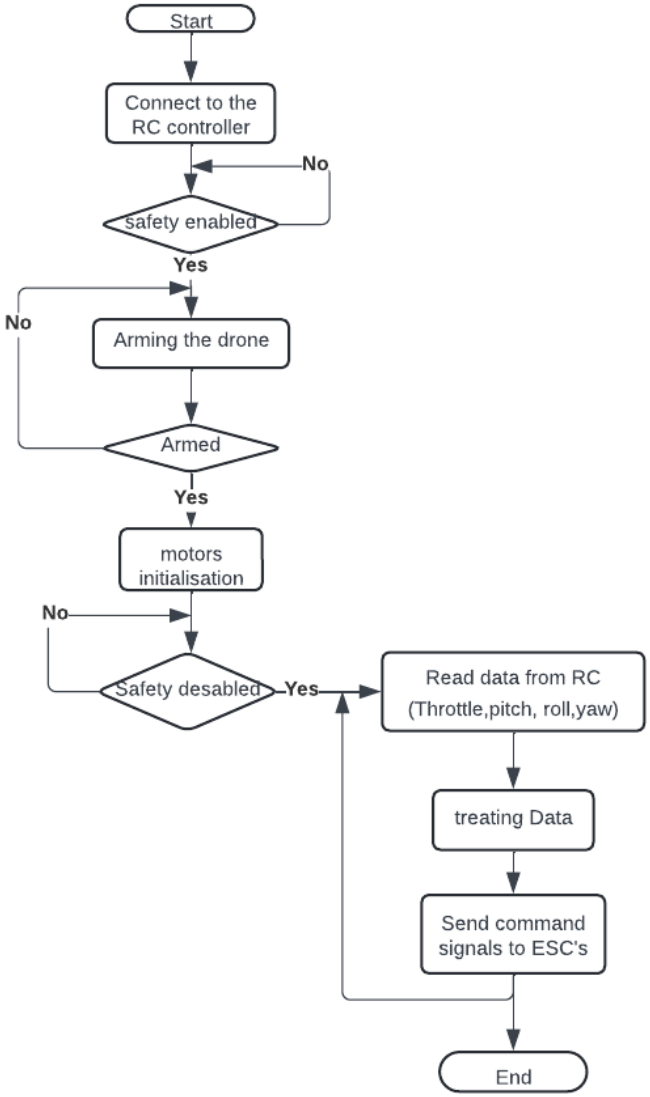


Figure 46: Org chart of the system

IV.3.2 Design and implementation Arduino based:

Initially, we focused on independently testing the motors, sensors—including the MPU (Motion Processing Unit) used for the drone calibration and the GPS for navigation—and the FlySky RC controller, which would allow for manual control inputs. Each component was tested to verify its operational integrity and to familiarize ourselves with its handling and response characteristics.

IV.3.2.1 MPU6050 test:

The MPU, a vital sensor in our drone's flight control system, measures acceleration and rotational motions, enabling us to determine the drone's orientation and movement in real-time. Given its central role in achieving flight stability and navigation, thorough testing of the MPU was essential.

We began by setting up a controlled environment where we could simulate different flight conditions. The MPU was mounted on a secure platform that could be manually tilted and shaken to mimic movements a drone might experience during flight. This setup allowed us to monitor the MPU’s outputs for consistency and responsiveness to movement.

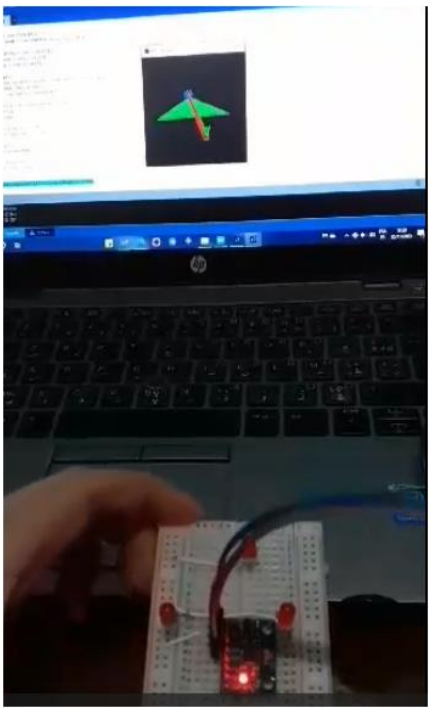


Figure 47: MPU testing result

Throughout the testing process, the MPU’s outputs were continuously recorded and analyzed. We looked for precise measurements of angular velocities and acceleration,

which are crucial for accurate flight control. The data collected helped us understand how the MPU would behave in real flight scenarios, and allowed us to fine-tune our flight control algorithms accordingly.

One challenge we faced was noise in the MPU data, which could lead to erratic flight behavior if not properly managed. To address this, we implemented the complementary filter in our software to smooth out the data, enhancing the accuracy of the information fed into the PID controller.

IV.3.2.2 Complementary Filter:

Indeed, the output of the MPU6050 is used to determine the acceleration in the three major axes X, Y, and Z, as well as the rotation velocities along the same axes for a precise control of the quadcopter. The control system requires a value of tilting, not the rotational speed. This can be done by integrating on the angular velocity to get the angle through which the object has moved with respect to the three axes.

All sensors contain an error, although the error is very small, but the error is present. This error in measuring of the speed leads to an error in the calculation of the angle resulting a cumulative error, because the present angle depends on the previous angle as a starting point. This accumulated error results in huge errors over time and so the plane becomes unstable.

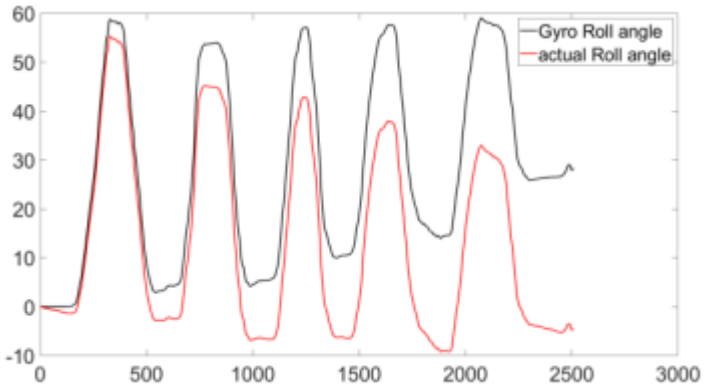


Figure 48: real and measured roll angle out of gyro

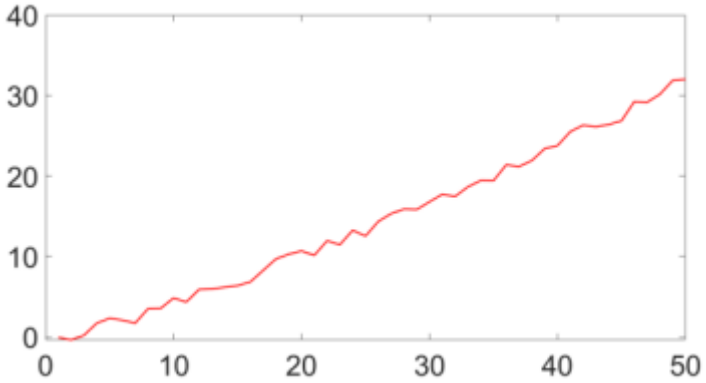


Figure 49: Gyro error signal

To solve the problem we use the complementary filter. For the gyroscope the large percentage of the data is generated while for the accelerometer it is the small percentage of the data. Since the data of the gyroscope contains a cumulative error, a small percentage of the accelerometer data eliminating that cumulative error (48).

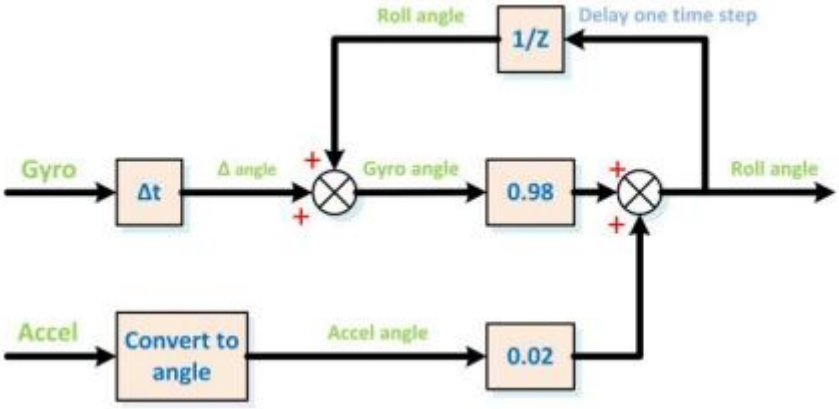


Figure 50: Complementary filter used

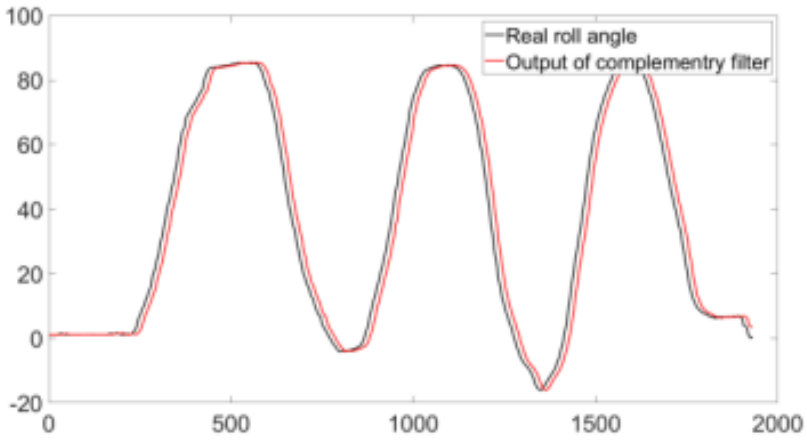


Figure 51: Real and measured roll angle out of optimized complementary filter

IV.3.2.3 GPS test:

The GPS (Global Positioning System) module is a pivotal component of our quadcopter, tasked with providing precise location data, crucial for advanced flight features such as waypoint navigation and return-to-home functionalities. Given its importance, comprehensive testing was conducted to validate its performance and reliability.

The primary objective was to ensure the GPS module could consistently provide accurate location data under various environmental conditions. Accuracy in GPS data is vital for executing precise maneuvers and maintaining the intended flight path of the drone.

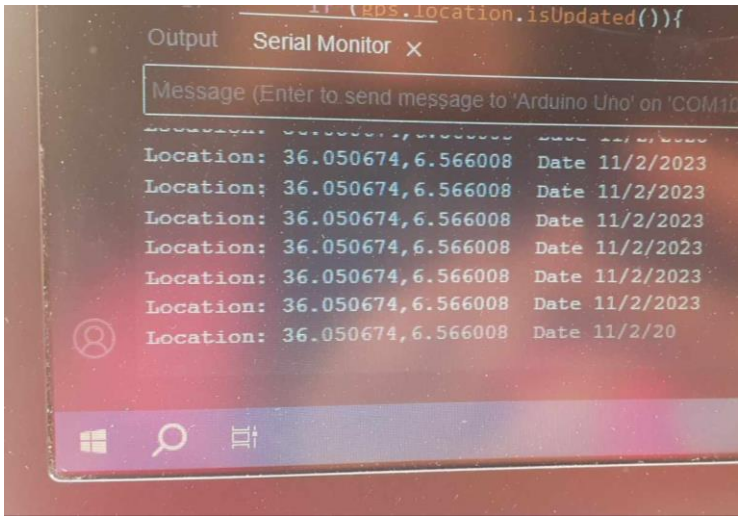


Figure 52: GPS testing results

The GPS module was tested while stationary to check the accuracy and stability of its location readings. This test was conducted in multiple locations to assess the impact of different environmental factors such as obstructions and interference. Data was collected continuously to assess the precision and drift of the GPS coordinates. Special attention was given to the time it took for the GPS to acquire satellites and stabilize its readings after power-up, a critical factor in operational scenarios.

IV.3.2.4 ESCs Calibration:

The Electronic Speed Controllers (ESCs) play a vital role in our quadcopter's performance by controlling the speed and direction of the motors based on the flight controller's commands. Proper calibration of the ESCs is essential to ensure that they interpret and respond to these commands precisely.

The main objective of ESC calibration is to standardize the response of all ESCs in the system so that each motor reacts uniformly to control inputs. This uniformity is critical for stable and balanced flight, as any discrepancy in motor speed can cause the drone to behave unpredictably.

The calibration process involved several key steps:

- **Initialization:** We started by ensuring all ESCs were in their factory default settings to begin with a clean slate.
- **Throttle Range Configuration:** Using our RC controller, we set the maximum and minimum throttle limits. This step is crucial as it defines the range within which the ESCs operate.
- **Manual Testing:** Each ESC was tested individually by manually adjusting the throttle from the lowest to the highest point and observing the motor responses. This was done to ensure that each motor starts and stops smoothly without any abrupt changes in speed.
- **Synchronization:** After individual testing, all ESCs were connected to the flight controller and tested together to ensure synchronized responses. This involved running them simultaneously and adjusting the throttle settings to confirm that all motors responded uniformly at every throttle point.

IV.3.2.5 Assembly and system integration:

Having calibrated and tested each component individually, we moved onto the final and critical phase of integration and assembly. This phase was about combining all individually tested parts into a unified and functional drone system.

First, we strategically placed each component within the drone's chassis, focusing on achieving optimal balance and reducing electronic interference. Secondly, we connected the ESCs to their respective motors and linked all components to the flight controller, with a focus on securing connections and shielding wires to prevent electromagnetic interference.

Once the physical setup of our drone was complete, we proceeded with the crucial task of integrating and configuring the software that controls its operations. This involved

uploading the meticulously developed firmware to the Arduino flight controller.

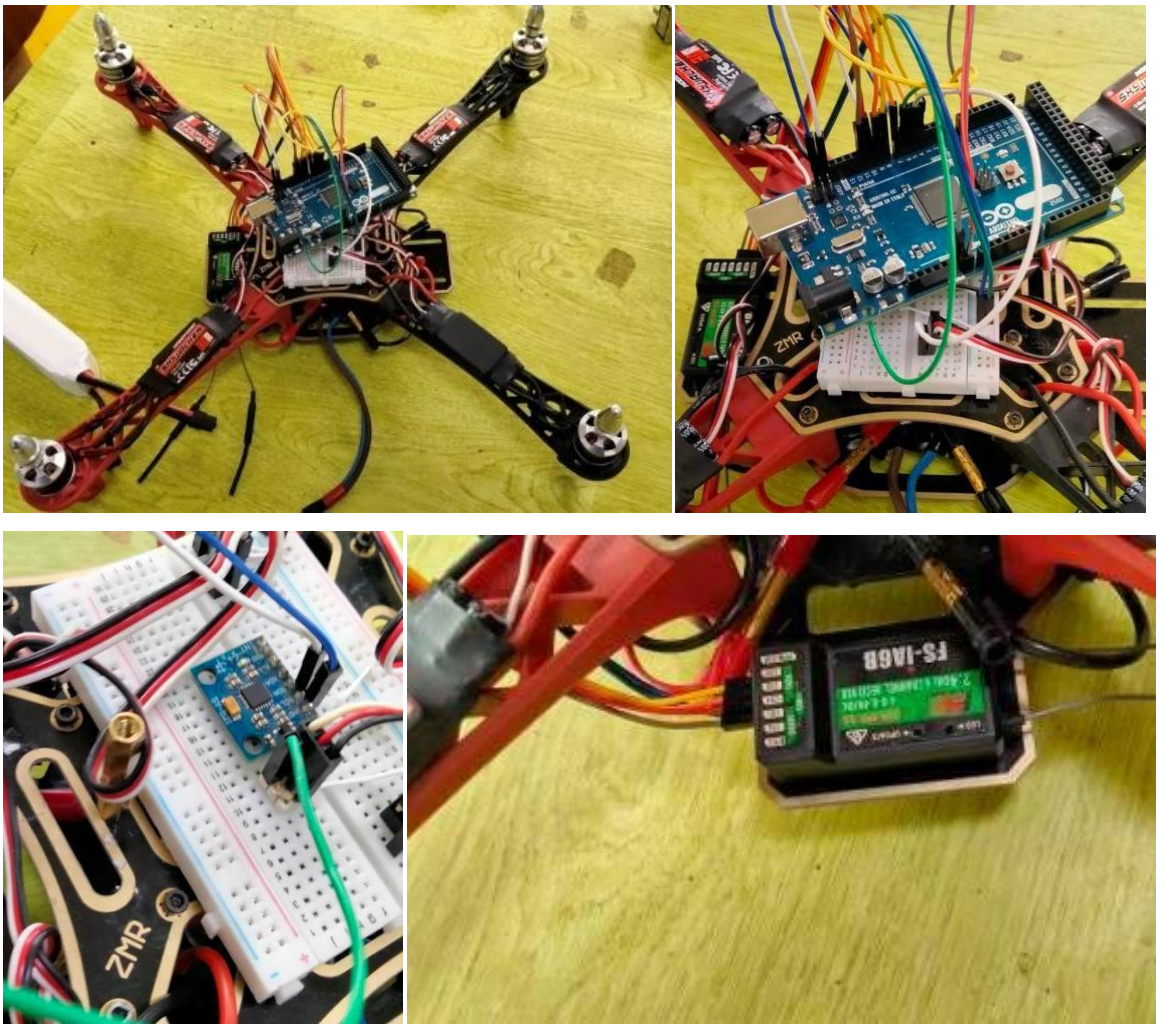


Figure 53: Assembling

IV.3.2.6 PID controller:

Building on the theoretical background and design considerations discussed in chapter I, we will explore the implementation and the c of the PID controller.

The implementation of a PID controller in a digital system requires discretizing the continuous-time controller. The discrete version of the PID is well-known and easy to implement. Regular sampling is crucial because the integral and derivative terms depend on consistent time intervals.

To approximate the integral, we sum the errors over time, and for the derivative, we compute the difference between errors. The sampling time must be much faster than the system's settling time to ensure the controller functions effectively.(for the code see the Appendix).

IV.3.2.7 Arduino based Quad limitations:

The Arduino, while excellent for basic control tasks and initial learning, struggled with the processing demands required for advanced flight dynamics. Its limited computational capacity hindered our ability to implement more sophisticated algorithms for stabilization, obstacle avoidance, and autonomous path finding, which are essential for more complex operations. Despite our efforts to fine-tune the drone's PID controller and optimize the software, we found it challenging to achieve the desired level of stability with the Arduino-based system. The drone's performance fell short of our expectations.

In light of these limitations and our project's evolving requirements, we made the strategic decision to transition to the ArduPilot Mega (APM). The APM platform offered advanced features, greater computational power, and a proven track record in the field of unmanned aerial vehicles (UAVs), positioning our project for success in realizing our vision of a stable, autonomous and versatile quadcopter system.

IV.3.3 Upgrade to ArduPilot Mega (APM):

This upgrade enabled us to remove the external MPU6050 sensor and utilize the APM's built-in IMU, enhancing both the simplicity and precision of our drone's flight control system. Additionally, the APM's integrated compass and altitude sensors have markedly improved the drone's navigational accuracy and stability.

IV.3.3.1 Software configuration using Mission Planner:

After installing the mission planner software, we Start by connecting the ArduPilot Mega (APM) to your computer using a USB cable or telemetry radio. Launch Mission Planner, select the appropriate COM port and baud rate, and establish communication with the APM. Utilize the Setup Wizard in Mission Planner to configure basic settings such as the type of frame (e.g., quadcopter).

Figure 54: Missiong planner interface

- **Firmware Installation:**

This step is crucial for keeping the flight controller up-to-date with the latest improvements and features. Follow the on-screen prompts in Mission Planner to complete the firmware installation, ensuring your APM runs the most current software version.

- **Sensors Calibration:**

Accurate sensor calibration is essential for stable and reliable flight. Start with the accelerometer calibration using Mission Planner to ensure the drone correctly perceives its orientation relative to the earth, crucial for maintaining stability. Next, calibrate the compass to minimize magnetic interference and enhance navigational accuracy; this involves following guided instructions to rotate the drone in various orientations.

- **Radio Calibration**:

Configure the communication link between your radio transmitter and the APM. In Mission Planner, follow the calibration routine by moving the transmitter's sticks and switches to their full ranges. This step defines the operational range for each channel, ensuring that the flight controller accurately interprets pilot commands.

- **ESC Calibration:**

Calibrate the electronic speed controllers (ESCs) using Mission Planner to synchronize motor responses at different throttle levels. This uniformity is essential for balanced and stable drone behavior, particularly during takeoff and maneuvers.

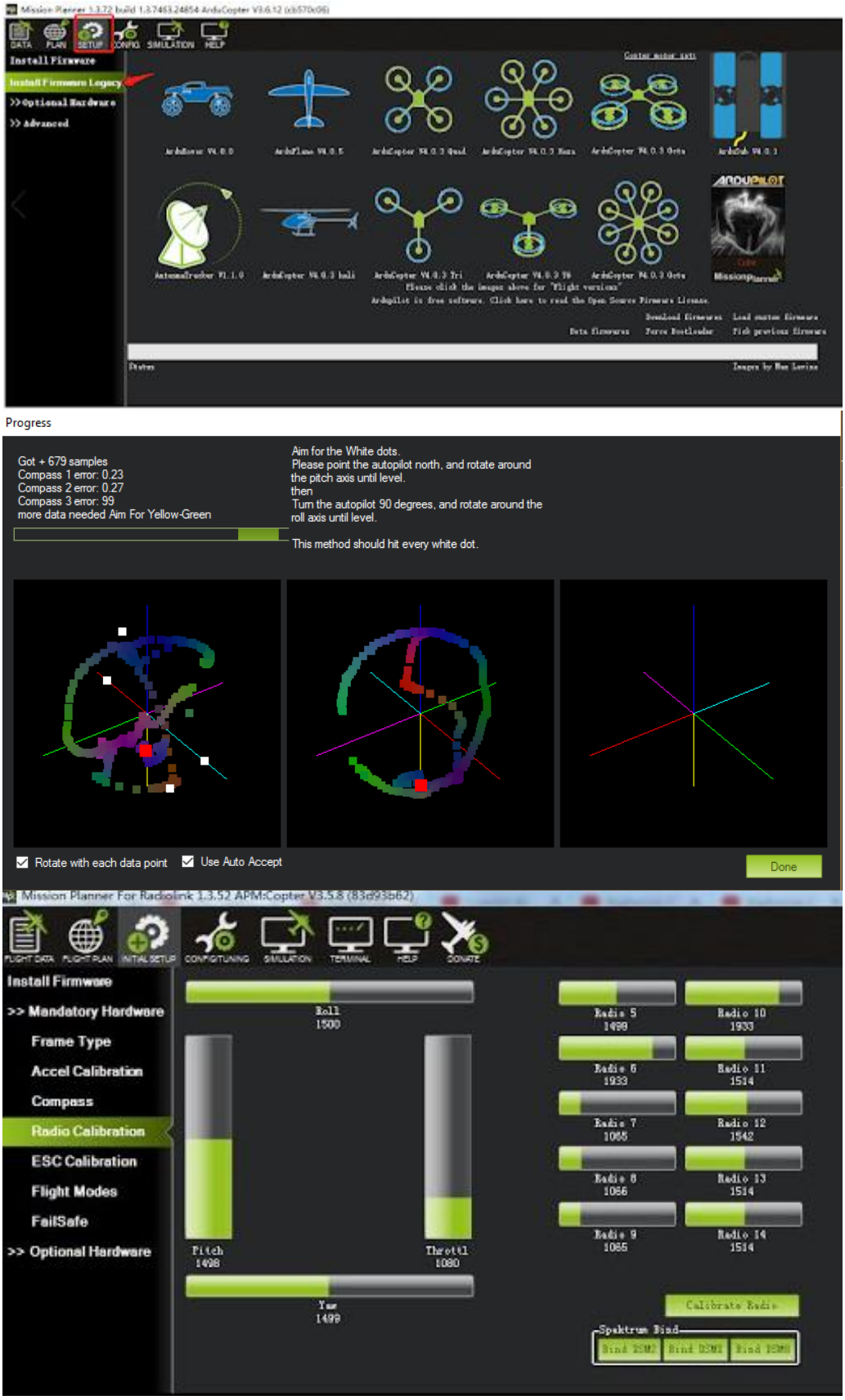


Figure 55: Mission planner configurations

IV.3.4 Testing:

Once the ArduPilot (APM) was configured and all safety measures were set using Mission Planner, we initiated a streamlined testing phase to ensure everything worked seamlessly. This began with ground tests to verify all connections and basic functions. Next, we conducted hover tests to assess the drone's stabilization and altitude hold capabilities, making necessary adjustments to the control settings.

Following successful hover tests, we moved on to controlled flight tests, where the drone was tested in various flight modes to evaluate its performance, navigation accuracy, and responsiveness.

Throughout these tests, we collected and analyzed data on flight behavior and system performance, leading to final adjustments. This comprehensive testing ensured the drone was ready for practical deployment, meeting all operational standards for safety and functionality.

Figure 56: Testing Results of the QuadCopter

IV.3.5 Bluetooth Telemetry radio:

Bluetooth telemetry radio enables real-time data transmission between the quadcopter and the ground control station. This system allows for continuous monitoring of the quadcopter's performance and environmental conditions, enhancing both operational control and safety. It is composed of a Bluetooth module HC-05(zs-040) connected to the flight controller. This setup allows for seamless wireless data exchange. The telemetry system transmits various flight parameters such as altitude, speed, orientation, battery status, and sensor readings to the ground control station.

IV.3.5.1 Connecting to the Ardupilot:

- **Hardware Setup:**

Connect HC-05 to ArduPilot's UART port:

VCC: 5V.

GND: Ground.

TXD: ArduPilot RXD.

RXD: ArduPilot TXD.

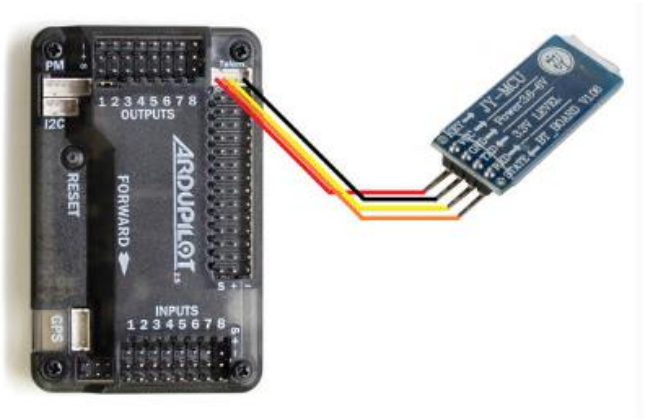


Figure 57: Ardupilot hc-05 connection

- **Configurations and ground station setup:**

We begin with configuring the Bluetooth module. Enter AT mode on the HC-05 and set the baud rate to 57600 (AT+UART=57600,0,0). Next, we configured the ArduPilot in Mission Planner by setting SERIALx_BAUD to 57600 and SERIALx_PROTOCOL to MAVLink (1 or 2). Pair the HC-05 with your ground station device, select the Bluetooth COM port in Mission Planner, set the baud rate to 57600, and connect (for more information visit ardupilot web site (49)).

IV.3.5.2 Mission planning:

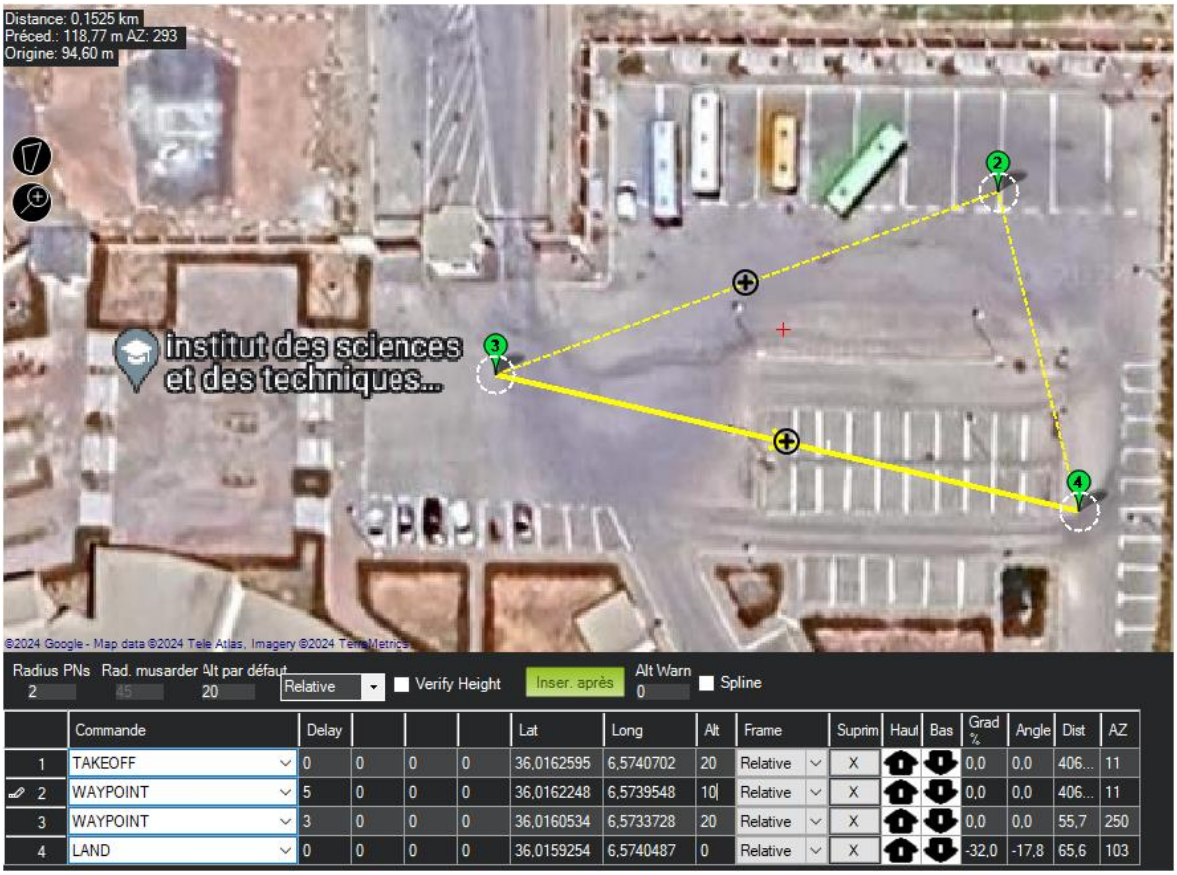
This process involves setting waypoints, defining flight parameters, and ensuring all safety protocols are in place. Effective mission planning allows the quadcopter to perform complex delivery tasks, navigating accurately to drop-off locations and returning to the base reliably, enhancing the overall efficiency and safety of the delivery system.

- **Setting the home position:**

In our project, the home position is set as the location where the vehicle is armed. This ensures that if a Return to Launch (RTL) command is executed, the quadcopter will return to the precise location where it was initially armed. To ensure a safe and accurate return, it's crucial to arm the vehicle at the desired return location. Alternatively, you can use a rally point to establish a different return point if the initial arming location is not suitable. This flexibility in setting the home position enhances the reliability and safety of delivery missions, ensuring the quadcopter returns to a known and secure location.

- **Multi-waypoint Mission:**

The quadcopter's mission starts with an automatic takeoff to an altitude of 20 meters. It then proceeds to Waypoint 2, descending to 10 meters altitude along the way. Upon reaching Waypoint 2, it hovers for 5 seconds. Next, the drone ascends to 20 meters while heading to Waypoint 3, where it waits for 3 seconds. Finally, it continues to Waypoint 4 and lands.



Radius PNs 2 | Rad. musarder 45 | Alt par défaut 20 | Relative | Verify Height | Inser. après | Alt Warn 0 | Spline

	Commande	Delay				Lat	Long	Alt	Frame	Suprim	Haut	Bas	Grad %	Angle	Dist	AZ
1	TAKEOFF	0	0	0	0	36,0162595	6,5740702	20	Relative	X			0,0	0,0	406...	11
2	WAYPOINT	5	0	0	0	36,0162248	6,5739548	10	Relative	X			0,0	0,0	406...	11
3	WAYPOINT	3	0	0	0	36,0160534	6,5733728	20	Relative	X			0,0	0,0	55,7	250
4	LAND	0	0	0	0	36,0159254	6,5740487	0	Relative	X			-32,0	-17,8	65,6	103

Figure 58: mission planning

IV.4 Simulink model simulation:

IV.4.1 Open Loop Model

Motors-Propellers:

In our Simulink model, we opted for a simplified approach for the motors-propellers section, utilizing provided motor constants. This decision was made to streamline the overall model while still maintaining accuracy.

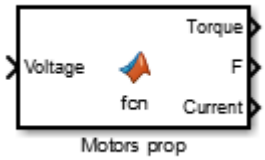


Figure 59: Motors/propellers block Simulink

The Motors/Propellers block in the model is designed to convert the voltage input into three key outputs; Torque, Thrust and Current. The block uses the motor constant KV, the voltage input and the propeller characteristics to calculate the resulting torque, thrust, and the current produced by the motors and propellers couples.

Rotational Dynamics:

The Rotational Dynamics block Receives torque and thrust as inputs from the Motors/Propellers block and outputs the rotational acceleration. The block simulates the quadcopter's angular motion by applying Newton's second law of rotation. The input torque τ directly influences the angular acceleration α, calculated using:

$$\alpha = \frac{\tau}{I}$$

Where 'I' is the moment of inertia of the quadcopter.

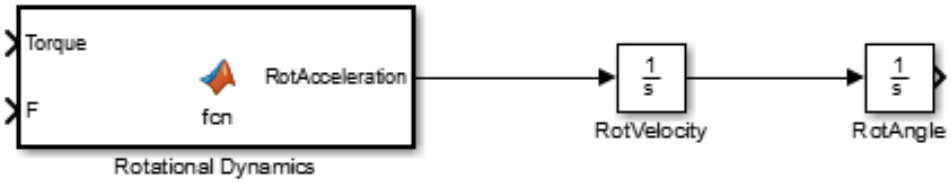


Figure 60: Rotational Dynamics - SIMULINK

Mainly we are intrested about the angle of rotation of our drone. So to get the rotational velocity we integrate the acceleleration, and by integration the velocity we get the rotational angle using the "Integrator Block".

Linear Dynamics:

This block calculates the acceleration in the x, y and z directions of our quadcopter and from this acceleration we obtain the velocity and position. This comprehensive modeling of linear dynamics is essential for predicting the quadcopter's response to control inputs and external disturbances, ensuring accurate simulation of its flight behavior.

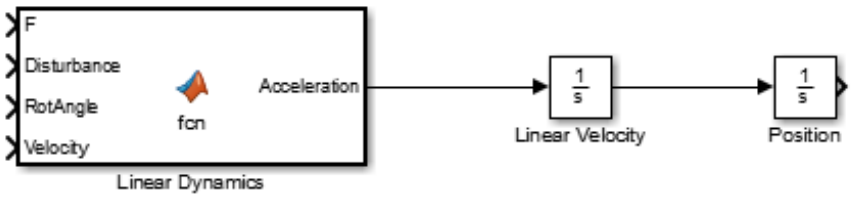


Figure 61: Linear Dynamics - SIMULINK

Disturbance Forces:

Disturbances D, such as wind or other environmental factors, introduce external forces that affect the quadcopter's stability and trajectory. The disturbances block take the Gusts in all directions (x, y and z), and calculate the Drag forces created due to it.

Figure 62: Disturbances - SIMULINK

Control system inputs:

This block converts Throttle, Roll, Pitch, and Yaw inputs to motor voltage for the 4 motors to control its movements. The transformation from control inputs to motor voltages is governed by the following equations that we had explained previously:

U1 = Throttle + Roll + Pitch + Yaw

U2 = Throttle – Roll – Pitch + Yaw

U3 = Throttle – Roll + Pitch – Yaw

U4 = Throttle + Roll – Pitch – Yaw

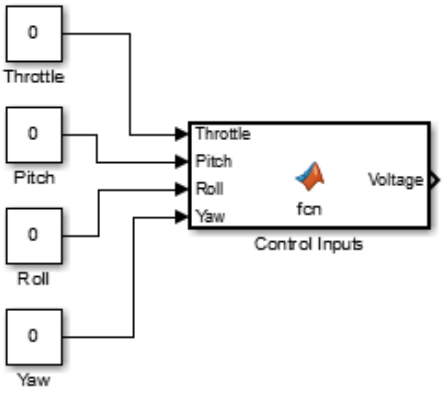


Figure 63: Control Inputs (T, P, R, Y)- SIMULINK

Testing the Open loop drone model:

To visualize and plot values obtained we used some scope blocks, to test our open loop drone model.

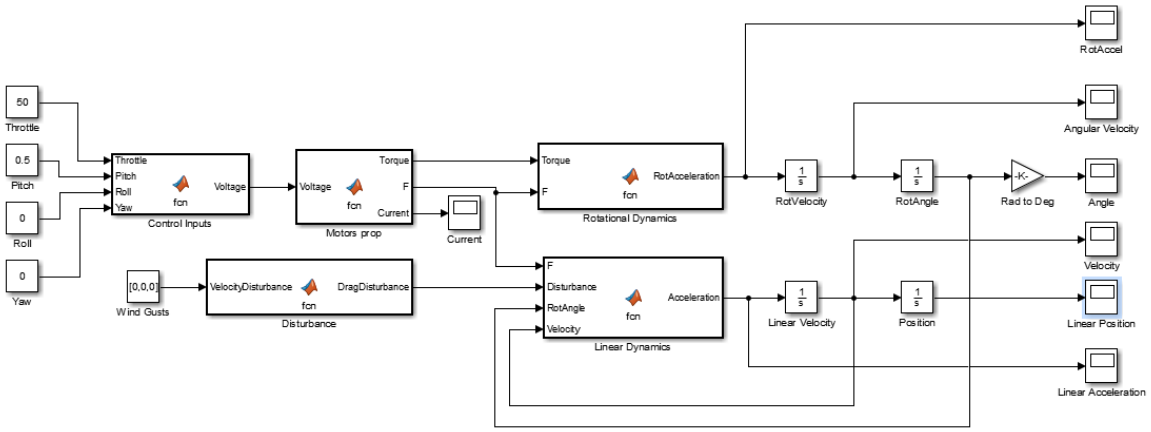


Figure 64: Open Loop Model - SIMULINK

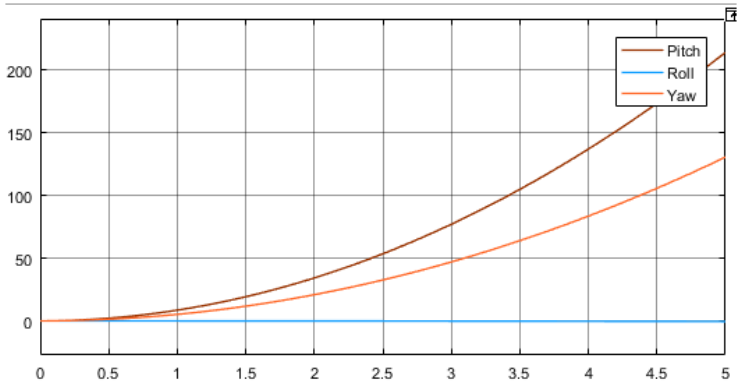


Figure 65: Open loop angles position

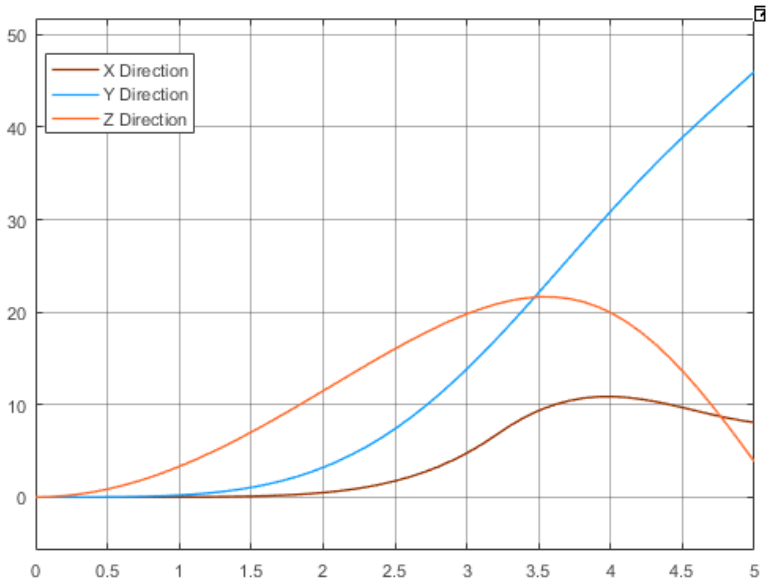


Figure 66: Open loop linear position

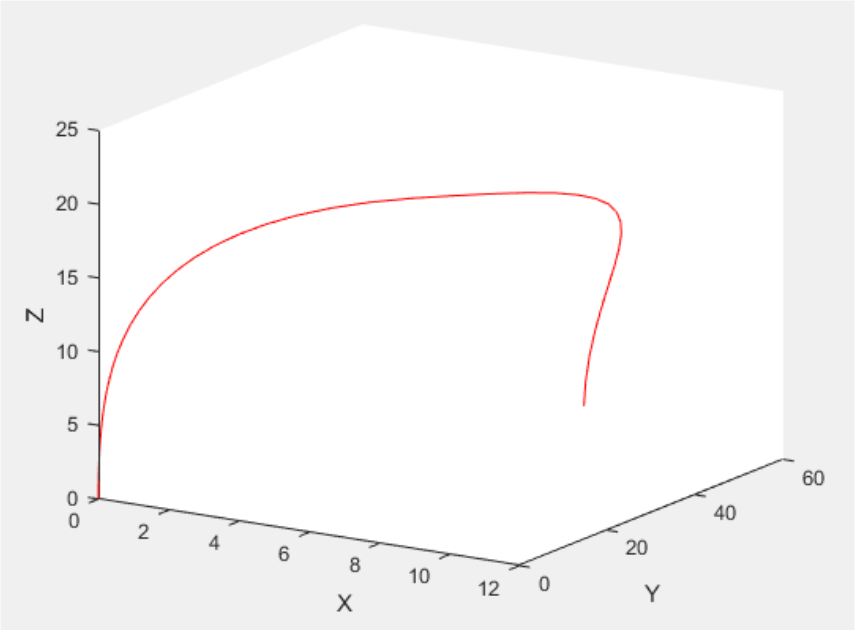


Figure 67: 3D mouvements of the drone (Open Loop Model)

Results are obtained with 50% of Throttle force, 0.5% yaw changing and 0.03% Pitch changing.

IV.4.2 Close Loop Model:

For the closed loop model we implement automated altitude control (PID control algorithm) to our open loop model to tune it closed.

After implementing the PID controller altitude hold and test it we found out that the best gains for it are $K_P = 9$, $K_I = 1$ and $K_D = 9$.

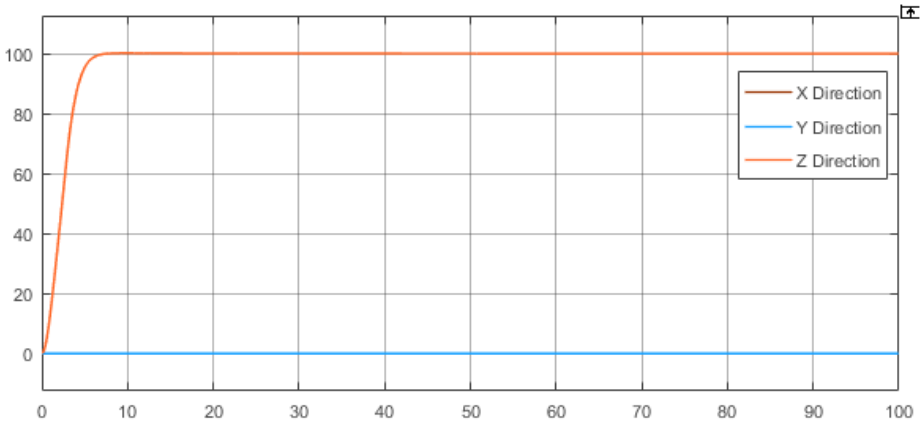


Figure 68: PID controller result

IV.5 Conclusion:

This chapter discusses the initial Arduino-based setup, including component testing, ESC calibration, and system integration. However, this setup presented limitations in terms of stability and advanced functionality, leading to an upgrade to the ArduPilot Mega (APM). The APM provided improved stability and more robust control, which was essential for reliable operation. This upgrade enabled comprehensive testing to ensure system reliability and performance.

Additionally, we explored the integration of Bluetooth telemetry for connecting to the ArduPilot, facilitating effective ground station setup and configuration. Mission planning, which was not feasible with the Arduino-based system, became possible with the APM, allowing for precise control over the quadcopter's flight paths, including setting home positions and creating multi-waypoint missions.

General Conclusion

General Conclusion:

In this thesis, an approach and challenges in designing, implementing, and testing a quadcopter Unmanned Aerial Vehicle (UAV) have been discussed. The research path ranged from the theoretical background and theories to practical application of UAVs and showed how the essential theoretical basis of engineering UAVs is still valid when facing real-world applications.

As with any project, there were some limitations we faced during the course of this project, and they include the following: The first drawback experienced in the application of Bluetooth telemetry was in the line of interconnectivity between the drone and the base station. This limitation posed a major challenge in the ability to monitor and obtain signals from the quadcopter at longer ranges, affecting the overall feasibility of the operations. Furthermore, the battery life for the quadcopter flight operations was limited by the power capacities of the LiPo batteries used. This limitation influenced the longevity and stability of the flight missions we conducted and required more than charging and replacement of batteries frequently.

To overcome these difficulties, the following solutions can be suggested. First, swapping Bluetooth telemetry with radio telemetry systems will be a solution to the range problem, which allows for better control and data transfer in longer distances, which is paramount for most UAV uses. Second, one of the ways of improving the performance of the quadcopter is by increasing the battery capacity so as to increase the operation time of the quadcopter. This will make it possible to incorporate better capacity LiPo batteries into the design of the drone and this will increase the flight time. Moreover, the battery limitations will also be supplemented by installing the wireless magnetic charging stations at different milestone locations. These charging stations would allow the quadcopter to recharge independently while on a mission and thus effectively increase its uptime.

As for the future development, there are several changes that are aimed at the improvement of the quadcopter: An important improvement is the change of the existing Raspberry Pi camera to a better one with higher resolution since this is a very important component that will highly impact the image processing aspect of the quadcopter. Increase in accuracy and reliability will be realized in such areas as identification of plant diseases and detection of flame.

References:

1. Lebedev, Alexander. Design and Implementation of a 6DOF Control System for an Autonomous Quadrocopter. s.l. : Julius Maximilian University of Würzburg, Faculty of Mathematics and Computer Science, 2013.

2. BÜYÜKSARIKULAK, Mehmet S. Autopilot design for a quadcopter. s.l. : The graduate school of natural and applied science of Middle East technical university, 2014.

3. Maza, I., et al. Classification of Multi-UAV Architectures. Handbook of Unmanned Aerial Vehicles. Dordrecht,The Netherlands : Springer , 2015.

4. Experimental investigation on hover performance of a single-rotor system for Mars helicopter. Zhao, P., et al. s.l. : Aerosp. Sci. Technol, 2019, Vol. 86, 582–591.

5. Chinmaya Ranjan Pattnaik, G. Surya Narayana, J. V. R. Ravindra, Sachi Nandan Mohanty, Y. Mohamed Sirajudeen. Drone Technology: Future Trends and Practical Applications. s.l. : John Wiley & Sons, 2023.

6. Mellinger, D., et al. Cooperative Grasping and Transport Using Multiple Quadrotors. Distributed Autonomous Robotic Systems: The 10th International Symposium. Berlin/Heidelberg, Germany : Springer, 2013. Vol. 545–558.

7. Cooperative Transportation Using Small Quadrotors Using Monocular Vision and Inertial Sensing. Loianno, G. et Kumar, V. s.l. : IEEE Robot. Autom. Lett, 2018, Vol. 3, 680–687.

8. A brief overview on miniature fixed-wing unmanned aerial vehicles. Cai, G., et al. Xiamen, China : IEEE ICCA 2010, 2010.

9. Comparison of a Fixed-Wing and Multi-Rotor UAV for Environmental Mapping Applications: A Case Study. Boon, M.A., Drijfhout, A.P. et Tesfamichael, S. s.l. : Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci, 2017.

10. A survey of hybrid Unmanned Aerial Vehicles. Saeed, A.S., et al. s.l. : Prog. Aerosp. Sci, 2018, Vol. 98, 91–105.

11. A Simple Controller of Minisize Quad-Rotor Vehicle. Hongning Hou, Jian Zhuang, Hu Xia, Guanwei Wang, Dehong Yu. s.l. : Mechatronics and Automation (ICMA), 2010.

12. National Aeronautics and Space Administration. Advanced Air Mobility: The Science Behind Quadcopters Reader—Student Guide. 2020.

13. yukai.chen, donkyu.baek, alberto.bocca, alberto.macii, enrico.macii, massimo.poncino. A Case for a Battery-Aware Model of Drone Energy Consumption. 2018. 1-8.

10.1109/INTLEC.2018.8612333.
14. Beard, R. W. Quadrotor dynamics and control. s.l. : Brigham Young University, 2008.
15. Boiffier, J. L. The dynamics of flight the equations. s.l. : John Wiley & Sons, 1998.
16. S. Windnall, J. Peraire. Relative Motion using Rotating Axes. Lecture Notes inDynamics. Boston, USA : MIT, 2008.
17. Newman., J. Seddon and S. Basic helicopter aerodynamics. . s.l. : Blackwell Science, 2002.
18. Wang, Jing. Quadrotor analysis and model free control with comparisons. paris : Université Paris Sud, 2013.
19. A. R. Bramwell, D. Balmford, G. Done. Bramwell's helicopter dynamics. s.l. : Butterworth-heinemann, 2001.
20. Research on attitude controller of quadcopter based on cascade PID control algorithm. Bao, N., Ran, X., Wu, Z., Xue, Y., Wang, K. s.l. : Proceedings of the IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC, 2017.
21. Metaheuristic algorithms for PID controller parameters tuning: review,. SB Joseph, EG Dada, A Abidemi, DO Oyewola, BM Khammas. s.l. : heliyon, 2022.
22. [En ligne] [Citation : 24 04 2024.] https://store.arduino.cc/products/arduino-mega-2560-rev3.
23. [En ligne] [Citation : 24 04 2024.] https://www.datasheethub.com/gy-neo6mv2-flight-control-gps-module/.
24. GO TRONIC. [En ligne] https://www.gotronic.fr/art-module-bluetooth-hc05-26097.htm#:~:text=Ce%20module%20Bluetooth%20HC05%20permet,(communication%20via%20s%C3%A9rie%20TTL).&text=Valable%20pour%20livraison%20en%20France%20M%C3%A9tropolitaine..
25. [En ligne] [Citation : 24 April 2024.] https://himalayansolution.com/product/emax-980kv-brushless-motor.
26. [En ligne] [Citation : 24 04 2024.] https://hobbycomponents.com/rc/487-skywalker-30a-electronic-speed-controller-esc.
27. [En ligne] [Citation : 24 April 2024.] https://www.ardupilot.co.uk/.
28. [En ligne] [Citation : 24 April 2024.] https://mhtronic.com/produit/batterie-lipo-2200mah-3s-11-1v-70c/.
29. [En ligne] 07 2017.

https://static1.squarespace.com/static/5bc852d6b9144934c40d499c/t/5c0787e10e2e721a7f17c998/1543997593953/FS-i6+User+manual+20160819.pdf.
30. Raspberry Pi. [En ligne] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.
31. FreeCAD. [En ligne] https://www.freecad.org/.
32. V7labs. [En ligne] [Citation : 01 june 2024.] https://www.v7labs.com/blog/image-processing-guide.
33. baeldung. The Viola-Jones Algorithm. [En ligne] 18 March 2024. [Citation : 01 June 2024.] https://www.baeldung.com/cs/viola-jones-algorithm.
34. Deformable Part Models are Convolutional Neural Networks. R Girshick, F Iandola, T Darrell, J Malik. 2015.
35. Object detection with discriminatively trained part-based models. Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. 2010.
36. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015.
37. Fast R-CNN. Girshick, Ross. 2015.
38. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015.
39. Kundu, Rohit. [En ligne] 17 January 2023. [Citation : 01 June 2024.] https://www.v7labs.com/blog/yolo-object-detection#what-is-yolo.
40. Focal Loss for Dense Object Detection. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar. 2017.
41. Jaskirat Kaur, Williamjeet Singh. Springer. [En ligne] 23 April 2022. [Citation : 01 June 2024.] https://link.springer.com/article/10.1007/s11042-022-13153-y.
42. Xailient. [En ligne] 26 March 2020. [Citation : 01 June 2024.] https://xailient.com/blog/how-to-annotate-your-images-using-the-makesense-tool/.
43. Nelson, Joseph. robotflow. [En ligne] 16 March 2020. [Citation : 01 June 2024.] https://blog.roboflow.com/labelimg/.
44. Real-Time Ear Detection Based On Embedded Systems. Li Yuan, Fei Lu. Chengdu, China : 2018 International Conference on Machine Learning and Cybernetics (ICMLC), 2018.
45. A survey on Real time Object Detection andTracking Algorithms. Gomathy Nayagam, K.Ramar. 2015.

46. Real-Time Object Detection and Recognition on Low-Compute Humanoid Robots using Deep Learning. Chatterjee, Sayantan, Zunjani, Faheem H. et Nandi, Gora C. Singapore : s.n., 2020.
47. Pedestrian detection: A benchmark. Dollar, Piotr, et al. Miami,FL, USA : s.n., 2009.
48. An Optimized Complementary Filter For An Inertial Measurement Unit Contain MPU6050 Sensor. Ahmed Fahem Albaghdadi, Abduladhem Abdulkareem Ali. 2, s.l. : Iraqi Journal of Electrical and Electronic Engineering , 2019, Vol. 15.
49. ArduPilot. [En ligne] https://ardupilot.org/copter/docs/common-mission-planner-bluetooth-connectivity.html.

Appendix

```
void calculate_pid(){
  //Roll calculations
  pid_error_temp = gyro_roll_input - pid_roll_setpoint;
  pid_i_mem_roll += pid_i_gain_roll * pid_error_temp;
  if(pid_i_mem_roll > pid_max_roll)pid_i_mem_roll = pid_max_roll;
  else if(pid_i_mem_roll < pid_max_roll * -1)pid_i_mem_roll =
pid_max_roll * -1;

  pid_output_roll = pid_p_gain_roll * pid_error_temp +
pid_i_mem_roll + pid_d_gain_roll * (pid_error_temp -
pid_last_roll_d_error);
  if(pid_output_roll > pid_max_roll)pid_output_roll =
pid_max_roll;
  else if(pid_output_roll < pid_max_roll * -1)pid_output_roll =
pid_max_roll * -1;

  pid_last_roll_d_error = pid_error_temp;

  //Pitch calculations
  pid_error_temp = gyro_pitch_input - pid_pitch_setpoint;
  pid_i_mem_pitch += pid_i_gain_pitch * pid_error_temp;
  if(pid_i_mem_pitch > pid_max_pitch)pid_i_mem_pitch =
pid_max_pitch;
  else if(pid_i_mem_pitch < pid_max_pitch * -1)pid_i_mem_pitch =
pid_max_pitch * -1;

  pid_output_pitch = pid_p_gain_pitch * pid_error_temp +
pid_i_mem_pitch + pid_d_gain_pitch * (pid_error_temp -
pid_last_pitch_d_error);
  if(pid_output_pitch > pid_max_pitch)pid_output_pitch =
pid_max_pitch;
  else if(pid_output_pitch < pid_max_pitch * -1)pid_output_pitch =
pid_max_pitch * -1;

  pid_last_pitch_d_error = pid_error_temp;

  //Yaw calculations
  pid_error_temp = gyro_yaw_input - pid_yaw_setpoint;
  pid_i_mem_yaw += pid_i_gain_yaw * pid_error_temp;
  if(pid_i_mem_yaw > pid_max_yaw)pid_i_mem_yaw = pid_max_yaw;
  else if(pid_i_mem_yaw < pid_max_yaw * -1)pid_i_mem_yaw =
pid_max_yaw * -1;

  pid_output_yaw = pid_p_gain_yaw * pid_error_temp + pid_i_mem_yaw
+ pid_d_gain_yaw * (pid_error_temp - pid_last_yaw_d_error);
  if(pid_output_yaw > pid_max_yaw)pid_output_yaw = pid_max_yaw;
```

```
  else if(pid_output_yaw < pid_max_yaw * -1)pid_output_yaw =
pid_max_yaw * -1;

  pid_last_yaw_d_error = pid_error_temp;
}
```

الجمهـــــــــــــورية الجزائـــــــــرية الديمقـــــــراطية الشـــــــعبية
Democratic and Popular Republic of Algeria
وزارة التعليـــــــــــم العـــــــــالى والبحـــــــــــث العلمـــــــى
Ministry of Higher Education and Scientific Research

Larbi Ben M'hidi University O.E.B
Institute of Applied Sciences and Technologies.
Department of Networks and Telecommunications

جامعة العربي بن مهيدي أم البواقي
معهد العلـوم و التقنيات التطبيقيـة
قسـم شبكات و اتصالات سلكية ولا سلكية

Internship report

Presented for the purpose of obtaining the Professional Master's degree in Systems Networks and Telecommunications.

Quad-rotor stabilization

Conducted at:

Research Center in Industrial Technologies
Intelligent Embedded Systems Technological Platform |Bou Ismail

By: **DERGHAL Safa**
DJABALLAH Badr

Directed by: Mr **ZABEL Abd El Ghani (CRTI-PTSEI)**
Dr **MOULAHCENE Fateh (ISTA)**

2023/2024

Content

List of figures

Chapter I:

Introduction

I.1 Introduction:

The primary objective of this internship was to stabilize the flight of a quadcopter and achieve the best possible PID correction. Quadcopter stability is essential for a wide range of applications, from aerial photography to delivery services, and requires precise control mechanisms to maintain balance and respond to external disturbances.

Stabilizing a quadcopter involves maintaining its orientation and position in space despite external forces such as wind or sudden movements. PID control is a widely used method for achieving such stability. It combines proportional, integral, and derivative actions to correct the error between the desired setpoint and the actual state of the quadcopter. By tuning the PID parameters, one can achieve optimal performance and ensure smooth, stable flight. The specific objectives of this report were as follows:

- **Stabilize the Quadcopter's Flight:** Ensure that the quadcopter can maintain its orientation and position in various conditions.
- **Optimize PID Controller:** Develop and fine-tune the PID parameters to achieve the best possible flight performance.
- **Test and Validate:** Conduct a series of flight tests to validate the effectiveness of the tuned PID controller and make further adjustments as necessary.

I.2 Organizational Overview:

The Research Centre in Industrial Technologies was created by the Executive Decree N°. 15-109 of May 3rd, 2015, amending the Executive Decree N°. 92-280 of July 6th, 1992 which launched this institutions as a Scientific and Technical Public Establishment (EPST).

The CRTI, ex-CSC, has a human potential of 750 officials, including 300 permanent researchers, 200 technical staff, and 150 experts and engineers of the subsidiary CSC expertise Spa. Moreover, the centre includes several valorization structures throughout the national territory.

Before acquiring the status of (EPST), the centre has passed through different stages of development, starting with the creation of welding and nondestructive laboratory (LSCND) in 1985 up to the creation of a national Research Centre in 1992. The CRTI, ex-CSC, has an overall responsibility for implementing necessary research programs for the development of welding, destructive and nondestructive testing

technologies. Nearly thirty years later, the mission remains as relevant. However, over the decades, the its missions has developed and enhanced, which has required redefining and changing the name of the centre. This the centre is now called “Research Centre in Industrial Technologies”.

The CRTI is mainly responsible for:

- Carrying out necessary research projects for the development of industrial technologies, including assembly techniques, nondestructive testing, and corrosion.
- Organizing, developing and promoting quality assurance and quality control of industrial plants.
- Developing and contributing to achieving compilations, norms, and standards related to the technologies of assemblies, nondestructive testing, industrial plants and corrosion of metallic materials.
- Improving, verifying and using the welding, nondestructive testing, analysis and measurement equipment.
- Developing applied research in the field of iron and steel, such as in elaboration and characterization of special steels and alloys.
- Mastering and developing mechatronics and applied maintenance to industrial plants.
- Developing research programs in the elaboration, the characterization and the study of nonmetallic materials behavior, such as composites, ceramics, etc…..
- Developing research programs in the technology of materials surface treatment and their applications.

The CRTI is organized into several research poles as it follows:

- The headquarter pole of Cherega, Algiers (CRTI) which is composed of five (05) research divisions.
- Setif pole (UDCMA), located in the industrial zone and specialized in thin films and coating

- Annaba pole (URMA), located at Badji Mokhtar university campus. It is specialized in advanced materials.
- Another pole (URASM), in Annaba located in El Hadjar Complex specialized in Iron and steel.
- Bousmail's pole which is composed of the Subsidiary and the technological platform that are valorization structures.

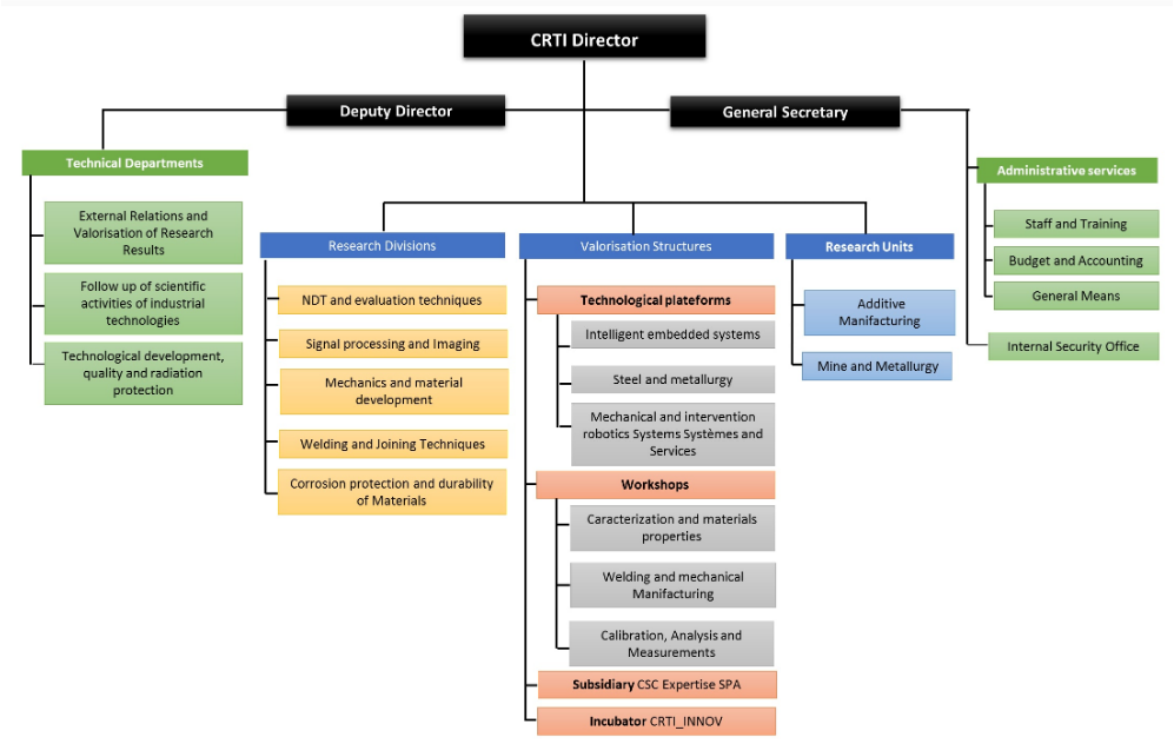


Figure 1: CRTI Organisational Chart

I.3 Intelligent Embedded Systems Platform:

CRTI plays an important position in the creation of science and technology poles, and is heavily involved in their organization and animation. The pole of Bou-Ismail contributes to the creation of a dynamic exchange between the researches structures of the center and the various industries. Therefore it is a consistent set for research, education and technology transfer. As part of its prototyping mission, the platform offers a structuring project whose socio-economic impact is of great importance.

Chapter II:
PID Controller

II.1 PID Controller:

PID controller is used to attenuate the error produced from the difference between the desired state and the actual state. It is a fundamental component in the control system of quadcopters, playing a critical role in ensuring stable and responsive flight.

The PID controller algorithm involves three separate constant parameters; Proportional (P), Integral (I), Derivative (D). These values can be interpreted in terms of time: 'P' denotes the current error, 'I' stands for the sum of past errors, and 'D' is the prediction of future errors, which is achieved by seeing the changing rate, at the moment.

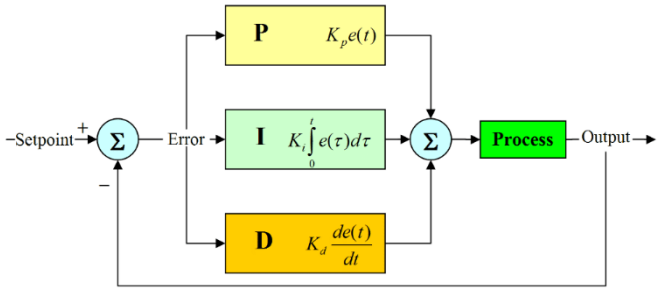


Figure 2: PID controller structure

II.1.1 Proportional term:

The proportional term creates an output value that is proportional to the current error value. In the simplest way, the more the error the more the output value and therefore the closer the setpoint is achieved (but beware of overshoot and instability). Proportional response can be adjusted by multiplying the error by a constant Kp, which is known as the proportional gain.

The proportional term is given by:

$$P_{Out} = K_P\, e(t)$$

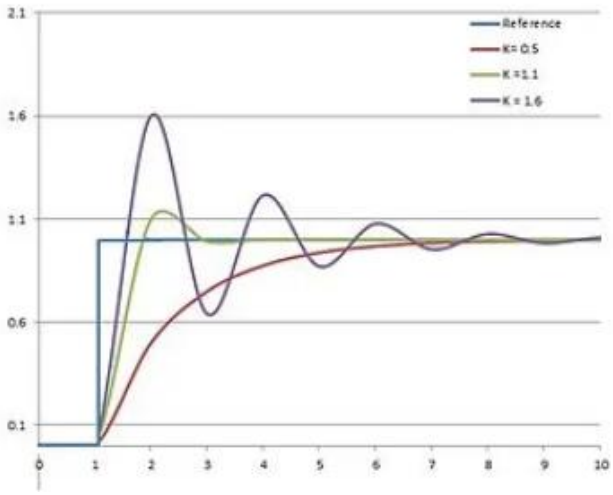


Figure 3: K_P influence in a PID Control system (K_d, K_i constants)

II.1.2 Integral term:

The contribution from the integral term is then inversely proportional to both the magnitude of the error and the error duration. PID controller has an integral part, which is the sum of the instantaneous error over the time and it gives the accumulated offset (steady-state error) that should have been corrected previously. The accumulated error is then multiplied by the integral gain (Ki) and added to the controller output. This parameter removes offset as it rises the nature and order of the system by 1. This parameter also rises the system reaction speed, but at the cost of continued oscillations.

$$I_{Out} = K_i \int_0^t e(t)\, dt$$

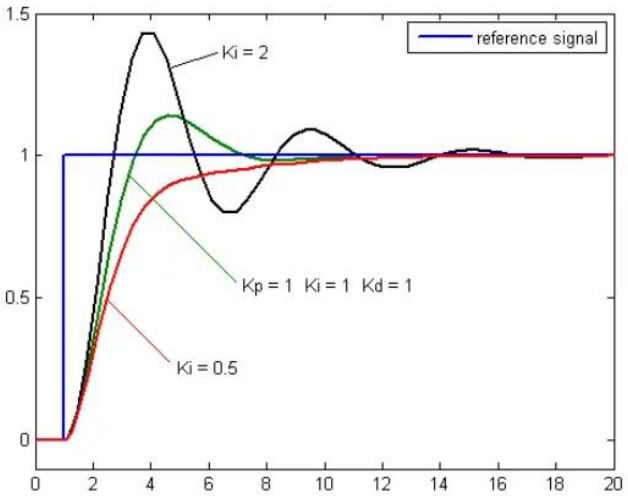


Figure 4: Ki Influence in a PID control system (KP and KD constants)

II.1.3 Derivative term:

In essence, this setting reduces the system's oscillatory response. It has no influence on the offset and it has no influence on the system's nature and order. It examines how quickly the error signal changes. The derivative is the cause of the larger system reaction to the increasing speed of change. Using too much derivative term may give the system a highly oscillatory behavior or even a control overshoot.

$$D_{Out} = K_D \frac{d\, e(t)}{dt}$$

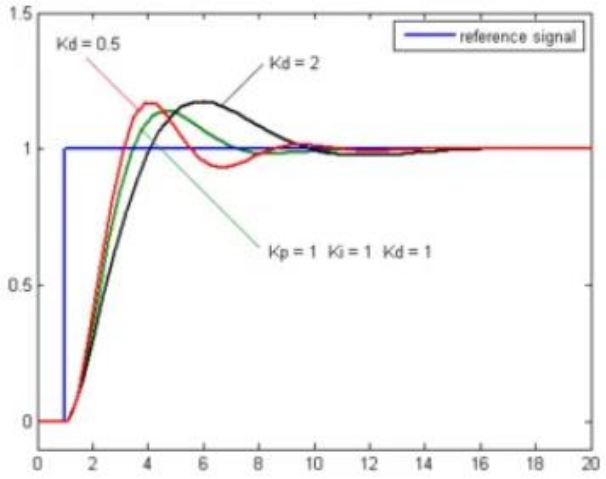


Figure 5: influence in a PID control system (Ki and KP are constants)

A PID controller is a control loop mechanism that continuously calculates an error value $e(t)$ as the difference between a desired setpoint and a measured process variable, and applies a correction based on proportional, integral, and derivative terms. Both altitude and attitude controller use PID controller to give out control inputs U1, U2, U3 and U4. All of these control inputs are the fundamental in producing the outputs z, θ, φ, ψ.

The system contains four PID controllers, so the tuning process will begin with the altitude controller and disconnecting the other three control inputs. The equations below show the PID controller equations to obtain all control inputs U1, U2, U3, and U4 which are used to produce the relative speed Xr. The control inputs of U2, U3 and U4 come from the use of angular error whereas U1 comes from the use of altitude error.

$$U_1(t) = K_p e_z(t) + K_I \int_0^t e_z(t)dt + K_D \frac{e_z(t) - e_z(t-1)}{dt}$$

$$U_2(t) = K_p e_\varphi(t) + K_I \int_0^t e_\varphi(t)dt + K_D \frac{e_\varphi(t) - e_\varphi(t-1)}{dt}$$

$$U_3(t) = K_p e_\theta(t) + K_I \int_0^t e_\theta(t)dt + K_D \frac{e_\theta(t) - e_\theta(t-1)}{dt}$$

$$U_4(t) = K_p e_\psi(t) + K_I \int_0^t e_\psi(t)dt + K_D \frac{e_\psi(t) - e_\psi(t-1)}{dt}$$

Chapter III:
Methodology

III.1 ESC Calibration:

Electronic Speed Controllers (ESCs) are critical components that regulate the speed of the drone's motors. Proper calibration ensures that each motor responds accurately to the control signals, which is essential for maintaining stability. The calibration process involved setting the throttle range for each ESC and verifying that the motors operated synchronously. This step was crucial to ensure that the drone could respond correctly to the PID control inputs.

III.2 PID Planification:

III.2.1 Throttle:

The throttle PID controller was designed to maintain a stable altitude by adjusting the overall thrust of the motors. The proportional term responded to altitude deviations, the integral term corrected cumulative errors, and the derivative term anticipated future changes. This approach allowed for smooth altitude control even in the presence of external disturbances.

III.2.2 Pitch and Roll:

For pitch and roll, the PID controllers were responsible for stabilizing the drone's orientation along the forward-backward and left-right axes. The controllers adjusted the motor speeds to counteract any tilting movements, ensuring that the drone remained level. The tuning process involved setting appropriate gains to balance responsiveness and stability.

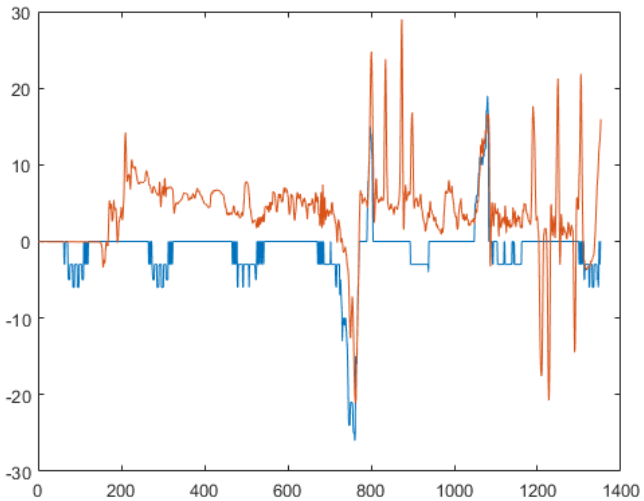


Figure 6: Pitch PID first test result

III.2.3 Yaw:

The yaw PID controller managed the drone's rotation around its vertical axis. By controlling the differential thrust between the motors, the yaw controller kept the drone oriented in the desired direction. Fine-tuning the yaw controller was essential to prevent oscillations and achieve precise heading control.

Using the mz-10 RC controller we defined the 5, 6, 7 channels to facilitate adjusting the K_P, K_i and K_d gain.

Figure 7: PID adjusting with RC

III.3 Code development and testing:

The development and testing of the PID controllers involved a systematic approach to tuning the proportional, integral, and derivative gains.

```
85  float Kp=5;
86  float Ki=0;
87  float Kd=1.5;
88
89  float Kp_Pitch = Kp ;
90  float Ki_Pitch = 0 ;
91  float Kd_Pitch = Kd ;
92
93  float pitch_P;
94  float pitch_i;
95  float pitch_d;
96
97  float pitch_PID;
98  float PID_Max = 50;
99  int pitch_PID_Out ;
```

```
279  ////////////////////////PID PITCH/////////////////////////////////////////////////////
280
281  Kp_Pitch = Kp;
282  Kd_Pitch = Kd;
283  Ki_Pitch  = Ki;
284
285  pitcherror= pitch_gyro-pitch_RC;
286  delta_pitcherror = (pitcherror - previous_pitcherror )/dT;
287  integral_pitcherror += pitcherror*dT;
288
289    pitch_P = Kp_Pitch*pitcherror;
290    pitch_i = Ki_Pitch*integral_pitcherror;
291    pitch_d = Kd_Pitch*delta_pitcherror;
292
293    pitch_PID = pitch_P + pitch_i + pitch_d;
294
295    pitch_PID_Out = (pitch_PID > PID_Max)? PID_Max:pitch_PID;
```

Figure 8: Pitch PID Arduino code

The process was as follows:

III.3.1 Increasing Proportional Gain :

The initial step was to increase the proportional gain to achieve a fast response from the PID controller. The proportional term directly affects how aggressively the system reacts to the error. By incrementing this gain, the controller could respond more quickly to deviations from the desired setpoint.

III.3.2 Adjusting Derivative Gain :

Once an adequate proportional response was achieved, the derivative gain was adjusted to reduce system oscillations. The derivative term predicts future errors based on the rate of change of the error, which helps to dampen the oscillations and stabilize the system. This tuning involved gradually increasing the derivative gain until the oscillations were minimized.

III.3.3 Tuning Integral Gain :

The final step in the tuning process was to adjust the integral gain to smooth out the overall control response. The integral term addresses accumulated errors over time, ensuring that the system can correct any steady-state errors. By carefully increasing the integral gain, the controller achieved the smoothest possible response without inducing additional oscillations or instability.

III.4 Testing and Tuning :

Testing involved flying the quadcopter in a controlled environment and adjusting the PID gains to achieve stable flight. The gains for the proportional, integral, and derivative

terms were incrementally adjusted based on observed flight behavior. This iterative process was repeated until the optimal gains were found for each control axis, resulting in smooth and stable flight.

III.5 Limitations:

III.5.1 Processor limitations:

The Arduino Mega used in the drone as a flight controller had several limitations, including limited processing speed, which affected the responsiveness of the PID controllers. Memory constraints limited the complexity of the control algorithms and the amount of sensor data that could be processed in real-time.

III.5.2 Sensors limitations:

The sensors on the drone, including the MPU6050 (accelerometer, gyroscope) had limitations in terms of accuracy and susceptibility to noise. Environmental factors such as electromagnetic interference and vibrations affected sensor readings, leading to potential instability. Latency in sensor data processing also posed challenges for real-time control.

References:

- Research centre in Industrial Technologies -CRTI- EChahid Mohammed ABASSI (https://www.crti.dz).

- S, DERGHAL. B, DJABALLAH. Designing and Implementation of a multi-purpose UAV: ISTA, LARBI BEN M'HIDI University, 2024.

- Research on attitude controller of quadcopter based on cascade PID control algorithm. Bao, N., Ran, X., Wu, Z., Xue, Y., Wang, K. s.l. : Proceedings of the IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC, 2017.

- Metaheuristic algorithms for PID controller parameters tuning: review,. SB Joseph, EG Dada, A Abidemi, DO Oyewola, BM Khammas. s.l. : heliyon, 2022.