ABSTRACT

<u>PROJECT TITLE:</u> Embedded Control System for Smart Military Drone

Unmanned Aerial Vehicles (UAVs), in particular small multi-rotor UAVs, have gained large popularity in the research community over the years. Due to their size, inherent safety and agility, UAVs are not only great research platforms, but have great potential in tomorrow's civil and industrial tasks such as: search and rescue, surveillance and inspection. The control of UAVs during autonomous flights relies on the knowledge of variables, such as the position, velocity and orientation of the vehicle. However, these cannot be measured directly or drift-free by inertial sensors typically used onboard the UAVs, necessitating an additional position reference. While GPS is a popular choice for outdoor applications like aerial photography, it suffers from accuracy issues in cluttered environments, while it is entirely unavailable in indoor missions. Another popular choice in estimating the pose of a UAV with respect to its surroundings, is the use of laser-range finders, with many successful approaches existing in literature.

For control of UAVs, we study the necessary dynamics and differential flatness of multi-rotor systems. Accurate state estimation is key for successful control of the UAV. Using a Kalman Filter to fuse Estimated information with inertial measurements from the onboard sensor suite. The challenges in design and formulation of this framework, both from the integration and the algorithmic point of view are great: not only are sensor calibration states essential for consistent state estimation throughout an entire mission, but also handling of delays and different rates of the sensors' measurements are vital. There is currently a limited diversity of vehicles capable of Vertical Take-Off and Landing. However, there is a visible interest by scholars and admirers of Unmanned Aerial Vehicles for a particular type of aircraft called quadrotor or quadcopter.

The building a Quadcopter includes 1) Modelling the dynamics 2) Designing the control law 3) Trajectory generation. While modelling is the easiest of the three, once the control algorithm is perfected different algorithms can used to generate trajectories. Throughout the process MATLAB® is used for performing complex calculations and designing control law. Finally, after the control law is fixed, way to produce best estimates using Kalman filter is discussed as when coming to hardware implementation, no system has an ideal behaviour and sensors have noise. For hardware implementation, motor is modelled using test setups to model thrust and moment produced by each motor. The behaviour of quadcopter when the controller is running at different rates is compared.

With development of different tools and hardware support packages in MATLAB®, designing the control law, simulation and realizing the developed system has become easier which reduces the cost and time for developing a complex system. The controllers of an unmanned aerial vehicle can be both either an automated set of computers or a human operator. UAVs provide useful missions to prevent injuries, accidents, and hazards when it involves navigating in an environment that is not suitable for human survival. Today, UAVs are now undergoing a commercialization process because it is now being manufactured and produced by technological and logistics companies around the world. UAVs may have many good applications but they can be misused. The requirement of rules and regulations are discussed.

*****

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Overview

Small unmanned rotorcrafts have the potential to change the world in a positive way. They can maneuver in three dimensions to collect information with onboard sensors and even physically interact with their environments using onboard grippers. Full-size helicopters are presently used for numerous tasks that their smaller brethren could perform such as traffic, crop, and weather monitoring; building, bridge, and power line inspection; general surveillance activities; as well as aerial photography and videography applications. Un- manned rotorcraft does not require an onboard pilot which means they can be made smaller, safer, and cheaper.

The small size of these vehicles enables them to operate indoors in constrained spaces. This capability will be particularly useful in dangerous situations such as searching for survivors in damaged buildings, entering and clearing buildings with armed adversaries, and collecting information in buildings with biological or nuclear contamination. In these scenarios the ability to create situational awareness without ever having to put a human in harm’s way is extremely valuable.

It is true that a ground robot could accomplish some of these described tasks. Ground robots do have a considerable advantage over aerial vehicles in some situations because they can carry larger payloads, are generally more robust to collisions with the environment, do not require active sensing in order to stay in one place, and do not require the continuous use of energy to stay aloft. However, the ability to fly gives a rotorcraft access to locations that are impossible for a ground robot to reach. Additionally, navigating rough terrain and climbing stairs are difficult tasks for ground vehicles but in most environments the air space is relatively free so the navigation problem for aerial vehicles is considerably easier.

Another alternative to rotorcrafts is fixed-wing vehicles. For navigating long distances, the efficiency of a fixed-wing vehicle cannot be beat by a rotorcraft. However, compared with fixed-wing vehicles rotorcraft have the distinct advantage of being able to hover in place. Fixed-wing vehicles, in comparison, must be constantly moving forward to produce lift. The ability to hover is valuable for navigating constrained spaces as well as precisely picking up and dropping off payloads.

Quadrotor helicopters (or quadcopter) are a type of small unmanned rotorcraft. They have four fixed-pitch propellers attached to motors typically mounted in a cross configuration. are available from several companies as research and commercial vehicles as well as toys. The long moment arms on which the propellers lie enable them to produce large control moments perform aggressive maneuvers. One of the biggest advantages of quadcopter is their mechanical simplicity. In contrast, small-scale standard helicopters and coaxial helicopters require mechanisms to change the pitch of the propeller in order to produce control forces and moments.

1.2 History

This story begins in the 20th century, when Charles Richet, a French scientist and academician, built a small, un-piloted helicopter [1].Although his attempt was not a success, Louis Bréguet, one of Richet's students, was inspired by his tutor's example. Later in 1906, Louis and his brother,

Jacques Bréguet began the construction of the first quadrotor. Louis executed many tests on airfoil shapes, proving that he had at least some basic understanding of the requirements necessary to achieve vertical flight.

In 1907 they had finished the construction of the aircraft which was named Bréguet-Richet Gyroplane No. 1 (Fig 1.a and Fig 1.b) a quadrotor with propellers of 8.1 meters in diameter each, weighting 578 kg (2 pilots included) and with only one 50 hp (37.3 kW) internal combustion engine, which drove the rotors through a belt and pulley transmission. Of course, at that time they had no idea how they would control it, the main concern was to ensure the aircraft would achieve vertical flight. The first attempt of flight was done in between August and September of 1907 with witnesses saying they saw the quadrotor lift 1.5 m into the air for a few moments, landing immediately afterwards. Those same witnesses also mentioned the aircraft was stabilized, and perhaps even lifted by men assisting on the ground.

Discouraged by the lack of success of the Gyroplane No. 1, Bréguet and his mentor continued their pursuit to build vertical flight machines and afterwards also temporarily dedicated themselves to the development of fixed-wing aircraft, area where they became very successful. Louis never abandoned his passion for vertical flight aircraft and in 1932 he became one of the pioneers of helicopter development [1].

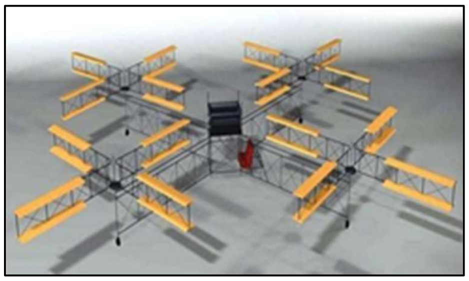


Fig 1.a 3D Model of gyroplane. [2]

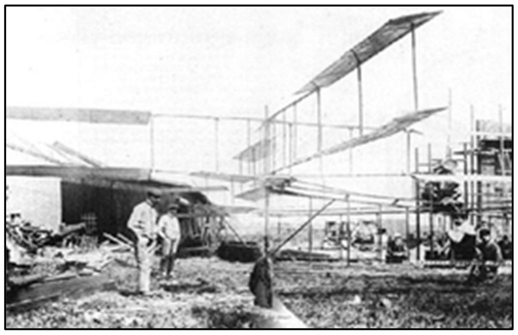


Fig 1.b Bréguet-Richet Gyroplane No. 1 of 1907. [2]

Etienne Oemichen, another engineer, also began experimenting with rotating-wing designs in 1920. He designed a grand total of six different vertical lift machines. The first model failed in lifting from the ground but Oemichen was a determined person, so he decided to add a hydrogen-filled balloon to provide both stability and lift. His second aircraft, the Oemichen No. 2 (Fig 1.c)had four rotors and eight propellers, supported by a cruciform steel-tube framework layout. Five of the propellers were meant to stabilize the machine laterally, another for steering and two for forward propulsion. Although rudimentary, this machine achieved a considerable degree of stability and controllability, having made more than a thousand test flights in the middle of that decade. It was even possible to maintain the aircraft several minutes in the air. In the 14th of May the machine was airborne for fourteen minutes and it flew more than a mile. But Oemichen was not satisfied with the poor heights he was able to fly, and the next machines had only a main rotor and two extra anti-torque rotors [2].

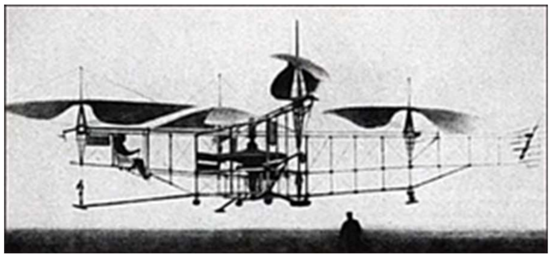


Fig 1.c The Oemichen No.2 of 1922. [2]

The army also had an interest for vertical lift machines. In 1921, Dr. George de Bothezat and Ivan Jerome were hired to develop one for the US Army Air Corps. The result was a 1678 kg structure with 9 m arms and four 8.1 m six-blade rotors (Fig 1.d)The army contract required that the aircraft would hover at 100 m high, but the best they achieved was 5 m. At the end of the project Bothezat demonstrated the vehicle could be quite stable, however it was underpowered and unresponsive, among other technical problems.

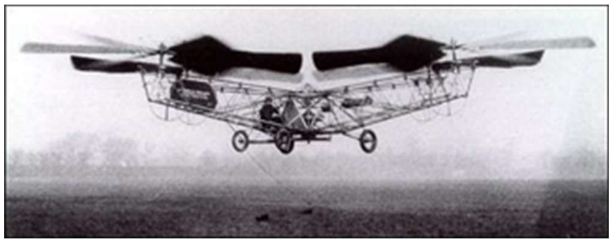


Fig 1.d Quadrotor designed by Dr. Bothezat an Ivan Jerome. 1922. [2]

Later in 1956, a quadrotor helicopter prototype called "Convertawings Model A" (Fig 1.e) was designed both for military and civilian use. It was controlled by varying the thrust between rotors, and its flights were a success, even in forward flight. The project ended mainly due to the lack of demand for the aircraft.

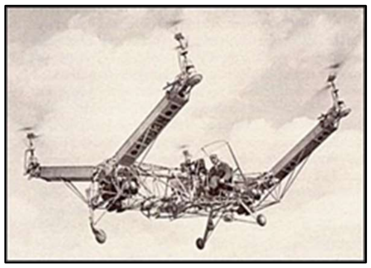


Fig 1.e Convertawings Model A helicopter. [2]

Recently there has been an increasing interest in quadrotor designs. Bell is working on a quad tiltrotor to overcome the V-22 Ospray (Fig 1.f)capable of carrying a large payload, achieving high velocity and while using a short amount of space for Vertical Take-Off and Landing (VTOL). Much of its systems come directly from the V-22 except for the number of engines. Also, the wing structure on the new design has some improvements, it has a wider wing span on the rear rotors. As a consequence, the Bell quad tiltrotor (Fig 1.g)aims for higher performance and fuel economy.

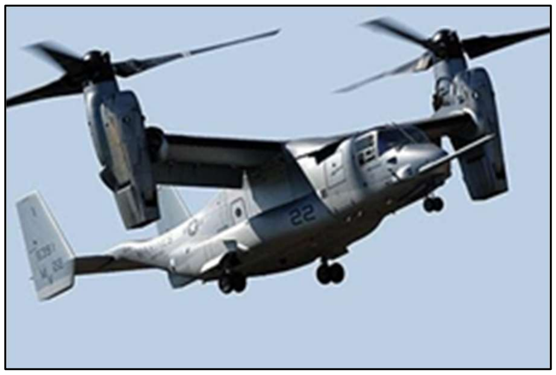


Fig 1.f V-22 Ospray. [2]

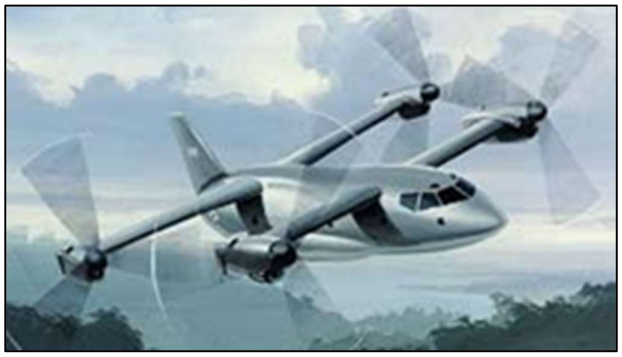


Fig 1.g Concept of Bell's quad tiltrotor. [2]

Another recent and famous quadrotor design is the Moller Skycar (Fig 1.h)a prototype for a personal VTOL "flying car". The Skycar has four ducted fans allowing for a safer and efficient operation at low speeds. It was a target for much criticism because the only demonstrations of flight were hover tests with the Skycar tethered to a crane. It's inventor, Paul Moller already tried to sell the Skycar by auction without success. Nowadays he focuses his work on the precursor of the Skycar, the "M200G Volantor", a flying saucer-style hovercraft. This later model uses eight fans controlled by a computer and is capable of hovering up to 3 m above the ground. This limitation is imposed by the on-board computer due to regulations of the Federal Aviation Administrations, stating that any vehicle that flies above 3 m is regulated as an aircraft.

Fig 1.h Skycar during a test flight. [2]

Quadrotors are also available to the public through radio-controlled toys. Some enthusiasts as well as researches have been developing their own quadrotor prototypes. This is possible due to the availability

Fig 1.i Ambulance stuck in a traffic jam in Lisbon, Portugal. [2]

of cheap electronics and lightweight resistant materials available to the public. Be it for personal satisfaction, entertainment, military or civilian use, quadrotors have played an important role in the evolution of aircrafts and may prove themselves as important means of transportation in a near future.

1.3 <u>Objective</u>

As it has been already stated in the abstract, this project is turning around an unmanned flying vehicle called drone. The aim of this project is to find an appropriate mathematical model for such a machine and develop a complete control architecture which will allow the drone to fly. Using these features then we develop our UAV, for observation and scouting missions for civilian or even military personnel.

- Define the mathematical model for the Quadcopter.
- Design Position Tracking Controller.

- Generate suitable Trajectories for the Quadcopter.
- Simulation of the Trajectory Tracking of Quadcopter using MATLAB®.

1.4 Organisation of Report

In CHAPTER 2 we discussed the basic physics and the mathematical modelling of the quadcopters.

In CHAPTER 3 we went through control systems design step by step in quadcopters. Further, we will be discussing about Trajectory Generation in general. We have also discussed about minimum jerk trajectory and its solving.

In CHAPTER 4 we would be seeing the implementation of quadcopters on MATLAB® and Simulink® and tuning of controllers and the results we obtained.

In CHAPTER 5 we will be discussing about various hardware required to implement the Quadcopter.

In CHAPTER 6 discussed about the future work in quadcopters.

CHAPTER 2
MATHEMATICAL MODEL

The most important target of this particular design process is to arrive at the correct set of requirements for the aircraft, which are often summarized in a set of specifications. In this chapter we discuss about the dynamics of the quadcopter.

The Propellers act as wings. They generate thrust by rotating at Fast speeds, which pulls the air downwards and keeps the quadcopter in the air.

2.1 <u>Basic physics</u>

- Weight of quadcopter is cancelled by Thrust produced by propellers.
- Net direction of thrust gives direction of movement of quadcopter.
- Conservation of angular momentum is done by rotating one set of opposite propellers in clockwise and another set of opposite propellers in anti-clockwise.

Movements of Quadcopter

1. **Take off:** To Rise above the ground, you need a net upward Force. The Motors generate Thrust that is greater than the Weight, making the quad rise upwards [3].

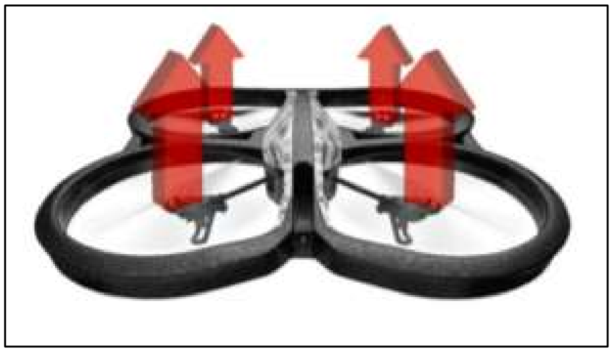


Fig 2.a : Take-off Motion [4]

2. **Hover:** Hovering in Air is simple. Motors generate Thrust. The Thrust should equal the weight of the System. The two forces cancel and our drone hovers.
3. **Attitude:**

- **Roll:**

  To Roll towards the Left (Our Left), the Thrust is increased on the 2 motors (X configuration) or a motor (+ configuration) on the Right. We also decrease the Thrust on the 2 motors (X configuration) or a motor (+ configuration) on the Left. To Roll towards the Right (Our Right), the Thrust is increased on the 2 motors (X configuration) or a motor (+ configuration) on the Left. We also decrease the Thrust on the 2 motors (X configuration) or a motor (+ configuration) on the Right.

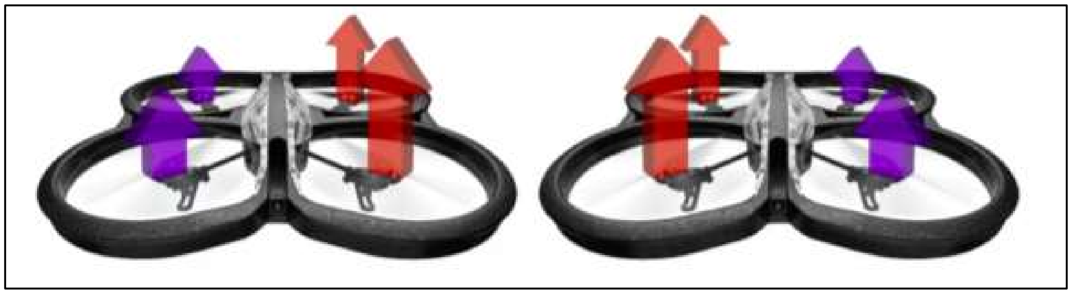


Fig 2.b : Roll Motion [4]

- **Pitch:**

  To Pitch Forwards (Towards us) The Power to the 2 rear motors (X configuration) or a rear motor (+ configuration) motors is increased. This creates a net forward force which causes the Drones nose to Pitch Downward. We also decrease the power to the 2 front motors (X configuration) or a front motor (+ configuration) motors to keep the angular momentum conserved.

  To Pitch Backwards (Away from us) The Power to the 2 rear motors (X configuration) or a rear motor (+ configuration) motors is decreased. This creates a net backward force which causes the Drones nose to Pitch Upward. We also increase the power to the 2 front motors (X configuration) or a front motor (+ configuration) motors to keep the angular momentum conserved.

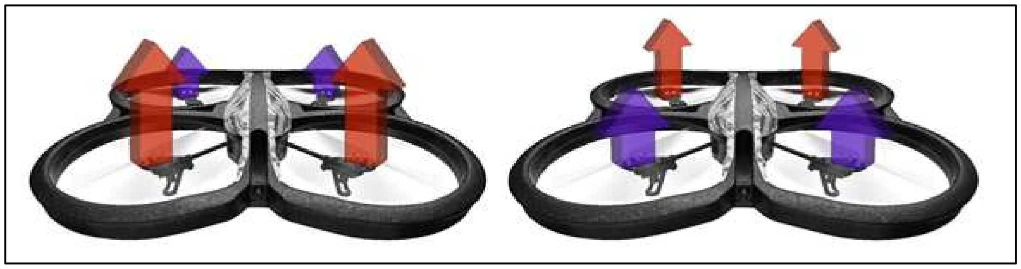


Fig 2.c : Pitch Motion [4]

- **Yaw:**

  To Yaw Clockwise. We increase the Thrust on the Anti-Clockwise moving Motors. Decrease the Thrust on Clockwise Rotating Motors. To keep the Net upward /downward force zero. There is a resulting Anti-Clockwise Torque. The Quad rotates Clockwise to conserve the Angular Momentum.

  To Yaw Anti-Clockwise. We decrease the Thrust on the Anti-Clockwise moving Motors. Increase the Thrust on Clockwise Rotating Motors. To keep the Net upward/downward force zero. There is a resulting Clockwise Torque. The Quad rotates Anti-Clockwise to conserve the Angular Momentum.

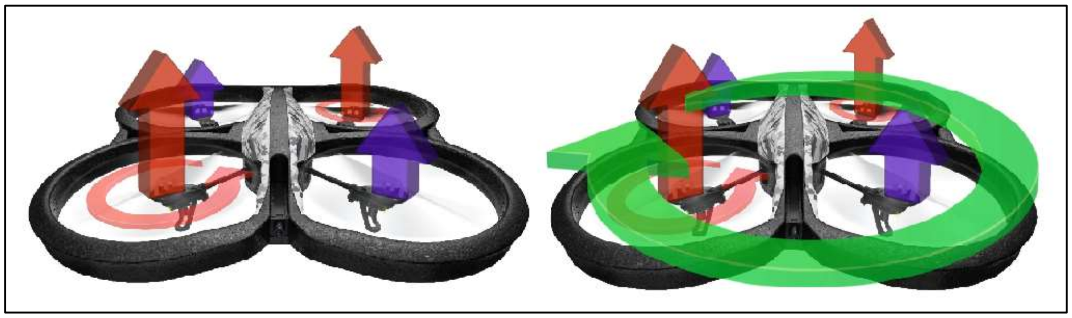


Fig 2.d : Yaw Motion [4]

There are two kind of propellers. Clockwise and Anti-Clockwise propellers which will produce thrust upward in their respective name directions. We must keep two clockwise propellers opposite arms and anti-clockwise propellers in another opposite arm so the angular momentum produced by one set propellers is countered by another set of propellers and angular momentum is conserved as discussed earlier.

Here one can ask the question that why opposite arms only. Why not on side-by-side arms? Even by keeping the same kind of propellers on side-by-side arms also angular momentum is conserved. But the answer is by placing the propellers like above mentioned way and try to perform any action like pitch or roll or yaw, then you will get not only that motion addition to that you can also get any one from other types of motion.

For example, if you are giving pitch motion by reducing the speeds of clockwise rotating motors then other motors are rotating at higher speeds and due to this the body or the frame of the quadcopter will rotate in opposite direction with respect to the direction of the motor at higher speed. So, at this point you will get yaw motion along with required pitch motion which causes the quadcopter more unstable and very difficult to control from that point. Similarly, in this kind of motor placement, you won't be able to perform yaw motion since by reducing the speeds of the diagonal motors won't affect the angular momentum. Because it is balanced by the other two motors which are rotating opposite to each other. So, to avoid these types of problems we will place motors rotation in same direction on diagonal (opposite) positions of the quadcopter.

2.2 Mathematical Model

Mathematical Model of a system is used to express physical systems using mathematical equations. After developing a conceptual model of a physical system, it is natural to develop a mathematical model that will allow one to estimate the quantitative behaviour of the system. In our project, we are using the mathematical model to understand the dynamics of quadcopter completely and to tune the control system parameters automatically.

2.2.1 Assumptions:

- The quadcopter is a rigid body.
- Quadcopter is symmetrical.
- Centre of mass coincides with the geometric centre.
- Inertia of the motor is considered to be neglected.
- Earth is assumed to be flat.

2.2.2 Notation:

Due to the complexity of a system with 6 degrees of freedom, various methods of notation have been developed and are required in order to sufficiently describe the critical variables.

$${}^{y}\boldsymbol{x}_{\boldsymbol{h}|\boldsymbol{f}}^{\boldsymbol{z}} \tag{2.1}$$

The base variable **x** is state variable. $\dot{x}$ is its first derivative. $\ddot{x}$ is second derivative. The left superscript **y** tells us from which frame the **x** variable is taken or performed. While, the right superscript **z** tells us from which frame the **x** variable vector components are represented. **h** represents from which point the variable value is and **f** is from which frame. **i** is inertial frame. **b** is body frame. **CM** is Centre of Mass.

2.2.3 Coordinate System:

Another important aspect of the math model is the coordinate system that is used. The chosen model and conventions will become very important as you work through your model, so be sure to keep your decisions in mind and clearly documented. The coordinate system will vary whether you use a X or plus (+) configuration as shown in Fig 2.e, Fig 2.f respectively. In this project X configuration is used because if you have a "+" configuration, then the thrust forces are applied at a distance r. If you have an X configuration, then the thrust forces are applied at distance of r*cos(π/4) approximately 0.71*r since the arms are at a 45^0 angle from the axis of rotation. The moment of inertia is the same when you do the math, so the difference is really that you can torque with all four motors, and therefore have √ 2 more available torque to rotate. This means you can get about 41% more rotational acceleration from X configuration than that of a "+".

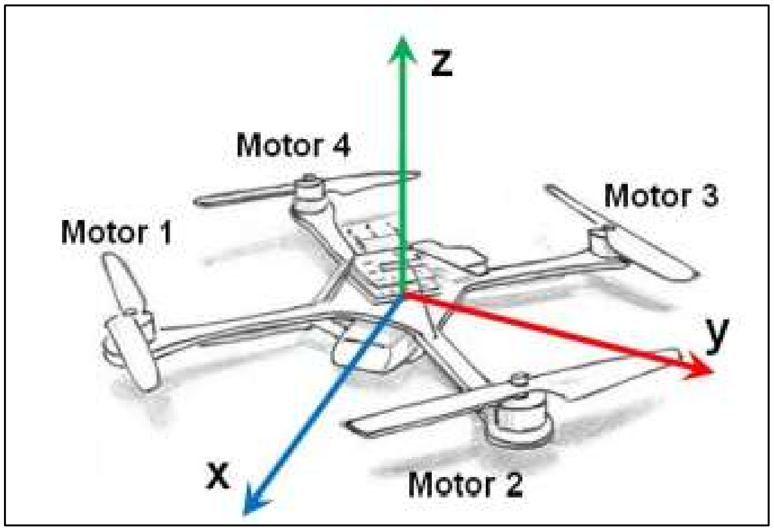


Fig 2.e : X configuration of the quadcopter [5]

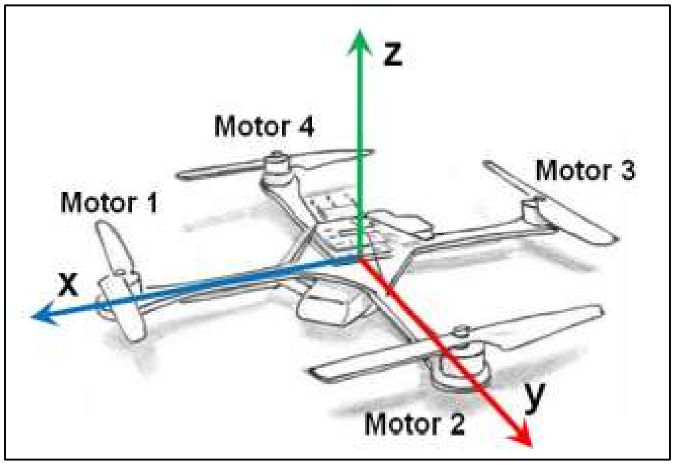


Fig 2.f : + configuration of the quadcopter [5]

2.2.4 Moment of Inertia:

Mass and its geometric distribution in an aircraft is something of extreme importance because it affects the entire dynamics of the system. Then, after building the quadrotor, it is time to evaluate some of its most important features like its moments of inertia and mass. We will assume the inertia matrix is diagonal and positive-definite, with the purpose of simplifying the calculations and also due to the particular symmetric geometry of quadrotors. As such, the calculation of inertia moments will only include the geometry and mass of the motors, as well as their geometric position on the quadrotor.

Moment of Inertia matrix w.r.t body is denoted by I^b is as follows:

$$I^b = \begin{bmatrix} \mathrm{I}_{xx} & 0 & 0 \\ 0 & \mathrm{I}_{yy} & 0 \\ 0 & 0 & \mathrm{I}_{zz} \end{bmatrix} \qquad (2.2)$$

Here, Moment of Inertia w.r.t x-axis is denoted by I_{xx}. Moment of Inertia w.r.t y-axis is denoted by I_{yy}. Moment of Inertia w.r.t z-axis is denoted by I_{zz}.

2.2.5 Modelling of Motor Propeller Assembly

The motor-propeller assemblies are essential components of our aircraft since they are responsible for the production of the lifting force that allows flight. This section focuses on modelling the dynamics of these components.

2.2.5.1 Thrust Coefficient:

The motors thrust is the driving force behind all quadcopters. The thrust provided by a single motor/prop system can be calculated as follows [6, 5]

$$T = C_T \rho A_r r^2 \omega^2 \qquad (2.3)$$

- C_T: thrust coefficient
- ρ: density of air
- A_r: the cross-sectional area of the propeller's rotation.
- r: is the radius of the rotor.
- ω: the cross is the angular velocity of the rotor.

For simple flight modelling, a lumped parameter approach can be used to simplify the characterization process

$$T = C_T \omega^2 \qquad (2.4)$$

Where C_T is the lumped parameter thrust coefficient that pertains to the individual motor/prop system.

2.2.5.2 Torque Coefficient:

In order to understand the motor effect on yaw, the torque force of the motor/prop system must also be determined and can be done in a similar fashion to that of the thrust tests. The related lumped parameter equation is shown below [6, 5]

$$Q = C_Q \omega^2 \qquad (2.5)$$

where C_Q is the lumped parameter torque coefficient that pertains to the individual motor/prop system. C_T and C_Q can be calculated experimentally. But, due to unavailability of hardware at present, we are not mentioning them in this report.

2.2.6 Thrust-Torque Matrix:

We can create a matrix describing the thrust and torques on the system as shown in equation (2.6.

$$\begin{bmatrix} \Sigma\ T \\ \tau_{\emptyset} \\ \tau_{\theta} \\ \tau_{\psi} \end{bmatrix} = \begin{bmatrix} C_T & C_T & C_T & C_T \\ -d * C_T & d * C_T & d * C_T & -d * C_T \\ -d * C_T & -d * C_T & d * C_T & d * C_T \\ -C_Q & C_Q & -C_Q & C_Q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2.6}$$

All of the parameters in equation (2.6 have been explained thus far except for d, which is simply the distance between the motors and the respective axes of rotation, where is the arm length from quadcopter hub centre to motor/prop.

If using an **X** configuration, d can instead be found by d*cos (45^0), as that would be the value for the distance between the motor/prop and the body's axes of rotation. Thus, experiences no change from this configuration adjustment, while the effect will be distributed across all four motors for both pitch and roll motion.

$$M_{A,T}^b = \begin{bmatrix} \tau_{\emptyset} \\ \tau_{\theta} \\ \tau_{\psi} \end{bmatrix} \tag{2.7}$$

Here, $M_{A,T}^b$ refers to the moments present in the body frame resulting from the aerodynamics, thrusts, and torques on the system. The quadcopter body also experiences forces that act on it from gravity and the lift of the rotors. The lift force can be expressed as follows:

$$F_{A,T}^b = \begin{bmatrix} 0 \\ 0 \\ \Sigma\ T \end{bmatrix} \tag{2.8}$$

$F_{A,T}^b$ refers to the forces acting in the body frame on the quadcopter due to aerodynamics and thrust (assumed oriented strictly in the positive z direction). It should be noted that while we say there are acting aerodynamic forces, it is assumed that the static thrust and torque tests capture the elements of aerodynamics that we are interested in. Additional effects (such as blade flapping, frame aerodynamic drag, etc.) could be added to the model after additional research and testing.

2.2.7 Rotation Matrix

We must deduce the equations describing the orientation of the body frame relative to the inertial one, which can be achieved by using a rotation matrix. According to the aerospace rotation sequence, the rotation of an aircraft is described as a rotation about the z-axis (yaw) then a rotation about the y-axis (pitch) followed by a rotation about the x-axis (roll). Each rotation is made based on a right-handed system and in a single plane. [7, 8]

$$R_{\emptyset} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\emptyset) & \sin(\emptyset) \\ 0 & -\sin(\emptyset) & \cos(\emptyset) \end{bmatrix} \tag{2.9}$$

$$R_{\theta} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.10}$$

$$R_{\psi} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Using these three rotations a composite rotation matrix can be created which can transform the motion of the aircraft from the body frame to a new reference frame. The resulting rotation matrix transforms rotations from the body frame with respect to the inertial frame and can be found using matrix multiplication.

$$R_{b|i} = R_{\emptyset} * R_{\theta} * R_{\psi} \tag{2.12}$$

$$R_{b|i} = \begin{bmatrix} c(\psi)\,c(\theta) & c(\theta)\,s(\psi) & -s(\theta) \\ c(\psi)\,c(\emptyset)\,s(\theta) - c(\emptyset)\,s(\psi) & c(\emptyset)\,c(\psi) + s(\emptyset)\,s(\psi)\,s(\theta) & c(\theta)\,s(\psi) \\ s(\emptyset)\,s(\psi) + c(\emptyset)\,c(\psi)\,s(\theta) & s(\psi)\,c(\emptyset)\,s(\theta) - s(\emptyset)\,c(\psi) & c(\emptyset)\,c(\theta) \end{bmatrix} \tag{2.13}$$

Rotational Matrices are Orthogonal and Unimodular.

2.2.8 <u>State Equations</u>

In control theory, knowledge about the dynamic behaviour of a given system can be acquired through its states. [9] For a quadrotor, its attitude about all 3 axes of rotation is known with 6 states: the Euler angles [$\emptyset$ θ ψ] (Roll – Pitch – Yaw) and the angular velocities around each axis of the body frame [P Q R].

Yet another 6 states are necessary: the position of the centre of gravity (or COG) [X Y Z] and respective linear velocity components [U V W] relative to the fixed frame. In sum, the quadrotor has 12 states that describe 6 degrees of freedom.

Now we move on to the state equations that define the dynamics model. The first we'll discuss is the Angular Velocity State Equation.

$$^{\mathrm{b}}\dot{\omega}_{b|i}^{b} = (I^b)^{-1}\left[M_{A,T}^{b} - \Omega_{b|i}^{b} I^b \omega_{b|i}^{b}\right] = \begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} \tag{2.14}$$

Equation (2.14 will be referred as Angular Velocity State Equation. This equation describes the change in roll (***P***), pitch (***Q***), and yaw (***R***) rates of the quadcopter by taking into account the inertia, angular velocity, and the moments applied by the motor/prop systems. $^{b}\dot{\omega}_{b|i}^{b}$ is the angular acceleration across each axis in the body frame with respect to the inertial frame. Having already the inertia matrix and moment matrices, we move on to $\Omega_{b|i}^{b}$ which is a cross-product matrix for rotational velocity. The form of this matrix is shown below:

$$\Omega_{b|i}^{b} = \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix} \tag{2.15}$$

Here, **P**, **Q** and **R**, are again the rotation rates about the **X**, **Y** and **Z** axis, respectively. The $\omega_{b|i}^{b}$ term is the rotational velocity of the quadcopter body within the body frame and is defined directly by **P**, **Q**, and **R**.

$$^{\mathrm{b}}\omega_{b|i}^{b} = \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \tag{2.16}$$

The next state equation defined is the Euler Kinematic Equation, which allows us to determine the rate of change of the Euler angles in the inertial frame.

$$\dot{\Phi} = H(\Phi)\omega_{b|i}^{b} = \begin{bmatrix} \dot{\emptyset} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.17}$$

Using sequential rotation matrices, the angular velocity of the aircraft in the body frame can be related to the changes in angle rotation as shown below, where the ***R*** matrices of $\emptyset$ and θ are those from ***R***.

$$H(\Phi) = \begin{bmatrix} \dot{\emptyset} \\ 0 \\ 0 \end{bmatrix} + R_{\emptyset}\left(\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_{\theta}\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}\right) \tag{2.18}$$

Performing matrix multiplication and addition and taking the derivative the Euler Kinematic Equation can be found.

$$\dot{\Phi} = \begin{bmatrix} 1 & \tan(\theta)\sin(\emptyset) & \tan(\theta)\cos(\emptyset) \\ 0 & \cos(\emptyset) & -\sin(\emptyset) \\ 0 & \sin(\emptyset)\,/\cos(\theta) & \cos(\emptyset)\,/\cos(\theta) \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \tag{2.19}$$

Equation (2.19 will be referred as Euler Kinematic Equation. While this approach is effective, there is one very important drawback; a singularity occurs when θ is equal to ± 90°. Due to this, the accuracy and numerical stability of a simulation can be compromised if the aircraft's pitch approaches or reaches ±90°. Considering the modest control design intentions of this simulation, this will not be an issue for most users. However, several approaches to avoiding this problem exist, including using quaternions for the simulation, and thus motivated users may choose to modify our simulation to utilize this or another approach to removing this singularity.

The next state equation we discuss is the Velocity State Equation, which describes the acceleration of the centre of mass of the rigid body quadcopter model based on the forces and accelerations acting on the body.

$$^{\mathrm{b}}\dot{v}^{b}_{CM|i} = \left(\frac{1}{m}\right) F^{b}_{A,T} + g^{b} - \Omega^{b}_{b|i} I^{b} \omega^{b}_{CM|i} = \begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix} \tag{2.20}$$

Here, $^{\mathrm{b}}\dot{v}^{b}_{CM|i}$ is the linear acceleration of the center of mass in the body frame with respect to the inertial frame. The variable m is the total mass of the quadcopter, while g^{b} is the acceleration of gravity translated to act in the body frame by the rotation matrix $\boldsymbol{R}$.

$$g^{b} = Rg^{i} \tag{2.21}$$

Using these equations, you can find the linear acceleration of the quadcopter in the X, Y, and Z directions of the body frame.

Finally, the last state equation to be covered is the Position State Equation, which describes the linear velocity of the center of mass of the quadcopter in the inertial frame.

$$^{\mathrm{i}}\dot{P}^{i}_{CM|i} = R^{T} v^{b}_{CM|i} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \tag{2.22}$$

Equation (2.22 will be referred as Position State Equation. We are taking full advantage of the orthogonality of S, meaning its inverse is equal to its transpose. Here, $\mathrm{v}^{b}_{CM|i}$ is simply the velocity of the quadcopter in the body frame that is rotated into the inertial frame using the transpose of $\boldsymbol{R}$, which is $\boldsymbol{R^T}$. This state equation allows us to determine the velocity of the quadcopter in the **X**, **Y** and **Z** directions of the inertial frame.

2.3 Linear model

We know that many controllers (like PID, LQR, etc.,) work efficiently only on Linear Systems. But this quadcopter is a non-linear system. So, classic controllers can't work efficiently on Quadcopter. To make use of classic controllers for Quadcopter system we must linearize the Quadcopter. To linearize Quadcopter, we make some assumptions [9] [10] [6].

- We assume that Quadcopter will be in near hover condition all through the flight
- The operating angles (pitch, roll) are small.

With above assumptions, model will become as

$$m\ddot{x} = (\theta\cos\psi + \emptyset\sin\psi)(\Sigma\, T) \tag{2.23}$$

$$m\ddot{y} = (\theta\sin\psi + \emptyset\cos\psi)(\Sigma\, T) \tag{2.24}$$

$$m\ddot{z} = -mg + (\Sigma\, T) \tag{2.25}$$

$$\ddot{\emptyset} = \tau_{\emptyset}/\mathrm{I_{xx}} \tag{2.26}$$

$$\ddot{\theta} = \tau_{\theta}/\mathrm{I}_{yy} \tag{2.27}$$

$$\ddot{\psi} = \tau_{\psi}/\mathrm{I}_{zz} \tag{2.28}$$

These are the final governing equations for the Quadcopter system to which we are designing the Controller. [9]

CHAPTER 3
CONTROL SYSTEM DESIGN

A control system manages, commands direct, or regulates the behaviour of other devices or systems using control loops. It can range from a single home heating controller using a thermostat to controlling a domestic boiler to large Industrial control systems which are used for controlling processes or machines.

The main feature of a control system is that there should be a clear mathematical relationship between input and output of the system. When the relation between input and output of the system can be represented by a linear proportionality, the system is called a linear control system. Again, when the relationship between input and output cannot be represented by single linear proportionality, rather the input and output are related by some non-linear relation, the system is referred to as a non-linear control system. There are two main types of control system. They are as follow

- Open Loop Control System
- Closed Loop Control System

3.1 <u>Open Loop Control system</u>

A control system in which the control action is totally independent of output of the system then it is called an open loop control system. A manual control system is also an open loop control system. The figure below shows a control system block diagram of an open loop control system in which process output is totally independent of the controller action.

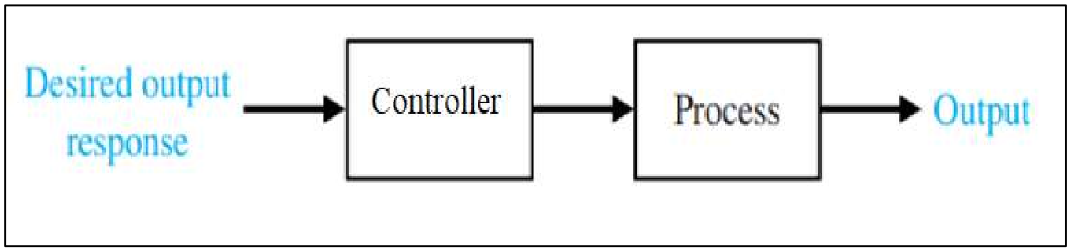


Fig 3.a : Open loop control system

3.2 Closed Loop Control system

Control system in which the output has an effect on the input quantity in such a manner that the input quantity will adjust itself based on the output generated is called closed loop control system. Open loop control system can be converted in to closed loop control system by providing System feedback. This feedback automatically makes the suitable changes in the output due to external disturbance. In this way closed loop control system is called automatic control system. Figure below shows the block diagram of closed loop control system in which feedback is taken from output and fed in to input.

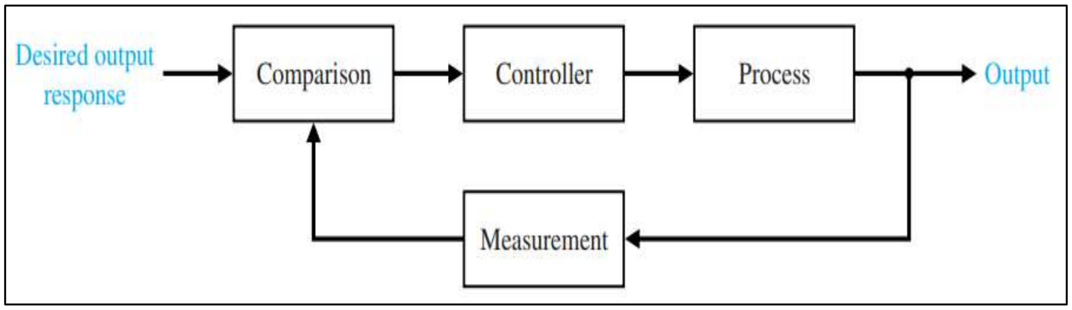


Fig 3.b : Closed loop control system

For continuously modulated control, a feedback controller is used to automatically control a process or operation. The control system compares the value or status of the process variable (PV) being controlled with the desired value or set point (SP), and applies the difference as a control signal to bring the process variable output of the plant to the same value as the set point.

Advantages of Feedback in Control Compared to open-loop control, feedback can be used to

- Reduce the sensitivity of a systems transfer function to parameter changes.
- Reduce steady-state error in response to disturbances.

- Reduce steady-state error in tracking a reference response (speed up the transient response).
- Stabilize an unstable process.

3.3 PID Controller

A proportional-integral-derivative controller (PID controller) is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates the correction value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable [11].

The PID controller algorithm involves three separate constant parameters, and is accordingly sometimes called three-term control [12]

$$\mathrm{u(t)} = \mathrm{K}\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{3.1}$$

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining Correction as the controller output [11]

$$u = K_P * \epsilon + K_I * \frac{1}{T_i}\int \epsilon dt + K_D * \frac{d\epsilon}{dt} \tag{3.2}$$

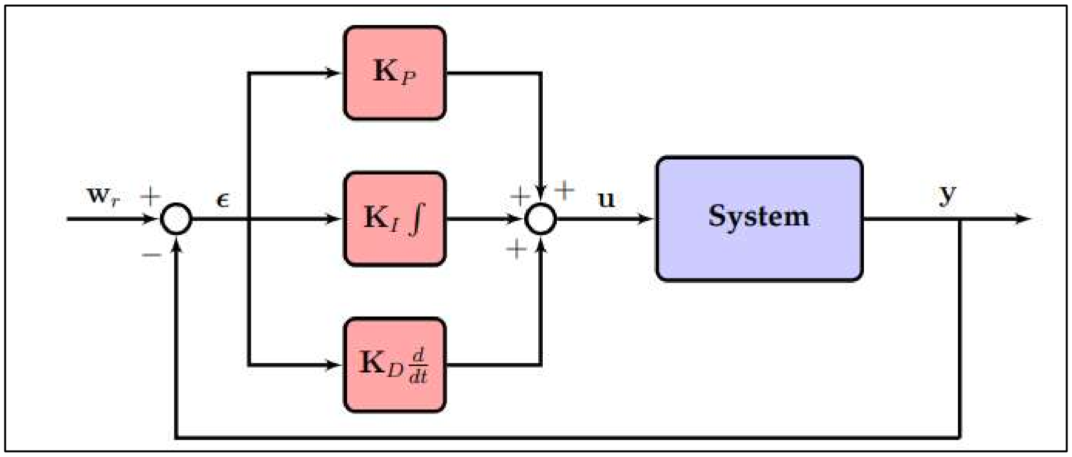


Fig 3.c : schematic structure of the PID controller. [10]

3.3.1 <u>Proportional Term</u>

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain constant. The proportional term is given by: [12]

$$P_{out} = K_p \mathrm{e}(t) \tag{3.3}$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change.

3.3.2 <u>Integral Term</u>

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain K_i and added to the controller output [12].

$$I_{out} = K_i \int_0^t \mathrm{e}(t)dt \tag{3.4}$$

The integral term accelerates the movement of the process towards set-point and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set-point value.

3.3.3 <u>Derivative Term</u>

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d. The derivative term is given by [12]

$$D_{out} = K_d \frac{de(t)}{dt} \tag{3.5}$$

Derivative action predicts system behaviour and thus improves settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low pass filtering for the derivative term, to limit the high frequency gain and noise. Derivative action is seldom used in practice though by one estimate in only 20% of deployed controllers because of its variable impact on system stability in real-world applications.

Table 3.1 Effects of Independent P, I and D tuning [12]

Closed Loop Response	**Risetime**	**Overshoot**	**Settling Time**	**Steady State Error**	**Stability**
Increasing K_p	Decrease	Increase	Small Increase	Decrease	Degrade
Increasing K_i	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing K_d	Small Decrease	Decrease	Decrease	Minor Change	Improve

3.4 Control System Design for Quadcopter

To design a control system for any system we must first identify the controlling parameters and measuring parameters. In the case of Quadcopter controlling parameters are rpms of four rotors and measuring parameters are its attitude (orientation w.r.t pitch, roll and yaw axes) and position (w.r.t X, Y, Z axes). Measuring Parameters of a system are inputs for the control system and controlling parameters of the system will be outputs of the control system. Designing the control systems includes manipulating the controllable parameters in such a way that measuring parameters are stabilized.

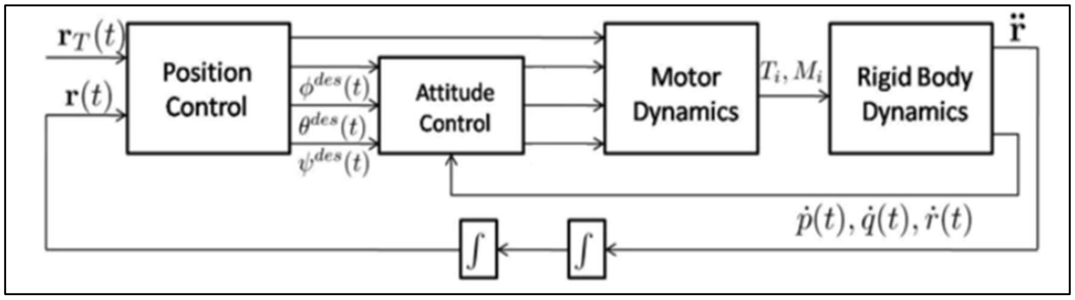


Fig 3.d : The nested control loops for position and attitude control

3.5 Small angle control

Each robot is controlled independently by nested feedback loops as shown in Fig. 3.2. The inner attitude control loop uses on-board accelerometers and gyros to control the roll, pitch, and yaw and runs at approximately 1 kHz [7], while the outer position control loop uses estimates of position and velocity of the centre of mass to control the trajectory in three dimensions. Our controllers are derived by linearizing the equations of motion and motor models at an operating point that corresponds to the nominal hover state, r = r_0, $\theta = \emptyset = 0$, $\psi = \psi$, $\dot{r} = 0$, and $\dot{\emptyset} = \dot{\theta} = \dot{\psi} = 0$, where the roll and pitch angles are small ($c\varphi \approx 1$, $c\theta \approx 1$, $s\varphi \approx \varphi$, and $s\theta \approx \theta$). At this hover state, the nominal thrusts from the propellers must satisfy

$$\mathrm{F} = \frac{\mathrm{mg}}{4} \tag{3.6}$$

and the motor speeds are given by

$$w = \sqrt{\frac{mg}{4c_T}} \tag{3.7}$$

3.5.1 Attitude Control

We now present an attitude controller to track trajectories that are close to the nominal hover state where the roll and pitch angles are small. From (2.1), if we assume that the products of inertia are small (ideally, they are zero because the axes are close to the principal axes) and $I_{xx} \approx I_{yy}$ because of the symmetry then:

$$\mathrm{I}_{xx}\,\dot{p} \;=\; \mathrm{u}_2 \;-\; \mathrm{qr}(\mathrm{I}_{zz} \;-\; \mathrm{I}_{yy}) \tag{3.8}$$

$$\mathrm{I}_{xx}\,\dot{q} = \mathrm{u}_3 - \mathrm{pr}(\mathrm{I}_{xx} - \mathrm{I}_{zz}) \qquad (3.9)$$

$$\mathrm{I}_{zz}\,\dot{r} = \mathrm{u}_4 \qquad (3.10)$$

We can also assume the component of the angular velocity in the z direction, *r*, is small so the rightmost terms in equations (3.8) and (3.9) which are products involving r, are small compared to the other terms. We note that near the nominal hover state $\dot{\emptyset} \approx p$, $\dot{\theta} \approx q$, and $\dot{\psi} \approx r$. For these reasons we can use simple proportional derivative control laws that take the form

$$\mathrm{u}_{2,des} = \mathrm{k}_{p,\varphi}(\varphi^{des} - \varphi) + \mathrm{k}_{d,\varphi}(\mathrm{p}^{des} - \mathrm{p}) \qquad (3.11)$$

$$\mathrm{u}_{3,des} = \mathrm{k}_{p,\theta}(\theta^{des} - \theta) + \mathrm{k}_{d,\theta}(\mathrm{q}^{des} - \mathrm{q}) \qquad (3.12)$$

$$\mathrm{u}_{4,des} = \mathrm{k}_{p,\psi}(\psi^{des} - \psi) + \mathrm{k}_{d,\psi}(\mathrm{r}^{des} - \mathrm{r}) \qquad (3.13)$$

The vector of desired rotor speeds can be found from the desired net force ($u_{1,des}$) and moments ($u_{2,des}$, $u_{3,des}$ and $u_{4,des}$) by inverting

$$u_{des} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_M & -K_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_{1,des}^2 \\ \omega_{2,des}^2 \\ \omega_{3,des}^2 \\ \omega_{4,des}^2 \end{bmatrix} \qquad (3.14)$$

3.5.2 Position control

Here we present two representative position control methods that use the roll and pitch angles as inputs via a method similar to a back stepping approach. The first, a hover controller, is used for station-keeping or maintaining the position at a desired *x*, *y*, and *z* location. The second tracks a trajectory in three dimensions.

3.5.2.1 Hover Control

Here we use pitch and roll angle to control position in the x_w and y_w plane, u_4 to control yaw angle, and u_1 to control position along z_w. We let $r_T(t)$ and $\psi_T(t)$ be the trajectory and yaw angle we are trying to track. Note that $\psi_T(t) = \psi_0$ for the hover controller. The command accelerations, $\ddot{r}_{des}$, are calculated from PID feedback of the position error,

$e_i = (r_{i,T} - r_i)$, as

$$\ddot{r}_1^{des} = g\left(\theta^{des}\cos(\psi_T) + \emptyset^{des}\sin(\psi_T)\right) \tag{3.15}$$

$$\ddot{r}_2^{des} = g\left(\theta^{des}\sin(\psi_T) - \emptyset^{des}\cos(\psi_T)\right) \tag{3.16}$$

$$\ddot{r}_3^{des} = \frac{u_{1,des}}{m} \tag{3.17}$$

These relationships are inverted to compute the desired roll and pitch angles for the attitude controller, from the desired accelerations, as well as $u_{1,des}$

$$\emptyset^{des} = \frac{1}{g}\left(\ddot{r}_1^{des}\sin(\psi_T) - \ddot{r}_2^{des}\cos(\psi_T)\right) \tag{3.18}$$

$$\theta^{des} = \frac{1}{g}\left(\ddot{r}_1^{des}\cos(\psi_T) + \ddot{r}_2^{des}\sin(\psi_T)\right) \tag{3.19}$$

$$u_{1,des} = m\ddot{r}_3^{des} \tag{3.20}$$

3.6 <u>3D Trajectory Control</u>

The 3D Trajectory Controller is used to follow three-dimensional trajectories with modest accelerations so the near-hover assumptions hold. We have a method for calculating the closest point on the trajectory, $\boldsymbol{r}_T$, to the current position, **r**. Let the unit tangent vector of the trajectory associated with that point be $\hat{\boldsymbol{t}}$ and the desired velocity vector be $\dot{\boldsymbol{r}}_{\boldsymbol{T}}$. We define the position and velocity errors as

$$e_p = \left((r_T - r)\,\hat{n}\right)\hat{n} + \left((r_T - r)\,\hat{b}\right)\hat{b} \tag{3.21}$$

and

$$e_v = \dot{r}_T - \dot{r} \tag{3.22}$$

Note that here we ignore position error in the tangent direction by only considering position error in the normal, $\hat{n}$**,** and binormal, $\hat{b}$, directions.

We calculate the commanded acceleration. $\ddot{r}_1^{des}$ from PD feedback of the position and velocity errors:

$$\ddot{r}^{des} = k_{p,i}e_{p,i} + k_{d,i}e_{i,v} + \ddot{r}_{i,T} \tag{3.23}$$

Note that the $\ddot{r}_{i,T}$ terms represent feedforward terms on the desired accelerations. At low accelerations these terms can be ignored but at larger accelerations they can significantly improve controller performance. Finally, we use equations (3.18), (3.19), and (3.20) to compute the desired roll and pitch angles as well as $u_{1,des}$

3.7 <u>Trajectory Generation</u>

Imagine you have a rigid body or a quadrotor that needs to go from a start position to a goal position. we're also interested in orientations. And in this problem, you may have intermediate positions that you want the rigid body or the quadrotor to go through. This is a very general problem, it arises in all contexts in robotics, and it's particularly important for motion planning for quadrotors. The general set up of this problem is as follows.

We're given start and goal positions, optionally orientations. We might want the quadrotor to visit intermediate positions or waypoints which can also include orientations. In general, we'll insist that the trajectories that the quadrotors follow are smooth because the quadrotor is a dynamical system, it cannot follow arbitrary trajectories. This generally translates to minimizing a rate of change of input. We'll be particularly interested in the order of the dynamical system.

If you consider a robot with a kinematic model, in other words, you assume that you can arbitrarily specify velocities, that's a first order system. If you have a robot with second order dynamics, that means that you can arbitrarily specify accelerations. For third order systems, you should be able to control or specify or command the third order derivative which is called jerk. And likewise, a fourth order system involves specifying or commanding the fourth derivative, which is called snap.

The order of the system determines the input. If it's an nth order system, in other words, you're specifying or commanding the nth derivative of position, you generally have boundary conditions on the first order, the second order, the third order, all the way up to the n-1th derivative.

3.7.1 <u>Calculus of Variation</u>

Calculus of variations refers to a body of literature in mathematics that deals with a very simple problem. [13]

$$x^*(t) = arg \underbrace{min}_{x(t)} \int_0^T \boldsymbol{\mathcal{L}}(\dot{x}, x, t)dt \tag{3.24}$$

In this equation, you're trying to minimize an integral. You're trying to find the best function $x(t)$, that minimizes a function. The function is the integral of L, which depends on the function $x(t)$, its derivative, $\dot{x}(t)$, and of course, time.

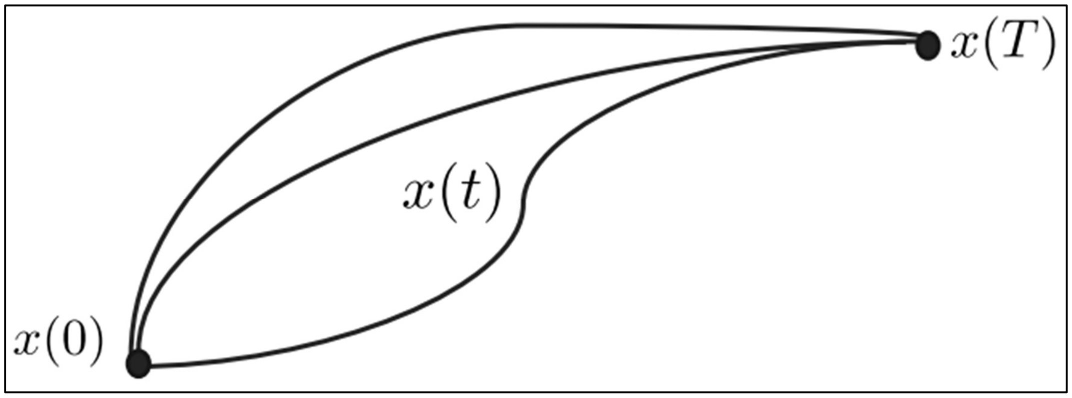


Fig 3.e some of the possible differentiable curves x(t), with a given x(0) & x(T).

In our case, we'll be interested in connecting two given points, x specified at time 0 and x specified at time T. We're looking for a trajectory, in fact, the best trajectory, and we restrict ourselves to differentiable curves, and there are, of course, many such differentiable curves. And again, we want to find something that minimizes a suitable function [13]

$$x^*(t) = arg \underbrace{min}_{x(t)} \int_0^T \dot{x}^2 dt \tag{3.25}$$

Independent of the function, the optimal curve, must satisfy what are called Euler Lagrange equations. These are necessary conditions that must be satisfied by this optimal function. [13]

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0 \tag{3.26}$$

The Euler Lagrange equations involve partial derivatives and total derivatives. So, you take the partial derivative of L with respect to $\dot{x}$ and the partial derivative of L with respect to x. You also need the total derivative with respect to time. On solving the PDE equation (3.26, we get

$$\boldsymbol{\mathcal{L}}(\dot{x}, x, t) = (\dot{x})^2 \text{ è } \ddot{x} = 0$$

$$x = c_1 t + c_0 \tag{3.27}$$

The resulting equation is simply the equation of a straight line. This is the case when you have a first order system with the input being the velocity, and it works for robots that have a kinematic model. Again, a kinematic model is one where you can specify the velocity, you can command the velocity.

In general, the system might be characterized by higher order dynamics. So, for an nth order system, you might not be able to specify the velocity or command the velocity. The system has inertia, and you can't willingly specify velocities. For an nth order system, the input is the nth derivative of position. [13]

$$x^*(t) = arg \underbrace{min}_{x(t)} \int_0^T \left(x^{(n)}\right)^2 dt \tag{3.28}$$

In this setting, the Euler Lagrange equation looks a little more complicated. But once again, it involves partial and total derivatives of L. [13]

$$x^*(t) = arg \underbrace{min}_{x(t)} \int_0^T \boldsymbol{\mathcal{L}}\left(x^{(n)}, x^{(n-1)}, \ldots, \dot{x}, x, t\right) dt \tag{3.29}$$

Necessary condition satisfied by the "optimal" function [13]

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \frac{d^2}{dt^2}\left(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\right) + \cdots + (-1)^n \frac{d^n}{dt^n}\left(\frac{\partial \mathcal{L}}{\partial x^{(n)}}\right) = 0 \tag{3.30}$$

If you take n=1, as we've seen, we get the shortest distance curve, the straight line. n=2 corresponds to minimizing a functional that involves the square of the acceleration. n=3 involves minimizing the square of the jerk. n=4 involves minimizing the snap. Accordingly, we get the minimum velocity trajectory, the minimum acceleration trajectory, the minimum jerk trajectory, and the minimum snap trajectory.

3.7.2 <u>Minimum Jerk Trajectory</u>

Now to look at a concrete setting, let's discuss the problem of designing a minimum jerk trajectory, which again, makes sense if you have a third order system. And let's assume that we're specified the starting position x(0) = a, and the end position x(T) = b.

Our function is simply the integral of the square of the jerk and we want to find the best $x(t)$, that minimizes this functional [13].

$$x^*(t) = arg\underbrace{min}_{x(t)} \int_0^T \boldsymbol{\mathcal{L}}(\dddot{x}, \ddot{x}, \dot{x}, x, t)dt \tag{3.31}$$

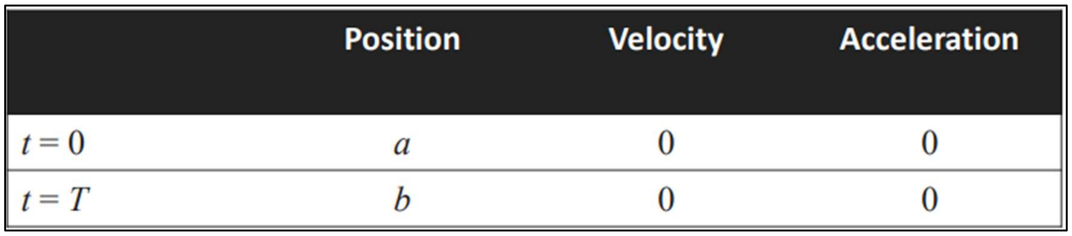
$$\boldsymbol{\mathcal{L}} = (\dddot{x})^2 \tag{3.32}$$

We write down the Euler Lagrange equations and we see that a whole bunch of terms just disappear because they don't explicitly depend on x triple dot. What remains can be simplified into the sixth order differential equation [13].

$$\begin{gathered}\underbrace{\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \frac{d^2}{dt^2}\left(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\right)}_{0} - \frac{d^3}{dt^3}\left(\frac{\partial \mathcal{L}}{\partial x^{(3)}}\right) = 0 \\ x^{(6)} = 0 \\ x = c_5t^5 + c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0\end{gathered} \tag{3.33}$$

Which then can be solved analytically and the result is this fifth order polynomial in time. What remains is the sub problem of solving for the coefficients in this fifth order polynomial. There are six such coefficients. In order to solve for these, you need to specify additional boundary conditions. We've assumed that we know the position at the start position and the end position. Now let's also assume that we know the velocities and, in this case, I'll just assume that these velocities are zero. In addition, let's assume we know the accelerations, and again assuming these accelerations are zero.

Table 3.2 Boundary Conditions



	Position	Velocity	Acceleration
$t = 0$	a	0	0
$t = T$	b	0	0

Each of these boundary conditions gives you an equation. You can write down six equations in terms of the unknown constants and in terms of the known boundary conditions.

$$\begin{bmatrix} a \\ b \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} \tag{3.34}$$

Solving for these constants is now a linear problem.

Until now we have discussed the entire concept in respect to single dimension. This basic idea can be extended to multiple dimensions.

If you're trying to generate trajectories in the x, y plane, then you end up at two Euler Lagrange equations. One for the x direction and one for the y direction. [13]

$$\left(x^*(t), y^*(t)\right) = arg \underbrace{min}_{x(t),y(t)} \int_0^T \boldsymbol{\mathcal{L}}(\dot{x}, \dot{y}, x, y, t)dt \tag{3.35}$$

Necessary condition satisfied by the optimal function

$$\begin{gathered} \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0 \\ \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}}\right) - \frac{\partial \mathcal{L}}{\partial y} = 0 \end{gathered} \tag{3.36}$$

CHAPTER 4
MATLAB® AND SIMULINK® IMPLEMENTATION

The Simulink® based quadcopter simulation is intended to reduce cost and, running a simulation will represent dynamic performance of your vehicle. The simulation is intended to act as a sort of starter kit to help accelerate you towards the goals of speeding up the process of system design and provide one with tools to explore quadcopter control design techniques. The simulation represents a synthesis of several author's approaches to modelling the behaviour of a quadcopter. Many important effects (most obviously aerodynamic effects such as blade flapping) are ignored or dramatically simplified, so one will need to make sure to understand the limitations of the model in order to get reliable service out of it. It is written in Simulink® and MATLAB® code.

4.1 <u>Simulation in MATLAB®:</u>

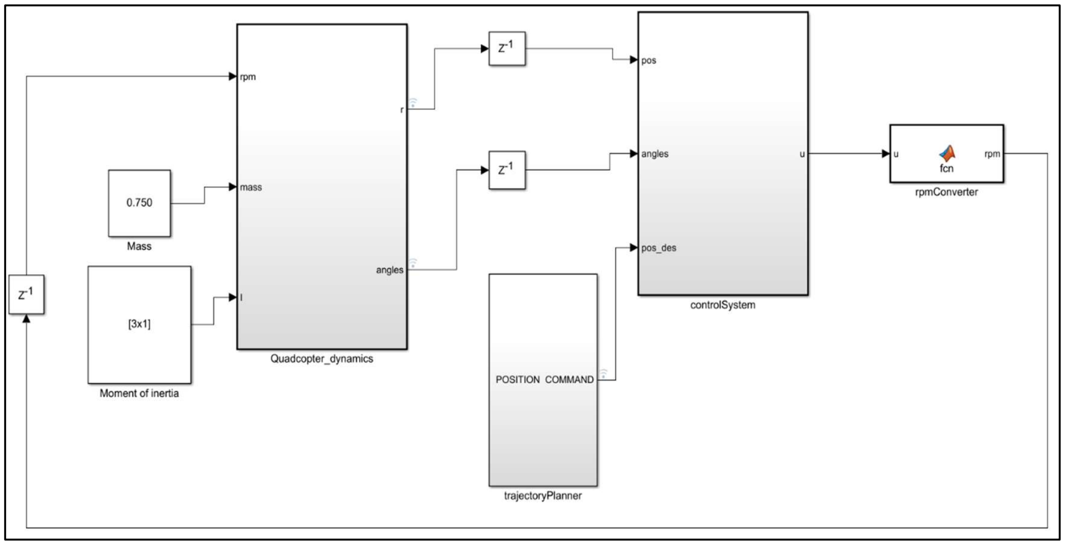


Fig 4.a Implementation of Quadcopter in SIMULINK®

We implemented the model in MATLAB® to verify the working. This consists of 3 major subsystems namely trajectory planner, controller and quadcopter dynamics.

4.1.1 <u>Quadcopter Dynamics</u>

In the quadcopter dynamics, we used mathematical model discussed in section 2.3. Here we considered angular velocities and position of the quadcopter as inputs. Angular velocities are the required inputs for the system. From this subsystem we get the angles and position of the quadcopter as the measured values. These measured values are given to the input of the control system block along with the desired attitude and position.

Firstly, from the angular velocities we find thrust and torque by using equation (2.6. Then, giving these as inputs for model (Section 2.3) we will get angles and position (integration need to done for it).

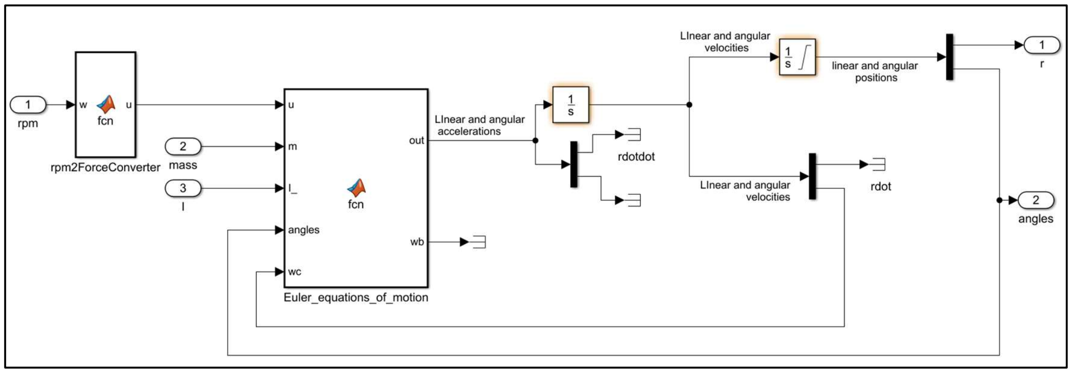

1
rpm
w
u
fcn
rpm2ForceConverter
2
mass
3
l
u
m
l_
angles
wc
out
wb
fcn
Euler_equations_of_motion
LInear and angular accelerations
1/s
rdotdot
LInear and angular velocities
1/s
linear and angular positions
LInear and angular velocities
rdot
1
r
2
angles

Fig 4.b Inside of Quadcopter Dynamics block

4.1.2 <u>Control System</u>

The control system block consists of those two controllers i.e., position control and attitude control. Attitude control is the inner loop which control the orientation of the quadcopter. Position control is the outer loop which provide required attitude to the inner loop i.e., attitude control.

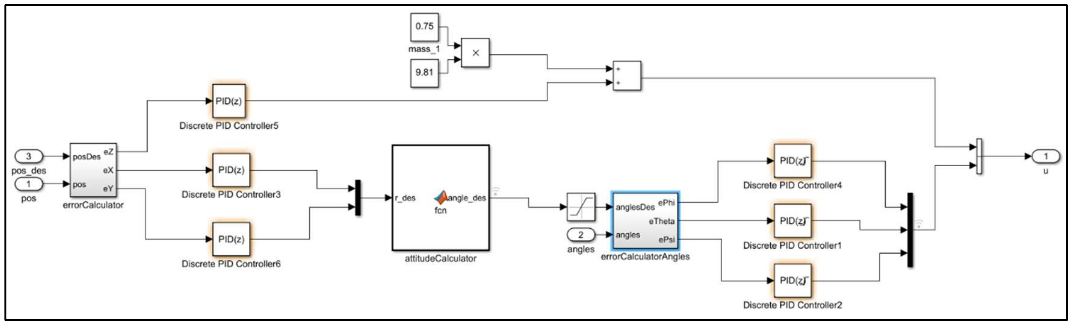


Fig 4.c Inside of Control Systems block

4.1.3 Trajectory Planner

Third block, trajectory planner, gives the desired position of the quadcopter. Calculations for the constants in equation (3.33 is done in the workspace and imported to main code and using those we calculate the intermediate points. These points are given to controller as desired position for quadcopter.

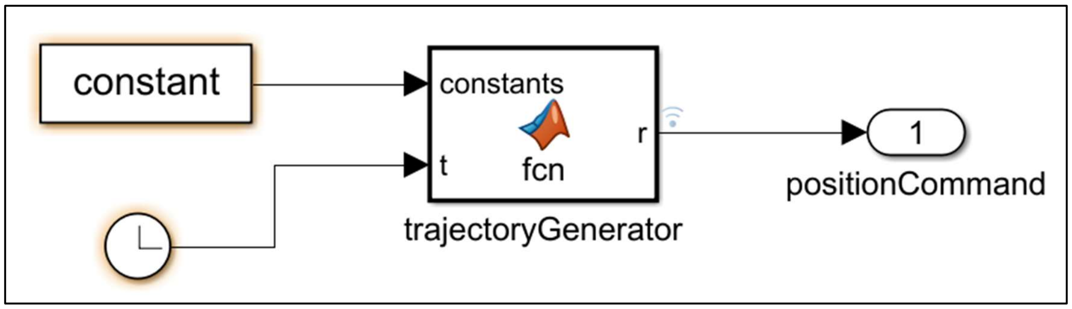


Fig 4.d Inside of Trajectory Planner block

4.2 Results

For the waypoints [(0,0,1), (0,0,2), (0,1,2), (1,2,3), (1,1,2), (0,0,2), (0,0,0)]

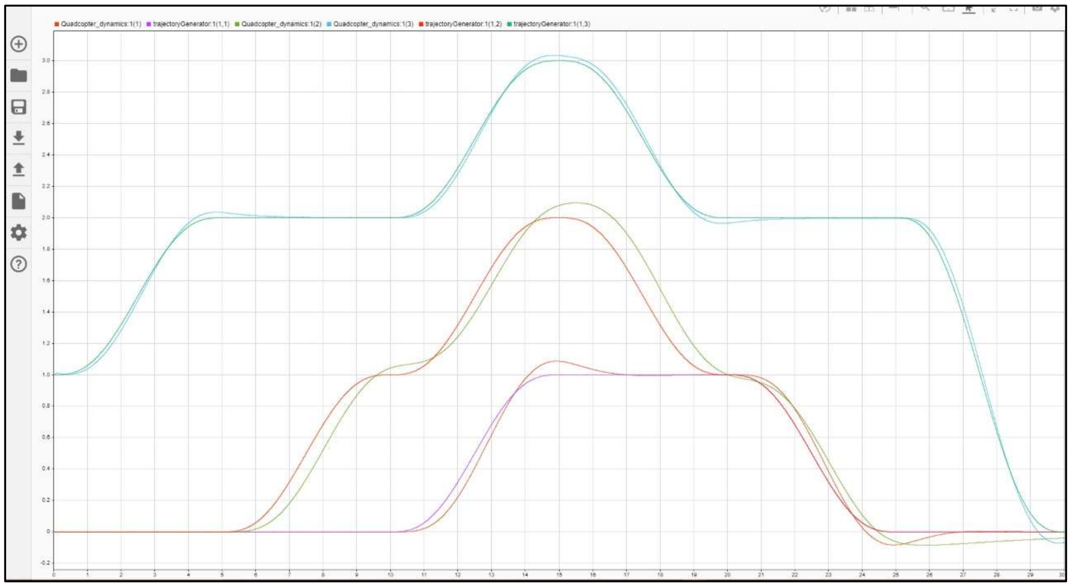


Fig 4.e Trajectory Tracking for the given trajectory as input

First, we generated the minimum jerk trajectory for the points [(0,0,1), (0,0,2), (0,1,2), (1,2,3), (1,1,2), (0,0,2), (0,0,0)]. The trajectory is then fed as reference trajectory for the position controller. The result can be seen in Fig 4.e, where we can see that the quadcopter following the reference trajectory closely with some overshoots.

As we discussed, we used minimum jerk trajectory(n=3), so according to section 3.7 we have up to (n-1)th order boundary conditions i.e., position, velocity and acceleration. We can see the tracking of acceleration in all 3 directions in Fig 4.f, Fig 4.g and Fig 4.h.

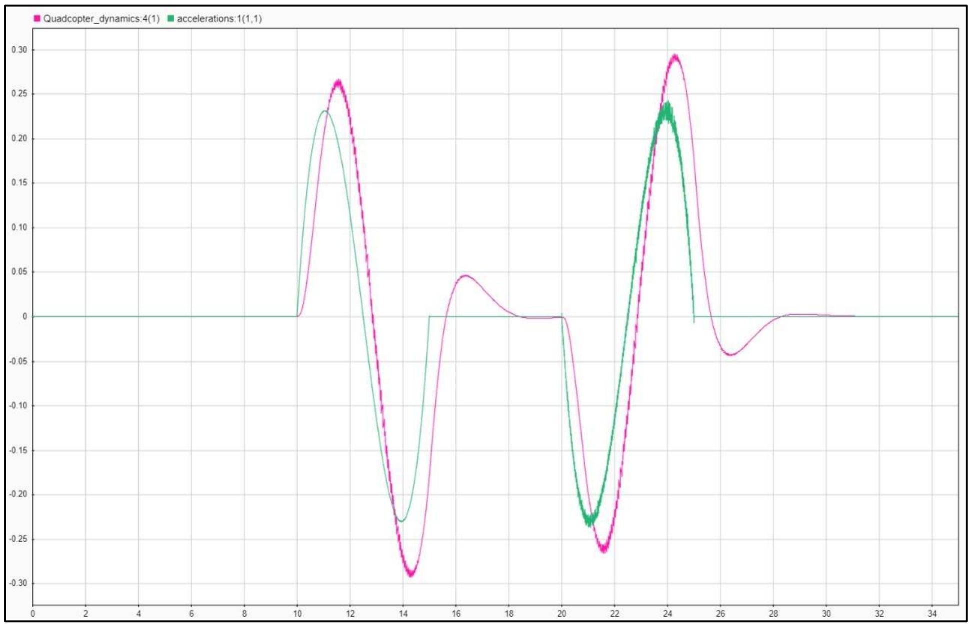


Fig 4.f Linear acceleration in X-Direction

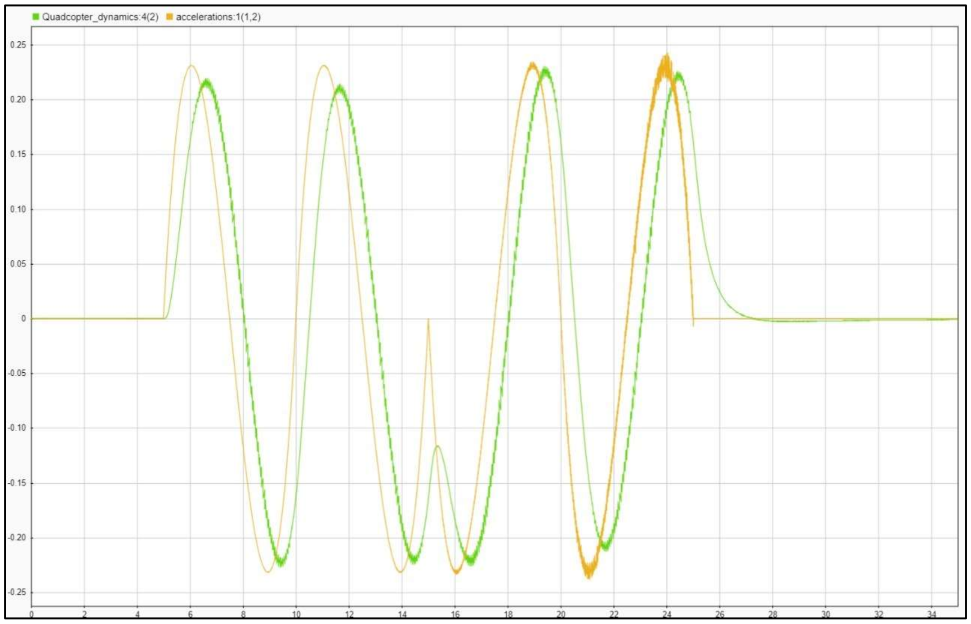


Fig 4.g Linear acceleration in Y-Direction

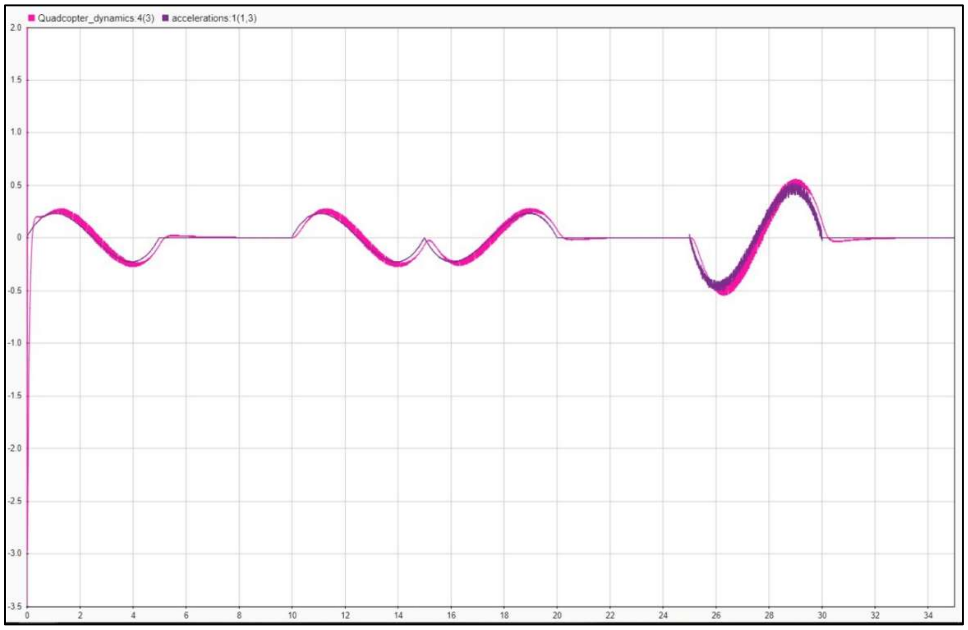


Fig 4.h Linear acceleration in Z-Direction

We can see the tracking of velocities in all 3 directions in Fig 4.i, Fig 4.j and Fig 4.k.

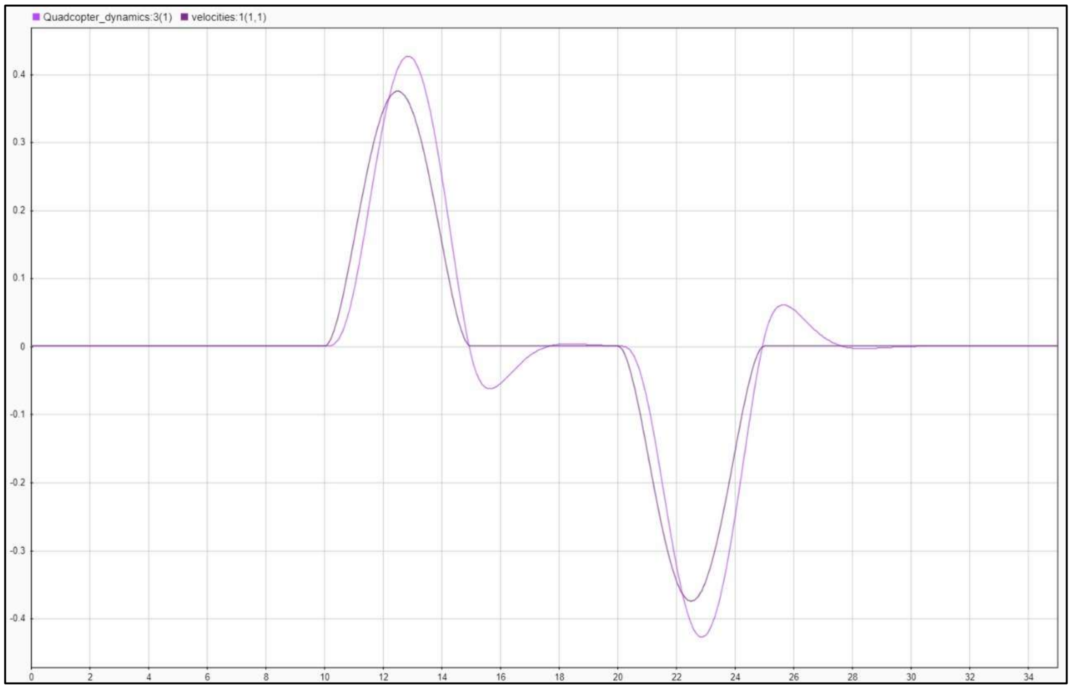


Fig 4.i Linear Velocity in X-Direction

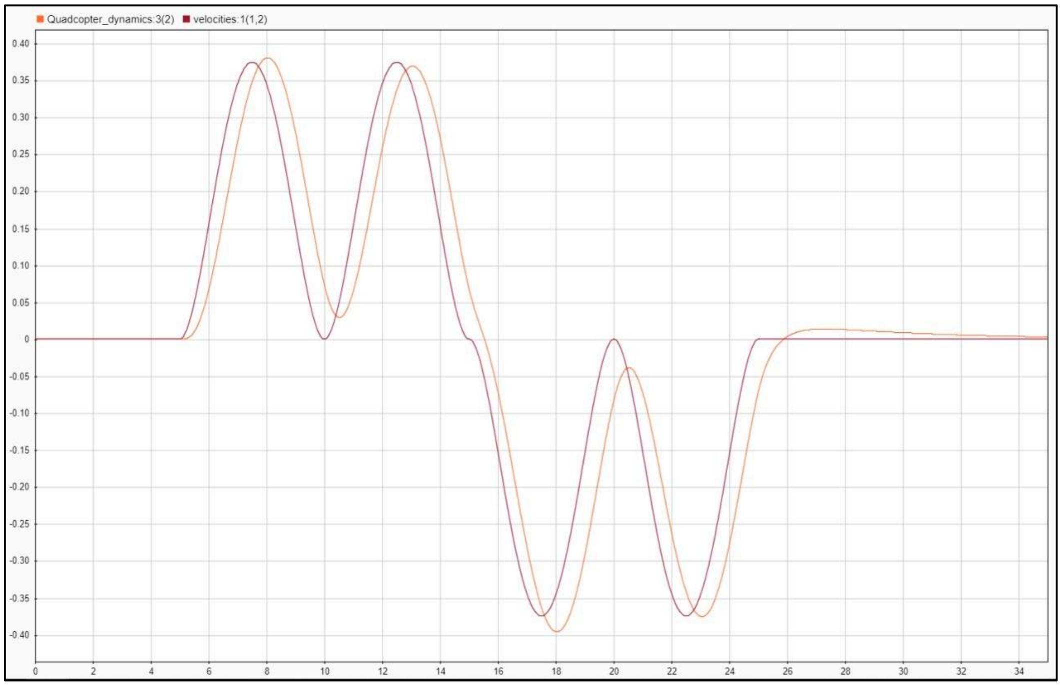


Fig 4.j Linear Velocity in Y-Direction

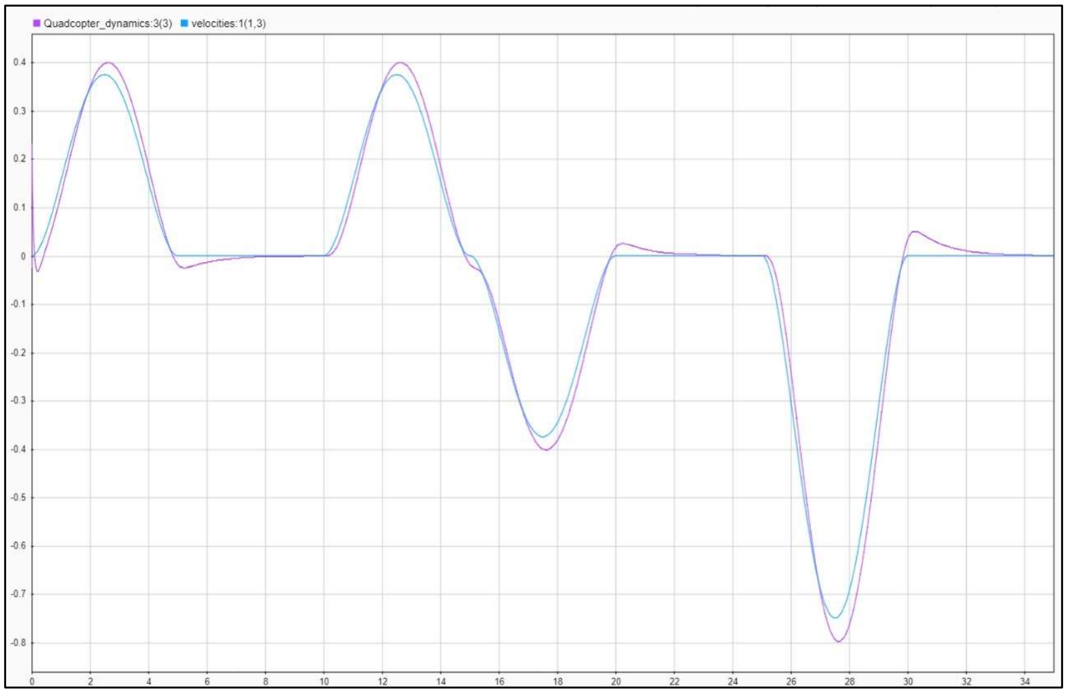


Fig 4.k Linear Velocity in Z-Direction

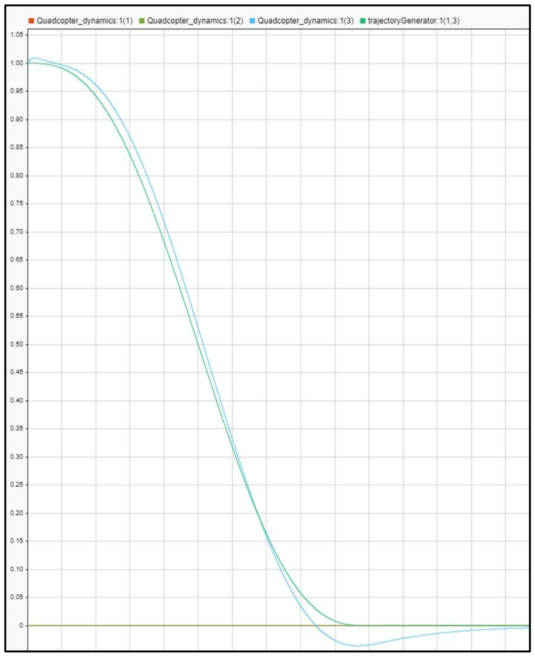


Fig 4.l : Quadcopter going to (0,0,0) from (0,0,1)

In Fig 4.l, quadcopter is moving from (0,0,1) to (0,0,0) i.e., landing operation.

CHAPTER 5

PROPOSED HARDWARE DESIGN

5.1 BLDC motor

As their name implies, brushless DC motors do not use brushes. With brushed motors, the brushes deliver current through the commutator into the coils on the rotor. So how Does a brushless motor pass current to the rotor coils? It doesn't because the coils are not located on the rotor. Instead, the rotor is a permanent magnet; the coils do not rotate, but are instead fixed in place on the stator. Because the coils do not move, there is no need for brushes and a commutator. With the brushed motor, rotation is achieved by controlling the magnetic fields generated by the coils on the rotor, while the magnetic field generated by the stationary magnets remain fixed. To change the rotation speed, you change the voltage for the coils. With a BLDC motor, it is the permanent magnet that rotates; rotation is achieved by changing the direction of the magnetic fields generated by the surrounding stationary coils. To control the rotation, you adjust the magnitude and direction of the current into these coils.

Advantages of BLDC Motors

A BLDC motor with three coils on the stator will have six electrical wires (two to each coil) extending from these coils. In most implementations three of these wires will be connected internally, with the three remaining wires extending from the motor body (in contrast to the two wires extending from the brushed motor described earlier). Wiring in the BLDC motor case is more complicated than simply connecting the power cells positive and negative terminals; we will look more closely at how these motors work in the second session of this series. Below, we conclude by looking at the advantages of BLDC motors.

One big advantage is efficiency, as these motors can control continuously at maximum rotational force (torque). Brushed motors, in contrast, reach maximum torque at only certain points in the rotation. For a brushed motor to deliver the same torque as a brushless model, it would need to use larger magnets. This is why even small BLDC motors can deliver considerable power.

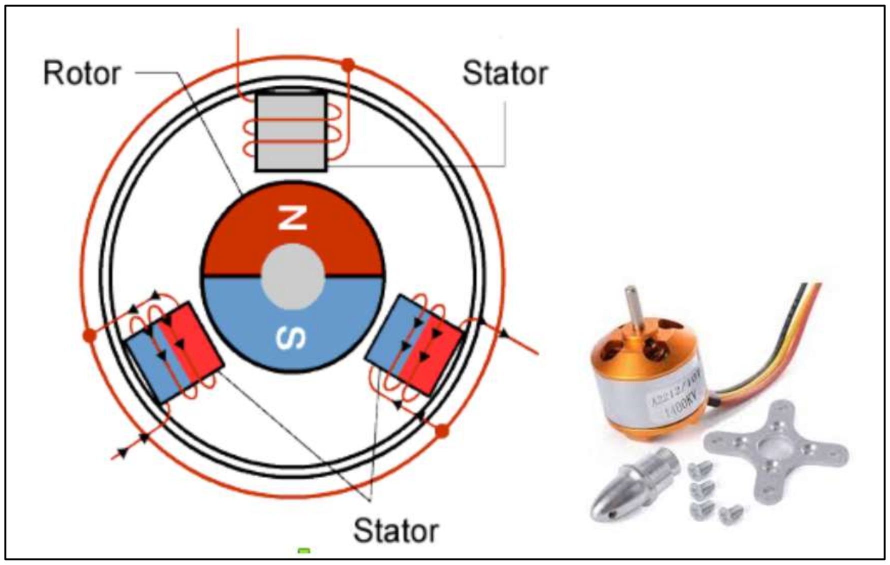


Fig 5.a : BLDC motor [14]

The second big advantage is controllability. BLDC motors can be controlled, using feedback mechanisms, to delivery precisely the desired torque and rotation speed. Precision control in turn reduces energy consumption and heat generation, and in cases where motors are battery powered lengthens the battery life.

BLDC motors also offer high durability and low electric noise generation, thanks to the lack of brushes. With brushed motors, the brushes and commutator wear down as a result of continuous moving contact, and also produce sparks where contact is made. Electrical noise, in particular, is the result of the strong sparks that tend to occur at the areas where the brushes pass over the gaps in the commutator.

5.2 <u>ESC (Electronic Speed Controls)</u>

An Electronic speed controller is an electronic circuit that acts as the interface between the pilot’s commands and the individual drone motors. There are several types of ESCs in the market, but brushless motors require a 3-phase ESC these are easily distinguished by the presence of three soldering pads that are meant to connect to the three motor phases of a brush less motor.

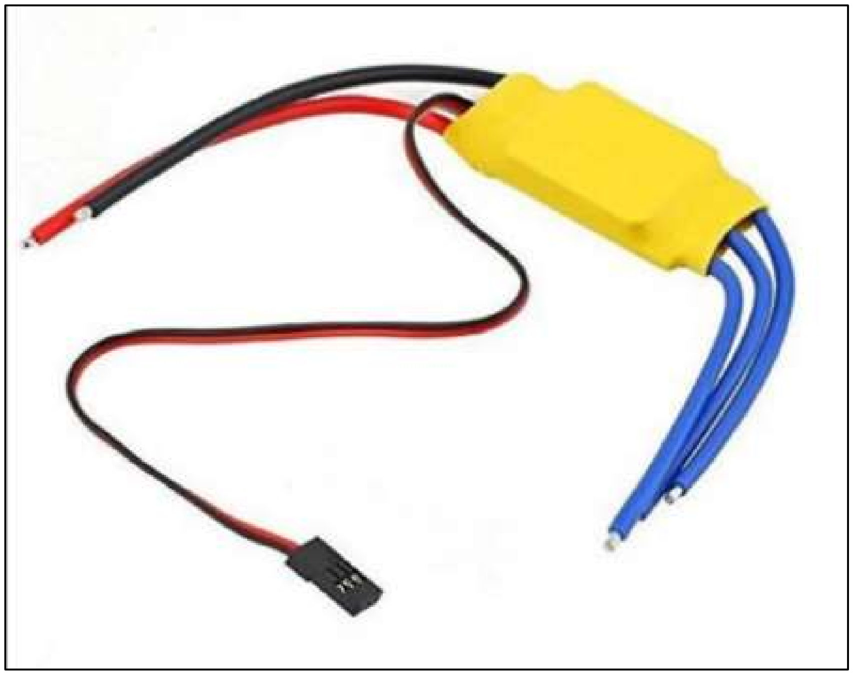


Fig 5.b : ESC [15]

Features

- This 30-amp ESC for BLDC has on board BEC.
- This ESC comes with a male dean T-connector.
- Auto Low BATTERY Slow down at 3.0V/cell LiPo, cut-off at 2.9V/cell LiPo.
- APPLICATION: Multirotor, RC Airplanes, etc.

5.3 Sensors

5.3.1 Inertial Measurement Unit:

Inertial Measurement sensors are widely used for acquiring orientations of required systems. IMU sensors will be working as feedback systems for Attitude of Quadcopter. BNO055 and MPU9250 are some of the IMUs used widely. These sensors contain 3 axis accelerometer, 3 axis gyroscope and 3 axis magnetometers. Accelerometer gives us the details about linear accelerations, Gyroscope gives us the angular velocities and Magnetometer gives us the magnetic heading. The main advantage using BNO055 is that it has an M0-cortex processor onboard which can be used for doing necessary calculations and filtering which will reduce computational work on the main processor.

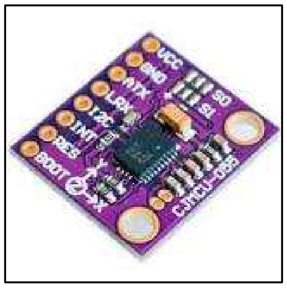


Fig 5.c : BNO055 9dof IMU [16]

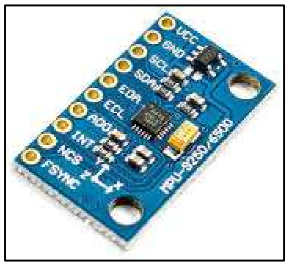


Fig 5.d : MPU 9250 9dof IMU [17]

Accelerometers and Gyroscopes are prone to noises individually which make them unreliable when only one of them is used. Accelerometers are prone to high frequency noise whereas Gyroscopes are prone to low frequency noise. Due to this reason these sensors compensate the problem in the other sensor when used simultaneously. We can use complementary filter, Mahony filter, etc., to fuse the readings of both the sensors and get more reliable readings.

We can interface these sensors with the main controller using I2C communication. [16, 17]

5.3.2 TFmini LIDAR:

TFmini is a mini-LiDAR (Light Detection and Ranging) module. It is mainly capable of the function of real-time and contactless distance measurement and is featured by accurate, stable, and high-speed distance measurement [18]. TFmini is based on TOF, namely, Time of Flight principle. To be specific, the product transmits modulation wave of near infrared ray on a periodic basis, which wave will reflect after contacting object. The product obtains time of flight by measuring round-trip phase difference and then calculates relative range between the product and the detection object, as shown in Fig 5.e.

We can interface this sensor with the main controller using UART communication.

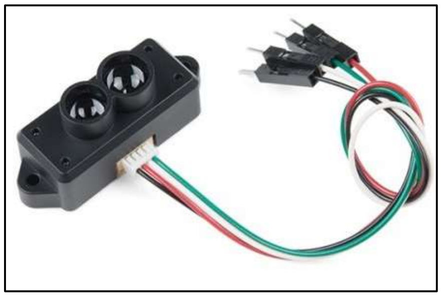


Fig 5.e : Tfmini LIDAR Sensor [18]

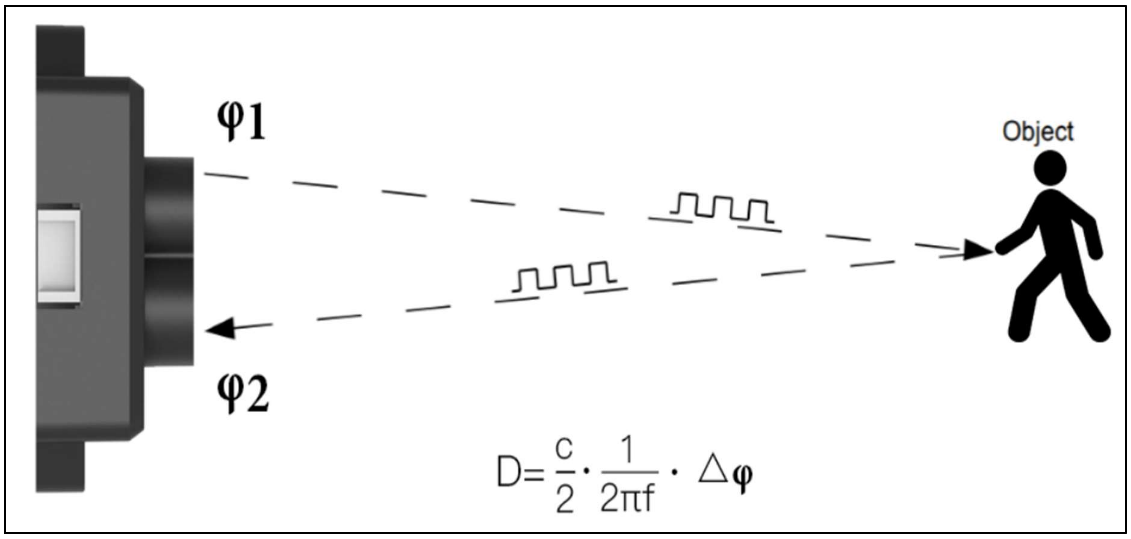


Fig 5.f : Schematics of ToF principle [18]

Table 5.1 : Key Characteristic Parameters of TFmini [18]

Description	**Parameter Value**
Operating Range (Indoor)	0.3m~12m
Measurement accuracy	±4cm@（0.3-6m) ±6cm@（6m-12m）
Default unit of distance	cm
Range resolution	5mm
Receiving half angle	1.15°
Transmitting half angle	1.5°
Frequency	100 Hz

5.3.3 BMP 180:

The BMP180 is the function compatible successor of the BMP085, a new generation of high precision digital pressure sensors for consumer applications.

The ultra-low power, low voltage electronics of the BMP180 is optimized for use in mobile phones, PDAs, GPS navigation devices and outdoor equipment. With a low altitude noise of merely 0.25m at fast conversion time, the BMP180 offers superior performance. The I2C interface allows for easy system integration with a microcontroller [19].

The BMP180 is based on piezo-resistive technology for EMC robustness, high accuracy and linearity as well as long term stability.

Robert Bosch is the world market leader for pressure sensors in automotive applications. Based on the experience of over 400 million pressure sensors in the field, the BMP180 continues a new generation of micro-machined pressure sensors [19].

Fig 5.g : BMP 180 [19]

We can interface this sensor with the main controller using I2C communication.

5.3.4 GPS:

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high-performance u-blox 6 positioning engines. These flexible and cost-effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package [20]. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints. The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppress jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments [20].

Table 5.2 : Key Characteristic Parameters of Neo-6 GPS [21]

Receiver type	50 channels, GPS L1(1575.42Mhz) C/A code, SBAS: WAAS/EGNOS/MSAS
Horizontal position accuracy	2.5mCEP (SBAS:2.0mCEP)
Navigation update rate	5Hz maximum (1HZ default)
Capture time	Cool start: 27s (fastest)；Hot start: 1s
Tracking & Navigation sensitivity	-161dBm
Communication protocol	NMEA (default)/UBX Binary
Serial baud rate	4800, 9600(default), 19200, 38400, 57600, 115200, 230400
Operating temperature	-40℃ ~ 85℃
Operating voltage	2.7V~5.0V (power supply input via VCC)
Operating current	45mA
TXD/RXD impedance	510Ohms

5.4 <u>Microcontroller</u>

The STM32 Nucleo-144 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode.

The ST Zio connector, which extends the ARDUINO® Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields. The STM32 Nucleo-144 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.

The STM32 Nucleo-144 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

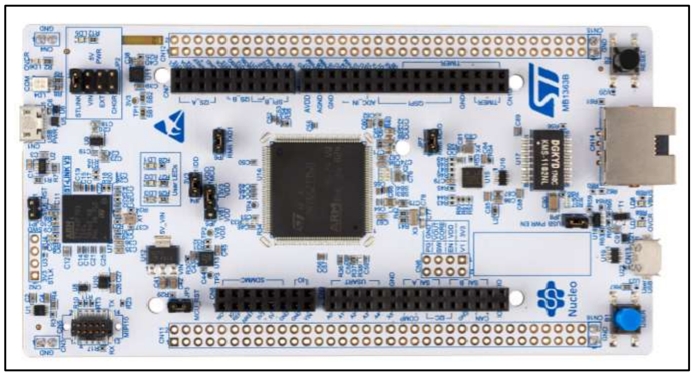


Fig 5.h : STM32 nucleo board [22]

Features:

- Common features
  - STM32 microcontroller in LQFP144 package
  - 3 user LEDs
  - 2 user and reset push-buttons
  - 32.768 kHz crystal oscillator
  - Board connectors:
    - SWD
    - ST Zio expansion connector including ARDUINO® Uno V3
    - ST morpho expansion connector
  - Flexible power-supply options: ST-LINK, USB VBUS or external sources.
  - On-board ST-LINK debugger/programmer with USB re-enumeration capability: mass storage, Virtual COM port, and debug port
  - Comprehensive free software libraries and examples available with the STM32Cube MCU Package
  - Support of a wide choice of Integrated Development Environments (IDEs) including IAR™, Keil®, and STM32CubeIDE
- Board-specific features

- External or internal SMPS to generate V_{core} logic supply
- Ethernet compliant with IEEE-802.3-2002
- USB OTG full speed or device only
- Board connectors:
  - USB with Micro-AB or USB Type-C™
  - Ethernet RJ45
- Arm® Mbed Enabled™ compliant

5.5 Battery

Battery which we are going to use is LIPO (Lithium Polymer).

Specifications:

- Model No: ORANGE 2200/3S-30C
- Weight: 175.0g
- Voltage: 11.1V
- Dimensions: 23x34x106(mm)
- Max Continuous Discharge: 30C (66.0A)
- Balance Plug: JST-XH
- Max Burst Discharge: 60C (132.0A)
- Discharge Plug: XT-60
- Charge Rate: 1-3C Recommended, 5C Max

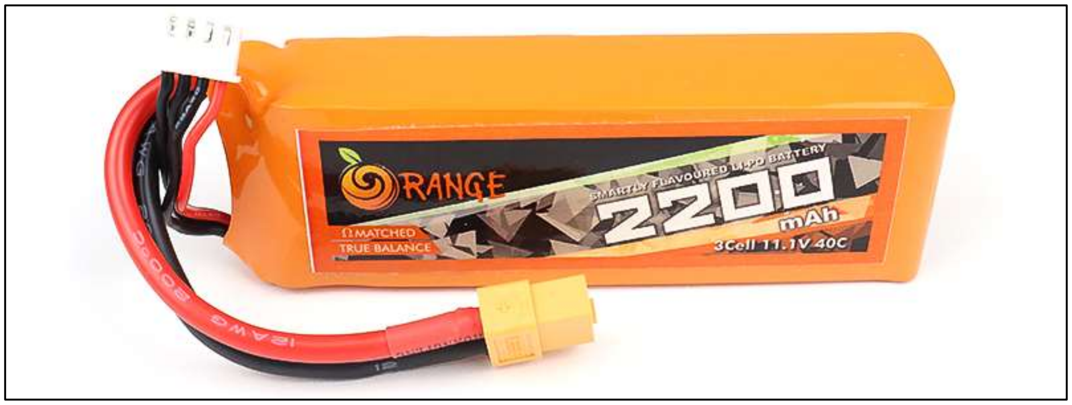


Fig 5.i : LIPO Battery

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

In this project we demonstrated the path tracking using quadcopter. We initially developed the mathematical model for the quadcopter and developed the attitude controller for controlling the orientation of the quadcopter. Then studied and developed the position controller. Afterwards learnt about trajectory which is the combination of several points and generated the path using the given points. Then using these inner and outer control loops we tried to track the generated the path. The simulation results shows that the controller functioning was fine with little disturbances.

We can extend this work further and make more robust controllers and obstacle avoidance during the path tracking in the external environment. Using GPS positions as the points we can generate the path and use that for delivery purposes. These quadcopters are used for surveillance in defense and in any religious gatherings etc., And these can be used during natural calamities to find trapped citizens.

BIBLIOGRAPHY

[1] J. Leishman., "The Breguet-Richet Quad-Rotor Helicopter of 1907," no. 3, pp. 1-4, 2001.

[2] S. Shen, Y. Mulgaonkar, N. Michael and V. Kumar, Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor, RSS, 2013.

[3] L. D. Dhriti Raj Borah, "A review on Quadcopter Surveillance and Control," *ADBU-Journal of Engineering Technology,* vol. 4(1), pp. 116-119, 2016.

[4] N. Johnson-Laird, "Basic Physics of Drones".

[5] *Mathematical Model of Quadcopter,* Quad Sim Master.

[6] H. t. M. N. ElKholy, Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches, 2014.

[7] A. A. M. M. B. Jeemol Mariya Alias, "Design Of Different Controllers In Quadcopter," *IOSR Journal of Engineering (IOSRJEN),* pp. 61-72, 2018.

[8] F. SABATINO, Quadrotor control: modeling, nonlinear control design, and simulation, Sweden, 2015.

[9] J. M. B. Domingues, Quadrotor prototype, 2009.

[10] S. A. G. A. W. Jinho Kim, "A Comprehensive Survey of Control Strategies for," *arXiv,* 2020.

[11] W. Daniel, "Trajectory Generation and Control for Quadrotors," *Publicly Accessible Penn Dissertations.,* 2012.

[12] H. A. Kiam, C. Gregory and L. Yun, "PID Control System Analysis, Design and Technology," IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, JULY 2005.

[13] V. Kumar, "Aerial Robotics," Coursera, Pennsylvania.

[14] *A2212/13T TECHNICAL.*

[15] *30A BLDC ESC datasheet.*

[16] *BNO055 Intelligent 9-axis absolute orientation sensor,* Bosch Sensortec .

[17] *MPU-9250 Register Map and Descriptions,* InvenSense , 01/07/2015.

[18] *Product Manual of TFmini-Mini LiDAR Module,* Benewake (Beijing) Co. Ltd..

[19] *BMP180 Digital pressure sensor data sheet,* Bosch Sensortec, may, 2015.

[20] *NEO-6 u-blox 6 gps modules datasheet,* U-blox, 2011.

[21] *UART GPS NEO-6M User Manual,* Waveshare.

[22] *STM32 Nucleo-144 boards,* ST microelectronics, 2020.