REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

FAR DISTANCE UNMANNED AERIAL VEHICLES CONTROL AND OBJECT DETECTION USING INTERNET OF THINGS NETWORK AND EMBEDED SYSTEMS

FAHAD YASSIR AL BAZZAZ

MSc. THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM OF COMPUTER ENGINEERING

ADVISOR
PROF. DR. HASAN HÜSEYİN BALIK

İSTANBUL, 2017

CHAPTER 1

INTRODUCTION

The Internet presents a unique interconnected system which enables devices to communicate globally using set of standard protocols and connecting various heterogeneous networks - academically, business, governments etc. In the first years, the Internet was represented by static web sites and email communication. Nowadays, different forms of Internet implementation could be seen everywhere, part of many different aspects of our lives providing plenty of services and applications, and trying to meet each user's needs no matter of time and place. The main "secret" is hidden behind the digitalization of the user and all of the user-friendly and automated mechanisms.

Internet of Things represents a general concept for the facility of network contrivances to sense and accumulate data from the world around us, and then share that data across the Internet where it can be processed and utilized for several motivating purposes.

1.1 Literature Review

The Internet of Things is an already known term nowadays and it is becoming bigger and bigger overtime with all sorts of sensors and systems being developed to help people ease up their lives. The number of connected devices continues to grow worldwide but also the diversity and the applications in the real world are immense, making it an appealing industry to work on [1].

A lot of companies are working on new devices and new solutions to deal with the growth of the connected devices, it is a massive growing industry. It possesses the power to transform every single environment such as agriculture, transportation, manufacturing, smart houses even entire cities. Companies are working on making new devices for every possible scenario but also improving communication protocols as well as security. It is

estimated that around 2015 10 billion devices were connected and by 2020 will be connected 20 to 30 billion devices, as costs continue to drop and demand continues to grow [2].

With today's advancements in technology, it became possible to address a larger number of problems regarding accessibility and also monitoring and interacting more thoroughly with systems. The capabilities of the system is based on the usage of unmanned surface vehicle (USV) that help son maritime monitoring tasks using multidrone swarms [3] for extended spatial resolution. The system may detect such as cleaning oil spill [4], or for general environmental condition monitoring.

The Internet of Things help to solve a lot of problems in different fields: in smart cities, IoT applications are related with parking issues, noise, traffic, illumination monitoring [5]; emergency systems for earthquakes [6]; precision agriculture applications in culture process optimization [7]. IoT are used to deliver information from the sensors and to the actuators.

The near future, millions of unmanned aerial vehicles (UAVs), also known as drones, are expected to be rapidly deployed in diverse sectors of our daily life performing wide-ranging activities from delivering a package to diving into water for a specific underwater operation [8].

Regarding their utilization, UAVs' applications can be broadly divided into civilian and military models. The former can be utilized for governmental or nongovernmental purposes; e.g., employing UAVs in rescue operations to recover from large-scale disaster events, such as the great East Japan earthquake [9], the natural disasters of Indonesia [10], and the earthquake of Nepal [11].

However, in the near future, drones will be used not only for public protection and disaster relief operations [12], [13] but also for many other civilian, commercial and governmental services. Some good examples are surveillance and reconnaissance [14], public safety [15], homeland security [16], [17], forest fire monitoring [18], environmental monitoring [19], security and border surveillance [12], farming [13], or even Internet delivery [20], [21], architecture surveillance [22], goods transportations [23], [24] such as Amazon Prime Air [25] designed to safely deliver packages to customers within 30 minutes using small drones. With their countless applications, UAVs will soon be influentially a part of our daily life; a necessary technology similar to today's smartphones.

Moreover, interestingly from the technology perspective, UAVs are foreseen as an important component of an advanced cyber-physical Internet of Things (IoT) ecosystem [26]. Based on the definition, IoT aims at enabling things to be connected anytime, anywhere ideally using any network and providing any service. The IoT concept allows UAVs to become an integral part of IoT infrastructure. This is due to the fact that UAVs possess unique characteristics in being dynamic, easy-to-deploy, easy-to-reprogram during run-time, capable of measuring anything anywhere, and capable of flying in a controlled airspace with a high degree of autonomy [27].

1.2 Objective of Thesis

This thesis proposes a case study for designing a system that uses an Internet of Things (IoT) network to make the control for Unmanned Aerial Vehicle (drone) from an infant distance by simulating the signals coming from RF receiver and remote control.

These signals will be provided from raspberry pi device to apply the signals of **Roll**, **Pitch**, **Yaw**, and **Throttle** to take control of UAV. The system uses an interaction between raspberry pi and flight controller to overcome the complexity of stability and PID control calculations.

The control is designed using python 2.7 GUI (Graphical User Interface) to take control for flight, also the design is containing the capability of tracking objects based on the BGR colour moment calculations.

1.3 Original Contribution

This thesis proposes a case study for designing a system that uses an Internet of Things (IoT) to make the control for Unmanned Aerial Vehicle (drone) from an infant distance by simulating the signals coming from RF receiver and remote control, these signals will be provided from raspberry pi device to apply the signals of **Roll**, **Pitch**, **Yaw**, and **Throttle** to take control of UAV, the system uses an interaction between raspberry pi and flight controller to overcome the complexity of stability and PID control calculations. The control is designed using python 2.7 GUI (Graphical User Interface) to take control for flight, also the design is containing the capability of tracking objects based on the BGR color moment calculations.

1.4 Background for the Internet of Things

The most vital part of achieving IoT is communication, because in order to interconnect different devices, they must be able to communicate. All other properties, such as sensing, maneuvering, being able to capture, store, and process data are unnecessary; unless the device specifically requires one of these properties. However, the ability to communicate is essential when labelling a device as an IoT device. How this communication is performed is less important, since the actual physical and link layer communication within IoT can be realized in many ways.

Case C in Figure (1.1) shows that devices are not always required to communicate through a communication network. For example, if two devices are close to each other it might be simpler to directly communicate via for example radio using technologies such as Bluetooth or ZigBee (protocols which both enable direct communication). In contrast, in Case A in Figure (1.1) a device might communicate via a gateway using one protocol (such a IPv6 over Low power Wireless Personal Area Networks (6LoWPAN)) and then the gateway could communicate using another protocol (e.g. IPv4) over a communication network such as the Internet. Case B in Figure (1.1) illustrates two devices which are directly communicating with one another without requiring a gateway where both devices are directly connected to the communication network and thus are able to communicate even if they are located in different places.

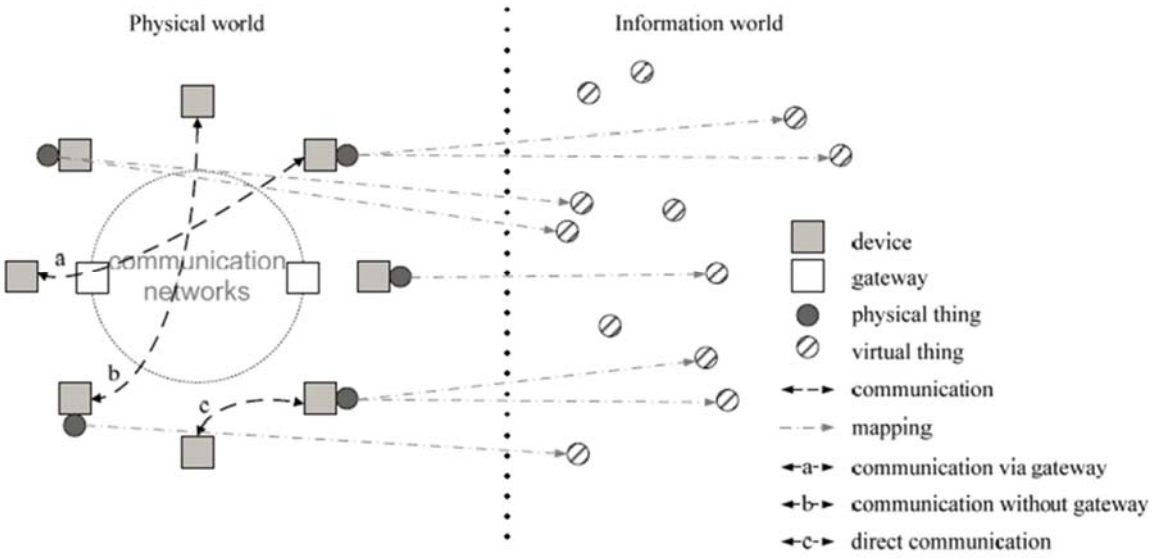


Figure 1.1 Overview of the Internet of Things [28].

A physical thing can be mapped into the information world via one or more virtual things, while virtual things do not necessarily need to be associated with any physical thing and

can exist independently of any physical existence. For example, a physical thing might execute multiple applications and thereby have multiple identities in the virtual world. Similarly a virtual thing might also have many identities in the virtual world. For example, a virtual thing could be a video (file) on a USB-drive. Such a file might have multiple file names that refer to it and it might even have multiple instances (copies), potentially these "copies" might have different encodings, resolutions, etc.

How does one differentiate an IoT device from any other device? Table (1.1) states some fundamental characteristics for IoT. These characteristics may provide a clearer picture of the actual differences between IoTs and other devices [28] .

Table 1.1 Characteristics of the Internet of Things

Characteristics	Description
Interconnectivity	Everything can be connected to the global information and communication infrastructure
Things-related services	Provides things-related services within the constraints of things, such as privacy and semantic consistency between physical and virtual thing.
Heterogeneity	Devices within IoT have different hardware and use different networks but they can still interact with other devices through different networks. (i.e., Case A in Figure (1.1). using different protocols or hardware, but still be able to communicate)
Dynamic changes	The state of a device can change dynamically, thus the number of devices can vary. (Device states: connected, disconnected, waking up, and sleeping)
Enormous scale	The number of devices operating and communicating will be larger than the number of devices in the current Internet. Most of this communication will be device to device instead of human to device.

Interconnectivity is the basic characteristic for IoT since the whole concept is built upon the idea of being able to interconnect everything (despite the traffic going through different networks). Things related services resolves around devices being constrained by its CPU performance, memory, and power which limits what a device can do, when it can do it, and how often it can do it.

To provide semantic consistency a physical thing reporting temperatures at some intervals may be mapped to a virtual thing that tries to estimate the temperature between measurements and thus may report a different temperate value than the physical value. When the next measurement arrives the virtual device may or may not update its estimate in order to maintain consistency with the physical thing.

In Table (1.1) the biggest challenge will be supporting heterogeneity because there are a lot of different protocols in use. Interacting with multiple devices through multiple networks will be challenging from both security and technical perspectives, because the protocols may differ depending upon whether the device is communicating through one interface or another (e.g., wide area cellular radio, Ethernet, or Wi-Fi). Therefore, there are some requirements relevant for IoT, such as security and privacy protection. If everything is connected, then multiple security threats will arise causing confidentiality, integrity, availability, and authenticity to become more important – especially because there will be more data and services available and because more and more activities will depend upon this information. Security also includes privacy consideration, since data collected by for instance a sensor might contain information that is sensitive personal information. Integrity has to be considered in all stages (sensing, storing, transmission, etc.) that means that the security within IoT will have to adapt to a variety of devices and networks [28].

A thing that reports a geographical location can for privacy reasons add noise to its position (i.e. degrade its accuracy) thus the physical location compared to the virtual location can differ. This prevents the device from having an exact location mapped to it thus protecting spatial privacy.

1.5 The IoT reference model

The ITU-T has defined a reference model for IoT. This model is divided into the four layers: application layer, service support and application support layer, network layer and device layer see Figure (1.2), each one of these layers also includes management and security capabilities. As shown in the Figure (1.2), these capabilities have both generic and specific capabilities that can cut across multiple layers.

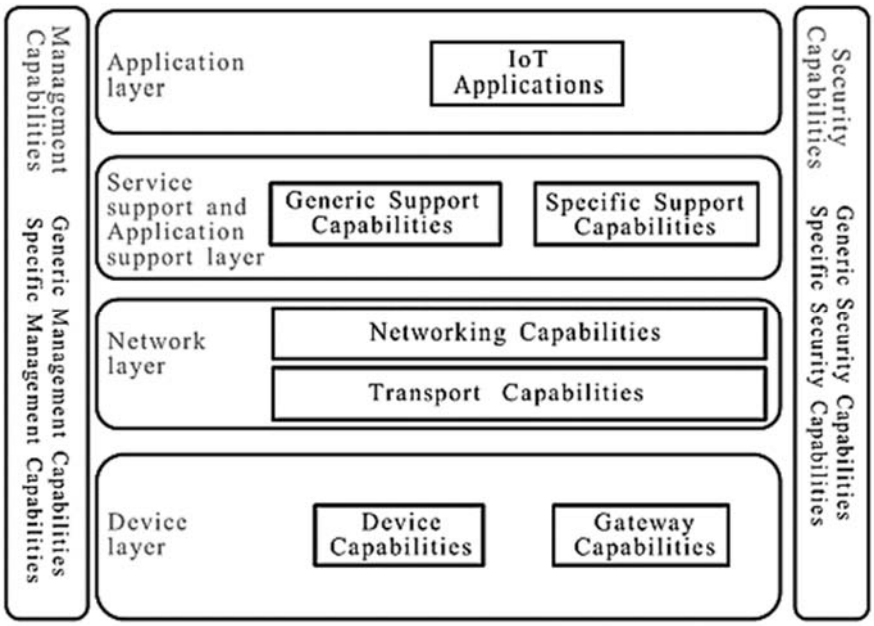


Figure 1.2 ITU-T reference model for IoT.

The application layer contains IoT applications which require certain support capabilities from the underlying layer to function. The service and application support layer consists of generic support capabilities which can be used by IoT applications, examples of such capabilities could be data processing or storage. The specific support capabilities are those other than the generic capabilities which are required to create support for diversified applications [28].

The network layer is divided into networking and transport capabilities. The networking capabilities provide relevant control functions for network connectivity, while the transport capabilities focus on the transport of IoT service and application specific data. At the bottom of the model, there is the device layer in which the device capabilities include direct and indirect interaction with the communication network. Unlike direct interaction, indirect interaction requires a gateway to be able to send and receive information via the network. Two other capabilities are ad hoc networking and sleeping and waking up which enable devices to connect in an ad hoc manner and saving energy (respectively) [28].

The device layer also includes gateway capabilities to support devices connected via different types of wired and wireless technologies by supporting multiple interfaces. In some situations, protocol conversion is needed to support communication between devices using different protocols at the device and network layer [28].

Generic management capabilities include device management (such as remote device activation, de-activation, diagnostics, and firmware or software updates) and local network topology, traffic, and congestion management [28].

1.6 The use of IoT today

Since 1999 the term IoT has been used in many places and in many ways. Multiple research papers, books, and white papers about IoT have been written in order to help both the public and companies understand what IoT is. Many definitions of IoT have been independently introduced by both individuals and companies [29].

Technical companies that are already somewhat involved in IoT and who believe that IoT has a business potential for their future mostly use the term to describe a way of improving efficiency of production and innovation. Cisco defines IoT as concept where more and more things will be connected to the Internet in order to ease people's daily life. However, as we connect more things, the need for IPv6, big data, and cloud computing will increase and the concept of IoT will transition into an Internet of Everything (IoE). Cisco views IoT as a phase where the number of connected devices increases, while this phase changes once everything connected [30].

IBM has a definition of IoT which is more about connecting systems together, rather than just connecting devices together; thus, their focus is on creating a system of systems. They describe IoT as a means to create a smarter planet. They split these means into two parts: "One is to be more efficient, be less destructive, to connect different aspects of life which do affect each other in more conscious, deliberate and intelligent ways. But the other is also to generate fundamentally new insights, new activity, new forms of social relations" [31].

Individual definitions include that given by Dr John Barrett, Head of Academic Studies for Embedded Systems Research at Cork Institute of Technology in a TEDx talk on the requirement for IoT: In the context of IoT all things will need a unique identity (IPv6), ability to communicate, in some way sense (see, smell, touch, etc.) and to be controlled. With all the collected data there is a need for a practical and efficient way to present the data that is relevant in a certain context. Deciding what is relevant becomes a core question. It is up to the things themselves to decide what is relevant and what is not. In some cases the "relevant" data may be misused in a way that negatively effects people. For example, a device monitoring your health can be used to notify the hospital if your

health is in critical condition. However, by using the same information as the hospital, your insurance company automatically increases your health insurance premium by 25% [32].

The definition by ITU-T highlighted the enabling of services and the interconnecting of things. This should be made possible by using existing and evolving communication technologies. ITU-T defines a three dimensional space in which IoT adds one of the dimensions (anything communication) to the information and communication technologies which already provide the two others: "any time" and "any place"). In other words, previously we could communicate at any time and place, but with IoT we can communicate with any “thing”[1].

Even though businesses, individuals, and papers explain IoT in slightly different ways, the similarity of their definitions centres on the interconnecting of things. The difference in their definitions is how they present the concept. Businesses mostly focus on the possibilities within IoT with regards to efficiency and innovation, but do not mention the security threats which may arise. This does not mean that these businesses are unaware of potential risks and that they do not have a suitable plan regarding IoT (although this could be true). However, business may simply choose not to publicly announce the risks they see with the concept of IoT nor how they plan to secure it. For a business it is always valuable to possess information which your opponent does not. While at the same time security via obscurity has been found time and again to not really provide security!

1.7 Fields of IoT

The term IoT is being used in different fields, such as the body, homes, cities, industry, and the global environment. The following points from [1].

- In terms of the body, IoT enables sensing and connectivity, for example tracking activity, health status, and other relevant information could improve not only the user’s daily life, but also their future health by preventing bad habits. However, this could come at the cost of a tremendous decrease in personal integrity and personal autonomy. Hence there are both individual and societal issues that have to be addressed with this sort of IoT.
- When talking about the home, IoT is often considered in terms of remote and local monitoring and management of different home electronics and lights, or simply to keep plants in the yard alive by using an automatic watering system. Today this

is becoming a very important area as more and more areas are facing shortage of water, hence traditional approaches to watering house plants and gardens are no longer feasible.

- In correlation to cities, the term IoT is used to describe systems that effectively gather and process information generated by various infrastructures, for example monitoring centres for traffic lights, street lights, camera surveillance and the power grid. These systems offer the potential to improve the flow of vehicles and people through the city centres and also greatly improving the energy efficiency of transport systems, while also improving personal and societal safety.
- Optimizations of operations, boosting productivity, saving resources, and reducing costs are typically the main goals of IoT solutions applied in industry. For example, industry might use IoT to keep track of business assets, improve environmental safety, and maintain quality and consistency in a production process. This is not only a matter of companies seeking to be "green" but also because there are very substantial economic advantages to understanding how to do better process control (in terms of maintaining quality), but also lessening the harmful effects upon the environment.
- Last, but not least important, is environmental monitoring where IoT can help us understand and better manage those resources we have. Sensors can help protect wildlife, track water usage and flows, monitor local weather, monitor use of natural resources, or give warnings before and after natural disasters to prepare people for what is to come. In fact, it appears that to achieve high environmental efficiency requires increasing use of information technology (whether this is in production, consumption, recycling, or post-recycling phases).

1.8 Summaries of IoT

IoT includes different objects with different capabilities, which have a common way of communicating (a communication chain through a communication network) for enabling transfer of information, where this information is understood by two or more objects in order to make a process more efficient; frequently by minimizing human factors and interaction.

Objects include both virtual and physical objects, but are not limited to:

- ✓ Electronic devices (e.g. computers, mobile phones, televisions, machines, and robots) and
- ✓ Sensors (connected through devices or gateways)

Communicating includes:

- ✓ Different protocols and technologies for sending digital or analogue signals through nodes (e.g. Constrained Application Protocol, File Transfer Protocol, Hypertext Transfer Protocol, etc. in Local Area Networks, Wide Area Networks, Body Area Networks, Wi-Fi, Ethernet, fibre optic links, radio etc.)

1.9 Unmanned Aerial Vehicles

Drone, as a definition is an aircraft without human on board, drones some times are called unmanned aerial vehicles or unmanned aircraft system, in another point of view it is a robot that flies.

These aircrafts may be controlled remotely or can be autonomous through flight system that use software on an embedded systems using sensors and GPS.

In the past years, the main use of drones was related military applications,

The drones were used mainly in anti-aircraft target practice and weapon platforms , now days they are used in civilian applications like search, rescue, weather monitoring , traffic monitoring, delivery services …etc.

The drone (plane shaped) called "Queen Bee" is considered as the first drone used, which was supplied with a radio signal for controlling the servos-operated parts.

The plane could be piloted from the seat in front, but it flews unmanned and it was shot by gunners in training.

In 2012, a man called Chris Anderson retired from his work to dedicate himself to a new drone company, 3D Robotics. The company started by constructing a personal drone for hobbyist. Nowadays it is marketing its solutions for film and photography companies, telecom businesses…etc.

In 2013, amazon was a leader in the use of drones in delivery activities, since then the others started to use the drones for the same purpose like State University and Virginia Polytechnic Institute in testing Project Wing, to make deliveries starting with burritos produced.

The most common use for drones in drone surveillance and drone journalism, the most important thing in drone is it can be used to access locations that is impossible for human to get and it can do things a human can not do.

The process of drone education is expanded, for example Embry-Riddle Aeronautical University now offers a Bachelor degree in unmanned systems and it is applications, also it gives a master degree for unmanned systems.

In 2016 Business Insider BI Intelligence forecasted that an increase in drone sectors in revenues and shipment by 2021, will reach 29 million shipments in worldwide.

Integration between drones technology with the IoT sensor networks can produce a great help for agricultural companies to monitor areas, lands and crops, also it can produce a great achievement to the energy companies for monitoring power lines and operational equipment. Insurance companies can take benefit from drones in properties monitoring and claims.

1.10 Thesis Outline

This thesis contains of five chapters. The first chapter about general concepts of IoT and related works with ours. The second chapter about hardware components that been used in our work. The third chapter we explained which materials used in our work and how assembled. In fourth chapter the quad copter simulation and PID parameter are showed. And the fifth chapter is the conclusion of thesis.

CHAPTER 2

HARDWARE COMPONENTS

This thesis proposes a case study for designing a system that uses an Internet of Things (IoT) to make the control for Unmanned Aerial Vehicle (drone) from an infant distance by simulating the signals coming from RF receiver and remote control, these signals will be provided from raspberry pi device to apply the signals of **Roll**, **Pitch**, **Yaw**, and **Throttle** to take control of UAV, the system uses an interaction between raspberry pi and flight controller to overcome the complexity of stability and PID control calculations. The control is designed using python 2.7 GUI (Graphical User Interface) to take control for flight, also the design is containing the capability of tracking objects based on the BGR color moment calculations.

2.1 UAVs Components

There are many different types of frames and formations that used to create UAVs.

2.1.1 UAV Frame Types

The most famous types of UAV frames are:

1. **Tricopter:** In this type, the UAV has three arms, each connected to one motor. The front of the UAV inclines to be between two of the arms (Y3). The angle between the arms can be any degree, but generally and most likely is to be 120 degrees, as shown in Fig (2.1).

   The Advantages of this type is flying more homogeneous to an airplane in the forward motion. Price is theoretically lowest since it utilizes the fewest number of a brushless motor. And the disadvantages are the design utilizes a normal servo motor to rotate the rear motor and as such, because of the copter is not symmetric.

The design is less modest than many other multi-rotors. The rear arm is more intricate because of servo motor needs to be mounted along the axis. In integration, not all flight controllers support this configuration.

Figure 2.1 Tricopter frame.

2. **Quadcopter:** In this type, a UAV has four arms, each arm connected to one motor. The front of the UAV looks to be between two arms (× configuration), but it can also be over an arm (+ configuration), as shown in Fig (2.2). Advantages of this type are simplest construction and more multifarious. In the standard configuration, the arms/motors are symmetric around two axes. In the markets, all flight controllers can work with this UAV design, the most popular UAV design. And the disadvantages is in this type there is no redundancy, so if there is a failure anywhere in the system, especially in a motor or fan, the craft is likely going to crash.

Figure 2.2 Quadcopter frame.

3. **Hexacopter:** In this type, UAV has six arms, each connected to one motor. The front of the UAV it can be between two arms, but can also be along one arm, as shown in Fig (2.3). From the advantages of this type, it increases the total thrust available, meaning the UAV can elevate more payload. Additionally, if a motor fails, there is still a chance the UAV can land rather than crash. Virtually all flight controllers support this configuration. And disadvantages is in this design extra components are utilized, so compared to a quadcopter which utilizes a minimum number of components, the equipollent hexacopter utilizing the same motors and fans would be more expensive and larger. These supplemental motors and components integrate weight to the UAV, so in order to get the same flight time as a quadcopter; the battery needs to be larger as well.

Figure 2.3 Hexacopter frame.

4. **Octocopter:** In octocopter design UAV has eight arms, each connected to one motor. The front of the UAV tends to be between any two neighbor arms, as shown in Fig (2.4). Advantages of this type is more thrust, as well as increased redundancy because of uses more motors. And disadvantages is more expensive and larger battery needed.

Figure 2.4 Octocopter frame.

2.1.2 UAV Frame Size

There are many different sizes of UAVs frames, from “Nano” which is smaller than your hand, to “Mega”, which can only be conveyed by the truck. For most users who are getting commenced in the field, a good size range which offers the most flexibility its size is between 350mm to 700mm. This measurement represents the diameter of the largest circle which intersects all of the motors [33].

In this thesis we used quadcopter frame with size 450mm, because of simplest construction, symmetric around two axis, uses a minimum number of parts, cheaper from other frames, low in weight and easy to find it in local markets.

2.1.3 Motors

A large effect on the payload (or maximum load) are in fact from the motors used which UAV can support, as well as the flight time. We vigorously suggest utilizing the same (propulsion) motor everywhere. Note that even if a pair of motors are the same brand and model, and from the same production run, their speeds may differ partially, which is the flight controller will take care of. Figure (2.4) show some types of motors.

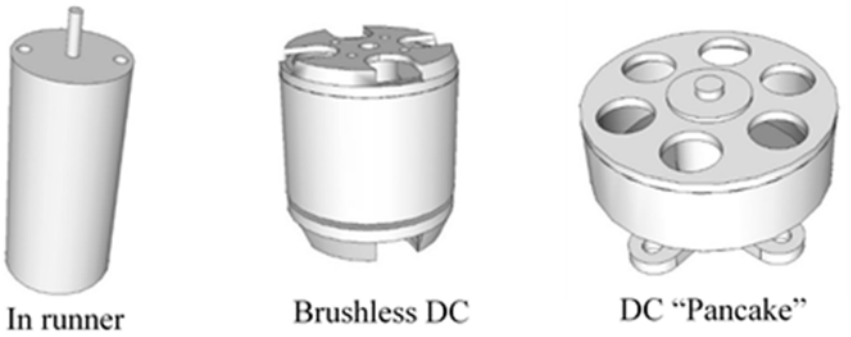


Figure 2.5 Some types of motors.

2.1.3.1 Brushed vs Brushless

Brushed motors spin the coil inside a case with fine-tuned magnets mounted around the outside of the casing. Brushless motors do the antithesis; the coils are fine - tuned either to the outer casing or inside the casing while the magnets are spun. In most statuses, you will be considering only brushless DC motors. “Pancake” brushless motors have a more sizably voluminous diameter and are essentially flatter and often sanction for higher torque and lower KV (details in 2.1.3.3). More minute UAVs (conventionally the size of the palm of your hand) incline to utilize minuscule-brushed motors because of the lower

price and simpler two-wire controller. Albeit brushless motors come in a variety of different sizes and specs, culling a more minute brushless motor infrequently designates it will be less extravagant. Brushed DC motors have two connectors: one for positive, the other for negative. Inverting the wires reverses the rotation of the motor. Brushless DC motors have three connectors. Refer to the ESC section (2.1.5) to know how to wire them and invert the direction of rotation.

2.1.3.2 In runner vs Out runner

There are a slight types of brushless DC motors:

- **Inrunner:** These have the fine-tuned coils mounted to the outer casing and the magnets are mounted to the armature shaft which spins inside the casing (incline to be utilized on RC cars because of the high KV.
- **Outrunner:** These have the magnets mounted on the outer casing, which is spun around the fine-tuned coils in the center of the motor casing (the bottom mounting of the motor is fine-tuned).
- **Hybrid outrunner:** Technically outrunners but have a static outer shell around them to make them look homogeneous to their inrunners.

Inrunner brushless DC motors incline to be utilized in airplanes and helicopters because of their high KV. They may additionally be geared down to increment the torque. Outrunners incline to have more torque.

2.1.3.3 KV

KV it refers to the rpm constant of a motor, it is the number of rotation per minute that the motor will turn when 1V (one Volt) is applied with no load annexed to the motor. In summary, we call it revs per volt but do not cerebrate you will obtain those revs when you affix a fan; distinctly the revs will be reduced because of the load. KV is cognate to the potency out from a motor, or more usefully the torque level of a motor. It is tenacious by the number of winds on the armature (or turns as we sometimes call it) and the vigor of the magnets, there are so many variables with electric motors. So KV sanctions us to get a handle on the torque we can expect from a particular motor. The KV is rating/value of a motor relates to how expeditious it will rotate for a given voltage. For most multirotor aircraft, a low KV is required (between 500 to 1000 for example) since this avails with stability. For acrobatic flight however, you might consider a KV between 1000 and 1500

and withal consider utilizing more minute diameter fans. If the KV rating for a particular motor is 650rpm/V, then at 11.1V, the motor will be rotating at 11.1V x 650 = 7215rpm. If you operate the motor at a lower voltage (verbalize, 7.4V), the rpm will be 7.4V x 650rpm/V = 4810rpm. It is consequential to note that utilizing a lower voltage inclines to designate that the current draw will be higher (power = current x voltage).

2.1.3.4 Thrust

Some brushless motor manufacturers give a designation of a motor's thrust corresponding to several propeller options (often presented in a table). The unit of thrust is often Kg, Lbs or N. For example, if you are building a quadcopter and find that a categorical motor can provide up to 0.5Kg of thrust with an 11 inch propeller, that signifies that four of these motors (with that given prop) can hoist 0.5Kg*4=2Kg at maximum thrust. Ergo if your quadcopter weighs just less than 2Kg, it will only take off at maximum thrust. You require to either cull a motor + propeller amalgamation which can provide more thrust, or reduce the weight of the aircraft. If the propulsion system (all motors and props) can provide 2Kg of thrust (max) then your entire copter should be at most about half this weight (1Kg, including the weight of the motors themselves). The same calculation can be done for any given configuration.

2.1.4 Propeller (Fans)

Propellers for multi-rotor aircraft are adapted from propellers used in RC airplanes. The material(s) used to make the propellers can have a moderate impact on the flight characteristics, but safety should be the primary consideration as shown in Fig (2.5). There are three types of propellers (plastic, Fiber-Reinforced Polymer, Natural such as wood).

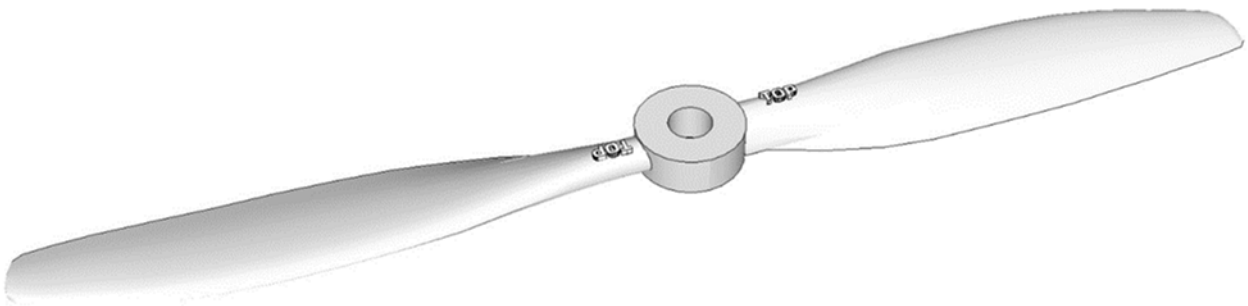


Figure 2.6 UAV Fan.

2.1.4.1 Blades & Diameter

Most multi-rotor aircraft have either two or three rotor blades, with the most mundane being two. Do not surmise that integrating more blades will automatically mean more thrust; each blade must peregrinate through the wake of the one, which precedes it, so the more blades, the more prevalent the wake will be. A more diminutive diameter propeller has less inertia and is ergo more facile to expedite and decelerate, which avails in acrobatic flight.

2.1.4.2 Pitch / Angle of Attack / Efficiency / Thrust

The thrust engendered by a propeller depends on the density of the air, on the propeller's RPM, on its diameter, on the shape and area of the blades and on its pitch. A propeller's efficiency relates to the angle of assailant which is defined as the blade pitch minus the helix angle (the angle between the resultant relative velocity and the blade rotation direction). The efficiency itself is a ratio of the output power to the input puissance. Most well-designed propellers have an efficiency of 80%+. The relative velocity affects the angle of an assailant, so a propeller will have different efficiency at different motor speeds. The efficiency is additionally greatly affected by the leading edge of the propeller blade, and it is very consequential that it be as smooth as possible. Albeit a variable pitch design would be best, the integrated involution required as compared to a multirotor's innate simplicity betokens a variable pitch propeller is virtually never utilized.

2.1.4.3 Rotation

Propellers are either designed to rotate clockwise (CW) or counter-clockwise (CCW). It is consequential to know which component of the propeller is intended to face upwards (the top surface is curved outward). The top of the propeller should always face the sky.

2.1.5 ESC (Electronic Speed Controller)

An ESC (Electronic Speed Controller) is what sanctions the flight controller to control the haste and direction of a motor. The ESC must be able to handle the maximum current, which the motor might consume, and be able to provide it at the right voltage, Fig (2.6) shows an ESC.

Figure 2.7 an ESC.

ESC Connectors:

An ESC might initially be embarrassing because it has several wires exiting on two sides.

- **Power input:** The two thick wires (normally black and red) are to obtain power from the power distribution board which itself receives power directly from the main battery.
- **3 black wires connectors:** These pins are what connects to the three pins on the brushless motor.
- **3-pin R/C servo connector:** This connector accepts RC signals, but rather than requiring 5V on the red and black pins, most of the time an internal BEC provides 5V to power the electronics.

BEC (Battery Elimination Circuit):

Most ESCs include what is called a "Battery Elimination Circuit" or BEC. This emanates from the fact that historically, only one brushless motor was needed in a given RC vehicle, and rather than splitting the battery, it would just need to be connected to the ESC, and the ESC would have an onboard voltage regulator to power the electronics. It is consequential to know the current which an ESC's BEC can provide, though it is normally in the range of 1A or above and is virtually always 5V.In a multi-rotor, you require to connect all ESCs to the flight controller, but only one BEC is needed, and having power emanating from multiple sources all being well fed to the same lines can potentially cause issues. Since there is normally no way to deactivate a BEC on an ESC, it is best to take off the red wire and wrap it with electrical tape for all but one ESC. It is still consequential to leave the black (ground) wire in place for "common ground".

2.1.6 Battery

Chemistry: Batteries utilized in UAVs are now virtually exclusively Lithium polymer (LiPo), with some more exotic ones being Lithium-Manganese or other Lithium variations. Lead acid is simply not an option and NiMh / NiCd are still too cumbersome for their capacity and often cannot provide the high discharge rates needed. LiPo offer high capacity with low weight, and high discharge rates. The downsides are their comparatively higher cost and persistent safety issues.

Voltage: Actually, you should only need to consider one battery pack for your UAV. This battery's voltage should correspond with the motors you opted for. Virtually all batteries utilized these days are lithium-based and incorporate a number of 3.7V cells, where 3.7V = 1S. Consequently, a battery which is marked as 4S is likely 4 x 3.7V = 14.8V nominal. Providing the number of cells, however, will help you to determine which charger to utilize. A single cell high capacity battery may physically look profoundly akin to a low capacity multi-cell battery.

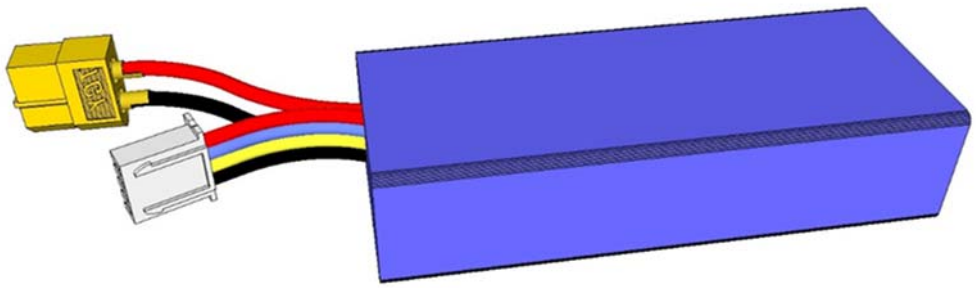


Figure 2.8 UAV Battery.

2.1.7 Flight Controller

A flight controller for a multi-rotor UAV is an integrated circuit customarily composed of a microprocessor, sensors and input / output pins. Out of the box, as shown in Fig (2.8), a flight controller does not magically ken your categorical UAV type or configuration, so set certain parameters in a software program are needed, and once consummate, that configuration is then uploaded to board. Rather than simply comparing flight controllers which are currently available.

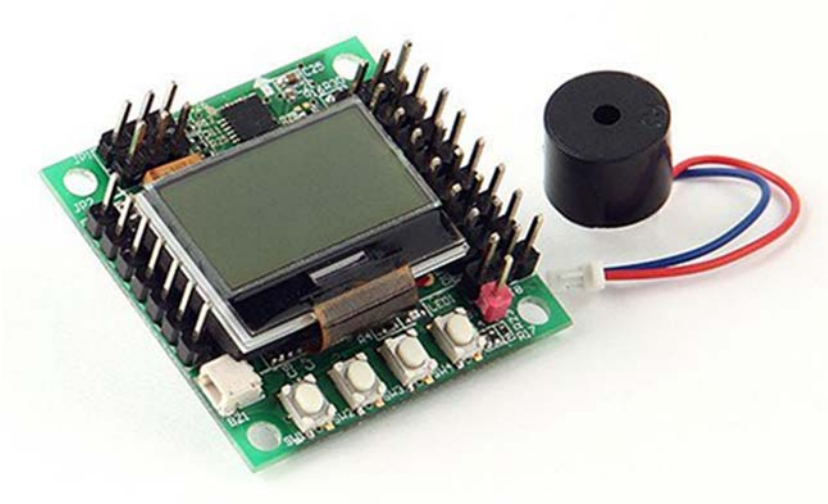


Figure 2.9 Flight Controller.

2.1.7.1 Main Processor

8051 vs AVR vs PIC vs ARM: These microcontroller families form the basis of most current flight controllers. Arduino is AVR based (ATmel) and the community seems to focus on MultiWii as being the preferred code. Microchip is the primary manufacturer of PIC chips. It is difficult to argue that one is better than the other, and it really comes down to what the software can do. ARM (STM32 for example) uses 16/32-bit architecture, whereas AVR and PIC tens to use 8 / 16-bit. As single board computers become less and less expensive, expect to see a new generation of flight controllers which can run full operating systems such as Linux or Android.

CPU: Normally these are in multiples of 8 (8-bit, 16-bit, 32-bit, 64-bit) and is a reference to the size of the primary registers in a CPU. Microprocessors can only process a set (maximum) number of bits in memory at a time. The more bits a microcontroller can handle, the more accurate (and faster) the processing will be. For example processing a 16-bit variable on an 8-bit processor is a bit of a chose, whereas on a 32-bit processor it is very fast. Note that the code also needs to work with the right number of bits.

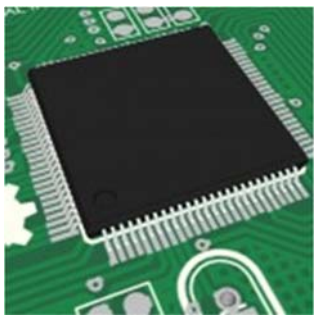


Figure 2.10 Main Processor in flight controller.

Operating frequency: The frequency at which the main processor operates. Frequency is measured in "Hertz" (cycles per second). This is also commonly referred to as the "clock rate". The higher the operating frequency, the faster it can process data.

Program Memory / Flash: The flash memory is essentially where the main code is stored. If the program is complex it may take up quite a bit of space. Obviously the greater the memory, the more information it can store. Memory is also useful when storing in-flight data such as GPS coordinates, flight plans, automated camera movement etc. The code loaded to the flash memory remains on the chip even if it power is cut.

SRAM: SRAM stands for "Static Random-Access Memory", and is the space on the chip which is used when making calculations. The data stored in RAM is lost when power is cut. The higher the RAM, the more information will be "readily available" for calculations at any given time.

EEPROM: Electrically Erasable Programmable Read-Only Memory (EEPROM) is normally used to store information which does not change in flight, such as settings, unlike data stored in SRAM which can relate to sensor data etc.

Additional I/O Pins: Most microcontrollers have a lot of digital and analog input and output pins, and on a flight controller, some are used by the sensors, others for communication and some may remain for general input and output. These additional pins can be connected to RC servos, gimbal systems, buzzers and more.

A/D converter: Should the sensors used onboard output analog voltage (normally 0-3.3V or 0-5V), the analog to digital converter needs to translate these readings into digital data. Just like the CPU, the number of bits which can be processed by the A/D determines the maximum accuracy. Related to this is the frequency at which the microprocessor can read the data (number of times per second) to try to ensure no information is lost. It is nevertheless hard not to lose some data during this conversion, so the higher the A/D conversion, the more accurate the readings will be, but it is important that the processor can handle the rate at which the information is being sent.

2.1.7.2 Power

Generally, there are two voltage ranges qualified in the spec sheet of a flight controller, the first being the voltage input range of the flight controller itself (most operate at 5V), and the second being the voltage input range of the main microprocessor's logic (3.3V or

5V). Since the flight controller is an obviously integrated unit, and attention must be taken to the input range for the flight controller itself. Most UAV flight controllers operate at 5V since that is the voltage provided by a BEC.

2.1.7.3 Sensors

In terms of hardware, a flight controller is basically a normally programmable microcontroller but has specific sensors onboard. At a minimum, a flight controller will have a three-axis gyroscope, but as such will not be able to auto-level. Not all flight controllers will have all of the sensors below and maybe include a combination of there.

The sensors:

1. **Accelerometer:** Is measure linear acceleration in up to three axes (let's call them X, Y and Z), Fig (2.10) shows the accelerometer axis's. The units are in "gravity" (g) which is 9.81 meters per second per second, or 32 feet per second per second. The output of an accelerometer can be integrated twice to give a position, though because of losses in the output, it is subject to "drift". A very important characteristic of three axis accelerometers is that they detect gravity, and as such, can know which direction is "down". This plays a major role in allowing multirotor aircraft to stay stable. The accelerometer should be mounted to the flight controller so that the linear axes line up with the main axes of the UAV.

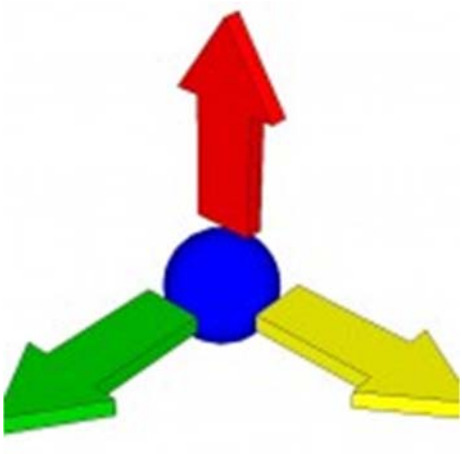


Figure 2.11 Accelerometer.

2. **Gyroscope:** A gyroscope measures the rate of angular change in up to three angular axes (let's call them alpha, beta and gamma), Fig (2.11) shows the gyroscope angular axis's. The units are often degrees per second. Note that a gyroscope does not measure absolute angles directly, but we can iterate to get the angle which, just like an accelerometer, is subject to drift. The output of the actual gyroscope tends to be analog or I2C, but in most cases you do not need to worry about it since this is handled by the flight controller's code. The gyroscope should be mounted so that its rotational axes line up with the axes of the UAV.

Figure 2.12 The Gyroscope Angular Axis's

3. **Inertia Measurement Unit (IMU):** An IMU is essentially a small board which contains both an accelerometer and gyroscope (normally these are multi-axis), as shown in Fig (2.12). Most contain a three axis accelerometer and a three-axis gyroscope, and others may contain additional sensors such as a three axis magnetometer, providing a total of 9 axes of measurement.

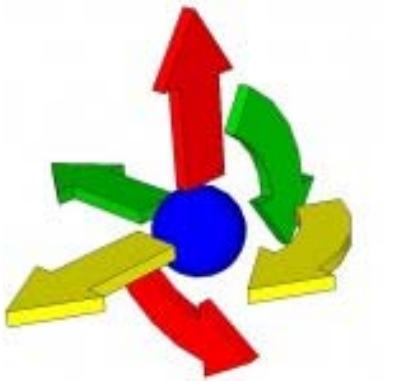


Figure 2.13 Inertia Measurement Unit (IMU).

4. **Compass / Magnetometer:** An electronic magnetic compass is able to measure the earth's magnetic field and used it to determine the UAV's compass direction (with respect to magnetic north), Fig (2.13) shows Magnetometer. This sensor is almost always present if the system has GPS input and is available in one to three axes.

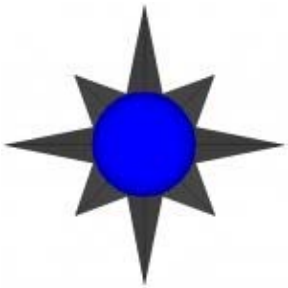


Figure 2.14 Magnetometer.

5. **Pressure / Barometer:** Since atmospheric pressure changes according to the altitude from sea level, a pressure sensor can be used to give you an accurate reading of the UAV's height. Most flight controllers take input from both the pressure sensor and GPS altitude to calculate a more accurate height above sea level. Note that it is better to have the barometer covered with a piece of foam to minimize the effects of wind over the chip.
6. **GPS:** Global Positioning Systems (GPS) use the signals sent by a number of satellites in orbit around the earth in order to determine their specific geographic location. A flight controller can either have onboard GPS or one which is connected to it via a cable. The GPS antenna should not be confused with the GPS chip itself, and can look like a small black box or a normal "duck" antenna. In order to get an accurate GPS lock, the GPS chip should receive data from multiple satellites, and the more the better.
7. **Distance:** Distance sensors are being used more and more on drones since GPS coordinates and pressure sensors alone cannot tell you how far away from the ground you are (think hill, mountain or building) or if you will hit an object. The distance of downward-facing sensor is based on ultrasonic, laser or lidar technology (infrared has issues in sunlight). Slight flight controllers include distance sensors as part of the standard package.

2.2 Raspberry Pi

The first model (Raspberry Pi 1 Model B) was produced in 2012. Then it was followed by another model, it was simple and cheap in comparing to the previous Model A. In early 2014, the Foundation produced another board with more improved design, which is Model B+. the main advantage of these boards is that they approximately credit-card sized , another generation is A+ and B+ models were produced in one year later. After that, another Module was produced in early 2014 and it is used for embedded applications. A new generation Raspberry Pi 2 produced in early 2015 added more RAM for it.

In early 2017, Raspberry Pi 3 Model B was introduced as the newest version of Raspberry Pi.

All models includes an on-chip graphics processing unit (GPU, a Video Core IV), ARM compatible central processing unit (CPU) and CPU speed starts from 700 MHz and ends at 1.2 GHz for the Pi 3 and on board memory starts from 256 MB and ends at 1 GB RAM. A Digital (SD) cards are used to store the program memory and operating system in Micro SDHC sizes. Most boards have one or more (four) USB slots, composite video output, HDMI, and a 3.5 mm jack for audio output or input. A number of GPIO pins provide Lower level output, which supports common used protocols like I²C. The B-models have an Ethernet port and the Pi 3 has an on board Wi-Fi 802.11n and Bluetooth.

The Raspberry Pi hardware can be summarized in the following diagram:

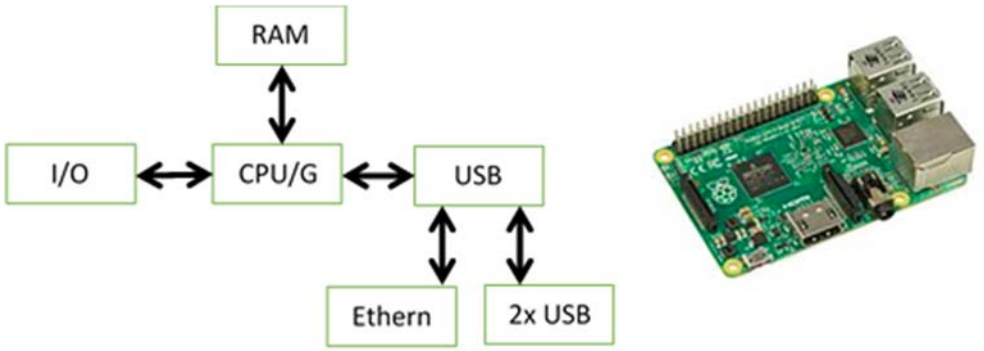


Figure 2.15 Raspberry Pi Hardware Components and B+ Model.

2.2.1 Processor

The main processor of raspberry pi 3 is a 64-bit quad-core ARM Cortex-A53 with 1.2 GHz a Broadcom BCM2837 SoC.

2.2.2 Performance

The Raspberry Pi 3, using a quad-core Cortex-A53 processor performance is 10 times the performance of a Raspberry Pi 1.[33] Benchmarks showed that Raspberry Pi 3 is approximately 80% faster than the Raspberry Pi 2 in parallelized applications.[34]

Raspberry Pi 2 includes a 1 GB RAM quad-core Cortex-A7 CPU,900 MHz. It the Benchmarks showed that it is 4–6 times more efficient than its previous processor. its GPU is the same as the original.[35] In parallelized benchmarks, the Raspberry Pi 2 is said to be up to 14 times faster than its previous model.[36]

2.2.3 RAM

On the older beta Model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. [37] On the first 256 MB release Model B (and Model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be enough for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with only a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D, possibly also with video decoding (e.g. XBMC). [38] Comparatively the Nokia 701 uses 128 MB for the Broadcom VideoCore IV. [39]

For the later Model B with 512 MB RAM initially there were new standard memory split files released(arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). But a week or so later the RPF released a new version of start.elf that could read a new entry in config.txt (gpu_mem=xx) and could dynamically assign an amount of RAM (from 16 to 256 MB in 8 MB steps) to the GPU, so the older method of memory splits became obsolete, and a single start.elf worked the same for 256 and 512 MB Raspberry Pis.[40]

The Raspberry Pi 2 and the Raspberry Pi 3 have 1 GB of RAM.[41][42] The Raspberry Pi Zero and Zero W have 512 MB of RAM.

2.2.4 Peripherals

Figure 2.16 Raspberry Pi 3.

For connecting peripherals the current Model B boards uses four USB ports. Raspberry Pi can be operated with generic USB computer keyboard or mouse. [43] It can also be used with USB to MIDI converters, USB storage, and any other devices with USB capabilities. Other devices can be connected through the various pins and connectors on the Raspberry Pi.[44]

2.2.5 Real-time clock

The available Raspberry Pi models do not have a real-time clock, so they are cannot keep tracking the time of the day independently. Some methods can be used to overcome this problem such as, some programs running on the Pi may have the ability to retrieve the time from a network time server or from the user at boot time, thus knowing the time while powered on.

To insure correct real-time tracking A real-time hardware clock with battery backup, such as the DS1307, which is binary coded, could be added to the system. Raspberry Pi 1 Models A+ and B+, Pi 2 Model B, Pi 3 Model B and Pi Zero (and Zero W) GPIO J8 have a 40-pin pinout.[45][46]

Table 2.1 Raspberry Pi pinout.

GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I²C)	3	4	+5 V	
3	SCL1 (I²C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7
(Pi 1 Models A and B stop here)					
EEPROM	ID_SD	27	28	ID_SC	EEPROM
5	N/A	29	30	GND	
6	N/A	31	32		12
13	N/A	33	34	GND	
19	N/A	35	36	N/A	16
26	N/A	37	38	Digital IN	20
	GND	39	40	Digital OUT	21

Model B rev. 2 also has a pad (called P5 on the board and P6 on the schematics) of 8 pins offering access to an additional 4 GPIO connections. [47]

Table 2.2 Model B additional 4 GPIO connections.

Function	2nd func.	Pin#		Pin#	2nd func.	Function
N/A	+5 V	1		2	+3.3 V	N/A
GPIO28	GPIO_GEN7	3		4	GPIO_GEN8	GPIO29
GPIO30	GPIO_GEN9	5		6	GPIO_GEN10	GPIO31
N/A	GND	7		8	GND	N/A

Models A and B provide GPIO access to the ACT status LED using GPIO 16. Models A+ and B+ provide GPIO access to the ACT status LED using GPIO 47, and the power status LED using GPIO 35.

2.2.6 Operating systems

Figure 2.17 Back view of Raspberry Pi 3 shows MicroSD.

The operating system on raspberry pi can be installed on MicroSD card, we can see from figure (2.16) above the MicroSD slot on the bottom of raspberry pi 2 or 3 board.

The most recommended operating system for raspberry pi by its foundation are Raspbian, a Debian-based Linux operating system. Also a third party operating systems such as include Ubuntu MATE, Snappy Ubuntu Core, Windows 10 IoT Core, RISC OS are available for raspberry pi2 and 3. Many other operating systems can also run on the Raspberry Pi.

2.3 Arduino

Arduino is an open-source hardware. Layout and production files for all versions of the hardware are available in company website. The source code for the IDE is released.

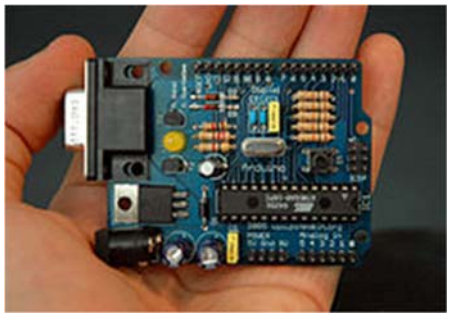


Figure 2.18 RS-232 serial interface.

An early Arduino board[18] started with an RS-232 serial interface as shown in the figure (2.17) above. Using an Atmel ATmega8 microcontroller chip (the black chip in the lower left side in figure above); at the top the 14 digital I/O pins are located, at the lower right the 6 analog input pins are located, and the power connector at the lower left in the figure above.

Arduino microcontrollers can be programmed with a boot loader that simplify the process of uploading programs to the on-chip flash memory. Boards are programed with the code via a serial connection to a computer. In Some Arduino boards there are a level shifter circuit to convert between RS-232 logic levels and (TTL) level signals. The Current Arduino boards now days are programmed using (USB), the connection between arduino and computer is made using cable, in some cases standard AVR in-system programming (ISP) programming is used.

Figure 2.19 Arduino Uno.

An official Arduino Uno R2 with descriptions of the I/O locations

The Arduino Uno[c] board provides 14 digital I/O pins, six of them can produce pulse-width modulated signals, and it has six analog inputs, also it can be used as six digital I/O pins. These pins are located on the top of the board, via female 0.1-inch (2.54 mm) headers.

A program for Arduino is written in any programming language for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio. [48][49][50]

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

A program written with the IDE for Arduino is called a sketch. [51] Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.

Using specific rules of code structuring Arduino IDE supports the languages C and C++. The Arduino IDE supplying a software library from the Wiring project, which provides a lot of input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in the hexadecimal encoding that is uploaded into the Arduino board by an uploader program in the board's firmware. There are many free public libraries for developers to use to boost their projects.

CHAPTER 3

ASSEMBLY AND PROCEDURE

3.1 UAV chosen components

According to the information given in the previous chapter (2), in this thesis we have chosen F450 Quadcopter frame with 1400KV brushless motors, EMAX ESC 30A, KK2 Flight controller, RC remote control 6 channels and polycarbonate propellers.

3.2 UAV Assembly

In F450 frame we get 4 arms and two boards as shown in Fig (3.1).

Figure 3.1 F450 UAV Frame Used.

Lower board is printed board (Power Distributed Board (PDB)) and we connect the main input power from battery to the PDB and connect all ESC's to the PDB too. The PDB have 4 connectors as an output and we connect it to the 4 ESC's and we connected the input to the battery, the PDB used mostly to split the main battery to each of the ESCs. The voltage is supplied to the ESCs at the same level, so there is no need to increase it (step up) or decrease it (step down) as shown in Fig (3.2) and Fig (3.3).

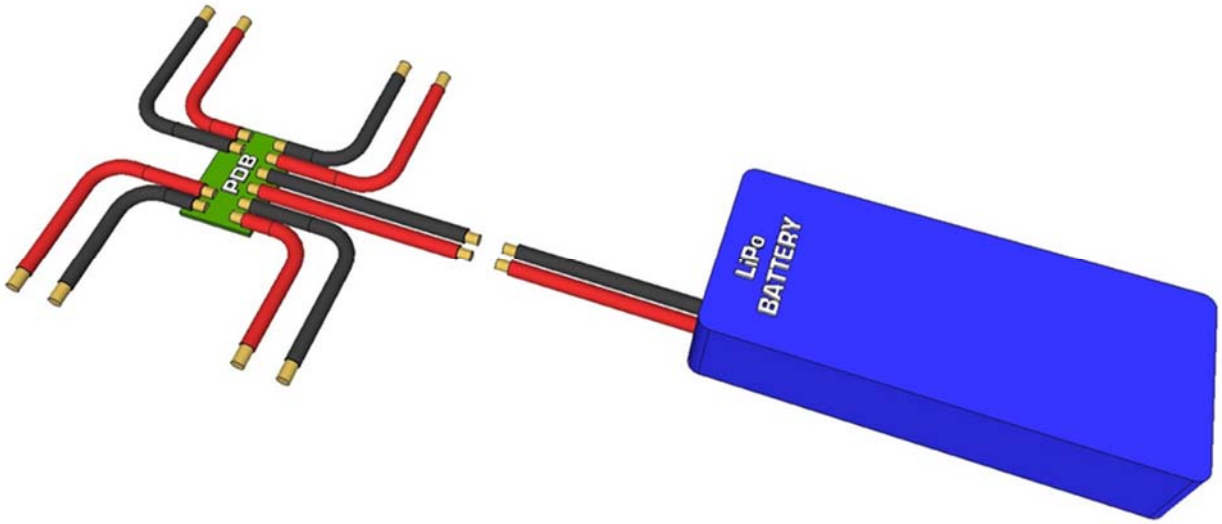


Figure 3.2 Power Distribution Board to Battery.

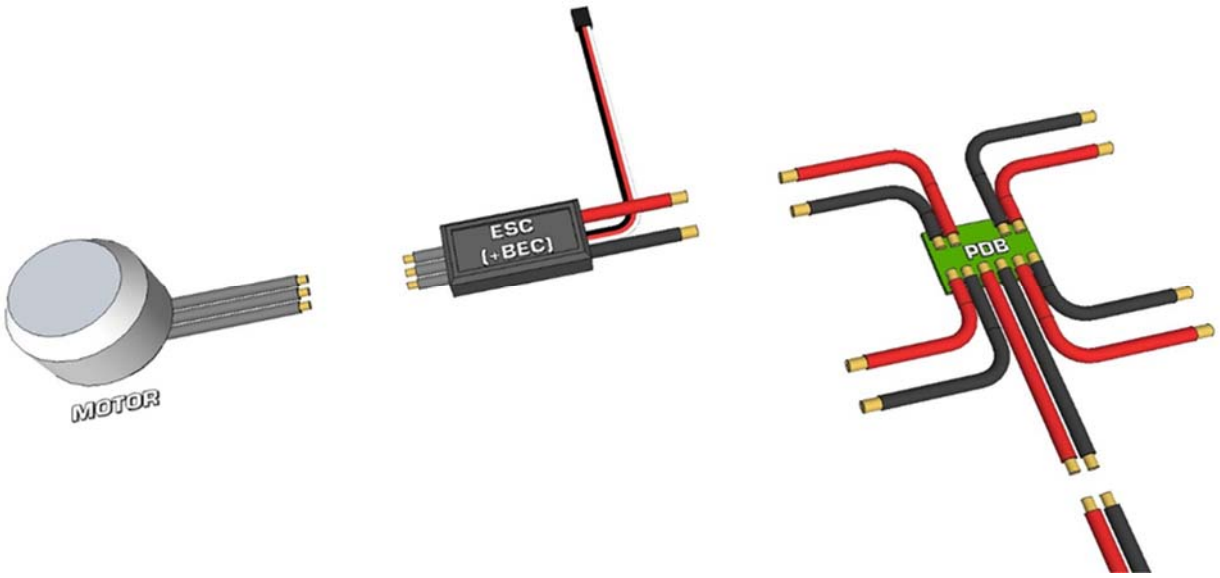


Figure 3.3 Motor to ESC (with BEC) to Power Distributed Board.

After attaching the motors on the frame body, we connect it to the ESCs, each motor to its related ESC. The direction of motor is important for that reason every two opposite motors are in the same direction and the other two in the reverse direction [52], we have

two direction clockwise (CW) and counter-clockwise (CCW). For the CW direction, the connection between motor and ESC is direct connect that mean left side from motor goes to left side from the ESC, the middle from the motor goes to the middle from the ESC and right side from the motor goes to the right side from the ESC. For the CCW direction, the connection between motor and ESC is in revers attach that mean left side from motor side goes to right side from the ESC, middle side from the motor goes to the middle side from the ESC, and right side from the motor goes to left side from the ESC as shown in fig (3.4).

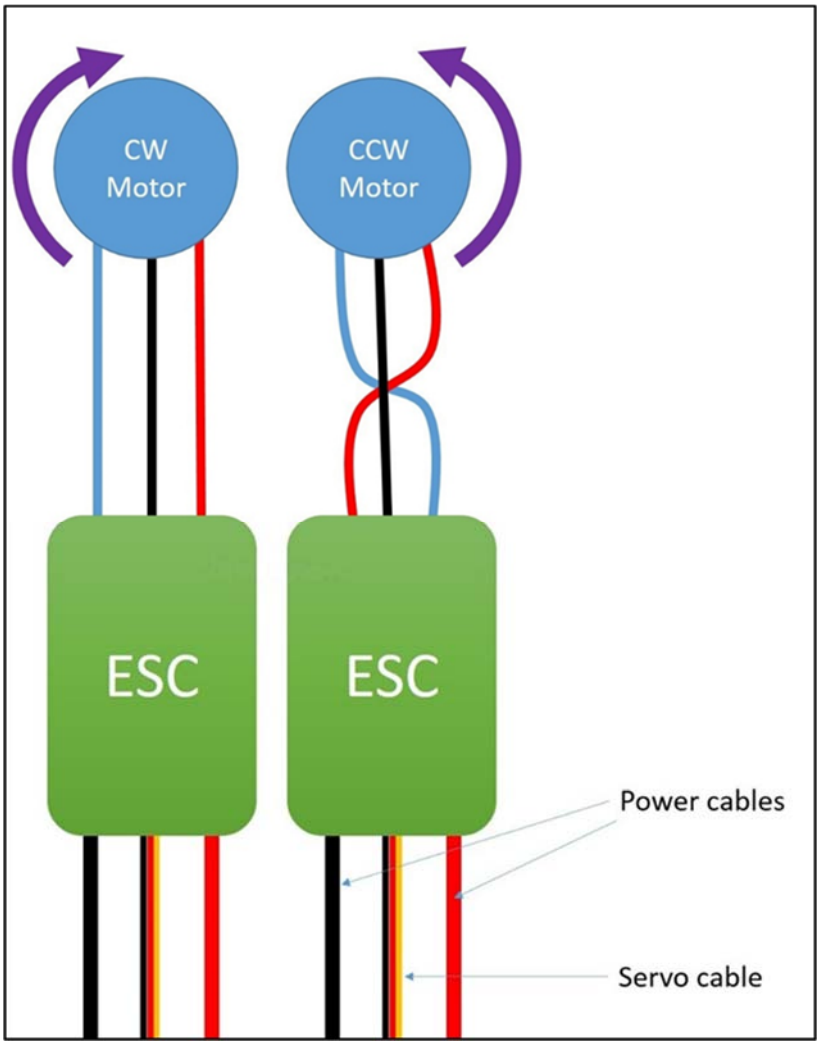


Figure 3.4 Connection types between motor and ESC.

If the motors are all in the same direction and due to Newton's third low of motion "**for every action, there is an equal and opposite reaction.**" As of it, the body of the quadcopter will tend to spin in the opposite directional to the motors. Therefore, the direction of all motors must be as in Fig (3.5).

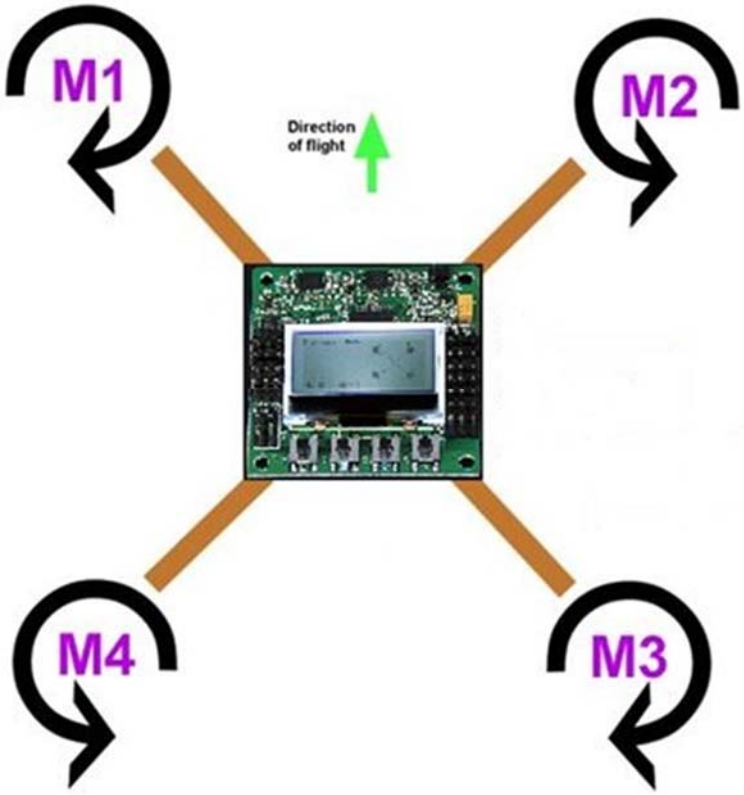


Figure 3.5 direction of UAV motors.

In our UAV we used An EMAX ESC (Electronic Speed Controller) as shown in Fig (3.6) below.

Figure 3.6 EMAX ESC 30 A.

The 3 pins servo connectors goes to flight controller, these wires goes to flight controller and receives the PWM (Pulse Width Modulation) from flight controller to control the rotational speed of the motor.

3.3 Pulse Width Modulation (PWM)

PWM is a technique uses digital circuits to control analog circuits, by generating analog signals from digital devices. PWM used in a wide variety of applications, measurements, communications, conversion and power controls.

Analog systems used to generate a lot of heat because they are basically variable resistors carrying a lot of currents. While digital systems do not generate much of heat, approximately all the generated heat by a switching device is during the transition, while the device is neither on nor off, but in between. This is because power follow the following formula:

$$\boldsymbol{P = E \times I} \quad \text{Or} \quad \boldsymbol{Watts = Voltage \times Current} \tag{3.1}$$

If any of voltage or current is near zero then the power will be near zero too, PWM takes advantage of this fact. A PWM signal can defined its behavior by two main components duty cycle, and frequency. The PWM waveform are shown in Fig (3.7) below.

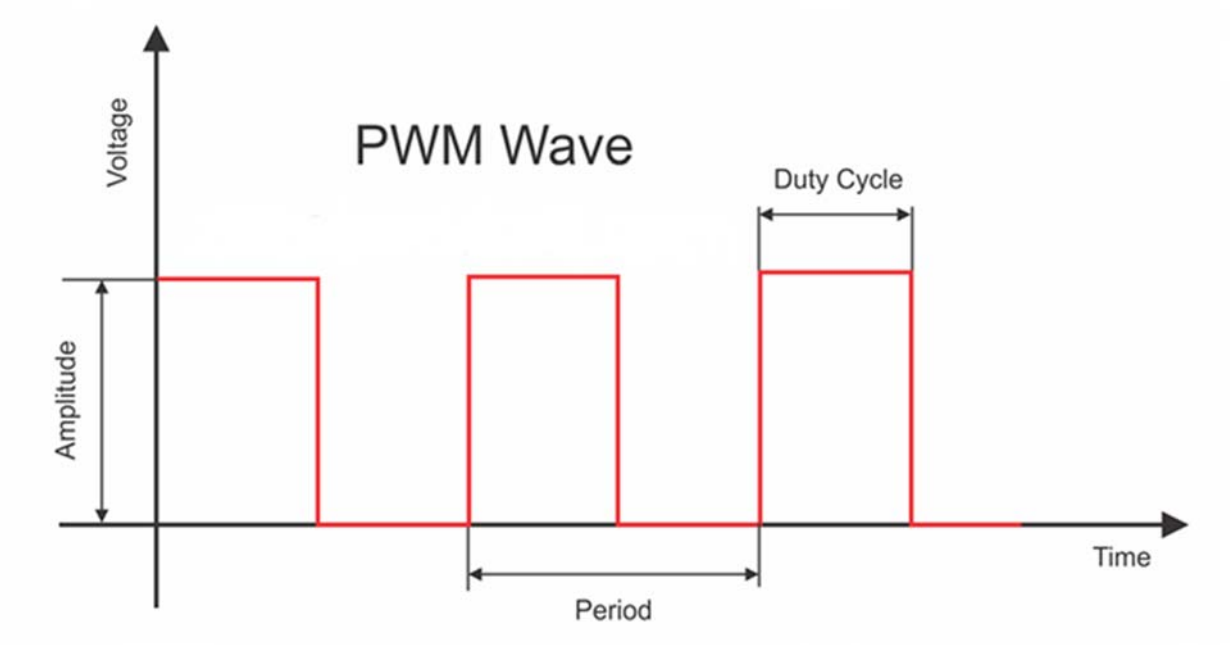


Figure 3.7 A PWM Waveform.

The duty cycle describes the time of signal in high (ON) status as a percentage of a total time of it takes to complete one cycle. One period pulse consist of time ON and time OFF, Fig (3.8) illustrate pulse timing. The frequency describes how fast the PWM complete a cycle (i.e. 500 Hz would be 500 cycles per second).

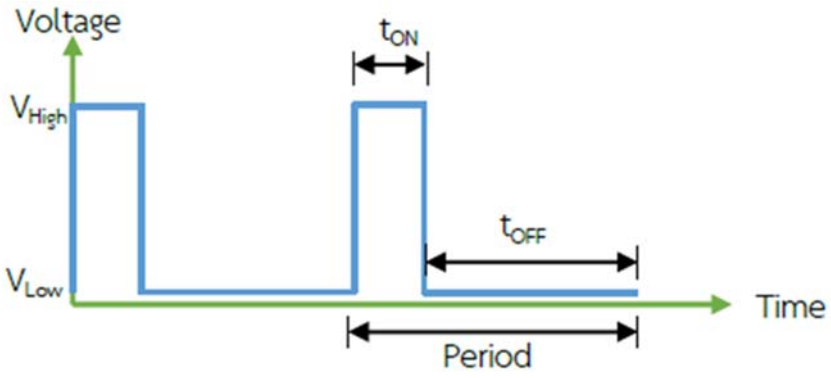


Figure 3.8 Period pulse timing.

The duty cycle can be calculated from the following formulas:

$$\boldsymbol{Duty\ Cycle} = \frac{\boldsymbol{t_{On}}}{\boldsymbol{T}} \tag{3.2}$$

Where T: is the total time ($t_{on} + t_{off}$).

Or

$$\boldsymbol{Duty\ Cycle = PWM\ (in\ Sec) \times Frequency\ (in\ Hz) \times 100} \tag{3.3}$$

Fig (3.9) shown below are examples of a 0%, 25%, 50%, 75% and 100% duty cycle. While the frequency is the same for each.

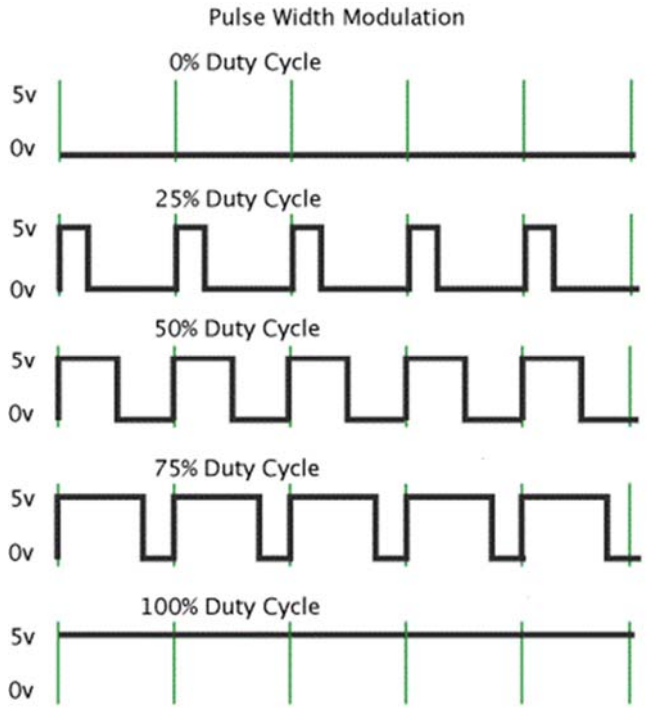


Figure 3.9 Examples of duty cycle.

3.4 Installing and connecting Flight Controller

Flight controller is the most important component in the UAV, it must be in the center of the frame (center of gravity) and with the same level of motors, in addition to its direction. Stability and controlling UAV is from flight controller responsibilities.

Controlling the motion of the Quadcopter (UAV) is by three main things, the Yaw (Rudder), Pitch (Elevator) and Roll (Ailerons), in addition to the Throttle, which is the distance from ground level.

3.4.1 Yaw (Rudder)

It is the rotating / deviation of the quadcopter (UAV) to right or to left, by rotating around the virtual axis Z. As shown in Fig (3.10) below.

Figure 3.10 Yaw axis.

3.4.2 Pitch (Elevator)

It is moved the UAV to the front or to the back, by rotating around the virtual axis Y. as shown in Fig (3.11).

Figure 3.11 Pitch axis.

3.4.3 Roll (Ailerons)

It moves the UAV to the sideward either to right or to left, by rotating around the virtual axis X. As shown in Fig (3.12) below. Many people are confusing between Yaw and Roll, Yaw is change the direction of the UAV fly but Roll is move the UAV to right or left.

Figure 3.12 Roll axis.

3.4.4 Center of Gravity (CG)

Is the effective point where all axes of flight (Roll, Pitch and Yaw) meet on it, also all weight is considered to be, Fig (3.13) shows the CG of UAV. CG point does not change in any aircraft but some times moves forward or backward along the longitudinal axis, depending on how the aircraft is loaded.

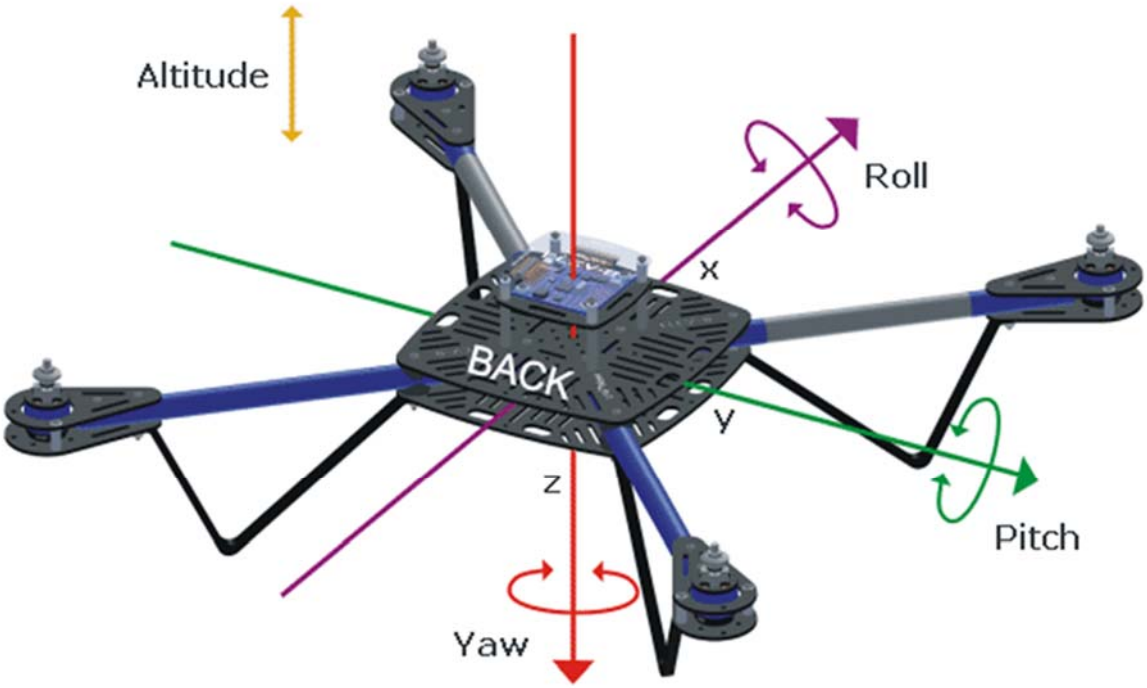


Figure 3.13 Center of Gravity of UAV.

3.4.5 PID

It is fundamentally a method utilized in programming and if made settings properly, can be incredibly effective and delicate. PID stands for Proportional Integral Derivative, three separate components joined together, though sometimes we do not require all three. For example, we could instead have just P control, PI control, PD control or PID control. Many flight controller software allow users to adjust PID values to get better performance of flight. PID is a function in the flight controller; it reads data from the sensors and inform the motors how fast they need to run. Finally, this is how the stability is obtained on UAV.

Proportional-Integral-Derivative (PID) is a closed loop control system that effort to get the actual result near or closer to the required result by regulating the input. The error is fed back to the beginning, and repeats the process, as shown in Fig (3.14) below.

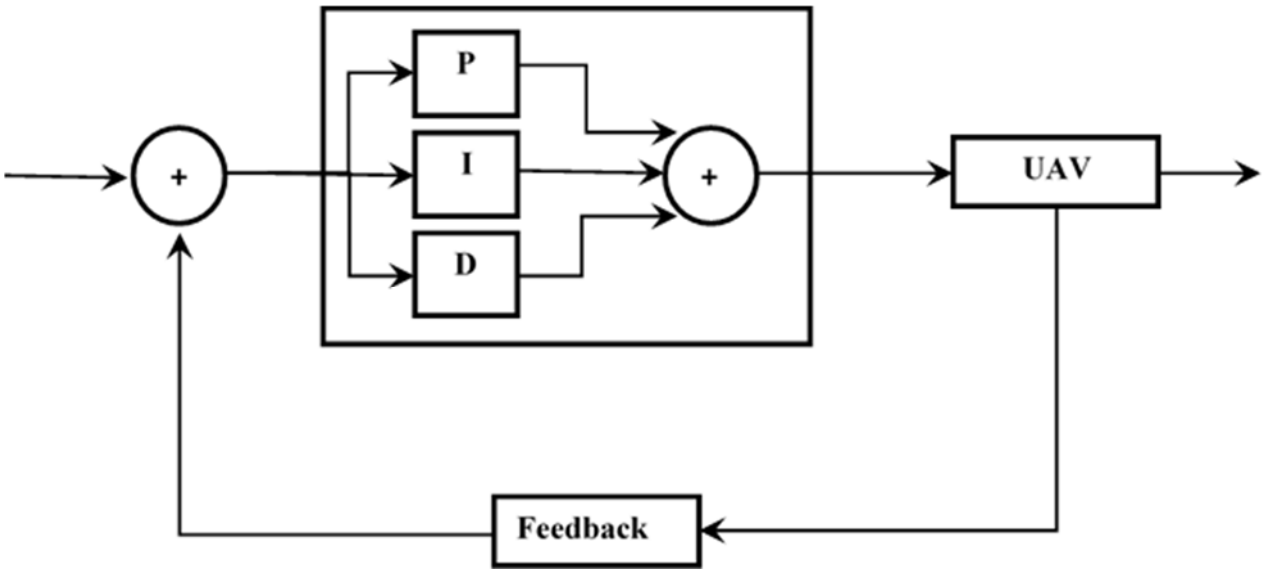


Figure 3.14 PID controller diagram.

UAV control is a mainly difficult and interesting problem. With six degrees of freedom in which three are translational and three are rotational and only four autonomous inputs which are rotor speeds, UAVs are severely underactuated. To achieve six degrees of freedom rotational and translational motion are mixed. The produced dynamics are highly nonlinear, particularly after accounting for the complicated aerodynamic effects. Finally, unlike ground vehicles, UAVs have very little friction to restrain their motion, so they must furnish their own damping in order to stop moving and stay stable. Simultaneously, these factors create a very motivating control problem.

3.4.5.1 UAV Dynamics

We will commence deriving UAV dynamics by introducing the two frames in which will operate. The inertial frame is defined by the ground, with gravity pointing in the negative z-direction. The body frame is defined by the orientation of the UAV, with the rotor axes pointing in the positive Z-direction and the arms pointing in the X and Y directions. Fig (3.15) shows the UAV body frame and inertial frame.

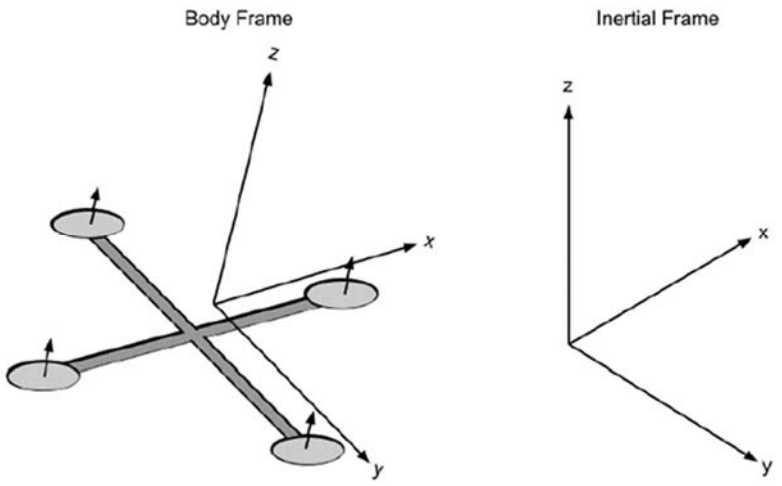


Figure 3.15 UAV Body frame and Inertial frame.

3.4.5.2 Kinematics

Let us formalize the kinematics in the body and inertial frames before delving into the physics of UAV motion. Position and velocity of UAV defined in the inertial frame as $position = (x, y, z)^T$and $elocity = (\dot{x}, \dot{y}, \dot{z})^T$. Similarly, the Yaw, Roll, and Pitch angles in the body frame was defined as $\theta = (\phi, \theta, \psi)^T$, with identical angular velocity as $\dot{\theta} = \left(\dot{\phi}, \dot{\theta}, \dot{\psi}\right)^T$. However, with the consideration that the angular velocity vector $\omega \neq \dot{\theta}$. The angular velocity is a vector pointed along the rotational axis, while $\dot{\theta}$ is the time derivative of Roll, Yaw and Pitch. To convert these angular velocities into angular velocity vector, we can utilize the following relation:

$$\omega = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix} \dot{\theta} \tag{3.4}$$

Where

ω is the angular velocity vector in the body frame.

The body frame can related with inertial frame by a rotation matrix R that goes from the body frame to the inertial frame. This matrix is determined by utilizing the ZYZ Euler angle conventions and successively “undoing” the Pitch, Roll and Yaw.

$$R = \begin{bmatrix} C_\phi C_\psi - C_\theta S_\phi S_\psi & -C_\psi S_\phi - C_\phi C_\theta S_\psi & S_\theta S_\psi \\ C_\theta S_\psi S_\phi + C_\phi S_\psi & C_\phi C_\theta C_\psi - S_\phi S_\psi & -C_\psi S_\theta \\ S_\phi S_\theta & C_\phi S_\theta & C_\theta \end{bmatrix} \quad (3.5)$$

For a given vector $\vec{v}$ in the body frame, the identical vector is given by $R\vec{v}$ in the inertial frame.

3.4.5.3 Physics

In order to fairly model the dynamics of the system, an understanding of the physical properties that govern it is needed. We will start with detailing of the motors being utilized for our UAV, and then use energy considerations to drive the forces and thrusts that these motors produced in the entire UAV. All motors on the UAV are indistinguishable, so we can investigate a single one without loss of generality.

3.4.5.4 Motors

All UAV applications are used brushless motors. The torque produced for our electric motors is given by:

$$\tau = K_t(I - I_0) \quad (3.6)$$

Where

τ : the motor torque.

I : Input current.

I_0 : The current when there is no load in the motor.

K_t : The torque relativity constant.

The voltage across the motor is the sum of the some resistive loss and the back-EMF:

$$V = IR_m + K_v\omega \quad (3.7)$$

Where

V: the voltage drop across the motor.

R_m : the motor resistance.

ω : the angular velocity of the motor.

K_v : a proportionality constant (indicating back-EMF generated per RPM).

The following description can be used to calculate the power consumes of our motors. The power is:

$$P = IV = \frac{(\tau + K_t I_0)(K_t I_0 R_m + \tau R_m + K_t K_v \omega)}{K_t^2} \tag{3.8}$$

Assuming negligible motor resistance. Then, the power becomes proportionate to the angular velocity:

$$P \approx \frac{(\tau + K_t I_0) K_v \omega}{K_t} \tag{3.9}$$

For further simplification, assuming that $K_t I_0 \ll \tau$. Since I_0 is the current when there is no load, even that is too small. This approximation is good enough. Thus, the final simplified equation for power is:

$$P \approx \frac{K_v}{K_t} \tau \omega \tag{3.10}$$

3.4.5.5 Forces

By keeping of energy, we realize that the energy of the motor consumes in a given duration is equal to the force generated on the propeller times the distance that the air it displaces moves ($P.dt = F.dx$). Equivalently, the *power* equals to the thrust times the air velocity $\left(P = \frac{dx}{dt}\right)$.

$$P = \mathrm{T}v_h \tag{3.11}$$

Assuming low vehicle speed, so v_h is the air velocity when flight. And also assuming that the free stream velocity, v_∞, is zero (the air in the surrounding environment is stationary proportional to the UAV). Momentum theory gives us the equation for flight velocity as a function of thrust.

$$v_h = \sqrt{\frac{\mathrm{T}}{2\rho A}} \tag{3.12}$$

Where

ρ : the density of the surrounding air.

A : the area swept out by the rotor.

By using simplified equation of power:

$$P = \frac{K_v}{K_t}\tau\omega = \frac{K_v K_\tau}{K_t}\mathrm{T}\omega = \frac{\mathrm{T}^{\frac{3}{2}}}{\sqrt{2\rho A}} \tag{3.13}$$

In the general case, $\tau = \vec{r} \times \vec{F}$; in this case, the torque is relative to the thrust T by some constant ratio K_τ determined by the code configuration and parameters. For solving the thrust magnitude *T*, thrust is proportional to the square of angular velocity of the motor obtained:

$$\mathrm{T} = \left(\frac{K_v K_\tau \sqrt{2\rho A}}{K_t}\,\omega\right)^2 = k\omega^2 \tag{3.14}$$

Where *k* is some fitly dimensioned constant. Summing over all motors, we find that the total thrust on the UAV (in the body frame) is given by:

$$\mathrm{T}_B = \sum_{i=1}^{4} \mathrm{T}_i = k \begin{bmatrix} 0 \\ 0 \\ \sum \omega_i^2 \end{bmatrix} \tag{3.15}$$

3.4.5.6 Torques

Since we have processed the powers on the UAV, we might like to figure the torques. Every rotor contributes some torque about the body Z-axis. This torque is the torque required to keep propeller turning and providing thrust; it makes the immediate angular acceleration and defeats the frictional drag forces. From fluid dynamics the drag equation gives us the frictional force:

$$F_D = \frac{1}{2}\,\rho C_D A v^2 \tag{3.16}$$

Where

ρ: the surrounding fluid density.

A: the reference area (propeller cross-section, not area swept out by the propeller).

C_D: dimensionless constant.

This, while just exact in some at times, is good enough for our motivations. This infers the torque because of drag is given by:

$$\tau_D = \frac{1}{2} R_\rho C_D A v^2 = \frac{1}{2} R_\rho C_D A (\omega R)^2 = b\omega^2 \tag{3.17}$$

Where

ω: the angular velocity of the propeller.

R: the radius of the propeller.

b: some appropriately dimensioned constant.

Note that we've expected that all the force is applied at the tip of the propeller, which is certainly inaccurate; in any case, the main outcome that issues for our motivations is that the drag torque is corresponding to the square of the angular velocity. We would then be able to compose the entire torque about the Z-axis for the *i*th motor:

$$\tau_z = b\omega^2 + I_M\dot{\omega} \tag{3.18}$$

Where

I_M : the moment of inertia about the motor Z-axis.

$\dot{\omega}$: the angular acceleration of the propeller.

b : drag coefficient.

Note that in stable status flight (not landing or taking off) $\omega \dot{\approx} 0$, since most of the time the propellers will be preserving a constant (or nearly constant) thrust and won't be accelerating. Thus, simplifying the entire expression to:

$$\tau_z = (-1)^{i+1} b\omega_i^2 \tag{3.19}$$

Where the $(-1)^{i+1}$ negative for the *i*th propeller if they spinning CCW and positive if its spinning CW. The sum of all the torques from each propeller gives us the total torque about the Z-axis.

$$\tau_\psi = b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{3.20}$$

From standard mechanics, the Pitch and Roll torques are derived. The i =1 and i =3 motors arbitrarily chosen to be on the Roll axis, so

$$\tau_\phi = \sum r \times \mathrm{T} = L(k\omega_1^2 - k\omega_3^2) = Lk(\omega_1^2 - \omega_3^1) \tag{3.21}$$

Correspondingly, a similar expression gives the Pitch torque:

$$\tau_\theta = Lk(\omega_2^2 - \omega_4^2) \tag{3.22}$$

Where

L: the distance from the center of the UAV to any of the propellers. So, the torques in the body frame are:

$$\tau_B = \begin{bmatrix} Lk(\omega_1^2 - \omega_3^2) \\ Lk(\omega_2^2 - \omega_4^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \quad (3.23)$$

3.4.5.7 Equations of Motion

The acceleration of the UAV is due to thrust, gravity, and linear friction in the inertial frame. The thrust vector in the inertial frame can be obtained by using the rotational matrix R to map the thrust vector from the body frame to the inertial frame. So, linear motion can summarized as:

$$m\ddot{x} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R\mathrm{T}_B + F_D \quad (3.24)$$

Where

$\vec{x}$: the position of the UAV.

g: the acceleration due to gravity.

F_D: the drag force.

T_B: the thrust vector in the body frame.

While it is advantageous to have the linear equations of motion in the inertial frame, the rotational equations of motion are helpful to us in the body frame, so we can express rotations about the center of the UAV rather than about our inertial center. We determined the rotational equations of movement from Euler's equations for inflexible body dynamics. Expressed in vector form, Euler's

Equations are written as:

$$I\dot{\omega} + \omega \times (I\omega) = \tau \quad (3.25)$$

Where

ω : the angular velocity vector.

I : the inertia matrix.

τ : vector of external torques.

We can rewrite this as :

$$\dot{\omega} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = I^{-1}\big(\tau - \omega \times (I\omega)\big) \tag{3.26}$$

We can show our UAV as two thin uniform rods crossed at the source with a point mass (motor) at the end of each. In view of this present, obviously, the symmetries result in a diagonal inertia matrix of the frame

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.27}$$

Thus, we obtain our last result for the body frame rotational equations of movement

$$\dot{\omega} = \begin{bmatrix} \tau_\emptyset I_{xx}^{-1} \\ \tau_\theta I_{yy}^{-1} \\ \tau_\psi I_{zz}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}} \omega_y \omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}} \omega_x \omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}} \omega_x \omega_y \end{bmatrix} \tag{3.28}$$

3.4.5.8 Control

The motivation behind deriving a numerical model of a UAV is to help with creating controllers for physical UAVs. The inputs to our framework comprise of the angular velocities of every rotor since everything we can control is the voltages over the motors. Note that we just utilized the square of the angular velocity, ω_i^2 and never the angular velocity itself, ω_i. For notational effortlessness, let us present the inputs $\gamma_i = \omega_i^2$. Since we can set ω_i we can obviously set γ_i also. With this, we can compose our system as a first order differential equation in state space. Suppose x_1 be the position in space of the UAV, x_2 be the UAV linear velocity, x_3 be the Roll, Pitch, and Yaw angles, and x_4 be the angular velocity vector. (al of these are 3-vectors.) With these being our state, we can compose the state space equations for the development of our state.

$$\dot{x}_1 = x_2 \tag{3.29}$$

$$\dot{x}_2 = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R \mathrm{T}_B + \frac{1}{m} F_D \tag{3.30}$$

$$\dot{x}_3 = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix}^{-1} x_4 \tag{3.31}$$

$$\dot{x}_4 = \begin{bmatrix} \tau_\emptyset I_{xx}^{-1} \\ \tau_\theta I_{yy}^{-1} \\ \tau_\psi I_{zz}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}} \omega_y \omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}} \omega_x \omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}} \omega_x \omega_y \end{bmatrix} \tag{3.32}$$

3.4.5.9 PD Control

In order to control UAV, we will utilize a PD control, with a component proportional to the error between our coveted path and the observed path, and a component proportional to the derivative of error. Because our UAV only have a gyro, so we only be able to utilize the angle derivatives $\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$ in our controller; the measured values gives us the derivative of error, and their integral provides us the actual error. We might want to stabilize the UAV in a level position (horizontal), so our desired velocity and angles will all be zero. Torques are associated to angular velocities by $\tau = I\ddot{\theta}$, so we might want to set the torques proportional to the output of controller, with $\tau = Iu(t)$. Thus,

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} -I_{xx}\left(K_d\dot{\phi} + K_p \int_0^{\mathrm{T}} \dot{\phi}\, dt\right) \\ -I_{yy}\left(K_d\dot{\theta} + K_p \int_0^{\mathrm{T}} \dot{\theta}\, dt\right) \\ -I_{zz}\left(K_d\dot{\psi} + K_p \int_0^{\mathrm{T}} \dot{\psi}\, dt\right) \end{bmatrix} \tag{3.33}$$

We have beforehand determined the relationship between torque and inputs, so we realize that

$$\tau_B = \begin{bmatrix} Lk(\omega_1^2 - \omega_3^2) \\ Lk(\omega_2^2 - \omega_4^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} = \begin{bmatrix} -I_{xx}\left(K_d\dot{\phi} + K_p\int_0^{\mathrm{T}}\dot{\phi}\,dt\right) \\ -I_{yy}\left(K_d\dot{\theta} + K_p\int_0^{\mathrm{T}}\dot{\theta}\,dt\right) \\ -I_{zz}\left(K_d\dot{\psi} + K_p\int_0^{\mathrm{T}}\dot{\psi}\,dt\right) \end{bmatrix} \quad (3.34)$$

This gives us an arrangement of three equations with four unknowns. We can compel this by enforcing the constraint that our information sources (inputs) must keep the UAV aloft:

$$\mathrm{T} = mg \quad (3.35)$$

Note that this equation eliminates the fact that the thrust won't be pointed straightforwardly up. This will restrict the applicability of our controller, however, should not cause real problems for small deviations from stability. If we had a way of deciding the current angle precisely, we can recompense. If our gyro is precise enough, we can integrate the values obtained from the gyro to obtain the angles θ and ϕ. In this case, we can compute the thrust necessary to keep the UAV aloft by projecting the thrust *mg* onto the inertial z-axis. We find that

$$\mathrm{T}_{proj} = \mathrm{mg}\cos\theta\cos\phi \quad (3.36)$$

Therefore, with an exact angle measurement, we can rather uphold the necessity that the thrust be equal to

$$\mathrm{T} = \frac{mg}{\cos\theta\cos\phi} \quad (3.37)$$

In which case the component of the thrust pointing along the positive z-axis will be equivalent to *mg*. We realize that the thrust is relative to a weighted sum of the inputs:

$$\mathrm{T} = \frac{mg}{\cos\theta\cos\phi} = k\sum\gamma_i \implies k\sum\gamma_i = \frac{mg}{k\cos\theta\cos\phi} \quad (3.38)$$

With this additional imperative, we have a set of four linear equations with four unknowns γ_i. By solving for each γ_i, the following input values obtained:

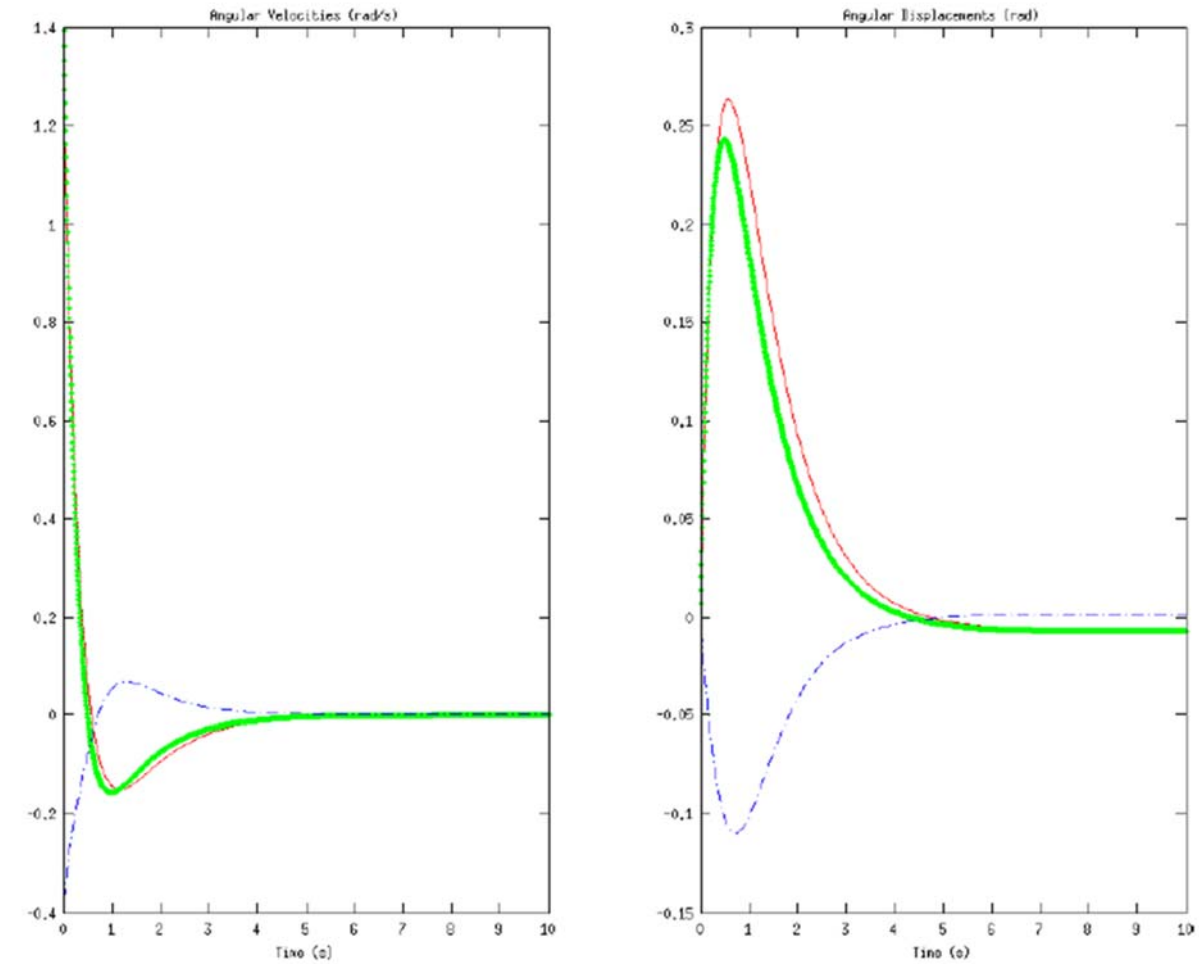
$$\gamma_1 = \frac{mg}{4k\cos\theta\cos\emptyset} - \frac{2be_{\emptyset}I_{xx}+e_{\psi}I_{zz}kL}{4bkL} \quad (3.39)$$

$$\gamma_2 = \frac{mg}{4k\cos\theta\cos\emptyset} + \frac{e_{\psi}I_{zz}}{4b} - \frac{e_{\theta}I_{yy}}{2kL} \quad (3.40)$$

$$\gamma_3 = \frac{mg}{4k\cos\theta\cos\emptyset} - \frac{-2be_{\phi}I_{xx}+e_{\psi}I_{zz}kL}{4bkL} \quad (3.41)$$

$$\gamma_4 = \frac{mg}{4k\cos\theta\cos\emptyset} + \frac{e_{\psi}I_{zz}}{4b} + \frac{e_{\theta}I_{yy}}{2kL} \quad (3.42)$$

This is an entire detailing for PD controller. The controller drives the angular velocities and angles to zero, as shown in Fig (3.16) bellow.



Angular velocities **Angular displacements**

Figure 3.16 Angular velocities and angular displacements. ϕ, θ, ψ are coded as red, green, and blue.

Nevertheless, note that the angles are not entirely driven to zero. The average steady state error is approximately 0.3 o . This is a prevalent problem with utilizing PD controllers for mechanical systems and can be partially alleviated with a PID controller.

In addition, take note of that since we are just controlling angular velocities, our positions and linear velocities don't go to zero. Nonetheless, the z position will stay consistent, in light of the fact that we have obliged the aggregate vertical thrust to be such that it keeps the UAV perfectly aloft, without ascending or descending.

3.4.5.10 PID Control

PD controllers are suitable in their straightforwardness and simplicity of implementation, but they are frequently deficient for controlling mechanical systems. Particularly in the presence of noise and disturbances, PD controllers will often lead to steady state error. A PID control is a PD control with another term included, which is corresponding to the integral of the process variable. Including an integral term makes any remaining steady-state error to develop and enact a change, so a PID controller ought to have the ability to track our path (and stabilize the UAV) with an essentially smaller steady-state error. The equations stay identical to the ones displayed in the PD case, but with an extra term in the error:

$$e_{\phi} = k_d\dot{\phi} + k_p\int_0^{\mathrm{T}}\dot{\phi}dt + k_i\int_0^{\mathrm{T}}\int_0^{\mathrm{T}}\dot{\phi}\,dt\,dt \tag{3.43}$$

$$e_{\theta} = k_d\dot{\theta} + k_p\int_0^{\mathrm{T}}\dot{\theta}dt + k_i\int_0^{\mathrm{T}}\int_0^{\mathrm{T}}\dot{\theta}\,dt\,dt \tag{3.44}$$

$$e_{\psi} = k_d\dot{\psi} + k_p\int_0^{\mathrm{T}}\dot{\psi}dt + k_i\int_0^{\mathrm{T}}\int_0^{\mathrm{T}}\dot{\psi}\,dt\,dt \tag{3.45}$$

However, PID controls come with their own inadequacies. One trouble that ordinarily happens with a PID control is known as integral windup. In some cases, integral wind-up can cause stretched oscillations instead of settling. In other cases, wind-up may indeed

cause the system to become unstable, instead of taking longer to reach a steady state, as shown in Fig (3.17) below.

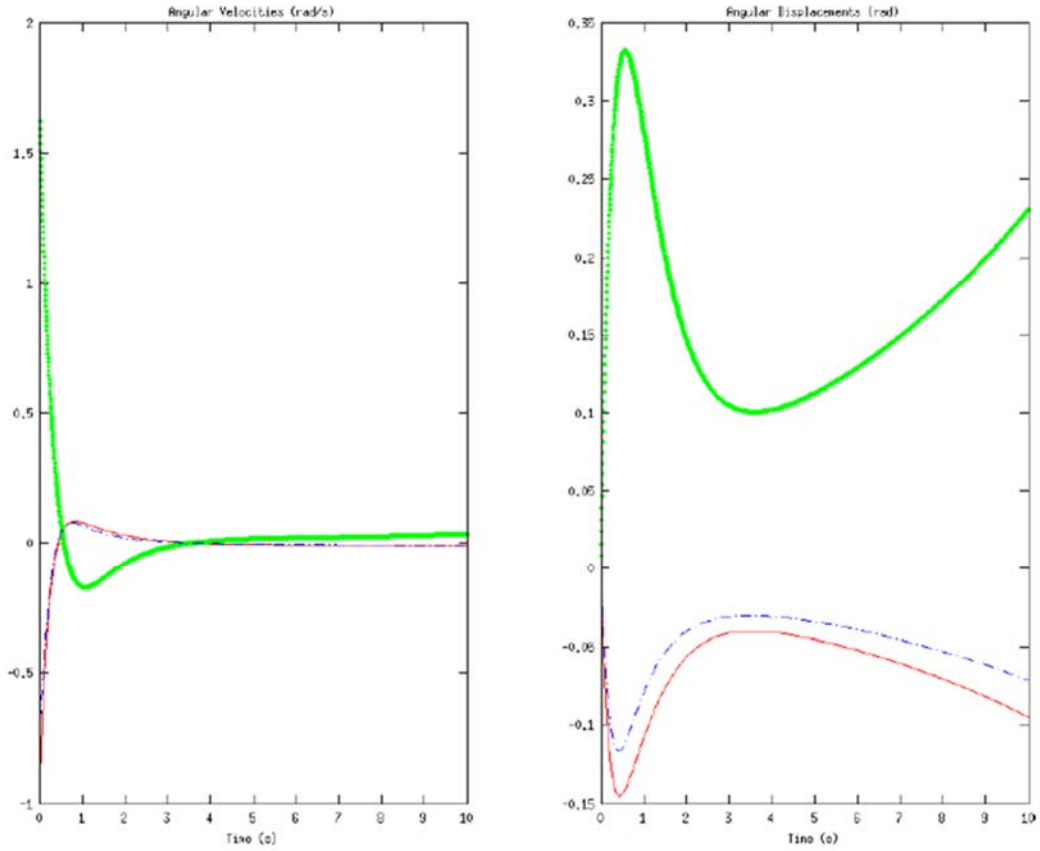


Figure 3.17 PID controller.

If there is a large trouble in the process variable, this large trouble is integrated over time, becoming a still bigger control signal (due to the integral term). However, even once the system stabilizes, the integral is still big, therefore making the controller overshoot its objective. It may then start a series of dieing down oscillations, become unstable, or basically take an incredibly long time to reach a steady state. In order to avert this, we disable the integral function until we reach something near to the steady state. When we are in a controllable region near the desired steady state, we turn on the integral function, which pushes the system towards a minimal steady-state error, as shown in Fig (3.18).

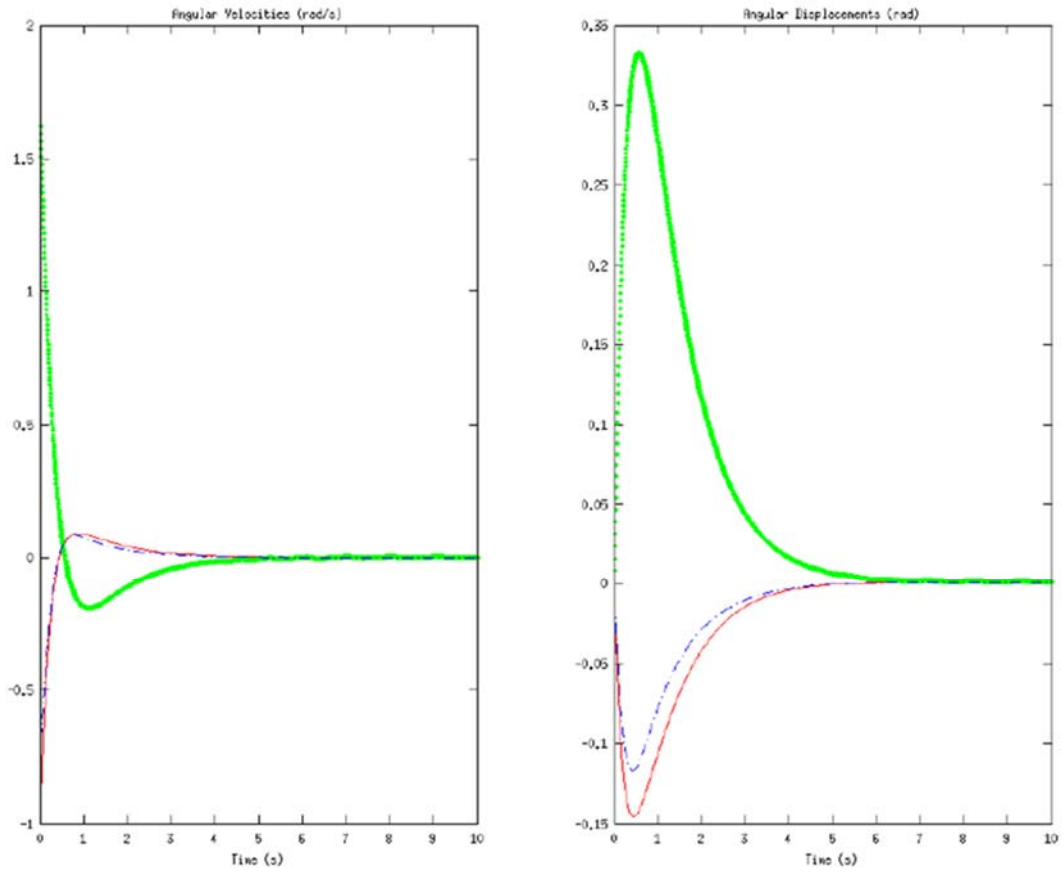


Figure 3.18 With a properly implemented PID, we achieve an error of approximately 0.06 after 10 seconds.

3.5 Remote Control (RC)

There are many types of remote controls in markets, in our thesis we chose Fly Sky – T6 6 channels 2.4 GHz AFHDS computerized digital proportional R/C airplane and helicopter system.

AFHDS: Stands for "Automatic Frequency Hopping Digital System". This highly developed radio transmission system will assure a long range, jamming free and long battery life experience. [53]

RF specifications:

Our radio system (remote control) works in the frequency range 2.4000 to 2.4835 GHz. 500 KHz channel bandwidth, this band has been divided into 160 independent channels. Each radio system uses 16 different channels and 160 different types of hopping

algorithm. And uses less than 20 dBm (100 mV) from RF power and GFSK modulation type, with -1058 dBm of RX sensitivity. This radio system uses a high gain and high quality multidirectional antenna. It covers the whole frequency band, also assure a jamming free long radio transmissions. Fig (3.19) shows fly-sky remote control (transmitter and receiver).

Figure 3.19 Fly – Sky remote control.

In this thesis, we connect the receiver of remote control to the UAV by using servo wires; we connect it with the flight controller as shown in Fig (3.20) below.

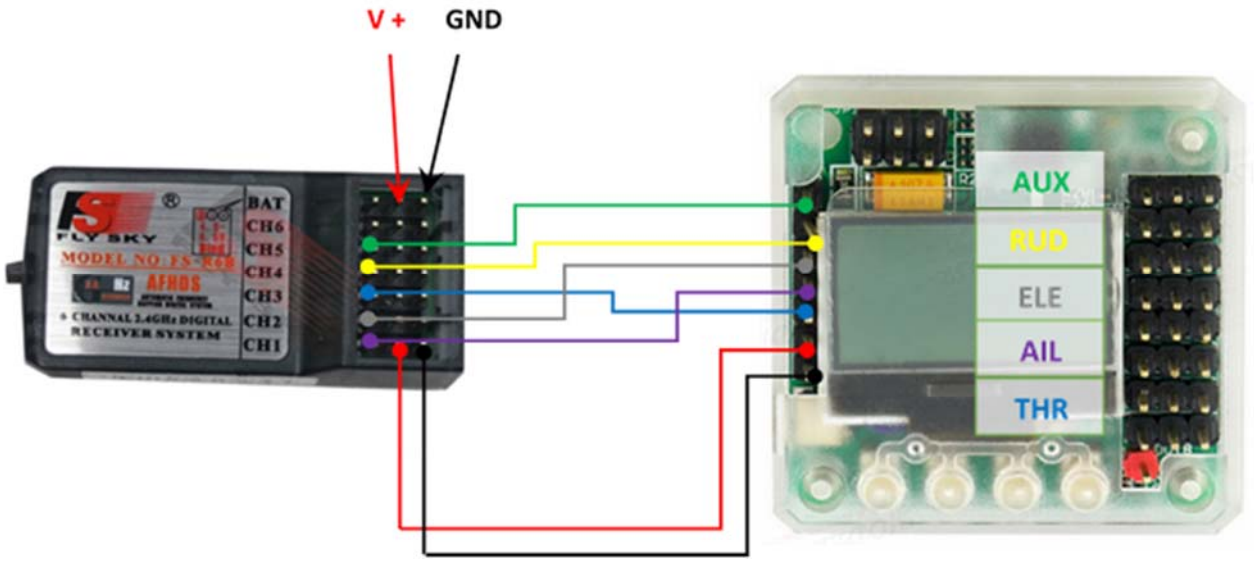


Figure 3.20 diagram shows the connection between RC receiver and flight controller.

3.6 Reading signals from RC receiver

Our contribution is designing a system that uses internet of things to make the control for Unmanned Aerial Vehicle (drone) from an infant distance by simulating the signals coming from RF receiver and remote control. So, we need to read the signals from RF receiver, actually we read it in two types:

First type: we connect the receiver to the oscilloscope and take signals and all information from that signals, but these signals not accurate enough as shown in Fig(3.21).

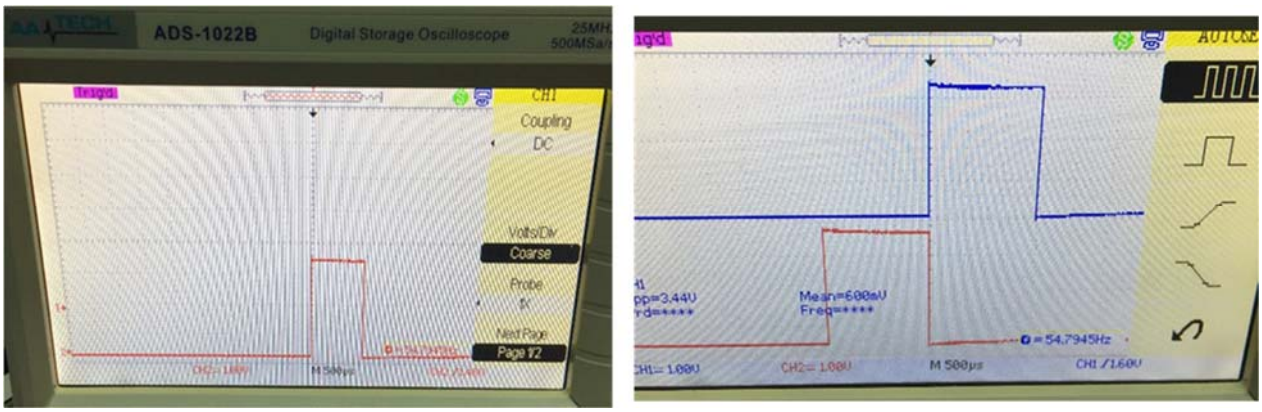


Figure 3.21 Oscilloscope signals from RF receiver.

Second type: we connect the RF receiver to the PWM pins in the Arduino, and we wrote a code for Arduino to read the signals from PWM pins. In Arduino to read the total pulse we need to read the high and the low duration of the pulse, and we get:

In High duration pulse: THR changes from 988 microsecond (down) to 1965 microsecond (up) (left bar in the remote), AIL (Roll) changes from (1250) microsecond (go to left side) to (1735) microsecond (go to right side) (right bar in remote) and the midpoint is (1495) microsecond. ELE (Pitch) changes from (1250) microsecond (go to forward) to (1735) microsecond (go to backward) (right bar in remote) and the midpoint is (1496) microsecond. RUD (Yaw) changes from (1250) microsecond (rotate to right) to (1739) microsecond (rotate to left) (left bar in remote) and the midpoint (1495) microsecond. AUX changes by the upper left tuning (VRB) from (983) microsecond (tune to the positive side) to (1907) microsecond (tune to the negative side) and the Midpoint (1373) microsecond (0 point), Fig (3.22) shows the diagram of high period pulses and Fig (3.23) shows the diagram of low period pulses.

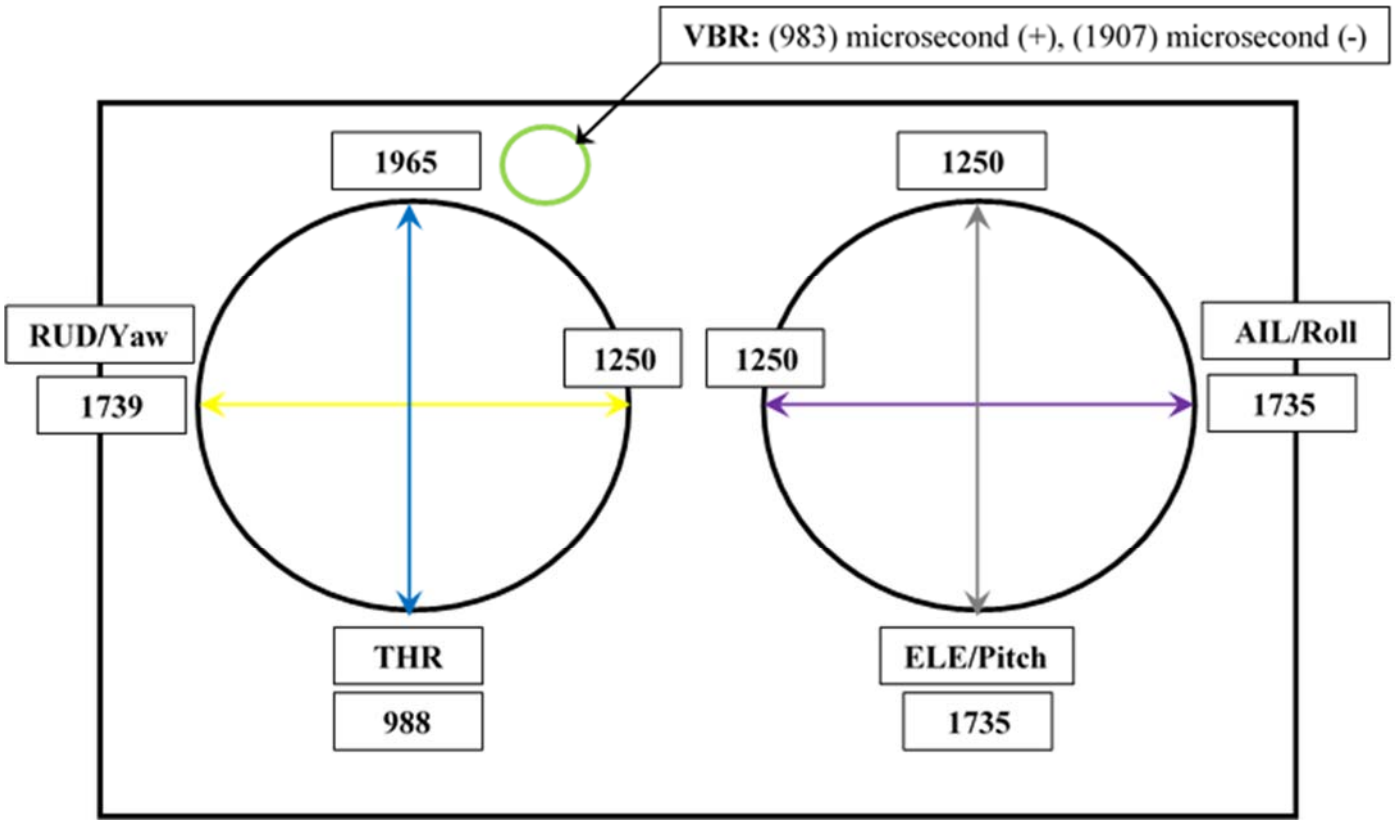


Figure 3.22 High period pulses in microsecond.

In Low duration pulse: THR changes from 17147 microsecond (down) to 16182 microsecond (up) (left bar in the remote), AIL (Roll) changes from (16894) microsecond (go to left side) to (16409) microsecond (go to right side) (right bar in remote) and the midpoint is (16693) microsecond. ELE (Pitch) changes from (16894) microsecond (go to forward) to (16412) microsecond (go to backward) (right bar in remote) and the midpoint is (16693) microsecond. RUD (Yaw) changes from (16893) microsecond (rotate to right) to (16406) microsecond (rotate to left) (left bar in remote) and the midpoint (16695) microsecond. AUX changes by the upper left tuning (VRB) from (17153) microsecond (tune to the positive side) to (16236) microsecond (tune to the negative side) and the Midpoint (16688) microsecond (0 point), Fig (3.22) shows the diagram of high period pulses and Fig (3.23) shows the diagram of low period pulses.

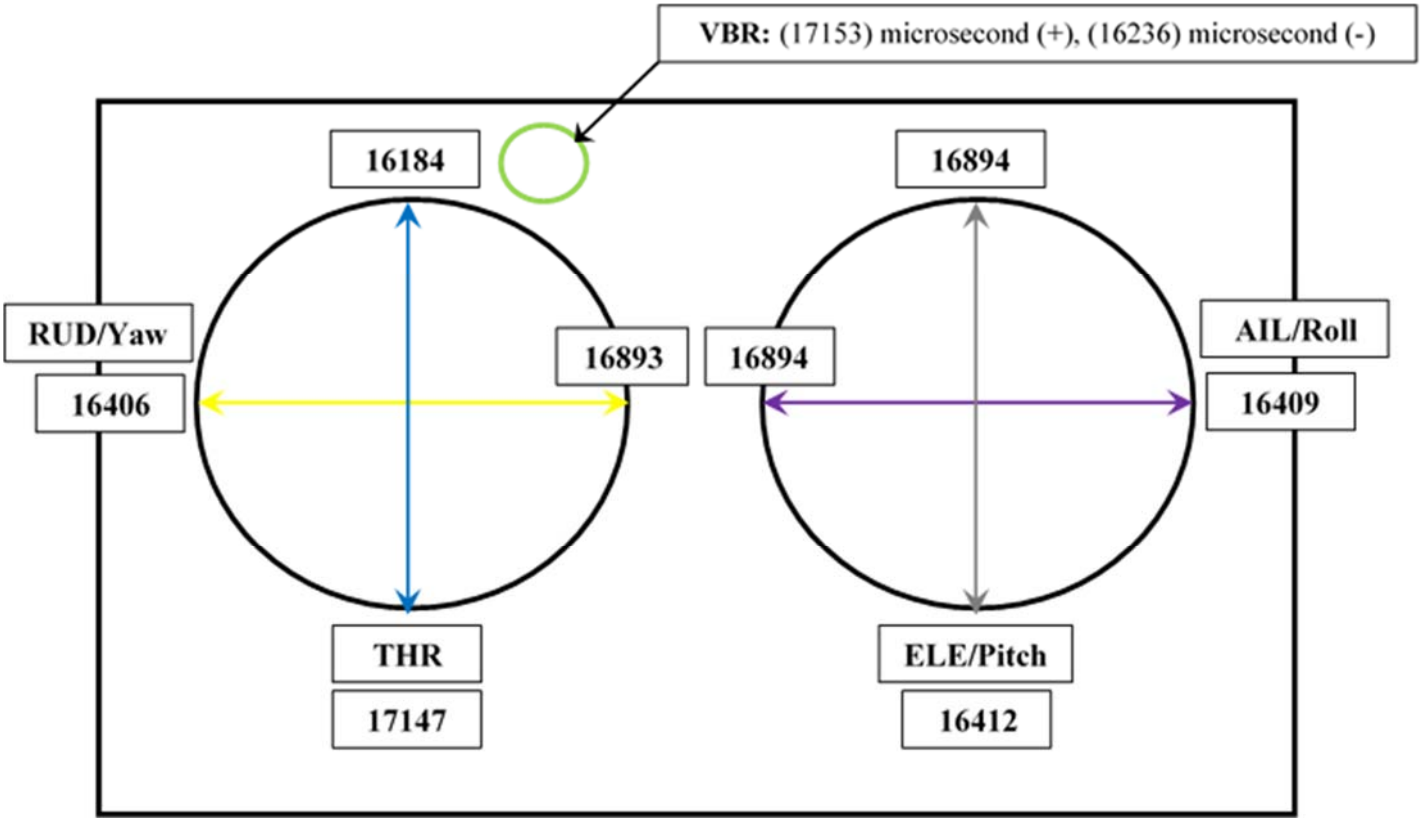


Figure 3.23 Low period pulses in microsecond.

We summarized the high and low durations of the pulses in the following table (3.1).

Table 3.1 High and low durations of pulses read from RF remote control.

		High duration of the pulse in µsec	**Low duration of the pulse in µsec**
THR	Min	988	17147
	Max	1965	16184
Yaw/RUD	Right	1250	16893
	Left	1739	16406
	Mid	1495	16695
Roll/AIL	Right	1735	16409
	Left	1250	16894
	Mid	1495	16693
Pitch/ELE	Forward	1250	16894
	Backward	1735	16412
	Mid	1496	16693

3.7 Generating PWM by using Arduino Uno and Raspberry Pi3

In order to achieve our contribution (IoT UAV) we replaced the RF remote control, by utilizing Arduino Uno and raspberry pi3 because RF remote control covers a limited distance and our contribution is to make our UAV infinite distance controlling by using the Internet.

We used Raspberry Pi3 for connection and designed a GUI (Graphical User Interface) to control the UAV. GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.

Our GUI was written in python programming language, and control the UAV through it. Figure (3.24) shows our GUI.

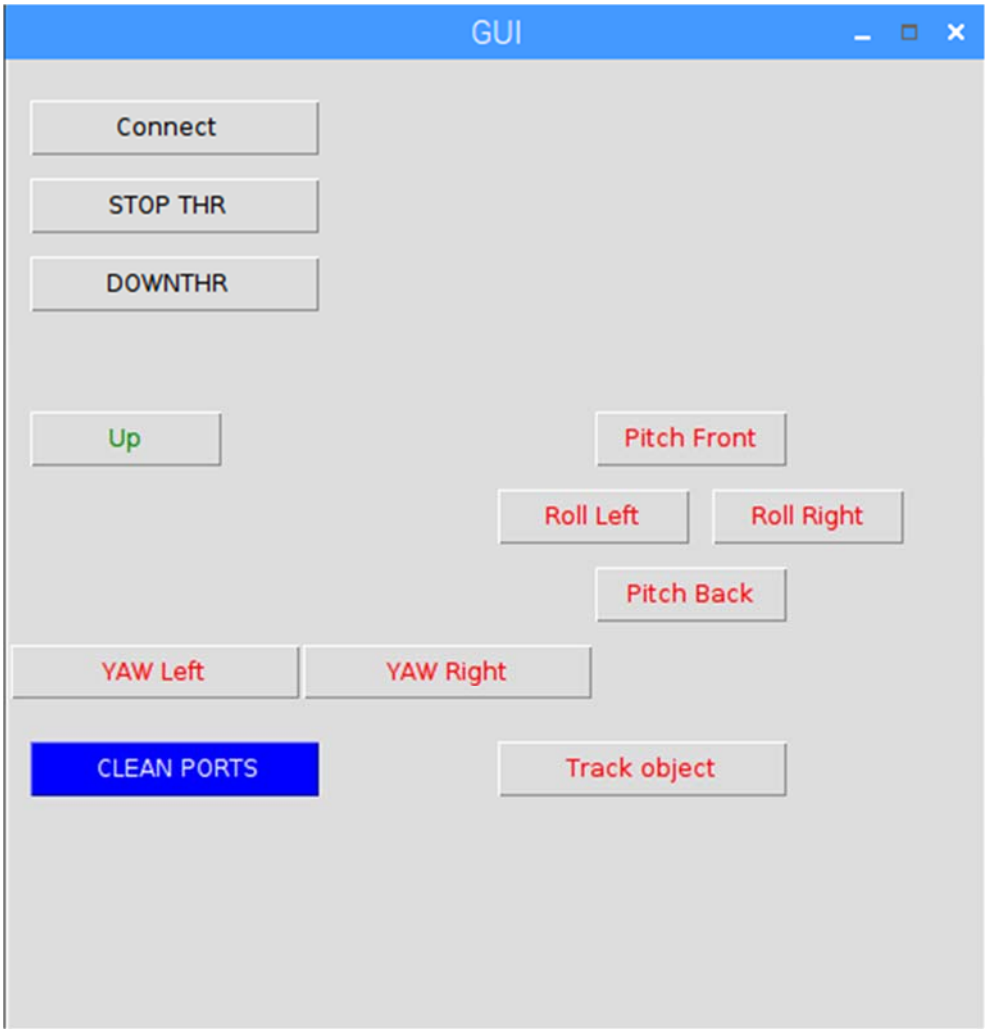


Figure 3.24 Designed GUI.

Because of our flight controller works in 5v and the Raspberry pi works in 3.3v therefore we need another electronic card that support 5v, this card is a microcontroller card Arduino Uno which gives us 5v with PWM ports.

Our procedure is to control the UAV from Raspberry Pi3 using GUI in python, python give orders to the microcontroller card (Arduino Uno) which the last generate the right pulses needed as we read it from the table (3.1) above, and send it to the flight controller.

3.8 Connection Establishment and object detection

In order to establishment a connection between UAV and the earth station (desktop or laptop) we used TeamViewer application. The TeamViewer have many advantages over using real IP. One of them it is free to use while the real IP providers requires fees per period. Also the TeamViewer is more easer in installation and use rather than real IP.

The earth station controlled the Raspberry Pi by running its operating system directly through TeamViewer and run the GUI of our UAV with all its instructions.

For more features, the UAV able to tracking objects by using Raspberry Pi camera and based on the BGR color moment calculations.

3.9 Color detection in OpenCV and Python

Color detection is the process of finding certain color and extracts it from its surroundings in order to detect objects having the same colors. The process in OpenCV is constructed using (cv2.inRange) function which is built in function and this function detects colors in a rage between lower and upper values that are provided for certain color range like blue for example in our work, in the following Figure (3.25) our detected object based on color.

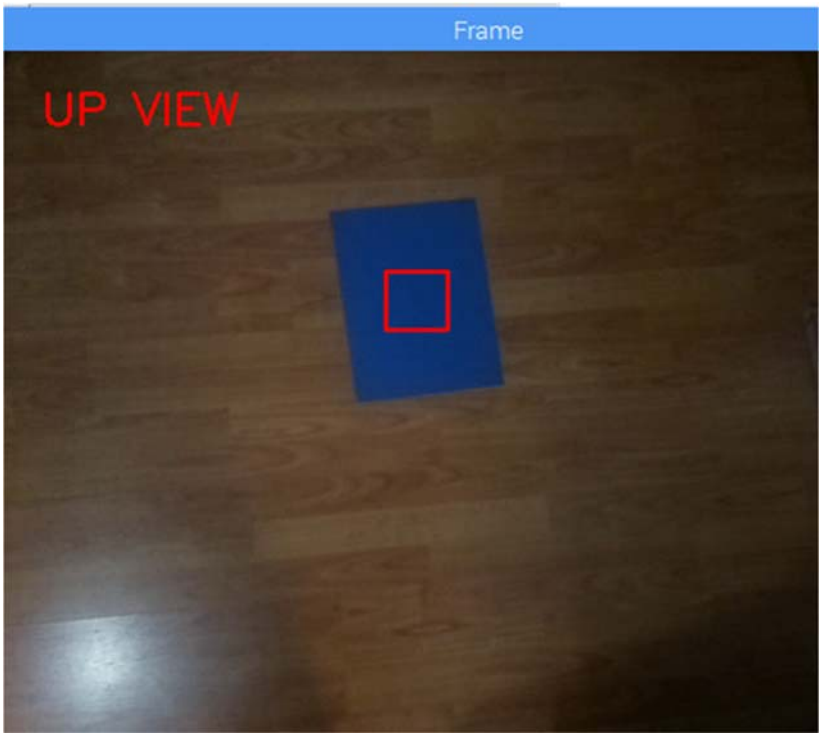


Figure 3.25 Detected object based on color.

In color detection it is recommended to convert the captured image from 'BGR' color range to HSV color range because it is more easer to isolate colors from each other's.

In numerical representation in our code:

`lower = np.array([76,31,4])` represents the lower value for the color that we intended to detect.
`upper = np.array([210,90,70])` represents the upper value for the color that we intended to detect.

`thresh = cv2.inRange(blur, lower, upper)` this part will detect the color in the image captured by camera that is limited between the lower and upper values of that is provided by us.

Then we will find the contours in the image or frame under test by the following line in our code:

```
contours,hierarchy=  cv2.findContours(thresh,cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
```

After that we will find the contour with the maximum area and we will consider it as our goal as shown in the following lines in code; finding contour with maximum area and store it as best_cnt:

```
max_area = 0
best_cnt = 1
for cnt in contours:
    area = cv2.contourArea(cnt)
    if area > max_area:
       max_area = area
       best_cnt = cnt
```

After that we will find the centroid of the best contour and we will draw a box surrounding it as shown in previous Figure (3.25).

```
M = cv2.moments(best_cnt)
cx,cy = int(M['m10']/M['m00']), int(M['m01']/M['m00'])
```

CHAPTER 4

SIMULATION

4.1 Quad copter simulation in Simulink

The simulation process should be made before the construction of Quad copter, for that reason the simulation was made in Simulink to study the effect of weights, torques, forces, resulted from the frame weight, motors rotation speed, electronic card's weight, also the selecting of PID controller parameters and fine tuning these parameters.

The simulation process is made using Sims cape multi body, before that CAD model should be imported from one of the mechanical 3d construction design programs, like solid works or others.

Figure (4.1) shown below represents the frame 3D design for our quad copter and it will be used in our study.

Figure 4.1 drone final mechanical model imported from auto cad or 3D max

The quad copter consists of many collected items which will represent the full Drone body like shown in previous figure.

The importance of importing the drone as separated parts forming the full body is to control some of these parts and leaving others like rotating the motor shaft and keeping the motor body stable same as in reality.

The process of controlling the Quad copter or making flight controller for it consists of many stages starting with inputs which are coming from the sensors which represents the values of (Roll, Pitch, Yaw, Altitude) as shown in figure (4.2) below:

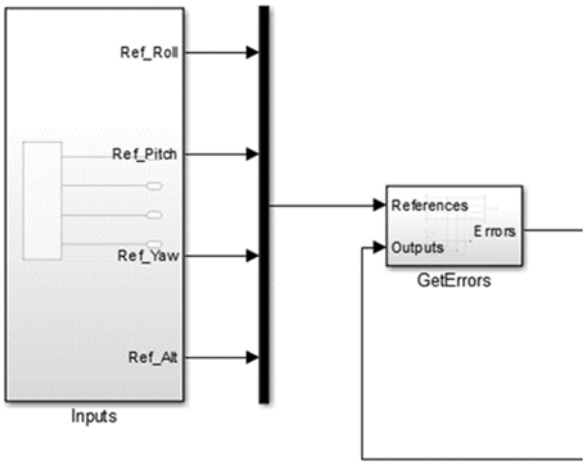


Figure 4.2 the sensor outputs

The main benefit of sensor values is to give feed back to the controller to check the level of the drone axis, for example if we have the value of Pitch is (-22) it means that the drone is turning to right side and if we will leave it turning for unwanted period of time that means it will fall down, so the controller is responsible of returning the drone to the horizontal plane to prevent it from falling down.

The same procedure will be applied for (Yaw, Roll and Altitude) to keep the drone stable in its flight.

The most important issue is that the input coming from the IMU which will represent the values of (Yaw, Roll, Altitude), these values should be accurate to provide the flight controller with suitable values to insure the best flight, the inputs are simulated by using the input block shown in figure (4.2) above.

The next step in the flight controller procedure is to calculate the error for (Yaw, Roll, Pitch and Altitude), error means the difference between the desired values and the measured ones, this step is made by the (GetErrors) block drown in figure(4.2) above.

After that the error values will be injected to flight controller in order to be evaluated and to be corrected as shown in figure (4.3) below.

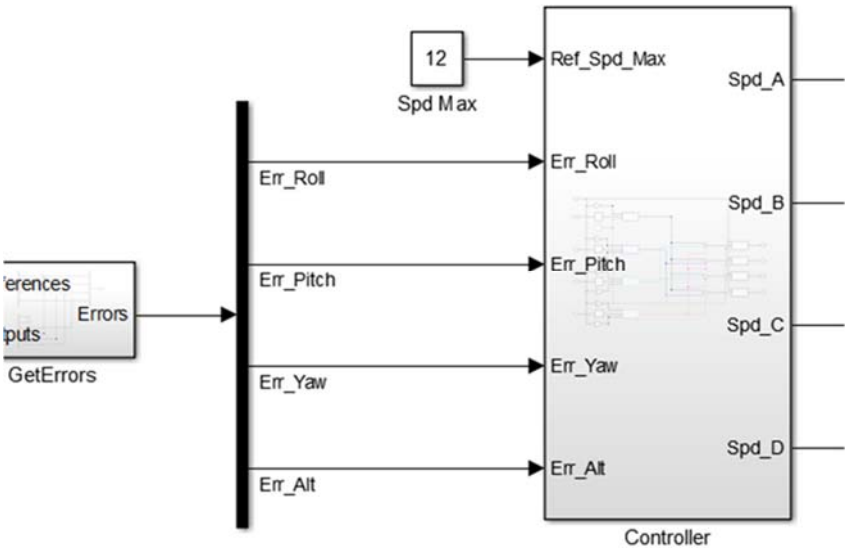


Figure 4.3 the injected values from error calculating block to the flight controller.

Inside the flight controller, a correcting process will be made to reduce the error for (Yaw, Roll, Pitch and Altitude) smoothly using PID algorithm for each value separately. The main purpose for applying simulation for drone is to estimate the correct values for (P gain, I gain, D gain) in a way that prevents the drone from doing extreme actions or to go unstable during flight.

To go further in our explanation we will show the inner circuits inside controller to show the hall procedure made by our system, as shown in Figure (4.4) below.

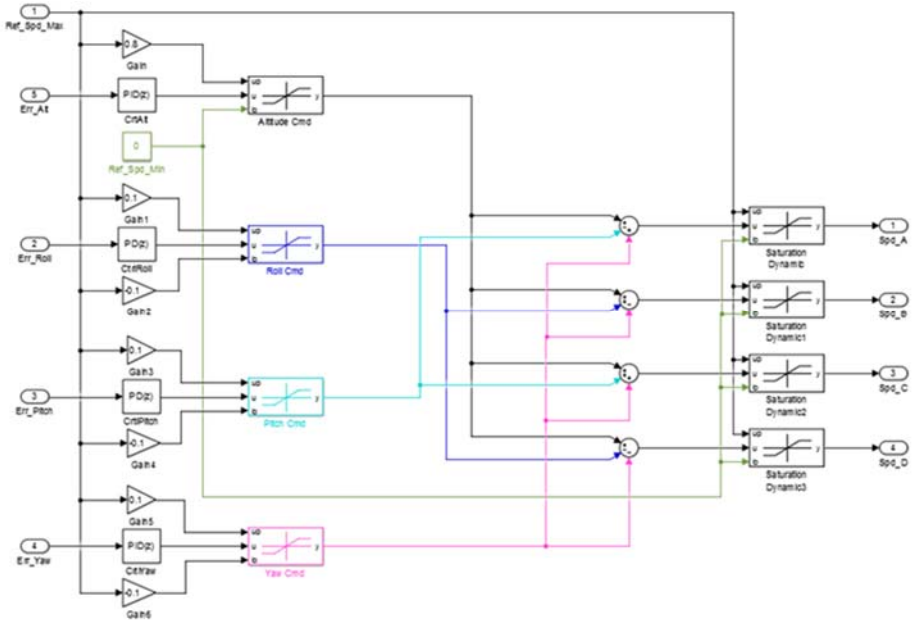


Figure 4.4 the PID controller for (Yaw, Roll, Pitch and Altitude).

To explain the operation of the circuit we will start with the first section as shown in figure (4.5) below:

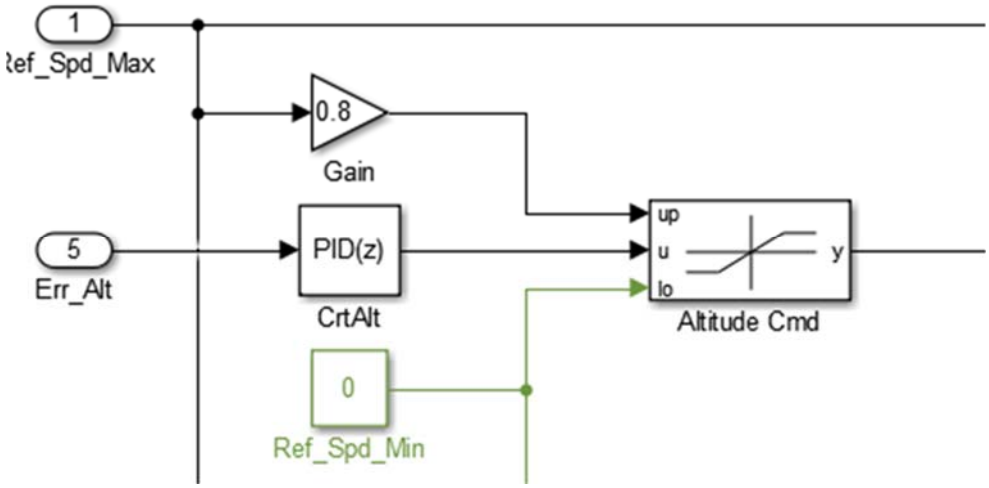


Figure 4.5 first section inside controller.

The PID controller for (z) means the PID for altitude; we can see that PID controller takes its input from the error signal (Err_Alt) which represents the difference between the actual altitude and the desired one, inside PID controller a PID control will be applied by selecting the values related to it as follows:

Proportional (P): 4.5

Integral (I):10

Derivative (D): 1

The output of PID controller is shown in figure (4.6) below and as we can see that both of its, stability and transient time is suitable (the most important goals of PID control), we will discuss the tuning effect on behavior later on.

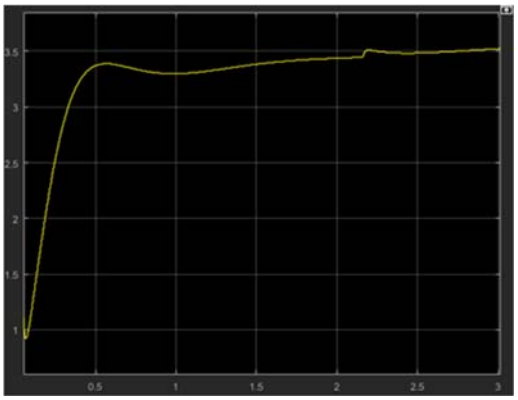


Figure 4.6 PID (z) output behavior.

The purpose of adding Altitude Cmd is to limit the boundaries of PID (z) control between the 0.8*(Ref_Spd_Max) as an upper limit and (Ref_Spd_Min) which represent the maximum reference speed and minimum reference speed, which means preventing the drone from reaching undesired speed. After that the (Saturation Dynamic) block also prevents the drone from exceeding the dynamic boundaries.

Finally the corrected values will be send out as a decision to Quad motors through terminals(Spd_A, Spd_B, Spd_C, Spd_D) to (Quadcotor 3D Model) block to reposition the drone to the correct level and frame.

We will discuss all the steps in the process to estimate the accurate values to our work.

After starting the simulation we can draw the trajectory of drone by drawing the XY graph as shown in figure (4.7) below:

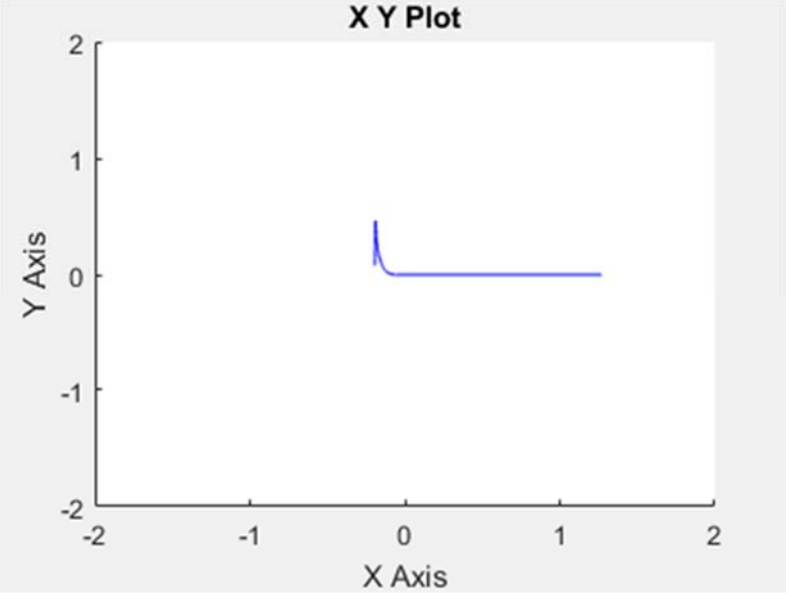


Figure 4.7 drone trajectory.

In the real time operation of quadcopter the commands operating it comes from remote control unit but in simulation in Simulink these commands can be applied from signal builder block in Simulink as shown figure(4.8) below:

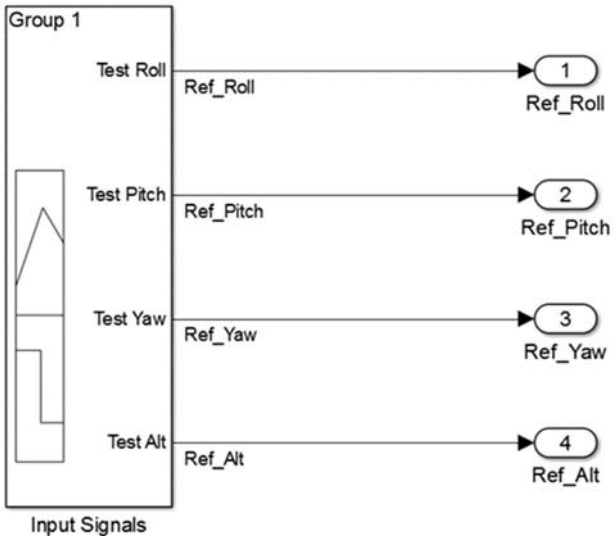


Figure 4.8 Signal builder as an input signals.

After starting simulation a 3D drone will be shown on mechanics explorer window as shown in figure (4.9) below:

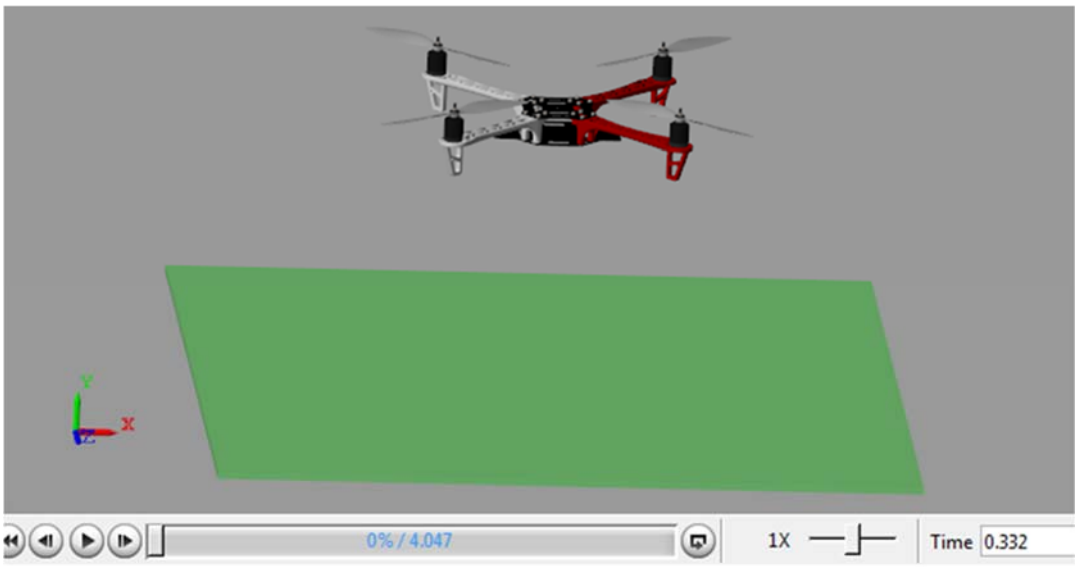


Figure 4.9 Mechanics explorer's window during simulation.

4.2 Drone mechanical parts

The mechanical parts forming the drone in Simulink is consist of the following drawing figure (4.10):

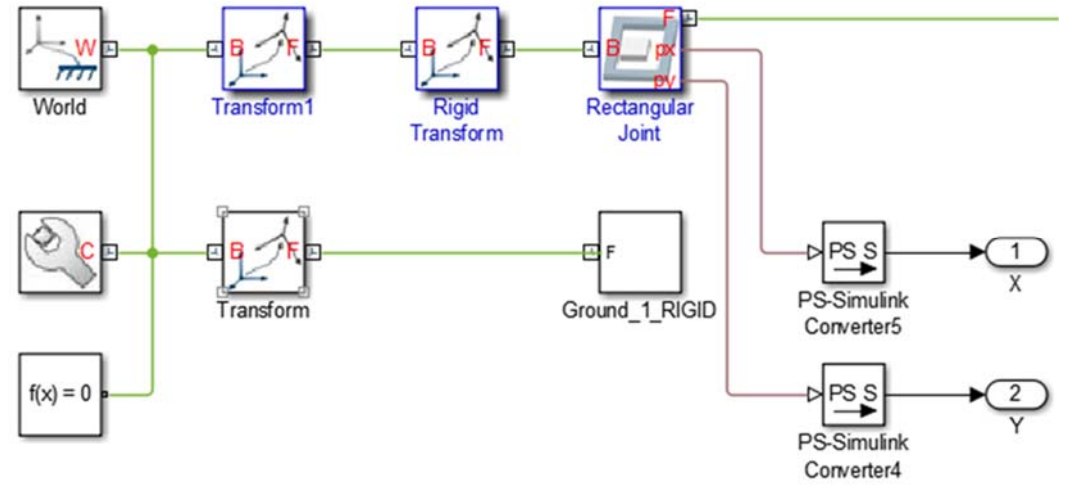


Figure 4.10 (a) Drone parts (first step).

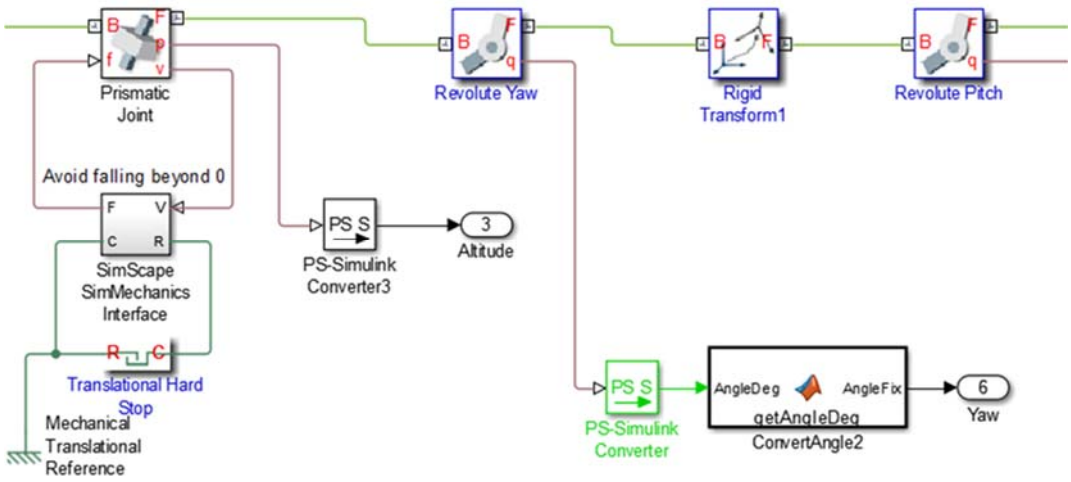


Figure 4.10 (b) Drone parts (second step).

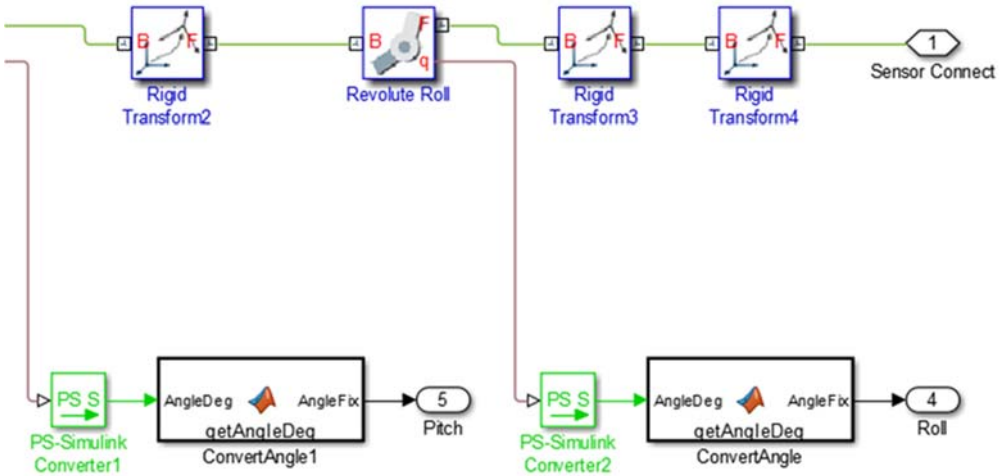
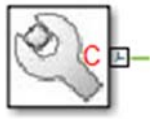


Figure 4.10 (c) Drone parts (third step).

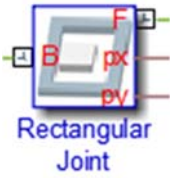
The first part consist of the following blocks:

World block: the world block represents the world frame which is motionless and represents ground.

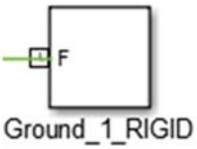


Mechanism configuration: it applies a gravity effect to the body and it can be added in any direction x, y or z direction.

The transform block represents connection between two frames.

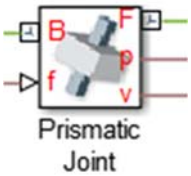


Represents a rectangular joint between two frames. This joint has two translational degrees of freedom represented by two prismatic primitives along a set of two mutually orthogonal axes.



The rigid body represents frame on ground (as a base).

This block converts the input Physical Signal to a unit less Simulink output signal.



This block represents a joint with one translational degree of freedom.

This block represents a joint with one rotational degree of freedom.

This block is a function block and if the angle value is more than 360 degree it will make it in range of 360.

4.3 PID different tuning values and its effect on flight

1. For the following values of PID (z) controller we will get the output signal as in figure (4.11).

Proportional (P): 4.5

Integral (I): 10

Derivative (D): 1

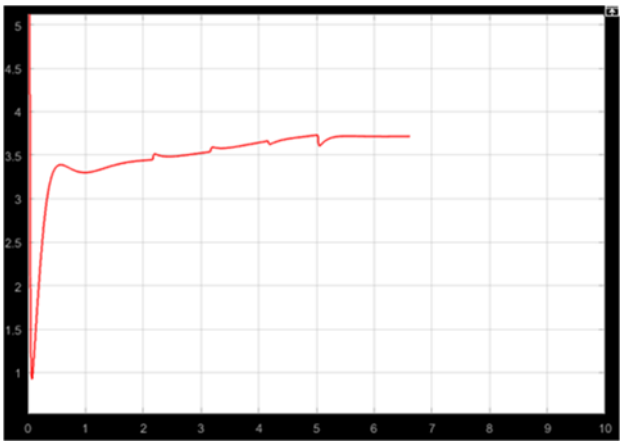


Figure 4.11 Output from PID (z) controller.

2. For the following values of PID (z) controller we will get the output signal as in figure (4.12).

Proportional (P): 3.5

Integral (I): 10

Derivative (D): 1

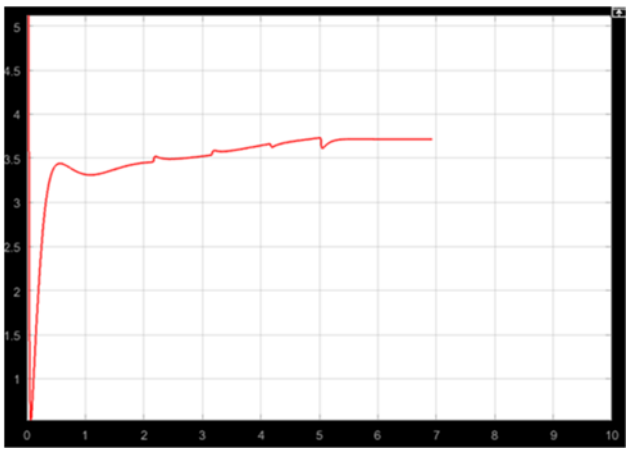


Figure 4.12 Output from PID (z) controller.

3. For the following values of PID (z) controller we will get the output signal as in figure (4.13).

Proportional (P): 2.5

Integral (I): 10

Derivative (D): 1

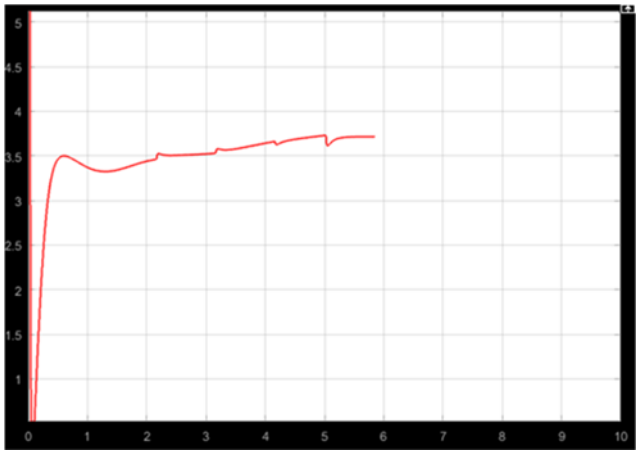


Figure 4.13 Output from PID (z) controller.

4. For the following values of PID (z) controller we will get the output signal as in figure (4.14).

Proportional (P): 0.5

Integral (I): 10

Derivative (D): 1

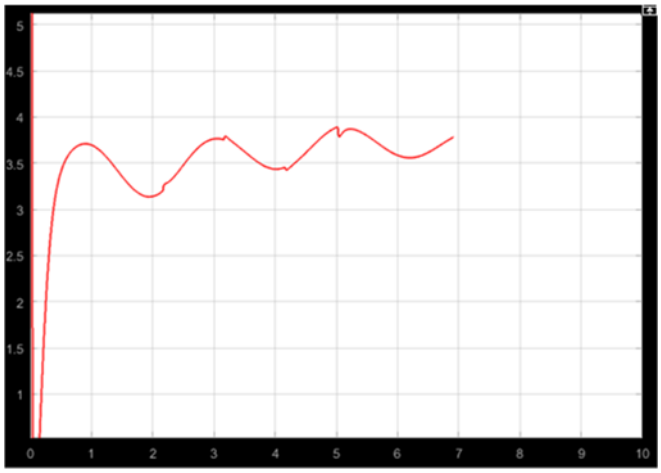


Figure 4.14 Output from PID (z) controller.

5. For the following values of PID (z) controller we will get the output signal as in figure (4.15).

Proportional (P): 4.5

Integral (I): 8

Derivative (D): 1

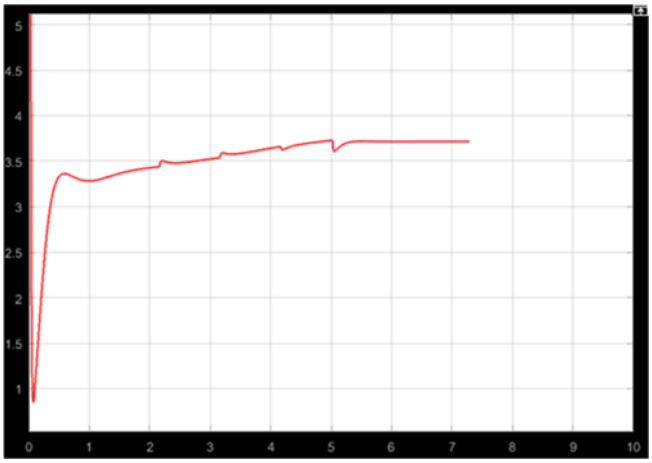


Figure 4.15 Output from PID (z) controller.

6. For the following values of PID (z) controller we will get the output signal as in figure (4.16).

Proportional (P): 4.5

Integral (I): 6

Derivative (D): 1

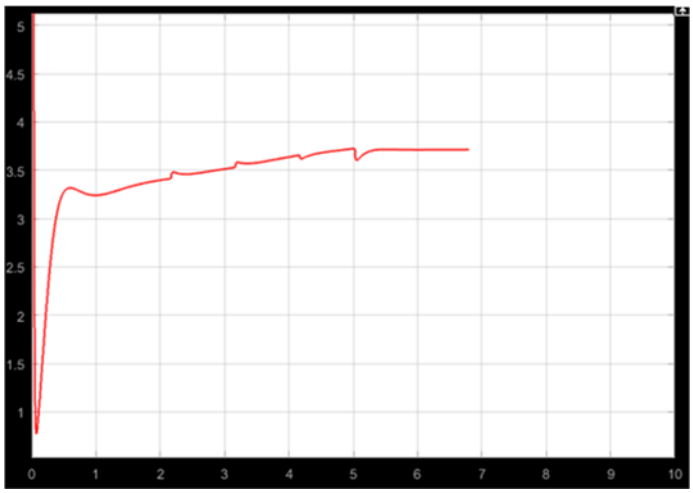


Figure 4.16 Output from PID (z) controller.

7. For the following values of PID (z) controller we will get the output signal as in figure (4.17).

Proportional (P): 4.5

Integral (I): 2

Derivative (D): 1

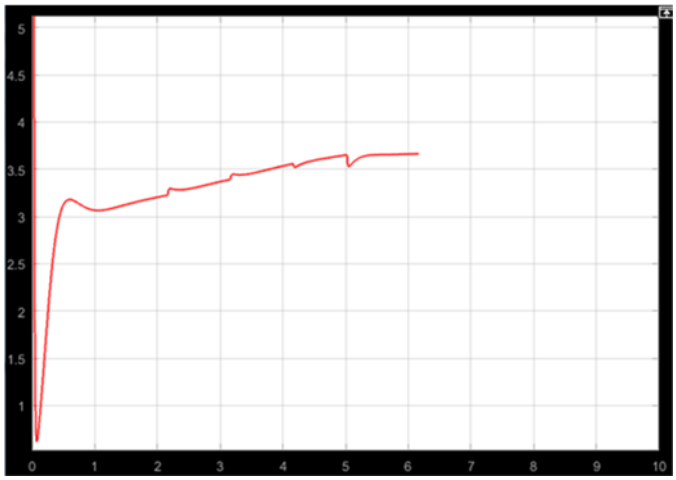


Figure 4.17 Output from PID (z) controller.

8. For the following values of PID (z) controller we will get the output signal as in figure (4.18).

Proportional (P): 4.5

Integral (I): 2

Derivative (D): 0.5

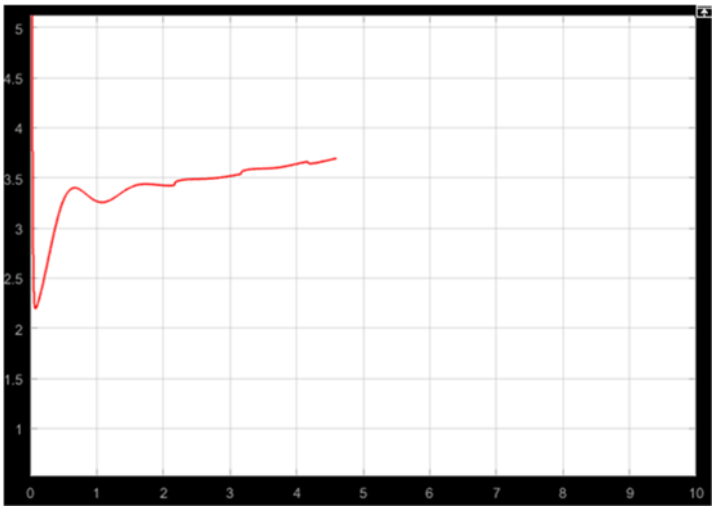


Figure 4.18 Output from PID (z) controller.

9. For the following values of PID (z) controller we will get the output signal as in figure (4.19).

Proportional (P): 4.5

Integral (I): 2

Derivative (D): 0.1

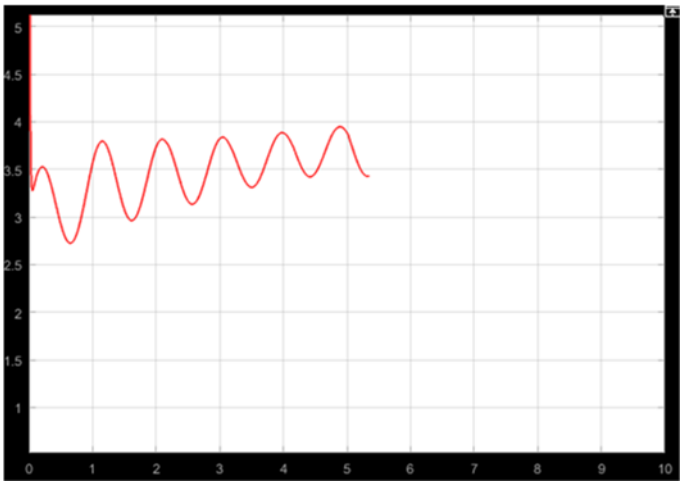


Figure 4.19 Output from PID (z) controller.

4.4 Conclusion of Simulation

As we can see from the previous drawings and values that There are infinite probability for changing PID gain values so the tuning of the right values need to be tried to get a small transient period and less oscillation but there will be limitations because the drone will response slowly to the process of control.

CHAPTER 5

RESULT AND DISCUSSION

The new generation of Wireless Sensor Networs, that is known as the the Internet of Things (IoT) enables the direct connection of physical objects to the Internet using microcontrollers.

The Internet of Things (IoT) is a technology that allows objects to be connected to hte internet, enabling them with communication capabilities (with other objects and with people).

In our study we developed an UAV system that is controlled using internet connection which gives capability for infinite distance control, the UAV in our system can detect certain objects using their colors.

after construction of the system and after making test for flight control and object detection we discovered that the process of flight control based on internet connection demands a high bit rate for communication to ensure the stability of the UAV system and online video transmission and object detection , also such a system needs an autonomous flight algorithm in case of the connection lost or jamming procedure from enemy in case of military applications.

Also the process of using raspberry pi 3 for such an action is somehow not reasonable because of the limited capabilities for such an embedded systems and the heavy task

needed in this operation.

In the futer work it is recommended to use Fuzzy Logic in after recognition step in order to control the movment of object toward target.

A detail study should be made to select the member ship functions for fuzzy logic, the selection also should cover the limit values for each member ship function and that would be made by experuments.

The movment should control the values of Yaw, Pitch and Roll to reach the target in a smooth motion and not to lose its trace during the tracking process.

The Defuzzification Methods should be made according to selected method from fuzzy known methods such as (Center of are, Center of sums, Center of maximum,...etc.).