IITB Summer Internship 2014

Project Report

Investigation of possible exits and windows in an indoor environment for Quadrotor navigation

Principal Investigator

Prof. D.B. Phatak

Project In-Charge

Mr. Rajesh Kushalkar

Project Mentors

Mr. SandeshPati
Miss Maitreyee Mordekar
Mr. SivakiranKodali
Mr. Varun Madkaikar

Project Team Members

Mr. AadityaHambarde
Mr. Ankit Kumar
Mr. D Yogendra Rao
Mr. ShubhamGarg
Miss. Shuchi Sharma

July 2, 2014

Summer Internship 2014 Project Approval Certificate

Department of Computer Science and Engineering Indian Institute of Technology Bombay

The project entitled **Quadcopter Navigation using Aakash Tablet with Onboard Image Processing** submitted by **Mr. Aaditya Hambarde**, **Mr. Ankit Kumar, Mr. D Yogendra Rao**, **Mr. Shubham Garg** and **Miss. Shuchi Sharma** is approved for Summer Internship 2014 programme from 10th May 2014 to 6th July 2014, at Department of Computer Science and Engineering, IIT Bombay.

Prof. Deepak B. Phatak	Mr. Rajesh Kushalkar
Dept of CSE, IITB	Dept of CSE, IITB
Principal Investigator	Project In-charge

Place: IIT Bombay, Mumbai

Date: July 2, 2014

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. AadityaHambarde
VNIT Nagpur

Mr. Ankit Kumar
NIT Rourkela

Mr. D Yogendra Rao
NIT Rourkela

Mr. ShubhamGarg
SVNIT Surat

Date

Miss. Shuchi Sharma
BIT Jaipur

Abstract

In the last few decades,Unmanned Aerial Vehicles (UAVs) have become more commonly used for many applications. The need for aircraft with greater maneuverability and hovering ability has led to current rise in quadcopter research. The four-rotor design allows quadcopters to be relatively simple in design yet highly reliable and maneuverable.

The purpose of our project is to create a Aakash controlled quadcopter with onboard image processing.Our quadcopter can be used for indoor and outdoor navigation.In outdoor navigation we can traverse from one way-point to another way-point using an on board GPS.But,in indoor navigation,we can't reach to the GPS satellites.So, we are detecting doors and obstacle using Raspberry pi with an onboard raspberry pi camera.We have even provided an USB Camera with it for live video streaming on ourGround Control Station (GCS).

The scheduler program arranges the following tasks: Controller input,sensor data received from the accelerometer, Gyroscope, and Magnetometer.The accelerometer/gyroscope and magnetometer both uses I2C protocol to communicate with the main controller(Atmega2560).We have implemented a 9-axis IMU for the sensor fusion using a Complementary filter.

List of Figures

List of Tables

Contents

Chapter 1

Introduction to Quadcopter

1.1Introduction

1.1.1 What is a quadcopter?

Humans have always dreamt of flying. Excluding rockets and balloons there exist two types of flying machines. They are:

i. Fixed wing (e.g. Airplane)
ii. Rotating wing (e.g. Multi-copters)

A quadcopter is a helicopter with 4 rotors. Hence it is also called a quad-rotor. They are Vertical Take Off and Landing (VTOL) aircraft similar to helicopters.

1.1.2 What is the difference between a regular helicopter and a quadcopter?

One may ask what is the difference between a helicopter and a quadcopter. A regular helicopter has one big rotor to provide all the lifting power and a little tail rotor to offset the aerodynamic torque generated by the big rotor (without it, the helicopter would spin almost as fast as the propeller). Unlike a helicopter, a quad-rotor has four rotors all work together to produce upward thrust and each rotor lifts only 1/4 of the weight, so we can use less powerful and therefore cheaper motors. Advantage of quadcopter over a regular helicopter is its mechanical simplicity. They don't require complex mechanical parts to control the flight. It is electronically controlled. Quadcopters are less efficient than helicopters. Compared to a helicopter, a quadcopter has much less blade area and hence generates lesser lift per blade.

1.1.3 Drag, Lift and Thrust

Lift is a force generated by individual blades and acts in a direction perpendicular to the direction of relative wind. Drag acts in a direction parallel to that of relative wind. The resultant of Lift and Drag (Lift +Drag) is known as *total reaction*. Break this force into horizontal and vertical directions. The vertical component is known as Total Rotor Thrust (TRT). In case of quadcopter hover, it is the TRT which opposes weight.Lift does not oppose weight. However, through a few manipulations, the lift provides a vertical force that opposes the weight. Weight is the easiest to understand--if the helicopter weighs *x* pounds, you need *x*pounds of force to overcome weight. Lift isn't necessarily in-line with weight, so you have to convert it to a force that is.

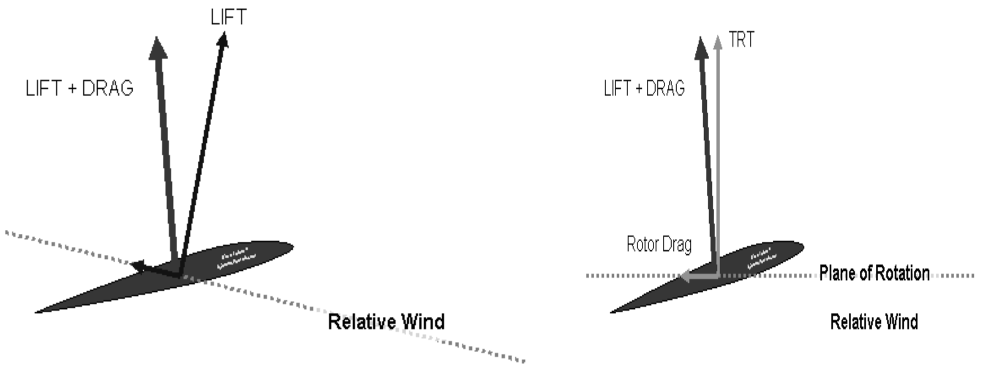


Figure 1.1: Difference between Thrust and Lift
Courtesy : Retrieved June 30,2014 from Rotorcraft Professional, http://helicopterforum.verticalreference.com/

In fluid dynamics, drag (sometimes called the air resistance) refers to forces acting opposite to the relative motion of any object moving with respect to surrounding fluid. In case Unmanned Air Vehicles (UAV), the surrounding fluid is air. Drag forces are dependent on velocity. We don't have to delve deep in to this topic. Basically what we have to understand is that drag is some form of air resistance.

1.2 Operation on RC Airplane

Now let us first understand how a normal RC-Airplane works. RC obviously means Radio Controlled. An important set of words is: Yaw, Roll and Pitch. Rotation around X axis is known as Pitch. Rotation around Z axis is Yaw and that around Y axis is Roll. Figure 4.2 explains this. Another set of important words are: Rudder, Aileron, Elevator. They are used to control Yaw, Roll and Pitch of the plane.

Rudder controls Yaw

Ailerons control Roll

Elevator controls Pitch

If you want to see how it works, you can refer to the figures given in this document, or http://howthingsfly.si.edu/flight-dynamics/roll-pitch-and-yaw would be an excellent stop too. This link gives excellent animations for Y-R-P control in a traditional RC-Airplane.

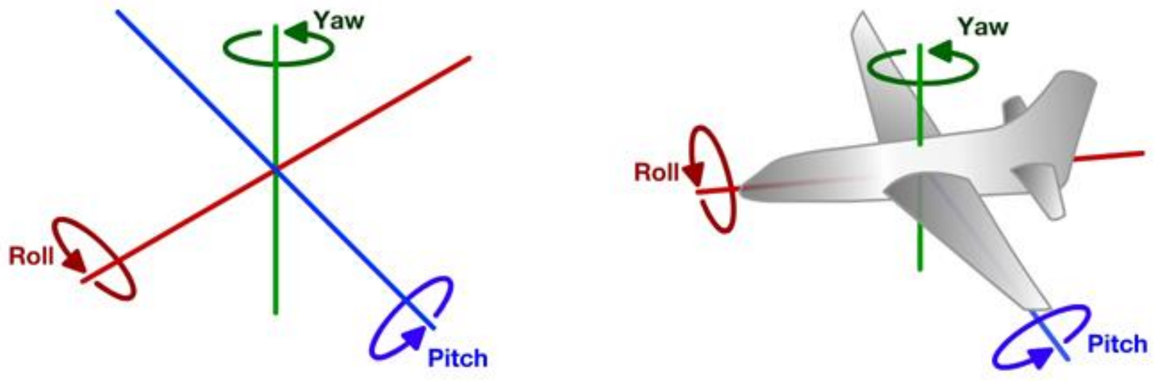


Figure 1.2: Yaw, Roll and Pitch

Courtesy: Retrieved June 30, 2014 from Smithsonian National Air and Space Museum, https://howthingsfly.si.edu/flight-dynamics

All the above said things involve complex aerodynamics and we have not delved deep in to it heresince our endeavor does not need an in depth knowledge of it. Another word that is not mentioned here is Throttle. Throttle controls the speed of the air-plane. Throttle is associated with the engine of the air-plane. It can be observed clearly in the figure below.

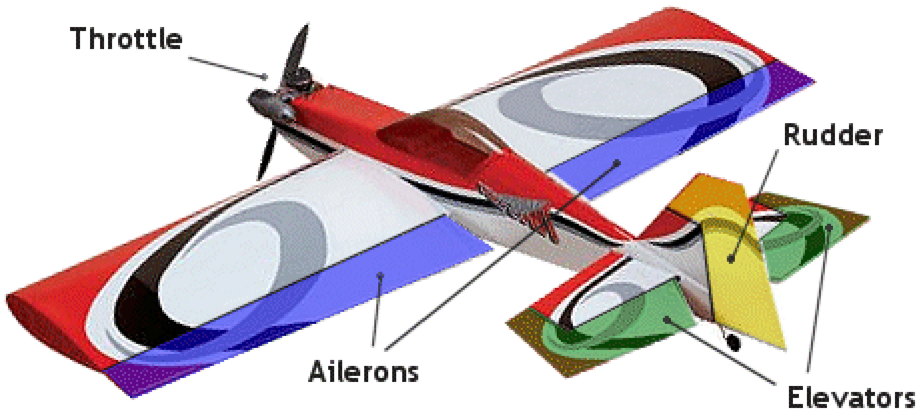


Figure 1.3: Structure of an RC- Airplane

Courtesy: Retrieved June 30, 2014 from RC Airplane World, http://www.rc-airplane-world.com/rc-airplane-controls.html

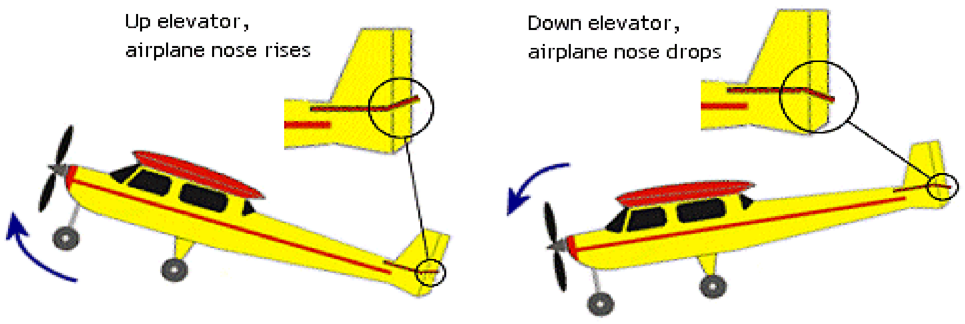


Figure 1.4: Elevator controls pitch

Courtesy: Retrieved from http://www.rc-airplane-world.com/

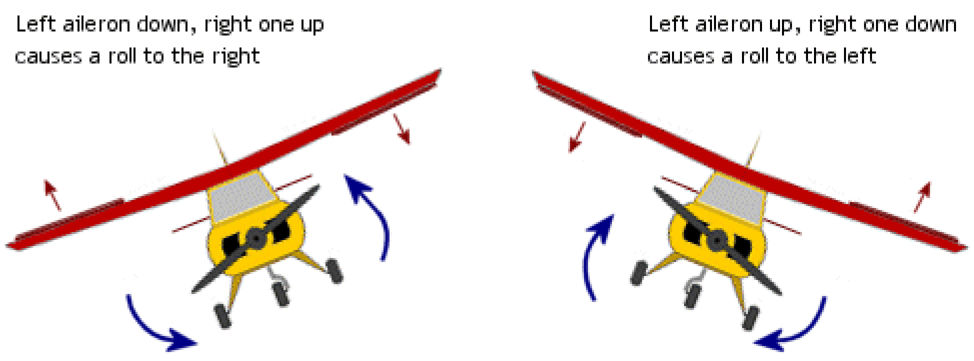

Left aileron down, right one up causes a roll to the right
Left aileron up, right one down causes a roll to the left

Figure 1.5: Aileron controls roll

Courtesy: Retrieved from http://www.rc-airplane-world.com/

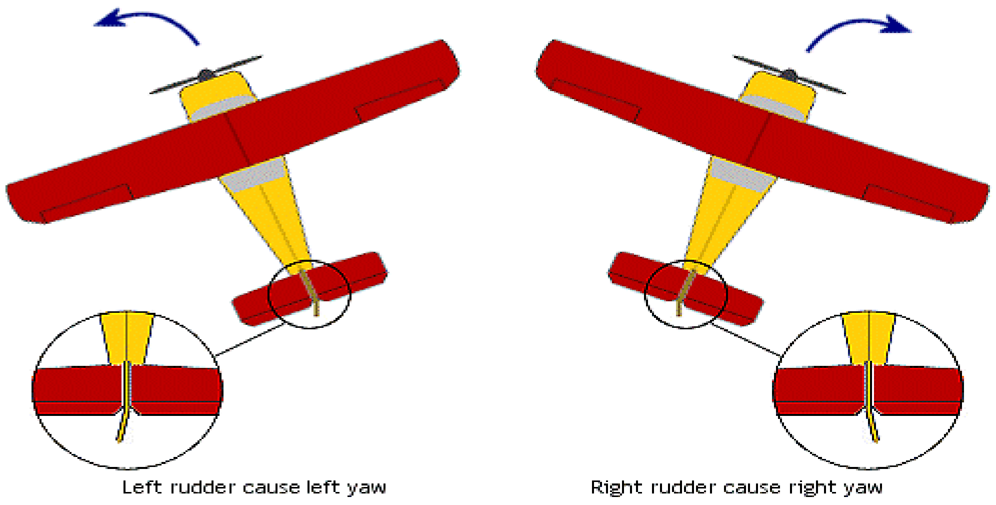


Figure 1.6: Rudder controls Yaw

Courtesy: Retrieved from http://www.rc-airplane-world.com/

1.3 Operation of Quadcopter

1.3.1 Introduction

Quadcopter consists of 4 rotors aligned in a square. There are 2 ways to look at this. Either they are aligned in a **'+'** configuration or **'X'** configuration. The figure below shows the **'+'** configuration.Of the 4 motors, we want 2 of them to spin in a clockwise manner and 2 in counter-clockwise manner. The same can be seen in the Figure 1.7.

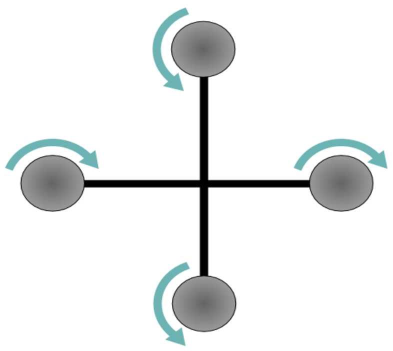


Figure 1.7: Motors in '+' configuration

The four motors should not rotate in the same direction because if they did, the quadcopter would spin about an axis and we don't want that. The same thing will happen if the helicopter does not have a tail rotor.The weight of the quadcopter is divided between 4 rotors.Because each rotor generates a portion of the total lift, the flight of a quadcopter is controlled by varying the relative lift and torque of each rotor.

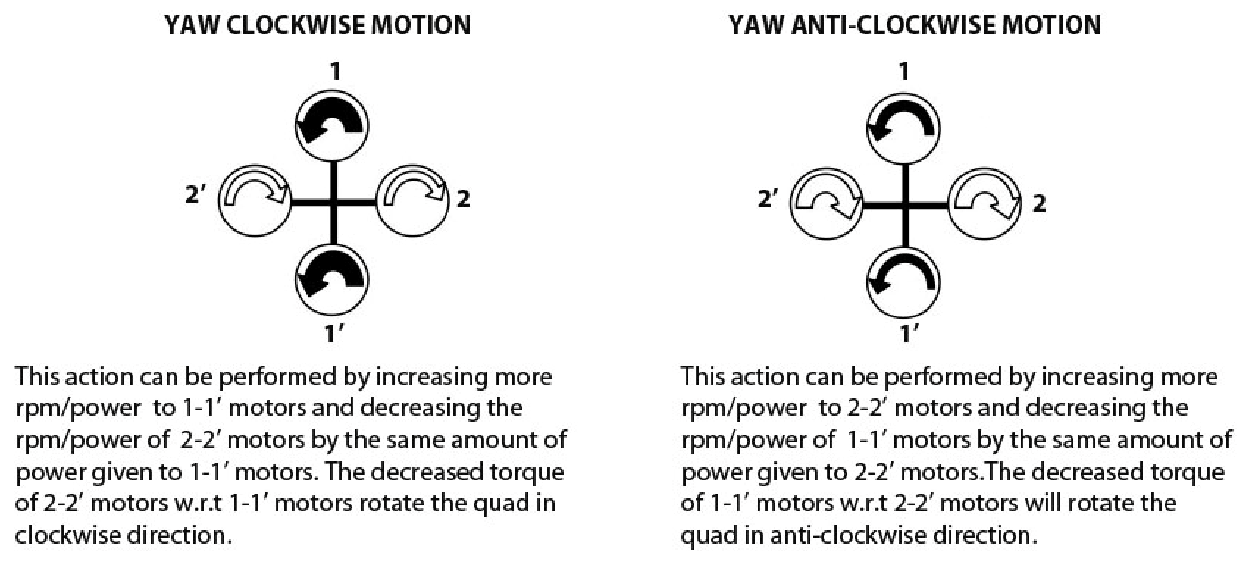
1.3.2 Quadcopter Hover

When a quadcopter is hovering in the mid-air each rotor is generating the same amount of lift and torques of the 2 sets of motor cancel out. Since the net torque is zero, the quadcopter does not spin. This is simple physics and does not warrant an explanation. For a quadcopter to hover, the vertical upward forces generated by four motors must be equal to the vertical downward weight of the quadcopter.

In reality however, due to many variables affecting a quadcopter's flight, hovering is not a fixed setting, but rather the result of continuously adjusting several flight parameters based on sensor input. A very sophisticated electronic control system is needed for this. A quadcopter has 4 degrees of freedom. They are Yaw, Roll, Pitch and Altitude. Motion along each degree of freedom can be controlled by adjusting thrusts of each motor.

1.3.3 Yaw Control

Yaw is induced by unbalanced aerodynamic torques. The question is how to make our quadcopter yaw. The figure below explains it all.



This action can be performed by increasing more rpm/power to 1-1' motors and decreasing the rpm/power of 2-2' motors by the same amount of power given to 1-1' motors. The decreased torque of 2-2' motors w.r.t 1-1' motors rotate the quad in clockwise direction.

This action can be performed by increasing more rpm/power to 2-2' motors and decreasing the rpm/power of 1-1' motors by the same amount of power given to 2-2' motors.The decreased torque of 1-1' motors w.r.t 2-2' motors will rotate the quad in anti-clockwise direction.

Figure 1.8: Yaw Control

Courtesy: Retrieved June 15, 2014 from http://www.robowiz.co.in/tutorial/quadrotor.aspx

From this, we come to understand that an upward resultant torque will rotate the quadcopter frame in a clock wise direction. Similarly, a downward resultant torque would rotate the body in counter-clockwise direction. If the power of motors 1-1' is increased and the power of motors 2-2’ decreased by the same amount, the quadcopter does not rise/fall, it just rotates.

1.3.4 Pitch Control

This action can be controlled by a pair of opposite motors in which speed of one motor is increased while the speed of other opposite motor is decreased by the same amount and the other opposite pair has the same Revolutions per minute (RPM) which in turn lift the quad from higher speeded motor side and tilt down the quad from lower speeded motor side.

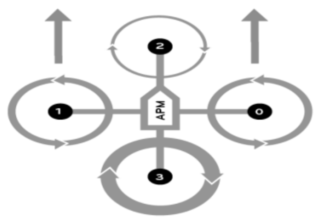
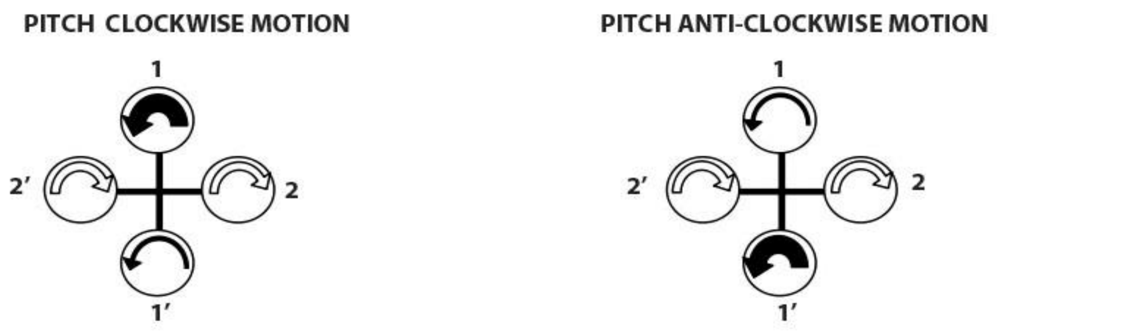


Figure 1.9: Quadcopter Forward Motion



This action can be attained by increasing the speed of motor 1 with respect to motor 1′ and keeping the speed of motor 2-2′ constant.This action will give more thrust from motor 1 side w.r.t motor 1′ and hence moves up the quad from motor 1 side (Clockwise motion)

This motion can be attained by increasing the speed of motor 1′ with respect to motor 1 and keeping the speed of motor 2-2′ constant.This action will give more thrust from motor 1′side w.r.t motor 1 and moves up the quad from motor 1′side(anti clockwise motion)

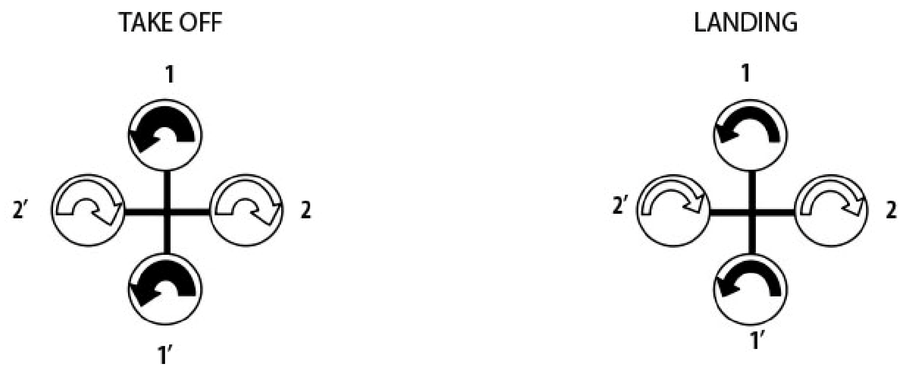
Figure 1.10: Pitch Control

Courtesy: Retrieved June 15, 2014 from http://www.robowiz.co.in/tutorial/quadrotor.aspx

‘If you are hovering at a specific height and want to go forward, you pitch forward.’ The same principle is explained by the Figure 1.9. Hence if you want to move forward, decrease the speed of the front motor and increase the speed of back motor by an equivalent amount.

1.3.4 Altitude Control

When the speed of all motors is high, the overall thrust will be more and hence the quadcopter would move in upward direction. When the speed of the motors is low, the overall thrust decreases which in turn helps in safe landing of the quadcopter.



1-1′ One pair of motors with anti-clockwise motion and 2-2′ other opposite pair of motor with clockwise motion

TAKEOFF - Speed of all the motors should be high and same
LANDING- Speed of all the motors should be low and same

Figure 1.11: Altitude Control

Courtesy: Retrieved June 15, 2014 from http://www.robowiz.co.in/tutorial/quadrotor.asp

1.3.5 Roll Control

Roll is like pitch, but in left-right direction. This may seem confusing but roll and pitch are determined from where the front of the quad is, and in a quadrotor they are basically interchangeable; but do take note that you have to decide which way is front and be consistent or your control may go out of control.Roll is used for controlling the sideways motion of the quadcopter.

1.3.6 Motion-A combination of Yaw, Roll, Pitch, Throttle

Roll and Pitch tilts the quadcopter.When the quadcopter tilts, the force vector splits in to 2 components.

1. Horizontal Component: The quadcopter begins to move in a direction opposite to this unbalanced force component and gives the quadcopter a straight left/right motion or straight forward/backward motion.
2. Vertical Component: Since the force vector has been split, the vertical component will be smaller. Hence to maintain the altitude, throttle has to be increased.

Hence, any motion of quadcopter is a beautiful combination of Yaw, Roll, Pitch and Throttle.

Chapter 2

Hardware Overview

2.1Sensors on Quadcopter

2.1.1 MPU6050 (3-axis accelerometer and 3-axis gyroscope)

The MPU-6050 is the world's first integrated 6-axis motiontracking device that combines a 3-axis gyroscope, 3-axis accelerometer. With its dedicated I2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis Motion Fusionoutput. The MPU-60X0 motiontracking device, with its 6-axis integration, on-board Motion Fusion.

The MPU-6050 features three 16-bit Analog-to-Digital Converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g.

An on-chip 1024 byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-6050 uniquely enables low-power motion interface applications in portable applications with reduced processing requirements for the system processor. By providing an integrated Motion Fusionoutput, the Digital Motion Processor (DMP) in the MPU-6050 offloads the intensive motion processing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output. Communication with all registers of the device is performed using either I2C at 400kHz or Serial Programming Interface(SPI) at 1MHz (MPU-6000 only). For applications requiring faster communications, the sensor and interrupt registers may be read usingSPI at 20MHz (MPU-6000 only). Additional features include an embedded temperature sensor and an on-chip oscillator with ±1% variation over the operating temperature range.

2.1.2 HMC5883 (3-axis digital compass)

The Honeywell HMC5883L magneto-resistive sensor circuit is a trio of sensors and application specific support circuits to measure magnetic fields. With power supply applied, the sensor converts any incident magnetic field in the sensitive axis directions to a differential voltage output. The magneto-resistive sensors are made of a nickel-iron (permalloy) thin-film and patterned as a resistive strip element. In the presence of a magnetic field, a change in the bridge resistive elements causes a corresponding change in voltage across the bridge outputs.

These resistive elements are aligned together to have a common sensitive axis that will provide positive voltage change with magnetic fields increasing in the sensitive direction. Because the output is only proportional to the magnetic field component along its axis, additional sensor bridges are placed at orthogonal directions to permit accurate measurement of magnetic field in any orientation.

2.1.3 Self Test

To check the HMC5883L for proper operation, a self test feature in incorporated in which the sensor is internally excited with a nominal magnetic field (in either positive or negative bias configuration). This field is then measured and reported. This function is enabled and the polarity is set by bits MS[n] in the configuration register A. An internal current source generates current of 10mA DC current from V_{dd} supply.This DC current is applied to offset straps of magneto-resistive sensor, which creates an artificial magnetic field bias on the sensor. The difference of this measurement and the measurement of the ambient field will be put in the data output register for each of the three axes. By using this built-in function, the manufacturer can quickly verify the sensor's full functionality after the assembly without additional test setup. The self test results can also be used to estimate or compensate the sensor's sensitivity drift due to temperature.

For each *selftest measurement*, the ASIC:

i. Sends a *Set* pulse.
ii. Takes one measurement (M1).
iii. Sends the (~10 mA) offset current to generate the (~1.1 Gauss) offset field and takes another measurement (M2).
iv. Puts the difference of the two measurements in sensor's data output register

Output will be [M2 – M1]

2.1.3 MS5611

The MS5611-01BA is a new generation of high resolution altimeter sensors from MEAS Switzerland with SPI and I2C bus interface.The sensor module includes a high linearity pressure sensor and an ultra-low power 24 bit ΔΣ ADC with internal factory calibrated coefficients. It provides a precise digital 24 bit pressure and temperature value and different operation modes that allow the user to optimize for conversion speed and current consumption. A high resolution temperature output allows the implementation of an altimeter/thermometer function without any additional sensor. The MS5611-01BA can be interfaced to virtually any microcontroller. The communication protocol is simple, without the need of programming internal registers in the device. Small dimensions of only 5.0mm x 3.0mm and a height of 1.7mm allow for integration in mobile devices. The sensing principle employed leads to very low hysteresis and high stability of both pressure and temperature signal.

2.1.4 GPS Receiver

Receiver MT3329 module is based on MT3329 chipset from MediaTek Inc.It has 66channels for GPS tracking. GPS receiver gives data output in standard NMEA format with update rate of 1 second at 9600 bps. GPS module has LED indication to indicate GPS lock and onboard battery for location information backup, which allows GPS module to quickly acquire GPS information on start up.It supports various protocols such as GGA, GSA, GSV, RMC and VTG.

A sample NMEA output sentence is

```
$GPGGA,121739.000,4124.8963,N,08151.6838,W,1,5,1.5,280.2,M,34.0,M,*75
```

GPGGA stands for Global Positioning System Fix Data.

Table2.1: Example of NMEA protocol
Courtesy: NMEA protocol data sheet

Name	Example Data	Description
Sentence Identifier	$GPGGA	Global Positioning System Fix Data
Time	121739	12:17:39 Z
Latitude	1907.5230, N	19d 07.5230' N or 19d 07' 52" N
Longitude	07255.1137, E	72d 55.6838' W or 72d 55' 11" E
Fix Quality: 0 = Invalid 1 = GPS fix 2 = DGPS fix	1	Data is from a GPS fix
Number of Satellites	9	9 Satellites are in view
Horizontal Dilution of Precision (HDOP)	0.9	Relative accuracy of horizontal position
Altitude	69.5, M	69.5 meters above mean sea level
Height of geoid above WGS84 ellipsoid	-62.9, M	62.9 meters
Time since last DGPS update	Blank	No last update
DGPS reference station id	Blank	No station id

Checksum	*4F	Used by program to check for transmission errors

2.2Electronic Speed Controllers

Electronic Speed Controllersor ESCs provide a pulsating signal to the motors to allow variation in speed resulting quad movement. A good ESC is paramount to stable flying. The ESC must be the proper current rating as one too small will burn up and result in engine failure. We have to ensure that the ESCs can support the draw of your motors. Many suggest mounting the ESCs at the end of the quad's arms underneath the props to allow for prop wash cooling. We have used the Turnigy 30A ESCs as they will provide sufficient flow of power to the Turnigy BLDCs.

ESCs use powerful and high-performance MCU processor. The users can set function of use according to theirrequirements.The starting mode can be set. The response speed of throttle is very quickly and with very stable linear ofspeed regulation.

Multi protection functionis the protection for abnormal input voltage/ protection for low battery/ protection foroverheat/protection of lowing power when lost of signals from throttle that is provided in the ESCs. In order to avoid short circuit and leakage, the joint are connected by thermal shrinkable pipe to ensure insulation as shown in Figure 2.1

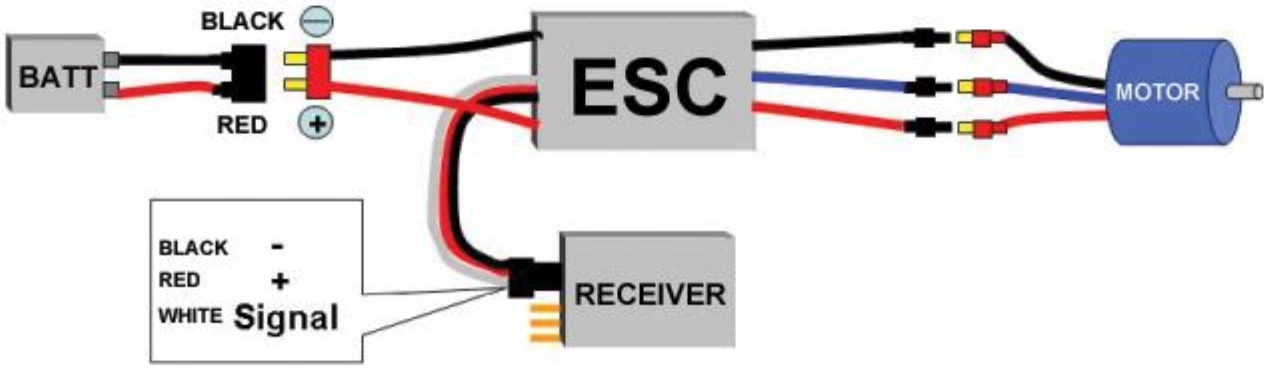


Figure 2.1: ESC Connections
Courtesy: Turnigy ESC 30Amp data sheet

The ESCs run with DC voltage and require PWM signal controllers for our electrical motors. It is important to utilize only the PWM signal controllers when connecting the motors to the Crius board. This is necessary since the Pulse-Width Modulation(PWM) is a better controlling technique when dealing with power-to-inertial electrical devices, such as in the case of our control units. The PWM uses a digitally-generated rectangular-pulse waveform to employ the switching between the supply and the load voltage. This ON and OFF (or HIGH and LOW) switching is measured as a frequency in hertz (Hz) per time (in msec), and varies on the duty cycle desired.

Since the power supplied will be greater depending how long the switch stays HIGH or ON, the corresponding duty cycle will enable more power into the motors. In order to incorporate the PWM technique, the Crius board uses its clock and counter to increment the frequency of the duty cycle periodically, and reset it at the end of every PWM period.

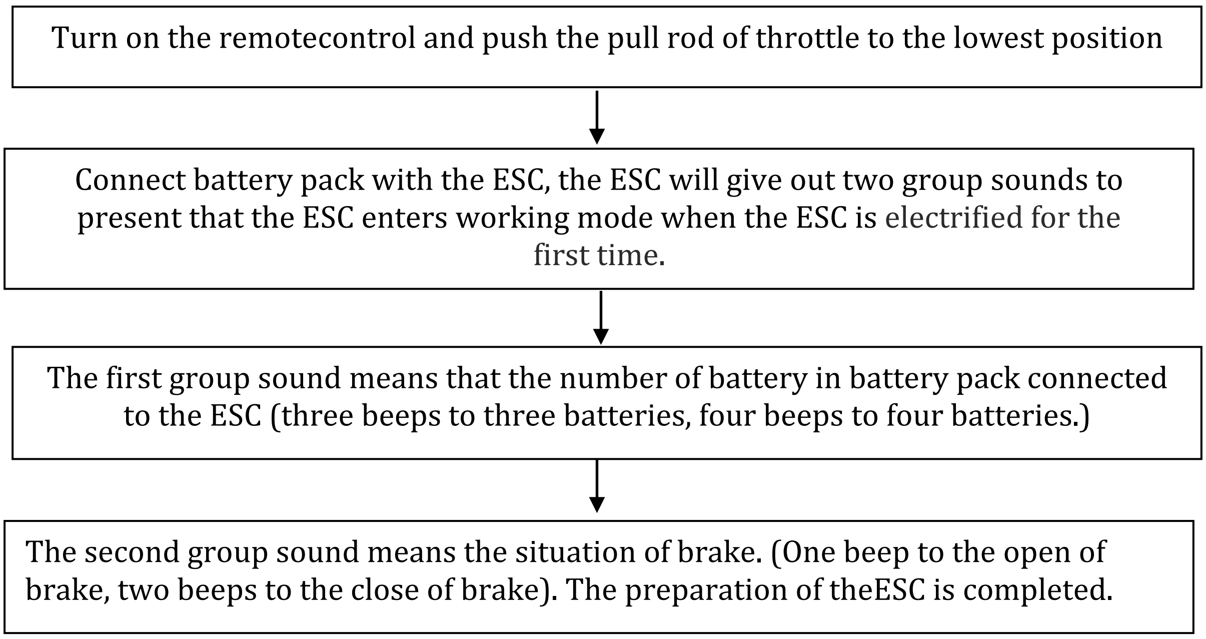

Turn on the remotecontrol and push the pull rod of throttle to the lowest position
Connect battery pack with the ESC, the ESC will give out two group sounds to present that the ESC enters working mode when the ESC is electrified for the first time.
The first group sound means that the number of battery in battery pack connected to the ESC (three beeps to three batteries, four beeps to four batteries.)
The second group sound means the situation of brake. (One beep to the open of brake, two beeps to the close of brake). The preparation of theESC is completed.

Figure 2.2: Configuring the ESCs

2.3 Brushless DC Motors

The brushless DC motor(BLDC motor) is actually a permanent-magnet synchronous machine .We have used Turnigy 1450kV BLDC motors.

Most motors on quadcopters are outrunners. In outrunner motors, the rotating part is on the outside, and not the inside (as opposed to the inrunner motors). Because of this layout this type of motors can generate much more torque. High torque is required for quadcopters, since you balance by changing the revolutions of the motor. The higher your torque the faster you can change the speed of your propellers. High torque also means you don't need a gearbox, and save a lot of mass. Outrunners are not very practical for brushed commutation, as it would require lots of wires and additional contacts. Hence most of the outrunners are brushless.

Some Features of BLDCs

i. Better speed versus torque characteristics
ii. High dynamic response
iii. High efficiency
iv. Long operating life

v. Noiseless operation
vi. Higher speed ranges

The correct motor/prop combination needs to be chosen by considering the size and weight of the quadcopter, the length of flying time and the power and agility that are desired. Also, the battery and ECSs must be chosen to effectively drive the motor/propcombination.

A rule of thumb is Required Thrust = (Weight x 2) / 4

Motor KV and thrust are not the only factors we need to consider when purchasing motors. Watts and efficiency are equally important.The KV rating is a measure for prop speed, it's determined by the thickness of the wire used in the motor windings. Thicker wire runs stronger, less efficient, faster, hotter, and has a higher KV rating. KV is RPMs per Volt. With a 11.1Volt battery, a 2200KV motor without a prop, will to run at 24,420 RPM (11.1 x 2200=24,420). The KV rating tells you what propeller you need with that motor.
Higher KV motors are for speed props and EDFs. They have slow acceleration and high speed. If you use a large prop, your motor and ESC will run very hot, and have a short life.Lower KV motors are for 3D planes and large props. They have fast acceleration and slow to medium speed. If you use a small prop, your plane will go slower and you will be under using the motors power.

2.4 Basic concepts of Propellers

A quadcopter uses two clockwise(CW) and two counter-clockwise(CCW) propellers. Propellers are classified by length and pitch. For example 9x4.7 propellers are 9 inch long and has a pitch of 4.7.

Generally, increased propeller pitch and length will draw more current. Also the pitch can be defined as the travel distance of one single prop rotation. In a nutshell, higher pitch means slower rotation, but will increase your vehicle speed which also use more power.

When deciding on length and pitch, you need to find a good balance. Generally a prop with low pitch numbers can generate more torque. The motors don’t need to work as hard so it pulls less current with this type of prop. If you want to do acrobatics, you will need torque propellers which provide more acceleration and it puts less pressure on the power system. Lower pitch propellers will also improve stability.

A higher pitch propeller moves greater amount of air, which could create turbulence and cause the aircraft to wobble during hovering. If you notice this with your quadcopter, try to choosing a lower pitched propeller.

When it comes to the length, propeller efficiency is closely related to the contact area of a prop with air, so a small increase in prop length will increase the propeller efficiency. (pretty much like swimmers with larger hands and feet can swim faster, but also more tiring for them)

A smaller prop is easier to stop or speed up while a larger prop takes longer to change speeds (inertia of movement). Smaller prop also means it draws less current, that is why hexacoptors and octocopters tend to use smaller props than quadcopter of similar size.

For larger quadcopters that carry payloads, large propellers and low-kv motors tend to work better. These have more rotational momentum, and will more easily maintain your aircraft's stability.

2.5 Raspberry Pi

The Raspberry Pi is a credit-card-sized single-board computer. We have used Raspberry Pi primarily for the following purposes.

i. USB webcam streaming over IP.
ii. Onboard Image Processing using Raspberry Pi camera.

We have used Raspberry Pi model B. We have used Raspbian OS in our implementation.

Specifications of the Raspberry Pi *single board computer* (SBC):

i. CPU of 700 MHz ARM1176JZF-S core with ARM 11 family, ARMv6 instruction set.
ii. Memory (SDRAM) consisting of 512 MB shared with GPU.
iii. Two USB 2.0 ports.
iv. Power ratings of 700mA (3.5 W).
v. Onboard storage of SD/MMC/ SDIO card slot 3.3V card power support only.
vi. Power source consisting of 5 Volt via MicroUSB or the GPIO header.
vii. Dimension 85.60 mm × 53.98 mm.
viii. Weighing 45 grams.
ix. Supports Arch Linux ARM, Debian Linux, Fedora, FreeBSD, NetBSD, Plan, Raspbian OS, RISC OS, Slackware Linux operating system.

We have used OpenCV for image processing and mjpeg streamer for live video streaming on Raspberry Pi that uses Raspbian operating system. The further details will be explained in Chapter 6.

2.4.1 Raspberry Pi Camera

The camera board attaches to the Raspberry Pi via a 15-way ribbon cable. There are only two connections to make; the ribbon cable need to be attached to the camera PCB and the Raspberry Pi itself. You need to get it the right way round, or the camera will not work. On the camera PCB, the blue backing on the cable should be facing away from the PCB, and on the Raspberry Pi it should be facing towards the Ethernet connection.

Although the connectors on the PCB and the Pi are different, they work in a similar way. On the Raspberry Pi, pull up the tabs on each end of the connector. It should slide up easily, and be able to pivot around slightly. Fully insert the ribbon cable into the slot, ensuring it is straight, then gently press down the tabs to clip it into place. The camera PCB itself also requires you to pull the tabs away from the board, gently insert the cable, then push the tabs back.

2.5 Crius Pro v1 Board

We have implemented 9-Axis sensor fusion for Attitude and Heading Reference System*(AHRS),* using Directional Cosine Matrix Algorithm(DCM). The calibration of the sensors is the key to the accuracy of the algorithm;therefore, the sensors' output must be calibrated before being input to the DCM algorithm. The algorithm is applied to the calibrated sensor readings to calculate the Euler angles describing the orientation of a body; consisting of the yaw, roll, and pitch angles.

2.5.1 Features of the Crius board

i. Up to 8-axis motor outputs.
ii. 8 input channels for standard receiver / support for PPM Sum / CPPM.
iii. 4 serial ports for debug/Bluetooth Module/OSD/GPS/telemetry.
iv. 2 servos output for pitch and roll gimbal system.
v. A servos output to trigger a camera button.
vi. 6 Analog output for extend device.
vii. A I2C port for extend sensors or devices.
viii. Separate 3.3V and 5V LDO voltage regulator.
ix. ATMega 2560 Microcontroller.
x. MPU6050 6 axis gyro/accelerometer with Motion Processing Unit.
xi. HMC5883L 3-axis digital magnetometer.
xii. MS5611-01BA01 high precision altimeter.
xiii. FT232RQ USB-UART chip and Micro USB receptacle.
xiv. On board logic level converter.

2.5.2 Flight modes for MegaPirateNG

The flight modes provided for the quadcopter are

i. Acro
ii. Alt Hold
iii. Simple
iv. Loiter (uses GPS)
v. Guided (uses GPS)
vi. Position (uses GPS)
vii. Circle (uses GPS)
viii. RTL (uses GPS)
ix. Auto (uses GPS)
x. Follow Me (uses GPS)

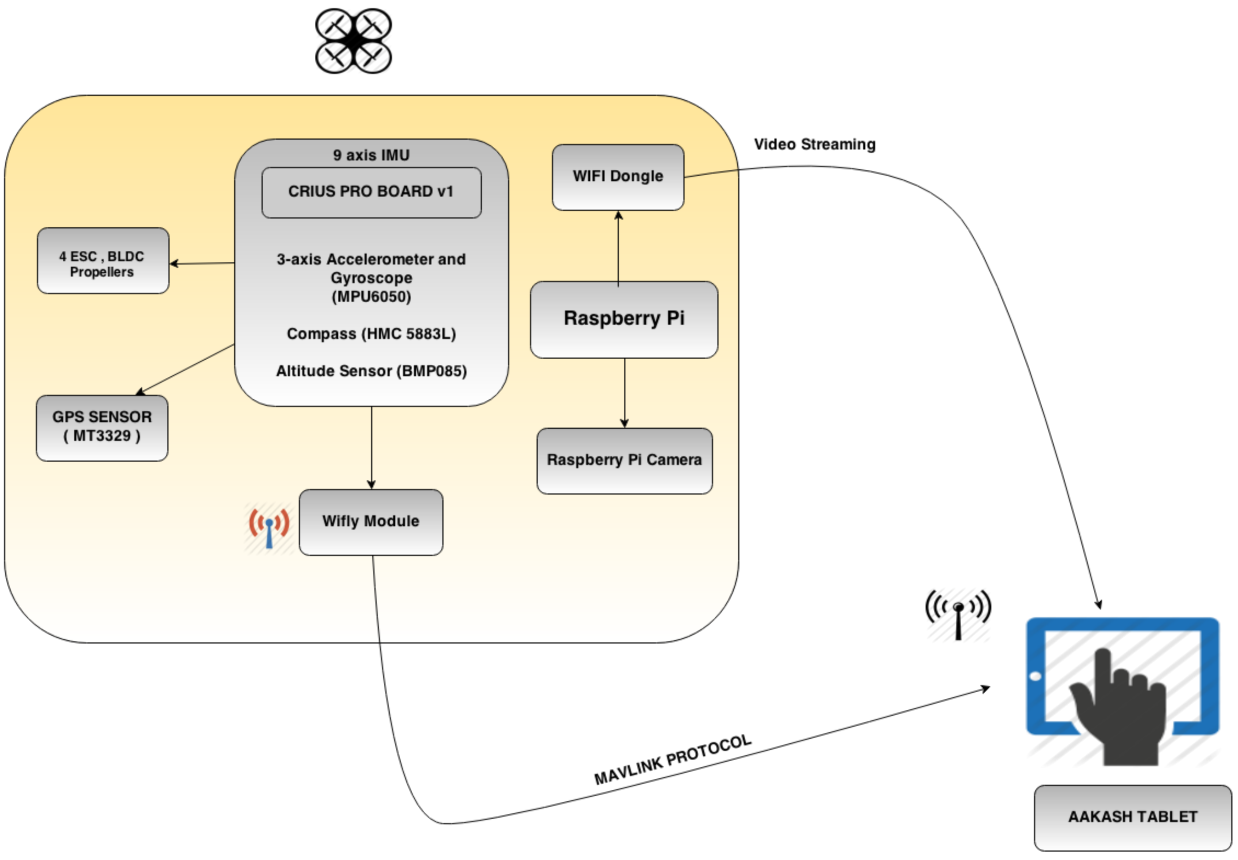


Figure 2.3: Block diagram of Quadcopter

2.6 Arducopter Code Overview

2.6.1 Arducopter.pde

1.`setup()`: It fills the memory with default parameters, flag values which important for loop termination.Sensor initialization is enabled in this `setup()` loop. So also ADC and GPS initialization takes place here. It calls `init_ardupilot()` that is defined in system.pde. This inturn calls the following important functions.

i. `load_parameters()`loads the parameters from EEPROM
ii. `init_rc_in()`sets up RC channels from radio
iii. `init_rc_out()`sets up the timer libraries

2. `Loop( ):` This function is called forever. Rate controllers are called at 100Hz from the main loop.

The `Fast_loop( )` is executed at 100 Hz (10 ms delay). The important functions called here are

i. `read_AHRS( )` has the Main IMU DCM Algorithm
ii. `read_radio()` is present incontrol_modes.pde.
If throttle_failsafe(), present in radio.pde, is enabled then update_GPS`()` is called. update_GPS`()` updates roll_sensor from sensor or GPS or updates pitch_sensor from sensor or GPS.

iii. `read_control_switch( )`, present in control_modes.pde, is used to change the mode of the quadcopter. The possible mode select options were ACRO, STABILIZE, LOITER, AUTO, ALT_HOLD, GUIDED, RTL
   - `read_IMU( )` reads the on board sensors value.
   - `calc_altitude_error( )`smoothens the altitude readings.
   - `update_current_flight_mode( )` changes the flight mode to one of the previously mentioned modes.
   - `stabilize( )`applies desired roll and pitch.

2.6.2 System.pde

1. `init_ardupilot()` does all the required initialization and processes everything we need for an in - air restart in whichserial ports 0 is configured as USB console, serial port 1 for GPS data and serial port 3 for xBee telemetry. The functions called inside `init_ardupilot()` are:

   i. `APM_RC.Init( )`initializes radio and read critical configurations.
   ii. `read_EEPROM_startup()` reads critical configuration information to start.
   iii. `APM_ADC.Init( )`initializes the APM ADC library.
   iv. `APM_MS5611.Init( )` initializes the APM Abs pressure sensor.
   v. `DataFlash.Init( )` initializes the DataFlash log.
   vi. `GPS.init( )` initializes the GPS.

2.`startup_ground( )` does all the calibrations that we need during a ground start

   i. `set_failsafe( )` sets the failsafe mode
   ii. `startup_IMU_ground( )` does all the calibrations that we need during a ground start
   iii. `update_GPS_light( )` sets GPS LED on if we have a fix or Blink GPS LED if we are receiving data
   iv. `resetPerfData( )`

2.6.3 APM_config.h

Defines you board features

```
#define MPNG_BOARD_TYPE   CRIUS_V1
#define GPS_PROTOCOL GPS_PROTOCOL_NMEA
#define FRAME_CONFIG QUAD_FRAME
#define FRAME_ORIENTATION X_FRAME

#define SERIAL_PPM SERIAL_PPM_DISABLED
//#define SERIAL_PPM SERIAL_PPM_ENABLED
//#define SERIAL_PPM SERIAL_PPM_ENABLED_PL1

#define LOGGING_ENABLED       DISABLED
```

2.6.4 Radio.pde

i. `init_rc_in()` sets rc channel ranges,dead zones and auxiliary ranges
ii. `APM_RC.InputCh(CH_ROLL)`reads roll input
iii. `APM_RC.InputCh(CH_PITCH)`reads pitch input
iv. `trim_control_surfaces()` stores control surface trim values
v. `read_radio_limits()` sets initial servo value limits for calibration routine
vi. `control_failsafe()` checks for failsafe condition based on loss of GCS control

2.6.5 Control_modes.h

- `Readswitch( )` : as knob is rotated ,the mode is correspondingly changed
- `Auto_trim( )` :meant to be called continuously while the pilot attempts to keep the copter level

2.6.6 Parameters.h

- Defines global (public) memory variables used to control the AP function.

2.6.7 Attitude.pde

i. `stabilize()` controls aileron/rudder, elevator, rudder (if 4 channel control) and throttle toproduce desired attitude and airspeed.
ii. `get_stabilize_roll()` calculates angle error,limit the error we're feeding to the PID,convert to desired rate and set targets for rate controller
iii. `get_acro_roll` set targets for rate controller
iv. `get_roll_rate_stabilized_ef()` calculates roll with rate input and stabilized in the earth frame
v. `update_rate_contoller_targets()`calculates `update_rate_contoller_targets` and converts earth frame rates to body frame rates for rate controllers
vi. `run_rate_controllers()` runs roll, pitch and yaw rate controllers and send output to motors targets for these controllers comes from stabilize controllers

2.7 Hardware in Loop Simulation

Hardware-in-the-loop (HIL) simulation is a technique that is used in the development and test of complex process systems. HIL simulation provides an effective platform by adding the complexity of the IMU under control to the test platform.

The HIL simulation includes a mathematical model of the process and a hardware device/ECU you want to test, e.g. a quadcopter PID controller. The hardware device is normally an embedded system.

The purpose of a Hardware-In-the-Loop system is to provide all of the electrical stimuli needed to fully exercise the ECU. In this way you fool the ECU into thinking that it is indeed connected to a IMU.

2.7.1 Steps for Hardware in Loop Simulation

i. Enable `#define HIL_MODE_ATTITUDE` in APM_Config.h

ii. Upload the ArduCopter 2.7.3 code on to the APM board using Arduino IDE, as soon as it completes compiling RESET the IMU board.To make sure it is uploading properly observe the Rx-Tx LEDs, they would be blinking super-fast. Once it is done, the *A* LED should turn green, which implies that the code is loaded onto the board.

Chapter 3

Wireless Communication with Quadcopter

3.1 Xbee and Zigbee

3.1.1 IEEE 802.15.4

802.15.4 is a standard for wireless communication issued by the *IEEE*. 802.15.4 was developed for lower data rate, simple connectivity and battery application. The 802.15.4 standard specifies that communication can occur in the 868-868.8 MHz, the 902-928 MHz or the 2.400-2.4835 GHz Industrial Scientific and Medical (ISM) bands. 2.4 GHz band is popular as it is open in most of the countries worldwide.

The 802.15.4 standard specifies that communication should occur in 5 MHz channels ranging from 2.405 to 2.480 GHz. In the 2.4 GHz band, a maximum over-the-air data rate of 250 kbps is specified, but due to the overhead of the protocol the actual theoretical maximum data rate is approximately half of that. While the standard specifies 5 MHz channels, only approximately 2 MHz of the channel is consumed with the occupied bandwidth. At 2.4 GHz, 802.15.4 specifies the use of Direct Sequence Spread Spectrum and uses an Offset Quadrature Phase Shift Keying (O-QPSK) with half-sine pulse shaping to modulate the RF carrier.

The 802.15.4 standard allows for communication in a point-to-point or a point-to-multipoint configuration.

3.1.2 ZigBee

ZigBee is a protocol that uses the 802.15.4 standard as a baseline and adds additional routing and networking functionality.The ZigBee protocol was developed by the ZigBee Alliance for commercial and industrial low data rate applications.

The ZigBee protocol within the radios will take care of retries, acknowledgements and data message routing. ZigBee also has the ability to self-heal the network.

Since the ZigBee protocol uses the 802.15.4 standard to define the physical and Medium Access Controllayer (MAC), the frequency, signal bandwidth and modulation techniques are identical.

3.1.3Why ZigBee?

The lower data rate of the ZigBee devices allows for better sensitivity and range, but offers fewer throughputs. The primary advantage of ZigBee lies in its ability to offer low power and extended battery life. Because ZigBee was designed for low power applications, it fits well into embedded systems and those markets where reliability and versatility are important but a high bandwidth is not.

Table 3.1 : Comparison of Wireless technologies

	802.15.4 and ZigBee	**GSM/GPRS CDMA**	**802.11**	**Bluetooth**
Focus Application	Monitoring and Control	Wide Area Voice and Data	High-Speed Internet	Device Connectivity
Bandwidth	250 Kbps	Up to 2 Mbps	Up to 54Mbps	720 Kbps
TypicalRange	100+ Meters	Several Kilometers	50-100 Meters	10-100 Meters
Advantages	Low Power, Cost	Existing Infrastructure	Speed, Ubiquity	Convenience

3.1.4 XBee ZigBee RF Module 2.4 GHz

The radio modules, which we used for communication between our computer and the quadcopter, are XBee modules from Digi Internationals. The model used is high power 100mW XBee PRO with a 20 pin through-hole form factor and a wired antenna.

We used XBee PRO in API Mode. In API mode the data is wrapped in a packet structure that allows for addressing, parameter setting and packet delivery feedback, including remote sensing and control of digital I/O and analog input pins. It communicates serially via Universal Asynchronous Receiver-Transmitter (*UART*) over a secured wireless connection.

Figure 3.1: XBee radio(through-hole with wire whip antenna type)
Courtesy : http://www.adafruit.com/

3.1.4.1 Features:

i. Low-cost.
ii. Low-power.
iii. 128-bit Advanced Encryption Standard (AES)encryption.
iv. Frequency agility.
v. Over-the-air firmware updates (change firmware remotely).
vi. Transmitter current:205mA @3.3V, Power-Down current: 3.5uA @25degree C.
vii. 1200bps to 1 Mbps serial data rates.

viii. 63mW output (+18dBm).
ix. (4) 10-bit ADC inputs.
x. 10 digital IO pins.
xi. Voltage requirement 2.7-3.6 VDC.
xii. Receive Current 47mA.
xiii. Interference immunity by Direct Sequence Spread Spectrum.
xiv. *Radio Frequency (RF)* Data Rate 250 Kbps.
xv. Line of Sight Range 3200 m.

The XBee radios can all be used with the minimum four numbers of connections – power (3.3 V), ground, data in and data out (UART), with other recommended lines being Reset and Sleep.Additionally, most XBee families have some other flow control, I/O, A/D and indicator lines built in.

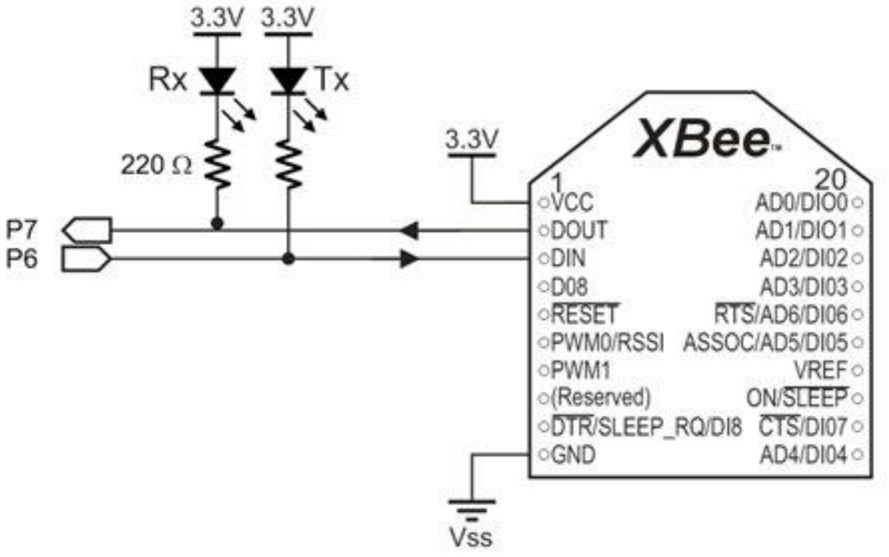


Figure 3.2: Pin Diagram of a XBee Module
Courtesy : http://www.learn.parallax.com/

3.1.4.2 Data Transfer

The message (the data we are sending) is packaged with other data the protocol requires. This added data typically includes: source address, destination address, error checking values, and other pertinent information needed by the protocol. Our message is encompassed with the other data to help ensure proper delivery to intended node.

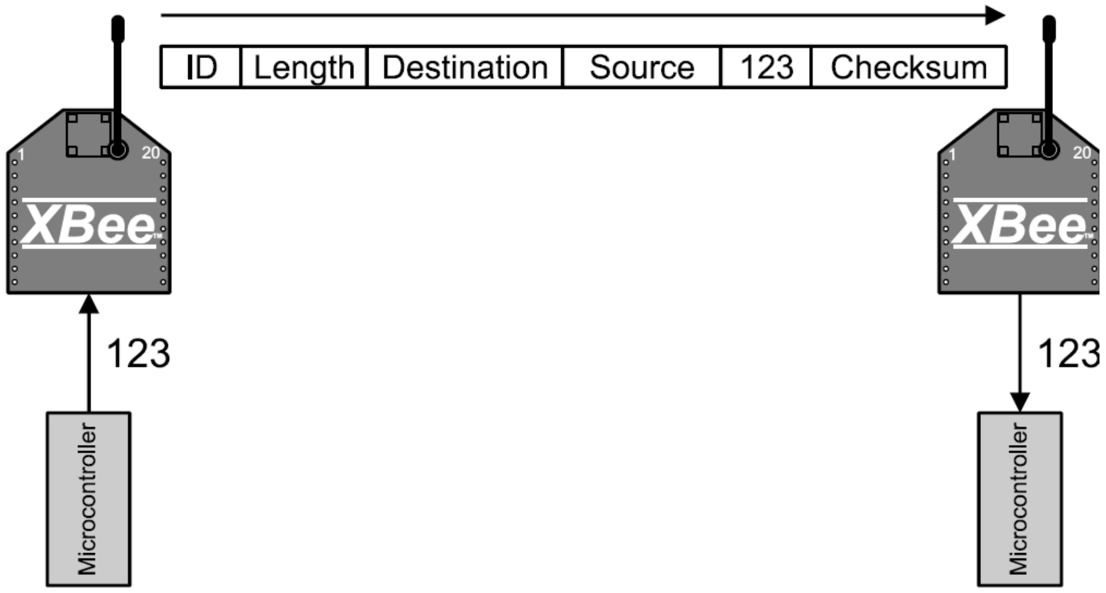


Figure 3.3: Pin Diagram of a XBee Module
Courtesy : http://www.makershed.com/

3.1.5 Configuring XBee Modules using X-CTU

The XBee modules that we use are configured using the software utility X-CTU. X-CTU is a Windows-based Applicationprovided by Digi. This program is designed to interact with the firmware files found on Digi's RF products and to provide a simple-to-use graphical user interface to them, to upgrade or change the firmware.

To configure the XBee module, pins of XBee module are connected to the shield. The shield plugged into the USB port. Open the X-CTU software. In the PC settings tab, select the COM port to which the module is connected. Choose the Baud Rate and select test/query to ensure that we are able to communicate with our XBee module. If we have chosen the baud rate same as that of the module, we will be able to communicate. If unable to test, try change the baud rate. Go to the Modem Configuration tab and read the parameters, by clicking on Read button. If we want we can change the parameters here and chose the Write button.

If we want to configure one module as transmitter and one as receiver, we connect each to the USB ports using shields. Then using two X-CTU windows for each module, we read their parameters. In the parameter list, address of one module is supplied as the destination for the other module and vice-versa. Also Coordinator Enable parameter for one module is set as zero (master) and one for the other module (acting as slave). Ensure that the systems work in coordination. Then only we will be able to communicate through them.

3.2 WiFly Module

WiFly module has host data rate up to 460Kbps over UART with Intelligent built-in power management with programmable wakeup events (timers and I/O). It has Real time clock for time stamping, auto-sleep and auto-wakeup modes .It can be Configured over WiFi or UART using simple ASCII commands. The air firmware can be upgraded via FTP. There are various secure WiFi authentication modes like WEP, WPA-PSK (TKIP), WPA2-PSK with built in networking applications *DHCP, DNS, ARP, ICMP UDP, Telnet, FTP, HTML* client. Transmitted power is configured between 2dBm to 12dBm. It has built-in HTML web client to send GPIO, UART and sensor data to remote web servers. The ADCs provide 14-bit resolution while the GPIOs can be configured to provide standard functionality or status signaling to a host microcontroller to reduce the need for serial polling between the Wi-Fi module and host microcontroller.

The module is pre-loaded with firmware to simplify integration and minimize development time of your application. In the simplest configuration, the hardware only requires four connections (PWR, TX, RX and GND) to create a wireless data connection.

Steps for installation of WIFLY Module

i. Setup the Wi-Fi Router
   - Power On the router
   - Connect Windows Laptop to the Router network using Wi-Fi. (e.g. For CISCO, default SSID: CiscoXXXXX)
   - Open any web browser and type the address 192.168.1.1in the URL bar
   - Default username : admin, password: admin
   - Change the SSID of the network to some convenient name

ii. Setup WiFly module drivers
   - Put the WiFly module into the USB to Serial converter board
   - Connect the USB to Serial converter board to the laptop
   - Start Device Manager>Other Devices >Click on Unknown Device >Update
   - Driver (If Installing for the first time)
   - Install the drivers. (Path :FT232>CDM 2.08.24 WHQL Certified)
   - Open Bray Terminal.
   - Select COM port and connect.
   - Enter command mode. (By typing $$$) and type following commands for Auto pairing type of UDP connection.

```
set w a 1  //authorization: 0=open
set w j 255     //policy for joining the
network, matched stored ssid
set w s <QUAD>        //SSID for the access
point to connect
set wlan channel  0           // for
scanning continuously
set ip proto 1 // enable UDP as the protocol
```

   - save
   - reboot

3.3 Reading the radio inputs and controlling the motors

We are using 6 channel radio controller for the controlling the quadcopter manually. Each radio output transmits a pulse at 50Hz with the width of the pulse determining where the stick is on the RC transmitter. Typically, the pulse is between 1000us and 2000us long with a 18000us to 19000us pause before the next - so a throttle of 0 would produce a pulse of 1000us and full throttle would be 2000us.

But most of the available radio modules are not at precise, so we have to calibrate the radio controller by calculating the minimum and maximum pulse width for each stick.

Motors are controlled through the Electronic Speed Controllers (ESCs). They work on pulse widths between approximately 1000us and 2000us like the RC radio receiver - sending a pulse of 1000us typically means off, and a pulse of 2000us means fully on. The ESCs expect to receive the pulse at 50Hz normally, but most of ESCs average the last 5-10 values and then send the average to the motors.This can work on quadcopter but it would be better if give instantaneous values to the motors without passing it through the averaging filter. So to compensate its effect the Arducopter library sends the pulse at ~490 Hz.

Chapter 4

Quadcopter Navigation Algorithms

4.1Altitude Heading Reference System

4.1.1 What is an AHRS?

This question has been bugging us since the start of the internship. Usually, AHRS is a dedicated IC chip which gives the result/final output as orientation of the quadcopter instead of the raw sensor data. The result/output is the 'Attitude'. The name of this chip in our case is MPU6050 with an in-built Digital Motion Processor(DMP)[2].

First we have to understand what an IMU is. An IMU consists of 3 accelerometers and 3 gyroscopes and optionally a magnetometer. The number of sensor inputs in an IMU is referred to as Degrees of Freedom (DOF), so a chip with a 3-axis gyroscope and a 3-axis accelerometer would be a 6-DOF IMU. Usually we have a 6 DOF IMU. In our project we have used the MPU 6050 IMU having 6 DOFs.We have HMC5883L magnetometer.

We must understand that there are two different things: Attitude estimation and Position estimation. Note that Attitude is in no way related to the X-Y-Z coordinates of the multi-copter. It is related with the orientation of our UAV. The Arducopter uses a Direction Cosine Matrix (DCM) based AHRS for attitude estimation. This has been coded in the AP_AHRS folder in the libraries folder of our code. This library is related to AHRS. The word 'DCM' has been mentioned in this file. This code is based on a paper by William Premerlani [9]. INS is related with position estimation and is explained in coming sections.

The orientation of a 3D rigid body is often described by *yaw*, *pitch*, and *roll* angles. They are known as Euler angles are (usually pronounced by Americans as *Oiler Angles)*.They are convenient for making figures like the one on the left, but later cause a lot of trouble due to numerical singularities (for example gimbal lock) and a huge variety of alternative, incompatible definitions . Representation by Quaternions does not suffer from gimbal lock. A unit vector multiplied by a quaternion will be taken from the basis orientation to the orientation represented by the quaternion. We are using DCM representation for orientation, and it too does not suffer from numerical singularities.

A generalized diagram of an AHRS system is shown in Figure 4.1 [8].

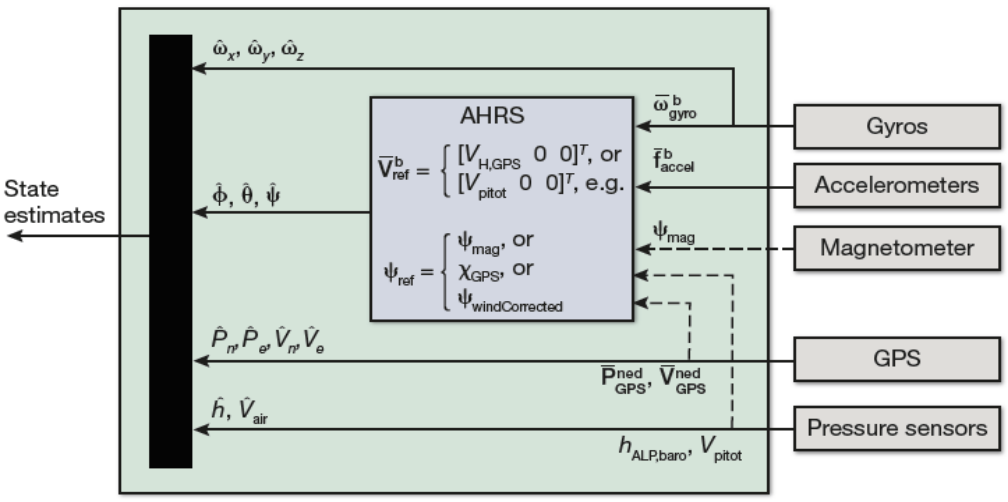

$\hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z$
$\bar{\omega}^b_{gyro}$
AHRS
$\bar{V}^b_{ref} = \{ [V_{H,GPS}\ 0\ 0]^T$, or $[V_{pitot}\ 0\ 0]^T$, e.g.
$\psi_{ref} = \{ \psi_{mag}$, or χ_{GPS}, or $\psi_{windCorrected}$
$\bar{f}^b_{accel}$
ψ_{mag}
State estimates
$\hat{\phi}, \hat{\theta}, \hat{\psi}$
$\hat{P}_n, \hat{P}_e, \hat{V}_n, \hat{V}_e$
$\bar{P}^{ned}_{GPS}, \bar{V}^{ned}_{GPS}$
$\hat{h}, \hat{V}_{air}$
$h_{ALP,baro}, V_{pitot}$
Gyros
Accelerometers
Magnetometer
GPS
Pressure sensors

Figure 4.1: Generalized AHRS system

Courtesy : Barton, Jeffrey D. "Fundamentals of small unmanned aircraft flight." Johns Hopkins APL technical digest 31.2 (2012): 132-149.

4.1.2 Basics of a Complementary Filter

Accelerometer measures gravity minus acceleration. The magnetometer acts like a compass.Actually, just the accelerometer output and the magnetometer output are sufficient to provide a full orientation of the UAV. The reader must be asking himself how accelerometers can decide orientation. Accelerometers can give orientation by deciding the direction of the gravity vector.

However, they (especially the magnetometer output) are very noisy.Here is where, the gyroscope comes in to the picture.Accelerometers measure all the forces present. Hence in addition to the gravity, the force acting on our quad-rotor would also be acting on the sensor. This is the vibration noise that we are talking about.Gyroscopes in the 3 axes, measure the angular velocity about 3 axes. Gyroscopes are highly accurate devices.Gyroscope readings, though noise immune are prone to a time varying drift.

To get the actual orientation the angular speed values need to be integrated over time. This is done by multiplying the angular speeds with the time interval between the last and the current sensor output. This yields a rotation increment. The sum of all rotation increments yields the absolute orientation of the device. During this process small errors are introduced in each iteration. These small errors add up over time resulting in a constant slow rotation of the calculated orientation, the gyro drift.Let us understand Gyro drift with the help of an example. Consider a quadcopter.Consider rotation only about one axis, say Y axis.Assuming 1000 sensor readings per second, the update equation is as follows.

Current orientation = Previous orientation + 0.001* ω [1]

The update equation of a multi-copter having angular velocity about all the 3 axes is also similar to the above equation, just a little more complicated. This update equation is in terms of Euler angles.However this can be visualized better in terms of quaternions.After many thousands of updates, the true orientation will drift away from the calculated orientation.

To avoid both, gyro drift and noisy orientation, the gyroscope output is applied only for orientation changes in short time intervals, while the magnetometer/accelerometer data is used as support information over long periods of time. Accelerometer data is reliable over long time intervals. Gyroscope data is reliable over a short interval of time.This is equivalent to low-pass filtering of the accelerometer and magnetic field sensor signals and high-pass filtering of the gyroscope signals. This is the basis of complementary filter.

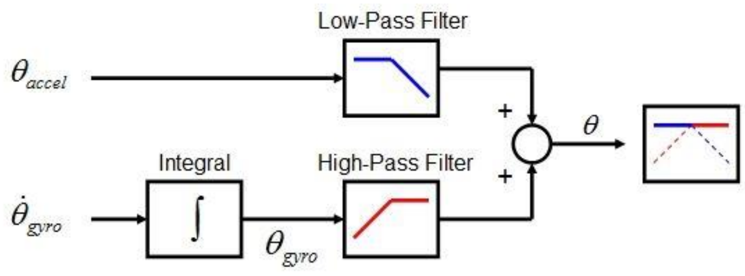


Figure 4.2: The basis of Complementary Filter
Courtesy : http://www.pieter-jan.com/node/11

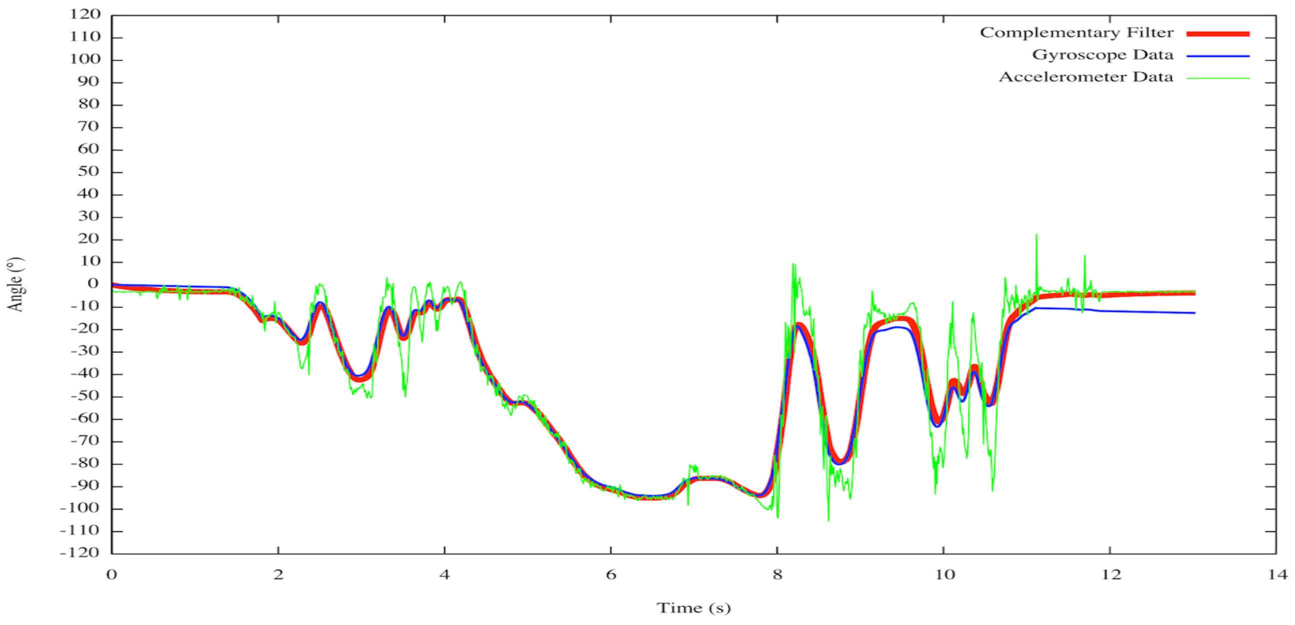


Figure 4.3: Sensor Fusion by Complementary Filter
Courtesy : http://www.pieter-jan.com/node/11

4.1.3 Comparison between Complementary Filter and Kalman Filter

Now we will compare the DCM algorithm and the Extended Kalman Filter (EKF) or Kalman Filter(KF). Actually the proper terminology is 'Comparison between EKF and Complementary Filter'.KF is for linear problems and EKF is for non-linear problems.

4.1.3.1 Advantages of Complementary Filter over KF/EKF

The DCM algorithm is simpler and much more computationally efficient. EKF will tolerate a larger initial gyro offset than the DCM algorithm, but once the algorithms achieve *gyro lock*, they produce nearly identical results. Usually when the quadcopter starts, it is stable and still. Hence there is very little gyro offset. Hence DCM and EKF have almost similar performance.The 2 primary disadvantages of implementing a Kalman Filter:

i. Very difficult to understand.
ii. Very difficult to implement on certain hardware platforms. DCM is better suited for 8-bit processors.
iii. High computational complexity.

Complementary Filter can be represented by just one equation (no complexity).

$$angle = 0.98 * (angle + gyrData * dt) + 0.02 * (accData)$$ [2]

The constants (0.98 and 0.02 here) have to add up to 1. The 0.02 here is a constant α and 0.98 is 1-α.

4.1.3.2 Advantages of KF over Complementary Filter

The advantage of the EKF over the simpler complementary filter algorithms used by DCM and INS is that by fusing all available measurements it is better able to reject measurements with significant errors so that the vehicle becomes less susceptible to jumps in GPS position or barometric altitude. Another feature of the EKF algorithm is that it is able to estimate offsets in the vehicles magnetometer readings and also estimate the earth's magnetic field for both plane, copter and rover applications. This makes it less sensitive to compass calibration errors current DCM and INAV algorithms. Advantages of Kalman Filter/EKF over Complementary filter:

i. Very Accurate. The EKF outperforms Complementary Filter by a factor of 10.
ii. KF converges to the solution quickly.

4.1.4 Basics of the DCM algorithm

This article is based on the paper by William Premerlani.The version of Arducopter which we are using has implemented AHRS with the help of DCM matrices.The reader should not get confused between the terms DCM algorithm and Complementary Filter.The actual process used by Bill Premerlani in filtering/combining data (called Sensor Fusion) from different sensors uses a complementary filter approach.The term 'DCM algorithm' is really a misnomer.DCM can be considered an integrated 3-dimensional non-linear complementary filter (with independent proportional and integral paths). This is the reason that we covered the theory of complementary filters first.

The overall block diagram of the implemented system is:

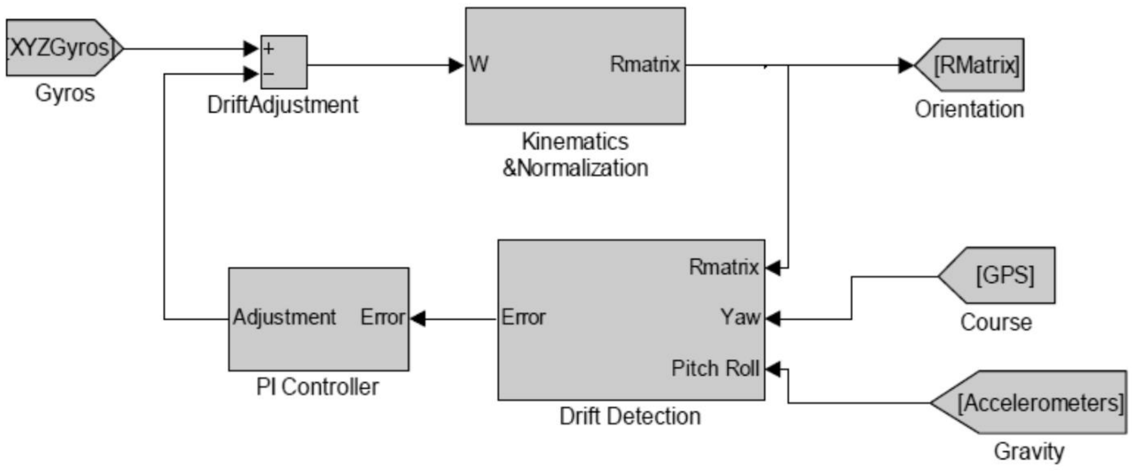


Figure 4.4: Overall Flow of our AHRS system

Courtesy : Premerlani, William, and Paul Bizard. "Direction cosine matrix imu: Theory." DIY Drones.[Online][Cited: 1 7 2012.] http://diydrones. ning. com/profiles/blogs/dcm-imu-theory-first-draft (2009).

Rotation matrices (instead of quaternions) are used in this paper. A rotation matrix describes the orientation of one coordinate system with respect to one another. A vector in one system can be transformed in to another system by multiplying it with the rotation matrix. Transformation in the reverse direction can be accomplished with the help of inverse of the rotation matrix. The inverse of a rotation matrix is its transpose (since it is a square matrix).

An advantage of DCM matrices is that they are free from singularities. Euler angle representation suffers from singularities like gimbal lock. Another advantage of DCM representation is that rotation matrices can be cascaded, i.e. multiplication by a product of 2 rotation matrices signifies that the unit vector undergoes 2 rotations in series, each corresponding to the 2 matrices respectively. Rotation matrices are used to convert orientation of body in earth fixed frame to body fixed frame and vice versa, as is demonstrated below.

Q_G : a vector Q measured in the frame of reference of the ground

Q_P:a vector Q measured in the frame of reference of the plane

R : *Rotation matrix*

$$R = \begin{pmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{pmatrix} \qquad [3]$$

We have, $Q_G = R.Q_P$ [4]

The relation between the rotation matrix and Euler angles is given by [5]:

$$R = \begin{pmatrix} \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\phi\sin\psi & \sin\theta\cos\phi\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\theta\sin\phi\sin\psi + \cos\phi\cos\psi & \sin\theta\cos\phi\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \quad [5]$$

This matrix R is called the Direction Cosine Matrix.This DCM is continuously updated at certain time intervals.The rate of change of a rotating vector is related to its angular velocity by the following kinematics equation:

$$\frac{dr}{dt} = (\omega)x(r) \qquad [6]$$

The above differential equation is non-linear. We have to integrate ω in order to get the angle. Hence any linear approach is going to be an approximation.

Integrating the above equation,

$$\mathrm{r(t)} = \mathrm{r(0)} + \int_0^t (d\theta(\tau))\,\mathrm{x}(r(\tau))$$

Gyroscopic measurements are made in the frame of reference of the air-craft.

But we would be interested to tack the orientations of the aircraft with respect to the earth frame!

To adjust for that,

$$\mathrm{r_{earth}(t)} = \mathrm{r_{earth}(0)} + \int_0^t r_{earth}(\tau)\,\mathrm{x}\,d\theta(\tau)$$

Consider the differential form of the above equation. We are going to use this for practical implementation.

$$\mathrm{r_{earth}(t+dt)} = \mathrm{r_{earth}(t)} + r_{earth}\mathrm{(t)}\,\mathrm{x}\,d\theta\mathrm{(t)}$$

$$d\theta\mathrm{(t)} = \omega(t).dt \qquad [7]$$

Due to gyro drift/bias, we need to add a correction factor-appropriately called ‘drift correction factor’.

$$\omega(t) = \omega_{gyro}(t) + \omega_{correction}(t) \qquad [8]$$

The gyro correction vector (sometimes called rotational error) is computed on the basis of accelerometer and GPS data and then is fed back through a feedback controller.

The update equation for the DCM is given by

$$R(t+dt) = R(t)\begin{pmatrix} 1 & -d\theta_z & d\theta_y \\ d\theta_z & 1 & -d\theta_x \\ -d\theta_y & d\theta_x & 1 \end{pmatrix} \qquad [9]$$

$$d\theta_x = \omega_x.dt$$
$$d\theta_y = \omega_y.dt$$
$$d\theta_z = \omega_z.dt$$

Thus, we can practically implement this system by performing matrix multiplications with short time steps.However if we implement the equation without correction factor, eventually, numerical round-off, gyro offset and gain errors would be accumulated.

4.1.4.1 Renormalization

We have, $\mathrm{R}^{\mathrm{T}}.\mathrm{R} = \mathrm{I}$

The above equation is known as orthogonality condition. Numerical errors (due to round off and quantization) violate this condition. The process of correcting this error is known as renormalization.

This is implemented in AP_AHRS_DCM.cpp of the AP_AHRS library [10]. The renormalization procedure is described below.

$$X = \begin{pmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{pmatrix} \quad , Y = \begin{pmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{pmatrix}$$

$$error = X^T.Y \tag{10}$$

$$X_{orthogonal} = X - \frac{error}{2}.Y$$

$$Y_{orthogonal} = Y - \frac{error}{2}.\mathrm{X}$$

It can be verified that the error is reduced greatly when we substitute the previous 2 equations in the error equation [Equation 9], the error is greatly reduced.

Further, we have,

$$\mathrm{Z}_{\mathrm{orthogonal}} = (X_{orthogonal})\,\mathrm{x}(\mathrm{Y}_{orthogonal}) \tag{11}$$

Finally,

$$X_{normalized} = \frac{1}{2}(3 - X_{orthogonal}.X_{orthogonal})X_{orthogonal}$$

$$Y_{normalized} = \frac{1}{2}(3 - Y_{orthogonal}.\mathrm{Y}_{orthogonal})\,\mathrm{Y}_{orthogonal} \tag{12}$$

$$Z_{normalized} = \frac{1}{2}(3 - Z_{orthogonal}.\mathrm{Z}_{orthogonal})\,\mathrm{Z}_{orthogonal}$$

The new DCM is then,

$$R^T = \begin{pmatrix} X_{normalized} & Y_{normalized} & Z_{normalized} \end{pmatrix} \tag{13}$$

All the above steps are performed at each time step interval.

4.1.4.2 Gyro Drift Cancellation

The idea is to use other orientation references and provide a negative feedback loop to the gyros so that they cancel the drift.

The steps are:

i. Detect the error. This can be done by computing the vector which will bring the computed reference vector and the measured reference vector into alignment. This forms our error vector.

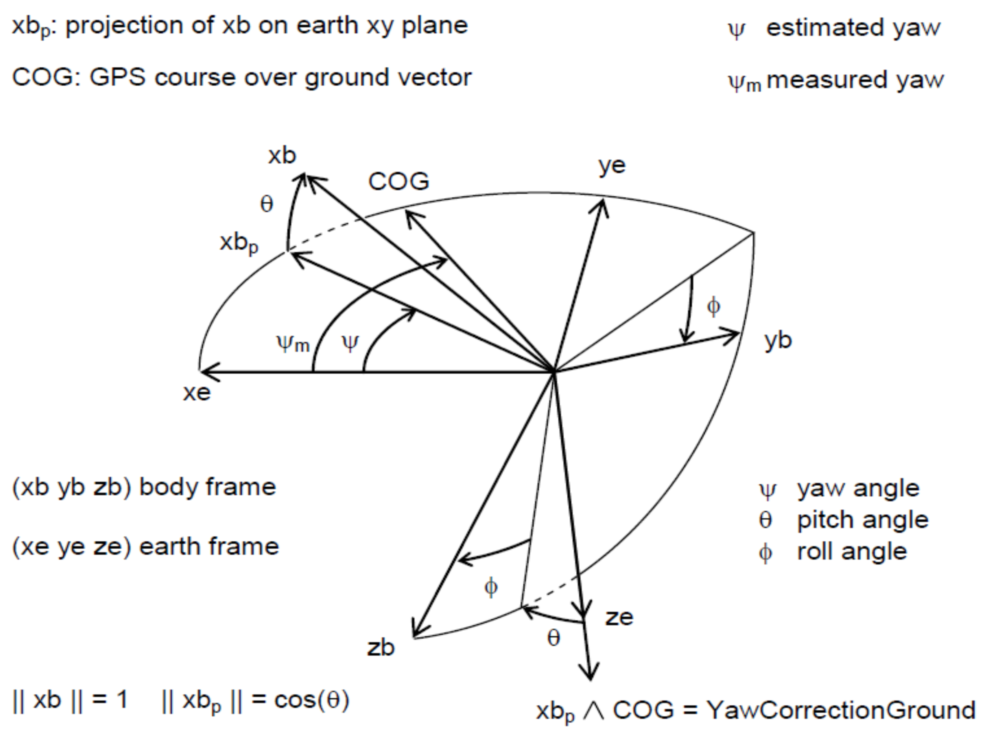
ii. Provide the error vector as input to a PI feedback controller.

iii. Algebraically add/subtract the output of the PI controller to the gyro values.

In this algorithm, the orientation vectors are supplied by the accelerometer and GPS. These 2 are used because they do not drift. Magnetometers are also useful, especially in Yaw correction where Altitude Hold/Hover is required.Accelerometers are used to provide a reference vector for Z axis of the quadcopter. GPS is used for providing reference vector for the roll axis of our quadcopter.The error vector is given by the cross product of the measured vector and that approximated by the DCM.We use a proportional plus integral feedback controller to apply the rotation correction to the gyros, because it is stable and because the integral term completely cancels gyro offset, including thermal drift, with zero residual orientation error.

A. Yaw Drift Correction

Before we start, we may state that Arducopter has a provision of estimating the Yaw drift either with a GPS or a Compass/Magnetometer.Some hobbyists are of opinion that magnetometer estimates Yaw drift better. GPS provides a drift free reference vector for Yaw correction.GPS provides us with position/location, velocity magnitude and even direction.Our GPS provides data in NMEA format.GPS delivers its information at continuous time intervals.GPS has a reporting latency-a time interval between calculation of the position and velocity and its appearance as a sequence of messages. All GPS units perform some sort of filtering to improve the accuracy of position and velocity estimation. This also introduces a delay. The GPS horizontal course over ground signal has zero drift over the long term, and can be used as a reference vector to achieve “yaw lock” for the IMU.



xb$_p$: projection of xb on earth xy plane
COG: GPS course over ground vector
ψ estimated yaw
ψ$_m$ measured yaw
xb
θ
COG
ye
xb$_p$
φ
ψ$_m$
ψ
yb
xe
(xb yb zb) body frame
(xe ye ze) earth frame
ψ yaw angle
θ pitch angle
φ roll angle
φ
ze
θ
zb
|| xb || = 1 || xb$_p$ || = cos(θ)
xb$_p$ ∧ COG = YawCorrectionGround

Figure 4.5: Yaw drift correction

Courtesy : Premerlani, William, and Paul Bizard. "Direction cosine matrix imu: Theory." DIY Drones.[Online][Cited: 1 7 2012.] http://diydrones. ning. com/profiles/blogs/dcm-imu-theory-first-draft (2009).

The rotational correction is the Z component of cross product of X column of R matrix and the COG vector.

$$COGX = \cos(cog)$$

$$COGY = \sin(\text{cog})$$

$$\text{YawCorrectionGround} = r_{xx} COGY - r_{yx} COGX \qquad [13]$$

To get the correction in body frame,

$$YawCorrectionQuad = YawCorrectionGround \begin{pmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{pmatrix} \qquad [14]$$

B. Roll-Pitch Drift Correction

Accelerometers provides reference vector for roll-pitch drift correction.

We know that accelerometers measure gravity minus acceleration. Hence an accelerometer is a roll-pitch indicator only when it is not accelerating.

Various accelerations possible are forward acceleration, centrifugal acceleration etc.

$$A_{centrifugal} = \omega_{gyro} \, \text{x} V \qquad [15]$$

$$\text{g}_{\text{reference}} = \text{Accelerometer output} + \omega_{gyro} \, \text{x} V \qquad [16]$$

$$\text{Accelerometer output}=\begin{pmatrix} Accelerometer_x \\ Accelerometer_y \\ Accelerometer_z \end{pmatrix} \quad [17]$$

To adjust for the difference in frame orientations,

$$RollPitchCorrectionQuad = \begin{pmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{pmatrix} \text{x } \mathrm{g}_{\text{reference}} \quad [18]$$

4.1.4.3 Feedback Controller

Each of the rotational drift correction vectors (yaw and roll-pitch) are multiplied by weights and fed to a proportional plus integral (PI) feedback controller to be added to the gyro vector to produce a corrected gyro vector.

$$\text{Total Correction} = W_{RP} RollPitchCorrectionQuad + W_Y YawCorrectionQuad$$

[19]

$$\omega_{PCorrection} = K_P.\text{Total Correction}$$

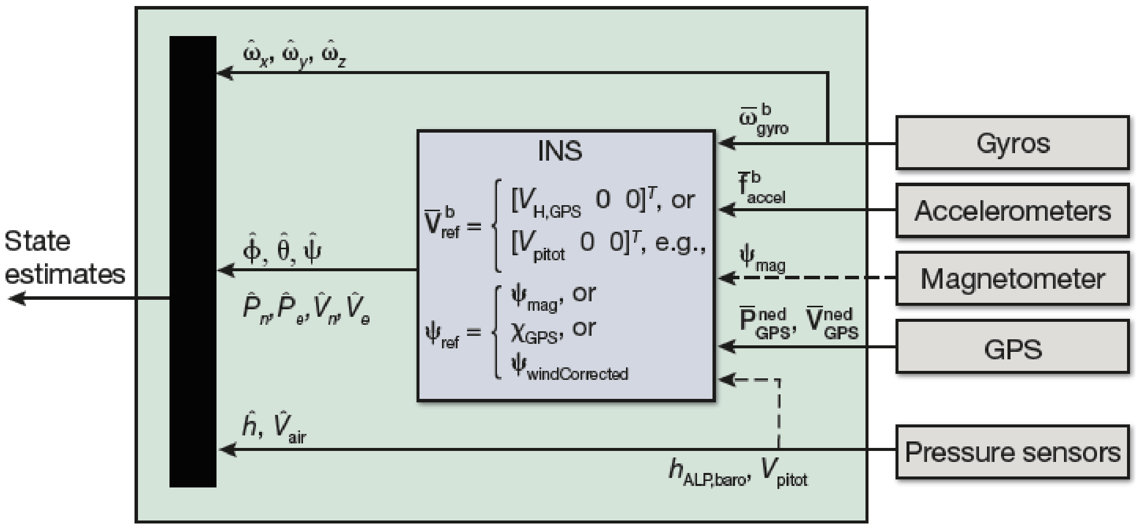
$$\omega_{ICorrection} = \omega_{ICorrection} + K_I.\text{dt.Total Correction} \quad [20]$$

$$\omega_{Correction} = \omega_{PCorrection} + \omega_{ICorrection}$$

This correction vector is then substituted in equation 8, and the new DCM matrix is estimated. All the above steps have to be done again, in the next time step.

4.2 Inertial Navigation System

For position estimation, you need to look in to AP_InertialNav library [10]. This library blends data from 3 sources: Accelerometer, GPS, Barometer and provides an X-Y-Z estimate of the UAV. This library is related to INS.3rd order complementary filter is used for INS in the Ardupilot code. The complementary filter is used for sensor fusion of Barometer and Accelerometer data. The complementary filter can be replaced by Kalman Filter and I think this is the case in newer versions.If the UAV is manually controlled (either through a RC or through AAKASH tablet), just AHRS is enough. However, if the UAV has to travel along a trajectory autonomously (similar to setting waypoints in Mission Planner) we need INS too. Recently, in newer versions, the Ardupilot community has migrated towards EKF citing many advantages.The AP_AHRS library is included in AP_InertialNav (.h file). This compels us to think that the output of the AHRS block is somehow used in INS. A generalized INS is shown in Figure 4.6 [8].



$\hat{\omega}_x, \hat{\omega}_y, \hat{\omega}_z$
$\bar{\omega}^b_{gyro}$
Gyros
INS
$\bar{f}^b_{accel}$
Accelerometers
$\bar{V}^b_{ref} = \begin{cases} [V_{H,GPS}\ \ 0\ \ 0]^T, \text{ or} \\ [V_{pitot}\ \ 0\ \ 0]^T, \text{ e.g.,} \end{cases}$
State estimates
$\hat{\phi}, \hat{\theta}, \hat{\psi}$
ψ_{mag}
Magnetometer
$\hat{P}_n, \hat{P}_e, \hat{V}_n, \hat{V}_e$
$\psi_{ref} = \begin{cases} \psi_{mag}, \text{ or} \\ \chi_{GPS}, \text{ or} \\ \psi_{windCorrected} \end{cases}$
$\bar{P}^{ned}_{GPS}, \bar{V}^{ned}_{GPS}$
GPS
$\hat{h}, \hat{V}_{air}$
Pressure sensors
$h_{ALP,baro}, V_{pitot}$

Figure 4.6: Generalized INS system

Chapter 5

PID Tuning of Quadcopter

5.1Introduction to *PID* Parameters

PID (Proportional, Integral, Derivative) system is a closed loop (i.e. with a feedback) control system which is used for process control.

It calculates error between the actual parameter and the desired parameter and generates control signal to manipulate the output, so as to minimize the error.

The PID loop reads the actual value, compares it with the target value and calculates their difference/error. The controller in the loop then gives output/correction according to the error with an aim to reduce this error to zero.

For the purpose of control, three standard parameters are used, namely, *P (proportional), I(integral)* and *D (derivative)* parameters.

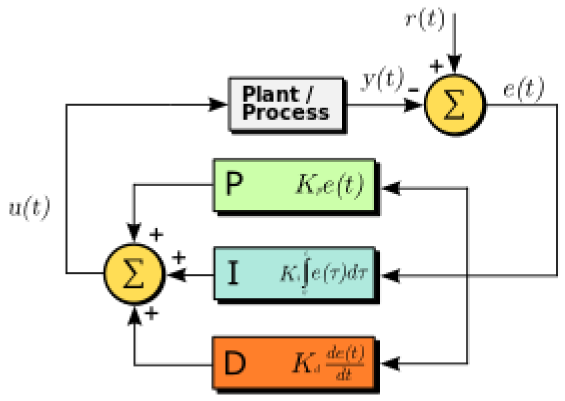


Figure 5.1 Block diagram of a PID feedback control mechanism
Courtesy :[10] http://dev.ardupilot.com/

P parameter depends on the present error, *I* component on the accumulation of past errors, and *D* parameter gives the future errors based on the rate of change of error. The weighted sum of these errors is used by the controller to adjust the process to get desired results, by varying the power supplied, by causing mechanical movements, etc. If u(t) is the controller output (or correction term), then

$$\mathrm{u}(t) = \mathrm{MV}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \qquad [1]$$

where

K_P: Proportional gain, a tuning parameter

K_i: Integral gain, a tuning parameter

K_d: Derivative gain, a tuning parameter

e: Error =Standard Parameter- Present Value

t: Time or instantaneous time (the present)

τ: Variable of integration; takes on values from time 0 to the present t.

5.1.1 Proportional Term

The first term in the correction is the *P* correction. It is equal to the product of proportional gain and present error. It gives the immediate error correction.

$$P_{\text{out}} = K_p \, e(t) \quad [2]$$

Larger value of proportional gain will make this correction greater with error. Too high value may cause the response to oscillate furiously, while too low value will cause the controller to be slow.

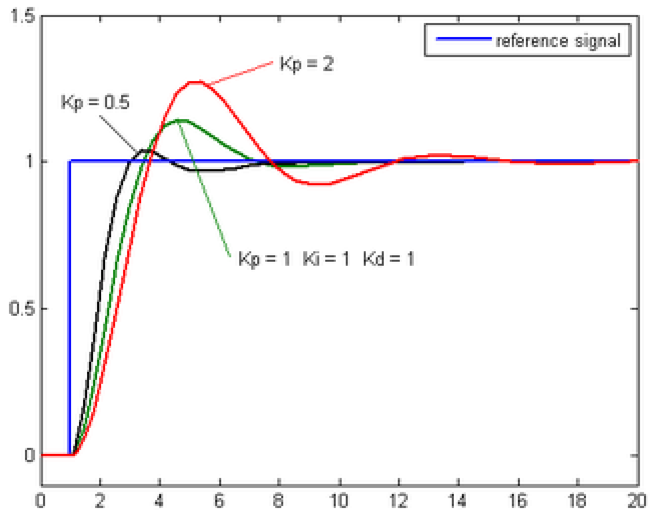


Figure 5.2 Plot of actual value for different values of Kp with Ki and Kd kept constant
Courtesy : http://dev.ardupilot.com/

5.1.2Integral Component

The second term in the correction is the integral component. It is the product of integral gain and cumulative error. It is dependent on accumulation/integration of instantaneous errors over time, i.e. on the magnitude, as well as, duration of error. The integral term is given by

$$I_{\text{out}} = K_i \int_0^t e(\tau) \, d\tau \quad [3]$$

It is used to remove steady state error and also to increase the speed of the response towards desired value. Larger value may cause the response to overshoot the target value.

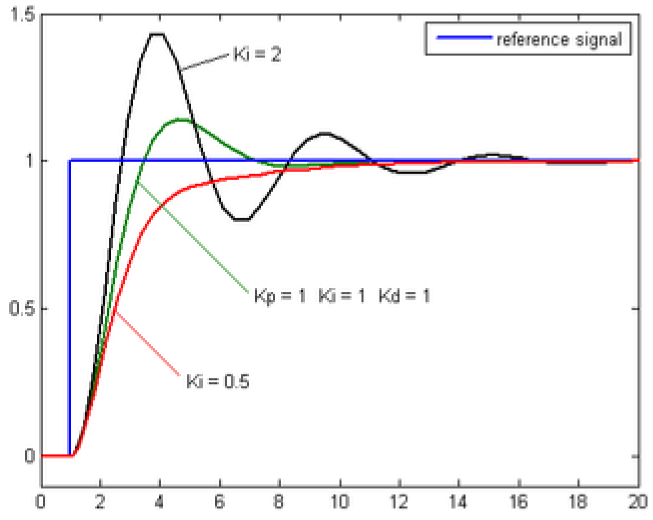


Figure 5.3 Plot of actual value for different value of Ki(Kp and Kd kept constant)
Courtesy : http://dev.ardupilot.com/

5.1.3 Derivative Component

The third term in the correction is the derivative term, which depends on the rate of change of error with time. It predicts the behavior of the system and helps improve stability and settling time of the system. The derivative term is given by

$$D_{\text{out}} = K_d \frac{d}{dt} e(t) \qquad [4]$$

This term is highly sensitive to noise, as it not causal. For controller with derivative action, it is accompanied by a low pass filter to limit the high frequency gain and noise.

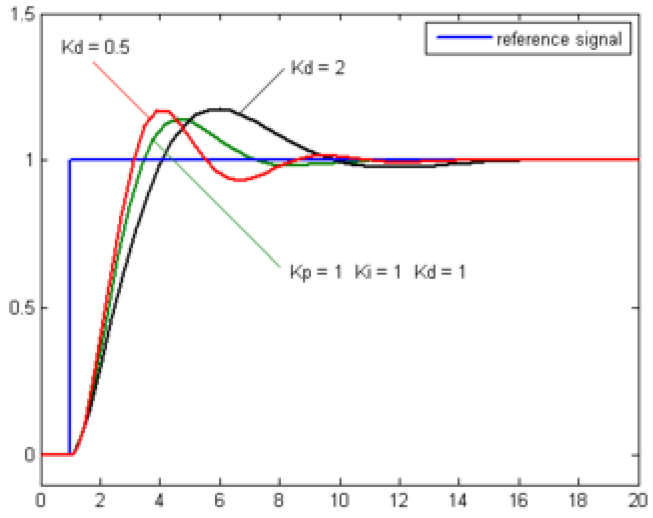


Figure 5.4 Plot of actual value for different values of Kd (keeping Ki and Kp constant)
Courtesy : http://dev.ardupilot.com/

5.2 PID Tuning

Tuning is the process of adjusting the controlling *PID* parameters to their optimum values to get the desired response. The aim of tuning is to make the system stable, as well as get the optimum or desired behavior from it. The optimum behavior can be achieving the set point, remaining on it, or following the set point changes, while stability can be oscillation or disturbance free output.

5.2.1 Tuning the quadcopter

Tuning is required for a stable and smooth flight of the quadcopter. We attempt to control the *P, I* and *D* parameters of the closed loop, which work to eliminate any deviation/error from the desired flight characteristics/path. These parameters take inputs from the stick movement of the transmitter and sensor readings and according to the error generated, will take necessary control actions.

In the absence of proper *PID* tuning, the quadcopter will not give desired flight characteristics. It may just topple over, may be sluggish or may become untamable.

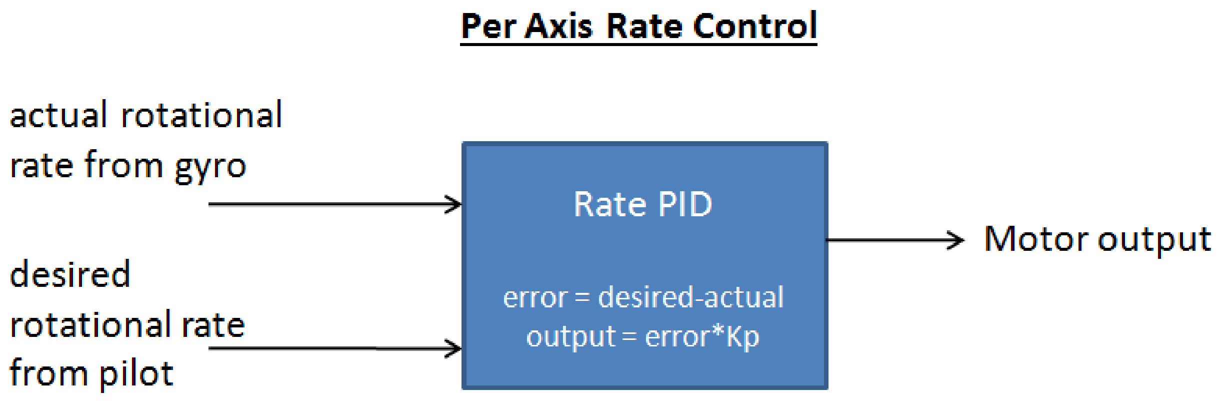
Apart from these, we need to keep in mind the surrounding conditions as well as, the weight and type of quad frame and propellers.

5.2.2 Insight to the working of *PID* in quadcopter

There are following underlying controllers in the quadcopter:

1. Rate Controllers
2. Throttle Controllers
3. Yaw Controllers
4. Roll/Pitch Controllers
5. Navigation Controllers

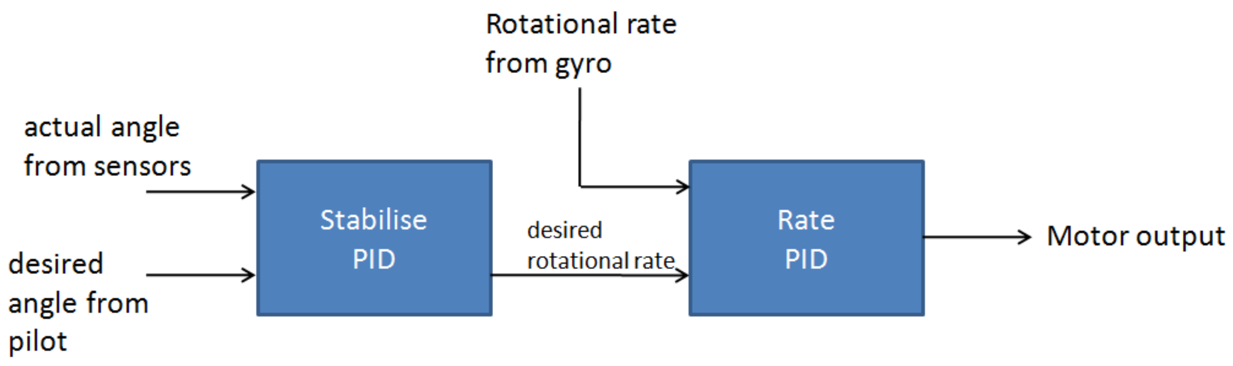
Rate controllers are the basic controllers and are always working, no matter which flight mode we work upon, when other controllers might/ might not be active. Rate controllers give the rate of rotation in terms of roll, pitch and yaw changes, or vertical acceleration in case of throttle.



Each of the roll-pitch, yaw and throttle controllers is responsible for:

1. Converting the pilot's input received from the tx/rx into a desired angle, altitude, etc.
2. Calling the appropriate *PID* controllers to implement the desired angle, altitude, etc.
3. Passing a desired rate or acceleration to the rate controllers.

Per Axis PID structure



The low level body-frame rate targets for these rate controllers are set when they call the `set_roll_rate_target`, `set_pitch_rate_target`, `set_yaw_rate_target` or `set_throttle`.

The rate controllers are called at 100 Hz from the main loop and are implemented as `run_rate_controllers()` in Attitude.pde.

A particular roll-pitch, yaw or throttle controller is activated by calling the following functions which can all be found in ArduCopter.pde.

1. `set_roll_pitch_mode()`
2. `set_yaw_mode()`
3. `set_throttle_mode()`

Similarly the following three functions are called regularly (100 Hz or 50 Hz) from the main loop to allow the controllers to read in the latest pilot input and call the appropriate controllers that are normally held in Attitude.pde:

1. `update_roll_pitch_mode()`

2. `update_yaw_mode()`

3. `update_throttle_mode()`

The earth frame controllers take input/target angles for pitch, roll and yaw from the pilot stick using `get_stabilize_pitch`, `get_stabilize_roll` and `get_roll_rate_stabilized_ef` functions respectively. These target angles are then compared with the actual angular values for pitch, roll and yaw, obtained from the attitude sensors. The difference of these is used to get correction value to give the output rate of rotation, which is fed to the low level body frame rate controllers, as desired rate of rotation, by calling functions `get_rate_pitch`, `get_rate_roll` and `get_rate_yaw.`

The actual rate changes for pitch, roll and yaw are obtained from the rate sensors. Their differences with their ideal values give errors for pitch, roll and yaw rate. PID tuning with proportional, integral and derivative corrections is performed so that error is minimized and we get the desired rate change. These tuned values/corrections are sent to ESCs to get the desired output.

With the derivative correction, a low pass filter has been provided to remove the effect of high frequency noise. In the case of integral correction, we specify a maximum limit for the correction as the cumulative error may grow out of the desired bounds.

5.3 PID Parameters

We use Mission Planner 1.3.5 config/tuning window for PID tuning. The parameters which we tune include:

1. Stabilize Pitch P
2. Stabilize Roll P
3. Stabilize Yaw P
4. Rate_Roll P
5. Rate_Pitch P
6. Rate_Yaw P
7. Rate_Roll I
8. Rate_Pitch I
9. Rate_Yaw I
10. Rate_Roll D
11. Rate_Pitch D
12. Rate_Yaw D
13. Loiter PID P
14. Loiter Rate P
15. Loiter Rate I

16. Loiter Rate D

17. Throttle Rate P

18. Throttle Rate D

19. Throttle Accel P

20. Throttle Accel I

21. Throttle Accel D

22. Altitude Hold

Proportional component gives output response in proportion to the error signal. Increasing the proportional gain will make the copter reach the desired position but increasing the gain too much will make the quadcopter oscillate furiously.

The integral component reduces the steady state error between the desired and actual response, till it increases to a saturated value where the steady state error reduces to zero. Since I component is based on integration of past errors, it may cause the response to overshoot the desired one.

Derivative component gives output in response to the rate of change of error signal. Increasing the derivative gain will make the quadrotor reach the desired position quickly. But the derivative gain is highly sensitive to surrounding noise, so it may make the quad flight very unstable. Hence we keep this component either very small or null.

IMAX gives the maximum possible build up, i.e. the maximum possible value we can provide for that parameter for tuning. Example, Rate Yaw Imax gives maximum possible value for Yaw.

Stabilize Roll/Pitch P alter angular rotation to get the desired angular position in the Stabilize mode. A high value will make copter respond quickly to stick inputs but larger values will cause it to oscillate more, while a smaller value makes it a sluggish system.

Stabilize Yaw P determine the heading of the quad. Too high values may cause the quad heading to oscillate, while a very small value makes it difficult for the quad to maintain its heading.

Roll Rate/Pitch P give desired rate of rotation in acro mode according to the stick input. Higher value means faster rate of rotation. But very high values may increase instability.

Roll Rate/Pitch I give finer adjustment by removing steady state error, while Roll Rate/pitch D cause faster response.

Roll Yaw P convert stick movements to desired rotation rate in the direction of heading. Very small values make it difficult for quad to maintain its heading, while larger values may cause oscillations.

Roll Yaw I remove error between desired and actual rate of rotation and Roll Yaw D increase the speed of response.

Loiter PID P is used to move the quadrotor to the desired location in the loiter mode by adjusting its speed. It is not altered much.

There is another parameter too, Loiter PID I which is not user editable. The copter uses it for overcoming the force of the wind.

Loiter Rate P provides the required angle/tilt. Higher this term, closer is the actual tilt rate to the desired one.

Loiter Rate I works for further adjustments in the rotation rate, counterbalancing the outside forces. Increasing this parameter causes the actual rate to quickly move to desired rate without any overshoot. Very low values make the rotor behavior sluggish and may cause overshoot.

Throttle Rate P gives the amount of throttle output used to change the rate at which our quad ascends.

Throttle Rate I compensates for error in achieving desired climb rate. Generally it is kept to zero.

Throttle Rate D causes the quadrotor to quickly achieve the desired rate.

Throttle IMAX is the maximum possible build up of throttle.

Altitude Hold *P* is used to convert altitude error in centimeters to a desired climb_rate in centimeter/second. Higher value of this, will lead to faster climb rate, while a lower value results in a slow response.

Altitude Hold *I* is used to account for a copter having trouble holding altitude, usually due to a low voltage battery.

Altitude Hold *IMAX* gives the amount of throttle we can adjust.

5.4 Steps for Tuning

First connect your quad for telemetry, using mavlink on Mission Planner. Perform radio calibration. Also calibrate your compass, GPS and accelerometer.

If tuning for the first time, work with the default parameters or look up for the ideal parameters. Update the parameters.

Generally the Yaw value does not need to be modified. We mainly work on the roll and pitch values.

We start with the Stabilize mode. Work with *P* parameter first. Observe the quad fly. Increase the P value if quad follows stick slowly, or decrease the value if quad is getting out of control.

Once we set the optimum P value for the stabilize mode, then we move to adjusting the I values. It works to remove steady state error.

D value is not altered much, as it is very noise sensitive. Either set it to get the desired output or leave it unchanged.

Once we get the right parameters for stabilize mode, then we move to acro mode tuning.

We adjust the Rate_Roll P and Rate_Pitch P parameters. Increase or decrease the values as per the quad behavior.

Once you set the P values, then go for I parameter tuning.

After the acro mode, we can go for the loiter mode and alt_hold mode by working in a similar manner, i.e. first working on P parameters, then I, and again adjusting P if required, and finally moving to D parameter.

Adjust the parameters till we get the desired behavior.

The PID parameter values which we used are shown below:

5.5 Various Controllers

Most people are aware of the various flight modes including Stabilize, Alt-Hold, Loiter, AUTO but what most do not realize is that each of these flight modes are made up of 4 or 5 underlying controllers:

i. rate controllers
ii. roll-pitch controllers
iii. yaw controllers
iv. throttle controllers
v. navigation controllers

5.5.1 Rate Controllers

R**ate controllers** are always running no matter which of the various flight modes, roll-pitch, yaw or throttle controllers may or may not be active at the time.These controllers are responsible for providing a roll, pitch or yaw rate of change or a vertical acceleration in the case of the throttle rate controller.

The low level body-frame rate targets for these controllers are set by the upper controllers when they call the `set_roll_rate_target`, `set_pitch_rate_target`, `set_yaw_rate_target` or `set_throttle`.

These rate controllers are called at 100Hz from the main loop and it is implemented `inrun_rate_controllers()` (Attitude.pde)

5.5.2 Roll-Pitch, Yaw, Throttle controllers

Each of the roll-pitch, yaw and throttle controllers are responsible for:

i. Converting the pilot's input received from the tx/rx into a desired angle, altitude, etc.
ii. Calling the appropriate PID controllers to implement the desired angle, altitude, etc.
iii. pass a desired rate or acceleration to the rate controllers.

A particular roll-pitch, yaw or throttle controller is activated by calling the following functions which can all be found in ArduCopter.pde.

```
  i.   set_roll_pitch_mode()
 ii.   set_yaw_mode()
iii.   set_throttle_mode()
```

Similarly the following three functions are called regularly (100Hz or 50Hz) from the main loop to allow the controllers to read in the latest pilot input and call the appropriate controllers that are normally held in Attitude.pde:

```
  i.   update_roll_pitch_mode()
 ii.   update_yaw_mode()
iii.   update_throttle_mode()
```

5.5.3 Navigation controllers

The **navigation controllers** are most like roll-pitch controllers except that they work with lat/lon coordinates instead of roll/pitch angles or rates. Like the other controllers there is a `set_nav_mode()` function to activate the navigation controller and an `update_navigation` to actually implement the control.

Chapter 6

On Board Image Processing

6.1 Raspberry Pi Setup

6.1.1 Start up and Initialization

After booting the Raspberry Pi for the first time or after rebooting it the following information would be useful.

```
Login: pi
```

```
Password: raspberrypi
```

then type, `startx`

We can use the Pi in two ways:

i. With HDMI
ii. Without HDMI (using ssh)

`sudo dpkg-reconfigure keyboard-configuration` (US not UK version) to change the keyboard layout and then save the changes and type `sudo reboot.`

6.1.2 Raspberry Pi Camera and OpenCV installation

We have connected the camera module to the CSI port, located behind the Ethernet port, and enabled the camera software using the following steps:

i. Open the raspi-config tool from the Terminal using
   `sudo raspi-config`
ii. Select `Enable` camera and hit `Enter`, then go to `Finish` and you'll be prompted to reboot.
iii. `sudo apt-getupdate`
iv. `sudo apt-get upgrade`

6.1.2.1 Raspivid and Raspicam

raspivid is a command line application that allows you to capture video with the camera module, while the application *raspistill* allows you to capture images.

- -o or –output specifies the output filename.
- -t or –timeout specifies the amount of time that the preview will be displayed in milliseconds. Note that this set to 5s by default and that `raspistill` will capture the final frame of the preview period.
- -d or –demo runs the demo mode that will cycle through the various image effects that are available.

Command to capture image from Raspberry Pi is

```
raspistill -o image.jpg
```

Command to capture image from Raspberry Pi is

```
raspivid -o video.h264 -t 10000
```

To open the image captured from raspberry pi camera

```
xdg-open image.jpg
```

6.1.2.2Raspicam with OpenCV

OpenCVlibraries support only USB-cameras for capturing the images but we have used Raspberry Pi camera which uses CSI interface.So, we require special library function for capturing the images in OpenCV.

You might also need gcc/g++ in case it's not already installed, along with some other libraries.

```
i.    sudo apt-get install cmake git
ii.   sudo apt-get install gcc g++ libx11-dev libxt-dev
      libxext-dev libgraphicsmagick1-dev libcv-dev
      libhighgui-dev
iii.  mkdir -p ~/git/raspberrypi
iv.   cd ~/git/raspberrypi
v.    git clone https://github.com/raspberrypi/userland.git
vi.   cd userland
vii.  ./buildme
viii. mkdir -p ~/git
ix.   cd ~/git
x.    git clonehttps://github.com/robidouille/robidouille.git
xi.   cd robidouille/raspicam_cv
xii.  mkdir objs
xiii. make
```

This will create raspicamtest.o and raspiCamCV.h

6.1.3 Procedure for executing and compiling the OpenCV codes

`g++ -o <file_name><file_name.cpp> `pkg-config --static --cflags --libs opencv`` and append the following three statements to this command.

- `-I/home/pi/git/robidouille/raspicam_cv`
- `-L/home/pi/git/robidouille/raspicam_cv -lraspicamcv`
- `-L/home/pi/git/raspberrypi/userland/build/lib`
  `-lmmal_core -lmmal -l mmal_util -lvcos -lbcm_host`

We need to make following changes to OpenCV code

```
 #include "RaspiCamCV.h"
```

and replace

```
“cvCreateCameraCapture” with raspiCamCvCreateCameraCapture
“cvQueryFrame” with“raspiCamCvQueryFrame”
“CvCapture”with “RaspiCamCvCapture”
```

6.2 Video Streaming

We have used Logitech Webcam C260 for video streaming. We have connected USB camera with Raspberry Pi and executed following commands

i. Install build dependencies
   The following command installs the three libraries that MJPG-Streamer uses:

```
sudo apt-get install libjpeg8-dev imagemagick libv4l-dev
```

ii. Add missing videodev.h
   The videodev.h header file that MJPG-Streamer needs has been replaced with a videodev2.h. To make MJPG-Streamer happy you have to create a symbolic link:

```
sudo ln -s /usr/include/linux/videodev2.h usr/include/linux/videodev.h
```

iii. Download MJPG-Streamer
   The source code for MJPG-Streamer is available at sourceforge.net, but it is tricky to find the direct download link:

```
wget                http://sourceforge.net/code-snapshots/svn/m/mj/mjpg-streamer/code/mjpg-streamer-code-182.zip
```

iv. Unzip the MJPG-Streamer source code
   The source code download is a compressed zip file. Put the file in your home directory (or a temporary folder, if you prefer) and run the following to extract the files:

```
     unzip mjpg-streamer-code-182.zip
```

v. Build MJPG-Streamer

MJPG-Streamer comes with several plugins, but only a couple of them are needed to stream video according to the method I explained in my previous article. The command below only builds what is needed:

```
cd mjpg-streamer-code-182/mjpg-streamer
make mjpg_streamer input_file.so output_http.so
```

vi. Install MJPG-Streamer

The following commands copy all the needed files into system directories:

```
sudo cp mjpg_streamer /usr/local/bin
sudo cp output_http.so input_file.so /usr/local/lib/
sudo cp -R www /usr/local/www
```

vii. Start the camera

We are almost there. Now it is time to start the camera module:

```
mkdir /tmp/stream
raspistill --nopreview -w 640 -h 480 -q 5 -o /tmp/stream/pic.jpg -tl 100 -t 9999999 -th 0:0:0
```

viii. Start MJPG-Streamer

The camera is now writing images, so all that is left is to start MJPG-Streamer:

```
LD_LIBRARY_PATH=/usr/local/lib mjpg_streamer -i "input_file.so -f /tmp/stream -n pic.jpg" -o "output_http.so -w /usr/local/www"
```

Now you can connect with your web browser and watch the stream live. If you want to watch from within the same Raspberry Pi you can enter http://localhost:8080 in the browser's address bar. If you want to watch from another computer in your network use http://<IP-address>:port_number.

6.3 Algorithms implemented on Raspberry Pi

Object detection and segmentation is the most important and challenging fundamental task of computer vision.
The easiest way to detect and segment an object from an image is the color based methods. The object and the background should have a significant color difference in order to successfully segment objects using color based methods.
Our door detection algorithm are implemented in c in opencv. We are using contours and thresholding is done in two different ways.

6.3.1 Ostu's Thresholding

Algorithm for

i. Compute histogram and probabilities of each intensity level
ii. Set up initial $\omega i(0)$ and $\mu i(0)$
iii. Step through all possible thresholds t=1 …… maximum intensity

- Update ωi and μi
- Compute σ_b^2(t)

iv. Desired threshold corresponds to the maximum σ_b^2(t)

v. You can compute two maxima (and two corresponding thresholds) . σ_{b1}^2(t) is the greater max and σ_{b1}^2(t) is the greater or equal maximum

vi. 6.Desired threshold = (threshold1 + threshold2)/2

We have following command in OpenCV for Ostu's thresholding.

```
cvThreshold(imgGrayScale,imgGrayScale,0,255,THRESH_OTSU |
THRESH_BINARY_INV);
```

6.3.2 HSV Thresholding

We have detected a red color object and created a binary image by thresholding the red color. Red color area of the video is assigned to '1' and other area is assigned to '0' in the binary image so that you will see a white patch wherever the red object is in the original video. We have used following opencv command for HSV thresholding.

```
cvInRangeS(img_HSV, cvScalar(iLowH, iLowS, iLowV),
cvScalar(iHighH, iHighS, iHighV), imgThresholded);
```

For our red object marker, the values of the H ~ 161-164, S ~ 74 ,V ~57

After thresholding the image, you'll see small white isolated objects here and there. It may be because of noises in the image or the actual small objects which have the same color as our main object. These unnecessary small white patches can be eliminated by applying morphological opening. Morphological opening can be achieved by a erosion, followed by the dilation with the same structuring element.We have used elliptical structuring element. Thresholded image may also have small white holes in the main objects here and there. It may be because of noises in the image. These unnecessary small holes in the main object can be eliminated by applying morphological closing. Morphological closing can be achieved by a dilation, followed by the erosion with the same structuring element.

6.3.3 Contours in OpenCV

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

```
cvFindContours(imgThresholded, storage, &contours,
sizeof(CvContour),CV_RETR_LIST,CV_CHAIN_APPROX_SIMPLE,cvPoint(0,0));
```

To draw the contours `cvFindContours( )` function is used. It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass -1) and remaining arguments are color, thickness etc.

Chapter 7

Communication Protocol: MAVlink

7.1 Introduction

It is a protocol for communication between a Ground Control Station (GCS) and unmanned vehicles, and in the inter-communication of the subsystem of the vehicle. It can be used to transmit the orientation of the vehicle, its GPS location and speed. The MAVlink message (we call it 'msg') is basically a stream of bytes that has been encoded by GCS (Ground Station Control) and is sent to the APM via USB serial or telemetry (both cannot be used at the same time. If they are plugged in at the same time, USB is given preference and Telemetry is ignored).

Each MavLink packet has a length of 17 bytes and here is the structure:

Message length = 17 (6 bytes header + 9 bytes payload + 2 bytes checksum)
6 bytes header
0. message header, always 0xFE
1. message length (9)
2. sequence number -- rolls around from 255 to 0 (0x4e, previous was 0x4d)
3. System ID - what system is sending this message (1)
4. Component ID- what component of the system is sending the message (1)
5. Message ID (e.g. 0 = heartbeat and many more! Don't be shy, you can add too)
Variable Sized Payload (specified in octet 1, range 0-255)
** Payload (the actual data we are interested in); Checksum: For error detection
Common interface for all MAVLink Messages
This is the anatomy of one packet. It is inspired by the CAN and SAE AS-4 standards.

The checksum is the same as used in ITU X.25 and SAE AS-4 standards (CRC-16-CCITT), documented in SAE AS5669A. Please see the MAVLink source code for a documented C-implementation of it. LINK TO CHECKSUM

The minimum packet length is 8 bytes for acknowledgement packets without payload. The maximum packet length is 263 bytes for full payload.

gcs0 is via USB and gsc3 is via Telemetry

Table 7.1: Structure of a message
Courtesy:Mavlink for Dummies Part 1

Byte	Index	Content	Value	Explanation
0		Packet start sign	v1.0: 0xFE	Indicates the start of a new packet. (v0.9: 0x55)
1		Payload length	0 - 255	Indicates length of the following payload.
2		Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3		System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network
4		Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5		Message ID	0 - 255	ID of the message - the id defines what the payload means and how it should be correctly decoded.
6 to (n+6)		Payload	0 - 255	Data of the message, depends on the message id.
(n+7)to(n+8)		Checksum (low byte, high byte)		ITU X.25/SAE AS-4 hash, excluding packet start sign, so bytes 1..(n+6) Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).

7.2 GCS to Quadcopter

Each message (we called a packet, which has the useful information for us), has a message ID and payload put into a data structure of ‘appropriate type’. We switch on the ‘main message’ (or MAVLINK_MSG_ID) type and once that message is detected, we store the Information received to permanent memory (called EEPROM).

```
MAVLINK_MSG_ID_HEARTBEAT: //0
MAVLINK_MSG_ID_REQUEST_DATA_STREAM:  //66
MAVLINK_MSG_ID_COMMAND_LONG: // 76
SET_MODE: //11
MAVLINK_MSG_ID_MISSION_REQUEST_LIST: //43
MAVLINK_MSG_ID_MISSION_REQUEST: //40
MAVLINK_MSG_ID_MISSION_ACK: //47
MAVLINK_MSG_ID_PARAM_REQUEST_LIST: //21
```

```
MAVLINK_MSG_ID_PARAM_REQUEST_READ: //20
MAVLINK_MSG_ID_MISSION_CLEAR_ALL: //45
MAVLINK_MSG_ID_MISSION_SET_CURRENT: //41
MAVLINK_MSG_ID_MISSION_COUNT: // 44
MAVLINK_MSG_ID_MISSION_WRITE_PARTIAL_LIST: //
MAVLINK_MSG_ID_SET_MAG_OFFSETS: //151
MAVLINK_MSG_ID_SET_MAG_OFFSETS: //151
MAVLINK_MSG_ID_PARAM_SET: //23
MAVLINK_MSG_ID_RC_CHANNELS_OVERRIDE: //70
MAVLINK_MSG_ID_HIL_STATE: //90
```

7.2 Quadcopter to Ground Station Control (GCS) to Quadcopter

GCS is only mediator between you and your copter. It gets the data from copter in return, displays on the GCS.

```
        . . .
        . . .
        { gcs_send_heartbeat, 100, 150 },
        { update_notify, 2, 100 },
        { one_hz_loop, 100, 420 },
        { gcs_check_input,    2, 550 },
        { gcs_send_heartbeat, 100, 150 },
        { gcs_send_deferred, 2, 720 },
        { gcs_data_stream_send, 2, 950 },
        . . .
        . . .
        . . .
```

The first parameter is the **function name**, the second is the '**time it is supposed to take'** (in 10ms units, i.e., 2 means, 20ms, i.e. 50Hz, i.e. this function runs 50 times a second). The third parameter is the '**max time beyond which the function should not run**'.
This helps in sending data down a link (gcs0 is via USB and gcs3 is via Telemetry). Then the data is sent back to GCS by Quadcopter in form of the data structures for the GCS to display.
Ground Control Station is a ground station running on Android tablets, it can be used to customize parameters

Chapter 8

GCS Overview

8.1 Introduction

Garuda Control Station is a ground station running on Android tablets, it can be used to customize parameters or prepare a flying mission with waypoints when a quadcopter is using the APM ported firmware.

8.2 Operation Manual

GCS connects to the APM board on the drone via Wi-Fi or USB. Let's do a quick explanation of how to do that:

8.2.1 TCP

A Raspberry PI board with a TCP to serial redirection software

Figure 8.1: Raspberry Pi
Courtesy: https://github.com/Droidplanner/droidplanner/wiki/How-to-connect

A Wi-Fi router to be able to connect wireless via Aakash Tablet.

Figure 8.2: Wi-Fi Router

Courtesy: https://github.com/Droidplanner/droidplanner/wiki/How-to-connect

8.2.2 USB

To connect via USB you will need a USB On-the-Go cable and a 3DR telemetry module (for e.g.: 3DR Radio USB "Ground" module)
Connect the 3DR module to the Android device via this cable and **GCS** should pop up. From there you just need to hit "Connect".

.

8.3 Operating Instructions

i. As soon as **GCS** app is connected to the Quadcopter via any of the before mentioned procedures, the app sends control signals to robot and receive the sensors data and other required information.
ii. Now you need to arm the quadcopter using the ARM option present in option menu. A voice will confirm whether the vehicle has been armed or not. If not check your connections and try again.
iii. Once the quadcopter is armed. You are ready to take flight.
iv. Switch to either of the activities named RC or Camera & RC from the navigation spinner
v. Switch on the RC control switch
vi. Touch anywhere on the left side of the empty space to hold the left joystick and touch anywhere on the right side of the empty space to hold the right joystick.
vii. While controlling in the RC activity, if you wish to see the flight data, just change the orientation of Aakash to portrait.

Here is a picture of the complete base station system

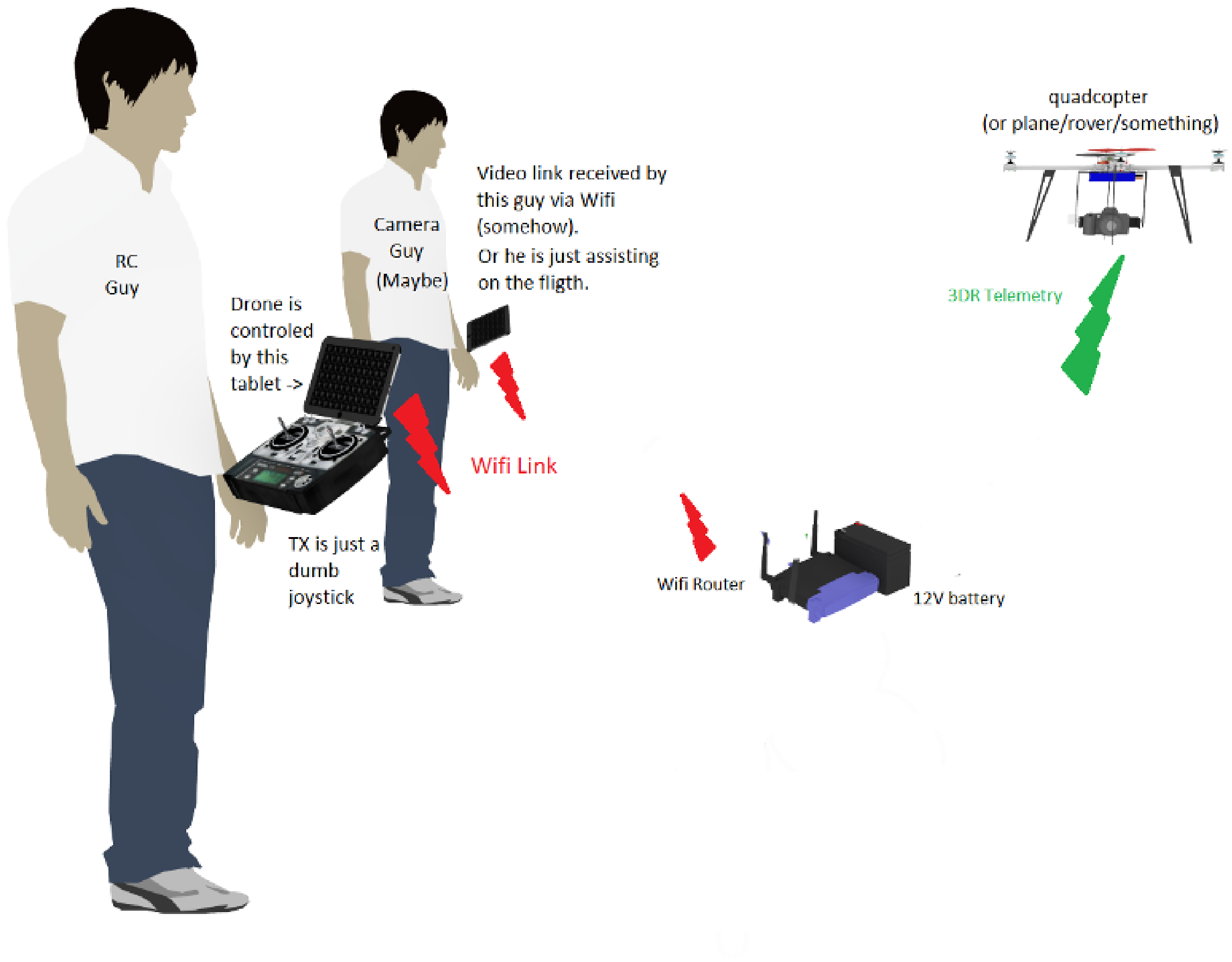

RC
Guy
Camera
Guy
(Maybe)
Video link received by this guy via Wifi (somehow).
Or he is just assisting on the fligth.
quadcopter
(or plane/rover/something)
3DR Telemetry
Drone is controled by this tablet ->
Wifi Link
TX is just a dumb joystick
Wifi Router
12V battery

Figure 8.3 Base station system

Courtesy : https://github.com/Droidplanner/droidplanner/wiki/How-to-connect

8.4 GCS Development Information

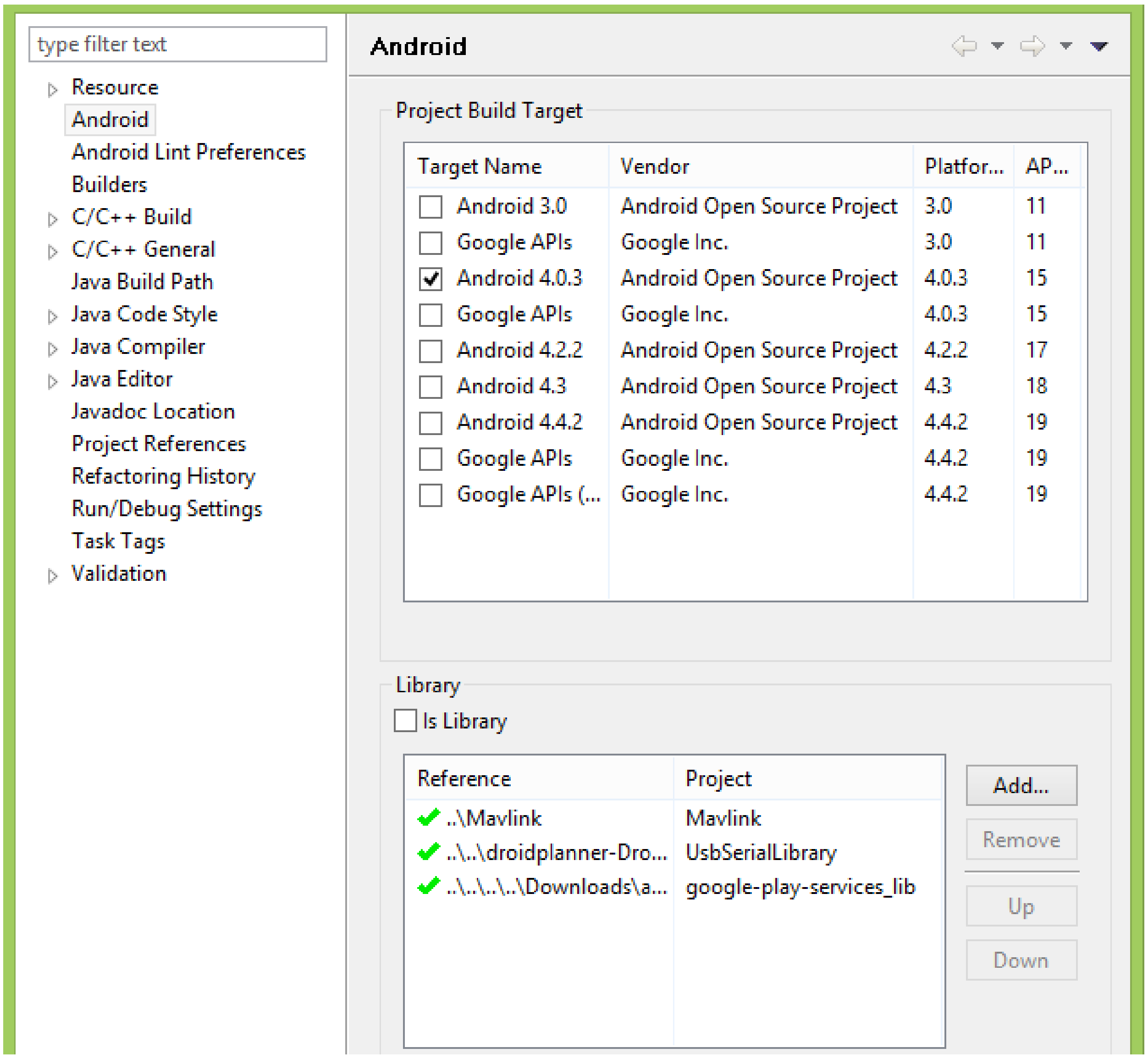
The GCS app is developed using ADT (Android Developer Tools) plugin in Eclipse IDE for Android 4.0.3 API level 15

8.4.1 LIBRARIES USED:

i. Google play services: Required for running google APIs such as google maps.

ii. USB Serial Library: This is a driver library for communication with USB serial hardware on Android, using the Android USB Host API available on Android 3.1+.

iii. Mavlink: This library allows communication between app and quadcopter via mavlink protocol.



Android

Project Build Target

	Target Name	Vendor	Platfor...	AP...
☐	Android 3.0	Android Open Source Project	3.0	11
☐	Google APIs	Google Inc.	3.0	11
☑	Android 4.0.3	Android Open Source Project	4.0.3	15
☐	Google APIs	Google Inc.	4.0.3	15
☐	Android 4.2.2	Android Open Source Project	4.2.2	17
☐	Android 4.3	Android Open Source Project	4.3	18
☐	Android 4.4.2	Android Open Source Project	4.4.2	19
☐	Google APIs	Google Inc.	4.4.2	19
☐	Google APIs (...	Google Inc.	4.4.2	19

Library

☐ Is Library

Reference	Project
✓ ..\Mavlink	Mavlink
✓ ..\..\droidplanner-Dro...	UsbSerialLibrary
✓ ..\..\..\..\Downloads\a...	google-play-services_lib

Figure 8.4: Project Properties

8.4.2 Coding

The entire coding of the app is divided into several stages:
Stage 1: Coding the connection files to establish connection using Mavlink protocol via TCP and USB. This can be found in MavLinkConnection.java, TcpConnection.java and UsbConnection.java.
Stage 2: Make a basic layout or a template that will be extended by all the activities. This is what we call SuperActivity, code in SuperActivity.java
Stage 3: Design and code each of the Activity in the app which are :-

- Flight Data
- RC
- Parameters
- GCP
- Chart
- Camera Feed
- Map

- Camera & RC

The functionality and coding of each activity is explained later in the section Activities.

Most activities use fragments such as RCFragment, HUDFragment, etc which has to be coded along with the corresponding activity.

Stage 4: Adding the Required permissions in the manifest file along with rechecking other parameters such as app name, version etc.

8.4.3 Coding

The Aakash Tablet and the APM board are wirelessly connected by a WI-FI Router. Any Interaction with the tablet is done through Mavlink protocol. The tablet sends and receives Mavlink packets via any of the Telemetry connection types – TCP or USB.

TCP: When using TCP telemetry connection, the server IP is converted to InetAdress which along with the server port is used to create a java.net.socket. This Socket is capable of reading and writing byte-oriented data using buffer.

USB: When using USB telemetry connection, an usb serial driver is created using a USB com port and given baud rate. This driver is capable of reading and writing byte-oriented data using buffer.

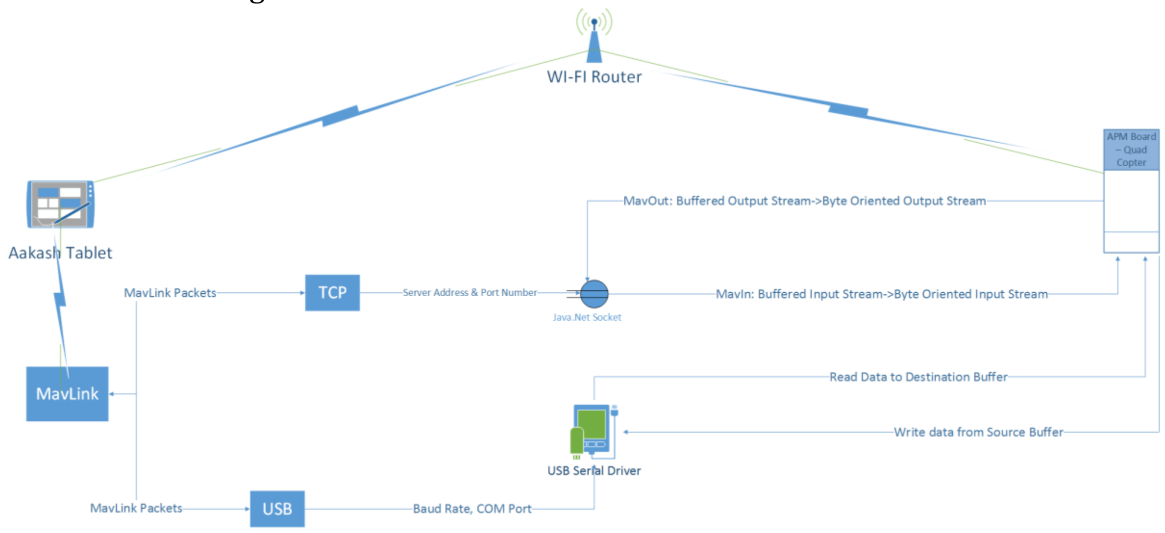

WI-FI Router
APM Board – Quad Copter
Aakash Tablet
MavOut: Buffered Output Stream->Byte Oriented Output Stream
MavLink Packets
TCP
Server Address & Port Number
Java.Net Socket
MavIn: Buffered Input Stream->Byte Oriented Input Stream
MavLink
Read Data to Destination Buffer
Write data from Source Buffer
USB Serial Driver
MavLink Packets
USB
Baud Rate, COM Port

Figure 8.5: Data Flow Diagram

Chapter 9

Interfaces

9.1 Introduction

The various interfaces that the app contains are:

1. RC
2. Flight Data
3. Planning
4. Parameters
5. GCP
6. Chart
7. Camera Feeds
8. Camera and RC

9.2 Interface 1: RC

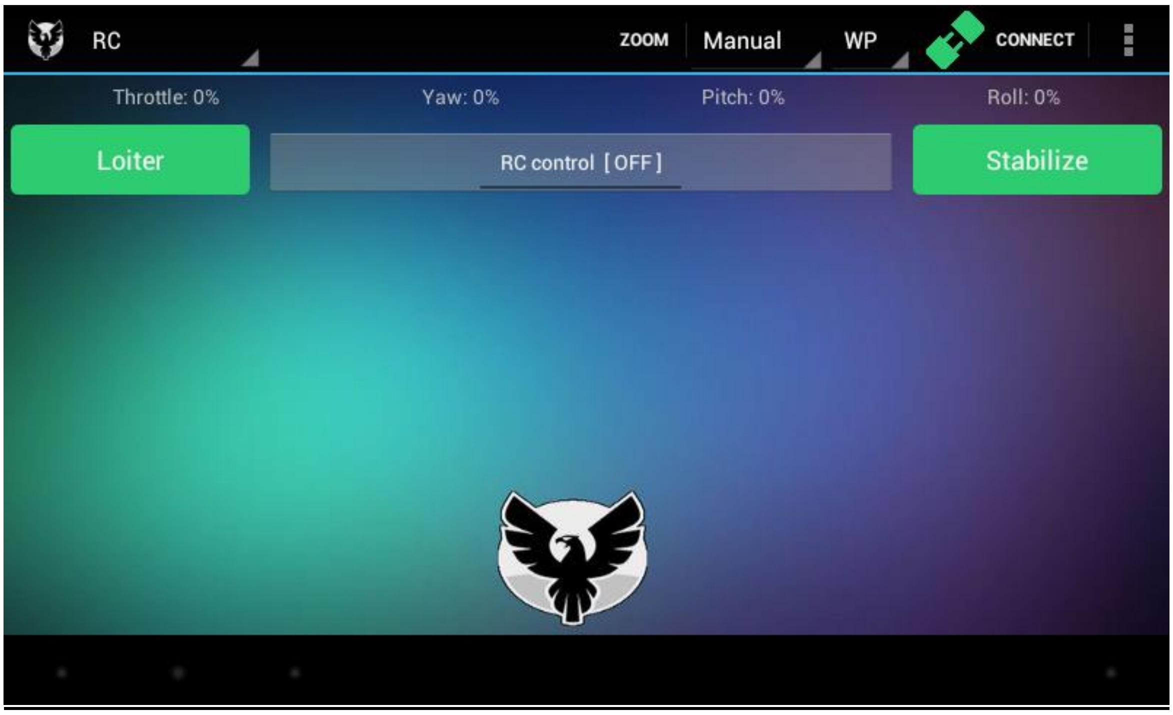


Figure 9.1: RC View

This interface enables you to control the quadcopter using the virtual Radio Control in your Aakash Tablet. The RC view is created in a file called **JoyStickView.java**[1] Using the MotionEventobject, we report movement (trackball) event. These Motion events may hold either absolute or relative movements and other data, depending on the type of device.

These are the setting up of indexes for the various channels

```
        publicstaticfinalintAILERON = 0;
```

```
publicstaticfinalintELEVATOR = 1;
publicstaticfinalintTROTTLE = 2;
publicstaticfinalintRUDDER = 3;

publicstaticfinalintRC5 = 4;
publicstaticfinalintRC6 = 5;
publicstaticfinalintRC7 = 6;
publicstaticfinalintRC8 = 7;
```

Then we send the values to the MavLinkService so that these values can be sent to the APM through a message.

```
        Arrays.fill(rcOutputs, DISABLE_OVERRIDE); // Start with all
channels disabled, external callers can enable them as desired
        MavLinkRC.sendRcOverrideMsg(drone, rcOutputs);
```

These values are sent to the MavLinkRC.java which using the MavLink library sends the values to the APM through the message

```
msg.chan1_raw = (short) rcOutputs[0];
msg.chan2_raw = (short) rcOutputs[1];
msg.chan3_raw = (short) rcOutputs[2];
msg.chan4_raw = (short) rcOutputs[3];
msg.chan5_raw = (short) rcOutputs[4];
msg.chan6_raw = (short) rcOutputs[5];
msg.chan7_raw = (short) rcOutputs[6];
msg.chan8_raw = (short) rcOutputs[7];
msg.target_system = 1;
msg.target_component = 1;
drone.MavClient.sendMavPacket(msg.pack());
```

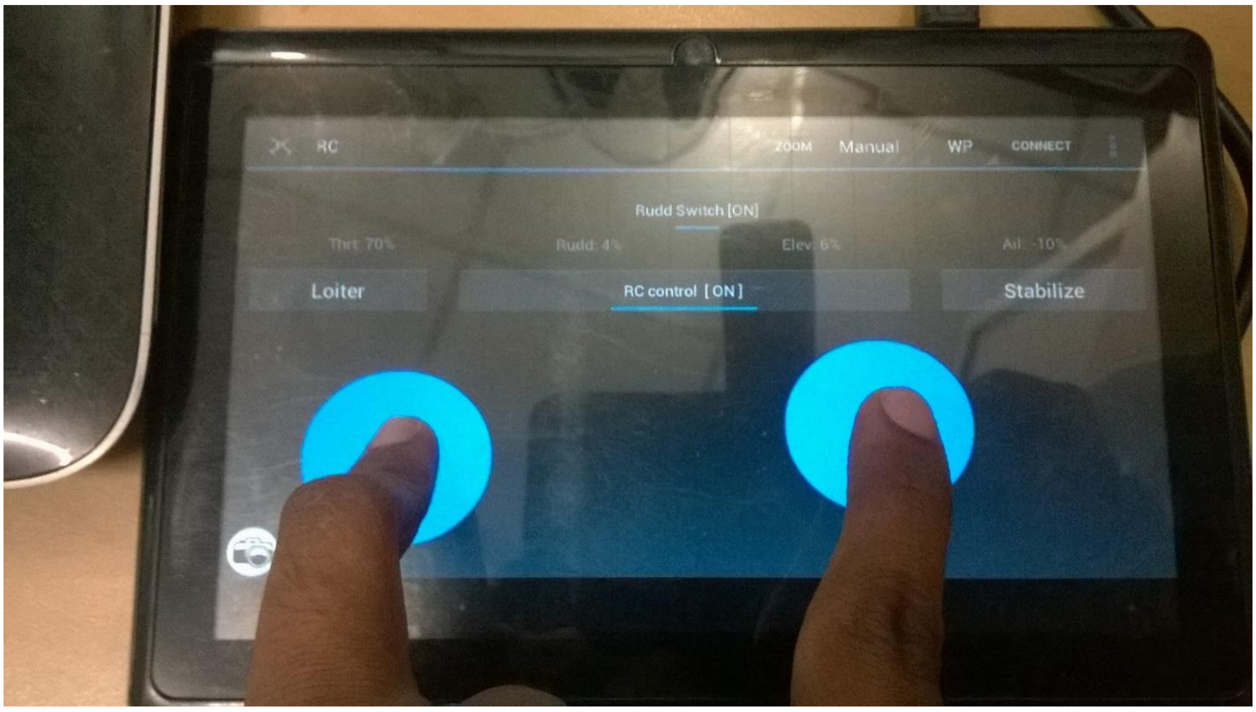


Figure 9.2: RC in action

9.2 Interface 2: Planning

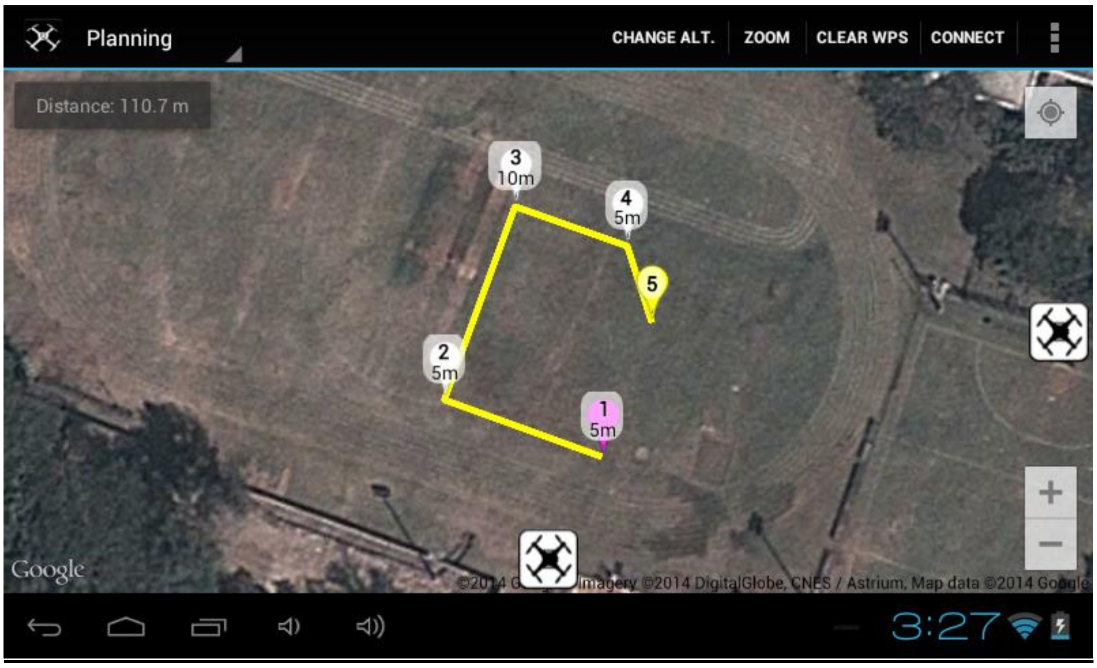


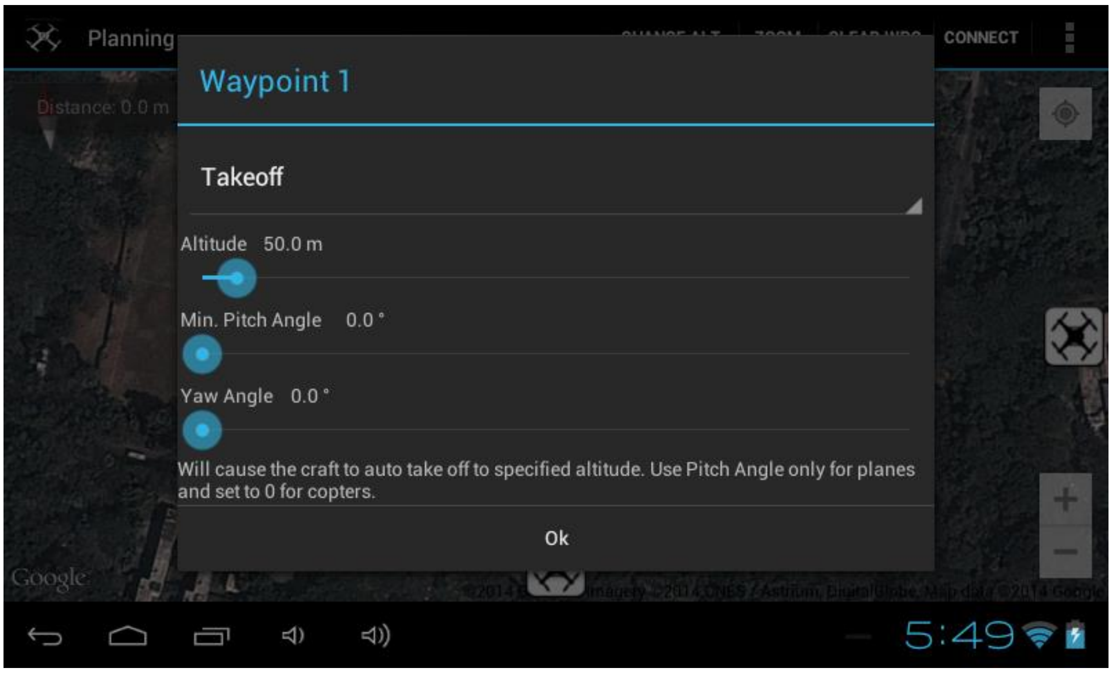
Figure 9.1: Planning View

This interface enables you to set waypoints for the quadcopter. Using the Google Maps, one can set different waypoints and then set the properties of those waypoints such as Landing, Loiter, Take-off etc.

Maps are being integrated in the application using the Google Maps API v2.

Some of the permissions that have been given to the API.

```
<uses-permissionandroid:name="android.permission.INTERNET"/>
<uses-permissionandroid:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- The following two permissions are not required to use
    Google Maps Android API v2, but are recommended. -->
<uses-permissionandroid:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permissionandroid:name="android.permission.ACCESS_FINE_LOCATION"/>
```



Altitude 50.0 m

Min. Pitch Angle 0.0 °

Yaw Angle 0.0 °

Will cause the craft to auto take off to specified altitude. Use Pitch Angle only for planes and set to 0 for copters.

Figure 9.4: Setting up of a WP

9.3 Interface 3: Flight Data

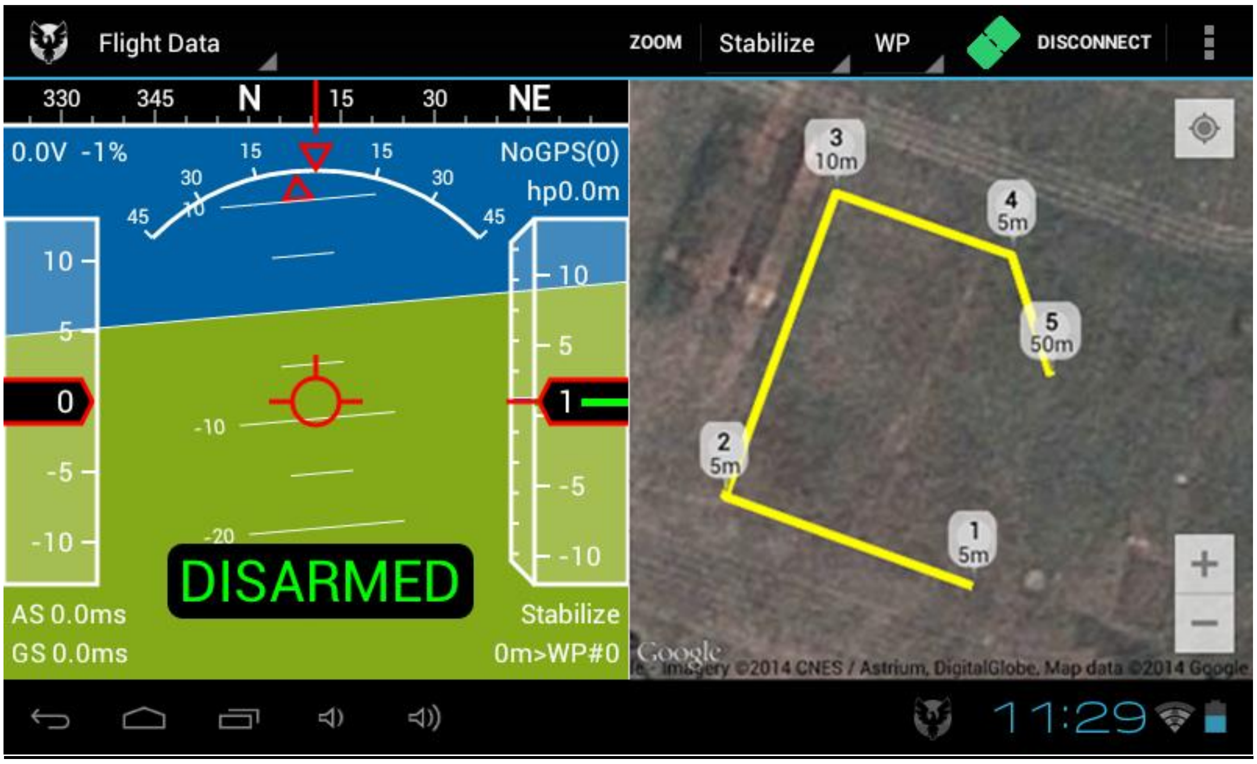


Figure 9.5: Flight Data with HUD

This interface enables the user to track the various flight information like the Altitude, Yaw angle , Pitch angle . This interface is divided in 2 halves, one Half gives the information of the flight and the other keeps the track of the waypoints covered.

9.4 Interface 4: Camera Feeds

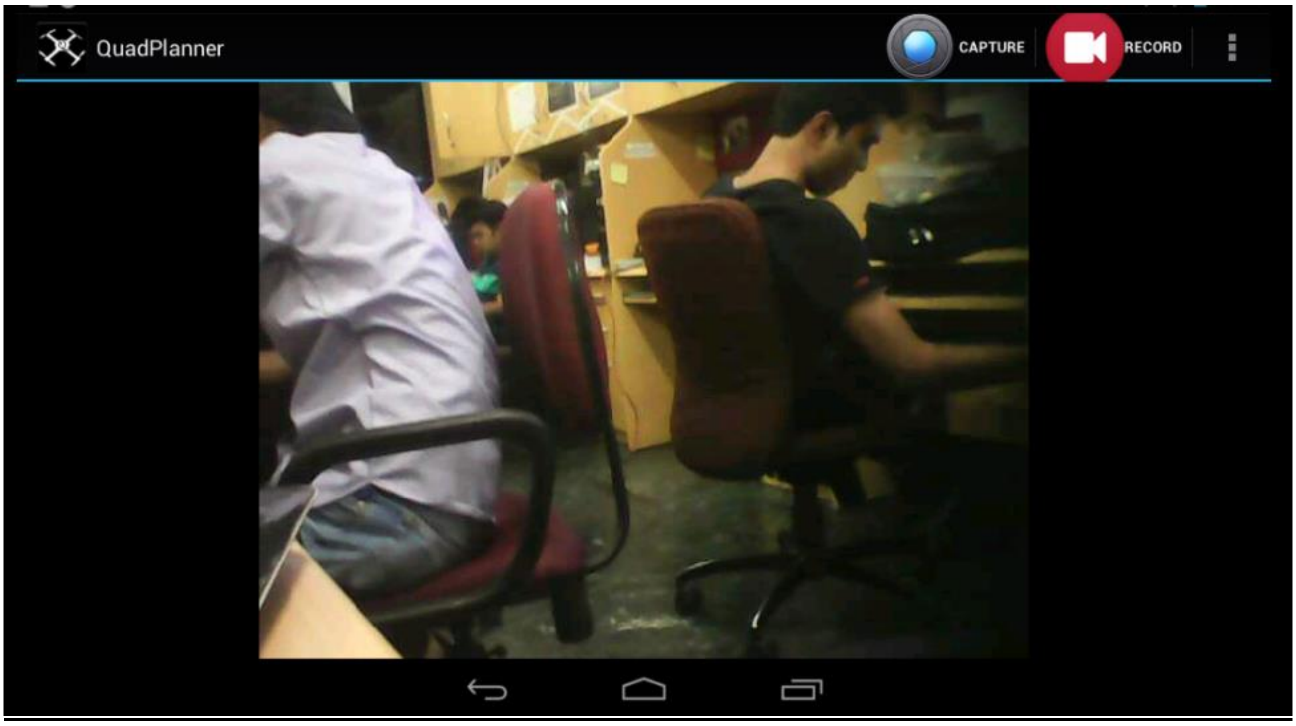


Figure 9.6: Camera Feeds

This interface shows the camera feeds sent by the IP Camera[2]. It has an option of setting up the IP and the port no. from where it captures the feed.This enables the user to open mjpeg video stream in the app and watch it. Mostly mjpeg is used for streaming images from cameras placed all over the world presenting nice or interesting places to website visitors. Also most security cameras use mjpeg. Our Android app can get the stream mjpeg with a lot of webcam.

Please note: Mjpeg video streams cannot be paused, rewinded or skipped forward. They are always shown as send by the server without further user interaction possible.

The User has the feature to capture Stills from the Video Feed. The user has an option to change the destination folder where the captured images are getting saved. Using the data of the Shared Preferences in Setting Activity , a URL is build and then is executed.

```
DoRead().execute(URL);
```

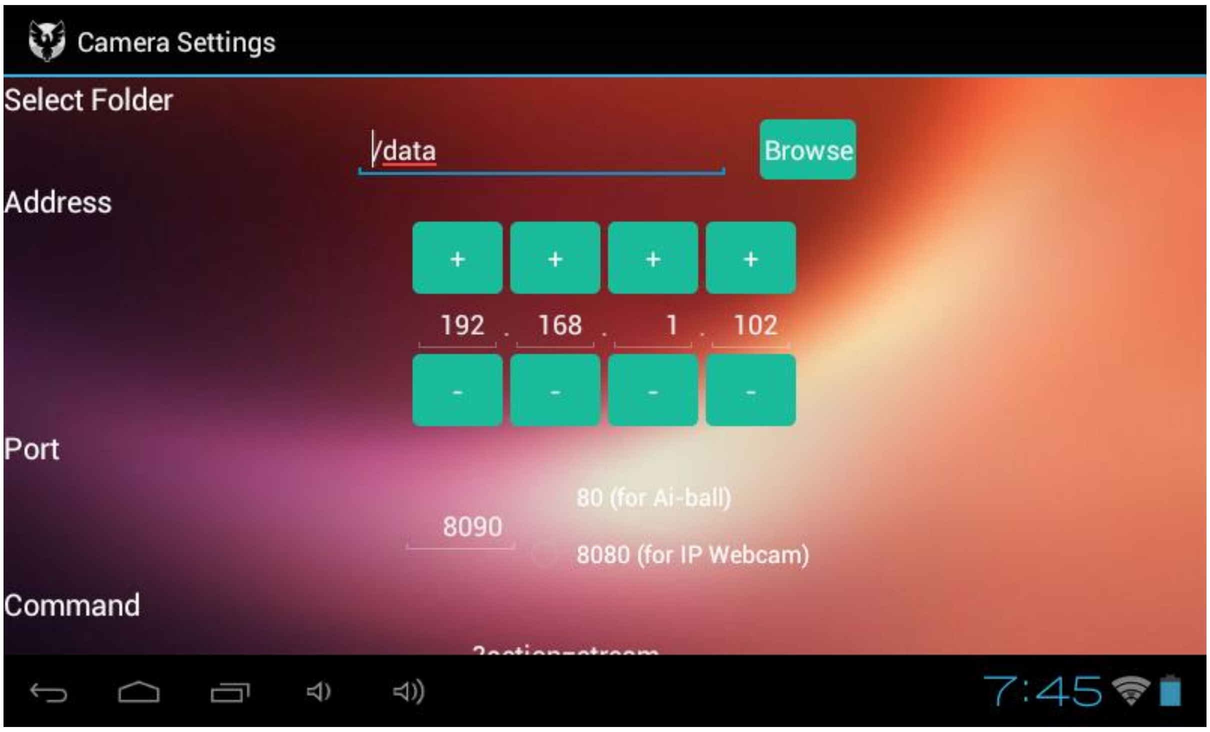

Camera Settings
Select Folder: /data
Address: 192 . 168 . 1 . 102
Port: 8090
80 (for Ai-ball)
8080 (for IP Webcam)
Command

Figure 9.7: Camera Settings

9.5 Interface 5: GCP

It is a tool to help with the placement of Ground Control Points (visual markers placed on the field for aerial photogrammetry purposes).

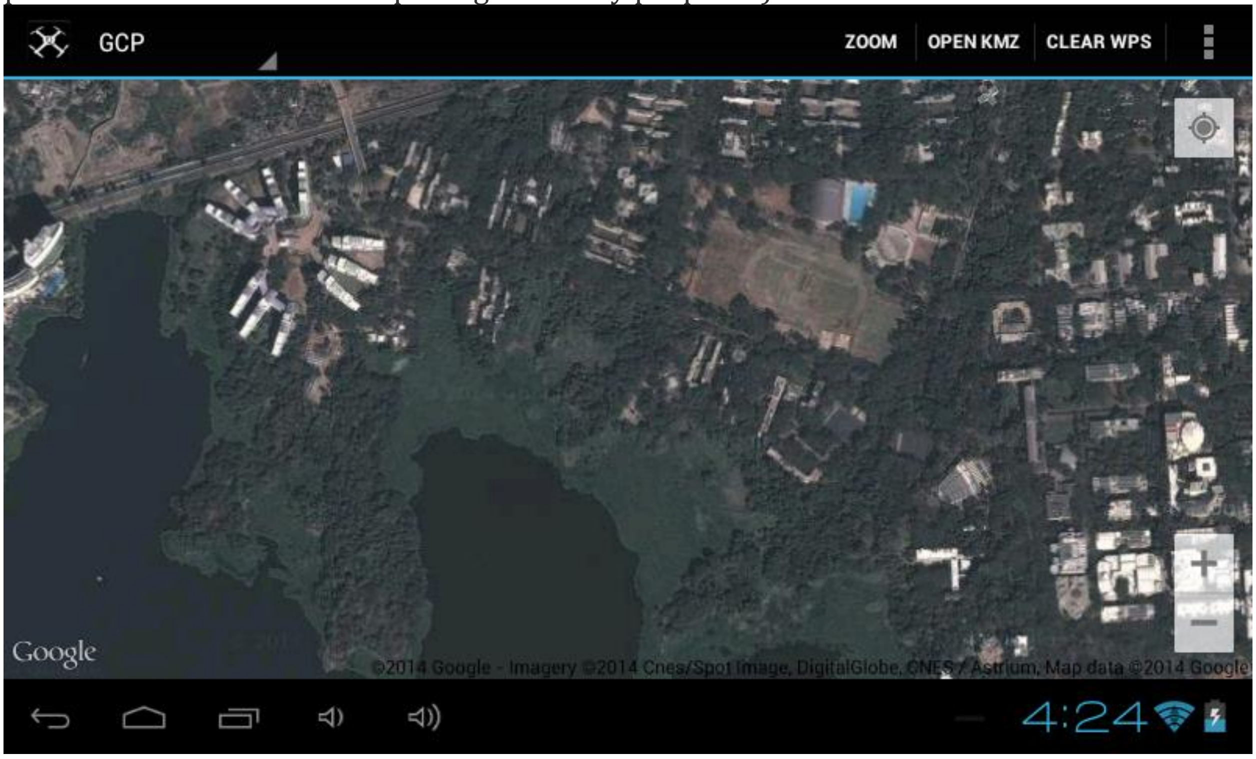


Figure 9.8: GCP View

The operation is simple:

i. Load a KMZ or KML file from Google Earth with Placemarks where the control points should be.
ii. Lockup your location at the blue dot.
iii. Go to a GCP location at a Red dot
iv. Drop a GCP marker on the ground
v. Click on the red dot of the GCP to turn it blue, to show that that place has been marked.
vi. If there are more red GCP then go back to step 2.
vii. To use your own KML or KMZ file you must save them into your Android device storage, under the folder "/QuadPlanner/GCP/". The files can be created with Google earth, using just standard markers.

9.6 Interface 6: Charts

This interface plots all the sensor data with respect to time.
Plots being plotted are:

i. Pitch
ii. Yaw
iii. Roll
iv. Altitude
v. Longitude Latitude

9.6.1 Pitch

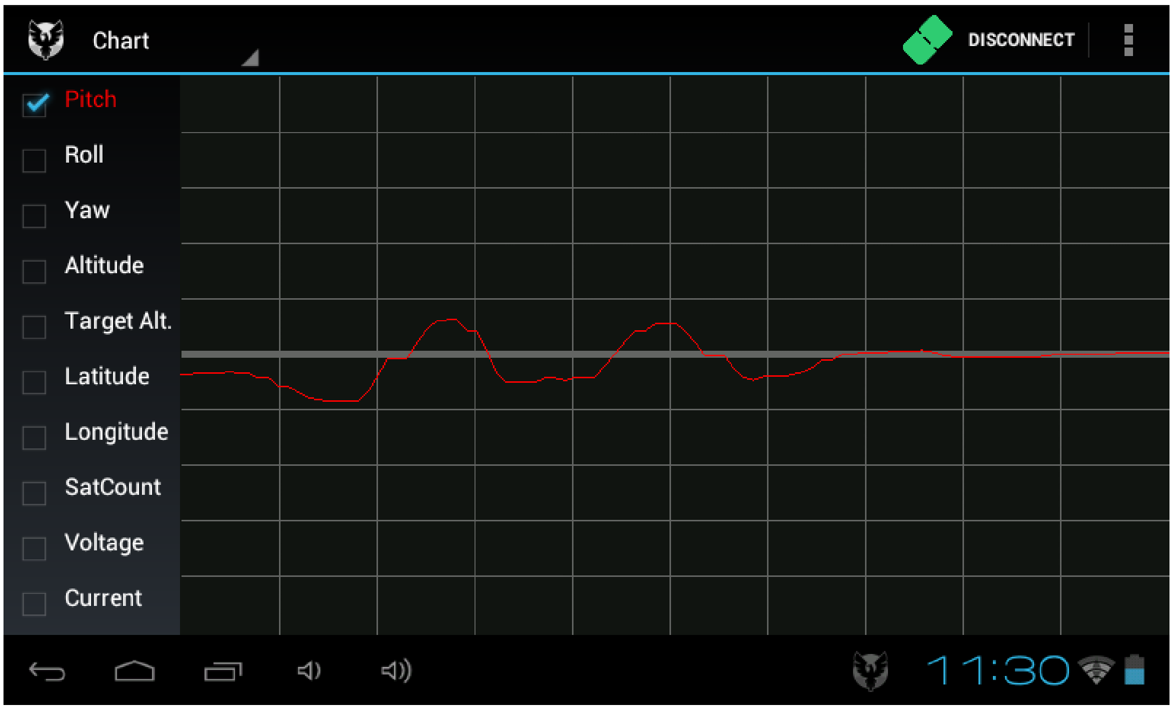


Figure 9.9: Pitch Chartplots the Pitch angle versus time.

9.6.2 Roll

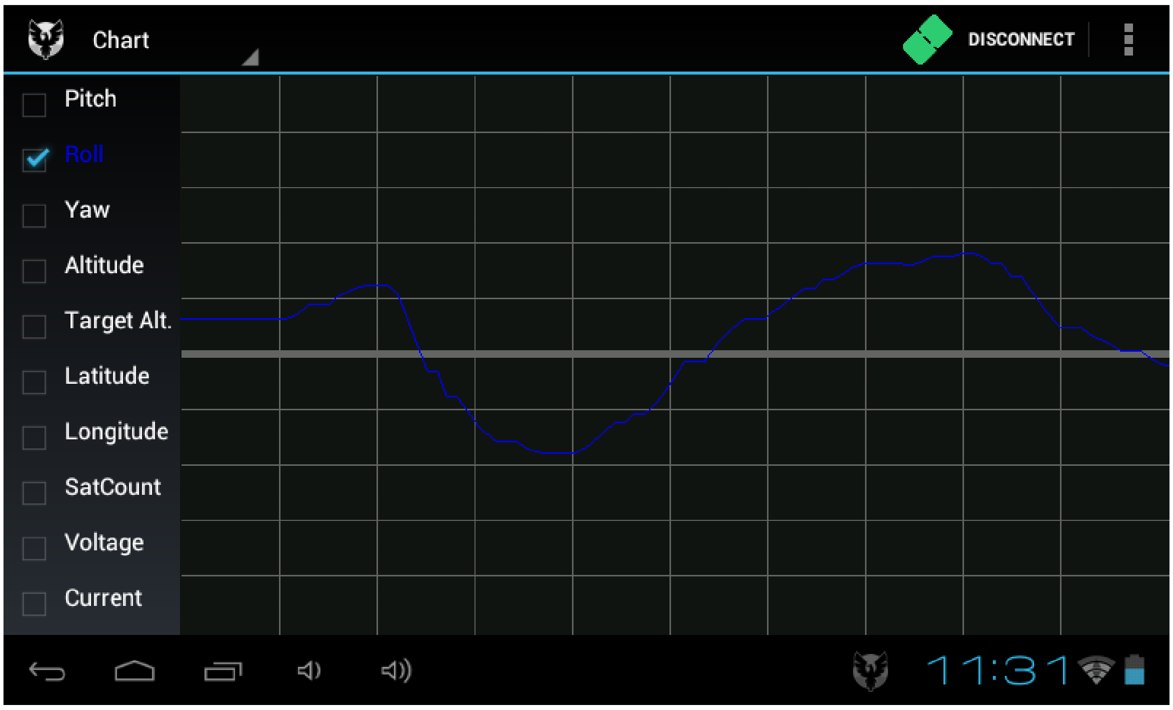


Figure 9.10: Roll chartplots the Roll angle versus time.

9.6.2 Yaw

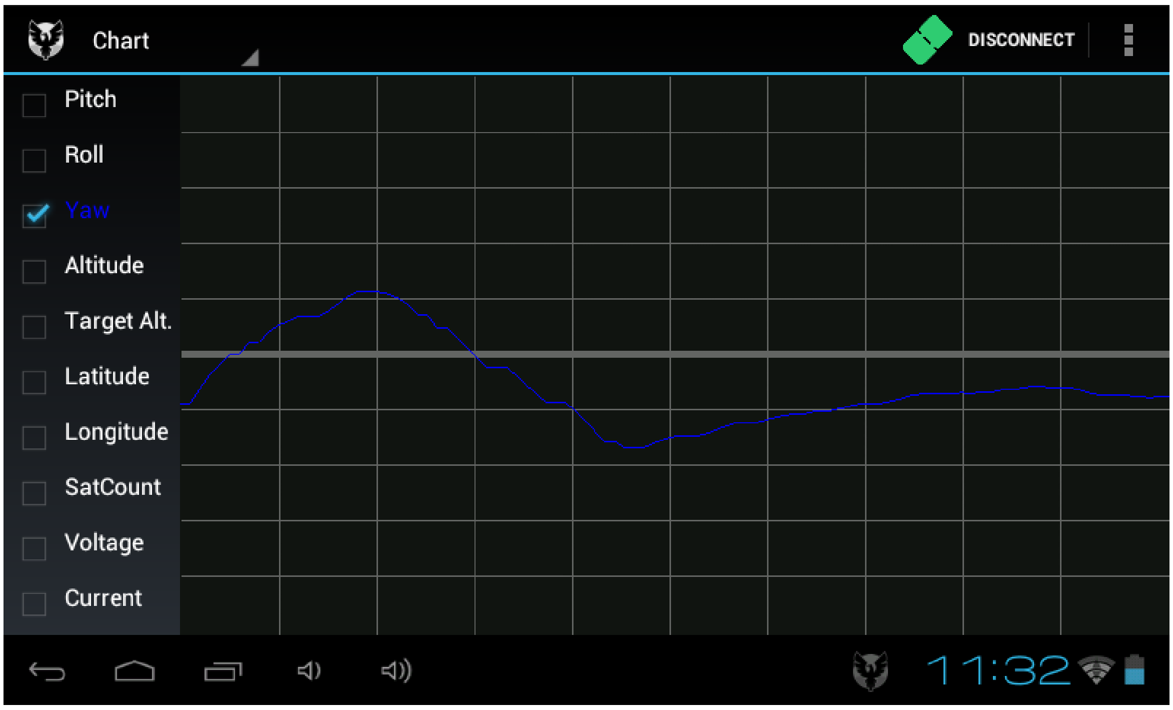


Figure 9.11: Yaw Chartplots the Yaw angle versus time

A plot that we got for a test fly of our quadcopter.

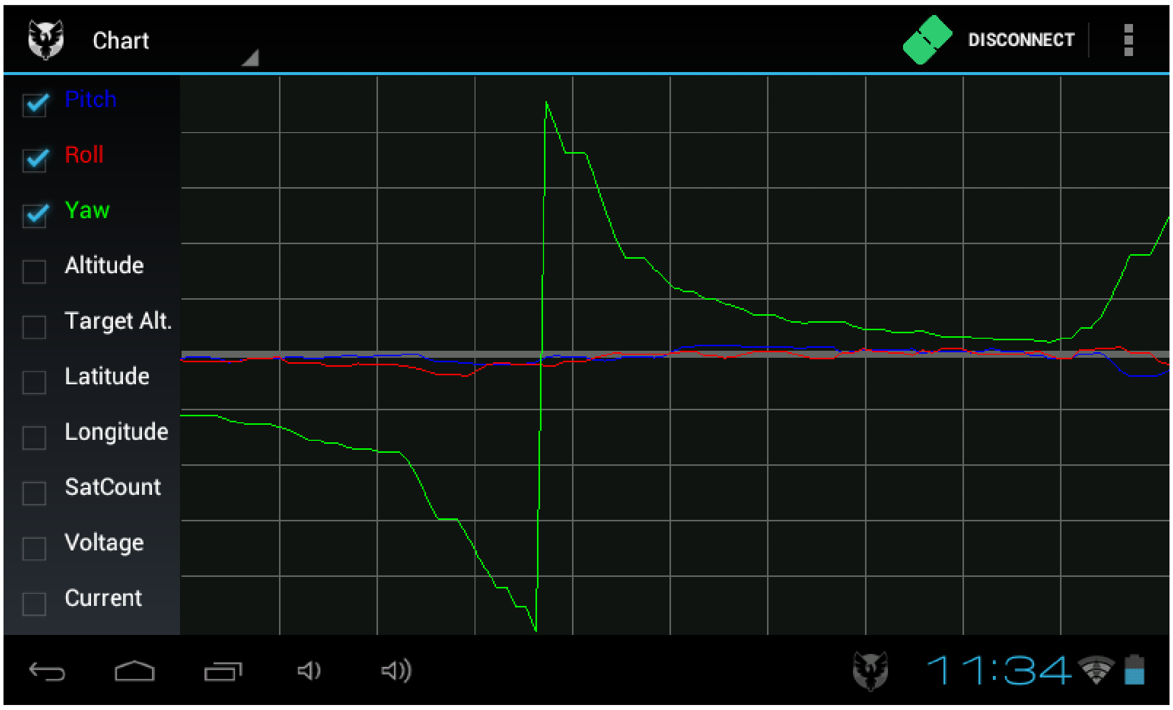


Figure 9.12: Test Plots

Conclusion and Future Work

The Garuda Quadcopter has been successfully tested both with the Radio Control and the Aakash Tablet. The sensor data is transmitted from the APM Board to the Aakash Tablet. An Android application namedGaruda Control Station has been successfully installed on the Aakash tablet for the control of the quadcopter, for stabilization of the quadcopter using PID Tuning and for retrieving the sensor data and displaying on the tablet. Live Video Streaming is being done using the USB Camera mounted on the quadcopter. Videos are streamed to the tablet in MJPEG format. It also has a feature to capture any image at a point of time.

1. At parking spaces for cars we can send the quadcopter out to fly down the car line and allow an operator to watch the onboard camera to see the permits.

2.Connecting ultrasonic sensors to the quadcopter,so that it can create a rough 3D map of the room and can avoid obstacles.With existing on board image processing and obstacle detection it can move autonomously inside any building.

3.Face Recognition can be implemented in the feeds received at the Aakash end to detect particular people’s face.

References

[1] https://github.com/AngusP/SkiTrack/blob/master/documentation/MS5611-01BA03%20datasheet.pdf

[2] http://www.invensense.com/mems/gyro/.../PS-MPU-6000A-00v3.4.pdf

[3] http://www.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC

[4] JoyStickView : https://code.google.com/p/mobile-anarchy-widgets/

[5] MjpegStream : https://bitbucket.org/neuralassembly/simplewebcam

[6] HUD : https://code.google.com/p/copter-gcs

[7] MAVlink : Mavlink for Dummies Part 1

[8] Barton, Jeffrey D. "Fundamentals of small unmanned aircraft flight." *Johns Hopkins APL technical digest* 31, no. 2 (2012): 132-149.

[9] Premerlani, William, and Paul Bizard. "Direction cosine matrix imu: Theory."*DIY Drones.[Online][Cited: 1 7 2012.] http://diydrones. ning. com/profiles/blogs/dcm-imu-theory-first-draft* (2009).

[10] http://dev.ardupilot.com/