Autonomous Quadcopter

**CMPE 295A**

**Project Workbook**

**By**

**Bhushan Gopala Reddy**

**Karthik Govindswamy**

**Rishikesh Nagare**

**Sujith Durgad**

**October 2015**

**Advisor: Preetpal Kang**

# Table of Contents

[Chapter 1. Literature Search, State of the Art](#h.d72ng8f7q8st)

[1.1 Literature Search](#h.wy38njz3me3w)

[1.2 State-of-the-Art](#h.erqhjtj2px4e)

[Chapter 2. Project Justification](#h.yqbbk1rdvb4d)

[Chapter 3. Project Requirements](#h.jbvb8owgk9dn)

[3.1. Essential](#h.eqs52y1nxjl)

[3.1.1 Mechanical Design](#h.9rnv7rz9w3zl)

[3.1.2 RC controlled quadcopter](#h.phuo3kxj1dv)

[3.1.3 Self-balancing quadcopter](#h.9z1alz3pqzje)

[3.1.4 Android application](#h.a030zxv1fuoa)

[3.2 Desired features](#h.lw4rh82avqwt)

[3.2.1 Pressure Sensors](#h.5big7rky7cxj)

[3.2.2 Ultrasonic sensor](#h.e9dszsj4lp6c)

[3.3 Optional features](#h.qvn1mv4ucqo)

[3.3.1 Obstacle avoidance](#h.szdso3o5nbgh)

[3.4 Non functional feature](#h.88yf3jl4rcyf)

[3.4.1 Surveillance](#h.64c9j4agyzo2)

[3.4.2 Tracking](#h.ib76u13bcfyh)

[Chapter 4. Dependencies and Deliverables](#h.ibkd18xm3yyc)

[4.1 Dependencies](#h.3nw5oz16drmg)

[4.2 Deliverables](#h.9opvnjexm97b)

[Chapter 5. Project Architecture](#h.bofltbxybprl)

[5.1 Hardware Architecture](#h.20xhnndkim6h)

[5.1.1 Quadcopter Frame](#h.6qz2z8dg5v9r)

[5.1.2 Controller](#h.tbwo2u434ey3)

[5.1.3 Sensor System](#h.4g9acbuzx4ae)

[5.1.4 Motors and Propellers](#h.1p8q5wtldu5m)

[5.1.5 ESC(Electronic Speed Control)](#h.wzza49ugyarq)

[5.1.6 Battery Unit](#h.mkf3vllgaphu)

[5.1.7 GPS and Compass](#h.7ffo4lfy8bey)

[5.1.8 Ultrasonic Sensor/Sonar Sensors](#h.qxuuemnx1mxt)

[5.2 Software Architecture](#h.865p1wxwngfp)

[Chapter 6. Project Design](#h.y4nvppb8zenw)

[6.1 Hardware Design](#h.ghajluu8wews)

[6.1.1 Frame Orientation](#h.kj9zlcirxs7r)

[6.1.2 Propellers and Motors](#h.1tkmgz2ynq2e)

[6.1.3 System Design](#h.24z54044j3eo)

[6.1.4 Hardware Interface](#h.pq3go06ovrvi)

[6.1.4.1 IMU Sensor Block](#h.c1td9t1x5r5t)

[6.1.4.2 Receiver Block](#h.gq7h2akuhuba)

[6.1.4.3 Power Distribution Block](#h.b05jjfu3apk9)

[6.2 Software Design](#h.6n8tkqat0yt2)

[6.2.1 PID](#h.eg6217rufwm1)

[6.2.3 UML Diagram](#h.r07uiajr1qt9)

[Chapter 7. QA, Performance, Deployment Plan](#h.x70qum1pu9sh)

[7.1 Quality Assurance and Performance](#h.gtqafya4rj86)

[7.1.1 IMU sensors](#h.5ve7athtnli9)

[7.1.2 PID Performance Tuning](#h.8lk7q8sy0yc6)

[7.1.3 Bluetooth Testing](#h.e31nmryj41da)

[7.2 Functionality testing](#h.xvztjyo7se00)

[7.2.1 Axis Balance](#h.vuoticutn836)

[7.2.2 Motor Balance](#h.rif0nhzemxm)

[7.2.3 Flight Testing](#h.pue06ziu5d50)

[7.2.4 Autonomous Testing](#h.fpe68elqgcb7)

[7.3 Deployment Plan](#h.ev3cxxxwt00)

[Chapter 8. Project Implementation Plan and Progress](#h.eanp0acrp2jd)

[8.1 Milestones achieved](#h.ckrp96hakr1n)

[8.2 Summary of Development Environment](#h.n1ju1qpfpz7o)

[Chapter 9. Project Schedule](#h.qqqcc9ece403)

[References](#h.yzpxij9y6s5v)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Chapter 1. Literature Search, State of the Art

## 1.1 Literature Search

A quadcopter is a popular form of UAV (Unmanned aerial vehicle). It is operated by varying the spin RPM of its four rotors to control lift and torque. The thrust from the rotors plays a key role in maneuvering and keeping the copter airborne. Its small size and swift maneuverability enables the user to perform flying routines that include complex aerial maneuvers. But for conducting such maneuvers, precise angle handling of the copter is required. The precise handling is fundamental to flying by following a user-defined complex trajectory-based path and also while performing any type of missions.

Angular maneuvering scheme along with standard flight operations such as taking-off, landing and hovering is proposed for a quadcopter with indoor or outdoor flying capabilities. This is achieved by simultaneously controlling the speed of the four rotors in order for the quadcopter to attain the correct orientation. The total thrust is determined using the inputs of altitude, pitch and roll angles. Then the thrust that the rotors must generate independently is obtained from the ratio of the angles and the calculated thrust for maintaining the input altitude. Voltage supply that is needed to spin the rotors at a certain RPM (Rotations per minute) is obtained to produce the thrust computed in the previous step. Moreover, the procedure on varying the thrust direction of rotors is also illustrated to perform the standard flight operations.

## 1.2 State-of-the-Art

Commercially available flight controller software’s are expensive and patented. As they are not free and for the learning purpose we are developing our own flight controller software. We are creating this software for LPC1758 ARM cortex M3 processor on FreeRTOS. The software will be open source and can be used in future for enhancements. We will be using MATLAB tool to tune the PID values. These PID values need to be set according to the Inertial Measurement Unit values. It is an integrated sensor device that measures acceleration and tilt by accelerometer, angular velocity and orientation by Gyroscope and gravitational forces by Magnetometer. Inertial Measurement Unit by Adafruit provides 11 DOF including 3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer, barometric pressure sensor and temperature sensor. These values are sent to SJ-1 board over I2C, where SJ-1 board to determine the angular position and orientation of the Quadcopter does the further processing. We will be developing an android application for setting the source and destination points. This navigation path will be determined using google maps.

# 

# Chapter 2. Project Justification

In this project we will be developing a new flight controller software to control the navigation of the quadcopter where the source and the destination point is set using android application. This application uses the google maps to navigate from one point to another. This is sent to the microcontroller using Bluetooth device. The co-ordinates are obtained from the GPS mounted on the quadcopter. This data is passed to the flight controller software to navigate.

The flight controller software is developed for LPC1758 ARM M3 cortex processor using FreeRTOS. FreeRTOS is an open source and the LPC1758 is compatible for the light load quadcopter. As it can process at the rate of 48MHz and has necessary computation to perform the required task. The flight controller software will be an open source, in future it can be integrated with other sensors or devices to enhancement of the applications of the quadcopter. As the commercial available software are restricted for enhancement, creating a new algorithm and making it an open source would be available for future betterment. This would also help us to understand the challenge faced in developing the algorithm to control the quadcopter when the source and destination is set. These help us to understand the PID algorithm and tuning of the PID values to obtain a smooth flight operation.

Algorithm developed may help for future research for the students and also this can help us to provide solutions for the better flight control. This may help in using the quadcopter for more applications. Thus developing this open source flight controller algorithm would provide the base for more research and enhancement opportunity.

# Chapter 3. Project Requirements

As a user, the android application will be used to set the destination for the quadcopter and the quadcopter will hover to the desired destination using the flight control algorithm. The quadcopter can be sent to a desired location for different applications such as:

1. A state of emergency where a person requires medical help as a first response by providing a first aid kit.

2. Delivering mails.

3. Surveillance drone to monitor a particular area, and many more.

## 3.1. Essential

### 3.1.1 Mechanical Design

Designing a quadcopter frame, which is symmetrical, light weight and can host all the components like motors, LPC1758 microcontroller, ESC, IMU sensors, battery and a RC receiver.

### 3.1.2 RC controlled quadcopter

The frame consists of receiver (RX), which receives data from the RC Transmitter (TX). The quadcopter will be controlled manually by RC transmitter.

### 3.1.3 Self-balancing quadcopter

During the flight, the quadcopter may be unbalanced due to the weather conditions such as wind. An algorithm should be developed where the flight controller will self-balance the quadcopter.

### 3.1.4 Android application

The user interface to the quadcopter will be through an android application. The application will provide the user to set the coordinates, where the quadcopter uses them as the destination.

## 3.2 Desired features

### 3.2.1 Pressure Sensors

For the autonomous flight the quadcopter has fly at a certain altitude to move from point A to point B, the pressure sensors are integrated.

### 3.2.2 Ultrasonic sensor

During the autonomous flight, when the quadcopter reaches the destination, ultrasonic sensors are used to measure the distance from the ground and can land safely.

Using all the above mentioned essential features and desired features, our aim is to obtain a stable flight without manual interference. The only feature that will require human involvement will be setting the destination using a mobile application. The quadcopter will hover from a source A to destination B.

## 3.3 Optional features

### 3.3.1 Obstacle avoidance

During the autonomous flight, obstacles such as trees, building can be avoided and an alternative path will be provided to reach the destination by using ultrasonic sensors and image processing.

## 3.4 Nonfunctional feature

### 

### 3.4.1 Surveillance

A camera can be attached to the quadcopter, wherein a live video feed can be provided during concerts and events.

### 3.4.2 Tracking

The autonomous quadcopter with a camera can be used to follow a human during biking and other numerous sporting events.

# Chapter 4. Dependencies and Deliverables

## 4.1 Dependencies

* The main concern of our project is tuning the PID (Proportional Integral Derivative) for pitch, roll and yaw, where PID is a closed loop control loop feedback mechanism used to balance the quadcopter.
* Another major dependencies would be battery in any quadcopter project. If the size of the battery is increased, the flight time also increases, quadcopter becomes heavier and less agile it becomes. Also, the battery drains faster during the flight providing less time for continuous tests.
* The expensive parts of quadcopters can be damaged frequently during the tests and sometimes would halt the process waiting the new parts.
* Quadcopters should be tested in outdoors, the weather conditions hinders testing and getting the proper GPS fix.
* Algorithms to detect obstacles and implementing an additional PID over the existing PID during its navigation towards the destination.

## 4.2 Deliverables

### 

* Attaining a well-tuned PID for achieving a self-balanced quadcopter. During the flight, the quadcopter balances itself when there is a change in quadcopter’s angle due to the wind.
* Developing a flight controller algorithm, which interfaces with all the peripherals to attain the overall stability of the quadcopter.
* Achieving autonomous flight would be the zenith aim for our project, where the quadcopter would self-navigate itself to the destination. The destination is set by an Android application.

### 

# Chapter 5. Project Architecture

Quadcopters also called as Unmanned Aerial vehicles or the UAVs are driven by four independent propellers which are fixed at the vertex of a plus or a cross shaped pattern. The fact that they have a relatively long lever arm between the opposite motors makes them more agile that the standard helicopters. The commercial quadcopters are very cheap now and are closed source. This chapter will discuss both the hardware and software architecture which we are focusing on to make a cheap quadcopter available for students for their future research and also allow them to modify the application and integrate other features with our existing Autonomous Quadcopter design.

## 5.1 Hardware Architecture

The main things we are focusing on in this section are the problems faced by consumers for deciding which quadcopter they should buy. The main goals driving our architecture design are as followed.

* Low cost for all the components will make sure that the end product is cost efficient as compared to other available products in the market.
* Using a controller which has adequate computational power for the reduced payload of a smaller version of drone.

Considering the above concerns, we came up with proper mechanical frame, inertial measurement unit sensors and LPC1758 controller as our flight controller board which is used widely by the San Jose State students for working on their embedded course. The overall architecture for the project is shown in the Figure 1.

### 5.1.1 Quadcopter Frame

The Quadcopter frame must be symmetrical to make the code simpler to implement and avoid any unnecessary complications. Ours is the SK450 Glass Fiber Frame of size 450mm. This is a high quality glass fiber frame while the arms are constructed from ultra-durable polyamide nylon. This quad not only looks great, it’s very well thought out as well. Assembly is a breeze with pre-threaded brass sleeves for all of the frame bolts, so no lock-nuts are required. It utilizes one size of bolt for the entire build, making the hardware very easy to keep in order and only requiring one size of hex wrench to assemble.

### 5.1.2 Controller

The brain of the quadcopter is the flight controller board. The board we are using for this project is NXPs LPC 1758 which has an ARM Cortex -M3 microcontroller. This controller is widely used for embedded applications because of its low power consumption. This controller has good debugging features and includes 512kb flash memory and supports all the protocols which we are required for interfacing the other peripherals.

### 5.1.3 Sensor System

The next important hardware component required will be the sensor system which consists of accelerometer, gyroscope and magnetometer and other sensors required to calculate the altitude. The Inertial Measurement Unit (IMU) is an integrated sensor device which measures acceleration and tilt by Accelerometer, angular velocity and orientation by Gyroscope and gravitational forces by Magnetometer. Inertial Measurement Unit by Adafruit provides 10 DOF (actually 11) i.e. 3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer barometric pressure sensor and temperature sensor. These values are sent to LPC1758 board over I2C, where the further processing is done by LPC1758 board to determine the angular position and orientation of the Quadcopter.

### 5.1.4 Motors and Propellers

The motors used for our project will be brushless motors. Brushless motors are the best when it comes to low power applications and consumption. The motors which we are using for this project are AIR Gear 350 brushless motors. These motors have 920Kv rating and have low noise and high speed response. The propellers used came with the same motors and were self-tightening and bullets holders. The motors are made of good quality material and are suitable for 1400-1700 g quadcopter. The propellers dimensions are as follows: Diameter - 9.4 inch and Thread Pitch - 4.5 inch.

### 5.1.5 ESC (Electronic Speed Control)

The ESCs which we are using for our project will be AfroESC which uses SimonK Firmware. The PWM signals generated by the Flight control board is fed to the ESC's which provide signal to the motors. The ESCs are running on SimonK firmware which gives a smooth power response and is suitable for multi-rotor use without the need to program or adjust settings.

### 5.1.6 Battery Unit

The whole system is powered by using 4000mAH LiPo battery which was power the Flight controller board. This battery will guarantee that the quad will hover for around 12mins over normal conditions.

### 5.1.7 GPS and Compass

The GPS module must be finely accurate to around 2 - 5meters of range. For achieving this we came up with the Adafruits GPS module which is fairly accurate in providing locations. The compass module used will be HMC6352 which is very cheap and highly accurate after proper calibration is done.

### 

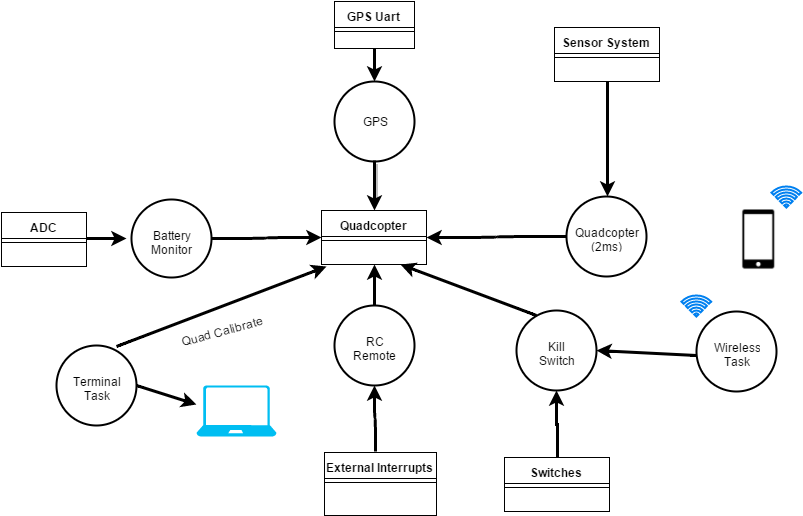
### 5.1.8 Ultrasonic Sensor/Sonar Sensors

The LVMax long distance sonar sensors are used to detect the objects sends sound waves when triggered and then the sensor signal emits a PWM that needs to be timed to detect the distance. The PWM width depends on the time it takes for the sound waves to bounce back from an object.

## 5.2 Software Architecture

The software architecture which we are developing for our project has the following specifications”

* An open source code which can be used by students for research and learning purpose.
* using an objected oriented approach and high level programming language such as C++/python
* Providing a working flight control system which can be tested easily.
* Providing automatize features for calibration process.
* Easy integration of extra functionalities and an interface to MATLAB simulink.
* A mobile application which will provide a track of the quad and has a UI which is user friendly.



**Figure 1: Project Architecture**

# Chapter 6. Project Design

This section will briefly discuss about the hardware and software design for our quadcopter.

## 6.1 Hardware Design

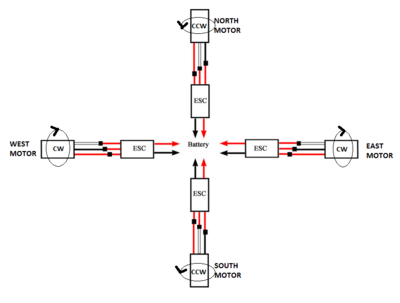
The hardware design part deals with the basic hardware design principles required for the quadcopter to achieve stability.

### 6.1.1 Frame Orientation

It is very important to decide which frame orientation one should use for building his quadcopter from scratch. In our implementation we have decided to go with ‘+’ or the plus configuration. The reason we went with this configuration is because:

1. It is easy to implement the pitch and roll with plus configuration as we have to deal with two motors for forward and backward movement of the quad whereas, the X will require all four motors to work.
2. In case of a crash, only one propeller will be damaged whereas for X, two propellers will be damaged.
3. Also, we are not using a camera which in that case would have required an X configuration or H and plus would have been a problem.

The Quadcopter frame must be symmetrical to make the code simpler to implement and avoid any unnecessary complications. Ours is the SK450 Glass Fiber Frame of size 450mm. This is a high quality glass fiber frame while the arms are constructed from ultra-durable polyamide nylon. This quad not only looks great, it’s very well thought out as well. Assembly is a breeze with pre-threaded brass sleeves for all of the frame bolts, so no lock-nuts are required. It utilizes one size of bolt for the entire build, making the hardware very easy to keep in order and only requiring one size of hex wrench to assemble.



**Figure.2 Plus configuration of our quadcopter**

* Figure shows that our quadcopter is implemented in the '+' orientation.
* The motors are named North, South, East and West motor.
* Figure shows the ESC and motor setup where CW is clockwise and CCW means counter-clockwise.
* The CW and CCW setup of the motor and propellers is very important to have control over the yaw parameter.

### 6.1.2 Propellers and Motors

Brushless motors are the best when it comes to speed and low power consumption. The motors used for this project are AIR Gear 350 brushless motors. These motors have 920Kv rating and have low noise and high speed response. The propellers used came with the same motors and were self-tightening and bullets holders. The motors are made of good quality material and are suitable for 1400-1700 g quadcopter. The propellers dimensions are as follows:

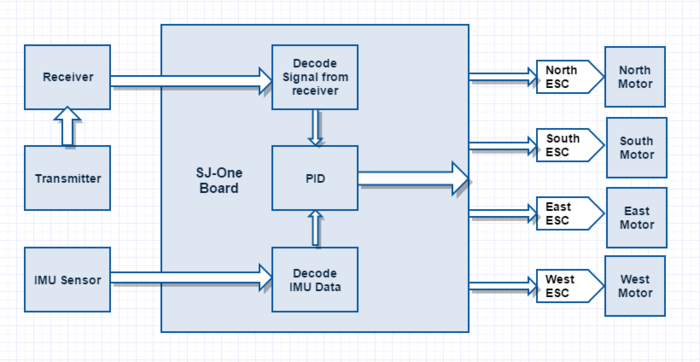
Diameter - 9.4 inch

Thread Pitch - 4.5 inch

### 6.1.3 System Design

The system design diagram for our quadcopter build is shown below. The basic flow of the system can be explained from the diagram.

The Transmitter will send a signal to Arm/Disarm the quadcopter which will be fed to the receiving unit of the system. The receiving unit will send the signal to the decoding unit and depending upon which channel it has received the signal the controller will take appropriate actions after sending it to the PID block. The PID control loop is continuously monitoring the values from the IMU sensor. The PID block will along with the sensor values and receiver values will ultimately make a decision of controlling the propellers.



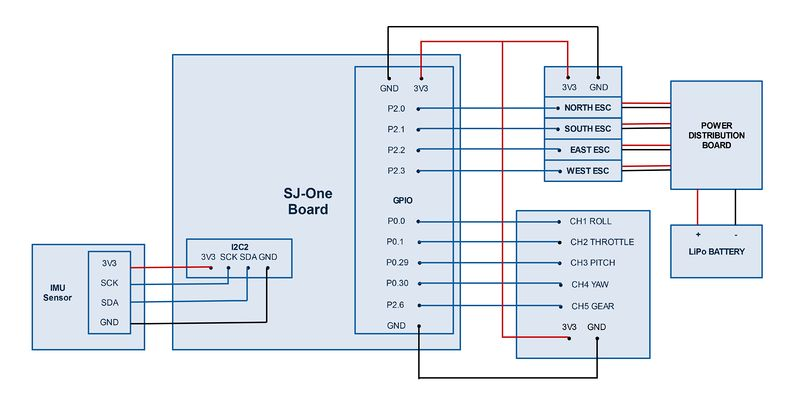
**Figure 3. System design**

### 6.1.4 Hardware Interface

The hardware interface for our implementation of the drone is shown below in the figure. The complete system during its final implementation will have two additional modules one as GPS and other as sensors module.

As shown in the figure below we have used our main controller as LPC1758 or the SJ-One Board. Using FreeRTOS as the microcontroller OS, we add these different modules to the scheduler for performing timely tasks. The motor controller must run at more than 100Hz so as to achieve a better control over the quadcopter while operating it with RC for the initial phase test. The other blocks in the hardware implementation have their functions as follows:

Power Distribution block.



**Figure. 4 Hardware Interface for quadcopter**

### 6.1.4.1 IMU Sensor Block

This connects the sensor system via i2c interface to the SJ-One board. This module must be placed on a flat non-vibrating surface and can be covered with vibration absorbing materials for best results.

### 6.1.4.2 Receiver Block

This block is connected to the SJ-One board using the onboard GPIOs. They are hooked with external interrupts which are configured as both falling and rising edge triggered.

### 6.1.4.2 Power Distribution Block

This block will be responsible to power up the controller, receiver and the motors. We are using 4500mAH LiPo battery to fly the quadcopter which will ensure a flight time of around 15-20mins.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Port and Pin Number** | **Pin Type** | **Purpose** |
| 1 | P2.0 | PWM | North ESC |
| 2 | P2.1 | PWM | South ESC |
| 3 | P2.2 | PWM | East ESC |
| 4 | P2.3 | PWM | West ESC |

**Table.1 Pin diagram for motors and SJ-One interfacing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Port and Pin Number** | **Pin Type** | **Purpose** |
| 1 | P0.0 | GPIO | Pitch (Rx Channel 3) |
| 2 | P0.1 | GPIO | Roll (Rx Channel 1) |
| 3 | P0.29 | GPIO | Yaw (Rx Channel 4) |
| 4 | P0.30 | GPIO | Throttle (Rx Channel 2) |
| 5 | P2.6 | GPIO | Arm/Disarm (Rx Channel 5) |

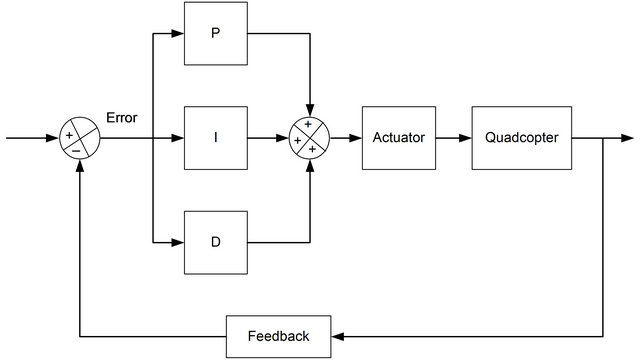
**Table.2 Pin diagram for receiver and SJ-One interfacing**

## 6.2 Software Design

### 6.2.1 PID

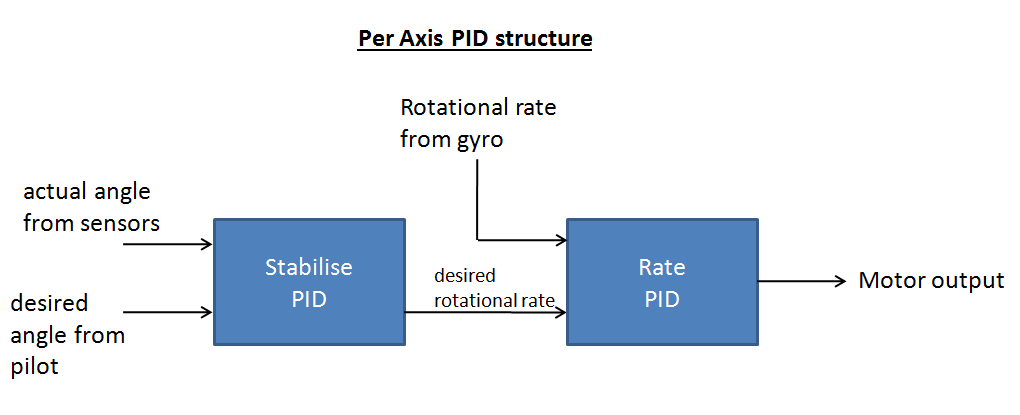
A proportional integral derivative controller (PID) is a closed loop control feedback mechanism. PID tries to get the actual result closer to the desired result by adjusting the input. All the Quadcopter and multi-copters use PID controller to achieve stability.

PID consists of 3 algorithms i.e. P (proportional), I (integral) and D (derivative). P dependent on the present error, I is dependent on error and D is dependent on future error.



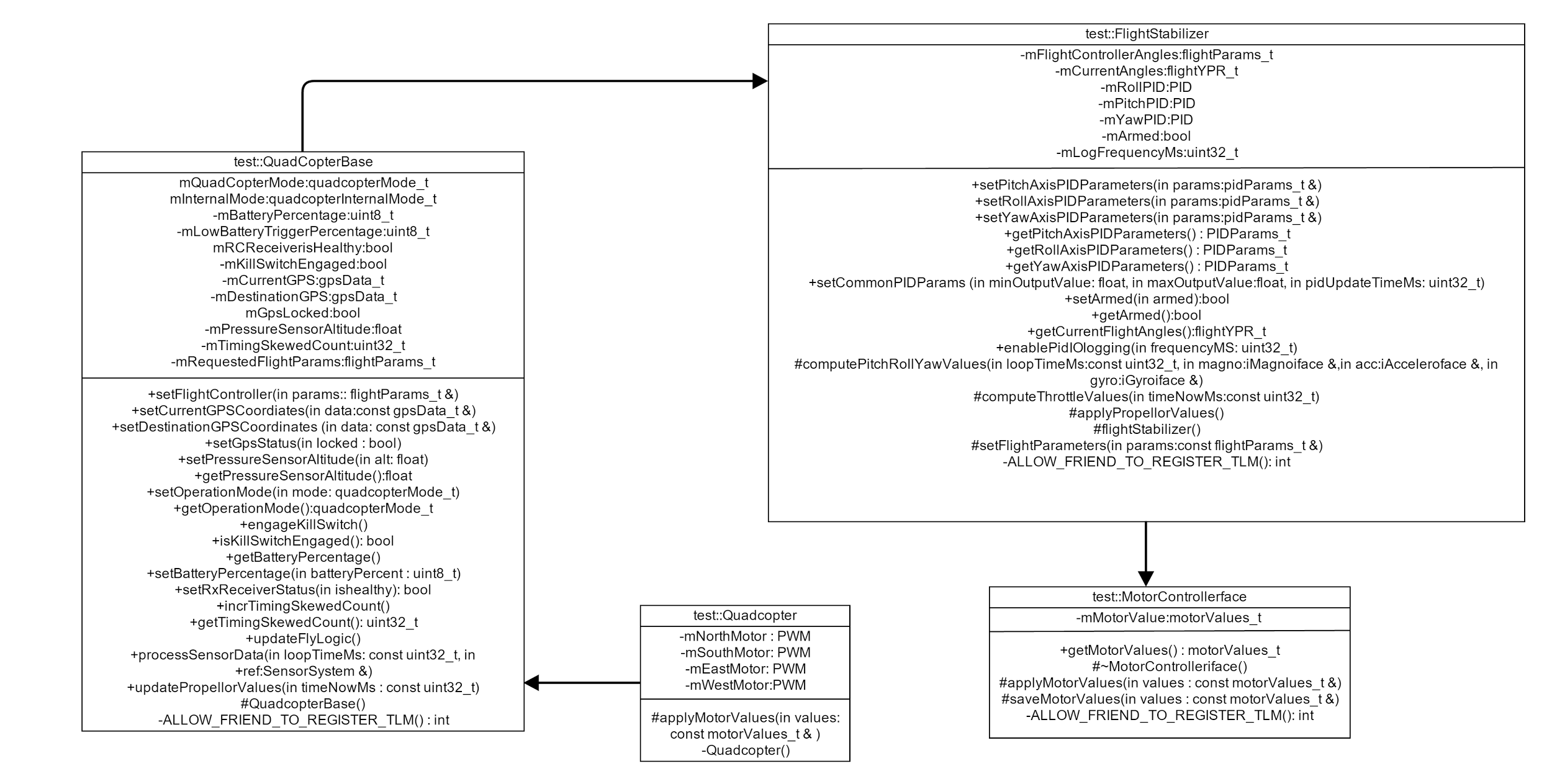
**Fig 5: Block diagram of PID**

To achieve the stability of the quadcopter, PID is applied on each motor. The way PID works is the sensors mounted on the quadcopter gives current pitch, roll and yaw angles to the flight controller. The flight controller has PID coded in software, when a new pitch value is provided by RC to the quadcopter, the difference is considered as error and P, I and D algorithm is applied to all the motors. The coefficients of PID applied to each motor is shown in the block diagram.



**Figure 6: PID per axis**

### 6.2.3 UML Diagram



**Figure 7: UML Block diagram**

The motorControllerIface is the parent class. FlightStabilizer is inherited from motorControllerIface, Quadcopter Base is inherited from FlightStabilizer and Quadcopter is inherited from QuadcopterBase.

# Chapter 7. QA, Performance, Deployment Plan

**Workbook 2**

## 7.1 Quality Assurance and Performance

After obtaining the parts of the quadcopter, it’s time to assemble and provide connections as per the design. Unit testing is carried out in each stage of the implementation to assure the total quality of the system.

### 7.1.1 IMU sensors

The first testing attempted was I2C read/write testing. Once the sensors were wired up, a quick while loop with a delay was implemented to show that the sensors could be read reliably. IMU sensors are tested using the application called IMU Razor 9DOF. This application shows the simulation of our IMU sensor. It displays values of pitch, roll and yaw.

### 7.1.2 PID Performance Tuning

The testing of PID was done for each axis separately. We started with pitch axis. We used two wooden planks and hooks to hold the quad on the east and west so that our pitch axis is free. Slowly started to increase the value of P. At some point the pitch axis was oscillating constantly. The amplitude of oscillation should neither increase nor decrease. Then we started increasing the values of D till the overshoot is reduced to an acceptable level. Then we increased the value of I till the final error is zero. The quad is very sensitive to the value of I. After the pitch axis we tuned the PID for the roll axis. We used some tuning methods such as Zeigler Nicholas and Cohen Coon to tune the PID. After the pitch and roll we tested the yaw axis. We tied the quad to a rope and held it in the air so that the quad is free in the yaw axis.

### 

### 7.1.3 Bluetooth Testing

Testing the Bluetooth is done using an android application called “SENA BTerm Bluetooth Terminal”. This application establishes a connection between the android device and the Bluetooth module. Program the development board to transmit character “A” to the Bluetooth module using UART. When this is transmitted the same should be received at the android application as these two devices are paired for communication.

## 7.2 Functionality testing

Next phase of testing is interface testing of IMU sensor output and the PID algorithm. It is carried out in four parts. The first three are tested using the remote controller. The last one is tested using the android app developed on a smartphone setting the start point and destination.

### 7.2.1 Axis Balance

Testing by tying the ropes on each side of the blades, and test the balance of each axis individually. With this setup, the team is able to fine tune the balancing algorithm.

### 7.2.2 Motor Balance

Testing by tying the quadcopter to an elevated platform. The platform allows the copter to full degree of motion. This platform enables us to test the quadcopter's balance with all motors on.

### 7.2.3 Flight Testing

Testing the flight controller software using remote controller and guiding the quadcopter to the destination.

### 7.2.4 Autonomous Testing

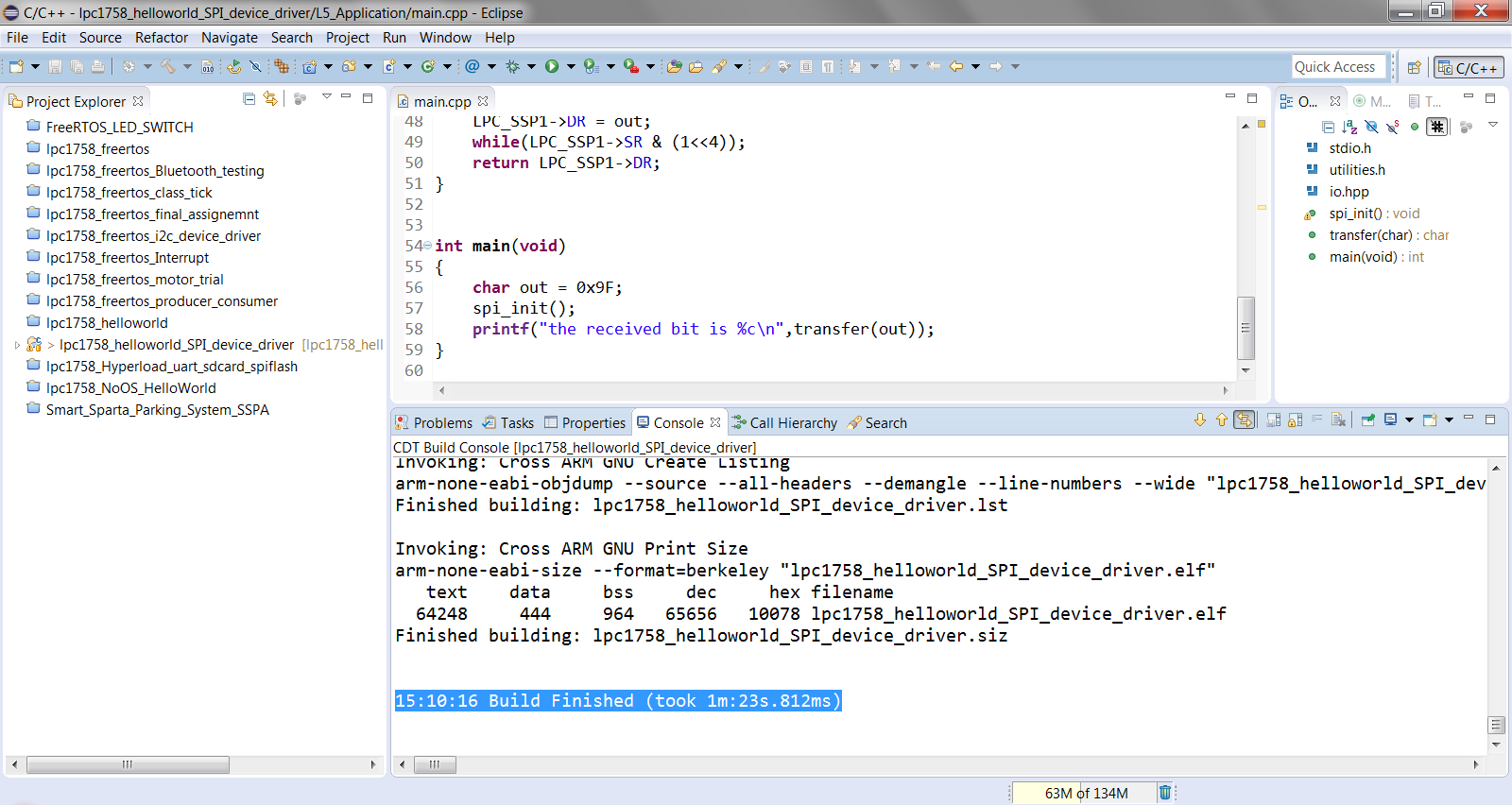
Testing the quadcopter by sending the start and destination points using an android app to the Bluetooth module mounted on the system.

## 7.3 Deployment Plan

The procedure for the deployment plan is as follows.

Below are the steps for generating a compiled file.

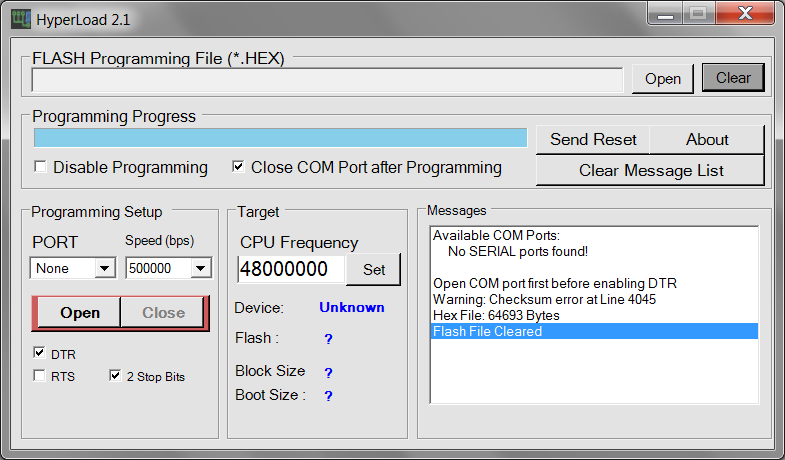
1. Download the software from the below link <http://sourceforge.net/projects/armdevpkg/files/>
2. Open the folder SJSU-Dev
3. Plug in the development board.
4. Once the device driver is installed. Open Eclipse32\_Emb.bat file. Eclipse framework will open in a new window.
5. Edit the main.c and other files according to the requirement.
6. Close all the other projects except the one you need to compile.
7. Build the project using cntl+B. Which is success when you receive a message “Build Finished”.



**Figure 7: Image showing the successful completion.**

Next step is to load the compiled file onto the microcontroller.

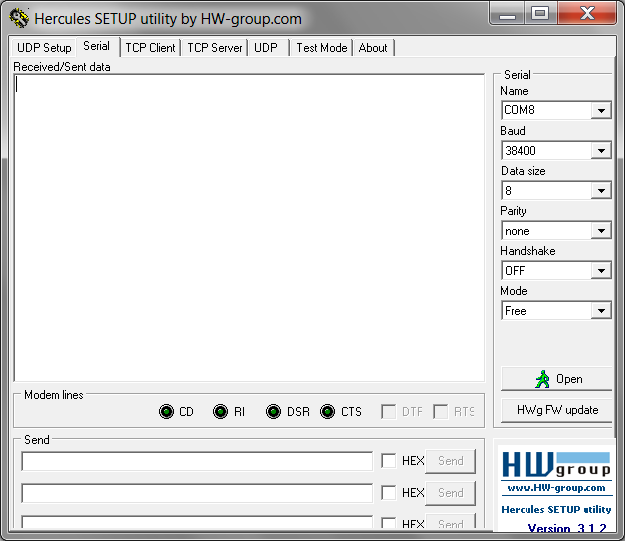
1. Open the Hyperload from the SJSU-Dev folder.
2. Board appears in the COM port.
3. In Eclipse, a file with extension \*.hex will be generated if the compilation was successful in the folder \_Build
4. Drag and drop the file in the Hyperload at Flash Programming File section.
5. Press “Send Reset” button on Hyperload.
6. After the programmed is flashed on to the microcontroller. Please close the COM port in Hyperload.



**Figure 8: Image of the Hyperload**.

Next step is to view the output in Hercules.

1. Open Hercules in the same folder SJSU-Dev
2. Browse to the “Serial” port tab and select the Baud rate to 38400bps.
3. Select the same COM port as we had selected in the Hyperload to flash the program into the development board.
4. Reset the development board to view the required output on the console.



**Figure 9: Image of Hercules Serial port tab**

# Chapter 8. Project Implementation Plan and Progress

1. Install Eclipse Luna and Extract SJSU development package for software development.

2. Setup GIT lab for code collaboration.

3. RTOS is used for our implementation.

4. Understand the data sheet and implement motor and ESC driver.

5. Understand the data sheet and implement IMU sensor driver.

6. Understand the data sheet and implement RC remote driver.

7. Implement flight control software.

8. Implement calibration software for IMU sensor and motors.

9. Create quadcopter task using RTOS.

* Initializes the Adafruit 10DOF IMU.
* Update and Process sensor data.
* Update the flight parameters.
* Update propeller values.
* Compute pitch, roll and yaw throttle.
* Apply throttle values if the quad-copter is armed.
* Indicate error if any the above process take more time.

10. Create a RC remote task.

* Generate Interrupt when data is send by RC remote.
* Configure five RC channels for Throttle, Pitch, Roll, Yaw and arm/disarm).
* Scale the throttle, pitch, roll and yaw axis.

## 8.1 Milestones achieved

* **Ordering the parts for quadcopter.**
* **Detailed study about PID.**
* **Assembling the quadcopter.**
* **Study the quadcopter’s software framework.**
* **Tested connection for Remote control.**
* **Calibrated the ESC and tested the motors**
* **Calibrated and tested the 10DOF sensor.**

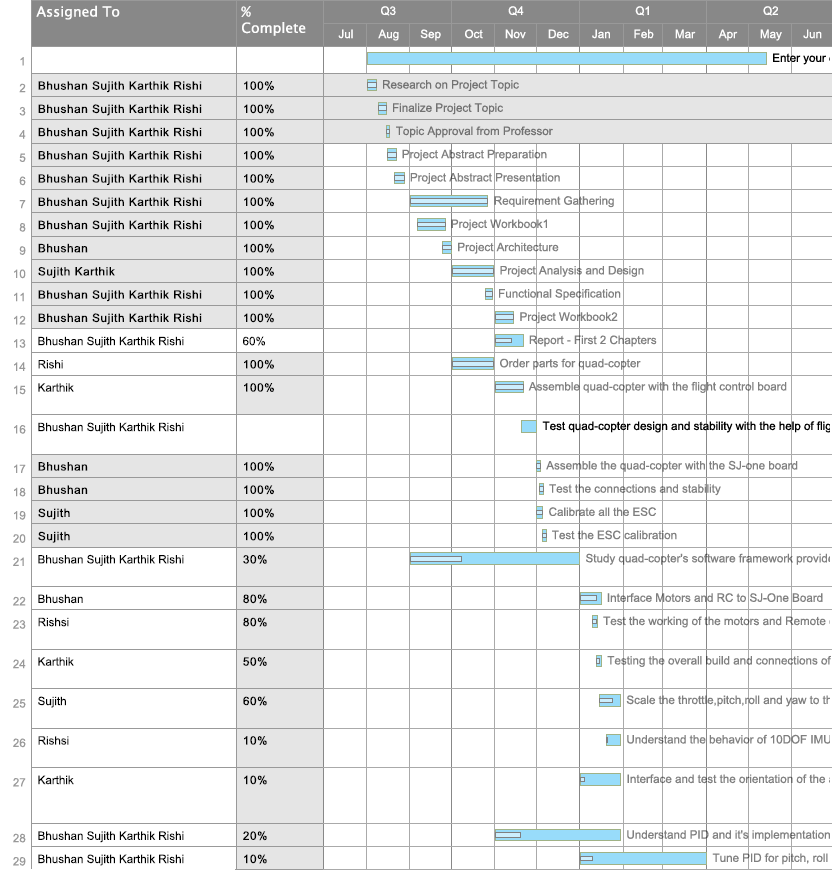
## 8.2 Summary of Development Environment

* **Programming Languages: C, C++.**
* **Mobile Development: Android SDK and eclipse.**
* **Version Control: GIT Lab.**

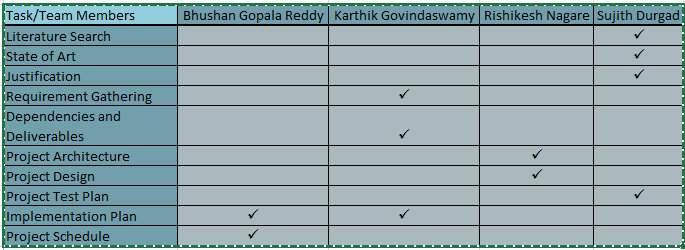
# Chapter 9. Project Schedule

**9.1 Project Progress**





**9.2 Workbook Individual Task Assignment**



## References

[1] P.J. Bristeau, F. Callou, D. Vissière, and N. Petit. The Navigation and Control technology inside the AR.Drone micro UAV. In 18th IFAC World Congress, Aug-Sep 2011.

[2] A. Franchi, C. Masone, V. Grabe, M. Ryll, H. H. Bülthoff, and P. Robuffo Giordano. Modeling and control of UAV bearing-formations with bilateral high-level steering. The International Journal of Robotics Research, Special Issue on 3D Exploration, Mapping, and Surveillance, 31(12):1504–1525, 2012.

[3] V. Grabe, H. H. Bülthoff, and P. Robuffo Giordano. On-board velocity estimation and closed-loop control of a quadrotor UAV based on optical flow. In 2012 IEEE Int. Conf. on Robotics and Automation, pages 491–497, St. Paul, MN, May 2012.

[4] I. Sa and P. Corke. System identification, estimation and control for a cost effective open-source quadcopter. In 2012 IEEE Int. Conf. on Robotics and Automation, pages 2035–2041, May 2012.

[5] Riccardo Spica. (2013) *Open Source Hardware/Software Architecture*. [Online] Available: <http://www.irisa.fr/lagadic/pdf/2013_spica_uas.pdf>

[6] Preetpal Kang. (Dec, 2014) *Drones.* Available: <http://www.socialledge.com/sjsu/index.php?title=S15:_Drone#Hardware_Interface>

[7] Preetpal Kang. (Dec, 2014) *Development\_Package.* Available: http://www.socialledge.com/sjsu/index.php?title=Development\_Package