Building a Quadcopter

USING ARDUINO WITH ANDROID

Prepared by:

M. Khdiri A. Amir

2015 - 2016

I. EXECUTIVE SUMMARY

A quadcopter is an aircraft based on four propellers, quadcopters can be controlled by a radio transmission or operate under the guidance of limited autonomous protocols.

The purpose of building a quadcopter is to solve a specific problem in air. Build a quadcopter is to get information in a new way that is flight, or to flight purpose itself. It can use a camera that can be used by traffic police in roads to see what drivers do in a long distance, or to film making.

Quadcopters are UAVs (Unmanned Aerial Vehicles) that can be used in military.

Arduino is an open-source system. Using Arduino is to decide by yourself which technologies and features you want to use into the quadcopter. This quadcopter project use sensor to control the top and landing of the quadcopter.

Android also is an open-source system, it is easy and low-cost to control the quadcopter flight with Android mobiles. So it needs to create a program for Android devices that will compatible with the Arduino to control the flight by controlling the quadcopter motors. It needs to connect the Android device with the Arduino by using wireless or Bluetooth shield.

II. TABLE OF CONTENTS

III. TABLE OF FIGURE

CHAPTER ONE

INTRODUCTION

CHAPTER ONE
INTRODUCTION

1.1 QUADCOPTER

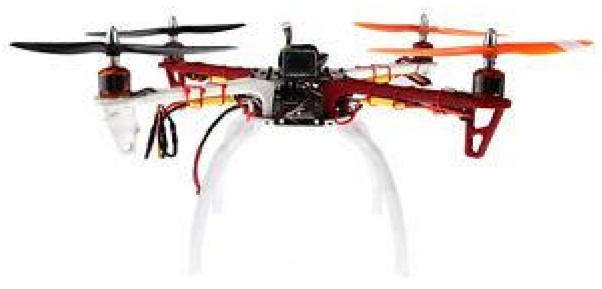


Figure 1: Quadcopter.

Quadcopter, also known as quadrotor, is a helicopter with four rotors. The rotors mostly consist of two or three rotor blades that are placed in a square format that they have equal distance from the center of the quadcopter, because of stabilization. The movement of the quadcopter is controlled by adjusting the angular velocity of the rotors that are rotated by electric motors. (Luukkonen, 2011).

Quadcopters are classification of multi-copters; multi-copters have different names depending on the number of propellers mounted on the machine. A multi-copter with four propellers is called a quad-copter, a six propeller multi-copter called a hexa-copter, and nowadays we even have octo-copters.

Quadcopter is an unmanned aerial vehicle (UAV), it is used in military, surveillance, film making, search and rescue mission, and other applications. (Parker, Robbiano, & Bottorff, 2011).

Quadcopter has advantages over helicopter because of its simple design. A helicopter provides the lift by only one blade that the lift is equal to helicopter's weight. Quadcopter needs the stability that provides by the four motors and their blades. These four propellers in a quadcopter use two pairs, one of them will spinning clockwise and the other counter-clockwise. (VECHIAN, 2012).

1.2 HISTORY OF MULTI-COPTER

Since 1907, some designer tried and designed multi-copters with more than two rotors. At 1922, Dr. George de Bothezat and Ivan Jerome developed a helicopter with six bladed rotors that was a manned version multi-copter named the De Bothezat helicopter, eventually two of the six rotors were became unnecessary and eliminated, so it became the quadcopter. Quadcopter at the first time developed and prototyped under a U.S. Army contract. (Norris, 2014).

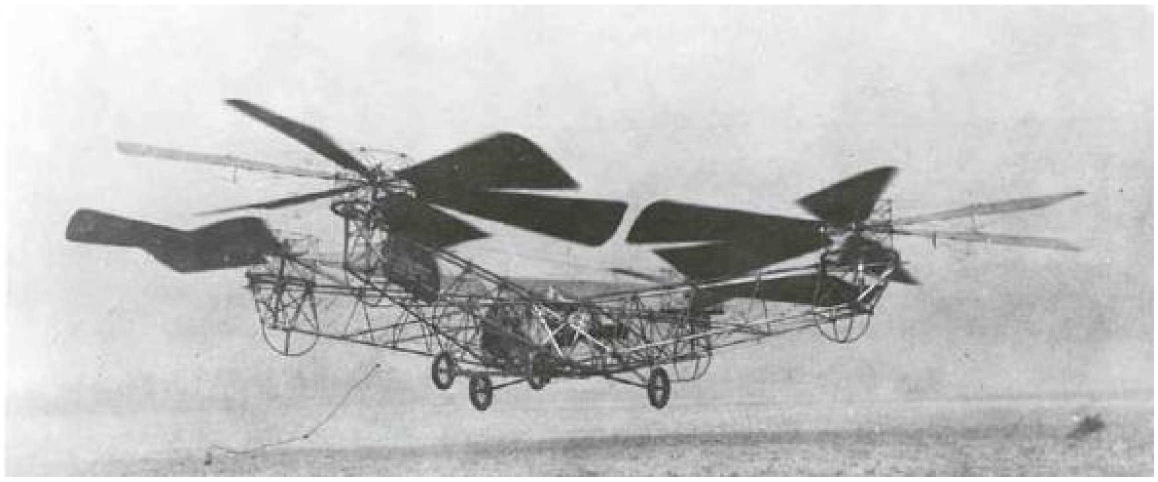


Figure 2: Quadcopter history (Norris, 2014).

1.3 APPLICATIONS

Quadcopters can be used in researches that can take photos and videos of places that human can't go there, or because of having camera, they can be used for image recognition systems.

Quadcopters are UAVs, so they are good in surveillance, in military to spy or to rescue teams in missions. Quadcopters can even save lives! They can be used in medical emergency cases like in cardiac arrest an ambulance drone can be sent quickly to the patient and it is used like some kind of medical emergency such as DC shock.

1.4 AIM OF THE PROJECT

The goal of this project is to build somehow self-reliant copter that can be capable to vertical lift, landing, moving forward/backward horizontally and hovering, and it can be able to stably rotating at one location. It has sensors to control flight and stability.

1.5 COMPONENTS OF THE QUADCOPTER

Quadcopter needs some basic components to be build includes:

- Frame
- Propellers
- Motors
- Electronic speed controllers (ESC)
- Battery

- Flight controller (FC)

1.6 UNMANNED AERIAL VEHICLES (UAVS)

quadcopters are unmanned aerial vehicles, there are three types of **UAV**s according to level of artificial intelligence **(AI)**:

1.6.1 FULLY REMOTED VEHICLES

A pilot in an earth terminal has fully controlled over the craft, tells the copter what to do and where to go by a remote-controller from the earth terminal, so every movement of the copter will be controlled by the pilot. (Tzivaras, 2015).

1.6.2 HALF-CASTE REMOTE-CONTROLLED

This level of the craft is between a fully controlled craft and independent flying quadcopter. The pilot communicates between the quadcopter and the earth terminal, and do not need to control everything the craft will do. It only needs to control and set the places that the craft should go. (Tzivaras, 2015).

1.6.3 INDEPENDENT CRAFT VEHICLES

This type of flying has no pilot, the quadcopter uses the sensors and other features that we need to operate our goals such as camera, GPS, etc... And programmed to go according the sensors. It gets data from the sensors so they tell where to go and what to do according to the programmed system. (Tzivaras, 2015).

1.7 FLIGHT CONTROLLER

1.7.1 ARDUINO

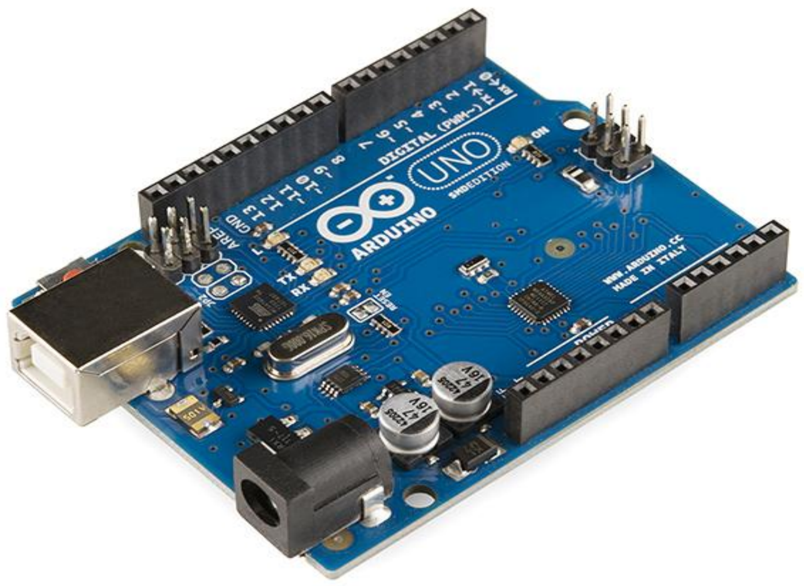


Figure 3: Arduino.

Why Arduino?

Arduino is an open-source system based on easy-to-use hardware and software, it is a microcontroller-based kits. (What is Arduino?, n.d.).

Arduino has sets of digital and analog input/output pins, so it can be used with many other circuits and shields, these shields can provide many features, so Arduino can be expansion using shields, shields can provide many features including: GPS, Ethernet, LCD, motor controls, Bluetooth, wireless, etc... And the shields can also be made DIY (Do it yourself). (What is Arduino?, n.d.).

For programming the microcontrollers, Arduino use the Arduino programming language Based-On-Wiring, and the Arduino software that is an integrated development environment (IDE) Based-On-Processing, that can support C and C++ programming languages. (What is Arduino?, n.d.).

Arduino is inexpensive and it can be programmed on various platform including Windows, Mac OS X, or Linux. It is simple and easy for programming even for beginners. It is open-source and extensible hardware-and-software. (What is Arduino?, n.d.).

1.7.2 ANDROID SOFTWARE (REMOTE-CONTROLLER)

Android is an open-source system that has many applications and users, generally many developers work on open-source systems so it is cheap and easy to use. In general, quadcopters use a specific transmitter that creates for the quad to control its movements, and a receiver that compatible with the transmitter. It needs a high-cost if we want to build a quadcopter with that remote-controller, and the developing works do not show as much as to see, also the enjoyment of the work will be missing, and the work will be done in a simple and routine way without any invention. But programming our own application for Android devices will be a low-cost, and it will be a bigger and significant thing if the quadcopter can be controlled by the mobile, it is impressive. (Todd & Barraclough, 2016).

Also this project will use some sensors that they also doing movement control in some ways if the quad needs to be controlled from its environment when the copter will get some unwanted situation such as unexpected rapid landing, so the quadcopter can control its landing using these programmed sensors. It means that this quadcopter will be some kind of half-caste (hybrid) remote-controlled vehicle and it will not be fully controlled by the pilot. (Todd & Barraclough, 2016).

CHAPTER TWO

COMPONENTS

COMPONENTS

Quadcopter needs some basic components to be build includes:

- Frame
- Propellers
- Motors
- Electronic speed controllers (ESC)
- Battery
- Flight controller (FC) and sensors

2.1 FRAME

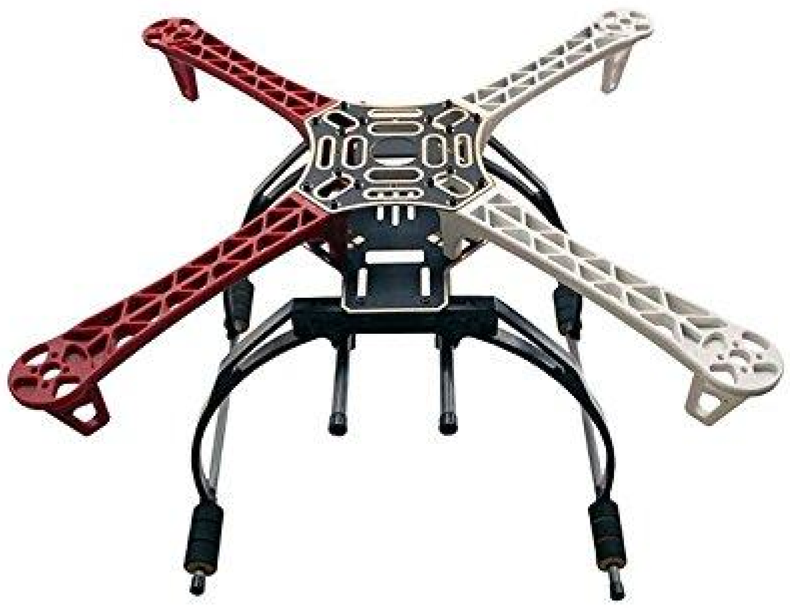


Figure 4: Quadcopter F450 frame.

Every quad-copter needs a frame to house all the other components, what do you consider is the weight, size and materials. we chose the RC FlameWheel F450. This is built from very strong materials, the arms are made from the ultra-strong PA66+30GF material which provides better resistance to damage on hard landings, while the main frame plates use a high strength compound PCB material, which makes wiring of ESCs and battery easy and safe on the lower of the two frame plates which is also the power distribution board. The overall frame design provides enough space when assembled to fit an autopilot system between the top and bottom plates of the Flame Wheel. The arms supplied in the kit are different colors, 2 red and 2 white to allow for improved visual orientation in flight The weight of the frame with legs is 430 grams this is normal weight and easy to lifting. And the size is 450 mm this is fit with the propellers. (Quadcopter Parts List | What You Need to Build a DIY Quadcopter, n.d.).

2.2 MOTORS

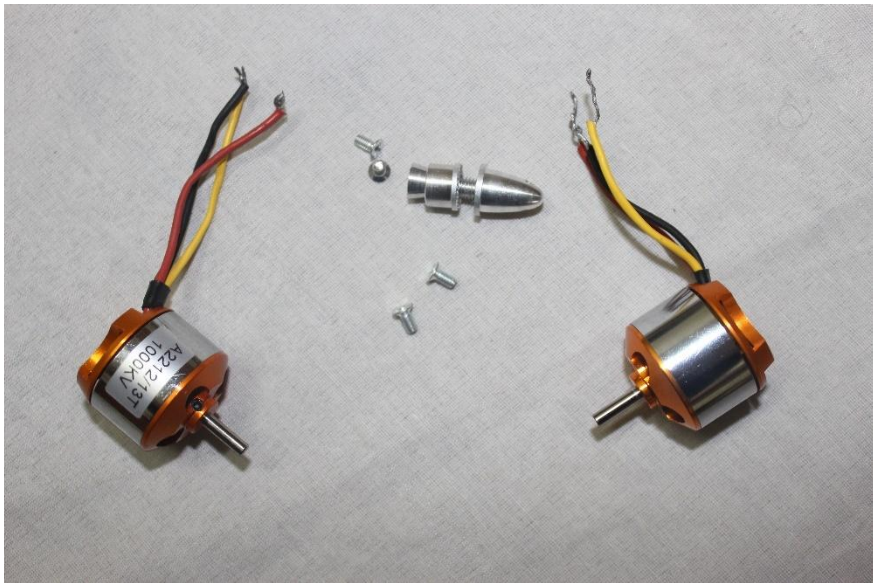


Figure 5: Motor.

Motors have a clear goal: to spin the propellers. whole power system depends on motor. The motors shall be powerful enough to spin the propellers, lift the quad-copter, and move the quad-copter at the required speed. Motors are rating by kilovolts (KV) Low kV means you can swing big propeller. To choose a motor we first need to how much weight we are planning to take, and then to work out the thrust required to lift the quad-copter. A general rule is that you should be able to provide twice as much thrust than the weight of the quad. If the thrust provided by the motors are too little, the quad will not respond well to your control, even has difficulties to take off. But if the thrust is too much, the quadcopter might become too agile and hard to control. (Oscar, How to choose Motor and Propeller for Quadcopter and Multicopter, 2013).

A rule of thumb is Required Thrust per motor = (Weight *2) / 4.

Our quad-copter weight with components might be around 1kg-1.2kg then by using the equation above.

Thrust per motor = (1000*2)/4=500g per motor.

we use one of the brushless motors A2212-t13. BRUSHLESS MOTOR are so much more efficient than brushed motors. Brushless motor provides more torque, which means the quad-copter can lift with less work from the motors. Brushless motors also do not produce heat at high speeds like brushed motors.

The motors are 1000 KV and 3.17 mm diameter shaft and the weight of each motor is 48g.

Kv:	1000 RPM/V
Max Efficiency:	80%
Max Efficiency Current:	4 - 10A (>75%)

No Load Current:	0.5A @10V
Resistance:	0.090 ohms
Max Current:	13A for 60S
Max Watts:	150W
Weight:	48 g / 1.7 oz
Size:	28 mm dia x 28 mm bell length
Shaft Diameter:	3.17 mm
Poles:	14

(A2212/13T TECHNICAL DATA, n.d.).

A2212-t13 1000KV thrust.

Propeller	Volt	Amp	Thrust	power	RPM
10*5	10.9V	13 A	28.3 O=802g	141 W	7650

RPM= Revolutions per minute.

2.3 PROPELLER

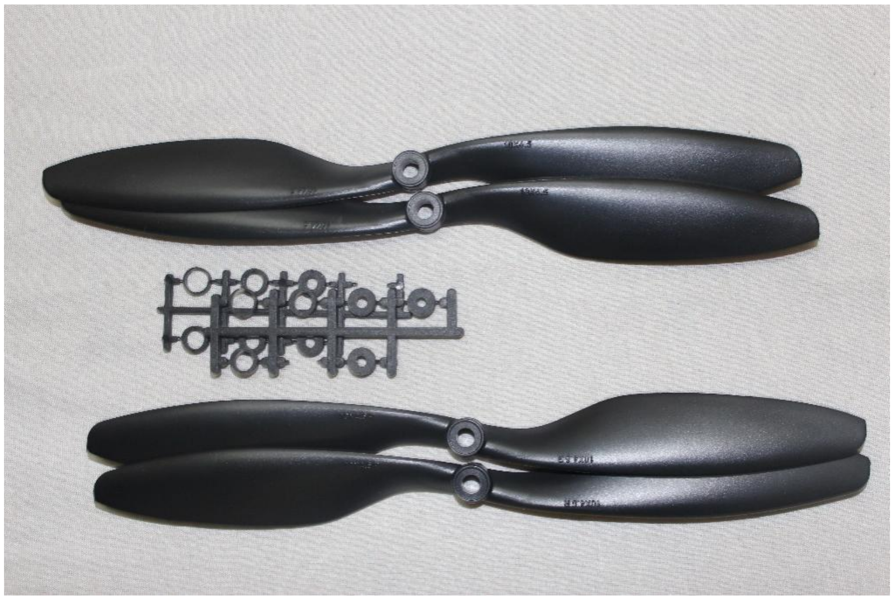


Figure 6: Propellers.

This is perhaps the most important part of it all. Someone say first you don’t choose the motor then suit it with a propeller. You must choose the propeller you want and then finding a motor that works well with it. Larger propellers give more thrust per revolution from the motor. A smaller prop is easier to speed up and slow down. A smaller prop requires a higher RPM motor because they must spin faster to generate equivalent lift. whereas a large prop takes a very long time to change speeds and have higher acceleration and result in stable slights. The propellers shall be large enough to provide adequate lift for the quadcopter. It’s necessary to match propellers with motors. We are use two clockwise(CW) and two counter-clockwise(CCW) propellers. Propellers are classified by length and pitch. Originally Plastic 10 x 4.5 propellers we were selected it because 10 x 4.5 propellers are ideal for quadcopters that are lifting heavy payloads and its strong and suitable with the motors that we were selected .10 x 4.5 the first number 10 inch is the diameter it’s the long of propeller, the second number 4.5 is the pitch of the propeller.

Diameter: Larger diameter propellers generate more thrust; smaller diameter propellers generate less thrust.

Pitch: This is the distance traveled for a single complete revolution. Lower the pitch, more the torque. Lower pitch propellers also have a lot less turbulence, can carry heavier loads and result in lower power consumption and longer flight times. On the other hand, higher pitch propellers are generally used mainly for acrobatics and racing.

If we have a motor runs at 11000 RPM but when you mount prop on it, RPM will be reduced. Here we will take example of two props 10x3.8 and 10x4.5. When you mount 10-inch diameter prop RPM of motor will be reduced to 7650 RPM (Revolutions Per Minute).

7650/60 = 127 Revolution Per Second

Our 1st prop has 3.8-inch pitch. Means per revolution it will travel 3.8inch. So

127 x 3.8 = 484 Inch/Sec = 12.3 m/sec

For 2nd prop, it has 4.5-inch pitch.

127 x 4.5 = 571 Inch/Sec = 14.5 m/sec

So we can say if we have 10x3.8 prop our quad will climb in the air at 12.3 m/sec, while with 10x4.5 prop climb rate will be increase to 14.5 m/sec.

(BEST QUADCOPTER MOTORS : CHOOSING RC QUADCOPTER OR MULTICOPTER MOTORS MADE EASY, n.d.).

2.4 BATTERY

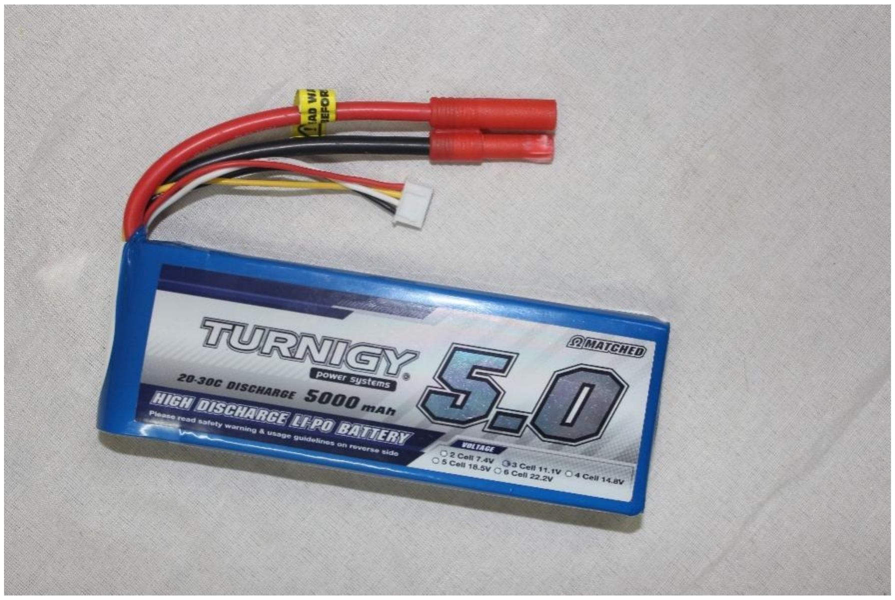


Figure 7: Battery TURNIGY 3 Cells.

Larger battery allows for longer flight time. It provides stable voltage, high current power to all of the components on the quad-copter. The battery's capacity is defined in mAh. mAh means milli-Ampere per hour. A battery with a 1000mAh capacity can deliver 1 Ampere (1000mA/1000) for 1 hour. The battery capacity, together with the battery's discharge rate will define its maximum current output (Ampère, A). The batteries we used on our current quadcopter is Turnigy 5000mAh 3S 20C Lipo.

Minimum Capacity: 5000mAh

Configuration: 3S1P / 11.1v / 3Cell

Constant Discharge: 20C

Peak Discharge (10sec): 30C

Pack Weight: 412g

If you remember our motor draws max 13amp. We are working on quadcopter and it has 4 motors, so all 4 motors will draw

4 x 13amp = 52Amp.

In order to know what the total current draw of our drone system, we can calculate it based on this simple formula:

Max Amp draw (A)= Battery capacity (Ah) x Discharge rate (C).

For example, we have a 5000mAh 3 cell Lipo battery with a 20C rating. To find the maximum continuous amp draw, we first convert the 5000mAh to 5.0Ah, and multiply that number by 20C, to give a total continuous output of (5.0 x 20) = 100A.

(Oscar, How to choose battery for Quadcopter, Tricopter and Hexacopter, 2014).

2.5 ELECTRONIC SPEED CONTROLLERS (ESC)

Figure 8: ESCs.

ESC stands for Electronic Speed Control. Either the flight controller or radio receiver sends signal to the ESC, and the ESC drives the brushless motor by providing the appropriate level of electrical power. The only requirements the ESC's have is that they must be able to output enough current to the motors, and they must operate at the required voltage. Ampere rating of ESC should be higher than max amp rating of motor. For example, the motor we selected draws maximum 13Amp so our ESC rating should be higher than 13amp. Here is simple formula:

ESC= (1.2-1.5) *max AMP rating of motor.

ESC=1.2*13=15.6A.

ESC=1.5*13=19.5A.

So you can select ESC between range of 16A to 20A or higher. We were chosen 30A ESC Brushless Motor.

(Oscar, How to choose ESC UBEC, BEC for Multicopters, 2015).

2.6 FLIGHT CONTROLLER (FC) AND SENSORS

A Flight controller is the Brain of your quadcopter or Multi-copter. it is a circuit board that reads sensors data and user commands, and makes adjustment to the motor speed, in order to keep the quadcopter balanced and in control. All multi-copter flight controllers have Gyro (Gyroscope: A sensor that measures orientation.) and Acc (Accelerometer). The quadcopter has various sensors that read data of the environment and send it to the flight controller for analysis and flight. We also had to provide a way to control the Quadcopter from the ground. We came up with three different options for wireless communication system for our quadcopter. There are three main options to choose from which include a radio transmitter and receiver, a Wi-Fi module, or a Bluetooth module. For our purposes we would need at least six channels to control our quadcopter. The six channels correspond to throttle, roll, yaw, pitch, one channel to switch between acrobatic mode and stable mode, and one or more channels for our auto-commands. We decided to choose the Bluetooth HC-05 receiver and use our smartphone as transmitter because it had the required amount of channels needed for our quadcopter purposes. The transmitter and receiver also had the greatest range of transmission was also easy to interface with our control but it need to be programmed, large amounts of programming needed. This Module work well with Arduino and other Microcomputers. (Oscar, Choose Flight Controller for Quadcopter Hexacopter, 2014).

2.6.1 HC-05

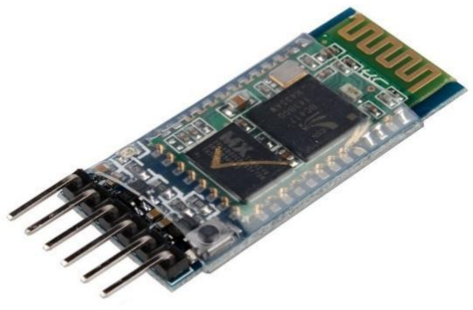


Figure 9: HC-05 Bluetooth.

- Bluetooth protocol: Bluetooth Specification v2.0.
- Frequency: 2.4GHz ISM band.
- Speed: Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps.
- Security: Authentication and encryption.
- Dimension: 26.9mm x 13mm x 2.2 mm.
- Range is approximately 10 Meters (30 feet).

(Serial-Port Bluetooth Module (Master/Slave), n.d.)

2.6.2 HC-SR04

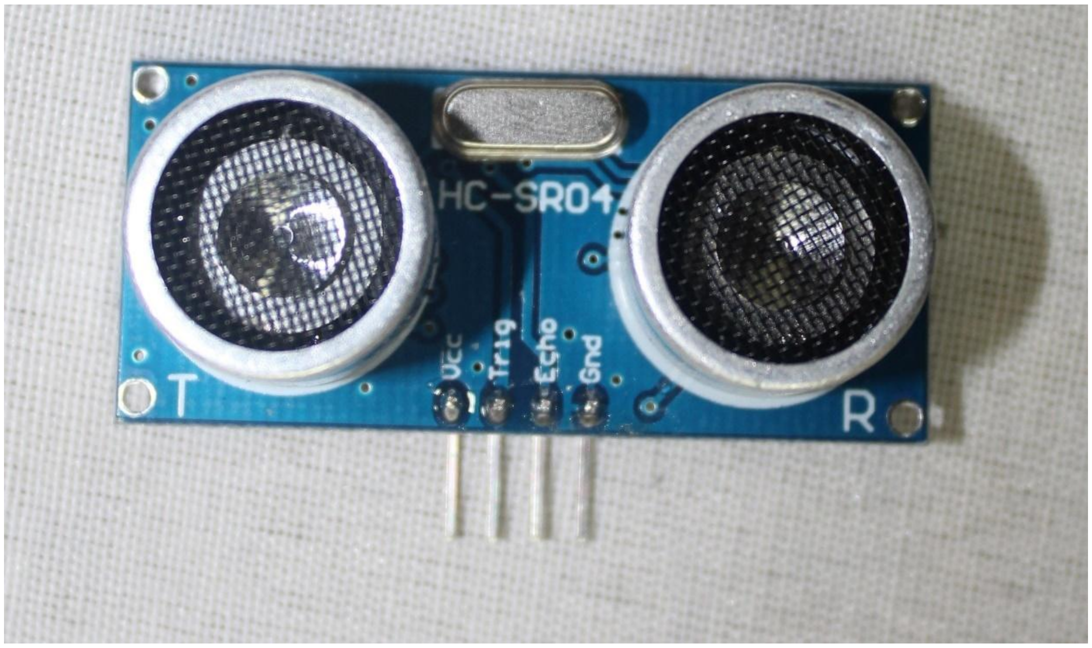


Figure 10: HC-SR04 Ultrasonic.

We bought Ultrasonic Distance Sensor Distance HC-SR04 to make our quad-copter to avoid obstacles in indoor flights and a step closer to automating it.

The HC-SR04 Ultrasonic Sensor is a very affordable proximity/distance sensor that has been used mainly for object avoidance in various robotics projects. It essentially gives your Arduino eyes / special awareness and can prevent your robot from crashing or falling off a table. It has also been used in turret applications, water level sensing, and even as a parking sensor. This simple project will use the HC-SR04 sensor with an Arduino and a Processing sketch to provide a neat little interactive display on your computer screen.

Power: 5V DC.

Quiescent Current: <2mA.

Ranging Distance: 2cm – 500 cm/1" - 16ft.

LxWxH: 45 x 20 x 15mm.

Weight: 8.5g.

CHAPTER THREE

HARDWARE DESIGN

CHAPTER THREE

HARDWARE DESIGN

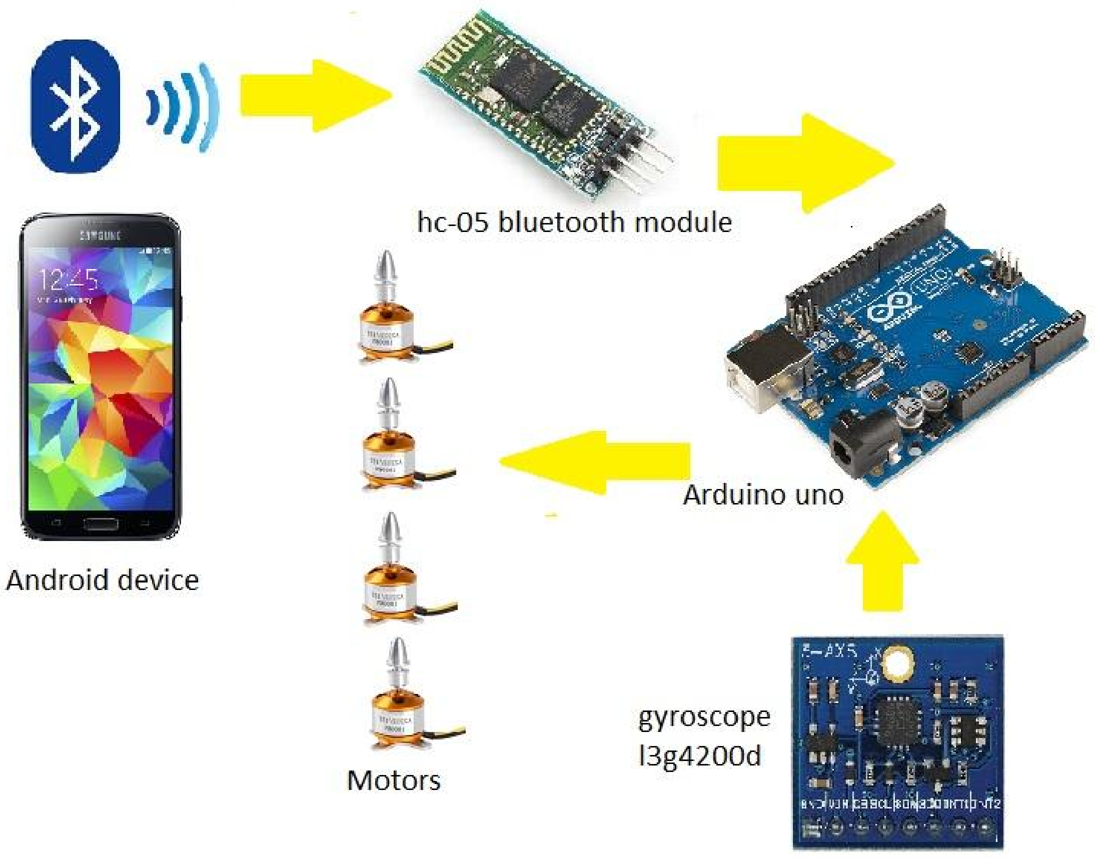


Figure 11: The system structure.

Quadcopter is a very sensitive drone; we need to design it very carefully to reduce shaking during the flight. And the components must be compatible with each other that we discussed in chapter 2.

3.1 JOIN COMPONENTS TO THE FRAME

First of all, we join the ESCs with the frame.

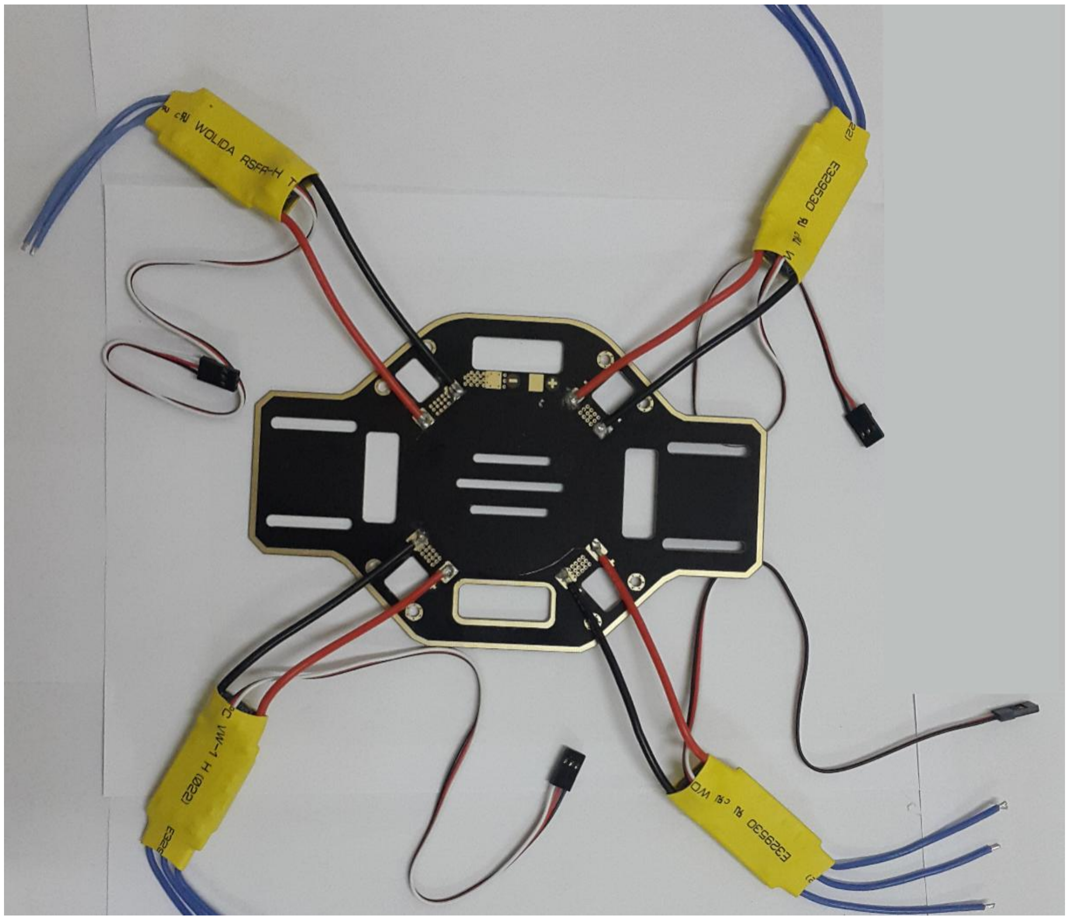


Figure 12: Connect ESCs to frame.

And then join the frame parts together.

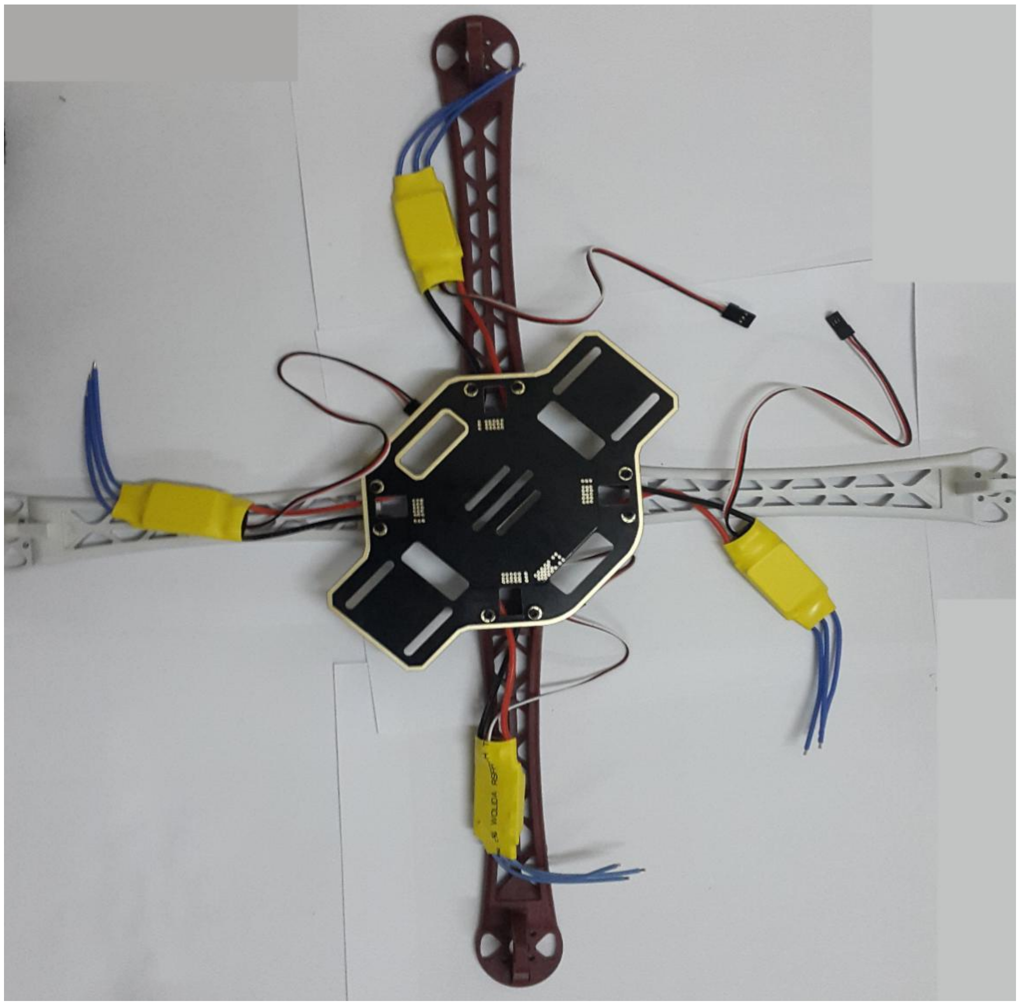


Figure 13: join frame parts.

After that we need to connect the ESCs with the motors.

But before we do that we need to discuss about some things, which are the rotation of the motors and the frame shape.

We have two types of the shape of the frame; X-Shape and + Shape.

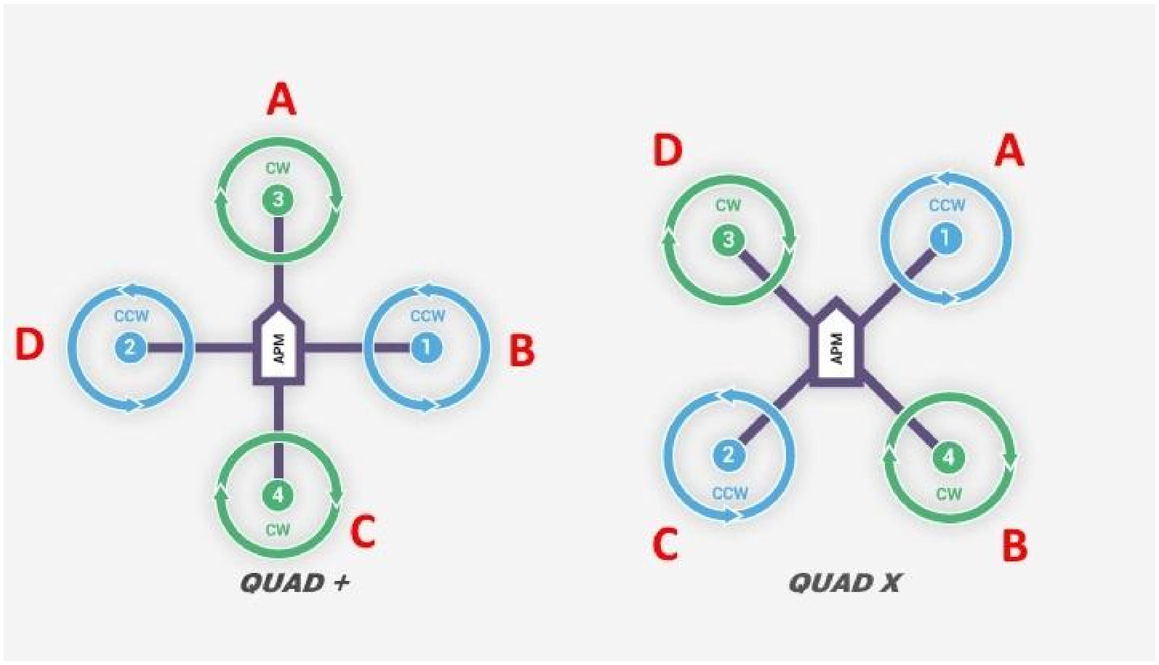


Figure 14: + shape, X-Shape (Team, n.d.).

We use the X-Shape for designing our quadcopter. The first motor will be in (A) position, the second will be in (B) and proceed clockwise.

Motors must be in 2-pair, one of them clockwise and the other anti-clockwise.

ESCs have three cables, the middle one is the received data and two others are power cables, so as motors. So if we swipe the power cables we can control the rotation of the motor.

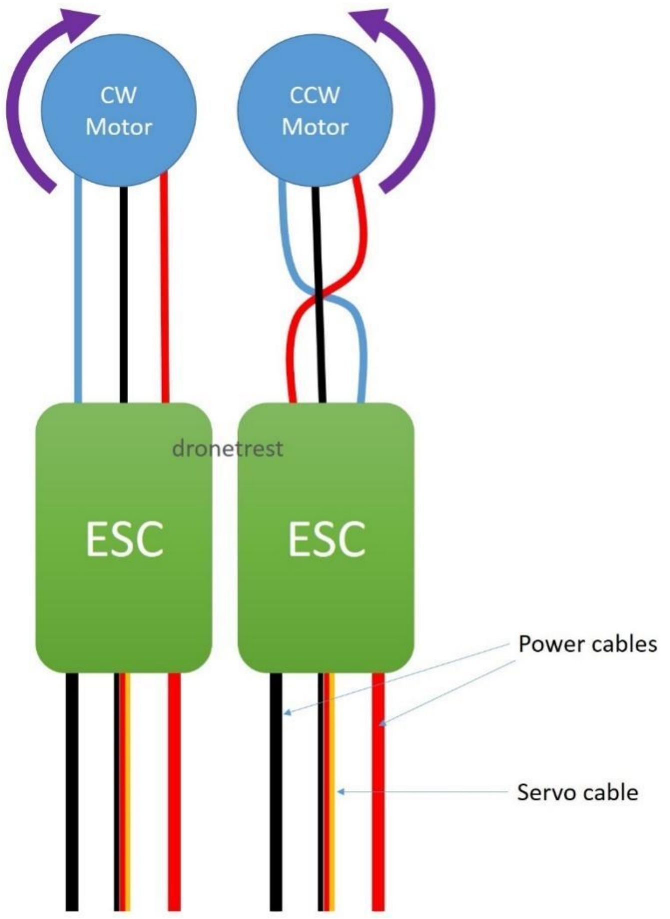


Figure 15: Connect motors to ESCs clockwise and anti-clockwise (Sam, 2015).

Also propellers need to install in clockwise and counter clockwise.

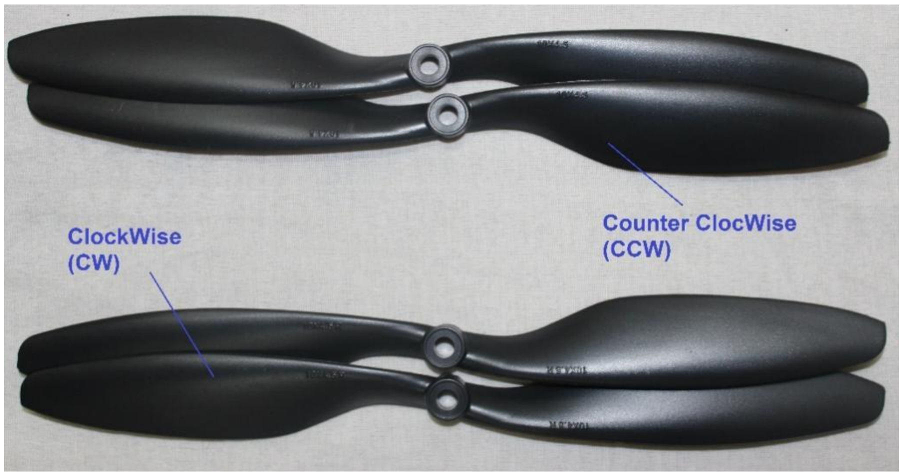


Figure 16:Propellers clockwise and anti-clockwise.

3.1.1 STABILITY

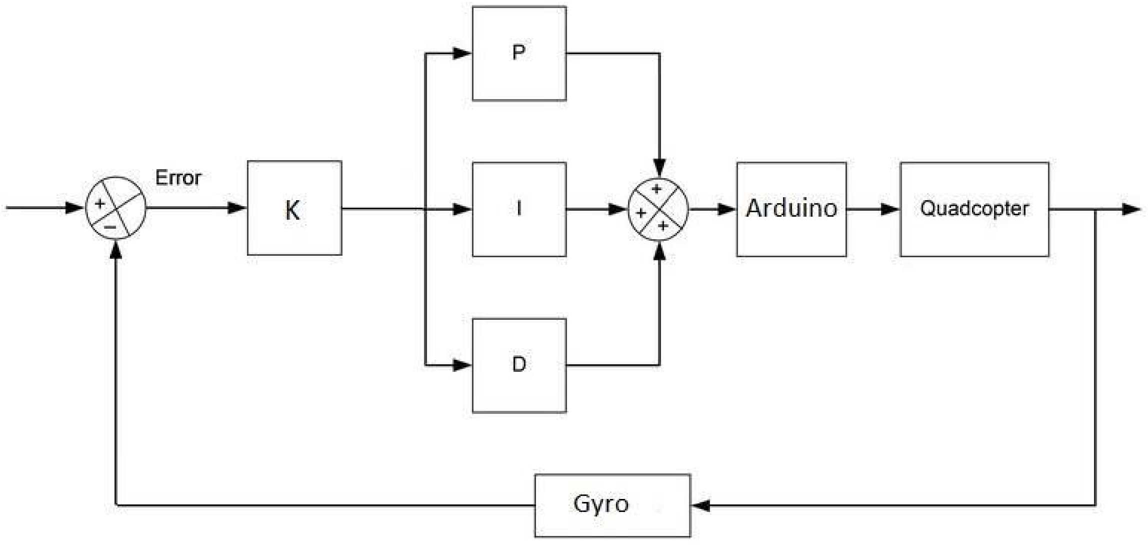


Figure 17: PID control block diagram.

What is PID?

PID (proportional-integral-derivative) is a closed-loop control system that try to get the actual result closer to the desired result by adjusting the input. Quadcopters or multi-copters use PID controller to achieve stability.

There are 3 algorithms in a PID controller. P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change. These controller algorithms are translated into software code lines. (Oscar, Quadcopter PID Explained and Tuning, 2013)

The effect of each parameter:

Proportional Gain coefficient – Quadcopter can fly relatively stable without other parameters but this one. This coefficient determines which is more important, human control or the values measured by the gyroscopes.

Integral Gain coefficient – this coefficient can increase the precision of the angular position... This term is especially useful with irregular wind, and ground effect (turbulence from motors).

Derivative Gain coefficient – this coefficient allows the quadcopter to reach more quickly the desired attitude. It is decrease control action fast when the error is decreasing fast. (Oscar, Quadcopter PID Explained and Tuning, 2013).

3.2 JOIN COMPONENTS TO THE ARDUINO

Arduino has not much volts (VCC) and grounds (GND), we use Arduino-UNO which it has 3 GND and one 5V, but each ESC has its own VCC and GND, and the Gyro also need to have ground and voltage, so does Bluetooth and other sensors.

So we build a design for our project to solve this. By connecting all grounds and volts together, and join all grounds with the frame plate that is a conductor plate, same thing for the volts. And join the volt and ground of the Arduino with the same conductor plates. Arduino needs a power to be on for sharing data and turning other components on. The battery that we use for our quadcopter project is designed in somehow it can provide power to Arduino in the same time that it provides power to the ESCs. So we can connect the voltages to (Vin) in the Arduino instead of (5V).

We almost complete many things, now we need to connect components data to Arduino pins.

L3G400D Gyroscope:

serial clock (SCL) in the Gyro -> Analog (A5) input pin in the Arduino.

serial data (SDA) in the Gyro -> Analog (A4) input pin in the Arduino.

HC-05 Bluetooth:

Receiver (RX) in the Bluetooth -> (pin 0) Transmitter TX in the Arduino.

Transmitter (TX) in the Bluetooth -> (pin 1) Receiver RX in the Arduino.

ECSs:

ESC 1 -> PWM (Pin 3).

ESC 2 -> PWM (Pin 5).

ESC 3 -> PWM (Pin 6).

ESC 4 -> PWM (Pin 9).

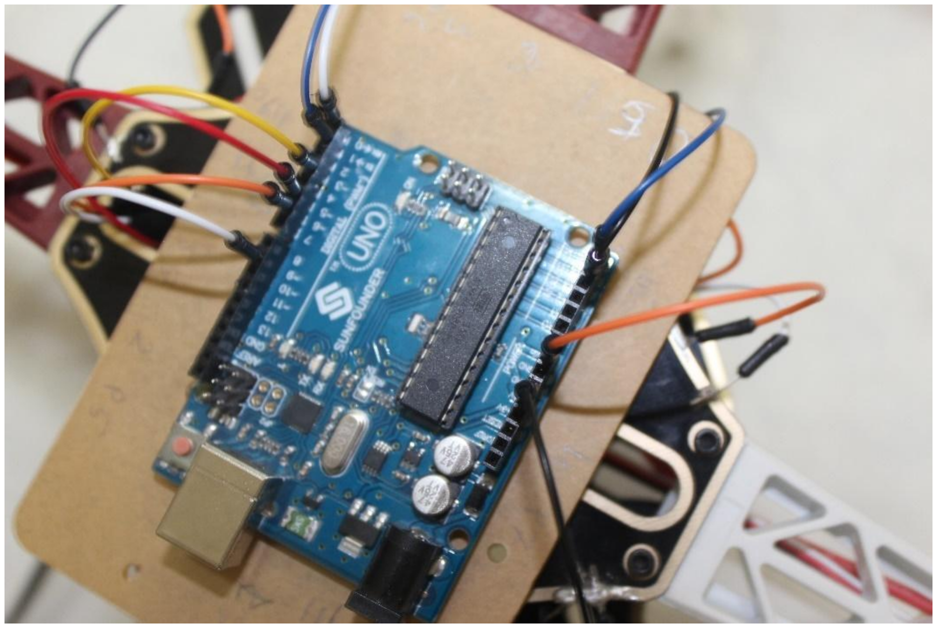


Figure 18: Connect data signals to Arduino pins

HC-SR04:

(Trig) in the Bluetooth -> Arduino pin 11.

(Echo) in the Bluetooth -> Arduino pin 12.

CHAPTER FOUR

SOFTWARE DESIGN

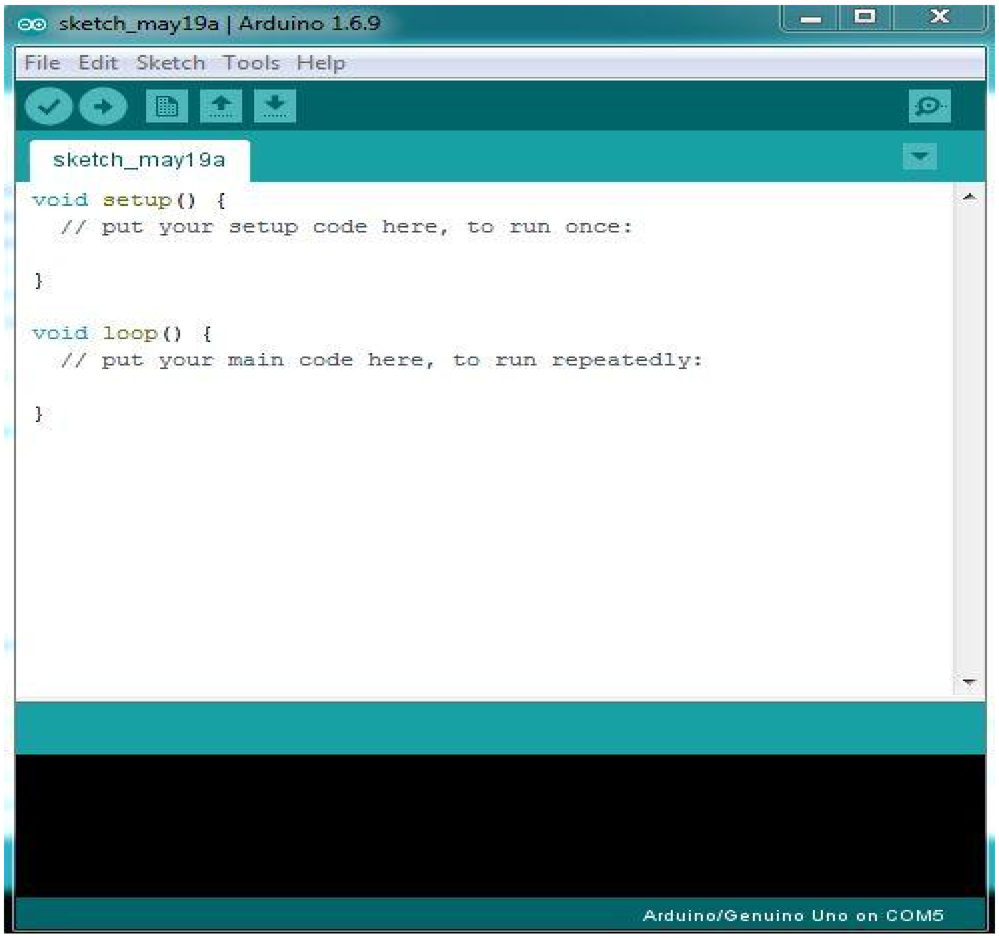
CHAPTER FOUR

SOFTWARE DESIGN

In this chapter we describe the software components that we used in our project. The Arduino software and Android application.

4.1 ARDUINO IDE

For our project we will be focusing on programming in the Arduino IDE. The Arduino IDE is a development environment that uses a simple user interface for adding and editing the Arduino codes, which is a based on the C++ programming language.


sketch_may19a | Arduino 1.6.9
File Edit Sketch Tools Help
sketch_may19a
Arduino/Genuino Uno on COM5

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 19: Arduino IDE.

4.2 SOFTWARE WALKTHROUGH

The header file <Wire.h> Include the Wire.h library to communicate with the gyro.

The header file <Servo.h> Include the servo library to controlling the motors.

The Arduino board has two main functions: Setup() and loop().

On startup, the Setup() function does the following tasks:

- Initialize serial communication for debugging. { Serial.begin(9600); }.
- Initialize and configure motors. { myservo.attach(pinNumber); }.
- Configure the input and output pins and internal timers for PPM input and output. { pinMode(pinNumber, OUTPUT); pinMode(pinNumber, INPUT); }.

Next, it enters the main loop, loop(), where it waits for commands from the Android device. These commands include movement orders and sensor readings. When a command is read, it extracts the data, if any, and performs the orders.

These are the other functions and their purpose:

- Serial.read() Reads incoming serial data from Bluetooth.
- Serial.parseInt() returns the first valid integer number from the serial buffer.
- myservo.writeMicroseconds() Writes a value in microseconds (μs) to the servo motor.
- The gyro_signalen() method is for reading the gyro.

4.3 ANDROID

Android is Google's phone and tablet operating system. Developing for Android is free. The software development tools are free and there are no fees associated with distributing your app. You can also deploy directly without having to use Google's Market. (MELLOULI, 2010).

ECLIPSE IDE

The main application for Android Smart Phone is coded in the environment of Eclipse Indigo IDE and compiled as a standard android executable that runs on platforms above API level 8 (Gingerbread or higher). (MELLOULI, 2010).

ANDROID SDK

The development kit used to program on Eclipse Indigo IDE is the ANDROID SDK developed by Google, Inc.

Issues

There are several issues that must be overcome before the Android device can successfully transmit and receive data via Bluetooth. First, the Android must determine if it supports Bluetooth, and if it does, if Bluetooth is turned on. Then, it must pair and connect with the Bluetooth module on the Arduino. Finally, the Android must actually send and receive data. Receiving data will be particularly troublesome. (MELLOULI, 2010).

Steps

To begin to program for Android we needed some basics, because some elements are very different, even if programming an application in Android uses the Java language, therefore, an object oriented language.

Activity: An activity is a user interface that allows the user to interact with the screen, to perform actions.

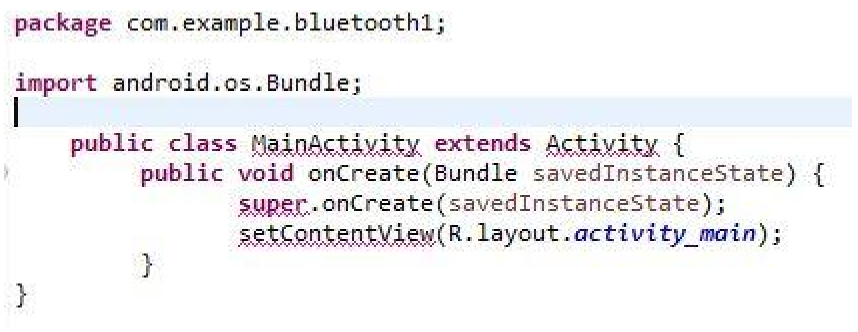
View: a view is the basic building block for user interface components.

Xml: Xml means Extensible Markup Language. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses. The goal of using Android's XML vocabulary, is to quickly design UI layouts and the screen elements.

Intent: An activity that can start another one.

Android Manifest: AndroidManifest.xml file is necessary for all android applications and must have this name in its root directory. In the manifest you can find essential information about the application for the Android system, information that the system must have before it can run any of the application's code.

With all these elements, an application can be created. (MELLOULI, 2010).



```
package com.example.bluetooth1;

import android.os.Bundle;

    public class MainActivity extends Activity {
         public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
         }
}
```

Source 1 starting project.

The first programming step is to create a new Android Application Project in Eclipse. Doing so will generate code similar to that in Source 1. (Wirsing, 2014).

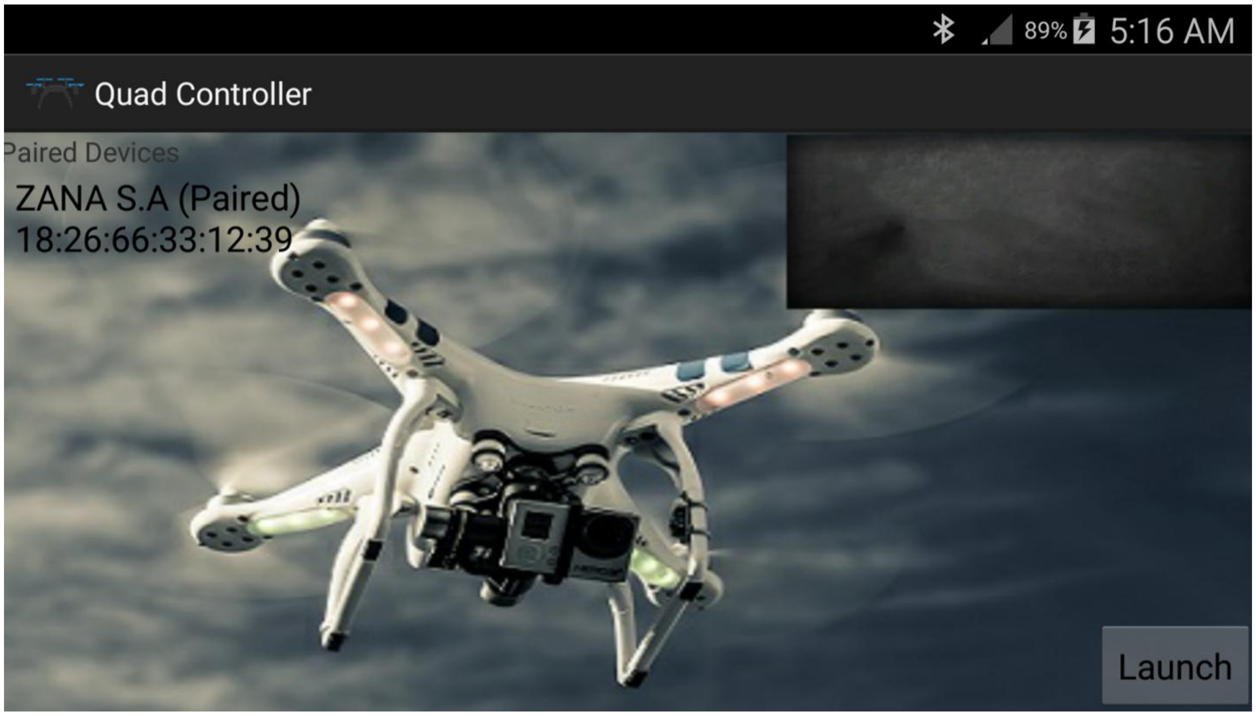


Figure 20: App interface.

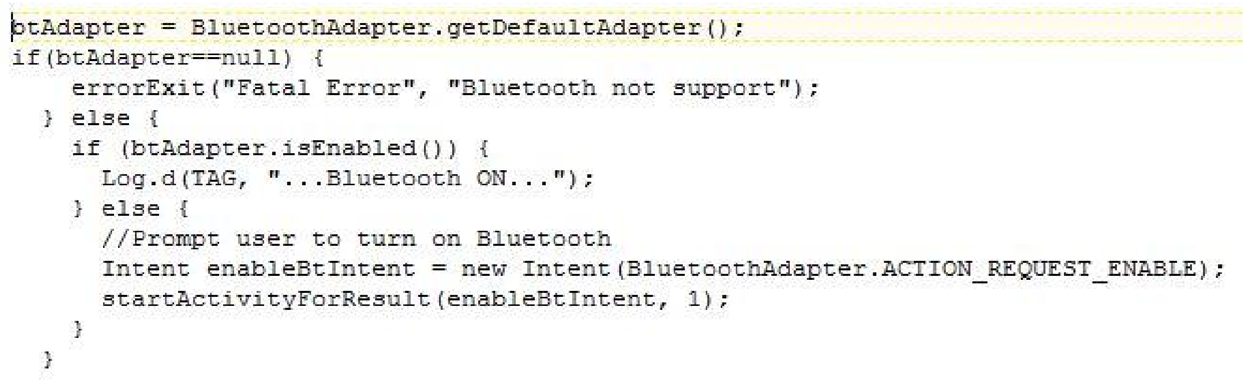
This application there are two different screens, and so two different activities, but it is a good training to try how to communicate between the activities, how to display elements (TextView, Buttons ...) on the screen and how to interact with them. (Wirsing, 2014)

In the first frame we have a (Launch) button to go to next frame, and there is a list that both have been declared in the Main XML and the MainActivity code for the first activity (frame). The Launch button does not work until the program find a Bluetooth device and pair to it. We will have an Intent as explained previously. (MELLOULI, 2010)

The aim of the list is to showing the discover Bluetooth and connecting with it.

The first thing the program should do is determine if the Android device supports Bluetooth. To do this, we create a BluetoothAdapter object using the function getDefaultAdapter(). If this returns null, then the Android device does not support Bluetooth. Source 2 shows how to do this. We will add this code to OnCreate() method. (Wirsing, 2014)

If getDefaultAdapter does not return null, then the Android supports Bluetooth. The next step is to determine if Bluetooth is enabled, and if it is not enabled, to enable it. Source 2 accomplishes this task. (Wirsing, 2014).



```
btAdapter = BluetoothAdapter.getDefaultAdapter();
if(btAdapter==null) {
    errorExit("Fatal Error", "Bluetooth not support");
  } else {
    if (btAdapter.isEnabled()) {
      Log.d(TAG, "...Bluetooth ON...");
    } else {
      //Prompt user to turn on Bluetooth
      Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(enableBtIntent, 1);
    }
  }
```

Source 2 Determine if Android supports Bluetooth and enable it.

Next, the program has to retrieve the actual Bluetooth device it will communicate with, in this case the Arduino’s Bluetooth module. The BluetoothAdapter’s getBondedDevices() function will do this. This function puts all of the Android’s currently-paired devices into a storage structure called a “Set”. Since only the Arduino’s Bluetooth module is paired with the Android (which was done at the beginning of these instructions), only this device will be in the Set. (Wirsing, 2014).

```
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
        long arg3) {
    // TODO Auto-generated method stub

    if(btAdapter.isDiscovering()){
        btAdapter.cancelDiscovery();
    }
    if(listAdapter.getItem(arg2).contains("Paired")){


        BluetoothDevice selectedDevice = devices.get(arg2);
        ConnectThread connect = new ConnectThread(selectedDevice);
        connect.start();
        Log.i(tag, "in click listener");
    }
    else{
        Toast.makeText(getApplicationContext(), "device is not paired", 0).show();
    }
}
```

Source 3: List of devices that turning on and paired devices will show in the app.

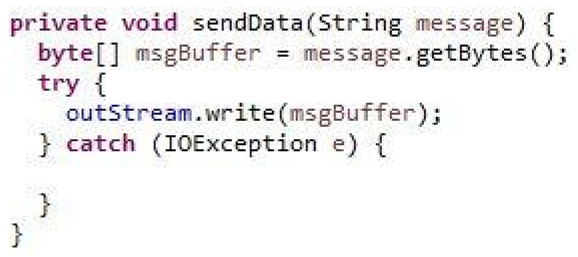
At this point, the Android has the Arduino's Bluetooth module stored in a BluetoothDevice object. The next objective is to form the connection between the Android and the Arduino. This work should take place in separate thread. This is because forming a connection can block a thread for a significant amount of time. Up until now, all of the program's code has been written in the main thread, or "user interface thread" (UI thread). The UI thread should never be blocked. Therefore, create a new thread class where the connection will form. Source 6 shows the code to accomplish this. Add it as an inner class of the main class. (Wirsing, 2014).

```
private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws IOException {
    if(Build.VERSION.SDK_INT >= 10){
        try {
            final Method  m = device.getClass().getMethod("createInsecureRfcommSocketToServiceRecord",
                    new Class[] { UUID.class });
            return (BluetoothSocket) m.invoke(device, MY_UUID);
        } catch (Exception e) {
            Log.e(TAG, "Could not create Insecure RFComm Connection",e);
        }
    }
    return  device.createRfcommSocketToServiceRecord(MY_UUID);
}
```

Source 4: Create a Bluetooth Socket if exists.

This thread requires a BluetoothDevice as a parameter and uses it to create a BluetoothSocket. This socket is what Bluetooth uses to transfer data between devices. The UUID used in Source 4 tells the socket that data will be transferred serially, which means one byte at a time. (Wirsing, 2014).

The code will now connect the Arduino's Bluetooth module with the Android. The last objective is to send and receive data using this connection.



```
private void sendData(String message) {
  byte[] msgBuffer = message.getBytes();
  try {
    outStream.write(msgBuffer);
  } catch (IOException e) {

  }
}
```

Source 5: Send data method in Android Eclipse.

sendData() method used for sending data to the Arduino. (Wirsing, 2014).

Second frame called bluework

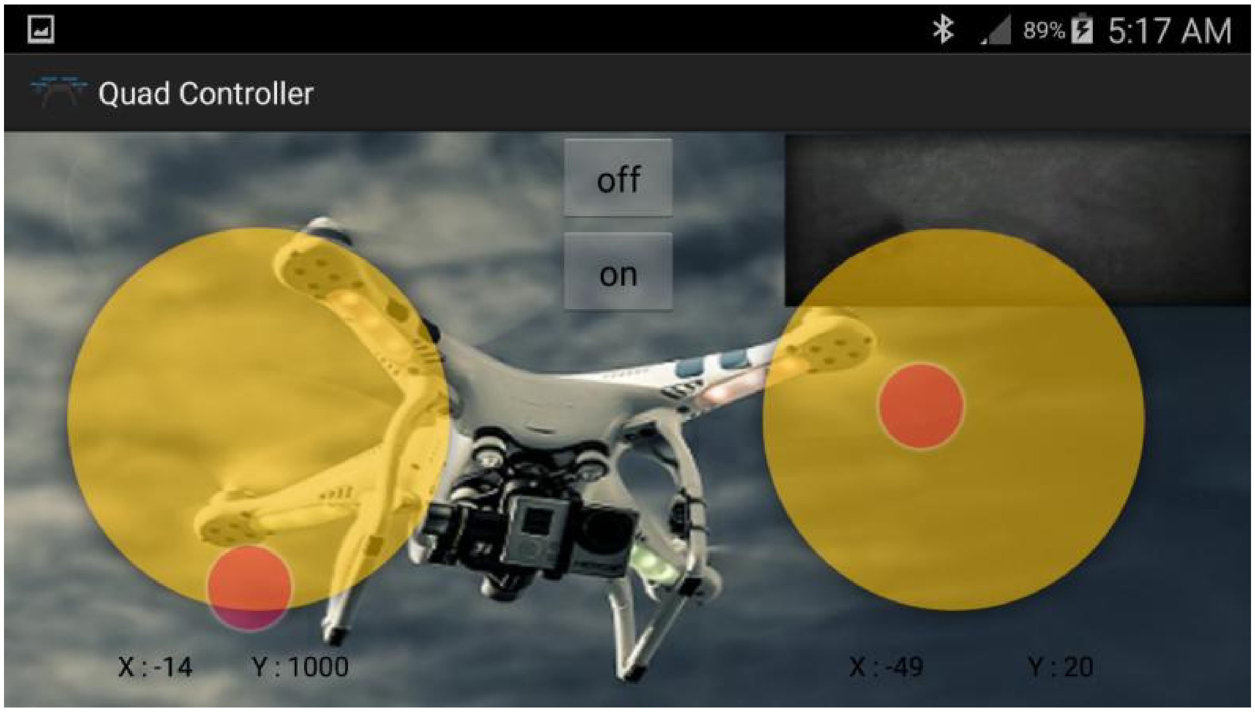


Figure 21: App working area.

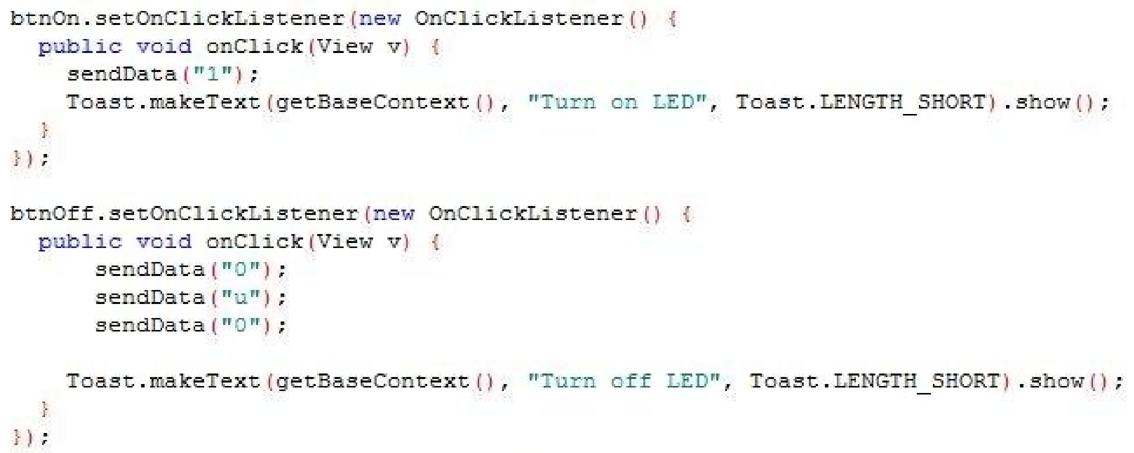
For the second frame we have buttons, (off) button sends (0) μs for turning off the motors, (on) button is to turning on the LED and sending data to the motors.

There is two JoySticks that send data every touching, and four labels that they have been declared in the (work) XML and the (bluework) code for the second activity (frame). that JoyStick on the right is the first JoyStick called (JoyS), it sends row and pitch data to the quadcopter, if we sending positive Y number, the quadcopter moves forwarding, and sending positive X will make the quad to move to right.

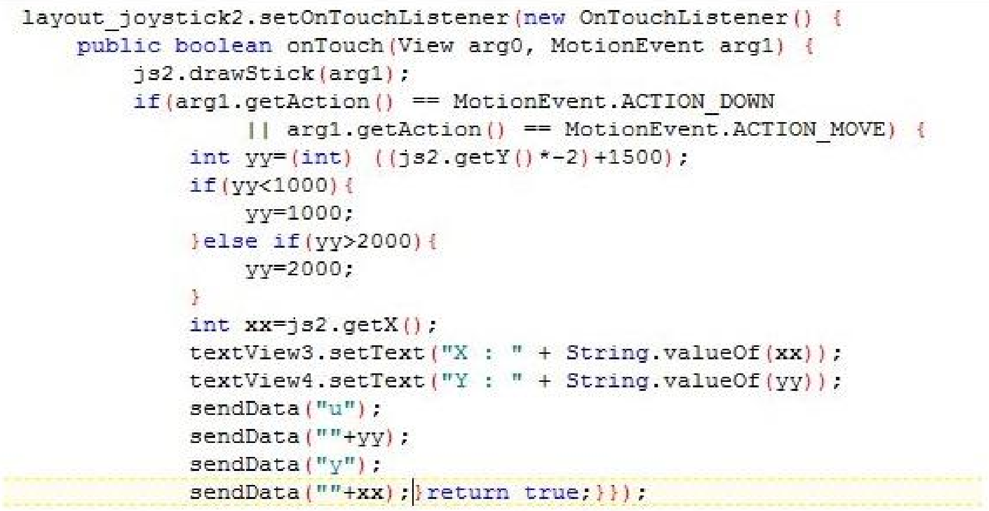
The second JoyStick called (JoyS2), it sends throttle and yaw data to the quad, throttle is beginning from 1000, so it sends 1000 μs, because the motors will work in 1200 µs. And the maximum throttle is 2000 µs. So sending greater number of Y cause the quadcopter to flight.

Sending positive X numbers move the quadcopter clockwise.

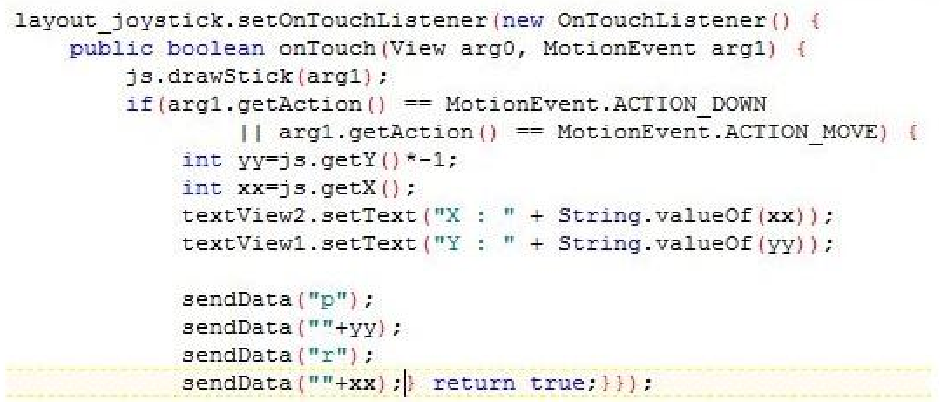
All these works shown in source 6.



```
btnOn.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
    sendData("1");
    Toast.makeText(getBaseContext(), "Turn on LED", Toast.LENGTH_SHORT).show();
  }
});

btnOff.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
      sendData("0");
      sendData("u");
      sendData("0");

    Toast.makeText(getBaseContext(), "Turn off LED", Toast.LENGTH_SHORT).show();
  }
});
```



```
layout_joystick2.setOnTouchListener(new OnTouchListener() {
    public boolean onTouch(View arg0, MotionEvent arg1) {
        js2.drawStick(arg1);
        if(arg1.getAction() == MotionEvent.ACTION_DOWN
                || arg1.getAction() == MotionEvent.ACTION_MOVE) {
            int yy=(int) ((js2.getY()*-2)+1500);
            if(yy<1000){
                yy=1000;
            }else if(yy>2000){
                yy=2000;
            }
            int xx=js2.getX();
            textView3.setText("X : " + String.valueOf(xx));
            textView4.setText("Y : " + String.valueOf(yy));
            sendData("u");
            sendData(""+yy);
            sendData("y");
            sendData(""+xx);}return true;}});
```

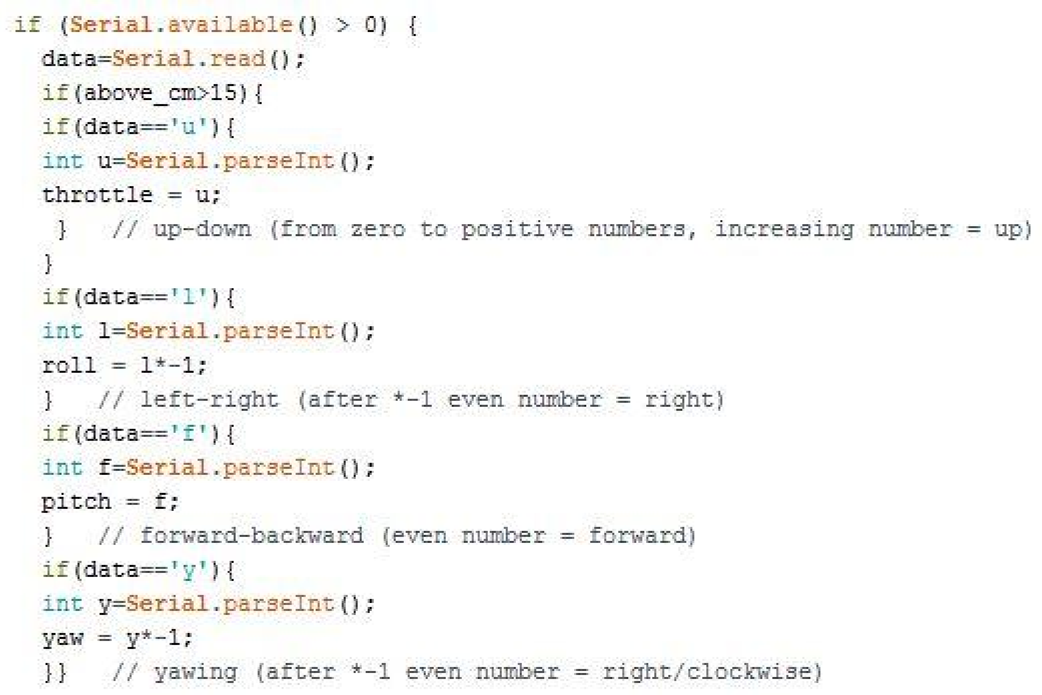


```
layout_joystick.setOnTouchListener(new OnTouchListener() {
    public boolean onTouch(View arg0, MotionEvent arg1) {
        js.drawStick(arg1);
        if(arg1.getAction() == MotionEvent.ACTION_DOWN
                || arg1.getAction() == MotionEvent.ACTION_MOVE) {
            int yy=js.getY()*-1;
            int xx=js.getX();
            textView2.setText("X : " + String.valueOf(xx));
            textView1.setText("Y : " + String.valueOf(yy));

            sendData("p");
            sendData(""+yy);
            sendData("r");
            sendData(""+xx);} return true;}});
```

Source 6: Setup Joysticks to send data.

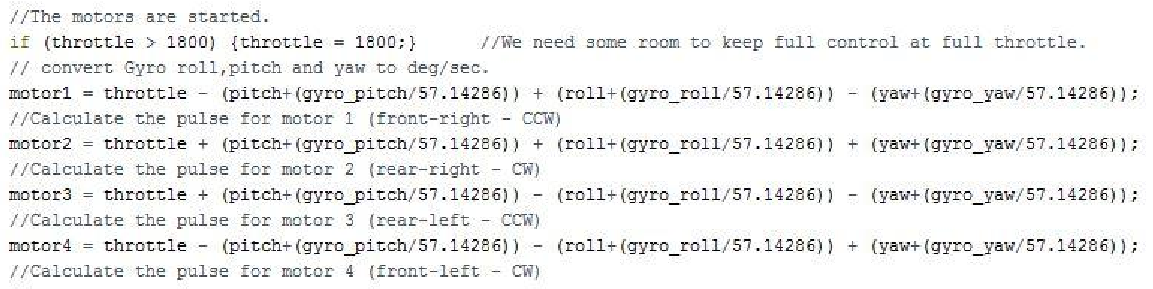
These “u”,” y”,” p”, and “r” strings controlled in the Arduino program.

How these data send and how Arduino work on it?

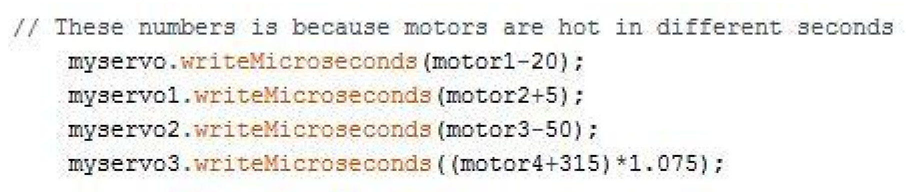


```
if (Serial.available() > 0) {
  data=Serial.read();
  if(above_cm>15){
  if(data=='u'){
  int u=Serial.parseInt();
  throttle = u;
   }   // up-down (from zero to positive numbers, increasing number = up)
  }
  if(data=='l'){
  int l=Serial.parseInt();
  roll = l*-1;
  }   // left-right (after *-1 even number = right)
  if(data=='f'){
  int f=Serial.parseInt();
  pitch = f;
  }   // forward-backward (even number = forward)
  if(data=='y'){
  int y=Serial.parseInt();
  yaw = y*-1;
  }}   // yawing (after *-1 even number = right/clockwise)
```

Source 7: Getting Bluetooth data from the android app.



```
//The motors are started.
if (throttle > 1800) {throttle = 1800;}      //We need some room to keep full control at full throttle.
// convert Gyro roll,pitch and yaw to deg/sec.
motor1 = throttle - (pitch+(gyro_pitch/57.14286)) + (roll+(gyro_roll/57.14286)) - (yaw+(gyro_yaw/57.14286));
//Calculate the pulse for motor 1 (front-right - CCW)
motor2 = throttle + (pitch+(gyro_pitch/57.14286)) + (roll+(gyro_roll/57.14286)) + (yaw+(gyro_yaw/57.14286));
//Calculate the pulse for motor 2 (rear-right - CW)
motor3 = throttle + (pitch+(gyro_pitch/57.14286)) - (roll+(gyro_roll/57.14286)) - (yaw+(gyro_yaw/57.14286));
//Calculate the pulse for motor 3 (rear-left - CCW)
motor4 = throttle - (pitch+(gyro_pitch/57.14286)) - (roll+(gyro_roll/57.14286)) + (yaw+(gyro_yaw/57.14286));
//Calculate the pulse for motor 4 (front-left - CW)
```



```
// These numbers is because motors are hot in different seconds
   myservo.writeMicroseconds(motor1-20);
   myservo1.writeMicroseconds(motor2+5);
   myservo2.writeMicroseconds(motor3-50);
   myservo3.writeMicroseconds((motor4+315)*1.075);
```

Source 8: Main pitch, yaw, and roll calculations and send signals to motors.

As shown in the source 7, the Arduino get the received Bluetooth data and calculate it for throttle, pitch, roll and yaw. Serial.available() check if the HC-SR04 Bluetooth is connected then work.

If the Bluetooth received "u" character and a number, like (u1200), the throttle will be equal to the number and send it to the motors, if it received "l" and a number, the quadcopter will pitch, and so on.

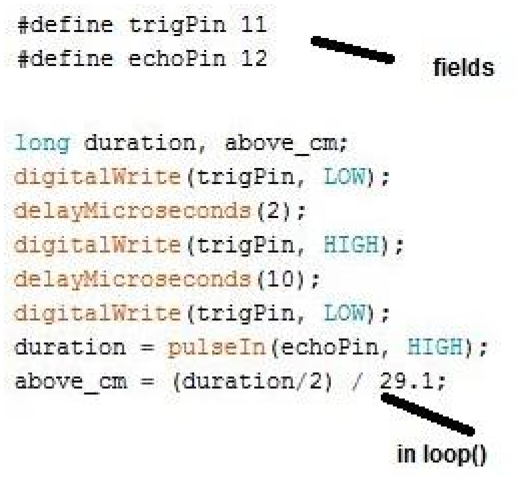
source 8; it is the most important part, we calculate throttle, pitch, roll and yaw for every motors, as we discussed before, we designed our quadcopter in X-Shape, so:

- if we want to roll right, we need to increase motor 3 and motor 4, and decrease motor 1 and motor 2, with the same value of roll. the equation control evens and odds. And vice versa for rolling left.
- If we want to pitch forward, we need to increase motor 2 and motor 3, and decrease motor 1 and motor 4, with the same value of pitch. the equation control evens and odds. And vice versa for pitch backward.
- If we want to yaw clockwise, we need to increase motor 2 and motor 4, and decrease motor 1 and motor 3, with the same value of yaw. the equation control evens and odds. And vice versa for yawing anti-clockwise.
- Gyro roll, yaw, and pitch is to stability.

All these works need to be in loop() method.

We have some other works need to be done in loop() method.

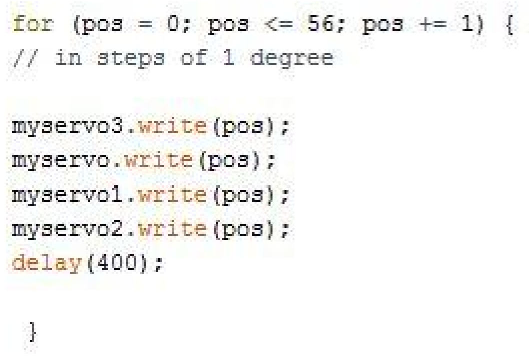
In the source 7, (above_cm) is the data that we get in HC-SR04 sensor, it checks the top of the quadcopter, if it gets 15cm, sending throttle in the app will be disable. Source 9 shows the code.



```
#define trigPin 11
#define echoPin 12

long duration, above_cm;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
above_cm = (duration/2) / 29.1;
```

Source 9: HC-SR04 Bluetooth distance calculation.

In the setup() method we have a routine for-loop shown in source 10



```
for (pos = 0; pos <= 56; pos += 1) {
// in steps of 1 degree

myservo3.write(pos);
myservo.write(pos);
myservo1.write(pos);
myservo2.write(pos);
delay(400);

 }
```

Source 10: For-loop to initialize motors.

In the loop we make all motors to go to 56 degrees from zero, this is for the first time for motors to be ready.

We attached motors with PWM ports; Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED. (Hirzel, n.d.)

4.4 GYRO STABILIZATION

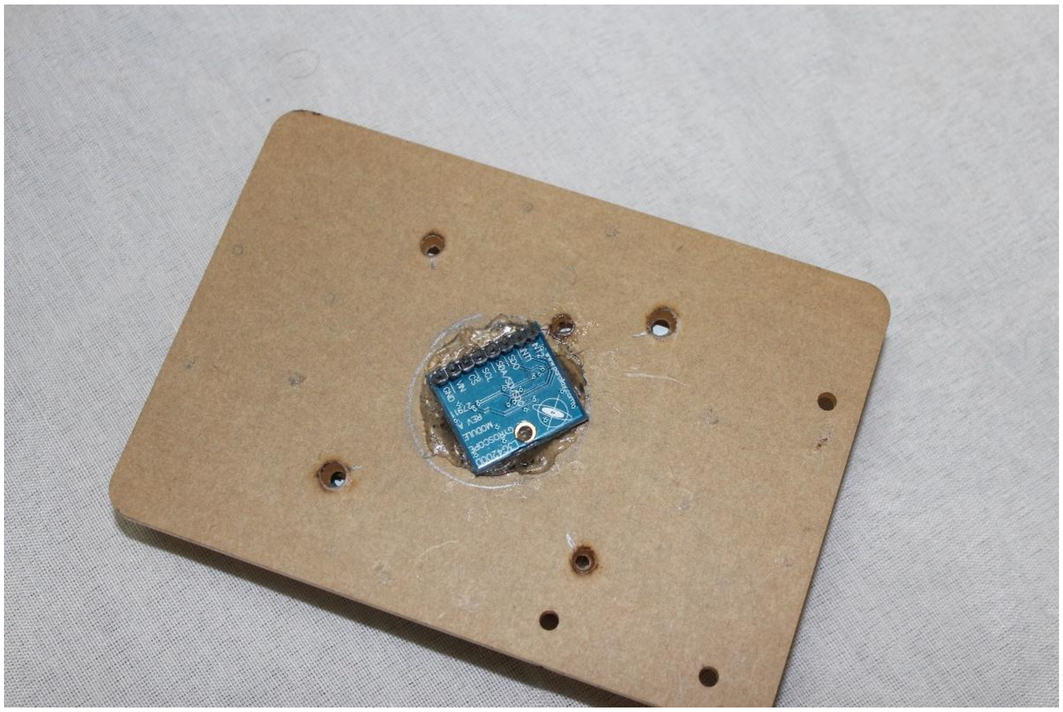
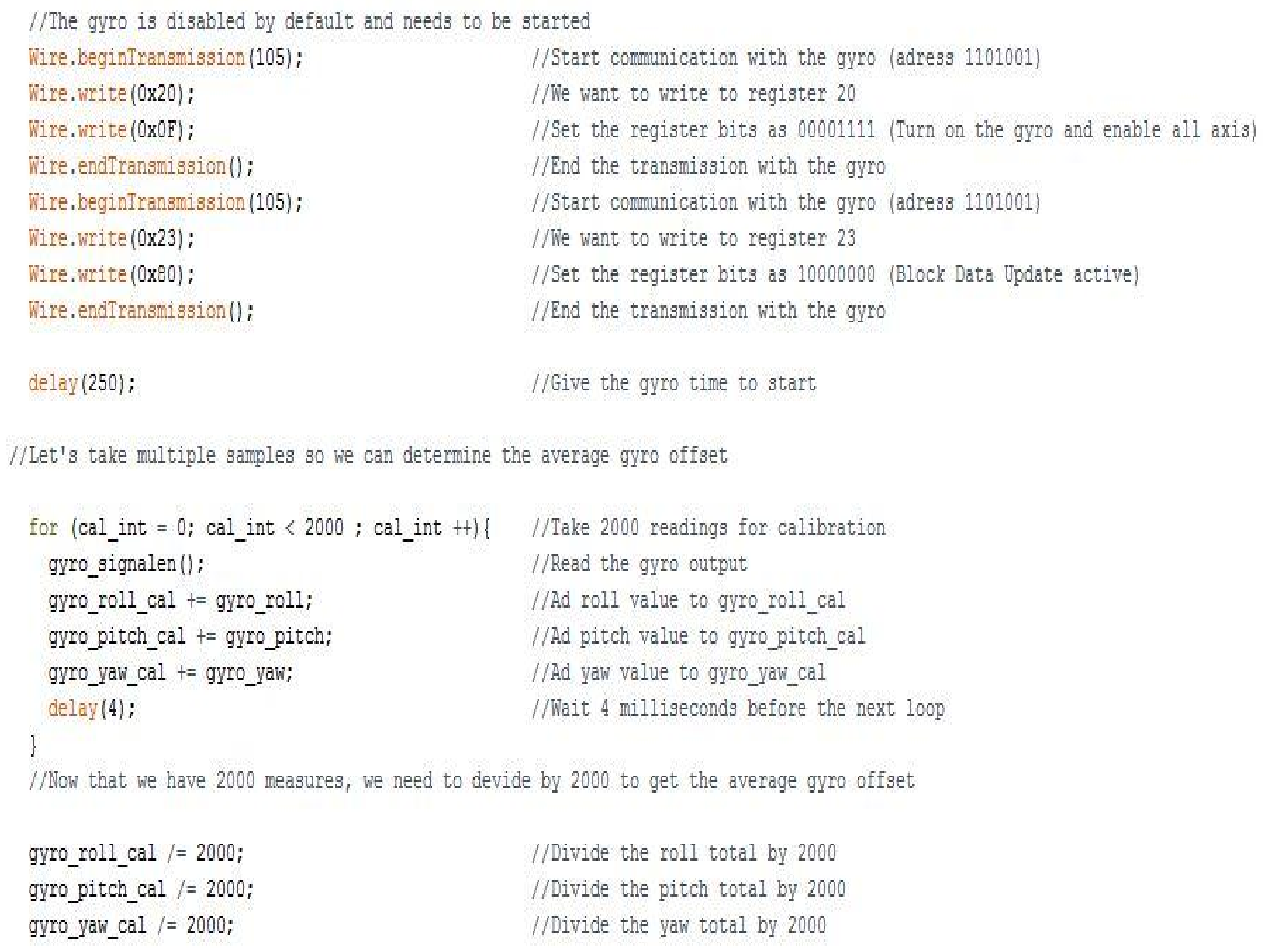


Figure 22: Gyroscope.

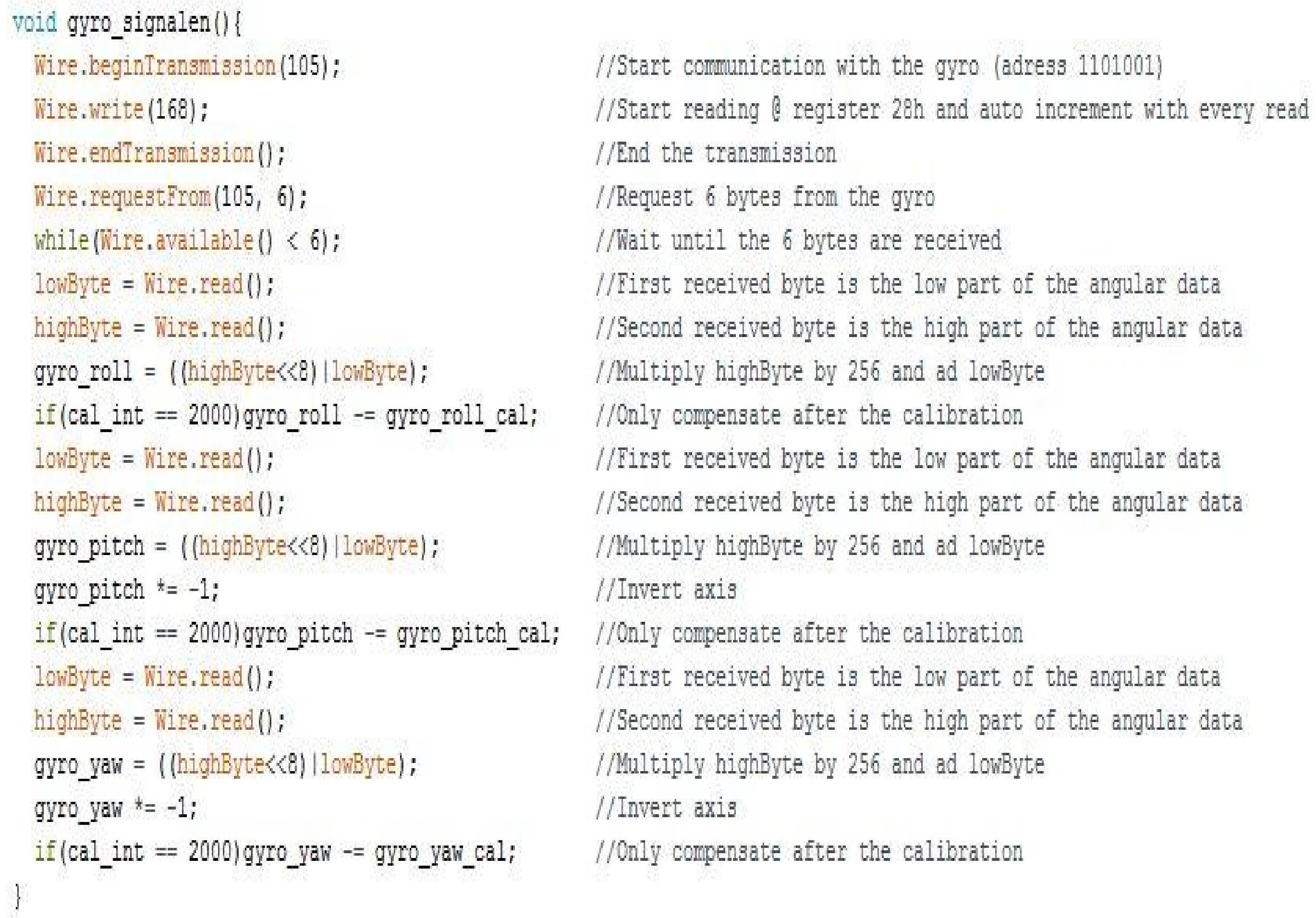
In the setup(), the Arduino communicate with the gyro registers and give some time to the gyro to be ready in the for loop as shown and explained in the Source 11.



```
  //The gyro is disabled by default and needs to be started
  Wire.beginTransmission(105);                    //Start communication with the gyro (adress 1101001)
  Wire.write(0x20);                               //We want to write to register 20
  Wire.write(0x0F);                               //Set the register bits as 00001111 (Turn on the gyro and enable all axis)
  Wire.endTransmission();                         //End the transmission with the gyro
  Wire.beginTransmission(105);                    //Start communication with the gyro (adress 1101001)
  Wire.write(0x23);                               //We want to write to register 23
  Wire.write(0x80);                               //Set the register bits as 10000000 (Block Data Update active)
  Wire.endTransmission();                         //End the transmission with the gyro

  delay(250);                                     //Give the gyro time to start

//Let's take multiple samples so we can determine the average gyro offset

  for (cal_int = 0; cal_int < 2000 ; cal_int ++){    //Take 2000 readings for calibration
    gyro_signalen();                              //Read the gyro output
    gyro_roll_cal += gyro_roll;                   //Ad roll value to gyro_roll_cal
    gyro_pitch_cal += gyro_pitch;                 //Ad pitch value to gyro_pitch_cal
    gyro_yaw_cal += gyro_yaw;                     //Ad yaw value to gyro_yaw_cal
    delay(4);                                     //Wait 4 milliseconds before the next loop
  }
  //Now that we have 2000 measures, we need to devide by 2000 to get the average gyro offset

  gyro_roll_cal /= 2000;                          //Divide the roll total by 2000
  gyro_pitch_cal /= 2000;                         //Divide the pitch total by 2000
  gyro_yaw_cal /= 2000;                           //Divide the yaw total by 2000
```

Source 11: Communicate with the gyro.

In the loop(), we call gyro_signalen() method that communicate with the gyro and make calculation for pitch, roll, and yaw. As shown in the source 8, gyro pitch, roll, and yaws give to the motors, so if one of the motors changed by a disturbance force, the gyro will stable the quadcopter. The calculation of the gyro shown in source 12.



```
void gyro_signalen(){
  Wire.beginTransmission(105);                     //Start communication with the gyro (adress 1101001)
  Wire.write(168);                                 //Start reading @ register 28h and auto increment with every read
  Wire.endTransmission();                          //End the transmission
  Wire.requestFrom(105, 6);                        //Request 6 bytes from the gyro
  while(Wire.available() < 6);                     //Wait until the 6 bytes are received
  lowByte = Wire.read();                           //First received byte is the low part of the angular data
  highByte = Wire.read();                          //Second received byte is the high part of the angular data
  gyro_roll = ((highByte<<8)|lowByte);             //Multiply highByte by 256 and ad lowByte
  if(cal_int == 2000)gyro_roll -= gyro_roll_cal;   //Only compensate after the calibration
  lowByte = Wire.read();                           //First received byte is the low part of the angular data
  highByte = Wire.read();                          //Second received byte is the high part of the angular data
  gyro_pitch = ((highByte<<8)|lowByte);            //Multiply highByte by 256 and ad lowByte
  gyro_pitch *= -1;                                //Invert axis
  if(cal_int == 2000)gyro_pitch -= gyro_pitch_cal; //Only compensate after the calibration
  lowByte = Wire.read();                           //First received byte is the low part of the angular data
  highByte = Wire.read();                          //Second received byte is the high part of the angular data
  gyro_yaw = ((highByte<<8)|lowByte);              //Multiply highByte by 256 and ad lowByte
  gyro_yaw *= -1;                                  //Invert axis
  if(cal_int == 2000)gyro_yaw -= gyro_yaw_cal;     //Only compensate after the calibration
}
```

source 12: Gyro calculation.

Object	Cost	Reason
iMax LiPo/NiMh battery charger	50.5$	Charges the battery.
Turnigy LiPo battery 5000 MAH	51.5$	Power the ESCs and motors.
Ultrasonic distance sensor	10$	Controlling lift.
4x motor +4x esc +4x propeller	54.6$	
Frame	20.1$	
Hc-05 Bluetooth	7$	To control the Quadcopter.
Gyroscope	36.62$	Stability.
Arduino Uno R3 with kit	47.84$	Base part (Board)
Total cost	278.16$	

4.5 TABLE OF COSTS THAT WE SPENT IN OUR PROJECT

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

This project is to build a Quadcopter to be able to flight and landing successfully. Out work is to this purpose as an our first building quadcopter project, using Android is a very main thing in this project, because by using android we can control many things and ideas we want. Also we using Arduino in other hand to sending data to the quadcopter. In this project we found many errors, first of all we learn that every sets of motors have a different VCC and Ground, the color of wires maybe changed from a set to another.

Also we knew that if one of the propellers setup in a wrong way, it will cause the quadcopter grounded.

Connecting any force to the quadcopter may cause the quad unbalanced.

The battery has a very causative affect. If the battery is in low power, the speed of the motors will decrease.

Motors are hot and flight in a different micro seconds, to make them as same as each other, we need a very accurate calculation.

But fortunately the gyro can correct many of these errors.

5.2 FUTURE WORK

Our aim in the future is to add a camera to our quadcopter to take photo and video in hard places that human can't go there. The very important thing that we will do in the future is making the quadcopter to be more and more stable as it, now of course we have some issues because it is a first and very beginning project that we made for building a quadcopter, and really it is not a small thing.

REFERENCES

A2212/13T TECHNICAL DATA. (n.d.). Retrieved from Hot Deals: http://www.batteryheatedclothing.com/

BEST QUADCOPTER MOTORS : CHOOSING RC QUADCOPTER OR MULTICOPTER MOTORS MADE EASY. (n.d.). Retrieved from The Fancy Voyager: http://thefancyvoyager.com/

Hirzel, T. (n.d.). *PWM*. Retrieved from Arduino: https://www.arduino.cc

Luukkonen, T. (2011). *Modelling and control of quadcopter.* Teppo Luukkonen.

MELLOULI, M. (2010). *Android Application.*

MONGKHUN. (2012). *WIRELESS CONTROL QUADCOPTER WITH STEREO CAMERA AND.*

MONGKHUN, & VECHIAN, M. Q. (2012). *WIRELESS CONTROL QUADCOPTER WITH STEREO CAMERA AND.*

Norris, D. (2014). *Build Your Own Quadcopter.* Donald Norris.

Oscar. (2013). *How to choose Motor and Propeller for Quadcopter and Multicopter*. Retrieved from Oscarliang: https://oscarliang.com

Oscar. (2013, October 13). *Quadcopter PID Explained and Tuning*. Retrieved from Oscar Liang - Sharing Knowledge and Ideas: https://oscarliang.com

Oscar. (2014). *Choose Flight Controller for Quadcopter Hexacopter*. Retrieved from oscarliang: https://oscarliang.com

Oscar. (2014). *How to choose battery for Quadcopter, Tricopter and Hexacopter*. Retrieved from Oscarliang: https://oscarliang.com

Oscar. (2015). *How to choose ESC UBEC, BEC for Multicopters*. Retrieved from oscarliang: http://oscarliang.com

Parker, M., Robbiano, C., & Bottorff, G. (2011). *Quadcopter.* Matt Parker;Chris Robbiano;Gerad Bottorff.

Quadcopter Parts List | What You Need to Build a DIY Quadcopter. (n.d.). Retrieved from Quadcopter Carage: www.quadcoptergarage.com

Sam. (2015). *ESC to motor connection guide - how to reverse your motor direction the easy way*. Retrieved from dronetrest: http://www.dronetrest.com/

Serial-Port Bluetooth Module (Master/Slave). (n.d.). Retrieved from watterott.com: http://www.watterott.com/

Team, A. D. (n.d.). *Connect ESCs and Motors*. Retrieved from ArduPilot Autopilot Suite: http://ardupilot.org/

Todd, A., & Barraclough, C. (2016, FEBRUARY 13). *What is Android and what is an Android phone?* Retrieved from Compare Mobile: www.recombu.com

Tzivaras, V. (2015). *Building a Quadcopter with Arduino.* Birmingham: Packt Publishing Ltd.

VECHIAN, M. Q. (2012). *WIRELESS CONTROL QUADCOPTER WITH STEREO CAMERA AND.* MONGKHUN QETKEAW A/L VECHIAN .

What is Arduino? (n.d.). Retrieved from Arduino: www.arduino.cc

Wirsing, B. (2014). *Sending and Receiving Data via Bluetooth with an Android Device.*