Software Controlled Drone Project

Team Members: Kevin Libdan & Denis Stepanov

Project's website: https://github.com/Libs1/SoftwareControlledDrone.github.io

Discipline: Computer Engineering Technology

Date of Submission: March 28, 2017

Declaration of Joint Authorship

The authorship of this project is evenly divided between Denis Stepanov and Kevin Libdan. Denis Stepanov is responsible for handling the android application which will be used to control the drone from an android device. He is also responsible for creating the design aspect of the android application (this includes all the layouts, animations, languages e.t.c). Kevin Libdan is responsible for creating the database which will be used to communicate with the android application and store information such as user information and drone information (flight duration, date and location). Denis Stepanov and Kevin Libdan will both be responsible for the hardware aspect. The hardware will allow the Eachine H8 drone be controlled by two analog joysticks connected to an Arduino and an android application which will have two virtual joysticks. Denis Stepanov and Kevin Libdan will also be responsible for the web interface. The web interface will mimic the same functionalities as the android application which will allow the users to register, login and view their drone's flight information.

Approved Proposal

Proposal for the Software Controlled Drone Project

Prepared by Denis Stepanov and Kevin Libdan
Computer Engineering Technology Student
github.com/libs1/softwarecontrolleddrone.github.io

Executive Summary

As students in the Computer Engineering Technology program, We will be integrating the knowledge and skills we have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with sensors and actuators for controlling a drone. The database will store information on flight control. The mobile device functionality will include software support for controlling the flight of the drone and will be further detailed in the mobile application proposal. We will be collaborating with the following company/department Future Humber Quadcopter Swarm. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the current term in CENG 355 Computer Systems Project as a member of a 2 student group.

Background

The problem solved by this project is to control a drone remotely with an application built for android devices. The problem is to recognize the radio signal used to communicate with the drone and make the android application to send proper flight instructions to the drone.

Drone technology is fairly new and it's becoming popular day by day. It has become popular enough so that an organization like Amazon is testing product delivery by drone but controlling manual or automated flight path of drones is difficult and requires dedicated controllers. Controllers can sometimes be more expensive or non-intuitive to use. In this situation widely popular can make the task more fun, intuitive and widely accessible. Android applications are easily obtainable and can have very intuitive user interface. In addition android devices are ubiquitous. So having proper android application for controlling the drones has very high potential to make drones more popular and user friendly.

I have searched for prior art via Humber's IEEE subscription selecting "My Subscribed Content" and have found and read three articles which provides insight into similar efforts.

The first article discusses how UAVS have been getting a lot of attention due to its low cost of implementation and how an AR drone is being controlled by a motion capture system to follow a moving target. (Gomes, Leal, Oliveira, Cunha, & Revoredo, 2016)

The second article discusses how to collect input information for the controller used for the AR drone. (Barták, Hraško, & Obdržálek, 2014)

The third article discusses estimating UAV systems total ownership cost including hardware components, software design, and operations. (Malone, Apgar, Stukes, & Sterk, 2013)

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

- Java Docs from CENG 212 Programming Techniques In Java,
- Construction of circuits from CENG 215 Digital And Interfacing Systems,
- Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
- Micro computing from CENG 252 Embedded Systems,
- SQL from CENG 254 Database With Java,
- Web access of databases from CENG 256 Internet Scripting; and,
- Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the winter semester. My coursework will focus on the first two of the 3 phases of this project:
Phase 1 Hardware build.
Phase 2 System integration.
Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the fall term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that I have learned to potential employers.

The tables below provide rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Labour Estimates	**Hrs**	**Notes**
Phase 1		
Writing proposal.	9	Tech identification quiz.
Creating project schedule. Initial project team meeting.	9	Proposal due.
Creating budget. Status Meeting.	9	Project Schedule due.
Acquiring components and writing progress report.	9	Budget due.
Mechanical assembly and writing progress report. Status Meeting.	9	Progress Report due (components acquired milestone).
PCB fabrication.	9	Progress Report due (Mechanical Assembly milestone).
Interface wiring, Placard design, Status Meeting.	9	PCB Due (power up milestone).
Preparing for demonstration.	9	Placard due.
Writing progress report and demonstrating project.	9	Progress Report due (Demonstrations at Open House Saturday, November 7, 2015 from 10 a.m. - 2 p.m.).
Editing build video.	9	Peer grading of demonstrations due.
Incorporation of feedback from demonstration and writing progress report. Status Meeting.	9	30 second build video due.
Practice presentations	9	Progress Report due.
1st round of Presentations, Collaborators present.	9	Presentation PowerPoint file due.
2nd round of Presentations	9	Build instructions up due.
Project videos, Status Meeting.	9	30 second script due.
Phase 1 Total	**135**	
Phase 2		
Meet with collaborators	9	Status Meeting
Initial integration.	9	Progress Report
Meet with collaborators	9	Status Meeting
Testing.	9	Progress Report

Meet with collaborators	9	Status Meeting
Meet with collaborators	9	Status Meeting
Incorporation of feedback.	9	Progress Report
Meet with collaborators	9	Status Meeting
Testing.	9	Progress Report
Meet with collaborators	9	Status Meeting
Prepare for demonstration.	9	Progress Report
Complete presentation.	9	Demonstration at Open House Saturday, April 9, 2016 10 a.m. to 2 p.m.
Complete final report. 1st round of Presentations.	9	Presentation PowerPoint file due.
Write video script. 2nd round of Presentations, delivery of project.	9	Final written report including final budget and record of expenditures, covering both this semester and the previous semester.
Project videos.	9	Video script due
Phase 2 Total	**135**	
Phase 3		
Interviews	TBD	
Phase 3 Total	**TBD**	
Material Estimates	**Cost**	**Notes**
Phase 1		
Arduino Uno R3	>$30.95	(Arduino) Amazon
SparkFun Transceiver Breakout - nRF24L01	>$29.37	CanadaRobotix
Eachine H8 Mini Quadcopter	>$28.99	(EachineDirect) Amazon
Lithium AA Batteries	>$5.64	CanadianTire (1 pack comes with 4 AA batteries)
Analog Joysticks	>$32.02	Brainy-Bits (2 analog sticks)
Breadboard	>$8.90	(Elenco) Amazon or Parts kit
Pin Header (8-pin) Male	>$0.50	Sparkfun
Jumper wires	>$3.13	(Sodial) Amazon
Phase 1 Total	**>$139.50**	
Phase 2		
Materials to improve functionality, fit, and finish of project.		
Phase 2 Total	**TBD**	
Phase 3		
Off campus colocation	<$100.00	
Shipping	*TBD*	
Tax	*TBD*	
Duty	*TBD*	
Phase 3 Total	**TBD**	

Concluding remarks

This proposal presents a plan for providing an IoT solution for *Future Humber Quadcopter Swarm*. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects. I request approval of this project.

Abstract

This project is aimed to control an Eachine H8 drone with two analog joysticks and to control the drone remotely with an application built for android devices. This technical report will go over the ways on how to do so. The android application would also allow users to register if they do not have an account, sign in and view their drone's flight information. We started this project by building a piece of hardware that would allow us to communicate with the drone and also being able to fly it. The major components of our hardware are an Arduino Uno R3, an NRF24l01 wireless transceiver, and two analog joysticks. Developing the application, database, and web interface allowed us to utilize with our hardware.

Table of Contents

Illustrations or Diagrams

NRF24L01 Solder Diagram

Figure 1:

This diagram is a representation of where to solder the pin header (8-pin) into the NRF24L01 transceiver.

Further instructions to do so can be found at 2.2.4 Mechanical Assembly

NRF24L01 To Arduino Hookup

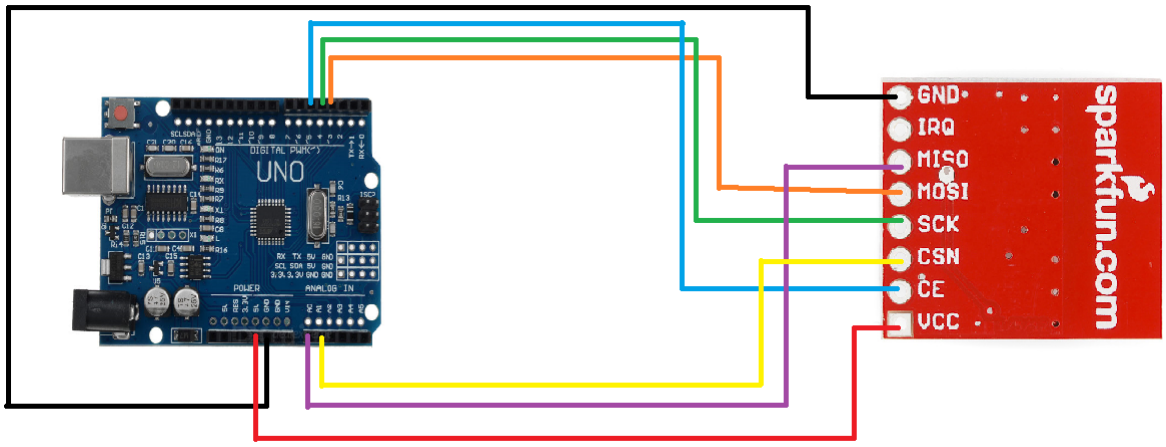

Figure 2:

This diagram displays the wiring between the Arduino Uno R3 and the NRF24L01 transceiver.

Further instructions to do so can be found at 2.2.4 Mechanical Assembly

Joysticks To Arduino Hookup

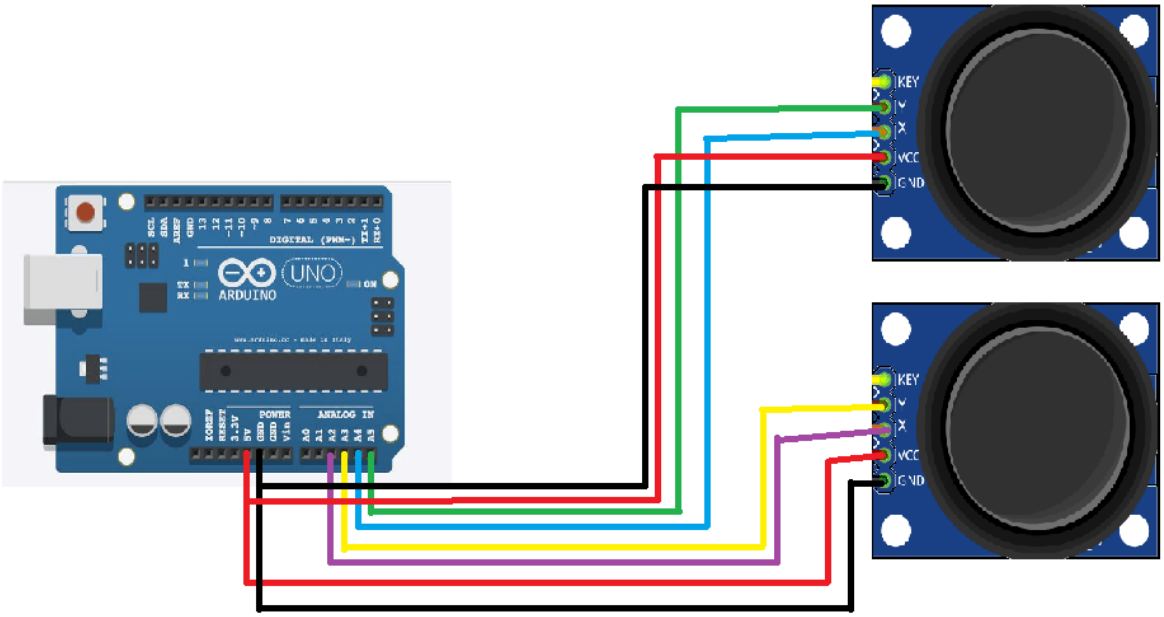

Figure 3:

This diagram displays the wiring between the Arduino Uno R3 and the two analog joysticks. Further instructions to do so can be found at 2.2.4 Mechanical Assembly

1. Introduction

1.1 Purpose

Drone technology is fairly new and it's becoming popular day by day. It has become popular enough that organizations like Amazon are testing product delivery by drones, but controlling manual or automated flight path of drones is difficult and requires dedicated controllers. Controllers can sometimes be more expensive or non-intuitive to use. Android applications are easily obtainable and can have very intuitive user interface and in addition android devices are ubiquitous. For this reason, having a proper android application that would be able to control drones would increase the popularity for drones. The Software Controlled Drone project is designed to manipulate the binding sequence and as well as controlling the Eachine H8 mini quadcopter drone. This project is focused on creating hardware and an android application that will be capable of controlling the drone. This Software Controlled Drone (SCD) project will have a hardware component, an android application, a database and a web interface.

1.2 Scope

The Software Controlled Drone project will be able to control the Eachine H8 mini quadcopter drone. The android application will have the capability of controlling drone remotely by using two virtual analog sticks in the controller activity and allowing the users to view their drone's flight information such as the duration of the flight and as well as the date of which the drone has been flown. The database will contain the user's information such as their first and last name, username and password. The database will also store information of the drone's flight information that will be displayed to the user. The web interface will allow users to review their drone's flight information.

1.3 Targeted Audience Group

Our targeted audience group would be anyone who is interested on flying drones. As drones are increasing in popularity, this project would give users a different feel of control, whether they would want to fly the drone with the physical or virtual joysticks and allow users to keep track of their runtime information.

2. Project Description

2.1 External Interface Requirements

2.1.1 Database

The MYSQL database is responsible for storing the users account information and as well as the drone's activity information. The database uses PHP in order for it to connect with the android application and as well, the database will be running on hostinger which is a free hosting website. There are two tables within the database; the first table is the DroneMembers table which contains the user's information and the second table which is the DroneInfo table which contains information of the drone's activity. The DroneMembers table contains four fields which are first name, last name, username and password. The first name and last name fields contain the user's first name and last name. The username field contains the user's username which is used for logging in. The password field contains the user's password which is used when logging in. The DroneInfo field contains two fields which are date and flight duration. The date field is contains the date of when the drone has been flown and the flight duration field contains the user's total flight time. (Developed by Kevin Libdan)

2.1.2 Web Interface

Depending the how long the binding process takes we will develop the web interface. The web interface will be developed to allow users to register an account. The website will also be capable of allowing the users to view their drone's information such the date of when the drone has been flown and as well as the flight duration. The website will handle basic functionality such as viewing the user's drone runtime information. Denis Stepanov is responsible for the HTML, CSS and the overall design aspect of the website. Kevin Libdan will be responsible for creating the PHP scripts that will be used to connect to the database when a user registers an account, sign in and view their drone's information. (Developed by Denis Stepanov and Kevin Libdan)

2.1.3 Hardware

The hardware built is used to control the Eachine H8 drone. The hardware to control the Eachine H8 drone is an Arduino, a NRF24L01+ transceiver and two analog joysticks. The hardware is capable of communicating with the drone and being able to control it with the two analog sticks. An OTG cable, micro male USB to Female USB, is used between the Arduino and an android application in order to establish a serial communication. This will allow the Arduino to receive data from the android application which will then be sent to the drone. Kevin Libdan is responsible for acquiring the materials and as well as the assembly of the materials. Denis Stepanov is responsible for creating the Arduino sketch that would be compatible with the hardware.(Developed by Denis Stepanov and Kevin Libdan)

2.1.4 Application

Software Drone Android Application is the software used to bind with the quadcopter, maneuver it and save data in MYSQL database using the readings from the drone's flight, implemented in one of the classes. Android Application consists of 10 different java classes which support multiple functionalities. Main classes, which would be visible to users, such as 'ControllerActivity','FlightsActivity','LoginActivity','MenuActivity', 'RegisterActivity' and 'PopActivity' use specific layouts assigned to them. Each one of those activities' layouts support phone, tablet, and large tablet (>10") size of the screen. All of visible activities have the function to adjust to both portrait and landscape mode (except 'ControllerActivity', which is always set on landscape to create more user-friendly interface). All of the main classes use custom 'drawables' to make the application more presentable. There are also 5 other classes which are used for better functionality of the application. All of those 5 classes have communication in some sort with database. For example, 'TableData' class is consists of the information about the drone flight (date and flight duration).Moreover, the application adjusts to different resolutions of images and icons for different devices automatically using Android Asset.

There is also a support for 4 different languages (English, French, Spanish, and Russian). (Developed by Denis Stepanov)

2.2 Build Instructions

2.2.1 Introduction

In this section of the technical report, there will be information on how to recreate the hardware to control the Eachine H8 Mini Quadcopter with an Arduino and an Android device, which will use an NRF24L01 transceiver to communicate with the drone.

2.2.2 Build of Materials/Budget

Item	Quantity	Total Price(With Tax + Shipping)
Arduino Uno R3	1	$30.95
SparkFun Transceiver Breakout - nRF24L01	1	$29.37
Eachine H8 Mini Quadcopter	1	$28.99
AA Batteries	1 pack(containing 4 batteries)	$5.64
BreadBoard	1	$8.90
Pin Header (8-pin) Male	1	$0.50
Jumper Wires	40 pack	$3.13
Analog Joysticks	2	$36.02

2.2.3 Time Commitment

Task	Time To Complete The Task
Ordering Parts	3 Days(Shipping)
Soldering the pin header to the NRF24L01 Transceiver	20 minutes
Wiring the NRF24L01 transceiver to the Arduino	5 minutes
Wiring the joysticks to the Arduino	5 minutes
Downloading the Arduino IDE	10 minutes(Depending on computer Performance)
Importing the NRF24L01_Multipro library to the Arduino	2 minutes
Running the code	5 minutes

2.2.4 Mechanical Assembly

Step 1: Purchase the required parts

Step 2: Preparing the NRF24L01

Solder the pin header (8-pin) into the pins indicated with a green dot as shown on Figure 1.1. The pin headers will act as legs for the transceiver when you place the transceiver on the breadboard. Soldering the pin headers to the pins of the transceiver may be challenging due to the size of the pins and as well for individuals who are unexperienced in soldering.

Figure 1.1

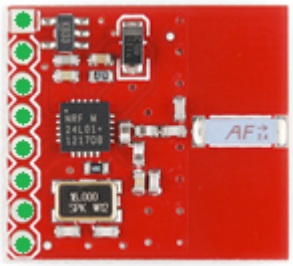

Figure 4:

Step 3: Hooking up the NRF24L01 to the Arduino

Before we make any wiring connections between the Arduino and the NRF24L01 transceiver, make sure the transceiver is sitting on the breadboard properly. When the transceiver is in place, make the following connections between the Arduino and the NRF24L01 transceiver. Figure 2.1 shows how the connections should look when all connections are made.

This is the pin set up for the Arduino to the NRF24L01 Transceiver:

5v -> VCC

GND -> GND

MOSI -> 3

SCK -> 4

CE -> 5

MISO -> A0

CSN -> A1

Figure 2.1

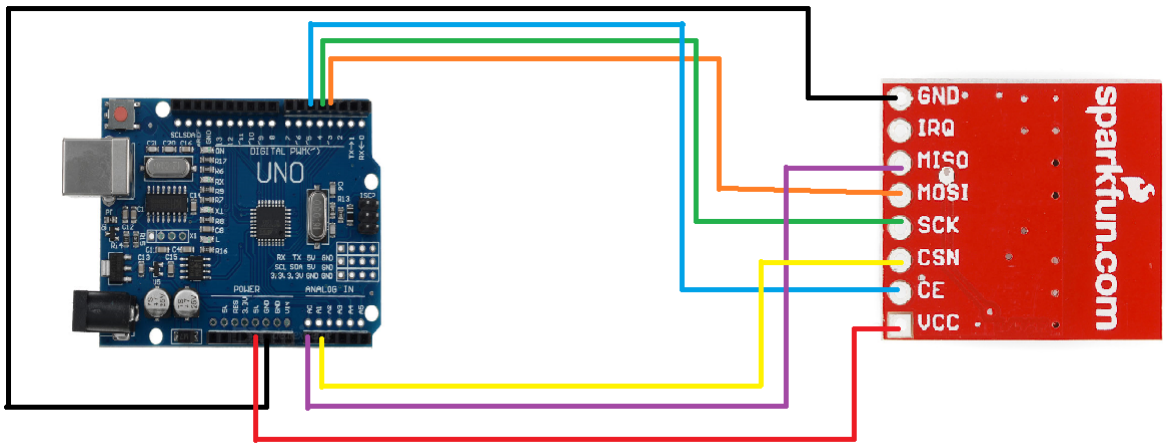

Figure 5:

Step 4: Hooking up the joysticks to the Arduino

Make the following connections to hook up both analog sticks to the Arduino. Figure 3.1 shows the connections made between the two analog joysticks and the Arduino. Whichever joystick you set up to A2 and A3 on the Arduino will be your left joystick as it will control the throttle (up and down movement of the drone) and rudder (left rotate and right rotate of the drone). The other joystick, which will be your right joystick, will control the Aileron (leftward and rightward movement of the drone) and elevator (forward and backward movement of the drone).

Left Joystick to Arduino:

X -> A2

Y-> A3

VCC -> 5v

GND -> GND

Right Joystick to Arduino:

X -> A4

Y -> A5

VCC -> 5v

GND -> GND

Figure 3.1

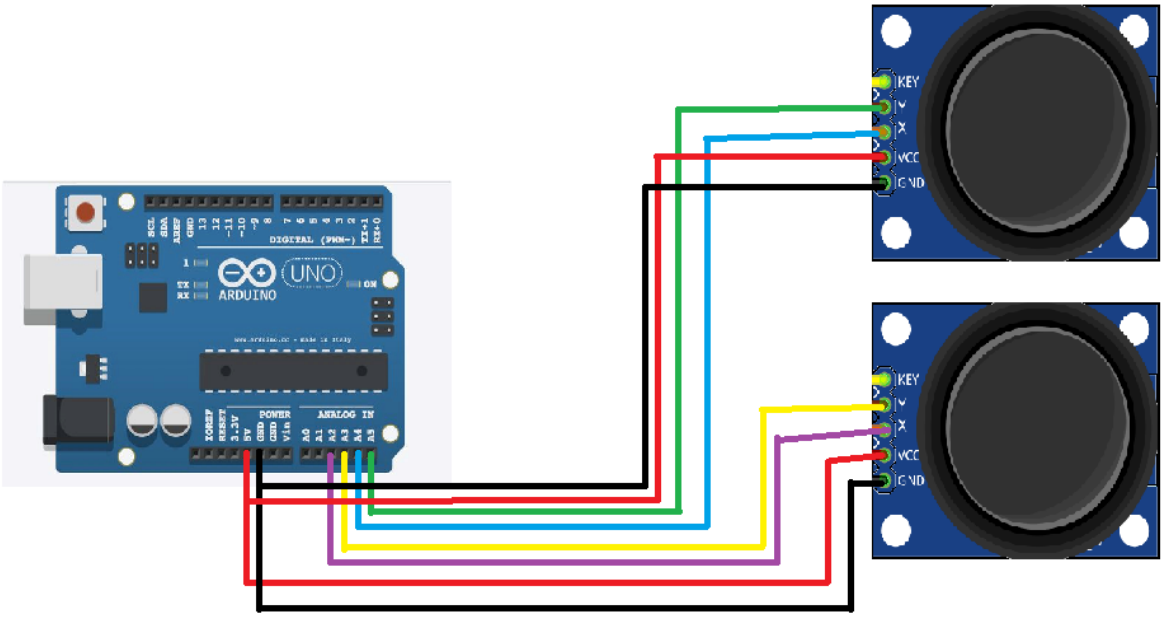

Figure 6:

2.2.5 Power Up

Make sure that all connections are in the appropriate pins and that there are not any loose connections. Once the circuit has been built, plug in the Arduino into the PC and open the Arduino IDE. Plug in the power to the Eachine H8 drone and the LED lights should be blinking in a steady pace. Now that we have the drone powered on, you can download the zip file which contains the Arduino code that will communicate with the drone. You can download it from here. The original code can be found on Goebish's github which is found here. Our code has been modified to be compatible with our two analog joysticks. Go back to your Arduino IDE and open up the "nrf24l01_multipro.ino" file and then upload the sketch to the Arduino. Once the code is uploaded, the drone's LEDs should be blinking rapidly, this indicates that the drone is ready to bind. At this point, moving your left joystick down should bind the drone and the drone's LEDs should be steady.

If you are using the **android application**, make sure to follow these steps in order:

· Ensure that the "nrf24l01_multipro.ino" file is uploaded to the Arduino as the Arduino will execute the most recent uploaded sketch.

· Connect the power to the drone. The LEDs should be blinking at a steady pace.

· Connect the OTG cable between the android device and the Arduino. The LEDS on the drone should now be blinking rapidly.

· Open the SCD app, login and proceed to the controller and then click on start

2.2.6 Unit Testing

Ensure that the drone's LEDs would be solid as it would indicate a successful binding sequence when the left joystick is moved. If the binding sequence was not successful, it would most likely be a hardware problem. It is important to check that all of the connections made between the NRF24L01, joysticks and the Arduino are made properly and secured. Another test is to ensure that the joysticks are controlling the drone as it is supposed to. The throttle (vertical movement) of the left joystick would move the drone up and down; the rudder (horizontal movement) of the left joystick would right and left rotate the drone,

the elevator (vertical movement) of the right joystick would move the drone forward and backwards and the Aileron (horizontal movement) of the right joystick would move the drone leftward and rightward.

If you are flying the Eachine H8 drone with the android application, ensure that the drone's LEDS are solid after following the steps in *2.2.5 Power Up*. The left virtual joystick would control the throttle and rudder movements, and the right virtual joystick would control the aileron and elevator movements.

2.2.7 Production Testing

At this final stage of testing, the hardware is capable of controlling the drone independently with the two physical joysticks or with the Android application. The following tests were performed to ensure that every component of the project is functioning as it should be.

1. Ensure that the hardware functionality is working properly independently and also with the android application by referring to 2.2.5 Power Up and 2.2.6 Unit Testing.
2. The android application functions such as login, and register will work based on information in the database. The two virtual joysticks will work as what was mentioned in *2.2.6 Unit Testing*. The drone's flight information is sent to database and displayed on both the "flights" option on the app and on the web interface correctly.

2.3 Project Specifications

2.3.1 System Interface

The hardware that is built by using the Arduino Uno R3, nRF24L01 transceiver and joysticks are capable of functioning independently without the need of the android application. Although the hardware does not depend on the application, the application would only be used to register, sign in and view the drone's flight information when it is used without the hardware. The web interface can also operate without the need of the hardware; however, the purpose of the web interface is to register or view the drone's flight information.

2.3.2 User Interface

The user interface of the android application is user friendly. When launching the application, the main page will consist of two fields for both username and password to be entered and as well as the option to register or login. If the user has successfully entered their username and password they will be brought to the menu page which will consist of two options, one being the controller and the other being the flight activity. When choosing the controller option, the user will be confronted with two virtual joysticks which are capable of controlling the drone. The flight activity option will allow the user to view their drone's flight information such as flight duration and the current date the drone has been flown.

2.3.3 Hardware Interface

The project consists of an Arduino Uno R3, an nRF24 transceiver and two joysticks. The transceiver is used for communicating with the drone and the two analog joysticks are used to control the drone's throttle, rudder, aileron and elevator.

2.3.4 Software Interface

The android application is the software we have designed in order to control the Eachine H8 drone. Both the application and the Arduino sketch are used hand-in-hand, however, the Arduino sketch is capable of functioning without the android application. The web interface was designed to allow users to simply register and view their drone's flight information.

2.4 Project Schedule/Progress Report

2.4.1 Project Schedule

Phase 1

· Creating Project Proposal

Wednesday (9/7/16) – Thursday (9/8/16)

· Creating Budget

Wednesday (9/14/16) – Thursday (9/15/16)

· Acquiring components, Progress Report

Wednesday (9/21/16) – Thursday (9/22/16)

· Mechanical Assembly, Second Progress Report

Wednesday (9/28/16) – Thursday (9/29/16)

· PCB Fabrication

Wednesday (10/5/16) – Thursday (10/6/16)

· Interface wiring, Placard design

Wednesday (10/12/16) – Thursday (10/13/16)

· Preparing demonstration

Wednesday (10/19/16) – Thursday (10/20/16)

· Writing progress report/demo project

Wednesday (10/26/16) – Thursday (10/27/16)

· Edit build video

Wednesday (11/2/16) – Thursday (11/3/16)

· Writing progress report/status meeting

Wednesday (11/9/16) – (11/10/16)

· Practice presentations

Wednesday (11/16/16) – Thursday (11/17/16)

· Conduct Presentations

Wednesday (11/23/16) – Thursday (11/24/16)

· Build instructions

Wednesday (11/30/16) – Thursday (12/1/16)

· Project videos, Status meeting

Wednesday (12/7/16) – Thursday (12/8/16)

Phase 2

· Group meeting

Monday (1/9/17) – Tuesday (1/10/17)

· Initial integration

Monday (1/16/17) – Tuesday (1/17/17)

· Software Requirement Specifications(SRS)

Monday (1/23/17) – Tuesday (1/24/17)

- Progress report

Monday (1/30/17) – Tuesday (1/31/17)

- Project status

Monday (2/6/17) – Tuesday (2/7/17)

- Progress report of independent progress

Monday (2/13/17) – Tuesday (2/14/17)

- Project status

Monday (2/20/17) – Tuesday (2/21/17)

- Progress report/project integration

Monday (2/27/17) – Tuesday (2/28/17)

- Testing

Monday (3/6/17) – Tuesday (3/7/17)

- Project status

Monday (3/13/17) – Tuesday (3/14/17)

- Prepare for demonstration

Monday (3/20/17) – Tuesday (3/21/17)

- Complete presentation

Monday (3/27/17) – Tuesday (3/28/17)

- Complete final report

Monday (4/3/17) – Tuesday (4/4/17)

- Write video script

Monday (4/10/17) – Tuesday (4/11/17)

- Project videos

Monday (4/17/17) – Tuesday (4/18/17)

2.4.2.1 Progress report

Recent project activities:

We have contacted the person who designed the Arduino code for NRF24L01 communication and realised that Syma x12s may be using a different protocol than the Syma x12. It resulted in the Syma x12s' incompatibility with our Arduino sketch. As a group we have decided to purchase a new quadcopter, Eachine H8, which was one of the supported drones for the Arduino sketch. (https://www.amazon.ca/Eachine-Quadcopter-Headless-Drone-Black/dp/B00XHOOWA0).

Current Objectives:

As of today, our primary goal would be to establish communication between Arduino and Android Application.

Problems / Opportunities, Solutions:

For almost a month and a half we had a problem with binding our quadcopter and Arduino. After finding out that Syma X12s might not be compatible with latest version of NRF24L01 breakout board we came up with a new solution. With some help from Goebish (https://github.com/goebish/nrf24_multipro), who has a lot of experience in deviation, we decided to purchase another quadcopter (Eachine H8, as it was stated above). It allowed us to bind the drone with Arduino and control it with the joysticks.

Financial Updates:

Pair of Joysticks for Arduino: $18.01

Eachine H8 mini: $30

All of the prices do not include HST nor Shipping estimates. However, they will be included in the final updated version of Budget Report after we would establish the connection between quadcopter and Android Application.

Additional Links:

https://www.makehardware.com/2016/07/04/how-to-control-your-drone-from-a-computer/https://www.makehardware.com/2016/07/04/how-to-control-your-drone-from-a-computer/

https://brainy-bits.com/tutorials/arduino-joystick-tutorial/https://brainy-bits.com/tutorials/arduino-joystick-tutorial/

2.4.2.2 Progress Report

Recent Project Activities:

As we now have full control over the drone with the two analog joysticks, we are now focusing on manipulating the drone by Android Application. We have also created a website where anybody is able to sign up, log in and view their drone's flight information. During this process, we had created several PHP files in order to display the user's respective flight information. In the drone application, we had modified our controller where it would be more user friendly than our initial controller we had designed. We have also added a small feature within the app where the user is able to delete their existing flight information from the database when the user clicks on the "Delete All" button.

Current Objectives:

Our current and main objective is to establish communication between our Arduino and Android application in order to control our Eachine H8 drone. We are testing 2 ways of manipulating the drone, such as using OTG Cable, which has direct connection between Arduino and Android device, and Bluetooth Module HC-05. The latter has more complications in order to achieve full communication, but it will be our prior way of connection. If the connection won't be established between Arduino and Android device by using the Bluetooth Module, we will be using an OTG cable since it is an easier way to communicate.

Problems/Opportunities, Solutions:

The main problem we are encountering is the communication between the Arduino and our Android application. We have two options in which we can establish the connection, the first option would be to use an OTG cable and establish a serial connection and the second option would be to use Bluetooth by using the HC-05 Bluetooth Module. As the solution, we are testing basic communication through lighting up the LED connected to the Arduino circuit right now.

Financial Updates:

OTG Cable Micro USB male to USB Female: $6.99 with shipping + Tax (Amazon Prime)

OR

HC-05 Bluetooth Module: $18.65 with shipping + Tax (CanadaRobotix)

Additional Links:

https://www.allaboutcircuits.com/projects/communicate-with-your-arduino-through-android/

http://www.instructables.com/id/Android-Bluetooth-Control-LED-Part-2/

2.4.2.3 Progress Report

Dear Kristian,

You can find below our project's integration status report followed by current objectives and goals in merging process:

Software Controlled Drone project includes three main components: main circuit (Arduino Uno with transceiver), software (Android Application) and web interface (website). The main collaboration between significant pieces is currently in progress because we have encountered some obstacles in establishing connection between the quadcopter and Arduino. However, we were able to merge some of our work progress in one and complete testing of several subjects.

Kevin Libdan was managing database structure and connection to online servers on application side, while I developed web interface for users to check their information online with option to sign up. In the end, user is able to sign up/sign in the Android Application and send some fake data (not connected to the drone) to the server's database, while at the same time user can input their credentials to login on website and check flight information, the same 'fake data' he or she just entered in the application.

Other part of the project is merging our software and hardware together. Kevin have worked on communication with OTG cable and Arduino while I tried to establish the connection through Bluetooth Module HC-05. After some research and testing on modules, we have came up with a conclusion that NRF24L01 and Bluetooth Module HC-05 can not work together simultaneously due to frequency interference. For yet unknown fact we have not understood why exactly they are not working but our simplest guess would be because they both set on 2.4 GHZ frequency; therefore, it is not possible establishing a connection without interference. Due to this fact, our next step in integration process will be basic testing using OTG cable and Arduino, such as sending text or lighting up the LED.

Sincerely,

Denis

2.4.2.4 Progress Report

Recent Project Activities:

We have been testing serial communication between our android application and microcontroller in order to control our Eachine H8 drone. We have also been modifying our controller in our android app to establish a connection with our Arduino whenever the two devices are connected with the OTG cable, as well as sending data to the microcontroller whenever the joysticks are moved. With some help from online resources and past projects (https://github.com/rmahenthiran/Micro-Drone-Flight-Control), we were able to control our drone with our android application.

Current Objectives:

Our current objective is to successfully control the drone with our app without the need of holding down the physical joysticks.

Problems/Opportunities, Solutions:

Although we were able to control our drone with our android application, it was not functioning properly as expected. The main problem we are encountering right now is that the android application would only take control of the drone when the physical joysticks initiate the binding sequence first. This would mean that we would have to hold down the "throttle" joystick while we control the drone with the app. The solution for this problem could be removing the spring from the physical joystick which would allow us better testing but still stop us from achieving desired result of binding the drone without using physical joystick. Another solution could be rewriting a binding function in the Android Application, therefore only using the virtual joysticks for initiating the binding sequence.

Another problem we have encountered is the connection with the smartphone that uses USB type C. Since we are using OnePlus Two as one of our testing smartphones, we have tried to establish connection with the drone using OTG cable. The problem is that USB type C adapters are new on the market and not all of them are functioning. We have purchased several adapters/cables but none of them worked. Currently, there are no solutions for this problem, but OnePlus support team suggested using official OnePlus microUSB adapter (link below) and USB-C adapter to establish the connection. When the microUSB adapter arrived it could read the data from any USB stick connected to the Samsung phone which uses microUSB as the main charging port; however, after connecting it to the USB-C adapter and then to OnePlus Two, it would not recognize anything. As of the result, we have purchased one more USB-C adapter (link below) and waiting for it to arrive.

Moreover, as it was mentioned in the last progress report, we have encountered a problem with using Bluetooth module HC-05 and Nordic semiconductor simultaneously. Due to their frequency interference, we could not use them at the same time so it was suggested that we try a different channel on the Bluetooth module. We have researched information about HC-05 and found out that it uses SPP (Serial Port Profile) which is made to send bursts of data between two devices but there are no supportive links on how to use different channels to prevent interference. Therefore, we have eliminated the option of using the Bluetooth HC-05 module and stick to the OTG connection.

Financial Updates:

We have purchased a total of 3 new adapters for OnePlus Two smartphone, but none of them would be included in the project except of the last one acquired(only if test will be successful). Here are links for the adapters we have purchased:

https://www.newegg.ca/Product/Product.aspx?Item=9SIAA0D4C44428&nm_mc=KNC-GoogleAdwordsCA&cm_mmc
GoogleAdwordsCA-_-DSA-_-CategoryPages-_-NA&gclid=Cj0KEQjw-73GBRCC7KODl9zToJMBEiQAj1JgfxBICDaeKat

https://www.amazon.ca/Type-Adapter-EZOPower-Micro-Female/dp/B013IZMZ3I

https://oneplus.net/ca_en/oneplus-otg-cable

Additional Links

https://www.allaboutcircuits.com/projects/communicate-with-your-arduino-through-android/

3. Conclusion

The Software Controlled Drone project will meet all the requirements specified in the technical report. We have created hardware that is capable of controlling and communicating with the Eachine H8 mini drone. The MYSQL database is able to record the user's flight duration, the date of when the drone has been flown and as well as the user's account information such as first and last name, username and password. The android application is capable of allowing users to sign in or sign up for an account if they do not have one and allow users to control the Eachine H8 drone. The android application is also capable of updating the database with the user's personal runtime information and giving the option to users where they will be able to view their flight information within the app. The web interface is capable of allowing users to sign up and sign in to their account and view their drone's flight information which is stored in the database.

Initially, the hardware and software component of the project were going to be merged by means of Bluetooth; however, after some research and testing on modules, we have come up with a conclusion that the NRF24L01 transceiver and Bluetooth Module HC-05 cannot work together simultaneously due to frequency interference.

4. Appendices

4.1 Eachine H8 Arduino Sketch

```
/*
 ##########################################
 #####   MultiProtocol nRF24L01 Tx   ######
 ##########################################
 #       by goebish on rcgroups         #
 #                                      #
 #   Parts of this project are derived    #
 #    from existing work, thanks to:     #
 #                                      #
 #   - PhracturedBlue for DeviationTX     #
 #   - victzh for XN297 emulation layer   #
 #   - Hasi for Arduino PPM decoder       #
 #   - hexfet, midelic, closedsink ...    #
 ##########################################
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License.
If not, see <http://www.gnu.org/licenses/>.
*/
#include <util/atomic.h>
#include <EEPROM.h>
#include ”iface_nrf24l01.h”
// ############ Wiring ################
#define PPM_pin   2  // PPM in
//SPI Comm.pins with nRF24L01
#define MOSI_pin  3  // MOSI - D3
#define SCK_pin   4  // SCK  - D4
#define CE_pin    5  // CE   - D5
#define MISO_pin  A0 // MISO - A0
#define CS_pin    A1 // CS   - A1
```

```
#define ledPin    13 // LED  - D13
//Analog stick wiring
const int X_pin1 = A2; // first analog stick, x pin connected to A2
const int Y_pin1 = A3; // first analog stick, y pin connected to A3
const int X_pin2 = A4; // second analog stick, x pin connected to A4
const int Y_pin2 = A5; // second analog stick, y pin connected to A5
int throttle = 0;
int rudder = 0;
int aileron = 0;
int elevator = 0;
int serialAvailable = 1;
int s = 0;
char incomingByte;
// SPI outputs
#define MOSI_on PORTD |= _BV(3)  // PD3
#define MOSI_off PORTD &= ~_BV(3)// PD3
#define SCK_on PORTD |= _BV(4)   // PD4
#define SCK_off PORTD &= ~_BV(4) // PD4
#define CE_on PORTD |= _BV(5)    // PD5
#define CE_off PORTD &= ~_BV(5)  // PD5
#define CS_on PORTC |= _BV(1)    // PC1
#define CS_off PORTC &= ~_BV(1)  // PC1
// SPI input
#define  MISO_on (PINC & _BV(0)) // PC0
#define RF_POWER TX_POWER_80mW
// tune ppm input for ”special” transmitters
// #define SPEKTRUM // TAER, 1100-1900, AIL & RUD reversed
// PPM stream settings
#define CHANNELS 12 // number of channels in ppm stream, 12 ideally
enum chan_order{
   THROTTLE,
   AILERON,
   ELEVATOR,
   RUDDER,
   AUX1,  // (CH5)  led light, or 3 pos. rate on CX-10, H7, or inverted flight on H101
   AUX2,  // (CH6)  flip control
   AUX3,  // (CH7)  still camera (snapshot)
   AUX4,  // (CH8)  video camera
   AUX5,  // (CH9)  headless
```

```
  AUX6,  // (CH10) calibrate Y (V2x2), pitch trim (H7), RTH (Bayang, H20), 360deg flip mode (H8-3D, H22)
  AUX7,  // (CH11) calibrate X (V2x2), roll trim (H7)
  AUX8,  // (CH12) Reset / Rebind
};
#define PPM_MIN 1000
#define PPM_SAFE_THROTTLE 1050
#define PPM_MID 1500
#define PPM_MAX 2000
#define PPM_MIN_COMMAND 1300
#define PPM_MAX_COMMAND 1700
#define GET_FLAG(ch, mask) (ppm[ch] > PPM_MAX_COMMAND ? mask : 0)
#define GET_FLAG_INV(ch, mask) (ppm[ch] < PPM_MIN_COMMAND ? mask : 0)
// supported protocols
enum {
  PROTO_V2X2 = 0,    // WLToys V2x2, JXD JD38x, JD39x, JJRC H6C, Yizhan Tarantula X6 ...
  PROTO_CG023,       // EAchine CG023, CG032, 3D X4
  PROTO_CX10_BLUE,   // Cheerson CX-10 blue board, newer red board, CX-10A, CX-10C, Floureon FX-10, CX-Stars (todo: add DM007 variant)
  PROTO_CX10_GREEN,  // Cheerson CX-10 green board
  PROTO_H7,          // EAchine H7, MoonTop M99xx
  PROTO_BAYANG,      // EAchine H8(C) mini, H10, BayangToys X6, X7, X9, JJRC JJ850, Floureon H101
  PROTO_SYMAX5C1,    // Syma X5C-1 (not older X5C), X11, X11C, X12
  PROTO_YD829,       // YD-829, YD-829C, YD-822 ...
  PROTO_H8_3D,       // EAchine H8 mini 3D, JJRC H20, H22
  PROTO_MJX,         // MJX X600 (can be changed to Weilihua WLH08, X800 or H26D)
  PROTO_SYMAXOLD,    // Syma X5C, X2
   PROTO_HISKY,          // HiSky RXs, HFP80, HCP80/100, FBL70/80/90/100, FF120, HMX120, WLToys v933/944/955 ...
  PROTO_KN,          // KN (WLToys variant) V930/931/939/966/977/988
 PROTO_YD717,    // Cheerson CX-10 red (older version)/CX11/CX205/CX30, JXD389/390/391/393, SH6057/6043/6044/6046/6047, FY326Q7, WLToys v252 Pro/v343, XinXun X28/X30/X33/X39/X40
  PROTO_FQ777124,    // FQ777-124 pocket drone
  PROTO_E010,        // EAchine E010, NiHui NH-010, JJRC H36 mini
  PROTO_END
};
// EEPROM locationss
enum{
  ee_PROTOCOL_ID = 0,
  ee_TXID0,
```

```
  ee_TXID1,
  ee_TXID2,
  ee_TXID3
};
uint8_t transmitterID[4];
uint8_t current_protocol;
static volatile bool ppm_ok = false;
uint8_t packet[32];
static bool reset=true;
volatile uint16_t Servo_data[12];
static uint16_t ppm[12] = {PPM_MIN,PPM_MIN,PPM_MIN,PPM_MIN,PPM_MID,PPM_MID,
                           PPM_MID,PPM_MID,PPM_MID,PPM_MID,PPM_MID,PPM_MID,};
void setup()
{
  randomSeed((analogRead(A4) & 0x1F) | (analogRead(A5) << 5));
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW); //start LED off
  pinMode(PPM_pin, INPUT);
  pinMode(MOSI_pin, OUTPUT);
  pinMode(SCK_pin, OUTPUT);
  pinMode(CS_pin, OUTPUT);
  pinMode(CE_pin, OUTPUT);
  pinMode(MISO_pin, INPUT);
  // PPM ISR setup
  //attachInterrupt(digitalPinToInterrupt(PPM_pin), ISR_ppm, CHANGE);
  TCCR1A = 0;  //reset timer1
  TCCR1B = 0;
  TCCR1B |= (1 << CS11);  //set timer1 to increment every 1 us @ 8MHz, 0.5 us @16MHz
  set_txid(false);
  Serial.begin(115200);
}
void loop()
{
    if(Serial.available()){
    if (Serial.available() > 4){
      if (Serial.read()!=23){
        throttle = rudder = aileron = elevator = 0;
      }
      else {
```

```
    throttle=Serial.read();//throttle
    rudder=Serial.read();//yaw
    aileron=Serial.read();//roll
    elevator=Serial.read();//pitch
   }
   update_ppm2();
 }
}
uint32_t timeout=0;
// reset / rebind
if(reset || ppm[AUX8] > PPM_MAX_COMMAND) {
   reset = false;
   Serial.println("Selecting Protocol");
   current_protocol = PROTO_BAYANG;
   //selectProtocol();
   Serial.println("Selected Protocol");
   NRF24L01_Reset();
   Serial.println("nrf24l01 reset");
   NRF24L01_Initialize();
   Serial.println("nrf24l01 init");
   init_protocol();
   Serial.println("init_protocol complete");
}
// process protocol
switch(current_protocol) {
   case PROTO_CG023:
   case PROTO_YD829:
      timeout = process_CG023();
      break;
   case PROTO_V2X2:
      timeout = process_V2x2();
      break;
   case PROTO_CX10_GREEN:
   case PROTO_CX10_BLUE:
      timeout = process_CX10();
      break;
   case PROTO_H7:
      timeout = process_H7();
      break;
```

```
        case PROTO_BAYANG:
            timeout = process_Bayang();
            break;
        case PROTO_SYMAX5C1:
        case PROTO_SYMAXOLD:
            timeout = process_SymaX();
            break;
        case PROTO_H8_3D:
            timeout = process_H8_3D();
            break;
        case PROTO_MJX:
        case PROTO_E010:
            timeout = process_MJX();
            break;
        case PROTO_HISKY:
            timeout = process_HiSky();
            break;
        case PROTO_KN:
            timeout = process_KN();
            break;
        case PROTO_YD717:
            timeout = process_YD717();
            break;
        case PROTO_FQ777124:
            timeout = process_FQ777124();
            break;
    }
    // updates ppm values out of ISR
    update_ppm();
    // wait before sending next packet
    while(micros() < timeout)
    {  };
}
void set_txid(bool renew)
{
    uint8_t i;
    for(i=0; i<4; i++)
        transmitterID[i] = EEPROM.read(ee_TXID0+i);
    if(renew || (transmitterID[0]==0xFF && transmitterID[1]==0x0FF)) {
```

```
      for(i=0; i<4; i++) {
        transmitterID[i] = random() & 0xFF;
        EEPROM.update(ee_TXID0+i, transmitterID[i]);
      }
    }
}
void selectProtocol()
{
    // wait for multiple complete ppm frames
    ppm_ok = false;
    //uint8_t count = 10;
    //while(count) {
      //while(!ppm_ok) {} // wait
      //update_ppm();
      //if(ppm[AUX8] < PPM_MAX_COMMAND) // reset chan released
        //count--;
      //ppm_ok = false;
    //}
    // startup stick commands
    if(ppm[RUDDER] < PPM_MIN_COMMAND)       // Rudder left
      set_txid(true);                   // Renew Transmitter ID
    // protocol selection
    // Rudder right + Aileron right + Elevator down
     if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[AILERON] > PPM_MAX_COMMAND &&
ppm[ELEVATOR] < PPM_MIN_COMMAND)
      current_protocol = PROTO_E010; // EAchine E010, NiHui NH-010, JJRC H36 mini
    // Rudder right + Aileron right + Elevator up
     else if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[AILERON] > PPM_MAX_COMMAND
&& ppm[ELEVATOR] > PPM_MAX_COMMAND)
      current_protocol = PROTO_FQ777124; // FQ-777-124
    // Rudder right + Aileron left + Elevator up
     else if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[AILERON] < PPM_MIN_COMMAND
&& ppm[ELEVATOR] > PPM_MAX_COMMAND)
       current_protocol = PROTO_YD717; // Cheerson CX-10 red (older version)/CX11/CX205/CX30,
JXD389/390/391/393, SH6057/6043/6044/6046/6047, FY326Q7, WLToys v252 Pro/v343, XinXun
X28/X30/X33/X39/X40
    // Rudder right + Aileron left + Elevator down
     else if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[AILERON] < PPM_MIN_COMMAND
&& ppm[ELEVATOR] < PPM_MIN_COMMAND)
      current_protocol = PROTO_KN; // KN (WLToys variant) V930/931/939/966/977/988
    // Rudder right + Elevator down
```

```
  else if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[ELEVATOR] < PPM_MIN_COMMAND)
    current_protocol = PROTO_HISKY; // HiSky RXs, HFP80, HCP80/100, FBL70/80/90/100, FF120, HMX120, WLToys v933/944/955 ...
  // Rudder right + Elevator up
  else if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[ELEVATOR] > PPM_MAX_COMMAND)
      current_protocol = PROTO_SYMAXOLD; // Syma X5C, X2 ...
  // Rudder right + Aileron right
  else if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[AILERON] > PPM_MAX_COMMAND)
      current_protocol = PROTO_MJX; // MJX X600, other sub protocols can be set in code
  // Rudder right + Aileron left
  else if(ppm[RUDDER] > PPM_MAX_COMMAND && ppm[AILERON] < PPM_MIN_COMMAND)
      current_protocol = PROTO_H8_3D; // H8 mini 3D, H20 ...
  // Elevator down + Aileron right
  else if(ppm[ELEVATOR] < PPM_MIN_COMMAND && ppm[AILERON] > PPM_MAX_COMMAND)
      current_protocol = PROTO_YD829; // YD-829, YD-829C, YD-822 ...
  // Elevator down + Aileron left
  else if(ppm[ELEVATOR] < PPM_MIN_COMMAND && ppm[AILERON] < PPM_MIN_COMMAND)
      current_protocol = PROTO_SYMAX5C1; // Syma X5C-1, X11, X11C, X12
  // Elevator up + Aileron right
  else if(ppm[ELEVATOR] > PPM_MAX_COMMAND && ppm[AILERON] > PPM_MAX_COMMAND)
      current_protocol = PROTO_BAYANG;    // EAchine H8(C) mini, BayangToys X6/X7/X9, JJRC JJ850 ...
  // Elevator up + Aileron left
  else if(ppm[ELEVATOR] > PPM_MAX_COMMAND && ppm[AILERON] < PPM_MIN_COMMAND)
      current_protocol = PROTO_H7;       // EAchine H7, MT99xx
  // Elevator up
  else if(ppm[ELEVATOR] > PPM_MAX_COMMAND)
      current_protocol = PROTO_V2X2;      // WLToys V202/252/272, JXD 385/388, JJRC H6C ...
  // Elevator down
  else if(ppm[ELEVATOR] < PPM_MIN_COMMAND)
      current_protocol = PROTO_CG023;       // EAchine CG023/CG031/3D X4, (todo :ATTOP YD-836/YD-836C) ...
  // Aileron right
  else if(ppm[AILERON] > PPM_MAX_COMMAND)
      current_protocol = PROTO_CX10_BLUE;  // Cheerson CX10(blue pcb, newer red pcb)/CX10-A/CX11/CX12 ...
  // Aileron left
  else if(ppm[AILERON] < PPM_MIN_COMMAND)
      current_protocol = PROTO_CX10_GREEN;  // Cheerson CX10(green pcb)...
  // read last used protocol from eeprom
  else
```

```
    current_protocol = constrain(EEPROM.read(ee_PROTOCOL_ID),0,PROTO_END-1);
  // update eeprom
  EEPROM.update(ee_PROTOCOL_ID, current_protocol);
  // wait for safe throttle
  while(ppm[THROTTLE] > PPM_SAFE_THROTTLE) {
    delay(100);
    update_ppm();
  }
}
void init_protocol()
{
  switch(current_protocol) {
    case PROTO_CG023:
    case PROTO_YD829:
      CG023_init();
      CG023_bind();
      break;
    case PROTO_V2X2:
      V2x2_init();
      V2x2_bind();
      break;
    case PROTO_CX10_GREEN:
    case PROTO_CX10_BLUE:
      CX10_init();
      CX10_bind();
      break;
    case PROTO_H7:
      H7_init();
      H7_bind();
      break;
    case PROTO_BAYANG:
      Bayang_init();
      Bayang_bind();
      Serial.println("Bayang Protocol Selected");
      break;
    case PROTO_SYMAX5C1:
    case PROTO_SYMAXOLD:
      Symax_init();
      break;
```

```
      case PROTO_H8_3D:
        H8_3D_init();
        H8_3D_bind();
        break;
      case PROTO_MJX:
      case PROTO_E010:
        MJX_init();
        MJX_bind();
        break;
      case PROTO_HISKY:
        HiSky_init();
        break;
      case PROTO_KN:
        kn_start_tx(true); // autobind
        break;
      case PROTO_YD717:
        YD717_init();
        break;
      case PROTO_FQ777124:
        FQ777124_init();
        FQ777124_bind();
        break;
    }
}
// update ppm values out of ISR
void update_ppm()
{
  ppm[RUDDER] = map(analogRead(A2), 0, 1023, 1000, 2000);
  ppm[THROTTLE] = map(analogRead(A3), 0, 1023, 1000, 2000);
  ppm[AILERON] = map(analogRead(A4), 0, 1023, 1000, 2000);
  ppm[ELEVATOR] = map(analogRead(A5), 0, 1023, 1000, 2000);
  ppm[THROTTLE] = 3000-ppm[THROTTLE];
  ppm[ELEVATOR] = 3000-ppm[ELEVATOR];
}
void update_ppm2()
{
  ppm[RUDDER] = map(analogRead(rudder), 0, 1023, 1000, 2000);
  ppm[THROTTLE] = map(analogRead(throttle), 0, 1023, 1000, 2000);
  ppm[AILERON] = map(analogRead(aileron), 0, 1023, 1000, 2000);
```

```
  ppm[ELEVATOR] = map(analogRead(elevator), 0, 1023, 1000, 2000);
  ppm[THROTTLE] = 3000-ppm[THROTTLE];
  ppm[ELEVATOR] = 3000-ppm[ELEVATOR];
}
```

4.2 Android Application

4.2.1 LoginActivity.java

```
package com.example.softwarecontrolleddrone;

import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.net.Uri;
import android.os.AsyncTask;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

public class LoginActivity extends AppCompatActivity implements View.OnClickListener{

   AlertDialog alertDialog;
    Context context = this;

   EditText editText1;
    EditText editText2;
    Button loginButton;
    Button registerButton;

   String username, password;

   /*SharedPreferences for remember me checkbox*/
private SharedPreferences loginPreferences;
    SharedPreferences.Editor editor;
    private CheckBox checkBox;
    boolean saveLogin;
```

```
   /*SharedPreferences for passing the username*/
private SharedPreferences uNamePreferences;
    SharedPreferences.Editor uNameEditor;

   @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        Log.d("ADebugTag", "Value: " + getResources().getBoolean(R.bool.portrait_only));

      if(getResources().getBoolean(R.bool.portrait_only)){
          // setContentView(R.layout.activity_controller);
Log.d("ADebugTag", "Value: " + getResources().getBoolean(R.bool.portrait_only));
          setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        }

      editText1 = (EditText) findViewById(R.id.usernameField);
        editText2 = (EditText) findViewById(R.id.passwordField);
        checkBox = (CheckBox)findViewById(R.id.rememberMeCheckBox);

      loginButton = (Button) findViewById(R.id.loginButton);
        loginButton.setOnClickListener(LoginActivity.this);

      registerButton = (Button)findViewById(R.id.registerButton);

      /*Preferences for passing the username*/
uNamePreferences = getSharedPreferences("uNamePrefs", MODE_PRIVATE);
        uNameEditor = uNamePreferences.edit();

      /*Preferences for remember me checkbox*/
loginPreferences = getSharedPreferences("loginPrefs", MODE_PRIVATE);
        editor = loginPreferences.edit();

      saveLogin = loginPreferences.getBoolean("saveLogin", false);

      if(saveLogin == true)
        {
            editText1.setText(loginPreferences.getString("username", ""));
        }
    }

   public void onClick(View view)
    {
        username = editText1.getText().toString();
        password = editText2.getText().toString();

      BackgroundTask backgroundTask = new BackgroundTask();
        backgroundTask.execute(username, password);
    }

   class BackgroundTask extends AsyncTask<String, Void, String>
    {
        String login_url;
        boolean loginVerified = false;

      @Override
        protected void onPreExecute()
        {
        }

      @Override
        protected String doInBackground(String... args)
        {
            String username, password;
            login_url = "http://softwarecontrolleddrone.esy.es/Login.php";
```

```
        username = args[0];
          password = args[1];

        try
{
           URL url = new URL(login_url);

          //Make a request to the url
HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
              httpURLConnection.setRequestMethod("POST");
             httpURLConnection.setDoOutput(true);
              httpURLConnection.setDoInput(true);

          OutputStream outputStream = httpURLConnection.getOutputStream();
             BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream,
"UTF-8"));

                  String data_info = URLEncoder.encode("username", "UTF-8") + "=" + URLEn-
coder.encode(username, "UTF-8") + "&" +
                     URLEncoder.encode("password", "UTF-8") + "=" + URLEncoder.encode(password,
"UTF-8");

          bufferedWriter.write(data_info);
             bufferedWriter.flush();
             bufferedWriter.close();

          outputStream.close();

          InputStream inputStream = httpURLConnection.getInputStream();
             BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream,
"UTF-8"));

          String response = "";
             String line = "";

          while((line = bufferedReader.readLine()) != null)
             {
                 response += line;
             }

          bufferedReader.close();
             inputStream.close();
             httpURLConnection.disconnect();
             loginVerified = true;
            return response;

        }
          catch(MalformedURLException e)
          {
             e.printStackTrace();
          }
          catch(IOException e)
          {
             e.printStackTrace();
          }

         return null;
        }

       @Override
        protected void onPostExecute(String result)
        {
          if(result!=null && result.equalsIgnoreCase("Login Successful"))
          {
             Intent intent = new Intent(context, MenuActivity.class);
```

```
            if (checkBox.isChecked()) {
                editor.putBoolean("saveLogin", true);
                editor.putString("username", editText1.getText().toString());
                editor.commit();
            } else {
                editor.clear();
                editor.commit();
            }
        //Passing the username to the MenuActivity
String passUsername1 = editText1.getText().toString();
            uNameEditor.putString("usernamePassed1", passUsername1);
            uNameEditor.commit();

        startActivity(intent);
        }
        else{
            new AlertDialog.Builder(context)
                    .setIcon(android.R.drawable.ic_dialog_alert)
                    .setTitle(R.string.dialogMsg5)
                    .setMessage(result)
                    .setPositiveButton("Continue", new DialogInterface.OnClickListener()
                    {
                      @Override
                        public void onClick(DialogInterface dialog, int which) {
                            dialog.dismiss();
                        }
                    })
                    .show();
        }
    }
  }

  public void RegisterAccount(View view)
  {
      Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
      startActivity(intent);
  }

  public static boolean isTablet(Context context) {
      return (context.getResources().getConfiguration().screenLayout
& Configuration.SCREENLAYOUT_SIZE_MASK)
              >= Configuration.SCREENLAYOUT_SIZE_LARGE;
  }

  @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      Intent intent = null;
      switch (item.getItemId()) {
          case R.id.help:
              intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.ca/"));
              startActivity(intent);
              break;

       case R.id.scd:
            intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.litehawk.ca/highroller/"));
```

```
            startActivity(intent);
            break;
        case R.id.drone:
            intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.dronestoronto.com/"));
            startActivity(intent);
            break;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onBackPressed() {
        new AlertDialog.Builder(this)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setTitle(R.string.dialogMsg)
            .setMessage(R.string.dialogMsg2)
            .setPositiveButton("Yes", new DialogInterface.OnClickListener()
            {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    finish();
                }

            })
            .setNegativeButton("No", null)
            .show();
    }
}
```

4.2.2 RegisterActivity.java

```
package com.example.softwarecontrolleddrone;

import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.AsyncTask;
import android.renderscript.ScriptGroup;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
```

```
import java.net.URL;
import java.net.URLEncoder;

public class RegisterActivity extends AppCompatActivity {

    Context context = this;

    EditText registerFirstName;
     EditText registerLastName;
     EditText registerUsername;
     EditText registerPassword;
     Button registerButton2;
     String firstName, lastName, username, password;

    @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_register);

       registerFirstName = (EditText)findViewById(R.id.registerFirstName);
         registerLastName = (EditText)findViewById(R.id.registerLastName);
         registerUsername = (EditText)findViewById(R.id.registerUsername);
         registerPassword = (EditText)findViewById(R.id.registerPassword);

       registerButton2 = (Button)findViewById(R.id.registerButton2);
         registerButton2.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View v) {

             firstName = registerFirstName.getText().toString();
                 lastName = registerLastName.getText().toString();
                 username = registerUsername.getText().toString();
                 password = registerPassword.getText().toString();

             BackgroundTask backgroundTask = new BackgroundTask();

             boolean firstnameTest = checkFirstNameRegister(firstName);
                 if(firstnameTest == false)
                 {
                     Toast.makeText(RegisterActivity.this, R.string.errorInput, Toast.LENGTH_SHORT)
                             .show();
                 }

             boolean lastnameTest = checkLastNameRegister(lastName);
                 if(lastnameTest == false)
                {
                     Toast.makeText(RegisterActivity.this, R.string.errorInput, Toast.LENGTH_SHORT)
                             .show();
                 }

             boolean usernameTest = checkUsernameRegister(username);
                 if(usernameTest == false)
                 {
                     Toast.makeText(RegisterActivity.this, R.string.errorName, Toast.LENGTH_SHORT)
                             .show();
                 }

             boolean passwordTest = checkPasswordRegister(password);
                 if(passwordTest == false)
                 {
                     Toast.makeText(RegisterActivity.this, R.string.errorPassword, Toast.LENGTH_SHORT)
                             .show();
                 }
```

```
            if(firstnameTest && lastnameTest && usernameTest && passwordTest)
                {
                    backgroundTask.execute(firstName, lastName, username, password);
                }

        }
        });
    }

    public boolean checkFirstNameRegister(String registeredFirstName)
    {
        if(registeredFirstName.length() >= 2 && registeredFirstName.matches("[a-zA-Z]+(\\\\-[a-zA-Z]+)?"))
            return true;
        else
return false;
    }

    public boolean checkLastNameRegister(String registeredLastName)
    {
        if(registeredLastName.length() >= 2 && registeredLastName.matches("[a-zA-Z]+(\\\\-[a-zA-Z]+)?"))
            return true;
        else
return false;
    }

    public boolean checkUsernameRegister(String registeredUsername) {
        if (registeredUsername.length() >= 4 && registeredUsername.matches("[a-zA-Z0-9]+(\\\\-[a-zA-Z0-9]+)?"))
            return true;
        else
return false;
    }

    public boolean checkPasswordRegister(String registeredPassword) {
        if (registeredPassword.length() >= 6 && registeredPassword.matches("[a-zA-Z0-9]+(\\-[a-zA-Z0-9]+)?"))
            return true;
        else
return false;
    }

    class BackgroundTask extends AsyncTask<String, Void, String>
    {

      AlertDialog alertDialog;

      String link;

      @Override
        protected void onPreExecute()
        {
        }

      @Override
        protected String doInBackground(String... args)
        {
            link = "http://softwarecontrolleddrone.esy.es/AddMember.php";

         String firstName, lastName, username, password;

         firstName = args[0];
            lastName = args[1];
```

```
            username = args[2];
            password = args[3];
        try
{
                URL url = new URL(link);
            //Opens connection to url
HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);
            OutputStream outputStream = httpURLConnection.getOutputStream();
                BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream,
"UTF-8"));
            //Encoded data to be written to the url
String data_string = URLEncoder.encode("firstName", "UTF-8") + "=" + URLEncoder.encode(firstName,
"UTF-8") + "&" +
                        URLEncoder.encode("lastName", "UTF-8") + "=" + URLEncoder.encode(lastName,
"UTF-8") + "&" +
                        URLEncoder.encode("username", "UTF-8") + "=" + URLEncoder.encode(username,
"UTF-8") + "&" +
                        URLEncoder.encode("password", "UTF-8") + "=" + URLEncoder.encode(password,
"UTF-8");
            bufferedWriter.write(data_string);
                bufferedWriter.flush();
                bufferedWriter.close();
                outputStream.close();
            InputStream inputStream = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream,
"UTF-8"));
            String response = "";
                String line = "";
            while((line = bufferedReader.readLine()) != null)
                {
                    response += line;
                }
            bufferedReader.close();
                inputStream.close();
                httpURLConnection.disconnect();
                return response;
        }
            catch(MalformedURLException e){
                e.printStackTrace();
            }
            catch(IOException e){
                e.printStackTrace();
            }
         return null;
        }
       @Override
         protected void onPostExecute(String result)
         {
           if(result!=null && result.equalsIgnoreCase("Registration Success"))
            {
                Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
```

```
        Toast.makeText(RegisterActivity.this, R.string.RegistrationSuccessful, Toast.LENGTH_SHORT)
                    .show();
            startActivity(intent);
        }
        else{
            new AlertDialog.Builder(context)
                    .setIcon(android.R.drawable.ic_dialog_alert)
                    .setTitle(R.string.dialogMsg3)
                    .setMessage(result)
                    .setPositiveButton("Continue", new DialogInterface.OnClickListener()
                    {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            dialog.dismiss();
                        }
                    })
                    .show();
        }

    }
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    Intent intent = null;
      switch (item.getItemId()) {
          case R.id.help:
              intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.ca/"));
              startActivity(intent);
              break;

       case R.id.scd:
              intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.litehawk.ca/highroller/"));
              startActivity(intent);
              break;

       case R.id.drone:
              intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.dronestoronto.com/"));
              startActivity(intent);
              break;

    }
      return super.onOptionsItemSelected(item);
  }

}
```

4.2.3 MenuActivity.java

```
package com.example.softwarecontrolleddrone;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.net.Uri;
```

```
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

public class MenuActivity extends AppCompatActivity {

   Button controllerButton, databaseButton;
    ImageView image;
   /*SharedPreferences for accessing FlightActivity*/
SharedPreferences accessPreference;
    SharedPreferences.Editor editor;
    boolean check;

   /*SharedPreferences for passing/receiving the username*/
SharedPreferences uNamePreferences;
    SharedPreferences.Editor uNameEditor;
    String recvUsername1;

   final Context context = this;

   @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_menu);

      /*SharedPreferences for passing the username*/
uNamePreferences = getSharedPreferences("uNamePrefs", MODE_PRIVATE);
        uNameEditor = uNamePreferences.edit();

      recvUsername1 = uNamePreferences.getString("usernamePassed1", "");

      /*SharedPreferences for accessing the flightActivity*/
accessPreference = getSharedPreferences("accessPrefs", MODE_PRIVATE);
        editor = accessPreference.edit();

      check = accessPreference.getBoolean("check", false);

      if(getResources().getBoolean(R.bool.portrait_only)){
            setContentView(R.layout.activity_controller);
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        }

      image = (ImageView) findViewById(R.id.dronePicture1);

      final int[] imageArray = { R.drawable.drone, R.drawable.drone_90,
              R.drawable.drone_180, R.drawable.drone_270,
             };

      final Handler handler = new Handler();
        Runnable runnable = new Runnable() {
            int i = 0;

          public void run() {
              image.setImageResource(imageArray[i]);
              i++;
              if (i > imageArray.length - 1) {
                  i = 0;
              }
              handler.postDelayed(this, 50);
```

```
            }
        };
        handler.postDelayed(runnable, 50);

    //image.startAnimation(
// AnimationUtils.loadAnimation(MenuActivity.this, R.anim.drone_rotation_anim) );

    controllerButton = (Button)findViewById(R.id.controllerButton);
      controllerButton.setOnClickListener(new View.OnClickListener(){

       @Override
          public void onClick(View v){

          Intent intent = new Intent(MenuActivity.this, ControllerActivity.class);

          //Passing username to ControllerActivity
intent.putExtra("usernamePassedContr", recvUsername1);

          startActivity(intent);

       }
      });

    databaseButton = (Button)findViewById(R.id.databaseButton);
      databaseButton.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {

          if(check == false){
                Toast.makeText(MenuActivity.this, R.string.emptyFlightActivity, Toast.LENGTH_SHORT)
                .show();
            }else {

            Intent intent = new Intent(MenuActivity.this, FlightsActivity.class);
                //intent.putExtra("usernamePassedFlights", recvUsername1);
uNameEditor.putString("namePassToFlights", recvUsername1);
                uNameEditor.commit();

            startActivity(intent);
             }

       }
      });
  }
  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
      Intent intent = null;
      switch (item.getItemId()) {
          case R.id.help:
              intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.ca/"));
              startActivity(intent);
              break;

       case R.id.scd:
        intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.litehawk.ca/highroller/"));
              startActivity(intent);
              break;

       case R.id.drone:
              intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.dronestoronto.com/"));
```

```
                startActivity(intent);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

4.2.4 ControllerActivity.java

```
package com.example.softwarecontrolleddrone;

import android.Manifest;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.database.sqlite.SQLiteDatabase;
import android.hardware.usb.UsbDevice;
import android.hardware.usb.UsbDeviceConnection;
import android.hardware.usb.UsbManager;
import android.icu.util.Calendar;
import android.icu.util.Output;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Handler;
import android.os.SystemClock;
import android.preference.PreferenceManager;
import android.provider.Settings;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.Chronometer;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.RadioGroup;
import android.widget.RelativeLayout;
```

```
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.felhr.usbserial.UsbSerialDevice;
import com.felhr.usbserial.UsbSerialInterface;
import com.google.android.gms.fitness.data.MapValue;

import org.w3c.dom.Text;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

public class ControllerActivity extends AppCompatActivity {

    /*Variables for internal DB*/
MySQLiteHelper mySQLiteHelper;
    SQLiteDatabase sqLiteDatabase;
    Context context = this;

    /*Variables for LED button*/
public static final String PREFS = "sharedPreferences";
    public static final String BRIGHTNESS = "brightness";
    public static Button b;
    boolean switch1;

   Chronometer chronometer;
    TextView timeText;
    String putFlightDuration;
    String formattedDate;
    private long timeWhenStopped = 0;

    /*SharedPreferences for accessing Flight Activity*/
SharedPreferences accessPreference;
    SharedPreferences.Editor editor;
    boolean check2;

   String recvUsername;

    /*Variables for joysticks*/
TextView txtX1, txtY1, txtX2, txtY2, textView;
    Button startButton, stopButton;
    DualJoystickView joystick;
    byte input[] = {23, 0, 0x7f, 0x7f, 0x7f};//default input

  public final String ACTION_USB_PERMISSION = "com.example.softwarecontrolleddrone.USB_PERM
    UsbManager usbManager;
    UsbDevice device;
    UsbSerialDevice serialPort;
    UsbDeviceConnection connection;
```

```
    @Override
     protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_controller);

       txtX1 = (TextView)findViewById(R.id.TextViewX1);
        txtY1 = (TextView)findViewById(R.id.TextViewY1);

       txtX2 = (TextView)findViewById(R.id.TextViewX2);
        txtY2 = (TextView)findViewById(R.id.TextViewY2);

       textView = (TextView)findViewById(R.id.textView);

       joystick = (DualJoystickView)findViewById(R.id.dualjoystickView);
        joystick.setOnJostickMovedListener(_listenerLeft, _listenerRight);

       usbManager = (UsbManager) getSystemService(this.USB_SERVICE);
        startButton = (Button) findViewById(R.id.buttonStart);
        stopButton = (Button) findViewById(R.id.buttonStop);

       setUiEnabled(false);
        IntentFilter filter = new IntentFilter();
        filter.addAction(ACTION_USB_PERMISSION);
        filter.addAction(UsbManager.ACTION_USB_DEVICE_ATTACHED);
        filter.addAction(UsbManager.ACTION_USB_DEVICE_DETACHED);
        registerReceiver(broadcastReceiver, filter);

       /*Retreiving the username that will be put into the database*/
recvUsername = getIntent().getStringExtra("usernamePassedContr");

       /*SharedPreferences for accessing the flight activity*/
accessPreference = getSharedPreferences("accessPrefs", MODE_PRIVATE);
        editor = accessPreference.edit();

       if(getResources().getBoolean(R.bool.portrait_only)){
            setContentView(R.layout.activity_controller);
           setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        }

       /*Retreiving the current date*/
java.util.Calendar c = java.util.Calendar.getInstance();
        SimpleDateFormat df = new SimpleDateFormat("dd-MMM-yyyy");
        formattedDate = df.format(c.getTime());

       //LED Button
b = (Button) findViewById(R.id.button);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(ControllerActivity.this, PopActivity.class));
            }
        });

       /*Chronometer and the TextView to display the time*/
chronometer = (Chronometer) findViewById(R.id.chronometer);
        timeText = (TextView) findViewById(R.id.timeDisplayed);
        chronometer.setVisibility(View.INVISIBLE);

    }

     UsbSerialInterface.UsbReadCallback mCallback = new UsbSerialInterface.UsbReadCallback() {
//Defining a Callback which triggers whenever data is read.
@Override
        public void onReceivedData(byte[] arg0)
        {
```

```
        }
    };
    private final BroadcastReceiver broadcastReceiver = new BroadcastReceiver() { //Broadcast
Receiver to automatically start and stop the Serial connection.
@Override
        public void onReceive(Context context, Intent intent) {
            try {
                if (intent.getAction().equals(ACTION_USB_PERMISSION)) {
                    boolean granted = intent.getExtras().getBoolean(UsbManager.EXTRA_PERMISSION_GRANTED)
                    if (granted) {
                        connection = usbManager.openDevice(device);
                        serialPort = UsbSerialDevice.createUsbSerialDevice(device, connection);
                        if (serialPort != null) {
                            if (serialPort.open()) { //Set Serial Connection Parameters.
setUiEnabled(true);
                                serialPort.setBaudRate(115200);
                                serialPort.setDataBits(UsbSerialInterface.DATA_BITS_8);
                                serialPort.setStopBits(UsbSerialInterface.STOP_BITS_1);
                                serialPort.setParity(UsbSerialInterface.PARITY_NONE);
                                serialPort.setFlowControl(UsbSerialInterface.FLOW_CONTROL_OFF);
                                serialPort.read(mCallback);
                            chronometer.setVisibility(View.VISIBLE);
                            tvAppend(textView, "Serial Connection Opened!\n");
                        } else {
                                Log.d("SERIAL", "PORT NOT OPEN");
                            }
                        } else {
                            Log.d("SERIAL", "PORT IS NULL");
                        }
                    } else {
                        Log.d("SERIAL", "PERM NOT GRANTED");
                    }
                } else if (intent.getAction().equals(UsbManager.ACTION_USB_DEVICE_ATTACHED))
{
                    onClickStart(startButton);
                } else if (intent.getAction().equals(UsbManager.ACTION_USB_DEVICE_DETACHED))
{
                    onClickStop(stopButton);
                }
            }catch(Exception e){}
        };
    };
    public void onClickStart(View view)
    {
        //byte a[] = {1};
//serialPort.write(a);
        editor.putBoolean("check", true);
        editor.commit();
        //chronometer.setVisibility(View.VISIBLE);
chronometer.setBase(SystemClock.elapsedRealtime());
        timeText.setVisibility(View.INVISIBLE);
        timeText.setText(R.string.zeroseconds);
        chronometer.start();
        try {
            HashMap<String, UsbDevice> usbDevices = usbManager.getDeviceList();
```

```
            if (!usbDevices.isEmpty()) {
                boolean keep = true;
                for (Map.Entry<String, UsbDevice> entry : usbDevices.entrySet()) {
                    device = entry.getValue();
                    int deviceVID = device.getVendorId();
                if (deviceVID == 0x2A03)//Arduino Vendor ID
{
                        PendingIntent pi = PendingIntent.getBroadcast(this, 0, new Intent(ACTION_USB_PERMISSIO
0);
                        usbManager.requestPermission(device, pi);
                        keep = false;
                    } else {
                        connection = null;
                        device = null;
                    }
              if (!keep)
                        break;
                }
            }
        }catch(Exception e){
            //Toast.makeText(MainActivity.this, "The Arduino Is Not Connected", Toast.LENGTH_SHORT).show();
}
    }
    public void onClickStop(View view)
    {
        byte input[] = {23, 0, 0x7f, 0x7f, 0x7f};
        serialPort.write(input);
        setUiEnabled(false);
        serialPort.close();
        tvAppend(textView, "\nSerial Connection Closed! \n");
      timeWhenStopped = chronometer.getBase() - SystemClock.elapsedRealtime();
        int seconds = (int) timeWhenStopped / 1000;
        timeText.setVisibility(View.VISIBLE);
        chronometer.setVisibility(View.INVISIBLE);
        //math.abs returns the absolute value of seconds
timeText.setText(Math.abs(seconds) + " Second(s)");
      chronometer.stop();
      putFlightDuration = timeText.getText().toString();
      mySQLiteHelper = new MySQLiteHelper(context);
        sqLiteDatabase = mySQLiteHelper.getWritableDatabase();
        mySQLiteHelper.putInformation(sqLiteDatabase, formattedDate, putFlightDuration);
        mySQLiteHelper.close();
      BackgroundTask backgroundTask = new BackgroundTask();
        backgroundTask.execute(formattedDate, putFlightDuration, recvUsername);
    }
    public void setUiEnabled(boolean bool)
    {
        startButton.setEnabled(!bool);
        stopButton.setEnabled(bool);
        textView.setEnabled(bool);
    }
    private void tvAppend(TextView tv, CharSequence text) {
        final TextView ftv = tv;
        final CharSequence ftext = text;
```

```
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                ftv.setText(ftext);
            }
        });
    }

    @Override
    protected void onStop() {
        super.onStop();
    }

    /*Joystick Listener for the left joystick*/
private JoystickMovedListener _listenerLeft = new JoystickMovedListener() {

        @Override
        public void OnMoved(int yaw, int throttle) {
            try {
                throttle = throttle + 128;
                yaw = yaw + 128;
                if(throttle >= 256)
                    throttle = 255;
                if(yaw >= 256)
                    yaw = 255;
                if(yaw >= 108 && yaw <= 148)
                    yaw = 127;

            txtX1.setText(Integer.toString(yaw));
                txtY1.setText(Integer.toString(throttle));
                input[1] = (byte) throttle;
                input[2] = (byte) yaw;
                serialPort.write(input);

            byte b[] = {0};
                serialPort.write(b);
            }catch(Exception e){
                //Toast.makeText(FlightController.this, "The Arduino Is Not Connected", Toast.LENGTH_SHORT).show()
}
        }

        @Override
        public void OnReleased() {
            try{

         }catch(Exception e){
            }
        }

        public void OnReturnedToCenter() {
            try{

         }catch(Exception e){
            }
        };
    };

    /*Joystick listener for the right joystick*/
private JoystickMovedListener _listenerRight = new JoystickMovedListener() {

        @Override
        public void OnMoved(int roll, int pitch) {
            try{
                pitch = pitch + 128;
```

```
                roll = roll + 128;
              if(pitch >= 256)
                  pitch = 255;
                if(pitch == 128)
                  pitch = 127;
                if(roll >= 256)
                  roll = 255;
                if(roll == 128)
                  roll = 127;

            txtX2.setText(Integer.toString(roll));
                txtY2.setText(Integer.toString(pitch));

            input[3] = (byte) pitch;//left or right
input[4] = (byte) roll;//forward or backward
serialPort.write(input);
            }catch(Exception e){
                //Toast.makeText(FlightController.this, "The Arduino Is Not Connected", Toast.LENGTH_SHORT).show()
}
        }

      @Override
        public void OnReleased() {
            try{

         }catch(Exception e){
            }
        }

      public void OnReturnedToCenter() {
            try{

         }catch(Exception e){
            }
        };
    };

   class BackgroundTask extends AsyncTask<String, Void, String>
    {
        String link;

      @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

      @Override
        protected String doInBackground(String... args)
        {
            link = "http://softwarecontrolleddrone.esy.es/FlightInfo2.php";

         String date, flightduration, username;

         date = args[0];
            flightduration = args[1];
            username = args[2];

         try{

            URL url = new URL(link);

            //Opens connection to url
HttpURLConnection httpURLConnection = (HttpURLConnection)url.openConnection();
                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);
```

```
            OutputStream outputStream = httpURLConnection.getOutputStream();
                BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream,
"UTF-8"));
            //Encoded data to be written to the URL
String data_string = URLEncoder.encode("date", "UTF-8") + "=" + URLEncoder.encode(date,
"UTF-8") + "&" +
                        URLEncoder.encode("flightduration", "UTF-8") + "=" + URLEncoder.encode(flightduration,
"UTF-8") + "&" +
                        URLEncoder.encode("username", "UTF-8") + "=" + URLEncoder.encode(username,
"UTF-8");
            bufferedWriter.write(data_string);
                bufferedWriter.flush();
                bufferedWriter.close();
                outputStream.close();
            InputStream inputStream = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream,
"UTF-8"));
            String response = "";
                String line = "";
            while((line = bufferedReader.readLine()) != null)
                {
                    response += line;
                }
            bufferedReader.close();
                inputStream.close();
                httpURLConnection.disconnect();
                return response;
            }
            catch(MalformedURLException e)
            {
                e.printStackTrace();
            }
            catch(IOException e)
            {
                e.printStackTrace();
            }
            return null;
        }
        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        Intent intent = null;
        switch (item.getItemId()) {
            case R.id.help:
                intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.ca/"));
```

```
            startActivity(intent);
            break;
        case R.id.scd:
            intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.litehawk.ca/highroller/"));
            startActivity(intent);
            break;
        case R.id.drone:
            intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.dronestoronto.com/"));
            startActivity(intent);
            break;
      }
      return super.onOptionsItemSelected(item);
   }
}
```

4.2.5 FlightActivity.java

```
package com.example.softwarecontrolleddrone;

import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.AsyncTask;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.List;

public class FlightsActivity extends AppCompatActivity {

   MySQLiteHelper mySQLiteHelper;
    SQLiteDatabase sqLiteDatabase;
    Context context = this;
```

```
    ListDataAdapter listDataAdapter;
    ListView listView;
    Button deleteInformationButton;
    String date, flightduration;

   /*SharedPreferences for accessing the FlightsActivity*/
SharedPreferences accessPreference;
    SharedPreferences.Editor editor;

   /*SharedPreferences for receiving the username*/
SharedPreferences uNamePreferences;
    SharedPreferences.Editor uNameEditor;

   String recvUsername;

   Cursor cursor;

   @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_flights);

      /*SharedPreferences for receiving the username passed from the MenuActivity*/
uNamePreferences = getSharedPreferences("uNamePrefs", MODE_PRIVATE);
        uNameEditor = uNamePreferences.edit();

      recvUsername = uNamePreferences.getString("namePassToFlights", null);

      //recvUsername = getIntent().getStringExtra("usernamePassedFlights");

      /*SharedPreferences for accessing the flightActivity*/
accessPreference = getSharedPreferences("accessPrefs", MODE_PRIVATE);
        editor = accessPreference.edit();

      if(getResources().getBoolean(R.bool.portrait_only)){
            setContentView(R.layout.activity_controller);
            setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        }

      listDataAdapter = new ListDataAdapter(context, R.layout.list_view_layout);
        listView = (ListView)findViewById(R.id.FlightInformationList);
        //set the listView to the adapter(listDataAdapter)
listView.setAdapter(listDataAdapter);

      mySQLiteHelper = new MySQLiteHelper(context);
        sqLiteDatabase = mySQLiteHelper.getReadableDatabase();

      cursor = mySQLiteHelper.getInformation(sqLiteDatabase);

      if(cursor.moveToFirst())
        {
            do{
                date = cursor.getString(0);
                flightduration = cursor.getString(1);

            DataProvider dataProvider = new DataProvider(date, flightduration);

            listDataAdapter.add(dataProvider);

         }while(cursor.moveToNext());
        }

      deleteInformationButton = (Button)findViewById(R.id.deleteInfoButton);
        deleteInformationButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v)
            {
                Intent intent = new Intent(FlightsActivity.this, MenuActivity.class);
```

```
            BackgroundTask backgroundTask = new BackgroundTask();
                backgroundTask.execute(recvUsername);

            editor.putBoolean("check", false);
                editor.commit();

            startActivity(intent);
            }
        });
    }

    class BackgroundTask extends AsyncTask<String, Void, String>
    {
        String link;

        @Override
        protected void onPreExecute()
        {
        }

        @Override
        protected String doInBackground(String... args)
        {
          link = "http://softwarecontrolleddrone.esy.es/DeleteFlightInfo.php";

         String username;

         username = args[0];

         try
{
                URL url = new URL(link);

            //Opens connection to the URL
HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);

            OutputStream outputStream = httpURLConnection.getOutputStream();
                BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream,
"UTF-8"));

            //Encoded data to be written to the url
String data_string = URLEncoder.encode("username", "UTF-8") + "=" + URLEncoder.encode(username,
"UTF-8");

            bufferedWriter.write(data_string);
                bufferedWriter.flush();
                bufferedWriter.close();
                outputStream.close();

            InputStream inputStream = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream,
"UTF-8"));

            String response = "";
                String line = "";

            while((line = bufferedReader.readLine()) != null)
                {
                    response += line;
                }

            bufferedReader.close();
                inputStream.close();
                httpURLConnection.disconnect();
```

```
            return response;
        }
        catch(MalformedURLException e)
        {
            e.printStackTrace();
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPostExecute(String result)
    {
        if(result != null && result.equalsIgnoreCase("Information Deleted Successfully"))
        {
            mySQLiteHelper = new MySQLiteHelper(context);
            sqLiteDatabase = mySQLiteHelper.getWritableDatabase();

        mySQLiteHelper.deleteInformation(sqLiteDatabase);
            mySQLiteHelper.close();

        Toast.makeText(FlightsActivity.this, result, Toast.LENGTH_SHORT)
                .show();
        }
        else
{
            Toast.makeText(FlightsActivity.this, result, Toast.LENGTH_SHORT)
                .show();
        }
    }
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
  }
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
      Intent intent = null;
    switch (item.getItemId()) {
        case R.id.help:
            intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.ca/"));
            startActivity(intent);
            break;

      case R.id.scd:
          intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.litehawk.ca/highroller/"));
            startActivity(intent);
            break;

      case R.id.drone:
            intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.dronestoronto.com/"));
            startActivity(intent);
            break;

    }
      return super.onOptionsItemSelected(item);
```

```
    }
}
```

4.2.6 PopActivity.java

```
package com.example.softwarecontrolleddrone;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.preference.SwitchPreference;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.SeekBar;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import java.util.prefs.Preferences;

public class PopActivity extends Activity {
    public static final String PREFS = "sharedPreferences";
    private Button save_button;
    public Switch switch1;
    private RadioGroup radioGroup;
    private RadioButton rb1;
    private RadioButton rb2;
    private RadioButton rb3;
    private static final String KEY_TEXT_VALUE = "button";
    String background;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (savedInstanceState != null) {
            CharSequence savedText = savedInstanceState.getCharSequence(KEY_TEXT_VALUE);
            //ControllerActivity.b.
}

        //window layout management
DisplayMetrics dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(dm);

        int width = dm.widthPixels;
        int height = dm.heightPixels;

        getWindow().setLayout((int) (width * .4), (int) (height * .5));

        //initializers
setContentView(R.layout.activity_pop);
        switch1 = (Switch) findViewById(R.id.switch1);
        radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
        rb1 = (RadioButton) findViewById(R.id.radio1);
```

```
        rb2 = (RadioButton) findViewById(R.id.radio2);
        rb3 = (RadioButton) findViewById(R.id.radio3);

      //Final values decalred above to simplify the code
final SharedPreferences sharedPreferences = getSharedPreferences(PREFS,0);
        final SharedPreferences.Editor editor = sharedPreferences.edit();

      //loading sets
switch1.setChecked(sharedPreferences.getBoolean("Switch",false));
        rb1.setChecked(sharedPreferences.getBoolean("rb1",false));
        rb2.setChecked(sharedPreferences.getBoolean("rb2",false));
        rb3.setChecked(sharedPreferences.getBoolean("rb3",false));

      save_button = (Button) findViewById(R.id.save);
        save_button.setOnClickListener(new View.OnClickListener() {

         @Override
            public void onClick(View v) {
                editor.putBoolean("Switch", switch1.isChecked());
                editor.putBoolean("rb1", rb1.isChecked());
               editor.putBoolean("rb2", rb2.isChecked());
                editor.putBoolean("rb3", rb3.isChecked());
                editor.commit();

            SharedPreferences sharedPreferences = getSharedPreferences(PREFS, 0);
                Boolean switch1 = sharedPreferences.getBoolean("Switch", false);
                Boolean rb1 = sharedPreferences.getBoolean("rb1", false);
                Boolean rb2 = sharedPreferences.getBoolean("rb2", false);
                Boolean rb3 = sharedPreferences.getBoolean("rb3", false);

            if(switch1) {
                    if(rb1){
                        ControllerActivity.b.setBackgroundResource(R.drawable.buttonshape_led_red);
                    }else if(rb2){
                        ControllerActivity.b.setBackgroundResource(R.drawable.buttonshape_led_green);
                    }else if(rb3){
                        ControllerActivity.b.setBackgroundResource(R.drawable.buttonshape_led_blue);
                    }else{
                      ControllerActivity.b.setBackgroundResource(R.color.white);
                        Toast.makeText(PopActivity.this, R.string.no_choice, Toast.LENGTH_SHORT)
                                .show();
                    }
                    ControllerActivity.b.setText(R.string.led_on);
                } else {
                    ControllerActivity.b.setText(R.string.led_off);
                    ControllerActivity.b.setBackgroundResource(R.drawable.buttonshape_led);
                }

            finish();
            }
        });
    }

}
```

4.2.7 TableData.java

```
package com.example.softwarecontrolleddrone;

public class TableData{
    public static abstract class DroneInfo
    {
```

```
        public static final String CURRENT_DATE = ”current_date”;
        public static final String FLIGHT_DURATION = ”flight_duration”;
        public static final String TABLE_NAME = ”DroneInfo”;
    }
}
```

4.2.8 MySQLiteHelper.java

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.support.design.widget.TabLayout;
import android.util.Log;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class MySQLiteHelper extends SQLiteOpenHelper {

   private static final String DATABASE_NAME = ”DroneDB”;
    private static final int DATABASE_VERSION = 1;

   public static final String DATABASE_CREATE =
            ”CREATE TABLE ” + TableData.DroneInfo.TABLE_NAME + ”(” + TableData.DroneInfo.CURRENT_DA
+ ” TEXT,” + TableData.DroneInfo.FLIGHT_DURATION + ” TEXT);”;

   public MySQLiteHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        Log.d(”MySQLiteHelper”, ”Database Created”);
    }

   @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DATABASE_CREATE);
        Log.d(”MySQLiteHelper”, ”Table Created”);
    }

   @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

   }

   public void putInformation(SQLiteDatabase db, String date, String flightDuration){

      ContentValues contentValues = new ContentValues();

      contentValues.put(TableData.DroneInfo.CURRENT_DATE, date);
        contentValues.put(TableData.DroneInfo.FLIGHT_DURATION, flightDuration);

      db.insert(TableData.DroneInfo.TABLE_NAME, null, contentValues);
        Log.d(”MySQLiteHelper”, ”one row inserted”);

   }

   public Cursor getInformation(SQLiteDatabase db)
    {
        Cursor cursor;

    String[] columnNames = {TableData.DroneInfo.CURRENT_DATE, TableData.DroneInfo.FLIGHT_DURATION

       cursor = db.query(TableData.DroneInfo.TABLE_NAME, columnNames, null, null, null, null,
null);
```

```
        return cursor;
    }
    public void deleteInformation(SQLiteDatabase db)
    {
        db.delete(TableData.DroneInfo.TABLE_NAME, null, null);
    }
}
```

4.2.9 DataProvider.java

```
package com.example.softwarecontrolleddrone;

public class DataProvider {

    private String date, flightDuration;

    public String getFlightDuration() {
        return flightDuration;
    }

    public void setFlightDuration(String flightDuration) {
        this.flightDuration = flightDuration;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {

        this.date = date;
    }

    public DataProvider(String date, String flightDuration)
    {
        this.date = date;
        this.flightDuration = flightDuration;
    }
}
```

4.2.10 ListDataAdapter.java

```
package com.example.softwarecontrolleddrone;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

public class ListDataAdapter extends ArrayAdapter {

    List list = new ArrayList<>();

    public ListDataAdapter(Context context, int resource) {
        super(context, resource);

    }
```

```
    static class LayoutHandler
    {
        TextView DATE, FLIGHTDURATION;
    }
    @Override
    public void add(Object object)
    {
        super.add(object);
      list.add(object);
    }
    @Override
    public int getCount()
    {
        return list.size();
    }
    @Override
    public Object getItem(int position)
    {
        return list.get(position);
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        View row = convertView;
        LayoutHandler layoutHandler;
      if(row == null)
        {
            LayoutInflater layoutInflater = (LayoutInflater)this.getContext().getSystemService(Context.LAYOUT_INFLA
            row = layoutInflater.inflate(R.layout.list_view_layout, parent, false);
         layoutHandler = new LayoutHandler();
            layoutHandler.DATE = (TextView) row.findViewById(R.id.displayDate);
            layoutHandler.FLIGHTDURATION = (TextView) row.findViewById(R.id.displayFlightDuration);
            row.setTag(layoutHandler);
        }
        else
{
            layoutHandler = (LayoutHandler) row.getTag();
        }
        DataProvider dataProvider = (DataProvider)this.getItem(position);
        layoutHandler.DATE.setText(dataProvider.getDate());
        layoutHandler.FLIGHTDURATION.setText(dataProvider.getFlightDuration());
      return row;
    }
}
```

4.2.11 JoystickView.java

```
package com.example.softwarecontrolleddrone;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
```

```
import android.util.Log;
import android.view.HapticFeedbackConstants;
import android.view.MotionEvent;
import android.view.View;

public class JoystickView extends View{
    public static final int INVALID_POINTER_ID = -1;

    // ============================================
// Private Members
// ============================================
private final boolean D = false;
    String TAG = "JoystickView";

    private Paint dbgPaint1;
    private Paint dbgPaint2;

    private Paint bgPaint;
    private Paint handlePaint;

    private int innerPadding;
    private int bgRadius;
    private int handleRadius;
    private int movementRadius;
    private int handleInnerBoundaries;

    private JoystickMovedListener moveListener;
    private JoystickClickedListener clickListener;

    //# of pixels movement required between reporting to the listener
private float moveResolution;

    private boolean yAxisInverted;
    private boolean autoReturnToCenter;

    //Max range of movement in user coordinate system
public final static int CONSTRAIN_BOX = 0;
    public final static int CONSTRAIN_CIRCLE = 1;
    private int movementConstraint;
    private float movementRange;

    public final static int COORDINATE_CARTESIAN = 0;    //Regular cartesian coordinates
public final static int COORDINATE_DIFFERENTIAL = 1;  //Uses polar rotation of 45 degrees
to calc differential drive paramaters
private int userCoordinateSystem;

    //Records touch pressure for click handling
private float touchPressure;
    private boolean clicked;
    private float clickThreshold;

    //Last touch point in view coordinates
private int pointerId = INVALID_POINTER_ID;
    private float touchX, touchY;

    //Last reported position in view coordinates (allows different reporting sensitivities)
private float reportX, reportY;

    //Handle center in view coordinates
private float handleX, handleY;

    //Center of the view in view coordinates
private int cX, cY;

    //Size of the view in view coordinates
private int dimX, dimY;
```

```
   //Cartesian coordinates of last touch point - joystick center is (0,0)
private int cartX, cartY;

   //Polar coordinates of the touch point from joystick center
private double radial;
    private double angle;

   //User coordinates of last touch point
private int userX, userY;

   //Offset co-ordinates (used when touch events are received from parent's coordinate origin)
private int offsetX;
    private int offsetY;

   private boolean LJoystick;

   // ========================================
// Constructors
// ========================================

   public JoystickView(Context context, boolean which) {
        super(context);
        LJoystick = which;
        initJoystickView();
    }

   public JoystickView(Context context, AttributeSet attrs, boolean which) {
        super(context, attrs);
        LJoystick = which;
        initJoystickView();
    }

   public JoystickView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        initJoystickView();
    }

   // ========================================
// Initialization
// ========================================

   private void initJoystickView() {
      setFocusable(true);

      dbgPaint1 = new Paint(Paint.ANTI_ALIAS_FLAG);
        dbgPaint1.setColor(Color.RED);
        dbgPaint1.setStrokeWidth(1);
        dbgPaint1.setStyle(Paint.Style.STROKE);

      dbgPaint2 = new Paint(Paint.ANTI_ALIAS_FLAG);
        dbgPaint2.setColor(Color.GREEN);
        dbgPaint2.setStrokeWidth(1);
        dbgPaint2.setStyle(Paint.Style.STROKE);

      bgPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        bgPaint.setColor(Color.WHITE);
        bgPaint.setStrokeWidth(1);
        bgPaint.setStyle(Paint.Style.FILL_AND_STROKE);

      handlePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        handlePaint.setColor(Color.GRAY);
        handlePaint.setStrokeWidth(1);
        handlePaint.setStyle(Paint.Style.FILL_AND_STROKE);

      innerPadding = 10;
```

```
        setMovementRange(128);
        setMoveResolution(1.0f);
        setClickThreshold(0.4f);
        setYAxisInverted(false);
        setUserCoordinateSystem(COORDINATE_CARTESIAN);
        setAutoReturnToCenter(true);
    }

    public void setAutoReturnToCenter(boolean autoReturnToCenter) {
        this.autoReturnToCenter = autoReturnToCenter;
    }

    public boolean isAutoReturnToCenter() {
        return autoReturnToCenter;
    }

    public void setUserCoordinateSystem(int userCoordinateSystem) {
        if (userCoordinateSystem < COORDINATE_CARTESIAN || movementConstraint > CO-
ORDINATE_DIFFERENTIAL)
            Log.e(TAG, "invalid value for userCoordinateSystem");
        else
this.userCoordinateSystem = userCoordinateSystem;
    }

    public int getUserCoordinateSystem() {
        return userCoordinateSystem;
    }

    public void setMovementConstraint(int movementConstraint) {
        if (movementConstraint < CONSTRAIN_BOX || movementConstraint > CONSTRAIN_CIRCLE)
            Log.e(TAG, "invalid value for movementConstraint");
        else
this.movementConstraint = movementConstraint;
    }

    public int getMovementConstraint() {
        return movementConstraint;
    }

    public boolean isYAxisInverted() {
        return yAxisInverted;
    }

    public void setYAxisInverted(boolean yAxisInverted) {
        this.yAxisInverted = yAxisInverted;
    }

    /**
* Set the pressure sensitivity for registering a click
* @param clickThreshold threshold 0...1.0f inclusive.  0 will cause clicks to never be reported, 1.0 is a
very hard click
*/
public void setClickThreshold(float clickThreshold) {
        if (clickThreshold < 0 || clickThreshold > 1.0f)
            Log.e(TAG, "clickThreshold must range from 0...1.0f inclusive");
        else
this.clickThreshold = clickThreshold;
    }

    public float getClickThreshold() {
        return clickThreshold;
    }
```

```
    public void setMovementRange(float movementRange) {
        this.movementRange = movementRange;
    }

    public float getMovementRange() {
        return movementRange;
    }

    public void setMoveResolution(float moveResolution) {
        this.moveResolution = moveResolution;
    }

    public float getMoveResolution() {
        return moveResolution;
    }

    // ========================================
// Public Methods
// ========================================

    public void setOnJostickMovedListener(JoystickMovedListener listener) {
        this.moveListener = listener;
    }

    public void setOnJostickClickedListener(JoystickClickedListener listener) {
        this.clickListener = listener;
    }

    // ========================================
// Drawing Functionality
// ========================================

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        // Here we make sure that we have a perfect circle
int measuredWidth = measure(widthMeasureSpec);
        int measuredHeight = measure(heightMeasureSpec);
        setMeasuredDimension(measuredWidth, measuredHeight);
    }

    @Override
    protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
        super.onLayout(changed, left, top, right, bottom);

      int d = Math.min(getMeasuredWidth(), getMeasuredHeight());

      dimX = d;
        dimY = d;

      cX = d / 2;
        cY = d / 2;

      bgRadius = dimX/2 - innerPadding;
        handleRadius = (int)(d * 0.25);
        handleInnerBoundaries = handleRadius;
        movementRadius = Math.min(cX, cY) - handleInnerBoundaries;
    }

    private int measure(int measureSpec) {
        int result = 0;
        // Decode the measurement specifications.
int specMode = MeasureSpec.getMode(measureSpec);
        int specSize = MeasureSpec.getSize(measureSpec);
        if (specMode == MeasureSpec.UNSPECIFIED) {
            // Return a default size of 200 if no bounds are specified.
```

```
result = 200;
        } else {
            // As you want to fill the available space
// always return the full available bounds.
result = specSize;
        }
        return result;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.save();
        // Draw the background
canvas.drawCircle(cX, cY, bgRadius, bgPaint);

      // Draw the handle
handleX = touchX + cX;
        handleY = touchY + cY;
        canvas.drawCircle(handleX, handleY, handleRadius, handlePaint);

      if (D) {
            canvas.drawRect(1, 1, getMeasuredWidth()-1, getMeasuredHeight()-1, dbgPaint1);

         canvas.drawCircle(handleX, handleY, 3, dbgPaint1);

         if ( movementConstraint == CONSTRAIN_CIRCLE ) {
               canvas.drawCircle(cX, cY, this.movementRadius, dbgPaint1);
            }
            else {
               canvas.drawRect(cX-movementRadius, cY-movementRadius, cX+movementRadius,
cY+movementRadius, dbgPaint1);
            }

         //Origin to touch point
canvas.drawLine(cX, cY, handleX, handleY, dbgPaint2);

         int baseY = (int) (touchY < 0 ? cY + handleRadius : cY - handleRadius);
            canvas.drawText(String.format("%s (%.of,%.of)", TAG, touchX, touchY), handleX-20,
baseY-7, dbgPaint2);
            canvas.drawText("("+ String.format("%.of, %.1f", radial, angle * 57.2957795) + (char)
0x00B0 + ")", handleX-20, baseY+15, dbgPaint2);
        }

//    Log.d(TAG, String.format("touch(%f,%f)", touchX, touchY));
//    Log.d(TAG, String.format("onDraw(%.1f,%.1f)\n\n", handleX, handleY));
canvas.restore();
    }

   // Constrain touch within a box
private void constrainBox() {
        touchX = Math.max(Math.min(touchX, movementRadius), -movementRadius);
        touchY = Math.max(Math.min(touchY, movementRadius), -movementRadius);
    }

   // Constrain touch within a circle
private void constrainCircle() {
        float diffX = touchX;
        float diffY = touchY;
        double radial = Math.sqrt((diffX*diffX) + (diffY*diffY));
        if ( radial > movementRadius ) {
            touchX = (int)((diffX / radial) * movementRadius);
            touchY = (int)((diffY / radial) * movementRadius);
        }
    }
```

```
    public void setPointerId(int id) {
        this.pointerId = id;
    }

    public int getPointerId() {
        return pointerId;
    }

    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        final int action = ev.getAction();
        switch (action & MotionEvent.ACTION_MASK) {
            case MotionEvent.ACTION_MOVE: {
                return processMoveEvent(ev);
            }
            case MotionEvent.ACTION_CANCEL:
            case MotionEvent.ACTION_UP: {
                if ( pointerId != INVALID_POINTER_ID ) {
//                  Log.d(TAG, "ACTION_UP");
if(LJoystick == true){
                        lReturnHandleToCenter();
                    }
                    else {
                        returnHandleToCenter();
                    }
                    setPointerId(INVALID_POINTER_ID);
                }
                break;
            }
            case MotionEvent.ACTION_POINTER_UP: {
                if ( pointerId != INVALID_POINTER_ID ) {
                    final int pointerIndex = (action & MotionEvent.ACTION_POINTER_INDEX_MASK)
>> MotionEvent.ACTION_POINTER_INDEX_SHIFT;
                    final int pointerId = ev.getPointerId(pointerIndex);
                    if ( pointerId == this.pointerId ) {
//                      Log.d(TAG, "ACTION_POINTER_UP: " + pointerId);
if(LJoystick == true){
                            lReturnHandleToCenter();
                        }
                        else {
                            returnHandleToCenter();
                        }
                        setPointerId(INVALID_POINTER_ID);
                        return true;
                    }
                }
                break;
            }
            case MotionEvent.ACTION_DOWN: {
              if ( pointerId == INVALID_POINTER_ID ) {
                    int x = (int) ev.getX();
                    if ( x >= offsetX && x < offsetX + dimX ) {
                        setPointerId(ev.getPointerId(0));
//                      Log.d(TAG, "ACTION_DOWN: " + getPointerId());
return true;
                    }
                }
                break;
            }
```

```
            case MotionEvent.ACTION_POINTER_DOWN: {
                if ( pointerId == INVALID_POINTER_ID ) {
                final int pointerIndex = (action & MotionEvent.ACTION_POINTER_INDEX_MASK)
>> MotionEvent.ACTION_POINTER_INDEX_SHIFT;
                    final int pointerId = ev.getPointerId(pointerIndex);
                    int x = (int) ev.getX(pointerId);
                    if ( x >= offsetX && x < offsetX + dimX ) {
//                  Log.d(TAG, "ACTION_POINTER_DOWN: " + pointerId);
setPointerId(pointerId);
                        return true;
                    }
                }
                break;
            }
        }
        return false;
    }

    private boolean processMoveEvent(MotionEvent ev) {
        if ( pointerId != INVALID_POINTER_ID ) {
            final int pointerIndex = ev.findPointerIndex(pointerId);

          // Translate touch position to center of view
float x = ev.getX(pointerIndex);
            touchX = x - cX - offsetX;
            float y = ev.getY(pointerIndex);
            touchY = y - cY - offsetY;

//          Log.d(TAG, String.format("ACTION_MOVE: (%03.0f, %03.0f) => (%03.0f, %03.0f)", x, y,
touchX, touchY));

          reportOnMoved();
            invalidate();

          touchPressure = ev.getPressure(pointerIndex);
            reportOnPressure();

          return true;
        }
        return false;
    }

    private void reportOnMoved() {
        if ( movementConstraint == CONSTRAIN_CIRCLE )
            constrainCircle();
        else
constrainBox();

      calcUserCoordinates();

      if (moveListener != null) {
            boolean rx = Math.abs(touchX - reportX) >= moveResolution;
            boolean ry = Math.abs(touchY - reportY) >= moveResolution;
            if (rx || ry) {
                this.reportX = touchX;
                this.reportY = touchY;

//        Log.d(TAG, String.format("moveListener.OnMoved(%d,%d)", (int)userX, (int)userY));
moveListener.OnMoved(userX, userY);
            }
        }
    }
```

```
   private void calcUserCoordinates() {
       //First convert to cartesian coordinates
cartX = (int)(touchX / movementRadius * movementRange);
       cartY = (int)(touchY / movementRadius * movementRange);

      radial = Math.sqrt((cartX*cartX) + (cartY*cartY));
       angle = Math.atan2(cartY, cartX);

      //Invert Y axis if requested
if ( !yAxisInverted )
           cartY  *= -1;

      cartX *= -1;

      if ( userCoordinateSystem == COORDINATE_CARTESIAN ) {
           userX = cartX;
           userY = cartY;
        }
        else if ( userCoordinateSystem == COORDINATE_DIFFERENTIAL ) {
           userX = cartY + cartX / 4;
           userY = cartY - cartX / 4;

         if ( userX < -movementRange )
              userX = (int)-movementRange;
           if ( userX > movementRange )
              userX = (int)movementRange;

         if ( userY < -movementRange )
              userY = (int)-movementRange;
           if ( userY > movementRange )
              userY = (int)movementRange;
        }

   }

   //Simple pressure click
private void reportOnPressure() {
//    Log.d(TAG, String.format("touchPressure=%.2f", this.touchPressure));
if ( clickListener != null ) {
           if ( clicked && touchPressure < clickThreshold ) {
              clickListener.OnReleased();
              this.clicked = false;
//         Log.d(TAG, "reset click");
invalidate();
           }
           else if ( !clicked && touchPressure >= clickThreshold ) {
              clicked = true;
              clickListener.OnClicked();
//         Log.d(TAG, "click");
invalidate();
              performHapticFeedback(HapticFeedbackConstants.VIRTUAL_KEY);
           }
        }
    }

   private void returnHandleToCenter() {
        if ( autoReturnToCenter ) {
           final int numberOfFrames = 5;
           final double intervalsX = (0 - touchX) / numberOfFrames;
           final double intervalsY = (0 - touchY) / numberOfFrames;

         for (int i = 0; i < numberOfFrames; i++) {
              final int j = i;
```

```
                postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        touchX += intervalsX;
                        touchY += intervalsY;

                    reportOnMoved();
                        invalidate();

                    if (moveListener != null && j == numberOfFrames - 1) {
                            moveListener.OnReturnedToCenter();
                        }
                    }
                }, i * 40);
            }

        if (moveListener != null) {
            moveListener.OnReleased();
        }
    }
}

private void lReturnHandleToCenter() {
    if ( autoReturnToCenter ) {
        final int numberOfFrames = 5;
        final double intervalsX = (0 - touchX) / numberOfFrames;
        final double intervalsY = (0 );

        for (int i = 0; i < numberOfFrames; i++) {
            final int j = i;
            postDelayed(new Runnable() {
                @Override
                public void run() {
                    touchX += intervalsX;
                    touchY = movementRadius + movementRadius + movementRadius;
                    //131;

                reportOnMoved();
                    invalidate();

                if (moveListener != null && j == numberOfFrames - 1) {
                        moveListener.OnReturnedToCenter();
                    }
                }
            }, i * 40);
        }

        if (moveListener != null) {
            moveListener.OnReleased();
        }
    }
}

public void setTouchOffset(int x, int y) {
    offsetX = x;
    offsetY = y;
}
}
```

4.2.12 JoystickMovedListener.java

```
package com.example.softwarecontrolleddrone;
```

```
public interface JoystickMovedListener {
    public void OnMoved(int pan, int tilt);
    public void OnReleased();
    public void OnReturnedToCenter();
}
```

4.2.13 JoystickClickedListener.java

```
package com.example.softwarecontrolleddrone;

public interface JoystickClickedListener {
    public void OnClicked();
    public void OnReleased();
}
```

4.2.14 DualJoystickView.java

```
package com.example.softwarecontrolleddrone;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;

public class DualJoystickView extends LinearLayout{
    @SuppressWarnings("unused")
    private static final String TAG = DualJoystickView.class.getSimpleName();

   private final boolean D = false;
    private Paint dbgPaint1;

   private JoystickView stickL;
    private JoystickView stickR;

   private View pad;

   public DualJoystickView(Context context) {
        super(context);
        stickL = new JoystickView(context, true);
        stickR = new JoystickView(context, false);
        initDualJoystickView();
    }

   public DualJoystickView(Context context, AttributeSet attrs) {
        super(context, attrs);
        stickL = new JoystickView(context, attrs, true);
        stickR = new JoystickView(context, attrs, false);
        initDualJoystickView();
    }

   private void initDualJoystickView() {
        setOrientation(LinearLayout.HORIZONTAL);

      if ( D ) {
            dbgPaint1 = new Paint(Paint.ANTI_ALIAS_FLAG);
            dbgPaint1.setColor(Color.CYAN);
            dbgPaint1.setStrokeWidth(1);
```

```
            dbgPaint1.setStyle(Paint.Style.STROKE);
        }
        pad = new View(getContext());
    }
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        removeView(stickL);
        removeView(stickR);
        float padW = getMeasuredWidth()-(getMeasuredHeight()*2);
        int joyWidth = (int) ((getMeasuredWidth()-padW)/2);
        LayoutParams joyLParams = new LayoutParams(joyWidth,getMeasuredHeight());
        stickL.setLayoutParams(joyLParams);
        stickR.setLayoutParams(joyLParams);
        stickL.TAG = "L";
        stickR.TAG = "R";
        stickL.setPointerId(JoystickView.INVALID_POINTER_ID);
        stickR.setPointerId(JoystickView.INVALID_POINTER_ID);
        addView(stickL);
    ViewGroup.LayoutParams padLParams = new ViewGroup.LayoutParams((int) padW,getMeasuredHeight());
        removeView(pad);
        pad.setLayoutParams(padLParams);
        addView(pad);
        addView(stickR);
    }
    @Override
    protected void onLayout(boolean changed, int l, int t, int r, int b) {
        super.onLayout(changed, l, t, r, b);
        stickR.setTouchOffset(stickR.getLeft(), stickR.getTop());
    }
    public void setAutoReturnToCenter(boolean left, boolean right) {
        stickL.setAutoReturnToCenter(left);
        stickR.setAutoReturnToCenter(right);
    }
    public void setOnJostickMovedListener(JoystickMovedListener left, JoystickMovedListener right)
{
        stickL.setOnJostickMovedListener(left);
        stickR.setOnJostickMovedListener(right);
    }
    public void setOnJostickClickedListener(JoystickClickedListener left, JoystickClickedListener right)
{
        stickL.setOnJostickClickedListener(left);
        stickR.setOnJostickClickedListener(right);
    }
    public void setYAxisInverted(boolean leftYAxisInverted, boolean rightYAxisInverted) {
        stickL.setYAxisInverted(leftYAxisInverted);
        stickL.setYAxisInverted(rightYAxisInverted);
    }
    public void setMovementConstraint(int movementConstraint) {
        stickL.setMovementConstraint(movementConstraint);
        stickR.setMovementConstraint(movementConstraint);
    }
```

```
    public void setMovementRange(float movementRangeLeft, float movementRangeRight) {
        stickL.setMovementRange(movementRangeLeft);
        stickR.setMovementRange(movementRangeRight);
    }
    public void setMoveResolution(float leftMoveResolution, float rightMoveResolution) {
        stickL.setMoveResolution(leftMoveResolution);
        stickR.setMoveResolution(rightMoveResolution);
    }
    public void setUserCoordinateSystem(int leftCoordinateSystem, int rightCoordinateSystem) {
        stickL.setUserCoordinateSystem(leftCoordinateSystem);
        stickR.setUserCoordinateSystem(rightCoordinateSystem);
    }
    @Override
    protected void dispatchDraw(Canvas canvas) {
        super.dispatchDraw(canvas);
        if (D) {
            canvas.drawRect(1, 1, getMeasuredWidth()-1, getMeasuredHeight()-1, dbgPaint1);
        }
    }
    @Override
    public boolean dispatchTouchEvent(MotionEvent ev) {
        boolean l = stickL.dispatchTouchEvent(ev);
        boolean r = stickR.dispatchTouchEvent(ev);
        return l || r;
    }
    @Override
    public boolean onTouchEvent(MotionEvent ev) {
        boolean l = stickL.onTouchEvent(ev);
        boolean r = stickR.onTouchEvent(ev);
        return l || r;
    }
}
```

4.3 PHP Files

4.3.1 AddMember.php

```
<?php

\$host = "mysql.hostinger.co.uk";

\$user = "u940148205\_root";

\$pass = "Drone123";

\$db = "u940148205\_drone";



\$con = mysqli\_connect(\$host, \$user, \$pass, \$db);



if(!\$con)
```

```
{
    die("Error in connection " . mysqli\_connect\_error());
}

\$firstName = \$\_POST["firstName"];
\$lastName = \$\_POST["lastName"];
\$username = \$\_POST["username"];
\$password = \$\_POST["password"];


 \$checkUser = "SELECT username FROM DroneMembers WHERE username
='$username';";
 \$result = mysqli\_query(\$con, \$checkUser);

 if(mysqli\_num\_rows(\$result) \>= 1)
 {
     echo "User already exists";
 }
 else
 {
    \$sql = "INSERT INTO DroneMembers VALUES('\$firstName', '\$lastName',
'$username', '$password');";
    if(mysqli\_query(\$con, \$sql))
    {
        echo "Registration Success";
    }
    else
    {
        echo "Error in insertion" . mysqli\_error(\$con);
```

```
    }
}
?>
```

4.3.2 DeleteFlightInfo.php

```
<?php
        /*Establishing Connection*/
        $host = ”mysql.hostinger.co.uk”;
        $user = ”u940148205_root”;
        $pass = ”Drone123”;
        $db = ”u940148205_drone”;
        $con = mysqli_connect($host, $user, $pass, $db);
        if(!$con)
        {
                die(”Error in connection ” . mysqli_connect_error());
        }
        $username = $_POST[”username”];
        $sql = ”DELETE FROM DroneInfo WHERE username = ’$username’;”;
        if(mysqli_query($con, $sql))
        {
                echo ”Information Deleted Successfully”;
        }
        else
        {
                echo ”Error Deleting Information”;
        }
        mysqli_close($con);
?>
```

4.3.3 FlightInfo2.php

```
<?php
        /*Establishing a connection*/
        $host = ”mysql.hostinger.co.uk”;
        $user = ”u940148205_root”;
        $pass = ”Drone123”;
        $db = ”u940148205_drone”;
        $con = mysqli_connect($host, $user, $pass, $db);
```

```
	if(!$con)
	{
		die("Error in connection " . mysqli_connect_error());
	}
	$date = $_POST["date"];
	$flightduration = $_POST["flightduration"];
	$username = $_POST["username"];
	//$myCheck = "SELECT username FROM DroneInfo WHERE username LIKE '$username';";
	//$result = mysqli_query($con, $myCheck);
	//if(mysqli_num_rows($result) % 2 == 0)
	//{
		$sql = mysqli_query($con, "INSERT INTO DroneInfo (date, flightduration, username) VALUES ('$date', '$flightduration', '$username')");
		if($con->query($sql) == TRUE)
		{
			echo "Flight info added";
		}
		/*
		  $sql = "INSERT INTO DroneInfo (date, flightduration, username) VALUES('$date', '$flightduration', '$username');";
		if(mysqli_query($con, $sql))
		{
			echo "Flight Info Added";
		}
		*/
	//}
	else
	{
		echo "Error in insertion " . mysqli_error($con);
	}
	mysqli_close($con);
?>
```

4.3.4 Login.php

```
<?php
	/*Establishing connection*/
	$host = "mysql.hostinger.co.uk";
	$user = "u940148205_root";
	$pass = "Drone123";
	$db = "u940148205_drone";
```

```
	$con = mysqli_connect($host, $user, $pass, $db);
	if(!$con)
	{
		die(”Error in connection ” . mysqli_connect_error());
	}
	$username = $_POST[”username”];
	$password = $_POST[”password”];
	$sql = ”SELECT username FROM DroneMembers WHERE username LIKE ’$username’ AND password LIKE ’$password’;”;
	$result = mysqli_query($con, $sql);
	if(mysqli_num_rows($result) > 0)
	{
		$row = mysqli_fetch_assoc($result);
		$name = $row[”username”];
		echo ”Login Successful”;
	}
	else
	{
		echo ”User Does Not Exist”;
	}
?>
```

5. Bibliography

Barták, R., Hraško, A., & Obdržálek, D. (2014). A controller for autonomous landing of aR.Drone. In *The 26th chinese control and decision conference (2014 cCDC)* (pp. 329–334). https://doi.org/10.1109/CCDC.2014.6852167

Gomes, L. L., Leal, L. P., Oliveira, T. R., Cunha, J. P. V. S. da, & Revoredo, T. C. (2016). Unmanned quadcopter control using a motion capture system. *IEEE Latin America Transactions*, *14*(8), 3606–3613. https://doi.org/10.1109/TLA.2016.7786340

Malone, P., Apgar, H., Stukes, S., & Sterk, S. (2013). Unmanned aerial vehicles unique cost estimating requirements. In *2013 iEEE aerospace conference* (pp. 1–8). https://doi.org/10.1109/AERO.2013.6496852