Final Project – CIS 330
Team: Segmentation Fault

Project Edison Drone

Last Update: 03/12/2016
Instructor: Professor Bouana Norris

Members: Matt Almenshad, David Zimmerly, Jacob Lin, Joseph McLaughlin

---

Project Overview

This project is an Intel Edison Operated Quadcopter.
The project consists of the following hardware:
- A light-weight frame that is sturdy enough to carry all the other parts.
- Four brushless motors, four propellers and four Electronic Speed Controllers (ESCs) to generate the controlled thrust needed to elevate the drone.
- Lithium Polymer Battery with enough capacity to power all the circuits and motors.
- Intel Edison with W/ Arduino Kit to operate the project.
- A (Bluetooth/Radio/Wi-Fi)-based method to remotely control the drone.
- A Gyroscope/Accelerometer stance tilt module to balance the drone.

All the software is written in C/C++/Arduino.
External Libraries used:
- *Arduino PID Library.*
- *Ethernet Library.*
- *Adafruit PWM Servo Driver Library.*
- *I2C Device Library.*
- *Wire Library.*
- *MPU6050 Library.*

Hardware Selection

Frame:

Our goal for the frame get the most strength possible for the least amount of weight.
We decided to go with *Jrelecs 4-Axis HJ450* Due to its durability which is delivered at a mere 260g.

Motor, Propellers and propeller adapters:

The motor selection is based on primarily the amount of thrust it can generate. The thrust generated by the motor must be 2 times as much as the weight pulling downwards in order for the drone to hover. To allow for adjustment and marginal error we increased this factor to 3 times the amount of target weight.
We selected *HobbyPower A2212 1000kv Brushless Motor* based on its ability to generate at thrust of 800g with a 10x5" propeller. Since we will be using four motors, this adds up to a thrust of 3200g, allowing for over 1000g of weight to be distributed amongst the other parts according to our module.
As for the propellers, we managed to get our hands on *Jrelecs® 1045 Propellers*. At 10x45" they are 0.5" shorter than planned, although they were still sufficient to handle our 1000g weight limit.
As for the propeller adapters, the original adapters that came with our motors were cheaply made and had a tendency to break. Combined with the tremendous amount of reviews stating that they are almost impossible to balance, we decided to pick *CO-RODE 3.17mm Aluminum Propeller Adapter* as an alternative.
The combined weight of the motors was 188g (4x47g). As for the propellers, the weight summed up to 56g (4x14g). Finally, the propeller adapters weighted at 16g (4x4g).

Electronic Speed Controllers (ESCs):

The Selection process for the controllers entailed finding a compatible controller with our motor, we found out that the *HobbyPower 30A Brushless ESC* was very compatible. Added to its very convent 5v BEC future it was perfect. Total ESC weight amounted to 100g (4x25g).

Intel Edison W/ Arduino Kit

Intel Edison has WiFi, Bluetooth, and is able to generate PWM signals and it runs a UNIX system. It was the perfect match for our project at approximately 53g of weight.

LiPo Battery

The 12v *Turnigy 2200mAh 3S 25C LiPo Battery* is able to generate 55 Amps of current at safely. Since all our motors can take up a maximum of 48A, this leaves us 7A to power up the Edison board and the Gyroscope/Accelerometer circuits which is more than sufficient.

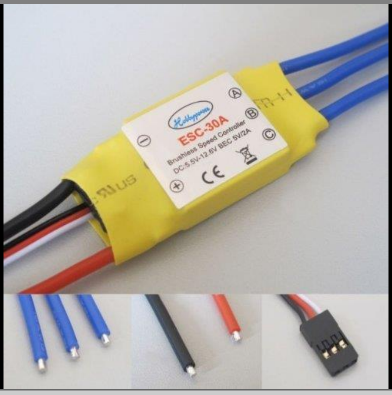
Gyroscope/Accelerometer

The *GY-521 6DOF MPU6050 Module* was our go-to choice due to its popularity and cheap price. It is compatible with the Arduino and only weighs at 5g

Hardware Gallery

Jrelecs 4-Axis HJ450 frame

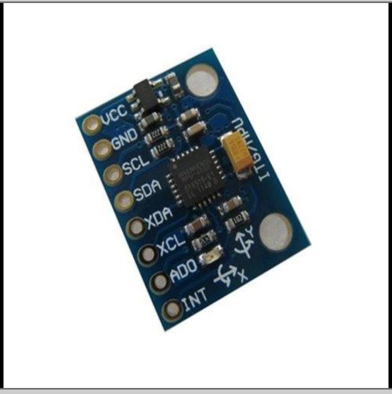
HOBBYPOWER A2212 1000kv Brushless Motor


HOBBYPOWER 30A Brushless ESC

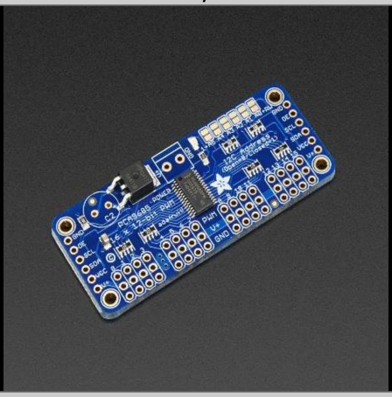
Jrelecs® 1045 Propellers

Turnigy 2200mAh 3S 25C LiPo Battery

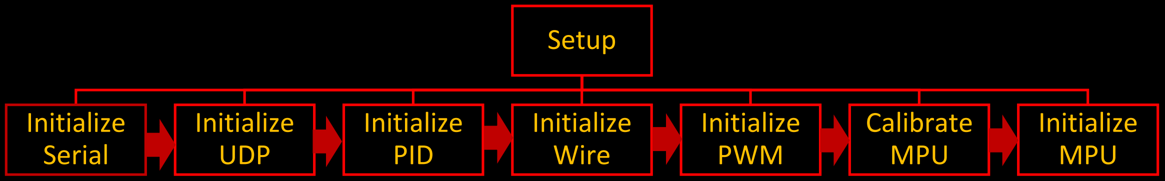
Intel Edison W/ Arduino Kit

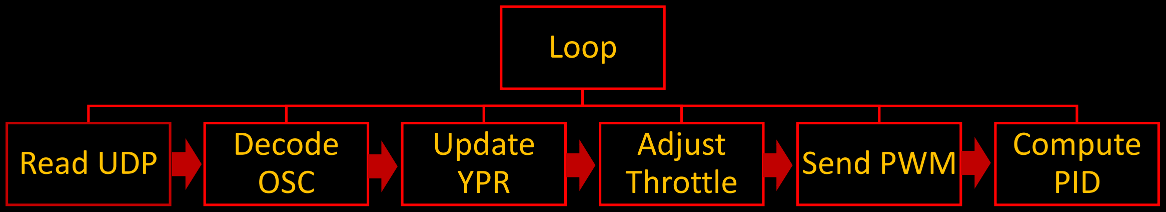

GY-521 6DOF MPU6050 Module

CO-RODE 3.17mm Aluminum Propeller Adapter


ADAFRUIT 16-CHANNEL 12-BIT PWM DRIVER

Arduino Sketch





Operation

We came up with a concept of turning the Edison Arduino into a Quadcopter control system. Originally, the idea was to use Edison's Bluetooth and a servo shield to spin the motors creating enough torque to pull the quadrouter to the air. The objective was to use our phone's Bluetooth to remotely send the commands

Remote Control

We learned that the SPI interface for Bluetooth control is not natively available for IOS devices (not without jailbreak). Our goal of using the phones was to make it a multiplatform solution. So we migrated to a multiplatform solution called Open Source Control. Open source control is a protocol of transmitting data over WiFi in an efficient manner suitable for control uses.
We found multiplatform tool called TouchOSC that allows for a touch-controlled layout to be designed and ported on TouchOSC's mobile app.

Our first challenge was to transmit the command to the Edison over WiFi. We decided to use the WiFiUDP library to connect to a third party modem. The connection needed to be initiated within the sketch's setup. After successfully integrating that we moved on to read the messages of the OSC. We learned that the library to decode OSC messages on Arduino (OSCuino) had issues of all sorts when it comes to the compatibility with Edison, during our attempt to debug the library, we realizing that the OSC protocol transmits messages with the following format:
/directory/subdirectory/(as many directories as you need) type, value

So a 4.44 float to be directed to function Foo that sends would need to be associated with a directory /bar and then written in the following format

/bar f, 4.44

then on the receiver side would need to be linked with the decoder in such manner:
MSG.route("/bar", foo)

The size of the package would vary depending on the length of the directory.

so instead, we decided to use a standard format for our commands, using the directory limited to one letter or number. The type sent by TouchOSC is always a float, so our format would be:
/f f, 205.00

to send a frequency of 5% duty cycle, setting the throttle at minimum (off) speed.

so all we had to do was read the package in location [1] to get the command's first letter and map its value to the proper function.

Then we faced a problem where the floats we received were in gibberish, they made no sense. The package contained random values suck as (/D塥⍰) at the location where the float was supposed to be.

Our research lead us to believe that our phones handle floats in Little Endian, while Edison reads the float in Big Endian. So all we had to do was read the last four bytes of the package in reverse, store them in a float structure then got our proper 205.00 value.

Following the perfection of our OSC package reception, we needed to find a solution for the logistic boundary that our choice of connection have created. Our WiFi depends meant that we would need to be in range of a WiFi Router. At first, we considered using a mobile router to overcome this problem. Then we realized that instead, we can have the Edison run in an Access Point (AP) mode, allowing us to connect to it directly with our phones and send data with no third party dependencies.

Internal Control

The most essential requirement to operate a Quadcopter is the ability to adjust the speed of the motors. The most watered down description of a brushless motor would be a barrel with three magnets on its wall, arranged in a triangular manner. Powering the corners of said triangle, would cause the two adjacent motors to power up pulling the pole towards their direction. Powering the next corners in a sequence would cause pole to spin. Reversing the sequence would cause the pole to spin in the opposite direction.

To generate this sequence we needed a device called “Electronic Speed Controller” or ESC for short. The way that the ESC work is by taking power from a battery and processing the sequence to the brushless motor. ESC is controlled through a signal wire that reads a signal at a rate of 50Hz (20ms). The duty cycle of the ESC is what determines the speed of the ESC-to-motor sequence speed. Upon being calibrated to do so, the ESC reads a duty cycle of 5% as 0% speed, and a duty cycle of 10% as 100% speed.

Our original intention was to use the Edison Arduino PWM generators to emit the signals to the ESC. However, the resolution Edison Arduino is only 8 bits, allowing 256 (2^8) adjustments over a period of 20ms. With a 5% variable range we only have about 13 adjustments (5% of 256) in speed. Which is not smooth enough to control adjust the drone’s movement as changing the speed in increments of ~8% throttle is too fast and prevents us from correcting the alignment properly.

In order to solve that problem, we obtained an external PWM generator. The *Adafruit PCA9685* provides 16 channels of 12 bit resolution at an adjustable frequency ranging from 24Hz up to 1.6KHz. Its chainable I2C interface allowed us to connect the MPU through rather than having to splice the wires. It was the perfect fit.

At this point we had the ability to adjust the throttle and read the MPU data. The next step for us was to calculate the needed adjustments from the MPU data.

We use the DMP algorithm from MPU6050 library to calculate the Yaw, Pitch and Roll off the Quadcopter using the MPU data. The Yaw is the rotation angle of the quad copter. The Pitch the front and back tilt of the Quadcopter, the Pitch is the left and right tilt of the Quadcopter.

We used the PID (Proportional Integral Derivative) algorithm to create adjustments based on the Yaw, Pitch and Roll to modify our throttle values. The PID algorithm uses a total of 6 constants to calculate for error in adjustment for fine tuning the outputs and make it stable in the air.

Our final adjustment was applied such as that:

Front Left Value = Throttle – Pitch based adjustment + Roll based adjustment - Yaw based adjustment

Front Right Value = Throttle + Pitch based adjustment + Roll based adjustment + Yaw based adjustment

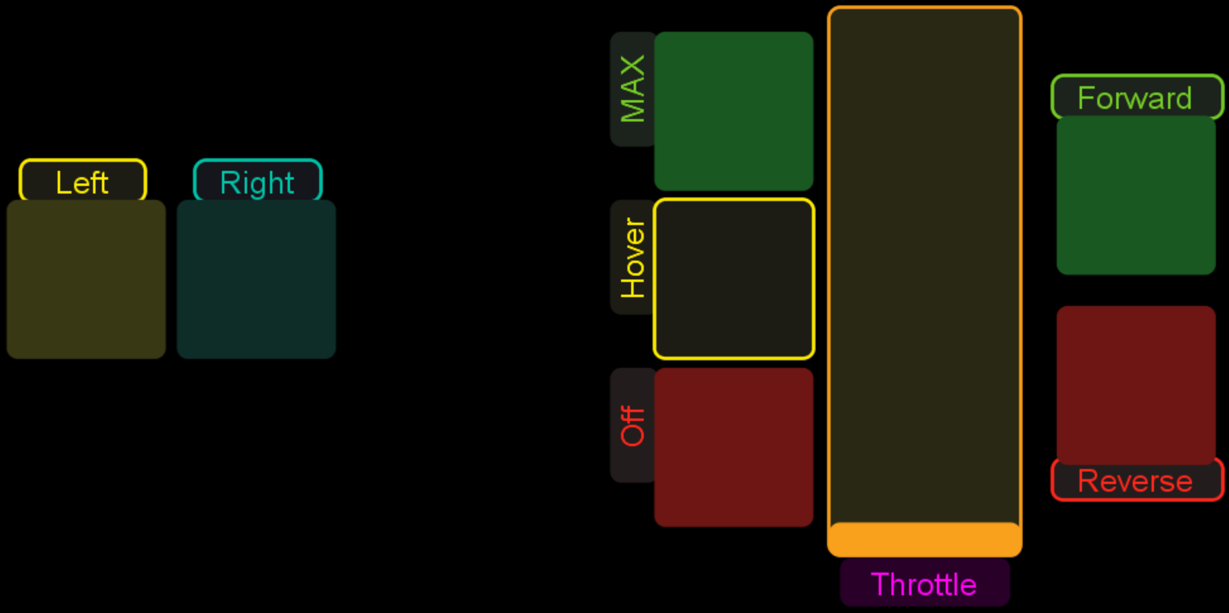
Back Left Value = Throttle + Pitch based adjustment - Roll based adjustment - Yaw based adjustment

Back Right Value = Throttle – Pitch based adjustment - Roll based adjustment + Yaw based adjustment

We then sent the values to their respective ESCs using PWM signals.

As of the time of the presentation we were still in the process of tuning the constants and values, as the only way to tune them is through trial and error. So we could not keep it in the air for an extended period of time safely, however, tuning those values ensure that the drone is not only able to stay in the air but also move around freely.

<u>TouchOSC Layout</u>



We intended to have a simple layout. As the Edison will do all the balancing itself, the use needs to only worry about the throttle and the kind of movement they want to use. The Throttle slider control the 70% of the Quadcopter's total throttle, we allowed 30% power for the PID adjustments.

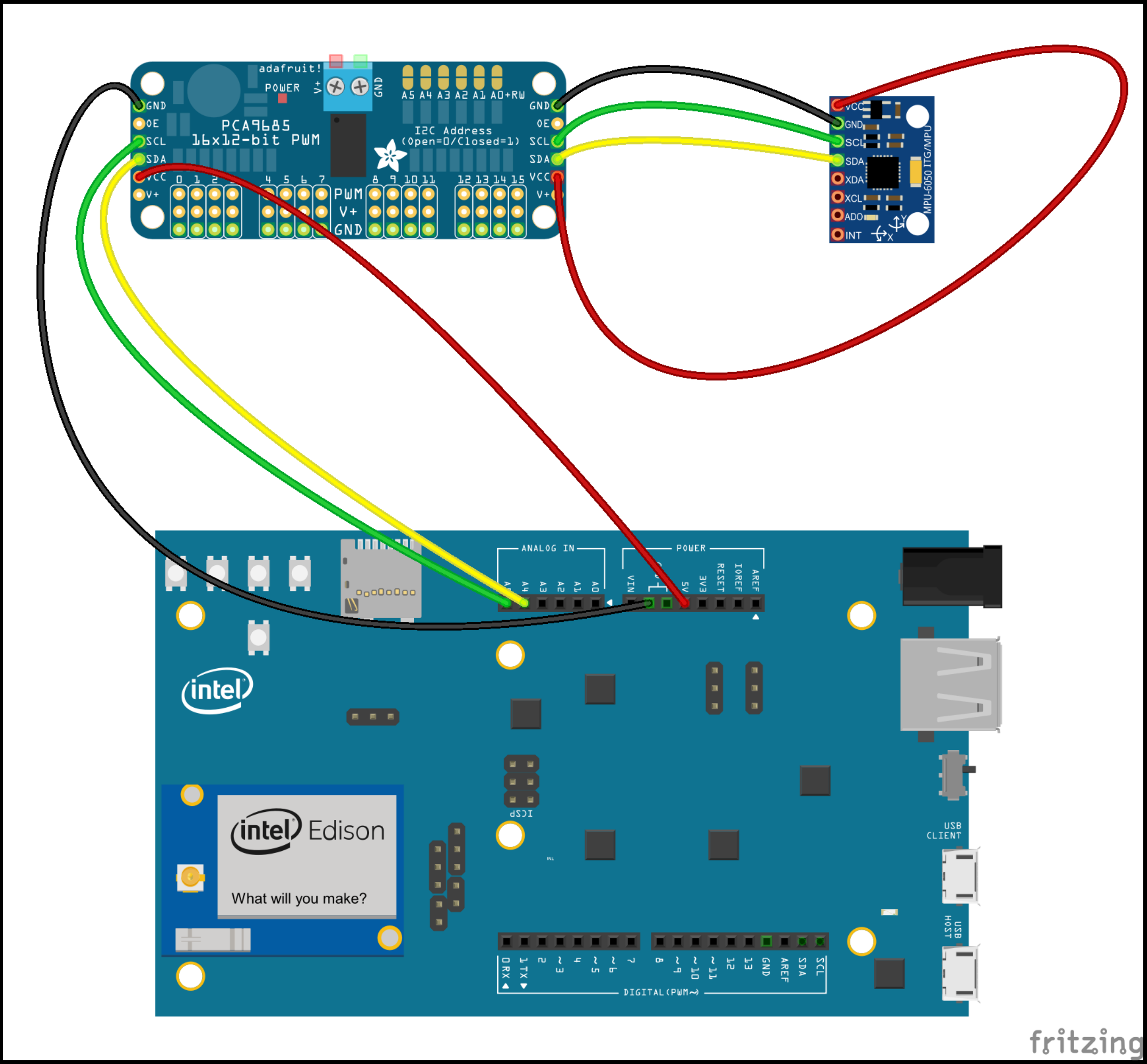
The Max, Off and Hover buttons will set the respective speeds immediately.

Pressing left or right will trigger a roll movement, forward and reverse will trigger a pitch movement, combining forward for example and left will trigger a yaw movement.

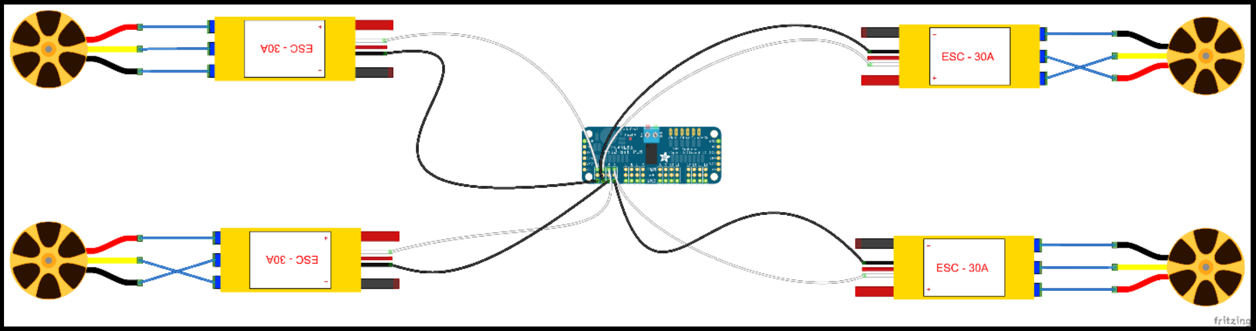
Wiring

We went through multiple configurations that required splice and soldering a few pins and wires to land on the following:

MPU to PCA to Arduino

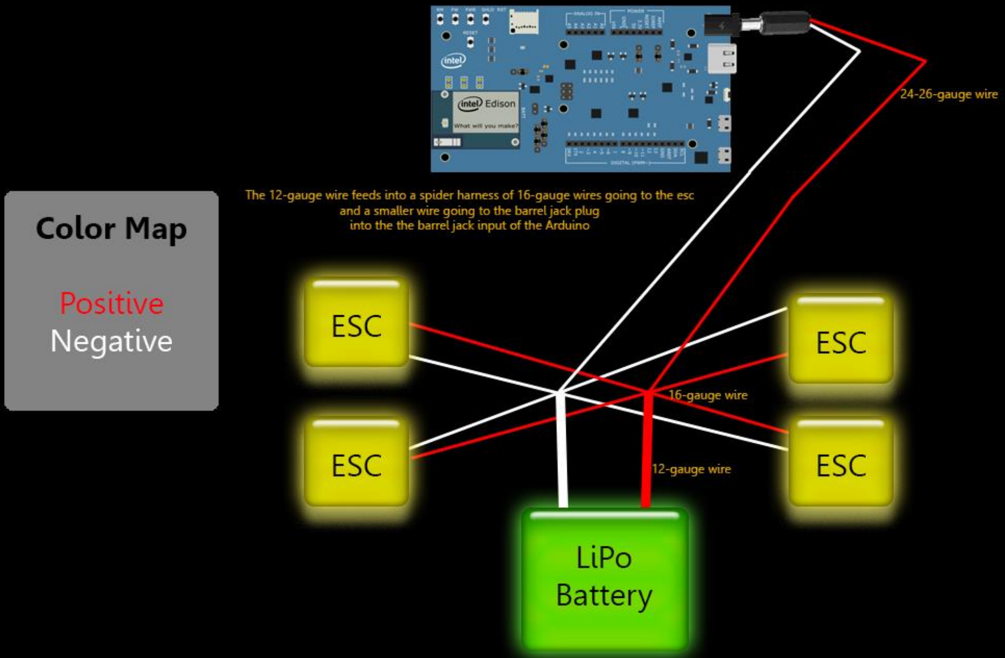


Motors to ESCs to PCA



Note: the front left and back right motor need to spin clockwise, the front right and back left motor need to spin counter-clockwise. Hence wire the wires are reversed.

Battery to Arduino and ESC Connection Sketch


Color Map
Positive
Negative
24-26-gauge wire
The 12-gauge wire feeds into a spider harness of 16-gauge wires going to the esc and a smaller wire going to the barrel jack plug into the the barrel jack input of the Arduino
ESC
ESC
ESC
ESC
16-gauge wire
12-gauge wire
LiPo Battery
Edison

```
Project: Edison Drone
Team: Segmentation Fault
Members: Matt Almenshad, David Zimmerly, Jacob Lin, Joseph McLaughlin
Class: CIS 330
Instructor: Professor Boyana Norris
```

Timeline:

Week 5:

- **Defined our plan and selected our parts.**

Week 6:

- **Acquired all the parts.**
- **Assimilated the drone.**
- **Soldered the wire connections from the ESCs to the motors.**
- **Soldered the wire connections from the battery to all the ESCs and the Arduino board.**
- **Calibrated the ESCs.**
- **Completed the software to control all motors through Serial connection (Video, Video2).**
- **Failed at attempting to auto-run the Arduino sketch without crashing.**
- **Balanced three out of four motors (Video).**

Week 7:

- **balance the final motor**
- **managed to auto-run Edison's sketch on boot.**
- **We also spent a few hours learning about UDP packets, Edison's WiFi, TouchOSC, OSCuino.**
- **managed to get our phones to communicate with the Arduino through TouchOSC, OSCuino and WiFi.**
- **debugged the OSCuino library.**
- **get the size of the packets sent from the phone to the Arduino.**
- **encountered a problem while attempting to parse the packet into the actual interface data.**
- **decided to push back the prop balancing as the ability to operate the drone with our phones is more critical.**

Week 8:

- **Wrote our own code to handle the OSC messages, got rid of OSCuino.**
- **Completed the soldering needed to power Edison through a LiPo Battery along with the motors simultaneously.**
- **Encountered a problem where Edison's automated Sketch auto-run crashes after 5 minutes.**
- **Integrated our code to control the motors with the phone through OSC messages.**
- **Modified the code to be compatible with Edison in AP mode.**
- **Elevated the drone remotely (Video).**

Week 9:

- Fixed the Auto-run crash by downgrading from version 159.dev to version 146 of Yockto.

- Fully integrated the Motion Processing Unit

- Encounter a problem with Edison's PWM resolution not being precise enough.

- Researched extensively to solve the problem.

- Discovered that the problem is due to hardware limitations.

Week 10:

- Resolved the issue of resolution by acquiring an external PWM generator (Adafruit's PCA9685).

- Integrated the PID algorithm.

- Spend a tremendous amount of time tuning the PID constants and adjustments.

- Managed to integrate PID instances for Pitch and Roll that allowed us to briefly fly the Quadcopter.

- Presented the project at Colloquium Room, DES 220 (video).