Appendix A Requirements and Verification

Table 4: System Requirements and Verifications

Requirement	Verification	Point Allocation
1. Power Supply (a) Capable of handling 286A burst output* (b) Supply 12.6V ± 0.25V when fully charged	1. Power Supply (a) Measure voltage difference across power source using a Digital multimeter placed in parallel with the power supply. voltage should read 12.6V ± 0.25V (b) Measure current difference across power source and maximum load using a Digital multimeter placed in series with the power source and load that would draw 143A. The current should read 143A ± 0.25A while maintaining the first tests temperature goals.	1. Power Supply (a) _____/1 (b) _____/1 Total Points:_____/2
2. Transceiver (a) Capable of transmitting clean signal to receiver within 10 feet. Values $\pm 100 \mu S$ (b) Capable of actuating motors	2. Transceiver (a) Turn on transmitter and check the RX values on the serial monitor. (b) Turn on transmitter and spin the motor using throttle's stick	1. Transceiver (a) _____/2 (b) _____/2 Total Points:_____/4
3. Microcontroller (a) Capable of receiving 6 signals from the transmitter using a receiver (b) Capable of outputting PWM signals to the ESC	3. Microcontroller (a) Display on Serial monitor that 6 channels are being read (b) Hook up PWM outputs to oscilloscope and test if output signal looks right, and then connect this signal to an ESC/motor to check that it correctly modulates the motors speed.	1. Microcontroller (a) _____/2 (b) _____/3 Total Points:_____/5
		Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Point Allocation
4. IMU module (accelerometer + gyro) (a) Capable of providing accurate angle measures in degrees with an accuracy of ± 10° (b) Capable of responding fast to changes of drone's position. Reading values at 100kHz ± 5kHz	4. IMU module (accelerometer + gyro) (a) The test for angles consists of tilting one of the sensor's axis (e.g. roll) while maintaining the other axis fixed (e.g. pitch). Tilt 45 ° using a triangle to check the angle. Check the measurement obtained by the sensor on Arduino's Serial Monitor. (b) Defining in the code a I2C clock rate of 100kHz. Additionaly, doing a simulation using the program called "Procesing" where is posible to simulate how well responds the drone to changes in angle.	1. IMU module (a) _____/4 (b) _____/3 Total Points:_____/7
5. Actuator (a) It must be able to be controlled by the transmitter and adjust an angle of 45°±5°	5. Actuator (a) Give the servo the signal for its 45° position, measure with precision angle measurement tool, then feed the servo the signal for its -45° position. Measure position again with the same tool	1. Actuator (a) _____/2 Total Points:_____/2
6. Centrifugal Fans (a) Must generate a thrust of 455g (1.4 thrust/weight) ± 10g in conjunction with its motor running at 100% power. (b) Must weigh (housing + impeller)less than 100g ± 5g (c) Each of the centrifugal fans must weigh within ± 10 grams of each other	6. Centrifugal Fan (a) Attach motors to centrifugal compressors and determine thrust by measuring force (in Newtons) of each individual compressor-motor pair on a stationary force sensor pad with the compressor thrusting itself against the pad. (b) Weigh module on a scale to determine mass meets requirements (c) Compare mass measurements between each module	1. Centrifugal Fan (a) _____/7 (b) _____/2 (c) _____/1 Total Points:_____/4

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Point Allocation
7. Flight Controller (a) Maintaining 0 ±5 setpoint (roll, pitch leveled) and changing sensor values, PID outputs must change (b) Control must update values at 100Hz ± 20Hz (sensor speed) (c) A good tunning of PID parameters must maintain the drone leveled parallel to the floor at 0°± 10°	7. Electronic Speed Controller (a) Check in the Serial Monitor that PID outputs changes depending on the sensor readings (b) Check in the Serial Monitor that PID outputs are updating at the same speed that sensors do (c) Make the drone fly leveled after pitching or rolling	1. Flight Controller (a) _____/2 (b) _____/2 (c) _____/4 Total Points:_____/8
8. Motors (a) Capable of providing 455g±10g of overall lift in junction with centrifugal fan	8. Motors (a) Attach motors to centrifugal compressors and determine thrust by measuring force (in grams) of each individual compressor-motor pair on a stationary force sensor pad with the compressor thrusting itself against the pad.	1. Motors (a) _____/5 Total Points:_____/5
9. Chassis Frame (a) Capable of withstanding a fall of 1m ± 0.1m (b) Must weigh in overall (including electronics and fans) less that 1.4kg ±0.1kg	9. Chassis Frame (a) Drop drone from 1.1m and repeat at 0.9m (b) Weigh drone on a scale. It must weigh less than 1.4kg±0.1kg	1. Chassis Frame (a) _____/1 (b) _____/3 Total Points:_____/8

Continued on next page

Table 4 – continued from previous page

Requirement	Verification	Point Allocation
9. PCB (a) Output signals biased from 0±0.25V to 3±0.3V* (b) Voltage alarm rings and lights up LED when the input voltage is less than 10.6±0.25V	9. PCB (a) Hook up power supply and use multimeter to measure voltage at output pins (b) SUpply the PCB with 10.8V with power source and see if LED lights up and alarm rings	1. PCB (a) _____/1 (b) _____/2 Total Points:_____/3
		Total Points _____/50

*3.a The six signal corresponds to the six commands that you have in our transceiver.

*9.a The Microcontroller may only take a maximum voltage input of 3.3V to prevent damage to the board.

Appendix B Circuit Schematic

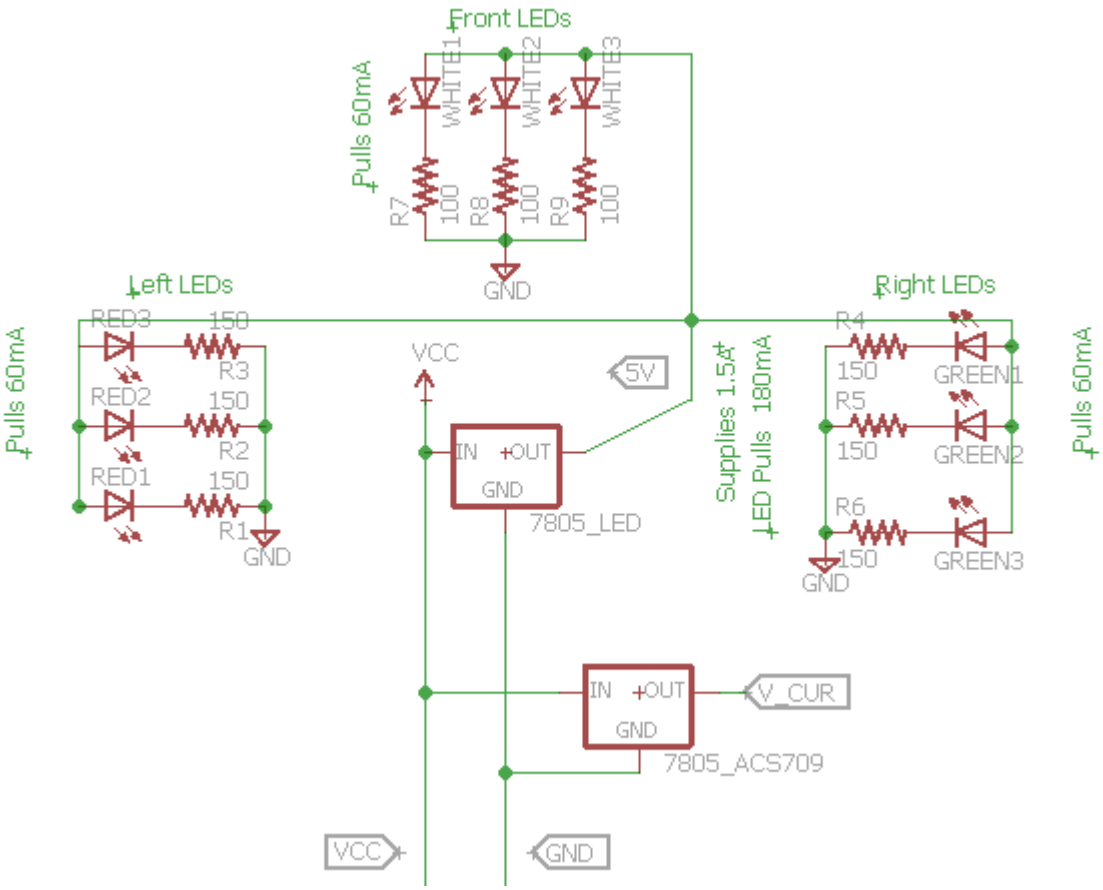

Figure 13: LED module to display orientation of drone

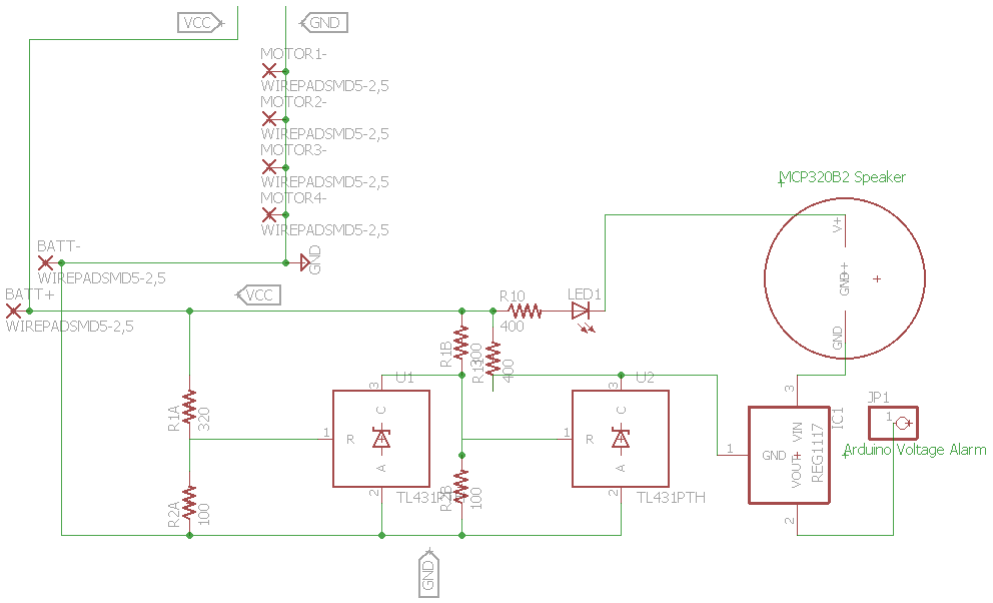

Figure 14: Voltage Alarm module and Mounting pads for motors

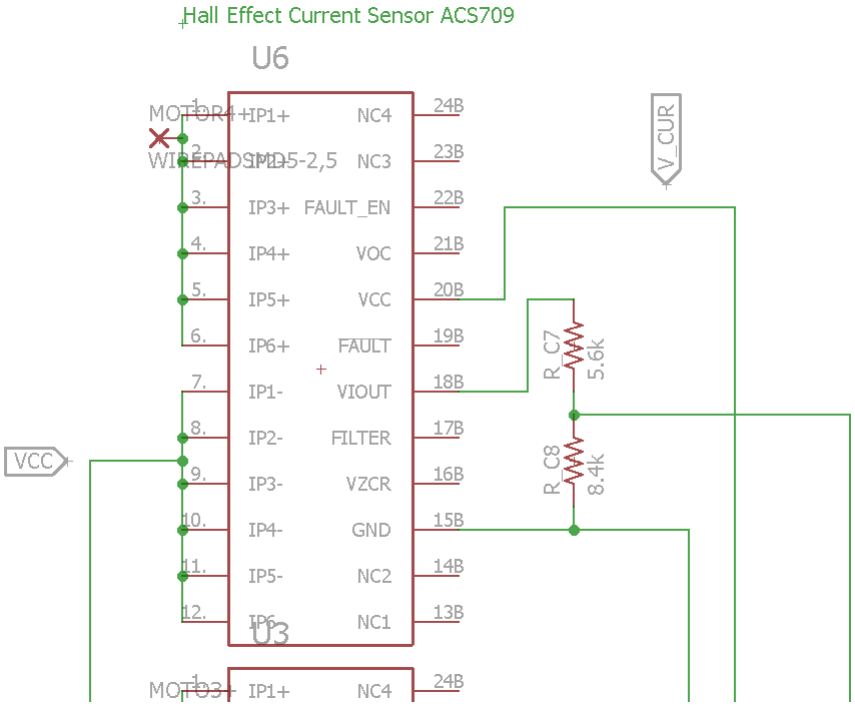

Figure 15: One of the four ACS709 Chips used for sensing current

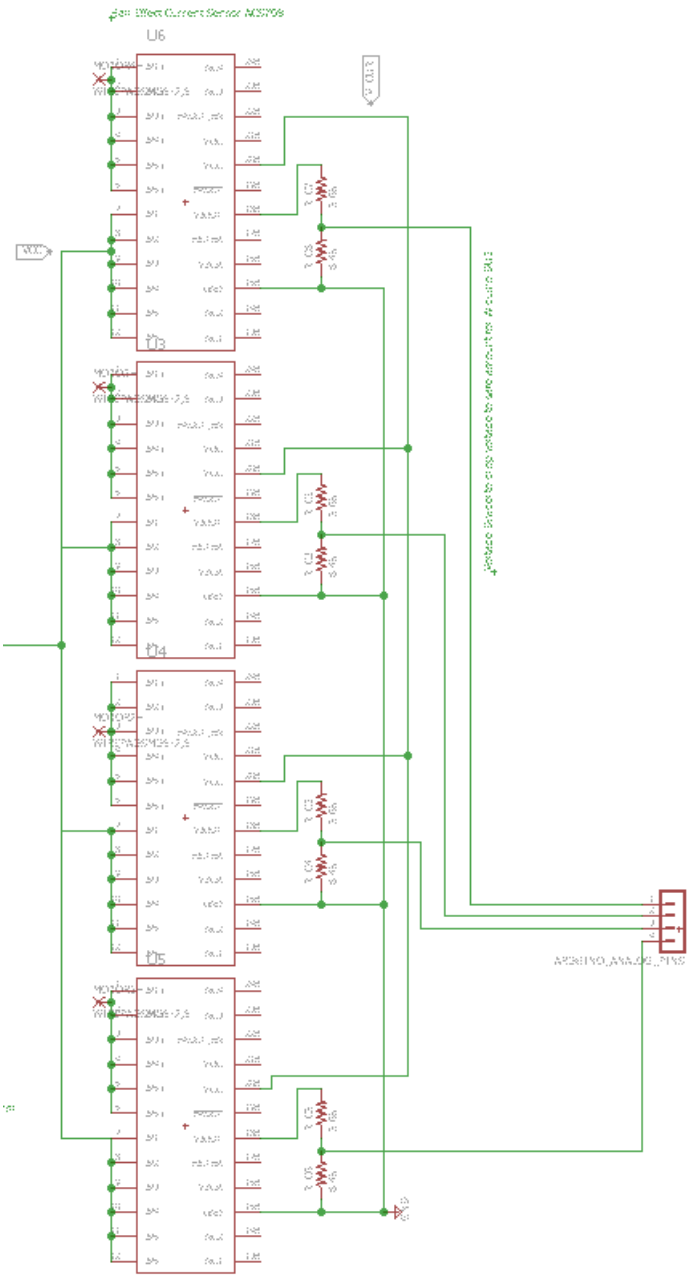

Figure 16: All four of the ACS709 Chips used for sensing current

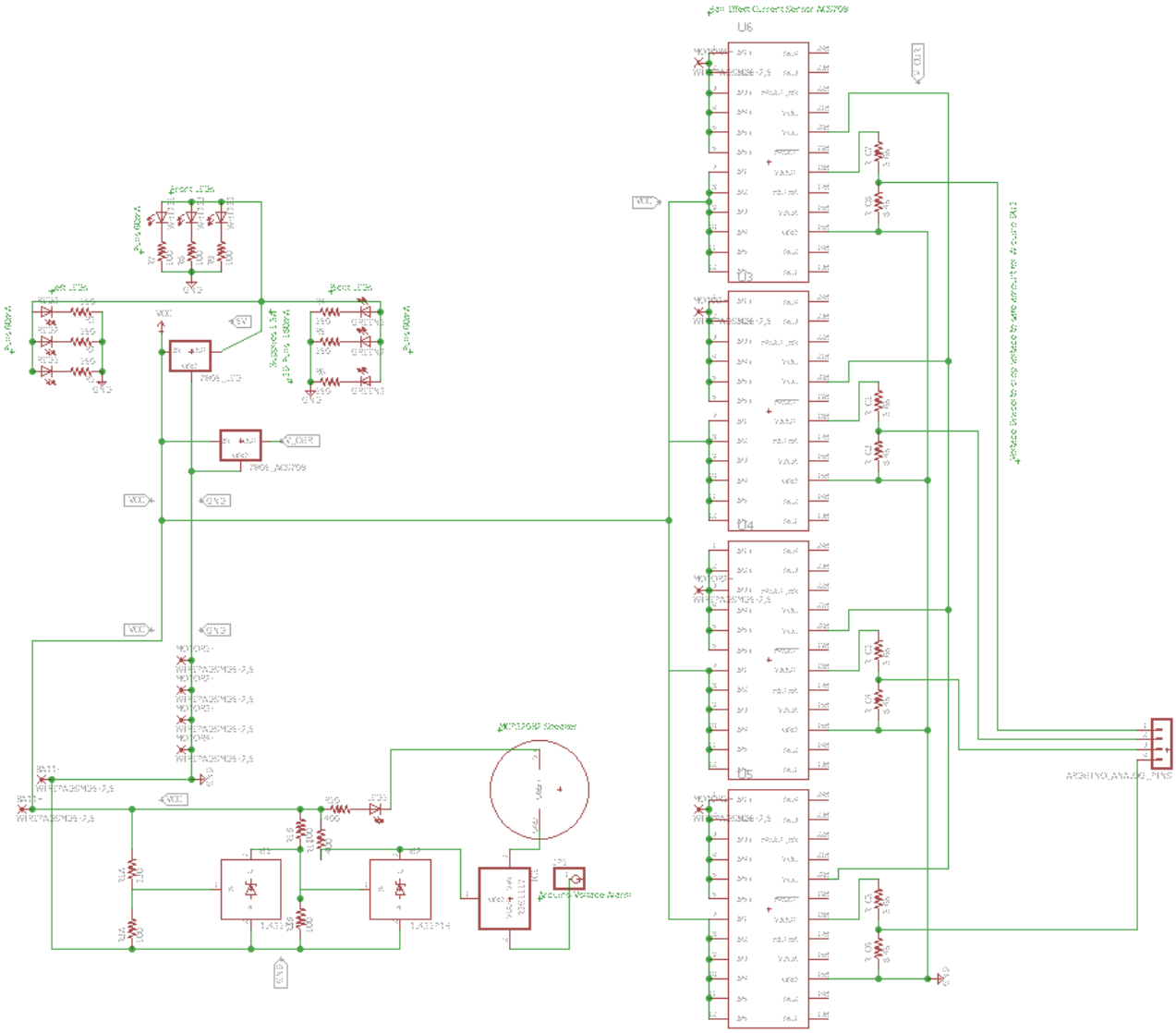

Figure 17: Complete Circuit

B.1 PCB Design

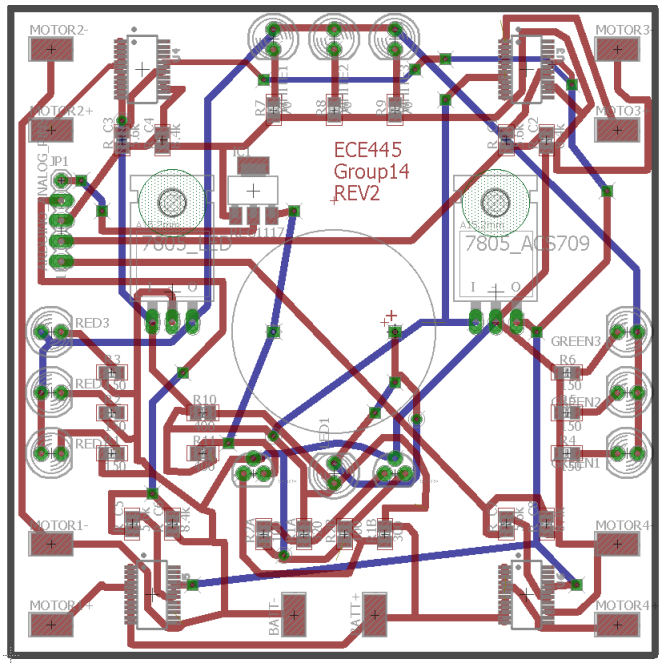

Figure 18: PCB design from EAGLE CAD

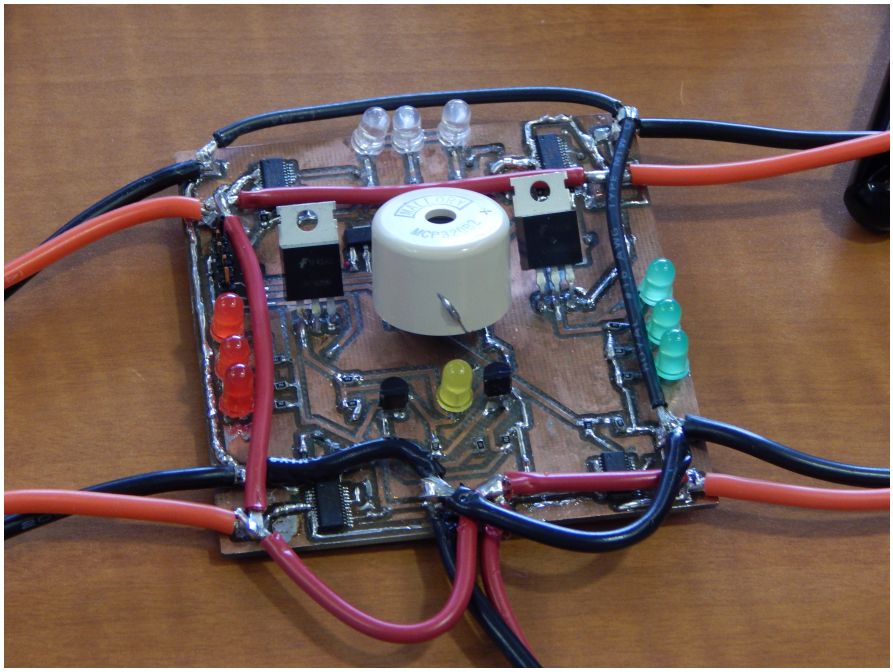

Figure 19: Completed PCB

Appendix C Software Flowchart

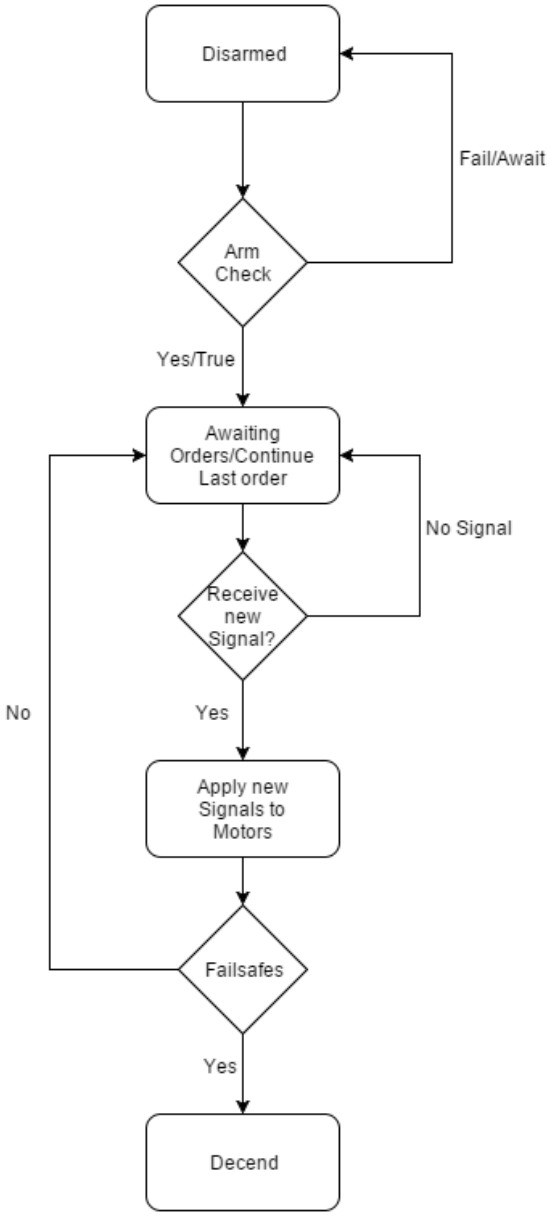

Figure 20: Flowchart for drone

Appendix D Schedule

Table 5: Schedule

Week	Task	Responsibility
9-Feb	1. Finalize Proposal 2. Study dynamics of drone 3. Prepare mock design review	1. Leo and Bree 2. Bree and Ignacio 3. Everybody
15-Feb	1. CAD Centrifugal fan 2. Design ESC 3. Prepare design review	1. Ignacio and Leo 2. Bree 3. Everybody
22-Feb	1. Purchase microcontroller 2. Purchase sensors and test 3. Purchases motors and test 4. Purchase centrifugal fan materials and prototype 5. Finalize Design Review	1. Leo 2. Bree 3. Ignacio 4. Leo 5. Everybody
29-Feb	1. Assemble Control Mechanism 2. Reiterate centrifugal design 3. Assembly prototype propulsion mechanism 4. Assemble acutuation and Chassis	1. Leo 2. Leo 3. Ignacio 4. Bree
7-Mar	1. Prepare mock demo 2. Finalize PCB Design 3. Run more tests on control 4. polish centrifugal fan design	1. Ignacio 2. Bree 3. Leo 4. Leo
14-Mar	1. Finalize mock demo 2. Polish drone design 3. Run final test on drone capability 4. Prepare individual progress report	1. Everybody 2. Bree and Ignacio 3. Leo 4. Everybody
4-Apr	1. Fix any remaining issues 2. Prepare presentation 3. Prepare demonstration	1. Leo 2. Bree 3. Ignacio

Continued on next page

Table 5 – continued from previous page

Week	Task	Responsibility
11-Apr	1. Prepare Final paper 2. Finalize presentation 3. Finalize demonstration	1. Everybody 2. Bree 3. Ignacio
2-May	1. Finalize Final paper 2. Lab checkout	1. Everybody 2. Leo

Appendix E Testing Apparatus

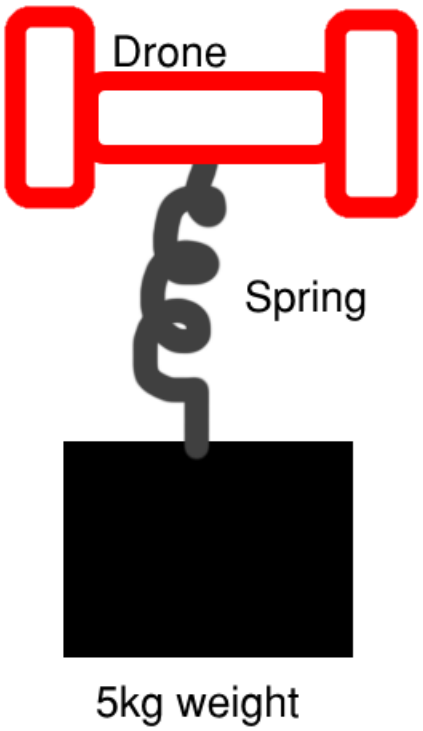

Figure 21: Testing apparatus for on-campus testing

Appendix F Code Snippets

```
typedef struct control_state{
        //holds the updated bit flags
        volatile unsigned char bUpdateFlagsShared;
        volatile unsigned short unThrottleInShared;
        volatile unsigned short unRudderInShared;
        volatile unsigned short unElevatorInShared;
        volatile unsigned short unAileronInShared;
        volatile unsigned short unGearInShared;
        unsigned char bUpdateFlags;
        unsigned int ulThrottleStart=0;
        unsigned int ulGearStart=0;
        unsigned int ulRudderStart=0;
        unsigned int ulElevatorStart=0;
        unsigned int ulAileronStart=0;

        int throttle;
        int rudder;
        int elevator;
        int aileron ;
        int gear;

        //Boolean to see if drone is amred
        bool armed = false;
        //Boolean to see if signal is good
        bool signalGood = false;
        //PID values
        double pid_roll_in, pid_roll_out, pid_roll_setpoint = 0;
        double pid_pitch_in, pid_pitch_out, pid_pitch_setpoint = 0;
        double pid_yaw_in, pid_yaw_out, pid_yaw_setpoint = 0;
        //Angles values for the sensor
        double angleX = 0;
        double angleY = 0;
        double angleZ = 0;

}control_state_t;
```

Figure 22: Control State Code

```
//Function to read in the signals
uint8_t COMMS::readSignals(control_state *ctrl)
{
  if(ctrl->bUpdateFlagsShared)
  {
    //turn off interrupts to take local copies of the sharead variables
    noInterrupts();
    //take local copy of which channels were updated
    ctrl->bUpdateFlags = ctrl->bUpdateFlagsShared;
    if(ctrl->bUpdateFlags & THROTTLE_FLAG)
    {
      ctrl->throttle = ctrl->unThrottleInShared;
    }
    if(ctrl->bUpdateFlags & AILERON_FLAG)
    {
      ctrl->aileron = ctrl->unAileronInShared;
    }
    if(ctrl->bUpdateFlags & ELEVATOR_FLAG)
    {
      ctrl->elevator = ctrl->unElevatorInShared;
    }
    if(ctrl->bUpdateFlags & RUDDER_FLAG)
    {
      ctrl->rudder = ctrl->unRudderInShared;
    }
    ctrl->bUpdateFlagsShared = 0;
    interrupts();//Turn interrupts back on
  }
  ctrl->bUpdateFlags = 0;
}
```

Figure 23: Communication Code - Read Signals

```
//Function to calculate the value read by the receiver
void calcThrottle()
{
  if (digitalRead(THROTTLE_PIN) == HIGH)
  {
    my_state.ulThrottleStart = micros();
  }
  else
  {
    my_state.unThrottleInShared = (short)(micros() - my_state.ulThrottleStart);
    my_state.bUpdateFlagsShared |= THROTTLE_FLAG;
  }
}
```

Figure 24: Communication Code - Calculate Signal

```
//Set motor velocities in terms of miliseconds
uint8_t MOTORS::updateMotors(int front, int back, int right, int left)
{
  /*
   * m0 = front left = throttle + pid_roll_out
   * m1 = front right= throttle + pitch
   * m2 = back right = throttle - pid_roll
   * m3 = back left = throttle - pid_pitch
  */
  ESC1.writeMicroseconds(front);
  ESC2.writeMicroseconds(back);
  ESC3.writeMicroseconds(right);
  ESC4.writeMicroseconds(left);
}
```

Figure 25: Motor Control Code

```
//Function to update the motor values
void updateCont()
{
  int m0, m1, m2, m3;
  int throttle_value;
  setpoint_update();
  pid_update();
  pid_compute();
  //Mapping values to useable range
  throttle_value = map(my_state.throttle, SIGNAL_MIN, SIGNAL_MAX, MOTOR_MIN, MOTOR_MAX);
  m0 = throttle_value + my_state.pid_roll_out;
  m1 = throttle_value + my_state.pid_pitch_out;
  m2 = throttle_value-50 - my_state.pid_roll_out;
  m3 = throttle_value - my_state.pid_pitch_out;
  my_motors.updateMotors(m0, m1, m2, m3);
  //Serial.println(m0);
  my_motors.updateServos(my_state.rudder);
  }
```

Figure 26: Updating motors with controller

Appendix G CAD Models

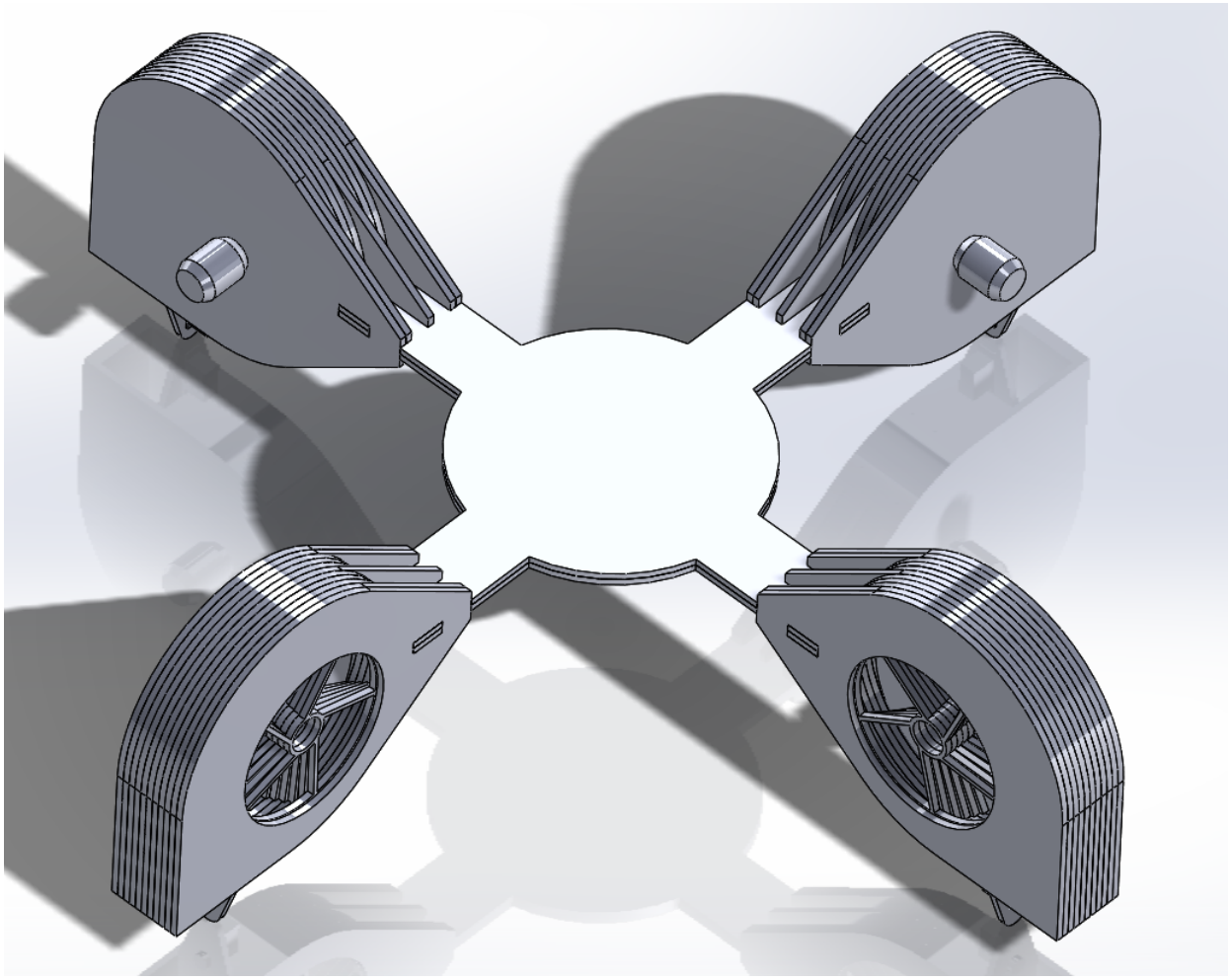

Figure 27: Isometric view of Drone CAD

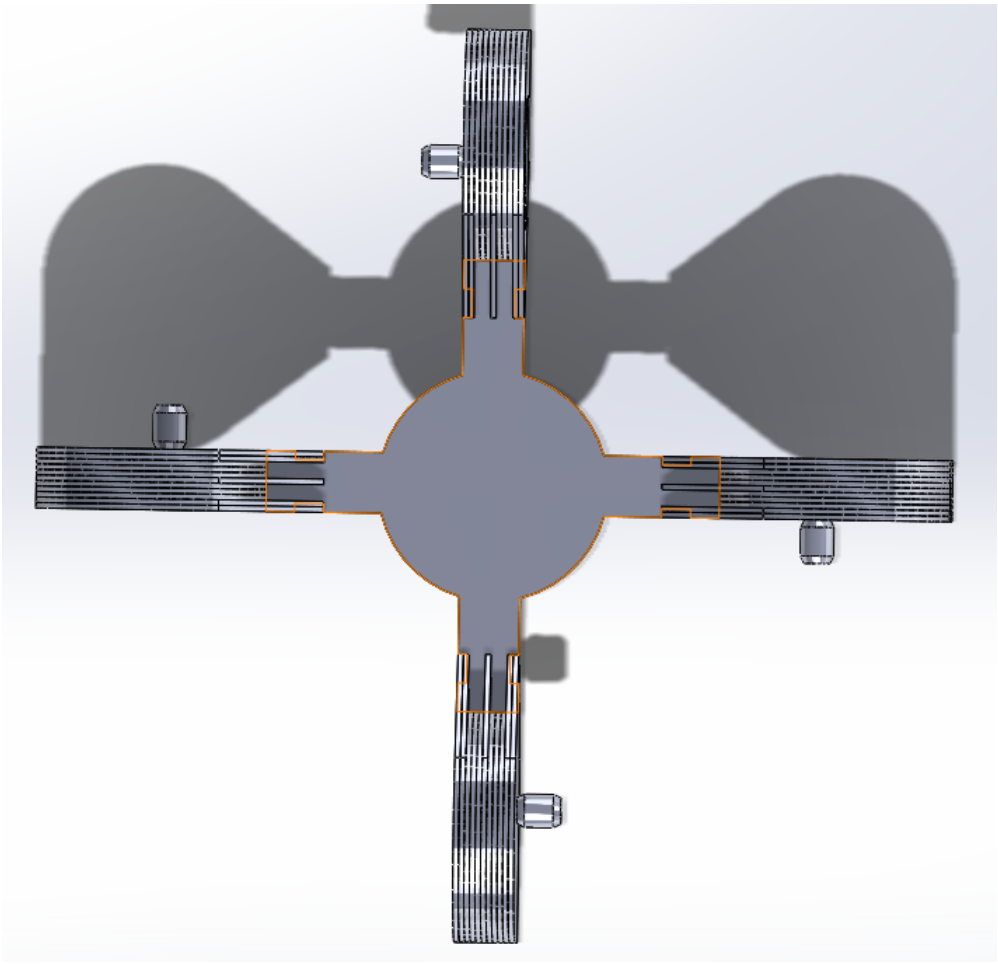

Figure 28: Top View of Drone CAD

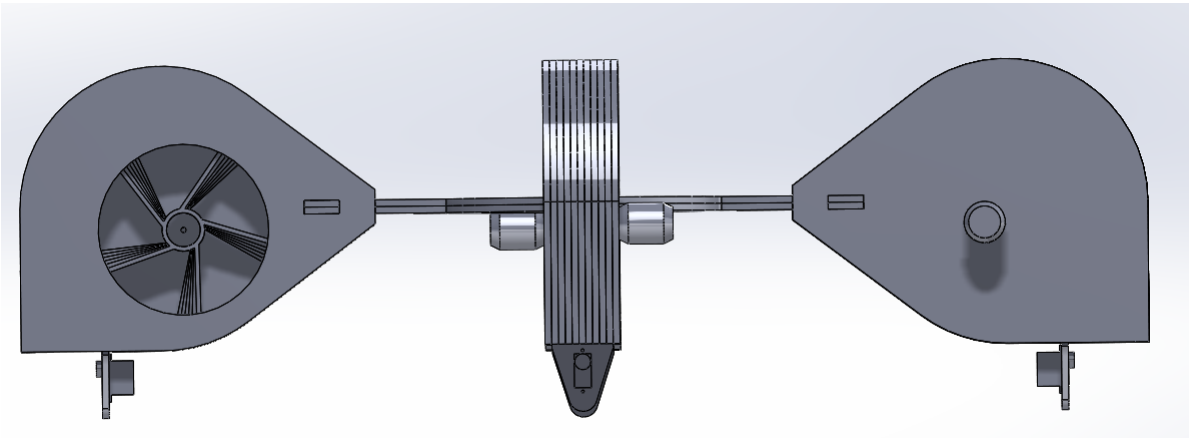

Figure 29: Side View of Drone CAD

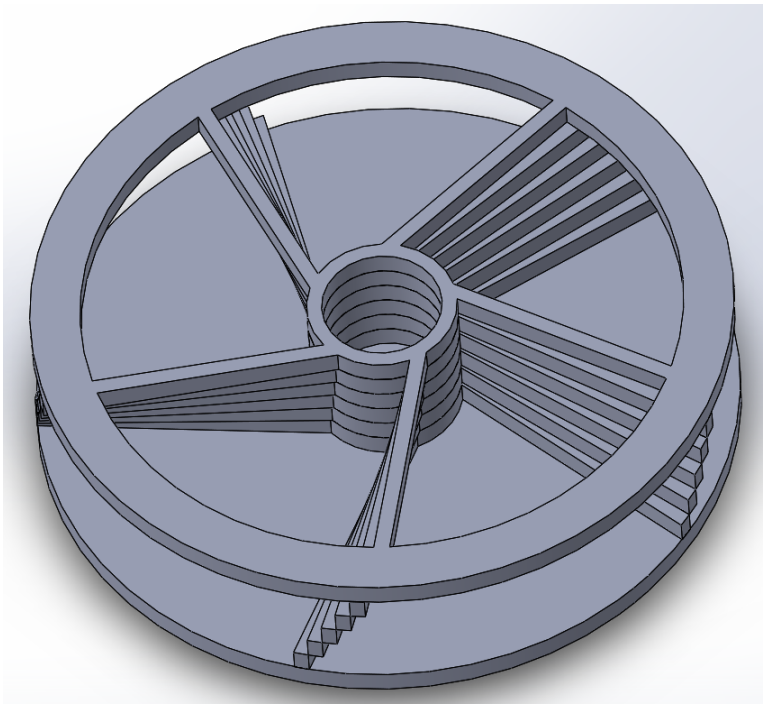

Figure 30: Impeller for Counter Clockwise Fan

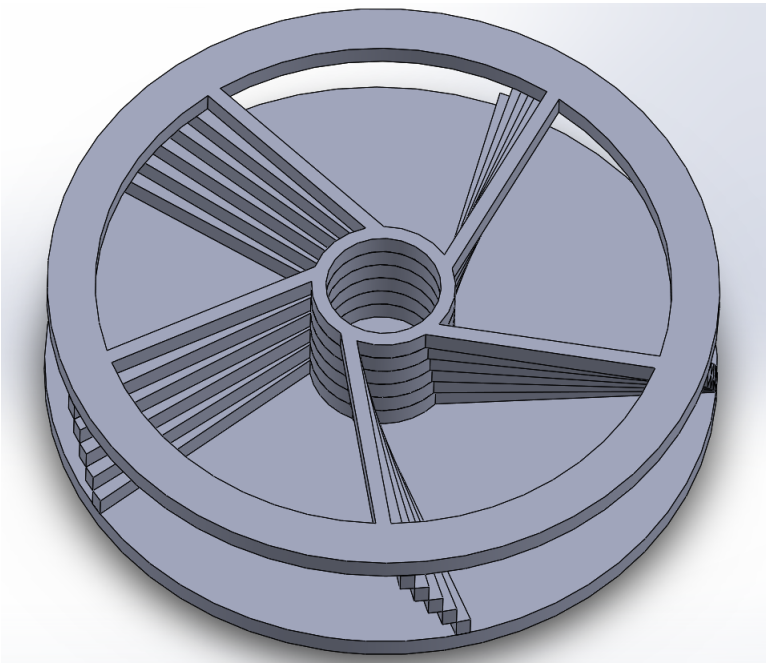

Figure 31: Impeller for Clockwise Fan

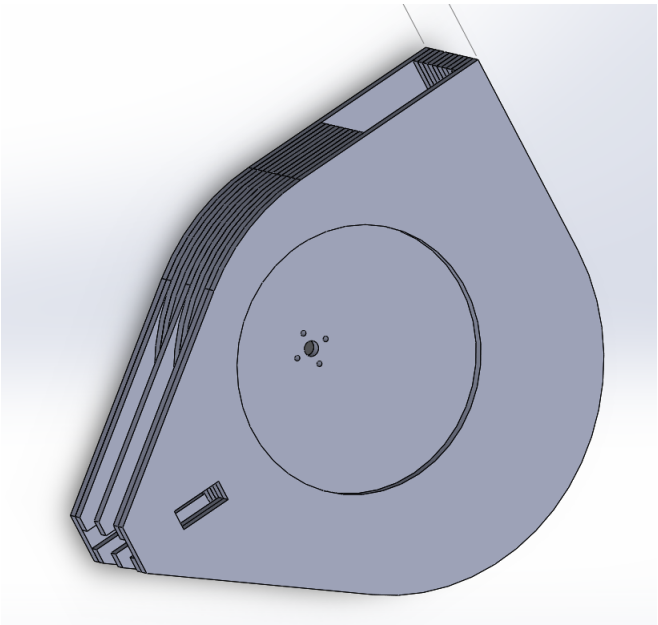

Figure 32: Turbine Housing for Counter Clockwise Fan

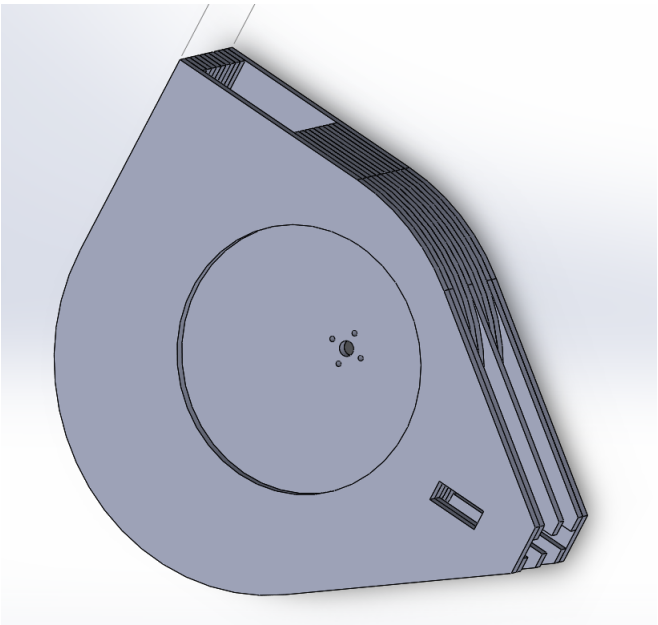

Figure 33: Turbine Housing for Clockwise Fan

Appendix H Sources

References

[1] Amazon.com, "Amazon Price Air",2016.[Online].
Available: http://www.amazon.com/b?node=8037720011.[Accessed: 03-Feb-2016].

[2] Ieee.org,"IEEE IEEE Code of Ethics", 2016. [Online].
Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 05- Feb- 2016].

[3] Faa.gov,"Model Aircraft Operations",2016.[Online].
Available: https://www.faa.gov/uas/model aircraft /. [Accessed: 05- Feb- 2016].

[4] Horizon Hobby, Lithium Polymer Batteries,2016. [Online].
Available: https://www.horizonhobby.com/pdf/EFL-LiPoSafetyWarnings.pdf. [Accessed: 15-Feb-2016]

[5] Arduino.cc, "Arduino - ArduinoBoardDue", 2016. [Online].
Available: https://www.arduino.cc/en/Main/ArduinoBoardDue. [Accessed: 16- Feb- 2016].

[6] Spektrumrc.com, "DX6i DSMX 6-Channel Full-Range without Servo MD2 (SPM6610): Spektrum - The Leader in Spread Spectrum Technology", 2016. [Online]. Available: http://www.spektrumrc.com/Products/Default.aspx?ProdID=SPM6610. [Accessed: 28- Feb-2016].

[7] Spektrumrc.com, "AR600 6-Ch Sport DSMX Receiver (SPMAR600): Spektrum - The Leader in Spread Spectrum Technology", 2016. [Online]. Available: http://www.spektrumrc.com/Products/Default.aspx?ProdId=SPMAR600. [Accessed: 28- Feb-2016].

[8] Arduino.cc, "Arduino - Compare", 2016. [Online]. Available: https://www.arduino.cc/en/Products/Compare. [Accessed: 28- Feb- 2016].

[9] Faa.gov, "Unmanned Aircraft Systems (UAS) Registration", 2016. [Online]. Available: https://www.faa.gov/uas/registration/. [Accessed: 28- Feb- 2016].

[10] Shepard,Sam. "This is NOT a Propeller", YouTube, 2016. [Online]. Available: https://www.youtube.com/watch?v=wg8ZbiC9IBw. [Accessed: 22- Mar- 2016].

[11] Piazza.com, 2016. [Online]. Available: https://piazza.com/class/ii3qa1u2pdp6w5?cid=70. [Accessed: 22- Mar- 2016].

[12] "Technical LED Color Chart", Oksolar.com, 2016. [Online]. Available: http://www.oksolar.com/led/led color chart.htm. [Accessed: 22- Mar- 2016].

[13] "Arduino Current Sensor", Hacktronics.com, 2016. [Online]. Available: http://www.hacktronics.com/Tutorials/arduino-current-sensor.html. [Accessed: 22- Mar- 2016].

[14] "TL43xx Precision Programmable Reference", Texas Instruments, 2016. [Online]. Available: http://www.ti.com/lit/ds/symlink/tl431.pdf. [Accessed: 22- Mar- 2016].

[15] "AfroESC 30A USER MANUAL", Arduino-Ua, 2016. [Online]. Available: https://arduino-ua.com/docs/AfroESC30A.pdf. [Accessed: 22- Mar- 2016].

[16] "Arduino - PulseIn", Arduino.cc, 2016. [Online]. Available: https://www.arduino.cc/en/Reference/PulseIn. [Accessed: 22- Mar- 2016].

[17] "How does a Brushless ESC work? - RC Groups", Rcgroups.com, 2016. [Online]. Available: http://www.rcgroups.com/forums/showthread.php?t=139189. [Accessed: 22- Mar- 2016].

[18] "ACS709-Datasheet", Allegromicro.com, 2016. [Online]. Available: http://www.allegromicro.com/ /media/Files/Datasheets/ACS709-Datasheet.ashx. [Accessed: 29- Mar- 2016].

[19] "Modelling and control of quadcopter", Teppo Luukkonen, school of Science Aalto University, August 22, 2011 [Online]. Available: http://sal.aalto.fi/publications/pdf-files/eluu11public.pdf

[20] "MODELADO, DISENO, CONSTRUCCION Y CONTROL BORROSO DE UN HELICOPTERO QUADROTOR", Carlos Mikel Esparza Martinez de Luco, Jorge Elso Torralba, Maria Jose Perez-Ilzarbe Serrano,Pamplona, 24 de febrero de 2012 [Online]

[21] "PID", multiwii.com, 2016 [Online]. Available:http://www.multiwii.com/wiki/?title=PID

[22] "MPU6050 DMP" Jeff Rowberg [Online] Available: https://github.com/jrowberg/i2cdevlib

[23] "RCArduino: How To Read Multiple RC Channels", Rcarduino.blogspot.com, 2012. [Online]. Available: http://rcarduino.blogspot.com/2012/04/how-to-read-multiple-rc-channels-draft.html. [Accessed: 04- May- 2016].

[24] Clarage.com, 2016. [Online]. Available: http://www.clarage.com/docs/fan-engineering-letters/fan-performance-characteristics-of-centrifugal-fans—fe-2400.pdf?sfvrsn=2. [Accessed: 05- May- 2016].