Turbo-Multirotor Drone

By

Ignacio Aguirre Panadero

Bree Peng

Leo Yamamae

Final Paper for ECE 445, Senior Design, Spring 2016

TA: Luke Wendt

4 May 2016

Project No. 14

Abstract

This project focuses on a drone that uses centrifugal fans instead of propellers. Propellers on drones are dangerous due to their high RPM typically greater than 20k RPM. When plastic propellers come into contact with a harder material, the plastic shatters leaving sharp shards and when the propeller comes into contact with a soft material like flesh, it will slice through it. We wanted design a safer alternative to propellers on drones as well as contribute to the current drone technology. We took inspiration from the "This is NOT a Propeller" video from Samm Shepard of using a centrifugal fan for propulsion. From our design we achieved a peak thrust of 550g from each fan and was able to control the drone using a Arduino microcontroller. When shattering our design broke apart like safety glass into harmless pieces of paper and polystyrene. In conclusion, we were able to explore another method of propulsion for a drone that is safer than propellers.

Contents

1 Introduction 1

1.1 Purpose 1

1.2 Objectives 1

1.3 Function and Features 1

2 Design 2

2.1 Block Diagram 2

2.2 General Schematic 3

2.3 Block Descriptions 4

2.3.1 Power Supply 4

2.3.2 Transmitter 4

2.3.3 Receiver 4

2.3.4 Electronic Speed Controller 4

2.3.5 Motors 5

2.3.6 Centrifugal Fan 5

2.3.7 Controller 5

2.3.8 IMU sensor 5

2.3.9 Actuator 6

2.4 Mechanical Design 7

2.4.1 A/R Ratio 7

2.4.2 Materials 10

2.4.3 Manufacturing 11

2.5 Code Design 12

2.5.1 Flow Chart 12

2.5.2 Control State 12

2.5.3 Communication 12

2.5.4 Motor Control 13

2.6 Custom Hardware 14

2.6.1 Circuit Overview 14

2.6.2 LED Module 14

2.6.3 Voltage Alarm 14

2.6.4 Current Sensor 15

2.7 Dynamics . . . 16
2.8 State Equations . . . 18
2.9 SISO system . . . 18
2.10 Control . . . 19
2.11 PID controller . . . 20
2.12 Cost Analysis . . . 22
3 Ethical / Legal Considerations . . . 23
3.1 IEEE Code of Ethics Considerations . . . 23
3.2 Legal Obligations . . . 23
3.3 Safety Statement . . . 24
4 Conclusion . . . 25
Appendix A Requirements and Verification . . . 26
Appendix B Circuit Schematic . . . 30
B.1 PCB Design . . . 34
Appendix C Software Flowchart . . . 35
Appendix D Schedule . . . 36
Appendix E Testing Apparatus . . . 38
Appendix F Code Snippets . . . 39
Appendix G CAD Models . . . 42
Appendix H Sources . . . 46

1 Introduction

1.1 Purpose

Drones are becoming very common today. Drones can be seen in the business, receational, and defense sectors in the US. For example Amazon has a project to deploy drones for delivery tasks [1]. One major problem with current drones is its durability and danger. Propellers on drones easily shatter when it collides with a hard surface and propellers can easily tear skin. There are some way to combat this issue. For example, one can attach a propeller guard with shields the propellers from the sides but not the top or the bottom. However, using the guards will not prevent the propellers from shattering when the drone crashes nor will it prevent unlucky people from slicing their fingers off.

Our project aims to make drones safer by utilizing centrifugal fans instead of propellers. A centrifugal is another method to generate thrust without the use of propellers. By nature, the fan is enclosed in an outer shell that will shield potential victims from the impeller.

1.2 Objectives

- Design a drone that flies without propellers
- Design a more robust drone than most on the market

1.3 Function and Features

- Flies like a commercial drone (control-wise)
- Interpret sensor data and determine stability
- Lightweight and portable
- Balanced design for stability
- State feedback control
- Low power, High thrust

2 Design

2.1 Block Diagram

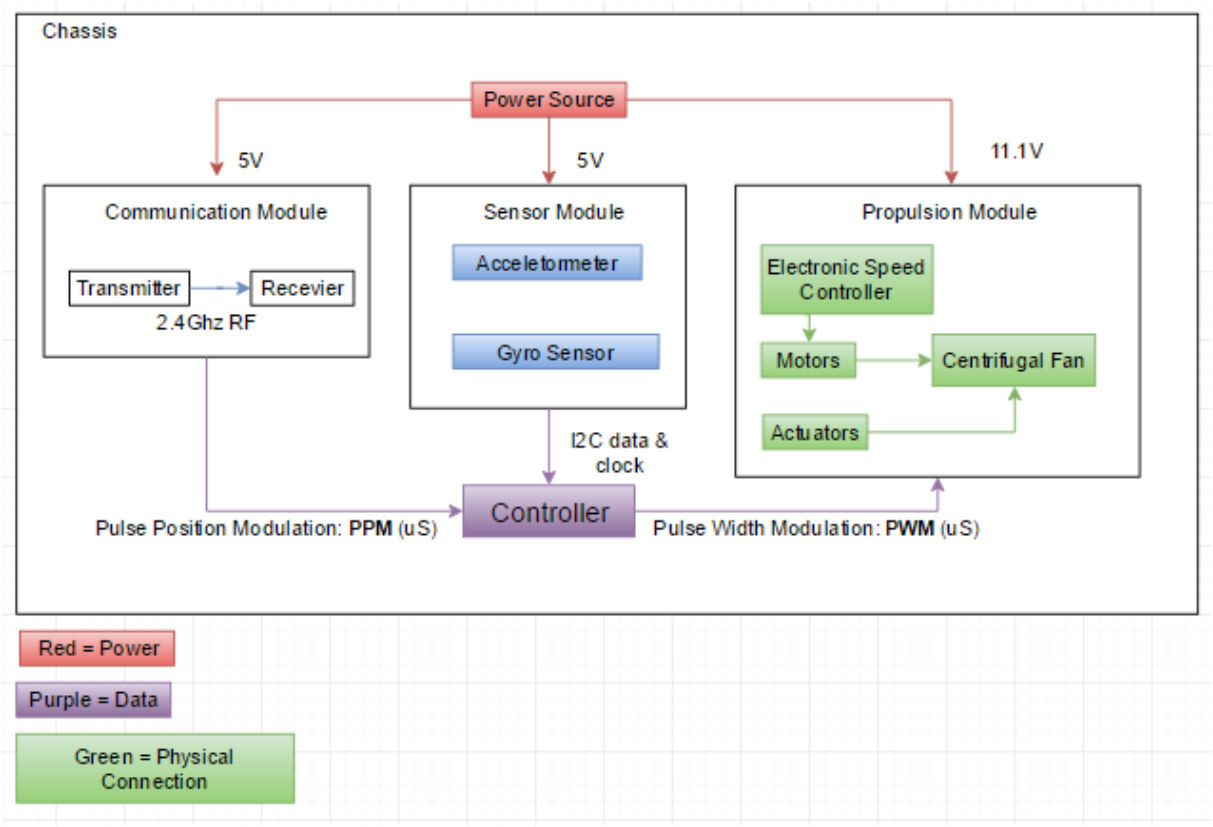


Figure 1: Block Diagram with the modules

The figure shown in Figure 1 is a general block diagram which shows the main modules of the project. The main modules are Power, Communication, Sensor, Propulsion, and Controller.

2.2 General Schematic

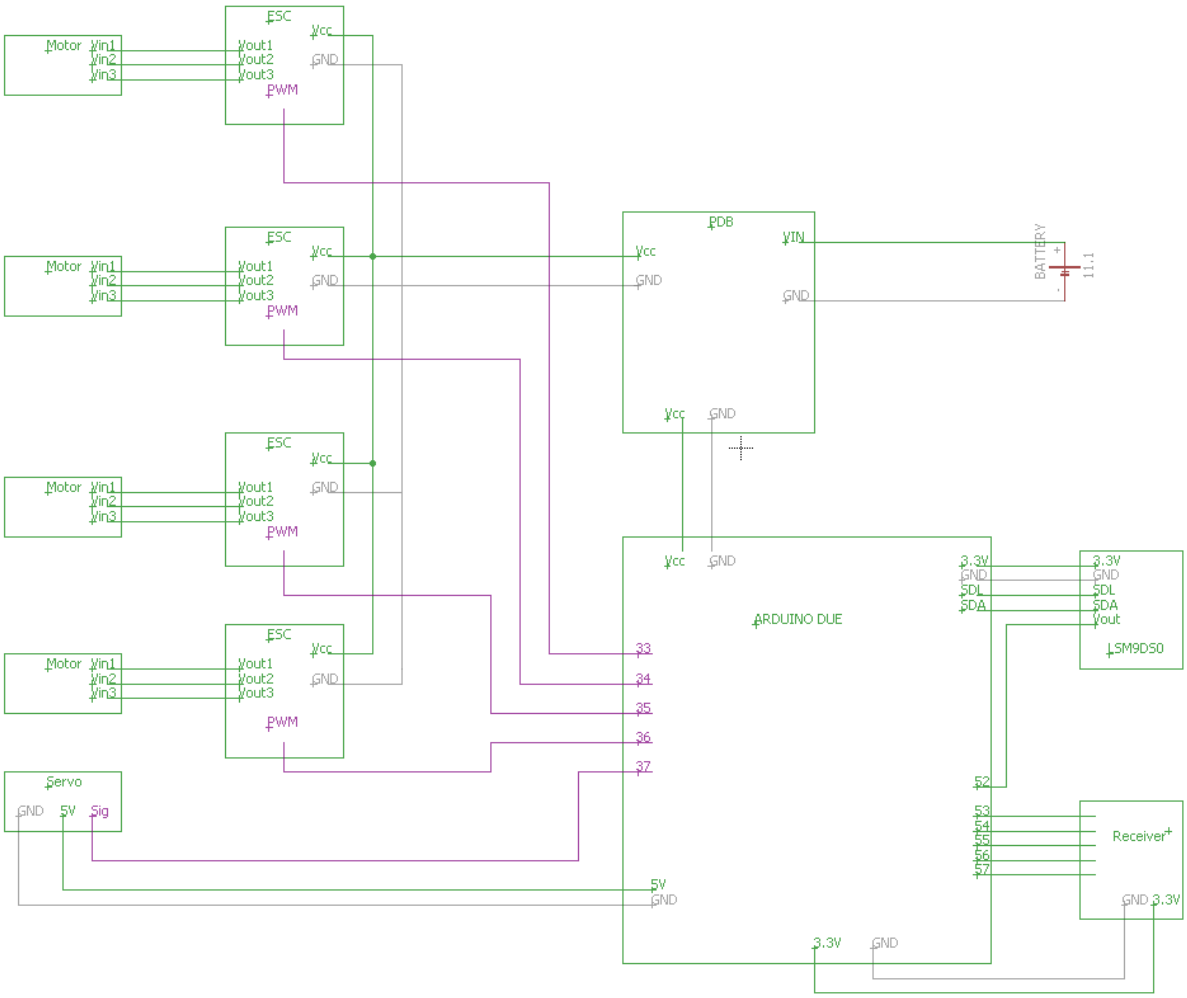


Figure 2: Genera Schematic for Drone

The general schematic shown in Figure 2 shows the overview of components that will be connected together.

2.3 Block Descriptions

2.3.1 Power Supply

The power supply distributes power for all components in the system. Lithium Polymer Batteries has been chosen for our project due to its high Power/Weight Ratio. Since the motors selected for our project draws a nominal current of 30A, the battery must be able to supply at least 120A for the 4 four motors used in the project.

The specific battery chosen for this project is the Turnigy Bolt 2200mAh 3S 65-130C 25.08Wh Lithium Polymer battery. This battery operates at 3S equivalent to 11.1V with a capacity of 2200mAh and a 65-130C Discharge rate. We can calculate the amount of current this battery can provide by multiplying the capacity by its discharge rate. The following is the calculation performed:

$$Current_{nominal} = 2200mAh * 65C * \frac{1A}{1000mAh} = 143A \tag{1}$$

$$Current_{burst} = 2200mAh * 130C * \frac{1A}{1000mAh} = 286A \tag{2}$$

2.3.2 Transmitter

The transmitter allows an operator to send signals to the receiver using pulse poistion modulation or PPM for short. For our project, we chose the Spektrum Dx6i as our transmitter. The Spektrum Dx6i transmitter uses the 2.4Ghz band and the DSM2 modulation[6]. The Dx6i supports up to 6-channels where 4 of the 6 is required for the drone movement, and the other 2 channels may be used for expanding the project or for safety features. The Dx6i was chosen for this projects as it provides a reliable communication to the receiver and is a similar type of controller used for typical drone and plane operation.

2.3.3 Receiver

The receiver is the component that will take in the signals from the transmitter. The receiver must be able to receive at least 6 channels of data in order to maximize our drone's potential.For our project,we chose the Spektrum AR600.The AR600 operates on the DSM2 modulation as well as the 2.4Ghz Band[7]. The receiver also supports up to 6-channels and is compatible with the Dx6i for our project.

2.3.4 Electronic Speed Controller

The electronic speed controllers (ESC) are the components that are calibrated to a range of PWM (Pulse Width Modulation) or PPM (Pulse Position Modulation) signals, and uses these to determine the speed at which they should be driving the motors. Each must have current-handling capabilities that include a safety factor in order to properly drive each motor on the drone without jeopardizing the integrity of the overall control system. For this project, we will be using the HobbyKing SS Series 50-60A ESC. This ESC is able to control the speed of a brushless motor and can handle up to 50A continuous and 60A. This ESC is perfect for our project because it has many features included like overheating protection as well a smooth linear throttle response and a auto-shutoff when signal is lost.

2.3.5 Motors

The motors are the elements that will allow the machine to move. Large brushless outrunner motors with a KV rating of greater than 2670. KV rating is the RPM generated per volt applied to the motor. Therefore, if we apply 11.1V to a 3400kV motor, the motor will be spinning at 44400 RPM. Brushless motors are chosen due to its light weight, lower operating costs, lower friction, higher efficiency, and high power they provide. This motor will need to be able to handle a great amount of torque for it to not burn during operation of the drone. For this reason, the NTM Rotor drive 450 Series 3400kV/500W motor is chosen. This motor pulls from 30A continuous and 46A burst. From these values, we can calculate the amount of current the power supply must provide.

$$Current_{required,nomial} = 4motors * 30A = 120A \tag{3}$$

$$Current_{required,burst} = 4motors * 46A = 184A \tag{4}$$

2.3.6 Centrifugal Fan

The centrifugal fan is the key component in the project. The centrifugal fan give the project its uniqueness. This and the use of the servo creates an alternative propulsion method for drones of this size. The centrifugal fan is similar to an automotive intake compressor found in cars equipped with turbines or belt-driven compressors. It has a circular hole on its large face which takes in air and then a spinning circular blade compresses the air and pushes it out the nozzle on the side of the round housing. This part will be custom-made. The part will be made with polystyrene impeller and housing. These materials are chosen mainly due to their weight.
Their CAD images can be found in Appendix G.

2.3.7 Controller

The controller is the brain of the drone. The controller will use signals from the receiver and L2C sensor and then compute the necessary values to output to the Speed controllers. The controller we chose for our project is the Arduino DUE. The Arduino DUE has 12 PWM pins which is used to send signals to the speed controllers and has a clock time of 84MHz.[8].

2.3.8 IMU sensor

For this project the sensor chosen is the MPU6050 with 6DoF(3 gyroscope axis+3 accelerometer axis) on the breakout board GY-521. The sensor uses the I2C-bus(for this reason we rejected the data conditioner circuit for the PCB requirement) to interface with the Arduino.
To obtain accurate angles a complementary filter algorithm to combine accelerometer and gyroscope measurements. Accelerometer data is noisy on short time scales, and gyroscope data drifts on longer timescales, so the complementary filter combines both for greater accuracy. However, the MPU-6050 contains a digital motion processor (DMP) which can perform the data fusion on the IMU chip iteslf and reports the orientation in yaw/pitch/roll or quaternion format. Though the exact details of how the DMP does this calculations are not published, the DMP is still useful because presumably Invensense has a pretty good fusion algorithm and because the DMP calculations occurs on the MPU-6050 chip, freeing up processor power. For this code we have used the libraries MPU6050.h and I2Cdev.h.
The code is based on Jeff Rowberg work MPU6050 DMP[22].

2.3.9 Actuator

The actuator is the component that is used to angle the centrifugal fans in order to yaw the drone. This is most-likely be a brushless or coreless servo motor due to its power to weight ratio requisites. The servo will will take in a PWM signal and 5V from the controller. The servo will be able to turn 125 degrees of rotation and should be able to output 20oz-in of torque. For this project, we will use the Sparkfun generic sub micro-size servo since we will use the servos to turn the shutters on the centrifugal fan's output port to direct the air flow.

2.4 Mechanical Design

The diagrams and images for the Mechanical Design portion was taken from "Information and Recommendations for the Engineer Fe-2400" by Clarage[24].

2.4.1 A/R Ratio

An A/R ratio is the ratio of the cross sectional area of the path that air travels versus the radius of the centerline of its path. As the cross sectional area of the compressor housing increases, the total volume of air that can flow through that area also increases. As the radius of the housing increases, the total volume of air contained along the flow path increases as well.

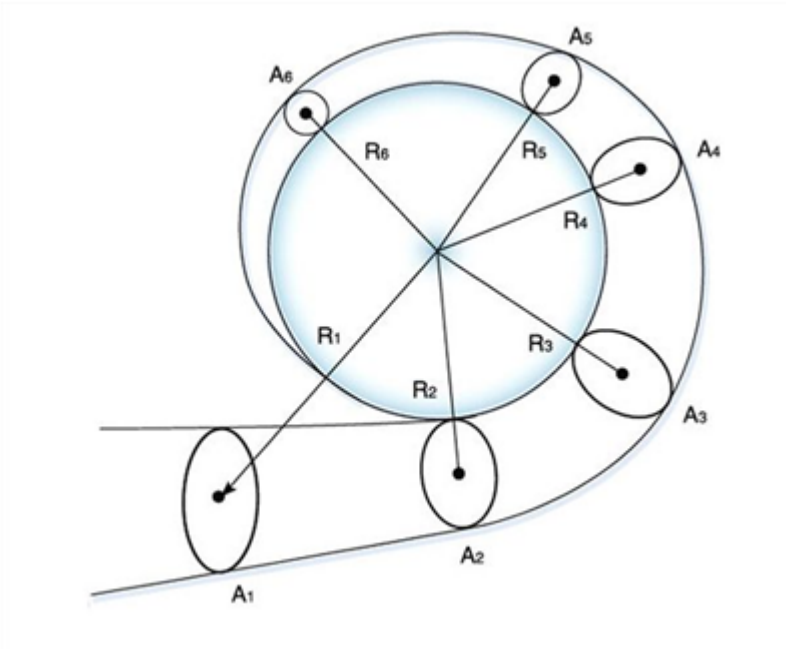


Figure 3: Image of Cross sectional Area vs Radius

Slower response in the centrifugal fan design directly correlates with the size of the fan. Large impellers with greater mass accelerate at a slower rate than smaller impellers with low mass. Larger impellers have a higher air flow than smaller impellers, however they require larger housings, which takes longer to pressurize due to the increased air volume of the housing. Thrust in a centrifugal fan design correlates with flow rate of the fan. A high flow rate means greater thrust. However, it is important to consider the responsiveness of the fan, as a fan that is too greatly optimized for maximum thrust will be very slow to respond. The size increase radially of a centrifugal impeller will also increase load and energy requirements exponentially as rotational kinetic energy is governed by

$$KE_{rotational} = \frac{1}{2} I w^2 \tag{5}$$

where I is the moment of inertia of the impeller, and w is the angular velocity of the impeller. The equation

shows that as the angular velocity of the impeller increases, the kinetic energy of the impeller increases by a 2nd order, and as mass of the impeller increases, the energy requirements increase proportionally as well.

The impeller blade profile greatly affects the characteristics of a centrifugal fan.

A radial bladed centrifugal fan will have peak efficiency earlier, and will reach its pressure and flow peak sooner than a backwards inclined impeller, however will exhibit severe diminishing returns with increases in RPM. This is due to the blunt blade profile with respect to its direction of motion. The flat, perpendicular surface does not impart deflection momentum on the air, but relies on the pressure difference of the fast-moving air at the blade tips versus the slow moving air in the center of the impeller. This design, however, generates a lot of turbulent flow, which in turn, increases its aerodynamic drag significantly.

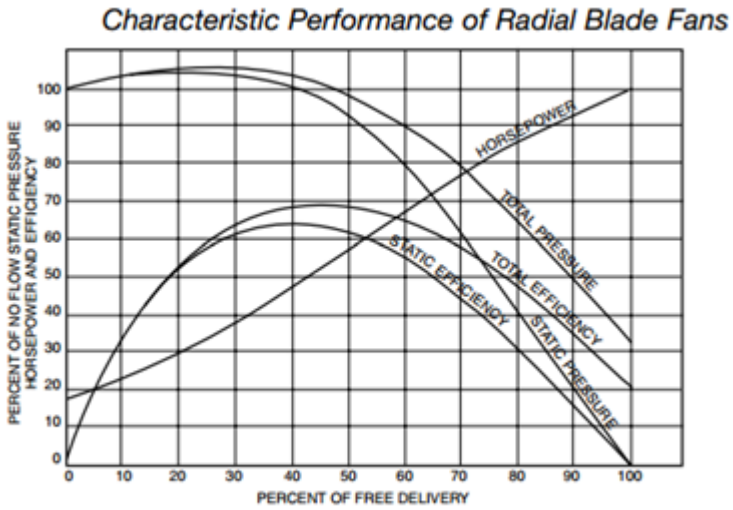


Figure 4: Performance of Radial Blade Fans in graph form

A backward inclined centrifugal fan will have peak efficiency at a higher RPM, but will reach its pressure peak slower than a radial fan. The slanted blade profile imparts deflection momentum to air particles as well as relying on pressure differences. However due to the deflection, the air particles at the tips of the blades leave the rotation of the impeller before reaching high velocity, so the pressure difference is less effective in moving air outwards. Thus, as the RPM increases, more of this pressure difference comes into effect in addition to the deflection, resulting in high flow, and an overall higher RPM peak efficiency. The downside to this, is that it is energetically draining to operate at higher RPM. This is especially important to consider because as the flow velocity of air increases, the force of drag on the impeller will also increase exponentially.

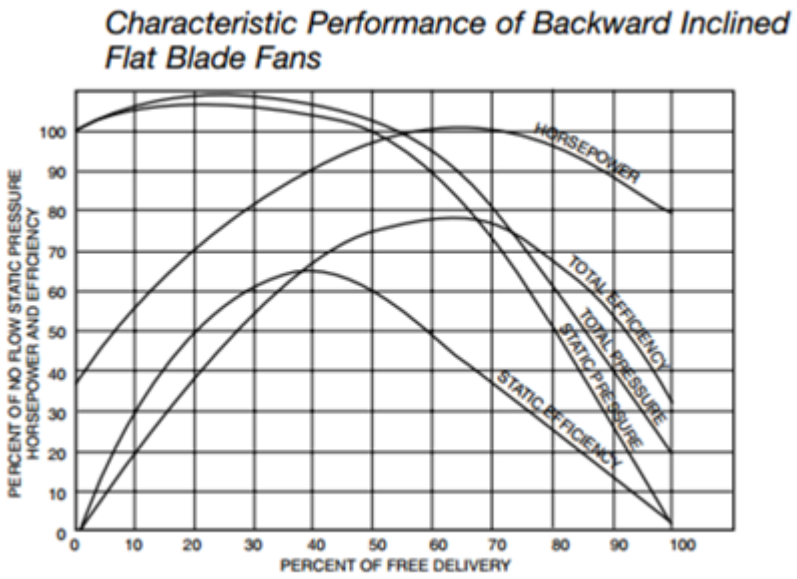


Figure 5: Performance of Backward Inclined Flat Blade Fans in graph form

$$F_D = \frac{1}{2}\rho u^2 C_D A \tag{6}$$

Where u is the flow velocity relative to the object, C_D is the coefficient of drag, A is the cross sectional area of the object, And ρ is the density of air.

A compromise between these two designs is an air wheel design, which is essentially a radial bladed design where each blade is very slightly inclined rearwards. This optimizes the impeller for nid-range RPM efficiency, increased flow compared to the radial blade impeller, and faster pressure build-up than a fully fledged backwards inclined impeller.

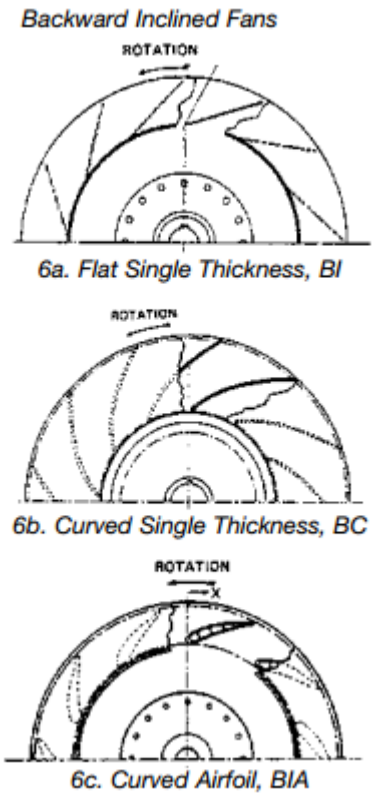


Figure 6: Images of Backward Inclined Fans of different types

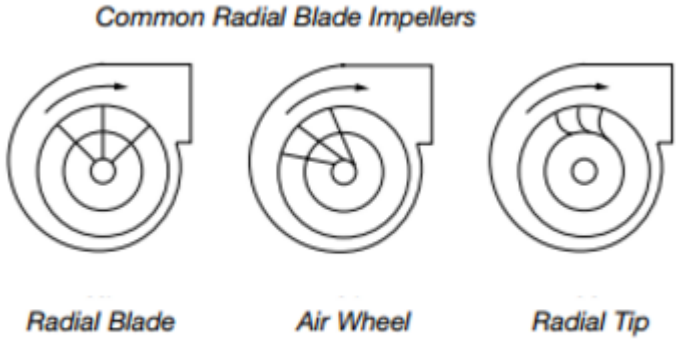


Figure 7: Radial Blade Fan Designs

2.4.2 Materials

For the construction of the frame and impellers, foam board was chosen. We considered corrugated paper board and composites, but neither was appropriate for the application. Corrugated paper board did not have

the stiffness desired, and composites were much too expensive. The foam core board is made from polystyrene foam with paper layered on the top and bottom. This creates a stiff, low-density layered material. It is very lightweight. The material exhibits brittle fracture characteristics which are desirable when considering impact with human appendages or other foreign material for safety purposes. The material undergoes fragmentation when dispersing impact energy, and this fragmentation does not require a high impact to occur. The material is moderately inexpensive, and easy to shape. Corrugated paperboard is a common material for boxes. It is less stiff than foam core, and it is tougher as it does not undergo brittle fracture or fragmentation. However, the bending of the material in response to stress beyond its elastic deformation point increases the amount of time it can apply force to a foreign object, and diminishes the energy dispersal of the impact. This leads to an increase in risk of injury if an impeller comes in contact with an appendage while spinning at high rpm. This material is very inexpensive, and about as easy to shape as the foam core. Composites such as fiberglass, carbon fiber, and kevlar are very stiff as well as very lightweight. They also exhibit brittle fracture, though they do not exhibit fragmentation as foam core does. They have a very high force threshold for elastic deformation, which is likely to guarantee injury should an appendage come in contact with an impeller of this material spinning at high rpm. Additionally, the expense of procuring, manufacturing, and finishing parts utilizing these materials is too great of a burden for our proof-of-concept application

2.4.3 Manufacturing

Manufacturing of the pieces of the drone were done through laser cutting. We decided on laser cutting due to its precision and consistency. Other considerations were hand-cutting the material, but the consistency and precision of such a method would have been greatly detrimental to the projects viability.

2.5 Code Design

The code of the drone was done on the Arduino IDE platform in C/C++. This platform was chosen for our project because of the many available resources on the open-source Arduino as well as many libraries what was available. We will go over the general flow of the software and the main libraries we wrote: control state, communication, and motor control. The important parts of the code can be seen in Appendix F.

2.5.1 Flow Chart

The flow chart shown in Appendix C is the how to software will act generally. The drone first checks if it is given the signal to arm itself and does so. Once the drone is armed, the drone will continue its previous instructions while waiting for any new commands. The instructions will update the motor controllers and while checking for any failsafes. Once a failsafe is detected, the drone will begin its descend.

2.5.2 Control State

The control state, as its name implies holds the states of the drone. This includes the constant values, transceiver values, arm state, alarm state, and the sensor values. We chose to create a control state library as well as other libraries in order to minimize the length of the code in the main .ino file as well as increase compilation speed. By using a control state, we are able to keep local copies of values like the calculated throttle and the previous throttle. By doing this, we have a back up of the data just in case something goes wrong.

A great advantage of using a control state is the ability to use and update variables defined in another library. If we were to approach this issue by creating one large program file, it will look unprofessional as well as prove difficult to debug modules.

The class can be seen in Appendix F Figure 22

2.5.3 Communication

The communication library deals with the reading of signals of the transmitter. There are generally 2 methods to read PPM signals from a receiver. One is using the function pulseIn() and the other is using interrupts. The pulseIn() method is the easiest to implement as it is already provided in the Arduino library. However, the signal reading are very noisy where the values we want and the one we get from the method is different. This is because the system is affected by the other signals and cannot filter the signals correctly. The interrupt method is how we dealt with the communication for our project. This method is harder to implement but produces values much cleaner compared to the pulseIn() method. The interrupt method works by first detecting a signal HIGH on one of the channels and starts a timer in microseconds. When the signal is LOW, signifying the end of the pulse, the duration of the pulse is calculated using the timer started earlier. After this, a bit flag is raised to show that there was a change in signal in one of the channels. When it is time to feed the new signals into the control state, the interrupt is turned off to take local copies of the values and update them according to which flag was raised. After all of the new values are implemented, the bit flag is set back to 0 to signify no changes and interrupts is turn back on. Using this method allows us to avoid any blocking code that will cause noise in the signal we read. This portion of the code was implemented based on the tutorial from the website RCArduino [23].

How we calculated the values can be seen in the Appendix F Figures 23 and 24.

2.5.4 Motor Control

The motor control library handled the writing to the motors as well as arming and disarming the motors. The arming/disarming was done by passing the minimum or zero speed to the speed controller for a duration of 1 second which is translated into disarm and arming the drone. Another task the library handled was writing the values to the motors using the function writeMicroseconds() which as it sounds like, writes a PWM signal with an input parameter of μS. This was a function that came from the servo library in the Arduino platform. When writing the code, the main procedure is to pass in the values we want, map the value to the limits of the speed controller and then use the writeMicroseconds function.

Parts of the code can be seen in the Appendix F Figure: 25 and 26.

2.6 Custom Hardware

2.6.1 Circuit Overview

For our custom hardware, we designed a custom power distribution board with integrated safety features for a drone. The power distribution board has 3 main modules,LED, Voltage Alarm, and a Current Sensor.

2.6.2 LED Module

The LED module was designed to show orientation of the drone. LEDs of the colors, white, red, and green was used to show the front, left, and the right of the drone respectively. LEDs were used instead of only coloring the drone chassis since there are instances where drones are flown in dark areas or far away. It is important to now the drone's orientation so that the operator can compensate for wind and fly properly. The module was designed using the following equation.

$$I = \frac{V_B - V_A}{R} \tag{7}$$

Using this equation, we were able to determine the resistor necessary for the module. After research, we determined that Red and Green LED's consumes 2V and draws 20mA while a White LED consumes 4V and draws 20mA[12].With this information, we were able to calculate the resistance required for the Red/Green and White LEDs (150Ω and 70Ω respectively).These value was calculated using 5V as the input voltage through a 7805 Linear voltage regulator so that we may adjust the LED brightness by changing the resistor and/or regulator.

2.6.3 Voltage Alarm

The Voltage Alarm is a system to protect the battery from over discharging. Lithium Polymer batteries can be damaged when overdischarged. In order to prevent overdischarging the battery we designed a voltage shunt that will trigger when the battery bias is below a set value. We found very similar circuit from the Texas Instruments' data sheet for the TL431 Precision Programmable Reference[14].The shown circuit in Appendix B uses two TL431 chips to turn an LED on when the input voltage is within a threshold. Threshold is determined using four resistors and the following equations from the data sheet:

$$LowLim = (1 + \frac{R_{1B}}{R_{2B}})V_{REF} \tag{8}$$

$$UppLim = (1 + \frac{R_{1A}}{R_{2A}})V_{REF} \tag{9}$$

$V_{REF}$was given in the data sheet as 2.5V and we wanted the upper limit to be 10.6V and the lower limit as 10V. These values comes from taking the maximum voltage of a lipo cell of 4.2V and using 80% of it as a safe value. With these values and the provided equations, we chose the resistor values of 100Ω and 300Ω for R_{2B} , R_{1B} and 100Ω and 330Ω for R_{2A}, R_{1A}

2.6.4 Current Sensor

The current sensor module ensures that the motor will not draw more current than it can handle due to the large load of the impeller. For this part of the circuit, we decided to used an ACS709 Hall effect based current sensor and a voltage divider. The used of the ACS709 was inspired by Hacktronics's arduino current sensor project [13].Even though the circuit was inspired from their project, our design differs significantly since the current we are measuring is massive compared to that of Hacktornic's circuit. We also needed to use a voltage divider circuit to bring down the output voltage to a safe range for our Arduino DUE since the Arduino DUE's I/O pins can only tolerate 3.3V[5]. For Since we knew that that output voltage of ACS709 is typically V_{CC}*0.5[18], we can assume that the output voltage is about 5V since the input is about 10V. Therefore, we can use the voltage divider formula shown below to choose the correct resistances.

$$V_2 = \frac{R_2}{R_1 + R_2} * V_1 \tag{10}$$

Using the voltage divider formula, we found that the following ratio for the resistors

$$R_1 = 0.4R_2 \tag{11}$$

With this in mind, we looked at the commonly available resistor values and chose the 3.9kΩ resistor for R_1 and 5.6kΩ resistor for R_2. This gives an output voltage of 2.95V with an input of 5V which is a safe range for our microconroller.

The circuit schematics as well as the PCB design may be viewed in Appendix B

2.7 Dynamics

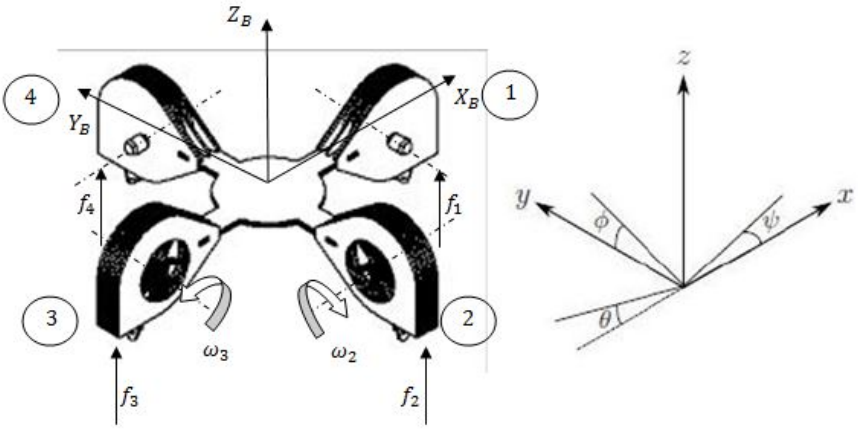


Figure 8: The inertial and body frames of a quacopter

We will start deriving quadcopter dynamics by introducing the two frames in which will operate. The inertial frame is defined by the ground, with gravity pointing in the negative z direction. The body frame is defined by the orientation of the quadcopter, with the rotor axes pointing in the positive z direction and the arms pointing in the x and y directions.

The absolute linear position of the quadcopter is defined in the inertial frame x,y,z axes with ξ. The attitude, i.e. the angular position, is defined in the inertial frame with three Euler angles η. Pitch angle θ determines the rotation of the quadcopter around the y-axis. Roll angle ϕ determines the rotation around the x-axis and yaw angle ψ around the z-axis [19].

$$\eta = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} \xi = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{12}$$

The origin of the body frame is in the center of mass of the quadcopter. In the body frame, the linear velocities are determined by V_B and the angular velocities by ω

$$V_B = \begin{pmatrix} v_{x,B} \\ v_{y,B} \\ v_{z,B} \end{pmatrix} \omega = \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{13}$$

The rotation matrix from the body frame to the inertial frame is[4]:

$$\boldsymbol{R} = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}$$

In which $S_x = sin(x)$ and $C_x = cos(x)$.

The transformation matrix for angular velocities from the inertial frame to the body frame is W_η

$$\omega = W_\eta \dot{\eta},\ \begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} \tag{14}$$

The quadcopter is assumed to have symmetric structure with the four arms aligned with the body x- and y-axes. Thus, the inertia matrix is diagonal matrix I in which $I_{xx} = I_{yy}$

$$I = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix} \tag{15}$$

The angular velocity of rotor i, denoted with Ω_i, creates force f_i in the direction of the z axis. The angular velocity and acceleration of the rotor also create torque τ_{Mi} around the rotor axis.

$$f_i = k\Omega_i^2,\ \tau_{Mi} = b\Omega_i^2 + I_M\dot{\Omega}_i \tag{16}$$

in which the lift constant is k, the drag constant is b and the inertia moment of the rotor is I_M. Usually the effect of Ω_i is considered small and thus it is omitted.

The combined forces of rotors create thrust T in the direction of the body z-axis. Torque τ_B consists of the torques τ_ϕ, τ_θ and τ_ψ in the direction of the corresponding body frame angles.

$$T = k\sum_{i=1}^{4}\Omega_i^2,\ T^B = \begin{pmatrix} 0 \\ 0 \\ T \end{pmatrix} \tag{17}$$

$$\tau_B = \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} lk(\Omega_4^2 - \Omega_2^2) \\ lk(\Omega_3^2 - \Omega_1^2) \\ b(\Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2) \end{pmatrix} \tag{18}$$

in which l is the distance between the rotor and the center of mass of the quadcopter. Thus, the roll movement is acquired by decreasing the 2nd rotor velocity and increasing the 4th rotor velocity. Similarly, the pitch movement is acquired by decreasing the 1st rotor velocity and increasing the 3th rotor velocity. Yaw movement is acquired by increasing the the angular velocities of two opposite rotors and decreasing the velocities of the other two.

This is the case of an ordinary propeller. Due to the design of our fans, yaw movement is achieved by directing the thrust force in order to create a torque in the z axis. Torques matrix will be the next:

$$\tau_B = \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} = \begin{pmatrix} lk(\Omega_4^2 - \Omega_2^2) \\ lk(\Omega_3^2 - \Omega_1^2) \\ lksin\gamma(\Omega_1^2 + \Omega_3^2) \end{pmatrix} \tag{19}$$

where γ is the angle between f_i and the vertical axis. For our system we have neglected de the torque produced by $b\Omega_i^2$ because it is much smaller than the torque produced by $lk\Omega_i^2$

The equations of motion are:

$$m\ddot{x} = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} + RT_B \tag{20}$$

$$\tau = I\dot{w} + w \times (Iw) \tag{21}$$

2.8 State Equations

The state vector is:

$$x(t) = (x, y, z, u, v, w, \phi, \theta, \psi, p, q, r)^T \tag{22}$$

Our system have eight control's inputs:

$$u(t) = (\Omega_1, \Omega_2, \Omega_3, \Omega_4.\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T \tag{23}$$

And the output vector:

$$y(t) = (u, v, w, \phi, \theta, \psi)^T \tag{24}$$

Finally, I have considered that angular accelerations are inputs that have not relationship with the speeds and for this reason can be put out of the model.

2.9 SISO system

After all the study of state equations, state linearization, linearization around the equilibrium point, Laplace transform (MIMO system) and decoupling the final SISO system is [20]:

$$\begin{pmatrix} u(s) \\ v(s) \\ w(s) \\ \phi(s) \\ \theta(s) \\ \psi(s) \end{pmatrix} = \frac{1}{s^2} \begin{pmatrix} 0 & 0 & \frac{gk_{MF}}{J_{Eyy}s^3} & 0 \\ 0 & -\frac{gk_{MF}}{J_{Exx}s^3} & 0 & 0 \\ \frac{b}{ms} & 0 & 0 & 0 \\ 0 & \frac{k_{MF}}{J_{Exx}} & 0 & 0 \\ 0 & 0 & \frac{k_{MF}}{J_{Eyy}} & 0 \\ 0 & 0 & 0 & \frac{k_T}{J_{Ezz}} \end{pmatrix} \begin{pmatrix} U_1(s) \\ U_2(s) \\ U_3(s) \\ U_4(s) \end{pmatrix} \tag{25}$$

where $U_1(s)$, $U_2(s)$, $U_3(s)$, $U_4(s)$ are:
Vertical Thrust

$$U_1(s) = \Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2 \tag{26}$$

Roll:

$$U_2(s) = \Omega_4^2 - \Omega_2^2 \tag{27}$$

Pitch:

$$U_3(s) = \Omega_3^2 - \Omega_1^2 \tag{28}$$

Yaw:

$$U_4(s) = \Omega_1^2 + \Omega_3^2 - \Omega_2^2 - \Omega_4^2 \tag{29}$$

Thanks to the axis of movement chosen, the system can be decoupled. Roll and Pitch depends just of two motors while when Yaw is active Pitch is also affected due to the position of the servomotors.

2.10 Control

The control schematic for the pitch can be modeled as the next one:

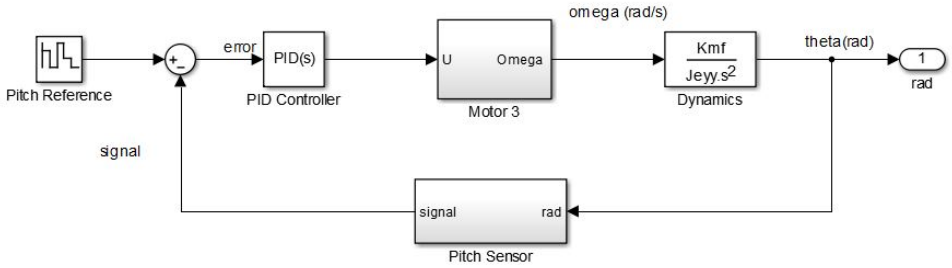


Figure 9: Pitch Control

After more research I have realized that most of the current drone's control are not design taking into account all the physical parameters like inertia, etc. Knowing the exact values of all the physical parameters will make possible to design an accurate control choosing the proper values of the PID's parameters. Due to the fact that this is not possible, we will design a control without paying much attention to the parameters which will be tunned later by doing stabilization tests.

Control compute and update

To compute the PID values we have use the Arduino library $PID_v1.h$:

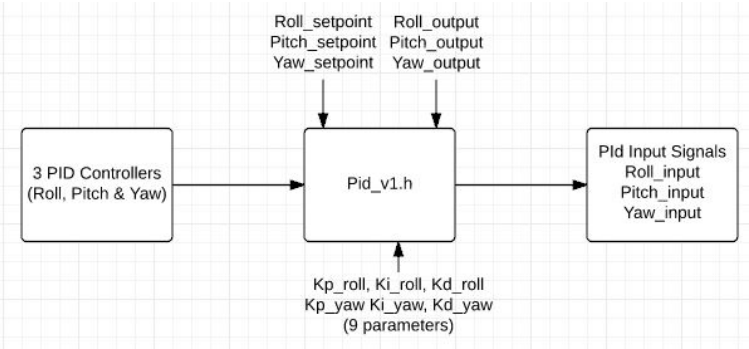


Figure 10: Control block

This library needs the three setpoint values obtained from the stick response of the pilot, the three current angles of the drone (pitch, roll and yaw) from the sensor readings and the nine parameters of the three PIDs (Kp, Ki and Kd of every axis of movement). In real there are just 6 different parameters because roll and

pitch have the same numbers. The implementation of this code can be found in the appendix G.

The input signals generated by the control block are the width of the pulses that must be added or subtracted to the throttle pulse of every motor. The range of our pulse modulation is 1500-2000μs(minimum and maximum width), for this reason establish a maximum control output(input signal of the block diagram, U) of $\pm 200\mu s$ should be enough to stabilize the drone.

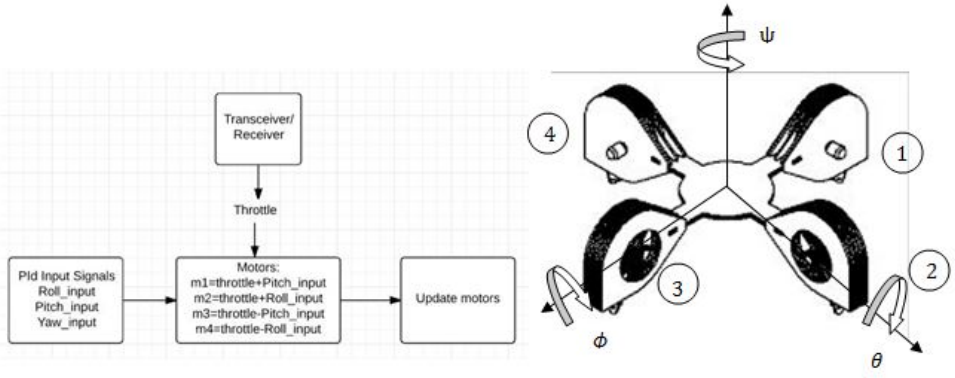


Figure 11: updating control on motors

Fans 1 and 3 controls pitch while 2 and 4 controls roll. Yaw's control is not needed for drone stabilization and for this reason has not been included in the control. The control updating can be found on appendix G figure 30.

2.11 PID controller

The PID controller of each axis is the key to have a stable flight experience. Three separate constant parameters for each angular control (roll and pitch shares parameters), K_P depends on the present error, K_I depends on the accumulation of past errors and K_D is a prediction of future errors.

K_P is the main value to worry about, the stability of the system depends on this parameter. The higher the P is, the fastest the system will respond. If P is too high overshoots will be big until the system becomes unstable.

The effect of K_I is keeping the last stick position. It's integrating the error over time, the longer the error persist, the greater the force is applied. This parameter set the steady state error to zero.

K_D changes the forced applied to correct position error, when it sees a decrease or increase in the position error. So you will see a softer movement in the quadcopter when D is used.

PID tuning process

For the purpose of adjust the PID parameters, we have to fix one axis of movement for testing, e.g roll, so only one axis of rotation is active at time. During the this test roll, pitch and yaw are set to 0 (setpoint=0) from the stick position in the receiver. The only signal different to zero is throttle. The idea is trying to tilt the drone by hand, motors must counteract this effect producing enough torque. A proper set of parameters would make very hard to move the drone by hand over time.

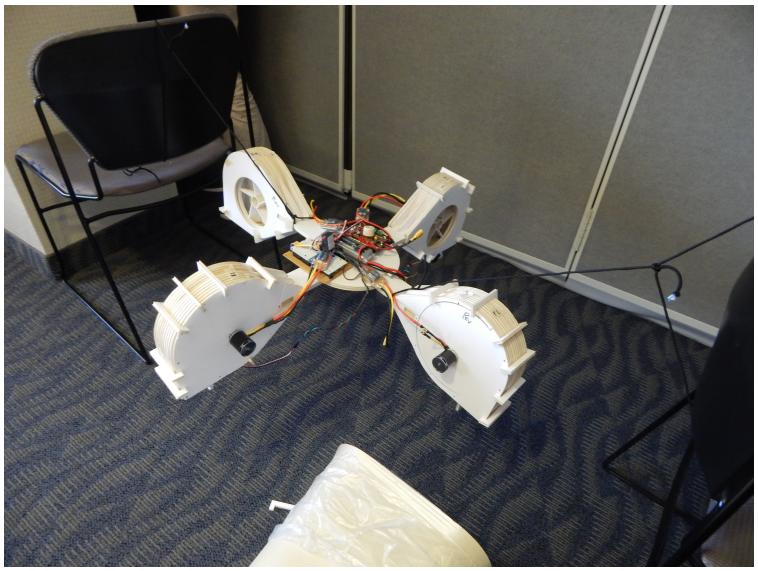


Figure 12: Setup, stabilization test

Let's describe the steps to adjust PID parameters[21]:

1. Set Yaw PID parameters to zero and roll/pitch I-D gain to zero. Let's increase P-gain.
    1.1. If P-gain is too low is easy to turn by hand (slow and imprecise to stick responses when a different to zero setpoint is set.).
    1.2. Increase P-gain until oscillations appears. This means that the system has become unstable. The proper P-gain must be below this ultimate P.
    1.3. A good P-gain must assure that it is hard to turn by hand and sharp stick response.
2. Now let's tune I-gain:
    2.1. A low I-gain makes easy to tilt the drone by hand over long time.
    2.2. When I-gain is too high the drone oscillates with a lower frequency than with a too high-P gain.
    2.3. A good I-gain makes very hard to move the drone by hand over long time. It stays in position when stick is released (in case you set a setpoint different to zero).
3. Finally, let's tune D-gain:
    3.1. For having a soft flight experience, increase D-gain slowly until shaking finishes.

2.12 Cost Analysis

Table 1: Parts Costs

Part	Manufacturer	Model	Quantity	Cost/unit ($)
Brushless Motor	NTM	NTM Prop Drive 450	4	22.99
Battery	Turnigy	Bolt 3S	1	29.08
Microcontroller	Arudino	Due	1	45.00
Gyro + Accelerometer	Adafruit	Breakout Board	1	6.88
Transceiver	Spektrum	Dx6i/AR600	1	120
Actuator	Sparkfun	Generic Sub-micro	2	10.00
ESC	Hobby King	Hobby King 50A ESC	4	20.00
Chassis Frame	-	-	1	92.69
PCB Parts	Panasonic	-	1	36.54
Custom Shield	-	-	1	2.16
Misc Parts	-	-	-	5.00
Total				$ 517.15

Table 2: Labor Costs

Personnel	Salary per Hour ($)	Hours Worked	Total = Salary per Hour * 2.5 * Hours Worked ($)
Bree Peng	37.5	180	16,875
Ignacio Aguirre Panadero	37.5	180	16,875
Leo Yamamae	37.5	180	16,875
Total			**$50,625**

Table 3: Grand Total Cost

Item	Cost ($)
1 Drone Part Cost	517.15
Labor Cost	$50,625.00
Grand Total	**$51,142.15**

3 Ethical / Legal Considerations

3.1 IEEE Code of Ethics Considerations

Since this drone will be airborne, this will involve the first code of the IEEE Code of Ethics where we must consider the decisions that we make which will involve the safety, health, and welfare of the public [2]. This is because even if the drone is safer than those that uses propellers, it may still fall from the sky and cause injury or property damage, cause injury or property damage from flight motions, catch fire if overheated, or catch fire if the battery is punctured. This is a concern that regarding the clause 9 in the IEEE code of ethics to avoid injuring others, their property, reputation, or employment by false or malicious action. Because since the drone will be capable of carrying and delivering a payload, we must deter its usage for malicious intent through its specifications.

3.2 Legal Obligations

A legal consideration that we must consider is the FAA regulations on Model Aircraft Operations. The FAA recently released the guidelines on flying for hobby or recreation, which restricts recreational model flight to below 400 feet, away from other people or stadiums, et cetera. The regulation that most applies to this drone states that one may not fly within 5 miles of an airport without clearance from the airport and control tower before flying [3]. This applies to our drone because the University of Illinois at Urbana-Champaign is within the 5-mile radius of the University of Illinois Urbana-Champaign Willard Airport. This means that before any testing is being done on the drone, we must contact the control tower at the Willard Airport for clearance. In addition, the University of Illinois has also placed regulations and restrictions on drone use on campus. Another legal issue that concerns our project is the registration of our drone. FAA has made it mandatory for anyone who has a unmanned aircraft that is heavier than 250g but lighter than 25kg to register their drone on their website to fly outdoors[3]. Our drone will weigh close to 1kg which falls into this category.

However, to be able to test the system without having to go out to an airfield every time, we will have the test apparatus as shown in Appendix E. By doing this, the project will not be technically a drone and should be safe enough to be tested from yaw, pitch, roll, stability even indoors.

3.3 Safety Statement

In this project, there are many safety considerations that must be taken into mind. This is because the project is a drone which upon completion will be airborne. During the build phase, there are many step that pertains safety concerns.

The fact that the project is a drone is one safety concern that we have to address according to the first and second IEEE code of ethics. The first statement is to to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.[2] This is a factor we need to consider because even though our drone will be safer than a drone that uses propellers, the drone could still damage unsuspecting people or property when it falls from a height.

Also the drone will be powered by a Lithium-Polymer battery which by itself is a safety concern. This is because a Li-Po has many factors that could cause it to explode releasing harmful chemicals. Some events that could cause the battery to explode include but is not limited to charging using an incorrect battery charger, shorting wires, puncturing cells, and overcharging[4]. This is a great concern as fires indoors is not a thing to be taken lightly especially when there are harsh chemicals released.

The second statement in the IEEE code of ethics needs to be considered because as we are flying a motorized object within 5-miles of an airport, we must let the airport control tower know that we will be flying a drone[3]. Even though from past experience, the control tower does not mind the drone being flown at our altitude, there are major concerns of drones interrupting plane and helicopter flight.

Another concern with our project is the safety of the new propulsion method. If the impeller shatters on impact and creates more harm than a normal propeller, the project has failed. From testing conducted, we saw that the Impeller shreds into a confetti or safety glass shards after exploding in a safe direction. From this, we were able to see that the drone propulsion method is safer and a typical propeller as nothing was cut when the impeller shattered.

4 Conclusion

In this project we were able to design and construct a drone that is capable of flight without the use of propellers. Our design used a centrifugal fan composed of Polystyrene which produced a peak thrust of 550g and was capable of withstanding falls without breaking. Also, when the fan failed, the impeller shattered like safety glass and posed no dangers unlike shattered plastic propellers. Using an Arduino microcontroller as the controller, we were able to interpret sensor data from the IMU and attempted to determine stability. However, due to power supply limitations and parameter change times, we were unable to calibrate the system to achieve stability during flight. Another issue we encountered was the PCB trances not being large enough withstand the current required by the motors.

If we were to continue this project, we would optimize the fan design to output greater thrust with a smaller overall design as well as explore other materials such as composites. We will also redesign our power distribution board to be formed in layers to withstand the high current of the system. Either the motors selected must draw less current or we must get a power supply with a greater capacity for sustained flight. With a longer flight time and the right apparatus, we will be able to improve upon control and achieve stability in flight.

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

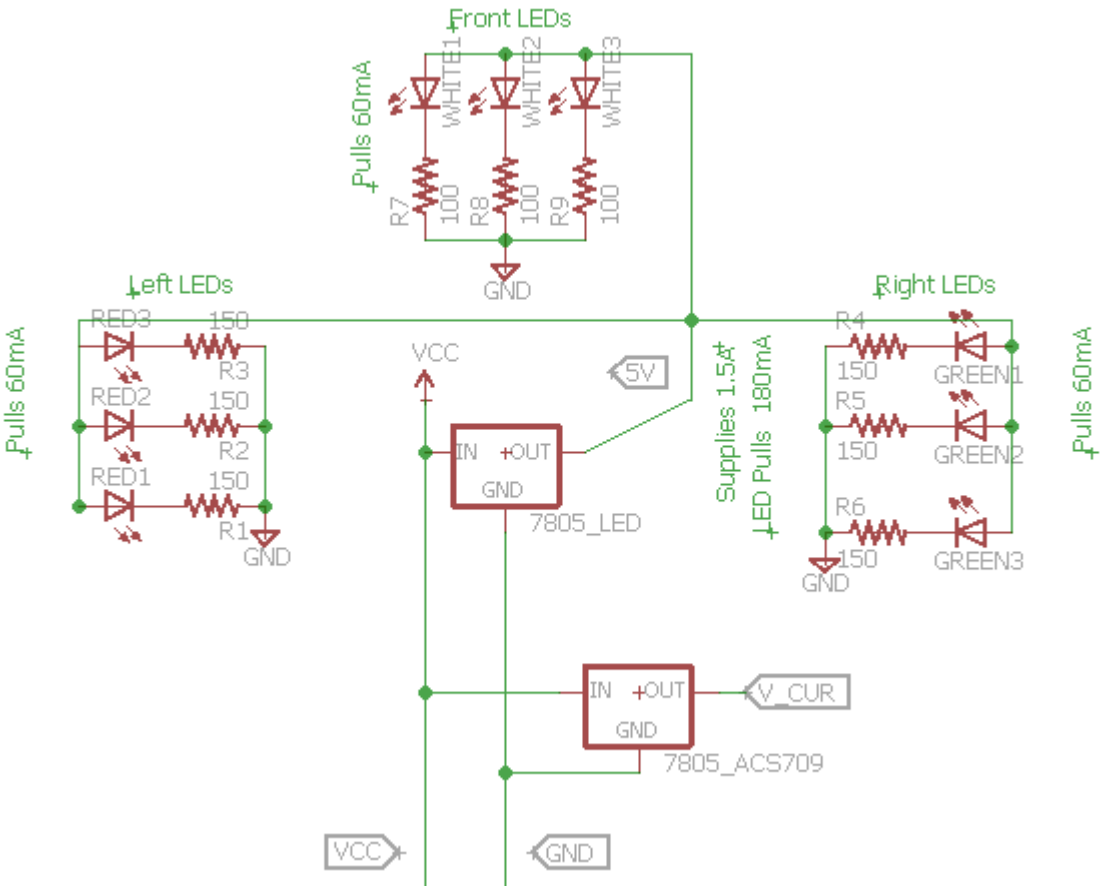


Figure 13: LED module to display orientation of drone

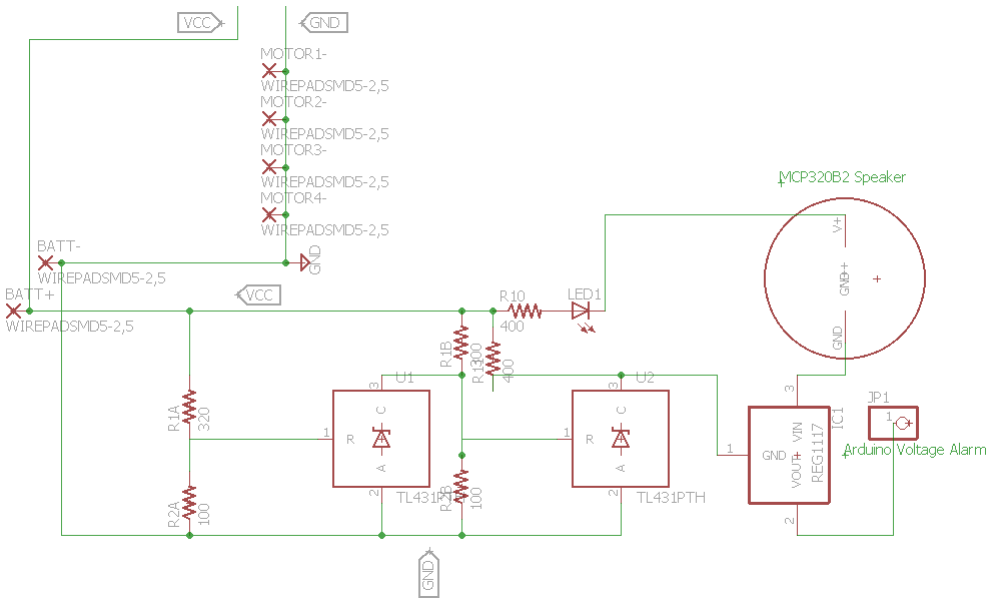


Figure 14: Voltage Alarm module and Mounting pads for motors

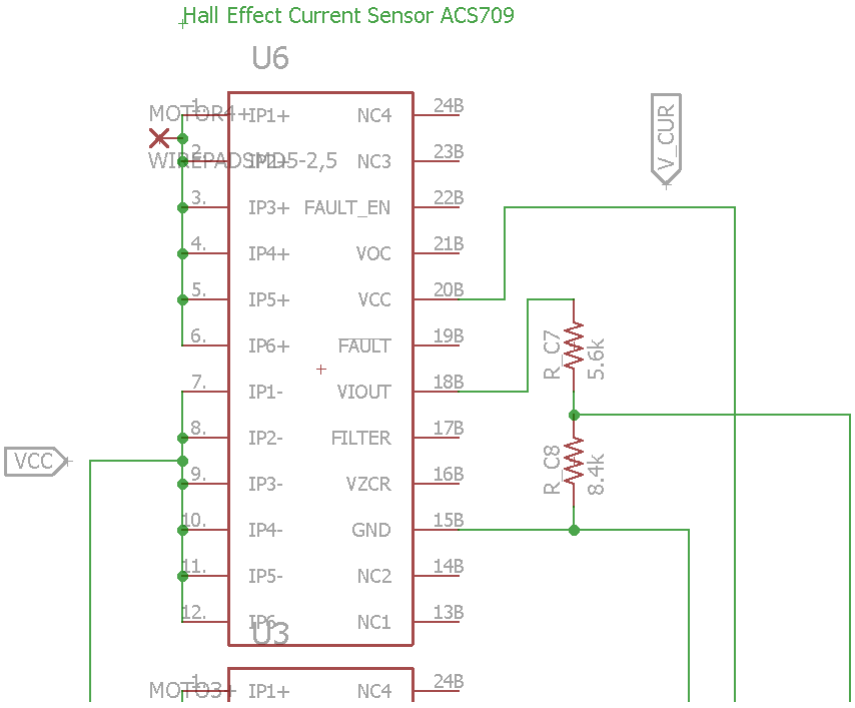


Figure 15: One of the four ACS709 Chips used for sensing current

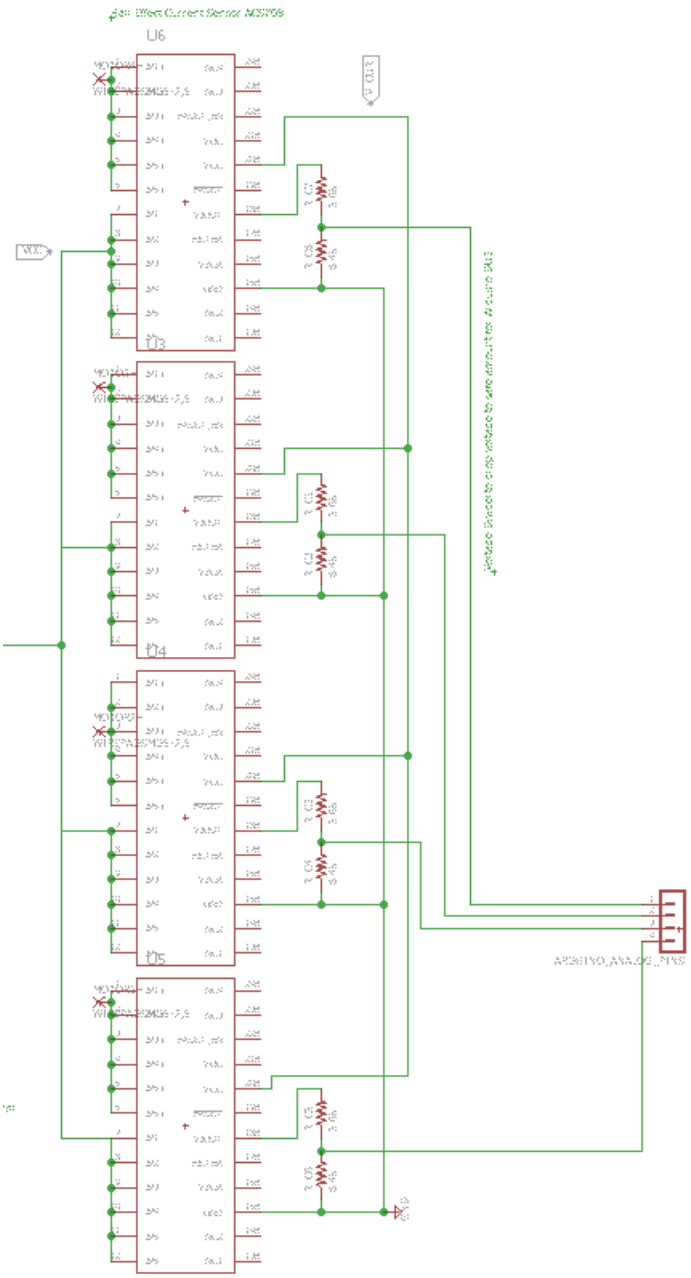


Figure 16: All four of the ACS709 Chips used for sensing current

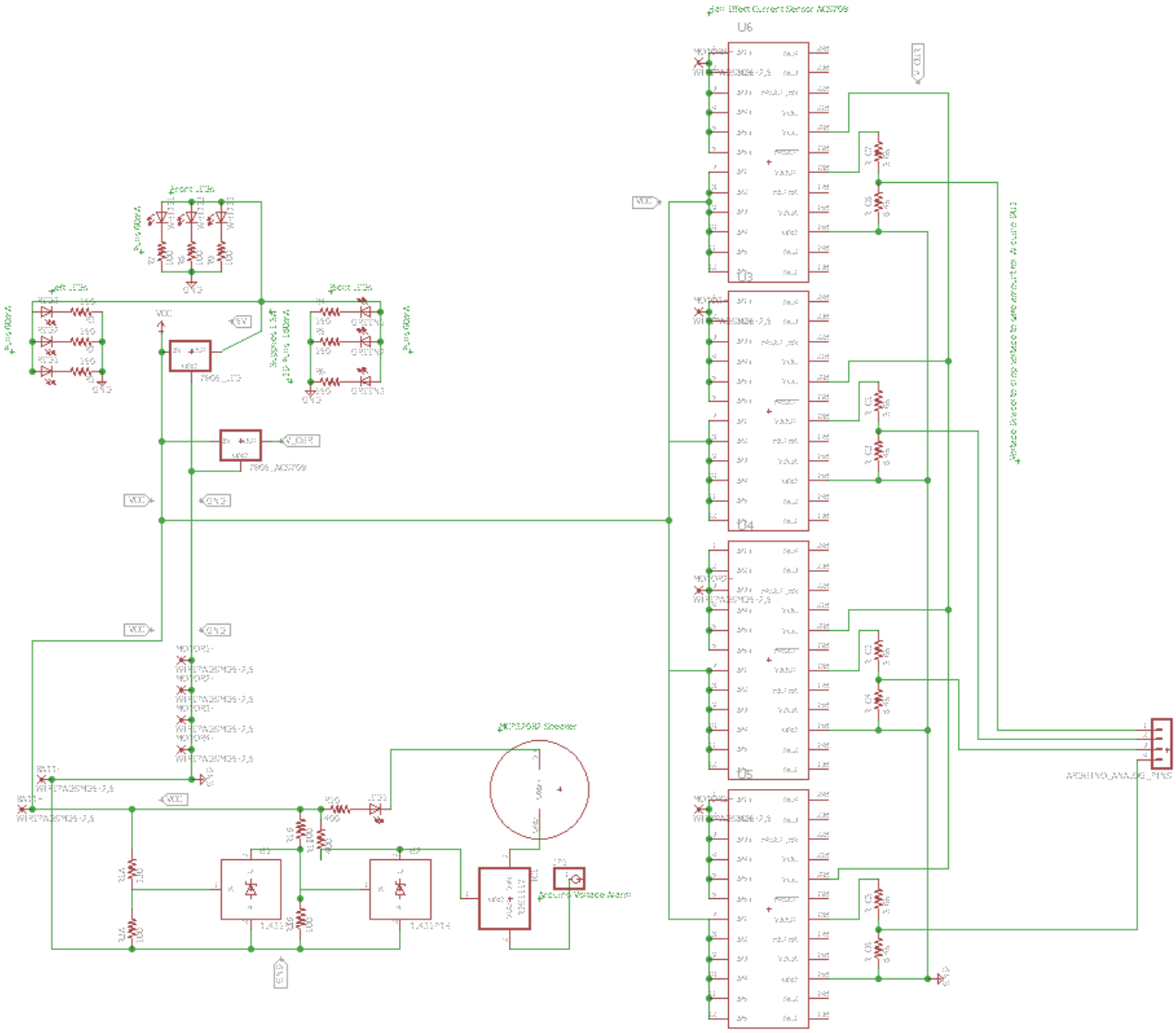


Figure 17: Complete Circuit

B.1 PCB Design

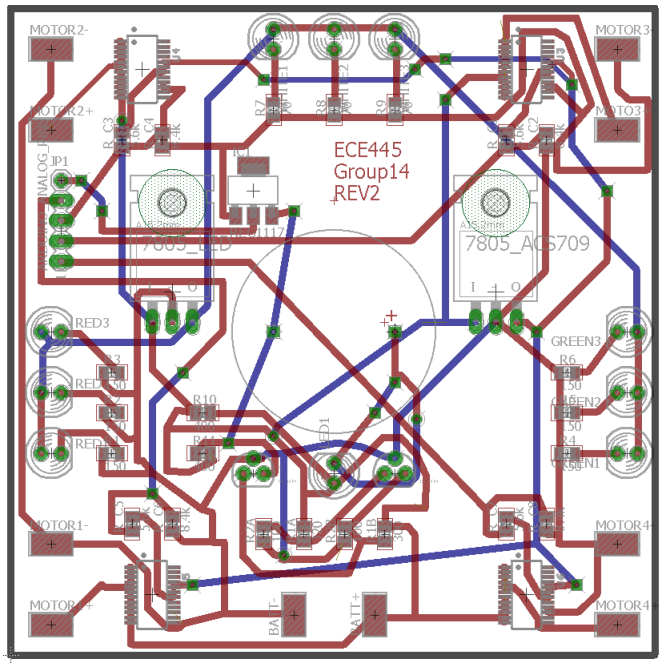


Figure 18: PCB design from EAGLE CAD

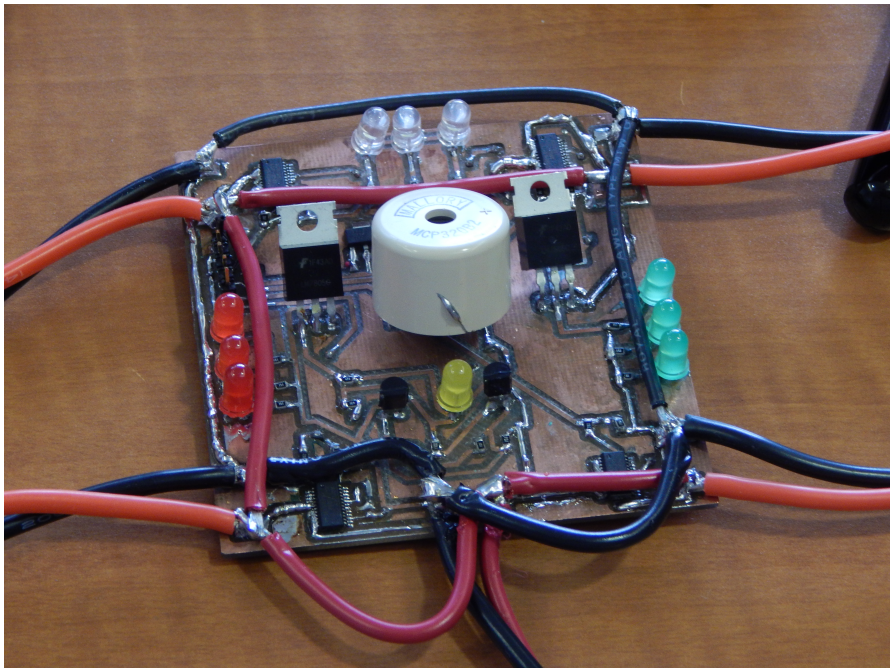


Figure 19: Completed PCB

Appendix C Software Flowchart

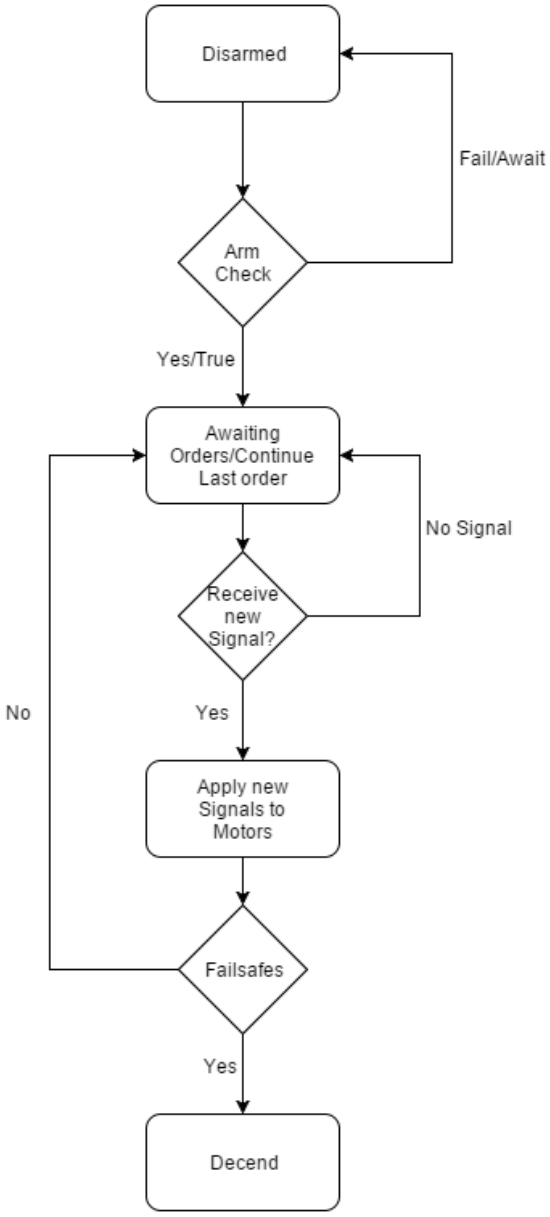


Figure 20: Flowchart for drone

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

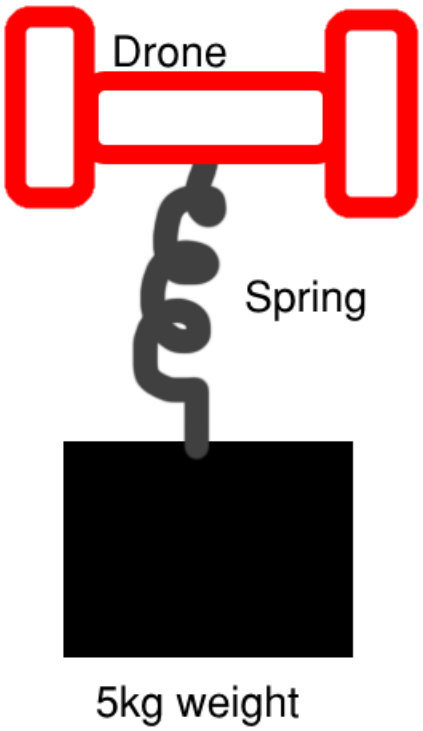


Figure 21: Testing apparatus for on-campus testing

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

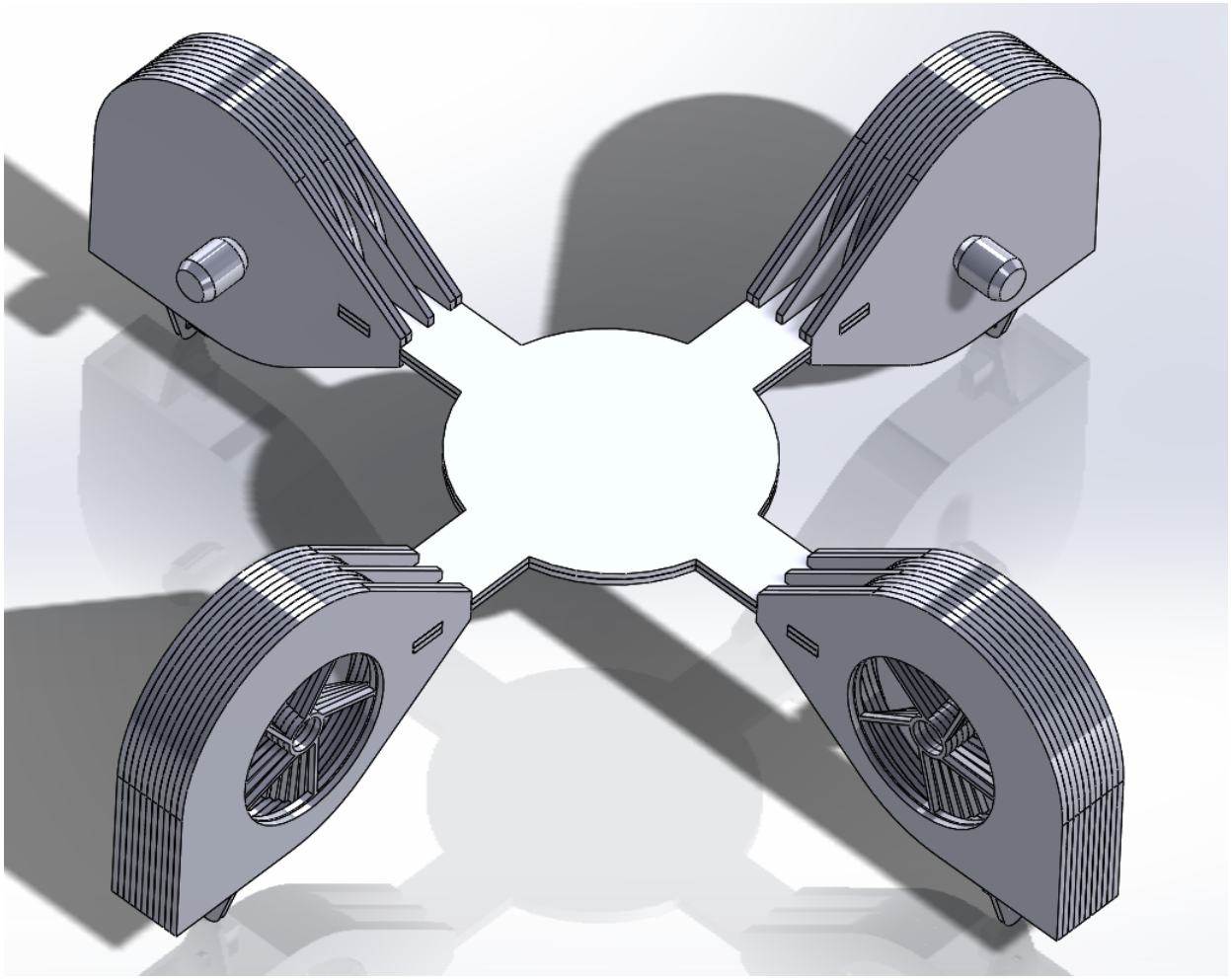


Figure 27: Isometric view of Drone CAD

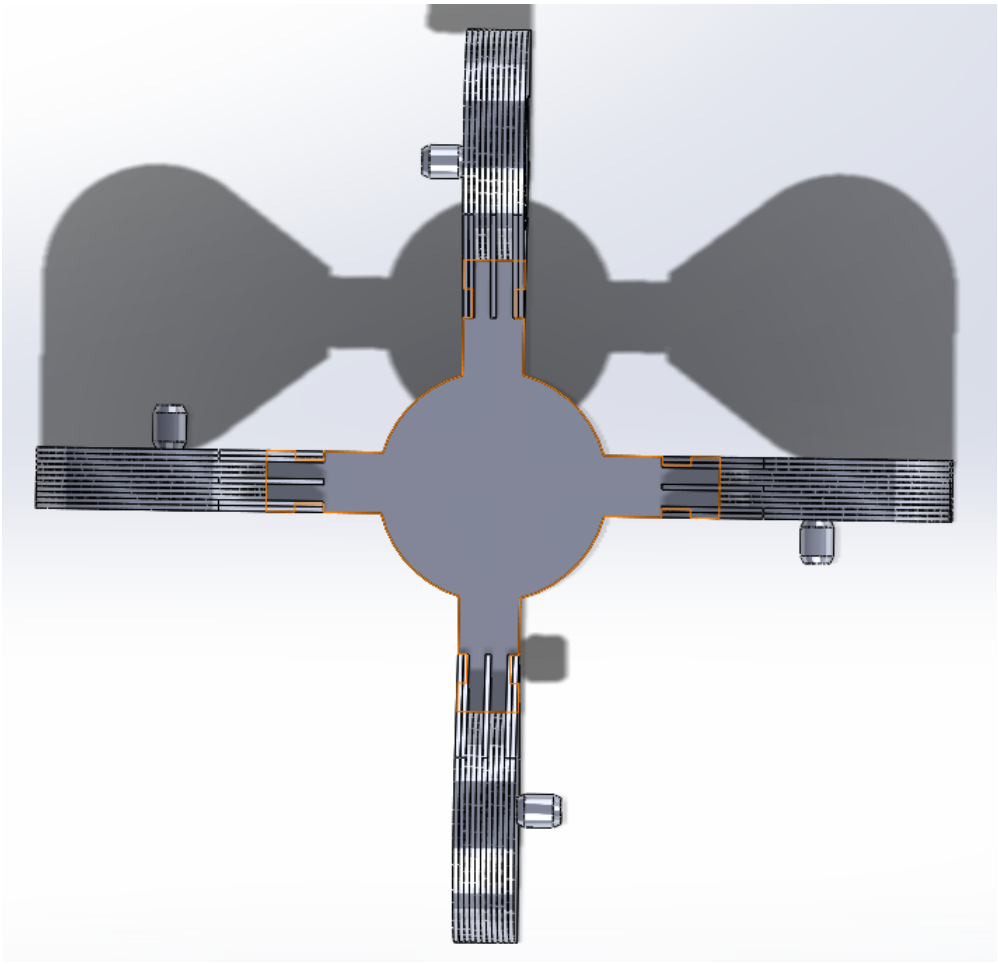


Figure 28: Top View of Drone CAD

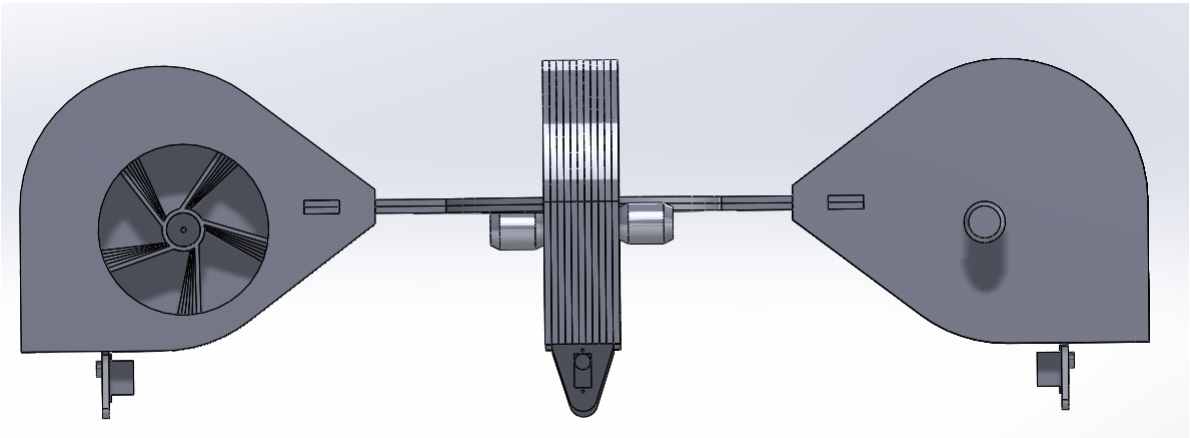


Figure 29: Side View of Drone CAD

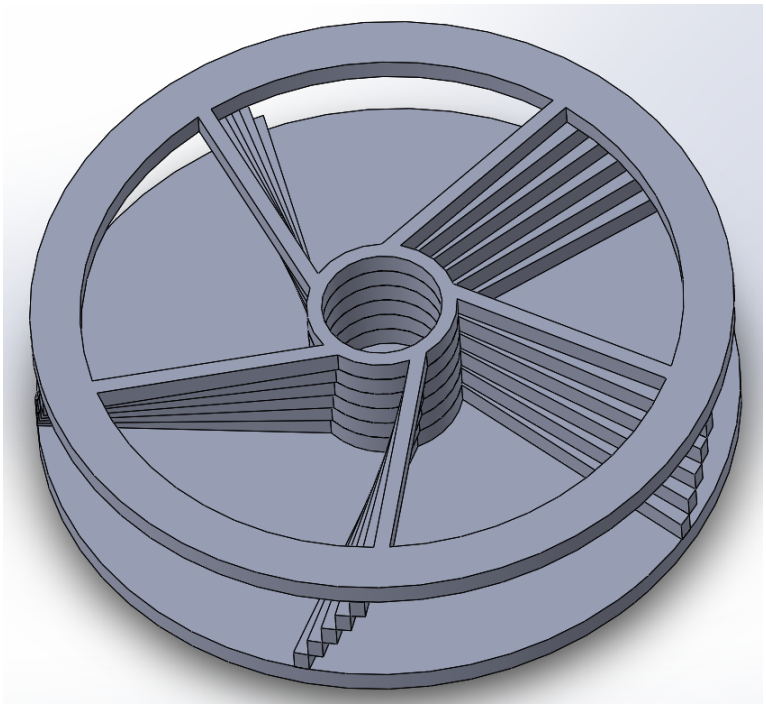


Figure 30: Impeller for Counter Clockwise Fan

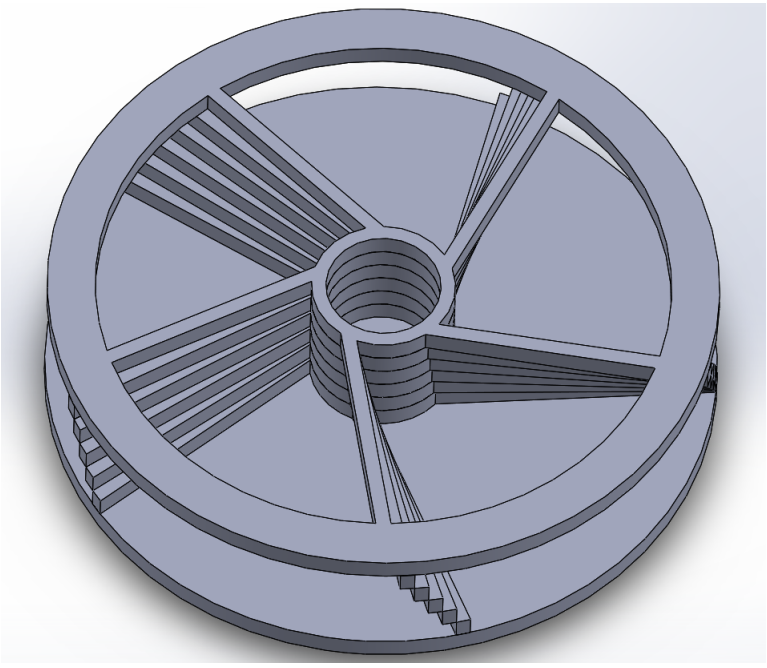


Figure 31: Impeller for Clockwise Fan

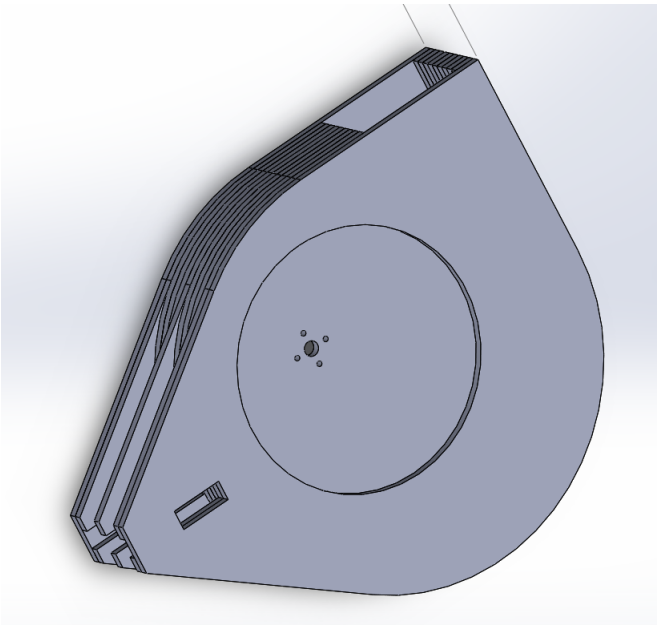


Figure 32: Turbine Housing for Counter Clockwise Fan

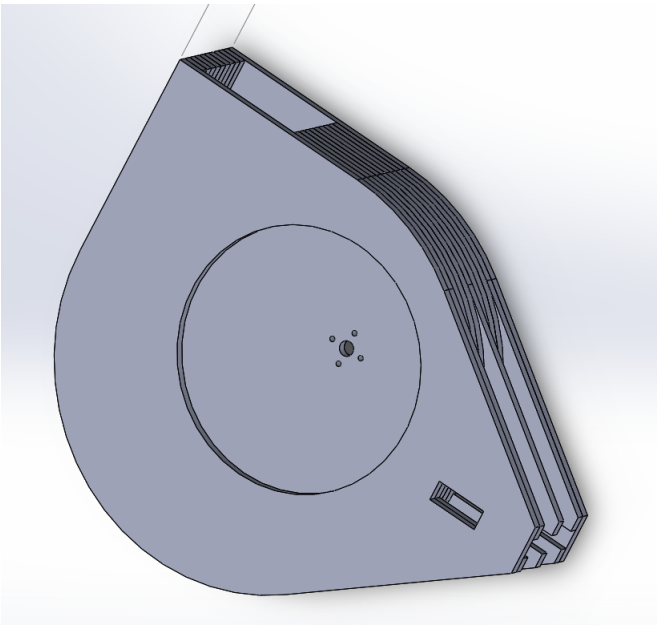


Figure 33: Turbine Housing for Clockwise Fan

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].