Alex Abulnaga

Miker Qiu

Moshe Lawlor

Seth Waugh

Executive Summary:. This report provides a detailed reference on the creation a simplified helicopter control system. Included are detailed schematics, documentation describing operation of components, and pictures of the final implemented design. The components described in the report are a rotating mechanism, a custom motor, a control system, and power electronics circuits. The engineering process is also documented through verbal descriptions and calculations included in our appendices.

Overall the final design was able to succeed requirements, and was able to complete a more extensive landing and takeoff sequence while still remaining stable.

Table of Contents

List of Tables

List of Figures

INTRODUCTION

Helicopter control design has to be implemented in a way that must account for six degrees of freedom. Each degree of freedom represents a different direction in which a helicopter may travel, and increased degrees of freedom lead to more complex designs. Our project is aimed at designing a helicopter control system in which a helicopter is constrained to moving radially, and vertically. Using only two degrees of freedom greatly simplifies the problem of control and the problem becomes as simple as controlling the speed, and hence the lift force generated by each motor. By controlling two different motors, one commercial, and one self-built motor, we can control the radial, and vertical position of our motor to demonstrate a takeoff and landing sequence.

Problem Description

Control System:

Requirements:

System:

1. 2 Degrees of Freedom, Lift and Yaw
2. System must be able to reach a setpoint in both yaw and lift
3. Controls must be integrated with custom motor

Component:

1. Utilizes a Microcontroller with PID implementation
2. Circuit must be Implemented on a PCB

Constraints:

System:

1. Utilizes motors to achieve lift and yaw movement
2. Fits within 2ft cube size

Component:

1. External sensors integrated into PID

Goals*:*

System:

1. Follows set path as accurately as possible
2. Be able to change setpoints during operation with minimal error
3. Lift and hover, then rotate 180 degrees and lower

Component:

1. High resilience to external disturbances using the PID controller
2. Extremely accurate sensors (noise filtration, etc)

There are several requirements for our system. Most importantly, our system must be able to achieve accurate control in both the lift and yaw directions utilizing a microcontroller with PID controls. The system must also be fully integrated with a custom motor in order to achieve lift. These requirements must be met while keeping the design within a 2ft cube size (including propeller reach) while also connecting the external sensors and PID controller using a custom PCB. If all of these requirements and constraints are met, the robot will be able to follow our variable set path as accurately as possible while remaining resilient to external disturbances.

Motor :

Requirements:

System:

1. At least 3 poles in the stator windings
2. Each pole should generate equal and strong force on rotor to spin at constant rate
3. Use power electronics in switching current phases and driving current
4. Control motor speed with PWM

Component:

1. Use permanent magnet rotor
2. Power electronic circuit must be implemented on a PCB

3. Use sensors to detect rotation position of rotor
4. Able to modify PWM output

Constraints:

System:

1. Needed to be small in order to be mounted on the mechanical arm
2. Needed to be light so it can lift up the arm with relatively low power input
3. Needed to be able to spin with limited power input (i.e, driven by some current drivers)

Component:

1. Magnetic copper wires (26 AWG) are rated with maximum current 2A
2. Temperature must be kept under 100°C (some 3D print parts might deform)

Goals*:*

System:

1. The motor could be limited within 500g of weight
2. The motor can generate at least 1N of thrusting force and lift itself upward
3. Efficient power consumption

Component:

1. Perform consistently and robustly
2. Able to withstand strong external and internal impact force

There are several requirements to fulfill in order to spin the Electronically Commutated DC Motor. By calculation, we were expecting the stator to generate relatively high torque force on the rotor, and thus a strong magnetic forces is required. Currently we have 200 windings on each pole of the 6-poles stator to ensure the force generated can meet the requirement. By using Arduino, we can also interpret the readings of Hall sensors and activate the related channels of the H-bridge, and thus we can control the force generating from each pole. Currently, all requirements had been fulfilled as we had demo in the lab.

System Level Design

On a basic level it is fairly easy to understand the operation of the entire system. Included below is a block diagram of our system, and also a picture of the implemented system. (figure 1, figure 2)

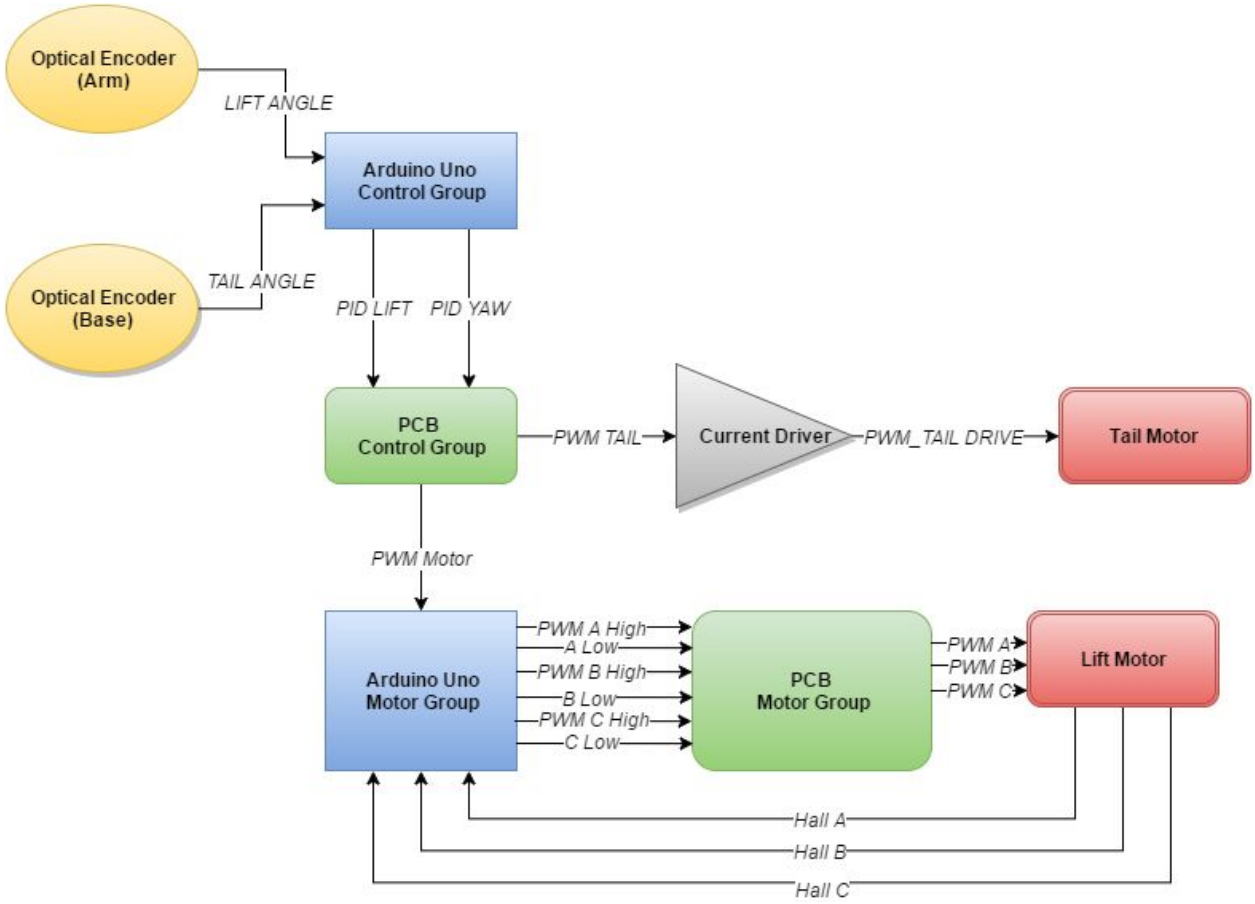


figure 1. Simplified System Block Diagram

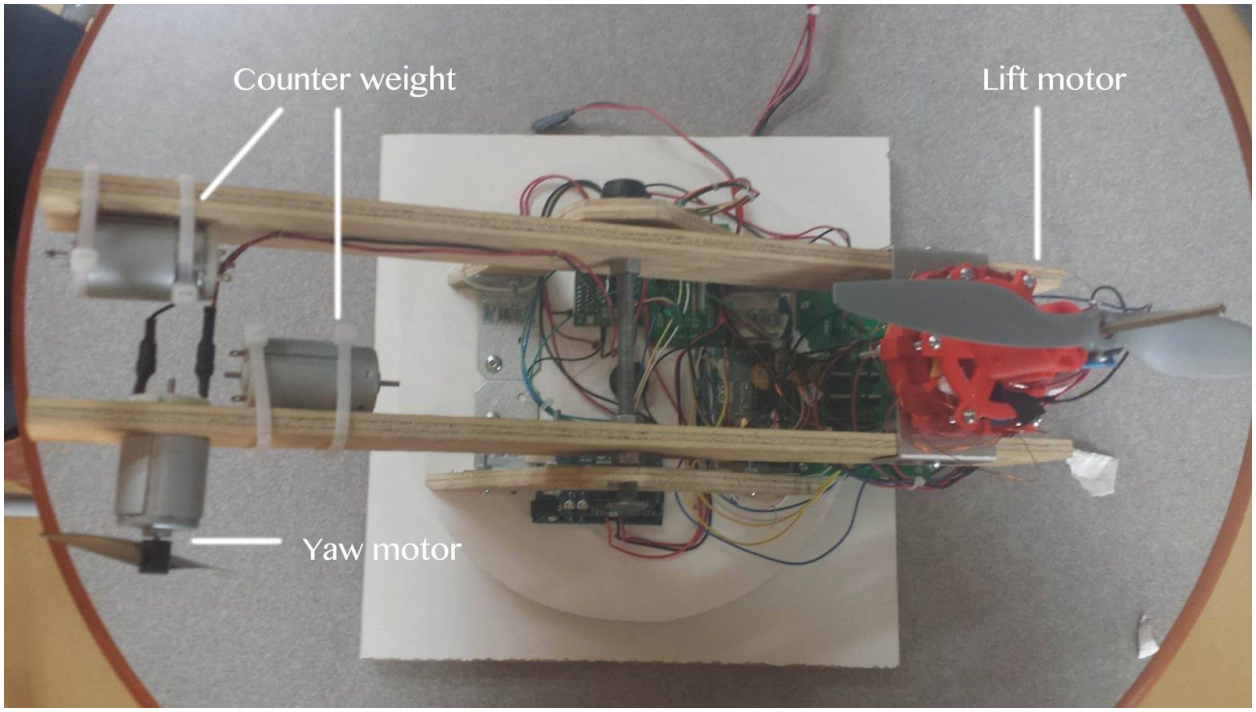


figure 2. Implemented System

The system in our block diagram (figure 1) mirrors the implementation of our system (figure 2). You are able to see all the functional blocks in each including the tail motor, lift motor, the Arduino Uno microcontrollers, the current driver, and the PCBs. The system operates based on feedback from two optical encoders. The first optical encoder is located on the base of the mechanism (figure 2). The second optical encoder is located on one of the legs of our mechanism which can be seen in both figure 2 and in figure 3.

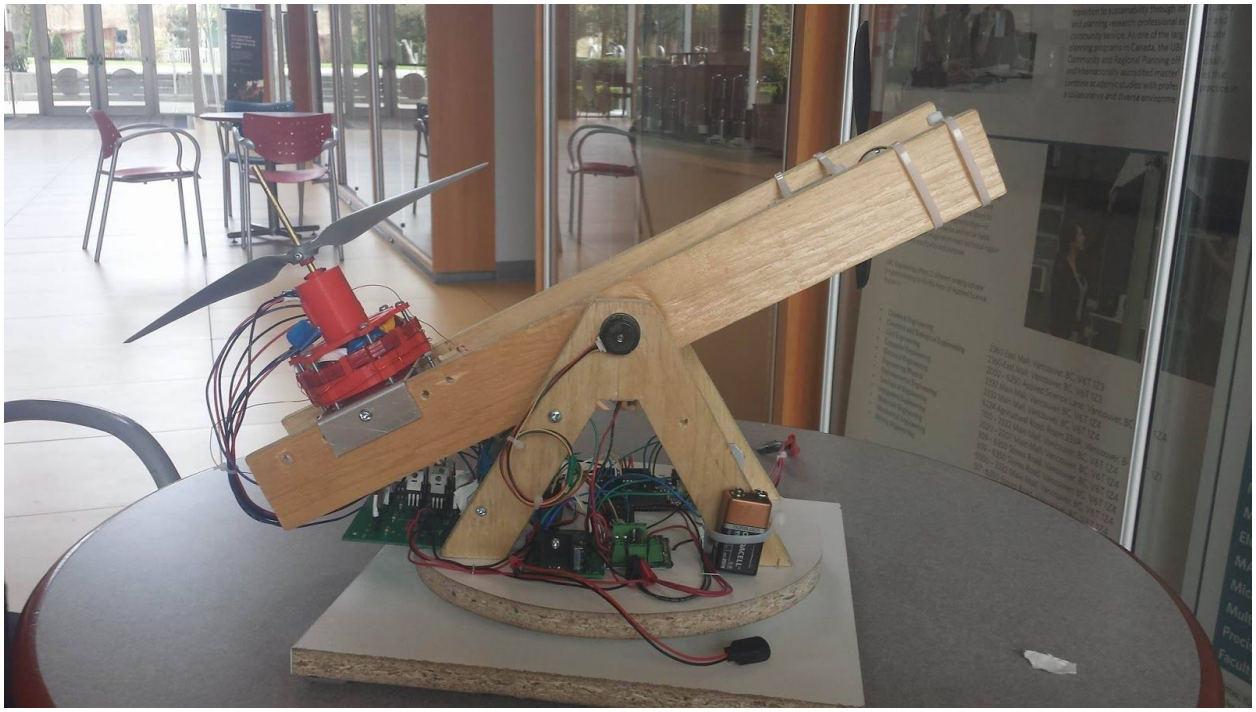


figure 3. Mechanism Side View

The first optical encoder detects the rotation angle of the entire system. This determines the position of the lift motor relative to its starting point. The second optical encoder senses the angle of the arm mechanism relative to its starting position. As the motor lifts, the angle will increase. The two inputs from the optical encoders are fed into the microcontroller programmed by the Controls group. This microcontroller is acting as the PID controller. This microcontroller outputs two PWM values to the Controls PCB. The PCB simply acts as a low pass filter for these two signals and then outputs one value to the motor's microcontroller, and the second value to the current driver. The current driver then drives the tail motor according to this PWM signal. The second PWM signal is fed to the microcontroller programmed by the Motor's group. The Motor's Microcontroller independently controls the lift motor by outputting six signals according

to feedback from the hall sensors. The Motor's Microcontroller only alters the duty cycle of the PWM according to the input from the Controls group.

The simplified communication between the two microcontrollers should allow for smooth integration of both the motor and control systems. This is because the control's system microcontroller only adjusts the duty cycle. Both the motor and control group were able to complete and verify designs in phase 2. For integration we merely need to ensure that we can transmit and receive a single value via a direct connection.

High Level Design

Control System:

The main components of the control system are a microcontroller, two sensors for lift and yaw, a PWM current driver, and two motors. The microcontroller we are using is an arduino uno which we coded to implement a PID controller in the system. The sensors that we used are both 100 counts per rotation optical encoders which are integrated into the microcontroller. The current driver is an off the shelf, 5A rated Sabertooth, and the motors are both 12V commercial motors. The communication channels between all the components can be seen in Figure 1 below.

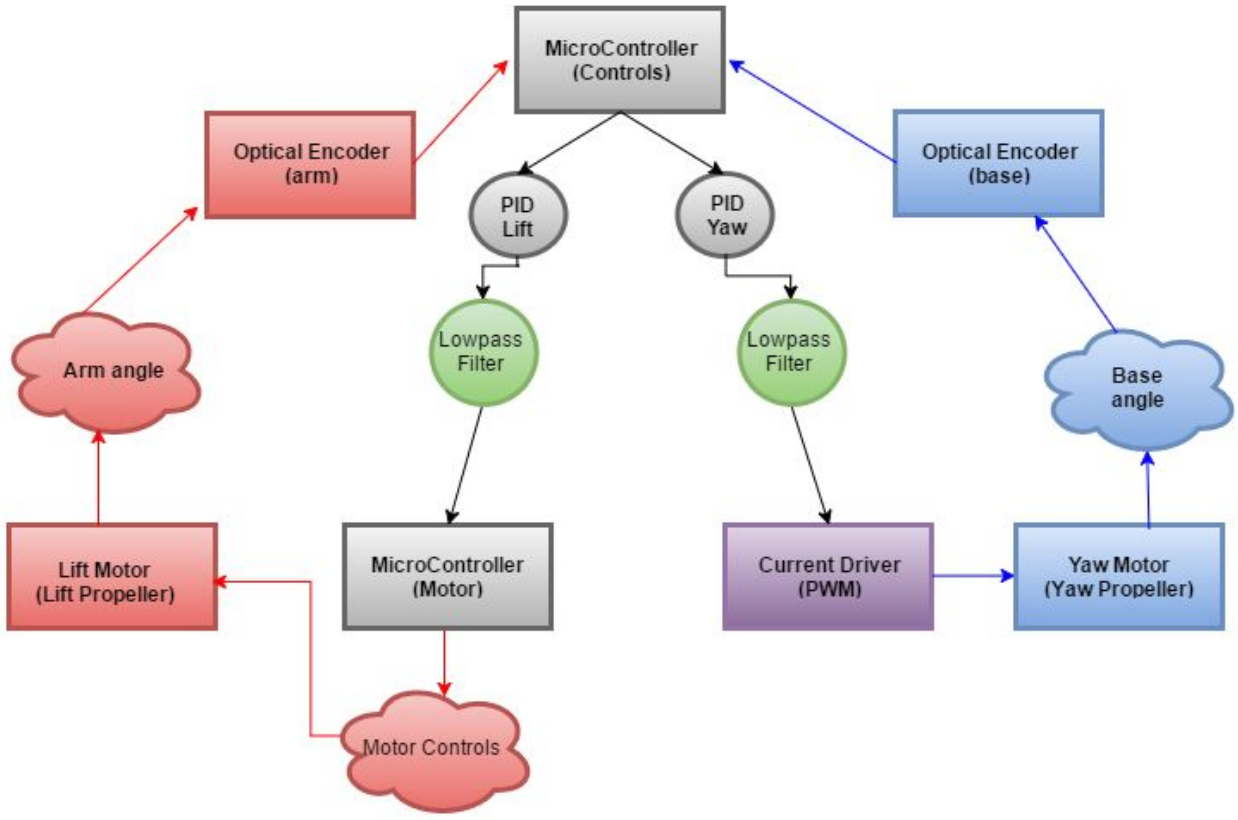


Figure 4: Control System Communication Loop

Communication begins with the actual lift and yaw angles being inputted to the microcontroller through the optical encoders located on the arm and base of the robot respectively. With these values, the microcontroller performs PID calculations in order to output a serial number. This serial number is then passed to the current driver. The serial number dictates to the motor driver a duty cycle to output to the motors. For example, passing the serial number 127 to the driver outputs maximal voltage to the lift motor. Depending on the serial number sent, the driver modulates the output to both motors. The motors utilize the power outputted from the driver in order to spin propellers attached to their shafts in order to generate thrust, which then alters the lift and rotational angles of the system, thus closing the loop.

Motor :

There are five main parts at work within the motor design. These are illustrated in a simplified Figure 5 below as a microcontroller, a driver circuit, a three phase h-bridge, a hall sensor circuit, and our motor itself. Inputs are illustrated using arrows that point into a block and outputs are pointing outwards from a block. Numbers indicated on each singular wire also indicate the amount of inputs or outputs.

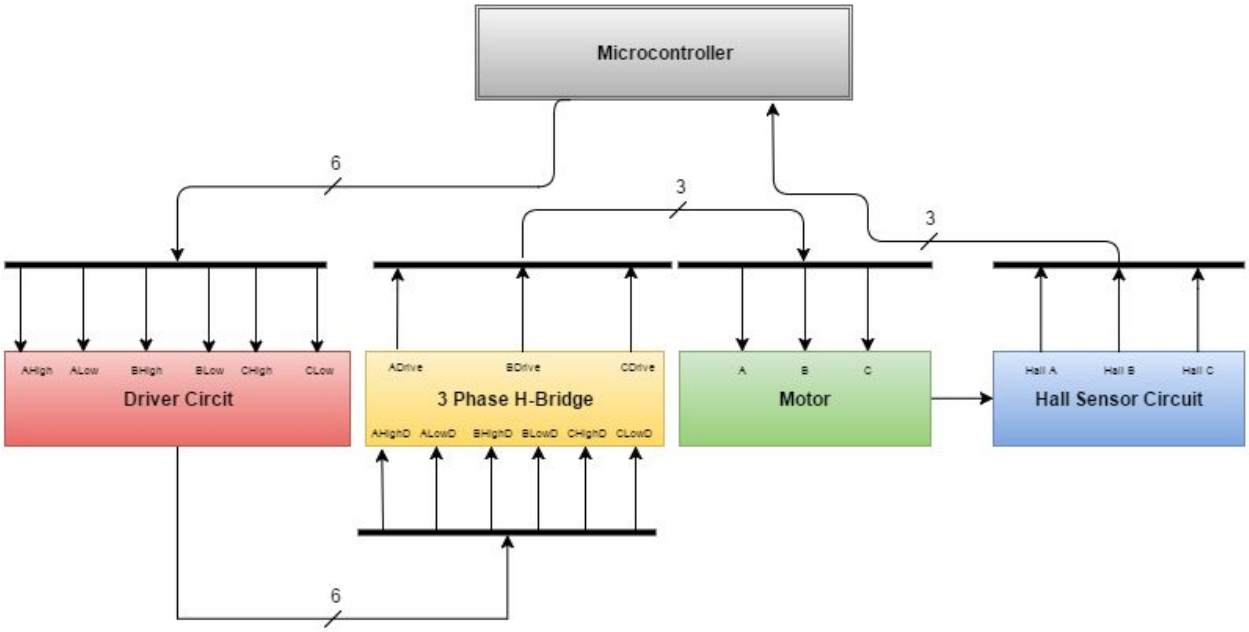


Figure 5: Motor Communications Loop

We can also see the how an actual design below in Figure 6 and 7 . This design mirrors our high level design diagram and we can again see a driver circuit, a three phase H-bridge, a hall sensor circuit

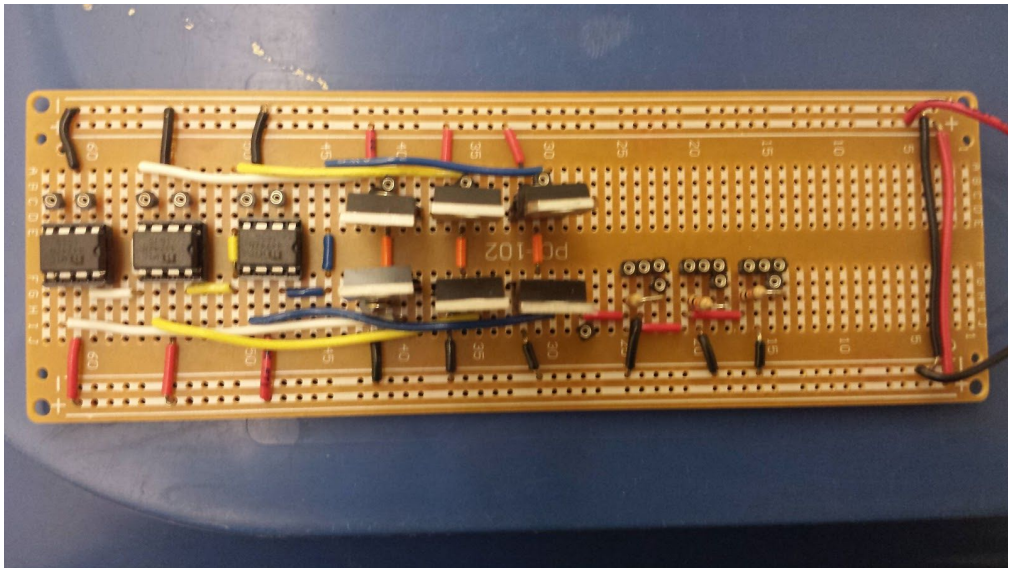


Figure 6. Complete Drive Circuit

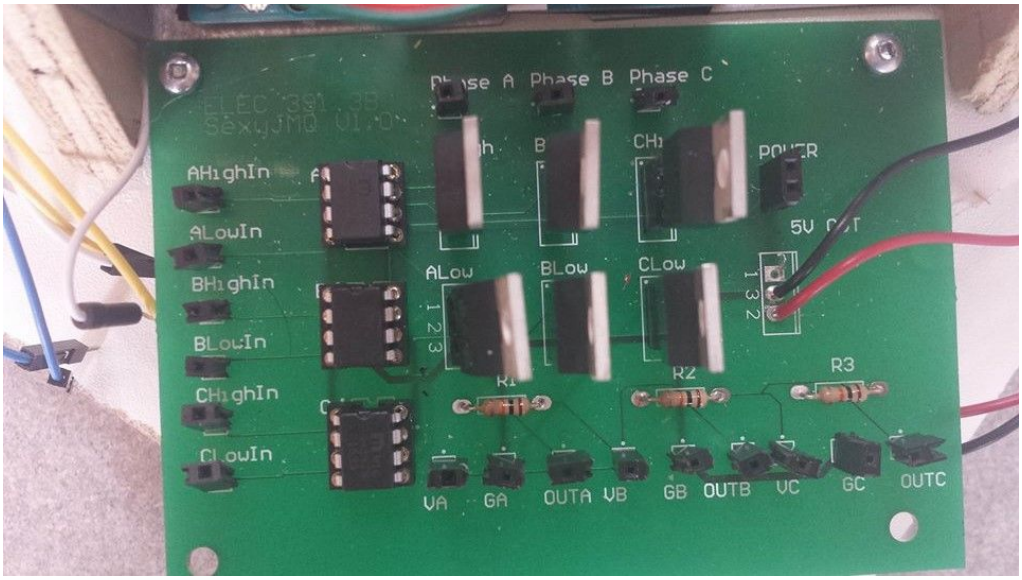


Figure 7. Complete Drive Circuit (PCB)

The microcontroller unit (MCU) that we are using is an Arduino Uno and it directly interface with our driver and hall sensor circuits. Starting from left to right, our driver circuit will take signals from our MCU, amplify a voltage, and feed an amplified voltage to our 3 Phase H-Bridge. Depending on the combination of inputs our H-bridge will either output a one, azero, or a floating voltage on our drive signals.The motor on a basic level is made up of a stator, a rotor, and coils. The motor is energized and hence commutated (made to spin) by the H-Bridge.

The hall sensor circuit is connected to our motor in such a way that we can sense the position of our rotor. These hall sensors are powered through the protoboard and are then connected the our microcontroller.

Detailed Design

Control System:

Microcontroller:

In the project we used an off the shelf Arduino UNO as a microcontroller.To implement the control system, we utilized a PID library to convert each of the external sensor inputs to a usable output for both the tail and the yaw motors (see appendix 5 for code). To measure the lift and yaw angles, we connected two optical encoders to the Arduino UNO microcontroller and wrote an interrupt service routine (ISR). This was triggered each time the optical encoder generated a pulse sequence and either incremented or decremented the respective angles depending on the relative phases of the quadrature pulses from the encoder. In order to generate proper PID values (Kp - Proportional, Ki - Integral, Kd - Differential), we modelled our robot in simulink in order to tune the values through simulation. To model our system, we began by drawing a force diagram to determine all the external and internal forces acting upon the system. These forces included things such as air drag, friction, gravity, propeller thrust, etc. To model gravity we weighed our motors with the propellers attached and related the torque about the arm's rotational axis to the measured lift angle. Modelling the thrust force required us to look up datasheets on our propellers to relate our motor's rpm to thrust through linear regression. To determine the dynamic friction forces acting on the robot's arms and base, we looked up the friction coefficients for the bearings used and calculated the frictional forces and converted them to equivalent frictional torques. To model air drag, we approximated our robot as a series of rectangles and calculated the

air drag relative to the robot's motion. These calculations and the simulation model can be seen in appendix 2. This followed with the manual tuning of the robot by altering the Kp, Ki, and Kd values after using the values from our simulation. This then allowed us to get a more stable system given the larger amount of real world constraints that must be accounted for.

Optical Encoder:

For understanding where the arms and base are relative to our desired set point, we opted to use two 100 count optical encoders. One of which is positioned on the base to detect our yaw, while the other is positioned on the arm to detect our lift. In order to understand the output that the optical encoders were giving us and use that information effectively in our design, we resorted to looking at a datasheet. This gave us information on what we should be looking at on the Arduino to record angle information as well us understanding how many degrees we increment on each pulse of the output. From there, we implemented code on the Arduino which uses the interrupts on the chip to detect any time the angle is increased or decreased. To determine which direction this is occurring, we look at which of the outputs is leading and which is lagging based on the design of the encoder. This worked effectively in determining the direction and we decrement or increment our current angle variable accordingly. All of this information is relayed instantly to our PID controller, allowing it to calculate the correct output for our system.

Current Driver:

In order to effectively control the tail motor using the microcontroller, we resorted to using a PWM current driver to control both the drivers. In order to use this off the shelf driver, we first looked at all the available information on how to control the driver by looking at a datasheet. This informed us that effectively communicate to the driver using an analog input. This we achieved through the use of a PWM output from our microcontroller and a low pass filter, which was used to level off the square wave output to ensure a relatively reliable analog signal. This proved to be effective in providing us an accurate control to the tail motor in both the clockwise and counterclockwise direction.

Robot:

To design the robot, we first ensured we understood at the design requirements of the project. This includes the requirements that the system is capable of moving with two degrees of freedom which is locked to the lift (pitch) and yaw. In order to achieve this, much collaboration was done at an early time between both the partners involved in the control group as to ensure that a design consensuous was met for a practical, reliable, and modular design. This was followed by some collaboration with the motor group as well to ensure we designed a system to accommodate the custom motor in phase III of the project.

Once we had an idea of an achievable design in mind, we worked out a rough sketch on paper. This then followed with the translation of that design into Solidworks

with our desired dimensions and matching dimensions to parts we knew we would be using. This 3D model was then used as a reference for the physical design and allowed both us control partners to effectively relay our thoughts on the final design. Despite attempting to follow the design a close as we can, we found early on that it was in our best interest for us to make changes as we went in order to accommodate the off the shelf parts and ensure that everything is built correctly, to speed up the design process, and still meet all RCG's as listed above. As a result, we found ourselves utilizing wood for the main body of the construction. We felt that this material was easiest to work with and gave us freedom in making changes from our original design which became extremely useful in the design process.

We also utilized various other components, both off the shelf and custom built using 3D printers and waterjet cutters, to finish the complete build. We used a 6 inch radius lazy suzan to give us our yaw rotation which we mounted to two pieces of wood. Inside the lazy susan we waterjet two aluminum rings which we used in conjunction with metal brushes to create a slip ring power system. This allowed our final design to rotate beyond the 180 degree requirement of the project as we were not constrained by external wiring.

From there we used off the shelf L brackets to connect wooden legs (cut by hand using the bandsaw) to the rotating base. These legs had two holes which we mounted two, low friction, skateboard bearings. We then placed a 5 inch carriage bolt with several nuts along it's shaft. This allowed us to connect two wooden arms in parallel to the shaft and clamp the arms to the bolt. This proved useful in detecting the angle of lift

as we could read the rotation of the bolt using the optical encoder (as described above). The commercial motor was mounted to the end of one of the arms using a carefully cut hole in which the motor could slide in. For the custom motor, a hand built, custom mount was incorporated to ensure a strong connection to the much larger size as opposed to its commercial counterpart.

Motor :

Microcontroller:

When choosing a microcontroller there were a list of criteria in selecting the best microcontroller. These criteria included ease of use, speed, features, and pin count. As far as ease of use, speed, and features, the Arduino was able to satisfy our requirements. The Arduino board is provided with a wealth of user defined functions and also an independent development environment (IDE) that allows for easy flashing of our MCU. The Arduino Uno doesn't sacrifice any features because you are still able to write directly to ports (groups of pins), able to use timers, and able to use interrupts. One of the most important functions that the Arduino Uno had was Pulse Width Modulation (PWM). Instead of having to use bit banging, we were able to make use of Arduino's user defined function library in order to provide a PWM input signal to our driver circuit. PWM in turn allows us to control the speed of our motor. The speed also played a large factor in choosing a suitable microcontroller. This requires fast processing of all of our signals in order to commutate the motor quickly enough. Given the 16 MHz Clock speed of the Arduino Uno, it was more than fast enough. Referring to our previous block diagram (Figure 5) there was a minimum pin requirement of 9 pins. The Arduino had more than enough pins to meet our requirements (Figure 8).

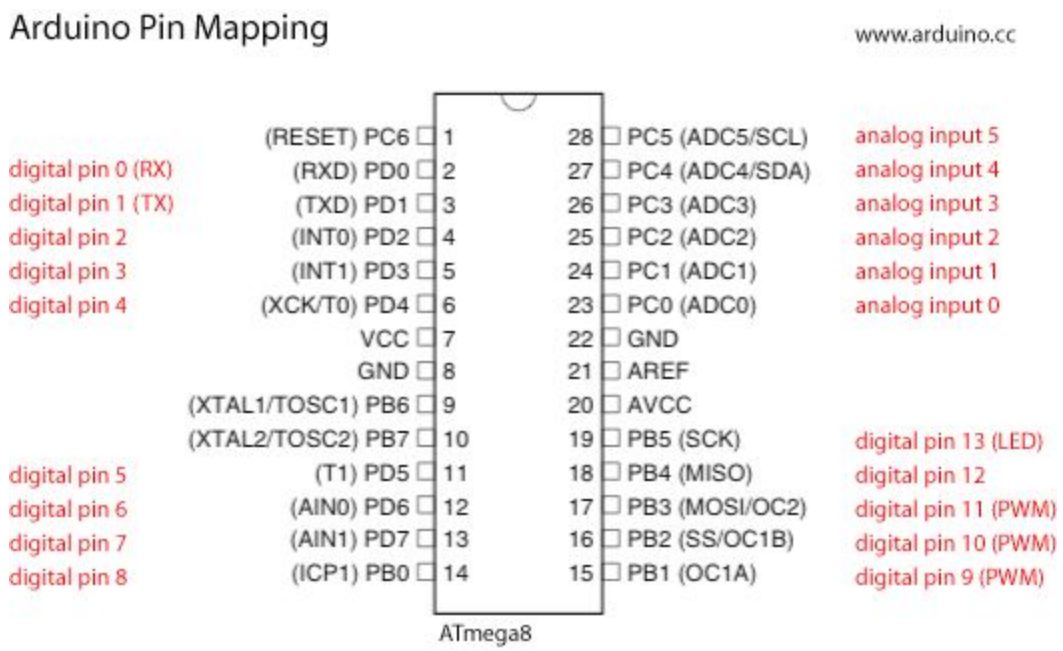


Figure 8: Arduino Uno Pin Map [9]

MOSFETs:

Given the importance in MOSFETs in our design we will include a brief section that outline the functionality of MOSFETs. For P- type MOSFETs when the gate voltage is low , the channel from drain to source is connected, and for N-type MOSFETs when the gate voltage is high, the channel from drain to source is connected. These channels from drain to source are low resistance channels that allow for current flow. This makes the A High, and B Low MOSFETs act as closed circuits, and hence allowing current flow from A to B. All the other MOSFETs are fed a gate voltage such that they act as open circuits. Drain current is determined by the amount of voltage at the source, and the impedance of a load. The Gate voltage must match the Source voltage such that the potential between them, Vgs (Voltage Gate Source), is zero. This allows for MOSFETs to properly turn on and off.

Figure 9: Enhancement Mode N and P Type MOSFETs Respectively [10]

3 Phase H-Bridge:

Power is absolutely necessary in driving any kind of loads. Given our force calculations we needed to draw a large amount of current. H-Bridge configurations are able to provide large amounts of current and are also able to provide currents in both positive and negative directions. H-Bridges work using MOSFETs as voltage enabled switches. A simplified version of a 3 phase H-Bridge is shown below (Figure 10) .

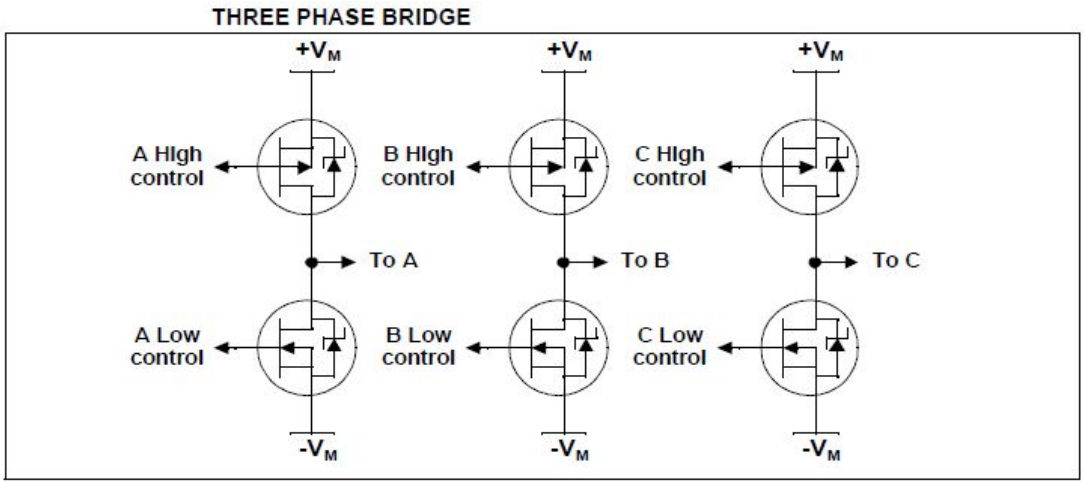


Figure 10: 3 Phase H-Bridge [5]

The top 3 MOSFETs are our P-Type MOSFETs, and our lower 3 MOSFETs are our N-Type MOSFETs. Our MOSFETs are connected to our motor in such a way that we only energize two phases at a time. A simplified version of our motor is represented in Figure 11.

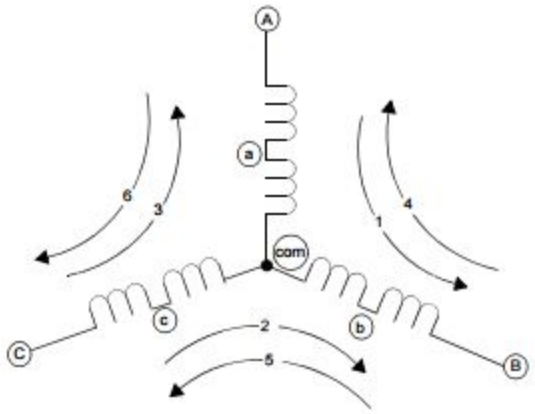


Figure 11: Simplified 3 Phase Motor [5]

If we are looking at the current path indicated by path 1, this is achieved by connecting the drain and source of A High (P-Type MOSFET), and connecting the drain and source of our B Low (N-type MOSFET). The exact commutation process to achieve full motor rotation is shown in the Appendix A3.3. An assembled version for the 3 Phase H-Bridge is shown on the next page (Figure 12.1 and 12.2).

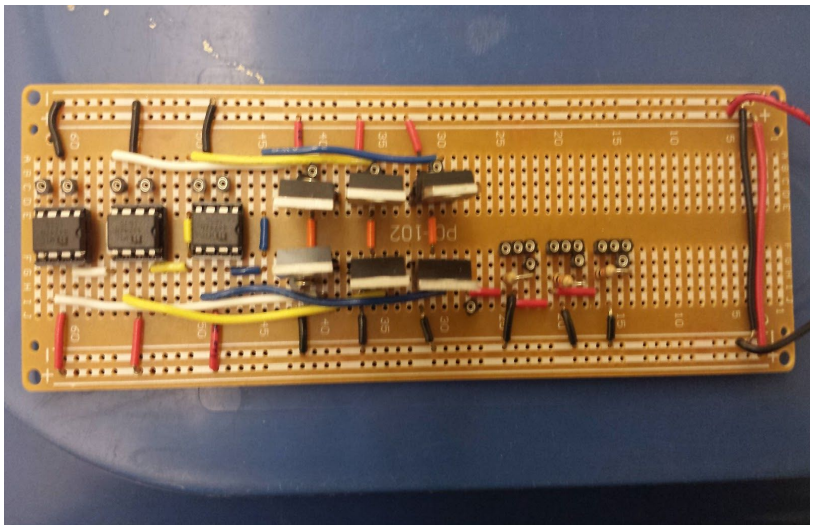


Figure 12.1: 3 Phase H-Bridge; Top Down (Protoboard)

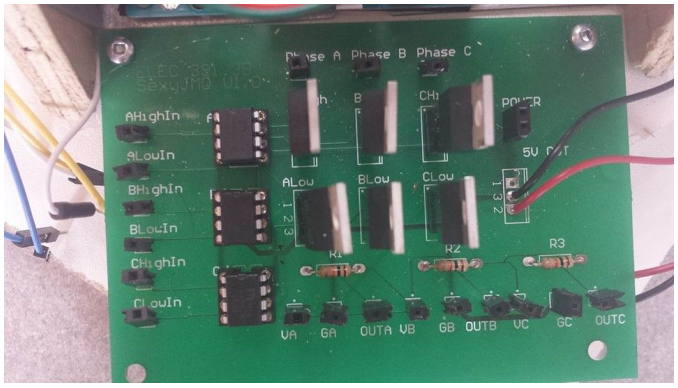


Figure 12.2: Circuit implemented on PCB

Both implementation of our circuit are shown on our protoboard and on the PCB. Although the orientation is slightly different, the functionality is identical. The circuits are split into three main functional blocks. The first block is the driver on the left hand side of both the protoboard and the PCB (figures 12.1 and 12.2). The driver circuit consists of the 6 inputs (AHigh, ALow, BHigh, etc) and, also of the three, eight pin driver chips. These driver chips are then connected to the second functional block which is at the

center of both the protoboard and the PCB. The H Bridge consists of 6 MOSFETs which power Phase A, B, and C, which are the windings of the motor. The third and final block of the circuit is the hall sensor circuit. This provides a common 5 Volts and a common ground to all the hall sensors. An external pullup resistor is included. The external pullup resistor was a design mistake as the particular hall sensors that we are using have an internal pullup resistor.

The detail connections of the circuit and the PCB are outlined in both schematics below (figures 12.3 and 12.4).

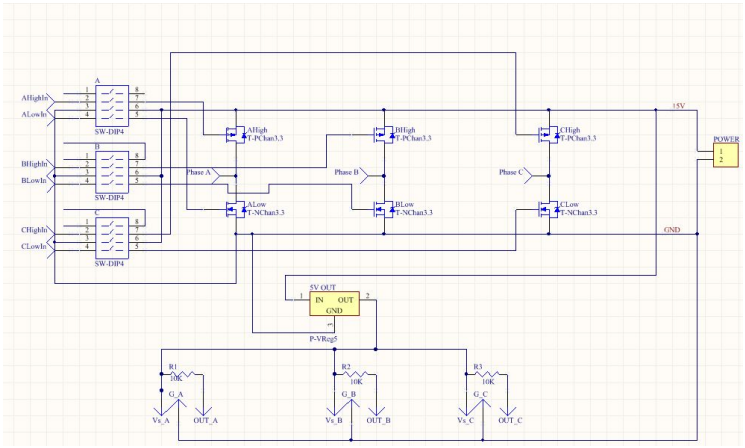


Figure 12.3: Circuit Schematic

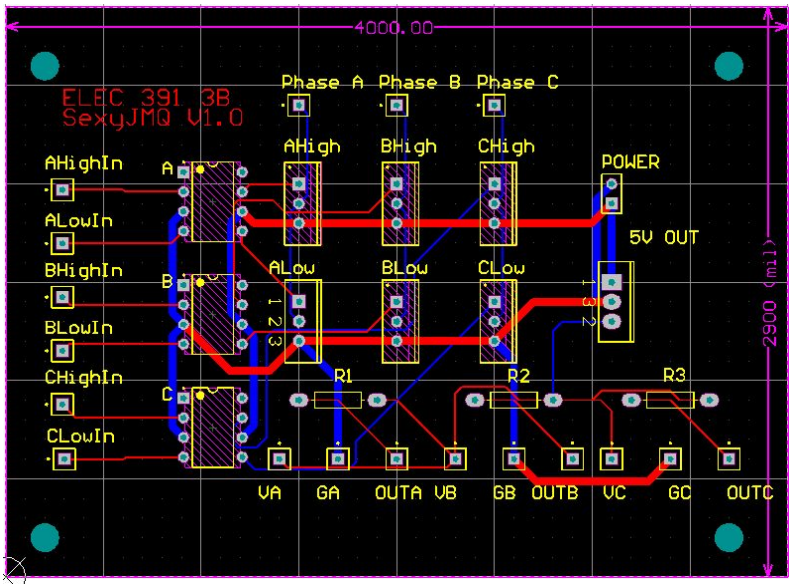


Figure 12.4: PCB Schematic

We will now finish our discussion by describing the driver, and hall sensor circuits.

Driver Circuit:

A driver circuit came about as a result of the three phase H Bridge. As previously mentioned, this is the leftmost portion of our previous circuit in figures 12.1 and 12.2. *This driver circuit operates by taking an input voltage and outputting that at the same level as the source voltage. For example if my source voltage Vs is 15V and my input voltage is a digital high (5V), then my output voltage will be 15V on the corresponding pin. A pin diagram of the voltage driver is shown below (figure 13).*

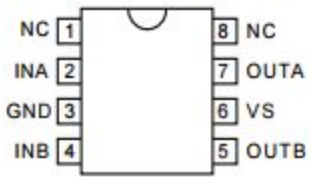


Figure 13. Driver Chip Pin Diagram [7]

There are three main concerns when switching an H-Bridge. The first concern arises when using higher source voltages. As outlined in the MOSFET section, higher drain current for a given load is achieved by using a higher voltage on the source pin of a MOSFET. The problem arises that with the P-Type MOSFET the source is directly tied to our supply voltage. This means that as we increase the supply voltage, we must equally increase the gate voltage such that we can turn off our MOSFET . As described before, the voltage between the gate and source (V_{GS}) must be zero in order for our N-type MOSFET to be on. Without the driver circuit we would never be able to turn our MOSFETs on or off. The second concern is with regards to switching speed. A gate of a MOSFET is non-ideal and actually has a capacitance. As a result when the voltage of the gate of a MOSFET is set high this gate capacitance starts to gain charge and hence a voltage. When you turn off your gate signal the MOSFET gate does not instantly go to zero because of the stored charge (Voltage) on the gate of the MOSFET. In order for the gate voltage to go to zero, the excess charge must be drained. This is achieved allowing current to flow off the gate of the capacitor to ground. Without a driver circuit our microcontroller is responsible for draining this current off. When Arduino Uno pins are acting as outputs they are able to sink a current of up to 40 mA. If we look at the formula for current in a capacitor we have i = C * dV/dt. As current increases, our time change in voltage also increases. In low switching speed applications, this is sufficient enough but as we start switching at higher and higher frequencies, we need our dV/dt to increase to match this switching frequency. Driver circuits allow for much more current to be sunk and hence allow for faster switching speeds of our MOSFETs. Without a

gate driver, we would not be able to switch our MOSFETs at a high enough frequency to obtain our desired RPM.

Hall Sensor Circuit:

A hall effect sensor is used to detect a changing magnetic field. In our case, every time we spin our motor we want to know exactly where we are. By using three Hall effect sensors we were able to achieve this. Below is a small graphic of how we wired our Hall sensor (Figure 14) and also a picture of our hall sensor circuit (Figure 15).

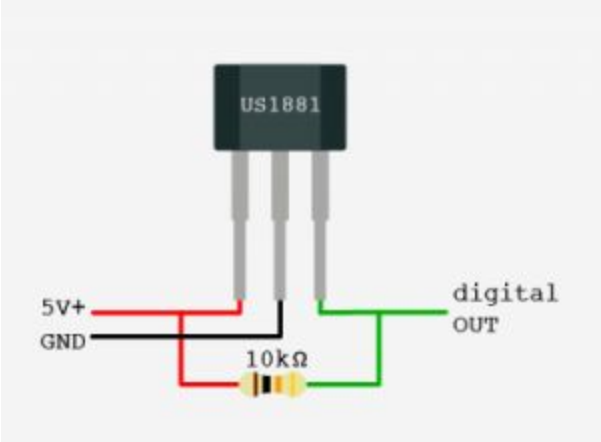


Figure 14: Hall Effect Sensor Setup [4]

The hall sensor circuit operates using four pins. These groups of four pins can be seen in figure 3, starting on the 15th column and ending on the 25th column. We will discuss the role of each pin starting from left to right. The leftmost pin in each of the groups provides a common 5V source to each of our hall sensors. The center pins provides a common ground for all of our hall sensors. The third and fourth pins share

the same node and they are both designed to transmit the output of our hall sensors. The top of the right pin connects directly to our hall sensor output, and the bottom right pin connects the same output of our hall sensors into our microcontroller.

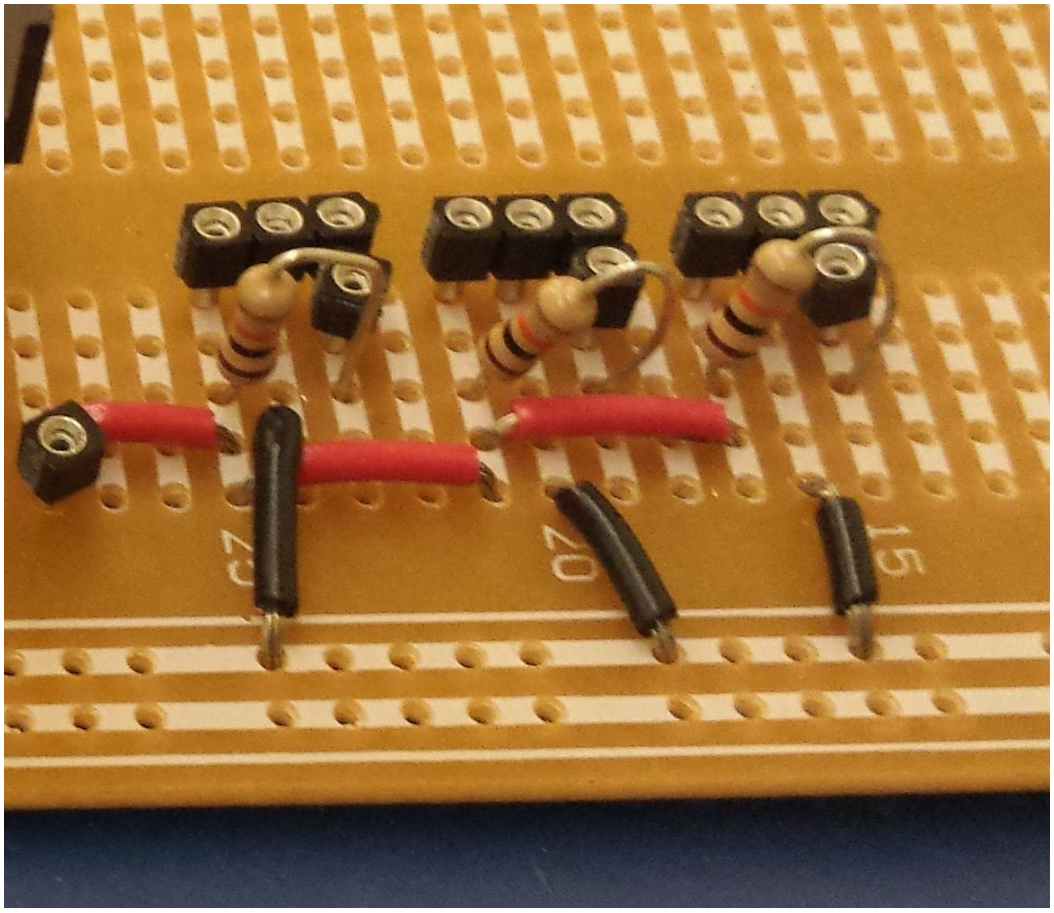


Figure 15: Hall Effect Sensor Circuit

The hall sensors output is currently connected to the Arduino board. The pin slots are put in place of the hall sensors such that the hall sensors can be at the proper physical location on the motor, and connected to the circuit at the same time. Using pull up resistors we were able to ensure that if our signal isn't being driven then it will always remain as high. Hall sensors function by outputting a 0 or a 1 depending on whether the hall sensor detects a magnetic north and south pole. This also depends upon which

side of the hall effect sensor you are using. Referring to Figure 14, we used the side that was not facing towards the viewer, to directly face the rotor magnets. By using three hall sensors we are able to sense the position of our rotor and commutate our motor accordingly. Our PCB is wired with the same connection but has a much neater layout (figure 12.2).

After testing we were able to determine uniquely the position of our rotor based on the input of our hall sensors. We then used this data to commutate a specific phase for a specific set of input of hall sensors. The commutation sequences with the phases can be seen in the Appendix (A3.2) and a table outlining our hall sensor values and our phases can be seen below (Table 1)

Phase	Hall A	Hall B	Hall C
1	0	1	1
2	0	0	1
3	0	0	0
4	1	0	0
5	1	1	0
6	1	1	1

Table 1: Hall Effect Sensor Values for a Given Phase

The Hall sensors are placed in three small grooves of the stator sensor cage (A3.4.6), and are each located 60° apart.

Stator:

Our stator consists of three parts, the Stator top, Cores and Stator bottom plate.

- Stator top

  Stator top consists of a round shape with a deep groove to keep the copper wires in place. (A3.4.3)

- Cores

  For each windings, we stacked up three layers of sanded and laminated Iron pieces as the core, and 200 windings with each pole. (A3.4.1)

Figure 16: Stator Core

- Stator bottom plate

  The stator bottom plate has similar shape as the stator top, but with small poles that can keep each core in place. (A3.4.0)

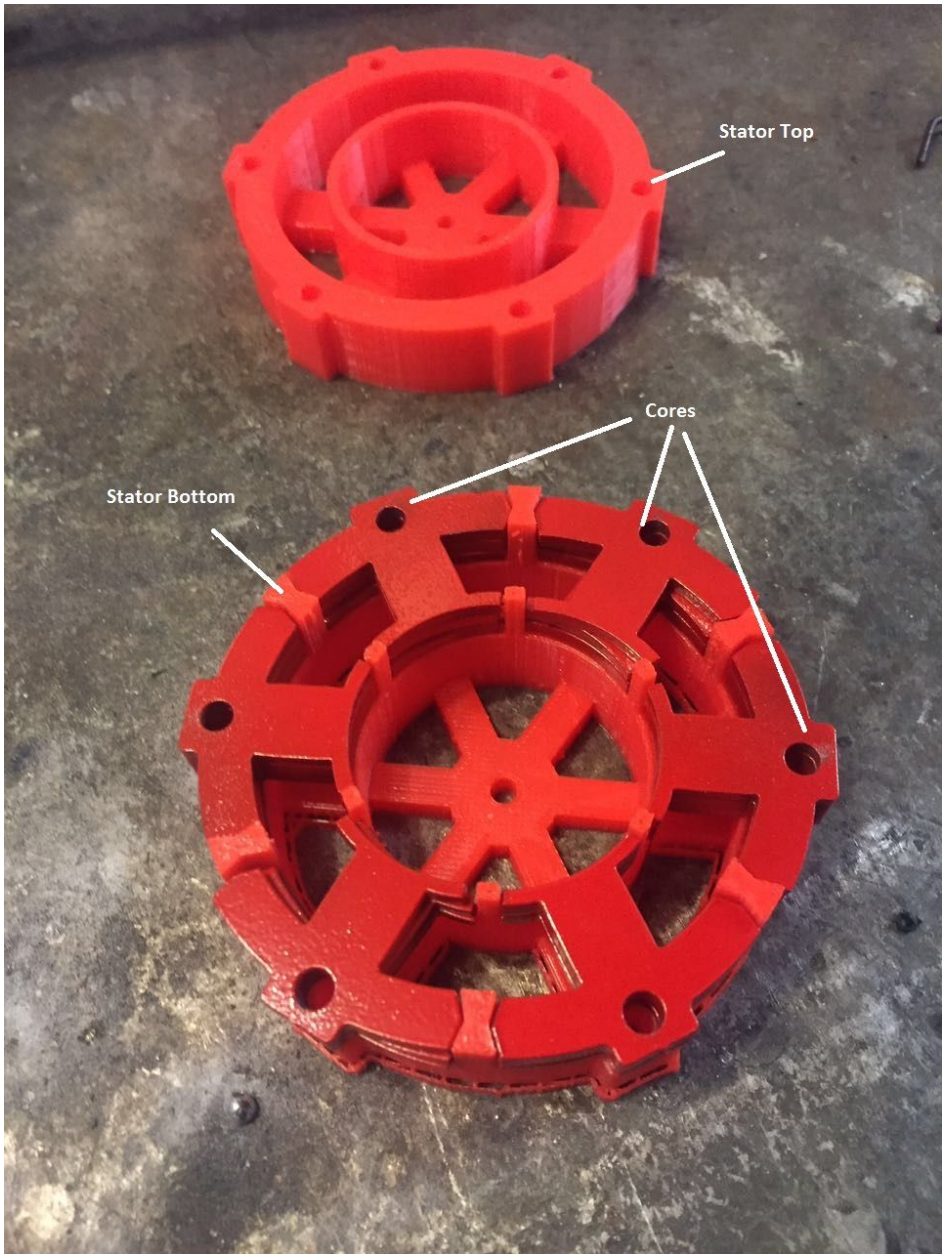


Figure 17: Stator Internal Structure

The biggest advantage of this design is its modularity. With the design of this stator bottom plate, each core is easy to inserted or removed when stator top is not

covering the top. This gives us a great flexibility in assembling the stator, troubleshooting the motor, and swapping components when necessary.

The deep groove in the stator top and stator bottom plate also allows us to including enough windings within the stator, and not creating any unnecessary gaps between each parts. Thickness was limited in order to limit eddy currents, and also to ensure precise cuts when using the waterjet. By having a thinner stator, this increases the resistivity of the stator such that any eddy currents will be greatly reduced. The thinner stator allows for us to use multiple layers of stators and laminate each sheet to also reduce eddy currents between different layers of stators.

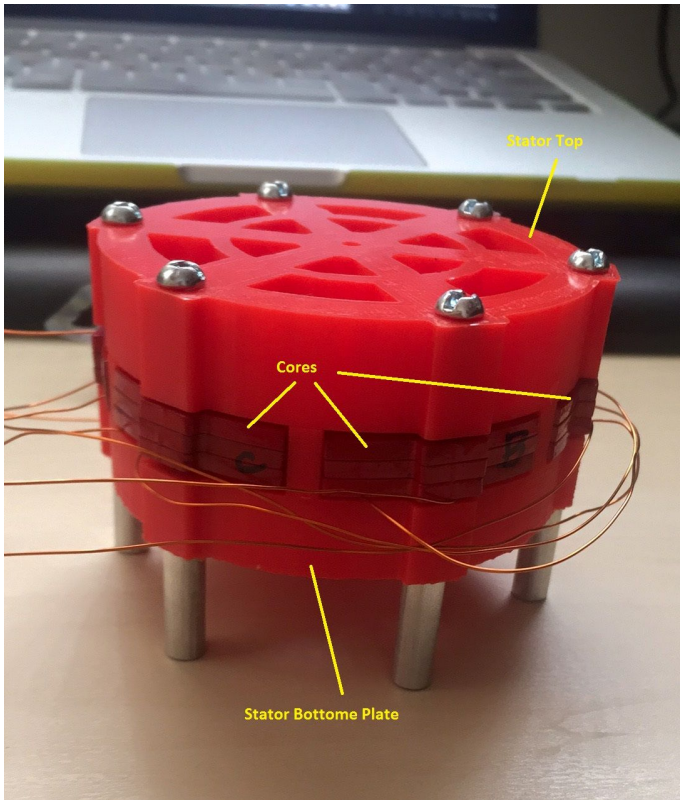


Figure 18: Exterior View of the Assembled Stator

Rotor:

Our rotor drawing is shown below (Figure 19)

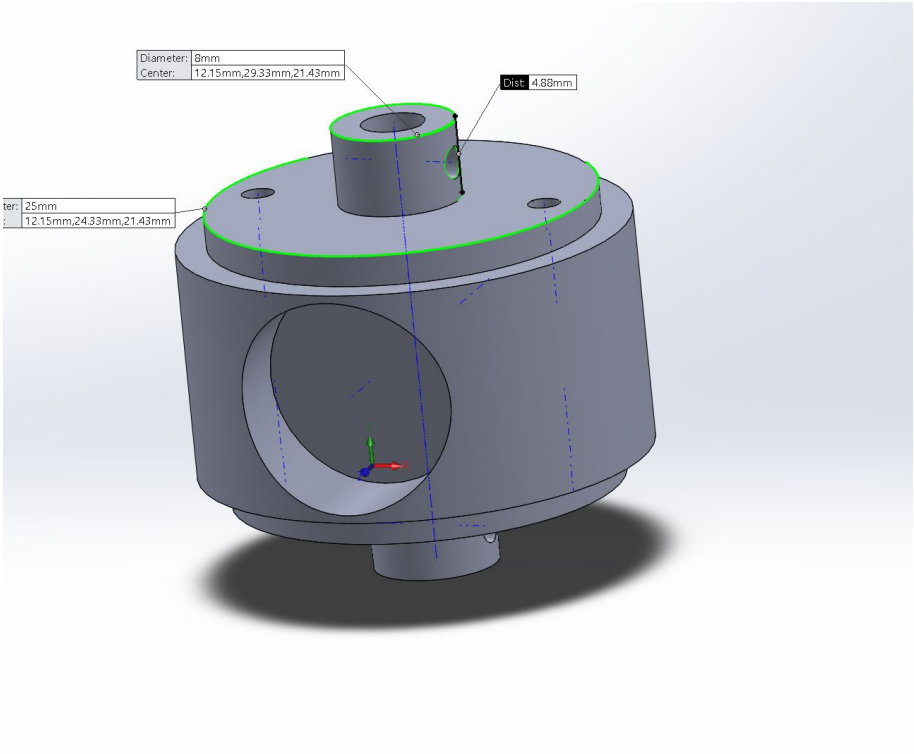


Figure 19: Rotor Drawing with Rotor Holders Attached

In order to commutate as simply as possible (A3.2). We only wanted to have two poles, one north and one south. The aim of the rotor was to be as lightweight as possible and store the strongest magnets possible to maximize force on the rotor, and hence maximize the torque and RPM.

Integration

In order to successfully integrate both the control system/robot with the custom motor, several design changes were necessary. One of these first changes came in the form of a PCB redesign. This redesign encompassed the addition of several low pass filters between both in inputs (optical encoders) and outputs (PWMs) to the microcontroller. From there, this PCB also included the addition of female header pin connectors to allow for easy connection between all external components. This was then mounted to the base and wired up with the proper connections.

From there, we then moved onto the mechanical changes. This first and foremost included the addition of the custom motor. In order to achieve this, we designed a custom motor mount with mounting holes that allowed for easy removability of the motor if needed. We then utilized some of our spare motors to counterbalance the additional weight of the custom motor to ensure the arm would lift. Lastly, given the extra weight added, we found it best for us to add a low friction skateboard bearing to significantly reduce the friction and eliminate the possibility of the arm getting stuck.

From a control perspective, we had to alter the communication method for the main lift motor as we had now shifted from a commercial motor to the custom one. This meant that we had to come up with a new system to quickly and effectively communicate between both microcontrollers. We opted to go with an analog signal which was generated using a PWM from the one controller. This in turn was read using the analog input from the second, motor microcontroller to reliably run at the correct speed.

Finally for the simulation, we made changes to the simulation given our updated design as these greatly affected our model. This included our counterbalance weight, rotation friction in the shaft, and all of the new custom motor parameters. These values were measured and implemented into the Simulink model.

Validation

Control System:

Microcontroller:

In order to test the microcontroller, we performed several tests by printing the various internal variables onto the serial monitor built into the Arduino software in order for use to capture both the inputs and outputs of the system. This allowed for accurate debugging and verification in all levels of the code including the input processing of the optical encoders as well as the PID output processing to the current driver.

After some debugging, we found that the microcontroller met the validation requirements with the exception of the PID which we found was not as responsive as we had outlined in our goals.

Optical Encoder:

The test of the optical encoder was also encompassed as part of the microcontroller validation. The validation of the encoder also included the probing of the outputs for both encoders to ensure that the output we saw on the scope matched that of the datasheets desired output.

We found that the encoder worked exactly as we had expected and meet all our requirements for the design.

Current Driver:

To perform the validation on the current driver, we performed various speed control tests by varying the input speed parameters. From there, we would check to see if we could correctly change the direction of the motors based on the required serial inputs of the current driver.

Through these tests we were fully successful in accurately varying both the speed and directions of the motors through the use of the current driver, although we found that it was not as reliable as we had hoped.

Robot:

To validate the robot, we first ensured that the system was able to correctly move along the two degrees of freedom (lift and yaw) as specified in our requirements. This also encompassed a smooth, consistent motion along both axis which was helpful in both the modeling and tuning of the system. We then placed the robot in the enclosure to ensure it met the required dimensions. As well, we checked the rigidity and reliability of the robot to ensure that the system would not fail mechanically.

In all of these test, we found that the robot met these requirements and was a reliable system, able to perform all of the required features.

Motor :

We tested the motor in various aspects. Since the Motor is composed of 5 components as indicated in the high level design, we examine the function of each component carefully.

Microcontroller:

For the microcontroller, we mainly test if the logic for switching current phases is correct. This is done by checking if the Arduino board is able to read the correct signal from hall sensors, and checking if it sends the correct digital signal out of the corresponding pins.

The test result shows that the microcontroller is working as we expected that it is able to read the position of rotor and switch current phases accordingly in a relatively high speed.

Hall sensor circuit:

For the Hall sensor circuit, we test whether it can detect the changes in magnetic field in time, and to check if it can flip the digital signal correspondly.

Our Test result shows that the Hall sensor circuit is able to detect the position of the rotor with very high accuracy and speed.

Driver circuit:

For the driver circuit, we test if it can send out the correct signal upon receiving the digital signal from Arduino. We also check if the driver circuit is able to handle PWM signals.

Our Test indicates that the driver circuit is working as expected.

3 Phase H-Bridge:

The main test we have on the H-bridge is if it can be turn on by the signal from current driver (either active low on the P-Mosfet, or active high on the N-Mosfet). We also check the current flow within the H-Bridge circuit to ensure that only one mosfet is turned on for each channel, and the backflow current would not damaged the mosfets.

Our test result indicates that the H-bridge is working as expected within the rated range. Therefore we are confident in continue using the H-bridge circuit.

Motor:

For the motor itself, we checked the resistance of wires to ensure there is no open circuit in the copper wires. We also test the strength of each solenoid in each phase by supplying constant current and then measure the force it generated.

Our test result indicates that the motor itself is able to produce the magnetic force to spin the rotor, but it is not able to provide sufficient torque.

Other than these 5 main components, we had also test the each requirement in RCG's. We found that we still need to work on the mechanical design the the motor to reduce the size, as well as strengthen the joints to ensure the motor the perform consistently, and to improve stability and durability. We also need to increase the torque of motor by providing higher current, and thus require us to modify the parameters of each solenoid. Testing to the RCG's requirement are provided in Appendix.

Appendix

A1: Robot

A1.0 Robot Conceptual Diagram

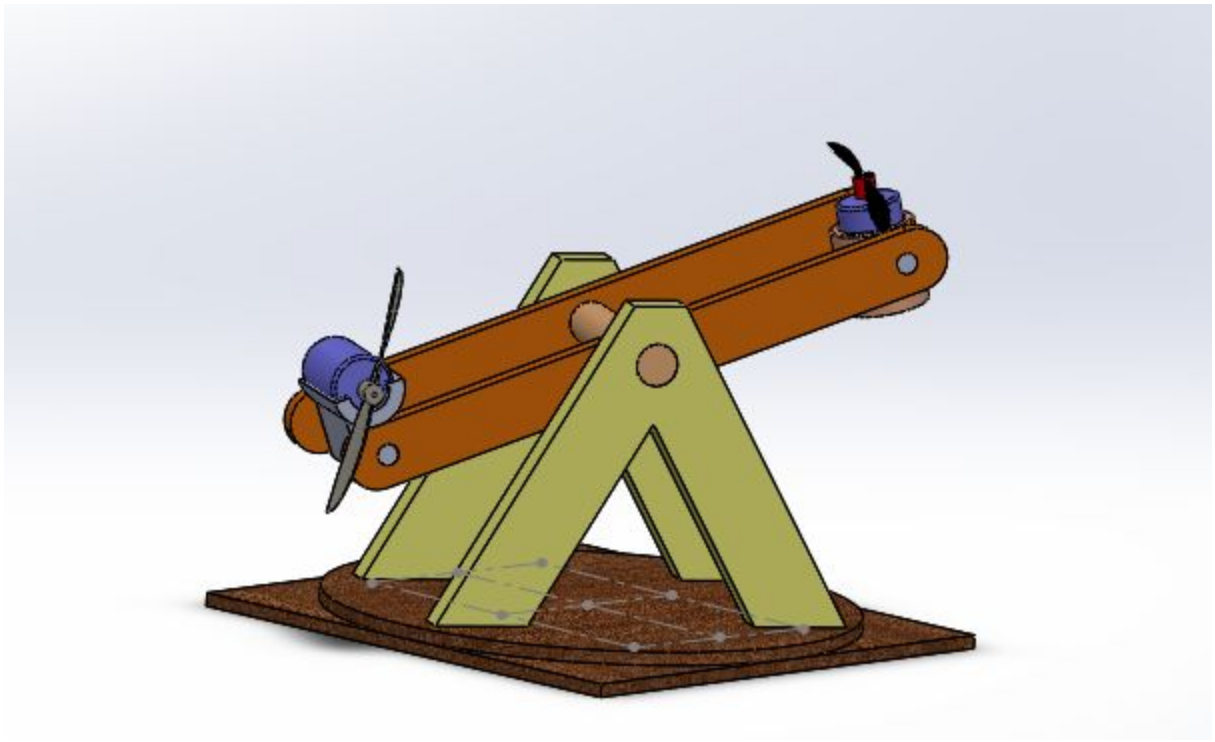


Figure A1.0: Conceptual Robot Design

A1.1 Robot Physical Implementation

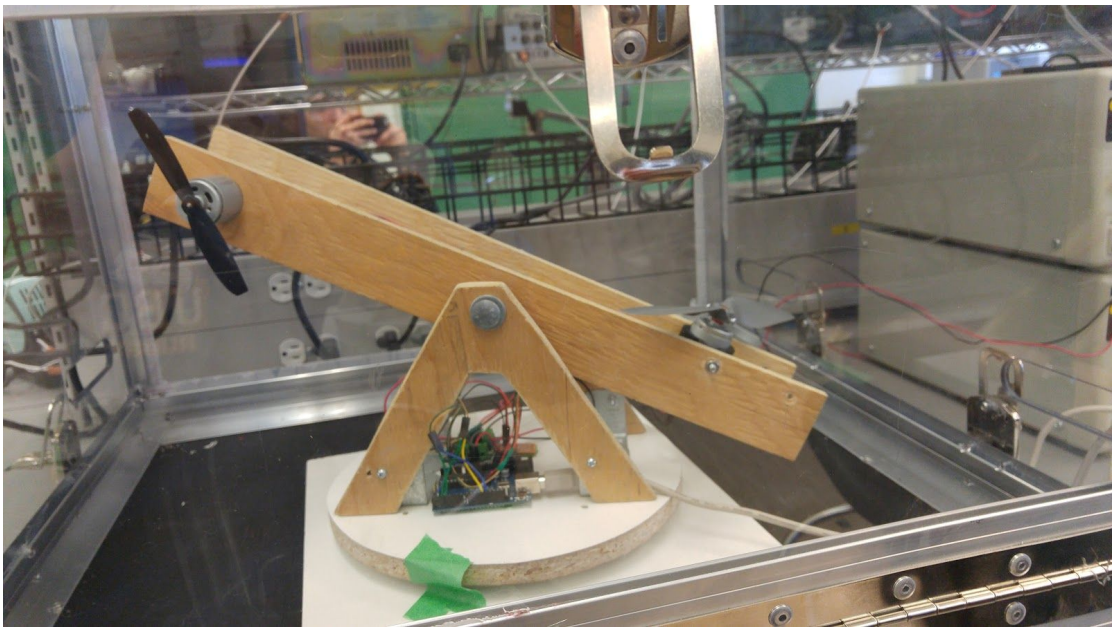


Figure A1.1: Physical Robot Implementation

A2.0 Robot Parameters

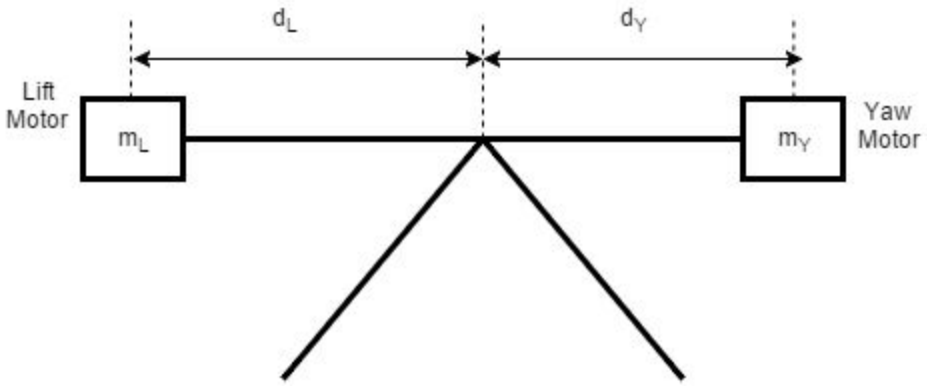


Figure A2.0: Sketch of Robot, Parameters Labelled

Measured parameters:

m_L = 0.315kg	m_Y = 0.06kg	d_L = 0.152m	d_Y = 0.180m

Table A1: Measured Robot Parameters

There are also two counter balances weighted at 0.06kg each and located at 0.068m and 0.19m to the left of the central axis as pictured above.

A2.1 Effective Mass of Lift Motor

The following equations take into account the counterbalancing effect of the yaw motor on the lift motor to find the net torque (T_{net}) of the two motors and find an effective mass for the lift motor. Here, g = gravitational acceleration and m_L* = effective mass of lift motor, m_c = mass of counterweight, d_{c1} and d_{c2} = distances to the counterweights.

$$T_{net} = m_L d_L g - m_Y d_Y g - m_c d_{c1} g - m_c d_{c2} g = m_L^* d_L g$$

$$(0.315)(0.152) - (0.06)(0.18 + 0.068 + 0.19) = (m_L^*)(0.152)$$

$$m_L^* = 0.14211 \text{ kg}$$

A2.0 Simulation Measurements

A2.1: Air Drag

In order to estimate the air friction of our robot, we will use the equation:

$$F_d = (0.5\rho C_d A)v^2$$

Where ρ = density of air, C_d = unitless constant depending on the shape of our object, A = area of our robot for a given axis of rotation, v = velocity of motion. In order to calculate the drag for rotational and lift motion, we need to find the area pushing against the air flow for a given direction of motion. To do this, we will assume all our parts to be rectangular (C_d = 2.1) and we will measure the area of our robot. The results are as follows:

For lift motion, $A_{eff} = 0.008691m^2$, and $F_d = (0.01686)v^2$

For rotational motion, $A_{eff} = 0.040231m^2$, and $F_d = (0.0506)v^2$

A2.2: Bearing Friction

The friction of the base was easily calculated using:

$$F_{fr} = (\mu_k)m_R$$

Where μ_k is the coefficient of friction of our lazy susan bearing, estimated as $\mu_k = 0.0050$ (based on values for similar bearings [8]). The mass of our robot was weighed to be:

$$m_R = 1.48kg$$

To convert this to a resistive torque, we simply multiply the frictional force by the radius of the lazy susan (6in) and get:

$$T_{fr} = 0.01327 \text{ Nm}$$

A2.3: Motor Parameters

For the simulation, we had to measure several parameters of our motor. The measured parameters included the internal resistance and inductance of the motor, the back emf constant, and torque constant. To measure the back emf of the motors, we used a drill to spin the motor's shaft and measured the voltage induced using a multimeter. The results were as follows:

	R (Ω)	L (mH)	K_t (N*m/A)	K_e (V/RPM)
Commercial Motor	5	1.05	0.01528	0.0016
Custom Motor	15.95	8.239	0.0009084	0.000055

Table A2: Measured Motor Parameters

A2.3 Thrust Force Data, Thrust as a Function of Shaft RPM

In order to relate thrust to motor rpm, we downloaded the data sheets for our propellers, plotted the data at 0mph as a function of rpm, and linearly approximated the curves to get thrust as a function of rpm using Excel. The data is as follows:
Lift Propeller: http://apcserve.w20.wh-2.com/v/PERFILES_WEB/PER3_10x55MR.dat
Yaw Propeller: http://apcserve.w20.wh-2.com/v/PERFILES_WEB/PER3_7x3.dat

Lift Propeller Data

====== PERFORMANCE DATA (versus advance ratio and MPH) ======

DEFINITIONS:
J=V/nD (advance ratio)
Ct=T/(rho * n**2 * D**4) (thrust coef.)
Cp=P/(rho * n**3 * D**5) (power coef.)
Pe=Ct*J/Cp (efficiency)

V (model speed in MPH)

PROP RPM = 2000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1204	0.0520	0.003	0.105	0.153
0.5	0.03	0.0649	0.1194	0.0527	0.003	0.107	0.152
1.1	0.06	0.1268	0.1183	0.0534	0.003	0.108	0.151
1.6	0.09	0.1859	0.1170	0.0540	0.003	0.110	0.149
2.2	0.11	0.2421	0.1155	0.0546	0.004	0.111	0.147
2.7	0.14	0.2953	0.1138	0.0551	0.004	0.112	0.145
3.3	0.17	0.3456	0.1118	0.0556	0.004	0.113	0.142
3.8	0.20	0.3929	0.1096	0.0559	0.004	0.113	0.140
4.3	0.23	0.4373	0.1068	0.0559	0.004	0.113	0.136
4.9	0.26	0.4787	0.1037	0.0558	0.004	0.113	0.132
5.4	0.29	0.5171	0.1001	0.0554	0.004	0.112	0.128
6.0	0.31	0.5522	0.0961	0.0548	0.004	0.111	0.122
6.5	0.34	0.5840	0.0918	0.0540	0.003	0.109	0.117
7.0	0.37	0.6125	0.0871	0.0529	0.003	0.107	0.111
7.6	0.40	0.6374	0.0827	0.0520	0.003	0.105	0.105
8.1	0.43	0.6592	0.0781	0.0509	0.003	0.103	0.100
8.7	0.46	0.6781	0.0735	0.0497	0.003	0.101	0.094
9.2	0.49	0.6945	0.0688	0.0482	0.003	0.098	0.088
9.8	0.52	0.7083	0.0636	0.0462	0.003	0.094	0.081
10.3	0.54	0.7196	0.0582	0.0440	0.003	0.089	0.074
10.8	0.57	0.7283	0.0527	0.0414	0.003	0.084	0.067
11.4	0.60	0.7345	0.0471	0.0386	0.002	0.078	0.060
11.9	0.63	0.7393	0.0425	0.0362	0.002	0.073	0.054
12.5	0.66	0.7405	0.0377	0.0335	0.002	0.068	0.048
13.0	0.69	0.7345	0.0321	0.0300	0.002	0.061	0.041
13.5	0.72	0.7174	0.0259	0.0259	0.002	0.052	0.033
14.1	0.74	0.6793	0.0196	0.0215	0.001	0.044	0.025
14.6	0.77	0.5950	0.0132	0.0172	0.001	0.035	0.017
15.2	0.80	0.4154	0.0066	0.0128	0.001	0.026	0.008
15.7	0.83	-0.0099	-0.0001	0.0085	0.001	0.017	0.000

PROP RPM = 3000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1205	0.0516	0.011	0.235	0.346
0.8	0.03	0.0643	0.1195	0.0523	0.011	0.239	0.343

1.6	0.06	0.1257	0.1184	0.0530	0.012	0.242	0.339
2.4	0.08	0.1841	0.1171	0.0537	0.012	0.245	0.336
3.2	0.11	0.2395	0.1156	0.0543	0.012	0.248	0.332
4.0	0.14	0.2919	0.1139	0.0549	0.012	0.251	0.327
4.8	0.17	0.3414	0.1120	0.0554	0.012	0.253	0.321
5.6	0.20	0.3879	0.1099	0.0558	0.012	0.255	0.315
6.4	0.23	0.4315	0.1074	0.0560	0.012	0.256	0.308
7.2	0.25	0.4722	0.1046	0.0561	0.012	0.256	0.300
8.0	0.28	0.5100	0.1013	0.0559	0.012	0.255	0.291
8.8	0.31	0.5448	0.0977	0.0555	0.012	0.253	0.280
9.6	0.34	0.5764	0.0938	0.0549	0.012	0.251	0.269
10.4	0.37	0.6051	0.0895	0.0541	0.012	0.247	0.257
11.2	0.39	0.6307	0.0850	0.0531	0.012	0.242	0.244
12.0	0.42	0.6534	0.0804	0.0519	0.011	0.237	0.230
12.8	0.45	0.6731	0.0756	0.0506	0.011	0.231	0.217
13.6	0.48	0.6902	0.0707	0.0490	0.011	0.223	0.203
14.4	0.51	0.7047	0.0656	0.0472	0.010	0.215	0.188
15.2	0.53	0.7168	0.0604	0.0451	0.010	0.206	0.173
16.0	0.56	0.7263	0.0550	0.0426	0.009	0.195	0.158
16.8	0.59	0.7333	0.0495	0.0399	0.009	0.182	0.142
17.6	0.62	0.7374	0.0438	0.0367	0.008	0.168	0.126
18.4	0.65	0.7375	0.0379	0.0332	0.007	0.152	0.109
19.2	0.68	0.7313	0.0318	0.0294	0.006	0.134	0.091
20.0	0.70	0.7150	0.0257	0.0252	0.005	0.115	0.074
20.8	0.73	0.6806	0.0194	0.0208	0.005	0.095	0.056
21.6	0.76	0.6025	0.0130	0.0164	0.004	0.075	0.037
22.4	0.79	0.4283	0.0066	0.0121	0.003	0.055	0.019
23.2	0.82	-0.0045	0.0000	0.0078	0.002	0.035	0.000

PROP RPM = 4000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1207	0.0517	0.027	0.420	0.615
1.1	0.03	0.0634	0.1197	0.0525	0.027	0.426	0.610
2.1	0.06	0.1239	0.1186	0.0532	0.027	0.431	0.604
3.2	0.08	0.1816	0.1173	0.0538	0.028	0.437	0.598
4.2	0.11	0.2365	0.1159	0.0544	0.028	0.442	0.591
5.3	0.14	0.2885	0.1142	0.0550	0.028	0.446	0.582
6.3	0.17	0.3376	0.1123	0.0554	0.029	0.450	0.572
7.4	0.19	0.3839	0.1102	0.0558	0.029	0.453	0.562
8.4	0.22	0.4273	0.1077	0.0560	0.029	0.454	0.549
9.5	0.25	0.4679	0.1048	0.0560	0.029	0.454	0.534
10.5	0.28	0.5057	0.1016	0.0558	0.029	0.453	0.518
11.6	0.31	0.5405	0.0980	0.0554	0.029	0.450	0.500
12.6	0.33	0.5723	0.0941	0.0548	0.028	0.445	0.480
13.7	0.36	0.6011	0.0899	0.0540	0.028	0.438	0.458
14.7	0.39	0.6269	0.0854	0.0530	0.027	0.430	0.435
15.8	0.42	0.6497	0.0808	0.0518	0.027	0.420	0.412
16.8	0.44	0.6696	0.0761	0.0505	0.026	0.409	0.388
17.9	0.47	0.6868	0.0711	0.0489	0.025	0.397	0.363
18.9	0.50	0.7014	0.0661	0.0471	0.024	0.382	0.337
20.0	0.53	0.7137	0.0609	0.0451	0.023	0.365	0.311
21.0	0.56	0.7234	0.0556	0.0427	0.022	0.346	0.283
22.1	0.58	0.7305	0.0500	0.0399	0.021	0.324	0.255
23.1	0.61	0.7351	0.0443	0.0368	0.019	0.299	0.226
24.2	0.64	0.7358	0.0384	0.0334	0.017	0.271	0.196
25.3	0.67	0.7304	0.0323	0.0295	0.015	0.239	0.165
26.3	0.69	0.7154	0.0261	0.0253	0.013	0.205	0.133
27.4	0.72	0.6830	0.0197	0.0208	0.011	0.169	0.100
28.4	0.75	0.6103	0.0132	0.0163	0.008	0.132	0.068
29.5	0.78	0.4415	0.0067	0.0118	0.006	0.095	0.034
30.5	0.81	-0.0021	0.0000	0.0073	0.004	0.060	0.000

PROP RPM = 5000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1208	0.0516	0.052	0.654	0.962
1.3	0.03	0.0630	0.1199	0.0523	0.053	0.663	0.955
2.6	0.05	0.1232	0.1188	0.0530	0.053	0.672	0.946
3.9	0.08	0.1806	0.1176	0.0537	0.054	0.681	0.936
5.2	0.11	0.2351	0.1161	0.0543	0.055	0.689	0.925
6.5	0.14	0.2869	0.1145	0.0549	0.055	0.696	0.912
7.8	0.16	0.3358	0.1126	0.0553	0.056	0.701	0.897
9.1	0.19	0.3820	0.1105	0.0557	0.056	0.706	0.880

10.4	0.22	0.4254	0.1081	0.0559	0.056	0.708	0.861
11.7	0.25	0.4660	0.1052	0.0559	0.056	0.708	0.838
13.0	0.27	0.5039	0.1020	0.0557	0.056	0.706	0.813
14.3	0.30	0.5389	0.0985	0.0553	0.056	0.701	0.784
15.6	0.33	0.5710	0.0946	0.0547	0.055	0.693	0.754
16.9	0.36	0.6002	0.0904	0.0539	0.054	0.683	0.720
18.2	0.38	0.6265	0.0860	0.0528	0.053	0.670	0.685
19.5	0.41	0.6499	0.0813	0.0516	0.052	0.654	0.648
20.8	0.44	0.6705	0.0766	0.0503	0.051	0.637	0.610
22.1	0.47	0.6884	0.0717	0.0487	0.049	0.617	0.571
23.4	0.49	0.7039	0.0667	0.0469	0.047	0.594	0.531
24.7	0.52	0.7169	0.0615	0.0448	0.045	0.568	0.490
26.0	0.55	0.7273	0.0562	0.0425	0.043	0.538	0.447
27.3	0.58	0.7353	0.0506	0.0397	0.040	0.504	0.403
28.6	0.60	0.7405	0.0449	0.0367	0.037	0.465	0.357
29.9	0.63	0.7419	0.0390	0.0332	0.033	0.421	0.310
31.2	0.66	0.7374	0.0328	0.0293	0.030	0.372	0.261
32.6	0.69	0.7235	0.0264	0.0251	0.025	0.318	0.211
33.9	0.71	0.6918	0.0200	0.0206	0.021	0.262	0.159
35.2	0.74	0.6216	0.0134	0.0160	0.016	0.203	0.107
36.5	0.77	0.4541	0.0067	0.0114	0.011	0.145	0.054
37.8	0.80	-0.0070	-0.0001	0.0070	0.007	0.088	0.000

PROP RPM = 6000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.1208	0.0505	0.088	0.922	1.385
1.6	0.03	0.0639	0.1198	0.0513	0.089	0.936	1.374
3.1	0.05	0.1249	0.1188	0.0520	0.090	0.949	1.362
4.7	0.08	0.1829	0.1176	0.0527	0.092	0.962	1.348
6.2	0.11	0.2380	0.1162	0.0534	0.093	0.974	1.332
7.8	0.14	0.2902	0.1146	0.0540	0.094	0.985	1.314
9.3	0.16	0.3395	0.1127	0.0545	0.095	0.994	1.293
10.9	0.19	0.3859	0.1107	0.0549	0.095	1.002	1.269
12.4	0.22	0.4295	0.1083	0.0551	0.096	1.006	1.242
14.0	0.25	0.4704	0.1055	0.0552	0.096	1.007	1.210
15.5	0.27	0.5086	0.1023	0.0550	0.096	1.004	1.173
17.1	0.30	0.5441	0.0988	0.0546	0.095	0.997	1.133
18.6	0.33	0.5768	0.0949	0.0540	0.094	0.986	1.089
20.2	0.36	0.6068	0.0908	0.0532	0.092	0.971	1.041
21.7	0.38	0.6342	0.0863	0.0521	0.091	0.951	0.990
23.3	0.41	0.6588	0.0817	0.0509	0.088	0.928	0.937
24.9	0.44	0.6808	0.0769	0.0494	0.086	0.902	0.882
26.4	0.46	0.7001	0.0720	0.0478	0.083	0.873	0.826
28.0	0.49	0.7171	0.0669	0.0459	0.080	0.838	0.768
29.5	0.52	0.7313	0.0617	0.0439	0.076	0.800	0.708
31.1	0.55	0.7430	0.0564	0.0415	0.072	0.757	0.647
32.6	0.57	0.7521	0.0508	0.0387	0.067	0.707	0.582
34.2	0.60	0.7579	0.0450	0.0357	0.062	0.651	0.516
35.7	0.63	0.7593	0.0390	0.0323	0.056	0.589	0.447
37.3	0.66	0.7548	0.0327	0.0285	0.049	0.519	0.375
38.8	0.68	0.7406	0.0264	0.0243	0.042	0.444	0.303
40.4	0.71	0.7079	0.0199	0.0200	0.035	0.365	0.228
41.9	0.74	0.6370	0.0134	0.0155	0.027	0.282	0.153
43.5	0.77	0.4670	0.0067	0.0110	0.019	0.201	0.077
45.0	0.79	-0.0032	0.0000	0.0069	0.012	0.126	0.000

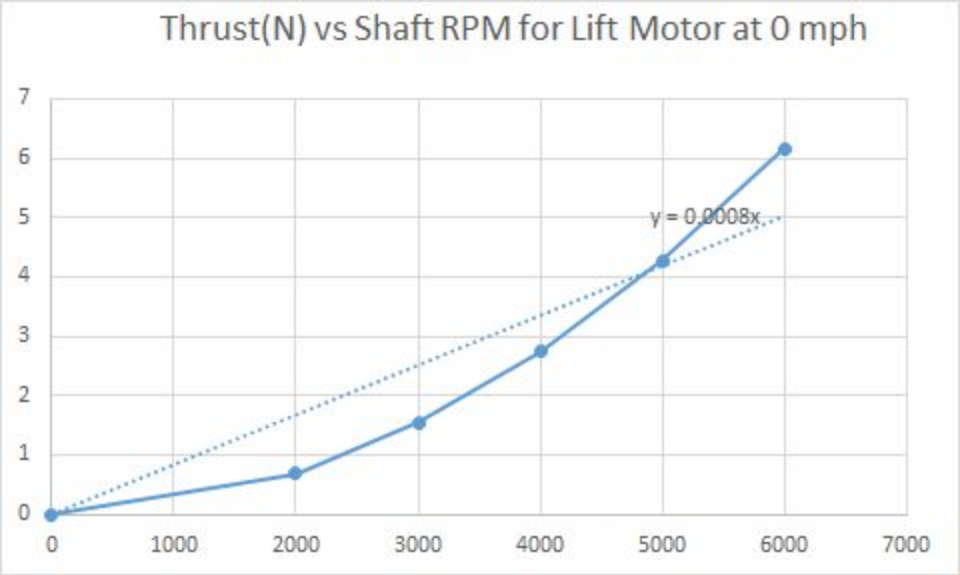


Figure A2.3.0: Plot of Thrust vs Shaft RPM for Lift Propeller

Yaw Propeller Data

====== PERFORMANCE DATA (versus advance ratio and MPH) ======

DEFINITIONS:
J=V/nD (advance ratio)
Ct=T/(rho * n**2 * D**4) (thrust coef.)
Cp=P/(rho * n**3 * D**5) (power coef.)
Pe=Ct*J/Cp (efficiency)
V (model speed in MPH)

PROP RPM = 1000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.0872	0.0318	0.000	0.003	0.007
0.1	0.02	0.0528	0.0855	0.0318	0.000	0.003	0.007
0.3	0.04	0.1032	0.0836	0.0319	0.000	0.003	0.006
0.4	0.06	0.1512	0.0818	0.0319	0.000	0.003	0.006
0.5	0.08	0.1967	0.0798	0.0319	0.000	0.003	0.006
0.7	0.10	0.2398	0.0777	0.0319	0.000	0.003	0.006
0.8	0.12	0.2804	0.0755	0.0318	0.000	0.003	0.006
0.9	0.14	0.3186	0.0733	0.0317	0.000	0.003	0.006
1.0	0.16	0.3542	0.0709	0.0315	0.000	0.003	0.005
1.2	0.18	0.3874	0.0683	0.0312	0.000	0.003	0.005
1.3	0.20	0.4181	0.0656	0.0309	0.000	0.003	0.005
1.4	0.22	0.4462	0.0629	0.0305	0.000	0.003	0.005
1.6	0.24	0.4720	0.0600	0.0300	0.000	0.003	0.005
1.7	0.26	0.4956	0.0571	0.0294	0.000	0.003	0.004
1.8	0.28	0.5170	0.0541	0.0288	0.000	0.002	0.004
2.0	0.30	0.5361	0.0510	0.0280	0.000	0.002	0.004
2.1	0.31	0.5533	0.0478	0.0272	0.000	0.002	0.004
2.2	0.33	0.5680	0.0446	0.0262	0.000	0.002	0.003
2.3	0.35	0.5806	0.0412	0.0251	0.000	0.002	0.003
2.5	0.37	0.5906	0.0378	0.0239	0.000	0.002	0.003
2.6	0.39	0.5987	0.0344	0.0226	0.000	0.002	0.003
2.7	0.41	0.6038	0.0308	0.0211	0.000	0.002	0.002
2.9	0.43	0.6041	0.0272	0.0195	0.000	0.002	0.002
3.0	0.45	0.5980	0.0236	0.0178	0.000	0.002	0.002
3.1	0.47	0.5834	0.0198	0.0161	0.000	0.001	0.002
3.3	0.49	0.5561	0.0160	0.0142	0.000	0.001	0.001
3.4	0.51	0.5088	0.0121	0.0122	0.000	0.001	0.001
3.5	0.53	0.4277	0.0082	0.0101	0.000	0.001	0.001
3.7	0.55	0.2801	0.0041	0.0081	0.000	0.001	0.000
3.8	0.57	-0.0033	0.0000	0.0063	0.000	0.001	0.000

PROP RPM = 2000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.0882	0.0317	0.000	0.011	0.027
0.3	0.02	0.0532	0.0865	0.0317	0.000	0.011	0.026
0.5	0.04	0.1039	0.0846	0.0318	0.000	0.011	0.026
0.8	0.06	0.1522	0.0827	0.0318	0.000	0.011	0.025
1.0	0.08	0.1980	0.0807	0.0318	0.000	0.011	0.025
1.3	0.10	0.2414	0.0787	0.0318	0.000	0.011	0.024
1.6	0.12	0.2822	0.0765	0.0317	0.000	0.011	0.023
1.8	0.14	0.3205	0.0742	0.0316	0.000	0.011	0.023
2.1	0.16	0.3564	0.0718	0.0315	0.000	0.011	0.022
2.3	0.18	0.3897	0.0693	0.0312	0.000	0.011	0.021
2.6	0.20	0.4204	0.0666	0.0309	0.000	0.011	0.020
2.8	0.21	0.4486	0.0638	0.0305	0.000	0.010	0.020
3.1	0.23	0.4744	0.0609	0.0301	0.000	0.010	0.019
3.4	0.25	0.4979	0.0579	0.0295	0.000	0.010	0.018
3.6	0.27	0.5193	0.0549	0.0289	0.000	0.010	0.017
3.9	0.29	0.5384	0.0518	0.0282	0.000	0.010	0.016
4.1	0.31	0.5555	0.0486	0.0273	0.000	0.009	0.015
4.4	0.33	0.5703	0.0453	0.0264	0.000	0.009	0.014
4.7	0.35	0.5828	0.0420	0.0253	0.000	0.009	0.013
4.9	0.37	0.5929	0.0385	0.0241	0.000	0.008	0.012
5.2	0.39	0.6007	0.0350	0.0228	0.000	0.008	0.011
5.4	0.41	0.6060	0.0314	0.0213	0.000	0.007	0.010
5.7	0.43	0.6067	0.0277	0.0196	0.000	0.007	0.008
6.0	0.45	0.6010	0.0240	0.0179	0.000	0.006	0.007
6.2	0.47	0.5867	0.0202	0.0161	0.000	0.005	0.006
6.5	0.49	0.5599	0.0163	0.0142	0.000	0.005	0.005
6.7	0.51	0.5135	0.0123	0.0122	0.000	0.004	0.004
7.0	0.53	0.4333	0.0083	0.0101	0.000	0.003	0.003
7.2	0.55	0.2849	0.0042	0.0080	0.000	0.003	0.001
7.5	0.57	-0.0010	0.0000	0.0061	0.000	0.002	0.000

PROP RPM = 3000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.0884	0.0316	0.001	0.024	0.061
0.4	0.02	0.0535	0.0866	0.0317	0.001	0.024	0.060
0.8	0.04	0.1045	0.0848	0.0317	0.001	0.024	0.058
1.2	0.06	0.1530	0.0829	0.0318	0.001	0.024	0.057
1.6	0.08	0.1991	0.0809	0.0318	0.001	0.024	0.056
1.9	0.10	0.2426	0.0788	0.0318	0.001	0.024	0.054
2.3	0.12	0.2836	0.0767	0.0317	0.001	0.024	0.053
2.7	0.14	0.3220	0.0744	0.0316	0.001	0.024	0.051
3.1	0.16	0.3580	0.0720	0.0315	0.001	0.024	0.050
3.5	0.18	0.3913	0.0694	0.0312	0.001	0.024	0.048
3.9	0.20	0.4221	0.0667	0.0309	0.001	0.024	0.046
4.3	0.22	0.4504	0.0639	0.0305	0.001	0.023	0.044
4.7	0.23	0.4762	0.0610	0.0301	0.001	0.023	0.042
5.1	0.25	0.4996	0.0581	0.0296	0.001	0.023	0.040
5.4	0.27	0.5210	0.0551	0.0289	0.001	0.022	0.038
5.8	0.29	0.5400	0.0519	0.0282	0.001	0.022	0.036
6.2	0.31	0.5571	0.0487	0.0274	0.001	0.021	0.034
6.6	0.33	0.5719	0.0455	0.0264	0.001	0.020	0.031
7.0	0.35	0.5843	0.0421	0.0254	0.001	0.019	0.029
7.4	0.37	0.5943	0.0386	0.0241	0.001	0.019	0.027
7.8	0.39	0.6019	0.0351	0.0228	0.001	0.017	0.024
8.2	0.41	0.6071	0.0314	0.0212	0.001	0.016	0.022
8.6	0.43	0.6075	0.0277	0.0196	0.001	0.015	0.019
8.9	0.45	0.6013	0.0239	0.0179	0.001	0.014	0.016
9.3	0.47	0.5864	0.0200	0.0160	0.001	0.012	0.014
9.7	0.49	0.5590	0.0161	0.0141	0.001	0.011	0.011
10.1	0.51	0.5117	0.0122	0.0121	0.000	0.009	0.008
10.5	0.53	0.4303	0.0082	0.0100	0.000	0.008	0.006
10.9	0.55	0.2826	0.0041	0.0080	0.000	0.006	0.003
11.3	0.57	-0.0012	0.0000	0.0061	0.000	0.005	0.000

PROP RPM = 4000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.0885	0.0316	0.003	0.043	0.108
0.5	0.02	0.0535	0.0867	0.0317	0.003	0.043	0.106
1.0	0.04	0.1045	0.0849	0.0318	0.003	0.043	0.104
1.6	0.06	0.1530	0.0830	0.0318	0.003	0.043	0.102
2.1	0.08	0.1991	0.0810	0.0318	0.003	0.043	0.099
2.6	0.10	0.2426	0.0789	0.0318	0.003	0.043	0.097
3.1	0.12	0.2836	0.0767	0.0317	0.003	0.043	0.094
3.6	0.14	0.3221	0.0745	0.0316	0.003	0.043	0.091
4.1	0.16	0.3580	0.0721	0.0315	0.003	0.043	0.088
4.7	0.18	0.3914	0.0695	0.0312	0.003	0.043	0.085
5.2	0.20	0.4222	0.0668	0.0309	0.003	0.042	0.082
5.7	0.22	0.4505	0.0640	0.0305	0.003	0.042	0.078
6.2	0.23	0.4763	0.0611	0.0301	0.003	0.041	0.075
6.7	0.25	0.4997	0.0581	0.0296	0.003	0.040	0.071
7.3	0.27	0.5211	0.0551	0.0289	0.003	0.039	0.067
7.8	0.29	0.5401	0.0520	0.0282	0.002	0.038	0.064
8.3	0.31	0.5572	0.0488	0.0274	0.002	0.037	0.060
8.8	0.33	0.5720	0.0455	0.0264	0.002	0.036	0.056
9.3	0.35	0.5844	0.0421	0.0254	0.002	0.035	0.052
9.8	0.37	0.5944	0.0386	0.0241	0.002	0.033	0.047
10.4	0.39	0.6019	0.0351	0.0228	0.002	0.031	0.043
10.9	0.41	0.6071	0.0314	0.0213	0.002	0.029	0.038
11.4	0.43	0.6076	0.0277	0.0196	0.002	0.027	0.034
11.9	0.45	0.6014	0.0239	0.0179	0.002	0.024	0.029
12.4	0.47	0.5866	0.0201	0.0161	0.001	0.022	0.025
13.0	0.49	0.5593	0.0162	0.0141	0.001	0.019	0.020
13.5	0.51	0.5122	0.0122	0.0121	0.001	0.017	0.015
14.0	0.53	0.4306	0.0082	0.0100	0.001	0.014	0.010
14.5	0.55	0.2848	0.0042	0.0080	0.001	0.011	0.005
15.0	0.57	-0.0007	0.0000	0.0061	0.001	0.008	0.000

PROP RPM = 5000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.0886	0.0317	0.005	0.067	0.169
0.6	0.02	0.0535	0.0868	0.0317	0.005	0.068	0.166
1.3	0.04	0.1045	0.0850	0.0318	0.005	0.068	0.163
1.9	0.06	0.1530	0.0831	0.0318	0.005	0.068	0.159
2.6	0.08	0.1991	0.0811	0.0319	0.005	0.068	0.155
3.2	0.10	0.2426	0.0790	0.0318	0.005	0.068	0.151
3.9	0.12	0.2836	0.0768	0.0318	0.005	0.068	0.147
4.5	0.14	0.3221	0.0745	0.0317	0.005	0.067	0.143
5.2	0.16	0.3580	0.0721	0.0315	0.005	0.067	0.138
5.8	0.18	0.3914	0.0696	0.0313	0.005	0.067	0.133
6.5	0.20	0.4223	0.0669	0.0310	0.005	0.066	0.128
7.1	0.22	0.4505	0.0640	0.0306	0.005	0.065	0.122
7.8	0.23	0.4763	0.0612	0.0301	0.005	0.064	0.117
8.4	0.25	0.4998	0.0582	0.0296	0.005	0.063	0.111
9.1	0.27	0.5211	0.0552	0.0290	0.005	0.062	0.105
9.7	0.29	0.5402	0.0520	0.0282	0.005	0.060	0.100
10.4	0.31	0.5573	0.0488	0.0274	0.005	0.058	0.093
11.0	0.33	0.5721	0.0456	0.0265	0.004	0.056	0.087
11.7	0.35	0.5845	0.0422	0.0254	0.004	0.054	0.081
12.3	0.37	0.5946	0.0387	0.0242	0.004	0.051	0.074
13.0	0.39	0.6020	0.0351	0.0228	0.004	0.049	0.067
13.6	0.41	0.6072	0.0315	0.0213	0.004	0.045	0.060
14.3	0.43	0.6077	0.0278	0.0196	0.003	0.042	0.053
14.9	0.45	0.6017	0.0240	0.0179	0.003	0.038	0.046
15.5	0.47	0.5869	0.0201	0.0161	0.003	0.034	0.038
16.2	0.49	0.5597	0.0162	0.0142	0.002	0.030	0.031
16.8	0.51	0.5125	0.0122	0.0121	0.002	0.026	0.023
17.5	0.53	0.4312	0.0082	0.0100	0.002	0.021	0.016
18.1	0.55	0.2835	0.0041	0.0080	0.001	0.017	0.008
18.8	0.57	-0.0009	0.0000	0.0061	0.001	0.013	0.000

PROP RPM = 6000

V (mph)	J (Adv Ratio)	Pe	Ct	Cp	PWR (Hp)	Torque (In-Lbf)	Thrust (Lbf)
0.0	0.00	0.0000	0.0887	0.0317	0.009	0.097	0.244

0.8	0.02	0.0535	0.0870	0.0318	0.009	0.098	0.239
1.6	0.04	0.1045	0.0851	0.0319	0.009	0.098	0.234
2.3	0.06	0.1530	0.0832	0.0319	0.009	0.098	0.229
3.1	0.08	0.1991	0.0812	0.0319	0.009	0.098	0.224
3.9	0.10	0.2426	0.0791	0.0319	0.009	0.098	0.218
4.7	0.12	0.2836	0.0769	0.0318	0.009	0.098	0.212
5.4	0.14	0.3221	0.0747	0.0317	0.009	0.097	0.206
6.2	0.16	0.3580	0.0722	0.0315	0.009	0.097	0.199
7.0	0.18	0.3915	0.0697	0.0313	0.009	0.096	0.192
7.8	0.20	0.4223	0.0670	0.0310	0.009	0.095	0.184
8.6	0.21	0.4506	0.0641	0.0306	0.009	0.094	0.177
9.3	0.23	0.4764	0.0613	0.0302	0.009	0.093	0.169
10.1	0.25	0.4998	0.0583	0.0296	0.009	0.091	0.160
10.9	0.27	0.5212	0.0552	0.0290	0.008	0.089	0.152
11.7	0.29	0.5402	0.0521	0.0283	0.008	0.087	0.144
12.4	0.31	0.5573	0.0489	0.0274	0.008	0.084	0.135
13.2	0.33	0.5721	0.0456	0.0265	0.008	0.081	0.126
14.0	0.35	0.5846	0.0422	0.0254	0.007	0.078	0.116
14.8	0.37	0.5946	0.0387	0.0242	0.007	0.074	0.107
15.5	0.39	0.6021	0.0352	0.0228	0.007	0.070	0.097
16.3	0.41	0.6074	0.0315	0.0213	0.006	0.065	0.087
17.1	0.43	0.6079	0.0278	0.0197	0.006	0.060	0.077
17.9	0.45	0.6020	0.0240	0.0179	0.005	0.055	0.066
18.7	0.47	0.5874	0.0202	0.0161	0.005	0.049	0.056
19.4	0.49	0.5602	0.0163	0.0142	0.004	0.043	0.045
20.2	0.51	0.5132	0.0123	0.0122	0.004	0.037	0.034
21.0	0.53	0.4319	0.0082	0.0101	0.003	0.031	0.023
21.8	0.55	0.2842	0.0041	0.0080	0.002	0.024	0.011
22.5	0.57	-0.0008	0.0000	0.0061	0.002	0.019	0.000
125.8	0.56	-0.0136	-0.0004	0.0151	0.800	1.483	-0.032

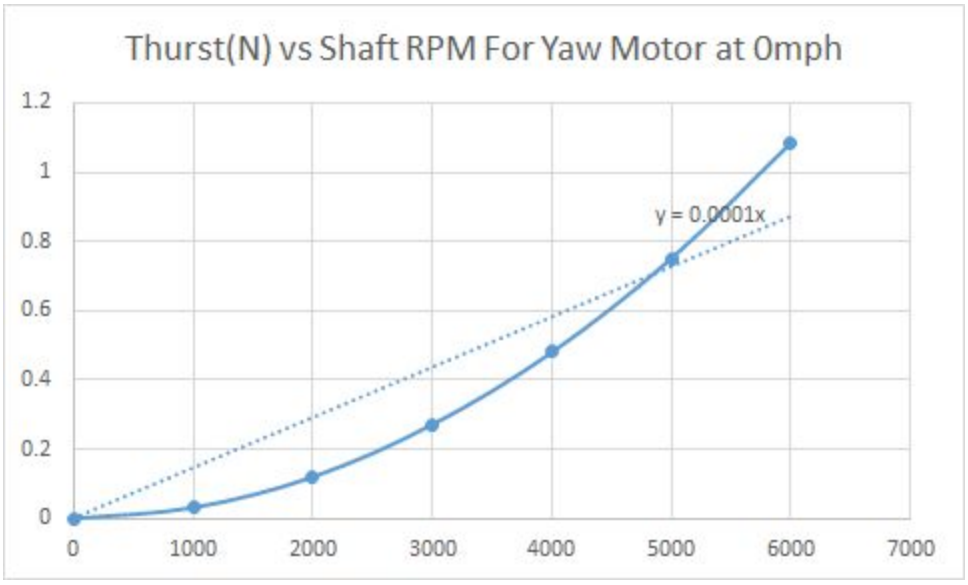


Figure A2.3.1: Plot of Thrust vs Shaft RPM for Yaw Propeller

A2.4 Simulation, Results

Our simulink model is as follows:

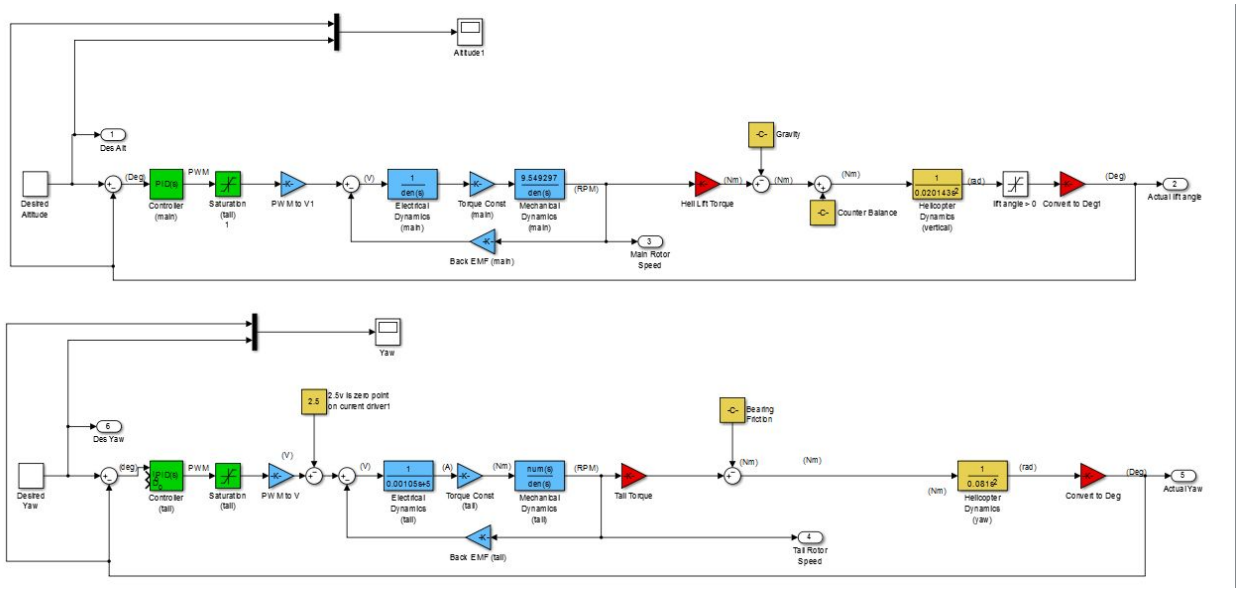


Figure A2.4.0: Simulink Model of Robot

Results:

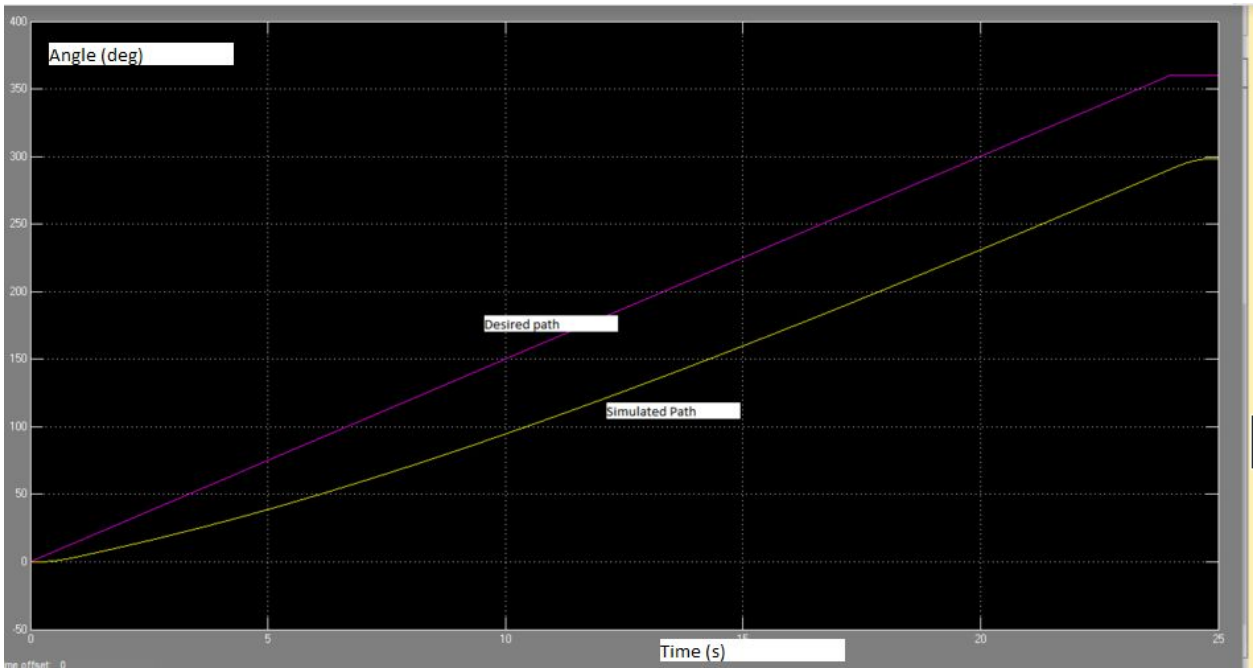


Figure A2.4.1: Results of Yaw Motor Simulink Simulation

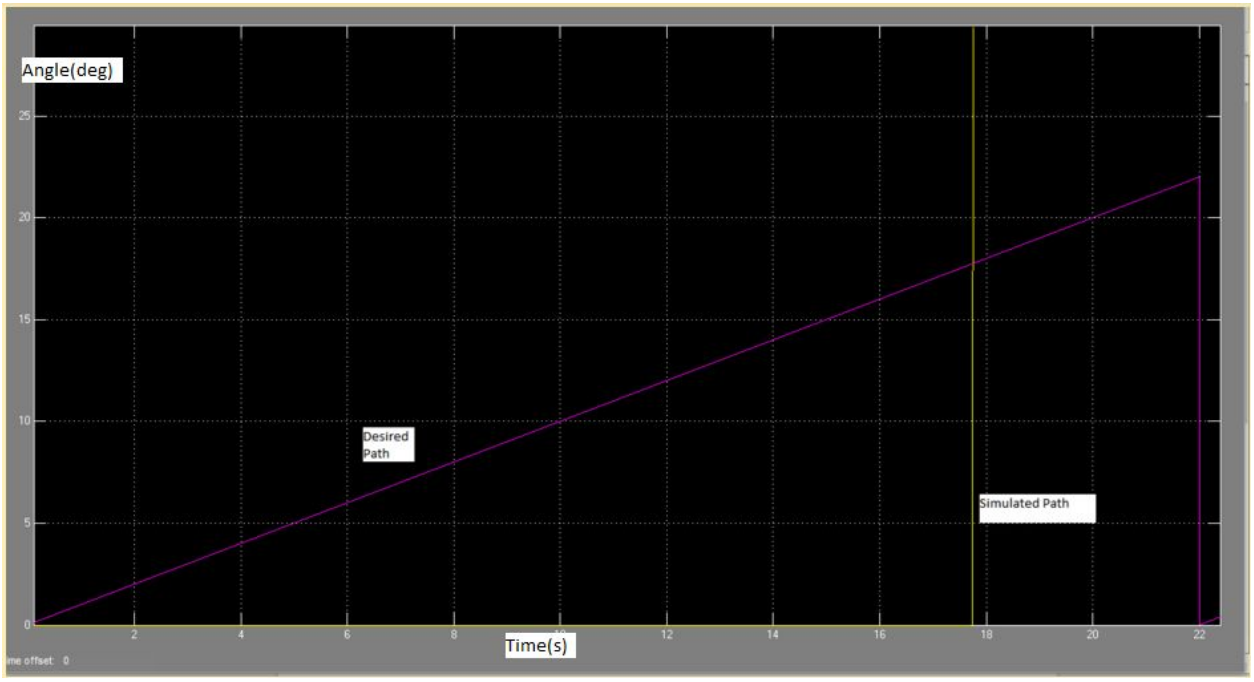


Figure A2.4.2: Results of Lift Motor Simulink Simulation

A3.0 : Self Built Motor Dynamics

A3.1 : Motor Lift Force Calculations

The lift force of a propellor is given by

$$F=\omega^2L^2l\rho\sin^2\phi$$

in which ω is the angular velocity, L is the length of the helix, l is the width of the helix (both in meters), ρis the air density at normal conditions, and ϕ is the angular deviation of the helix related to the rotating axis. So a 4-helix propeller would lead to $F=4\omega^2L^2l\rho\sin^2\phi$ and so on.

We estimate that our motor will weigh roughly 315g. This means that to lift the motor by itself without counterweighting we need to generate a lift force of

$$F_{lift} = mg = (0.315\ kg) * (9.8\ m/s^2) = 3.087\ N$$

So if we have a 4-blade propeller working:
Where
$\rho=1.293kg/m^3$
$\phi=1^\circ$
$l=10^{-2}m$
$L=5\times10^{-2}m$. T

Then the RPM needed to lift up the motor on its own would be :

$$3.087N=4\times(2\pi\times\omega/60)^2\times(5\times10^{-2})^2x1.293\times(\sin1^\circ)^2$$

$$\therefore\omega = 8454\ RPM$$

Given that the highest RPM we achieve at no load is 5400 RPM, we need to use counter-weighting to our advantage. That is, we may only have to effectively generate a small lift force.

A3.2 : Rotational Force Calculations

Two things that we know is that

1) Our stator acts like a solenoid. Such that that force generated by an induced magnetic field is :

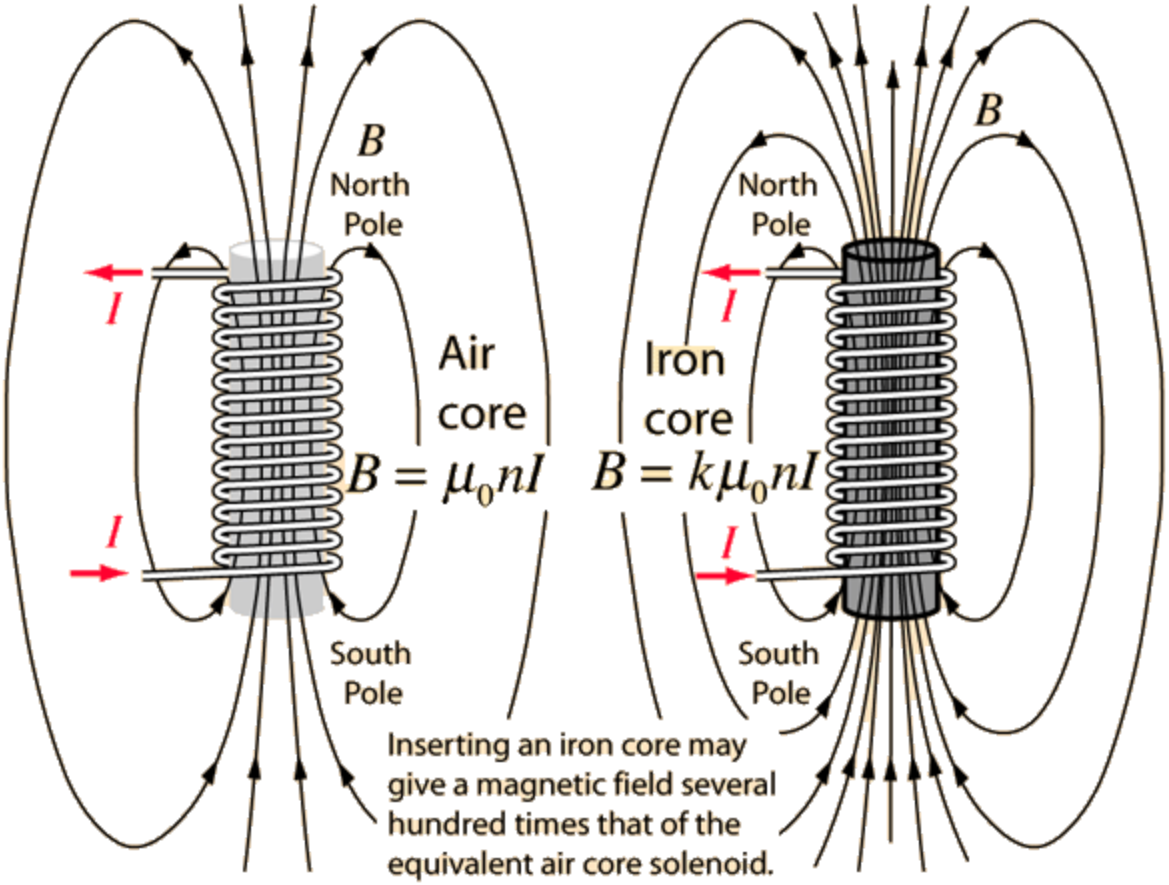


Figure A3.2.0 Ideal Solenoid [6]

$$F = ((NI)^2 * \mu * A)/g$$

Where:
*μ=5000*μ*0 (For iron)
F is the force in Newtons
N is the number of turns
I is the current in Amps
A is the area in length units squared
g is the length of the gap between the solenoid and a piece of a ferromagnetic material (in our case this is our magnet)

2) We also know that centripetal force can be expressed in terms of angular velocity as

$$F_C = mrw^2$$

If we make the simplification that if the motor is commutated exactly as needed such that a constant force is exerted by each of the solenoids we can then equate our centripetal force to the force generated by our solenoid.

Setting our values
I = 1A
Because our estimated RPM required is 8454 RPM

g = 0.1mm
This is as measured.

A = 0.0000954 m^2
As calculated from our dimensions

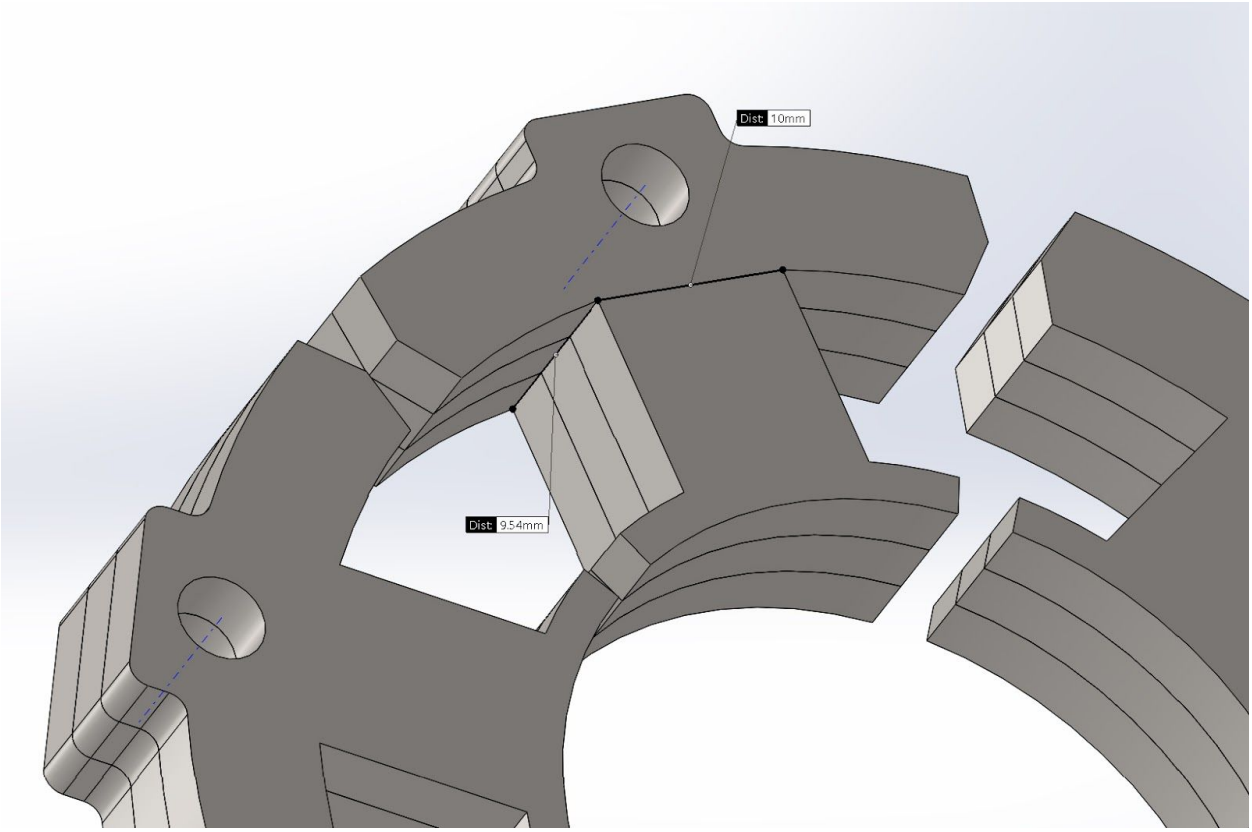


Figure A3.2.1 dimension of the stacked cores

m = 0.100 kg
As estimated

r = 14.5 mm
As measured

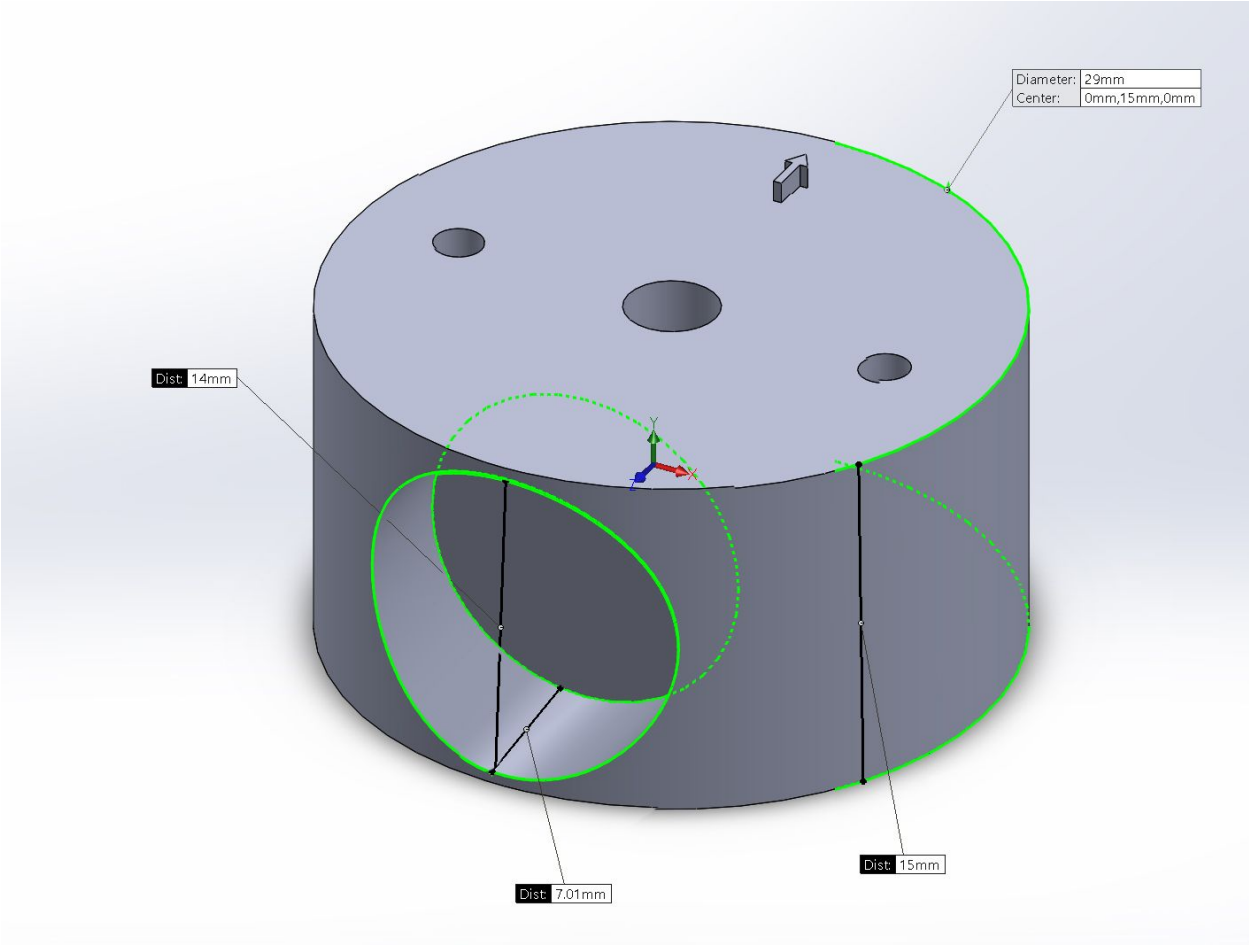


Figure A3.2.2 Rotor Designed

We are then able to calculate N as

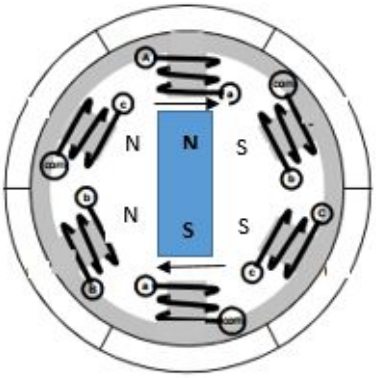
N = 500

This calculation might be a little high for our design. To reduce the size of the motor, we really do not need to lift the entire motor. We can use counter weight on the other end of the arm, in which will reduce the effective weight that the motor needs to lift. So at the end we settled for 200 windings for each pole.

A3.3: 3 Phase Motor Commutation with Hall Effect Sensor Values

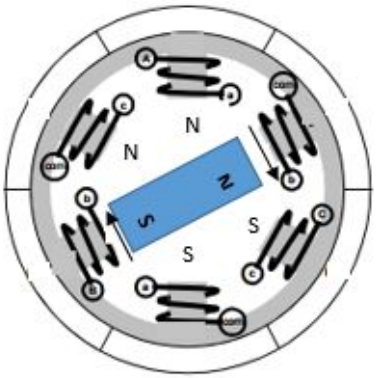
Phase 1: Hall ABC = 100

A High ON, B Low ON



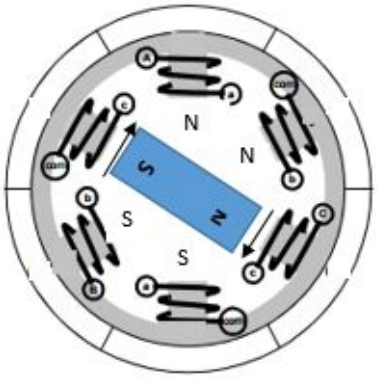
Phase 2: Hall ABC =110

C High ON, B Low ON



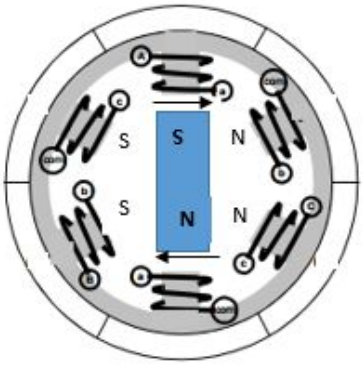
Phase 3: Hall ABC = 010

C High ON, A Low ON



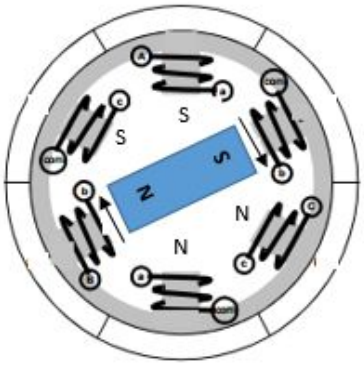
Phase 4: Hall ABC = 011

B High ON, A Low ON



Phase 5: Hall ABC = 001

B High ON, C Low ON



Phase 6: Hall ABC = 101

A High ON, C Low ON

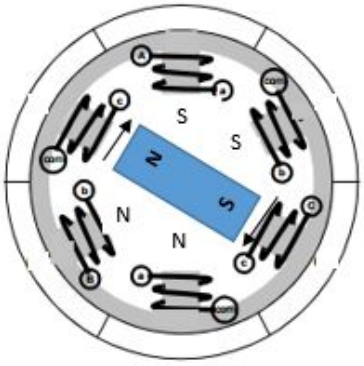


Figure A3.3: 3 Phase Motor Commutation [5]

A3.4 Mechanical Drawing of the motor

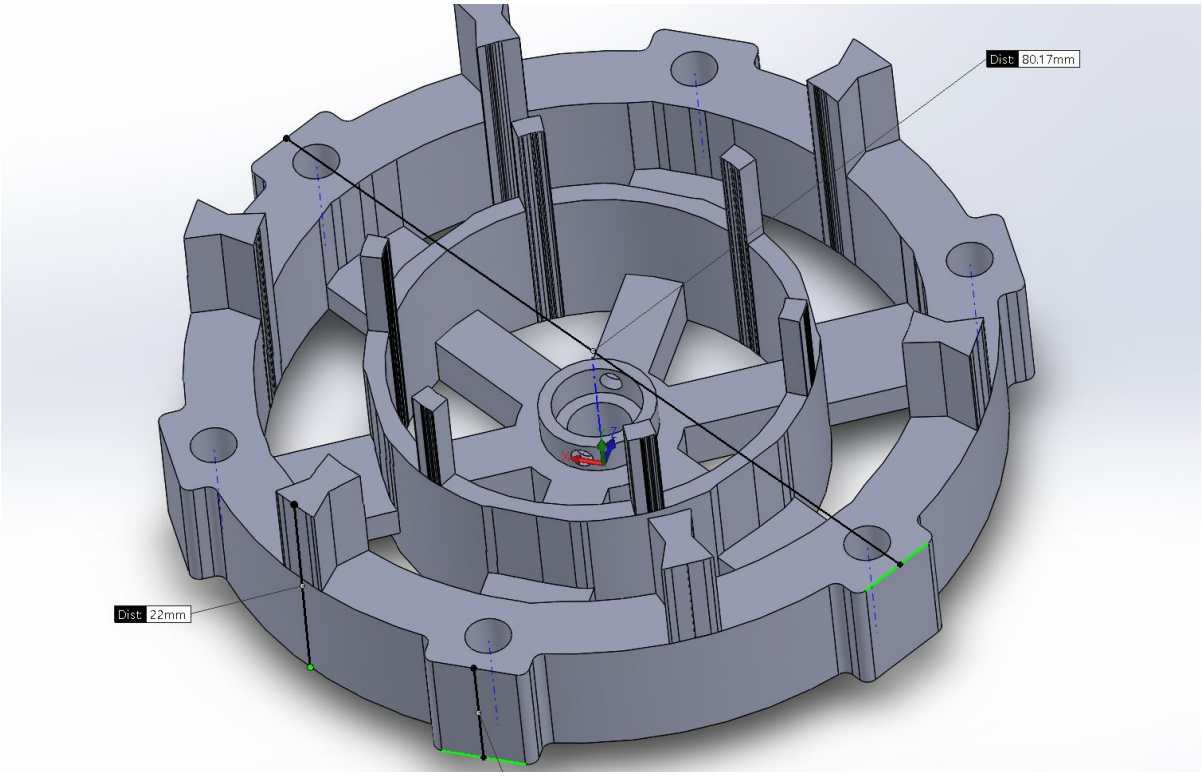


Figure A3.4.0: Stator bottom plate

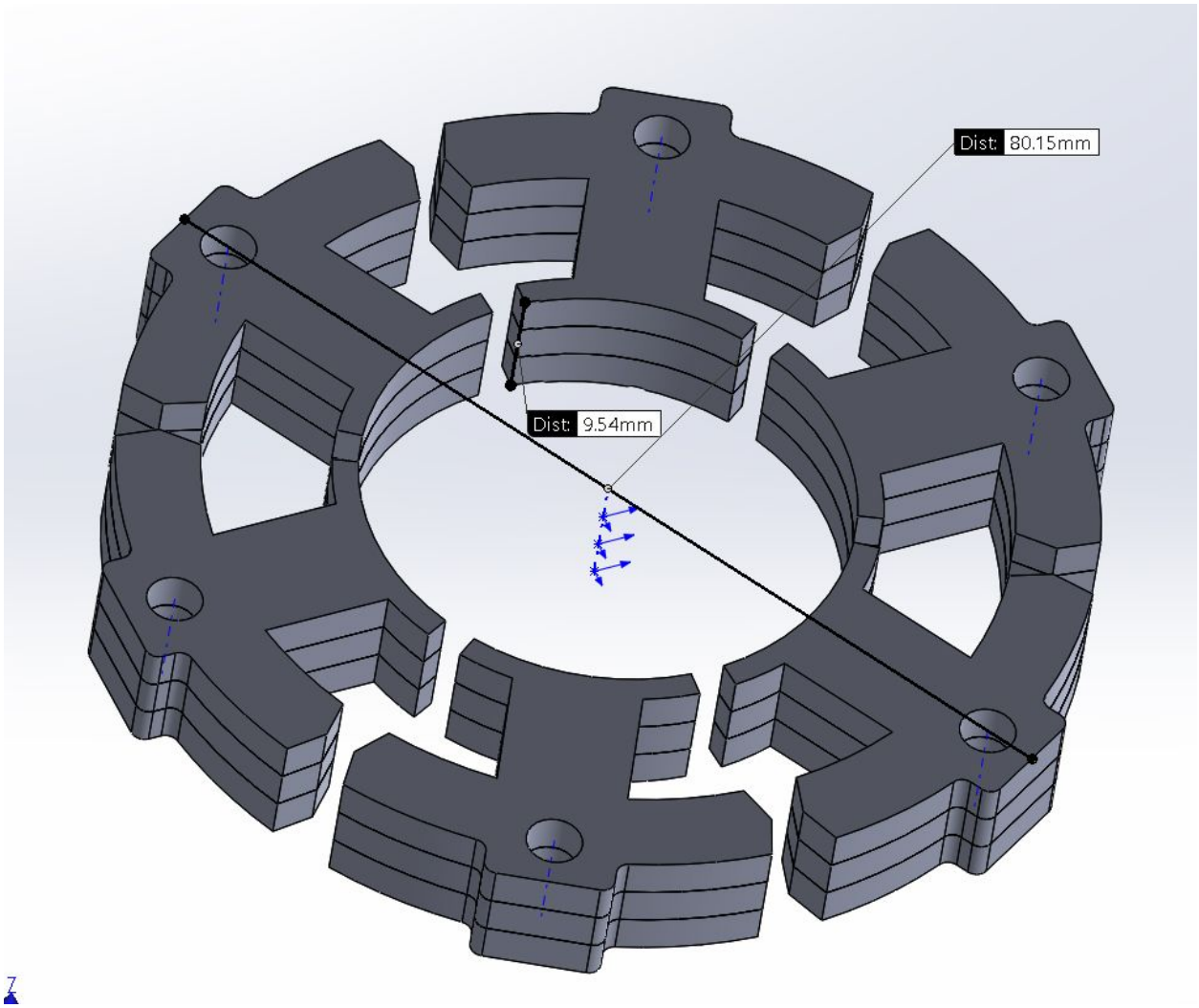


Figure A3.4.1: Stacked layout of all the Iron cores

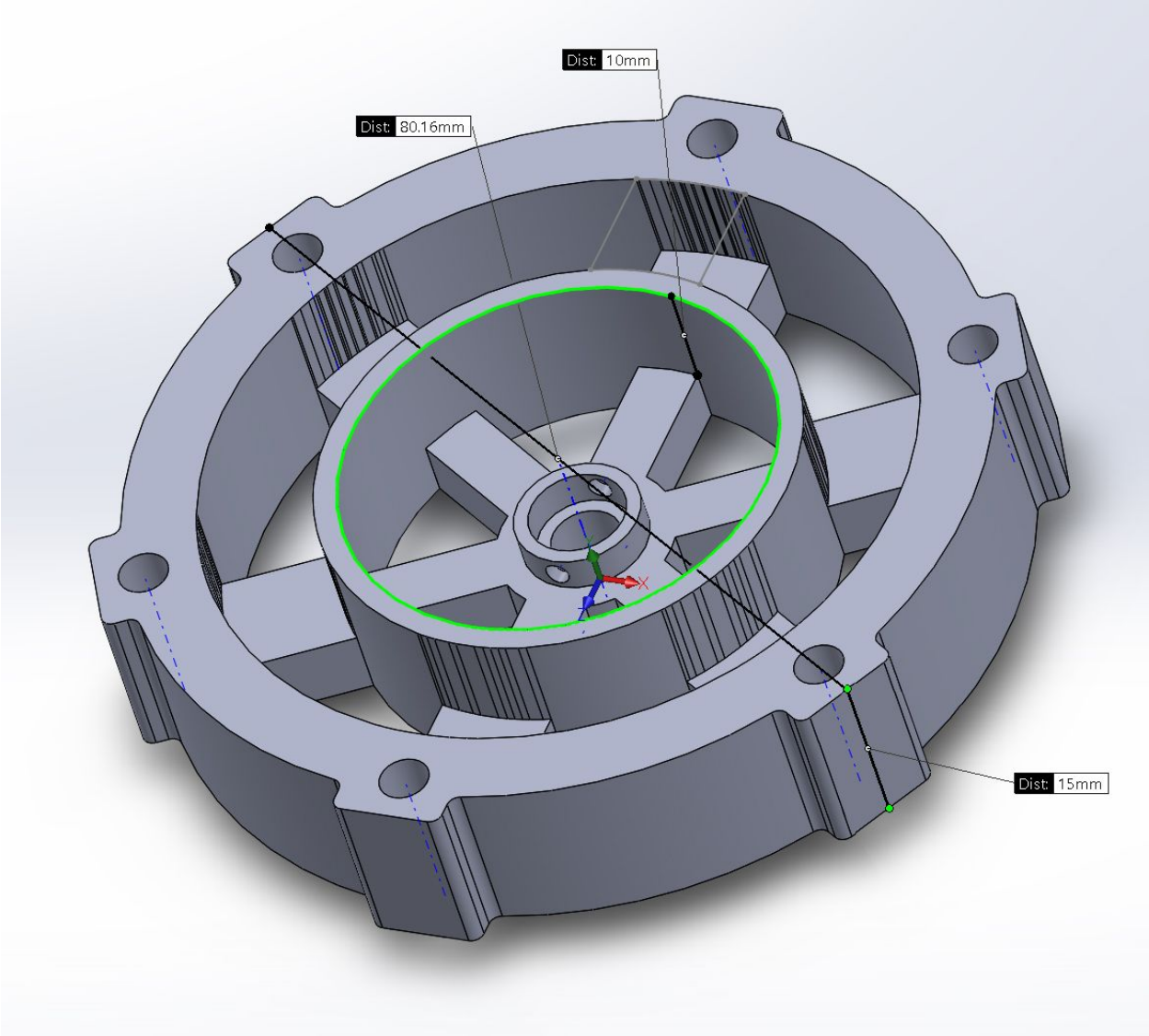


Figure A3.4.3: Stator top

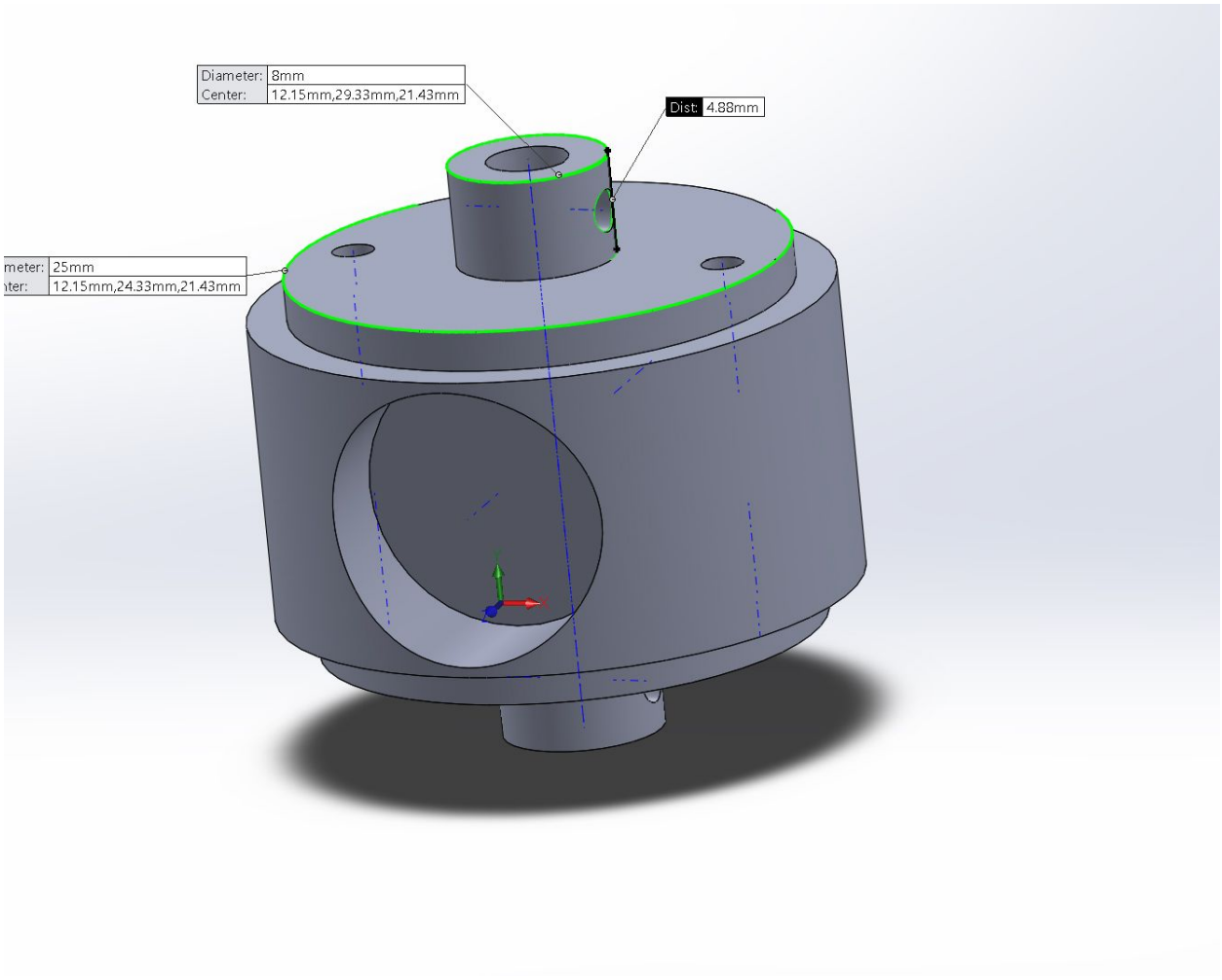


Figure A3.4.4: Rotor with two attached rotor holder

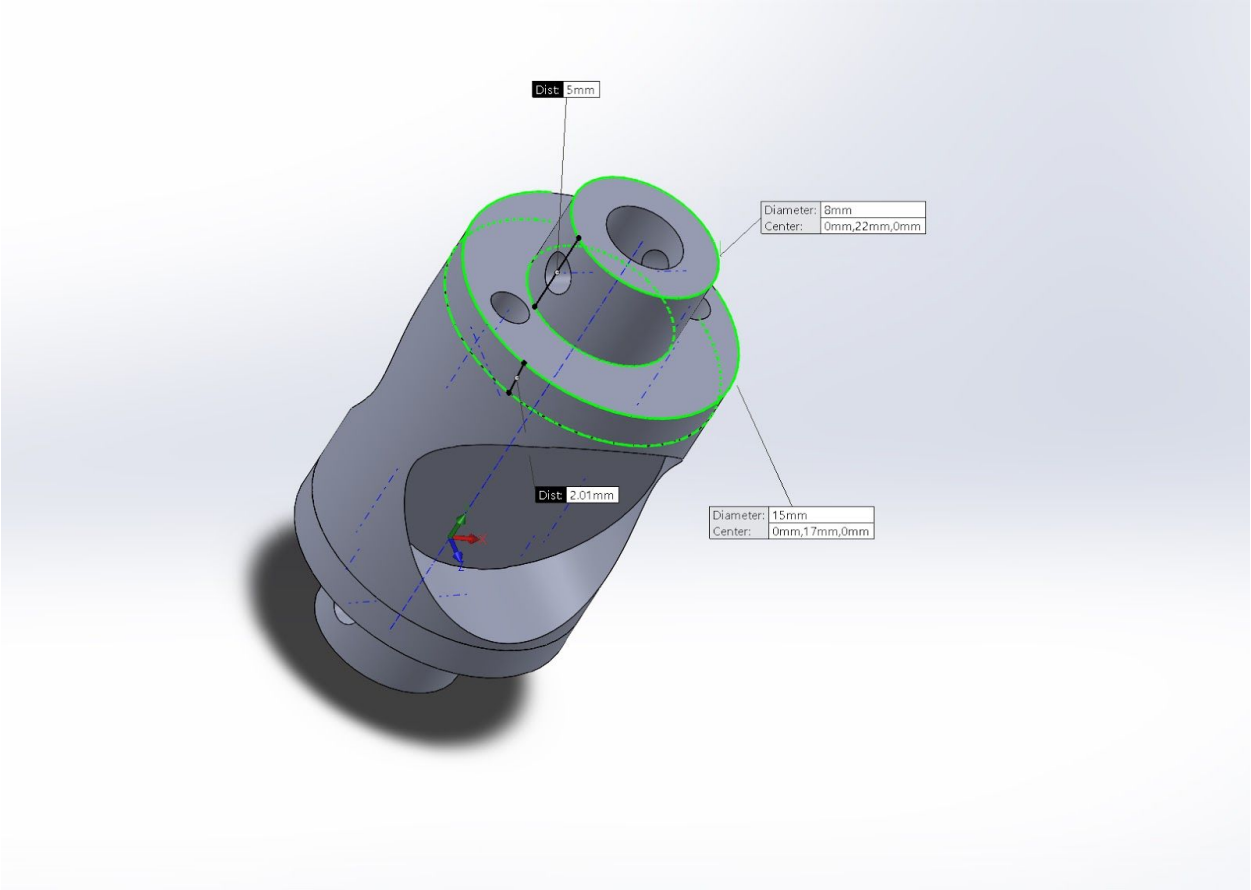


Figure A3.4.5: Secondary holder with rotor holder attached on top and bottom

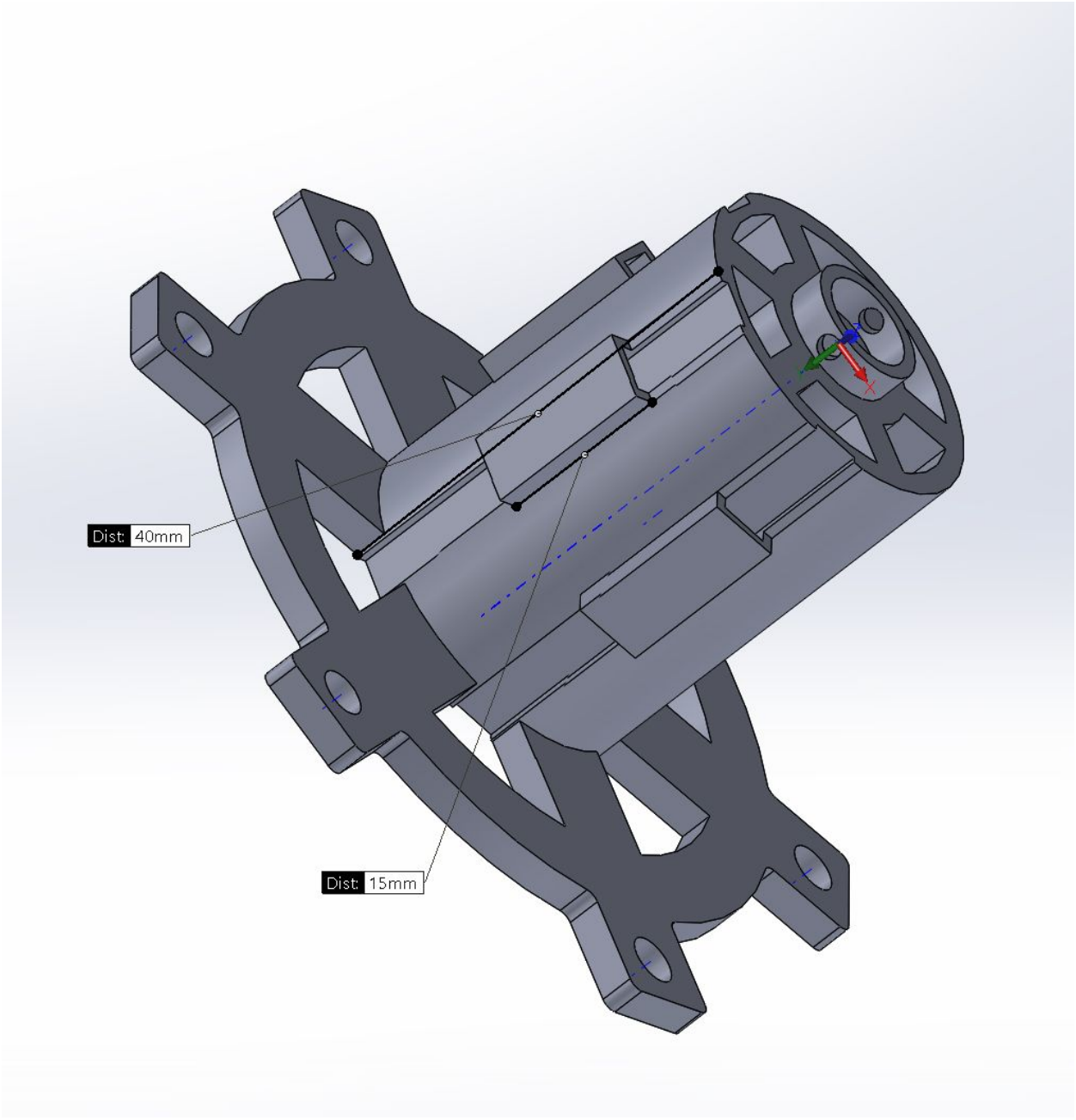


Figure A3.4.6: Sensor cage, with three grooves and covers to hold the hall sensors in place

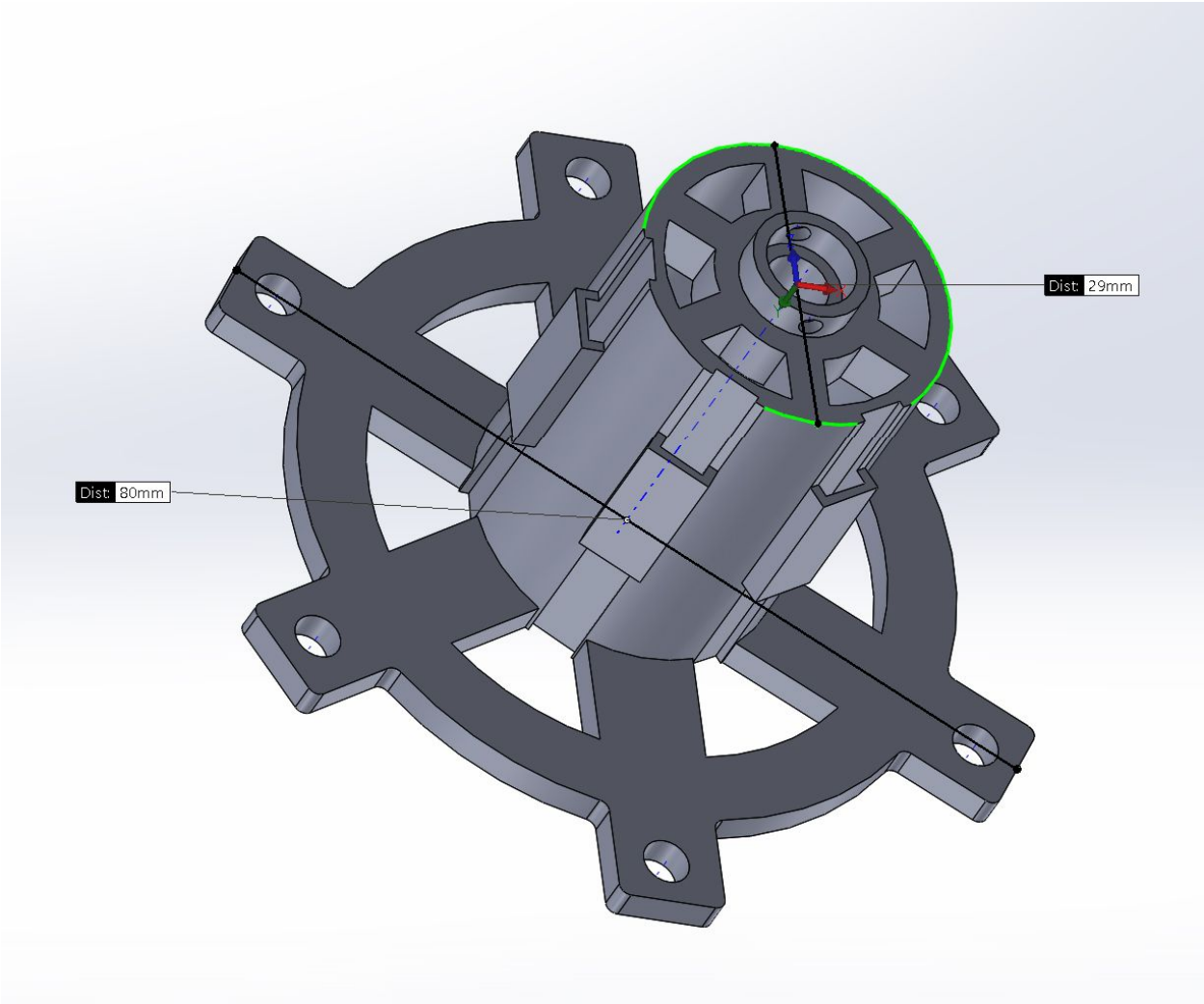


Figure A3.4.7: Sensor cage from the top view. The top opening allows a bearing to be locked in place

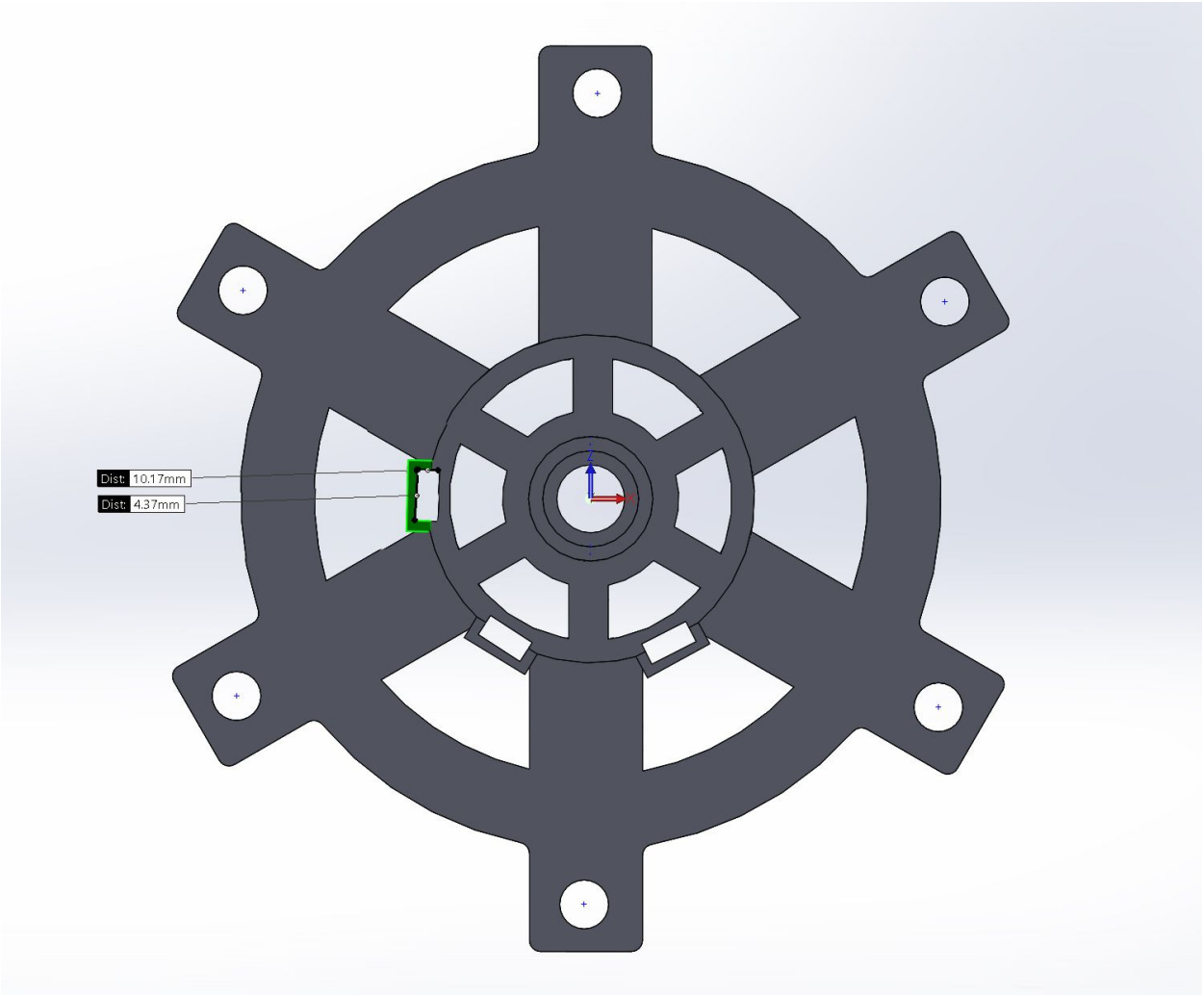


Figure A3.4.8: Sensor cage, the three hall sensors are located 60 degrees apart in the small opening.

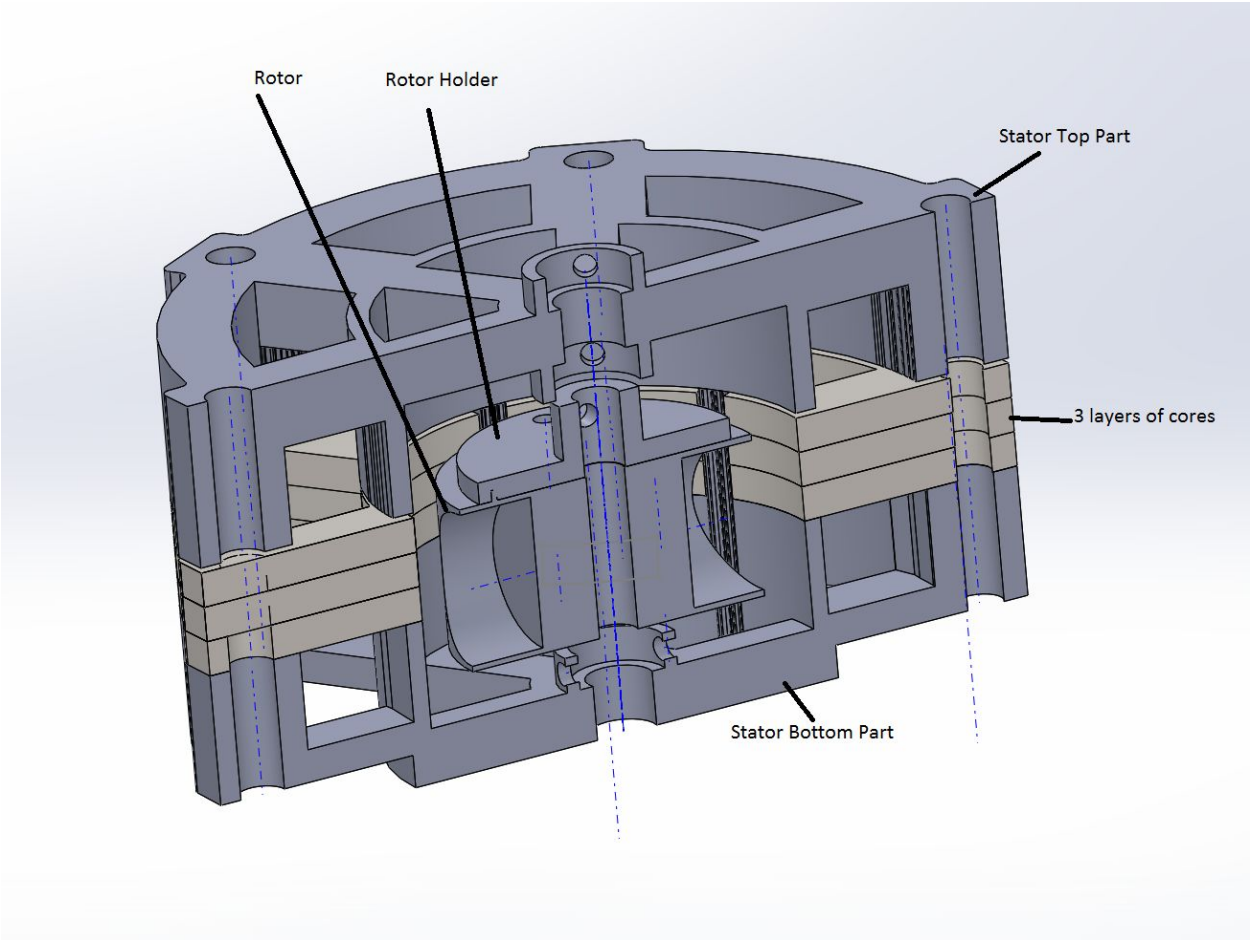


Figure A3.4.9: Section view of the assembled stator

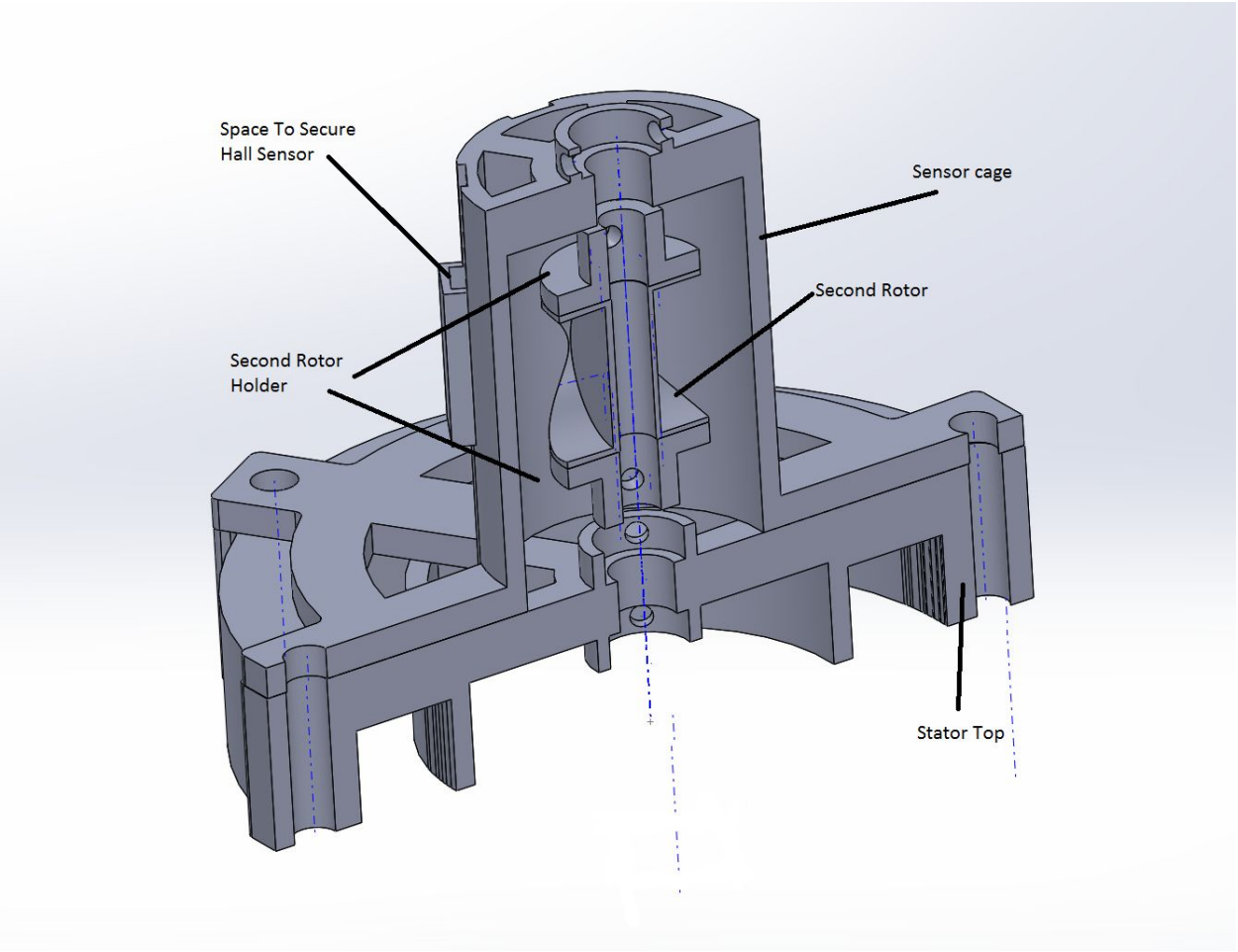


Figure A3.4.10: Section view of the assembled sensor cage

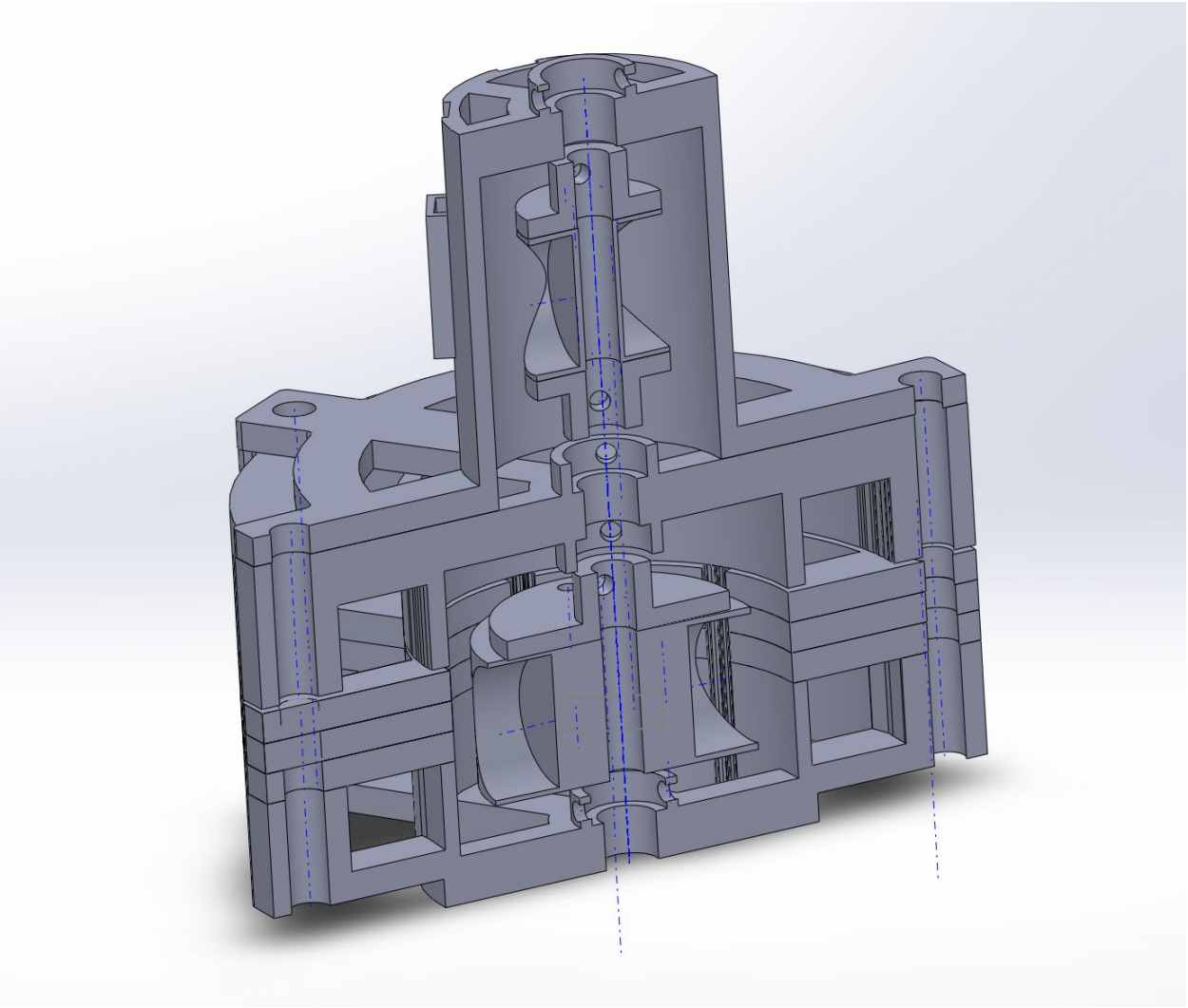


Figure A3.4.11: Section view of the complete motor mechanical design

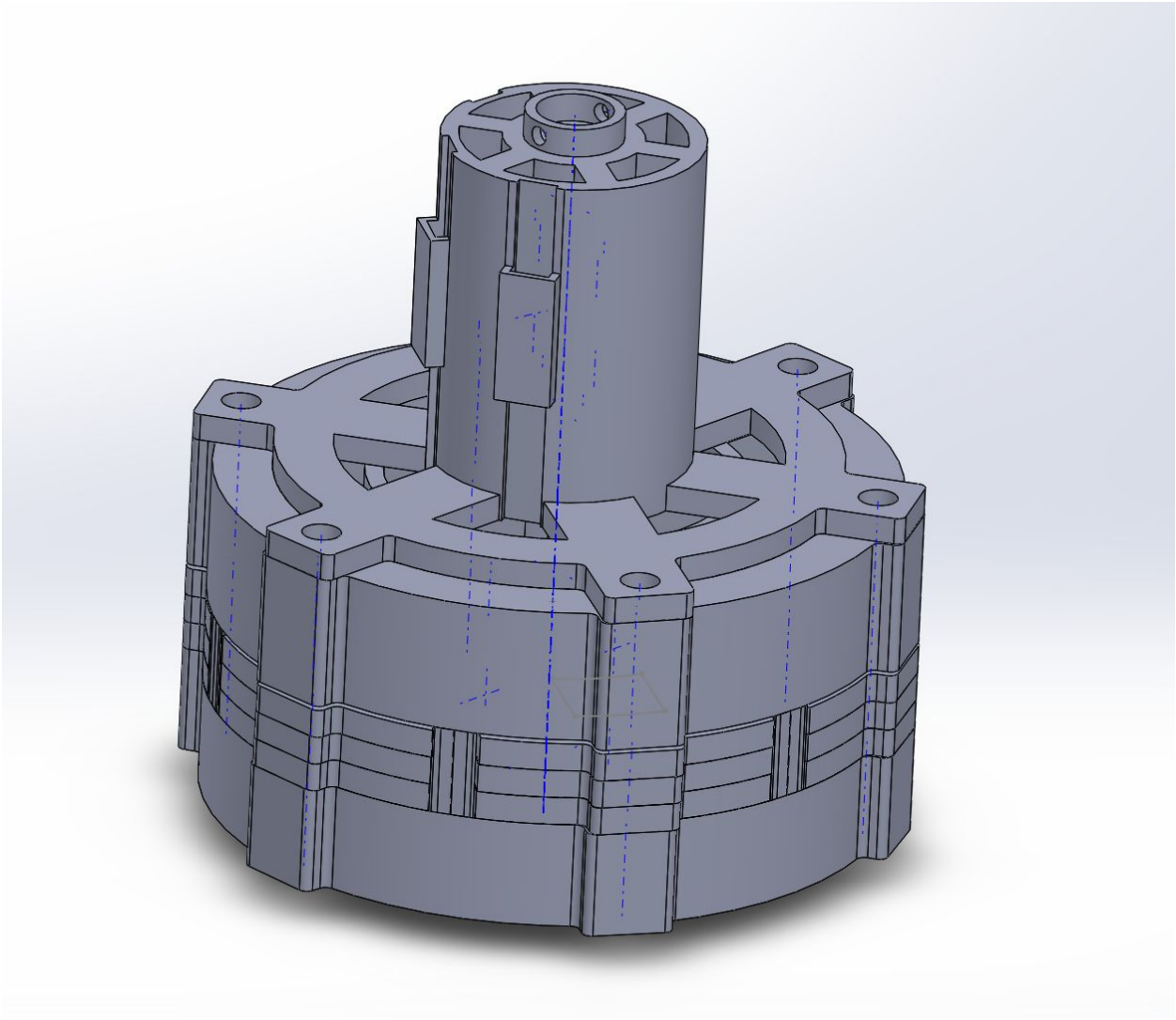


Figure A3.4.11: Exterior view of the complete motor mechanical design

A4 Test Cases

A4.0 Test Cases for Custom Motor

Test Case TC001

System / Subsystem: Motor Testing

Purpose of Test: To verify that the customized motor is functional as specified by the RCGs for the motor.

Test Procedure:

1. Load the Arduino hall sensor testing program, and spin the propeller blade by hand.

   Do the hall sensors read phases correctly? (Yes) No

2. Connect a power supply capped at 18V to the red and black power plugs located at the base of the robot. Turn the current to 0.5 Amp.

3. As the propeller blade spin through different phases, is the voltage reading relatively stable? i.e., not showing sharp voltage drop.

   Is the reading relatively stable? (Yes) No

4. Load the Arduino slow phase testing program, turn the current to 1 Amp, and have the motor spinning at a very slow rate. Does the rotor spin and strop at the correct speed and position for each phase?

   Does the rotor spin and stop correctly? (Yes) No

5. Load the fully functional motor control Arduino program, and turn the current to 2 amp. Press the reset button on Arduino.

   Can the motor start on its own? (Yes) No

6. While the motor is spinning. Is the motor making any noise?

   Is the motor quiet? (Yes) No

7. While one hand is touching the metal rack of the lab bench, use the other hand to touch the Mosfets in the H-bridge and check the temperature.

   Are these Mosfets maintaining a relatively low temperature?
   (Yes) No

9. Enter the PWM values wanted in the serial monitor, does the motor speed change according to PWM?

Does Motor speed change? Yes No

Deviations from Protocol: _____

Tested By: M.L Date: Apr. 06. 2016

Verified by: M.Q Date: Apr. 06. 2016.

A4.1 Test Cases for System

Test Case TC002

System / Subsystem: Control System Testing

Purpose of Test: To verify that the control system is functional as specified by the RCGs for the system.

Test Procedure:

1. Connect a power supply capped at 18V to the red and black power plugs located at the base of the robot. Please have a timer for the following test cases.
2. Press the restart button on the Arduino UNO. Verify the lift motor begins to turn.

   Does the lift motor turn? (Yes) No
3. Do not disturb the system. Verify that the lift motor is providing enough power for the arm to lift.

   Does the arm lift? (Yes) No
4. Do not disturb the system. Is the arm able to balance at the horizontal within 60 seconds from startup?

   Does the arm balance? (Yes) No
5. If the arm has balanced, verify that the yaw motor begins the spin within 30 seconds of stabilization.

   Does the yaw motor turn? (Yes) No
6. Do not disturb the system. Verify that the yaw motor is providing enough power for the base to rotate.

   Does the base rotate? (Yes) No
7. Do not disturb the system. Is the base able to rotate 180 degrees and maintain that angle within 60 seconds from when the yaw motor first started spinning?

   Does the base rotate 180 degrees? (Yes) No

8. If the motor has rotated 180 degrees, verify that the lift motor is still balancing at the horizontal.

   Is the arm still balanced? (Yes) No

9. Turn off the power supply. Using a metre stick, verify that the dimensions of the robot do not exceed 2ft by 2ft by 2ft.

   Do the robot dimensions exceed 2ft by 2ft by 2ft? Yes (No)

Deviations from Protocol: _____

Tested By: A.A Date: Apr. 6, 2016

Verified by: S.W Date: Apr. 6, 2016

A4.3 Test Cases for Control

Test Case TC003

System / Subsystem: Full System

Purpose of Test: To verify the operation of the fully integrated system while under our predefined bath

Test Procedure:

1. Connect a 15 V, 3 Amp, power supply to the slip ring power keeping the power off for the time being.
2. Attach a 9V battery to the two Arduinos.
3. Reset the control system controller before proceeding.
4. Immediately after, power up the slip ring connection.

Did the system hover effectively?	YES	NO
Did the system rotate effectively?	YES	NO
Did the system remained powered during the test?	YES	NO

Deviations from Protocol:

Tested By: M.Q Date: Apr. 6. 2016

Verified by: A.A Date: Apr. 6. 2016

A5: Arduino Code

A5.0: Control code

```
 #include <PID_v1.h>
//Pin 5 = pwm output to yaw motor
//pin 6 = pwm output to lift motor
 const int PWM_LIFT_PIN = 6;
 const int PWM_YAW_PIN = 5;
 //Note: RE is now actually a secondary OE
 volatile double RE_angle; //rotary encoder angle
 volatile int RE_dir; //rotary encoder direction
 const int RE_lead_pin = 3;
 const int RE_lag_pin = 13;

 volatile double OE_angle; //optical encoder angle
 volatile int OE_dir; //optical encoder direction
 const int OE_lead_pin = 2;
 const int OE_lag_pin = 12;


 //OE PID variables
 //LIFT MOTOR
  double OE_Setpoint, OE_PID_Input, OE_PID_Output;
  int OE_Count = 0;
  int RE_Count = 0;
  PID OE_PID(&OE_PID_Input, &OE_PID_Output, &OE_Setpoint,4.5,3,5, DIRECT); // in, out, Kp, Ki, Kd, direction Best So far 4.5,3,5

 //OE PID variables
 //YAW MOTOR
  double RE_Setpoint, RE_PID_Input, RE_PID_Output;
  PID RE_PID(&RE_PID_Input, &RE_PID_Output, &RE_Setpoint,0.35 ,0.1,1.5, DIRECT); // in, out, Kp, Ki, Kd, direction Best So far 0.35 ,0.1,1.5

void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 pinMode(RE_lag_pin, INPUT);
 pinMode(OE_lag_pin, INPUT);
 attachInterrupt(0, OE_ISR, RISING); //optical encoder isr, will trigger on rising edge of channel A
 attachInterrupt(1, RE_ISR, RISING); //rotary encoder isr, will trigger on rising edge of channel A

 //initalize Optical PID
 OE_PID_Input = OE_angle; //input to RE PID controller is the actal measured angle
 OE_Setpoint = 28.8; //for now we set a test setpoint to 36 deg
 OE_PID.SetMode(AUTOMATIC); //turn on the PID to automatic mode
 OE_PID.SetOutputLimits(0, 255); //cap the output of the OE_PID to 0(motor off) to 255 (max speed)

 //initalize Rotary PID
 RE_PID_Input = RE_angle; //input to RE PID controller is the actal measured angle
 RE_Setpoint = 360; //for now we set a test setpoint, test on -36,-72
 RE_PID.SetMode(AUTOMATIC); //turn on the PID to automatic mode
 RE_PID.SetOutputLimits(0, 255);
}

void loop() {
  //Compute PID output for OE and RE
  OE_PID_Input = OE_angle; //input to OE PID controller is the actal measured angle
  OE_PID.Compute();

  RE_PID_Input = RE_angle; //input to RE PID controller is the actal measured angle
  RE_PID.Compute();
```

```
  //write output to motor 1 (Lift)
  analogWrite(PWM_LIFT_PIN,(int)OE_PID_Output);

  if((OE_Setpoint - OE_angle) < 3.6 && (OE_Setpoint - OE_angle) > -3.6)
      OE_Count = OE_Count + 1;

  if(OE_Count <1000){
   RE_Setpoint = 0;
  }
  else{
   RE_Setpoint = 360;
  }

   Serial.print(OE_Count);
   Serial.print('\n');
   if (RE_PID_Output < 127 && RE_angle<RE_Setpoint)
        RE_PID_Output = 127;
   if (RE_angle>RE_Setpoint && RE_PID_Output >0)
        RE_PID_Output = 127;

   analogWrite(PWM_YAW_PIN,(int)RE_PID_Output);

   if((RE_Setpoint - 360) < 7.2 && (RE_Setpoint - 360) > -7.2){
     if((OE_Setpoint - OE_angle) < 3.6 && (OE_Setpoint - OE_angle) > -3.6)
          RE_Count = RE_Count + 1;
   }
   if (RE_Count > 1000)
     OE_Setpoint = OE_Setpoint -0.01;
}

 void OE_ISR (){

     if (digitalRead(OE_lag_pin) == HIGH){
     OE_dir = -1;
     OE_angle = OE_angle - 3.6;
     if (OE_angle <= 0)
         OE_angle = 0;
   }

   else if (digitalRead(OE_lag_pin) == LOW){
     OE_dir = 1;
     OE_angle = OE_angle + 3.6;
      if (OE_angle >= 360)
        OE_angle = OE_angle - 360;
   }
 }

 void RE_ISR (){

     if (digitalRead(RE_lag_pin) == HIGH){
     RE_dir = -1;
     RE_angle = RE_angle + 3.6;
     //if (RE_angle <= -360)
      //   RE_angle = RE_angle + 360;
     }

   else if (digitalRead(RE_lag_pin) == LOW){
     RE_dir = 1;
     RE_angle = RE_angle - 3.6;
   }
 }
```

A5.1: Hall Sensor Test

```
volatile int A, B, C;
int startupDelay = 5000;

//H Bridge inputs
int A_low  = 2;
int A_high = 3;

int B_low  = 4;
int B_high = 5;

int C_high = 6;
int C_low  = 7;

//PWM Variable
int duty_cycle;


  //A Low   2
  //A High  3

  //B Low   4
  //B High  5

  //C High  6
  //C Low   7

void setup()
{
  Serial.begin(9600);
  Serial.println("Boe");
  //THE P MOSFETS ARE ACTIVE LOW
  //port D controls pin 0-7
  DDRD = B11111100;  // sets Arduino pins 2 - 7 as outputs

  //Initialize all inputs as off
  PORTD = B0101000;

  InitialiseIO();

}

void loop() {


}

void InitialiseIO(){
  pinMode(A0, INPUT);         // Pin A0 is input to which a switch is connected
  pinMode(A1, INPUT);         // Pin A1 is input to which a switch is connected
  pinMode(A2, INPUT);         // Pin A2 is input to which a switch is connected

}

ISR(PCINT1_vect) {    // Interrupt service routine. Every single PCINT8..14 (=ADC0..5) change
          // will generate an interrupt: but this will always be the same interrupt routine


  A = digitalRead(A0);
  B = digitalRead(A1);
  C = digitalRead(A2);



    if((A == 0) && (B == 1) && (C == 1))
  {
   //Serial.print("Phase 1 \n");
   //B high, A low
   PORTD = B01001100;
   analogWrite(B_high, dutyCycle);
  }

   else if((A == 1) && (B == 1) && (C == 1))
  {
   //Serial.print("Phase 2 \n");
```

```
  //C high, A low
  analogWrite(B_high,offPwm);
  PORTD = B00101100;
  analogWrite(C_high,dutyCycle);

 }

  else if((A == 1) && (B == 1) && (C == 0))
 {
 //Serial.print("Phase 3 \n");
 //C high, B Low
 PORTD = B00111000;
 analogWrite(C_high,dutyCycle);
 }

  else if((A == 1) && (B == 0) && (C == 0))
 {
  //Serial.print("Phase 4 \n");
  //A high, B Low
  analogWrite(C_high,offPwm);
  PORTD = B01110000;
  analogWrite(A_high, dutyCycle);
 }

  else if((A == 0) && (B == 0) && (C == 0))
 {
 //Serial.print("Phase 5 \n");
  //A high, C low
  PORTD = B11100000;
  analogWrite(A_high, dutyCycle);
 }

  else if((A == 0) && (B == 0) && (C == 1))
 {
 //Serial.print("Phase 6 \n");
  //B High, C Low
  analogWrite(A_high, offPwm);
  PORTD = B11001000;
  analogWrite(B_high, dutyCycle);
 }
 else
 {
  //Serial.print("WARNING; CATASTROPHIC FAILURE");
 }


}
```

A5.2: Motor Slow Spin Test

```
volatile int A, B, C;
int startupDelay = 2000;


void setup()
{
  Serial.begin(9600);
  Serial.println("Boe");
  //THE P MOSFETS ARE ACTIVE LOW
  //port D controls pin 0-7
  DDRD = B11111100;  // sets Arduino pins 2 - 7 as outputs

}

void loop() {


    //Phase 1
   Serial.print("Phase 1 \n");
   delay (startupDelay);
   PORTD = B01001100;



 //Phase 2
   Serial.print("Phase 2 \n");
   delay (startupDelay);
   PORTD = B00101100;



 //Phase 3
   Serial.print("Phase 3 \n");
   delay (startupDelay);
  PORTD = B00111000;


 //Phase 4
  Serial.print("Phase 4 \n");
   delay (startupDelay);
   PORTD = B01110000;



 //Phase 5
   Serial.print("Phase 5 \n");
   delay (startupDelay);
   PORTD = B11100000;

 //Phase 6
   Serial.print("Phase 6 \n");
   delay (startupDelay);
    PORTD = B11001000;


}
```

A5.3: Integrated Motor Code

```
//This variable takes our value from



volatile int A, B, C;

//Variables for our PWM shit :)
int dutyCycle;
int dutyRaw;
int offPwm = 255;
double dutyCyclePercentage;

//Moshe, since you can't seem to get this through your fucking thick skull. Wire things properly. You just spent hours on having A_high in port 2 instead of 3
//The only pwm pins are 2,5,6

//H Bridge inputs
int A_low  = 2;
int A_high = 3;

int B_low  = 4;
int B_high = 5;

int C_high = 6;
int C_low  = 7;

  //A Low   2
  //A High  3

  //B Low   4
  //B High  5

  //C High  6
  //C Low   7



void setup()
{
  Serial.begin(9600);
  Serial.println("Boe");

  DDRD = B11111100;  // sets Arduino pins 2 - 7 as outputs
  //Initialize all inputs as off
  PORTD = B0101000;

  //Initialize our duty cycle as 0%
  //Remember we are switching the P Mosfets high so these are active
  dutyCycle = 0;

  //We need to increase the frequency on the PWM pins so let us do that
  //Timers 0 controls Pins 5 and 6
  TCCR0B = TCCR0B & B11111000 | B00000010;    // set timer 0 divisor to     8 for PWM frequency of  7812.50 Hz

  //Timer 2 controls pin 3
  TCCR2B = TCCR2B & B11111000 | B00000001;    // set timer 2 divisor to     1 for PWM frequency of 31372.55 Hz


  InitialiseIO();
  InitialiseInterrupt();
}

void loop() {

  dutyCycle = 255 - ((255.0/1023.0)*analogRead(A5));
  //Serial.print("\n");
  //Serial.print(dutyCycle);

}

void InitialiseIO(){
  pinMode(A0, INPUT);         // Pin A0 is input to which a switch is connected
  pinMode(A1, INPUT);         // Pin A1 is input to which a switch is connected
  pinMode(A2, INPUT);         // Pin A2 is input to which a switch is connected
```

```
}

void InitialiseInterrupt(){
  cli();                          // switch interrupts off while messing with their settings
  PCICR =0x02;          // Enable PCINT1 interrupt
  PCMSK1 = 0b00000111;
  sei();                          // turn interrupts back on
}

ISR(PCINT1_vect) {    // Interrupt service routine. Every single PCINT8..14 (=ADC0..5) change
          // will generate an interrupt: but this will always be the same interrupt routine



  A = digitalRead(A0);
  B = digitalRead(A1);
  C = digitalRead(A2);

  //We need to add some code to read Alex's value
  //We are going to read a value on our analog pin
  //We will need to assign this value to our dutyCycle

  //int sensorValue = analogRead(A5);
  //float voltage = sensorValue *(5.0/1023.0);
  //We need to scale this voltage then to a max value of 255
  //dutyCycle = voltage*51;




//The phases are changed around a little for hall sensor values but the commutation is the same per phase



  //A Low   2
  //A High  3

  //B Low   4
  //B High  5

  //C High  6
  //C Low   7

    if((A == 0) && (B == 1) && (C == 1))
  {
   //Serial.print("Phase 1 \n");
   //B high, A low
   PORTD = B01001100;
   analogWrite(B_high, dutyCycle);
  }

   else if((A == 1) && (B == 1) && (C == 1))
  {
   //Serial.print("Phase 2 \n");
   //C high, A low
   analogWrite(B_high,offPwm);
   PORTD = B00101100;
   analogWrite(C_high,dutyCycle);

  }

   else if((A == 1) && (B == 1) && (C == 0))
  {
  //Serial.print("Phase 3 \n");
  //C high, B Low
  PORTD = B00111000;
  analogWrite(C_high,dutyCycle);
  }

   else if((A == 1) && (B == 0) && (C == 0))
  {
   //Serial.print("Phase 4 \n");
   //A high, B Low
   analogWrite(C_high,offPwm);
   PORTD = B01110000;
   analogWrite(A_high, dutyCycle);
  }
```

```
  else if((A == 0) && (B == 0) && (C == 0))
  {
  //Serial.print("Phase 5 \n");
   //A high, C low
   PORTD = B11100000;
   analogWrite(A_high, dutyCycle);
  }

  else if((A == 0) && (B == 0) && (C == 1))
  {
  //Serial.print("Phase 6 \n");
   //B High, C Low
   analogWrite(A_high, offPwm);
   PORTD = B11001000;
   analogWrite(B_high, dutyCycle);
  }
  else
  {
   //Serial.print("WARNING; CATASTROPHIC FAILURE");
  }


}
```

A6: References

[1]"Arduino - ArduinoBoardUno", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed: 05- Mar- 2016].

[2]"BLDC Motor and Controller Theory", *Hackaday.io*, 2016. [Online]. Available: https://hackaday.io/project/3176-gator-quad/log/11053-bldc-motor-and-controller-theory. [Accessed: 14- Mar- 2016].

[3]"Arduino - Constants", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Reference/Constants. [Accessed: 14- Mar- 2016].

[4]"Arduino Hall Effect Sensor Tutorial: The Easy Way", *DIY Hacking*, 2014. [Online]. Available: http://diyhacking.com/arduino-hall-effect-sensor-tutorial. [Accessed: 14- Mar- 2016].

[5]W. Brown, *Brushless DC Motor Control Made Easy*, 1st ed. Microchip, 2016.

[6]"Solenoids as Magnetic Field Sources", *Hyperphysics.phy-astr.gsu.edu*, 2016. [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/solenoid.html. [Accessed: 14- Mar- 2016].

[7]*Dual 3A-Peak Low-Side MOSFET Driver*, 1st ed. Micrel, 2016.

[8]*http://www.skf.com/group/products/bearings-units-housings/ball-bearings/principles/friction/estimating-frictional-moment/index.html*

[9]https://www.arduino.cc/en/Hacking/PinMapping

[10]W. Storr, "MOSFET as a Switch - Using Power MOSFET Switching", Basic Electronics Tutorials, 2013. [Online]. Available: http://www.electronics-tutorials.ws/transistor/tran_7.html. [Accessed: 11- Mar- 2016].