MSE 4499 — Mechatronic Design Project

CubeSat Ground Station

by

Jianhui Li, Tushar Mahajan, Paavan Raj, and Annie Wu

Design Analysis and Detailed Design Documentation

Faculty Advisor: Dr. Sabarinathan and Dr. Cross

Date Submitted: March 14, 2019

Total Design Analysis Word Count: 2977

Total Prototype Feasibility Word Count: 1415

Table of Contents

List of Figures

List of Tables

Introduction

Western University's CubeSat Project is designing two parallel CubeSat satellite systems, with different communication frequency bands, that will undergo a trade-off, launching only one of them in 2022. They lack an accessible system on the ground that can provide real-time communication with the satellite, monitor and predict movement of the satellite, and correct antenna direction based on its trajectory. This ground system must be robust to either communication schemes.

The problem statement is to design and build a functional satellite tracking ground station with high alignment accuracy between the CubeSat in orbit and the ground station, to achieve better communication and data transfer.

The overall scope of this project is to design a fully functioning satellite tracking ground station for Western's CubeSat. This project will cover the electrical and mechanical subsystems for motor actuation and sensory data acquisition, control system for mitigating tracking error, and the orbital prediction algorithm that the ground station will employ to track the satellite's orbit. However, since the satellite will not be launched by the end of the semester, the plan is to accomplish successfully aligning with and tracking an orbiting object in space whose data is readily available, such as the moon or the International Space Station.

Some of the key constraints that we must follow are that the ground station must maintain high alignment accuracy with the satellite, be able to predict the orbital trajectory of the satellite, correct the antenna's direction and orientation, and finally, be able to withstand the external environmental conditions. The main project objectives are for the ground station to be easily transportable, require low maintenance, and be manufactured from as many off the shelf commercial components as possible.

Section 1.1: Design Analysis – Calculations and Design Verification

Material Analysis:

Machined Components

The following are all custom components, and all were fabricated from 4140 steel.

- Bearing Holder
- Bearing Holder Flange
- Bar Topper Flange
- Bar Topper Flange Plates
- Bar Topper Motor Plate
- Motor Coupler
- Motor Holder
- Base for elevation motor
- Motor Plate
- Motor Shaft

4140 is an extremely common low alloy steel, with extremely high yield strength, torsional strength and Youngs Modulus [1]. These are all extremely important as these components carry heavy loads which are discussed in section 2. Furthermore, 4140 has a relatively high strength to cost ratio, is easily machined, and is readily available in the required sizes. Although, there are materials that would have been easier to machine, such as aluminum, these materials are much more expensive, so they were disregarded.

Main Shaft

The main pole in the ground station was originally supposed to be a telescoping pole, and the material selection was performed under this impression. After altering the concept to only have a pole of a fixed height, the material selected would still be a great choice, although the pole would have a square base as opposed to a circle. The material selected was medium carbon steel, which is the typical material of choice for structural beams. Medium carbon steel has a great balance of ductility and strength [2]. It was also originally chosen for its wear resistance, as the telescoping poles would need to retract into one another, causing a large surface area of contact. Since the main pole is such an important part of the design structurally and is an exposed part of the ground station, it was important that carbon steel is extremely strong and also not prone to corrosion as many other metals are.

Base

The base is entirely constructed out of wood, with the inner trusses and supports made of pine planks, and the outer box of a weatherproofed wood composite. Wood was selected as it is inexpensive and easier to work with than metal substitutes. Pine was selected as opposed to other options because it was inexpensive and resists shrinking and swelling, which is important as it will be resistant to temperature changes. As well, pine was readily available for this project.

The outer box that encloses the base structure and houses the electronics was originally planned to be made of sheet metal, however a reviewing the design, sheet metal would be extremely expensive and unnecessary. Instead, a weatherproof composite wood was selected, which was an even better fit as it was more inexpensive and easier to fabricate into the desired box shape safely.

Mass:

The mass of the box was approximated to be 21 kilograms and the mass of the other components was approximated to be 53 kilograms.

Inertia of Arm:

Of the antenna possibilities, the parabolic dish is the heavier and more directional antenna, so it will be used as a basis for determining the required torque outputs. Approximating the mass of the antenna and the arm to be a 15 kg point mass acting halfway along the moment arm (0.20 m from the axis of the motor) (Figure 1 below), the following mass moment can be calculated:

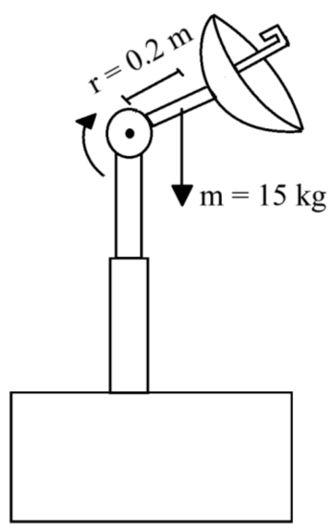


Figure 1: Freebody diagram of antenna and arm

$$\Sigma M = Fr = 15kg \times 9.81\frac{m}{s^2} \times 0.20\ m \times \cos(\alpha) = \ 29.43\cos(\alpha)\ N \cdot m$$

This result includes the angle, α, of the arm from a flat horizon.

Angle of Detection:

The maximum angle that the azimuth motor would have to travel is 150˚ or $\frac{5}{6}\pi\ rads$. This number was obtained by taking the standard 180˚ of a completely flat field, with 15˚ being removed at either horizon due to an expected treeline and signal power loss at low elevations.

Reaction Time:

Given that the maximum uplink time is approximately 10 minutes [3], and the maximum angle that the azimuth motor would have to travel is $\frac{5}{6}\pi\ rads$, the maximum speed required is then:

$$\omega_{max} = \frac{\frac{5}{6}\pi}{10\ min} = 0.2618\frac{rad}{min} = 0.0417\ rad/s$$

Wind Considerations:

The external force of wind has a large impact on the torque required to actuate the dish. In Canada, the maximum windspeed recorded is approximately 80 kph, which will serve as the worst-case scenario [4]. Using the surface drag equation of a body in a fluid [7 from phase 2], the force exerted on the antenna when the face is at 45˚ to the wind can be found using the following equations:

$$F_{wind} = \frac{1}{2}\rho_{air}v_{wind}^2A_{dish}$$

$$\rho_{air} = 1.2\ \frac{kg}{m^3}$$

$$v_{wind} = 80\frac{km}{h} = 22.2\frac{m}{s}$$

$A_{dish} = 0.622\ m^2 \cdot cos45° = 0.439.82\ m^2$ (antenna tilted slightly upwards)

$$F_{wind} = \frac{1}{2} \times 1.2\ \frac{kg}{m^3} \times \left(22.2\frac{m}{s}\right)^2 \times 0.439m^2 = 130.0\ N$$

And the torque applied is thus:

$$\tau = Fr = 130.0\ N\ \cdot 0.4\ m\ \cdot \cos 45° = \ 36.8\ N \cdot m$$

Torque of Elevation Motor:

From the *Inertia of Arm* section, the maximum moment occurs when cos(α) = 1, or when the arm is horizontal. In addition to the mass moment, the motor will also need to overcome the inertial loads to begin moving, with a steady state operation of 0.042 rad/s from the *Reaction Time* section and inertial load of parabolic dish given as 0.5449 kg·m^2 (approximate calculation using a flat circle to model the dish). An acceleration period from rest to a steady state operation of 0.1s is enough for tracking, so minimum required torque is then:

$$\tau = I_z \alpha$$

$$\alpha = \frac{d\omega}{dt} = \frac{0.043\frac{rads}{s} - 0\frac{rads}{s}}{0.1s - 0\ s} = 0.0043\frac{rads}{s^2}$$

$$\tau = 0.5449kg \cdot m^2 \times 0.043\frac{rads}{s^2} = 0.023\ N \cdot m$$

During steady operation, since the angular speed stays constant, the motor will not need to supply additional torque as there will be no acceleration.

With the values derived in the *Inertia of Arm* and *Wind Considerations* sections, and this calculated torque, the maximum required torque is the summation of the three, which gives a critical torque of 66.25 N·m. With a 160:1 gear reduction however, the motor will only need to supply 1/160th of this, therefore the maximum required torque is 0.414 N·m. The motor can provide this maximum torque, as evidenced by the torque curve from the motor specifications below.

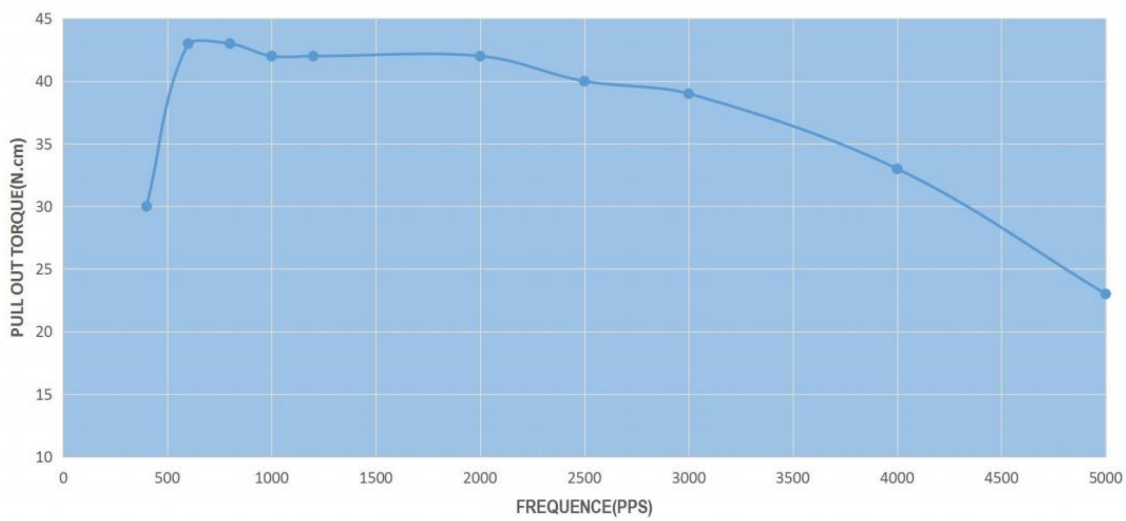


Figure 2: Torque curve for NEMA 17 stepper motor [6]

Tipping Point:

With the pole height and the significant wind force, a critical failure mode would be tipping. With pole height being 1.4m, and the entire mass of the system being 55 kg at the center of mass, only the wind force is needed before the tipping point can be calculated. The max wind force would be when the antenna faces the wind fully.

$$A_{dish} = 0.622\ m^2 * \cos(0) = 0.622\ m^2$$

$$F_{wind} = \frac{1}{2} \times 1.2\ \frac{kg}{m^3} \times \left(22.2 \frac{m}{s}\right)^2 \times 0.622 m^2 = 183\ N$$

$$M_{wind} = \frac{L_{base}}{2} \cdot m_{total} \cdot g$$

$$1.4m\ \cdot 183N = \frac{L_{base}}{2} * 55\ kg * 9.81 \frac{m}{s^2}$$

$$L_{base} = 0.9497\ m$$

This result is the minimum length each side of the base needs to be so that the ground station will not tip over.

Stress Analysis and Failure Point Analysis:

Gearbox

Although FEA was conducted, a multitude of factors can affect the final strength of the 3D printed components such as:

- The specific material properties of the filament used as the properties can change between different manufacturers
- Infill Pattern
- Print defects
- Nozzle thickness
- Printing temperature

This, along with the inherent errors in FEA studies, results in high inaccuracies between simulated strength of the part and the actual strength of the part. A study done at the United States Merchant Marine Academy [7] has created a formula for converting the material properties given by the manufacturer of the filament to actual properties of the printed part based on the wall thickness, infill, and the infill pattern. Tests done with the use of this formula show that it allows for a better approximation of the material properties. It is recommended, however, that a factor of safety above 2 should be used in order to account other factors that the strength of the 3D print is dependent on. For a property P, the material property P_m can be used to find the design property P_d as

$$P_d = P_m * K_{FFF}\left(\frac{A_w}{A_x} + \frac{A_i}{A_x} * K_{i,\%} * K_{i,t}\right)$$

Calculations based on this formula can be found in Appendix 1.

A new material was created in Fusion 360 using these adjusted material properties. Due to the complexity of the model, it was not possible to run a dynamic analysis on all of the parts of the gearbox. Therefore, a static analysis was run on the two components, the output ring gear and the input connector, at the points of maximum load.

The output ring gear has a maximum load when the moment of the antenna and the wind load are at their greatest. The most critical point of failure is the face attached to the pole holding the antenna. Therefore, a force of 210 N (130 N of force from the weight and 150*cos(45) from the wind load), was applied. A fixed constraint was applied to the body of the output ring gear. The result of this simulation can be seen in Figure 3. With a maximum stress of 2.481 MPa, the part has a factor of safety of above 19. There is a maximum displacement of 0.02 mm that will not affect the positioning of the antenna as seen in Figure 4.

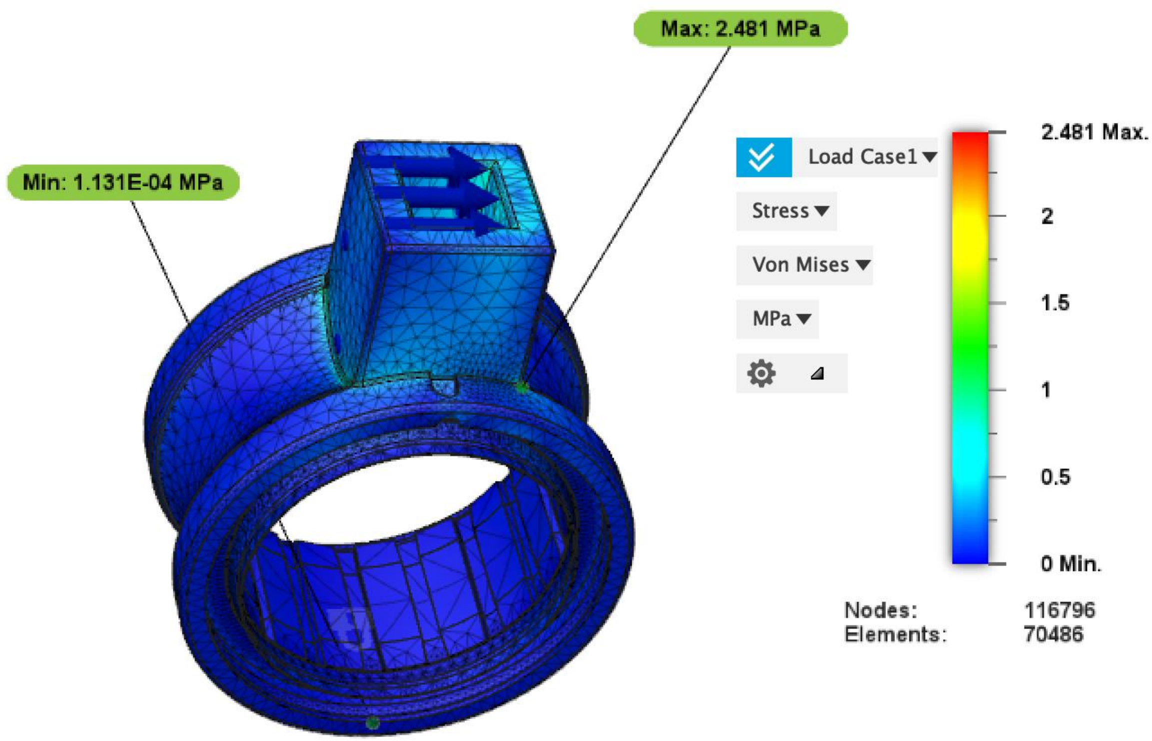


Figure 3: Maximum Von Mises Stress in The Output Ring Gear

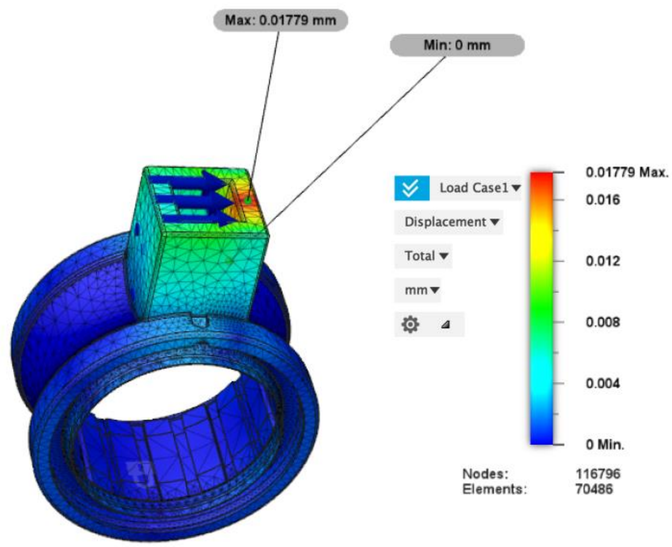


Figure 4: Maximum Displacement in The Output Ring Gear

As before, the most critical point of failure for the input connector occurs when the output gear has the maximum load. Though the calculated maximum load was 0.414 Nm between the gears, a torque of 1 Nm was applied to the input connected to account for friction. A fixed constraint was applied to the body of the shaft. The maximum stress in the part was 3.828 MPa as seen in Figure 5, resulting in a factor of safety of approximately 13.

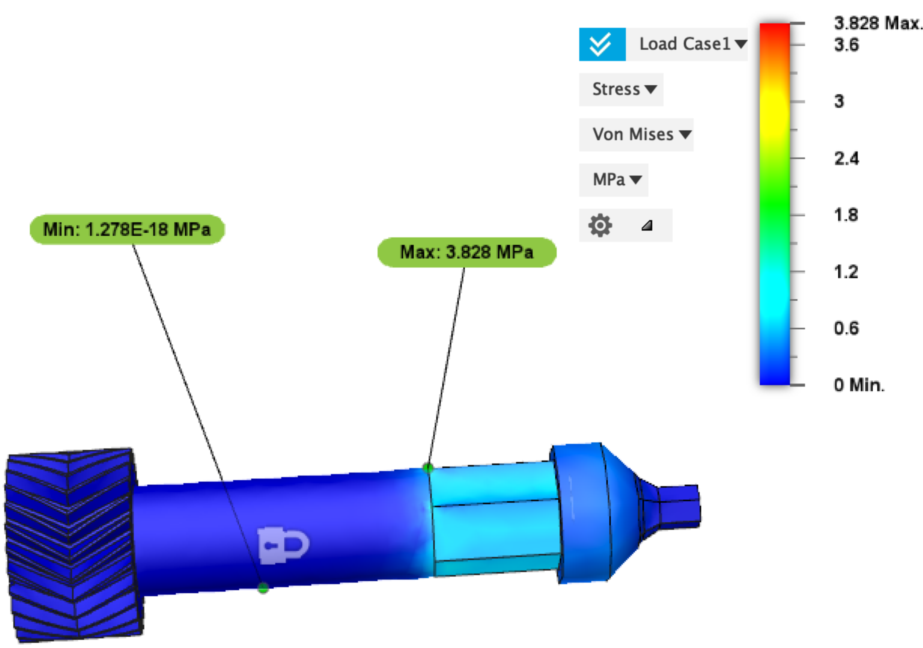


Figure 5: Maximum Von Mises Stress in The Input Connector

The forces on the teeth of the gears can be assumed to be negligible compared to the force on the output ring gear and the torque on the input connector as the total force is spread out over the contact between 9 planetary gears, 3 sun gears and 3 ring gears. As the two components most likely to fail have factors of safety of well above 10, it can be stated that the gearbox will not fail under the required loading.

Bearing Holder

Another point of failure could be the bearing holder as it holds the weight of the 10 inch long, 0.5 inch thick square steel plate as well as the entire elevation subassembly. In order machine the part with the tolerances required to press fit radial bearings, as well as to have sufficient access to the motor coupler, the 2.5 inch wide steel round bar was machined down to a thickness of 0.25 inches. As the total weight transferred through a thrust bearing to the bearing holder is approximately 40 pounds (the exact value is not known due to the 3D printed components), an FEA analysis was run to determine the factor of safety of this component. The outside bottom 0.125 inches as well as the bottom of the component were fixed as these faces will be welded to rigid body. A force of 110 (50 pounds for the subassembly (rounding up as the exact value was unknown), 40 pounds for the antenna (heaviest antenna being considered by the communications team), and 20 pounds for the downward component of the wind load) was applied to the face that the thrust bearing will rest on. This FEA analysis resulted in a minimum factor of safety of 332 as seen in Figure 6.

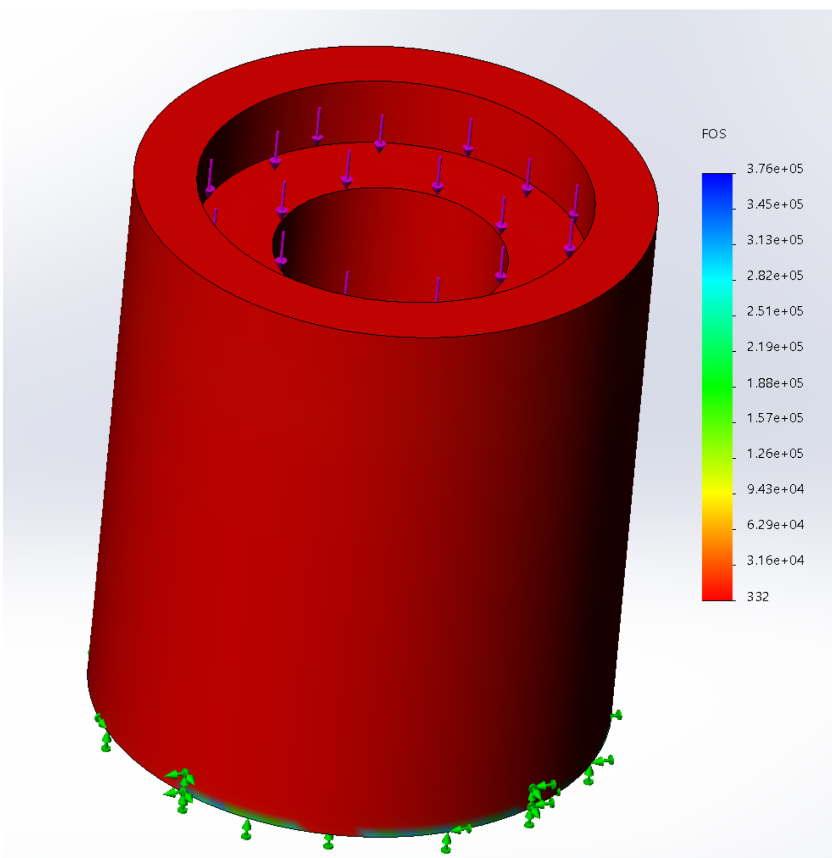


Figure 6: Bearing Holder Factor of Safety

Pole Connecting Base To Other Assemblies

The original concept utilized a round pole with flanges and contacts at three different heights with the base. This resulted in minimal deflection of the Azimuth and Elevation subassemblies with respect to the base. However, the flanges used to make the connections between the base components and the pole would need to be custom made. To avoid this, a square tube was used and connections were made using angle brackets. Using an FEA simulation shows the stressed on the points of contact with the circular tube the square tube (See Figures 7 and 8). On both poles, the outer face was split into sections where there was contact with the base in order to fix those sections of the pole, with a force of 150 Newtons applied to the top face. The displacement during extreme weather conditions of the square tube with two points of contact (0.0029 inches) was less than the displacement of the circular tube with three points of contact (0.094 inches). Therefore, the trusses would be redundant if used with the square tube.

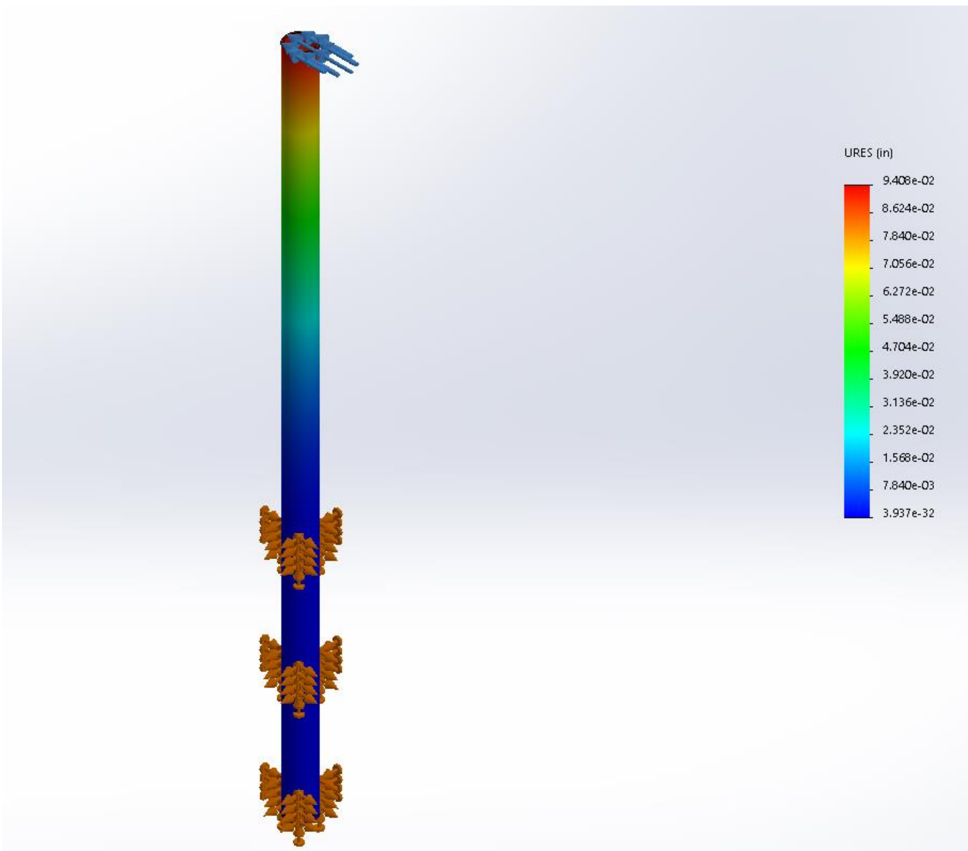


Figure 7: FEA Analysis showing displacement for a round tube with three points of contact.

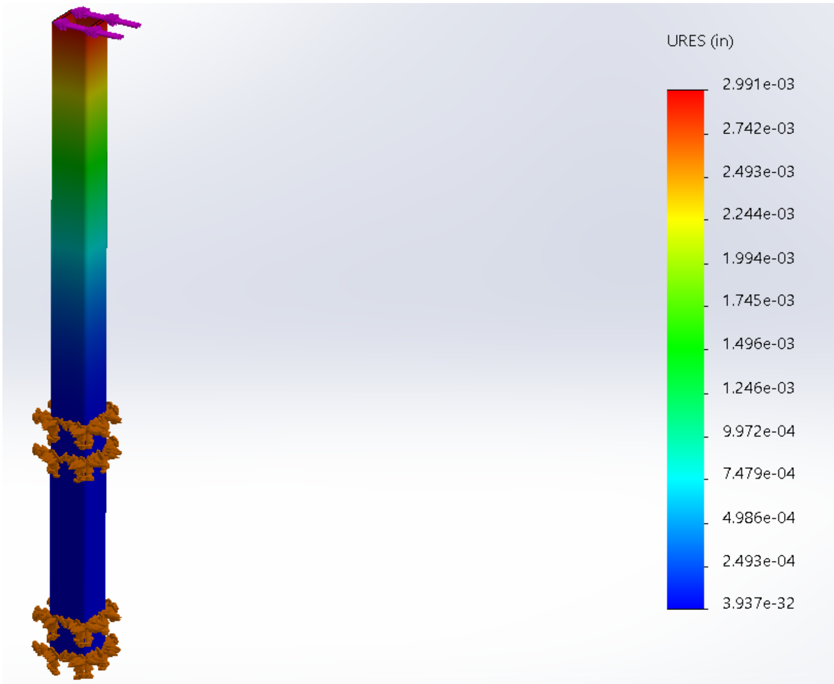


Figure 8: FEA Analysis showing displacement for a square tube with two points of contact.

Orbital Mechanics:

Simplified General Perturbations (SGP) models are used to model the orbit of LEO objects. The two-line element set is a data format that encodes the orbital elements of a satellite at a given point in time. The two-line elements that are created from the SGP orbit determination can be used to model an adequately accurate propagation of the CubeSats motion [9]. These systems both use Keplerian orbit mechanics to mathematically approximate an orbit change over time, using the following equations to describe location of the orbital element.

$$\Pi = \varpi + \Omega$$

$$u_0 = \varpi + v_0$$

$$l_0 = \Omega + u_0$$

Where: Π – longitude at periapsis, ϖ – argument of periapsis, Ω – longitude of the ascending node, u_0 – argument of latitude at epoch, v_0 - true anomaly at epoch, and l_0 – true longitude at epoch

The orbital trajectory software uses this latitude and longitude data in conjunction with the locational data of the ground station from the GPS module to calculate the exact angle between them. The Keplerian orbit data will be pulled from the CubeSat every certain number of seconds, the trajectory software will recalculate the angle, at which point the azimuth and elevation motors will move so that the antenna is pointing in the right location. This process is illustrated in the following figure.

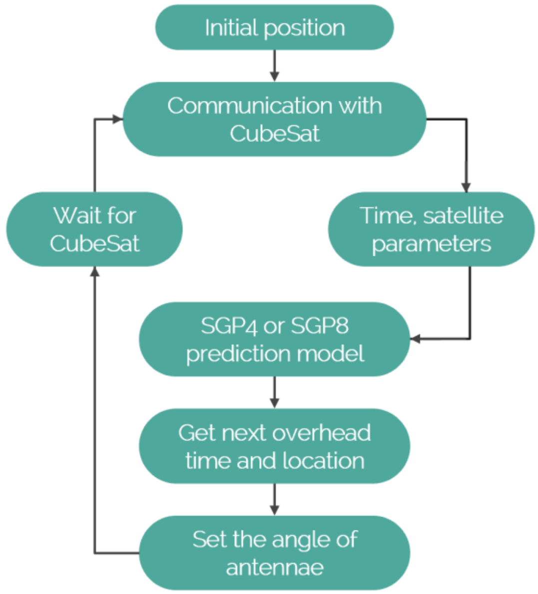


Figure 9: Software flow using orbital elements

Electrical Power Requirements:

Given the various electrical components below, as well as their prescribed current and voltage requirements, total power requirement can be calculated as follows

$$P_{total} = \Sigma P_{components} = \Sigma V_{components} I_{components}$$

$$P_{total} = 5V(0.1A \times 2 + 0.3A + 0.1A + 2.5A) + 2 \times 2.4\ A \times 18\ V = 101.9W$$

Assuming a factor of safety of 1.5 (i.e. that at max 50% of the power supplied by the source will be lost to heat and sound), the minimum power supplied by the PSU should be 152.9 W.

An auxiliary calculation to the power requirement would be regarding the varying voltages at which the components rely on. Since the smaller electronics operate on much lower potential voltage than the motors, consideration must be taken to acquire the appropriate voltage. One possible solution is a voltage divider and using the following equation we can analyze the effective resistances required

$$V_{out} = \frac{V_{in} R_2}{R_1 + R_2} \rightarrow 5V = \frac{18V \times R_2}{R_2 + R_1}$$

$$\frac{5}{18} = 0.28 = \frac{R_2}{R_2 + R_1} \rightarrow R_1 = 2.57 R_2$$

Since we need 3.1A to flow through the circuit, and the voltage drop is 18V between the two resistors,

$$V = IR \rightarrow 18V = 3.1A \times (R_1 + R_2) = 3.1A \times (R_2 + 2.57R_2)$$

$$\frac{18}{3.1} = 3.57\ R_2 \rightarrow R_2 = 1.62\ \Omega\ and\ R_1 = 4.11\ \Omega$$

Power is then equal to

$$P = \frac{V^2}{R_{eq}} = \frac{18^2}{1.62 + 4.11} = 56.5W$$

Thus, to use a voltage divider, the power supply must supply an additional 56.5W of power to accommodate for the power lost to heat through the resistor.

Table 1: Electric Specifications of Components

	Quantity	Operating Voltage (V)	Required Current (A)
Encoders	2	5	0.1
Arduino	1	5	0.3
Raspberry Pi 3B+	1	5	2.5
Motors and Motor Drivers	2	18 - 36	2.4
GPS Module	1	5	0.1

Heat Flow:

Heat flow is a large concern for the Raspberry as seen in the table below, as it can reach 90°C when operating at full load. The two methods of managing this large heat buildup would be to either allow for natural cooling with no cooling elements or introducing passive cooling elements, such as a heat sink.

Table 2: Operating Temperatures

Name of devices	Operating temperature
Motor Driver	0°C to 50°C [9]
Stepper Motor	-10°C to 50°C [10]
Encoder	-40°C to 125°C [11]
GPS module	-40°C to 80°C [1

Arduino board	-40°C to 80°C

1. Natural cooling without heat sinks:

Table 3: Thermal Resistance Characteristics of ARM microprocessor [9]

Parameter	Description	°C/W(1)	Air Flow (m/s)(2)
θ_{JC}	Junction-to-case	0.82	N/A
θ_{JB}	Junction-to-board	3.78	N/A
θ_{JA}	Junction-to-free air	11.1	0
	Junction-to-moving air	8.8	1
		8.0	2
		7.5	3
Ψ_{JT}	Junction-to-package top	0.62	0
		0.66	1
		0.66	2
		0.66	3
Ψ_{JB}	Junction-to-board	3.43	0
		3.22	1
		3.12	2
		3.04	3

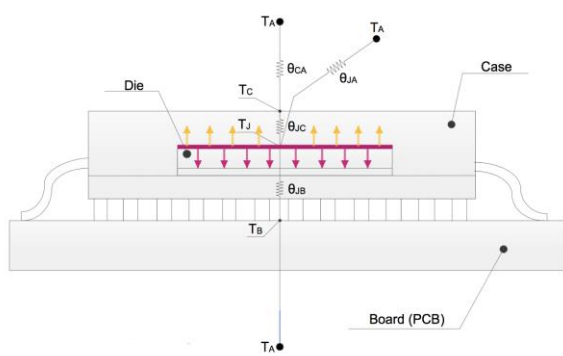


Figure 10: Thermal model of a processor [10]

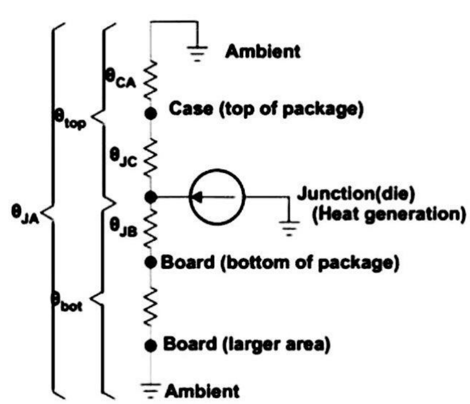


Figure 11: Heat flow of microprocessor

• T_J is the junction temperature

• T_A is the ambient temperature

• P_D is the total power dissipation

• θ_{CA}: Resistance from top of processor through air or heat sink to environment

• θ_{JB} : Resistance from die (junction) to the board (near the processor)

• θ_{JC}: Resistance from die (junction) to the top of the package (case)

from the Table 3,

$$\theta_{\mathrm{CA}} = 11.1\ \mathrm{°C/W}$$

$$\theta_{\mathrm{JB}} = 3.78\ \mathrm{°C/W}$$

$$\theta_{\mathrm{JC}} = 0.62\ \mathrm{°C/W}$$

$$\theta_{\mathrm{JA}} = \theta_{\mathrm{CA}} + \theta_{\mathrm{JB}} + \theta_{\mathrm{JC}}$$

From the instruction of Raspberry Pi, the power consumption of microprocessor is

$$P_D = 5\ W$$

The assumed ambient temperature is

$$T_A = 30\ \mathrm{°C}$$

$$T_J = T_A + \left(\theta_{\mathrm{CA}} + \theta_{\mathrm{JB}} + \theta_{\mathrm{JC}}\right) * P_D = 30 + (11.1 + 3.78 + 0.62) * 5 = 107.5\ \mathrm{°C}$$

The calculated junction temperature is 107.5 °C, which is far over the operating temperature 85 °C. Therefore, a heatsink is considered to attach above the microprocessor.

2. Natural cooling with heat sinks:

Heat sink is attached to the motor controller to dissipate heat. The heat analysis was done to check what the thermal resistance of heat sink should be enough to cool the microprocessor. θ_{JA} is unknown in this analysis.

$$T_J > T_A + \left(\theta_{\mathrm{CA}} + \theta_{\mathrm{JB}} + \theta_{\mathrm{JC}}\right) * P_D$$

$$\theta_{\mathrm{CA}} < (T_J - T_A)/P_D - (\theta_{\mathrm{JB}} + \theta_{\mathrm{JC}})$$

$$\theta_{\mathrm{CA}} > \frac{85 - 30}{5} - (3.78 + 0.62)$$

From the equations above, θ_{CA} should be below 6.6 $°\mathrm{C/W}$ to make sure the microprocessor is within the operating temperature. From the research, the aluminum and copper heatsink both can meet this requirement

Section 1.2: Design Analysis – Component Selection

Motors and Motor Drivers:

In the initial considerations, stepper, DC brushless, and DC brushed motors were considered. The most important specification throughout the selection process was torque, as there were very specific motor torque requirements that needed to be met. Other important specifications were speed and repeatability, as the antennae needs to be able to move to the point at which the satellite is detected and needs to be able to return to the same position each time. Cost is also important as there are strict budget requirements. The decision matrix and result of evaluation can be seen below.

Table 4: Decision matrix of motor specifications

	Speed	Torque	Repeatability	Cost	Total	Weight
Speed	1	0.5	0.5	2	5	0.246
Torque	2	1	3	2	8	0.394
Repeat	2	0.33	1	1	4.33	0.213
Cost	0.5	0.5	1	1	3	0.147
					20.33	**1**

Table 5: Results of evaluation of motors

	Stepper Motor	DC Brushless	DC Brushed
	0.246	0.246	0.246
	0.394	0.394	0.394
	0.213	0.213	0
	0	-0.147	-0.147
Percentage	**85.3**	**70.6**	**49.3**

The results from the decision matrix suggest the stepper motor as the best option. Stepper motors provide a constant holding torque without the need for the motor to be powered and the torque of a stepper motor at low speeds is greater than a DC motor of the same size [11]. Three different standard NEMA sized stepper motors were considered, and comparisons can be seen in Table 6 below.

Table 6: Comparison of motor options

	Nema 34	**Nema 23**	**Nema 17**
Holding Torque	7.07 Nm	3.0 Nm	0.6 Nm
Step Angle	1.8 deg	1.8 deg	1.8 deg
Operating Voltage	-	4.2 V	10 V
Weight	3.13 kg	1.6 kg	0.5 kg
Cost	$125.92	$49.86	$22.09

The NEMA 17 motors were selected because they easily met the minimum torque requirements for the worst-case scenario of both the azimuth and elevation motors, and the difference in cost was extremely helpful to staying within budget.

Encoders:

Since the stepper motor is also a positional transducer itself, it technically needs no source of feedback on the positioning of the gears in the elevation system. However, to ensure accuracy, the following encoders were compared.

Table 7: Comparison of encoder options

	Cui AMT20 Series	**Broadcom AR18 Series**	**HEDM-55xx**
Encoder Type	Absolute	Optical	Optical
Resolution	0.08˚ (12 bits)	1.4˚ (8 bits)	0.35˚
Accuracy	±0.2˚	±0.2˚	-

Serial Protocol	SPI	SSI 3-Wire	SPI 3-Wire
Maximum Speed	8000 RPM	15000 RPM	30000 RPM

Comparing the three encoders that were considered, the main differences are the resolutions, the maximum speed and the communications protocols. For the purposes of this project, high angular resolution is important and operating are much less so. Finally, the communication protocol (SPI) works very well with Arduino as an interface as all Arduino models are equipped with at least one SPI port, making integration simple.

GPS:

A GPS module was needed to be able to locate the ground station's coordinates, to be used in the calculation of where the orbiting CubeSat is in reference to the ground station. This is used in determining the angle at which the antenna needs to be pointed. There were two main considerations for selection of the GPS – startup time and accuracy. Within startup timing, the cold start is the time it takes for the GPS to get the first fix if it has moved several hundred kilometers. The hot start time is if the receiver has been off for less than an hour of time. Sensitivity of a GPS can be affected by clock errors, ephemeris errors, and atmospheric effects [12]. A few GPS modules were compared as seen below.

Table 8: Comparison of GPS module options

	NEO-6M	**NEO-7M**	**EM-506**
Cold Start	27 s	30 s	35 s
Hot Start	1 s	1 s	1 s
Sensitivity	-161 dBm	-161 dBm	-163 dBm
Compatibility	Arduino, RPI	Arduino, RPI	Arduino
Price	$20-40	$90	$40

The GPS module options were extremely similar. The NEO-6M was selected fully based on the slightly better cold start and the significantly better price.

Power Supply:

For power, a wall powered supply unit would need to be used as the constant power draw from the ground station and remoteness of the station would mean that a portable power supply such as a battery would not be able feasible since maintenance would be a large issue. The choices were narrowed down to either a computer power supply with various power rails or a laptop charger with a DC-DC converter to achieve desired voltage requirements.

Table 9: Comparison of Power Supply options

	EVGA 600	200W Laptop Charger
Input Voltage	120 VAC	120 VAC
Output Voltage	12V, 5V, 3.3V	18V
Output Power	600 W	200 W
Converter	In-built	Needs External
ESA Approval	Approved	Approved
Price	$70	$60

For the power supply, the two choices were very similar in most regards, mainly in price and being ESA approved, which was a very important deciding factor, however, the EVGA provides various power rails whereas the laptop charger has one single voltage. This is deterring because to achieve the 5V necessary for the electronics, a voltage divider or buck converter would need to be used. As previously analyzed however, if a voltage divider is used approximately 50W of power is wasted as heat, so the EVGA power supply was selected.

Controller:

For controlling the individual electronic components, a microcontroller is needed to process the data from the system and provide meaningful outputs that ensure the system behaves as intended. The following microcontrollers were analyzed for effectiveness in providing the desired outputs

Table 10: Comparison of controller options

	Raspberry PI 3B+	Arduino Uno	PIC Microcontroller

RAM	1 GB	2 KB	256 KB
Flash Memory	-	32 KB	1 MB
IO Pins	40	20	62
Clock Speed	1.4 GHZ	16 MHz	180 MHz
Internet Capabilities	Yes	No	No
Cost	$50	$25	$30 (w/o headers)

From the above table, the Raspberry Pi easily outperforms the other two microcontrollers in terms of RAM and clock speed, making it easily the best choice for use in this project because it is able to do many complex calculations in quick succession (such as orbital prediction). One caveat is that the Raspberry Pi is optimized for certain peripherals such as keyboards and LCD displays, and not so much motors and small sensors. To bypass this issue, Arduino Unos will be used in a master-slave configuration with the Pi as they are optimized for running simple tasks in a loop and thus better suited for handling general IO tasks while the Pi is better for handling task management and heavy computations.

System Architecture:

The system currently is comprised of many smaller electronics with one main processing unit (the Raspberry Pi). To effectively process data and make sure that the system is communicating effectively, a framework must be implemented to ensure the smooth processing of data. Two different methods were researched, Serial and I2C communications (with no framework) and ROS.

The I2C and Serial Communication approach provides and easy implement, fast solution as Arduino and Raspberry Pi's all come with the necessary ports required for these communications protocols. Raspberry Pi also has the PiConfig tool and Serial Communications package which are specifically meant to aid this type of communication. The main disadvantage with this method is that only simple tasks can be run since there are only enough pins to allow one I2C device to be connected. Additionally, even though serial connections can made, given the sheer number of tasks and data being collected, the buffer would fill up too quickly and processes would need to wait for previous tasks to complete first given the non-preemptive implementation that Arduino uses.

From the considerations above, the core architecture was selected to be based on Robotic Operating System (ROS). There are three main parts in the system architecture: the control of devices through the Arduino board, CubeSat orbit prediction using Simplified General Perturbations models (SGP4 Python 1.4), and CubeSat location. As seen in Figure 12 below, the three parts work together as an individual node – the core ROS Master node.

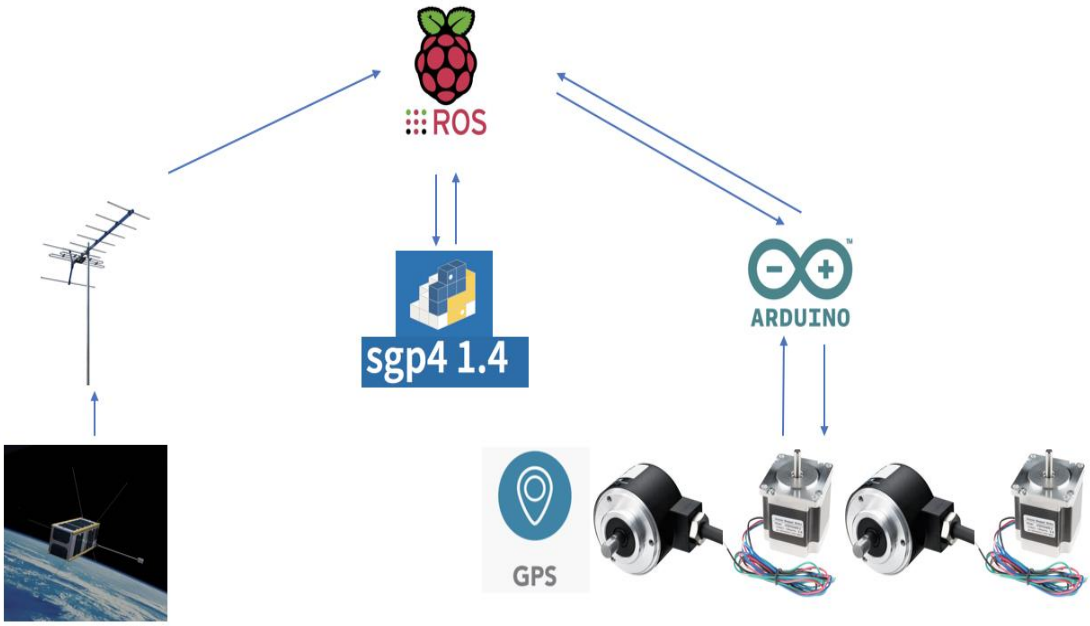


Figure 12: System architecture diagram

Bearings:

Table 11: Ball bearings

Bearing Type	Ball
For Load Direction	Radial
Construction	Single Row
Seal Type	Open
Inner Ring Type	Standard
Ball Bearing Type	Standard
Trade No.	R8
For Shaft Type	Round
For Shaft Diameter	1/2"
ID	0.5"
ID Tolerance	-0.0003" to 0"
For Housing ID	1 1/8"

OD	1.125"
OD Tolerance	-0.0004" to 0"
Width	1/4"
Width Tolerance	-0.005" to 0"
Ring Material	Steel
Ball Material	Steel
Cage Material	Steel
Radial Load Capacity, lbs.	
Dynamic	1,100
Static	530
Maximum Speed	25,500 rpm
Lubrication	Required

The ball bearings were selected based on their internal and external diameter. The rated static and dynamic loads were much higher than required, as was the rated maximum speed.

Table 12: Thrust Bearing Specifications

System of Measurement	Inch
Bearing Type	Ball
For Load Direction	Thrust
Seal Type	Open
For Shaft Type	Round
Ball Bearing Type	Standard
For Shaft Diameter	1 1/4"
ID	1.253"
ID Tolerance	0" to 0.009"

OD	1 7/8"
OD Tolerance	-0.012" to -0.001"
Thickness	0.437"
Thickness Tolerance	-0.01" to 0.01"
Ball Material	Steel
Cage Material	Nylon Plastic
Washer Material	Steel
Thrust Load Capacity, lbs.	
Dynamic	95
Static	540
Maximum Speed	2,500 rpm
Lubrication	Required
Temperature Range	-40° to 220° F

The thrust bearings required an inner diameter greater than diameter of the shaft used (0.5 inch) and less than the outer diameter of the round bar used to make the bearing holder (3 inch). Thrust bearings that met these criteria made from steel were significantly more expensive. Therefore, Nylon thrust bearings were used. This bearing offered a dynamic and static load much greater than the required 50 lbs and a maximum speed that was also higher than required (1 rpm).

Retaining Rings:

Table 13: Retaining Ring Specifications

Retaining Ring Type	External
Retaining Ring Style	Standard
System of Measurement	Inch
Material	1060-1090 Spring Steel

Finish	Black Phosphate
For OD	1/2"
For Groove	
Diameter	0.468"
Diameter Tolerance	-0.002" to 0.002"
Width	0.039"
Width Tolerance	0" to 0.003"
Ring	
ID	0.461"
ID Tolerance	-0.005" to 0.002"
Thickness	0.035"
Thickness Tolerance	-0.002" to 0.002"
Min. Hardness	Rockwell C40
Thrust Load Capacity	1,670 lbs.

The retaining rings were selected based solely on the diameter of the shaft as the thrust load capacity of all retaining rings were significantly larger than required.

3D Printing Filament:

Table 14: 3D Printing Filament Specifications

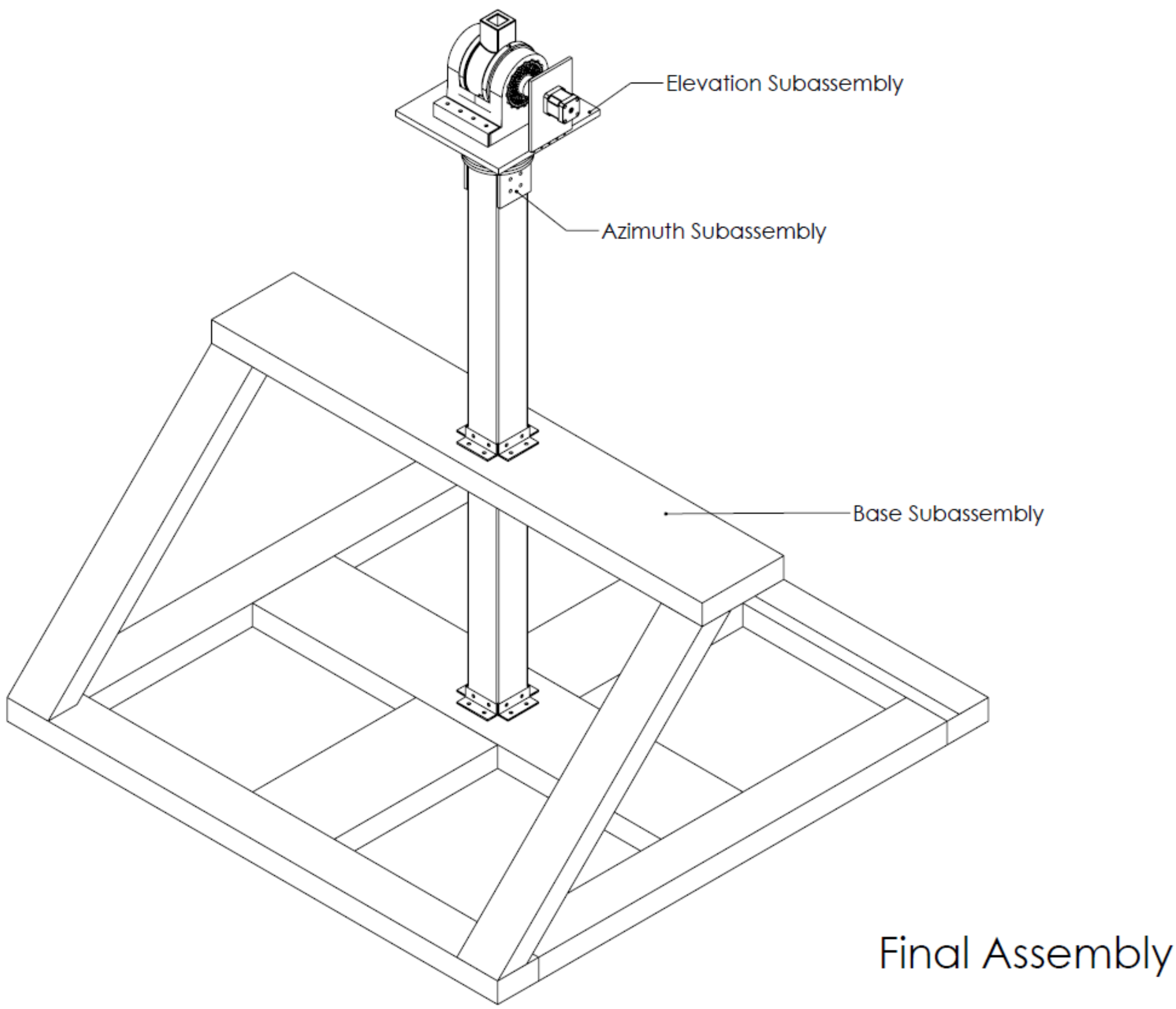
Property	
Glass Transition Temperature	60 °C
Maximum Temperature: Mechanical	50 °C
Density	1.3 g/cm^3
Young’s Modulus	3.5 GPa
Ultimate Tensile Strength	59 MPa

Yield Strength	70 MPa
Filament Diameter	1.75 mm
Dimensional Accuracy	±0.03mm
Printing Temperature	190 °C – 220 °C

The 3D printing filament required needed to be slightly flexible as the ring gears need to be slightly flexed in order to ensure proper contact between the gears in the gear box. Due to this, ABS, even with its higher strength, was not a suitable material. Similarly, Nylon was also too rigid for this application. PLA was the only material with the flexibility and strength for this application. HATCHBOX PLA was selected as it has a high dimensional accuracy.

Section 2: Complete Detailed Design Documentation

Full Model Drawings:



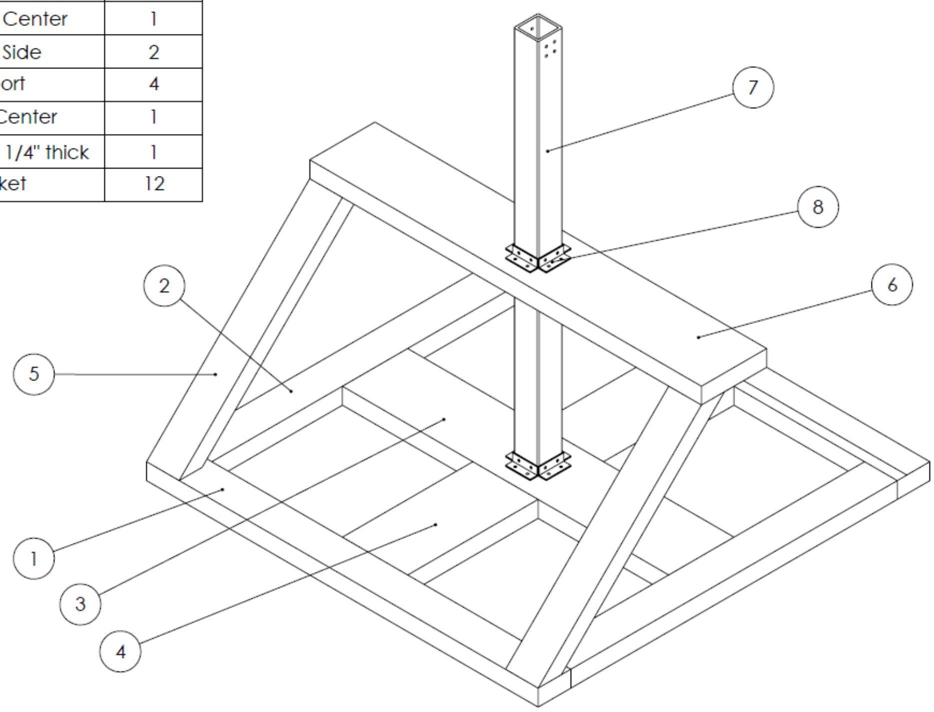
The three main subassemblies are shown above.

Note: Tolerances for all wood components are ±0.25” unless otherwise stated.

Tolerances for all angle brackets are ±0.05” unless otherwise stated.

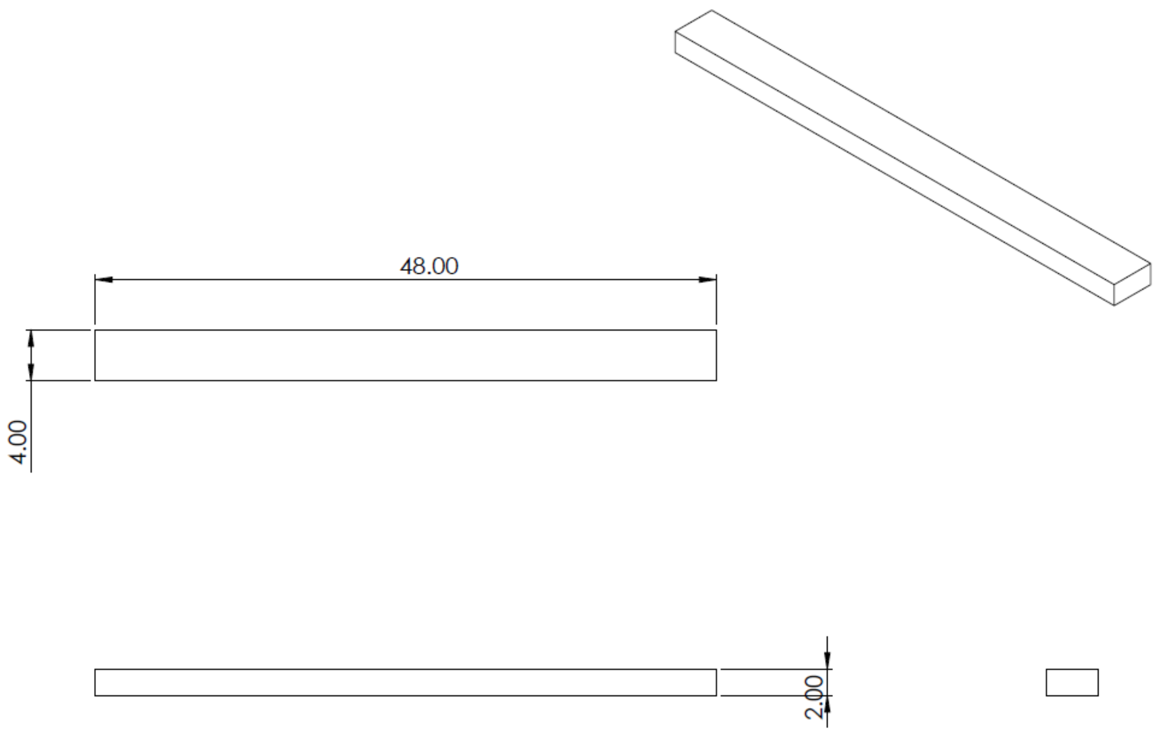
Tolerance for all metal components are ±0.01” unless otherwise stated.

ITEM NO.	PART NUMBER	QTY.
1	2" x 4" for base	2
2	2" x 4" for base short	2
3	2" x 8" Base Center	1
4	2" x 8" Base Side	2
5	2" x 4" Support	4
6	2" x 8" Top Center	1
7	3"x 3" Tube, 1/4" thick	1
8	Angle Bracket	12



Base Subasssembly

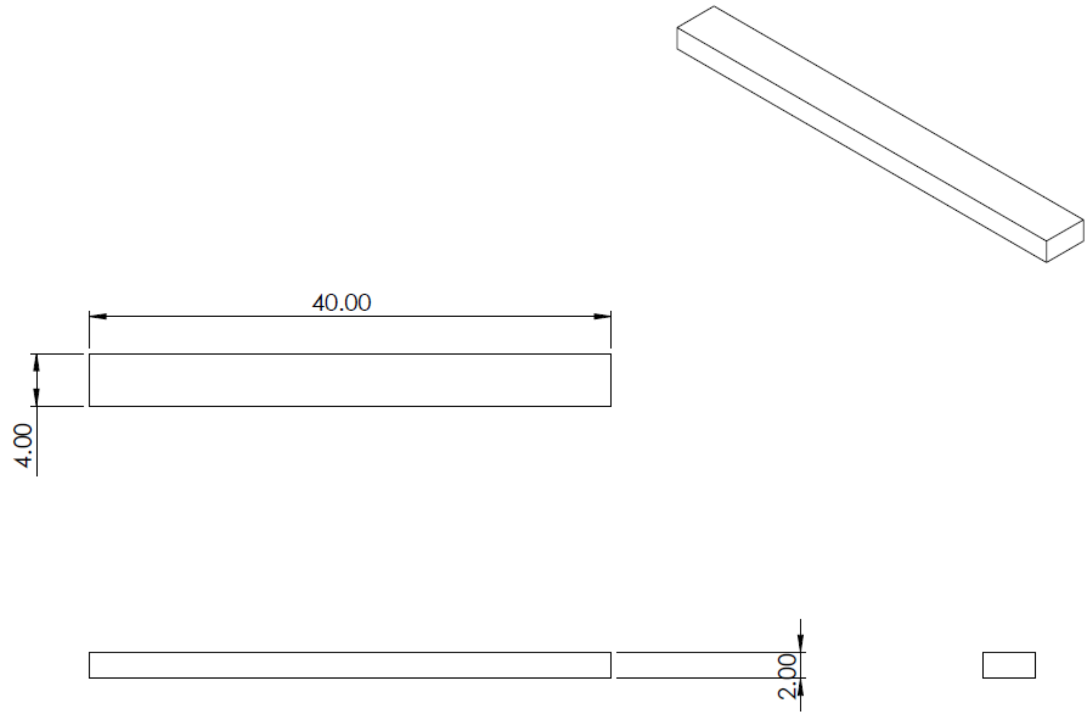
The base subassembly is made from 2”x4” and 2”x8” pieces of pine lumber along with angle brackets to connect the 3”x3” Square Tube. The following drawings show the dimensions of each of the pieces in the Base Subassembly.



2X4 FOR BASE

All Dimensions in INCHES
Quantity: 2

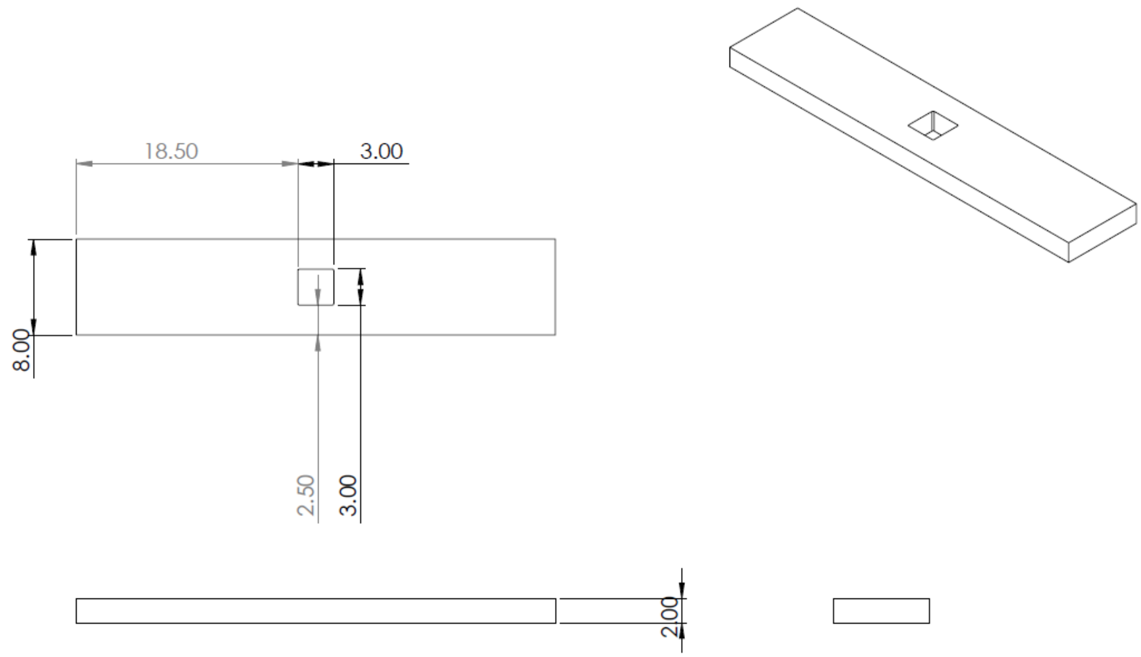
2”x4” piece of lumber. Longest piece of 2”x4” required for this project.



2X4 FOR BASE SHORT

All Dimensions in INCHES
Quantity: 2

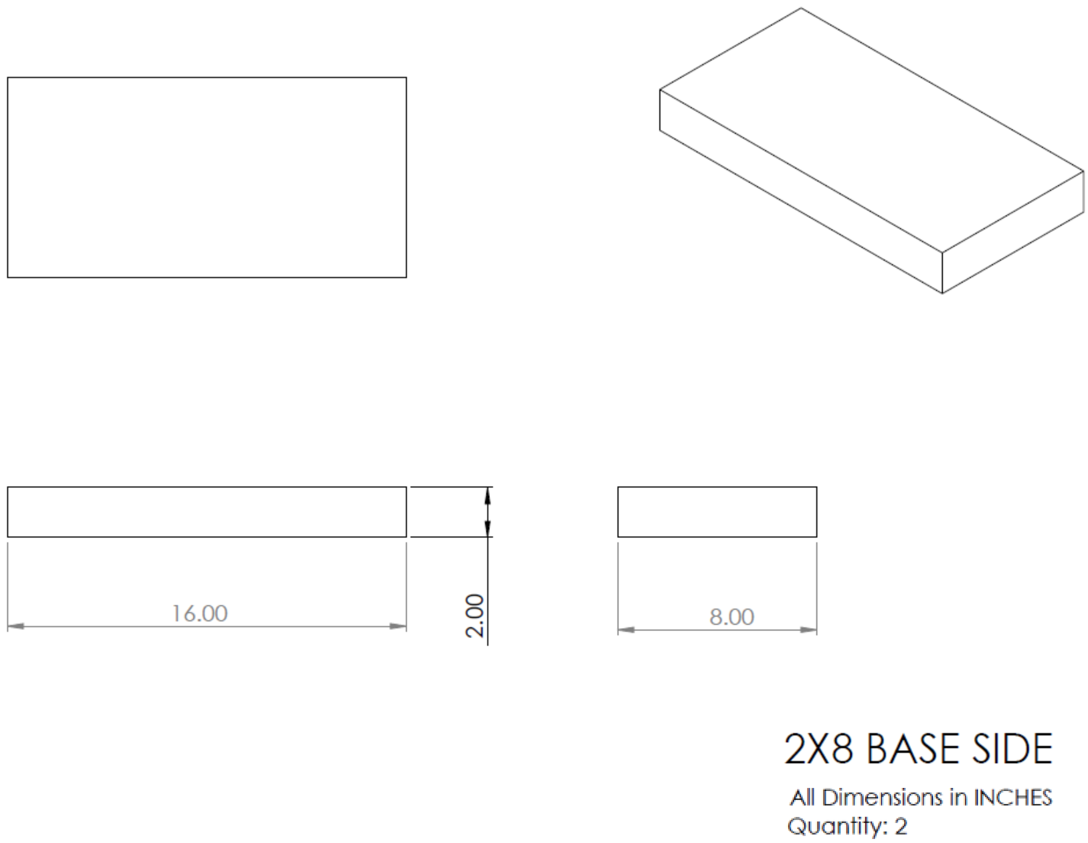
2”x4” piece of lumber. Shorter than the previous piece of 2”x4” in order to form a square box base.



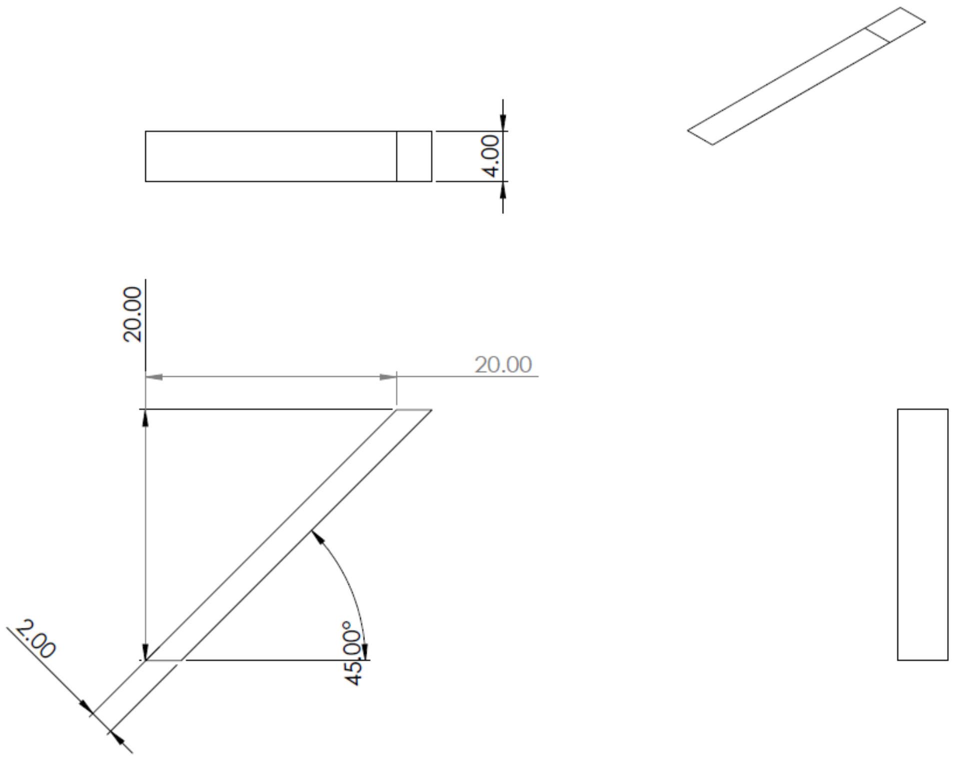
2X8 BASE CENTER

All Dimensions in INCHES

2”x 8” piece of lumber. Used to hold the center pole in place with the use of 4 angle brackets that go around the hole and are screwed into the wood using self-drilling screws.



2”x 8” piece of lumber. Two piece are used on either side of the 2” x 8” Base Center to provide more rigidity to the base.

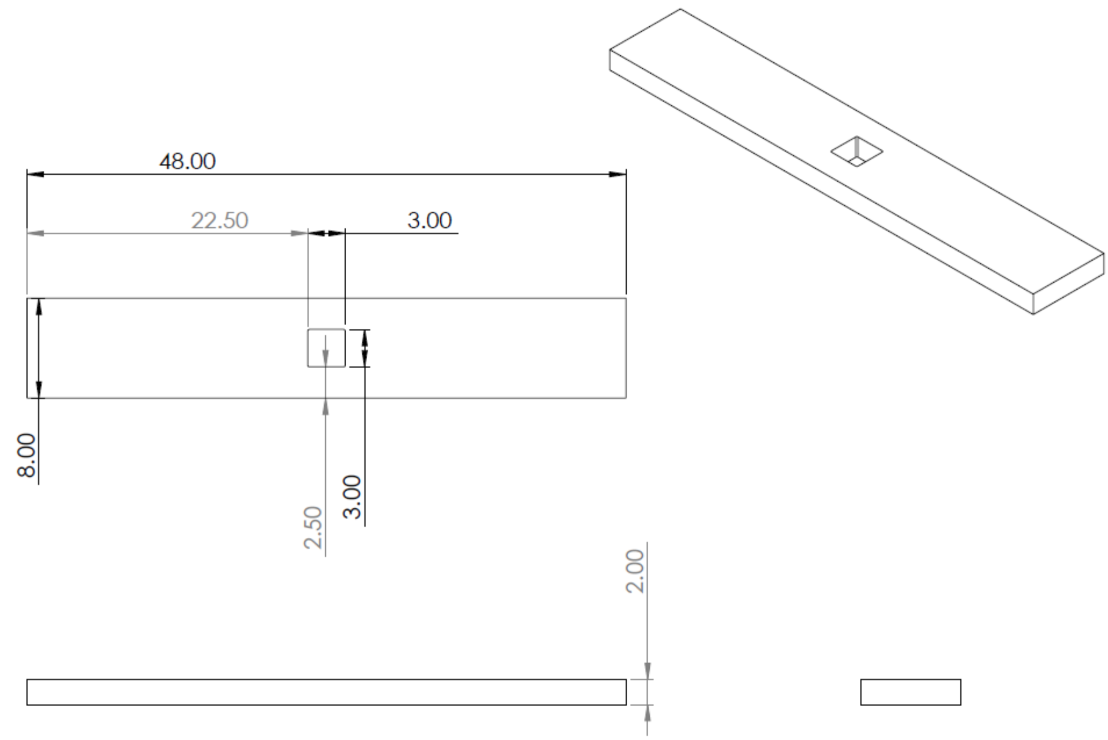


2X4 SUPPORT

All Dimensions in INCHES

QUANTITY: 4

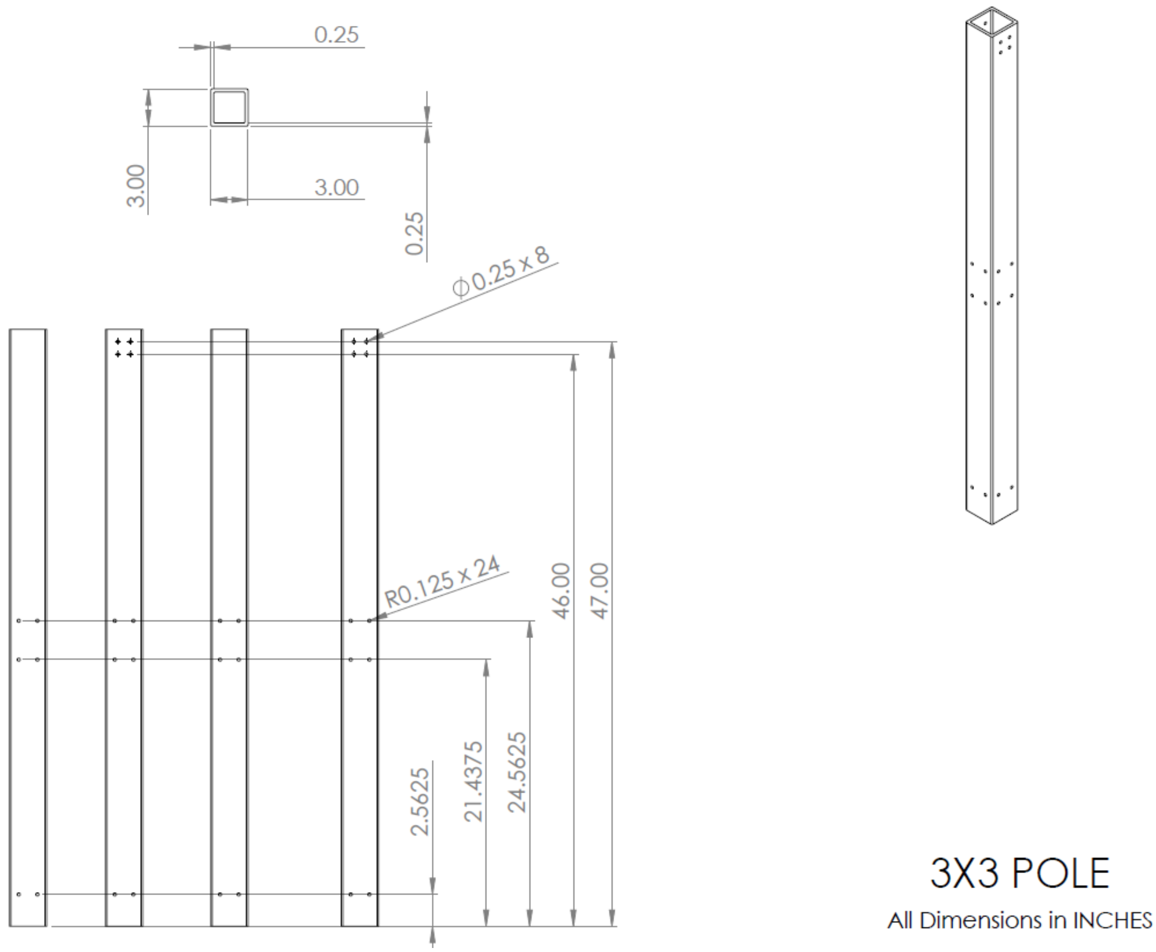
2”x4” piece of lumber used hold the top 2” x 8” Top Center at a height of 20 inches.



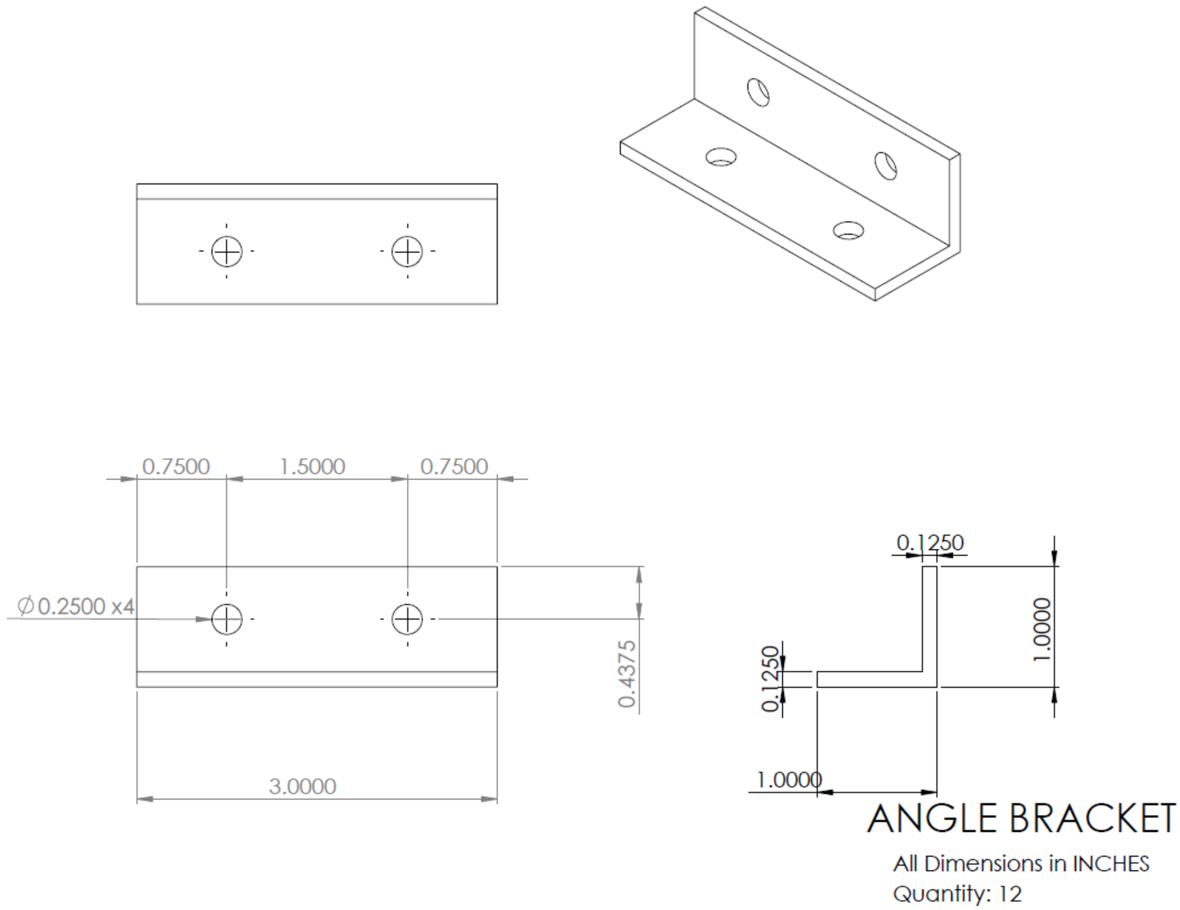
2X8 TOP CENTER

All Dimensions in INCHES

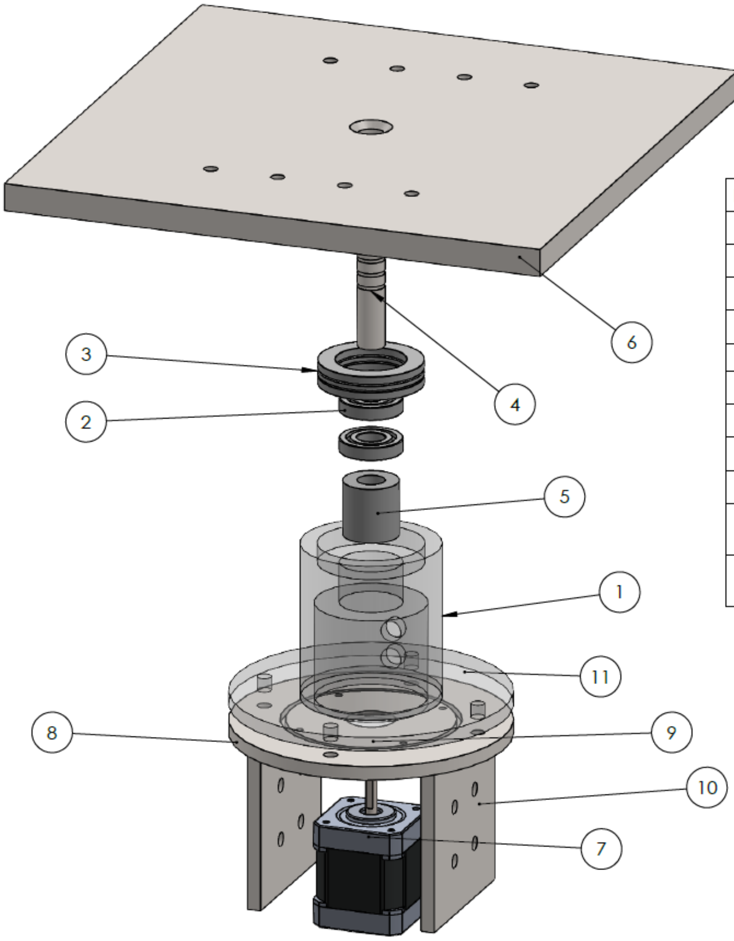
2”x 8” piece of lumber. Like the 2” x 8” base center, this piece is used to hold the center pole in place with the use of 8 angle brackets that go around the hole (4 on the top and 4 on the bottom) and are screwed into the wood using self-drilling screws.



3"x3", ¼" thick square tube used to connect the base subassembly to the elevation and azimuth subassemblies. The bottom 24 1/4th inch holes are used to attach the angle brackets on the base to the pole. These holes need to be threaded as it is not possible to install nuts in these locations. The 8 1/4th inch holes are used to connect the pole to the azimuth subassembly. These holes do not need to be threaded as it is possible to use nuts for these holes.



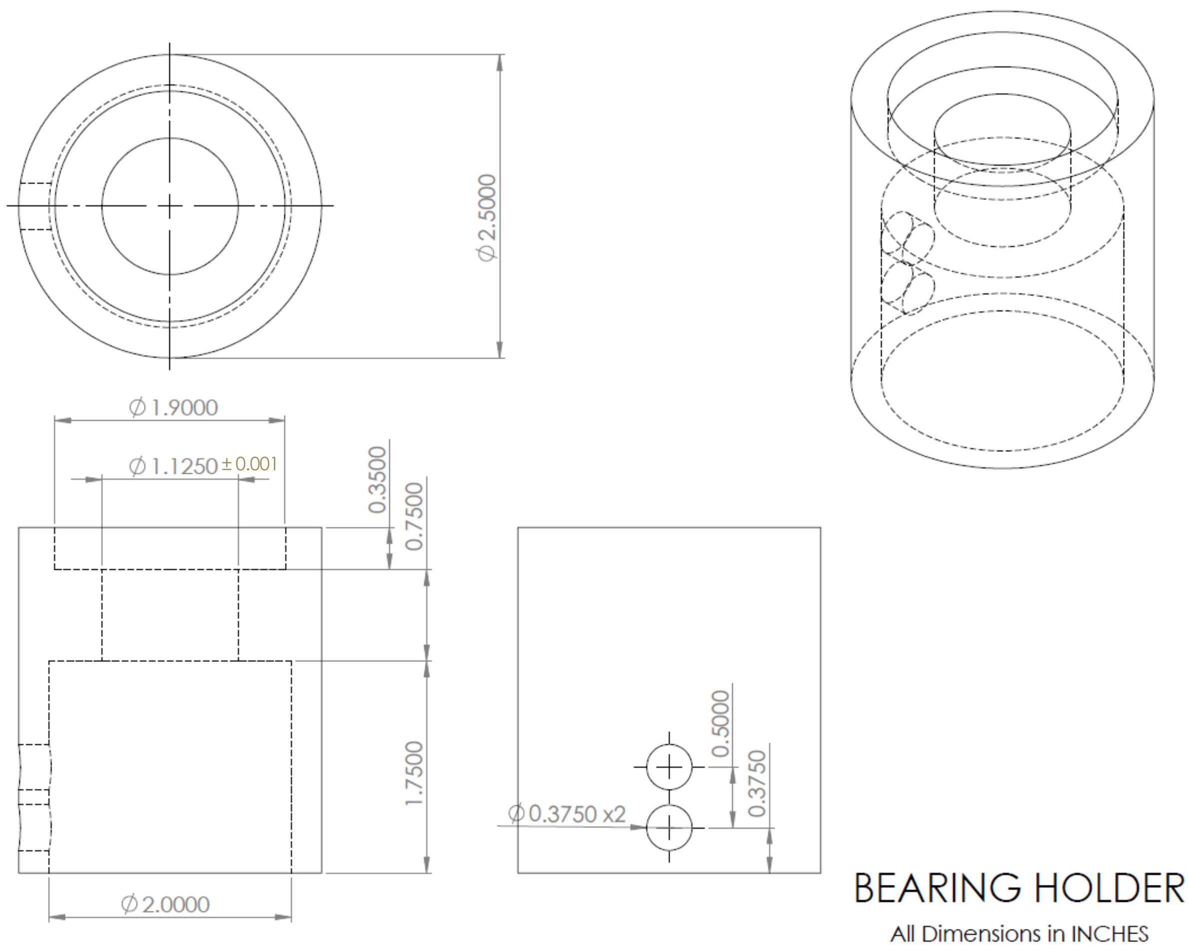
1 inch angle brackets used to connect the base to the pole. 1/4th inch nuts are used to connect these to the pole and 1/4th inch self-drilling screws are used to connect these to the wood.



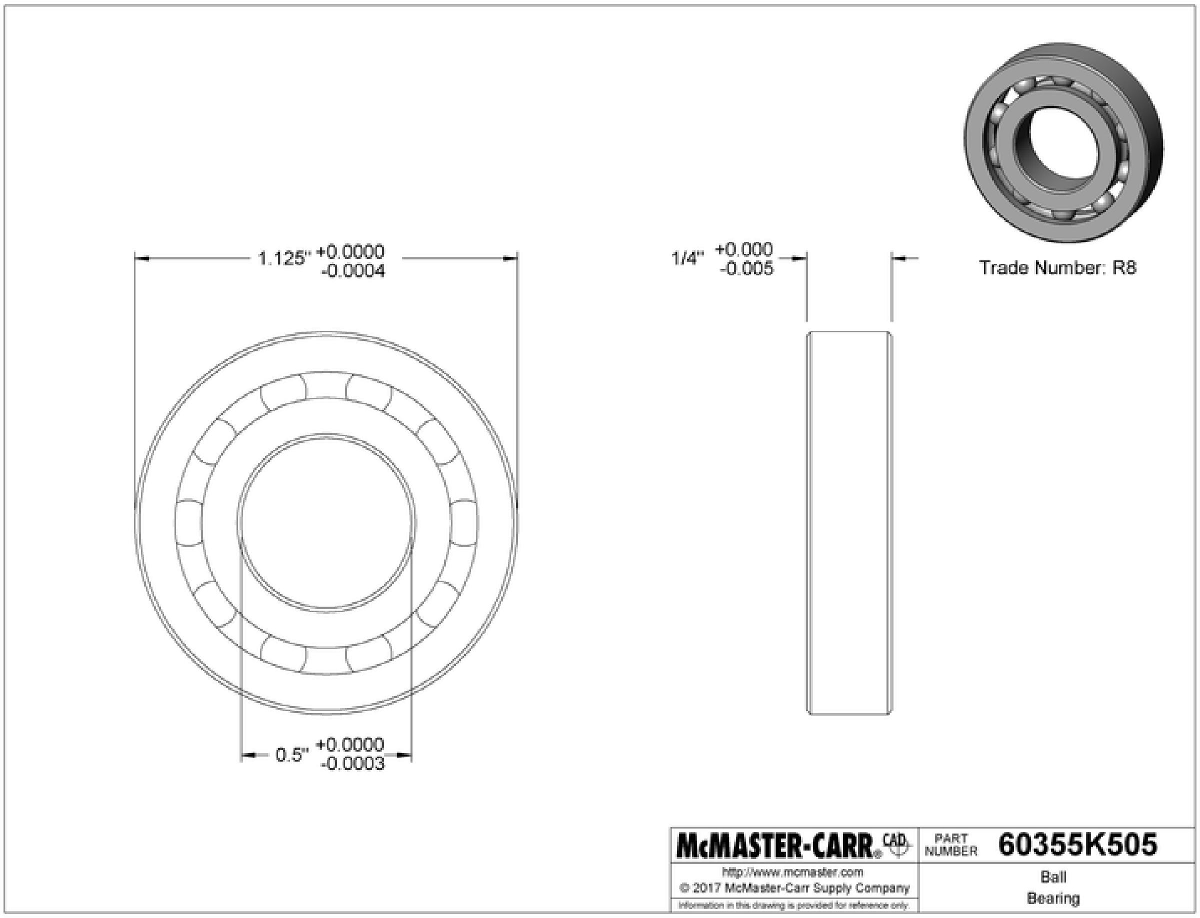
Item Number	PART NUMBER	QTY.
1	Bearing Holder	1
2	Ball Bearing	2
3	Thrust Bearing s	1
4	Shaft	1
5	Motor Coupler	1
6	Base	1
7	Nema 17	1
8	Motor Holder Flange	1
9	Motor Holder	1
10	Motor Holder Flange Plates	2
11	Bearing Holder Flange	1

Azimuth Subassembly

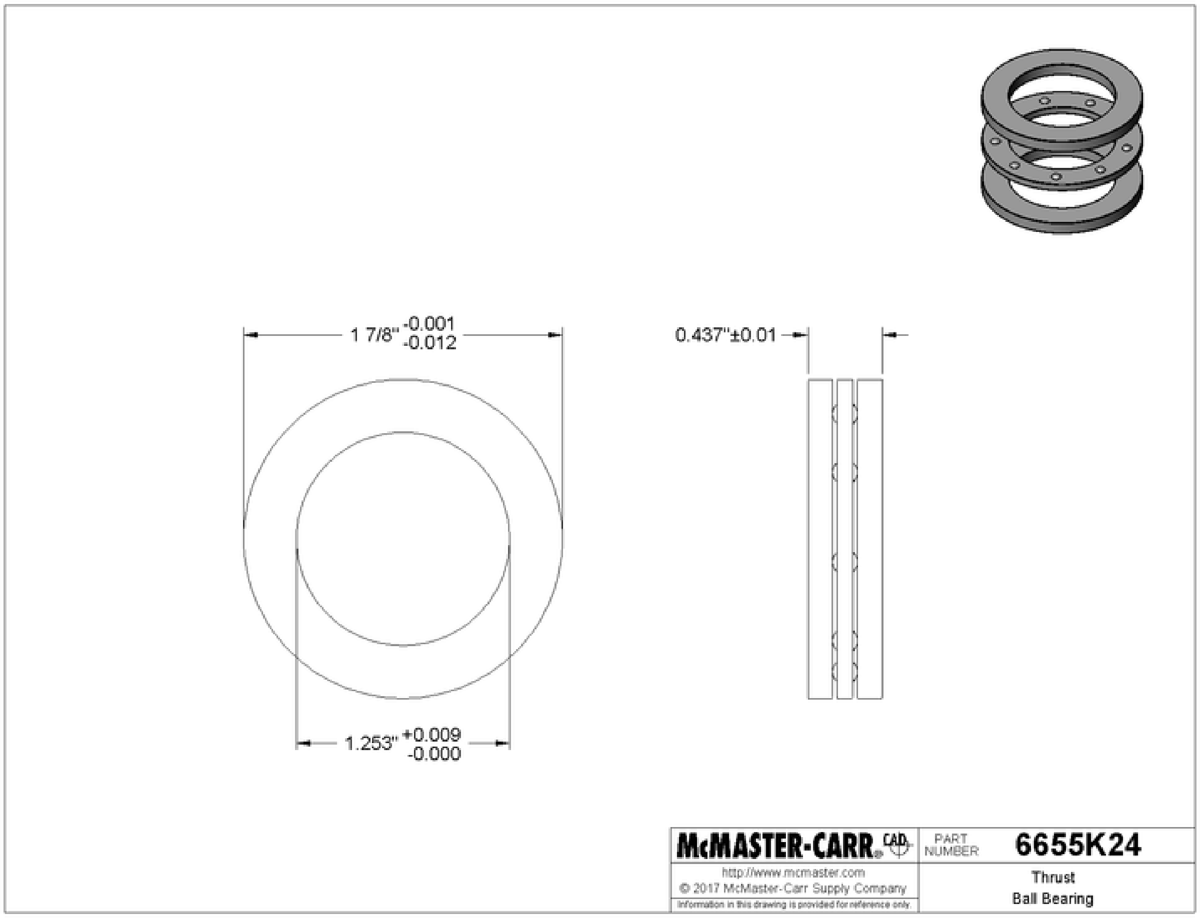
An exploded view of the Azimuth Subassembly along with a Bill of Materials for the subassembly.



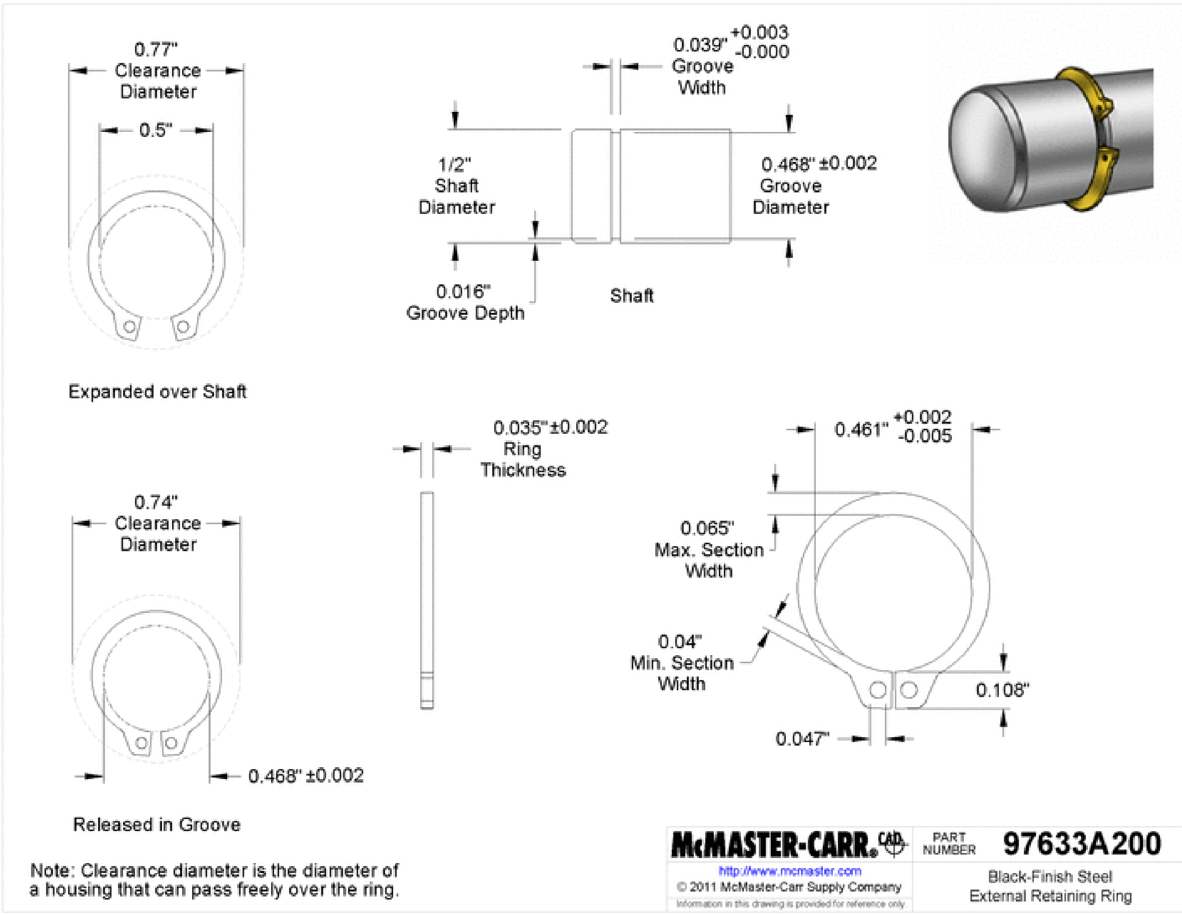
Used to hold the two ball bearings, held in place via press fit on the bearing holder, and the thrust bearing. The holes on the side allow access to the motor coupler for assembly and maintenance.



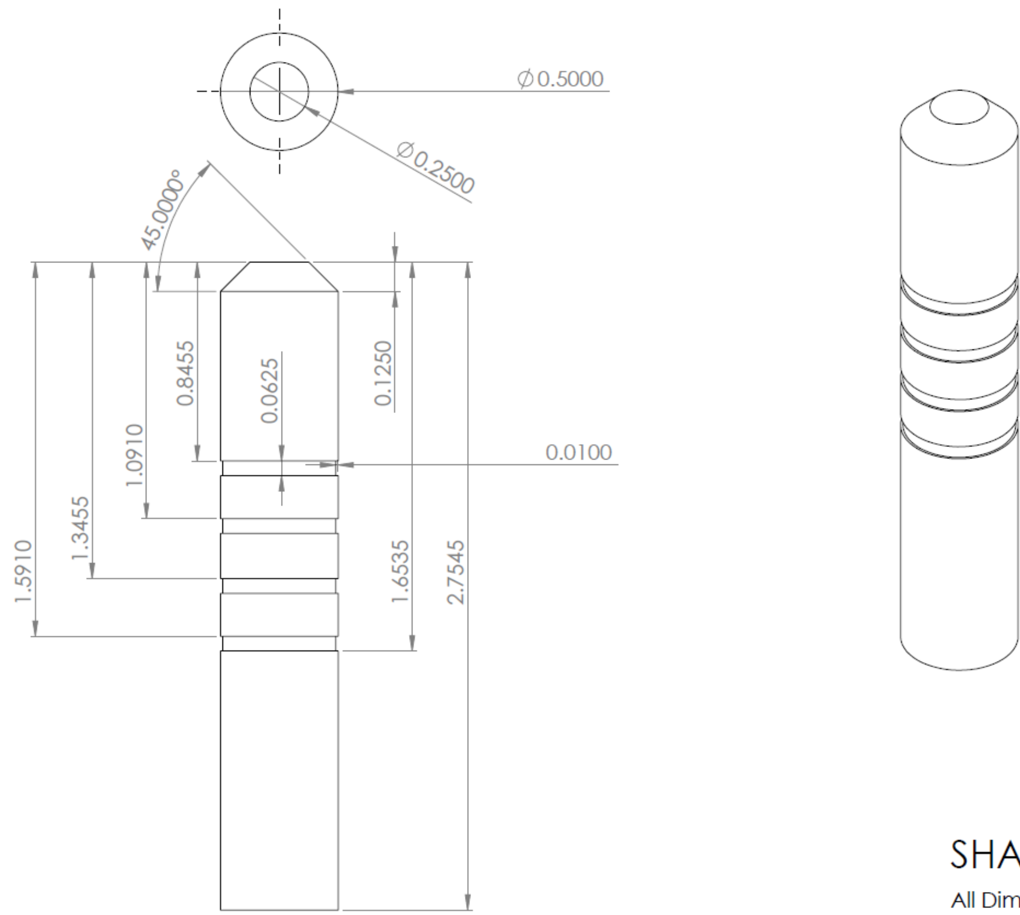
Ball bearing drawing from McMaster Carr (https://www.mcmaster.com/60355k505).



Thrust bearing drawing from McMaster Carr. (https://www.mcmaster.com/6655k24)

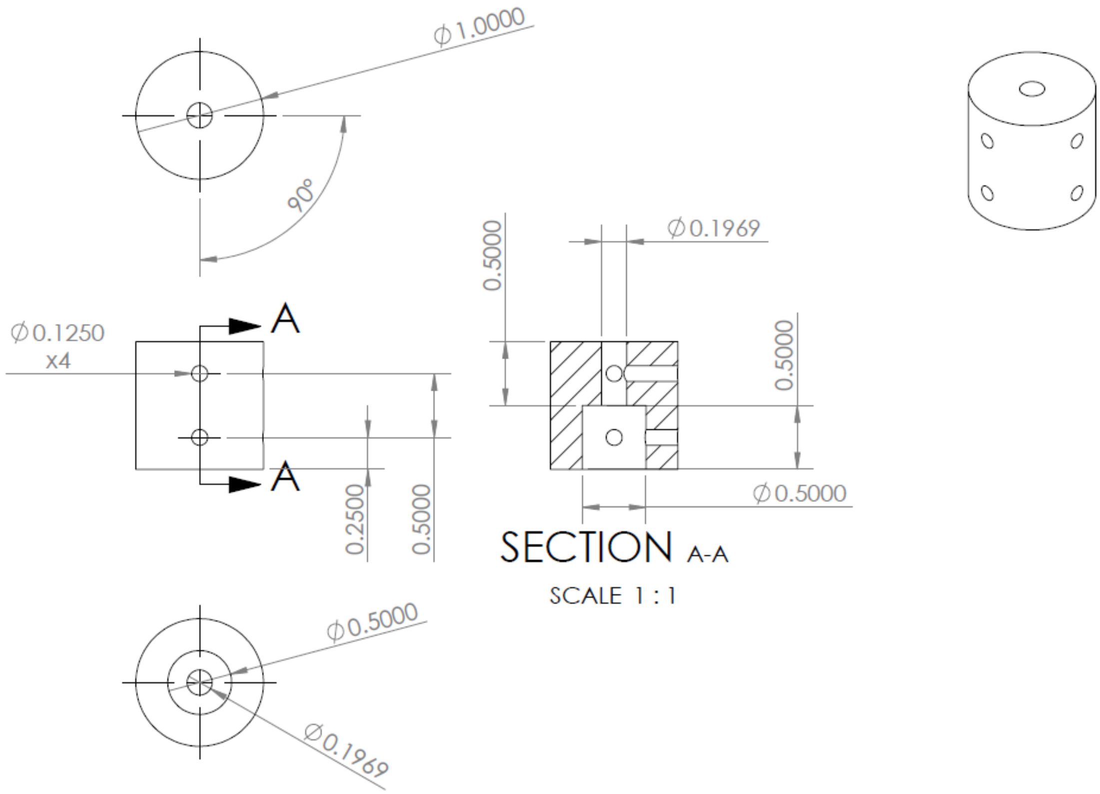


Retaining Rings from McMaster-Carr. (https://www.mcmaster.com/97633a200) Quantity required: 4.

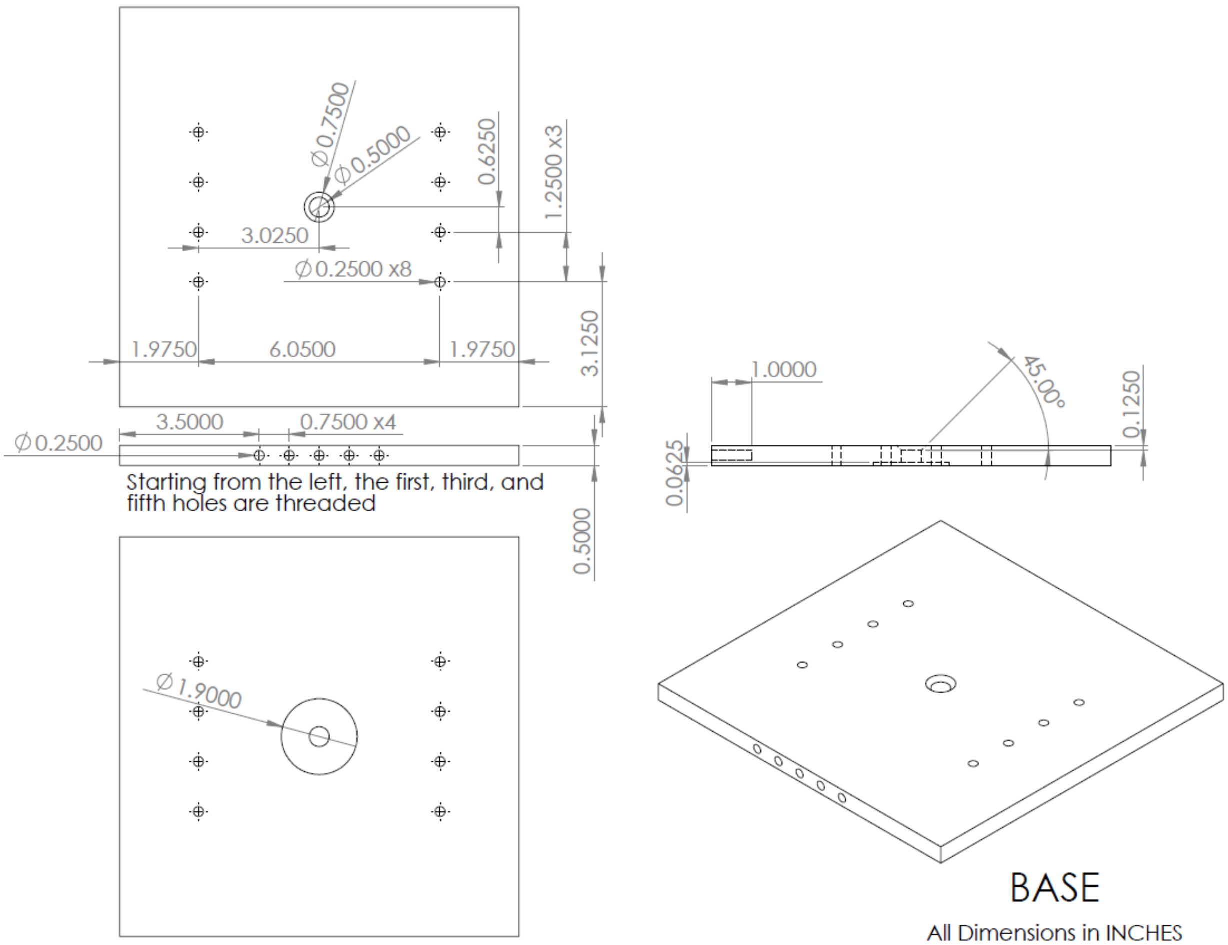


SHAFT

All Dimensions in INCHES

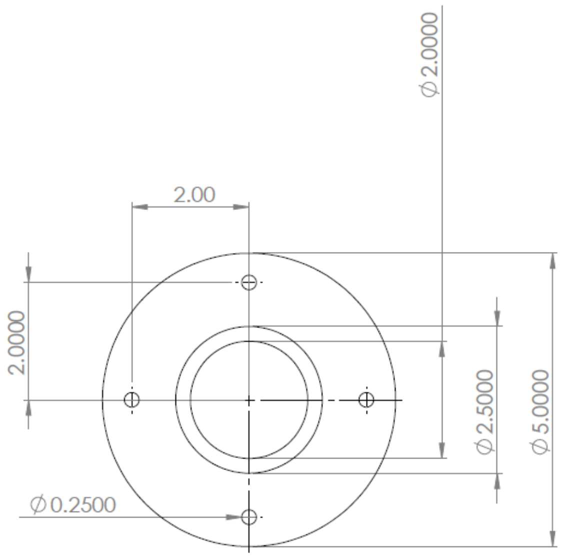


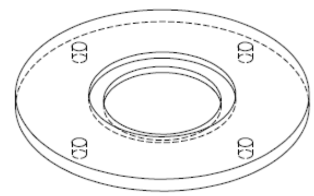
Coupler
All Dimensions in INCHES

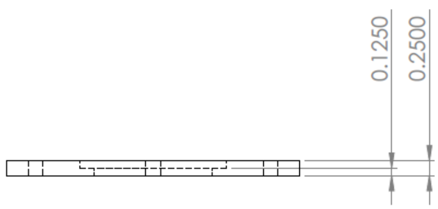


BASE

All Dimensions in INCHES

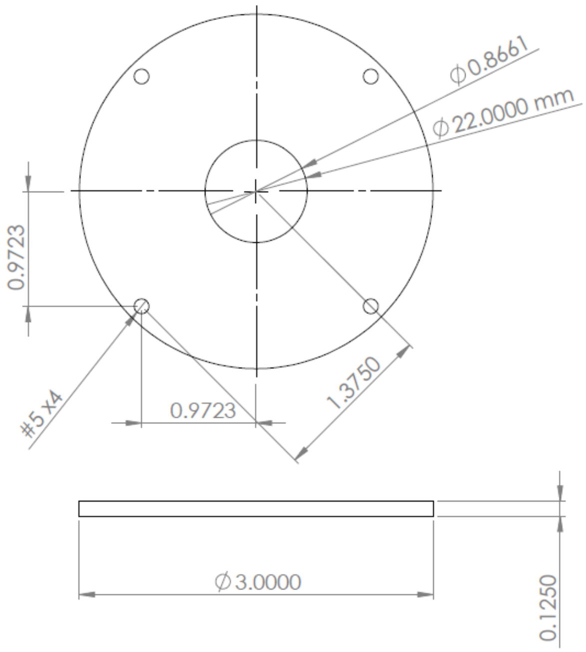


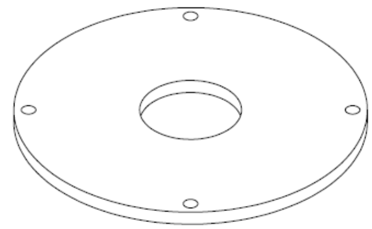




MOTOR HOLDER FLANGE

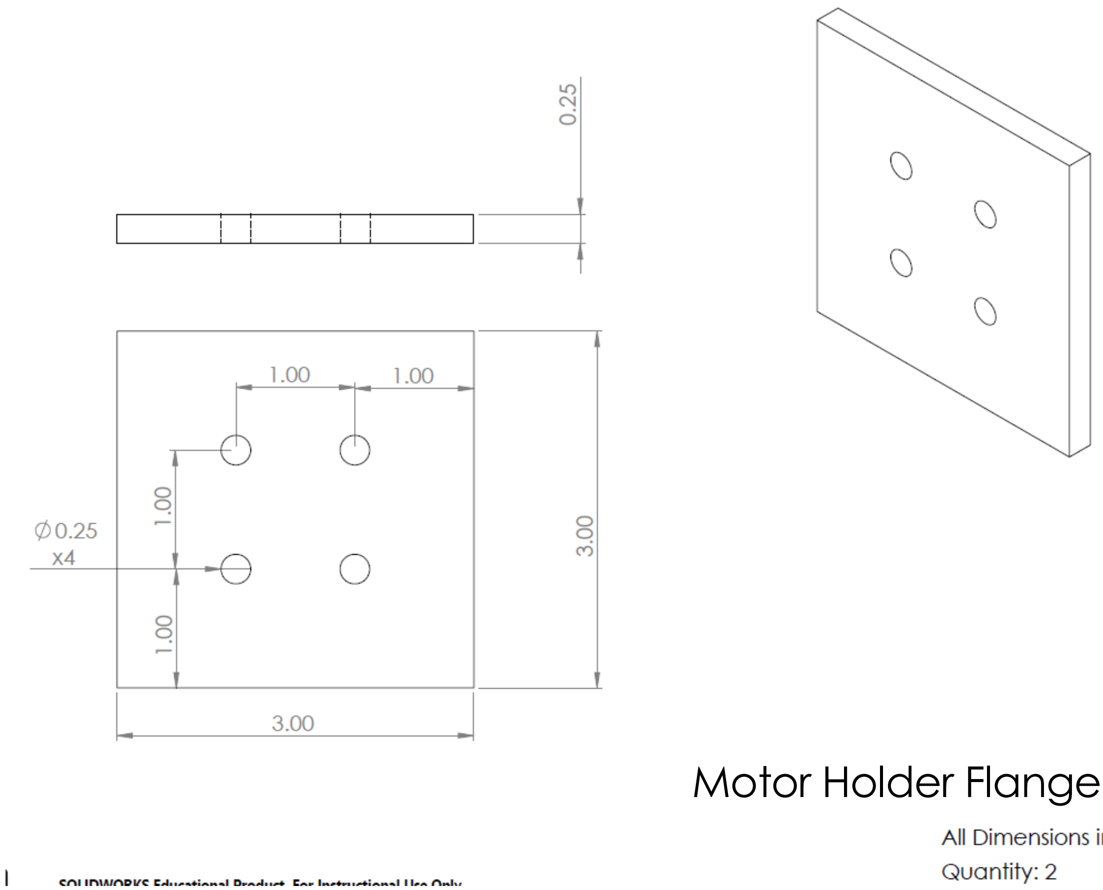
All Dimensions in INCHES





Motor Holder

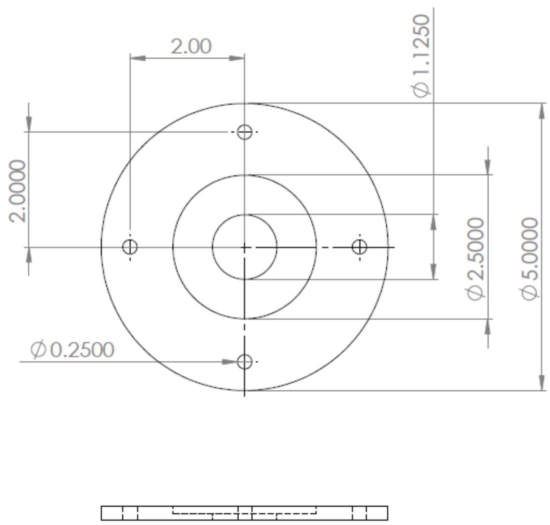
All Dimensions in INCHES

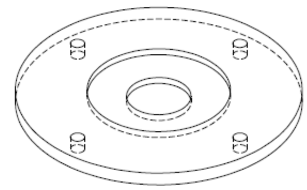


Motor Holder Flange Plates

All Dimensions in INCHES

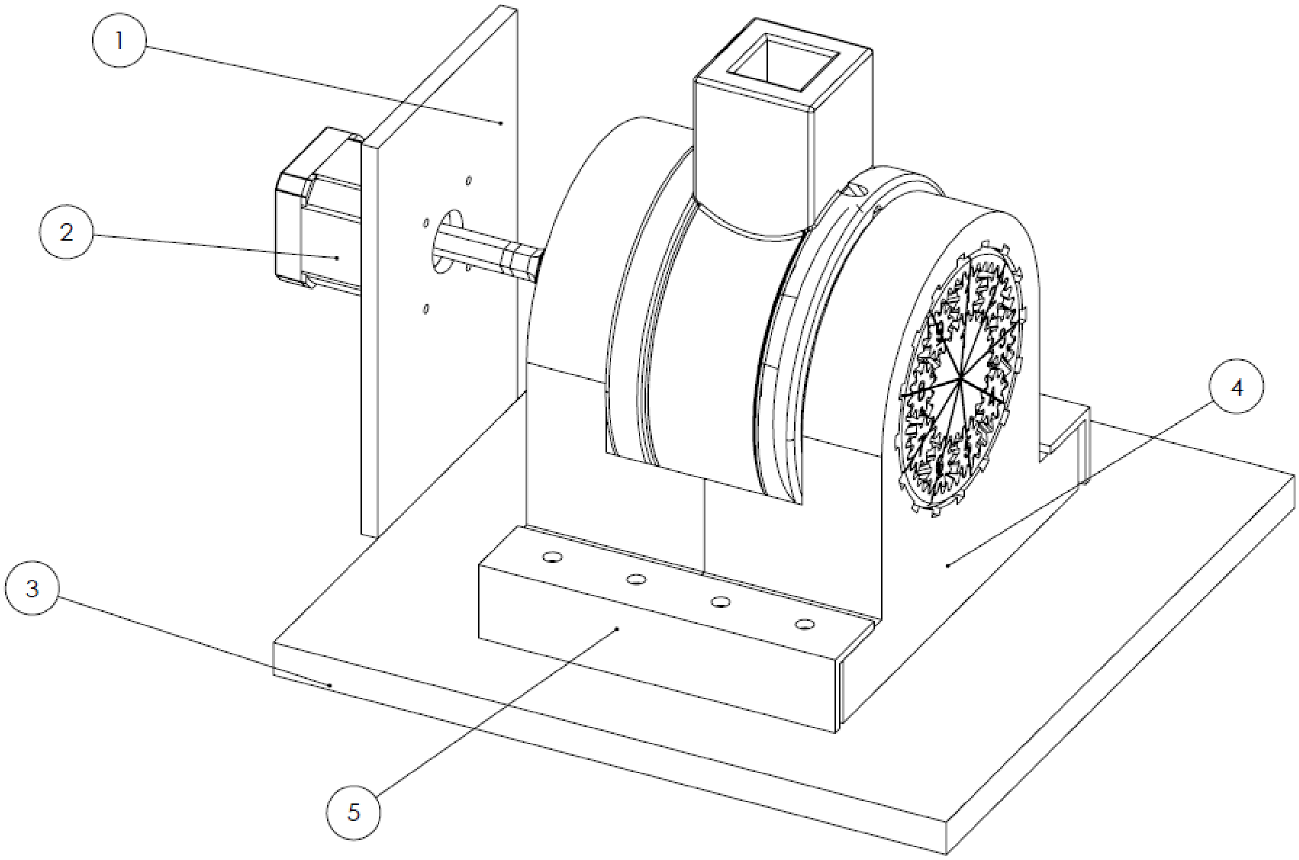
Quantity: 2





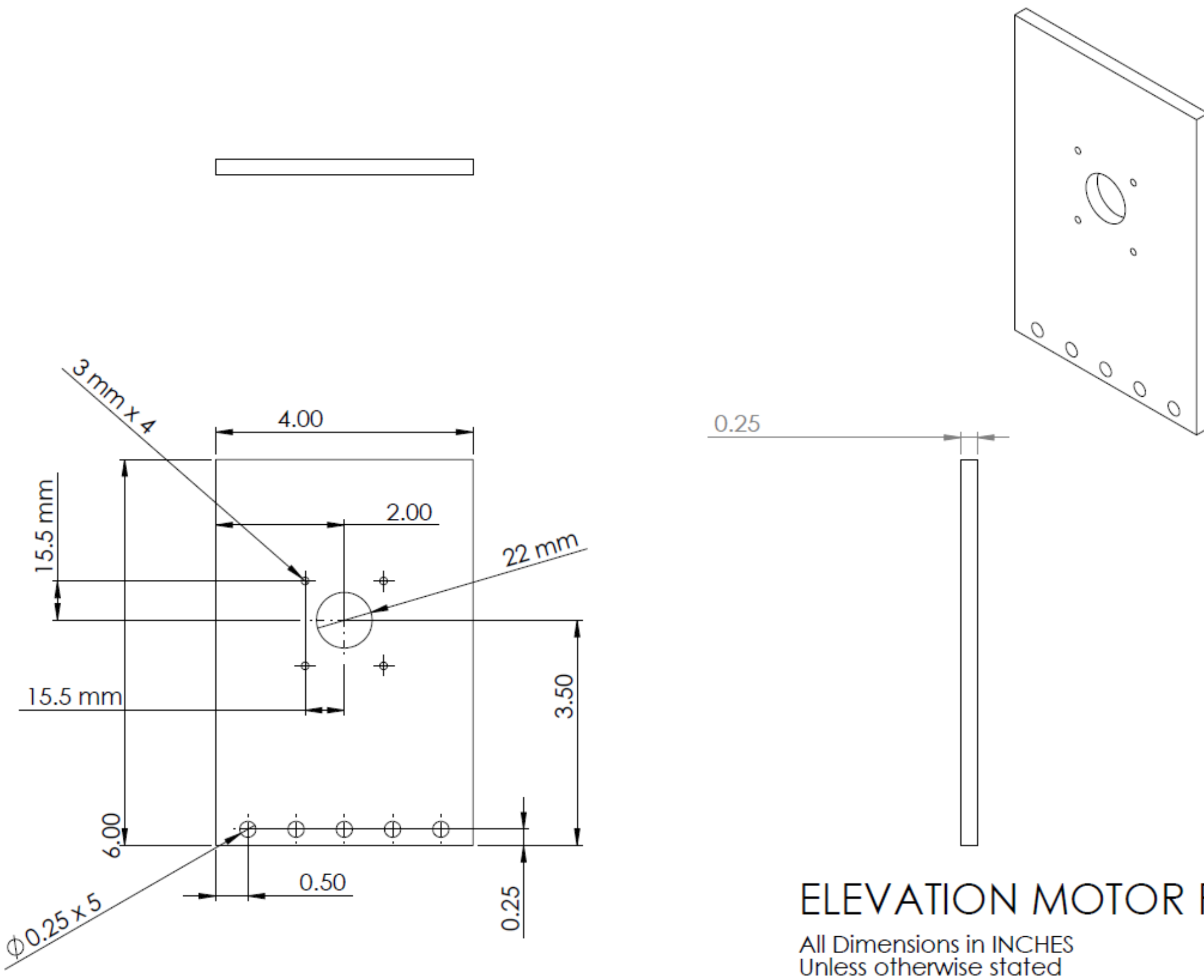
Bearing Holder Flange

All Dimensions in INCHES



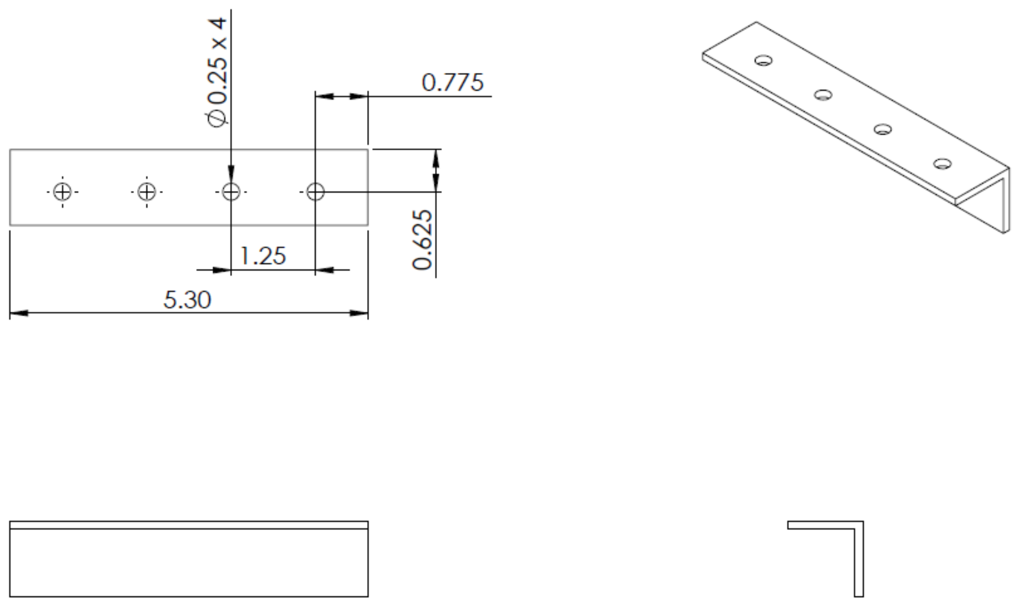
Item Number	Name	Quantity
1	Motor Plate	1
2	Nema 17	1
3	Base	1
4	GearBox	1
5	Angle bracket for gearbox	2

Elevation Subassembly



ELEVATION MOTOR PLATE

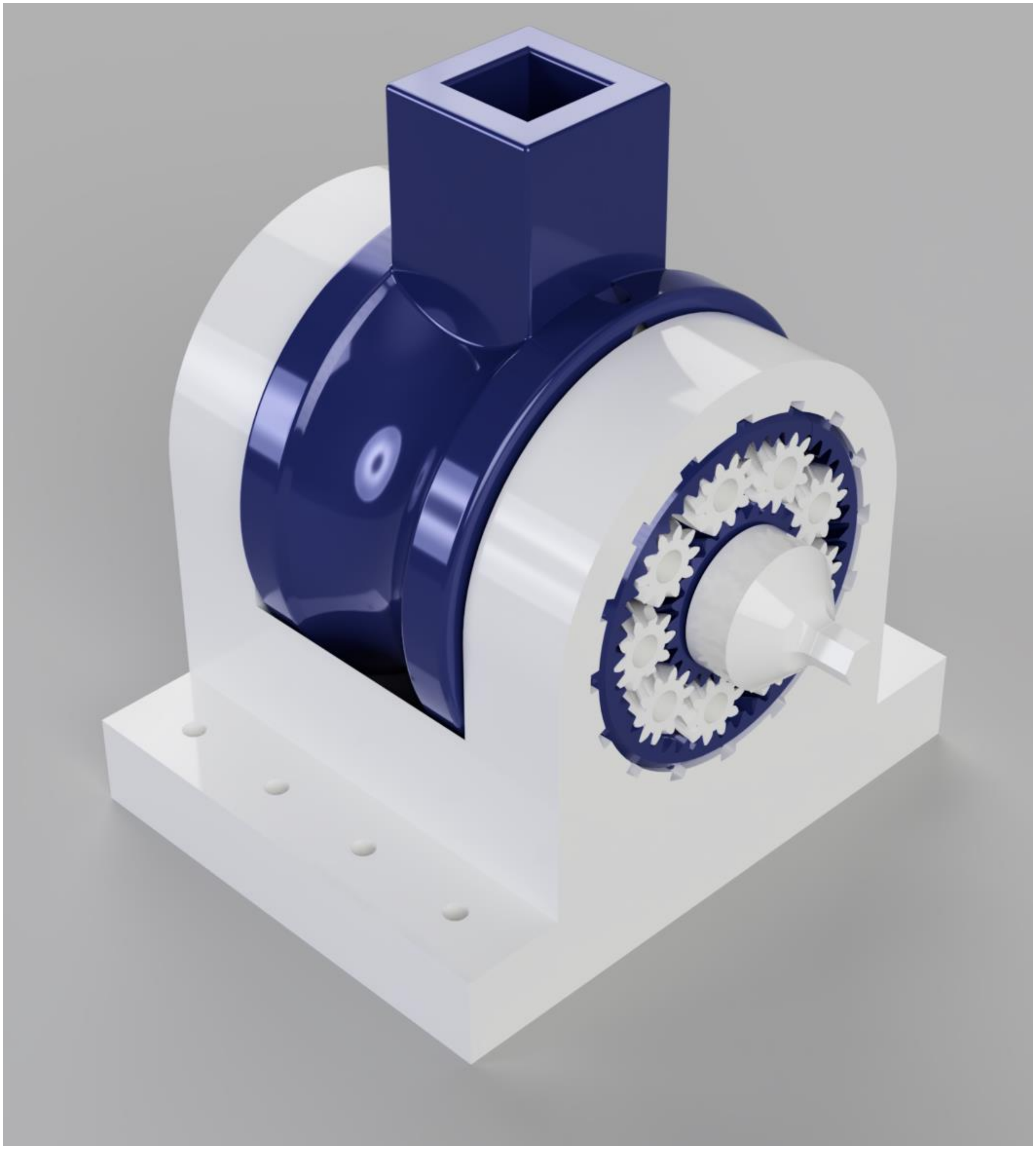
All Dimensions in INCHES
Unless otherwise stated



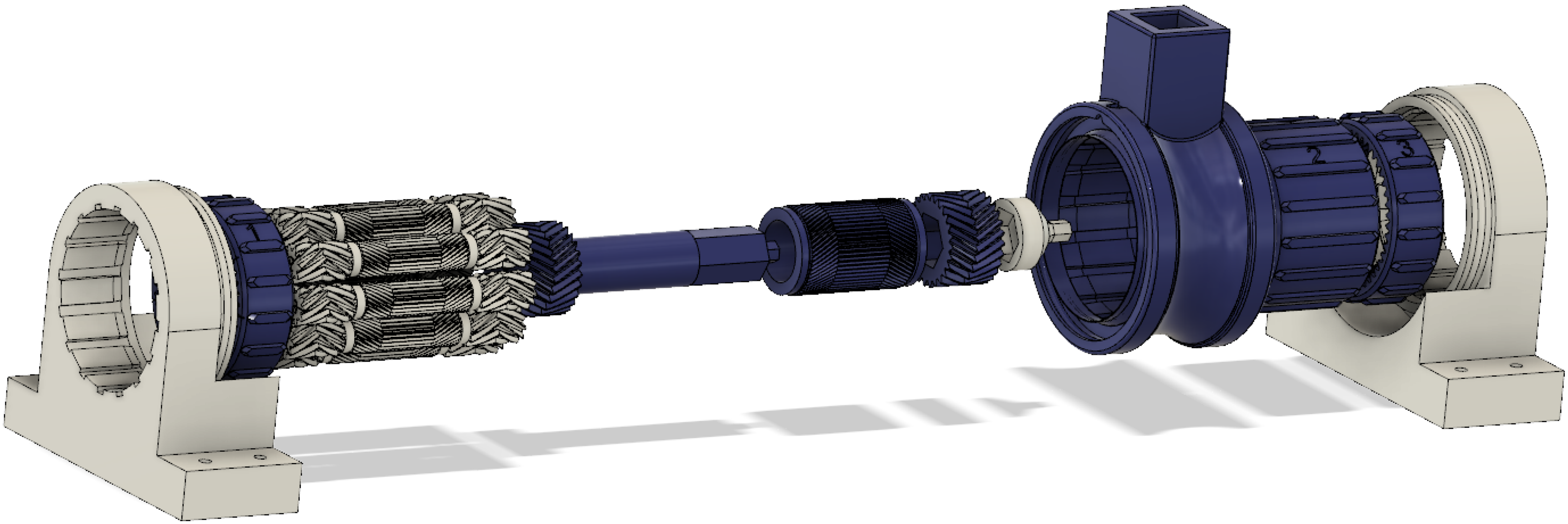
ANGLE BRACKET FOR GEARBOX

All Dimensions in INCHES

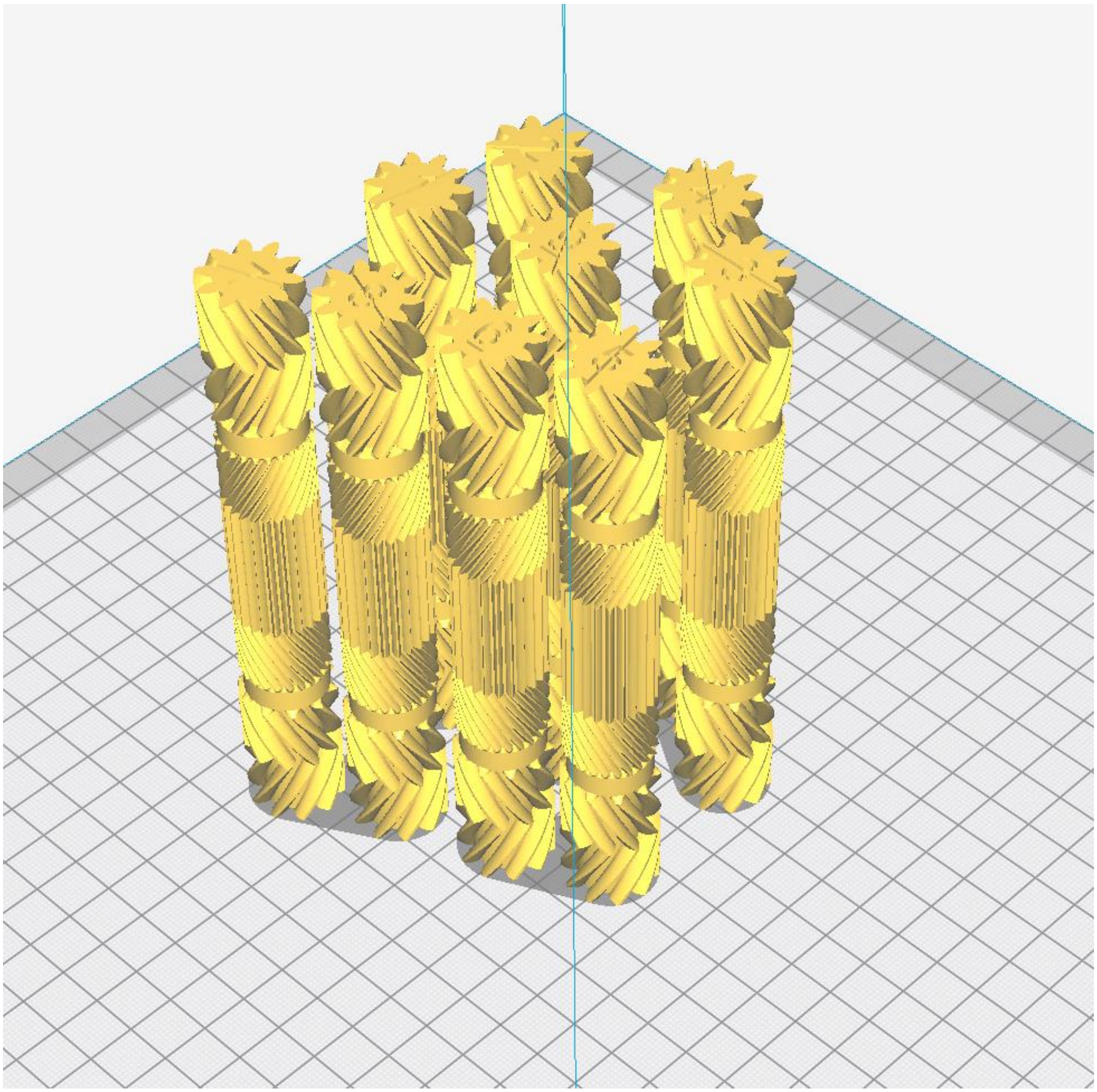
3D Printed Gear Box:



Exploded View:

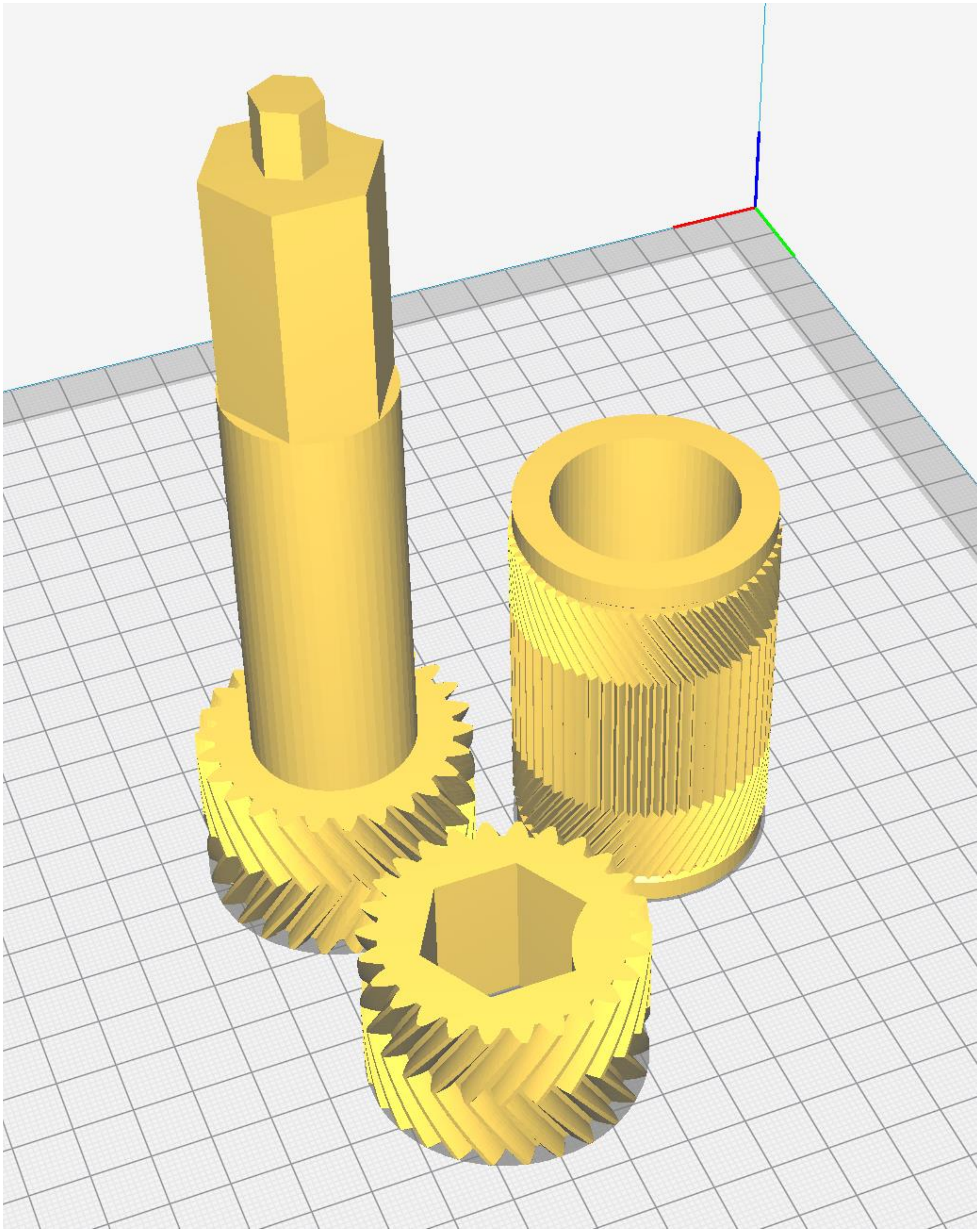


As the printing orientation matters for print strength and surface quality of the gears in the gearbox, the following pictures illustrate the printing orientation for different components of the Gears.

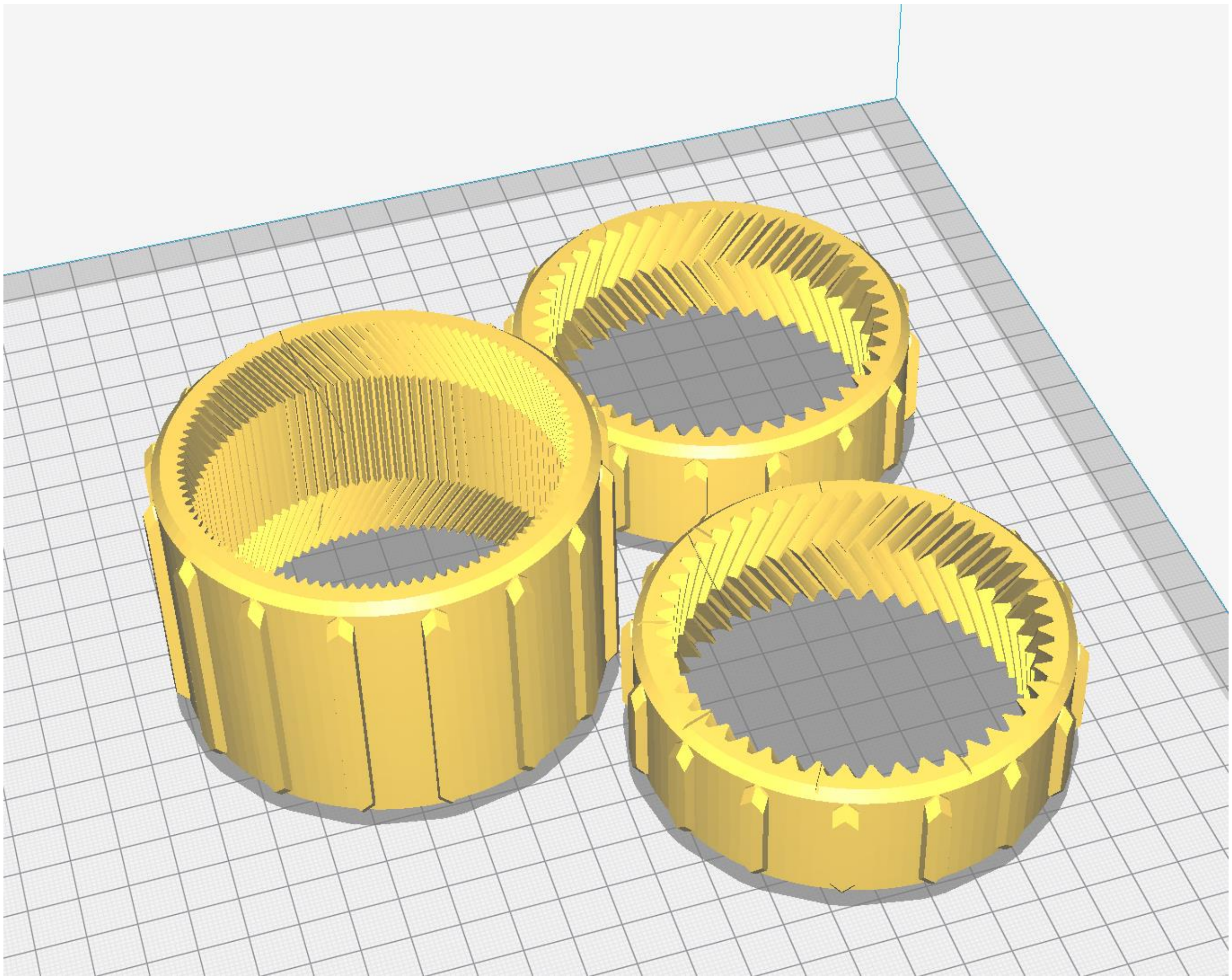


All 9 Planetary Gears, printed upright with the numbers facing up.

Approximate Print Time: 1 day, 12 hours.

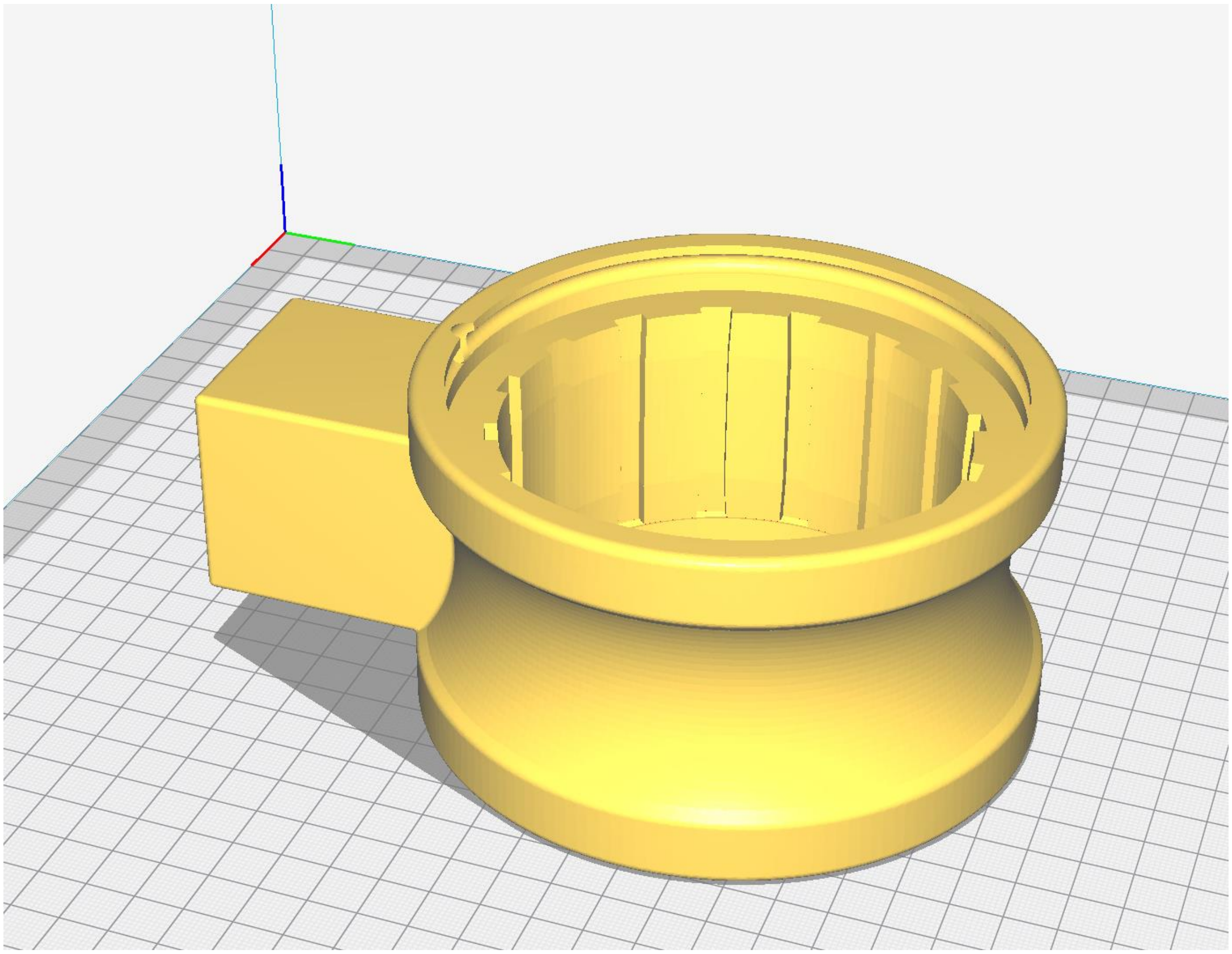

3 Components required for the sun gears.

Approximate print time: 23 hours.



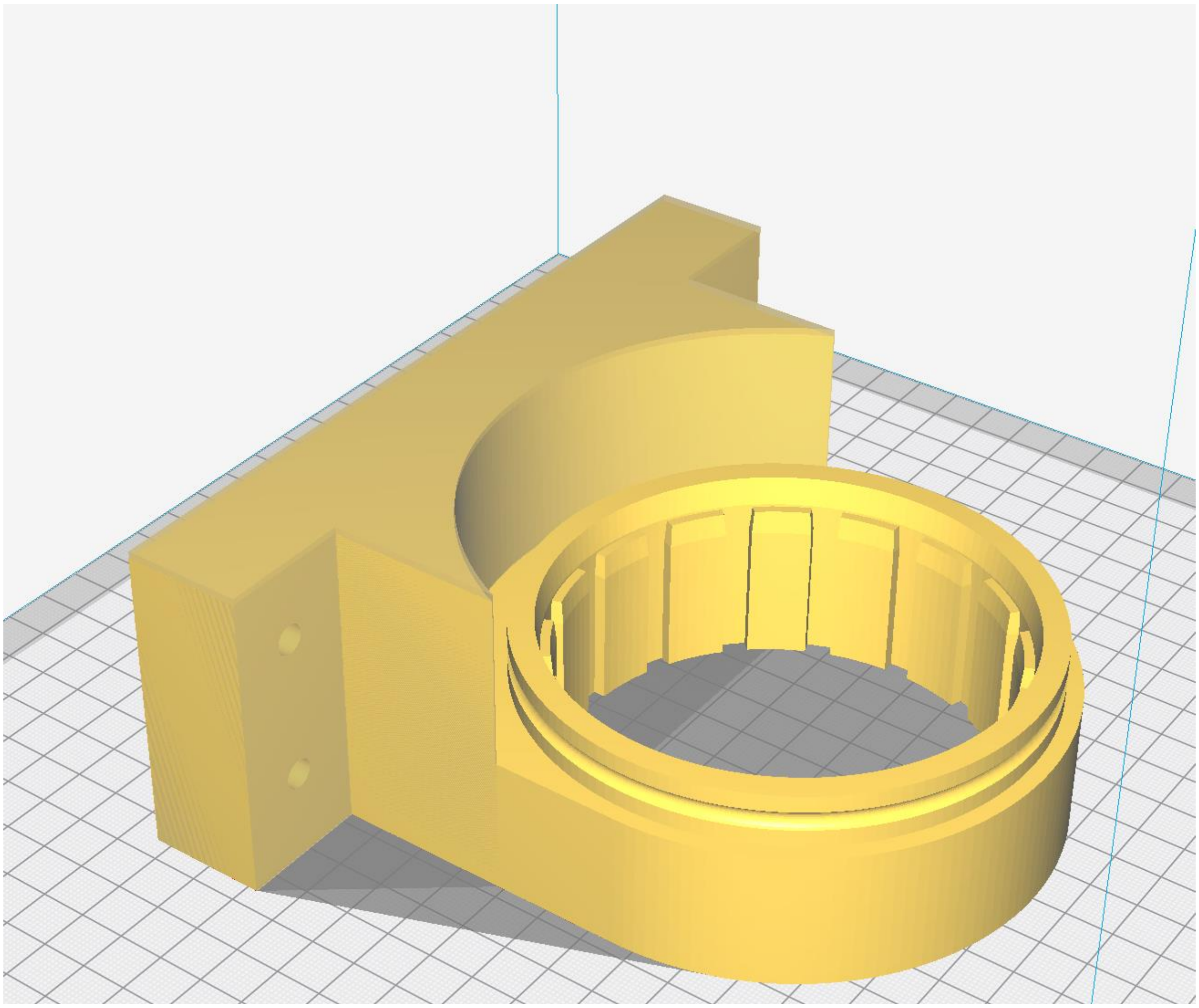
3 Ring gears (tune filament extrusion before printing as tolerances for the gaps are tight)

Approximate Print Time: 1 day, 14 hours.

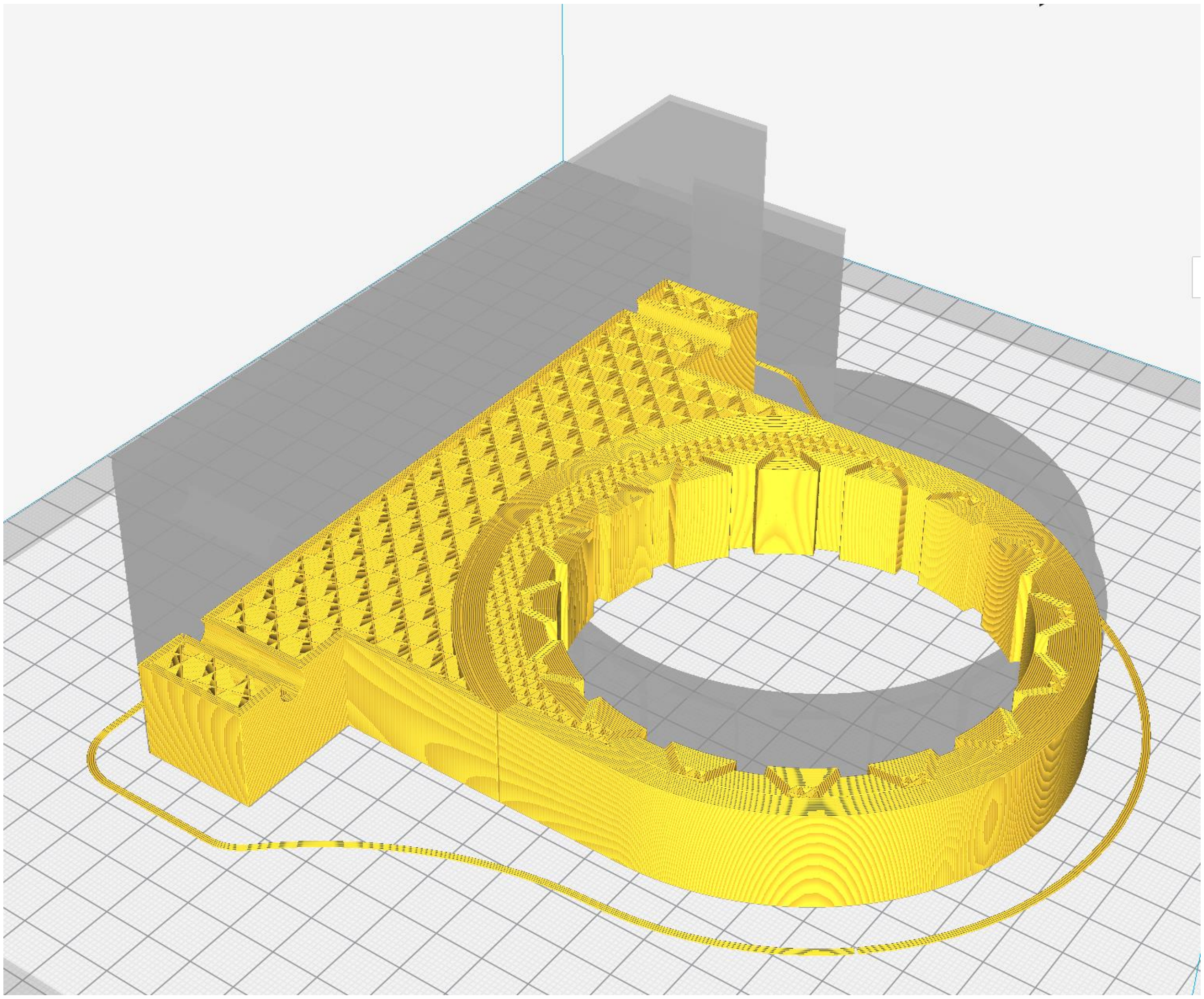


The output shaft is the only component that requires supports. There is support built into the part of the output shaft that houses the ring gear. Custom supports should be added in the slicer to support only the 90º overhangs of the antenna shaft holder.

Approximate print time: 1 day 16 hours.



The stationary ring gear holders require approximately 600 grams of filament if printed with the setting used for the other components. In order to reduce this and the print time, an STL of the bottom part of the holders (the part not in contact with the ring gears) was overlaid onto the regular model. Then, it was used to modify the print settings of only the base part of this model to reduce infill and number of walls in those locations. This can be seen in the image below.



Approximate reduced print time: 1 day, 2 hours. Amount of filament used: 390 grams.

Original print time: 2 days, 1 hour. Amount of filament used: 600 grams.

Schematics:

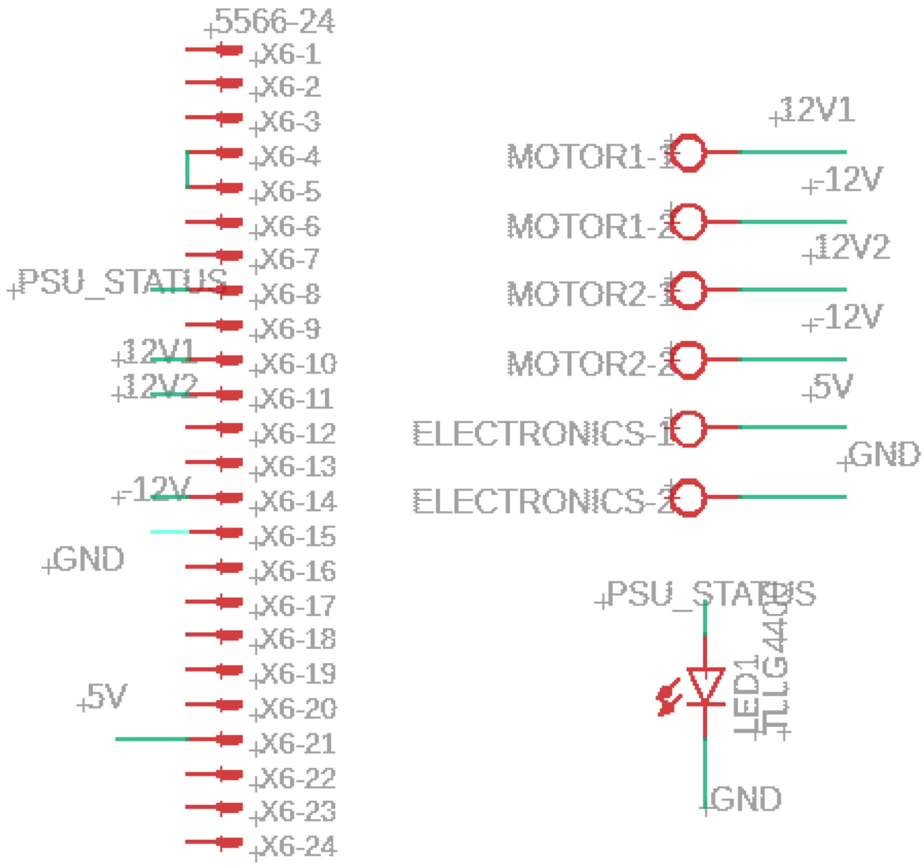


Figure 13: Schematic Diagram of Power Distribution

This schematic is for the power distribution from the power supply unit's 24 pin input to the corresponding electrical components. The motor terminal blocks take a 12V input and pass it to the motor drive through copper wires with the return line being -12V, giving an effective potential of 24V. The terminals for the electronics are simply a 5V output which will be connected to a power rail through copper wires as well.

PCB Layouts:

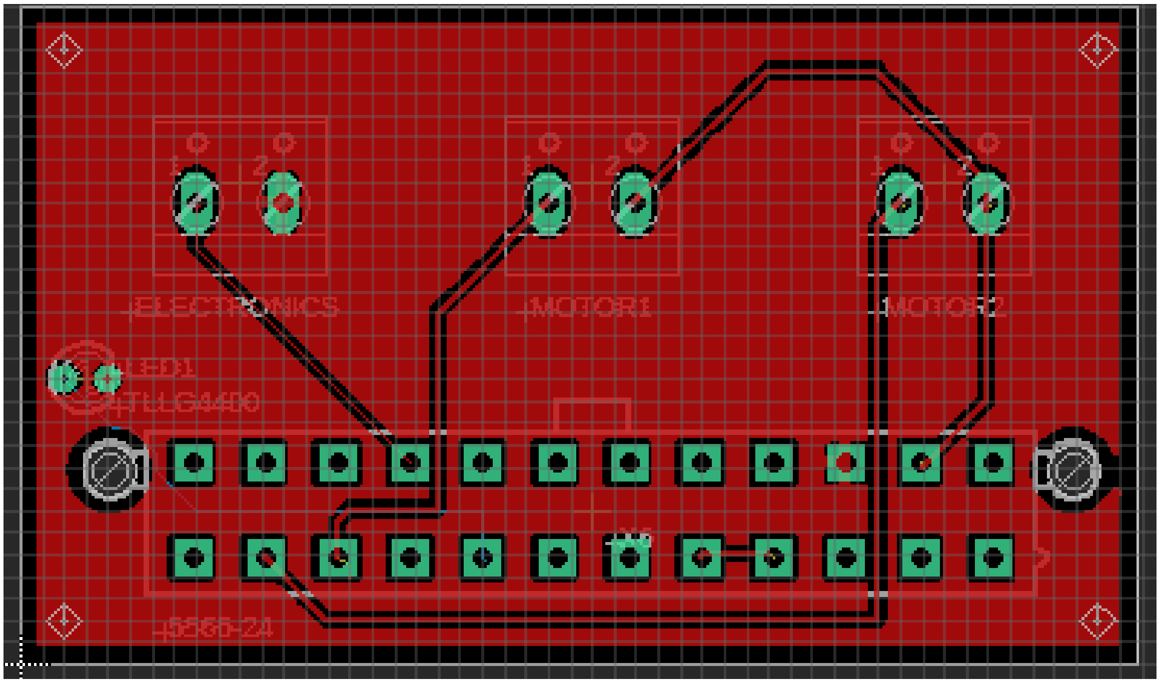


Figure 14: Power Distribution Board Layout

This PCB is designed to simply route power from the input pins of the power supply to easily accessible screw terminals that will connect to power rails and the motor drivers.

Program Flowcharts and Codes:

This segment of code is an example of the algorithm for determining the location of the satellite using two line elements, and propagating those results to estimate the next overhead pass time

```
# Orbit prediction (python)


from sgp4.earth_gravity import wgs72
from sgp4.io import twoline2rv
import datetime

import rospy
from std_msgs.msg import String
from std_msgs.msg import UInt8
```

```
line1 = ('1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11606-4 0  2927')
line2 = ('2 25544  51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537')

#this value is for ISS (Internation Space Station)
# 1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11606-4 0  2927
# 2 25544  51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537



currentDT = datetime.datetime.now()

print ("Current Year is: %d" % currentDT.year)
print ("Current Month is: %d" % currentDT.month)
print ("Current Day is: %d" % currentDT.day)
print ("Current Hour is: %d" % currentDT.hour)
print ("Current Minute is: %d" % currentDT.minute)
print ("Current Second is: %d" % currentDT.second)
print ("Current Microsecond is: %d" % currentDT.microsecond)

satellite = twoline2rv(line1, line2, wgs72)
# 12:50:19 on 4 March 2019:
position, velocity = satellite.propagate(currentDT.year, currentDT.month, currentDT.day,
currentDT.hour, currentDT.minute, currentDT.second)  # set the time you want to predict

print(satellite.error)    # nonzero
print(satellite.error_message)



print("pisition : ", position)

print("velocity : ", velocity)



def talker():
   pub = rospy.Publisher('chatter', String, queue_size=10)
   pub1 = rospy.Publisher('motor_1',UInt8, queue_size=10)
   rospy.init_node('talker', anonymous=True)
   rate = rospy.Rate(10)  # 10hz
   while not rospy.is_shutdown():

# the hello world is used to test and show the current time in RPI, to prevent error
         hello_str = "hello world %s" % rospy.get_time()
```

```
        rospy.loginfo(hello_str)
        pub.publish(hello_str)

# the motorspd, is the speed of stepper motor, or change it to angle, set for the stepper motor
        motorspd = 10
        rospy.loginfo(motorspd)
        pub1.publish(motorspd)
        rate.sleep()


if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

This program is the test code in Arduino for utilising the GPS module through Serial communication

```
#GPS module control code (Arduino)



# for testing

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

/*

  It requires the use of SoftwareSerial, and assumes that

  9600-baud serial GPS device hooked up on pins 4(rx) and 3(tx).

*/

static const int RXPin = 4, TXPin = 3;

static const uint32_t GPSBaud = 9600;
```

```
//set the Serial communication frequency


// The TinyGPS++ object

TinyGPSPlus gps;


// The serial connection to the GPS device

SoftwareSerial ss(RXPin, TXPin);


void setup()

{

  Serial.begin(115200);

  ss.begin(GPSBaud);


  Serial.println(F("DeviceExample.ino"));

  Serial.println(F("A simple demonstration of TinyGPS++ with an attached GPS module"));

  Serial.print(F("Testing TinyGPS++ library v. "));
Serial.println(TinyGPSPlus::libraryVersion());

  Serial.println(F("by Mikal Hart"));

  Serial.println();

}


void loop()

{

  // This sketch displays information every time a new sentence is correctly encoded.
```

```
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displayInfo();

  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
  }
}

void displayInfo()
{
  Serial.print(F("Location: "));
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
  }
  else
  {
    Serial.print(F("INVALID"));
  }
```

```
Serial.print(F("  Date/Time: "));
if (gps.date.isValid())
{
  Serial.print(gps.date.month());
  Serial.print(F("/"));
  Serial.print(gps.date.day());
  Serial.print(F("/"));
  Serial.print(gps.date.year());
}
else
{
  Serial.print(F("INVALID"));
}

Serial.print(F(" "));
if (gps.time.isValid())
{
  if (gps.time.hour() < 10) Serial.print(F("0"));
  Serial.print(gps.time.hour());
  Serial.print(F(":"));
  if (gps.time.minute() < 10) Serial.print(F("0"));
  Serial.print(gps.time.minute());
  Serial.print(F(":"));
```

```
    if (gps.time.second() < 10) Serial.print(F("0"));

    Serial.print(gps.time.second());

    Serial.print(F("."));

    if (gps.time.centisecond() < 10) Serial.print(F("0"));

    Serial.print(gps.time.centisecond());

  }

  else

  {

    Serial.print(F("INVALID"));

  }



  Serial.println();

}
```

Helper code to convert the received GPS data and converting it into implementable longitude and latitude data

```
# Coordinate conversion



# from geometric to ECEF X (python)



def geodetic2ecef(lat: float, lon: float, alt: float,
              ell: Ellipsoid = None, deg: bool = True) -> Tuple[float, float, float]:


#    ECEF (Earth centered, Earth fixed)  x,y,z


    if ell is None:
```

```
        ell = Ellipsoid()

    if deg:
        lat = radians(lat)
        lon = radians(lon)

    with np.errstate(invalid='ignore'):
        # need np.any() to handle scalar and array cases
        if np.any((lat < -pi / 2) | (lat > pi / 2)):
            raise ValueError('-90 <= lat <= 90')

        if np.any((lon < -pi) | (lon > tau)):
            raise ValueError('-180 <= lat <= 360')

    # radius of curvature of the prime vertical section
    N = get_radius_normal(lat, ell)
    # Compute cartesian (geocentric) coordinates given  (curvilinear) geodetic
    # coordinates.
    x = (N + alt) * cos(lat) * cos(lon)
    y = (N + alt) * cos(lat) * sin(lon)
    z = (N * (ell.b / ell.a)**2 + alt) * sin(lat)

    return x, y, z

def get_radius_normal(lat_radians: float, ell: Ellipsoid = None) -> float:

    radius : float
        normal radius (meters)
    if ell is None:
        ell = Ellipsoid()

    a = ell.a
    b = ell.b

    return a**2 / sqrt(a**2 * cos(lat_radians)**2 + b**2 * sin(lat_radians)**2)



# From ECEF frame to ENU (East, North, Up) frame

def ecef2enu(x: float, y: float, z: float,
             lat0: float, lon0: float, h0: float,
             ell: Ellipsoid = None, deg: bool = True) -> Tuple[float, float, float]:

    x0, y0, z0 = geodetic2ecef(lat0, lon0, h0, ell, deg=deg)
```

```python
    return uvw2enu(x - x0, y - y0, z - z0, lat0, lon0, deg=deg)


def uvw2enu(u: float, v: float, w: float,
            lat0: float, lon0: float, deg: bool = True) -> Tuple[float, float, float]:

    if deg:
        lat0 = radians(lat0)
        lon0 = radians(lon0)

    t = cos(lon0) * u + sin(lon0) * v
    East = -sin(lon0) * u + cos(lon0) * v
    Up = cos(lat0) * t + sin(lat0) * w
    North = -sin(lat0) * t + cos(lat0) * w

    return East, North, Up
```

Function for determining the motor angles to achieve accurate pointing to the satellite using location of the satellite and the position of the ground station as inputs.

```python
# Satellite antenna alignment (python)

#calculate the Azimuth and elevation angle of antenna pointing angle


def enu2aer(e: np.ndarray, n: np.ndarray, u: np.ndarray, deg: bool = True) -> Tuple[float, float, float]:
  #e n u unit meters

    e = np.atleast_1d(e)
    n = np.atleast_1d(n)
    u = np.atleast_1d(u)

    with np.errstate(invalid='ignore'):
        e[abs(e) < 1e-3] = 0.
        n[abs(n) < 1e-3] = 0.
        u[abs(u) < 1e-3] = 0.

    r = hypot(e, n)
    slantRange = hypot(r, u)
    elev = arctan2(u, r)
    az = arctan2(e, n) % tau

    if deg:
        az = degrees(az)
```

```
        elev = degrees(elev)

    return az[()].squeeze(), elev[()].squeeze(), slantRange[()].squeeze()



#geodetic coordinate given from GPS, and perturbation module gives the ECEF frame
[x,y,z]




import rospy
from std_msgs.msg import String
from std_msgs.msg import UInt8
import mypymap3d as pm
from datetime import datetime

currentDT = datetime.datetime.utcnow()

def callback0(data):
    rospy.loginfo(rospy.get_caller_id() + 'I heard %s', data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber('position', String, callback0)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()



now = datetime.utcnow()


#x,y,z = pm.geodetic2ecef(42.961234288436515,-81.2613305416536,248)
#pm.ecef2enu()

#pm.geodetic2ecef(42.961234288436515,-81.2613305416536,248)

#pm.enu2aer()
print(pm.eci2geodetic((-285741.5181748932, -6492280.739821421, 1844943.6773562581),now))
# returns ECEF (Earth centered, Earth fixed) x,y,z in meter
```

```python
print(pm.ecef2geodetic(-285741.5181748932, -6492280.739821421, 1844943.6773562581))
print(pm.geodetic2ecef(20,-120,831000))


print("pisition : ", position)
#pm.eci2geodetic
#pm.ecef2geodetic
#pm.geodetic2ecef
A=pm.eci2geodetic(position,currentDT)
print(A)
#az,el,range = pm.geodetic2aer(lat, lon, alt, observer_lat, observer_lon, 0)

##this is the actual returned Azimuth and elevation angle
if __name__ == '__main__':
    listener()
```

Simulation testing for the motor to ensure that they work properly

```
#motor control for simulation test



#include <Stepper.h>

#include <ros.h>

#include <std_msgs/UInt8.h>




ros::NodeHandle  nh; //Next, we need to instantiate the node handle, which allows our

//program to create publishers and subscribers.

//The node handle also takes care of serial port communications.


const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution

// for your motor
```

```
int motorSpeed;
// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);



void stepper_cd( const std_msgs::UInt8& cmd_msg){
  motorSpeed=cmd_msg.data; //set speed of stepper
//  serial.print("the speed value is ", motorSpeed)


}


ros::Subscriber<std_msgs::UInt8> sub("motor_1", stepper_cd);


void setup() {
  // nothing to do inside the setup
  nh.initNode();
  nh.subscribe(sub);
}


void loop() {
  // read the sensor value:
  //int sensorReading = analogRead(A0);
  // map it to a range from 0 to 100:
 // int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
```

```
  // set the motor speed:
  if (motorSpeed > 20) {


     //unit RPM
    // step 1/100 of a revolution:
     myStepper.setSpeed(30);
     myStepper.step(stepsPerRevolution / 100);


 } else
 {
  myStepper.setSpeed(10);
     myStepper.step(stepsPerRevolution / 100);
 }
   nh.spinOnce();//all ros callback
     delay (1);
}
```

Specific helper code for interfacing the NEMA 17 Motors with the Arduino slave board

```
# Code for NEMA 17 stepper motor



#include <Stepper.h>
```

```
const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(60);
  // initialize the serial port:
  Serial.begin(9600);
}

void loop() {
  // step one revolution  in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);

  // step one revolution in the other direction:
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

User Interface

The user interface will enable the user to have simple control and be able to set tasks for the ground station. A convenient method to get remote access to the RPI is through Secure Shell (SSH). With this method, users can access to the Linux control tools including the Raspbian desktop system and Linux console. There are two choices of the telnet clients, PuTTY and REALVNC, and the SSH connection can be through Ethernet or mobile Wi-Fi. The RPI are set as fix IP address 192.168.137.214 for Ethernet connection, and 192.168.137.245 for the WIFI connection. The user interface is illustrated in Figure 15 below.

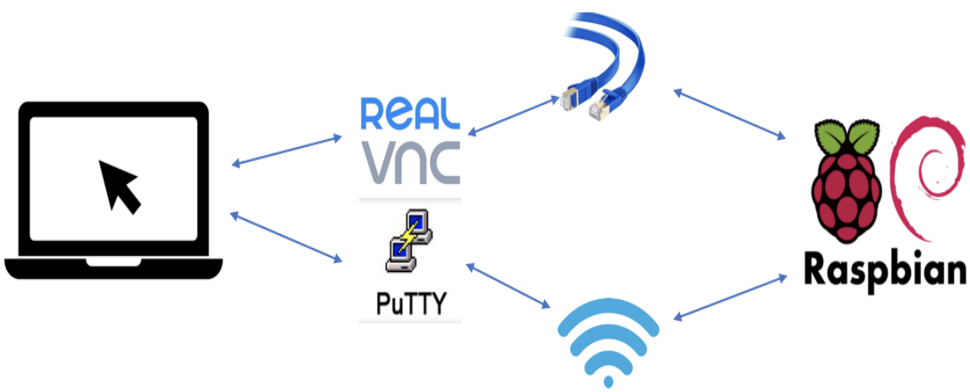


Figure 15: User interface diagram

Data Flow

The primary design for data flow is based on ROS. There are six nodes created for six different tasks, which can be seen in Figure 16 on the following page.

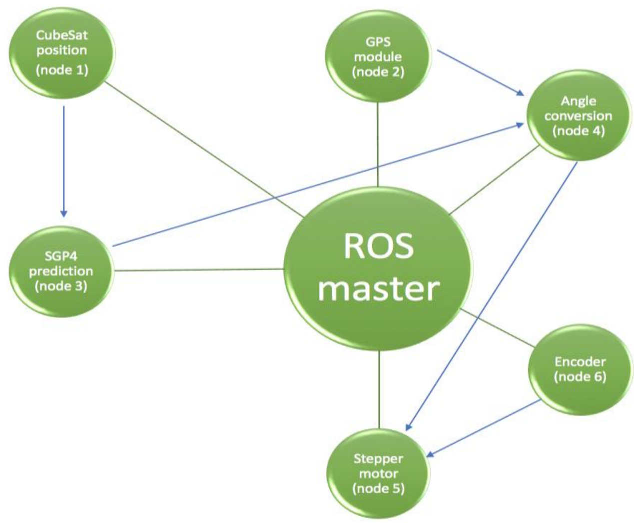


Figure 16: Data flow diagram

The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer [12]. The GPS module (node 2) publishes the position of the ground station and sends this message to Angle conversion (node 4) as a subscriber. CubeSat position (node 1) sends the CubeSat position and speed to the SGP4 prediction model (node 3). After calculations are complete, node 3 publishes the next overhead position to node 4, which will convert the position of CubeSat to the earth-based coordinate system and publish the rotation angles for each of the stepper motors. The stepper motors (node 5) will rotate to their desired angles. The encoder (node 6) publishes the angle value to node 5, to correct the error.

Bill of Materials:

The materials for the ground station were sourced from the electronics shop and University Machine Services. The following three tables show a detailed cost breakdown.

Table 15: Costs of materials from electronics shop

Item	Quantity	Cost/Unit	Shipping Cost	Total Cost
Rotary Encoder	2	$76.19	$0.00	$152.38
GPS Module	1	$20.99	$9.81	$30.80
Raspberry Pi 3B+	1	$94.95	$13.95	$108.90
Nema 17 Motors	3	$7.33	$24.64	$46.63

Stepper Motor Drivers	2	$15.56	$0.00	$31.12
EVGA Power Supply	1	$79.99	$0.00	$79.99
			Subtotal	$449.82
			Tax	13%
			Final Total	$508.30

Table 16: Costs of materials from University Machine Services

Item	**Quantity**	**Cost per Unit**	**Total Cost**
2" x 8" x 12' Wood	1	Exact breakdown unknown	
2" x 4" x 8' Wood	4		
4' x 8' Wood Weather Treated	2		
UMS Wood Materials Cost			$119.39
3" x 3" x 4' (1/4" thick) Square Metal Tube	1	Exact breakdown unknown	
1" Steel Angle, 1/8" thick x 60"	1		
1" diameter, 1" long Metal Rod	1		
1/2" diameter, 4" length Shaft	1		
1" x 1" x 1", 12" long Z Bracket	1		
6" x 4", 1/4" Thick Steel Plate	1		
Low-Carbon Steel Disc 1/2" Long, 5" Diameter	2		
4.5" long, 2.5" diameter Steel Rod	1		
10" x 10" x 1/2" Steel Plate	1		
1/2" long, 3" Diameter Low-Carbon Steel Disc	1		

UMS Metals Estimate			$87.86
Thrust Ball Bearing	1	$22.25	$22.25
Ball Bearings	2	$11.12	$22.25
		Final Total (incl. Tax)	$251.75

Table 17: Costs of materials from other locations and services

Item	Quantity	Cost/Unit	Shipping Cost	Total Cost
3D Printing Filament	2	$30	0	$60
Nuts (assorted)	1	$40	0	$40
Bolts (assorted)	1	$50	0	$50
Dowels	1	$2	0	$2
Linear rods	3	$12	0	$36
Wood Glue	1	$20	0	$20
Welding (UMS)	1 hour	$60/hr	N/A	$60
			Subtotal	$268
			Tax	13%
			Final Total	302.84

Table 18: Labor Costs

Type	Number of Hours	Cost Per Hour	Total Cost
Engineering Time	400	$25	$10,000
Machinist Time	20	$40	$800
			$10,800

Section 3: Prototype Feasibility

For the prototype of the ground station, a functional tracking system will be fabricated and built to demonstrate the intended use of the system. Given that certain key elements that would be incorporated into the ground station (such as antennae, data relay systems and user interfaces), are at the discretion of other subsystems in the CubeSat, the ground station will replicate the desired tracking motion as if it were following a celestial body. The prototype will feature the mechanical design of the ground station, the actuation systems and their implementation, the control system and how all the components are integrated as well as more subtle details such as power distribution and weatherproofing. A live demo of the tracking capability of the ground station will accomplished by walking a balloon in front of the ground station to simulate a moving celestial body. Using digital imaging processing, the ground station will predict the motion of the balloon and accurately align its camera with the balloon at the center. This will be done in lieu of an actual orbiting satellite as it is much harder to prove that the ground station is tracking an un-seeable satellite than a balloon when there is no data being received from the satellite.

The following steps were followed/will be followed in order to complete the fabrication of the ground station.

Base Subassembly:

- Cut the lumber down to the required sizes using a table saw.
- Drill pilot holes in the 2” x 8” Base Center and Top Center. Then, with the use of a jigsaw, cut the out the square profile of the 3 inch x 3 inch square tube.
- Join the following pieces by first drilling pocket holes. The apply wood glue liberally on the surfaces that need to be joined. Finally, screw the wood screws into the pocket holes.
    - 2” x 4” for base x 2.
    - 2” x 4” for base short x 2.
    - 2” x 8” base side x 2.
    - 2” x 8” Base Center x 1.
        - Note: The 2” x 4” pieces only require 2 pocket holes while the 2” x 8” connection require 4 pocket holes.

- Ensure that the 2” x 4” support pieces line up with the 2” x 8” top piece before proceeding with the next step. Also ensure that the square holes are lined up with each other and that the square tube can fit inside these holes.

- Connect the 2” x 4” support pieces to the base with the use of wood screws and wood glue.

- Connect the 2” x 8” Top center to the 2” x 4” support pieces with the use of wood screws and wood glue.

- Make 12 angle brackets by cutting an angle bracket to size and drilling the holes.

- Drill the holes in the 3”x3” center tube with the use of a milling machine or a drill press and use a tap to thread the required holes.

- Attach the square tube to the wood base with the use of the angle brackets. Use nuts to attach the angle brackets to the square tube and self-drilling screws to attach the brackets to the wood.

Azimuth Subassembly:

- Machining the parts:

    - Bearing Holder: Machine the bearing holder on the lathe to achieve the right geometry with the right tolerances. Use a drill press or a milling machine to cut the holes on the side.

    - Shaft: Cut a 0.5” diameter round stock to size and machine the grooves and create the chamfer on the top.

    - Motor Coupler: Machine the holes for the motor shaft and shaft using a lathe. Use a milling machine or drill press to drill the holes on the side of the part and use a tap to thread these holes.

    - Base: Use a milling machine to drill the holes in the base plate and create the blind hole on the bottom of the plate. The milling machine is required for this step as misalignment of the holes could lead to a misalignment between the elevation motor and the gearbox.

    - Motor Holder Flange, Motor Holder, Bearing Holder Flange Plates: Use the lathe to machine round stock to the required size and geometry. Use a milling machine to drill holes the through holes in the parts for proper alignment.

    - Motor Holder Flange Plates: Use a drill press or a milling machine to drill the holes.

- Assembly:
    - Weld the motor holder flange plates to the motor holder flange, the bearing holder to the bearing holder flange plate, and the shaft to the base plate.
    - Attach the ball bearings to the shaft with the use of the retaining rings.
    - Put the thrust bearing onto the bearing holder. Press fit the ball bearings into the bearing holder.
    - Attach the motor coupler to the shaft with the use of grub screws.
    - Attach the motor holder flange to the square tube with the use of nuts and bolts.
    - Screw motor into the motor holder and screw the motor holder into the motor holder flange.
    - Nut and bolt the motor holder flange to the bearing holder flange.

Elevation subassembly

3D print the components in the orientations mentioned above with the following settings:

- 5 mm walls with 50% infill and 0.2 mm layer height with a 0.4 mm nozzle.
- Keep the extruder as hot as possible to maximize layer adhesion without drooping.
- Print at 75% of regular speed. This ensures the least amount of print artifacts on the gears and allows proper mating of the gear teeth.

- Press 8mm diameter linear rods into the planetary gears.
- Assemble the gearbox by aligning the planetary gears with the lines with the lines on the sun gears and zip tying them together. Then, starting with the first ring gear, flex the ring gear outward and align the lines again and release over the planetary gears. Lubricate the gears with 3 in 1 oil.
- Slide the output ring gear holder onto the middle sun gear. Then, slide the two stationary ring gear holders onto either side of the gear box. Fill the hole in between the output ring gear holder and the stationary ring gear holders with 4.5 mm plastic bb's used in bb guns.

- Cut 1 inch angle brackets to size and drill the required holes in them.
- Using a milling machine for precision, drill the holes required to make the elevation motor plate. Attach the motor to this.
- Attach the motor plate to the base with the use of dowels to locate the plate and nuts to hold it into place.
- Secure the gearbox to the base with the use of bolts, washers, and nuts, using the angle bracket to disperse the force evenly.

Not including the costs of labor, the final cost of all materials and services was $1062.89. The budget provided by our supervisors was $1,000, along with another $300 from our capstone budget. Therefore, we are well within our budget. The only expense not incurred due to the purchase of materials was the welding done by the UMS. Although the welds were not complex, there was no one in the group with the required welding skill in order to perform these welds. Therefore, the welding work had to be outsourced to the UMS. The task assigned to us by our supervisors was to design and build the actuation system for a ground station within the budget provided to us. The costs of the entire project is just within the budget supplied to us by our supervisor in conjunction with the capstone budget.

The fabricated prototype will be a valid representation of the final product because it will be able to validate many of the key design choices and calculations that were made to ensure proper function as the scale will be 1:1. The base for instance, will be made to the same size as specified in the earlier sections, and thus will be able to validate that the system will not tip when high lateral force is applied to the top of the mast. Adding a weight to the top of the mast to simulate an antenna can also be done to show that the minimum speed/torque requirements were met, ensuring that the ground station functions well under a load. Finally, since all the hardware and software will be completed, the functional demonstration will show the operation of the program and data handling, which will verify that the choices to use the particular microcontroller, power supply, systems architecture, etc, were well thought out and selected.

References

[1] "AISI 4140 Alloy Steel (UNS G41400)", *AZoM.com*, 2019. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=6769. [Accessed: 12- Mar- 2019].

[2] "The Carbon Steel Advantage - FedSteel.com", *FedSteel.com*, 2019. [Online]. Available: https://www.fedsteel.com/our-blog/carbon-steel-advantage/. [Accessed: 12- Mar- 2019].

[3] Cakaj, S., Kamo, B., Lala, A., & Rakipi, A. (2014). The Coverage Analysis for Low Earth Orbiting Satellites at Low Elevation. (IJACSA) International Journal of Advanced Computer Science and Applications, 5(6). doi:http://thesai.org/Downloads/Volume5No6/Paper_2-The_Coverage_Analysis_for_Low_Earth_Orbiting_Satellites_at_Low_Elevation.pdf /. [Accessed: 12- Mar- 2019].

[4] *Flagpole.ca*, 2019. [Online]. Available: http://www.flagpole.ca/windspeeds/TorontoMaximumWindSpeed5Yrs.pdf. [Accessed: 15- Mar- 2019].

[5] The Drag Equation. (n.d.). Retrieved from https://www.grc.nasa.gov/www/k12/rocket/drageq.html /. [Accessed: 12- Mar- 2019].

[6] 17HS19 Motor Torque Curve. (n.d.). Retrieved from https://www.omcstepperonline.com/download/17HS19-2004S1_Torque_Curve.pdf /. [Accessed: 12- Mar- 2019].

[7] R. Dix, “Material Considerations in Fused Filament Fabrication,” 2018.

[8] TS Kelso, "Models for Propagation of NORAD Element Sets", 1980. Available: http://www.celestrak.com/NORAD/documentation/spacetrk.pdf

[9] "Thermal Design Guide for DSP and ARM Application Processors", *Texas Instruments*, 2019. [Online]. Available: http://www.ti.com/lit/an/sprabi3b/sprabi3b.pdf. [Accessed: 13- Mar- 2019].

[10] “Thermal management guidelines for STM32 32-bit ARM Cortex MCUs applications", *Comm.eefocus.com*, 2019. [Online]. Available: http://comm.eefocus.com/media/download/index/id-1015333. [Accessed: 12- Mar- 2019].

[11] "AMCI : Advanced Micro Controls Inc :: Stepper vs Servo", *Amci.com*, 2019. [Online]. Available: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo. [Accessed: 10- Mar- 2019].

[12] ROS Documentation. (n.d.). Retrieved from http://wiki.ros.org/Master

[13] "How GPS Works", *Maptoaster.com*, 2019. [Online]. Available: https://www.maptoaster.com/maptoaster-topo-nz/articles/how-gps-works/how-gps-works.html. [Accessed: 14- Mar- 2019].

Appendix 1: Material Property Calculations for FEA on 3D printed objects

K_{FFF} is used to reduce the uniform isotropic material property to the value that 3D printing produces. These values are listed for different properties in Table 1 and were found via testing. A_w/A_x is the ratio of wall cross sectional area to total cross sectional area. A_i/A_x is the ratio of infill cross sectional area to total cross sectional area. $K_{i,\%}$ is a function of the 3D print's infill percentage, and $K_{i,t}$ is a function of the 3D print's infill shape. These values were also determined via testing and can be found in Table 2 and Table 3.

Table 19: K_{FFF} Values

	Ultimate tensile strength S_{ut}	Yield Strength S_y	Young's Modulus, E
K_{FFF}	0.8330	0.8274	0.5451

Table 20: $K_{\%}$ Values

Infill %	Ultimate tensile strength, S_{ut}	Yield Strength, S_y	Young's Modulus, E
10	0.1298	0.1419	0.0871
20	0.1348	0.1517	0.0990
30	0.1817	02067	0.1434
40	0.2060	0.2269	0.1603
50	0.2450	0.2778	0.2058
60	0.2722	0.3224	0.2364
70	0.3356	0.3224	0.2878
80	0.4117	0.4822	0.3539
90	0.5746	0.6821	0.5143
100	0.7307	0.8658	0.5408

Table 21: K_{Infill} Values

	Ultimate tensile strength, S_{ut}	**Yield Strength, S_y**	**Young's Modulus, E**
Grid	1	1	1
Triangle	1.348	1.468	2.258
Cubic	1.904	1.844	1.867

Table 4 shows the values used to 3D print the two components most likely to fail: the output ring gear and the input connector.

Table 22: Print settings

Setting	Ring Gear	Input Connector
Wall Thickness	5 mm	5 mm
Infill Percentage	50%	50%
Infill Type	Cubic	Cubic

Using these values, the values required for the formula were calculated. As the parts had a wall thickness of 5 mm and were not thicker than 10 mm at any given place, there was no infill in either part. Therefore, $A_i/A_x = 0$ and $A_w/A_x = 1$. Using these values, and the values from Table 1, more accurate part properties can be calculated as seen in Table 4. As the physical properties used in the formula of the two parts are the same, the material properties were only calculated once.

Table 23: Material properties

	Ultimate tensile strength, S_{ut}	**Yield Strength, S_y**	**Young's Modulus, E**
$\frac{A_w}{A_x} + \frac{A_i}{A_x} * K_{i,\%} * K_{i,t}$	1	1	1
K_{FFF}	0.8330	0.8274	0.5451
Rated Material	59 MPa	70 MPa	3500 MPa

Properties [13]			
Adjusted Material Properties	49.147 MPa	57.918 MPa	1907.85 MPa