

Avocado

Final Report



Winter/Spring 2018

Northwestern University: Wildcat Robot Design Studio

Akash Borde, Tsong Chen, Rachel Hughes, Weilin Ma, Ryan Miller, Nathan Shelly, Allen Tang

Table of Contents

[The Challenge 2](#_Toc516522269)

[The Team 3](#_Toc516522270)

[The System 3](#_Toc516522271)

[The Analysis 4](#_Toc516522272)

[Mechanical 4](#_Toc516522273)

[Electrical 14](#_Toc516522274)

[Software 15](#_Toc516522275)

[The Design Evolution 17](#_Toc516522276)

[The Final Design 28](#_Toc516522277)

[The Testing 31](#_Toc516522278)

[The Performance 31](#_Toc516522279)

[The Future 34](#_Toc516522280)

[The Others 35](#_Toc516522281)

# The Challenge

#### Sponsor

Our team sponsor was Bill Hunt, a technician/research engineer in the Neuroscience and Robotics Laboratory at Northwestern University. As Bill’s research often involves investigating how robotic linkages behave and interact, he wanted an accessible, general-purpose actuator. Specifically, Bill wanted an inexpensive actuator with high power density while that was as close to direct-drive as possible.

#### Project Definition

Along with Bill, the Smart Actuator Team arrived at the following requirements for our actuator:

###### Cost:

BOM ≤ $500

###### Torque/Power:

2N-m at stall

1N-m for 1 minute

30W core power

Speed - according to motor kt

###### User Interface/Controls:

Capable of position, velocity, & current control

RS232 communication protocol

Able to connect and control several (7)

###### Electromechanical:

~24 V input

≤ 0.5° backlash

Built-in heat sinking

Through-hole for cabling

###### Software:

Velocity, current, and position are settable and gettable

Provide control API through a C wrapper around RS232 communication

# The Team

The roles and responsibilities were divided amongst the team members as follows:

###### Team Leaders

Akash Borde, Rachel Hughes

###### Mechanical Design and Manufacturing

Rachel Hughes, Weilin Ma

###### PCB Design and Assembly

Akash Borde, Allen Tang

###### Motor Control

Akash Borde, Tsong Chen, Allen Tang

###### API Development

Ryan Miller, Nathan Shelly

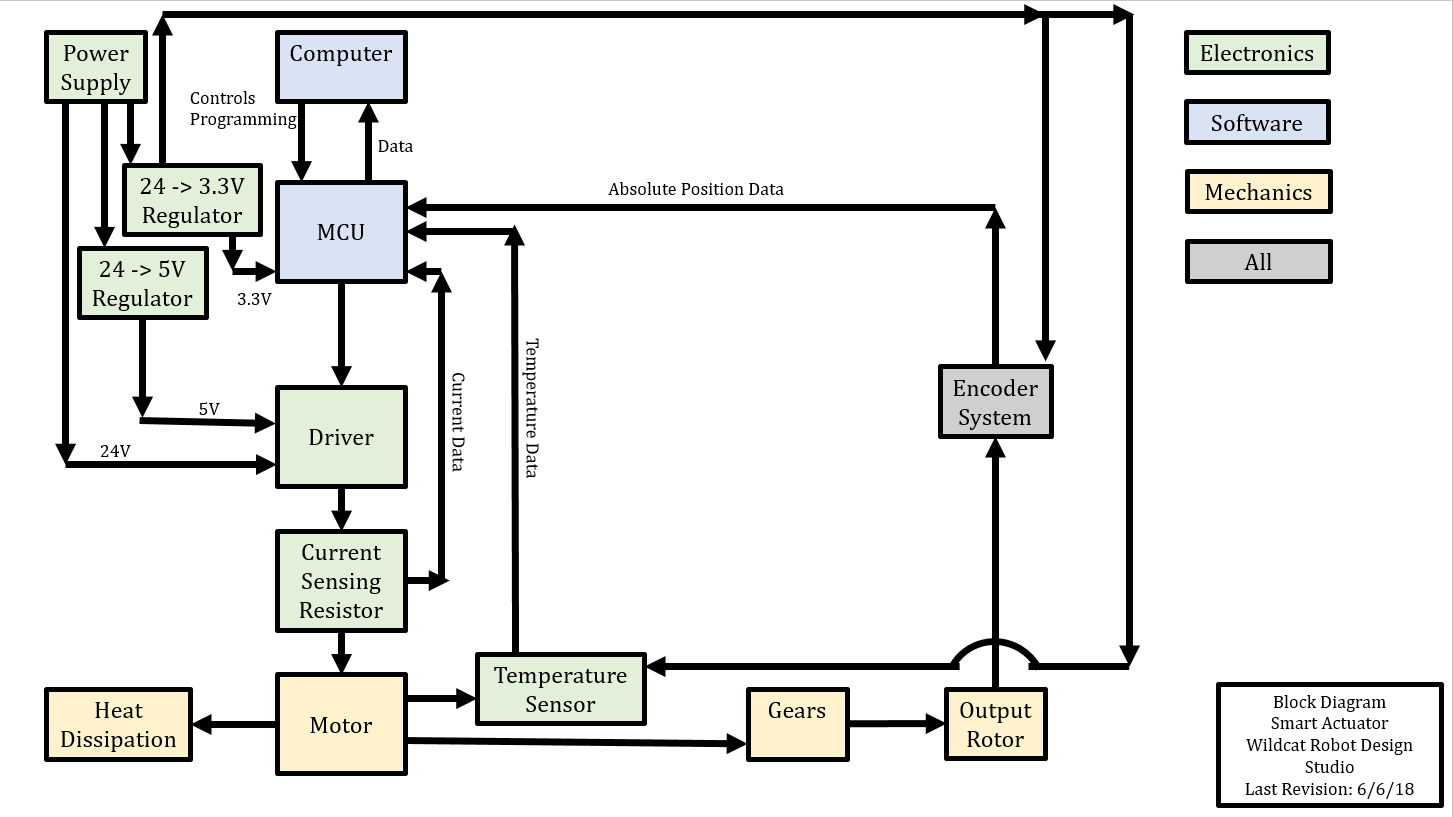
###### Communications

Ryan Miller, Nathan Shelly, Allen Tang

# The System

The system follows a flow shown in Figure 1, below.

Figure 1: Block diagram of power and data flow through the system



Using a 24V power supply and regulators for 3.3 and 5V, the entire Avocado is run. A current sensing resistor takes current data, a thermocouple takes temperature data, and an encoder system takes position data. All of the data is sent to its respective control loop in the MCU, which communicates with the “computer,” either another microcontroller or some other interface the user has access to. The position data comes from the output rotor, which rotates as a result of one-stage gearing from the motor.

# The Analysis

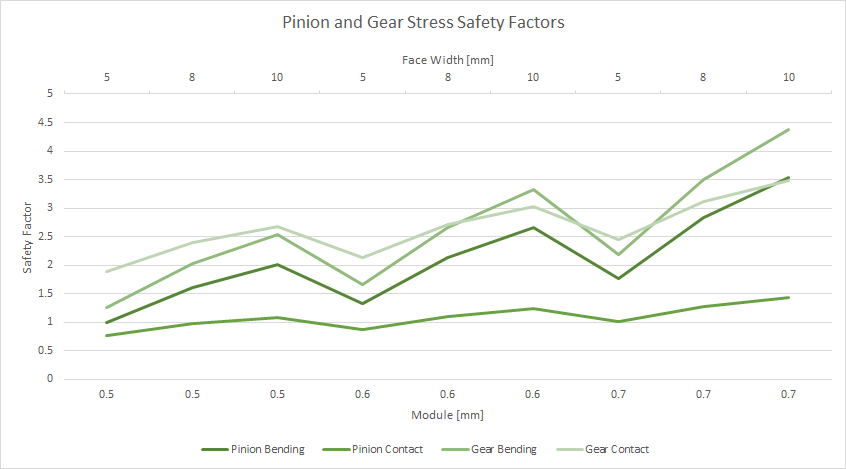
### Mechanical

#### Gearing

To determine the necessary materials and geometries for the gears, we created a MATLAB script (gears.m, found [here](gears.m)) that computes the safety factors on the gears for contact and bending stresses. The algorithm uses material properties, gear geometries, and loading conditions to calculate the stresses through a series of knockdown factors, thus making it a very conservative model. Although the model is typically for steel gears, approximations for plastic gears work as well.

The gear ratio was fixed by the requirements as 8:1, implying that had to be the ratio of the teeth gear to pinion. The numbers of teeth, pressure angle, diameters, and materials were all manipulated until the best combination for our requirements was selected. The parameters were changed and the analysis iterated based on feasibility of manufacturing, safety factors under load, and desired compact size. An example of several trials of gear parameters is shown below in Figure 2, where the material properties were held constant and the face width and module were manipulated.

Figure 2: Gear Safety Factor iterations



Due to the unusual size requirement of the output gear and the necessity of a 15mm output bore hole, we decided to print one, which would result in an ABS plastic material. Analysis of this material with a commercial steel gear used the following properties:

Teeth on pinion: 12 Teeth on gear: 96

Face width (gear and pinion): 7mm Module: 32 pitch

Pressure angle: 14.5 degrees Pinion Brinnell hardness: 200

Gear Brinnell hardness: 170 Quality factor (gear and pinion): 6

Pinion Elastic Modulus: 200 x 109 Gear Elastic Modulus: 15 x 109

Pinion Poisson’s Ratio: 0.30 Gear Poisson’s Ratio: 0.35

Using these properties for analysis, the resulting safety factors were as follows:

**Pinion Safety Factors:**

Bending: 1.824244

Contact: 1.935544

**Gear Safety Factors:**

Bending: 1.514236

Contact: 3.864253

Convention gives the ideal safety factor in the range of 2-3 and no less than 1.1, which puts our conservative model close enough to the desired range for our satisfaction.

Additionally, to confirm the accuracy of our model, we used the same parameters in the Lewis Factor model. The Lewis factor model accepts the face width, tensile strength, diametral pitch, and the Lewis Factor itself derived from the pressure angle and number of teeth. The results of this model were in the 3-4 range for all safety factors, agreeing with the previous results and verifying that our chosen gear parameters were satisfactory.

#### Bearing Selection

The bearings were chosen after an extensive analysis of potential loading on the bearings. Several different cases were tried, with the shaft bearing load as well as not, and the shaft being hollow or solid. The bearing choice began with calculations that not only arrived at the required bearing loads, but led to the decision to have a non-load-bearing shaft on the output. The axial loading on the bearing was given by the gears, although the thrust load had to be approximated. We assumed the maximum thrust load case to be with an arm attached to the Avocado, extending out a length of about 310mm (1 foot) and having an evenly distributed weight of 5kg. The following calculations, in Figures 3-5, show the mathematical procedure behind the necessary loading conditions.

Figure 3: Shaft Analysis for thrust load

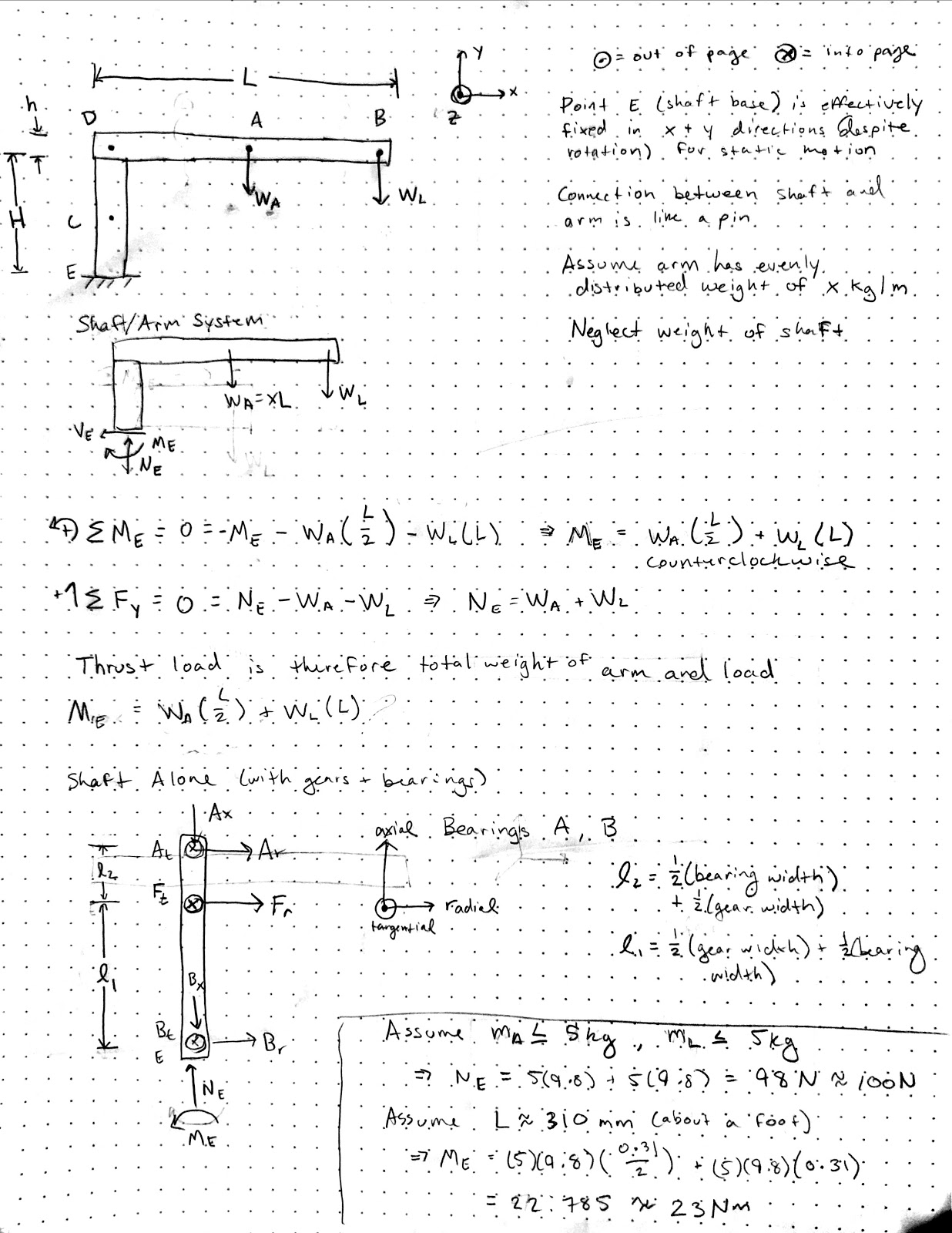


Figure 4: Force analysis for thrust load

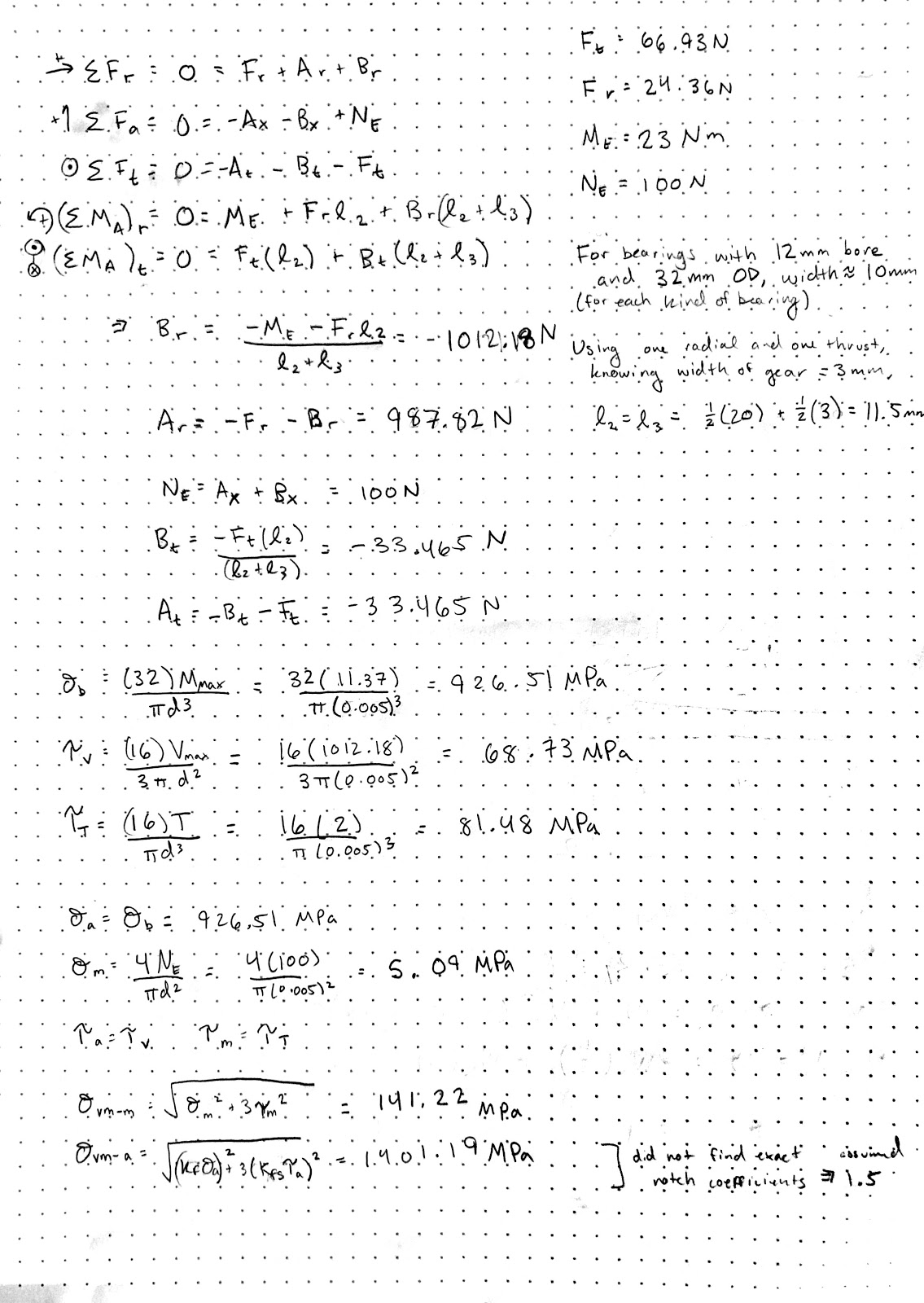
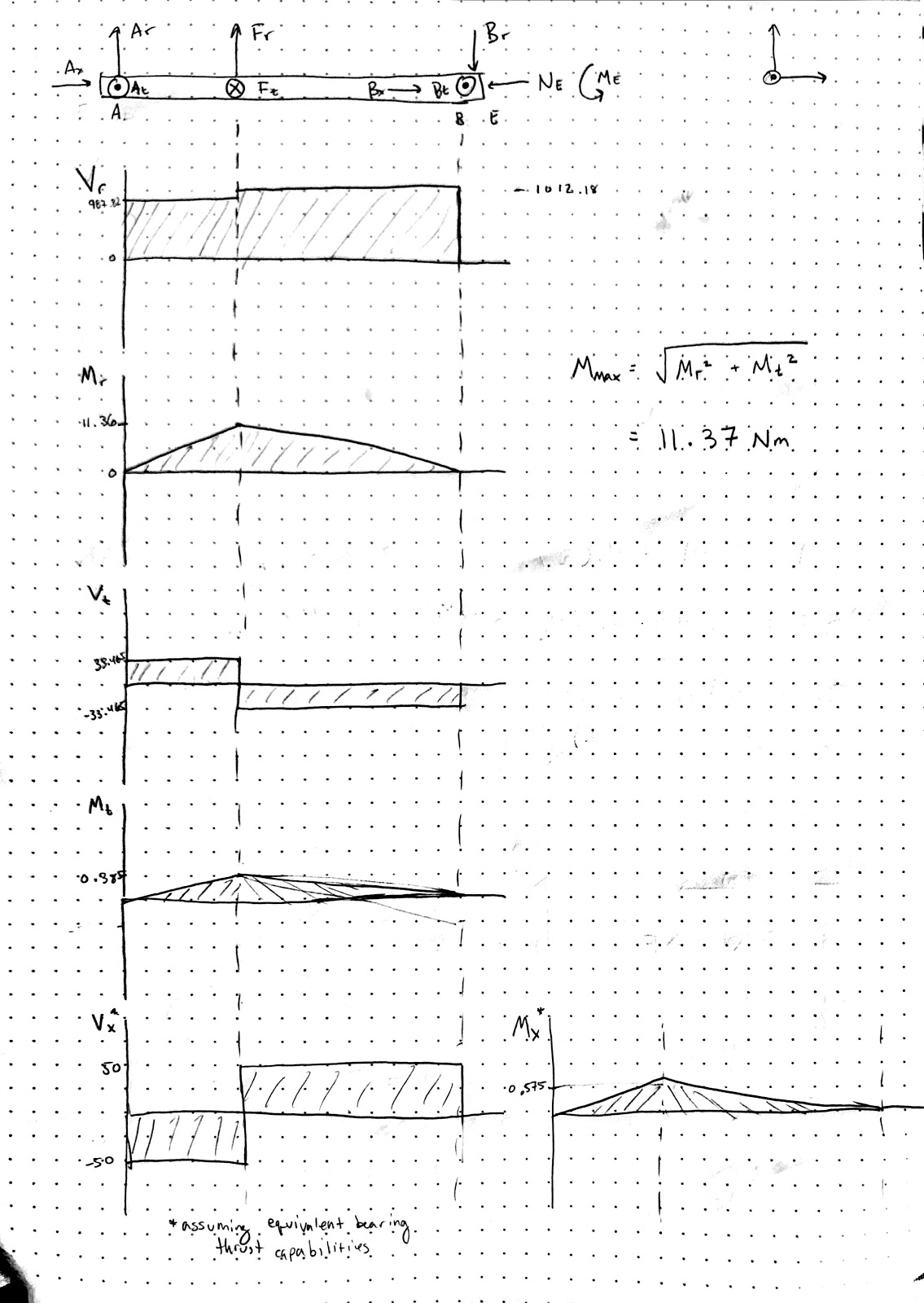
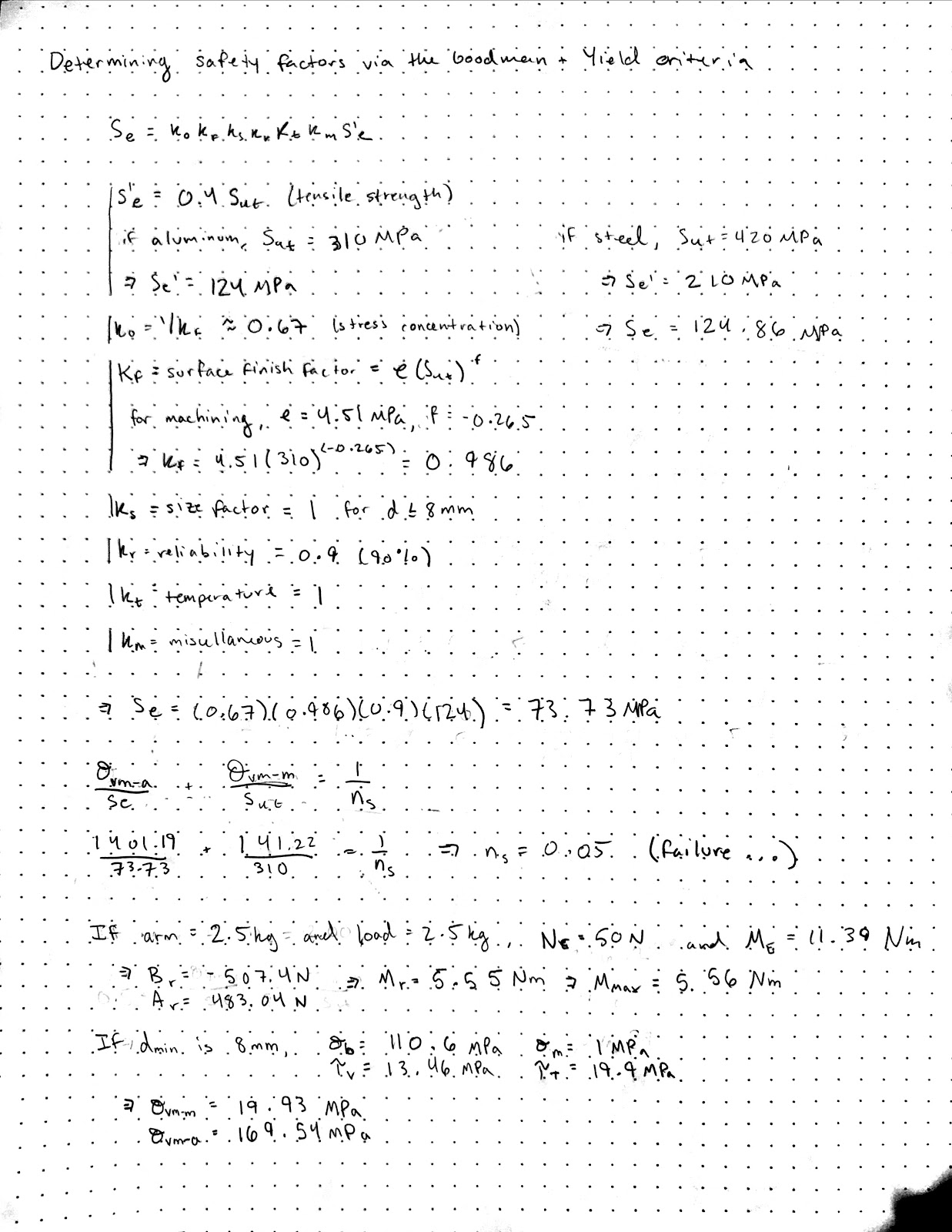


Figure 5: Shear Force and Bending moment diagrams for an internal shaft



The below calculations in Figure 6 consider a standard aluminum and steel shaft to calculate coefficients of the endurance limit of the shaft.

Figure 6: Safety factors of internal shaft



The Goodman line uses the von-Mises stresses to determine the safety factor - ideally, something above 1.2. Upon initial calculation of the stresses, the bending stress caused immediate failure of the part. Therefore, the forces were re-calculated assuming half of the loading - the arm and load are each now 2.5kg. This reduces the overall bending moment by half, thus decreasing the bending stress. These calculations are shown in Figures 7 and 8, below.

Figure 7: Adjusted loading calculations

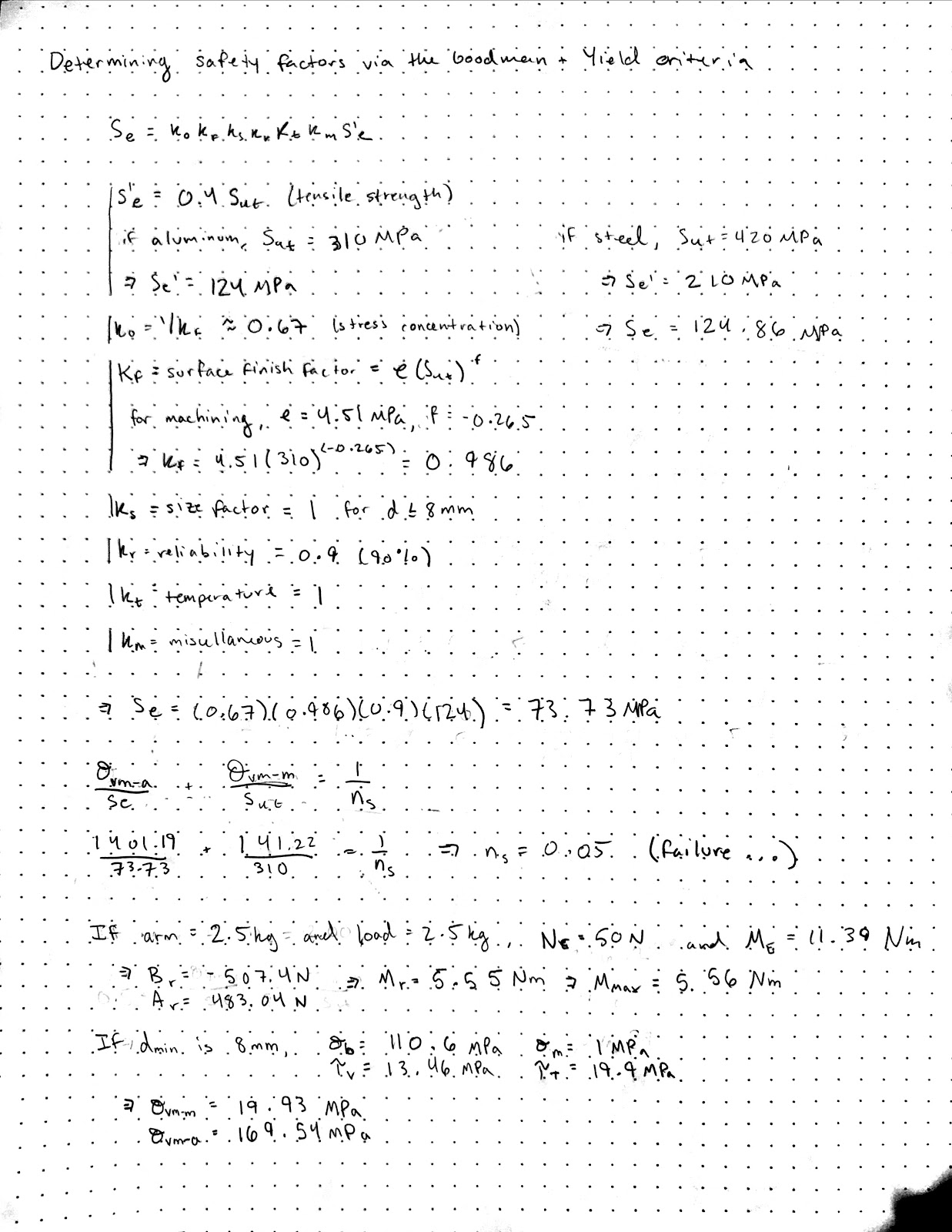
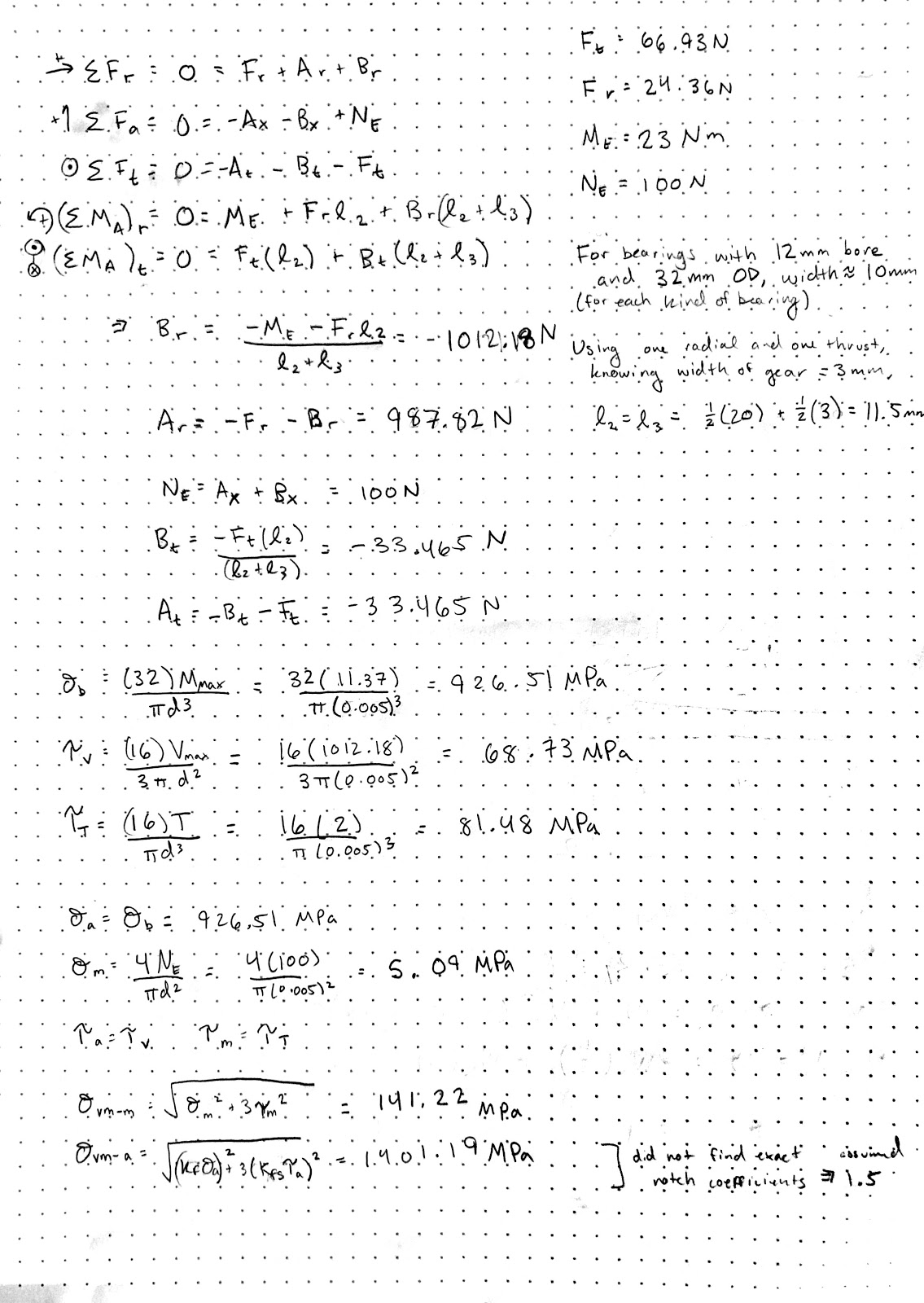
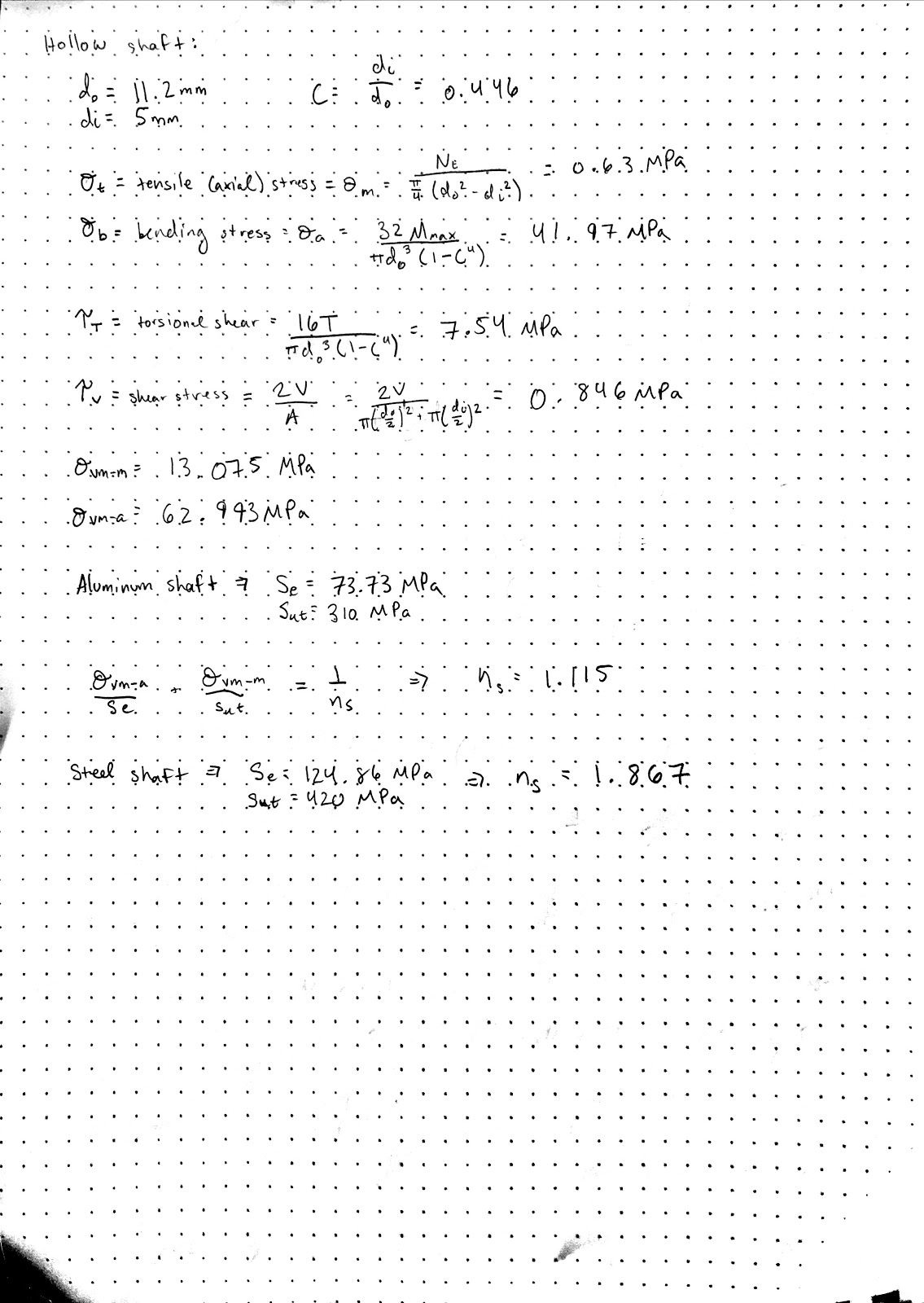


Figure 8: Corresponding forces and stresses



An initial calculation was done as if the shaft were a solid cylinder of radius 8mm - however, this always resulted in failure due to bending and is not an accurate depiction of our system. The following calculations, shown in Figure 9, were done next assuming a hollow shaft. Both aluminum and steel were tested.

Figure 9: Hollow shaft calculations



The result of these calculations was the design decision that our shaft, in order to accommodate the large bore hole, needed to be non-load-bearing, which resulted in an irrelevant wall thickness. The bearing loads indicated that there would be some thrust load, but significantly less than the axial load. Deep-groove ball bearing would fulfill this loading requirement. Geometric convenience as well as functionality led to the decision to use thin-section deep-groove ball-bearings. SKF bearings are well-documented, and using our known bearing loads and desired geometry, we were able to utilize their safety calculator and verify which bearings would work in the system. The bearing around the output is larger than the one at the base, although both contribute to the stability of the system. The larger bearing is an equivalent to SKF 618062RS1, and although the smaller bearings did not have an exact equivalent, the load ratings were available on the manufacturer’s site and fit our desired range. The bearings present in the final design are as follows:

|  |  |
| --- | --- |
| **Large Bearing** | **Small Bearing** |
| 30mmx42mmx7mm | 20mmx27mmx4mm |
| Dynamic load: 4.49kN | Dynamic load: 1.05kN |
| Static load: 2.9kN | Static load rating: 0.73kN |
| Deep-groove ball bearing | Deep-groove ball bearing |

#### Motor Selection

Our motor, the Maxon EC 45 flat, 42.9mm, brushless, 30 watt motor, came as the result of power, geometry, and control constraints. Since we wanted a small actuator, we selected a pancake motor to permit a thin-profile actuator. In order to increase the power density, we selected a brushless motor to reduce power loss compared to a brushed motor.

Our goal was to provide around 30 W of power at 2 N-m, so we selected the motor power rating to be right at 30W. To reduce ohmic heating, we picked the highest permissible voltage rating (24V) in order to deliver the desired power with the lowest possible current. Finally, we opted to have hall sensors in order to have closed-loop control and to obtain the best possible information about the motor motion.

### Electrical

#### Heat Analysis

A heat analysis was performed for the FETS such that the following was found:

Resistance of the FETS (Drain-to-Source): 1.5 mΩ Max

Power at 5A (stall) = 0.0375 W

Thermal resistance (junction to ambient) of the FETS: 125 C/W max

Heat change (stall): 4.6875 C

The heat analysis for the traces can be found below with regard to the PCB.

#### PCB Decisions

Placement of items on the PCB were driven by mechanical constraints, current draw traces, and distance between complex elements.

Since our imperative was to design the smallest, most powerful actuator possible, we had to have a PCB that would fit in the small housing size. This small PCB required us to pick very tiny components: we selected 0.02” x 0.04” SMDs wherever possible. In addition, the location of our motor and output gear determined the placement of magnetic and optical encoders. We also placed the connectors for cabling, MCU flashing, and debugging near accessible points on the PCB.

For powering the motor, we used the general flow: 24V in -> Capacitors -> FETs -> Motor. Our motor had a stall current of ~5A, so our motor drive traces had to be fairly large in order to accommodate the current that the PCB would carry during extreme cases. The trace thickness was calculated through online references ([Bittele Turn-Key PCB Assembly](https://www.7pcb.com/trace-width-calculator.php)) which used the IPC-2221 standard using the formula:

https://lh4.googleusercontent.com/njPGMyHQ9Mrn8geyY4SGsldAVLTra89TRJuOYH17EpdrJdK5deYkQpQWe9LnskfNGzc7UDjhuiX291h46tK0tzsyVn0XNb9v2UFe9XTTAHlxs0W8kN8UbWZiZo25ZJYLtju8d2su

This yielded power traces of 177mil. Each of the 3 current inputs to the motor had their own respective polygons on the PCB. In addition, there were large power planes to accommodate the large motor and 24V traces, as well as ground planes for further noise reduction. Both of these features also provided heat dissipation from the MOSFETs.

On the other side of the board from the power components, we had the logic components. The most complex components (those with the highest # of pins) -- the MCU and motor driver -- were placed close together to minimize the number and length of connections that needed to be made. The debug and programming pins were also placed close to the MCU. The digital and power components were placed on opposite sides to increase isolation.

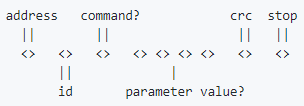
### Software

#### Communication Implementation

To allow systems to communicate to one or more Avocado Actuators, a message protocol has been implemented on top of a RS-232 communication bus. The message protocol is abstracted out to several functions that the user is given. These functions construct the message send it along the communication bus.

All communication with avocados is initiated by the master computer (heretofore referred to as the brain). The brain has a static address, each individual avocado has its own user settable address. All messages fit in a maximum of 8 bytes to maximize bandwidth and throughput. The standard message (a few exceptions are discussed in detail later in the guide) has an address byte, a command byte, optionally four bytes specifying a parameter value in the case of sets, a crc byte and a stop byte. These components are discussed further in the protocol section. Figure 10, below, depicts the protocol (<> represents one byte, ? means optional).

Figure 10: Protocol



###### Multiple Avocados

Each message is prefixed with a byte address. This allows specification of addresses well past the limits of our electronics or communication bandwidth. Once an avocado's address has been set communicating with it is as simple as specifying the correct address in each message.

To set an address one must connect the brain and avocado one wishes to set with no other avocados on the bus. One can then send a set address message (see specific messages listed later in the guide for details) with the new address. This message is broadcasted to all avocados on the bus which is why it's important to only have one connected at a time while setting addresses.

###### Protocol

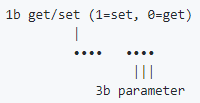
The standard message has an address byte specifying which device the message is intended for, a command byte specifying the action to take, a crc byte for corruption checking and a stop byte indicating the end of the message. Messages generally fall into two categories of gets and sets. If the command is a set then the new value is given by 4 bytes following the command byte. These four bytes give the new float value for the parameter specified in the command byte.

The address byte simply specifies an integer address corresponding to a specific avocado. The crc byte is computed using a CRC-8 implementation included in our source code. The stop byte is currently the ASCII character "!". The command byte is explained in detail in the next section.

###### Command Byte

The command byte contains two pieces of information. The first bit specifies if the command is a get (a 0) or set (a 1). The middle four bits are currently unused. These bits can be expanded too as functionality is added. Finally, the last three bits specify which parameter to get or set. The specific bitmasks corresponding to each parameter are given in the next section. If the message is a set command the the four bytes following the command byte specify the float value with which to set the given parameter. Figure 11, below, shows the byte separated into nibbles with their different information.

Figure 11: Byte separation



The parameter bits (least significant three bits) are encoded as follows:

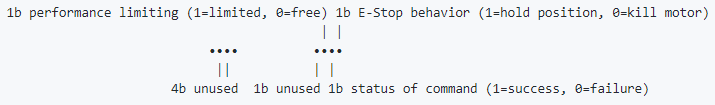
1. 000 (set) - Address
2. 001 (get) - Temperature
3. 010 (get|set) - Current
4. 011 (get|set) - Velocity
5. 100 (get|set) - Position
6. 101 (get|set) - Maximum current
7. 110 (get|set) - Emergency stop behavior
8. 111 (get) - Status register

The middle four bits are for future parameter expansion.

###### Command Responses

Every command from the brain (aside from the exceptions given in the following section) receives a response specifying E-Stop behavior, status of the most recent command and whether or not the avocado is being performance limited. If the E-Stop behavior bit is a 1 that means the avocado will hold position until a safety threshold is reached, while a 0 means it will kill motor power. A 1 for the performance limiting bit indicates the avocado is currently being limited while a 0 indicates it's running freely. Finally, a 1 for the status bit means the last command succeeded, a 0 means it failed. The command responses are shown in Figure 12, below.

Figure 12: Command responses



###### Special Commands

**Heartbeat:** Gives no response. Broadcast to all connected devices using address 11111111 (address reserved for heartbeat).

**Setting Address:** Gives no response. Broadcast to all connected devices using address 11111110 (address reserved for setting avocado addresses).

# The Design Evolution

There has been extensive evolution from its first iteration to now, beginning in the middle of the winter. The most notable changes are categorized below.

#### Gearing

###### Winter Midterm

Both of the gears were proposed to be standard sizes, available online, with the following parameters:

|  |  |
| --- | --- |
| **Pinion:**  15 teeth  5mm bore  17mm OD  Steel | **Gear:**  120 teeth  25mm bore  122mm OD  Steel |

Although the availability and strength of gears worked, the bore hole on the pinion was too large for the 4mm motor shaft, and the outer diameters of both gears were larger than desired.

###### Winter Final

The gearing remained loosely the same, with the following parameters:

|  |  |
| --- | --- |
| **Pinion:**  15 teeth  4mm bore  12mm OD  Steel | **Gear:**  120 teeth  10mm bore  96mm OD  Steel |

The sizes were slightly smaller, but still larger than ideal. The output bore size was now too small for the proper connectors to fit, and holes for the output would need to be machined into the gear.

###### Spring

By midterm of spring, the gears had been finalized. The output gear is now a full output apparatus including a shaft and rotor, SLA printed to fit the strength requirement. The pinions were found online and the bore hole machined from 3mm to 4mm.

|  |  |
| --- | --- |
| **Pinion:**  12 teeth  4mm bore  8.4mm OD  Steel | **Gear:**  96 teeth  15mm bore  77mm OD  ABS |

The custom output gear allows for manipulation in accordance with the assembly, and the 96:12 tooth ratio gives still allows for the desired torque output.

#### Position Encoding

###### Winter Midterm

The biggest challenge in absolute position encoding was the possibility of an off-axis encoder, or one with a through-bore large enough to accommodate our connectors (minimum of 15mm). At this point, we had found a Nonius track offset encoder that could be ordered from iC-Haus/Bogen, Renishaw, or PWB Encoders. We contacted the vendors and hoped for the best.

###### Winter Final

After calculation of the exact size of connectors, we determined the bore hole absolute minimum was 11mm. We had selected an on-axis encoder with the following specifications:

**Absolute Position Encoder**

iC-MU DFN16 5x5

SPI Communication

Image of encoder in Figure 13, below.

Figure 13: Absolute On-Axis Position Encoder



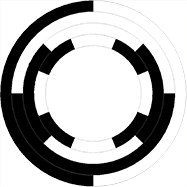
This encoder, although doable, had a lengthy lead time, complex calibration, and very tight alignment requirements.

###### Spring

By the spring midterm we had determined that the hassle of the on-axis encoder was not worth its value. The iC-MU150 absolute magnetic encoder, while was an off-axis encoder that could go around the output shaft, didn’t have a large enough magnetic disk readily available, required an additional EEPROM to initialize device, and was very dependent on precise calibration. The new and final encoder system consists of two encoders: a standard magnetic encoder on the pinion, and an optical encoder on the output. The encoders have the following specifications:

|  |  |
| --- | --- |
| **Magnetic Encoder** | **Optical Encoder** |
| AS5048A High-Resolution | Custom sticker with 4-bit greycoded sectioning |
| 14-bit rotary position sensor | Miniature Reflective Object Encoder QRE1113GR by Fairchild Electronics |
| Digital angle interface and PWM output | Image below in Figure 14 |

Figure 14: The optical encoder disk



The two encoders together give higher resolution on the output position, which satisfies the desired position control requirement. The magnetic encoder gives a 14-bit resolution within 1/8 of the output gear due to the gear ratio. The optical encoder indicates which 1/8 of the gear the Avocado is positioned in. The fourth bit on the optical encoder is to account for any misalignment between the edge of each optical section and the 360-0 edge of the magnetic encoder.

#### Mechanical Housing

###### Winter Midterm

The housing has undergone many iterations, but the major modifications will be documented here. The first housing iteration accommodated for larger gear sizes and did not take into account electrical integration. It consisted of one primary case with e atop cap. The inside was open with the exception of a thin casing around the motor for heat sinking. The intended material was aluminum. The general housing structure can be seen in Figure 15, with an exploded view in Figure 16.

Figure 15: Initial housing

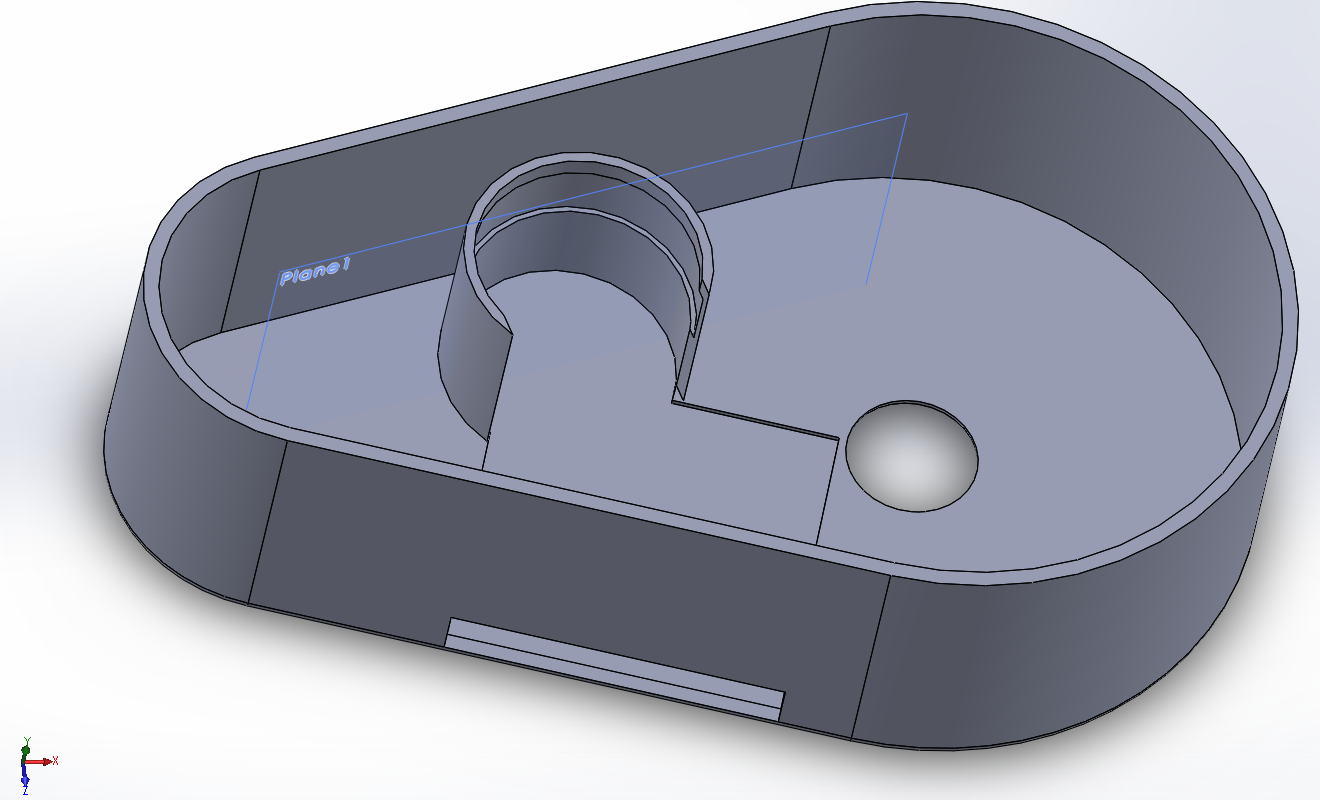


Figure 16: Exploded view of initial housing



###### Winter Final

The next iteration of the housing took the PCB into more detailed consideration, as well as furthered the idea of heat sinking. The gear geometries were still oversized, and the housing was only a case with fins and a top cap with an additional output rotor, as shown in Figures 17 and 18.

Figure 17: Second housing assembly

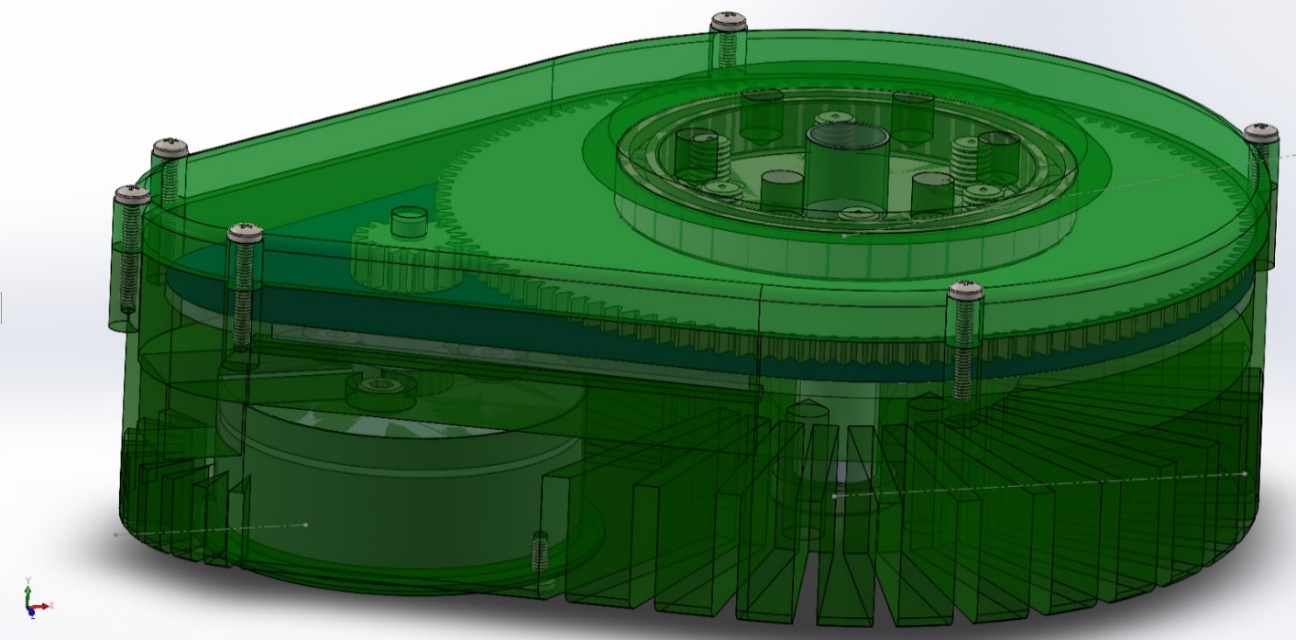
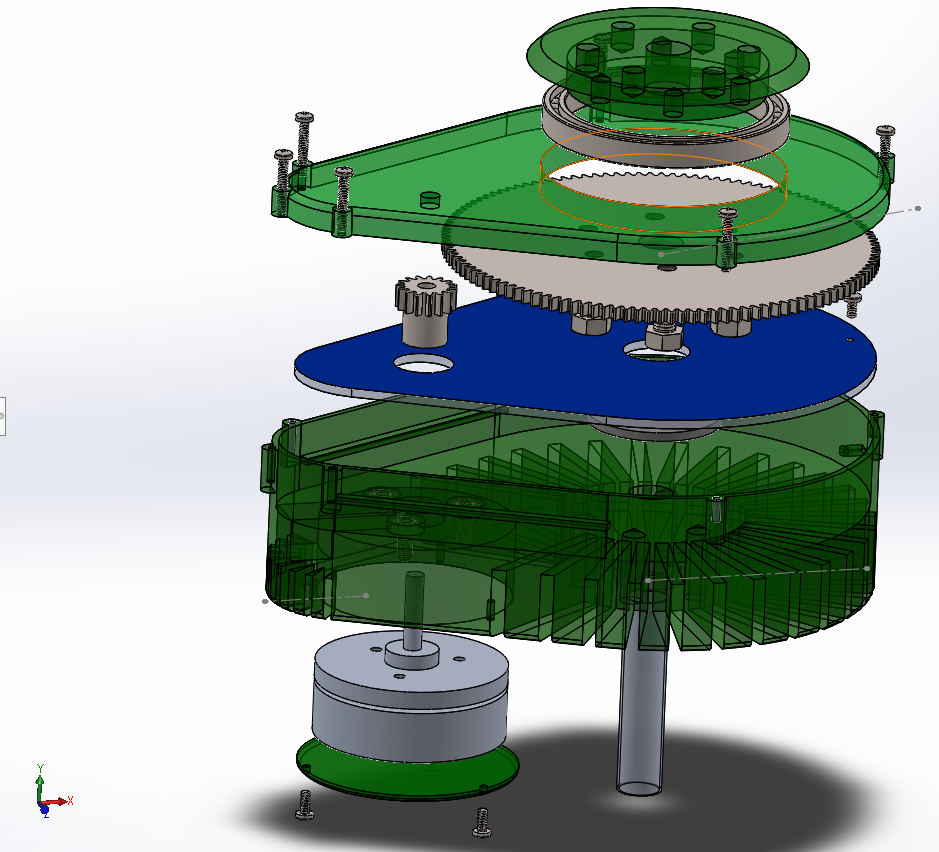


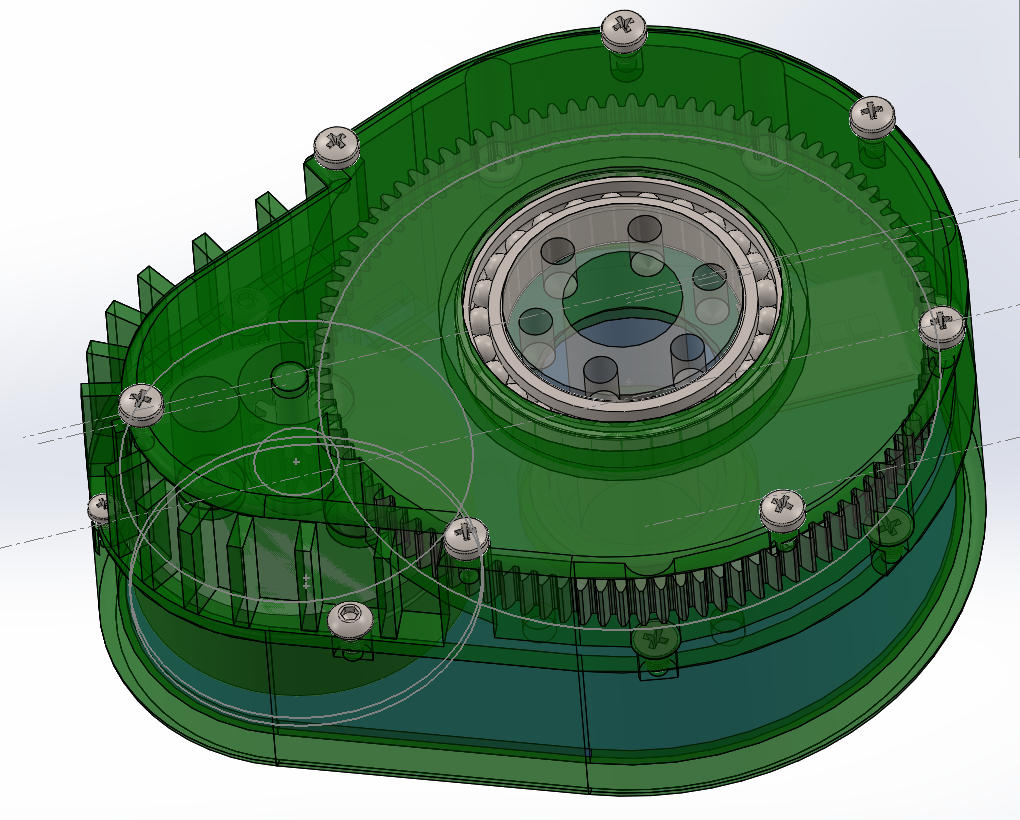
Figure 18: Exploded view of second housing assembly



###### Spring Midterm

At the midterm of this quarter, the design remained mostly the same but with further development to include the PCB and its various connectors, as well as accommodate the off-axis encoder setup. The fins for heatsinking were removed, as thermal mass was proven sufficient, and the output gear/rotor/shaft part had been finalized. The new design was divided into two halves and a top cap, with the bottom encasing the motor and PCB and top encasing the gearing. Although the design was a more efficient improvement, it was going to be difficult, if not impossible, to manufacture. The design is shown below in Figures 19.

Figure 19: Third housing assembly



###### Spring Final

The final design follows a similar compartmental structure, although with an easier to manufacture setup. The design features a middle plate with the primary heatsinking mass, and top and bottom plate with bearing seats and mounting holes but few other features. The bottom plate is connected with standoffs, leaving openings for all electronics and allowing a controlled height between the magnet and the encoder. The design is shown in Figures 20 and 21, below.

Figure 20: Final Avocado assembly

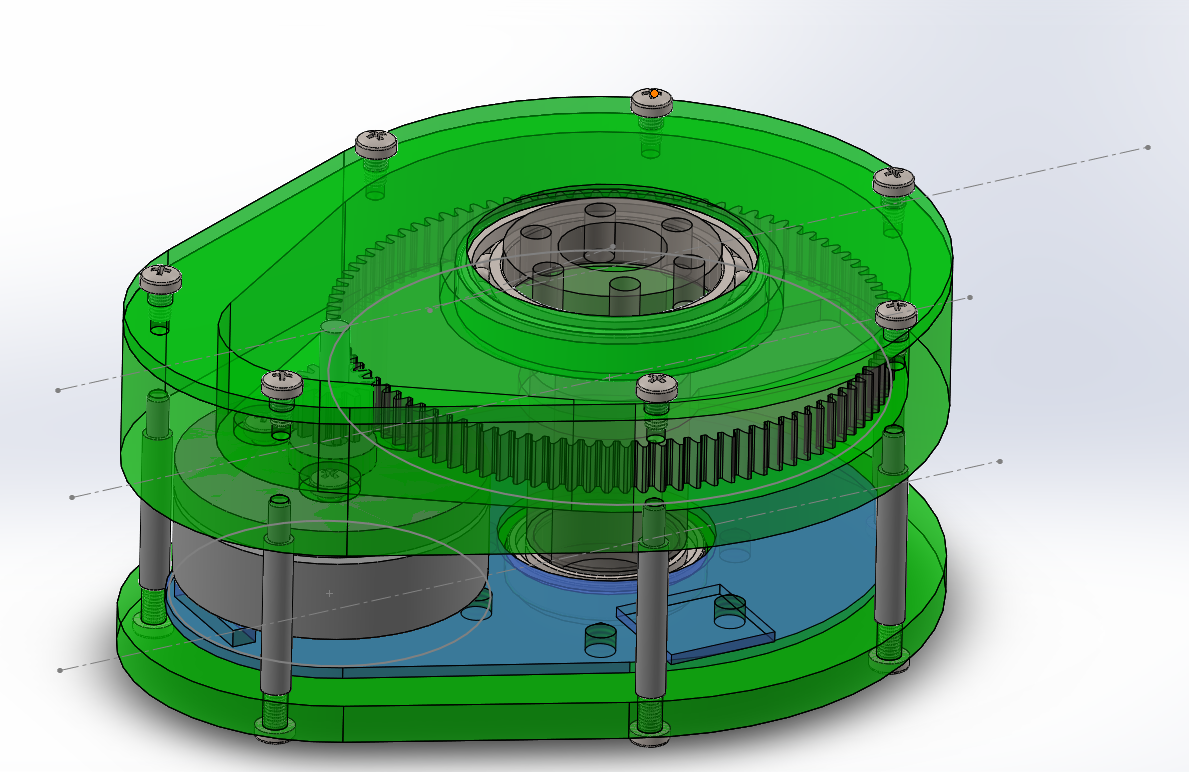
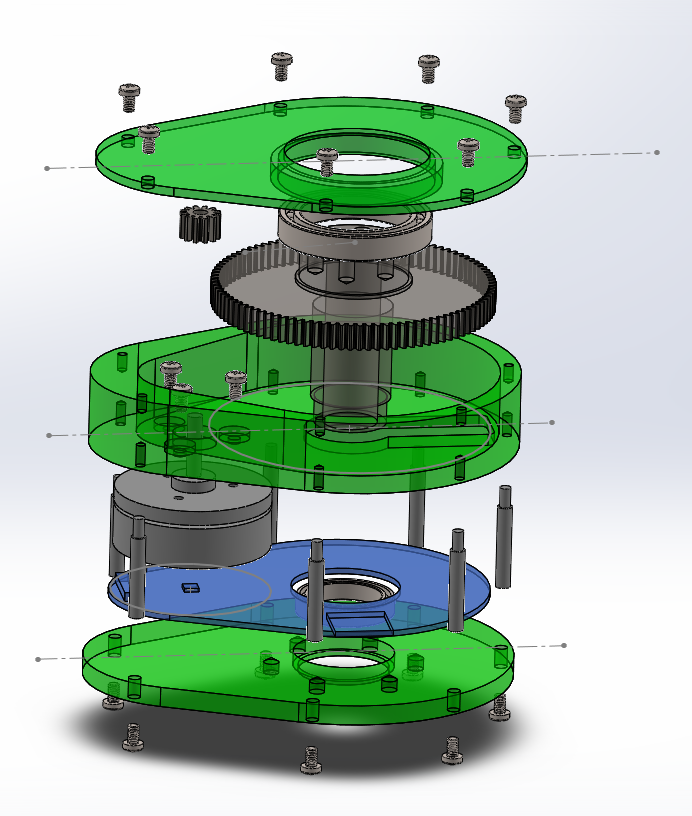


Figure 21: Exploded view of final assembly



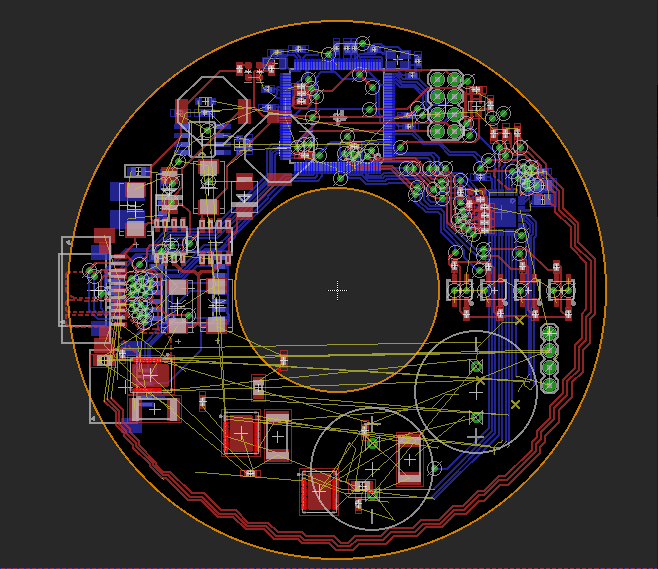
#### PCB Design

The general schematic of our circuit did not change significantly over the course. We swapped out a few small components based on availability and rating (picked a higher power voltage regulator and higher inductance inductor). Due to the height of our original bypass capacitors, we swapped out the large cylindrical ones with several flat SMDs in parallel. The design overall was primarily limited by size and shape constraints; we had to design around the large through-hole, which required strategic routing of digital lines between gate driver and MCU.

###### Winter

There was one major redesign of PCBs, V1.0 and V2.0.In V1.0, we had 3 PCBs: One primary donut PCB, a peripheral board, and an encoder board. These boards had several interconnecting cables, which we sought to reduce in the next version. We also wanted to have an optical encoder that was able to be mounted closer to the sticker. The winter design is shown below in Figure 22.

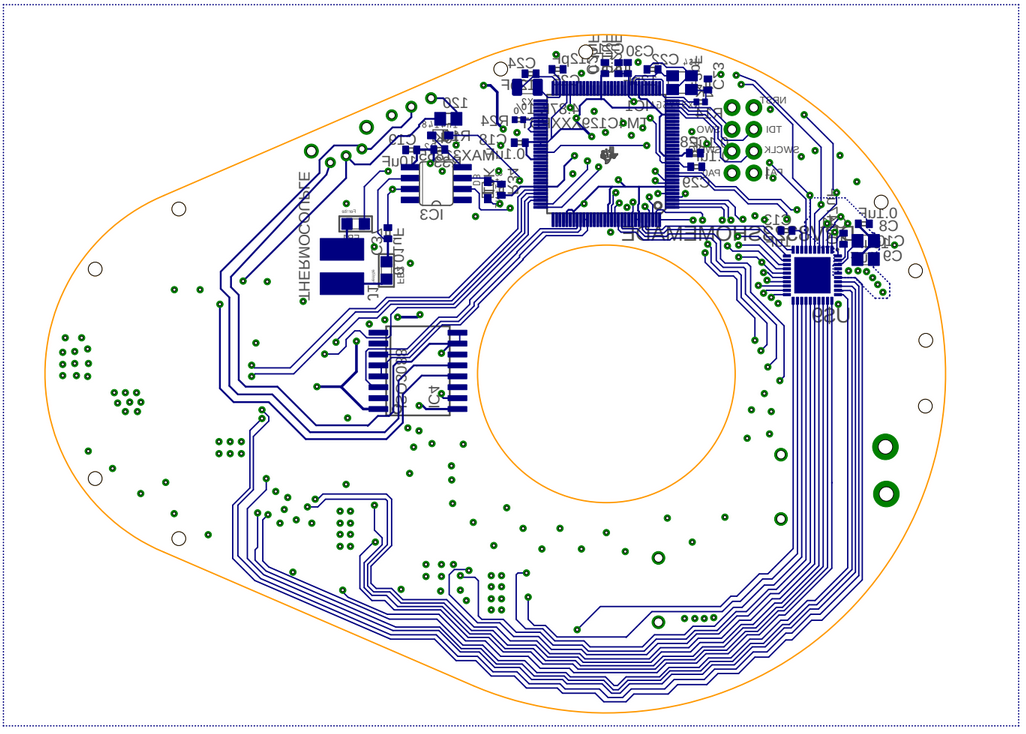
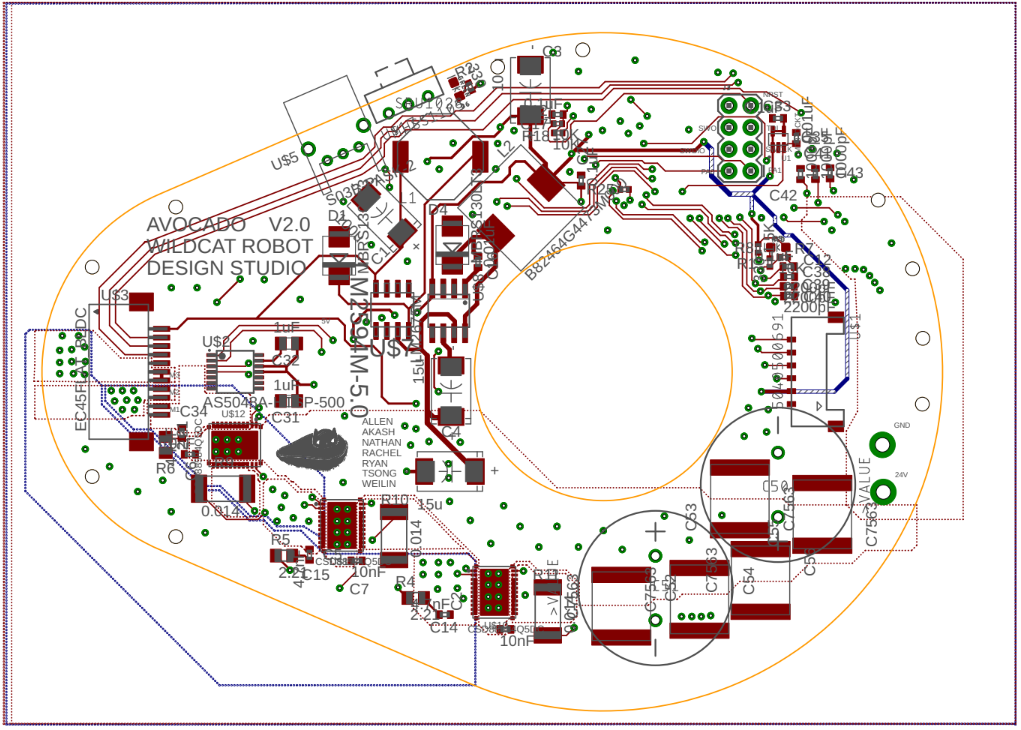
Figure 22: Initial PCB design



###### Spring

In V2.0, we have an avocado-shaped PCB that allowed for easier wiring and decreased the internal congestion in our already tiny device. After the V2.0 redesign of the PCB, we realized that we needed hall feedback from the Motor in order to have accurate current control. Although our PCB had no specific traces between the MCU and the hall sensors, we were able to solder fine magnet wire from the hall sensors to unused GPIO pins. The updated design is shown below in Figure 23.

Figure 23: Final main PCB overlays



#### Communication

###### Winter

The original specification for the Avocado had EtherCAT as the physical layer, and left the above layers in the network stack unspecified. After looking into EtherCAT, it was deemed overkill for the project, and the spec was changed to RS-485, an industry serial communication standard used in many large distributed system applications.

RS-485 is known for its resistance to noise and robustness over long bus distances by using a twisted-pair cable and differential signals. The Avocado Actuator is based on a Tiva TM4C1294NCPDT Microcontroller, which does not have any native RS-485 capability built in, so after successfully testing the feasibility of RS-485, a RS-485 transceiver was included in the PCB design. The Tiva chip would output RS-232 (standard UART communication) and the transceiver would translate to RS-485 before the data entered the communication bus between the main system brain and other Avocados.

###### Spring

Once the PCB was finished and delivered, testing the physical layer of communication (the RS-485) could begin. Unfortunately, there were many problems with the transceiver and in the end it was preventing the Avocado from communicating with anything. After discussing the issue with the project sponsor, the transceiver was removed from the PCB, the correct pins were bridged, and the physical protocol became RS-232, standard UART communication. After continued testing, a functioning RS-232 bus was established that Avocados and other Tiva microcontrollers could connect to via RS-232.

In parallel with developing the physical layer of communication, the message protocol was also being developed. It started out as a keyword-based system; if the programmer wanted an Avocado to turn to 90 degrees, the message passed to the avocado would look like “set pos 90”. However, while this protocol is easy for a human to understand, it’s very bloated when compared to what a machine would be able to understand. The message protocol went through several iterations of slimming down so that the maximum possible bandwidth could be achieved. For a look at the current message protocol, please see the section on the communication implementation.

# The Final Design

After the extensive design revisions detailed above, we settled on the final design of the Avocado. The design consists of the following major parts and features:

#### Parts

1. Top cap
2. Aluminum middle plate
3. Bottom plate
4. Avocado PCB
5. Encoder PCB
6. 96 tooth output gear and shaft, with output rotor and optical encoder disk
7. 12 tooth pinion gear and set screw
8. EC 45 Flat 40W Maxon Motor
9. 20x27x4mm deep groove ball bearing
10. 30x42x7mm deep groove ball bearing
11. M1.2, M3, and M5 screws
12. 25mm standoffs

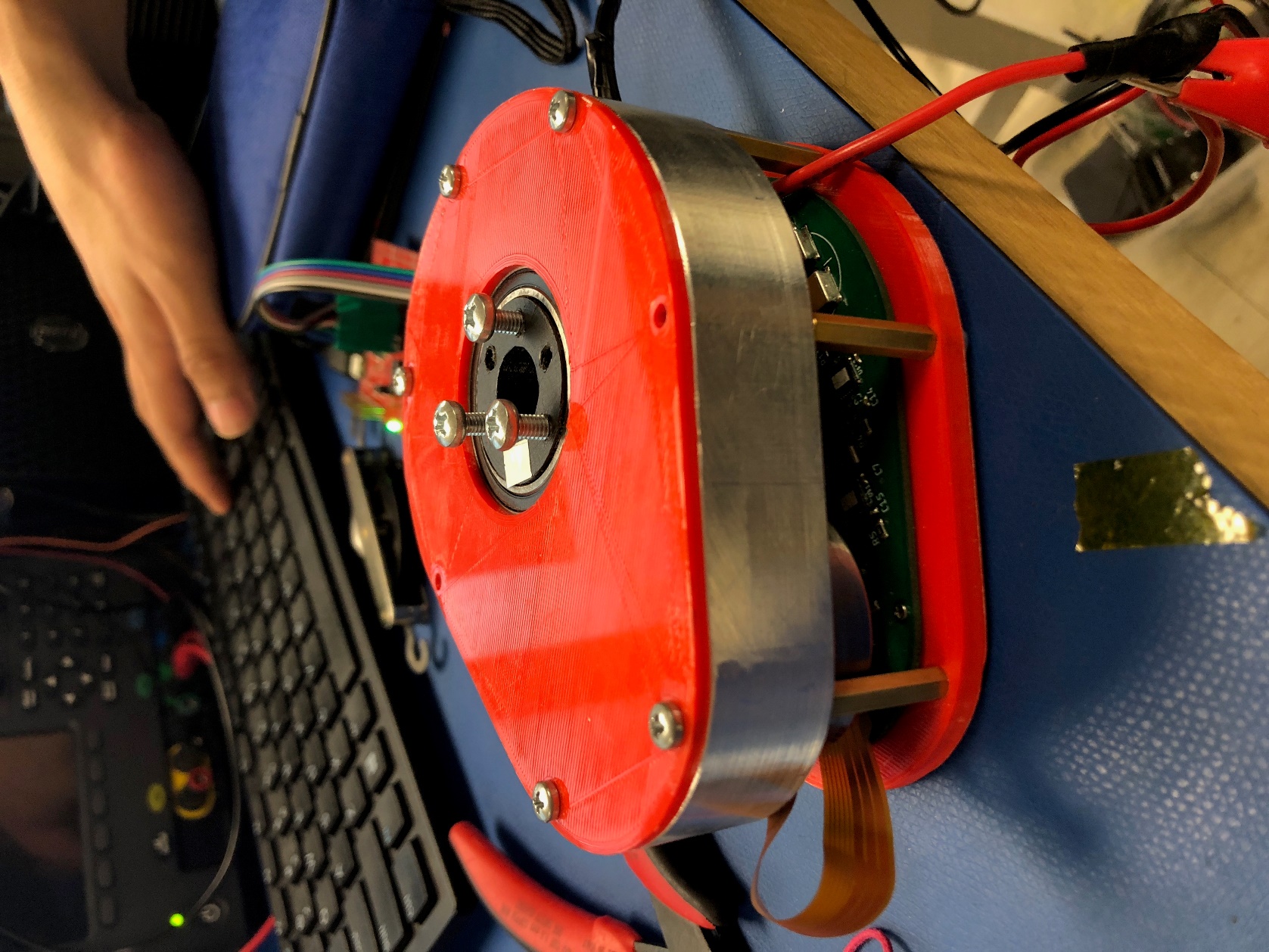
#### Features

1. API written in C
2. Position control
3. Velocity control
4. Current control
5. 2Nm stall torque
6. RS232 communication

The system is fully integrated, with the total assembly as shown in Figure 21, above. Moreover, the complete manufactured assembly can be seen in Figure 24, below. The Avocado gets its name from its shape, a result of one-stage 8:1 gearing and a through-hole. The design is compact, measuring 93mm wide, 118.36mm long, and 54.7mm tall with a 15mm through-hole for all cabling. The output rotor has 6 mounting holes for any desired attachment, with bearings rated up to 2.5kg of force at 300mm away from the axis of rotation. The device runs on 24V, and has fully programmable limits for temperature and current as well as built-in safety buffers. The Avocado can be controlled with absolute position data, derived velocity data, and current data, as well as limits set using temperature data.

Overall, the Avocado is a high-powered actuator with user-specified position, velocity, and current control all integrated in its compact housing. The BOM cost of one Avocado is $337.18.

Figure 24: Manufactured and assembled Avocado



# The Testing

#### Position Testing

Position control was tested by attaching visible, rigid rods to the output of the Avocado and measuring the angle at which those rods were pointing relative to zero with a protractor. We sent several different angle measurements to the Avocado in multiple trials and averaged the resulting experimental angle to compare to the desired angle.

#### Velocity Testing

Velocity testing was performed similarly to position testing but with a different measurement technique. Several desired velocities were sent to the Avocado in different trials and multiple readings of the Avocado’s speed were recorded with the tachometer. These speeds were averaged and compared to the desired speeds.

#### Torque Testing

Torque testing utilized the current control feature. A rod of known length was attached to the Avocado output rotor as well as to force gauges held by members of our team. The desired current sent to the Avocado was progressively increased and the force read until the desired range (or failure) was reached.

# The Performance

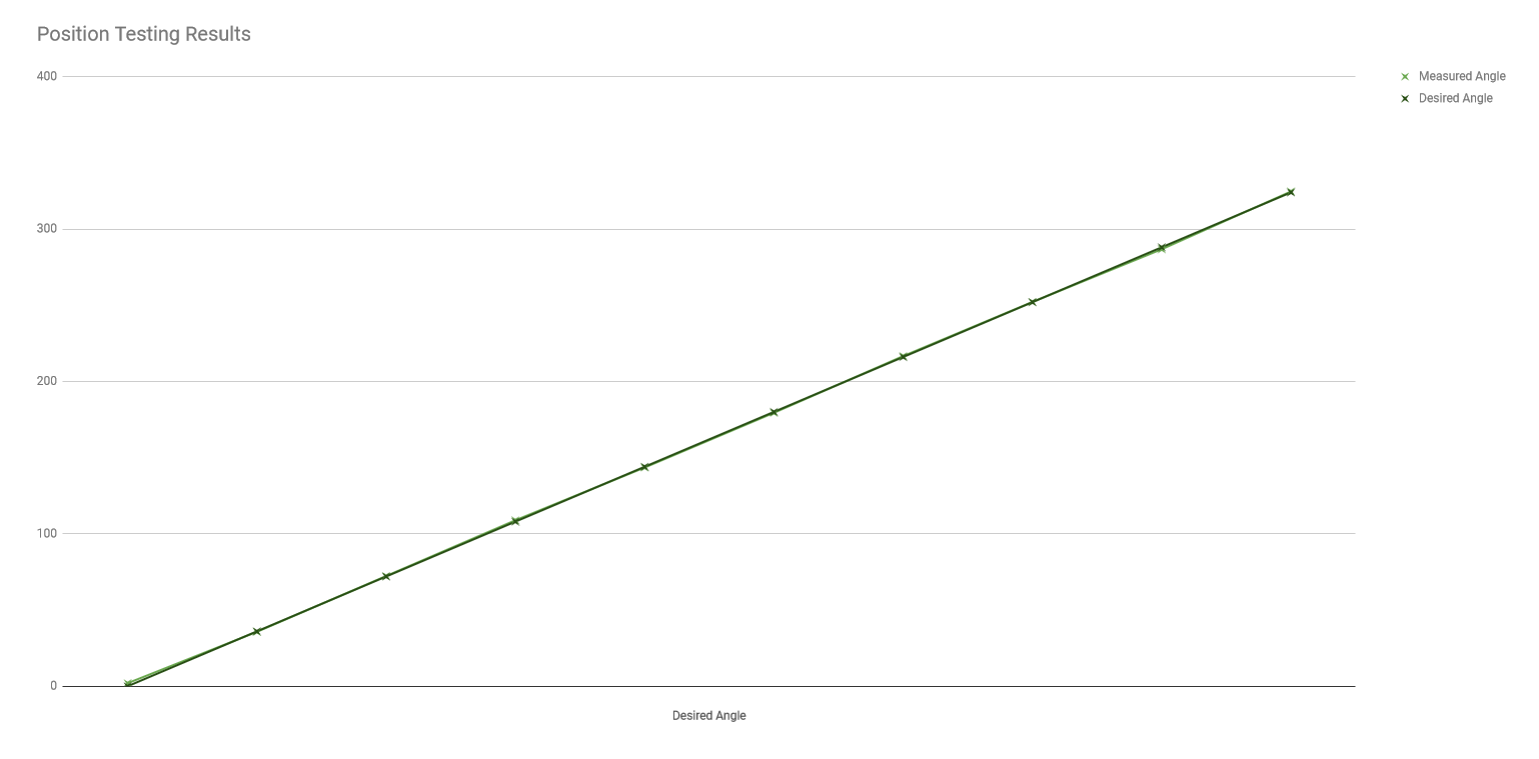
#### Position Testing

The results of the position testing are tabulated in Table 1, below, and plotted in Figure 25. Taking a sample point of 180 degrees, a standard error analysis was performed on the data.

Table 1: Position testing data

|  |  |
| --- | --- |
| **Desired Angle** | **Measured Angle** |
| 0 | 2 |
| 36 | 35.7 |
| 72 | 72.3 |
| 108 | 108.9 |
| 144 | 143.5 |
| 180 | 179.4 |
| 216 | 216.6 |
| 252 | 252.1 |
| 288 | 286.7 |
| 324 | 324.6 |

Figure 25: Position testing plot



Using the following analysis technique where σ is the measurement uncertainty and Δ is some multiple of σ and is defined as the difference between the predicted and measured values of a quantity, it can be evaluated whether or not the results are valid. In this particular case, comparing experimental and desired angle values,

and

This gives =1.2which implies 3and therefore the error is insignificant and the results are consistent such that position control is within the range of acceptability, although there is always room for finer tuning.

#### Velocity Testing

The results of the velocity testing are tabulated in Table 2, below, and plotted in Figure 26. Taking a sample point of 100 rpm, a standard error analysis was performed on the data.

Table 2: Position testing data

|  |  |
| --- | --- |
| **Desired Velocity** | **Measured Velocity** |
| 20 | 19.5 |
| 40 | 39.6 |
| 60 | 61.4 |
| 80 | 81.675 |
| 100 | 100.725 |
| 120 | 120.85 |
| 140 | 139.5 |
| 160 | 161.95 |
| 180 | 181.28 |

Figure 26: Position testing plot



Using the following analysis technique where σ is the measurement uncertainty and Δ is some multiple of σ and is defined as the difference between the predicted and measured values of a quantity, it can be evaluated whether or not the results are valid. In this particular case, comparing experimental and desired velocity values,

and

This gives =1.45which implies 3and therefore the error is insignificant and the results are consistent such that velocity control is within the range of acceptability, although there is always room for finer tuning. Velocity control in general was less accurate than the position control, although in both cases there was immense uncertainty in the measurement methods and further testing and tuning is recommended.

#### Torque Testing

The torque was calculated from the equation T = kI where k is the motor torque constant, in our case 51 mNm/A. The Avocado sustained 0.45Nm of torque for 30 seconds prior to unexpected melting of the motor. This test, therefore, was insufficient in proving the current control functionality and torque requirement and further testing is required.

# The Future

With more time, there are several modifications and improvements that could be made to the Avocado, as detailed below. The modifications consist of ideas that either came too late in the process or arose as long-term solutions to problems encountered during the last phases of testing.

###### Infrared Temperature Sensing

The current design uses a thermocouple attached to the stationary disk of the motor, but the hottest part of the motor is actually the rotating canister that houses the coils. Future designs would look into integrating a contactless temperature sensor, like the TI TMP006, which measures the IR radiation being emitted from an object using an embedded thermopile. This would also eliminate the need for additional thermocouple wires, which can get tangled or break down over time.

###### Current Sensing Method

Additional analog filtering would assist the current sensing accuracy. Moreover, connecting the hall sensors to GPIO pins to determine which current sense resistor to read from will yield cleaner results.

###### Slots and Adjustability

The addition of slots for adjustable gear mounting, most easily installed on the pinion side, would allow the user to account for backlash and different conditions during assembly. Although the specific gear meshing and choice of one-stage gearing are both helpful to reduce backlash, further adjustability would allow for even finer-tuning of the position control.

###### Communication

Initially the communication protocol was intended to be RS485, although the transceivers proved difficult to troubleshoot within our time frame. However, in the future we believe RS485 is a good option for communication.

###### GUI

Currently the Avocado functions from an API, although for a more expansive range of users, a simple user interface would improve the marketability of the Avocado.

# The Others

Other relevant documents can be found in the following locations:

[Manufacturing Guide](Avocado%20Manufacturing%20Guide%20-%20Final.docx)

[Quickstart Guide](Avocado%20Quickstart%20Guide-%20Final.docx)

[User Guide](Avocado%20User%20Guide%20-%20Final.docx)