Daily Log

Sunday, June 16, 2024 7:25 PM

6/10

- Cut router table 2 x 4s
- 

6/11

- Assembled router table 2 x 4 frame

6/12

- Reduced table height by 4 inches and added side panels
- Cut top work surface

6/13

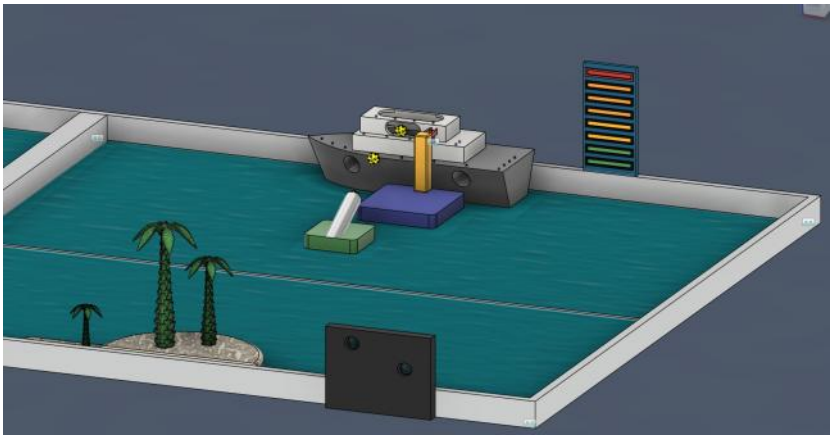
- Installed top, bottom, and back panels
- Added wheels
- First 2 coats of paint

6/14

- Last 2 coats of paint on router table

6/15

- Completed first iteration of game design
    - Initial CAD design of game
    - First iteration of rules and point system

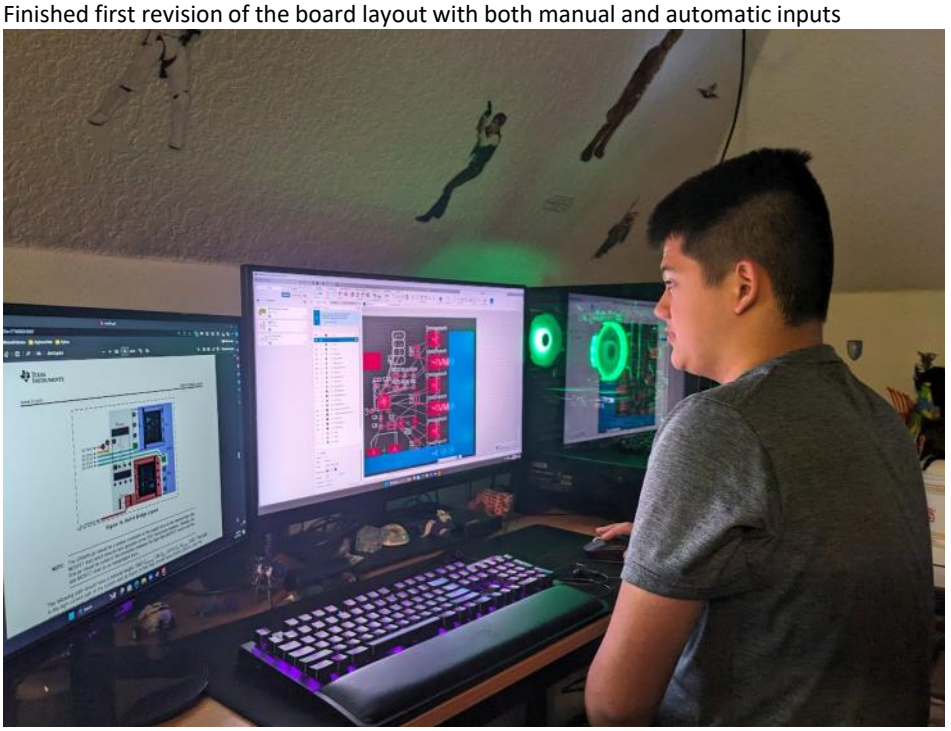
- 

6/16

- Created schematic for DRV8320 with parts selected from JLCPCB

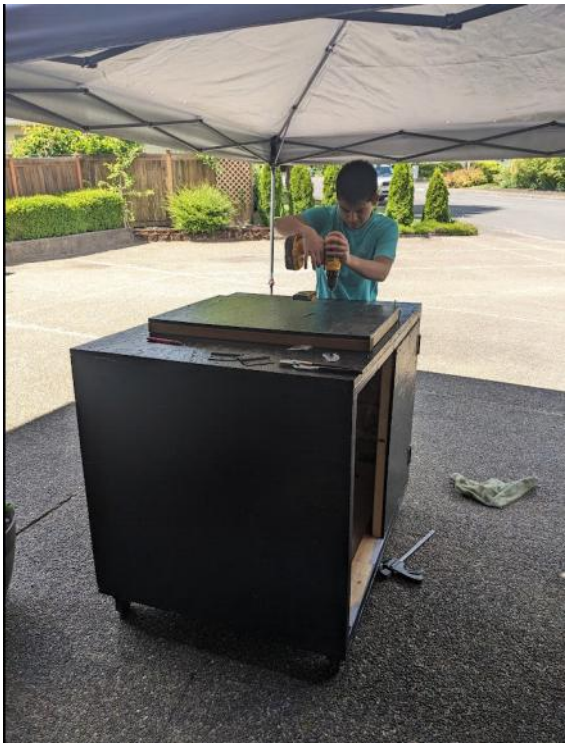
6/17

- ~ 4 hours
- Finished first rev of schematic
- Finished first revision of the board layout with both manual and automatic inputs



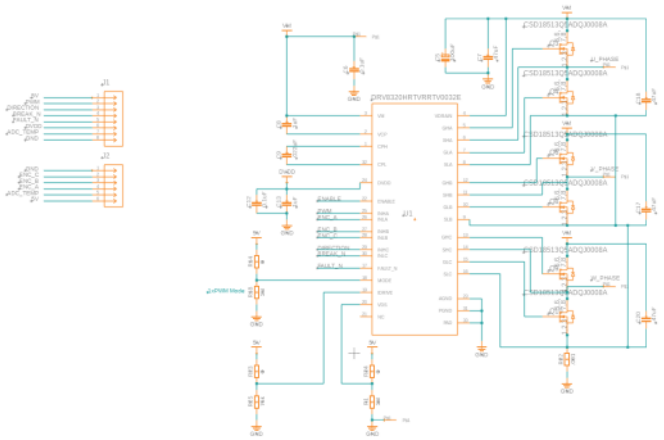
6/18

- ~ 7 hours
- Added doors to the router table
- Assembled the router
- Added PCB connectors and started looking at JLCPCB assembly requirements



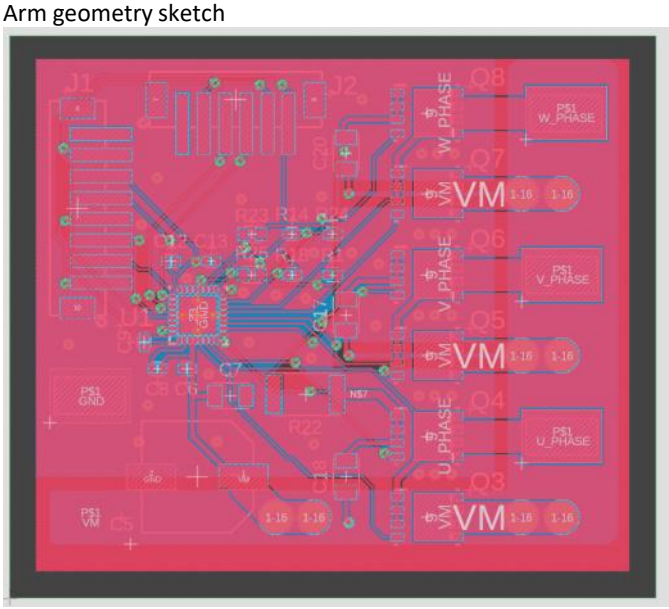
6/19

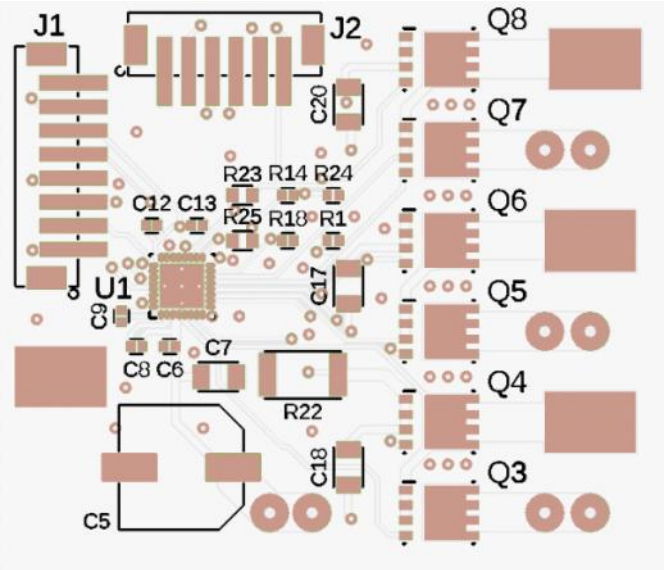
- ~ 4 hours
- CAM day 8 CADvent for the first router test
- Finished the first revision of the motor controller PCB

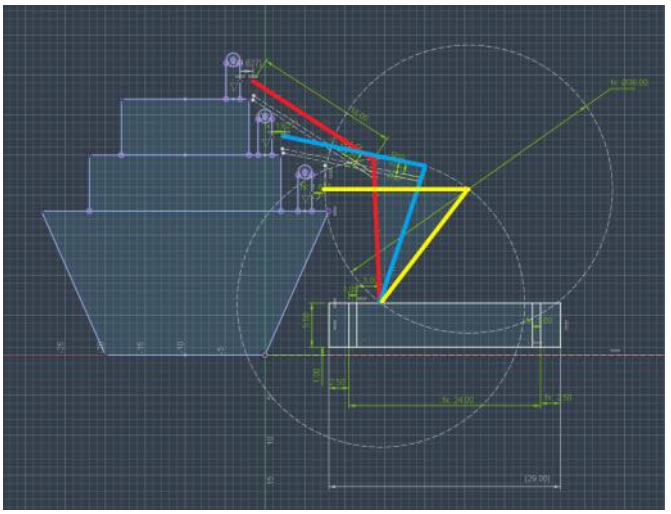


6/20

- ~ 5 hours
- Fixed PCB based off of visual inspection and JLCPCB DFM tool
- Submitted motor controller PCB to JLCPCB
- Brainstorming robot designs for game
- Arm geometry sketch



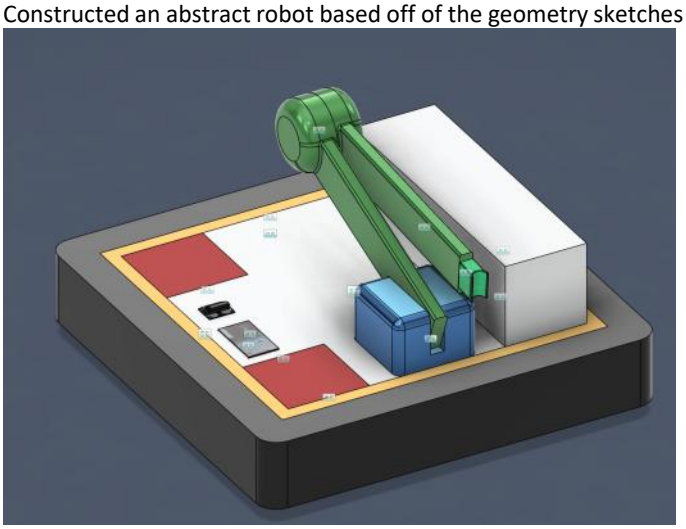


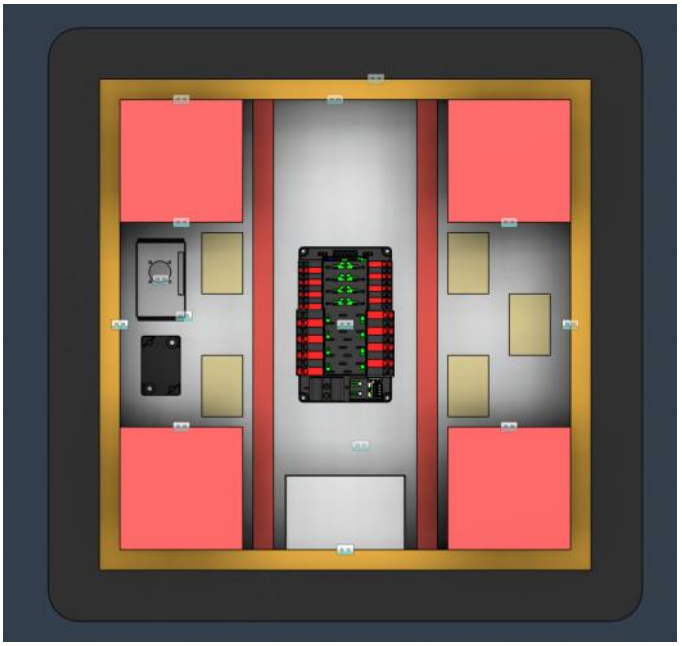


6/21

~ 3 hours

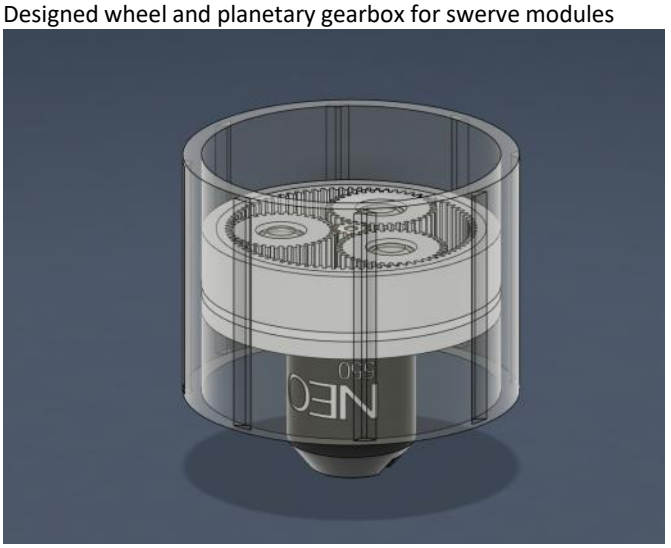
- Constructed an abstract robot based off of the geometry sketches





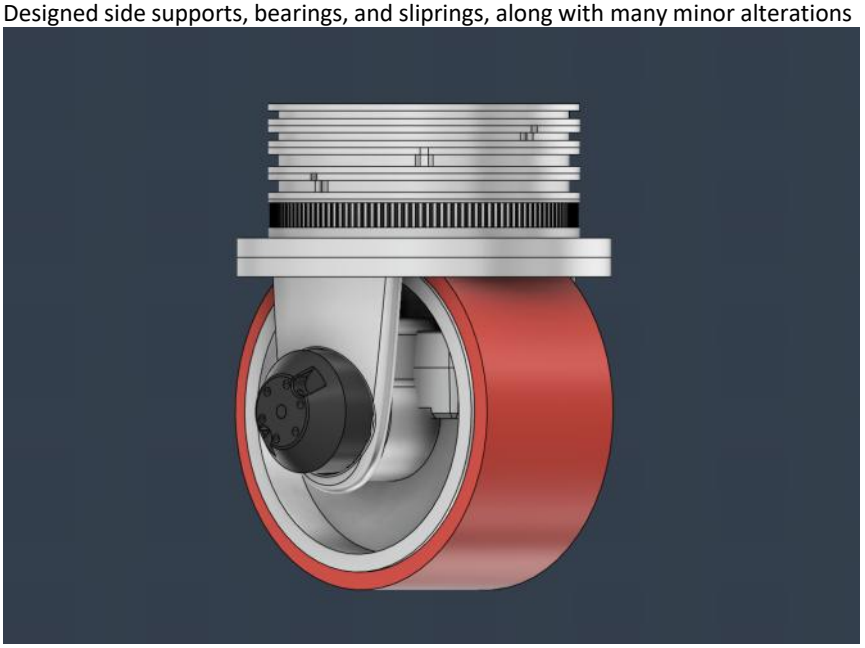
6/22

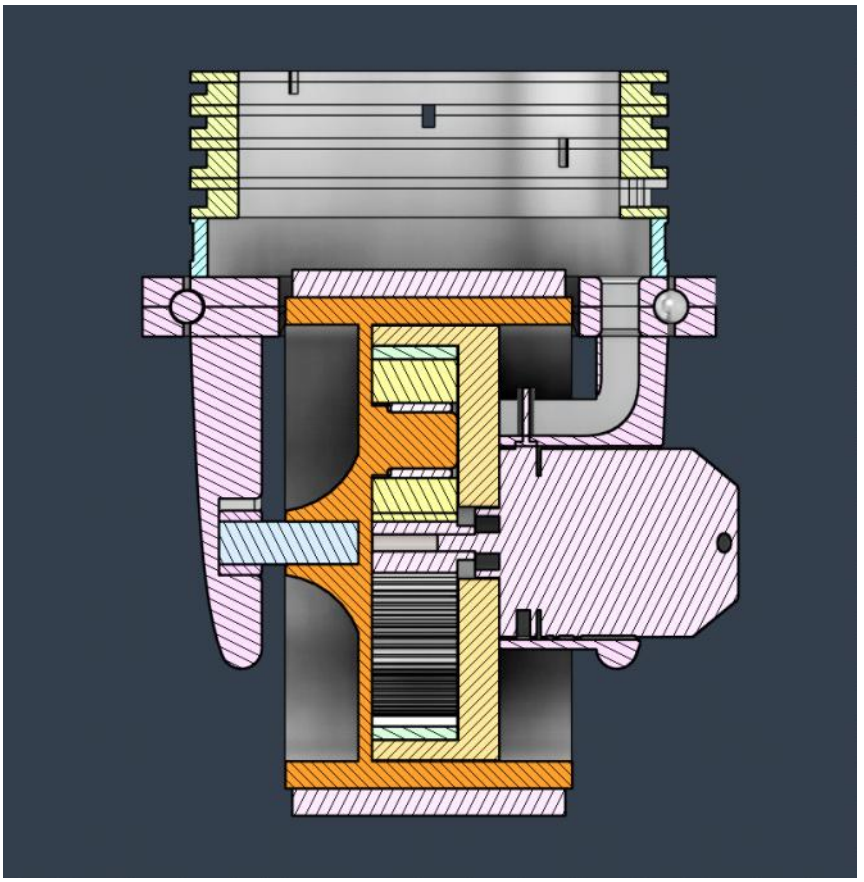
- ~ 5 hours
- Designed wheel and planetary gearbox for swerve modules

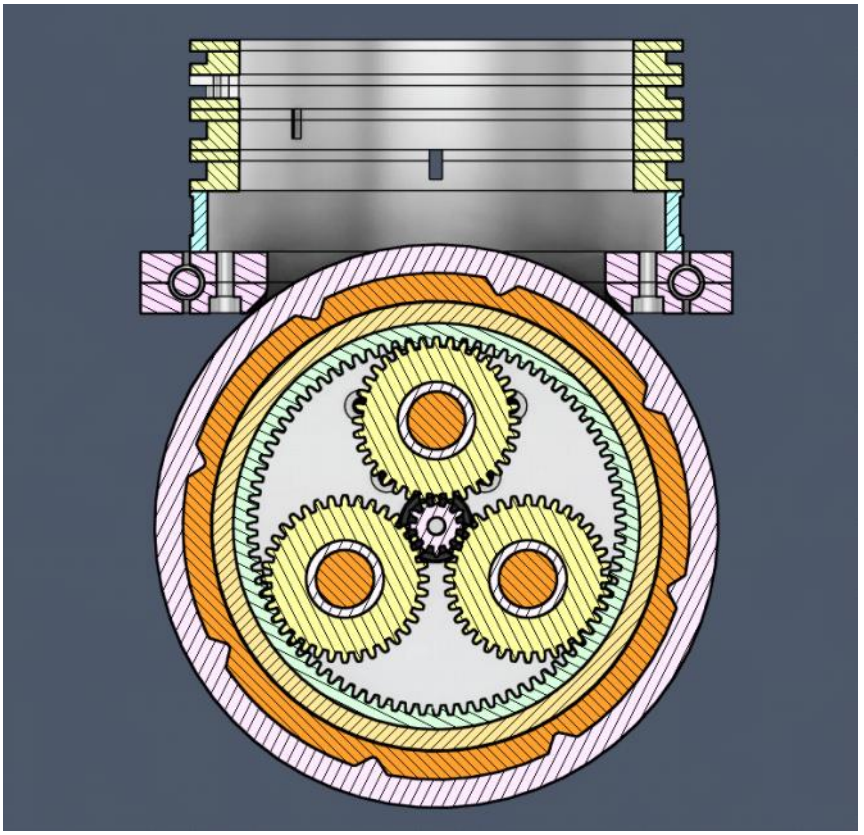


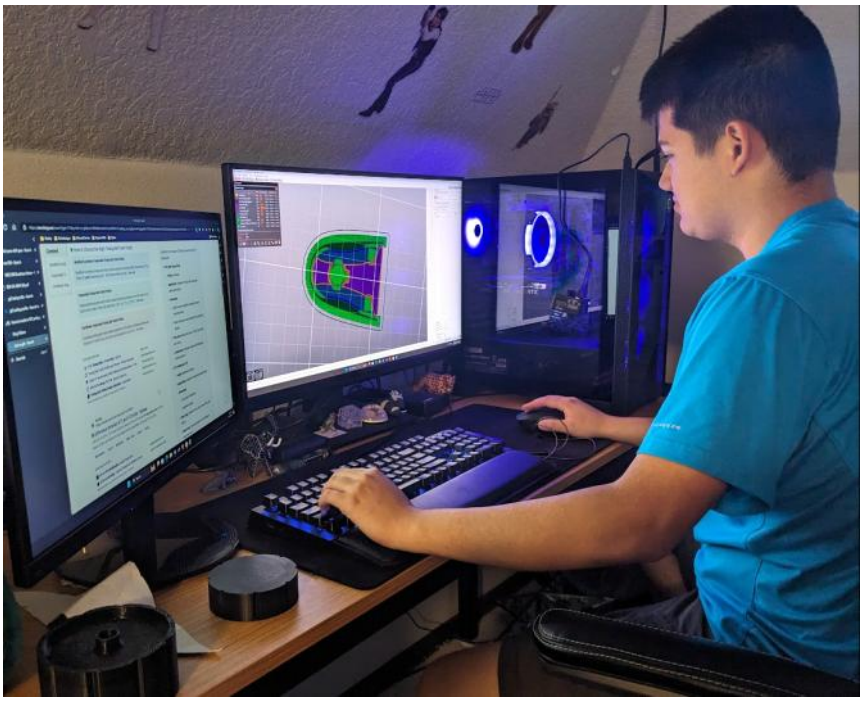
6/24

- ~ 7 hours
- Designed side supports, bearings, and sliprings, along with many minor alterations



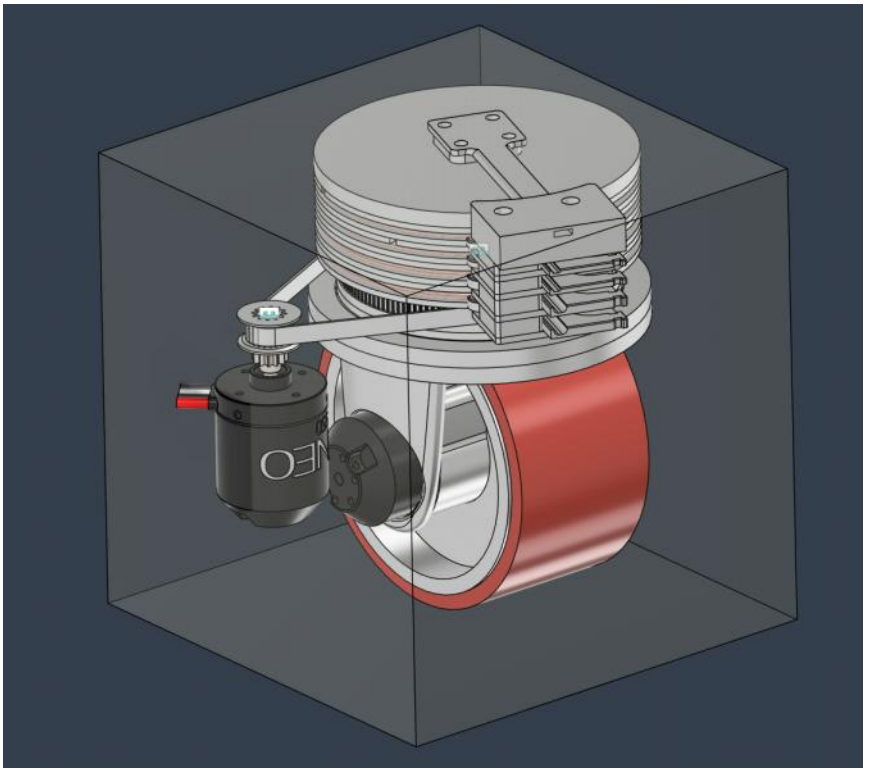


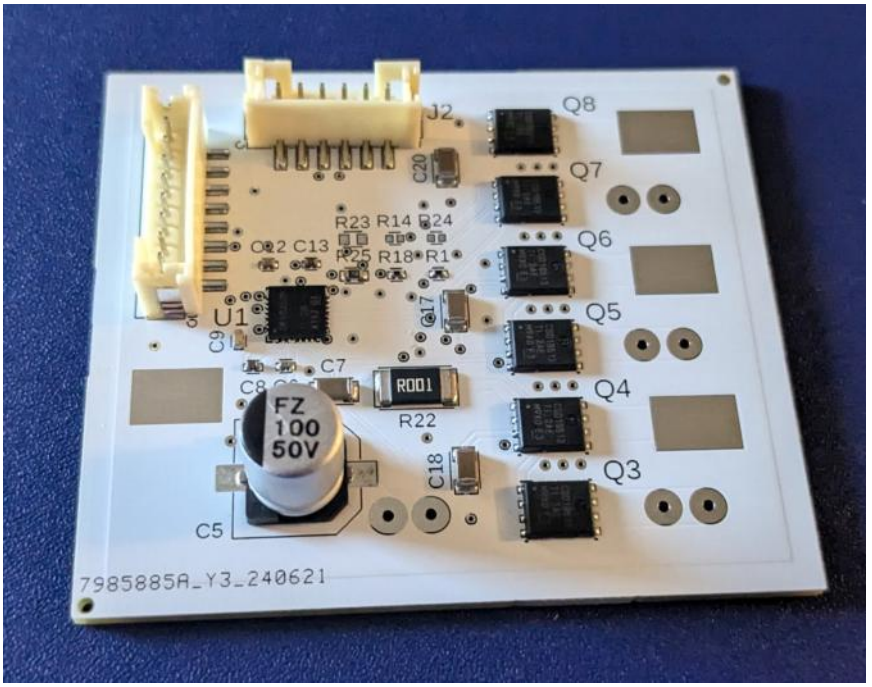




6/25

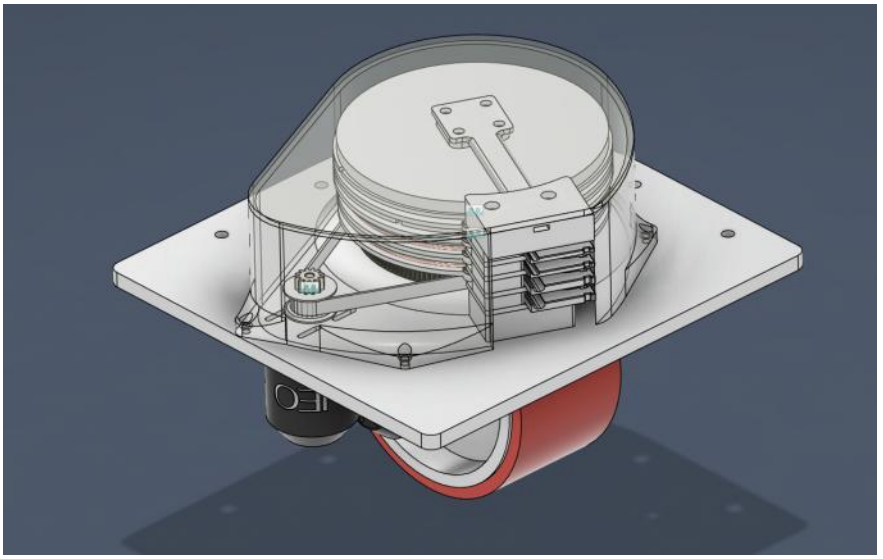
- ~ 7 hours
- Added detail to sliprings and created the slipring brushes
- Created GT2 belt assembly
- The motor controller board arrived





6/26

- ~ 4 hours
- Built the swerve platform and cover
- Made adjustments to the GT2 pulley and slipring tolerances



6/27

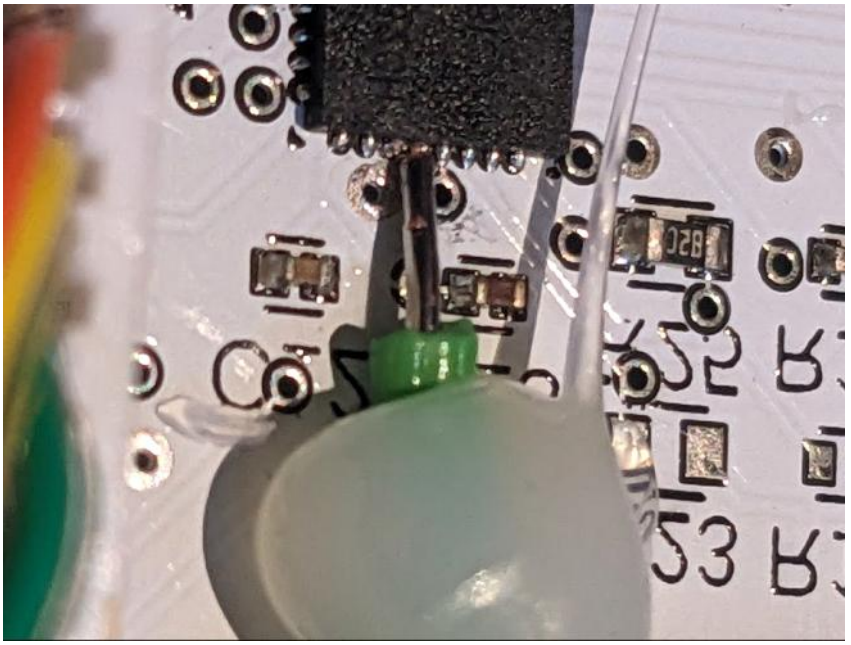
- ~ 4 hours
- Manually crimping JST connectors with needle nose pliers to test motor controllers
- Constructed the sliprings
- Wrote Arduino code for motor controller testing
- Created the corner of the frame

6/28

- ~ 3 hours
- Made minute changes to the top slipring cap, the slipring brushes, and a few other changes to the swerve modules
- Began testing the motor controllers, unfortunately no luck

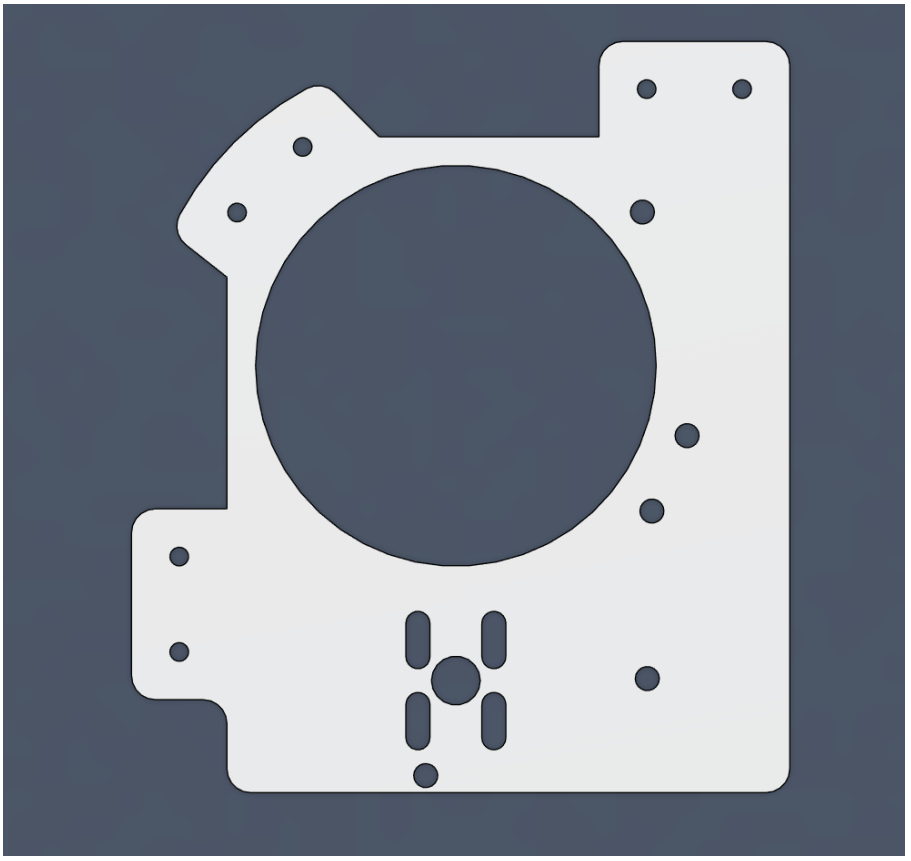
6/29

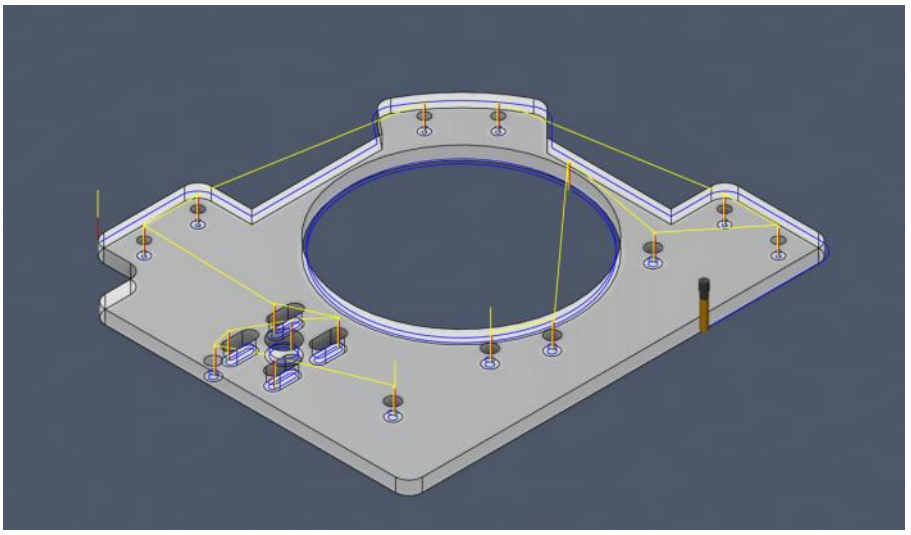
- ~ 3 hours
- Continued to debug the motor controller
    - After reviewing the schematic we found that the DVDD pin had been connected to the connector pin instead of ENABLE.
    - After spending a few hours we managed to connect a wire to the ENABLE pin and connected it back to the Arduino.
    - Unfortunately, the pin next to enable was ground so shorting the pins together was not an option; fortunately, the pin to its right is a NO CONNECT pin which we could short to. This allowed us to wedge the wire between the two pins.



6/30

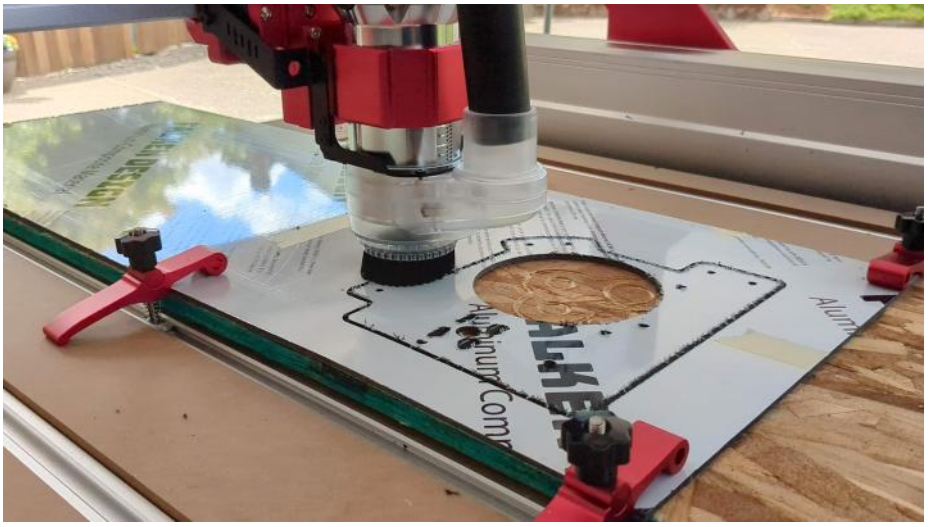
- ~ 5 hours
- Additional motor controller testing to figure out reliability of different wiring techniques (single vs multi-threaded wire, gauge, etc.)
- Designed the chassis bars and chassis connectors, and altered the swerve base plate to fit in via slots.
- 3D printing the swerve base plate was not successful, so we CAMed the plates so that we could cut them on the CNC router
    - Due to the router bit being larger than an M3 hole, we had to increase the size of those holes.

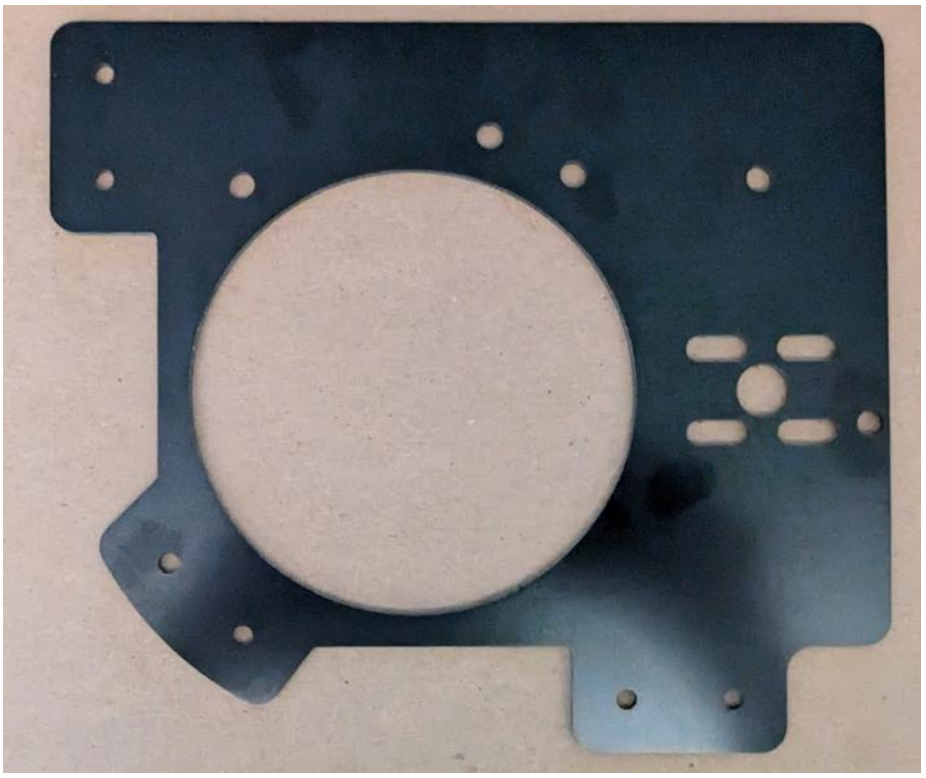




7/1

- ~ 4 hours
- Using the CNC router we cut the base plate for the swerve modules
- We began to assemble the swerve modules, making minor adjustments in CAD when we found issues with tolerances





7/2

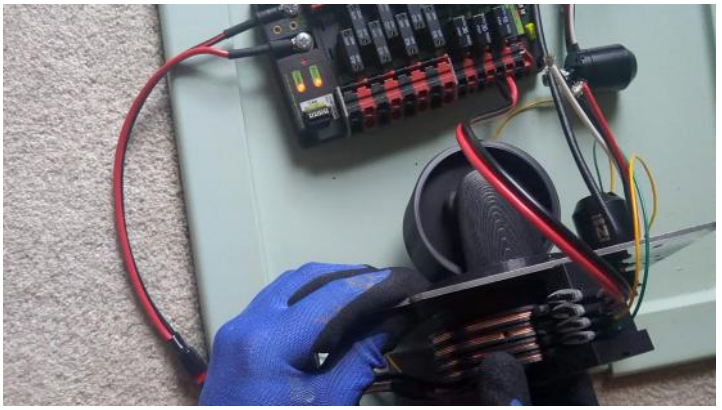
- ~ 1.5 hours
- Test fit the plate into the corner chassis and found a few issues with the tabs
- Fixed the tabs in CAD and modified the corner mount to have one robust support instead of two
  To Do
- Adjust power sliprings
- Fix bar inserts and corner

7/3

- ~ 4 hours
- Fixed the base plate tabs and re-routered it
- Fixed the chassis bars and corner sizing
- Other tolerance adjustments to the sliprings and side supports
- Began creating the arm in CAD

7/5

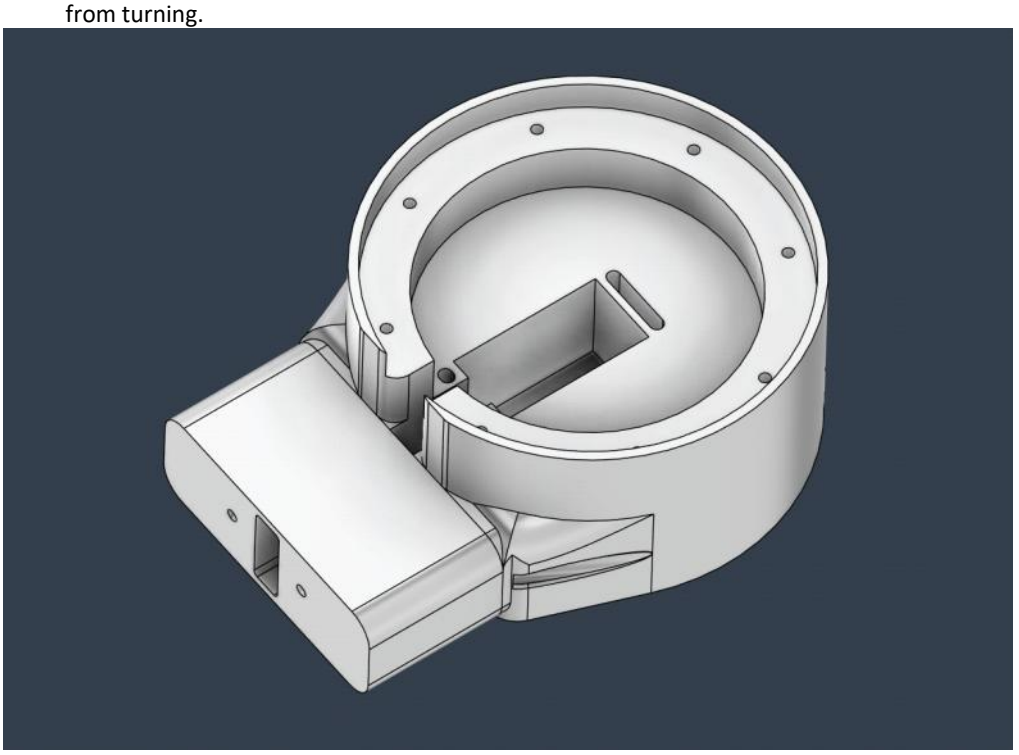
- ~ 5 hours
- Integrated SparkMAX into the slipring assembly. The sliprings were able to power the SparkMAX like normal however the sliprin gs would slip or get caught due to the uneven pull created by the spring on the one side
- Tested the swerve modules turning with the sliprings; it was successful however the belts slipped more often than we preferre d
- The variance created by the 3D printing created tolerance issues in our 3 x 1 bar inserts and the slipring assembly
- In order to find the correct interface for the bar inserts we tested small 3D printed frames to find the correct size that ac counted for the 3D printer tolerance
- Created a insert for the turning 550
- Created a Jig for the Dremel to fix the sliprings

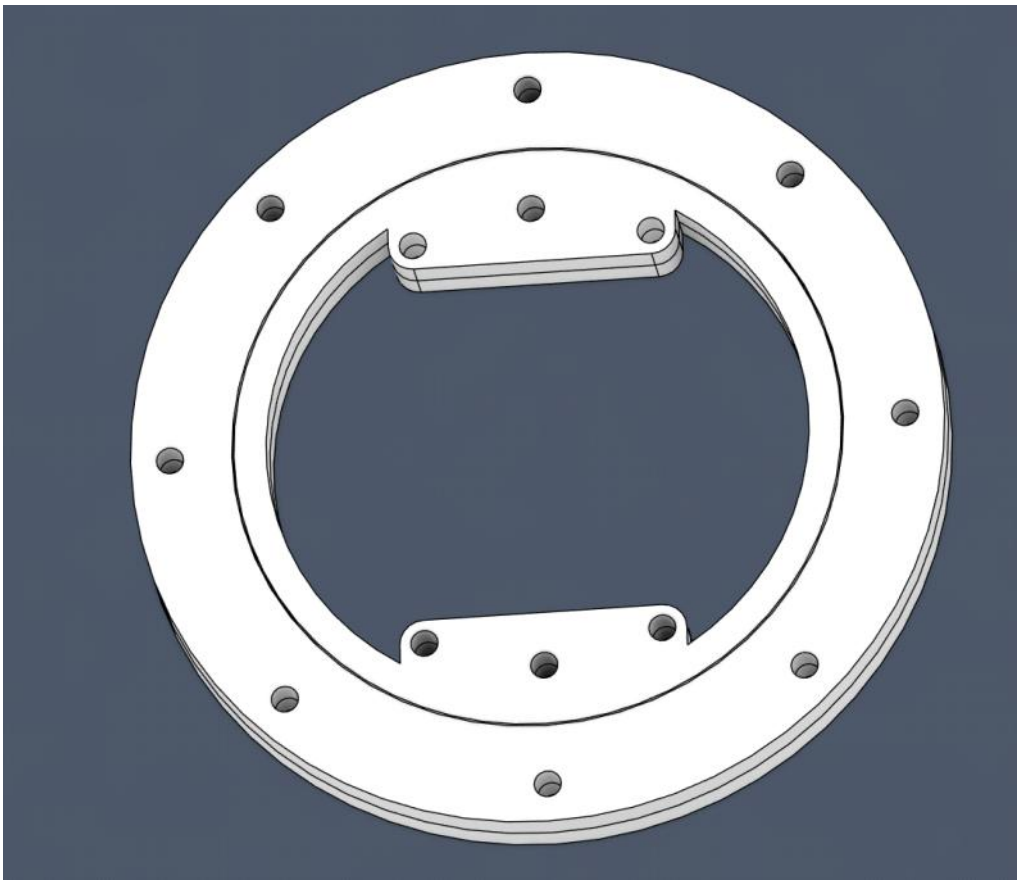


7/6

- ~ 7 hours
- Fixed the Dremel holder L-bracket hole by creating an insert that converted the 7mm to 3.5 mm hole
- Rebuilt CAN sliprings and sanded the inconsistencies out of the sliprings using the Dremel holder
- Modified the brushes and supports to have tension bands on both sides. Having it on one side was created inconsistent pressur e issues when going one direction over the other
- Transferred all the wires to the new slipring brushes
- Decided to use Debian Linux on the OrangePi, got VimbaX API working in Anaconda
- Began creating the arm servo joint and bearing and selected the servo
    - Instead of my previous intention of putting all the weight on the servo we decided to added a bearing that would hold the

weight for the servo. If all the weight was put on the servo the axle may have broken and the weight may prevent the servo from turning.





7/7

- ~ 7 hours
- Adjusted corner chassis pieces and re-enforced the connecting supports
- Created pin assignments for the Orange Pi and LCD screen
- Remote connected to the Orange Pi via NoMachine

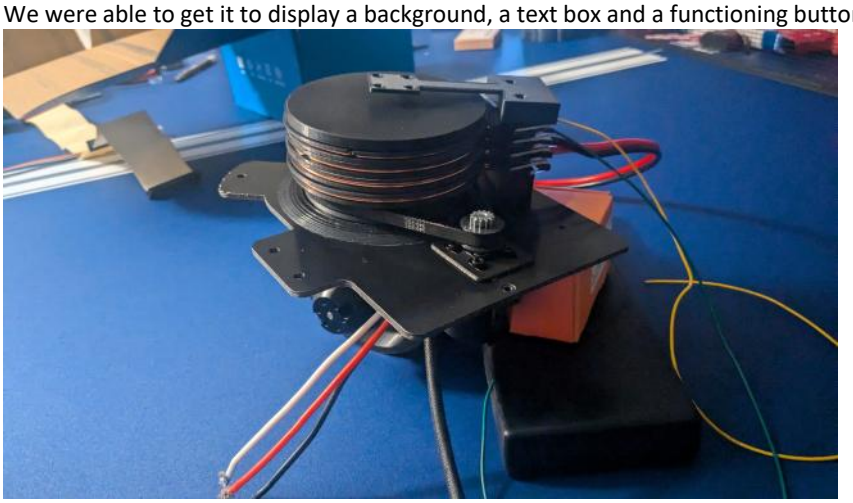
- Worked on debugging Python libraries for the Hosyand LCD TFT with ili9341 chip
    - Had a lot of problems with the spidev library. Turns out that the /dev/spi* permissions have to be manually set
      sudo chmod u+rw /dev/spidev0.0
    - Referenced: GitHub - sonocotta/ili9341-orangepi-python: Python library to control an ILI9341 TFT LCD display on the Orange Pi SBC
- To Do
    - Figure out why GPIO cannot be set
        - Possibly a timeout issue?

7/8

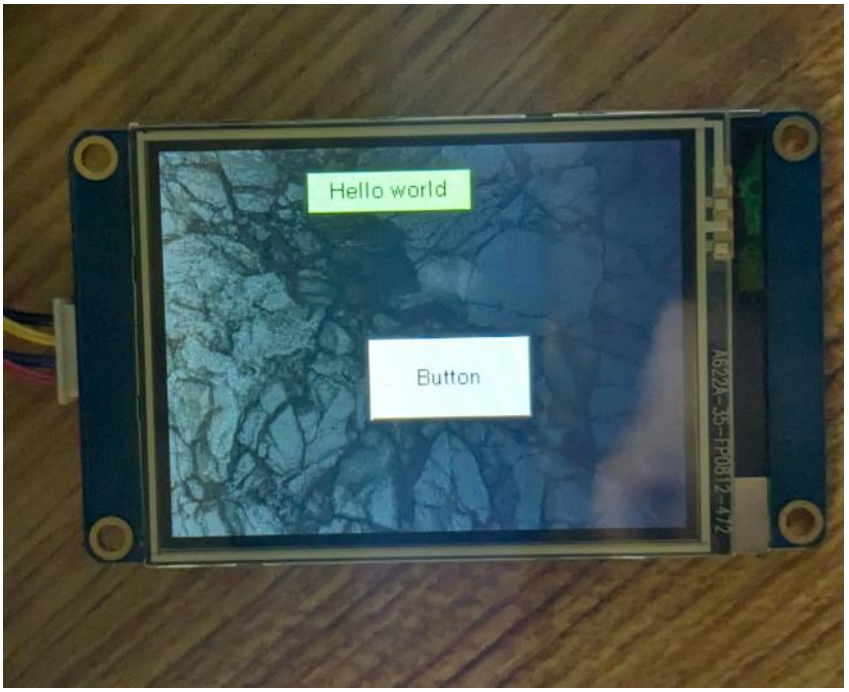
- ~ 8 hours
- Focused on debugging the LCD screen
    - Used numerous libraries on the OrangePi
    - Spent a lot of time attempting to circumvent the gpio/direction permissions
    - Found an problem where enabling both chip selects on the Spi0 caused an issues where none of the Spi bus would function (pins did not toggle)
    - After fixing the chip select we were finally able to send and receive using the spi. Previously the oscilloscope showed nothing on the SCLK, CS and the MISO and MOSI pins were not functioning as normal. After the fix everything was working as expected; however the screen was not displaying anything we sent.
    - We found out that we had been using packages for another LCD screen - ili9341 which was similar to the 9488
    - After a while we moved to an Arduino Esp8266 and attempted to utilize the Arduino_GFX provided by moononournation: GitHub - moononournation/Arduino_GFX: Arduino GFX developing for various color displays and various data bus interfaces
    - Arduino_GFX : 31 Steps (with Pictures) - Instructables
    - Moononournation's Arduino_GFX looked promising; unfortunately, we could not get it to work with our setup
    - After using various libraries and other packages, and externally powering the LCD we found no success.
    - After searching for other solutions a library for the OrangePi created by Adafruit was found
    - We installed Armbian on the Pi instead of Debian and going to see if Adafruit's software can be used

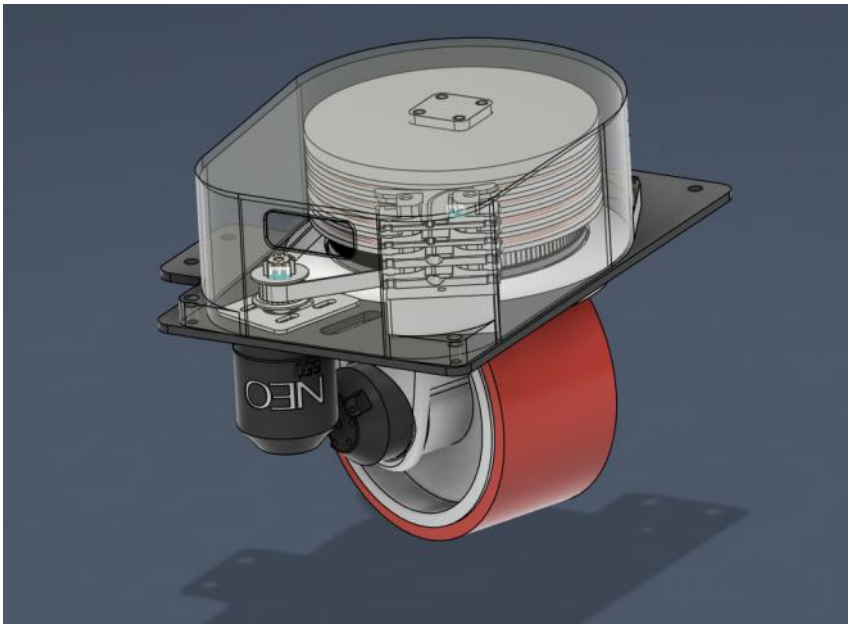
7/9

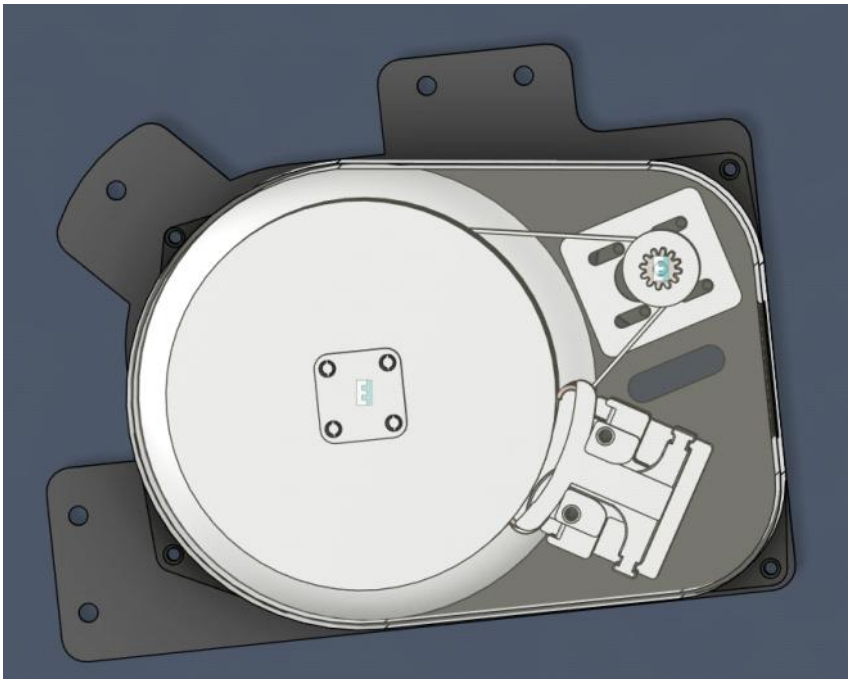
- ~ 6 hours
- Assembled the second revision of the swerve module
- Began working on a third revision of the swerve module
    - Fixed some tolerance issues in various places
    - Altered the brush stack so they are symmetric and cut the base so that it naturally will lean forward. The intention is that the forward force will counter act the force of the sliprings pushing the brushes backwards.
    - The placement of the 550 and the brush stack were changed so that the space was utilized more efficiently
    - The cover was redesigned to match the new footprint of the module. The mag encoder will now be a part of the cover instead of it being on its own mount connected to the brush stack.
- We decided to move away from the Ili9488 and instead use a Nextion LCD display which uses the UART protocol instead of SPI.
    - We were able to get it to display a background, a text box and a functioning button.



Swerve Module
Rev 2

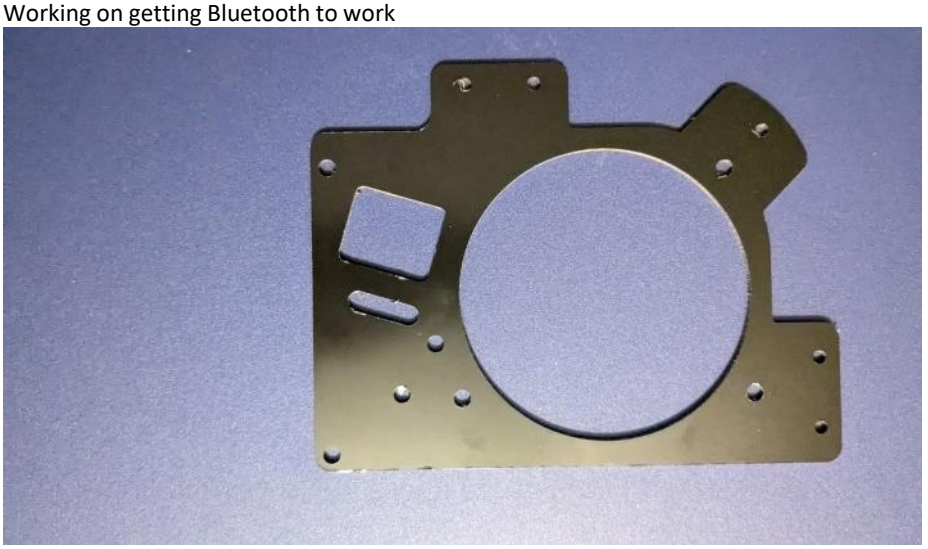

Hello world
Button


NEO



7/10

- ~ 6 hours
- Routered out the third revision of the swerve module plate
- Assembled the third revision of the swerve module and began creating the 4th version
    - Instead of gluing the bearing holder onto the plate we are going to add 6 screws
    - The bearing holder was made taller to accommodate the screws
    - The cover had to be altered so that there are gaps for the new bearing holder
- Found a candidate python library to work with the Nextion screen
- Working on getting Bluetooth to work



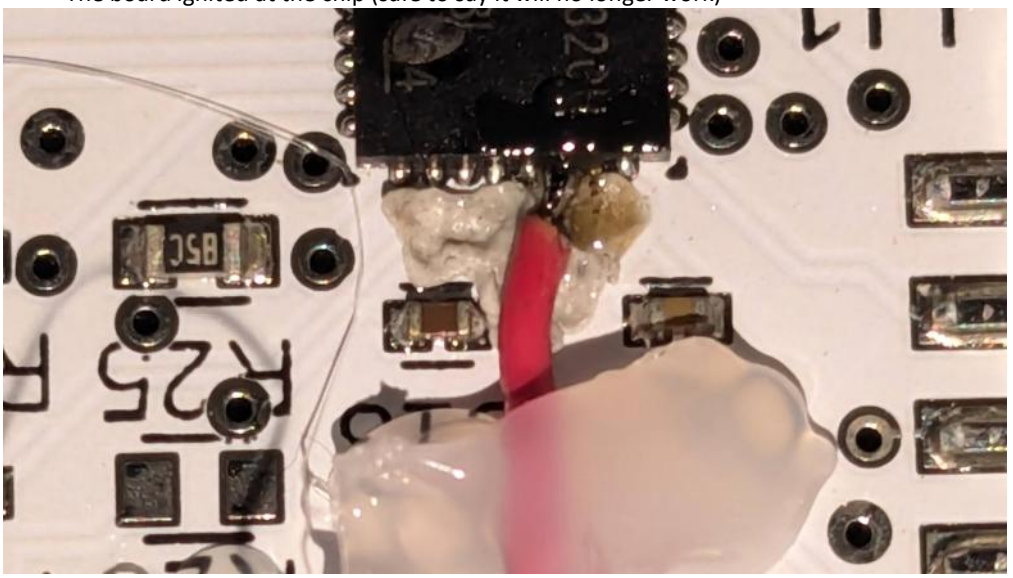
Swerve Module
Rev 3

7/11

- ~ 4 hours
- Working getting a PS4 controller to work with the Orange Pi
- Working on trying to set up another of the motor controller boards to do some more testing with

7/12

- ~ 5 hours
- Working on getting the PS4 controller to work with Arduino
    - Had success once, but we were unable to replicate out success until later
    - We found out that the Esp32 had to be scrubbed with the removed paired devices file in the git hub link in order for the controller to connect
    - Referenced:
        - GitHub - un0038998/PS4Controller_ESP32: This repository contains code and diagram for using PS4Controller with esp32
        - Video: https://www.youtube.com/watch?reload=9&v=dRysvxQfVDw
- Set up another motor controller board and tested it
    - The board ignited at the chip (safe to say it will no longer work)



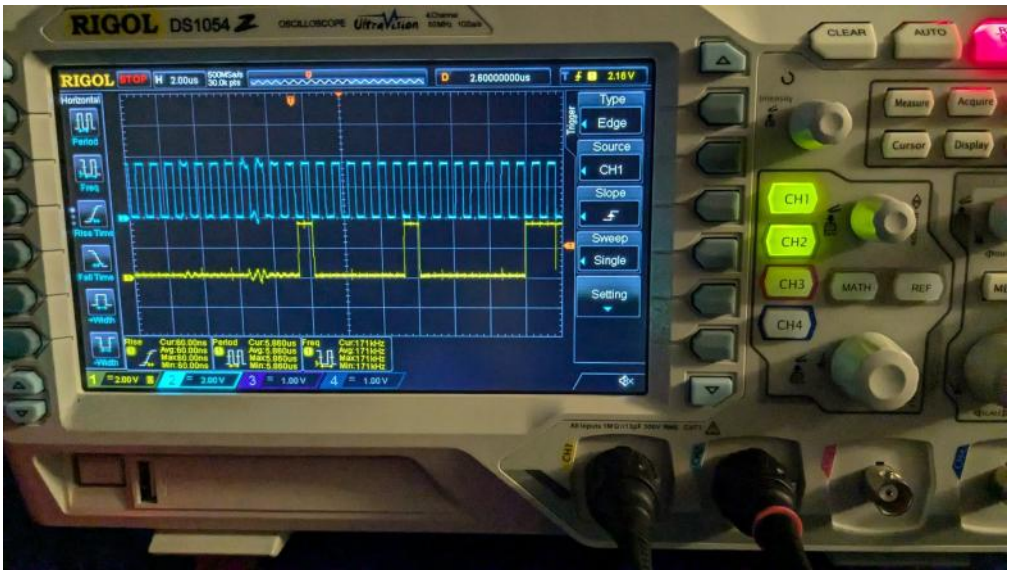
After math of testing

7/13

- ~ 5 hours
- Added cross braces to the chassis corners so they do not fold in when put into the bars
- Got the Orange Pi and the Esp32 to communicate with each other via SPI with: GitHub - hideakitai/ESP32SPISlave: SPI Slave library

for ESP32

- Initially, the Esp32 kept sending 0, 1, 2, 3 later we found that the tx_buf was being initialized later and was overwriting the our data
- Afterwards messages were received and sent both ways. The messages sent from the Orange Pi have some consistency errors that need to be looked into



Esp 32 sending:
0, 1, 2, 3

Esp 32 after fixing:
Ca, fe, ba, be

7/14

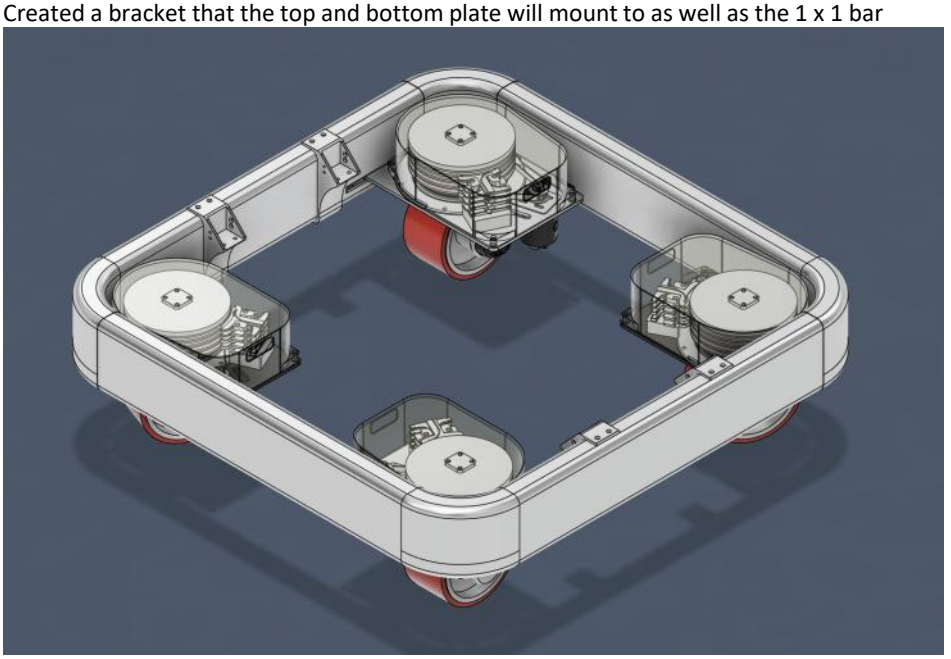
- ~ 3 hours
- Got the Esp32 to take inputs from the ps4 controller and send them over to the Orange through SPI reliably
  - Have to add 128 on the Arduino side because the SPI library expects a uint_8, but the ps4 function is returning a signed int
  - Once sent to the OrangePi 128 should be removed from the transmitted data
- Introduction · FRC Swerve Drive Programming (gitbooks.io)

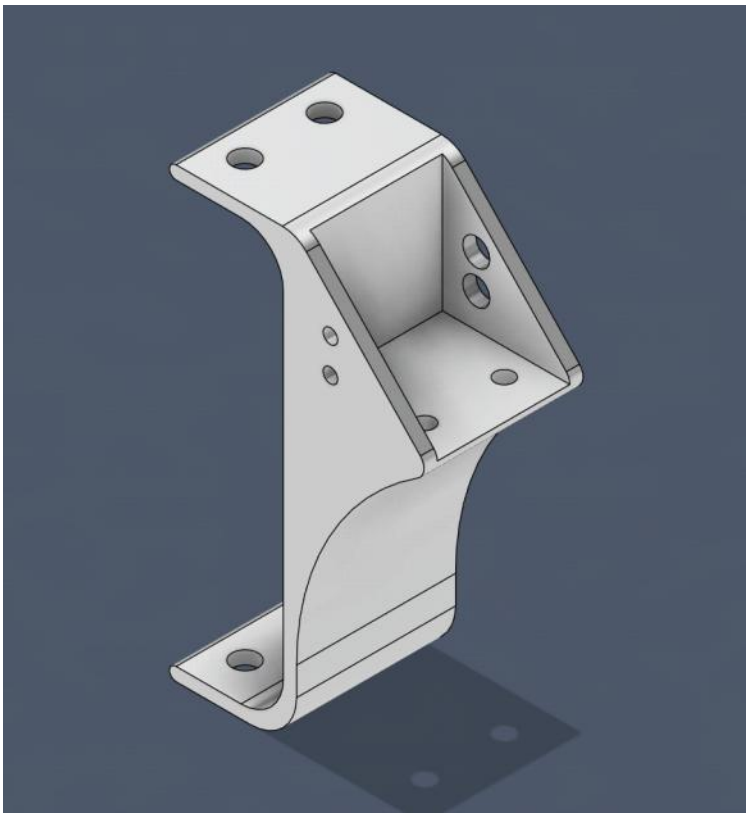
7/15

- ~ 1 hour
- Began looking at the motor controller board design and stated making some changes
  - Increased the size of parts to 0805 if they were smaller
  - Pulled the Mosfets together and closer to the DRV8320
  - Decreased the overall package size

7/16

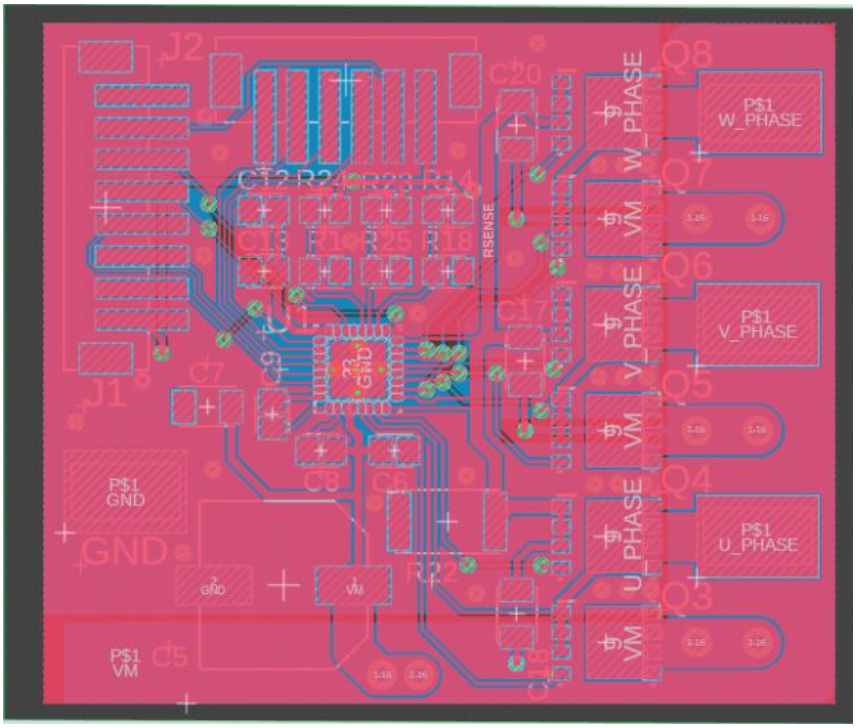
- ~ 3 hours
- Assembled the chassis in CAD
- Created a mirror version of the case and plate of the swerve modules
- Created a bracket that the top and bottom plate will mount to as well as the 1 x 1 bar





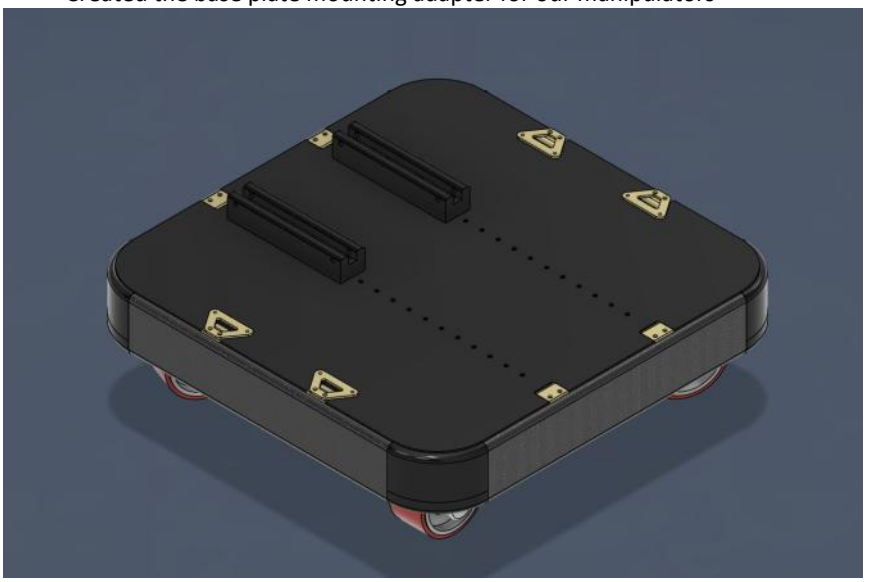
7/17

- ~ 5 hours
- Created the Skid plate in CAD and modified the 1 x 1 brackets to attach to the top
    - Looking potentially into using gussets to attach the top plate to the brackets
- Continued board development and finished creating a second revision



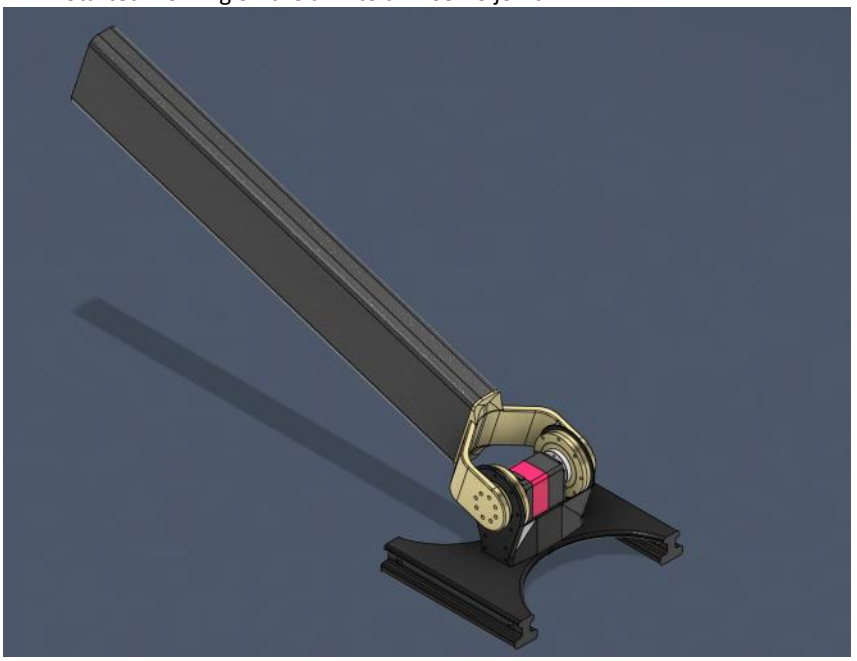
7/18

- ~ 4 hours
- Worked on chassis and top plate mounting
    - Decided to design it so there is a mounting gusset that is put onto the base plate and then that is screwed into the chassis
    - Created the base plate mounting adapter for our manipulators



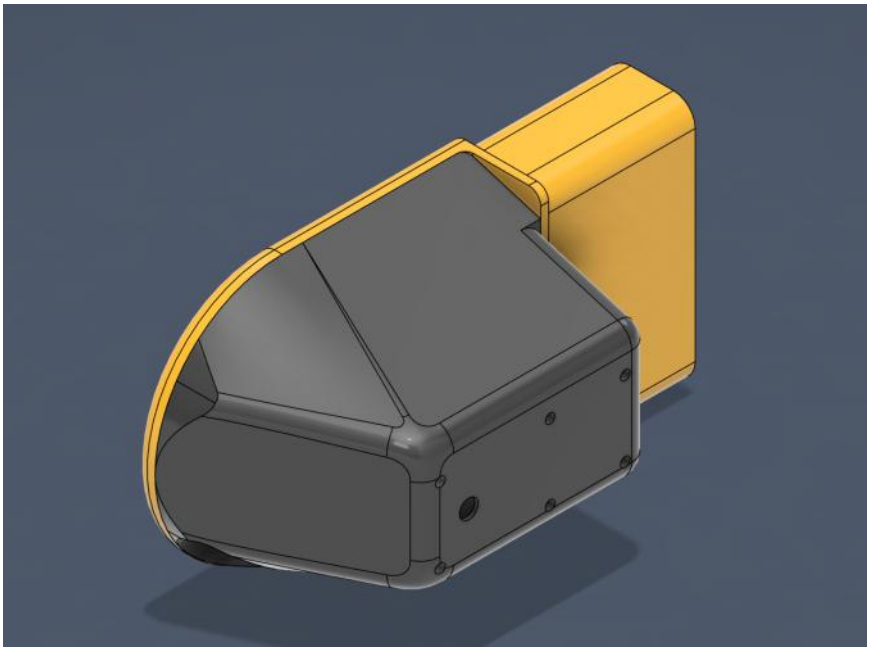
7/19

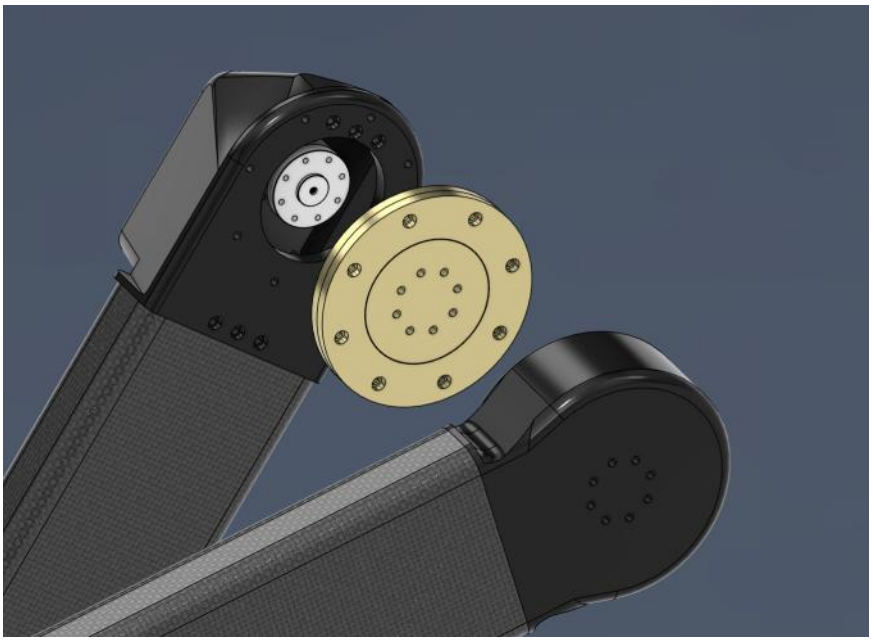
- ~ 6 hour
- Worked on the arm in CAD
    - Created the arm to base mount, the base of the arm, the base bearing, and the arm base to arm bar
    - Started working on the arm to arm servo joint

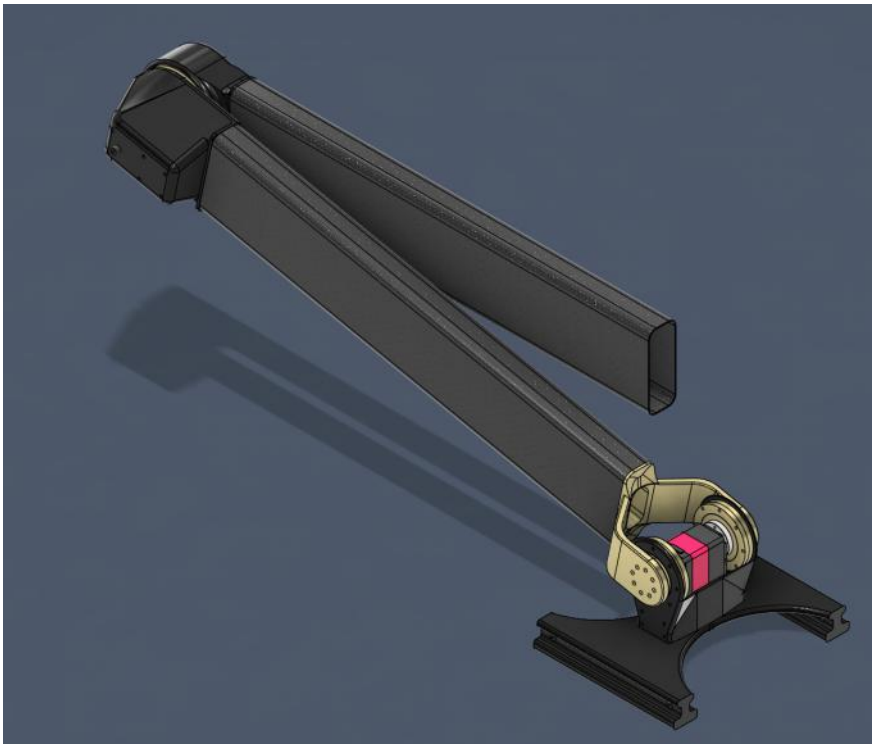


7/20

- ~ 5 hours
- Programming
    - Got Debian onto the Nvme drive, used the Orange Pi 5 user guide and Orange Pi 5 NVMe/SATA SSD Boot Guide - James A. Chambers (jamesachambers.com)
    - Applied all known updates to the OS
    - Tested the Esp32 PS4 connection with the Pi
    - Created a Git repository for the code
    - Started a skeleton of the code
    - Plan to use multiprocessing: https://www.datacamp.com/tutorial/python-multiprocessing-tutorial
- Design
    - Worked on the arm joints
    - Designed a 4 part joint to connect the two arms together

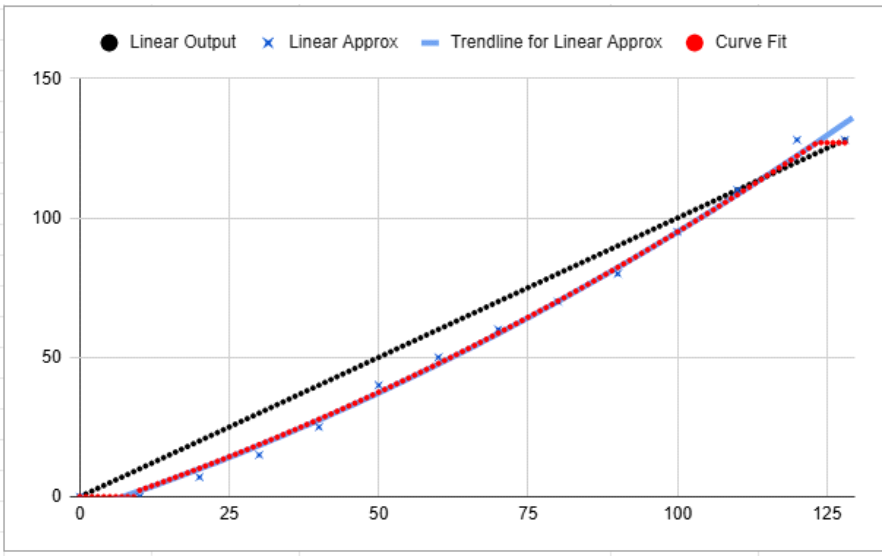






7/21

- ~ 7 hours
- Got the Git repository to work
- Moving from Arduino to VS Code
- Worked on creating a framework for the rest of the code
    - Created set up information that is sent to the ESP32 from the Pi - this includes response curves and error checking on the SPI Bus



Spi Communication		
byte0	0	0 = poly, 1 = sigmoid
byte1	5	constant value, negated by ESP32
byte2	7	x-coefficent, /10 by ESP32
byte3	3	x^2-coefficent, /1000 by ESP32

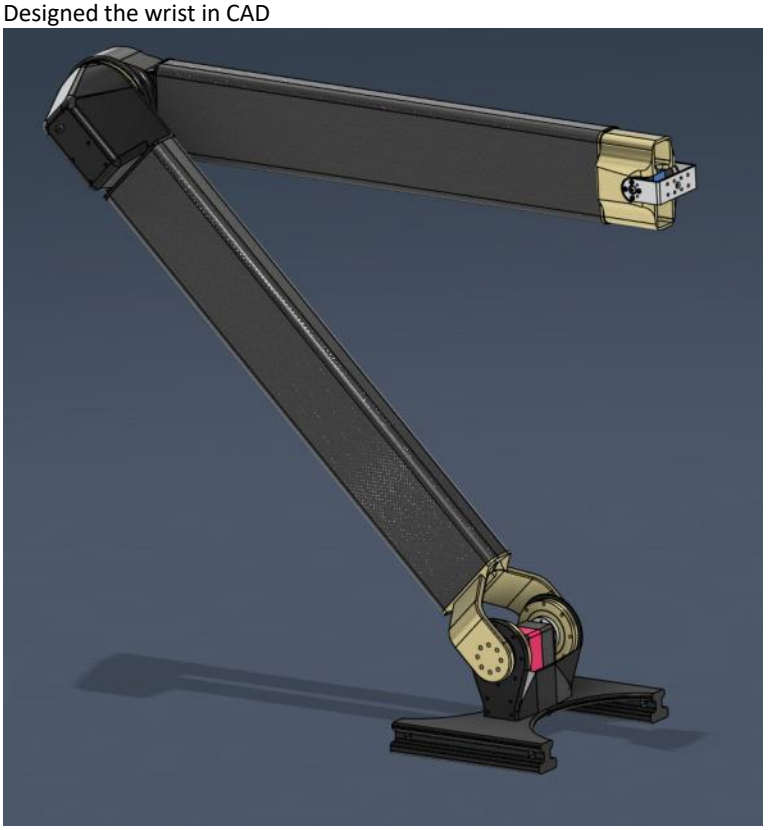
7/22

- ~ 5 hours
- Worked on getting the programming to work

- Had scope issues and linking errors

7/23
- ~ 2 hours
- Attempted to fix the linking errors

7/24
- ~ 1.5 hours
- Designed the wrist in CAD



7/25
- ~ 3 hours
- Finally resolved the issue of the ESP32 sending 0s on SPI: the tx buffer size needs to be a multiple of 8 bytes
- Fixed the curve coefficient communication handshake
- Implemented the response curve on the ESP32 (complies, not yet tested)
- Started a "Pi Startup" tab in the spreadsheet to describe the Orange Pi interaction with the LCD, ESP32, FPGA, Swerve, and Ar m

7/26
- ~ 2 hours
- Started writing the PWM spec document

7/27
- ~ 4 hours
- Completed the first revision of the PWM spec
- Installed and debugged the VS code toolchain for the Tang Nano using: Lushay Labs
- Wrote RTL for the PWM and wrote a test bench, and confirmed all the basic operations of the PWM are working as expected

7/28
- ~ 3 hours
- Wrote the first revision of the SPI spec

- Wrote RTL for the SPI and wrote a test bench, and confirmed all the basic operations of the SPI is working as expected

7/29
- ~ 6 hours
- Wrote the first revision of the Address Decoder spec
- Started FPGA Sub-system spec
- Wrote RTL for the Address Decoder
- Identified some updates that need to be applied to the PWM

7/30
- ~ 5 hours
- Created test bench for Address Decoder
- Started working on PWM Control and PID spec
- Started thinking about how to implement PID tuning and I2C into the system

7/31
- ~ 3 hours
- Started working on PWM Rotation Controller spec

8/1
- ~ 2 hours
- Wrote RTL for PWM Rotation Controller

8/2
- ~ 2.5 hours
- Finished writing RTL for PWM Rotation Controller and wrote test bench

8/3
- ~ 3 hours
- Started writing RTL for pwm_ctrl

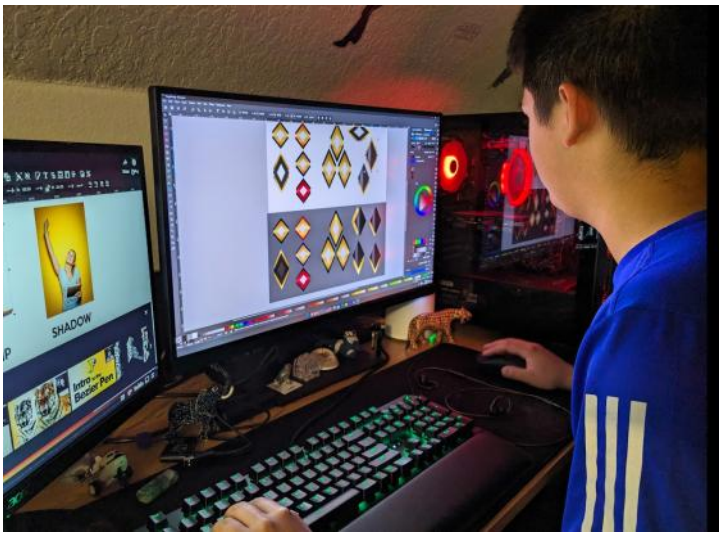
8/5
- ~ 5 hours
- Designed the first functional grabber for the robot
- Finished writing RTL for pwm_ctrl, and modified Lushay Labs' I2C code (preventing the D latches from being formed; and reformatting to our preference)
- Created the top level file and got the RTL code to build

8/6
- ~ 3 hours
- Began working on a new version of the grabber

8/7
- ~ 3 hours
- Finished new version of grabber and updated the arm lengths

8/9

- ~ 6 hours
- Worked on creating a logo



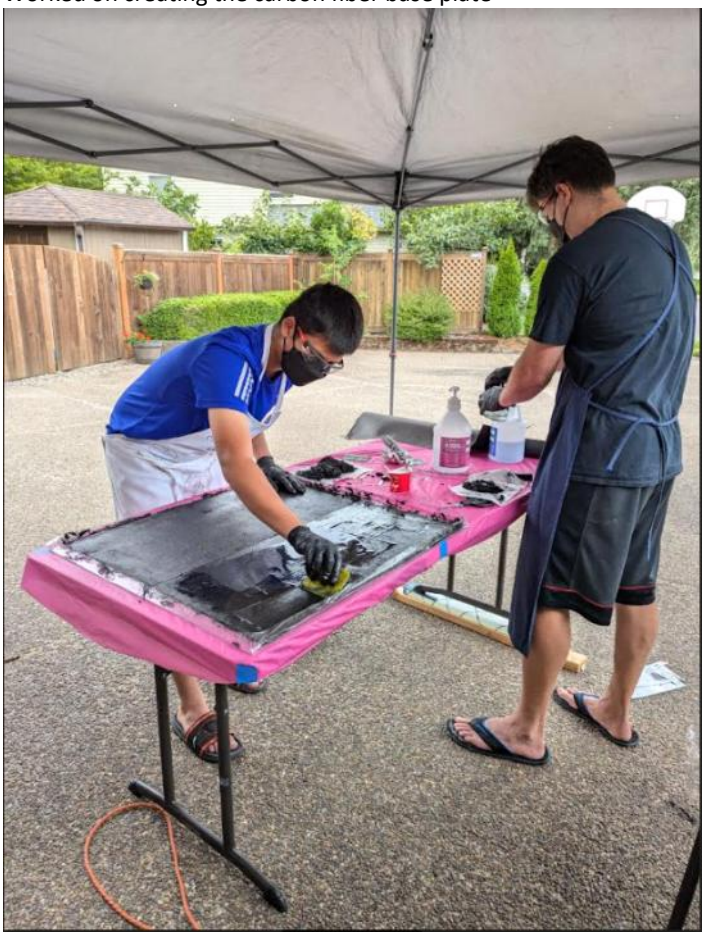
8/10

- ~ 4 hours
- Worked on animating logo and began ordering more supplies for the robot

8/11

- ~ 4 hours
- Finished animating logo with Da Vinci Resolve and began 3D printing a tire

8/12

- ~ 2 hours
- Worked on creating the carbon fiber base plate
- 

- 

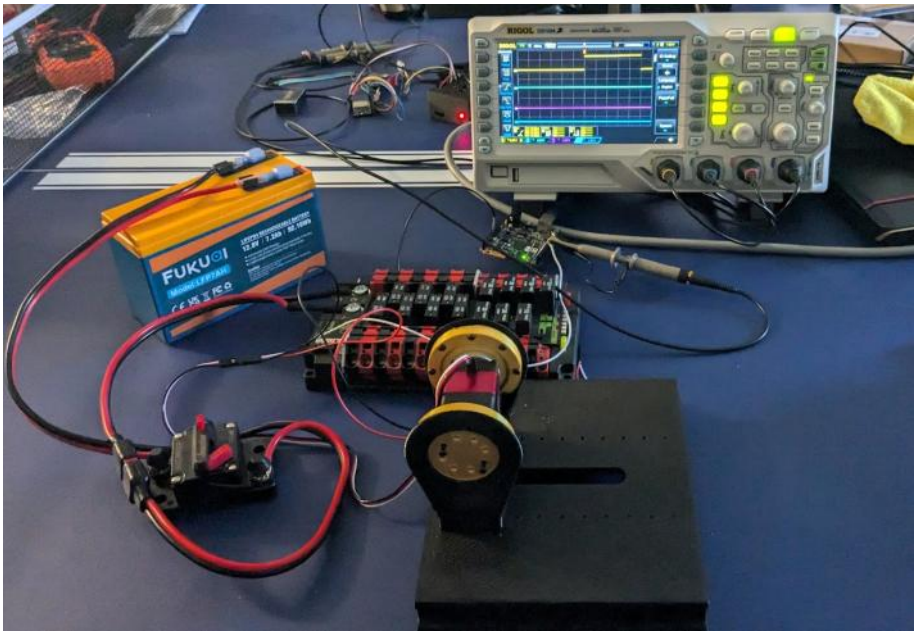
8/13
- ~ 2 hours
- Worked on the website

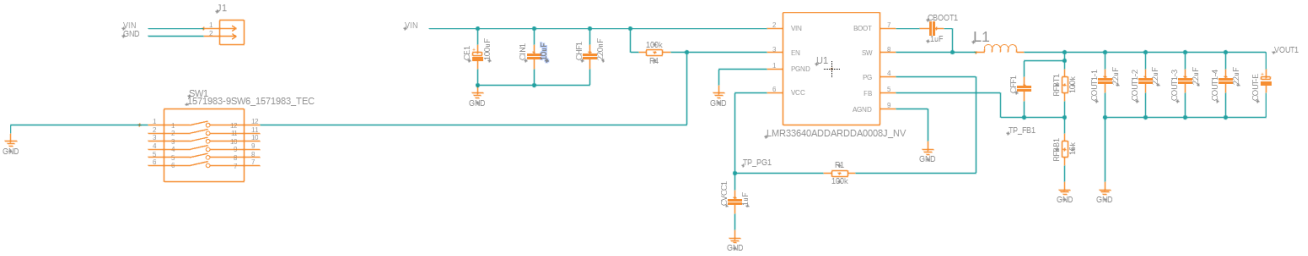
8/14
- ~ 3 hours
- Worked on the website and pushed the first revision to Github

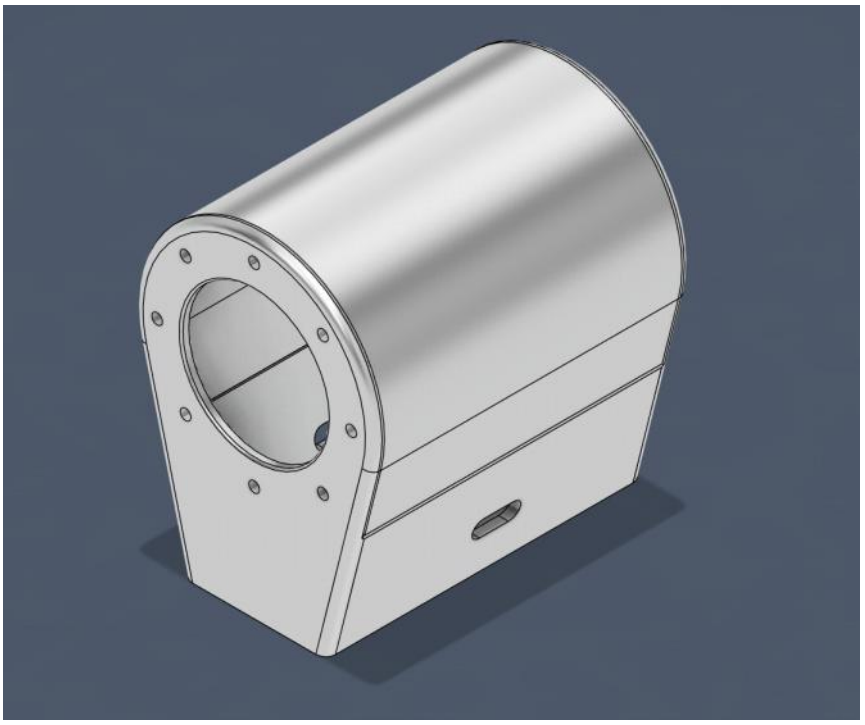
8/15
- ~ 3 hours
- Worked on website
- Started 3D printing and building arm bearings
- Started looking at the voltage regulators for the motherboard

8/16
- ~ 2 hours
- Worked on getting parts 3D printed

8/17
- ~ 5 hours
- Tested gold paint of the gold 3D printed parts
- Tested the 150kg PWM servo to make sure it works
- Redesigned base mount to make it more robust
- Worked on the motherboard and created the schematic for the first voltage regulator (12V)

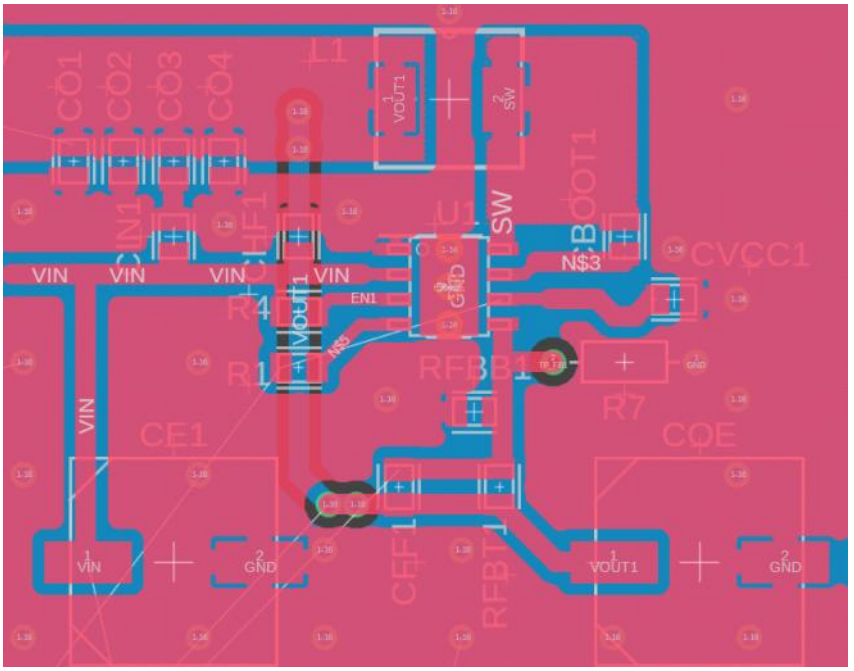


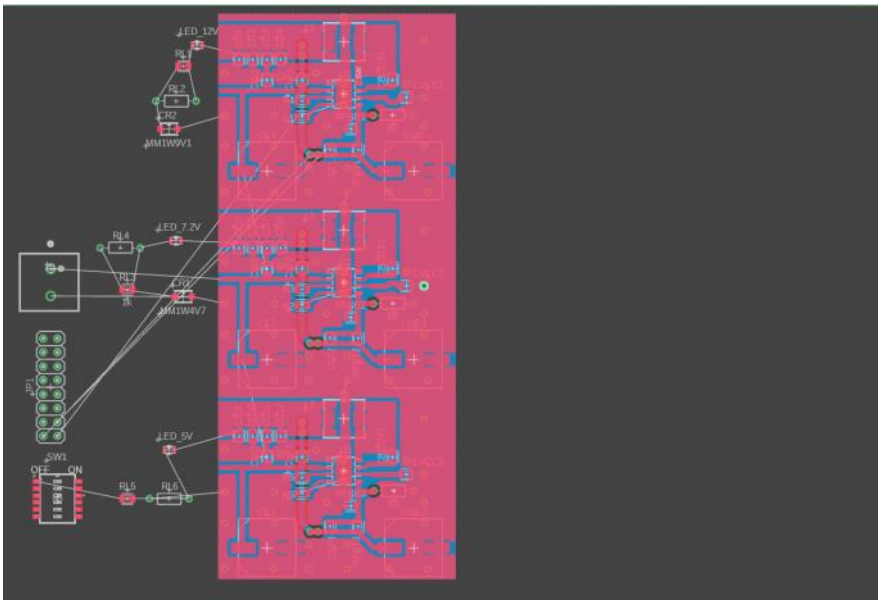




8/18

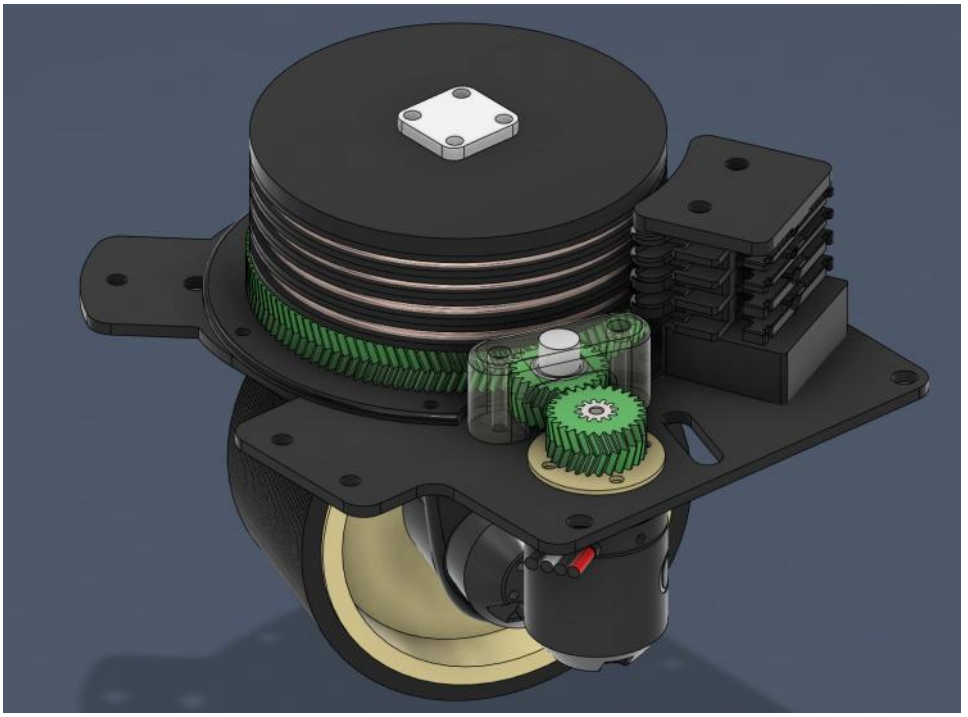
- ~ 4 hours
- Worked on motherboard
    - Finished creating each of the adjustable voltage regulators for 12V, 7.2V, and 5V

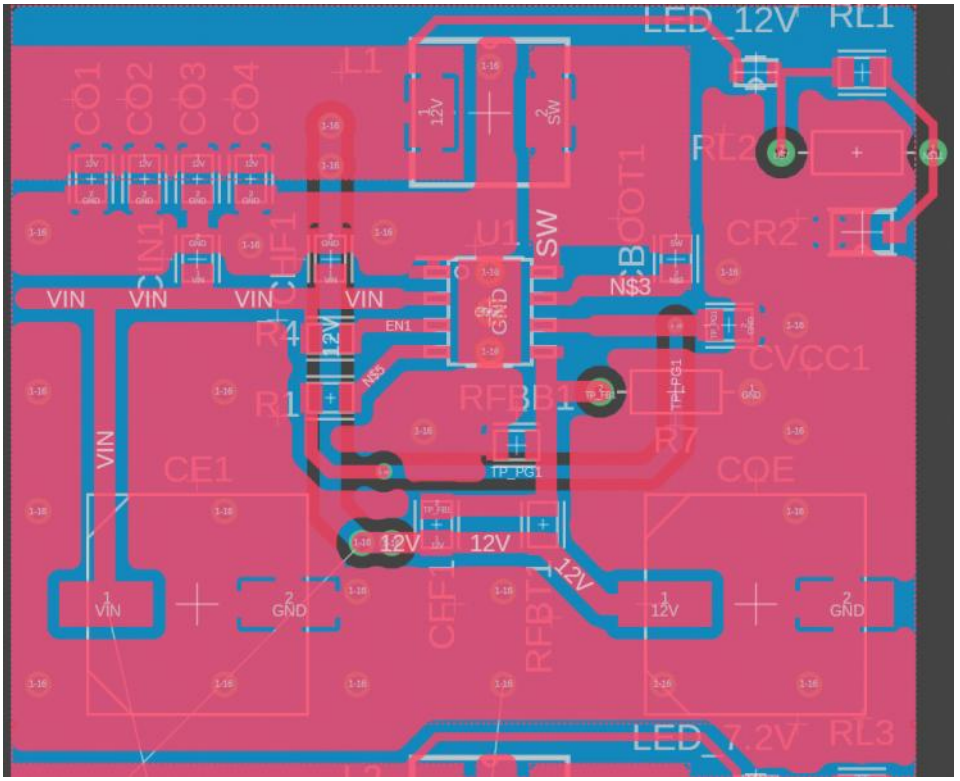




8/19

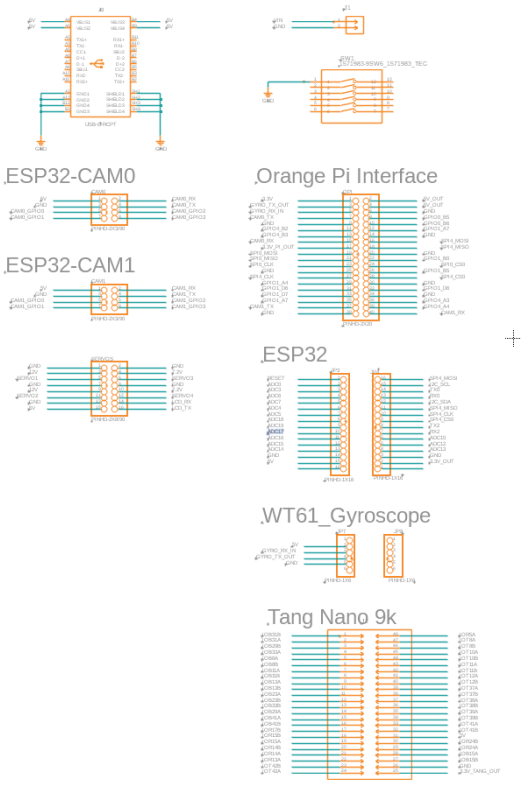
- ~ 5 hours
- Converted the rotation on the swerve modules from belts to gears
- Worked on the motherboard
    - Connected the LEDs and added the 40 pin header for the Orange Pi
    - We are thinking about what to separate from the motor feedback to the Orange Pi and the Tang Nano 9k
- Started to print the other 3 swerve modules

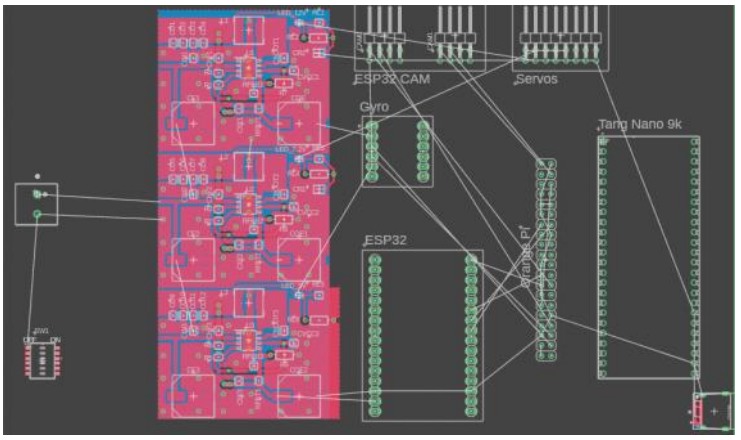




8/20

- ~ 6 hours
- Worked on the mother board design
    - Added the Orange Pi, ESP32, ESP32 Cams, and Tang Nano interfaces
- Worked on building the arm, and continued to manufacture parts for the other swerve modules





8/21

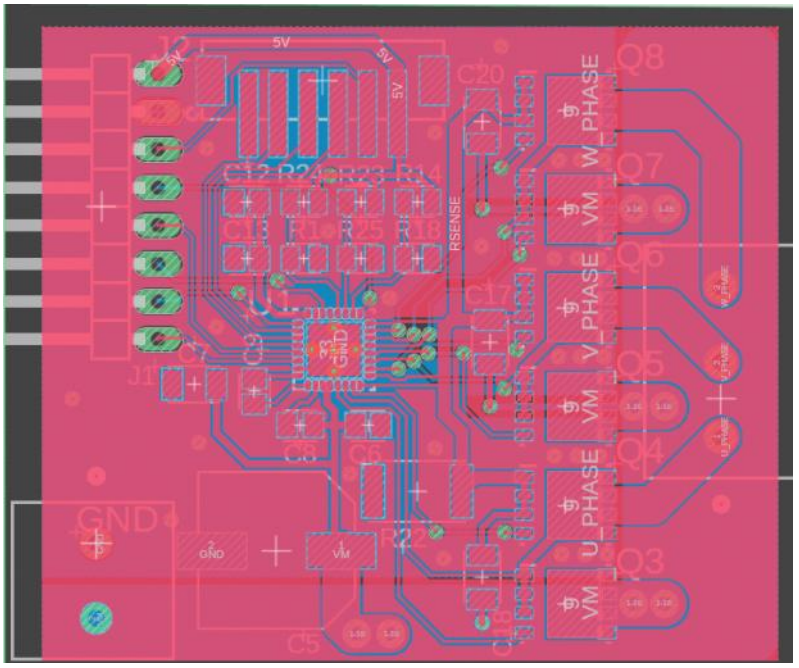
- ~ 2 hours
- Looked at board design
    - Looking at connector for the power wires to the boards

8/22

- ~ 3 hours
- Worked on getting the carbon fiber weave onto the fiberglass bars

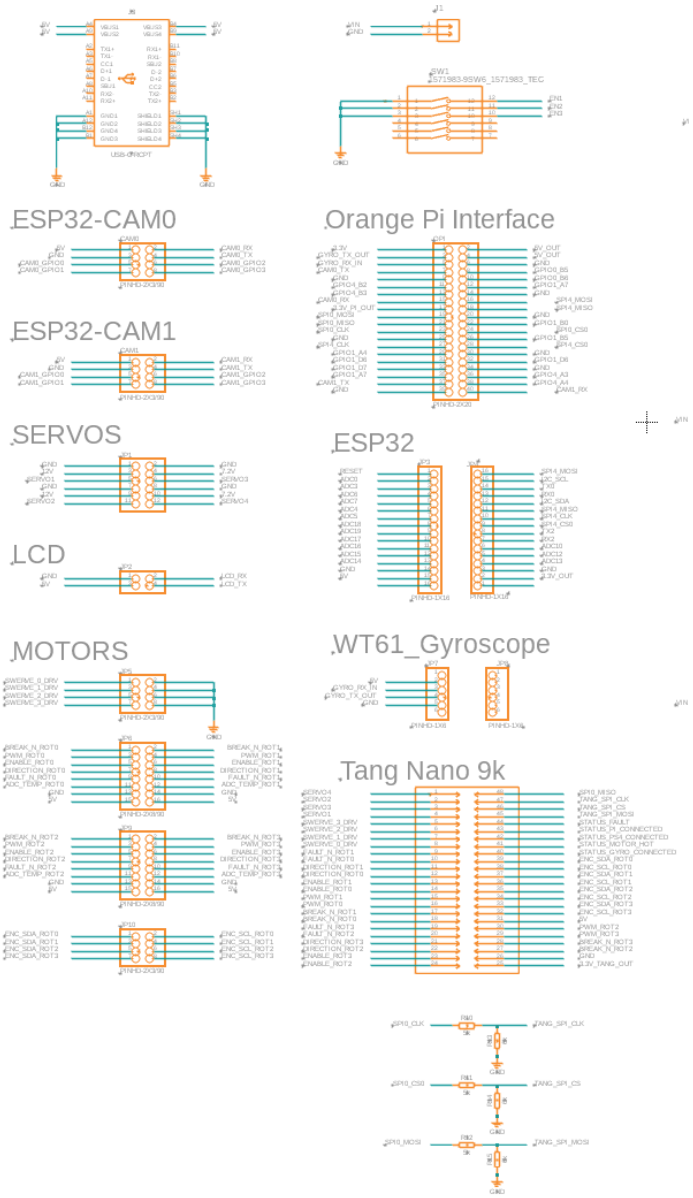
8/23

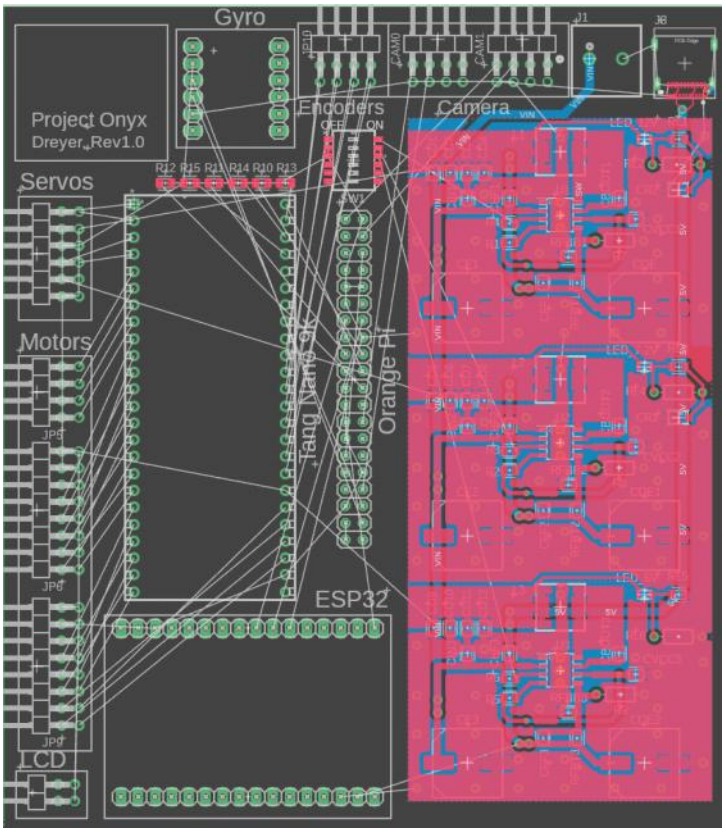
- ~ 2 hours
- Worked on DRV8320 to replace the connectors



8/24

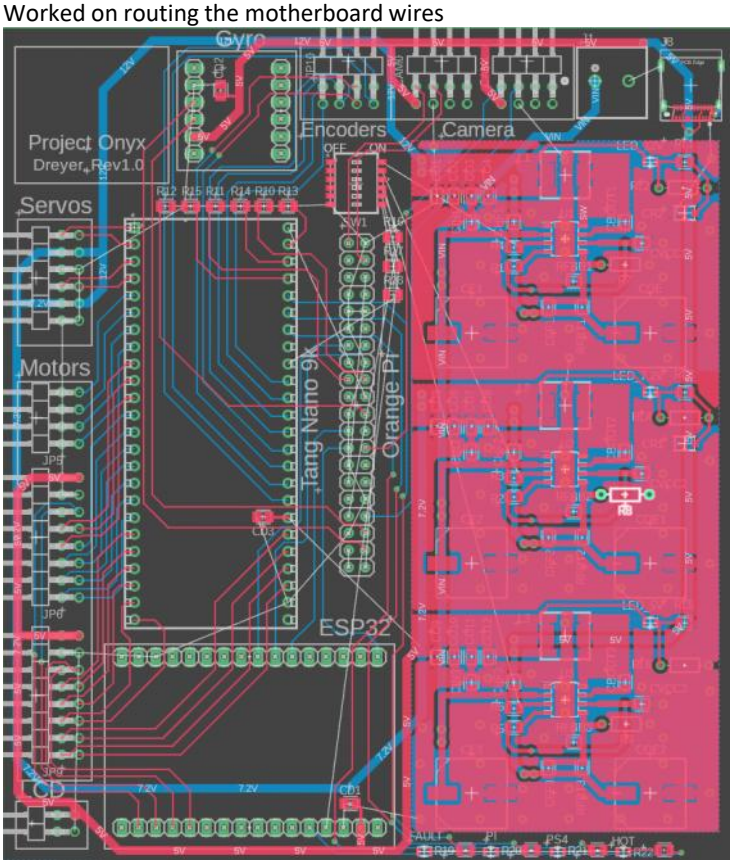
- ~ 4 hours
- Worked on the Motherboard
    - Added missing connectors and assigned all the Tang Nano 9k pins
    - Reoptimized the position of all the parts


ESP32-CAM0
Orange Pi Interface
ESP32-CAM1
SERVOS
ESP32
LCD
MOTORS
WT61_Gyroscope
Tang Nano 9k



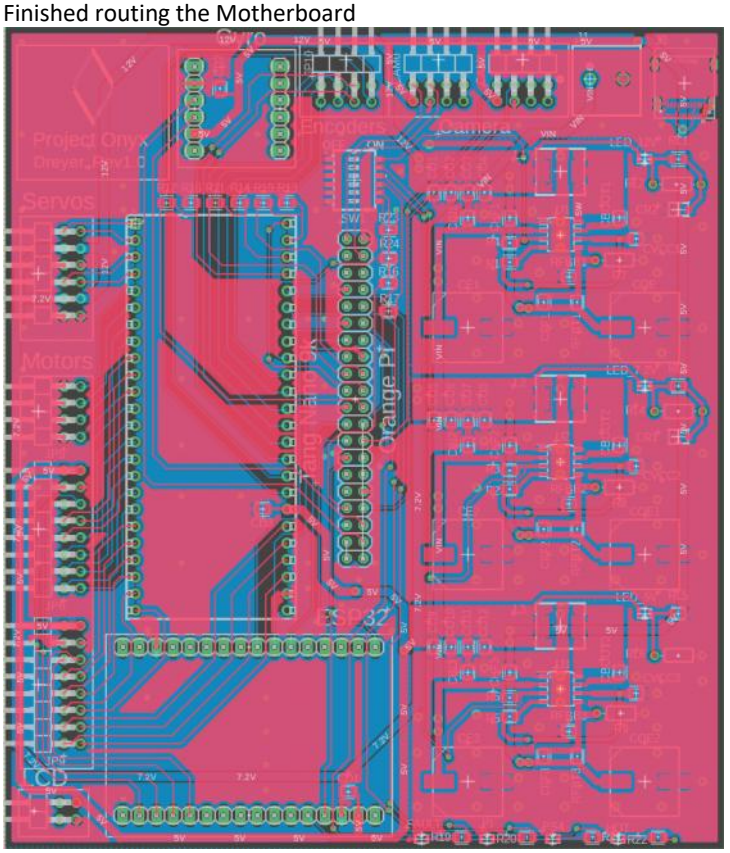
8/25

- ~ 4 hours
- Worked on routing the motherboard wires



8/26

- ~ 3 hours
- Finished routing the Motherboard

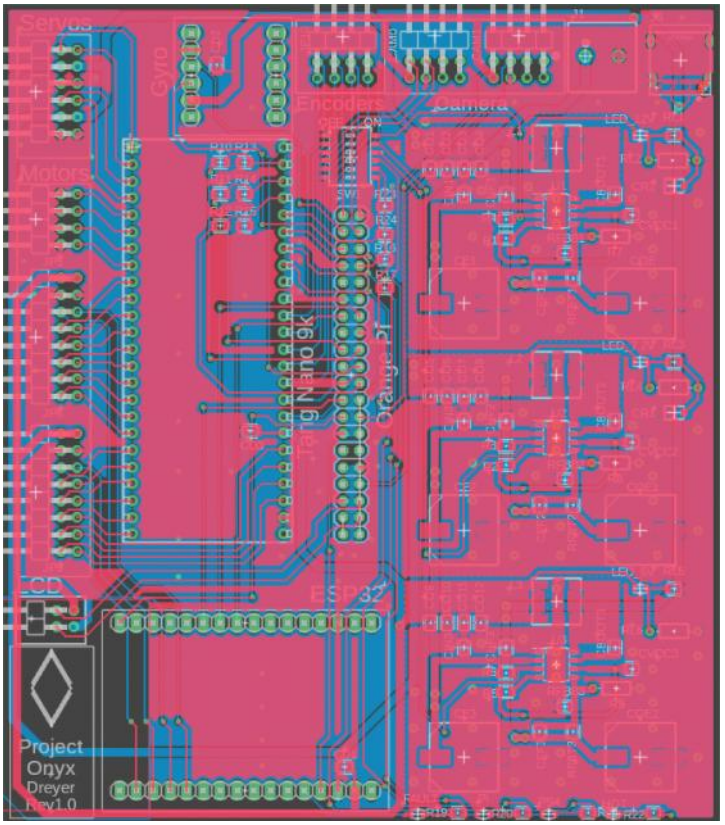


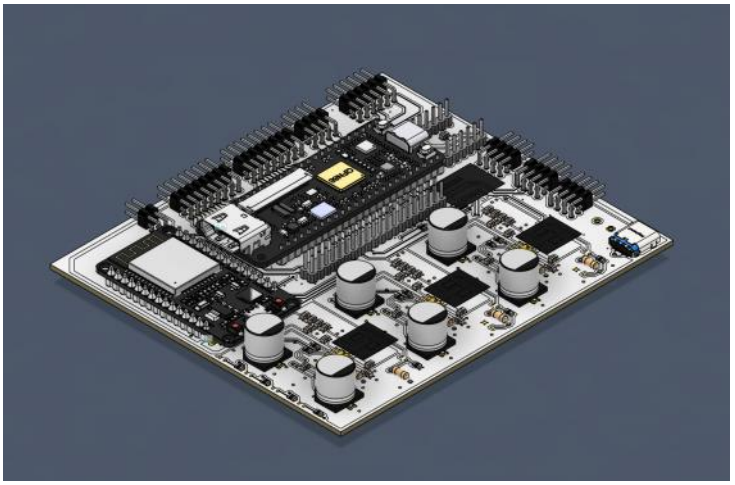
8/30

- ~ 2 hours
- Worked on refining the website by updating images and links
- Fixed the skid plate in CAD

8/31

- ~ 5 hours
- Sanded down the carbon fiber bars and cut them to length
    - Did some impact testing to see what structural failure looked like
- Fixed spacing between the Tang Nano and the ESP 32
    - Had to re-route the Tang Nano and moved the left connectors up to use our space more efficiently


Project
Onyx
Dreyer
Rev1.0



9/3
- ~ 3 hours
- Worked on finishing the carbon fiber bars and top plate
- Worked on the website and my resume

9/4
- ~ 1.5 hours
- Fixed Website video link
- Fixed some minor issues with the swerve modules

9/5
- ~ 2 hours
- Fixed some problems in CAD and assembled some of the rev 3 swerve module

9/7
- ~ 1.5 hours
- Attempting to fix low areas in carbon fiber by adding more epoxy
- Assembled sliprings

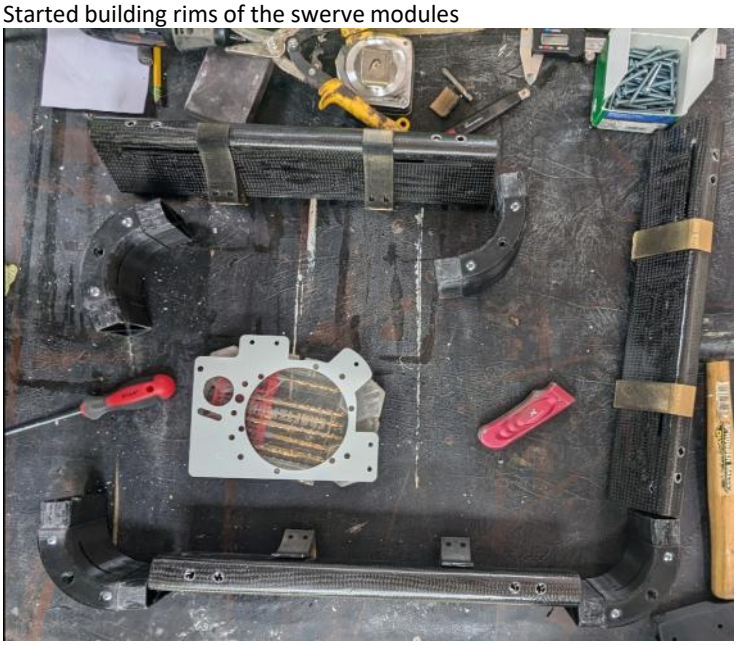
9/8
- ~ 4 hours
- Assembling sliprings and bearings
- Added clear coat to carbon fiber bars

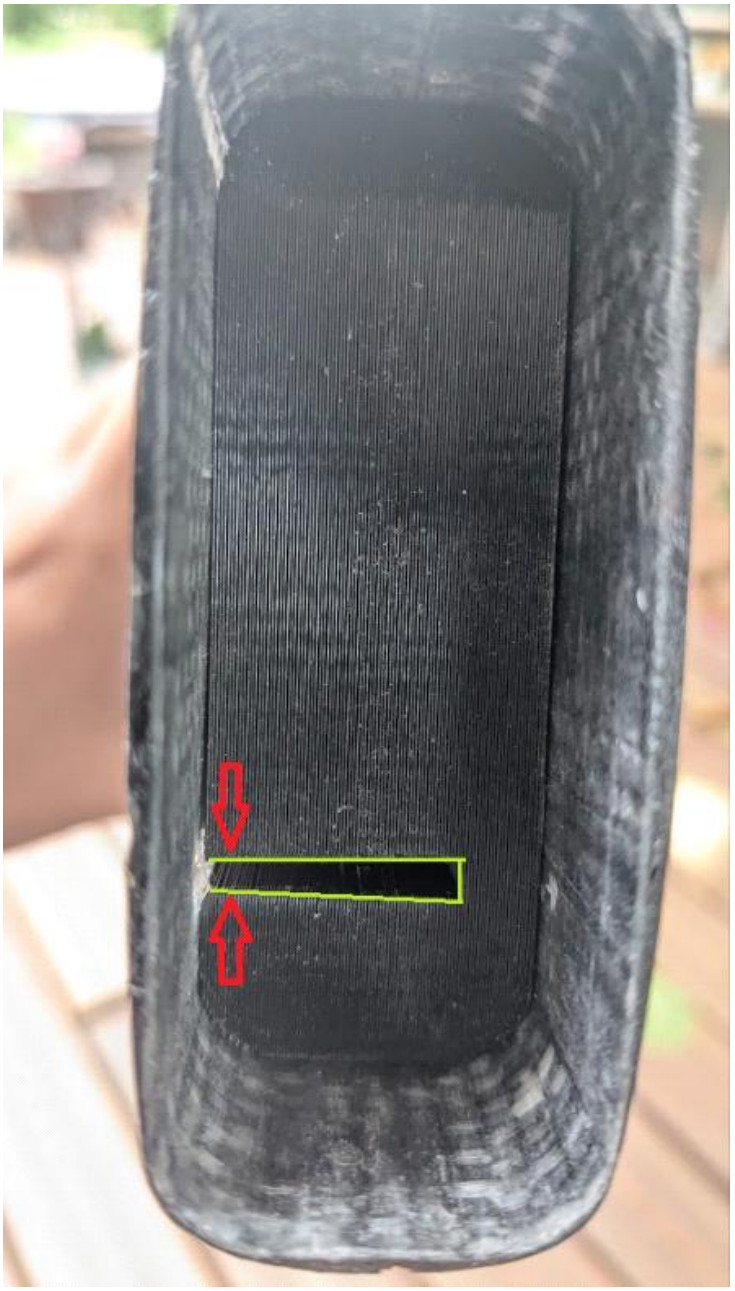
9/14
- ~ 5 hours
- Started drilling holes in the carbon bars, so we can assemble the chassis
- Buffed the top plate of the carbon
- Wrote the first revision of the motor controller report
- Added motor controller report to website and removed useless buttons

9/15
- ~ 3 hours
- Assembled the chassis
    - Noticed the bar inserts flexed a little because the outer profile did not match the inner profile of the bar. Using the router we cleared out the area so the swerve modules would fit
    - One of the swerve plates broke at one of the bearing holes. If I were to re-make it I would remove that hole or add more

material

- Started building rims of the swerve modules



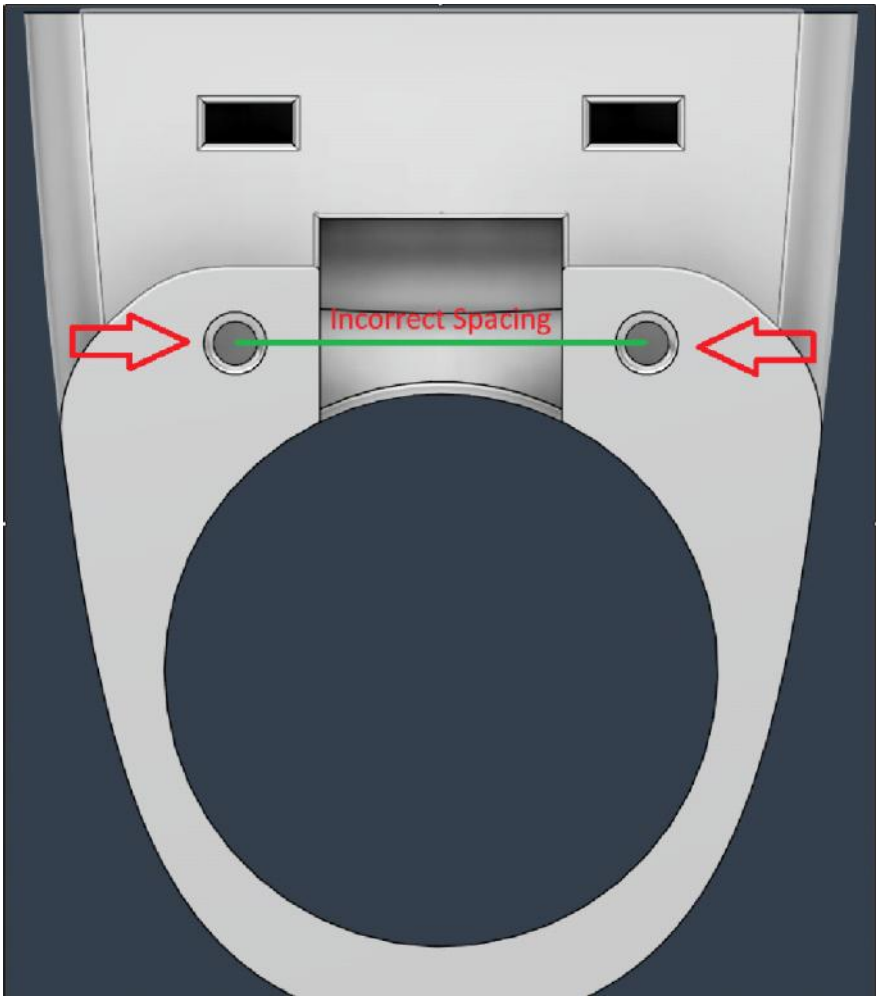


9/16

- ~ 1.5 hours
- Put tires on rims, assembled planetary gear systems, and ran into an issue with the motor side support
    - The rim seemed to have a taper that was only present in the physical form - likely due to the 3D printer
    - The side support was adjusted to be stronger and was given slots to increase our tolerances

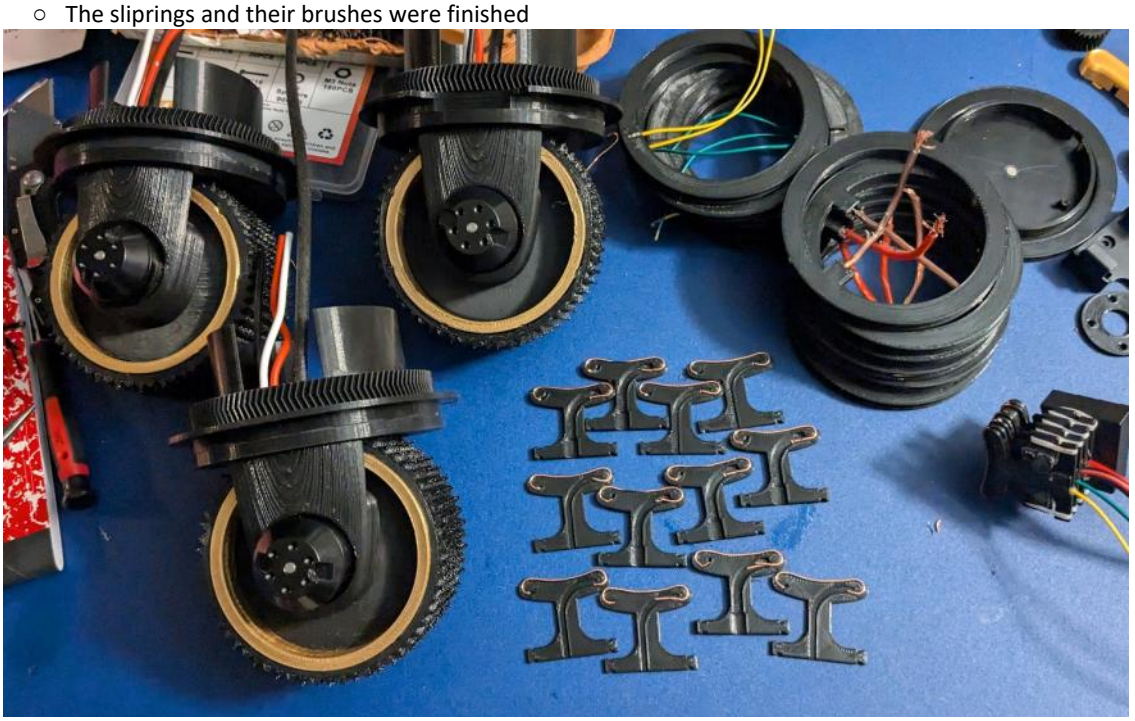
9/17

- ~ 1 hour
- Tested new motor side support, but ran into an issue where the planetary side holes did not match up anymore
    - This was caused by increasing the width of the side support. The hole was referred off of the exterior of the side support rather than the distance apart from each other.



9/21

- ~ 4 hours
- Build more of the swerve modules
    - Attached wheel supports and connected wheels to bearings
    - The sliprings and their brushes were finished



9/23

- ~ 1.5 hours
- Connected wires to brushes and sanded down high points of sliprings

9/24

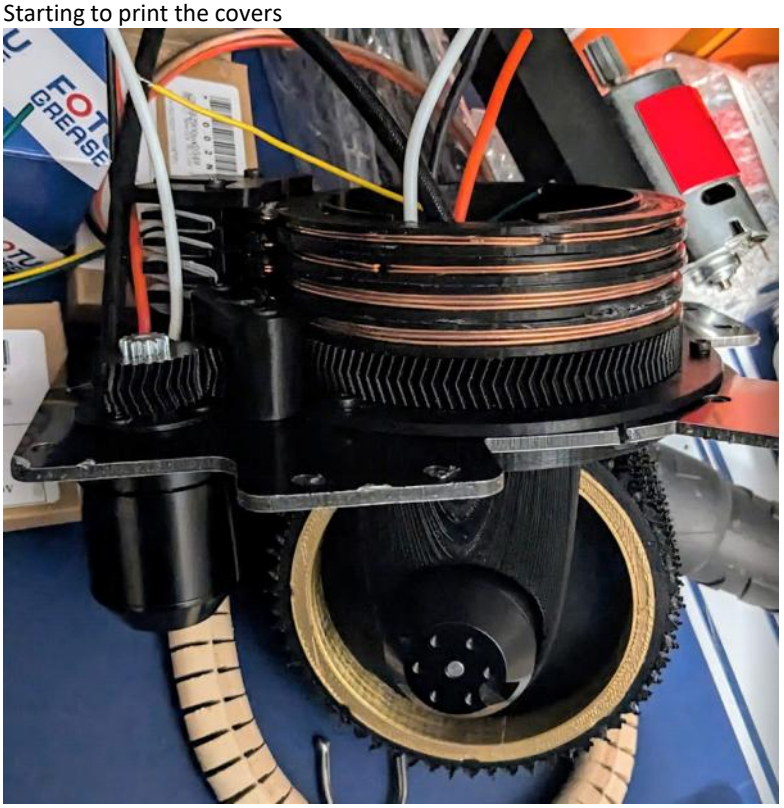
- ~ 2 hours
- Glued the sliprings to the swerve
- Fixed some problems with parts not fitting in CAD
    - The axle was too large, and the lower 550 support was angled 45 degrees off

9/26

- ~ 1 hour
- Found some more problems with the swerve modules
    - The space between the rotation NEO 550's bolts and the gear was not enough. To fix this I added a recess into the 550 mounting plate, which should create enough space between the two components
    - Found the spacing between the sliprings and the brushes do not match due to the increase in width of the power sliprings. To address this I created spacers that are 1mm and .5mm in depth. Together I can create spaces with the factor of .5, if we need more granularity sanding may be used or additional spacers may be created

9/28

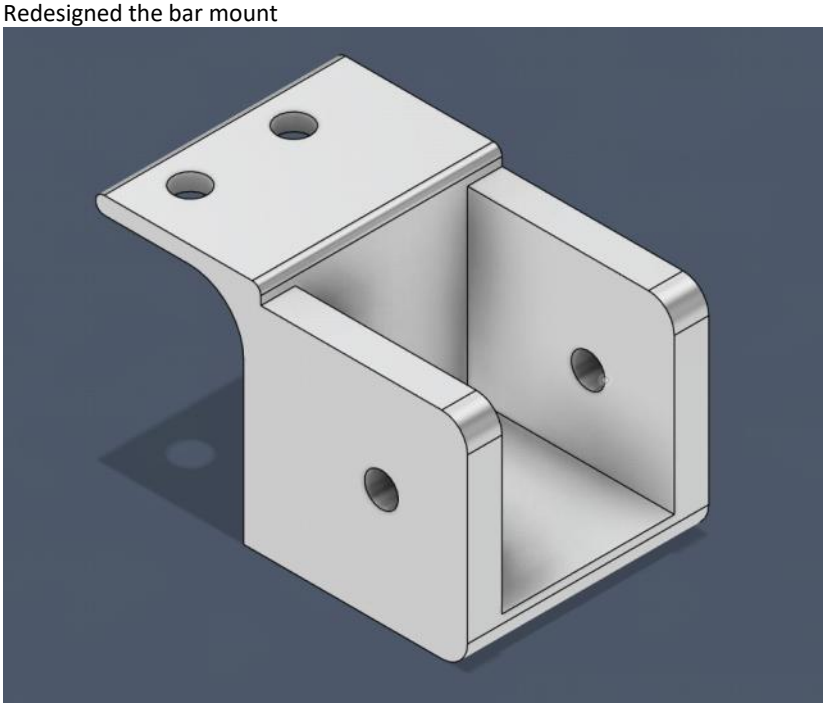
- ~ 4 hours
- Continued assembling swerve modules
    - Finished getting the brushes attached
    - Connected all the gears
    - Starting to print the covers



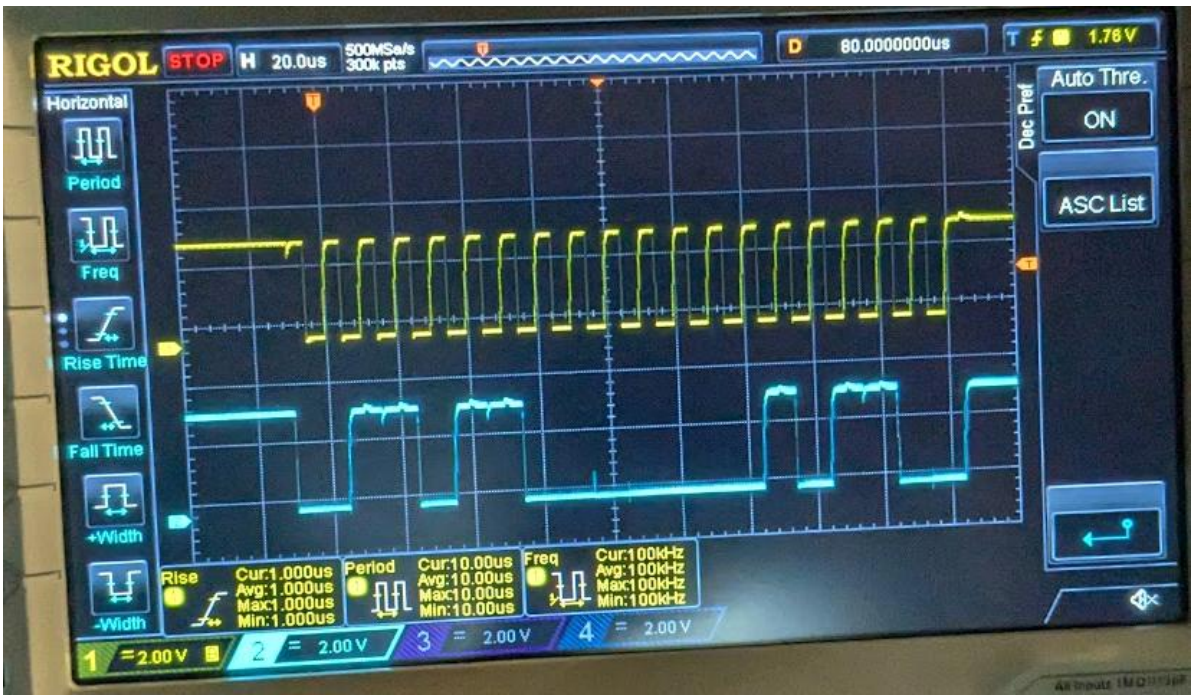
9/29

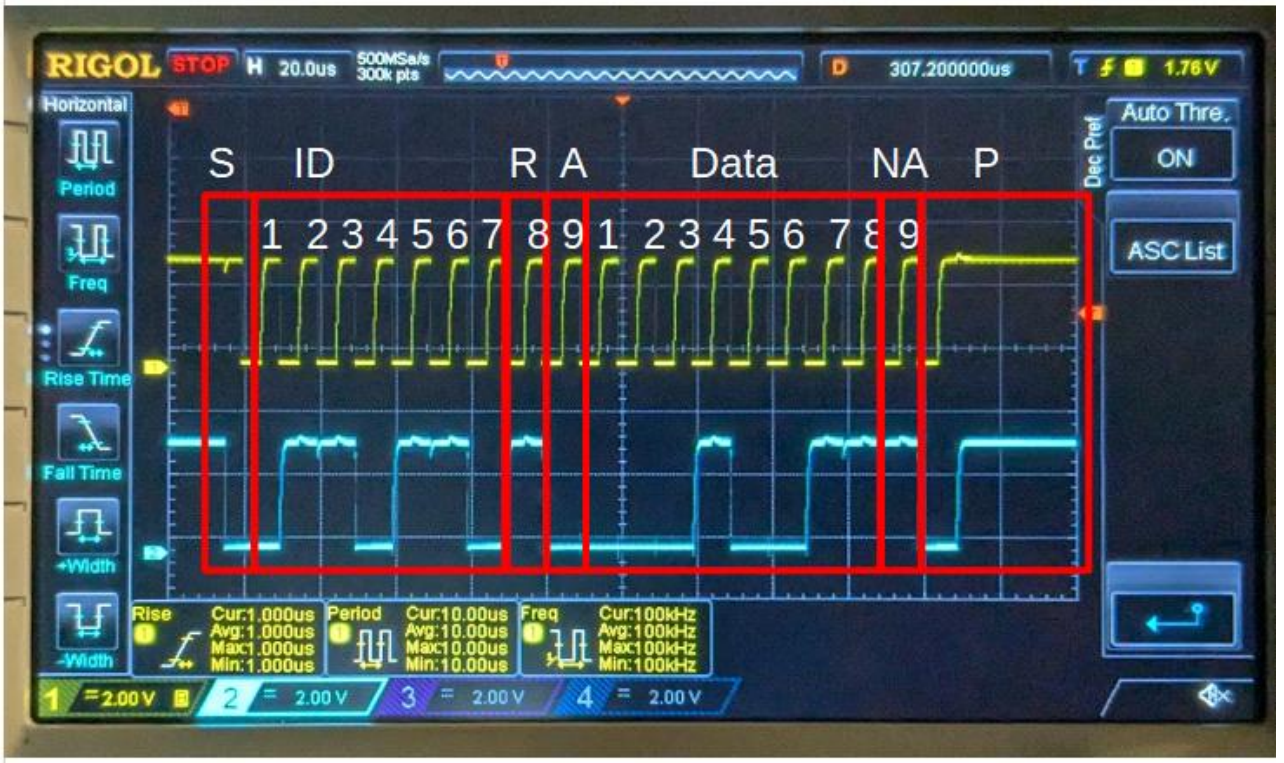
- ~6 hours
- Adjusted the cover top support to a chamfer instead of filet due to support issues
- Printed 3 out of the 4 covers

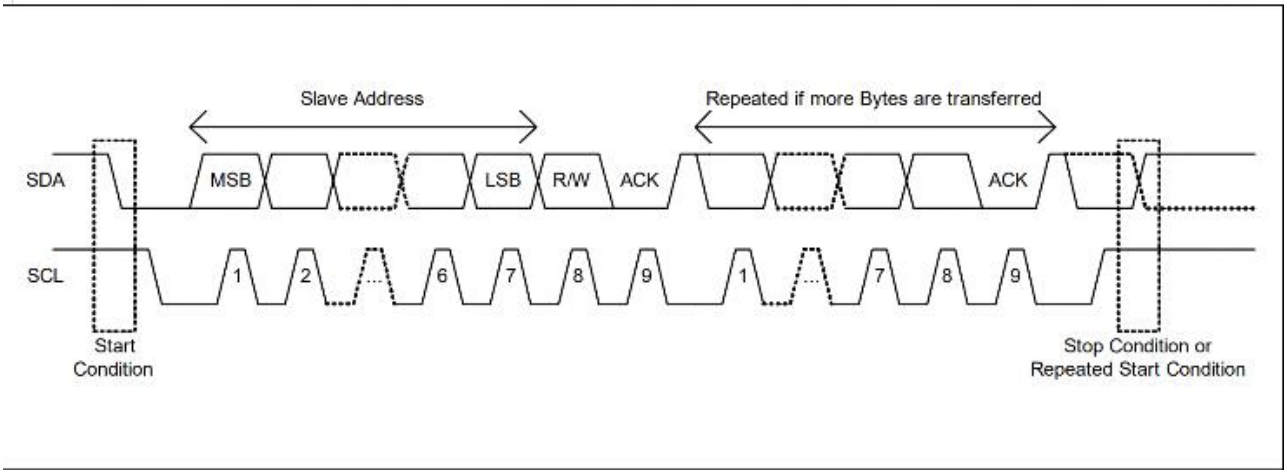
- Cut the aluminum bars down to length
- Redesigned the bar mount



- Reviewed the motor controller schematic and board
    - Updated the report and added logo
- Updated the website with the GitHub link
- Started looking at the AS5600 and got it to work using Arduino





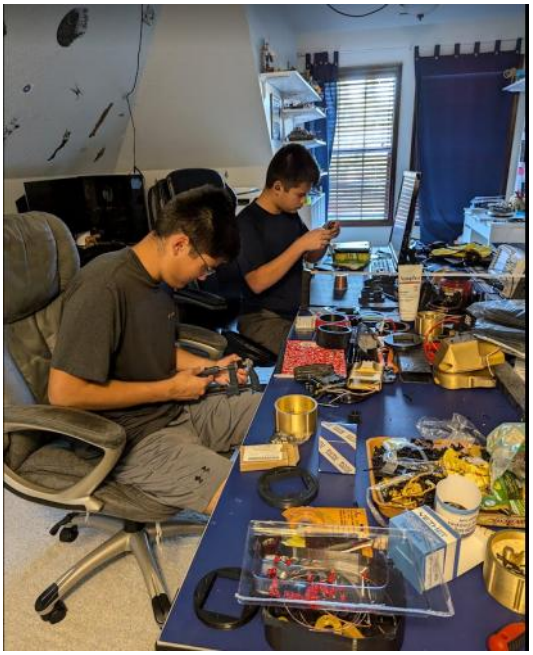


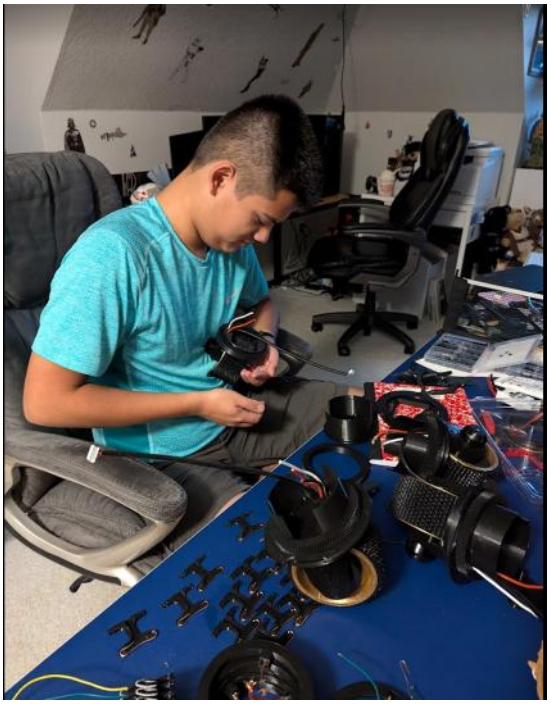
9/30

- ~ 2 hours
- Working on getting the I2C code to work on the Tang
  - Looking at the oscilloscope and simulations
  - The issue likely stems from Lushay Labs I2C code fundamentally or the stimulus that we us to activate the I2C code from Lushay Labs.

---------- 10/1 -10/11 ----------

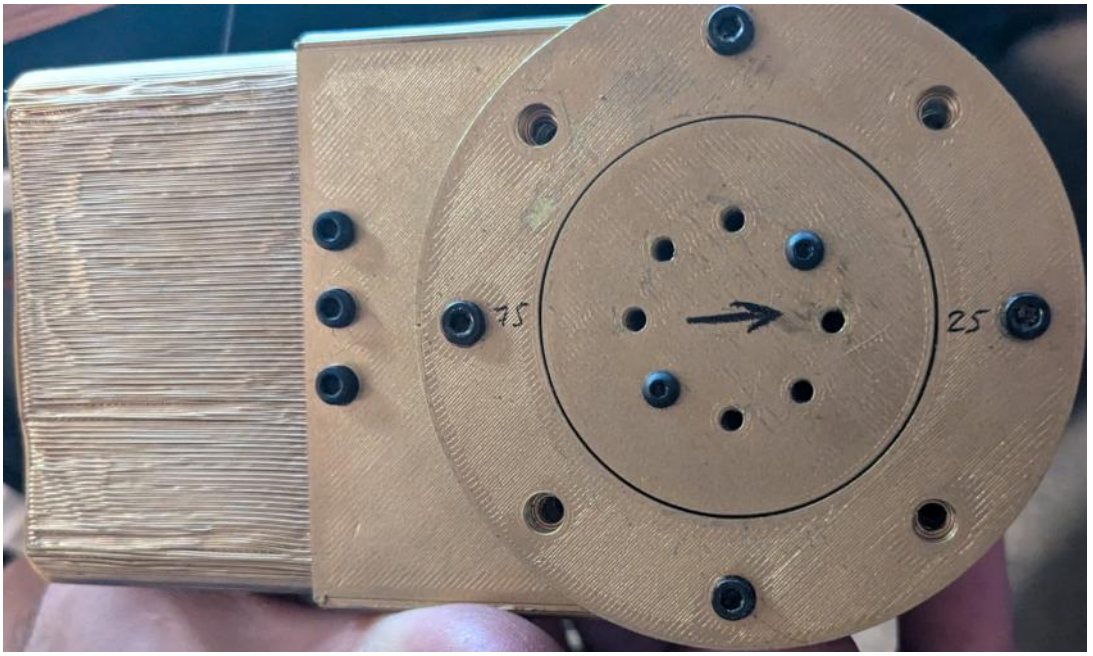
- Worked on Swerve modules
  - Finished getting swerve modules fabricated and constructed
    - Creating brushes
    - Getting brushes to align with sliprings
  - Created new version of cover with an acrylic top because the 3D printed top had many inconsistencies
  - Created an initial cover that protects hands from the top carbon plate edges





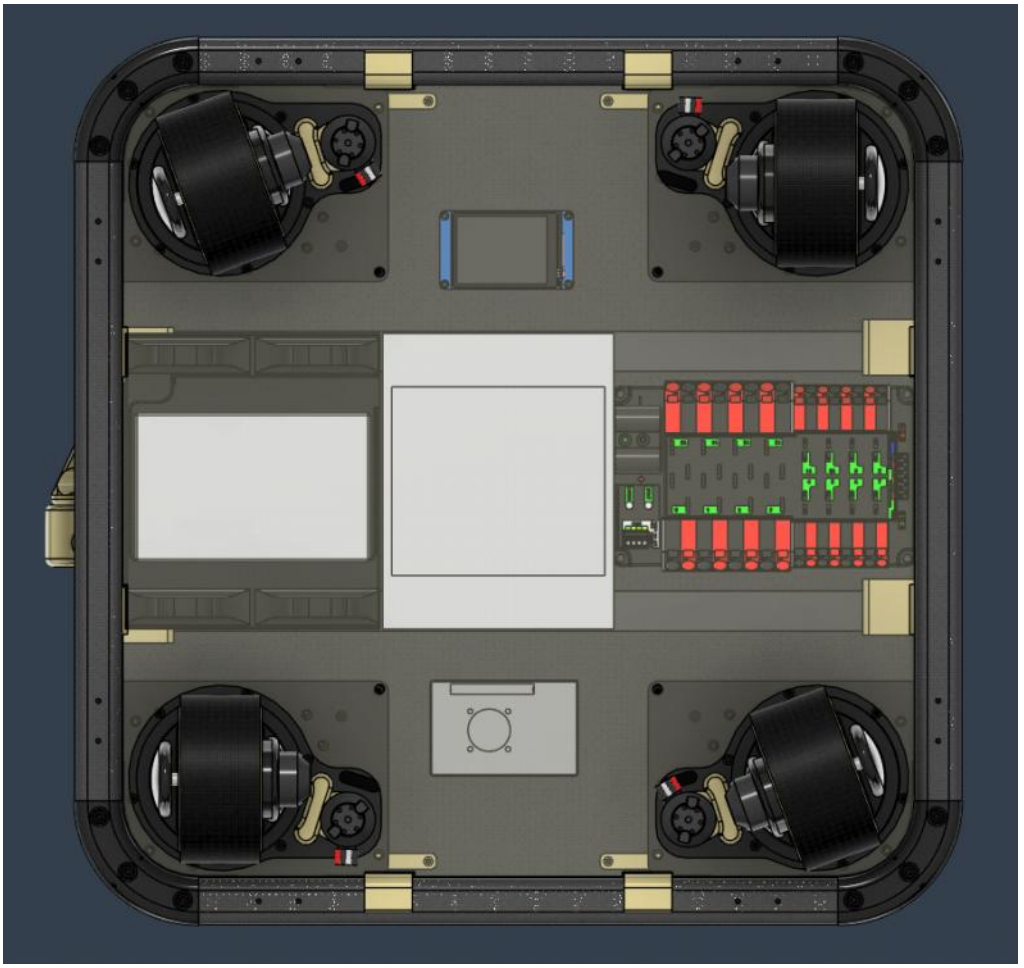
10/12

- 7 hours
- Base Plate
    - Cut the carbon plate to exact size
    - Refined the covers for the carbon plate edges
- Arm
    - We lost the initial program to create PWM for the servos, so a new program had to written
    - With this we could take our keyboard inputs to set a PWM value
    - Used this to determine the range which the servos operated within
Center Arm Servo	75 (Closed), 25 (Extended)
Base Servo	20 (Fully Backward), 70 (Fully Forward)
Grabber	80 (Closed), 50 (Open)
Wrist	TBD
    - Assembled the arm for scale



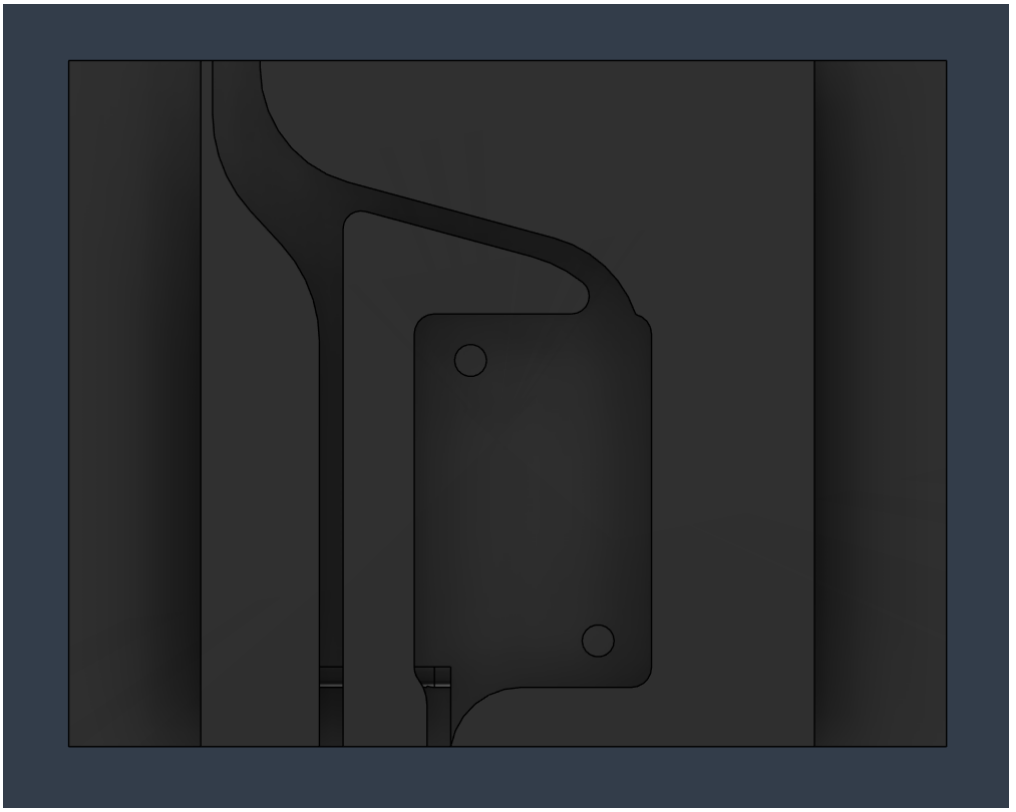
10/13

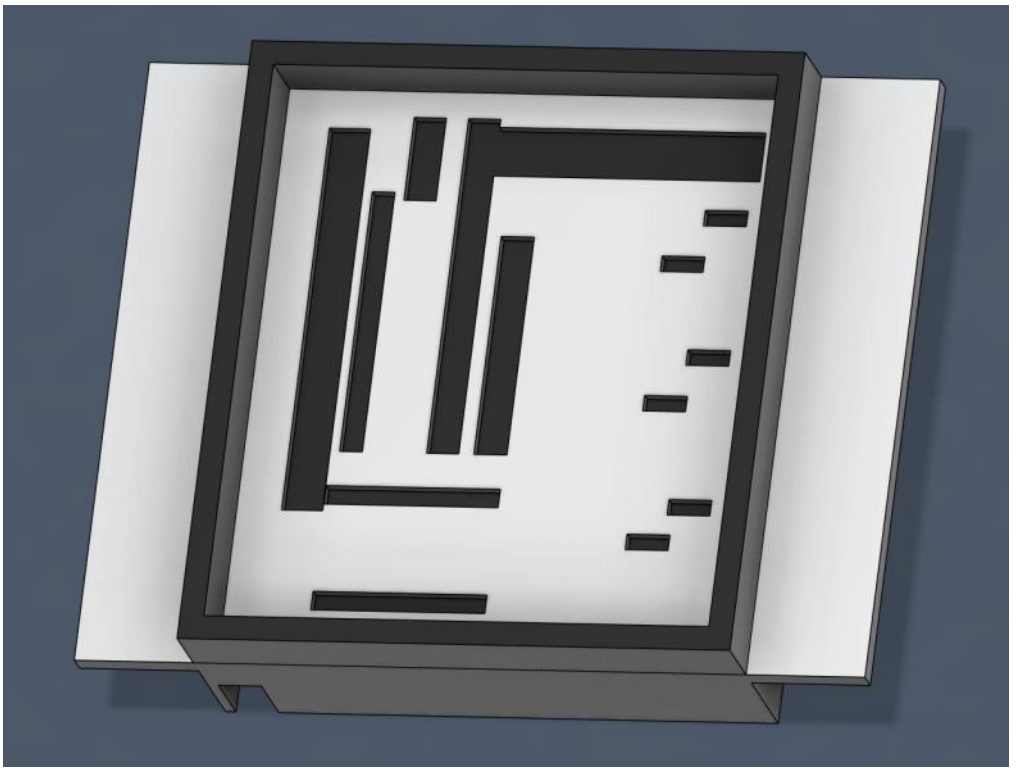
- ~ 4 hours
- Worked on the motherboard
    - Added a jumper to change the drive wire's adjacent to 5V or GND
- Started looking at electrical component placing on the robot
    - Made a battery case
    - Started working on a case for the mother board



10/20

- ~ 4 hours
- Worked on the case for the motherboard and breaker
    - The first image shows the cutout for the breaker and its wire routes
    - The second image shows the top which has indents for the motherboard solder joints



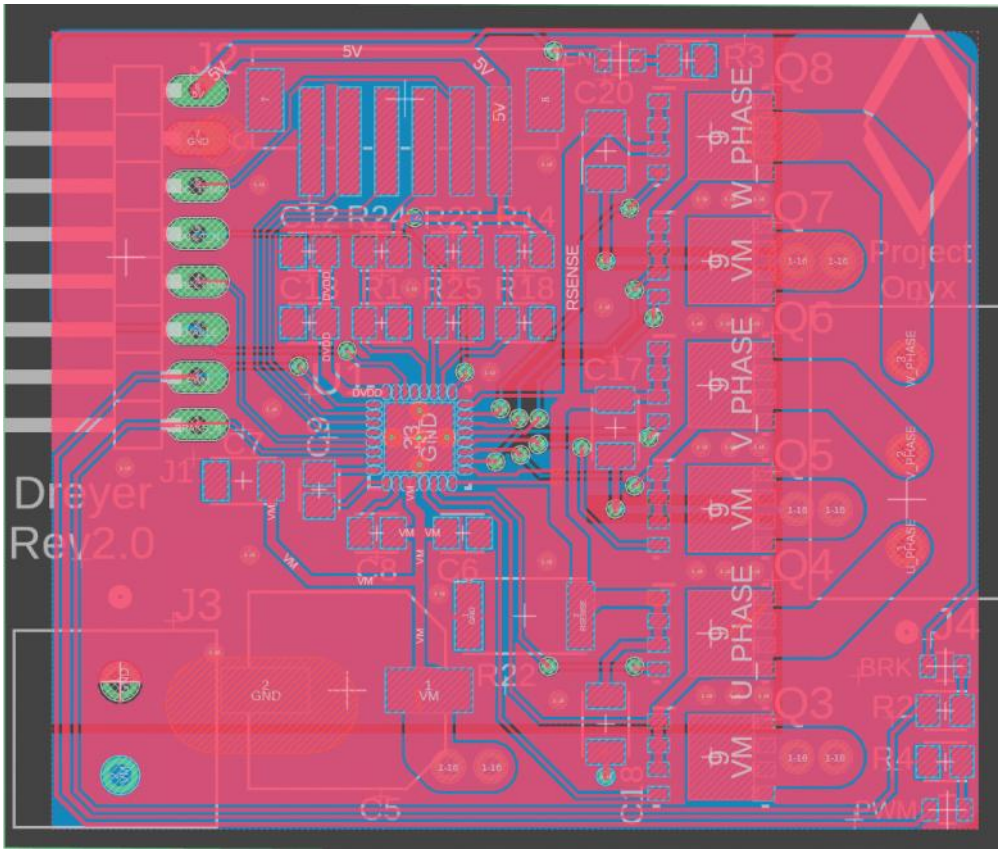


10/21

- ~ .5 hour
- Optimized the motherboard wires

10/24

- ~ 1 hours
- Looked at motor controller boards, and added status LEDs for: Brake, PWM, and Enable



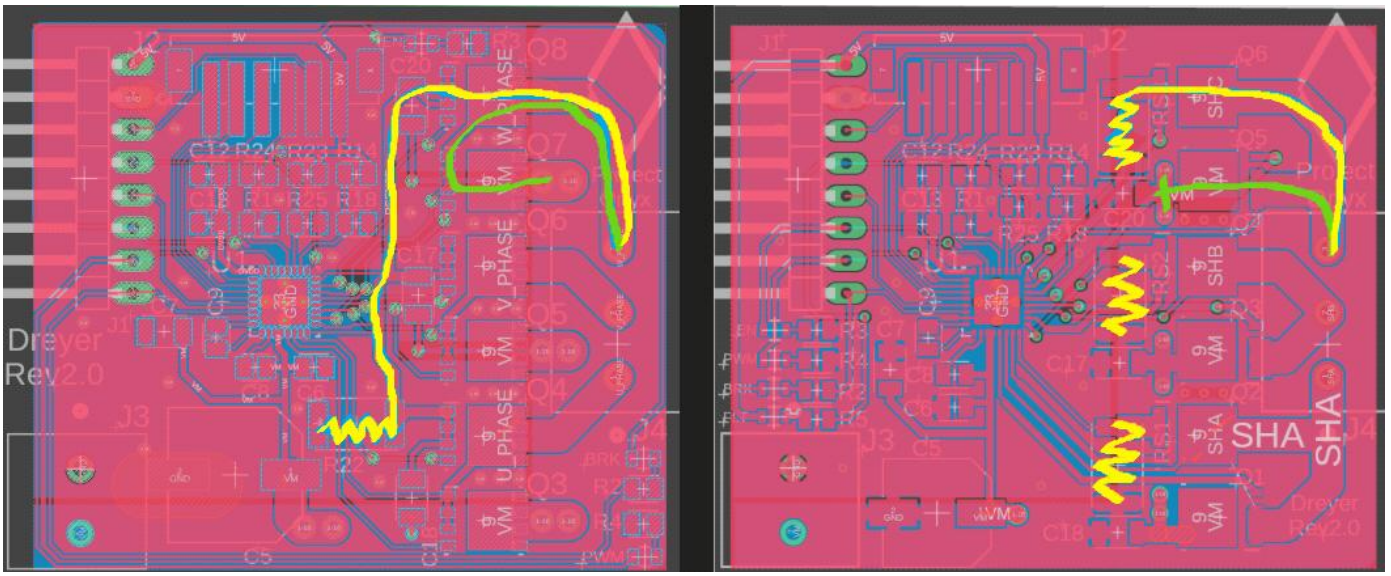
10/25
- ~ 2 hours
- Adjusted the resistor and Zener diode values for the LEDs on both the Motherboard and Motor controllers

10/26
- ~ 6 hours
- Worked on I2C communication for the AS5600
    - Updated the FSM
    - Looked into documentation about repeated starts, nacks and acks for the I2C bus
    - Updated the I2C section of the PWM Control MAS

10/27
- ~ 5 hours
- Worked on I2C communication for the AS5600, and finally got the AS5600 to communicate with the Tang
    - Re-updated the FSM
    - Updated the I2C section of the PWM Control MAS

11/1
- ~ 2 hours
- Added 3.3V to the motherboard for the magnet encoders (AS5600)
- Created BOM and CPL file for the motor controller V2

11/2
- ~ 7 hours
- Completely reworked the connections from the NMOS to the motor outputs (old on the left shows the imbalance of paths to VM vs GND)
- Decided to do 3 individual sense resistors
- Moved LEDs all to the left
- Updated BOM and Pick-n-place
- Started Motherboard BOMs

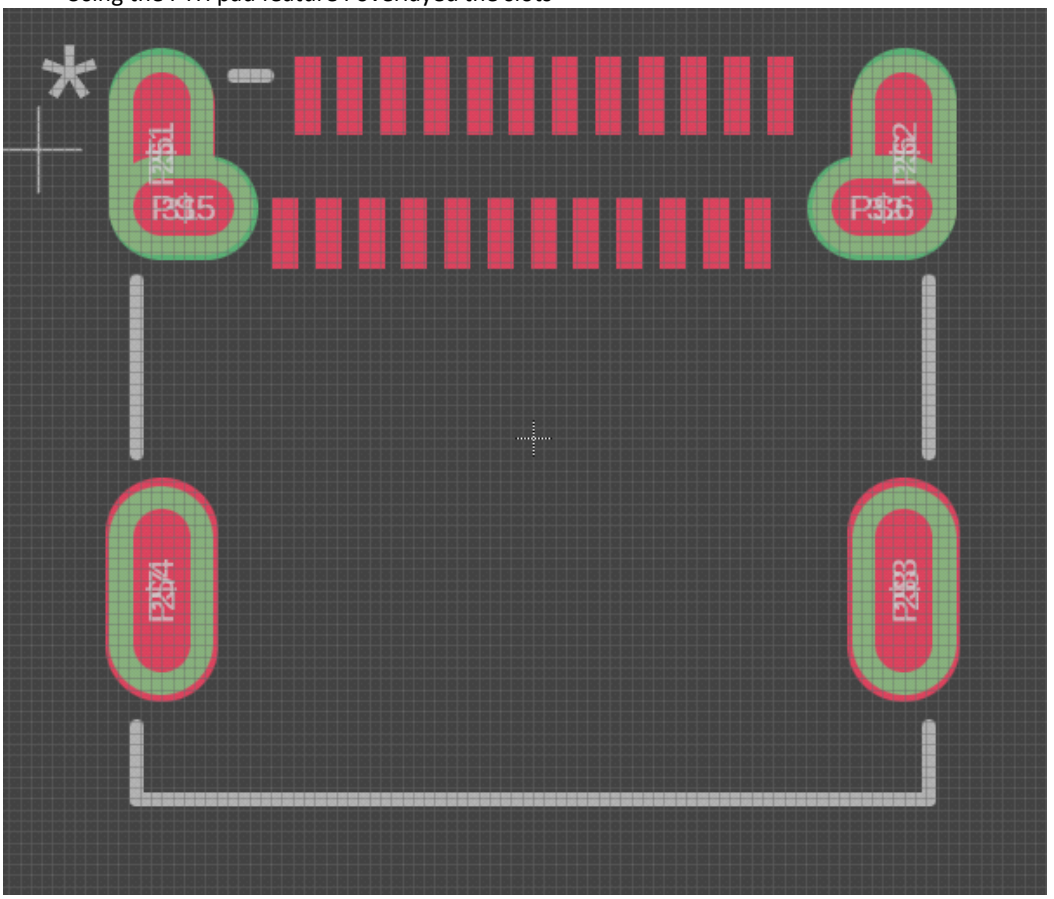


11/3

- ~ 4 hours
- Created BOM and CPL files for the motherboard
- Used JLCPCB DFM to check for problems in our boards

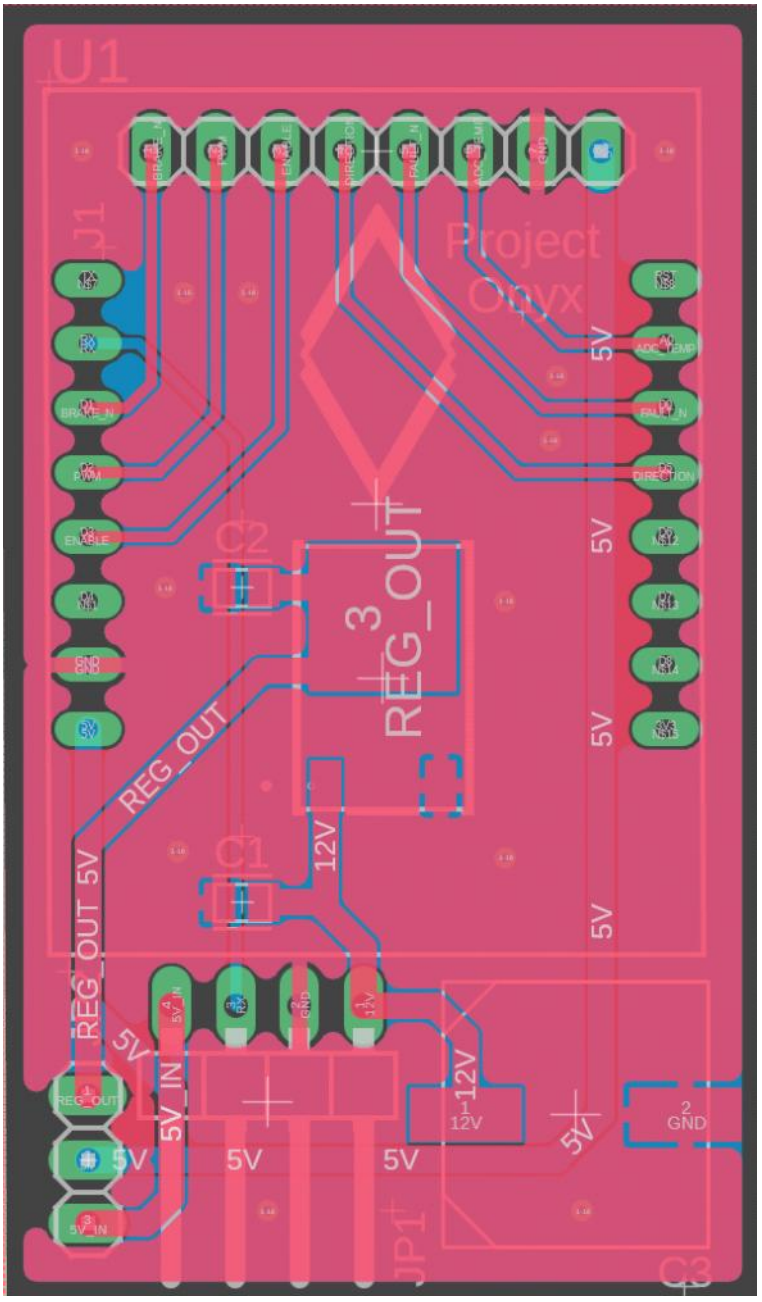
11/4

- ~ 3 hours
- Looked over the motherboard and motor controller PCBs to check for any problems
- Had to fix the ultra-librarian file for the USB C connector because the slots were registered as holes
    - Using the PTH pad feature I overlayed the slots



11/5

- ~ 3 hours
- Created a ESP8266 board when combined with our motor controller board can replace the SparkMax
- Sent orders out to JLPCB

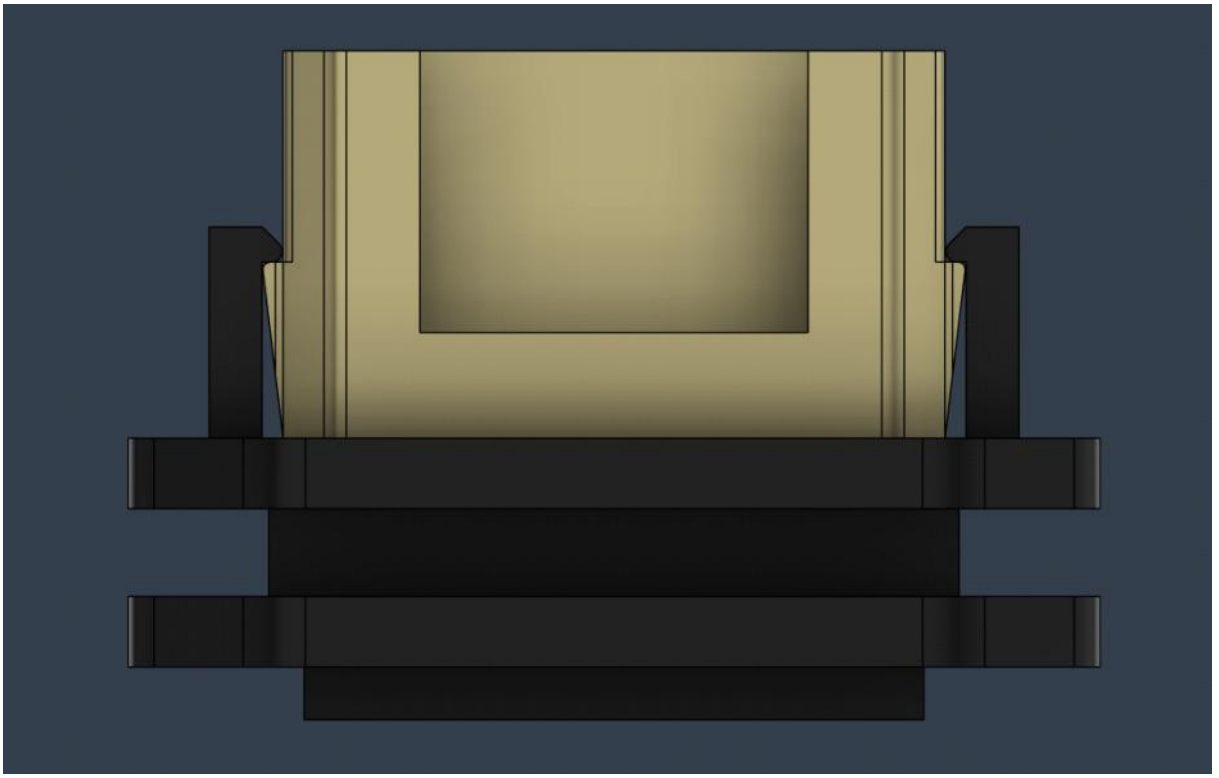


11/7

- ~ 3 hours
- Fixed the JLCPCB board and re-sent them out to order

11/8

- ~ 2 hours
- Created a housing for all the arm connections

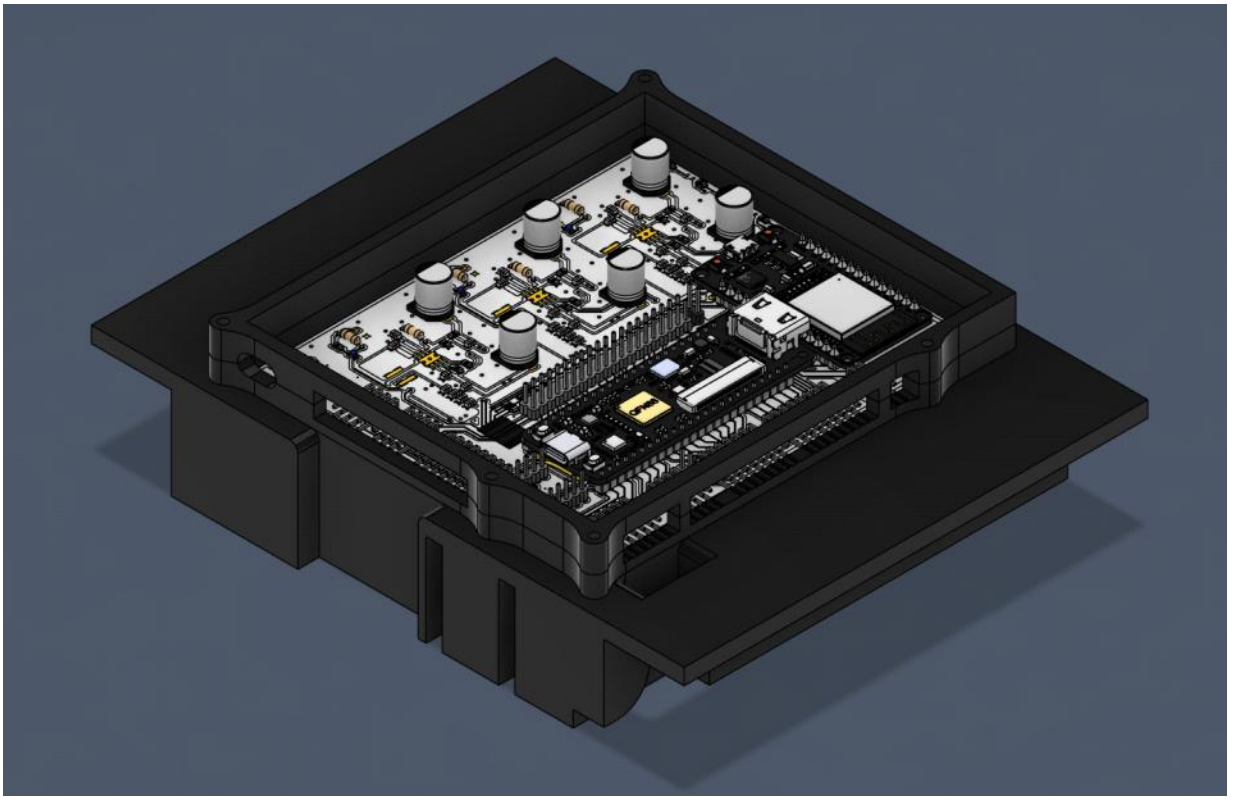


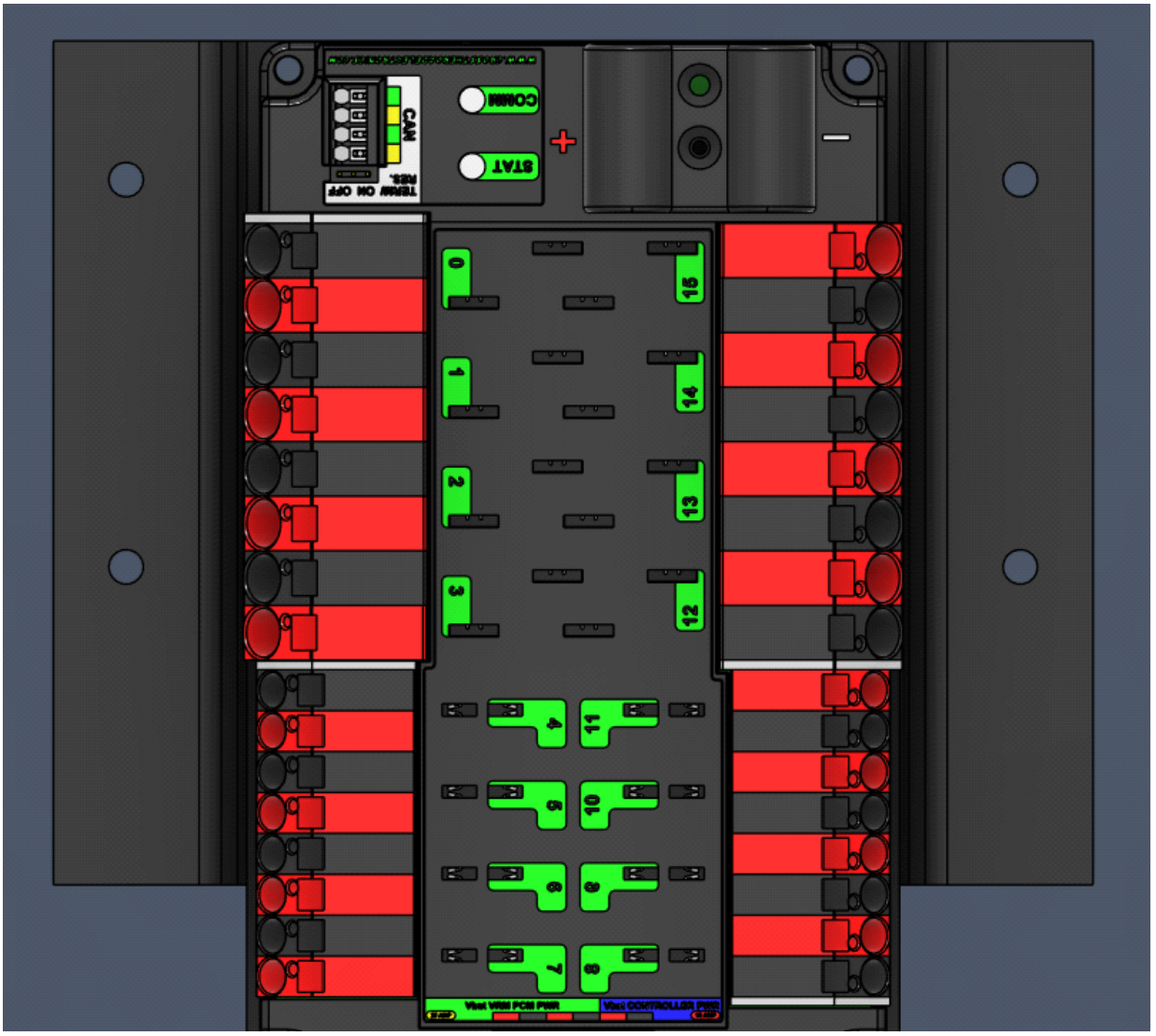
11/9

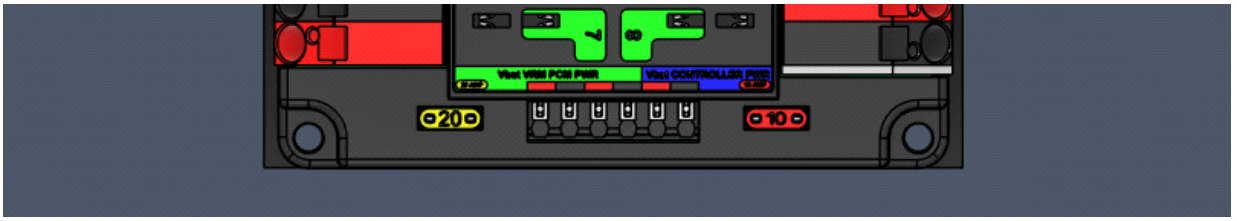
- ~ 4 hours
- Wrote Verilog code for UART
- Created a MAS for the UART communication
- Found out that the waves on the HDMI capable pins of Tang Nano were worse than the other pins due to their capacitors.

11/10

- ~ 6 hours
- Debugging UART communication to the Arduino
    - Found that a clock division code of 110-120 was "stable" (outside of this, start to see occasional errors), so picked 116
- Created PDP mount, updated the motherboard mount




CAN
COMM
STAT
TERM ON OFF
RES.
+
—
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

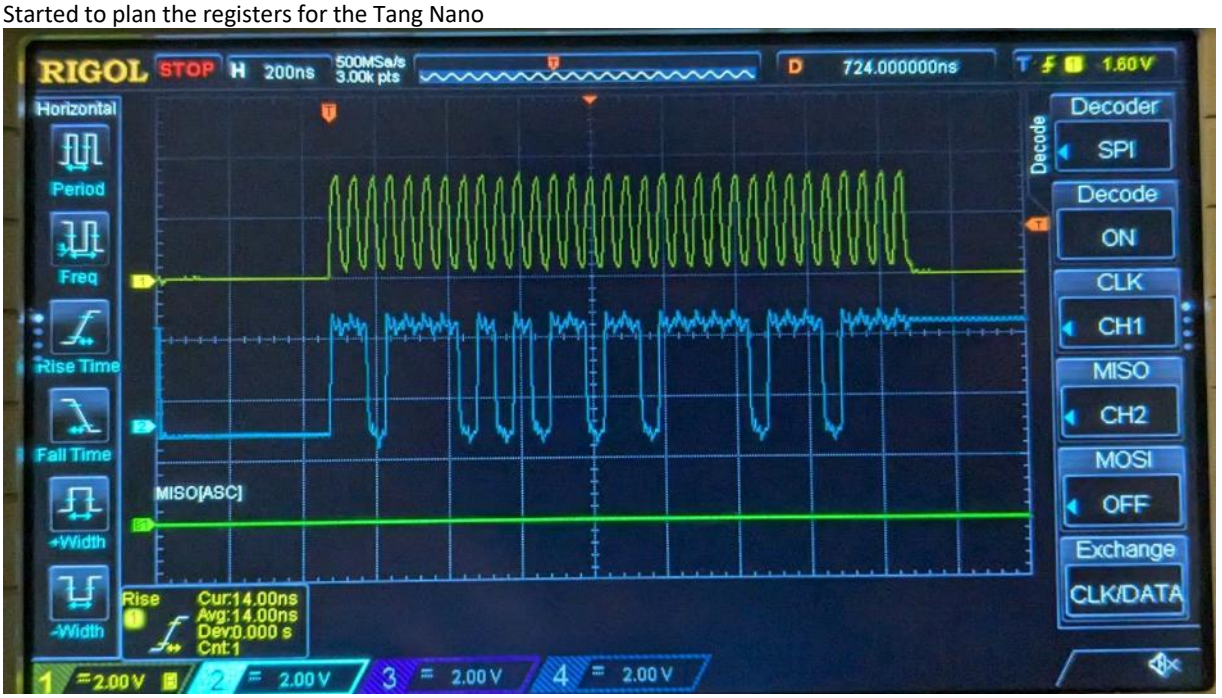


11/11

- ~ 3 hours
- Made some modifications to the PDP, Motherboard and Battery mounts
    - Added an extension to the "wings" that goes over to the other side of the bar. A bolt will travel through the outer part of the "wing" through the bar and screw into the main body of the mount.
- Updated diagrams in the motor controller MAS document

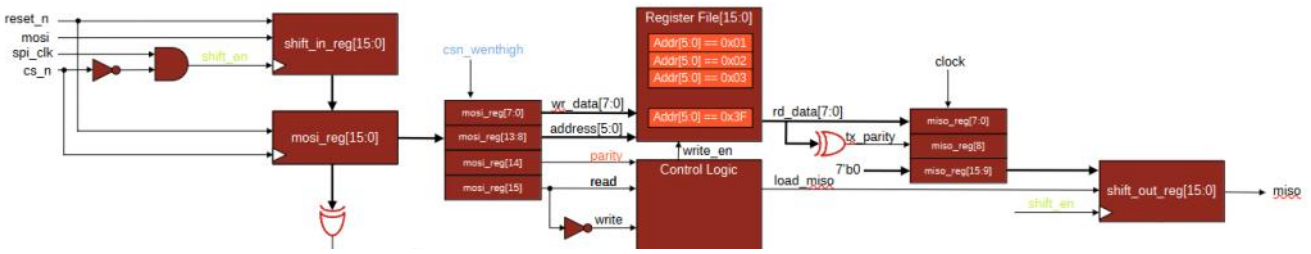
11/12

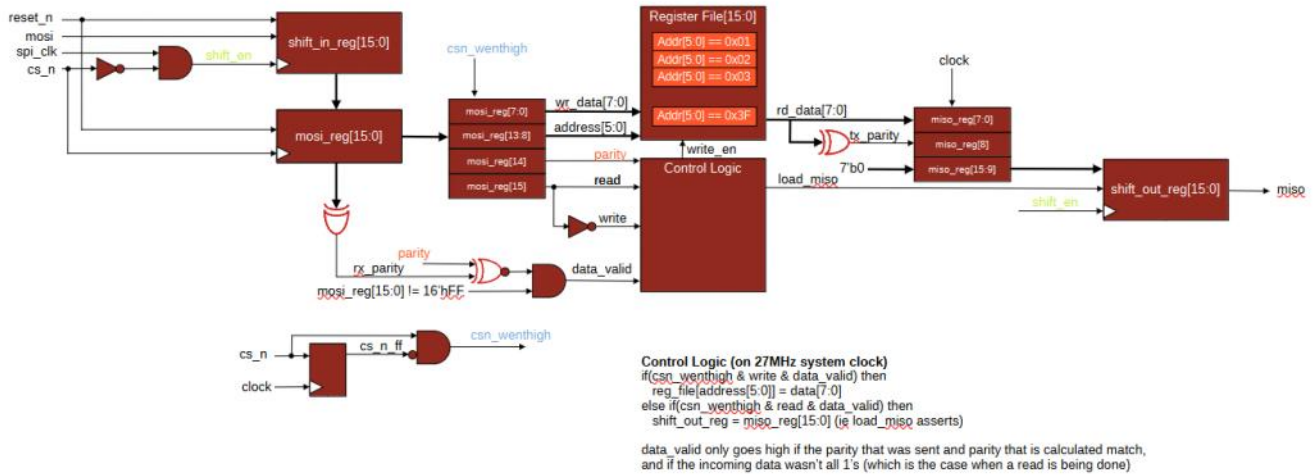
- ~ 2 hours
- Tested the SPI on the Orange Pi
    - Got the Pi to send "DEAD BEEF"
    - Found that there is no set limit on the amount of bits sent in the Conda environment
- Started to plan the registers for the Tang Nano



11/14

- OrangePi's Debian somehow got corrupted: the spidev in Python wouldn't work, and when loading the OrangePi Configuration - all the ports were just marked [] (instead of having a label next to the []) . Was luckily able to "fix" it by copying the /boot/orangepiEnv.txt from the boot SD card, and deleting the /boot/config-6.1.43-rockchip-rk3588 file. On restart, Debian seems to have created a new version of this file, which is working!
- Diagram of Tang Nano function





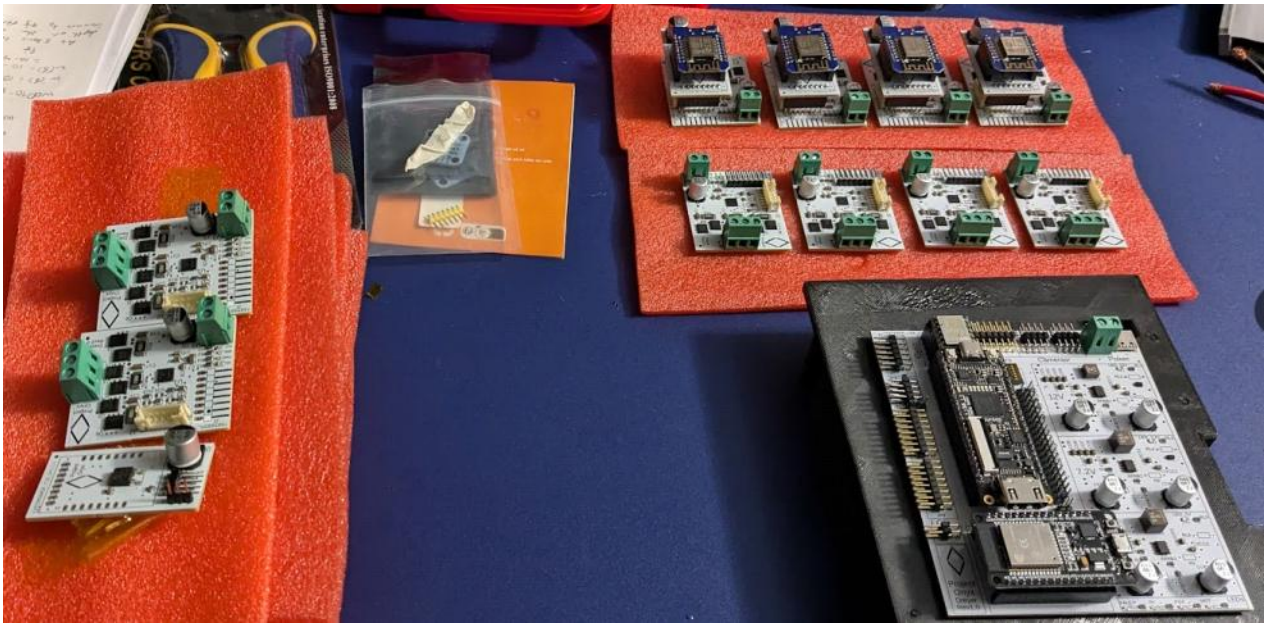
---------- 400 hours to this point! ----------

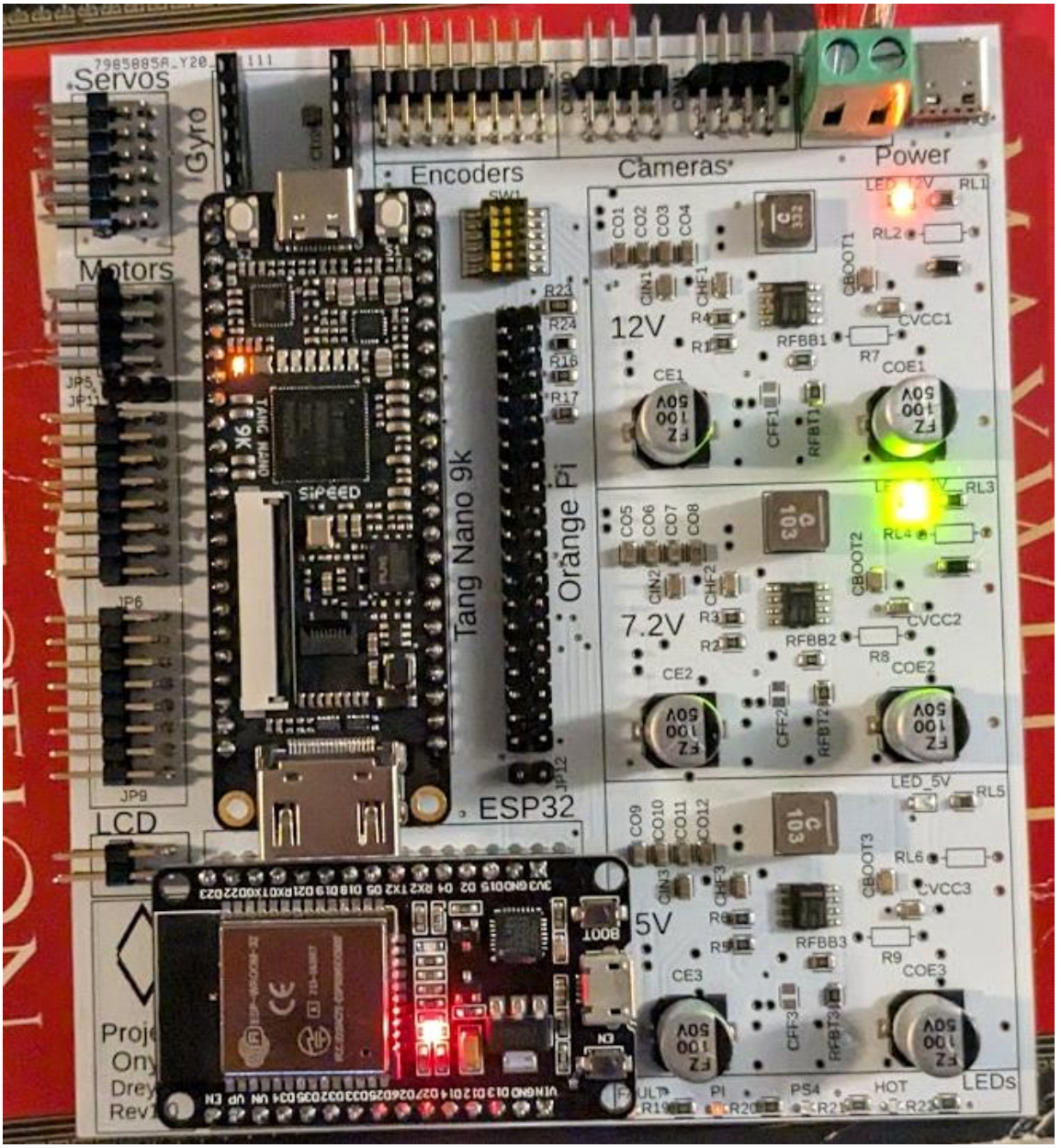
11/16
- ~ 2 hours
- Worked on writing the register file

11/17
- Working on register file

11/18 & 19
- ~ 3 hours
- Got SPI working with the addition of the MISO pin and the new data framing
- Working on creating the top level file

11/21
- ~ 2.5 hours
- JLCPCB boards have arrived
- Soldered pin headers onto the boards
- Power up test on the Motherboard with the ESP, Tang, and Pi





11/22

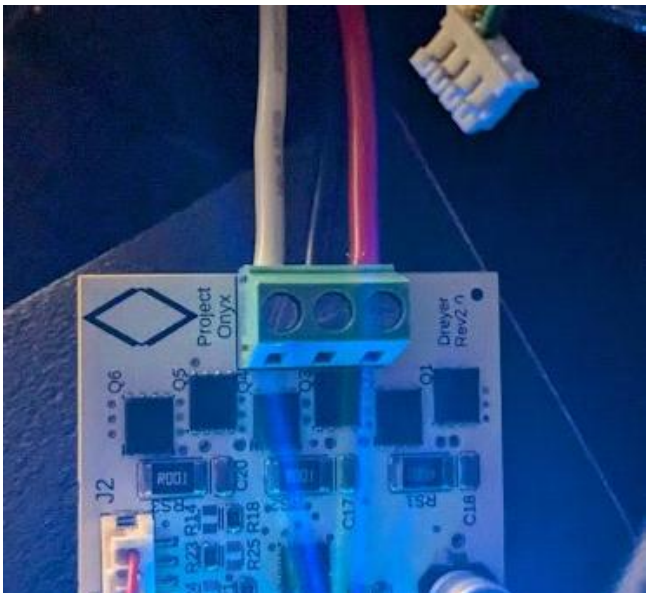
- ~2.5 hours
- Worked on assembling the electrical covers for the robot
    - Had to make a spacer for the motherboard because the mounting points were not changed with the other alterations for the raised Power Distribution Panel

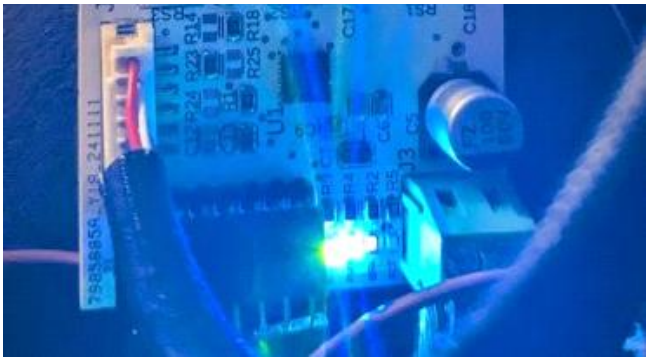
11/23

- ~ 8 hours
- Worked on the central main connector
- Tested motor controller
    - The electrolytic cap on the first exploded after plugging the U and W phases into opposite sides
    - The second board worked as we had hoped


J2
U1
C17
RS1
C13
J4
Q1
Q2
Dreyer
Rev2.0

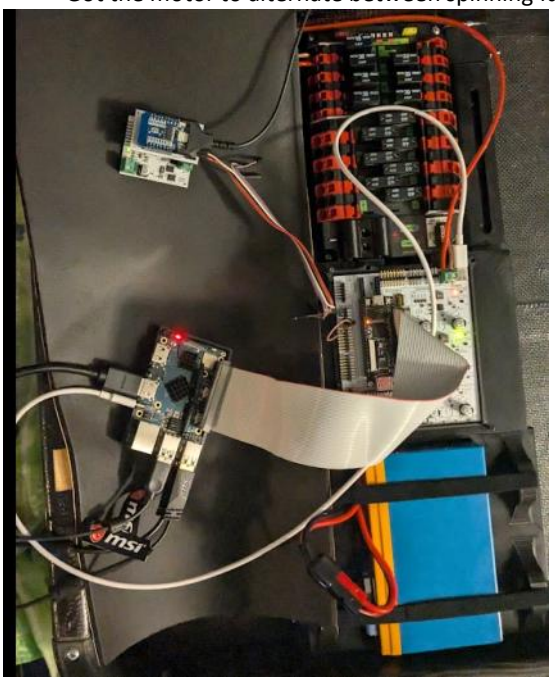

J2
Q6
Q5
Project
U1
C17
J3
FZ
100
50V
RS1
C13
J4
Q1
Dreyer
Rev2.0


Project
Onyx
Dreyer
Rev2.0
Q6
Q5
Q1
J2



11/24

- ~ 5 hours
- Tested the subsystem UART option for driving the DRV8320
    - Orange Pi -> Motherboard -> Tang Nano -> ESP32 (UART) -> Motor Controller -> Motor
    - Got the motor to alternate between spinning faster and slower



11/26

- ~ 5 hours
- Working on the subsystem I2C from the AS5600 to the Tang and ultimately the Orange Pi

11/27

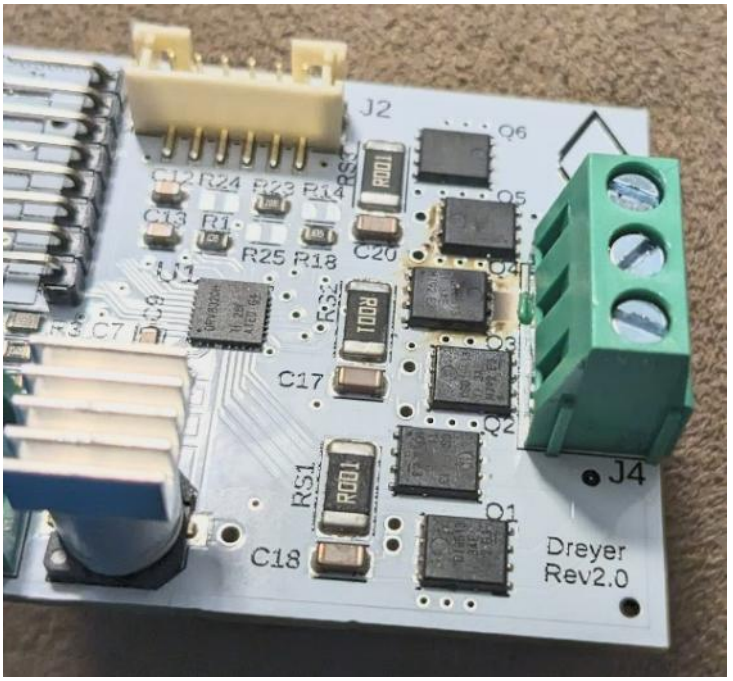
- ~ 4 hours
- Got the I2C from the Tang to work with the AS5600 and got the Tang to relay the raw angle information to the Orange Pi

12/5

- ~ 3 hours
- Working on the motor rotation code

12/9

- ~ 2 hours
- Tested the rotation motor controllers
- The Mosfet failed as seen the photo below
    - Unsure of the cause currently

- 

12/16

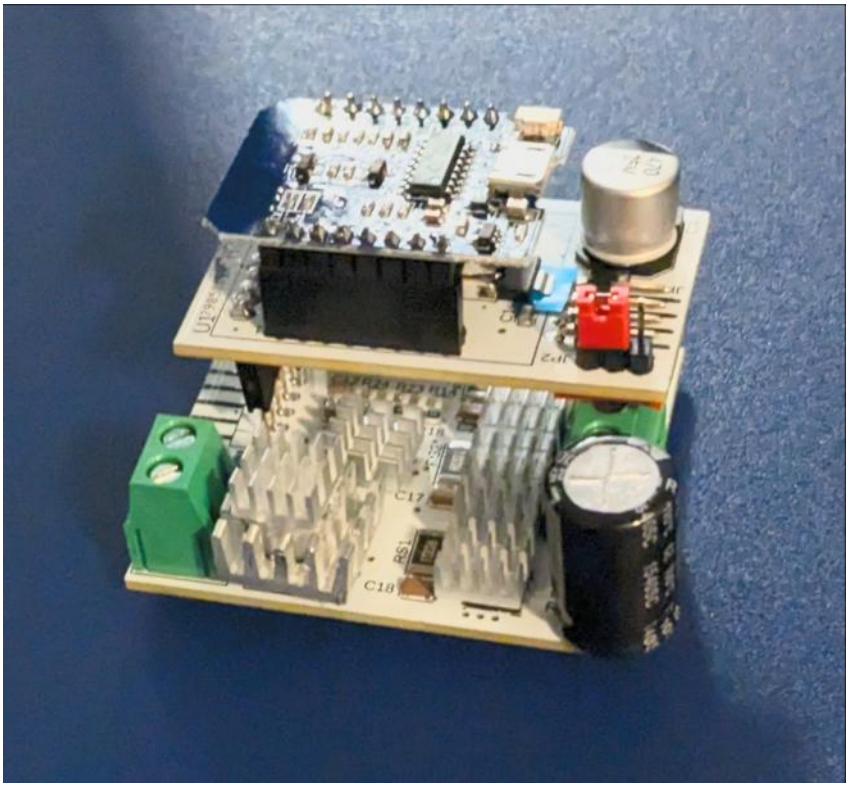
- ~ 2 hours
- Working on the rotation motor functionality
- Revisions made after Mosfet failure
    - Added a hammer feature - a sequence of short bursts of energy - to unstick the motor
    - Installed a motor stall monitor to prevent future damage

12/20

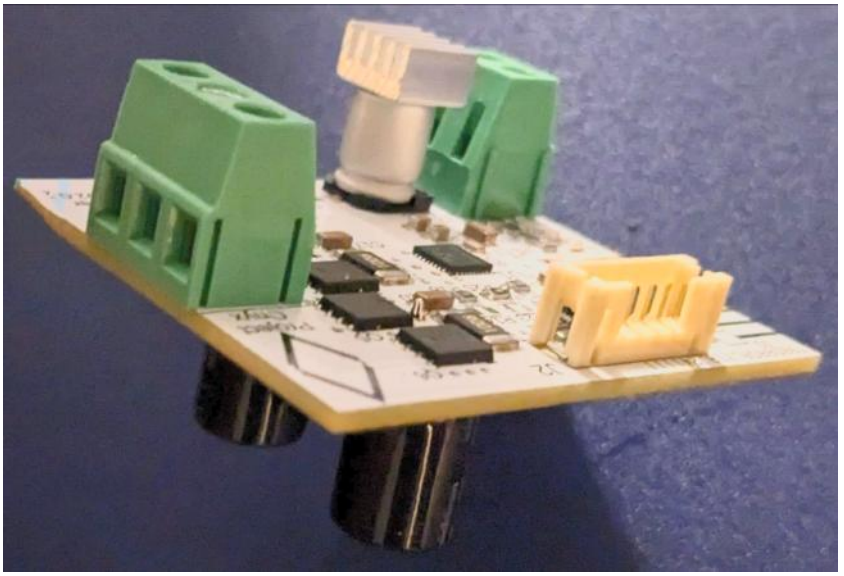
- ~ 2 hours
- Working on the rotation motor functionality
- Opened SparkMAX and analyzed the BLDC motor phases to find discrepancies between our boards and theirs
    - Rev uses 6 100uF capacitors while we use 1 100uF capacitor for filtering
- After looking at Ti documentation it appears that the Idrive setting needs to be 18k to GND instead of VCC

12/21

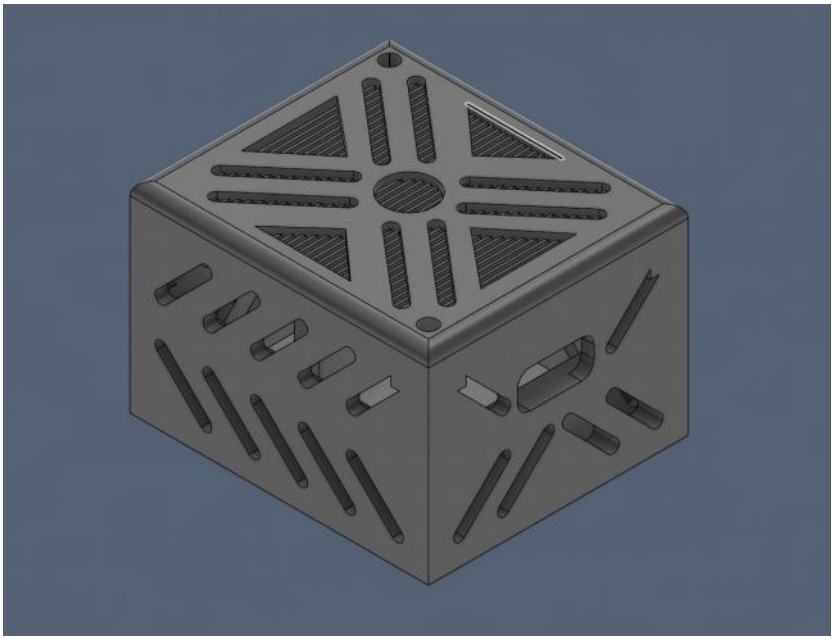
- ~ 4 hours
- Fixed motor controller boards
    - Moved Idrive resistor from VCC to GND
    - Added 1 1000uF capacitor to the drive boards; added 2 470uF capacitors to the rotation boards
- Created a case for the rotation motor controllers



Drive Motor Controller

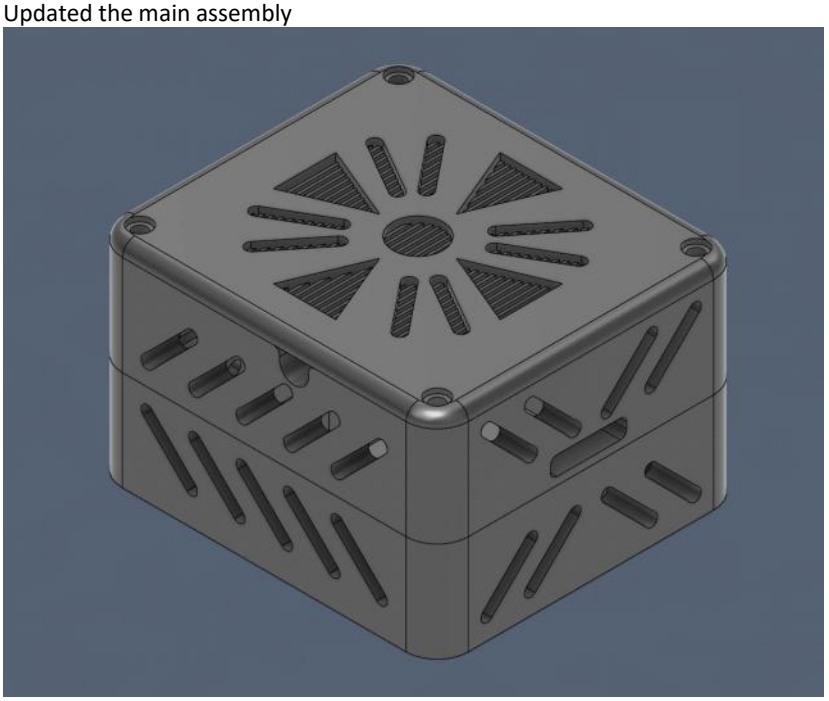


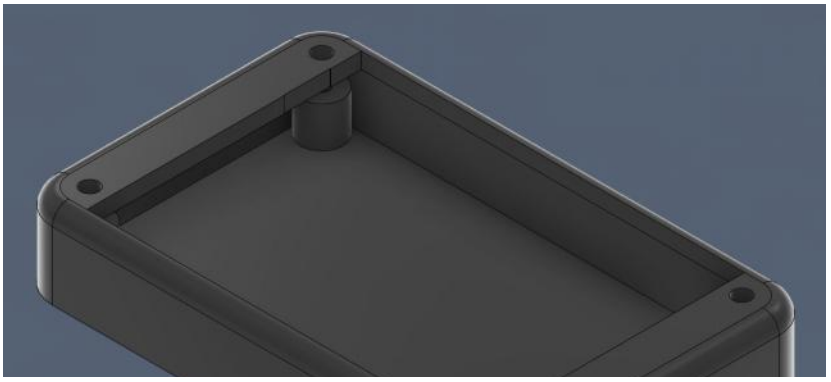
Rotation Motor Controller

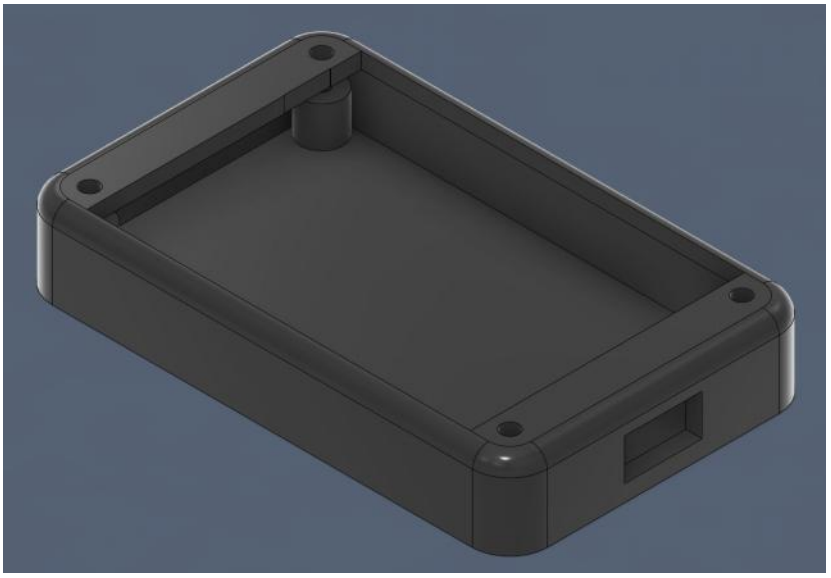


12/22

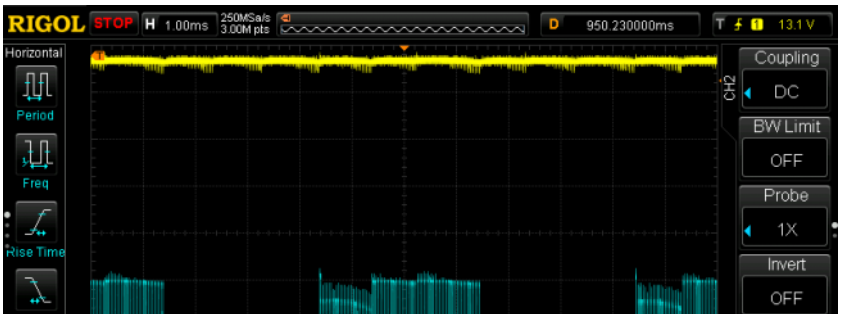
- ~ 3 hours
- Made rev2 of the motor controller case
- Made a case for the LCD
- Updated the main assembly



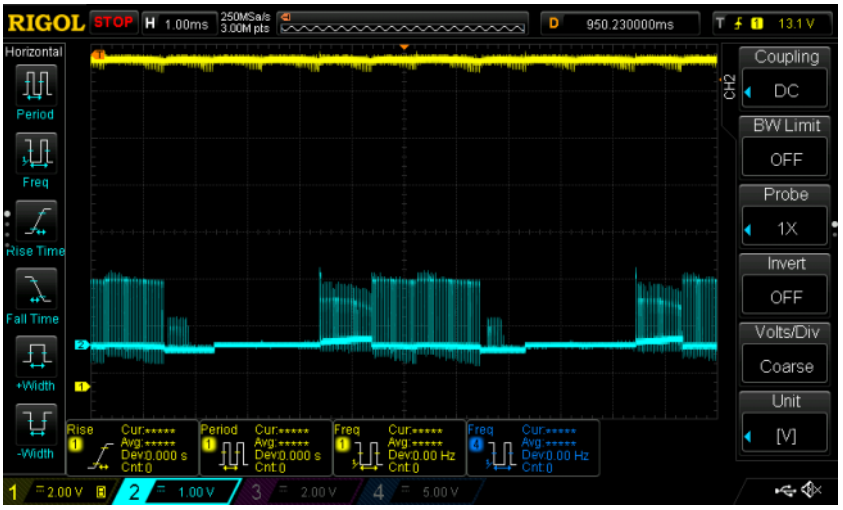




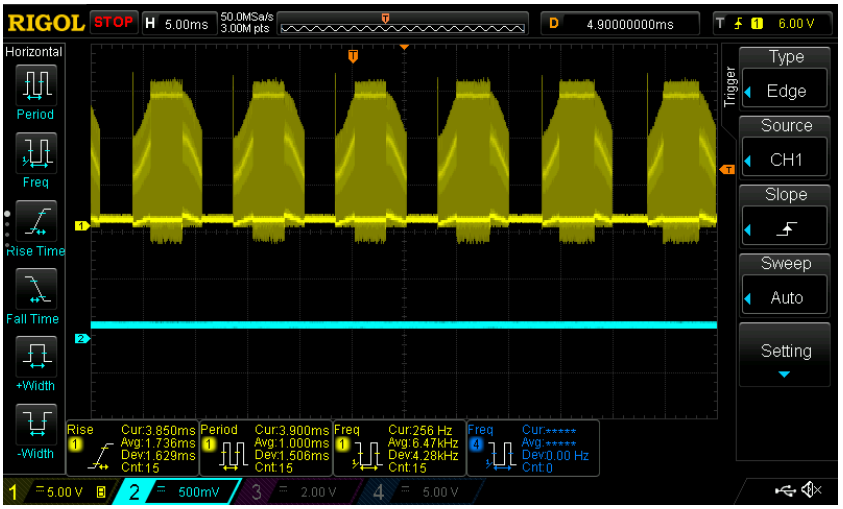
12/23
~ 2 hours
- Looking into motor controller



Our Motor Controller
No load
PWM set to 25 (out of 255)
With the new capacitors added



No load
PWM set to 25 (out of 255)
With the new capacitors added



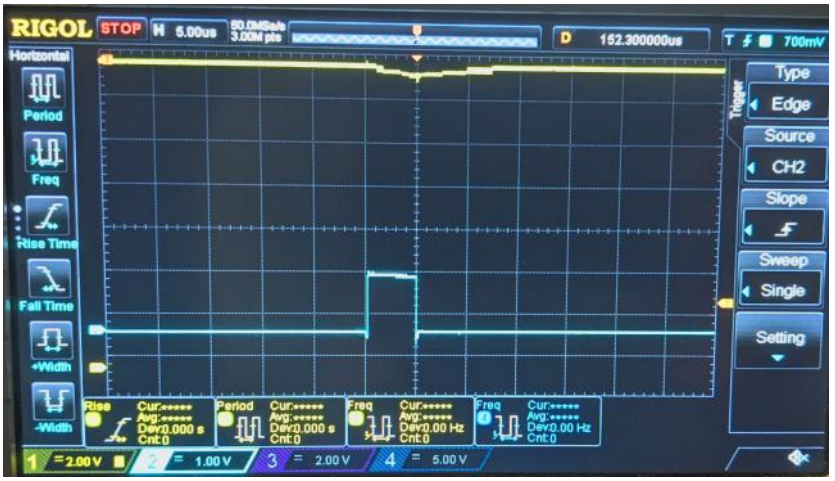
Our Motor Controller
No load
PWM set to 25 (out of 255)
With the new capacitors added

Our Motor Controller
With load & Stalled
PWM set to 25 (out of 255)

With the new capacitors added

Commercial Motor Controller
No Load
PWM set to 10%

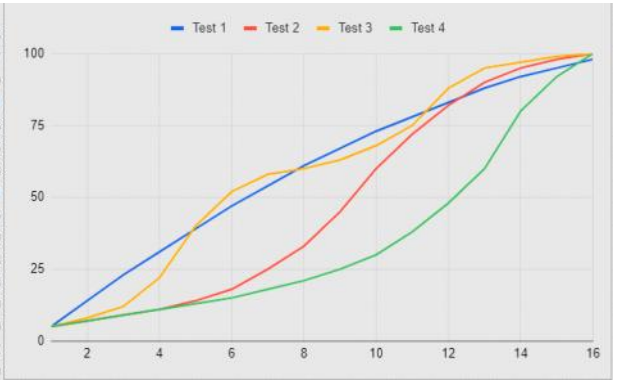
Commercial Motor Controller
With load & Stalled
PWM set to 10%



12/26

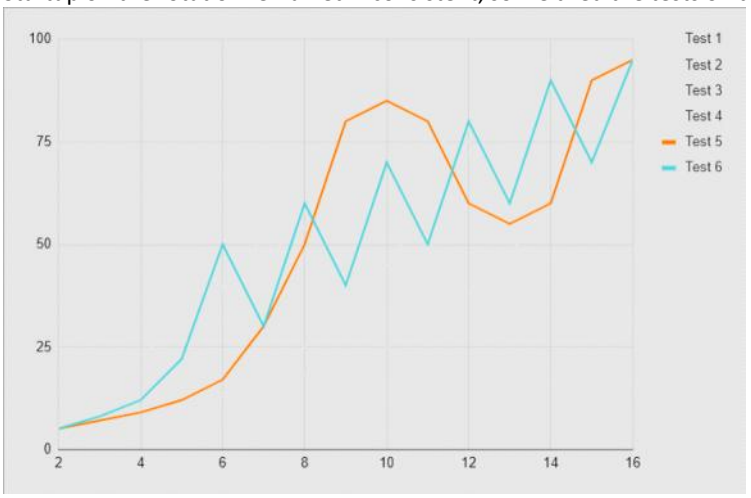
- ~ 6 hours
- Created a control register for the acceleration profiles to determine if one profile yields higher startup success
- Test 1 is our initial profile based on a poly-numeric convex function
- Test 2 and 4 are variations of an S-curve
- Test 3 is similar to a step function having 3 distinct levels: Base, Mid, and High

Step	Test 1	Test 2	Test 3	Test 4
1	5	5	5	5
2	14	7	8	7
3	23	9	12	9
4	31	11	22	11
5	39	14	40	13
6	47	18	52	15
7	54	25	58	18
8	61	33	60	21
9	67	45	63	25
10	73	60	68	30
11	78	72	75	38
12	83	82	88	48
13	88	90	95	60
14	92	95	97	80
15	95	98	99	92
16	98	100	100	100



12/27

- ~ 5 hours
- Added 2 more tests for the swerve
- Startup on the rotation remained inconsistent, so we tried the tests on the drive motor



- Unfortunately the drive motor did not work - the lower mosfet giving out during our testing

12/28

- ~ 3 hours
- Moving over to the arm servos
    - Wrote initial code for controlling servos including their offset bound limits

12/29

- ~ 6 hours
- Updated GUI
- Created wires for the arm, and created the wire for the Orange Pi
- Arm Signal Wire Colors: Grabber - Blue; Wrist - Green; Center and Base - White
- Central Connector Pinout
-

GND	7.2V	Wrist	GND	7.2V	Grabber
GND	12V	Base	GND	12V	Center

- Tuned the ranges (slamming into the table a few times)

12/30
- ~ 2.5 hours
- Created an acceleration profile for the servos to ensure smooth movement from the initial position to the target position

12/31
- ~ 5 hours
- Fixed the servo control code and tuned the acceleration profile for the servos

---------- FRC Competition Season ----------

3/29
- ~ 3 hours
- Connected SparkMAX controllers to the rotation motors on all four swerve modules
- Created and connected wires for swerve rotation magnet encoders

4/3
- ~ 1.5 hours
- Testing SparkMAX control through FPGA PWM signal
    - When no load is present the motor spins controllably at the expected speeds
    - Under any load (friction of the swerve rotation) the motor spins uncontrollably. This may be due to the FPGA remaining in the acceleration state, which causes the motor to speed up until it reaches its maximum speed.

4/6
- ~4 hours
- Working on swerve rotation with the SparkMAXs

4/11
- 2 hours
- Got motor rotation functionality, but the overshoot was very bad, approx 400 / 4096 or 35 degrees from the intended target

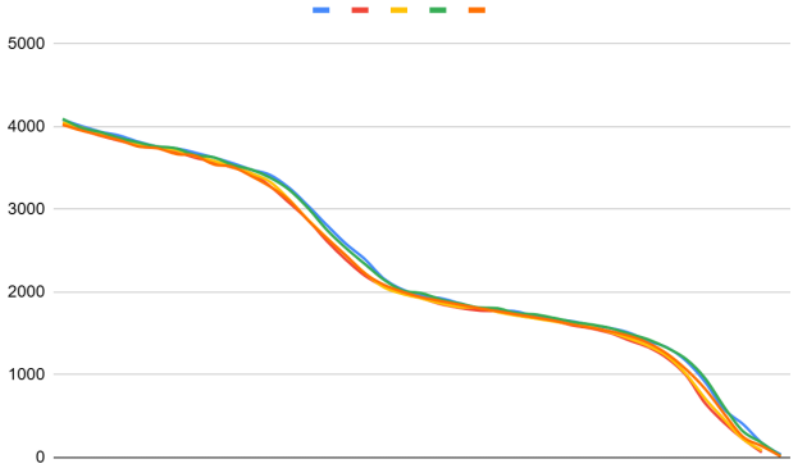
4/13
- 4 hours
- Started implementing a trapezoidal PID tuning

4/14
- Started to integrate PID into the subsystem
    - Starting to update register file

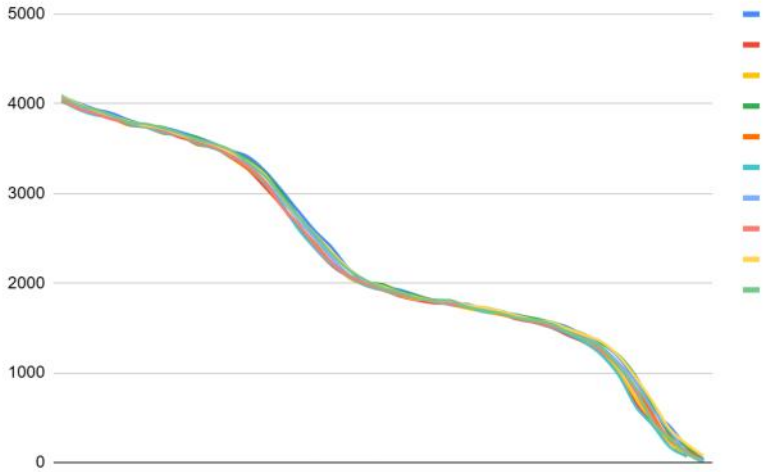
4/17
- Tested PID tuning
- PID control loop is working, however the module is overshooting the intended angle

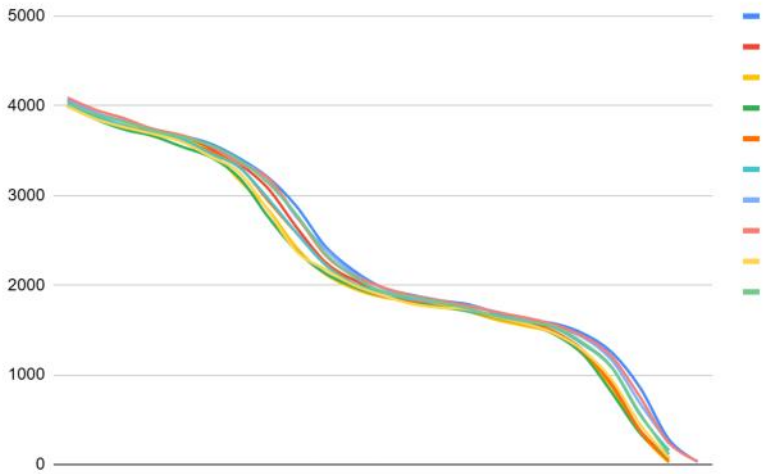
4/19
- Implemented code to manually control the pwm value to the swerve, so we can plot the resistance profile of each module

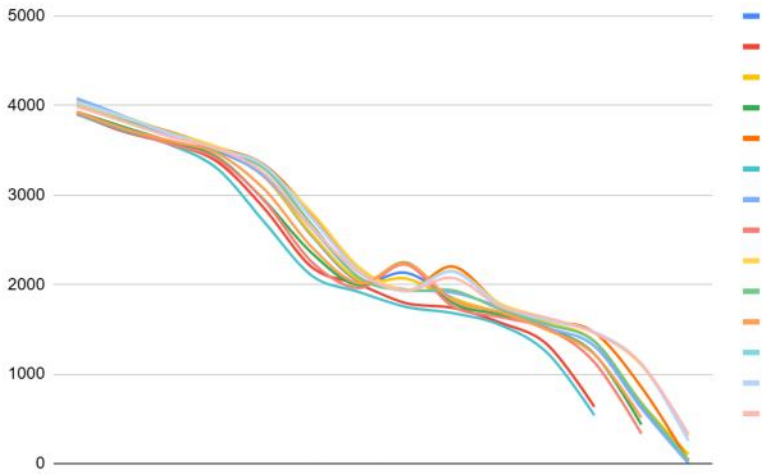
4/20
- Resistance profiles:
- At 10:



At 15 (3 distinct speeds):







If the line was equal to y = x that would indicate a consistent resistance, however as seen above there are two distinct "zon es" were the resistance is higher. Note that as the speed increases more noise is introduced.

By applying a "boost" coefficient to the PWM ratio at the higher resistance zones, we hope to smooth out the curve by applyin g additional power. This should make the curve more linear which effectively, increases our tuning capabilities with the PID co ntrol system. Note: due to the non-linearity of the current system, tuning the derivative and integral value for all instances is generally ineffective; hence the need for the "boost" coefficient to establish general linearity.

The "boost" coefficient is calculated based on the difference between the highest magnitude Angle Rate of Change and the Angl e Rate of Change at index n. Essentially we are calculating the Angle Rate of Change error between the higher resistance zones and the least resistant zones to determine the magnitude of the "boost".

4/22

- Unfortunately the boost coefficient, while theoretically effective only increased the variance of the modules.
- Started to increase the power value and reintroduced the derivative term to help.
- At the moment we see the module get stuck before reaching the target angle, to address this we may increase the p -value or altering the control profile to reduce the reduction in speed when in the DECEL state.

4/23

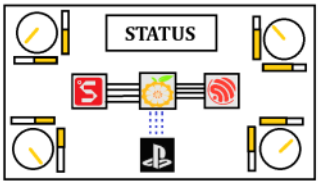
- Connected the Orange Pi to the new ESP32 Dev kit board. The ESP connects to the Orange Pi via the VSPI ports.
- Reconnected PS4 controller and tested inputs, data seems to be transferring correctly, but something with the controller dead zone profile is not working

4/24

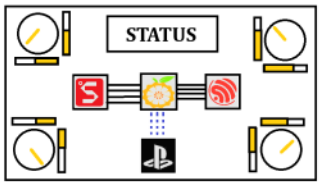
- Looking into Nextion user interface to display status messages on the robot

4/25

- Planning overall UI interface
- Looking into PyQt since Nextion designer interface is difficult to work with
- PyQt also works with a larger monitor instead of the small Nextion Screen



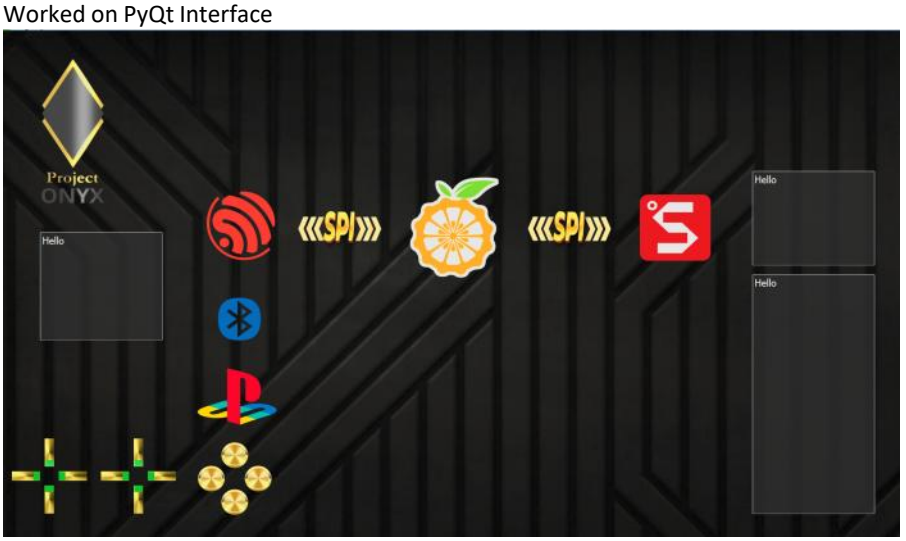
-



-

4/26

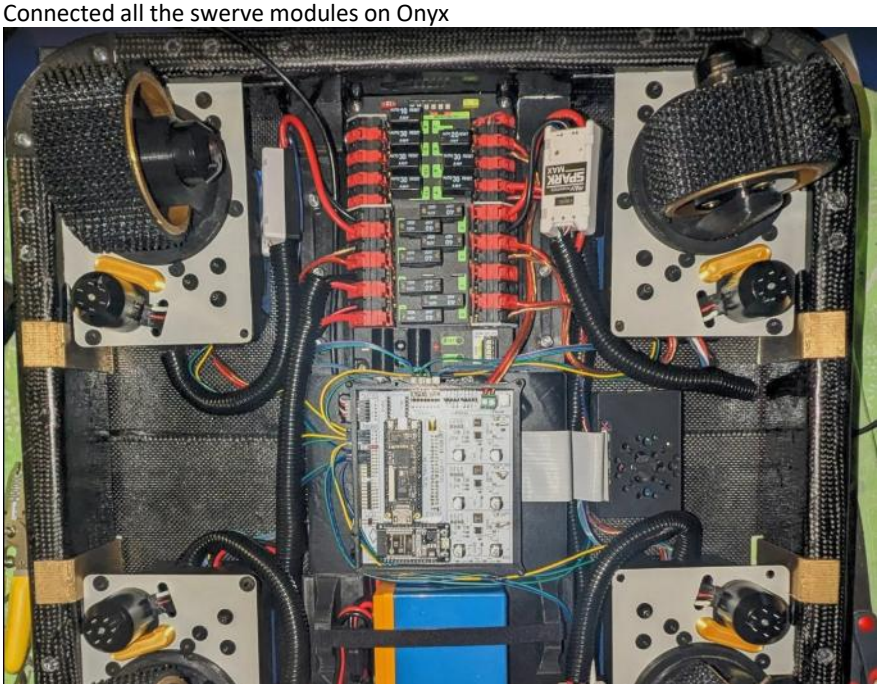
- Worked on PyQt Interface



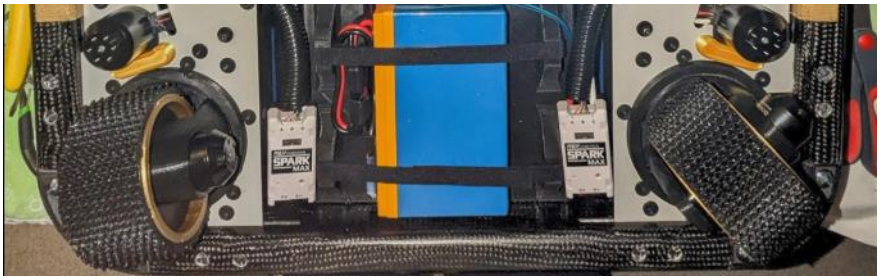
-

4/27

- Connected all the swerve modules on Onyx

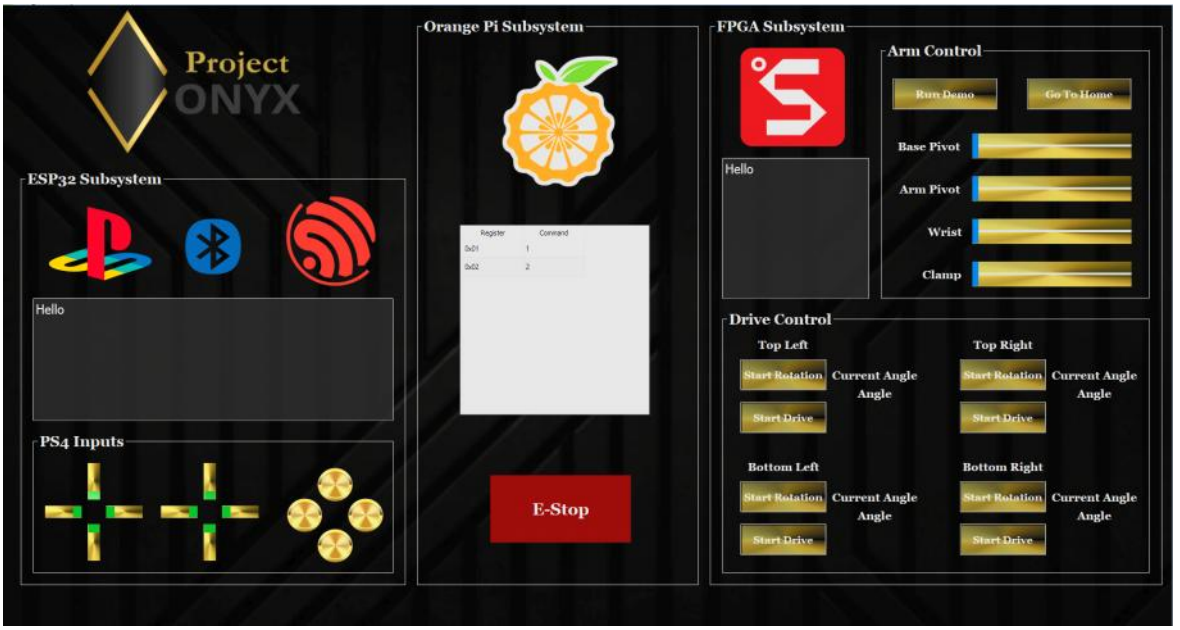


-



4/28

- Worked on the Ui interface



4/29

- Named elements in the PyQt ui
- Started to work on the top level reg file

4/30

- Condensed the register and top files
- Reinstated the remaining swerve modules to the top file
    - Added the swerve drive modules to the top file

5/1

- Working getting the swerve modules to work

5/2

- Fixed swerve enable registers
- Moved position of swerve rotation connections
    - Note: when the FPGA runs the placing software there is some variability, causing us to repeatably run the placer until it successfully finds a routing solution
- Got most swerve motors working
    - Top left swerve rotation's pinion is disconnected from the gear
    - Top right rotation's PDP pins or breaker is faulty

5/3

- Worked on GUI functionality
    - Motor controls works for each module, the e-stop button has been connected and PS4 inputs are properly displayed