

---

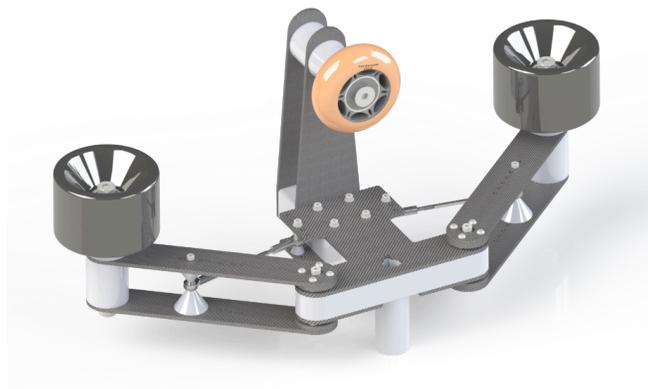
*San José State University*

***Spartan Superway***

**A Solar-Powered Automated Transportation Network:  
Switching Mechanism for the Bogie**

Kris Gonzalez, Emilio Cruz, Jesse Kavros, Ian Geiger, Jake Espinoza

Yanzhen Chen, Jose Bravo, Andrew Poli, & Jingwen Tan



*Charles W. Davidson College of Engineering*

ME 195B Section 4: Senior Design Project II

Advisors: Burford Furman & Ron Swenson

Written: May 21, 2021



**Team Member Roles**



Kris Gonzalez (**Project Lead**)

[Krisgonzalez117@gmail.com](mailto:Krisgonzalez117@gmail.com)

Electrical Team



Yanzhen Chen

[Johnathancyz@gmail.com](mailto:Johnathancyz@gmail.com)

Electrical Team



Jose Bravo

[Jose.m.bravo.04@gmail.com](mailto:Jose.m.bravo.04@gmail.com)

Electrical Team



Jesse Kavros

[jesse.kavros@sjsu.edu](mailto:jesse.kavros@sjsu.edu)

CAD Team



Ian Geiger

[ian.geiger01@gmail.com](mailto:ian.geiger01@gmail.com)

CAD Team



Emilio Cruz

[ekcruz07@gmail.com](mailto:ekcruz07@gmail.com)

CAD Team



Jake Espinoza

[Jt.espinoza9@gmail.com](mailto:Jt.espinoza9@gmail.com)

Programming Team



Andrew Poli

[Andrewpoli42@gmail.com](mailto:Andrewpoli42@gmail.com)

Programming Team



Jingwen Tan

[Barrytan21th@gmail.com](mailto:Barrytan21th@gmail.com)

Programming Team

## 1. **Abstract**

- a. What you have done (one sentence)
- b. Why you do it – state the significance/impact of your work (one or two sentences)
- c. How you do it – state what challenges you were facing and how did you solved
- d. the problems in which methods/technologies. State the unique part of your work (one paragraph)
- e. Major/important results – state the most significant results and findings.

*Keep in mind that the purpose of the abstract is to give a reader a general idea of your work without reading through the entire report. Make sure that your abstract is a standalone component and is succinct. Typically, an abstract is the last component of the report that you write.*

### **Abstract**

The 2020-2021 Superway Switch-Arm team developed and tested an adjustable switching mechanism that holds over 543 Newtons of force and rotates 0-180 degrees for a Bogie to switch in a small-scale track by contacting an overhead structure known as a passive Y-switch (PYS). Since avoiding objects by turning and reaching various locations by switching to different routes is the basic requirement of a transiting system, the development of Switch-Arm became the necessity that assists Superway to properly operate in cities.

Using Solidworks, the Mechanical/CAD team came up with multiple concepts and went through six iterations of design for the switch arm, then ran multiple Finite Element Analysis (FEA) for the final design, which is an adjustable-3-wheels boomerang. After the modeling process, the team manufactured the arm with carbon fiber for the physical testing in the 10-meter

track. The integration with Bogie was a major obstacle due to the uncertainty in the shaft's length, but it's solved with the beta prototype. The Electrical team created and fabricated four versions of the printed circuit board for the switch arm using KiCAD, OSH Park, JLCPCB, and applied bootloaders to set the microcontroller for coding. The team was stuck in the bootloading process for a while but eventually debugged the uploading issue by modifying the resistance and layout. The Programming team has built a state machine in the Arduino IDE to perform homing, switching and communicating. Board-to-board communication was the major challenge due to the limitation of the pin's availability, but the issue was solved by adjusting the code for the logic signal to fully operate in one pin.

Throughout the project, our team has manufactured two working switch arms that can smoothly handle torque and force from the current version of the track. Since we designed our arms to be fully adjustable in angle and length, it can be implemented and tested in the future despite changes to the track. We also created a compact PCB that can handle all switching requirements and maximize the additional space needed for future teams. As for programming, we developed a state machine that is ready to integrate and a side-testing program that can easily adjust the switching angle to meet a specific need. Building a basic prototype that meets all functions was the core milestone we reached, but performance based on the stability, unexpected situation and extreme environment are still unknown solutions to seek.

## **2. Acknowledgement (ME195B Final Report)**

- a. Sponsor(s)
- b. Professors, technicians, staff, other students, friends and relatives, etc.

### **Acknowledgement**

## **3. Table of Contents (ME195B Final Report)**

Includes titles of each section and subsection, and page numbers. Appears after Acknowledgement and before List of Figures. Nicely formatted.

	5
<b>Abstract</b>	<b>3</b>
<b>Acknowledgement</b>	<b>4</b>
<b>Introduction</b>	<b>7</b>
Problem Definition	7
Project Objectives and Specifications	9
Current Status	9
Team Work	14
Gantt Chart	15
<b>Theoretical Background</b>	<b>16</b>
<b>Senior Project Design</b>	<b>17</b>
<b>Analysis and Documentation:</b>	<b>32</b>
<b>Fabrication and Assembly</b>	<b>37</b>
Fabricated Parts	37
Purchased Parts	40
Challenges	41
Final Assembly	43
<b>Testing Results and Analysis</b>	<b>43</b>
Testing Results	43
Analysis of the Results	47
<b>Conclusion and Recommendations for Future Work</b>	<b>49</b>
<b>References</b>	<b>54</b>
<b>Appendix</b>	<b>55</b>

**4. List of Figures (ME195B Final Report)**

Includes caption of each table and page number that is consistent or matches the figure in the text.

**List of Figures**

**5. List of Tables (ME195B Final Report)**

Includes caption of each table and page number that is consistent or matches the table in the text.

**List of Tables**

**6. Chapter 1 - Introduction (ME195A report #1)**

**a. Problem Definition**

Describe the motivation and background for the problem e.g., market needs, community needs, new technology needs, automation needs, energy saving needs, environmental protection needs.

**b. Project objectives and specifications**

- i.** How would you tackle the problem differently from the existing methods
- ii.** Which technologies or principles or methodologies you are proposing?
- iii.** What specifications and results to be expected (e.g., including materials you choose, geometry, accuracy, speed, weight, capacity, constrains)

**c. Current status (literature review)**

- i.** How other people solve this problem in the past or currently
- ii.** Which technologies they used
- iii.** What kinds of results they have achieved
- iv.** What issues or disadvantages of their methods

Please don't forget to put the reference/source when listing each work

**d. Significance of your project and its uniqueness and challenges**

**e. Team work**

- i.** Indicate the responsibility of each team member on the project
- ii.** How does he/she conduct the task(s)/project.

*It would be ideal to have a multi-disciplinary team (e.g., the first member is in design, the second member is in mechatronics, and the third one is in thermal and fluid area).*

**f. Gantt Chart**

*A Gantt chart is a type of bar chart, developed by Henry Gantt, to illustrate a project schedule from the start date to finish date. Terminal elements and summary elements comprise the work breakdown structure of the project.*

## **Introduction**

### ***Problem Definition***

As increasing investment and development occurs in urban areas around the world, transportation problems resulting from an overwhelming population bring more threats to society and the environment. The increased number of personal vehicles led to over 36,000 traffic-related fatalities in 2019 in the US; the Bay Area averages 1.25 deaths per day in traffic accidents (NHTSA, 2020). Being actively focused on the road for both drivers and pedestrians are extremely important to avoid the increase in incidents, but as the disruption from surroundings grows (such as roadside advertising and interactive technology) it becomes harder for people to focus on the road. This precautionary situation brings a need for creating a new transportation system with safe operation. The current system people use is mainly powered by fossil fuel, which creates another problem in pollution. Although California is a pioneer for leading an environmental revolution, the annual pollutant level of the Bay Area still ranks 5th over 204 metropolitan areas in the nation (American Lung Association, 2020). On one hand, the pollution from the traffic can cause health issues, such as stroke, lung cancer, and chronic respiratory diseases. However, it also accelerates global warming by contributing over 36% of the total greenhouse gas emission in the bay (Sukaran, et al, 2010). For a short-term solution, the current system does properly move people and goods to the right destination (such as transiting from home to work or from factory to store), but in the long run, the traditional transportation will definitely push health and environmental issues to an irreversible stage.

With the overwhelming growth of personal vehicles and the need for public transportation, congestion becomes another major issue that not only decreases transporting efficiency but also generates an unhealthy impact on the regional economic growth from increasing waste of time and resources. In the Bay Area, there are over 6,000,000 vehicle trips in a single day (Vital Signs, 2020). The majority of the trip happens during the early morning and late afternoon when people are all headed into the highway for work. Especially in high traffic areas such as I-80 eastbound from Cesar Chavez to Treasure Island or US-101 southbound from Mountain View to Santa Clara, it's common to spend twice or even more time traveling the same distance than in regular hours. Since the economic boom in 2000, the congestion delay has increased by 65% compared to a 12% growth of job opportunities in the Bay Area (Vital Sign, 2018). The time and energy congestion wastes are slowing the efficiency and damaging the economy. Furthermore, for traditional transportation, such as VTA light rail or buses, they can also be time-consuming when passengers have to wait over 5 minutes for the next scheduled transit for an experience of loading and unloading at each station. Since traditional public transportation always results in a crowded environment, it can become an extremely dangerous place during an urgent situation, such as the pandemic; people have no way to maintain a minimum social distance within a limited-share space. To solve all these problems, the system requires immediate reformation. Silicon Valley is one of the most influential areas around the world, pioneers in the Bay Area need to continue leading sustainable development and take urgent action to fit the increasing social need for a new transportation system, such as Automated Transit Networks (ATNs).

ATNs, which can also be addressed as personal rapid transit (PRT), are a non-scheduled and fully-customized public transit that drives from origin to destination without intermediate stop or transfers (Furman, 2014). Thanks to the feature of a fully-customized and call-to-arrival experience, podcars cannot only boost the transiting efficiency by avoiding unnecessary pauses but also providing a safe and private space for passengers to avoid health infection or physical threat. Since the commonest threat in driving is the accident caused by a vehicle collision, the fully automated podcar is designed with a 24/7 monitored guideway system to secure safety from all unexpected emergencies. To completely upgrade the current transiting system, solving pollution is an essential goal. With new renewable methods for electric generation, the zero-emission ATN becomes an effective solution to make a clean revolution. This desire for

clean energy transportation led to the birth of the Spartan Superway, a solar-powered ATN for a sustainable future. To successfully build a safe and functional system, the project brings young engineers together to design a 10-meter scaled prototype that allows an electric-powered Bogie to steadily travel and switch on a wooden guideway under a dynamic monitorization. It is an arduous challenge for the team, but it's also the duty and responsibility of the new-generation to secure their future from contaminated transportation.

### ***Project Objectives and Specifications***

The objectives for the Switch-Arm team were to design and manufacture a device for the Spartan Superway that would allow the Bogie chassis to safely change directions along the track in under three seconds at the operating speed. This device was originally planned to fit within the 6x8x6 inch space that was provided by the other teams at the beginning of the spring semester. However, due to changes in all teams' designs, the size objective had to be revised. The Switch-Arm must interact with the passive Y-switch developed by the track team. Interaction with the switch must produce a turning force that is enough to overcome the friction of the wheels on the track. In addition to withstanding the turning force required, the Switch-Arm must provide enough of a moment to keep the Bogie from slipping off the track.

The Switch-Arm team's final design exceeds the original space objective, but meets the current requirement and was tested using both a motion study and a physical prototype on the 10 meter track in the SPARTAN Superway Design Center. The team was able to complete fabrication and assembly of two physical prototypes and test them in a physical setting.

### ***Current Status***

ATN is a developing category of public transportation. The way ATNs differ from traditional modes of public transportation like busses is through their personalized and private transportation units. Each trip is personalized for each rider, ATNs adjust the path taken based on real-time data to take the most efficient route. This differs from a bus which will stop at each bus station where someone is waiting or needs to get off. Because ATNs are personalized for each rider, the car will be able to go directly to their destination, skipping intermediate stops.

The criteria to be classified as an Automated Transit Network is somewhat niche since there are currently only five systems around the world that can be classified as an ATN. These five are in West Virginia (Figure S1), Rotterdam (Figure S2), Abu Dhabi (Figure S3), London

(Figure S4), and South Korea (Figure S5). However, these examples do not represent the full potential of what an ATN system can provide.



The Morgantown PRT (Personal Rapid Transit), shown in Figure S#, is a rail system designed to transport students around the West Virginia University campus. It was developed in the early 1970s and has been running since 1975. It can provide transportation to five stations along an eight-mile-long track. It is free to ride for students, and fifty cents for others. Each car can seat eight passengers and comfortably carry about fifteen passengers (West Virginia University, 2020).



The Rivium ParkShuttle has been in operation since 1999. It has undergone multiple revisions, currently on its third. Shown in Figure S# is the second generation of the Rivium system. The ParkShuttle differs from the other transit systems in this list because its route is not exclusive to itself. This differs from the goal of the Spartan Superway and other ATN systems whose goals are to reduce traffic congestion. They are also much larger, accommodating more passengers. The first generation was smaller and more similar to the other systems allowing eight seated passengers and two standing passengers. The second generation is much larger, allowing up to 24 passengers.



Masdar City is a city project in Abu Dhabi to create a carbon-neutral, car-free city. The Masdar City PRT system, shown in Figure S#, has been open to the public and running since 2010 and has a reliability rating of 99.9%. It was designed and developed by 2getthere, which is the same company that developed the Rivium Park shuttle. However, the Masdar City PRT system does not meet the potential that an automated system can provide due to it only having two stations that each vehicle travels to.



The ULtra PRT at the London Heathrow Airport is a PRT system that transports passengers and their luggage from the Business Car Park and Terminal 5. On the Ultra Global PRT website, it claims that the ULtra PRT can dispatch up to 100-120 vehicles per hour. This provides on-demand shuttles from passengers with very little wait time regardless of demand. Like the Masdar PRT, the PRT at Heathrow Airport is very limited, only transporting passengers between 3 different stations.



The SkyCube PRT at Suncheon, South Korea is a PRT system that transports passengers between two stations over a length of 4.6 km. The SkyCube system, much like the Masdar PRT, only transports passengers between two locations. The SkyCube PRT travels through a nature

reserve, making the SkyCube more of a scenic attraction or proof of concept, rather than a functional service for everyday use.

The Spartan Superway Project at San Jose State University has been a senior design project since 2012. Since then, students have been working on developing the Superway's design. Spartan Superway teams in the past have designed and manufactured full scale, half scale, and small scale prototypes. In 2015, the Full-Scale team was able to create a full-scale prototype. However, they were unable to make progress on a switching model or propulsion (Ornellas, et. al.,2015). In Figure X, the progress of the Full-Scale team up to 2015 is shown with their Model of the Pod as well as a rail system.



Previous Spartan Superway teams have done iterative design work on the track and switching arm systems. Major problems encountered by past teams have been an inability to switch tracks at the desired operating speed and load and failures of the drive mechanism and sensors associated with the Switch-Arm. The 2019-2020 team made improvements to the Switch-Arm created by previous teams in several areas. Previous designs for the Switch-Arm had poor contact between the track and the switching wheels. This was addressed by adding adjustment points to the Switch-Arm assembly. These points allowed individual wheels to be adjusted independently of each other and their mounting brackets. The 2019-2020 team also determined that using 2 wheels on each side of the Switch-Arm made the entire Bogie more

resistant to twisting while crossing tracks. The Switch-Arms developed previously also used two contact points on each side, one at the top and the bottom of the Bogie (White, 2020).

### ***Team Work***

The 2020-2021 SPARTAN Superway Switch-Arm subteam is made up of three project teams: Mechanical, Electrical, and Programming. Each project team has separate design tasks related to the overall success of the Switch-Arm design. The Mechanical team, made up of Emilio Cruz, Jesse Kavros, and Ian Geiger, was responsible for the physical design of the Switch-Arm mechanism and the production of CAD models and part drawings. The Mechanical team worked in conjunction with the Electrical team to specify the hardware needed for driving the Switch-Arm. Collaboration between these teams was needed to reserve space in the design for electronics and to select a suitable motor for the Switch-Arm. The Electrical team, made up of Kris Gonzalez, Jose Bravo, and Yanzhen Chen, was responsible for the selection of electronic components and their interface with the rest of the electronic subsystems of the Bogie. The Electrical team also worked to reduce the overall electronics footprint of the Switch-Arm by creating a prototype PCB design with a small footprint. Their selection of the microcontroller (MCU) and the motor driver was made while keeping the programming requirements in mind. The Programming team, made up of Jake Espinoza, Andrew Poli, and Jingwen Tan, was responsible for writing the code that controls the Switch-Arm. This code was designed to integrate with the code being used for the other MCUs of the Bogie.



## Theoretical Background

The switching mechanism needs to overcome the moving Bogie's impact force when hitting the PYS mechanism while turning. To find the force that the Switch-Arm must withstand, Equation 1 must be used:

$$\text{Equation 1: } F_{avg} * s = \frac{1}{2} * m * v^2$$

The dynamic energy was converted to work, in which  $s$  is the deformation distance. The deformation deceleration distance is very important and is the key to limiting the force acting on the Bogie and the theoretical value for the deformation distance used here is

$$s = 0.02m$$

With the parameter all the other group agrees on in this phase, the scale model robot should be operating in the speed of

$$v = 1 \text{ m/s}$$

Radius of the PYS mechanism is

$$r = 3.45 \text{ m}$$

Mass of the scaled Bogie is

$$m = 70 \text{ kg}$$

The theoretically calculated impact force is  $F_{avg} = 1750N$

However, the PYS mechanism has been designed with a glanced angle to reduce the impact force, with glanced angle

$$\theta = 14.96^\circ$$

And theoretical impact time is about

$$t = 0.03s$$

The theoretical value of the force that the switching mechanism should overcome is calculated based on Equation 2:

$$\text{Equation 2: } \bar{F}_{avg} = m * \frac{2*v*\sin(\theta/2)}{t} = 543N$$

FEA was performed to verify the safety of the Switch-Arm design. These studies were performed on the entire Switch-Arm assembly using the loading estimates from the calculations above. Analyses were performed to find the displacement, vonMises stress distribution, and factor of safety. The results of these studies can be seen below in Figures C3-C6. The location of maximum von Mises stress coincided with the location of the smallest factor of safety for the assembly. The motor output shaft and location where the motor is mounted experience the highest stress of any part in the assembly. Since these are the critical locations for stress, they will be the first area to be strengthened should the physical prototypes fail.

## 7. Chapter 3 - Senior Project Design (ME195A report # 2-3)

- a. **Mechanical:** Illustrate how many design concepts and options the team had come up at the early stage of ME 195A, and how the team selected the most optimal solution through:
  - i. Analysis of pros and cons of all the available options, and/or
  - ii. preliminary experiments (if any), and/or
  - iii. computer simulation and/or FEA, and/or
  - iv. theoretical calculations
- b. **Electrical:** (if the project includes Mechatronics) use of microcontrollers and electronic components discussed and justified.
- c. **Programming:** (if the project includes Mechatronics) Block diagram of electronic circuits included.

### Senior Project Design

#### ***Mechanical:***

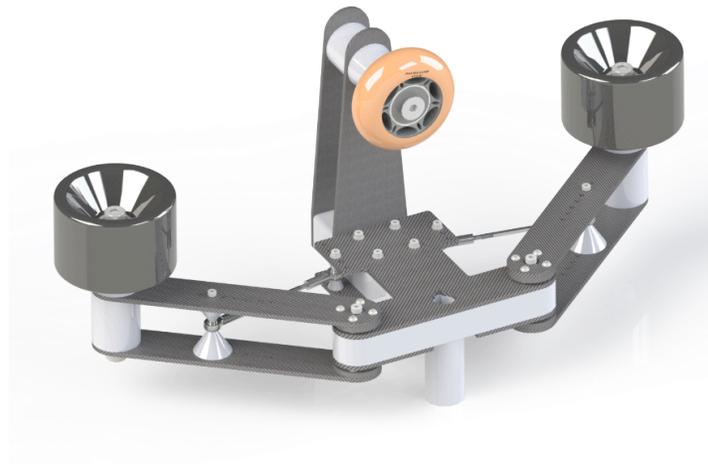
Leading up to the final design, many other design iterations were proposed and discussed. Figures MX through MXX in the appendix show the progress of design concepts as more information became available about the developing track and Bogie designs. The initial concepts in Figure MX were based on the assumption that the track would be of an I-beam construction, instead of the flat track with PYS that is the current design. Once more communication between the Switch-Arm and Track teams was established, more realistic concepts began to take shape, with some interesting electro-magnetic applications, as shown in Figures M# and M#. The electromagnetic Switch-Arm concept in Figure M# made its way into a CAD model, as shown in Figure M#, but was deemed too weak for the expected system forces. The fully electro-magnetic concept shown in Figure M# never made it past the initial ideation stage due to lack of research and confidence in the concept, but may be worth further research in the future.

After more brainstorming, the team came up with the design shown in Figure M#. This initial design leveraged a rubber wheel sandwiched between two carbon plates, all of which were mounted to a high-torque servo. Leveraging a single wheel out translates to a large moment being put onto the servo, thus needing an expensive and durable servo. The need for stronger materials was also a concern for testing and not wasting money on pieces that wouldn't make it to the final design iteration. Figure M# was the next iteration of the initial design. The overall

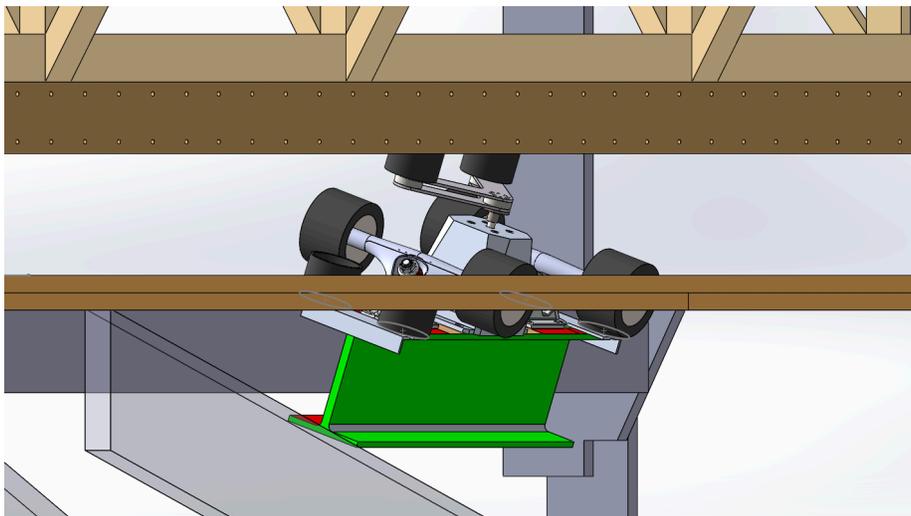
moment was canceled out with the addition of a second wheel. After eliminating the torque required, the team then found that with the wheel mounted between the carbon plates gave unwanted interference between the Switch-Arm mechanism and the PYS. Raising the wheels above the carbon plates allowed for no adverse interference with the PYS. The next decision was to add mechanical stops to reduce the number of wheels back to just one. Figure MXX shows the design with the leveraged wheel mounted further above the carbon plates. These iterations were deemed too weak and intrusive to the PYS. Thus, the “boomerang” design came to fruition, shown below in Figure M#. By the end of the fall semester, the team was ready to move forward with the prime “boomerang” design going into the spring semester. Following feedback early in the spring, the issue of safety was brought up, necessitating a re-think of the Switch-Arm design.



Through collaboration with the Track team, a design for the PYS and Switch-Arm interface was established. The main goal was to maximize the contact between the guide wheels on the Switch-Arm and the surface of the PYS while ensuring that in the case of the Bogie’s drive wheels slipping during a turn, the Switch-Arm would not lose contact with the PYS. This was accomplished by adding a third, vertically positioned guide wheel on top of the PYS, acting as a safety net in case the Bogie were to fall. In addition to the safety wheel, the “boomerang” shape of the arm was divided into two manually-adjustable arms connected to a base assembly. The arms are secured with turn-buckles attached between the base assembly and the arm by heim joints. This allows for movement and fine adjustability of the angle of the arms, which controls the distance between the center of the Bogie and the centerline of the guide wheels. This in turn controls how far the Switch-Arm tilts the Bogie during a transition over the PYS, which is vital for ensuring a smooth transition.

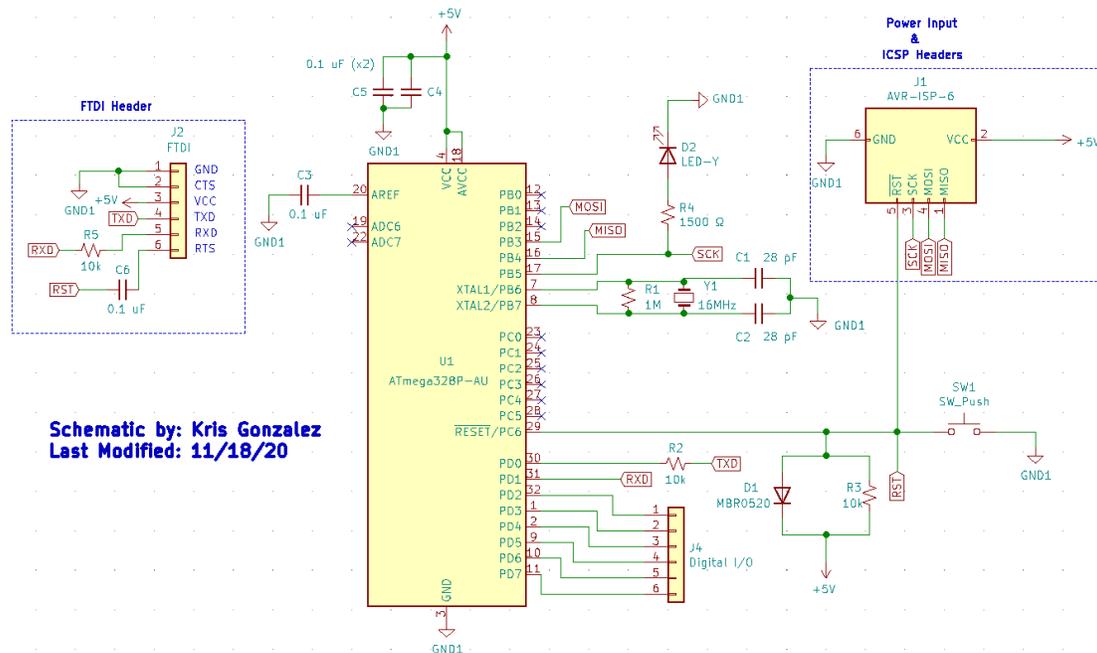


The final mechanical design for the Switch-Arm is shown above in Figure M#. The design consists of a stepper motor, 30:1 ratio gearbox that is mounted to the floor of the Bogie, and the adjustable Switch-Arm with horizontal guide wheels on each arm and vertical guide wheel suspended above. Advantages of this design include adjustability - making the machine easy to both fine-tune and control - and the added safety of the vertical wheel positioned on top of the PYS. With only one force direction, the previous “boomerang” design neglected the possibility of wheels slipping off the track, leaving the entire Bogie assembly without a safety net. Figure M# below gives an example of this occurrence. The disadvantages to the current design include the addition of moving parts and complexity of assembly, adding to the manufacturing cost and set-up time.



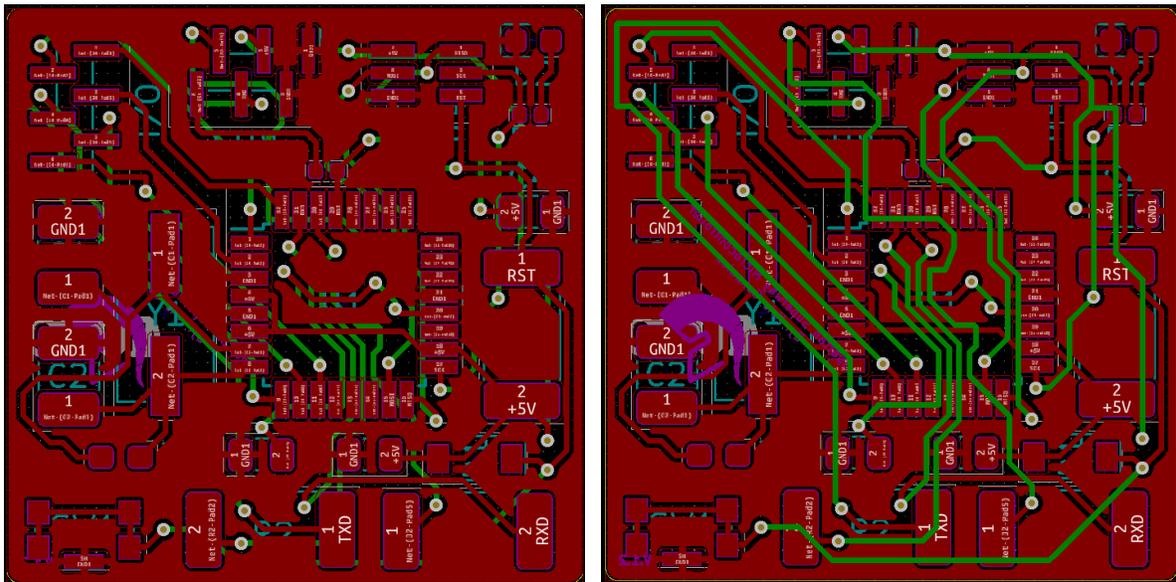
### Electrical:

Instead of buying an Arduino board online and using that as the foundation of the Switch-Arm, the team decided on creating a Nano-like device from scratch. The advantages of designing the exclusive board for the project include integrating all the electrical components into a single print circuit board for saving space and avoiding tanglesome wirings. Because the initial objective of designing the board was to have a PCB with similar functionalities with the Arduino Nano board and integrate all of the electrical components that need to be used in the Switch-Arm, the design of the Switch-Arms board was taken the inspiration of the Arduino open-source schematics. The PCB has been through four times of design modifications and three times actual fabrications. It started from the first version of the Switch-Arm schematic shown below that it has fewer digital and analog pins, no voltage regulator, and a lack of a USB connection to the FTDI protocol, to the final design that has most of the functionalities that a Nano has, the team spent a lot of effort on optimizing the board. Figure E# below is showing the schematic of the first version Switch-Arm PCB.

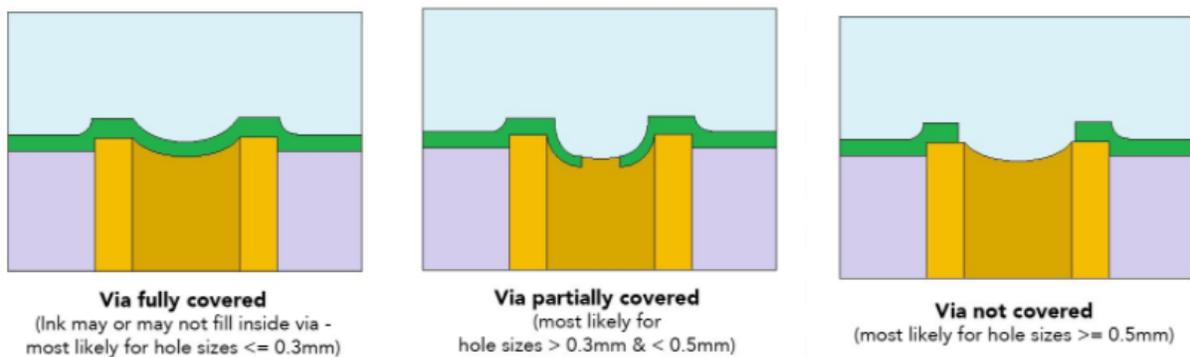


The microcontroller design for the Switch-Arm was designed in the PCB software known as KiCAD. Figure E# shows the front layer and the back layer of the PCB. Note that the front copper plane is used for all of the electrical grounds (GND). For this design, a back copper layer was not used but was still covered in the substrate for the vias and silkscreen. The track widths

were all kept at a size of 0.250 mm (or 9.84 mils) and the diameters for the vias size and drill holes were 0.6 mm and 0.3 mm, respectively.



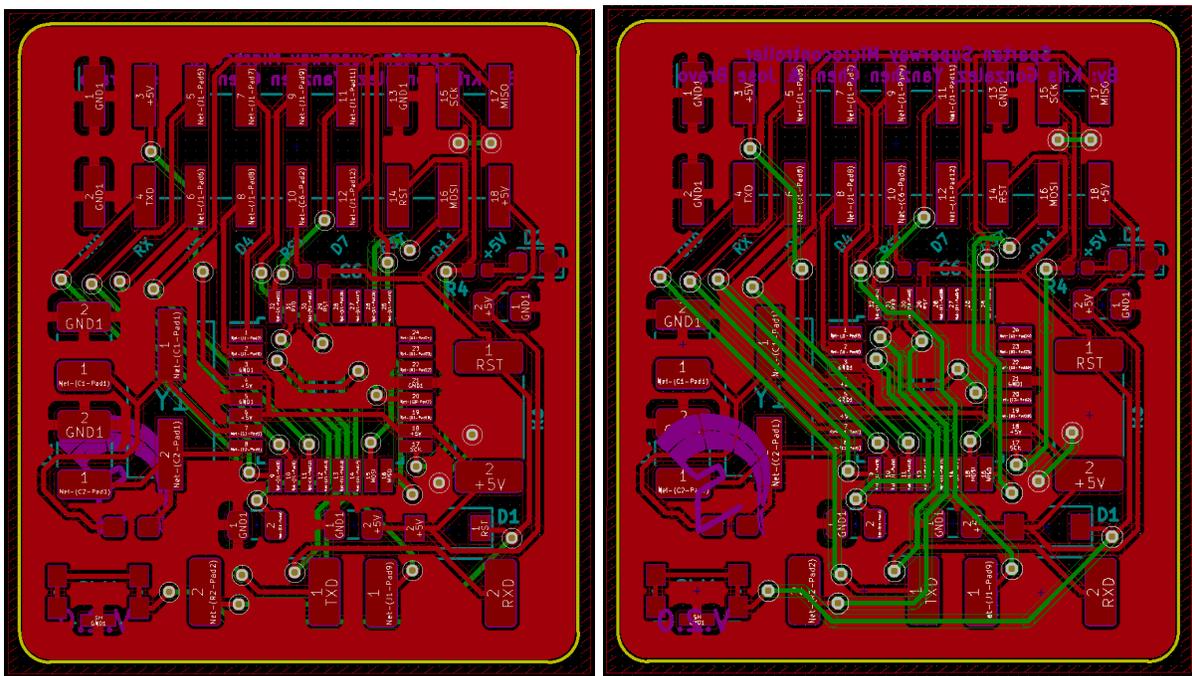
The drill holes for the vias were changed from the default 0.4 mm to 0.3mm because as the soldermask is applied to the PCB, the vias will be more likely to be protected from potential soldering shorts. Figure E# describes how a smaller via drill diameter can be protected from a short cause from improper soldering or other causes. This method is often referred to as tented vias, where the soldermask covers the exposed copper on a hole. Ideally, all via drill holes should be covered, but too large of a drill size can lead to a via only being partially covered or not covered at all with the solder mask.



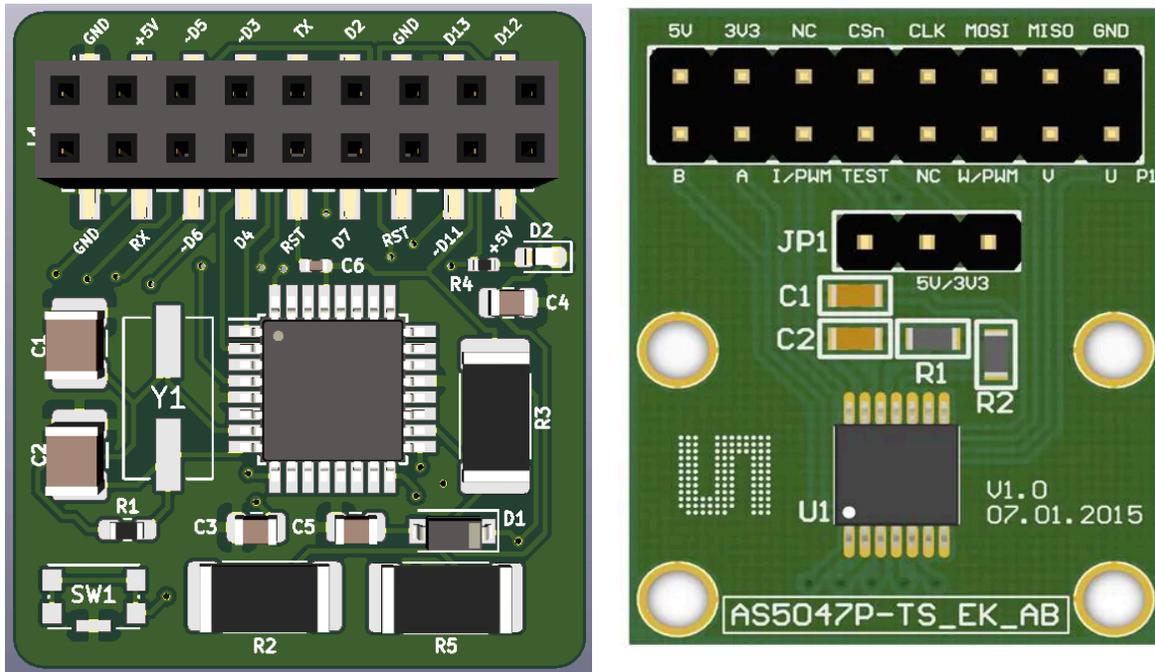
The microcontroller does not come bootloaded with Arduino, natively, so the In-Circuit Serial Programming (ICSP) pins are used for the setup of the programming language on the ATMega328p chip. The FTDI pins mimic the functionality of a standard USB and allow the

Arduino IDE to upload sketches onto the chip. The remaining pins on the PCB are for digital input and output connections.

The changes of the PCB design are varying with the objective of the whole Switch-Arm design. From version one having the most basic functional microcontroller to version two changed the layout of the female pin socket on the board, the reason behind that was to meet the objective when making version two as an add-on board, and the rotary sensor, AS5047P, was going to the top of the PCB, in which the sensor was required on determining the changes of the Switch-Arm's rotation. Due to the rotary position sensor requiring a magnet on top of the AS5047P chip, this rotary sensor needs to be installed directly below the Switch-Arm where the magnet was attached. Therefore, having the rotary sensor able to attach to the PCB and both of them being installed below the Switch-Arm was the original idea. Figure X below is showing the new layout for version two PCB.



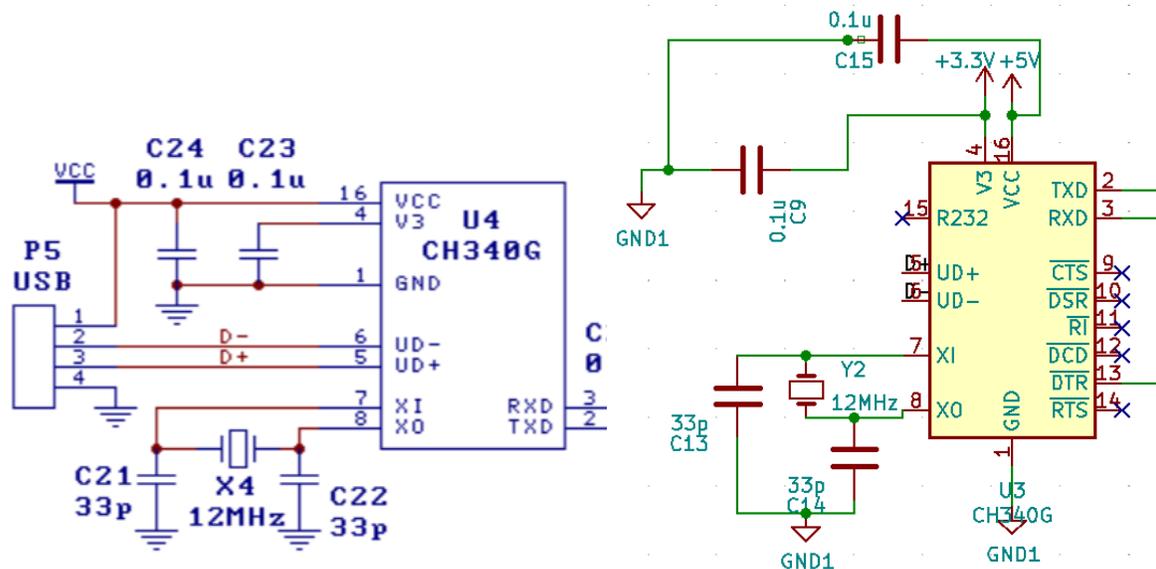
From the 3D views below, it is easier to see the expected result that the sensor board on the right, AS5047P, should be able to stack on the PCB on the left by connecting the male pins to the female socket. And the stacked PCB and rotary sensor board is going to be installed under the Switch-Arm, where the base has a magnet attached and the sensor chip right below that to detect the rotation change.



However, following with the change of the Switch-Arm design and more requirements for functions that the PCB needs to have. The version three PCB requires more functionalities and is more similar to the Arduino Nano board. Based on the characteristics that version two design has, including the surface mount devices design and basic functionalities, version three have the following improvements. Mounting holes on the four corners of the PCB to mount the board on the Bogie's surface, and which are in M3 size and electrically isolated to the board. Having the bottom plane of the PCB connected to the ground and made the width of the tracks for the 5V and V<sub>in</sub> thicker, which are 0.3mm (11.81mils). The calculation of the width of the track is shown in the electrical analysis in Chapter four.

Moreover, because the input voltage from the bogie team to supply the Switch-Arm and the PCB is 25.2V provided by a custom 12S battery pack, it requires the voltage regulator to drop the voltage from input to the PCB and remain at a constant level. The LM1117-5v and LM2940 regulators were considered, and an LM1117-5v in the SOT-223 package was finally selected because of the need to be in SMD and the inventory advantage from the manufacturer. And the mini-USB is an important feature that version three got upgraded for data transporting. For version two, the programming process was relying on connecting the RXD and TXD pins from the other Arduino board to the Switch-Arm PCB. With the mini-USB port, the Switch-Arm PCB is able to communicate with the computer and get programmed with Arduino IDE.

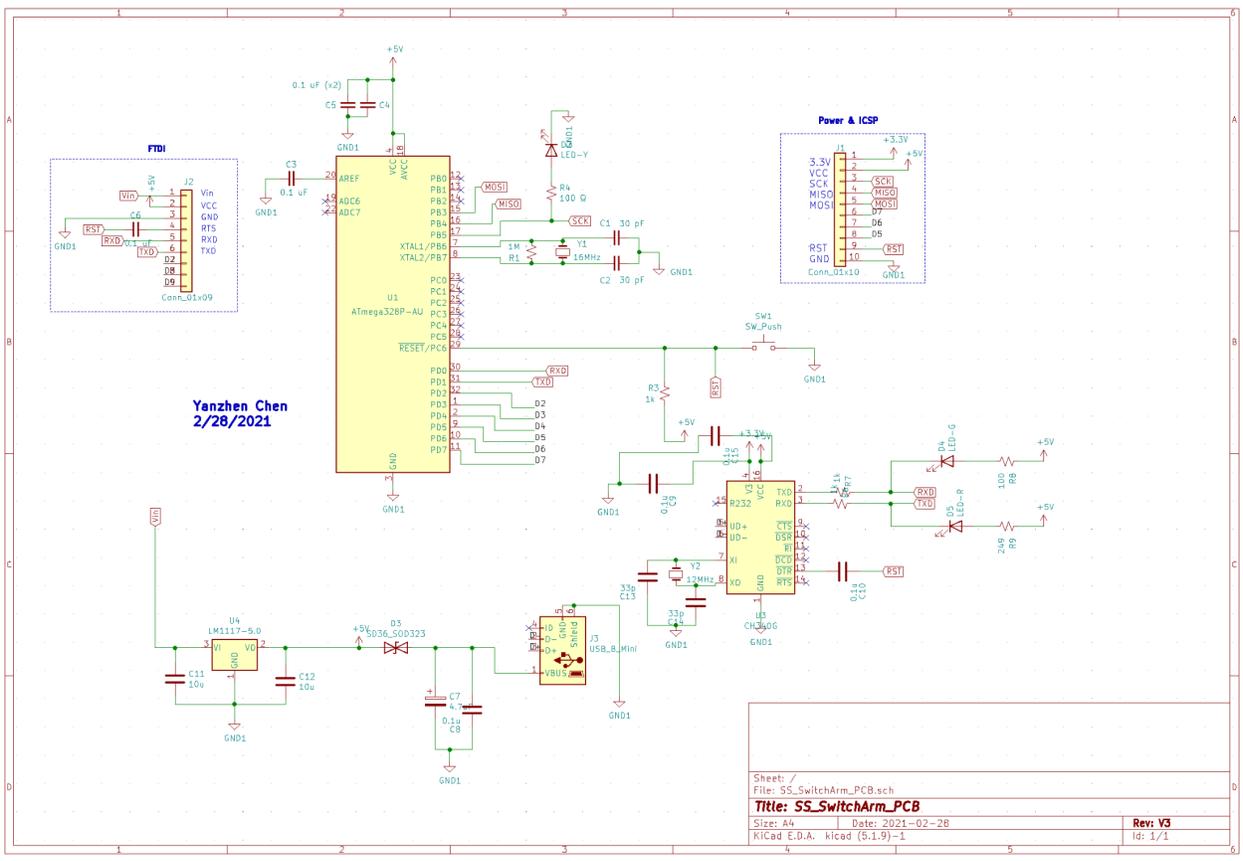
However, using the mini-USB requires a USB chip such as CH340G to convert the USB to the serial interface. According to the datasheet of the CH340, "In serial interface mode, CH340 supplies common MODEM liaison signal, used to enlarge asynchronous serial interface of computer or upgrade the common serial device to USB bus directly." In order to implement adding the USB and CH340G to the PCB, there are tips that the CH340 datasheet provided to do it properly, such as that CH340 chip has set up USB pull-up resistor internal and UD+ and UD- pins must be connected to USB bus directly. From the datasheet application example shown in the figure E# on the left, the capacitance of C8 varies from 4700pF to 0.02uF, used to internal power node decoupling of CH340; the C9 is 0.1uF, used for external power decoupling and a 12 MHz crystal X2 with two capacitances C6 and C7 are used to clock surge circuit. The figure E# on the right shows the setup for the CH340G chip portion of the Switch-Arm board, and some pins on the CH340G chip are not connected to anything because they were not in use in this project.

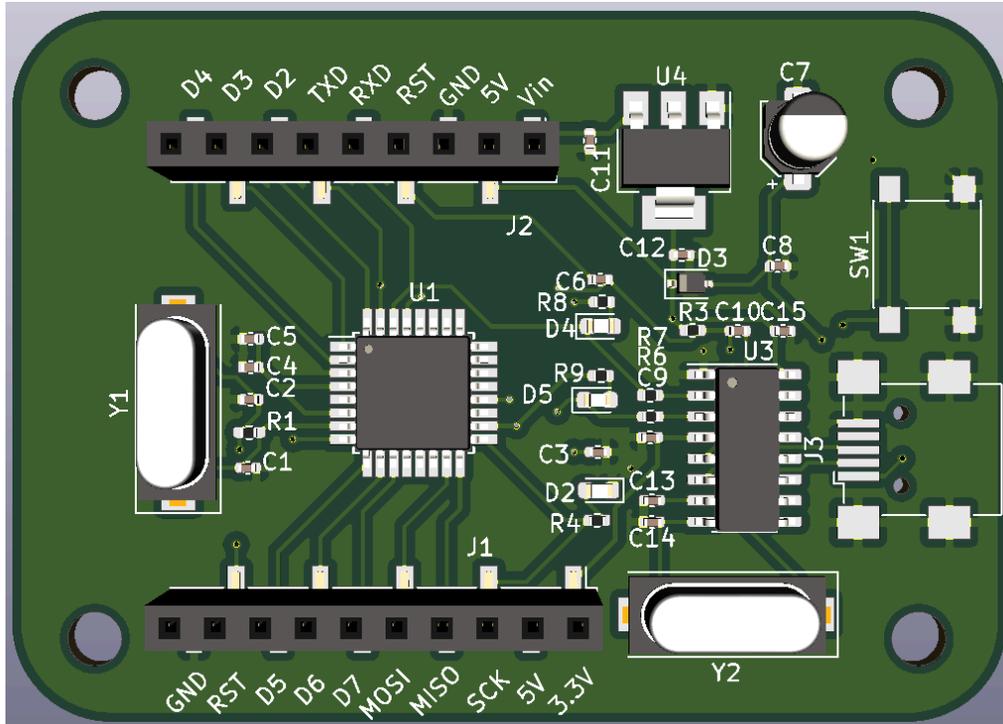


In addition, the changes for version three also include added LED as indicators for signal transmissions, such as RCD and TXD pins, fillet the edges of the PCB and keep the size as small as possible, and added another ground plane to the back of the PCB. The most significant changes for version three are the layout of the PCB that the additional parts need to be properly placed on the board, relative parts need to be more closed to each other, and they also need to be in the appropriate location for reducing the length of the track and avoiding signal loss.

Moreover, the team decided to use the JLCPCB's surface mount service starting from version

three, so smaller SMD components are available for keeping the dimension specification. Some components changed from the 1206 or 0804 package to the smaller size of the 0402 packages, such as the resistors and capacitors. Figure E# and E# below is showing the schematic of the version three PCB design and its 3D view respectively.

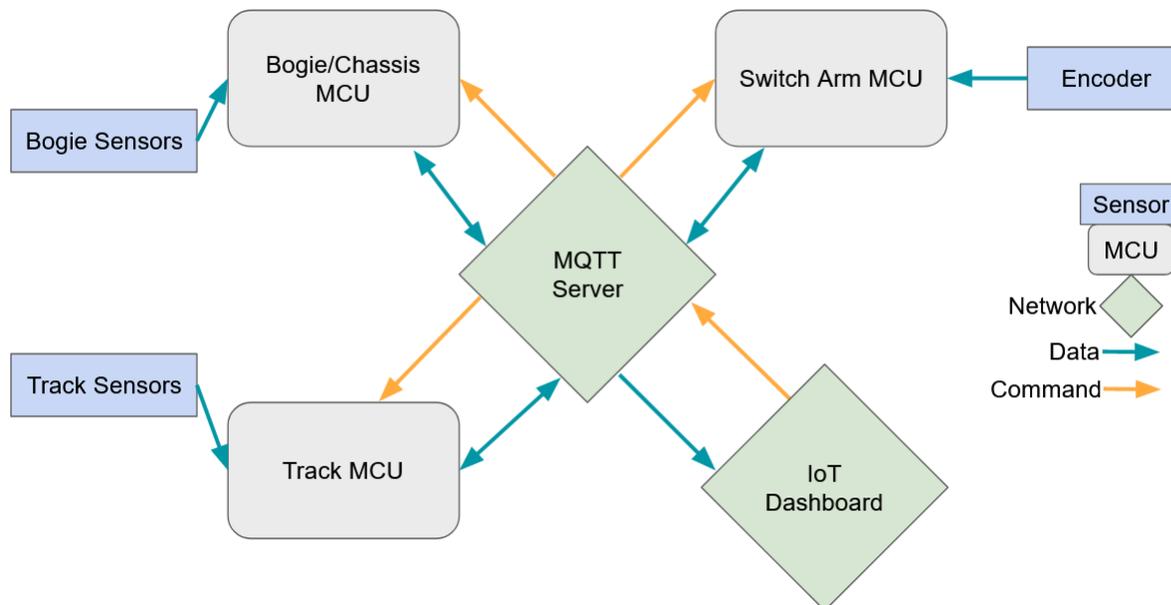




### ***Programming:***

The code required for the control of the Switch Arm went through many different design stages. These stages were tailored to the changing coding requirements and progressed through 3 general stages: (1) wireless communication and motor control, (2) motor and encoder control, (3) motor control. Each stage dropped aspects from the one before as the requirements had their scope narrowed down. This refinement was driven by design changes within both the Switch Arm team and the overall Superway project group. Each stage will be discussed below.

Stage one occurred very early on in the project. Consisting of the majority of the fall semester, this stage lasted until just before the third presentation. It was focused on planning for wireless communication and motor control. Based on current designs by the mechanical team, the programming team was focused on controlling either a stepper or servo motor. On the electrical side, wireless communication was being explored via the use of an ESP32. This ESP32 would have communicated directly with the other wireless MCUs in the Superway system. Commands for turning would have been passed to the Switch Arm MCU from the IoT dashboard via MQTT protocols. Applicable sensor data would have been shared between the IoT dashboard, the Bogie/chassis MCU, and the Switch Arm MCU. A block diagram showing the overall system configuration can be seen below in Figure P1.



As mentioned previously, the majority of work done by the programming team during stage one was research. Our general unfamiliarity with wireless communication protocols meant that this was the main focus of our research. We explored various different protocols for communicating data. Very early on, HTTP and HTTPS protocols were considered for transmitting messages between each group's systems. Based on work by the other Superway groups, the focus of our research quickly shifted to the MQTT protocol. This protocol would allow a wide variety of messages and data to be transmitted between systems. The major benefit of using MQTT was low usage of resources. MQTT works by allowing publishers to send data packets to a centralized server. These packets are organized into topics; this would allow the teams to easily distinguish data from different sensors as well as commands and messages for the various systems. Once the data packets are published, a subscriber can access the packets and utilize the information. Another benefit of MQTT is that a client can perform both publisher and subscriber roles. (Petrova & Solovev, 2020). This makes it easy to both send and receive data from a single MCU. Having all these data packets and messages stored on a central server also makes recovery of telemetry easy. A client can simply access all messages in a topic and use them to reconstruct any events leading up to a system failure.

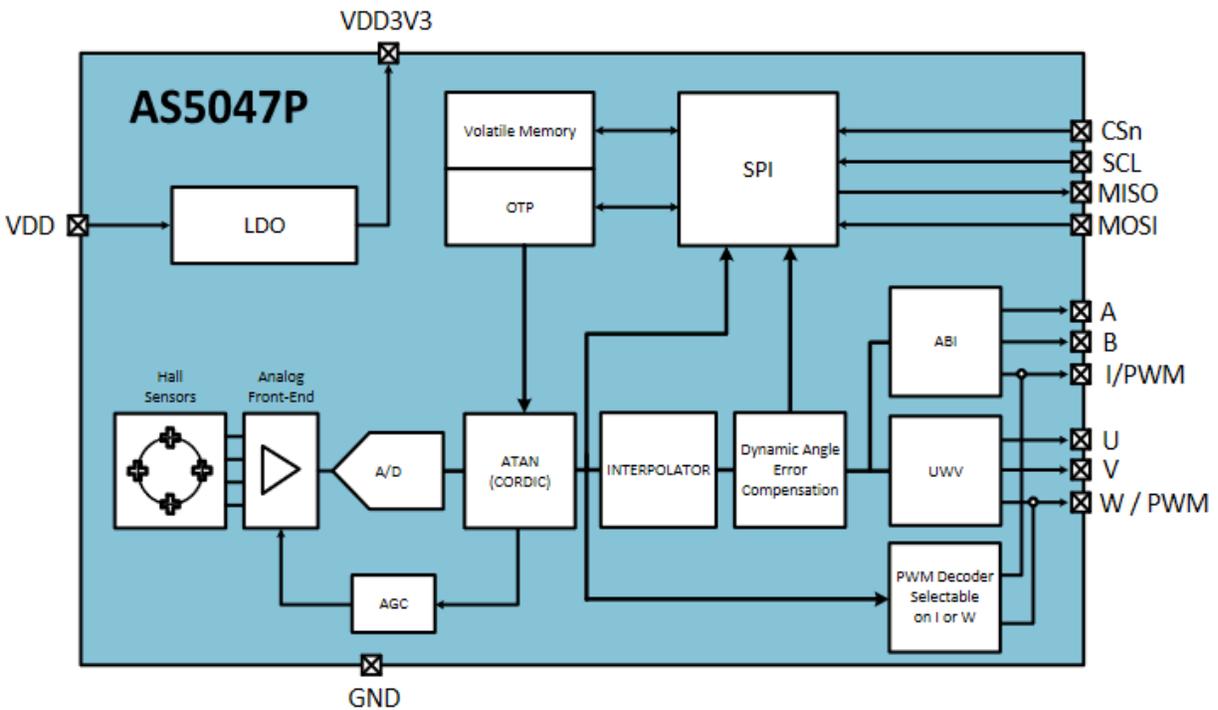
The second stage the programming team entered was accompanied by more research and planning. This stage was entered some time before the final presentation of the spring semester

and lasted through half of the spring semester. This stage dropped the wireless communication requirement for the Switch Arm system. Our new plan was to communicate directly with the Bogie/Chassis MCU and to let them handle the wireless communication with the IoT dashboard. This decision to drop wireless communication from the scope of the Switch Arm design was based on two factors: (1) we had very little information to transmit and (2) the control of the Switch Arm did not require wireless communication.

The Switch Arm had very few sensors; at most two limit switches or an encoder. The data from these sensors would be used by the Switch Arm MCU to position the Switch Arm. It was decided that there was no need to transmit this sensor data to other teams. The other teams would be provided the status of the Switch Arm with either a simple confirmation that the arm was in the correct position or an error if the arm was out of position.

Controlling the Switch Arm was determined to be a simple process, not requiring a full wireless connection. By hardwiring a communication line to the Bogie/Chassis MCU, we could be assured of receiving turning commands without needing to troubleshoot a wireless connection. The short runs of wiring needed for this communication would have limited susceptibility to interference. Commands would also be received faster as the Switch Arm MCU could simply be put in a waiting state until a command came through.

With the requirement of wireless communication dropped, the programming team could focus more effort on researching motor control. Approximately halfway through stage two, a decision was made to switch to using a brushless DC motor coupled with an incremental encoder. This introduced new challenges as none of the members of the programming team had experience interfacing with the ESC required for the brushless motor. Some members had limited experience with encoders, but the AS5047P selected for use had used the SPI communication protocol. The AS5047P also featured a wide range of options for accessing the angular position data. Figure P2 below shows a block diagram of the internal architecture of the AS5047P. Below that is a table, Table P1, listing the volatile registers we planned on accessing.

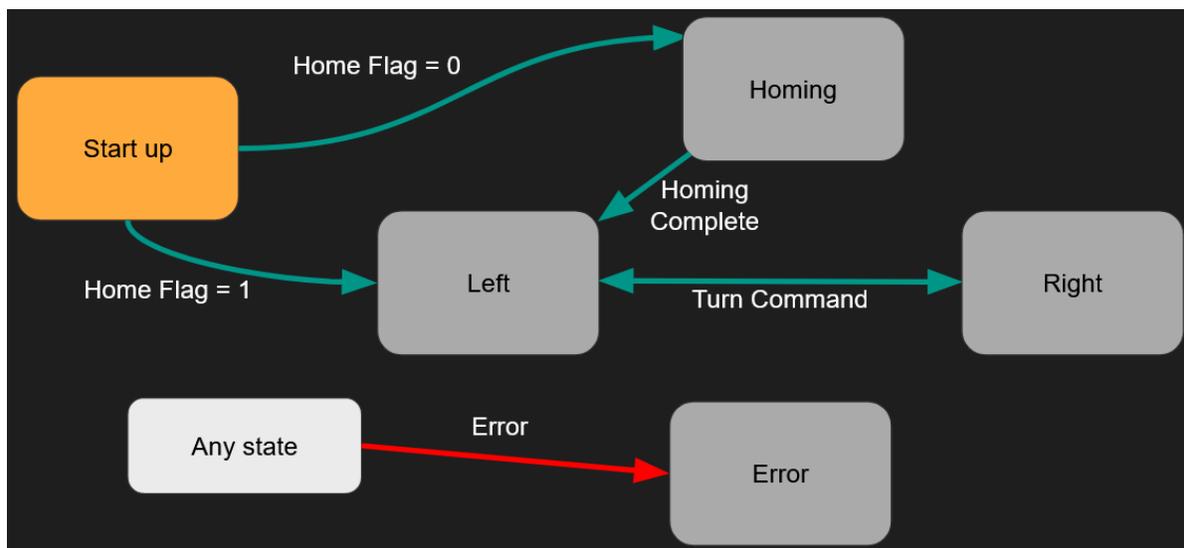


Address	Name	Default	Description
0x0000	NOP	0x0000	No operation
0x0001	ERRFL	0x0000	Error register
0x0003	PROG	0x0000	Programming register
0x3FFC	DIAAGC	0x0180	Diagnostic and AGC
0x3FFD	MAG	0x0000	CORDIC magnitude
0x3FFE	ANGLEUNC	0x0000	Measured angle without dynamic angle error compensation
0x3FFF	ANGLECOM	0x0000	Measured angle with dynamic angle error compensation

The research in stage two was accompanied by some programming. These programs can be seen in Appendix #. The main objective for the coding completed during this was to generate code that could be used to test the assembled hardware of the Switch Arm. A program was written to test the ESC-motor connection; this was intended to prove that our ESC and BLDC motor worked and to test the angular velocity that may be attainable for the Switch Arm. Another program was created to take care of the SPI communication with the AS5047P. We

hoped to figure out which angular position data would provide the quickest and most accurate readings. When the programming team finally had a chance to test the electrical components at the Design Center, we found that our ESC-motor combo was causing problems. We were unable to diagnose the root cause of the issues; the Superway Mentor team attempted to help with troubleshooting. Based on the feedback from the mentor team, it was decided to drop the BLDC and encoder from our design and focus on using a closed-loop stepper. This is the point at which the programming team transitioned to stage three.

Stage three consisted of the final programming stage of the Switch Arm project. The programming team finalized the code used for the control of the closed-loop stepper and worked out a communication strategy with the Bogie/Chassis team. The finalized code can be seen at the end of Appendix #. An early version of this code, SerialToMove.ino, relied on a serial communication with the Bogie/Chassis MCU to receive the turning commands. This code is a basic state machine that follows the block diagram shown in the Figure P3 below. The code for stage 3 had to incorporate two important features: (1) a homing function and (2) a minimally blocking stepper control function.



A homing function was necessary as the final design of the Switch Arm didn't include any sensing for position. The closed-loop stepper featured an encoder, but that data was not accessible to us due to lack of communication pins on our custom PCB. Including limit switches, either mechanical or magnetic, would have eliminated the need for a homing function but these were not included in the design. The homing function operated by moving the stepper in small, 5-10 step increments. When the arm reached the home position (oriented for a left turn), the user

pressed a button attached to a digital interrupt pin. This caused the Switch Arm to stop moving and the arm was put into the Left state. From there, the only moves necessary were 180° rotations; the Switch Arm would rotate from the homing position (left) to the right side and back again. While this homing feature worked for testing, it was susceptible to lost steps or mechanical interference. If the Switch Arm somehow ended up getting moved out of either the left or right positions, the homing would be off and would need to be recalibrated manually.

The library we used for interfacing with our stepper motor only had a blocking movement function. To ensure the Switch-Arm would be able to react to commands while in the process of switching, we had to create a non-blocking move function with the blocking one. The method for this was similar to the homing function in that the stepper moved in increments. For the move function the 180° rotation was divided into 10 sections. The function would call the blocking move function for this 18° of motion and then check for a new command in the serial buffer. If no command was present, the blocking function was called again for the next 18° move. The inertia of the Switch Arm would carry it through the brief pauses between movement steps without any noticeable deceleration.

When we began to implement our final prototype on the chassis, we discovered that we were unable to use serial communication to receive commands from the Bogie/Chassis MCU. The Bogie/Chassis team had other sensors communicating on all of their serial lines. We looked into using other communication protocols like SPI and I2C but discovered that we didn't have the necessary pins broken out on our custom PCB. To overcome these issues, we had to develop a communication protocol that utilized only 2 digital pins. This protocol will be discussed further in the documentation section below.

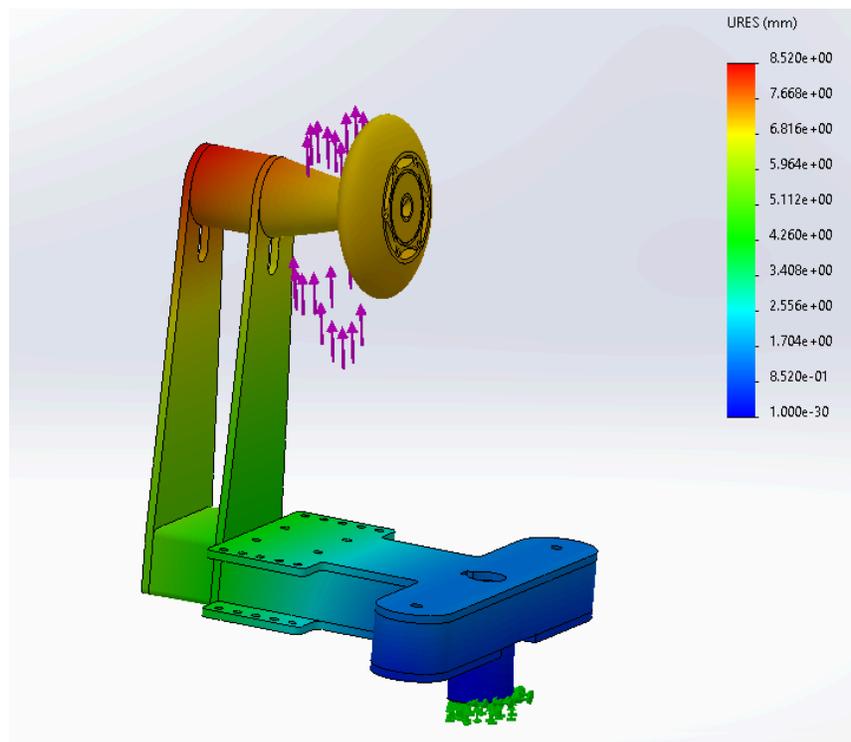
## 8. Chapter 4 - Analysis & Documentation (ME195A Report #3)

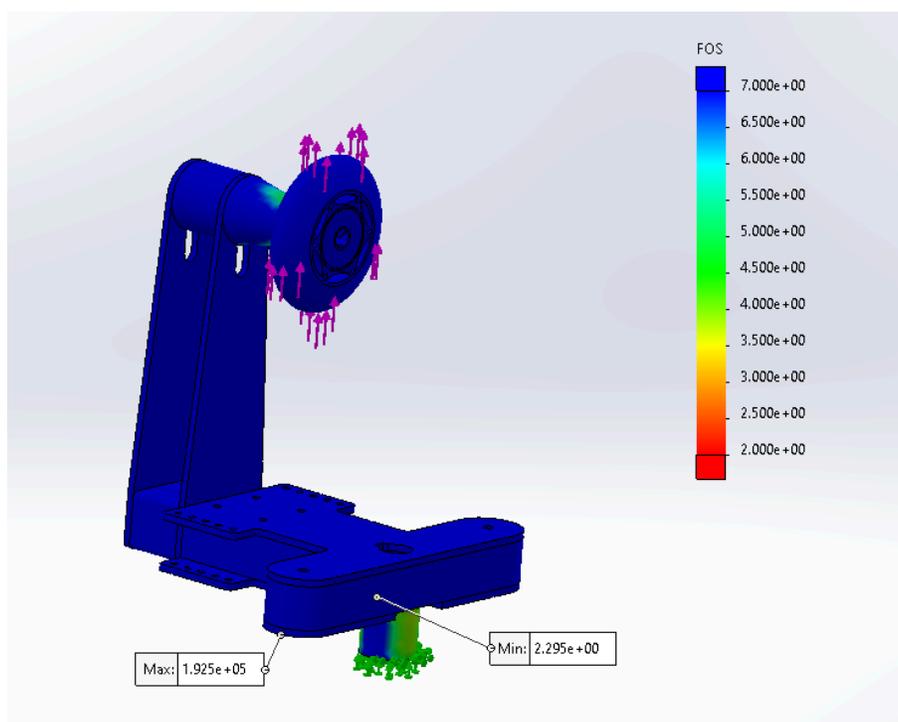
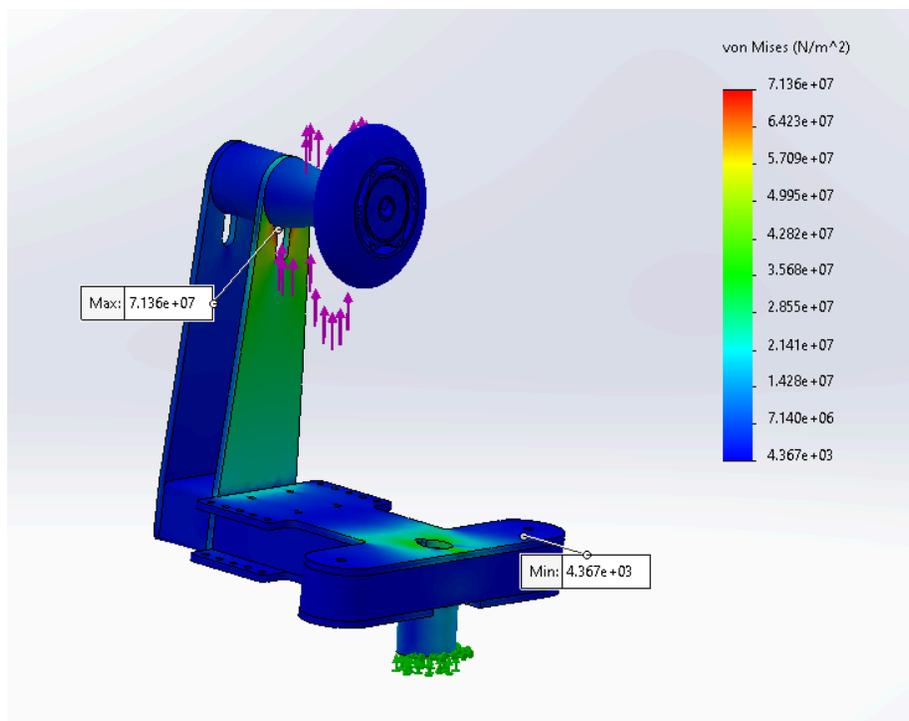
- a. Compare and show the results that support the selection of the optimal solution
- b. Show your finalized design
- c. (if your project includes Mechatronics) Show how you interface your prototype to the electronic system, what data acquisition system (DAQ) you used and what kind of software you used for data analyses.

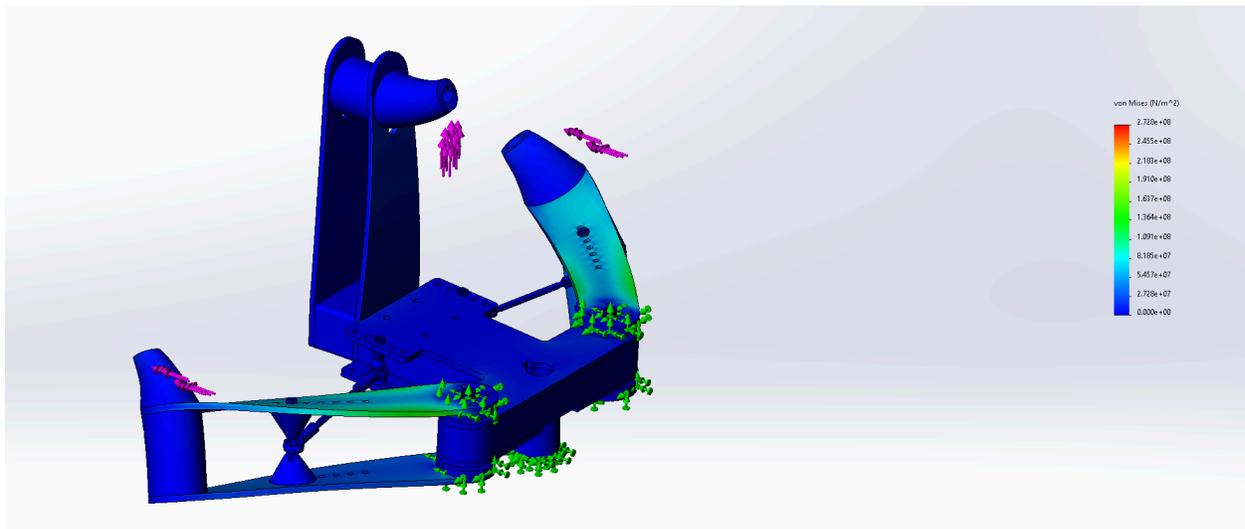
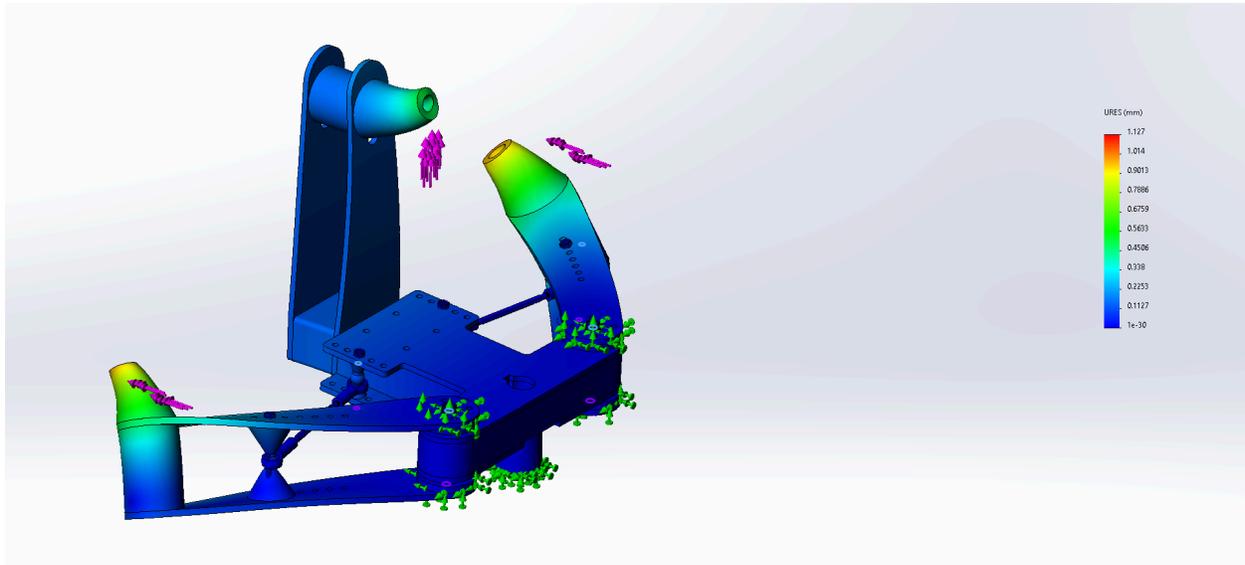
### Analysis and Documentation:

#### *Mechanical:*

The final switch arm was designed to hold 450 N of weight and withstand a contact force of 543 N with the passive y-switch. Finite element analysis was applied to our final switch arm design to ensure it would meet these requirements without any issues. The five figures found below show the displacement, von Mises stress, and factor of safety for both the 450 N vertical force and 543 N lateral force given by our finite element analysis results. The 450 N force was applied at the vertical wheel to simulate a scenario in which the bogie is hanging by that wheel only. The 543 N force was divided by two and placed on both of the lateral wheels of our design. The results showed a minimum factor of safety at the mounting spacer of 2.295 and maximum displacements of 8.52 mm for the vertical force and 1.127 mm caused by the lateral force.







***Electrical:***

Sizing a proper motor that can meet the operating requirement is essential. Before choosing the motor, some questions need to be determined to pick the right one. Know that the switching mechanism is for an arm rotating in the horizontal condition. Sizing the motor is relatively easy by utilizing the motor sizing tool provided by Orientalmotor company. With knowing the dimensions of the rotational arm:

*Length  $l = 102 \text{ mm}$ ; Width  $d = 30 \text{ mm}$ ; Offset length  $r = 51 \text{ mm}$ ;*

*Mass of the Arm = 0.2kg*

The moment of inertia can be with Equation 3:

**Equation 3:**

$$J = \frac{1}{12} * m * (l^2 + d^2 + 12 * r^2) = 7.086 * 10^{-4} kg * m^2$$

Then the most important part of sizing a motor is the torque, the load torque for this switching mechanism can be determined with Equation 4:

$$\text{Equation 4: } T_{ext} = F * d = 2.068 N * m$$

The total torque required from the motor can be determined with Equation 5:

$$\text{Equation 5: } T_{total} = T_{ext} + J * a = 2.069 N * m$$

Therefore, finding one motor that can provide at least 2.069N can meet the requirement for the switching arm, and the NEMA-23 was chosen finally.

The width of the track needs to increase because of the high voltage input. The width of the track can be calculated with the formulas 6 that provided in the Digikey website:

$$\text{Equation 6: } \text{Width[mils]} = \text{Area [mils}^2\text{]} / (\text{Thickness[oz]} * 1.378[\text{mils/oz}])$$

in which the Area can be calculated with the formula 7 below:

$$\text{Equation 7: } \text{Area[mils}^2\text{]} = (\text{Current[Amps]} / (k * (\text{Temp\_Rise[deg. C]}^b))^{1/c}$$

and the constants are defined by the IPC standards under IPC 2221 that  $k = 0.024$ ,  $b = 0.44$ ,  $c = 0.725$  and common values for thickness: 1 oz, ambient: 25 degree Celsius, temperature rise: 10 degree Celsius.

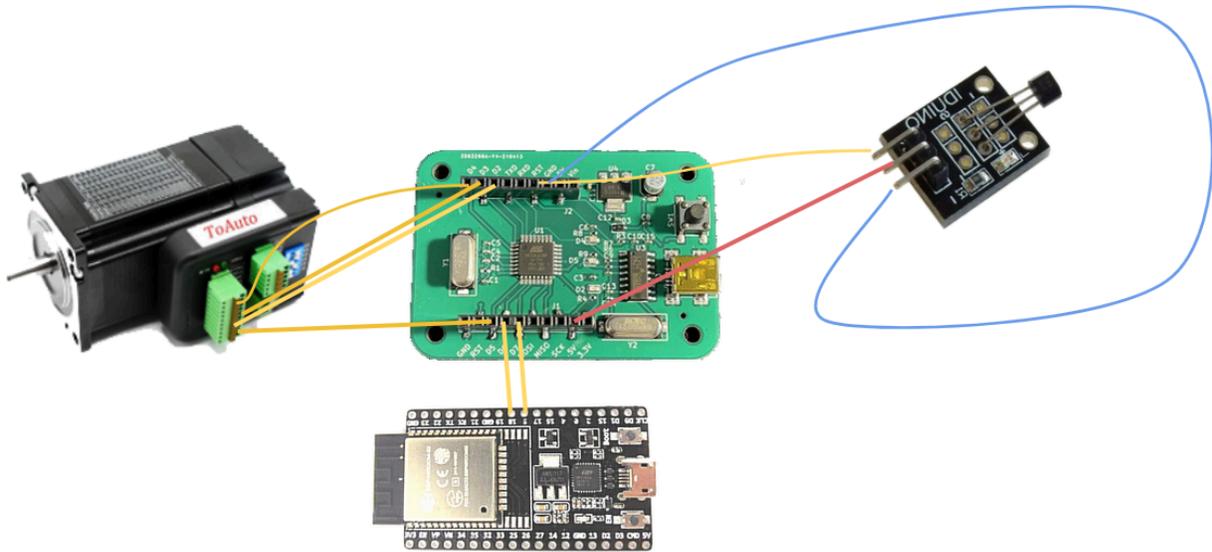
***Programming:***

Since there were only two states of importance for controlling the path of the Bogie/Chassis, Left and Right, we were able to implement a basic, binary communication strategy. The table below illustrates how this communication scheme works. One pin on each MCU is set as output and the other pin is set as input; the output of one MCU is then connected to the input of the other. When the output pin of the Bogie/Chassis MCU changes state, the Switch Arm MCU reads the new state. If the new state is different from the old one, the Switch-Arm completes the move. Once the move is completed, the Switch Arm MCU sets its output pin to match that of the Bogie/Chassis MCU. From the perspective of the Bogie/Chassis MCU, commands are sent down one line by changing the output high or low. The state of the Switch Arm is then read on the other line.

We tried different approaches of communication, including SPI, UART and I2C, but all the methods were dead- end due to either lack of appropriate pin out or lacking the availability in Bogie's ESP 32. Along with the change of electric connection, our code also needs to update to the correct version. The latest set-up is the binary communication as mentioned above. It's the most straightforward method, but it will limit the function and stability as the system gets more complex with not only communication between the Bogie but also the IOT or the Track team. An integration that includes both wired and wireless communication is a must for the future system.

	Command Line (B/C MCU Output, SA MCU Input)	State Line (B/C MCU Input, SA MCU Output)
Left	1	1
Right	0	0

A circuit diagram that shows how we connected and used the binary pin as communication is shown in the Figure below. We used Pin 2-5 on our pcb as motor data pins, and Pin 6-7 as the communication pins to receive and send commands with the EPS32's opened pins. Since our switch arm has only two states of rotation, pin 7 was used to transfer the state and error back to the master. However, in real integration with the Bogie there is no digital display or serial monitor to reflect the data, an add-on of speaker or indicating LED could be installed in the future development. As for the Hall effect sensor, it was connected to the RX pin, which is 0 in arduino code, to calibrate the homing position. We did create a state machine that require no sensor to operate the homing and reporting states, but setting up the initial location every time as the power on is inconvenient and not the best solution for a long run.



## 9. Chapter 5 - Fabrication and Assembly (ME195B Report #1-2)

- a. Bill of Materials and Cost Analysis
  - i. Include a bill of materials and table showing all costs
  - ii. Component choices are reasonable to reduce costs.
- b. Show how you made the prototype
  - i. Fabricated parts
  - ii. Purchased parts
  - iii. What challenges on making parts
- c. Show the final assembly of your prototype

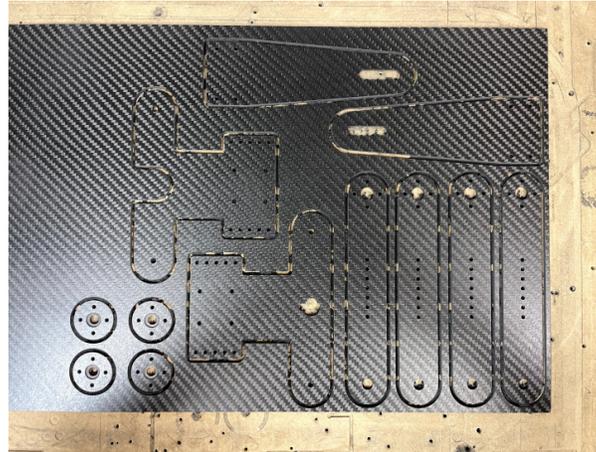
### Fabrication and Assembly

#### *Fabricated Parts*

##### **Mechanical:**

After the designs were finalized and it was off to the fabrication stage. Because the design needed to withstand a large amount of force, carbon fiber plates were used for a majority of the design, see Figure M#. The carbon parts were transferred from Solidworks to Fusion 360, since Fusion has an excellent and easy to use manufacturing portion to their application. After the CAM (Computer-Aided Manufacturing) files were created, they were then loaded to the CNC router and then cut out from the carbon sheet. Because of the differences in precision between the carbon parts and their interactive counterparts (screws, bearings, etc.), some pieces

had to be re-cut in order to ensure proper fitment in critical areas. Once the tolerancing was perfected, it was used to manufacture the second Switch Arm final design. There was no post-processing of the carbon pieces needed as the CNC routers setup, along with the CAM files, produced clean parts right off the table.



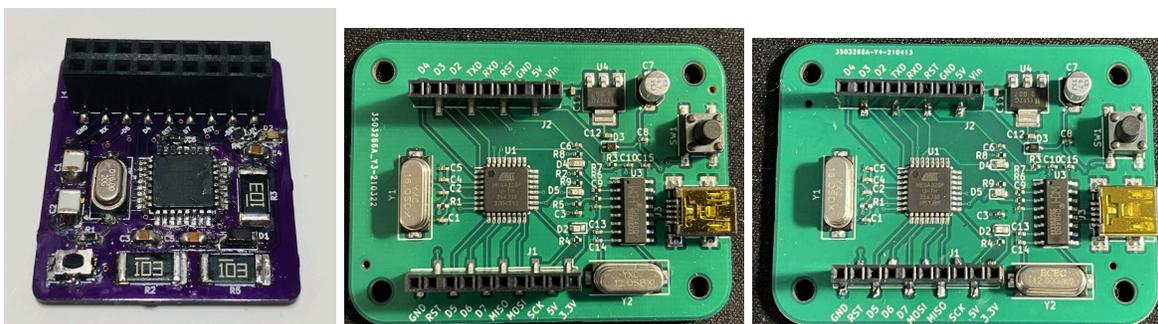
For the 3D printed parts, a different approach was necessary. Utilizing an Artillery Sidewinder X1 printer, all the necessary spacers, main plates, etc. were produced. In order to get the parts from Solidworks to the 3D printer, we used “Slic3r”, which takes the input of an STL file and outputs the appropriate g-code file that is needed. For all the 3D printed parts, the infill was set to 20%, using a rectilinear interior structure. This made the parts light, yet strong, see Figure M#. The filament used was 1.75mm PLA from Hatchbox. Because of the multiple iterations of the design, the rapid prototyping allowed for these design changes to be implemented overnight.



### Electrical:

The final version for the PCB was fabricated using the assembly service provided by the Jia Li Chuang Printed Circuit Board (JLCPCB) company. The online assembly service allowed us to paste our Gerber files and component files for manufacturing. The JLCPCB provided the surface mount assembly service to benefit the customers from having trouble with soldering those tiny SMD components, but there are some rules that need to be followed to take advantage of the service properly. JLCPCB provides an instruction page for generating the bill of materials and component placement list, known as a Centroid file/Pick and place file, from KiCAD, which are essential for the manufacturer to know what components the board needs and where they are going to be placed. Before following the instruction page to generate the necessary files from KiCAD, one last step that needs to be done beforehand is to find the components from the manufacturer's inventory and update the part numbers in the BOM file. After properly generating the files and uploading them to the manufacturer's website, following the instructions and paying for the cost, the fabrication of the electrical parts is done.

The Switch-Arm PCB has been through three times the fabrication that manually soldered the components to the ordered PCB for version one, and JLCPCB's service for version three and the final version. All of the fabrications are shown in the figure E# below, and they are version two, three, and the final version respectively, from left to right. Even though the final version of the PCB looks like barely having any change been made, it is necessary to have that it resolves many issues version three encountered, like the uploading and boot-loading issues, and this is the one being perfectly used on the project right now.



### ***Purchased Parts***

#### **Mechanical:**

The parts used in the assembly of the Switch-Arm prototype were almost entirely manufactured by the Switch-Arm team. Only the nuts and bolts that held the pieces together, as well as the guide wheels and bearings, were purchased from multiple suppliers (Amazon and McMaster-Carr). After getting all of the modeling finished, we used Solidworks to give us the overall lengths of multiple parts, in order to get accurate screw lengths. All items purchased were in bulk which led to the team being able to manufacture more than 1 Switch Arm, with the ease to continuously produce more.

The hardware was chosen with multiple things in mind: cost, strength, and weight. In order to ensure that the Switch Arm is reliable, we had to choose hardware that would last and not fatigue under the stresses that each component would see. Components that had lower load characteristics received smaller hardware to save on weight. However, when the components had a large load applied to them, we chose hardware that could support multiple countless fatigue cycles. Taking cost into account along the way allowed us to pick and choose the necessary items for the entire Switch Arm assembly.

#### **Electrical:**

The components used on the PCB were purchased through JLCPCB with the female header being the only component to be bought separately. The NEMA-23 stepper motor, encoder, motor driver, sensors, and the logic-level converter were all purchased via Amazon given the quick shipping times and competitive pricing, as shown in the bill of materials (BOM) (Table EX). It is important to note that some components on the BOM were never used such as the buck-converter and the FTDI converter - primarily because the final version of the PCB added these components directly to the board.

	Item or Order	Quantity	Purchase Date	Cost	Shipping	Total	Bought By	
<b>Electrical</b>	PCB Parts - Mouser	1	11/27/20	\$21.91	\$7.99	\$29.90	Kris Gonzalez	
	PCB Manufacturing - OSH Park	1	12/1/20	\$5.15	\$0	\$5.15	Kris Gonzalez	
	PCB Manufacturing - OSH Park	1	2/8/21	\$14.00	\$0	\$14.00	Kris Gonzalez	
	PCB Parts - DigiKey	1	2/8/21	\$11.68	\$5.00	\$17.75	Kris Gonzalez	
	Buck Converter 12v to 5v	1	2/17/2021	\$10.91	\$0	\$10.91	Jingwen Tan	
	Gikfun FTDI Conerter	1	2/27/2021	\$9.28	\$0	\$9.28	Jingwen Tan	
	PCB V3 Manufacturing - JLCPCB	1	3/22/2021	\$69.36	\$2.08	\$71.44	Yanzhen Chen	
	PCB Parts - DigiKey	1	3/22/2021	\$31.73	\$0	\$31.73	Yanzhen Chen	
	Nema23 with encoder and driver	1	3/29/2021	\$109.00	\$0	\$109.00	Mentor Team	
	30:1 Worm Gearbox	1	3/29/2021	\$62.00	\$0	\$62.00	Mentor Team	
	PCB V4 Manufacturing - JLCPCB	1	4/14/2021	\$69.65	\$2.09	\$71.74	Yanzhen Chen	
	PCB Parts - DigiKey	1	4/14/2021	\$29.73	\$0	\$29.73	Yanzhen Chen	
<b>Mechanical</b>	PLA 1.75mm Filament	1	2/10/2021	\$24.99	\$0	\$24.99	Jesse Kavros	
	Mercer 75mm Wheels (Pack of 4)	1	4/11/2021	\$39.95	\$0	\$39.95	Jesse Kavros	
	ABEC 7 Bearings (Pack of 8)	1	4/11/2021	\$10.00	\$0	\$10.00	Jesse Kavros	
	10x4x4mm Bearings (Pack of 5)	2	4/13/2021	\$13.98	\$0	\$13.98	Jesse Kavros	
	M4x50mm Socket Head Cap Screws (Pack of 20)	1	4/13/2021	\$8.49	\$0	\$8.49	Jesse Kavros	
	M3 Turnbuckles, 70-90mm travel (Pack of 2)	2	4/13/2021	\$21.78	\$0	\$21.78	Jesse Kavros	
	AOWISH 64mm Inline Skate Wheels (Pack of 4)	1	4/18/2021	\$17.99	\$0	\$17.99	Jesse Kavros	
	M8 Nyloc Nut (Pack of 50)	1	4/13/2021	\$8.68	\$22.53	\$31.21	Jesse Kavros	
	M8 x95mm Socket Cap Screw (Pack of 10)	1	4/13/2021	\$9.55	\$0	\$9.55	Jesse Kavros	
	M3x45mm Socket Cap Screw (Pack of 50)	1	4/13/2021	\$12.76	\$0	\$12.76	Jesse Kavros	
	M3x40mm Socket Cap Screw (Pack of 25)	1	4/13/2021	\$3.42	\$0	\$3.42	Jesse Kavros	
	M3x35mm Socket Cap Screw (Pack of 25)	1	4/13/2021	\$3.90	\$0	\$3.90	Jesse Kavros	
	M8 Washer (Pack of 100)	1	4/13/2021	\$7.14	\$0	\$7.14	Jesse Kavros	
	M4 6mm OD, 5mm Long, Spacer	8	4/13/2021	\$16.72	\$0	\$16.72	Jesse Kavros	
	M8x100 Flat Head Screw	2	5/5/2021	\$6.24	\$0	\$6.24	Jesse Kavros	
	3mm Carbon Fiber Plates	2	-	\$0.00	\$0	\$0.00	Jesse Kavros	
						Running Total:	\$640.57	
						Amount Divided:	\$71.17	
					Cost Allowance:	\$1,000.00		
					Remaining Allowance:	\$359.43		

## Challenges

### Mechanical:

The decision to use additive manufacturing for most of the components came from the inherent ability to rapidly prototype a working model of the Switch-Arm without the cost and time of stronger, alloy components. The 3D printed parts gave up strength for the ability to quickly have physical components that could be tested for fit and adjustability. The tolerances of the 3D printed parts were expectedly inexact, leading to some difficulties during assembly, including the insertion of screws and small bolts through 3D printed holes. Because of the change of design at the beginning of the Spring 2021 semester, the expectation for a fully-manufactured and tested prototype was dismissed. Re-designing the entire assembly, as well as switching from a brushless DC motor to a stepper motor, required much more time and re-testing than moving forward with the fall semester's design. The initial design had only a handful of components and no moving parts aside from the motor and the wheel bearings. The new design presented the challenge of making rotating arms that could lock into place and withstand the moment placed on them by the force of the Bogie's weight behind the Switch-Arm coming into contact with the PYS. The hinge points between the arms and the base assembly required tight tolerances and bearings to allow for smooth rotation during adjustment without

introducing any instability. The initial main plate design was all 3D printed. Additionally, screws and nyloc nuts were used to keep the bearing caps attached to the boomerang arms. Once assembled, those would interfere with one another. That interference was scratching the top and bottom surfaces of the main plate, shown in Figure M# below. The main plate was later revised to be sandwiched between two carbon plates, and the bearing caps were switched to support self-clinching PEM nuts, which yielded more clearance between the two parts.



**Electrical:**

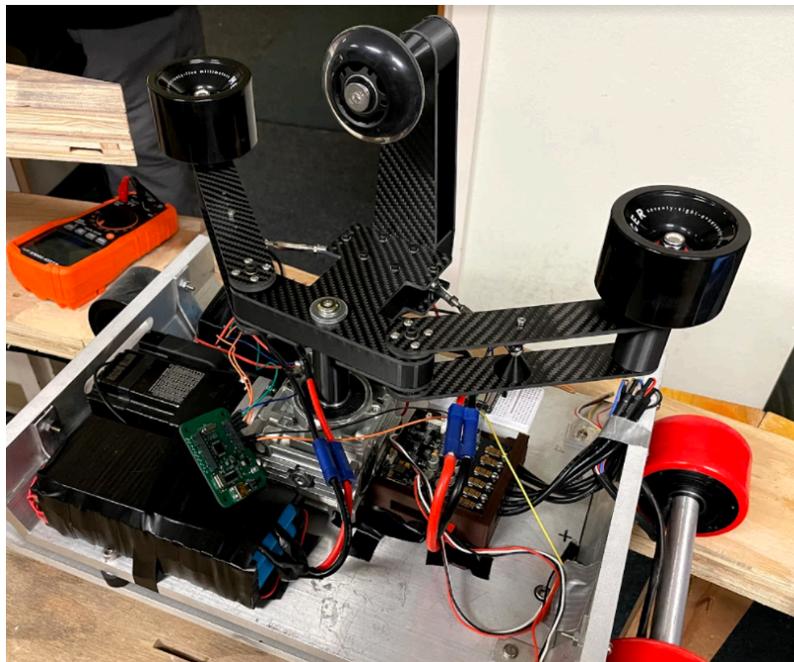
The challenges early on in the semester concerned whether or not to include a dedicated signal plane onto our board to have a 4-layer PCB (Power, GND, Signal, & GND). However it was decided to focus on the original specification of getting a working PCB, so this plan had to be ignored. This meant wiring would still be abundant. Also, since the SMD components take up more area on the PCB compared to THT components the board size could not be made close to the 1-in<sup>2</sup> dimension. As with many PCBs, routing became more difficult as we started to add more components so the use of vias became necessary. Finding the correct components, such as the crystal oscillator with a pair of capacitors and the diodes to protect from spikes became difficult to select for our microcontroller. The issues with the crystal oscillator stem from how a

crystal is selected when given a set of capacitors. Equation EX shows how choosing a crystal is dependent on finding the required load capacitance from the two capacitors and stray capacitance  $C_s$ . Every trace and lead on our components has some stray capacitance. The total of these values is represented by  $C_{stray}$ . The general estimated value of stray capacitance for our type of PCB can be a value of 2-5pF as long as we followed good layout practices and kept the trace from the crystal to the pins on the MCU as short as possible with no vias and low impedance. We decided on a value of 4 pF.

The entire process of creating a PCB to be assembled through JLCPCB was easy and convenient, but some of the components that have initially been chosen while designing the PCB have to change to those that are available on the manufacturer's inventory. Otherwise, the SMD assembly service would not proceed with those parts that are in shortage. In this case, version three and the final version of the PCB had to be changed accordingly, and the female headers are the only manually assembled part in this stage.

### ***Final Assembly***

Our final assembly for both the Switch-Arm and PCB are shown in Figure CX and EX. All of our components integrated nicely with the other Superway teams. Our switch arm and stepper were attached, our microcontroller was unfortunately taped to the Bogie body. The Bogie never ran at high enough speeds to make this a problem. The assembled system is shown below in Figure FA1.



The gearbox output shaft was able to fit nicely into the arm's keyway. This fit was not perfect as there was some slack. The arm was then fastened to the output shaft with a bolt that ran through the top of the baseplate. Although the full system was not perfect, this is just a prototype and most connections were made to be adjustable.

### **10. Chapter 6 - Testing Results and Analysis (ME195B Report# 2-3)**

- a. Show the overall testing setup
- b. Show your testing procedure
- c. Show the testing results in figures and/or tables. Indicate what does each curve or plot means, how do they indicate the functions of your final design
- d. How do these results match your specification, as well as meet your design criteria.
- e. Indicate whether or not the final design meets the safety requirements.
- f. Discuss the global, social, environmental, political and/or health and safety issues, including issues leading to a need for the project and the results of the project

### **Testing Results and Analysis**

#### ***Testing Results***

##### **Mechanical:**

Once the team had built two identical prototypes, physical testing could be implemented. Based on the FEA that was performed for the vertical safety wheel, with the max Bogie mass assumed to be around 70 kg, the team tested the vertical safety wheel's effectiveness in holding the mass of the Bogie, should it slip off the track. A pull-test was performed using a handheld force gauge by applying force between the vertical wheel and the key-way attached to the main plate in order to simulate the weight of the Bogie through the entire vertical portion of the Switch-Arm. The assembly was tested with up to 50 pounds of force, or about 220 Newtons. This static test showed that the prototype, made from 3D printed polymer and carbon sheets, can withstand the full weight of the Bogie.

The stepper motor's holding torque was measured experimentally and compared to the product's documented value. This was initially done with a 10:1 gearbox because the 30:1 gearbox that we used on the final build had not arrived yet. To test the holding torque, a torque

was applied while monitoring the force with a force sensor, shown below in Figure M5. The applied force would increase until the stepper could not provide enough force to stop it.

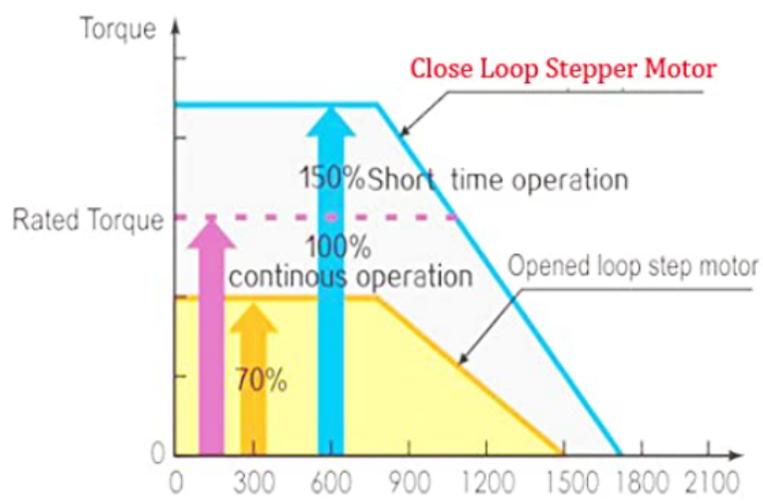


This test was repeated 5 times to get an average value for static torque. It was also done 5 times while the arm was moving to calculate dynamic torque. The results for both are shown below in Table X. The gearbox that was used on the final setup was a 30:1 gearbox, so the static torque would be 3 times the measured value of the 10:1 gearbox.

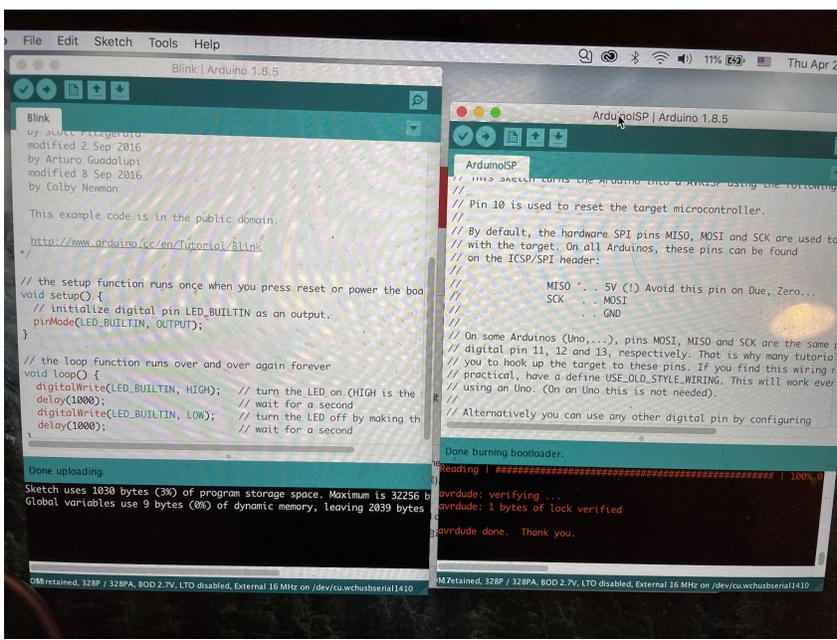
Test	Weight (kg)	Static Torque (Nm)	10:1	Test	Weight (kg)	Dynamic Torque (Nm)
1	19.34	20.87	Gearbox	1	15.58	16.81
2	20.37	21.98		2	14.23	15.36
3	18.33	19.78		3	14.15	15.27
4	21.37	23.06		4	15.32	16.53
5	21.4	23.09		5	14.86	16.04
Ave	20.162	21.76		Ave	14.97	16.15

### Electrical:

Given the values shown above, in Table X, and Figure X below, we saw that a NEMA-23 stepper motor would suffice in combination with the gearbox. Given the results from testing in Table X, we can see that the actual max holding torque is around 2.2 Nm without the gear ratio. This holding torque is more than enough for the Switch-Arm.



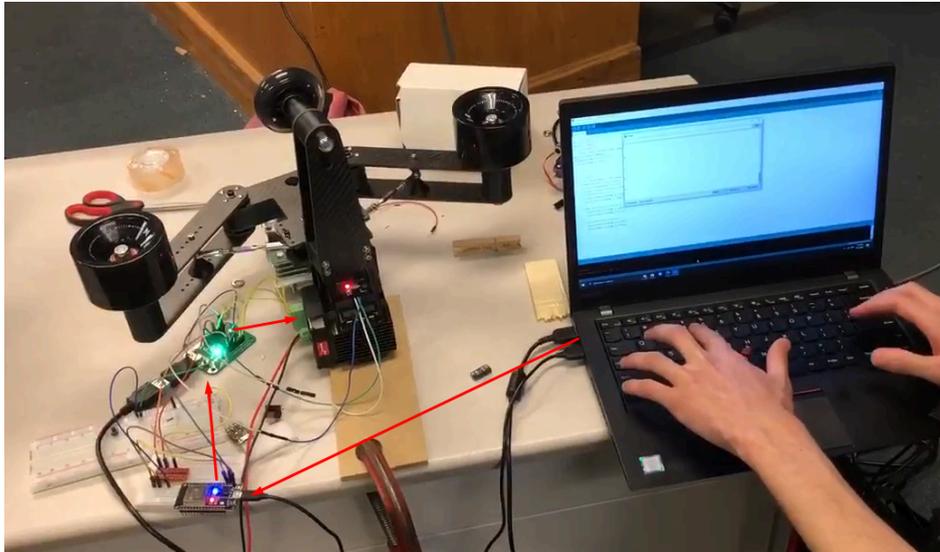
There was multiple testing of the PCB because there were four different versions of the PCB. Versions one through three had issues with the bootloader, and the programming team could not use the PCB. The final version of the PCB was successful with bootloading the Arduino hardware, as shown in the figure E# below, using the In-Circuit Serial Programmer, known as Arduino as ISP, and performing the programming team’s tasks, such as digital input and output and serial communication. The LED was able to blink down to a delay of 62.5-nanoseconds which proves that our PCB can digitally switch the Arm in less than 1-second



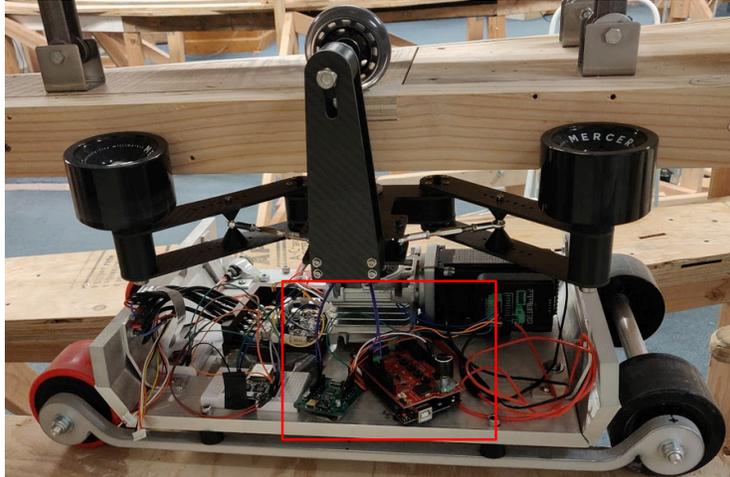
**Programming:**

Using the final version of the code presented below in Appendix #, the Switch Arm was able to move between the Left and Right positions. The homing function worked perfectly and allowed us to finetune the starting position of the Switch Arm. Our nonblocking movement function also worked very well. While we didn't get a chance to test it on a full test run, we didn't observe any jerkiness between movement steps. The inertia of the Switch Arm meant that the speed was maintained even when the motor stopped stepping to check for new commands.

To confirm the validity of our program, we connected our PCB to a second microcontroller that would receive commands from the serial monitor, and then relay the signals to our PCB controller. The connections are shown in figure P5 below. Using this set-up, we were able to get the switch arm to rotate on command.



For testing purposes, a program to fine tune the arm position. The angle of the arm on the PYS going straight had a wide margin for success. However the turn required an angle that was not a simple 180 turn from the other position. Creating a program to adjust the angle in small increments made the search for the correct angle much easier. This was controlled with a master microcontroller with buttons to set the pin states. The two controllers are shown in the figure below.



### *Analysis of the Results*

#### **Mechanical:**

During prototype testing at the SPARTAN Superway Design Center, the Switch-Arm was successfully mounted to the stepper motor, which was secured to the Bogie chassis. The setup functioned as intended during interfacing with the updated PYS built by the Track team. The success from our switch arm shows that its initial design goal of allowing the Bogie to switch directions along the track was accomplished. After multiple redesigns, we were able to find a design that met all of our safety requirements. The vertical wheel and lateral wheels allowed the Bogie to remain in its intended position without derailing. In addition, we were also able to use finite element analysis to determine a maximum deformation of 1.127 mm on contact with the PYS and a minimum factor of safety of 2.295, greater than our expected minimum value of 2.

#### **Electrical:**

The testing result shows the final version of the PCB design and fabrication are successful. The board is successfully boot-loaded with Arduino as ISP means the ICSP pins, such as MOSI, MISO, SCK, and so on, are working correctly and not connection or layout error. The board runs the programs that have been uploaded to the PCB, proving that the USB port is receiving and transmitting the signal properly. Moreover, the board went through the blinking tests for every digital pin showing the six digital pins are working and the LED indicators performance matching the design specifications. The team uploaded a small YouTube video (<https://youtu.be/Rj02lCkxmXc>) showing that the PCB can communicate with another Arduino using the serial pins, where the Switch-Arm PCB acted as a master, sending the message "This message is coming from the PCB," to the Arduino Uno, which served as a receiver.

**Programming:**

The program that we used to control the Switch Arm was successful in getting the arm to move in the way that we wanted. Because our application was simple in terms of logic, the code was straightforward and easy to implement. In testing we were able to get the Switch Arm to rotate 180 degrees fully on command from a second master arduino. However, when testing, it was found that the required arm positions were not a simple 180 degree rotation between the two. Furthermore, each PYS junction is unique so the best angle in each case would be different. Further testing would need to be done to find the optimal angle or the program would need to be changed to attempt to adjust for the best position for each case.

We had problems when connecting to the Bogie Chassis' microcontroller due to the voltage logic levels being different. The microcontroller that we made used a 5v logic where the Bogie used an ESP32 which has a 3.3v logic. We did heavy testing connecting our PCB to an Arduino Uno which also has a 5v logic, and our program worked as intended. We believe that with proper voltage conversion, or a switch to a 3.3v microcontroller, our program would work without any problems.

## **11. Chapter 7 - Conclusions and Future Work (ME195B Final Report)**

- a. Draw conclusions from your design, analytical calculations, and prototyping
- b. Comment on whether or not they meet your specifications.
- c. Draw conclusions from your team work
- d. Draw conclusions from cost analyses
- e. State the future improvements.

### **Conclusion and Recommendations for Future Work**

#### ***Mechanical:***

From a mechanical standpoint, the Switch Arm successfully travelled across the PYS all while keeping the entire Bogie on course. The horizontal wheels on the Switch Arm withheld the maximum load applied, however, we noticed that the arms of the Switch Arm were bending when the Bogie travelled across the curved portion of the PYS. There are multiple factors that led to this: PYS to Switch Arm misalignment, Switch Arm positioning adjustment, and the overall torsional characteristics of the boomerang arms.

Solidworks drawings give each team a good understanding and estimation of what each design can do and how they will interact with each other. When it came to implementing all the

designs together, adjustments were needed to have any sort of successful data. Because of the slight complexity in adjusting the PYS' height, we chose to verge on the safe side and move our vertical wheel to its maximum height position, which meant the PYS did not have to move down as far. However, this left the two horizontal wheels only partially contacting the face of the PYS. Because of this, the contact point on the horizontal wheels was raised, thus inducing a larger moment on the boomerang arms. This translated to an increased torsional load on the boomerang arms, causing the large deflection of the horizontal wheels.

Watching the entire system in motion showed that the Switch Arm design was doing its job, just not to the best of its ability. The leading horizontal wheel saw most of the torsional load, which meant a lot of rotation causing the wheel to no longer be parallel to the PYS surface. We took scenarios like this into consideration when designing and this gave validity to adding in turnbuckles to our project requirements. The turnbuckles allowed for fine tuning of the leading horizontal wheels contact point which in return would distribute the load more evenly across the leading and trailing horizontal wheels.

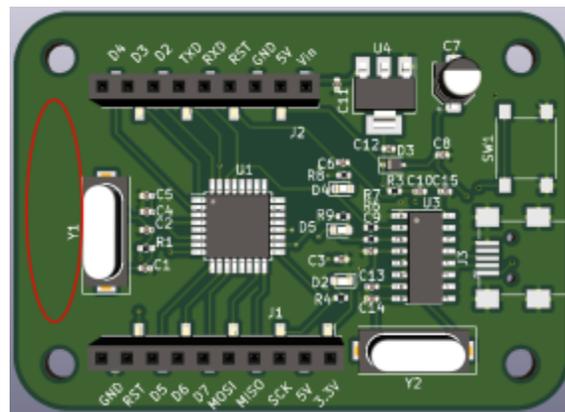
No prototype is ever perfect from the start; taking what we learned from the little bit of testing, it was evident that the boomerang arms would need some stiffening up to give the team confidence that this design would never fail after multiple interactions with the PYS. The design met the specifications on paper, but after the testing concluded, we felt that there were changes that could vastly improve the design.

Taking all the data and feedback into consideration, it was evident that the boomerang arms needed more rigidity. One of the recommendations we have is to switch some of the material from PLA over to aluminum. Aluminum is a light and strong material that is easy to machine. If large aluminum spacers were sandwiched between the boomerang arms, the overall strength would greatly increase. The same can be said for the vertical wheels design. We were able to test the current design by loading the vertical wheel with 50 pounds. In addition, the entire Bogie/Switch Arm assembly was placed on the track and the entire weight of the Bogie rested on the single vertical wheel. The vertical carbon plates were deflecting which led the team to believe that adding aluminum stiffeners between the two vertical carbon plates would increase the overall strength as well. The last iteration we felt would be necessary is the main plate. The entire core of the main plate is made out of PLA, this is great for prototyping but not for final manufacturing. If the main plate was made from aluminum it would increase the stiffness of the

vertical wheel attachment point. The current design has 4 bolts that thread into the main plates PLA material. This is not the best for overall strength and the threads can tear out if enough load is applied. Aluminum would have much tighter tolerances for the threads and could withstand increased loads that could be applied.

### ***Electrical:***

For the PCB we accomplished the majority of what we hoped to include in our microcontroller by the end of the semester. The final version was able to be bootloaded with Arduino firmware which was unsuccessfully done with previous iterations. Also, the final PCB version could execute code without any hardware bugs or shorts between signals, so the improvement was drastic. One thing we recommend for future iterations of the PCB is to use any unused space for extra female headers, similar to the spot in Figure X. We also suggest that the board size not be reduced if it means less components have to be placed onto the PCB, because the goal should be to reduce the most amount of wiring needed for our system. So including a female header specifically for any motor driver (or similar PCB) can help with wire reduction, and small sensors can even be added to the microcontroller similar to the light-emitting diodes already placed.



### ***Programming:***

Overall, the programming effort for the Switch Arm was successful. Ultimately, we produced code that controlled the direction of the Switch Arm upon receiving commands from the Bogie/Chassis MCU. This code served as testing code, allowing the Superway Car to be tested on the 10m scale track. These tests demonstrated that the Switch Arm could be used to switch direction on the track. While we were unable to complete any test laps that put our code to a full test, we believe that our code would be up to the task. This is due to the lack of

complexity of the Switch-Arm function; we only need to be able to move the arm through 180° of rotation. Our main concern with a full test would be electromagnetic interference. Our communication lines to the Bogie/Chassis MCU could easily pick up errors during full speed operation. Interference from the Bogie/Chassis motors could cause fluctuations in the command line signal which would trigger a Switch Arm movement when it wasn't needed. We are also concerned about the lack of position sensing for the Switch Arm. Without limit switches, we must take it for granted that the Switch Arm will only ever stay in the Left or Right positions. If a failure were to occur and the Switch Arm ended up out of position, a human would need to intervene and reset the home position. While the programming team had little say in the use of limit switches, we did make a plan to implement them on the Switch Arm. Unfortunately, we ran out of time to get them mounted and had to stick with using the homing function we created.

Future work on the programming side could be completed to switch to using a non-blocking stepper move function. Including code to regulate the start and stop accelerations of the stepper would help reduce torques during these actions. While we were able to mitigate the amount of blocking with our movement function, it would be best to eliminate any blocking code from the Switch Arm programming. The StepperDriver library written by Laurentiu Badea (GitHub repo: <https://github.com/laurb9/StepperDriver/tree/master/src>) includes non-blocking functions for controlling stepper movement. We implemented this library in small-scale test setups using a Nema-17 and A4988 stepper driver. The connections needed for our closed-loop stepper driver were noncompatible with the library however so we reverted to using the standard Arduino Stepper library. The StepperDriver library also includes functions for ramping the speed and acceleration which would help combat the large torques created by starting and stopping the stepper. We attempted to include speed-ramping in our stepper movement function but didn't have time to get the code ironed out.

The communication strategy we worked out with the Bogie/Chassis team works for testing purposes, but should not be considered a long-term solution. Changing this depends on improvements made to the PCB or the use of an off the shelf MCU. However, this would be an important first step for further testing to continue. Using either serial, SPI or I2C would improve the reliability of communication and increase the amount of information that could be transmitted. Currently, if the Switch Arm fails in some way, the only method for stopping the operation of the Superway car is via human intervention.



**12. Reference (All Reports)**

List all resources/references for your project work.

**\*See below**

## **References**

Digikey (n.d.). *Pcb trace width calculator*. Retrieved May 20, 2021, from

<https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width>

JLCPCB. (n.d.). How to generate the BOM And Centroid file from KiCAD. Retrieved May 21, 2021, from

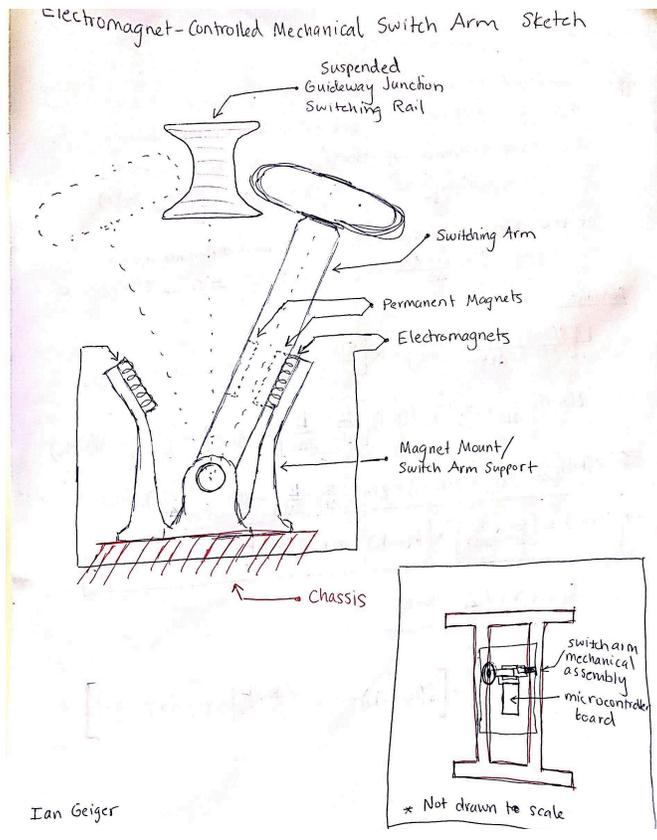
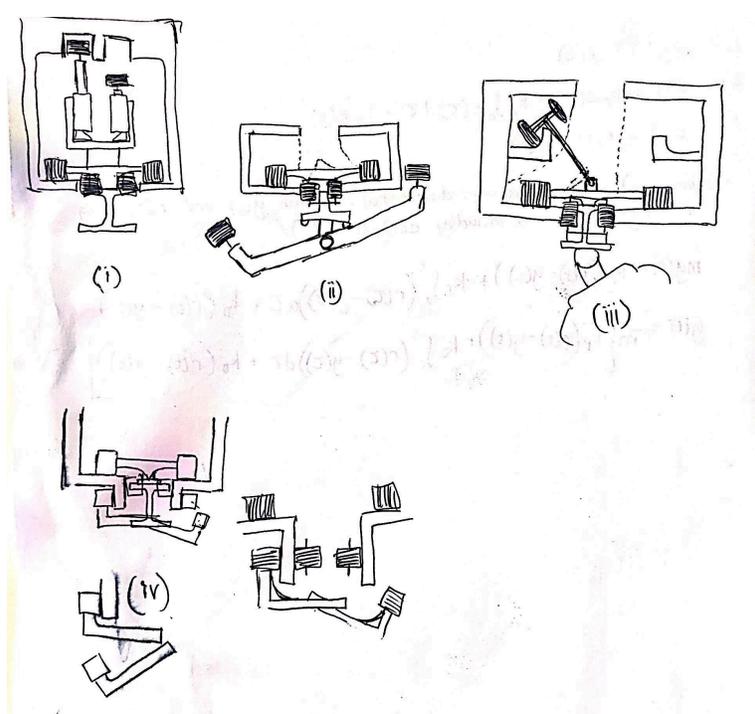
<https://support.jlpcb.com/article/84-how-to-generate-the-bom-and-centroid-file-from-kicad>

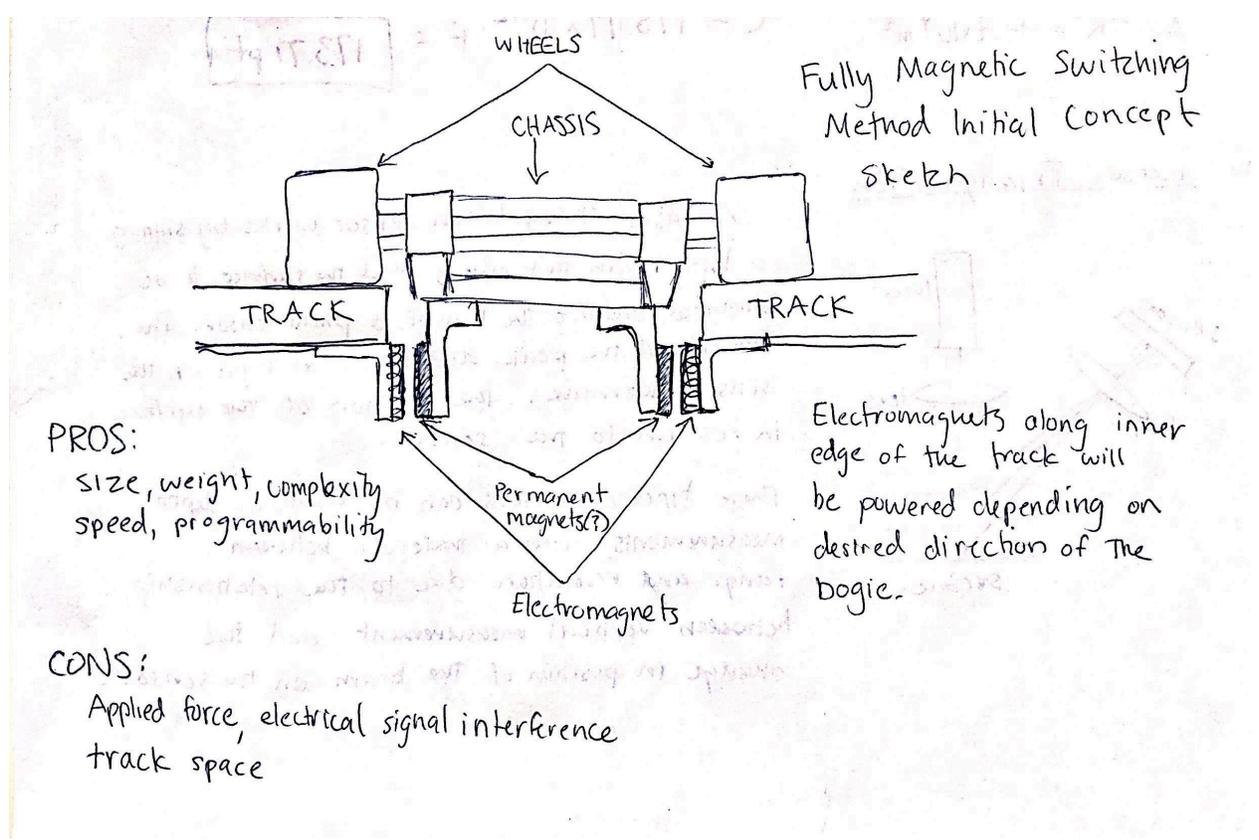
Petrova, A., Solovev, A. (2020, November 2). *What Is the MQTT Protocol and Why Choosing It for IoT Devices*. <https://www.integrasources.com/blog/mqtt-protocol-iot-devices/>.

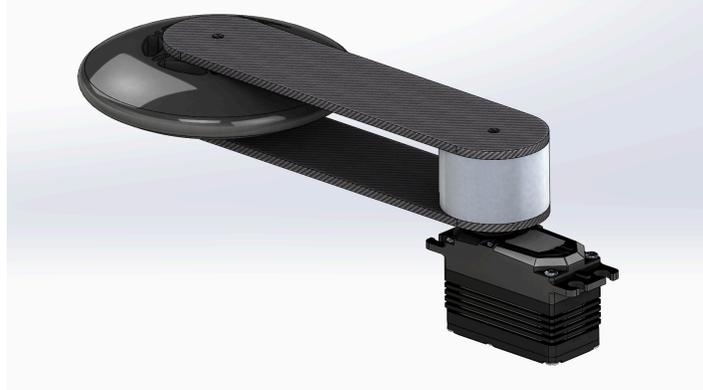
### Appendix

#### 13. Appendix (Add to your report as needed)

- a. Show the detailed calculations here.
- b. Include the detailed simulation results and computer programming codes here
- c. Given the detailed design drawings, dimensions with GD&T, and materials features
- d. Attach all datasheets of your intended-to-be-ordered components here (if applicable)







## CH340G Datasheet:

[https://datasheet.lcsc.com/lcsc/2008191806\\_WCH-Jiangsu-Qin-Heng-CH340G\\_C14267.pdf](https://datasheet.lcsc.com/lcsc/2008191806_WCH-Jiangsu-Qin-Heng-CH340G_C14267.pdf)

## Page 6 of the datasheet:

CH340 手册 (一)

6

外部电源退耦。

对于 CH340G/T/R 芯片，晶体 X2、电容 C6 和 C7 用于时钟振荡电路。X2 是频率为 12MHz 的石英晶体，C6 和 C7 是容量为 33pF 的独石或高频瓷片电容。如果 X2 选用低成本的陶瓷晶体，那么 C6 和 C7 的容量必须用该晶体厂家的推荐值，一般情况下是 47pF。对起振困难的晶体，建议 C6 容量减半。

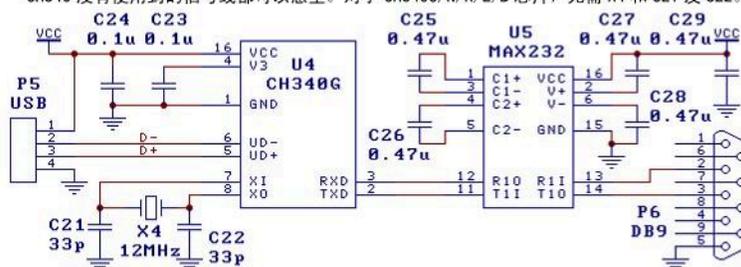
对于 CH340C/N/K/E/B 芯片，无需晶体 X2 和电容 C6 及 C7。

在设计印刷电路板 PCB 时，需要注意：退耦电容 C8 和 C9 尽量靠近 CH340 的相连引脚；使 D+ 和 D- 信号线贴近平行布线，尽量在两侧提供地线或者覆铜，减少来自外界的信号干扰；尽量缩短 X1 和 X0 引脚相关信号线的长度，为了减少高频干扰，可以在相关元器件周边环境地线或者覆铜。

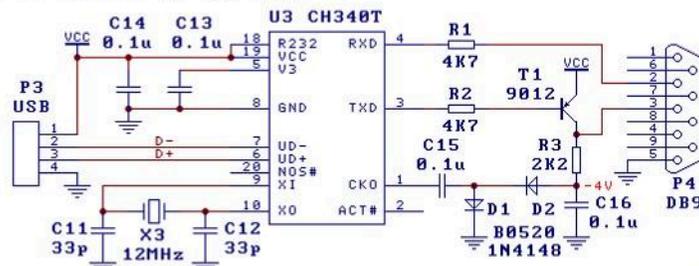
## 7.2. USB 转 RS232 串口 (下图)

图中是 USB 转最基本也最常用的 3 线制 RS232 串口，U5 为 MAX232/1CL232/SP232 等。

CH340 没有使用到的信号线都可以悬空。对于 CH340C/N/K/E/B 芯片，无需 X4 和 C21 及 C22。



## 7.3. USB 转 RS232 串口, 简版 (下图)



图中也是 USB 转 3 线制 RS232 串口，该电路与 7.2 节的功能相同，只是输出 RS232 信号的电平幅度略低。CH340 的 R232 引脚为高电平，启用了辅助 RS232 功能，只需外加二极管、三极管、电阻和电容就可代替 7.2 节中专用的电平转换电路 U5，所以硬件成本更低。

## 7.4. USB 转 RS485 串口

可以用 TNOV 引脚控制 RS485 收发器的 DE (高有效发送使能) 和 RE# (低有效接收使能) 引脚。

## 7.5. USB 红外适配器 (下图)

下图是由 USB 转 IrDA 红外芯片 CH340R 和红外线收发器 U14 (ZHX1810/HSDL3000 等类似型号) 构成的 USB 红外线适配器。电阻 R13 用于减弱红外线发送过程中的大电流对其它电路的影响，要求不高

## LM1117 Datasheet:

[https://www.ti.com/lit/ds/symlink/lm1117.pdf?ts=1621457781883&ref\\_url=https%253A%252F%252Fwww.google.pl%252F](https://www.ti.com/lit/ds/symlink/lm1117.pdf?ts=1621457781883&ref_url=https%253A%252F%252Fwww.google.pl%252F)

## Page 1 of the datasheet:

Product Folder
 Order Now
 Technical Documents
 Tools & Software
 Support & Community
 Reference Design

**LM1117**  
SNOS412O – FEBRUARY 2000 – REVISED JUNE 2020

---

**LM1117 800-mA, Low-Dropout Linear Regulator**

### 1 Features

- For a newer drop-in alternative, see the [TLV1117](#)
- Available in 1.8 V, 2.5 V, 3.3 V, 5 V, and Adjustable Versions
- Space-saving SOT-223 and WSON packages
- Current limiting and thermal protection
- Output current: 800 mA
- Line regulation: 0.2% (maximum)
- Load regulation: 0.4% (maximum)
- Temperature range:
  - LM1117: 0°C to 125°C
  - LM1117I: –40°C to 125°C

### 2 Applications

- AC drive power stage modules
- Merchant network and server PSU
- Industrial AC/DC
- Ultrasound scanners
- Servo drive control modules

### 3 Description

The LM1117 is a low dropout voltage regulator with a dropout of 1.2 V at 800 mA of load current.

The LM1117 is available in an adjustable version, which can set the output voltage from 1.25 to 13.8 V with only two external resistors. In addition, it is available in five fixed voltages, 1.8 V, 2.5 V, 3.3 V, and 5 V.

The LM1117 offers current limiting and thermal shutdown. Its circuit includes a Zener trimmed bandgap reference to assure output voltage accuracy to within ±1%.

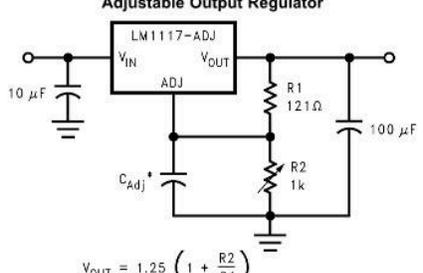
A minimum of 10-μF tantalum capacitor is required at the output to improve the transient response and stability.

#### Device Information<sup>(1)</sup>

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM1117, LM1117I	SOT-223 (4)	6.50 mm × 3.50 mm
	TO-220 (3)	14.986 mm × 10.16 mm
	TO-252 (3)	6.58 mm × 6.10 mm
	WSON (8)	4.00 mm × 4.00 mm
	TO-263 (3)	10.18 mm × 8.41 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

#### Adjustable Output Regulator



$$V_{OUT} = 1.25 \left( 1 + \frac{R_2}{R_1} \right)$$

\*  $C_{Adj}$  is optional, however it will improve ripple rejection.

---

An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

## Appendix #

```

/* BASIC_ESC_TEST.ino
 * This code is for testing our ESC and motor setup of the Switch Arm
 * Jake Espinoza, 3/10/21
 */
#include <Servo.h>

Servo monsterESC

const int SERVO_PIN = 9;

value = 0

void setup() {
  monsterESC.attach(SERVO_PIN);
  Serial.begin(9600); // start serial at 9600 baud
}

void loop() {
  monsterESC.write(value); // for cont. rot. servo (what we have)
  // Use values 0-89 for forward, 90 for stop, 91 - 180 for reverse
  if(Serial.available())
    value = Serial.parseInt(); //set a new value in serial
}

/* encoder_spi.ino
 * This code will interface with the AS5047P encoder used for position sensing.
 * This sensor uses SPI to communicate the angular position
 * Functions are included to read the most frequently used registers
 * These are the angle, error, and diagnostic registers
 *
 * By: Jake Espinoza
 * v1 --- March 1, 2021
 */

#include <SPI.h>
#include <Arduino.h>

```

```

//#include <Servo.h>

// Target angles
#define LEFT_TARGET    90; // degrees
#define RIGHT_TARGET   270; // degrees

// ENCODER STUFF
const int ENCODER_PIN =          10;
// volatile registers
const int NONOP_REGISTER =       0x0000; // no operation
const int ERROR_REGISTER =      0x0001; // 0b000000000000001 -- error register
const int PROG_REGISTER =       0x0003; // programming register
const int DIAAGC_REGISTER =     0x3FFC; // 0b11111111111100 -- diagnostics and AGC
const int CORDIC_MAG_REGISTER = 0x3FFD; // CORDIC magnitude
const int ANGLE_UNCOMP_REGISTER = 0x3FFE; // Uncompensated error value
const int ANGLE_COMP_REGISTER = 0x3FFF; // 0b11111111111111 -- compensated error
value
// non-volatile registers (ONE TIME PROGRAM)
const int ZERO_MSB_REGISTER =   0x0016; // Zero position MSB
const int ZERO_LSB_REGISTER =   0x0017; // zero position LSB and MAG Diagnostics
const int SETTINGS1_REGISTER =  0x0018; // custom setting register 1
const int SETTINGS2_REGISTER =  0x0019; // custom setting register 2
// most used commands
#define READ_ANGLE_COMMAND      0b1111111111111111
#define READ_ERROR_COMMAND      0b0100000000000001
#define READ_DIAGC_COMMAND      0b1111111111111100

void setup() {
    // set the encoder as an output:
    pinMode(ENCODER_PIN, OUTPUT);
    // initialize SPI:
    SPI.begin();
}

void loop(){
    delay(1000); //a 1 sec delay so everything can start up

    // THIS IS WHERE WE PUT CODE TO TEST STUFF

}

int read_angle() {
    // read angle from AS5047P
    // uses SPI_MODE1 (CPOL = 0, CPHA = 1 ) & even parity bits at MSB (MSB FIRST) @
10 MHz
    // angle data is read from Dynamic Correction register
    // this data is a 14-bit value with the 2 MSB not used

    static int angle;

    // send read command w/ angle register & receive 14 bit angle value (MSB FIRST)
    angle = SPI.transfer16(READ_ANGLE_COMMAND);

    return angle;
}

```

```

}

// NEEDS TO BE WRITTEN TO GIVE 2 VALUES OUT
// MAYBE USE A STRUCT OR CLASS
int recover_telemetry(){
    // reads and returns the contents of error register
    // error register is emptied after it is read
    // error register contains 3 bits (see datasheet for errors)
    static int telemetry[2];

    // send read command w/ error register & receive 16 bit error value (MSB FIRST)
    telemetry[0] = SPI.transfer16(READ_ERROR_COMMAND); // read error register into
first index
    telemetry[1] = SPI.transfer16(READ_DIAGC_COMMAND); // read error register into
first index

    return telemetry;
}

int generate_command(int address ,int rw=1){
    // this function will automatically generate a command frame for the AS5047P
    // the AS5047P uses 16-bit frames MSB->LSB with even parity bit in MSB(15)
    // Read/Write command is in 2nd MSB (14). 1 = read, 0 = write
    int on_bits;
    String command_str;
    String parity_bit;
    String binary_address = str(int(address, 2));

    command_str = str(rw) + binary_address;
    on_bits = count(command_str.begin(), command_str.end(), 1);

    if(on_bits % 2 == 0){
        command_str = str('0') + command_str;
    }
    else{
        command_str = str('1') + command_str;
    }

    return int(command_str, 2);
}

/* SerialToMove.ino
 * Receiver code
 * This code should run on the SA PCB to receive commands from the Bogie MCU
 * Code for controlling Switch-Arm via serial commands
 * Jake Espinoza, 4/19/21
 */

#include <EEPROM.h>
#include <Stepper.h>

```

```

#define HOMED_FLAG_EEPROM_ADDRESS 0

volatile int buttonState = HIGH;

const int ledPin    = 13; // the pin that the LED is attached to
const int buttonPin = 2; // CHANGE THIS TO RIGHT PIN. IT MUST HAVE INTERRUPTS ON IT
-> PIN 2 has this on most arduino boards
int incomingByte;      // a variable to read incoming serial data into
char event = 'X';      // event variable X=no event, R=right command, L=left
command
char state = 'X';      // state variable X=no event, R=right command, L=left
command

// Stepper stuff
const int stepsPerRev = 800;          // CHANGE THIS TO RIGHT AMOUNT
const int stepsToRight = (stepsPerRev*30); // CHANGE SIGN AS NEEDED
const int stepsToLeft = -(stepsPerRev*30); // CHANGE SIGN AS NEEDED
const int maxSpeed    = 1000;        // CHANGE THIS TO RIGHT AMOUNT
const int homingSpeed = 10;          // CHANGE THIS TO RIGHT AMOUNT
Stepper stepper(stepsPerRev, 3,4,5,6); // CHANGE PINS TO RIGHT PINS

// FUNCTIONS
// ISR function for detecting change in button state
void buttonInt(){
  buttonState = digitalRead(buttonPin);
}

// this function moves stepper a specified amount of steps in increments defined
within the function
// after each increment of steps is moved, the function checks for
void stepperMove(int stepsToMove){
  const int stepIncrement = stepsToMove/10;
  int stepsRemaining = abs(stepsToMove);
  stepper.setSpeed(maxSpeed);

  // begin moving if steps are left
  while (stepsRemaining > 0){
    stepper.step(stepIncrement);
    stepsRemaining -= abs(stepIncrement);
    readByte();
  }
}

// this will move the stepper a defined number of steps (Default=10)
// then check to see if the button has been pressed. When a button is pushed, the
motor will stop stepping and the
// EEPROM ADDRESS is set to 1. during setup, this address is checked and if it is 1
it will not run this home code
// calling this home code resets the EEPROM home variable and gives you a chance to
reset it
// EEPROM can only be changed ~100,000 times
// limit switches should be implemented asap. It will simplify the process of moving
// as well. You can just tell the stepper to move until it hits the limit switch
// limit switches could be set up to use interrupts too

```

```

void setHome(int stepIncrement = 10){
  // check EEPROM homed flag, set to 0 if it is 1
  EEPROM.update(HOMED_FLAG_EEPROM_ADDRESS, 0);
  stepper.setSpeed(100);

  while (true){
    if (buttonState == LOW){
      state = 'L';
      EEPROM.update(HOMED_FLAG_EEPROM_ADDRESS, 1);
      return;
    }
    else{
      stepper.step(200);
      delay(100);
    }
  }
}

// we may run into issues if commands get sent too quickly
// may need to use a software interrupt for this somehow
char readByte(){
  // read and return the oldest byte in the serial buffer:
  return Serial.read();
}

void setup() {
  Serial.begin(9600);
  Serial.println("serial");
  delay(1000);
  Serial.println("Sup");

  // set up for interrupts
  pinMode(buttonPin, INPUT_PULLUP); //button set to HIGH w/ pullup
  attachInterrupt(digitalPinToInterrupt(buttonPin), buttonInt, HIGH); //attaches
  buttonInt as ISR on buttonPin //triggered by
  buttonPin falling high
  // check if stepper has been homed
  Serial.println("check home");
  if (EEPROM.read(HOMED_FLAG_EEPROM_ADDRESS) == 0) {
    setHome();
  }

  // initialize serial communication THIS SHOULD BE AS FAST AS POSSIBLE IDEALLY
  // WE MAY NEED TO DO STUFF FOR MCU TO MCU SERIAL COMMUNICATION
}

void loop() {
  // see if there's incoming serial data:
  if (Serial.available() > 0) {
    Serial.println("event recieved");
    event = readByte();
  }
}

```

```

}

if (event == 'R'){
  Serial.println("event R");
  if (state == 'R'){
    ;
  }
  if (state != 'R'){
    Serial.println("Moving R");
    stepperMove(stepsToRight);
    state = 'R';
  }
}
else if (event == 'L'){
  Serial.println("event L");
  if (state == 'L'){
    ;
  }
  if (state != 'L'){
    Serial.println("moving L");
    stepperMove(stepsToLeft);
    state = 'L';
  }
}

else if (event == 'H'){
  Serial.println("Starting home");
  setHome();
}

else if (event == 'X'){
  ;
}
}

```

```

/* SLAVE_ADJUST.ino

```

```

*This code is designed to fine tune the angle position.

```

```

*Pins 7 and 6 should be connected to two digital pins on another microcontroller.

```

```

*The Master controller should have ways to manually change the state of pins, like
buttons.

```

```

*This way the angle can be adjusted slowly to make sure the arm has the best angle.

```

```

*Andrew Poli, 5/13/21

```

```

*/

```

```

//Include the Arduino Stepper Library

```

```

#include <Stepper.h>

```

```

const float STEPS_PER_REV = 800; //Steps per full rotation.

Stepper steppermotor(STEPS_PER_REV, 1, 2, 3, 4); //Set Stepper pins to 1-2-3-4

void setup() {
  pinMode(7,INPUT_PULLUP); //Begin monitoring pins 7 and 6
  pinMode(6,INPUT_PULLUP);
}

void loop() {
  steppermotor.setSpeed(500); //Sets motor speed
  if (digitalRead(7)==HIGH){ //When pin 7 is high, rotate CW
    steppermotor.step(100); // Step count is small for better adjustments
  }
  if (digitalRead(6)==HIGH){ //When pin 6 is high, rotate CCW
    steppermotor.step(-100);
  }
}

/* SLAVE_FINAL.ino
 * This code rotates the stepper motor 180 based on the state of a digital pin (7).
 * The microcontroller should be connected to a master controller to receive commands.
 * Currently the arm will rotate when the state of the pin changes.
 * Andrew Poli, 5/13/21
 */

//Include the Arduino Stepper Library
#include <Stepper.h>
#define HALL_SENSOR 0

const float STEPS_PER_REV = 800;

// Amount of Gear Reduction
const float GEAR_RED = 30;

// Number of steps per geared output rotation
const float STEPS_PER_OUT_REV = STEPS_PER_REV * GEAR_RED;

```

```
// Define Variables

// Number of Steps Required
int StepsRequired;

Stepper steppermotor(STEPS_PER_REV, 1, 2, 3, 4);

int previous;
bool start_turn = false;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(7, INPUT_PULLUP);
  pinMode(6, INPUT_PULLUP);
  pinMode(HALL_SENSOR, INPUT);

  while (digitalRead(HALL_SENSOR) != LOW) { // Begin homing. Will turn until in left
position.
    steppermotor.setSpeed(500);
    steppermotor.step(100);
  }
  previous = digitalRead(7); //Records pin state to begin checking for change.
}

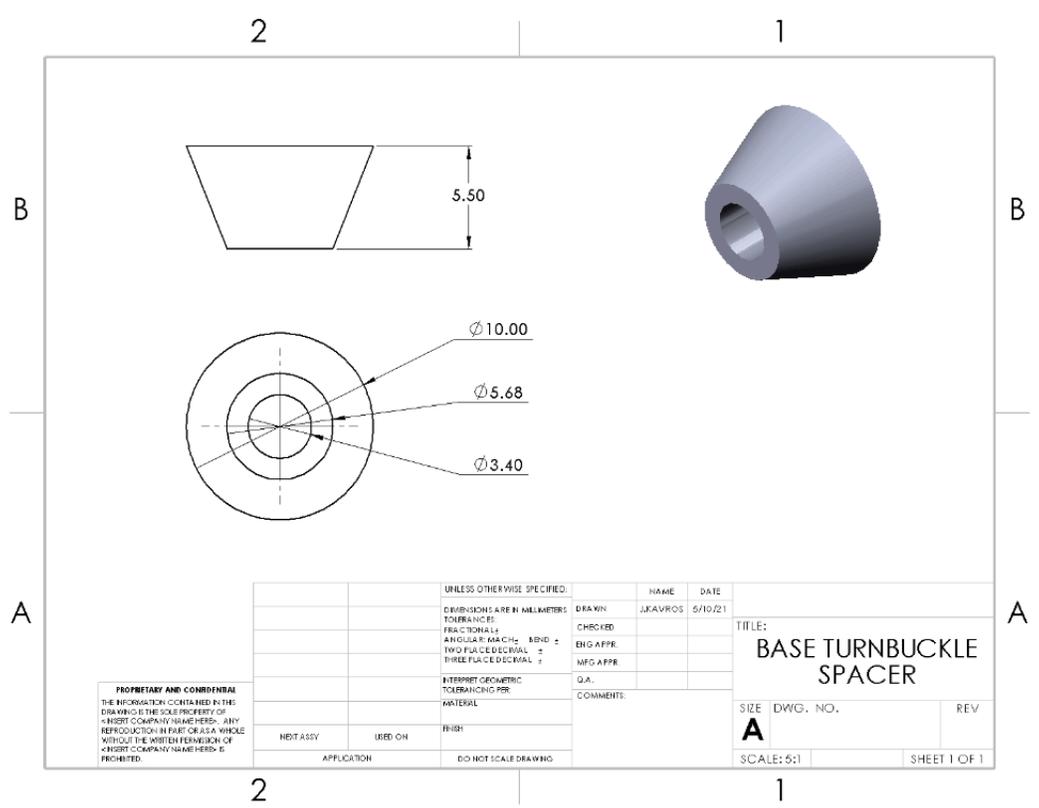
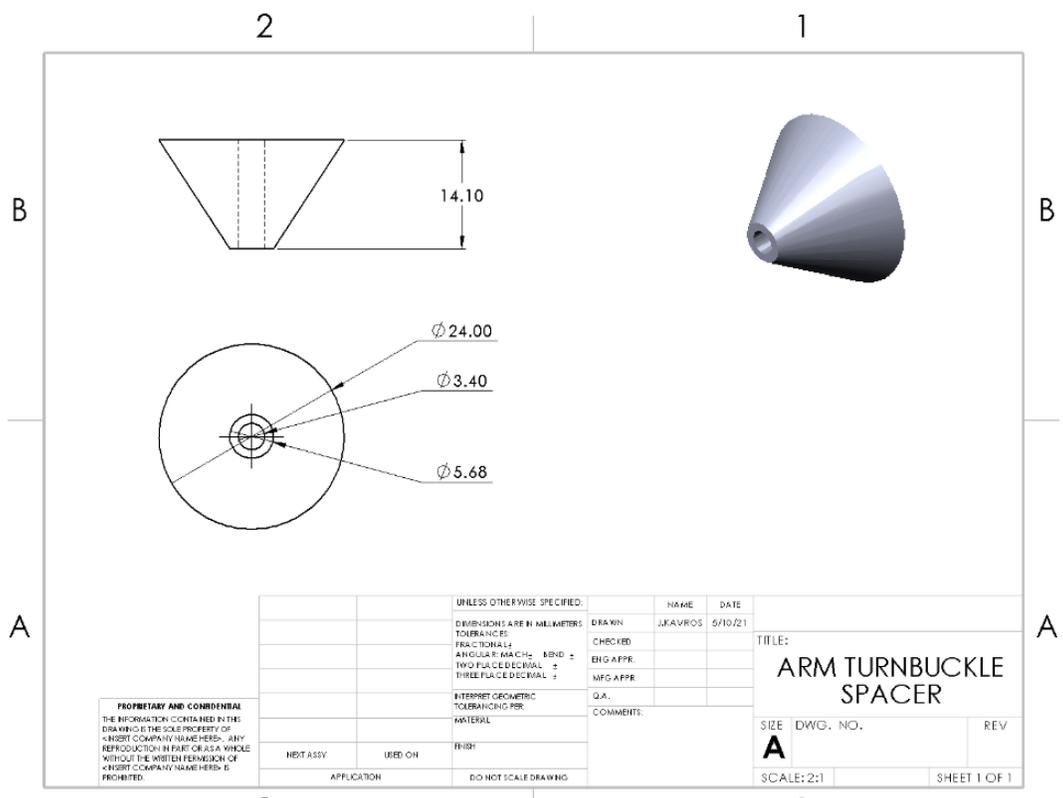
void loop() {

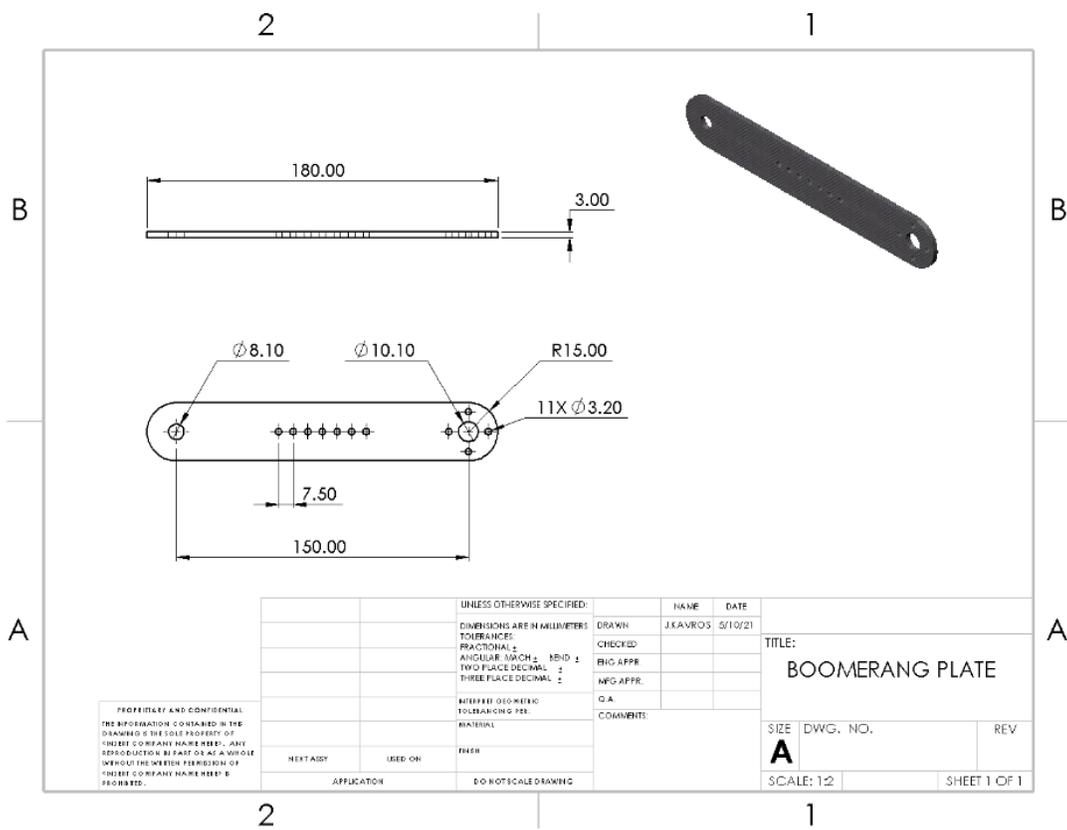
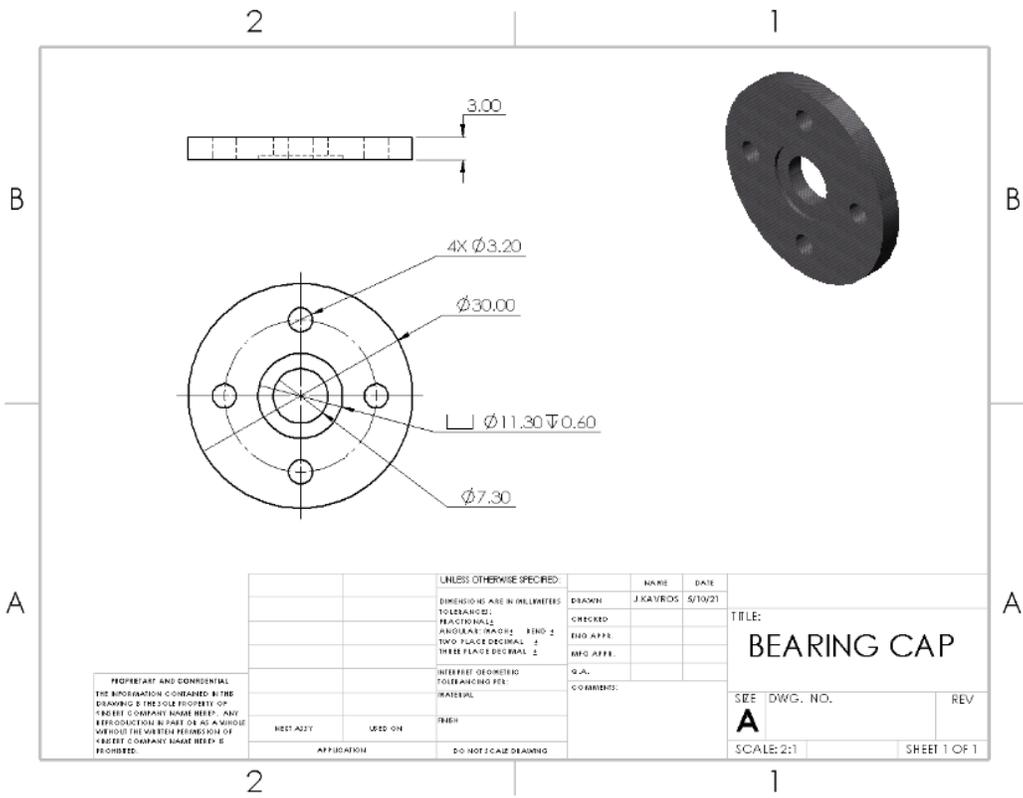
  // Checks to see if the signal pin has changed states
  if (digitalRead(7) == previous) {
    delay(100); // If signal pin is the same, continue
  } else {
    start_turn = true; //IF pin changed, set turn to true.
    previous = digitalRead(7); //Reset pins by recording state.
  }

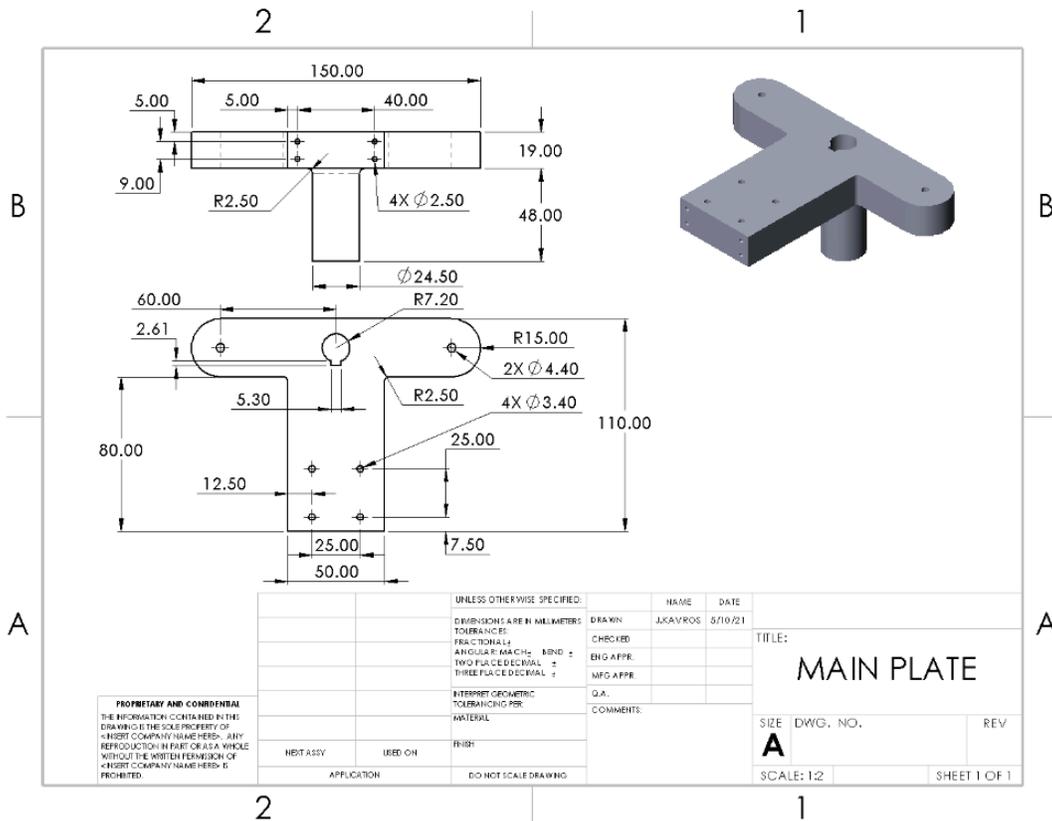
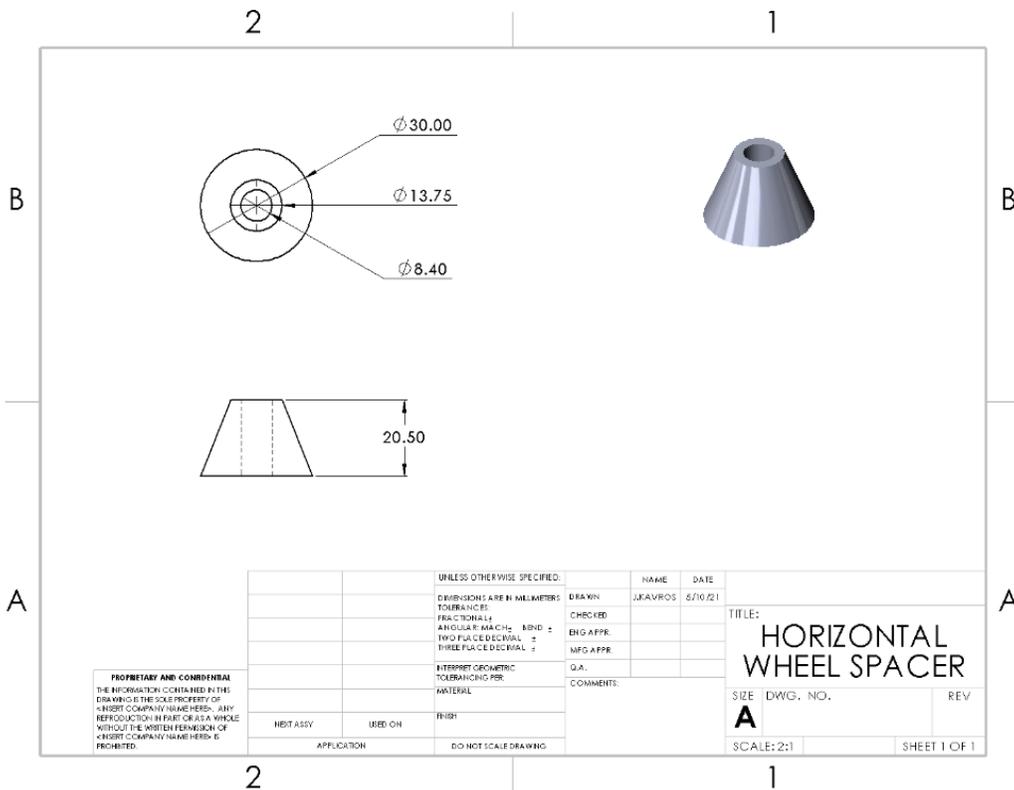
  if (start_turn) { //if start_turn is true after being set, turn...
    StepsRequired = STEPS_PER_OUT_REV; //Sets steps required for 180 turn.
    steppermotor.setSpeed(1000); //Sets speed.
  }
}
```

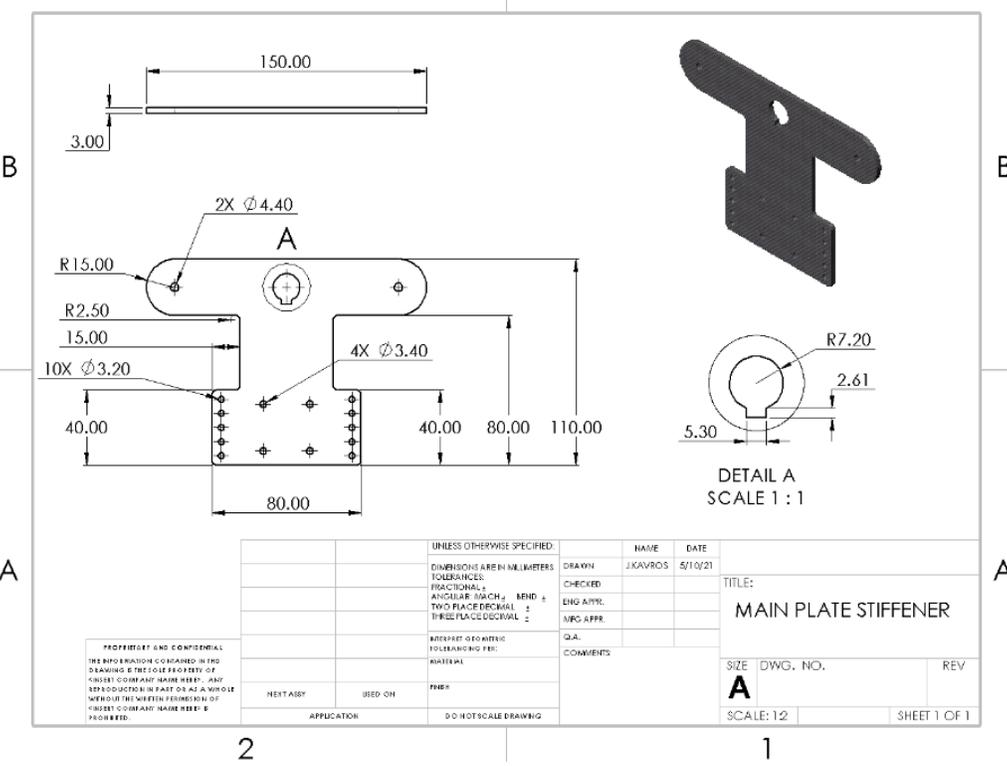
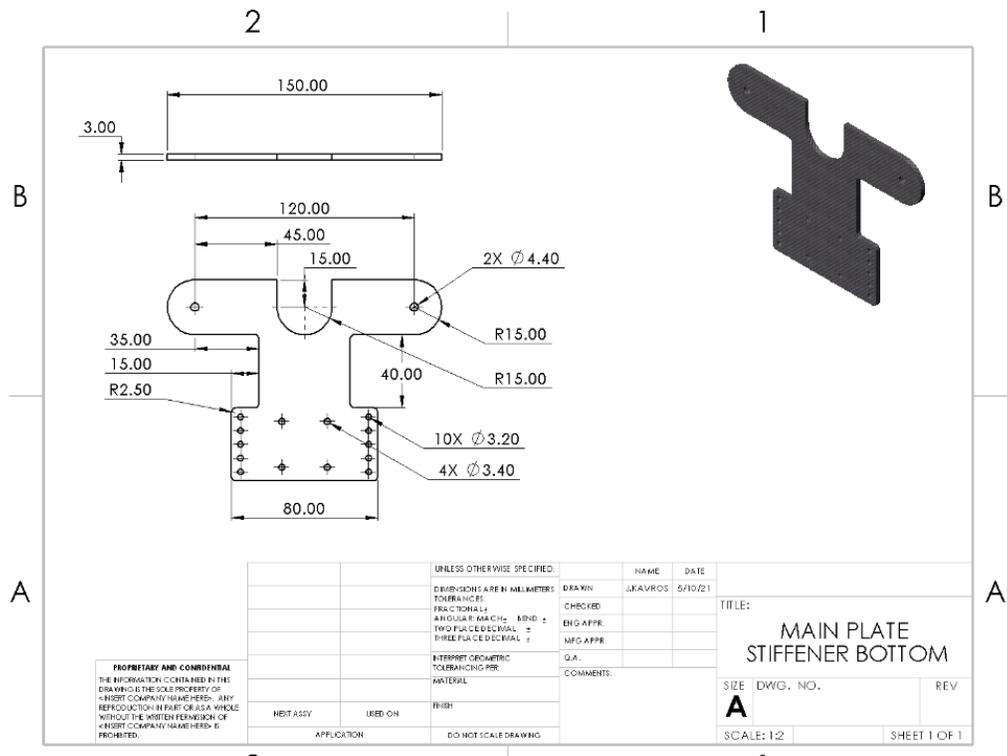
```
steppermotor.step(StepsRequired); //Begin turn

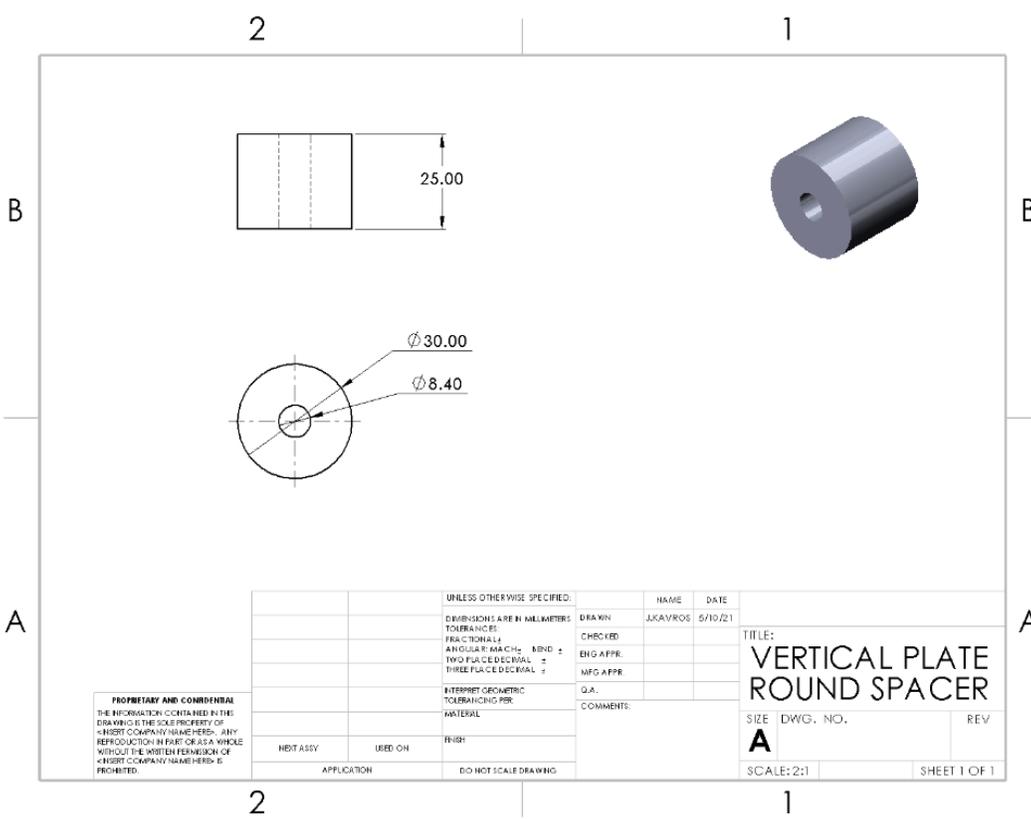
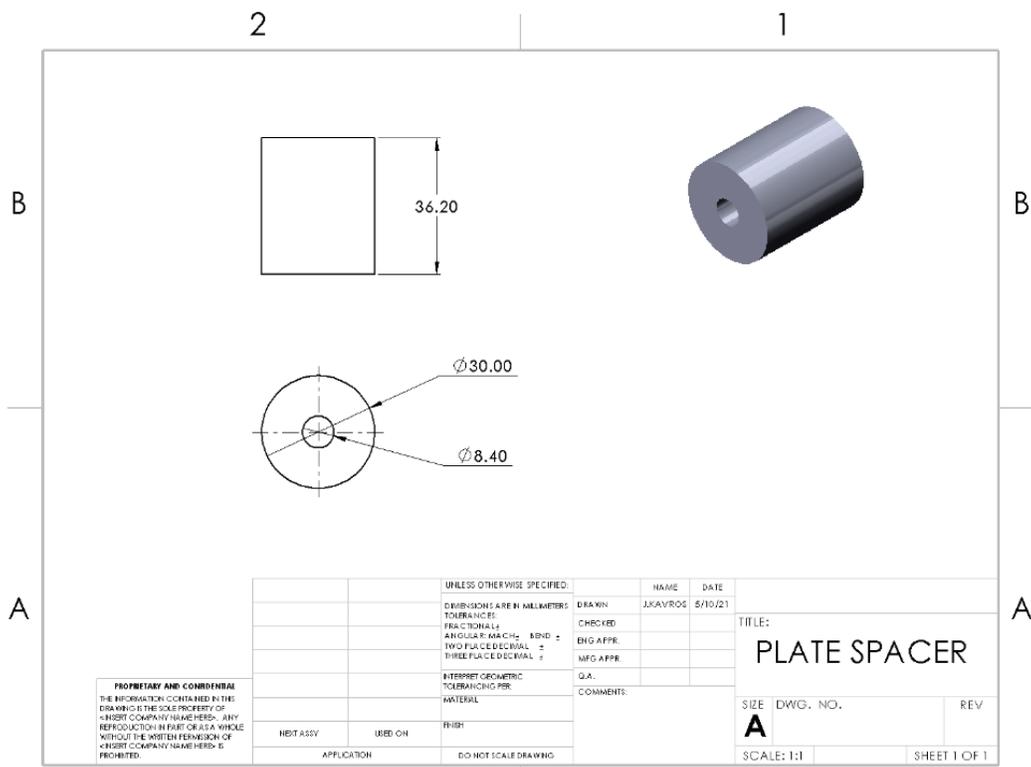
delay(500); // Delay for good manners.
start_turn = false; // Set start_turn to false arm stops.
}
}
```

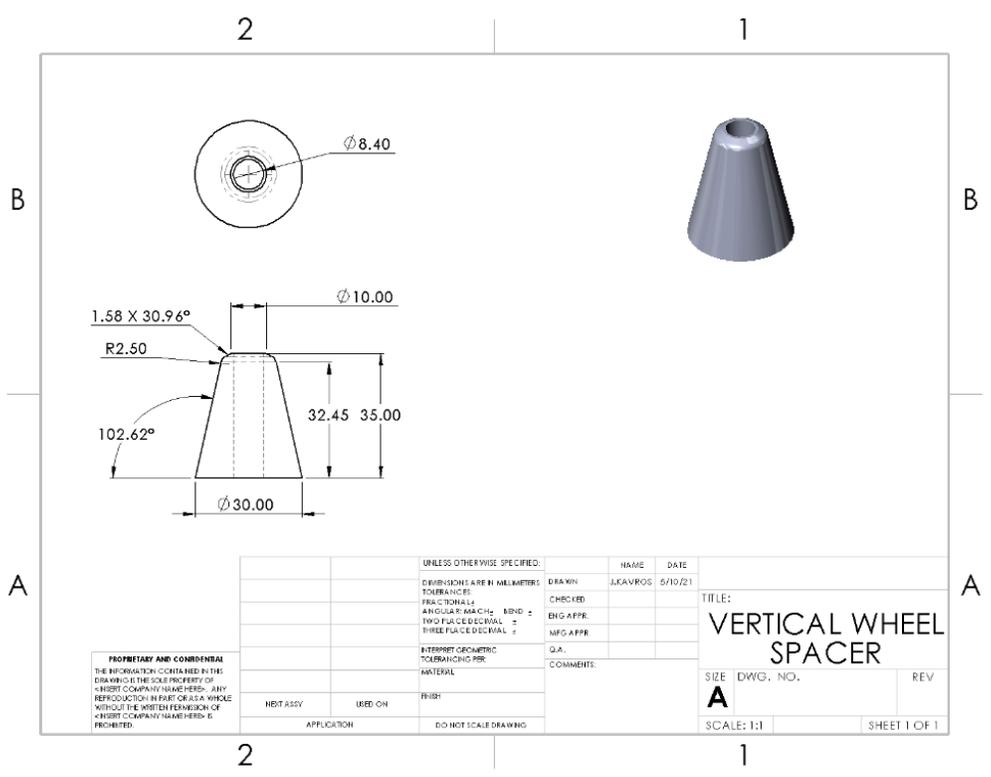
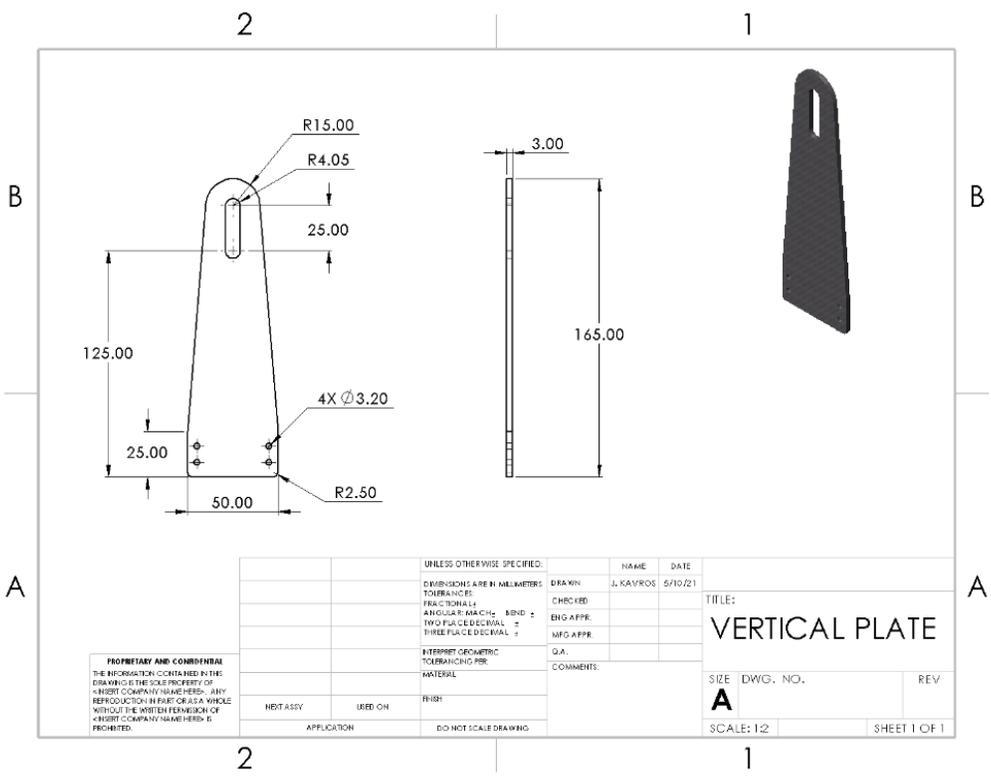






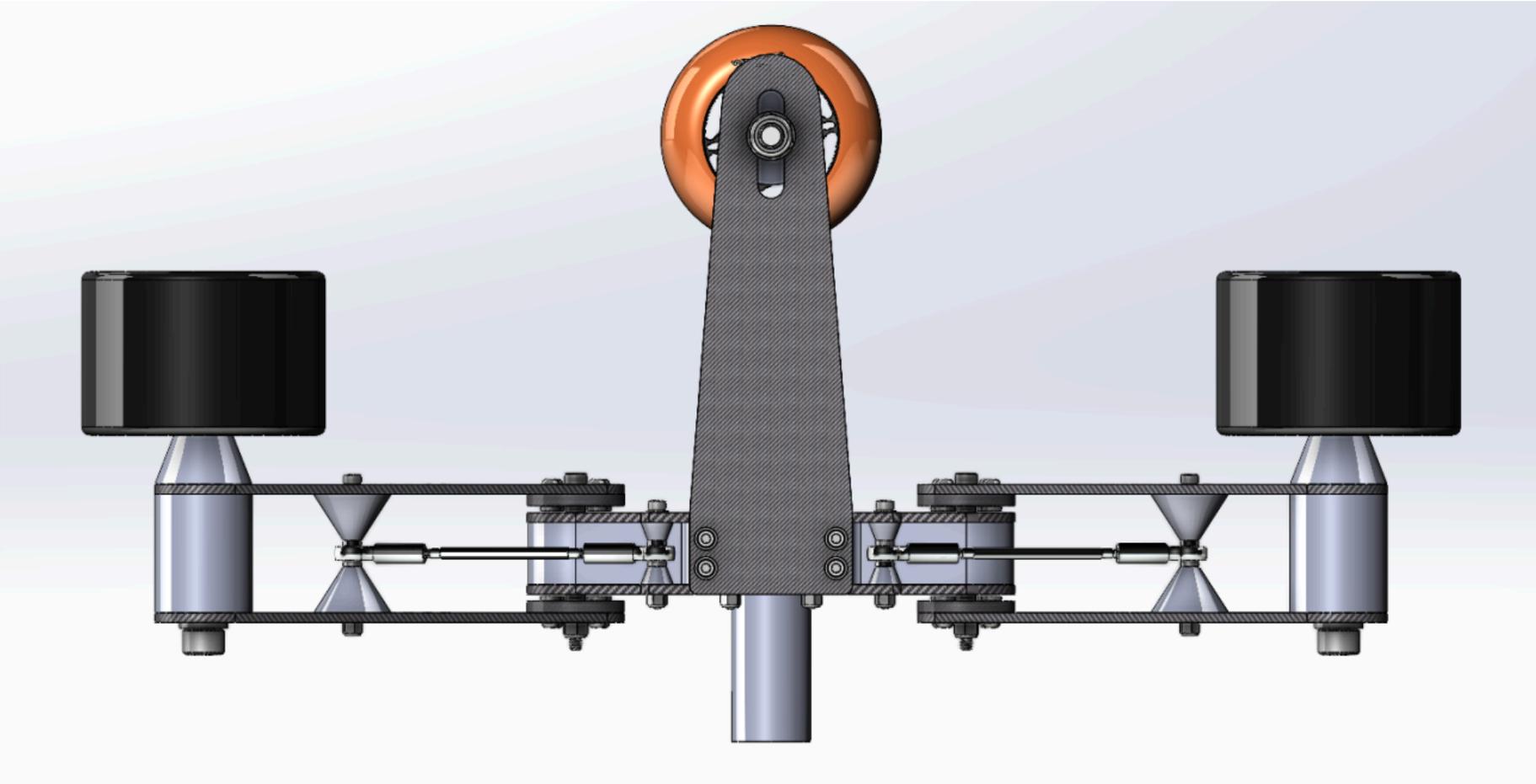




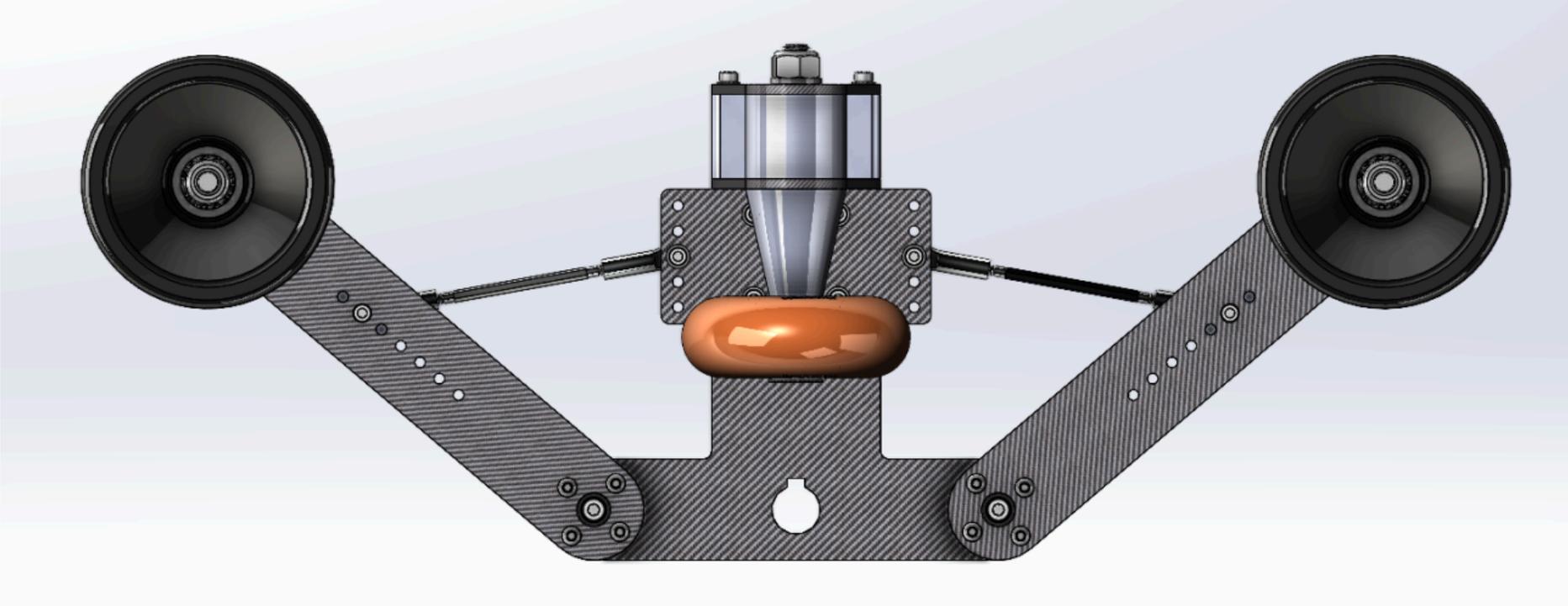




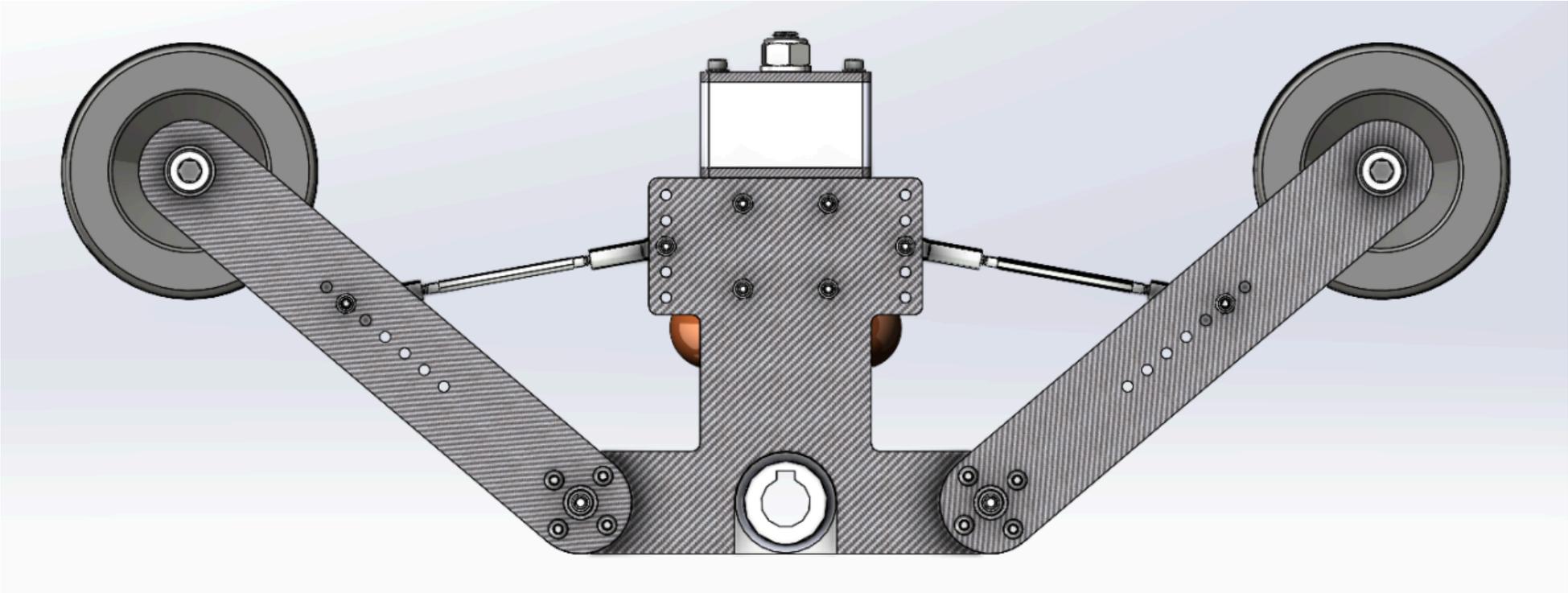
Front View



Top View



Bottom View



Side View

