A Solar Powered Automated Public Transportation System

San Jose State University

College of Engineering

May 23rd, 2014

1st Edition, Vol. 2

Table of Contents

Table of Figures

Table of Tables

Spring 2014 Goals & Accomplishments

Once the Spartan Superway Team received confirmation that they would present their work at Bay Area Maker Faire 2014, each team needed to reassess their semester goals in order achieve a physical deliverable that could be transported to San Mateo for the event. For all of the teams, this meant scaling down the theoretical designs in some fashion.

The Bogie Team was able to complete a full-scale bogie chassis that is the most representative of the theoretical design in relation to the other teams. The bogie uses the actual wheels incorporated in the theoretical model and was fabricated from laser-cut steel plate, as well as tube steel. The bogie only lacks a fully functional steering mechanism, motor assembly, and the point at which it attaches to the cabin was scaled down due to the smaller mock cabin incorporated for the Maker Faire display.

Although the Guideway that the bogie rides along is also full-scale, it does not sit at the full-scale height, nor do the supports represent the full-scale model as. The Guideway Team's Maker Faire model required the most significant changes in regards to theoretical design due to safety and transportation logistics. Instead of fabricating the guideway from steel, the team opted to use wood instead. Not only did this choice make transporting the guideway significantly easier, it also provided a safer exhibit be dramatically reducing the likelihood that the guideway might tip. Redesigning new supports for the guideway also diverted the team's development of the theoretical model. Instead of performing structural analysis of the theoretical model, the team had to focus on the shorter steel supports for the wooden guideway to ensure it would not tip, even when fully loaded and experiencing 27 mph winds.

Instead of fabricating a full-scale cabin, complete with a steel frame, the Cabin Team had to redirect its efforts into producing a cabin that could easily attach to the bogie, without subjugating the guideway to too great of an additional load. By retrofitting a towable wagon designed for snowmobiles, the Cabin Team was able to present a streamlined cabin that was representative of the ATN concept to Maker Faire spectators.

Similarly, the Solar Team had to modify their theoretical solar panel mount to fit the new guideway supports, and also to accommodate solar panels that were donated by a sponsor, rather than implement the solar panels specifically chosen for the theoretical model. Additionally, they opted to fabricate a static solar panel mount due to the fact that developing a new solar tracking system for a one weekend event was both time and cost-prohibitive.

The Controls Team did not need to scale down their design to the same extent as the other teams, primarily due to the fact that safety and transportation were less an issue. The only transportation problem arose from the limited setup time prior to the event. With only a two day window to setup, the Controls Team chose to operate the pods on batteries rather than rely on the 1/12-scale solar power transmission system. The only other significant modification was to use fewer microcontrollers on each pod for better reliability in operation and communication.

Team Roster

Project Advisor: Dr. Burford Furman

Mentors: Maria Blum-Sullivan, Bryan Burlingame, Sam Ellis, Lizie Michel, and Ron Swenson

Program Managers: Alex Cowley and Jaston Rivera

Bogie Team: Max Goldberg (Lead), David Lhotak, and Paolo Mercado

Brochures: Eugenia Tai –Civil Engineer, Lead Creative Designer, Graphic Design Specialist

Business Team: Laisz Lam

Cabin Team: Alex Cowley (Lead), and Ken Ho

Controls Team: Cory Ostermann (ME Lead), Eriberto Velazquez (CMPE Lead), Man Ho, Marjo Mallari, Randall Morioka, Elizabeth Poche, Trent Smith, and Anthony Vo (Spartan Superway Club member)

Human-Centered Design Team: Maria Blum-Sullivan (Lead), Alex Cowley, and Ken Ho

Report Review Team: Alex Cowley, Cory Ostermann, Jaston Rivera

Solar Team: Francisco Martinez (Lead), Jaston Rivera, Tim Santiago, and Henry Tran

Special Projects Lead: Keith McKenna – Civil Engineer Graduate Student

Station/Guideway Team: Cormac Wicklow (Lead), Daniel Conroy, and Carlos Guerrero

Pod and Station Industrial Design Team: the students of DSID 125: Advanced Industrial Design, led by Jim Shook.

User-Interface/User-Interaction Design Team: the students of DSID 131: Interactive and Interface Design, led by Tingbin Tang.

Website Design Lead: Francisco Martin

Club Officers

President: Cory Ostermann

Vice President: Tim Santiago

Treasurer: Jaston Rivera

Secretary: Henry Tran

Sponsorships / Acknowledgments

Guideway Team

Theoretical System Model

At the beginning of the semester, the working design was very similar to the design from the fall 2013 semester. That design is based on an I-beam, as shown in Figure 1. In the process of analyzing this design, the guideway team discovered a number of flaws with this design. The main shortcoming involved the switching mechanism, as mentioned in the previous section.

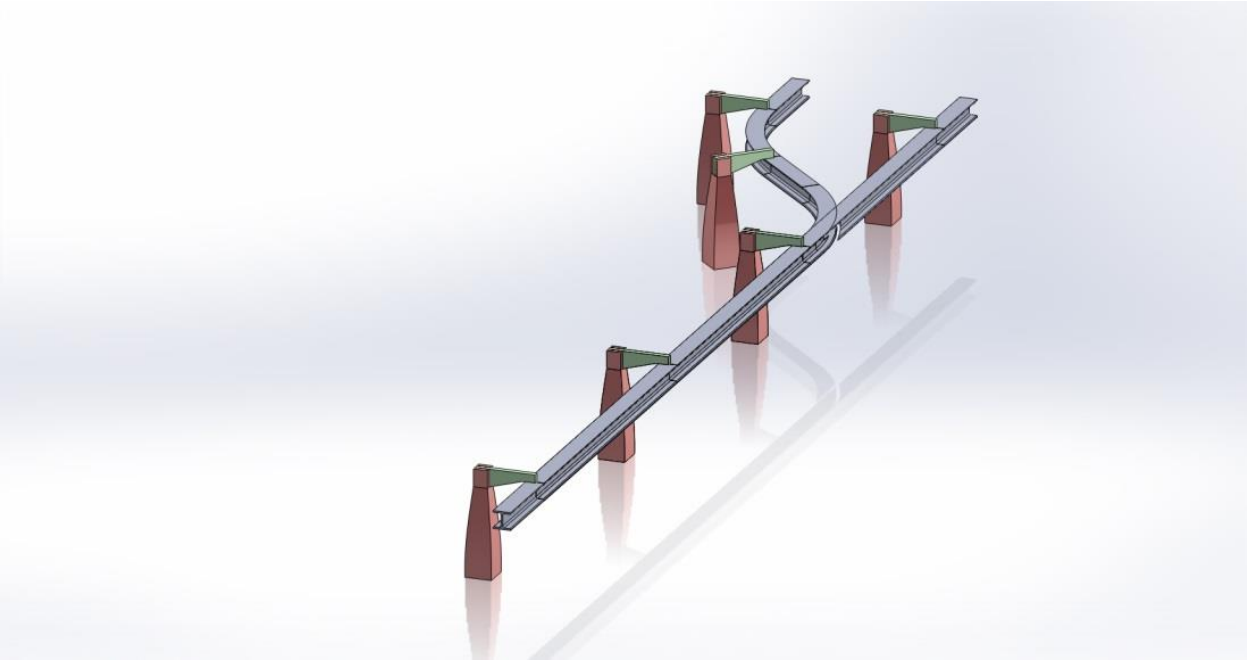


Figure 1. Previous guideway design

At this point, a radical redesign is required. The guideway team was fortunate enough to connect with Bengt Gustafsson of Beamways AB. This is a Swedish company with a number of years invested in creating a personal rapid transit system (PRT) for use in Sweden. Due to the influence of the advisors and professors in the SMSSV team, Mr. Gustafsson flew out to meet with the team for a few days in early February. It was during this time that the guideway team discussed a number of possible options for the guideway. Due to his experience in the field and shear breadth of design work already completed, the team decided to emulate his design, henceforth known as the Beamways Superways Guideway. There are a number of advantages for this design, which is a single rail design with traverse supporting rails, the profile of which is shown in Figure 2. Also shown is an optional covering, which is to be discussed later.

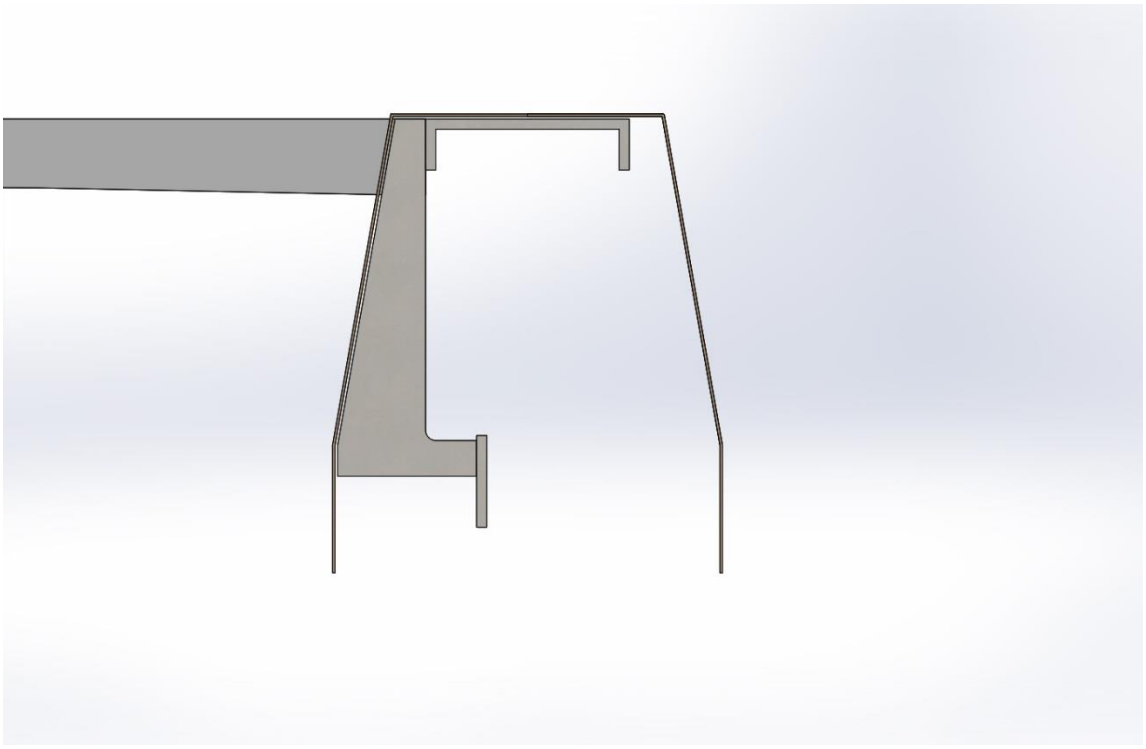


Figure 2. Beamways Superways Guideway profile

The first major advantage of this design is the shear simplicity of it. The entire profile can be made of 20mm steel, without heavily or costly I-beams. The weight of the design is much lower than the previous semester's model, which is shown in Figure 1. To effectively create a guideway, more than just straight sections are required. Nearly every rail system requires curves, which are difficult to create with I-beams. This model uses thin steel, which is easy to bend and form as desired. A major disadvantage of the previous model involved the cantilever sections as mentioned in a previous section of this report. The Beamways Superways Guideway profile eliminates the need for a cantilever overhang during switches, which will be discussed further through greater examination of this design. Due to a number of time constraints, this guideway design has not been fully completed. As mentioned in the introduction, there are a number of sacrifices which were made during the semester. By the time the current cross-section was decided on with Mr. Gustafsson, only two months were left until the end of the semester. The focus was split between designing this theoretical model, and building the wooden model for display at Maker's Faire. After discussion and weighing the benefits of both courses of action, it was decided that the focus should be on creating a full-scale model for display. This would provide a tangible object to educate the public on the idea of PRTs as a whole, and would benefit the cause more than a fully fleshed out theoretical design. Because of this, the guideway team spent a majority of its resources and time analyzing and building the wooden model which is detailed in the next section of this report. A working computer model of the theoretical design is completed, but full analyses were not performed on this design due to the above mentioned time constraints. The bogie team, working concurrently with the guideway team, created a working design for the switching section, which is a simplified design of Bengt Gustafsson's Beamways system. Because of this, the switching mechanism is not defined by the guideway team, and should be completed by future teams working on this project. The completed

theoretical design contains a computer model of the proposed design for a straight section of track, along with the support structures.

As mentioned earlier, the design is utilizes a single rail with traverse supporting walls located above. As shown with the current bogie design, the main load is focused on the single rail. It is made of 20mm thick steel, which is 180mm tall. For each section of the bogie, this piece must support the main drive wheel, which rides above rail, as well as a switching wheel located on either side of the lower portion. The highest lateral stresses on this rail will occur during switching sections, when the two lower wheels are used to change the course of the bogie. During straight and curved sections, the main lateral load is supported by the traverse walls near the top of the track. The weight which must be supported is approximately 300 pounds for each section of the bogie, plus the amount of weight of the cabin and passengers. Due to the nature of the priorities during this semester, a definite amount of weight to be supported is not known. Because of this, as well as other issues, the guideway team is not able to provide analysis for the loads supported by this guideway model.

The upper and lower rails are attached by a series of steel ribs, which are mounted along the guideway every 200mm. These provide lateral and torsional stability to keep the rails inline. The full design of this Beamways Superways Guideway is shown below in Figure 3, which contains the rails, ribs, covers, and supports. All of the steel pieces are created with the same 20mm steel plates, which can be fabricated using laser cutting to provide relatively cheap and quick parts. For the main sections, there are no components which require special machining or processing. The rails are connected to the ribs with a series of welds, which provide the highest possible strength, at the cost of reparability. As mentioned in the previous section regarding design requirements, the guideway is designed to last for 20 years without replacement, and only preventative maintenance. However, the PRT systems will be useful long after this amount of time has elapsed, so further solutions will need to be developed in the future.

As shown in Figure 3, there are two sets of rails operating from the same set of supports. In practice, this would allow pods to travel in opposite directions without needing to create excess infrastructure. The two rails are attached with a steel tube, and welded into place. The steel cross beam is placed into a notch in the support and bolted into place. This is so that the supports can be erected while the rails are assembled on the ground or offsite. This allows the rails to be hoisted into place, and attached into place.

Figure 3. Guideway straight section

The supports themselves are made with thick steel tubing, which is sunk into the ground to provide maximum stability. To create each support, a 20” hole is dug into the ground, and concrete is poured in, with a cavity to place the steel vertical tube. At this point, the exact dimensions of the steel tube are unknown, and will require further discussion and calculations. To provide a safe and effective design for the supports, a civil engineer should be consulted. This is because the design involves a large amount of concrete and interface with the ground, which is better suited for other disciplines. The current guideway team is composed of mechanical engineers, which do not have sufficient experience with these types of projects.

As shown in Figure 3, there is an optional cover place on the rightmost rail. This is created to provide relief from environmental effects. Steel will exhibit some amount of rust when exposed to the elements, and does look attractive at this point. To provide a beautiful skyline, the metal can be covered with a painted façade which can be refurbished at regular intervals. This also keeps small animals and birds from entering the track as easily. In the same vein, the ribs do not provide a sufficient surface area for birds and small animals to nest on the track.

Maker Faire Model

Engineering of Support Structure

The Guideway was purposely overbuilt due to the complex nature of wood as a material. Additionally the complex internal structure of the guideway made mathematical and computer analysis beyond what was capable. A broad isotropic analysis was completed in Solidworks and the results indicated that with

the additional material and fasteners used the guideway would be capable of supporting the loads without failure. The verification of these assumptions were done during the assembly of the system, which is discussed in a subsequent section.

It was determined that the Support Structure would be the focus of the Engineering for the system. There were several concerns for failure of these supports and it was necessary to account for aspects to insure the safety of the public and the team. The scope of the engineering analysis for this structure was buckling, bending, and deflection.

Finite Element Analysis (FEA) was done for the deflection of the support structure under the applied loads. See Appendix F for the software generated report. Figure 4 shows the result for the FEA. The areas in red indicate the higher levels of deflection and the blue areas indicate areas of low deflection. As seen on Table 1, the horizontal deflection of the top of the support was determined to be 0.368 inches. This value was used to calculate the vertical deflection of the top, outer edge of the guideway. That deflection was determined to be 0.167 inches. The displacement was measured when the system was fully assembled and found a difference from ten feet (the designed height) to be 0.25 inches. The displacement fell within an acceptable range and the deviation from the theoretical was attributed to the support and guideway not being square.

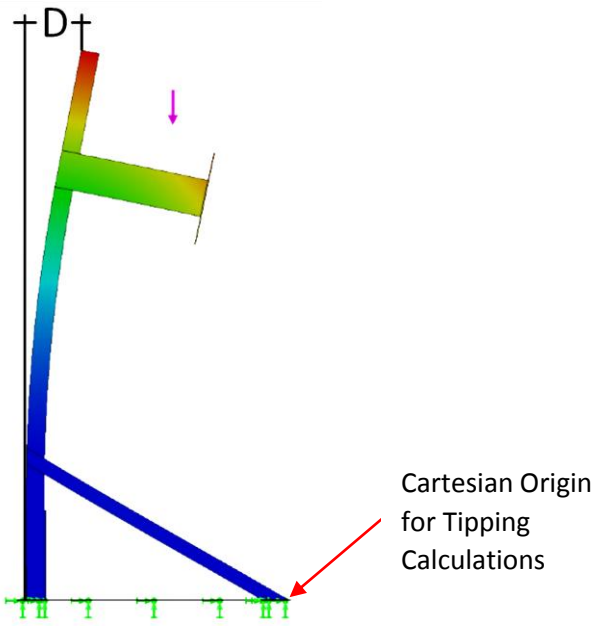


Figure 4: The results for the deflection analysis performed using FEA. The distance, D, was the horizontal deflection due to applied loads. It was found to have a deflection of 0.368 inches from the vertical.

Table 1: The results of multiple engineering analyses on the support structure. The weakest point for the support structure was the long angled braces that had a buckling safety factor of 2.9.

Column Buckling	Safety Factor = 6.1
Brace Buckling	Safety Factor = 2.9
Column Bending	Safety Factor = 7.1
Horizontal Deflection [D]	0.368 in
Vertical Deflection	0.166 in
Wind Speed to Tip	22.5 mph

Several analyses were completed without the use of a computer. The first analysis (Appendix A) was for the buckling of the main support column. The column was a 10 foot long, 4 inch square, ¼ inch thick tube. The slenderness and tensile yield strength of the steel indicated that the secant column formula (Appendix A) was to be used to calculate the critical buckling load. It was determined that the safety factor for buckling of the main support column was 6.1. This showed that the main support column was much stronger than necessary to resist loading on the member.

Following the buckling analysis for the main support column, an analysis was done for the bending of the support under the applied loads (Appendix B). The bending moment stress equation was used to determine the maximum moment that could be applied the column before yielding would occur. This value was compared to the moment that was applied from loading the structure and it was determined that the safety factor for bending of the main support column was 7.1.

The large, angled, support braces at the base of the support were determined to be the members in which failure would occur first, if failure were to occur. This was determined based on the location of the center of mass for the system and the length of the members. The secant column formula was used to determine the critical buckling load of the member. The analysis (**Error! Reference source not ound.**) was completed assuming fixed-fixed ends and an AISC recommend length of 65% of the actual length. The critical buckling load was compared to the load applied, and it was determined that the members had a safety factor of 2.9.

Another concern for the system was tipping due to wind load. The system was to be presented at an outdoor exhibit and the safety of the public was a primary concern. An analysis (Appendix D) was completed to determine the maximum wind speed that the support structure could withstand before tipping occurred. The critical wind speed for tipping was found to be 22.5 mph. This speed is greater than the average wind speed for the location that the system was to be presented in and was determined to be a safe design. An additional contingency plan was put into place which would allow the solar panels mounted on the top of the system to tilt and reduce the cross sectional area in the event that higher wind speeds were encountered.

In addition to determining the critical wind speed for tipping, the force required to tip the system was determined (Appendix D) and it was found that the wind force for tipping was less than the force required to buckle the brace members. Another concern was for the buckling of the braces after tipping commenced. An analysis was done that evaluated the system while tipped (Appendix E). As the system tipped, the center of mass would create a greater moment on the supports and could cause buckling. The point at which this moment would be greatest was after the system tipped 22°, which was when the center of mass was directly above the axis of rotation. After this point, gravity would pull the support the rest of the way to the ground. The force that the members experienced under the loading produced by the 22° tipping angle was found to be greater than the critical buckling load as determined previously, however other considerations led to the conclusion that the supports were acceptable.

The safety factor was 0.86, which was less than 1.0. The safety factor indicated a potential for material deformation. However, the critical buckling load was determined with the AISC recommended effective length, which is more conservative than the theoretical prediction. The theoretical prediction was a critical buckling load of 2300 lb., which resulted in a safety factor of 1.4. Additionally there were other factors contributing to the strength of the joint that were not taken into consideration with these calculations. The two major contributors not accounted for were the resistive force provided by the welds between the main support column and the baseplate, and the baseplate resistive force both in tension and bending. If those were calculated it was determined that the maximum force on each member would be less. Coupled with the use of the AISC recommended length, it was determined that the angled brace members were acceptable to use in the design.

It was then determined that under all wind load conditions that the system would always tip and the support members would not buckle from wind loads or wind tipping situations. It was noted that the steel support braces slid easily on the surfaces that they were located. Provided the base did not encounter an obstruction it was determined that the system would likely slide rather than tip.

Manufacturing of Support Structure

The support structures were constructed entirely of steel. The steel was provided at a reduced cost from PDM steel, one of SMSSV's sponsors. The San Jose State University Civil Engineering Technician, Patrick Joice, welded the parts of the support structure together. The parts of the support were fabricated from different stock pieces of steel. The angled brace members were cut from 3-inch wide 5/16-inch thick flat stock. There were four short braces and two long braces for each support, as shown in Figure 5. The base was made from 8-inch wide ¼-inch thick flat stock. The base was a "T" shape, which was fabricated by welding two short sections of flat stock to either side of the end of a long section of flat stock. The main support column was a 10-foot long, 4-inch square, ¼-inch thick tube. The tube was welded to the base, and the angled braces were then welded to the tub and to the base, to insure that the tube was square with the base. The guideway support arms and backplate where fabricated from 8 inch wide, ¼ inch thick flat stock. The backplate had 9/16-inch diameter holes drilled through it at a specific pattern for mounting the guideway with bolts. The backplate was welded to the support arms, and then the support assembly, shown in Figure 6, was welded to the main support column. The support assembly was positioned down the tube so the top of the tube would be level with

the top of the guideway. All welds that were done were solid seam welds to insure exceptional strength under loading.

Table 2: The bill of materials for the manufacturing of the support structure. See Appendix G for details.

Bill of Materials for Support Structures		
Component	**Quantity**	**Dimensions [in]**
Main Support Column	2	4 x 4 x ¼ thick tube, 120 long
Small Brace Member	8	3 x 5/16 thick flat stock, 25 long
Large Brace Member	4	3 x 5/16 thick flat stock, 62 long
Long Base Section	2	8 x ¼ thick flat stock, 60 long
Short Base Section	4	8 x ¼ thick flat stock, 16 long
Support Arm	4	8 x ¼ thick flat stock, 32 long
Back Plate	2	8 x ¼ thick flat stock, 20 long

Figure 5: The bottom of the support structures. The base pieces were made from 8-inch wide ¼-inch thick flat stock. The angled braces were made from 3-inch wide 5/16 inch thick flat stock. The main support column was made from 4-inch square ¼-inch thick tube. All welds were solid seam welds

Figure 6: The support assembly. The support arms and backplate were made from 8-inch wide, ¼-inch thick flat stock. A specific hole pattern was drilled in the backplate for mounting the support to the guideway

Manufacturing of Guideway Section

The guideway was constructed with the assistance of Keith McKenna, a Civil Engineering Graduate Studies Student. The materials that the guideway was constructed from was 2x4 planks, ¾ inch thick plywood, nails, construction adhesive, L-brackets, wood screws, bolts, a C-channel rail, and a 2x8 plank. The body of the guideway was constructed by fabricating 14 "J" shaped rib from 2x4 planks, as shown in Figure 7. The parts of the ribs were held together with nails, and then L-brackets were screwed to the joints to provide additional strength and deflection resistance. The Ribs were positioned ever 16 inches on center down the length of the guideway with an additional rib at each end, and an additional rib at each point of connection to the support structure. The ribs were attached to plywood sheets on all faces with construction adhesive and nails. This produced a "sandwich" of 2x4 ribs between two plywood sheets. Additional braces between each rib were added at the top portion of the guideway for increased torsional rigidity.

The 2x8 plank was machined by Amberwood, a SMSSV sponsor. The final dimensions were a 1x7 plank, 16 feet long with a 1/8 inch cap section machined out along the length of the top portion of the plank. The additional, machined portion was to allow the C-channel cap to sit on the rail and be flush with the

wooden portion. The steel rail was necessary for the bogie to ride on. If the rail was not in place there was a concern about deformation of the wood that could cause failure in operation.

Figure 7: The main guideway body. It was constructed by sandwiching “J” shaped ribs between plywood sheets. The ribs were strengthened by attaching L-brackets to the joints.

Figure 8: The rail mounted on the main guideway section. The rail is capped with a 0.120 inch thick custom C-channel and attached to the guideway with galvanized steel bolts

The rail, shown in Figure 8, was custom fabricated. A standard C-channel could not be used because the interior surface of C-channel has tapered sides. The tapered sides would require the plank to be machined down farther, which would result in a lack of material needed for support. Therefore, the solution was to purchase a 1x2 inch 0.120 inch thick ornamental steel tube, 8 feet long. The tube was ripped down the length to produce two 8 foot, 1x1 inch by 0.120 inch thick "C-channel pieces. The custom C-channels were glued and fastened to the 1x7 wooden rail with construction adhesive and #8 machine bolts. The rail was then attached to the main guideway body with ¼ inch diameter galvanized steel bolts. At the locations of the support structure connections ½ inch diameter bolts were used, and the bolts extended completely through the rail, guideway, and support structure backplate. This was done to provide additional strength to the rail, as it was the part of the guideway that supported the load form the bogie and the cabin.

Table 3: The bill of materials for the guideway. See Appendix G: Guideway Part Drawings for details.

Bill of Materials for the Guideway		
Components	**Quantity**	**Dimension [in]**
Rib Pieces	14	2 x 4 planks
Face Pieces	N/A	¾ plywood
Guiderail	1	1 x 7 plank
Rail Cap	1	1 x 1 x 0.12 thick steel C-channel
Bolt	16	½ diameter 6 long
Bolt	4	½ diameter 8 long
Bolt	10	¼ diameter 8 long
Nut	20	½ diameter
Nut	10	¼ diameter

Washer	40	½ diameter
Washer	20	¼ diameter
Nails	N/A	varied
Construction Adhesive	8 tubes	N/A
L-Bracket	56	1/8 thick 4 inch sides

Assembly of Guideway/Support System

The guideway system was assembled by first wrapping the main guideway body in hoisting straps. A chain was wrapped between the two straps, and then a forklift lifted the guideway from the chains. The Guideway was positioned against the support structure. Holes were drilled through the guideway using the hole pattern on the backplate as a pattern. Bolts were then plunged through the holes, and secured. Once all bolts were connecting the guideway and support structure together the forklift released the guideway and the supports solely held the guideway. The chain and hoisting strap was removed. A hole was drilled on the top of the guideway and an eye-bolt was fastened at the center of mass, that would allow for future relocation and assembly by attaching a chain to the eye-bolt and wrapping it around a forklift. Plywood caps were constructed and screwed to the ends of the guideway. The caps prevented the bogies from rolling out of the guideway and onto the ground. The caps were only screwed on to allow for easy removal. Easy removal was necessary so the bogies could be slid out of the guideway when relocation was desired. The guideway was painted with flat white exterior paint, and the back of the guideway received a Spartan Superway logo stencil. The bogie were slide into the guideway and the cabin was fastened to the bogies. The solar panels were mounted on the top of the support structures.

Figure 9: The assembled guideway system, set up for display at the Maker Faire in May 2014.

Next Steps

Theoretical Model

As mentioned in the system model section, there are a number of areas which require further work. The guideway team was unable to provide a number of calculations due to insufficient data from other teams, as well as time constraints. A major objective which must be completed before work can progress is the analysis of stresses on the rail. Once the combined weight of the bogies, cabin, and passengers is known, FEA analysis of the design can proceed. There are a number of parameters which can be optimized. The design of the ribs can be changed to provide effective support against torsion and bending. At this point, it is unknown which of these is a larger problem. The height of the rail may be adjusted to ensure that it does not bend under the combined load.

The switching section is integrated heavily with the bogie team. At this point in time, the bogie team has created a rough design which interfaces with their wheel setup. To create a working design, collaboration with their team is required. This is to ensure that a number of parameters, including wheel height and position during switching, are taken into account.

To create a safe and effective design for the support system, collaboration with civil engineers is required. There are too many unknown variables which come into play when working with concrete and ground soil. The basic design is provided, but it is assumed at this point that a number of changes will need to be made. Also, the visual impact of these supports must be taken into account because they will be placed frequently in urban centers.

Maker Faire Model

The guideway was constructed while considering additional iterations, or additions to the system. What was determined to be a significant improvement to the system would be an addition of a switching section of track. The switching section of track can be built with the same dimensions and height as the straight section. Similar supports could be constructed, although some engineering would need to be completed in order to insure that the angled braces could support any changes in loading. The switching section could be attached to the straight section in a similar fashion as the support structure is attached to the straight section. The ribs at the ends of each section, could be butted against one another and the support could be positioned at the joint. An additional steel plate can be added to the opposite side of the backplate and it would act as a multiple bolt washer for the guideway. The switching junction is a vital part, and arguably to most important part of the guideway. It is important to showcase the functionality of this part, so the concept can be "sold" to municipalities and politicians, who will be fundamental in implementing the ATN in their cityscapes.

Bogie Team

On the Superway PRT vehicle, the bogie is the component which navigates the guideway and supports the cabin and its passengers. It is responsible for propulsion, braking, structural support, sway control, track switching, and interfacing with the power conduits located on the track. The bogie is located above the cabin.

Figure 1: Superway Bogie Prototype V1.0 in Guideway at Maker Faire (Photo Credit: Dr. Buff Furman)

The bogie team is responsible for all structure and components between the guideway and the roof of the cabin. The following interfaces exist between the bogie subsystems and the other components of the Superway design:

- Wheels/rollers in contact with the guideway
- Mounting points on the roof of the cabin
- Contact points to transmit power to the bogie from the 3rd rail
- Communication between control system and bogie controller (method TBD)

Requirements and Specifications

The guidelines, criteria, and constraints of the Superway Bogie were defined during the fall 2013 semester. All system model components have been or will need to be selected and designed to meet

them. For a full breakdown of the Design Requirements and Specifications, see appendices Appendix I: Bogie Design Requirements and Appendix J: Bogie Design Specifications.

Design Challenges

There are a number of design challenges with regards to the Spartan Superway Bogie. The critical design challenge which was resolved this semester was defining wheel placement to enable bogie-selected guideway switching for a suspended podcar design.

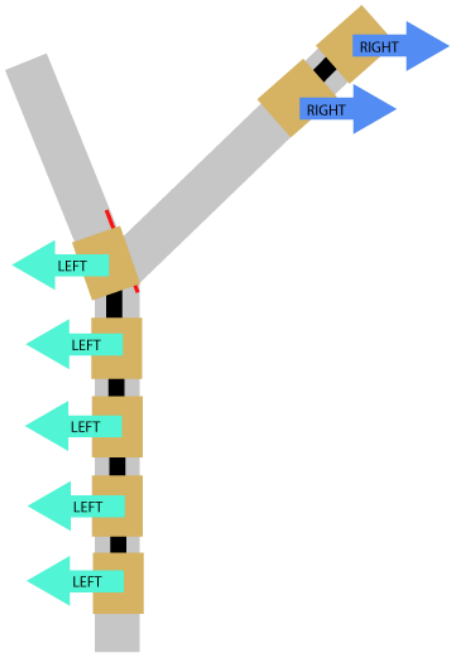


Figure 2a: Train Switch

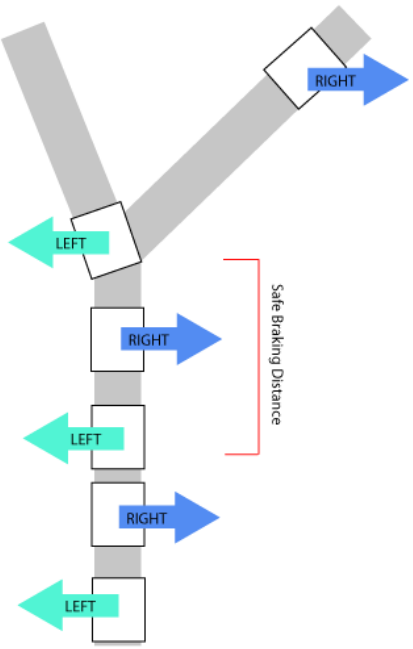


Figure 2b: ATN Switch

In a conventional train or light rail, it is acceptable to have moving parts in the track in order to direct trains in one direction or another. This is because each car in the train is heading in the same direction as the one in front of it, as illustrated in Figure 2a. Additionally, the lead time between each train is typically large. In the event of a switch malfunction, the train is nowhere near the switch, and can be safely stopped before it reaches the switch, avoiding a derailment.

By contrast, an ATN guideway must be completely static at the guideway switch. Hypothetically, each car in an ATN may wish to travel the opposite direction of the car in front of it, as illustrated in Figure 2b. In order to achieve this with moving parts in the guideway, the parts must be able to move very rapidly, as the lead time between cars will be very small. Additionally, the cars will not be able to stop before entering the switch if there is a malfunction. In the case of a suspended vehicle, the subsequent derailment could equate to the car falling off the guideway, causing injury, death, and property damage for the occupants and everyone in their path.

It is possible to increase the gap between cars, thereby increasing lead time and allowing cars to be beyond a safe braking distance of the switch, which could signal a malfunction. Doing so, however, would enlarge the space between cars significantly, resulting in inefficient use of guideway space. Since infrastructure efficiency is the hallmark of the ATN, this approach is undesirable. It is a commonly accepted requirement of an ATN that switching must be handled by the vehicle, not the guideway.

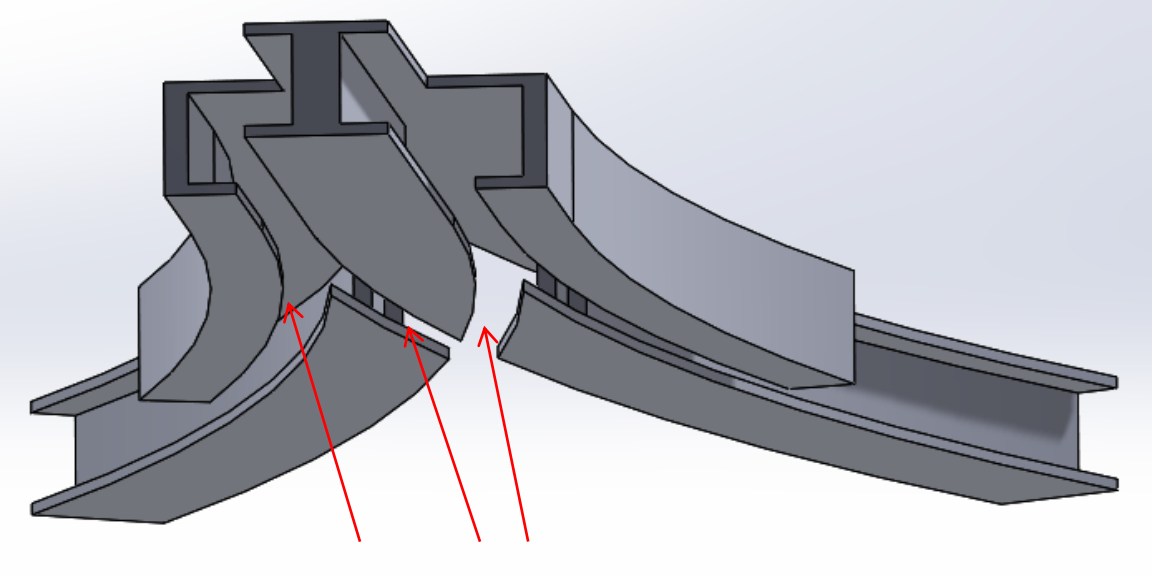


Figure 3: Guideway Groove Problem

Designing for the static-guideway condition with a suspended cabin is a unique challenge. The guideway itself is a barrier between the bogie and the cabin, so guideway designs usually have grooves in them. Figure 3 is an image of a previous design iteration wherein the bogie traveled along an I-Beam shaped guideway. As the red arrows point out, at the switch, the bogie must travel over a groove in their rolling surface. This problem is common to most designs. Allowing the support wheels to roll over such a groove causes wear on the vehicle and discomfort for the passengers.

The particular I-Beam design shown above was shown to not suffer from this jolt in the ideal case, but the realities of deflection on cantilevered pieces of the guideway made the design unviable.

Theoretical System Model

The majority of progress on the system model achieved this semester was in designating the wheel placements during normal rolling and switching operations, as well as the basic structure of the bogie. Other bogie subsystems have been updated to reflect changes to the overall system design. Currently, there is no 3D assembly of the complete system model with all of its subsystem components, as our efforts were focused on building the V1.0 Prototype.

Chassis

The structure of the bogie is comprised of three pieces: Two rolling sections or “half-bogies”, and one connecting segment or “H-Bar”. The Half-Bogies are not constrained for rotation on the horizontal axis

perpendicular to the guideway direction if they are not attached to the H-bar. Both half-bogies can rotate along a vertical axis with respect to the H-bar to enable travel through turns.

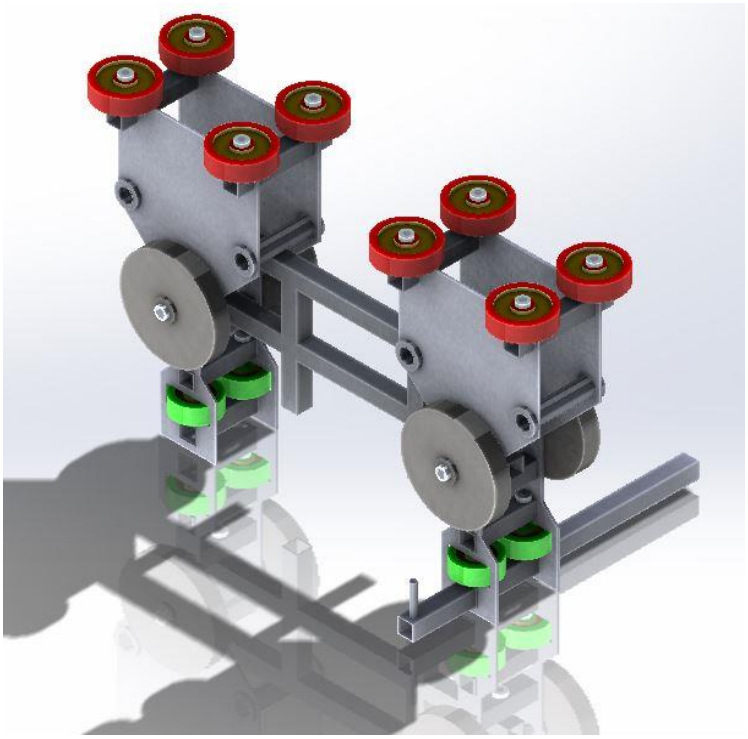


Figure 4: Solidworks rendering of Superway Bogie Prototype V1.0

The H-Bar is what attaches to the roof of the cabin and bears the weight of the cabin and its occupants. It is also a potential mounting point for other components which may be necessary in a complete design.

Wheels

The wheels on each half-bogie serve particular purposes. As colored in Figure 4 above, the red wheels are polyurethane and serve to keep the bogie vertical in the guideway and to rotate it as it goes through a curve.

The grey wheels are steel, and support the weight of the pod car. Only one of the two wheels on each half-bogie are needed to support the weight of the podcar. During normal operation, one support wheel is not in contact with the guideway.

The green wheels serve to keep the support wheels on the guideway rail. Though not shown in Figure 4, another green wheel is present on the bare axle in Figure 4, which serves to sandwich the support rail of the guideway and keep the support wheel in position. This also functions as part of the switching mechanism which is discussed below.

Suspension has been deemed unnecessary for the following reasons. Firstly unlike a road surface, we have precise control over the rolling surface for the bogie, and steel is exceptionally smooth. Secondly, being a suspended design, the walls of the cabin and the weight-bearing members over distance act like a spring-damper system, alleviating small amounts of noise and vibration. Thirdly, most other suspended systems have no suspension system to speak of. If suspension was deemed important for

passenger comfort, it could be placed at the half-bogie H-bar interface or between the H-Bar and the cabin.

Iterative Design Process

As discussed above, arriving at this chassis wheel placement design is the result of multiple iterations of guideway and bogie designs.

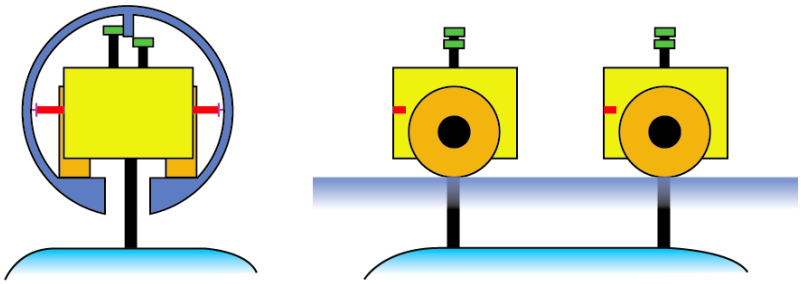


Figure 5a: Square Split-Bogie Design

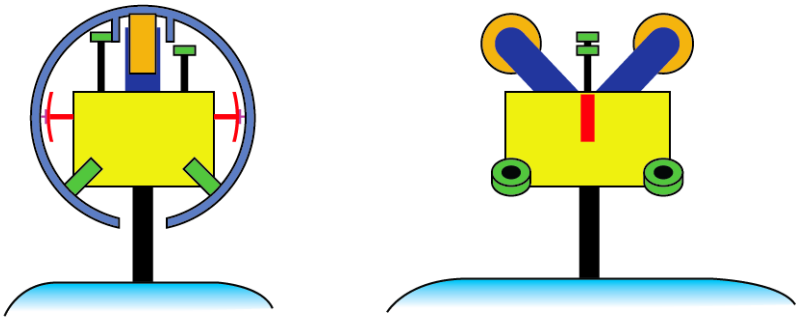


Figure 5b: Tubular Bogie Design

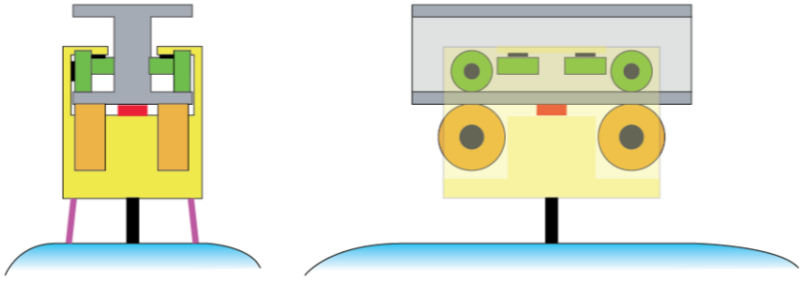


Figure 5c: I-Beam Design

Initial designs were based on the 1/12 scale model built by last year's team, wherein the guideway was a square tube with a groove cut in the bottom for cabin support. It was decided early on to change to a circular cross section to improve aesthetics and wind loading properties. Avoidance of the "groove

bump" phenomenon was attributed to the steering system, which provides a moment arm resisting the wheel falling into the groove, though flexure of the bogie could still make “groove bump” present a problem.

The idea to use half-bogies originated from the split-bogie design shown in figure 5a. This minimizes the widening of the guideway necessary in a corner

The tubular design shown in figure 5b sought to allow rotation of the bogie in the tube rather than hinge the cabin from the bogie to allow banking during cornering.

Both tubular guideway designs were ultimately rejected due to stiffness concerns from the guideway team, wherein the I-Beam design was developed. Details of the I-Beam design can be found in last semester’s report. The design was shown to travel through a switch with no “groove bump” in the ideal case of zero guideway deflection.

Collaboration with Beamways

The Bogie Team has had the distinct pleasure of working with Bengt Gustafsson of Beamways AB from Sweden. In January 2014, Bengt presented his design for a suspended ATN bogie, shown in Figure 6a below. Initial concerns with the complexity of the bogie and the sheer number of wheels created skepticism. When analyzing the Beamways Guideway Switch, shown in Figure 6b, we presented Bengt with the idea of consolidating the support rail and the steering rail. He was satisfied with us pursuing that design change, and agreed on neutral ground for intellectual property.

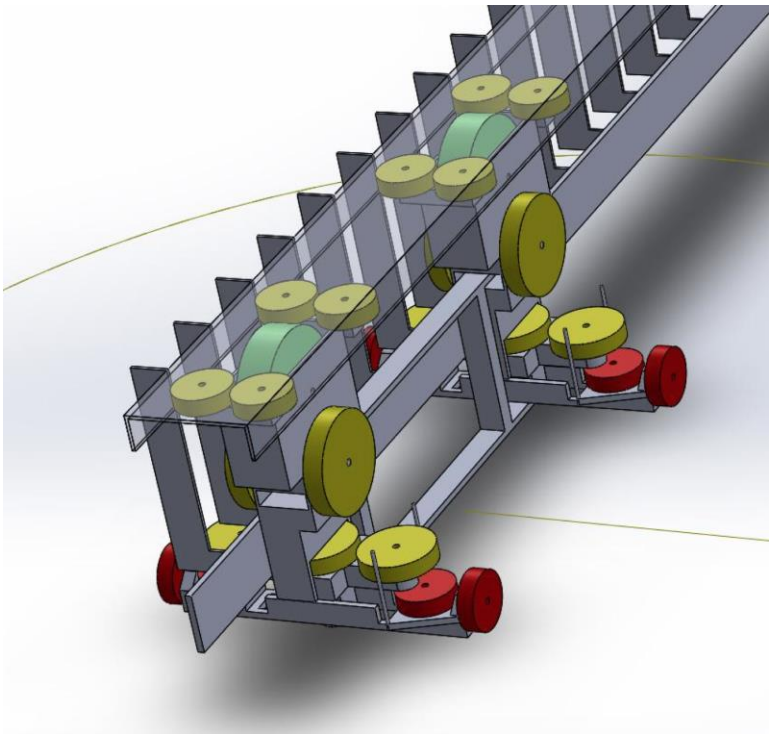


Figure 6a: Beamways Bogie in Guideway

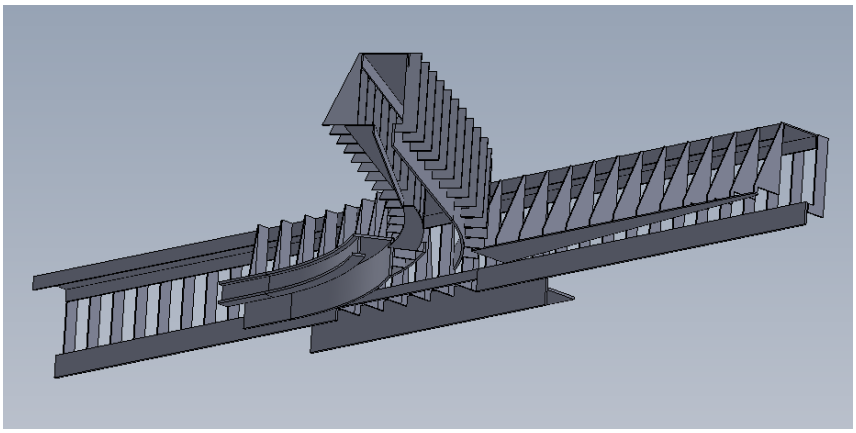


Figure 6b: Beamways Guideway Switch

Another notable design difference is that the switching mechanism on the Beamways design operates on a slider rather than a rocker. This enables guideway override of the steering arm if the mechanism to move it is broken, eliminating the need for a bogie with a broken steering mechanism to come to a stop.

It is also important to note that all wheels on the Bogie Prototype V1.0 are property of Beamways AB and are in the custody of the Spartan Superway project with the understanding that our development of the prototype is valuable to Beamways. Should Beamways deem the Superway Prototype too divergent from their design intentions, they reserve the right to recall their property.

Propulsion

Propulsion for the podcar is achieved by electric motors in the half-bogie bodies. Driving force will be applied to the ceiling of the guideway, and power for the motors will come from an electric power conduit in the guideway.

Motors

Figure 7: A Wheel Hub Motor (Image Credit: www.proteanelectric.com)

The motors used for the propulsion system are electric wheel hub motors with self-contained friction brakes. These motors are capable of regenerative braking as well, and may need external liquid cooling.

The wheel pokes up through the top of the half-bogies, between the upper wheels. This is identical to the Beamways Design.

Traction System

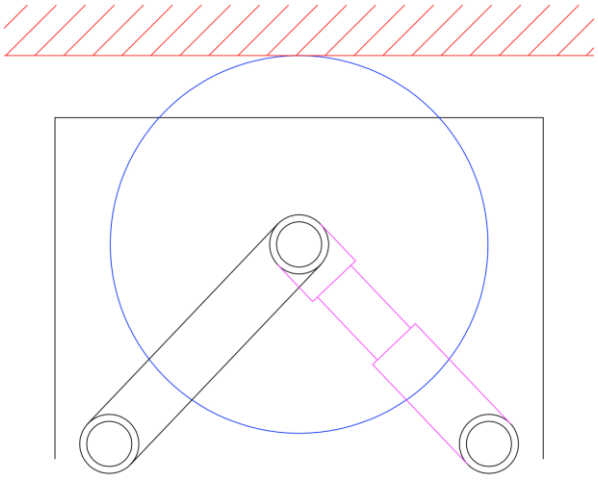


Figure 8: Traction System

In order to force the drive wheel into the ceiling of the guideway for traction, a mechanism which can move the drive wheel is installed in the bogie. This system can increase traction in order to enable climbing of steep grades and can reduce rolling friction by dropping the drive wheel down into the body of the bogie. It should be noted that increasing force from the traction system increases the reaction forces all over the bogie where it contacts the guideway.

A potential configuration for the traction system is shown in Figure 8. The linear actuator (pink) changes the angle of the axle arm (black) pressing the drive wheel (blue) into the guideway (red).

Undetermined Components

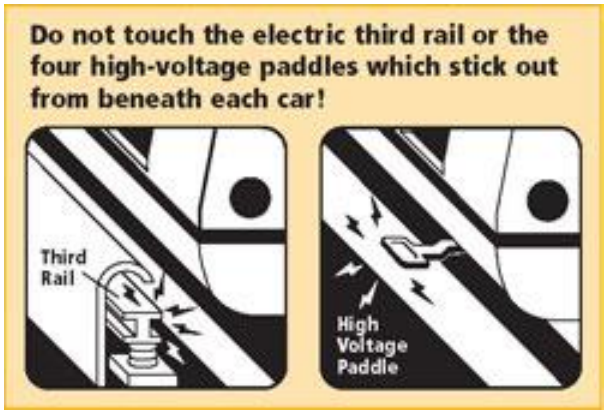


Figure 9: High Voltage paddles on Bart (Image Credit: www.bart.gov)

The remainder of the propulsion system, namely the cooling system, power supply, and power interface have yet to be designed, and are dependent of the requirements of the specific wheel hub motor. If the power supply can take single phase power, then a single powered rail can be attached to the guideway wall, with a high-voltage paddle making contact with it such as on Bart. Ground can be transmitted through the support wheel into the guideway.

Switching Mechanism

The method of guideway switching for the Superway ATN is both elegant and effective. Figure 10 shows the guideway switch.

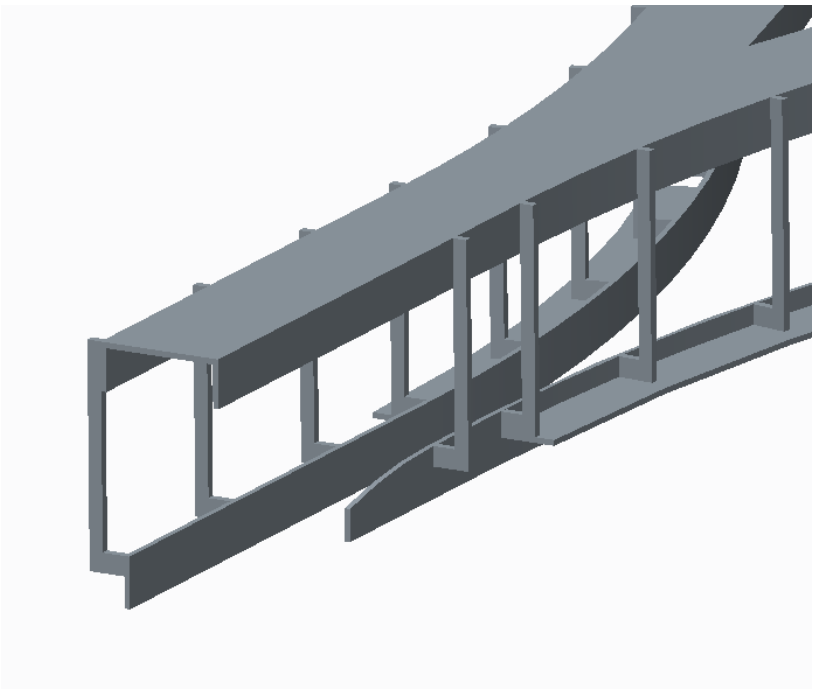


Figure 10a: Guideway Switch Entry

When approaching a guideway switch, the bogie rolls onto a second support rail on the other side of the bogie. The two support rails sandwich the bogie between them, making contact with the green wheels at the bottom of the bogie.

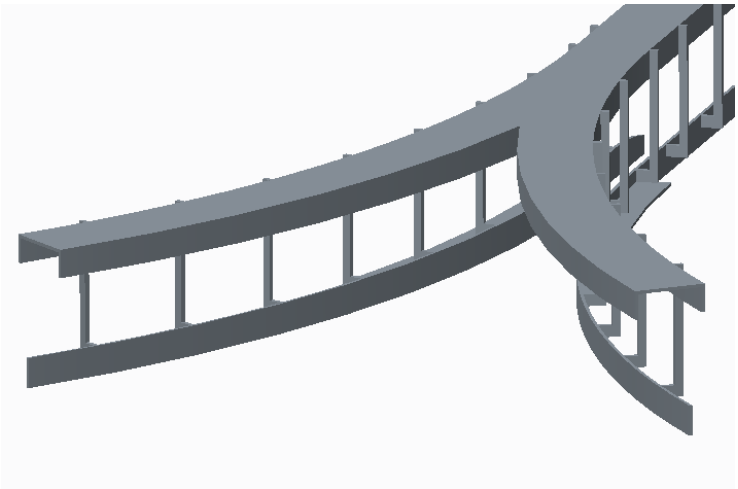


Figure 10b: Guideway Switch Exit

With the bogie securely supported and constrained, the wheel which gripped the outside of the support rail (herein called the steering wheel) can be safely moved. If it remains in place, it the bogie will follow the support rail it is grabbing. If it moves and grabs the other support rail, the bogie will follow the other support rail instead.

Movement of the steering wheel is achieved with a rocker arm, as shown in in Figure 11. The actuation method of the steering arm is not yet determined. During normal operation, the steering wheel must remain in contact with the support rail to ensure safe operation, and it cannot require power to do so.

Latches (shown in pink) will hold the steering arm in the position it is in. They will be mechanically released when the bogie is in the section of Guideway where both support rails are present. This can be achieved mechanically or electronically.

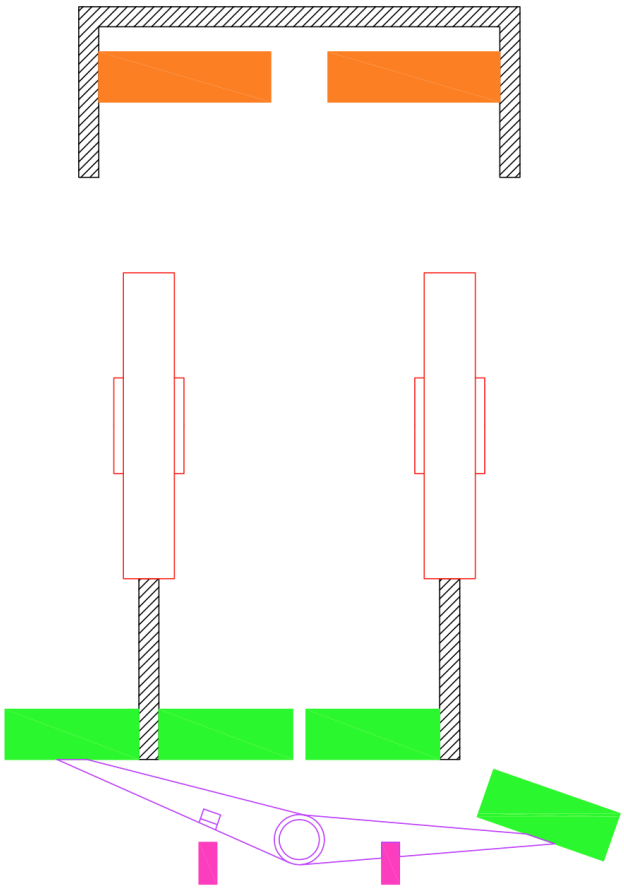


Figure 11: Bogie Cross-Section with Steering Arm

For a brief time, the wheels at the top of the bogie must be unconstrained during a guideway switch. This is a problem because those wheels provide the moment which counteracts the moment caused by the offset of the center of mass from the support rail. To keep the bogie in the track, a flange is added for a few meters (visible in Figure 10a). An additional wheel is added to each side of the steering arm which engages that flange and prevents the bogie from falling out of the guideway. Those additional wheels are not shown here.

Sway Control

Another undetermined element is the sway control system. In order to provide a comfortable ride for the passengers while navigating guideway corners at speed, the cabin must be allowed to tilt, banking during cornering. While that could be accomplished passively with a hinge, the cabin would rock back and forth during passenger entrance and egress, and wind would make an uncomfortable ride. Therefore, an active system for controlling the swaying motion of the cabin is needed.

There is the potential to allow 2-axis sway to occur, so that the pod car can accelerate faster than would otherwise be comfortable, and so the pod car could climb steep inclines without tilting at an alarming angle to the occupants.

Emergency Systems

In the case of an emergency, backup systems are required in an ATN pod car to ensure that the network is a safe mode of transportation.

Emergency Brake

In order to stop the Bogie if the propulsion system fails to, a simple friction brake functions as an emergency brake. This is especially important considering the propulsion system's braking ability is dependent on both the wheel hub motor's friction brake and the traction system's actuation.

Figure 12: Brake Pads (Image Credit: www.zigwheels.com)

Brake pads, as shown above, can be pressed into the guideway, possibly clamping the support rail. Actuation of this system must be able to be triggered remotely and by passengers in the cabin.

Tow Bogie

In order to move non-functional bogies if they break down, a bogie with no cabin can be employed. In order for the bogie to safely control the broken pod car, it must be able to interface with a rigid part of the frame. If possible, it should also be able to actuate the steering mechanism on the broken bogie.

Maker Faire Model

Figure 13: Superway Bogie Protoype V1.0 shortly after being constructed

In order to generate public interest in our project, as well as to show off our engineering prowess, the Superway Bogie Prototype V1.0 was built for display at Maker Faire. It is a full-scale prototype, utilizing the same wheels in the same configuration as are intended for the final design.

Our design of the frame was limited to our manufacturing ability. The frame was built out of plate steel, plasma cut by a CNC machine, and various sizes of tubes and axles. Analysis suggests that the bogie can handle around 1000 lb of weight, which is not enough for the full cabin, but should be enough to carry one to two people for demonstration purposes.

In leu of a steering rocker arm, the steering mechanism is simulated by a 2”x2” square pipe with an axle and wheel on it which fits into a receiver tube located at the bottom of the bogie. To simulate the switching of the steering arm, the pipes can be removed and inserted from the other side of the bogie. A ¾” axle holds the pipe in place. Because of the low clearance for the steering wheel on the wooden guideway model, a collar clamp couldn’t fit above it, so inverting the steering arm causes the spacers and wheel to fall off.

Though a propulsion system was not used, 2” round pipe was installed with collar clamps to be a mounting point for such a system in the future. The receiver tube for the steering pipe can also be a mounting point for a true rocker mechanism for V1.1.

Finite Element Analysis

Creo Parametric 2.0 was used to render and perform a static finite element analysis (FEA) on the Maker Faire exhibition model of the bogie. Two versions of the full bogie model were created: one specific for static analysis, and another for detailed assembly. In the analysis-specific version (Figure 14), each bogie

unit was made as a single part and joined together in an assembly with the center h-beam, in order to simplify the number of elements and reduce analysis time.

The detailed version of the model recognizes all of the separate components needed for assembly, allowing individual editing of components and detailed exploded views as shown in Figure 15. A static analysis of this model could not successfully be performed due to continuous errors in Creo's solver. The center h-beams rendered in the models do not depict the actual design in the Maker Faire model. This is due to the later timing of design decisions and construction. A simplified point/surface of loading was made in this model.

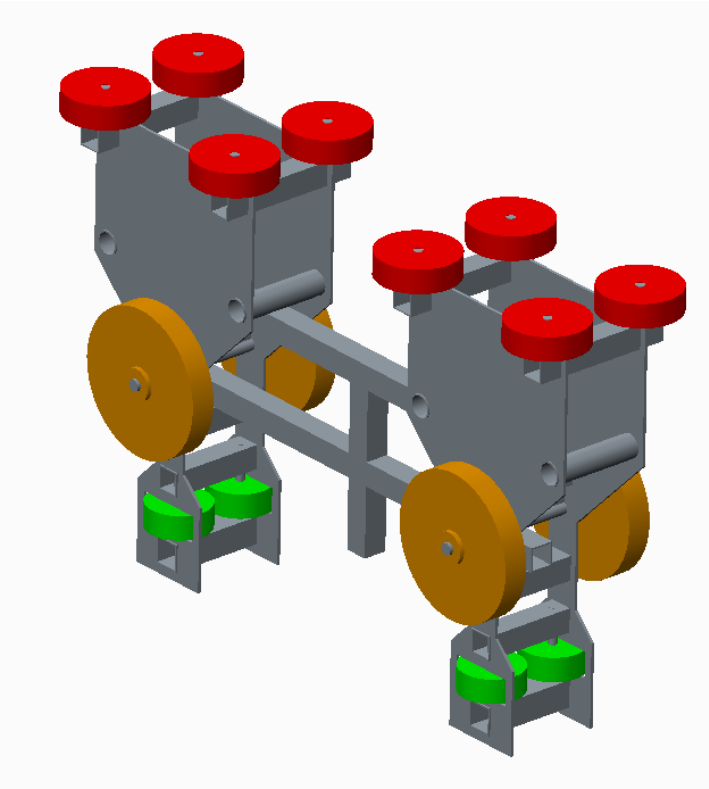


Figure 14: Solid FEA Model (Simplification)

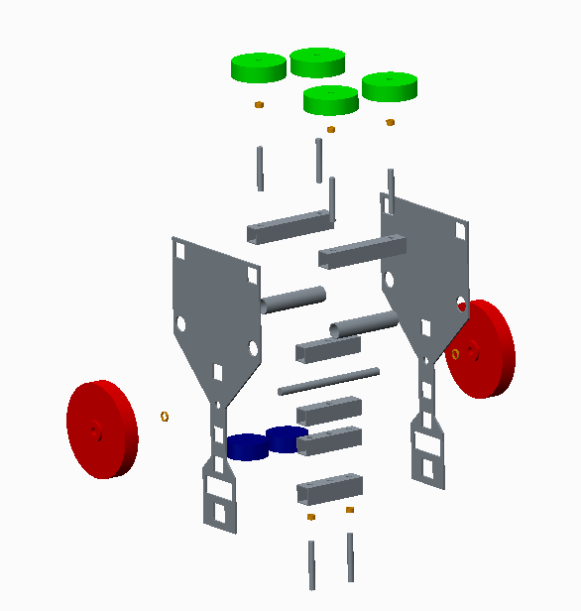


Figure 15: Exploded view of a half-bogie (Full Assembly)

Static Analysis Preparation

The loaded and constrained model is shown in Figure 16. The vertical support wheels (gold) and upper guide wheels (red) are constrained in the same manner that the bogie would interface with the guideway under load. The upper wheels required constraints in all directions in order to meet Creo's requisite of sufficiently constrained entities. A 500lbf load was applied to the bottom end of the center h-beam. While that weight figure was much larger than the actual mock-up cabin, there was concern for unexpected loads – such as additional loading or leaning on the cabin.

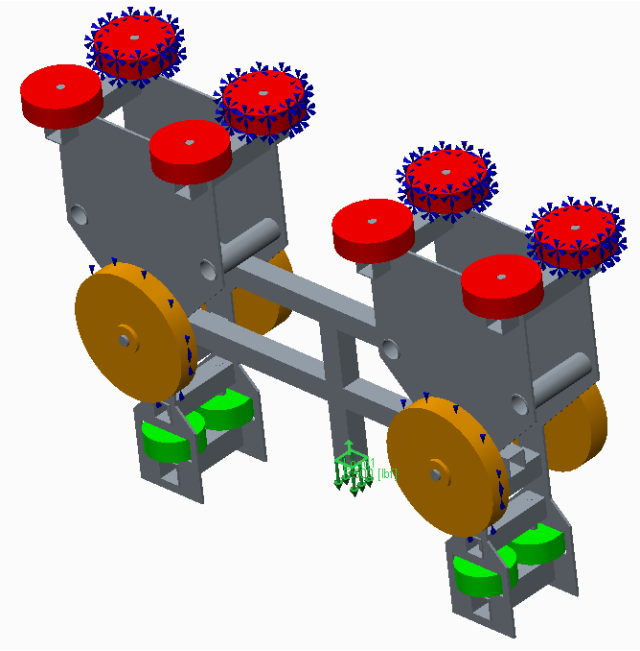


Figure 16: Bogie Model with loads and constraints

The goal of this analysis was to determine von Mises stress to determine safety factor, and deflection to identify any considerable deformation that could cause the bogie to slip out of the guideway. Wind loads were not addressed at the time of analysis because of the late arrival of the mock up cabin to determine dimensions for rendering.

The analysis was set as a Multi-Pass Static Analysis with a polynomial order of 9 and convergence set to 10%. With the expectation to materialize for Maker Faire, no shell idealizations were utilized as they would further simplify the analysis, but at the cost of accuracy; therefore the model was treated entirely as 3D solids. All parts were assigned as steel per Creo's Materials Assignment menu. The polyurethane layers on the wheels were mainly intended for noise and interface wear reduction, as opposed to attributing to the strength of the wheels which are dominantly steel.

Using the AutoGEM feature in Creo, element sizing was further refined by decreasing maximum edge turn from 95 to 30 degrees to better recognize round features of the model. The AutoGEM graphic and summary is shown in Figure 17. with the statistical data regarding the types and amounts of elements subject to analysis.

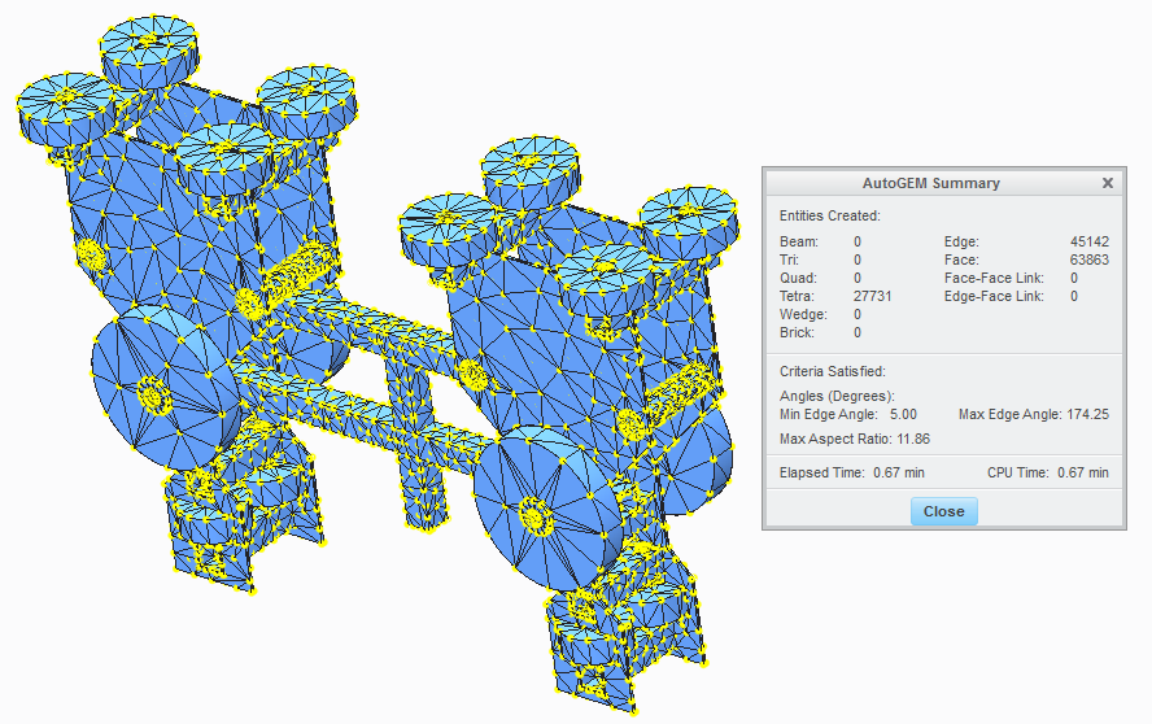


Figure 17: Bogie AutoGEM graphic and summary

Results

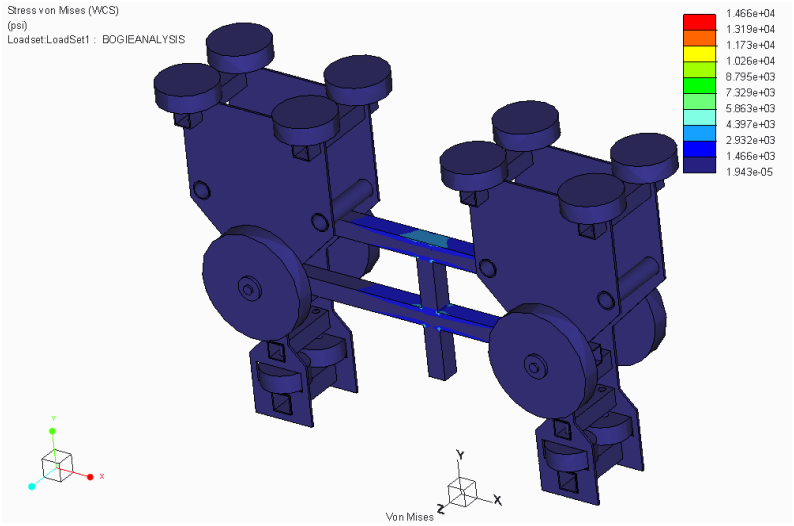


Figure 18: Bogie von-Mises stress fringe plot

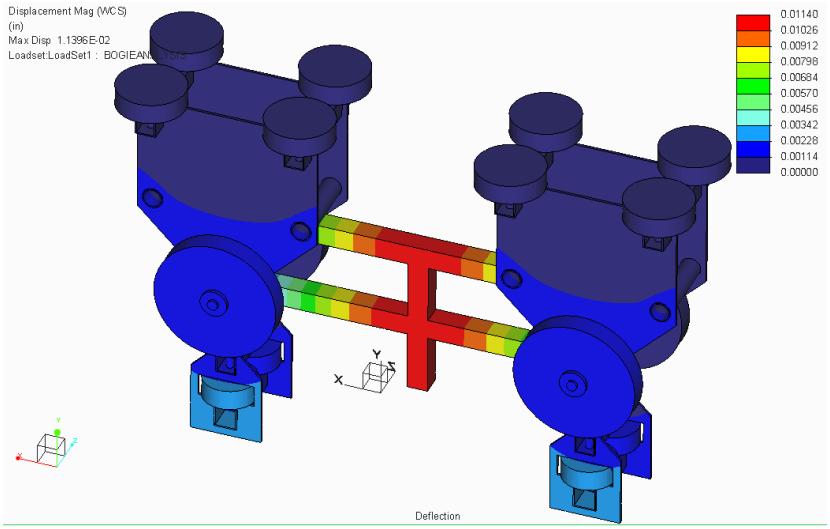


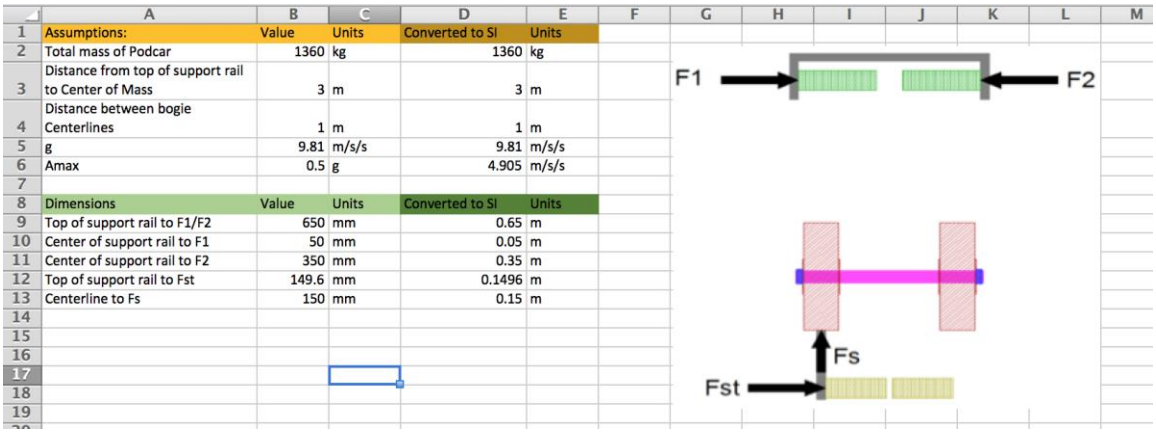
Figure 19: Bogie Deflection fringe plot

Figure 18 and figure 19 show the von Mises and deflection fringe plots respectively. The static analysis required 7 passes to reach its maximum polynomial order of 9. As reported in the .RPT file (see Appendix K: Bogie Static Analysis), the maximum von Mises stress was 14541.46psi, and concentrated in the middle of the center h-beam, particularly at the junctions. However, the convergence of this value was only 13% as opposed to the 10% as set in the preliminary analysis menu, which could lead to the possibility of a higher von Mises stress value. This value results in a safety factor of 2.88, as it occurs on the A500 Grade B steel tubing which has a yield strength of 42,000psi.

When accounting the weight of the actual mock-up pod to be less than the tested 500lbf, the unsatisfactory convergence value did not guarantee a higher risk of structural failure. The maximum deflection reported in the .RPT file was 0.01139 inches. As Figure 19 indicates, the maximum deflection takes place again in the center h-beam, and increased levels are also found at the support wheels and bottom end of each bogie unit. Given the small magnitude of deflection, in addition to the high safety factor, this analysis suggested that the Maker Faire model would be able to bear static vertical loads greater than that of the mock-up cabin.

Excel Force Model

Utilizing static force and moment equations from CE99, an excel model for forces acting on the contact surfaces (wheels) of the bogie was generated for various scenarios. This was used to determine what sorts of forces the various wheels would need to cope with.



	A	B	C	D	E
1	Assumptions:	Value	Units	Converted to SI	Units
2	Total mass of Podcar	1360	kg	1360	kg
3	Distance from top of support rail to Center of Mass	3	m	3	m
4	Distance between bogie Centerlines	1	m	1	m
5	g	9.81	m/s/s	9.81	m/s/s
6	Amax	0.5	g	4.905	m/s/s
7					
8	Dimensions	Value	Units	Converted to SI	Units
9	Top of support rail to F1/F2	650	mm	0.65	m
10	Center of support rail to F1	50	mm	0.05	m
11	Center of support rail to F2	350	mm	0.35	m
12	Top of support rail to Fst	149.6	mm	0.1496	m
13	Centerline to Fs	150	mm	0.15	m
14					
15					
16					
17					
18					
19					

Figure 20: Force Model Assumptions

One page of the workbook (shown in Figure 20) is dedicated to assumptions made about the dimensions of the bogie and the guideway. Information such as the mass of the podcar and the distance from the center of mass to the guideway were assumed, and can be changed as more accurate information becomes available

Results of the model (shown in Figure 21) are hardly conclusive, given the amount of guesswork involved, but were a good starting point to know how to load a completed bogie frame in FEA, and for selecting wheels.

	A	B	C	D	E	F	G	H	I	J
1	**Case: Static Hanging**	Value	Units							
2	Left Guide Wheels	0	N			Fs	13341.6	N		
3	Right Guide Wheels	625.7004	N			Ma(mg)	2001.24	Nm		
4	Center Wheels	1251.401	N			F1	0	N		
5	Support Wheels	6670.8	N			F2	2502.801	N		
6	Steering Wheels	0	N			Fst	2502.801	N		
7										
8										
9	**Case: Left Turn Amax**	Value	Units							
10	Left Guide Wheels	2467.186	N			Fs	13341.6	N		
11	Right Guide Wheels	0	N			Ma(mg)	2001.24	Nm		
12	Center Wheels	0	N			Fa	6670.8	N		
13	Support Wheels	6670.8	N			Ma(Fa)	4080	Nm		
14	Steering Wheels	4934.371	N			Fst	-9868.74	N		
15						F1	9868.743	N		
16						F2	0	N		
17										
18	**Case: Right Turn Amax**	Value	Units							
19	Left Guide Wheels	0	N			Fs	13341.6	N		
20	Right Guide Wheels	4006.676	N			Ma(mg)	2001.24	Nm		
21	Center Wheels	8013.352	N			Fa	6670.8	N		
22	Support Wheels	6670.8	N			Ma(Fa)	4080	Nm		
23	Steering Wheels	0	N			Fst	16026.7	N		
24						F1	0	N		
25						F2	16026.7	N		
26										
27	**Case: Traction Modes**	0.5x (N)	1x (N)	2x (N)			0.5	1	2	
28	Left Guide Wheels	0	0	0		Fs	20012.4	26683.2	40024.8	N
29	Right Guide Wheels	938.5505	1251.401	1877.101		Ma(mg)	2001.24	2001.24	2001.24	Nm
30	Center Wheels	1877.101	2502.801	3754.202		Ft	6670.8	13341.6	26683.2	N
31	Support Wheels	10006.2	13341.6	20012.4		Ma(Ft)	1000.62	2001.24	4002.48	Nm
32	Steering Wheels	0	0	0		F1	0	0	0	N
33						F2	3754.202	5005.603	7508.404	N
34						Fst	3754.202	5005.603	7508.404	N

Figure 21: Inertial Force Excel Model Results

Construction

The construction phase of the bogie began with quarter inch thick steel metal sheets from PDM. PDM laser cut the sheets of metal to the desired shape. All of the square tubing and solid circular bars were purchased through PDM as well due to convenience and cost. Receiver tube was purchased as well from an outside manufacturer to provide the necessary support and functionality for the steering arm.

Once all of the material was purchased, the square tubing and solid circular bars were cut and drilled to specifications. Drawings are available in Appendix L: Engineering Drawings. Once all the necessary materials were cut to size, the fitting stage began and all of the pieces were fit together to ensure that everything was the proper length and that there was no interference.

The bogies were built one at a time by standing up two plates about 6 inches apart from one another and sliding the proper tube or solid bar through the designated slot that was laser cut by PDM. This allowed the bogie to stand freely and shape. The bogie was then tipped over onto its long side from the standing position for aligning and clamping. Three solid bars were cut on the lathe to exact lengths to ensure that a very tight tolerance was achieved. These solid bars were used as spacers to put in between the two bogie plates so that the distance from one plate to another all throughout the bogie was equal. Three U-shape clamps were used to clamp the plates of the bogie to the proper orientation

to ensure that the plates were as close to parallel as possible. Seen below in Figure 22 are images of how the clamps and spacers were incorporated into assembling and preparing the bogie for welding.

Figure 22a: Clamps and Spacers used to keep plates parallel

Figure 22b: Tbogie clamped together on the welding table

A Miller MIG electric welding machine was used to provide the necessary welds to withstand the calculated loads. Each piece of square tubing was adjusted into the exact place and then tack welded. Once all of the square tubing was tack welded, the alignment of each of the tubes was checked to make sure that they were all in the desired location still. Everything was still in the right position which meant that permanent welds could be completed. Each square tube was welded all the way around to ensure maximum bonding and strength. Figure 23a is what the welds looks like around one of the square tubes.

Figure 23a: A complete weld around one of the square tubes

The same procedure as stated above was done for the circular solid bars. Each bar was tack welded into place and the checked for alignment. After alignment was verified, a full complete weld was done.

Figure 23b: Full weld done around solid axle bar

After all the welds were completed, a wire brush and grinder was used to clean and polish the welds. The same process was done for the second bogie with alignment of the tubes and rods, tack welding, and complete welding to ensure that both bogies were identical once completed. After both bogies were welded in all the necessary spots, the h-bar was welded on the ground to ensure that all surfaces were perpendicular with one another. The final step to building the bogie was to install all of the bearings and wheels. All of the wheels fit into place as planned due to good tolerances. Below in Figure 24 is an image of the bogie fully assembled.

Figure 24: Fully Assembled Bogie

Next Steps

Though a lot of progress was made this year, only a few critical components have been fully developed. Nearly everything other than the placement of the wheels needs design work, and the V1.0 model can be modified and improved to more closely resemble a final bogie design. The 2013-2014 Bogie team recommends the following aspects be considered and designed.

System Model Development

Though the wheel placement, the dimensions, and the distance between bogies has been constrained, the actual structure of the frame has not. What we modeled was based on our ability to build it. In reality, the bogie will need to be much more specialized, with cast parts and unique pieces. A bogie frame should be developed which can withstand all the forces that it will need to in real world operation.

Another consideration for the frame is the degree of movement for the H-Bar. The amount that the H-bar can rotate with respect to the bogies determines our turning radius. As of right now, the H-bar hits the plate steel with less than 10 degrees of rotation. Perhaps a design wherein the half-bogies don't rotate around their center could be designed, or a large connecting-rod like design could develop for the H-bar, wherein the bogie can articulate 360 degrees, and all the weight would be supported through the center of the connecting rod loop. This would allow the turning radius to shrink to a meter so the podcar can turn in place, making compact stations easier to design.

All other aspects of the design which are not represented on the prototype have yet to be placed in the system model. Continued research is called for. 2014-2015 students should be able to develop a 3D model assembly containing all the subsystems described in this document.

Bengt Gustafsson has interesting ideas about using sliding mechanisms rather than rockers to handle the steering wheels.

Another approach which bears some consideration, although it is problematic, is to allow the bogie to be unpowered, and have a cable in the guideway which the bogie clamps on to like a cable car. Such a system could reduce the possibility of podcar collisions, though it would mean coasting around junctions or corners.

Prototype V1.1

The Prototype built for Maker Faire was specifically designed to let systems to be tacked onto it. At minimum, the 2014-2015 students should succeed in installing a propulsion system and complete the steering system. Even if a rocker/latch mechanism is not built, there needs to be an additional moment-counteracting wheel for guideway junctions attached to the steering arm.

The H-bar is not difficult to replicate, and if a different shape would be beneficial for mounting components, a new H-bar can be fabricated and installed.

The V1.0 Prototype is the first suspended ATN bogie prototype in the country, and is worth about $6000 in materials. Use it to promote the cause of ATN and adapt it to be a better demonstrator.

Scale Model with Switch

Because building a full scale guideway switch takes not just money, but land, it would be advantageous to build a 1/12 scale model of the bogie with a working rocker steering arm and a guideway switch that can be moved by hand. It could be built using the 3D printer, laser-cut acrylic, or ME41 shop skills. The

rocker/steering arm can be screwed into position. This will be useful in proving to the skeptics that our design will cleanly navigate junctions.

Cabin Team

After reviewing the 2012-2013 Cabin Team's design work, the 2013-2014 Cabin Team proceeded from where they left off. In the 2013 Fall Semester the current Cabin Team, along with the entire Superway Team conducted a field survey of our target market to further justify the design work previously done and the slight changes made to this years Cabin.

The biggest change was made to the Cabin's size and interior configuration. Based on the survey 20% of the riders interviewed mentioned that overcrowding at the station and on the trains was a major issue for them, especially during commuter hours. Therefore, the Cabin Team decided to slightly increase the interior size of the previously designed cabin from 80 inches to 120 inches long illustrated in Figure 1. This extra length still allows for 4 people sitting but also adds more room for those that travel with bicycles and extra personal items while not feeling crammed and still enjoy a comfortable ride. In conjunction with the Industrial Design Team the Cabin Team also decided to add a bench on one side to accommodate families or riders with a more personal relationship.

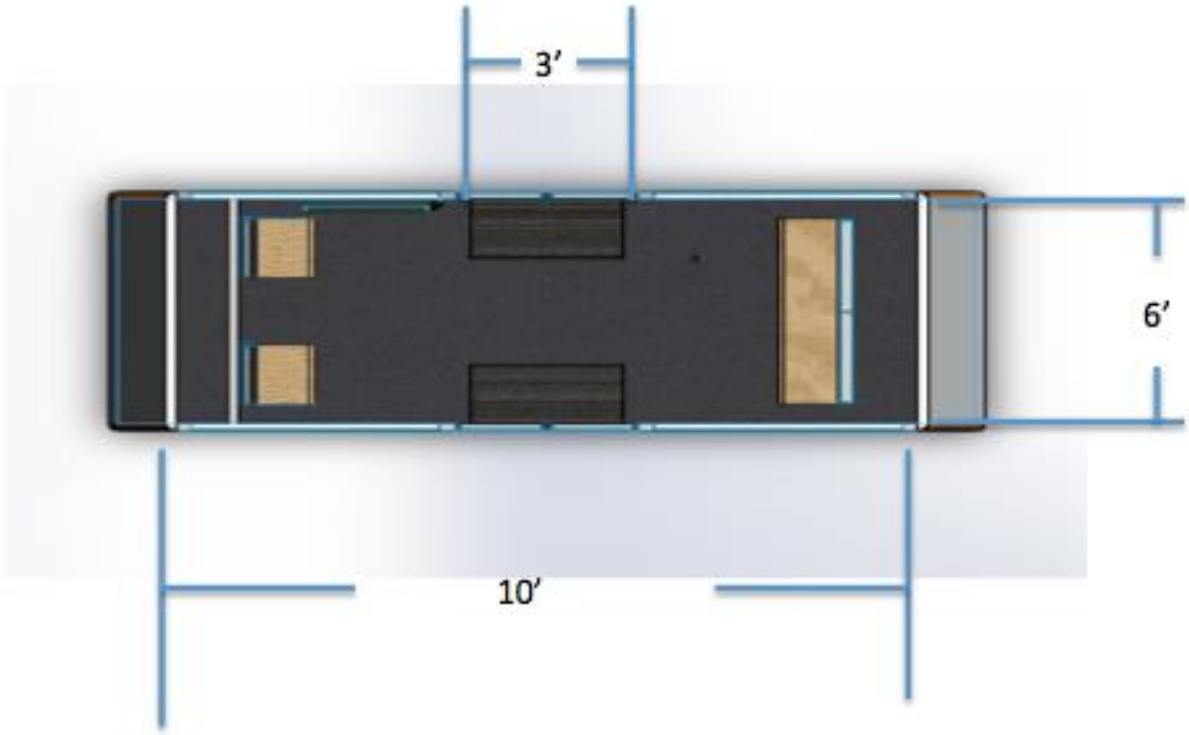


Figure 10. Interior dimensions of cabin

Theoretical System Model

Frame

The frame structure is always a main concern when it comes to vehicles. It allows the vehicles to properly support the passengers and hold everything together. Of the 3 conceptual designs of the cabin developed in the 2013 Fall Semester the frame shown in Figure 2 is the one that was chosen. The frame was chosen based on ease of FEA simulation and the ease of manufacturability once the cabin goes into construction phase in the future. The frame base shown in Figure 3, where most of the load will be concentrated, was derived and slightly modified from the frame design developed last semester.

The cabin frame must have a durable and lightweight structure, while being cost efficient for manufacturing purposes. Based on common automobile chassis and frames the team decided to use hollow aluminum 1060 for both side sub-frames and hollow alloy steel for center main frame. The purpose for using aluminum for the side sub-frames is because it is lighter, meaning it will allow the cabin to use less energy and higher performance.

According to the Center for Disease Control and Prevention (CDC) the average male weighs 195lbs. The frame was designed to support a max of 6 men at 250lbs, which is a safety factor of 0.78, this comes to a total of 1500lbs. For further reliability a safety factor of 2 was added to bring the total allowable supported by the cabin to 3000lbs.

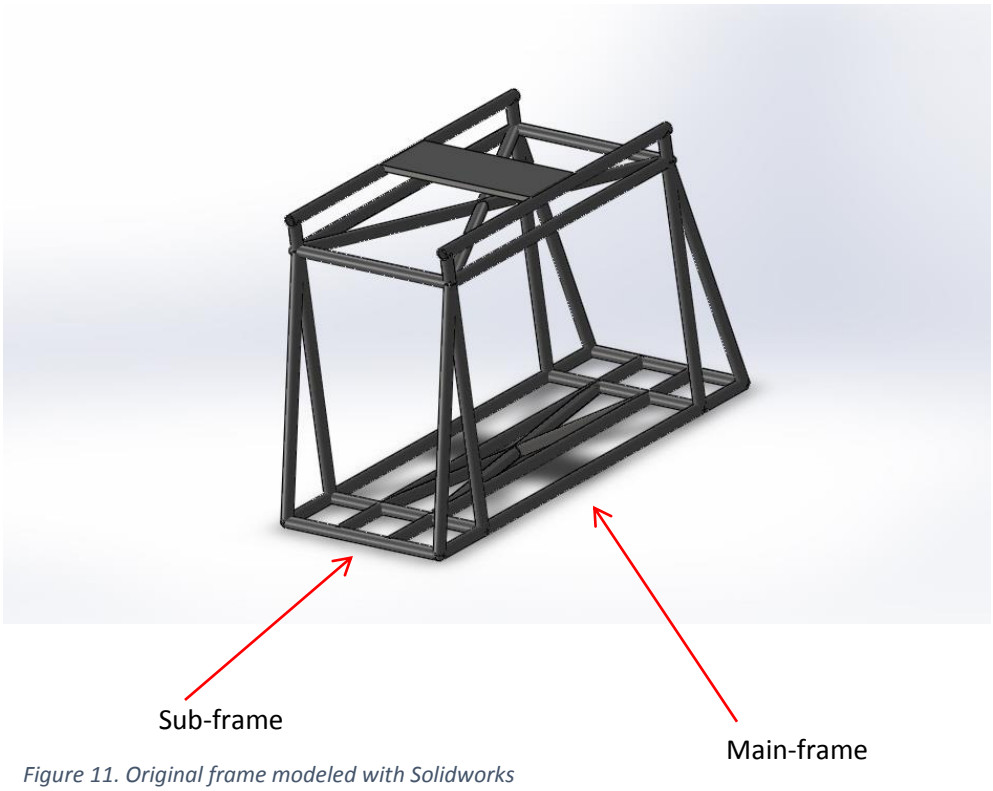


Figure 11. Original frame modeled with Solidworks

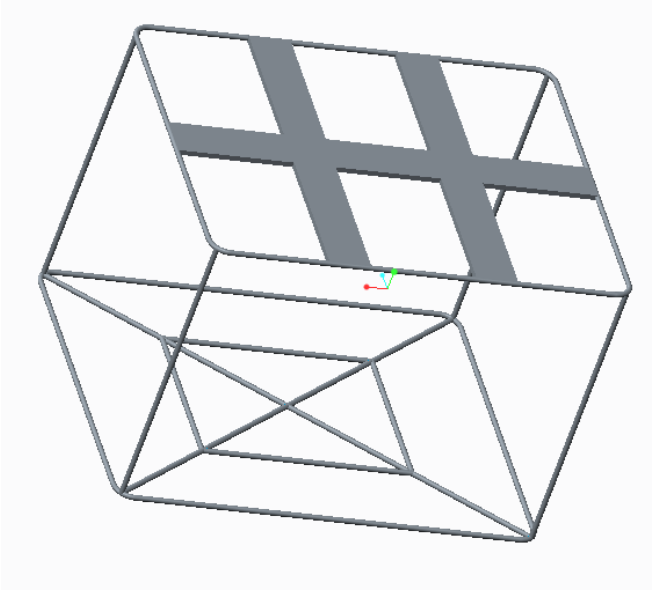


Figure 12. Modified main-frame section of cabin

Finite Element Analysis (FEA) was done using Creo Parametric 2.0 and the results are shown in Figure 4 and Table 2. The cabin is overly designed and will hand the load of passengers with minimum stress and negligible deflection.

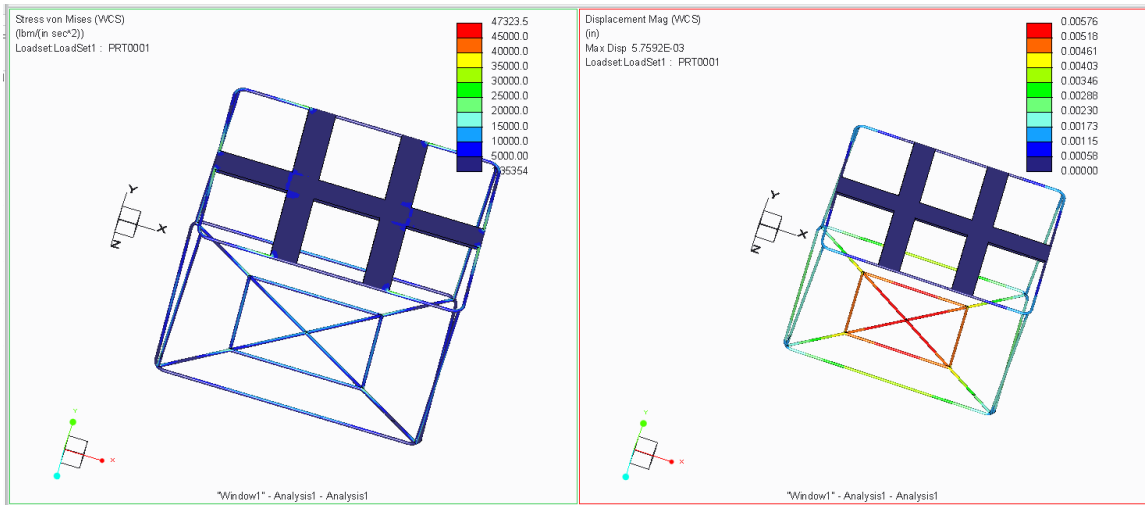


Figure 13. Creo Parametric 2.0 FEA results for von Mises (left) and displacement (right)

Table 4 - Frame Properties

Tube Materials	Aluminum 1060 and Alloy Steel
Materials Tubing Size	1'' OD x 0.095'' Thickness
Frame Weight	266.88 lbs

Table 5 - FEA results

FEA Software	Creo Parametric 2.0
Constraints	Fixed at bogie attachment locations
Passes	6
Yield Strength	63,100 psi
Maximum von Mises	47,323.5 psi
Maximum displacement	0.00576 in
Factor of Safety	2

The frame was modeled based on the several industrial designs and the cabin team decided to go with the trapezoid prism. This frame structure is aerodynamically efficient as well as esthetically pleasing. For this design, the cabin would be able to withstand an incoming air velocity between 25mph to 50mph. The team calculated the cabin would travel at an average speed of 35mph. Although 30% higher, due to the increased size, than 2012-2013 Cabin Team's design, while assuming the same drag coefficient of 0.8, from Figure 5, the following calculations shown below show an acceptable drag force. To compensate for the increased drag force the dual material frame is lighter than previous Cabin Team's design. This will allow the cabin to keep the power consumption to a minimum.

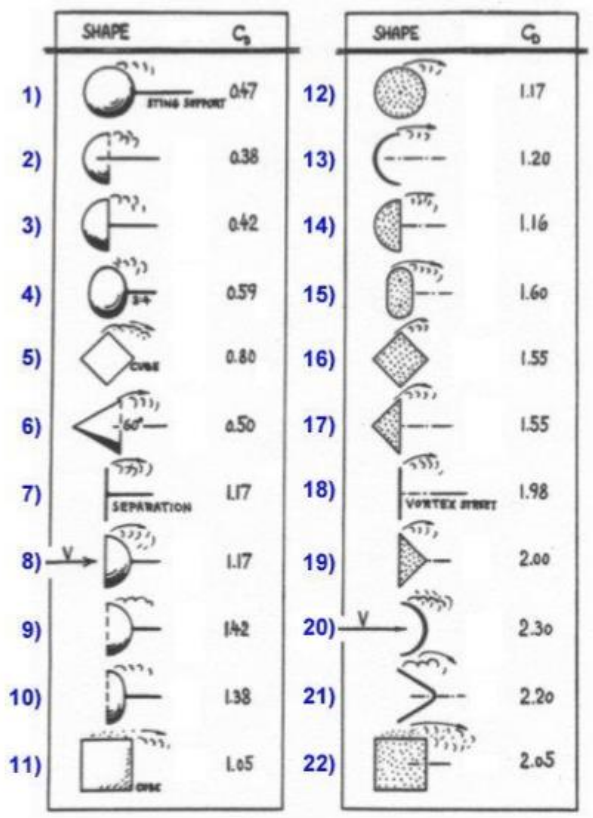


Figure 14. Drag coefficents

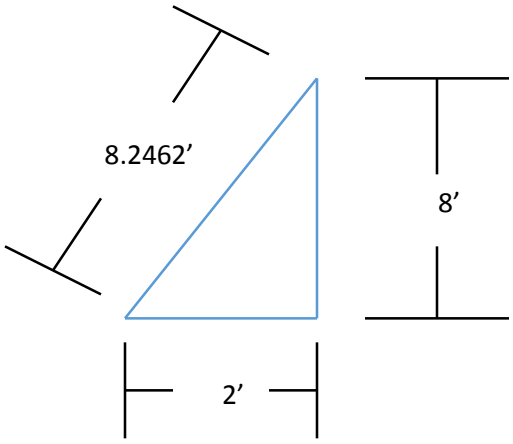


Figure 15. Side cross sectional view of sub-frame dimensions

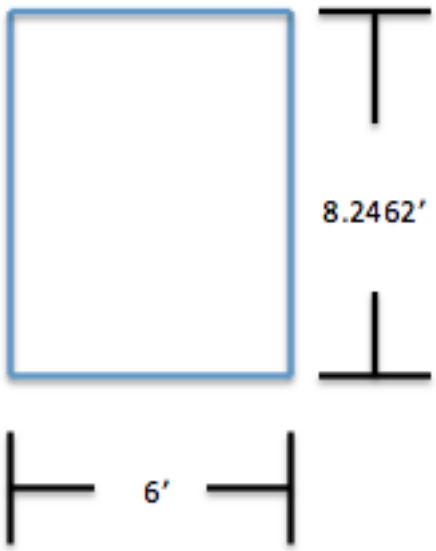


Figure 16. Front cross sectional view of sub-frame dimensions

Table 6 - Unit conversions

	English (in)	SI (m)
a	2	0.6096
b	6	1.8288
c	8	1.8288
d	8.2462	2.5134
Area (b*d)	49.4772	4.5965

Aerodynamic resistance formula:

$$F_D = \frac{1}{2} * \rho * v^2 * C_D * A$$

$$= \frac{1}{2} * 1.225 * 15.65^2 * 0.8 * 4.5965 = \mathbf{551.6}\boldsymbol{N}$$

Table 7 - Equation symbols and values

Variable	Symbol	Value	Unit
Drag Force	F_D	551.6	Newtons
Density	ρ	1.225	kg/m^3

Velocity	v	15.65	m/s
Drag Coefficient	C_D	0.8	unitless
Area	A	4.5965	m^2

Exterior

The exterior of the cabin is always the main concern for aesthetically pleasing and aerodynamics purposes illustrated in Figure 8. Based on the calculation, the team decided to go with a trapezoid shaped cabin to allow bi-directional travel with the least amount of drag force. The cabin will have doors on both sides to provide bi-directional entrance and exit for the passengers. The doors are 3ft wide to be ADA compliant and 7ft tall the height of a standard door. Providing enough of a scenic view for riders to enjoy was taken into the design consideration therefore, the doors come with a 42.38 inches tall window, shown in Figure 9 and has 31.8in tall side windows for seating passengers. Based on the transit industry, most of the vehicles are using laminated glass to reduce the cause of injury if it breaks. The laminated glass will be manufactured by using 2 pieces of glass and one piece of transparent polyvinyl butyl plastic (PVB).

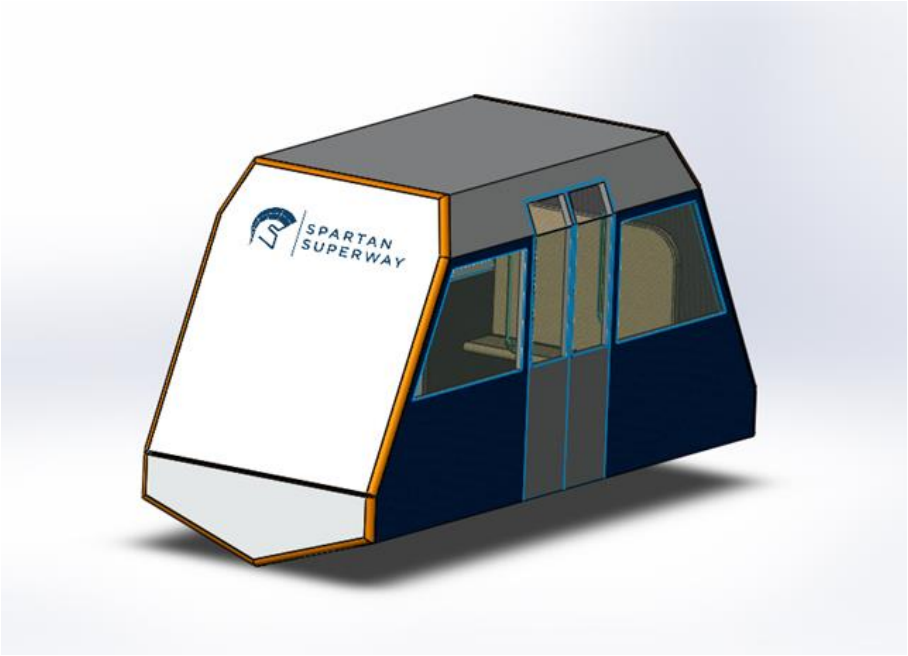


Figure 17. Exterior of the cabin

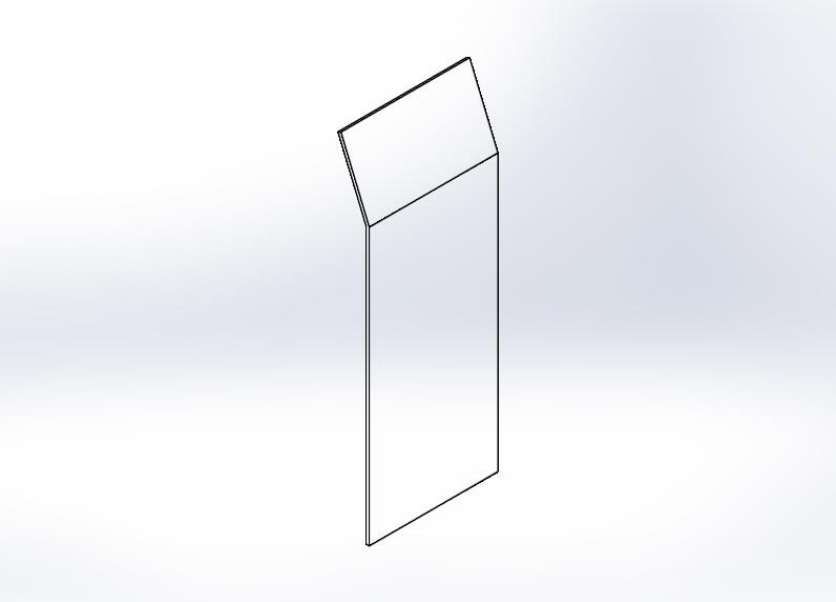


Figure 9. Door windows

Interior

The interior of the cabin is always the main concern for accommodating the passenger's needs. As previously mentioned, in order to be able to fit 4 to 6 passengers and provide storage space, the team decided to design the interior to be 10' x 6' x 8'. The interior of the cabin includes 2 one seaters and 1 two seater bench. Both seats and bench will be foldable and come from Freedman Seating, a company found by the 2012-2013 Cabin Team that builds seats for public transit use, Figure 10. The seats are foldable to easily create space for standing passengers and wheelchairs. There is 1.5 ft space to allow passengers to store bikes in middle of the 2 one seaters, Figure 11.

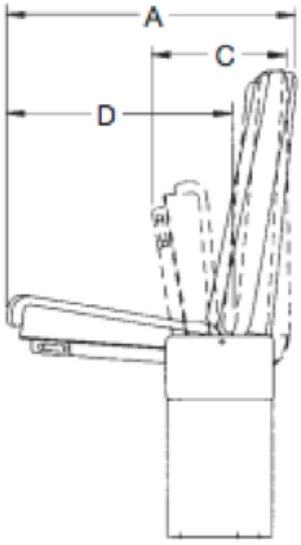


Figure 18. CitiSeat by Freedman Seating

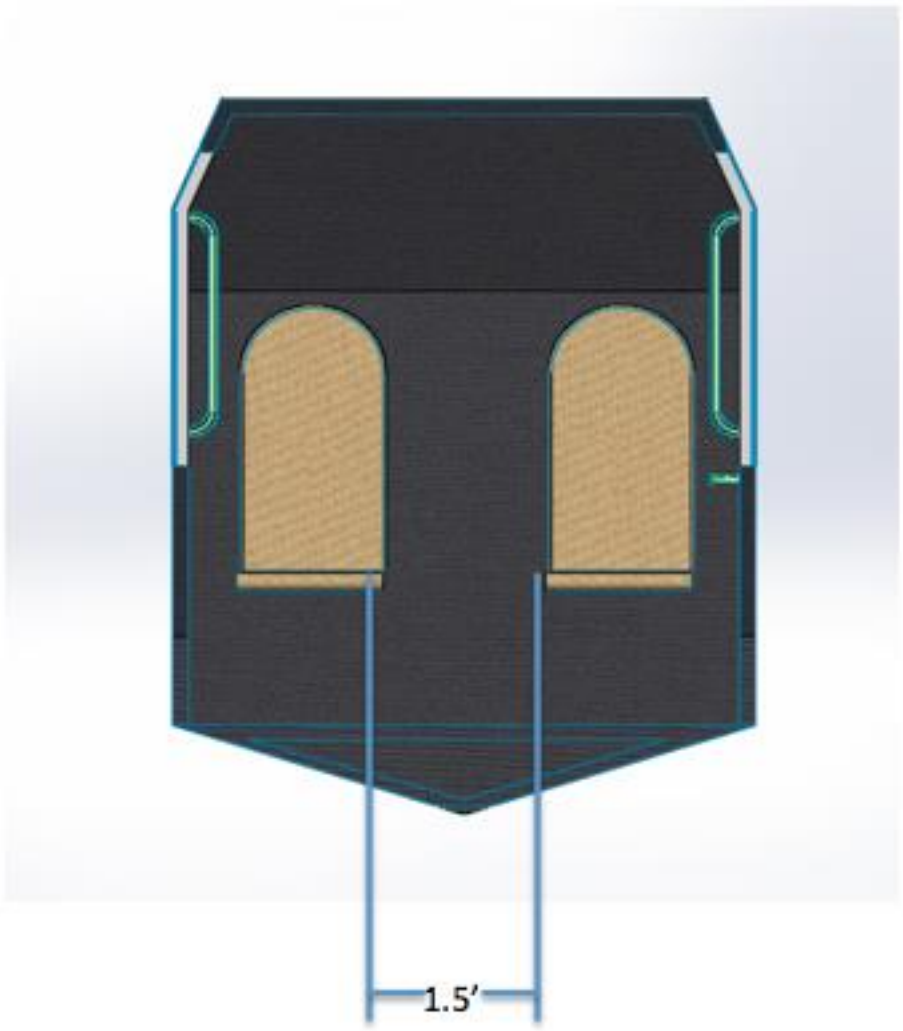


Figure 19. Distance between seats for bikes and other personal belongings

Figure 20. Side view of interior

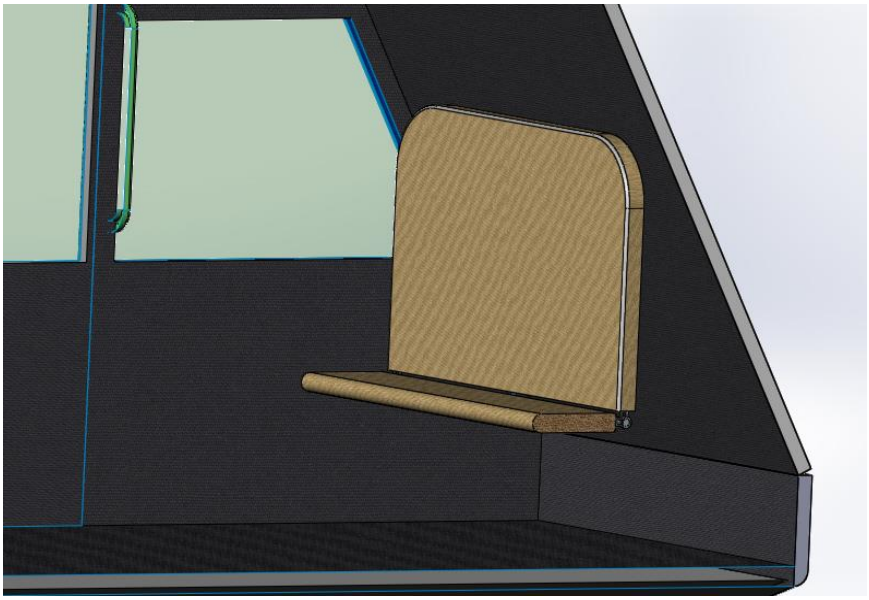


Figure 21. The 2 seater bench

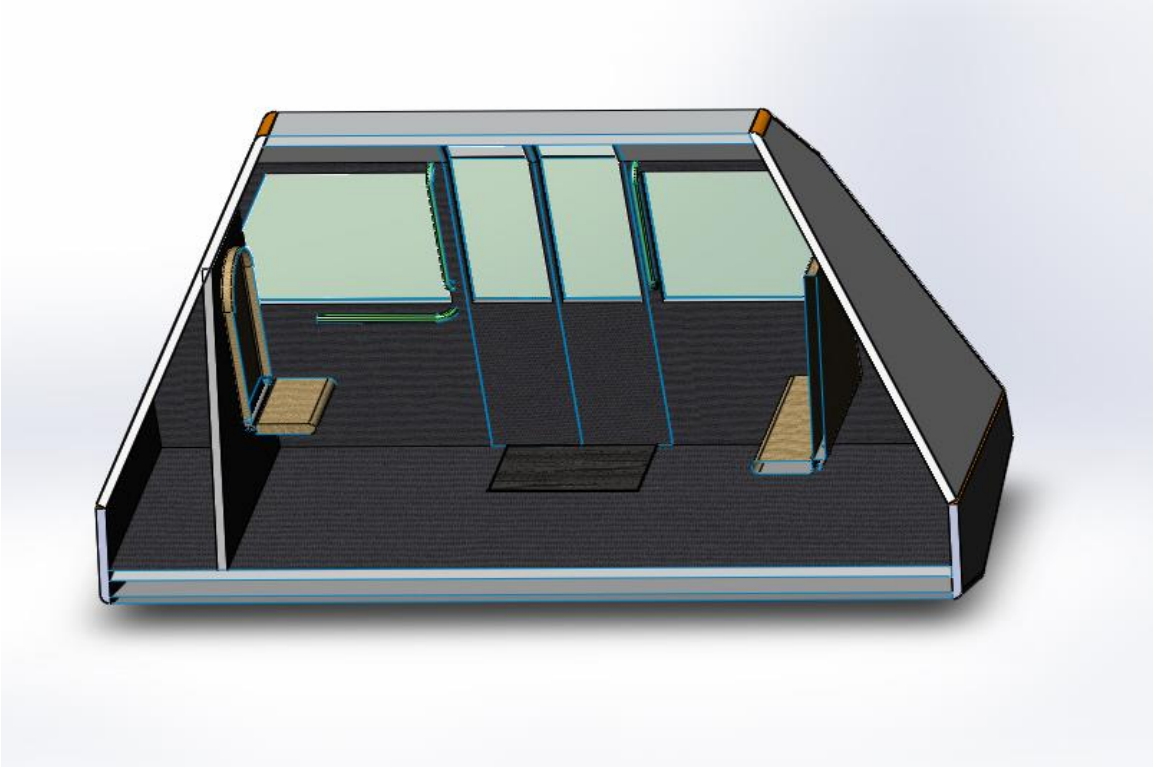


Figure 22. Section view showing the door mat and hand rails

Maker Faire Model

For the Maker Faire model, the cabin was donated by one of our sponsors, shown in Figure 15. Although not to full scale the cabin was chosen due to time constraints and because it easily represented the concept of how an actual cabin would attach to the bogie. Initially the model was a yellow snow coach. As a transformation to a more visually appealing cabin the team power washed then painted the exterior with multiple coats of white plastic dip spray paint and used a blue glossy interior paint for the strip down the middle, Figure 16 and Figure 17. Logo decals were added on either side of the cabin to give it a nice touch.

Figure 23. Original condition of donated snow coach

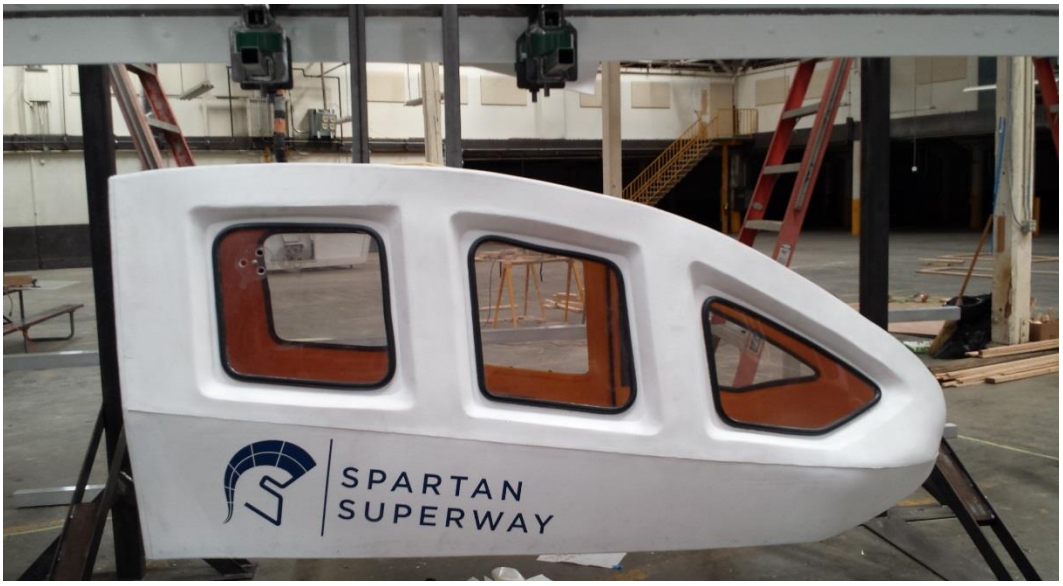


Figure 24. Finished cabin connected to the bogie by the H-bar at the warehouse

Figure 25. Completely assembled system at the 2014 Bay Area Maker Faire

The CAD model with the H-bar attachment for the bogie is shown in Figure 18. The cabin is 105 inches long and 45 inches wide. It has 3 windows on both sides; the back 2 windows are 28” x 28”. The thickness of the cabin is approximately 1.5 inches.

Figure 26. The Maker Faire Model with attachment modeled in Solidworks

In order to able to support and attach the cabin and the bogie together, the Cabin and Bogie Teams decided to attach a steel H-bar with a supportive cabin flat bar.

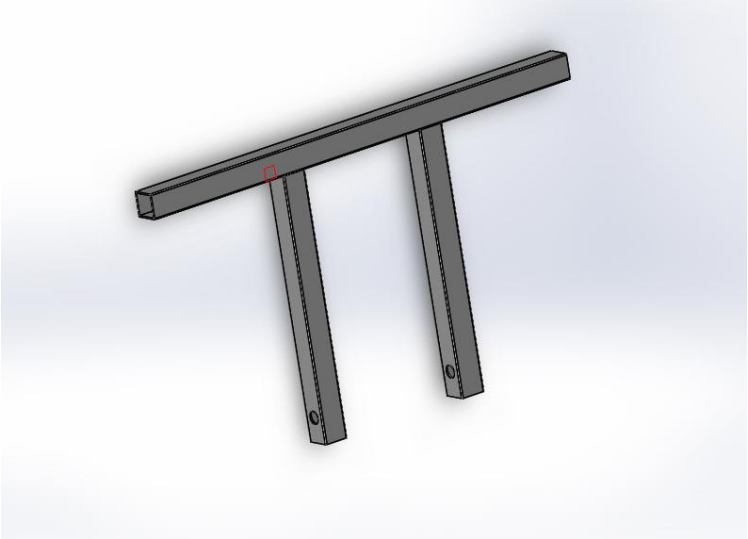


Figure 27. H-bar attachement

In Figure 19, the H-bar is 2” x 2” with the thickness of 0.125 inches. The extension bar shares the same geometry dimensions and it is 21 inches long. The screw hole is 2 inches above the bottom edge with a 1 inch diameter and a 1.25 inch spacer.

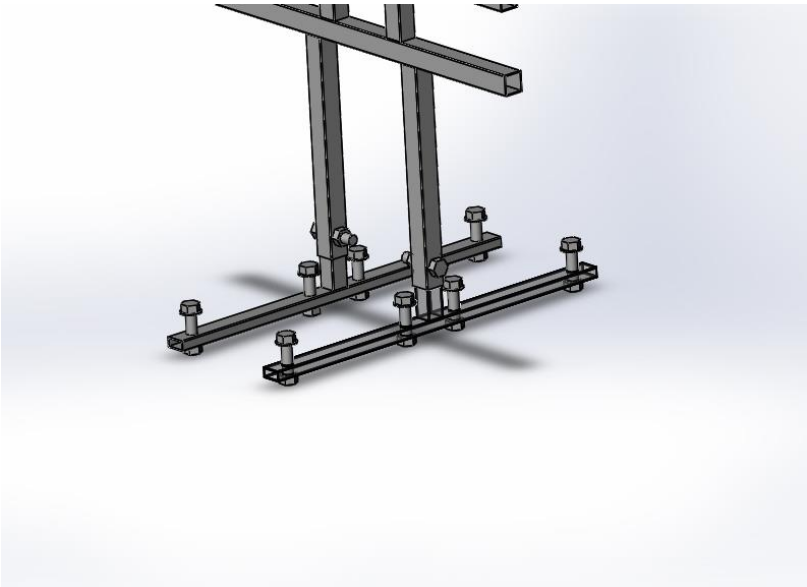


Figure 28. Supportive cabin flat bar

In Figure 20, the supportive cabin flat bar is 2x1in with the thickness of 0.125in. The flat bar will be installed from the inside of cabin and it will be supported by 4 screws on each flat bar. The centered

extension is 1.75” x1.75” with the thickness of 0.125 inches and it is 6 inches long. Also, it will go through the cabin and attach to the H-bar, which will be connected with a 1 inch screw.

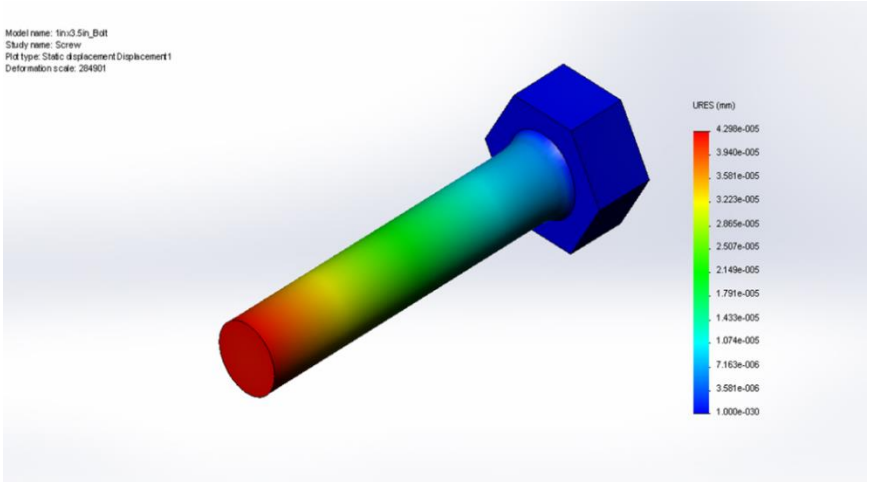


Figure 29. FEA of displacement of the screw with applied load using Solidworks

In Figure 21 and Figure 22, shows the 1 inch screw would be fully able to support the cabin. In this case, the team applied 300lb on the screw, which has 0.000042mm displacement and the applied load is greater than the cabin weigh (200lb).

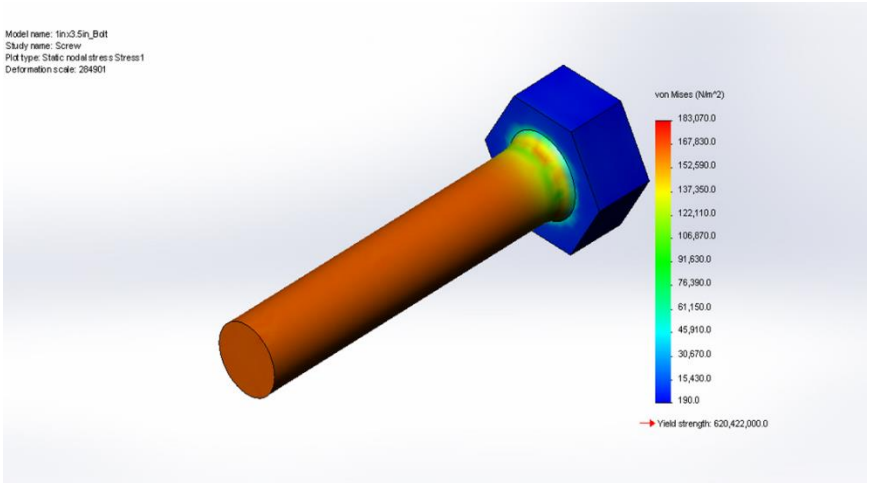


Figure 30. FEA of stress on the screw with applied load using Solidworks

Based on our analysis, we have concluded with this attachment design for the bogie and the cabin. This design would be able to easily attach and detach on the bogie.

Next Steps

Moving forward a functional full size model needs to be constructed with the advised design and materials. In order to accomplish this a scale model should be tested in a wind tunnel to determine the aerodynamic efficiency of the cabin design along with a more rigorous finite element analysis on the frame structure. Interior design and amenities need to begin to be incorporated into the cabin as well as user interaction capabilities. A cost analysis needs to be done on the selected materials to determine feasibility with allotted budget

Solar Team

What is the best way to spread the word about a solar powered Automated Transit Network? SMSSV wants more people in the public to learn about the idea of a solar powered ATN system, and the construction of a life size model at the annual Maker Faire in San Mateo, California would be the way to do that. The Maker Faire is an annual get together of thousands of innovators, inventors, and artists in the technology community. The goal of the Maker Faire is to bring together a group of like-minded individuals to inspire others to create, invent, or make something, anything, that can possibly improve our daily lives or for fun. SMSSV wants to show others why an ATN would be more cost effective than other public transportation networks that have been proposed and explain how the system is more sustainable than other alternatives. With a life size model, a 1/12 scale running track with moving pods, and a replica model of what a pod car looks and feels like, others will be able to better understand what ATN is all about.

Maker Faire Model

The main objective for the solar team in this second semester of the SMSSV project is to construct a life size solar panel frame that would then mount onto the guideway that the suspended pod car would be running on. The solar panel frame would be highly engineered to withstand the loads, stresses, and weather conditions that would occur in the area for the Maker Faire. From last semester, design requirements and design specifications have already been thought out for the final design. For the Maker Faire design, the same criteria will be implemented in order to see if those specifications will hold up to real world manufacturability. The five subsystems that will be used for the Maker Faire model will be:

- Power Collection
- Power Transfer
- Frame
- Guideway Mount
- Tracking

Each one of these subsystems will be driven by safety, system efficiency, manufacturability, and operation. This is to ensure that the Maker Faire model is created with the mindset that the model is a prototype to a real world model that can and will be made in the near future.

Power Collection

The means by which all power is to be generated for the Spartan Superway is through the solar power system. The goal is to have an efficient system that balances individual module efficiency and cost. Through much research, the modules incorporated into the final design are SunPower X21-345 Panels. It was determined by the Solar Team that these panels would best fit the design criteria established in the Fall Semester. The design is centered on efficiency, and the SunPower X21-345 panels boast a 21.5% efficiency rating, which is higher than most other commercial competitors. The more efficient each module is, the better the entire system will perform.

An added benefit in the decision of using the SunPower X21-345 panels is that they are among one of the most favorably aesthetic panels on the market.

Figure 31: SunPower X21-345

Power Transfer

Once the energy is gathered from the sun to the SunPower X21-345 modules, it must then be transferred throughout the system to provide power to the electrical systems within the Cabin, Bogie, Guideway, and Station. To accomplish this, the Solar Team has sourced an inverter that will provide a high efficiency as well as hold up to the robust power demands of the system. The method of power transfer the system will be using an Enphase M250 Microinverter. This inverter will be able to handle all possible loads from the solar modules, as well as preform at approximately 96% efficiency. These two points were critical in the Solar Team's assessment, but there are many more benefits. The Enphase M250 has a simple design, making it every easy to install and blend in to any location. In this micro inverter, the DC circuit is isolated and insulated from Ground, so no ground electrode conductor is required. This saves expenses on extra parts and labor. The Enphase M250 also has an option of including a 25 year warranty, insuring the reliability of the product.

Figure 32: Enphase M250 Microinverter

Although the goal of the Solar Team is to provide 100% of the power to the Spartan Superway system, it is still important that the inverters be tied to the grid. This is more a safety precaution and a backup plan within the system. It is always important to be connected to the grid in case of any power failure issues. This ensures that the Spartan Superway will be able to operate properly at all times.

Power Distribution Diagram

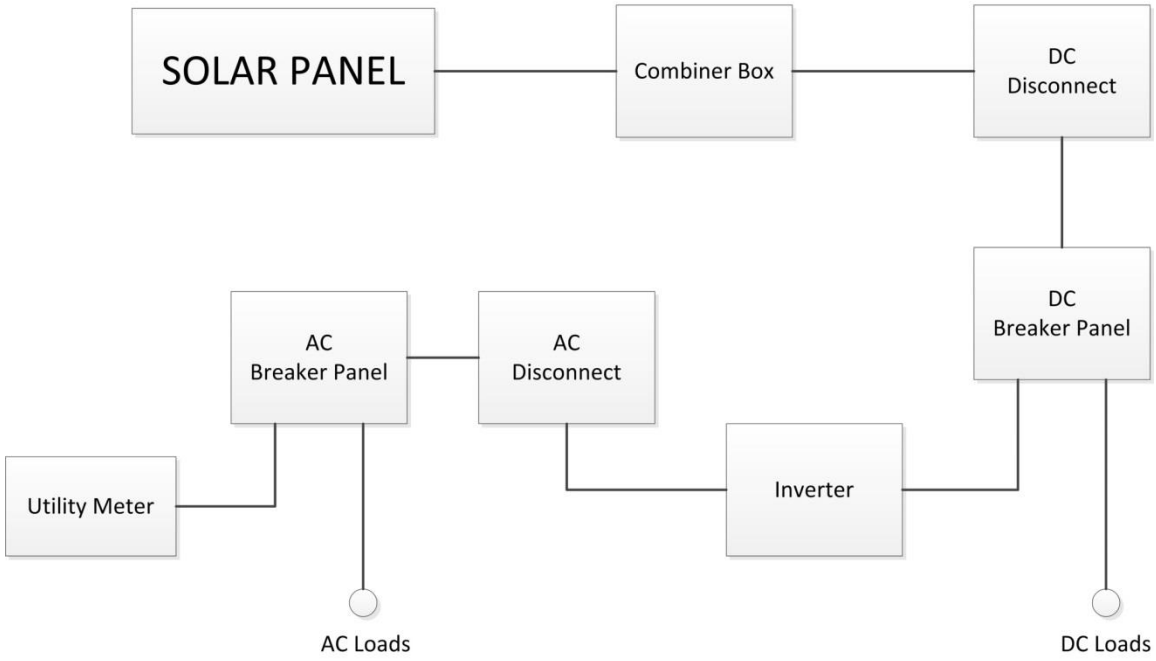


Figure 33: Power Distribution Diagram

Frame

The design behind the Maker Faire model was to keep it simple, yet make sure it fulfills its requirements of holding up the solar panels used to collect energy and to connect to the guideway itself. The material that was chosen was Aluminum because it is lightweight, easy to work with, yet durable. A 3" x 3" square beam was used for the frame with a 1/8" thickness. The design was a simple H-bar design with an overall dimension of 18ft x 2 ½ ft. The dimensions are based on the size of the solar panels used as well as the decision of how long of a guideway was to be made for the Maker Faire model. The frame is consisted of four pieces that were TIG welded together by the SJSU Civil Engineering department's technician, Pat Joyce. Pat made beautiful, strong welds on the pieces for the H-bar design so that they would be strong enough to hold together no matter what.

The structural integrity of the frame was also very important because real life applications were implemented in the Maker Faire model in order to see if our design model would be successful. Therefore, extreme weather conditions were considered as a precaution. Such conditions included maximum operating temperature of 133^{o}F, minimum operating temperature of -20^{o}F, maximum gust of 69mph, maximum sustained winds of 9.96mph, maximum days of rain of 7 days, maximum days of snow of 2 days, and just in case, maximum earthquake strength of 9.48 on the Richter scale. Of all these conditions the one that was most realistic was the maximum gusts because the solar panels would be so high up above the guideway and attached to the guideway columns, high winds in the San Mateo area could definitely become a factor in tipping the whole structure over if not engineered properly.

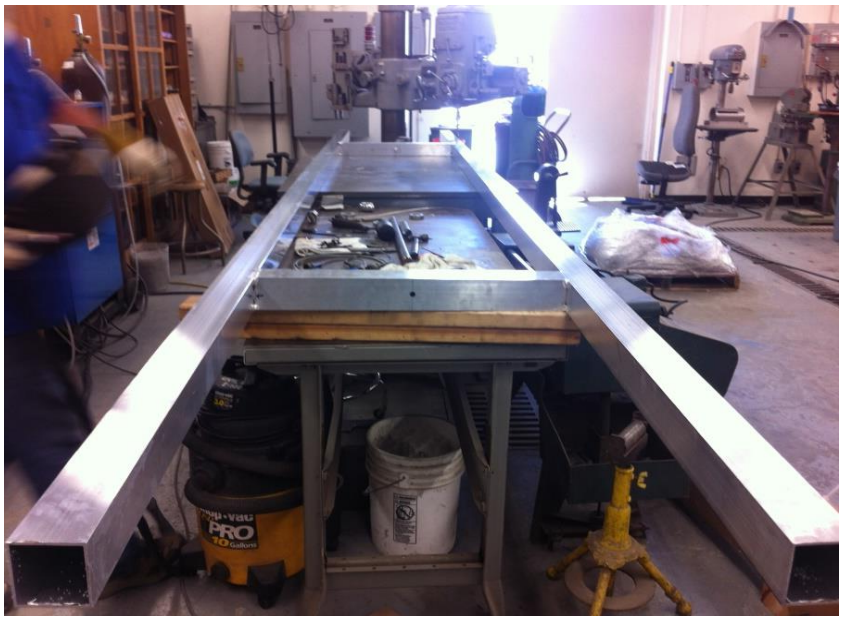


Figure 34. Simple H-bar design of the solar panel frame with aluminum as the material of choice.

Figure 35. SJSU Civil Engineering Technician, Pat Joice doing the welding of the individual pieces of the solar panel frame.

Figure 36. Final outcome of the aluminum solar panel frame with the thin film solar panels that would be later on mounted onto the frame itself.

A static failure analysis was performed in order to verify the design specifications set by the engineering team. This simulation was first performed at the H-frame itself to consider the extreme wind loads of 788 lbf. This design parameter was derived from assuming a gust wind force of 69 mph loaded onto the H-frame. As a result, a maximum Von Mises Stress point occurred at the center of the frame (9 ft from the end points). Here we can see a maximum Von Mises Stress of 3000 psi and maximum displacement of 1.173e-001 inches.

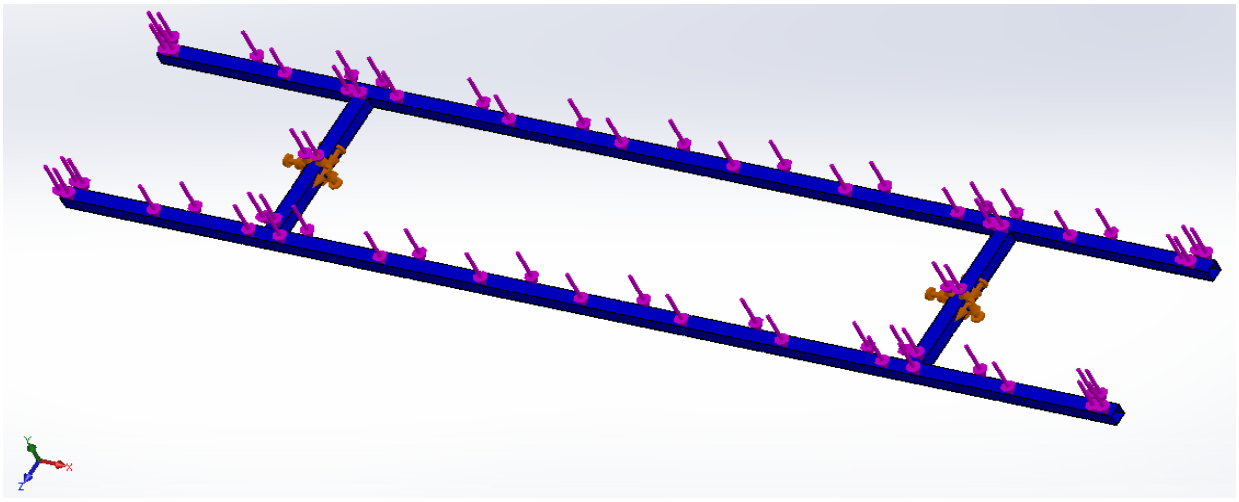


Figure 37. Wind Load Simulation – Static Test

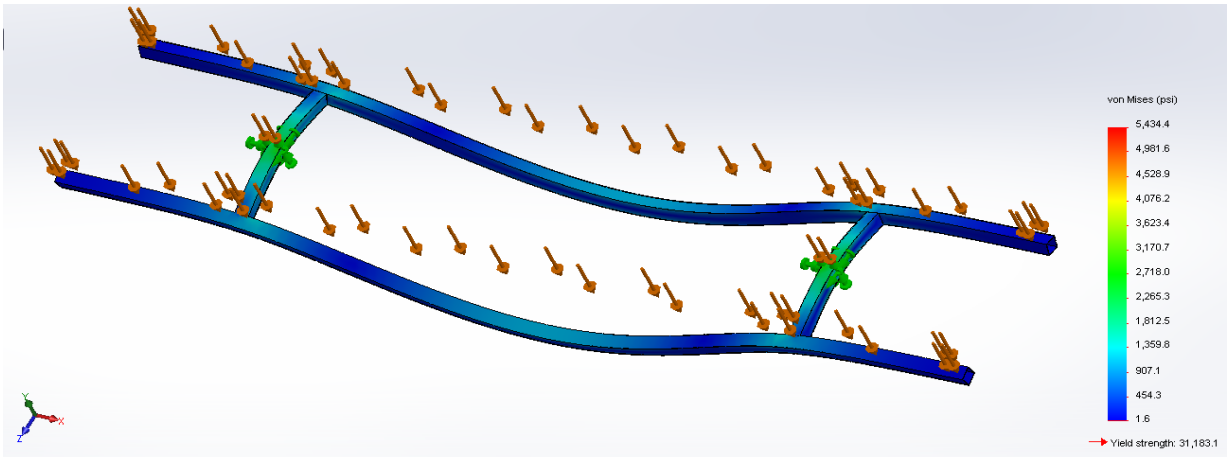
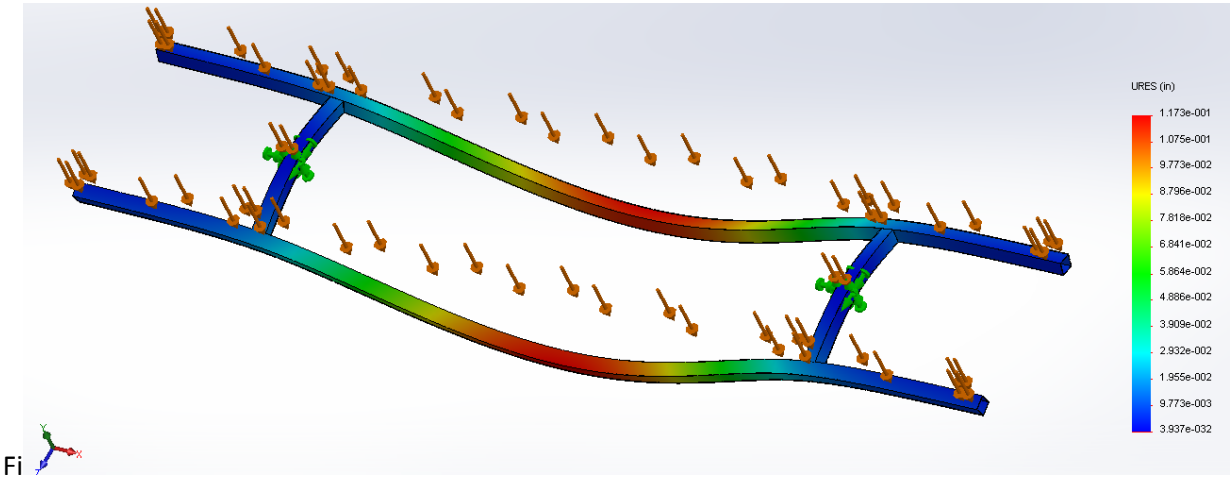


Figure 38. Von Mises Stress Plot



Fi

Figure 39. Displacement Plot

Guideway Mount

The mounting of the solar frame to the guideway is important because the structural integrity of the overall system depends on it. Just like the theoretical model, if by some chance the solar frame gets dismounted from the guideway then the whole system would lose a portion of its power because the solar panels are connected to the frame. To ensure a solid interface between the solar frame and the guideway, engineering analysis must be taken into account before the actual construction and manufacturing of the mounts. Finite Element Analysis for loads, stresses due to loads, and displacement were done on two parts, the frame mount and the guideway mount. The two mounts are between a three foot column to increase the visibility of the solar panels and increase its efficiency. Both mounts were also designed for easy installation. ½ inch hole were to be drilled into the mounts and the guideway with a bolt to connect the two together. Once the guideway mount and guideway column are aligned, the two can be easily bolted together, and likewise for the frame mount. This is a necessary step for the final theoretical design as well because in real world applications, SMSSV wants to design a system that saves time and money when it comes to installation.

As a result of the discussion of the mounting points with the solar panel to the guideway, a static failure analysis was performed at both mounts on each column in order to verify the satisfaction of engineering design. From the analysis, loads were applied on the guideway and frame mount at the surfaces where most deflection would be seen due to wind loads and fixed where the machine screws would attach to the guideway and frame column. From the images as a result from the analysis seen below, the highest stresses were seen at the guideway mount showing a maximum Von Mises Stress of 30 ksi, which showed the lowest safety factor of 1.67. This is due to the mount showing deflections on the H-frame due to high wind loads and potential failure locations in the compression of the mount.

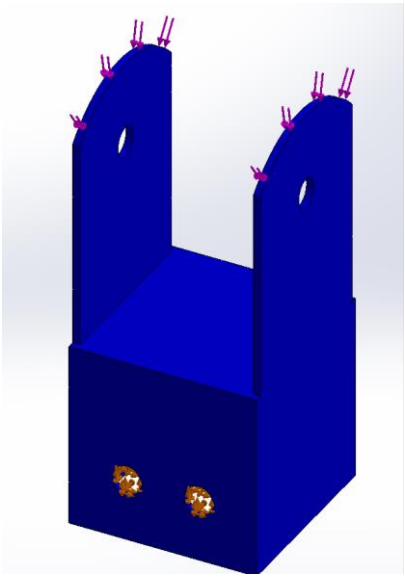


Figure 40. Frame Mount – Static Analysis Setup

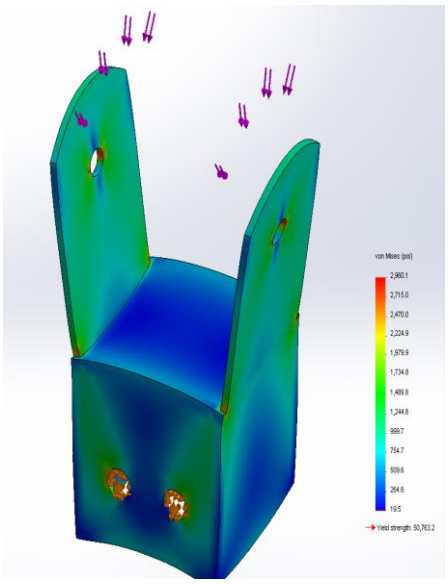


Figure 41. Frame Mount - Von Mises Stress Plot

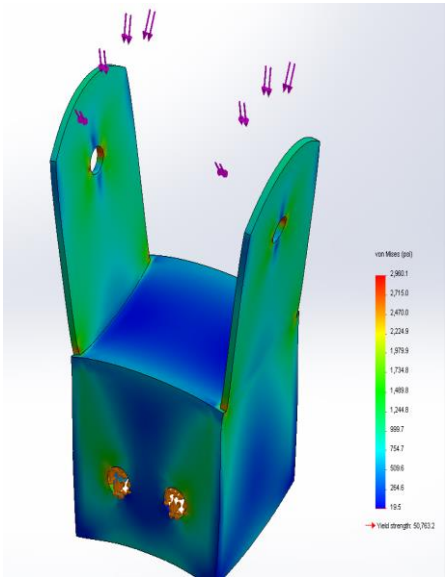


Figure 42. Frame Mount – Displacement Plot

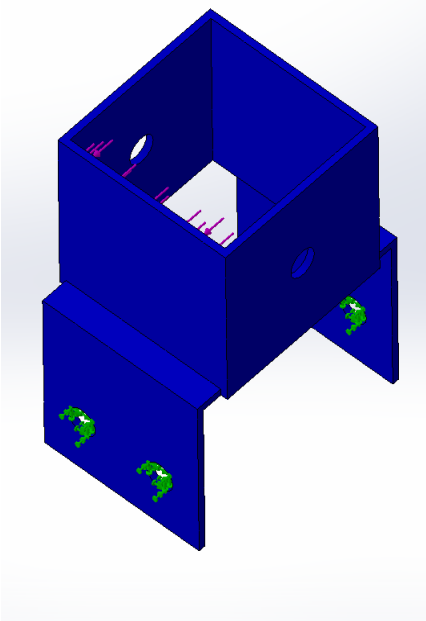


Figure 43. Guideway Mount – Static Analysis Setup

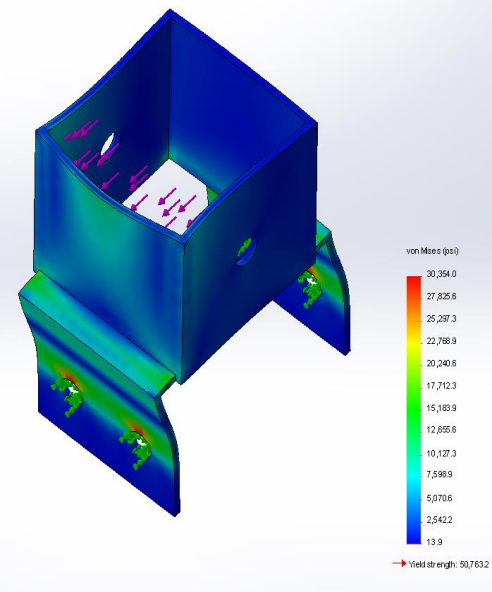


Figure 44. Guideway Mount - Von Mises Stress Plot

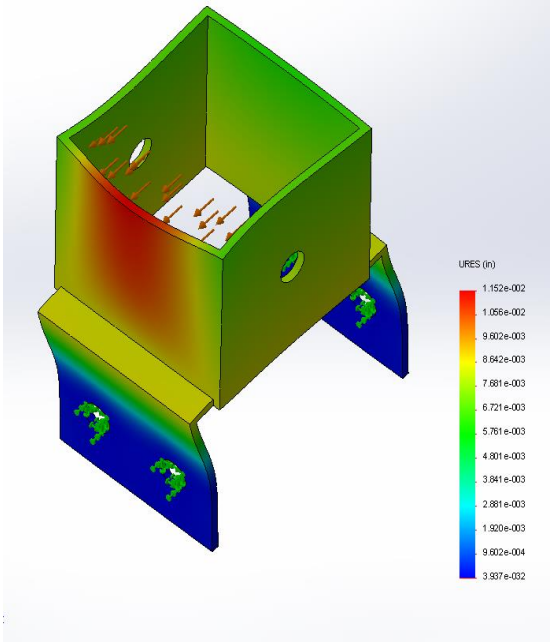


Figure 45. Guideway Mount – Displacement Plot

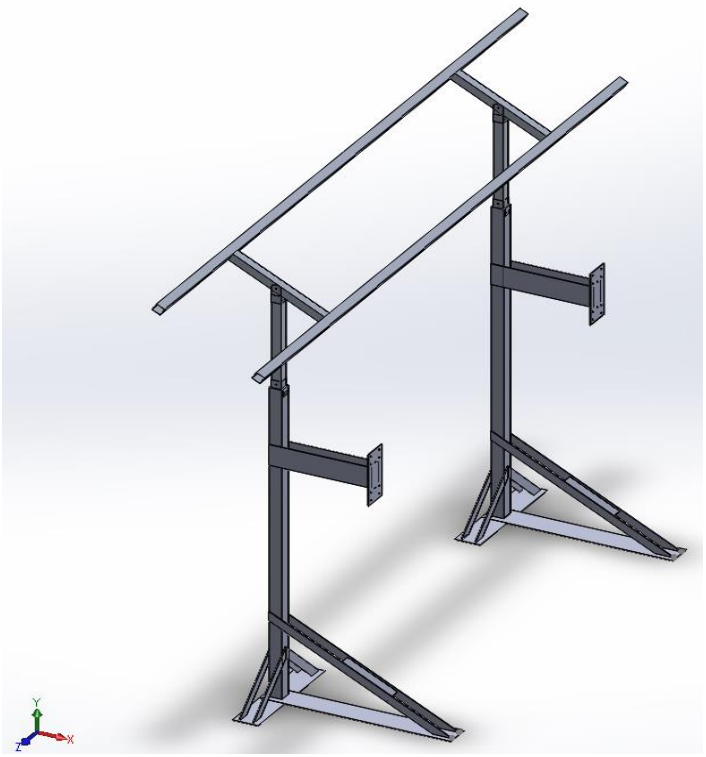


Figure 46. Solar Panel Frame – Maker Faire Final Design

Figure 47. Guideway with Solar Panels – Maker Faire Final Design

Tracking

For the sake of the Maker Faire, tracking was not implemented into the design. The reason for this was because of the limited time we had to create the model and to test out its functionality. A realistic model would be static to represent how the solar panels would look on top of the guideway and to show how the solar panels collect energy for the system. The final theoretical design was designed for tracking to be more efficient by tracking the path of the sun throughout the day, so if the static mount works than the tracking system would be even more efficient.

Maximum efficiency drives the incorporation of solar tracking into the solar power system. To make the design as efficient as possible, the control system is designed for optimized system performance. A robust design is incorporated to ensure a durable drive unit and control unit. Tracking stability is a required feature of the system, and is integrated into the design via the control system and the physical design of the overall solar system.

Theoretical System Model

Design Requirements and specifications

The Superway Solar Power System design requirements are based weeks of research, interviews with solar engineering professionals, and team brainstorming sessions. Per accepted systems engineering practice, each design requirement is expressed as a single, discrete statement describing a specific need, function, performance level, quality, or constraint relevant to the system.

The list of requirements has been broken down into Goals & Objectives at various Tier Levels. The top-level goals and objectives are noted as [Tier 0] requirements. As top-level requirements, these objectives address the most basic solar power generation and functionality of the system along with additional goals of interest to the entire Superway Team. All lower Tier requirements are ultimately derived from these top-level goals and objectives.

The overall scope of the Superway Solar power system is to design and implement a system of energy generation that produces energy efficiently, reliably, and is easy to manufacture. The overall scope leads to the [Tier 0] design goals and objectives. The most important [Tier 0] goal that is executed in the design is ensuring that the Solar Power system generates enough energy to power the ATN system in its entirety, while remaining as a reliable source during the lifetime of panels. The execution of this goal is seen in the stretch of solar panels above every inch of the guideway. This maximizes the possible surface area available for the panels, while not having to take up any space on the ground. There is also an aesthetic appeal to the user seeing that form of transportation they are using is powered with a completely renewable source of energy. Another important [Tier 0] goal applied to the final design is to maximize the overall efficiency of the system. This goal is executed in our design through a single-axis tracking system. Applying this tracking system ensures that the solar panels will have the maximum exposure to sunlight at any given time during the day.

Throughout the design process the Solar Team has closely studied the works from last year's report in order to better this year's design. Along with last year's report, the solar team has conducted research on current weather conditions in Silicon Valley, other competitive solar panel companies, other possible inverters, and also tracking systems that are currently available. Comparisons and information on research are found in the Appendices B, C, and D.

In order to account for a situation that might occur that are out of our control, a safety factor of 1.2 is calculated into the team's final specifications. Knowing the maximum and minimum temperatures that the system can handle in the Silicon Valley area is crucial in order for the system to not overheat or freeze over. The maximum and minimum operating temperatures that we calculated were 133^{o}F and-20^{o}F, respectively. Wind is a major factor when it comes to safety of solar panels because high winds can dismantle solar panel setup or even shift its position which would result in inefficiency. A maximum gust of 69 mph and maximum sustained winds of 9.96mph is factored into the overall system. Another common weather factor that the team is considered is rain. The amount of rain can definitely affect the energy output if there are too many days of rain and not enough days of sun. A maximum of seven days of rain is accounted into the overall system. Although Silicon Valley has not experienced snow since 1976, as a precautionary measure, a maximum of 2 days of snow is also factored into the overall system. Lastly, the most common natural disaster that occurs in the Bay Area is earthquakes. These usually occur unannounced and for an unpredictable amount of time, so the worst case scenario is accounted for. On that note, the system is designed to sustain a maximum earthquake of 9.48 on the Richter scale.

Along with the overall system specifications to operate safely and normally, the overall system design also has specifications. In order for the solar system to transfer the energy output into the ATN, specifications have been determined for the local electrical grid. The voltage connections to the grid should are 208, 240, 277, 400, and 480 VAC. Also, the typical frequency synchronization is 60 Hz. Since the module system can induce an excess amount of energy the power transfer system can handle, a maximum produced power of 13kW requirement is applied. Although a single-axis tracking system is

implemented in the final design, it is important that the system has flexibility with rotation for the purpose of maintenance. With respect to rotation, the maximum vertical rotation is 360^{o} and a horizontal rotation of 180^{o}. The mount that will hold that solar panels is designed to hold a maximum of 8 panels and is able to support a combined weight of 537.6 pounds (67.2 lbs. x 8 panels, with 1.2 safety factor applied). Lastly, the overall system is required to support both off-grid and grid-tie inverters up to 15kW of power.

Conceptual Design

In order to evaluate the proper design, Pugh's method is used to determine relative importance with alternative design concepts in a basic decision matrix. This method is chosen instead of the robust decision matrix due to the evaluation being used to determine preliminary design parameters which will have more detail once a specific system is chosen. In this case four different design concepts for system specification were taken into account as seen in the figures below.

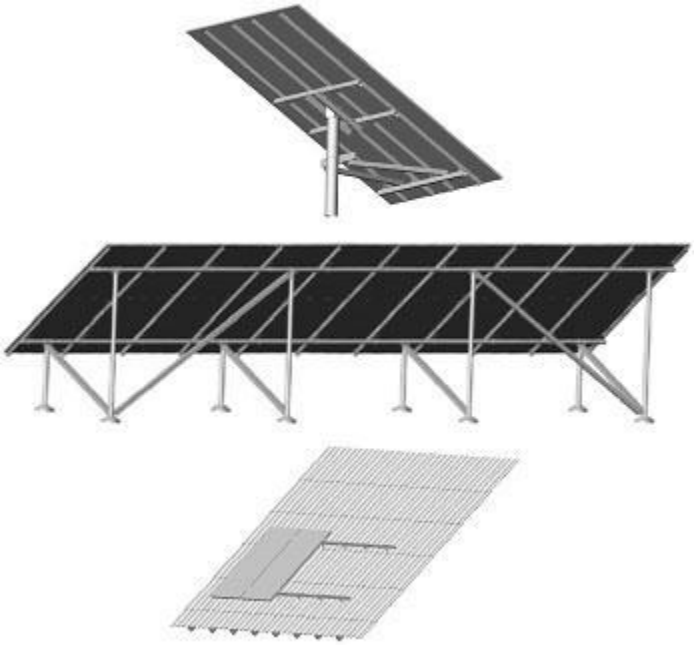


Figure 48. Static Mount Concept

Table 8. Alternative Concepts for Design Consideration.

Alternative Concepts	
Single Axis Tracking on a Horizontal Axis	Concept 1
Single Axis Tracking on a Vertical Axis	Concept 2
Single Axis Tracking on a Tilted Axis	Concept 3

Figure 49. Single Axis Tracking on a Horizontal Axis.

Figure 50. Single Axis Tracking System on a Vertical Axis

Figure 51. Single Axis Tracking System on a Tilted Axis

The four design concepts chosen were evaluated with chosen parameters including relative importance in order to decide which system is preferred based on the criteria shown in Table 2.

Table 9. Criteria for Basic Decision Matrix

Effect	Criteria
Postive	+1
None	0
Negative	-1

Table 10. Basic Decision Matrix

Basic Decision Matrix - Solar Panel Mount						
Issue: Choose a Solar Panel Mount System		**Baseline**	**Static Mount**	**Concept 1**	**Concept 2**	**Concept 3**
Mass Efficiency	15	**Datum**	+1	0	0	+1
Manufacturability	30		+1	0	+1	-1
Power Generation Efficiency	20		-1	+1	+1	+1
Safety	15		+1	0	0	0
Maintenance	10		+1	0	+1	0
Ease of Installation	10		+1	0	+1	0
	Total		5	1	4	0
	Weighted		60	20	70	5

Based on the evaluation of the design concepts in the basic decision matrix, the most important factor taken into consideration is the manufacturability and power generation efficiency of the solar mount. With these two factors along with the other parameters chosen, the results indicate that a single-axis solar tracking system on a vertical axis is the best design concept to be used when considering the solar panel mount design. This design concept seems to be the easiest system to design, manufacture and maintain because of less moving parts in this type of single-axis tracking system. The following figures show 3 dimensional models of the final concepts evaluated. Figure 7 is the final concept chosen for further design and analysis.

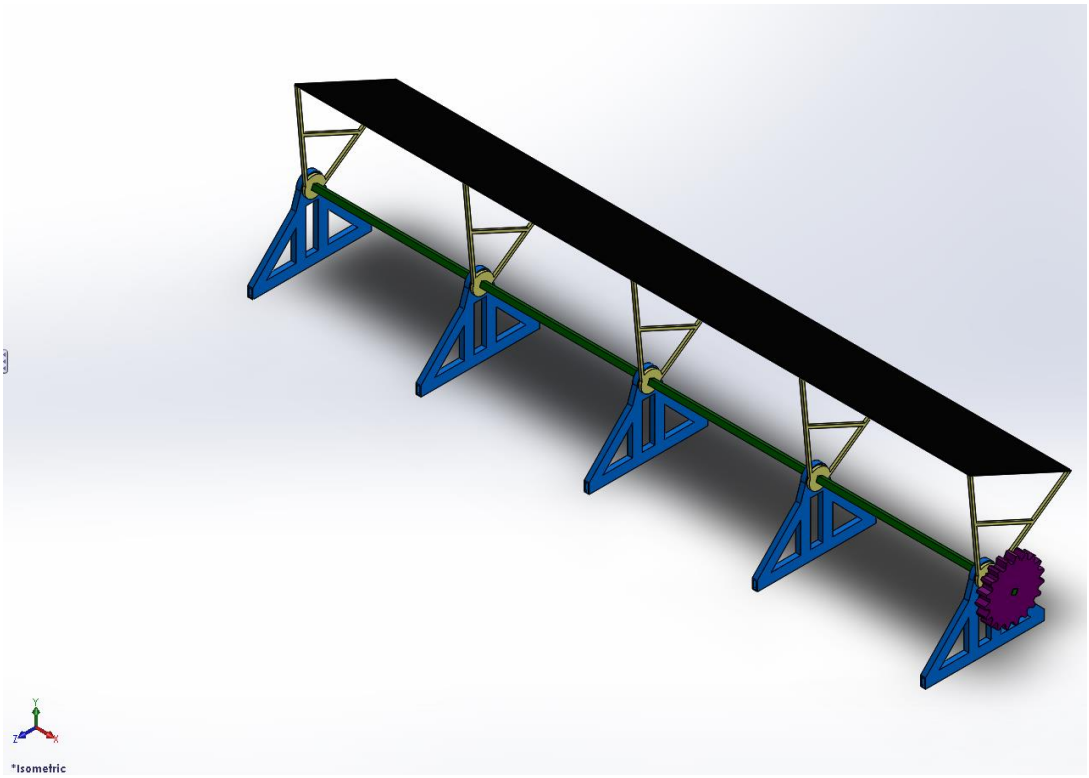


Figure 52. 3D cad design of initial brainstorm session designs.

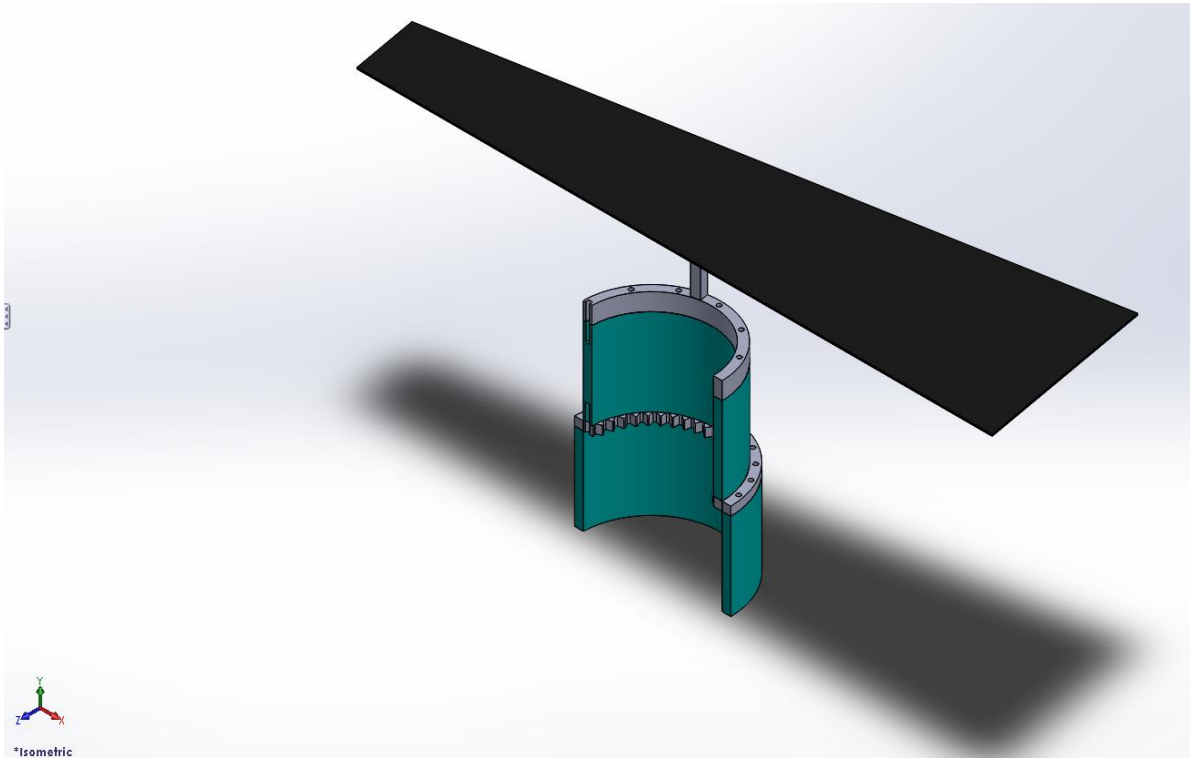


Figure 53. 3D cad design cut out of single axis system from brainstorm sessions.

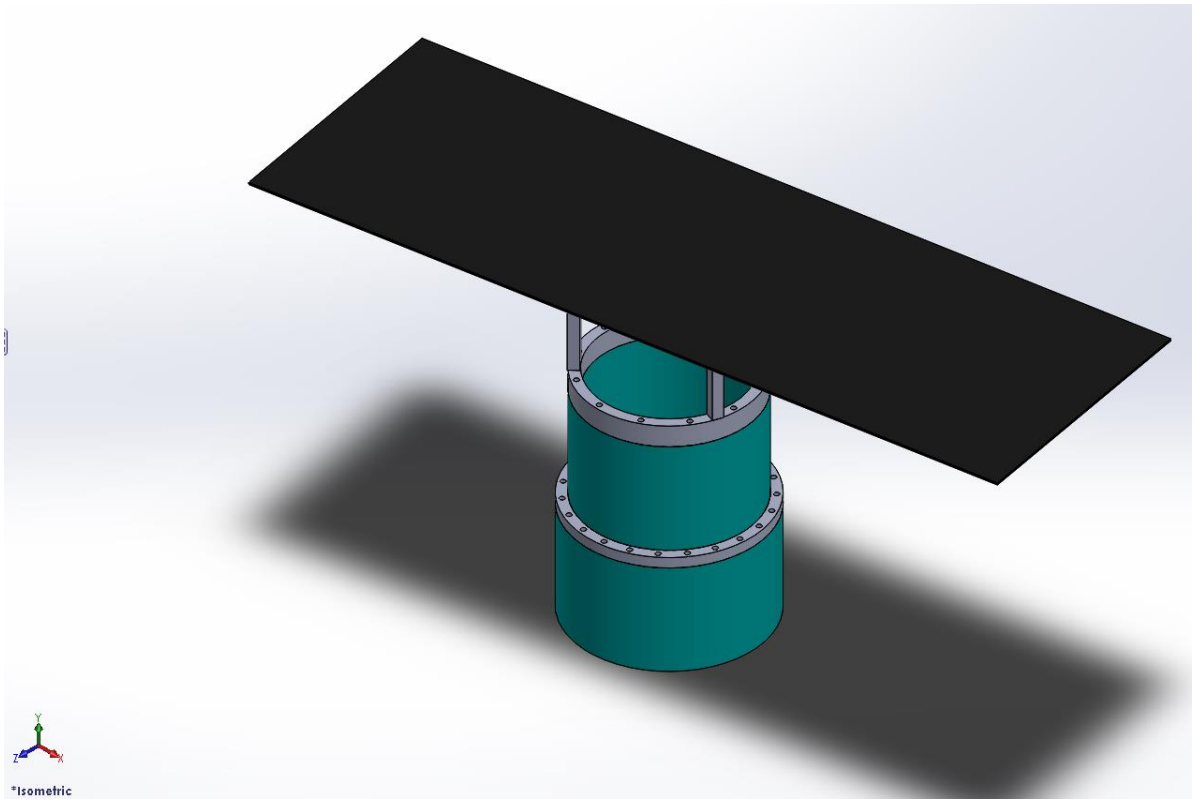


Figure 54. 3D cad design of single axis tracker concept derived from brainstorm sessions.

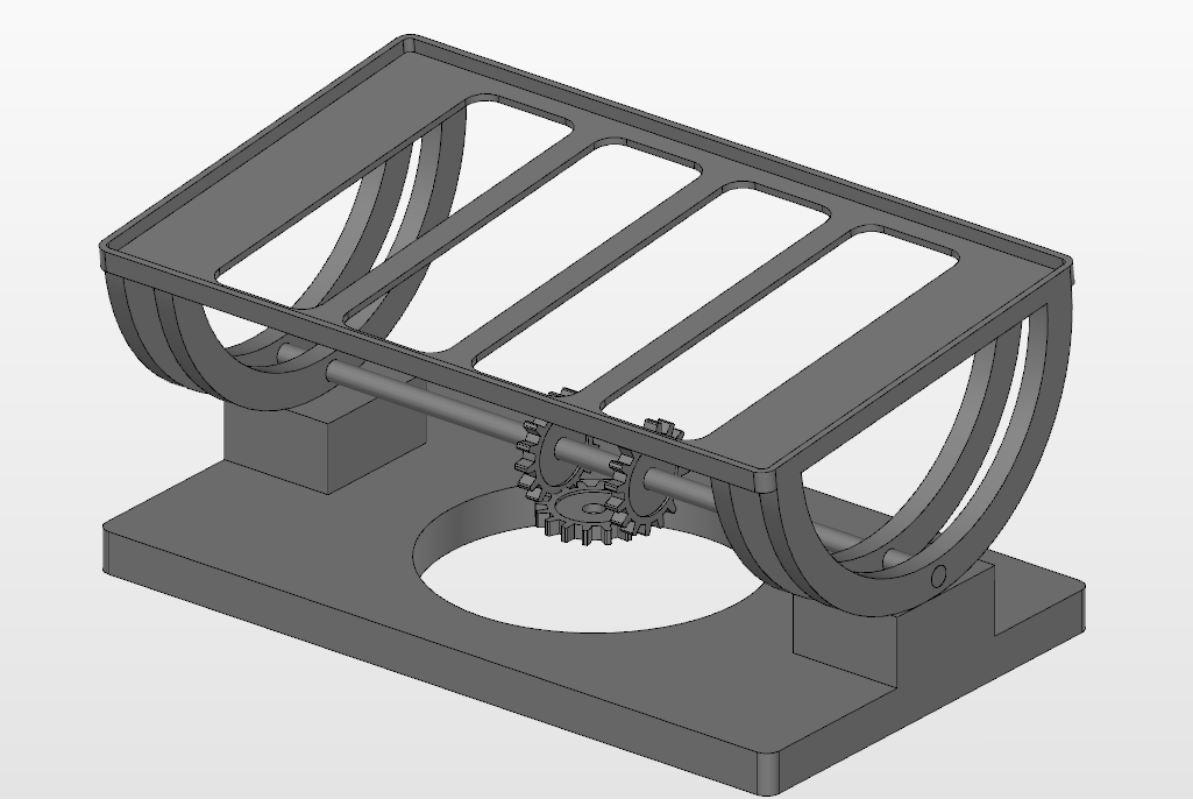


Figure 55. 3D cad design of horizontal single axis tracker derived from brainstorm session.

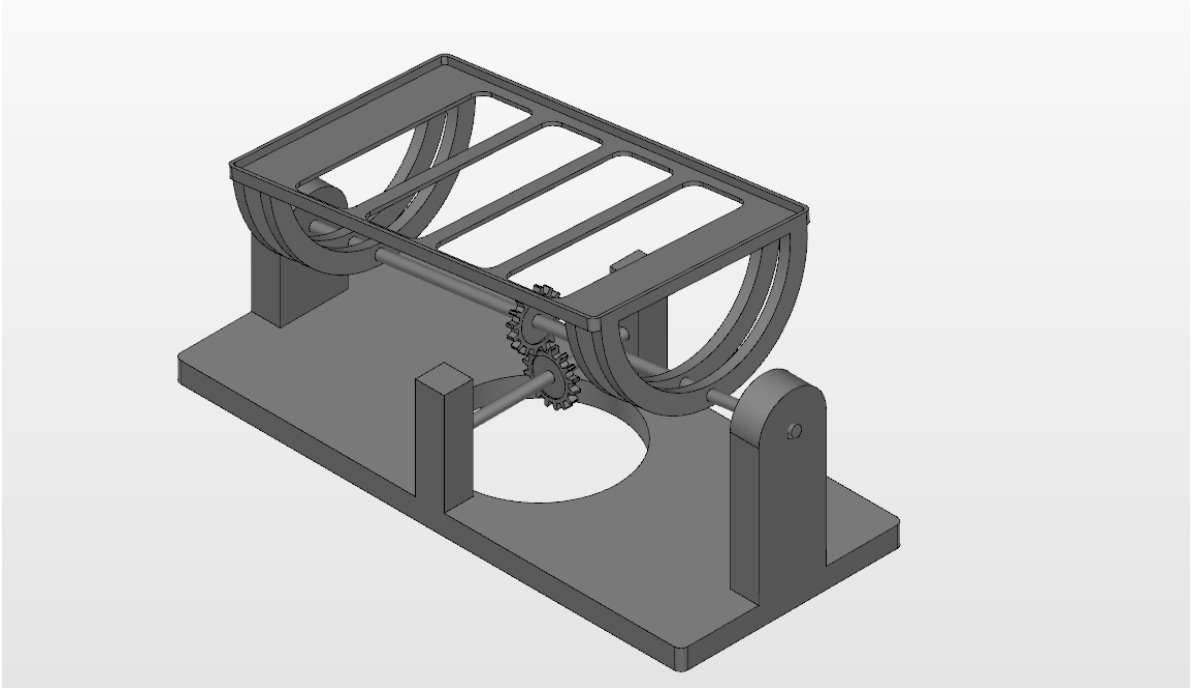


Figure 56. 3D cad design of tilted tracker derived from brainstorm session.

Next Steps

The final design is a single-axis tracking system and that evolves from research conducted at the initial stages of the design process. The next step in the project is to implement the specifications and do more in depth analysis on life cycles, critical stresses, costs of materials, construct a full scale prototype. The Life cycle analysis is to ensure that the solar system can last through the elements of nature and up time. Critical stress analysis is essential to understand how the mount is able to hold the solar panels and withstand the forces of the solar panels and other environmental factors. The cost analysis is to get an understanding of how much money the final design will cost. Besides this kind of analysis, it is important to come up with a system to install the design efficiently. If the guideway towers are going to be high above the ground, it is most efficient to assemble everything on the ground and use a system of some sort to hoist the structures safely into position. Another design feature that requires further development is the power transfer mechanism and how it integrates with the final design of the bogie and the electrical power grid.

The final design set for the maker faire was only a demonstration of the potential analysis that needs to be done in order to create a realistic design for the ATN system. A realistic design of the system should utilize a single axis tracking system which will require additional research and fund allocations in order to implement a dynamic system into the SMSSV Superway project. Due to time constraints and budgeting issues, the design of the solar power generation system was kept simple. Most of the challenges faced were seen in the manufacturing of the frame design and additional avenues could be explored for future designs. The mounting attachments to the guideway system were custom mounts that were created by weldments which in the future could be improved by creating a design that utilizes existing brackets and mounts seen on the market to ensure the feasibility of the design for the actual system if implemented. If resources are utilized more, the cost as well as the feasibility of the design of the solar power generation system in the future will satisfy system specifications.

Controls Team

The Spartan Superway Controls System sub-team (Controls Team) develops and implements the software-related components necessary to automate the transit network. At the highest level, this means that any scheduled trip on the Spartan Superway system will run entirely without manual intervention, from the moment a Superway rider purchases their ticket, to when they step off the pod at their final destination. The underlying responsibilities within the Controls System to achieve such a goal range from manipulating the pod motors to go forward, to providing a city-wide communications network capable of monitoring and scheduling pod travel routes.

The Controls team responsibilities primarily consist of three main areas: the network architecture, the pod local intelligence, and a 1/12 scale physical system model. The network architecture comprises all the front end and back end elements that process the various client/server functions, such as receiving ride requests, scheduling pod routes, and relaying information between the main server, stations, and pods. The pod local intelligence encompasses the software and hardware required to provide each individual pod with sufficient onboard intelligence to allow autonomous operation, meaning that after receiving a single message from the server, a pod will complete a trip without additional commands from the server. The 1/12 scale system model provides the testing platform for the network architecture and pod operations, ensuring that the two components merge seamlessly, and also serves as a public demonstration platform during Spartan Superway exhibitions.

During the Fall 2013 semester, the Controls Team's primary concern and goal was to develop Design Requirements and Specifications to provide standards and guide the future development of the Spartan Superway Controls System. With this goal achieved approximately half way through the semester, the team proceeded focus its efforts on developing a system of electronic control units (ECUs) that could govern individual subsystems and connect to a master device via a standardized communication protocol.

At the beginning of the Spring 2014 semester, the confirmation of a Bay Area Maker Faire exhibition lead to an additional goal of having the 1/12 scale model ready for public demonstration. Presenting at Maker Faire created an opportunity to display the efforts of Spartan Superway to potentially thousands of Silicon Valley residents. At this same time, it created an enormous amount of pressure to produce a reliable 1/12 scale model, which ultimately lead to a simplified design that was less representative of the full-scale design the team must simultaneously work towards.

Theoretical System Model

As previously stated, the primary components of the theoretical design at the current stage of development are the network infrastructure and the pod-level intelligence infrastructure. Going forward, the network infrastructure will fall within the Computer Engineering domain, and the pod-level intelligence infrastructure will reside in the Mechanical Engineering domain. The main goal of the 2013-2014 Controls Team was to develop a modular development platform that would facilitate independent development of different subsystems and allow for easier changes in regards to integrating new sensors or actuators to accomplish tasks, rather than redesigning from the ground up to accommodate new hardware.

Electronic Control Unit Network

For developing the Control Systems required for the pods to function autonomously, the Controls Team looked at similar systems of ECUs used in modern automobiles. The electronics and sensors governing vehicle operation are networked using standardized communication protocols, which in the case of automobiles is Controller Area Network (CAN) bus. The hardware developed by the 2013-2013 team physically resembled this system, with the Raspberry Pi as the primary ECU and the PICAXE microcontrollers operating as the subsystem ECUs; however, the limited capabilities of the PICAXEs prevented the possibility of a standardized communication protocol. In addition to facilitating independent subsystem development, a standardized communication protocol would also help the transition to a more robust, industry-level computer system once the Controls System is sufficiently developed.

Switching to the Arduino-compatible platform provided the greatest opportunity to proceed in the necessary direction for a modular development platform. The first key benefit of switching to Arduino-compatible microcontrollers is the familiarity to new Mechanical Engineering students when they join the Spartan Superway team, as it is the platform that is taught throughout Mechatronics-concentration classes.

The second key benefit is that it provides the greatest opportunity of scalability as the performance needs of an individual subsystem increases or decreases, particularly in regards to the number of pins required on a microcontroller to accommodate all the necessary sensors and actuators. Increasing subsystem redundancy will often require the addition of more sensors to monitor or verify the states of actuators. If a greater number of pins are required, or greater processing power, the subsystem can be migrated from an Arduino Uno to a Due or Mega. Scaling down the subsystem could be done by migrating to the Arduino Nano, the Teensy, or even one of the ATTiny chips.

Regardless of the size of the microcontroller used, the Arduino-compatible platform supports numerous communication protocols either directly or through the use of Arduino shields. The most notable protocols are Inter-Integrated Circuit (I^2C), Serial Peripheral Interface (SPI), and CAN bus. Although, CAN bus is more common in industry, it is not as fully-developed within the Arduino platform as the other protocols. The CAN shield for Arduino costs almost twice as much as the Arduino itself as well. The chips used to produce the shield are surface-mount components, so fabricating a shield manually is cost-prohibitive as well. After consulting the Computer Engineering students, SPI was chosen for its speed and reliability. With SPI, the subsystem network would consist of a single master device and numerous slave devices, which transmit data to the master device when requested by the master device.

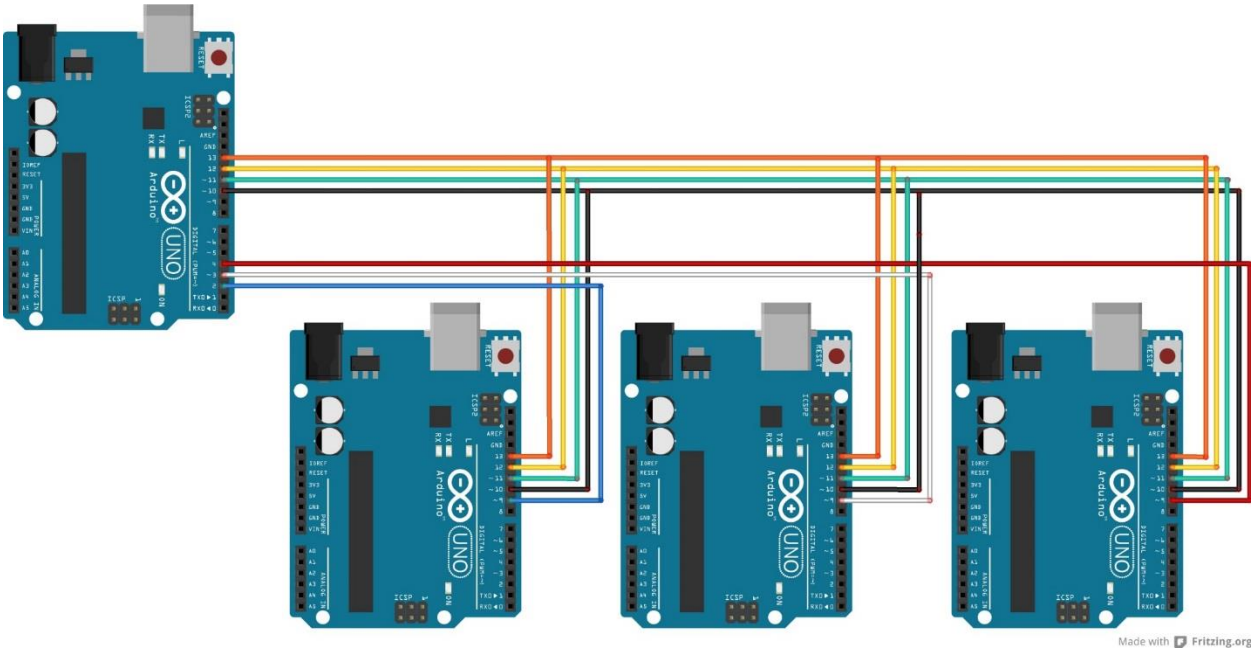


Figure 57. SPI System Network. The master and slave devices share three common pins, and each slave has its own selection pin on the master device.

In developing the Controls System with respect to the 1/12-scale model development platform, the master device will consist of an SJOne microcontroller running FreeRTOS, a free, real-time operating system. The SJOne is a microcontroller with an ARM Cortex-M3 processor that is exclusive to the Computer Engineering department of San José State University, so future Computer Engineering team members will already be familiar with its operation. Using a real-time operating system is critical for the future development of the Controls System so that the system can respond to events in real-time rather than waiting for prior tasks in the sequence to complete. The SJOne microcontroller also has multiple methods of wireless communication between itself and a computer or other SJOne microcontrollers.

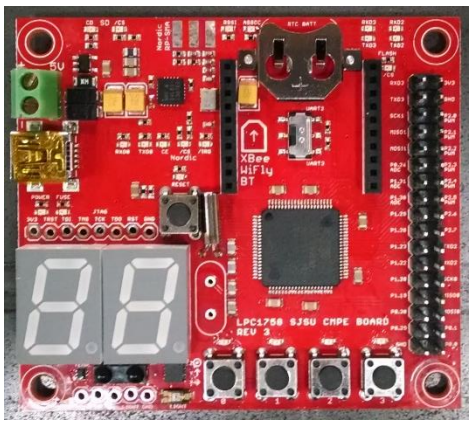


Figure 58. SJOne Microcontroller. The SJOne microcontroller has wireless capability and runs a real-time operating system.

Speed Control

At the end of the Fall 2013 semester, it was determined that the Speed Control System would be a state-space system. However, after further discussion with professor Hemati, the team realized that directly

controlling both motor speed and acceleration was highly improbable given our current level of knowledge and experience in control systems. As a result, it was decided to use of PID controller to directly control the motor speed, hence the pod's speed. The acceleration limits would be met indirectly through tunings of the PID constants. As the motors being used for the scale model remained DC motors, the transfer function of the complete system (controller and plant) remained

$$T(s) = \frac{G_C(s)G_P(s)}{1 + G_C(s)G_P(s)}$$

$$\rightarrow T(s) = \frac{\left(K_d s^2 + K_p s + K_i\right)K}{s[(Js + b)(Ls + R) + K^2] + K\left(K_d s^2 + K_p s + K_i\right)}$$

With: K_p = Proportional control constant

K_i = Integral control constant

K_d = Derivative control constant

J = Rotor moment of inertia

b = Motor viscous friction constant

L = Motor armature inductance

R = Motor armature resistance

K – Motor torque/electromotive force constant (SI units)

These motor characteristics were determined experimentally using the methods described in the Determine Motor Specifications section. Also, changes would be made to the feedback loop of the system. Instead of using a phototransistor to monitor the speed of the motor, an optical quadrature encoder would be attached to the motor to read the speed of the output shaft. The sensing principle remained relatively the same but the integrated encoder would offer better accuracy and easier packaging. To simplify the system, the feedback gain was assumed to be unity. The input to the system was modeled to be a step input

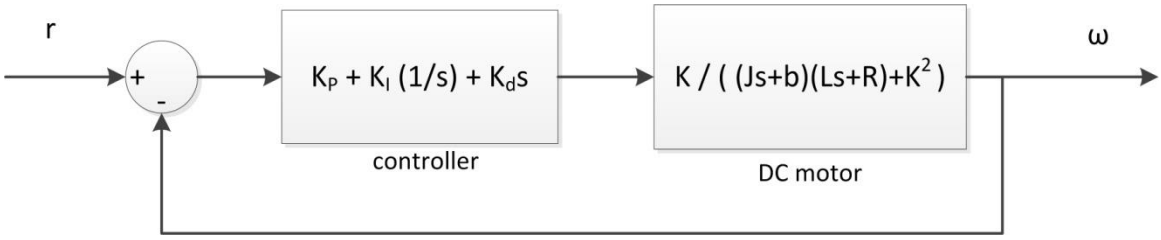


Figure 59. Block Diagram of Speed Control System

The PID controller was designed with two main goals: to bring the motor to the desired speed and to maintain that speed until a new speed limit is received. To achieve these two goals, the steady state response of the system would be the dominating aspect that dictates the design of the controller, specifically the value of the PID constants. On the other hand, to make controlling the acceleration indirectly possible, the controller will be tuned to meet the transient response criteria, specifically the

rise time of the system's step response. Per APM standards, the maximum allowable acceleration of the pod is 0.25g (≈8.043 ft/s^2). The scaled maximum speed along the 1/12th scale model was set to be 2.5 mph (≈3.667 ft/s). Consequently, the time required for the pod to move from 0-2.5mph at 0.25g acceleration was calculated to be approximately 0.456 seconds. This value would be the rise time value for the step response of the system.

Before designing the controller, the natural (uncompensated) step response of the plant (DC motor) was acquired using MATLAB and is shown in Figure 60. The response showed a very fast rise time of less than 0.006 seconds, accompanied by overshoots and a steady state error of 15.6. These response characteristics were obviously unacceptable since the rise time was too quick, which translated to high acceleration rate, and the significantly large steady state error which meant failure to meet desired output.

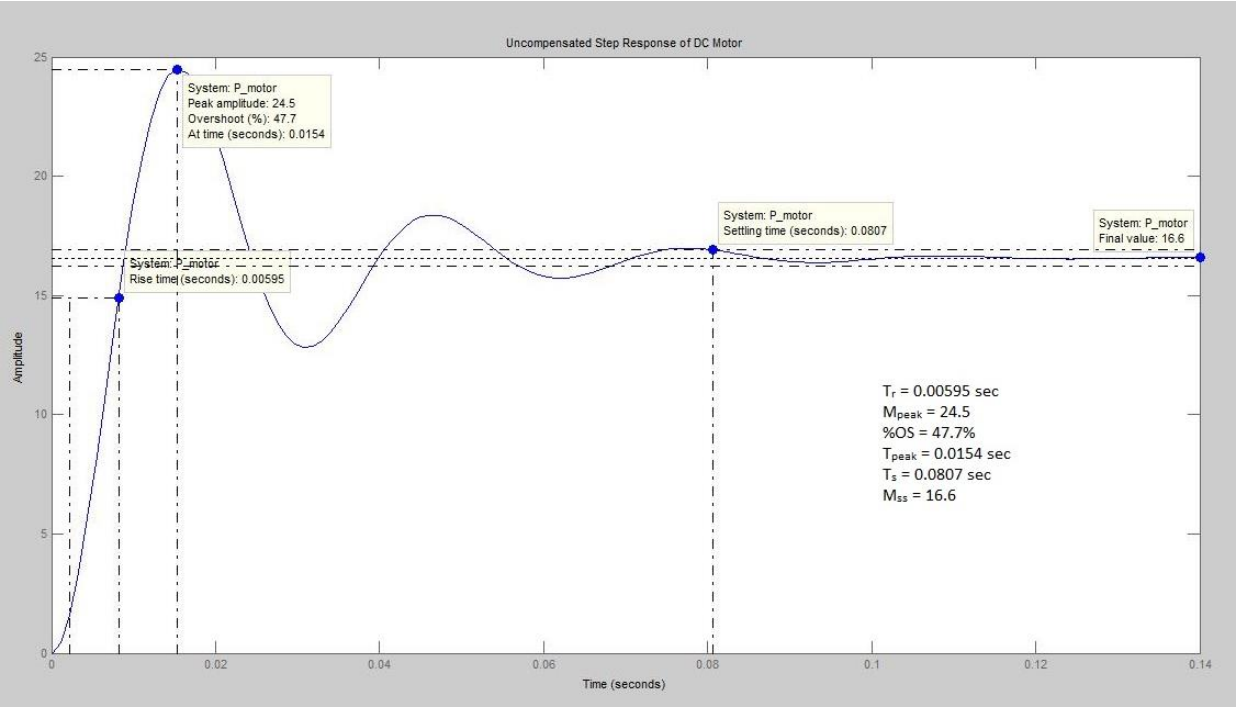


Figure 60. Uncompensated Step Response of DC Motor

Based on the natural response, the design criteria for the controller were set as follow:

- Zero steady state error (achieving and maintaining desired speed)
- No overshoot (providing smooth acceleration)
- Lengthening rise time (limiting maximum acceleration rate to 0.25g)

Using MATLAB's `pidtool()` function, the controller was designed to meet the above performance criteria. The final PID constants were determined to be:

- $K_p = 0$
- $K_i = 0.2892$
- $K_d = 0$

It is worth noticing that both the Proportional and Derivative constants were set to zero. The Derivative term was zero because it was found that having the term in the controller had little effects on the response of the system. Regarding the Proportional term, as the goal of the controller was to reduce the speed of response of the system, the Proportional term, whose effects are to speed up system response, was set to zero. Therefore, the designed controller was effectively an Integral controller which yielded the desired step response shown in Figure 61.

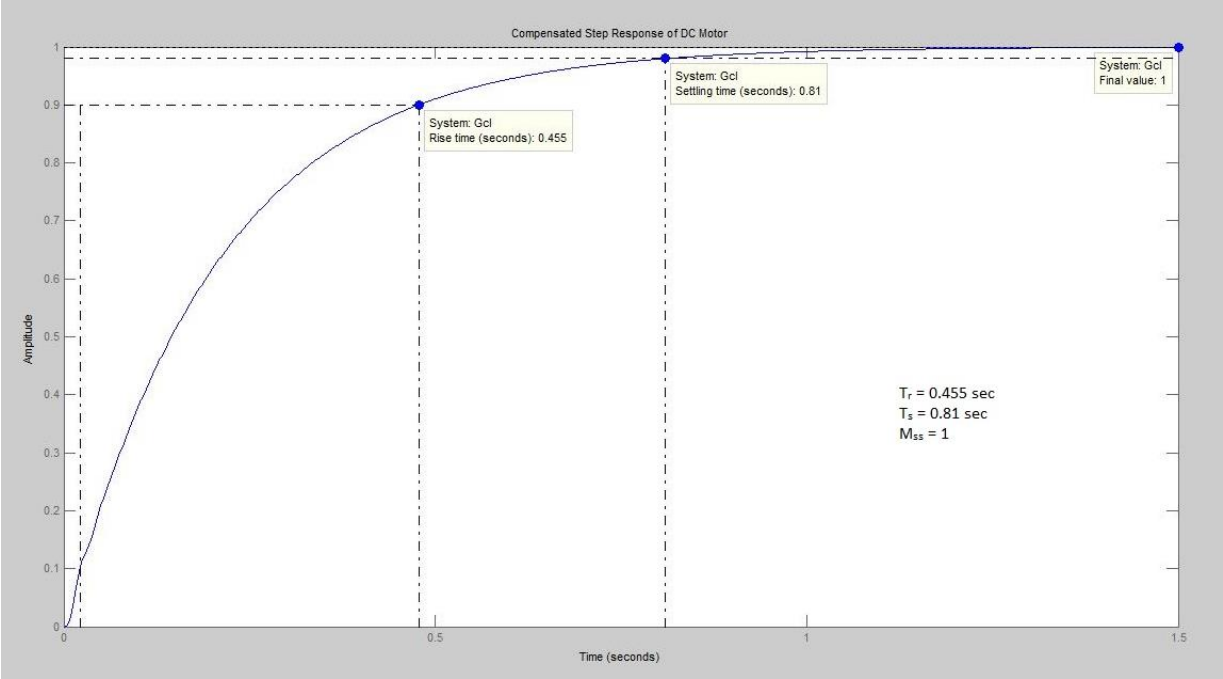


Figure 61. Compensate Step Response of DC Motor

Finally, to ensure that the system was stable, the root locus of the close-loop system was plotted. Figure 62 shows that the system had the tendency to become unstable if the gain increased too much. To ensure that this was not the case, the Bode plots of the system were created in Figure 63. The Bode plots specified a Gain Margin of 25.5 dB and Phase Margin of -180° which indicated a stable closed-loop system.

Unfortunately, time constraints did not allow for complete testing of the PID control code to verify that the motors operate as intended. And thus the Maker Faire model was set to run at a fixed PWM value. Consequently, further testing and refining of the source code included in Appendix R: Arduino PID Speed Control Source Code is required to successfully implement the PID control system.

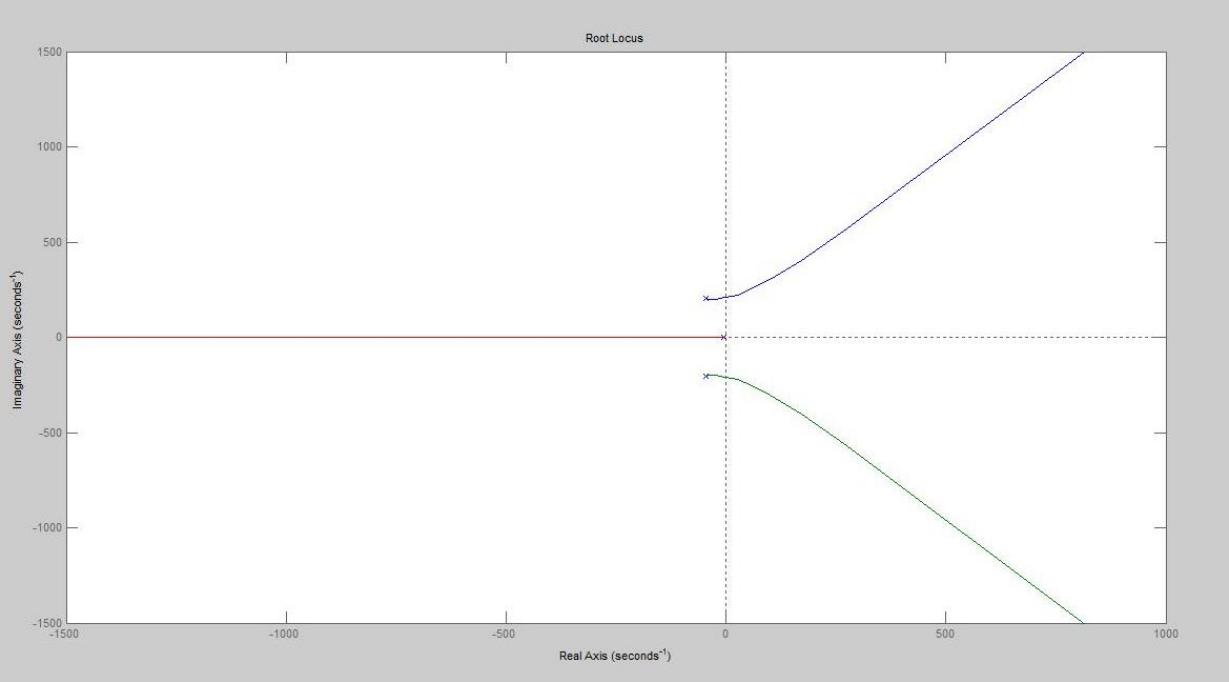


Figure 62. Root Locus of Closed-loop System

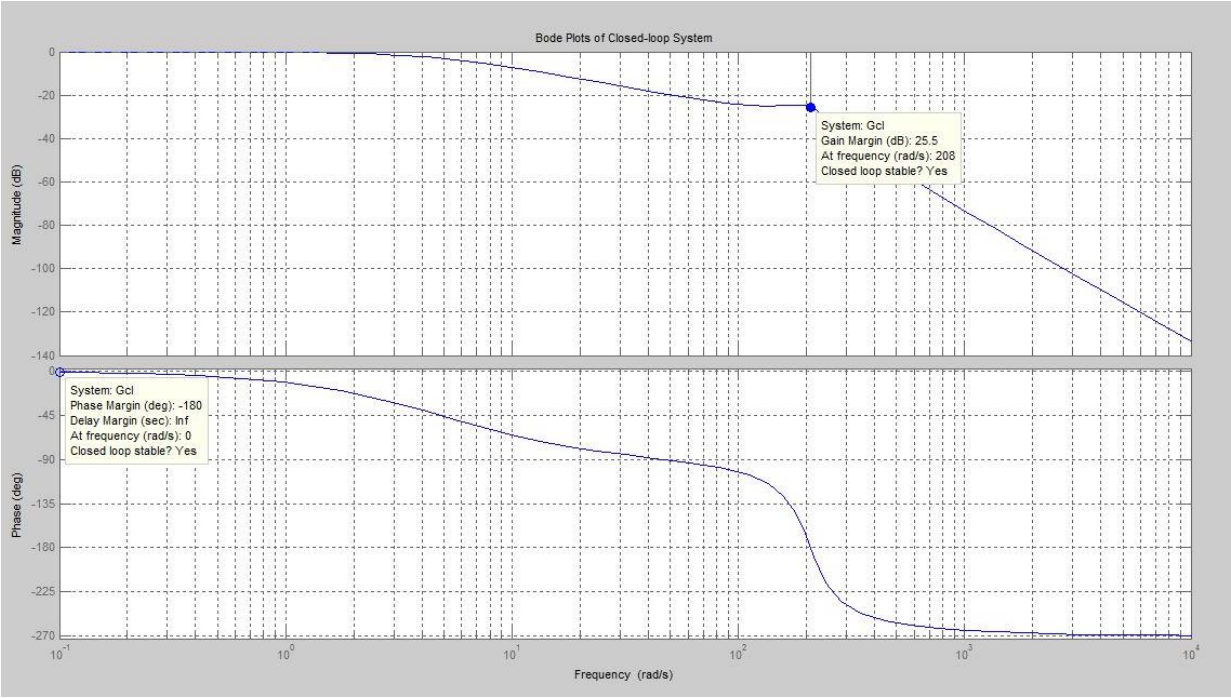


Figure 63. Bode Plots of Close-loop System

Determining motor specifications

For the speed control sub-system, Man determined that despite the motors being so small it would still be a good idea to find all the motor specifications as a precaution. With this in mind Man and Randall used the following guidelines in order to determine the motor characteristics.

Armature Resistance

The armature resistance of the motor can be described as the resistance of the windings inside of the motor. In order to measure this, take the motor and connect it to an ohmmeter and without applying voltage, record the resistance. Take extra precaution however and try turning the rotor to different positions as the resistance may fluctuate. At each measurement record the resistance value and once finished, take the average resistance.

Please note that there is a second method involving locking the rotor and applying a voltage to the motor. While it is effective, it requires you to wait a while until you can get a resistance value. In the meantime however, the coils are heating up and it could possibly burn out your motor. This is especially true when you apply a high voltage to the motor.

Armature Inductance

There are two ways to measure the armature inductance. The first method is to use a LCR measurement device that can explicitly measure the inductance. However, just like the resistance you need to measure the inductance at different rotor angles and take the average value.

The second method requires the use of an oscilloscope and function generator. In order to get the inductance you also need to apply the following equations as described by ctc-control.com.

$$Impedance = \frac{phase\ voltage}{current}$$

$$reactance = \sqrt{impedance^2 - resistance^2}$$

$$Inductance = \frac{reactance}{2\pi x frequency}$$

Where the phase voltage is a low powered AC voltage, the current is a value that needs to be measured. Once the impedance is calculated, enter that value into the reactance equation. The resistance of the equation is the armature resistance that was calculated earlier. In order to find the inductance the frequency was needed in this case Man and Randall set the function generator to have a frequency of 60 Hz.

Motor Constant

In order to measure the motor constant the motor itself needs to be under no load conditions. It is recommended that when calculating this part the input voltage should be relatively low so that the motor doesn't burn out. Monitor the motor until a steady state value for the current is achieved and multiply that value by the armature resistance. At the same time find a way to measure and record the angular velocity. Once finished apply the values to this equation in order to calculate the motor constant.

$$K_b = \frac{V}{\omega_{NL}}$$

The motor constant K can also be calculated theoretically using the specifications (no-load speed and current) provided by the motor vendor/manufacturer. NOTE: there are two different motor constants (electromotive force constant and torque constant), if SI units are used, the two constants should have the same numerical value.

Viscous Friction Constant

In order to calculate the viscous friction constant, once all of the other constants were found Man and Randall entered the values into the following equation.

$$B = \frac{K_T I_a}{\omega}$$

Where K_T is equal to K_b, assuming the measurements are in SI units, Ω is the angular velocity, and I is the measured no load current.

Measured Motor Characteristics

Given the many disadvantages of the motors being used in the Fall semester, the team decided to use a different motor for the scale model bogie. The newly chosen motor is a 30:1 Micro Metal Gearmotor from Pololu that is rated at 6V. The new motor offers relatively impressive performance in terms of torque and rpm that come in a small package. The motor also offer the versatility of attaching an optical quadrature encoder to its rear shaft to measure rotational velocity as well as direction. Being a brushed DC motor, the transfer function for the new motor remains the same. As a result, Randall and Man set out to measure the motor characteristics experimentally and were able to determine the needed motor constants.

As the wheel of the bogie has not been changed since last semester's design, the moment of inertia term (J) remains to be $5.051*10^{-7}$ m^4.

Through experimental methods, the motor's no-load speed (ω_{NL}) at 6.01 V is approximately 983 rpm. And the no-load current was measured to be 0.069 A at 6.01V.

$$\omega_{NL}=983 \text{ rev/min}*2\pi 1 \text{ rev}*1 \text{ min}/60 \text{ sec}=102.94 \text{ rad/sec}$$

The electromotive force constant of the motor can then be calculated

$$\omega_{NL}=VK_e$$

$$\rightarrow K_e=V\omega_{NL}=6.01 \text{ V}102.94 \text{ rad/sec}=0.0584 \text{ Vrad/sec}$$

As K_e and K_T are the same numerically in SI units, the motor torque constant is

$$K_T=K_e=0.0584 \text{ NmA}$$

$$\rightarrow K=0.0584$$

Based on these values, the motor viscous friction constant can be calculated from the equation

$$K_T Ia=b\omega_{NL}$$

$$\rightarrow b=KTIa\omega NL=(0.0584 \text{ NmA})*(0.069 \text{ A})102.94 \text{ rad/sec}=3.913*10\text{-}5\text{N.m.sec}$$

The armature winding resistance was also measured to be about 2.895 Ω.

The motor inductance was indirectly measured (using CTC-Control manual) to be 0.161 H.

Navigation

The Navigation subsystem combines the previously developed Position Tracking and Guideway Switching Control subsystems into a single subsystem. As a whole, this subsystem is one of the most important monitoring aspects of the Superway network at the pod level, particularly in terms of system safety. The ultimate goal of the subsystem is to guarantee that each individual pod recognizes its real-time position on the Superway network at all times, thus minimizing the possibilities of lost pods, mitigating collisions, and ensuring proper network navigation.

Each pod in the Superway system maintains its own Navigation subsystem to sense the location of the pod along the guideway network on a real-time basis. Whereas the driver/conductor on a conventional transit vehicle recognizes the vehicle's current physical location, Superway pods are fully automated and thus require a sensing mechanism to determine each pod's location at all times. The importance of this subsystem for the entire Superway network is that it allows the pod to know its exact location, even if it is at the wrong location. Standard procedure for the Navigation system will consist of monitoring the pod's location on the Superway guideway network, reporting the position to the Master Device when queried, and manipulating the guideway switch if the pod is approaching a junction and requires a path change.

The 2013-2014 Controls team designed the system for the 1/12 scale model to accommodate the track design and position tracking method developed by the 2012-2013 Controls team. The subsystem hardware includes an Arduino microcontroller as the plant and two reflective object sensors equivalent to the sensor implemented in the Speed Control subsystem. The sensors are mounted on two sides of the model pod roof, oriented towards the track. Non-reflective, felt markers were placed on both sides of the track at key points before, within, and after junctions. The felt markers provide sufficient contrast against the highly-reflective, aluminum tape used for the grounding rail of the track power transmission system. When the reflective object sensors detect change in reflected light as the pod travels across the marker, the microcontroller will determine the position of the pod along the track by counting each signal change on each side of the track. The markers placed along the inner edge of the track notify the pod when it is entering or leaving a junction. The markers placed on the outer edge of the track manipulate the solenoids used in the switching mechanism so that they are not active for too long, preventing damage to the solenoids due to overheating. They are also used to determine when the pod is pulling into a station.

The pod's route is determined on the master device level using Djikstra's algorithm, an algorithm used to determine the shortest route between two points in node based network. In this context, each junction serves as a node. This particular application algorithm uses an un-weighted track system, meaning that all distances between nodes will be treated as equal. This prevents the pod from taking shorter routes by cutting through offline station segments. The Navigation subsystem manipulates the switch at each junction based on the route determined by the path-finding algorithm. Using this algorithm allows the Navigation subsystem to scale to any track layout, rather than limiting it to the current 1/12-scale track layout.

This setup, which utilizes reflective object sensors and felt markers as the position tracking mechanism, was specifically designed for the 1/12 scale track, and provides very low resolution in regards to the pods absolute position within the track network. The full-scale Navigation subsystem will likely require a more robust sensing mechanism such as a global positioning system (GPS) to determine pod location. As it is, Figure 64 shows the 1/12 scale model subsystem schematic with the specified hardware.

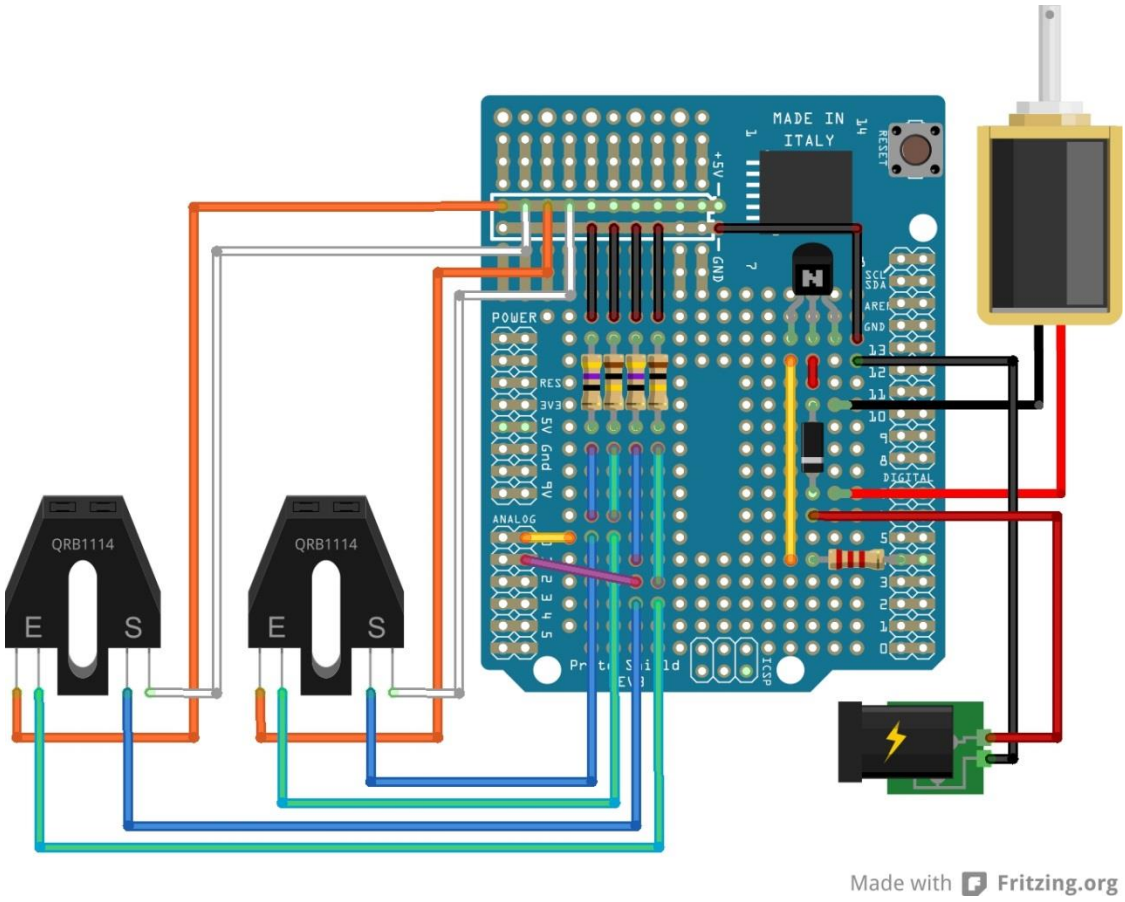


Figure 64. Navigation Subsystem. The 1/12-sclae model uses reflective object sensors and a solenoid to navigate the track network.

In addition to the to the reflective object sensors for reading the track markers, future iterations of this subsystem should include one or more sensors for detecting the position of switching mechanism (i.e. left or right) in order to verify whether or not it is actually in the correct position. The need for this sensor resulted from a discussion with Gene Nishinaga of Transit Control Solutions, a veteran of the automated transit industry and one of the developers of the controls system for Bay Area Rapid Transit (BART). To summarize the discussion, having the switching mechanism in the wrong state is less important than not knowing the switching mechanism is in the wrong state. Knowing that the switching mechanism is in the wrong state allows for the ability to correct the state, whereas being in the wrong state, and also not being able to verify the state, will result in the Superway rider ending up in the wrong location, without notifying the network.

Object Detection

From the Fall 2013 semester, the Controls Team continued to use the same ultrasonic sensors that the 2012-2013 group used. In the Spring 2014 semester, the team was able integrate the sketch needed to utilize the sensor into the Superway library.

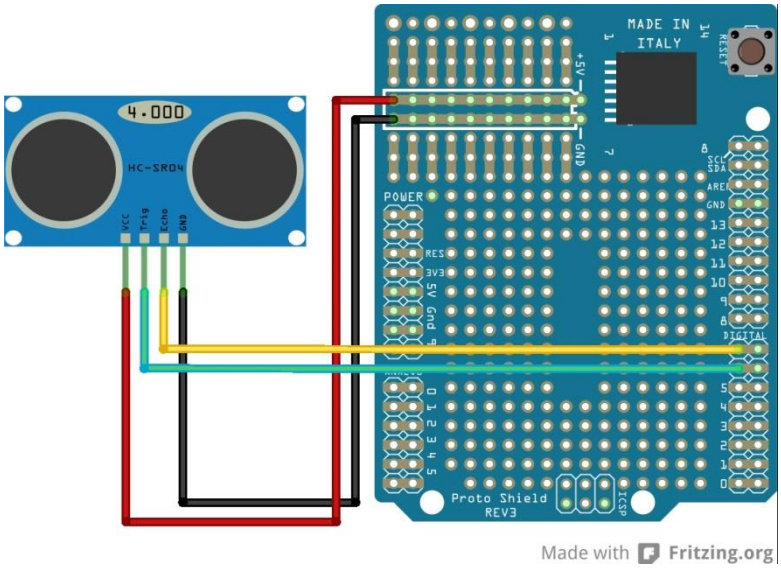


Figure 65. Object Detection Subsystem. The ultrasonic sensor is used detect whether or not there is a path obstruction and the distance to any path obstructions.

The object detection subsystem was unable to be utilized in time for Maker's Faire. This was in part due to the priority given to the navigation system since one of the main requirements is that the Pod will always know where it is on the track. Due to the somewhat buggy nature of the navigation subsystem, due to the inconsistent lighting, it was also decided to only have one Pod on the track and since it would always be constrained to a track it would not need to make use of the object detection subsystem.

Maker Faire Model

When the team decided to scale down for the Maker Faire display model, the two most critical areas to inspect were the number of microcontrollers used, and the method of providing the pod with power.

Due to the limited time frame of a single semester, the team was not confidant that four Arduinos integrated with the SJOne board could be fully tested to the point that it would yield predictable and reliable results. For this reason, all the functions performed by separate subsystems were consolidated to a single Arduino Uno.

Knowing that the 1/12-scale track would have to be disassembled and transported to Maker Faire, the team chose not to develop a connection to the power rails of the track as developed by the Electrical Engineering team the previous semester. The two day setup time would not be a sufficient amount of time to ensure that the connections between track segments would work through the entirety of the weekend. Instead, the team opted to run solely on battery power.

Additionally, the motor control system developed for the DC motors was not implemented. This was due to the fact that the team chose to operate the pod at a lower overall speed, as opposed to the 1/12-scale speed of 2.5 miles per hour, to reduce the risk of damage to the pod or collisions with people

attempting to test the object detection. The algorithm governing route navigation was not utilized either due to reasons relating to SPI integration with the SJOne board.

Power

After the announcement that Spartan Superway was going to Maker Faire, the team decided to abandon plan to acquire power for the pod from the guideway and instead switched to using on-board power source for portability and reliability issues. Since the chosen DC motors were rated at 6V, power would come from four AA batteries connected in series that supply 6V DC. This setup was chosen to reduce the number of components needed to power different devices with different voltages. However, since the sensors being used on the pod can only operate on less than 5V, a voltage regulator was needed to reduce the 6V output from the battery pack down to 5V using an L4940V5 1.5A Voltage Regulator. The circuit for the regulator was constructed based on the schematic given in the provided datasheet that is shown in Figure 66 where the 6V input is applied across the 0.1 µF capacitor and 5V is output across the 22µF capacitor. This 5V supply was then used to power the Arduino, along with the infrared reflective sensors and the ultrasonic distance sensor. Realizing the limited space inside the cabin frame of the model, the regulator circuit was put on an Arduino Proto Shield and the resulting circuit is similar to that shown in Figure 67 where the shield would also be used as a compact physical platform to mount other components. Furthermore, anticipating that the motors and solenoids would draw significant currents, the team chose to have two separate but identical battery packs. One pack would be used to power the microcontrollers (Arduino and SJOne boards) and sensors through the regulator while the second pack would be used exclusively to power the DC motors and solenoids.

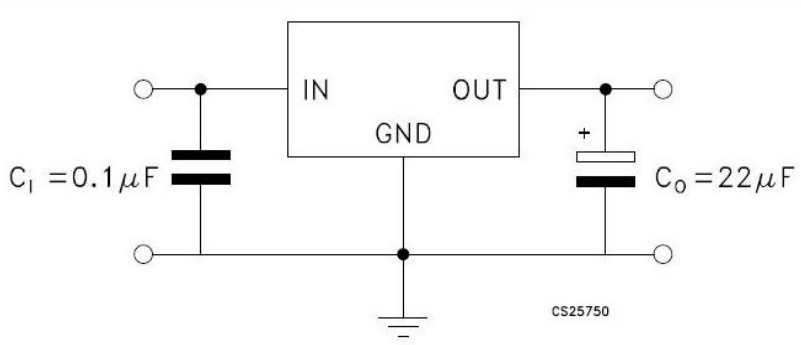


Figure 66. Circuit Diagram for L4940V5 Voltage Regulator (Source: STMircoelectronics)

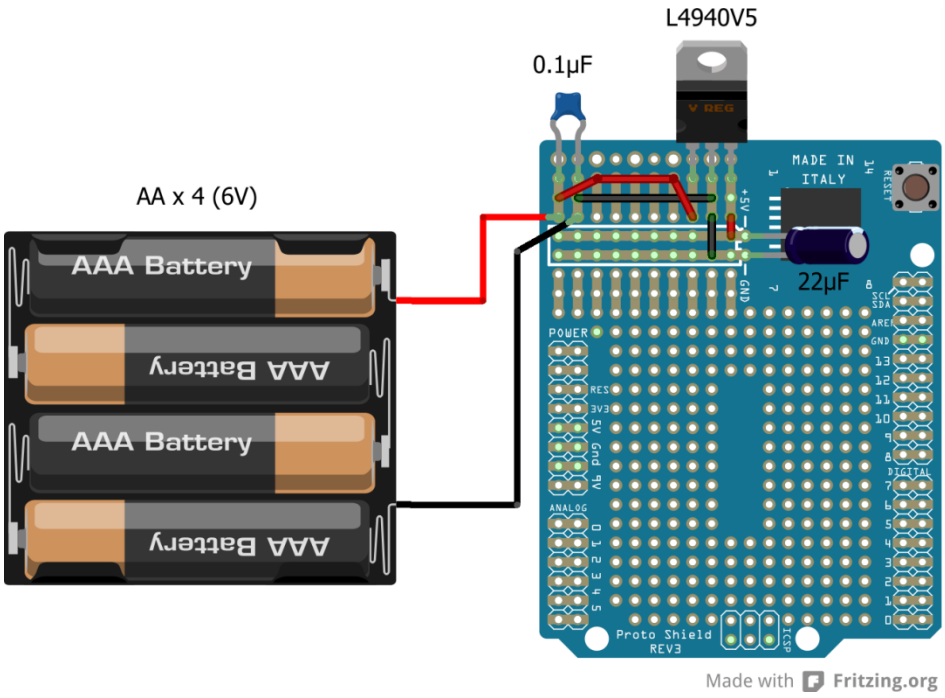


Figure 67. Circuit of Voltage Regulator on Arduino Proto Shield Converting 6V from 4-AA Batteries to 5V

Sensors

The design of the Maker Faire model incorporates two types of sensor for two different subsystems, navigation and distance control/object detection. The navigation subsystem utilizes two Optek OPB704WZ infrared reflective object sensors pointing upward from the top of the cabin frame to the underside of the guideway. Each sensor is rated at 2V and thus was connected to power from the Arduino through a pair of 470Ω and 100kΩ resistors that are mounted on the Proto Shield. Signal output by the sensors would be read by the Arduino through its analog pins.

The object detection subsystem uses an HC-SR04 ultrasonic distance sensor mounted in front of the cabin frame. Since the sensor is rated at 5V, it was connected directly to the output voltage from the Arduino while two outputs were connected to two of the Arduino's digital pins as shown in Figure 68.

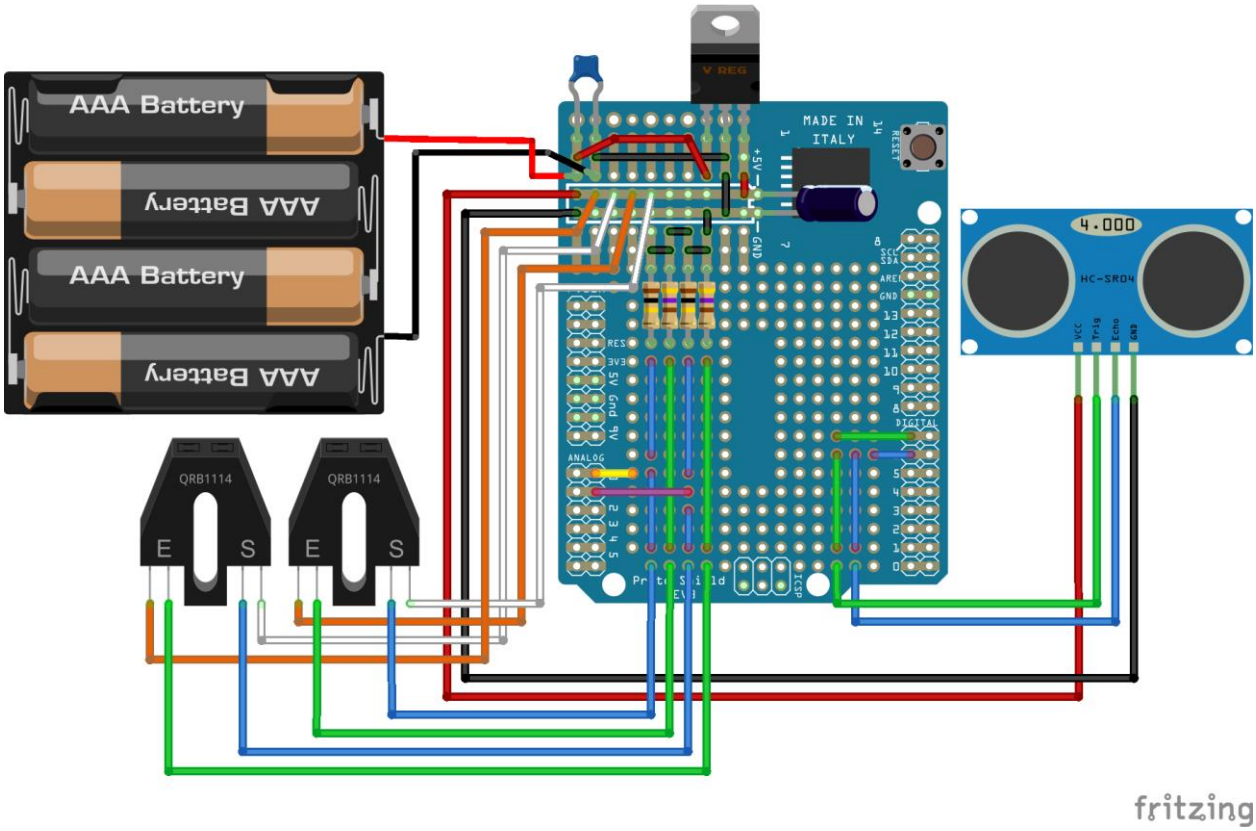


Figure 68. Circuit of Voltage Regulator and Sensors on Arduino Proto Shield

Switching Solenoids

To activate the switching arms that are located on top of the bogie, two 5V solenoid were used. Each solenoid is rated at 5V and 1.2W continuous and actuated through a TIP102 NPN Darlington 100V 8A Transistor with transient-response-protection diodes. The solenoids would be powered by the second battery pack. It is worth noticing that although the solenoids were rated at 5V, the team decided to run them directly on 6V from the battery pack. The reason for such decision was that through testing, it was observed that the switching arms were heavy enough to prevent the solenoid from fully extend when supplied with 5V. The solenoids had to be fully extended because the switching arms were designed to reach the top guiding midsection of the track only when the solenoids are fully pushed out.

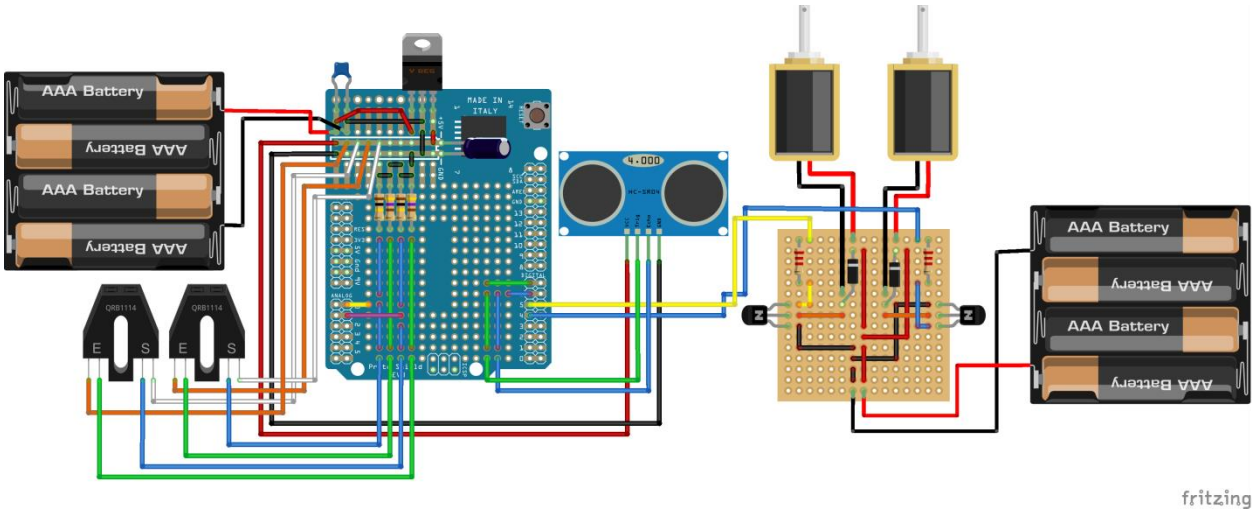


Figure 69. Circuit of Voltage Regulator with Sensors and Switching Solenoids with Separate 4-AA Battery Packs

DC Motors

The DC motors from the Fall semester were deemed to be too weak while taking up too much space inside the bogie and thus new motors were chosen. The new motors are 30:1 Micro Metal Gearmotor from Pololu. Each motor is rated at 6V and 120mA free-run current with a no-load output shaft velocity of 1000 rpm and 9 oz.in of stall torque. The motor has an extended back shaft that could be used to monitor shaft velocity using an integrated optical quadrature encoder. The motors share the same 6V input with the solenoids from the second battery pack. Driving the motors would be a Pololu DRV8833 Dual Motor Driver Carrier, which is represented in Figure 70 by an H-bridge placeholder. Similarly to the previous setup, there would be two motors in each bogie driving the diagonally opposing wheels.

Heat dissipation from the actuators as well as the transistors and regulator was a concern during the design process. However, the open design of the cabin frame would help effectively dissipate any generated heat.

Besides the Arduino microcontroller that controls the electronics onboard, a second microcontroller, the SJOne board was also incorporated into the system. The SJOne would provide wireless communication between the pod and the computer. The board was connected to the Arduino through SPI and was powered from the same 5V source from the voltage regulator.

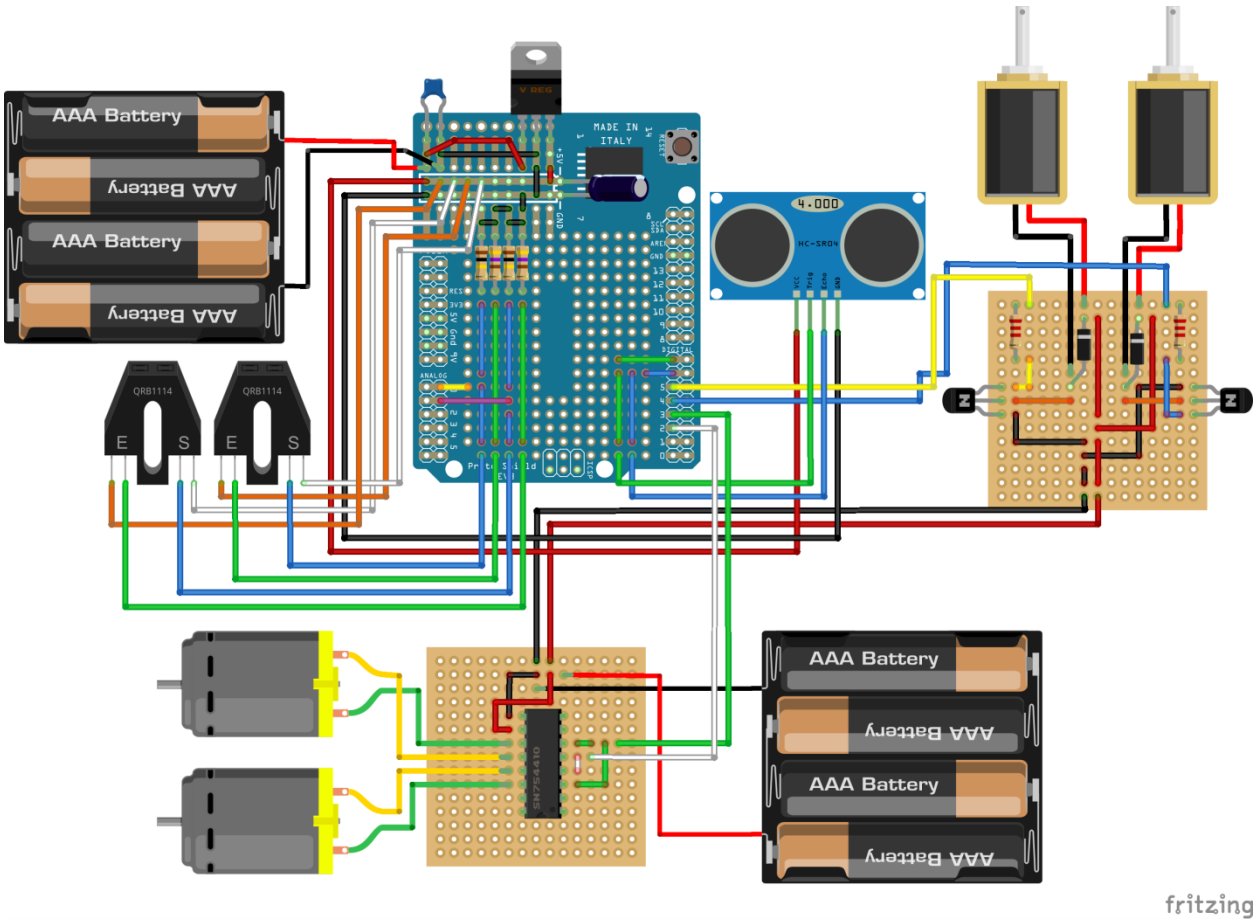


Figure 70. Circuit of Complete Pod Electronics Connected to Arduino Proto Shield

Chassis

The 2012-2013 Controls team produced a 1/12-scale bogie and pod chassis that were fabricated using a 3D printer. At the beginning of the 2013-2014 academic year, access to a 3D printer was no longer reliable due to the growing popularity of the emerging technology and TechShop's promotion of the 3D printing classes. Limited access to the required tools, and the overall printing time of approximately 54 hours, resulted in 3D printing being a prohibitive and inefficient method of fabricating the necessary parts. The best alternative was to use the computer numerical controlled (CNC) laser-cutter located in the Mechatronics Lab of the Engineering building to cut parts from 1/8" acrylic sheets leftover from the 1/12-scale track. New sheets could then be purchased from any of the TAP Plastics in Bay Area.

Rather than permanently glue each piece together with acrylic cement, the acrylic pieces were designed to connect using M3 machine screws and nuts, following the technique commonly known as interlocking T-bolt construction. The technique is detailed in an Instructables guide produced by Oomlout, a United Kingdom based organization specializing in Arduino-based products and projects (Oomlout, 2014).

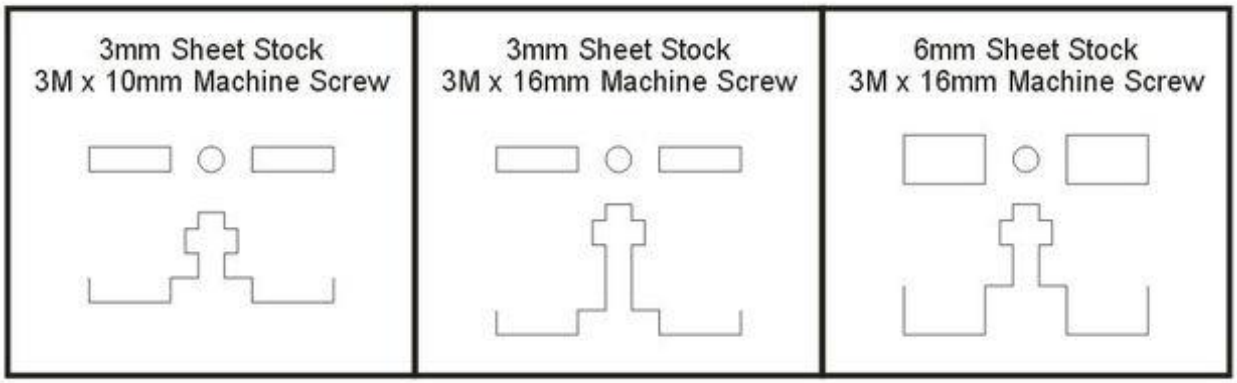


Figure 71. Interlocking T-bolt Construction. Using this technique on acrylic pieces allows temporary assembly and simple disassembly. (Oomlout, 2014)

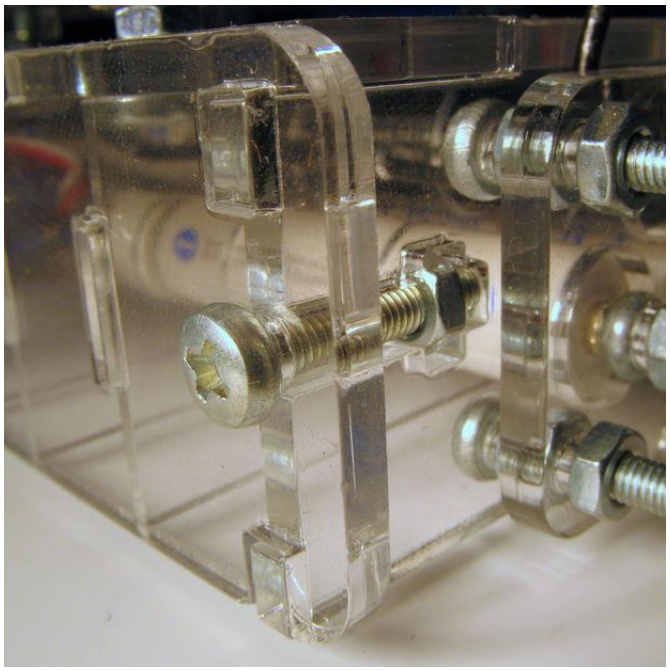


Figure 72. Joined Acrylic Components. Interlocking T-bolt construction provides a tight fight and prevents the nut from freely spinning (Oomlout, 2014).

The acrylic chassis required several iterations with each iteration focused around a particular weakness of the previous design, which generally consisted of structural weaknesses based on the brittleness of acrylic or unforeseen difficulties in assembling the chassis. Once completed, the entire chassis could be cut from a single sheet of acrylic measuring approximately 17” by 17” in just over an hour by efficiently orienting the individual components of the chassis (see #include <SPI.h>

```
int buf [3];
volatile byte pos;
volatile boolean process_it;
int gasPedal = 0;
unsigned long timerStop;
```

```
void setup (void)
{
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);
  pinMode(2, OUTPUT);
  digitalWrite(2, LOW);
  Serial.begin (9600);   // debugging
  // have to send on master in, *slave out*
  pinMode(MISO, OUTPUT);

  // turn on SPI in slave mode
  SPCR |= _BV(SPE);

  // get ready for an interrupt
  pos = 0;   // buffer empty
  process_it = false;

  // now turn on interrupts
  SPI.attachInterrupt();

}  // end of setup


// SPI interrupt routine
ISR (SPI_STC_vect)
{
byte c = SPDR;  // grab byte from SPI Data Register
Serial.print("ISR\n");
  // add to buffer if room
  if (pos < sizeof buf)
    {
    buf [pos++] = c;

    // example: newline means time to process buffer
    if (c == 0x00)
      process_it = true;

    }  // end of room available

}  // end of interrupt routine SPI_STC_vect

// main loop - wait for flag set in interrupt routine
void loop (void)
{
  if (process_it)
    {

      gasPedal = buf[0];
      Serial.println(gasPedal);
      timerStop = millis();
      /*
      for(int i=0; i<(pos-1); i++)
      {
        Serial.print (buf[i]);
        Serial.print(" ");
```

```
      }
      */
      pos = 0;
      process_it = false;
    }  // end of flag set
  if( gasPedal )
  {
    //Serial.println("Gas pedal is true!");
    analogWrite(3, 70);
    //digitalWrite(3, HIGH);
  }
  if( !gasPedal )
  {
    digitalWrite(3, LOW);
  }
//   Serial.print("HI\n");
}  // end of loop
```

Appendix V: 1/12-Scale Part Drawings). This also included spare pieces of particularly small components, like screw spacers. The final design proved to be particularly durable throughout the duration of Maker Faire, even surviving a direct broadside hit from a beach ball, after which it continued with its route.

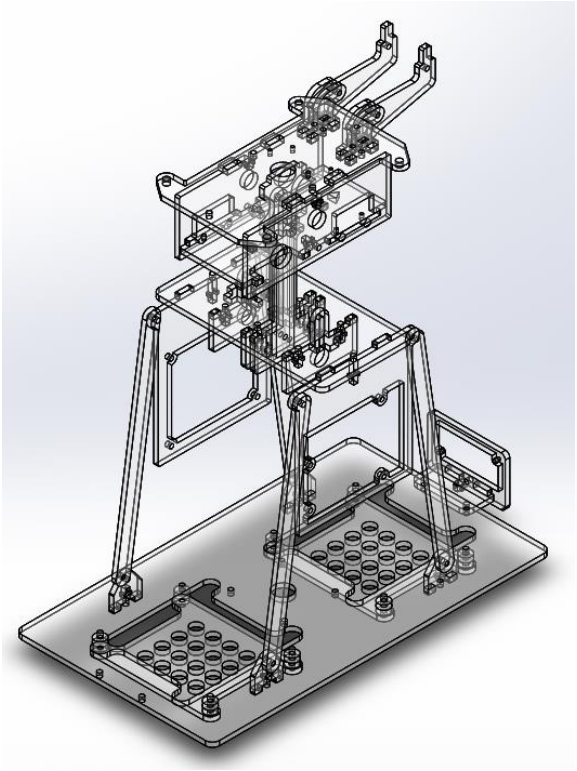


Figure 73. 1/12-scale CAD Assembly. The chassis is constructed from approximately 15 unique parts.

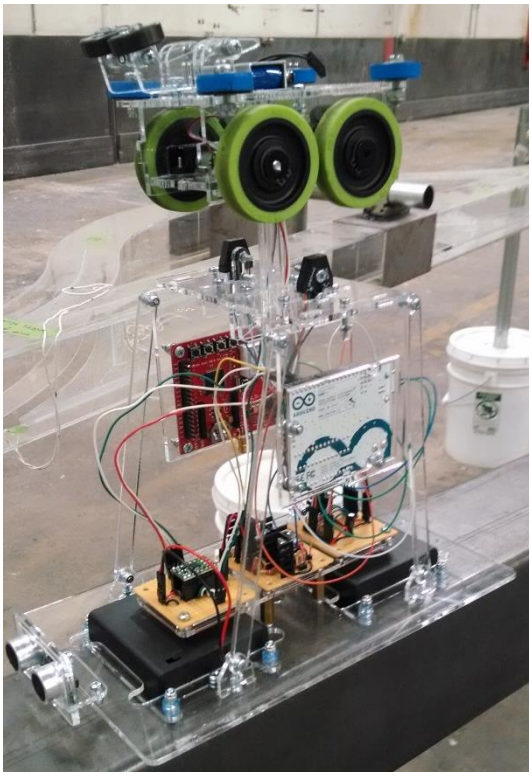


Figure 74. 1/12-scale Chassis. Off-the-shelf components were used whenever possible.

Programming

Once the coding process began at the start of the semester, the team chose to develop a versatile Superway Library that would allow for the subsequently developed Arduino sketches to be simple and easy to read. The core of this process was to create a "Pod" class such that within an Arduino sketch, the programmer could create and manipulate a "Pod" object with commands specific to the subsystem, such as Speed Control, Navigation, and Object Detection.

Developing the class for the Superway Library began with creating the header file, in which the necessary private variables and functions with appropriate pseudocode were listed. The private variables primarily represent characteristics of the pod, including speed, location, destination, and the distance to a detected object. Some additional variables were implemented for state tracking and timing purposes, such the current and previous readings of the reflective object sensors, and the time since the last encoder signal was received. For each private variable included within the class, a corresponding read function, and in some cases write function, were also necessary. Private variables within a class are only able to be manipulated or accessed through their respective class, therefore necessitating the read and write functions. Because the Superway Library was developed specifically within the context of the Arduino-compatible platform, the development adhered closely to the Arduino API Style Guide located on the Arduino website (Arduino, 2014).

Once the fundamental needs of the Pod class were outlined with respective pseudocode, further development consisted of unit testing functions within the context of a simple Arduino sketch. For the read and write functions, this was a relatively straightforward process. For more complicated functions, such as the triggering of the reflective object sensors to indicate marker readings, the Arduino sketch was used to develop the code, and upon successful completion, the code was transferred to the sketch. In most cases, this was a generally successful practice; however, insufficiently testing the code following the transfer led to significant problems later on in the coding process.

The Object Detection code governing the use of the ultrasonic sensor was begun in the early stages of the coding process, prior to the confirmation of the Maker Faire exhibition. This code was developed according to the ideal that for optimal performance and safety, all code within the Controls System must be non-blocking (i.e. no use of the delay() function). Although the ultrasonic sensor code functioned as required as a standalone Arduino sketch, once it was transferred to the Pod class as a function, it no longer worked. Because the Object Detection subsystem was less important than other subsystems for the sake of Maker Faire, primarily Navigation, the non-blocking code developed was substituted for the NewPing Arduino Library developed by Tim Eckel (Eckel, 2014). Although the library uses blocking code methods, the time delays were in the region of microseconds as opposed to milliseconds, so the team decided it was acceptable for implementing at Maker Faire if the original code was not fully developed and integrated in time for the event.

A similar problem arose in developing the Navigation subsystem, specifically in regards to simplifying the process of reading the triggering events for the markers placed along the track. The initial use of state tracking alone was sufficient for the test sketch, but like the ultrasonic sensor code, it did not transfer to the Pod class. Once testing sessions began lasting into the evening, and the ambient light within the testing environment began to decrease more rapidly, state tracking alone was no longer sufficient. A short non-blocking timer was added to the code to prevent the pod from reading more than one marker within the span of 0.25 seconds. This time interval was determined based on the speed that the pod would run during the Maker Faire exhibition.

The Controls Team determined the operating speed of the pod based on two characteristics: the minimum possible speed, and the response time of the switching mechanism solenoids. The minimum speed of the pod was set as the lowest possible PWM signal to create instantaneous motion of the pod. Once the pod started moving, its speed would naturally increase slightly as it overcame friction and gained momentum. This speed of the pod was set slightly higher for increased momentum and smoother travel across the track gaps at junctions. The speed ceiling was determined based on the amount of time required for the switches to fully elevate after reading a track marker. After choosing the ideal travel speed, the track markers were placed to ensure the pod would stop at defined stations and that the switches would fully elevate before reaching the guiding rails. The final Arduino sketch used in the Maker Faire exhibition, as well as the unfinished Pod class are detailed in

Appendix S: Arduino Maker Faire Sketch Maker Faire sketch and /*
Spartan Superway 1/12 Scale Model Maker Faire Demonstration Sketch

2013-2014 ENGR 195D / Spartan Superway

Created by: Cory Ostermann, Man Ho, Randall Morioka, Eriberto Velzquez, and Anthony Vo

This sketch provides functionality for a user to send new destinations to a pod via an SJOne microcontroller connected to a laptop.

*/

```
#include "SPI.h"
#include "NewPing.h"

// Starting Location Variables:
#define START_NODE  0
#define START_LINE  0
#define START_TICK  1
#define START_DESTINATION -1

// Pin definitions:
#define TRACK_OFFLINE A0  // The pin attached to the Offline reflective
object sensor
#define TRACK_ONLINE  A1  // The pin attached to the Online reflective object
sensor
#define SWITCH_OFFLINE 4  // The pin attached to solenoid actuating the
Offline Switch
#define SWITCH_ONLINE  5  // The pin attached to solenoid actuating the
Online Switch
#define MOTOR_A1  3
#define MOTOR_A2  2
#define TRIG  6
#define ECHO  7

// Miscellaneous definitions:
#define ONLINE   1
#define OFFLINE  0
#define THRESHOLD_ONLINE  100    // The maximum amount of light reflected
from the markers to trigger the sensors
#define THRESHOLD_OFFLINE 100
#define BOUNCE_TIME  250  // The time delay to prevent multiple readings
#define NO_DESTINATION  -1   // The value for when a pod does not have a
specified destination
#define MAX_RANGE 20      // Maximum detection range (cm) at which the pod
will begin to slow
#define MIN_RANGE 10      // Distance (cm) at which the pod will perform a
active braking
#define MAX_DUTY  65      // Maximum allowable motor duty cycle
#define MIN_DUTY  60      // Minimum duty cycle for motion
#define BRAKE  1          // State tracker for active braking
#define BRAKE_TIME  1000  // The timer that determines when the motors go
from active braking to coasting
#define STATION_1  0
#define STATION_2  2
#define STATION_3  6
```

```
// SPI Variables:
int buf [5];
volatile byte pos;
volatile boolean process_it;

// Navigation variables:
int readingOffline, readingOnline;  // Stores the reflective object sensor
readings
int sensorOnlineNew, sensorOnlineStored, sensorOfflineNew,
sensorOfflineStored;  // Sensor state tracking variables
unsigned long timerOnline, timerOffline;  // Timers to prevent marker
bouncing
int destination;
int locationNode, locationTick, locationLine;  // locationNode tracks the
section of track
                                                // locationTick tracks the
marker readings
                                                // locationLine tracks whether
the pod is Online or Offline
int locationNodePrevious;
// Object Detection Variables:
NewPing sonar(TRIG, ECHO, MAX_RANGE);
int range;
// Speed Control Variables:
int dutyCycle;
int activeBrake;
unsigned long timerBrake;

void setup()
{
  // Setup pin modes:
  pinMode(TRACK_ONLINE, INPUT);
  pinMode(TRACK_OFFLINE, INPUT);
  pinMode(SWITCH_ONLINE, OUTPUT);
  digitalWrite(SWITCH_ONLINE, LOW);
  pinMode(SWITCH_OFFLINE, OUTPUT);
  digitalWrite(SWITCH_OFFLINE, LOW);
  // Initialize the Serial Monitor
  Serial.begin(9600);
  // Initialize variables:
  locationNode = START_NODE;
  locationTick = START_TICK;
  locationLine = START_LINE;
  destination = destinationWrite(START_DESTINATION);
  range = 0;          // No objects detected
  activeBrake = 0;  // Turn off the brake
  // SPI related initialization:
  pinMode(MISO, OUTPUT);
  // turn on SPI in slave mode
  SPCR |= _BV(SPE);
  // get ready for an interrupt
  pos = 0;   // buffer empty
  process_it = false;
  // now turn on interrupts
  SPI.attachInterrupt();
```

```
  printSnapshot();
  // ****************************************************
  //delay(5000);
  // ****************************************************

}  // end setup

void loop()
{
  /*
  SPI COMMUNICATION
  */
  if (process_it)
    {
      if( -1 == destination )
      {
        switch( buf[0] )
        {
          case 1:
            destination = destinationWrite(buf[1]);
            locationNode = destinationWrite(buf[2]);
            locationTick = 1;
            locationLine = 0;
            break;
          case 2:
            destination = destinationWrite(buf[1]);
            break;
          case 111:
            locationNode = buf[1];
            locationTick = buf[2];
            locationLine = buf[3];
            break;
          default:
            break;
        }  // end buf[0] switch
        printSnapshot();
        //pos = 0;
      }
      /*
      for(int i=0; i<(pos-1); i++)
      {
        Serial.print (buf[i]);
        Serial.print(" ");
      }
      */
      pos = 0;
      process_it = false;
    }  // end of flag set
  /*
  NAVIGATION
  */
  // Take new sensor readings:
  readingOffline = analogRead(TRACK_OFFLINE);
  readingOnline = analogRead(TRACK_ONLINE);

  // Determine the new sensor states ( 1 = on marker, 0 = off marker )
  if( readingOnline < THRESHOLD_ONLINE )
```

```
  {
    sensorOnlineNew = 1;
  } else {
    sensorOnlineNew = 0;
  }
  if( readingOffline < THRESHOLD_OFFLINE )
  {
    sensorOfflineNew = 1;
  } else {
    sensorOfflineNew = 0;
  }
  // Compare and possibly reset the Stored values based off of the New
values.
  // Offline Update:
  if( sensorOfflineStored < sensorOfflineNew )  // The leading edge condition
  {
    sensorOfflineStored = sensorOfflineNew;
  }
  else if( sensorOfflineStored > sensorOfflineNew )  // The trailing edge
condition
  {
    sensorOfflineStored = sensorOfflineNew;
    if( millis() - timerOffline > BOUNCE_TIME )  // This if state prevents
multiple readings
    {
      Serial.println("Offline: Trailing Edge");
      Serial.println("----------------------");
      // Increment the tick counter:
      switch( locationTick )
      {
        case 2:
          locationTick = 0;
          locationLine = 1;
          break;
        default:
          locationTick++;
          break;
      }  // end locationTick update
      timerOffline = millis();
      // Manipulate the Switch:
      switch( locationTick )
      {
        case 1:
          switchWrite(2);  // Turn the switch off
          if( OFFLINE == locationLine && locationNode == destination )
          {
            destination = -1;  // Arrived at specified destination
          }
          break;
        case 2:
          if( ONLINE == locationLine )  // The pod is exiting an Online
section of a junction
          {
            switchWrite(ONLINE);  // Turn on the online switch
          }
          else if( OFFLINE == locationLine )
          {
```

```
            switchWrite(OFFLINE);
          }
          break;
        default:
          break;
      }  // end locationTick switch
      printSnapshot();
    }  // end bounce
  }
  // Online Update:
  if(sensorOnlineStored < sensorOnlineNew )  // The leading edge condition
  {
    sensorOnlineStored = sensorOnlineNew;
  }
  else if( sensorOnlineStored > sensorOnlineNew )  // The trailing edge
condition
  {
    sensorOnlineStored = sensorOnlineNew;
    if( millis() - timerOnline > BOUNCE_TIME )  // This if statement prevents
multiple reads
    {
      Serial.println("Online: Trailing Edge");
      Serial.println("---------------------");
      // Increment the node counter:
      locationNodePrevious = locationNode;
      switch( locationNode )
      {
        case 4:
          if( STATION_3 != destination )
          {
            locationNode = 10;
          }
          else
          {
            locationNode++;
          }
          break;
        case 9:
          locationNode = 0;
          break;
        case 10:
          locationNode = 8;
          break;
        default:
          locationNode++;
          break;
      }  // end locationNode update switch
      //  Manipulate the switch:
      switch( locationNode )
      {
        case STATION_1:  // approaching Station 1
        case STATION_2:  // approaching Station 2
        case STATION_3:  // approaching Station 3
          if( locationNode == destination )       // At the destination node
          {
            locationLine = 0;                    // Taking the Offline route
            switchWrite(OFFLINE);
```

```
          } else {                                  // Not the destination node
            locationLine = 1;                       // Stays on the Online
section
            switchWrite(ONLINE);
          }
          break;
        case 4:                     // Approaching the bypass junction
          if( STATION_3 != destination )  // If Station 3 (node 5) is not the
destination...
          {  // Use the Mainline switch
            switchWrite(ONLINE);  // Turn on the Online switch
          }
          else // Use the Offline switch
          {
            switchWrite(OFFLINE);  // Turn on the Offline switch first
          }
          break;
        case 8:
          if( 10 == locationNodePrevious )
          {
            switchWrite(ONLINE);
          }
          else
          {
            switchWrite(OFFLINE);
          }
          break;
        default:  // This case will appear everytime the pod is exiting a
junction
          switchWrite(2);    // Turn off the switch
          locationTick = 0;  // Reset the tick counter
          locationLine = 1;
          break;
      }  // end locationNode switch
      printSnapshot();
      timerOffline = millis();
    }
  }
  // end Navigation

  /*
  OBJECT DETECTION
  */
  // end Object Detection
  /*
  SPEED CONTROL
  */
  // Update the motor duty cycle based on location and detected object range:
  if( (0 < range) && (MIN_RANGE >= range) )  // If the pod detects an object
within the critical range
  {
    digitalWrite(MOTOR_A1, HIGH);  // Perform active braking (HIGH-HIGH)
    digitalWrite(MOTOR_A2, HIGH);
    dutyCycle = 0;                // Set the duty cycle to 0
  }
  else if( (MIN_RANGE < range) && (MAX_RANGE >= range) )
  {
```

```
    dutyCycle = map(range, MIN_RANGE, MAX_RANGE, MIN_DUTY, MAX_DUTY);  //
Adjust the speed based on the distance
  }
  else if( NO_DESTINATION == destination )
  {
    digitalWrite(MOTOR_A1, HIGH);  // Perform active braking (HIGH-HIGH)
    digitalWrite(MOTOR_A2, HIGH);
    dutyCycle = 0;                // Set the duty cycle to 0
  }
  else
  {
    dutyCycle = MAX_DUTY;
  }
  if( MAX_DUTY < dutyCycle )  // Safety precaution in case the duty cycle
somehow becomes higher than the maximum duty cycle
  {
    dutyCycle = MAX_DUTY;
  }
  // Run the motors at the adjusted duty cycle:
  analogWrite(MOTOR_A1, dutyCycle);
  digitalWrite(MOTOR_A2, LOW);
  // end Speed Control


}  // end loop

void printSnapshot()
{
  Serial.print("Destination:  ");
  Serial.println(destination);
  Serial.print("Location:  ");
  Serial.print(locationNode);
  Serial.print("  (from ");
  Serial.print(locationNodePrevious);
  Serial.println(")");
  Serial.print("Tick:  ");
  Serial.println(locationTick);
  Serial.print("Line:  ");
  Serial.println(locationLine);
  Serial.print("Switch:  ");
  if( digitalRead(SWITCH_ONLINE) == HIGH && digitalRead(SWITCH_OFFLINE) ==
LOW)
  {
    Serial.println("Online");
  }
  else if( digitalRead(SWITCH_OFFLINE) == HIGH && digitalRead(SWITCH_ONLINE)
== LOW)
  {
    Serial.println("Offline");
  }
  else if( digitalRead(SWITCH_OFFLINE) == LOW && digitalRead(SWITCH_ONLINE)
== LOW)
  {
    Serial.println("Off");
  }
  else {
    Serial.println("Error");
```

```
  }
  Serial.print("Range:  ");
  Serial.println(range);
  Serial.print("Duty Cycle:  ");
  Serial.println(dutyCycle);
  Serial.println("-----------------------------------");
}

int destinationWrite(int station_num)  // Converts a station number to the
corresponding location node
{
  int node;
  switch( station_num )
  {
    case 1:  // Station 1
      node = STATION_1;
      break;
    case 2:
      node = STATION_2;  // Station 2
      break;
    case 3:  // Station 3
      node = STATION_3;
      break;
    default:
      node = -1;
      break;
  }  // end station_num switch
  return node;
}  // end destinationWrite;

void switchWrite(int line)
{
  switch(line)
  {
    case ONLINE:
      digitalWrite(SWITCH_OFFLINE, LOW);
      digitalWrite(SWITCH_ONLINE, HIGH);
      break;
    case OFFLINE:
      digitalWrite(SWITCH_ONLINE, LOW);
      digitalWrite(SWITCH_OFFLINE, HIGH);
      break;
    default:
      digitalWrite(SWITCH_ONLINE, LOW);
      digitalWrite(SWITCH_OFFLINE, LOW);
      break;
  }  // end line switch
}  //  end swtichWrite

// SPI interrupt routine
ISR (SPI_STC_vect)
{
byte c = SPDR;  // grab byte from SPI Data Register
Serial.print("ISR\n");
  // add to buffer if room
  if (pos < sizeof buf)
    {
```

```
    buf [pos++] = c;

    // example: newline means time to process buffer
    if (c == 0x00)
      process_it = true;

    }  // end of room available

}  // end of interrupt routine SPI_STC_vect
```

Appendix T: Arduino Pod class, respectively.

Network Integration

One of the key goals for the 2013-2014 Controls Team was to integrate the network architecture to the point that the 1/12-scale pods could be controlled wirelessly from a user interface run on a laptop, a feature that not fully realized in the previous generations team. Reaching this goal required integrating the Arduino with a separate microcontroller capable of wireless communication. The communication protocol chosen for the interface was SPI, based on the protocol being fully supported by Arduino UNO. Originally, the BeagleBone Black was planned for the wireless access, but it has since been substituted for the SJOne microcontroller running FreeRTOS. The SJOne board mounted on the pod communicates to the laptop through a second SJOne board connected to the laptop via USB.

Once wireless communication was established from the laptop to the Arduino through the two SJOne boards, the primary goal was two allow a person manning the laptop to be able to send a new destination to the pod once the current destination was complete. Two secondary goals were to further enhance the code to allow the laptop user to reset the pod's starting location as a different station (instead of the default), and also to reset the pods location on the track even if it was not currently stopped at a station. These two secondary goals were to allow the user to reset the pods location without having to first manually move the pod back to the default starting station.

The code to send and receive data on both of the SJOne boards, as well as the segment of code to process the data on the Arduino were all developed by Eriberto Velzquez, one of the Computer Engineering students on the Controls Team. Once the Arduino receives new data from the SJOne board, it waits until it no longer has a specified destination to process it. One of the most important things to keep in mind during the coding process was that once Arduino data interrupt is triggered by the SJOne, very few actions can be taken while the interrupt is active without potentially causing unpredictable results. For this reason, the data is only stored while the interrupt is active, and only after the interrupt has ended does the Arduino actually process the data.

The data exists as an array of integers, with the first being the type of command it is receiving, or mode. The mode determines whether the pod is simply receiving a new destination, a new destination along with a different starting station, or a new destination with a starting location not at a station. The particular code required to process the data buffer on the Arduino is detailed in /*
  Pod.h - Library for controlling Spartan Superway 1/12 scale pods
  Created by Cory Ostermann (Lead), Man Ho, Randall Morioka, and Anthony Vo
  Spartan Superway 2013-2014 Controls Team
*/

```
#ifndef Pod_h
#define Pod_h

#include "Arduino.h"

class Pod
{
  public:
         Pod(int identifier);
         //void initialize();             // Sets up intial pod state, including Depot
location and SPI settings
         void statusWrite(int status_num);
```

```
        int statusRead();
        int destinationRead();    // Returns the pods current destination
        void destinationWrite(int station_id);  // Sets the pods next destination
        int locationRead();              // Returns the pods current location
        void locationUpdate();    // Increments the pod's location as it reads tick marks
        //boolean ReflectiveSensor(int sensor, int edge);
        void switchWrite(int line);             // Manipulates the switching mechanism
        int DetectObject();              // Senses if an object is obstructing the path
(Ultrasonic)
        int speedReadPWM();              // Returns the speed of the pod as a PWM duty
cycle
        void speedWritePWM(int duty_cycle);     // Sets the speed of the of the pod using
a duty cycle
        void Cruise();
  private:
        /*
        Pod Characteristics
        */
        int _podID;        // An identifying number that the Network can use to
coordinate pods
        int _podStatus;    // Pod status: ready for instruction or not ready
        int _dutyCycle;    // The PWM duty cycle being applied to the motors
        int _locationNode; // The most recent node passed by the pod
        int _locationLine; // Mainline or Offline
        int _locationAlt;         // For deactivating the switch and slowing the pod
        int _destination;         // The next station the pod will stop at
        int _range;               // The range to the detected object (0 if no object
detected)
        int _stateTrack1New;             // Tracks the state of the location Node sensor
(true means HIGH)
        int _stateTrack1Prev;            // The previous state of the sensor
        int _stateTrack2New;             // Tracks the state of the location Alt sensor
(true means HIGH)
        int _stateTrack2Prev;            // The previous state of the sensor

        /*
        State Trackers:
        State trackers will be boolean variables to determine if systems are currently
active
        (i.e. a track switch is on/true or off/false)
        */
        boolean _stateSwitchMainline;          // Tracks if the Mainline switch solenoid
is active (true)
        boolean _stateSwitchOffline;           // Tracks if the Offline switch solenoid
is active (true)
        /*
        Time Trackers:
        Time trackers are unsigned long variables used in conjunction with the millis()
command
        in order to produce non-blocking code

        NOTE: Unless agreed upon by all team members, delay() should never be used.
        */

};

#endif
```

```
/*
  Pod.cpp - Library for controlling Spartan Superway 1/12 scale Maker Faire exhibition
pods
  Created by Cory Ostermann (Controls Team Lead), Man Ho, Randall Morioka, Eriberto
Velazquez, and Anthony Vo
  Spartan Superway 2013-2014 Controls Team
*/

// Arduino Pins
#define MOTOR_A1     3             // The M1 pin for the motor driver
#define MOTOR_A2     2
//#define MOTOR_B1   5             // The M2 pin for the motor driver
//#define MOTOR_B2   4
#define SOLENOID_MAINLINE   4      // The pin to manipulate the transistor attached to the
Mainline switch solenoid (left)
#define SOLENOID_OFFLINE    5      // The pin to manipulate the transistor attached to the
Offline switch solenoid (right)
//#define M2_ENCODERA
//#define M2_ENCODERB
#define       TRIG   6
#define ECHO  7
#define TRACK_SENSOR1       A1
#define TRACK_SENSOR2       A0
#define SPI_INTERRUPT       9

// Additonal Properties
#define WHEEL_DIAMETER      1.875  // The diameter of the bogie wheels in inches
#define MAX_DUTY     75                    // The maximum allowable duty cycle (normal
cruise speed)
#define MIN_DUTY     60                    // The minimum duty cycle the pod will travel at
before stopping completely
#define HALF_DUTY    60                    // The duty cycle for entering or leaving a
station
#define MAX_DISTANCE 50            // The limit when the pod will reduce speed to account
for detected objects (TBD)
#define MIN_DISTANCE 25            // The limit when the pod will apply the Emergency
Brake (i.e. stop the motors) (TBD)
#define MAINLINE     1
#define OFFLINE             0
#define READY        1
#define NOT_READY    0
#define THRESHOLD    30                    // Threshold for the reflective sensors
#define LEADING_EDGE 1             // Leading Edge of the track marker
#define TRAILING_EDGE       0             // Trailing Edge of the track marker

#include "Arduino.h"
#include "Pod.h"
//#include "SPI.h"
//#include "SD.h"                  // The Adadfruit library for using an SD card for data-
logging
//#include "RTClib.h"              // The Adafruit library for using a real-time clock to
timestamp data in data-logging
#include "NewPing.h"

Pod::Pod(int identifier)
{
       _podID = identifier; // Give the pod it's identifier
       _podStatus = READY;         // Set the pod's initial state
```

```
        _locationNode = 0;          // Set the initial location at the Depot
        _locationLine = OFFLINE;
        _locationAlt = 2;
        _range = 0;
        _stateTrack1Prev = LOW;
        _stateTrack2Prev = LOW;

        /*
        Setup the pins for actuators and sensors
        */
        // Motors
        pinMode(MOTOR_A1, OUTPUT);
        digitalWrite(MOTOR_A1, LOW);
        pinMode(MOTOR_A2, OUTPUT);
        digitalWrite(MOTOR_A2, LOW);
        //pinMode(MOTOR_B1, OUTPUT);
        //digitalWrite(MOTOR_B1, LOW);
        //pinMode(MOTOR_B2, OUTPUT);
        //digitalWrite(MOTOR_B2, LOW);
        // Solenoids
        pinMode(SOLENOID_MAINLINE, OUTPUT);
        digitalWrite(SOLENOID_MAINLINE, LOW);
        pinMode(SOLENOID_OFFLINE, OUTPUT);
        digitalWrite(SOLENOID_OFFLINE, LOW);
        // Motor Encoders
        //pinMode(M1_ENCODERA, INPUT);
        //pinMode(M1_ENCODERB, INPUT);
        //pinMode(M2_ENCODERA, INPUT);
        //pinMode(M2_ENCODERB, INPUT);
        // Ultrasonic Sensor
        pinMode(TRIG, OUTPUT);
        pinMode(ECHO, INPUT);
        // Track Sensor(s)
        pinMode(TRACK_SENSOR1, INPUT);
        pinMode(TRACK_SENSOR2, INPUT);

}

void Pod::statusWrite(int status_num)
{
        _podStatus = status_num;
}

int Pod::statusRead()
{
        return _podStatus;
}

int Pod::destinationRead()  // Returns the pods current destination (node)
{
        return _destination;
}

void Pod::destinationWrite(int station_id)      // Sets the pods next destination (node)
{
        switch (station_id)
        {
        case 1:
```

```
                _destination = 0;
                break;
        case 2:
                _destination = 2;
                break;
        case 3:
                _destination = 5;
                break;
        default:
                break;
        }
}

int Pod::locationRead()           // Returns the pods current location
{
        return _locationNode;
}

void Pod::locationUpdate()  // Increments the pod's location as it reads tick marks
{
        // Get readings from the Track Sensors
        int readingSensor1 = analogRead(TRACK_SENSOR1);
        int readingSensor2 = analogRead(TRACK_SENSOR2);
        // Update the previous sensor states
        _stateTrack1Prev = _stateTrack1New;
        _stateTrack2Prev = _stateTrack2New;
        // Update the new sensor states based on the sensor readings
        if( readingSensor1 > THRESHOLD )
        {
                _stateTrack1New = HIGH;
        } else {
                _stateTrack1New = LOW;
        }
        if( readingSensor2 > THRESHOLD )
        {
                _stateTrack2New = HIGH;
        } else {
                _stateTrack2New = LOW;
        }

        if( (_stateTrack1New == LOW) && (_stateTrack1Prev == HIGH) )  // The Primary Track
Sensor is triggered
        {
                // Increment the location Node to the next section
                if( _locationNode == 7 )
                {
                        _locationNode = 0;
                }
                else
                {
                        _locationNode++;
                }
                // And the pod is about to enter junction
                if( _locationNode == (_destination) )           // If the next sector is
the destination node
                {
                        switchWrite(OFFLINE);                                   // Switch to
the Offline segment
```

```
                        _locationLine = OFFLINE;                        // Set location to
Offline
                }
                else if( _locationNode != (_destination) )      // If the next sector is
not the destination
                {
                        switchWrite(MAINLINE);                                  // Activate
the switch for the Mainline
                        _locationLine = MAINLINE;
                }
                else if( _locationLine == OFFLINE )      // If the pod is entering the
Mainline from Offline
                {
                        _locationLine = MAINLINE;   // Set the line to Mainline
                        switchWrite(2);                         // Turn off the switch
solenoids
                }
                else if( _locationLine == MAINLINE ) // If the pod is exiting the Mainline
section of a section with a station
                {
                        switch (_locationNode)
                        {
                        case 0:
                        case 2:
                        case 5:
                                switchWrite(2);             // Turn off the solenoid
                                break;
                        default:
                                break;
                        }       // end switch
                }       // end if
        }       // end Primary sensor check
        if( (_stateTrack2New == LOW) && (_stateTrack2Prev == HIGH) )  // The Secondary
Track Sensor is triggered
        {
                if( _locationLine == MAINLINE )          // If on the Mainline
                {
                        switchWrite(MAINLINE);                  // Toggle the switching
mechanism
                }
                if( _locationLine == OFFLINE )           // If Offline...
                {
                        switch (_locationAlt)
                        {
                        case 0:
                                switchWrite(OFFLINE);                   // Toggle the
switching mechanism
                                _locationAlt = 1;
                                break;
                        case 1:
                                if( _locationNode == _destination )
                                {
                                        _locationAlt = 2;
                                        break;
                                } else {
                                        switchWrite(OFFLINE);                   // Toggle the
switching mechanism
                                        _locationAlt = 0;
```

```
                                        break;
                                }
                        case 2:
                                _podStatus = NOT_READY;
                                break;
                        }       // end switch
                }       // end OFFLINE if
        }       // end Secondary Sensor check
}       // end locationUpdate
/*
boolean Pod::ReflectiveSensor(int sensor, int edge)    // COULD NOT GET TO WORK
{
        static int reading, newVal, prevVal;                    // Stores Reflective Object
Sensor reading
        //boolean result;           // Stores the result to be returned
        // Get relevant sensor data
        switch( sensor )
        {
        case 1:         // Primary track sensor
                reading = analogRead(TRACK_SENSOR1);
                prevVal = _stateTrack1Prev;
                break;
        case 2:         // Secondary track sensor
                reading = analogRead(TRACK_SENSOR2);
                prevVal = _stateTrack1Prev;
                break;
        default:
                break;
        }       // end sensor switch
        if (reading > THRESHOLD)    // The Reflective Object Sensor reads the felt markers
        {
                newVal = LOW;
        } else {
                newVal = HIGH;
        }
        // Update the new state variables:
        switch( sensor )
        {
        case 1:
                _stateTrack1New = newVal;
                break;
        case 2:
                _stateTrack2New = newVal;
                break;
        default:
                break;
        }       // end sensor switch
        if( newVal == LOW && prevVal == HIGH)     // Triggering on the leading edge
        {
                prevVal = LOW;
                switch( sensor )
                {
                case 1:
                        _stateTrack1Prev = prevVal;
                        break;
                case 2:
                        _stateTrack2Prev = prevVal;
                        break;
```

```
                default:
                        break;
                }       // end sensor switch
                if(edge == LEADING_EDGE)          // The leading edge was specified
                {
                        return true;
                } else {
                        return false;
                }
        }
        else if ( newVal == HIGH && prevVal == LOW)     // Triggering on the trailing edge
        {
                prevVal = HIGH;
                switch( sensor )
                {
                case 1:
                        _stateTrack1Prev = prevVal;
                        break;
                case 2:
                        _stateTrack2Prev = prevVal;
                        break;
                default:
                        break;
                }       // end sensor switch
                if(edge = TRAILING_EDGE)          // The trailing edge was specified
                {
                        result = true;
                } else {
                        result = false;
                }
        }
        else
        {
                result = false;
        }
        return result;

}       // end ReflectiveSensor
*/

void Pod::switchWrite(int line)
{
        if( line == MAINLINE && SOLENOID_MAINLINE == LOW)
        {
                digitalWrite(SOLENOID_OFFLINE, LOW);
                _stateSwitchOffline = false;
                digitalWrite(SOLENOID_MAINLINE, HIGH);
                _stateSwitchMainline = true;
        }
        else if( line == MAINLINE && SOLENOID_MAINLINE == HIGH)
        {
                switchWrite(2);       // Turn off both solenoid
        }
        else if( line == OFFLINE )
        {
                digitalWrite(SOLENOID_MAINLINE, LOW);
                _stateSwitchMainline = false;
                digitalWrite(SOLENOID_OFFLINE, HIGH);
```

```
                _stateSwitchOffline = true;
        }
        else if( line == OFFLINE && SOLENOID_OFFLINE == HIGH)
        {
                switchWrite(2);      // Turn off both solenoid
        }
        else
        {
                digitalWrite(SOLENOID_OFFLINE, LOW);
                _stateSwitchOffline = false;
                digitalWrite(SOLENOID_MAINLINE, LOW);
                _stateSwitchMainline = false;
        }
}

int Pod::DetectObject()     // Use the Ultrasonic Sensor make sure there are no
obstructions
{
        NewPing sonar(TRIG, ECHO, MAX_DISTANCE);
        _range = sonar.ping_cm();
        return _range;       // If no objects are detected, return 0;
}

int Pod::speedReadPWM()
{
        return _dutyCycle;
}

void Pod::speedWritePWM(int duty_cycle)
{
        _dutyCycle = duty_cycle;
}

void Pod::Cruise()
{
        if ((_range != 0) && (_range < MIN_DISTANCE))    // if the pod is within critical
range
        {
                // Activate the "emergency brakes" (i.e. stop motors)
                digitalWrite(MOTOR_A1, HIGH);
                digitalWrite(MOTOR_A2, HIGH);
                //digitalWrite(MOTOR_B1, HIGH);
                //digitalWrite(MOTOR_B2, HIGH);
                _dutyCycle = 0;                                  // set the duty cycle to
zero
        }
        else if ((_locationAlt == 2) && (_podStatus != READY)) // if the pod
arriving/sitting at a station
        {
                // Stop the motors
                digitalWrite(MOTOR_A1, HIGH);
                digitalWrite(MOTOR_A2, HIGH);
                //digitalWrite(MOTOR_B1, HIGH);
                //digitalWrite(MOTOR_B2, HIGH);
                _dutyCycle = 0;                                  // set the duty cycle to
zero
        }
```

```
	else if ((_range >= MIN_DISTANCE) && (_range <= MAX_DISTANCE))// If the pod
detects an object
	{
		_dutyCycle = map(_range, MIN_DISTANCE, MAX_DISTANCE, MIN_DUTY, MAX_DUTY);
	// Reduce speed based on the distance to the object
	}
	else if (_locationAlt == 1)	// If the pod is entering the Offline segment of
track
	{
		_dutyCycle = HALF_DUTY;		// Reduce duty cycle by half
	}
	else if ((_range == 0) && (_locationAlt == 0))  // If the pod is leaving the
Offline segment
	{
		_dutyCycle = MAX_DUTY;		// Increase duty cycle to merge onto the
Mainline
	}
	// Manipulate the motors according to the current duty cycle
	analogWrite(MOTOR_A1, _dutyCycle);
	//analogWrite(MOTOR_B2, _dutyCycle);
	digitalWrite(MOTOR_A2, LOW);
	//digitalWrite(MOTOR_B1, LOW);

}
```

Appendix U: Arduino SPI Sketch.

Maker Faire Shortcomings

Navigation

Despite careful preparations, the 1/12-scale model still experienced difficulties in sensing its location along the track under direct sunlight. It was decided earlier in the semester that infrared reflective object sensors would be used to help the pod detect its location along the track. Through extensive testing in different lighting scenarios inside the design space, the team observed great fluctuations in readings from the sensors under different ambient lightings due to changing infrared compositions. As a precaution, the bottom section of the entire track was lined with reflective metal foil to help increase the consistency of the sensor readings. During our testing indoor, it was benchmarked that reading values from ambient light (reflected from metal foils and ambient light) would be around 1000 and values read from the felt markers would be in the 200-400 range, leaving a significant delta between detection and non-detection for better accuracy and prevent bouncing.

However, the conditions at Maker Faire proved to be of a different magnitude as that inside the design space. The clear-sky condition during both days of Maker Faire meant ambient light was flushed with infrared spectrum. The fact that the scale model was position in open space, away from any shade means that the entire model (all sides and angles) was flushed with a very large amount of infrared coming from the sun. Having the sensors pointing upwards did not help the matter, even though they were positioned near the bottom surface of the guideway which theoretically should have provided some sort of shade. This resulted in ambient condition (non-detection) being registered around 1000 while the markers (detection) being read in the upper 900 range. The significantly smaller difference between the two threshold values meant that the pod frequently misread the markers as it traveled along the track, resulting in early or/and late actuations of the switching mechanisms, which ultimately led to failures in navigating the track and reaching the intended destination. This caused problems in demonstrating the validity of the system to spectators with the negative effects being most pronounced during the hours between early morning and late afternoon. When the intensity of sunlight started to decrease around 5 pm, the readings became more accurate and the pod was able to navigate the track better, resulting in more flawless and consistent runs.

To help alleviate the problems cause by the sun, the team raised the reflective sensor closer to the marker to the point of almost touching. This helped lower the value read on the marker but came with its own disadvantages. Due to the pod's dynamics around the track and relatively high wind gusts throughout the day at the Faire location, the lower portion of the pod experienced wiggling frequently. With the sensor mounted very close to the marker, the frequent wobbling led to the risk of collision between the sensors and the markers which could damage the sensors or knocked the markers off the guideway. Evidently, the choice of infrared reflective sensors, while providing a simple sensing method, did not prove to be robust enough to be used under different light environments.

Power Management

The current electronic components being used on the model consumed more power than intended and caused power management problems at Maker Faire. Prior to the Faire, the team designed the scale model to run on two packs of 4-AA batteries with one pack powering the microcontrollers together with the sensors and the other powering the actuators (switching solenoids, DC motors). The solenoids were rated to draw about 250mA of current each and the DC motors at 120mA with no load. Assuming that

the model would be running throughout the day for both days, the team anticipated high energy consumption by the system and prepared 40 single-use alkaline batteries for the Faire but ended up using more than the number prepared.

Power consumption by the pod increased as problem with sensor readings was alleviated. During the first half of the first day, due to problems caused by the reflective sensors, very few complete runs were carried out by the pod and thus power consumption was low. However, as the problems were troubleshot and the pod making more successful runs in the latter half of the day, the batteries were drained faster. With the pod making a successful loop of the track every 2-3 minutes, it was estimated that a set of four batteries for the actuators was depleted after an average of 30-45 minutes. The culprit of this problem was determined to be the switching solenoids. After a new set of batteries was changed in, the solenoids were switched on completely and consistently. However, after several runs, while the motors could still run, the solenoid did not fully push out when actuated and resulted in the bogie getting stuck at track junctions or failed to switch to the intended section of the track. At the end of the first day, the initially prepared battery set was completely used up and the team had to acquire more batteries for the second day.

Although new type of battery was used for the second Faire day, problem with power consumption persisted and rendered more failed runs. With the hope that rechargeable NiMH batteries would provide more charge and current, the team acquired at least 12 batteries of such type together with a quick charger. Unfortunately, the problem persisted with the new batteries and became worse. Initially, from being fully charged through the night before, the batteries were drained at about the same rate as the single-use type. After being charged by the quick charger with the charger, as indicated by the charger's indicator, the batteries held up for even shorter amount of time with some of them not able to power the pod fresh out of the charger. As a result, the team had to spend more than an hour troubleshooting the problem with the pod sitting still and not being able to demonstrate it to many interested spectators. The team then resorted to switching back to single-use alkaline batteries which proved to be more reliable as well as being able to power the pod through the rest of the day.

Besides problems with the sensor reading, power consumption was the second biggest drawback of the current design that, unfortunately, prevented the team from demonstrating the system to a significant number of interested spectators. Given the large number of Faire goers being attracted to the scale model, each failure to run the pod was a significant missed opportunity to demonstrate our project to the public. Such missed opportunities should have been minimized to help create a positive impression of Spartan Superway and ATN as a whole for as many spectators as possible. Fortunately, the very dedicated team lead and several other team members were able to enthusiastically explain the concept of the project to spectators while the pod was disabled, thus lessening the negative impacts caused by the lack of actual demonstration.

1/12-Scale Track

Besides the two major problems caused by power consumption and sensor readings, the current track model have several disadvantages that were pronounced during the preparation for the Faire as well as during and after the Faire. First of all, the configuration of the track was quite fragile and caused difficulties during disassembling and assembling of the track. The acrylic sections of the track, while aesthetically pleasing and allow spectators to see the bogie inside, are fragile and require great care while handling. In fact, at least on piece was broken while the track was disassembled to be move to the

Faire location. The bottom surfaces of another section of the track, where the bogie drive wheels get traction, were flexed out of balance and impeded the pod's movement at the first turn after the home station. This problem was exacerbated by the track being not level. For reasons that included the ground surface at the Faire, the entire track could not be leveled even though each individual section was leveled when they were mounted on the supporting posts. The track was also inflexible in the sense that several of theoretically universal pieces could not be used interchangeably. For example, the metal brackets that were used to connect track sections together could not provide seamless connections. Some brackets worked well for some connections but could not fit in other connections; some brackets allowed for flushed connections between the sections while others left relatively large gaps that impeded movement of the pod. Furthermore, the wires connecting the actuators on the bogie to the microcontroller in the cabin as well as the protective electrical tape around them were scraped off due to friction caused by the small gap in the guideway bottom surface. In addition, transporting the track was not an easy task as at least three types of vehicles were needed: a heavy duty pickup truck to carry the steel sections, a compact pickup truck for all the posts, and two cars to transport the acrylic sections.

Secondly, the track posed several safety issues to spectators, especially small children. The cylindrical supporting posts of the track have openings with sharp edges that posed serious safety hazards. Fortunately, a member of the team, Eddie Velasquez, realized the problem early and proposed the ingenious idea of taping the openings to prevent cutting when people come in contact with them. The solution proved to be very effective as there was at least one occasion where a young spectator, while chasing the moving pod, bumped into one of the post openings and was startled by the impact but fortunately was not cut or seriously hurt.

Next, the height of the track caused some trouble during troubleshooting, entering, and exiting the exhibition area, as well as the potential damage due to people leaning on it. With the height of the track being relatively low and the pod hanging underneath it, troubleshooting problems on the pod and inspecting the track proved to be somewhat difficult as team members had to crouch down then turn to face upward for extended periods of time. The exhibition station being inside the track loop and track's low height also means that team members had to crouch down quite low to enter or exit the exhibition area. Another problem with the track that was only obvious at Maker Faire is that the height of the track seemed to be positioned perfectly for young spectators. As the guideway was at the eye level of many child spectators, it was often seen that they would push or pull the pod along the guideway forcefully or poking the electronics inside the cabin frame which might cause damages to the model. In other instances, people were caught leaning on the track, which was very dangerous, especially with the acrylic sections.

Overall, the current track design, while operational, contains several disadvantages. The inflexible and fragile design means that the team risked damaging the track every time it needed to be taken apart and reassembled after which its condition seemed to deteriorate. Some components of the track posed serious safety issues but were properly fixed. Finally, the height of the track caused some practicality problems when operating on the model.

Maker Faire Successes

Despite the shortcomings as previously mentioned, the Controls Team believes that the scale model as well as the other exhibits at Maker Faire proved effective at garnering attention from the spectators.

For the Controls Team's exhibit specifically, when the pod was functional, it was able to navigate itself around the track while the members were able to communicate with the growing spectators.

When it was working the navigation subsystem was a huge success because the pod was able to read the felt markers underneath the track. From there it was able to keep track of its location, with relation to the guide way, and from there depending on the situation, turn specific solenoids on and off depending on the situation. Finally, it was able to reach its destination depending on the location that was programmed onto the Arduino.

In addition to this, despite the fact that the speed control was never implemented, the pod was able to go around the track at a relatively decent speed around the track. The speed that the team chose also gave the pod the ability to navigate through the gaps that were located near the joint brackets most of the time.

As previously mentioned, the Controls Team believes that by attending Maker Faire and being able to run it in conjunction to the computer engineer's line following robot, the team was able to demonstrate a proof of concept that ATN can be done. When working in conjunction, the scaled model proved that it was able to navigate itself around the track, take a shortcut when it needs to, and arrive at its destination. The line follower, kept generating different values, which simulated the reservation system and it was able to navigate itself along the course and pick up and let off its potential customer. By being able to run the two together the Controls Team including the computer engineers, were able to generate interest among the spectators, receive positive feedback, and most importantly make the public aware of what Spartan Superway is trying to achieve.

Next Steps

Update Navigation/Object Detection Sensors

On September 13, 2013 a Kickstarter campaign to produce an Arduino-compatible camera system that incorporates image and color recognition was successfully funded. The system is called CMUcam5, but is more commonly known as Pixy. Pixy was developed cooperatively between Charmed Labs and Carnegie Mellon University, and as of April 2014, it is available for purchase at the reasonable price of approximately $70.

Figure 75. Pixy CMUcam5. Pixy is an Arduino-compatible image recognition system that integrates an automotive industry-level camera.

The Pixy is capable of communicating with any microcontroller that has SPI, I^2C, or a universal asynchronous receiver/transmitter (UART). The actual image processing is done on the Pixy itself, and it only transmits relevant data to the connected microcontroller to prevent overloading it with the data stream. The Pixy can be "taught" to recognize objects based on color and shape, and this is further enhanced by the ability to combine colored shapes to create color codes (CMUcam, 2014). This means that instead of using felt markers and reflective object sensors to read them, unique color codes could be placed along the track for more specific signals, such as a particular station, a junction, or the back or side of another pod. Using color codes to denote the different sides of a pod would allow the pod to determine its proximity and orientation to another pod, and allow it to prevent collisions, particularly at merge points.

As image recognition is becoming the standard in autonomous, driverless cars, and also in high-end assisted cruise control systems like Subaru's EyeSight system, switching to an image processing based sensor as early as possible would greatly enhance development of the system. The actual camera within the Pixy is targeted for the automotive market, and the camera itself can also be replaced as newer and better cameras become available.

Update Track Design

As previously mentioned, one of the problems with the current scaled model is that the acrylic sections are very fragile. During Maker's Faire the team noticed that a lot of people would to lean on the track more importantly there were several instances when children would run around the track which caused the team some concern since there were some moments when the kids would accidentally run into the track. Also, due to carelessness around the track when assembling and dissembling there are many cracks here and there along the acrylic sections.

If a new track becomes the goal here are some recommendations from the 2013-2014 Controls Team. First, make the track similar to the current guide way plans, by doing this the public will have an easier time visualizing the workings of Spartan Superway's ATN. Second, make the track out of sheet metal

that way concerns about the track breaking can be put at ease. Third, make the entire guide way higher than the current model or create a way inside and outside the track.

If it is decided to continue working with the current model and still work with the reflective sensors please consider the following. First, play with the idea of making a light box around the reflective sensors in order to keep out a majority in not all ambient light. Second, round the edges of the support poles in order to ensure that the poles won't cut anyone. Third, make one of the curved sections an entry point and attempt to make that part into a doorway to give easy access to the inside of the track.

References

Arduino. (2014, May 21). *Arduino Style Guide for Writing Libraries*. Retrieved from Arduino: http://arduino.cc/en/Reference/APIStyleGuide

CMUcam. (2014, May 22). *Introduction and Background*. Retrieved from CMUcam: Open Source Programmable Embedded Color Vision Sensors: http://www.cmucam.org/projects/cmucam5/wiki/Introduction_and_Background

Eckel, T. (2014, May 21). *NewPing Library for Arduino*. Retrieved from Arduino Playground: http://playground.arduino.cc/Code/NewPing

Oomlout. (2014, May 21). *How to Make Anything (Using Acrylic and Machine Screws)*. Retrieved from Instructables: http://www.instructables.com/id/How-to-Make-Anything-Using-Acrylic-and-Machine-Sc/

SuperWay. (2013). *SuperWay - An Solar Powered Automated Transportation System.* San Jose: San Jose State University.

Younkin. (2014, March 3). *Measuring Motor Parameters*. Retrieved from http://support.ctc-control.com/customer/elearning/younkin/motorParameters.pdf

Appendix A: Eccentric Loading Calculations

The calculations for the eccentric loading of the main support column

Mechanical Properties of the Main Support Column

$$Dimensions: 4\ in. X\ 4\ in. X\ .25\ in. thick\ cross\ section\ square\ tube, 10\ feet\ long$$

$$Modulus\ of\ Elasticity = E = 30\ Mpsi$$

$$Tensile\ Yield\ Strength = 42\ kpsi \quad | \quad Assumption: compressive\ yield\ strength = tensile\ yield\ strength$$

$$Area\ Moment\ of\ Inertia = I = \frac{1}{12} * b * h^3 = 8.83\ in^4$$

$$Cross\ Sectional\ Area = A = b * h = 3.75\ in^2$$

$$Radius\ of\ Gyration = k = \sqrt{\frac{I}{A}} = 1.53\ in$$

$$Length = 120in \quad | \quad l_{eff} = 2.1 * Length = 252in \quad | \quad AISC\ recommended\ length\ for\ fixed - free\ ends$$

$$Slenderness\ Ratio = S_r = \frac{l_{eff}}{k} = 164$$

$$Johnson \rightarrow Euler\ Threshold = (S_r)_D = \pi\sqrt{\frac{2E}{S_{yc}}} = 118.7$$

$$S_r \geq (S_r)_D \therefore\ Eccentric\ Buckling\ is\ Within\ Euler\ Region$$

$$Eccentricity = e = 40in$$

$$Distance\ From\ Neutral\ Axis = c = 2in$$

$$Force\ of\ load = P_{load} = 650\ lb\ \downarrow$$

Calculations for Critical Buckling Load and Safety Factor

$$Secant\ Column\ Formula = \sigma_c = \frac{P}{A}\left[1 + \left(\frac{ec}{k^2}\right)\sec\left(\frac{l_{eff}}{k}\sqrt{\frac{P}{4EA}}\right)\right]$$

$$Software\ Used\ to\ Iterativly\ Solve\ for\ P$$

$$P_{critical} = 3988\ lb$$

$$Safety\ Factor = SF = \frac{P_{critical}}{P_{load}} = 6.1$$

Appendix B: Bending Stress Calculations

The calculations for bending stresses on the main support column

Mechanical Properties of the Main Support Column

$$Dimensions: 4\ in. X\ 4\ in. X\ .025\ in. thick\ cross\ section\ square\ tube, 10\ feet\ long$$

$$Tensile\ Yield\ Strength = 42\ kpsi \quad | \quad Assumption: compressive\ yield\ strength = tensile\ yield\ strength$$

$$Area\ Moment\ of\ Inertia = I = \frac{1}{12} * b * h^3 = 8.83\ in^4$$

$$Force\ Distance\ from\ column = d = 40in$$

$$Distance\ From\ Neutral\ Axis = c = 2in$$

$$Force\ of\ load = P_{load} = 650\ lb\ \downarrow$$

$$Bending\ Moment\ Applied\ to\ Column = M = P_{load} * d = 26{,}000\ lb \cdot in$$

Calculations for Bending Stress and Safety Factor

$$Stress\ from\ Applied\ Moment = \sigma_{applied} = \frac{Mc}{I} = 5.89\ kpsi$$

$$Saftey\ Factor\ for\ Bending = SF = \frac{Tensile\ Yield\ Strength}{\sigma_{applied}} = 7.1$$

Appendix B: Bending Stress Calculations

The calculations for bending stresses on the main support column

Mechanical Properties of the Main Support Column

$$Dimensions: 4\ in. X\ 4\ in. X\ .025\ in. thick\ cross\ section\ square\ tube, 10\ feet\ long$$

$$Tensile\ Yield\ Strength = 42\ kpsi \quad | \quad Assumption: compressive\ yield\ strength = tensile\ yield\ strength$$

$$Area\ Moment\ of\ Inertia = I = \frac{1}{12} * b * h^3 = 8.83\ in^4$$

$$Force\ Distance\ from\ column = d = 40in$$

$$Distance\ From\ Neutral\ Axis = c = 2in$$

$$Force\ of\ load = P_{load} = 650\ lb\ \downarrow$$

$$Bending\ Moment\ Applied\ to\ Column = M = P_{load} * d = 26{,}000\ lb \cdot in$$

Calculations for Bending Stress and Safety Factor

$$Stress\ from\ Applied\ Moment = \sigma_{applied} = \frac{Mc}{I} = 5.89\ kpsi$$

$$Saftey\ Factor\ for\ Bending = SF = \frac{Tensile\ Yield\ Strength}{\sigma_{applied}} = 7.1$$

Appendix D: Center of Mass and Moment of Tipping Calculations

Calculations for center of mass and wind speed for tipping

Calculations for Center of Mass of System and Moment at Tipping Axis

Center of Mass on Cartesian Plane with Origin at Tipping Corner (Figure 4)					
Component	**Weight [lb]**	**Horizontal Distance from Tipping Axis [in]**	**Horizontal Distance X Weight**	**Vertical Distance From Tipping Axis [in]**	**Horizontal Distance X Weight**
Cabin	150	-18	-2700	50	7500
Bogie	317	-18	-5706	102	32334
Guideway	400	-18	-7200	110	44000
Backplate	24	-26	-624	94	2256
Support Arm	74	-42	-3108	94	6956
Tube	262	-56	-14672	60	15720
Baseplate	68	-26	-1768	0	0
SideBase	36	-56	-2016	0	0
SideBrace	48	-56	-2688	0	0
LongBrace	68	-27	-1836	0	0
Total/Average	1447	-29.25	-42318	75.17	108766

Parameters Used For Wind Load Calculations		
Component	**Cross Sectional Area for Horizontal Wind Load [in^2]**	**Vertical Distance for Resultant Wind Force [in]**
Solar Panels	3456	156
Guideway	6912	102
Support Column	960	60
Cabin	4608	48

Calculations for Maximum Wind Speed, and Safety Factor when Tipping

$$Force\ of\ Wind\ Incident\ on\ Object = F_{wind} = Area \times Wind\ Pressure \times Coefficent\ of\ Drag$$

$$Wind\ Pressure\ [\frac{lb}{in^2}] = 0.0000178 \times v^2 \quad | \quad v\ is\ in\ mph$$

$$Coefficent\ of\ Drag\ on\ Rectangular\ Box = C = 2.1$$

$$Moment\ to\ Overcome\ by\ Wind\ Force = M_{weight} = X_{cm} \times W_{total} = 42318\ lb \cdot in$$

$$Wind\ Velocity\ to\ Overcome\ System\ Weight = v_{tipping} = \sqrt{\frac{M_{weight}}{0.0000178 \times C \sum A_i d_i}} = 27\ mph$$

$$Vertical\ Distance\ Resultant\ Wind\ Force\ Applies = d_{wind} = 95\ in$$

$$Resultant\ Wind\ Force\ Required\ to\ Tip = F_{wind} = 443\ lb$$

$$Wind\ Force\ and\ Load\ Weight\ on\ a\ Single\ Large\ Angled\ Brace\ Member = F_{total} = 821\ lb$$

$$Safety\ Factor\ for\ Large\ Angled\ Brace\ Member\ Under\ Tipping\ Conditions = SF = \frac{P_{critical}}{F_{total}}$$
$$= 1.67$$

Appendix E: System Integrity Calculations

Calculations to determine system integrity at maximum tipping angle

$$Angle\ Support\ is\ Tipped\ when\ Center\ of\ Mass\ is\ above\ Tipping\ Axis = \theta = \tan^{-1}\left(\frac{29.25}{75.17}\right) = 21.26°$$

$$Angle\ of\ Support\ Member\ off\ Horizontal\ when\ Tipped = 51.26°$$

$$Moment\ Applied\ to\ System\ at\ Maximum\ Tip\ angle = M_{max} = 58147\ lb \cdot in$$

$$Maximum\ force\ on\ Support\ Brace\ Member = P_{max} = \frac{M_{max}}{29\ in \cdot \cos(51.26) \cdot 2} = 1600\ lb$$

$$Safety\ Factor\ for\ Brace\ Under\ Maximum\ Moment\ Tipping\ Condition = SF = \frac{P_{critical}}{P_{max}} = 0.86$$

Appendix F: Guideway SolidWorks Analysis

Study Properties

Study name	Study 1
Analysis type	Static
Mesh type	Solid Mesh
Thermal Effect:	On
Thermal option	Include temperature loads
Zero strain temperature	298 Kelvin
Include fluid pressure effects from SolidWorks Flow Simulation	Off
Solver type	FFEPlus
Inplane Effect:	Off
Soft Spring:	Off
Inertial Relief:	Off
Incompatible bonding options	Automatic
Large displacement	Off
Compute free body forces	On
Friction	Off
Use Adaptive Method:	Off
Result folder	SolidWorks document (C:\Users\Daniel\Dropbox\Reference\Previous Classes\SJSU - Spring 2014\ENGR 195D Senior Project\Wood Model)

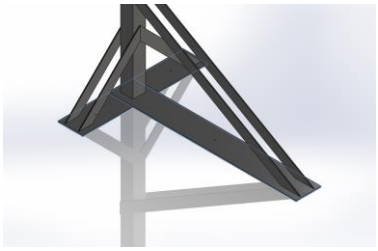
Units

Unit system:	SI (MKS)
Length/Displacement	mm
Temperature	Kelvin
Angular velocity	Rad/sec
Pressure/Stress	N/m^2

Material Properties

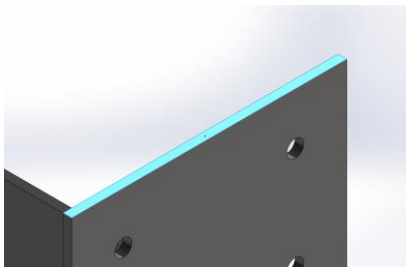
Model Reference	Properties	Components
	Name: **Plain Carbon Steel** Model type: **Linear Elastic Isotropic** Default failure criterion: **Unknown** Yield strength: **2.20594e+008 N/m^2** Tensile strength: **3.99826e+008 N/m^2** Elastic modulus: **2.1e+011 N/m^2** Poisson's ratio: **0.28** Mass density: **7800 kg/m^3** Shear modulus: **7.9e+010 N/m^2** Thermal expansion coefficient: **1.3e-005 /Kelvin**	**SolidBody 1(Mirror2)(Support_mounting_plate-2),** **SolidBody 1(Boss-Extrude1)(support_arm-1),** **SolidBody 1(Boss-Extrude1)(support_arm-2),** **SolidBody 1(Boss-Extrude1)(support_base_long-1),** **SolidBody 1(Boss-Extrude1)(support_base_short-1),** **SolidBody 1(Boss-Extrude1)(support_base_short-3),** **SolidBody 1(Boss-Extrude1)(support_brace_long-1),** **SolidBody 1(Boss-Extrude1)(support_brace_long-2),** **SolidBody 1(Boss-Extrude1)(support_brace_short-1),** **SolidBody 1(Boss-Extrude1)(support_brace_short-2),** **SolidBody 1(Boss-Extrude1)(support_brace_short-4),** **SolidBody 1(Boss-Extrude1)(support_brace_short-5),** **SolidBody 1(Boss-Extrude1)(support_post-1)**
Curve Data:N/A		

Loads and Fixtures

Fixture name	Fixture Image	Fixture Details
Fixed-1		**Entities:** 3 face(s) **Type:** Fixed Geometry

Resultant Forces

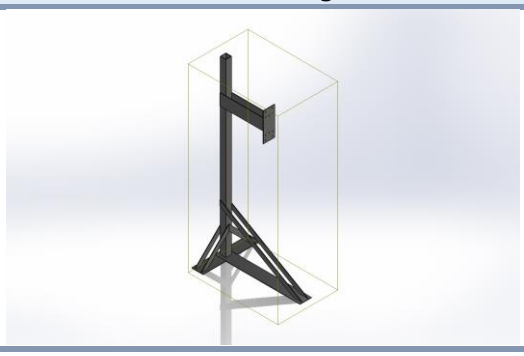
Components	X	Y	Z	Resultant
Reaction force(N)	**4.08361**	**4447.7**	**-5.36478**	**4447.7**
Reaction Moment(N·m)	**0**	**0**	**0**	**0**

Load name	Load Image	Load Details
Force-1		**Entities:** 1 face(s) **Type:** Apply normal force **Value:** 1000 lbf

Connector Definitions

No Data

Contact Information

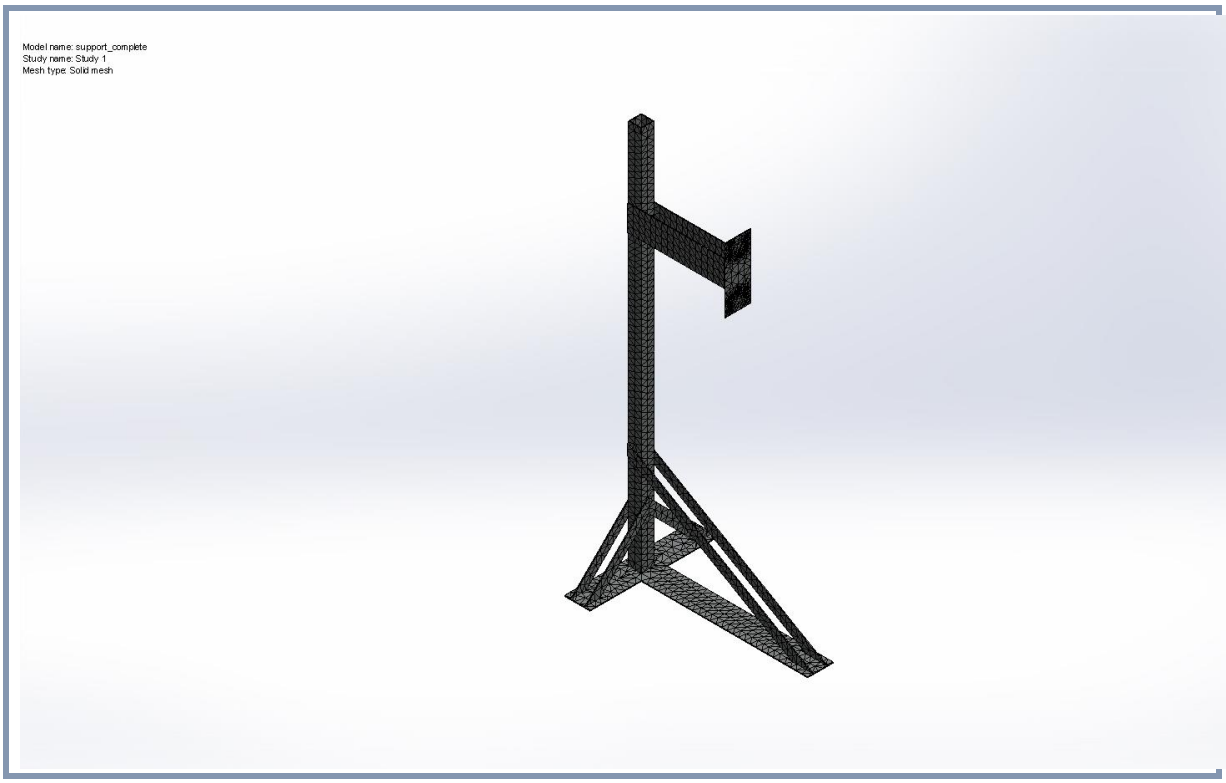
Contact	Contact Image	Contact Properties
Global Contact		**Type:** **Welded** **Components:** **1 component(s)** **Options:** **Compatible mesh**

Mesh Information

Mesh type	Solid Mesh
Mesher Used:	Standard mesh
Automatic Transition:	Off
Include Mesh Auto Loops:	Off
Jacobian points	4 Points
Element Size	1.793 in
Tolerance	0.0896502 in
Mesh Quality	High
Remesh failed parts with incompatible mesh	Off

Mesh Information - Details

Total Nodes	19951
Total Elements	9399
Maximum Aspect Ratio	53.153
% of elements with Aspect Ratio < 3	12.7
% of elements with Aspect Ratio > 10	2.47
% of distorted elements(Jacobian)	0
Time to complete mesh(hh;mm;ss):	00:00:05
Computer name:	DANIEL-PC



Sensor Details

No Data

Resultant Forces

Reaction Forces

Selection set	Units	Sum X	Sum Y	Sum Z	Resultant
Entire Model	N	4.08361	4447.7	-5.36478	4447.7

Reaction Moments

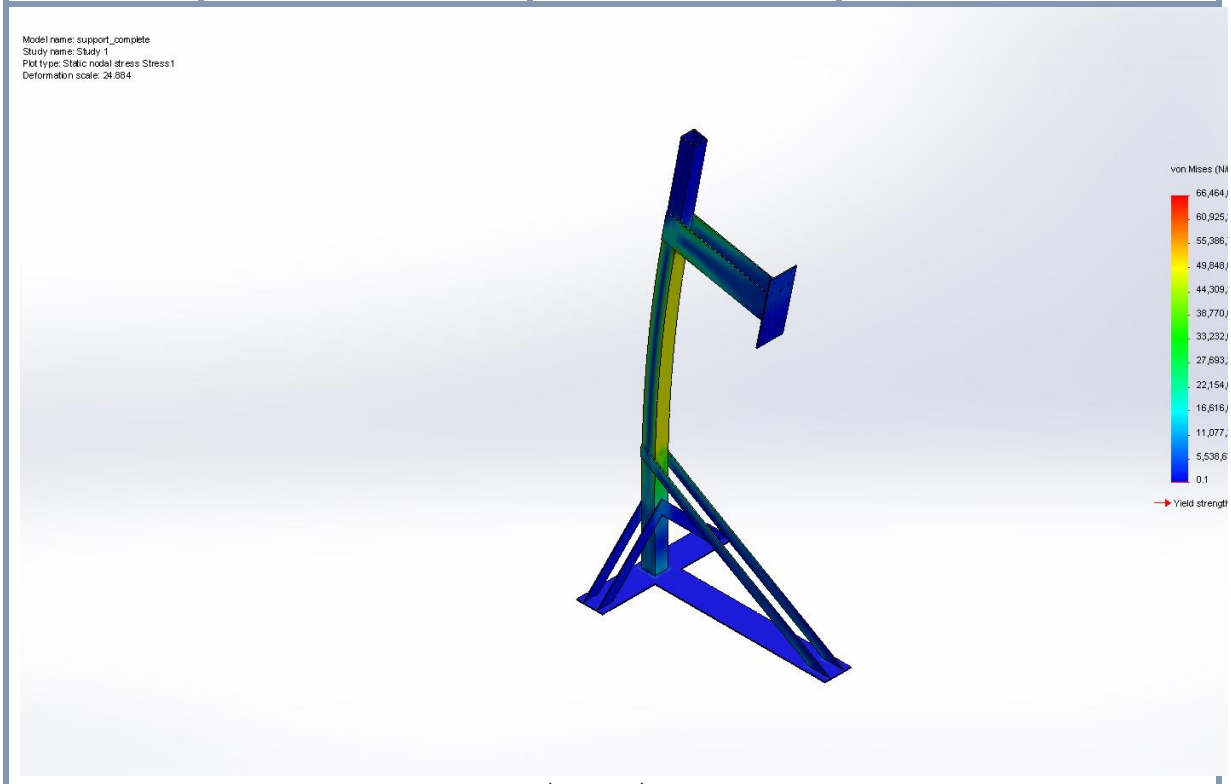
Selection set	Units	Sum X	Sum Y	Sum Z	Resultant
Entire Model	N·m	0	0	0	0

Beams

No Data

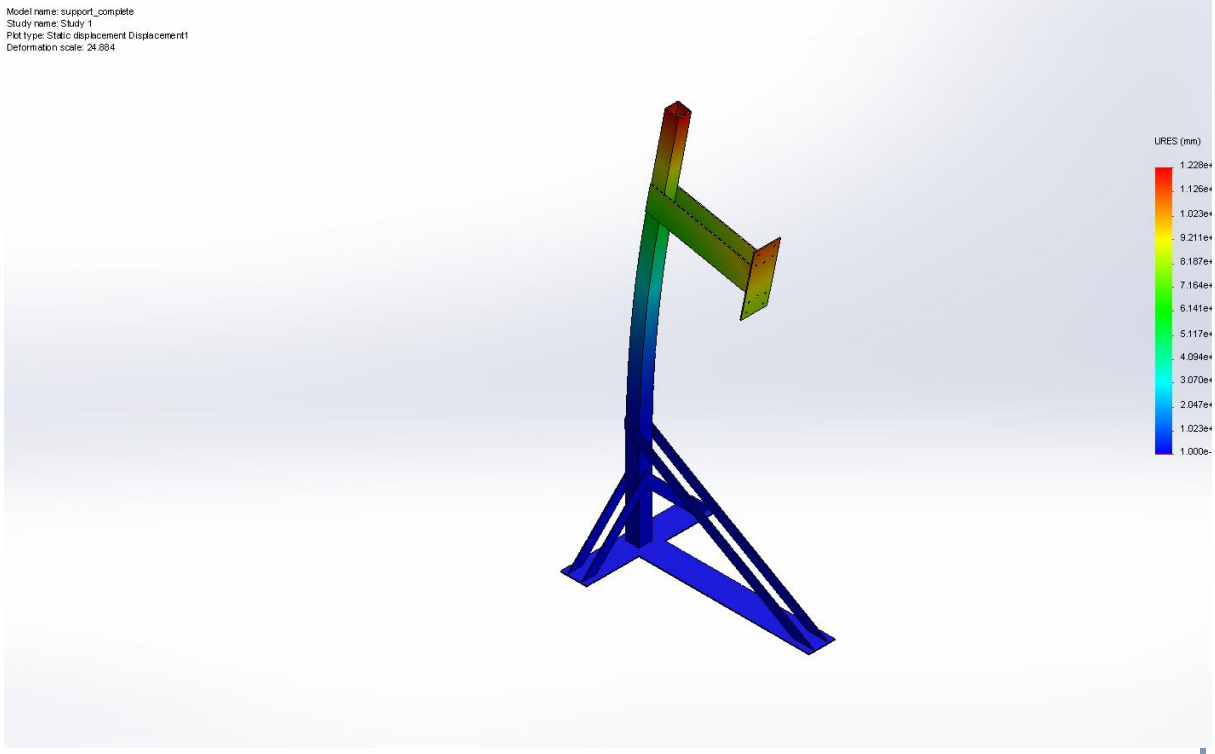
Study Results

Name	Type	Min	Max
Stress1	VON: von Mises Stress	0.129884 N/m^2 Node: 7238	6.6464e+007 N/m^2 Node: 6717



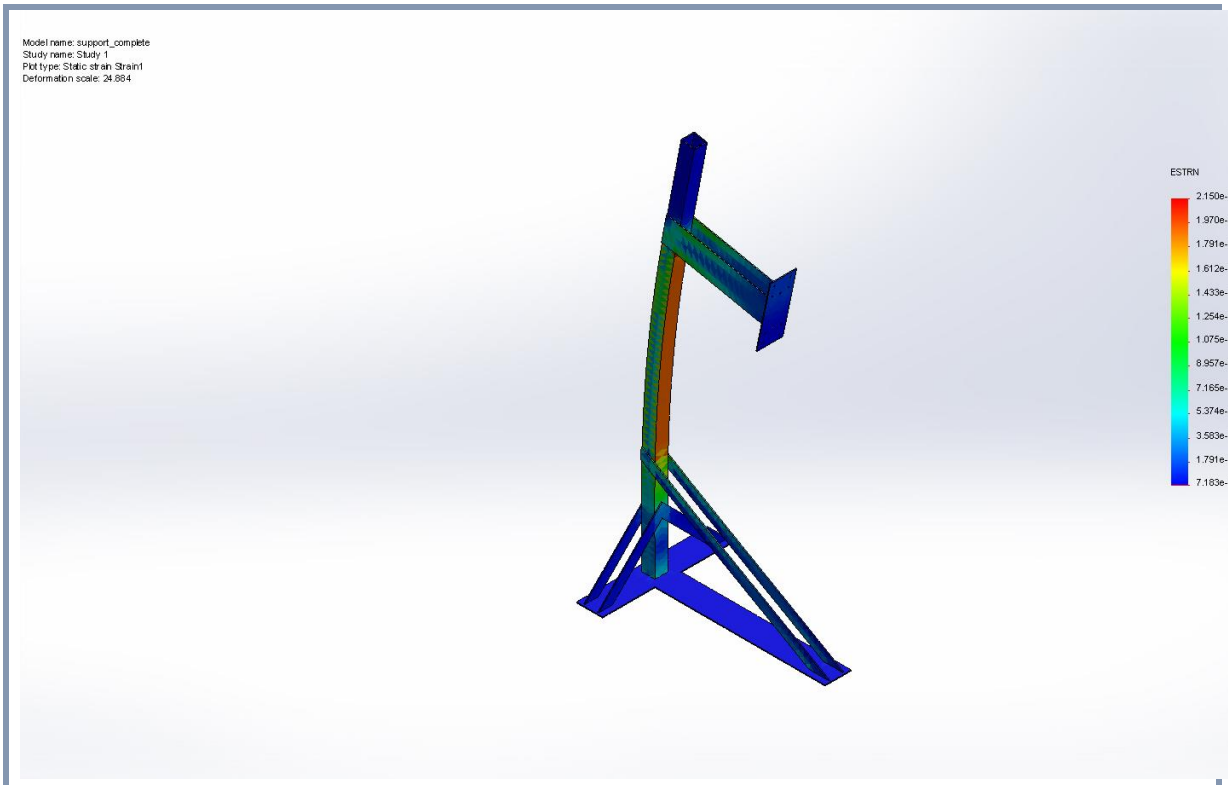
support_complete-Study 1-Stress-Stress1

Name	Type	Min	Max
Displacement1	URES: Resultant Displacement	0 mm Node: 6923	12.2811 mm Node: 13618



support_complete-Study 1-Displacement-Displacement1

Name	Type	Min	Max
Strain1	ESTRN: Equivalent Strain	7.18254e-013 Element: 3911	0.000214963 Element: 8751

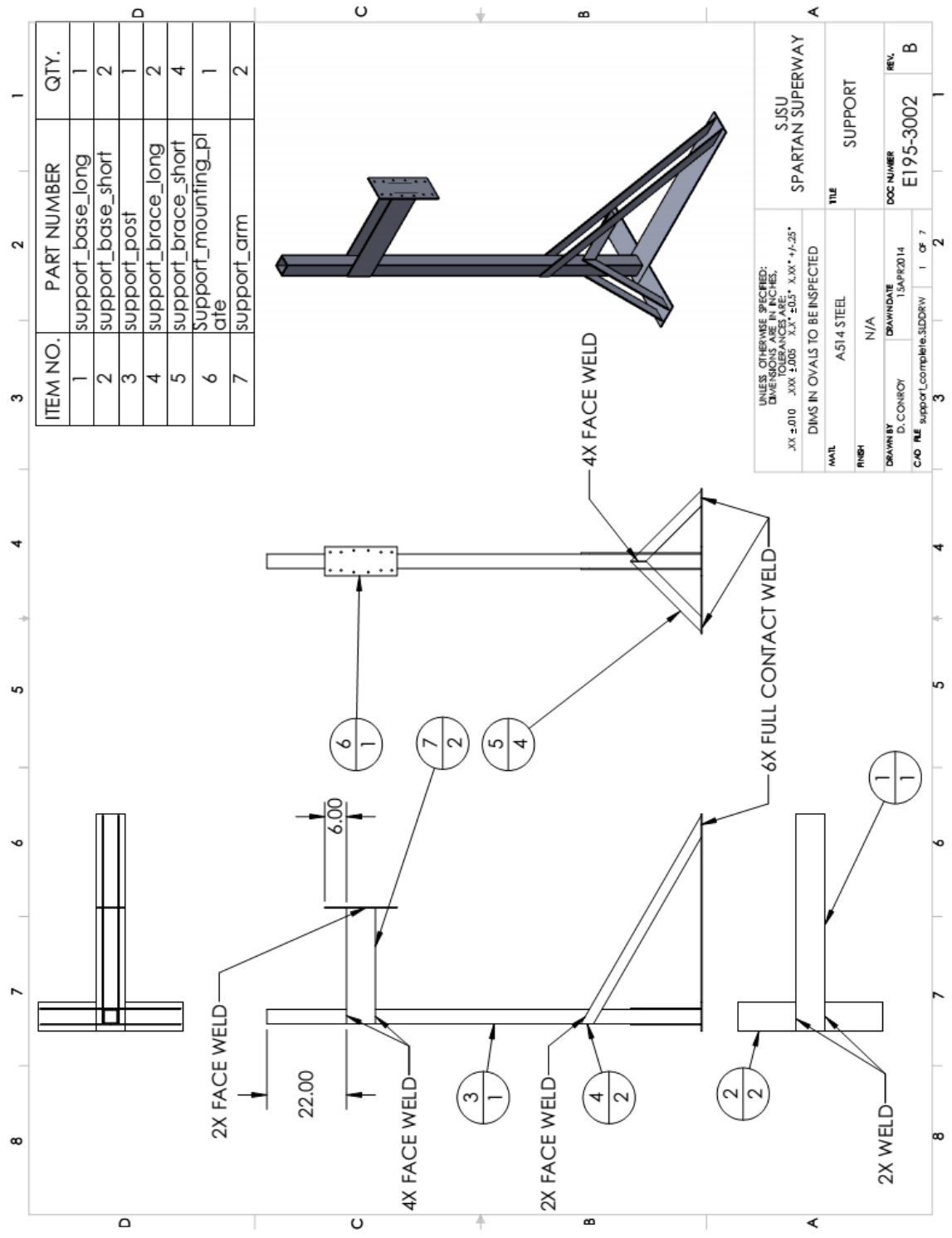


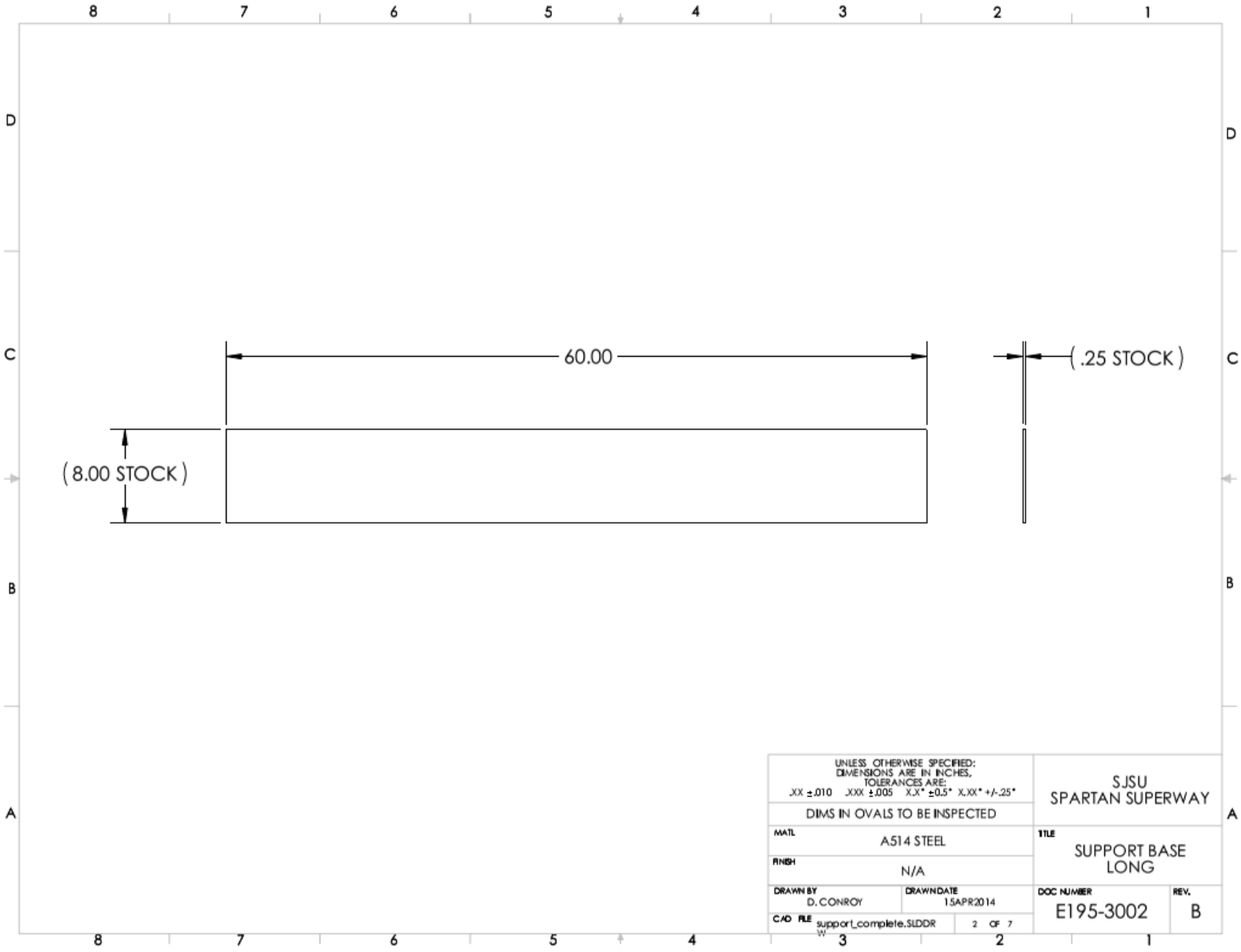
support_complete-Study 1-Strain-Strain1

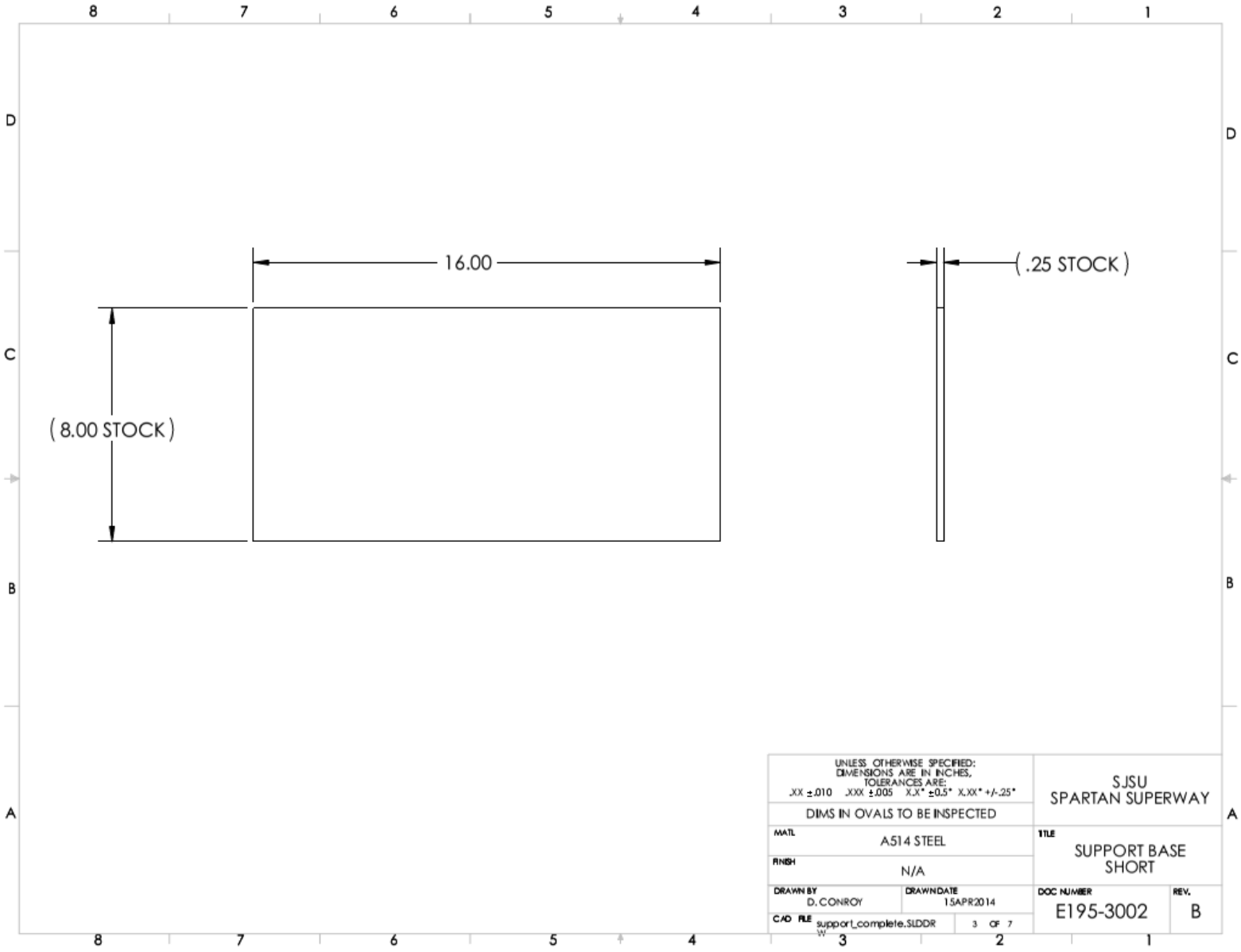
Name	Type
Displacement1{1}	Deformed Shape

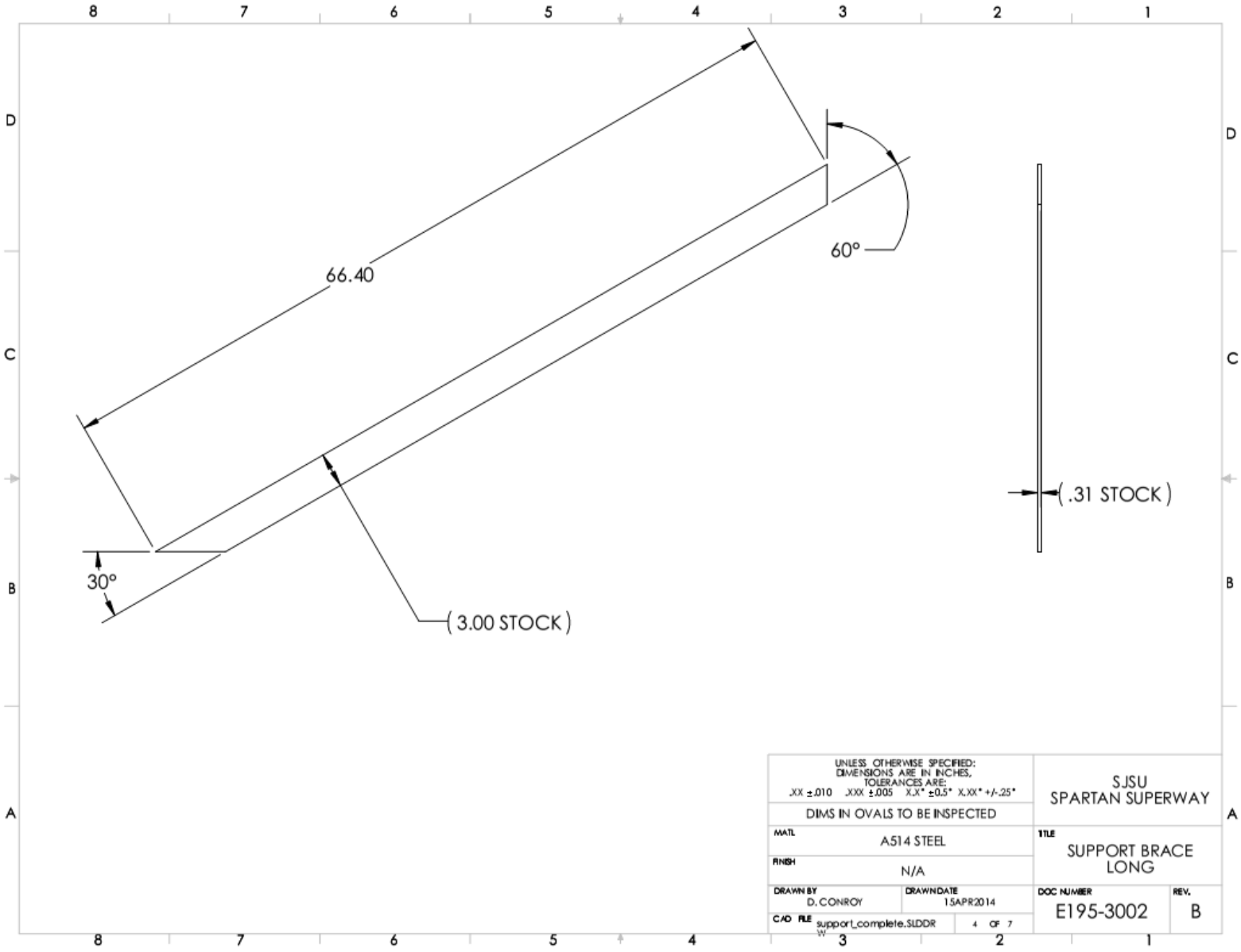
support_complete-Study 1-Displacement-Displacement1{1}

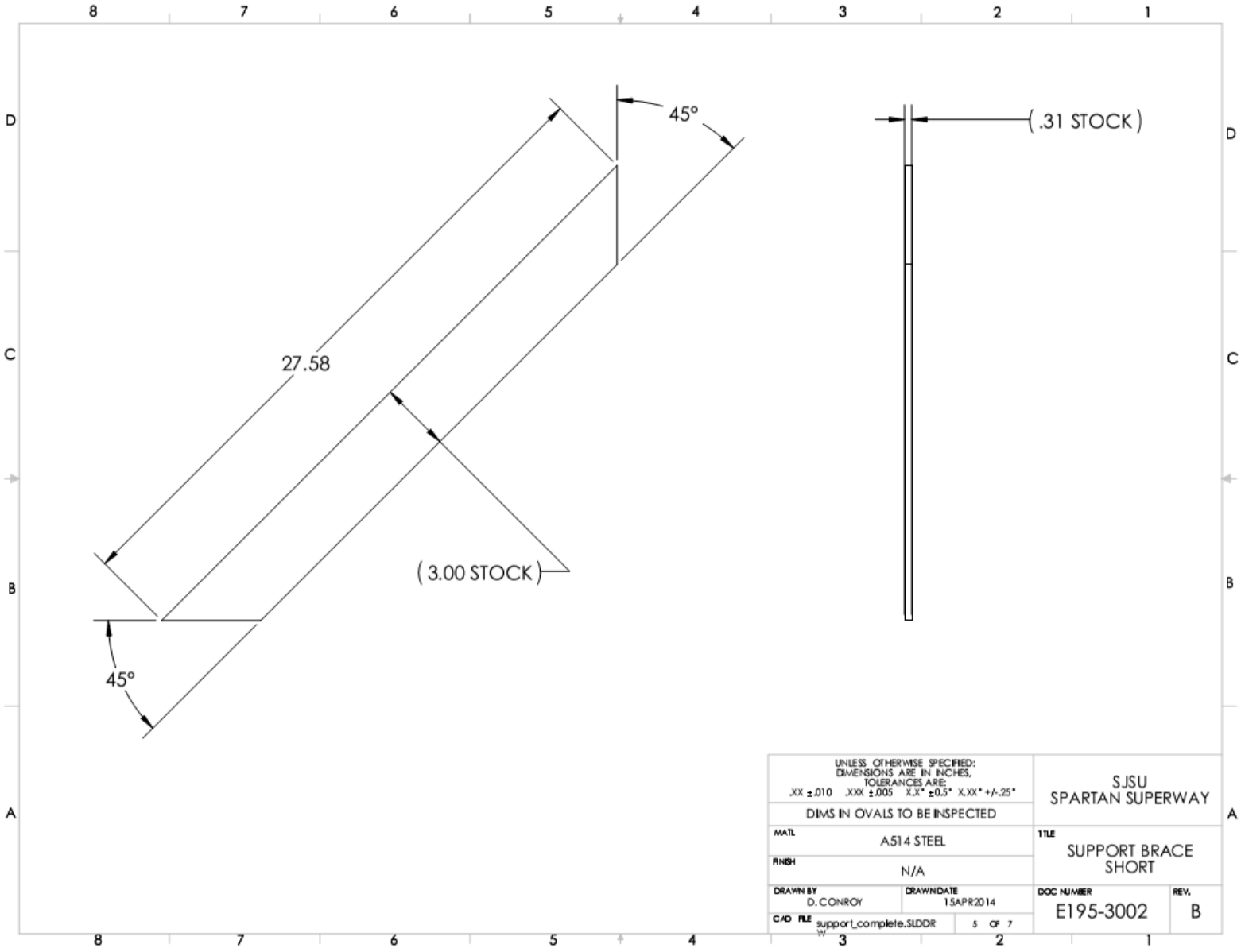
Appendix G: Guideway Part Drawings

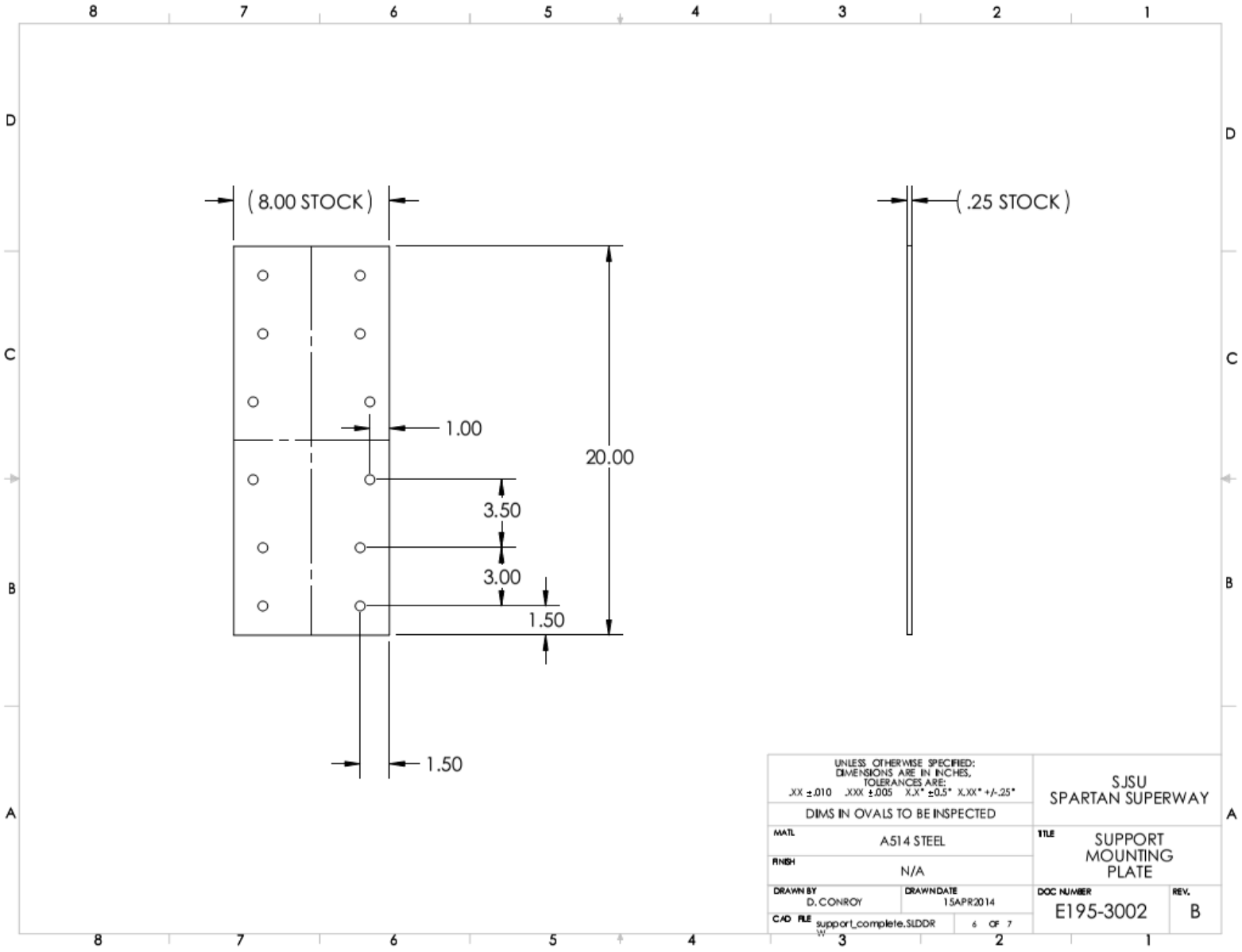


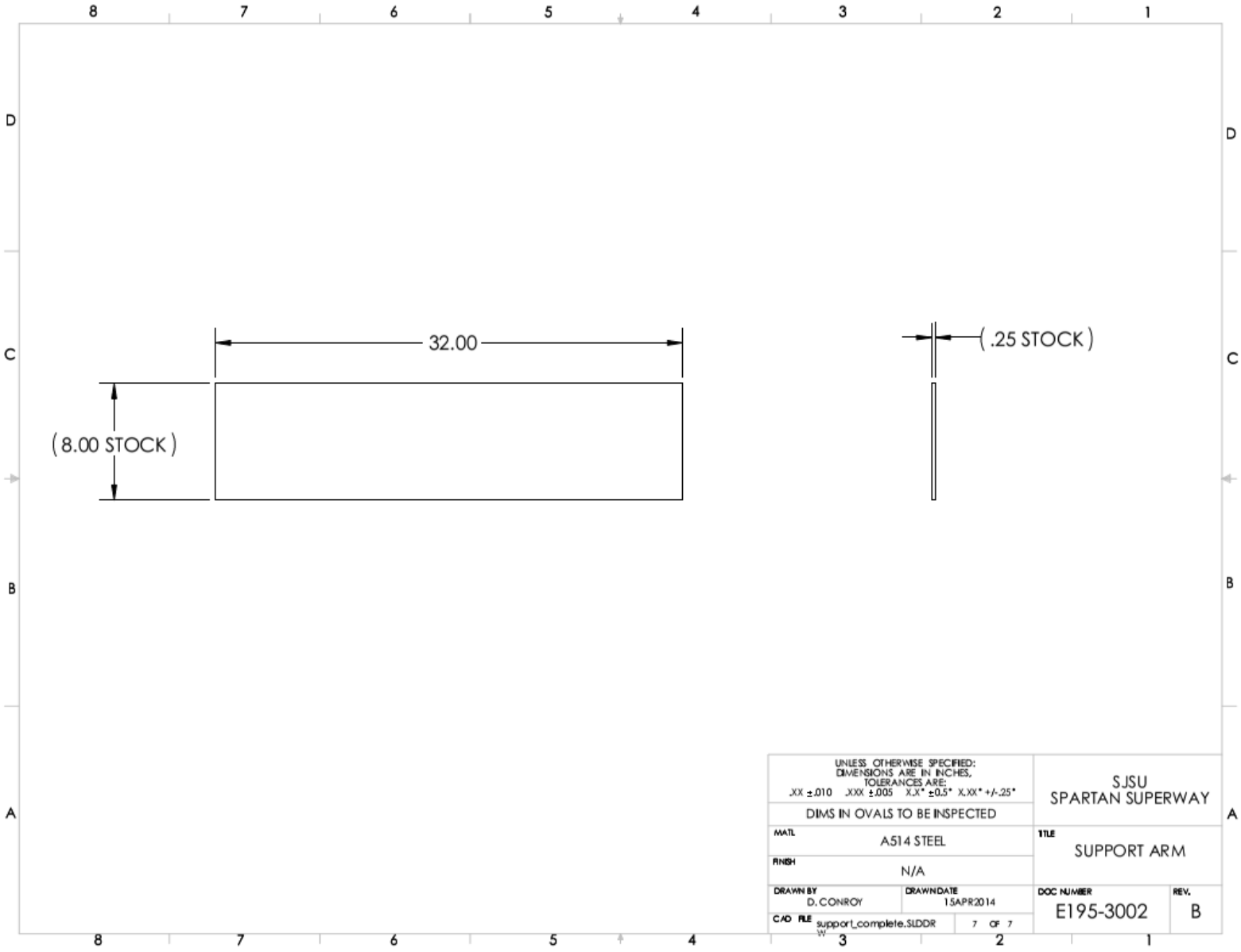

8
7
6
5
4
3
2
1
D
C
B
A
60.00
(.25 STOCK)
(8.00 STOCK)
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
A514 STEEL
FINISH
N/A
DRAWN BY
D. CONROY
DRAWN DATE
15APR2014
CAD FILE
support_complete.SLDDR
2 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
SUPPORT BASE
LONG
DOC NUMBER
E195-3002
REV.
B


16.00
(.25 STOCK)
(8.00 STOCK)
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL A514 STEEL
FINISH N/A
DRAWN BY D. CONROY
DRAWN DATE 15APR2014
CAD FILE support_complete.SLDDR
3 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
SUPPORT BASE
SHORT
DOC NUMBER E195-3002
REV. B

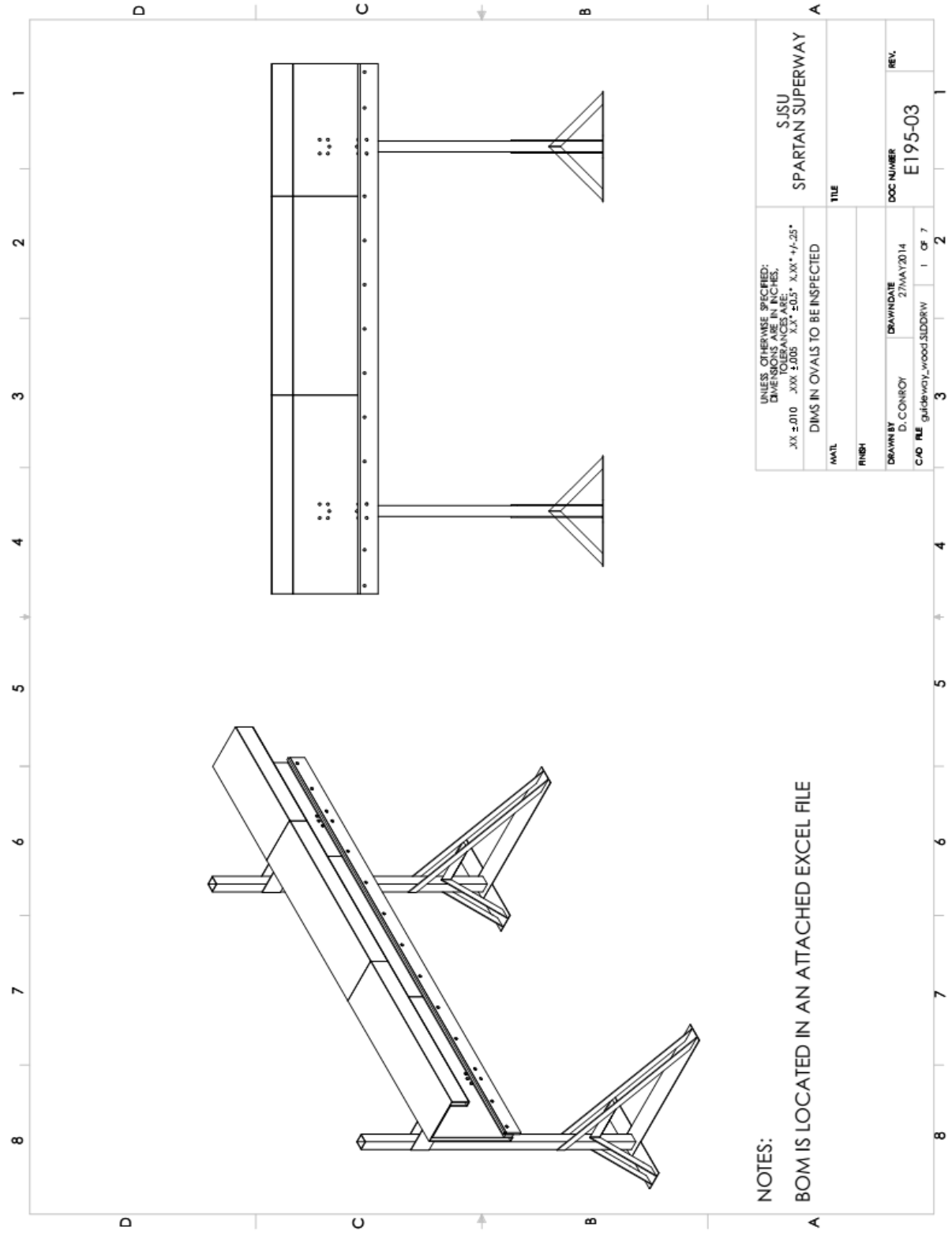

66.40
60°
30°
(3.00 STOCK)
(.31 STOCK)
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL A514 STEEL
FINISH N/A
DRAWN BY D. CONROY
DRAWN DATE 15APR2014
CAD FILE support_complete.SLDDR
4 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
SUPPORT BRACE
LONG
DOC NUMBER E195-3002
REV. B

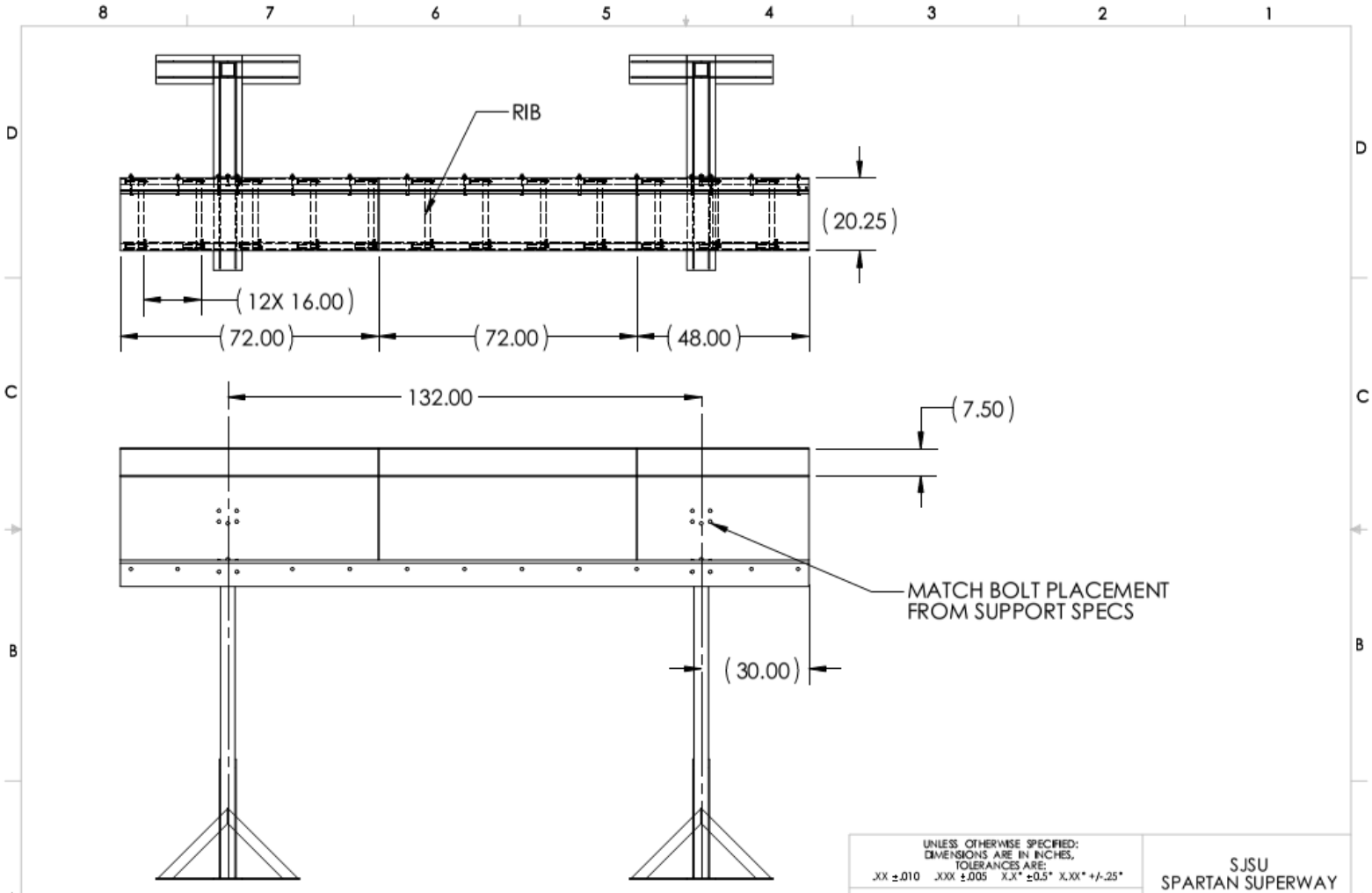

8
7
6
5
4
3
2
1
D
C
B
A
45°
27.58
(3.00 STOCK)
45°
(.31 STOCK)
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
A514 STEEL
FINISH
N/A
DRAWN BY
D. CONROY
DRAWN DATE
15APR2014
CAD FILE
support_complete.SLDDR
5 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
SUPPORT BRACE
SHORT
DOC NUMBER
E195-3002
REV.
B


8
7
6
5
4
3
2
1
D
C
B
A
(8.00 STOCK)
(.25 STOCK)
1.00
20.00
3.50
3.00
1.50
1.50
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
A514 STEEL
FINISH
N/A
DRAWN BY
D. CONROY
DRAWN DATE
15APR2014
CAD FILE
support_complete.SLDDR
6 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
SUPPORT
MOUNTING
PLATE
DOC NUMBER
E195-3002
REV.
B


8
7
6
5
4
3
2
1
D
C
B
A
32.00
(8.00 STOCK)
(.25 STOCK)
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
A514 STEEL
FINISH
N/A
DRAWN BY
D. CONROY
DRAWN DATE
15APR2014
CAD FILE
support_complete.SLDDR
7 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
SUPPORT ARM
DOC NUMBER
E195-3002
REV.
B

Appendix H: Guideway Assembly Drawings

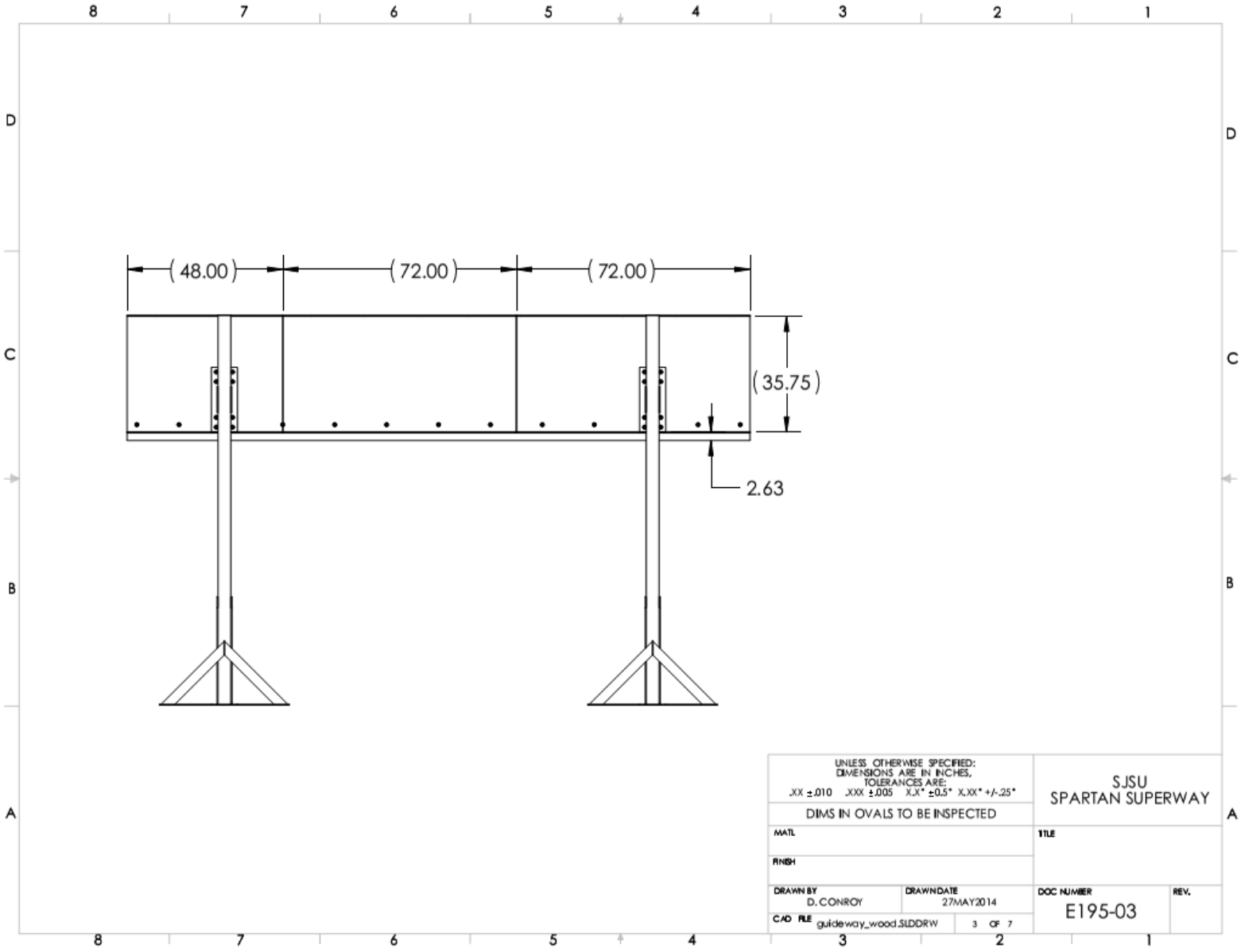


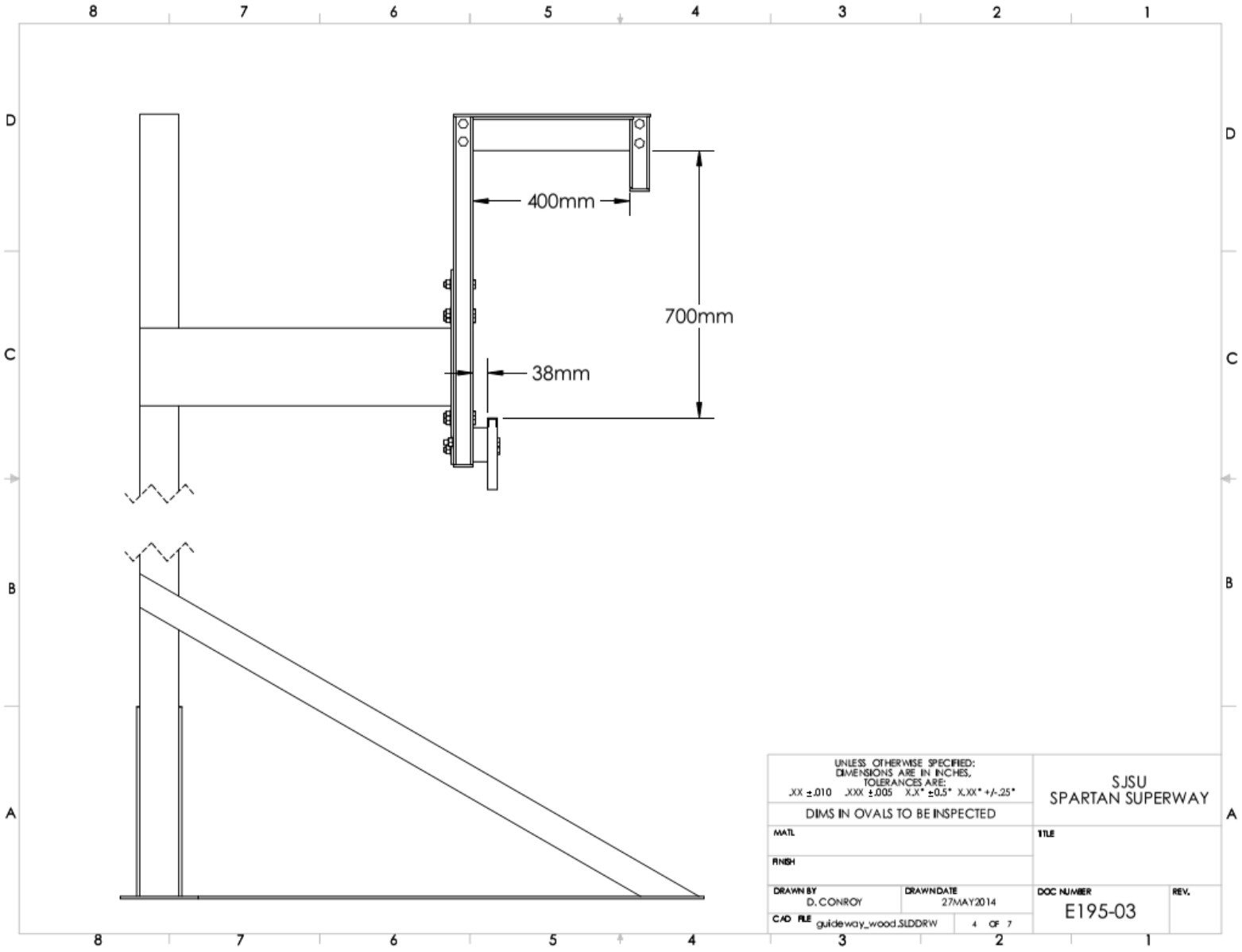


NOTES:

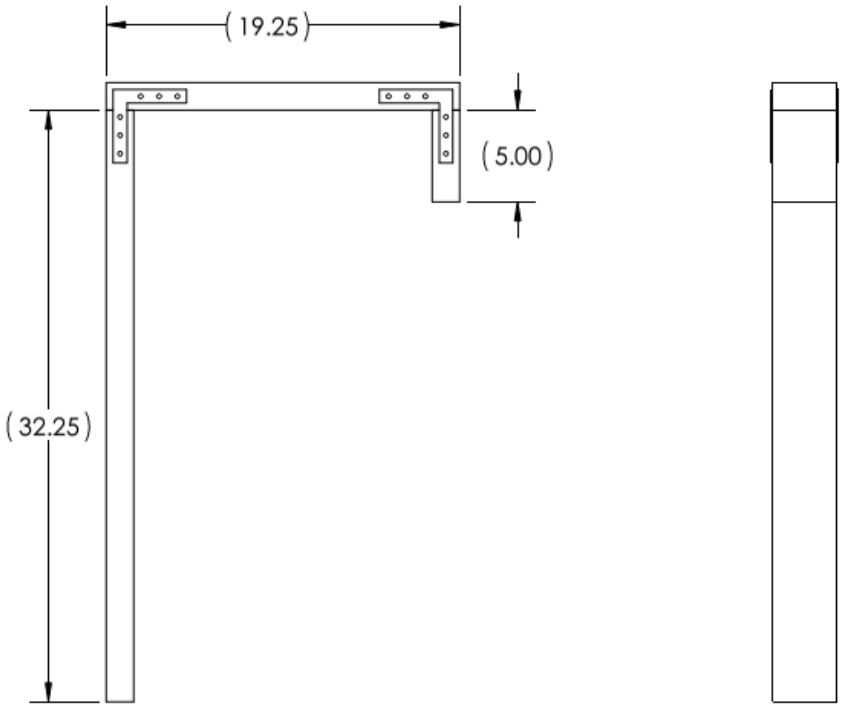
1. PLATES ARE 3/4" THICK PLYWOOD
2. RIBS ARE PLACED AT 16" ON CENTER
3. DRILL ALL THRU HOLES ONCE GUIDEWAY HAS BEEN ASSEMBLED WITH NAILS AND GLUE

UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES, TOLERANCES ARE: .XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°	SJSU SPARTAN SUPERWAY
DIMS IN OVALS TO BE INSPECTED	
MATL	TITLE
FINISH	
DRAWN BY D. CONROY / DRAWN DATE 27MAY2014	DOC NUMBER E195-03 / REV.
CAD FILE guideway_wood.SLDDRW / 2 OF 7	


8
7
6
5
4
3
2
1
D
C
B
A
(48.00)
(72.00)
(72.00)
(35.75)
2.63
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
FINISH
DRAWN BY
D. CONROY
DRAWN DATE
27MAY2014
CAD FILE guideway_wood.SLDDRW
3 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
DOC NUMBER
E195-03
REV.


400mm
700mm
38mm
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
FINISH
DRAWN BY
D. CONROY
DRAWN DATE
27MAY2014
CAD FILE guideway_wood.SLDDRW
4 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
DOC NUMBER
E195-03
REV.

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	rib_inner	2X4, 32.25" LENGTH	1
2	rib_mid	2X4, 19.25" LENGTH	1
3	rib_outer	2X4, 5.00" LENGTH	1
4	rib_brace	4" ANGLE BRACKET	4



NOTES:

MAKE FROM 2X4 WOOD BEAMS AND ANGLE BRACKETS ASSEMBLED 16" ON CENTER

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°

DIMS IN OVALS TO BE INSPECTED

SJSU
SPARTAN SUPERWAY

MATL

TITLE

FINISH

DRAWN BY: D. CONROY

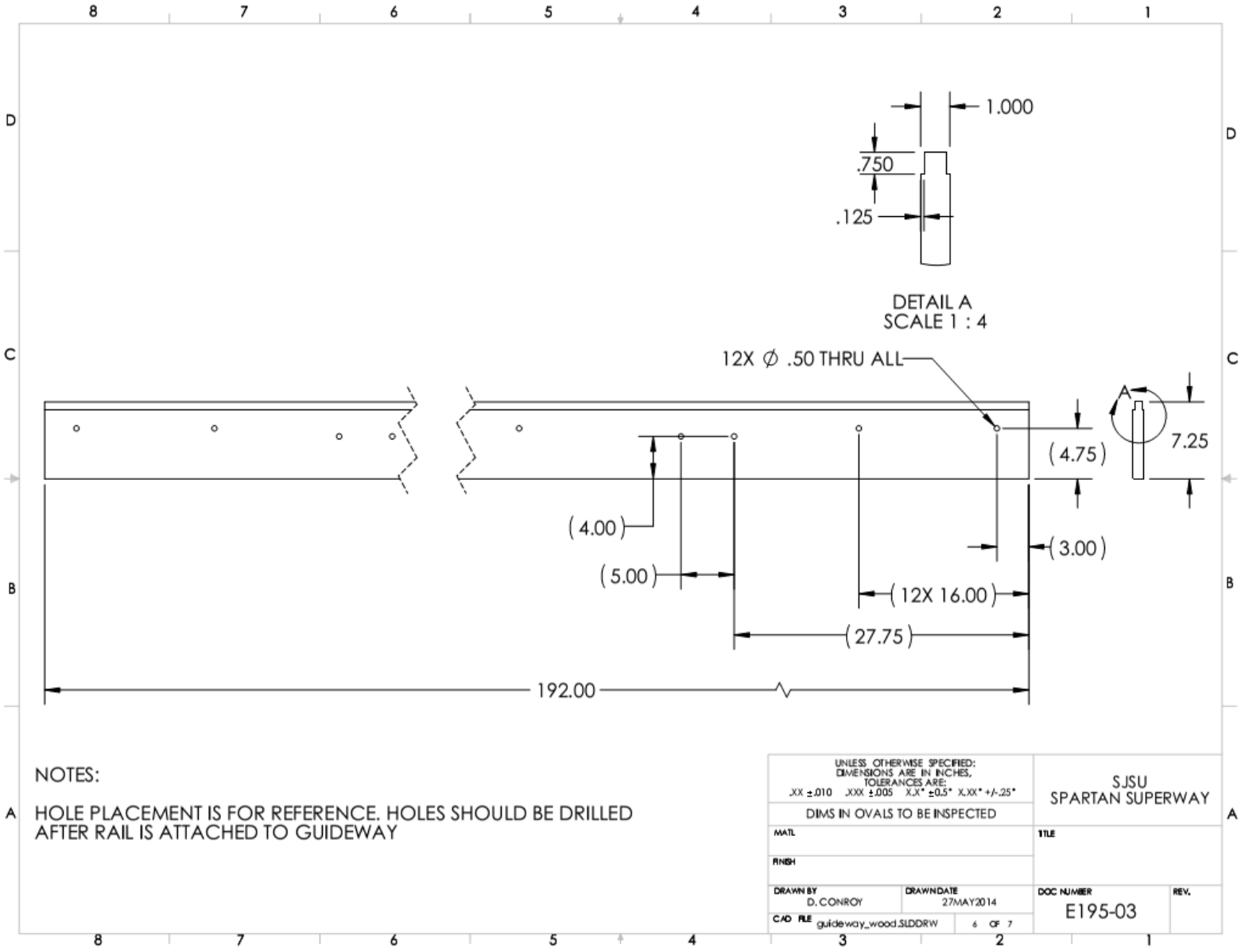
DRAWN DATE: 27MAY2014

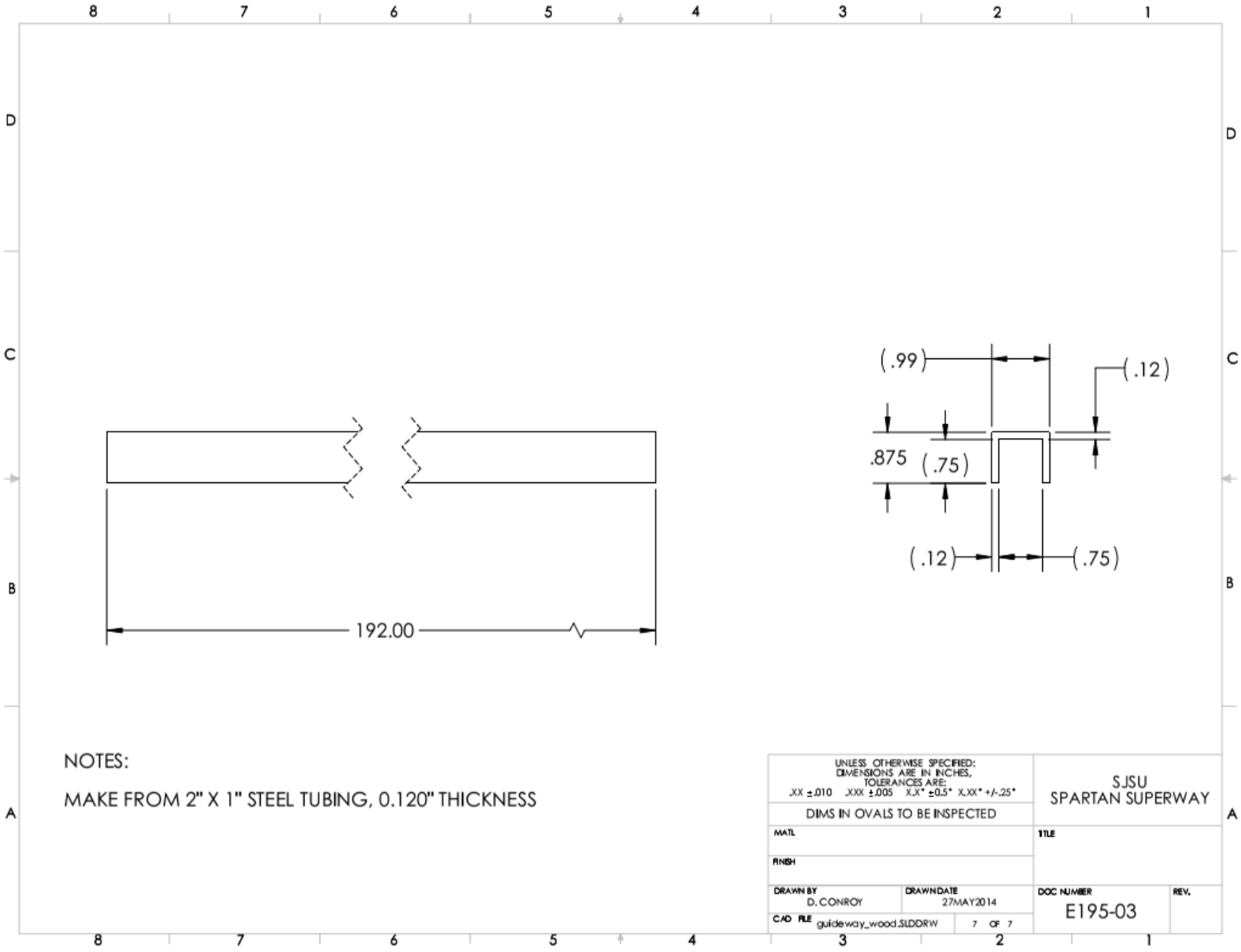
DOC NUMBER: E195-03

REV.

CAD FILE: guideway_wood.SLDDRW

5 OF 7


1.000
.750
.125
DETAIL A
SCALE 1 : 4
12X ∅ .50 THRU ALL
A
7.25
(4.75)
(4.00)
(5.00)
(3.00)
(12X 16.00)
(27.75)
192.00
NOTES:
HOLE PLACEMENT IS FOR REFERENCE. HOLES SHOULD BE DRILLED AFTER RAIL IS ATTACHED TO GUIDEWAY
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
FINISH
DRAWN BY
D. CONROY
DRAWN DATE
27MAY2014
CAD FILE guideway_wood.SLDDRW
6 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
DOC NUMBER
E195-03
REV.


8
7
6
5
4
3
2
1
D
C
B
A
192.00
(.99)
(.12)
.875
(.75)
(.12)
(.75)
NOTES:
MAKE FROM 2" X 1" STEEL TUBING, 0.120" THICKNESS
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES,
TOLERANCES ARE:
.XX ±.010 .XXX ±.005 X.X° ±0.5° X.XX° +/-.25°
DIMS IN OVALS TO BE INSPECTED
MATL
FINISH
DRAWN BY
D. CONROY
DRAWN DATE
27MAY2014
CAD FILE guideway_wood.SLDDRW
7 OF 7
SJSU
SPARTAN SUPERWAY
TITLE
DOC NUMBER
E195-03
REV.

Appendix I: Bogie Design Requirements

Bogie Design Goals and Objectives

1. The bogie must be able to suspend the pod cabin.
   1.1. The bogie should be able to suspend the cabin's weight in both dynamic and static conditions.
      1.1.1. The bogie must be able to suspend the mass of the cabin during cornering at specified speeds and radii.
      1.1.2. Bogie must be able to suspend the mass of the cabin over crests and dips in the guideway.
   1.2. The bogie must be able to mount to the specified cabin top surface and frame design.

2. The bogie must be able to travel and navigate through the Superway guideway network.
   2.1. The bogie must be able to interface with power conduits placed along the guideway.
      2.1.1. Bogie must interface with power conduits during all operations, including guideway switching.
      2.1.2. Power from conduit must be transmitted to cabin.
   2.2. The bogie must have variable acceleration and speed capabilities up to its performance targets.
   2.3. The bogie must be able to navigate guideway corners.
      2.3.1. Bogie must be able to navigate the tightest specified/possible radii.
      2.3.2. Bogie must smoothly ride around corners (no crashing into guideway walls).
   2.4. The bogie will be able to switch between guideways.
      2.4.1. Bogie will not be jolted when switching guideways.
      2.4.2. Switch will not be capable of making an "indecisive guideway selection".
   2.5. The bogie must be able to climb and descend maximum specified grades.
   2.6. The bogie must be able to coordinate with station procedures.
      2.6.1. Bogie must operate at a reduced speed and stop and accordingly.
      2.6.2. Bogie must be able to hold steady during boarding and offloading at stations.
   2.7. The bogie must provide a comfortable ride.
      2.7.1. Vibration must be dampened/minimized within the guideway interface.
      2.7.2. Vibration must be dampened/minimized at the cabin interface.
      2.7.3. Bogie must allow the pod to bank during cornering.

3. The bogie must be able to perform emergency operations.
   3.1. The bogie must be able to perform an emergency stop when prompted.
      3.1.1. Bogie must have an electrical cut off/safety switch.
      3.1.2. Bogie must have a manual emergency braking system.
   3.2. The bogie must be able to interface with emergency power or tow vehicle.
   3.3. The bogie must maintain basic functionality during a power loss.
      3.3.1. The guideway switching mechanism state must be unaffected by a power loss.
      3.3.2. The manual emergency braking system must function during a power loss.

4. The bogie must be optimized for the operating efficiency.
   4.1. The bogie must be able to consume as little as power to meet speed and acceleration targets.
   4.2. The bogie must be able to recapture braking energy during normal operation.

5. The bogie must meet reliability targets
   5.1. The bogie drivetrain must have a long life cycle .
   5.2. The bogie chassis and support wheels and bearings must have a long life cycle.
   5.3. The guideway switch must be designed for infinite life.
   5.4. The bogie must be able to dissipate heat effectively from the drive system.
   5.5. The power interface must have a long life cycle.
   5.6. The Sway damper must have a long life cycle.
   5.7. The Emergency Brake must prove reliable in testing.

Appendix J: Bogie Design Specifications

This document was created to outline the specifications of the proposed bogie design under the Superway Autonomous Transportation Network (ATN) project for the 2013-2014 SJSU Senior Design term. This document contains projected performance and dimension specifications of the bogie design based on sourced hardware, simulation, and calculation. Subjects addressed are material usage, motor specification, wheel and chassis dimensions, hydraulic traction adjustment, the track switching mechanism, power interface, and suspension.

1. **Chassis:**
   - **1.1. Structure** *[1][3.2]*
     - 1.1.1. **Mounting Points:** The Structural Frame of the Bogie will have mounting points for the propulsion system, the rolling system, the switching system, and all other bogie components.
     - 1.1.2. **Strength**: The Structural Frame of the Bogie will be strong enough to support the weight of the cabin suspended from a minimum of 4 mounting points and supported by the support wheels. It will be strong enough to support the weight of the cabin during cornering forces of up to 1g.
     - 1.1.3. **Tow Points**: The Structural Frame of the Bogie will have tow points for the tow bogie to attach to.
   - **1.2. Rolling System** *[2.3][2.7.1]*
     - 1.2.1. **Configuration**: The support wheels will be horizontal and vertical rollers which run upon the lower flanges as well as center of the guideway profile, enabling them to suspend and stabilize the bogie within the track.
     - 1.2.2. **Safety:** The rollers will be non-pneumatic wheels in order to prevent flat-tire derailments.
     - 1.2.3. **Cornering ability:** The Rolling System will be able to navigate a guideway corner with an 8 meter radius.
     - 1.2.4. **Traction**: The rollers will be made from an elastic material in order to improve traction and minimize vibration.
2. **Propulsion:**
   - **2.1. Powertrain** *[2.2][2.6.1][4.2][4.3]*
     - 2.1.1. **Speed:** The propulsion system will allow the pod to reach speeds up to 26 m/s.
     - 2.1.2. **Acceleration:** The propulsion system will allow the pod to accelerate and decelerate at $2 m/s^2$ on horizontal sections of guideway.
     - 2.1.3. **Climbing Speed:** The propulsion system will allow the pod to maintain speeds of up to 15 m/s up a grade of 10%.
     - 2.1.4. **Variable Acceleration:** The propulsion system will have variable acceleration/deceleration and speed control.
     - 2.1.5. **Emergency Stops:** The propulsion system will be able to perform emergency stops or be able to be overridden in emergency situations.

2.1.6. **Regenerative Braking:** The powertrain will be able to capture a minimum of 60% of the braking energy and return it to the system in some useful form.

3. **Switching System** *[2.4][3.3.1]*
   3.1. Decisive Switch: The system will be mechanically incapable of engaging both sides of the switching flange at once.
   3.2. Passive Support: The system will maintain its position during a power failure.

4. **Swing System** *[1.1.1][1.2][2.7.2][2.7.3]*
   4.1. Hinge Support
      4.1.1. **Strength:** The hinge support will be strong enough to support the cabin during cornering of up to 1g.
      4.1.2. **Cabin Mounts:** The hinge support will be able to mount to the cabin through some sort of elastic bushing to minimize vibration.
   4.2. Sway Control
      4.2.1. **Wind Resistance**: The sway control system will minimize rocking due to gusts of wind
      4.2.2. **Damping:** The sway control system will properly dampen the swinging of the cabin to prevent it from swinging when on a straight section of guideway.
      4.2.3. **Maximum Swing angle:** The sway control system will limit the swinging of the cabin to an angle of 30 degrees off center

5. **Power Interface** *[2.1]*
   5.1. Power: The Power interface will deliver 3-phase power at 460 Volts.

6. **Emergency Mechanisms**
   6.1. Emergency Brakes *[3.1.2][3.3.2]*
      6.1.1. **Manual Brakes:** A manual activated braking system will enable to bogie to stop at a rate of no more than 0.6g even during a total failure of all electrical systems
   6.2. Mechanical Cut Off *[3.1.1]*
      6.2.1. **Manual Cut Off:** A mechanical cut off will engage when the manual brakes are applied.
      6.2.2. **Remote Cut Off:** A mechanical cut off will be able to be engaged remotely.

Appendix K: Bogie Static Analysis

Full Bogie Static Analysis RPT. file

```
------------------------------------------------------------

Creo Simulate Structure Version P-10-17:spg

Summary for Design Study "Analysis1"

Wed May 07, 2014   12:33:24

------------------------------------------------------------


Run Settings

  Memory allocation for block solver: 3500.0


Parallel Processing Status

  Parallel task limit for current run:          8

  Parallel task limit for current platform:     64

  Number of processors detected automatically:    8

Checking the model before creating elements...

These checks take into account the fact that AutoGEM will

automatically create elements in volumes with material

properties, on surfaces with shell properties, and on curves

with beam section properties.

  Generate elements automatically.

Checking the model after creating elements...

No errors were found in the model.

Creo Simulate Structure Model Summary


  Principal System of Units: Inch Pound Second (IPS)


  Length:         in
```

Force: lbf

Time: sec

Temperature: F

Model Type: Three Dimensional

Points: 8958

Edges: 45142

Faces: 63863

Springs: 0

Masses: 0

Beams: 0

Shells: 0

Solids: 27731

Elements: 27731

Standard Design Study

Static Analysis "Analysis1":

Convergence Method: Multiple-Pass Adaptive

Plotting Grid: 4

Convergence Loop Log: (12:34:09)

>> Pass 1 <<

```
    Calculating Element Equations          (12:34:10)
      Total Number of Equations:   26508
      Maximum Edge Order:            1
    Solving Equations                    (12:34:10)
    Post-Processing Solution             (12:34:10)
    Calculating Disp and Stress Results     (12:34:11)
    Checking Convergence                 (12:34:30)
      Elements Not Converged:      27731
      Edges Not Converged:         45142
      Local Disp/Energy Index:    100.0%
      Global RMS Stress Index:    100.0%
    Resource Check                       (12:34:31)
      Elapsed Time     (sec):      66.92
      CPU Time         (sec):      52.67
      Memory Usage      (kb):    8674344
      Wrk Dir Dsk Usage (kb):     221184


>> Pass  2 <<
    Calculating Element Equations          (12:34:31)
      Total Number of Equations:  161202
      Maximum Edge Order:            2
    Solving Equations                    (12:34:32)
    Post-Processing Solution             (12:34:35)
    Calculating Disp and Stress Results     (12:34:38)
    Checking Convergence                 (12:34:55)
      Elements Not Converged:       9746
      Edges Not Converged:         14967
      Local Disp/Energy Index:    100.0%
      Global RMS Stress Index:     87.3%
    Resource Check                       (12:34:56)
```

```
        Elapsed Time    (sec):     92.50
        CPU Time        (sec):     67.13
        Memory Usage    (kb):    8674344
        Wrk Dir Dsk Usage (kb):    221184

>> Pass  3 <<
     Calculating Element Equations         (12:34:56)
       Total Number of Equations:  516659
       Maximum Edge Order:           4
     Solving Equations                   (12:35:02)
     Post-Processing Solution            (12:35:12)
     Calculating Disp and Stress Results   (12:35:19)
     Checking Convergence                (12:35:35)
       Elements Not Converged:      6900
       Edges Not Converged:         1592
       Local Disp/Energy Index:   100.0%
       Global RMS Stress Index:    44.4%
     Resource Check                      (12:35:36)
       Elapsed Time    (sec):    132.06
       CPU Time        (sec):    113.21
       Memory Usage    (kb):    8674344
       Wrk Dir Dsk Usage (kb):    606208

>> Pass  4 <<
     Calculating Element Equations         (12:35:36)
       Total Number of Equations:  839769
       Maximum Edge Order:           6
     Solving Equations                   (12:35:47)
     Post-Processing Solution            (12:36:06)
     Calculating Disp and Stress Results   (12:36:15)
```

```
    Checking Convergence              (12:36:33)
      Elements Not Converged:      3900
      Edges Not Converged:         0
      Local Disp/Energy Index:   100.0%
      Global RMS Stress Index:    31.0%
    Resource Check                    (12:36:35)
      Elapsed Time    (sec):    191.58
      CPU Time        (sec):    215.33
      Memory Usage    (kb):   8674344
      Wrk Dir Dsk Usage (kb):   1098752


>> Pass  5 <<
    Calculating Element Equations        (12:36:36)
      Total Number of Equations: 1238710
      Maximum Edge Order:          7
    Solving Equations                 (12:37:32)
    Post-Processing Solution          (12:38:53)
    Calculating Disp and Stress Results   (12:39:10)
    Checking Convergence              (12:39:40)
      Elements Not Converged:      799
      Edges Not Converged:         0
      Local Disp/Energy Index:    96.6%
      Global RMS Stress Index:     8.4%
    Resource Check                    (12:40:04)
      Elapsed Time    (sec):    400.81
      CPU Time        (sec):    443.96
      Memory Usage    (kb):   8674344
      Wrk Dir Dsk Usage (kb):   6650880


>> Pass  6 <<
```

Calculating Element Equations (12:40:05)

Total Number of Equations: 1673248

Maximum Edge Order: 8

Solving Equations (12:41:58)

Post-Processing Solution (12:45:35)

Calculating Disp and Stress Results (12:45:58)

Checking Convergence (12:47:02)

Elements Not Converged: 120

Edges Not Converged: 0

Local Disp/Energy Index: 30.3%

Global RMS Stress Index: 6.5%

Resource Check (12:47:39)

Elapsed Time (sec): 855.32

CPU Time (sec): 926.94

Memory Usage (kb): 8674344

Wrk Dir Dsk Usage (kb): 11039744

>> Pass 7 <<

Calculating Element Equations (12:47:39)

Total Number of Equations: 2097235

Maximum Edge Order: 9

Solving Equations (12:50:48)

Post-Processing Solution (13:00:34)

Calculating Disp and Stress Results (13:02:10)

Checking Convergence (13:04:13)

Elements Not Converged: 4

Edges Not Converged: 0

Local Disp/Energy Index: 10.7%

Global RMS Stress Index: 5.4%

RMS Stress Error Estimates:

Load Set	Stress Error	% of Max Prin Str
LoadSet1	6.90e+01	0.3% of 2.18e+04

** Warning: Convergence was not obtained because the maximum
polynomial order of 9 was reached.

Resource Check (13:05:12)
Elapsed Time (sec): 1907.85
CPU Time (sec): 1870.58
Memory Usage (kb): 8674344
Wrk Dir Dsk Usage (kb): 16633856

The analysis did not converge to within 10% on
edge displacement, element strain energy,
and global RMS stress.

Total Mass of Model: 1.647274e+00

Total Cost of Model: 0.000000e+00

Mass Moments of Inertia about WCS Origin:

Ixx: 1.15974e+03
Ixy: 2.58887e+02 Iyy: 7.62509e+02
Ixz: -9.08058e+01 Iyz: 2.91361e+02 Izz: 1.63793e+03

Principal MMOI and Principal Axes Relative to WCS Origin:

Max Prin Mid Prin Min Prin

1.72633e+03 1.28288e+03 5.50975e+02

WCS X: -2.41962e-02 9.11491e-01 -4.10608e-01

WCS Y: 2.83321e-01 4.00141e-01 8.71560e-01

WCS Z: 9.58720e-01 -9.52456e-02 -2.67927e-01

Center of Mass Location Relative to WCS Origin:

(-6.99839e+00, 2.24552e+01, -7.87680e+00)

Mass Moments of Inertia about the Center of Mass:

Ixx: 2.26926e+02

Ixy: 1.76222e-02 Iyy: 5.79626e+02

Ixz: -2.38960e-08 Iyz: 1.81718e-06 Izz: 7.26638e+02

Principal MMOI and Principal Axes Relative to COM:

Max Prin Mid Prin Min Prin

7.26638e+02 5.79626e+02 2.26926e+02

WCS X: 6.17586e-13 4.99635e-05 1.00000e+00

WCS Y: 1.23607e-08 1.00000e+00 -4.99635e-05

WCS Z: 1.00000e+00 -1.23607e-08 0.00000e+00

Constraint Set: ConstraintSet1: BOGIEANALYSIS

Load Set: LoadSet1: BOGIEANALYSIS

Resultant Load on Model:

in global X direction: 2.549112e-09

in global Y direction: -5.000000e+02

in global Z direction: 2.376936e-09

Measures:

Name	Value	Convergence
max_beam_bending:	0.000000e+00	0.0%
max_beam_tensile:	0.000000e+00	0.0%
max_beam_torsion:	0.000000e+00	0.0%
max_beam_total:	0.000000e+00	0.0%
max_disp_mag:	1.139629e-02	0.2%
max_disp_x:	-2.690803e-03	0.1%
max_disp_y:	-1.136679e-02	0.2%
max_disp_z:	9.177293e-04	0.2%
max_prin_mag*:	-2.180339e+04	13.1%
max_rot_mag:	0.000000e+00	0.0%
max_rot_x:	0.000000e+00	0.0%
max_rot_y:	0.000000e+00	0.0%
max_rot_z:	0.000000e+00	0.0%
max_stress_prin*:	1.341492e+04	8.3%
max_stress_vm*:	1.454146e+04	13.0%
max_stress_xx*:	-1.524289e+04	9.2%
max_stress_xy*:	7.537692e+03	13.4%
max_stress_xz*:	-3.454368e+03	5.0%
max_stress_yy*:	-1.328363e+04	13.7%
max_stress_yz*:	-5.153434e+03	8.7%
max_stress_zz*:	-8.612713e+03	9.9%
min_stress_prin*:	-2.180339e+04	13.1%
strain_energy:	2.838509e+00	0.2%

** Warning: The measures marked by an asterisk (*) were evaluated at (or close to) results singularities. The values of these

measures may be inaccurate, and you must use engineering judgment

when interpreting them.

Analysis "Analysis1" Completed (13:05:12)

Memory and Disk Usage:

Machine Type: Windows 7 64 Service Pack 1

RAM Allocation for Solver (megabytes): 3500.0

Total Elapsed Time (seconds): 1913.96

Total CPU Time (seconds): 1871.50

Maximum Memory Usage (kilobytes): 8674344

Working Directory Disk Usage (kilobytes): 16633856

Results Directory Size (kilobytes):

476043 .\Analysis1

Maximum Data Base Working File Sizes (kilobytes):

1048576 .\Analysis1.tmp\kblk1.bas

1048576 .\Analysis1.tmp\kblk10.bas

1048576 .\Analysis1.tmp\kblk11.bas

117760 .\Analysis1.tmp\kblk12.bas

1048576 .\Analysis1.tmp\kblk2.bas

1048576 .\Analysis1.tmp\kblk3.bas

1048576 .\Analysis1.tmp\kblk4.bas

1048576 .\Analysis1.tmp\kblk5.bas

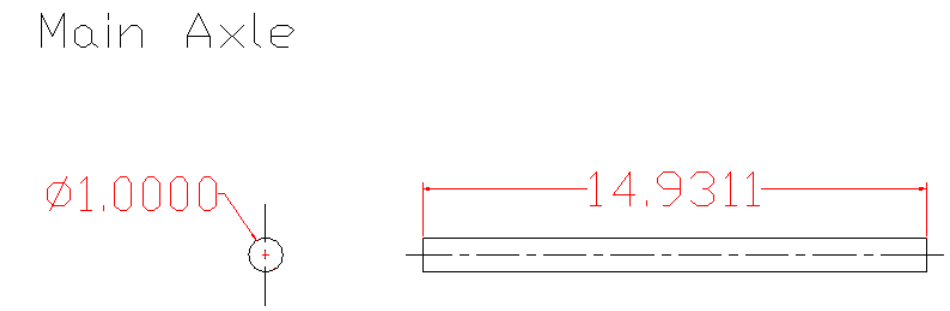
1048576 .\Analysis1.tmp\kblk6.bas

1048576 .\Analysis1.tmp\kblk7.bas

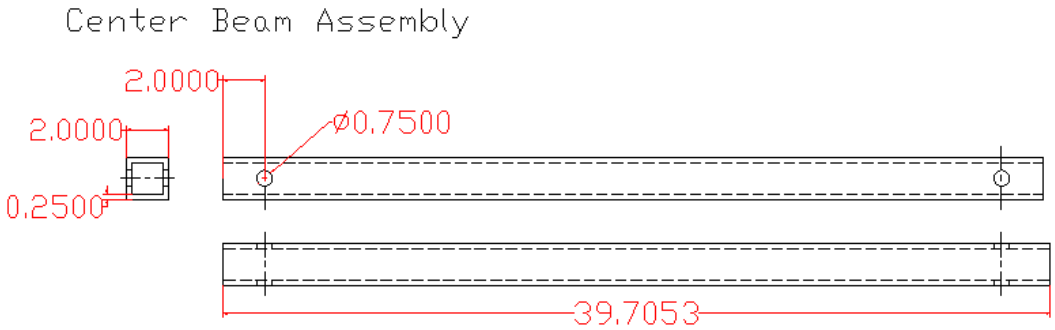
```
  1048576 .\Analysis1.tmp\kblk8.bas
  1048576 .\Analysis1.tmp\kblk9.bas
  1048576 .\Analysis1.tmp\kel1.bas
  1048576 .\Analysis1.tmp\kel2.bas
  1048576 .\Analysis1.tmp\kel3.bas
  1048576 .\Analysis1.tmp\kel4.bas
   231424 .\Analysis1.tmp\kel5.bas
   556032 .\Analysis1.tmp\oel1.bas
------------------------------------------------------------

Run Completed

Wed May 07, 2014   13:05:20

------------------------------------------------------------
```

Appendix L: Engineering Drawings

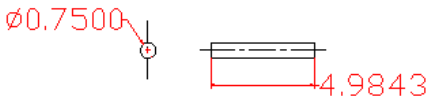
2x Main Axle (inches)



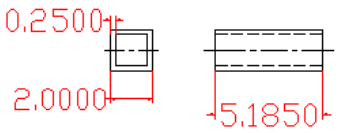
2x H-Bar long member (inches)



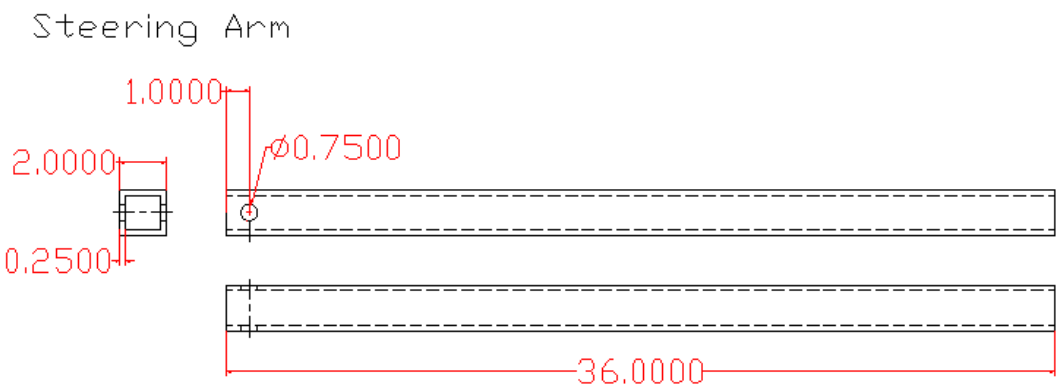
4x H-bar Axle (inches)



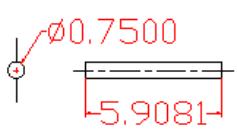
1x H-Bar short member (inches)



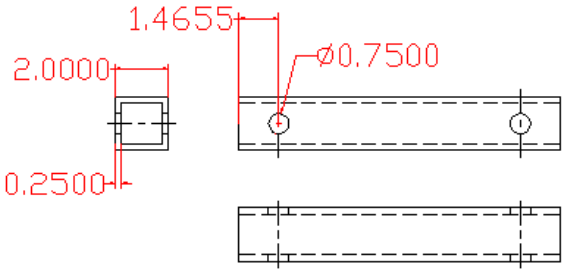
2x Steering arm (inches)



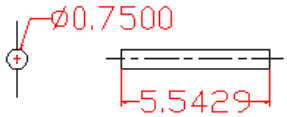
2x Steering arm axle (inches)



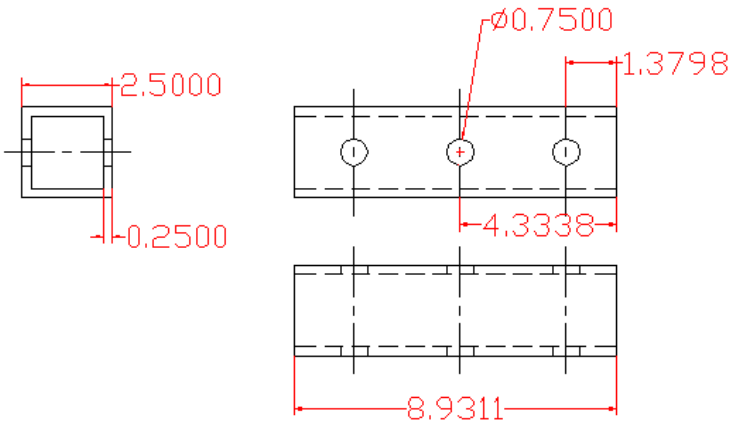
4x upper tubing (inches)



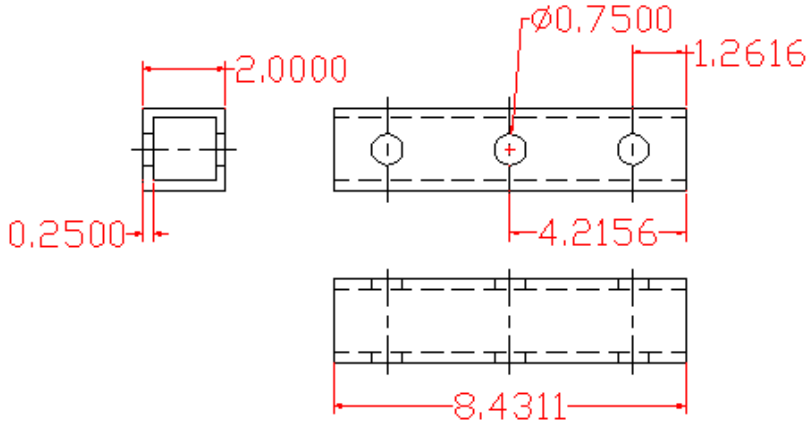
8x upper axles (inches)



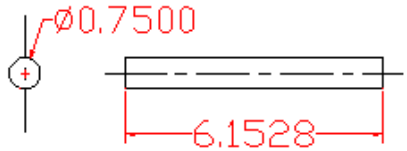
2x Reciever tube (inches)



2x Lower 2” tube (inches)

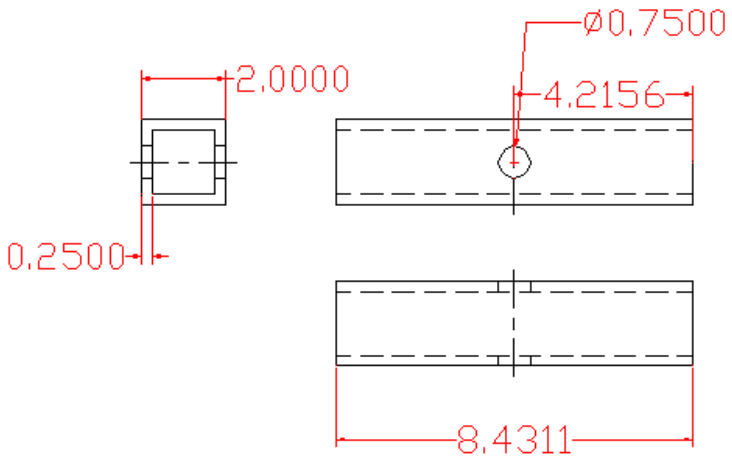


4x Lower axles (inches)

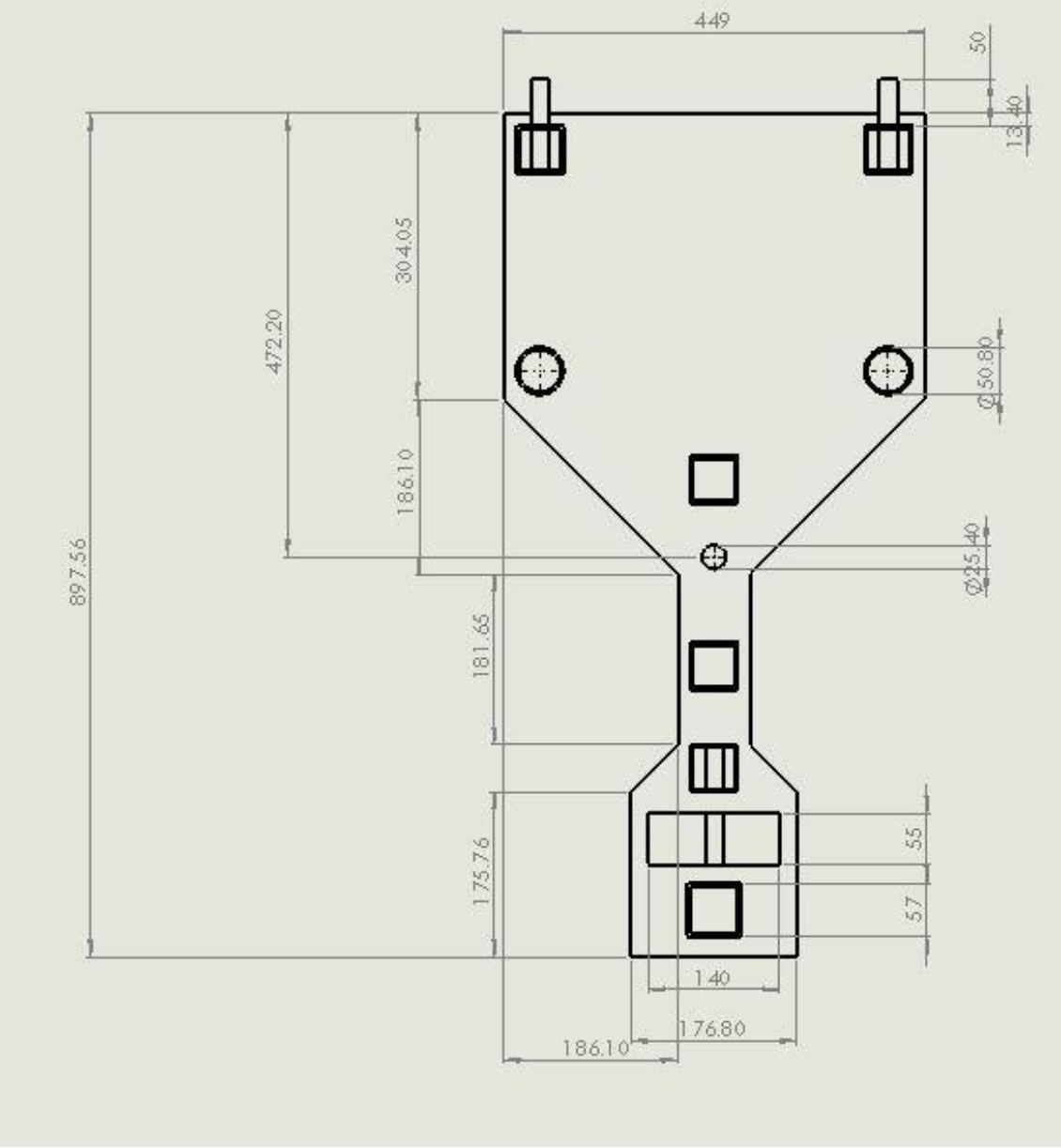


4x H-bar connecter 2” tube (inches)

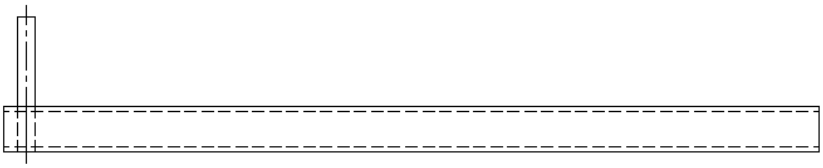
Tubing for Center Beam Connection



4x Plasma Cut ¼” steel plate (mm)

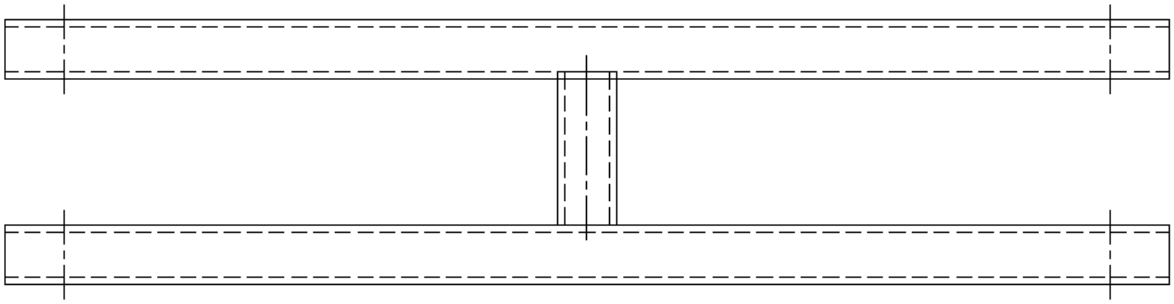

449
50
13.40
304.05
472.20
897.56
186.10
Ø50.80
Ø25.40
181.65
175.76
55
57
140
176.80
186.10

Appendix M: Bogie Assembly Drawings

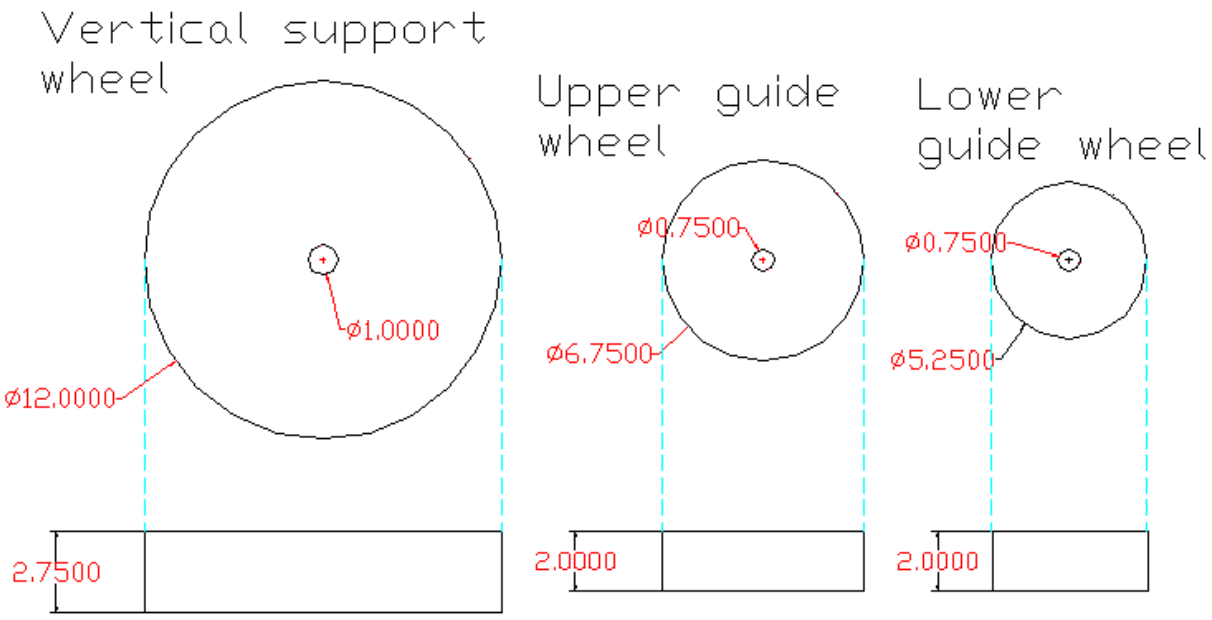


Assembled steering arm

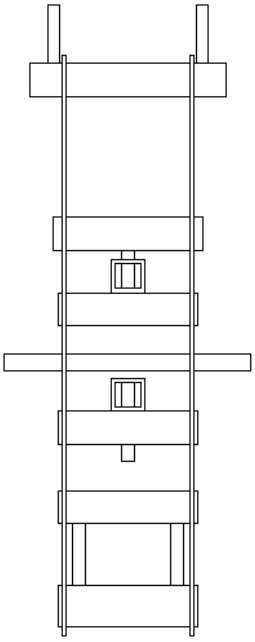
Assembled h-bar

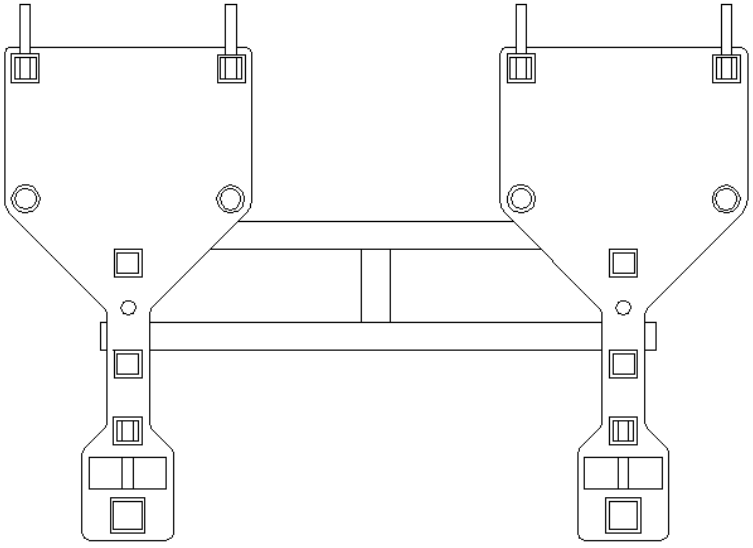


General wheel sizing

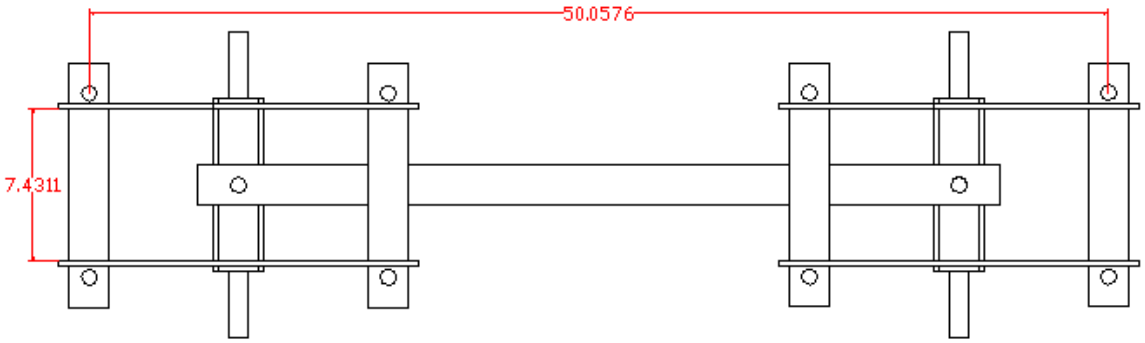


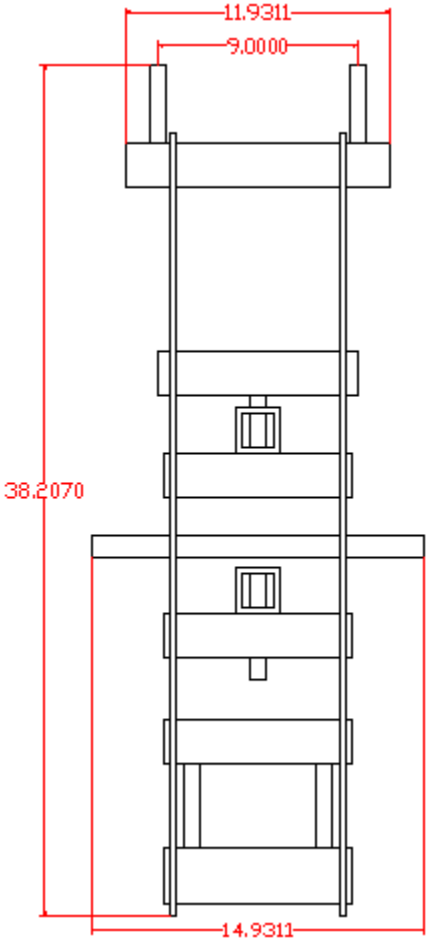
Assembly w/o wheels or dimensions

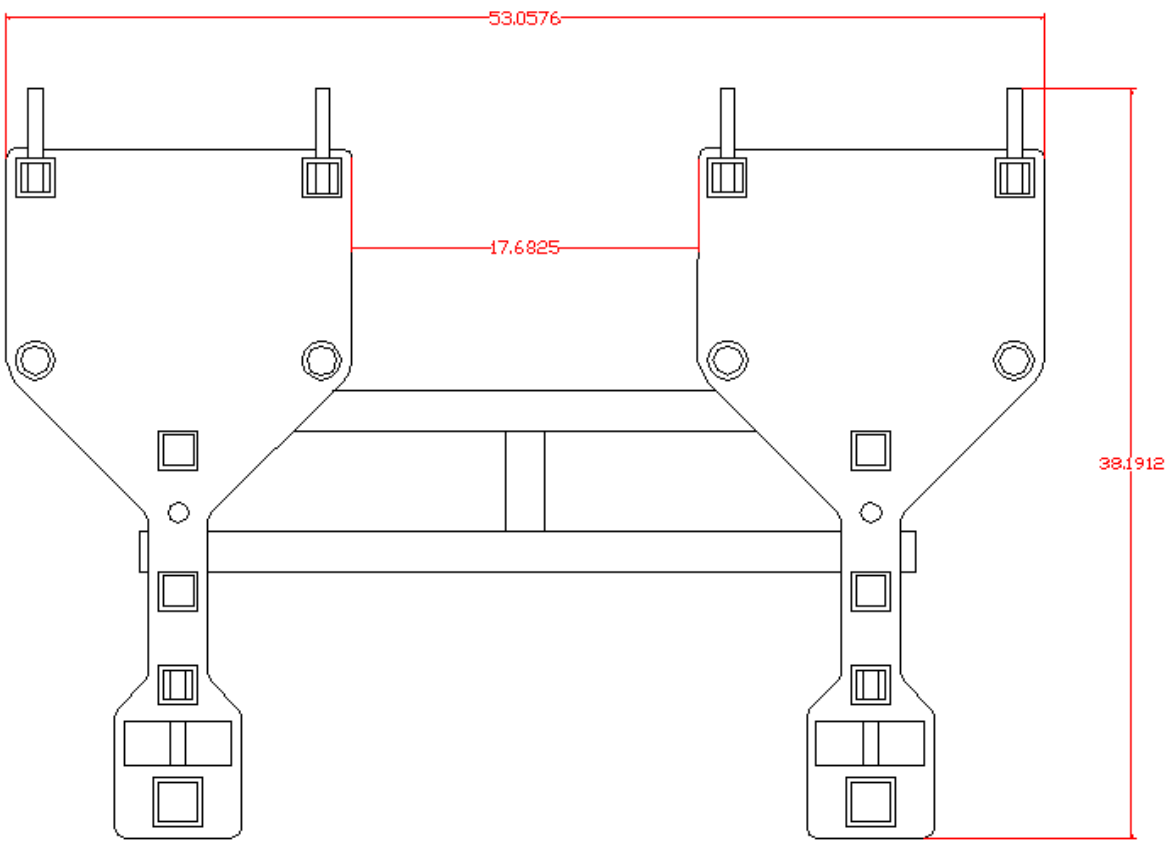




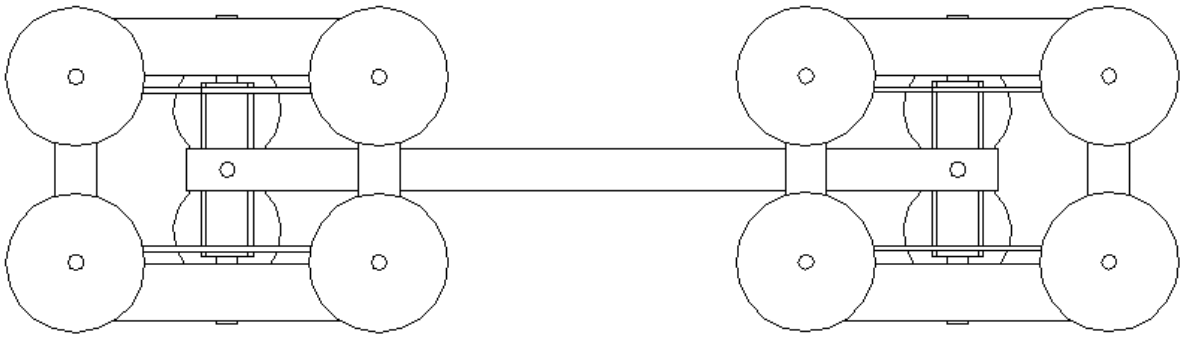
Assembly w/o wheels with dimensions

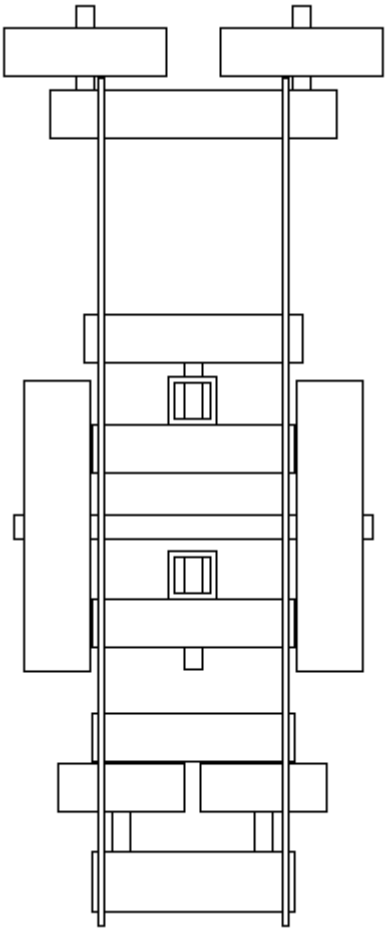


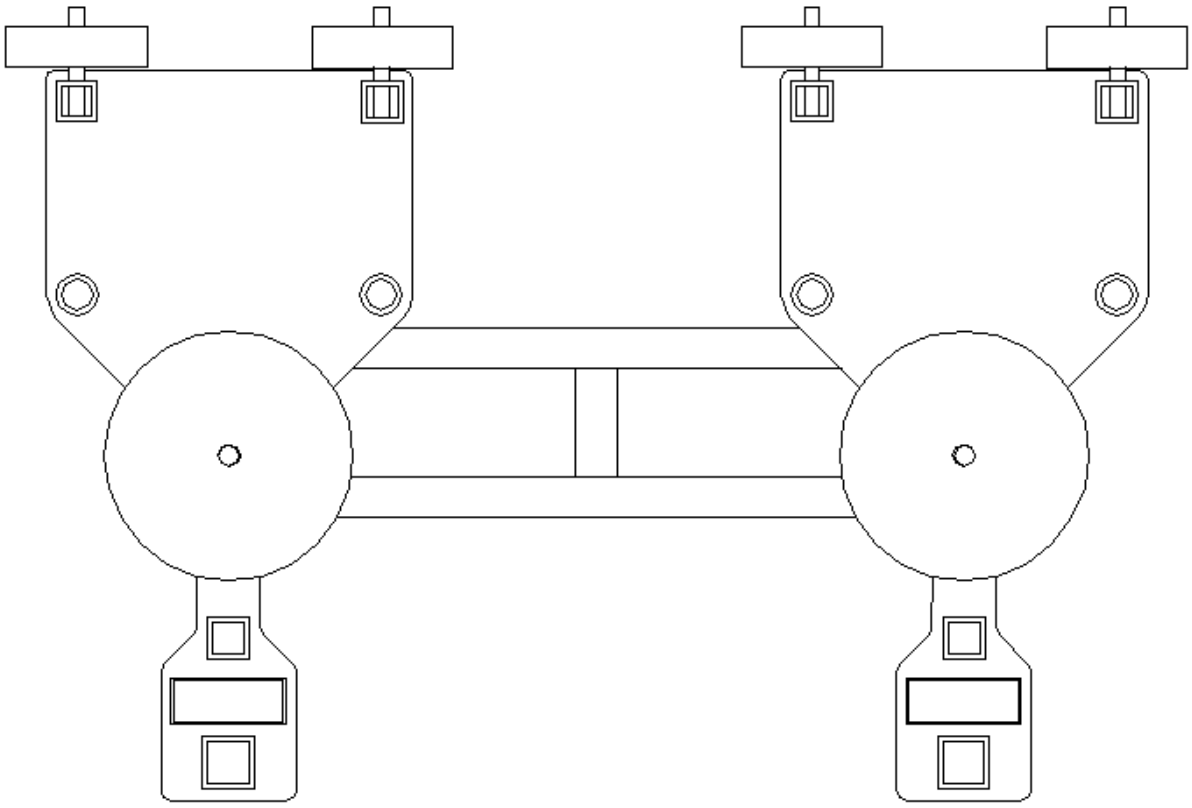




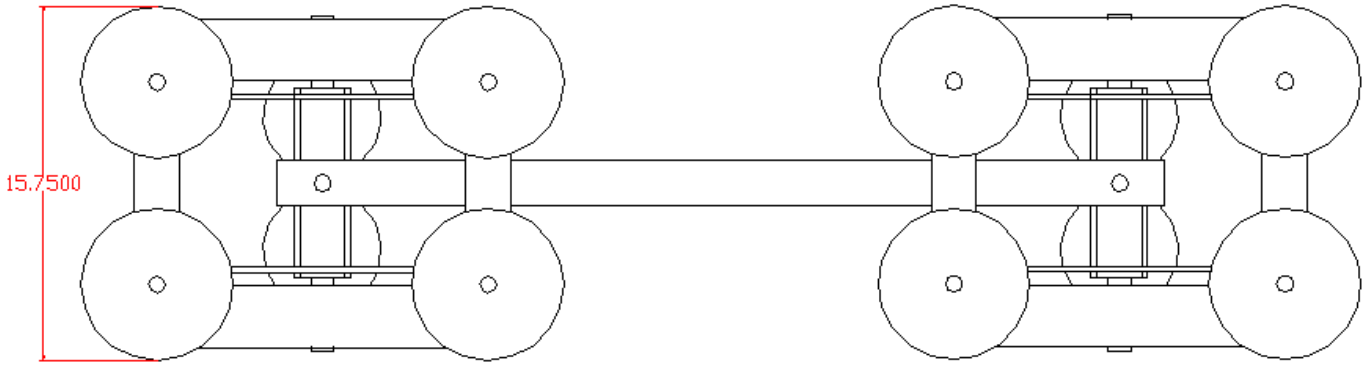
Assembly with wheels, no dimensions

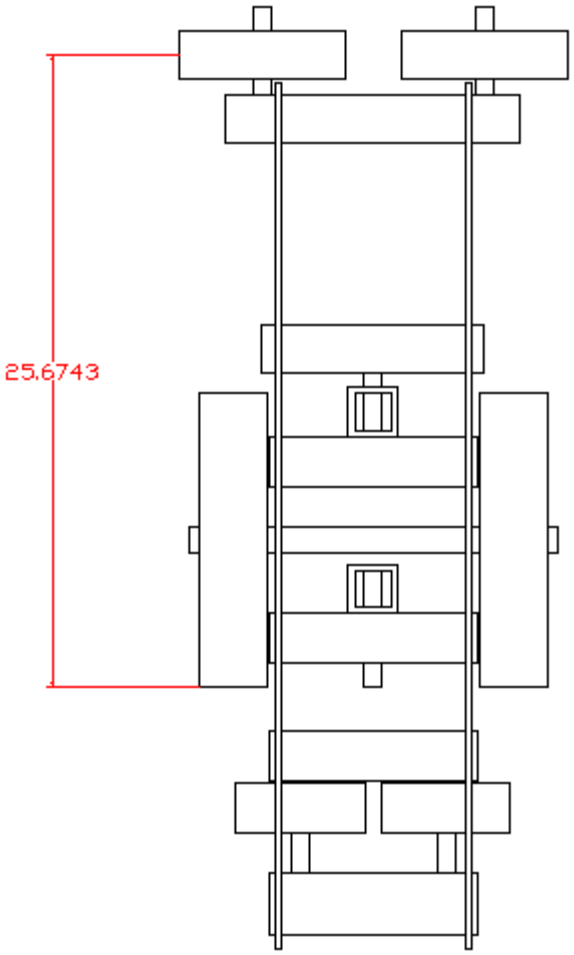


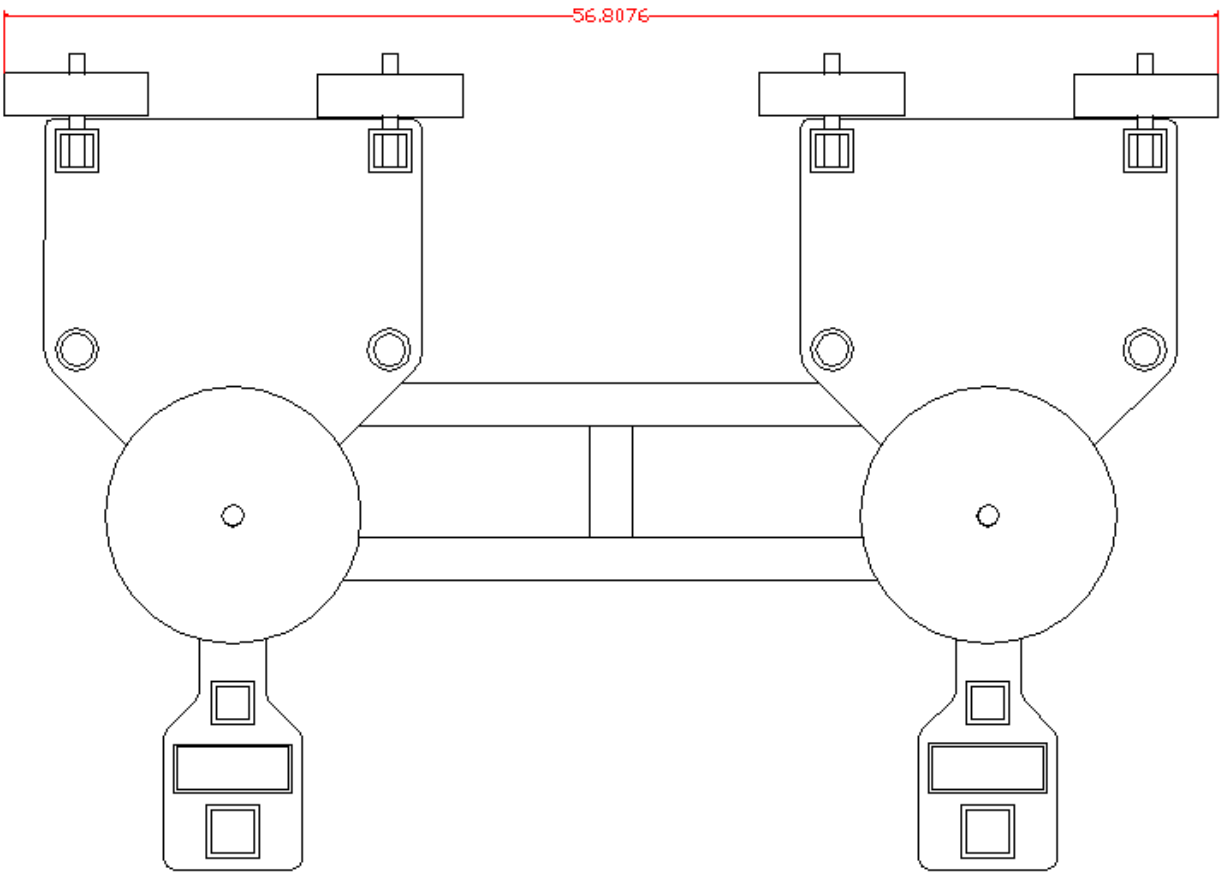




Assembled with wheels and dimensions




25.6743


56.8076

Appendix N: Bogie Bill of Materials

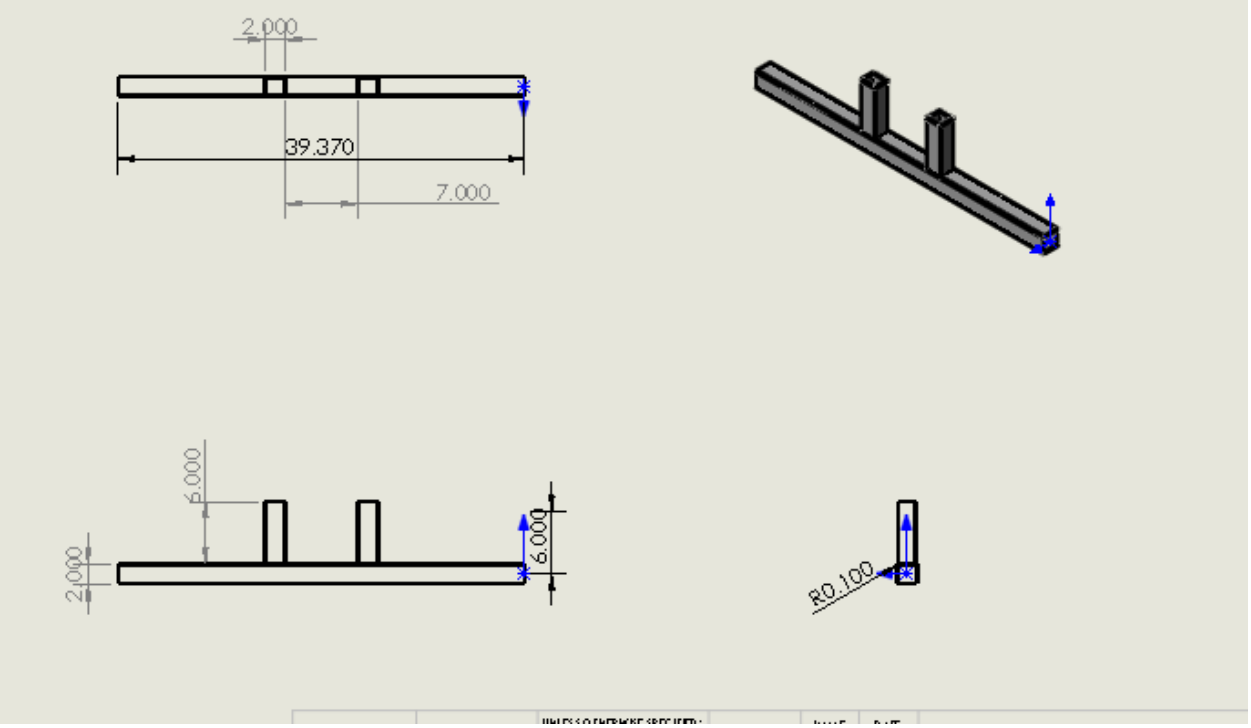
4x	Plasma Cut ¼” thick steel plate
3’	1” diameter Cold-Rolled 1018 steel bar
40’	¾” diameter Cold-Rolled 1018 steel bar
10’	2”x2”x0.120 Square tubing A513 steel
2’	2”x2” receiver tube
4’	2” Steel Pipe (exhaust grade)
8x	Sunray 6.75”x2.00” S6-N6L2XA 60D Orange Polyethylene wheels
6x	Sunray 5.25”x2.00” S9-N5D2XA 60D Green Polyethylene wheels
4x	Hamilton W-1220-MT-1 with Tapered bearings
8x	2” black oxide coated mild steel collar clamps
4x	1” black oxide coated mild steel collar clamps
30x	¾” black oxide coated mild steel collar clamps
4x	20x35x10mm Thrust bearings
10x	18-8 Stainless Steel Round Shim, 0.090” Thick 1” inner dia 1-1/2” outer dia
50x	¾” SAE standard Flat Washer zinc plated

Appendix O: Cabin Bill of Materials

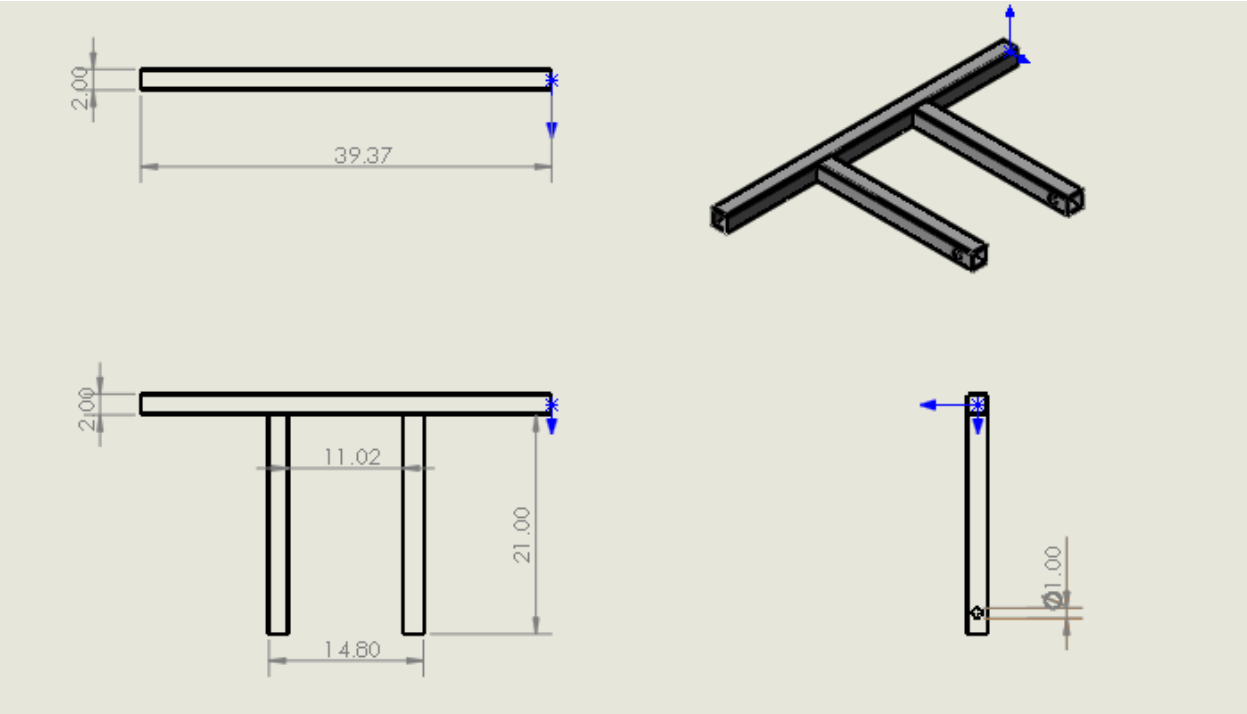
Item	Vendor	Product #	Size	Price	Quantity	Total		
Styrofoam	Home Depot	614-645	1.5"x4'x8'	$ 16.28	10	$ 162.80		
Flat Punch Zinc	Home Depot	584-265	1 3/8"x1/16"x48"	$ 6.51	2	$ 13.02		
Bolts	Home Depot	661-767	5/16"x1 1/2"	$ 0.20	6	$ 1.20		
Washer	Home Depot	328-141	5/16"	$ 2.68	1	$ 2.68		
Nuts	Home Depot	328-639	5/16"	$ 2.46	1	$ 2.46		
Plastic Dip	Home Depot		11oz	$ 5.98	5	$ 29.90		
SHS Super Gold Glue	Sheldon's Hobby Shop		1oz	$ 12.99	2	$ 25.98	10% discount	$ 24.68
Plywood	Sheldon's Hobby Shop		1/32"x12"x48"	$ 19.99	5	$ 99.95	10% discount	$ 97.95
Thermal Shrink	Sheldon's Hobby Shop			$ 17.99	2	$ 35.98	10% discount	$ 34.18
								$ 368.87

Appendix P: Cabin Drawings

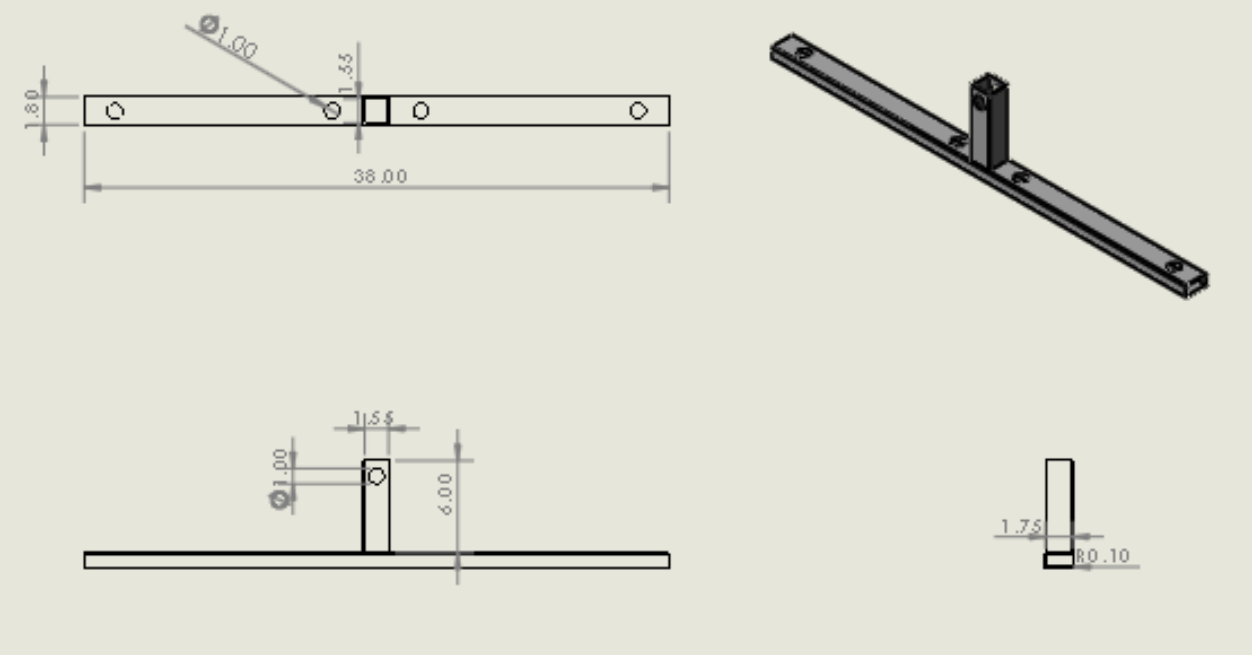
Top H_Bar (Inches)



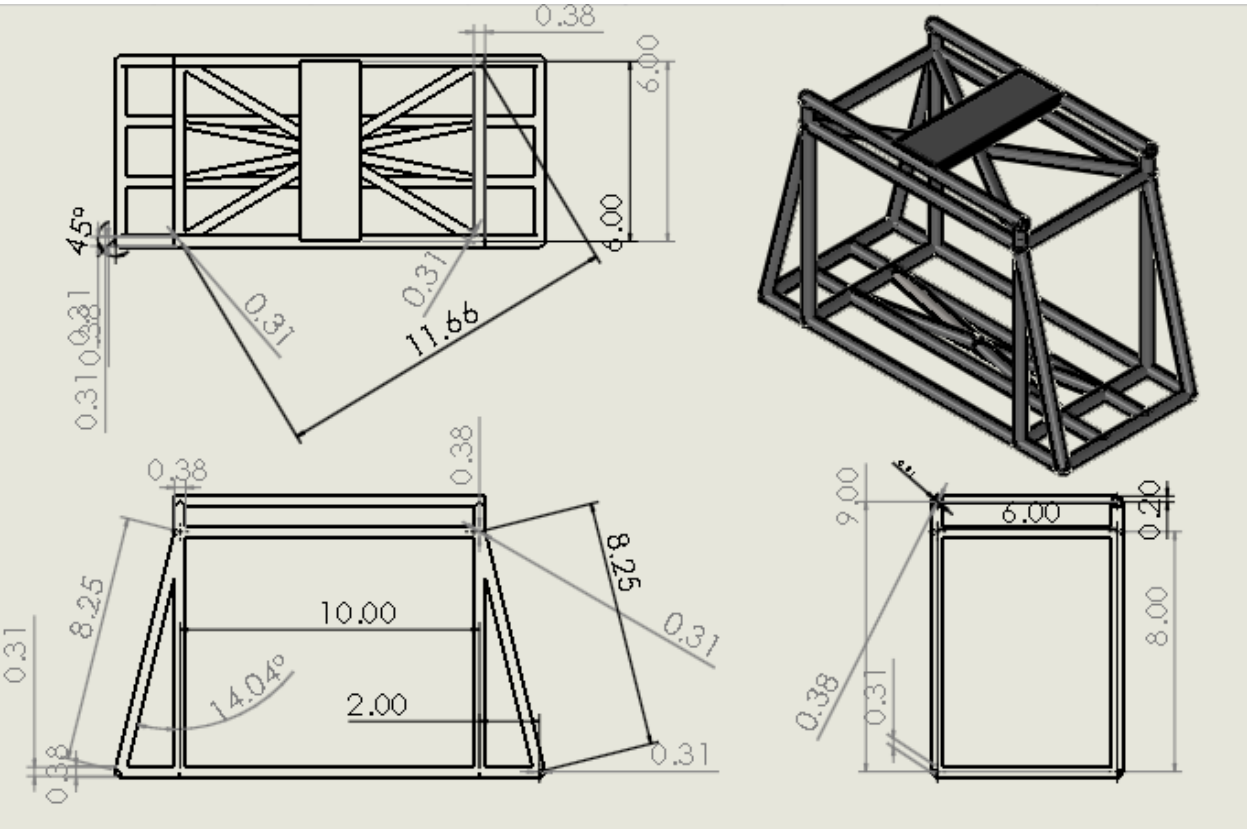
Bottom H_Bar (Inches)



Cabin Connection (Inches)



Frame (Inches)



Appendix Q: MATLAB Code for Speed Control PID Tuning

```
%% ME 195D - Spring 2014 - DC Motor PID Tuning
%  Description: MATLAB script used to calculate DC motor transfer function
%               then using pidtool() to graphically tune the PID to acquire
%               the gain constants Kp, Ki, Kd that yield the desired system
%               response characteristics
%  Team members: Cory Ostermann
%                Man Ho
%                Randall Morioka
%

clear;
clc;
J = 5.051*10^(-7);  %calculated wheel moment of inertia
b = 3.913*10^(-5);  %calculated motor viscous friction constant
K = 0.0584;         %calculated motor torque/EMF constant
R = 2.895;          %motor armature resistance
L = 0.161;          %motor armature inductance
s = tf('s');
P_motor = K/((J*s+b)*(L*s+R)+(K^2)) %DC motor transfer function
figure(1);
step(P_motor);   %plot step response of open-loop system without any control
title('Uncompensated Step Response of DC Motor');
Kp = 0;          %arbitrary Kp value to establish baseline control system
Ki = 0.2892;    %arbitrary Ki value to establish baseline control system
Kd = 0;          %arbitrary Kd value to establish baseline control system
C = pid(Kp,Ki,Kd)   %create PID motor control
Gcl = feedback(P_motor*C,1)
figure(2);
step(Gcl);
title('Compensated Step Response of DC Motor');
figure(3);
rlocus(Gcl);
title('Root Locus of Closed-loop System');
figure(4);
bode(Gcl);
title('Bode Plots of Closed-loop System');

%pidtool(P_motor,C);      %call PID GUI to tune Kp,Ki,Kd
```

Appendix R: Arduino PID Speed Control Source Code

```
/*
 * Speed and Acceleration Control for DC Motor
 * ENGR 195D - Spring 2014 - Spartan Superway
 *
 * Description:  Controlling DC motor to achieve and maintain desired speed
 *               while not exceeding maximum acceleration of 0.25g.
 *               PID parameter acquired based on motor characteristics
 *               and through MATLAB
 * Adapted from: Test MD03a -----
 *                    http://forum.arduino.cc/index.php/topic,8652.0.html
 *
 * Team members: Cory Ostermann
 *               Man Ho
 *               Randall Morioka
 * Notes: Code not completely tested flawless operations.
 */

#define M1  10
#define M2  11
#define PWMM  6
#define encoder1  3     //encoder pin A
#define encoder2  8     //encoder pin B
#define PIDloop  100    //PID loop time

unsigned long lastMillis = 0;
unsigned long lastMillisPrint = 0;
int set_speed = 300;    //set speed
int act_speed = 0;      //actual speed
int PWM_val = 0;        //(25% = 64; 50% = 127; 75% = 191; 100% = 255)
volatile long count = 0;    //rev counter
float Kp = 0;            //PID Proportional control gain
float Ki = 0.2892;       //PID Integral control gain
float Kd = 0;            //PID Derivative control gain

void setup()
{
  Serial.begin(115600);
  pinMode(M1, OUTPUT);
  pinMode(M2, OUTPUT);
  pinMode(PWMM, OUTPUT);
  pinMode(encoder1, INPUT);
  pinMode(encoder2, INPUT);
  //turns on pull-up resistor
  digitalWrite(encoder1, HIGH);
  digitalWrite(encoder2, HIGH);
  attachInterrupt(1, rencoder, FALLING);

  analogWrite(PWMM, PWM_val);
  digitalWrite(M1, LOW);
  digitalWrite(M2, HIGH);
}

void loop()
{
  if((millis() - lastMillis) >= PIDloop)
```

```
  {
    lastMillis = millis();
    getSpeed();            //calculate motor speed
    PWM_val = updatePID(PWM_val, set_speed, act_speed);  //calculcate PWM
                                                         //output
    analogWrite(PWMM, PWM_val);    //output PWM signal to motor
  }
}

void getSpeed()
{
  static long countAnt = 0;
  //20 counts per revolution of backshaft x 30 backshaft rev per frontshaft
  //rev = 600 counts/frontshaft rev
  act_speed = ((count - countAnt)*(60*(1000/PIDloop)))/(20*30);
  countAnt = count;
}

int updatePID(int command, int targetVal, int currentVal)
{
  float PIDterm = 0;
  int error = 0;
  static int last_error = 0;
  error = abs(targetVal) - abs(currentVal);
  PIDterm = (Kp * error) + (Ki*(error + last_error)) + (Kd*(error -
last_error));
  last_error = error;
  return constrain(command + int(PIDterm), 0, 255);
}

void rencoder()
{                                    // pulse and direction, direct port
                                     //reading to save cycles
 // if(digitalRead(encodPinB1)==HIGH)   count ++;
 if (PINB & 0b00000001)
 {
   count++;
 }
 // if (digitalRead(encodPinB1)==LOW)   count --;
 else
 {
   count--;
 }
}
```

Appendix S: Arduino Maker Faire Sketch

```
/*
Spartan Superway 1/12 Scale Model Maker Faire Demonstration Sketch

2013-2014 ENGR 195D / Spartan Superway

Created by: Cory Ostermann, Man Ho, Randall Morioka, Eriberto Velzquez, and
Anthony Vo

This sketch provides functionality for a user to send new destinations to a
pod via an SJOne microcontroller connected to a laptop.

*/

#include "SPI.h"
#include "NewPing.h"

// Starting Location Variables:
#define START_NODE  0
#define START_LINE  0
#define START_TICK  1
#define START_DESTINATION -1

// Pin definitions:
#define TRACK_OFFLINE A0  // The pin attached to the Offline reflective
object sensor
#define TRACK_ONLINE  A1  // The pin attached to the Online reflective object
sensor
#define SWITCH_OFFLINE 4  // The pin attached to solenoid actuating the
Offline Switch
#define SWITCH_ONLINE  5  // The pin attached to solenoid actuating the
Online Switch
#define MOTOR_A1  3
#define MOTOR_A2  2
#define TRIG  6
#define ECHO  7

// Miscellaneous definitions:
#define ONLINE   1
#define OFFLINE  0
#define THRESHOLD_ONLINE  100    // The maximum amount of light reflected
from the markers to trigger the sensors
#define THRESHOLD_OFFLINE 100
#define BOUNCE_TIME  250  // The time delay to prevent multiple readings
#define NO_DESTINATION  -1   // The value for when a pod does not have a
specified destination
#define MAX_RANGE 20      // Maximum detection range (cm) at which the pod
will begin to slow
#define MIN_RANGE 10      // Distance (cm) at which the pod will perform a
active braking
#define MAX_DUTY  65      // Maximum allowable motor duty cycle
#define MIN_DUTY  60      // Minimum duty cycle for motion
#define BRAKE  1          // State tracker for active braking
#define BRAKE_TIME  1000  // The timer that determines when the motors go
from active braking to coasting
#define STATION_1  0
```

```
#define STATION_2  2
#define STATION_3  6


// SPI Variables:
int buf [5];
volatile byte pos;
volatile boolean process_it;

// Navigation variables:
int readingOffline, readingOnline;  // Stores the reflective object sensor
readings
int sensorOnlineNew, sensorOnlineStored, sensorOfflineNew,
sensorOfflineStored;  // Sensor state tracking variables
unsigned long timerOnline, timerOffline;  // Timers to prevent marker
bouncing
int destination;
int locationNode, locationTick, locationLine;  // locationNode tracks the
section of track
                                               // locationTick tracks the
marker readings
                                               // locationLine tracks whether
the pod is Online or Offline
int locationNodePrevious;
// Object Detection Variables:
NewPing sonar(TRIG, ECHO, MAX_RANGE);
int range;
// Speed Control Variables:
int dutyCycle;
int activeBrake;
unsigned long timerBrake;

void setup()
{
  // Setup pin modes:
  pinMode(TRACK_ONLINE, INPUT);
  pinMode(TRACK_OFFLINE, INPUT);
  pinMode(SWITCH_ONLINE, OUTPUT);
  digitalWrite(SWITCH_ONLINE, LOW);
  pinMode(SWITCH_OFFLINE, OUTPUT);
  digitalWrite(SWITCH_OFFLINE, LOW);
  // Initialize the Serial Monitor
  Serial.begin(9600);
  // Initialize variables:
  locationNode = START_NODE;
  locationTick = START_TICK;
  locationLine = START_LINE;
  destination = destinationWrite(START_DESTINATION);
  range = 0;          // No objects detected
  activeBrake = 0;  // Turn off the brake
  // SPI related initialization:
  pinMode(MISO, OUTPUT);
  // turn on SPI in slave mode
  SPCR |= _BV(SPE);
  // get ready for an interrupt
  pos = 0;   // buffer empty
  process_it = false;
```

```
  // now turn on interrupts
  SPI.attachInterrupt();
  printSnapshot();
  // ****************************************************
  //delay(5000);
  // ****************************************************

}  // end setup

void loop()
{
  /*
  SPI COMMUNICATION
  */
  if (process_it)
    {
      if( -1 == destination )
      {
        switch( buf[0] )
        {
          case 1:
            destination = destinationWrite(buf[1]);
            locationNode = destinationWrite(buf[2]);
            locationTick = 1;
            locationLine = 0;
            break;
          case 2:
            destination = destinationWrite(buf[1]);
            break;
          case 111:
            locationNode = buf[1];
            locationTick = buf[2];
            locationLine = buf[3];
            break;
          default:
            break;
        }  // end buf[0] switch
        printSnapshot();
        //pos = 0;
      }
      /*
      for(int i=0; i<(pos-1); i++)
      {
        Serial.print (buf[i]);
        Serial.print(" ");
      }
      */
      pos = 0;
      process_it = false;
    }  // end of flag set
  /*
  NAVIGATION
  */
  // Take new sensor readings:
  readingOffline = analogRead(TRACK_OFFLINE);
  readingOnline = analogRead(TRACK_ONLINE);
```

```
  // Determine the new sensor states ( 1 = on marker, 0 = off marker )
  if( readingOnline < THRESHOLD_ONLINE )
  {
    sensorOnlineNew = 1;
  } else {
    sensorOnlineNew = 0;
  }
  if( readingOffline < THRESHOLD_OFFLINE )
  {
    sensorOfflineNew = 1;
  } else {
    sensorOfflineNew = 0;
  }
  // Compare and possibly reset the Stored values based off of the New
values.
  // Offline Update:
  if( sensorOfflineStored < sensorOfflineNew )  // The leading edge condition
  {
    sensorOfflineStored = sensorOfflineNew;
  }
  else if( sensorOfflineStored > sensorOfflineNew )  // The trailing edge
condition
  {
    sensorOfflineStored = sensorOfflineNew;
    if( millis() - timerOffline > BOUNCE_TIME )  // This if state prevents
multiple readings
    {
      Serial.println("Offline: Trailing Edge");
      Serial.println("----------------------");
      // Increment the tick counter:
      switch( locationTick )
      {
        case 2:
          locationTick = 0;
          locationLine = 1;
          break;
        default:
          locationTick++;
          break;
      }  // end locationTick update
      timerOffline = millis();
      // Manipulate the Switch:
      switch( locationTick )
      {
        case 1:
          switchWrite(2);  // Turn the switch off
          if( OFFLINE == locationLine && locationNode == destination )
          {
            destination = -1;  // Arrived at specified destination
          }
          break;
        case 2:
          if( ONLINE == locationLine )  // The pod is exiting an Online
section of a junction
          {
            switchWrite(ONLINE);  // Turn on the online switch
          }
```

```
          else if( OFFLINE == locationLine )
          {
            switchWrite(OFFLINE);
          }
          break;
        default:
          break;
      }  // end locationTick switch
      printSnapshot();
    }  // end bounce
  }
  // Online Update:
  if(sensorOnlineStored < sensorOnlineNew )  // The leading edge condition
  {
    sensorOnlineStored = sensorOnlineNew;
  }
  else if( sensorOnlineStored > sensorOnlineNew )  // The trailing edge
condition
  {
    sensorOnlineStored = sensorOnlineNew;
    if( millis() - timerOnline > BOUNCE_TIME )  // This if statement prevents
multiple reads
    {
      Serial.println("Online: Trailing Edge");
      Serial.println("---------------------");
      // Increment the node counter:
      locationNodePrevious = locationNode;
      switch( locationNode )
      {
        case 4:
          if( STATION_3 != destination )
          {
            locationNode = 10;
          }
          else
          {
            locationNode++;
          }
          break;
        case 9:
          locationNode = 0;
          break;
        case 10:
          locationNode = 8;
          break;
        default:
          locationNode++;
          break;
      }  // end locationNode update switch
      //  Manipulate the switch:
      switch( locationNode )
      {
        case STATION_1:  // approaching Station 1
        case STATION_2:  // approaching Station 2
        case STATION_3:  // approaching Station 3
          if( locationNode == destination )       // At the destination node
          {
```

```
            locationLine = 0;                     // Taking the Offline route
            switchWrite(OFFLINE);
          } else {                                 // Not the destination node
            locationLine = 1;                      // Stays on the Online
section
            switchWrite(ONLINE);
          }
          break;
        case 4:                    // Approaching the bypass junction
          if( STATION_3 != destination )  // If Station 3 (node 5) is not the
destination...
          {  // Use the Mainline switch
            switchWrite(ONLINE);  // Turn on the Online switch
          }
          else // Use the Offline switch
          {
            switchWrite(OFFLINE);  // Turn on the Offline switch first
          }
          break;
        case 8:
          if( 10 == locationNodePrevious )
          {
            switchWrite(ONLINE);
          }
          else
          {
            switchWrite(OFFLINE);
          }
          break;
        default:  // This case will appear everytime the pod is exiting a
junction
          switchWrite(2);    // Turn off the switch
          locationTick = 0;  // Reset the tick counter
          locationLine = 1;
          break;
      }  // end locationNode switch
      printSnapshot();
      timerOffline = millis();
    }
  }
  // end Navigation

  /*
  OBJECT DETECTION
  */
  // end Object Detection
  /*
  SPEED CONTROL
  */
  // Update the motor duty cycle based on location and detected object range:
  if( (0 < range) && (MIN_RANGE >= range) )  // If the pod detects an object
within the critical range
  {
    digitalWrite(MOTOR_A1, HIGH);  // Perform active braking (HIGH-HIGH)
    digitalWrite(MOTOR_A2, HIGH);
    dutyCycle = 0;                // Set the duty cycle to 0
  }
```

```
  else if( (MIN_RANGE < range) && (MAX_RANGE >= range) )
  {
    dutyCycle = map(range, MIN_RANGE, MAX_RANGE, MIN_DUTY, MAX_DUTY);  //
Adjust the speed based on the distance
  }
  else if( NO_DESTINATION == destination )
  {
    digitalWrite(MOTOR_A1, HIGH);  // Perform active braking (HIGH-HIGH)
    digitalWrite(MOTOR_A2, HIGH);
    dutyCycle = 0;                // Set the duty cycle to 0
  }
  else
  {
    dutyCycle = MAX_DUTY;
  }
  if( MAX_DUTY < dutyCycle )  // Safety precaution in case the duty cycle
somehow becomes higher than the maximum duty cycle
  {
    dutyCycle = MAX_DUTY;
  }
  // Run the motors at the adjusted duty cycle:
  analogWrite(MOTOR_A1, dutyCycle);
  digitalWrite(MOTOR_A2, LOW);
  // end Speed Control


}  // end loop

void printSnapshot()
{
  Serial.print("Destination:  ");
  Serial.println(destination);
  Serial.print("Location:  ");
  Serial.print(locationNode);
  Serial.print("  (from ");
  Serial.print(locationNodePrevious);
  Serial.println(")");
  Serial.print("Tick:  ");
  Serial.println(locationTick);
  Serial.print("Line:  ");
  Serial.println(locationLine);
  Serial.print("Switch:  ");
  if( digitalRead(SWITCH_ONLINE) == HIGH && digitalRead(SWITCH_OFFLINE) ==
LOW)
  {
    Serial.println("Online");
  }
  else if( digitalRead(SWITCH_OFFLINE) == HIGH && digitalRead(SWITCH_ONLINE)
== LOW)
  {
    Serial.println("Offline");
  }
  else if( digitalRead(SWITCH_OFFLINE) == LOW && digitalRead(SWITCH_ONLINE)
== LOW)
  {
    Serial.println("Off");
  }
```

```
  else {
    Serial.println("Error");
  }
  Serial.print("Range:  ");
  Serial.println(range);
  Serial.print("Duty Cycle:  ");
  Serial.println(dutyCycle);
  Serial.println("---------------------------------");
}

int destinationWrite(int station_num)  // Converts a station number to the
corresponding location node
{
  int node;
  switch( station_num )
  {
    case 1:  // Station 1
      node = STATION_1;
      break;
    case 2:
      node = STATION_2;  // Station 2
      break;
    case 3:  // Station 3
      node = STATION_3;
      break;
    default:
      node = -1;
      break;
  }  // end station_num switch
  return node;
}  // end destinationWrite;

void switchWrite(int line)
{
  switch(line)
  {
    case ONLINE:
      digitalWrite(SWITCH_OFFLINE, LOW);
      digitalWrite(SWITCH_ONLINE, HIGH);
      break;
    case OFFLINE:
      digitalWrite(SWITCH_ONLINE, LOW);
      digitalWrite(SWITCH_OFFLINE, HIGH);
      break;
    default:
      digitalWrite(SWITCH_ONLINE, LOW);
      digitalWrite(SWITCH_OFFLINE, LOW);
      break;
  }  // end line switch
}  //  end swtichWrite

// SPI interrupt routine
ISR (SPI_STC_vect)
{
byte c = SPDR;  // grab byte from SPI Data Register
Serial.print("ISR\n");
  // add to buffer if room
```

```
  if (pos < sizeof buf)
    {
    buf [pos++] = c;

    // example: newline means time to process buffer
    if (c == 0x00)
      process_it = true;

    }  // end of room available

}  // end of interrupt routine SPI_STC_vect
```

Appendix T: Arduino Pod class

```
/*
  Pod.h - Library for controlling Spartan Superway 1/12 scale pods
  Created by Cory Ostermann (Lead), Man Ho,  Randall Morioka, and Anthony Vo
  Spartan Superway 2013-2014 Controls Team
*/

#ifndef Pod_h
#define Pod_h

#include "Arduino.h"

class Pod
{
  public:
         Pod(int identifier);
         //void initialize();             // Sets up intial pod state, including Depot
location and SPI settings
         void statusWrite(int status_num);
         int statusRead();
         int destinationRead();    // Returns the pods current destination
         void destinationWrite(int station_id);  // Sets the pods next destination
         int locationRead();              // Returns the pods current location
         void locationUpdate();    // Increments the pod's location as it reads tick marks
         //boolean ReflectiveSensor(int sensor, int edge);
         void switchWrite(int line);             // Manipulates the switching mechanism
         int DetectObject();              // Senses if an object is obstructing the path
(Ultrasonic)
         int speedReadPWM();              // Returns the speed of the pod as a PWM duty
cycle
         void speedWritePWM(int duty_cycle);     // Sets the speed of the of the pod using
a duty cycle
         void Cruise();
  private:
         /*
         Pod Characteristics
         */
         int _podID;        // An identifying number that the Network can use to
coordinate pods
         int _podStatus;    // Pod status: ready for instruction or not ready
         int _dutyCycle;    // The PWM duty cycle being applied to the motors
         int _locationNode; // The most recent node passed by the pod
         int _locationLine; // Mainline or Offline
         int _locationAlt;         // For deactivating the switch and slowing the pod
         int _destination;         // The next station the pod will stop at
         int _range;               // The range to the detected object (0 if no object
detected)
         int _stateTrack1New;              // Tracks the state of the location Node sensor
(true means HIGH)
         int _stateTrack1Prev;             // The previous state of the sensor
         int _stateTrack2New;              // Tracks the state of the location Alt sensor
(true means HIGH)
         int _stateTrack2Prev;             // The previous state of the sensor

         /*
         State Trackers:
```

```
        State trackers will be boolean variables to determine if systems are currently
active
        (i.e. a track switch is on/true or off/false)
        */
        boolean _stateSwitchMainline;          // Tracks if the Mainline switch solenoid
is active (true)
        boolean _stateSwitchOffline;           // Tracks if the Offline switch solenoid
is active (true)
        /*
        Time Trackers:
        Time trackers are unsigned long variables used in conjunction with the millis()
command
        in order to produce non-blocking code

        NOTE: Unless agreed upon by all team members, delay() should never be used.
        */

};

#endif

/*
  Pod.cpp - Library for controlling Spartan Superway 1/12 scale Maker Faire exhibition
pods
  Created by Cory Ostermann (Controls Team Lead), Man Ho, Randall Morioka, Eriberto
Velazquez, and Anthony Vo
  Spartan Superway 2013-2014 Controls Team
*/

// Arduino Pins
#define MOTOR_A1     3             // The M1 pin for the motor driver
#define MOTOR_A2     2
//#define MOTOR_B1   5             // The M2 pin for the motor driver
//#define MOTOR_B2   4
#define SOLENOID_MAINLINE   4      // The pin to manipulate the transistor attached to the
Mainline switch solenoid (left)
#define SOLENOID_OFFLINE    5      // The pin to manipulate the transistor attached to the
Offline switch solenoid (right)
//#define M2_ENCODERA
//#define M2_ENCODERB
#define       TRIG   6
#define ECHO  7
#define TRACK_SENSOR1       A1
#define TRACK_SENSOR2       A0
#define SPI_INTERRUPT       9

// Additonal Properties
#define WHEEL_DIAMETER      1.875  // The diameter of the bogie wheels in inches
#define MAX_DUTY     75                    // The maximum allowable duty cycle (normal
cruise speed)
#define MIN_DUTY     60                    // The minimum duty cycle the pod will travel at
before stopping completely
#define HALF_DUTY    60                    // The duty cycle for entering or leaving a
station
#define MAX_DISTANCE 50            // The limit when the pod will reduce speed to account
for detected objects (TBD)
#define MIN_DISTANCE 25            // The limit when the pod will apply the Emergency
Brake (i.e. stop the motors) (TBD)
```

```
#define MAINLINE     1
#define OFFLINE            0
#define READY        1
#define NOT_READY    0
#define THRESHOLD    30                  // Threshold for the reflective sensors
#define LEADING_EDGE 1             // Leading Edge of the track marker
#define TRAILING_EDGE       0             // Trailing Edge of the track marker

#include "Arduino.h"
#include "Pod.h"
//#include "SPI.h"
//#include "SD.h"                  // The Adadfruit library for using an SD card for data-
logging
//#include "RTClib.h"              // The Adafruit library for using a real-time clock to
timestamp data in data-logging
#include "NewPing.h"

Pod::Pod(int identifier)
{
       _podID = identifier; // Give the pod it's identifier
       _podStatus = READY;         // Set the pod's initial state
       _locationNode = 0;          // Set the initial location at the Depot
       _locationLine = OFFLINE;
       _locationAlt = 2;
       _range = 0;
       _stateTrack1Prev = LOW;
       _stateTrack2Prev = LOW;

       /*
       Setup the pins for actuators and sensors
       */
       // Motors
       pinMode(MOTOR_A1, OUTPUT);
       digitalWrite(MOTOR_A1, LOW);
       pinMode(MOTOR_A2, OUTPUT);
       digitalWrite(MOTOR_A2, LOW);
       //pinMode(MOTOR_B1, OUTPUT);
       //digitalWrite(MOTOR_B1, LOW);
       //pinMode(MOTOR_B2, OUTPUT);
       //digitalWrite(MOTOR_B2, LOW);
       // Solenoids
       pinMode(SOLENOID_MAINLINE, OUTPUT);
       digitalWrite(SOLENOID_MAINLINE, LOW);
       pinMode(SOLENOID_OFFLINE, OUTPUT);
       digitalWrite(SOLENOID_OFFLINE, LOW);
       // Motor Encoders
       //pinMode(M1_ENCODERA, INPUT);
       //pinMode(M1_ENCODERB, INPUT);
       //pinMode(M2_ENCODERA, INPUT);
       //pinMode(M2_ENCODERB, INPUT);
       // Ultrasonic Sensor
       pinMode(TRIG, OUTPUT);
       pinMode(ECHO, INPUT);
       // Track Sensor(s)
       pinMode(TRACK_SENSOR1, INPUT);
       pinMode(TRACK_SENSOR2, INPUT);

}
```

```
void Pod::statusWrite(int status_num)
{
	_podStatus = status_num;
}

int Pod::statusRead()
{
	return _podStatus;
}

int Pod::destinationRead()  // Returns the pods current destination (node)
{
	return _destination;
}

void Pod::destinationWrite(int station_id)	// Sets the pods next destination (node)
{
	switch (station_id)
	{
	case 1:
		_destination = 0;
		break;
	case 2:
		_destination = 2;
		break;
	case 3:
		_destination = 5;
		break;
	default:
		break;
	}
}

int Pod::locationRead()		// Returns the pods current location
{
	return _locationNode;
}

void Pod::locationUpdate()  // Increments the pod's location as it reads tick marks
{
	// Get readings from the Track Sensors
	int readingSensor1 = analogRead(TRACK_SENSOR1);
	int readingSensor2 = analogRead(TRACK_SENSOR2);
	// Update the previous sensor states
	_stateTrack1Prev = _stateTrack1New;
	_stateTrack2Prev = _stateTrack2New;
	// Update the new sensor states based on the sensor readings
	if( readingSensor1 > THRESHOLD )
	{
		_stateTrack1New = HIGH;
	} else {
		_stateTrack1New = LOW;
	}
	if( readingSensor2 > THRESHOLD )
	{
		_stateTrack2New = HIGH;
	} else {
```

```
                _stateTrack2New = LOW;
        }

        if( (_stateTrack1New == LOW) && (_stateTrack1Prev == HIGH) )  // The Primary Track
Sensor is triggered
        {
                // Increment the location Node to the next section
                if( _locationNode == 7 )
                {
                        _locationNode = 0;
                }
                else
                {
                        _locationNode++;
                }
                // And the pod is about to enter junction
                if( _locationNode == (_destination) )           // If the next sector is
the destination node
                {
                        switchWrite(OFFLINE);                                   // Switch to
the Offline segment
                        _locationLine = OFFLINE;                        // Set location to
Offline
                }
                else if( _locationNode != (_destination) )      // If the next sector is
not the destination
                {
                        switchWrite(MAINLINE);                                  // Activate
the switch for the Mainline
                        _locationLine = MAINLINE;
                }
                else if( _locationLine == OFFLINE )      // If the pod is entering the
Mainline from Offline
                {
                        _locationLine = MAINLINE;   // Set the line to Mainline
                        switchWrite(2);                         // Turn off the switch
solenoids
                }
                else if( _locationLine == MAINLINE ) // If the pod is exiting the Mainline
section of a section with a station
                {
                        switch (_locationNode)
                        {
                        case 0:
                        case 2:
                        case 5:
                                switchWrite(2);             // Turn off the solenoid
                                break;
                        default:
                                break;
                        }       // end switch
                }       // end if
        }       // end Primary sensor check
        if( (_stateTrack2New == LOW) && (_stateTrack2Prev == HIGH) )  // The Secondary
Track Sensor is triggered
        {
                if( _locationLine == MAINLINE )          // If on the Mainline
                {
```

```
                        switchWrite(MAINLINE);                  // Toggle the switching
mechanism
                }
                if( _locationLine == OFFLINE )          // If Offline...
                {
                        switch (_locationAlt)
                        {
                        case 0:
                                switchWrite(OFFLINE);                   // Toggle the
switching mechanism
                                _locationAlt = 1;
                                break;
                        case 1:
                                if( _locationNode == _destination )
                                {
                                        _locationAlt = 2;
                                        break;
                                } else {
                                        switchWrite(OFFLINE);                   // Toggle the
switching mechanism
                                        _locationAlt = 0;
                                        break;
                                }
                        case 2:
                                _podStatus = NOT_READY;
                                break;
                        }       // end switch
                }       // end OFFLINE if
        }       // end Secondary Sensor check
}       // end locationUpdate
/*
boolean Pod::ReflectiveSensor(int sensor, int edge)     // COULD NOT GET TO WORK
{
        static int reading, newVal, prevVal;                    // Stores Reflective Object
Sensor reading
        //boolean result;           // Stores the result to be returned
        // Get relevant sensor data
        switch( sensor )
        {
        case 1:         // Primary track sensor
                reading = analogRead(TRACK_SENSOR1);
                prevVal = _stateTrack1Prev;
                break;
        case 2:         // Secondary track sensor
                reading = analogRead(TRACK_SENSOR2);
                prevVal = _stateTrack1Prev;
                break;
        default:
                break;
        }       // end sensor switch
        if (reading > THRESHOLD)    // The Reflective Object Sensor reads the felt markers
        {
                newVal = LOW;
        } else {
                newVal = HIGH;
        }
        // Update the new state variables:
        switch( sensor )
```

```
{
case 1:
	_stateTrack1New = newVal;
	break;
case 2:
	_stateTrack2New = newVal;
	break;
default:
	break;
}	// end sensor switch
if( newVal == LOW && prevVal == HIGH)	// Triggering on the leading edge
{
	prevVal = LOW;
	switch( sensor )
	{
	case 1:
		_stateTrack1Prev = prevVal;
		break;
	case 2:
		_stateTrack2Prev = prevVal;
		break;
	default:
		break;
	}	// end sensor switch
	if(edge == LEADING_EDGE)		// The leading edge was specified
	{
		return true;
	} else {
		return false;
	}
}
else if ( newVal == HIGH && prevVal == LOW)	// Triggering on the trailing edge
{
	prevVal = HIGH;
	switch( sensor )
	{
	case 1:
		_stateTrack1Prev = prevVal;
		break;
	case 2:
		_stateTrack2Prev = prevVal;
		break;
	default:
		break;
	}	// end sensor switch
	if(edge = TRAILING_EDGE)		// The trailing edge was specified
	{
		result = true;
	} else {
		result = false;
	}
}
else
{
	result = false;
}
return result;
```

```
}      // end ReflectiveSensor
*/

void Pod::switchWrite(int line)
{
	if( line == MAINLINE && SOLENOID_MAINLINE == LOW)
	{
		digitalWrite(SOLENOID_OFFLINE, LOW);
		_stateSwitchOffline = false;
		digitalWrite(SOLENOID_MAINLINE, HIGH);
		_stateSwitchMainline = true;
	}
	else if( line == MAINLINE && SOLENOID_MAINLINE == HIGH)
	{
		switchWrite(2);      // Turn off both solenoid
	}
	else if( line == OFFLINE )
	{
		digitalWrite(SOLENOID_MAINLINE, LOW);
		_stateSwitchMainline = false;
		digitalWrite(SOLENOID_OFFLINE, HIGH);
		_stateSwitchOffline = true;
	}
	else if( line == OFFLINE && SOLENOID_OFFLINE == HIGH)
	{
		switchWrite(2);      // Turn off both solenoid
	}
	else
	{
		digitalWrite(SOLENOID_OFFLINE, LOW);
		_stateSwitchOffline = false;
		digitalWrite(SOLENOID_MAINLINE, LOW);
		_stateSwitchMainline = false;
	}
}

int Pod::DetectObject()     // Use the Ultrasonic Sensor make sure there are no
obstructions
{
	NewPing sonar(TRIG, ECHO, MAX_DISTANCE);
	_range = sonar.ping_cm();
	return _range;       // If no objects are detected, return 0;
}

int Pod::speedReadPWM()
{
	return _dutyCycle;
}

void Pod::speedWritePWM(int duty_cycle)
{
	_dutyCycle = duty_cycle;
}

void Pod::Cruise()
{
	if ((_range != 0) && (_range < MIN_DISTANCE))    // if the pod is within critical
range
```

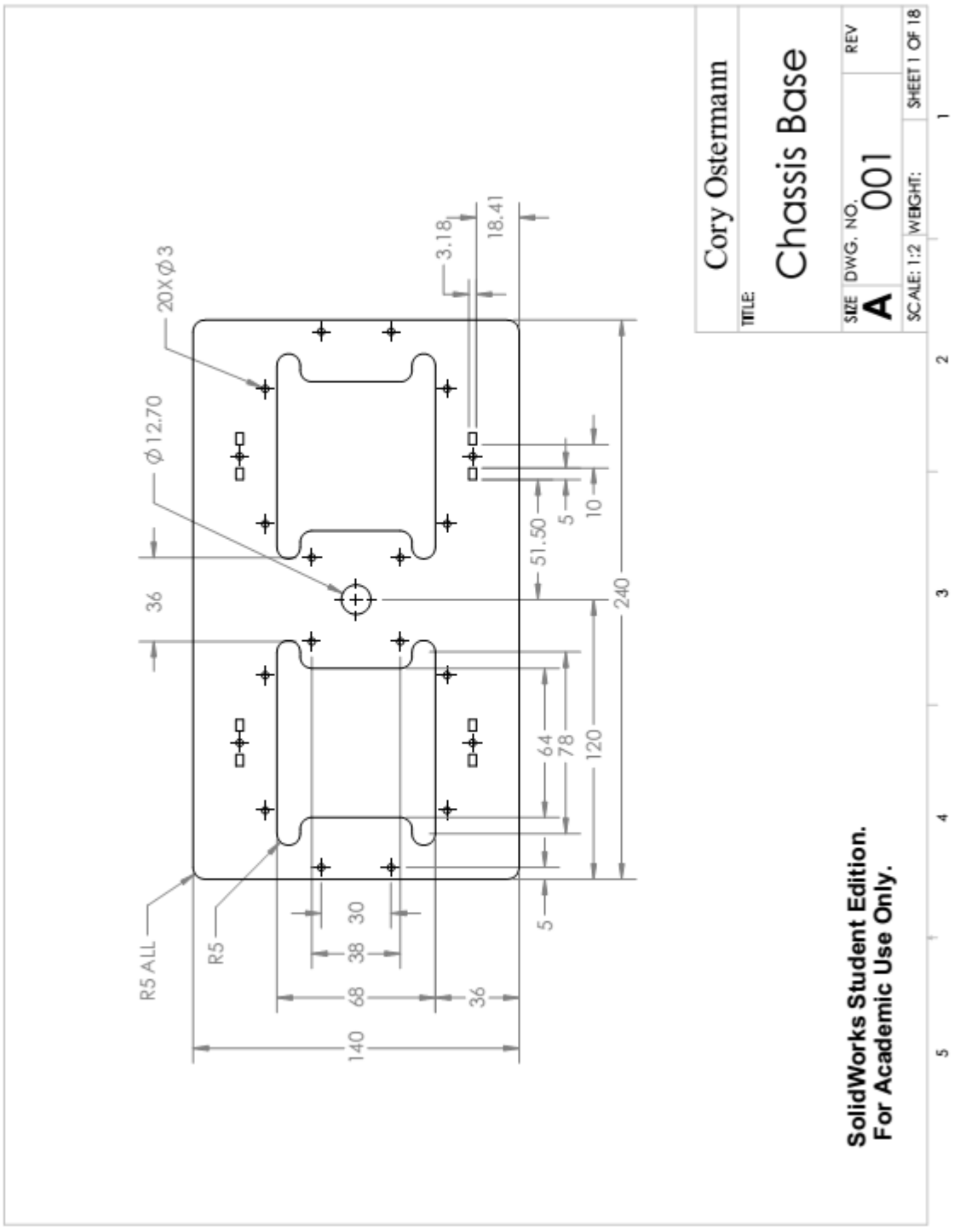
```
	{
		// Activate the "emergency brakes" (i.e. stop motors)
		digitalWrite(MOTOR_A1, HIGH);
		digitalWrite(MOTOR_A2, HIGH);
		//digitalWrite(MOTOR_B1, HIGH);
		//digitalWrite(MOTOR_B2, HIGH);
		_dutyCycle = 0;                                    // set the duty cycle to
zero
	}
	else if ((_locationAlt == 2) && (_podStatus != READY)) // if the pod
arriving/sitting at a station
	{
		// Stop the motors
		digitalWrite(MOTOR_A1, HIGH);
		digitalWrite(MOTOR_A2, HIGH);
		//digitalWrite(MOTOR_B1, HIGH);
		//digitalWrite(MOTOR_B2, HIGH);
		_dutyCycle = 0;                                    // set the duty cycle to
zero
	}
	else if ((_range >= MIN_DISTANCE) && (_range <= MAX_DISTANCE))// If the pod
detects an object
	{
		_dutyCycle = map(_range, MIN_DISTANCE, MAX_DISTANCE, MIN_DUTY, MAX_DUTY);
	// Reduce speed based on the distance to the object
	}
	else if (_locationAlt == 1)       // If the pod is entering the Offline segment of
track
	{
		_dutyCycle = HALF_DUTY;           // Reduce duty cycle by half
	}
	else if ((_range == 0) && (_locationAlt == 0))  // If the pod is leaving the
Offline segment
	{
		_dutyCycle = MAX_DUTY;            // Increase duty cycle to merge onto the
Mainline
	}
	// Manipulate the motors according to the current duty cycle
	analogWrite(MOTOR_A1, _dutyCycle);
	//analogWrite(MOTOR_B2, _dutyCycle);
	digitalWrite(MOTOR_A2, LOW);
	//digitalWrite(MOTOR_B1, LOW);

}
```

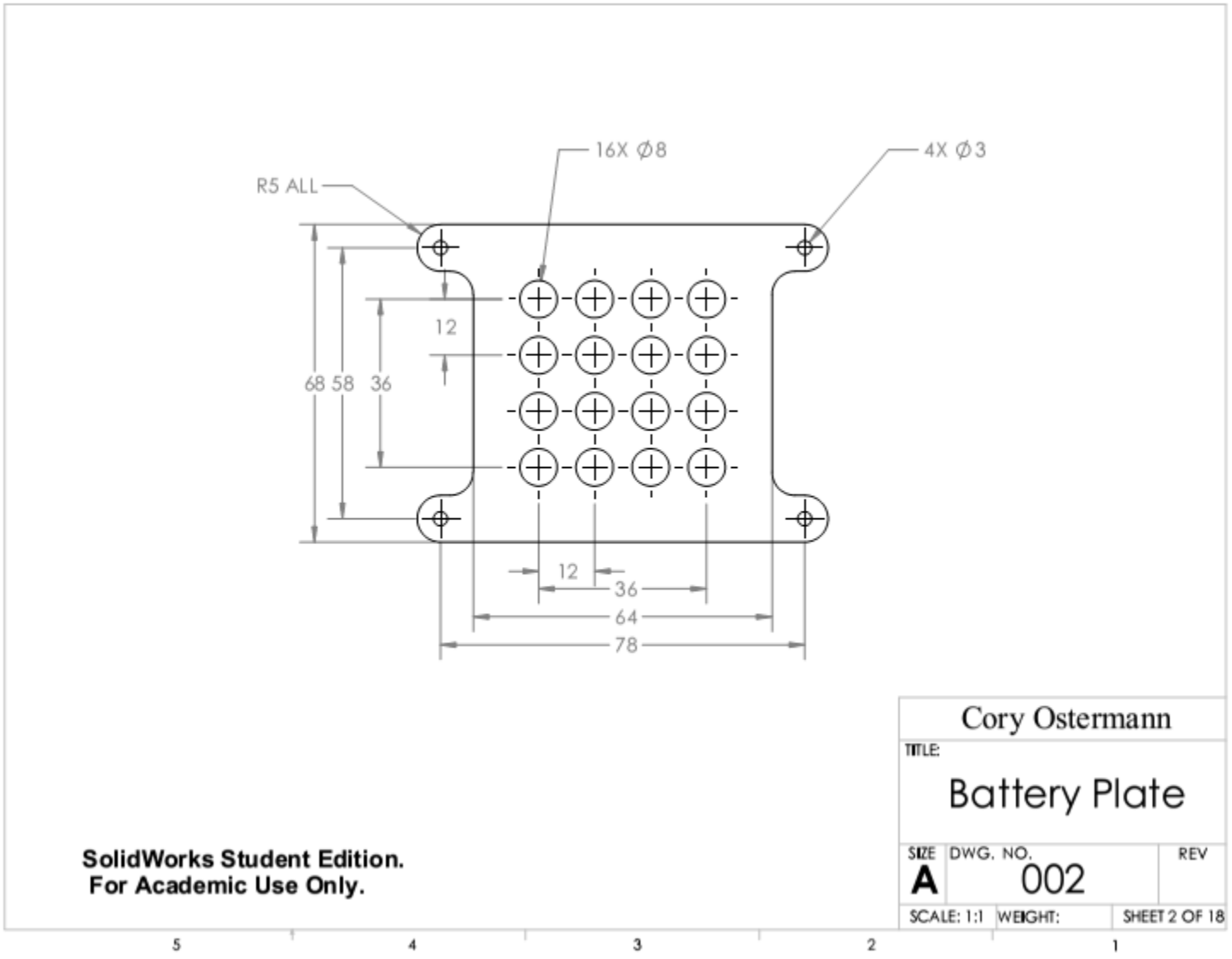
Appendix U: Arduino SPI Sketch

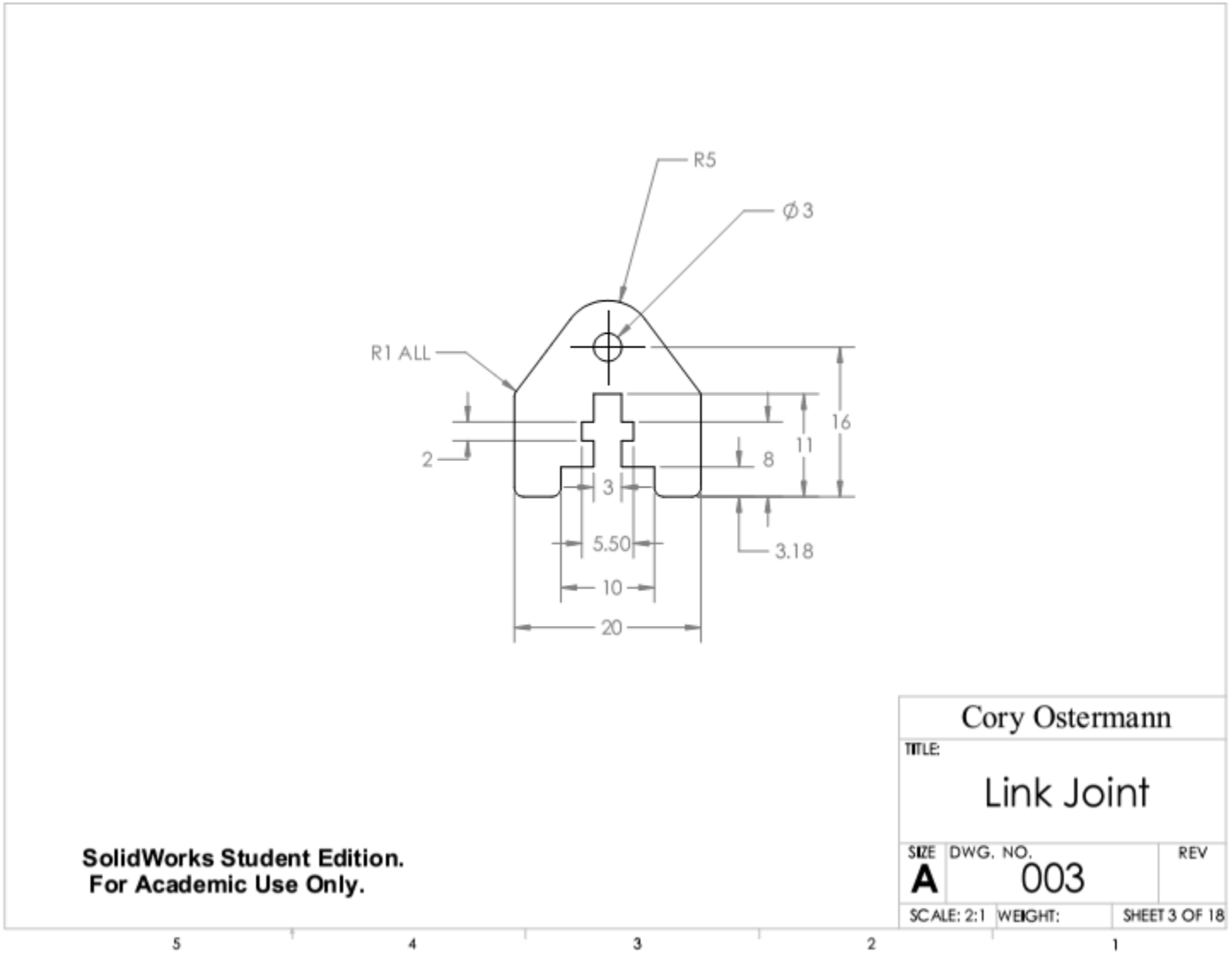
```
#include <SPI.h>

int buf [3];
volatile byte pos;
volatile boolean process_it;
int gasPedal = 0;
unsigned long timerStop;


void setup (void)
{
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);
  pinMode(2, OUTPUT);
  digitalWrite(2, LOW);
  Serial.begin (9600);   // debugging
  // have to send on master in, *slave out*
  pinMode(MISO, OUTPUT);

  // turn on SPI in slave mode
  SPCR |= _BV(SPE);

  // get ready for an interrupt
  pos = 0;   // buffer empty
  process_it = false;

  // now turn on interrupts
  SPI.attachInterrupt();

}  // end of setup


// SPI interrupt routine
ISR (SPI_STC_vect)
{
byte c = SPDR;  // grab byte from SPI Data Register
Serial.print("ISR\n");
  // add to buffer if room
  if (pos < sizeof buf)
    {
    buf [pos++] = c;

    // example: newline means time to process buffer
    if (c == 0x00)
      process_it = true;

    }  // end of room available

}  // end of interrupt routine SPI_STC_vect

// main loop - wait for flag set in interrupt routine
void loop (void)
{
  if (process_it)
    {
```

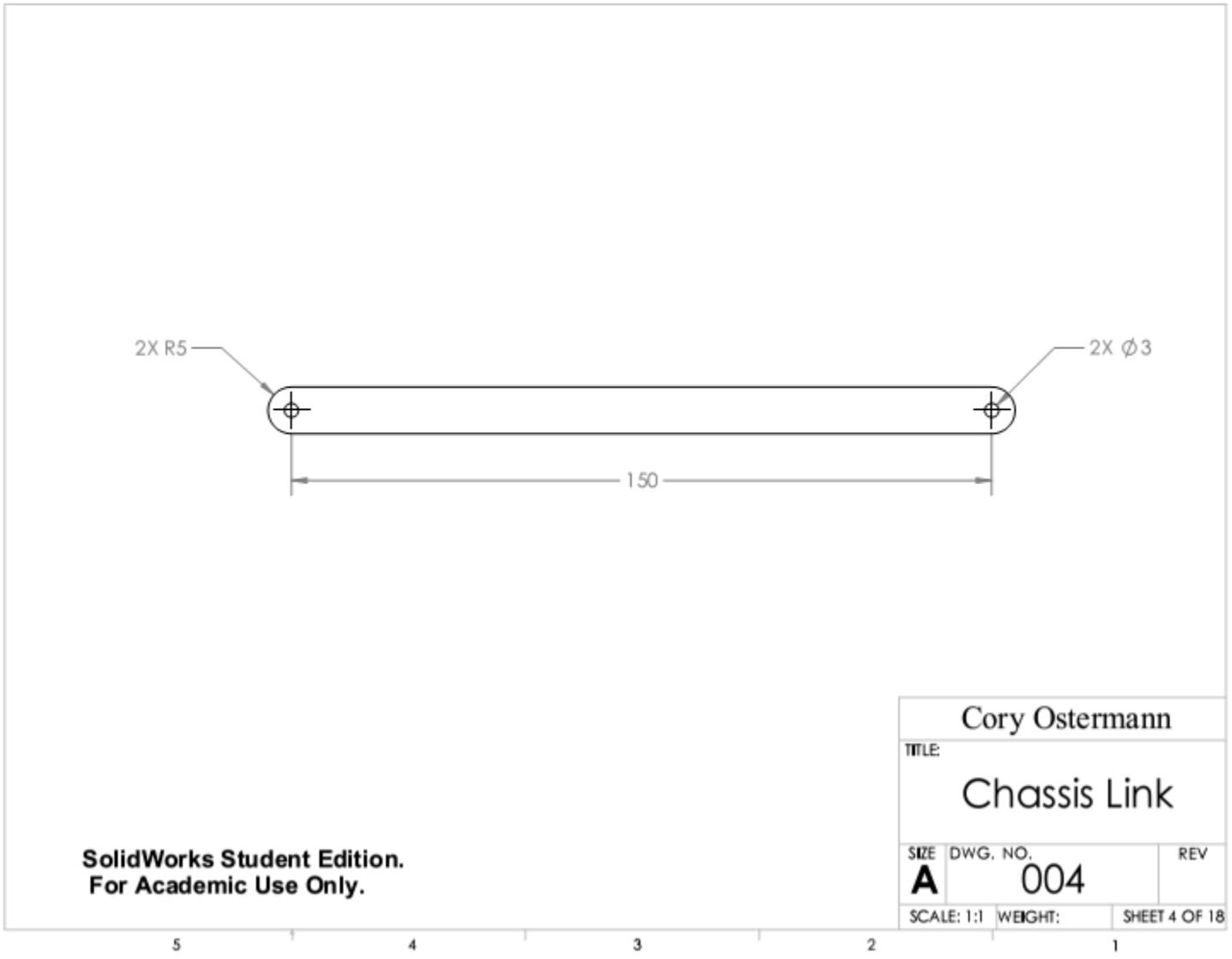
```
      gasPedal = buf[0];
      Serial.println(gasPedal);
      timerStop = millis();
      /*
      for(int i=0; i<(pos-1); i++)
      {
        Serial.print (buf[i]);
        Serial.print(" ");
      }
      */
      pos = 0;
      process_it = false;
    }  // end of flag set
  if( gasPedal )
  {
    //Serial.println("Gas pedal is true!");
    analogWrite(3, 70);
    //digitalWrite(3, HIGH);
  }
  if( !gasPedal )
  {
    digitalWrite(3, LOW);
  }
//   Serial.print("HI\n");
}  // end of loop
```

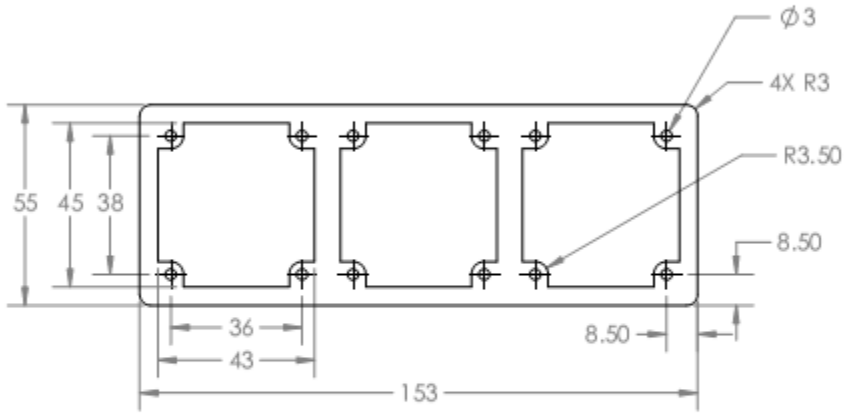
Appendix V: 1/12-Scale Part Drawings




16X Ø8
4X Ø3
R5 ALL
12
68 58 36
12
36
64
78
SolidWorks Student Edition.
For Academic Use Only.
Cory Ostermann
TITLE:
Battery Plate
SIZE
A
DWG. NO.
002
REV
SCALE: 1:1
WEIGHT:
SHEET 2 OF 18
5
4
3
2
1


R5
Ø3
R1 ALL
16
11
8
2
3
5.50
10
20
3.18
SolidWorks Student Edition.
For Academic Use Only.
Cory Ostermann
TITLE:
Link Joint
SIZE
A
DWG. NO.
003
REV
SCALE: 2:1
WEIGHT:
SHEET 3 OF 18
5
4
3
2
1


2X R5
2X Ø3
150
SolidWorks Student Edition.
For Academic Use Only.
Cory Ostermann
TITLE:
Chassis Link
SIZE
A
DWG. NO.
004
REV
SCALE: 1:1
WEIGHT:
SHEET 4 OF 18
5
4
3
2
1



Cory Ostermann

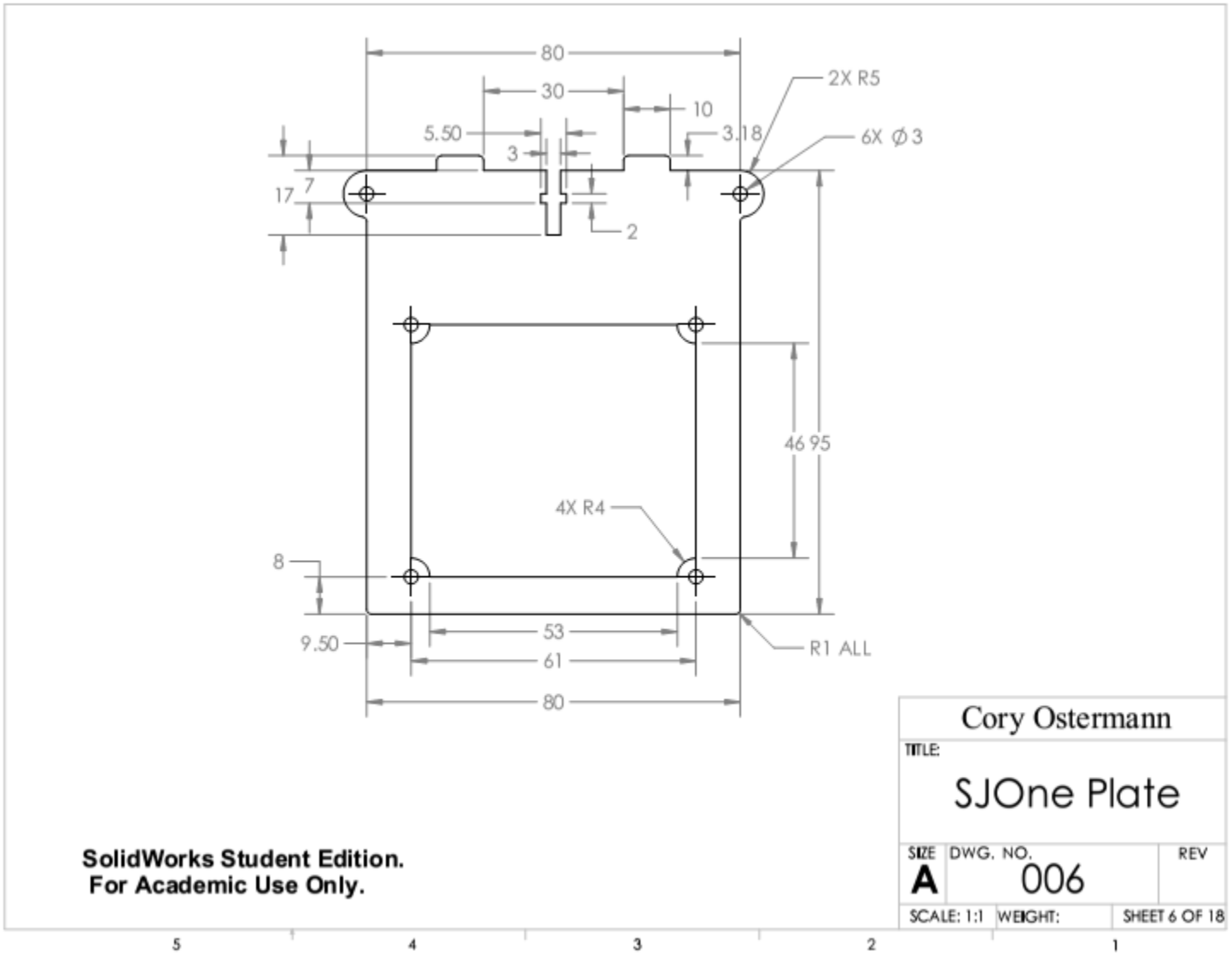
TITLE:

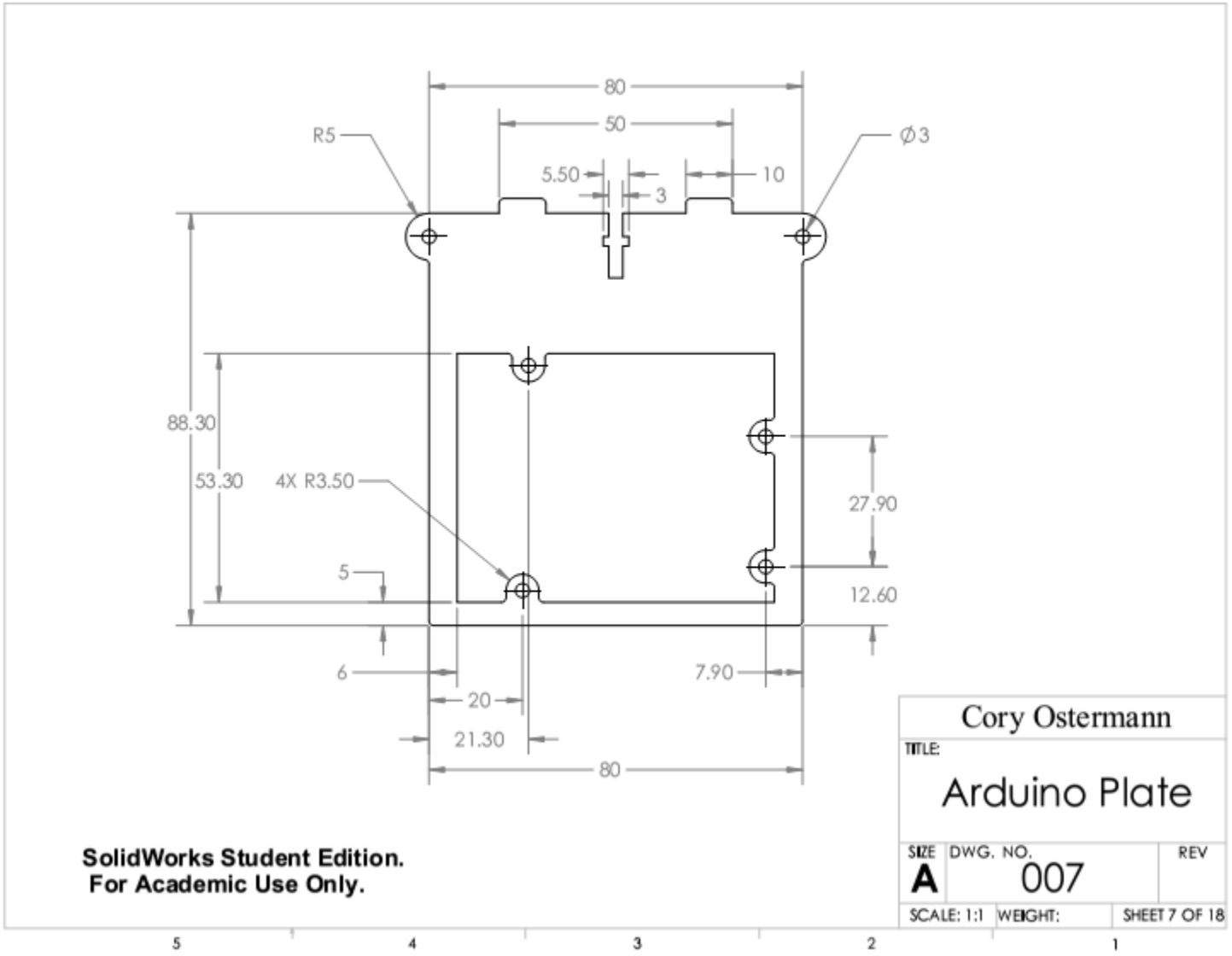
PCB Mount

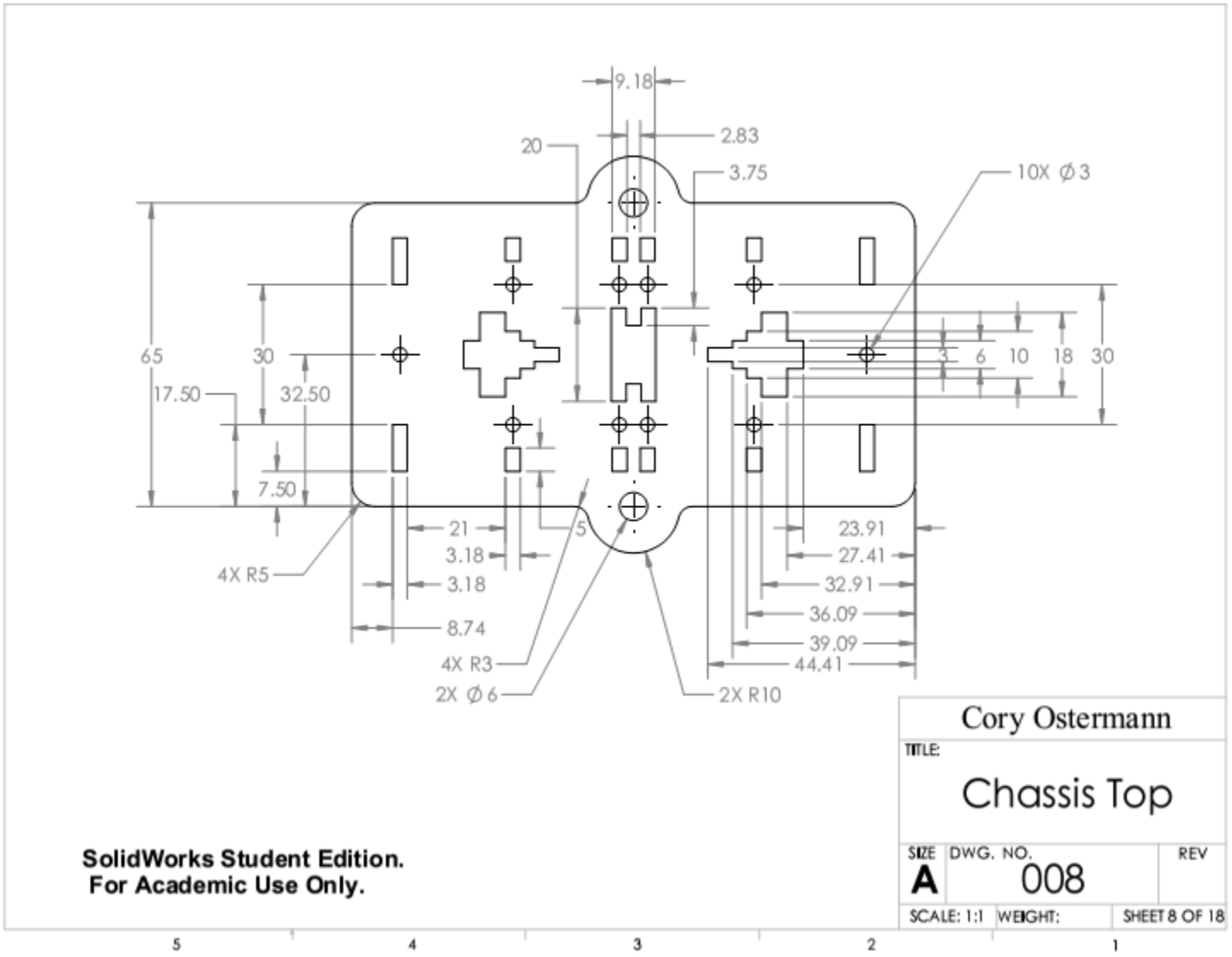
SIZE	DWG. NO.	REV
A	005	

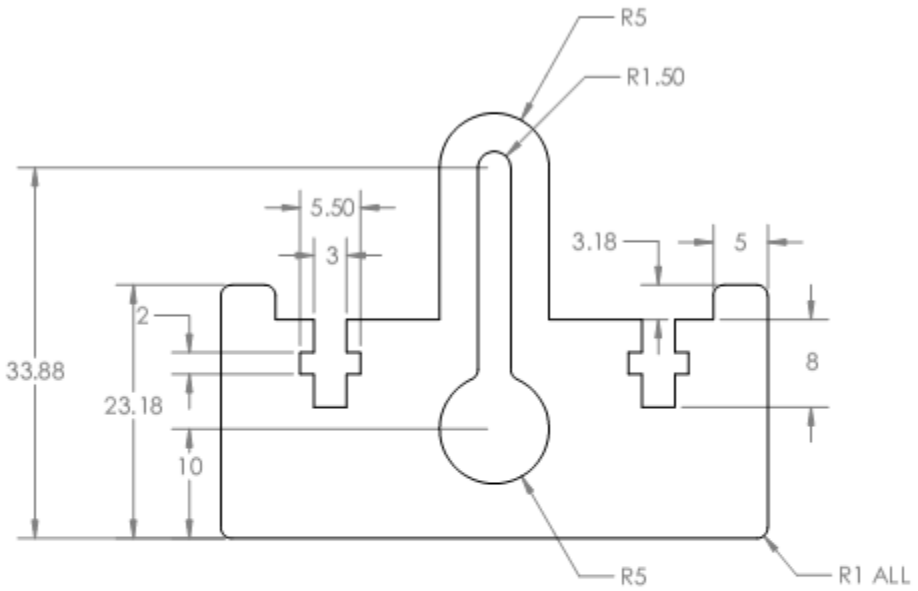
SCALE: 2:3 WEIGHT: SHEET 5 OF 18

5 4 3 2 1


80
30
10
5.50
3
3.18
2X R5
6X Ø3
17
7
2
46
95
4X R4
8
9.50
53
61
R1 ALL
80
SolidWorks Student Edition.
For Academic Use Only.
Cory Ostermann
TITLE:
SJOne Plate
SIZE
A
DWG. NO.
006
REV
SCALE: 1:1
WEIGHT:
SHEET 6 OF 18
5
4
3
2
1


80
50
R5
Ø3
5.50
3
10
88.30
53.30
4X R3.50
27.90
5
12.60
6
7.90
20
21.30
80
SolidWorks Student Edition.
For Academic Use Only.
Cory Ostermann
TITLE:
Arduino Plate
SIZE
A
DWG. NO.
007
REV
SCALE: 1:1
WEIGHT:
SHEET 7 OF 18
5
4
3
2
1


9.18
2.83
20
3.75
10X Ø3
65
30
17.50
32.50
7.50
3
6
10
18
30
21
3.18
3.18
8.74
5
4X R5
4X R3
2X Ø6
2X R10
23.91
27.41
32.91
36.09
39.09
44.41
SolidWorks Student Edition.
For Academic Use Only.
Cory Ostermann
TITLE:
Chassis Top
SIZE
A
DWG. NO.
008
REV
SCALE: 1:1
WEIGHT:
SHEET 8 OF 18
5
4
3
2
1



Cory Ostermann

TITLE:

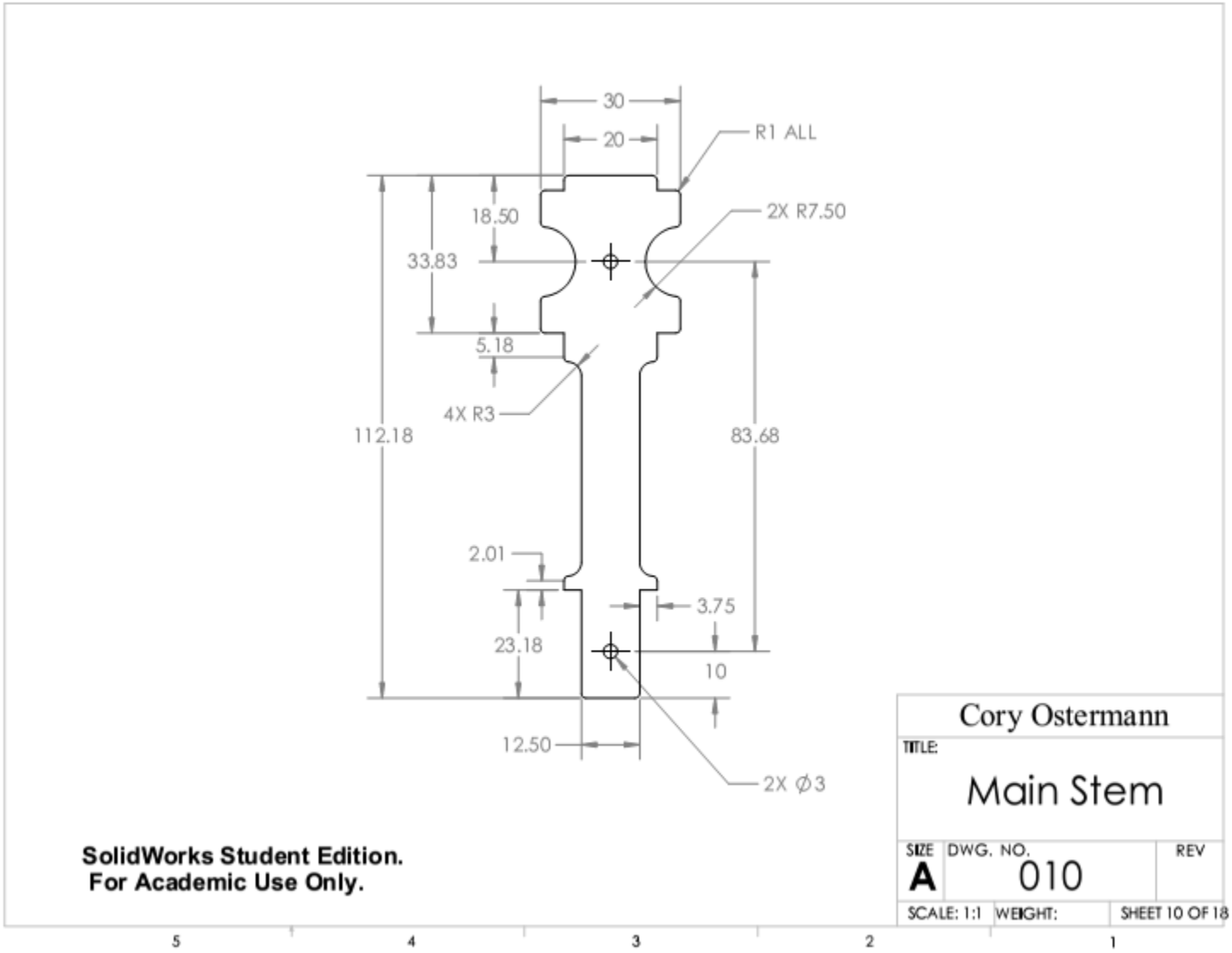
Sensor Plate

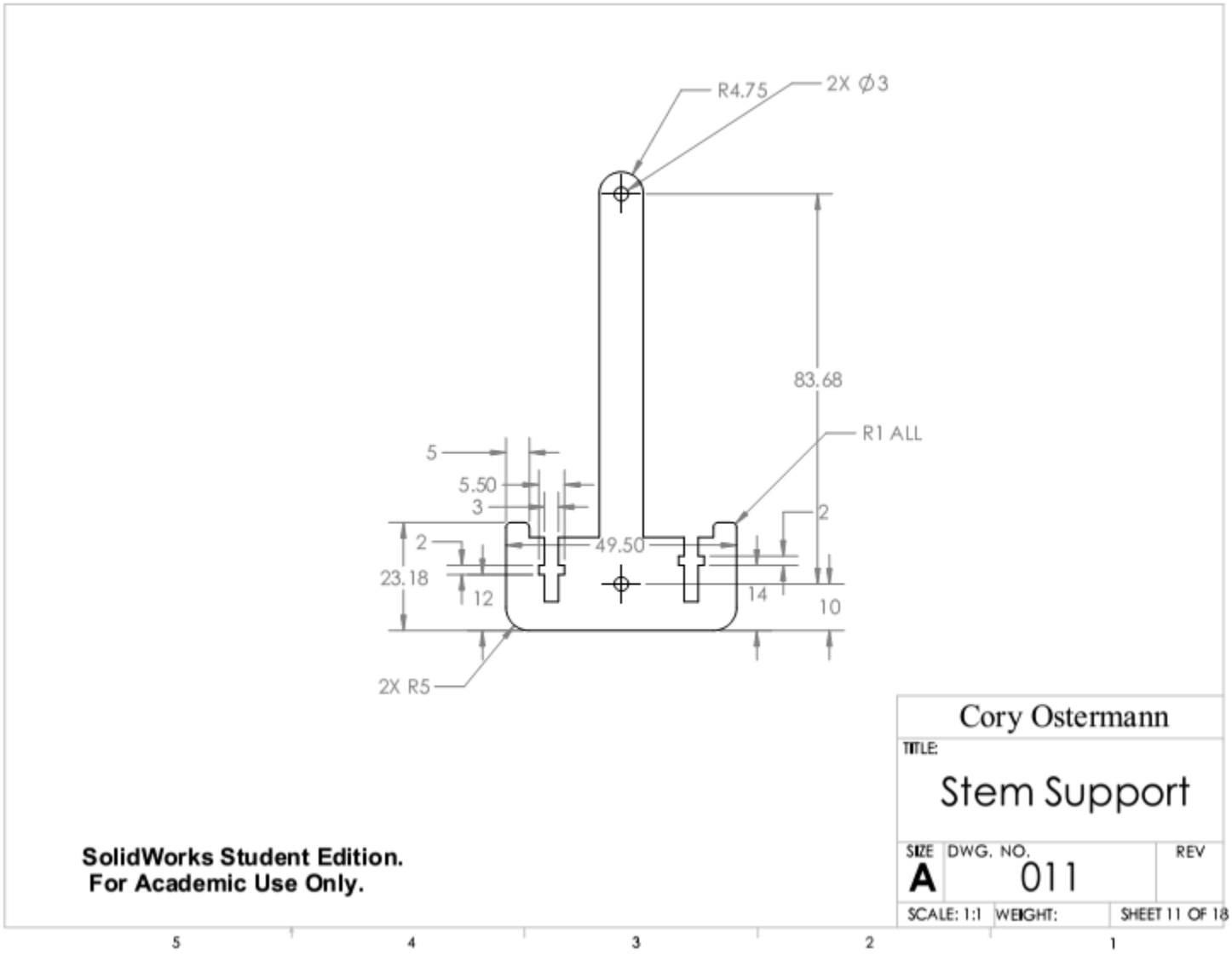
SIZE	DWG. NO.	REV
A	009	

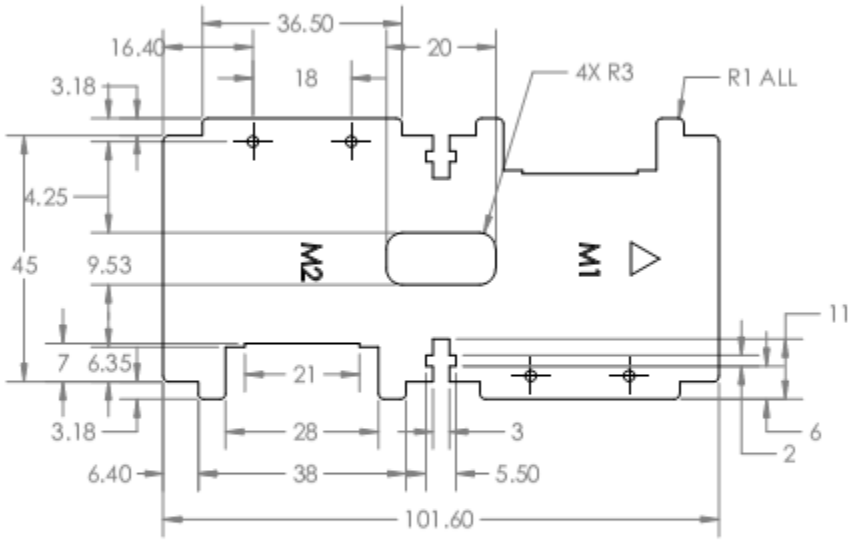
SCALE: 2:1 WEIGHT: SHEET 9 OF 18

SolidWorks Student Edition.
For Academic Use Only.

5 4 3 2 1


30
20
R1 ALL
2X R7.50
18.50
33.83
5.18
4X R3
112.18
83.68
2.01
3.75
23.18
10
12.50
2X Ø3
SolidWorks Student Edition.
For Academic Use Only.
Cory Ostermann
TITLE:
Main Stem
SIZE
A
DWG. NO.
010
REV
SCALE: 1:1
WEIGHT:
SHEET 10 OF 18
5
4
3
2
1


R4.75
2X Ø3
83.68
R1 ALL
5
5.50
3
2
23.18
12
49.50
2
14
10
2X R5
Cory Ostermann
TITLE:
Stem Support
SIZE
A
DWG. NO.
011
REV
SCALE: 1:1
WEIGHT:
SHEET 11 OF 18
SolidWorks Student Edition.
For Academic Use Only.
5
4
3
2
1



Cory Ostermann

TITLE:

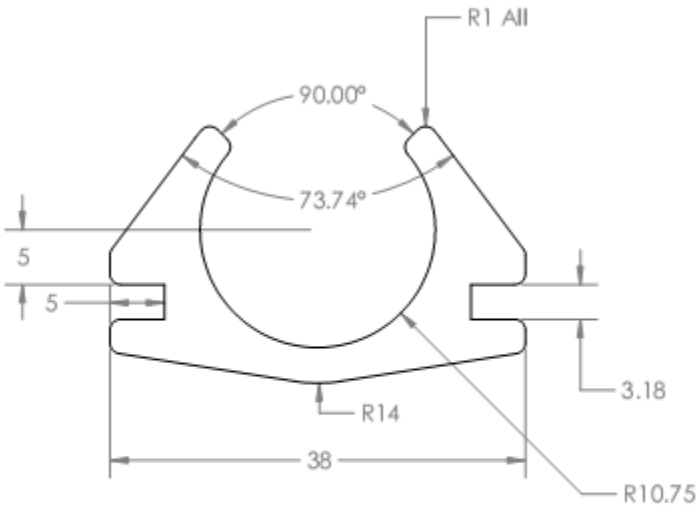
Bogie Base

SIZE	DWG. NO.	REV
A	012	

SCALE: 1:1 WEIGHT: SHEET 12 OF 18

SolidWorks Student Edition.
For Academic Use Only.

5 4 3 2 1



Cory Ostermann

TITLE:

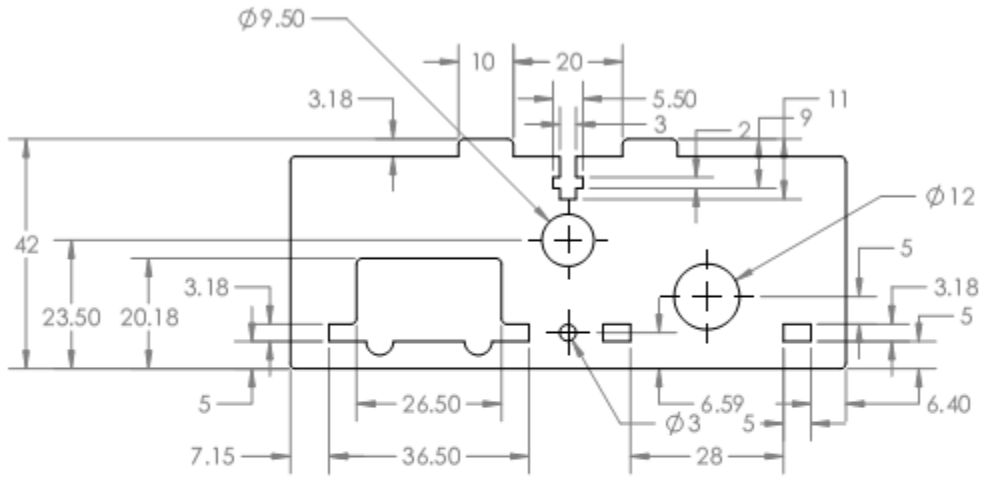
Bearing Plate

SIZE	DWG. NO.	REV
A	013	

SCALE: 2:1 WEIGHT: SHEET 13 OF 18

SolidWorks Student Edition.
For Academic Use Only.

5 4 3 2 1



Cory Ostermann

TITLE:

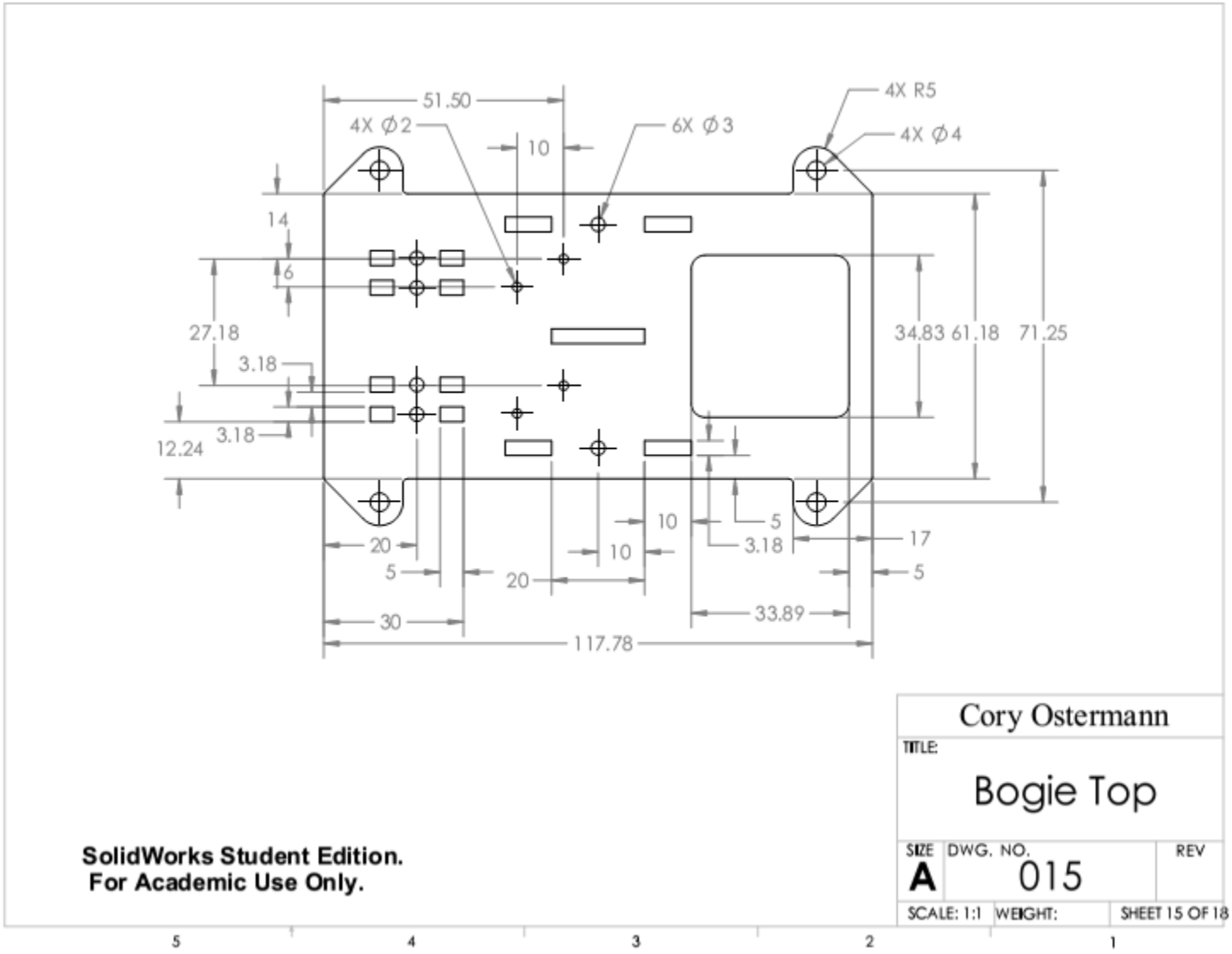
Bogie Side

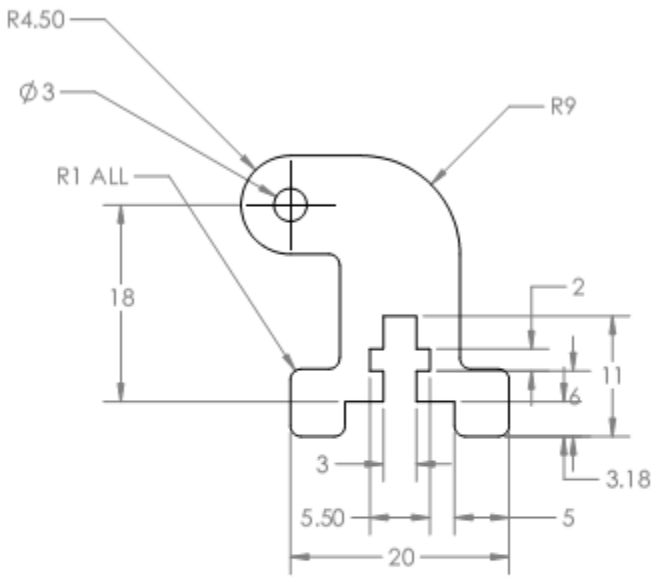
SIZE	DWG. NO.	REV
A	014	

SCALE: 1:1 WEIGHT: SHEET 14 OF 18

SolidWorks Student Edition.
For Academic Use Only.

5 4 3 2 1


51.50
4X Ø2
10
6X Ø3
4X R5
4X Ø4
14
6
27.18
3.18
3.18
12.24
34.83
61.18
71.25
10
5
20
10
3.18
17
5
20
5
33.89
30
117.78
Cory Ostermann
TITLE:
Bogie Top
SIZE
A
DWG. NO.
015
REV
SCALE: 1:1
WEIGHT:
SHEET 15 OF 18
SolidWorks Student Edition.
For Academic Use Only.
5
4
3
2
1



Cory Ostermann

TITLE:

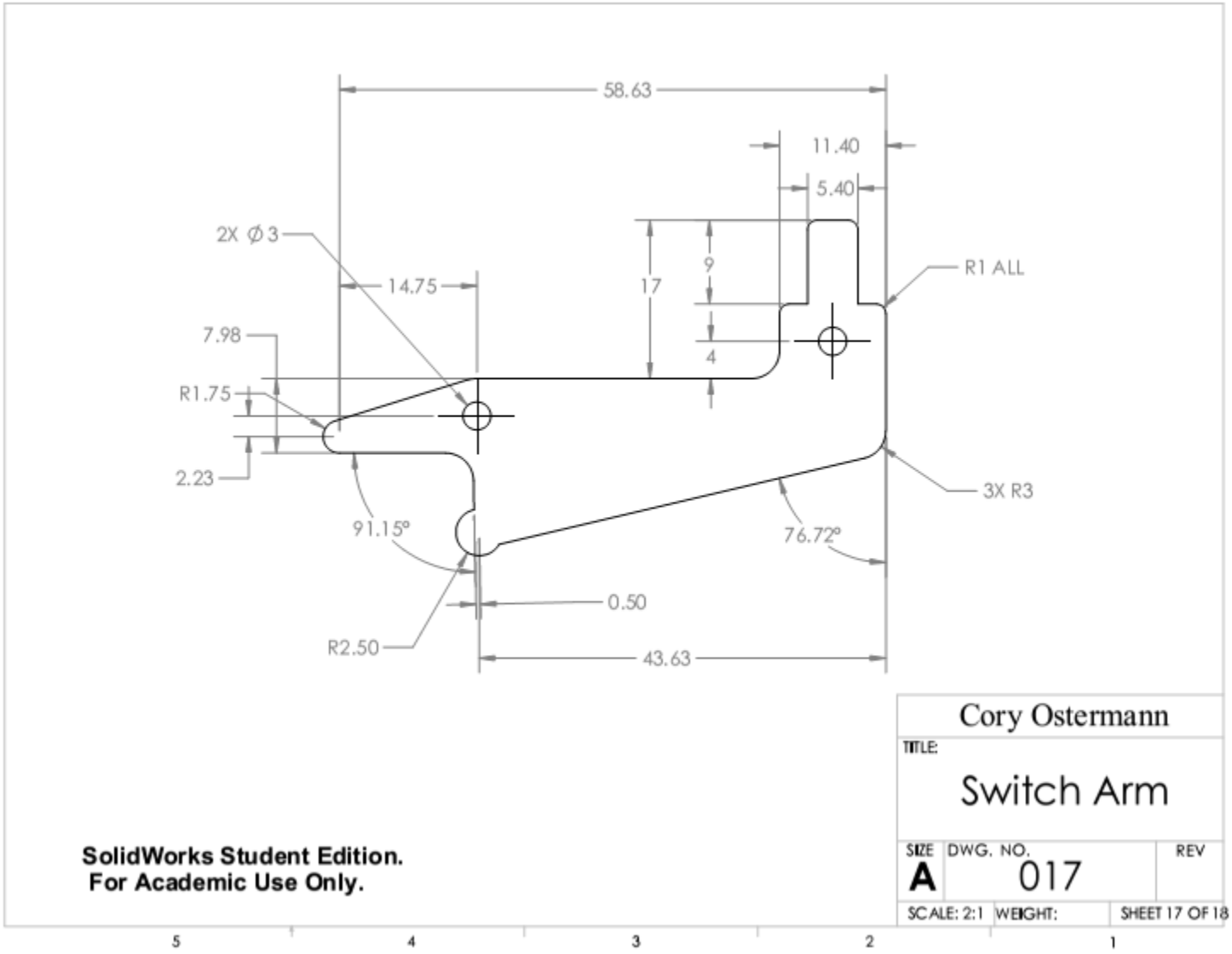
Switch Joint

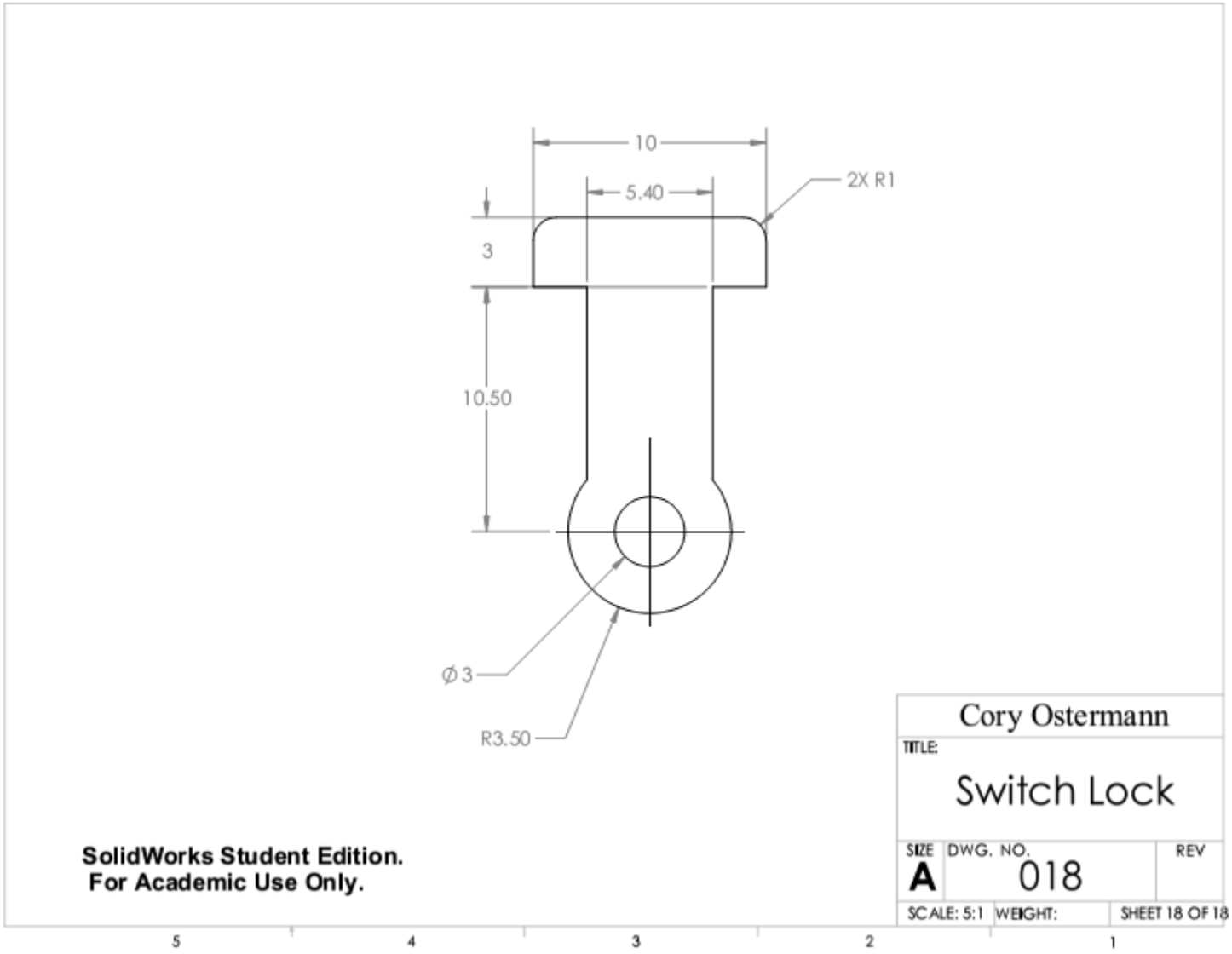
SIZE	DWG. NO.	REV
A	016	

SCALE: 2:1 WEIGHT: SHEET 16 OF 18

SolidWorks Student Edition.
For Academic Use Only.

5 4 3 2 1


58.63
11.40
5.40
2X Ø3
14.75
17
9
4
R1 ALL
7.98
R1.75
2.23
91.15°
76.72°
3X R3
0.50
R2.50
43.63
Cory Ostermann
TITLE:
Switch Arm
SIZE
A
DWG. NO.
017
REV
SCALE: 2:1
WEIGHT:
SHEET 17 OF 18
SolidWorks Student Edition.
For Academic Use Only.
5
4
3
2
1


10
5.40
2X R1
3
10.50
Ø 3
R3.50
Cory Ostermann
TITLE:
Switch Lock
SIZE
A
DWG. NO.
018
REV
SCALE: 5:1
WEIGHT:
SHEET 18 OF 18
SolidWorks Student Edition.
For Academic Use Only.
5
4
3
2
1

Appendix W: Bill of Materials

Microcontrollers	**Quantity**	**Distributor**	**Unit Price**	**Total**
SJOne	1	SJSU CMPE	$80.00	$80.00
Arduino Uno	1	Jameco	$27.95	$27.95
Proto-Shield	1	Jameco	$6.95	$6.95
Actuators & Accessories				
30:1 Micro Metal Gearmotor HP with Extended Motor Shaft	2	Pololu	$16.95	$33.90
Micro Metal Gearmotor Bracket Pair	1	Pololu	$4.99	$4.99
Motor Drivier Carrier	1	Pololu	$4.95	$4.95
5V solenoid	2	Sparkfun	$4.95	$9.90
JST Vertical Connector	2	Sparkfun	$0.95	$1.90
TIP102 Transistor	2	Jameco	$0.45	$0.90
1N4004 Diode	2	Jameco	$0.05	$0.10
Sensors				
Optical Encoder Pair Kit (5V)	1	Pololu	$8.95	$8.95
Reflective Phototransistor	2	Jameco	$4.95	$9.90
Ultrasonic Sensor	1	Amazon	$12.00	$12.00
Hardware				
1-7/8" Wheels	4	BaneBots	$2.50	$10.00
3mm Shaft Wheel Hubs	4	BaneBots	$4.00	$16.00
608 Bearings	8	Amazon	$1.50	$12.00
1/8" Acrylic Sheet	17" x 22"	TAP Plastics	$10.00	$10

Laser-Cut Components (from 1/8" Acrylic Sheet)	Quantity	Drawing Number
Chassis Base	1	001
Battery Plate	2	002
Link Joint	4	003
Chassis Link	8	004
PCB Mount	1	005
SJOne Plate	1	006
Arduino Plate	1	007
Chassis Top	1	008
Sensor Plate	2	009
Main Stem	1	010
Stem Support	2	011
Bogie Base	1	012
Bearing Plate	4	013
Bogie Side	2	014
Bogie Top	1	015
Switch Joint	4	016
Switch Arm	2	017
Switch Lock	2	018

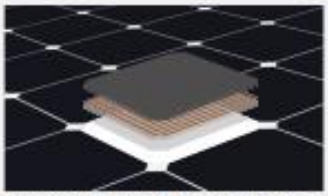
Appendix X: SunPower Data Sheet

SUNPOWER

MORE ENERGY. FOR LIFE.™

X-SERIES SOLAR PANELS

- **21.5% efficiency**
  Ideal for roofs where space is at a premium or where future expansion might be needed.
- **Maximum performance**
  Designed to deliver the most energy in demanding real world conditions, in partial shade and hot rooftop temperatures.[1, 2, 3]
- **Premium aesthetics**
  SunPower® Signature™ Black X-Series panels blend harmoniously into your roof. The most elegant choice for your home.



Maxeon® Solar Cells: Fundamentally better.
Engineered for performance, designed for durability.

Engineered for peace of mind
Designed to deliver consistent, trouble-free energy over a very long lifetime.[4,5]

Designed for durability
The SunPower Maxeon Solar Cell is the only cell built on a solid copper foundation. Virtually impervious to the corrosion and cracking that degrade Conventional Panels.[4,5]

Same excellent durability as E-Series panels.
#1 Ranked in Fraunhofer durability test.[10]
100% power maintained in Atlas 25+ comprehensive PVDI Durability test.[11]

UNMATCHED PERFORMANCE, RELIABILITY & AESTHETICS

SIGNATURE™ BLACK
X21 - 335 PANEL

X21 - 345 PANEL

X21 SERIES

HIGHEST EFFICIENCY[6]

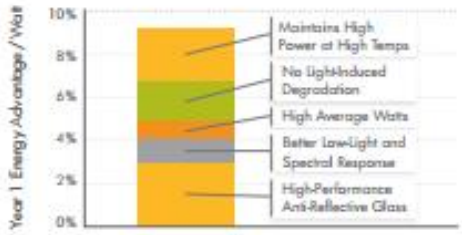
Generate more energy per square foot

X-Series residential panels convert more sunlight to electricity producing 44% more power per panel,[1] and 75% more energy per square foot over 25 years.[3,4]

HIGHEST ENERGY PRODUCTION[7]

Produce more energy per rated watt

High year one performance delivers 8-10% more energy per rated watt.[3] This advantage increases over time, producing 21% more energy over the first 25 years to meet your needs.[4]



sunpowercorp.com

SUNPOWER

MORE ENERGY. FOR LIFE.™

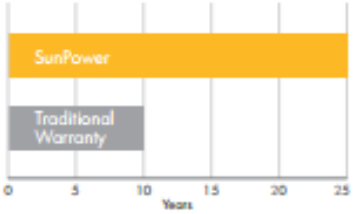
X-SERIES SOLAR PANELS

SUNPOWER OFFERS THE BEST COMBINED POWER AND PRODUCT WARRANTY

POWER WARRANTY

More guaranteed power: 95% for first 5 years, -0.4%/yr. to year 25.[8]

PRODUCT WARRANTY



Combined Power and Product Defect 25 year coverage that includes panel replacement costs.[9]

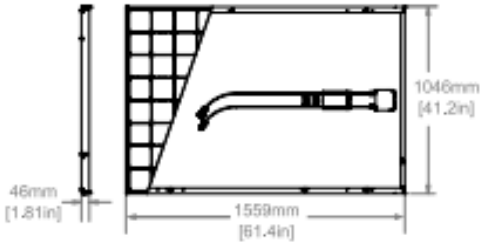
ELECTRICAL DATA	X21-335-BLK	X21-345
Nominal Power[12] (Pnom)	335 W	345 W
Power Tolerance	+5/–0%	+5/–0%
Avg. Panel Efficiency[12]	21.1%	21.5%
Rated Voltage (Vmpp)	57.3 V	57.3 V
Rated Current (Impp)	5.85 A	6.02 A
Open-Circuit Voltage (Voc)	67.9 V	68.2 V
Short-Circuit Current (Isc)	6.23 A	6.39 A
Maximum System Voltage	600 V UL ; 1000 V IEC	
Maximum Series Fuse	20 A	
Power Temp Coef. (Pmpp)	–0.30% / °C	
Voltage Temp Coef. (Voc)	–167.4 mV / °C	
Current Temp Coef. (Isc)	3.5 mA / °C	

OPERATING CONDITION AND MECHANICAL DATA	
Temperature	– 40°F to +185°F (– 40°C to +85°C)
Max load	Wind: 50 psf, 2400 Pa, 245 kg/m² front & back Snow: 112 psf, 5400 Pa, 550kg/m² front
Impact resistance	1 inch (25 mm) diameter hail at 52 mph (23 m/s)
Appearance	Class A+
Solar Cells	96 Monocrystalline Maxeon Gen III Cells
Tempered Glass	High Transmission Tempered Anti-Reflective
Junction Box	IP-65 Rated
Connectors	MC4 Compatible
Frame	Class 1 black anodized, highest AAMA Rating
Weight	41 lbs (18.6 kg)

TESTS AND CERTIFICATIONS	
Standard tests	UL 1703, IEC 61215, IEC 61730
Quality tests	ISO 9001:2008, ISO 14001:2004
EHS Compliance	RoHS, OHSAS 18001:2007, lead-free
Ammonia test	IEC 62716
Salt Spray test	IEC 61701 (passed maximum severity)
PID test	Potential-Induced Degradation free: 1000V [10]
Available listings	CEC, UL, TUV, MCS

REFERENCES:

1 All comparisons are SPR-X21-345 vs. a representative conventional panel: 240W, approx. 1.6 m², 15% efficiency.
2 PVEvolution Labs "SunPower Shading Study," Feb 2013.
3 Typically 8-10% more energy per watt, BEW/DNV Engineering "SunPower Yield Report," Jan 2013, with CFV Solar Test Lab Report #12063, Jan 2013 temp. coef. calculation.
4 SunPower 0.25%/yr degradation vs. 1.0%/yr conv. panel. Campeau, Z. et al. "SunPower Module Degradation Rate," SunPower white paper, Feb 2013; Jordan, Dirk "SunPower Test Report," NREL, Oct 2012.
5 "SunPower Module 40-Year Useful Life" SunPower white paper, Feb 2013. Useful life is 99 out of 100 panels operating at more than 70% of rated power.
6 Higher than E Series which is highest of all 2600 panels listed in Photon Int'l, Feb 2012.
7 1% more energy than E-Series panels, 8% more energy than the average of the top 10 panel companies tested in 2012 (151 panels, 102 companies), Photon Int'l, Mar 2013.
8 Compared with the top 15 manufacturers. SunPower Warranty Review, Feb 2013.
9 Some exclusions apply. See warranty for details.
10 X-Series same as E-Series, 5 of top 8 panel manufacturers were tested by Fraunhofer ISE, "PV Module Durability Initiative Public Report," Feb 2013.
11 Compared with the non-stress-tested control panel. X-Series same as E-Series, tested in Atlas 25+ Durability test report, Feb 2013.
12 Standard Test Conditions (1000 W/m² irradiance, AM 1.5, 25° C).
13 Based on average of measured power values during production.



See http://www.sunpowercorp.com/facts for more reference information.
For further details, see extended datasheet: www.sunpowercorp.com/datasheets Read safety and installation instructions before using this product.

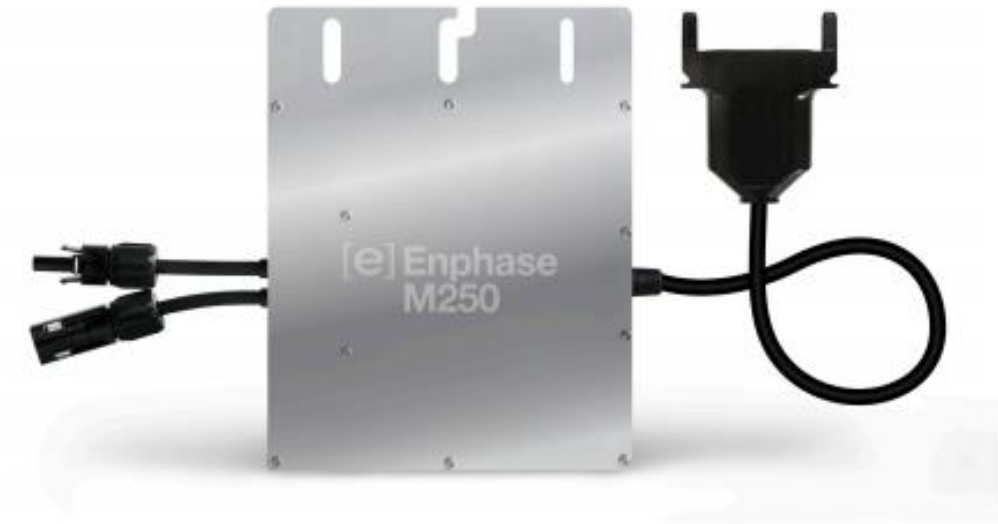
sunpowercorp.com

Document # 504808 Rev A /LTR_EN

Appendix Y: Enphase M250 Microinverter Data Sheet

Enphase® **Microinverters**

Enphase® **M250**



The **Enphase® M250 Microinverter** delivers increased energy harvest and reduces design and installation complexity with its all-AC approach. With the M250, the DC circuit is isolated and insulated from ground, so **no Ground Electrode Conductor (GEC) is required for the microinverter.** This further simplifies installation, enhances safety, and saves on labor and materials costs.

The Enphase M250 integrates seamlessly with the Engage® Cable, the Envoy® Communications Gateway™, and Enlighten®, Enphase's monitoring and analysis software.

PRODUCTIVE

- Optimized for higher-power modules
- Maximizes energy production
- Minimizes impact of shading, dust, and debris

SIMPLE

- No GEC needed for microinverter
- No DC design or string calculation required
- Easy installation with Engage Cable

RELIABLE

- 4th-generation product
- More than 1 million hours of testing and millions of units shipped
- Industry-leading warranty, up to 25 years

[e] enphase® ENERGY

C SP US

[redacted]

Enphase® **M250** Microinverter // DATA

INPUT DATA (DC)	**M250-60-2LL-S22/S23/S24**	
Recommended input power (STC)	210 - 300 W	
Maximum input DC voltage	48 V	
Peak power tracking voltage	27 V - 39 V	
Operating range	16 V - 48 V	
Min/Max start voltage	22 V / 48 V	
Max DC short circuit current	15 A	
Max input current	10 A	
OUTPUT DATA (AC)	**@208 VAC**	**@240 VAC**
Peak output power	250 W	250 W
Rated (continuous) output power	240 W	240 W
Nominal output current	1.15 A (A rms at nominal duration)	1.0 A (A rms at nominal duration)
Nominal voltage/range	208 V / 183-229 V	240 V / 211-264 V
Nominal frequency/range	60.0 / 57-61 Hz	60.0 / 57-61 Hz
Extended frequency range*	57-62.5 Hz	57-62.5 Hz
Power factor	>0.95	>0.95
Maximum units per 20 A branch circuit	24 (three phase)	16 (single phase)
Maximum output fault current	850 mA rms for 6 cycles	850 mA rms for 6 cycles
EFFICIENCY		
CEC weighted efficiency, 240 VAC	96.5%	
CEC weighted efficiency, 208 VAC	96.0%	
Peak inverter efficiency	96.5%	
Static MPPT efficiency (weighted, reference EN50530)	99.4 %	
Night time power consumption	65 mW max	
MECHANICAL DATA		
Ambient temperature range	-40°C to +65°C	
Operating temperature range (internal)	-40°C to +85°C	
Dimensions (WxHxD)	171 mm x 173 mm x 30 mm (without mounting bracket)	
Weight	2.0 kg	
Cooling	Natural convection - No fans	
Enclosure environmental rating	Outdoor - NEMA 6	
FEATURES		
Compatibility	Compatible with 60-cell PV modules.	
Communication	Power line	
Integrated ground	The DC circuit meets the requirements for ungrounded PV arrays in NEC 690.35. Equipment ground is provided in the Engage Cable. No additional GEC or ground is required. Ground fault protection (GFP) is integrated into the microinverter.	
Monitoring	Free lifetime monitoring via Enlighten software	
Compliance	UL1741/IEEE1547, FCC Part 15 Class B, CAN/CSA-C22.2 NO. 0-M91, 0.4-04, and 107.1-01	

* Frequency ranges can be extended beyond nominal if required by the utility

To learn more about Enphase Microinverter technology, visit enphase.com

[e] enphase ENERGY

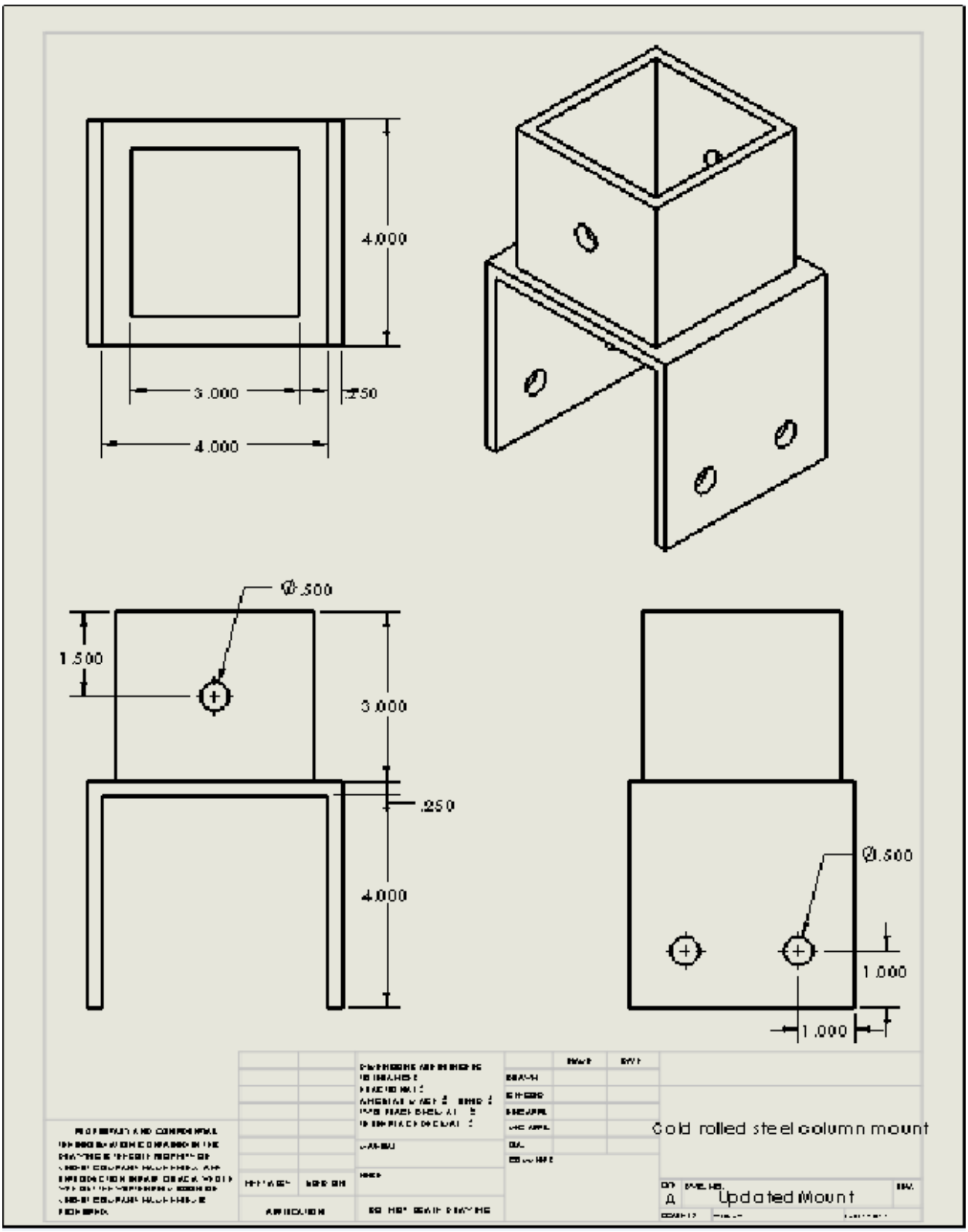
Appendix Z: Aluminum 6063-T5 Data Sheet

Aluminum 6063-T5

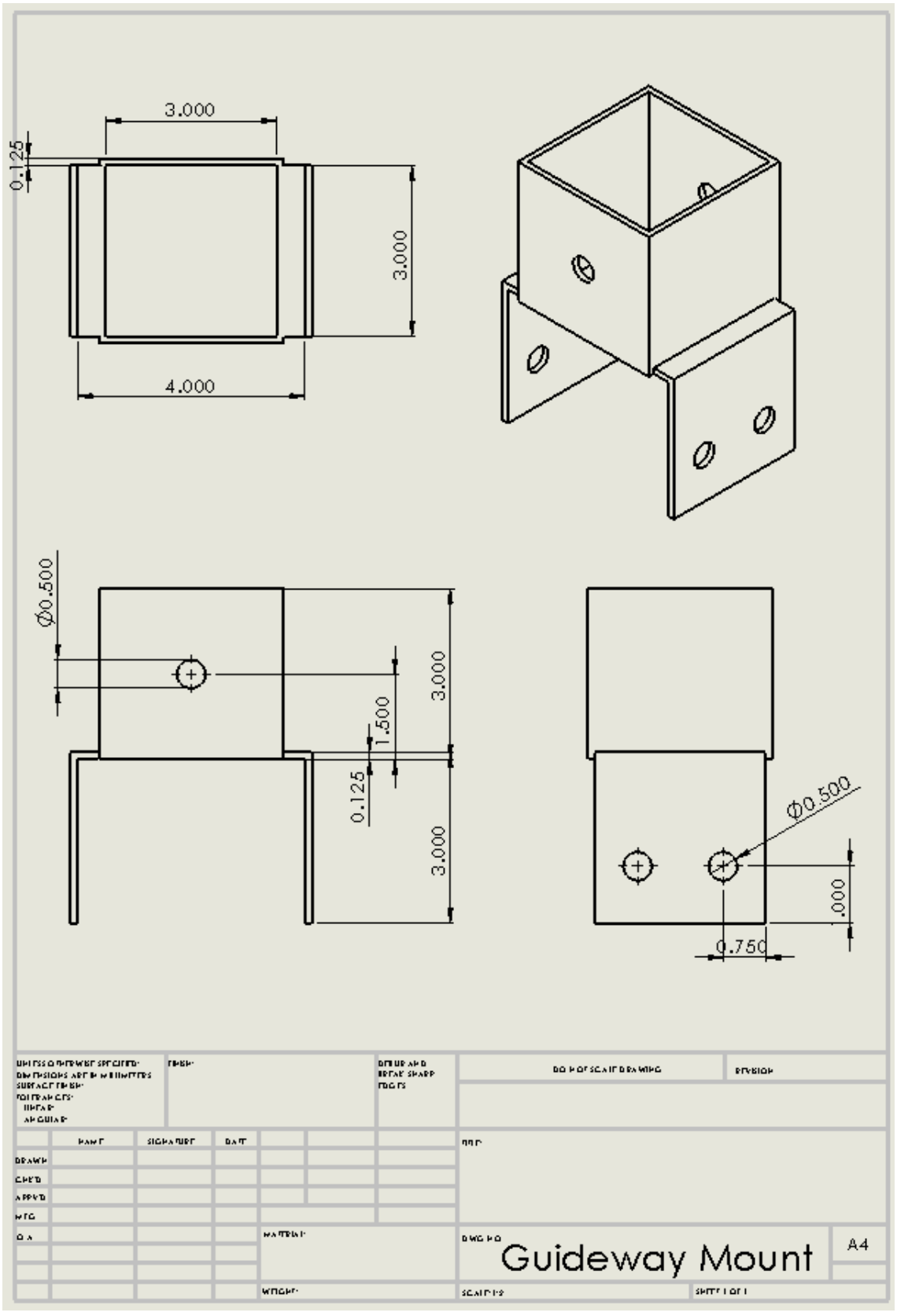
Component	Wt. %	Component	Wt. %	Component	Wt. %
Al	Max 97.5	Mg	0.45 - 0.9	Si	0.2 - 0.6
Cr	Max 0.1	Mn	Max 0.1	Ti	Max 0.1
Cu	Max 0.1	Other, each	Max 0.05	Zn	Max 0.1
Fe	Max 0.35	Other, total	Max 0.15		

Physical Properties	Metric	English	Comments
Density	2.7 g/cc	0.0975 lb/in³	AA; Typical
Mechanical Properties			
Hardness, Brinell	60	60	AA; Typical; 500 g load; 10 mm ball
Ultimate Tensile Strength	186 MPa	27000 psi	AA; Typical
Tensile Yield Strength	145 MPa	21000 psi	AA; Typical
Elongation at Break	12 %	12 %	AA; Typical; 1/16 in. (1.6 mm) Thickness
Modulus of Elasticity	68.9 GPa	10000 ksi	AA; Typical; Average of tension and compression. Compression modulus is about 2% greater than tensile modulus.
Poisson's Ratio	0.33	0.33	
Fatigue Strength	68.9 MPa	10000 psi	AA; 500,000,000 cycles completely reversed stress; RR Moore machine/specimen
Shear Modulus	25.8 GPa	3740 ksi	
Shear Strength	117 MPa	17000 psi	AA; Typical
Electrical Properties			
Electrical Resistivity	3.16e-006 ohm-cm	3.49e-006 ohm-cm	AA; Typical at 68°F
Thermal Properties			
CTE, linear 68°F	23.4 µm/m-°C	13 µin/in-°F	AA; Typical; Average over 68-212°F range.
CTE, linear 250°C	25.6 µm/m-°C	14.2 µin/in-°F	Average over the range 20-300ºC
Heat Capacity	0.9 J/g-°C	0.215 BTU/lb-°F	
Thermal Conductivity	209 W/m-K	1450 BTU-in/hr-ft²-°F	AA; Typical at 77°F
Melting Point	616 - 654 °C	1140 - 1210 °F	AA; Typical range based on typical composition for wrought products 1/4 inch thickness or greater
Solidus	616 °C	1140 °F	AA; Typical
Liquidus	654 °C	1210 °F	AA; Typical
Processing Properties			
Annealing Temperature	413 °C	775 °F	hold at temperature for 2 to 3 hr; cool at 50 °F per hour from 775 to 500 °F
Solution Temperature	521 °C	970 °F	
Aging Temperature	182 °C	360 °F	hold at temperature for 1 hr

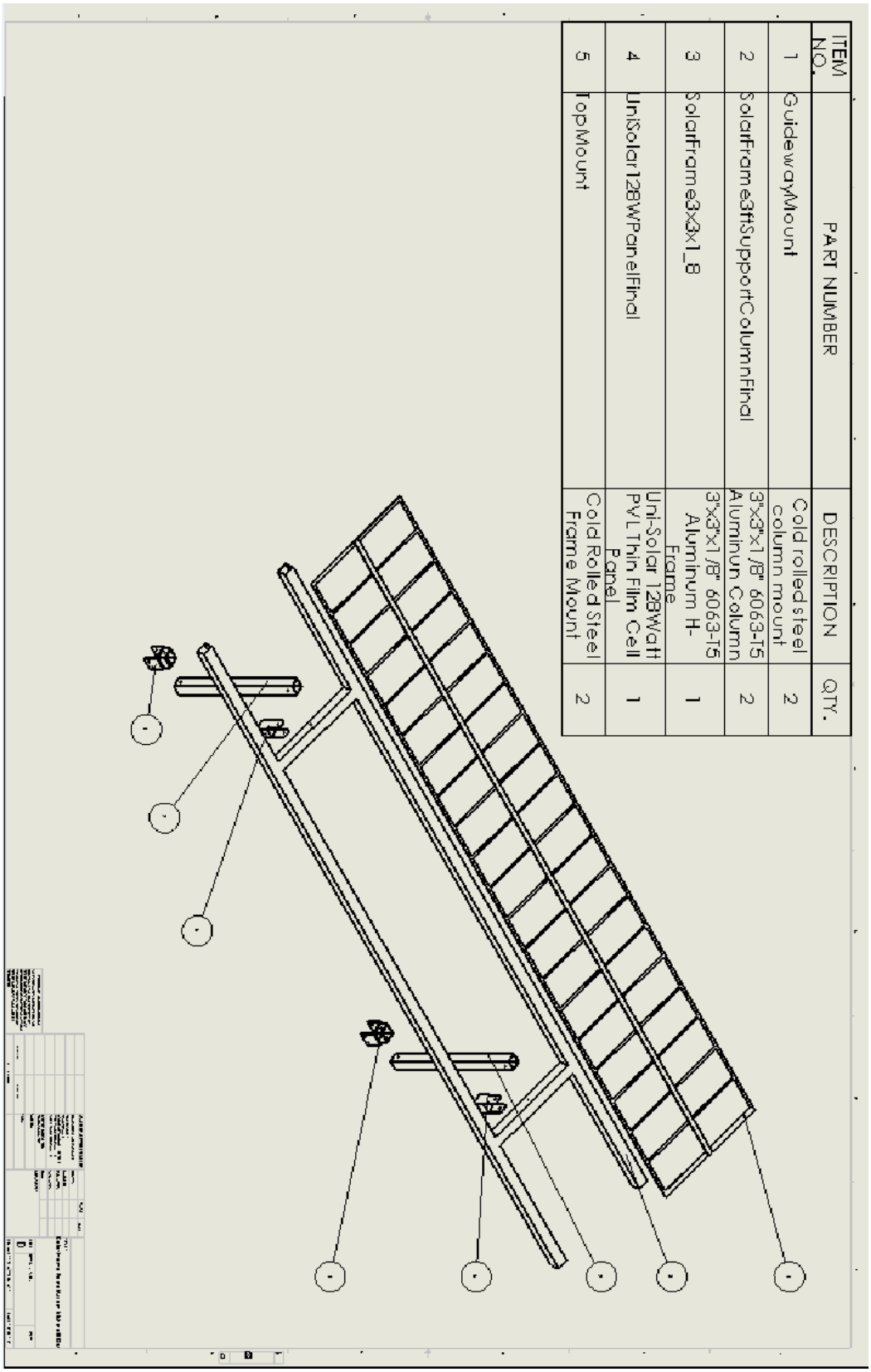
Appendix AA: Solar – Cold Rolled Steel Column Mount



Appendix AB: Solar – Guideway Mount



Appendix AC: Solar Frame Assembly Drawing



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	GuidewayMount	Cold rolled steel column mount	2
2	SolarFrame3ftSupportColumnFinal	3"x3"x1/8" 6063-T5 Aluminum Column	2
3	SolarFrame3x3x1_8	3"x3"x1/8" 6063-T5 Aluminum H-Frame	1
4	UniSolar128WPanelFinal	Uni-Solar 128Watt PVL Thin Film Cell Panel	1
5	TopMount	Cold Rolled Steel Frame Mount	2

Appendix AD: Solar Team Bill of Materials

Part	Unit Price	Qty	Cost	Details
Combiner Box	$150.00	1	$150.00	MNPV6 Combiner Box Product #: 8910041
Grid Tie Inverter	$200.00	1	$200.00	Enphase M250 - Micro Inverter, 208/240 VAC for MC4 Product #: 2930488
Sunpower T5 327W Panel	$0.00	1	$0.00	Unisolar 128 Watt Flexible Solar Panel PV Laminate
Framer Material	$500.00	1	$500.00	6063 T-6 Aluminum Frame and Tower Mounts
Wiring	$300.00	1	$300.00	12 Gauge Primary Wire by Del City