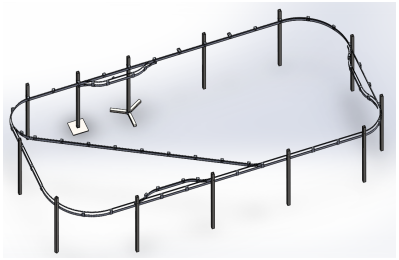
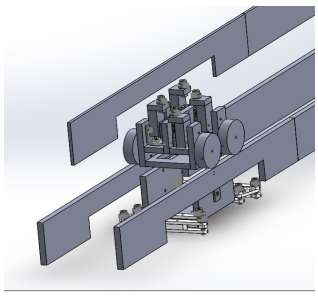
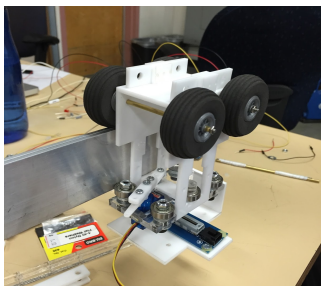
SPARTAN SUPERWAY

1/12th Scale Model Report on Design and Prototyping

San Jose State University
Charles W. Davidson – College of Engineering
Fall 2014
Dr. Furman

Jared Besson, John Fidel, Matthew Lewinsky, Cheng-Hsien Liu, Ryan Luc, Jake Pichel, Christopher Rose, Pavel Smrz, Oscar Suen, Vaibhav Tank

San José State UNIVERSITY

Executive Summary

The Spartan Superway project is a multi-year and interdisciplinary project that is currently in its third year in an academic environment. The Spartan Superway project is comprised of the development of an Automated Transit Network (ATN) that is powered by solar energy. For the 2014-15 academic year, the project is composed of 4 subteams. These include the Full Scale, 1/12th Scale, Cabin, and Solar teams. This report focuses on 1/12th Scale model and its development during the first semester. Reports for the other teams are available in separate documents.

The previous model provided a good foundation to test the system controls components and algorithms. The old model was never fully utilized, which could be related to some of its faults. The construction of the guideway was robust but was not held to tolerances that were tight enough and therefore resulted in degraded reliability. When initially constructed, the model functioned, but when the model was moved to a new location it had to be meticulously put back together so that it would function. Hours were spent filing the track at Maker Faire last year to ensure the bogie could completely traverse the track. Other problems were related to the hanging cabin design. The bogie would get stuck in or jump at the junctions due to the track gaps. Another design hurdle was the use of batteries to run the pods. It was possible to go through 12-16 batteries every few hours to keep the model running. These were mechanical problems, but there were also aesthetic problems. This was because there was a disconnect between the Full Scale and the 1/12th scale model. The Full Scale track section used a vertical guide rail and a c-channel section to support the bogie. This was drastically different from the enclosed square tube design of the 1/12th model. The feedback from onlookers was that they were not able to merge the ideas together.

This year's 1/12th scale model team focused on these important aspects. The main task of the 1/12th scale team was to redesign the track and bogie while considering a few stipulations. These stipulations were that the 1/12th model should better represent the Full Scale design with a vertical rail guideway instead of the enclosed one. This design was also seen as a solution to the reliability concerns since the bogie would not be getting stuck in the track gaps. The new track design also allowed the team to simplify the assembly of the track. The track is composed of simple aluminum bar that is made into modular sections of guideway. These modular sections will look similar to the Full Scale model and ultimately close the gap between Full Scale and 1/12th. The final piece of the improvements has to do with supplying power to the bogies. Batteries are being replaced by wayside pickup using multiple power supplies powering electrical rails around the track. The bogie will use this available power without the need to store it on board.

The new design emphasizes ease of use and reliability. With the integration of what the Computer Engineering team has done with the master controller, the results should be very fruitful. The beginning of next semester will focus on finalizing any outstanding design considerations. Following that will be the ordering of material and the fabrication of the parts. Then will be the building of the model and testing the system level controls. Thereafter, there will be testing and optimizing of all components to create the best outcome possible.

Special Thanks to the following:

Dr. Buford Furman

Ron Swenson

Sam Ellis

Bryan Burlingame

Anuradha Munshi

Lizie Michel

Max Goldberg

Paul Albulet

These people have given advice and lent their skill for the benefit of this team and toward the advancement of our goals. We thank them.

Table of Contents

➢ Description of the Subteam and Objectives

The 1/12th scale team is focused on creating a working model of the full-scale Automated Transit Network (ATN). Due to the large scope of the project and budget and time constraints, it is somewhat impossible to design and build a full-scale prototype of the system. In order to prove that the concept is viable, it is important to have the 1/12th scale model. Fancy animations are fun to look at but it is difficult to convince an audience without a physical model. Thus, the team has adopted the slogan "Seeing is Believing" because this model will be presented at exhibitions to demonstrate how the actual system would work. Through successful completion of this model, the project will gain more momentum and support from potential sponsors as well as customers.

As with any project, there were several issues that needed to be resolved. Consistency was a real concern in the previous years' model, and thus that was one of the key focuses throughout the design process. For the Fall semester, the team's objectives were to quickly iterate three designs, construct a prototype of the 1/12th scale model, complete a redesign of the 1/12th scale guideway, implement wayside pickup, match the full scale design within reason, and to interface the code provided by the Computer Engineers.

In prior years, there had been issues in the switch sections of the track causing the bogie to jerk at the interface. While conducting tests, batteries would also run out very often, creating a need for wayside pickup. Many such issues needed immediate attention so the team quickly got to work. Initially, the team split up into three sub-teams in order to work efficiently and generate more ideas. Implementing a divergent-convergent work process, the sub-teams were able to come up with several concepts of their own and then narrow it down to one design. Each subteam was in charge of quickly prototyping its idea for the track and bogie for further analysis. The entire 1/12th scale team then met up to discuss the strengths and weaknesses of each of these ideas and to come up with a final design that would make use of the strengths from all of the iterations. Certain standards had to be met and the team attempted to match the full-scale design as much as possible, but also made adjustments as necessary for the 1/12th scale model. The next section will talk about the engineering specifications that the project had to meet.

➢ Design Requirements

- Support 24.5 N

It has been decided to build a bogie and track that will support up to 24.5 Newtons. This works out to about 5.5 lbs. This target weight of 24.5 N will include the bogie, cabin, and all necessary hardware or electronics for operation. With this weight, track deflections should not exceed a vertical deflection of 3mm and a torsional deflection of 2mm. FEA will be used in the upcoming semester to determine these numbers are met.

- Move up to .5 m/s

One half meter per second was chosen as the max speed for the bogie. This was decided because it will accurately represent a scaled down full scale operating speed of 35 mph. There are certain standards that need to be pertained to. These standards come from the American Society of Civil Engineers (ASCE). Within the ASCE are a system of codes and regulations that the Automated People Mover Standards Committee (APM) use to determine safe operating speeds for automated people movers. Additional standards limit the rate of acceleration that can be used. For linear acceleration the standard states that it can not exceed 0.35g and for lateral acceleration that number cannot exceed 0.25g. Lateral acceleration would come into effect around any curves in the track. The curves forming the track will involve 0.5m and 1m radi sections. In order to meet these requirements PID code will be written which will maintain safe operation. Even if the bogie has the ability to accelerate faster or operate at a higher speed, that will be kept down. This speed will require a relatively fast switching mechanism. It has been decided that the switch will need to occur in 0.5 seconds. If the switch occurs in 0.5 seconds than the bogey will have only traveled 0.25 meters at a speed of 0.5 m/s.

- Build at least 3 bogies

It will be necessary to have at least three bogies operating on the track at one time in order to correctly demonstrate operation of the system. Three bogies will allow the system to demonstrate a variety of different things. If a bogie is being called to a station, the closest one will respond to it. If one bogie is merging onto a track with another one approaching, it will either speed up or slow down depending on what the safest operation would be. Having three bogies will also force the master controller to know where each one is at any specific time. This will keep them from crashing into each other or from practicing unsafe operation. It has been determined that a minimum operating distance of at least 20cm must be maintained between each bogey.

- Design track with at least 3 stations

One reason for having three stations is the requirement to have three bogies. If all stations are requesting a bogie, then there needs to be a place to dock each one. Another reason for three stations is because it helps to prove the concept and ideas of the 1/12 scale model. The three stations will be spread throughout the track layout. Included in the track layout is a shortcut route that allows a bogie to skip a station if it needs to hurry to the next one in line. It is also important to have two stations in a row along the same path of travel. One of the main benefits of the Spartan Superway is that a bogie does not have to stop at each station. There are two stops during a groups travel time. One stop is for them to load into the cabin and the second stop is for them to unload. Having two stations along the same loop of track demonstrates this ability for direct transportation without the need to make any additional stops.

-

Computer controlled

It is absolutely necessary for this 1/12 scale model to be computer controlled. The main controller will be a Raspberry Pi and all of the bogie controllers will consist of Arduino Uno microcontrollers. The Raspberry Pi will control all operation relating to the system. User input will be limited calling a bogie to a specific station. The less involvement the user has with the system the better. The more control the user has the less automated the system becomes which takes away from the idea of the track being an automated transit network.

- Emulates the full scale model

It is necessary to have the 1/12 scale model emulate the full scale model as closely as possible. Seeing is believing, and it will be hard to convince people about the importance of an automated transit network if the model does not even come close to mimicking the full scale system. The switching mechanism, drive, and track will seek to emulate the full scale model to the highest degree.

- Implement wayside pickup 12 volts that supply up to 12 amps

In order to improve upon the progress created by last years model, the team will work to implement wayside power pickup. There will be no need to constantly swap out battery packs. The absence of battery packs will also decrease the weight of the bogie overall, which can help improve the performance of the bogie. In order to power all three bogies wayside pickup will need to supply at least 12V and close to 12 amps. This will power all of the motors, Arduinos, actuators, and sensors.

➢ Literature Review

- Spring 2014 Report

The Current 1/12th Scale Model was built by the 2012 - 2013 Superway Team. It was improved by last years team and the controls were demonstrated by the Computer Engineering team this semester. The original intention of the 1/12th scale model was to closely represent the full scale model and prove system-wide concepts. This proved to be a larger task that was possible in a single year. The model ended up being less representative of the full scale model, but managed to demonstrate a basic ATN system. This was important work for the Superway project because it clearly demonstrated the fundamental concept of an automated transportation network. Last year's work to improve reliability to implement the current control system was a large step in the right direction.

The mechanical part of the model is comprised of a square guideway with a slit cut out of the bottom. This square guideway easily supports the bogie at the cost of some efficiency. Square tubing is usually very good at resisting torque, but once a slot is made it it is no better than a wide, flat beam. The square guideway also caused the switching to be unreliable due to the wheels having to pass over cut outs in the guideway. This would sometimes cause the bogie to

stall and always make for a bumpy transition between sections. The guideway is supported by metal poles set into concrete in plastic buckets. This is very stable, but does not allow for much adjustment to compensate for uneven ground.

The controls part of the model is currently set up for a central controller to wirelessly communicate with the controllers on each bogie. The bogies each have two Arduinos, one for communication with the master controller and one for processing sensor input and actuation. The master controller is a Raspberry Pi and the control system is programmed using python. The control program determines the position of each bogie and then takes user input to determine where the bogies need to go. Then it chooses the bogie that will execute the user input most efficiently and calculates a path for that bogie. This path takes into account where the other bogies are and where they will be. The bogies use hall sensors to detect where they are on the track and use an Xbee to wirelessly communicate that information to the master controller. The bogies also have ultrasonic proximity sensors to prevent collisions with other bogies.

This year the team intends to use the fundamental lessons learned by the previous teams to produce a new 1/12th scale model. If applied, these lessons will make the model easier to transport, assemble and adjust. The model will be more mechanically reliable while still faithful to under-the-rail ATN concepts. The controls will remain essentially the same as they are now, but with some changes toward a more robust system. The previous teams' efforts were valuable and will go a long way to make this year's model successful and impressive.

-

Patent by Bengt Gustaffson

Mr. Gustaffson's patent was created several years ago and in that time the particular configuration of the bogie, pictured in Figure 1, has changed. However the fundamental components are still present.

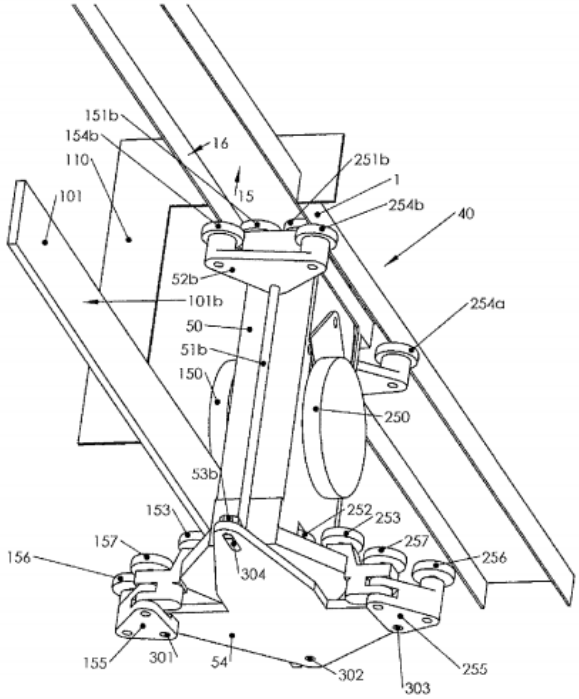


Figure 1: Gustaffson's Bogie Design. Note the upper and lower steering mechanisms, two upper rails and one lower rail.

In his patent Mr. Gustaffson describes a device that is supported by a system of rails that are in turn supported by some grounded structure. Large wheels that roll along some rail or rails would support the bogie vertically. The bogie would be laterally constrained by several smaller wheels in a rectangular section above the previously described rails. The bogie would steer using small horizontal wheels that could catch onto another set of switching rails. This switching portion of the guideway requires many special considerations like timing and supplemental support while the bogie shifts configuration.

- Previous Specifications

The specifications for the previous model were not considered for the design of the year's 1/12th scale model. The previous year's model was inaccessible at the time the current specifications were decided. This forced the new designers to be creative and decisive and was a positive outcome.

-

Previous Software Programming

The Intersolar Arduino code was created by the previous year's team and sets up Serial Peripheral Interface communication protocol between the Arduino and the SJSUONE board. The code allowed the bogie to interface with two hall sensors, a motor, and two solenoids. The previous team also left the foundation for an ultrasonic sensor interface and navigational controls commented out. The software also made use of a manually built PID controller to regulate the speed of the bogie. The previous team has left sufficient debugging tools to adapt the code for future use, but changes in hardware may render large sections of the code obsolete or inapplicable.

➢ Design Concepts

- Convergent divergent design philosophy

The 1/12th scale model team used a convergent divergent design philosophy in order to maximize iterations in a short period of time. The philosophy begins in the divergent stage; the team was split into three teams of 3 to 4 members: Honey I Shrunk the Superway (HISTS), team Red, and team Not a Number (NaN). In the divergent phase, individual members were tasked with creating as many designs as possible. These ideas were then filtered by the first convergence; teams deliberated upon which idea was optimal. Each team was then tasked with building a physical manifestation of their best ideas. The results are shown below.

HISTS

HISTS's design maximized simplicity of the guide rails. The two guide rails minimize cost and time to build track. The bogie switch created two mutually exclusive states; this prevents a floating state in which the bogie is neither turning nor going straight.

Red

Team Red's design uses a single travel rail, with a secondary rail at the switch. The switching is initiated by the lower four bar mechanism pictured in green below. This design required a notch in order to give the four bar mechanism room to actuate.

NaN

Team Nan's design modified the previous team's scale model. The boxed cross section remains, with a notch cut in the top in switching regions. The method of actuation was changed to a lever arm that would lift the bogie when it reached the notch. This removes the catching that occurred in previous years.

The final convergence dissolved the teams back into a full team, whose task was to decide which model had the most potential to meet the team's overall objectives.

By splitting into smaller teams, all members were able to give input, without the administrative and moderation that is necessary in large group discussions.

- Engineering Design Process

The engineering design method is the governing path that Engineers use in order to complete a design, this project is no different. The problem that the 1/12th team has sought to solve is the lack of a reliable and representative test course. The convergent divergent philosophy was a means of coming up with concepts; building proof-of-concept models was a phase of research. The team is in the process of completing further research and getting ready to create prototypes.

➢ Analysis and Concept Selection

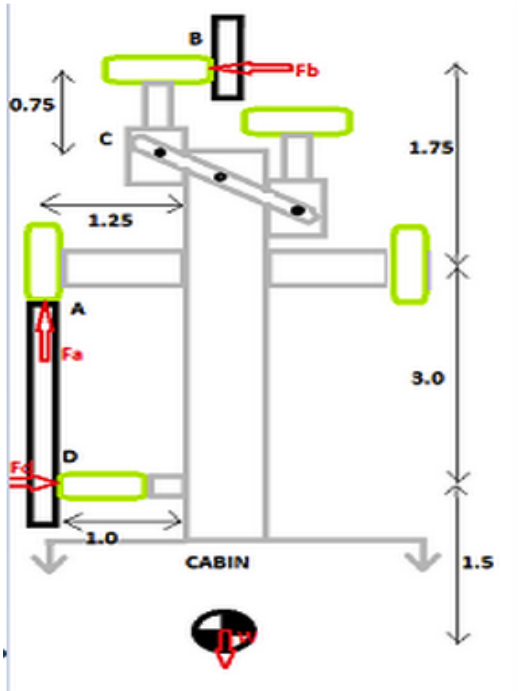
- Moment and force equation

As stated, the goal of the 1/12th scale team is to demonstrate the Superway concept with the 1/12th scale model. Prior to designing the scaled bogie and track, moment and force analysis were studied and calculated. The bogie has to maintain equilibrium any time while running on top of the track.

Thus, all the initial concepts of prototypes from sub-teams were designed around balancing the moment and force acting on the bogie.

HISTS

HISTS's design uses the upper track for force balance and the side track to counter the moment produced by the weight of bogie and cabin.



Assume W=3lbs

$$\sum F_y = 0 = F_A - W$$
$$F_A = 3\ lbs$$

$$\sum F_x = 0 = F_D - F_B$$
$$F_D = F_B$$

$$\sum M_A = 0 = F_B * 1.75 - W * 1.25 + F_D * 0.25$$
$$\sum M_A = 0 = F_B * 1.75 - W * 1.25 + F_B * 0.25$$
$$2 * F_B = W * 1.25$$
$$F_B = 1.875\ lbs$$

$$M_{B\ due\ to\ Fa} = F_A * 1.25$$
$$M_{B\ due\ to\ Fa} = 3.75\ lb * in$$

$$M_{C\ due\ to\ Fb} = F_B * 0.75$$
$$M_{C\ due\ to\ Fb} = 3.75\ lb * in$$

Figure 2: Force and Moment Analysis on HISTS team bogie design

Red

Red's team design have the moment and force balanced around the bottom bearings. The outer bearing will counter the force and the track will counter the moment produced by the weight of bogie and cabin.

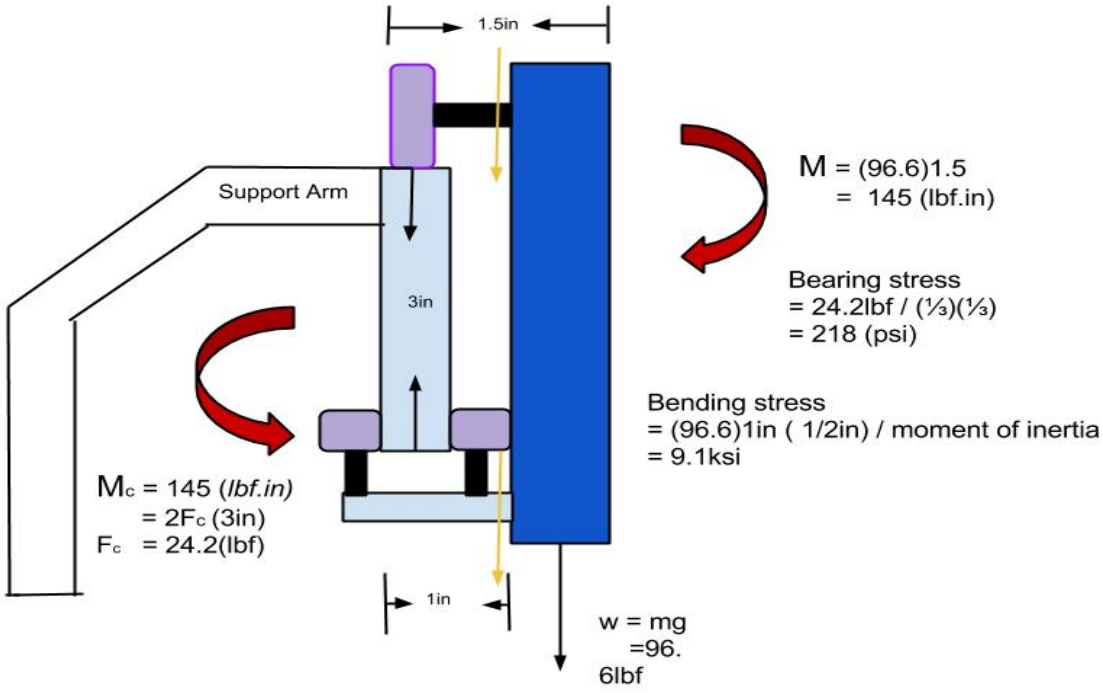


Figure 3: force and moment analysis on Red team bogie design

- Pugh chart

  With each sub-team coming out with one design, the 1/12th team decided to use a pugh chart to evaluate the strength and flaw of each prototype design. The evaluation was based on the ease of construction, overall cost, safety issue and stability of bogie. The result showed that design from HISTS team was relatively desirable out of all three designs. Thus, the 1/12th team used the HISTS team's design concept as the base for the final design.

➢ Results and Discussion

- A bogie design was selected that absorbed the HISTS, Team Red, and Team NaN

While designing the bogie, there are two requirements that the 1/12th team will specifically focus on. First, after entering the switching section, the bogie has to be capable of making a smooth transition from one track to the other. Second, to address safety issues, the bogie has to

have good stability when running on the guide way. To meet these two specifications, the 1/12th scale team combined the switching mechanism from team HISTS and Red to ensure smooth switching and also enhance the stability of the bogie.

In the previous year's design, the 1/12th team used battery packs to provide the required power for system operation. It was inefficient and required constantly replacing the batteries to ensure the system would work. To correct this problem, the 1/12th team will implement wayside power pickup. The concept is taken from Team NaN's design. The idea is to store and transmit the solar power into electrical power. Through powering of the track, the bogie can pick up power from an external source.

- 1/12th team splits into the following teams
    - Hardware
        - Motor Sizing

The motor, essentially the key element of the system, has to meet certain requirements in order to drive the bogie and cabin at required speed. Thus, calculations for motor sizing were performed.

1. The goal is to have the bogie running at 0.5m/s (19.685in/s), and to be accelerated within 0.5 second
2. From bogie design, the 1/12th team decided to make assumption for a size of NEMA 11 motor with diameter of 0.022 m, 110 g
3. To calculate the RPM of the motor required, the motor has to reach: 0.5m/ (0.022 m x pi) = 7.24 rotations per second to get the speed of 0.5m/s
4. Convert the rotation per second to RPM: 7.24 x 60 = 435 RPM. Thus, a motor within 440~450 RPM seems to be a reasonable choice, depending on the actual size of the motor.
5. While running on the guide way, friction torque will exert resistance to the bogie. Thus, friction torque has to be part of the motor sizing concerns.
6. System inertia is needed to calculate friction torque and the total torque required.
   The motor inertia is 0.05 oz-in^2 (32mm in length) = 9.1e-7 kg-m^2
   Friction torque:
   ω (rotational speed) = 450 RPM = 47.12 rad/sec,
   a (acceleration) = 47.12/(0.5sec) = 94.24 rad/sec^2
   T_f(friction torque) = Ia = (94.24)(9.1e-7) = 0.0000857 N-m
7. Now apply the formula: $T = (J*N*\pi)/(ta) + T_f$
   $T = (9.1e-7)(450)(\pi) / (0.5) + 0.0000857 = 0.00267$ N-m

For the bogie to reach 0.5m/s with 0.5 sec, the motor requires a minimum of 0.00267N-m of torque at 450 RPM when powered.

- Selecting a Solenoid

In the current bogie design, (see 14.2.3) a switch mechanism is designed to complete the switching when the bogie changes direction. To move the switching bar to the desired position, a linear actuator would be used to actuate the switching mechanism.

One of the concerns of using the linear actuator is that the actuator would retract back to it's initial position if the control system lost power. And it will result in presenting a failed concept during any demonstration.

Thus, instead of regular linear actuator, a latching solenoid actuator will be used for actuation. The feature of latching solenoid is that the actuator will maintain at extended position once it is actuated. The built in magnet will hold the plunger at energized state. In addition, the latching solenoid operates on low duty cycle with power requirement between 3 to 6 Volts. Therefore, using latching solenoid actuator leads to steady actuation and requires low power for operation.

- Implementation of wayside pickup

As stated in section 13.1, one of the major differences that distinguish the 1/12th scale model from last year's design is the implementation of wayside pickup. The implementation of wayside pickup is important because it helps to mimic the idea of the full scale. It will also remove some of the issues that were evident with the previous years 1/12th scale model. The previous year had battery packs attached to each bogie. During their display time at the Maker Faire they constantly had to replace the battery packs. The lack of a practical use time from the battery packs created the need to find alternative solutions. This led the team to deciding upon the use of wayside pickup.

The physical implementation of wayside pickup is also a challenge and needs to be further explored. Initial thoughts were to use already proven methods of wayside pickup. Some of these ideas included the use of model train track. Model trains are able to are able to pull power from the train tracks through their wheels. The idea would be to line the entire inside of the track with the model train track. The benefit of using train track is that it would come fully prefabricated, including connection joints. This would greatly ease the implementation of wayside pickup if it already comes fully fabricated. The only need for custom fabrication would be the device used to pick up power from the rails and transmit it to the bogie motor. It would be possible to use the wheels that the model trains have, however, that would require an extreme amount of precision especially when switching tracks. If the wheels slipped off of the track, even a little bit, the motor would lose power. Because of this, a custom pickup tool would need to fabricated to help cover for any potential errors in precision. This pickup tools could be a metal wheel or bearing, a metal ski, or even a metal brush would work. These power pickup tools would be built wider than the model train track rail. Concerns for using the model railroad track involved the precision required as well as the challenges that would arise from trying to bend the track. The way the beams are formed would make them very challenging to bend around corners. Producers do office flexible track, however, they will still not meet the required bend radius on the 1/12th scale track without additional bending.

Another idea was to use conductive tape. This would allow the team to get around the challenges that would occur from trying to bend the model railroad track. The tape would easily apply to the side of the track and could be bent to whatever dimensions are required. The conductive tape would need to be applied on top of electrical tape. This would keep the powered conductive tape from grounding itself out on the track. The same power pickup mechanisms from the previous paragraph could be used to pull power from the electrical tape. Concerns with using tape involve reliability. If three bogies are constantly circling the track it is possible the tape would get worn out or start to peel. Because this track will be functioning in front of audiences, it needs to be aesthetically pleasing to the eye. The use of tape might take away from that experience.

Fabricating the entire wayside pickup track is also an option. Aluminum or any other metal could be bent and applied to the side of the track. It would need to be applied over electrical tape, similar to the previous idea so that it would ground itself out on the track. Problems with using an aluminum power rail would involve cost and practicality. It will take a large amount of time and energy to fabricate aluminum power rails to run along the entire inside of the track. However, it would be the most visually appealing.

One last idea that was brought up during the 1/12th scale teams final presentation was to use the drive rail as the power rail. This would make use of already existing material and would require no additional parts or materials. With this method it would then have to be decided how to best isolate the drive rail from the remainder of the track. This would be the ideal method and will be further researched in the upcoming months.

Power sources will be used to power the power rail. This power will come from car batteries. It would be ideal to use power from a wall outlet; however, at the Maker Faire the team might not have access to this and needs to plan for alternative solutions.

Initial tests have been performed to test the practicality of wayside pickup. This was done at the most basic level practical. A 12V 2A power supply was used power two aluminum rails. The positive wire was wrapped around one rail and the ground wire around the other. A 12V DC motor was then used to prove whether or not power was successfully transmitted through the rails. Initially the wires coming off of the motor were placed directly on the rails. Once both wires were on the motor ran. The wires were then slid along the length of the rails to test if the motor could maintain power while moving. This also worked. The next step was to attach some kind of pickup mechanism to the motor wires and see if power could be transmitted that way. This was done to mimic a wheel or ski being used on the bogie to pick up power. This was tested the same way as the wires. The use of the metal skis actually worked better just the wires. The extra weight and larger surface area helped maintain a greater contact between the skis and the powered rails. Electrical tape was also used to see if it could isolate the rails from each other.

This was another successful test. The tape successfully isolated one rail from the other allowing the motor to run. A photo of this experiment can be seen below in Figure 4.

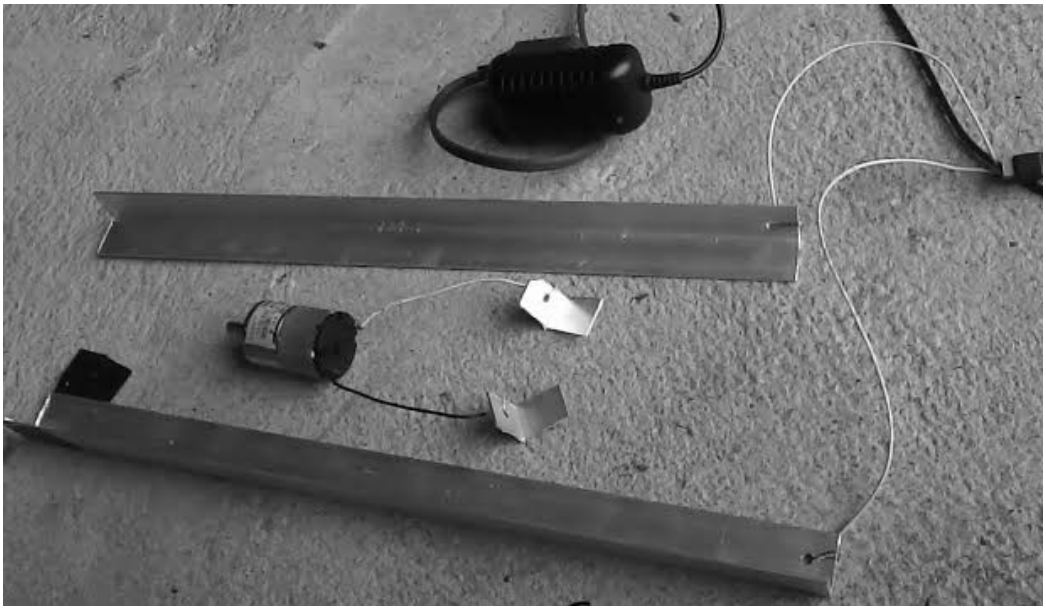


Figure 4: Initial test of wayside power pickup using aluminum rails.

- **Controls**
  - Introduction

The 1/12th scale team is primarily designed to demonstrate the controls functionality of an ATN system. The Computer Engineering department allocated a 3-person team to help design the control system for the 1/12th scale track. Both the mechanical and computer engineers worked together to establish the communication protocol.

  - Computer Engineering Team

The computer engineering team established the main control system for the 1/12th scale track. This included a B+ Raspberry Pi, 6 arduinos, 6 OH4EE hall sensors, and 3 proximity sensors. There are 3 important components to the 1/12th control design: a main hub, a communication Arduino, and a controls Arduino. A graphical user interface, which is part of the main hub, monitors the location, the status of each pod on the track, and provides the operator to designate routing instructions for each pod. The Pi serves as the main hub, which both communicates with each Arduino and gives routing instructions. Two Arduinos are designated to each pod, which serves as a communication board and a controls board. The communication boards are equipped with a Series 2 Xbee shield. Xbee is a wireless communication protocol that can both send and receive data within a 300 ft range. Refer to **Appendix B** for a more detailed explanation of the sender/receiver communication protocol between the Pi and a pod's communication board via Xbee wireless. The control boards primarily serves to control motor speed, hall sensors, and proximity sensors. The controls board will be coded and designed by the Superway Mechanical Engineering team.

- Navigation System

A navigation system was created to help supplement the Computer Engineers' work. Refer to the figure below:

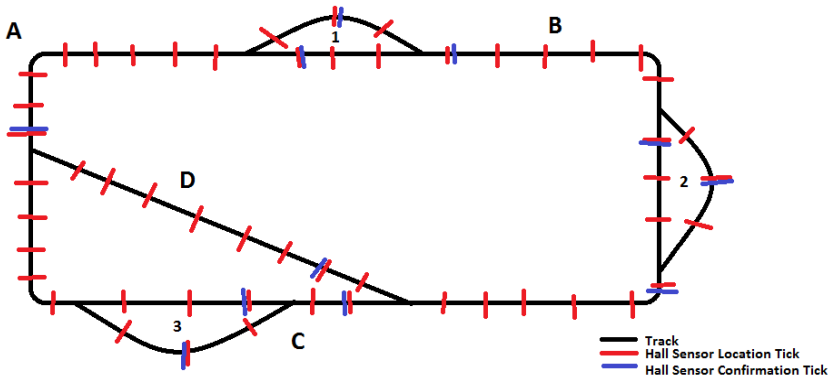


Figure 5: Navigation protocol for the 1/12th scale track will use hall sensors.

In the figure above, there are 4 regions and 3 stations: A, B, C, and D. and 1,2 and 3 respectively. Magnets will be place around the track, which are shown in the figure as red and blue lines. These magnets will be used with the two hall sensors equipped on each pod. A red line is the *location* tick, and the blue line is the *confirmation* tick. Or in other words, the red lines are magnets which designate location, whereas the blue line are magnets that will confirm a pod is following the correct path. The red lines (magnets) will be equally placed around the track so that both the pod and main hub will know where the bogie is at all times. The main purpose for the second magnet is to confirm a pod has followed its instructed path. When compared with the second magnet count, the pod will always know if it is on course or took a wrong turn.

- Navigation Example

To tidy up the terminology use, assume the red lines are red magnets and the blue lines are blue magnets. Know that when a pod reads a red or blue magnet, it is actually the hall sensor that reads the magnet. One hall sensor for the red magnets; and one hall sensor for the blue magnets.

The main hub instructs a pod in station 3, which at rest on a blue and red magnet, to station 2. Starting from rest, the pod will count 6 red magnets before entering region A. Region A is designated by *both* a blue and red magnet. When the pod passes over these two magnets, it will confirm that a blue magnet was read on the 7th red magnet count. Thus confirming that the pod is on the correct path. The pod is now aware that it is in Region A. The counter resets and the pod continues course.

The pod moves along and counts 8 more red magnets until it counts another blue magnet. The pod confirms that a blue magnet was read on the 8th red magnet count. Thus confirming that the pod is following the correct path. The counter resets and the pod moves along.

Now consider that the pod read a blue magnet on the 9th tick. The pod would then be in station 1, which is not the correct station. The pod would stop at this station and wait for instructions from the main hub.

Back to the example, with the pod on the correct path and the counter reset, the pod moves along towards its destination, station 2. The pod counts 3 more red magnets until it reads a blue magnet. This confirms that the pod is now in Region B. The counter resets and the pod continues course.

The pod moves along and counts 7 more red magnets until is counts another blue magnet. The pod confirms that a blue magnet was read on the 7th red magnet count. The pod is now at its destination, station 2. Confirming that a blue magnet was read on the 7th red magnet, also confirms that the pod followed the correct path. The counter resets and the pod waits for instructions from the main hub.

If the pod read a blue magnet on the 6th red magnet count, then the pod would know that it did not make the turn necessary and lost course. It would stop on the track and wait for further instruction from the main hub.

- Discussion about Navigation System

Since each region, except for region D, consists of a station, a 90-degree turn, and two straight sections. It would be wise have equal number of location magnets and confirmation magnets in each region. Furthermore, the confirmation magnets should be placed in the same place for each region, except for region D. This would establish consistency among the regions and make code easier to write.

For example, all regions will have 20 magnets location magnets. Regions A, B, C (which are all the same and consist of a station, 90 degree turn, and a 2 straight sections) will have 3 confirmation magnets and Region D (which does not have a station) will also have 2 confirmation magnets (one being a dummy magnet).

If the first of 3 confirmation magnets in a region is used to confirm that the pod has entered a region, then the number of location magnet to the next confirmation magnet will be the same for each region. If a 2nd confirmation magnet is *always* placed with the location magnet *immediately* after a junction (the first location magnet passed a turnoff), then confirmation that a pod has followed did not turn will *always* align with the number of location magnets it has previously read. If the 3rd confirmation magnet is placed in station, and each station has equal number of magnets, then the confirmation will always align with the number of location magnets it has previously read. For the special case of Region D, the first magnet will confirm pod has entered region D, the second will confirm that a pod has taken the shortcut, and the third will be used as

a dummy for the purpose of keeping code constant. A special SWITCH case can be created for the dummy confirmation magnet in region D.

- Note about Navigation System

This navigation system in section 15.3.2.4 is currently under review and may not be implemented during fabrication because a robust location system may not be needed to prove the concept of an ATN. A couple of hard coded pre-designated paths that demonstrates the concept of an ATN may be sufficient. Furthermore, the Arduino processing power may need to be reserved for higher priority functions such as motor control.

- Interpret last year code and modify it to be usable this year
    - PID Library for Arduino

It is important to not waste any progress made by previous years work. Last year, the Superway team created a PID library for bogie speed control rather than use the open source Arduino PID library. The primary reason for this is because it is believed that the Arduino PID library is inherently flawed and requires lots of processing power to run the functions. It will be ideal to use the library created by the previous semester's team.

Further evidence shows that the Arduino PID library is flawed because a loop of track and bogie was prototyped to test the Arduino PID library. A track was built to test a single Arduino running a PI control for bogie speed. Figures **6** and **7** show the CAD rendering as well as the fabricated track.

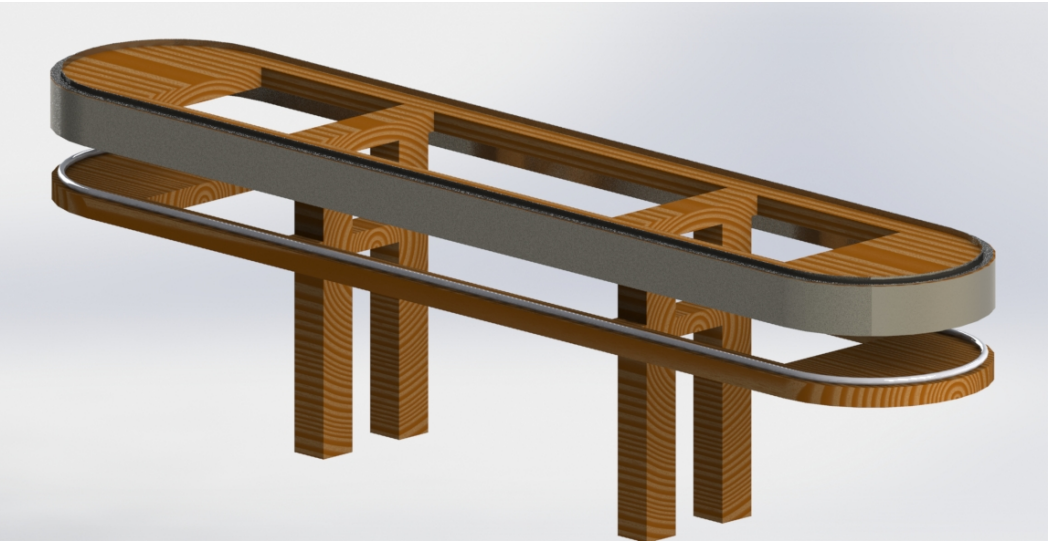


Figure 6: CAD rendering of a 1/12th scale track loop. This was created to determine a PID controller for a speed control of a bogie.

Figure 7: A 1/12th scale track loop was created to test PID speed control

The code for the PID library can be seen in **Appendix D**. The limited amount of processing an Arduino is capable of restricts the team to use this Library. Many teams in the ME 190 course attempted to use the Arduino PID library for control systems. General consensus was that the PID library is inherently flawed and should not be used.

In addition, wood was found to not be a good construction material for the track because of its permeability to moisture. Over time, the wood warped which, as a result, changed the tracks intended shape design. The warped shape track affected bogie's performance while moving along the track by introducing more resistance. This increase in resistance sometimes caused the bogie to get stuck or slow down dramatically.

- Hall Sensor functions

The previous years team utilized hall sensors to actuate mechanisms on the bogie. These functions will most likely not be used because a more robust design will be implemented. Two hall sensors may be used on each pod to identify a pod's location on the track.

- **Bogie**
  - Current Design

As discussed earlier, there were three initial bogie and guideway combination prototypes built prior to the development of the final design. Each prototype's relative merits and weaknesses were analyzed, and the merits of each design would be implemented into one final design.

To begin designing a bogie that follows a guideway, the guideway cross section and width dimensions must first be defined. Figure 8 below shows a figure of the guideway dimensions.

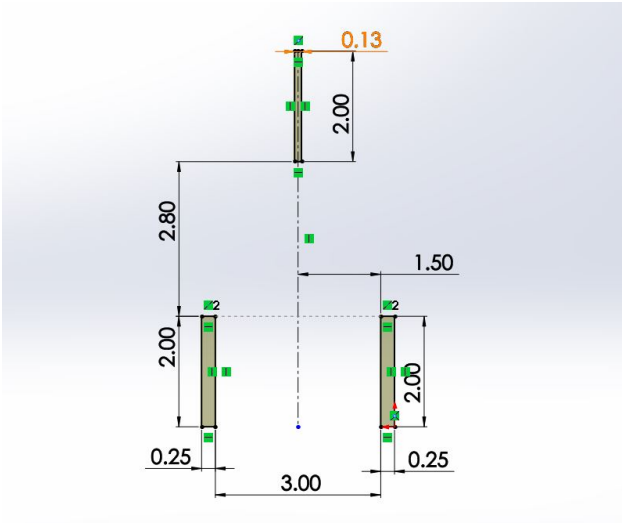


Figure 8: Computer-aided model of the defined 1/12 scale model rail cross section.

There are three rails with the top rail being in different width than the lower two rails. The top rail is used to guide a switching mechanism utilizing a set of bearings linearly actuated to each side of the rail for direction selection. These bearings are linked in such a way that only one side (left or right) will be selected at any given time. The pair of rails at the bottom of the cross section are used for propulsion and switching. On the top surface will sit driven wheels that propel the bogie. At the bottom of those rails will be a linearly actuated switching mechanism. Similar to the upper switching mechanism, this lower switching mechanism will utilize bearings to select either the left or the right rail for direction selection. These concepts were shown in the previous prototypes, and this final design of the bogie will effectively integrate these concepts into a working model. Figure 9 below shows the current final design of the bogie.

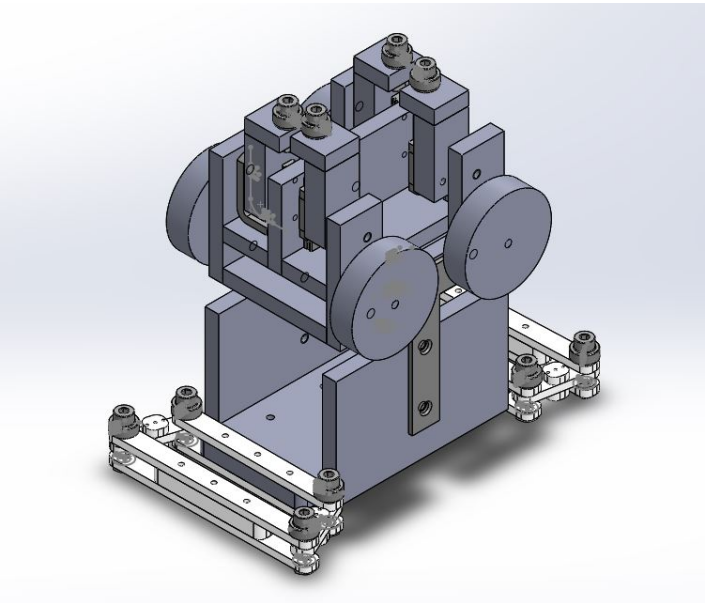


Figure 9: Computer-aided model of the bogie final design. The model does not currently show actuation hardware, as these have not yet been decided upon.

This final design has been made to integrate the previous concepts in a manner where each concept has been separated into individual portions. This allows simultaneous development of individual concepts. Also, this design has been built for fabrication via laser-cut acrylic chemically bonded together.

As such, these portions may be iterated quickly to improve the overall efficiency of production. Figures 10, 11, and 12 below show the individual portions:

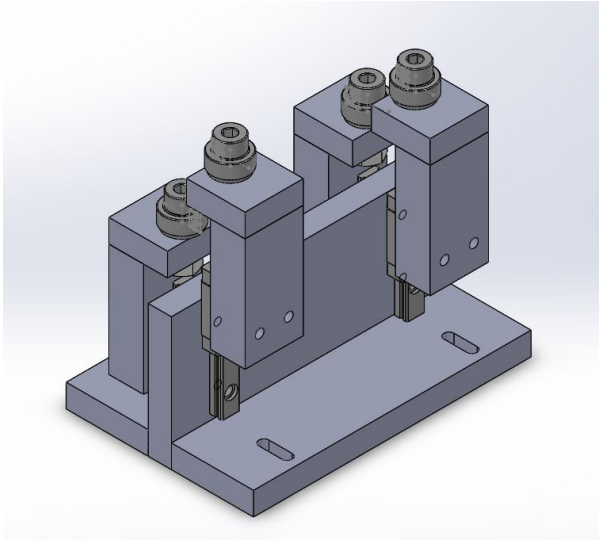


Figure 10: Computer-aided model of the upper switch mechanism. The model currently does not show how each opposite pair of linear sliders will be linked together in an inverse configuration. The base pieces have slots for mating to the propulsion portion via L-brackets.

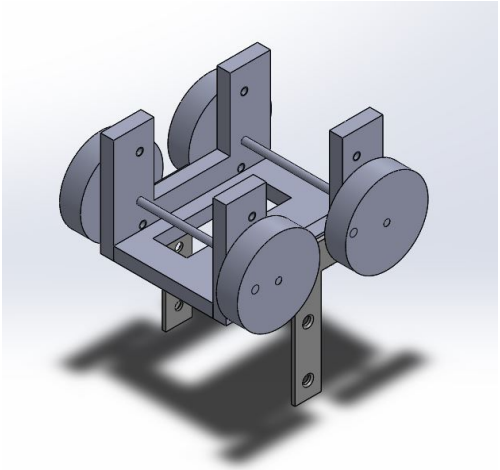


Figure 11: Computer-aided model of the propulsion portion of the bogie. This portion connects the upper switch to the lower switch. Propulsion wheels will be motor driven. The link between the motor and the wheels has not been decided.

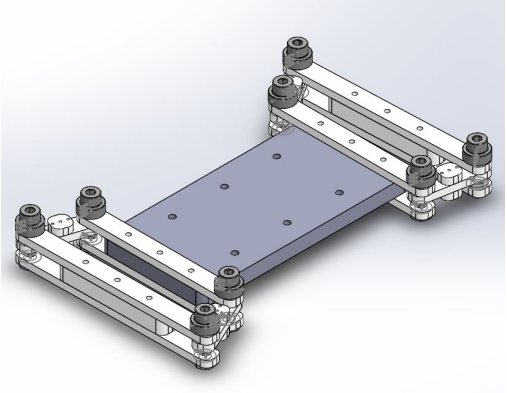


Figure 12: Computer-aided model of the lower switching mechanism of the bogie. This switching mechanism utilizes two four-bar mechanisms to latch onto either the left or the right rail to determine movement direction. Actuation of this switching mechanism will involve linking two of the long couplers at each end of the bogie in an inverse manner.

Each of these portions will be assembled together with brackets and fasteners. These brackets and fasteners were located in such a way where an operator will be able to find them and assemble/disassemble the bogie.

Switching between left and right directions of the guideway involves actuating both upper and lower switching mechanisms before the change in guideway occurs. Figure 10, 11, and 12 illustrate how the current bogie design switches between rails.

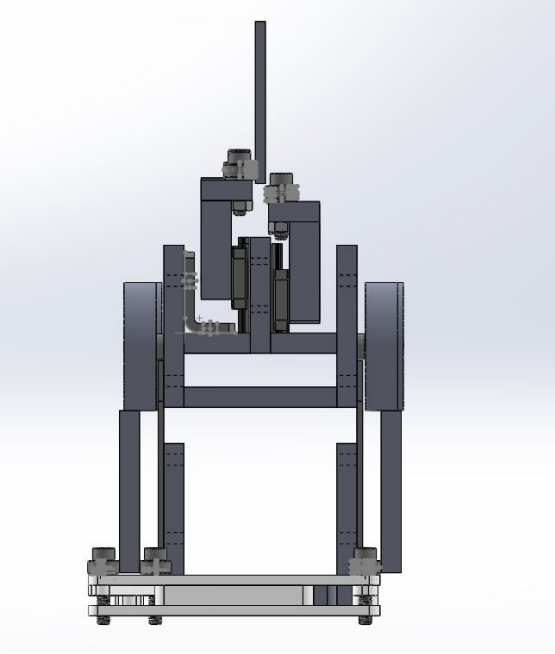


Figure 13: Rear view of the bogie as it traverses along the guideway. The upper switch has selected the left rail by moving its bearings upward. The lower switch selected by latching onto the left rail with its four bar mechanism.

The bogie has been given a command to switch to the right rail. Since there are currently no switching sections, no action beyond traversing forward can be taken.

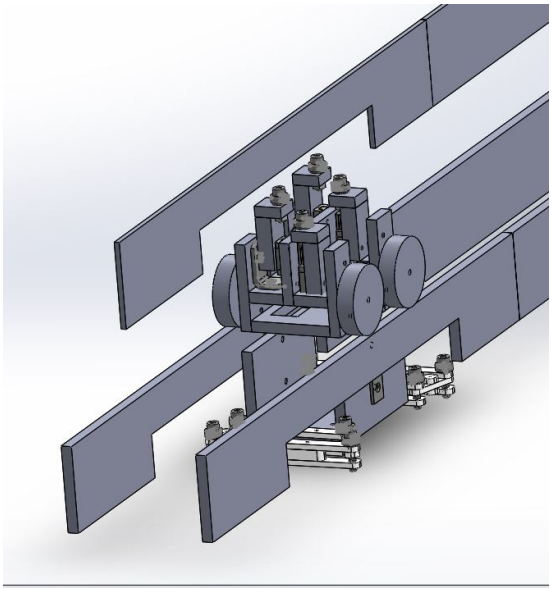


Figure 14: Isometric view of the bogie as it arrives at a switching portion of the track. This is noted by the notches cut into the rail material. The controls of the bogie will allow it to recognize that it has arrived in a switching portion of the track. It is now allowable to switch rails.

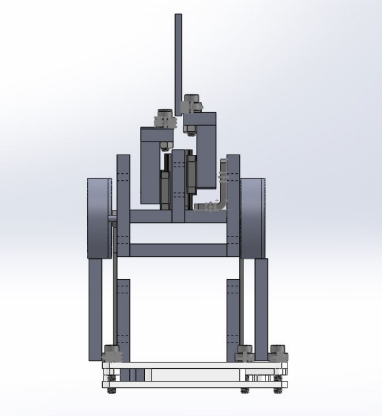


Figure 15: Back view of the bogie with switching mechanisms selecting the right rail of the guideway. Notice the upper and lower switch have moved their bearings away from the left side of the guideway cross sections to the right.

There are a number of concerns with this current bogie design. Inherently, this design allows the upper and lower switch to operate independently. A catastrophic failure will occur if the upper and lower switches do not select the same side. Also, there are currently many components to assemble for this bogie design. The overall wheelbase may also be shortened by selecting smaller wheels, thereby decreasing the turning radius of the bogie. Further development must be

made to reduce the amount of components to shave weight and mitigate unforeseeable circumstances. At the same time, the bogie must be designed to hold all other components such as microcontrollers and sensors for the controls of the bogie. The lower switching mechanism was also deemed to be too complicated and so requires significant design alteration.

- Development Effort

From the various concerns regarding the current bogie design, a development effort was made to significantly alter the lower switching mechanism. Also, the small scale of the bogie design required a proof-of-concept to be made for verification of capability. This effort proved another solution to a linearly actuated switching mechanism that latched onto the lower pair of rails of the guideway with a significant reduction of material. **Figure 16** below illustrates this next-generation lower switching mechanism's two states.

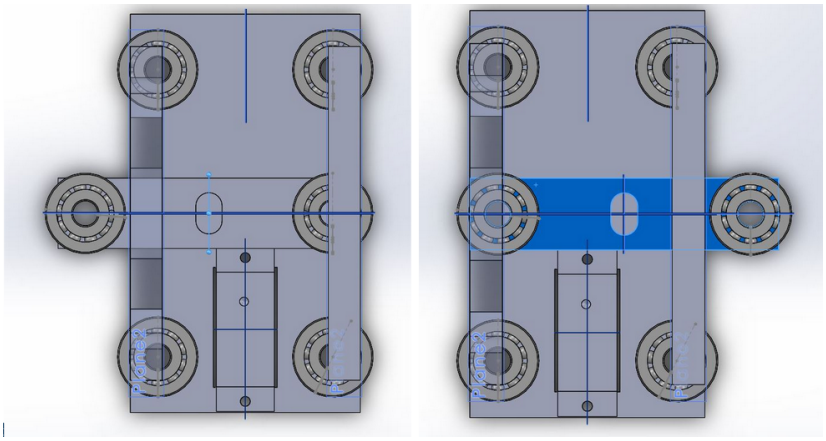


Figure 16: Graphical representation of the new lower switching mechanism's two states. The left image corresponds to left rail selection, and the right corresponds to the right rail selection. Actuation will move the link left and right through these two states.

This new switching mechanism was designed with the dimensions of the defined guideway cross section in mind. Since this mechanism was the only concept requiring proof-of-concept, a prototype was rapidly produced. To further this development effort, actuation was also designed and implemented in the prototype. As such, this development effort was invaluable to the overall design process of the final bogie design. Figure 17 below shows the computer-aided model of the prototype that proved this new lower switching mechanism.

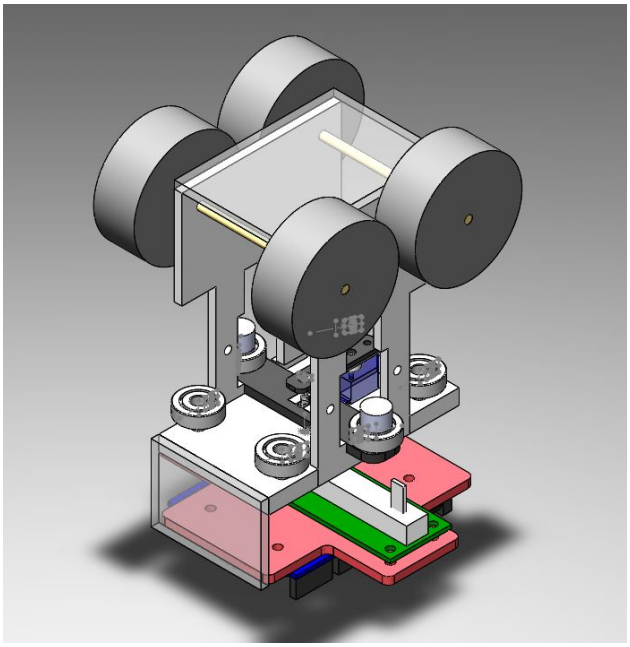


Figure 17: Computer-aided model of the new lower switching mechanism. This prototype was physically small, and therefore was relatively light and easy to handle.

This prototype was also designed to be fabricated with laser-cut acrylic chemically bonded together. Various hardware were purchased and implemented. A separate guideway model was also built to demonstrate effectiveness of this switching mechanism. These attributes of this development effort, therefore, proved more than just this new switching mechanism. Figure 18 below shows the prototype stemming from this development effort.

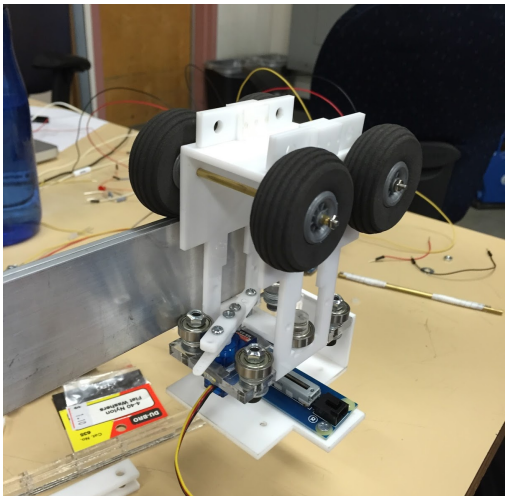


Figure 18: Image of the prototype of this development effort. The prototype bogie is capable of hanging on the single rail by means of constant force applied by a Hitec HS-55 servo motor.

Form factor of this prototype is similar to the form factor of the final bogie. The wheels used in this prototype are too large for the final bogie design. Bearings used in this prototype can be

scaled down, but factors such as cost and availability of fastening methods lead to an inclination away from scaling down further. This prototype also revealed that such a small form factor limits the amount of external components attached to the bogie within the guideway cross section. This could lead to hardware organization issues, and messiness of the overall bogie. **Figure 19** below illustrates this organization concern.

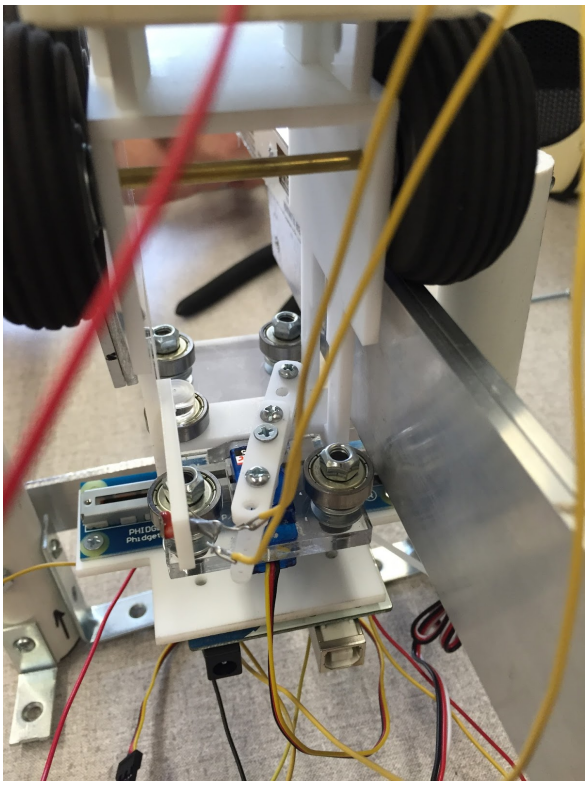


Figure 19: Image of the prototype's state of organization. Wires and components were tightly packed onto the prototype.

This prototype had many wires hanging off of it. Also, the actuation of the linkage would occasionally misalign wires, causing certain portions of the control logic to fail in function. Also, the Arduino microcontroller used to actuate the switching mechanism was mounted upside down, as shown in Figure 19 above. With this physical iteration, many design considerations have been proven and there is now a clearer understanding of the direction the bogie development must move towards.

Concluding Remarks

The current bogie design is not set in any way. From the continuous stream of input and output regarding each iteration and concept, as well as clarity that comes with more conversation regarding the bogie design, this design will continue to evolve. More hardware must also be selected before the design may settle. The goal of fabricating a single bogie design before March will be maintained. As such, producing detailed mechanical drawings of individual acrylic pieces

that make up the bogie is not currently worth the effort. Some mechanical drawings useful for scaling have been included in **Appendix E.**

- **Guideway**

The new 1/12 scale track needed to be created from modular pieces, easily assembled, and adjustable.

The track can be broken down into seven modular aluminum strips shown in **Appendix F**. The pieces consist of a ½ meter straight way and curves at a ½ and 1 meter radii that are bent in 15,30, and 45 degrees. The advantage to having modular pieces is that it allows sections of the track to be replaced if they break or if future work calls for an upgrade. With this design, extra stations can be added by simply putting a 15 degree bend piece parallel to a straight way to make an additional path for the bogie to travel. Furthermore, the variety of the parts also allows the track shape to be changed into other desired configurations. This is important, because it demonstrates that the track can be customized to path of any city streets.

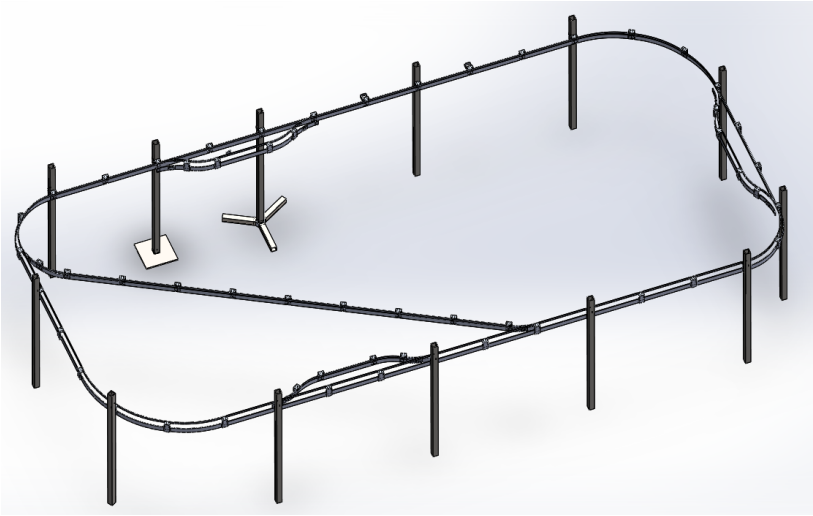


Figure 20: A SolidWorks part assembly of a three-station track and shortcut.

Assembly needed to be quick and easy. With this in mind, the track pieces were designed with both peg and screw connectors. The use of pegs ensures that the pieces line up while the screws serve to lock the pieces together. Each piece of aluminum has a set of two pegs and two screws on each side in order to line up horizontally and vertically. The pieces are short and they require many connectors to be fastened which could be time consuming. But there is a way to reduce the reassembly time. The solution is to only disassemble a fraction of the track connectors to leave sections of 3-4 pieces together. To reassemble, the segments of 3-4 pieces would only need to be reconnected later on.

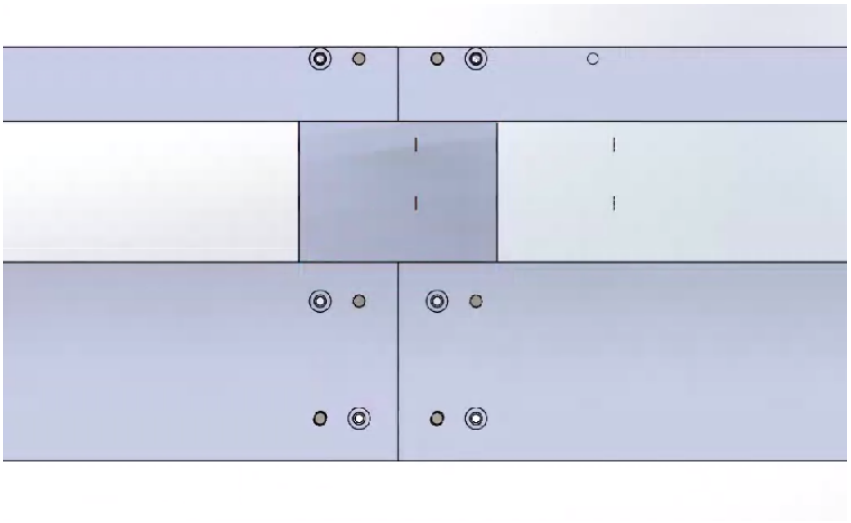


Figure 21: This figure depicts the connection between two aluminum strips.

The adjustability in this track comes from the height adjustment from the support-guideway connectors. The support-guideway connector locks onto the support bar through the use of a nut and bolt. The bolt is put into the nut and tightened to pinch against the bar. To adjust the height, the bolt would be loosened and the support-guideway connector would only need to be manually slid up or down before being refastened.

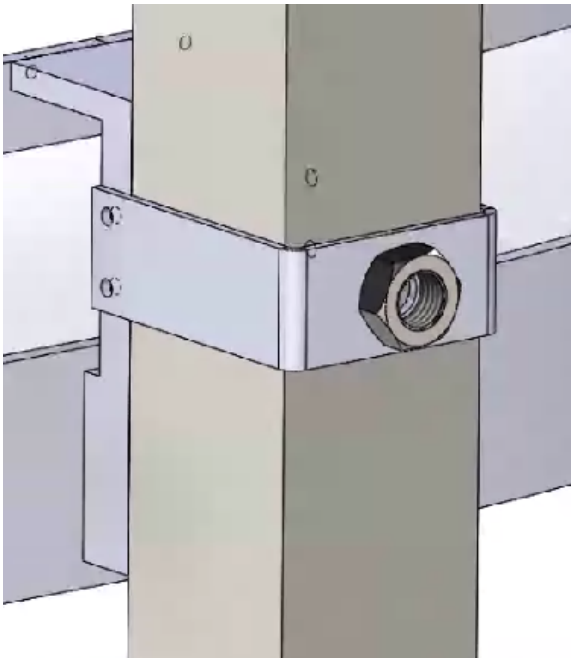


Figure 22: A SolidWorks part assembly support-guideway connector fastened

➢

Major Accomplishments

The 1/12th Scale Model Teams began the semester with a large amount of work ahead; scale track, bogie, and controls, each with a unique set of goals, guidelines, and requirements, had to be designed. To lay the groundwork for the scale of the work ahead, the team broke into three sub-teams to investigate as many possible solutions in the shortest amount of time. The three sub-teams successfully created three rapid prototypes to dial in on a final design. The final design path consists of aspects of all three prototypes to ensure a robust and reliable design.

Bogie Team

The bogie design incorporates two redundant switching mechanisms to ensure the switching section is both reliable and secure while maintaining the smallest possible footprint.

Track Team

The track utilizes a single rail design with modular straight, curse, and switch sections. The duplicated modular design allows for adaptable configurations as well as added ease of manufacturability.

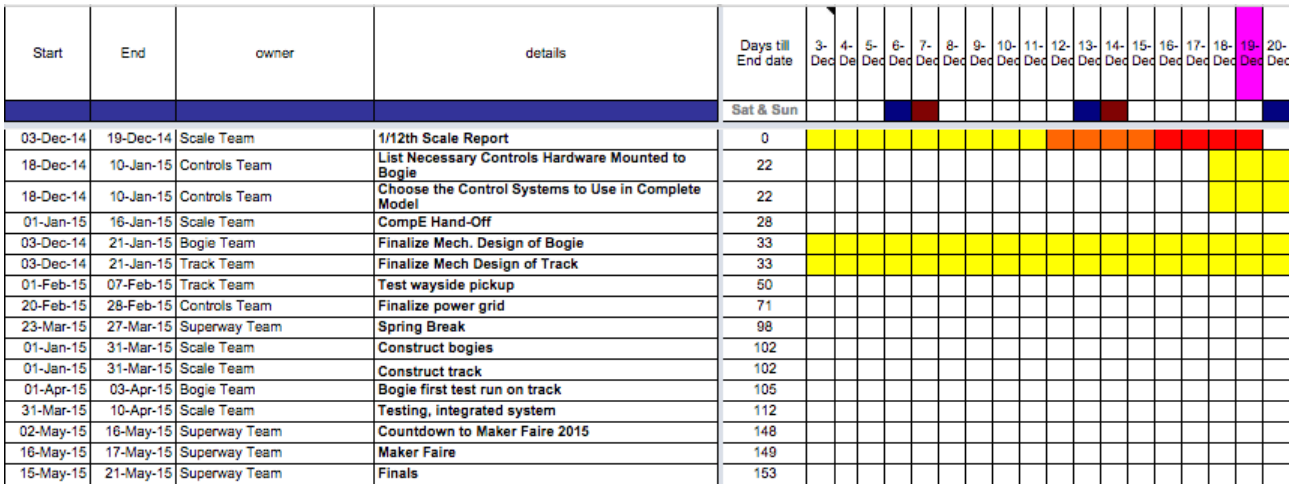
Controls Team

The controls teams has worked with the computer engineering students to determine the protocols for communication between the central controller and individual bogies as well as a system to track bogie positions along the track. Magnetic and distance sensor hardware have also been specified for controls.

➢ Conclusion & Next Steps

Preparation for next semester is underway with any final design changes scheduled for completion by the end of the year. The computer engineering team has completed their portion of the project that will be handed off to the 1/12th mechanical team in the first weeks of the new year. Select team members of the 1/12th mechanical team will be familiarizing themselves with Python over Winter in preparation of receiving them computer engineer's work. Additional parts and material will be ordered over Winter break to ensure prompt arrival for fabrication and assembly in the Spring.

With Maker Faire 2015 scheduled to begin May 16th 2015, the 1/12th scale model's first public demonstration is fast approaching. To prepare for the Maker Faire deadline, a major milestone of March 31st, 2015 as be chosen as the completion date for the fabrication and assembly of both the track and bogie assemblies. Completion by the end of March allows for adjustment and tuning of final design/hardware once full testing begins in April 2015. Detailed scheduling of a completion timeline has begun in preparation for the end of the Spring ME195B Semester. The current Gantt chart can be seen in Figure 23.



Start	End	owner	details	Days till End date
				Sat & Sun
03-Dec-14	19-Dec-14	Scale Team	**1/12th Scale Report**	0
18-Dec-14	10-Jan-15	Controls Team	**List Necessary Controls Hardware Mounted to Bogie**	22
18-Dec-14	10-Jan-15	Controls Team	**Choose the Control Systems to Use in Complete Model**	22
01-Jan-15	16-Jan-15	Scale Team	**CompE Hand-Off**	28
03-Dec-14	21-Jan-15	Bogie Team	**Finalize Mech. Design of Bogie**	33
03-Dec-14	21-Jan-15	Track Team	**Finalize Mech Design of Track**	33
01-Feb-15	07-Feb-15	Track Team	**Test wayside pickup**	50
20-Feb-15	28-Feb-15	Controls Team	**Finalize power grid**	71
23-Mar-15	27-Mar-15	Superway Team	**Spring Break**	98
01-Jan-15	31-Mar-15	Scale Team	**Construct bogies**	102
01-Jan-15	31-Mar-15	Scale Team	**Construct track**	102
01-Apr-15	03-Apr-15	Bogie Team	**Bogie first test run on track**	105
31-Mar-15	10-Apr-15	Scale Team	**Testing, integrated system**	112
02-May-15	16-May-15	Superway Team	**Countdown to Maker Faire 2015**	148
16-May-15	17-May-15	Superway Team	**Maker Faire**	149
15-May-15	21-May-15	Superway Team	**Finals**	153

Figure 23: Gantt chart for Spring ME195B Semester

References

Faludi, Robert. "1,2 and 3." *Building Wireless Sensor Networks*. Beijing: O'Reilly, 2011. Print.
Gustafsson, Bengt. Track and Bogie for Suspended Vehicles. Beamways Ab, assignee. Patent US20120125221. 2 June 2009. Print.

College of Engineering. *Spartan Superway - A Solar-Powered Automated Transportation System*. Rep. 1st ed. Vol. 2. San Jose: San Jose State U, 2013. Print.

Appendix

Appendix A – 2013 Superway 1/12th Arduino Code
Appendix B – X-Bee Communication Protocol
Appendix C – Ultrasonic Distance Sensor Function
Appendix D – Arduino Code using Arduino PID library for Bogie Speed
Appendix E – Mechanical Drawings of Current Bogie Design
Appendix F – Mechanical Drawings of Current Track Design

Appendix A: 2013 Superway 1/12th Arduino Code

```
/*
This sketch is used to manually test the Navigation system.
The motors are not used in this sketch, so the pod has to be pushed by hand.
*/

#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 6  // The pin attached to the Offline Hall sensor
#define TRACK_ONLINE  7  // The pin attached to the Online Hall 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         // The pin used to control the motor via PWM
#define MOTOR_A2  2         // The pin second motor pin
#define TRIG  6         // The pin attached to the ultrasonic sensor emitter
#define ECHO  7         // The pin attached to the ultrasonic sensor receiver

// Miscellaneous definitions:
#define ONLINE   1   // Not on an offline segment
#define OFFLINE  0   // On an offline segment
#define THRESHOLD_ONLINE  800          // The maximum amount of light reflected from the
markers to trigger the ONLINE sensor
#define THRESHOLD_OFFLINE 800          // The maximum amount of light reflected from the
markers to trigger the OFFLINE sensor
#define BOUNCE_TIME  250  // The time delay to prevent multiple readings (debounce)
#define NO_DESTINATION  -1   // The value for when a pod does not have a specified
destination, and able to receive a new destination
#define MAX_RANGE 20     // Maximum detection range (cm) at which the pod will begin to
slow down
#define MIN_RANGE 10      // Distance (cm) at which the pod will perform active braking
#define MAX_DUTY  75      // 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        // Node for station 1
#define STATION_2  2        // Node for station 2
#define STATION_3  6        // Node for station 3


// SPI Variables:
int buf [5];                // buffer data array
volatile byte pos;          // postion in the buffer array
volatile boolean process_it;  // tracks when data is ready for processing

// 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 control debounce
int destination;  // destination station number (1,2, or 3)
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;       // previous locationNode value
// Object Detection Variables:
//NewPing sonar(TRIG, ECHO, MAX_RANGE);    // ultrasonic sensor object from NewPing Library
int range;                          // most recent ultrasonic sensor reading
// Speed Control Variables:
int dutyCycle;              // motor duty cycle (PWM)
//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;     // set the starting node (most likely a station)
  locationTick = START_TICK;        // set the starting node (1 if at a station)
  locationLine = START_LINE;        // set whether the pod starts online or offline
  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(3000);           // time delay before the pod begins to move
  // ********************************************************

}  // end setup

void loop()
{
  /*
  SPI COMMUNICATION
  */
  if (process_it)           // if new data is ready for processing
          {
          if( -1 == destination )  // if the pod is also ready to receive a new destination
          {
          switch( buf[0] )  // the 0 index determines the type of command the pod is receiving
          {
          case 1:  // mode 1 specifies a new destination and a new starting point at a station (pod
must be at a station)
          destination = destinationWrite(buf[1]);          // the index 1 number is a new destination
          locationNode = destinationWrite(buf[2]);   // the index 2 number is a station the pod is
departing
          locationTick = 1;  // if the pod is at a station, the tick number will be 1
```

```
        locationLine = 0;  // and the pod will be on the offline
        break;
        case 2:  // mode 2 specifies only a new destination and maintains the pods current
position
        destination = destinationWrite(buf[1]);  // the index 1 number is the destination
        break;
        case 111:  // mode 111 specifies and new postion for remotely reseting the pod (without
physically moving it to a station)
        locationNode = buf[1];  // index 1 is the node
        locationTick = buf[2];  // index 2 is the tick (0, 1, or 2)
        locationLine = buf[3];  // index 3 is the line (online or offline)
        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;  // reset the buffer index
        process_it = false;  // and the processing tracker
        }  // end of flag set
 /*
 ***NAVIGATION***
 */
 // Take new sensor readings:
 sensorOfflineNew = digitalRead(TRACK_OFFLINE);  // store the offline sensor reading
 sensorOnlineNew = digitalRead(TRACK_ONLINE);        // sotre the online sensor reading
 //Serial.print("Offline:  ");
 //Serial.print(sensorOfflineNew);
 //Serial.print("  Online:  ");
 //Serial.println(sensorOnlineNew);
 /*
 // Determine the new sensor states ( 1 = on marker, 0 = off marker )
```

```
  if( readingOnline < THRESHOLD_ONLINE )  // if the online reading is less than the the lighting threshold
  {
        sensorOnlineNew = 1;  // the sensor is reading the marker
  } else {
        sensorOnlineNew = 0;  // the sensor is not reading the marker
  }
  if( readingOffline < THRESHOLD_OFFLINE )   // if the offline reading is less than the lighting threshold
  {
        sensorOfflineNew = 1;  // the offline sensor is on the marker
  } else {
        sensorOfflineNew = 0;  // the offline sensor is off the marker
  }
  */
  // Compare and possibly reset the Stored values based off of the New values.
  // Offline Update:
  if( sensorOfflineStored > sensorOfflineNew )  // The leading edge condition
  {
        sensorOfflineStored = sensorOfflineNew;  // Update the previous state
  }
  else if( sensorOfflineStored < sensorOfflineNew )  // The trailing edge condition
  {
        sensorOfflineStored = sensorOfflineNew;     // Update the previous state
        if( millis() - timerOffline > BOUNCE_TIME )  // Debounce condition
        {
        Serial.println("Offline: Trailing Edge");
        Serial.println("----------------------");
        // Increment the tick counter:
        switch( locationTick )  // the Offline sensor is triggered by secondary markers (locationTick) at station junctions
        {
        case 2: // the pod is leaving an offline segment and re-entering the online segment
        locationTick = 0;  // reset locationTick for the next station junction
        locationLine = 1;  // set the line to the online segment
        break;
        default:  // if the pod is not leaving an offline segment
        locationTick++;  // increment the tick counter
        break;
        }  // end locationTick update
```

```
        timerOffline = millis();  // reset the debounce timer
        // Manipulate the Switch:
        switch( locationTick )
        {
        case 1:  // the pod has just passed a fork at the beginning of the junction
        switchWrite(2);  // Turn the switch off
        if( OFFLINE == locationLine && locationNode == destination ) // if the pod is at the
offline segment of the destination node
        {
        destination = -1;  // Arrived at specified destination
        }
        break;
        case 2: // the pod is entering the merging point at the end of the junction
        if( ONLINE == locationLine )  // The pod is exiting an Online section of a junction
        {
        switchWrite(ONLINE);  // Turn on the online switch
        }
        else if( OFFLINE == locationLine ) // The pod is exiting the offline section of a junction
        {
        switchWrite(OFFLINE); // Turn on the offline switch
        }
        break;
        default:
        break;
        }  // end locationTick switch
        printSnapshot();
        }  // end bounce
  }
  // Online Update:
  if(sensorOnlineStored > sensorOnlineNew )  // The leading edge condition
  {
        sensorOnlineStored = sensorOnlineNew;      // Update the previous condition
  }
  else if( sensorOnlineStored < sensorOnlineNew )  // The trailing edge condition
  {
        sensorOnlineStored = sensorOnlineNew;      // Update the previous condition
        if( millis() - timerOnline > BOUNCE_TIME )  // This if statement prevents multiple
reads
        {
        Serial.println("Online: Trailing Edge");
```

```
        Serial.println("----------------------");
        // Increment the node counter:
        locationNodePrevious = locationNode;  // Update the previous location node
        switch( locationNode )          // locationNode monitors the segment of track the pod is
currently on
        {
        case 4:  //  the pod has reached the junction immediately before the bypass
        if( STATION_3 != destination )  // if the station 3 is not the destination
        {
        locationNode = 10;  // the pod wil take the bypass (node 10)
        }
        else
        {
        locationNode++;  // the pod will not take the bypass and move to node 5
        }
        break;
        case 9:  // the pod has done a complete loop of the track
        locationNode = 0;  // reset the location to the station 1 node (node 0)
        break;
        case 10:  //  the pod is exiting the bypass
        locationNode = 8;  // set the node appropriate node for the exit
        break;
        default:  // no special conditions are met
        locationNode++;  // increment the location normally
        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:  // approaching the merge point for the main track and the bypass exit
        if( 10 == locationNodePrevious )  // if the pod is entering from the bypass
        {
        switchWrite(ONLINE);  // Activate the online switch
        }
        else  // The pod is entring from the station 3 track length
        {
        switchWrite(OFFLINE);  // Activate the offline switch
        }
        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;  // Set the track line to Online
        break;
        }  // end locationNode switch
        printSnapshot();
        timerOffline = millis();  // Reset the debounce timer
        }
}
// 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 )  // The pod does not have a specified 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;  // Set the speed to normal operating speed
  }
  if( MAX_DUTY < dutyCycle )  // Safety precaution in case the duty cycle somehow becomes
higher than the maximum duty cycle
  {
        dutyCycle = MAX_DUTY;  // Set the speed to normal operating speed
  }
  // Run the motors at the adjusted duty cycle:
  analogWrite(MOTOR_A1, dutyCycle);
  digitalWrite(MOTOR_A2, LOW);
  // end Speed Control


}  // end loop

void printSnapshot()  // Prints relevant data to the serial monitor (for debugging)
{
  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 B: Xbee Communication Protocol

Wireless Message Format

The following discusses the format of the messages which will be sent between the main hub and pods throughout the duration of the systems runtime. Each message will be 32-bit long, which is sufficient for the current design of the system's hardware. There are four sections that make up the wireless message. The first two bits indicate the ID of the receiver, the next two bits indicate the sending device. Following the two sections is the Message Type, which is four bits and can represent sixteen different types. The payload that takes the next three bytes represent the data relative to the message type.

The 32-bit wireless message format is as follows:

Receiver	Sender	Message Type	Payload

The receiver and sender sections follow the following table:

Binary Representation	Device
00	Main Hub
01	Pod 1
10	Pod 2
11	Pod 3

Message Type

Hex Value	Sent By	Message Type	Description
0x0	Main Hub	Path	Initial Path Message
0x1	Pod	Path	Confirm Path Message
0x2	Main Hub	Path	Go Path Message
0x3	Pod	Path	Confirm Path Message
0x4	Main Hub	Status	Request Status
0x5	Pod	Status	Status Info
0x6		(Reserved)	
0x7		(Reserved)	
0x8		(Reserved)	
0x9		(Reserved)	
0xA		(Reserved)	
0xB		(Reserved)	
0xC		(Reserved)	
0xD		(Reserved)	
0xE		(Reserved)	
0xF	Main Hub	EMERGENCY SHUT DOWN	Emergency Shut Down

The receiver, sender, and message type sections are represented by one byte altogether. The remaining three bytes represent the payload relative to each particular message type.

Payload information:

Message Type	Payload Description
Path	Every two bits, from left to right, represents an instruction: forward, left, right, stop.
Status	(See Next Table)

Status Message format

Speed	Proximity Sensor Readings						Location			State
23 - 10	9	8	7	6	5	4	3	2	1	0

Bit	Description
0	Indicates if the board is running or not running. 0 for running, 1 for standby.
1-3	Represents location on track. Can represent up to eight different locations on the map.
4-9	Bits 4-5 represents the right sensor readings. Bits 6-7 represents the front center sensor readings Bits 8-9 represents the left sensor readings. Note: Sensor readings have three different enumerations. 0 for extremely close – stop. 1 for relatively close – slow. 2 for far – move at regular speed.
10-23	(Reserved for speed)

Appendix C: Ultrasonic Distance Sensor Function

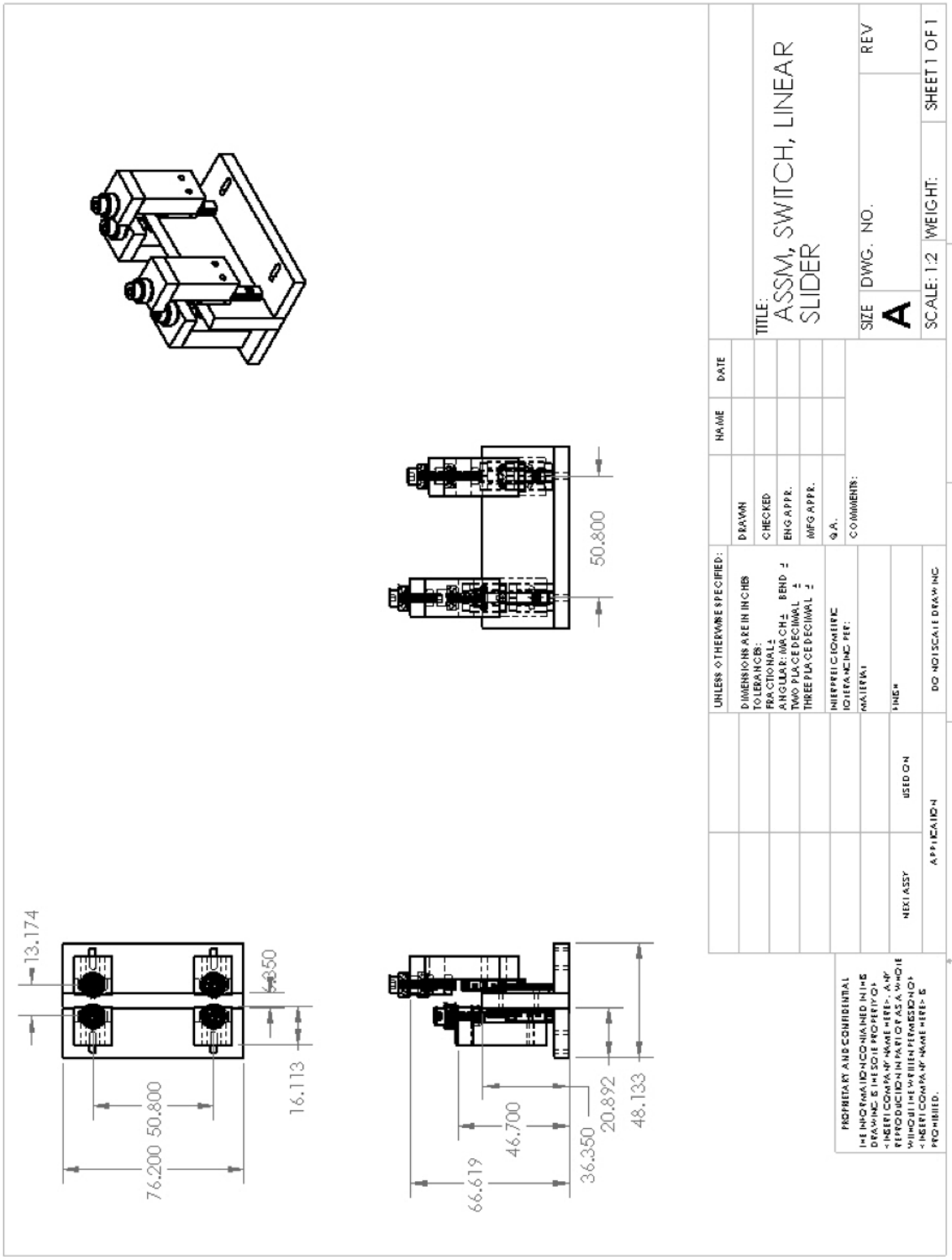
```
//  GetDistance() by Matthew Lewinsky
// Spartan Superway Fall 2014

// Include the following definitions in your code
#define Trig 10
#define Echo 9
#define MAX_Distance 80

// Call GetDistance() to ping distance sensor
float GetDistance()
{
 long duration, cm; // long = integer output, double = floar output
 pinMode(Trig, OUTPUT);// attach pin 3 to Trig
 digitalWrite(Trig, LOW);
 delayMicroseconds(2);
 digitalWrite(Trig, HIGH);
 delayMicroseconds(5);
 digitalWrite(Trig, LOW);
 pinMode (Echo, INPUT);//attach pin 4 to Echo
 duration = pulseIn(Echo, HIGH);
 cm = duration/(29.38*2.0); // Distance in cm
 // Speed of sound = 29.3866996 MICROseconds per Centimeter
 if (cm > MAX_Distance)
 return MAX_Distance;
 else
 return cm;
}
```

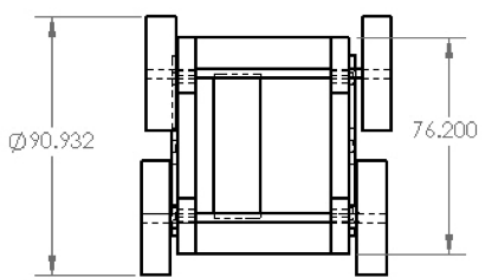
Appendix D: Arduino Code using Arduino PID library for Bogie Speed

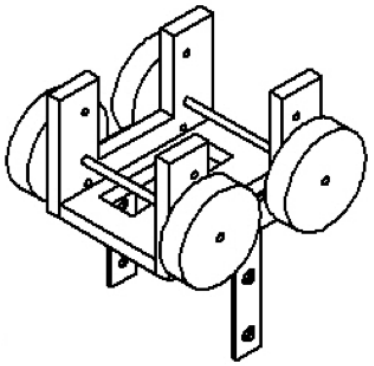
```
//  Speed Control Created by Jared Benson and Jake Pichel
// Spartan Superway Fall 2014
**********************************************************************
/*Program will calculate distance traveled by using a linear encoder
and then adjust the motor speed accordingly using PID control.
**********************************************************************
Count Test Program + Addition of Distance + Nonblocking Timer + PID + Deviation
Program counts each transition from white to black
as read by the QRE1113 IR sensor and gives travel distance.
 */

//****************Revision Note 12/9/2014*******************/////
//Please refer to discussion section above for details about this code
//Code is not consistant with Xbee communication
//to a command terminal. Modifications to the Arduino PID library
//is neccessary before implemented on the 1/12th scale design

#include <PID_v1.h>  //Include PID Library

#define read_time 500 //nonblocking delay time
#define IN1  10   //IN1 and IN2 go to motor
#define IN2  11
#define SensorPin 8  //QRE1113


int previousValue = 0;  //Set Initial Starting Value for counter
float count = 0;   //Start initial count
float BlockLength = 0.0521;  //Length of each transistion block in feet
float TravelDistance = 0;   //Var. for distance traveled
float Velocity = 0; //Velocity of bogie
unsigned long start_time=0;  //Var. for starting time of nonblocking code
unsigned long current_time=0;  //Var. for millis count of nonblocking code
double Setpoint, Input, Output;  //Variables for PID control
double Kp=0.2, Ki=5, Kd=0;  //Tuning Parameters--->Kp proportional,Ki integral, Kd derivative

PID myPID(&Input, &Output, &Setpoint,Kp,Ki,Kd, DIRECT);  //Initialize PID

void setup()
{
  pinMode(SensorPin, INPUT);    //Set as input
  pinMode(IN1, OUTPUT);  //IN1 and IN2 outputs
  pinMode(IN2, OUTPUT);
  digitalWrite(IN1,LOW);  //Write IN1 Low (PID will only change IN2)
  Serial.begin(9600);
  Input = map(count,0,12,0,255);  //map count to analog value
  Setpoint = 149;  //Target value for count (corresponds to # transistions)
  myPID.SetMode(AUTOMATIC);  //turn PID on
}

void loop()
{
```

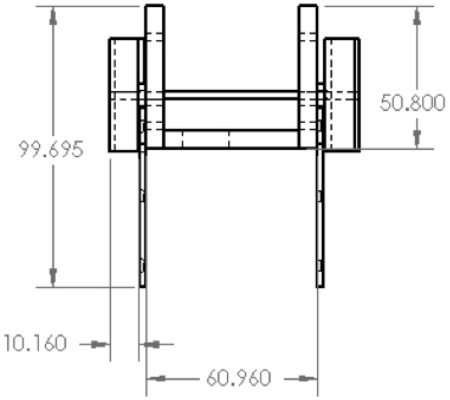
```
current_time=millis();  //start timer

int SensorValue=digitalRead(SensorPin);  //current value of sensor

if(SensorValue != previousValue) //if transition has occurred
{
  count=count+1;    //increase number of transitions by one
  previousValue=SensorValue;  //change to current value of sensor
}

else  //if no transition has occurred
{
  count=count;        //leave all values the same
  previousValue=previousValue;
}

TravelDistance= count * BlockLength;  //calculate distance traveled
Velocity=TravelDistance/read_time;  //calculate velocity

if(current_time - start_time > read_time)  //if desired read time has passed
{
 Input = map(count,0,12,0,255);    //begin PID control
 myPID.Compute();  //performs all PID computations
 analogWrite(IN2,Output);  //writes desired output to system

 Serial.print("Count Value:  ");  //print current count value
 Serial.println(count);

 Serial.print("PID Output Value:  ");  //print current count value
 Serial.println(Output);

//TravelDistance= count * BlockLength;  //calculate distance traveled
//Velocity=TravelDistance/read_time;
//Serial.print("Velocity: ");  //print Velocity in ft/s
//Serial.print(Velocity);
//Serial.println(" ft/s");
//Serial.println(" ");

 start_time=current_time;  //start count over again
 count=0;  //reset count
 }
 //Serial.print("Sensor Value:  ");  //print current sensor value (0 or 1)
 //Serial.println(SensorValue);

 //delay(1000);
 Serial.print("Outside Count Value:  ");  //print current count value
 Serial.println(count);
 //Serial.print(" ");
}
```

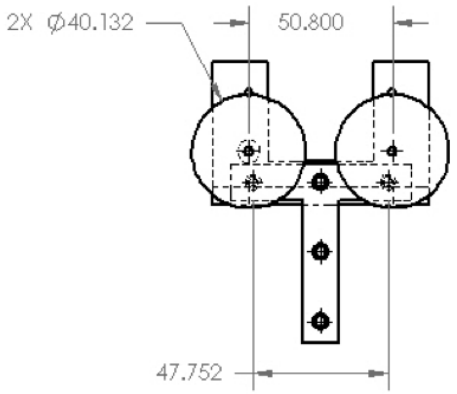
Appendix E: Mechanical Drawings of Current Bogie Design









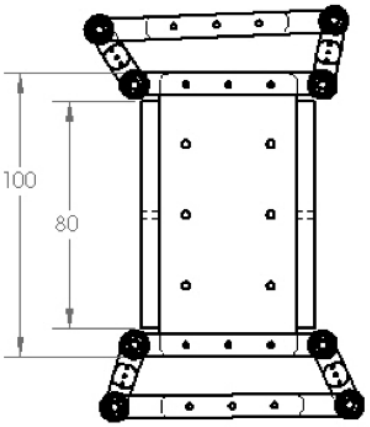


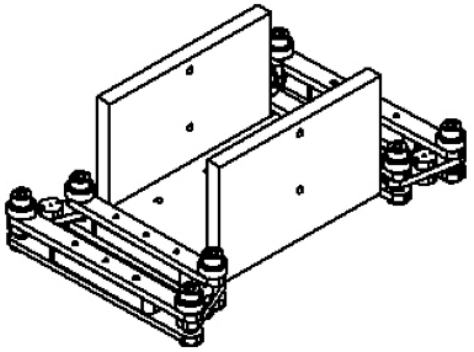
		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
		FINISH			
NEXT ASSY	USED ON				
APPLICATION		DO NOT SCALE DRAWING			

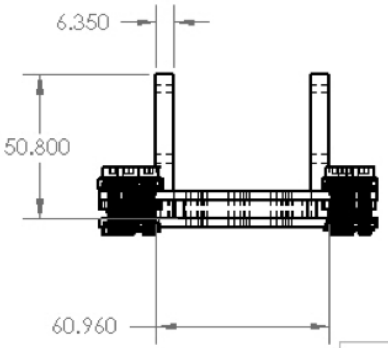
TITLE:
ASSM, PROPULSION SECTION

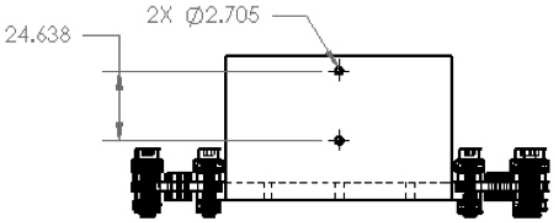
SIZE	DWG. NO.	REV
A		
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1







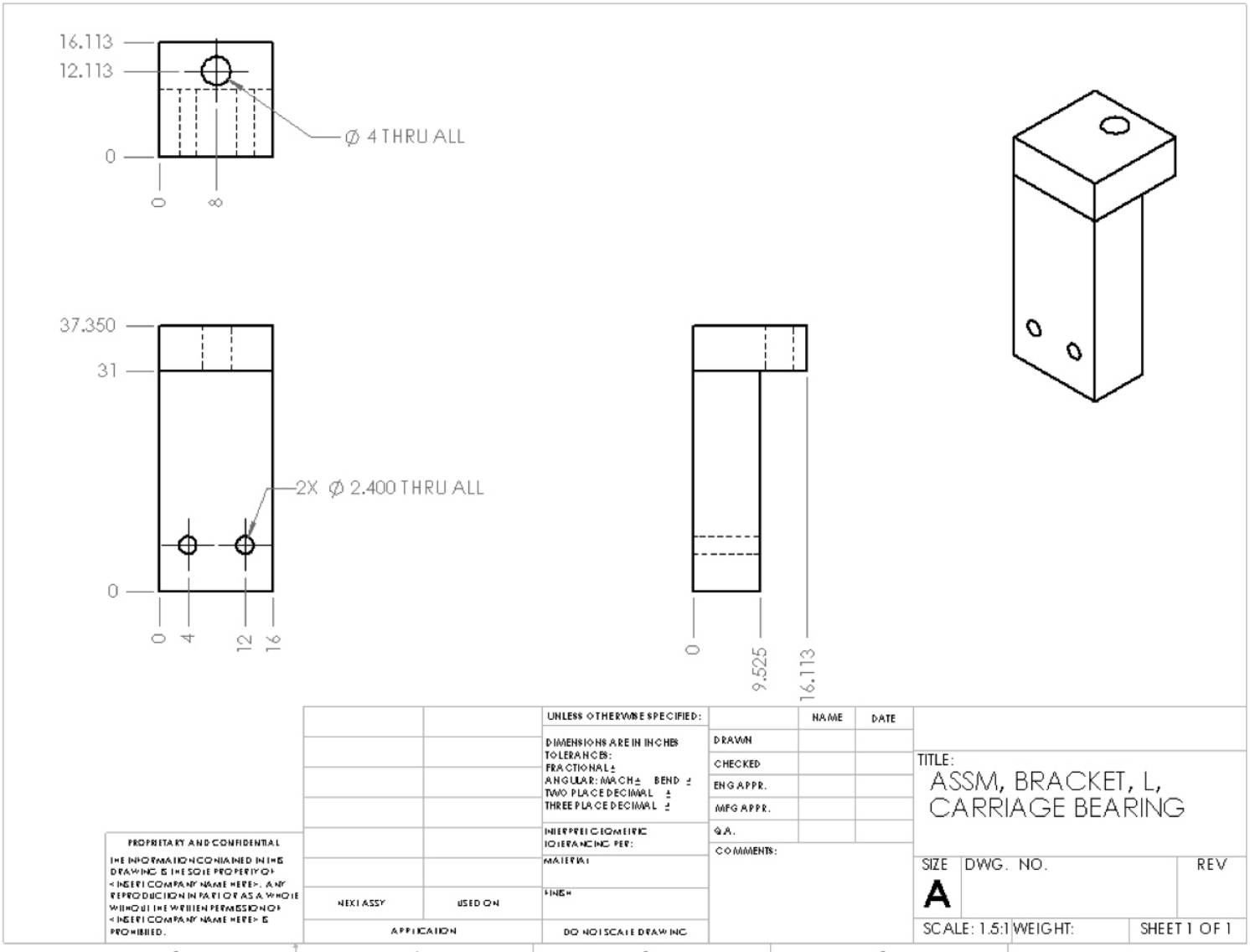


		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

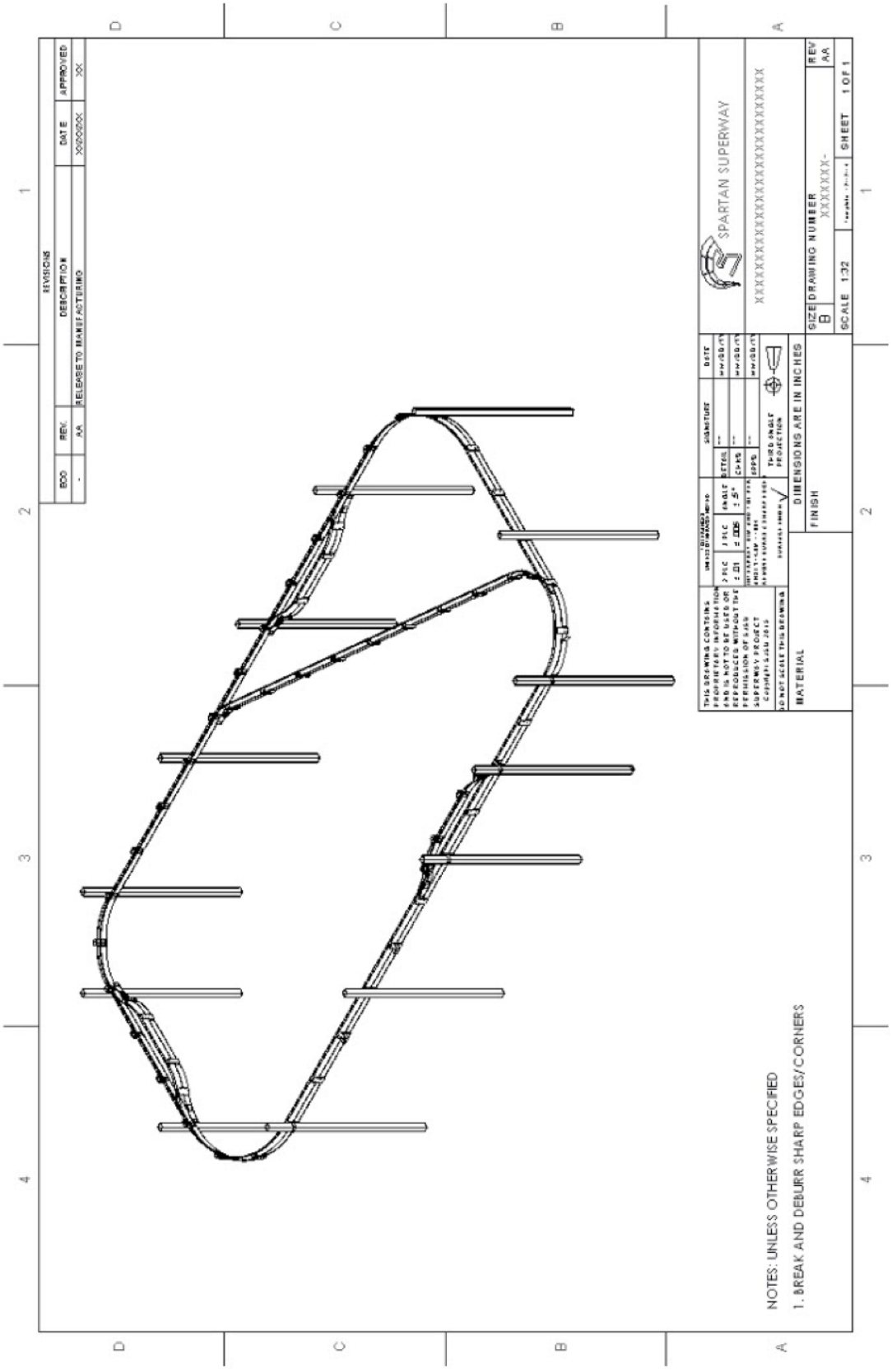
TITLE:
ASSM, FOUR BAR SECTION

SIZE	DWG. NO.	REV
A		
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1

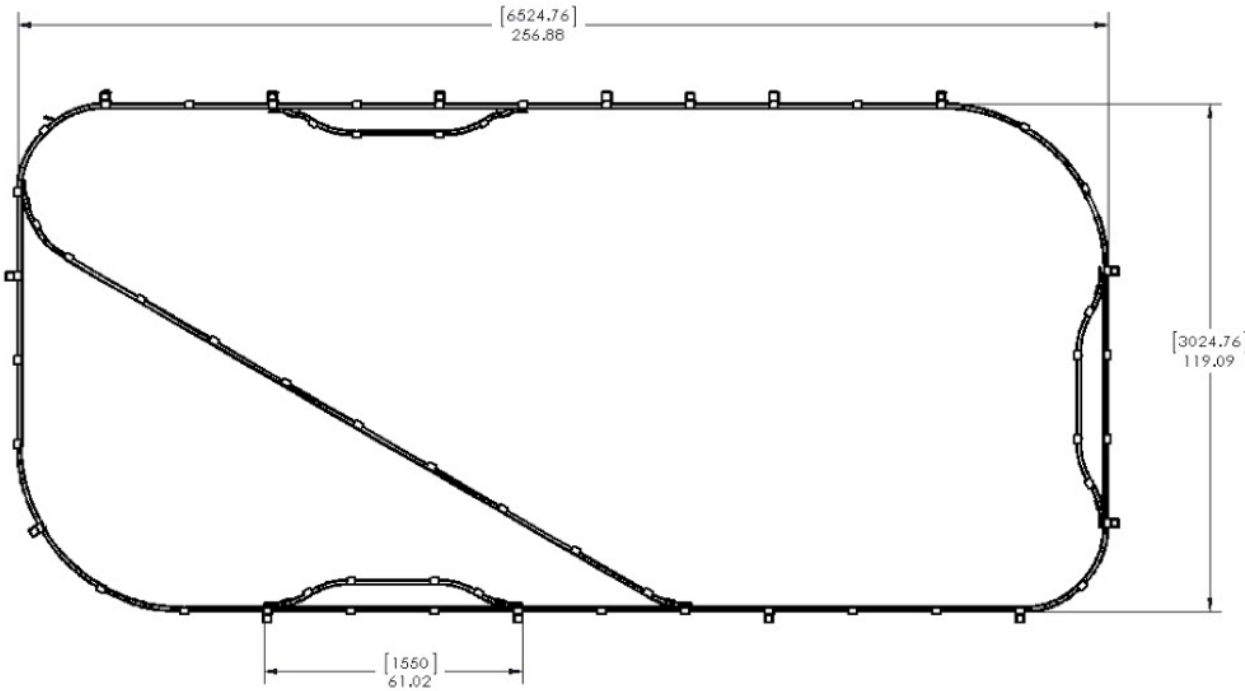
5 4 3 2 1


16.113
12.113
0
0
8
Ø 4 THRU ALL
37.350
31
0
0
4
12
16
2X Ø 2.400 THRU ALL
0
9.525
16.113
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
TITLE:
ASSM, BRACKET, L, CARRIAGE BEARING
SIZE
A
DWG. NO.
REV
SCALE: 1.5:1
WEIGHT:
SHEET 1 OF 1
5
4
3
2
1

Appendix F: Mechanical Drawing of Current Track Design



REVISIONS				
ECO	REV.	DESCRIPTION	DATE	APPROVED
-	AA	RELEASE TO MANUFACTURING	XX/XX/XX	XX



NOTES: UNLESS OTHERWISE SPECIFIED

1. BREAK AND DEBURR SHARP EDGES/CORNERS

TOLERANCES UNLESS OTHERWISE NOTED			
2 PLC	3 PLC	ANGLE	
± .01	± .005	± .5°	

INTERPRET DIM AND TOL PER ANSI Y14.5M - 1994
REMOVE BURRS & SHARP EDGES
SURFACE FINISH √

	SIGNATURE	DATE
DRAWN	--	MM/DD/YY
CHK'D	--	MM/DD/YY
APP'D	--	MM/DD/YY

THIRD ANGLE PROJECTION

MATERIAL

DIMENSIONS ARE IN INCHES

FINISH

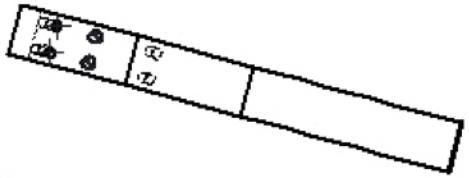
SPARTAN SUPERWAY

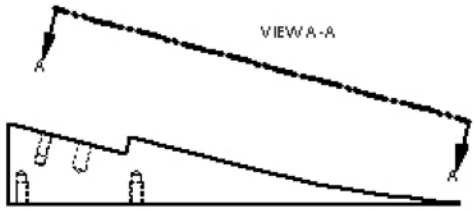
XXXXXXXXXXXXXXXXXXXXXXXXXXXX

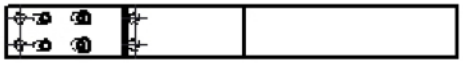
SIZE	DRAWING NUMBER	REV
B	XXXXXXX-	AA

SCALE 1:24 | Template: 12/12/14 | SHEET 1 OF 1

REVISIONS				
ECO	REV.	DESCRIPTION	DATE	APPROVED
-	66	RELEASE TO MANUFACTURING	XXXXXXX	XX







NOTES: UNLESS OTHERWISE SPECIFIED

1. BREAK AND DEBURR SHARP EDGES/CORNERS

			SIGNATURE	DATE
2 PLC. ± .01	3 PLC. ± .005	ANGLE ±.5°	DRAWN	--
			CHKD	--
			APPD	--

THIRD ANGLE PROJECTION

SPARTAN SUPERWAY

X

MATERIAL

DIMENSIONS ARE IN INCHES

FINISH

SIZE	DRAWING NUMBER	REV
B	X-	AA

SCALE 1:1 | SHEET 1 OF 1

REVISIONS				
ECO	REV.	DESCRIPTION	DATE	APPROVED
-	66	RELEASE TO MANUFACTURING	XXXXXXX	XX