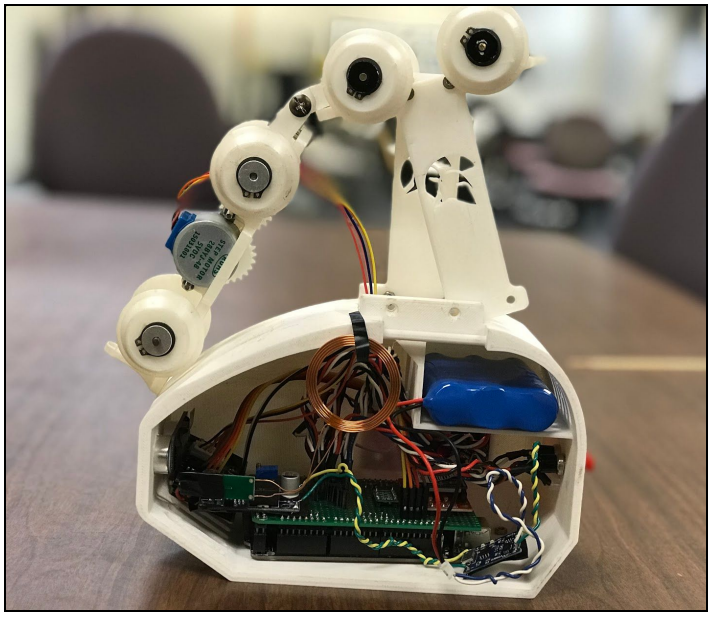
The SPARTAN-Superway: A Solar Powered Automated Rapid Transportation Ascendant Network

Small-Scale Controls System Design



San Jose State University
Mechanical Engineering Department
ME 195B: Senior Project Design II - Section 04
Dr. Burford Furman

May 19, 2019

Authors:
Patrick Barrera
David Mapapa

iNiST SPARTAN SUPERWAY SMALL-SCALE SQUAD

Abstract

As a response to the transportation related difficulties faced by large cities, San Jose State University through their Mechanical Engineering senior design project program have attempted to create a sustainable alternative way of transportation called SPARTAN Superway. As part of the 2018/2019 small scale control's Superway team, we attempted to improve the design made on last year's controls model.

The design changes were made to improve the functionality of the system. This was made possible by substituting the electronic components used to automate, steer and power the system and restructuring the operative design of the system via a new control and User Interface model. The basis of these changes was the difficulties faced last year during the Makers Faire exhibition.

The changes made on the electronics components were the results of a comparative performance between brushed motors and brushless motors in term of usage durability, speed and torque. The motor displayed during the exhibition day was a 28YBJ-48 DC 5V stepper motor and although our stepper motor had a high usage durability, it did not have enough speed to overcome some of track imperfections. As for the charging method, last year's required the team to manually take the podcar off the track in order for it to be charged. Therefore, this year's model have implemented an inductive charging system in the track that for the most part did charge the podcars at a reasonable rate. As for the control system functionality, it was changed from last year by adding functions such as, a pick-up station for the user, sending the nearest podcar to the pick-up location and adding a cancel ride button. As a result of these changes the Arduino code controlling the podcars was altered as well and still requires some improvement.

Acknowledgements

First and foremost, we would like to express our utmost gratitude to: Dr. Burford Furman, our senior project advisor and mentor, for continuously guiding us through this project; Ron Swenson, our supporter for fervently advocating for us; and Eric Hagstrom, for remotely, yet effectively, supervising us and providing useful advices to our project.

In order to obtain some of the concepts and ideas presented in this report, we were inspired by the work done by the previous years control teams. Therefore, we would like to thank Colin Ilas Jr., Andisheh Khosravi-Sereshki, Benjamin Trump as well as Wood Eddie for their inspiring work. During our programing phase, we watched Youtube videos tutorials from Bucky Roberts and Brian Douglas on programing and control system concepts.

Finally, we would like to recognize San Jose State University, the SJSU Mechanical Engineering Department, SJSU Maker Space, and SJSU Associated Students for sponsoring us and offering their services for our project.

Table of Contents

List of Figures

List of Tables

Executive Summary

Introduction and Objectives

In majority of cities in the United States and around the world, CO_2 emitting cars are the most used mean of transportation, therefore creating issues such as fossil fuels pollution, dusts pollution, congestions, accidents, etc. In order to solve those transportation related issues, scientists, engineers and innovators came up with green and eco friendly solutions such as hybrids cars, electric trains, electric vehicles etc. Although those solutions improve some aspects of the transportation industry, they do not resolve problems related to accidents, overcrowding and traffic.

As an attempt to solve these issues SPARTAN Superway offers a Solar Power Autonomous Public transportation system. The system would be composed of fully sustainable network of pod cars facilitating the transit of users. The podcars would be mounted on elevated rails, and fixed through a bogie assembly. Moreover, each Superway stations would allow users to order podcars at any given moment without any schedule restriction, unlike buses and light rail. As aforementioned, the podcars would be on elevated rails, which would not compete with other transportations vehicles or infrastructure located on the ground.

The Spartan Superway project is composed of three different teams (Full scale, half scale and small scale) representing the functionality of the system on different scale. As part of the small scale controls team, we focused on building a controllable system that would allow the user to order any podcar at a pick up stations, to any desired drop off stations, as well as avoiding collisions with other podcars in the process. This whole system would be controlled via a Raspberry Pi tablet User Interface and Arduino Megas in each pod car device.

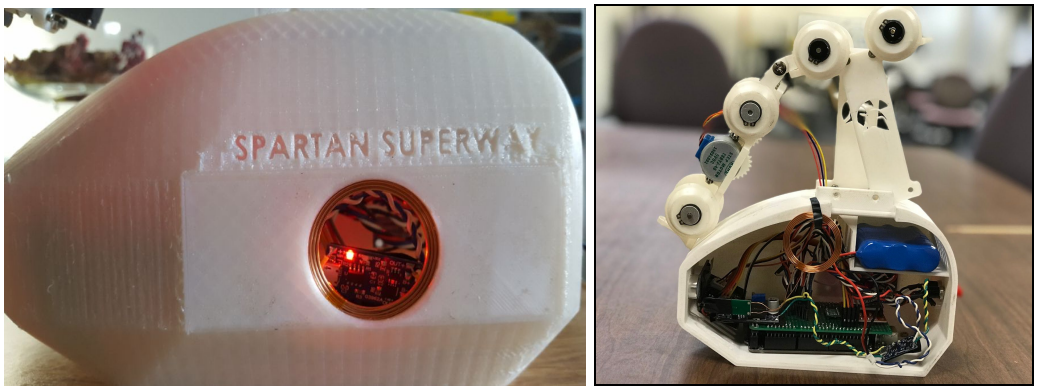


Figure 1. Completed podcar

Procedure and Results

The above illustrated figure was the end design of this year model. Extra components such as inductive charger emitter and receptor were added to the system as well as a charging station. Furthermore, some Graphical User Interface functions on the Raspberry Pi tablet were added to answer real life demand. Lastly, the motor was changed from a brushed DC motor to a 28YBJ-48 DC 5V stepper motor to attempt to increase the usage durability and avoid over usage during the display date.

Conclusion and Recommendations

As our control system design changes were implemented, the functionality of the system were improved, but also resulted in severals unattended consequences. The changes made to the motor coupled with the new track and boogie design, did not allow for the pod cars to freely move along the track as desired due to some slippage between the wheels and track. Nonetheless, this setback can be improved by proper leveling of the track as well as a bogie system that would allow for the wheel to stick to the track at all time. The charging station were operational as designed, however adjusting the height and the horizontal distance from the receiver coil placed on the door of the podcar, to the emitter coil on the charging station would result in an even better charging power. As for the system's automation, the new Graphical User Interface (GUI) design was operational. Although few testing were made on the track due to the aforementioned issue of the wheels slipping, a strong base code system and device modules were established this year for upcoming years to improve on. Additional technical background information would further be discussed in order to facilitate future design improvements.

Introduction

Goals of the SPARTAN Superway

The main objective of the Superway team is to provide a simple, but yet pertinent solution to the issue of air pollution. The automated network system of podcars used in the project, will be powered by solar energy source making it totally sustainable. Moreover, Spartan Superway intends to tackle issues related to public transportation and congestion in major cities. As more cities expand and become urbanized, the number of vehicles per household also increase. This phenomenon leads to major traffics, accidents, degradation of roads, constant road repairs etc. Having a rail system placed at a height on top of the ground so that it may not

compete with any infrastructure and objects whether it is vehicle, train, buse or even human beings and their activities.

Transportation Issues in big cities

In multiples big conglomerate cities around the world, transportations related issues are very common. Those issues range from air pollutions, congestions, accidents, etc. Matter of fact in a study made by Peijun Rong on the Spatial differentiation of daily travel carbon emissions in small and medium sized cities, his results showed that in areas with a great amount of infrastructures, carbon emissions were primarily in the new urban development zone and in the more rapidly-expanding built-up areas in the outer layer of the city. Moreover, he argued that rapidly urbanizing areas could be a leading factor increasing resident travel carbon emissions (Peijun 1). As those cities get larger and more populated, people buy more vehicles to go from point A to B. He goes on and shows the spike in percentage of households owning cars, varying from 24 to 35% in less than a year period in some region of China (Peijun 1370). In a study made by Evans, Tom on how Green the Silicon Valley is, he concluded that the more tech companies in Silicon Valley the more the ecology and environment was affected. This increase in pollution index was present in infrastructures, commute and transportations and much more (Evans 2).

Berkeley Planning Journal 17 (2004)

Table 1: Silicon Valley Environmental Index Trends, 1999.

	SV Environmental Index	
Topic	Measure	Total Change of Indicator
Resource Use	Energy Use	+20%
	Urban Land	+4%
	Density	+18%
	Water Use	+34%
	Waste	-17%
Population	People	31%
Air	Ozone days	-46%
	PM10 days	-73%
	commute alone	+9%
	VMT	+85%
	VMT / cap	+45%
	CO2	+19%
	Ozone Depletion	-77%
	Indoor Air	CA spend 87% of time in-
Water	Watershed Health	5 (scale 1-6, 6 worst)
	Drinking Water Compliance	over 99%
	MTBE cases	-38%
Species/Habitat	Endanger Species	33 threatened
	Tidal Marshes	-84%
	Clapper Rail Pop	+160%
	Burrowing Owl Pop.	-50%
Hazardous Materials	Toxic Chemical Release	-55%
	Toxic Recycling or Incineration	+50%
	Agriculture Toxic Pesticide Use	-51%
	Total Pesticide Use	+6%
	Hazardous Waste Generation	+15%

Table 1. Adaptation of the 1999 Silicon Valley Environmental Index

Introduction to Automated Transit Networks (ATN)

An Automated Transit Network (ATN) can be defined as a self operative system of vehicles synchronized by a predefined control system. An ATN does not require a driver steering the vehicle around in real time, but instead the trajectory or path of each vehicle is determined by the user's preference and the other surrounding cars. ATN system finds the optimal path and route to avoid congestion, unnecessary wait and human error such as missing an exit, not knowing the route etc. While using ATN users do not have to make multiple stops in addition to the one they selected. ATN also decreases public transportation overall cost due to the predefined simple and robust design.

Our paper would describe the objectives and background context of the small scale controls team. Then, we will further discuss the goals and accomplishments of our subteam for this year's project. Followed by the design specifications that were necessary to build our prototypes. We will also thoroughly detail our design, our analysis, alterations and as well as all the testings performed in order to obtain our results. Finally, we will discuss about the pricing and improvements that can be made on our design by next year's team.

Small Scale Controls Background and Importance

As SPARTAN-Superway thrive to improve the transportation industry in a groundbreaking manner, it can be quite challenging to design, analyse, test and validate a scalable prototype due to the size of the project. Therefore, as the small scale team, we can implement ideas, test and improve on different concepts. Past years have made plenty of great improvements and accomplishments, such as implementing a two rail system, setting a foundation for wireless data transmission, implementing a servo arm in order to switch rails, packaging an overall robust working system, etc.

As part of this year's control team we focused on improving the system's functionality by adding a wireless charging station, which helped make our system more autonomous and helped depict what a full scale model should aspire to provide as a working ATN system. Although the full scale model might not use the exact charging system, we however can demonstrate that the powering system would be automated. Moreover, the small scale control team also set a foundation for an application user interface that would be a big part of Spartan-Superway. As a small scale controls, our goals is to maximize the autonomous factor of the ATN system so that we can demonstrate a scalable fully working prototype

Objectives

The purpose of the SPARTAN-Superway Small-Scale model is to act as the proof-of-concept model that represents a scaled-down version of the Full-Scale model, as it is more easily transportable and than the full-scale model. The Small-Scale team is divided into three subteams: Track Design, Bogie Design, and Control System Design, with the collective goal of developing the small-scale model of SPARTAN-Superway. As the Small-Scale Controls Team, our ultimate objective for senior project in particular, is to design a functional and effective controls system that will power and drive the network of pod cars along the track. This involved hardware and software developments.

While our first semester of senior project, ME 195A, focused on brainstorming ideas for controls systems designs and proposing our design ideas, the second semester, ME 195B, saw us

developing a prototype to test our control system, finalizing our designs, and constructing our controls system. Ultimately, we addressed five objectives: (1) implement wireless charging, (2) develop 3D CAD models to incorporate the wireless charging stations, (3) improve battery charge rates, (4) replace motors driving the pod cars, and (5) improve system functionality and user interface. The first four objectives involved hardware developments and resulted in the creation of a circuit supplementary to the pre-existing circuit. The fifth objective involved software development and thus improvement of the Arduino and Python code of last year's team.

Design Requirements and Specifications

In order for our design system to be truly operational, we had to at least satisfy those predefined requirements and specifications. First, select a motor that would not wear off after few hours of display, as well as move the podcar system at a speed of at least 0.5 m/s. Then, we would have to design a charging hub station that would allow the battery, powering all the electronics and Arduino inside the podcar, to be able to charged without having to take the pod car outside of the track. Design a GUI that would allow users to request a podcar from their pick up location, then drop them off at their destination, with an option of cancelling the ride. Finally, allow the instructions selected by the user on the Raspberry Pi tablet to be communicated to the Arduino Mega that are located inside the podcars and vice versa. In recap our specifications design can be listed as such:

- Select a motor that would move the podcar system at a speed of at least 0.5m/s and implement it in the Arduino code
- Select a battery that would charge the system at 5V and 800mA
- Select inductive charging components that would allow the battery to be charged with current discharged of at least 700 mA
- Improve the python code on the Raspberry Pi tablet, so that users would have more menu selection
- Allow data flow between the Raspberry Pi and the Arduino Mega through a communication module that would go up to 20 feet

State-of-the-Art Literature Review

One of the most effective illustration of ATN used to this date is the Morgantown Personal Rapid Transit connecting the two campuses of West Virginia University. Both campuses are approximately 2 miles apart from each other. The system was designed such that users would pay a certain fee at the station, pick their destination and the vehicle available would transport them to their desire stop without stopping through unwanted stations. The vehicles

however can contain an approximate number of 20 individuals and would therefore stop at each required station requested by the individual in the podcar. According to the MINETA transportation transit, the Morgantown system serves up to 15 000 people a day and more than 60 millions users since 1975.

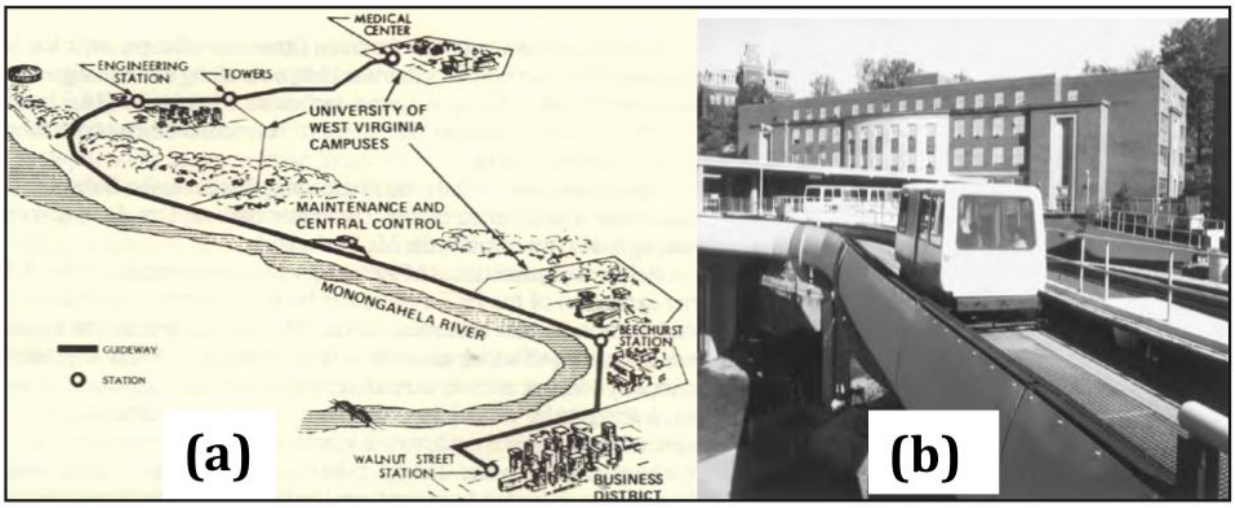


Figure 2. Morgantown Personal Rapid Transit

Prior and current work done on the SPARTAN Superway

Since 2012, Spartan Superway has been generating innovative ideas in regards to ATN to improve public transport. The project started in 2012 as a senior project class under the supervision of Professor Burford Furman and Ron Swenson. Both came up with the innovative idea of creating an ATN powered by sustainable energy. In order to fully tackle the issue, they created two major teams that would try to replicate the system functionality at different scale. Each major team consisted of design, control, guideway, solar power, propulsion and guideway. The previous year each team made improvements on the design specifications of the previous year to ameliorate the system. Therefore, the implementation of a testing track stations, full scale switching section, bogie prototype and even a half scale functional system were made over the year.

As part of the 2018-2019 small scale control system, the scope of our work was to improve the electronics components used by last year team to actuate the podcars, to find an optimal way of charging the podcars during usage, to decrease collisions occurrence and to improve the user interface design.

Design Details

Wireless Charging

To enforce autonomy, we introduced wireless charging stations into the small-scale model to re-charge the batteries powering the podcars. This was accomplished with induction charging coils, which are comprised of a transmitter coil and a receiver coil. The induction module we ultimately used, shown in Figure 3, below, was a 12V, 600mA wireless charging module because it featured the largest distance between the transmitter and receiver, which was 0.787 inches. Additionally, the induction module seemed to feature a boost converter, as it amplified the supply voltage. For example, supply the transmitter with 5V could boost the voltage up to 12V, dependent on the distance between the coils.

Figure 3 . The induction module used was a 12V, 600 mA wireless charging module with a maximum charging distance of 0.787 inches.

Shown in Figure 4, below, are the placement of each coil: the transmitter is mounted in the charging station (purple), while the receiver coil is placed within the pod car's door (green).

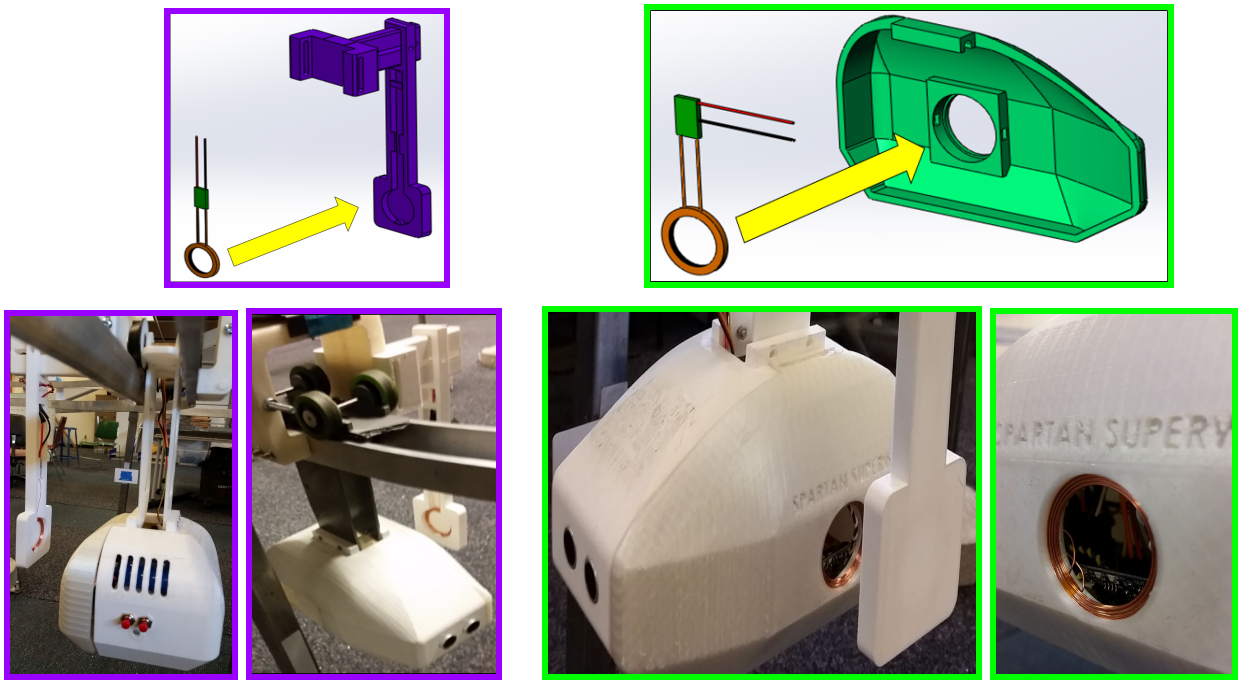


Figure 4. Left 3 images: the induction coil transmitter is mounted in the charging hub. Right 3 images: the induction coil receiver is placed inside the pod car door.

There are two charging station - one in each offline rail, i.e. inner rail. Figure 5 depicts the locations of the two offline rails.

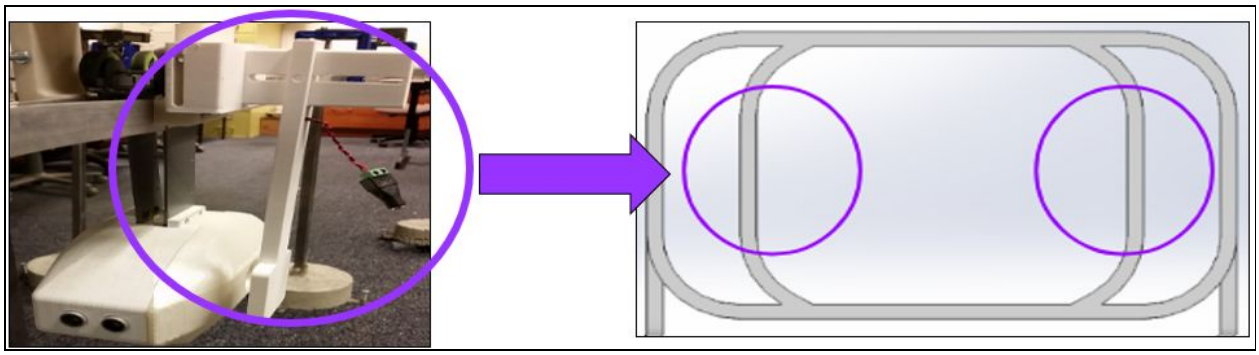


Figure 5 . The two induction charging stations are located in the offline rails along the inner track.

As shown in Figure 6, the charging hub is composed of two pieces: a T-shaped foundation piece and a spatula-shaped piece that slides onto the foundation piece and hangs from it. The foundation has a cut-out vertical slot for height adjustment, while the hanging piece has a cut-out at the top to allow it to slide along the foundation piece for adjusting the distance between the induction hub and the pod car.

Figure 6. Two separate pieces constitute the induction charging hub: a T-shaped foundation piece and a spatula-shaped piece that slides onto the foundation piece and hangs from it.

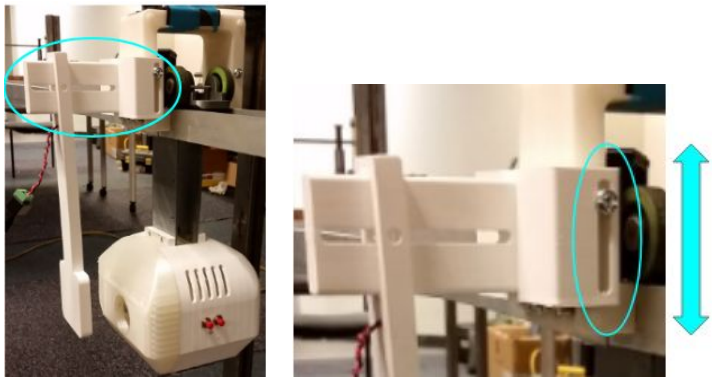


Figure 7. The T-shaped piece of the charging station has a cut-out slot for height adjustment.

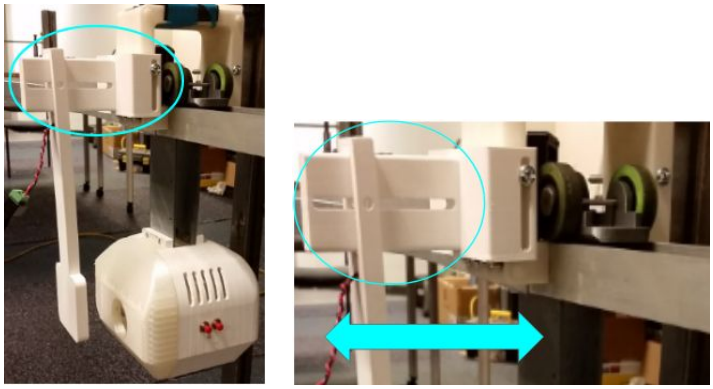


Figure 8 . The spatula-shaped piece has a cut-out at the top to allow it to slide along the foundation piece for adjusting the distance between the induction hub and the pod car.

The pod car door we modeled are modifications of last year's door design: we simply added a hole and an interior square flange into which the receiver coil fits into.

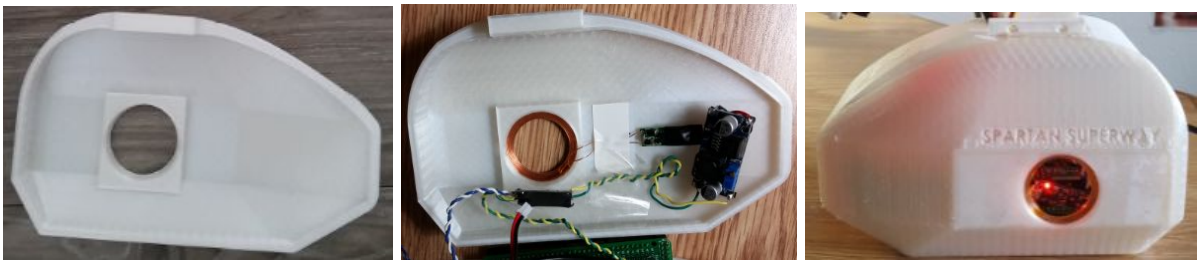


Figure 9. The modified pod car door has a hole and a square flange on the inside into which the receiver coil fits into.

3D CAD Models

Our team created CAD models for two components: the induction charging hub to hold the transmitter coil and the new pod car doors to hold the receiver coil. The charging hub is

comprised of two pieces: a T-shaped foundation piece and a spatula-shaped piece that slides onto the foundation piece and hangs from it. The pod car door is a modified version of last year's SolidWorks part. Our 2D-detail drawings of both components are found in Appendix 1.

One factor that we had to take into consideration was that all of the 3D printers that were available to us had a bed of length 10 inches. Thus, we made sure that the maximum length of the charger hub was 9.5 inches, to prevent possible overshoot past the length of the bed.

Battery Charge Rate

Lithium-polymer and lithium-ion batteries require a specific charger. The battery charger we were to select was supposed to be small enough to fit into the pod car, charge the battery when at an induction charging station, and further enforce autonomy by bypassing the need for the team to manually charge the battery. The only mini lithium battery chargers available are ones that can charge up to 3.7V batteries. Thus, we had to replace last year's 7.4V batteries. The battery we ultimately selected to power each pod car was a 3.7V, 6600 mAh lithium-ion battery. And the charger we ultimately selected to charge this battery was the Makerfocus TP4056 charging module, which has a 5V maximum output charge voltage, a 1A maximum output charge current. The charger also features a protection circuit so that the load, i.e. the Arduino and motor driver board, can be connected to the charger. In the end, this setup circumvented the need to disconnect the battery from the charger, and then connect the battery to the load. It made it possible to simply connect the battery to the charging module to power the circuit, and then charge the battery when at an induction charging station.

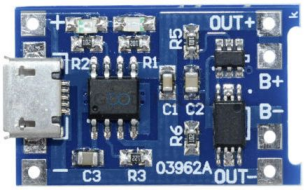


Figure 10. The TP4056 charging module has three connections. The + and - on the left side is where the induction receiver is connected. The OUT+ and OUT- on the right side is where the load (the Arduino and motor driver board) is connected. And the B+ and B- on the right side is where the battery is connected.

The battery charger possesses a red LED that illuminates when the battery is charging, and a blue LED when the battery is fully charged. The charger was placed directly behind the hole in the pod car door so that the user could see that the battery is charging at a charging station.

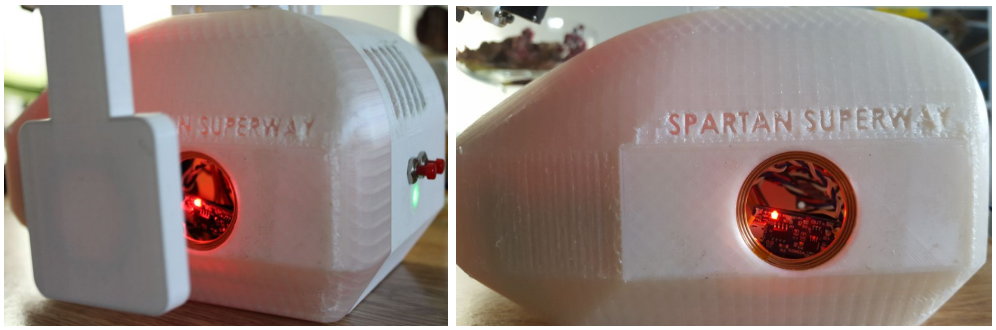


Figure 11. The battery charger was placed behind the hole in the modified pod car door so that the user could easily see that the battery is charging at a charging station; the red LED indicates the battery is charging.

Furthermore, since induction charging is not the most efficient, i.e. fast, way of charging batteries due to their miniscule charge current, we had to find a way to increase the induction charge current. This was resolved by using a buck converter to amplify the current and placing it between the induction receiver and the battery charger. The tradeoff to using the buck converter is that while it does increase the current, it does decrease the voltage. However, this worked to our expectation because, again, the battery charger has a maximum output voltage of 5V, and using using a 12V power supply to power the 12V, 600mA induction charger would fry the battery charger. Additionally, one drawback to incorporating the buck converter was that since it decreased the voltage, it also decreased the maximum charging distance, meaning the transmitter and receiver were required to be closer than without the buck converter.

Figure 12. The buck converter increases current at the cost of decreasing voltage; it was placed between the induction receiver and the battery charger.

With a 9V power supply connected to the induction transmitter, the buck converter decreased the voltage down to approximately 6V, but increased the current to around 900 mA. However, as a precautionary measure to prevent burning out the battery charger, we further adjusted these parameters by decreasing the voltage to approximately 5V and 800 mA through the on-board potentiometer.

Lastly, since the Arduino and stepper motor driver board require at least 5V for proper operation, we had to increase the voltage from the 3.7V that the battery was delivery. This was

accomplished by adding a boost converter to amplify the voltage and placing it after the battery charger and before the Arduino and the stepper driver.

Figure 13. The boost converter increases voltage current at the cost of decreasing current; it was placed after the battery charger and before the Arduino and stepper motor driver board.

Motor replacement

The motor used during our exhibition display was the 28YBJ-48 DC 5V stepper motor. The selection of this motor was a result of several motor changes made over the past two semesters. The intended design objective was to select a motor that would move the podcar at least 0.5 m/s and last over a full day of exhibition. Therefore, we initially moved from a brushed Pololu motor used last year to a Gimbal brushless motor with an ESC. Unfortunately, the gimbal motor and the ESC required more voltages and Amps than the one offered by our battery system. Moreover, the code used for the gimbal motor did not work as intended with the overall system. Therefore, we decided to select a different brushless motor, that ended up not being supported by the ESC as well.

Figure 14. Tiger GB2208 Gimbal motor

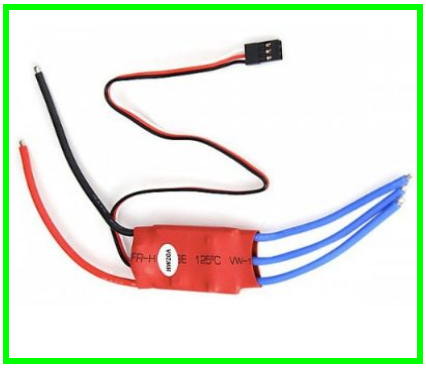


Figure 15. ESC

Finally, we selected a stepper motor that met the minimum speed and longevity requirements. Once tested the stepper did move the podcar system, but due to disability of the bogie to stick on the track the podcar did not move as intended. Switching from a brushed motor to a stepper motor modified the code and functionality of the system. Since stepper motors operate in an open loop system manner, we had to predefined the number of steps that the motor should make before stopping at each station. To determine the maximum speed that the mini-stepper could run at, we utilized the Arduino Stepper library. Specifically, we used the setSpeed( ) function, and discovered that the maximum speed of the motor, with 5V applied, is 30 rpm. Using the internal gears of the stepper motor, this translated to approximately 0.5 m/s.

Figure 16. 28YBJ-48 DC 5V stepper motor and ULN2003 stepper motor driver board for Arduino

User Interface

In order to improve on last year's GUI, we implemented new user inputs functions on the Raspberry Pi tablet. As part of the newly added functions, we added a pick up station selection button, which would offer a 14 stations selection drop down menu that the user could defined as the current station to be picked at. That station option was added to the drop down menu option that consisted of 14 drop off stations that would represent the destination of the user, with station 9 and 14 being the charging station.

Another user input button added was the "cancel ride button" that would display a pop-up box asking the user to confirm his selection or disregard it. Depending on the user's selection, he would be dropped at the next station or continue his ride. Moreover, a request podcar button was added in order to select the pod car closest to the user and send the data to the podcar with the corresponding Arduino in it, to the assigned destination. Finally, we added severals of our sponsors logo on the screen display for publicity and perhaps funding reasons.

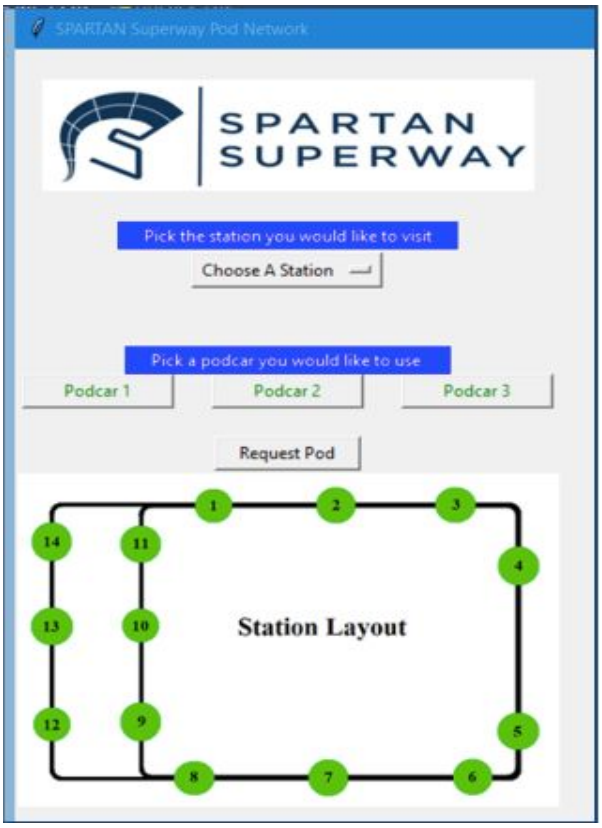
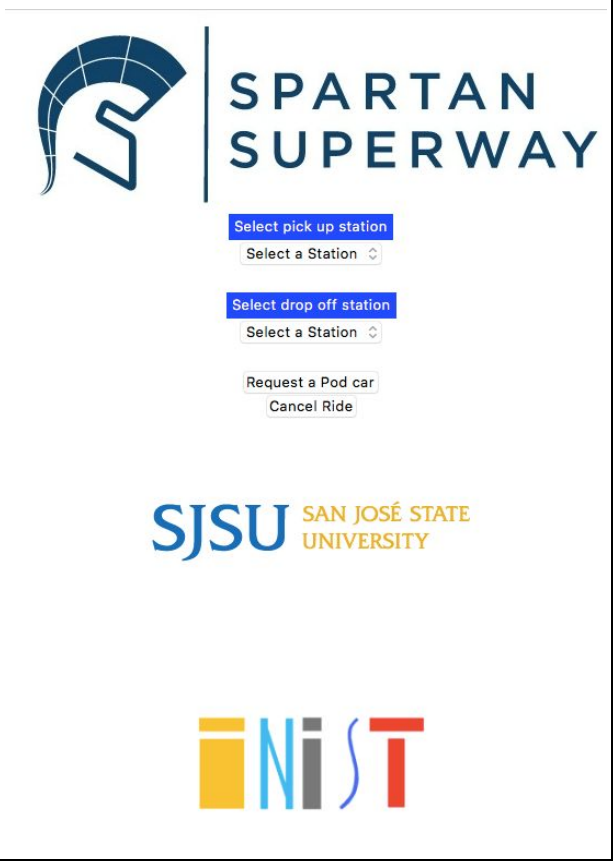
Old GUI	New GUI
	

***Figure 17*.** Old vs New GUI

Xbee Module

In order to send the data selected by the user on the raspberry Pi tablet, a communication module had to be necessary. The selected module for this year's model was the xbee module S2. This module consisted of a set coordinator Xbee S2 module located in the Raspberry Pi tablet and three different Xbee module S2 routers located in each podcars. The Xbee router modules

were interfaced with the arduino Mega on each podcar and would receive the data from the Pi and send it to the Arduino Serial Monitor so that each podcar would know where to go.

Figure 18. Xbee S2

Arduino Mega

An arduino Mega board was located in each podcar, allowing the different sensors to be interfaced, as well as controlling the functionality of the system. In order for our system to be operational, each arduino Mega had to receive data from from the Raspberry Pi tablet through the Xbee serie S2 module. The arduino would also run a parallel CheckDistance() and a CheckButton() function to avoid collision with other podcar through the ultrasonic sensor as well as to stop the podcar whenever the stop button is pressed. The arduino would also have LED lights indicating if the podcar reached the station or if it's waiting for any command from the raspberry Pi. To connect all the sensors and components to the arduino, we used the same connection board as the previous years.

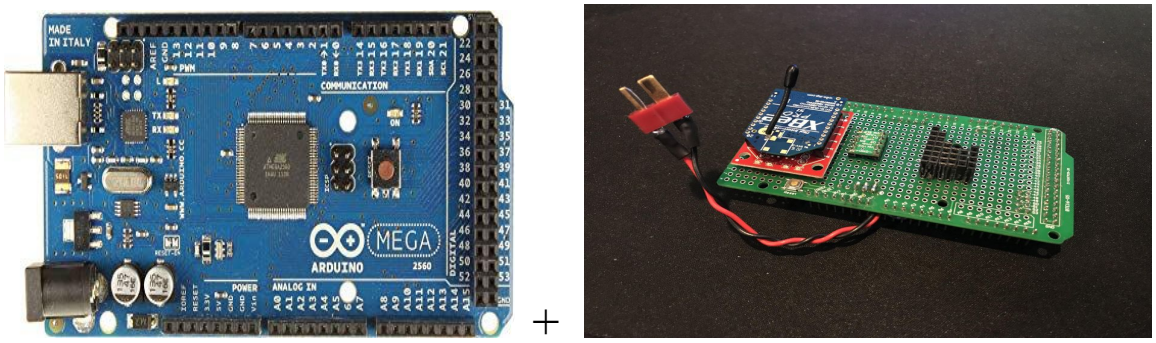


Figure 19. Arduino Mega and connection board shield

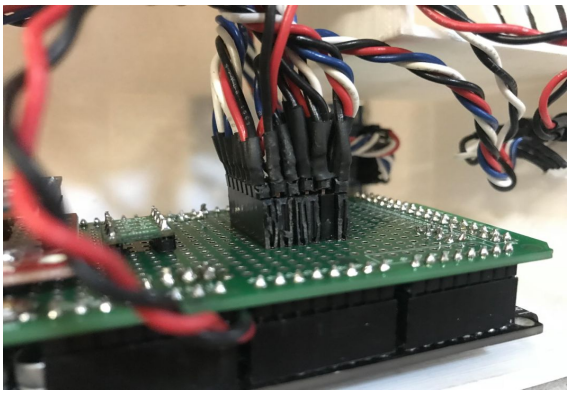


Figure 20. Arduino and shield assembled

Analysis/Validation/Testing

After selecting all the components necessary to automate our system, we proceeded to the testing and validation of our design. Few different design tests were done in parallel and finally the whole system was put back together.

Motor

In order to move the podcars on top of the rail, we decided to selected motors that would last for a great period of time. To do so, we first selected a Tiger GB2208 brushless gimbal motor without a shaft. We then calculated the power requirements in order to integrated that motor with our system on Appendix 5. But unfortunately, the motor did not perform as wanted.

We therefore ended using a stepper motor with max rotation speed of 30 rpm to get us through at least 0.5m/s coupled with gearing calculation made by the bogie team. The stepper motor being high torque, gave the speed that we initially planned but because of the track and bogie setup couldn't be used to its full capacity.

XBee

In order to start the serial communication between the Raspberry Pi tablet and the Arduino boards, we had to set up the xbee modules to the same frequencies and set up. A total of four S2 xbees were used for our system, a coordinator for the Raspberry Pi tablet, and one router for each podcar.

In order for the Xbees to be set up, we had to set them up via the XCTU software. We initially mixed the Xbee S1 model with the Xbee S2 and tried to pair them off without any success. This was due to the fact that the two series were not compatible. We proceeded to select

the S2 models because they were easily detected by the XCTU software and offered better upside as far as pairing the modules and the available options.

We went ahead and downloaded the XCTU software on our computer and connected the Xbees modules via USB port. Once a module was detected, we either set it up as Zigbee coordinator AT mode function for the Pi or as Zigbee router AT mode for the Arduino.

In order for the Xbee to be on the same connection frequency, we first had to set all of them on the same PAN ID and SCAN Channel. Then we had to set the coordinator module DL option as FFFF for broadcast mode (in order to receive and transmit data to all routers). As for the routers, we had to set their DH and DL number to the SH and SL number of the coordinator to specify what module they'll receive information from.

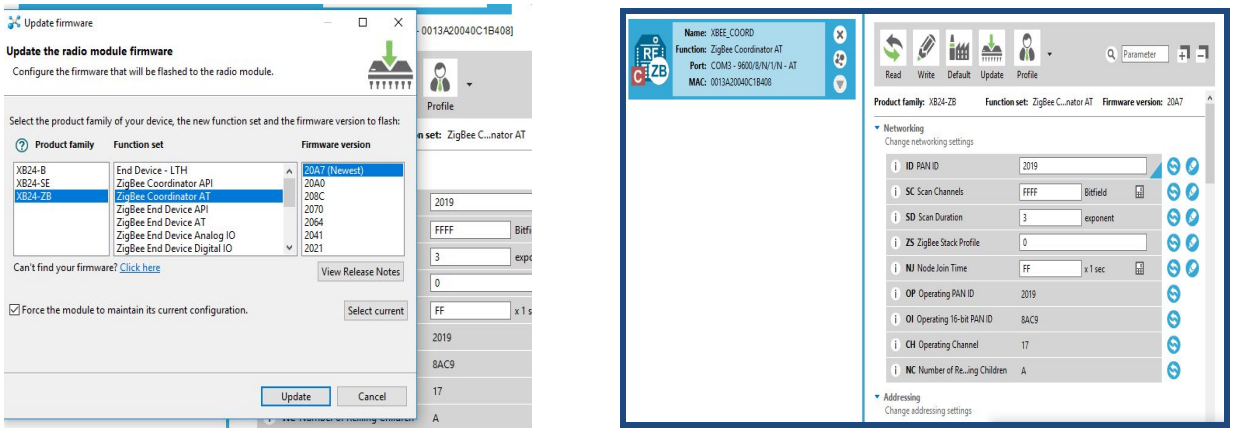


Figure 21. XCTU software module setup

User Interface

The user interface was controlled by the Raspberry Pi tablet. Last year's Raspberry Pi located inside the tablet, did not reboot properly. Therefore, we had to buy a new Raspberry Pi and download the Raspbian software on our computer. Then, we transferred the software to the raspberry Pi via a SD card. Once the Pi was booted and the different program were uploaded via the terminal, we proceeded to code the GUI via the python.

In order to do so, we watched video of Bucky Roberts on youtube about Tkinter User interface. Once we grasp the necessary syntax to update last year's code, we went ahead and modify it to operate the way we wanted to.

The code did incorporate some drop down Menu for both the pickup and Dropoff Menu selection. A cancel ride button, and the sponsors logo as designed. The code can be found in Appendix .

Since the Raspberry Pi requires a monitor (Raspi tablet), a mouse and a keyboard to operate, we decided to download the pycharm software and test our code on its python interface. Then the code was transferred to the Raspi by storing it through our drive and downloading it via internet on the Raspberry Pi. We went through two alterations of our design showed in the figure below.

Figure 22: Old vs. Updated GUI

Arduino Mega

The Arduino code was the base of our system, it received data from the Raspberry Pi and depending on the podcar an if statement was added to read the data and have the podcar be automated. The arduino code was separated in different section depending on the sensor or the electronic components interfaced.

First was the Xbee module, which was tested by sending data from the XCTU coordinator to each xbee on the Arduino Mega. Once those data were received by each module, we went ahead and sent data from the xbee coordinator connected to the Raspberry Pi tablet.

Then, we tested the RFID and Ultrasonic sensors, which both played big role in the system. The RFID emitters were connected to the Arduino Mega and mounted on the door of the podcars. Those RFID readers were set to read the RFID cards set on the , which were representation of the stations. As for the ultrasonic sensor, they were set on the front of the car and were connected to arduino so that the car would stop if any object would come to its proximity.

Finally, we made sure the LED lights lit up blue while waiting for instructions, green while reaching the station, and that push button stop the motor of the podcar.

Bill of Materials

Shown in Table 2, below, is our final bill of materials. Note that this includes shipping and handling costs, but does not include extra parts we ordered for backup and does not include components we ultimately eliminated.

Table 2. Bill of Materials

			FOR 1 POD CAR	
Item No.	**Item Name**	**Cost per Unit**	**Quantity per pod**	**Cost per pod**
1	Induction Charger Module (Transmitter + Receiver)	$14.90	1	$14.90
2	Buck Converter	$10.99	1	$10.99
3	3.7V LiPo Battery mini-Charger	$8.99	1	$8.99
4	3.7V, 6600 mAh LiPo Battery	$39.50	1	$39.50
5	Boost Converter	$8.48	1	$8.48
6	5V Stepper Motor + ULN2003 Driver Board for Arduino	$13.99	1	$13.99
			Total per pod:	***$96.85***

Results and Discussion

In summary, during this year we were able to redesign the control system of last year's small scale team. The changes made were simple, but allowed the system to be more user

friendly, more robust and more performant. Those changes will be used at the foundation of future improvements from future teams.

We substituted the brushed DC motor used last year by a stepper motor in order to increase the usage time at the exhibition date or Maker Fair for next year. The new motor was equipped with some gearing mechanism to increase the speed of the motor to the wanted speed or even better.

We added two inductive charger stations in the middle of the track, in a way that would not require members of the team to constantly have to change the batteries manually. The inductive charging calculations were made prior to buying the components and were soldered to the overall system.

An additional major change made from last year's model, is the change made on how the system operates. Added options such as pick-up stations, a cancel ride option, and a foundation to update the code and bring the closest podcar arrival upon any trip request. These changes were mostly reflected on the GUI of the Raspberry Pi tablet and would eventually have to be fully updated on the arduino.

The first term of the project was focused on designing the optimal control system for the small scale, as well as rapid prototyping. Over Winter Break, we focused on testing our system and debugging our code. During the spring semester we dedicated specifically to the fabrication, specifically developing a CAD model for the pod car, soldering different electronics to the PCB Arduino shield, and constructing the induction charging stations and motor changes. We then, finally, assembled the podcar and implement the code that would enable the system to work as desired, ultimately culminating in our final senior project presentation and last exhibition.

Conclusions and Recommendations for Future Work

We do have suggestions for future research for the next Small-Scale Controls team, in order for them to maximize their time and efforts to get an optimal outcome production.

First, they should spend time thoroughly understanding the different xbee protocols and modules. The difference between an S1 and S2 xbee, the different function settings such as AT mode or API mode, as well as which one will be used for their application. Also, they'll have to get familiar with the different xbee libraries used for python (raspi) and arduino. Since both libraries and syntax are different, this knowledge would be important in terms of sending and receiving data from on end to the other.

They should spend at least the first 2 weeks of their ME 195A class scanning through our report and perhaps last year's report. This will be crucial in terms of setting up their design specification as well as their area of improvements. Moreover, it would make it easier for them not to have to redo works or testings already done in the past, unless decided otherwise.

As far as the GUI and Arduino code are concerned, having a strong foundation on python and C would be necessary to improve that part of the project. Youtube videos by Bucky Roberts as well as talking to the previous team helped us understanding the code. However, if next team of engineer are already familiar with those languages, they should look into other Graphical User Interface such as GTK, pyside or any other program that can be interfaced with windows, Linux and iOS.

Finally, next years team should look into different brushless motor and ESC that can be power interfaced with the arduino and the every other electronics components in the system. An alternative should be to use a stepper motor with a higher speed than the one used this year; keeping in mind that stepper motors are usually open loop motor unless equipped with an encoder. Selecting a compact, durable and programmable motor adequate for their system would be a large part of their project.

References

"Battery Charge Time Calculator". http://www.csgnetwork.com/batterychgcalc.html

"Battery Life Calculator". https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life.

Carryer, J.E., et al. (2012). *Introduction to Mechatronic Design*.

Evans, Tom. "How Green is Silicon Valley? Ecological Sustainability and the High-Tech Industry." Berkeley Planning Journal 17.1 (2004) ProQuest. 7 Dec. 2018.

Wang S., Shi C., Fang C., Feng K. "Examining the spatial variations of determinants of energy-related CO2 emissions in China at the city level using Geographically Weighted Regression Model" Elsevier Journal of Cleaner Production. October 2018.

Appendices

***Appendix 1**: Detail Drawings*

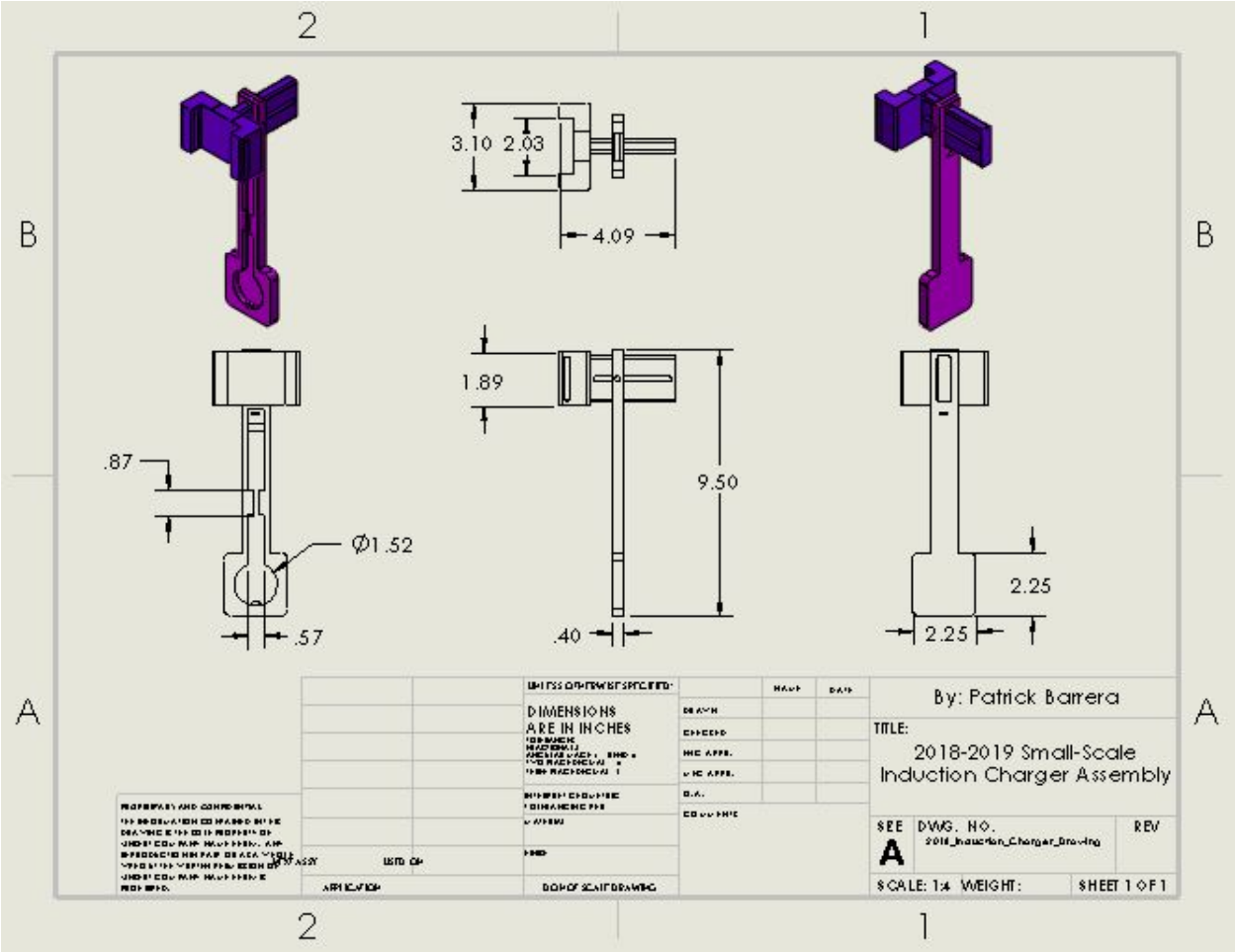


Figure 23. SolidWorks 2D-detail drawing of induction charging hub

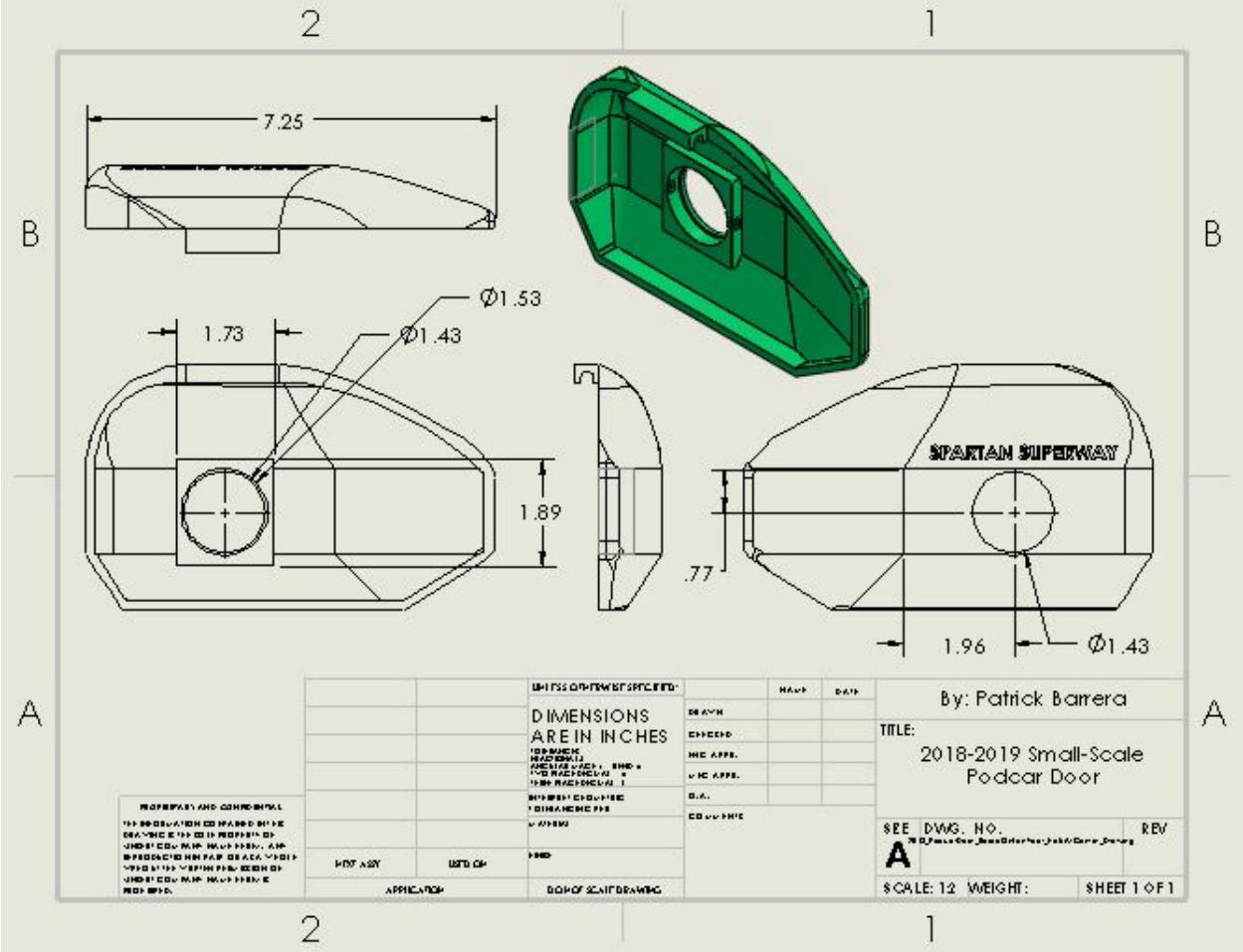


Figure 24. SolidWorks 2D-detail drawing of new pod car door

***Appendix 2**: GUI code (updated code on the raspi tablet)*

```
from tkinter import *
from xbee import XBee
import serial
from PIL import Image, ImageTk

ser = serial.Serial("COM8", 9600)
xbee = XBee(ser)

instructions = [0, 0]

def Confirm_selection(*args):
  print('Station Chosen:', options_var.get())

  if options_var.get() == 'Station 01':
      instructions[0] = "01"
      return instructions
  elif options_var.get() == 'Station 02':
      instructions[0] = "02"
      return instructions
  elif options_var.get() == 'Station 03':
      instructions[0] = "03"
      return instructions
  elif options_var.get() == 'Station 04':
      instructions[0] = "04"
      return instructions
  elif options_var.get() == 'Station 05':
      instructions[0] = "05"
      return instructions
  elif options_var.get() == 'Station 06':
      instructions[0] = "06"
      return instructions
  elif options_var.get() == 'Station 07':
      instructions[0] = "07"
      return instructions
  elif options_var.get() == 'Station 08':
      instructions[0] = "08"
      return instructions
  elif options_var.get() == 'Station 09':
      instructions[0] = "09"
      return instructions
  elif options_var.get() == 'Station 10':
      instructions[0] = "10"
      return instructions
  elif options_var.get() == 'Station 11':
      instructions[0] = "11"
      return instructions
  elif options_var.get() == 'Station 12':
      instructions[0] = "12"
```

```
        return instructions
    elif options_var.get() == 'Station 13':
        instructions[0] = "13"
        return instructions
    elif options_var.get() == 'Station 14':
        instructions[0] = "14"
        return instructions

def pod1():
    instructions[1] = "01"
    return instructions

def pod2():
    instructions[1] = "02"
    return instructions

def pod3():
    instructions[1] = "03"
    return instructionsz

def pod_req():
    xbee.tx(dest_addr='\x00\x01', data=instructions[0])
    xbee.tx(dest_addr='\x00\x01', data=instructions[1])
    xbee.tx(dest_addr='\x00\x01', data=instructions[2])
    print(instructions[0])
    print(instructions[1])
    print(instructions[2])

"""Everything for the GUI"""

root = Tk()
width = root.winfo_screenwidth()
height = root.winfo_screenheight()
screen_geometry = "%dx%d" % (width, height)
root.geometry(screen_geometry)
root.resizable(0, 0)

# Window Title
root.title("SPARTAN Superway Pod Network")

station_label = Label(root, text="     Pick the station you would like to visit      ", bg="blue", fg="white")
station_label.grid(row=4, column=0, columnspan=3)

podcar_label = Label(root, text="     Pick a podcar you would like to use      ", bg="blue", fg="white")
podcar_label.grid(row=8, column=0, columnspan=3)

image1 = ImageTk.PhotoImage(Image.open("267.jpg"))
panel1 = Label(root, image=image1)
panel1.grid(row=1, column=0, columnspan=3)
```

```
image2 = ImageTk.PhotoImage(Image.open("642.jpg"))
panel2 = Label(root, image=image2)
panel2.grid(row=16, column=0, columnspan=3)

spacing0 = Label(root)
spacing0.grid(row=0, column=0, columnspan=3)
spacing1 = Label(root)
spacing1.grid(row=3, column=0, columnspan=3)
spacing2 = Label(root)
spacing2.grid(row=7, column=0, columnspan=3)
spacing4 = Label(root)
spacing4.grid(row=6, column=0, columnspan=3)
spacing1 = Label(root)
spacing1.grid(row=11, column=0, columnspan=3)
spacing3 = Label(root)
spacing3.grid(row=10, column=0, columnspan=3)
spacing9 = Label(root)
spacing9.grid(row=99, column=0, columnspan=3)

OPTIONS = ["Station 01", "Station 02", "Station 03", "Station 04", "Station 05", "Station 06", "Station 07",
        "Station 08", "Station 09", "Station 10", "Station 11", "Station 12", "Station 13", "Station 14"]

OPTIONS.sort()
options_var = StringVar(root)
options_var.set('Choose A Station')

popupMenu = OptionMenu(root, options_var, *OPTIONS, command=Confirm_selection)
popupMenu.grid(row = 5, column = 1)

podcar1 = Button(root, text="      Podcar 1      ", fg="green", command=pod1)
podcar2 = Button(root, text="      Podcar 2      ", fg="green", command=pod2)
podcar3 = Button(root, text="      Podcar 3      ", fg="green", command=pod3)

podcar1.grid(row=9, column=0)
podcar2.grid(row=9, column=1)
podcar3.grid(row=9, column=2)

request_pod = Button(root, text="    Request Pod    ", fg="black", command=pod_req)
request_pod.grid(row=11, column=0, columnspan=3)

root.mainloop()
```

Appendix 3: *Arduino code*

```
/*                            PIN LAYOUT
 * ---------------------------------------------------------------------------------------
 *           MFRC522      Arduino       Arduino   Arduino    Arduino          Arduino
 *           Reader/PCD   Uno/101       Mega      Nano v3    Leonardo/Micro   Pro Micro
 * Signal    Pin          Pin           Pin       Pin        Pin              Pin
 * ---------------------------------------------------------------------------------------
 * RST/Reset   RST         9             5         D9         RESET/ICSP-5     RST
 * SPI SS      SDA(SS)     10            53        D10        10               10
 * SPI MOSI    MOSI        11 / ICSP-4   51        D11        ICSP-4           16
 * SPI MISO    MISO        12 / ICSP-1   50        D12        ICSP-1           14
 * SPI SCK     SCK         13 / ICSP-3   52        D13        ICSP-3           15
 *
 * DIN = pin D18
 * DOUT = pin D19
 */

//#include <SoftwareSerial.h>

#include <MFRC522.h>
#include <Servo.h>

#define StopButton 46
#define StartButton 47
#define R_LED 4
#define G_LED 6
#define B_LED 7
#define trigPin 48
#define echoPin 49
#define SDA_pin 53
#define RST_pin 5
MFRC522 rfid(SDA_pin, RST_pin);      //Instance  the class
MFRC522::MIFARE_Key key;

byte tagID[4];
byte location = 0;

char pick_up[2];
char char_destination[2];
char pod_number[2];

int nothing = 0;
int destination = 0;
int pickup=0;
int pod_num=0;
```

```
int a= 0;
int b= 0;
int c= 0;
int d= 0;

long duration = 0;
int distance = 0;

int motor =3;
Servo servo;


// define the connections to motor driver inputs
#define IN_1 10
#define IN_2 11
#define IN_3 12
#define IN_4 13

int CW_coilA [4] = {IN_4, IN_3, IN_2, IN_1};
int CW_coilB [4] = {IN_3, IN_2, IN_1, IN_4};

int CCW_coilB [4] = {IN_1, IN_2, IN_3, IN_4};
int CCW_coilA [4] = {IN_4, IN_1, IN_2, IN_3};

/* Delay of 6000 microseconds = 0.006 sec */
const int minDelay = 1600;

/* (III) Function Prototypes */
void steps_hitorque(int mySteps);

//SoftwareSerial XBee(2, 3); // RX, TX
//SoftwareSerial XBee(10, 11); // RX, TX

void setup()
{
  // setup the pins to motor driver as outputs
  pinMode(IN_1, OUTPUT);
  pinMode(IN_2, OUTPUT);
  pinMode(IN_3, OUTPUT);
  pinMode(IN_4, OUTPUT);

  // Set up both ports at 9600 baud. This value is most important
  // for the XBee. Make sure the baud rate matches the config
  // setting of your XBee.
  //XBee.begin(9600);
  Serial.begin(9600);
  Serial1.begin(9600);
  servo.attach(9);
```

```
 // Serial Communicatioon begins
 SPI.begin();      // Initiate  SPI bus
 rfid.PCD_Init();
 Serial.println("Scan UID card");

 for (byte i=0; i<6; i++)
 {
  key.keyByte[i] = 0xFF;
 }

 Serial.println(F("Initialization complete"));
}

void loop()
{
//while (a < 2)
 //{
  //Serial1.begin(9600);
  if(Serial1.available())
  {
   //char received = Serial1.read();
   CheckButtons();
   //CheckDist();
   GreenLight();
   char received3 = Serial1.read();
   if (a < 2 )
   {
   pick_up[a++] = received3;
   //pick_up[a] = received;
   //a++;
  // a = 0;
   //c = 0;
  }

  else
  {
   pick_up[a]=0;
   pickup = atoi(pick_up);
   //a=0;
  }
 }

//while (b < 2)
 //{
    CheckButtons();
    //CheckDist();
    GreenLight();
```

```
  if(Serial1.available())
  {
    char received1 = Serial1.read();
    if (b < 2 )
    {
    char_destination[b++] = received1;
    //b++;
   // c = 0;
  }
  else
  {
   char_destination[b]=0;
   destination = atoi(char_destination);
  // b=0;

  }
}

 while (c < 2)
{
  CheckButtons();
  CheckDist();
  if (Serial1.available())
  {
    char received = Serial1.read();
    pod_number[c] = received;
    c++;
  }
}
if(d == 0)
{
  //digitalWrite(light,HIGH);
  Serial.print("Your pick up Station is: \n");
  pickup = atol(pick_up);
  Serial.println(pickup);
  Serial.print("The destination is Station: \n");
  destination = atol(char_destination);
  Serial.println(destination);
  Serial.print("Requested pod: \n ");
  pod_num = atoi(pod_number);
  Serial.println(pod_num);


////////////////////////////////// Add the pod number of a specific pod in the if statement ////////////////////////

  if(pod_num == 1|| pod_num == 2|| pod_num == 3 )
  {
```

```
  Serial.print("Your pick up Station is: ");
  Serial.println(pickup);
  Serial.print("The destination is Station: ");
  Serial.println(destination);
  Serial.print("Requested pod: ");
  Serial.println(pod_num);
   d++;
  }

  else
  {
    d=0;
    a=0;
    c=0;
  }
  CheckDist();
  CheckButtons();
}


                                    // RFID CODE
// Look for new cards
 if ( ! rfid.PICC_IsNewCardPresent())
 {
  return;
 }
 // Select one of the cards
 if ( ! rfid.PICC_ReadCardSerial())
 {
  return;
  rfid.PICC_DumpToSerial(&(rfid.uid));
 }

 if (rfid.uid.uidByte[0] != tagID[0] ||        //Only reads a card once
    rfid.uid.uidByte[1] != tagID[1] ||
    rfid.uid.uidByte[2] != tagID[2] ||
    rfid.uid.uidByte[3] != tagID[3])
    {

      for (byte i=0; i<4; i++)
      {
        tagID[i] = rfid.uid.uidByte[i];
      }

      nothing = 0;
    }
 else
 {
```

```
  nothing = 1;
 }

 rfid.PICC_HaltA();
 rfid.PCD_StopCrypto1();

 ////////////////////////////////DETERMINE LOCATION OF BOGIE////////////////////////////

if (tagID[0] == 179 && nothing != 1)
{
  location = 1;
  //return location;
}

else if (tagID[0] == 19 && nothing != 1)
{
  location = 2;
 // return location;
}

else if (tagID[0] == 35 && nothing != 1)
{
  location = 3;
  //return location;
}

else if (tagID[0] == 99 && nothing != 1)
{
  location = 4;
  //return location;
}

else if (tagID[0] == 51 && nothing != 1)
{
  location = 5;
  //return location;
}

else if (tagID[0] == 163 && nothing != 1)
{
  location = 6;
  //return location;
}

else if (tagID[0] == 227 && nothing != 1)
{
  location = 7;
 // return location;
```

```
}

else if (tagID[0] == 211 && nothing != 1)
{
  location = 8;
  //return location;
}

else if (tagID[0] == 83 && nothing != 1)
{
  location = 9;
  //return location;
}

else if (tagID[0] == 67 && nothing != 1)
{
  location = 10;
  //return location;
}

else if (tagID[0] == 84 && nothing != 1)
{
  location = 11;
 // return location;
}

else if (tagID[0] == 36 && nothing != 1)
{
  location = 12;
  //return location;
}

else if (tagID[0] == 116 && nothing != 1)
{
  location = 13;
  //return location;
}

else if (tagID[0] == 244 && nothing != 1)
{
  location = 14;
  //return location;
}

else
{
  location = 0;
  //return location;
```

```
}


switch (location)
{
 case 0:
   location = 0;
 break;

 case 1:
   Serial.print("\nLocation 1 reached");
   if (pickup != location)
   {
     MoveForward();
   }
   else if (pickup == location)
   {
     pickup = Stop(0);
     Dropoff();
   }
 break;

 case 2:
   Serial.print("\nLocation 2 reached");
   if (pickup != location)
   {
     MoveForward();
   }
   else if (pickup == location)
   {
     pickup = Stop(0);
     Dropoff();
   }
 break;

 case 3:
   Serial.print("\nLocation 3 reached");
   if (pickup != location)
   {
     TurnRight();
   }
   else if (pickup == location)
   {
     pickup = Stop(0);
     Dropoff();
   }
 break;
```

```
case 4:
  Serial.print("\nLocation 4 reached");
  if (pickup != location)
  {
    MoveForward();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 5:
  Serial.print("\nLocation 5 reached");
  if (pickup != location)
  {
    TurnRight();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 6:
  Serial.print("\nLocation 6 reached");
  if (pickup != location)
  {
    MoveForward();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 7:
  Serial.print("\nLocation 7 reached");
  if (pickup != location)
  {
    MoveForward();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
```

```
    Dropoff();
  }
break;

case 8:
  Serial.print("\nLocation 8 reached");
  if (pickup == 9 || pickup == 10 || pickup == 11)
  {
    TurnRight();
  }
  else if (pickup == 12 || pickup == 13 || pickup == 14)
  {
    TurnLeft();
  }
   else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 9:
  Serial.print("\nLocation 9 reached");
  if (pickup != location)
  {
    MoveForward();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 10:
  Serial.print("\nLocation 10 reached");
  if (pickup != location)
  {
    MoveForward();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 11:
```

```
  Serial.print("\nLocation 11 reached");
  if (pickup != location)
  {
    TurnRight();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 12:
  Serial.print("\nLocation 12 reached");
  if (pickup != location)
  {
    MoveForward();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 13:
  Serial.print("\nLocation 13 reached");
  if (pickup != location)
  {
    MoveForward();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
  }
break;

case 14:
  Serial.print("\nLocation 14 reached");
  if (pickup != location)
  {
    TurnRight();
  }
  else if (pickup == location)
  {
    pickup = Stop(0);
    Dropoff();
```

```
    }
   break;

   default:
    Serial.print("\nCard is not catalogged");
   break;
  }
}

void Dropoff()
{
 switch (location)
 {
  case 0:
   location = 0;
  break;

  case 1:
   Serial.print("\nLocation 1 reached");
   if (destination != location)
   {
    MoveForward();
   }
   else if (destination = location)
   {
    destination = Stop(0);
   }
  break;

  case 2:
   Serial.print("\nLocation 2 reached");
   if (destination != location)
   {
    MoveForward();
   }
   else if (destination == location)
   {
    destination = Stop(0);
   }
  break;

  case 3:
   Serial.print("\nLocation 3 reached");
   if (destination != location)
   {
    TurnRight();
   }
   else if (destination == location)
```

```
  {
   destination = Stop(0);
  }
break;

case 4:
  Serial.print("\nLocation 4 reached");
  if (destination != location)
  {
   MoveForward();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 5:
  Serial.print("\nLocation 5 reached");
  if (destination != location)
  {
   TurnRight();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 6:
  Serial.print("\nLocation 6 reached");
  if (destination != location)
  {
   MoveForward();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 7:
  Serial.print("\nLocation 7 reached");
  if (destination != location)
  {
   MoveForward();
  }
  else if (destination == location)
```

```
  {
   destination = Stop(0);
  }
break;

case 8:
  Serial.print("\nLocation 8 reached");
  if (destination == 9 || destination == 10 || destination == 11)
  {
   TurnRight();
  }
  else if (destination == 12 || destination == 13 || destination == 14)
  {
   TurnLeft();
  }
   else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 9:
  Serial.print("\nLocation 9 reached");
  if (destination != location)
  {
   MoveForward();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 10:
  Serial.print("\nLocation 10 reached");
  if (destination != location)
  {
   MoveForward();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 11:
  Serial.print("\nLocation 11 reached");
  if (destination != location)
```

```
  {
   TurnRight();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 12:
  Serial.print("\nLocation 12 reached");
  if (destination != location)
  {
   MoveForward();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 13:
  Serial.print("\nLocation 13 reached");
  if (destination != location)
  {
   MoveForward();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

case 14:
  Serial.print("\nLocation 14 reached");
  if (destination != location)
  {
   TurnRight();
  }
  else if (destination == location)
  {
   destination = Stop(0);
  }
break;

default:
  Serial.print("\nCard is not catalogged");
break;
```

```
  }
}

void MoveForward()
{
  Serial.println("\nMoving Forward\n");
  //Rotate motors until a new card is read
  //analogWrite(motor, 255);
  steps_hitorque(-20480);
  if ( ! rfid.PICC_IsNewCardPresent())
  return motor;
}

void TurnRight()
{
  Serial.println("\nTurning Right\n");
  //Rotate servo to the right to steer the bogie
  servo.write(255);
  //analogWrite(motor, 255);
  steps_hitorque(-2048);
  steps_hitorque(-2048);
  if ( ! rfid.PICC_IsNewCardPresent());
  return;
}

void TurnLeft()
{
  Serial.println("\nTurning Left\n");
  //Rotate servo to the left to steer the bogie
  servo.write(0);
  //analogWrite(motor, 255);
  steps_hitorque(-2048);
  steps_hitorque(-2048);
  if ( ! rfid.PICC_IsNewCardPresent());
  return;
}

int Stop(int destination)
{
  Serial.println("\nDestination Reached");
  Serial.println("Await new directions");
 // digitalWrite(light,LOW);
  //analogWrite(motor, 0);
  steps_hitorque(0);
  destination = 0;
  a = 0;
  b = 0;
  return destination;
```

```
 return a;
 return b;
 //Stop motor movement
 //Await new instructions
}

void printDec(byte *buffer, byte bufferSize)
{
 for (byte i = 0; i < bufferSize; i++) {
  Serial.print(buffer[i] < 0x10 ? " 0" : " ");
  Serial.print(buffer[i], DEC);
 }
}

void CheckButtons()
{
 if(digitalRead(StopButton) == LOW)
 {
  while(digitalRead(StartButton) == HIGH)
  {
   //analogWrite(motor, 0);
   steps_hitorque(0);
   digitalWrite(R_LED, LOW);
   digitalWrite(G_LED, LOW);
   digitalWrite(B_LED, LOW);
   Serial1.begin(9600);
  }
  //analogWrite(motor, 200);
  steps_hitorque(-20480);
  destination = 0;
  a = 0;
  b = 0;
  //return destination;
  //return a;
  //return b;

 }
}

void RedLight()
{
 digitalWrite(R_LED, HIGH);
 digitalWrite(G_LED, LOW);
 digitalWrite(B_LED, LOW);
}

void BlueLight()
{
```

```
  digitalWrite(B_LED, HIGH);
  digitalWrite(G_LED, LOW);
  digitalWrite(R_LED, LOW);
}

void GreenLight()
{
  digitalWrite(G_LED, HIGH);
  digitalWrite(R_LED, LOW);
  digitalWrite(B_LED, LOW);
}

void CheckDist()
{
  digitalWrite(trigPin, LOW);
  // delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration*0.034/2;
  if (distance < 20)
  {
    Serial.println("Too close to next pod car");
   // analogWrite(motor,0);
    steps_hitorque(0);
    delay(100);
   // analogWrite(motor, 200);
    steps_hitorque(-20480);
  }
}

/* (VI) Function Definitions */
void steps_hitorque(int stepsToMake)
{
  /* Current step number (used as a counter) */
  int mySteps = 0;

  /* If user inputted a pos number, turn the stepper CW */
  // int CW_coilA [4] = {IN_1, IN_2, IN_2, IN_1};
  // int CW_coilB [4] = {IN_3, IN_3, IN_4, IN_4};
  if(stepsToMake > 0)
  {
    while(mySteps <= stepsToMake)
    {
      for(int i = 0; i < 4; i++)
      {
        digitalWrite(CW_coilA[i], HIGH);
```

```
      digitalWrite(CW_coilB[i], HIGH);
      delayMicroseconds(minDelay);
      digitalWrite(CW_coilA[i], LOW);
      digitalWrite(CW_coilB[i], LOW);
      mySteps++;
    }
   }
}

/* If user inputted a neg number, turn the stepper CCW */
// int CCW_coilA [4] = {IN_1, IN_2, IN_2, IN_1};
// int CCW_coilB [4] = {IN_4, IN_4, IN_3, IN_3};
if(stepsToMake < 0)
{
  while(mySteps >= stepsToMake)
  {
    for(int j = 0; j < 4; j++)
    {
      digitalWrite(CCW_coilB[j], HIGH);
      digitalWrite(CCW_coilA[j], HIGH);
      delayMicroseconds(minDelay);
      digitalWrite(CCW_coilA[j], LOW);
      digitalWrite(CCW_coilB[j], LOW);
      mySteps--;
    }
  }
}
}
```

Appendix 4: *Motor calculation*

Gimbal Motor Velocity Calculations:

Known Parameters:

Radius of new Wheels: r = 35 mm

Gimbal Motor Voltage Constant: $K_v = 80\frac{rpm}{V}$

LiPo Battery Voltage: V = 7.4V

Additional LiPo Battery Characteristics:

Battery Capacity = 2600mAh (milliAmp Hour),

Cell Count = 2S (2 cells in Series)

Discharge Rating = 90C (Capacity in Amps),

Battery Energy = 19.2 Wh (Watt Hour)

Calculate circumference of new wheels using the radius:

$$C_{wheel} = 2\,\pi\,r$$
$$C_{wheel} = 2\times(\pi)\times(35\ mm)$$
$$C_{wheel} = 219.8\ mm$$

Calculate Gimbal Motor's angular velocity using the motor's voltage constant and the battery voltage:

$$\omega = \left(80\frac{rpm}{V}\right)\times(7.4\ V) = 592\ rpm$$

We want to keep the max velocity around 1 m/s, so we must use proper gearing. It seems a 2:1 gear ratio will help us achieve the desired speed. In other words, we want to halve the 592 rpm:

$$Gear\ Ratio = N = \frac{\omega_{in}}{\omega_{out}}$$
$$2 = \frac{592\ rpm}{\omega_{out}}$$
$$\omega_{out} = 296\ rpm$$

Convert the geared-down angular velocity to linear velocity:

$$v = \omega_{out}\,C_{wheel}$$
$$v = \left(296\frac{rev}{min}\right)\times(219.8\ mm)$$
$$v = 65060.8\frac{mm}{min}$$

Convert linear velocity from mm/min to m/s:

$$\frac{65060.8\ mm}{min}\times\frac{1\ m}{1000\ mm}\times\frac{1\ min}{60\ sec} = 1.084\frac{m}{s}$$

The desired velocity of 1.084 m/s has been achieved with the new bogie wheels, the new gimbal motor, the old 7.4V LiPo battery, and a 2:1 gearing set.

Source: Carryer, J.E, et. al. (2012) *Introduction to Mechatronic Design*. Pps. 536-541, 547-548.

Gimbal Motor Maximum Acceleration Calculation:

Ideally, we want the pod car to move at a constant speed throughout the rest of the trip. So, the only acceleration we have to consider is the start-up acceleration to speed up from rest to the

max velocity. The max velocity, given 7.4V, is 1.084 m/s. The minimum amount of time we can then say that we want to accelerate for is about 2 seconds. So, the acceleration would be:

$$a = \frac{v}{t} = \frac{1.084 \frac{m}{s}}{2\ sec} = 0.542 \frac{m}{s^2}$$

So, this would be the max acceleration if we want to accelerate to the max speed in 2 seconds.

Battery Life Calculation [7.4V, 2600 mAh LiPo]:

Known Parameters:

Gimbal Motor Voltage Constant = $K_b = 80 \frac{V}{krpm}$

Gimbal Motor Output Torque = $T = 32.62\ oz \bullet in$

LiPo Battery Capacity = $2600\ mAh$

Additional LiPo Battery Characteristics: (7.4 V, 2S, 90C, 19.2 Wh)

Convert voltage constant (K_b) to torque constant (K_t):

$$K_t \left[\frac{oz \cdot in}{A}\right] = 1.3542 \times K_b \left[\frac{V}{krpm}\right]$$
$$K_t = 1.3542 \times 80 \frac{V}{krpm}$$
$$K_t = 108.336 \frac{oz \cdot in}{A}$$

Use motor output torque-current relation to obtain load current:

$$T = (Kt) \times (i)$$
$$(32.62\ oz \cdot in) = \left(108.336 \frac{oz \cdot in}{A}\right) \times i$$
$$i = 0.301\ A$$
$$i = 301\ mA$$

Battery Life (the 0.7, below, makes allowances for external factors which can affect battery life):

$$t = \frac{Battery\ Capacity}{Load\ Current} \times 0.70$$
$$t = \frac{2600\ mAh}{301\ mA} \times 0.70$$
$$t = 6.0\ hours$$

With a torque output of $32.62\ oz \cdot in$, the current draw of the motor is 301 mA. With this current draw, the 7.4 LiPo battery will last 6 hours. Source: Battery Life Calculator

Battery Charge Time Calculation [7.4V, 2600mAh LiPo]:

The calculation for the amount of time to charge the 7.4V, 2600mAh LiPo battery utilizes the same equation as the discharge/battery life equation, with just a different load current (in this case, charge current, which is provided by the induction charger) and a different percentage multiplier.

Given that the charge current is 600 mA and using the normally accepted standard for efficiency loss of 20% (Source: Battery Charge Time Calculator), the charge time is:

$$t = \frac{Battery\ Capacity}{Charge\ Current} \times 1.20\ Efficiency\ Lost$$
$$t = \frac{2600\ mAh}{600\ mA} \times 1.20$$

$$t = 5.2\ hours$$

However, since we will be using a current amplifier to amplify the current from the induction charger to (hopefully) 6 A (6 A = 6000 mA), the charge time becomes:

$$t = \frac{Battery\ Capacity}{Charge\ Current} \times 1.20\ Efficiency\ Lost$$
$$t = \frac{2600\ mAh}{6000\ mA} \times 1.20$$
$$t = 0.52\ hours$$
$$t = 32\ minutes$$

Battery Charge Time Calculation [3.7V, 6600 mAh LiPo]:

However, we did mention, in our Presentation #3, that one of our contingency plans (specifically, Plan B.2) is to use a 3.7V, 6600 mAh LiPo battery since mini LiPo battery chargers for 7.4V don't seem to exist. Therefore, the amount of time to charge the 3.7V, 6600mAh LiPo battery is calculated below. Again, given that the charge current is 600 mA and using the normally accepted standard for efficiency loss of 20% (Source: Battery Charge Time Calculator), the charge time is:

$$t = \frac{Battery\ Capacity}{Charge\ Current} \times 1.20\ Efficiency\ Lost$$
$$t = \frac{6600\ mAh}{600\ mA} \times 1.20$$
$$t = 13.2\ hours$$

However, since we will be using a current amplifier to amplify the current from the induction charger to (hopefully) 6 A (6 A = 6000 mA), the charge time becomes:

$$t = \frac{Battery\ Capacity}{Charge\ Current} \times 1.20\ Efficiency\ Lost$$
$$t = \frac{6600\ mAh}{6000\ mA} \times 1.20$$
$$t = 1.3\ hours$$

Source: Battery Charge Time Calculator

While it will take longer to charge the 3.7V, 6600 mAh LiPo battery, it will be able to last longer than the currently used 7.4V, 2600 mAh LiPo battery thanks to its larger battery capacity of 6600 mAh. Therefore, the updated battery life calculation for the 3.7V, 6600 mAh LiPo battery is below:

Battery Life Calculation [3.7V, 6600mAh LiPo]:

Known Parameters:

Gimbal Motor Voltage Constant = $K_b = 80\frac{V}{krpm}$

Gimbal Motor Output Torque = $T = 32.62\ oz \bullet in$

LiPo Battery Capacity = $6600\ mAh$

Additional LiPo Battery Characteristics: (3.7V, 3S, 1.5C)

Convert voltage constant (K_b) to torque constant (K_t):

$$K_t\ \left[\frac{oz \cdot in}{A}\right] = 1.3542 \times K_b\ \left[\frac{V}{krpm}\right]$$
$$K_t = 1.3542 \times 80\frac{V}{krpm}$$

$$K_t = 108.336 \frac{oz \cdot in}{A}$$

Use motor output torque-current relation to obtain load current:

$$T = (Kt) \times (i)$$
$$(32.62\ oz \cdot in) = \left(108.336 \frac{oz \cdot in}{A}\right) \times i$$
$$i = 0.301\ A$$
$$i = 301\ mA$$

Battery Life (the 0.7, below, makes allowances for external factors which can affect battery life):

$$t = \frac{Battery\ Capacity}{Load\ Current} \times 0.70$$
$$t = \frac{6600\ mAh}{301\ mA} \times 0.70$$
$$t = 15.3\ hours$$

With a torque output of $32.62\ oz \cdot in$, the current draw of the motor is 301 mA. With this current draw, the 3.7 LiPo battery will last 15.3 hours. Source: Battery Life Calculator

Appendix 5: *Key Data Sheets and Product Information*

- 12V 600mA Wireless Charging Module
  https://www.robotshop.com/en/12v-600ma-wireless-charging-module.html

- Erayco 10 Pack LM2596 DC to DC Buck Converter 3.0-40V to 1.5-35V Power Supply Step Down Module
  https://www.amazon.com/gp/product/B07JNQFV7F/ref=ppx_yo_dt_b_asin_title_o01_s01?ie=UTF8&psc=1

- Makerfocus 10pcs TP4056 Charging Module 5V Micro USB 1A 18650 Lithium Battery Charging Board with Protection Charger Module
  https://www.amazon.com/gp/product/B071RG4YWM/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1

- Anmbest 5PCS MT3608 Step-Up Adjustable DC-DC Switching Boost Converter Power Supply Module 2-24V to 5V-28V 2A
  https://www.amazon.com/gp/product/B07FYXSWJ8/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1