College of Engineering
TEMPLE UNIVERSITY®

Benchtop Testbed for Autonomous Vehicle Research for the Dissipation of Stop-and-Go Traffic Waves

a senior design project documentation report

December 9, 2019

Team 18, T.U. Autonomy

Chad Martin

Von Kaukeano

Nissan Gelb

Philip Cameron

Advisor: Dr. Philip Dames

Coordinator: Dr. Brian Thomson

Abstract

Large scale results start from small scale experiments. It has been proven that an automated vehicle is able to dampen stop-and-go waves that can arise even in the absence of geometric or lane changing triggers†. This paper documents the methods of how to scale down the test bed that allows for researchers to home in on effective control programs. This inexpensive design combined with realistic kinematics creates a pathway to pair multiple patterns of human driving with predetermined algorithms on an automated car all the while collecting metrics per each vehicle. This project combines on-the-road research and computer simulations to pursue the portion of the larger goal, of a cyber-physical system, which is to find "new control algorithms for traffic flow regulation" (NSF, 1446690) (Stern et al. 2017)

Table of Contents

Table of Figures

Tables

Design Overview

Project Context

Autonomous vehicles are currently being tested on some public roadways throughout the nation. These tests include mobility, traffic law obedience, safety to pedestrians and infrastructure, and traffic control (NHTSA). Dr. Seibold's research of Temple University's Math Department involves traffic flow modeling, particularly of "phantom" traffic jams or "jamitons" (Science Daily). A "phantom" traffic jam is a small congestion in vehicular traffic that occurs spontaneously, in the absence of bottlenecks, obstacles, or any discernible causes on the road. These jamitons enforce unexpected braking maneuvers, and thus impose stress on drivers and materials, waste fuel and increase pollution, and are hot spots for potential vehicle collisions (NSF, 1007899). He defines autonomous vehicles as those that conduct their velocity control -- car-following, acceleration, and deceleration -- in an automated fashion. On roadways, in the far-future, vehicles will be self-driven and connected. In the near future, a small percentage of vehicles will be autonomous or have adaptive cruise control and it can be leveraged to control the whole, human-driver-dominated, flow into a more safe and efficient uniform flow regime. (Seibold)

Figure 1: Full size test on a circular, one-lane, endless highway.

This senior project intends to assist in the traffic control research. Previously, researchers assembled a fleet of vehicles for two specific reasons; first to demonstrate the emergence of single lane phantom waves, and secondly, to demonstrate that these phantom waves can be dampened by one vehicle following specific driving controls. Prior to the large scale demonstration, researchers gathered enough evidence from traffic observations and comparable traffic computer simulations and presented the findings to the National Science Foundation, NSF. The group received a grant for further study.

Autonomous vehicles are being tested on closed and public roadways which is necessary for hardware and software validation, but it is difficult, expensive, and dangerous. Simulations tend to solve any issue ideally. This project bridges the gap between the two and allow researchers to test algorithms on hardware and examine the effects on the individual vehicle and the whole traffic level.

A previous design team pursued this goal and reached some checkpoints. The team built three test vehicles on a powered track with a notched path to follow. They used a Remote Operating System on a Raspberry Pi. The shortcomings are: 1) too expensive at about $180 per car, 2) scale was too large, for track and vehicle, 3) lacking documentation for repeatability and fabrication, and 4) a

dangerous, inconsistent power source. This current prototype is fully documented, less expensive, down to scale, more versatile, and safer.

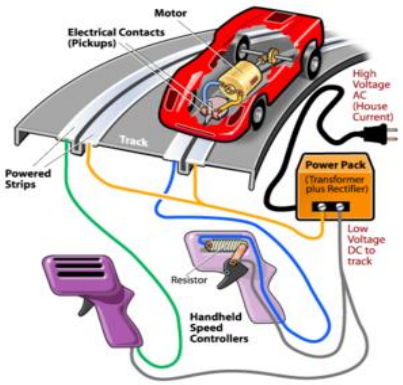


Figure 2: Comic of Previous Team's design

Product Concept and Components

The concept is simple: build a car that can behave like a human on its own. The car, which is part of a "testbed," is boiled down to its most fundamental parts. In order for the prototype to have the abilities of an autonomous vehicle, four components are required: motor controller, distance sensor, path sensor, and onboard computer. To be considered a vehicle, two components are necessary: wheels and a chassis. Added to which, the car needs a capable power supply in order to perform specific amounts of work in a specific time frame. The end product will be used for its behavior controls and metrics. Therefore, the required minimum of a testbed is its movement, sight, and computing. The "computing" piece is the heart of the product.

For mobility, the car needs to carry its weight while moving about. It has a chassis that supports a tripod-like arrangement under its belly with two wheels and a fixed castor wheel for balance. The tires have grip and add to the radius of the wheel. The motors are fixed to the chassis, one motor for each wheel. The gearbox that links the motor shaft to the wheel hub has a ratio of 120 to 1.

For vision, the testbed needs to see the road and see the testbed ahead of it. The leading testbed is detected by means of echolocation. An ultrasonic sensor is mounted to the front of the vehicle. As for the lane management, two infrared (IR) sensors detect a color contrast to follow on the road underneath the front of the vehicle.

For control, there is the Raspberry Pi that sends out commands and reads input to execute commands based on the programmed algorithm (like adaptive cruise control). The Pi talks to the motor controller, the ultrasonic sensor, and infrared track sensor. The Pi listens to the motors and the sensors. After listening to the equipment, the Pi then executes a command. These commands are input and predefined by the end-user/researcher/operator.

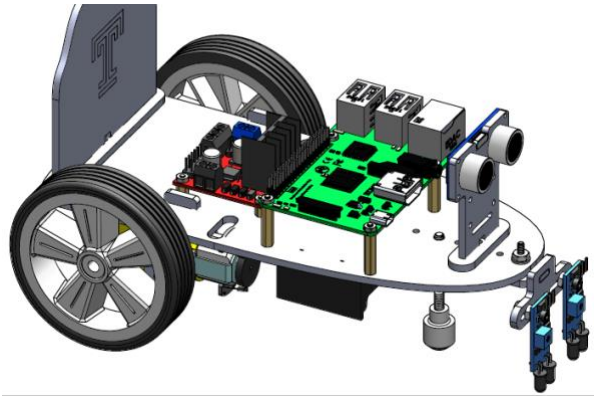


Figure 3: Prototype Version 2

Requirements and Constraints

Design Constraints

The key words to remember are “benchtop testbed.” Quantitatively, the cars and track need to fit in a relatively small space. The constraint is negotiable to have a track diameter of 8 feet which calls for a vehicle size less than 5% of the average vehicle length (RFP). The average vehicle size is about 14 to 15 feet (4.5 meters) (Mechanic Base), making 5% to be 8 to 9 inches. All other components of the car, besides kinematics, do not need scaling down.

Expense is a motivator for this project. The total value of the materials for a single car should cost less than $100 to fabricate (RFP). Each component was selected based on its anticipated expectations versus its cost. The quantity of cars to be built helped with bulk costs to make a more appealing bottom line price. A previous solution cost $180 and this project demanded less than $100.

Considering that the creator will not be the user, safety for this product is another top priority. To consider safety for the hardwiring portion of the system, NFPA70 2017 was referenced. Due to the low current levels conductor sizing charts such as NFPA70 Table 310.15(B)(16) were not applicable. This table does reference Article 240 which talks about the closes current level to our system. This is wiring of 18AWG because the system does not exceed 5.6 amperes. Additionally, lists class CC, J, and T as acceptable. Due to the use of an L298N Dual H-Bridge Motor Driver with an onboard +5VDC regulator with protection diodes, there was not a reason to use fuses for over current protection. The conductors used for all I/O was 28AWG which can carry about 0.25AMP for power transmission according to PowerStream. This is sufficient to the maximum current provided by the Raspberry Pi’s GPIO being 16mA. The wires off of the Battery pack were 24AWG which is capable of a sustained voltage of about 0.75A which is well above the operational current of the Motor controller which is 36mA

Realistic motion can be defined as the way people drive and how they expect other vehicles to move. This constraint is both quality and quantity. It needs to be elegant enough and hit specific parameters as well.

In further terms of quality for the project, the cars need to be durable, versatile, and documented well. Under the Request for Proposal, it stated that there will be researchers that will test out their algorithms on the project's hardware. This implies a regular handling of the cars and so a ruggedness to them is of demand. The versatility in the coding must be flexible enough for any desirable algorithm.

This document serves the purpose of dissemination to interested onlookers, to project advancement, and expected users.

Functional Requirements

Objective: *A set of cars that can be programmed individually to simulate traffic on a roadway.*

The functional requirements naturally arose from the necessity of what was described earlier under Project Concept. The requirements are based on realistic vehicle behaviors and user-defined controls. Each of these requirements breakdown to multiple systems working together, more specifically the vehicle behaviors. Each of these requirements need to be met by means of multiple systems working together.

The kinematics are scaled with the car. Since the on-the-road cars reached speeds of between 10 to 20 mph, about 15 to 30ft/s, then the cars need to run a top speed of 1.5 ft/s. The stopping distance should compare to a car. This deceleration must be programmed in because the stopping distance of a car is naturally proportional to the *square* of speed value. All motion must scale compared to full size car, not necessarily the forces.

The second requirement is to detect the distance to the leading vehicle. This is a standard feature on many 2019 vehicles (Cars.com). The distance to the leading car will help the driver/computer determine the appropriate target speed.

The third requirement is another standard on 2019 cars -- keeping a lane (Cars.com). The cars need to keep to a path and follow it in order to stay in one lane traffic.

The fourth requirement is for the end-user to place commands in each car remotely. This is a test bed for research. The team needs input controls, behaviors, and algorithms for each car to behave

as they wish. Most will need to act like common human drivers, a few will act as control or any combination imaginable. The team needed to anticipate scenarios and keep the software flexible.

Design Approach

Chassis and Fittings

The car needed to be small and fabricated with minimal cost. It will not be subjected to (sizeable) forces and it will hold its own weight. Using cheaper and lightweight materials paired with an efficient fabrication process was the goal. These cars will be mass produced as a fleet of 20-25. Plastics are the most appropriate in this case. Each and every part of the frame can be laser cut with a high precision. Additionally, in the prototype phase, having movable support fittings for component placement helped make the decision for plastics.

Motor and Motor Controller

These cars are being propelled by two motors. The calculations that appear in Appendix B were proved that inexpensive hobby motors can sufficiently power the cars. The real cars that are used for a scaled down version had a 0-60 mph velocity range; in actuality, the cars in the research had a maximum velocity of 25 mph (~12 m/s). The scale factor goal is 1:20, and so that means that the car must be able to reach 0.2 mph or 1 m/s.

Battery Selection

A challenge for any powered operation is a suitable energy supply. There are multiple options of batteries, and size and capability are considered. The components need variable electric potentials. Initially, a 12-Volt battery would suffice, but what batteries supply this. AA batteries supply 1.5 Volts and eight would have to fit on a car. Among a selection of batteries, lipo battery pack, lithium ion battery pack, and 18650 batteries, the smaller dimensions of the 18650 batteries were ideal. Each battery is 3.7 Volts and so 2-4 batteries would complete the task. It was the least expensive.

Sensory System

The eyes on the road and the obstacles is a challenging subject. The two most important factors for this particular project was cost and processing power. The options considered were a Pi Camera, Ultrasonic Distance Sensor, and Infrared Line Sensors (Appendix C). The ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. It works by sending out a sound wave at a frequency above the range of human hearing. The signal is reflected back and the time is measured from sent a received to acquire distance.

$$\text{Distance} = \frac{1}{2} * (\text{speed}\textit{Equation 1}\text{ of sound}) * (\text{time to receive echo}) \qquad \text{Equation 1}$$

The Ultrasonic Sensor-HC-SR04 was decided as the component to use to detect distance between two testbeds (Appendix D). The sensor provided 2cm to 400cm of measurement functionality. The sensor comes with a 5V DC power supply, 15mA working current, resolution of ~.3m, and a measuring angle of 30 degrees. It is a very inexpensive piece of equipment that can detect distance and change the speed of the motor as needed as well as computationally inexpensive. The Pi-cam was a different option that was debated, but after further investigation of computation, learning curve, and only needing a linear distance, then the ultrasonic would be best to add to the system. Supporting The infrared line sensor would cost effectively monitor the driving lane.

Central Processing Unit

The main processing components were between Raspberry Pi and Arduino. The Arduino Uno is equipped with sets of I/O pins that can be interfaced with many expansion boards and other circuits. The board has 14 digital I/O with 6 capable of PWM output, 6 analog I/O, and programmable in the Arduino IDE. The Raspberry Pi 3 has a 4xARM Cortex-A53 CPU, storage on a microSD card, 40-pin header GPIO, and can run on the Linux operating system. The biggest factor in deciding between the two were that the Raspberry Pi has a storage system that can have files saved and processed in the future. This is ideal for researchers as they will be able to run the simulations needed and save the data from a text file on a flash drive and process it however they decide is best. The Raspberry Pi 3 was decided, but there are many different models that could have been selected. There is also the comparison between the Raspberry Pi 3 and the Raspberry Pi Zero. The Raspberry Pi Zero specs consist of a 1GHz single core ARMv6 CPU, HAT-compatible 40-pin header, and storage on a microSD card. The decision between the Pi Zero and the Pi 3B was not only that the 3B has 1GB of RAM versus the 512MB of RAM, but there are no USB-C input to collect files to be processed. Ease of access of the information was a vital piece in the design of the system.

Operating System

The Raspberry Pi traditionally runs a Linux operating system, although Windows OS is common. A few desktop environment names would be NOOBS, Raspbian, Ubuntu Core, and Ubuntu MATE. Comparing each option, Raspbian was the better consideration as it is highly supported, and, based on Debian buster, that works well with the ROS environment. (Appendix C) After adding the image, it was quickly discovered that connecting to the school's enterprise wifi was a big issue. It was found that a file in the systems operating system had to be changed to "backdoor" the connection. The file was the wpa_supplicant located above the home directory of the Pi (~/../../etc/wpa_supplicant/wpa_supplicant.conf). There was then added network configurations to allow a connection and begin installing packages. [See Appendix C]

Furthermore, in attempting to secure shell (SSH) into the Raspberry Pi, access was denied. The goal was to have control over the system through a personal, or research, computer and be able to run experiments and access data. Ubuntu became the best choice for operating system selection and choosing to run the system on a local access network, LAN. It would allow the Temple University researcher to control all testbed systems connected to the LAN.

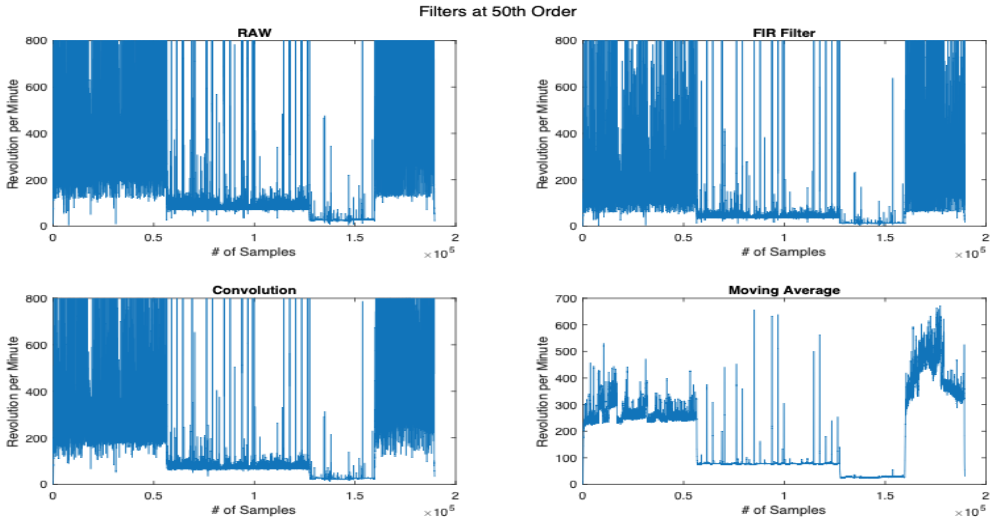
Data Filtering

Data filtering is critical to the engineering design of this system. In anticipation of noisy data coming from a hobbyist motor encoder, filtering methods were necessary to incorporate. The data that was received was taken and used to calculate revolutions per minute after taking the time interval between rising edges on the spinner.

The first filtering method tested was to use the filterDesigner function in Matlab because the RPM's were printed to a text file to be processed. It was proved that between 10% and 60% of the applied max voltage (11.8 volts) there was linearity of power relation in the motor; therefore at 10% duty cycle the cycles per second(Hz) were ~0.28 , and at 60% ~4 cycles per second. Setting the designer to be FIR and lowpass, the team let the designer build our low pass filter. The outcome was a 1648 order filter from the designer that the data was fed into using the filter function which takes input of data and filter and output of filtered data. The most important item to take from using the function was that the arrays had to be the same size. The output was clean and now have to succeed in Python. Using the same filter function from the scipy library, we would attempt to equate the outcome in Matlab, but the biggest issue was that again the arrays had to be of equivalent size. Having a 1648 order filter we had to have the same size of data before the function would run so we created a buffer of zeros. The RPM is calculated one point at a time and had to run to 1648 samples of data which caused a large delay in the output. The next idea was to use convolution on a triangle wave which takes a weighted average. Again using the data revolutions per minute, we used the convolve function in matlab and triang of approximately 300 order and ran into the same issue. Post processing the data these methods were ideal, but in real time we were not achieving the same. Finally, the choice was to use a moving average which was simple, identical in real time and post processing, did not have to worry about the filter gain, and less computationally expensive. With the other methods the computation was affecting the motor drastically and slowing it down the speeds at duty cycles we knew were running faster.

Along with the encoder data the decision would be applied to filter the ultrasonic sensor. The ultrasonic sensor can sense correctly, but sometimes the readings can be very large and cause the system to speed up when not necessary. It would be better to take the moving average every 5 samples to receive more accurate readings then to change the speed of the motor based on each individual measurement.

Table 1: Filtering Methods example



Programming

When deciding with what language to code, the comparison was among Python, C++, and using ROS. Both Python and C++ are general purpose languages and ROS is the robot operating system that allows for different code to be run simultaneously and share variables. We decided on using Python based on some major factors. Python is mostly used for scripting and C++ must be compiled; variables in Python do not have to be declared unlike C++; the team was familiar with libraries of Python such as numpy, pandas, and scipy, which helps with the signal processing.

The programming process began by writing modules for each component in the system. The modules can be broken down into four different parts. For example, motor control with pulse width modulation, IR sensor for track following, distance detection with the ultrasonic sensor, and encoder data collection from the motor. The best practice for writing software is to write each piece of code and secure the functionality of each component and eventually piecing together to become a system level program.

The ultrasonic sensor was tested first by researching the circuit and how the sensor works. The four pins are connected to the Pi and a signal is sent out from the sensor as a timer begins and then retrieved. The timer is then clocked and using the formula, the distance is calculated in centimeters.[See appendix C]. The distance is eventually filtered and then used to calculate the speed in revolutions per minute at the system level.

The programming for the motor controller was next on the task list. The motor controller has two enable pins (left/right), a PWM signal input, and power. A pin on the Raspberry Pi was selected

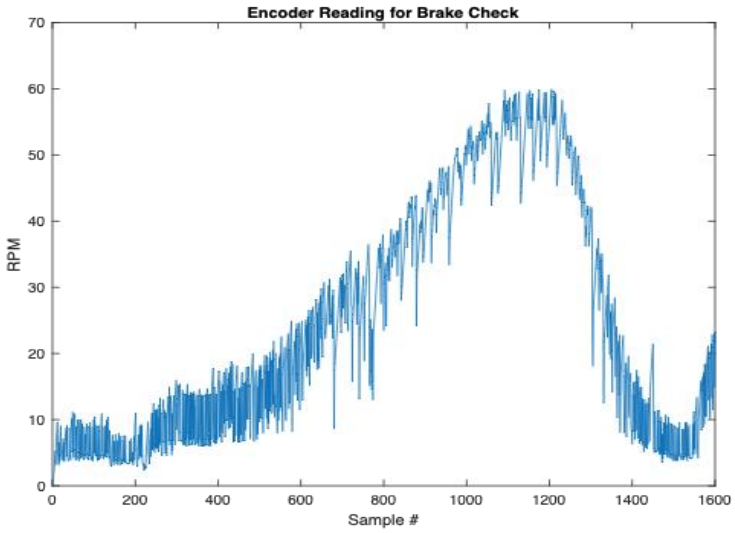
as the PWM signal and a ratio can be selected between the wheels. The user is able to set the ratio and the PWM percentage allowing turns of the test bed that can be tested and measured to discover the limitations of the motor.[See appendix]

The next task was to drive the test bed around a radial track. That was accomplished by adding IR sensors to the front of the system to detect the track. The IR sensor has a transmitter and receiver and constantly is reading for the signal. Once the sensor passes over the track (black line) the receiver no longer retrieves the signal causing a binary one as output. For example, the test bed begins to lose track going toward the center and the right IR sensor crosses the track. This is corrected by leaving the left motor running and shut off the right motor until the sensor can retrieve the signal from the white platform surrounding the track. The same is done for the left IR sensor. [See appendix]

Finally, the encoder data was the last module to be written. All of the code prior has been in terms of duty cycle percentages and ready to be controlled by revolutions per minute. The specifications on the data sheet would help with writing the program as the motor is equipped with four pins power, ground, and two encoder pins. The motor encoder collects data digitally from with a hall effect sensor. As the motor spins a magnet inside that when passes the sensor creates a rising edge and falls until the next rotation. [See appendix]. The code begins with a timer and counter initialized at zero. The program has a hardware interrupt that begins the timer and counter once the initial rising edge is detected and clocked once the counter reaches two. It calculates revolutions per minute based on the characteristics of our motor. The motor has 16 ticks per revolution, a gear box ratio of 120:1, and drive shaft of 1920 ticks per revolution. The RPM is then calculated with the formula RPM = 60/(**Δ**t *960). The values are pushed into a buffer array of fifty data points and the output is the moving average, which is the method of filtering the data for more accurate results. [See appendix]

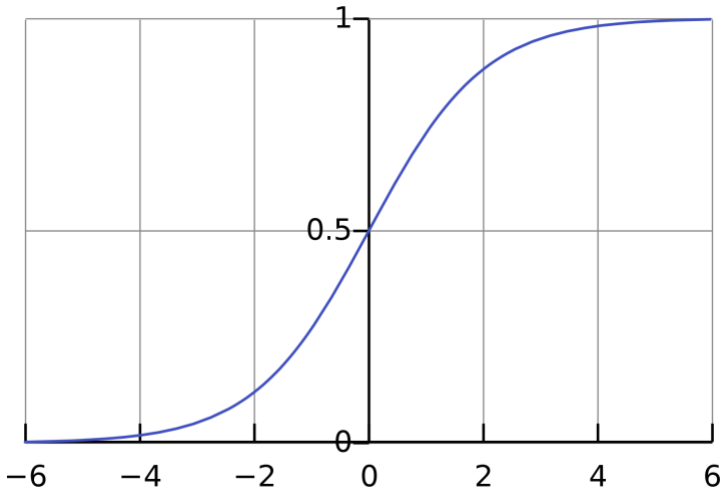
At the system level, the primary motivation to implement line sensing on each car was to enable the researchers to have the flexibility to create any track size they desire. The first porotype of the testbed resembled that of a slot car track and was a scaled model of the original full-scale experiment. Expansion of this track was not only costly, but it was time intensive. The “power strips” were also had a small cross-sectional area that would need to be replaced with one of a larger cross-sectional area to support the current draw from more vehicles. The line following was implemented with two IR sensors that straddle a line of dark contrast to its background. The operation was that if one of the IR sensors detected the line it would temporarily suspend the wheel on the respective side of the car. This would result if the car taking a slight turn, and once the sensor no longer saw the line it would resume wheel’s operations. Major pitfalls to this method come when there are turns of an acute angle. This was not detrimental to the operation of car because the environment that is being simulated is highways, and their inherent geometries do not include acute turns.

Table 2: Logistic Function acceleration example



- e = the natural logarithm base (also known as Euler's number),
- x_0 = the x-value of the sigmoid's midpoint,
- L = the curve's maximum value, and
- k = the logistic growth rate or steepness of the curve.[1]

Figure 4: The Logistic Curve



Design Verification and Test Methods

Chassis and Fittings

For the material of the chassis and support brackets, an eighth inch thick acrylic sheet is the choice. Acrylic is popular and available, and its strength is appropriate for this application. Among the thickness choices, we qualitatively chose the eighth inch. More importantly, an extruded acrylic was the choice over cast acrylic. Its laser cutting is cleaner and has tighter clearances. The fits and tolerances were created and tested on a sample sheet. The following was documented:

Material:	Acrylic 1/8"	**Measured:**	0.105 in
Slot Dimensions for Press Fit (inches)			
Slots:	**Motor**	**IR Sensor**	**Tag**
Test1	0.1	0.1	0.1
Comments	Loose	Loose	Loose
Test2	0.0975	0.09	0.09
Comments	A bit tight	A bit tight	Good
Test3	0.0985	0.093	
Comments	Perfect	A bit loose	
Test4		0.0915	

Table 3: Fitting Comparison Size Chart

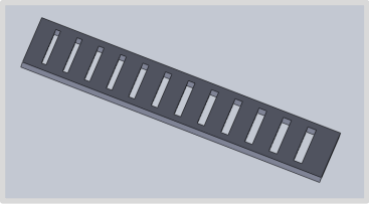


Figure 6: variable slot size

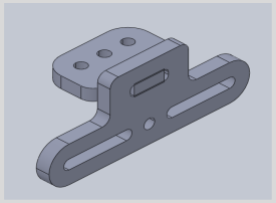


Figure 5: example of fitting components

Among the family of 18650 batteries, the most significant specifications are energy capacity, max current, and ,of course, cost. A comparison of four worthy batteries varied in these areas. The estimate of required current draw was based on published documentation of the motor controller and motors (appendix C). The energy capacity ranges from 2500mAh to 3500mAh. Based on published specs, the cars would run about 800 mA. The 2500mAh capacity translates to over 3 hours of run time. This seemed more than sufficient for an experimenting session and thus meets the needs. The decision is the low cost Samsung 25R, boasting max current draw of 20A and 2500 mAh energy capacity.

Figure 7: The Samsung Battery

The motors need the critical energy supply. Originally, the aim was to reach a scaled speed of 3 mph. It was nearly achievable, but not dependable. In Figure 8, you see that the speed is inconsistent beyond 60% of maximum duty cycle. After further research, a scaled speed of 1 mph was sufficient and thus the motor selection was complete.

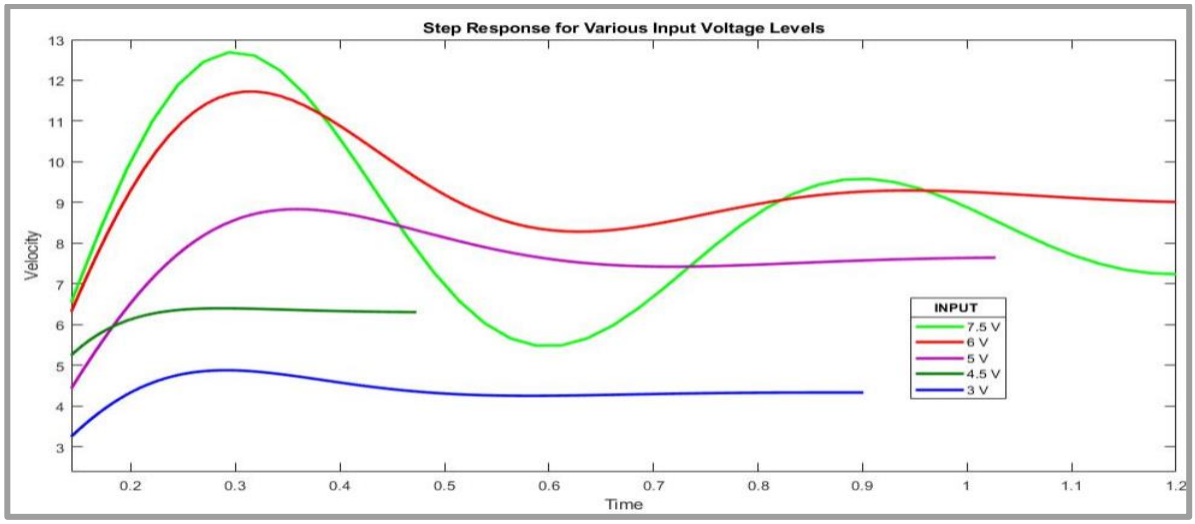


Figure 8 Step response of motors using Simulink system ID toolbox

In order to use the motors within their linear response range, deriving the transfer function was necessary to understand the where this range existed.

Ultrasonic sensor was tested before introducing it to any system programming. Rough tests were first administered to get a ballpark of its accuracies for viability. Using a simple code through the Raspberry Pi, “echo.py,” the sensor put out multiple data points. In figure 8, the actual distance is compared to the measured distance. The line following sensor is a ready-to-go tool that is essentially a true or false logic.

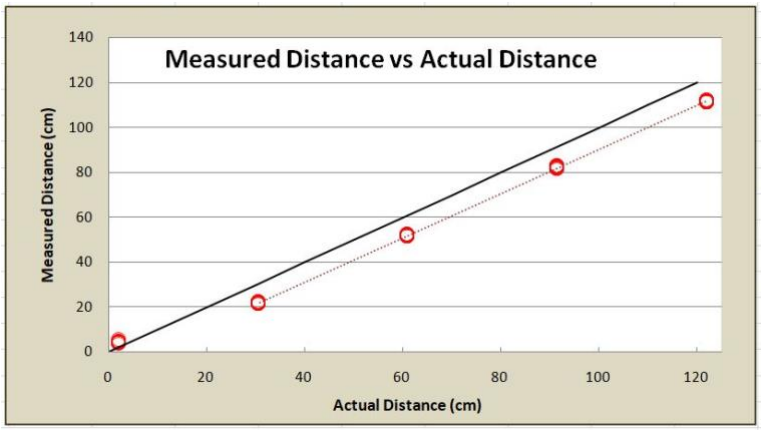


Figure 9: Apparent vs Actual Distance from Ultrasonic sensor

Performance Validation and Test Results

The fabrication decisions were proven successful in cost and efficiency. The estimate of extruded acrylic material cost is about $3 per car, and it can be cut in less than 10 minutes. The design of the chassis is laser cut to a total length of 8 inches and a max width of 4 inches. The components are mounted on top and bottom of the chassis. This footprint is smaller than any previous design. The bracket for the distance sensor is secured to the chassis with one screw which allows it to pivot and be optimally angled to face perpendicular to the lead car while on a round track. The IR sensor bracket is designed with horizontal slots which allows for variance in the IR sensor placement. The Raspberry Pi and motor controller component placement was determined mostly for ease of access to important pins, ports and terminals. [refer to V2 assembly- projected views in appendix] Also on V2, component placement was optimized. The wheel caster was placed up front which creates a more natural look. An upgraded motor tab and other supporting brackets were secured using screws and the switch from an L-shape motor to an I-shape motor allowed a slimming of the chassis enough to get 3 full chassis from one standard 12” acrylic sheet.

The prototype length is 8 inches. Wheel separation is 5 inches and it has a weight of 2 pounds. The center of mass is 1 inch ahead of wheel axles and 0.5 inches to the right side (closer to power supply). An 8-mm castor wheel is the front support which is left-right centered and 1 inch behind the front edge and 4 inches in front of the wheel axles. That puts the axles 3 inches from the rear edge. See appendix B for exploded view of the car.

Modular tests are necessary before a system wide connection. Each of the components were tested through the CPU. Later, a combination test was put together by the team to prove usability among programming, CPU, motor controller, motor, wheel differential, battery power, castor wheel, et al.

To ensure that the system level deployment of the software would go smoothly, each module was tested and validated. These modules still exist as individual programs that are called upon the

system level program. It is because of this that it is easier to explain from the top down. The only program that will be called upon by the user is a shell script named TestBed.sh. This will launch serval python scripts and have their outputs sent to a text file for post processing.

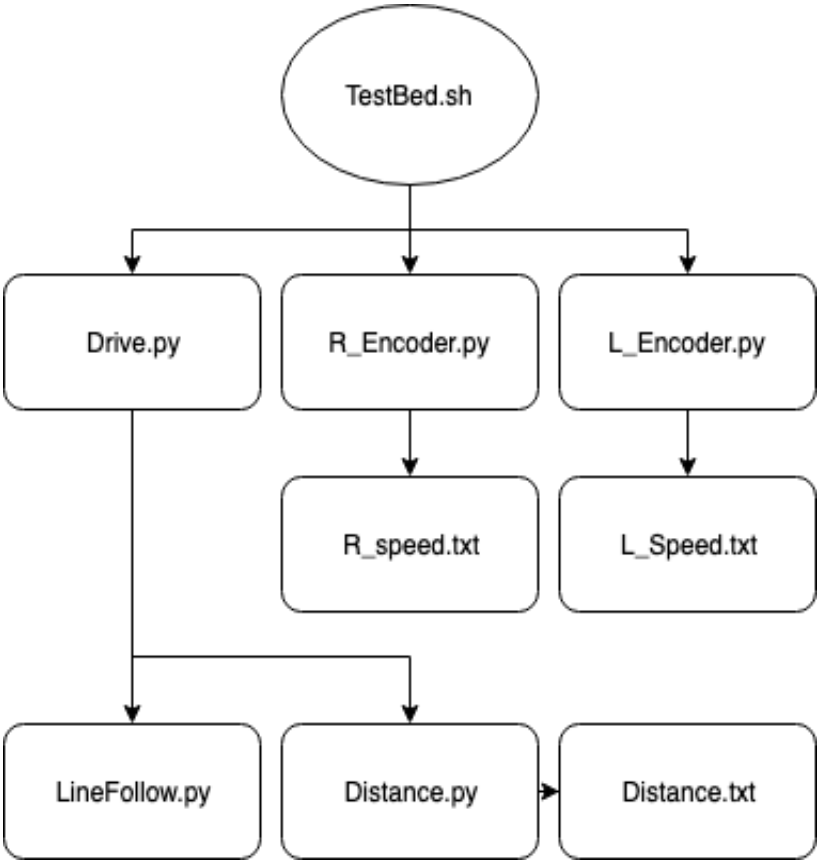


Figure 10: Flow Chart of System Software

Drive.py:
This program is a hybrid of the line follow and forward distance sensing programs that have been merged. The periodically sampled distance is the data that is sent to a text fil. Appendix C.10

Encoder.py:
These encoder programs are kept separate for ease of troubleshooting and faster post processing. When the data files are kept separate, so that they can be easily imported into MATLAB or excel as an individual variable column vectors. Appendix C.11

Text files were chosen as the storage method because of the simplicity of importing the data into a computation engine. The user is able to drag the file into the MATLAB workspace and create a variable of the data set, or select all the text and copy it into excel.

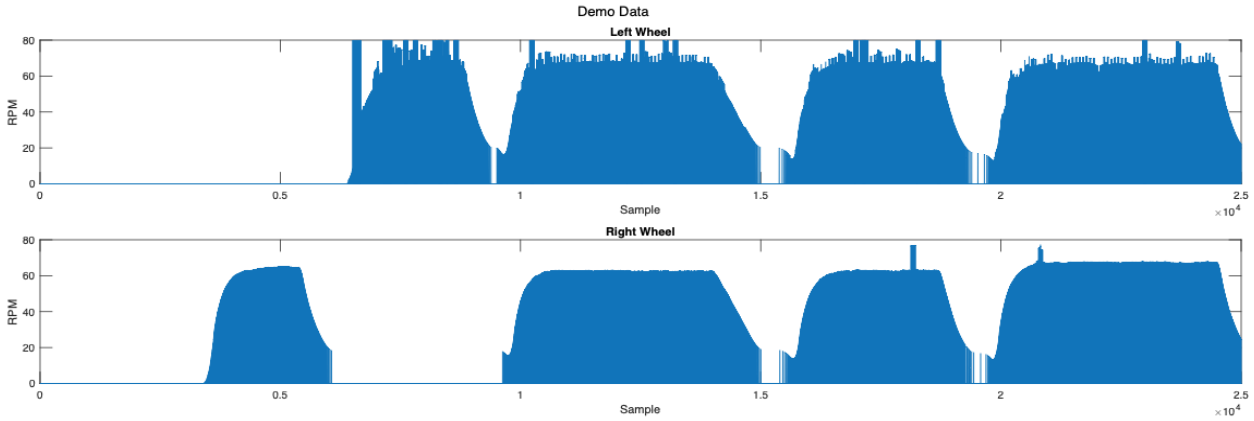


Figure 11: Left and Right Wheels data sample

In essence, the lines of code that directs the data and controls the moving parts have coalesced and we deem it a success.

Operational validation, Summary, and Conclusions

This product was effectively a research platform that has been developed for further continuation of Dr. Seibold's research. Over the past year and a half three prototypes have been built following the constraints of 1:23rd scale, approximately $99, and a function to enable the motors to have realistic vehicle dynamics. In addition to constraints meeting our goals, a streamlined interface was created. The interface only requires the user to have a learn a series of simple Linux terminal commands and understand computation software similar to MATLAB or Excel.

This Product was not able to successfully verify Dr. Seibold's initial experiments, but it was able to leave behind the documentation to duplicate the prototype for expansion of the car fleet. The foundation that has been laid for the successors of this project will be instrumental to their success. One recommended system level improvement is that a node on the LAN that will poll each of the cars during a test and display the behavior live on a monitor.

Appendices

A.1. RFP

College of Engineering
TEMPLE UNIVERSITY®

Request for Senior-Design Project Proposal
Fall, 2018

1) **Title**: Benchtop Testbed for Autonomous Vehicle Research

2) **Number of students and majors**:
4 students, mixture of MEE (2) and ECE (2) recommended
Student Qualifications:
ECE: Control theory, Programming experience (Python or C++)
MEE: Dynamics, SolidWorks, Laser cutter

3) **Project description:**
Autonomous vehicles are currently being tested on some public roadways throughout the world. While such in situ testing is necessary to validate the performance of the end-stage hardware and software in a vehicle, it is difficult, expensive, and dangerous. On the other hand, early-stage development is typically done in a simulated environment. However, "the problem with *simulations* is that they are *doomed to succeed*," as iRobot founder Rodney Brooks puts it. This project aims to bridge the gap between testing in simulation and using full-scale vehicles by creating a benchtop testbed. This will allow researchers to test out their algorithms in hardware and examine the effects at both a micro (individual car) and macro (traffic flow) level.

This project is a continuation of a previous senior design project to realize this goal. The previous team designed and built three test vehicles and a powered track for the vehicles to drive on. The team also created the basic software to move the robots using ROS and deployed it on a Raspberry Pi.[1] The next team to work on this project will redesign the cars to be smaller and cheaper (desired cost of $100/car, down from ~$180/car). They will also improve the control software so that the dynamics of the testbed vehicles are similar to those of a full-scale car. For example, if the test vehicle is built at a scale of 1/20 then it should have a braking distance of 1/20. If successful, the team will build 20+ vehicles and validate their control algorithm using data collected from prior tests conducted using full-scale vehicles. This project is part of a research effort in conjunction with Prof. Benjamin Seibold of the Math Department at Temple.

4) **Funding:**

[1] For a video of the previous team's results see:
htt //dri e. oo le.co /o en id 1EcDl efU_ wT lcXhRIiU d R4 a82

Additional funding supplied by the PI to create a full fleet of 20+ vehicles available contingent on student performance.

5) **Additional reading:**
-Article describing the project's research goals: https://news.temple.edu/temple-magazine/2018/spring/drive-time
-Paper on full-scale vehicle results: https://arxiv.org/pdf/1705.01693.pdf and accompanying video: https://www.youtube.com/watch?v=2mBjYZTeaTc
-ROS: http://wiki.ros.org/
-Linux on the Raspberry Pi: https://ubuntu-mate.org/raspberry-pi/
-Dynamic Similarity: https://en.wikipedia.org/wiki/Similitude_(model)

Item	**Submission deadline**	**Graded by**[2]
RFP (request for proposal) by faculty advisor	Sep 10, 2018 (Monday)	(submitted by faculty)
LoI (letter of intent)	Sep. 29, 2018 (Saturday)	(not graded)
Draft proposal	Oct 13, 2018 (Saturday)	Oct. 22, 2018 (Monday)
Final proposal	Nov 3, 2018 (Saturday)	Nov 12, 2018(Monday)

[2] To grade, note that the first page of the student's work will be a rubric to be filled in, and uploaded in OwlBox.

B.2. Motor Power Calculation

Power Required for max acceleration to max speed

ΣF=ma

F_traction - F_rolling resistance = mass* max_acceleration

F_ rolling resistance = c*m*g where "c" is the coef._rolling_resistance (unitless)[1]

F_t = m*accel_max + F_r

Note: The information about rolling resistance coefficients (for airless rubber tires) was obtained from the following website: https://www.engineeringtoolbox.com/rolling-friction-resistance-d_1303.html

- we assume "c" to be 0.005
- A very conservative guess for total car mass is 2 kg (all plastic components except for battery)
- The maximum estimated driver acceleration is 3.5 m/s^2 but for 1:23 scale equals about 0.15 m/s^2 (research conducted by Mathew Snare)
- For a scaled down range of 0-60mph we want a top velocity of 1.12 m/s

F_traction = m*accel_max + c*mg

F_traction= (2 kg)(0.15 m/s^2) + (0.005)(2 kg)(9.8 m/s^2) = 0.4 N

Power = F*V

Motor Power Required = (0.4 N)(1.12 m/s) = 0.45 Watts (peak) or 64 in-oz/s

But since we have two motors per car we only need half of that: 0.23 Watts (peak) or 32 in-oz/s

Note: This combination of acceleration and velocity would be rare since once peak velocity is reached, acceleration would go to zero.

Calculating actual Motor Power

Actual Motor Power = Txω = torque x angular velocity (rad/s)

Again, for convenience we want to input torque in terms of in-oz, angular velocity in terms of rpm and output power in terms of in-oz/sec. Also, K accounts for the two motors we will be using. We introduce a conversion constant, K = 0.21

Motor Power = (0.21)(T)(ω)

From a data sheet found on digi-key for a "TT" style DC motor (our motors):
https://www.adafruit.com/product/3777

- Torque: 0.15-0.6 Nm
- Min RPM at 6V: 180 RPM

min torque x min RPM (crazy conservative!) = (0.15 Nm)(19 rad/s) = 2.85 watts from one motor (we have two) but we only need 0.5 watts peak!

We are using a very economical 80mm diameter wheel:

ω=V/r = (1.12 m/s)/40e-3 m) = 28 rad/s = 267 RPM we need to make sure that at the upper allowable limit of voltage to the motor, we can reach this RPM and that there be sufficient torque at that speed.

Torque required:

T=F_traction(peak) x wheel radius = (0.4 N)(0.08 m) = 0.032 Nm required

Motor torque available at 267 RPM:

T_motor = Power/ω = (2.85 Nm/s)/(28 rad/s) = 0.102 Nm available or 0.204 Nm for two motors.

This means that at peak RPM we need about 16% of the available torque at this speed.

[1] htt //www.en ineerin toolbo .co /rollin -friction-re i tance-d_1303.ht l

B.3 The Logistic Function

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

C.1 Duty Cycle to Speed Comparison

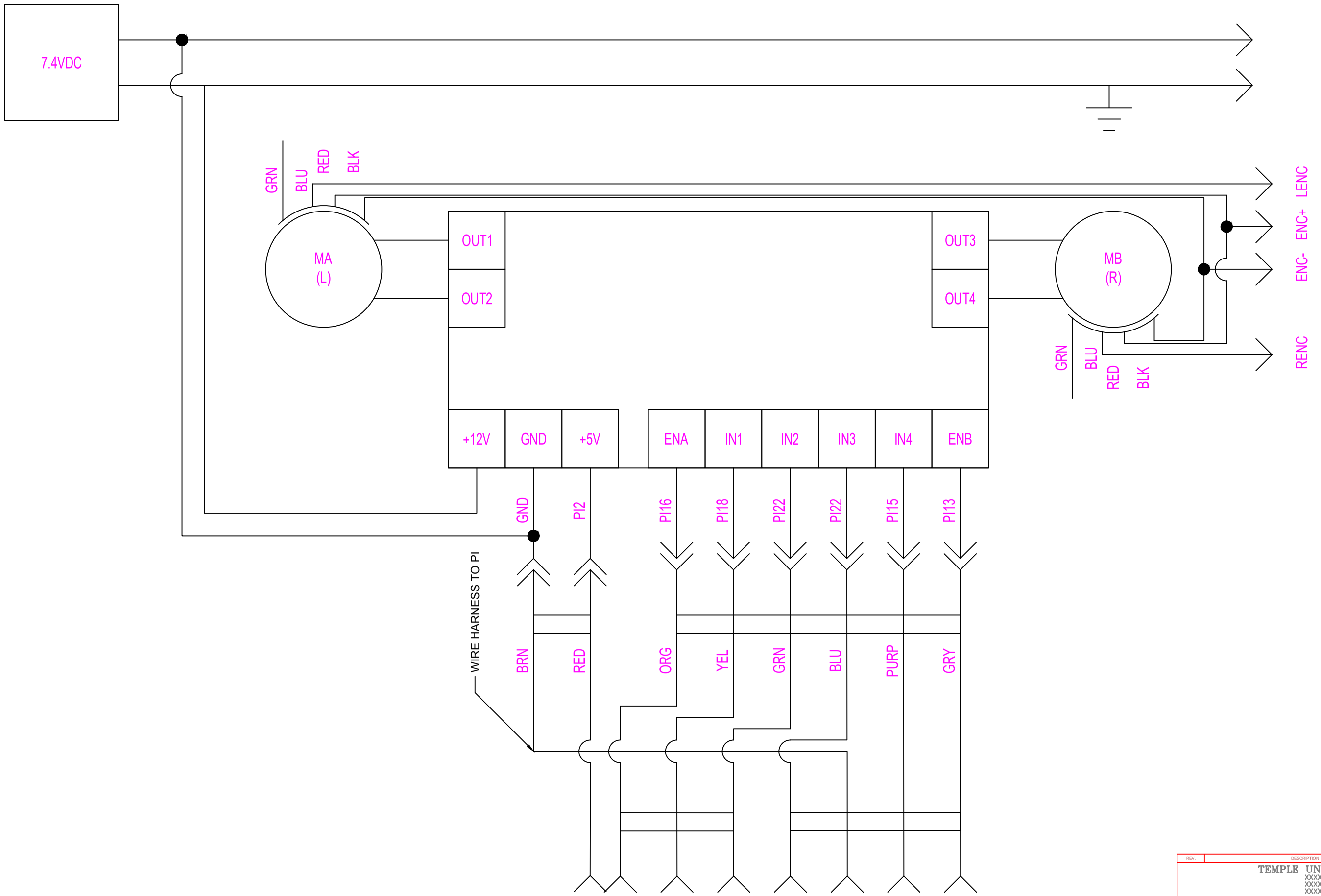
Table 4: Duty Cycle to Speed Comparison

D t C cle ( )	Ratio (L/R)	Di tance(in)	Ti e( )	S eed (in/)	A S eed (in/)	Standard De iation
15	1	50.250	9.800	5.128		
15	1	48.250	9.150	5.273	5.238	0.097
15	1	46.000	8.660	5.312		
20	1	213.000	26.700	7.978	7.978	n/a
35	1	63.750	5.650	11.283		
35	1	65.650	5.930	11.071	11.334	0.292
35	1	65.000	5.580	11.649		
50	1	86.000	6.210	13.849		
50	1	77.750	5.230	14.866	14.603	0.663
50	1	80.000	5.300	15.094		
60	1	69.500	4.260	16.315		
60	1	130.000	7.820	16.624		0.164
60	1	112.000	6.840	16.374	16.438	
60	1	-	-	-		
60	1	NA				
75	1	162.500	8.380	19.391		
75	1	-	-	-		
75	1	-	-	-	.	
75	1	-	-	-		
95	1	-				

C.2Wiring Diagrams

C.3 Procedure for Track Data Collection

101-00
101-01
101-02
101-03
101-04
101-05
101-06
101-07
101-08
101-09
101-10
101-11
101-12
101-13
101-14
101-15
101-16
101-17
101-18
101-19
101-20
101-21
101-22
101-23
101-24



101-25
101-26
101-27
101-28
101-29
101-30
101-31
101-32
101-33
101-34
101-35
101-36
101-37
101-38
101-39
101-40
101-41
101-42
101-43
101-44
101-45
101-46

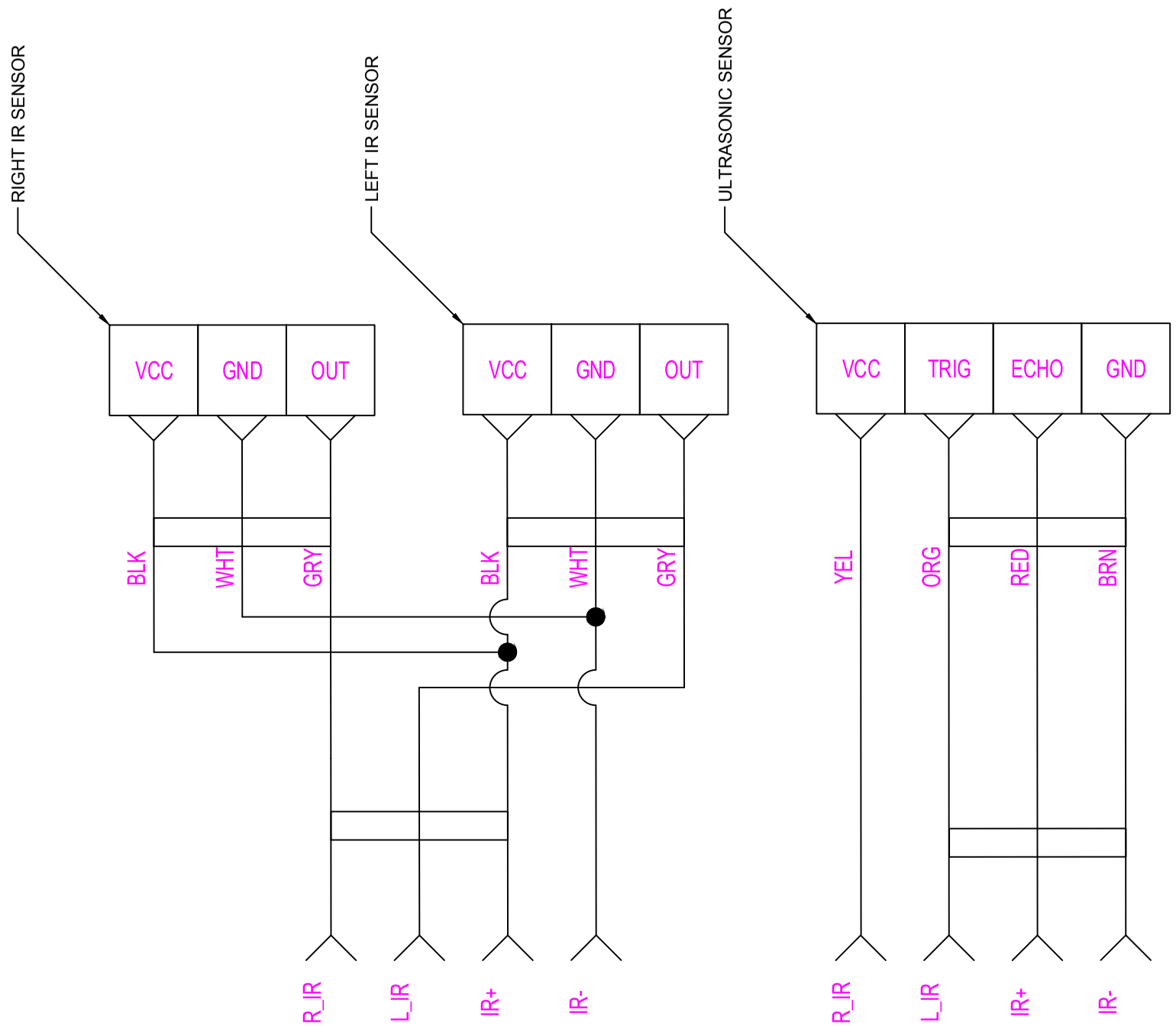
REV.	DESCRIPTION	DATE

TEMPLE UNIVERSITY
XXXX
XXXX
XXXX

TITLE: WIRING DIAGRAM FA2019 SDII TEAM 18

EREF: ELC_SYS335	DRW BY: CWM	DRAWING NO. ECE-0001
1 ST. USED: SYS-335	DATE: 7/18/3019	
CAD No.: SAME AS DWG	SOFTWARE: XXX	P.O. 1 of 4

102-00 102-01 102-02 102-03 102-04 102-05 102-06 102-07 102-08 102-09 102-10 102-11 102-12 102-13 102-14 102-15 102-16 102-17 102-18 102-19 102-20 102-21 102-22 102-23 102-24 102-25 102-26 102-27 102-28 102-29 102-30 102-31 102-32 102-33 102-34 102-35 102-36 102-37 102-38 102-39

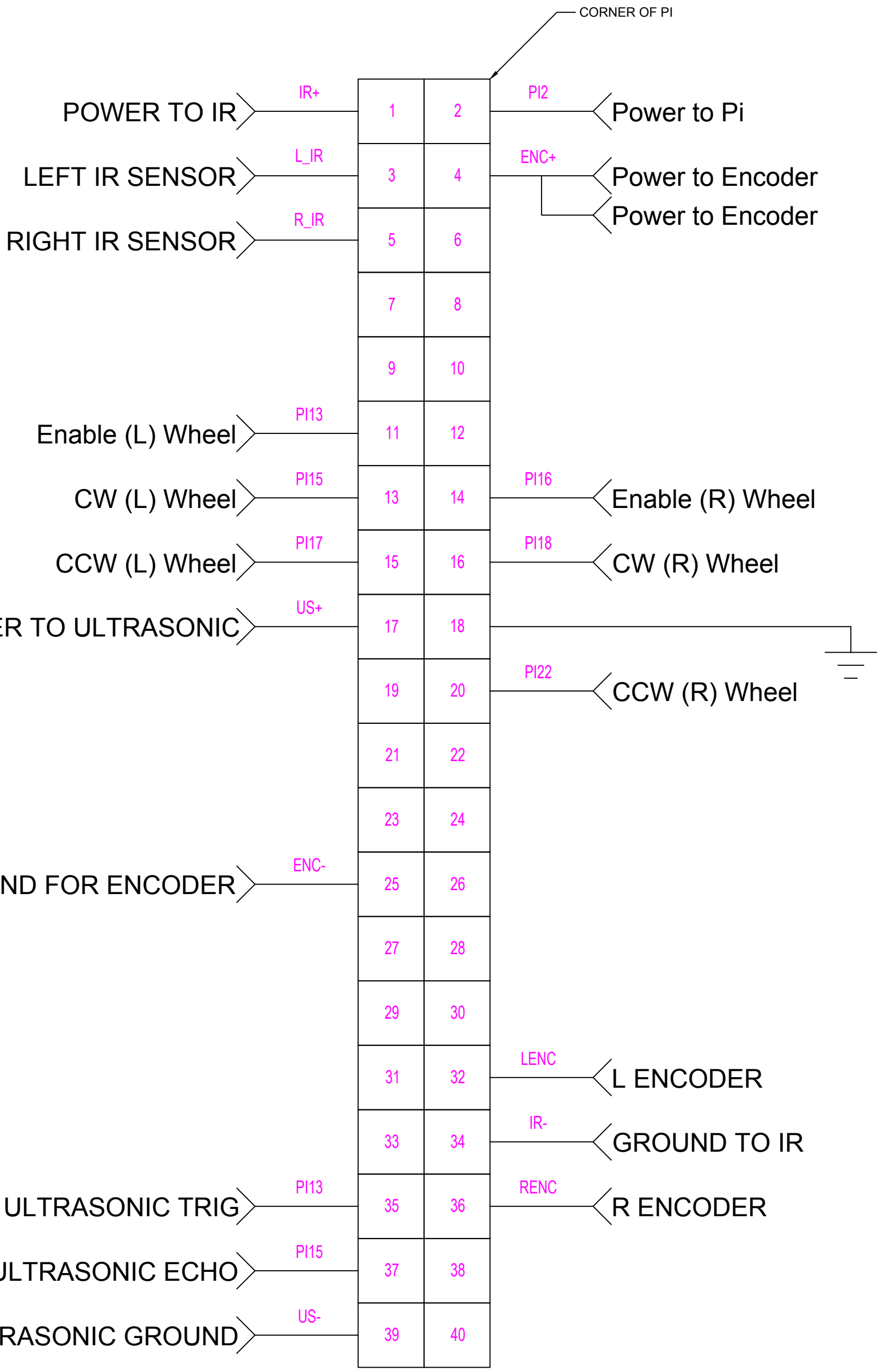


REV.	DESCRIPTION	DATE

TEMPLE UNIVERSITY
XXXX
XXXX
XXXX

TITLE: WIRING DIAGRAM FA2019 SDII TEAM 18

EREF: ELC_SYS335	DRW BY: CWM	DRAWING NO. ECE-0001
1 ST. USED: SYS-335	DATE: 7/18/3019	
CAD No: SAME AS DWG	XXX	PG 2 of 4



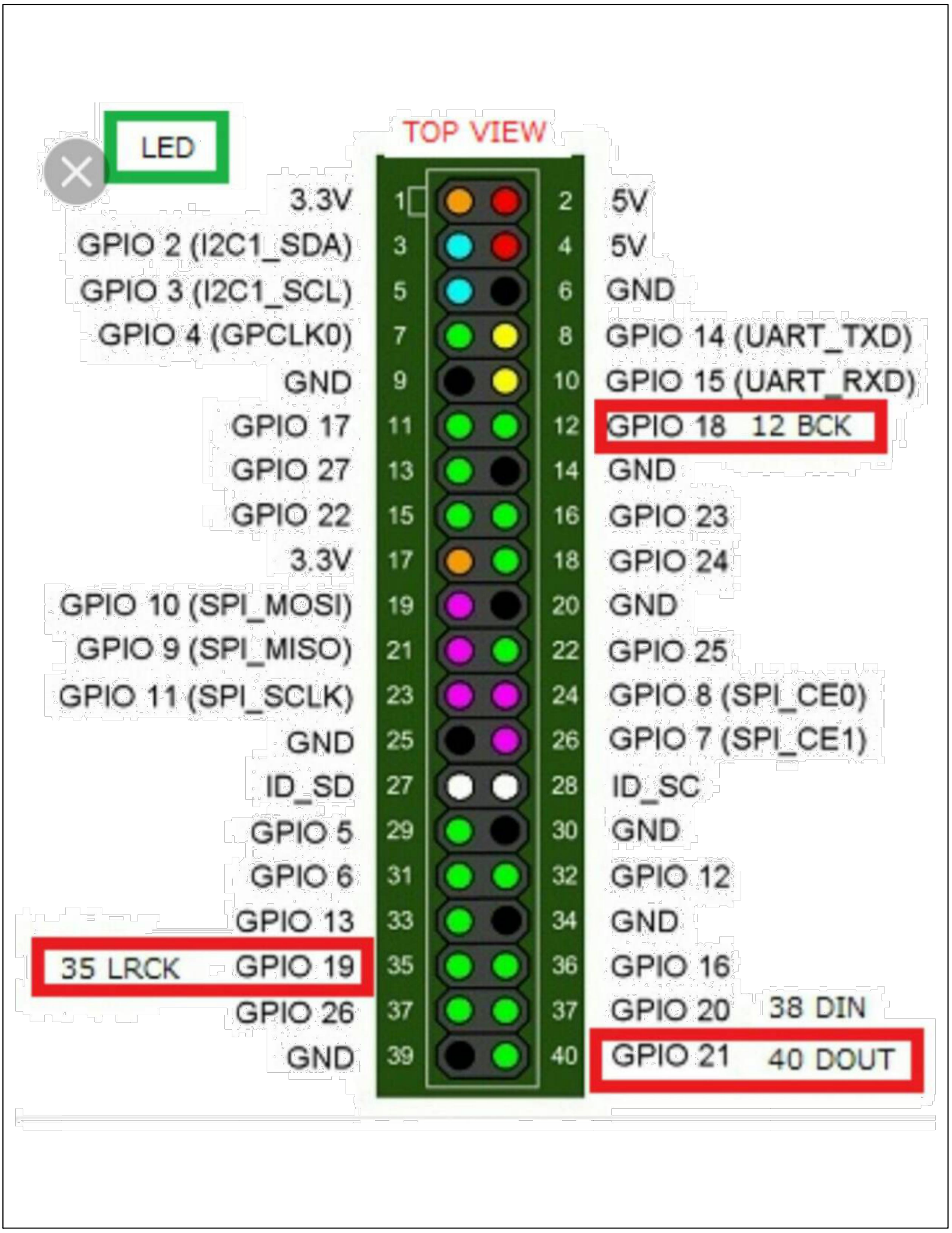
130-00 130-01 130-02 130-03 130-04 130-05 130-06 130-07 130-08 130-09 130-10 130-11 130-12 130-13 130-14 130-15 130-16 130-17 130-18 130-19 130-20 130-21 130-22 130-23 130-24 130-25 130-26 130-27 130-28 130-29 130-30 130-31 130-32 130-33 130-34 130-35 130-36 130-37 130-38 130-39

REV. DESCRIPTION DATE

TEMPLE UNIVERSITY
XXXX
XXXX
XXXX
XXXX

TITLE
WIRING DIAGRAM FA2019 SDII TEAM 18

EREF ELC_SYS335
1 ST. USED SYS-335
DRW BY CWM
DATE 7/18/3019
DRAWING NO. ECE-0001
CAD No SAME AS DWG
XXX
PG. 3 of 4



REF: ELC_SYS335
1 ST. USED: SYS-335
CAD No.: SAME AS DWG

TITLE: WIRING DIAGRAM FA2019 SDII TEAM 18

DRW BY: CWM
DATE: 7/18/3019

DRAWING NO.: ECE-0001

XXX

PG. 4 of 4

REV | DESCRIPTION | DATE

TEMPLE UNIVERSITY
XXXX
XXXX
XXXX

104-00
104-01
104-02
104-03
104-04
104-05
104-06
104-07
104-08
104-09
104-10
104-11
104-12
104-13
104-14
104-15
104-16
104-17
104-18
104-19
104-20
104-21
104-22
104-23
104-24
104-25
104-26
104-27
104-28
104-29
104-30
104-31
104-32
104-33
104-34
104-35
104-36
104-37
104-38
104-39

TESTING PROCEDURE: Evaluating motor behavior	Senior Design Team 18 Temple University Fall 2019

Objective

To determine the correlation of the wheel differential and input voltage to the turning radius and speed

Description and Theory

The prototype has two wheels that control the speed and direction. There is a castor wheel for stability. The midpoint between these wheels is our point under scrutiny. The vehicle movement is measured by this centerpoint.

The wheels are subject to a DC motor that receives power from the motor control that is controlled by a raspberry pi. The code in the Raspberry Pi Prompts the user to input a percentage of full speed and a ratio of right to left wheel speed. By manipulation of the (1) power supply and the (2) wheels through a differential ratio, we can control the speed and turning radius of the centerpoint.

Formulae

$$\omega = \frac{V}{R} \quad (1)$$

$$V = \frac{r}{2}(\omega_R + \omega_L) \quad (2)$$

$$\omega = \frac{r}{L}(\omega_R - \omega_L) \quad (3)$$

$$\omega_R = \frac{V}{r}\left[\frac{L}{2R} + 1\right] \quad (4)$$

$$\frac{\omega_L}{\omega_R} = \left[\frac{2R-L}{2R+L}\right] \quad (5)$$

ω - Angular velocity of the car (Rad/s)
V - Tangential velocity of the car
ω_R - Angular velocity of right wheel
ω_L - Angular velocity of left wheel
R - radius of track9
L - distance between car wheels
r- radius of wheel

Constant Variables: wheel radius (r), distance between wheels(L), distance from centerpoint to marker tip (d)
Input variables: left rotation speed (wl) in ratio to right rotation speed (wr), right rotation power level(duty cycle, 0<P<=1)

Goals

1. Correlation of power supplied to motor (scale from 0 to 100) to vehicle speed

2. Correlation of wheel differential (scale 0 to 1) to vehicle's turning radius.
3. Accuracy of a linear relationship of power supply to vehicle speed.
4. Accuracy of wheel differential to radial turns.
5. Preliminary range of speeds
6. Preliminary range of turning radii

Equipment

Vehicle (w/ raspberry pi, motor control, battery pack, motor, wheels, and all connecting parts), keyboard, marker attachment, marker, ruler, cleaning spray, rag

The placement of the marker creates a tangent line from the centerpoint to the marker tip. The radius of turn is created by the wheel differential. When the vehicle turns, the period of turn equals the period of rotation around the centerpoint. It is the centerpoint that creates the turn, but the marker, at a distance (d) from the centerpoint, marks the arc being measured.
For a linear path, the marker is offset by distance d and accurate for the path measurement.

Procedure

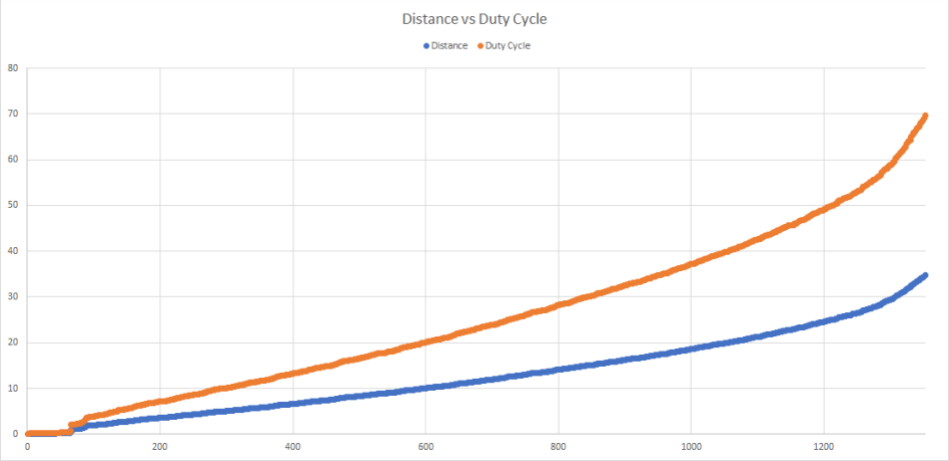
Experiment 1
Data to be collected: Distance, Time

1) The cars start position was marked
2) The car was wirelessly signaled to begin driving with start time recorded
3) After several yards the car was signaled to stop and the stop time recorded
4)The cars end position was marked
5) The distance between the stop and end position was measured and recorded
6) Steps 1-5 were repeated 3 times
The procedure above was executed at various power settings with a wheel ratio of 1 (no turns).

Experiment 2
Data to be collected: Number of rotations, Time

1) The cars start position was marked
2) The car was wirelessly signaled to begin driving with start time recorded
3) After 10 complete turns the car was signaled to stop and the stop time recorded
4) Steps 1-3 were repeated 3 times
The procedure above was executed at various power settings with a wheel ratio of -1 (pure rotation).

C.4 Ultrasonic sensor and motor speed graph



C.5 Ultrasonic Code

```
i     ort RPi.GPIO a   GPIO
i     ort ti   e
GPIO.  et   ode(GPIO.BOARD)


TRIG    29
ECHO    31



   rint  Di  tance Mea    re   ent In Pro  re
tr
   GPIO.  et    (TRIG GPIO.OUT)
   GPIO.  et    (ECHO GPIO.IN)
   while Tr  e
                  GPIO.o  t    t(TRIG  Fal  e)
                rint  Waitin   For Sen  or to Settle


                  ti   e.  lee  (.00001)


                  GPIO.o  t    t(TRIG  Tr  e)
                  ti   e.  lee  (0.00001)
                  GPIO.o  t    t(TRIG  Fal  e)


                  while GPIO.in    t(ECHO)    0
                        l  e_  tart    ti   e.ti   e()


                  while GPIO.in    t(ECHO)    1
                        l  e_end    ti   e.ti   e()


                    l  e_d  ration       l  e_end -    l  e_  tart


                  di  tance       l  e_d  ration    17150
                  di  tance    ro  nd(di  tance  20)


                   rint  Di  tance    di  tance    c
e  ce  t    e  boardInterr    t


finall
   GPIO.clean    ()
```

C.6 Motor Code

```
 P  r  o  e  to act a  a  roof of conce  t for    in  the PWM of the ra   berr    i
 to control the    eed of two diffrent   otor    i  en ar      ent  of
 d  t  c  cle of ri  ht wheel and ratio of ri  ht wheel to left Wheel


 Note   there i    o   e   ne   ected latenc  in the PWM for the ri  ht and left wheel
 The latenc    ee     to  ar . At thi    oint in ti  e there i   no  roble   with
 thi  b  there   a  be a  roble   in the f  t  re


i    ort RPi.GPIO a  GPIO
RW_PIN   18
LW_PIN   13
RW_ENA   16
LW_ENA   11
tr
   GPIO.  etwarnin   (Fal  e)
   GPIO.  et   ode(GPIO.BOARD)
   GPIO.  et   (RW_PIN GPIO.OUT)   Ri  ht Wheel
   GPIO.  et   (RW_ENA GPIO.OUT)   Ri  ht Wheel
   GPIO.o  t   t(RW_ENA  1)
   GPIO.  et   (LW_PIN GPIO.OUT)   Left Wheel
   GPIO.  et   (LW_ENA GPIO.OUT)   Left Wheel
   GPIO.o  t   t(LW_ENA  1)


   r   GPIO.PWM(RW_PIN 50)   Ar      ent  are  in and fre   enc
   r.  tart(0)   Ar      ent i  initial d  t  c  cle  it  ho  ld be 0


   l   GPIO.PWM(LW_PIN 50)   Ar      ent  are  in and fre   enc
   l.  tart(0)   Ar      ent i  initial d  t  c  cle  it  ho  ld be 0


   ratio   1   Wheel  will  tart off at the  a  e ratio


   while Tr  e
      d  t     in   t(  n nSelect d  t  c  cle in the ran  e 0.0 to 100.0  )   Ta  e d  t  c  cle a  in   t
      ratio    in   t(  nSelect a wheel ratio (Ri  ht/Left)  )   Ta  e ratio of ri  ht to left a  in   t
      r.Chan  eD  t  C  cle(d  t  )   Chan  e d  t  c  cle for ri  ht wheel
      l.Chan  eD  t  C  cle((d  t     ratio))   Chan  e d  t  c  cle for left wheel with ratio
e  ce  t    e  boardInterr   t


finall
   GPIO.clean   ()
```

C.7 Infrared Sensor Code

```
i    ort ti  e
i    ort o
i    ort RPi.GPIO a  GPIO
GPIO.  et  ode(GPIO.BOARD)
  For Line Followin
LOGGER   1
L  in   3
R  in   5
GPIO.  et   (L  in  GPIO.IN)
GPIO.  et   (R  in  GPIO.IN)


  For PWM
RW_PIN   18
LW_PIN   13
RW_ENA   16
LW_ENA   11


tr
   GPIO.  etwarnin   (Fal  e)
   GPIO.  et   (RW_PIN  GPIO.OUT)    Ri  ht Wheel
   GPIO.  et   (RW_ENA  GPIO.OUT)    Ri  ht Wheel
   GPIO.o  t   t(RW_ENA  1)
   GPIO.  et   (LW_PIN  GPIO.OUT)    Left Wheel
   GPIO.  et   (LW_ENA  GPIO.OUT)    Left Wheel
   GPIO.o  t   t(LW_ENA  1)


     initiali  e PWM
   r   GPIO.PWM(RW_PIN  50)    Ar      ent  are  in and fre   enc
   r.  tart(0)    Ar      ent i  initial d  t  c  cle  it  ho  ld be 0
   l   GPIO.PWM(LW_PIN  50)    Ar      ent  are  in and fre   enc
   l.  tart(0)    Ar      ent i  initial d  t  c  cle  it  ho  ld be 0
```

```
    while Tr  e
        if (GPIO.in    t(L  in)      Tr  e)
            rint( STOP LEFT WHEEL    )
          r.Chan  eD  t  C  cle(0)
            ti   e.  lee  (.25)
        el  e
             rint( 0 Left )
          r.Chan  eD  t  C  cle(20)


        if (GPIO.in    t(R  in)      Tr  e)
            rint( STOP RIGHT WHEEL    )
          l.Chan  eD  t  C  cle(0)
            ti   e.  lee  (.25)
        el  e
             rint( 0 Ri  ht )
          l.Chan  eD  t  C  cle(20)     Chan  e d  t   c  cle for ri  ht wheel
e  ce  t
      e  boardInterr    t


finall
    GPIO.clean    ()
```

C.8 Encoder Code

```
i     ort RPi.GPIO a   GPIO
i     ort ti   e
i     ort   ci    . i  nal a
NUM
GPIO.  et   ode(GPIO.BCM)       et GPIO n      berin          te     to BCM
GPIO.  et     ( en  or  GPIO.IN)       et o  r   en  or    in to an in    t
  a     le    2     N     ber of tic   co  nt   between calc  lation
N_Window       1      N window      er rotation
co  nt     0
 tart     0
end     0
def   et_  tart()
      lobal   tart
      tart    ti   e.ti   e()
def   et_end()
      lobal end
    end    ti   e.ti   e()
def   et_r     (c)
      lobal co  nt      delcear the co  nt   ariable    lobal    o we can edit it
    if not co  nt
         et_  tart()     create    tart ti   e
        co  nt     co  nt     1      increa  e co  nter b    1
    el  e
        co  nt     co  nt     1
    if co  nt        a      le
         et_end()      create end ti   e
        delta     end -  tart      ti   e ta  en to do a half rotation in    econd
        window      len(delta)
          delta     delta / 60      con  erted to     in  te
          con  erted to ti   e for a f  ll   in  le rotation
          r         (  a     le / delta) / (960)    1 120 i   the    ear ratio
            rint delta
        ti   e.  lee  (.001)
        co  nt     0      re  et the co  nt to 0
        h         .  et_window( trian      window)
        fil         .con  ol  e(delta  h / h.       ())
          rint fil
  e  ec  te the   et_r      f  nction when a HIGH    i  nal i   detected
GPIO.add_e  ent_detect(  en  or  GPIO.RISING  callbac      et_r      )
tr
    while Tr  e       create an infinte loo    to   ee    the    cri  t r  nnin
        ti   e.  lee  (0.1)
e  ce  t    e  boardInterr    t
      rint         it
    GPIO.clean    ()
```

C.9 WPA Supplicant File Mod

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
ap_scan=1
update_config=1

network={
                                                    scan_ssid=1
                                                ssid="tusecurewireless"
                                                 key_mgmt=WPA-EAP
                                                       eap=PEAP
                                                identity="tug96858"
                                        phase1="peaplabel=auto peapver=0 "
                                                phase2="auth=MSCHAPV2"
                                                  password="Ale3v$22"
}

network={
                                                       ssid="eduroam"
                                                        scan_ssid=1
                                                   key_mgmt=WPA-EAP
                                                         eap=PEAP
                                                   identity="tug96858"
                                                       password="#"
                                                  phase1="peaplabel=0"
                                                phase2="auth=MSCHAPV2"
}
```

C.10 Encoder X 2

```
i    ort RPi.GPIO a   GPIO
 i    ort  ci    . i  nal a    i  nal
i    ort ti   e
i    ort collection
i    ort n          a   n
n  . et_  rinto  tion  (thre  hold    n  .inf)


 en  or    32     define the GPIO   in o  r   en  or i   attached to


GPIO.  et   ode(GPIO.BOARD)       et GPIO n     berin         te     to BCM
GPIO.  et    (  en  or  GPIO.IN)      et o  r  en  or   in to an in    t


 a     le    2     N     ber of tic   co  nt   between calc  lation
N_Window      1   N window     er rotation
co  nt    0
delta    0
 tart    0
end    0
r         0
 lobal a
a



d    collection  .de    e(   a  len  50)
for i in   ran  e(50)
                    d.a    end(0)
 h        .  et_window( trian    3000)


def   et_  tart()
     lobal   tart
     tart    ti   e.ti   e()
```

```
def  et_end()
     lobal end
   end   ti   e.ti   e()
def  et_r    (c)
     lobal co  nt     delcear the co  nt   ariable   lobal   o we can edit it
     lobal a
   if not co  nt
        et_  tart()     create   tart ti   e
      co  nt    co  nt    1     increa  e co  nter b   1
   el  e
      co  nt    co  nt    1



   if co  nt        a     le
       et_end()     create end ti   e
      delta    end -   tart     ti   e ta  en to do a half rotation in   econd
      delta    delta / 60     con  erted to    in  te
      r        (1 / delta) / (960)   1 120 i   the   ear ratio
         rint delta
      d.a    end(r     )
      a        n  .   ean(d)
       fil        .con  ol  e(d h/h.       ())
         rint fil 2999           (d)/len(d)
       ti   e.  lee  (.001)
       o  t    t      i  nal.lfilter(NUM 1 d)
      co  nt    0     re  et the co  nt to 0
                      e  ec  te the   et_r      f  nction when a HIGH   i  nal i   detected
GPIO.add_e  ent_detect(  en  or  GPIO.RISING  callbac      et_r     )


tr
   while Tr  e      create an infinte loo   to   ee   the   cri  t r  nnin
      ti   e.  lee  (1)
       lobal a
       rint  L    a
      a        0
e  ce  t    e  boardInterr     t
     rint         it
   GPIO.clean     ()
```

C.10 Motor Code X2

```
i     ort ti   e
i     ort o
i     ort RPi.GPIO a   GPIO
i     ort    ath
GPIO.  et   ode(GPIO.BOARD)
   For Line Followin
LOGGER    1
L  in    3
R  in     5
TRIG    35
ECHO    37
L    60
      .1
Xo    35
GPIO.  etwarnin    (Fal  e)
GPIO.  et    (TRIG GPIO.OUT)
GPIO.  et    (ECHO GPIO.IN)
GPIO.  et    (L  in  GPIO.IN)
GPIO.  et    (R  in  GPIO.IN)
c  rrent_    eed    0
c  rrent_    eed2    0


   For PWM
RW_PIN    18
LW_PIN    13
RW_ENA     16
LW_ENA     11
GPIO.  et    (RW_PIN GPIO.OUT)     Ri  ht Wheel
GPIO.  et    (RW_ENA GPIO.OUT)     Ri  ht Wheel
GPIO.o  t    t(RW_ENA  1)
GPIO.  et    (LW_PIN GPIO.OUT)     Left Wheel
GPIO.  et    (LW_ENA GPIO.OUT)     Left Wheel
GPIO.o  t    t(LW_ENA  1)


  initiali  e PWM
r    GPIO.PWM(RW_PIN 50)     Ar       ent  are   in and fre    enc
r.  tart(0)    Ar       ent i  initial d  t   c  cle  it   ho  ld be 0
l    GPIO.PWM(LW_PIN 50)     Ar       ent  are   in and fre    enc
l.  tart(0)    Ar       ent i  initial d  t   c  cle  it   ho  ld be 0
   l  e_  tart    0
while Tr  e
                  GPIO.o  t    t(TRIG  Fal  e)
                  ti   e.  lee  (.5)     Sho  ld be 2   econd     In  ide or o  t  ide Loo
                  GPIO.o  t    t(TRIG  Tr  e)
                  ti   e.  lee  (0.00001)
                  GPIO.o  t    t(TRIG  Fal  e)
                  while GPIO.in    t(ECHO)      0       FIX THIS
                                    l  e_  tart    ti   e.ti   e()
                  while GPIO.in    t(ECHO)      1
                                    l  e_end    ti   e.ti   e()
                                    l  e_d  ration        l  e_end -     l  e_  tart
                                 di  tance        l  e_d  ration    17150
```

```
                    di  tance    ro  nd(di  tance  20)


                     rint( DISTANCE   di  tance)
                    d  t     di  tance / .5
                    if(d  t     60)
                                d  t     60
                    if(d  t     0)
                                d  t     0
            if (c  rrent_   eed     d  t )
                    for i in ran  e(int(d  t ) int(c  rrent_   eed) 1)
                                c  rrent_   eed -   1
                                e         ath.e   (-    (c  rrent_   eed - Xo))
                                c  rrent_   eed2   (L / (1    e   ))
                    if(c  rrent_   eed   60)
                                c  rrent_   eed   60
                                if(c  rrent_   eed2   10)
                                            r.Chan  eD  t  C  cle(0)
                                            l.Chan  eD  t  C  cle(0)
                                if(c  rrent_   eed2   10)
                                            r.Chan  eD  t  C  cle(c  rrent_   eed2)
                                            l.Chan  eD  t  C  cle(c  rrent_   eed2)
                                ti   e.  lee  (.01)


            if (c  rrent_   eed     d  t )
                    for i in ran  e(int(c  rrent_   eed) int(d  t ) 1)
                                 rint c  rrent_   eed
                                c  rrent_   eed     1
                                e         ath.e   (-    (c  rrent_   eed - Xo))
                                c  rrent_   eed2   (L / (1    e   ))
                    if(c  rrent_   eed   60)
                                c  rrent_   eed   60
                    if(c  rrent_   eed   0)
                                c  rrent_   eed   0
                                r.Chan  eD  t  C  cle(c  rrent_   eed2)
                                l.Chan  eD  t  C  cle(c  rrent_   eed2)
                     rint c  rrent_   eed2
            if (GPIO.in    t(L  in)     Tr  e)
                     rint( STOP LEFT WHEEL    )
                    r.Chan  eD  t  C  cle(0)
                     ti   e.  lee  (.25)
            el  e
                    r.Chan  eD  t  C  cle(c  rrent_   eed2)
            if (GPIO.in    t(R  in)     Tr  e)
                     rint( STOP RIGHT WHEEL    )
                    l.Chan  eD  t  C  cle(0)
                     ti   e.  lee  (.25)
            el  e
                    l.Chan  eD  t  C  cle(c  rrent_   eed2)
                    ti   e.  lee  (.01)


            c  rrent_   eed   d  t
            ti   e.  lee  (1)
tr
   while Tr  e     create an infinte loo   to   ee   the   cri  t r  nnin
      ti   e.  lee  (.001)
       lobal di  tance
       rint  Di  tance    di  tance
      a       0
e  ce  t   e  boardInterr    t
    rint        it
   GPIO.clean    ()
```

D. 1Bill of Materials

Finalized Detailed BOM

Part	Unit Cost	Quantity per car	Cost Per Car
Ra berr Pi 3	34.49	1.00	34.49
Ultra onic Mod le HC-SR04	9.68	0.20	1.94
Motor controler	14.98	0.20	3.00
IR Sen or	10.99	0.20	2.20
Holder for 2 X 18650 with 6 Wire Lead	7.59	0.20	7.59
Acr lic Sheet 12 24 1/8	10.99	0.25	2.75
Cla 10 32GB Micro SD card	13.14	0.50	6.57
TT Motor w/ Encoder (6V 160RPM 120 1)	7.40	2.00	14.80
D80 Silicone Wheel For TT Motor	1.66	2.00	3.32
Ca tor Wheel 8 Dia eter	3.07	1.00	3.07
Sa n 25R INR 18650	3.90	2.00	7.80
Philli head crew M3 30	4.20	0.04	0.17
Medi -Stren th Cla 8 M3 0.5 Thread (ac of 100)	0.88	0.04	0.04
M2.5 He Bra S acer/Standoff it	8.59	0.13	1.15
M2.5 Screw 0.45 Thread 8 Lon	3.93	0.04	0.16
Steel He N t M2.5 0.45 Thread	1.94	0.10	0.19
er Wire 1M (Connector Set)	18.99	0.10	1.90
N ber 4 Wa her	1.40	0.10	0.14
4-40 Screw Thread Si e 5/8 Lon	8.96	0.03	0.26
4-40 N t Thread Si e	0.89	0.08	0.07
		Total Car Cost:	**$91.59**
Accesories			
4 Ba 18650 Batter Char er	$15.00	0.5	7.50
		Total Car Cost w/ Accessories:	**$99.09**
		20 Car Fleet Cost:	**1,981.82**

D.2 All Expenses

All Purchases For Senior Design

Item	Supplier	Unit Cost	Number	Total Cost
Raspberry Pi 3B+	Amazon	$ 38.40	2	$76.80
Paspberry Pi Zero W starter kit	Amazon	$ 32.99	2	$65.98
2 Pack of Class 10 32GB Micro SD cards	Amazon	$ 16.30	1	16.3
Smraza 5pcs Ultrasonic Module HC-SR04	Amazon	$ 9.68	1	9.68
Raspberry Pi Power Supply kit	Amazon	$ 6.99	2	13.98
Motor controlers	Amazon	$ 14.98	1	$14.98
IR Sensor	Amazon	$ 10.99	1	10.99
Clear Cast Acrylic Sheet, 6" x 12" x 3/32"	McMaster	$ 2.72	1	2.72
Clear Cast Acrylic Sheet, 6" x 12" x 1/8"	McMaster	$ 5.37	1	5.37
HDPE Sheet, 12" x 12" x 1/8"	McMaster	$ 4.44	1	$4.44
Scratch-Resistant Acrylic, 12" x 12" x 3/16"	McMaster	$ 8.28	1	8.28
Raspberry Pi heatsinks	Amazon	$ 7.99	1	7.99
Raspberry Pi Camera	Amazon	$ 13.49	1	13.49
Rasberry Pi 3	Amazon	$ 34.49	1	$34.49
Holder for 2 X 18650 with 6" Wire Leads,5 pack	Amazon	$ 7.59	1	7.59
EBL 18650 3.7V Li-ion Batteries w/ Charger	Amazon	$ 20.00	1	20
M2.5 Series Hex Brass Spacer/Standoff kit	Amazon	$ 8.59	1	8.59
Wire kit	Amazon	$ 29.99	1	$29.99
Zinc-Plated, M8 x 1.25 mm Thread (coupler)	McMaster	$ 0.48	1	0.48
Metric Medium-Strength Steel Hex Nuts	McMaster	$ 4.84	1	4.84
Medium-Strength, Class 8, M3 x 0.5 mm Thread	McMaster	$ 0.88	1	0.88
M3 x 0.5 mm Thread, 30 mm Long	McMaster	$ 4.08	1	$4.08
Socket Head Screw, M2 x 0.4 mm Thread, 12 mm	McMaster	$ 4.02	1	4.02
Nuts, M2 x 0.4 mm Thread	McMaster	$ 1.12	1	1.12
Rasberry Pi 3	Amazon	$ 36.58	5	182.9
10 IR Sensor	Amazon	$ 9.98	1	$9.98
2 Pack of Class 10 32GB Micro SD cards	Amazon	$ 13.14	2	26.28
Holder for 2 X 18650 with 6" Wire Leads,5 pack	Amazon	$ 7.59	1	7.59
Smraza 5pcs Ultrasonic Module HC-SR04	Amazon	$ 9.99	1	9.99
Samsung 25R INR 18650	batteryjunction.cc	$ 3.90	14	$54.60
TT Motor with Encoder (6V 160RPM 120:1)	Mouser	$ 7.40	16	118.4
D80mm Silicone Wheel For TT Motor	Robot Shop	$ 1.66	14	23.24
Castor Wheel 8mm Diameter	Robot Shop	$ 3.27	3	9.81
Phillips head screws M3 30 mm Long	mcmaster	$ 4.20	1	$4.20
Clear Acrylic Sheet 12" x 24" x 1/8"	mcmaster	$ 10.99	2	21.98
Motor controlers	Amazon	$ 14.98	1	14.98
Samsung 25R INR 18650	batteryjunction.cc	$ 3.90	4	15.6
Smart Rechargeable Battery Charger	Amazon	$ 15.00	2	$30.00

Note: Team 18's goal under the advisment of Dr. Phillip Dames is to produce as many cars as our $1000 budget allows. According to this budget we can fully fabricate 9 cars.

Subtotal: **$896.63**

Budget Remaining: **$103.37**

E.1 User Guide

E.2 Troubleshooting

Citations

Stern, R.E., Cui, S., Monache, M.L., Bhadani, R., Bunting, M., Churchill, M., Hamilton, N., Haulcy, R., Pohlmann, H., Wu, F., Piccoli, B., Seibold, B., Sprinkle, J., & Work, D.B. (2018). Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *ArXiv, abs/1705.01693*.

https://www.semanticscholar.org/paper/Dissipation-of-stop-and-go-waves-via-control-of-Stern-Cui/3c1f6f2ae1d37102dd8b26af58569a7669920dda

Temple University. (2014, October 8). Using autonomous vehicles to improve traffic flow. *ScienceDaily*. Retrieved November 7, 2019 from www.sciencedaily.com/releases/2014/10/141008131256.htm

Seibold, Benjamin. Research web page. https://math.temple.edu/~seibold/research/traffic/

NSF. Award Abstract #1446690. "CPS: Synergy: Collaborative Research: Control of Vehicular Traffic Flow via Low Density Autonomous Vehicles" 2015-18

NSF. Award Abstract #1007899. "Collaborative Research: Phantom traffic jams, continuum modeling, and connections with detonation wave theory" 2010-13

RFP, Senior Design Fall 2018. Request for Senior-Design Project Proposal. "Title: Benchtop Testbed for Autonomous Vehicle Research."

Mechanic Base. https://mechanicbase.com/cars/average-car-length/

Cars.com.https://www.cars.com/articles/which-cars-have-self-driving-features-for-2018-1420699785509/

NHTSA. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

PowerStream.com. https://www.powerstream.com/Wire_Size.htm

Yang, Ye, Pan, Ma. The Implementation of S-curve Acceleration and Deceleration Using FPGA. Telkomnika, Vol 11, No 1. Pp 279 - 286. January 2013