MECHATRONICS AND PROPULSION CONTROL SYSTEM DESIGN FOR THE SPARTAN SUPERWAY

First Semester Report

Presented to

The Faculty of the Department of Mechanical Engineering

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Colin Ilas, Jr.

December 15, 2020

SAN JOSE STATE UNIVERSITY

The Undersigned Committee Approves

Mechatronics and Propulsion Control System Design

For the Spartan Superway

Of

Colin Ilas, Jr.

APPROVED FOR THE DEPARTMENT OF MECHANICAL ENGINEERING

Dr. Burford Furman, Committee Chair	Date
Dr. Saied Bashash, Committee Member	Date
Mr. Ron Swenson, Committee Member	Date

ABSTRACT

This report details the work completed in the first semester of creating a mechatronics and propulsion control system for the Spartan Superway. A mechatronics system consisting of motors, servos, sensors, and other components will be created for the Spartan Superway to provide a platform for continued testing of different control systems. In addition, a propulsion control system for the motors will be created and optimized to meet certain design goals.

A propulsion control system was created for the 10-meter scale bogie system with plans to upscale to a full-scale model once all hardware has been retrieved. Its mechatronic system consists of generic 83mm brushless DC hub motors, a Flipsky VESC, and an ESP32 microcontroller. The brushless DC motors found on the 10-meter scale bogie were characterized using steady-state sampling and retrieving values using VESC Tool, an open-source brushless DC motor control software for the electronic speed controller. Using the characteristic values obtained, a Simulink model of the system was made in MatLAB using the Simscape module. Varying load conditions, acceleration curves, and were applied in simulation to tune the velocity PID controller to meet our design goals of rise time under 1.5 seconds, steady-state velocity of 2m/s, instantaneous power draw of less than 100W, and steady-state error of less than 2%.

As a result, we were successfully able to retrieve PID gain values that allowed us to achieve these goals. These values can be imported into VESC Tool which will implement our PID controller to the motors. Further testing includes testing the motors using a dynamometer and implementing the controller onto the 10-meter scale bogie for on-track testing.

ACKNOWLEDGEMENTS

I would like to thank my committee for providing guidance throughout this project. Specifically, I would like to thank Dr. Burford Furman and Mr. Ron Swenson for not only providing this research opportunity but for also allowing me to continue my progress on the Spartan Superway. When I first joined the team during my undergraduate studies, I felt that the project had a lot of potential. Being able to learn from their years of progress and share my own work has been a great opportunity for me to practice my education and gain more experience throughout the past few years. In addition, I would like to thank Dr. Saeid Bashash for providing me with the knowledge of mechatronics and control systems not only through my coursework but through his guidance on this project. My choice to become a mechanical engineer has been driven by my desire to create the newest robotic technologies, and Dr. Bashash has provided me the tools I need to make that dream a reality.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The Spartan Superway seeks to bring a new paradigm to public transportation as we know it. Started in 2012, the Spartan Superway is a research initiative founded by Dr. Burford Furman of San Jose State University and Mr. Ron Swenson of Swenson Development. It aims to bring forth a new mode of public transportation by way of its personal "podcar" pedestrian vehicle and elevated guideway design designed to eliminate carbon emission, improve travel efficiency, and increase pedestrian safety. Through its use of solar panels, it aims to not only have a zero-carbon footprint but be able to provide energy back to the grid. With its individual podcar model, each destination is routed through its unique guideway system by personalizing the public transportation experience and optimizing travel time. By nature of its ascended infrastructure, the Spartan Superway separates the distance between other vehicles and pedestrians, avoiding traffic and increasing overall safety. As demonstrated in figure 1, all these technologies seek to redefine public transportation as we know it.

Figure 1: Spartan Superway demonstration prototype. This prototype depicts the many technologies the Spartan Superway plans to implement such as its solar panels, personal pedestrian podcar, and an elevated infrastructure.

The Spartan Superway seeks to be an autonomous system fully capable of navigating through its guideway system without the need for a driver, otherwise known as an automated transit network or ATN for short. Along with being autonomous, ATN's typically feature a complex guideway system to allow every podcar to reach its destination more efficiently without having to make constant stops. This is made possible with the combination of electrical, electronic, and mechanical systems, otherwise known as a mechatronic system. These mechatronic systems take in data from mechanical sensors, user inputs, and other sources and utilize software to compute a desired output to be translated by mechanical or electrical means.

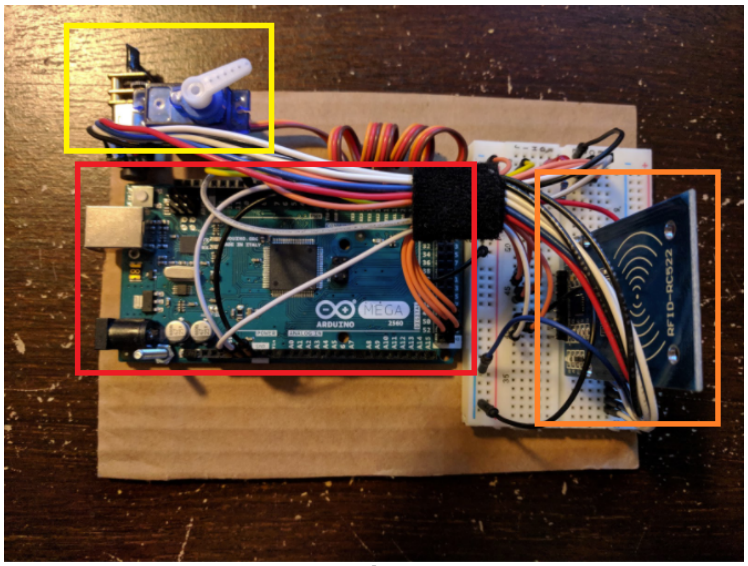

Figure 2: Mechatronic system of the 1/12th scale model. A mechatronic system combines electronic and mechanical hardware, as seen by previous Spartan Superway models which utilized an Arduino microcontroller (red), electronic RFID sensor (orange), and mechanical motors and servos (yellow).

In the case of the Spartan Superway, mechanical and electronic sensors are used to track the position, velocity, and acceleration of a podcar, and a microcontroller takes in this data to calculate how much power a motor needs to meet its required goals. These goals are defined by a control system which dictates the actions of the system at any time. A control

system tells the podcars when they should start, stop, switch guideways, and so many other functions through the components of a mechatronic system. As seen in figure 2 below, a control system is the brains of a mechatronic system and is what will be used to autonomously control the podcars of the Spartan Superway.

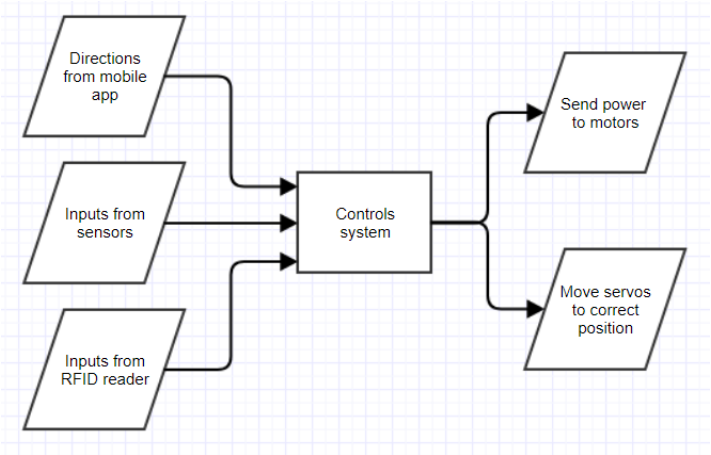

Figure 3: Control system layout of the Spartan Superway. The control system takes in user input and sensor data and outputs directions to the motors and servos so that the podcar can navigate effectively throughout the guideways.

The Spartan Superway is a project that has been conceptualized on many different scales. A small-scale design, known as the $1/12^{th}$ scale model, allowed for testing of the ATN concept on a closed railway system with multiple loops. This model, shown below in figure 4, demonstrated the ability of these podcars to navigate autonomously after being given user destination input. In addition, the system proved the feasibility of the podcars to switch between different railway loops.

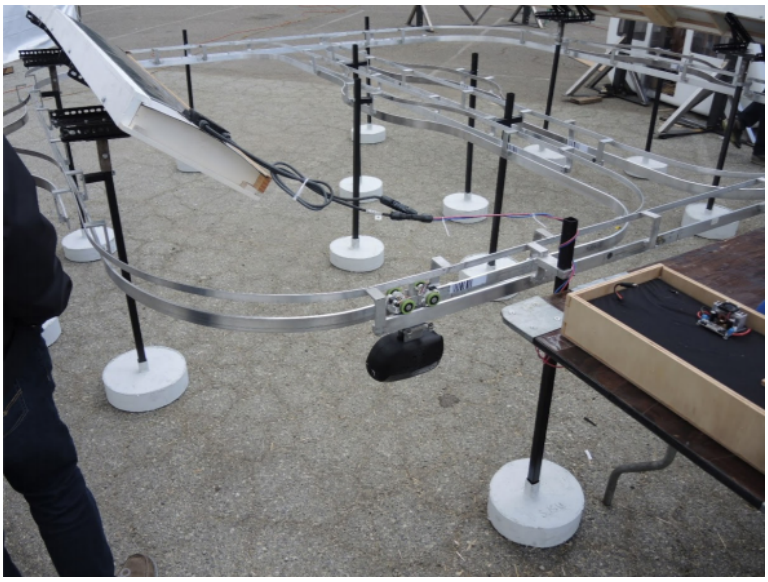

Figure 4: 1/12th scale podcar and guideway system. A small-scale prototype of the Spartan Superway allows for testing and analyzation to achieve full-scale optimization.

In addition to the small-scale model, past half-scale and full-scale model have also been created. As seen in figure 5 below, the full-scale model features a splitting guideway which helped test the guideway switching components of the bogie. In figure 6, an early half-scale model was created to experiment with the possibility of picking up passengers from street level.

Figure 5: Full-scale Spartan Superway model. This full-scale model allowed the mechatronics team to create a switching mechanism to allow the podcar to travel along different railways.

Figure 6: Half-scale Spartan Superway Model. The half-scale Spartan Superway model featured a scissor mechanism that allowed the podcars to pick up passengers on a street level.

Currently, the Spartan Superway is focused on another small-scale model known as the 10-meter scale model. This model seeks to achieve the same goals as the older 1/12th scale model, allowing for testing and optimization of the navigation and propulsion control systems on a small scale. However, this model brings improvements that were not possible on the 1/12th scale model. As seen in figure 7, the 10-meter mechatronic system utilizes brushless DC (BLDC) motors which provide more torque, velocity, longer lifetime, and better position and velocity tracking with its internal sensors. With the previous 1/12th scale model, the motors would quickly die out due to its brushed design.

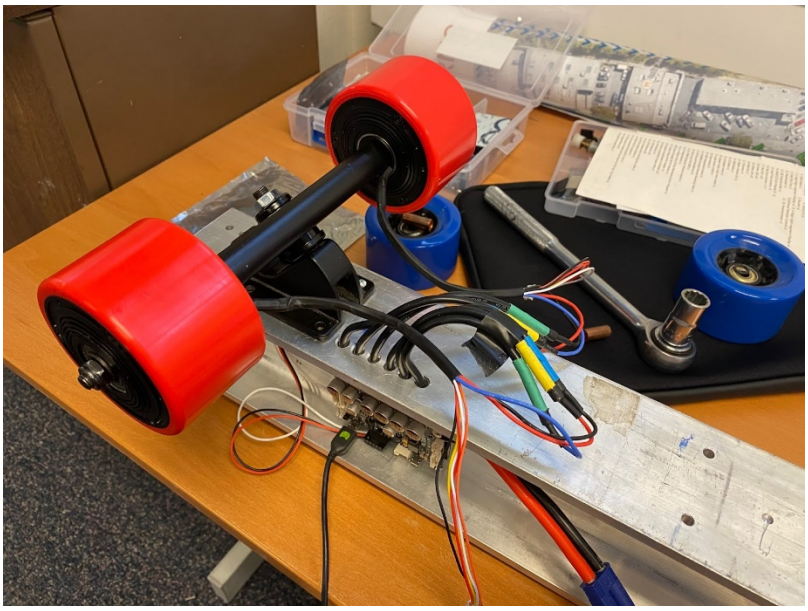

Figure 7: Brushless DC (BLDC) motors on the 10-meter scale bogie. Brushless DC motors provide many benefits over the older brushed DC motors used on the 1/12th-scale model.

LITERATURE REVIEW

Global warming and climate change

Among the many problems we are facing today, global warming is certainly one of the largest detriments to our future. Climate change has immensely impacted the world by destroying ecosystems and endangering many species of animals. Not only are animal ecosystems destroyed, but humans are also affected by the increase in natural disasters. Storms, hurricanes, heat waves, and forest fires are all causal problems of increased air and sea temperatures (Garcia, 2016, p. 83). In California alone, millions of acres of land have been lost due to the increased amount of forest fires due to climate change. The Philippines is currently dealing with high amounts of category 4 and 5 typhoons in the last few months. With this, we can see that climate change is a global problem and affects everyone and everything. Through

this, we can see that pollution causes a large amount of problems for the earth and a solution is needed now more than ever.

Carbon dioxide is one of the many greenhouse gasses that are the main cause of global warming, and transportation accounts for a large majority of carbon dioxide emission. In the United States, a large part of the transportation sector is made up of personal vehicles, with public transportation barely making up a quarter of the total at 26% according to the Department of Transportation (2018). In fact, transportation has surpassed the power sector in carbon dioxide emissions in 2016 (Ezike, 2019, pp. 19-20). With electric vehicles being a newer innovation, the fact remains that the majority of carbon dioxide emission lies within personally owned vehicles.

Personal rapid transit and automated transit networks

Personal rapid transit (PRT) is a mode of public transportation that utilizes small vehicles, commonly known as podcars, to transport pedestrians through a complex railway infrastructure. Typically designed to fit a small group, these podcars traverse on a network of railways designed for non-stop, point-to-point travel. The concept of these types of vehicles can be seen over 50 years ago through the work Stanford University's Howard Ross and his personalized transit capsule powered by a compressed air railway system (Attoh, 2019, pp. 84-85). Today, with operational systems in London, South Korea, and West Virginia, personal rapid transit has proven to be a viable and successful method of public transportation.

Also known as automated transit networks (ATN), the concept of driverless podcars is built upon the idea that these podcars must be fully autonomous. The system will determine the best route to take, avoiding any stops and bringing the user from point A to point B efficiently. In addition, the podcars should be available for 24 hours a day, 7 days a week to revolve around the availability of the user and not a fixed schedule. Lastly, the system will operate upon its own

exclusive infrastructure to avoid traffic congestion and prevent pedestrian accidents (Furman, 2014, pp 7-8). Through these ideals, an automated, personal rapid transit can be made.

OBJECTIVES

The objective of this project is to create both a mechatronic system and a propulsion control system for the full-scale model of the Spartan Superway. The mechatronic system will require obtaining motors to propel the podcars along the guideway, a motor driver to power the motors, sensors to analyze the speed and position of the podcars, and a microcontroller to autonomously control the podcar. A successful full-scale mechatronic system will provide the Spartan Superway a platform to begin creating the various control systems required for an autonomous system.

A propulsion control system will also be created for the full-scale mechatronic model. This controller will power the motors to ensure that their position and velocity meet the desired values while also optimizing the power efficiency and longevity of the motors. A typical PID controller will be utilized to allow for the system to automatically correct for any discrepancies between the desired and actual values. Two PID controllers will be created for this model: a velocity controller and a position controller. This will allow for flexibility for future groups in creating the navigation control system where position control might be better suited in cases like platooning.

METHODOLOGY

The first step of beginning this project is to familiarize myself with the current status of the 10-meter mechatronic system including the electronics being used and how they are being controlled. The current remote internet-of-things (RIoT) team has been working on this project for the previous two years and have gotten a lot of progress done. In the figure below,

we can see how the project has been split into separate systems. For this project, I will be working on the propulsion system located at the top left of the figure.

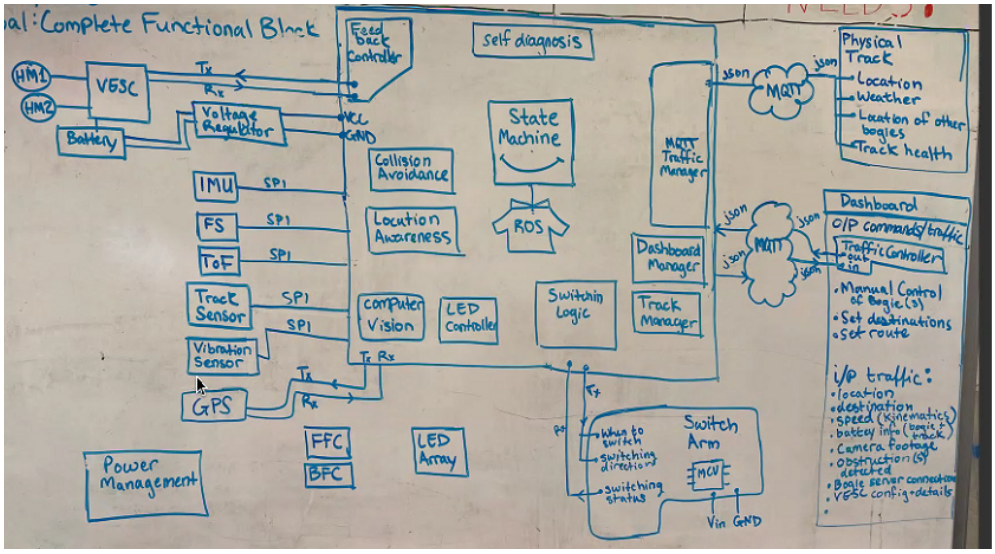

Figure 8: ESP32 subsystem breakdown. This diagram illustrates the different systems at work on the 10-meter track's microcontroller.

The next step of this project is to utilize VESC Tool, an open-source software used to analyze and program the electronic speed controllers for the brushless DC motors shown in figure 9. VESC Tool will allow us to run a variety of tests on the BLDC motors, such as duty cycle or RPM stepping. Through these experiments, we can extrapolate data to be exported into MatLAB for analysis, as seen in figure 10.

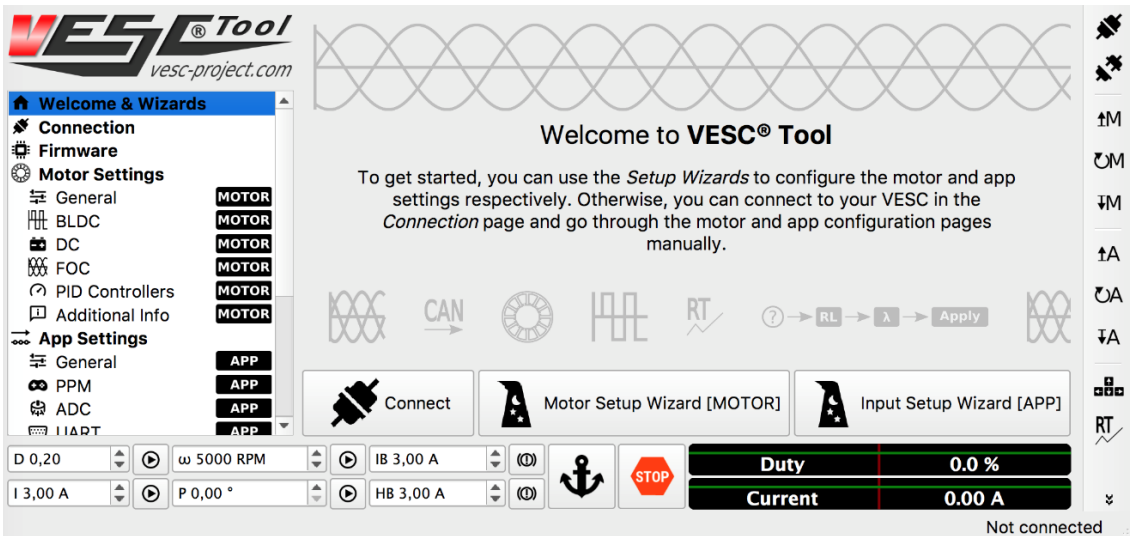

Figure 9: VESC Tool homepage. VESC Tool is an open-source software that provides many tools and conveniences when programming control systems for BLDC motors.

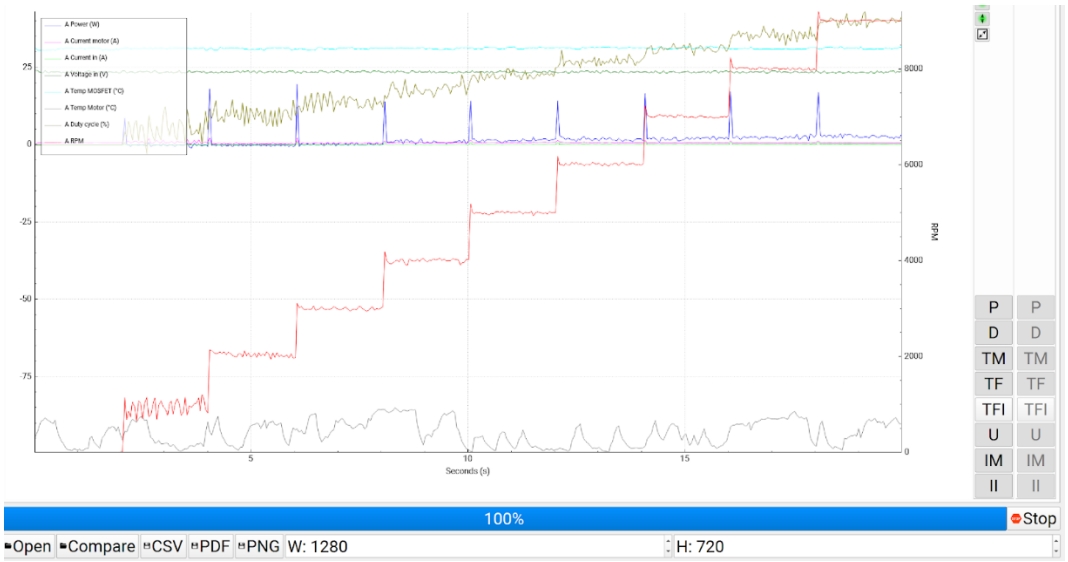

Figure 10: VESC Tool duty cycle stepping data log. VESC Tool has built-in experiments with data logging for testing of the motors and PID controllers. The data log above shows the RPM, voltage, current, power, and other data achieved from our BLDC motor when its duty cycle is incrementally stepped from 0 to 100%.

With this data, we can analyze the motors at steady-state conditions. This is important because it will allow us to calculate the motor characteristic values for use in simulation. Once simulated, we can apply a propulsion PID controller which will allow us to control certain goals like rise time, steady-state error, overshoot, and power draw.

The methodology of creating the propulsion PID controller are as follows:

- Experimentally determine motor characteristic values.
- Solve governing differential equations and create a state-space model in MatLAB
- Utilize MatLAB's Simscape BLDC motor model to simulate different conditions
- Tune PID controller through simulation to ensure design goals are met
- Apply PID values to the electronic speed controller using VESC tool

As seen below, VESC Tool makes it simple to integrate a PID controller into the velocity and position control of the brushless DC motors. This eliminates the need to incorporate any PID libraries in the microcontroller programming.

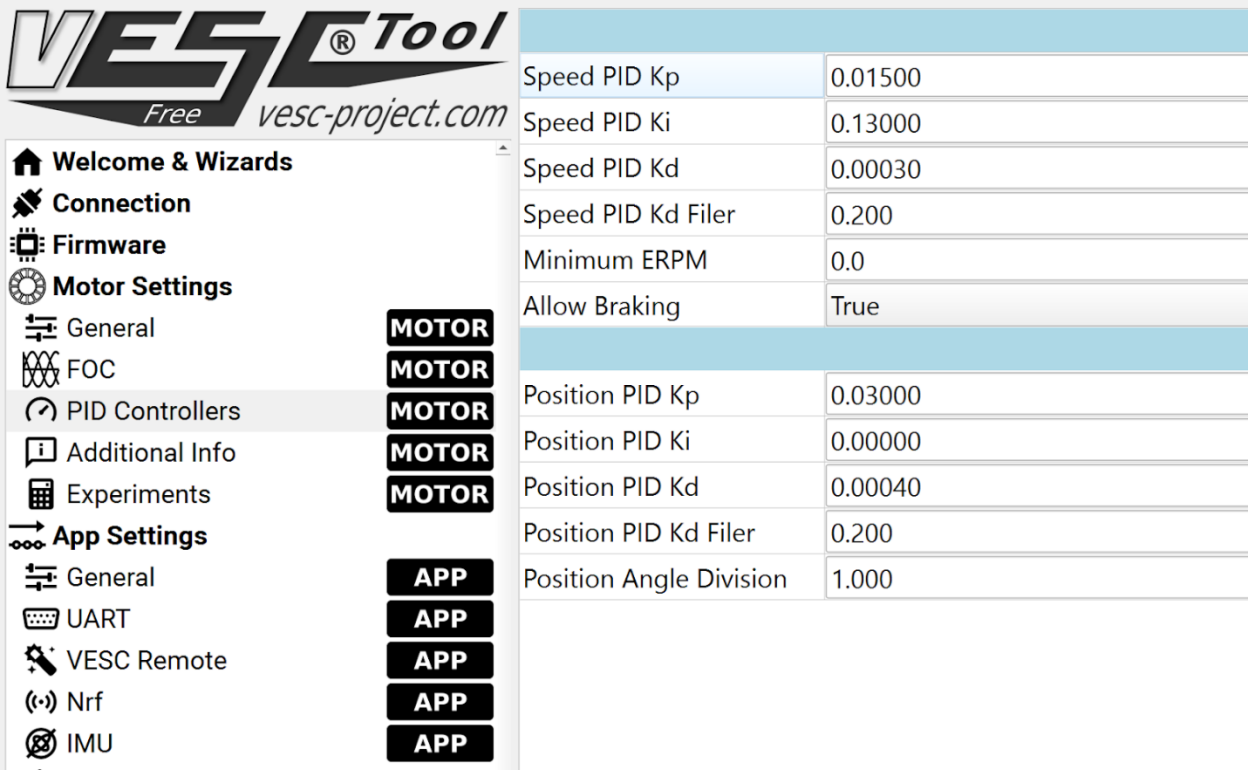

Figure 11: VESC Tool velocity and position PID controller input. VESC Tool allows the user to easily input gain values for a PID controller through its software, simplifying the coding process on the microcontroller side.

Simulating the system will be done using MatLAB's Simulink setup. With the use of Simscape, a visual modeler found in Simulink, we can simulate our exact brushless DC motors and apply different loads, request certain velocities, and analyze different values such as the power draw, how long it takes to reach our requested velocity, and any discrepancies or errors that may occur.

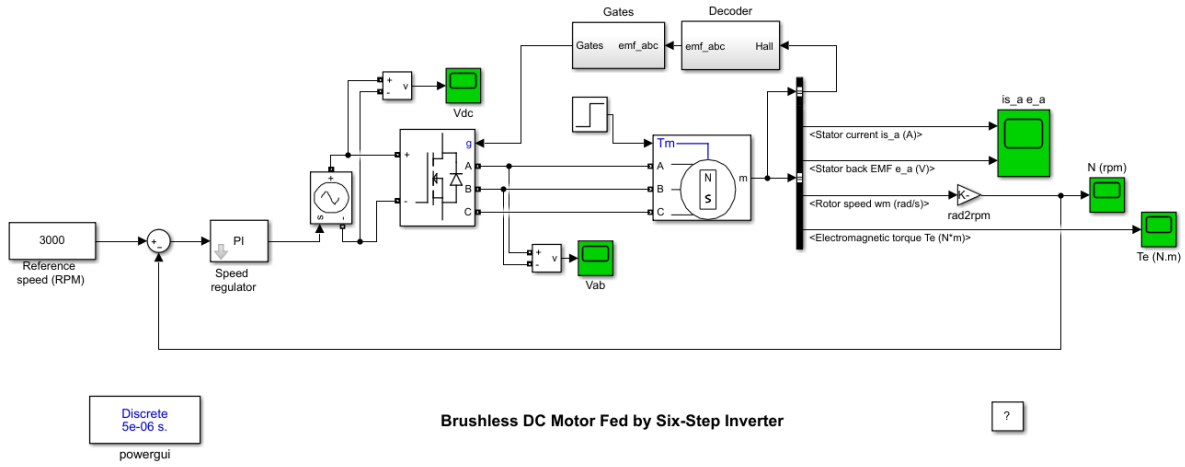

Figure 12: Simscape model of a BLDC motor model. Simscape, a module built into MatLAB's Simulink, comes with an example brushless DC motor model that would allow us to simulate different loads and conditions our motors might encounter and how they would react.

Simscape simply has a block that represents our BLDC motor. By applying the parameters that we achieve through experimental methods, we can accurately simulate our exact motor. Luckily, we are able to achieve some values through VESC Tool and the rest can be achieved using steady-state analysis which will be covered in the experiment setup of this report.

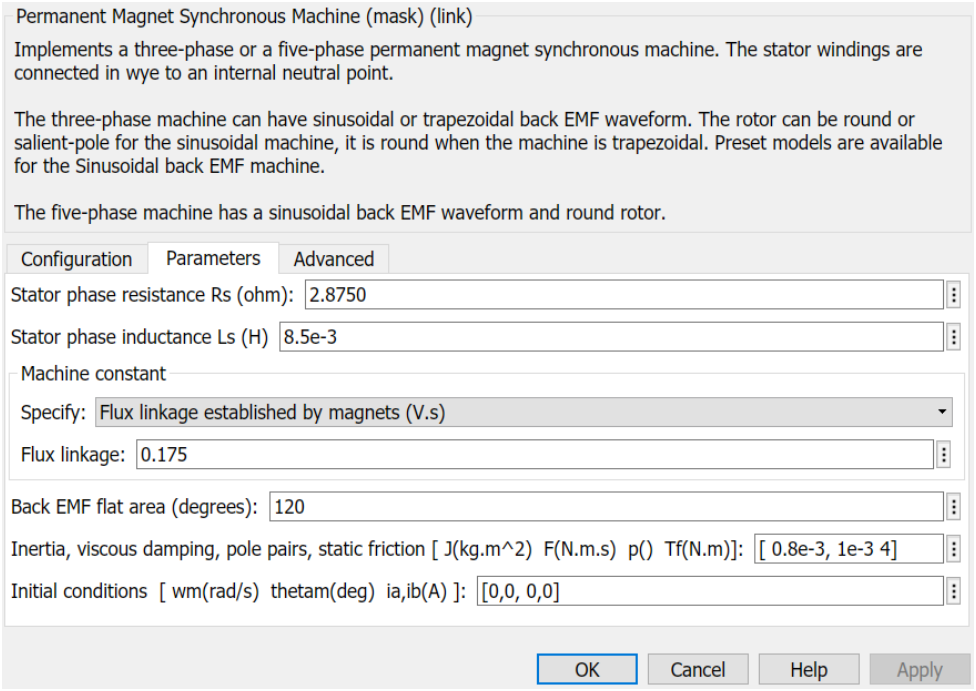

Figure 13: Simscape BLDC motor block diagram parameter input. In Simscape's model, the motor parameters achieved through testing can be applied here and would effectively simulate the motor that we are using.

DESIGN CONCEPT

Design Goals

For our propulsion control system, we are aiming to create a PID controller to meet the following conditions:

- Achieve steady-state velocity under 1.5 seconds
- Maintain an error of less than 5%
- Maintain a speed of 2 m/s
- Minimize instantaneous power draw to under 100W

Design Specifications

The current mechatronics system of the 10-meter bogie utilizes the following components. No changes will be made to the components as they are sufficiently able to meet our design goals as it is.

- ESP32 Microcontroller
- 83mm Brushless DC (BLDC) Hub Motors (qty. 2)
- Flipsky Dual FSESC6.6 Electronic Speed Controller
- 22.2V 6S LiPo battery

No changes will be made to the components as they are sufficiently able to meet our design goals as it is.

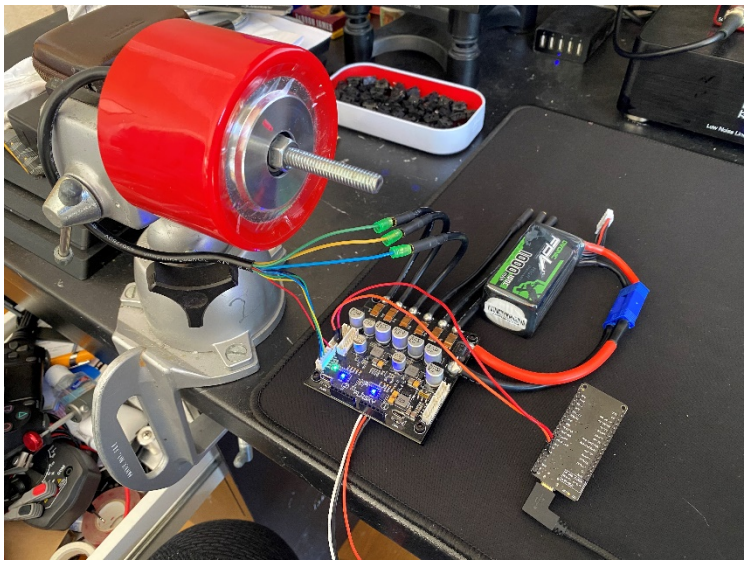

Figure 14: Mechatronics system test bench. The 12-meter bogie will be utiziling an ESP32 microcontroller to handle decision making, while the brushless propulsion motors will be controlled using a Flipsky VESC.

Design Description

Brushless DC motors By accurately modeling the brushless motor in Simulink, we will be able simulate differing torque loads on the motor and different acceleration and deceleration profiles. Typical simulations would include acceleration and deceleration plots,

steady-state response to varying loads, and energy efficiency plots. Through this, we will be able to tune our PID to controller to meet various requirements.

A simple script that would allow us to communicate between the Flipsky VESC and the ESP32. The current RIoT (remote internet-of-things) team has been able to control the motors using UART communication however were limited to only duty cycle control. Ideally, RPM control would allow for more complex mechatronics control in cases such as setting individual wheel speed through a turn.

ANALYTICAL MOTOR MODELING

Theory and Calculations

A brushless DC motor contains 3 stators, as seen in figure 15, compared to a typical DC motor which only contains 2. The differential equations of the electrical system must then be separated into 3 different equation using components a, b, and c to represent the three different stators.

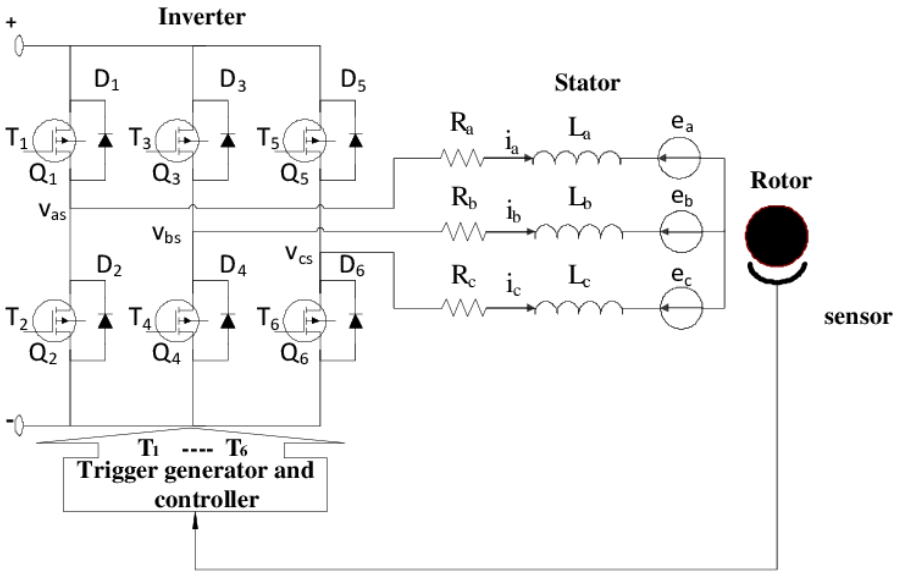

Figure 15: Electric schematic of a brushless DC motor and controller (Lu, 2011). A BLDC contains 3 stators that are switched between using a 3-phase inverter. For our purposes, we can produce three KVL loops for each stator and labeling them as components a,b, and c.

With this, the governing equations for a brushless DC motor are calculated as:

$$\begin{aligned} V_a &= i_a R + L\dot{i}_a + e_a \\ V_b &= i_b R + L\dot{i}_b + e_b \\ V_c &= i_c R + L\dot{i}_c + e_c \end{aligned} \qquad (1)$$

Next, we can analyze the back EMF constant of the brushless DC motors. If we look at the physical representation, the stators are separated by an angle of 2π/3 radians. Thus, our resulting back EMF equations will be separated by this angle.

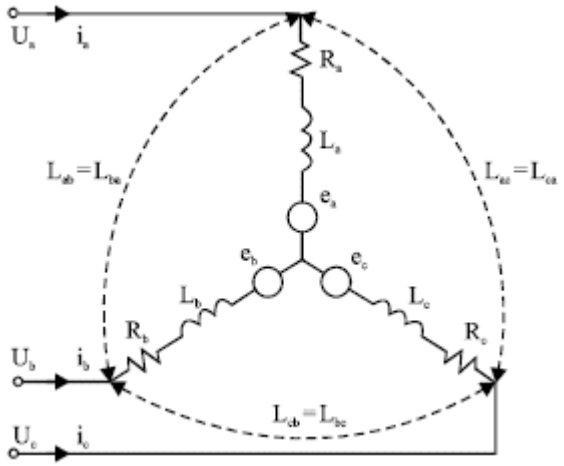

Figure 16: BLDC physical stator layout (Lu, 2011). The 3 stators in a BLDC motor are physically separated by 2π/3 radians, thus we can calculate the three separate back EMF functions for each loop being separated by that angle.

The governing equations for a brushless DC motor are calculated as:

$$\begin{aligned} V_a &= i_a R + L\dot{i}_a + e_a \\ V_b &= i_b R + L\dot{i}_b + e_b \\ V_c &= i_c R + L\dot{i}_c + e_c \end{aligned} \qquad (1)$$

$$\begin{aligned} e_a &= K_b f(\Theta_e)\dot{\Theta} \\ e_b &= K_b f(\Theta_e - 2\pi/3)\dot{\Theta} \\ e_c &= K_b f(\Theta_e + 2\pi/3)\dot{\Theta} \end{aligned} \qquad (2)$$

where:

V = terminal voltage (V)
i = stator current (A)
e = back EMF (V)
R = armature resistance ()
L = armature inductance (H)
a,b,c = component of the 3 phases
K_b = back EMF constant (V/rad-s)
$f(\Theta_e)$ = trapezoidal function based on rotor angle (rad)
$\dot{\Theta}$ = angular velocity of the rotor (rad/s)

The torque produced by the BLDC motor can be described as:

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\dot{\Theta}} \qquad (3)$$

where:

T_e = electromagnetic torque (N-m)

Under ideal conditions, the electromagnetic torque will be equal to the total torque acting upon the motor, which we calculate as:

$$T_e = T_L + T_F + J\ddot{\Theta} + c\dot{\Theta} \qquad (4)$$

where:

T_L = load torque (N-m)
T_F = internal Coulomb friction (N-m)
J = rotor inertia (kg-m^2)
$\ddot{\Theta}$ = rotor acceleration (rad/s^2)
c = rotor viscous damping (Nm-s/rad)

These equation dictates the overall electrical system as a function of the mechanical load applied. Using the Laplace Transform, we can create a state-space model to easily output a desired set of variables with a certain set of inputs, known as a MIMO (multiple in, multiple out) system.

In addition, the brushless DC motor model can be created using Simulink. By means of Simscape, a visual electronic and mechanical system modeler in Simulink, we can easily define the motor characteristics achieved and simply simulate different scenarios without having to apply conditions to a physical model.

Description of the design

The control system will be implemented by an ESP32 microcontroller which will define the logic involved in directions handling, railway switching, and other various functions. Propulsion will be handled by the Flipsky VESC whose PID controller will be

defined by the results of our simulation. Instructions will be sent to the VESC through the UART protocol.

EXPERIMENTAL PROCEDURE

Obtaining Motor Parameters

Before a simulation can be conducted, we must first achieve the motor characteristic values of the brushless DC motors. These values are used to solve the governing differential equations which will be used to simulate our system. Because a datasheet was not provided with our motors, we must observe the motor behavior at differing steady-state RPMs and data fit a linear line to the resulting plots. The equation of this line will give us the missing motor characteristics.

First, the values of resistance, inductance, and flux linkage can be easily extracted from the VESC Tool as seen in figure 17. Using it, we found the following values for our motor:

- R = 0.2464 ohm
- L = .000156 Henry
- Flux linkage = .00813 Weber

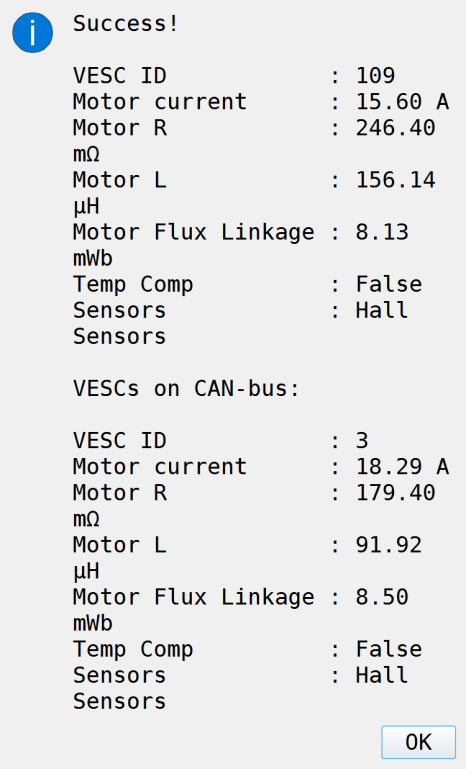

Figure 17: VESC Tool BLDC motor characteristics. VESC Tool can automatically retrieve some motor parameters during its calibration process. It runs the motor back and forth and calculates these values for resistance, inductance, and flux linkage. It also identifies what types of sensors the motors utilize.

Next, the motor was run at three different rotational velocities using the experiment functions found in VESC tool. During these times, the average current and voltage was measured using a multimeter. With this data, we can plot the data on a chart and fit a linear line to the plots to give us our motor characteristic values.

To simplify the problem, we can ignore the a, b, and c components of the different stators in equations 1 and 2 and treat the problem as if we were dealing with normal DC motors. This is because we can assume that the motor is symmetrical, thus the resistances and inductances are similar. We also assume that no Eddy currents are present due to ideal conditions (Cham, 2014). For our application, most of these component data will be handled

by the Flipsky VESC and does not impact the results. In this, our final electromechanical equations are similar to that of a normal DC motor:

$$V = L\dot{i} + Ri + K_b\dot{\Theta}$$
$$J\ddot{\Theta} + c\dot{\Theta} = K_t i + T_L + T_F \quad (5)$$

If we evaluate the first KVL equation from (5) in steady-state conditions, there will be no derivatives to evaluate. We can then solve for the value of the back EMF constant by plotting voltages at different steady-state velocities in figure 18:

$$V = K_b\omega + iR \quad (6)$$

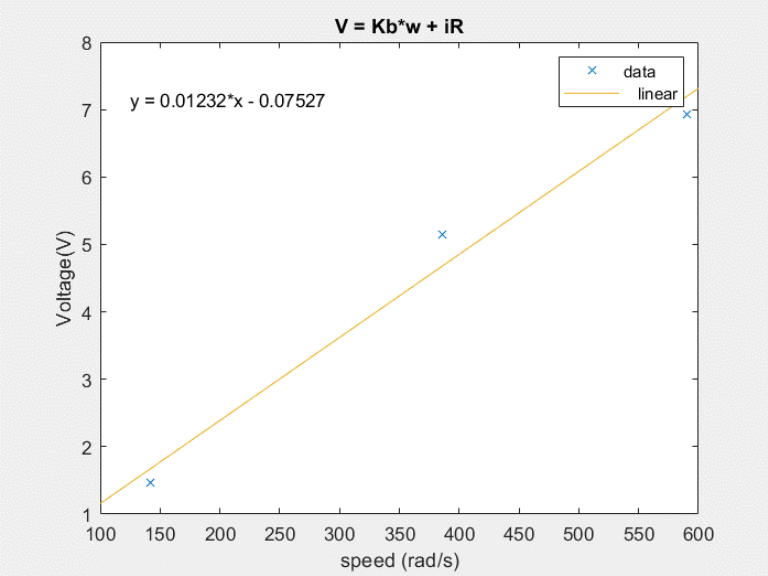

Figure 18: Back EMF constant linear plot. After plotting three steady-state results of speed vs. voltage, we can achieve the back EMF constant by applying a linear data fit line.

From the plot above, we can see that 0.01232 is the value of K_b for the steady-state equation. Because we are dealing with a hub motor with no losses from a drivetrain, this value also represents K_T.

The next plot, figure 19, compares the torque of the motor at different velocities. In the same fashion, we can remove the differentials from the second equation of (5). The steady-state equation for torque in this scenario is:

$$\tau = c\omega + T_F \qquad (7)$$

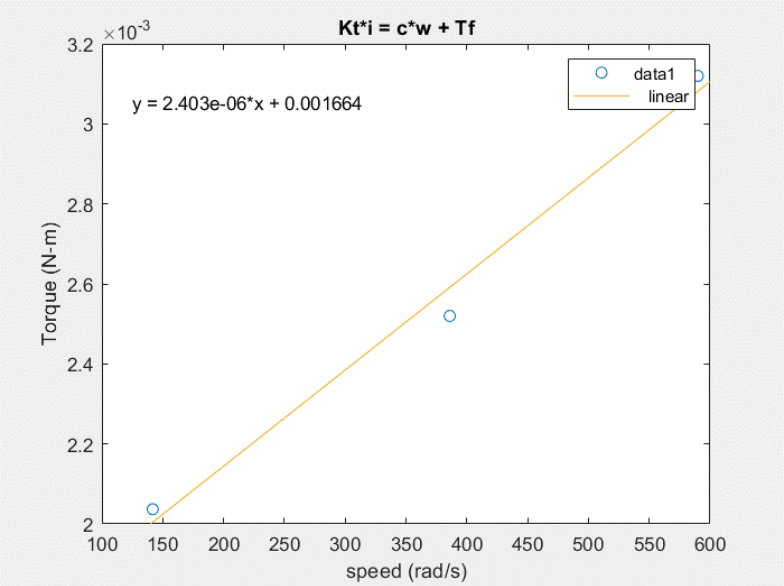

Figure 19: Viscous damping and internal friction plots. By repeating the previous steps but plotting the steady-state velocity vs. torque, we can calculate the viscous damping of the motor and the internal friction.

With this, we can compare our linear fit line to the original steady-state equation and extract the missing motor characteristics. From the tests applied above, we were able to achieve the following motor parameters:

R (ohm)	0.2474
L (H)	1.56e^-4
Flux linkage (Wb)	8.13e^-3
Kt/Kb	.01232
C (Nm-s/rad)	2.403e^-6

Tf (N-m)	.001664
J	1.95e^3

Table 1: Motor characteristics obtained from VESC Tool and experimental procedures. Listed here are the resulting values that will be applied to our Simscape simulation.

Simulink Simulation

With these values, we are finally able to create a Simulink simulation of our model. Thankfully, Simscape comes included with a BLDC motor model. In this, we can change the values of the motor characteristics we just calculated and observe many different points throughout the system as seen in figure 20. For this, I will be closely watching the rotor speed, power draw, and error. By adjusting the PID controller values these factors will become manipulated, and with the right tuning the design goals should be met.

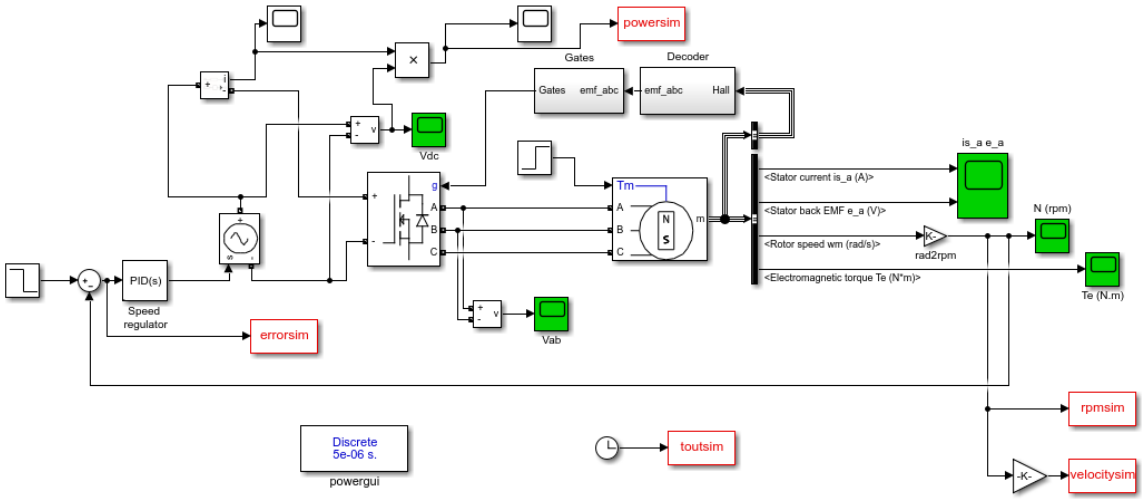

Figure 20: Complete Simscape model with data ouputs and motor parameters inserted. Several blocks were added to the Simscape model to allow us to analyze power draw, error, and velocity.

SIMULATION RESULTS

Acceleration/Deceleration

For the first design goal, we wanted the system to reach its desired velocity within 1.5 seconds. This applies to both acceleration and deceleration, and from the figure 21 and 22 below we have achieved this.

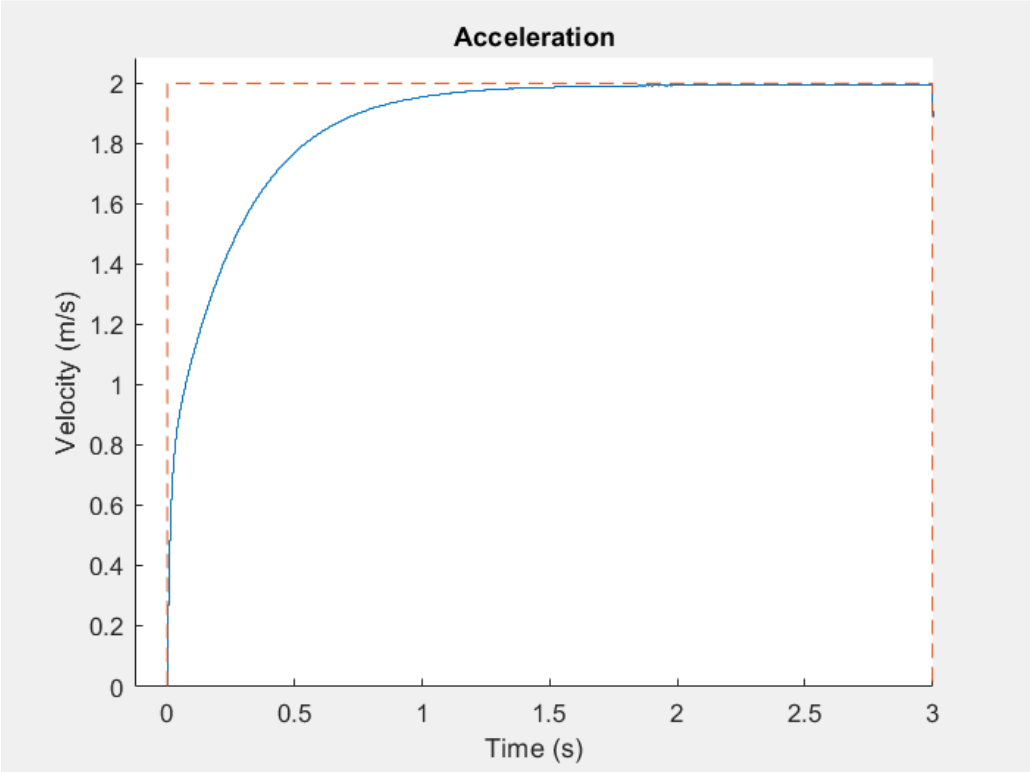

Figure 21: Instantaneous acceleration response to 2 m/s. The PID controller allows our motor to reach 2m/s under 1.5 seconds without any overshoot.

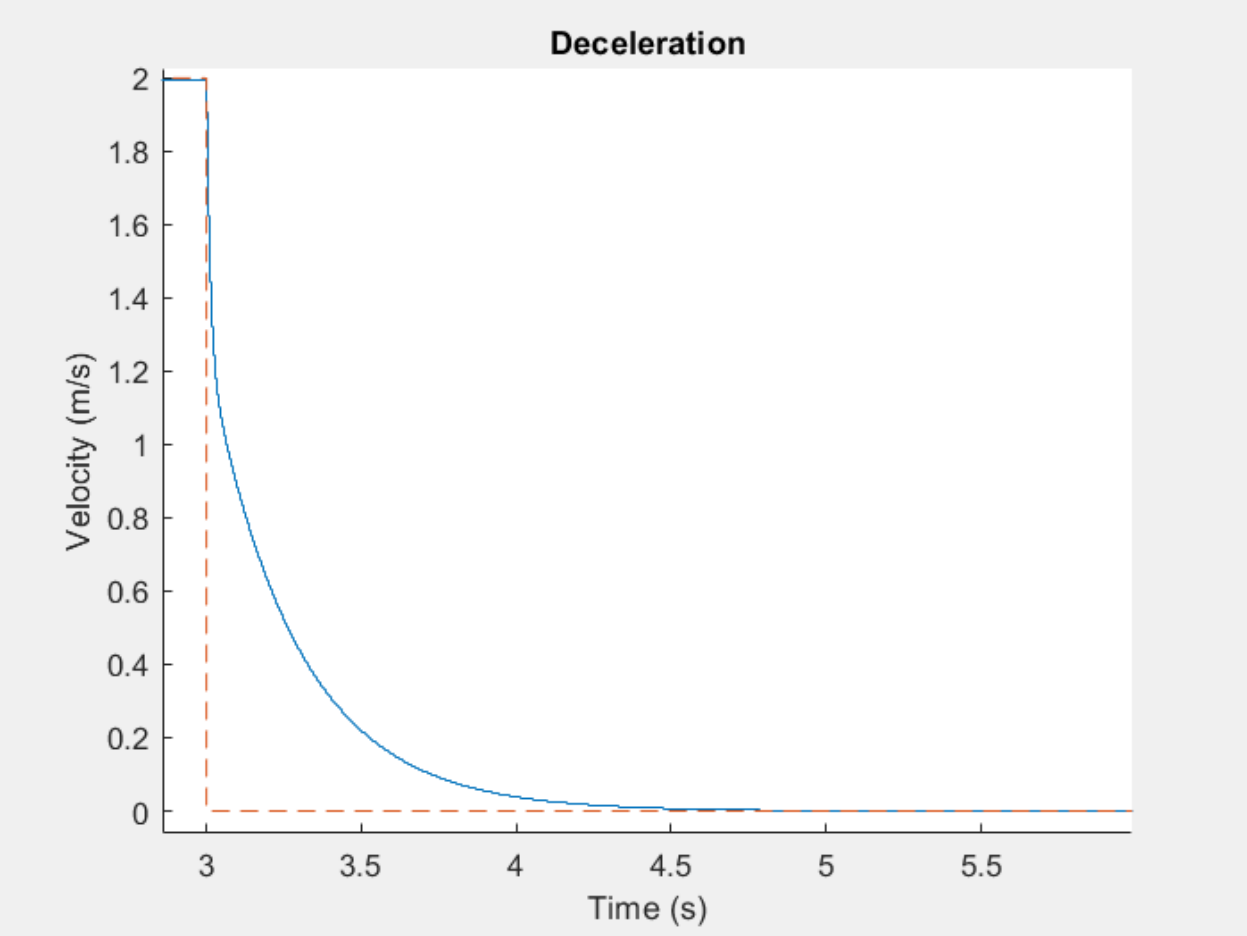

Figure 22: Instantaneous deceleration from 2m/s to 0m/s. Just like acceleration, we are able to decelerate in under 1.5 seconds without any undershoot.

Instantaneous/Steady-State Power Draw

In terms of power, we aimed for an instantaneous power of no more than 100W and a steady-state power draw of less than 10W. These goals were set to ensure the longevity of the Flipsky VESC and BLDC motors as well as minimizing the load on the batteries. As seen by our plots below in figure 23, we were able to satisfy our design goal. We also applied an acceleration profile instead of an instantaneous change in velocity which resulted in a much lower power draw, as seen in figure 24.

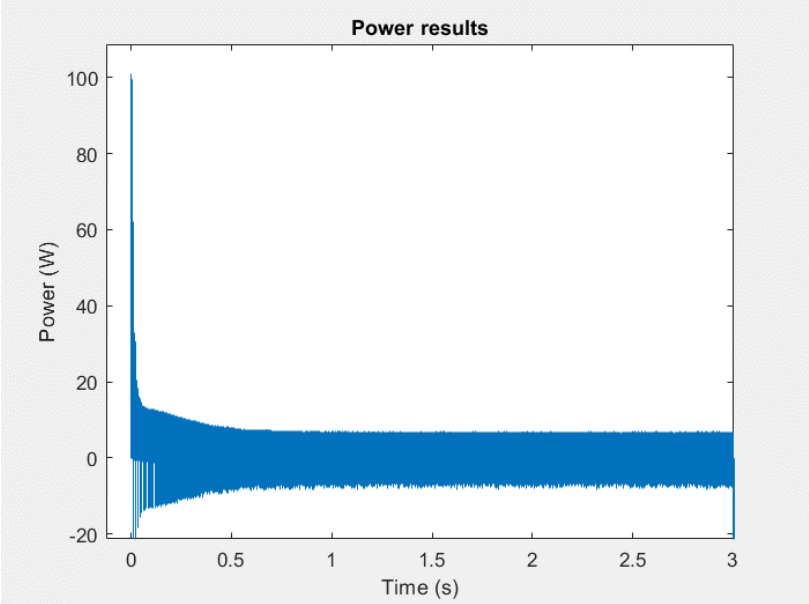

Figure 23: Instanteous power draw to 2m/s. The maximum power that this system will draw is 100W when a velocity of 2m/s is instantaneously requested.

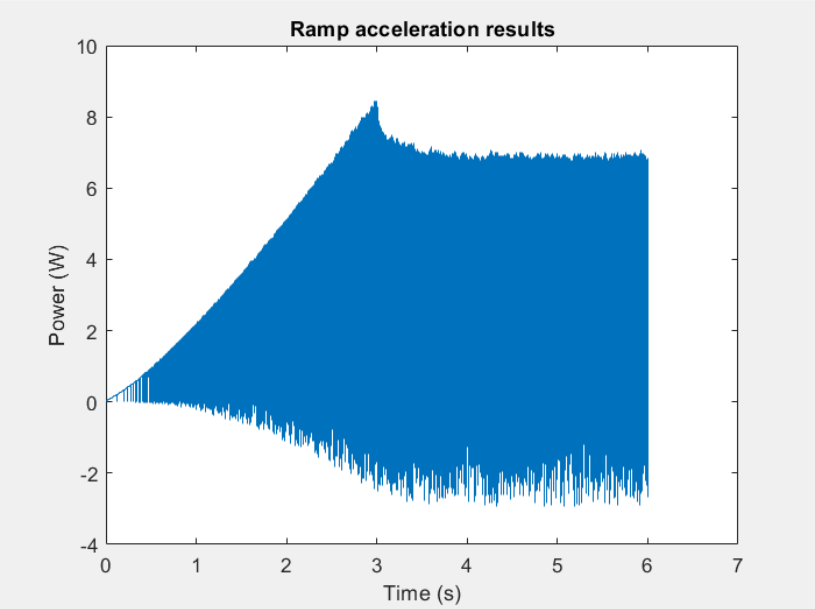

Figure 24: Ramp acceleration power draw to 2m/s. If we ramp the velocity slowly to 2m/s instead of instantaneously requesting it, we can see that our power draw is much lower.

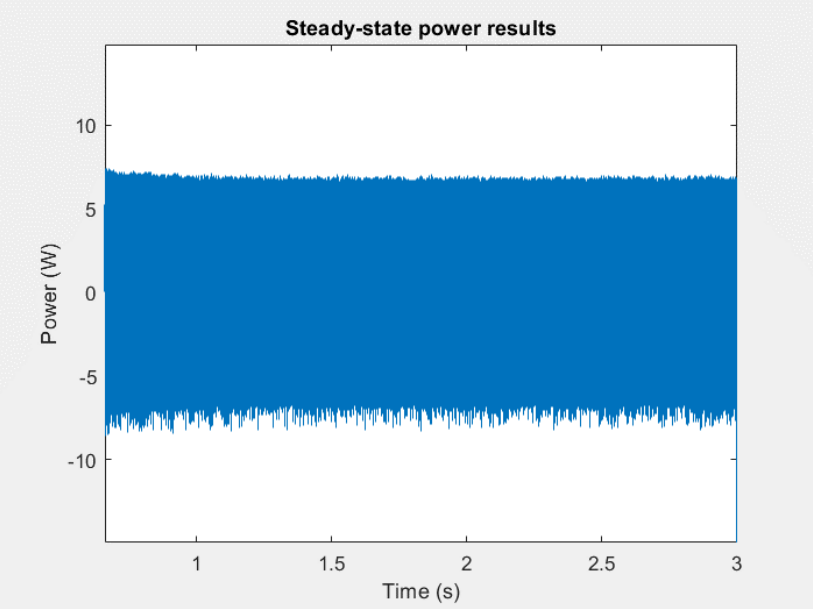

Figure 25: Steady-state power draw at 2m/s. Once the motors reach their steady-state velocity, they pull under 10W.

Velocity Error

Lastly, we wanted our velocity to stay within a 5% error. As we can see from the figure below, we were able to achieve this goal with the error only occurring during big changes in RPM. In this case, we requested the system to accelerate directly to 2 m/s at the beginning of the simulation then back down to a stop starting at 3 seconds. As we can see in figure 26 and 27, the error is able to stay under 5% in both cases.

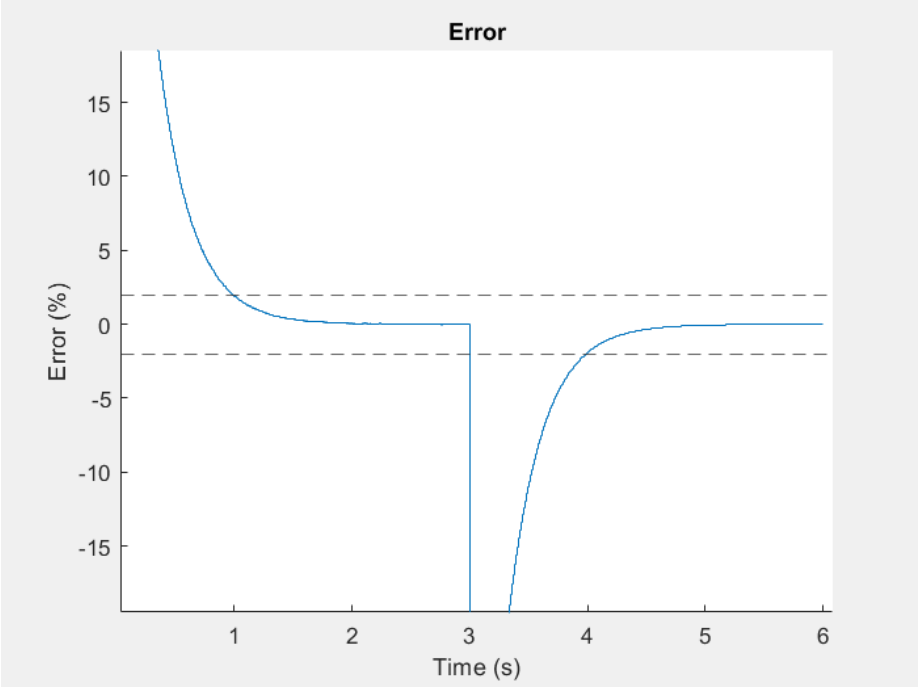

Figure 26: Velocity error after accelerating to 2m/s and decelerating to 0m/s. Our errors only occur when instantaneously requesting 2 or 0m/s.

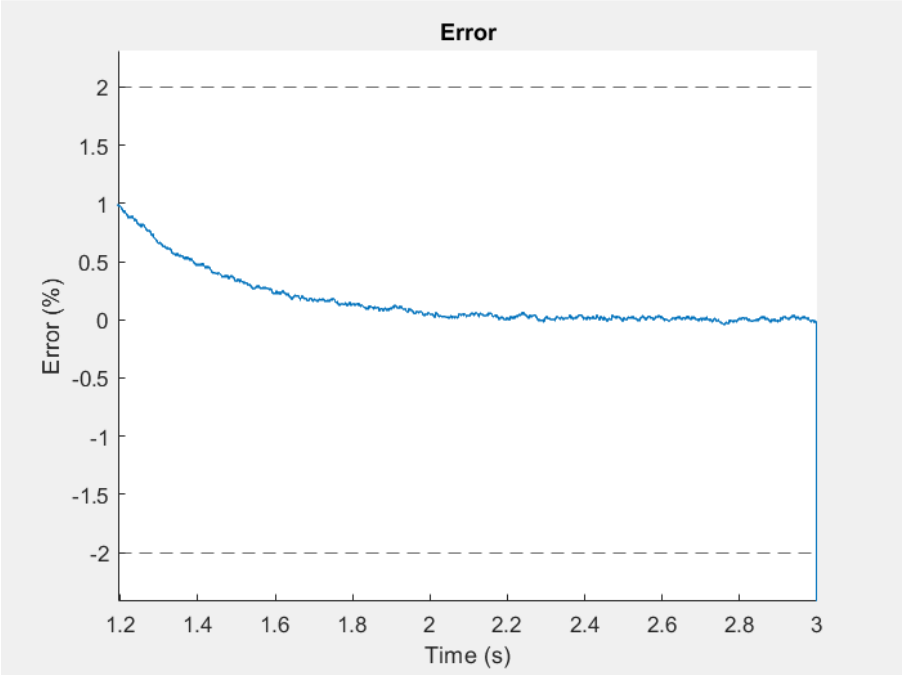

Figure 27: Zoomed in plot to show variation in error due to added noise in loading torque. Our error fluctuates less that 0.5% under simulation, however we expect to see more error in real-world testing.

DISCUSSION OF RESULTS

Through this experiment, I achieved the following values for the PID controller:

Kp	0.15
Ki	0.013
Kd	0.0003

Table 2: PID values. These values allowed our controller to meet our design goals.

Explanation of results

The PID controller is tuned such that each design goal is met. Changing the of our Kp, Ki, and Kd can be a positive change for one aspect, whereas it may cause another design goal to become out of spec. For example, increasing our proportional gain (Kp) could reduce our error as the motors would react with a quicker acceleration, however this would also increase our instantaneous power draw. In addition, changing the integral gain (Ki) may speed up the time it takes to reach steady-state velocity, however it may also cause the controller to overcompensate and cause a shakiness in the velocity, thus resulting in more error and steady-state power draw. With this, I believe that the values we achieved for our PID controller worked well in simulation and hope to be able to test this controller further next semester.

CONCLUSION

Gantt Chart

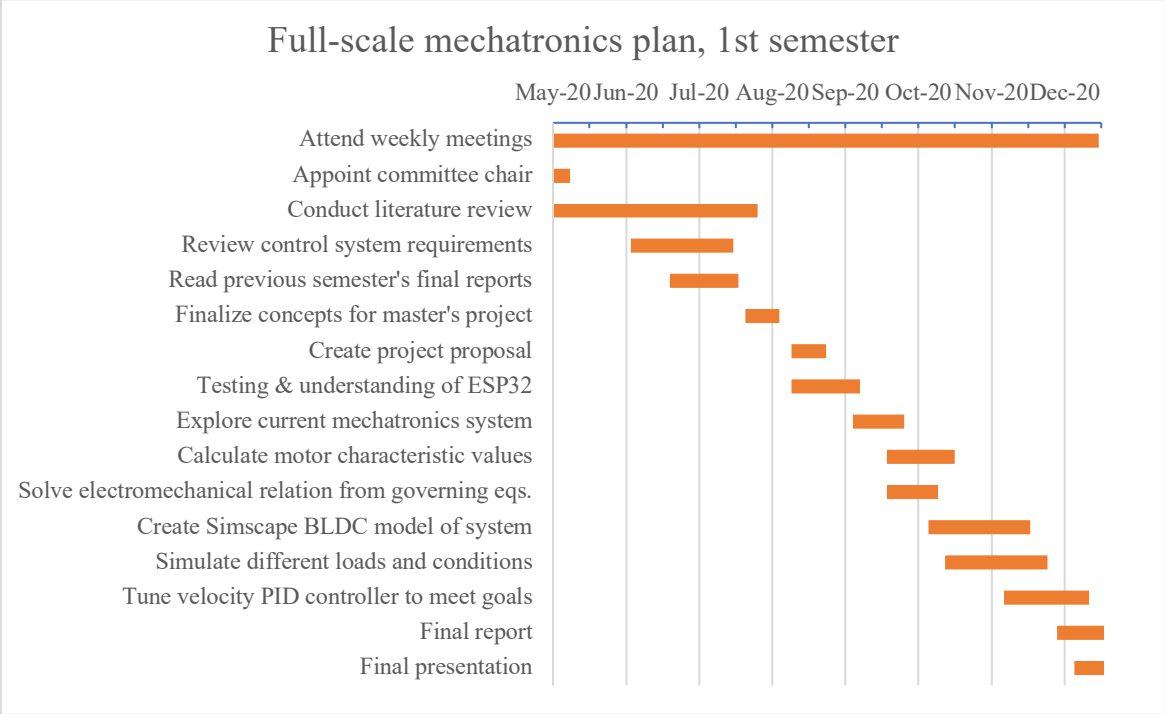

Figure 28: First semester Gantt chart. A timeline of all the tasks completed in the first semester can be found here.

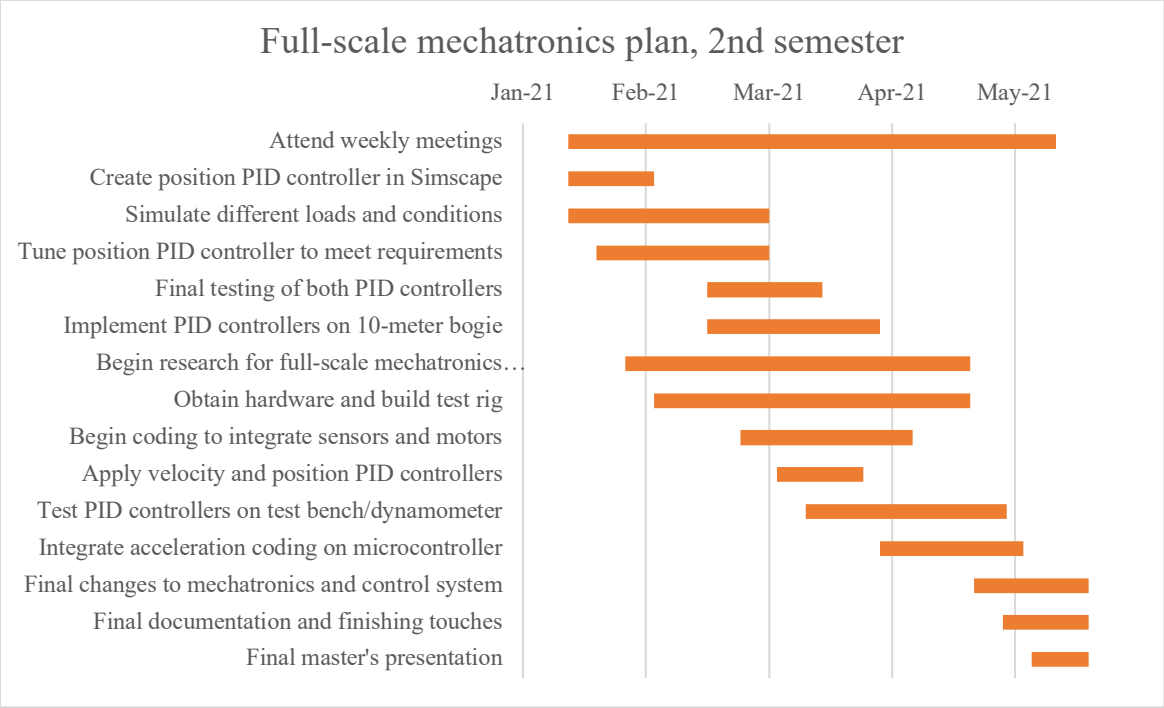

Figure 29: Second semester Gantt chart. Our expected tasks for next semester can be seen in this Gantt chart.

Next steps

Through the experiments above, we were successfully able to achieve the motor characteristics on the BLDC motors used on the 12-meter track. With this, we were able to take these values and simulate the motors under various conditions in Simulink. With this, we will be able to test any changes to the propulsion system before we deploy it to the bogie. In its current state, the 12-meter track only has one bogie. It is very difficult any changes to the bogie off the track, so having a simulation of the system will make for a much easier for making changes and tweaking anything.

The next steps for this project are to take the system from its current simulation stage to the testing phase. We have brought up the idea of using a dynamometer to apply a constant load to the motor while it is on a test bench. Once we have concluded that the motor works,

we can then deploy our PID controller to the 12-meter bogie and test it in real world conditions. This will help reveal any flaws that may have been missed in simulation but also ensure that our PID controllers can correct for any outside error that would not occur in simulation.

Lastly, research will need to begin for full-scale components. Because there is no current full-scale mechatronics system, research into what components are necessary and how they will be programmed and controlled will be necessary. Taking this project to the full-scale will bring us one step closer to innovate the future of transportation with the Spartan Superway.

REFERENCES

Attoh, Kafui Ablode. (2019). "Alternatives in Transit." In *Rights in Transit: Public Transportation and the Right to the City in California's East Bay*, 82-101. Athens: University of Georgia Press. Retrieved from: http://www.jstor.org/stable/j.ctt22nmc4p.10.

Cham, Chin-Long & Samad, Zahurin Bin. (2014). "Brushless DC Motor Electromagnetic Torque Estimation with Single-Phase Current Sensing." J Electr Eng Technol Vol 9. Retrieved from: https://pdfs.semanticscholar.org/5055/9e60567769e1cf2f364a00c01cc75442d1c3.pdf

CTC Control (2016). "Measuring motor parameters." Retrieved from: https://support.controltechnologycorp.com/customer/elearning/younkin/motorParameters.pdf

D. Lu, J. Li, M. Ouyang and J. Gu. (2011) "Research on hub motor control of four-wheel drive electric vehicle," *2011 IEEE Vehicle Power and Propulsion Conference*, Chicago, IL. pp. 1-5, doi: 10.1109/VPPC.2011.6043150.

DOT (Department of Transportation). (2018). "National Transit Database Provides Key Stats on Public Transportation in the U.S.". *The Federal Transit Administration*. Retrieved from: https://www.transportation.gov/connections/national-transit-database-provides-key-stats-public-transportation-us.

Ezike, Richard, Jeremy Martin, Katherine Catalano, and Jesse Cohn. (2019). "*Where Are Self-Driving Cars Taking Us?: Pivotal Choices That Will Shape DC's Transportation Future*". Report. Union of Concerned Scientists. pp. 17-21. doi:10.2307/resrep24063.9.

Furman, Burford, Sam Ellis, Lawrence Fabian, Peter Muller, and Ron Swenson. (2014). "Automated Transit Networks (ATN): A Review of the State of the Industry and Prospects for the Future," *Mineta Transportation Inst.* San José State University, San José, CA, Report 12-31, pp. 1, 7, Retrieved from: https://transweb.sjsu.edu/research/automated-transit-networks-atn-review-state-industry-and-prospects-future.

Furman, Burford. (2016). "The Spartan Superway: A solar-powered automated transit network," in Proc. *ASES National Solar Conf.* pp. 1-2. Retrieved from: http://proceedings.ises.org/paper/solar2016/solar2016-0019-Furman.pdf.

Garcia, Daniel Martinez, and Mary C. Sheehan. (2016) "Extreme Weather-driven Disasters and Children's Health." *International Journal of Health Services* 46, no. 1: 79-105. Retrieved from: https://www.jstor.org/stable/48512866.

SPUR (San Francisco Bay Area Planning and Urban Research Association). (2015). Report. *SPUR (San Francisco Bay Area Planning and Urban Research Association)*, doi:10.2307/resrep22946.

APPENDICES

Appendix A: Arduino Code

```
//-----------------------------------------------------//
// ACCELERATION VARIABLES HERE:
float m_duty = 0.1;
float acc = 0.0001; //define acceleration here (more 0's = longer acceleration)
float decel = 0.0005; //define deceleration here (more 0's = longer deceleration)
//-----------------------------------------------------//

#include <Arduino.h>
#include <VescUart.h>
// Creat VescUart object to provide access to
// the class functions

VescUart UART;
VescUart UART1;
HardwareSerial* serialPort = &Serial2;
HardwareSerial* serialPort1 = &Serial1;

float currentduty = 0;

volatile int count;
int count_prev = 0;

float t = 0;
float dt;

//hw_timer_t * timer = NULL;
//portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

// Allows count to increase without affecting main loop
//void IRAM_ATTR onTimer() {
//  portENTER_CRITICAL_ISR(&timerMux);
//  count++;
//  portEXIT_CRITICAL_ISR(&timerMux);
//}

void setup() {

  Serial.begin(115200);

  Serial1.begin(115200, SERIAL_8N1, 16, 17);
  Serial2.begin(115200, SERIAL_8N1, 23, 22);
  while (!Serial2 && !Serial1) {;}
  UART.setSerialPort(serialPort);
  UART1.setSerialPort(serialPort1);

// Setup below begins the timer to count 0.1 seconds
//  timer = timerBegin(0, 80, true);
//  timerAttachInterrupt(timer, &onTimer, true);
//  timerAlarmWrite(timer, 100000, true);
//  timerAlarmEnable(timer);
}

void loop() {
  accelDuty();
}
```

```
void accelDuty() {
        if (currentduty < m_duty && currentduty >= 0) {
            currentduty = currentduty - acc;  //ramp acceleration
            Serial.println(currentduty);
            ////vesc1.setDuty(currentduty);
        }

        else if (currentduty > m_duty && currentduty <= 1) {
            currentduty = currentduty * (1 + decel); //curve down deceleration
            Serial.println(currentduty);
            //vesc1.setDuty(currentduty);
        }

        else if (currentduty > 1) {
            currentduty = 1;
            //vesc1.setDuty(currentduty);
            Serial.println(currentduty);
            Serial.println("maxed out");
        }

        else if (currentduty > 1) {
            currentduty = 1;
            //vesc1.setDuty(currentduty);
            Serial.println(currentduty);
            Serial.println("maxed out");
        }

        else if (currentduty < 0) {
            currentduty = 0;
            //vesc1.setDuty(currentduty);
            Serial.println(currentduty);Serial.println("below min");
        }

        else if (currentduty = m_duty) {
            currentduty = m_duty; //stabilize velocity
            //vesc1.setDuty(currentduty);
            Serial.println(currentduty);
        }
}
```

Appendix B: Matlab Plots

```
load testdata.mat

%% Process data

R = .2464; % ohm
L = .00015614; % Henry
f = .008133 % Weber (flux linkage)

t = motorData(:,1);
rpm = motorData(:,6);
v = motorData(:,5);
i = motorData(:,4);

vtest = [1.5, 5.2, 7.0]; % multimeter measurement
rpmtest = [1354, 3686, 5633]
itest = [0.16973, 0.210, 0.260]

rads = rpmtest * pi/30;

figure(1);
plot(rads, vtest - (itest*R),'x')
xlabel('speed (rad/s)');
ylabel('Voltage(V)');
title('V = Kb*w + iR');

Kt = 0.01232;

figure(2);
plot(rads, Kt*itest,'o')
xlabel('speed (rad/s)');
ylabel('Torque (N-m)')
title('Kt*i = c*w + Tf')

c = 2.403e-6;
Tf = 0.001664;

%% Simulink simulation

open('SuperwayBLDCModel.slx')
sim('SuperwayBLDCModel.slx')

%% Plots

v_des = [0 0; 2 0.0000000001; 2 2.999999; 0 3; 0 6];

figure(3); hold on
plot(velocitysim/1000)
plot(v_des(:,2),v_des(:,1),'--');
xlabel('Time (s)');
ylabel('Velocity (m/s)');
title('Acceleration');

figure(4);
plot(powersim);
xlabel('Time (s)');
ylabel('Power (W)');
title('Power results')

figure(5); hold on
plot(100 - ((500-errorsim)/500) * 100);
yline(2, '--');
yline(-2, '--');
xlabel('Time (s)');
ylabel('Error (%)');
title('Error');
```

Appendix C: Simscape setup