ABSTRACT

Design and Development of Autonomous Electric Vehicles Capable of Following an EPA Drive Cycle on a Dynamometer Testbed and Navigating an On-Road Obstacle Course

Ezekiel B. Brown, Ph.D.

Advisor: Annette von Jouanne, Ph.D.

This dissertation presents the background, research and further advancement of the design and development of an autonomous electric vehicle that can follow an Environmental Protection Agency (EPA) drive cycle on a dynamometer testbed. Design, development, and test data collected, analyzed, presented, and referred to in this dissertation stems from an all-electric Chevy Bolt and an electric converted Chevy Tahoe. The work done on the Chevy Bolt enables the vehicle with the capability to autonomously follow an EPA drive cycle on a dynamometer and the work done on the Chevy Tahoe advances on the autonomous acceleration system, incorporating autonomous steering and sensory integration, enabling the vehicle to follow a path autonomously. The programmable throttle and the programmable brake research on the all-electric Bolt are discussed followed by the implementation of a programmable acceleration system, programmable steering, and the development of a sensory system on an all-electric converted Chevy Tahoe.

Design and Development of Autonomous Electric Vehicles Capable of Following an EPA Drive Cycle on a Dynamometer Testbed and Navigating an On-Road Obstacle Course

by

Ezekiel B. Brown, B.S., M.S.

A Dissertation

Approved by the Department of Electrical and Computer Engineering

Kwang Lee, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Doctor of Philosophy

Approved by the Dissertation Committee

Annette von Jouanne, Ph.D., Chairperson

Alex Yokochi, Ph.D.

Emmanuel Agamloh, Ph.D.

Mack Grady, Ph.D.

Accepted by the Graduate School
May 2023

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES v

CHAPTER ONE 1
Introduction 1
Motivation 1
Background 1
Literature Review 4

CHAPTER TWO 13
Batteries and Battery Management Systems 13
Batteries 13
Battery Management Systems 16

CHAPTER THREE 24
Chevy Bolt's Autonomous Vehicle Speed System 24
Programmable Throttle 24
Programmable Brake 33

CHAPTER FOUR 40
Electric-Converted Tahoe's Autonomous Vehicle Speed Control System 40
Throttle And Brake Setup 40
Power System Safety Design 47
Pid Tuning For Vehicle Speed 50

CHAPTER FIVE 56
Sensor Implementation 56
Global Positioning System (Gps) 56
Ultrasonic Sensors 57
Camera And Yolo (You Only Look Once) 59

CHAPTER SIX 63
Programmable Steering And Data Analysis 63
Steering Rack Control 63
Data Gathering 72

CHAPTER SEVEN 74
Conclusions And Future Work 74
Conclusions 74
Future Work 76

REFERENCES 81

LIST OF FIGURES

Figure 1.1. Converted all-electric Chevy Tahoe on the red and black Dynamometer 2

Figure 1.2. Autonomous vehicle sensors 6

Figure 1.3. Autonomous levels 7

Figure 1.4. Encoder 8

Figure 1.5. Neural Network 9

Figure 1.6. Map drawing of the data coverage using crowdsourcing 11

Figure 1.7. Another map drawing of the data coverage using crowdsourcing 12

Figure 2.1. Original battery case donated by Proterra's electric bus 13

Figure 2.2. Lithium titanate 10-cell battery module 14

Figure 2.3. Chevy Tahoe's traction battery pack containers 15

Figure 2.4. Closer view of a battery case and BMS 16

Figure 2.5. 100A DALY BMS 17

Figure 2.6. Right-side up view of traction battery case with BMS 18

Figure 2.7. 300A LTO DALY BMS 19

Figure 2.8. 300A Smart BMS communication screen 3 20

Figure 2.9. 300A Smart BMS communication screen 1 21

Figure 2.10. 300A Smart BMS communication screen 2 22

Figure 3.1. Programmable throttle connected to Chevy Bolt 25

Figure 3.2. Programmable throttle user interface 26

Figure 3.3. Example notepad drive cycle file to be uploaded into LabView 27

Figure 3.4. Example drive cycle displayed in LabView 28

Figure 3.5. Road load forces 30

Figure 3.6. Autonomous acceleration system block diagram 31

Figure 3.7. Programmable brake 34

Figure 3.8. Data Acquisition (DAQ) setup 35

Figure 4.1. Motor rpm to speed in MPH 41

Figure 4.2. Throttle VCU setup 41

Figure 4.3. Arduino to digital potentiometer to voltage decipher 43

Figure 4.4. Arduino to digital potentiometer (AD5206) to voltage decipher schematic ..43

Figure 4.5. Serial interface of VCU AF trace output.....44

Figure 4.6. Predicted range of vehicle 44

Figure 4.7. UQM diagnostic software 45

Figure 4.8. VCU wiring diagram 46

Figure 4.9. Power vs. motor rpm graph 48

Figure 4.10. Traction battery pack on/off switch 49

Figure 4.11. Servos and VCU battery supplies on/off switch 50

Figure 4.12. Tuned PID constants with a negative I constant 51

Figure 4.13. Same PID values as Fig. 36 with completely different results.....52

Figure 4.14. Tuning PID with values: .5, .01, and 3.2.....53

Figure 4.15. Tuning PID with values: .7,0,3.2.....54

Figure 4.16. Tuned PID with nonzero P and D constants and 0 for the I constant.....55

Figure 5.1. GPS unit 57

Figure 5.2. Chevy Tahoe sporting ultrasonic sensors 58

Figure 5.3. YOLO object detector and classifier 60

Figure 5.4. Stop sign for obstacle course put through YOLO analysis 61

Figure 5.5. Statistics on the YOLO analysis shown in Fig. 44.....61

Figure 6.1. High torque servo motor and steering connection 64

Figure 6.2. Steering fluid pump system 65

Figure 6.3. Lower torque servo motor 66

Figure 6.4. Dyno ultrasonic sensor EMI 69

Figure 6.5. Autonomous steering guidance 70

Figure 6.6. Obstacle course 71

Figure 7.1. EMI shielding 76

Figure 7.2. Different sensors used to detect objects around it 77

Figure 7.3. Future conference seating arrangement 78

Figure 7.4. Parallel component to make BMS able to monitor battery strings 79

Figure 7.5. Parallel battery management systems' schematic 80

CHAPTER ONE

Introduction

Motivation

In order to improve safety, transportation efficiency and reduce the carbon emissions from transportation, electric vehicles are being researched and developed by many different automotive companies, such as Tesla, Lexus, BMW, Mercedes, Volvo, and more. In development, it is important to be able to carefully quantify the energy efficiency of the vehicle, which is standardly done by driving the electric vehicle through a drive profile on a dynamometer through what are known as EPA drive cycles. However, these are currently accomplished by having a human driver in control of the vehicle and so results vary from more than just the vehicle but also from driver to driver that may make conclusions from data less convincing or reliable. That is a significant motivation for developing autonomous acceleration and braking systems that can give reliable, consistent, and very accurate data to make much clearer conclusions and evaluations of the electric vehicle's performance. In addition, autonomous vehicles would enable the aging population to continue to be independent. These are some of the driving forces for the research and development of autonomous vehicles for transportation that will be further detailed in the following sections.

Background

An autonomous acceleration system was developed on a Chevy Bolt all-electric vehicle and will be referred to as programmable throttle and brake throughout this

dissertation. In the work pertaining to the programmable throttle and brake, the research was conducted on the all-electric Chevy Bolt, which was placed on a dynamometer as the test bed. A dynamometer is shown as the platform the vehicle is resting on in Fig. 1.1, which is essentially a treadmill for vehicles, allowing the vehicle to accelerate and decelerate within a fixed location.

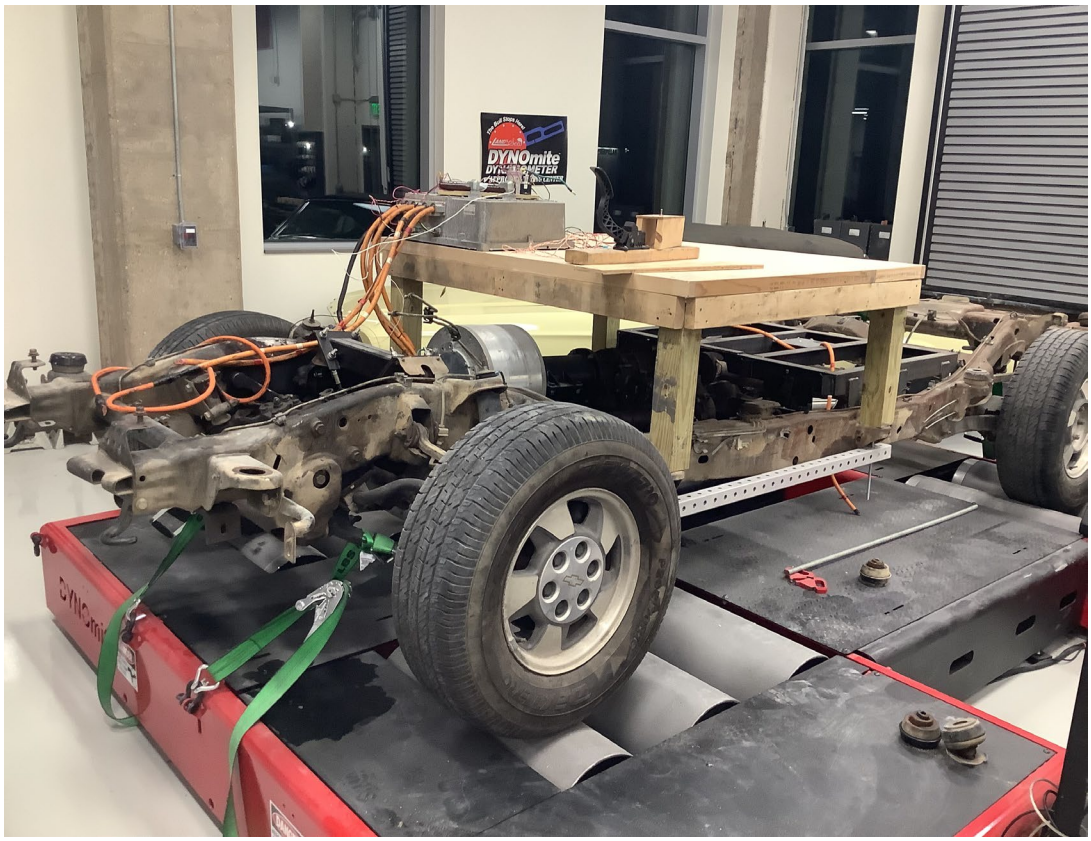

Figure 1.1. Converted all-electric Chevy Tahoe on the red and black Dynamometer.

Dynamometers are commonly used for vehicle testing by running various vehicles through EPA drive cycles. An EPA drive cycle puts a vehicle through a driving routine that simulates what a driver may experience in a city with stop lights and lower speed driving, as well as highway driving [1]. With the programmable throttle and programmable brake, an electric vehicle like the Chevy Bolt, which was used in this

research, can be put through a drive cycle for hours and do so with machine precision and consistency. The data collected from the Chevy Bolt was consistent and reliable, so when subjecting the car to differing charging applications, the effects the charging has on the battery can be analyzed and reliable conclusions and observations can be made. With the data gathered, the progression of electric vehicles may be furthered with consistent and reliable test results. The EPA has regulations for both on-road and non-road vehicles and has differing expectations and criterion to be met based on the vehicle's classification such as passenger vehicles, commercial trucks and buses, and motorcycles. The criterion to be met includes smog, soot, and other air pollution emissions from the vehicle in question. When testing and judging these vehicles, they put them through a drive cycle on a dynamometer and have professional drivers follow the speed the drive cycle indicates. This is done with city and highway drive cycles to simulate realistic driving patterns, and these tests can take hours to complete. The tolerance for the drivers is to maintain within 3 MPH above or below the drive cycle's indicated speed at all times. With there being different drivers, the data gathered during these drive cycles may not be consistent which can cloud inferences and conclusions. To solve this problem, the ideas of programmable throttle and brake were proposed by Dr. von Jouanne. Being able follow EPA drive cycles with reliable accuracy, consistent data can be obtained and makes observations and conclusions clearer and more reliable.

In many electric vehicles, there is a regenerative braking feature that is very important for the design and performance of the vehicle as stated in [2]. In the infrastructure of an electric vehicle there is a motor that produces the traction force or propulsion force to the wheels of the vehicle; this motor can usually also act as a

generator. This means as the motor produces power, it can also generate power. When the throttle pedal is released completely, the electric motor now acts as a generator and the power is transferred to the battery by the inverter [3]. The inverter controls and transforms the electric power necessary in running the motor. The inverter also takes the power generated by the motor and converts and transfers this power to the battery storage unit when in regenerative braking mode. When the throttle pedal is released, the kinetic energy stored in the rotating inertia of the motor is taken and harnessed by the motor, converting this energy to electrical energy. This slows down the car without the use of traditional friction brakes [4]. Regenerative braking is very important for electric vehicles, reducing greatly the handicap of low specific energy density of electric vehicle batteries compared with traditional gasoline. This braking helps increase the range of the vehicle without adding more batteries and much extra weight or size to the vehicle. When driving, friction brakes are needed much less, and may hardly be needed while driving.

Literature Review

According to the National Highway Transportation Safety Administration (NHTSA) in [5], 90% of all serious car accidents and between 94% and 96% of all vehicle accidents occur due to human error, so displacing the human component will eliminate the human error and result in a drastic increase to transportation safety. These human errors can occur for various reasons, the more serious and frequent offenses include texting while driving and driving while under the influence. Also, for trucks carrying cargo, driverless trucks can travel longer distances without breaks, lowering delivery times with a reduced chance of monetary loss involved with accidents. In

addition, autonomous vehicles would enable the aging population to continue to be independent. These are some of the driving forces for the research and development of autonomous vehicles for transportation. Autonomous systems are systems characterized as capable of making decisions partially or completely independently of human interference, but unlike mere automation, they are able and expected to make these decisions while facing uncertainties. Uncertainties are what make development of these vehicles difficult, because if not designed properly, disasters can happen given vehicles with a rather large mass moving at high speeds. Such vehicles need to be carefully programmed and capable of virtually handling any scenario that can happen on the road.

Some of the technology incorporated within an autonomous vehicle's design include lidar, radar, cameras, and a Global Positioning System (GPS) system [6]. An example of how these systems could be incorporated onto a vehicle is shown in Fig. 1.2. Lidar sensors are used for determining distances, recognizing road lines and edges by shooting pulses of light around the vehicle [7]. Lidar is very accurate and gives a detailed and a more thorough description or "view" of the vehicles' surroundings than radar, however complex data analysis techniques are required for lidar and it is not as reliable as radar including the fact that lidar does not work very well in adverse conditions such as in rain or fog [8]. Radar works by emitting pulses of radio waves that travel very fast and bounces and returns to the sensor, and can determine the distance between an object and the sensor along with the speed of the object [9,10].

Cameras are also used for object detection and will be instrumental for the vehicle to operate and adhere to road signs in addition to avoiding cars and other objects on the road. Ultrasonic sensors are also utilized in autonomous vehicles to help determine the

surroundings of the vehicle at very close range. Cameras and ultrasonic sensors are often implemented in standard vehicles so this technology is not uncommon. Many people have cars, SUVs, and trucks that use cameras for extra rear-view support when in reverse, with some vehicle packages having cameras on top, sides, and front of these vehicles as well, in order to aid with parking. These cameras are often coupled with ultrasonic sensors to help the driver in providing greater visibility around the car and alert the driver if the vehicle is approaching any objects. Often the sensor data is mapped on some display screen where the camera feed is in the vehicle along with a beeping sound corresponding to the distance from a nearby object determined by the ultrasonic sensors. The closer the vehicle gets to an object detected by the ultrasonic sensors, the beeping frequency increases. Taking this to a more advanced application, an increasing number of vehicles are offering "assisted parking" where the vehicle uses cameras, sensors, and actuators for the vehicle to autonomously park itself into a parking space.

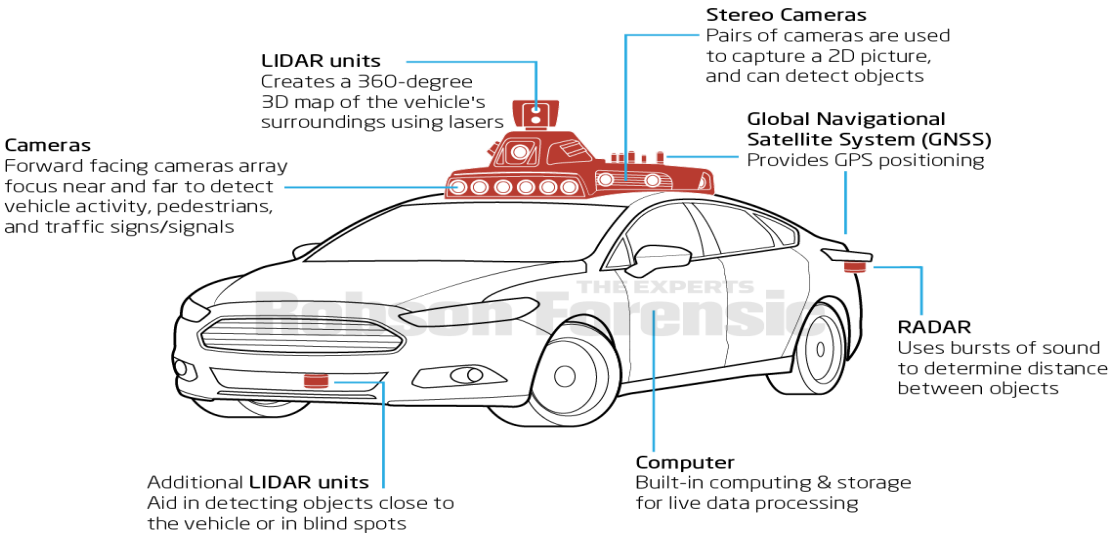

Figure 1.2. Autonomous vehicle sensors [11].

There are 5 levels in autonomous vehicles for determining how far removed they are from needing human assistance when operating as shown in Fig. 1.3 [12]. The goal of this work is to develop a level 5 autonomous vehicle, i.e., one not needing human assistance when operating. Currently, Tesla is at level 2 in its Model 3 where it can drive roads and highways as long as the human driver is alert at all times monitoring throughout the trip.

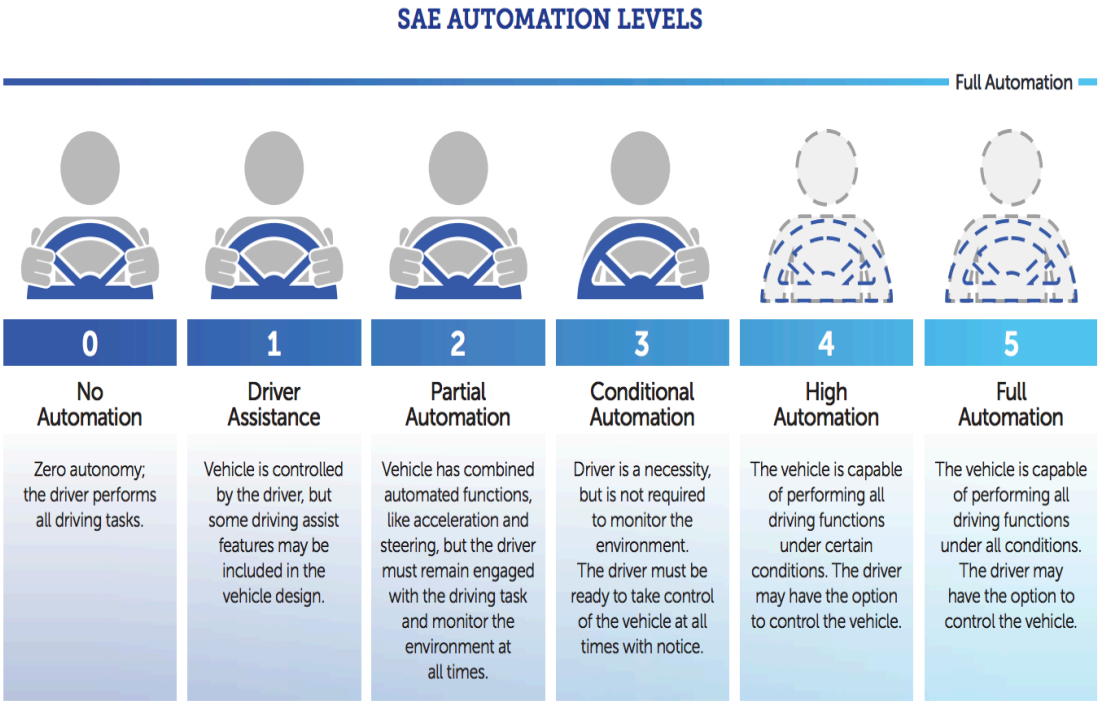

Figure 1.3. Autonomous levels [12].

For a vehicle to rely on cameras to observe the environment and the objects in the vicinity of the vehicle, the data gathered must be very reliably analyzed for the vehicle to operate properly and most importantly, safely. The computer system in which the vehicle is operating from must be able to do this under various weather conditions, extreme or not. Otherwise, autonomous vehicles would be too unreliable and dangerous to operate in

practice. To aid in this, the computer system can use a neural network to clean up images fed in from the cameras when analyzing before making decisions as seen from the encoder shown in Fig. 1.4 [13].

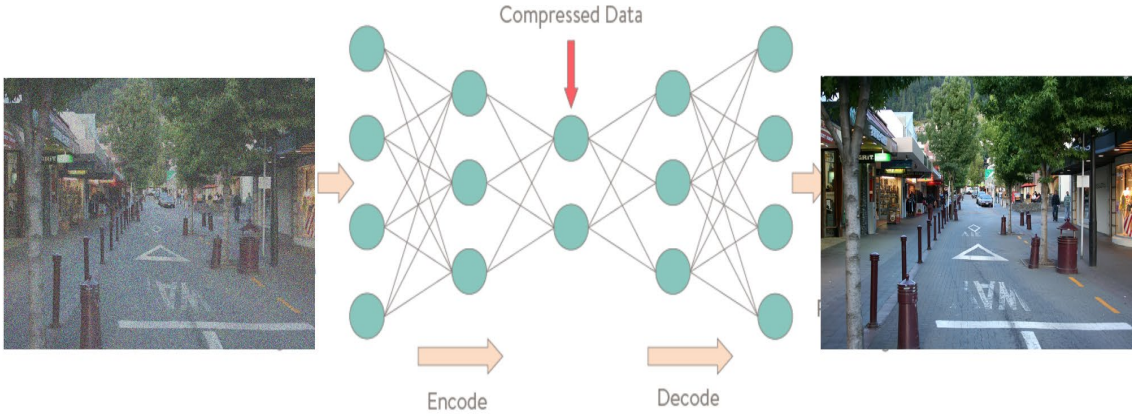

Figure 1.4. Encoder.

The use of a neural network can greatly increase the reliability of the image analysis using cameras [14]. In addition, traffic signs, objects, and lights emitted from traffic lights can be read and analyzed properly so the vehicle can be operated as expected. For example, a Convolutional Neural Network (ConvNet/CNN) is a deep learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other as shown in Fig. 1.5 [15].

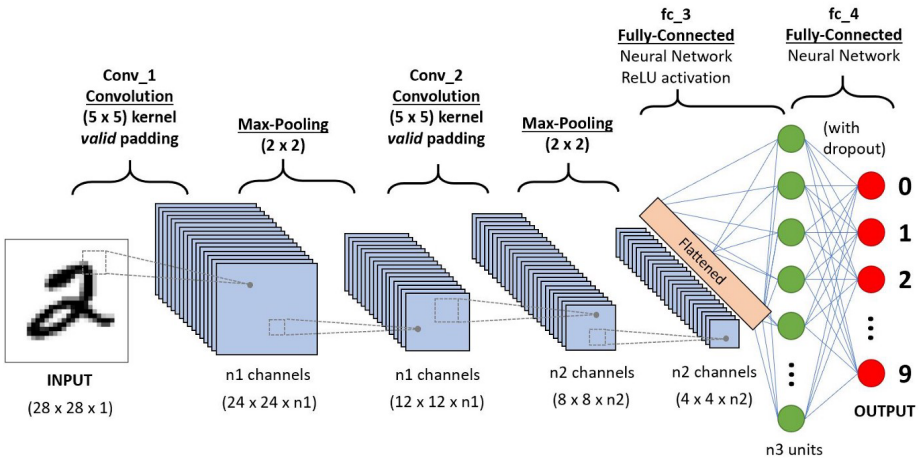

Figure 1.5. Neural Network.

Fig. 1.5 illustrates the purpose of a neural network to be used for data analysis in the computer system, especially for the cameras' input as discussed above. The number 2 on the very left represents data input as an image and goes through the neural network process where the system at the end to the right recognizes the number 2 is the input and so the output would be 2 [15,16]. This can also be applied with traffic signs and other specific objects. The vehicle's cameras can feed in a traffic sign image to the neural network input as seen in Fig. 1.5 after removing noise by running it through an auto encoder as shown in Fig. 1.4. The image will be processed through the neural network process shown above and the output of what the sign is will match the input, and that output will be passed to the vehicles control system input that will affect the behavior and path planning of the vehicle. Deep Learning is also used by the on-board autonomous vehicle computer system to predict the trajectory of the path the vehicle should follow [17]. For example, it would be able to see the road path ahead like a wide bend and calculate how it should turn at a specific position on the road.

Some autonomous vehicles are programmed to abide by an accurate map of its surroundings before actually driving on autopilot, this is done by 3rd party companies driving highways and main roads throughout the region and selling the data to vehicle companies [18]. An alternate approach being developed is where a source of data some companies use falls under crowdsourcing, where data from a company's vehicles on the road harvest this data and analyzes it to get a depiction of the surroundings [19]. Crowdsourcing seems to be most useful for determining changes in a vehicle's population's behavior such as roadblocks, construction, or traffic delays. With this data acquisition software being in more and more vehicles, projections of the coverage crowdsourcing can have show that as time goes on, mapping data will be harvested on most major roads [18]. The idea in using mapping in programming an autonomous vehicle is that the vehicle would have less data to compute and process in real time and in turn being able to make decisions faster, resulting in a smoother driving experience. The tradeoff is being able to activate autopilot only on roads and highways that it already has data on instead of being able to react to its environment in real-time, whereas a real-time algorithm can operate on autopilot in the gaps between the black paths displaying the road and highways with mapping data as described in Fig. 1.6 [18] and Fig. 1.7 [19].

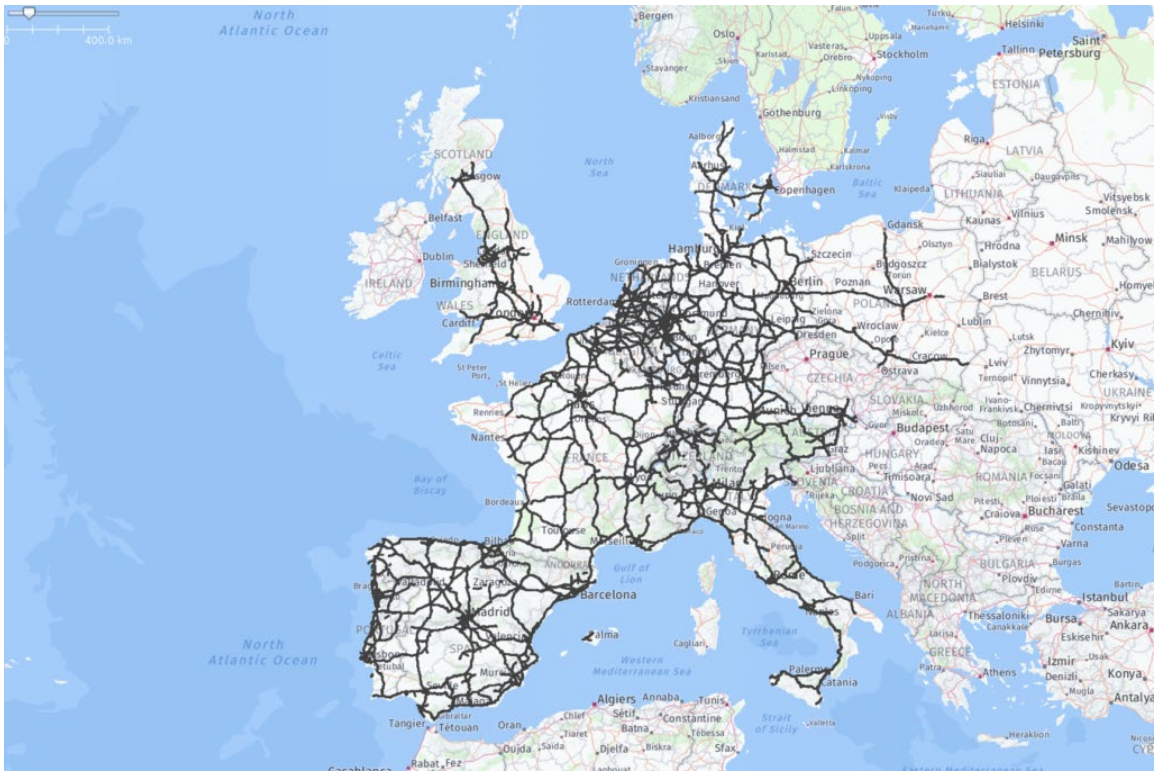

Figure 1.6. Map drawing of the data coverage using crowdsourcing.

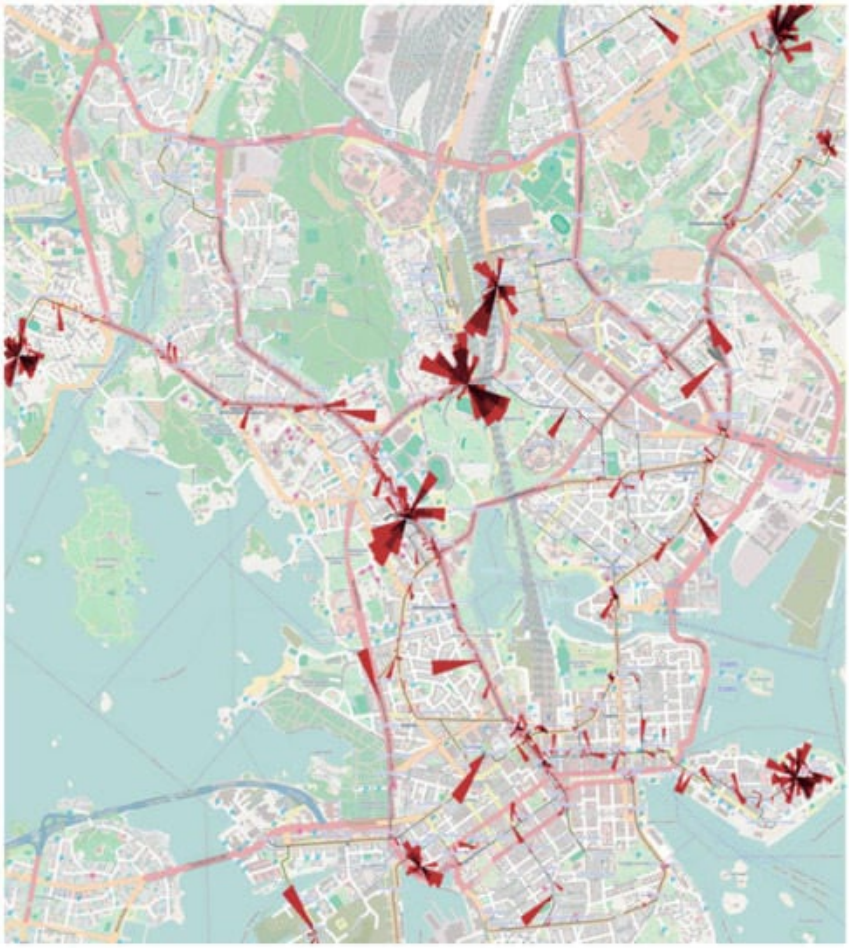

Figure 1.7. Map drawing of the data coverage using crowdsourcing.

For the contributions in this paper, ultrasonic sensors were used as ping sensors, and a neural network, “you only look once” (YOLO), was interfaced with a camera where each image is processed for object and classification directly without using an encoder.

CHAPTER TWO

Batteries and Battery Management Systems

Batteries

An electric bus was donated to Baylor's Energy and Renewable Systems research lab by Proterra. The bus was striped for the Lithium-titanate batteries, permanent magnet motor and inverter. The batteries were encased, with each case containing 10 battery modules, which is a little over 23V in each along with 50Ah in capacity. Each case is designed for coolant to run through the tubes running throughout the case keeping the batteries cool, as well as a battery management system (BMS) on each case shown in Fig. 2.1 and Fig. 2.2.

Figure 2.1. Original battery case donated by Proterra's electric bus.

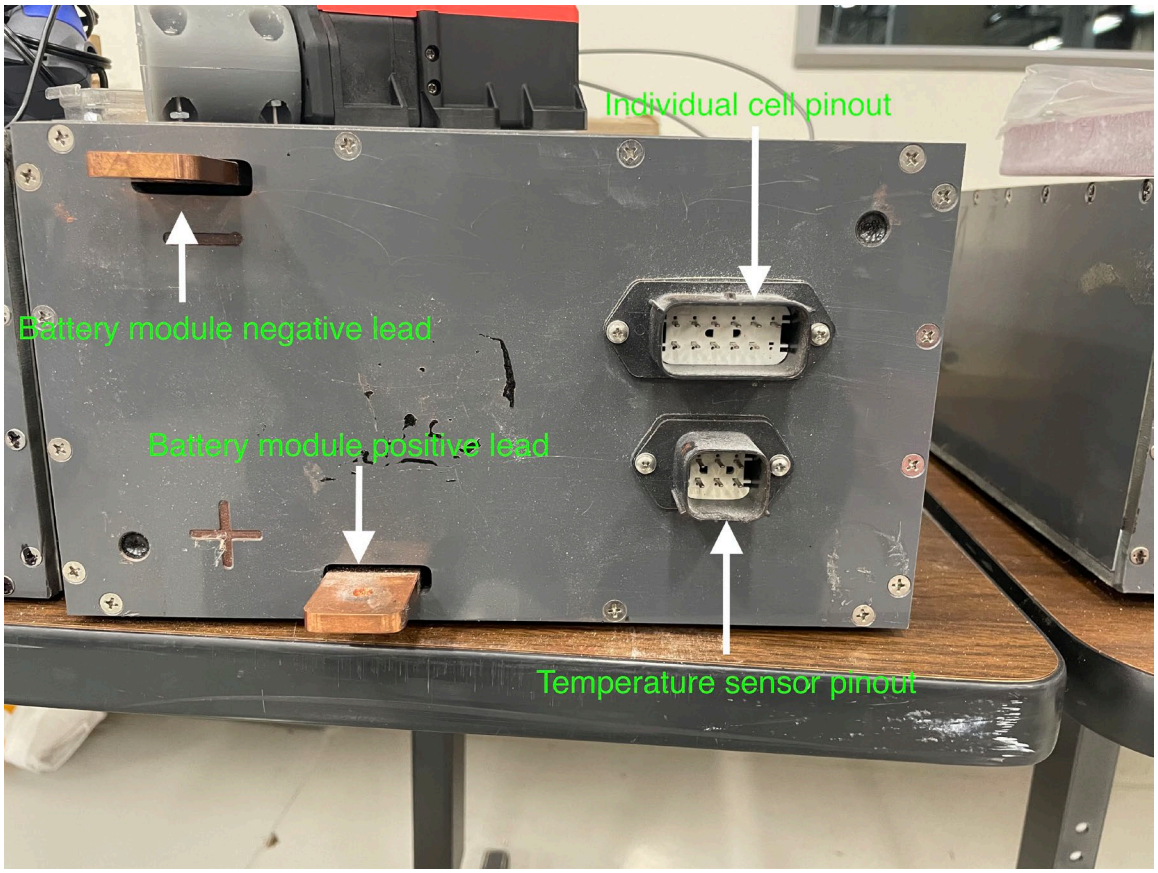

Figure 2.2. Lithium titanate 10-cell battery module.

The BMS has been disconnected for proprietary reasons. 12 battery modules were taken from the cases and connected in series and placed in the target vehicle's battery container in Fig. 2.3.

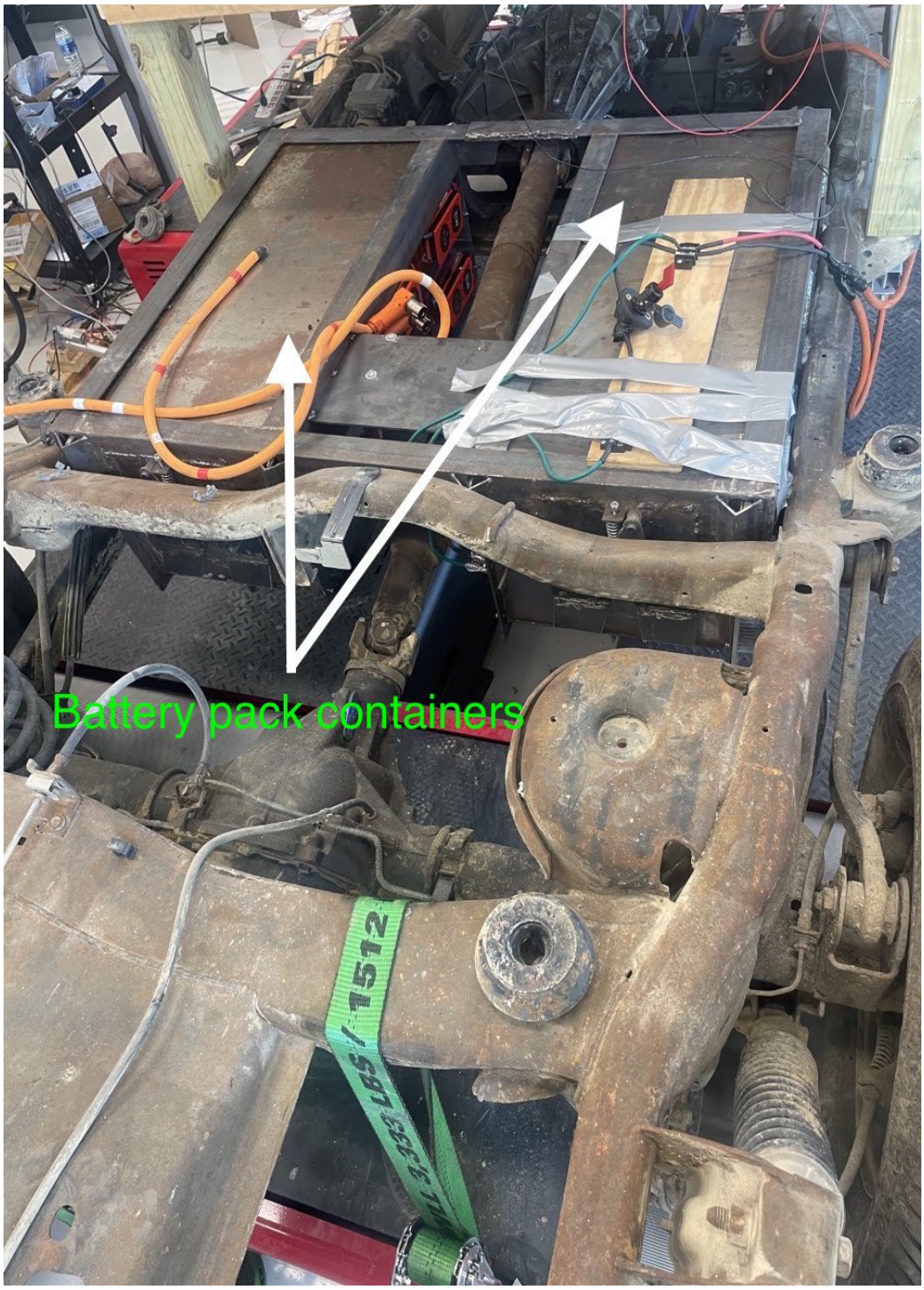

Figure 2.3. Chevy Tahoe's traction battery pack containers.

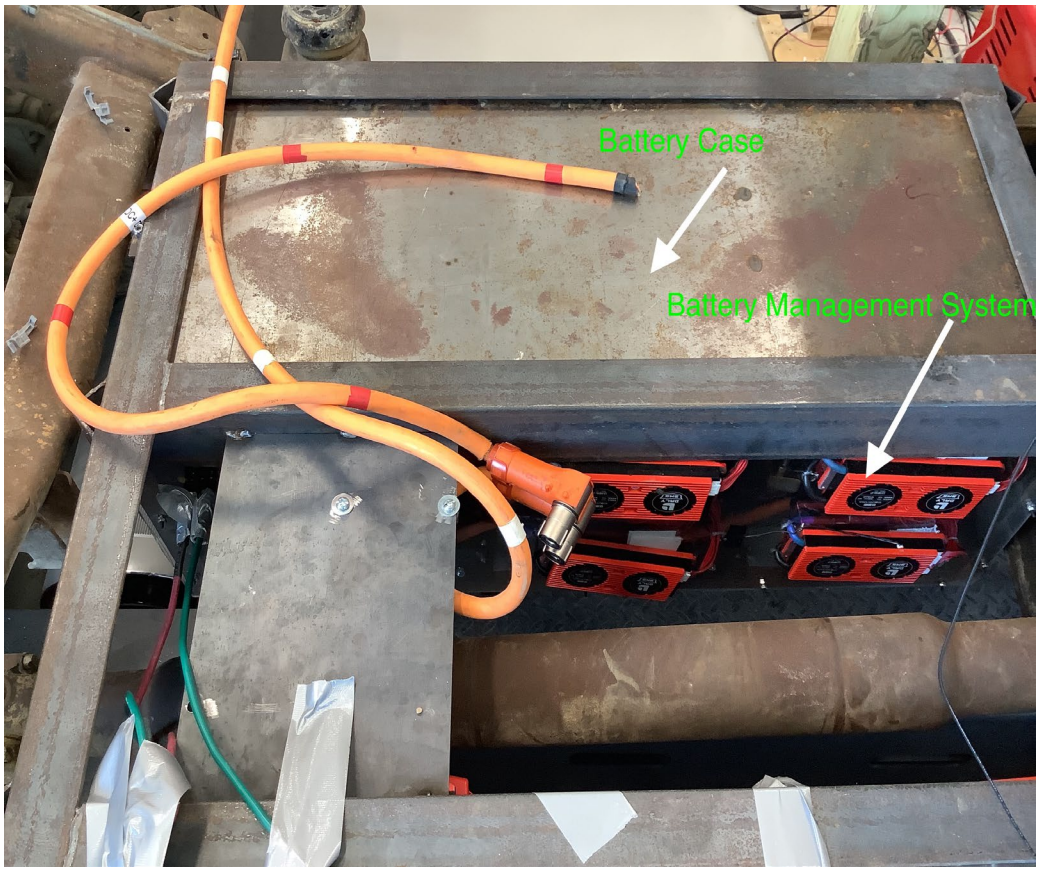

Figure 2.4. Closer view of a battery case and BMS.

On the dynamometer test-bed is a 2001 Chevy Tahoe that has had its internal combustion engine and transmission stripped and replaced with the electric bus's permanent magnet motor and inverter. Currently, the Chevy Tahoe is stripped down to its skateboard chassis platform with a wooden platform mounted on top of it to secure the inverter and have somewhere to sit when testing until the original body is placed back unto the vehicle as shown in Fig. 2.4.

Battery Management Systems

Battery management systems (BMSs) are used to regulate the battery modules in the battery pack used for traction power. Using an actual motor rpm vs. current graph, the

maximum speed the vehicle can go and stay within the current limitations of the lithium-titanium-oxide (LTO) BMS available of 100A is 19 MPH (Fig. 2.5). Already having a 300A 24V LTO BMS as seen as the biggest BMS in Fig. 2.6 and Fig. 2.7, 9 other BMSs were ordered to manage the other 11 battery modules. Two of these BMSs are 100A 48V LTO BMSs and can each manage 2 battery modules and the rest are 100A 24V LTO BMSs that each manage 1 battery module.

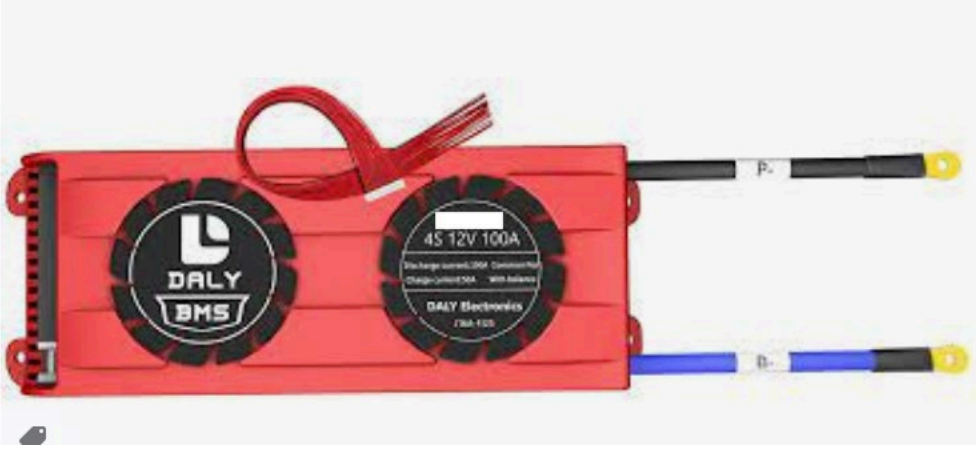

Figure 2.5. 100A DALY BMS.

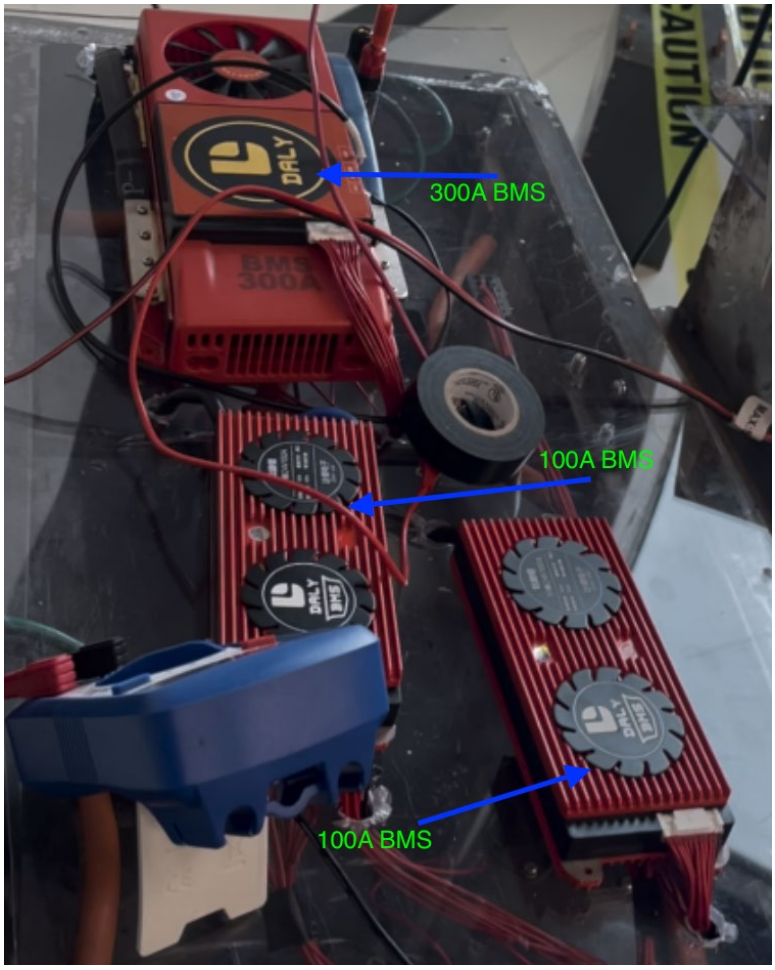

Figure 2.6. Right-side up view of traction battery case with BMS.

Figure 2.7. 300A LTO DALY BMS.

The BMSs are connected in series with the modules for short circuit protection and overcurrent protection. Cell balancing wires (red) are connected to the positive of every individual cell and the ground pin (black). The BMS blue B- cable connects to the negative of the battery module it is managing and the black P- cable connects to the battery with a lower potential difference from its positive lead to ground, or to the inverter negative rail in the case of the first module in series as shown in Fig. 2.4. The BMS also trips if the battery is below or above its lowest or highest threshold voltage, being undervoltage and overvoltage. If these conditions occur the battery module/cell will need to be brought back to its required voltage range and the BMS will have to be reset by shorting the blue B- and black P- cables to reactivate. The BMS are needed to keep the battery modules and cells within the modules balanced to avoid battery damage while charging and discharging while the vehicle is and is not in operation [20,21]. The BMS

will be regulating the current flow [22] into and out of the batteries keeping it within 100A and monitors the temperature of the battery packs. The three figures below are the user interface screens for monitoring and altering parameters set within the 300A smart BMS.

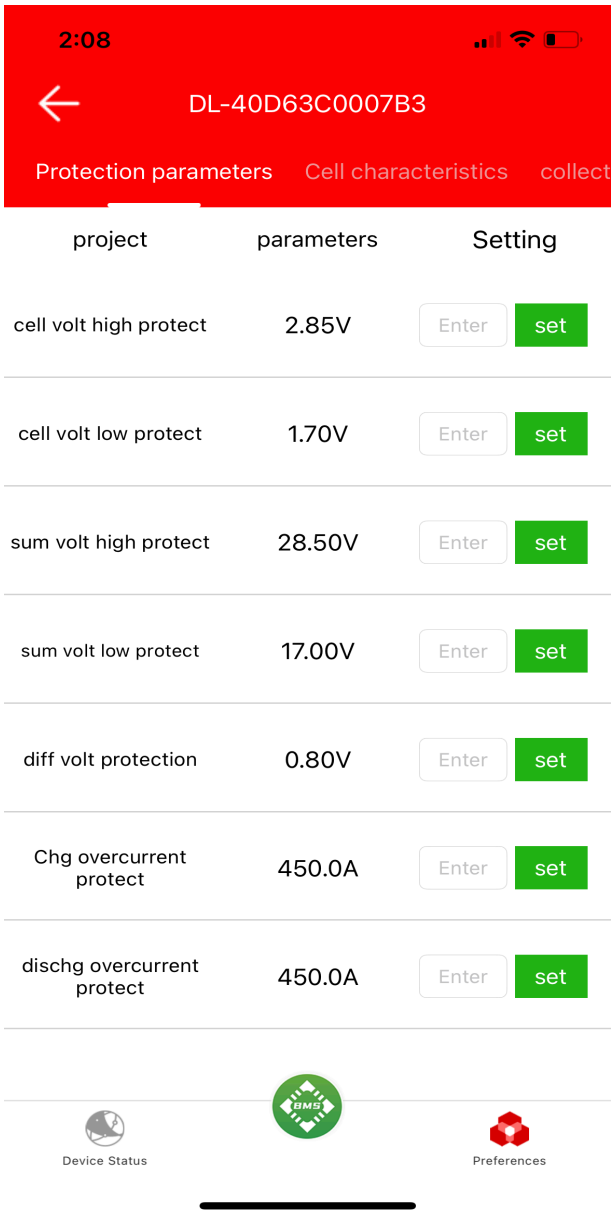

Figure 2.8. 300A Smart BMS communication screen 3.

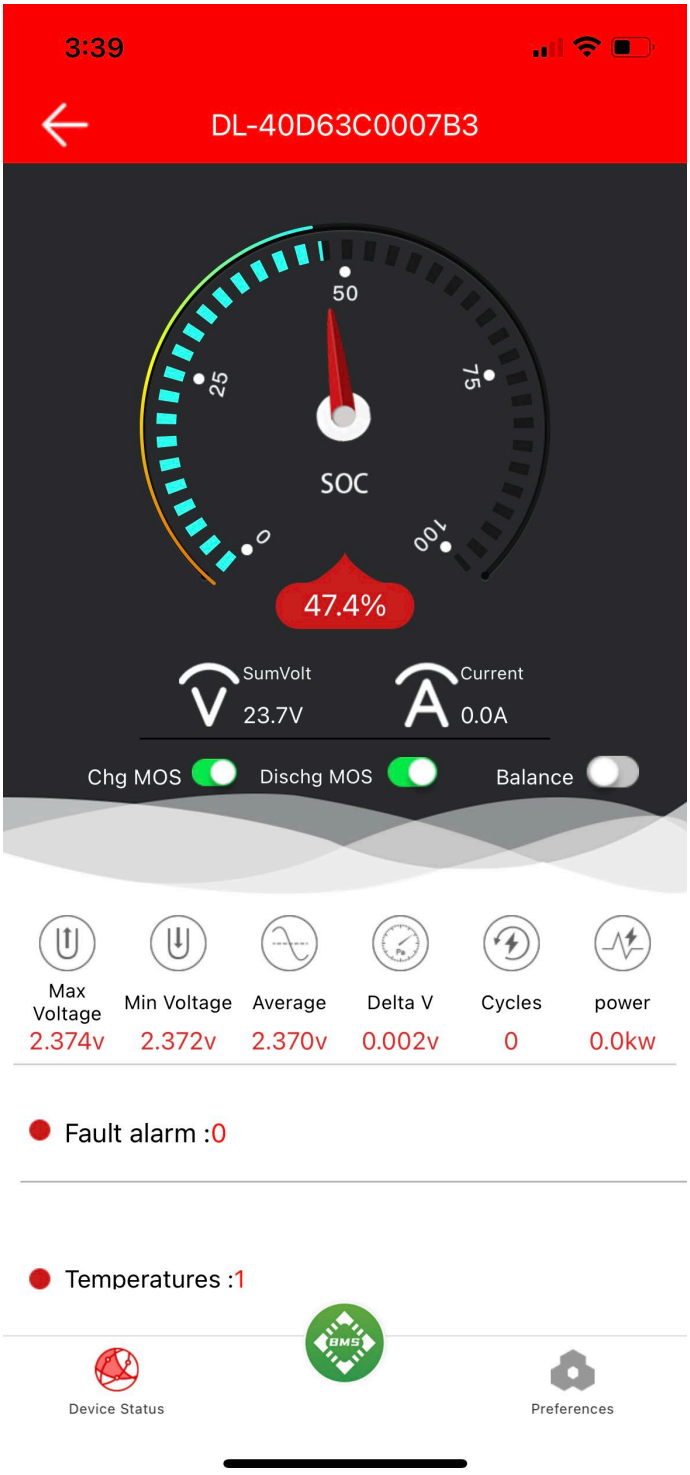

Figure 2.9. 300A Smart BMS communication screen 1.

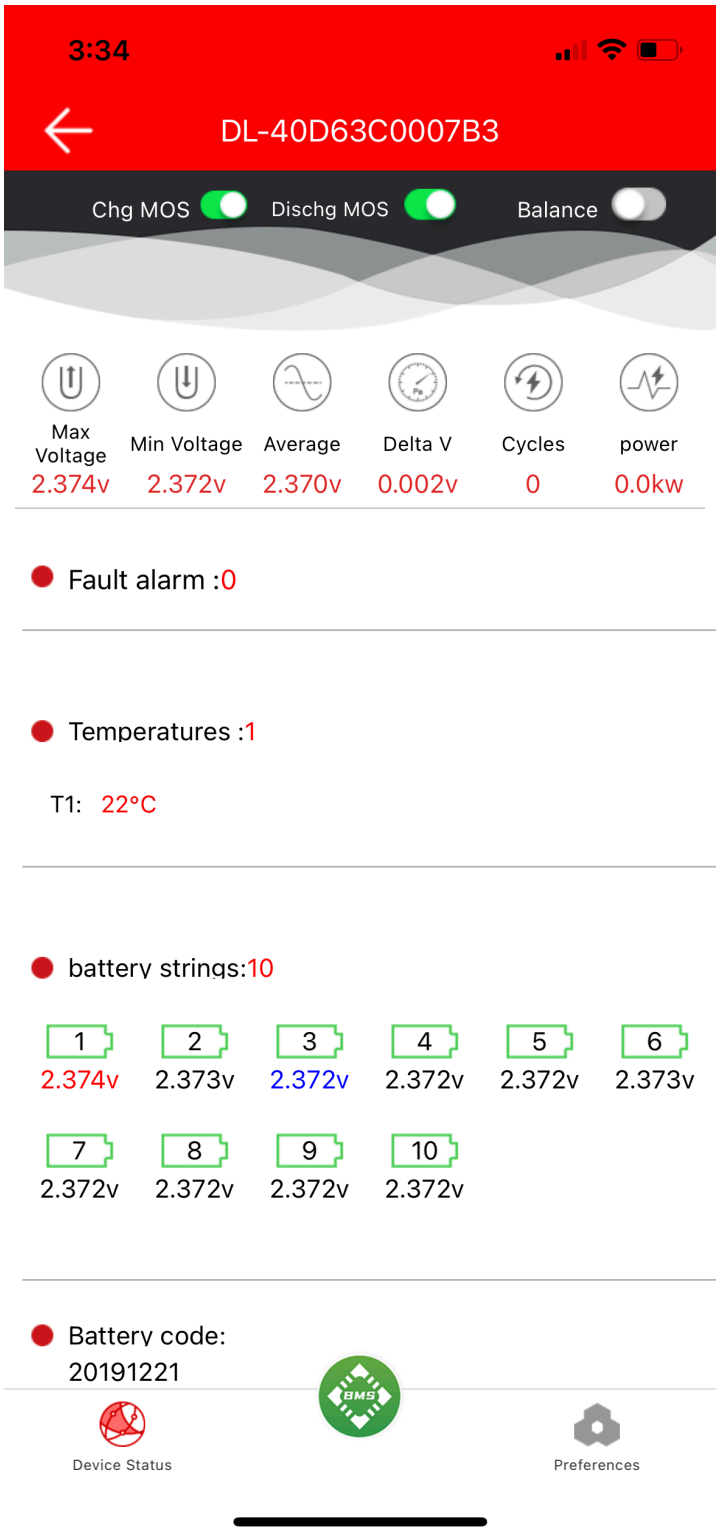

Figure 2.10. 300A Smart BMS communication screen 2.

The individual cells' voltage within a battery module is indicated, highlighting the highest and lowest cells. If one of the cells is above or below the acceptable threshold of 2.85V and 1.7V respectively shown in Fig. 2.8, the BMS actively balances the cells to evenly distribute the state of charge (SOC) of each cell by transferring energy from higher charged cells to lower charged cells to the other cells [23]. These BMSs use active balancing vs. passive balancing because active balancing conserves energy within the battery whereas the resistors are used to dissipate energy of higher charged cells to keep the cells relatively even. Balancing the cells are important in battery longevity and also battery capacity as the battery only has as much energy capacity as the cell with the lowest charged cell [24]. This only occurs when the differences between cells is above the threshold parameter, which is set as .002V as shown in Fig. 2.9 and Fig. 2.10. The other 100A BMSs are doing this as well but with a third of the current rating and cannot be communicated with like the 300A smart BMS. For instance, its parameters are set in factory and cannot be changed or even viewed whereas the smart BMS configuration, parameters, and status can be monitored in real time, in this case using Bluetooth. The temperature inside the battery cases is displayed as well. When a condition occurs that causes the BMS to "trip", the BMS opens the circuit where they would need to be reset to be able to draw power from the traction battery pack.

CHAPTER THREE

Chevy Bolt's Autonomous Vehicle Speed System

Programmable Throttle

Kenneth Ulibarri and Patrick Brantz were previous M.S. students working with Dr. von Jouanne on the programmable throttle project and I assisted. The testbed for the programmable throttle consisted of a Chevy Bolt placed on the dynamometer and a computer connected to the electric vehicle through a USB to the on-board diagnostics (OBD) port. The Chevy Bolt uses a sensor for reading throttle input by the driver. Electrical signals are sent to the car's electric control unit (ECU) that then commands the car to accelerate corresponding to the signals received by the ECU [25,26]. The pedal is not reading or calculating force, but instead the ECU is only reading the position of the throttle pedal. With that being the case, Kenneth and Patrick disconnected the factory throttle pedal from the car and connect the ECU with the specially designed and built programmable throttle. The programmable throttle is contained in a small grey box and has a silver lever on the inside, where its position will act as the pedal's position. The lever is manipulated with a small servo that runs on 5V and is only in contact and can interact with the lever when the solenoid, located on the outside of the grey box shown in Fig. 3.1, is

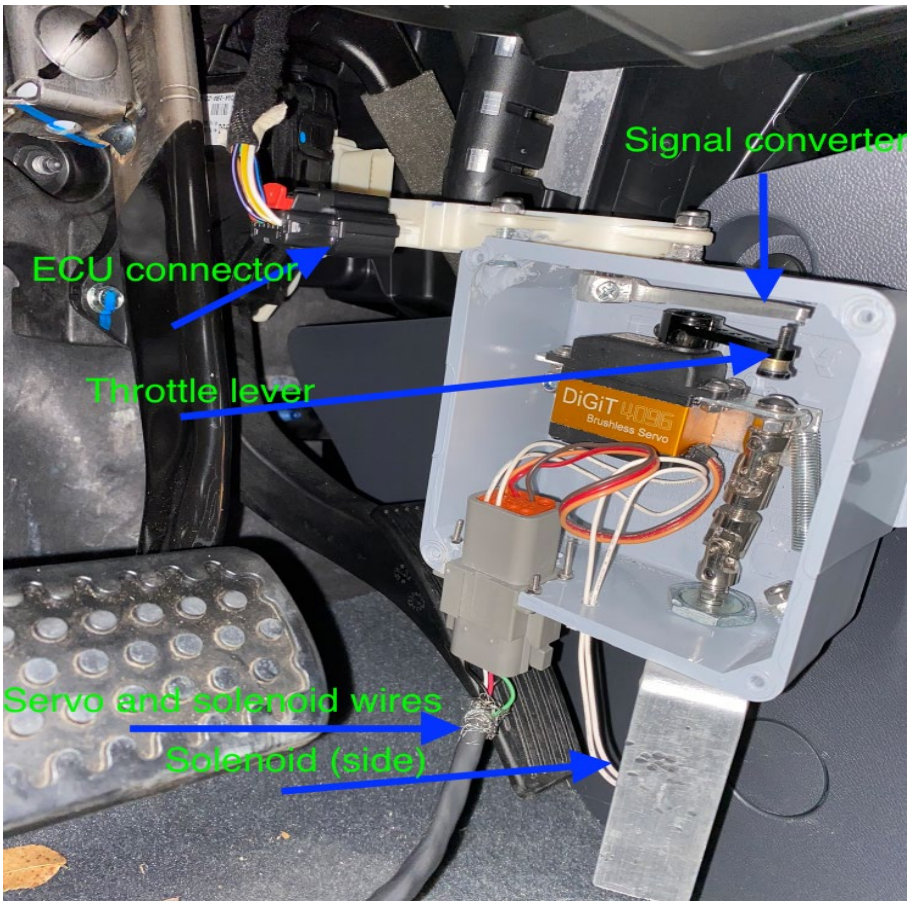

Figure 3.1. Programmable throttle connected to Chevy Bolt.

energized. When energized, the solenoid pushes the servo arm in reach of the lever, where the solenoid is the long silver piece in the bottom right of the grey box and the servo is the black unit in the same grey box. When a command is sent to the servo from the LabView software on the laptop to increase vehicle speed, power is supplied to the solenoid and the servo lowers the silver lever proportionately to the acceleration demand signal calculated by LabView. Conversely, to reduce vehicle speed, the signal sent to servo will raise the servo arm, raising the lever and reducing the acceleration of the vehicle. The silver lever is constantly being pushed upward by a spring, where without the servo resisting it, the lever would return to the initial position of no torque demand

from the vehicle. This is an important concept for fail safety. As discussed above, for the servo arm to even come into contact with the lever, the solenoid has to be energized, so unless the vehicle is actively running the solenoid is disengaged by default. However, the user can use the interface to manually engage and disengage the solenoid at will independent of the simulation for system checks. By default, once the drive cycle simulation ends, the solenoid disengages and the lever is returned to the initial zero torque demand position by the spring. If there is an error like a missing system connection, LabView generates an error code and disengages the solenoid, which returns the vehicle to the initial zero torque demand position. Also, the user can click stop on the user interface, shown in Fig. 3.2, to disengage the solenoid or even physically press the big red button on the top of the circuit box that is drilled on the side of the table closest to the Dynamometer to stop and disengage the solenoid. The user interface displays values such as vehicle speed and target drive cycle speed. Also, the user can manually toggle the "Enable Throttle" button to engage (bright green) or disengage (dark green as shown in the Fig. 3.2) the programmable throttle on the Chevy Bolt, engaging the solenoid and disengaging the solenoid.

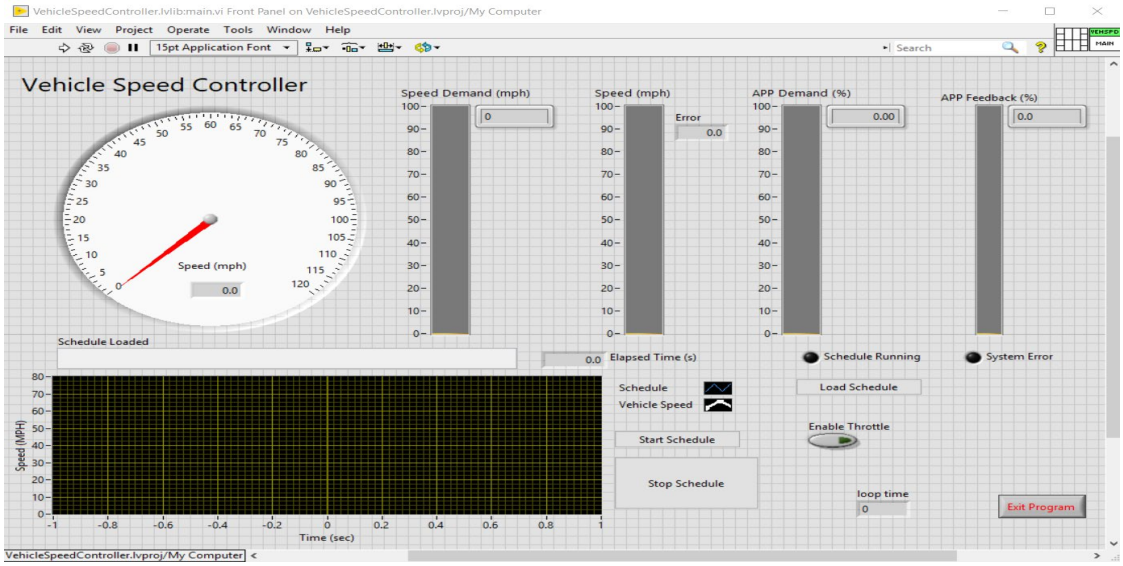

Figure 3.2. Programmable throttle user interface.

A drive cycle simulation works by creating a sequence of speeds of various durations and uploading the notepad file (shown in Fig. 3.3) into LabView. Once uploaded the user can click “start schedule” on the interface on the laptop. LabView takes the data points of (time, speed) and interpolates them for a continuous drive cycle schedule as shown in Fig. 3.4.

```
AutoSpeed Test Drive
"Test Time, secs"    "Target Speed, mph"
0    0
1    0
2    0
22   45
32   45
52   70
62   70
82   0
83   0
84   0
```

Figure 3.3. Example notepad drive cycle file to be uploaded into LabView.

A white dot represents the vehicle speed on a graph at a specific time superimposed with the blue line representing the speed of the uploaded drive cycle at any given time point shown in Fig. 3.4.

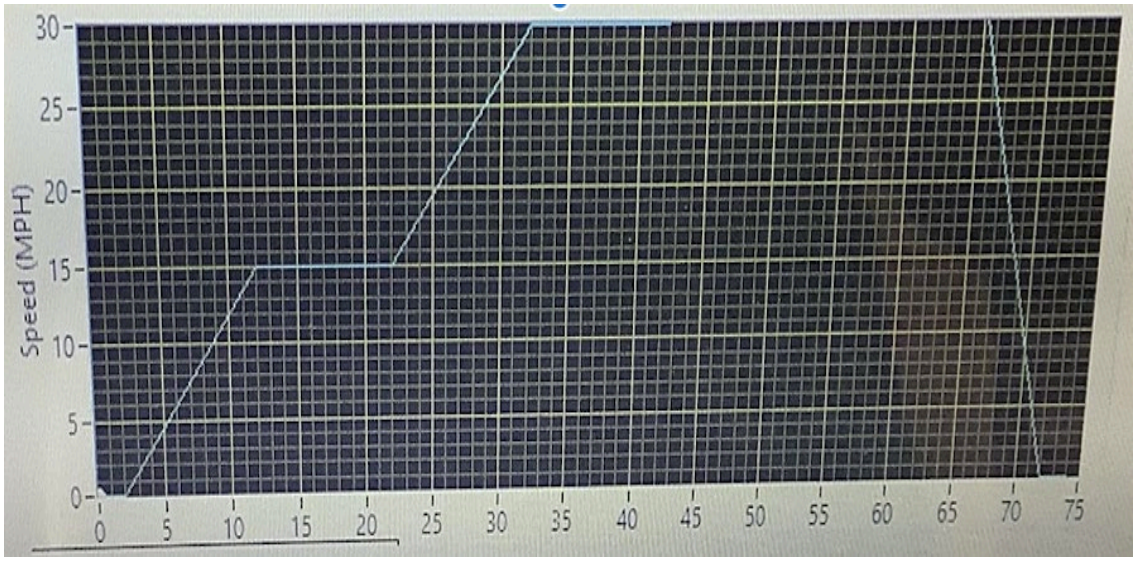

Figure 3.4. Example drive cycle displayed in LabView.

The goal is for the white dot to follow the speed profile throughout the entire drive cycle, which means that the actual vehicle speed is following the drive cycle speed at every point during the drive cycle. If the dot is below or above the blue line, there is a vehicle speed error that is fed into the proportional integral (PI) controller module in LabView to minimize the absolute value of the error. The acceptable error limits in vehicle speed when running the simulation are +/- 3 MPH above or below the drive cycle speed. To achieve the desired speed at all times, LabView is constantly reading in the vehicle speed from the Chevy Bolt's OBD through the black cable connected to the grey box in Fig. 3.1 and comparing it to the target speed at a given time in the drive cycle.

With the Chevy Bolt being an all-electric vehicle, it utilizes regenerative braking and so when implementing the programmable throttle, as long as the decrease in target vehicle speed of the drive cycle is not too steep, the regenerative braking was sufficient in following a drive cycle in acceleration and deceleration.

When testing the programmable throttle in LabView, there are modules used and dedicated for theoretical simulation and not experimental testing. There was a dry-run simulator module that looks and behaves somewhat similarly in overview to the testing user interface. However, the main simulator module uses the programmable throttle only and is also independent of the vehicle. This module allows for testing the programmable throttle's theoretical performance when given a drive cycle and vehicle's characteristics. During this non-testing simulation, the programmable throttle is disconnected from the Chevy Bolt and the grey box containing the servo, solenoid, and such and can be seen and observed as a drive cycle is simulated. Since the programmable throttle works on values and feedback from a vehicle's OBD to function properly in a drive cycle, a LabView module uses a road load equation to simulate the road load forces that a vehicle would experience like aerodynamic drag and friction shown in Fig. 3.5, then feed in the theoretical vehicle speed of the vehicle to the processing system. A lot of the values used to simulate the Chevy Bolt in preliminary testing without having to actually utilize the car for road load and air resistance forces, were provided by the vehicle's company. Again, to find the coefficient for drag, the Chevy Bolt was taken on the highway and was put in neutral, to avoid regenerative braking, after speeding up to 60 MPH to see how long it took to drop 10 MPH in speed to 50 MPH. Using this data, the coefficient of drag was found.

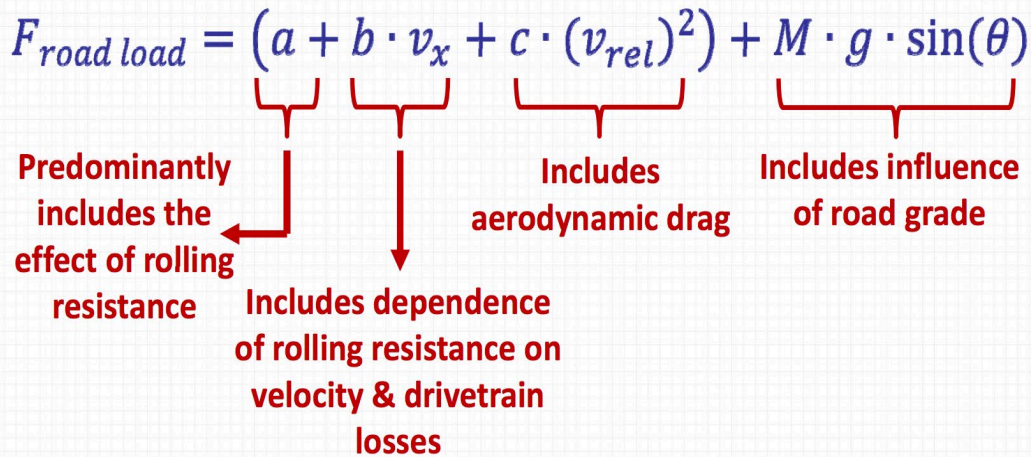

Figure 3.5. Road load forces [27].

In the beginning of the testing, the vehicle's speed oscillated around the target speeds very violently with much excess of overshoot when simulating a short and simple drive cycle. The first part of the problem was the dynamometer; it resisted the vehicle's change in speed in unexpected and unintended ways, resisting the electric vehicle excessively to keep the car from accelerating properly. When the car attempted to accelerate, the dynamometer treated the Chevy Bolt as though it was a very heavy vehicle and provided large road load forces in reaction. Additional tests and troubleshooting were conducted and eventually this undesired behavior was corrected by contacting the dynamometer's developers and having them run testing and they helped setup the software that solved this issue. After several iterations of troubleshooting, the test bed was properly able to simulate proper road load forces like rolling friction force and aerodynamic drag force. Eventually, the Chevy Bolt's velocity increased and so did the

aerodynamic drag force applied by the dynamometer by increasing the resistance of the dynamometer's grey rollers the tires rest on as shown in Fig. 1.1.

The other reason contributing to the oscillating vehicle speeds in attempting to match the target vehicle speeds, which is obtained from the drive cycle, was the tuning of the proportional integral derivative (PID) controller aiding in vehicle speed control A PID controller was used in the control system to accurately have the vehicle's speed track the drive cycle speed at some point in time within 3 MPH over or under the target speed, using the programmable throttle. An example illustration is shown in Fig. 3.6 [28].

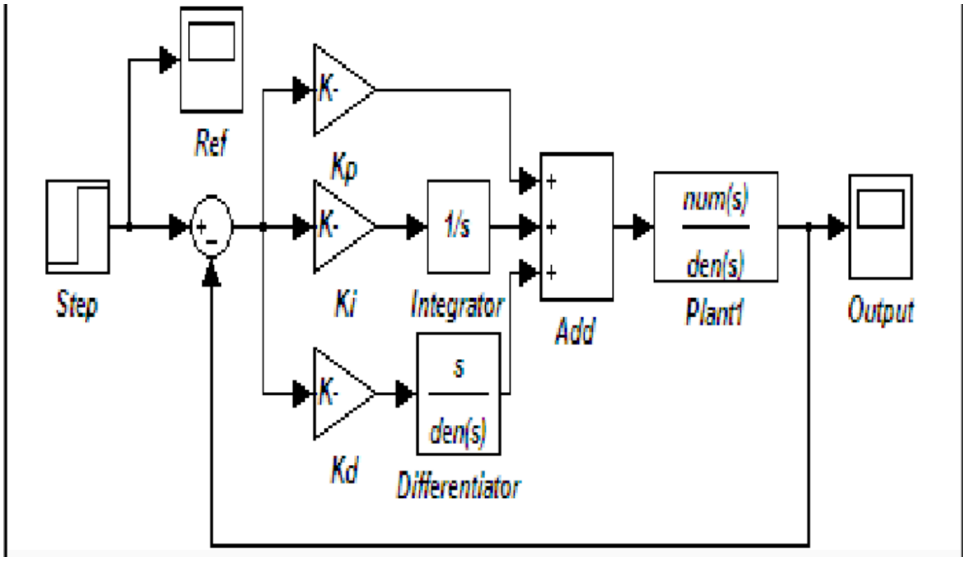

Figure 3.6. Autonomous acceleration system block diagram.

In Fig. 3.6 the box labeled step would be the input target speed from the speed limit that is compared with current vehicle speed and that error is passed to the PID control and to the plant function that would include the voltage sent to the inverter from the digital potentiometer that controls the motor, where the output is the vehicle speed that is fed back to compare with the target speed. After some testing the derivative

constant was set to zero effectively making the PID controller into a PI controller. In the beginning tests, the controller constants were very small and the vehicle was not ideal with quite amount of over and undershoot error due to the system not being responsive enough with the PID values. With trial and error, many tests were observed and the controller constants were adjusted. Also, the min and maximum throttle demand that can be accessed by the programmable throttle was also adjusted in these trial and error debugging sessions. Lowering the maximum throttle demand accessible did virtually eliminate the oscillations, however, the actual speed of the vehicle was constantly 5 MPH under the target values. Adjusting the minimum throttle demand accessible did not reduce oscillations or solve the constant 5 MPH error between vehicle and target speed. In most of the times adjusting the controller constants, the integral constant was left at 0.01 and the proportional constant was increased and decreased. The smaller P constant values led to the vehicle's speed oscillating with smaller amplitude but with larger frequency when compared to that of larger proportional constants tested.

For a very responsive system, a very large proportional constant was tried, but the vehicle's speed continued to over-oscillate for all of the drive cycle. Then, the integral constant was adjusted, trying values such as 0.1 and 1.0. These I-values seemed to make the oscillations worse. After more trial and error, the final constant I-value chosen was 0.045. The change in the integral constant is really small but makes a very large impact in the results. Using these constant PI values, the oscillations virtually stopped and the vehicle's speed tracks very closely to the target speeds, well within the 3 MPH over or under the drive cycle EPA constraint throughout a drive cycle. Also, there was some delay in between the drive cycle and the programmable throttle reaction, which

accounted for some difference error between the vehicle and target speed. This was caused from the vehicle's OBD saying the acceleration pedal position (APP) demand (value representing how much throttle is being used) was about 11.4 even when the car is not moving. This affected the programmable throttle from reacting as it should until overcoming that initial APP demand value. When the target speed increased beyond the vehicle speed, there was a delay before the programmable throttle reacted and increased the throttle. The delay was caused by the time it took for the torque demand needed to exceed the APP demand from the OBD, which needed a big enough difference between 0 MPH and the target speed. Once an offset was added to the APP demand in the vehicle speed control system to account for this, the delay problem was solved.

Programmable Brake

A different approach was used in designing and implementing the programmable brake compared with the programmable throttle. As mentioned earlier, the Chevy Bolt uses electrical sensors to control the accelerator. The physical force required to push the throttle down to change the position of the pedal, but the force itself is not the main concern for the Chevy Bolt's throttle system. The signals that are sent to the electronic control unit (ECU) based on the position read in by the sensor measuring the position of the pedal shown in Fig. 18 as the white component connected on the outside of the grey box that connects to the vehicle's black component sporting a red tab. The position signals are what are really important for this accelerating system. The force is necessary for driver to ECU communication. If the throttle pedal were stiffer or looser, the system would work in the same manner; by the position of the pedal. The only thing that changes

are how easy or hard it is for a driver to put the throttle in a certain position. Therefore, the throttle was removed and the position sensor was hooked onto something like a microcontroller to send the sensor signals, the vehicle would be able to accelerate accordingly to the signals rather than mechanical force. This is the basis used for designing the programmable throttle. However, for the programmable brake, physical force is needed which is why a linear actuator was chosen, so it could depress and release the brake pedal using physical contact.

The linear actuator is propped up on wooden blocks, which are mounted on a wooden board and is attached to the physical brake system on the vehicle as shown in Fig. 3.7. The wooden board is placed in front of the driver's seat and fastened.

Figure 3.7. Programmable brake.

When the linear actuator is fully extended, at the furthest position the brake position can go, it is physically impossible for the programmable brake to depress the brake pedal too far in error and damage the pedal. The wooden board is fastened on the car seat rails using bolts, screws, and Unistrut nuts on each side to support the linear actuator. The actuator is propped onto wooden blocks so the actuator can be about level with the brake pedal. The base of the actuator that is attached to wood is able to rotate freely, so as the brake is depressed and is tilted, the actuator will rotate mechanically with the brake pedal. The power and ground wires are connected to the motor driver for changing voltage polarity of the actuator. The motor driver is also connected to the 12 V power supply and the National Instruments data acquisition (DAQ) as shown in Fig. 3.8 for the pulse width modulation (PWM) for the linear actuator and an output pin that determines the polarity of how the power is supplied to the linear actuator, thus determining the direction of the actuator.

Figure 3.8. Data Acquisition (DAQ) setup.

The approach used for the programmable throttle could not be used to design the programmable brake. The Chevy Bolt uses a traditional braking system that is essentially a hydraulic system. This process is not electrically or computer based like the throttle system, but mostly mechanical. The force applied to the brake pedal is the principle in this system, where the amount of force necessary was determined by using a weight scale while applying pressure on it while it rested on the brake pedal to give an idea of the kind of linear actuator that was chose. Applying force to the brake pedal, pushes down on a lever attached to the brake in the vehicle. The force then goes through mechanical operations, multiplying the force on the brake pedal several times over onto the brake pads. The brake pads then clamp on to the vehicle's wheels, applying friction to the wheels to stop or slow down the vehicle [29]. The more force applied to the brake, the more friction applied to the vehicle's tires, and the faster the vehicle will decelerate if

moving. With this said, the design for the programmable throttle is centered around the necessary force needed to stimulate the brakes and complete the autonomous driving system for EPA drives cycles on a testbed.

The programmable throttle works very well in following a drive cycle with very little error with that error occurring on sharp transitions. When there is overshoot or the vehicle target speed decreases, the throttle approaches “0” position. This works very well, however when the target speed decreases sharply, faster than the regenerative braking can reduce the vehicle speed, the programmable brake is needed to completely and accurately follow an EPA drive cycle. In pursuit of altering a vehicle capable of following a drive cycle accurately and consistently without a driver, programmable brake and throttle systems need to be as separate as possible. With the programmable throttle already built and well-tuned to follow a drive cycle minus sharp deceleration, the goal is to minimally alter the throttle code when adding the brake to the system. For example, if the vehicle overshoots the target speed using the programmable throttle, the programmable brake is deactivated, and the throttle control system handles it using regenerative braking. The programmable brake should only enable when the programmable throttle cannot match the target speed at the times indicated by a drive cycle because of a decrease in speed at a rate that is steeper than the deceleration rate of regenerative braking. The regenerative braking affects the vehicle with a constant deceleration rate and this rate was used in the LabView code to determine when programmable brake needs to kick in. This value was found using an acceleration module in LabView with regenerative braking active. In Lab View, using the PID module, essentially the brake system does not need to activate unless the module output is negative. The more negative the PID output, the more the linear

actuator should depress the brake, and vice versa, the less negative the PID output becomes, the more the actuator should let off the brake until the brake pedal is no longer depressed. In keeping the brake and throttle systems from being activated simultaneously, the program is made through LabView to disengage one of the systems before engaging the other no matter the PID output. The throttle system is “disengaged” when the silver lever in the gray programmable throttle box is returned to its initial zero torque demand position, and the programmable brake system is “disengaged” when the brake pedal is no longer depressed.

The linear actuator extends when power is applied red-to-red and black-to black (red is positive and black is negative) and will continue to extend until either the power is cut off or the actuator hits the limit switch. The actuator has two limit switches, where one is at the maximum length the actuator can extend and the other is at the maximum that the actuator can retract. Once either of these limit switches is reached, the actuator no longer allows power to be supplied to it in that direction, therefore halting movement. To retract, a red-to-black and red-to-black connection has to be made when applying power. This change in polarity is the reason why the motor driver is used. With this said, after the brake has been active and now needs to deactivate, the throttle needs to activate, which is when the PID output becomes positive, power must be applied in opposite polarity in retracting configuration until the linear actuator no longer presses on the brake. Again, only after the brake is no longer being pressed down, will the autonomous system allow the throttle to activate and vice versa. This function is achieved in LabView by using flags that are enabled and disabled with these conditions being met. One of the obstacles of the brake system was the linear actuator not having position feedback like

the servo used for the throttle did. This is because it would be less than desirable for the actuator to depress the brake pedal too far or retract too far to be effective in proper braking on time without knowing how far extended or retracted the actuator is. As mentioned earlier, the idea is for the actuator to retract if the output of the PID module is positive and the programmable throttle system is disengaged, but stop retracting right when it ceases to depress the brake pedal. This is important because if the actuator retracts too much, when a steep deceleration happens on a drive cycle; there will be relatively significant delay between the command to engage the brake system and the programmable brake actually reducing the vehicle's speed. Thus, LabView commands the actuator to only retract if the PID output is positive when the vehicle speed is below 2 MPH. A positive PID output means throttle is needed. When the vehicle is below 2 MPH the programmable brake must be depressing the brake pedal because that is the speed it maintains without depressing the throttle or brake while in drive.

CHAPTER FOUR

Electric-Converted Tahoe's Autonomous Vehicle Speed Control System

Throttle and Brake Setup

The goal with the Chevy Tahoe was to expand upon the work that Ken, Patrick, and I had done on the Chevy Bolt with adding programmable steering and sensors. With the Chevy Tahoe being different from the Chevy Bolt architecturally, the approach for the throttle and braking system will be different but the idea will be somewhat similar. Firstly, the throttle will be controlled using the vehicle control unit (VCU) purchased for the Chevy Tahoe from Thunderstruck. The VCU has been configured to communicate through MATLAB so that all parts of the systems can communicate effectively (throttle, brake, steering, and sensors) and the user can monitor these systems, effectively tying these systems together. As been tested, MATLAB will send commands to the VCU, constantly requesting and reading in revolutions per minute (rpm) for vehicle speed and other variables used for calculations and decision making such as inverter current, voltage, and time stamp of each data output to MATLAB though a USB serial connection. This rpm value is converted to MPH by doing a gear ratio analysis, calculations shown in Fig. 4.1, from motor shaft to the wheels.

```
%% rpm to speed
maxvoltage=288;
maxcurrent=100;
maxpower=maxvoltage*maxcurrent;
GR=4;
power_rpm_curve_slope=32;
wheel_radius=.3683;
maxrpm=maxpower/power_rpm_curve_slope;
vehicle_speed_MPH=(maxrpm*2*pi*wheel_radius*60)/(1000*GR)*.621371;
```

Figure 4.1. Motor rpm to speed in MPH.

For the programmable throttle implemented on the Chevy Bolt, the silver lever controls the electrical signals sent to the ECU determining the torque demand to be sent to the motor. Looking at the throttle pedal in Fig. 4.2, there are wires connected to the VCU from the pedal for the converted Chevy Tahoe. The throttle pedal is a transducer that outputs a voltage level depending on the how much the pedal is depressed.

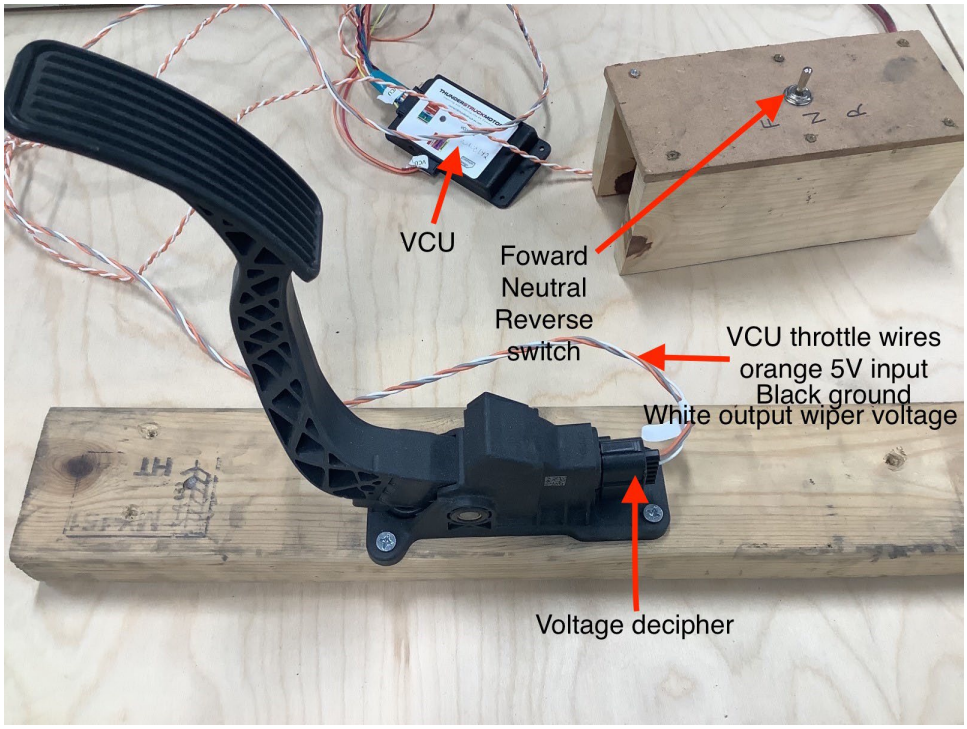

Figure 4.2. Throttle VCU setup.

The voltage decipher is disconnected from the converted Chevy Tahoe throttle pedal shown above and sent the required voltages to the VCU through these wires dictated by how much torque demand is desired from system code running in MATLAB without the need of the pedal or some sort of mechanical lever. In essence this approach is bypassing the mechanical workings that correspond to the electrical signals sent to the ECU, but sending the ECU the signals directly from the system software. The same approach is used for the braking system. Currently the vehicle's brakes are composed solely of regenerative braking. Voltage signals are sent to the VCU to request varying amounts of counter torque to slow down the motor depending on the signal voltage.

To send the analog voltage signal to the VCU to control the AC motor's speed and acceleration, an Arduino was connected to MATLAB and a digital 6-channel potentiometer to get an analog range of the voltage output (0V-5V) from the digital output ports of the Arduino (Fig. 4.3 and Fig. 4.4). With the throttle pedal shown in Fig. 4.2, voltage decipher is disconnected and the throttle wiper (white wire) connection from the VCU input connected to the channel 1 wiper of the digital potentiometer, the vehicle speed and acceleration is manipulated.

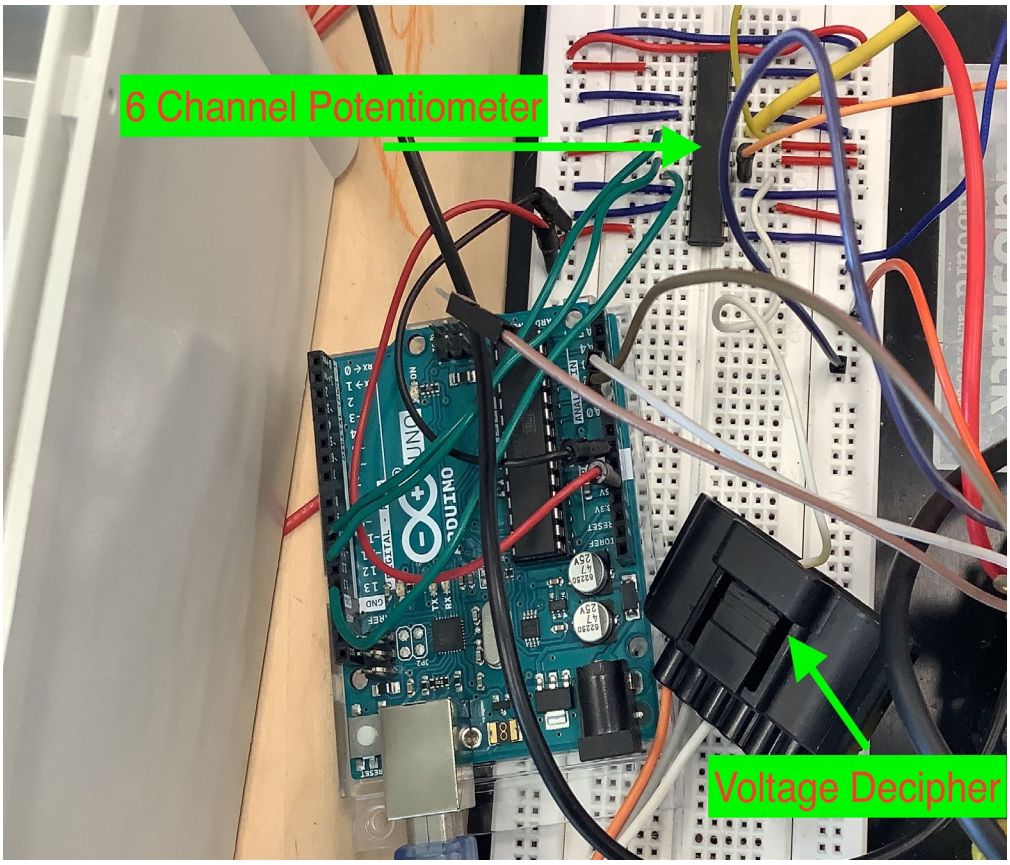

Figure 4.3. Arduino to digital potentiometer to voltage decipher.

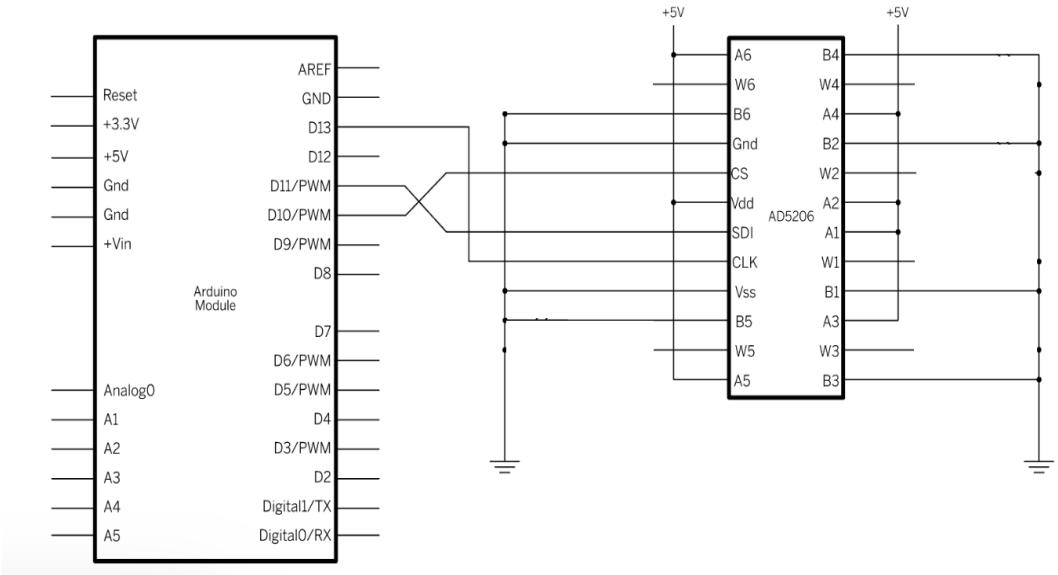

Figure 4.4. Arduino to digital potentiometer (AD5206) to voltage decipher schematic.

MATLAB communicates with the VCU by USB serial communication, constantly updating its asset framework (AF) message string shown in Fig. 4.5. This message outputs: a time stamp, voltage, current, torque, and motor rpm. MATLAB takes this data and calculates the vehicle speed in MPH from the motor rpm (Fig. 4.1), takes voltage and current at each iteration and multiply the two to get instantaneous power at that iteration and subtract it from energy of the working energy from the traction battery pack to predict the range of the vehicle (Fig. 4.6). The program also calculates the relative range which means the range from fully charged (288V) until the motor will no longer operate, which is under 230V instead of the range variable which the range of the vehicle it the motor works on the full 288V to 0V voltage range. The UQM software in Fig. 4.7 is a screenshot of the UQM electric motor software program which gives greater insight into the status of the inverter for instance indicating if there is a fault with the inverter setup that will cripple operation of the motor and what kind of fault it is.

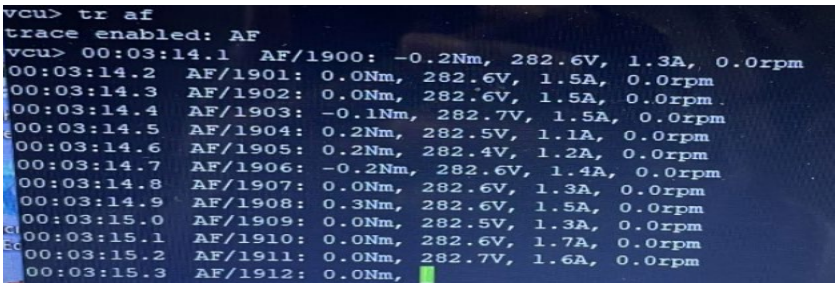

Figure 4.5. Serial interface of VCU AF trace output.

```
battery_energy=50*inverter_voltage/1000; %kWhr battery module capacity 50 Ahr
inverter_power=inverter_voltage*inverter_current;
ave_kwh_per_mile=.346;
avemaxrange=battery_energy/ave_kwh_per_mile;
battery_energy=(battery_energy-hr*inverter_power/1000);
range=battery_energy/ave_kwh_per_mile;
range_relative=((288-230)/288);
```

Figure 4.6. Predicted range of vehicle

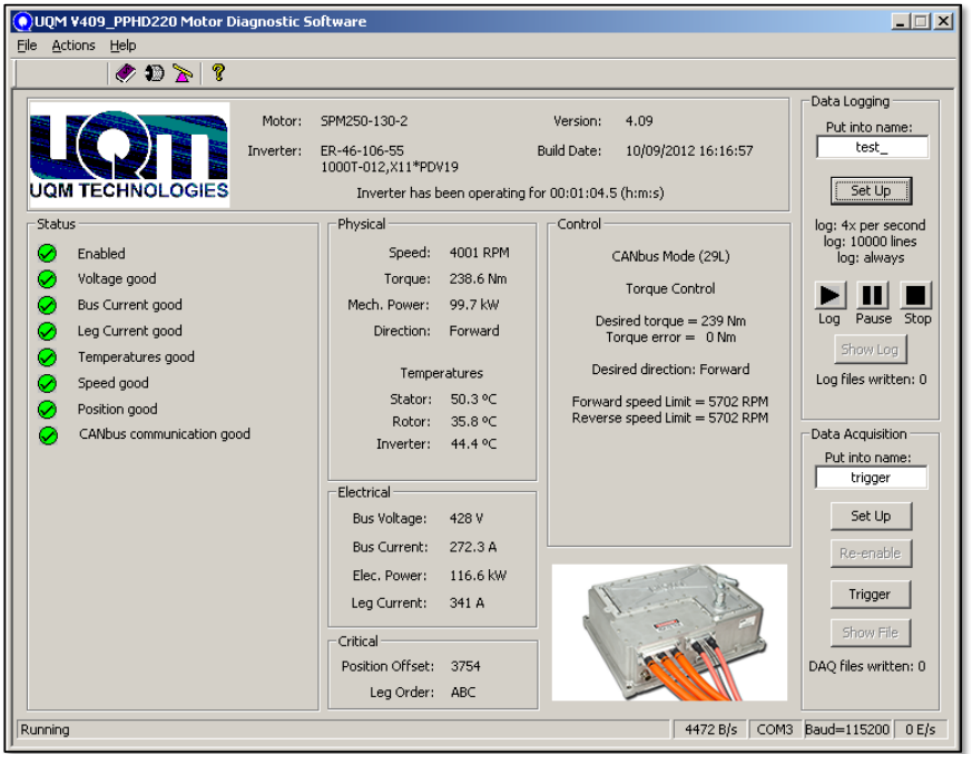

Figure 4.7. UQM diagnostic software.

The inverter also logs data including operation failures. High fidelity motor operational data such as currents, voltage, torque, rpm and can be used for a higher degree of troubleshooting such as communication errors; the VCU rather than the UQM software because the UQM software doesn't output the desired data needed that can be extracted serially that be readily accessible.

The actual voltage range set in the VCU software for the throttle is 0.8V to 4.59V where 0.8V is 0% throttle (APP) and 4.59V is 100% throttle (APP). A built-in inverter safety feature is that if the throttle voltage is not 0% or slightly lower, the VCU will give an error to open the high voltage contactor, denying high voltage and denying motoring. Also, if there is a fault with the inverter, the contactor will open disabling the connection with the high voltage input. To send the analog voltage signals to the VCU to slow down

the motor, sending a negative torque demand to the motor, an analog voltage signal between 0V and 5V is sent to the VCU through a separate channel of the potentiometer. Based on to the configuration of the VCU connection to the Forward/Reverse/Neutral input described in Fig. 4.2, removing the wiper F&R/brake VCU input and connecting it to channel 2 wiper of the digital potentiometer, the amount of regenerative braking can be requested to slow down the vehicle.

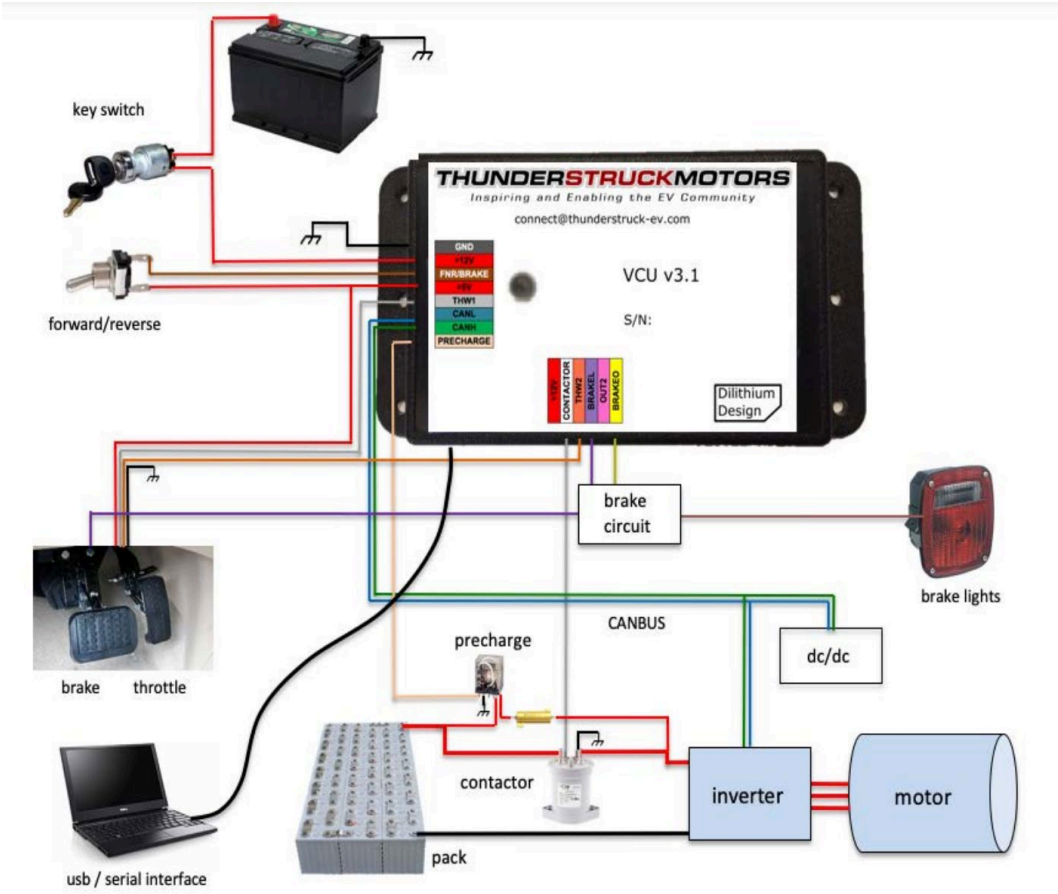

Figure 4.8. VCU wiring diagram [30].

Fig. 4.8 is a high-level architecture of the low voltage and high voltage components of the converted Chevy Tahoe power train controls. In the inverter firmware,

if a nonzero voltage signal is on the F&R/brake port on the VCU port shown in Fig. 4.8 while variable regenerative braking is set to active, the throttle signal is ignored because regenerative braking occurs as the brake signal has a higher priority.

Power System Safety Design

In addition to the BMS regulated voltage and current, MATLAB also monitors the inverter voltage and current and ends the program in the case of overvoltage or undervoltage and reduces the vehicle speed when the magnitude of the current goes above 100A, setting a flag to let a passenger know that there is an issue and what issue it is. Parameters are also set in the inverter software itself to avoid overcurrent such as limiting the motor to 695 rpm. The value was calculated as the max rpm using the power vs. motor rpm plot in Fig. 4.9, looking at the left most black linear curve with a zero rpm starting speed to get the differential relationship to calculate the speed at maximum power with the voltage being 288V and current being 100A (28.8kW), the black line was used to get an estimate on the current rating of BMS needed to be purchased to be able operate the vehicle properly.

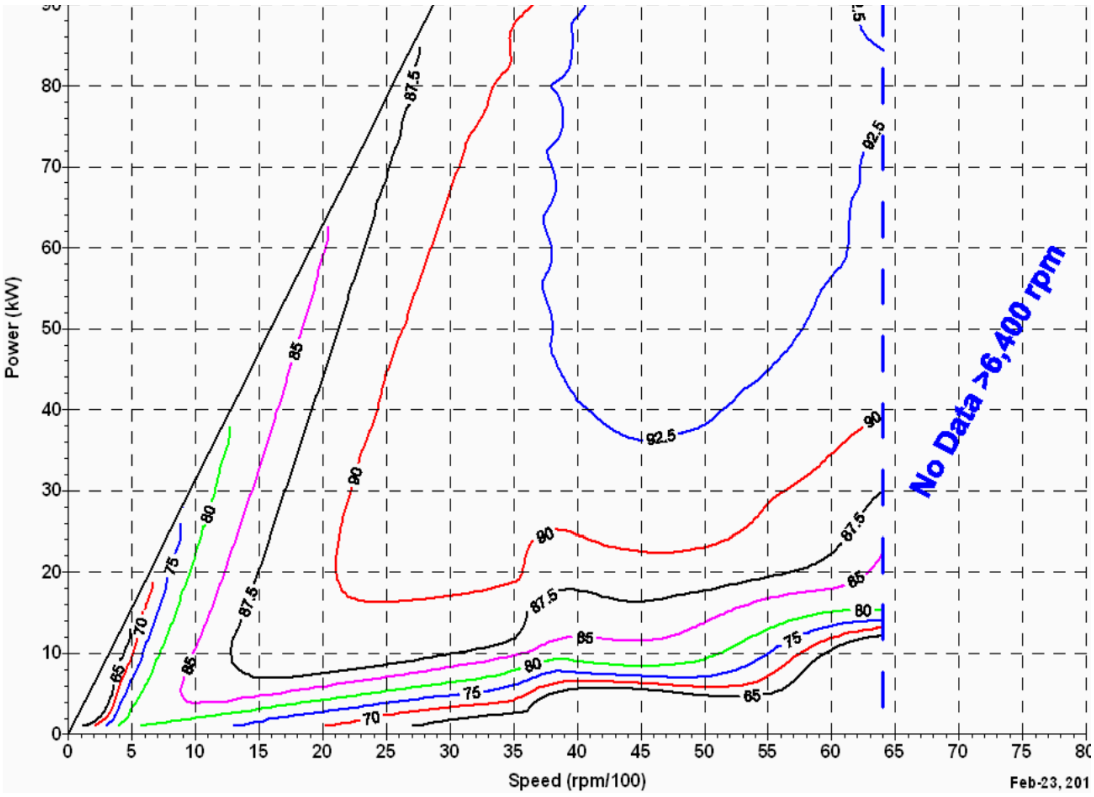

Figure 4.9. Power vs. motor rpm graph [31].

There are switches connected to the traction battery pack shown in Fig. 4.10 and Fig. 4.11 and the servos' and VCU's battery, 24V and 12V respectively. These switches are for safety, for instance, cutting power from the VCU 12V supply opens the contactor that allows the inverter to accept power from the traction battery pack; this is also true when disconnecting the USB hub that contains the Arduino that is connected with the digital potentiometer, because 0V is now across the throttle wiper and causes a fault with the inverter. In case these shutdown methods fail, there is a switch for disconnecting the traction pack battery entirely.

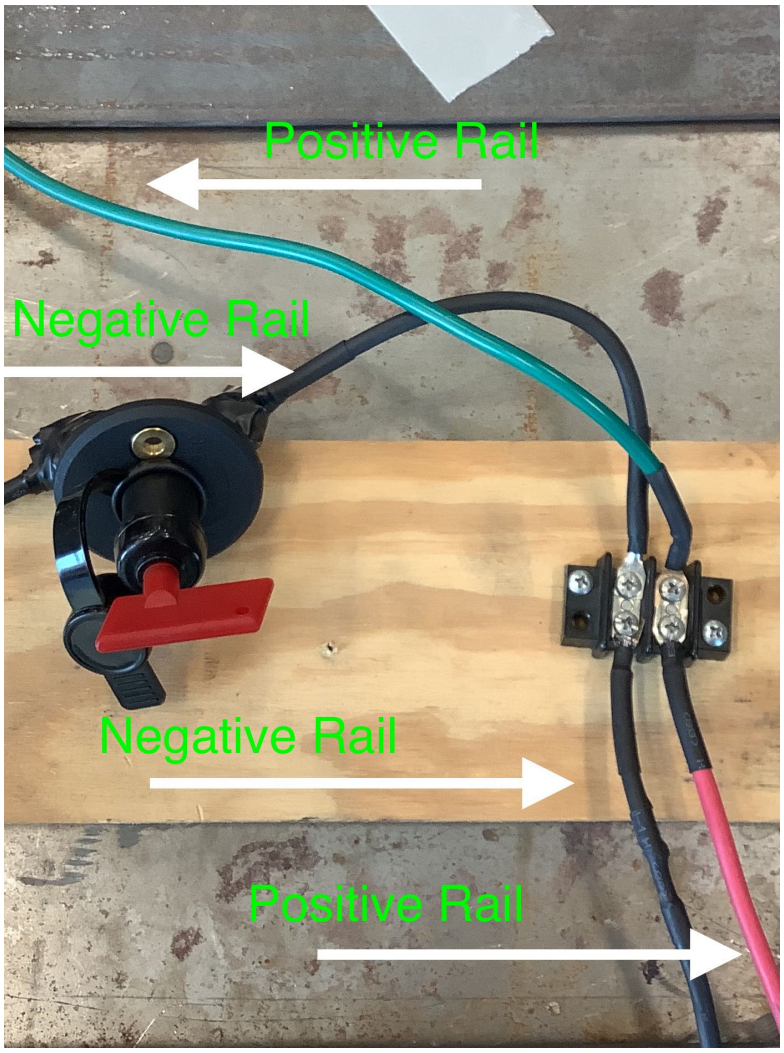

Figure 4.10. Traction battery pack on/off switch.

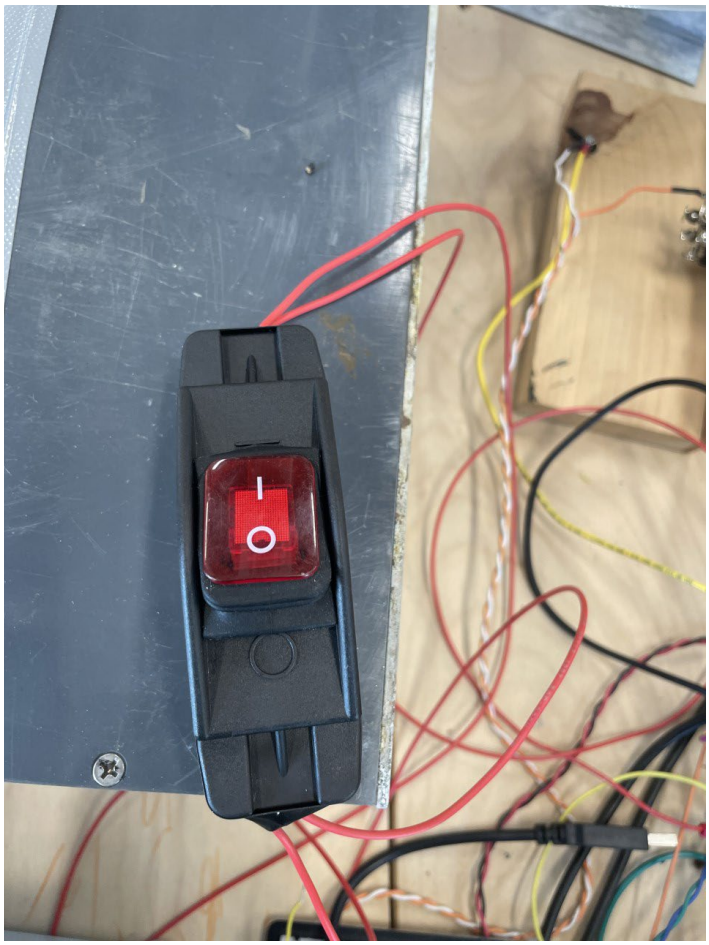

Figure 4.11. Servos and VCU battery supplies on/off switch.

PID Tuning For Vehicle Speed Control

Using the same dynamometer test bed setup as with the Chevy Bolt, the Chevy Tahoe was pushed onto and strapped onto the dynamometer. In a similar testing process as with the Chevy Bolt, the PID controller that was coded in MATLAB and tuned to get some good values to get the vehicle to converge onto a target speed (MPH) as the program iterations increase in a laboratory setting as shown below in Fig. 4.12: Throttle_demand=Throttle_demand+Kp*error(count)+Ki*trapz(time,error)+Kd*((error(count)-error(count-1))/(time(count)-time(count-1))). For the integral contribution in the

PID controller, the trap function was used to integrate discrete data points. For the derivative contribution, the program divides the speed difference between the current data point and the previous data point by the difference in the time between the two.

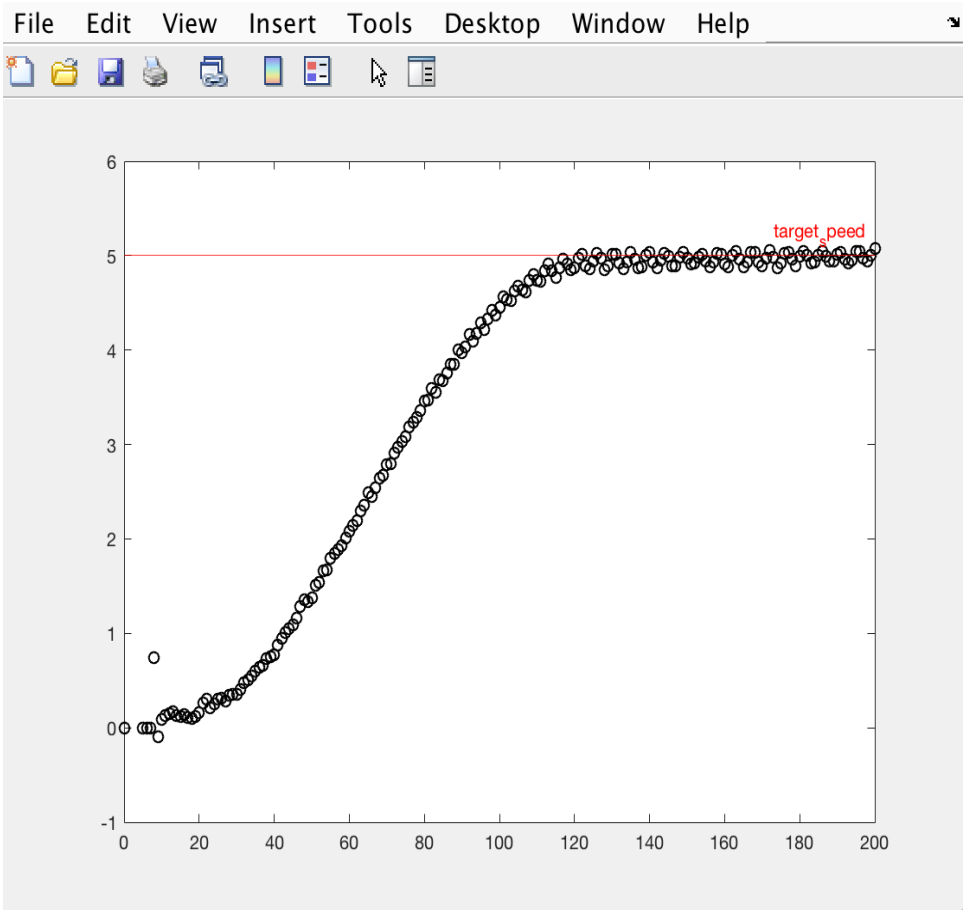

Figure 4.12. Tuned PID constants with a negative I constant.

The results in Fig. 4.12 are with nonzero P and I constants with zero for the derivative constant. Working with tuning both vehicles, one can very easily discern the differences in the two vehicles. The Chevy Bolt was much more responsive with increasing the throttle voltage. The Chevy Tahoe's throttle demand had to be increased to over 53% just to get the vehicle to roll and so that led the vehicle speed to overshoot and then undershoot the target speed quite dramatically because of the error it accumulated

starting from 0% throttle until it actually started moving. To remedy this, while running, the 0% throttle was now set to the previous 53% throttle voltage. This, improved results greatly and was able to achieve the convergence shown in Fig. 4.12. However, with the Chevy Bolt being a commercially finished product, it behaved consistently when testing, but initially, the Chevy Tahoe reacted with enough variance to get completely different results as shown in Fig. 4.13 in subsequent runs of the vehicle under the same conditions.

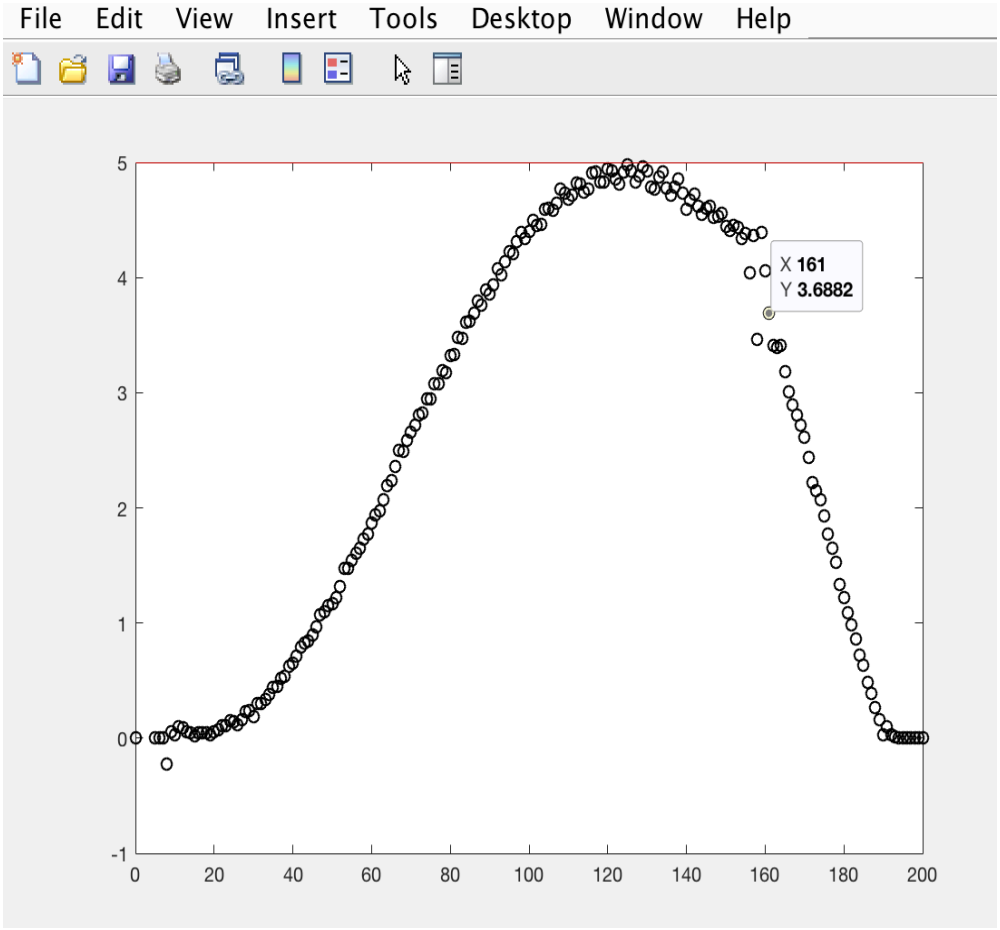

Figure 4.13. Same PID values as Fig. 4.12 with completely different results.

The same exact program and PI constants were used but the results differed greatly. For both Fig. 4.12 and Fig. 4.13, a negative I constant was used which helped

eliminate the oscillations. With the way the PID control is written, the array holding the error values over time are cleared when the vehicle speed equals the target speed or crosses the target speed line as depicted as the red line in Fig. 4.12 and Fig. 4.13. So if the vehicle speed can equal or be greater than the target speed by certain amount of iterations, the array clears and the influence by the integral part of the PID controller is minimum and the vehicle speed converges as desired as in Fig. 4.12. Conversely, if the vehicle speed does not reach the target speed by a certain number of iterations, the array continues to grow and the integral contributor of the PID controller becomes more influential than the proportional piece, and with the I constant being negative, the requested percent throttle will decrease as the iterations decrease. This is shown in Fig. 4.13 as the vehicle speed gets really close to the target speed but does not actually reach the target speed. To fix such a drastic variance of results with just little vehicle behavior, entering a D constant and making the I constant positive. Values chosen in Fig. 4.14 are P-.5, I-.01, and D-3.2.

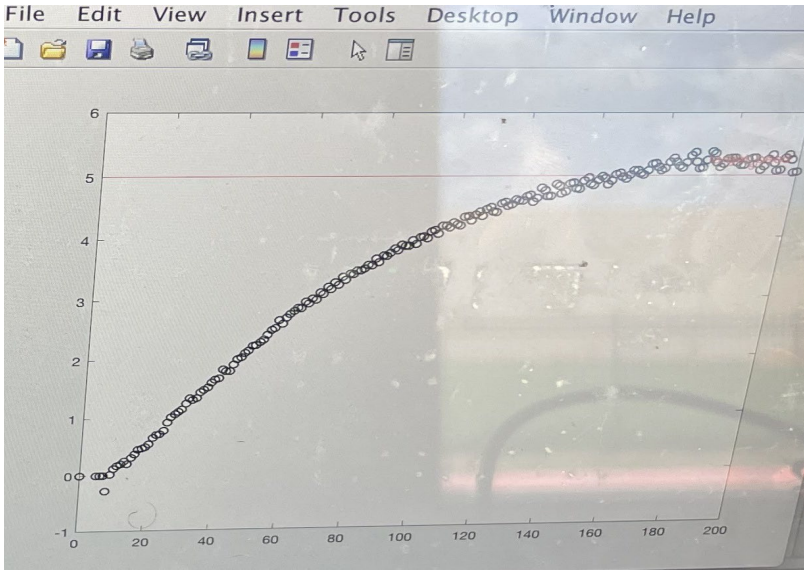

Figure 4.14. Tuning PID with values: 0.5, 0.01, and 3.2.

In Fig. 4.14, the vehicle speed approaches the target speed slower than in Figs. 4.12 and 4.13 and overshoots the target value. Upon more testing, values: P-0.7, I-0, D-3.2 were chosen; the idea was to reduce overshoot by eliminating the accumulation error or integral I-term to increase the proportionate P-term to increase the responsiveness of the system by being capable of converging to the target speed faster as well as the ability to more quickly react and compensate for any overshoot and undershoot shown in Fig. 4.15.

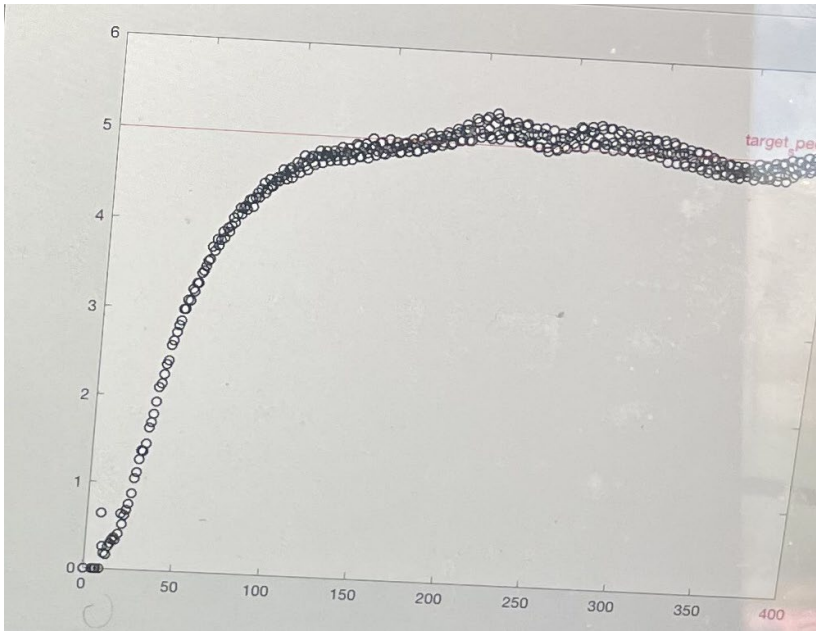

Figure 4.15. Tuning PID with values: 0.7,0,3.2.

The performance of the autonomous acceleration system has improved with these values, however, in an attempt to reduce the oscillation, more values were tested and the final values landed on are P-1, I-0, D-3, increasing the systems responsiveness and reducing the derivative terms which reduces rise time and with a lower derivative value,

the vehicle has less oscillation about the target speed. The vehicle speed converges to target speed consistently and reliably with very low oscillations as displayed in Fig. 4.16.

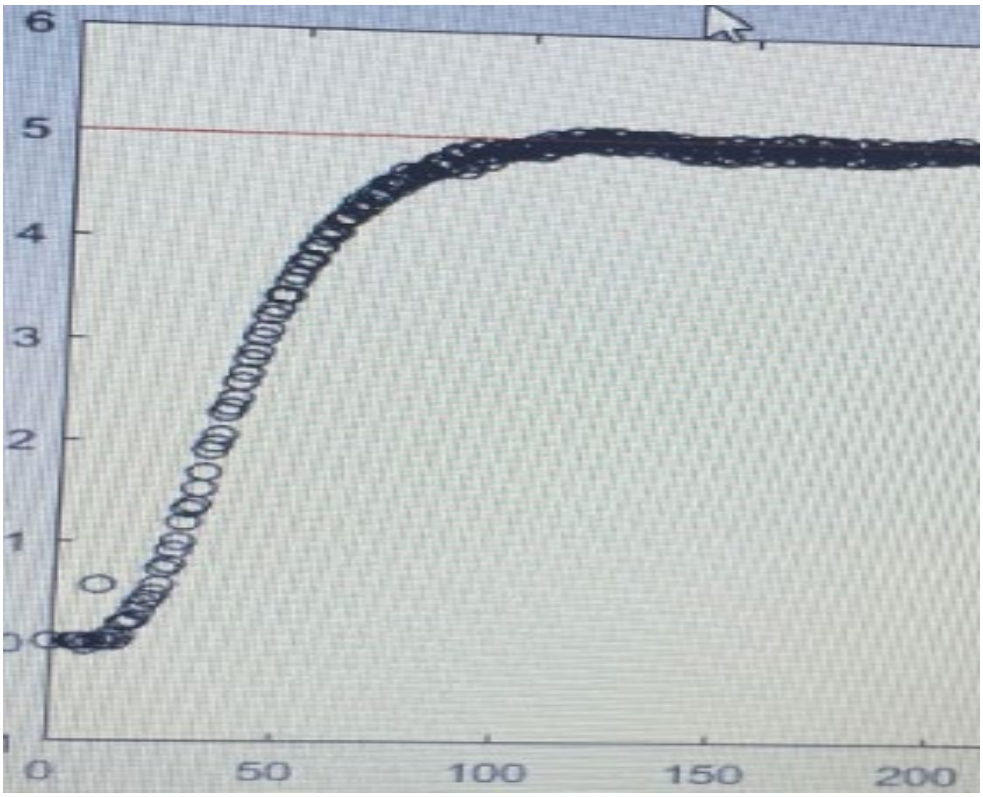

Figure 4.16. Tuned PID with nonzero P and D constants and 0 for the I constant.

CHAPTER FIVE

Sensor Implementation

Global Positioning System (GPS)

For the vehicle to be able to know where it is in the world and where the destination is, a GPS unit (Fig. 5.1) is used by integrating it into MATLAB. For the vehicle to be able to sense and react to the environment around it, sensors are embedded in the vehicle. Ultrasonic sensors are used to determine the distance of objects from the vehicle. Serially connecting the GPS unit to MATLAB through USB, the module outputs 6 different GPS messages with no specific order or rhythm. MATLAB parses through the $GPGGA, $GPGLL, and $GPRMC to find the latitude and longitude coordinates. The other 3 coordinates do not give latitude and longitude of the module.

$GPGGA - Global Positioning System Fix Data

$GPGLL - Geographic position, latitude / longitude

$GPGSV - GPS Satellites in view

$GPVTG - Track made good and ground speed

$GPRMC - Recommended minimum specific GPS/Transit data

$GPGSA - GPS DOP and active satellites

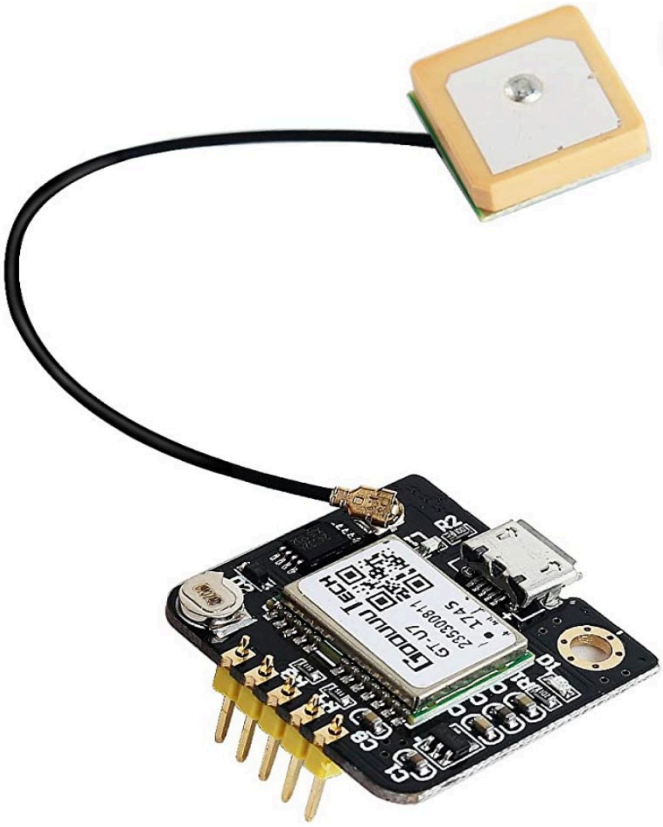

Figure 5.1. GPS unit.

Ultrasonic Sensors

Two Arduino boards are connected to MATLAB and 9 ultrasonic sensors are connected to one of those boards. MATLAB commands the Arduino to send an ultrasonic pulse and convert the reflected wave delay to a calculation of the distance to the target. The range of each sensor is about 15 feet. The location of these is shown in Fig. 5.2.

Figure 5.2. Chevy Tahoe sporting ultrasonic sensors.

Sensors labeled 1 and 9 are turned to the side and are oriented that way and placed so low to detect curbs and other path trace makers. In addition to these sensors, the onboard laptop connects to a USB camera with its images processed in MATLAB and used for “you only look once” (YOLO) object detection and classification. In adding ultrasonic sensors to the processor load, it is actually visually apparent that the program processing speed decreases once adding over about 5 of these sensors. A lot of these sensors were used to compensate for the narrow angular view that the sensors can provide.

Camera and YOLO (you only look once)

A USB camera is integrated in the control system for the vehicle to be able to appropriately adhere to various different road signs a driver would need to abide by very quickly and accurately. The classification neural networks, YOLO [32,33,34] object detector and classifier as shown below in Fig. 5.3 completes an analysis and annotates on each image fed into MATLAB, including boxes and labels around the objects in each image. YOLO is used to recognize an object and to be able to differentiate between different objects such as discerning between a stop sign and yield sign. YOLO scans an image and places a box around a determined object and then displays a label for each object, determining what kind of object it is, using a neural network as introduced in the literature review, affecting the algorithm of how the vehicle will behave. For example, if the vehicle recognizes a stop sign is a certain distance in front of it, it will come to a complete stop and remain for 3 seconds and then continue to operate as normal. In this case YOLO acts like an interrupt to the typical operation of the vehicle. Also, YOLO only scans the image once and so this process is happening very quickly making it ideal for vehicle implementation for autonomous application. YOLO has proven to be very accurate in testing and demonstrating the viability [34].

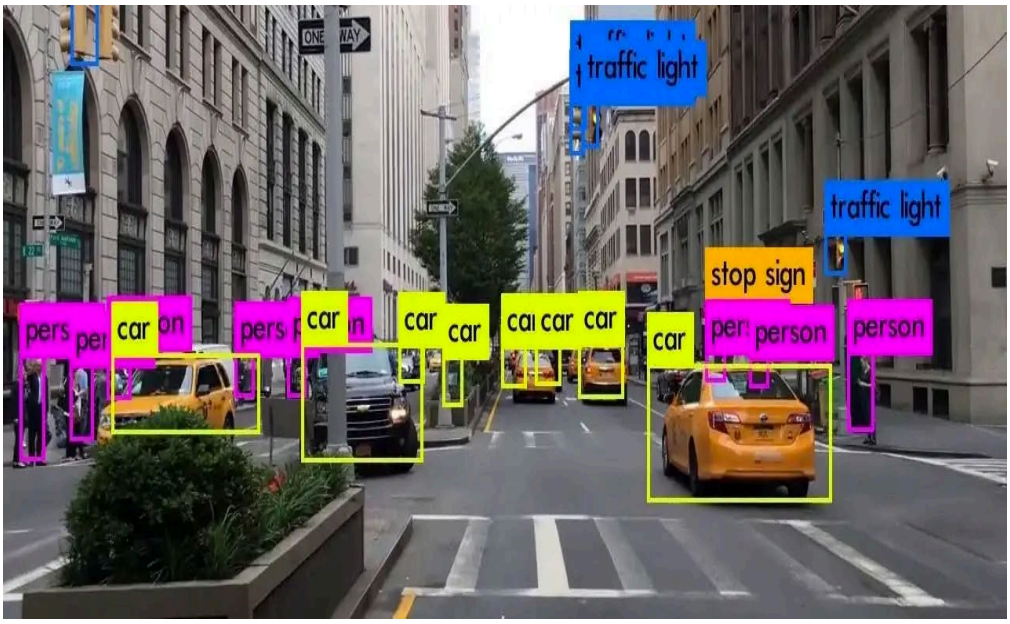

Figure 5.3. YOLO object detector and classifier [35].

In actual implementation of this project, YOLO through MATLAB was used to detect a stop sign, feed that information to the rest of the program and make the appropriate decisions from the information. For instance, once a stop sign is detected, the program then decides when the vehicle should begin its decent to a stop by how large the box outline is around the stop sign. This method of indirectly measuring the distance between the vehicle and stop sign is used because adding too many ultrasonic sensors will slow down processing speed and in turn disrupts the serial communication timing of the other peripheral devices connected to the computer affecting every other system. Also, the very narrow view of the ultrasonic sensor makes for a larger chance the stop sign will not be detected reliably. To use a stop sign for the obstacle course setup, a stop sign was purchased and fastened on a rail and an actual YOLO analysis done on the onboard camera shown in Fig. 5.4 with some statistics on the image shown in Fig. 5.4 displayed in Fig. 5.5.

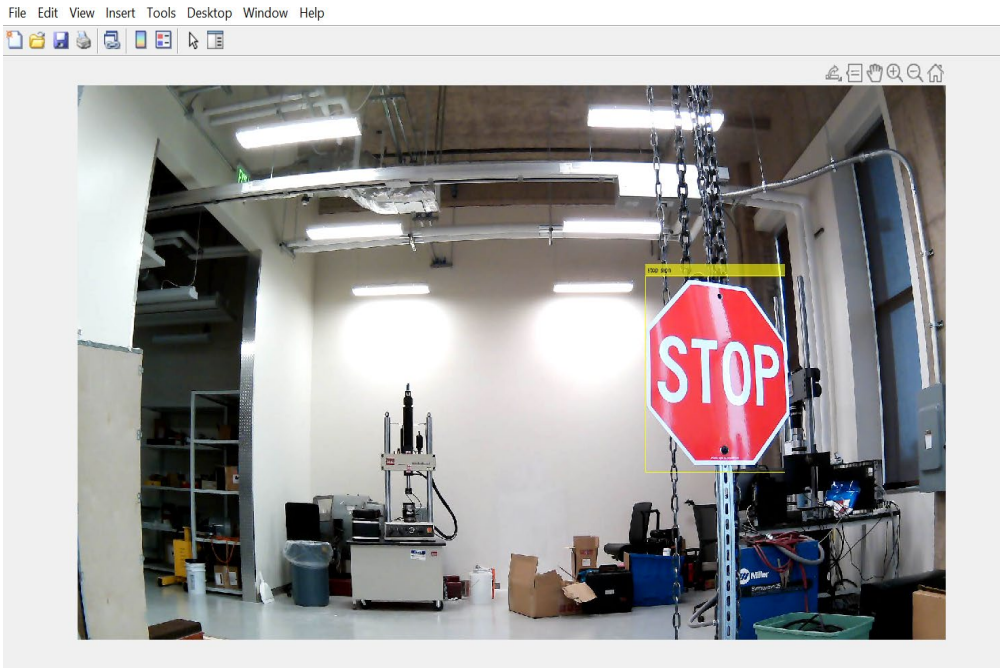

Figure 5.4. Stop sign for obstacle course put through YOLO analysis.

```
>> labels

labels =

  categorical

     stop sign

>> scores

scores =

  single

    0.9995

>> bboxes

bboxes =

  1×4 single row vector

   1.0e+03 *

    1.2570    0.3720    0.3090    0.3827
```

Figure 5.5. Statistics on the YOLO analysis shown in Fig. 5.4.

The labels in Fig. 5.5 represent the list of all the classes of the objects the computer recognizes with each image, in this case just the stop sign. The scores represent the percentage of certainty the computer is that the object belongs to the displayed class, which ranges from 0 to 1 with 0 being unsure and 1 being 100% certain. The computer is 99.95% sure that the object in Fig. 5.4 is a stop sign. The variable named boxes represent the size and location of the rectangle placed around each object recognized, each object recognized having four numbers that describes the rectangle encasing it. The first two numbers are the x,y coordinates respectively of the bottom left corner and the last two numbers are width and height of the box. In the function for the vehicle the program looks at the third column of the boxes variable when it corresponds to a stop sign and looks at the width and determines when to stop the vehicle based on the size of the width of the stop sign. When the vehicle gets closer to the stop sign, the width of the rectangle around increases and when it reaches the threshold, thus, the car is programmed to go into a stopping protocol, pauses for three seconds and then continue along its path.

Unfortunately, when trying to incorporate the YOLO subsection in the larger integrated program, the data processing was too intensive for the program not to have adverse effect from slower processing. The image data acquired from the camera was a very large matrix for each iteration, which means just acquiring the image at each iteration was taking up too much time, so the camera and YOLO code was commented out when running the program.

CHAPTER SIX

Programmable Steering And Data Analysis

Steering Rack Control

For MATLAB to determine which direction and how far to turn the servo, to turn the vehicle in order to drive on the correct side of a road, including road curvatures and turns, MATLAB reads in the distances from ultrasonic sensors 1 and 9. “Mapping” is not used in decision making for path planning of the vehicle, but instead the vehicle reacts to the surrounding environment completely in real time. This allows the vehicle to be a lot less rigid but a lot more dynamic and so this works very well in the nature of testing at the Baylor Research and Innovation Center (BRIC) parking area having limited space. This is because if mapping was used, for every path the vehicle could take, it would have to be already driven through manually to gather data to then be used for autopilot driving.

With mapping the sensor values for each GPS location, there is a very limited amount of different paths or obstacle courses that can be set up in testing how well the car does in various road setups. But reacting to an obstacle course in real-time means one can set up a virtually limitless number of different course setups to test the vehicle performance with limited space and the vehicle can do this on the spot without any manually driving prep time. On a macro scale, the “mapping” approach would limit autopilot available areas due to the fact that sensor data would have to be gathered on every location before the vehicle being able to drive autonomously at said location. This would mean with mapping, a data gathering vehicle would have to drive down every road

and highway manually first to operate which as the reader can imagine would be quite unrealistic especially with the fact that any change in a road or highway like construction would cripple the autopilot availability or require the need to remap the area. Conversely, with good sensors, operating system, and efficient algorithm, a vehicle can be on autopilot in a much greater range of driving areas and driving scenarios.

To get the vehicle to turn, a large gear is fastened on one high torque servo's shaft and the large gear turns a smaller gear attached to the steering rack's shaft as shown in Fig. 6.1.

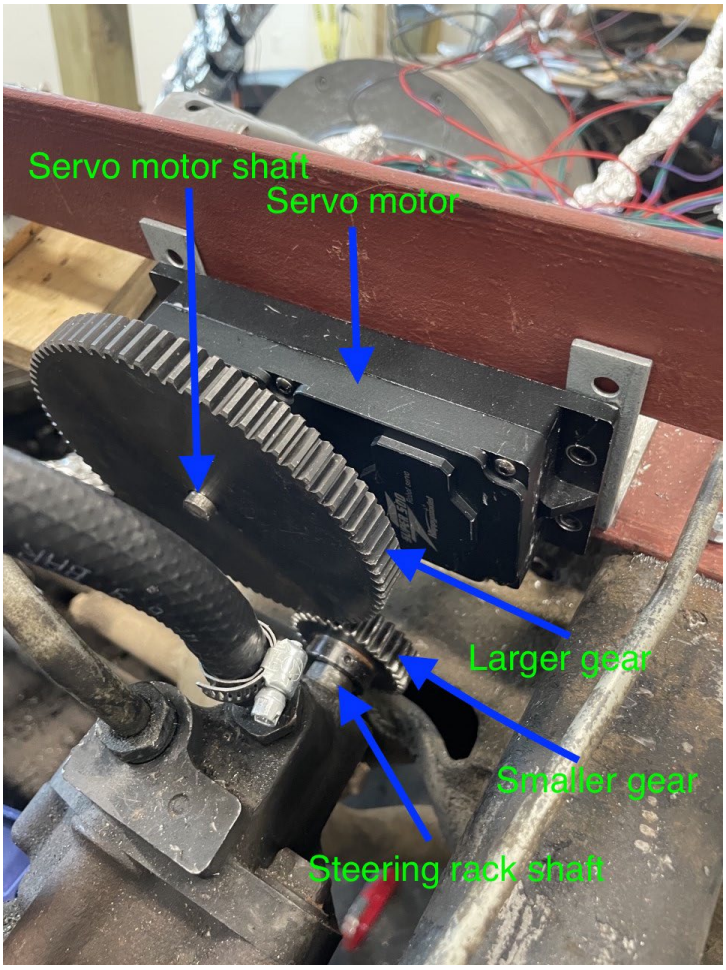

Figure 6.1. High torque servo motor and steering connection.

The steering fluid tank and steering pump was reconnected on the vehicle along with liquid tubes to keep the steering fluid in the vehicle system to keep the power steering enabled to lower the torque needed to turn the vehicle steering rack. To introduce and keep pressure in this steering fluid system, a high torque motor is connected to the pump's shaft shown in Fig. 6.2.

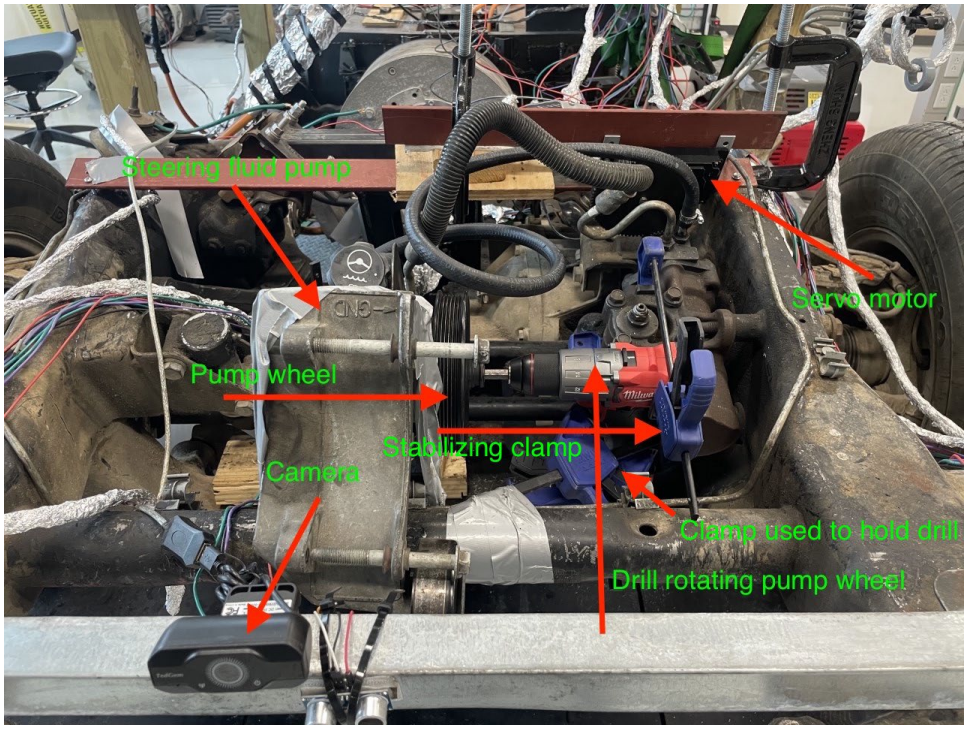

Figure 6.2. Steering fluid pump system.

The difference in using the pump was shown when just one lower torque servo motor, shown in Fig. 6.3 was used to turn the steering rack while the front wheels were jacked up and suspended off the ground. While connecting and using a drill gun to turn the pump, the steering rack turned noticeably faster and the servo motor drew 1/3rd of the current it did while not using the pump to pressurize the steering fluid system.

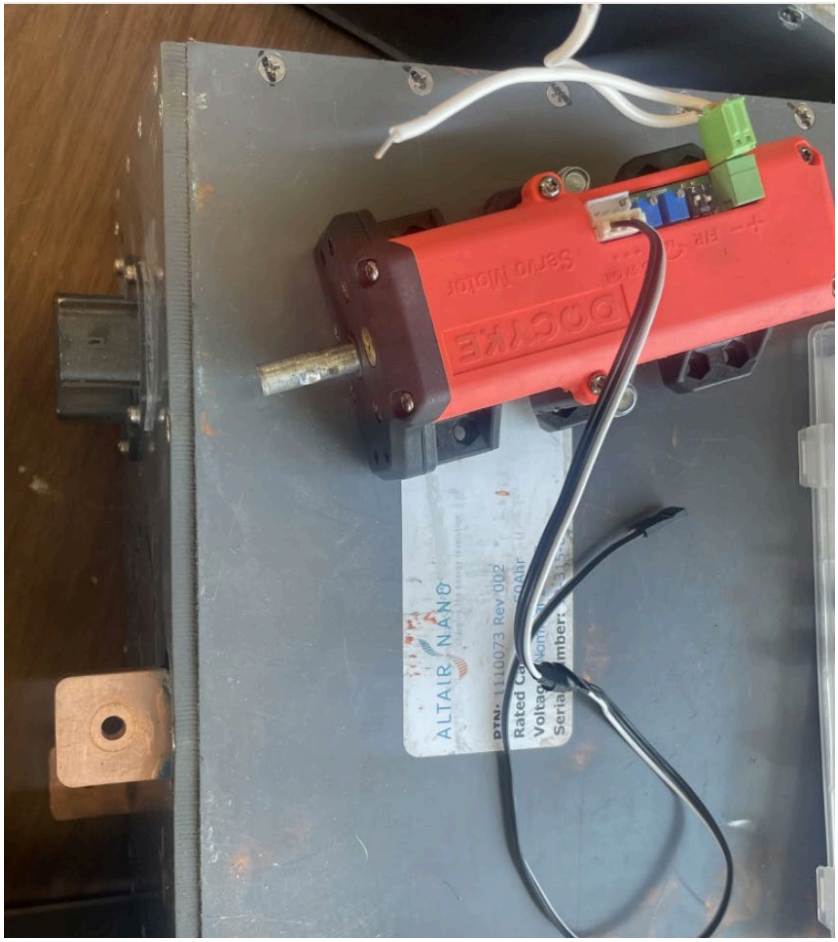

Figure 6.3. Lower torque servo motor.

One of the higher torque servos wasn't permanently coupled onto the steering rack's shaft. Due to the fact that the servo can only turn 360 degrees but the steering rack turns about 2 ½ times from end to end, coupling the servo directly to the shaft would not allow the servo to turn the wheels enough to get enough of the steering range and so a gear was placed on the servos' shaft and the steering rack, that was about twice the diameter of the gear placed on the steering rack to allow for the full steering range and this setup was replicated with the high torque servo.

During the setup of the high torque servo, the servo rotated the steering rack when coupled, but when the server was uncoupled and the wheels were manually turned to find

out how many rotations the steering rack has from end to end, some steering fluid shot out of the steering pump fluid inlet. Due to this, after replacing the gear on the servo to turn the steering rack, the servo burned out because the torque required was much greater than before because the system was no longer pressurized and circulation of the fluid to aid the vehicle in steering was absent. This experiment was indicative of how important the pressurized steering fluid system was in aiding the steering of the vehicle especially at low speeds.

The programmable steering system works similarly to the autonomous acceleration system in terms of control, for instance, they both work on PID control where the MATLAB uses Arduino to send a PWM signal to the servo motor that controls the direction and angle of the rotation which in turn, turns the steering rack and vehicle from left to right. The error of the difference between the intended distance from the obstacles, gathered from the ultrasonic sensors, is fed into the PID control to hold a position, turn left or right, and how much to turn in real time. Initially, the autonomous steering system was programed to communicate with the VCU for a timestamp that was used for time-integration for the PID integral term. However, when testing and PID tuning the steering system outside using the obstacle course, the vehicle was not reacting as desired or expected, which seemed like unideal P,I, and D constants were used or some error in the algorithm. It also appeared as though the program was processing too slow to make the proper decisions on time, greatly affecting the steering's performance. In an attempt to increase the processing speed, the steering program was rewritten to do the same task but without the dependence of other USB serial communication devices connected to MATLAB while running, which included an Arduino that controls the

speed manually using a joystick that maps to the digital potentiometer analog voltage range, another Arduino that has the ultrasonic connected to it, and a VCU.

The first Arduino and joystick is used to control speed manually to isolate the task of tuning the steering PID controller. However, with the joystick setup, the voltage decipher was connected to the throttle pedal as it has a more secure connection to the inverter and is much less affected by electromagnetic interference (EMI) from the inverter, which keeps the throttle enabled when driving the vehicle without malfunction. This EMI issue was witnessed before ever leaving the laboratory when testing if the joystick could properly control the speed and steering manually when transporting the vehicle to and from the lab and the testing parking lot. Often with the joystick, while steering and trying to add speed, the throttle signal from the digital potentiometer to the VCU gets interrupted and disengages the throttle. For more consistent and safer operation, the code still communicates with its devices but the VCU voltage decipher is then connected to the throttle. Removing these serial connections speeds up the processing about three times over, however, the vehicle still wouldn't turn while moving but only turns while the vehicle is stopped. After further investigation, the ultrasonic sensors behave normally when not driving, but as soon as the inverter requests torque from the motor, the EMI from the 3 switching wire outputs from the inverter would result in poor performance from the sensors. This interference had the values of the ultrasonic sensors each go to about zero, making the error of the steering appear to be zero as shown in Fig. 6.4. Fig. 6.4 and Fig. 6.5 do not show the red marker because it is superimposed by the blue marker because in the scenarios, the target left and right sensors equal each other.

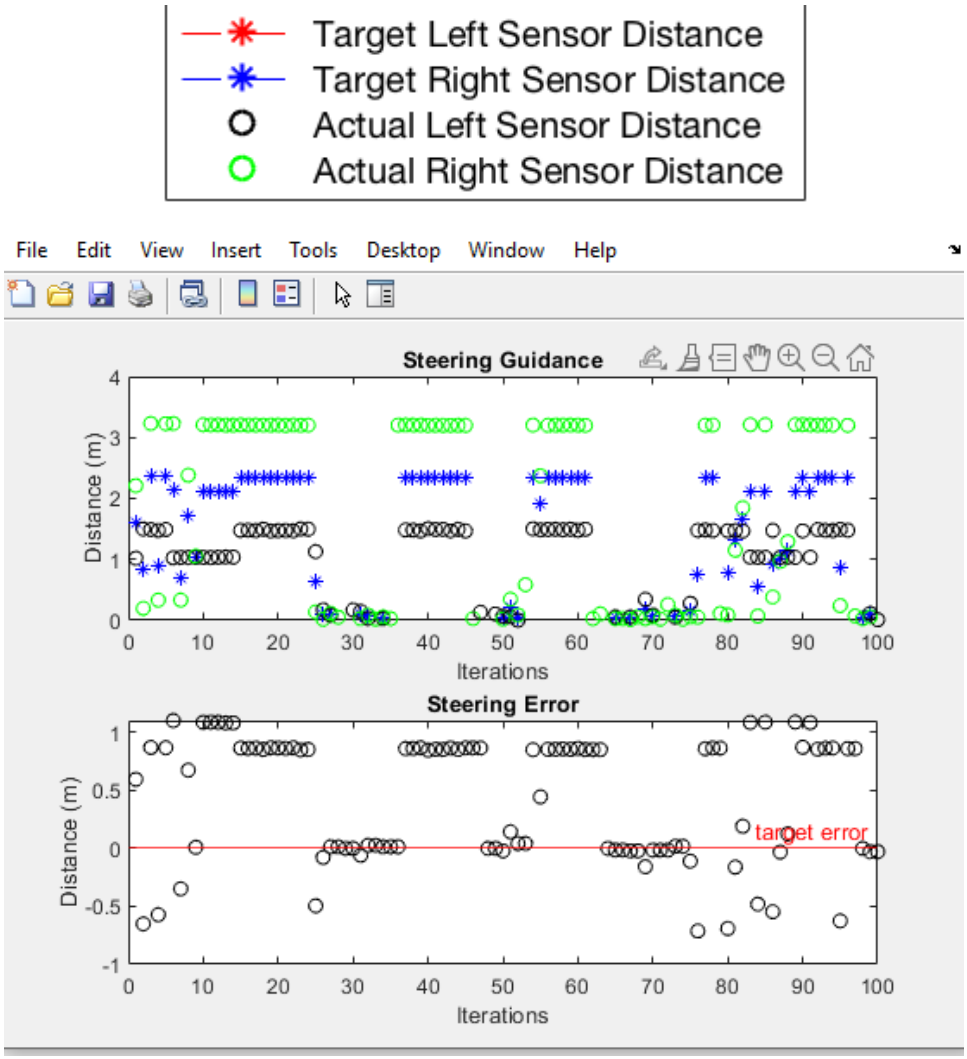

Figure 6.4. Dyno ultrasonic sensor EMI.

In Fig. 6.4 and Fig. 6.5, the black circles represent the left front sensor (sensor 1) where the green circles represent the right front sensor (sensor 9) and the blue asterisk represents the target distance of each sensor. The breaks in the data where the blue and green circle suddenly decrease and approaches zero are the points when the inverter gives the motor a torque demand. The interference of the sensors was the main culprit of the steering program. To mitigate the issue of EMI interference, Foil was used on the sensor wires and the inverter switching cables to shield

the sensors against the EMI. This helps the sensor data significantly read data more consistently and accurately. Also, the computer and Arduino and connected devices were moved away from the inverter and this also helped. Both acceleration and steering systems are both PD controllers and so when running the steering program with manual control of the speed, the final values for the constants of the steering PD controller are P=1.5, and D=1.5. The idea was to avoid oscillation in switching back and forth over the target error line in the error curve in Fig. 6.5. Due to the latency of the servo when changing direction, the program was intended to undershoot to avoid overshooting and more error. The figure below is the autonomous steering performance of the vehicle as it ran through an obstacle course with a path that starts straight, curves to the right and curves to the left as shown in Fig. 6.6.

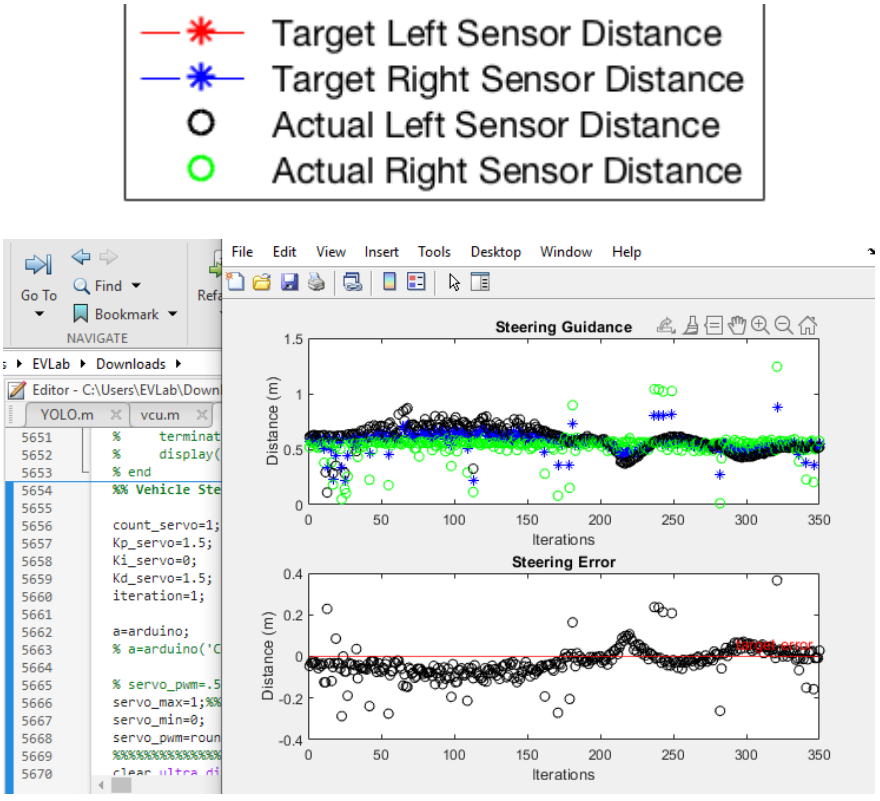

Figure 6.5. Autonomous steering guidance.

Figure 6.6. Obstacle course.

The distance readings on the ultrasonic sensors which have seemingly random distance points are noisy inputs from the smaller amount of EMI still observed from the inverter. In Fig. 6.5, the control algorithm undershoots the target distance intentionally to slightly increase the margin of safety in this research project. It was also noticed that occasionally severe EMI occurred, corrupting the ultrasonic sensor readings. The dip at about 220 iterations in the Fig. 6.5 above occurs because of the much slower speed that was

commanded manually while maneuvering through the obstacle course. The idea was to have the vehicle center itself on the right side of the path to simulate how it behaves on the road, however the performance of the sensors drop significantly outdoors and while the vehicle is in motion so the obstacles were moved much closer together for much better results. So instead of the vehicle centering itself on the right side of the road, it centers itself in the middle because there is not much space between the obstacles to operate so closely to one side. In addition, with EMI, the sensor performance increases greatly with obstacles set closer to the ultrasonic sensors.

Data Gathering

In addition to the sensor data being used for real-time calculations and decisions, other data can be logged and mined for further analysis. For instance, the current code records all of the sensor data for every GPS location, with the measurements stored in a .MAT file. If desired, this data can be readily used to create a detailed map of the surroundings to then be used in decision making or planning in the future. One application could be to process the information through an AI, like the CNN described above, to both evaluate the performance of the driving, and continuously improve the path planning such as adjusting the PID values of the vehicle speed and steering systems to reduce unwanted error (wanted error includes the small error in the vehicle speed control when a new target speed is set and the vehicle needs to change speed, the vehicle does not need to accelerate to the new speed as fast as possible but at a smooth and yet responsive pace). A driver traversing in a 20 MPH speed zone would not normally floor it when reaching the 55MPH zone but accelerate at a reasonable pace, this will mean more error because the target speed and vehicle speed will differ for a longer time when

comfortably accelerating but this error is desired. The same is true for steering. When changing lanes, drivers usually do not jerk their car as much as possible, but instead smoothly move into a neighboring lane which causes more error but again, this error is desired.

CHAPTER SEVEN

Conclusions and Future Work

Conclusions

In this work, autonomous electric vehicle capabilities were developed enabling an EPA drive cycle to be followed on a dynamometer testbed by a modified Chevy Bolt and navigation through an on-road obstacle course by a converted electric Chevy Tahoe. The acceleration system on the Chevy Bolt relies on the connection to the vehicle's OBD communication and the timing of this communication with several safety factors built in and inherent in its design. The OBD outputs speed, time, and APP to give the LabView the necessary values to operate and additional variables like APP for feedback to compare with LabView; the APP is for further debugging capabilities which was helpful and useful as discussed in Chapter 3. After developing both the Chevy Bolt and Tahoe's autonomous acceleration systems, the inherent safety features of the Bolt's was a great design with having physical mechanical parts to control the throttle such as the servo and solenoid that disengages if something goes wrong with the program including cords being disconnected. Having mechanical parts in the program makes for complexity but makes for solid disconnect, in addition to the digital stop button and big E-stop. This is very useful because many unexpected problems can arise.

The acceleration system implemented for the Tahoe also works on many shutdown contingencies, by monitoring current and voltage levels being read in to MATLAB by the VCU. The program can be shutdown if throttle is too low by the

inverter firmware or having a wire or chord disconnected. The brake by wire works in the same way as the throttle by sending a voltage signal from a digital potentiometer which keeps things pretty seamless; the two work together, whereas with the Bolt, the regenerative braking was constant and in order to control the brakes better, the physical brakes were used by using mechanical interaction with the linear actuator.

Setting up the steering system for the Tahoe was a challenging accomplishment because mounting the servo to be able to make contact with the aligning gear with the steering rack shaft, and to do so consistently, considering the servo was skipping teeth due to unstable alignment. The angle of the steering rack shaft was a simple but difficult problem to solve long term. Even after monitoring correctly, just getting the gear on the servo shaft was difficult to keep tightened while operating. The shaft had to be drilled into so that the set screws from the gear can grip the shaft better which helped tremendously.

With processing speed and EMI being the two most limiting factors in this project, the decision was made to separate autonomous acceleration and steering systems, which noticeably improved in both of these categories. It is like night and day after splitting the work between two different laptops and Arduinos with the autonomous steering system. The EMI is very minimal in comparison as it was before the EMI shielding precautions were taken. The concern is with the autonomous acceleration system; EMI significantly affects the performance and consistency of this system. Even just running the acceleration system in isolation on the dyno testbed as in Chapter 4 of this paper, the results may vary and the operation is very rough as if the inverter is turning off and on. The EMI affects the signals controlling the digital potentiometer

resulting in some unpredictable and unexpected behavior even after foil and tape EMI shielding several cords and cables and moving the computers and Arduinos further away from the inverter and placing the Arduinos and digital potentiometer inside a metal box shown in Fig. 7.1

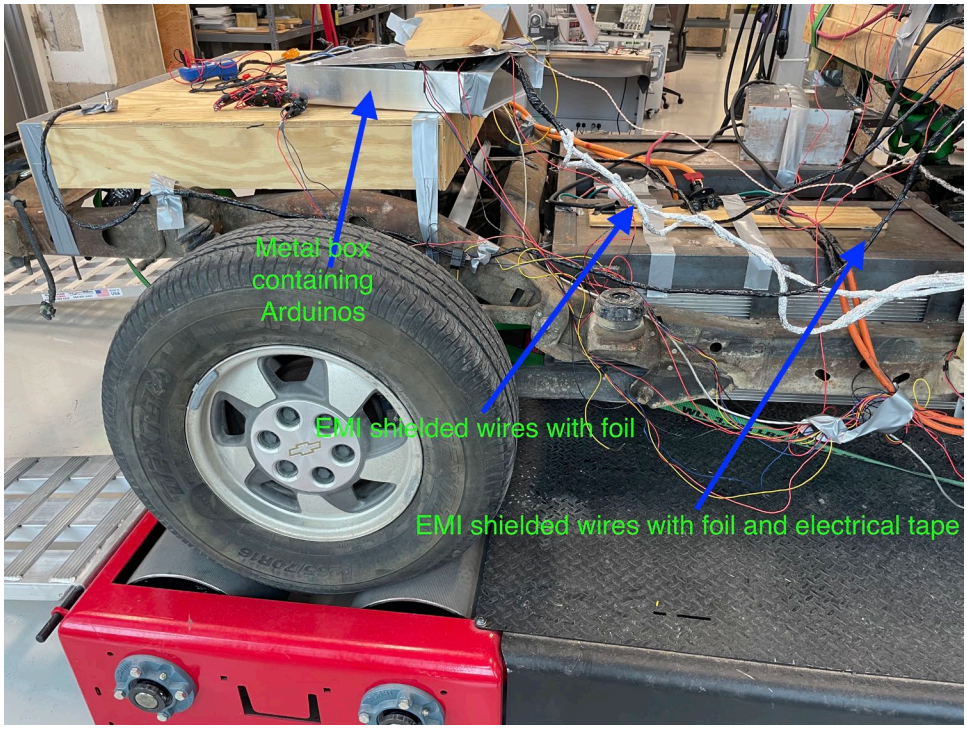

Figure 7.1. EMI shielding

In an attempt to keep the programs running quickly to operate properly on both computers, the GPS and YOLO systems were commented out when running the large integrated programs, also preventing more EMI in the system than already present.

Future Work

A potential future use for this autonomous vehicle is to become an autonomous shuttle between the BRIC and Baylor's campus. The sensors need to be upgraded to

make this possible. The ultrasonic sensors used should be replaced with radar/lidar and long-range radar shown in Fig. 7.2.

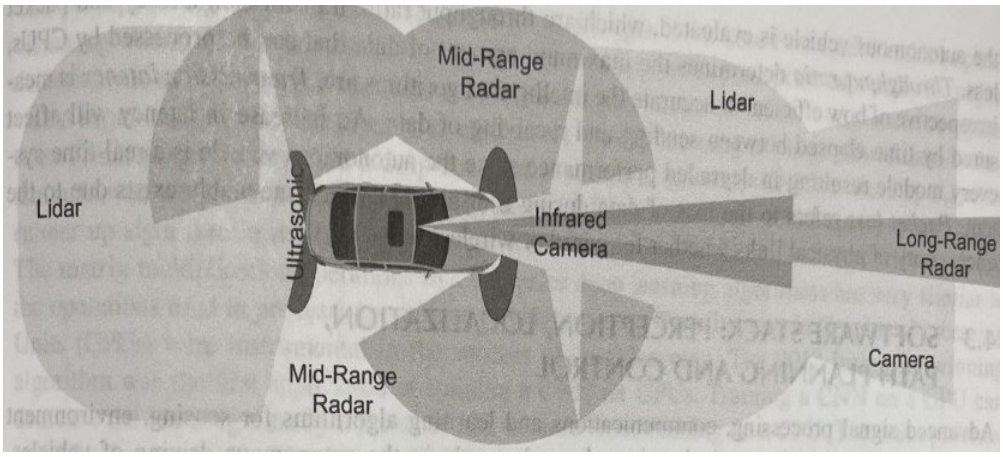

Figure 7.2. Different sensors used to detect objects around it [36].

This way the vehicle can get a full 360-degree field of vision that can allow for a higher resolution decision algorithm. Lane detection will need to be an added feature. The YOLO neural network can be added upon, in addition to detecting and classifying objects, by training it to recognize lane and road lines and make use of this feature in the vehicle's path planning. Of course, the vehicle needs headlights and taillights with turn signals and hazard light programmed with them. A device like a WIFI extender that can enable the vehicle to access the internet should be put into the vehicle so any updates to the code can be updated between trips and if there is a fleet of these vehicles at some point, they all can be monitored and updated remotely. When connected to the internet, the status of speed, voltage, current, direction, and more can be monitored in real time remotely while the vehicle is connected to the internet using the MATLAB drive connector application that automatically updates the files in the MATLAB drive folder, when connected to the internet and to every computer connected on that MathWorks

account. The values of the variable desired to be monitored can be saved as a .MAT file in the MATLAB drive folder and constantly read in values can be sent to another computer remotely that can even take some of the processing load off the main onboard computer. With being able to remotely share data, another computer can take data and do processes like graphing them for the riders to view during a trip while the main onboard computer does the essential real-time and time-sensitive data analysis during a driving trip. The original Chevy Tahoe body can be put back on, including the seats and seatbelts, or a new body can be constructed with conference style seating similar to Fig. 7.3.

Figure 7.3. Future conference seating arrangement [37].

Airbags need to be installed; imagine seating around some table like the figure above with a circular dashboard with airbags all around. Another string of batteries should be

added to double the energy capacity of the car, doubling its mile range. In having another string of batteries there will need to be a parallel BMS that will keep the power drawn from each string, an example of this is shown in Fig. 7.4 and Fig. 7.5.

Figure 7.4. Parallel component to make BMS able to monitor battery strings [38].

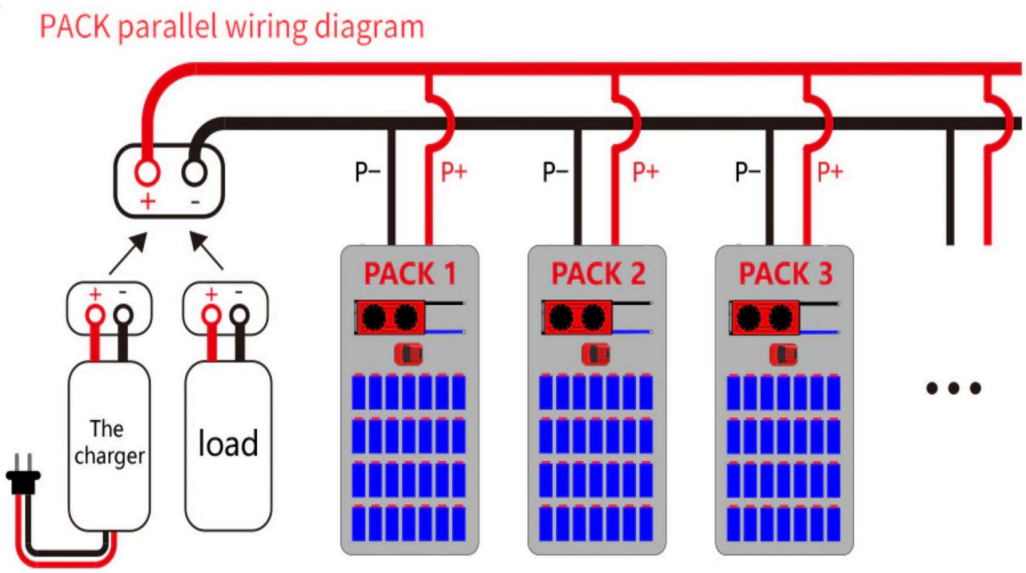

Figure 7.5. Parallel battery management systems' schematic [38].

Also, with the abundance of the lithium titanate-oxide (LTO) batteries from Proterra, battery swapping is very viable, but a more readily swappable case will have to be developed along with an autonomous battery case swapper. Alternatively, with the available power in the laboratory, fast charging is also a good option and can be used while the battery swapping technology is being developed. Potentially, the BMS can be upgraded to be capable of handling more amps and in turn being able to achieve higher vehicle speeds, this can also be achieved by charging each of the battery modules above their nominal 24V with the maximum voltage being 29V for each module. Finally, a BMS should be purchased and connected to the LTO battery modules used to power the servo motors and VCU.

REFERENCES

[1] Emission Test Cycles, “EPA Highway Fuel Economy Test Cycle (HWFET),” available: https://dieselnet.com/standards/cycles/hwfet.php#:~:text=The%20Highway%20Fuel%20Economy%20Test,on%20the%20FTP%2D75%20test.

[2] Lizzy Rosenberg, “What Is Regenerative Braking? Here’s Why It’s So Important to EV Drivers,” 2021, Available: https://www.greenmatters.com/p/regenerative-braking

[3] The brake report, “DEMONSTRATING CHEVY BOLT REGEN BRAKING,” 2021. Available: https://thebrakereport.com/demonstrating-chevy-bolt-regen-braking/

[4] Christopher Lampton, “How Regenerative Braking Works,” 2021. Available: https://auto.howstuffworks.com/auto-parts/brakes/brake-types/regenerative-braking.htm

[5] Caroselli, Beachler, Coleman, “What Percentage of Car Accidents Are Caused by Human Error?,” 2021. Available: https://www.cbmclaw.com/what-percentage-of-car-accidents-are-caused-by-human-error/#:~:text=A%202016%20study%20by%20the,96%25%20of%20all%20auto%20accidents.

[6] David Atkinson, “Autonomous Vehicles,” 2021. Tu simple: pp.

[7] American geosciences institutes, “What is Lidar and what is it used for?,” Available: https://www.americangeosciences.org/critical-issues/faq/what-lidar-and-what-it-used#:~:text=%22LIDAR%2C%20which%20stands%20for%20Light,variable%20distances)%20to%20the%20Earth.

[8] Merrill I. Skolnik, “radar,” available: https://www.britannica.com/technology/radar/Transmitters

[9] The News Wheel, “How Do Radar Detectors Work?,” 2019. Available: https://thenewswheel.com/how-do-radar-detectors-work/

[10] Shannon Mattern, “Mapping’s Intelligent Agents,” 2017. Available: https://placesjournal.org/article/mappings-intelligent-agents/?gclid=EAIaIQobChMIpLyRs6CX9wIVS21vBB0BsQEJEAAYASAAEgKqxfD_BwE

[11] Robson Forensic, “The Functional Components of Autonomous Vehicles,” 2020. Available: https://www.robsonforensic.com/articles/autonomous-vehicles-sensors-expert

[12] Accolade Technology, “5 Levels of Autonomy (L1 and L2),” available: https://accoladetechnology.com/5-levels-of-autonomy-l1-and-l2/

[13] Jeremy Jordan, “Introduction to autoencoders.,” 2018. Available: https://www.jeremyjordan.me/autoencoders/

[14] Charter Global, “USING AUTOENCODERS FOR IMAGE CLASSIFICATION,” Available: https://www.charterglobal.com/using-autoencoders-in-ai-for-image-classification/

[15] Sumit Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,” 2018. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[16] Oliver Knocklein, “Classification Using Neural Networks,” 2019. Available: https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f

[17] IEEE Xplore, “Trajectory Prediction of Vehicles Based on Deep Learning,” Available: https://ieeexplore.ieee.org/document/8880168

[18] ADAS & Autonomous Vehicle INTERNATIONAL, “The road to everywhere: are HD maps for autonomous driving sustainable?,” Available: https://www.autonomousvehicleinternational.com/features/the-road-to-everywhere-are-hd-maps-for-autonomous-driving-sustainable.html

[19] MDPI, “Visualization of Urban Mobility Data from Intelligent Transportation Systems,” 2019. Available: https://www.mdpi.com/1424-8220/19/2/332

[20] Battle Born Batteries, “What Is A BMS (Battery Management System)?,” 2021. Available: https://battlebornbatteries.com/battery-management-system/#:~:text=The%20primary%20function%20of%20the,loose%20connections%20and%20internal%20shorts.

[21] Synopsys, “What is a Battery Management System?,” available: https://www.synopsys.com/glossary/what-is-a-battery-management-system.html

[22] David Wenzhong Gao, “Interfacing Between an ESS and a Microgrid,” 2015. Available: https://www.sciencedirect.com/topics/engineering/battery-management-system

[23] NXP, “Active Cell Balancing in Battery Packs,” 2012. Available: https://www.nxp.com/docs/en/application-note/AN4428.pdf

[24] Analog Devices, “ Passive Battery Cell Balancing,” available: https://www.analog.com/en/technical-articles/passive-battery-cell-balancing.html

[25] ECU Testing, “ECU EXPLAINED,” Available: https://www.ecutesting.com/categories/ecu-explained/

[26] Mobility Insider, “What Is an Electrical Control Unit?,” 2020. Available: https://www.aptiv.com/en/insights/article/what-is-an-electronic-control-unit

[27] Dr. Rahul Ahlawat, Dr. Jürgen Bredenbeck, and Mr. Tatsuo Ichige, “Estimation of Road Load Parameters via On-road Vehicle Testing,” 2013. Available: https://www.aandd.jp/support/dsp_papers/estimation.pdf

[28] Semantic Scholar, “PID controller design for cruise control system using genetic algorithm,” 2016. Available: https://www.semanticscholar.org/paper/PID-controller-design-for-cruise-control-system-Rout-Sain/7bc1fd91e1c71d7f427fe1fbbf86d992037f1e2a

[29] Know Your Parts, “From Pedal to Pads: Brake Systems Explained,” available: https://www.knowyourparts.com/technical-resources/brakes-and-brake-components/from-pedal-to-pads-brake-systems-explained/

[30] Thunderstuck Motors, “Thunderstruck Motors Vehicle Control Unit v3.1,” available: http://www.thunderstruck-ev.com/images/companies/1/DD_VCUv3.1R3.pdf?1605898763828

[31] UQM Technologies, “PowerPhase 100 Traction System,” available: http://www.thunderstruck-ev.com/images/companies/1/UQMPP100DataSheet.pdf?1550791147819

[32] Open Data Science, “Overview of the YOLO Object Detection Algorithm,” 2018. Available: https://odsc.medium.com/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0#:~:text=YOLO%20is%20a%20clever%20convolutional,and%20probabilities%20for%20each%20region.

[33] Manish Chablani, “YOLO — You only look once, real time object detection explained,” 2017. Available: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

[34] Hmrishav Bandyopadhyay, “YOLO: Real-Time Object Detection Explained,” 2022. Available: https://www.v7labs.com/blog/yolo-object-detection

[35] Sreedhar Achari, “Practical Implementation of Object Detection On Video with OpenCV and Yolo v3 pre-trained weights on coco data,” 2020. Available: https://medium.com/@vsreedharachari/practical-implementation-of-object-detection-on-video-with-opencv-and-yolo-v3-pre-trained-weights-a2d2995aac41

[36] Iqbal Husain, “Electric and Hybrid Vehicles,” third edition.

[37] CanStockPhoto, “Business Meeting Seats' Layout In Autonomous Car,” available: https://www.canstockphoto.com/business-meeting-seats-layout-in-43320146.html

[38] Aliexpress, available: https://www.aliexpress.us/item/3256803539838087.html?spm=a2g0o.detail.0.0.4ebe27acBf1wvK&gps-id=pcDetailBottomMoreThisSeller&scm=1007.13339.274681.0&scm_id=1007.13339.274681.0&scm-url=1007.13339.274681.0&pvid=3b14a23e-7701-4e49-8bf6-3ee35f4c1f6f&_t=gps-id%3ApcDetailBottomMoreThisSeller%2Cscm-url%3A1007.13339.274681.0%2Cpvid%3A3b14a23e-7701-4e49-8bf6-3ee35f4c1f6f%2Ctpp_buckets%3A668%232846%238110%231995&pdp_ext_f=%7B%22sku_id%22%3A%2212000026947445393%22%2C%22sceneId%22%3A%223339%22%7D&pdp_npi=2%40dis%21USD%21%2131.7%21%21%21%21%21%402101d1b116579146649473889e171b%2112000026947445393%21rec&gatewayAdapt=glo2usa&_randl_shipto=US