5-7-2021

Smart E-Bike Conversion Kit and Helmet

Megan West
mwest14@lion.lmu.edu

Marena Trujillo
mtrujil8@lion.lmu.edu

Hossein Asghari
masghari@lmu.edu

Loyola Marymount University
University Honors Program

Smart E-Bike Conversion Kit and Helmet

Marena Trujillo and Megan West

A thesis submitted in partial satisfaction

of the requirements of the University Honors Program

of Loyola Marymount University

by

Megan West

05/03/2021

Smart E-Bike Conversion Kit and Helmet

Senior Project 2020-2021

Team Members
Marena Trujillo
Megan West

Project Advisor
Hossein Asghari, Ph.D

Department of Electrical and Computer Engineering
Seaver College of Science and Engineering
Loyola Marymount University
Los Angeles, CA

Abstract

With many state governments across the United States implementing shutdowns to prevent the spread of COVID-19, many bike retailers have seen dramatic increases in bike sales. Electric bicycles (e-bikes) have become increasingly popular as the public searches for alternatives to public transportation. Unfortunately, e-bike users assume some risk by riding bikes that are faster than conventional bicycles. While kits to convert conventional bicycles to e-bikes exist, no "smart" e-bike/helmet kit specifically designed to keep the user safe is currently available on the market. To mitigate risks, a smart e-bike conversion kit with multiple novel safety features, including auditory and visual signals to alert the rider of obstacles, will be presented in this paper. Additionally, the conversion kit will enable the rider to employ pedal assist by measuring the rider's cadence and providing power to the motor accordingly. Pedal assist will be provided at a gradual rate, ensuring rider safety and a smooth speed boost. The proposed product provides the customer with an affordable, safe, and sustainable alternative to a traditional bike or e-bike experience.

Contents

1 Introduction

During the course of the COVID-19 pandemic, many bike retailers have seen dramatic increases in bike sales [8]. Electric bicycles (e-bikes), have become increasingly popular as the public searches for new alternatives to public transportation in order to commute. In addition to helping people avoid situations where they might be exposed to COVID-19, e-bikes also incentivise people to exercise more often. Studies have shown that, on average, people who ride e-bikes spend more time exercising on their bicycle than those who ride conventional bicycles [9]. Additionally, e-bikes provide a more environmentally sustainable form of transportation for distances too far to walk on foot.

A state of the art e-bike costs around $1500, but less expensive alternatives exist, namely "e-bike conversion kits." A conversion kit can convert a standard bicycle into an electric bicycle by affixing a motor and battery onto the bike. Unfortunately, all e-bikes users assume some risk by riding such a bike. E-bikes are faster than conventional bicycles, and as a result e-bike users are at risk of seriously injuring themselves in the event of a collision.

2 Project Objectives

2.1 Background Information

There is a need for an affordable e-bike conversion kit that enables people to commute safely and efficiently. This kit must be environmentally responsible and affordable for college students, many of whom already own bicycles. The conversion kit should be particularly appealing for those commuting in urban areas, as the safety features are designed to keep cyclists safe from colliding with cars. Finally, people without advanced knowledge of bicycle mechanics should be able to install the kit with relative ease.

Even though e-bikes can be more dangerous than regular bicycles, the design of this e-bike conversion kit helps to improve upon the safety of e-bikes. With the inclusion of safety features in the form of sensors, lights, and sound, cyclists will be more aware of hazards and obstacles. The product will be a safe alternative to regular e-bikes for those who still want the functionality of an e-bike. Additionally, people exercise more on an e-bike than a traditional bicycle [9]. The design makes it easier for people to ride their bicycles in the short-term and users will be encouraged to ride their e-bikes more often; thus, this design helps to promote a healthy lifestyle. This design addresses safety concerns of current e-bikes while also providing a more sustainable and affordable mode of transportation.

2.2 Customer Requirements

The customer requirements were determined by keeping in mind the motivations of the project and its target audience. They are focused around safety, sustainability, and affordability. The customer requirements for the project are as follow:

1. The system should be safe and promote additional safety for its users.
2. The system should be environmentally sustainable.
3. The system should have low cost, as compared to similar products.
4. The system should be easy-to-install for users without an advanced knowledge of bicycle mechanics.
5. The system should allow for faster commute times than travel on a normal bicycle.

As e-bikes and e-bike conversion kits are still relatively new to the market, the product still has to be designed to appeal as an e-bike: a faster and easier way to travel than on a traditional bicycle. On top of that, the requirements reflect the emphasis on safety, sustainability, and affordability. Additionally, the conversion kit has to be usable and installable for all users to be an effective product, because otherwise the ease of the conversion kit is lost. The conversion kit should be an easier and cheaper alternative to purchasing an entire e-bike. The customer requirements do not include specifications for operating temperatures or extreme inclement weather conditions.

3 Proposed Solution

3.1 Trades Leading to Proposed Solution

Many factors were considered when deciding on components to purchase and approaches to pursue in terms of the design of the project. Concept fans, pairwise comparisons, and decision matrices helped to guide those decisions and methods. One major conceptual decision was related to the obstacle detection system. A key feature of the "smart" system is obstacle detection, which will be implemented using HC-SR04 Ultrasonic Sensors. The HC-SR04 module was chosen over several other alternatives, which are shown in Fig. 1.

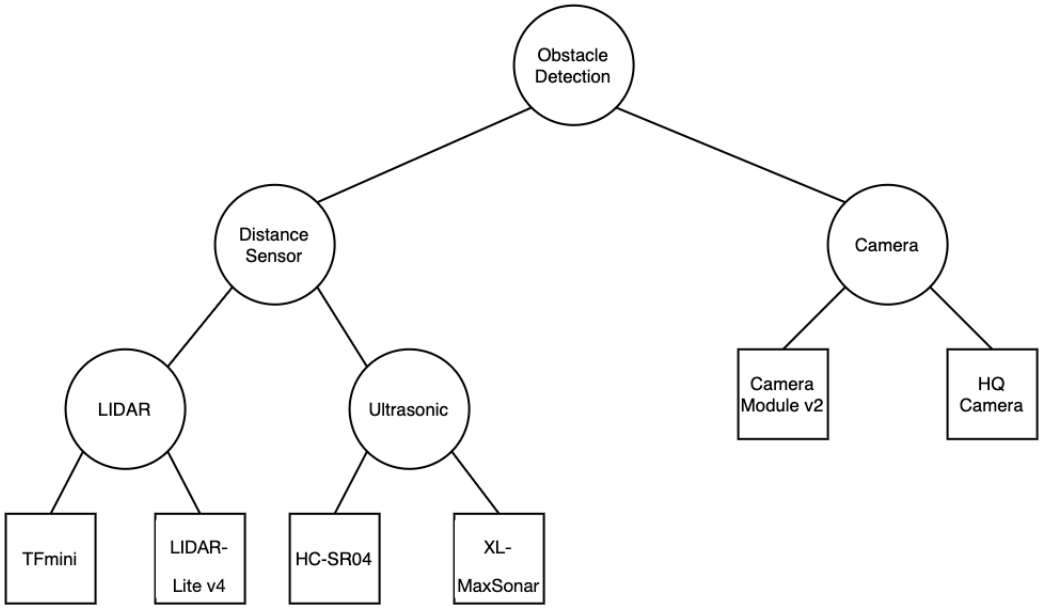


Figure 1: Concept Fan for the Obstacle Detection System

Distance sensors were chosen over a camera because to detect obstacles with a camera, a machine learning algorithm would need to be implemented. While this application of machine learning would likely result in a sophisticated obstacle detection system, it would also increase the overall cost of the conversion kit. Additionally, developing this system would be a timing-consuming effort and nearly worthy of a distinct capstone project. For these reasons, the use of cameras was ruled out for the obstacle detection.

Although obstacle detection with cameras or Light Detection and Ranging (LiDAR) would be more accurate, it would be far more difficult and expensive to implement. Additionally, for the scale of obstacle detection that would be suitable for an e-bike, the ultrasonic distance sensors will be sufficient, and the HC-SR04 modules are inexpensive and easy to use.

A microcontroller will be used to interpret the data obtained from the HC-SR04 sensor. A Raspberry Pi 4B was chosen for its reasonable price, robust computing power, and Bluetooth

capability. The decision process is further outlined in Fig. 2, Table 2, Table 3, and Table 4. Fig. 2 shows a concept fan to outline the different platforms and models of potential microcontrollers. Table 2 puts this information into a decision matrix to choose a platform. A Raspberry Pi computer was chosen over an Arduino and Jetson microcontrollers. One key feature in the design of the sensor system was the ability for two microcontrollers or computers to be able to communicate wirelessly. If this is to be done using an Arduino, additional hardware that is complicated to install would be needed. While wireless communication is easier to implement using a Jetson microcontroller as compared to Arduino, the cost of a Jetson microcontroller made it an unrealistic option for this project.

After Raspberry Pi was chosen, the decision matrix shown in Table 4 was used to determine the best model. While the Raspberry Pi Zero WH was the most affordable option, it lacked the computation power needed to control the sensor system. The Raspberry Pi 4B 8GB was ruled out primarily because of its high cost. Raspberry Pi models 4B and 3B+ are similar, but the 4B is compatible with the newest version of Bluetooth, while the 3B+ is not. Since the cost of the 4B was equal to that of the 3B+, the Raspberry Pi 4B was best option.

The weights for the decision matrices were calculated using pairwise comparison. The criterion were compared to each other and weighted with appropriate strengths. The geometric mean for each category is then found and normalized to determine the weights. Tables 1 and 3 show these pairwise comparisons.

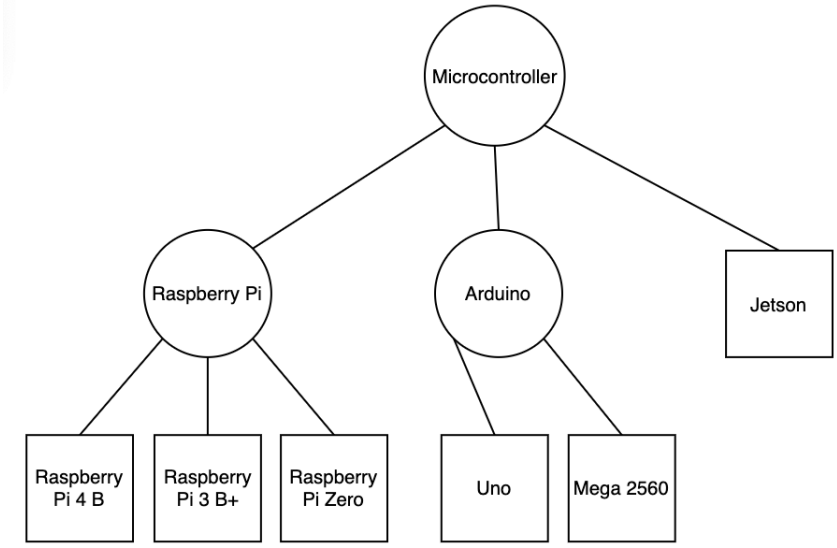


Figure 2: Concept Fan for Microcontroller

Table 1: Microcontroller Platform Pairwise Comparison

	Compitational Power	Wireless Communication	Battery Power	Speed	Storage	Cost	Cost of Additional Hardware	Weights
Computational Power	1	0.125	0.333	1	3	3	2	**0.12**
Wireless Communicaiton	8	1	1	2	3	3	3	**0.3**
Battery Power	3	1	1	1	2	2	3	**0.2**
Speed	1	0.5	1	1	2	1	5	**0.15**
Storage	0.333	0.333	0.5	0.5	1	0.333	0.25	**0.05**
Cost	0.333	0.333	0.5	1	3	1	1	**0.1**
Cost of Additional Hardware	0.5	0.333	0.333	0.2	4	1	1	**0.08**

Table 2: Decision Matrix for Microcontroller Type

	Weight	Arduino	RaspberryPi	Jetson
Computational Power	0.12	0.25	0.35	0.4
Wireless Communicaiton	0.3	0.18	0.6	0.22
Battery Power	0.2	0.4	0.28	0.32
Speed	0.15	0.29	0.34	0.37
Storage	0.05	0.37	0.33	0.3
Cost	0.1	0.4	0.45	0.15
Cost of Additional Hardware	0.08	0.4	0.33	0.27
Score		0.298	0.417	0.285

Table 3: Raspberry Pi Pairwise Comparison

	Price	Number of Cores	Bluetooth	Power Ratings	Wi-FI	Weight	Weight
Price	1	0.5	1	2	6	4	0.25
Number of Cores	2	1	2	3	4	2	0.3
Bluetooth	1	0.5	1	2	3	3	0.2
Power Ratings	0.5	0.333	0.5	1	2	1	0.1
Wi-Fi	0.167	0.25	0.333	0.5	1	0.5	0.05
Weight	0.25	0.5	0.333	1	2	1	0.1

Table 4: Decision Matrix for Raspberry Pi Model

		4B 8GB	4 B	3B+	Zero WH
Price	0.25	0.10	0.25	0.25	0.40
Number of Cores	0.30	0.30	0.30	0.30	0.20
Bluetooth	0.20	0.35	0.35	0.20	0.10
Power Rating	0.10	0.23	0.23	0.23	0.31
Wi-Fi	0.05	0.25	0.25	0.25	0.25
Weight	0.10	0.23	0.23	0.24	0.30
Score		0.244	0.281	0.252	0.224

A battery type was chosen based on the criteria specified in Table 6. The weights for the decision matrix were determined in Table 5. The types of batteries included are all batteries that exist in other e-bike products on the market. It was determined that a Li-po battery was the most appropriate option for the project. Lead-Acid and NiMh batteries were inferior to Li-ion and Li-po batteries in nearly ever category listed. Although Li-po and Li-ion batteries are very comparable, Li-po batteries are more readily available and easier to purchase.

Table 5: Battery Type Pairwise Comparison

	Lifespan	Safety	Weight	Sustainability	Energy Density	Cost	Weights
Lifespan	1	1	4	2	0.5	2	**0.22**
Safety	1	1	3	2	1	1	**0.22**
Weight	0.25	0.333	1	0.333	2	0.5	**0.08**
Sustainability	0.5	0.5	3	1	1	1	**0.15**
Energy Density	2	1	0.5	1	1	2	**0.18**
Cost	0.5	1	2	1	0.5	1	**0.15**

Table 6: Decision Matrix for Battery Type

	Weight	Lead-Acid	NiMh	Li-ion (with BMS)	Li-po (with BMS)
Lifespan	0.22	0.1	0.2	0.4	0.3
Safety	0.22	0.18	0.25	0.27	0.3
Weight	0.08	0.1	0.2	0.3	0.4
Sustainability	0.15	0.1	0.2	0.35	0.35
Energy Density	0.18	0.1	0.2	0.35	0.35
Cost	0.15	0.4	0.3	0.1	0.2
Score		0.1626	0.226	0.302	0.310

A decision matrix was used to determine the best placement for the motor. The weight distribution of the bicycle after motor installation, the total weight of the bicycle after installation, and the ease of installation were considered when making this decision. The decision matrix is shown in Table 8. The weights calculations are shown in Table 7. The back wheel drive option refers to a motor that physically rests on the back tire of the bicycle. When the motor spins it spins the back wheel of the bicycle along with it, providing pedal assistance. The back wheel drive motor is the easiest for the user to install because it does not require that either wheel of the bicycle be removed, nor does it's installation require the use of special bicycle tools.

Table 7: Motor Placement Pairwise Comparison

	Weight Balance	Total Weight	Ease of Installation	Weights
Weight Balance	1	1	0.5	**0.25**
Total Weight	1	1	0.5	**0.25**
Ease of Installation	2	2	1	**0.5**

Table 8: Decision Matrix for Motor Placement

	Weight	Rear Hub	Central	Front Hub	Back Wheel Drive
Weight Balance	0.25	0.2	0.2	0.2	0.4
Total Weight	0.25	0.2	0.2	0.25	0.35
Ease of Installation	0.5	0.15	0.15	0.2	0.5
Score		0.175	0.175	0.2125	0.4375

3.2 Technical Requirements

From the qualitative Customer Requirements, various technical targets were set in the form of Engineering Requirements. The Engineering Requirements are as follows:

1. The bicycle should be able to maintain a speed of 17 mph.
2. Production cost should not exceed $600.
3. The sensor system should be at least 90% accurate in detecting and alerting to potential obstacles.
4. The complete system should not contain more than 5 pieces that the user needs to install.
5. The battery life should last for at least 30 miles of city travel.
6. The weight of the kit components should not exceed 15lb.

Table 9 shows the requirements for the system, including the Marketing (Customer) and Engineering requirements.The justifications for each Engineering requirements are listed, as well as how they each relate to the Marketing requirements.

Table 9: System Requirements

Marketing Requirements	Engineering Requirements	Justification
1, 2, 5	1. The bicycle should be able to maintain a speed of 17 mph.	The specified speed will ensure the user can commute efficiently without compromising safety.
3	2. Production cost should not exceed $600.	This is based upon competitive benchmarking and exisiting technologies.
1	3. The sensor system should be at least 90% accurate in detedcting and alerting to potential obstacles.	Too many false positive readings from sensors would make the system more dangerous.
2, 4	4. The complete system should not contain more than 5 pieces that the user needs to install.	This can be done by combined components together to form 5 or less discrete modules
2, 5	5. The battery life should last for at least 30 miles of city travel.	This is based upon competitive benchmarking, as well as the usability and sustainability of the design.
1, 2, 4, 5	6. The weight of the kit components should not exceed 15lb.	Added weight to the system will decrease efficiency and maximum speed.
Marketing Requriements 1. The system should be safe and promote additional safety for its users. 2. The system should be environmentally sustainable. 3. The system should have low cost as compared to similar products. 4. The system should be easy-to-install for users without an advanced knowledge of bicycle mechanics. 5. The system should allow for faster commute times than travel on a normal bicycle.		

Table 10 shows the Customer and Engineering requirements in the form of an Engineering-Marketing Tradeoff Matrix, where the direction of improvements are shown with the plus and minus signs, and the positive and negative correlations are shown with arrows. No arrow represents no correlation. The purpose of this table is to understand the relationships between the Engineering Requirements and the Customer Requirements, in terms of how adjusting one will affect the other.

Table 10: Engineering-Marketing Trade-off Matrix

		Speed	Cost	Sensor Accuracy	# of Components	Battery Life	Weight
		+	-	+	-	+	-
1) Safety	+	↓↓		↑↑			↑
2) Environmental Sustainability	+	↓			↑	↑	↑
3) Cost	-			↑↑			
4) Install Ease	+				↑↑		↑
5) Commute Time	-	↑↑				↑	↑

From this Tradeoff Matrix, it is seen, for example, that when the speed of the bike is increased (denoted by the +), the commute time is decreased (denoted by the -). Therefore, these two requirements have a strong positive correlation.

Table 11 shows the Engineering requirements as an Engineering Tradeoff matrix, where the positive and negative correlations among the engineering requirements is displayed, similar to the previous table. This helps to show how each of the requirements affect each other, and to what extent they are all simultaneously possible.

Table 11: Engineering Trade-off Matrix

		Speed	Cost	Sensor Accuracy	# of Components	Battery Life	Weight
		+	-	+	-	+	-
Speed	+		↓				↑
Cost	-			↓	↓	↓	↓
Sensor Accuracy	+						
# of Components	-						
Battery Life	+						↑
Weight	-						

While some Engineering Requirements are unrelated to one another, some are interdependent. For example, if the speed of the bike was to be increased (improved), the cost would increase (worsen).

Table 12 shows competitive benchmarks for similar e-bike conversion kit products, and compares them to our design. This analysis demonstrates that this product is competitive with similar products on the market [8] , [20].

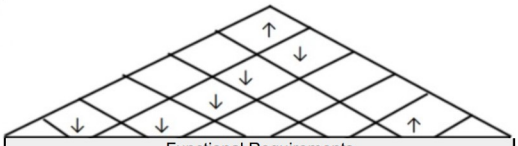
Table 12: Competitive Benchmarks

	Vekkit	Revos	Our Design
Max Speed	15 mph	15.5 mph	17 mph
Cost	$650	$780	$600
Battery Range	30 mi	18 mi	30 mi
Weight	7 lb	6 lb	15 lb

The House of Quality, shown in Table 13, combines information from the previous tables along with customer importance and relative weight information. Therefore, in this table, the requirements are summarized and the importance of each requirement is stated. The customer importance was determined based on the priority of the customer requirements. The relative weights were determined by dividing the Customer Importance value by the sum of all the Customer Importance values.

Table 13: House of Quality

Project:	Smart E-Bike Conversion Kit and Helmet
Date:	11/8/2020



			Functional Requirements					
		Direction of Improvement	+	-	+	-	+	-
Relative Weight	Customer Importance	Direction of Improvement	Customer Requirements	Speed	Cost	Sensor Accuracy	# of Components	Battery Life	Weight
33%	9	+	1) Safety	↓↓		↑↑			↑
11%	3	+	2) Environmental Sustainability	↓			↑	↑	↑
19%	5	-	3) Cost			↑↑			
26%	7	+	4) Install Ease				↑↑		↑
11%	3	-	5) Commute Time	↑↑				↑	↑
			Importance Rating Sum (Importance x Relationship)	135	27	139	95	51	115
			Relative Weight	24%	5%	25%	17%	9%	20%

Direction of Improvement	
Maximize	+
Minimize	-

Relationships		Weight
Very Positive	↑↑	9
Positive	↑	5
No Relationship		1
Negative	↓	5
Very Negative	↓↓	9

Correlation	
Positive	↑
None	
Negative	↓

From this table, it is shown that sensor accuracy and speed are of the greatest relative weights, with weight closely following. Cost and battery life, however, are shown to be less consequential.

3.3 System Description

The system was designed using a process of functional decomposition, designed at multiple hierarchies to determine necessary input and output parameters for the various subsystems.

At its most basic level, the system will take in inputs of power (Voltage), an on/off signal, and sensor input and produce pedal assist and warning signals to the user. This Level Zero Functionality block diagram is shown in Fig. 3.

Figure 3: Level zero e-bike functionality

Table 14 describes in further detail the logistics of the input and output parameters, as well as a description of the functionality at this hierarchical level.

Table 14: Level zero e-bike functionality

Module	E-bike
Inputs	Sensor Input: Distance, Photoresistor, Pressure-sensitive Resistors Power: 22.2V battery
Outputs	Pedal Assist: Friction motor Warning Signals: Audio, Lights, Braking
Functionality	The system will take in inputs from the on/off switch to turn on battery and motor for the bike to output pedal assist. The user will be able to control this using increment and decrement speed buttons. Receive inputs from sensors and process data to output warnings and signals to the motor.

The Level One functional design breaks the system up into two subsystems: Battery/Motor and Sensor/Helmet. Although these systems do interact with each other in reality, they can be largely designed independently. Fig. 4 shows the block diagrams for the Level One Functionality.

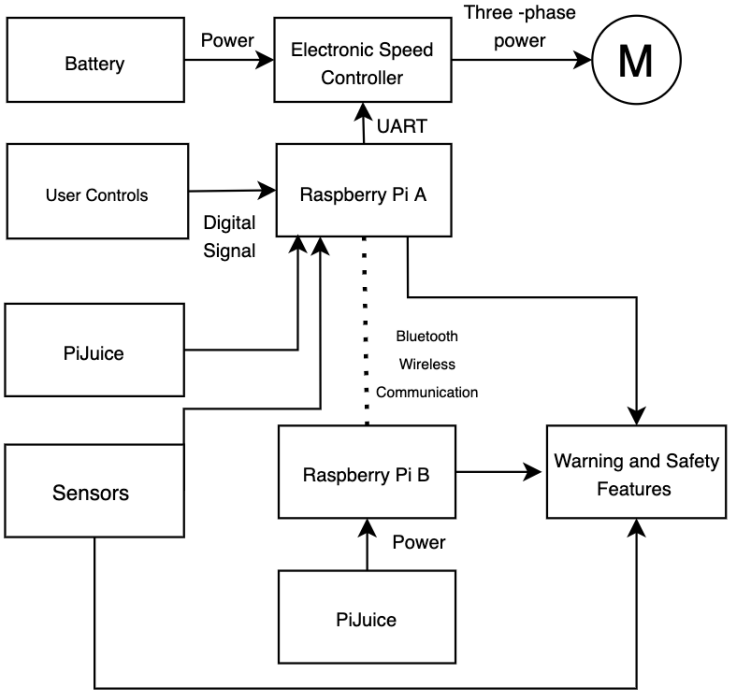


Figure 4: Level One E-Bike Functionality

Tables 15-21 break down the parameters and functionality for the Level One E-Bike. The electronic speed controller, as described in Table 15 is an essential part of the e-bike design because it ensures the motor and battery are functioning under safe voltage and current levels. The Vedder's Electronic Speed Controller (VESC) software allows for the speed controller to be programmed to stop operating once the battery voltage falls beneath a safe threshold. Programming the speed controller also functions as a safeguard against drawing too much current from the battery.

Table 15: Electronic Speed Controller

Module	ESC
Inputs	UART Signal
Outputs	PWM Signal
Functionality	Provides power to motor as indicated by PWM signal

The Raspberry Pi computers are fundamental to the motor control circuit as well as the warnings and safety features system. In terms of motor control, the Raspberry Pi is used to interpret information from the User Controls (see Table 18) and send the appropriate signals to the speed controller.

Table 16: Raspberry Pi Computers

Module	Raspberry Pi Computers
Inputs	Sensor data, control information, power
Outputs	Warning signals
Functionality	Processes information from sensors and user controls. Outputs signals for motor control and safety features.

The sensors subsystem is comprised of the components used to obtain information regarding

the surrounding environment of the cyclist. The information from the sensors is used to activate the safety features described in Table 20.

Table 17: Sensors

Module	Sensors
Inputs	Power from Raspberry Pi, trigger signal, sunlight
Outputs	Digital signals (5V, 0V)
Functionality	Sensors relay information about the proximity of surrounding objects.

Table 18: User Controls

Module	User Controls
Inputs	Power from Raspberry Pi
Outputs	Digital signals for changing the speed of the motor
Functionality	User can change the speed of the motor by pressing buttons near the handlebars of the bicycle.

A flowchart for the User Controls Subsystem is shown in Fig. 5 to describe the high-level process and algorithm. This code is executed in Core 4 of the Raspberry Pi A.

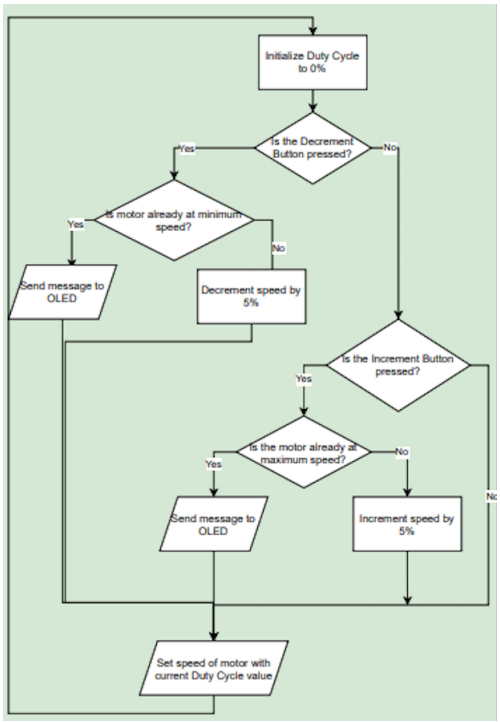


Figure 5: Flowchart for Battery/Motor System

The motor is controlled using a combination of hardware and software to change the duty cycle of the motor. Changing the duty cycle alters the speed of the motor, and thus the speed of the bicycle. Furthermore, Universal Asynchronous Receiver/Transmitter (UART) Communication is used to send information to from the Raspberry Pi to the ESC. The signal sent from the ESC to the motor is a pulse width modulated (PWM) signal. In pulse width modulation, the amplitude and frequency of signal stays constant and the width of each pulse is modulated. The duty cycle of the signal is calculated using (1). The duty cycle is adjusted according to what

the desired DC level is. The higher the duty cycle, the higher the average DC level, and the lower the duty cycle, the lower the average DC level.

$$\text{Duty Cycle} = \frac{d}{t_c} * 100\% \tag{1}$$

In (1), d represents the number of base clock cycles for which the output should be high, while t_c is the device cycle time.

The battery used to power the motor is equipped with a battery management system (BMS). Since this e-bike is designed to keep the user safe, it was important the battery have this safety feature. The functionality of the battery is further described in Table 19.

Table 19: Battery

Module	Battery
Inputs	Power (while charging)
Outputs	Power
Functionality	Powers the motor

Table 20 shows the functionality of the Warning and Safety Features. Sensors take in information about the user's surroundings. This information is processed in the Raspberry Pi's and displaying on a variety of warning devices.

Table 20: Warning and Safety Features

Module	Warning and Safety Features
Inputs	Power from Raspberry Pi, display information, digital signals relaying sensor information
Outputs	Visual and auditory warnings to the user.
Functionality	Alert the user to potential obstructions, and alert those nearby to the presence of a cyclist.

Table 21 shows the functionality of the PiJuice Hat module. The PiJuice includes a battery to power the Raspberry Pi and sits on top of the Pi's GPIO pins. The device is also capable of being programmed for safe shut downs, battery warnings, and button customizations.

Table 21: PiJuice

Module	PiJuice
Inputs	Power (while charging)
Outputs	Power to Raspberry Pi computers
Functionality	Powers Raspberry Pi computers so they do not need to remain plugged in.

The User Controls, outlined in Table 18, can be broken up into Level two functionality as shown in Fig. 6.

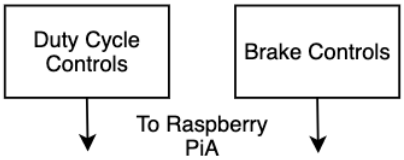


Figure 6: Level Two Controls

Table 22 outlines the functionality of the Duty Cycle Controls. The changing duty cycle controls the speed of the motor.

Table 22: Duty Cycle Controls

Module	Duty Cycle Controls
Inputs	Power
Outputs	Digital signal to Raspberry Pi
Functionality	Allow the user to change the speed of the motor by changing the duty cycle.

Table 23 outlines the functionality of the Brake Controls, which allows the user to stop the motor when they apply the bicycle brakes.

Table 23: Brake Controls

Module	Brake Controls
Inputs	Power
Outputs	Digital signal to Raspberry Pi
Functionality	Deactivates the motor when the brake button is pressed.

Fig. 7 shows the level two functionality for the sensors. This system can be broken into two different parts: the HC-SR04 ultrasonic sensors and the photosensitive lights.

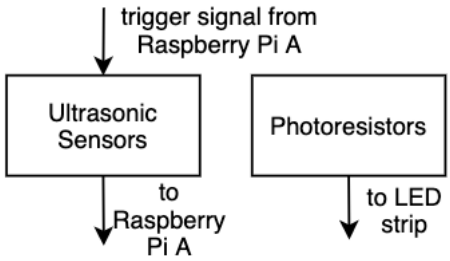


Figure 7: Level Two Sensors

Table 24 shows the level two functionality for the ultrasonic sensors. Theses sensors are part of the system of obstacle detection. The distances are calculated from the five sensors and then used to determine the probability of potential hazards or obstacles.

Table 24: Ultrasonic Sensors

Module	HC-SR04 Ultrasonic Sensors
Inputs	Power and trigger signals
Outputs	Measurements about distance
Functionality	Used to determine if there are potential obstacles in dangerous proximity to the cyclist.

There will be five HC-SR04 distance sensors as a part of the conversion kit, placed as shown in Fig. 8. Each sensor can be referenced by its corresponding label (S1-S5).

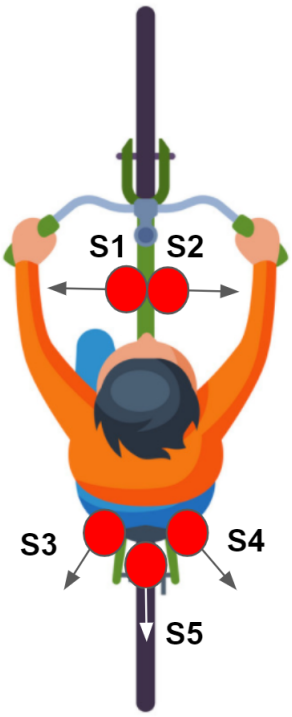


Figure 8: Placement of Distance Sensors

The Obstacle Detection System will alert the rider if an obstacle is detected within the ”Very Close” distance range of S3, S4, or S5. Currently, ”Very Close” is defined as 40 cm, although this benchmark will be improved with testing.

Additionally, the system will detect if an obstacle is moving closer to sensors S3, S4, or S5 within the ”Moderately Close” distance. The ”Moderately Close” distance is defined as a maximum of 200 cm currently. However, if the distance detected from S1 and S2 is less than the set ”Moderately Close” distance, than S3, S4, and S5 will use that new distance as the ”Moderately Close” measurement. This was determined to avoid false positives if the rider is riding consistently close to parked cars or some sort of structure that they can see clearly.

The theoretical range of the HC-SR04 modules is 450 cm at $\pm 15^{\circ}$. The experimental range of the module was determined using the requirements set in assignment ”2 The Testing of the HC-SR04 Ultrasonic Ranging Module” for the LMU ELEC 401 course [2]. The experimental results are shown in *Data Analytics*, Fig. 75. Therefore, an estimate of $\pm 15^{\circ}$ can be used for the following calculations.

The rider should not be alerted simply if a car is passing them within the ”Moderately Close” distance, if the car is not moving closer to the bike perpendicularly. Therefore, Fig. 9 shows

how distances can be calculated to determine if the obstacle is actually moving closer to the bike, or if it will just pass.

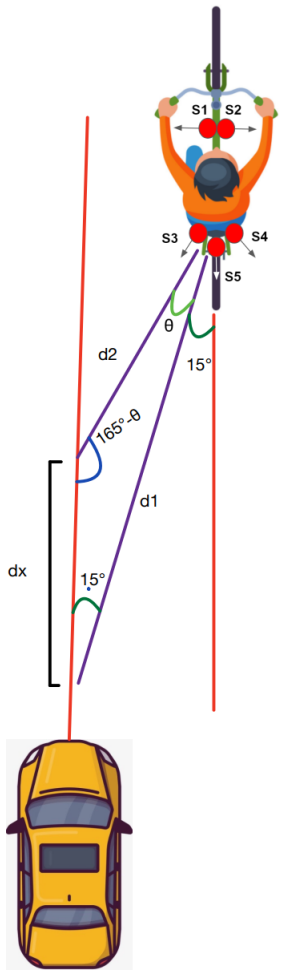


Figure 9: Calculation of Safe Passing Distance

If it is assumed that the distance 'dx' is parallel with the bike, it can be calculated what the distance 'd2' should be if the car is not on track to hit the cyclist. This calculation is shown in (2).

$$d_2 = \frac{\mathrm{d}_1 sin(15°)}{sin(165° - \theta)} \tag{2}$$

If the measured d2 from S3 or S4 is then smaller than the calculated value, it is known that the car is a potential threat and the rider will be alerted.

Potential scenarios for Obstacle Detection and their designed outcomes and justifications are included in *Obstacle Detection Scenarios* subsection.

Equation 3 was used to determine the maximum velocity of an object traveling towards the bike for which the rider would have enough time to react to the obstacle.

$$\text{Max Velocity} = \frac{\text{"Dangerously Close" Distance}}{\text{Average Reaction Time}} = \frac{2m}{215ms} = 9.3m/s \tag{3}$$

The algorithm for detecting obstacles is outlined in a flowchart shown in Fig. 10. The obstacle detection code is executed on Core 1 of Raspberry Pi A and sends information to other processes (cores) that are running in parallel.

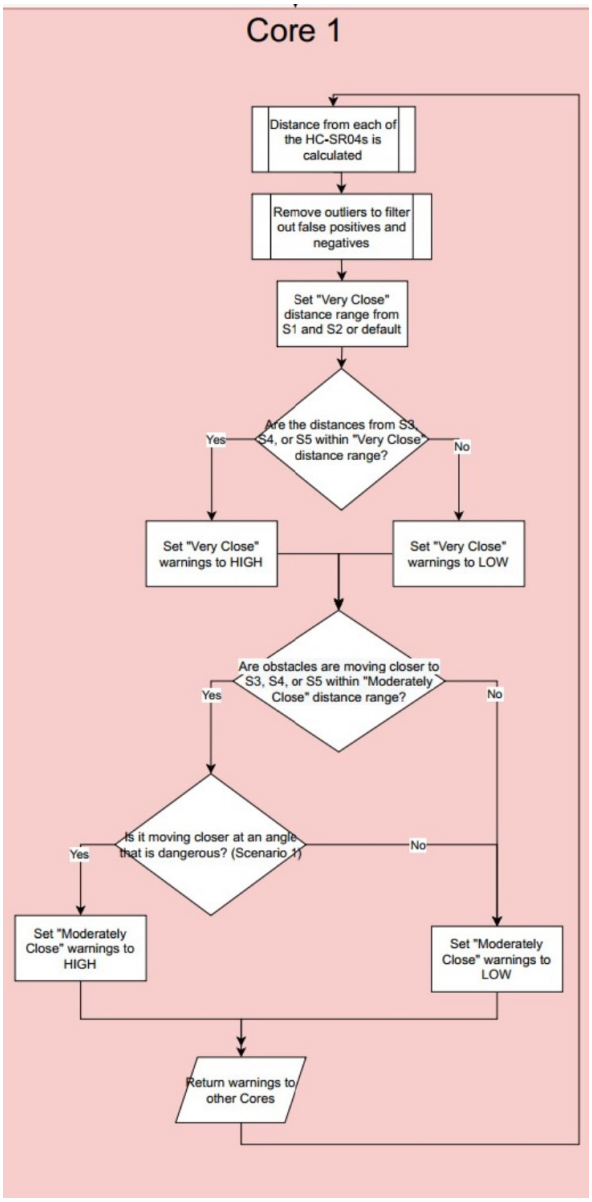


Figure 10: Flowchart for Obstacle Detection

The photosensitive lights use photosensitive resistors to take input from the sun or other sources of light and activate LEDs accordingly. At night, or if it is dark, the LEDs will be illuminated. If it is daytime or there is external light, the LEDs will turn off. The system will be placed on the helmet to alert other people to the user's presence. Table 25 shows the level two functionality of the photosensitive lights. The system utilizes the varying resistance of photoresistors in response to the amount of exposed light.

Table 25: Photosensitive Lights

Module	Photoresistors
Inputs	Light
Outputs	Analog Voltage (0V-Vcc)
Functionality	Relays information about the light present in the surrounding environment

Fig. 11 shows the Level two functionality of the warnings and safety features on the system. a combination of lights, buzzers, and screens produce a variety of warnings to the user and to surrounding people.

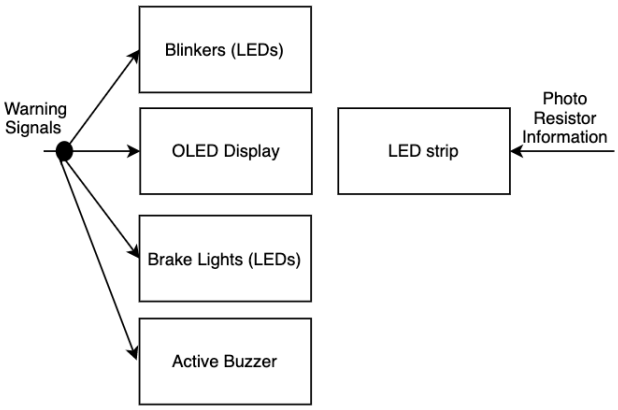


Figure 11: Level Two Warnings and Safety Features

Table 26 shows the functionality of the blinker buttons and signals at this level. Blinker buttons exist on Raspbery Pi A. If a user presses the blinkers, a signal is sent to Raspberry Pi B to activate the turn signal lights (LEDs).

Table 26: Blinkers

Module	Blinkers
Inputs	Vcc, button press
Outputs	0 or Vcc output voltage
Functionality	Provides information to Raspberry Pi B (helmet) to activate turn signal lights

There are left and right blinker buttons on the bike that the user can press to enable turn signals on the helmet. Fig. 12 shows a simple schematic of the blinker system.

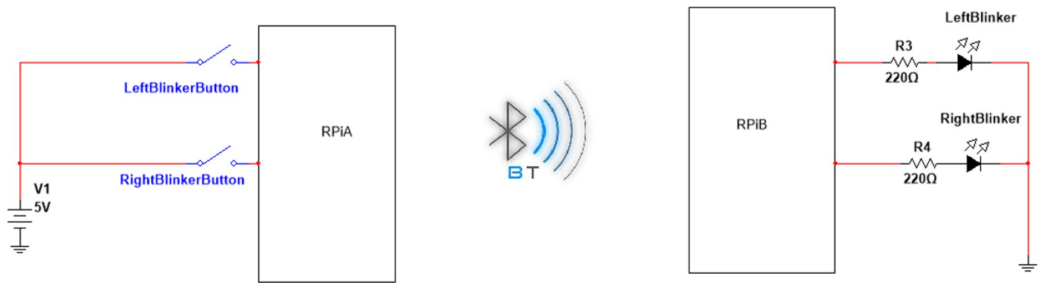


Figure 12: Blinker Schematic

Table 27 shows the functionality of the Organic Light Emitting Diode (OLED) Screen at this level. The screen serves as communication to the user to present warnings and status updates. The screen physically sits on the handlebars of the bike.

Table 27: OLED Display

Module	OLED Display
Inputs	I2C Serial Communicationr
Outputs	Display messages on screen
Functionality	Display messages and warnings to the user about obstacles, battery life, and speed

Table 28 shows the functionality for the brake lights at this level. The brake lights are activated when the user presses the brake. Attached to the brakes are pressure sensitive resistors. The voltage across these resistors are read by Raspberry Pi A to determine when the brakes are pressed. A signal is then sent to Raspberry Pi B to activate the brake lights (LEDs) on the helmet).

Table 28: Brake Lights (LEDs)

Module	Brake Lights (LEDs)
Inputs	Pressure Resistor, Power
Outputs	0 or Vcc output voltage
Functionality	Send information to Raspberry Pi B (helmet) to activate brake lights when the user presses the brake

Table 29 shows the functionality for the active buzzer at this level. The buzzer responds to obstacle hazards calculated in Raspberry Pi A. A signal is then sent to Raspberry Pi B to activate the GPIO pins connected to the corresponding buzzers.

Table 29: Active Buzzer

Module	Active Buzzers
Inputs	GPIO pin from Raspberry Pi B (output)
Outputs	Sound when GPIO is HIGH
Functionality	Alerts the user to potential obstacles with audio signal

Table 30 shows the functionality of the LED strip at this level. The strip sits on the helmet and activates at night or in the dark to illuminate the user while they are riding and make them visible to other people.

Table 30: LED Strip

Module	LED Strip
Inputs	Voltage output from photoresistor amplifier circuit
Outputs	LED strip activated
Functionality	Activate lights when it is dark (night), and keep lights off when it is light (day)

Some warnings are calculated and processed on Raspberry Pi A and then sent to the helmet (Raspberry Pi B) to enable buzzer, LED, blinkers, and brake light warnings. This wireless communication is conducted via Bluetooth. The code for the Bluetooth communication is

executed on Raspberry Pi B and Core 2 of Raspberry Pi A. Fig. 13 and Fig. 14 shows flowcharts of the wireless communication algorithm.

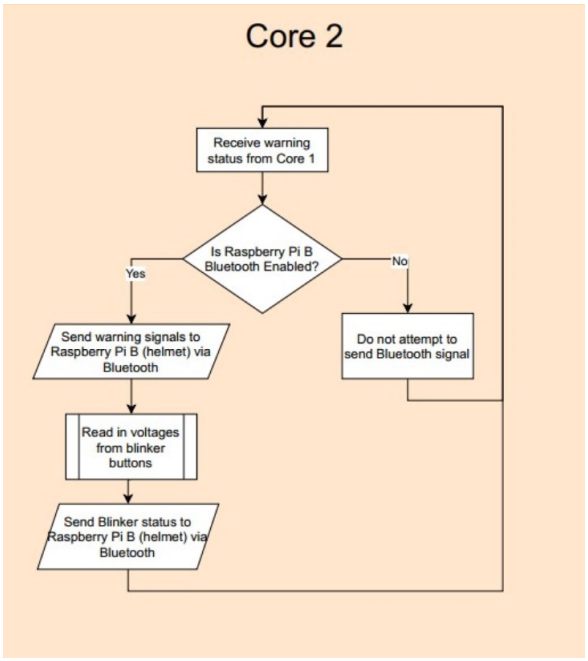


Figure 13: Raspberry Pi A Bluetooth Flowchart

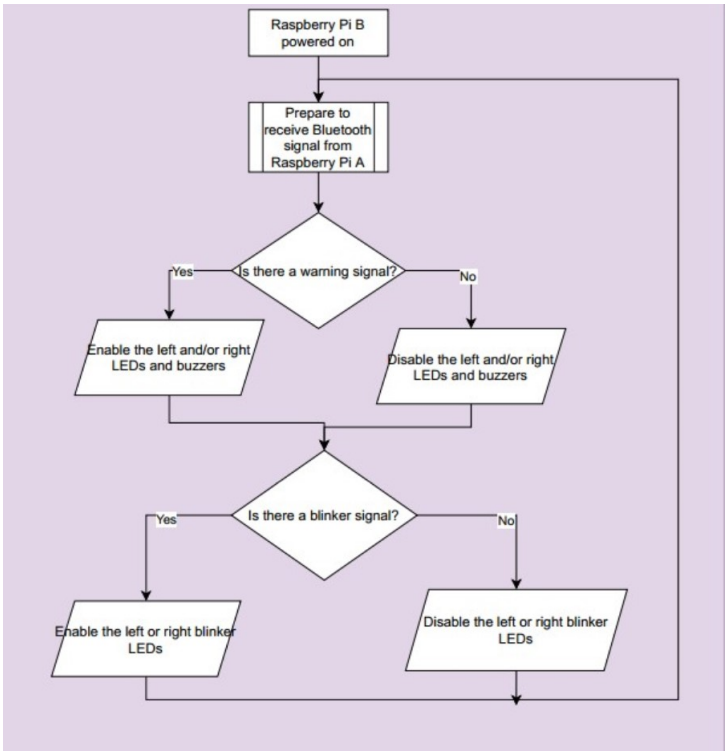


Figure 14: Raspberry Pi B Bluetooth Flowchart

The active buzzers and LEDs on the helmet respond to the obstacle detection calculations from Raspberry Pi A. The OLED displays obstacle detection warnings, speed, and low battery warnings.

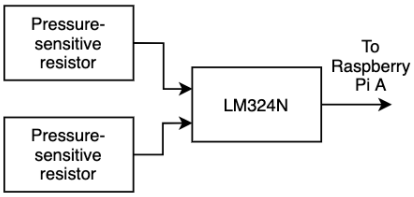


Figure 15: Level Three Controls - Brake Controls

Table 31: Pressure Sensitive Resistors

Module	Pressure Sensitive Resistors
Inputs	Power
Outputs	Digital signal to Raspberry Pi
Functionality	Deactivates the motor when the brake button is pressed.

Table 32: LM324N

Module	LM324N
Inputs	Digital signals from buttons
Outputs	Digital signals to Raspberry Pi
Functionality	An LM324N IC is used to form a voltage follower circuit between the buttons and the Raspberry Pi. Through testing it was determined the voltage follower was needed to ensure digital signals are readable by the Raspberry Pi.

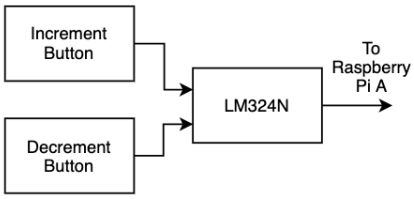


Figure 16: Level Three Controls - Duty Cycle Controls

The duty cycle controls allow the user to control the speed of the bicycle, and are therefore fundamental to the safety and functionality of the bicycle. The increment and decrement buttons are implemented using tactile square buttons placed on the handlebars of the bicycle. A voltage follower is placed on the output of each button to ensure the output digital signal is readable by the Raspberry Pi. Tables 33 and 34 further explain the functionality of these buttons.

Table 33: Increment Button

Module	Increment Button
Inputs	5V Vcc
Outputs	Digital signal
Functionality	Allow user to increment the speed of the motor

Table 34: Decrement Button

Module	Decrement Button
Inputs	5V Vcc
Outputs	Digital signal
Functionality	Allow user to decrement the speed of the motor

Fig. 17 shows the Level three functionality of the HC-SR04 Ultrasonic Sensor Modules. The five modules are connected to Raspberry Pi A with four pins each: 5V, Ground, Trigger, and Echo.

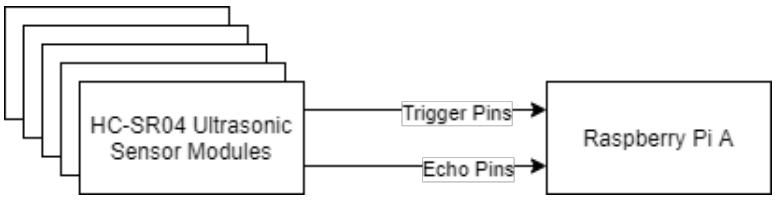


Figure 17: Level three HC-SR04 Ultrasonic Sensor Modules

Table 35 shows the functionality of the HC-SR04 sensors at this level. Table 36 shows the functionality of Raspberry Pi A at this level.

Table 35: Ultrasonic Sensor Modules

Module	Ultrasonic Sensor Modules
Inputs	Trigger, Vcc, and Ground
Outputs	Echo Signal
Functionality	Sends out ultrasonic pulse and the distance to the nearest obstacle is encoded in the width and time of the echo signal

Table 36: Raspberry Pi A

Module	Raspberry Pi A
Inputs	Echo Signal
Outputs	Trigger Signal, 5V, Ground
Functionality	Used to calculate distance in cm and then alert to detected obstacles

The Trigger and Echo pins of the HC-SR04 module collect the data necessary to determine the distance, as shown in the flowchart. As shown in the Timing Diagram in Fig. 18, when a short pulse is supplied to the trigger input the sensor sends out an 8 cycle burst of ultrasound at 40kHz. An echo pulse is then detected by the Ultrasonic sensor, and the echo pulse width is proportional to distance to the target object. The distance to the target is then calculated using the equation (4).

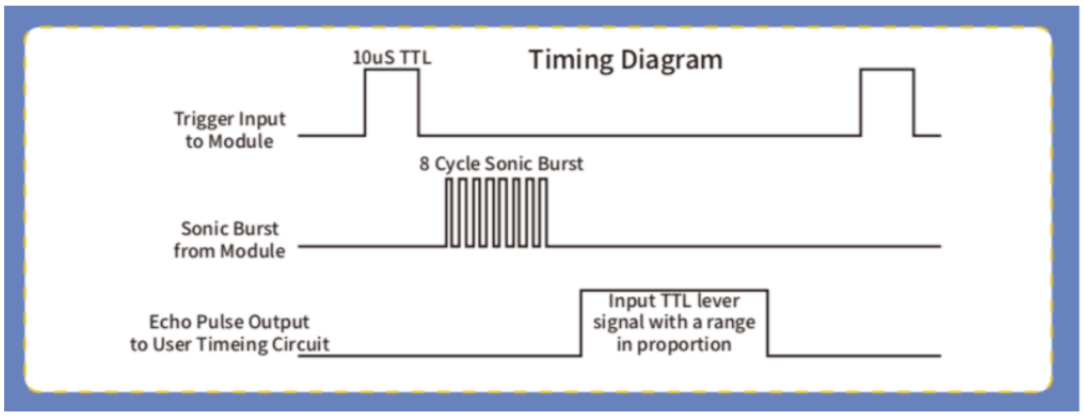


Figure 18: HC-SR04 Timing Diagram

$$\text{Distance} = \frac{\text{High Level Time of Echo Pulse * Velocity of Sound}}{2} \tag{4}$$

This distance is calculated from the width and timing of the echo pulse.

Fig. 19 shows the Level three functionality for the photosensitive lights. The lights are physically located on the helmet, but are not connected to the Raspberry Pi.

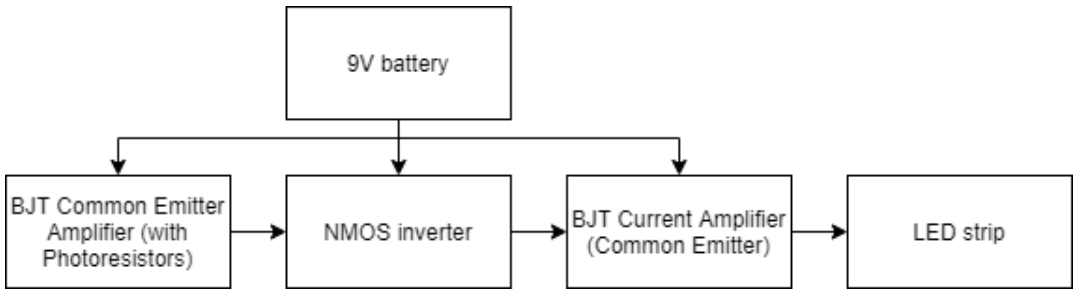


Figure 19: Level three photosensitive lights functionality

The circuit is a three stage amplifier in order to output the correct voltage and current levels needed to activate the LED strip. Tables 37, 38, and 39 show the functionalities of the amplifiers specified.

Table 37: BJT Common Emitter Amplifier

Module	BJT Common Emitter Amplifier
Inputs	Vcc, variable photoresistancel
Outputs	High and Low voltage
Functionality	Depending on where there is light present, a high or low voltage will be output

Table 38: NMOS Inverter

Module	NMOS Inverter
Inputs	High or Low Voltage from Common Emitter
Outputs	Inverted voltage
Functionality	Will invert high voltage to low voltage and vice versa for a Vcc of 9V

Table 39: Emitter Follower Current Amplifier

Module	Emitter Follower Current Amplifier
Inputs	High or Low Voltage from Inverter
Outputs	Similar voltage as input with amplified current
Functionality	Amplifies current from inverter to power the LED strip

Fig. 20 shows the level three functionality of the OLED's interface. The OLED uses I2C, a serial communicatin protocol, to communicate with the Raspberry Pi A. The display has four pins: 3.3V, Ground, SDA, and SCL. SCL is the clock signal and SDA is the data signal.

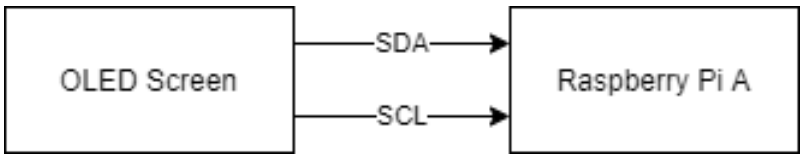


Figure 20: Level three OLED screen functionality

Table 40 shows the functionality of the OLED Screen at this level. Table 41 shows the functionality of Raspberry Pi A at this level.

Table 40: OLED Screen

Module	OLED
Inputs	I2C Serial Communication: Clock Signal (SCL) and Data Signal (SDA)
Outputs	Display message on screen
Functionality	Displays warnings and messages to user

Table 41: Raspberry Pi A

Module	Raspberry Pi A
Inputs	N/A
Outputs	I2C Serial Communication: SCL and SDA
Functionality	Sends relevant messages to the screen

The code for the OLED communication is executed in Core 3 of Raspberry Pi A. It receives inputs from other cores, running in parallel. Fig. 21 shows a flowchart of the algorithm.

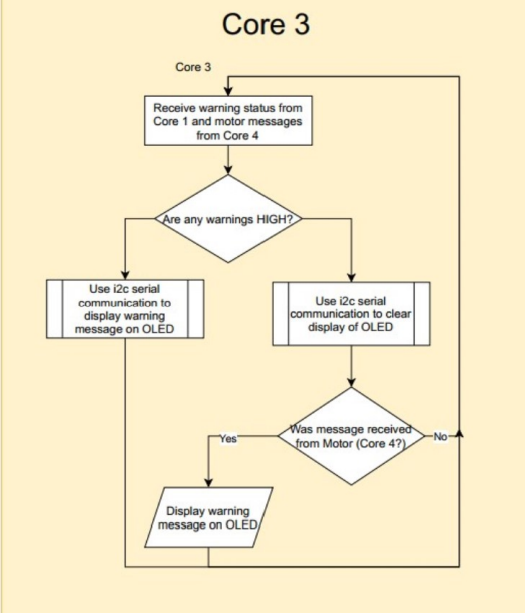


Figure 21: OLED Flowchart

Messages come from other cores to alert to obstacles, battery status, and speed.

Through this process of functional decomposition, four distinct levels were outlined in the hierarchy. The necessary parameters were then able to be determined, as well as the interactions between smaller components and subsystems.

3.4 Standards and Constraints

The design of this e-bike, which is outlined in System Description, was guided by Institute of Electrical and Electronics Engineers (IEEE) standards, and the components utilized in the design, such as the Raspberry Pi computers, were also designed to comply with IEEE standards.

As detailed in the section System Description, the two Raspberry Pi computers incorporated in the design must have the ability to communicate with each other. Raspberry Pi computers have wifi and Bluetooth capabilities that were designed to comply with the IEEE standards for Bluetooth and wireless fidelity. These standards were detailed in working groups 802.15 and 802.11, respectively [13], [12]. Additionally, standards for serial communication - UART, SPI - were adhered to [5]. I2C is a synchronous, serial communication protocol standards were also utilized for interfacing between the Raspberry Pi A and the OLED [11].

Other elements of the design, such as the brushless DC motor, also follow IEEE guidelines. The official definition of a brushless DC motor is included in [1], and motors that deviate from this definition cannot be considered true brushless DC motors.

In addition to ensuring the design of the e-bike aligns with IEEE standards, the design must take into account the constraints associated with constructing an e-bike during a pandemic, as well as monetary constraints. For instance, access to laboratory resources are limited by virtue of the fact that partners must remain in their designated areas. Finally, while the goal of this capstone project is to produce an e-bike that is safe and affordable for most people, the manufacturing cost of a single e-bike will be much greater than the cost per bike if the manufacturing process was scaled up. In other words, the manufacturing cost of the e-bike constructed for this project will not provide an accurate gauge of what actual retail value of the bike should it be manufactured on a larger scale.

3.5 Design Impact

As addressed in the Ethics section, this project has environmental, economic, and safety benefits to its users. Not only does this project address social justice concerns, it also serves as an initiative and learning opportunity for other LMU students. The system can continue to be developed modularly by LMU Engineering students in future years. For example, LIDAR could be implemented to better detect obstacles, or steering assist could be added to the handlebars. Additionally, this project is targeted at people like LMU students. This provides an impact to the LMU Community as thus has local and societal impacts. Additionally, the affordability, safety, and sustainability are what LMU students are lacking in their transportation needs. Whether it be on campus or in the surrounding urban areas of Los Angeles, this e-bike project design meets the needs of LMU community members.

Economically, this project provides a less expensive option to existing conversion kits, which can cost upwards of $1000, and whole e-bikes can cost even more than that. The estimated cost for development and production of this product's prototype is under $600. However, if it were to be produced on a greater scale, the cost would be even lower.

Environmentally, this design provides a sustainable alternative to cars or public transportation. Studies show that people would be more likely to commute using e-bikes instead of regular bikes due to the ease of riding long distances and uphill [9].

Globally, this design appeals to a wide range of audiences. Since many people around the world use bicycles as their primary means of transportation, a conversion kit to turn their existing bicycle into a much faster and safer e-bike would be beneficial. Additionally, many cities have dangerous car and cycling traffic, so a design of this type would allow cyclists to be more safe in high-traffic environments.

4 Electrical Design

4.1 Schematics and Circuit Diagrams

The system is designed using a combination of hardware and software, using two Raspberry Pi's. A 22.2V 5000mAh battery is used to power the VESC. The VESC was programmed using the VESC Tool software, shown in Fig. 22. With this, three phase power is provided to the motor. The motor can be controlled using UART serial communication from the VESC to the microcontroller. The setting for the voltage, current, and duty cycle provided to the motor are configured using the VESC Tool.

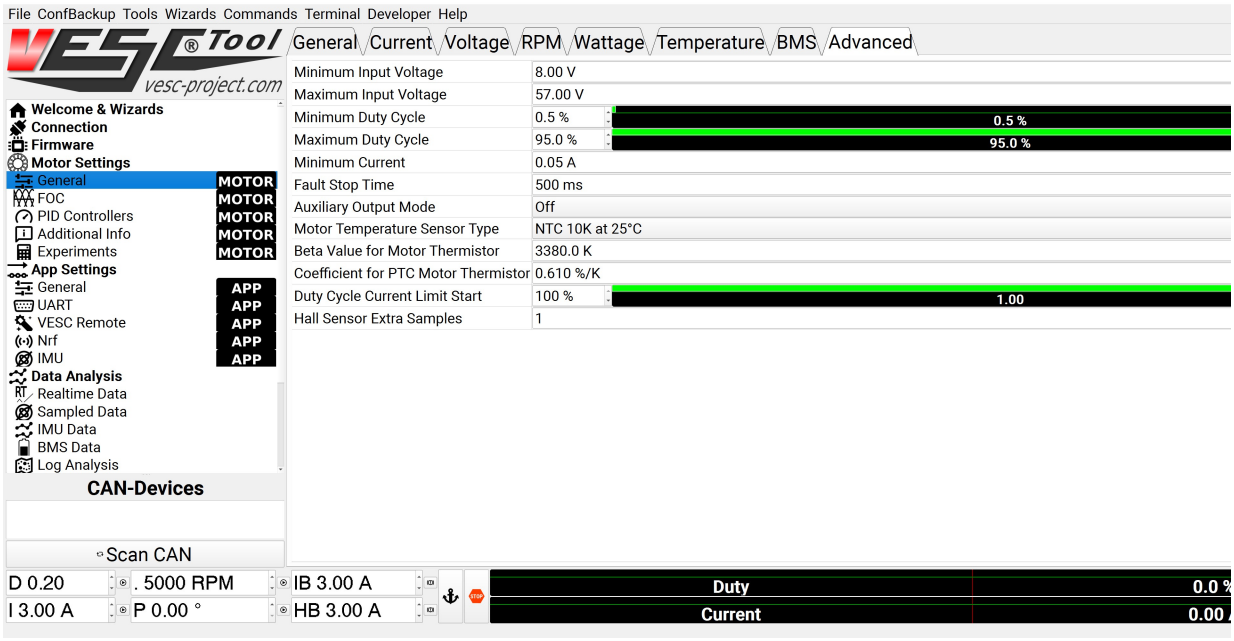


Figure 22: VESC Tool GUI

The user has the ability to control the speed of the motor with controls on the bicycle. The pyVESC Python library was utilized to change the duty cycle according to the user's preferences. The new duty cycle can be set, and that information is then transmitted to the VESC using UART. The VESC adjusts power to the motor accordingly, effectively slowing down or speeding up the bicycle. Additionally, pressure resistors are attached to the hand brakes of the bicycle, so when the user brakes, a signal is sent to the Raspberry Pi; this translates into the VESC shutting off the motor at a safe rate. The overview of the battery and motor circuit is shown in Fig. 23.

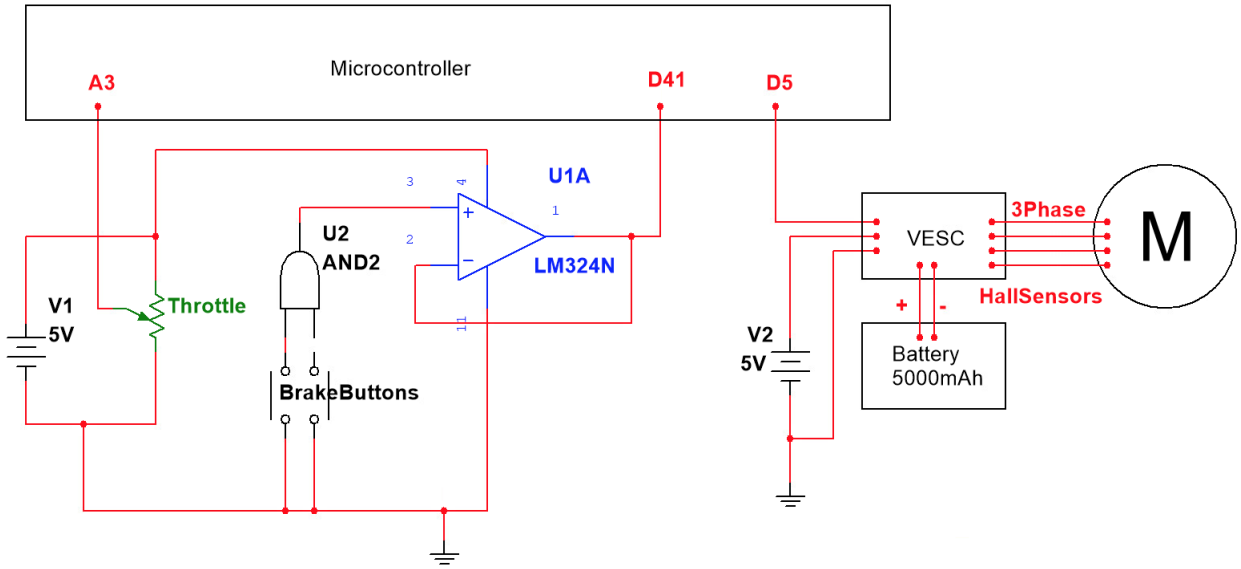


Figure 23: Battery and Motor Schematic

Although most of the controls for the sensors on the project are controlled through the software on the Raspberry Pi's, there is some hardware integration, as well. Fig. 24 shows a schematic for the photosensitive lights on the helmet. R3 represents a photoresistor, whose resistance

changes when exposed to different amounts of light. When there is enough light, the circuit does not provide adequate power to the LED strip. When it is dark, the LED strip will be powered, illuminating the rider.The LED strip requires 7-9V to turn on. The circuit includes a common emitter Bipolar Junction Transistor (BJT) amplifier circuit, with the output connected to an NMOS inverter. The current is then amplified with an emitter follower BJT amplifier circuit. When it is dark, the photoresistor (R3) takes on a value of close to 200kΩ and when it is light, it becomes about 20kΩ.

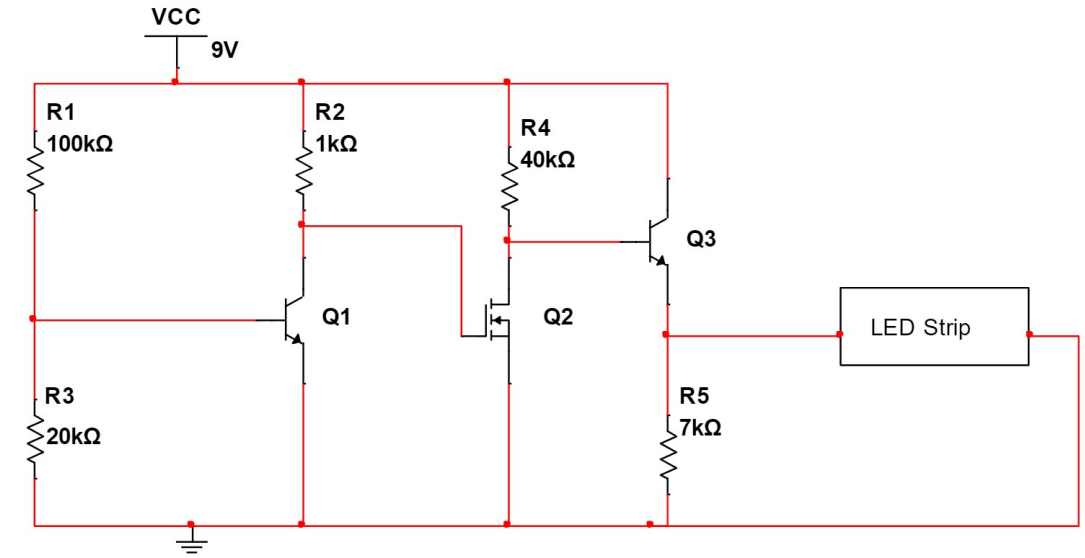


Figure 24: Photosensitive Lights Schematic

There are also turn signal buttons on the bicycle's user controls. The user can press the left or right blinker button; a signal is then transmitted from the Raspberry Pi on the bike (RPiA) to the Raspberry Pi on the helmet (RPiB). The corresponding LEDs are then powered as blinkers to signal the user's turns. This schematic is shown in Fig. 25.

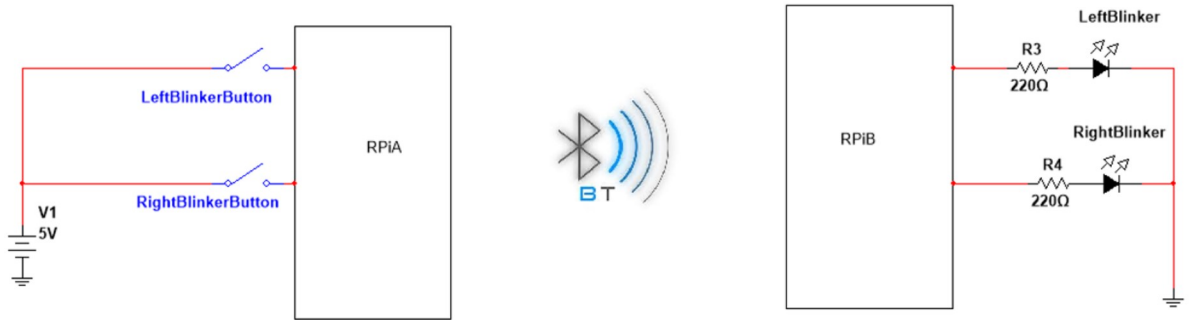


Figure 25: Blinker Schematic

Bluetooth communication is also used to wireless transmit obstacle detection warnings and battery life status.

4.2 Wiring and Cable Diagrams

All of the components for the project are either attached to the bicycle or the helmet. Components on the bicycle are connected to a Raspberry Pi (RPiA) for GPIO and power. A 22.2V 5000mAh Li-po battery also powers the VESC and motor on the bike. The VESC is also connected to RPiA for UART serial communication. The distance sensors, buttons, OLED display, and brakes are all connected to GPIO pins of RPiA. Fig. 26 shows the wiring diagram for all the components on the bike.

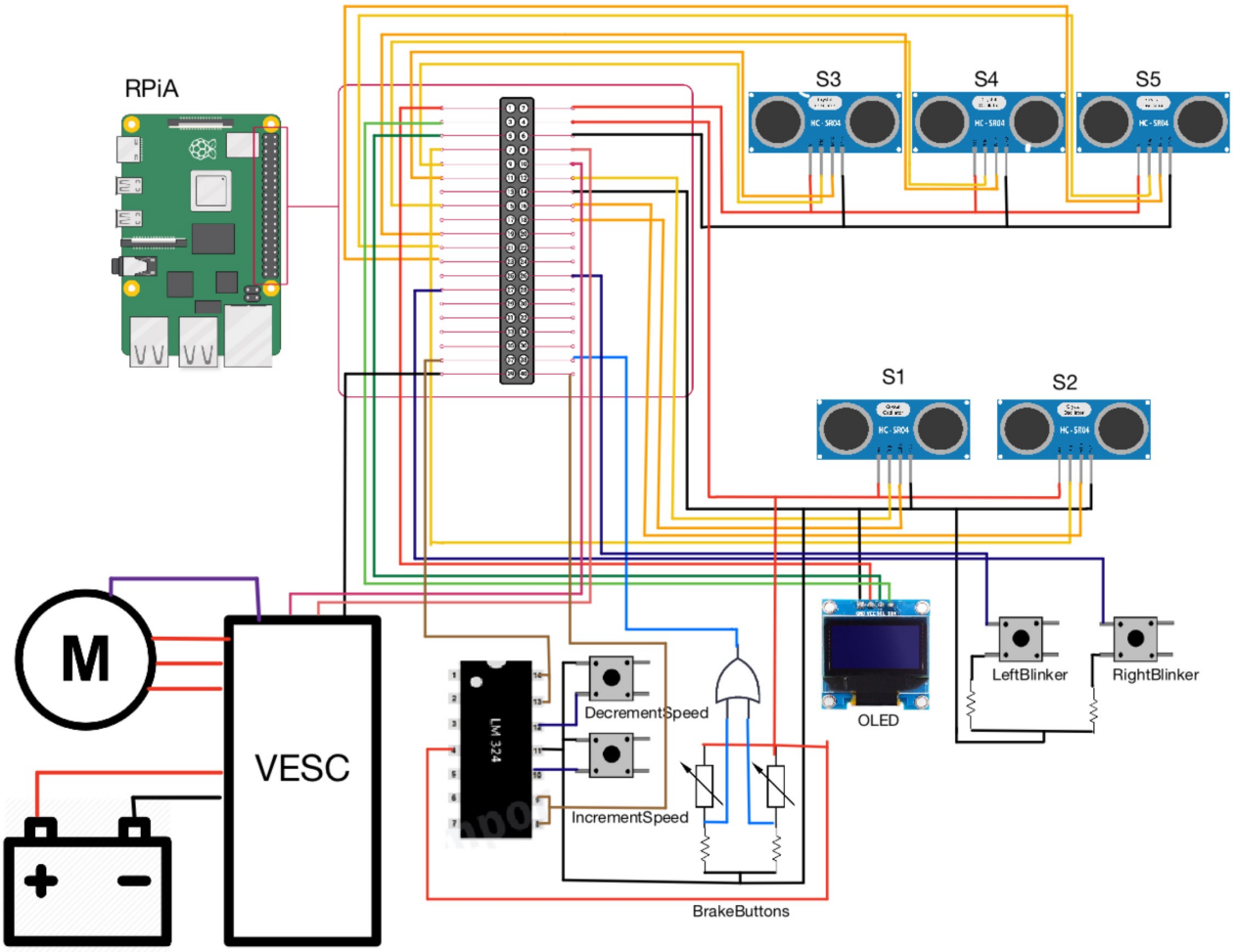


Figure 26: Wiring Diagram for Components Attached to the Bike

The components on the helmet are mostly connected to another Raspberry Pi (RPiB). RPiA and RPiB communicate wirelessly via Bluetooth. Information about obstacles, battery life, and status are exchanged between the two. The warning buzzers and LEDs, blinker and brake lights, and photosensitive lights are all powered on the helmet. Fig. 27 shows the wiring for the components attached to the helmet.

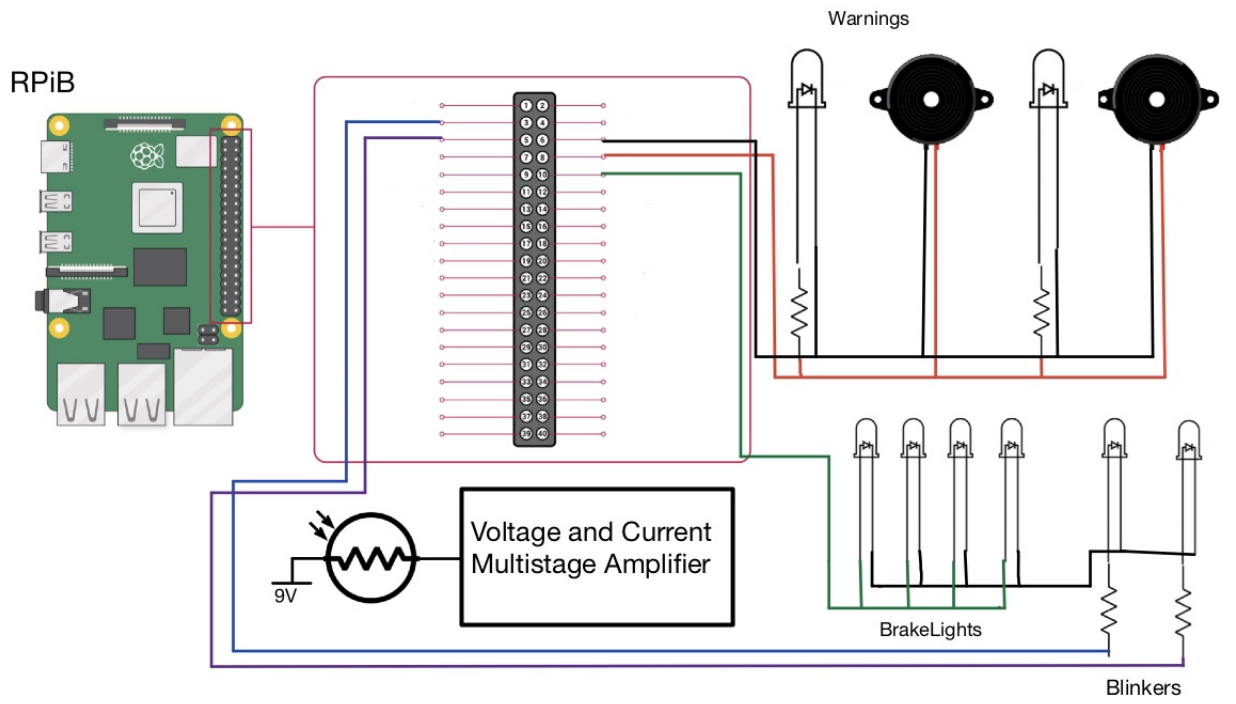


Figure 27: Wiring Diagram for Components Attached to the Helmet

The channelization list of all the connections to the GPIO pins of the Raspberry Pi's are shown in Fig. 28 and Fig. 29. There are GPIO pins still available on both the microcontrollers so components and features could be added in the future.

RPi Pin #	GPIO Pin #	To (Component	Component #	To (Pin)	IO Configurati	Power/GPIO	Signal Type
1		OLED	1	Vcc		Power	3.3V
2		HC-SR04	1	Vcc		Power	5V
2		HC-SR04	2	Vcc		Power	5V
2		HC-SR04	3	Vcc		Power	5V
2		HC-SR04	4	Vcc		Power	5V
2		HC-SR04	5	Vcc		Power	5V
3	2	OLED	1	SDA	Output	GPIO	SDA
4		Photoresistor	1	Vcc		Power	5V
4		Photoresistor	2	Vcc		Power	5V
4		Button	1	Vcc		Power	5V
4		Throttle	1	Vcc		Power	5V
5	3	OLED		SCL	Output	GPIO	SCL
6		HC-SR04	1	Ground		Power	Ground
6		HC-SR04	2	Ground		Power	Ground
6		HC-SR04	3	Ground		Power	Ground
6		HC-SR04	4	Ground		Power	Ground
6		HC-SR04	5	Ground		Power	Ground
7	4	HC-SR04	2	Trigger	Output	GPIO	GPCLK0
8	14	VESC	1		Output	GPIO	TXD
9		OLED	1	Ground		Power	Ground
10	15	VESC			Input	GPIO	RXD
11	17	HC-SR04	3	Trigger	Output	GPIO	
12	18	HC-SR04	1	Trigger	Output	GPIO	PCM_CLK
13	27	HC-SR04	3	Echo	Input	GPIO	
14		Buzzer	1		Ground	Power	Ground
14		Buzzer	2		Ground	Power	Ground
14		LED	1		Ground	Power	Ground
14		LED	2		Ground	Power	Ground
15	22	HC-SR04	4	Trigger	Output	GPIO	
16	23	HC-SR04	2	Echo	Input	GPIO	
17						Power	3.3V
18	24	HC-SR04	1	Echo	Input	GPIO	
19	10	HC-SR04	4	Echo	Input	GPIO	MOSI
20						Power	Ground
21	9	HC-SR04	5	Trigger	Output	GPIO	MISO
22	25	Buzzer	1	Vcc	Output	GPIO	
22	25	LED	1	Vcc	Output	GPIO	
23	11	HC-SR04	5	Echo	Input	GPIO	SCLK
24	8	Buzzer	2	Vcc	Output	GPIO	CE0
24	8	LED	2	Vcc	Output	GPIO	CE0
25		Photoresistor	1	Ground		Power	Ground
25		Photoresistor	2	Ground		Power	Ground
26	7	Blinker Button	1		Input	GPIO	CE1
27	0	Blinker Button	2		Input	GPIO	ID_SD
28	1					GPIO	ID_SC
29	5	Photoresistor	1		Input	GPIO	
30		Button	1	Ground		Power	Ground
31	6	Photoresistor	2		Input	GPIO	
32	12	VESC	1	Control	Output	GPIO	PWM0
33	13		1			GPIO	PWM1
34			1	Ground		Power	Ground
35	19					GPIO	PCM_FS
36	16					GPIO	
37	26	Decrement Button			Input	GPIO	
38	20	Brake Button			Input	GPIO	PCM_DIN
39						Power	Ground
40	21	Increment Button			Input	GPIO	PCM_DOUT

Figure 28: Raspberry Pi A Channelization

RPi Pin	GPIO Pi	To (Component	Component #	To (Pin)	IO Conf	Power/	Signal T
3	2	Blinker	1		Output	GPIO	SDA
5	3	Blinker	2		Output	GPIO	SCL
6		Blinker	1	Ground		Power	Ground
6		Blinker	2	Ground		Power	Ground
6		Buzzer	1	Ground		Power	Ground
6		Buzzer	2	Ground		Power	Ground
6		Brake Lights	1	Ground		Power	Ground
8	14	Buzzer	1	Vcc	Output	GPIO	TXD
10	15	Buzzer	2	Vcc	Output	GPIO	RXD
12	18	Brake Lights	1	Vcc	Output	GPIO	PCM_CLK

Figure 29: Raspberry Pi B Channelization

Many of the 5V and Ground connections are soldered together due to the limited GPIO pins and physical space.

4.3 Bill of Materials

Using the proposed designs for the system and the results of the various trades conducted, shown in the *Proposed Solution* section of the report, necessary components were purchased and used. After some modifications and testing, Fig. 42 shows the Bill of Materials for the final design.

Table 42: Bill of Materials

Supplies	Quantity	Model #
Raspberry Pi	2	4 B (2 GB)
Pi MicroSD (2-pack)	1	Class 10 32 GB
Photoresistor	30	EBOOT-RESISTOR-05
Active Buzzer	2	active buzzer 5V
OLED Display	1	i2c
HC-SR04 Distance Sensor (5pack)	1	
Motor	1	Flipsky 6354 190KV 2450W BLDC
ESC	1	FSESC 4.12 50A
Battery	1	6S 5000mAh 22.2v 50C Lipo Battery
Battery Charger	1	LiPo/Li-ion 3.7V-22.2V（1-6 cell）
Wire	1	12 AWG
Connectors and Heat Shrink	1	4.0mm Gold Bullet Banana Connector Plug 4mm Male
EC5 Connector	1	
M4 Bolts	1	
PiJuice HAT	2	Model #: BP7X 1820mAh / 6.7Wh
Non-slip Tape	1	
Pressure Sensitive Resistors	2	
Motor Mount	1	
Sensor Mount	1	
Bike	1	
Helmet	1	
Bike Trainer	1	

Components like the bike and the bike trainer are not a part of the completed system, but were necessary to test, design, and demonstrate the project. Small components that were readily available, such as resistors and wires, were not included.

4.4 Mechanical Drawings

The mechanical drawing of the Flipsky Brushless DC Motor is shown in Fig. 30. This mechanical drawing was used to design a mount to secure the motor onto the frame of the bicycle. Furthermore, the motor mount is comprised of three distinct pieces. Two of the three pieces are shown in Fig. 31. These objects fit around of the frame of a bicycle, and are adjustable using M4 bolts. For different sized bike frames, the bolts can be tightened or loosened appropriately. The flat portion of the object on the upper right side of Fig. 31 is attached to the object shown in Fig. 32 using four more M4 bolts.

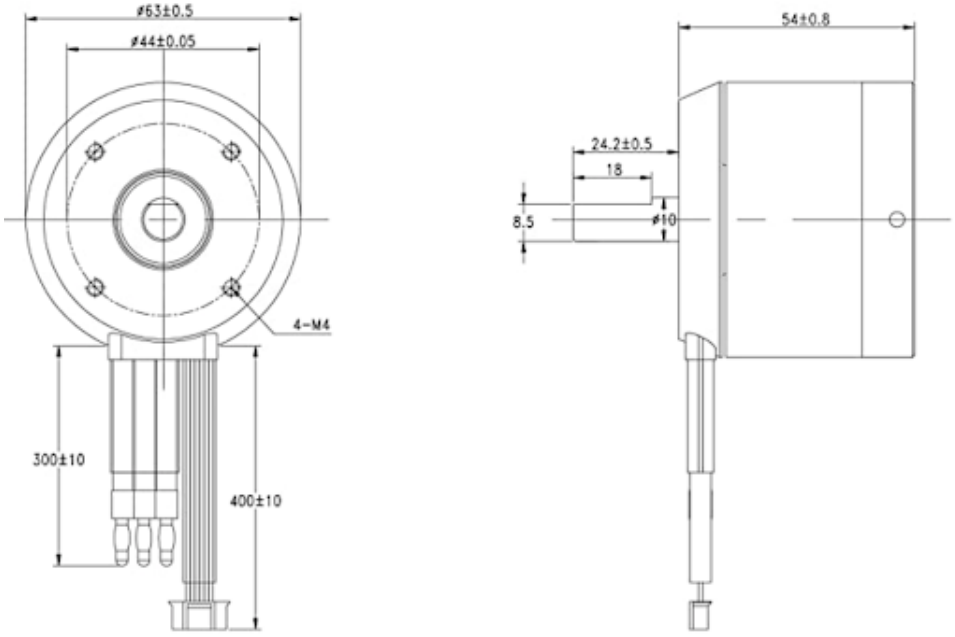


Figure 30: Motor Mechanical Drawing [7]

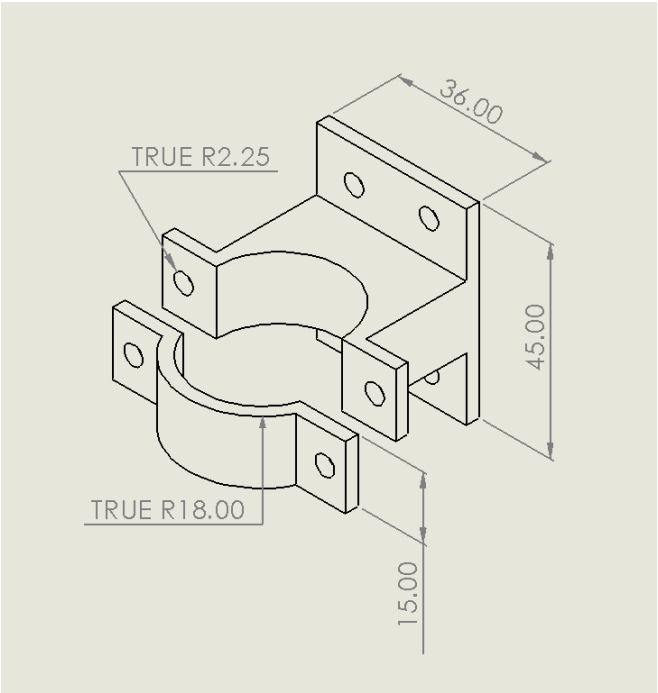


Figure 31: Motor Mount Frame Mechanical Drawing

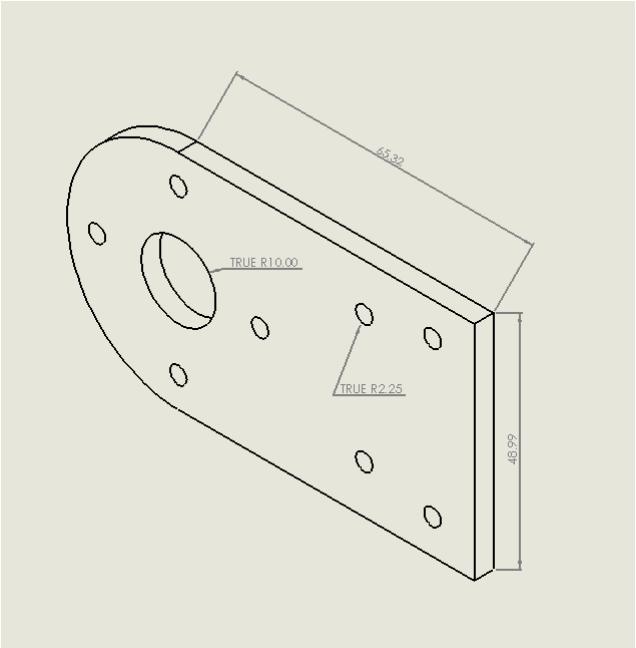


Figure 32: Motor Mount Mechanical Drawing

The preliminary design of the system is shown in Fig. 33. The motor mount that has been implement differs slightly from that shown in Fig. 33, but the placement of the motor in relation to the bicycle tire is unchanged.

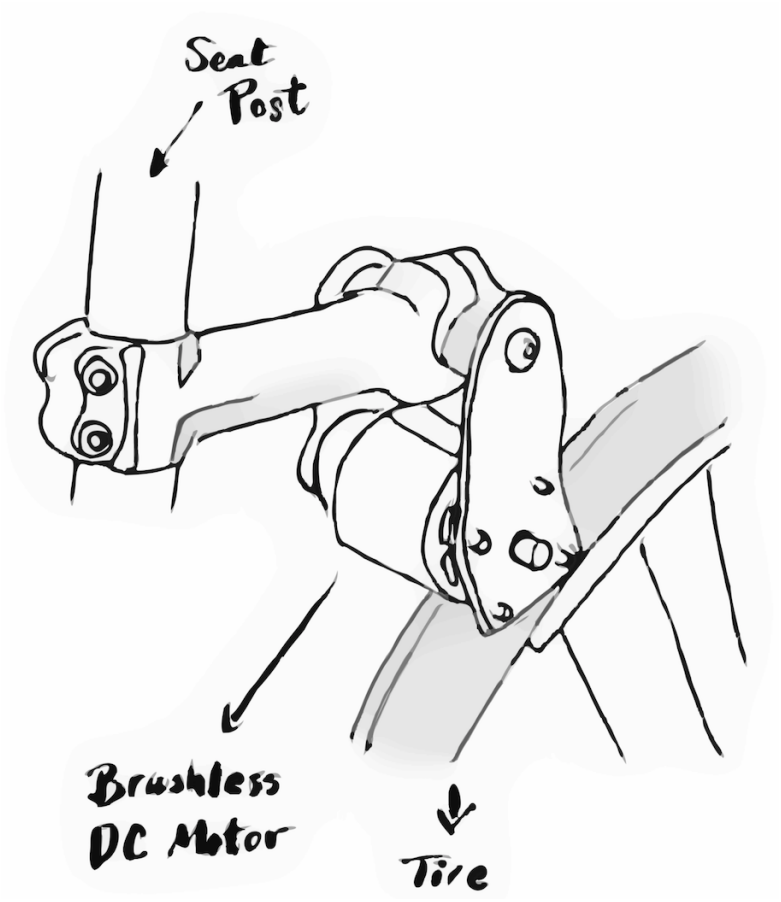


Figure 33: Motor Placement

The mechanical drawing of the sensor mount is shown in Fig. 34. In order for the mount to fit a variety of bicycle frames, there is a small gap in the circular portion of the mount, which allows for some flexibility. The circular part of the mount fits around the seat post of a bicycle, and the sensors are attached to the flat surface.

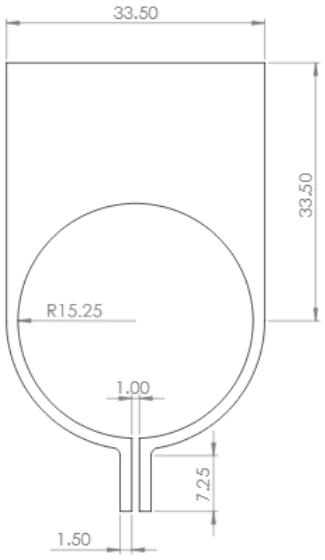


Figure 34: Sensor Mount Mechanical Drawing

4.5 System Design

The complete layout of the system's components is shown in Fig. 35 and Fig. 36. The motor and rear sensors are attached to the bicycle using the manufactured mounts shown in Figures 31, 32, and 34.Most of the components are wired to RPiA. On the helmet, the electronics are wired to RPiB. Both Raspberry Pi's are connected to PiJuice HATs via the GPIO pins.

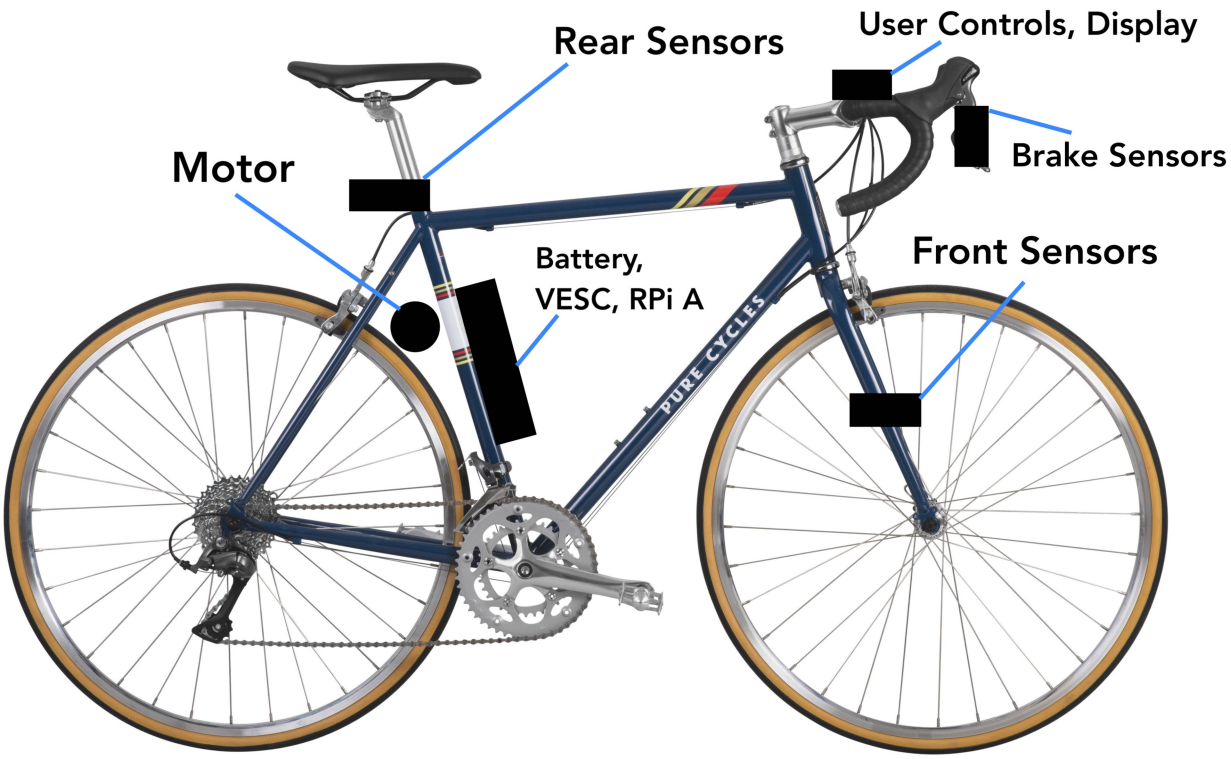


Figure 35: Bike Design Overview

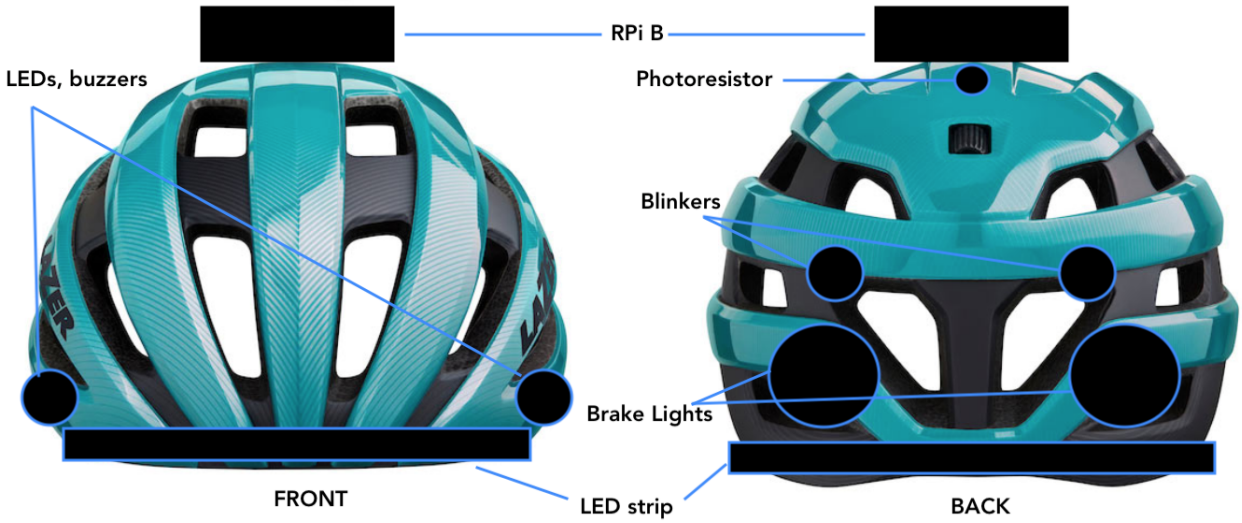


Figure 36: Helmet Design Overview

Five HC-SR04 ultrasonic sensor modules are used for obstacle detection. They are placed in the configuration shown in Fig. 37

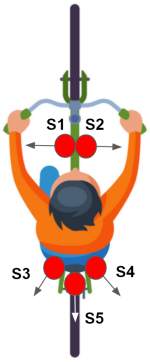


Figure 37: Distance Sensor Placement

The front sensors, S1 and S2, are used to set the ”Very Close” distance, to inform the back sensors. If anything is detected within the ”Very Close” distance or moving closer to the user within the ”Moderately Close” distance and it determined to be hazardous, the warnings are activated. The buzzer, LEDs, and OLED Display are all triggered to alert the user to the potential obstacle. The obstacle detection code is executed in Core 1 of RPiA; the flowchart is shown in Fig. 38.

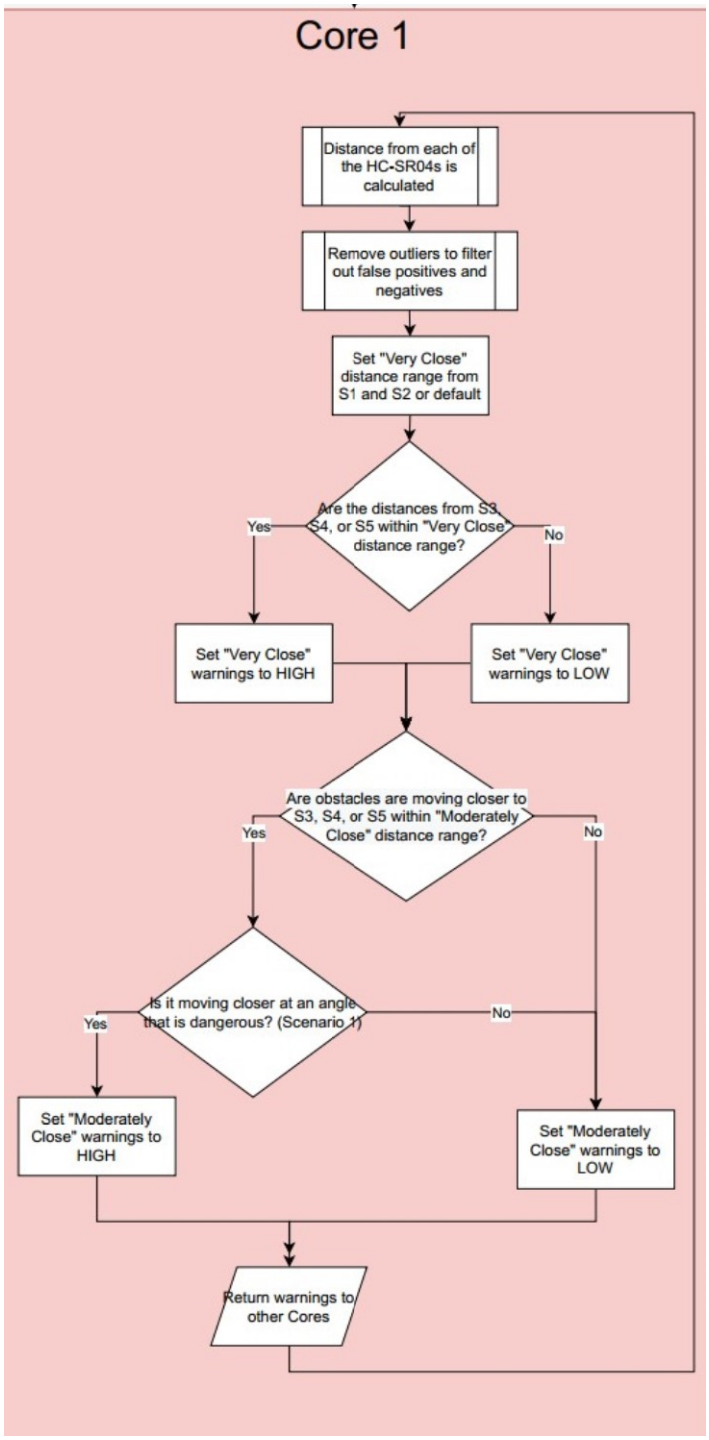
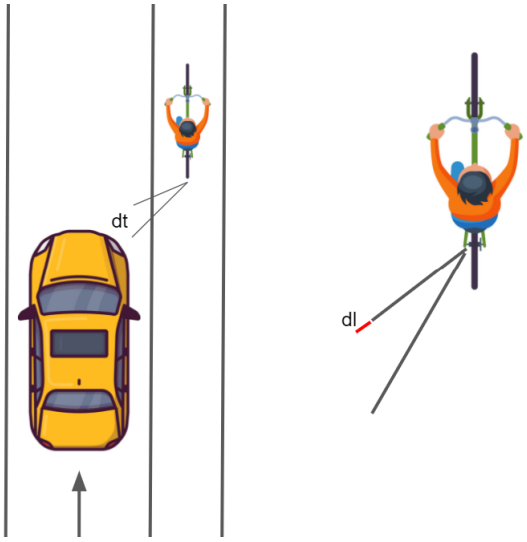


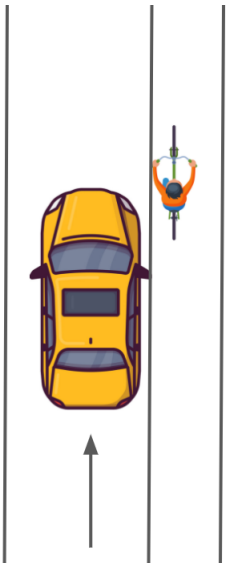
Figure 38: Obstacle Detection Flowchart

Figures 39-47 show the obstacle detection scenarios that the system is designed for. Under each scenario photo, there is a description of the scenario, the warning status, and the justification for the design decisions. Although these represent a finite number of possible instances, they are representative of some of the most common possibilities for cyclists. The project could be further improved by taking into account other more specific scenarios.



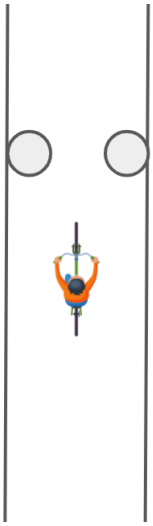
Description of Scenario	Obstacle passes bike from within the “Moderately Close” distance and remains parallel to the bike.
Will the user be alerted?	No
Justification	Even though the obstacle is technically getting closer to the S3 and S5 sensors, since it is moving in parallel and not getting closer perpendicularly, it is not considered dangerous.

Figure 39: Obstacle Detection Scenario 1



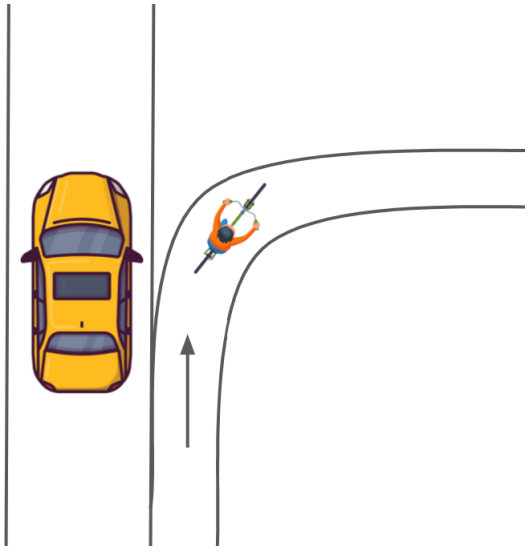
Description of Scenario	Obstacle passes bike from within “Very Close” distance.
Will the user be alerted?	Yes
Justification	Any obstacle, moving or non-moving, within the “Very Close” distance is considered dangerous to the rider.

Figure 40: Obstacle Detection Scenario 2



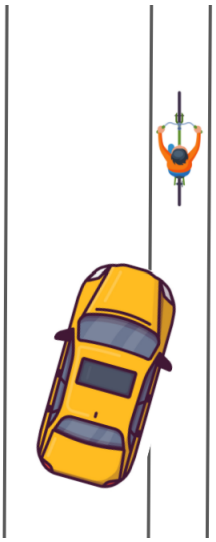
Description of Scenario	Bike goes between two still obstacles from within “Moderately Close” or “Very Close” distance
Will the user be alerted?	No
Justification	The rider can clearly see the obstacles they are riding between, and would be distracting to alert the rider to this non-threat.

Figure 41: Obstacle Detection Scenario 3



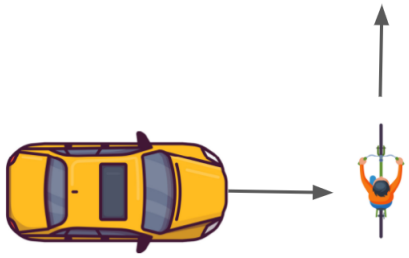
Description of Scenario	Bike is riding next to obstacle and turns away, so rear of bicycle is facing the obstacle
Will the user be alerted?	No
Justification	Even though the S5 sensor is getting closer to the obstacle, no threat is posed by the turn, because the S5 sensor is not closer to the obstacle than the S1 or S3 sensors were.

Figure 42: Obstacle Detection Scenario 4



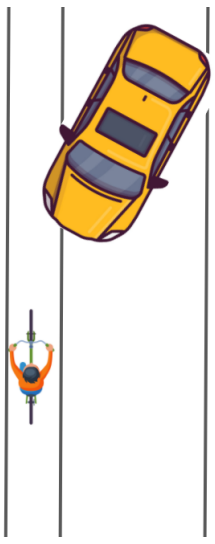
Description of Scenario	Obstacle gets closer to bike from the rear, and enters into "Moderately Close" distance
Will the user be alerted?	Yes
Justification	The obstacle is moving closer to the bike within a dangerous distance, outside of the riders' field of vision.

Figure 43: Obstacle Detection Scenario 5



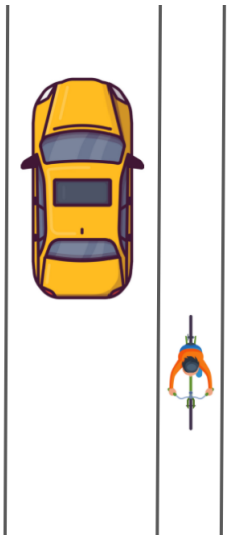
Description of Scenario	Obstacle gets closer to the side of the bike, and enters the "Moderately Close" distance.
Will the user be alerted?	Yes
Justification	The obstacle is moving towards the bike within a dangerous distance, potentially outside of the rider's field of vision.

Figure 44: Obstacle Detection Scenario 6



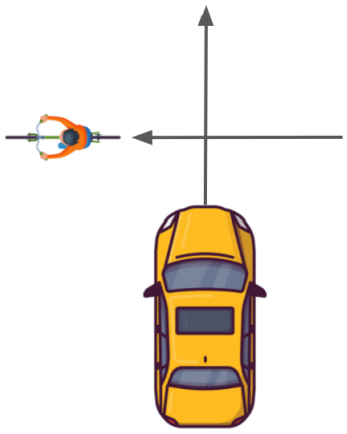
Description of Scenario	Obstacle gets closer to the bike from the front.
Will the user be alerted?	No
Justification	Even though the obstacle is on a trajectory to hit the bike, the obstacle is clearly within the rider's field of vision. Therefore, the alert would not be useful to the rider.

Figure 45: Obstacle Detection Scenario 7



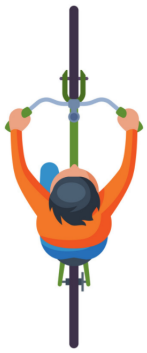
Description of Scenario	Obstacle passes bike (parallel) from the opposite direction as bike is moving within "Moderately Close" distance
Will the user be alerted?	No
Justification	Even though the obstacle is moving within the "Moderately Close" distance, the obstacle is not moving towards the bike, and thus does not pose an immediate threat.

Figure 46: Obstacle Detection Scenario 8



Description of Scenario	Obstacle passes perpendicular to the rear of the bike and passes within “Very Close” distance
Will the user be alerted?	Yes
Justification	Even though the bike may have successfully passed the obstacle because the distance is so close, it still poses a potential risk to the rider, and they should be aware of the obstacle’s presence.

Figure 47: Obstacle Detection Scenario 9



Description of Scenario	No obstacles are within the “Moderately Close” or “Very Close” distances
Will the user be alerted?	No
Justification	There are no potential obstacles to alert the rider to.

Figure 48: Obstacle Detection Scenario 10

The results of these scenarios are summarized in Fig. 43. The questions asked represent the major calculations in the algorithm. Each scenario’s response to these questions determine whether a warning is displayed to the user.

Table 43: Scenario Summary

	Scenarios								
	1	2	3	4	5	6	7	8	9
Is "Very Close" Distance is set by S1 and S2 smaller than the default?	No	No	Yes	Yes	No	Maybe	Maybe	No	Yes
Are distances from S3, S4, and S5 within the "Very Close" range?	No	Yes	No	No	No	Maybe	No	No	Yes
Are distances from S3, S4, and S5 moving closer within "Moderately Close" range as compared to previous measurements?	Yes	Yes	No	Maybe*	Yes	Yes	No	Maybe**	No
If distances are deemed to be moving closer, is the obstacle moving parallel the bike?	Yes	Yes	N/A	No	No	No	N/A	Yes	N/A
Is there a warning displayed to the user?	No	Yes	No	No	Yes	Yes	No	No	Yes

*Blinkers will deactivate warnings for S5 to decrease false alarm rate on turns
**Depending on the speed of the user and the obstacle

While obstacle detection is calculated in Core 1 of RPiA, in Core 2, code is executed to exchange Bluetooth communication with RPiB. This is implemented using Python's multiprocessing library. Bluetooth signals are sent to RPiB to exchange information about obstacle detection warnings and blinkers. Bluetooth information is received about the battery status of RPiB. The flowchart for this algorithm is shown in Fig. 49.

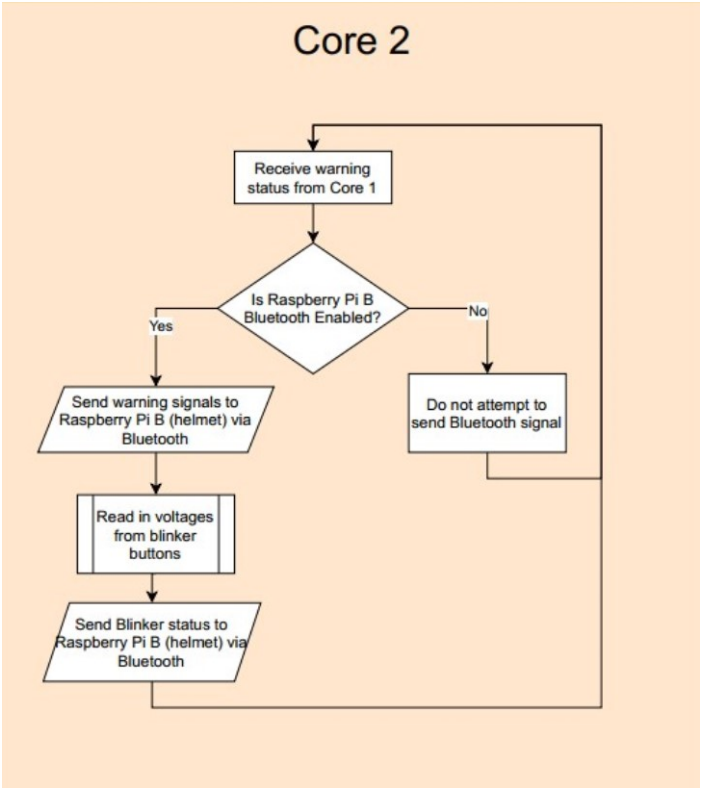


Figure 49: Core 2 Flowchart

In Core 3 of RPiA, code is executed to control the OLED Display. The OLED uses I2C communication to display images or text on the screen using the SDA and SCL pins. The screen

is located on the user controls panel, on the handlebars of the bike. A warning message is displayed if obstacles are detected. The speed of the bike and battery life can also be displayed. The flowchart for Core 3 is shown in Fig. 50.

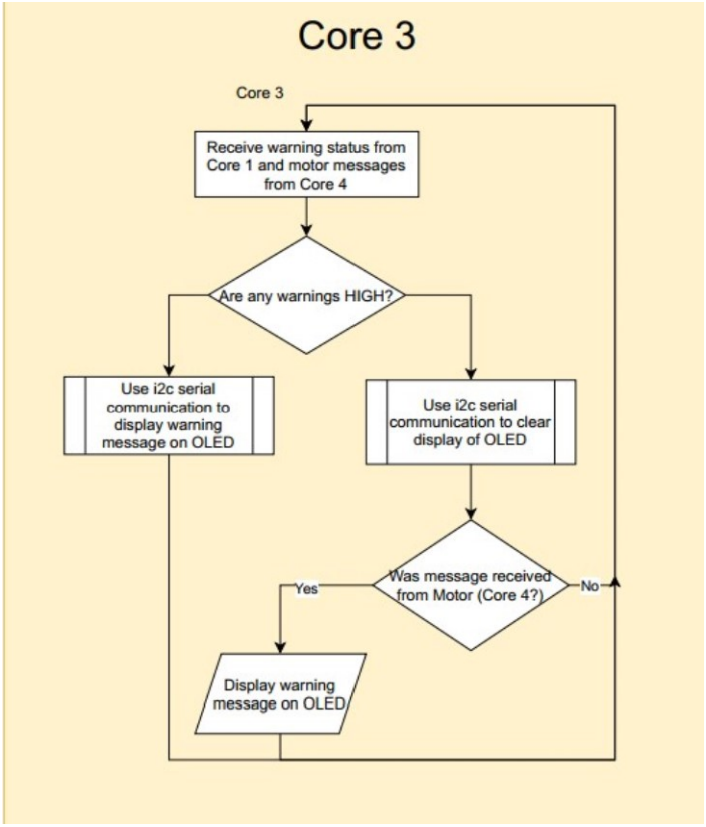


Figure 50: Core 3 Flowchart

In Core 4, the code for the motor control is executed. The Raspberry Pi interfaces with the VESC using UART communication. The VESC then powers the motor with the updated duty cycle. The user can control the speed using increment and decrement speed buttons, and can stop the motor using the brakes. The code for Core 4 is shown in Fig. 51.

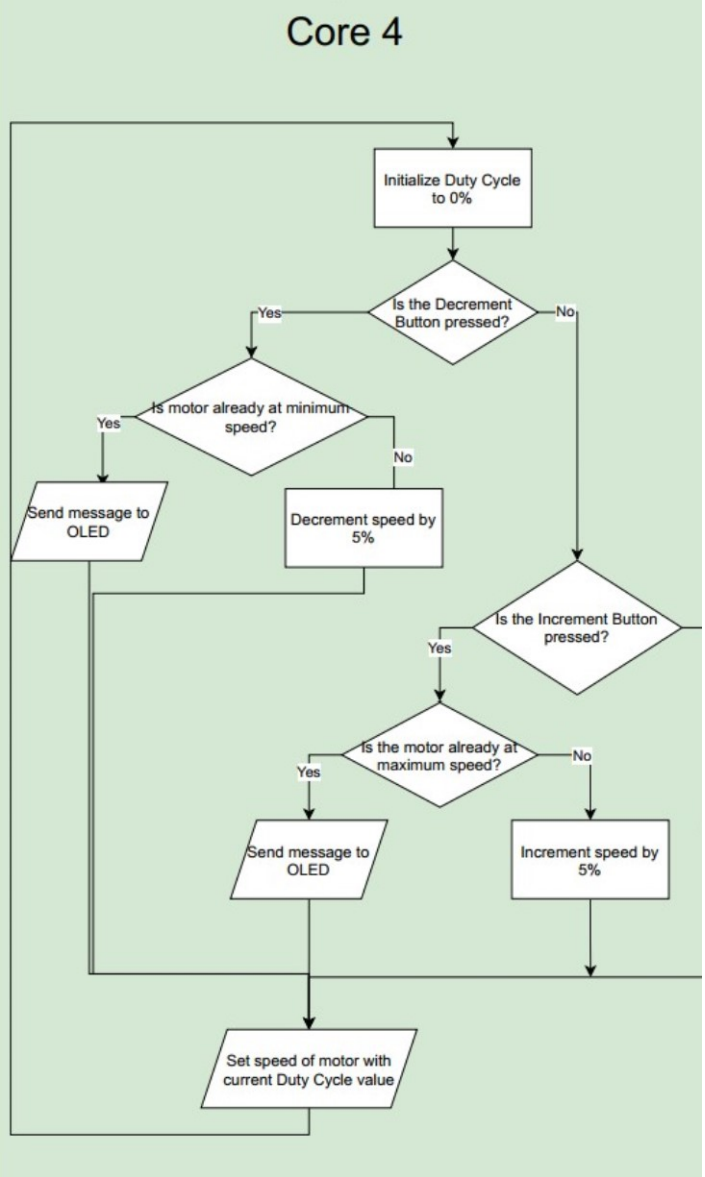


Figure 51: Core 4 Flowchart

Raspberry Pi B, on the helmet, receives Bluetooth communication and activates the warnings on the helmet. This is all executed in one core, without multiprocessing. Battery status updates on the PiJuice HAT are also transmitted back to RPiA. This code is illustrated in the flowchart in

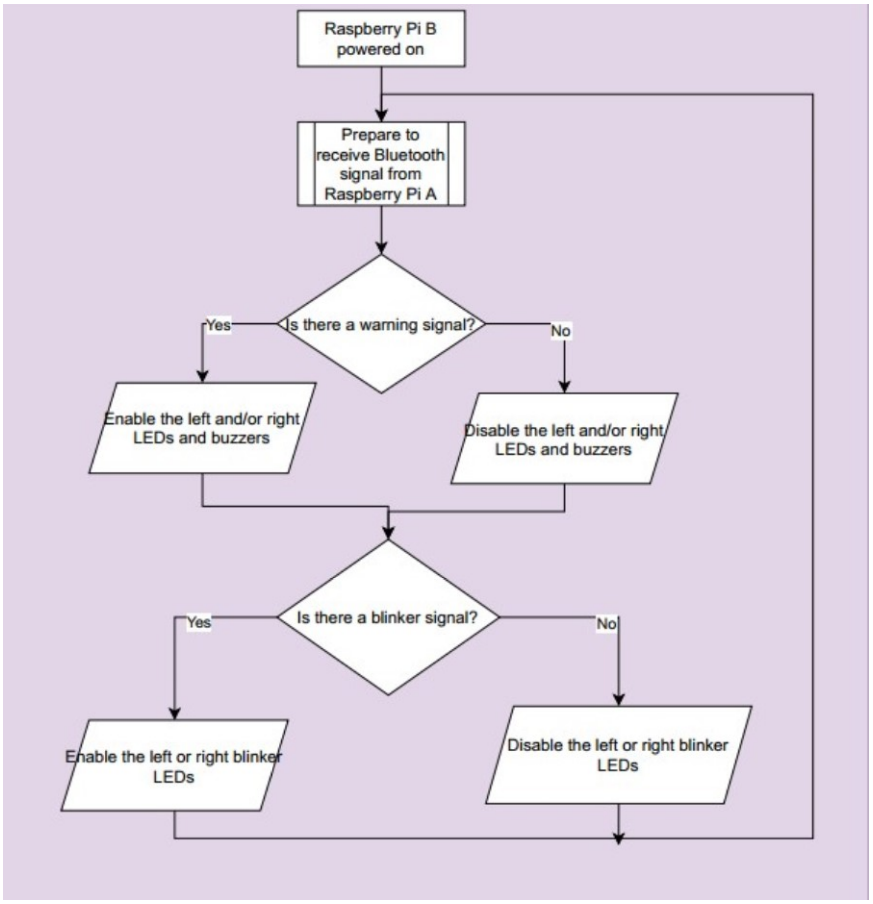
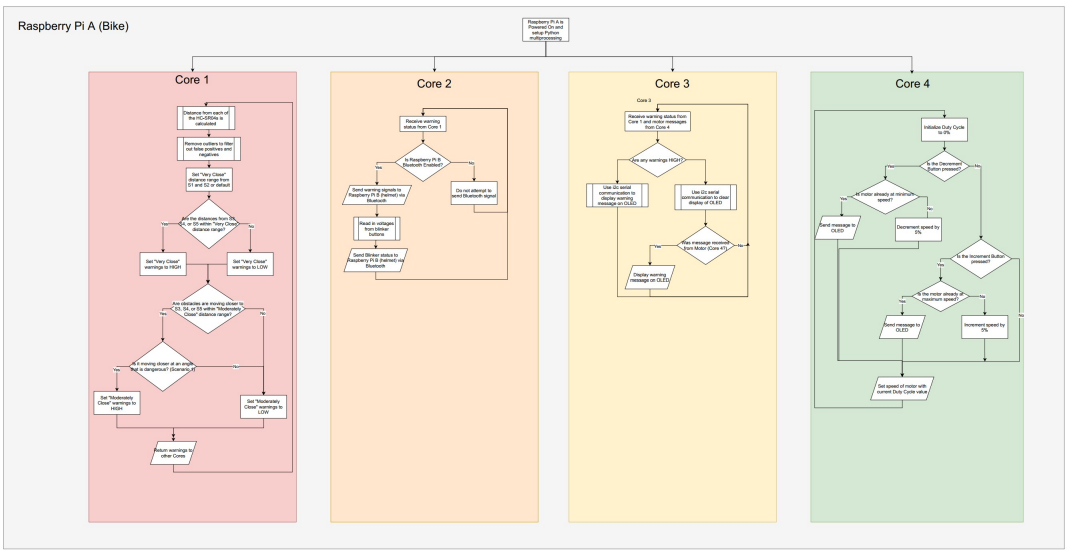


Figure 52: RPiB Flowchart

The comprehensive flowchart, including the Raspberry Pi software of the whole system, is shown in Fig. 53. Multiprocessing was used to decrease latency. This is important to speedily alert users to potential obstacles and change the speed of the motor quickly.



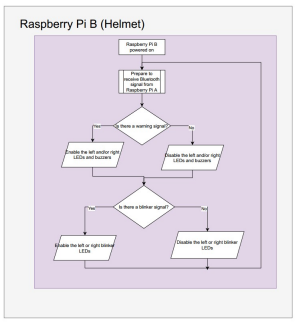


Figure 53: Comprehensive Software Flowchart

A bicycle trainer is a piece of equipment that makes it possible to ride a bicycle while remaining

stationary. The axle of the rear wheel is secured between two adjustable clamps, and a smaller wheel applies pressure to the tire to mimic the resistance felt when riding a bicycle on the road. A picture of the bike trainer setup is shown in Figure 54.

Figure 54: Bike Trainer Setup

To test the motor/battery system, the motor control program was executed using the Raspberry Pi attached to the bicycle frame. When the Raspberry Pi was running said program, the user controls mounted near the handlebars were tested. First, the brakes were pressed to confirm the motor gradually slowed in response. Secondly, the increment speed and decrement speed buttons were pressed one at a time to test if the motor sped up and slowed down accordingly. The user controls were tested multiple times to verify they perform as intended.

The scenario summary shown in Fig. 43 was used as a guide to test the obstacle detection system. For each scenario listed, the situation was recreated and the false positive rate was determined and recorded. The false positive and accuracy rates from each scenario were compared against the requirement to determine the success of the design.

The run time of the program was also assessed to ensure the code was executing fast enough to be effective and safe for the user. Finally, for each scenario, the performance of the auxiliary aspects of the obstacle detection/warning system (i.e. blinkers, brake lights) was evaluated.

After these tests were performed on the stationary system using the trainer, the system was testing with a rider. Obstacles were placed to test the ”real world” accuracy of the system. Additionally, the rider tested the speed and brake buttons, and code was adjusted for a smoother transition from speed to speed. The results of these tests can be found in the section *Experimental Test and Demonstration*.

4.6 Cost Estimates

The proposed budget for the project was between $485-$565. The actual cost of the system, broken down in Fig. 44, was $510.68. This falls within the estimated range at the time of proposal.

Table 44: Cost Estimate

Supplies	Quantity	Cost ($) (ea)	Cost ($) (total)	Model #
Raspberry Pi	2	47	94	4 B (2 GB)
Pi MicroSD (2-pack)	1	8.98	8.98	Class 10 32 GB
Photoresistor	30	0.17	4.95	EBOOT-RESISTOR-05
Active Buzzer	2	0.95	1.9	active buzzer 5V
Mini Magnets	25	0.28	6.95	
Hall Effect Switch	2	0.58	1.16	DRV5013AGQLPG
OLED Display	1	6.95	6.95	i2c
HC-SR04 Distance Sensor (5pack	1	9.5	9.5	
Motor	1	82.99	82.99	Flipsky 6354 190KV 2450W BLDC
ESC	1	79.99	79.99	FSESC 4.12 50A
Battery	1	70.99	70.99	6S 5000mAh 22.2v 50C Lipo Battery
Battery Charger	1	40.69	40.69	LiPo/Li-ion 3.7V-22.2V（1-6 cell）
Wire	1	6.98	6.98	12 AWG
Connectors and Heat Shrink	1	7.99	7.99	4.0mm Gold Bullet Banana Connector Plug 4mm Male
EC5 Connector	1	8.99	8.99	
M4 Bolts	1	12.99	12.99	
PiJuice HAT	2	79.95	159.9	Model #: BP7X 1820mAh / 6.7Wh
Non-slip Tape	1	13.99	13.99	
Pressure Sensitive Resistors	2	12.59	25.18	
Bike	1	0	0	
Helmet	1	0	0	
Bike Trainer	1	0	0	
		Total Cost:	510.68	

If this product were to be produced commercially, the cost would fall well below $500. Because of the costs of experimentation and testing, some purchases would not be recurring for the production of each new kit. Additionally, ordering components in bulk would drive down the cost significantly. The product could be offered at a very competitive price, in comparison to some e-bike conversion kits and e-bikes presently on the market.

5 Experimental Test and Demonstration

5.1 Tests Used to Benchmark System Performance

System performance was evaluated based on the satisfaction of the system's Engineering requirements. The Engineering and Customer (Marketing) Requirements are listed again below.

Marketing Requirements	Engineering Requirements	Justification
1, 2, 5	1. The bicycle should be able to maintain a speed of 17 mph.	The specified speed will ensure the user can commute efficiently without compromising safety.
3	2. Production cost should not exceed $600.	This is based upon competitive benchmarking and exisiting technologies.
1	3. The sensor system should be at least 90% accurate in detedcting and alerting to potential obstacles.	Too many false positive readings from sensors would make the system more dangerous.
2, 4	4. The complete system should not contain more than 5 pieces that the user needs to install.	This can be done by combined components together to form 5 or less discrete modules
2, 5	5. The battery life should last for at least 30 miles of city travel.	This is based upon competitive benchmarking, as well as the usability and sustainability of the design.
1, 2, 4, 5	6. The weight of the kit components should not exceed 15lb.	Added weight to the system will decrease efficiency and maximum speed.

Marketing Requriements

1. The system should be safe and promote additional safety for its users.
2. The system should be environmentally sustainable.
3. The system should have low cost as compared to similar products.
4. The system should be easy-to-install for users without an advanced knowledge of bicycle mechanics.
5. The system should allow for faster commute times than travel on a normal bicycle.

Figure 55: Engineering Requirements

In order to satisfy Engineering Requirement 1, the speed of the bike was measured using the bike trainer. The motor controller was programmed to have a maximum duty cycle of 50%, although the rider can exceed this speed with their own force. The system was also tested with a rider to ensure the bike could still maintain the required speed. The results of these tests are shown in Figures 62 in *Working Prototype* and Figure 76 in *Data Analytics*.

Engineering Requirement 2 is satisfied, as shown in the Cost Estimate, shown in Fig. 44. The cost estimate for the project was well below $600, which indicates the production cost of a single kit would also be lower.

The accuracy of the obstacle detection system, as indicated in Engineering Requirement 3, was tested by recreating the scenarios shown in Figures 39 - 47. To test obstacle detection, the bike was run in the trainer and obstacles were moved around it. For the necessary scenarios, the bike was also removed from the trainer and walked or ridden to simulate the scenario. The accuracy rate was determined for each scenario, which indications of whether it represents a false positive or false negative rate. The results of these tests are shown in Figures 46-55.

Engineering Requirement 4 was satisfied, and the five pieces included in the conversion kit are shown in Fig. 64. The two ultrasonic sensors mounted on the front fork of bicycle are not circled because they are attached they are are attached to the same cable of wires that is connects the back sensors to the Raspberry Pi, Therefore, all the sensors are connected (physically, not electrically) and can be considered as one item.

To satisfy Engineering Requirement 5, tests were performed on the time of the battery life. The test was considered a success if the battery was on for a duration of what converted to 30 miles of travel. These tests are shown in Table 61. Lastly, Engineering Requirement 6 was satisfied. The final weight of the conversion kit was approximately eight pounds.

5.2 Working Prototype

A prototype of the design was created and tested. The conversion kit was installed on a bicycle and the necessary components were added to the helmet, as shown in Fig.56.

Figure 56: Bike and Helmet Prototype

Fig. 57 shows a side view of the bike. the motor, battery and VESC case, distance sensors, and User Controls are visible.

Figure 57: Bike Prototype Sideview

User Controls were installed to the handlebars. This module includes the OLED Display, turn signal buttons, and increment and decrement speed buttons. These components were all soldered onto a board and a wooden frame was mounted on top.

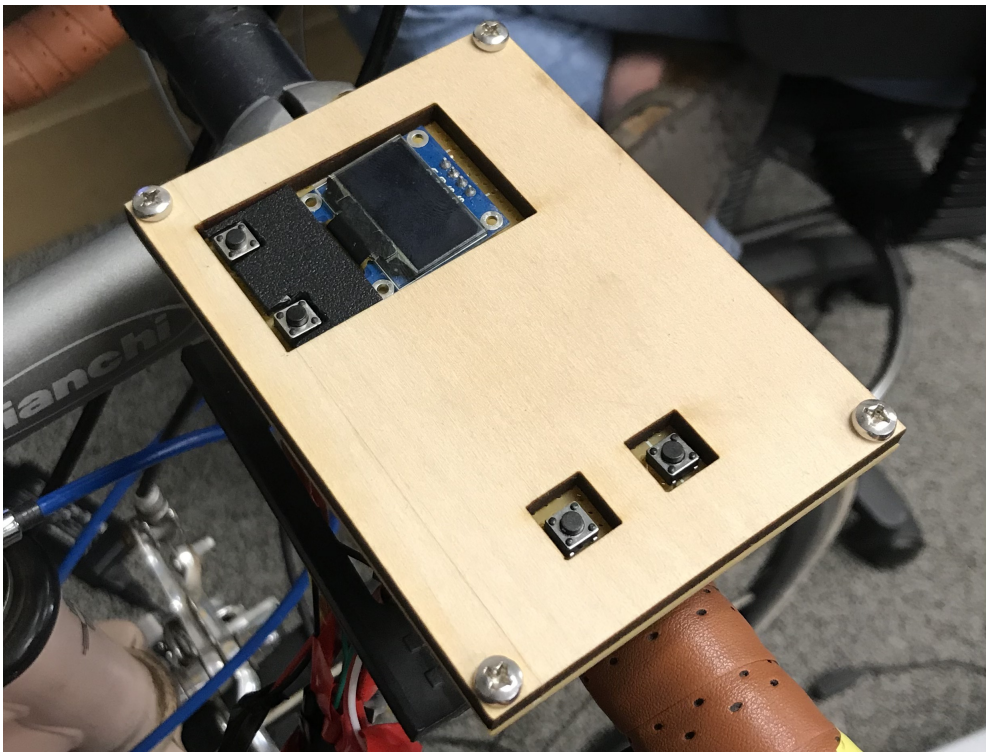


Figure 58: Bike Prototype User Controls

In Fig. 59, the User Controls and brakes are visible. Pressure resistors are added to the hand brakes in order to provide braking information to the VESC.

Figure 59: Bike Prototype Front View

The blinker lights, brake lights, warning lights and buzzers, LED strip and Circuit, and RPiB were attached to the helmet, as shown in Fig. 60. These components receive information via Bluetooth from the system on the bicycle.

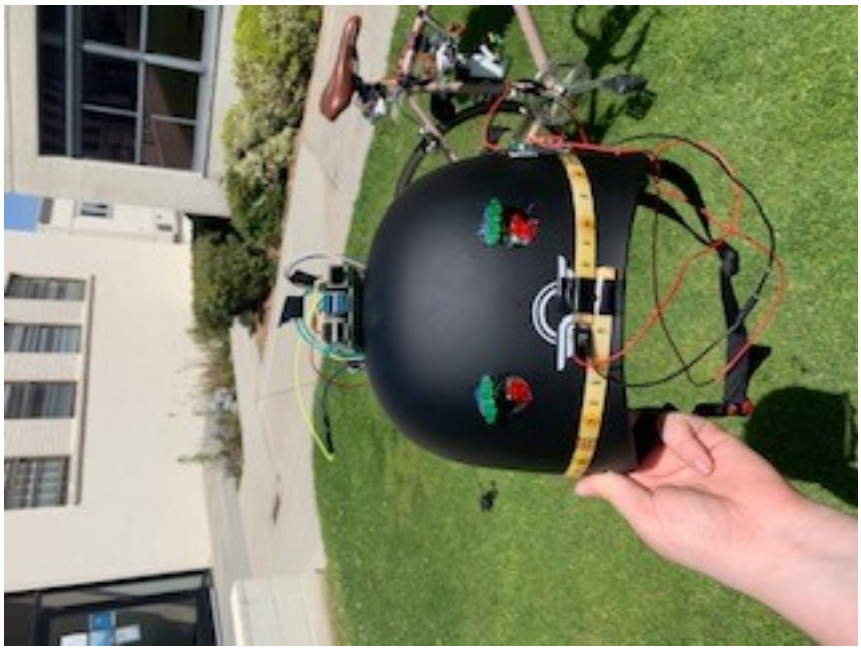


Figure 60: Helmet Prototype Front View

The system on the bicycle is shown again in Fig. 61.

Figure 61: Bike Prototype

The OLED screen displays information about obstacle warnings, speed, and connections. Fig. 62 shows the screen displaying speed information, at the maximum speed of 17 miles per hour. Warnings are prioritized to be displayed when obstacles are present. Then speed information is displayed when the user changes the speed using the buttons or brakes. Also, if the Bluetooth is disconnected, the user is notified on the screen, as well. Fig. 63 shows the OLED screen displaying an obstacle detection warning.

Figure 62: Speed Reaches 17 MPH

Figure 63: Warning on OLED Display

Fig. 64 shows the distinct components installed to the bike as part of the conversion kit, as per Engineering Requirement 4.

Figure 64: Pieces to Install

The PiJuice HATs were configured and programmed to execute certain code at the press of buttons and levels of battery life. Fig. 65 shows the general HAT setting configuration.

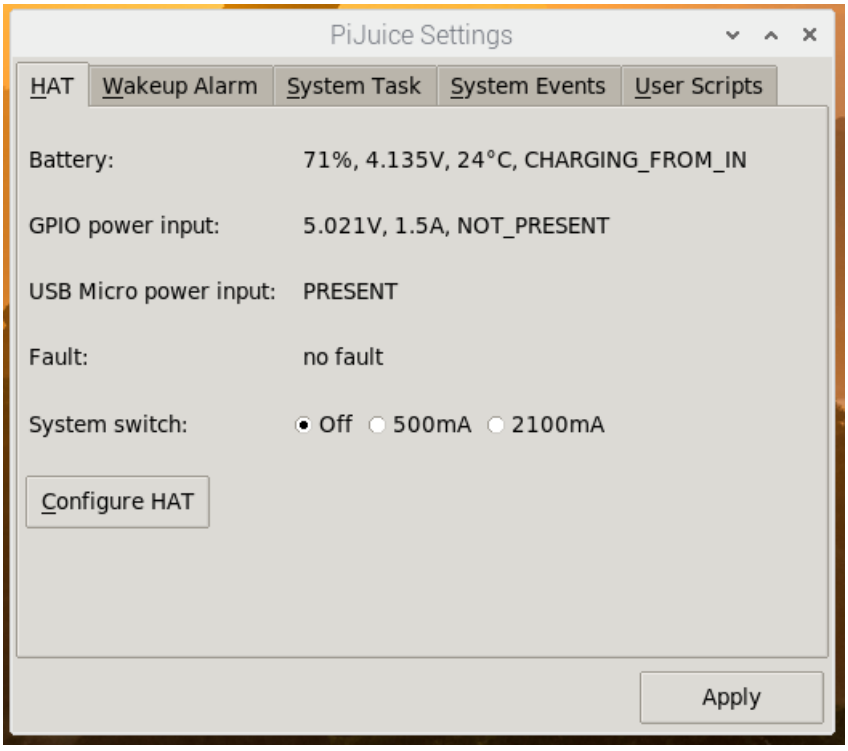


Figure 65: PiJuice General Settings

Fig. 66 shows the configuration settings for the PiJuice module to interact with the battery.

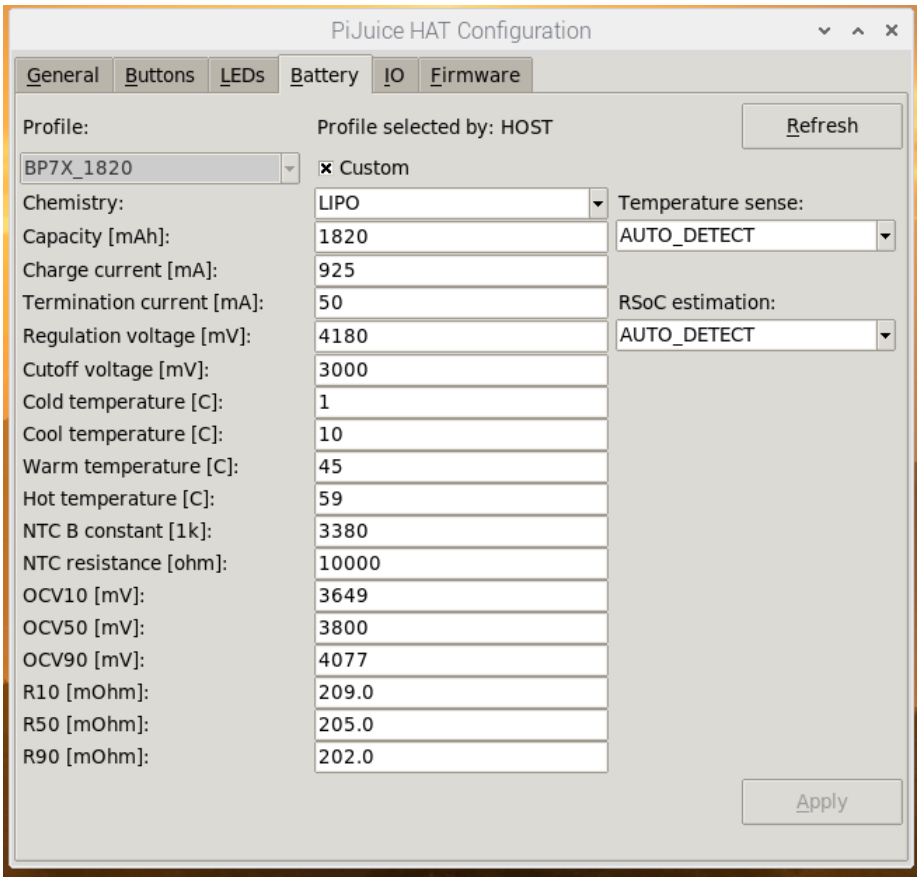


Figure 66: PiJuice Hat Battery Settings

Fig. 67 and Fig. 68 show the configuration for the Hat to execute the specified user functions in response to the buttons.

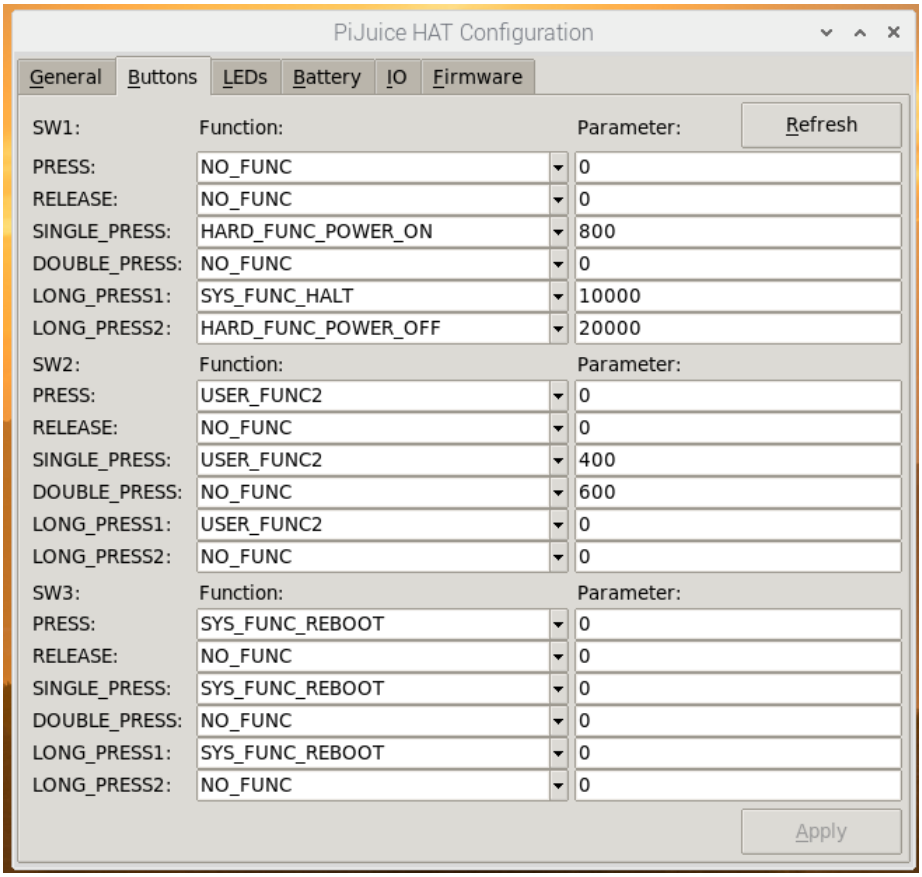


Figure 67: PiJuice Hat Button Configuration Settings

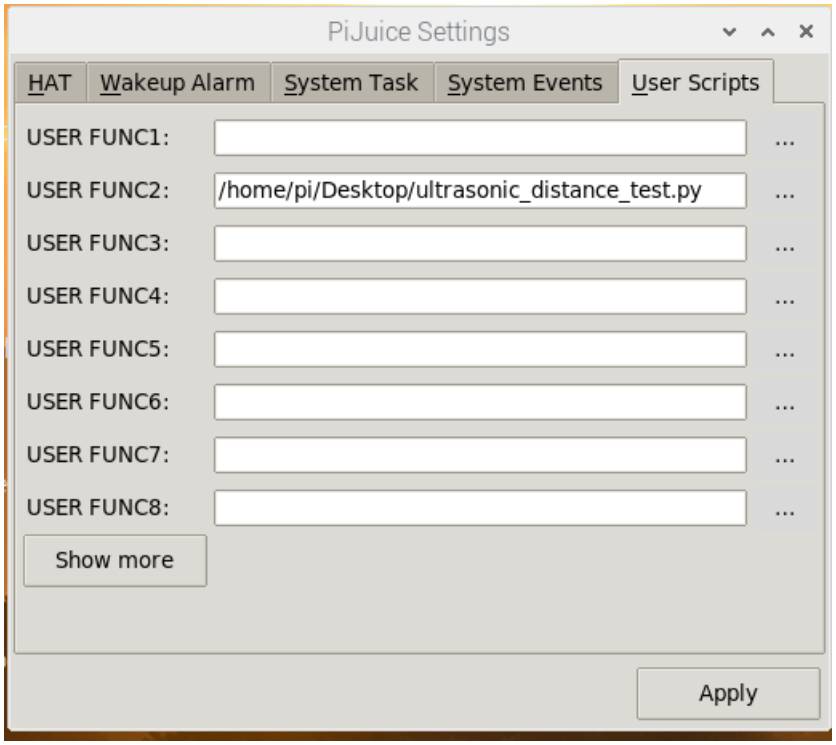


Figure 68: PiJuice User Scripts Settings

The codes executed on RPiA, ”bike.py”, and on RPiB, ”helmet.py”, are included as appendices. Fig. 69 shows output to the terminal of the distance measurements and calculations in a single iteration. Raw distances are first collected from each of the five ultrasonic sensor modules. Then, the data is filtered to remove outliers. The filtered data is then analyzed to determine if warning should be activated.

Figure 69: Distance Measurements Screenshot

The signals received via Bluetooth on RPiB are shown in Fig. 70.

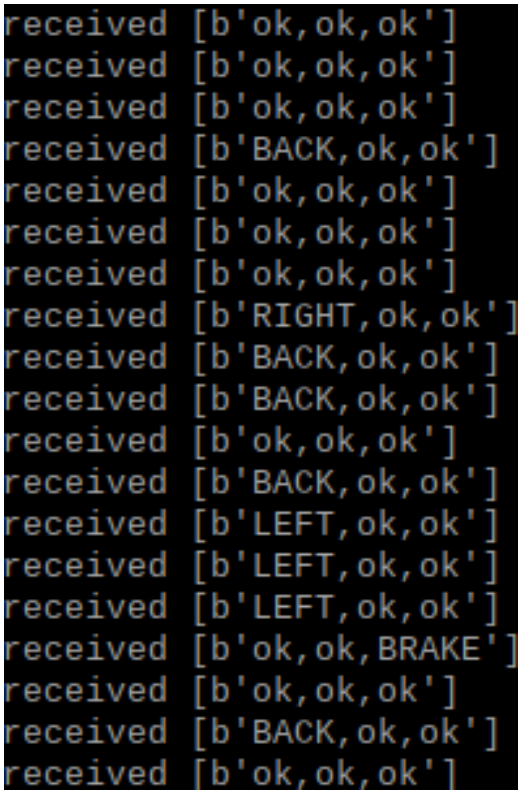


Figure 70: Received Bluetooth Signal Screenshot

Fig. 71 shows the terminal output of some relevant motor information, when the Increment and Decrement buttons are pressed, and when the brakes are activated. The updated duty cycle's are printed after each button press.

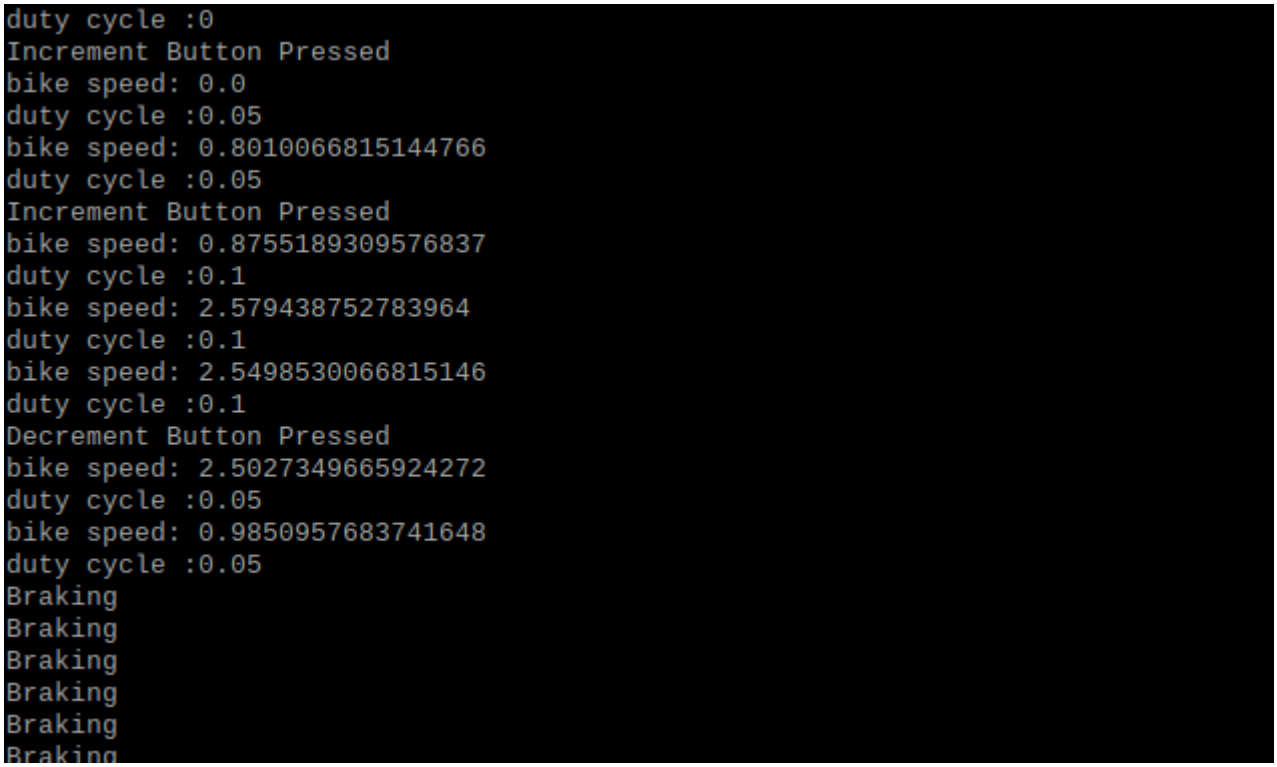


Figure 71: Motor Screenshot

The execution time for each iteration of the code was tested. The screenshot of the measured run times are shown in Fig. 72. The first iteration is more time consuming, as it takes time to configure settings for the motor and OLED.

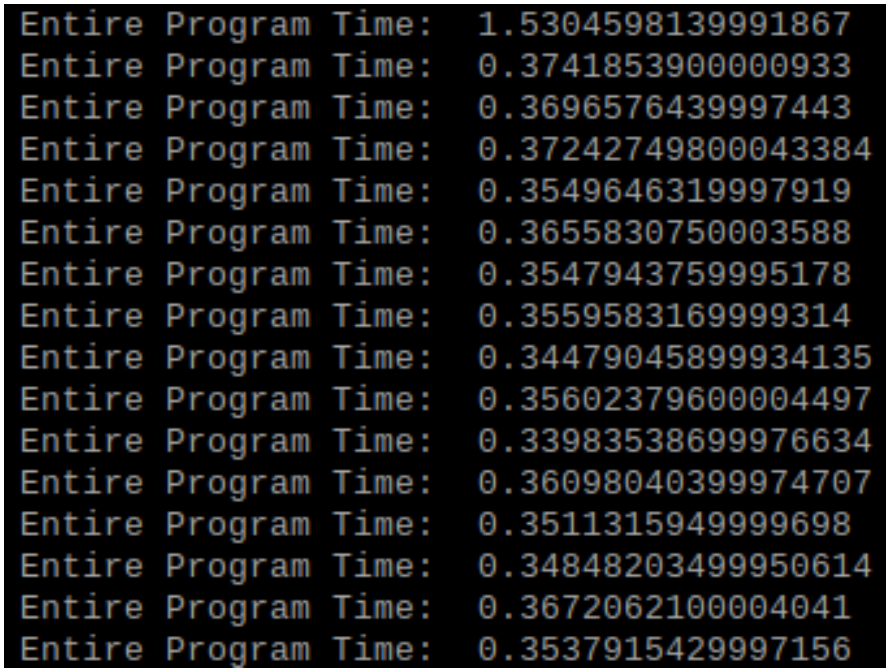


Figure 72: Execution Time Screenshot

For further explanation of the code, the full flowchart of the algorithm is shown in Fig. 53 and the commented code is included in the appendices.

5.3 Demonstration

The project's success is demonstrated through the tests shown in *Data Analytics*. Additionally, a video showing the system and its features in operation is included in the follow link and QR code in Fig. 73. A screenshot of the video is shown in Fig. 74

`www.youtube.com/watch?v=R1YVmbAiM0w&ab_channel=MarenaT`

Figure 73: Video Demonstration

Figure 74: Video Screenshot

5.4 Meeting Customer Requirements

The customer requirements are listed again here.

1. The system should be safe and promote additional safety for its users.
2. The system should be environmentally sustainable.
3. The system should have low cost, as compared to similar products.
4. The system should be easy-to-install for users without an advanced knowledge of bicycle mechanics.
5. The system should allow for faster commute times than travel on a normal bicycle.

Customer Requirement 1 was satisfied because the sensor system alerts the rider to potential obstacles with a 91.5% average success rate. The performance of the sensor system is discussed further in *Data Analytics*. Also, the conversion kit utilizes a Li-po battery, which unlike other batteries, can be recycled.

The total cost of materials for the conversion kit was $510.68. If the conversion kit was to be priced for a 50% profit margin, then the retail price would be about $765, which is a more affordable option as compared to other conversion kits currently on the market [10]. Additionally, to attach the conversion kit onto the bicycle, no speciality bike equipment is needed, and it is not necessary to remove items from the bike, i.e. a wheel, handlebars, etc.

The conversion kit allows commuters to move at a fast past for longer distances than might be possible otherwise. As mentioned previously, the conversion kit is designed maintain a speed of up to about 17mph.

5.5 Data Analytics

The obstacle detection system utilizes HC-SR04 Ultrasonic Sensor Modules to obtain data to process distance measurements and determine potential hazards. The range of the sensors were determined experimentally.To estimate the range of the HC-SR04 module, measurements were take over a span of 100°, 50° to the left and right of center. The experimental results are shown in Fig. 75 and Table 45 [2].

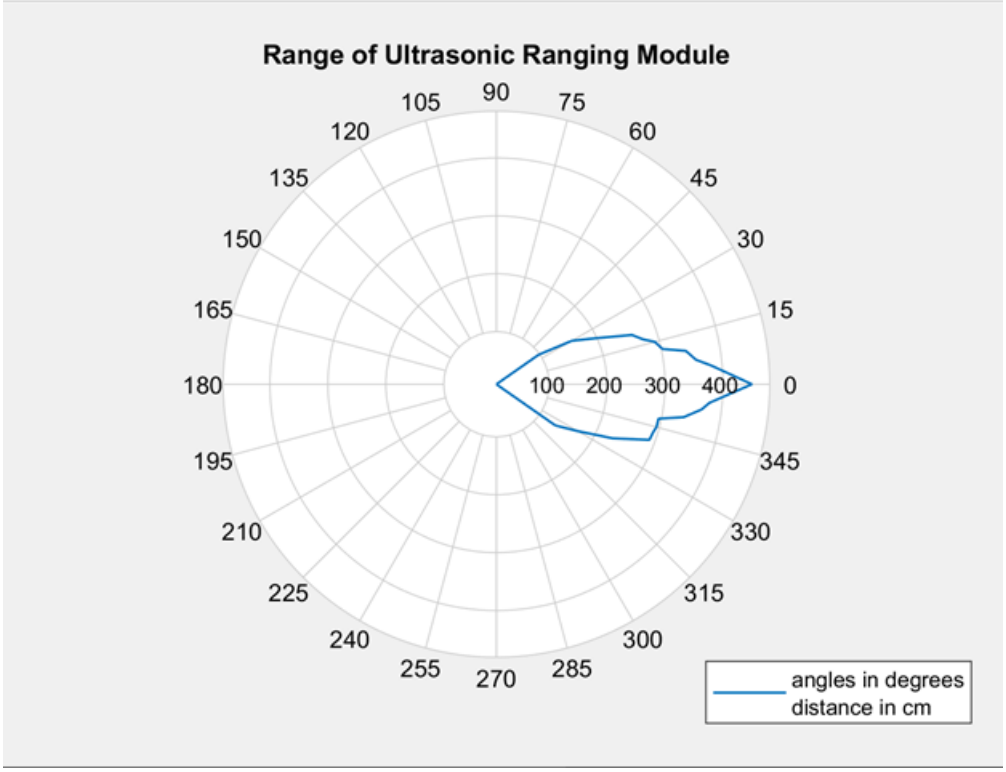


Figure 75: Range of HC-SR04 Ultrasonic Ranging Modules

Table 45: HC-SR04 Experimental Range Data

Angle (degrees)	Distance (cm)
-50	0
-45	0
-40	10
-35	133
-30	171
-25	230
-20	290
-17	292
-15	295
-12	295
-10	337
-7	366
-5	378
0	450
5	381
7	356
10	341
12	302
15	292
17	274
20	258
25	197
30	159
35	98
40	0
45	0
50	0

The range needed for the system is 200 cm; therefore the experimental range satisfied this requirement.

Each of the Obstacle Detection scenarios was tested, as described in *Tests Used to Benchmark System Performance*. For each trial, an ”S” denotes a success and an ”F” denotes a failure. For Scenario 1, shown in Fig. 39, a warning indicates a failure and no warning indicates a success. Table 46.

Table 46: Obstacle Detection Scenario 1 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	F	S	F	F	S	S	S	F	F	S
Right (S4)	S	S	F	S	F	S	S	S	S	F

For Scenario 1, there was a 60% success rate and a 40% false positive rate for the twenty trials conducted.

For Scenario 2, described in Fig. 40, a success is described by the presence of a warning and a failure is described by no warnings. The test results for Scenario 2 are shown in Table 47.

Table 47: Obstacle Detection Scenario 2 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	S	S	S	S	S	S	S	S	S
Right (S4)	S	S	S	S	S	S	S	S	S	S

For Scenario 2, there was a 100% success rate and a 0% false negative rate for the twenty trials conducted.

For Scenario 3, described in Fig. 41, a success is described by no warning and a failure is described by a warning. The test results for Scenario 3 are shown in Table 48.
90% success rate, 10 % false positive

Table 48: Obstacle Detection Scenario 3 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	S	S	F	S	S	S	S	S	S
Right (S4)	S	S	S	S	S	F	S	S	S	S

For Scenario 3, there was a 90% success rate and a 10% false positive rate for the twenty trials conducted.

In Scenario 4, described in Fig. 42, no warnings indicate a success and warnings indicate a failure. The test results for Scenario 4 are shown in Table 49.

Table 49: Obstacle Detection Scenario 4 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	S	S	S	S	S	S	S	S	S
Right (S4)	S	S	S	S	S	S	S	S	S	S

For Scenario 4, there was a 100% success rate and a 0% false positive rate for the twenty trials conducted, with the inclusion of turn signal buttons to indicate to the algorithm when the rider is turning.

In Scenario 5, described in Fig. 43, a success occurs when warnings are activated and a failure occurs when warnings are not activated. The test results for Scenario 5 are shown in Table 50.

Table 50: Obstacle Detection Scenario 5 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	F	S	S	F	S	S	S	S	S
Right (S4)	S	S	S	S	S	S	S	S	S	S

For Scenario 5, there was a 90% success rate and a 10% false negative rate for the twenty trials conducted.

In Scenario 6, described in Fig. 44, a success occurs when a warning is present and a failure occurs when a warning is not present. The test results for Scenario 6 are shown in Table 51.

Table 51: Obstacle Detection Scenario 6 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	S	F	S	S	S	S	S	S	S
Right (S4)	S	S	S	S	S	S	S	S	F	S

For Scenario 6, there was a 90% success rate and a 10% false negative rate for the twenty trials conducted.

In Scenario 7, described in Fig. 45, a success is indicated by no warnings and a failure is indicated by a warning. The test results for Scenario 7 are shown in Table 52.
100% success rate, 0% false positive

Table 52: Obstacle Detection Scenario 7 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	S	S	S	S	S	S	S	S	S
Right (S4)	S	S	S	S	S	S	S	S	S	S

For Scenario 8, described in Fig. 46, a success is indicated by no warnings and a false is indicated by a warning. The test results for Scenario 10 are shown in Table 53.

Table 53: Obstacle Detection Scenario 8 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	S	S	S	S	F	S	S	F	S
Right (S4)	S	S	S	S	F	S	S	S	S	S

For Scenario 8, there was an 85% success rate and a 15% false positive rate for the twenty trails conducted.

For Scenario 9, described in Fig. 47, a success is described by a warning and a failure is described by no warning. The test results for Scenario 9 are shown in Table 54.

Table 54: Obstacle Detection Scenario 9 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left to Right	S	S	S	S	S	S	S	S	S	S
Right to Left	S	S	S	S	S	S	S	S	S	S

In Scenario 9, there was a 100 % success rate and a 0% false negative rate for the twenty trials conducted.

In Scenario 10, described in Fig. 48, a success occurs when there are no warnings and a failure occurs when there are warnings. The test results for Scenario 10 are shown in Table 55.

Table 55: Obstacle Detection Scenario 10 Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Left (S3)	S	S	S	S	S	S	S	S	S	S
Right (S4)	S	S	S	S	S	S	S	S	S	S

For Scenario 10, there was a 100% success rate and a 0% false positive rate for the twenty trials conducted.

Table 56 summarizes the results of the Obstacle Detection tests.

Table 56: Obstacle Detection Test Success Rates

Scenarios	Success Rate	Failure Rate	False Positive Rate	False Negative Rate
1	60	40	40	
2	100	0		0
3	90	10	10	
4	100	0	0	
5	90	10		10
6	90	10		10
7	100	0	0	
8	85	15	15	
9	100	0		0
10	100	0	0	
Average Percentages	91.5	8.5	10.83	5

The tests average a 91.5% success, which meets the Engineering Requirement 3. The false positive rate, averaged from the applicable scenarios, is 10.83%, with most of the cases occurring in Scenario 1. The false negative rate is 5%, averaged from the applicable scenarios.

The turn signals were tested when obstacles and brakes were present and absent. The trial was considered successful (”S”) if the corresponding blinker light activated on the helmet when the buttons as pressed on the bike’s User Controls. The test results are shown in Table 57.

Table 57: Turn Signals Test Results

	Left Blinker										Right Blinker									
Trials	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
No Obstacles Present	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
Obstacles Present	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
No Brakes Pressed	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
Brakes Pressed	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S

When the helmet Bluetooth is activated, the blinkers were shown to have a 100% success rate in the trials described.

The system is designed so that the user is able to connect and disconnect the Bluetooth using the buttons on RPiB’s PiJuice Hat. When the helmet is connected, obstacle detection warnings, blinker lights, and brake lights display on the helmet. When the helmet is disconnected, the functionality of the bike (RPiA) should stay constant, just without the ability to send signals to the helmet. In the tests conducted, if the helmet connected to Bluetooth successfully with a button push, meaning the signals display properly, the trial was considered a success (”S”). If the helmet was disconnected successfully, the trial was a success if the bike resumed normal operation and a ”No Bluetooth” warning was displayed to the user on the OLED Display. The results of these tests are shwon in Table 58.

Table 58: Connecting and Disconnecting Helmet Bluetooth Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Disconnect Helmet Bluetooth	S	S	S	S	S	S	S	S	S	S
Connect Helmet Bluetooth	S	S	S	S	S	S	S	S	S	S

For the twenty trails conducted, connecting and disconnecting the Bluetooth had a 100% success rate.

The photosensitive lights on the helmet were also tested in a light environment and a dark environment. For a light environment, the trial was considered a success if the LED strip did not activate. In a dark environment, the trial was considered a success if the lights activated. The results of these tests are shown in Table 59.

Table 59: Photosensitive Lights Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Light	S	S	S	S	S	S	S	S	S	S
Dark	S	S	S	S	S	S	S	S	S	S

For the trials conducted, the photosensitive lights had a 100% success rate.

Each of the Raspberry Pi's have a PiJuice HAT with a battery. The battery life of each of these was tested, with and without the execution of the relevant programs. The test results are shown in Table 60.

Table 60: PiJuice HAT Battery Test Results

	PiJuice Battery Life	
	With Code Execution	Without Code Execution
RPiA	3 hours 7 minutes	4 hours 23 minutes
RPiB	3 hours 39 minutes	4 hours 51 minutes

The battery life of all of the modules exceed three hours, which alligns with the battery life requirements for the Li-pol battery life.

One of the Engineering Requirements for this project was that the battery life should last for at least 30 miles of city travel. A series of battery life tests were conducted in which the battery was fully charged at the initiation of the test. First, the bike speed was set to 10mph for a total of three hours. The bicycle was able to maintain this speed for the duration of the test. This test was then repeated, and the conclusion was the same. For the third and fourth tests, the speed was set to 17mph for 1 hour and 45 minute intervals (the time it would take to travel 30 miles). These tests were also successful because the bike was able to maintain a speed of about 17mph.

Table 61: Li-pol Battery Life Test Results

	Trials	
	1	2
10mph	S	S
17mph	S	S

To test the responsiveness of the increment and decrements speed buttons, ten trials were conducted in which the increment or decrements speed button was pressed, and the new speed of the bicycle was recorded. The results are shown in Table 62 and Fig. 76. From Table 62, it can be seen that the speed of the motor was changed every time a button was activated. The graph in Fig. 76 depicts the change in speed of the bicycle when the number of times the increment speed button was pressed increased. From this graph, it is obvious the relationship between the speed of the bicycle and the button activation frequency is linear. Additionally, the speed of the bicycle linearly decreases as the decrement button is pressed repeatedly.

Table 62: Speed Control Buttons - Test Results

	Trials									
	1	2	3	4	5	6	7	8	9	10
Inc	S	S	S	S	S	S	S	S	S	S
Dec	S	S	S	S	S	S	S	S	S	S

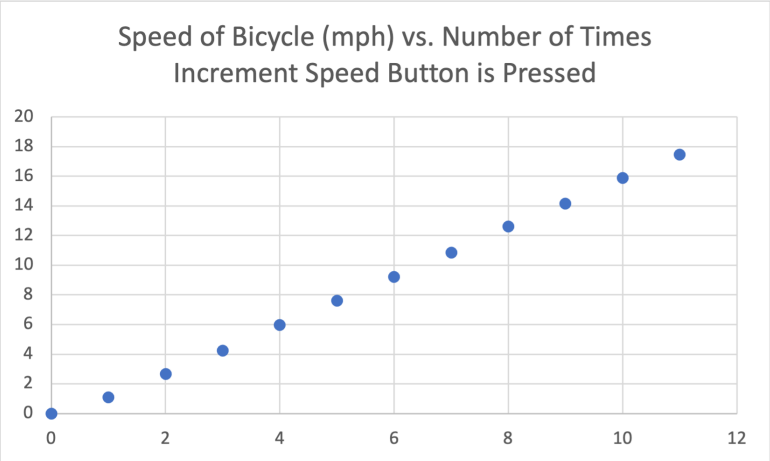


Figure 76: Plot of Bicycle Speed vs Increment Button Activation

After measuring the rotations per minute of the motor using a tachometer and from observational data, it was confirmed the measurements of RPM obtained from the VESC differed from the actual RPM by a constant factor of about 6.74. Using this information, the RPM of the motor was accurately calculated by dividing the value obtained by decoding the UART signal from the VESC by 6.74. The speed of the bicycle was calcuated from the RPM of the motor using (5).

$$\text{Speed of Bicycle (mph)} = RPM_{motor} * 0.00738 \tag{5}$$

In (5), the factor of 0.00738 converts RPM to mph and accounts for the difference in circumference between the motor and the bicycle wheel.

The bike can be slowed by braking mechanically using the preexisting padded brakes or using the pressure-sensitive brakes attached to the handlebars, as shown in Fig. 59. The time it took for the bike to completely stop from a starting speed of about 17.4 mph was recorded three times, as shown in Table 63. For these three trials, only the sensors on the handlebars were used to slow the bike; the mechanical brakes were not activated.

If the user desired to slow the bike down rather than bring it to a stop, the brake sensors can be activated for a shorter time duration than the values included in Table 63. Additionally, if the user needed to brake at a faster rate, the mechanical brakes can still be activated by pulling the brake levers.

Table 63: Braking Duration From Top Speed

	Trial		
	1	2	3
Braking Duration (sec)	4.15	4.15	4.21

The temperature of the battery was measured with an infrared thermometer over the course of twenty minutes in five minute intervals. During this time, the bicycle was traveling at approximately 10mph (while affixed to the bicycle trainer).

Table 64: Temperature of the Battery

Time Running (min)	Temperature (C)
0	21.3
5	21.5
10	21.8
15	22.3
20	22.6

The results indicate there was slight warming of the battery over twenty minutes, but the increase is minimal and therefore not a significant cause for concern.

6 Ethics Considerations

In the article “E-bike hype” from the Chicago Tribune, the health benefits and safety of e-bikes are discussed, specifically during the COVID-19 pandemic [16]. Bicycle and e-bike sales drastically increased during the pandemic as people were eager to “escape” their homes. These themes relate to the motivations of the capstone project, to design and construct a safe e-bike. Moreover, the article presents statistics on accidents involving e-bikes and evaluates their effectiveness in helping people get the recommended amount of exercise. In a study in Hamburg, Germany, it was found that when given the option, people ride e-bikes more often than traditional bicycles. Therefore, even with the pedal-assist of the e-bikes, they spent more time sufficiently exercising on e-bikes. In another study from 2000 to 2017, 3000 people arrived at the emergency room after an e-bike related injury. Because e-bikes were not popularized until recently, this statistic is likely to grow past 2017. E-bike accidents can partially be attributed to the speed of the bike, as well as the unfamiliarity of the rider [16].

According to the second principle of the IEEE Code of Ethics, it is unethical to manufacture a device that puts the user at a high risk of injuring themselves [14]. If many precautions are taken to ensure the user’s safety, however, the production of such a device provides a net

positive impact. Consumers must be able to trust companies to produce products that are not dangerous, at least to a degree. For instance, thousands of people die annually in car accidents, but this does not make production of cars unethical because manufacturers take reasonable steps to provide adequate protection for passengers. On the other hand, it would be unethical for a car manufacturer to actively ignore potential hazards in favor of increased revenue.

Therefore, even though e-bikes can be more dangerous than regular bicycles, the design of this capstone helps to improve upon the safety of e-bikes. With the inclusion of safety features in the form of sensors, lights, and sound, cyclists will be more aware of hazards and obstacles. The product will be a safe alternative to regular e-bikes for those who still want the functionality of an e-bike. Additionally, as discussed in the article, people exercise more on an e-bike than a traditional bicycle. So while this design does make it easier for people to ride their bicycles in the short-term, users will be encouraged to ride their e-bikes more often; thus, this design helps to promote a healthy lifestyle. In conclusion, our design solves some concerns the article presents about the safety of e-bikes and aligns with the positive effects of e-bikes outlined.

This product was tested, as shown in the *Experimental Test and Demonstration*, including individual testing of each component to ensure universal safety. Only trusted and verified components were purchased, as outlined in the Cost Estimate in Table 42. Maximum speed, weight, and obstacle detection parameters were defined to ensure safety in the design, as shown in Table 9.

Unlike other products, this project utilizes renewable energy to make this form of transportation even more sustainable and environmentally friendly. E-bikes are an environmentally sustainable to other forms of transportation and commuting. Additionally, components used on the design were chosen with sustainability and recyclable in mind. For example, the Li-pol battery was chosen as it has a longer life span and is safe to recycle. It is more environmentally sustainable than other batteries used in e-bikes. This analysis was done in a design matrix, shown in Fig. 6. Therefore, the design is considerate of environmental factors and practices sustainable development principles.

7 Contributions to ABET program, LMU values, diversity, social community, multidisciplinary, IEEE values

7.1 ABET Program

This project contributes to all seven of the ABET Student Outcomes [4].

This design of this system involves solving many “complex engineering problems”, as is addressed in outcome (1). The creation of an obstacle detection system with a low false positive rate and a motor controller that responds to the user’s speed and environment are nontrivial problems that involve multiple iterations of designing and testing. The flowchart for the obstacle detection system is shown in Fig. 38 and schematics and designs for the motor system can be found in *Electrical Design*

This project also contributes to solutions to public health, safety, and environmental concerns, as is addressed in outcome (2). The motivations for the project are to create an e-bike conversion kit with smart features that consider safety and power efficiency. As shown in the Marketing and Engineering Requirements in Fig. 9, the project has specific benchmarks for maximum speed and success of obstacle detection. Additionally, to address environmental concerns, a Li-pol battery was chosen over less recyclable alternatives, shown in Fig. 6. Public health concerns are addressed in the motivations of this project, as well. As the project is being developed during the COVID-19 pandemic, this e-bike conversation kit design creates an alternative to public transportation or bike-share services, both of which can increase the risk of spreading disease, specifically COVID-19. The conversion kit will allow users to travel independently, within exposing themselves to other individuals.

Outcome (3) is satisfied with presentations to audiences of a variety of technical backgrounds. Presentations were given to faculty, students, and visitors about project. Therefore, team members communicated in both a technical and nontechnical fashion about the project.

The ethics of this project were considered in the design process, and are addressed in the Ethics section of this report per outcome (4). Safety, community, environment, health, and global factors were considered throughout the duration of project development.

Outcome (5) is achieved through the collaboration of team members and faculty to "establish goals, plan tasks, and meet objectives." As shown in Teammate Roles and Responsibilities (Appendix), specific tasks were delegated to team members with defined goals and target deadlines. Team members also collaborated on many of these tasks and to solve major project issues and roadblocks.

Much experimentation and testing will be completed during the course of the project development and decisions will be made accordingly, as outlined in outcome (6). Preliminary testing is also ongoing throughout the project, and designs were adjusted and changed depending on outcomes, as shown in the Data Analytics section.

Lastly, outcome (7) is addressed throughout the research process, particularly in learning about batteries, motors, and sensors. Much background research needed to be done, including learning about existing products. The decision matrices, shown in Tables 6, 2, 33, and 4 reflect research of components considered for the design. The concept fans in Figs. 1 and 2 and Table 12 about Competitive Benchmarks.

7.2 LMU Values

LMU's mission statement states, "By intention and philosophy, we invite men and women diverse in talents, interests, and cultural backgrounds to enrich our educational community and advance our mission: The encouragement of learning. The education of the whole person. The service of faith and the promotion of justice."

The pursuit of this project certainly aligns with LMU's "encouragement of learning" pillar. Team members are learning about the technical nuances of the project, as well as skills in project management, team collaboration, and time management, as shown in the *Detailed Schedule* (Appendix) and *Teammate Roles and Responsibilities* (Appendix). This also relates to the Jesuit notion of educating the "whole person." In this project and design development, team members are not only exploring electrical engineering concepts, but also becoming educated on issues of ethics and interpersonal skills, as outlined in the Ethics and Design Impact sections of this report. Lastly, this project works towards LMU's pillar of the "promotion of justice", as it seeks to address a social justice issue related to safety, health, and environmental concerns, as addressed in the ABET Program section.

7.3 Diversity

The e-bike conversion kit was not designed to appeal to one specific demographic, rather, it is expected the conversion kit will appeal to a diverse crowd. Popular e-bike conversion kits currently available to the public often cost upwards of $1000, and therefore are not a practical option for people with more stringent budgets. On the other hand, this e-bike conversion kit will be affordable enough for many college students, or other frugal people, to afford. Also, the kit does not require the user to have advanced knowledge of bicycle mechanics. Therefore, the user-friendliness also expands the potential market for the kit.

7.4 Social Community

Not only does this project address social justice concerns, it also serves as an initiative and learning opportunity for other LMU students. The system can continue to be developed modularly by LMU Engineering students in future years. For example, LIDAR could be implemented to better detect obstacles, or steering assist could be added to the handlebars.

Additionally, this project is targeted at people like LMU students. The affordability, safety, and sustainability are what LMU students are lacking in their transportation needs. Whether it be on campus or in the surrounding urban areas of Los Angeles, this e-bike project design meets the needs of LMU community members. The *Ethics* and *Design Impact* sections further outlines the benefits this product would have for community members.

7.5 Multidisciplinary Nature

The technical nature of this project intersects with mechanical engineering, systems engineering, and safety engineering, as well as marketing and entrepreneurship. In addition to research in these fields, team members consulted with individuals in these disciplines to develop solutions for the project. For instance, when developing the design of the friction drive motor system, team members consulted with mechanical engineering students regarding how to secure the motor to the bicycle and ensure the motor maintains contact with the back wheel.

7.6 IEEE Values

IEEE's defines its "core purpose" as "to foster technological innovation and excellence for the benefit of humanity." As this project is novel and involves developing new engineering designs as shown in the *Technical Requirements* and *System Description* sections of the report.

The IEEE Code of Ethics has pillars that it expects its members to abide by. The first pillar promises to "hold paramount the safety, health, and welfare of the public" and to "improve the understanding by individuals and society of the capabilities and societal implications of conventional and merging technologies, including intelligent systems." This project addresses these concerns by prioritizing safety, as shown in the *System Requirements* (Fig. 9), Ethics Consideration section, and ABET Program section.

The project also incorporates IEEE Standards such as i2c, UART, and SPI communication, Bluetooth signaling, and brushless DC motors, as outlined in *Standards and Constraints*.

8 Conclusion

Due to concerns involving using public transportation during the COVID-19 pandemic, many bike retailers have seen dramatic increases in sales of both traditional bicycles and electric bicycles. Unfortunately, e-bike users assume some risk by riding bikes that are faster than conventional bicycles. To help mitigate the risks associated with riding e-bikes, a smart e-bike conversion kit and helmet designed to keep the user as safe as possible was developed.

The designed system is able to successfully maintain a speed of 17 miles per hour with no added pedaling. The rider is able to control the speed using buttons and brakes. The battery lasts for at least 30 miles of city travel while in constant use. The obstacle detection system, through the testing of certain scenarios, was shown to have a success rate of 90%, with a 10.8% false positive rate and 5% false negative rate in the applicable and tested scenarios. The user is alerted to potential obstacles through a screen on the bike, as well as audio and visual warnings on the helmet. The helmet also displays turn signals, brake lights, and photosensitive lights to increase safety for the user. The kit consists of 5 individual pieces, so that a user without an advanced knowledge of bicycle mechanics could install it on their own bike. The complete kit also weights less than 10 lb. The total cost of the project was $510. The estimated cost of

product of one kit and helmet would fall below this, due to costs of experimentation and testing.

The impact of this design has environmental, economical, and social benefits, both locally and globally, and if brought to market, would be a viable and preferable alternative to some existing e-bikes.

9 Suggestions

While an algorithm for removing outliers from the ultrasonic sensor measurements was developed, there are still limitations associated with the use of ultrasonic sensors over more expensive, accurate devices such as LIDAR sensors. Implementing machine learning and afixing a camera onto the bicycle would further improve the accuracy of the obstacle detection system. Automatic steering for the bicycle might also promote safety for users.

To make the system more eco-friendly, a dynamo could be attached to the bicycle wheel to power the on-board electronics like the OLED display. Additionally, the current design could benefit from Raspberry Pi batteries with long lifespans. The Raspberry Pi batteries could also be charged using the larger battery used to power the motor.

Finally, to protect the electronics from water damage and the accumulation of dirt, a better mount could be fabricated for the sensors, as well as an enclosures for the Raspberry Pi computers.

References

[1] A. Kusko and S. M. Peeran, "Definition of the brushless DC motor," *Conference Record of the 1988 IEEE Industry Applications Society Annual Meeting*, Pittsburgh, PA, USA, 1988, pp. 20-22 vol.1, doi: 10.1109/IAS.1988.25036.

[2] B. Marino, "The Testing of the HC-SR04 Ultrasonic Ranging Module." 17-Sep-2020.

[3] "Clipart Car From Above Race Top Down Clipground - Car Clip Art From Top - Free Transparent PNG Clipart Images Download. ClipartMax.com," *ClipartMax.com.* [Online]. Available: https://www.clipartmax.com/middle/m2i8i8H7H7G6d3N4. [Accessed: 11-Dec-2020]

[4] "Criteria for Accrediting Engineering Programs", 2019 – 2020 — ABET. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2019-2020/. [Accessed 10 Nov. 2020]

[5] D Shenouda and P Dawoud, "Serial Communication Protocols and Standards RS232/485, UART/USART, SPI, USB, INSTEON, Wi-Fi and WiMAX," in Serial Communication Protocols and Standards RS232/485, UART/USART, SPI, USB, INSTEON, Wi-Fi and WiMAX , River Publishers, 2020, pp.i-xl.

[6] "Electric Bike Batteries Explained ," *E-Bike Kit*. [Online]. Available: https://www.ebikekit.com/blogs/news/electric-bike-batteries-explained

[7] "BLDC Belt Motor 6354 190KV," Flipsky. [Online]. Available: https://flipsky.net/products/6354-190kv-2450w-2

[8] G. Christina, "Thinking of Buying a Bike?" The New York Times. [Online]. Available: https://www.nytimes.com/2020/05/18/nyregion/bike-shortage-coronavirus.html

[9] G. Reynolds, "E-bike hype: With their popularity soaring, studies shed light on whether e-bikes are safe or good for exercise," *Chicago Tribune*, pp 6, 2020. Available: http://electra.lmu.edu:2048/login?url=https://www-proquest-com.electra.lmu.edu/docvie w/2440832807?accountid=7418.

[10] "How Vekkit works," *Vekkit*. [Online]. Available: https://vekkit.com/

[11] "P1687.1 - Standard for the Application of Interfaces and Controllers to Access 1687 IJTAG Networks Embedded Within Semiconductor Devices." IEEE Standards Association, 2016, [Online]. Available: https://standards.ieee.org/project/1687$_1$.*html*.[*Accessed*2 – *Feb* – 2021]

[12] "IEEE 802.11-2020 - IEEE Approved Draft Standard for Information Technology – Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks – Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 03-Dec-2020. [Online]. Available: https://standards.ieee.org/standard/802$_1$1 – 2020.*html*.[*Accessed* : 12 – *Dec* – 2020]

[13] "IEEE 802.15.4-2020 - IEEE Standard for Low-Rate Wireless Networks," 23-Jul-2020. [Online]. Available: https://standards.ieee.org/standard/802_15_4-2020.html. [Accessed: 12-Dec-2020]

[14] "IEEE Code of Ethics." [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 12-Dec-2020]

[15] "Man riding bike view from above cyclist on vector image on VectorStock," *VectorStock*. [Online]. Available: https://www.vectorstock.com/royalty-free-vector/man-riding-bike-view-from-above-cyclist-on-vector-32287526. [Accessed: 11-Dec-2020]

[16] "Minnesota Bicycle Accident Lawsuits ," Sand Law. [Online]. Available: https://www.sandlawllc.com/minnesota-bicycle-accident-lawsuits-work/

[17] M. Toll, "Electric bicycle hub motors vs mid-drive motors: Which should be on your next e-bike?," *Electrek*. [Online]. Available: https://electrek.co/2018/06/07/electric-bicycle-hub-motors-vs-mid-drive/

[18] "L293x Quadruple Half-H Drivers." Texas Instruments, Jan-2016 [Online]. Available: https://www.ti.com/lit/ds/symlink/l293.pdf. [Accessed: 11-Dec-2020]

[19] "Raspberry Pi Camera: Comparison of High Quality Camera with Camera Module V2," *SeeedStudio*. [Online]. Available: https://www.seeedstudio.com/blog/2020/05/14/raspberry-pi-camera-comparison-of-high-quality-camera-with-camera-module-v2/

[20] "Revos," *RevolutionWorks*. [Online]. Available: https://revolutionworks.com/

[21] "Types of distance sensors and how to select one," *SeeedStudio*. [Online]. Available: https://www.seeedstudio.com/blog/2019/12/23/distance-sensors-types-and-selection-guide/

[22] "Ultrasonic Sensor Module." Elegoo. [Online] Available: https://www.elegoo.com/blogs/arduino-projects/elegoo-HC-SR04-ultrasonic-sensor-module-tutorial

Appendices

Codes

bike.py:

```
#!/usr/bin/python3.7
from pijuice import PiJuice
import os
pijuice = PiJuice(1, 0x14)



#Libraries
import RPi.GPIO as GPIO
import time
import numpy as np;
import timeit
from multiprocessing import Process
import os
import Adafruit_GPIO.SPI as SPI
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
import math
import bluetooth
import pyvesc
from pyvesc import VESC
from pyvesc.VESC.messages import GetValues, SetRPM, SetCurrent,
                                    SetDutyCycle, SetRotorPositionMode,
                                     GetRotorPosition
from pyvesc.VESC.messages import VedderCmd
import serial
import time

#set up motor serial port
serialport = '/dev/serial0'
ser = serial.Serial(serialport, baudrate=19200, timeout=0.05)
motor = VESC(serial_port=serialport)

#set up GPIO and declare global variables
GPIO.setmode(GPIO.BCM)
dec_button = 19
inc_button = 13
brake_button = 6
brake_button2 = 5
GPIO.setup(dec_button, GPIO.IN)
GPIO.setup(inc_button, GPIO.IN)
GPIO.setup(brake_button, GPIO.IN)
GPIO.setup(brake_button2, GPIO.IN)
bike_speed = 0

dutyCycle = 0 #motor starts at rest
max_dutyCycle = 0.5
min_dutyCycle = 0
speed_changed = False
disp_warning = False
```

```
bt = True


# OLED i2c
RST = None
DC = 23
SPI_PORT = 0
SPI_DEVICE = 0

disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)

# Initialize library.
disp.begin()

# Clear display.
disp.clear()
disp.display()

#load images
if disp.height == 64:
    image = Image.open(r'/home/pi/Desktop/warningword.ppm').convert('1'
                                                    )
    bimage = Image.open(r'/home/pi/Desktop/nobluetooth.ppm').convert('1
                                                    ')
else:
  image = Image.open(r'/home/pi/Desktop/warningword.ppm').convert('1')
  bimage = Image.open(r'/home/pi/Desktop/nobluetooth.ppm').convert('1')

#prepare for drawing text to OLED
width = disp.width
height = disp.height
image_speed = Image.new('1', (width, height))

draw = ImageDraw.Draw(image_speed)
draw.rectangle((0,0,width,height), outline=0, fill=0)

padding = -2
top = padding
bottom = height-padding
x = 0

font = ImageFont.load_default()



#set GPIO Pins
GPIO_TRIGGER = [21,16,17,25,9] #check if pin 18 works
GPIO_ECHO = [24, 20, 27,8,11]




GPIO_LEFT_BLINKER = 7
GPIO_RIGHT_BLINKER = 1
GPIO.setup(GPIO_LEFT_BLINKER, GPIO.IN)
GPIO.setup(GPIO_RIGHT_BLINKER, GPIO.IN)

theta = math.pi / 4
```

```
numSensors = 5
data_collection_iterations = 10

moving_distance = 100; #cm
danger_distance = [40,40]; #cm [S1 distance, S2 distance]
getting_closer = [0,0,0] #cm [S3, S4, S5]

false_positive_threshold = 2;



#set GPIO direction
x = 0
for x in range(numSensors):
    GPIO.setup(GPIO_TRIGGER[x], GPIO.OUT)
    GPIO.setup(GPIO_ECHO[x], GPIO.IN)

rows= numSensors;
cols = data_collection_iterations;

def distance():

    unfiltered_data = [[0]*cols]*rows
    distance = [0]*numSensors
    warning_too_close =[False, False, False] #for S3-5
    warning_moving = [False, False, False]; #for S3-5
    smallest = [1000,1000,1000];


    # set Trigger to HIGH
    x = 0

    for x in range(numSensors):
        StartTime = 0
        StopTime = 0

        for i in range(data_collection_iterations):

            toohigh = False;

            #Output Trigger
            GPIO.output(GPIO_TRIGGER[x], True)
            time.sleep(0.00001)
            GPIO.output(GPIO_TRIGGER[x], False)

            #to correct issues with Echo loop
            looptime = time.time()
            endloop = False

            #collects time while Echo is low
            while ((GPIO.input(GPIO_ECHO[x]) == 0) and (endloop ==
                                                    False)):
                StartTime = time.time()

                #to reduce error
                if ((time.time()-looptime) > 0.01):
                    endloop = True
```

```
        #collects time while Echo is high
        while ((GPIO.input(GPIO_ECHO[x]) == 1) and (toohigh ==
                                                    False)):
            StopTime = time.time()

            #if too much time (object is far enough away), exit
                                                        loop
            if ((StopTime-StartTime) >= 0.02):
                toohigh = True;

        TimeElapsed = StopTime - StartTime

        #calculate distance
        unfiltered_data[x][i]= (TimeElapsed * 34300) / 2

    #remove outliers and save filtered data into distance[]
    distance[x] = outliers(unfiltered_data[x][:],
                                        data_collection_iterations)


    #setting "Very Close" distance from S1 and S2
    if (x<2): #only S1 and S2
        danger_distance[x] = set_danger_distance(distance[x])

    #check if obstacles moving closer in "Moderately Close"
                                        distance
    elif ((x==2) or (x==3)): #only S3 and S4
        warning_too_close[x-2] = check_danger_distance(distance[x],
                                            danger_distance[x-2])
        warning_moving[x-2], smallest[x-2] = check_moving_distance(
                                        distance[x], x-2)

    else: #only S5
        warning_moving[x-2], smallest[x-2] = check_moving_distance(
                                        distance[x], x-2)
        if (danger_distance[x-3] < danger_distance[x-4]) : #use
                                        whichever side is
                                        closer
            warning_too_close[x-2] = check_danger_distance(distance
                                            [x],
                                            danger_distance[x-3
                                            ])

            #check angle between sensors to see if obstacle is
                                            moving in parallel
            #if yes, disable "Moderately Close" warning on that
                                            side
            if (warning_moving[x-3] and warning_moving[x-2]):
                sc1_check = check_scenario_1(smallest[x-2],
                                                smallest[x-3])
            else:
                sc1_check = True
            warning_moving[x-3] = warning_moving[x-3] and sc1_check

        else:
            warning_too_close[x-2] = check_danger_distance(distance
                                            [x],
```

```
                                                    danger_distance[x-4
                                                    ])

                if (warning_moving[x-4] and warning_moving[x-2]):
                    sc1_check = check_scenario_1(smallest[x-2],
                                                 smallest[x-4])
                else:
                    sc1_check = True
                warning_moving[x-4] = warning_moving[x-4] and sc1_check
                #heres where we should do scenario 1 for right side

            warning_moving[x-2] = warning_moving[x-2] and sc1_check

    #returns warning status for "Very Close" and "Moderately Close"
    return warning_too_close, warning_moving


#If determined to be moving closer in "Moderately Close" range, checks
#if obstacle is in parallel to bike
def check_scenario_1(d1, d2_actual):

    d2 = (d1 * math.sin(math.pi/12)) / (math.sin(11*math.pi/12 - theta)
                                        )

    if (d2_actual <= (d2+10)): #for error
        return True
    else:
        return False

def outliers(unfiltered_data, data_collection_iterations):

    Q1 = np.quantile(unfiltered_data, 0.25);
    Q3 = np.quantile(unfiltered_data, 0.75);

    IQR = Q3-Q1 #Inter Quartile Range

    LowerBound = Q1 - 1.5*IQR #standard practice for identifying
                                        outliers
    UpperBound = Q3 + 1.5*IQR

    values_to_remove = []
    temp = 0

    #saves outliers to 'values_to_remove'
    for j in range(data_collection_iterations):

        if ((unfiltered_data[j] < LowerBound) or (unfiltered_data[j] >
                                            UpperBound)):
            values_to_remove.append(unfiltered_data[j])

    #removes outliers from list to filter data
    for i in range(len(values_to_remove)):
        unfiltered_data.remove(values_to_remove[i])

    return unfiltered_data

#set "Very Close" distance from S1, S2
def set_danger_distance(filtered_data):
```

```
    smallest_distance = 60; #default "Very Close"

    for i in range(len(filtered_data)):

        if (filtered_data[i] < smallest_distance):
            smallest_distance = filtered_data[i];

    #return updated "Very Close" distance
    return smallest_distance

#check if S3, S4, S5 measure distances within set "Very Close" range
def check_danger_distance(filtered_data, danger):
    too_close = False

    for i in range(len(filtered_data)):

        if (filtered_data[i] <= danger):
            too_close = True

    #return status
    return too_close

#check if obstacle is moving closer within "Moderately Close" distance
def check_moving_distance(filtered_data,sensor):

    moving_distance_danger = False
    smallest = 1000

    #define how much closer is significant
    if ((np.mean(filtered_data)+10)  < getting_closer[sensor]):
        moving_distance_danger = True

    getting_closer[sensor] = np.mean(filtered_data);

    for i in range(len(filtered_data)):
        if filtered_data[i]< smallest:
            smallest = filtered_data[i]

    #return status
    return moving_distance_danger, smallest

#ONLY FOR TESTING (not used in algorithm)
def warning_lights(left_warning, right_warning):

    if (left_warning == True):
        GPIO.output(GPIO_LEFT_WARNING, True)
    else:
        GPIO.output(GPIO_LEFT_WARNING, False)
    if (right_warning == True):
        GPIO.output(GPIO_RIGHT_WARNING, True)
    else:
        GPIO.output(GPIO_RIGHT_WARNING, False)

#display images on OLED Display
def oled_display(disp_warning, bt, speed_changed, bike_speed) :

    #won't end program if OLED can't be detected
    try:
```

```
        #if there is a warning, display warning image
        if (disp_warning):
            disp.image(image)
            disp.display()

        #if the speed was changed (inc, dec, or bake), display speed
        elif (speed_changed == True):
            draw.rectangle((0,0,width,height), outline=0, fill=0)
            draw.text((x, top), "Speed : " + str(round(bike_speed,2)) +
                                                    " mph",  font=font,
                                                   fill=255)
            disp.image(image_speed)
            disp.display()

        #don't clear display for bt image
        elif (bt==False):
            disp.image(bimage)
            disp.display()
        else:
            disp.clear()
            disp.display()
    except:
        print("oled broken")

#for Testing, not used in algorithm
def oled_display_bluetooth():
    disp.image(bimage)
    disp.display()

#Bluetooth communication
def sendMessageTo(targetBluetoothMacAddress, leftw, rightw, leftb,
                                    rightb, brakeb, time_since_checking
                                    ):

    port = 1
    bt = True
    sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

    #when Bluetooth is not connected, only check again every 60s
    if ((time.time() - time_since_checking) > 60):
        try:
            sock.connect((targetBluetoothMacAddress, port))

            time_since_checking = 61

            #Warnings
            if (leftb or rightb):
                msg = "ok,"

            elif ((leftw and rightw) == True):
                msg = "BACK,"
            elif(leftw == True):
                msg = "LEFT,"
            elif (rightw == True):
                msg = "RIGHT,"
            else:
                msg = "ok,"
```

```
            #Blinkers
            if (leftb == True):
                msg = msg + "LEFT,"
            elif (rightb == True):
                msg = msg + "RIGHT,"
            else:
                msg = msg + "ok,"

            #Brakes
            if (brakeb == True):
                msg = msg + "BRAKE"
            else:
                msg = msg + "ok"
            bt = True

            #send signal in format "WARNING, BLINKER, BRAKE"
            sock.send(msg)
            sock.close()

        except bluetooth.btcommon.BluetoothError:
            bt = False
            time_since_checking = time.time()

    return bt, time_since_checking


#Decrement Button (Motor)
def decrement_duty_cycle(current_dutyCycle):

  #Decrements duty cycle when user presses 'down' button
  if (current_dutyCycle > min_dutyCycle):
    next_dutyCycle = current_dutyCycle - 0.05

    #if minimum Duty Cycle is reached
  else:
    next_dutyCycle = current_dutyCycle

    #returns new DutyCycle to write to motor
  return next_dutyCycle

#Increment Button (Motor)
def increment_duty_cycle(current_dutyCycle):
  #Increments duty cycle when user presses 'up' button
  if (current_dutyCycle < max_dutyCycle):
    next_dutyCycle = current_dutyCycle + 0.05

    #if maximum Duty Cycle is reached
  else:
    next_dutyCycle = current_dutyCycle

  return next_dutyCycle

#For Testing, not used in algorithm
def get_values_example(dutyCycle):
    i = 0

    with serial.Serial(serialport, baudrate=19200, timeout=0.05) as ser
                                            :
```

```
        try:
            while (True):
                i = i+1
                if (GPIO.input(dec_button) == True):
                    speed_changed = True
                    dutyCycle = decrement_duty_cycle(dutyCycle)
                    while(GPIO.input(dec_button) == True):
                        motor.set_duty_cycle(dutyCycle)

                elif (GPIO.input(inc_button) == True):
                    dutyCycle = increment_duty_cycle(dutyCycle)
                    while(GPIO.input(inc_button) == True):
                        motor.set_duty_cycle(dutyCycle)

                while ((GPIO.input(brake_button) == True) or (GPIO.
                                                    input(brake_button2
                                                    )==True)):
                    speed_changed = True
                    motor.set_duty_cycle(dutyCycle)
                    if (dutyCycle > 0):
                        dutyCycle = dutyCycle - 0.0005

                motor.set_duty_cycle(dutyCycle)

                try:
                    temp = motor.get_rpm()
                    print(temp)
                    bike_speed = temp
                    print(bike_speed)

                except:

                    print("didn't pass")
                    pass
                if (i % 2 == 0):

                    time.sleep(0.5)
                else:
                    time.sleep(0.35)

        except KeyboardInterrupt:
            # Turn Off the VESC
            ser.write(pyvesc.encode(SetCurrent(0)))

    return dutyCycle


if __name__ == '__main__':
    time.sleep(2)
    button_event= pijuice.status.GetButtonEvents()
    y = button_event['data']
    try:

        time_since_checking = 61
        time_before = time.time()
        with serial.Serial(serialport, baudrate=19200, timeout=0.05) as
                                                ser:
            while (y['SW2'] == 'NO_EVENT'):
```

```
start = timeit.default_timer()
leftb = False
rightb = False
brakeb = False

#multiprocessing
p = Process(target=oled_display, args= (disp_warning,
                                        bt, speed_changed,
                                        bike_speed, ))
p.start()
time.sleep(0.005)
dist = [0]*numSensors
[gettingclosewarning, movingwarning] = distance()


if GPIO.input(GPIO_LEFT_BLINKER) == 1:
    leftb = True
if GPIO.input(GPIO_RIGHT_BLINKER) == 1:
    rightb = True

leftw = gettingclosewarning[0]  or movingwarning[0] or
                                        gettingclosewarning
                                        [2] or
                                        movingwarning[2];
rightw = gettingclosewarning[1] or movingwarning[1] or
                                        gettingclosewarning
                                        [2] or
                                        movingwarning[2];
disp_warning = leftw or rightw
if (leftb or rightb):
    disp_warning = False
p.join()

if ((GPIO.input(brake_button) == True) or (GPIO.input(
                                        brake_button2)==
                                        True)):
    brakeb = True
bt, time_since_checking = sendMessageTo("DC:A6:32:D0:A5
                                        :E1", leftw, rightw
                                        , leftb, rightb,
                                        brakeb,
                                        time_since_checking
                                        )


#SET MOTOR
speed_changed = False
if (GPIO.input(dec_button) == True):
    print("Decrement Button Pressed")
    speed_changed = True
    dutyCycle = decrement_duty_cycle(dutyCycle)
    while(GPIO.input(dec_button) == True):
        motor.set_duty_cycle(dutyCycle)

elif (GPIO.input(inc_button) == True):
    print("Increment Button Pressed")
    speed_changed = True
    dutyCycle = increment_duty_cycle(dutyCycle)
```

```
                while(GPIO.input(inc_button) == True):
                    motor.set_duty_cycle(dutyCycle)


            while ((GPIO.input(brake_button) == True) or (GPIO.
                                                input(brake_button2
                                                )==True)):
                speed_changed = True
                motor.set_duty_cycle(dutyCycle)
                if (dutyCycle > 0):
                    print("Braking")
                    dutyCycle = dutyCycle - 0.0005

            motor.set_duty_cycle(dutyCycle)

            try:
                temp = motor.get_rpm()
                bike_speed = (temp / 6.735) * 0.00738
                print("duty cycle : " + str(dutyCycle))
                print("bike speed : " + str(bike_speed))

            except:
                pass
            button_event = pijuice.status.GetButtonEvents()
            y = button_event['data']

            stop = timeit.default_timer()
            #print(' Entire Program Time: ', stop - start)
        if (not (y['SW2'] == 'NO_EVENT')):
            from subprocess import call
            call("sudo shutdown -h now", shell = True)


    except KeyboardInterrupt:
        print("Measurement stopped by User")
        GPIO.cleanup()
```

helmet.py:

```
#!/usr/bin/python3

from pijuice import PiJuice
import os
pijuice = PiJuice(1, 0x14)

path = '/home/pi/stopbluetooth.txt'

bt_file = open(path, 'w')

bt_file.write("go")

bt_file.close()

import bluetooth
import RPi.GPIO as GPIO
import sys
import time
GPIO.setmode(GPIO.BCM)

gpio_left = 25;
gpio_right = 8;
gpio_blinker_left = 7
gpio_blinker_right = 1
brake = 12
brake2 = 16

GPIO.setup(gpio_left, GPIO.OUT)
GPIO.setup(gpio_right, GPIO.OUT)
GPIO.setup(gpio_blinker_left, GPIO.OUT)
GPIO.setup(gpio_blinker_right, GPIO.OUT)
GPIO.setup(brake, GPIO.OUT)
GPIO.setup(brake2, GPIO.OUT)

def receiveMessages():

  server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

  port = 1
  server_sock.bind(("",port))
  server_sock.listen(1)

  client_sock,address = server_sock.accept()
  #print "Accepted connection from " + str(address)

  data = client_sock.recv(1024)
  print ("received [%s]" % data)
  x = data.decode('UTF-8')
  signals = x.split(",")


  if (signals[0] == 'BACK'):
    GPIO.output(gpio_left, True)
    GPIO.output(gpio_right, True)
  elif (signals[0] == 'LEFT'):
    GPIO.output(gpio_left, True)
```

```
    GPIO.output(gpio_right, False)
  elif (signals[0] == 'RIGHT'):
    GPIO.output(gpio_right,True)
    GPIO.output(gpio_left, False)
  else:
    GPIO.output(gpio_left, False)
    GPIO.output(gpio_right, False)


  if (signals[1]== 'LEFT'):
    GPIO.output(gpio_blinker_left, True)
    GPIO.output(gpio_blinker_right, False)
  elif (signals[1] == 'RIGHT'):
    GPIO.output(gpio_blinker_left, False)
    GPIO.output(gpio_blinker_right, True)
  else:
    GPIO.output(gpio_blinker_left, False)
    GPIO.output(gpio_blinker_right, False)

  if (signals[2] == 'BRAKE'):
    GPIO.output(brake, True)
    GPIO.output(brake2, True)
  else:
    GPIO.output(brake, False)
    GPIO.output(brake2, False)

  client_sock.close()
  server_sock.close()

def sendMessageTo(targetBluetoothMacAddress):
  port = 1
  sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
  sock.connect((targetBluetoothMacAddress, port))
  sock.send("low")
  sock.close()

def lookUpNearbyBluetoothDevices():
  nearby_devices = bluetooth.discover_devices()
  #for bdaddr in nearby_devices:
    #print str(bluetooth.lookup_name( bdaddr )) + " [" + str(bdaddr) +
                                          "]"



if __name__ == '__main__':
  #lookUpNearbyBluetoothDevices()
  #sendMessageTo("DC:A6:32:B5:61:88")

    bt_file = open(path, 'r')

    sw2_status = bt_file.read()

    bt_file.close()
    time_since_blinker = 0
    time.sleep(2)
    x = pijuice.status.GetButtonEvents()
    y = x['data']
```

```
    try:
      while (y['SW2'] == 'NO_EVENT'):
        receiveMessages()
        x = pijuice.status.GetButtonEvents()
        #print(x)
        y = x['data']
        #print (y)
        print(y['SW1'])
        print(y['SW2'])
        print(y['SW3'])

        #if (y['SW2'] == 'NO_EVENT')
      GPIO.output(gpio_blinker_left, False)
      GPIO.output(gpio_blinker_right, False)

      GPIO.output(gpio_left, False)
      GPIO.output(gpio_right, False)
      GPIO.output(brake, False)
      GPIO.output(brake2, False)
      if (not(y['SW2'] == 'NO_EVENT')):
        from subprocess import call
        call("sudo shutdown -h now", shell=True)

    except:
      GPIO.output(gpio_left, False)
      GPIO.output(gpio_right, False)
      GPIO.output(brake, False)
      GPIO.output(brake2, False)
```

Teammate Roles and Responsibilities

Throughout the course of the project, Marena has primarily focused on the battery and motor system. This involved programming and configuring the motor using the VESC, as well as establishing serial communication between the VESC and the Raspberry Pi. She also helped to develop Python code for the user to control the motor. Marena designed the mechanical aspects of the project, including the manufactured mounts for the motor and sensors.

Megan has primarily focused on the sensors and smart components of the project. She developed and tested the algorithm for obstacle detection, as well as integrated the other electronics into the system. This included displaying warnings safely to the user. She wrote, tested, and integrated the codes on the Rasbperry Pi's ('bike.py' and 'helmet.py'). Megan also worked on developing wireless communication between the bike and the helmet and safe external executions and terminations of the codes.

Especially during the testing phases of the project, Marena and Megan worked collaboratively to integrate their pieces of the project. They tested the units and the system together and made decisions and actions to improve the efficiency, safety, and functionality of the system.