ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Cycling Assist System with Rear Camera Detection

Team #12

JACOB BETZ

(jmbetz2@illinois.edu)

TRISHA YADAV

(tyadav4@illinois.edu)

JINGDI LIU

(jingdil2@illinois.edu)

TA: Kai Chieh Chang

December 4, 2022

Abstract

Cycling is an extremely popular activity. Every year, biking related purchases increase. However, many individuals get injured in bike related accidents. Over 130,000 people are injured while biking in the United States every year [1]. Along with this, the number of preventable bicycle related deaths increased by 16% in 2020. One thing that can help prevent accidents for cyclists is a rear view detection and warning system. This could be imperative to the safety of cyclers on main roads or busy trails. This project successfully created a system where a user can place a camera on the seat of their bicycle to capture a live video feed. This live video feed is shown on a display system on the handlebar of the user's bike. This display system will also use LEDs and a buzzer to inform users about approaching vehicles and bikes. It can also provide warnings about objects approaching in left or right blind spots.

Contents

1 Introduction

1.1 Purpose and Functionality

Cycling is a very popular activity among many different age groups. With new eco-conscious initiatives launching across the United States, more and more people are opting for cycling as a mode of transportation. However, as common of an activity as it is, biking can also be extremely dangerous. Because more people are using cycling for transportation, there are even more bicycles riding on main roads. While new monitors and tracking systems for bicycles appear on the market all the time, there are a few safety features that have yet to be made.

A necessary feature for frequent cyclers is rear view detection. Along with this, while there are many different detection systems available, many require you to order separate sensors to monitor different features. There are very few systems that encompass different cyclist safety features in one compact dashboard. For example, the Garmin Varia RTL515 uses RADAR blindspot detection [2]. However, in order to detect other features, another sensor must be bought and installed in parallel to this sensor. This sensor system can also be very costly.

In order to solve this issue, this project contains a sensor system paired with a display dashboard to assist cyclists. The sensor system contains a small 1080p USB camera that is placed so that it faces behind the bicycle. The display system is a 3D-printed enclosure containing an LCD screen, an array of LEDs, and a buzzer. The LCD screen will display the live feed coming from the rear facing camera. This video feed will then be sent into TensorFlow Lite object detection, running on the Raspberry Pi 4. The algorithm will detect vehicles and bicycles approaching the cyclist. This will then get sent to OpenCV to perform distance calculations on these objects. If an object comes within 10 meters of the cyclist, the user will need to be informed on the display system. The display includes an STM32 chip mounted directly on a PCB. The STM32 microcontroller receives warning signals from the Raspberry Pi about approaching objects. The STM32 then turns on the appropriate LED to alert the cyclist. Along with this, a buzzer will create noise if an ob-

ject comes within 2 meters of the cyclist. Another feature is that there are LEDs assigned for objects in the left and right blindspots of the cyclist. These systems together should successfully prevent cyclists from accidents on the road.

1.2 Subsystem Overview

Our project has three subsystems. Figure 1 below shows our block diagram with these subsystems. The first is the power subsystem with a 10Ah Lithium Polymer battery, a 5V Boost Converter, and a 3.3V linear voltage regulator. This is able to power our rear view camera and object detection subsystem. This subsystem is responsible for capturing the live video and displaying it on the dashboard. It is also responsible for processing this video output. The last subsystem is the dashboard warning system. This system contains the PCB, LEDs, STM32 microcontroller, and the buzzer. The design of each of these subsystem is further broken down below in the Design Description section.

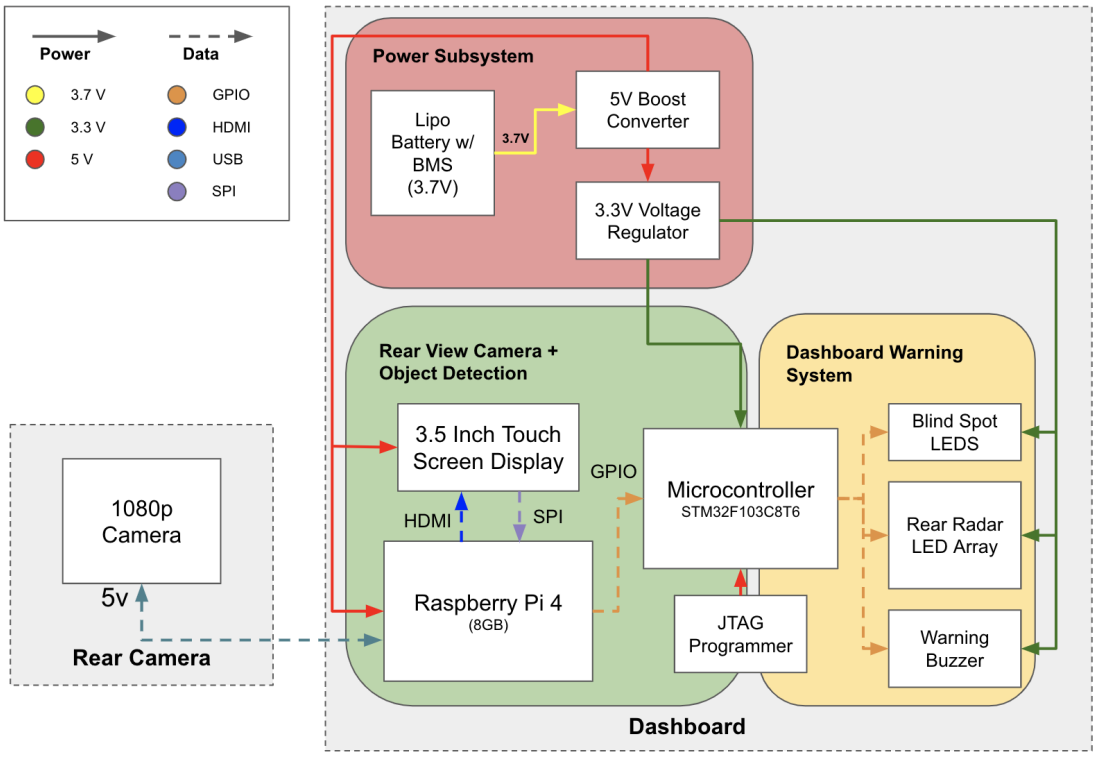

Figure 1: Block Diagram

1.3 High-Level Requirements List

Three quantitative characteristics that this project must exhibit in order to solve the problem mentioned include:

1. The system needs to be able to use a camera and display combination to show a live rear view of the bicycle while riding.
2. Device needs to detect rear approaching objects within at least 10 meters using object detection libraries and algorithms.
3. Device needs to be able to inform the user using LEDs and a buzzer how close a vehicle is to the cyclist and if an object was detected in the bicycle's blind spot.

2 Design

2.1 Design Description and Justification

2.1.1 Power Subsystem

The power subsystem is responsible for supplying power to our entire project. Due to the nature of our project, various voltages are needed to power our entire system. The power subsystem features a 10Ah Lithium Polymer battery, a unique 5V Boost Converter built specifically for Raspberry Pi 4 based projects, and a 3.3V linear voltage regulator. The Lithium Polymer battery will provide the power at 3.7V and with a high continuous discharge rate of of 5A continuous. This connects through a JST connector to the Amp Ripper 3000 Boost Converter that will supply a continuous 3A current through the USB-C output. Using the onboard pins, it is also the 5V supply to the PCB, providing a 5V input to the 3.3V linear voltage regulator to power our LEDs, buzzer and microcontroller.

2.1.2 Rear View Camera + Object Detection

The Rear View Camera and Object Detection subsystem is the core sensing and processing part of our project. This subsystem is responsible for capturing the live video from the rear of the bicycle, displaying the live video on our 3.5 inch display and processing the video using complex computer vision algorithms and libraries. The subsystem features a unique USB camera capable of filming in both day and night conditions using an IR system. The camera will connect directly to the Raspberry Pi 4 (8GB model) to be displayed and processed. The 3.5 inch display clearly shows the cyclist what is behind them.

The Raspberry Pi 4 includes custom algorithms for processing the video and computer vision libraries, namely OpenCV [3] and TensorFlow Lite[4], to detect the objects behind the cyclist. The entire process uses TensorFlow Lite to detect the vehicles, pedestrians and other cyclists behind the cyclist and OpenCV to determine distance and location of those objects behind the cyclist. Once the video has been processed, the Raspberry Pi 4 sends flags over GPIO to the warning system to warn the cyclist when rear objects could be in a dangerous position.

2.1.3 Dashboard Warning System

The dashboard warning system is built on a custom PCB and will feature multiple LEDs, an STM32 microcontroller and a buzzer to warn the cyclist effectively. The STM32F103 series microcontroller has the necessary power to drive all our LEDs and buzzer as well as connect to the Raspberry Pi through GPIO pins. The STM32F103 series supplies up to 25mA per GPIO pin and runs at a 3.3V input. Since the project features a total of 16 LEDs of various colors (20ma), MOSFETs are used to power each set of LEDs at one time. Along with the warning LEDs, a piezoelectric buzzer is turned on when objects are dangerously close to the cyclist. The microcontroller controls which set of LEDs and when the buzzer powers by using flags sent through the 5 GPIO connections to the Raspberry Pi. Finally, to program the STM32F103 microcontroller, JTAG connections for a ST LINK-V2 are used to quickly reprogram the processor. All of these connections can be seen in the circuit schematic below.

2.1.4 Hardware Schematics

The following 2 3 shows the final circuit schematic connecting all major subsystems together as described above. Further images of individual parts and close ups are shown in Appendix A below.

2.1.5 Physical Design

The physical design of the dashboard includes an entirely custom 3D printed enclosure, made out of PLA, that will hold the Raspberry Pi 4, Amp Ripper 3000 boost converter, custom PCB and battery securely in place on the bicycle. The dashboard is securely mounted around the front handle bars of the bicycle and its total dimensions are 7.3x6.0x5.4 inches (236.52 cubic inches).

To secure the dashboard to the bicycle, hose clamps wrap through the open loop on the underside of the dashboard, attaching to the cylindrical structure of the bicycle below it. Our design also includes a quick access battery and charge port box at the bottom of the enclosure. This was designed so that users could quickly hot swap batteries and have

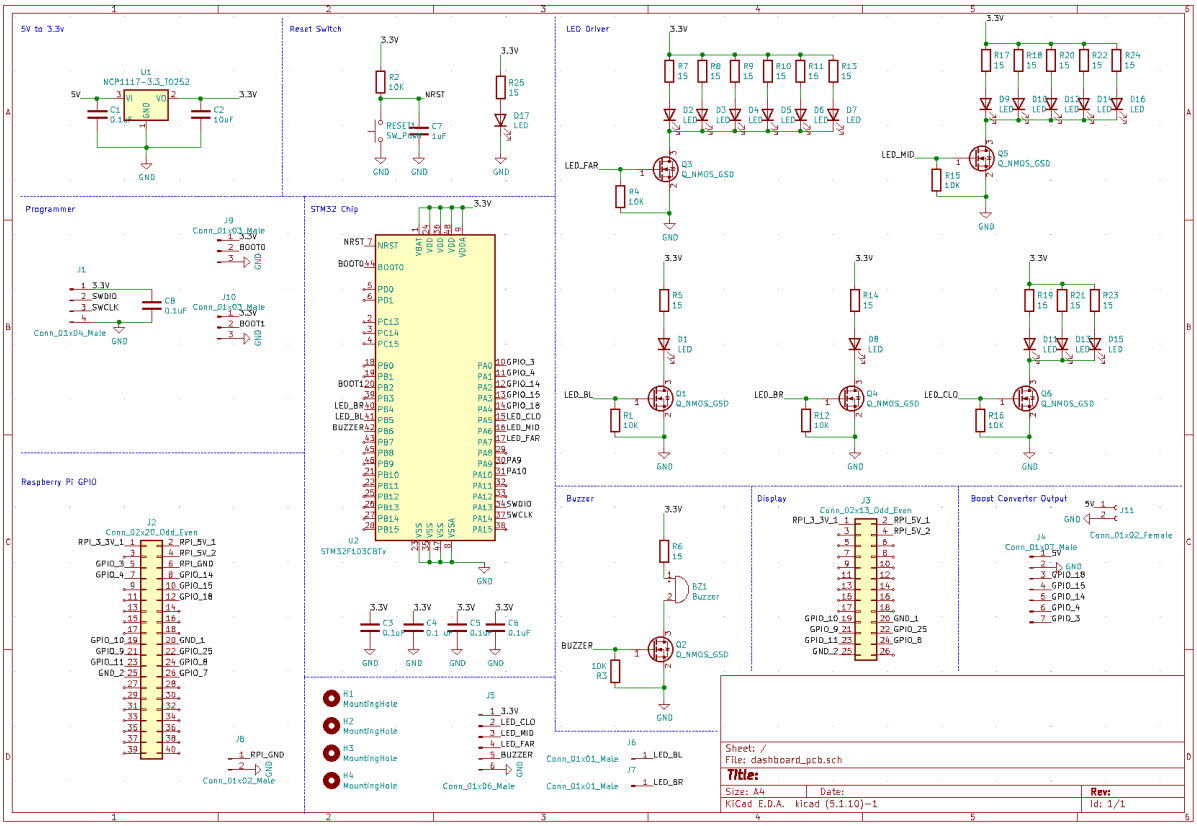

Figure 2: Circuit Schematic

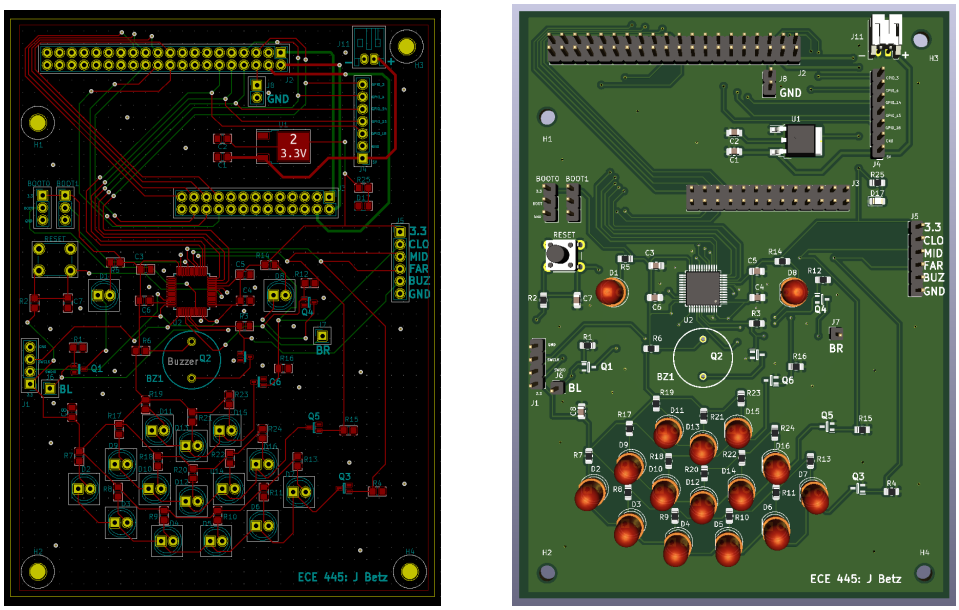

Figure 3: PCB Schematics

access to recharge the battery easily. Also note the square shield surrounding the USB ports of the Raspberry Pi 4. This is used to cover the Raspberry Pi's outputs as well as route the USB camera cable to the rear of the bicycle.

2.1.6 Software Design Raspberry Pi

The Raspberry Pi takes input from the ArduCam USB camera and performs object detection before estimating the distance to the object. The image is first preprocessed by OpenCV [3], and then is input to a pre-trained TensorFlow Lite model. The model we used is a MobileNet V2 Single Shot Detection model [4] that was trained on COCO dataset and additionally trained on Berkeley Deep Drive 100k dataset [5]. The model was used to detect objects behind the cyclist such as vehicles, pedestrians, and other cyclists with more than 60% accuracy while keeping the video output on display to be more than 10fps. Then in OpenCV we utilized triangle similarity [6] to estimate the distance between the cyclist and detected objects. This process of distance estimation is explained below in the Distance Estimation 2.2.2 section. Once the distance has been calculated, warning signals will be generated by Raspberry Pi 4 to the STM32 IO based on the distance estimated to be less than 2m, 5m, and 10m.

2.1.7 Software Design STM32 Software

The STM32F103C8T6 is a low cost, low power microcontroller that is used to drive warning system LEDs and buzzer on the dashboard. The STM32 HAL integrated library makes it possible to connect to the various warning devices and to the Raspberry Pi 4 using GPIO pins. A custom loop is ran while the system is powered on that monitors all input GPIO from the Raspberry Pi then uses frame counters to decide which warning devices should be turned on. Sensitivity code was produced to remove erroneous frame counts from the computer vision software. This code allows for a warning that is longer sustained to warn the cyclist of approaching vehicles. All GPIO inputs and outputs are set to source or sink at a constant 3.3V in the HAL library code. Figure 17 highlights this decision making process.

2.2 Equations and Simulations

2.2.1 Operating Voltage and Current Calculations

One of our most challenging portions of this project is the power requirement. We have a unique system that has a computer that processes live video, therefore the Raspberry Pi 4 requires a lot of power. We also have an entire other low power microcontroller that is responsible for controlling many different warning devices. Therefore a complex analysis of our power requirements was crucial for our success in this project.

The Raspberry Pi 4 requires at least 5v at 2.5A [7] with a USB device less than 500mA. In our case, our Arducam USB camera has a maximum current of 370mA and our 3.5 inch Touch Display has a maximum current of 120mA [8], meaning that our Raspberry Pi 4 will need a 5V power supply at least 2.5A of continuous current.

Next our STM32F103 microcontroller requires a 0.3-4V max main supply voltage at a maximum current of 150mA [9]. Therefore a 3.3V supply up to 150mA would be sufficient to power our STM32F103. Each LED has a current consumption of 20mA peak a piece [10]. With 16 total LEDs, that is 320mA of total current consumption. Finally our piezoelectric buzzer has a maximum working current of 10mA [11]. This brings our total Dashboard Warning Subsystem current consumption to 480mA.

Another component to include in our operating voltage and current calculation is the 3.3V linear voltage regulator. Since our requirement for our Dashboard Warning subsystem is 480mA, the NCP1117DT33G works perfectly [12]. It has an input voltage range of 4.3V-20V and a maximum current of 1000mA. It has a maximum current consumption of 10mA, therefore this regulator will fit our requirements for the warning system.

Therefore our total current consumption for our entire project is 2985mA or roughly 3A maximum current consumption.

The Amp Killer 3000 Boost converter can supply 5V at a continuous 3A (with sufficient cooling). Therefore even at peak loads, the Raspberry Pi 4 will not be throttled. And it can supply the entire project system with the necessary power.

The final component to include is the battery. The Amp Killer 3000 requires a battery that has a 3.7V-4.2V input and at least 3000mAh capacity. We have chosen a 3.7V Lithium Polymer with 10000mAh capacity. This will be able to supply the boost converter with the correct voltage and will have enough capacity for several hours of use, even at peak consumption.

From this analysis we learned that our power requirements are met in our system, even when every single device is active and under peak load (a situation that should theoretically never happen). However, it is easy to see how not computing the requirements and analysing the outcome could have had consequences to the performance of our project. For example, if the Raspberry Pi 4 does not meet its power requirements the performance will be throttled. This could cause the object detection algorithm's accuracy to drop and miss a moving object.

Benchmarks and power measurements results [13] can be helpful for us to estimate the power consumption for our Raspberry Pi 4 performing object detection. Under an idling condition, the Raspberry Pi consumes 575mA with a 5V input voltage. And the consumption peaked at roughly 885mA when loading IXDE. The 1080p video shooting power consumption was measured to be 640mA under required input voltage. Since our Raspberry Pi will be required to perform at least 480p video shooting, object detection, and video display with 10+ fps, an estimated power consumption for these tasks should be about 2500mA.

2.2.2 Distance Estimation

To keep our cost low, computer vision was used to estimate objects' distance to the cyclist, instead of using LiDAR. The design of distance estimation algorithm has incorporated the idea from triangle similarity [6]. Precisely, we used a marker with known width W, and took a picture of the marker with some known distance D from the object, which enabled us to find the perceived width P in pixels in the image. we calculated the focal length F of our USB camera using formula: F = (P x D) / W because we have a ratio equation F : D = P : W due to triangle similarity when we take pictures using a camera. The marker

we used was a paper with known width w = 11.7 inches. The distance D when we took a picture for the white paper was 54 inches. Eventually, the focal length for our camera was calculated to be 1180.

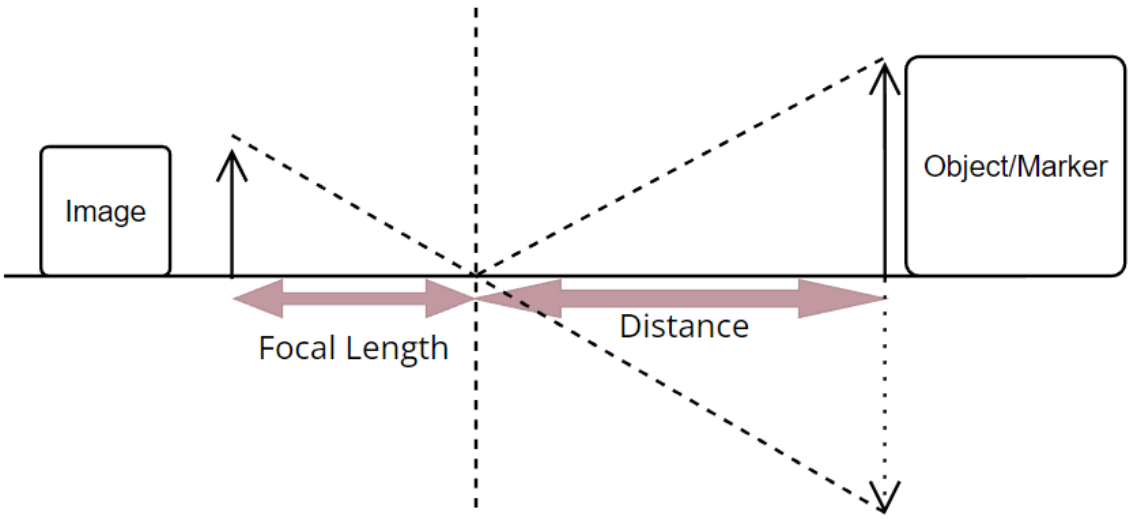

Figure 4: Triangle Similarity for Distance Estimation

Knowing the focal length F of our camera, we were able to estimate the other objects' distance D' to the camera with known object width W', known pixel width P, and the calculated focal length F using formula: D' = (W' x F) / P. Note that we used the average car width W' = 70 inches in order to apply this formula to estimate distance to a car. And we also used average width of a person W' = 16 inches because our project not only detects cars, but also pedestrians and cyclists. Therefore, we had to differentiate actual width between different captured objects.

2.3 Design Alternatives

2.3.1 Power Subsystem

As stated in the 2.2.1 our power requirements were fairly complex in this project. While spending extra time calculating our peak power requirements helped tremendously, we still ran into power issues when testing the project on the bike itself. We found that when running our Raspberry Pi without any throttling of performance, the entire power subsystem would shut down and only supply 1.0V. This initially puzzled us as all of our hardware was designed to run at peak power consumption. Upon further investigation we found that our 3.7V LiPo battery had an internal BMS that would limit at a 2.5A current draw despite the battery cell itself being rated for 10A peak current. To surpass this issue, we turned on our Amp Ripper 3000's internal battery monitor using its onboard microcontroller and soldered new connections to the battery. This allowed us to pull the current we would need while still monitoring the battery's voltage for safety concerns. Looking back, we are happy that we accurately calculated the exact power requirements otherwise this issue may have not been able to be resolved.

2.3.2 Computer Vision Software

Rather than training an object detection model by our own, we carefully selected an object detection model from TensorFlow [4]. The model consists a MobileNet V2 model as a backbone network, and a Single Shot Detection model to perform classification and to make predictions.

The advantage that MobileNet V2 carries along with is its efficiency and performance running on edge devices such as Raspberry Pi. It was used as a backbone network for feature extraction in our project. Within each block of MobileNet V2, it has an expansion layer to expand the data dimension, a depthwise layer to extract and filter features from data, and a projection layer to compress data. With this bottleneck structure within each block, the MobileNet V2 model gave us 72% mean average precision as well as above 4.5 average fps.

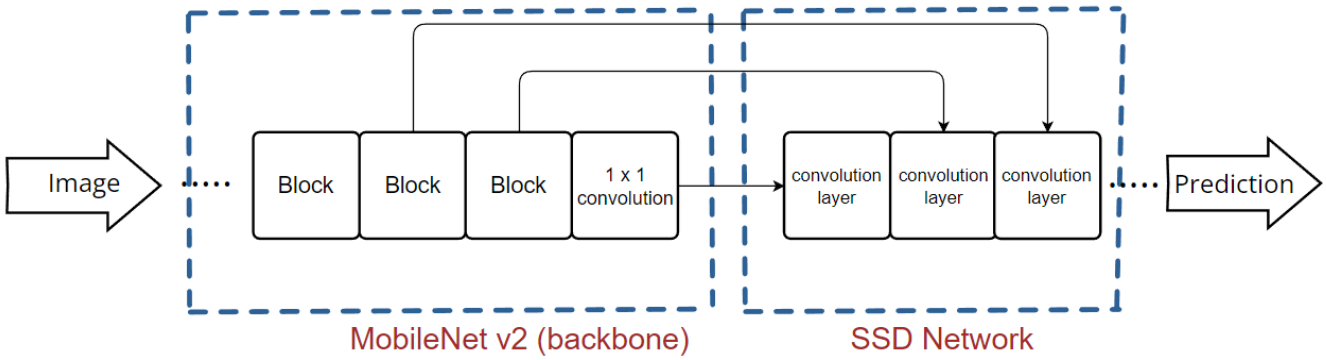

Figure 5: Model for Object Detection

Single Shot Detection has been used as classification network in our pre-trained object detection model. The reason we chose SSD instead of YOLO is because SSD takes feature maps of different sizes from many previous blocks in MobileNet V2, so that it is capable of using lower level features for classification. Since SSD network has many convolutional layers, it is crucial that we choose a efficient feature extractor network. This pre-trained network has been trained on COCO dataset, and then trained on Berkeley Deep Drive 100k dataset [5], which is a dataset that covers extensive data related to various kinds of road conditions. For example, it has information covering different lighting condition, different weather condition, and different lane setup on the road from different cities.

3 Requirements and Verifications

3.1 Power Subsystem

Requirements	Verification
• The power subsystem must be able to supply at least a 2.5A continuous current output and up to 3A total at peak loads.	• Apply a constant 2.5A load for 30 minutes (time of average bike ride). Monitor that the current stays at a constant 2.5A for all 30 minutes of operation. • Apply a constant 3A load for 10s. Monitor that the current stays at a constant 3A for all 10s of operation.
• The power subsystem must be able to supply both a 5V and 3.3V output to the system at a tolerance of ±10% each.	• Connect battery and boost converter to the system. Check voltage using multimeter that no output voltage falls outside of 5V±10% and 3.3V±10%.
• The battery must be able to be easily recharged via USB input and will stop when the battery reaches 4.2V. The battery must be able to last at full charge for at least 3 hours.	• Start with battery voltage around 3.7V, connect USB cable to charging input of boost converter, monitor current delivered to battery and monitor battery voltage. Verify that the charging stops when the battery reaches 4.2V. • Apply constant 2.5A load and record time it takes for battery voltage to drop under 3.7V. Verify that this time is at least 3 hours.

3.1.1 Results

The power subsystem was our first subsystem to test and verify since it is crucial to run and test the other subsystems. For our first requirement, we were able to verify that we could supply 2.5A constant for 30 minutes of operation in the lab along with a 3A load for 10s. We were also able to find we could supply a peak of 3.5A for a few seconds since we had efficient cooling on our boost converter. The power subsystem was also able to supply a final voltage at any battery voltage at 5.21V and 3.33V which was within our tolerance of 10%. This was verified using the digital multimeters in the lab. Finally, when our project was fully assembled we were able to charge the battery in roughly 2 hours to the max voltage of 4.2V. Then we fully discharged the battery while running the system, finding that the battery life lasted 3.44 hours. Therefore all requirements were met.

3.2 Rear View Camera + Object Detection

Requirements	Verification
• The Raspberry Pi 4 must be able to drive the 3.5 inch display at a frame rate of 10fps or higher with the live video from the rear camera.	• Connect the rearview camera to the Raspberry Pi 4 via USB connection and connect the 3.5in display to the Raspberry Pi. Verify using the Raspberry Pi's onboard fps counter that the display consistently displays at least 10fps when viewing the camera image.
• The Raspberry Pi must be able to use object detection libraries and algorithms to detect moving objects behind a cyclist and shine the correct LED array depending at 2m, 5m and 10m ±20% each.	• Have a pedestrian carefully run behind the cyclist at 2m, 5m and 10m. Verify using a multimeter, that each of the GPIO pins for the 2m, 5m, and 10m flags send a high signal depending on the pedestrian's location.
• The Raspberry Pi must be able to send flagged objects to the STM32 microcontroller through GPIO pins.	• Have a pedestrian stand behind the cyclist at 5m. Use a multimeter to verify that the GPIO pin for a 5m warning is sent.
• The rearview camera must be able to capture video at a resolution of at least 480 x 380 in both day and night conditions.	• Connect the rearview camera to the Raspberry Pi in both day and night conditions. Verify that the pixel count in OpenCV for the video is at least 480 x 380.

3.2.1 Results

This subsystem met all verification criteria fairly easily. The first requirement was verified through the display frame counter which was set up to display the live video feed without computer vision running. We were able to achieve an average frame rate of roughly 29 fps on 10 minutes of live video. Even when running with computer vision we can achieve 10 fps. When testing the accuracy

of our distance estimation, we verified using a team member located behind the bicycle at 2m, 5m, and 10m of distance with pedestrian tracking on. We were able to detect the pedestrian at each level of distance at about 5-10% accuracy. We also ran this same test on a car while riding the bicycle to verify that the the distance measurement was accurate still. Next all the GPIO was verified using the digital multimeter in the lab that an object at 2m, 5m, and 10 would send a GPIO signal to the STM32. Finally, the resolution of the camera was found to capture images at a resolution of 1280x720 which exceeds our requirement of 480x380. Therefore, each requirement was verified successfully with no issues.

3.3 Dashboard Warning System

Requirements	Verification
• The STM32 must be able to source each gate of the MOSFET to turn on each set of LEDs: L Blind Spot, R Blind Spot, 2m LEDs, 5m LEDs, and 10m LEDs within 500ms when conditions are met.	• Using a voltage supply, supply a 3.3V input to one of the GPIO flags from the Raspberry Pi to the STM32. Using a camera with high capture rate, verify that as the input is set the LEDs turn on within 500ms.
• Verify that when conditions are met for the Piezoelectric buzzer to sound that the buzzer is at least 50dB. Verify that the buzzer is under the safe level of 120dB.	• Apply a 3.3V input to the buzzer inside the dashboard enclosure. Using a decibel meter, verify that the sound at the average cyclist head level is within the bounds of 50dB and 120dB.
• Verify that when the LED warning indicators are turned on, it can be seen in both day and night conditions.	• Have a pedestrian stand behind the cyclist at 5m in both day and night, verify that the LED light is visible even in sunny conditions.

3.3.1 Results

Each requirement in the Dashboard Warning System was also verified with no issues. All of the STM32 GPIO pins were able to source the gate of the MOSFET for each warning device at a 3.3V

voltage at a response rate of 50ms even with the added sensitivity code. This was verified using our phone camera when clicking run on the software to send the high GPIO signal as well as using the internal clock rate of the microcontroller 72MHz and counting our loop time of roughly 1ms. Next, the decibel measurement was found, at cyclist height with the lid on, to be 75dB which was within our safe level. Finally, Figure 16 shows an object being detected at 5m at night in low-light conditions without issue.

4 Conclusion

4.1 Accomplishments

This project successfully met the high level requirements declared at the beginning of the semester. The final product is able to use a combination of a camera and display to show a live rear view of the bicycle while in motion. It also can detect other vehicles and bikes within at least 10 meters using object detection. Lastly, the product is able to inform the user about the distance away from approaching objects using LEDs and a buzzer, along with blindspot warnings. The subsystem requirements for this project were also met. One requirement that is important to highlight is the power subsystem. Because this project requires a high current load, it was important to make sure that the power subsystem was able to supply enough current to support object detection. This was successfully accomplished. Another subsystem requirement worth highlighting is the object detection requirement. Accurate distance detection was an integral part of this application. After multiple rounds of testing different object detection libraries, a library was selected that was specifically trained for vehicles. This allowed for an accurate distance detention algorithm, which ensured that users would accurately understand how far objects were away from them. Another project success is the use of night vision. When the project was tested at night, it was still extremely accurate in low light scenarios. With all of these accomplishments combined, this project was able to be extremely successful.

4.2 Uncertainties

One uncertainty regarding the project is the sensor system's ability to be used in all weather conditions. For this sensor to be ready to be put on the market, cyclists would need to be ensured that they could use this technology in any condition. With this in mind, the display system consists of a waterproof, 3D-printed enclosure. However, a similar enclosure was not built for the camera. This could lead to potential issues regarding the durability of the product. Due to this, the product was not tested in all weather conditions. The object detection library stated that it was trained in all weather conditions, meaning that it could detect objects in rain, snow, or hail. However, the product was only tested in snow. After a waterproof enclosure is built for the camera, further testing will need to occur. In the event of unsatisfactory results, the code and algorithm may have to be adjusted.

4.3 Future Work and Alternatives

One thing to explore in the future is the use of LiDAR, radar, and sonar sensors. These sensors are what traditional distance detection products on the market utilize. For example, the Garmin Varia RTL515 discussed in the introduction utilizes RADAR distance detection. This project chose to use a distance detection algorithm with OpenCV instead. The main reason for this was to ensure that this product was low cost. However, it would be an important next step to explore the difference in accuracy between the different distance detection methods. The team would have to research if the difference between the accuracy of the different methods was enough to prevent more accidents, and determine the best choice for the product.

Another consideration for this project is a user interface that can be utilized to adjust the sensitivity of the warnings. In the first iteration of the product, the LEDs would flash warnings sporadically. This was due to the object detection algorithm quickly detecting an object, and then the object disappearing. The team then adjusted the code so that the LEDs would only flash if an object was detected for ten frames. The LEDs would then shine for at least 5 counts. However, different users may have different preferences as to how sensitive to warnings they would want their device to be. A user interface where cyclists could turn down or turn up the sensitivity could help make the product more applicable to different types of cyclists.

4.4 Ethicical Considerations

There are some ethics and safety policies that should be considered carefully. The purpose of this project is to assist cyclists using a sensor system and a user display system to keep track of the rearview and stability of a bike. This purpose falls under safety standards by IEEE's Code of Ethics Section I.1, which is "to hold paramount the safety, health, and welfare of the public... and to promptly disclose factors that might endanger the public or the environment" [14]. This project aims to assist cyclists by giving effective warnings and informing them of their status while they are cycling. However, the system will not be able to physically assist cyclists by preventing accidents, which means the risk of cycling will not be eliminated. Therefore, group members will explicitly mention this to users before giving any further instruction, and this follows the IEEE's Code of Ethics Section I.5, which is to "acknowledge and correct errors, to be honest, and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others"[14].

Furthermore, the group members working together respected each other and treated others fairly through frequent and effective communication either in person or online. This practice follows IEEE's Code of Ethics Section II.2, "to not engage in discrimination based on characteristics such as race… gender identity, or gender expression" [14]. Along with this, the team made sure to follow all of the ECE lab safety rules, as stated in the university's Laboratory Safety Training by the Division of Research Safety.

Finally, the team followed the COVID-19 CDC Guideline when planning to meet in person to work on the project. This falls under the COVID-19 CDC Guideline, "Reiterating that regardless of vaccination status, you should isolate from others when you have COVID-19" [15] and "Recommending that if you test positive for COVID-19, you stay home for at least 5 days and isolate… Wear a high-quality mask when you must be around others at home and in public" [15]. The team also followed the Lab Safety Guidelines when working on PCB and circuits.

A Schematics, Charts, and Photos

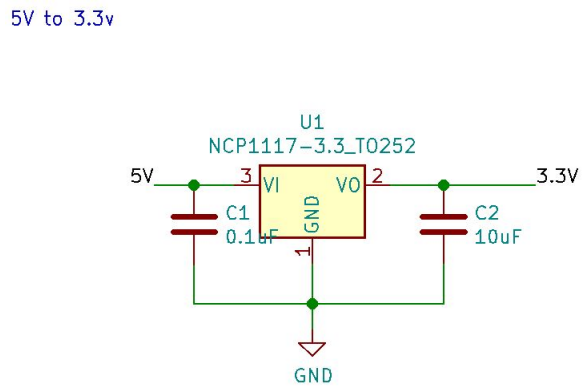

Figure 6: Voltage Regulator

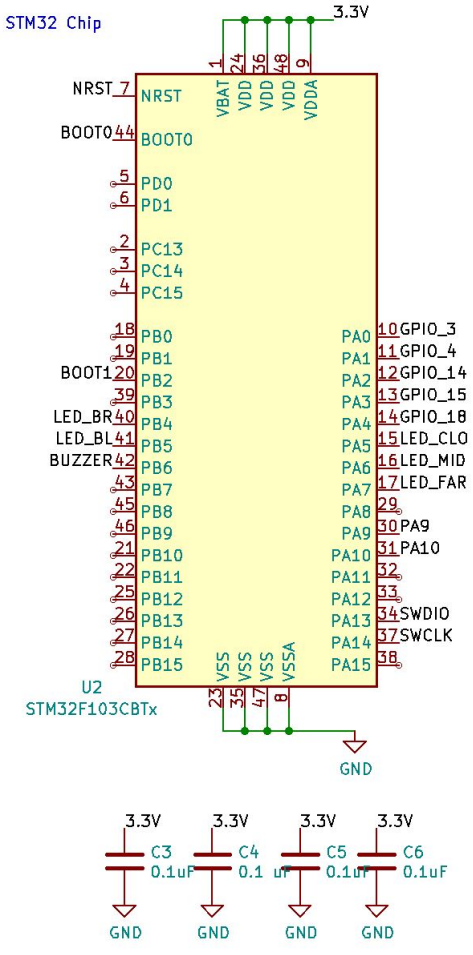

Figure 7: STM32 MCU Schematic

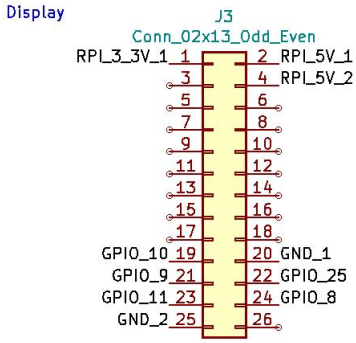

Figure 8: Display Connections

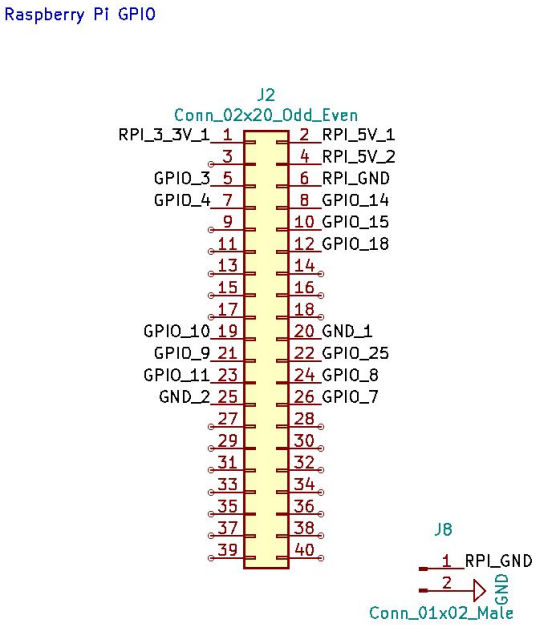

Figure 9: Raspberry Pi Circuit Connections

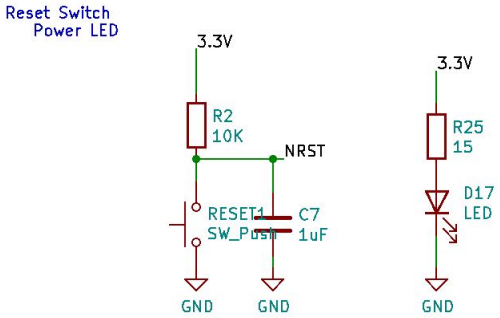

Figure 10: Reset Switch and Power LED

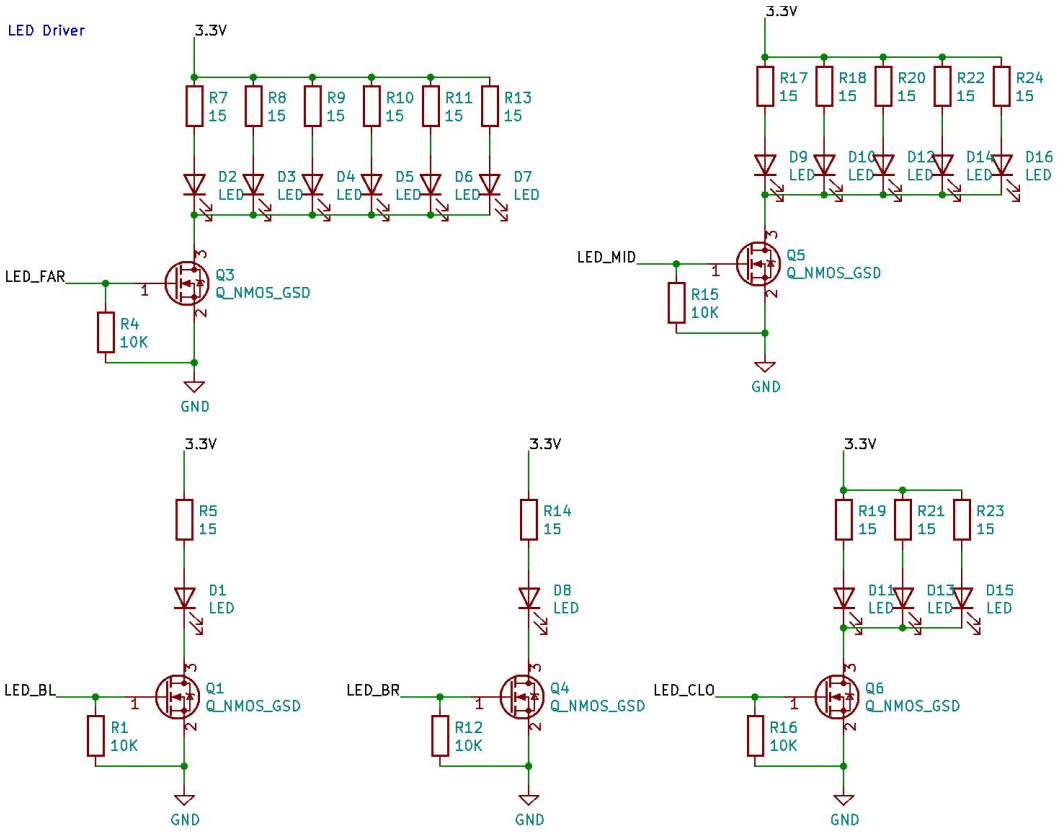

Figure 11: LED Drivers

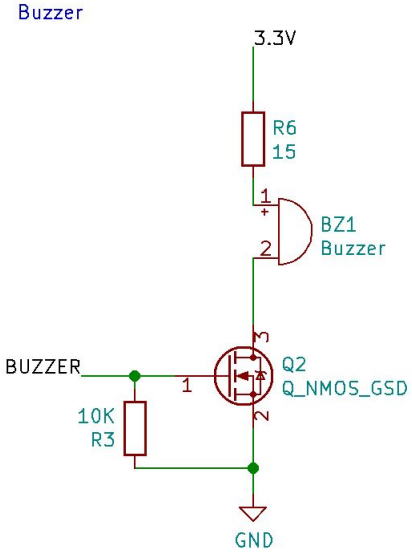

Figure 12: Buzzer Driver

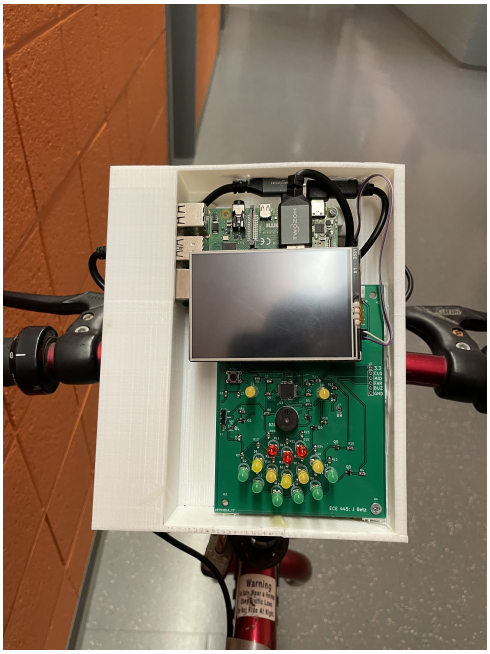

Figure 13: Front View of Dashboard

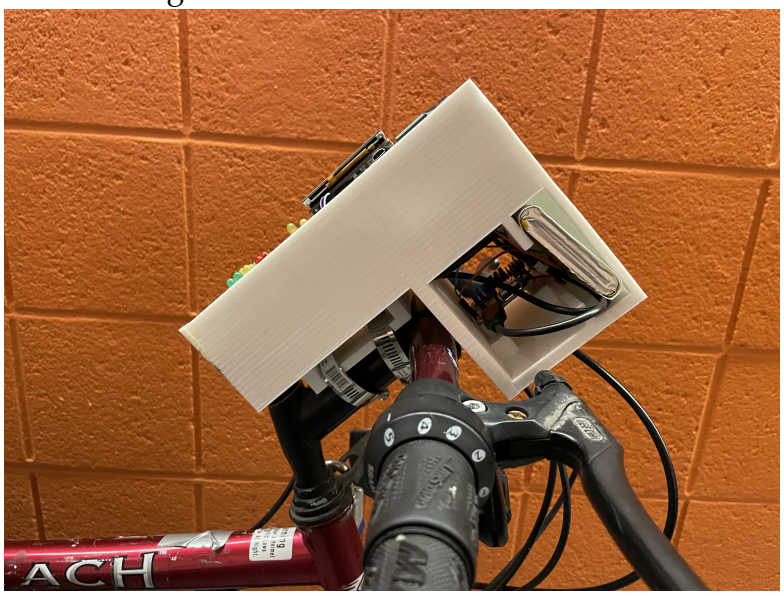

Figure 14: Side Profile of Dashboard on Bicycle

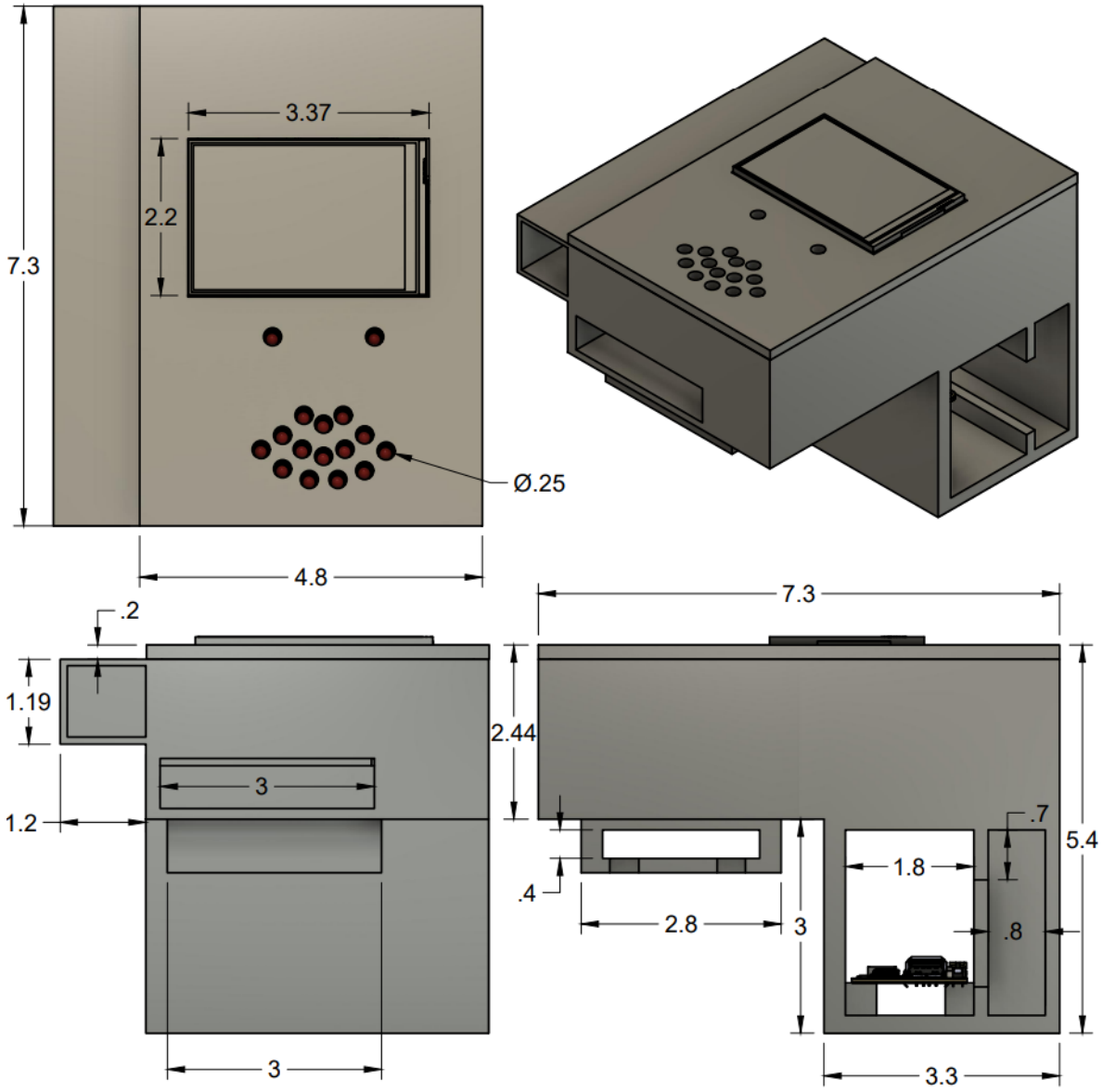

Figure 15: Fusion 360 CAD Drawing

Figure 16: Night Condition Object Detection Verification

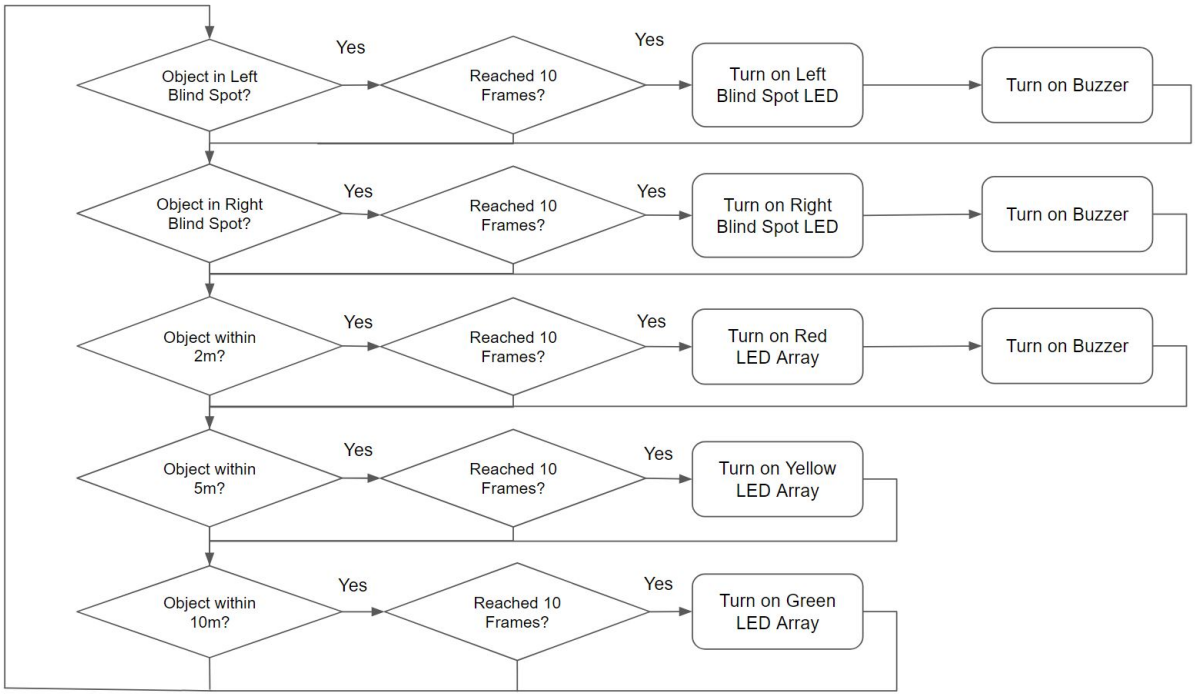

Figure 17: STM32 Microcontroller Software Flow Chart

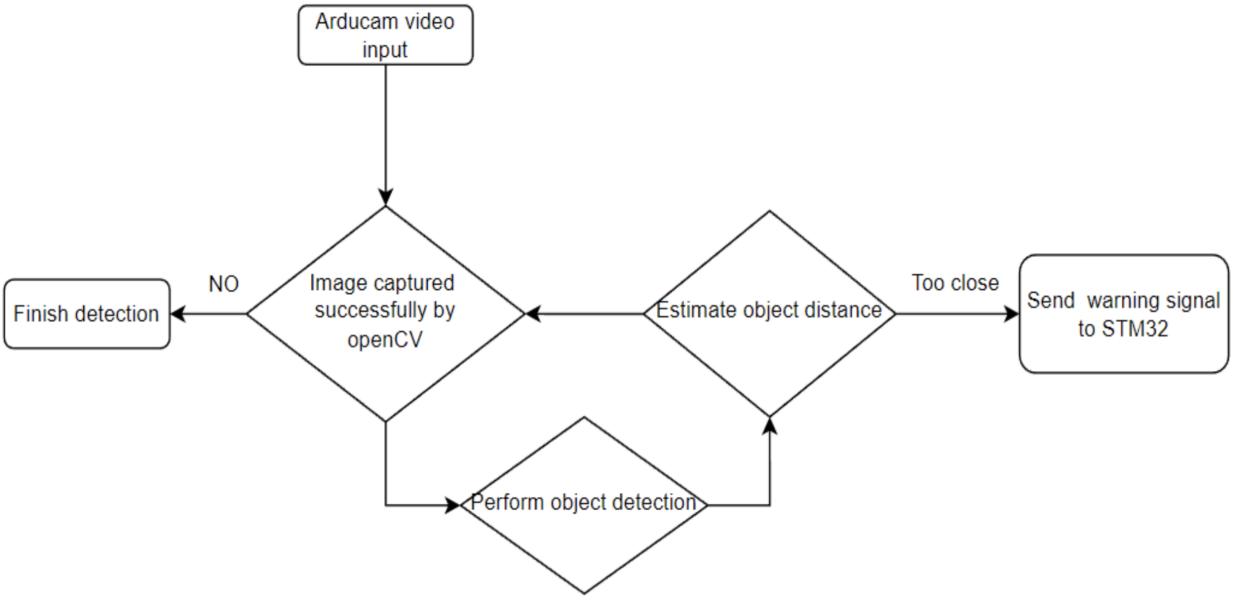

Figure 18: Raspberry Pi Object Detection Algorithm Flow Chart

B Cost and Schedule

B.1 Cost Analysis

The first calculation needed in order to determine the total cost of this project is the cost of all the components needed to complete the project. The sum of all the components listed in the figure below is $279.35. After this, we must calculate the labor costs associated with the project as well. The average starting salary for a computer engineering graduate is $105,352 [2]. With a 40 hour work week, and 52 weeks in a year, this salary equates to about $50.65/hr. The time from project approval to the end of this course is about 12 weeks, and the team will work about 8 hours a week. There are three members in our team. Therefore, the cost of the labor will be: $50.65/hr x 2.5 x 60 hours * 3 team members = $22792.50 This makes the total the sum of the component costs and the labor costs: $22792.5 + $279.35. The total cost of this project is $23,071.85.

Description	Manufacturer	Quantity	Cost	Link
1080P Day and Night Vision USB Camera	Arducam	1	34.99	Link
3.5″ LCD Touch Screen Display	Waveshare	1	19.80	Link
0.1″ 2x20-pin Strip Dual Male Header	Adafruit	1	0.95	Link
Angled Coiled Micro HDMI to HDMI Cable	Twozoh	1	12.79	Link
USB 3.0 Adapter 90 Degree Male to Female Coupler Connector	Oxsubor	1	6.99	Link
Raspberry Pi 4 Model B - 8 GB RAM	Adafruit	1	75.00	Link
Green 5mm LED	Adafruit	1	4.00	Link
Yellow 5mm LED	Adafruit	1	4.95	Link
Red 5mm LED	Adafruit	1	4.00	Link
3.7V 10000mAh 1165114 Lipo Battery Rechargeable Lithium Polymer ion Battery with JST Connector	AKZYTUE	1	25.99	Link
AmpRipper 3000	Kickstart Design	1	24.99	Link
STM32F103CBT6TR Microcontrollers	ST	1	9.19	Link
BUZZER MAGNETIC 3V 12MM TH	MallorySonalert Products	1	4.11	Link
3.3V Linear Voltage Regulator - NCP1117DT33G	onsemi	1	0.76	Link
N-Type MOSFET - DMG1012UW-7	Diodes Incorporated	6	0.38	Link
Custom-Made PCBs	JLCPCB	1	18.00	Link
SD/MicroSD Memory Card (8 GB SDHC)	Adafruit	1	9.95	Link
10K Ohm Resistors RC0805FR-0710KL	YAGEO	10	0.10	Link
68 Ohm Resistor	Stackpole Electronics	20	0.10	Link
15 Ohm Resistor	Stackpole Electronics	20	0.10	Link
0.1uF Capacitor	Taiyo Yuden	10	0.23	Link
10uF Capacitor	YAGEO	10	0.19	Link

Figure 19: List of Parts and Costs Needed for Project

B.2 Schedule

Date	Task	Person
10/3	Design Review, order PCB	Everyone
10/10	Buy all needed parts	Everyone
	Continue research on object detection software	Trisha and Jingdi
10/17	Receive PCB and start building	Jacob
	Begin STM setup	Jingdi
	Begin working on object detection prototype	Trisha
10/24	Finalize 3D printed enclosure	Jacob
10/31	First prototype object detection	Trisha
	Continue working on STM software	Jingdi
11/7	Test object detection	Trisha and Jingdi
11/14	Mock Demo	Everyone
	Finalize STM software and object detection	Trisha and Jingdi
	Work on presentation and report	Everyone
11/21	Fall break	-
11/28	Final demo, work on presentation and report	Everyone
12/5	Final presentation	Everyone

Figure 20: Schedule for Project

References

[1] "Bicycle Safety." (2022), [Online]. Available: https://www.cdc.gov/transportationsafety/bicycle/ (visited on 09/14/2022).

[2] "Garmin Varia™ RCT715." (), [Online]. Available: https://www.garmin.com/en-US/p/721258 (visited on 09/14/2022).

[3] "Open CV." (), [Online]. Available: https://opencv.org/ (visited on 09/28/2022).

[4] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[5] "A Diverse Driving Dataset for Heterogeneous Multitask Learning." (2022), [Online]. Available: https://www.bdd100k.com/ (visited on 12/07/2022).

[6] "Distance(webcam) Estimation with single-camera OpenCV-python." (2021), [Online]. Available: https://medium.com/mlearning-ai/distance-estimation-with-single-camera-opencv-python-298a96383c2b (visited on 12/07/2022).

[7] "Raspberry Pi 4 Model B." (2019), [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf (visited on 09/20/2022).

[8] "3.5inch HDMI LCD." (2022), [Online]. Available: https://www.waveshare.com/3.5inch-hdmi-lcd.htm (visited on 09/18/2022).

[9] "STM32F103x8." (2022), [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf (visited on 09/20/2022).

[10] "Super Bright White 5mm LED." (2022), [Online]. Available: https://www.adafruit.com/product/754?gclid=CjwKCAjwhNWZBhB_EiwAPzlhNmcI7ga2VqS-8DLy1G5JT_2It4aqkaQh0yR22mEY0RDuhuZCPVFu5xoCieIQAvD_BwE (visited on 09/20/2022).

[11] "PKM22EPPH2001-B0." (2022), [Online]. Available: https://www.digikey.com/en/products/detail/murata-electronics/PKM22EPPH2001-B0/1219322 (visited on 09/20/2022).

[12] "NCP1117DT33G." (2022), [Online]. Available: https://www.onsemi.com/pdf/datasheet/ncp1117-d.pdf (visited on 09/24/2022).

[13] "How much power does the Pi4B use? Power Measurements." (2019), [Online]. Available: https://raspi.tv/2019/how-much-power-does-the-pi4b-use-power-measurements (visited on 09/29/2022).

[14] "IEEE Code of Ethics." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 09/14/2022).

[15] "CDC streamlines COVID-19 guidance to help the public better protect themselves and understand their risk." (2022), [Online]. Available: https://www.cdc.gov/media/releases/2022/p0811-covid-guidance.html (visited on 09/14/2022).