Bicycle HUD Project

EE/CSE 475 Group 4:

Oliver Burton, Esther Huang, Nick Langhorne,

Katharine Lundblad, Max McKelvey

ELECTRICAL & COMPUTER ENGINEERING

UNIVERSITY of WASHINGTON

Dept. of Electrical and Computer Engineering, Box 352500

University of Washington

Seattle, Washington 98195-2500 U.S.A.

Sponsor:

University of Washington EE/CSE 475

Rev. 1 February 25, 2024

Rev. 2 March 11, 2024

Team, Roles, Responsibilities

In order to complete this project within 10 weeks our team made the decision to distribute work according to group members' interest. Each member was assigned a particular aspect of the project detailed below. All team members contributed to integrate each individual component into the final product.

- Oliver:
    - ESP32C3
    - OLED display
    - Helmet Design

- Esther/Kathy:
    - STM32F4Discovery
    - Hall Effect Sensor
    - Speed Sensor Design
    - Mounting Electronics/3D Printing

- Max:
    - Raspberry Pi 4 Model B
    - Computer Vision/Vehicle Detection
    - Google Coral ML Accelerator Interface

- Nick:
    - USB Connection (STM32F4Discovery → Raspberry Pi 4 Model B)
    - BLE Server/Client (Raspberry Pi 4 Model B → ESP32C3)

Product Requirements Document

Bicyclists face additional hazards when sharing roadways with motor-vehicles. Riders have low visibility in the periphery and rear fields of view which can make it difficult to ascertain potential threats especially from behind. We aim to help riders better detect hazards and avoid collision or injury.

The US Census Bureau's most recent estimates from 2021 show bicycling in the United States may be declining. Many of those who choose to bicycle live in urban areas where space allocated for cyclists is often shared with motor-vehicles. According to the National Highway Transportation Safety Administration 966 bicyclists were fatally injured in motor-vehicle accidents in 2021. Increasing safety from road hazards, like vehicles, may increase rates of cycling by preventing injury or death. As of November 2023, the Bicycle Helmet Safety Institute claims more than 200 localities in the United States have helmet-use laws which creates opportunities to bootstrap new safety mechanisms onto an existing safety device. DataHorizzon Research projects a compound annual growth rate of 8.9% in the bicycle accessory market over the next decade from USD 12.3 Billion in 2022 to USD 28.6 Billion by 2032. Couple safety concerns for bicyclists with a projected increase in the accessory market and there is potentially large demand for a product of this type.

To address the problem of safety for bicyclists on shared roadways we propose a heads up display, or HUD, attached to the bicyclist's helmet. The HUD will be a transparent display that can alert the cyclist if a motor-vehicle is approaching from the rear. Our goal is to create a system that uses a raspberry pi to communicate data wirelessly to a microcontroller attached to the HUD on the helmet. The intent is for this data to include rear motor-vehicle detection from a pi camera with ML computer vision capabilities as well as speed information gathered from an STM32F4Discovery which will be attached to a Hall effect sensor on the rear bicycle wheel. An ESP32C3 microcontroller attached to the helmet will be used to control the HUD, and a transparent OLED display will pass information to the rider.

Our market research found previous projects attempting to create a heads up display for a cyclist but most of these endeavors utilize glasses or some other stand alone peripheral unit instead of the rider's helmet. We found products that include a rear motor-vehicle sensor but none that couple a HUD with a camera utilizing modern computer vision libraries to detect hazards from behind. Many of these projects also have the additional constraint of adhering to a small form factor, a problem we hope to avoid by attaching our system to a helmet.

Realistic Constraints and Engineering Standards

We expect to deliver a product that can reliably collect and deliver information to the cyclist. In building this product, the project was constrained by our limited budget and time.

Given the time frame of 8 weeks to have our final product, a budget of $250 for materials, and the requirement to use the STM32 and Raspberry Pi, we constrained our project to support computer vision, a wireless notification system, and a speed sensor. Throughout the timeline of our project, we made use of recycled materials from the EE 475 lab, utilized borrowed materials from our team members (ie. Bicycle, Mounting Systems, 3D printer filament).

Specifically we discovered that with more time and resources we could improve our final product with:

1. Custom machine learning models
2. Custom hardware

By developing custom machine learning models built from custom datasets we expect that we could increase the efficiency and accuracy of vehicle detection on our device. Additionally, with custom hardware we expect that our final product will be far more compact, power efficient, and performant than our current prototype based on off-the-shelf components.

Our engineering team used some industry standard practices, specifically with our software development. We worked hard to write code following common style guidelines with clear levels of abstraction. Modern libraries were utilized where possible when integrating software written in Python, Arduino, and C.

The final product is expected to be capable of displaying an image with a refresh rate no less than 1 frame per second. This constraint will limit the reactivity of the system but transparent screens with higher refresh rates would increase product cost. Our team hopes to offset this limitation by increasing the reactivity of the speed sensor and computer vision components with hopes that any delay in data transfer can be effectively minimized. This is a rigid constraint as displays with lower refresh rates will greatly reduce the responsiveness and therefore safety provided by using the device.

The speed sensor is calibrated to accept sensor input with a minimum time interval of 5 ms. The typical bike wheel has a diameter of 622 mm making the velocity required to achieve the minimum time interval for sensor input well outside the range of human bicyclist velocities. In cases where the wheel is fully stopped we chose to incorporate a timer that checks the time since the last speed update that will adjust the speed to 0 once the wheel has stopped for a full 7 seconds. This time interval will bring the velocity below 1 km/h which our team chose to round to 0. An alternative choice would have been to update the velocity by estimating a decreasing velocity based on the lack of sensor input; however, our team decided the costs associated with processor utilization and battery efficiency made this option less desirable. These were negotiable choices our team made in order to best stick the previously stated design priorities.

The Raspberry Pi was found to have limitations regarding the number of frames it could capture and process in a given timeframe. It was decided that maximizing the processor time scheduled for the computer vision component of the program was necessary in order to increase accuracy and therefore potential safety provided by the system. In order to accomplish this we chose to use a high speed USB connection from the STM32 to the Raspberry Pi. The BLE connection between the ESP32 and Raspberry Pi were initially designed with the Pi as a GATT server and ESP32 as the client in order to minimize the Pi's processor usage as well. The processor of the ESP32 proved to have high variability when finding the Pi's advertised BLE service which is detailed in the experimental outcomes section. This constraint caused our team to choose to compromise on a priority to minimize processor utilization of the Raspberry Pi but was necessary to make the system usable in a reasonable period of time after system startup.

System Requirements Document

1. Functional Requirements
    a. Reliably collects and delivers information to the cyclist.
2. Size, Weight, and Cost Requirements
    a. The size and weight of the system can fit on an off-the-shelf bicycle without compromising the riding experience.
    b. The cost of all of the components is less than $500.
3. Mechanical Requirements
    a. The entire system is built with zero moving parts, and is impact resistant to the extent that the system could be subjected to during a ride.
4. Power Requirements
    a. The full system is powered entirely by batteries onboard the bicycle, and the system is fully user rechargeable.
5. Environmental Requirements
    a. The product should be designed to withstand various environmental conditions typically encountered during bicycling, including exposure to sunlight, rain, dust, and varying temperatures.
    b. The product should be designed to minimize its environmental impact, including considerations for recyclability and energy efficiency.
6. Communication and Interface Requirements
    a. The system should establish reliable wireless communication between the Raspberry Pi, microcontroller, and HUD.
    b. The interface between the microcontroller and HUD should be seamless, providing real-time display updates without latency.
    c. The system should support easy pairing and configuration with the user's smartphone or other devices for setup and customization.
7. Computation Requirements
    a. The Raspberry Pi should have sufficient computational power to process data from the Pi camera and perform real-time object detection using machine learning algorithms.
    b. The microcontroller should be capable of efficiently handling data transmission from the Raspberry Pi, and controlling the HUD display.
8. Software and Firmware Requirements
    a. The software should be developed using reliable programming languages and

frameworks suitable for embedded systems.
    b. Firmware updates should be easily deployable to address bugs, improve performance, or introduce new features.
    c. The system should incorporate robust error handling and fail-safe mechanisms to ensure continued functionality even in the event of software or firmware issues.
9. Precision and Accuracy Requirements
    a. Object detection algorithms should achieve high precision and accuracy in identifying motor vehicles approaching from the rear.
    b. Speed measurement using the Hall effect sensor should be precise and calibrated to accurately reflect the bicycle's velocity based on the wheel diameter.
    c. The HUD display should provide clear and accurate information to the cyclist without ambiguity.
10. Test and Validation Requirements
    a. The system should undergo rigorous testing under simulated riding conditions to evaluate its performance in detecting hazards and providing timely alerts.
    b. Validation testing should involve real-world scenarios, including varying speeds, weather conditions, and traffic densities.
    c. The system should be tested for reliability, durability, and adherence to specified requirements before being released to the market.
11. Safety Requirements
    a. The product should comply with relevant safety standards and regulations for electronic devices, including electromagnetic compatibility (EMC) and electrical safety.
    b. Components should be securely mounted to prevent detachment or interference with the rider's movement.
    c. The HUD display should not obstruct the rider's view or distract them from focusing on the road.
12. Materials Requirements
    a. The materials used in the construction of the system should be lightweight, durable, and non-toxic.
    b. Components should be selected for their resistance to impact and vibration to ensure long-term reliability.
    c. The product should be designed for easy disassembly and recycling at the end of its lifecycle to minimize environmental impact.

Project Schedule

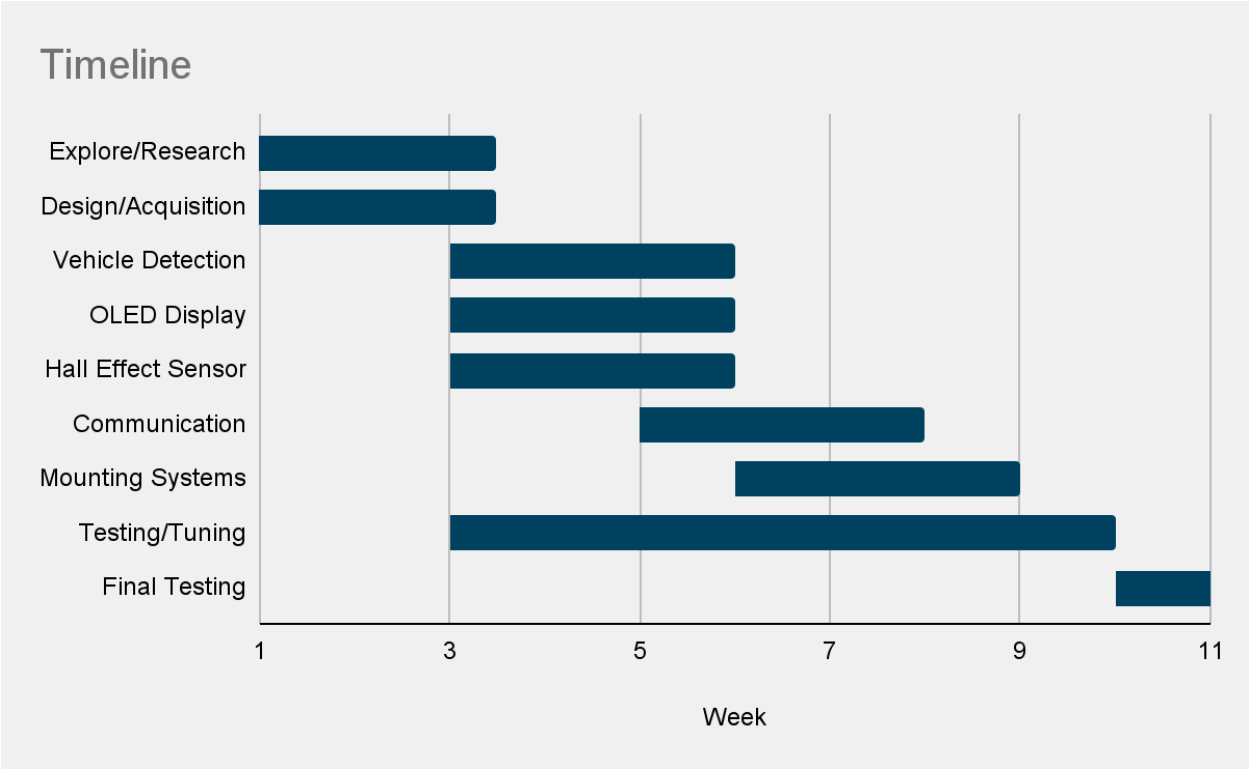

Figure 1: Gant chart showing the breakdown of project and a timeline for completion

Schedule

Week 1-3: Project Exploration/Design & Component Acquisition
Week 3-6: Computer Vision & Hall Effect Sensor & HUD
Week 5-8: BLE Server/Client & USB Communication
Week 6-7: Design Mounting Systems & 3D Printing
Week 7-9: Mount Components & Testing/Tuning
Week 10: Final Testing & Finishing Deliverables

Project Resources

Equipment	Cost
Raspberry Pi 4 Model B	$0 (provided)
Google Coral	$90
Pi Camera	$20
Power Supply	$36
STM32F4Discovery	$0 (provided)
Hall Effect Sensor Components	$30
ESP32C3	$10
OLED Display	$30
ESP32 Breakout Board & Power Supply	$15
3D Printer & Filament (Mounting)	$0
Total	$231

Figure 2: Total cost and breakdown of individual component costs for the project

Outline of Experiments

We plan to run an experiment by riding the bicycle and then driving a car past the bicycle. We will then evaluate the cyclist's experience with being notified before seeing the car pass them. We will repeat the experiment multiple times.

This experiment assesses the effectiveness of pre-notification systems for cyclist safety in traffic scenarios. The experiment involves a cyclist riding a bicycle, with a car passing them, while evaluating the cyclist's reaction to various notifications. The speed of the cyclist is tracked using a hall effect sensor, and the reaction time to speed notifications and notifications of approaching cars is measured using an OLED display on the cyclist's helmet. The experiment is repeated multiple times to gather sufficient data for analysis.

Materials and Methods:

1. Bicycle with hall effect sensor for speed tracking.

2. Car for passing the bicycle.

3. Helmet with OLED display for notifications.

5. Stopwatch for measuring reaction time.

Procedure:

1. Set up the bicycle with the hall effect sensor and the helmet with the OLED display.

2. Ensure the pre-notification system is operational and can deliver notifications to the cyclist.

3. Have the cyclist ride the bicycle at a constant speed of 15 km/h on a straight road.

4. At a distance of 30 meters behind the cyclist, start the car moving at a speed of 40 km/h to pass the cyclist.

5. Activate the pre-notification system to deliver a speed notification when the cyclist reaches 20 km/h and a car approaching notification when the car is 15 meters behind the cyclist.

6. Measure the reaction time of the cyclist to the speed notification and the car approaching notification using the stopwatch.

7. Repeat the experiment to gather sufficient data.

8. Analyze the data to assess the effectiveness of the pre-notification system.

The results of the experiment include the reaction times of the cyclist to the speed notification and the car approaching notification. These results will be analyzed to determine if the pre-notification system is effective in improving cyclist safety. Based on the results of the experiment, conclusions will be drawn regarding the effectiveness of pre-notification systems for cyclist safety in traffic scenarios. Recommendations for further research and development of such systems may also be provided.

Trial Designs

Overall System Design

In designing how the various aspects of this project would work together, we split it into three distinct parts: the speed measuring microcontroller, machine learning and sensor synthesis on the raspberry pi, and the heads-up display unit.

For the speed measurement, we decided to use the industry standard practice of adding a magnet to one of the bicycle wheel spokes and using a hall effect sensor to measure the frequency of the rotation of the wheel. Through this frequency measurement and the diameter of the bicycle wheel it was possible to compute the speed of the bicycle.

We decided to use an ESP32-C3 microcontroller for our HUD because of its low power consumption, internal bluetooth stack, and arduino abstraction layer. This combined with a rechargeable lithium battery and our display became the HUD for the cyclist.

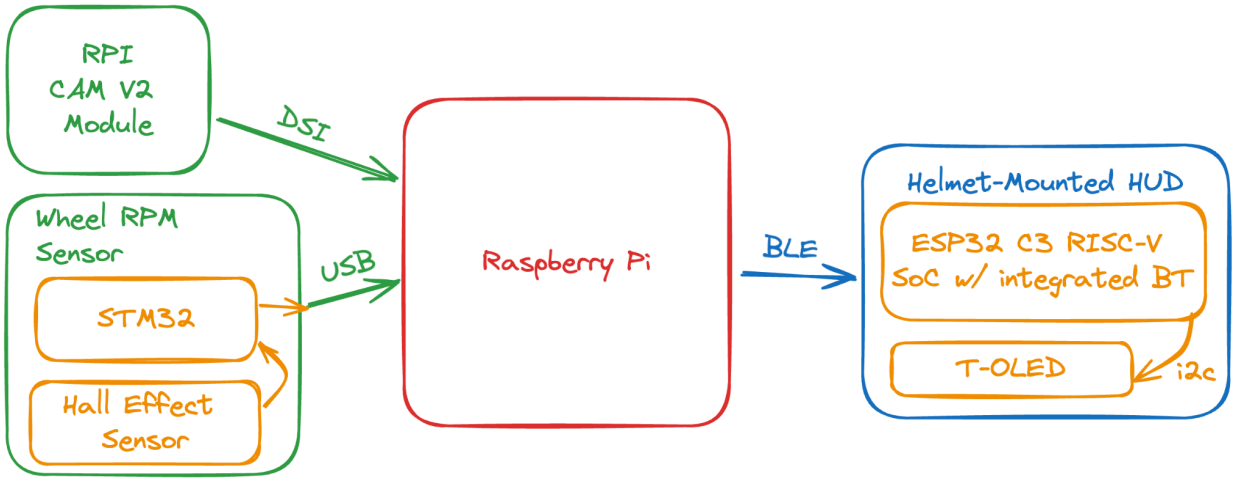

Figure 3: UML Diagram showing the overall design and communication between microcontrollers

Hall Effect Sensor Circuit

The design of a circuit for a 3-pin hall effect sensor to create a speed sensor using KiCAD modeling software was conducted in a first trial design via a breadboard. The next trial design was done using a Printed Circuit Board (PCB) in order to be mounted on the bike. In the final design, longer wires are connected to the output to reach the STM32 and the PCB was dipped in Epoxy Resin in order to withstand the environmental conditions of the outdoors when tested in the field. The circuit speed sensor circuit includes a 0.1 uF capacitor for noise filtering and a 10k resistor between power and output. This configuration ensures proper sensor operation and reduces noise in the output signal.

Using a PCB over a breadboard for this design was advantageous as it allowed for the creation of a modular and compact circuit. The circuit can be easily replicated and integrated into a larger system. KiCAD's tools for PCB layout and design facilitated the creation of a professional-grade PCB for attachment to a moving vehicle. In contrast, using a breadboard for the sensor would have been less modular and more prone to noise interference due to the lack of proper noise filtering components and the loose connections on the breadboard.

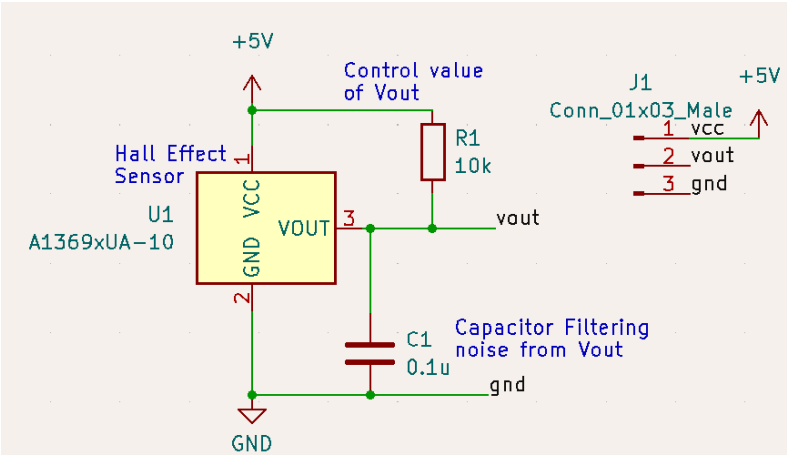

Figure 4: Hall Effect Sensor Circuit Schematic in KiCAD

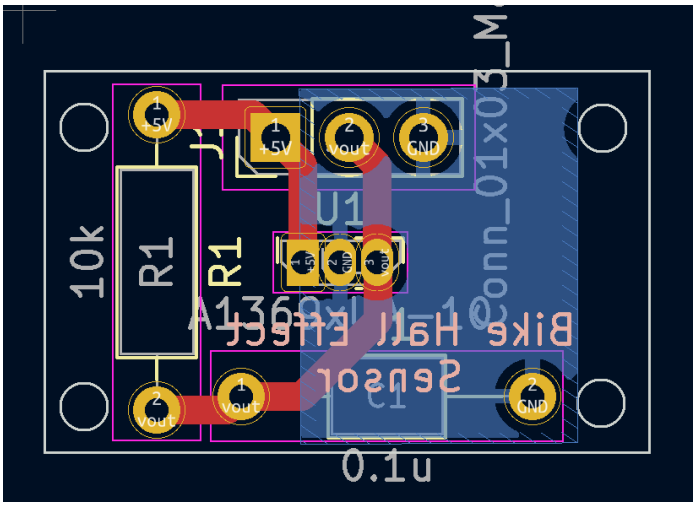

Figure 5: Hall Effect Speed Sensor Circuit Layout in KiCAD

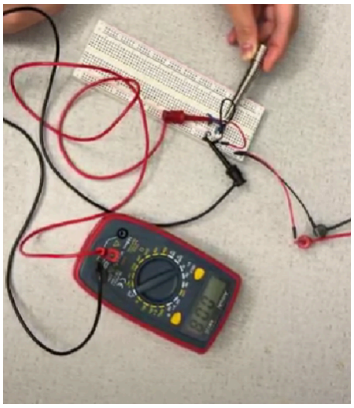

Figure 6: Hall Effect Speed Sensor First Trial Design

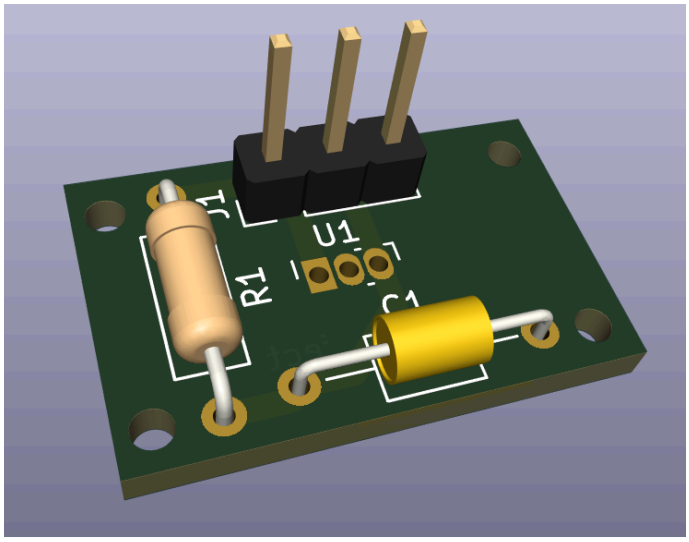

Figure 7: Hall Effect Speed Sensor PCB Second Trial Design

Mounting Systems

Two figures are presented detailing the design of 3D printed parts to mount a PCB onto a bike. The first design is intended for the front wheel attachment, specifically for the fork of the bike. The second design is tailored for the back wheel attachment. Upon evaluation, it was found that the second design was superior to the first. This conclusion was drawn based on several factors, notably its smaller size and closer proximity to where the STM32 and the Raspberry Pi are located. These attributes enhance the overall functionality and efficiency of the mounting system, making the second design more suitable for the intended purpose.

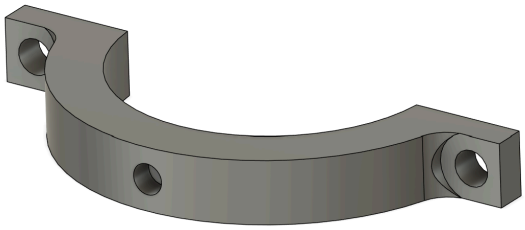

Figure 8: PCB Mounting 3D Model First Trial Design

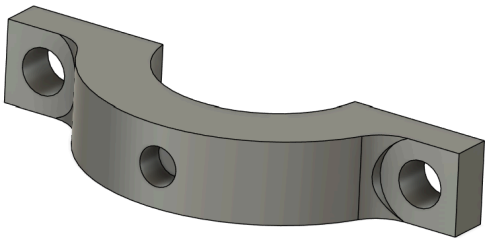

Figure 9: PCB Mounting 3D Model Second Trial Design

Figure 10: PCB Mount Printed Model Mounted to Bike

Machine Learning Model Selection and Acceleration

The ability of our product to ensure the safety of the cyclist relies on its ability to detect cars approaching from behind, which is based on object detection with off-the-shelf machine learning models. Through experimentation with different model architectures using different hardware, we were able to find a combination that enabled us to effectively detect cars and inform the cyclist of their approach.

Initially, we tested the YOLOv8 model architecture made by Ultralytics. We used their pretrained models on the COCO dataset, which included vehicle detection capabilities, and ran the models on a M2 Macbook Air. This model performed extremely well, even exceeding our expectations, but we ran into problems when attempting to run the same models on a Raspberry Pi 4 board. When running even the least computationally expensive model on the Raspberry Pi, we achieved fewer than one frame per second. This proved to us that we needed a machine learning hardware accelerator.

For hardware acceleration we decided on the Google Coral USB accelerator for its good performance per watt and compatibility with the Raspberry Pi. However, using the coral accelerator limited us to 8bit quantized tensorflow-lite models, which is different from the pytorch library that YOLOv8 is built on. This meant that we had to convert and quantize our YOLOv8 model, or find a new model entirely.

Initially we tried to convert and quantize our YOLOv8 model. After testing many different build options and platforms, we finally got the conversion to work. However, upon loading the model onto the Google Coral accelerator, it would crash. Our copious amount of debugging didn't save us from the limited documentation and tooling, so we went looking for edge-tpu native models.

Our search for new models brought us to find the EfficientDet-Lite pretrained models, trained on the same COCO dataset. These models worked natively with the Coral accelerator, so they ran with ease, but we still had to choose the right size model for our use case. There are multiple EfficientDet-Lite pretrained models, of increasing accuracy and size. We wanted to optimize for getting multiple frames per second while also having as high of an accuracy as possible. We landed on the EfficientDet-Lite2 model, which gave us over 5 frames per second while also having a 36.0% mAP. Through our model experimentation, we believe that this model is the best for our use case.

ESP32c3 Wiring and BLE Communication

There are two key requirements for the HUD system mounted to the cyclist's helmet. First, it must be a standalone system, and second, that data must be quickly processed and displayed on the transparent OLED display.

To address the first requirement, we selected the Esp32c3 microcontroller. The advantage of this device is that it is both able to run on a 3.7V rechargeable lithium battery, and has built-in Bluetooth Low Energy (BLE) capability. These two features allow the Esp32c3 to be mounted on

the helmet separate from the raspberry pi, with BLE communication used to transmit information about the speed of the bicycle and the location of cars. To communicate with BLE, we created a GATT server, with the Esp32c3 acting as the server and the raspberry pi as the client. The server has two characteristics, the speed and the car location, and both of these characteristics are editable. The raspberry pi, with its superior computing power compared to the Esp32c3 is able to quickly locate this server, and communicate changes in speed and car location by editing the characteristics. These characteristics are then read by the Esp32c3 and sent to the OLED display.

To communicate between the Esp32c3 and the OLED display, speed and simplicity were our top priorities. Therefore, we chose to implement SPI communication between the two devices. This approach was also chosen due to the breakout board for the OLED display being configured to primarily support SPI communication. The table below indicates the wiring between the Esp32c3 and the OLED.

Esp32c3	T-OLED Breakout Board	Purpose
3.3 V	3.3 V	Power
GND	GND	Ground
D0	D/C	Data/Command
D1	RST	Reset
D7	CS	Chip Select
D8	D0	Clock
D10	D1	Data (MOSI)

Figure 11: Connections between Esp32c3 and OLED Breakout board

Once wiring was complete, the U8g2 graphics library was used to send text and images to the display.

HUD Component Mounting

To mount the Esp32c3 and the 3.7V lithium battery, zip ties were routed through the holes in the helmet and tightened to secure the devices. Additionally, a slide switch was added between the battery and the Esp32c3 to act as an on/off switch. Turning the circuit on also acts to reset the Esp32c3 and restart the program. The connections between the battery and microcontroller were all soldered to ensure safety and avoid loose circuits. The final configuration is shown in figure 12.

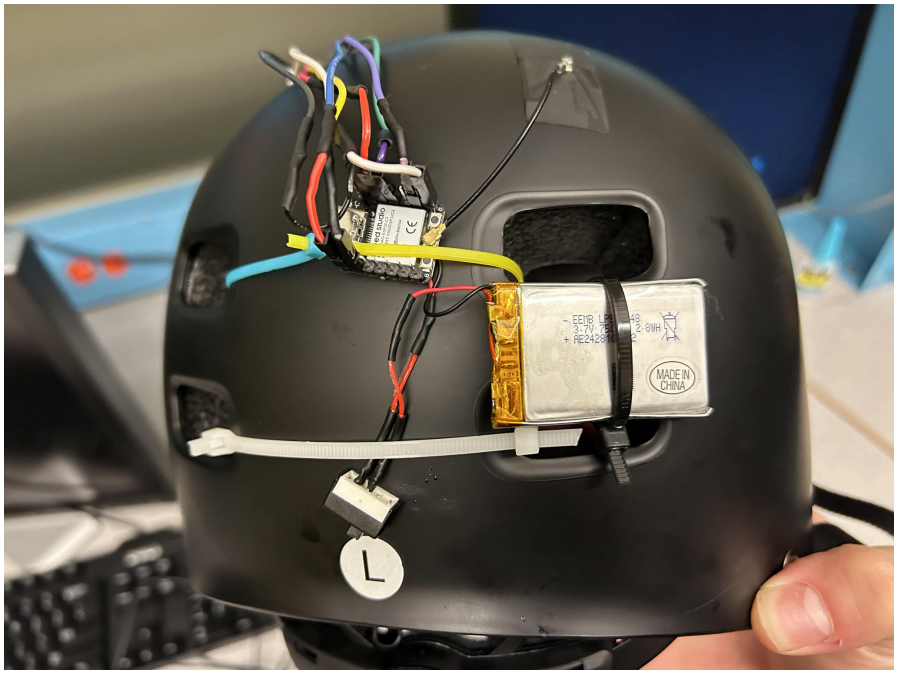

Figure 12: Esp32c3, battery, and switch mounted to the rear of the bicycle helmet.

To mount the OLED screen, a 3D printed mount was created to match the curvature of the helmet. Figure 13 shows a rendering of this mount. The mount was epoxied to the helmet, and the breakout board with the OLED attached was epoxied to the mount.

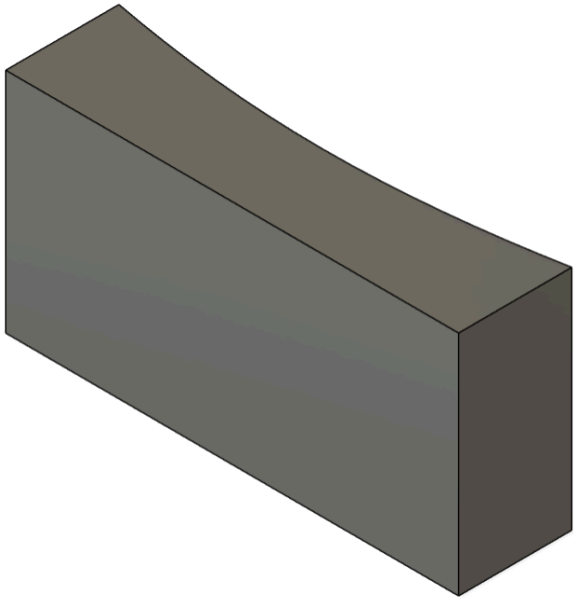

Figure 13: Rendering of OLED display mount.

We chose to position the screen just above and to the right of the right eye. This way, the screen is in the peripheral vision of the cyclist so they are aware of its changes, but not so in focus that it would distract the cyclist. Figure 14 pictures the OLED and breakout board epoxied to the front of the helmet.

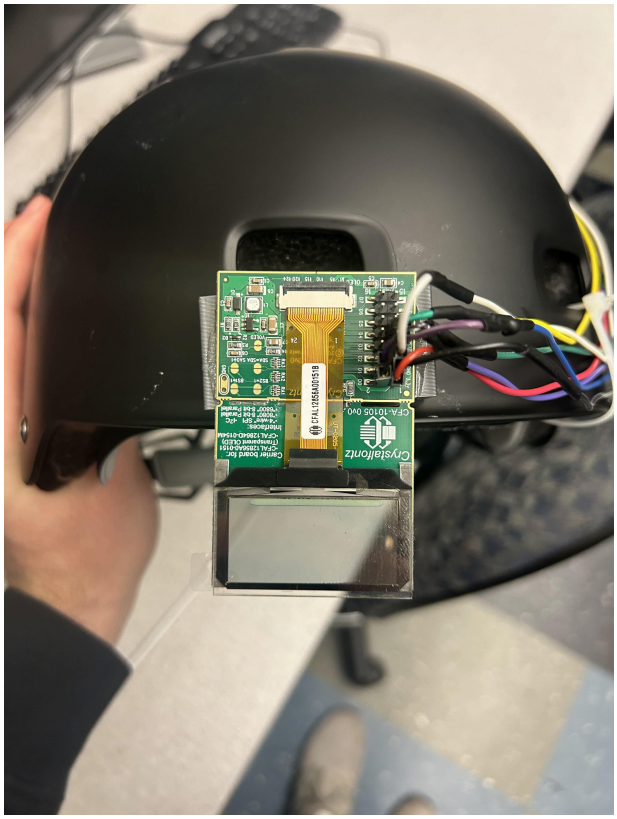

Figure 14: OLED display epoxied to the front of the helmet.

STM & Raspberry Pi USB Communication

The STM32 will need to communicate its captured speed information with the Raspberry Pi so that it can be conveyed to the rider through the ESP32's BLE service. For communicating millisecond scale data between the STM32 and Raspberry Pi the differential sampling and higher speed of a USB serial connection make it a suitable option.

The STM32F407G-DISC1 board does not have functionality for the ST-Link connection, used for uploading/debugging, to be utilized as a serial connection. There are no GPIO pins initially connected to the serial debugger port but based on the board's datasheet, fly wires can be used to connect pins and add serial functionality. We chose to use the board's mini USB connection to set up a virtual com port instead. This requires a second wired connection between the Raspberry Pi and STM32 as it still needs the ST-Link connection for power but avoids permanently adjusting the testing board.

Raspberry Pi & ESP32 BLE Communication

In the implementation of the GATT server for BLE communication, our initial instinct was to use the Raspberry Pi as the server device and the ESP32 as the client. While the BLE server was correctly broadcasted, the slow computing speed of the ESP32 meant that it could take in excess of 5 minutes for it to find and connect to the server, which we deemed was too long for use in real-world scenarios. By swapping the roles of the devices and using the ESP32 as the server instead of the client, we were able to leverage the Raspberry Pi's faster compute speeds to quickly connect the two devices.

The Raspberry Pi needs to convey information about hazards to the rider and speed data received from the STM32. This was done using the bluetooth low energy (BLE) module of the Raspberry Pi allowing it to function as a GATT (Generic ATTribute) client communicating with the ESP32 GATT server using the bluetooth Attribute Protocol.

The ESP32 was programmed as a GATT server using Arduino libraries that advertise a service with a randomly generated custom UUID. Using custom service UUIDs and characteristics allows different BLE services to be distinguished from one another. We chose to use two characteristics under the same service to relay speed and car status information to the ESP32.

The Raspberry Pi uses a message bus system, dbus, to facilitate interprocess communication using the Python programming language. Utilizing the system bus and a service daemon allows the GATT client application to communicate with the bluez system process which regulates BLE hardware on the Raspberry Pi. We used the Bleak library for Python which abstracts away some of the fine details of dbus and bluez leaving more time to study the Generic ATTribute protocol.

The Raspberry Pi is programmed as a GATT client that will scan for the custom service UUID advertised by the ESP32. Once the Raspberry Pi finds the appropriate service and characteristics it will pair with the device offering them. Both characteristics of the service offered by the ESP32 are writable and so will allow the Raspberry Pi to adjust the values for each characteristic associated with bike speed or car status. These characteristics can be used by the ESP32 to adjust what it displays to the rider via the OLED.

Final Assembly

Our final assembly consisted of the HUD mounted to the cyclist's helmet, the camera and speedometer mounted securely to the frame of the bicycle, and the control electronics contained within an enclosure mounted at the rear of the bicycle.

Figure 15: The Final Assembly on the Bicycle

It was important to us that the cyclist was unobstructed by the vehicle detection system, speedometer, and control hardware. Figure 15 above illustrates that we were able to contain the entire system in a compact package that didn't affect the handling of the bicycle in practice.

Experimental Outcomes

Our team defined reaction time as the time between stimulus and response. There are two stimuli in our trials; one that occurs when the car status changes and another when the bicycle velocity reaches 20 km/h. We gathered reaction time data for car status and bicycle speed indicators in ten testing trials.

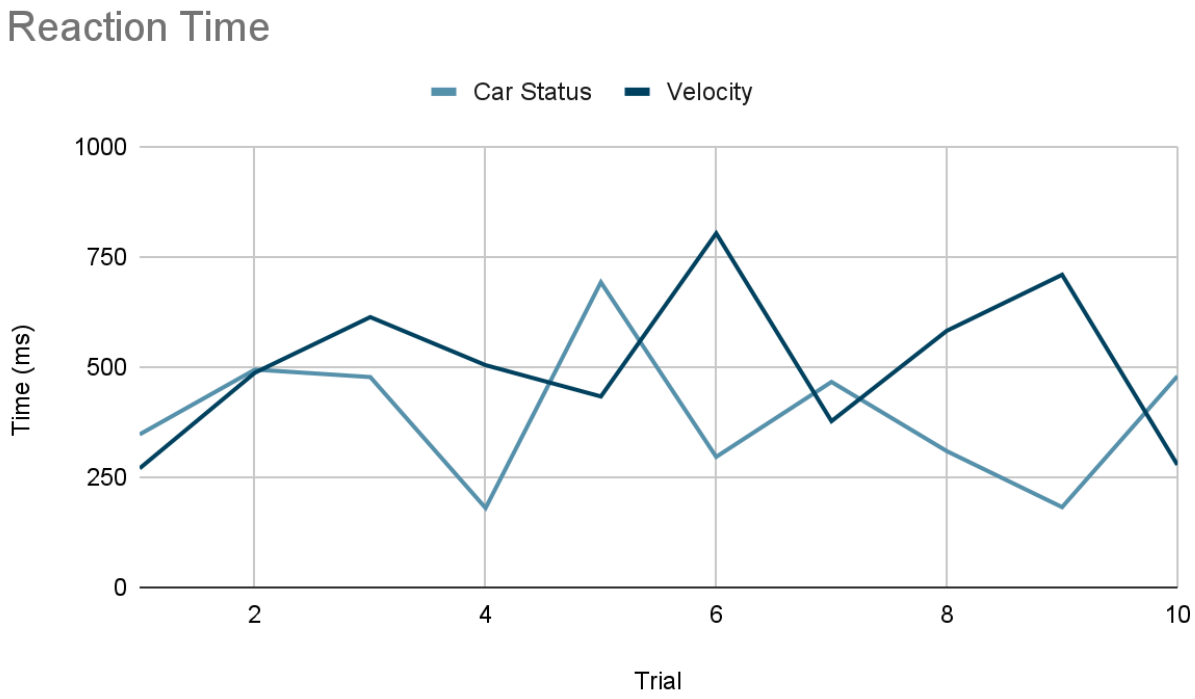

Figure 16: Graph showing average reaction time based on car status & bicycle velocity

Our team approximated the average human reaction time to visual stimuli and found that available research puts this value at roughly between 200 and 500 milliseconds depending on the studied task. The graph above shows that the user was able to identify car status changes quickly but typically needed more time to notice a change in car status. The users ability to identify when a velocity threshold had been reached was slightly worse than the users ability to detect car status changes. Still the average reaction time in all our trials was less than 1 second for both stimuli which should provide a degree of increased safety over not using the unit. This will be especially true in instances where a collision will occur after the reaction time interval when the user may be able to avoid potential harm.

Our team recognizes that more optimization should be done with regard to display placement and visibility. The reaction time associated with velocity is less concerning compared to the reaction time for changing car status as there are fewer safety concerns regarding velocity. In the case of car status queues, there is some potential to redesign the notification icons in order to decrease a user's reaction time to car status changes. A display with higher visibility in ambient light may also improve these reaction times and should be explored further in future studies.

Impact and Consequences

The successful development and implementation of the proposed bicycle HUD system carry significant potential impacts and consequences across various domains, ranging from individual safety to societal trends and environmental considerations.

By providing real-time alerts and information about approaching motor vehicles, the HUD system has the potential to substantially enhance the safety of cyclists sharing roadways with vehicular traffic. By mitigating the risks associated with low visibility and potential blind spots, the system can help prevent accidents, injuries, and fatalities among cyclists.

Increased safety for cyclists may lead to a positive shift in societal attitudes towards cycling as a viable mode of transportation. As concerns about safety are addressed, more individuals may feel encouraged to adopt cycling as a means of commuting, exercise, or recreation, leading to potential reductions in traffic congestion, air pollution, and reliance on motor vehicles.

The introduction of a novel bicycle HUD system represents an opportunity for market growth and innovation within the bicycle accessory industry. As the demand for safety-enhancing technologies continues to rise, the development of cutting-edge solutions like the proposed HUD system can drive product diversification, competition, and technological advancement within the market.

The adoption of bicycle safety technologies, such as the HUD system, may prompt regulatory bodies to consider updates to existing safety standards and regulations governing cycling equipment. Depending on the success and widespread adoption of the technology, policymakers may explore initiatives such as incentivizing or mandating the use of safety-enhancing devices like the HUD system, potentially influencing future legislation and enforcement efforts.

While the primary focus of the HUD system is on enhancing cyclist safety, its broader implications extend to environmental considerations. By encouraging more individuals to cycle and reducing the reliance on motor vehicles, the system has the potential to contribute to reductions in carbon emissions, noise pollution, and fossil fuel consumption, thereby supporting broader environmental conservation and sustainability goals.

The development and deployment of the HUD system necessitate advancements in various technological domains, including sensor technology, wireless communication, machine learning algorithms, and energy-efficient design. The successful integration of these technologies not only supports the functionality and efficacy of the HUD system but also contributes to the overall advancement of related fields and disciplines.

The impact and consequences of the proposed bicycle HUD system extend beyond individual safety to encompass societal, economic, regulatory, environmental, and technological dimensions. By addressing key challenges and leveraging emerging technologies, the system has the potential to foster safer, more sustainable, and technologically advanced cycling environments, with far-reaching implications for individuals, communities, and the broader transportation landscape.

Conclusions and Recommendations

The results found by our team over the course of this project highlight a need for effective safety tools for bicyclists navigating roadways with automobiles but also the difficulty that comes with designing them. We ran into several unexpected hurdles over the course of the project teaching us that even a thorough design process does not guarantee smooth implementation. Still, based on these results, our device has the ability to increase bicyclist safety especially when navigating vehicle congested roadways. Bicyclists can more readily react to their environment and potential hazards that would otherwise might be unavoidable, like vehicles approaching from the rear.

Individual components of the system also taught valuable lessons to our team. Hall effect sensors are an effective tool for determining the velocity of a rotating body so experience working with these circuits will likely be advantageous in any future embedded projects where capturing a velocity or cyclic action through a sensor is required. Sensors are the basis of many devices today with prolific use in medical, and other diagnostic industries. Edge cases exist when using sensors and mitigating the chance that erroneous data is captured should be a priority but only where it will not inhibit the overall function of the device. We learned through the process of designing and implementing our speed sensor that some edge cases are handled best in ways that decrease the accuracy of the displayed velocity which seems counter-intuitive. In our case a more accurate speed sensor comes with the cost of higher processor utilization that adversely affects our systems ability to determine when vehicles are approaching. These types of design tradeoffs are inherent to the design process and we chose safety over accuracy.

Our team's exploration of various deep learning models and specialized datasets to train them will certainly be an asset in the future as modern computer vision libraries are used increasingly today . Nearly every industry could see increased use of neural networks and machine learning moving forward. Specialized datasets exist but depending on use case may be more or less plug and play. Building specialized datasets with accurate ground truth classifications can be time consuming but classification can be made substantially more accurate when using neural networks that are specialized to a particular task. With more time this is an aspect of the project we would have explored more in order to increase the safety provided to the user.

The Internet of Things grows daily as a result of smaller, faster, and generally more capable microcontrollers. The most important experience this project provided our team was the opportunity to add to the space of embedded devices making up the Internet of Things. Regardless of any future projects we work on, it has been invaluable gaining hands-on experience in the design and operation of these embedded devices which will slowly make up more of the technological landscape we all share.

The final result of the project conforms to our team's design and intent but we discovered a few recommendations we would explore further if time permitted.

- We recommend that a product of this nature be developed ongoingly.
- We recommend exploring different display options that could increase rider visibility and potentially offer more space to convey information without obscuring the user's view.
- We recommend exploring microcontroller options beyond the STM32 for capturing speed as we found it to be more than was required for the task. The STM32 could also be used for auditory queues or other expanded audio capabilities including streaming music.
- We recommend an input mechanism, likely an app, for this system which would allow for custom adjustments to wheel diameter and unit of measure for velocity for various uses.

- We acknowledge that our product currently isn't necessarily as good as the current industry options, but has the potential to exceed them with much more development

References, Acknowledgements, and Intellectual Property

- [1] "Bluetooth® Technology for linux developers," Bluetooth® Technology Website, https://www.bluetooth.com/bluetooth-resources/bluetooth-for-linux/ (accessed Mar. 8, 2024).
- [2] K. Buchholtz and T. Burgess, "An evaluation of bicycle-specific agility and reaction times in mountain bikers and road cyclists", *SA J Sports Med*, vol. 32, no. 1, pp. 1–5, Sep. 2020.