BikeBuddy

Authors: Jack Wang, Jason Lu, Johnny Tian
Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—BikeBuddy is a system that is intended to improve safety for bicyclists by allowing them to have better situational awareness and giving other road users a better understanding of cyclists' intentions. This is accomplished through a system incorporating microwave radars that sense vehicles located behind and in front of the bicycle along with turn signals mounted on the rear of the bicycle. An embedded computer integrates information from the radars and provides visual alerts to the bicyclist on a centrally mounted display. Compared to a previous project, BikeBuddy brings 20% accuracy improvements in blind spot and collision detections.**

***Index Terms*—Bicycle, Embedded Systems, Radar, Raspberry Pi, Safety, User Interface, Vehicle Detection**

1 INTRODUCTION

Bicycling is a healthy and environmentally friendly way to travel. Cycling has become more popular over the past years. However, the safety of cyclists on the road remains a significant problem, especially in the congested metropolitan areas. Bike commuters need a better safety system to protect them when sharing the road with larger vehicles. A bicycle safety system with blind spot detection, rear and forward collision alerts, and turn signaling with automatic cancellation will improve bike commuter safety through increased cyclists' awareness of their surroundings and vehicle awareness of cyclists' intentions. The alerts can be displayed on a mounted screen, enhancing the situational awareness of the cyclists.

There are some existing technologies for bike safety, including mirrors, Garmin Varia RTL510, and wearable devices. A group in 2019 [7] did a similar project with LiDAR and a safety vest to improve bike safety, and another team in 2021 used microwave radars [3] for blind-spot detection and had turn signals. Our project will use radar for blind spot and collision detection, enabling all-weather operation. The screen display will also centralize the information to one place for cyclists, making the system easy to use. The overall goal of the system is to provide bike commuters with a more affordable option and improvements in detection accuracy to enhance their safety.

2 USE-CASE REQUIREMENTS

To fulfill the functionality of the system, the following requirements are proposed:

- Battery Life: The system should have more than 2 hours of endurance. According to data from Strava [2], the average commute distance in the U.S. is 8.3 miles (ca. 13 km). Assuming a biking speed of 15 mph (ca. 24 km/h), a 2-hour endurance allows bikers to use the system for a day of commute with buffer time.
- Detection Lead-Time: The system should give users enough time to react. The human response time is between 100 ms and 300 ms. Some studies have found that people need approximately 1.5 s response lead time to react to road hazards [21]. So, the system should give a warning at least 1.5 seconds before collision.
- Uptime: $\geq 99.999\%$.
- Confusion Matrix:
  - $\leq 40\%$ False Negative
  - $\leq 30\%$ False Positives

  The result from the 2019 project [7] had a false negative rate of 60% and a false positive rate of 41%. The confusion matrix is set this way so that our system performs at least 20% better than the previous version.
- Ruggedness Rating: IPX4. The device should work for commuters who commute in rainy conditions. IPX4 means no rating for protection against solids and "protection from sprays and splashing of water in all directions" [12].
- Ease of Installation: The system should be easy to install.
- Turn Signal Visibility: 100 feet of visibility in daytime, 500 feet visibility after sunset. Pennsylvania law requires a red rear reflector or light to be visible from 500 feet away between sunset and sunrise [9]. Although our turn signals do not fall under that law, it sets a reasonable baseline requirement for the nighttime visibility. The 100 feet (ca. 30 m) is a little more arbitrary, but it should be sufficient for drivers behind to see and react in time.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

For readability, we have placed the block diagram as Figure 8 at the end of this document.

At a high level, our system consists of two functional areas integrated into one system: (1) Vehicle detection and (2) turn signaling.

3.1 Vehicle Detection

The vehicle detection subsystem is responsible for detecting vehicles around the bicycle, determining which ones are potential risks for rear and forward collisions, and alerting the user to collision threats and vehicles in adjacent lanes (for blind spot monitoring).

The sensing portion consists of two radar modules, one mounted on the front of the bicycle and one mounted on the rear of the bicycle. The front module is responsible for tracking vehicles ahead for forward collision warnings, and the rear module monitors vehicles both directly behind for rear collision warnings and in the adjacent lanes for blind spot monitoring.

Both radar modules are Doppler radars which allows them to sense the speed and direction of objects in addition to distance and angle. Doppler radars work by measuring the time it takes for radar pulses to return along with sensing the phase shift in the return signal due to the movement of the object [22] [18].

One complexity is that we are using only a single radar to cover the entire rear area. This requires that we segment the detected radar targets according to the angle of the target in order to identify whether vehicles are in the same lane as us or in adjacent lanes. Specifically, we can imagine the radar detection zone as a semicircle centered on the radar emitter. Then, we can divide the semicircle into three slices based on the angle, 1 slice for the lane we're in and 1 slice each for the two lanes besides us. This is visualized in Fig. 1

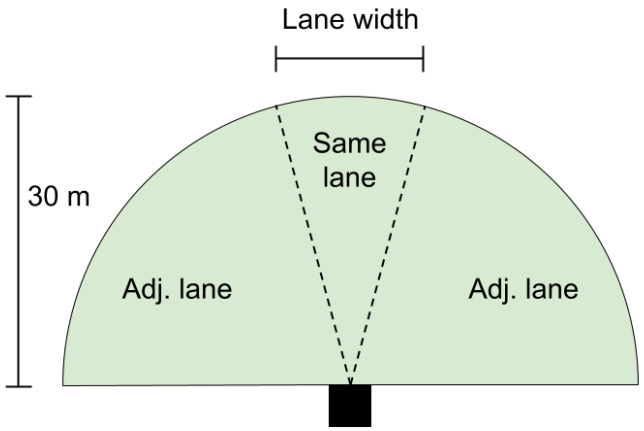

Figure 1: Dividing the rear radar detect zone into lanes; the black box is the radar and the green semicircle is the radar detection zone (not drawn to scale)

The radars will provide detected target information to the Raspberry Pi, which in turn will filter the data and identify vehicles based on the list of targets (since a single vehicle can show up as multiple radar targets [14]). Based on the list of vehicles and their respective velocities, angle from the bicycle, and distance, the Raspberry Pi can determine which vehicles are in the bicyclist's blind spot or risk having a collision.

To display warnings, we also have a small screen mounted on the handlebar of the bicycle that is connected to the Raspberry Pi. The display will flash colors in various parts of the screen to indicate danger and warnings.

3.1.1 Forward Collision Warning

The forward collision warning system works by constantly calculating the minimum distance required to stop before colliding with the vehicle in front and alerting the user if the distance is almost no longer sufficient. It is calculated based on [20]:

$$S = \frac{V^2}{254(f \pm G)} + \frac{V}{1.4} \tag{1}$$

where S is the stopping distance in m., V is the velocity in km/h, f is the coefficient of friction, and G is the grade that the bicycle is on in m/m.

This equation takes into account the time needed for bicyclists to react to alerts and brake. Note that we define "stopping" in this case to be bringing the *relative* velocity between the bicycle and the vehicle in front to 0. This is sufficient as a relative velocity of 0 will prevent collision even if both have non-zero absolute speeds, so for velocity V we will use the relative velocity between the vehicle and bicycle instead of absolute velocity. The relative velocity of the vehicle in front is obtained from the front radar system.

The coefficient of friction f varies with weather conditions and the specific bicycle, but for the MVP we will assume dry conditions and set $f = 0.32$ which is a typical value for bicycles under dry conditions from [20]..

For the grade G, we will currently assume the grade is simply 0 (e.g., flat). This assumption is safe if the bicyclist is ascending a hill as we will overestimate the necessary stopping distance, but on descents, this equation will underestimate the stopping distance required and warn the user too late to prevent a collision. However, for the sake of simplicity in building the MVP we will use this assumed value. Further work could consider adding an accelerometer to calculate the grade.

In order to determine whether to warn the user, we first add the minimum required stopping distance as calculated using (1) to some margin. Then, we compare it to the measured distance to the vehicle in front. If the measured distance is less than what we calculated to be the minimum stopping distance (+ margin), then we will alert the user. The margin will help ensure that we aren't alerting the user at the last possible moment to brake safely, although (1) already takes reaction time into account.

3.1.2 Rear Collision Warning

The rear collision warning detection is conceptually similar to the forward collision warning, where we compare the distance required for the vehicle directly behind to slow down to a relative velocity of 0 to the actual remaining distance between the vehicle and bicycle. However, instead of a formula for calculating minimum stopping distance, we can use a table of vehicle stopping distances such as the one from [26].

We will use the relative velocity of the vehicle behind as detected by the car as the lookup value into the table (with linear interpolation between entries) to get the stopping distance. If the minimum stopping distance required is no longer sufficient for the vehicle behind to stop, we will trigger an alarm so that the user can attempt to move out of the way or speed up/start moving.

Using this table, we will probably underestimate the stopping distance required under slippery conditions, for larger vehicles such as semi-trucks, or for vehicles that otherwise have poor braking performance. For simplicity, however, we will use this for the MVP.

3.1.3 Blind Spot Monitoring

For blind spot monitoring, we will look for simply the presence of vehicles in the adjacent lanes, meaning targets that are beyond a certain angle from the centerline.

We plan to warn the user through visual indications if there is a car detected in adjacent lanes regardless of distance.

3.2 Turn Signalling

The turn signalling mechanism consists of the turn signal buttons that are located in the front of the bicycle, the auto-cancellation system located in the middle of the bicycle, along with the actual turn lights located in the back of the bicycle.

For the turn signal controls, we will have two momentary switches mounted on the handlebar of the bicycle. Pressing on the momentary switch will close a circuit to cause a voltage change that is detectable by the Raspberry Pi, causing it to turn on the turn signals. The turn signals can then be turned off again by pressing either of the momentary switches.

Additionally, for user convenience, the system will also include an auto-cancellation system that is intended to automatically deactivate the turn signals once a turn is completed, much like a car would. This is implemented using a compass mounted on the center of the bicycle.

The compass will give us the heading of the bicycle, and the idea is that we'll record the starting heading of the bicycle when the turn signal is first activated. Then, once the heading of the bicycle relative to the starting heading exceeds a certain threshold, we'll turn off the turn signals (with a possible additional delay once the threshold is crossed).

The turn signals themselves are simply LEDs mounted in the rear of the bicycle, with transistors controlled by the Raspberry Pi.

4 DESIGN REQUIREMENTS

4.1 Battery Capacity and Power Limitations

The users would like a solution that will last long enough for their daily commute, so this requires that the design utilize low-power consumption modules with sufficient energy supply. The total power consumption of the system should be less than 30 W. The Raspberry Pi runs off a 5V/3A supply, which gives us a worst-case power usage of 15W for the Pi itself and whatever peripherals are connected to it:

$$Power = I * V = 3A * 5V = 15W \tag{2}$$

Assuming in the worst case that the same amount of power is used for other miscellaneous components like the turn signals, we additionally allocate 15 W for everything else, which gives us the total 30 W:

$$PowerConservative = 15 + 15 = 30W \tag{3}$$

Therefore, we need a power bank that can support at least 3A outputs on two ports with a capacity of at least 60W-h to power the system for at least 2 hours. Furthermore, the entire power draw of the system must not exceed 30 W.

4.2 Simultaneous Targets

The rear radar needs to be capable of tracking at least three separate vehicles in three different lanes, one for a car directly behind the bicycle and one car each in the adjacent lanes. A typical car lane width in the U.S. is 9-12 feet for urban roads [23]. So, the minimum distance resolution of the radar should be more than 9 feet (2.74 meters).

4.3 Data Transmission

The radar must be capable of communicating with a baud rate of 115200 for UART communication with the Raspberry Pi.

4.4 Radar Accuracy

The radar shall provide a minimum accuracy of $\pm 10\,\%$ for all quantitative measurements (speed, distance, angle).

For velocity, it shall identify the correct direction of the target $\geq 95\,\%$ of the time.

4.5 Radar Data Update Frequency

The radar shall provide location updates for all detected vehicles at a minimum frequency of 10 Hz.

4.6 Detection Distance

The system should have a minimum detection range of 14 meters. Section 2 describes that the system should provide 1.5 s response lead time for blind spot collision.

Assuming a speed differential of 15 mph between the car and the bicycle:

$$15\,\text{mi/h} = 24\,\text{km/h} * \frac{1000\,\text{m}}{1\,\text{km}} * \frac{1\,\text{h}}{3600\,\text{s}} = 6.67\,\text{m/s} \quad (4)$$

$$6.67\,\text{m/s} * 1.5\,\text{s} = 10\,\text{m} \quad (5)$$

Therefore, the system needs to be able to detect objects that are 10 m away to fulfill the requirement. Adding 30% buffer, we need to ensure the detection range of the radar will be at least 14 m.

5 DESIGN TRADE STUDIES

5.1 UI Framework

Here are some options that we considered:

1. Electron
2. Qt
3. Gtk
4. Tauri - this was added during testing because of the high memory usage of Electron, and we wanted something conceptually similar to it but that had hopefully lower resource usage.

Here are the criteria that we used to evaluate the frameworks. Note that not every factor was weighted equally:

1. **Cross-platform** – Can it run on both MacOS and Linux?
2. **Language/Framework/Tooling Familiarity** - How familiar are already with the language, frameworks, and tools?
3. **Adoption** – Who are the major users of it?
4. **Baseline resource usage** – How much CPU usage and RAM do we use to render roughly the same things?
5. **Cross-compileable** – Can we easily build packages for the RPi 4 from a host computer?

Sample "Hello world" apps were created in all 4 frameworks and their memory and CPU usage were measured. Testing was conducted on a M1 MacBook Air running MacOS Sonoma (14.2.1) with 16 GB RAM.

The memory usage was obtained by using Activity Monitor (like Task Manager, except for MacOS) and summing up the relevant threads' memory usage - e.g., for Electron we have multiple threads running around so we need to account for all of them. For calculating CPU usage, we used an admittedly lot less scientific method where we just watched the CPU usage of the threads and a value as the max only if the CPU usage repeatedly hit it – e.g., hitting 1% CPU usage just once wouldn't count, only if the usage fluctuated up to 1% a few times.

Table 1 contains a summary of our testing, and a full version with citations can be found here.

For Tauri, we were unable to find any major applications that we knew of that used it, which made us a little more hesitant about whether it would be as production-ready and usable for real-world apps as the other three. Therefore, we ruled out considering it further.

In the end, the decision came down to performance overhead vs. ease of development. Given that we were most familiar with HTML/CSS/JavaScript for building UIs and were comfortable using HTML/CSS/JavaScript (equally so with C, more than C++), we'd rather use Electron despite the performance overhead simply because the lower learning curve is more important to us.

5.2 Embedded System

Another decision we had to make was the choice of the device to use as the compute module. Our final two contenders were the Raspberry Pi 4 and 5. Some reasons why we were focusing on Raspberry Pis is because they are available in the ECE inventory (hence zero cost for us) and have good documentation and resources.

The Raspberry Pi 5 features significantly better performance compared to the Raspberry Pi 4 (2 - 3x on many benchmarks according to [27]). However, in exchange, the peak power draw for the Pi 5 is 12 W vs. only 8 W for Pi 4, and in one set of benchmarks the Pi 5 drew nearly double power at idle and under load [5]. This can be an issue for us as battery life endurance is one of the requirements, and having excessive power draw can make hitting that difficult.

There were a few other minor considerations such as the Pi 5 running hotter [5] and the USB max current being limited on the Pi 5 to 600 mA compared to 1.2 A limit on the Pi 4 using the same 3A charger [5] [16]. However, in the end, the main tradeoff was between performance and power draw.

We felt that the Pi 4 would have enough performance to run our workload, so the additional performance was not necessary. Therefore, we settled on the Pi 4.

5.3 Radar

Many sensors can provide distance information between two objects, aiding blind spot detection and collision detection. Based on the use-case requirements described in 2, we are interested in the following performance of a sensor:

- All-weather operation
- Line of Sight Requirements
- Range

Table 1: Comparison of various UI frameworks according to our metrics

	Cross-Platform	**Language + Framework Familiarity**	**Adoption**	**Baseline Resource Usage (Memory/CPU)**	**Cross Compilable**
Electron	Yes	We are familiar with Javascript, HTML, CSS, and some JavaScript UI frameworks, but otherwise no prior experience with Electron	Used by many major apps, like 1Password, VSCode, Slack, and Discord	106 MB/0.1%	Yes
Qt	Yes	We are less familiar with C++ and Python, no prior experience with Qt. It seems as if you don't need to use Qt tools, but it's recommended and we aren't familiar with it	VLC , FreeCAD , many KDE apps	19.8 MB/0.0%	Seems to be possible but very complicated
Gtk	Yes	We are familiar with C, but otherwise no prior experience with Gtk	Firefox , GIMP, Transmission, GNOME desktop	22.6 MB/0.0%	Seems to be possible but complicated
Tauri	Yes	We are familiar with Javascript, HTML, CSS, and some JavaScript UI frameworks, but otherwise no prior experience with Tauri	No known major apps using it	74.6 MB/0.0%	Currently not possible

- Accuracy

The following sensors are considered:

- Radar
- LiDAR
- Ultrasonic Sensor

In order to support the usage of the system in rainy conditions, the detection sensor will be covered by a waterproofed cage. This means that we would require a sensor that cannot be affected by ambient conditions. Due to the enclosed cage, the sensor should also not be limited by direct line of sight. Radar has advantages over LiDAR as radar signals will be able to better penetrate weather, allowing detection in bad weather conditions [19] [10]. Furthermore, radar can penetrate solid materials, whereas it is unlikely that LiDAR can pass through opaque materials, which limits our enclosure material choices.

The range of the detection sensors needs to be greater than 10 m per the use-case requirements. Both LiDAR and radar can achieve this range. Ultrasonic sensors, although they might be more cost-effective, typically have a shorter range than what we require.

While accuracy is comparable, LiDAR typically has very high precision in measuring distance, especially in the short to medium range, whereas radar has a slightly lower precision [10].

Combining all the factors above, radar stands out as the most ideal detection sensor for this system. We have chosen K-LD7 radar as the module because it would provide direct serial output, reducing the workload of signal processing. However, one thing to note is that the K-LD7 will not register a target if the radar and the target are relatively stationary [24]. For the specific case of bike blindspot detection, it is very unlikely that the vehicle behind the bike will travel at the same speed as the bicycle. So, it is considered reasonable to sacrifice such drawback for the ability to detect in all weather conditions.

5.4 Turn signal and Auto-Cancellation

Since we're planning to implement turn signals that users can control with auto-cancellation, purchasing off-the-shelf flashing lights from Amazon or other sources is not feasible. The uncertainty of the wiring diagrams and the complexity of figuring it out is comparable to designing entirely new turn signals. Consequently, we explored motorcycle signals, often sold as LED chunks. However, our research revealed that common lightweight motorcycle turn signal lights operate on a 12V power supply, typical for motorcycles. Introducing motorcycle LEDs and a DC-to-DC converter adds another layer of complexity. Therefore, we've opted to purchase bright yellow LED chips to craft our custom turn signals.

Regarding the turn switch, our initial plan was to buy

motorcycle turn switches, primarily for easy momentary activation and convenient handlebar mounting. However, an aftermarket motorcycle switch we acquired from Amazon turned out to not be momentary, meaning it doesn't return to neutral after activation, causing it incompatible with the auto-cancellation feature. As a solution, we're purchasing waterproof momentary switches and customizing our turn switches.

For the auto-cancellation feature, our original idea was to use a potentiometer as a handlebar position sensor, considering the option of utilizing gearing to amplify turning for more drastic readings. However, through observations, we noticed that during bike rides, the handlebar rarely turns more than 5 degrees on each side when making a turn. Additionally, the maximum amplification achievable through gearing would be at most three times, increasing the required torque. This compounds the existing issue of high-resistance potentiometers. Therefore, our current plan involves using a magnetometer mounted on the Raspberry Pi to measure the bike's heading and implement auto-cancellation based on the entire bike's dynamics.

6 SYSTEM IMPLEMENTATION

6.1 Turn Signal

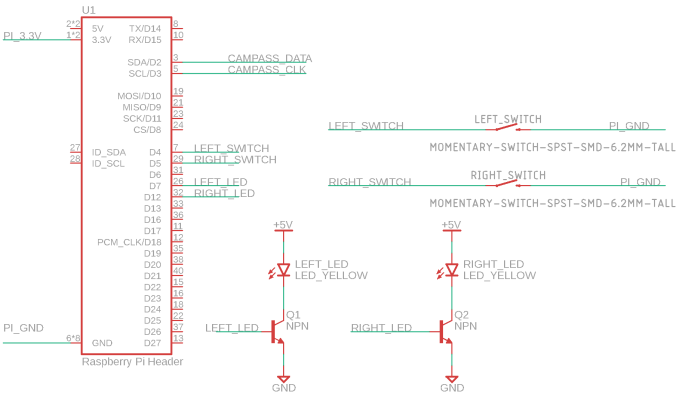

Figure 2: Turn Signal Schematic[4][11][6][25]

6.1.1 Turn Signal Lighting

Our solution integrates numerous 5V 1W LEDs (Hyunduo 5V 1W 200lm Yellow), each emitting 200 lumens of vibrant yellow light. This configuration ensures robust signaling capabilities both at the front and back of the bike, significantly improving visibility for bikers and making them easily noticeable by other participants on the road. To control the power supply, we will send a lighting signal to an N MOSFET (BS170 from 220 lab), regulating the power on and off from a separate 5V 3A power source. As we haven't had the chance to test the brightness of the LEDs, our plan is to use 3 LEDs on each side to provide ample lighting. Including both the front and back, this results in a 1.2A output for each side. Since this exceeds the maximum current output for Raspberry Pi GPIO pins, we will use the combination of MOSFET and a separate power source to illuminate the LEDs.

6.1.2 Auto-Cancelling

Elevating the user experience, we're implementing auto-cancellation for bikers. Utilizing a magnetometer (Adafruit MMC5603), we precisely detect changes in the bike's heading, providing an accurate interpretation of user bike dynamics. This enables us to intelligently determine when to deactivate the turn signal. The magnetometer will be able to communicate with PI through I2C.

Our current strategy involves turning off the signal if the bike's turn exceeds 60 degrees from its heading at the time of the user's signal activation. This sophisticated system ensures smooth and efficient signal management for bikers. However, given that the optimal threshold for the auto-cancellation feature is contingent on specific user experiences and turning conditions, we are currently testing to validate whether 60 degrees is an ideal cutoff.

Furthermore, we are exploring the possibility of incorporating the rate of angle change speed as an additional parameter for determining when to stop the turn signal. The specific implementations will be fine-tuned through testing to best cater to user needs."

6.1.3 User Control

The user will control the turn signals through two custom-combined waterproof buttons (Twidec Momentary Push Button), communicating with the Raspberry Pi. The buttons will communicate with the PI through GPIO pins with pull up resistors. Upon the initial press, the system activates, and the Pi records the current heading using the magnetometer. If the bike's orientation deviates by 60 degrees from the recorded direction during the turn, the turn signal will automatically deactivate. In cases where the user intends to change lanes using the turn signal, they can press either of the two buttons to cancel the signal.

6.2 Radar

6.2.1 Hardware

As mentioned in 3, the radar used for blind spot detection will be mounted in the back of the bike, facing the rear. The radar used for collision detection will be mounted in the front of the bike, facing front. Both radars will be enclosed by a waterproof box, which is to ensure the all-operation requirements. The radar will be powered by the 5V output from the Raspberry Pi. The Tx and the Rx pins of the radar will be connected to the UART Tx and Rx pins of the Raspberry Pi. All the wiring of the radar will go between it and the Raspberry Pi.

6.2.2 Software

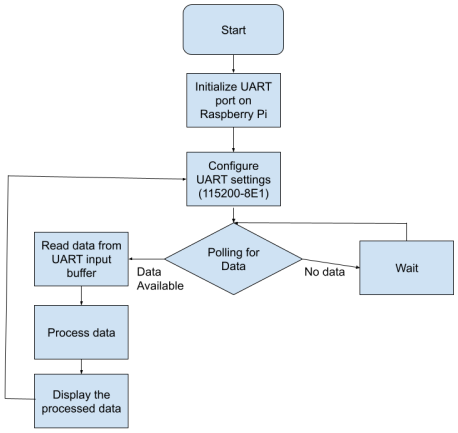

Figure 3: Flow Chart

The radar sensor will communicate with the Raspberry Pi over UART. We will open the UART port and use the protocol 115200-8E1. A Python driver is available to interface with the K-LD7 radar module. This helps us to tune different parameters. 3 shows the flow chart of communication. 22 configurable radar parameters can be tuned to obtain better detection results. We are particularly interested in distance range parameters to adjust the range for our detection purpose. If the detection detects a return of a distance less than the threshold (10 m), then it will trigger the warning system on our user interface, alerting the biker.

6.3 User Interface Software

Our Raspberry Pi runs off a standard Raspberry Pi OS (which is based on Debian Linux [15]) installation, created using the Raspberry Pi Imager.

Using Linux instead of an RTOS or other custom operating system gives us access to pre-written drivers for the various peripherals (especially the display) and access to more traditional desktop graphical frameworks. This simplifies bring up significantly.

The user interface itself is built using Electron which "is a framework for building desktop applications using JavaScript, HTML, and CSS ... that work on Windows, macOS, and Linux — no native development experience required" [8].

A mockup of the UI designed in Figma is shown in Fig. 4

6.4 Enclosure and installation

We plan to use a combination of poly-carbonate, PLA/ABS, or resin printing materials for our project enclosures. The majority of the components will be constructed using 3D printing methods, as it allows for a greater variety of shapes compared to simple laser cutting. For the screen casing, we intend to use clear acrylics to provide users with a clear view.

Regarding the bike attachment, we are considering employing a bolt clamping technique, similar to the one depicted in Fig. 5. We've opted for 2 bolts on both sides due to the lack of elasticity in many 3D printed materials. For wiring, we will create a sandwich casing for the components and wires, applying waterproof sealant at the connecting points.

Figure 5: Image from Amazon

For the power bank opening, we plan to use a design similar to one of Johnny's previous projects, as shown in the figure below

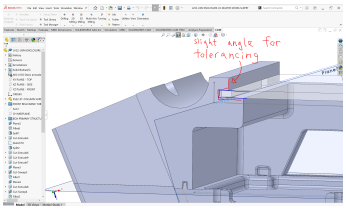

Figure 6: Previous waterproof enclosure design Johnny did for CMR

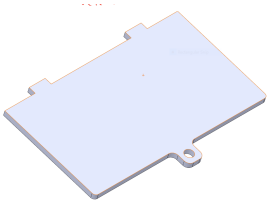

Figure 7: Previous waterproof enclosure design Johnny did for CMR

Finally, for the enclosure of the radar, we need to be very careful with the material and thickness covering the radar. Our current best plan is to test 3D printed ABS since it can provide the best stability for mounting, but it may result in inconsistency in density, potentially causing inaccurate readings.

The second option is to use a combination of laser-cut polycarbonate and a 3D printed casing. This approach can ensure a good density for the radome. However, the concern lies in maintaining a stable connection between them, as any vibration between the radar and the radome could lead to misreadings [1]. Therefore, achieving a tight and snug design is crucial.

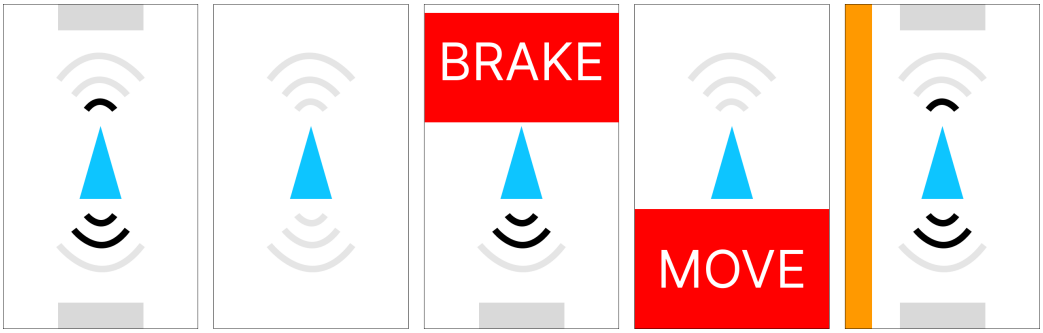

Figure 4: Mockup of the interface displaying various scenarios. From left to right: (1) Vehicles detected in front and behind. (2) No vehicles detected. (3) Forward collision warning. (4). Rear collision warning. (5) Vehicle in left lane

7 TEST, VERIFICATION AND VALIDATION

For most of the design specifications, we will utilize both stationary testing and real-world testing. Stationary testing involves evaluating the device indoors without installation on a bike, and conducting tests outdoors on the street with the device installed on a stationary bike. Real-world testing will entail assessing the device in actual traffic conditions, with video recording for subsequent analysis.

7.1 Power Consumption

Power consumption is a key component of our project. Our goal is to ensure that our system can continuously operate for more than 2 hours, as described in 4. During our testing of other design requirements, we will also maintain a log to record the duration for which the system lasts each time. To mitigate power consumption risks, we'll measure the current draw for each component, optimize with lower-power devices, and consider increasing battery size for extended operation.

7.2 Radar Accuracy Baseline

These tests aim to benchmark the actual performance of the radar, noting the possible offsets between the real performance and the datasheet. We will perform the following tests:

- Distance and Angle: We will park a car at a known distance (e.g., measured using a tape measure) and angle (e.g., measured by a protractor) from the radar and cross-check it with the output of the radar. For better test coverage, we will place the car at different locations and repeat this check several times.
- Speed: We will have a driver drive a vehicle past the radar at a fixed speed (say 10 mi/h). Then, we'll check that the reported speed is within the accuracy requirement.
- Direction: We can have the car drive towards and away from the radar several times and verify the sign of the reported velocity is correct.
- Range: We will fix the radar at one location, and have an object approaching the radar. We will record the farthest distance that the radar can detect for that object. We will repeat this by having the object approach from different locations and see the range of the detection.
- Data Update Frequency: To ensure the radar data update frequency is at least 10 Hz as specified in 4, static tests will be conducted with a person walking towards the radar. We will capture the radar data output and visualize the intervals between successive updates. We can achieve this by counting the number of data points received within a specific period. We will repeat this process at least 3 times and take the average of the frequency. The person should be walking so we can verify the radar is sending live data, not sending the same thing over and over.

7.3 Radar Accuracy Confusion Matrix

We will conduct static testing in both indoor and outdoor environments to assess its ability to detect all planned incoming vehicles. Additionally, we'll utilize a car's speedometer to evaluate the efficacy of our collision warning system. Once these components have undergone rigorous testing, yielding a confusion matrix with fewer than 40% false negatives and 30% false positives of more than 40 instances, we will proceed to install these devices on bikes for real-world testing.

Real-world testing involves riding bikes on the streets, recording videos to capture real-world footage, and subsequently analyzing the system's performance. We will also aim to have a confusion matrix with fewer than 40% false negatives and 30% false positives.

7.4 Simultaneous Targets

To test that the radar is able to track at least three separate targets, we plan to use static testing, since it will be difficult to find a big enough place to park three cars side-by-side and to find them in the first place. Instead, with

the radar on a bench top, we will draw three imaginary 10 ft. wide "lanes" from the radar. Then, we will ask three people to move along the lanes, one person in each lane. As that occurs, we'll check that the radar output correctly identifies three distinct people moving around.

7.5 Waterproof

We are trying to reach an IPX4 waterproof rating. This testing is also relatively simple to accomplish. There has been standard procedure online and example videos for us to reference to[13]. In short, all we need is to find a sprinkling hose and spray the system to ensure that there are no leaks and that the system functions after the sprinkling. We will first do a couple of testing on the casing itself before putting the electronics in to prevent poor sealing from causing short circuits.

7.6 System Uptime

We will measure this through similar methods as power consumption, we will record logs anytime that our system is unable to perform detection. If there is any time the system is down, we will check the log and see what is going on and try to replicate the scenario and fix the problem.

7.7 Ease of Installation

We will set up a survey and invite 10 bike commuters to install such a device on their bikes and see how would they rate the difficulties of installation. We consider this a success if more than half of the participants think it is easy. We will also ask what part the participants think is hard to install, and modify the designs to make it a convenient design.

8 PROJECT MANAGEMENT

8.1 Schedule

The schedule and responsibilities are shown in Fig. 9.

8.2 Team Member Responsibilities

- Jack: Radar initialization and implementation, radar tuning, system integration
- Jason: Raspberry Pi software implementation, radar tuning (FCW)
- Johnny: Turn Signal with Auto-Cancellation, CAD design for exterior, Parts Installation.

8.3 Bill of Materials and Budget

The BOM as of writing this report is in Table 2. For the most up-to-date BOM, please see this spreadsheet.

8.4 Risk Mitigation Plans

Several risk factors are involved in this project:

- Radar: We are not sure about the detection accuracy of the radar. To mitigate this, we need to set up the radar and conduct some static testing to benchmark performance.
- Waterproof case: We need to make the material and the setup of the case will not compromise the performance of our system. To mitigate this, we need to carefully review the specs of the radar and the available materials.
- Auto-cancelation turn signals: We are not sure how sensitive the magnetometer will be to detect heading changes of the bike, which is the information supplier for the auto-cancelation turn signals. To mitigate this, we have backup plans like using sensors such as potentiometer or flex sensors.

In addition, we also have slack time described in 8.1 so that we can address issues when they arise.

9 RELATED WORK

A group in 2019 [7] has developed a similar bike safety suit that included a safety vest, blind spot detection using LiDAR, and a user interface app to control settings.

Another group in 2019 seems to have developed a system that seems to have used ultra-wideband technology to localize bicycles and vehicles relative to each other [17]. Unlike other systems listed here, it seems that both the vehicle and bicycle will receive alerts.

Another group in 2021 [3] developed a microwave-based bicycle safety and awareness tool.

10 SUMMARY

BikeBuddy provides a new solution to cycle safety by allowing cyclists to gain more situational awareness with tools like blind spot and collision detections. The added turn signal will also allow the cyclist to communicate with other road users better, enhancing safety. The Radar sensor and the waterproof cage allow the system to function even during adverse weather conditions, which allows the commuter to bike with such a system in the rain.

The system is designed so that urban cyclists can afford this product and enhance their safety in their day-to-day lives. In addition, on a macroeconomic level, this system could reduce total healthcare and vehicle repair costs. If we lower collision rates between vehicles and cyclists, we reduce the number of hospital visits needed for cyclists and the repairs needed to repair vehicles damaged in collisions.

The upcoming challenges would be the interaction of sensors and systems so that BikeBudy will yield the performance that is described in the requirement.

Table 2: Bill of materials

Description	Model	Manufacturer	Quantity	Cost	Shipping	Total
On Bike Embedded System	4	Raspberry Pi	1	$0	$0	$0
Radar	K-LD7	RFBeam	2	$85.78	$6.99	$178.55
Display	5 inch	Hosyond	1	$42.99	$0	$42.99
Power Bank	26,800mAh	Anker	1	$56.20	$0	$56.20
LED	Yellow	Hyuduo	1	$11.75	$0	$11.75
Momentary Switches	PBS-33B-BK-X	Twidec	1	$9.99	$0	$9.99
Triple-axis Magnetometer	MMC5603	Adafruit	1	$5.95	$4.61	$10.56
Transistor	BS170	from 220 lab	2	$0	$0	$0
Resistors	10k	from 220 lab	2	$0	$0	$0
						$323.67

Glossary of Acronyms

- RPi – Raspberry Pi
- FCW - Front Collision Warning
- RCW - Rear Collision Warning
- BSM - Blind Spot Monitoring

References

[1] *Application note AN-03. RADOME (Radar Enclosure)*. Version V1.2. RFBeam Microwave GmbH. May 2023. URL: https://rfbeam.ch/wp-content/uploads/dlm_uploads/2023/05/AN-03-Radome.pdf.

[2] Outside Online. *Strava's End-of-Year Insights*. https://velo.outsideonline.com/road/road-racing/strava-end-year-insights-live-fastest-state/.

[3] Albany Bloor, Emily Clayton, and Jason Xu. *ECE 18-500 Project: Bikewardsview*. https://course.ece.cmu.edu/ ece500/projects/f21-teamb3/. 2021.

[4] SparkFun Electronics. *SparkFun-Boards*. https://library.io/libraries/509-SparkFun-Boards. 2023.

[5] Bret. *Raspberry Pi 5 Review. Harder, Better, Faster, Stronger*. Feb. 2024. URL: https://bret.dk/raspberry-pi-5-review/#Raspberry-Pi-5-Benchmarks.

[6] SparkFun Electronics. *SparkFun-Connector*. https://library.io/libraries/513-SparkFun-Connectors. 2023.

[7] Michael You, Sid Lathar, and Benjamin Huang. *ECE 18-500 Project: CycleSafe*. https://course.ece.cmu.edu/ ece500/projects/s19-teama1/. 2019.

[8] OpenJS Foundation and Electron Contributors. *What is Electron?* URL: https://www.electronjs.org/docs/latest.

[9] Roy Gothie. *Visibility and conspicuity while riding your bike*. Mar. 2019. URL: https://www.penndot.pa.gov/PennDOTWay/pages/Article.aspx?post=206.

[10] Muhammad Hasanujjaman, Mostafa Zaman Chowdhury, and Yeong Min Jang. "Sensor Fusion in Autonomous Vehicle with Traffic Surveillance Camera System: Detection, Localization, and AI Networking". In: (Mar. 2023). URL: https://www.researchgate.net/publication/369468206_Sensor_Fusion_in_Autonomous_Vehicle_with_Traffic_Surveillance_Camera_System_Detection_Localization_and_AI_Networking/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6Il9kaXJlY3QiLCJwYWdlI

[11] SparkFun Electronics. *SparkFun-IC-Special-Function*. https://library.io/libraries/527-SparkFun-IC-Special-Function. 2023.

[12] Petra Industries. *Be the Expert: Waterproof, Water-Resistant and Understanding the Standards Rating Charts*. Mar. 2022. URL: https://blog.petra.com/blog/understanding-water-standards-rating-charts/.

[13] Puneet Sharma. *IPX4 Testing Video*. https://www.youtube.com/watch?v=myf5kv6eo4A. 2015.

[14] *K-LD7. data sheet*. Version Rev. B. RFBeam Microwave GmbH. Mar. 2023. URL: https://rfbeam.ch/wp-content/uploads/dlm_uploads/2022/10/K-LD7_Datasheet.pdf.

[15] Simon Long. *Bullseye - the new version of Raspberry Pi OS*. Nov. 2021. URL: https://www.raspberrypi.com/news/bookworm-the-new-version-of-raspberry-pi-os/.

[16] Raspberry Pi Ltd. and Documentation Contributors. *Raspberry Pi hardware*. URL: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html.

[17] *MDEK1001. Ultra-Wideband (UWB) Transceiver Development Kit*. URL: https://www.qorvo.com/products/p/MDEK1001.

[18] WI - Weather Forecast Office Milwaukee/Sullivan. *Using and Understanding Doppler Radar*. URL: https://www.weather.gov/mkx/using-radar.

[19] Lea Moussy. *LiDAR vs. RADAR*. May 2020. URL: https://www.yellowscan.com/knowledge/lidar-vs-radar/.

[20] N/A. *Guide for the Development of Bicycle Facilities (4th Edition)*. American Association of State Highway and Transportation Officials (AASHTO), 2012. ISBN: 978-1-56051-527-2. URL: https://app.knovel.com/hotlink/toc/id:kpGDBFE008/guide-development-bicycle/guide-development-bicycle.

[21] MIT News. *How fast can humans react to car hazards?* 2019. URL: https://news.mit.edu/2019/how-fast-humans-react-car-hazards-0807.

[22] National Oceanic and Atmospheric Administration. *How radar works*. Sept. 2023. URL: https://www.noaa.gov/jetstream/doppler/how-radar-works.

[23] Outside Online. *Lane Width*. URL: https://safety.fhwa.dot.gov/geometric/pubs/mitigationstrategies/chapter3/3_lanewidth.cfm.

[24] *Products*. URL: https://rfbeam.ch/products/.

[25] SparkFun Electronics. *SparkFun-Switches*. https://library.io/libraries/535-SparkFun-Switches. 2023.

[26] Driver License Division Texas Department of Public Safety. *Texas Driver Handbook*. Jan. 2022, p. 45. URL: https://www.dps.texas.gov/internetforms/Forms/DL-7.pdf.

[27] Elliot Williams. *A Raspberry Pi 5 Is Better Than Two Pi 4s*. Sept. 2023. URL: https://hackaday.com/2023/09/28/a-raspberry-pi-5-is-better-than-two-pi-4s/.

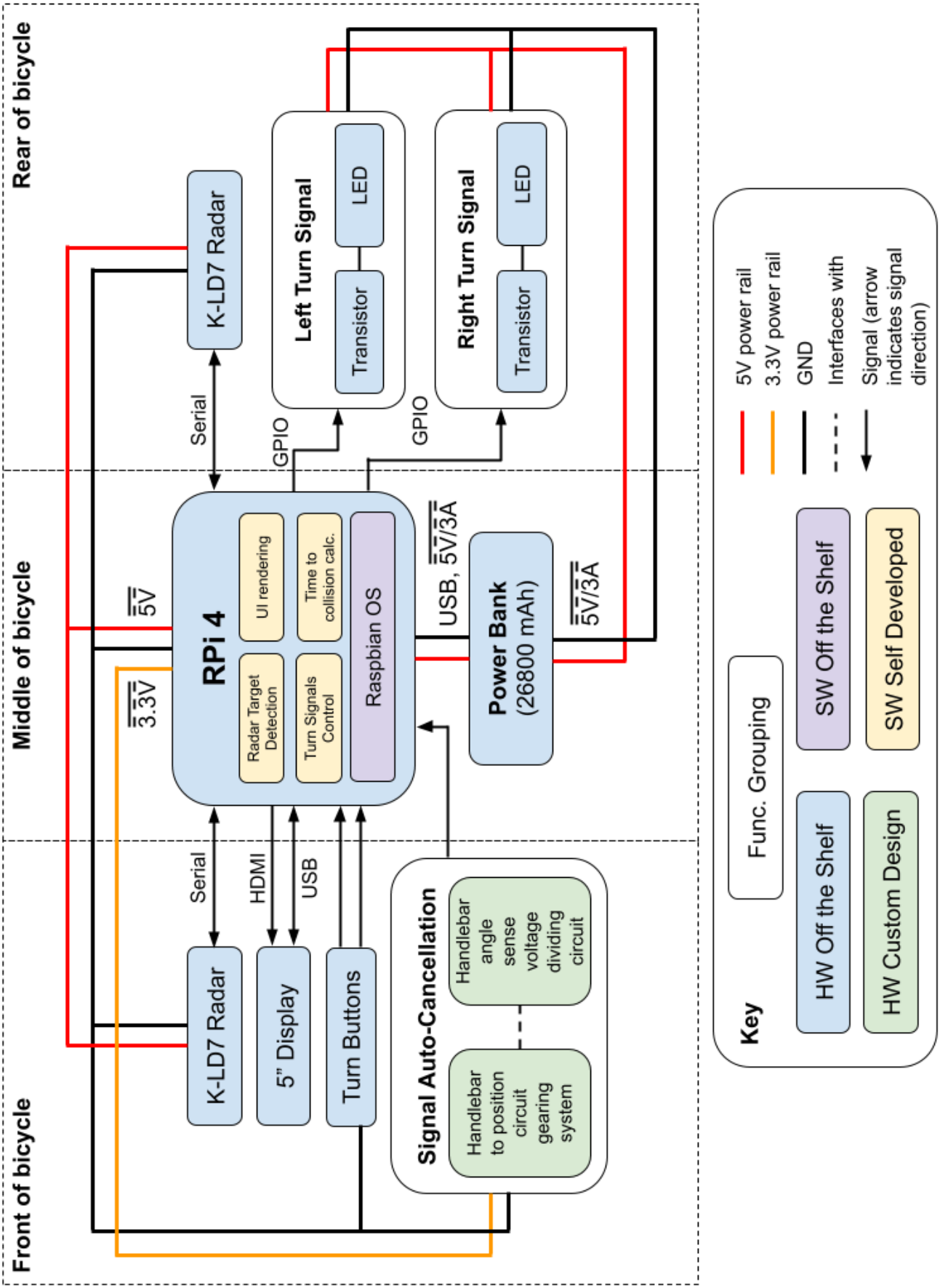

Figure 8: A full-page version of the same system block diagram as depicted earlier.

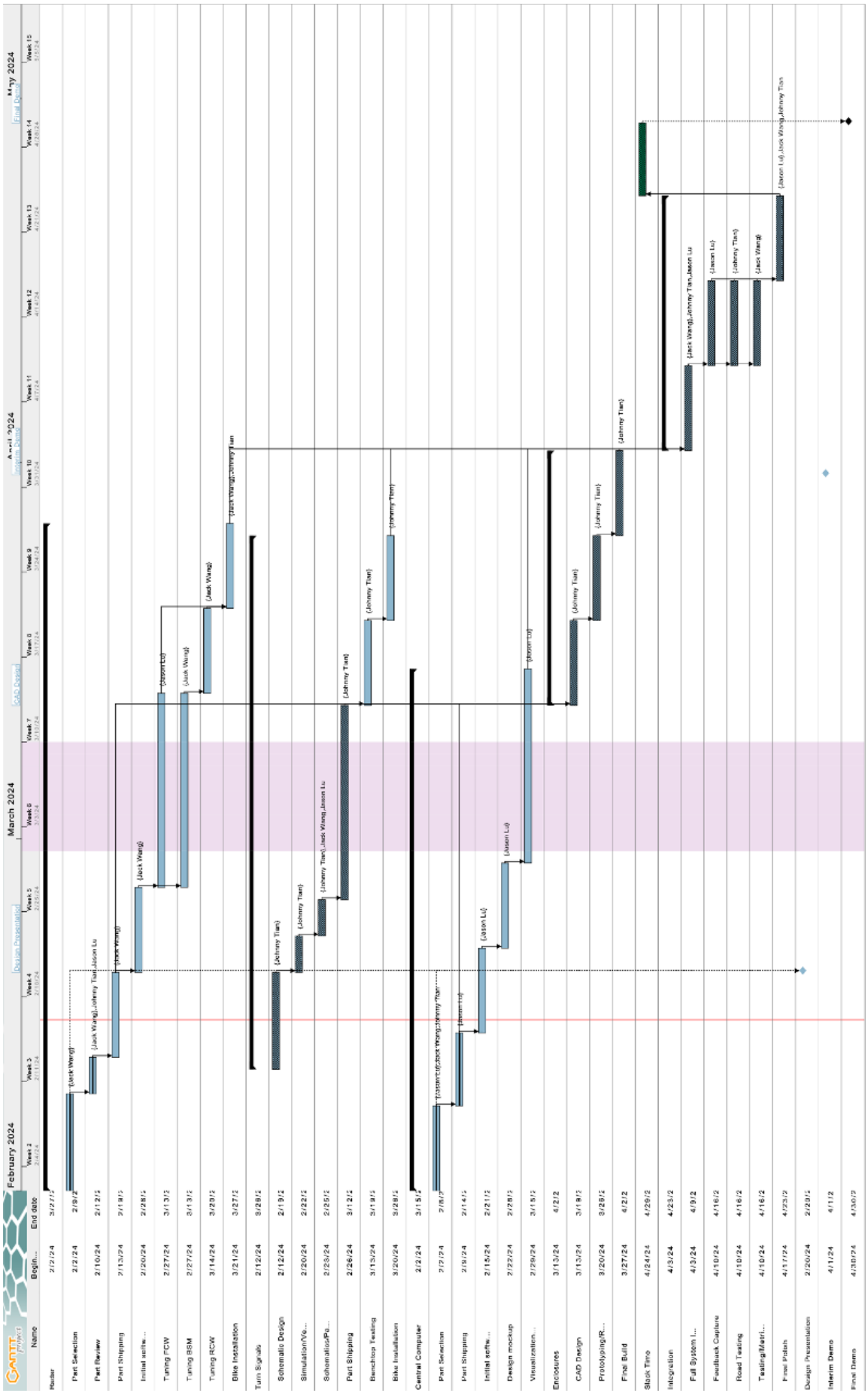

Figure 9: Gantt Chart