Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

BikeGuard

by
Team 8
BikeGuard

Team Members

Beren Yagmur Donmez bydonmez@bu.edu
Margherita Piana mpiana@bu.edu
Marissa Ruiz mbruiz@bu.edu
Bennet Taylor betaylor@bu.edu
Albert Zhao albertz@bu.edu

Submitted: April 18, 202

Table of Contents

Table of Contents **2**
Executive Summary **3**
1 Introduction **4**
2 System Overview and Installation **5**
2.1 Overview Block Diagrams 5
2.2 User Interface 7
2.3 Physical Description 11
2.4 Installation, Setup, and Support 14
3 Operation of the Project **15**
3.1 Operating Mode 1: Normal Operation 15
3.2 Operating Mode 2: Abnormal Operations 19
3.3 Safety Issues 21
4 Technical Background **22**
5 Operation of the Project **25**
Electrical and Safety Standards 25
Wireless Communication and Internet Standards 25
Software Development and Cybersecurity Standards 26
Regulatory and Governmental Compliance 26
6 Cost Breakdown **27**
7 Appendices **28**
5.1 Appendix A - Specifications 28
5.2 Appendix B – Team Information 29

Executive Summary

BikeGuard offers a comprehensive theft detection system designed to protect 80% of bike owners affected by theft. Combining hardware components such as a Raspberry Pi 4 Model B, an MPU-6050 accelerometer, a Piezo buzzer, and a Raspberry Pi Camera Module V3 NoIR Wide in a durable and inconspicuous 3D printed enclosure mounted on the bike, BikeGuard ensures robust security and seamless integration. Utilizing machine learning, the system classifies theft attempts, such as lock picking, with an accuracy of 88.16%. It delivers real-time alerts via Wi-Fi with potential for cellular data usage for enhanced connectivity and GPS tracking. The user interface offers a web and phone-based platform, built with Flask and React, featuring live video streaming and push notifications triggered by motion-detection sensors. Extensive testing of our final prototype demonstrated the effectiveness and reliability of both the hardware and software components.

With its innovative approach, BikeGuard goes beyond traditional locks to offer cyclists a scalable, intelligent solution for active theft prevention, reimagining bike security with real-time technology and advanced analytics.

1 Introduction

Bicycle theft is a pervasive issue, with an estimated 2.4 million bikes stolen annually in the US and Canada. This loss is valued at around $1.4 billion [1]. Traditional bike locks, while effective at delaying theft, fail to provide real-time alerts or tracking capabilities, leaving bike owners vulnerable and without actionable recourse. BikeGuard is a theft detection and prevention system designed to address these failures. Our project integrates advanced hardware and software to provide cyclists with a proactive and intelligent solution to safeguard their bikes. By combining sensors, real-time data analysis, and mobile connectivity, BikeGuard aims to transform bike security from passive protection to active prevention. Our team's approach focuses on hardware and software integration by making use of a key switch, a Raspberry Pi 4 Model B, an accelerometer, a buzzer, and a camera housed in a durable 3D printed enclosure, along with an innovative interface for a more friendly user monitoring and control. Our product will leverage machine learning to train a logistic regression model on pitch and roll data to distinguish between theft attempts and benign activities. For our final product, we achieved connectivity and interface through a web-based user interface built on Flask and React, which provides live updates, video feeds, and notifications to the user. We are currently working on a mobile app interface that will provide the same functionality as the web interface. BikeGuard's proactive alert system ensures that bike owners are informed the moment suspicious activity occurs, letting the users intervene in the situation when it's happening. BikeGuard also offers GPS tracking so that the user knows where their bike is at all times. We are currently implementing a cellular modem into the device to allow users to connect to a cellular provider when wifi is not available. The highlights of our product include real-time theft detection and alerts, a combination of hardware and software in a compact, user-friendly design, and high accuracy in identifying theft attempts.

1. https://thebestbikelock.com/bike-theft-statistics-us/

2 System Overview and Installation

2.1 Overview Block Diagrams

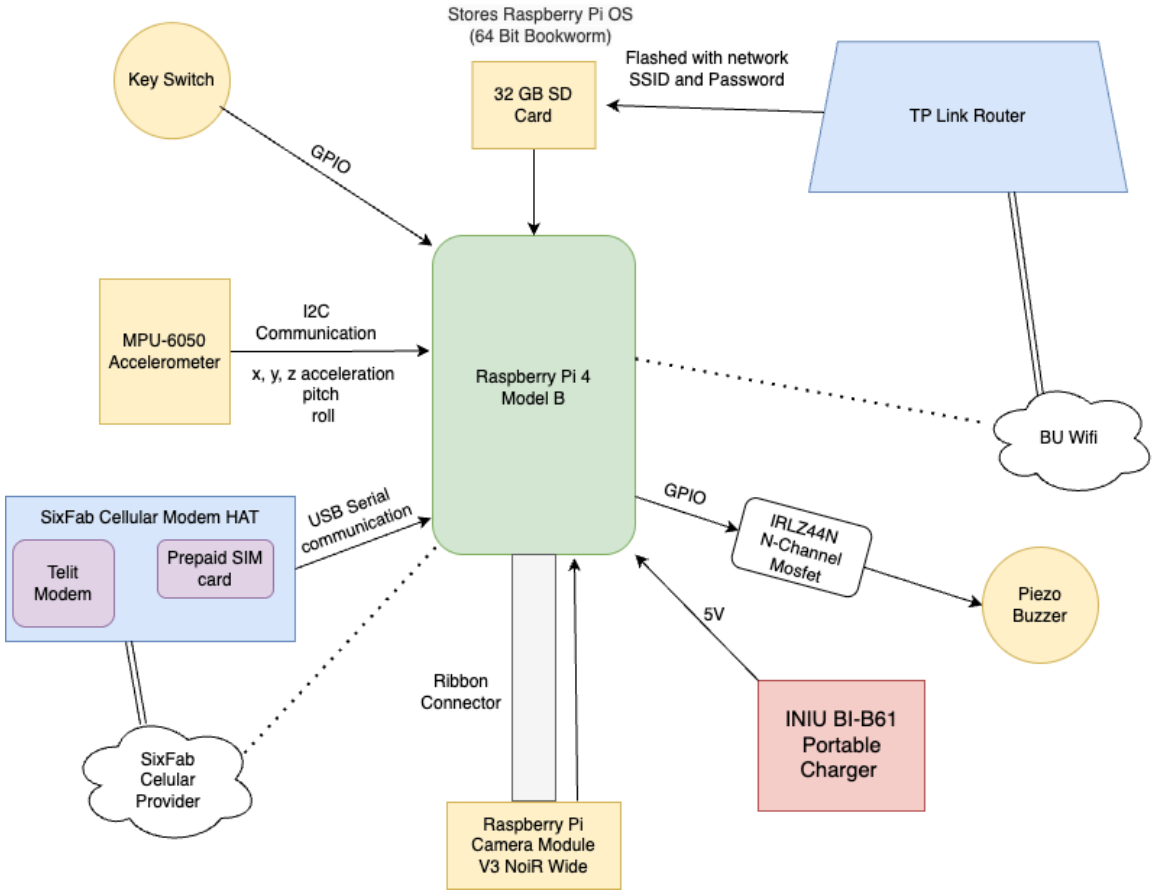

Figure 2.1.1 Hardware Component Block Diagram

Figure 2.1.1 describes the hardware components in the BikeGuard system and their respective communication/connection methods. The system is powered by an INIU BI-B61 Portable Charger that supplies a 5V to the Raspberry Pi. The Raspberry Pi software has been flashed to an SD card that will take effect at boot. The operating system used for the Raspberry Pi is the 64-bit bookworm Raspberry Pi OS, which is recommended for the Raspberry Pi 4 Model B. The SD card also holds the SSID and password for the TP-Link Router. The MPU-5060 accelerometer communicates with the Raspberry Pi via I2C communication. The Raspberry Pi camera sends video data to the Raspberry Pi via a ribbon connector. The Piezo buzzer is controlled by an IRLZ44N MOSFET, which is connected to the Raspberry Pi via a GPIO pin. The key switch is connected to the Raspberry Pi via a GPIO connection. Finally, the modem is connected to the Raspberry Pi via USB serial communication.

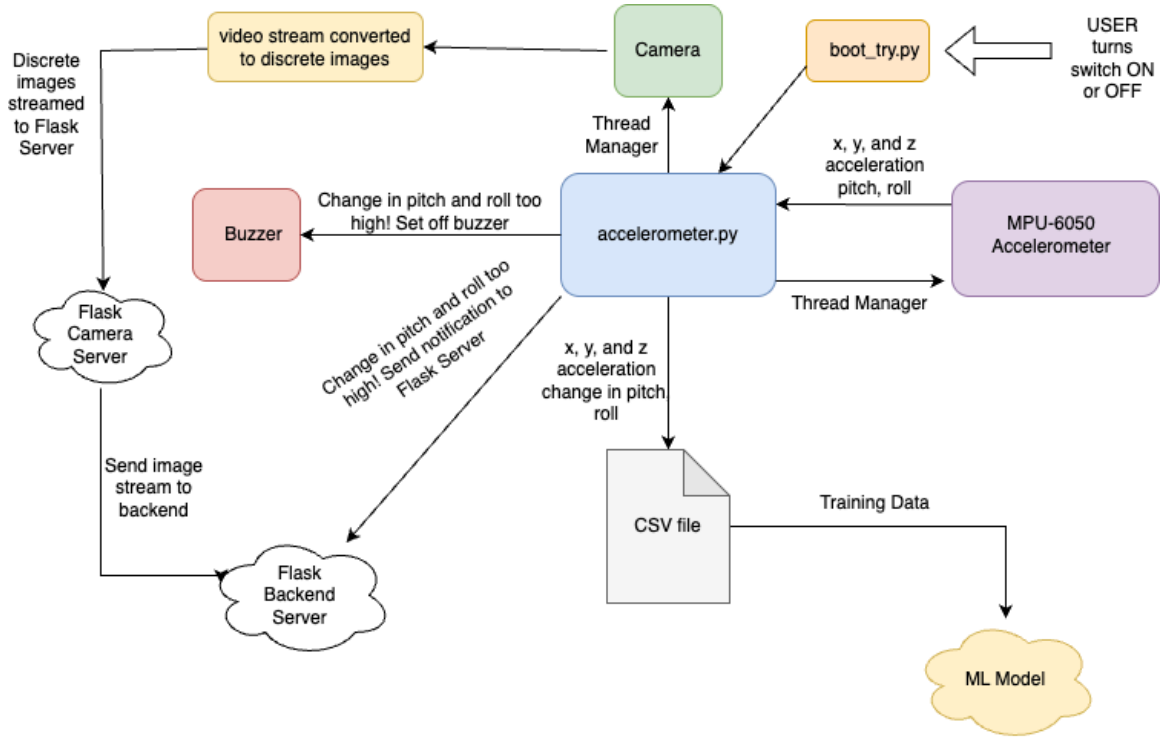

Figure 2.1.2 Onboard Software Flow Chart

Figure 2.1.2 shows the onboard software flow chart. The Python script boot_try.py is run as soon as the Raspberry Pi boots. When the user switches the key switch to ON, the boot_try.py is alerted via GPIO connection and runs the accelerometer.py script. The accelerometer.py script is the device's driving code. It has two processes running under a thread manager: the camera and the accelerometer. The camera thread continuously polls video data from the Raspberry Pi camera. This video stream is converted to a stream of discrete images for easier front-end implementation. This stream of discrete images is then sent to a Flask server hosted on the Raspberry Pi. This is then sent to our main backend server to be embedded into the front-end user interface. The accelerometer thread polls x, y, and z acceleration from the accelerometer, which the accelerometer thread then converts to pitch and roll. The accelerometer thread also keeps track of changes in pitch and roll by comparing current pitch and roll values to pitch and roll values from the previous time step. The change in pitch and roll is more useful for identifying shaking. The accelerometer thread then writes the x, y, and z acceleration as well as the change in pitch and roll values to a CSV fil,e which can be used later to train the machine learning model. If the accelerometer thread detects changes in pitch and roll values that are too high, it will set off the buzzer and send a notification to the back-end server.

2.2 User Interface

When the user first opens the web-based user interface, they will see an option to log in (Figure 2.2.1).

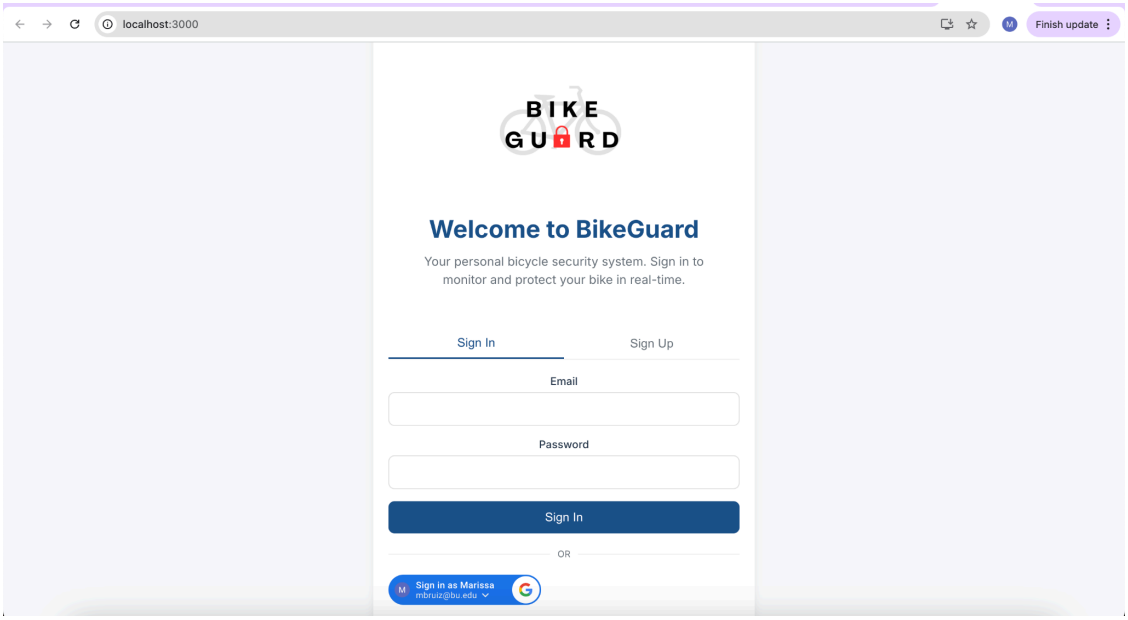

Figure 2.2.1 User login page

If the user does not have an account, they can sign up for an account under the “Sign Up” tab (Figure 2.2.2)

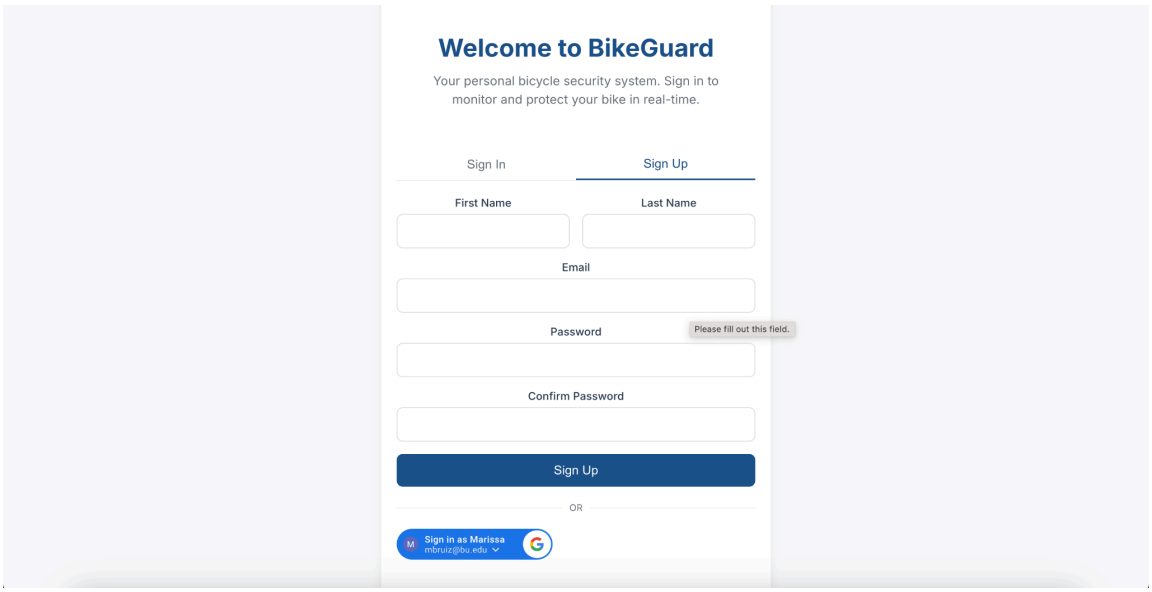

Figure 2.2.2 User sign-up page

Once the user logs in, they will see their account dashboard (Figure 2.2.3). Here they will see the live video feed, the live location map, notifications from the device, and buttons to start/stop alarm or turn tracking off.

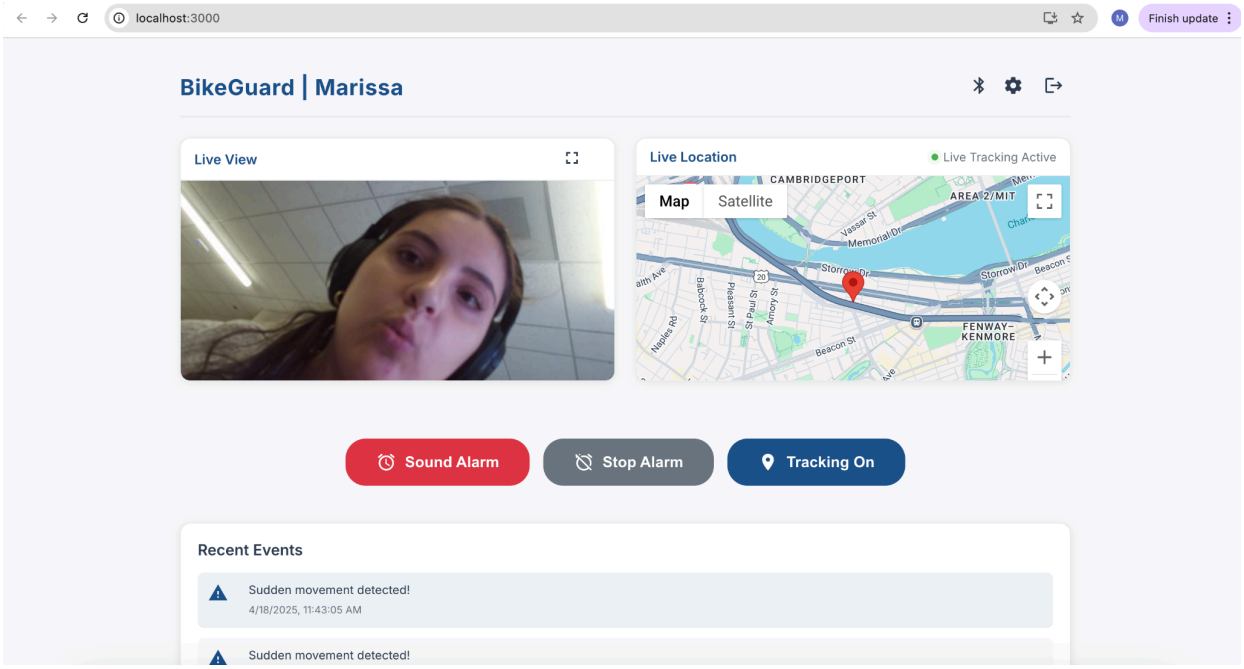

Figure 2.2.3 BikeGuard user dashboard

Once the user is logged in, they can also access a Settings page (Figure 2.2.4). Here they can find more information.

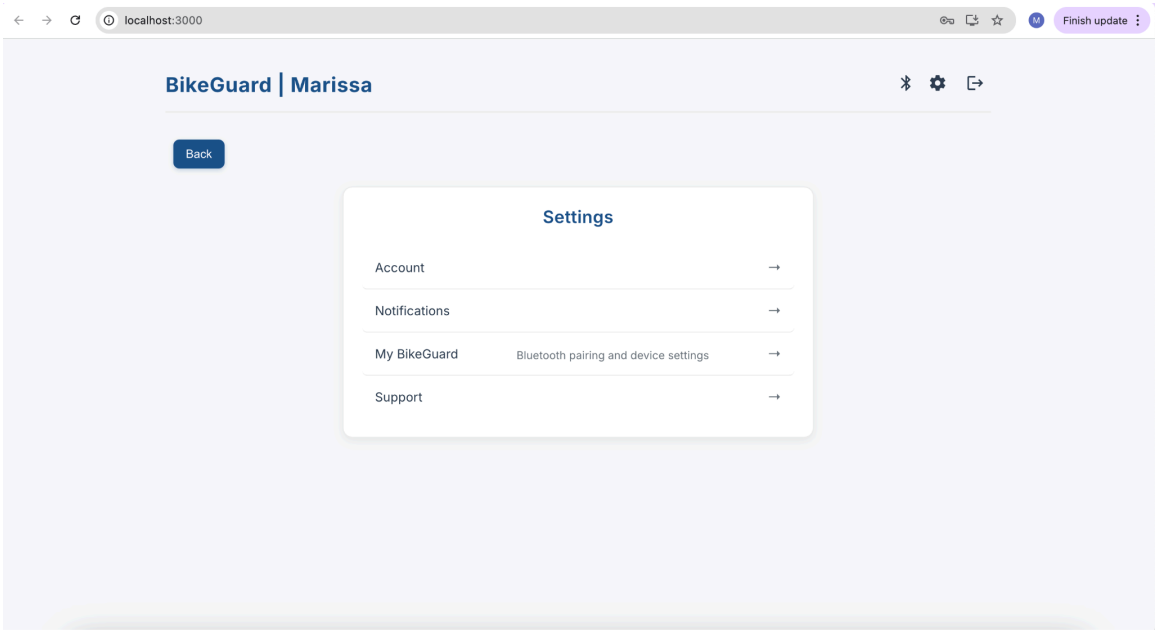

Figure 2.2.4 Settings Page

On the settings page, the user can access their account settings where they can change their password (Figure 2.2.5), they can access a support page with frequently asked questions and contact information (Figure 2.2.6), and can check their Bluetooth pairing for their bike (Figure 2.2.7).

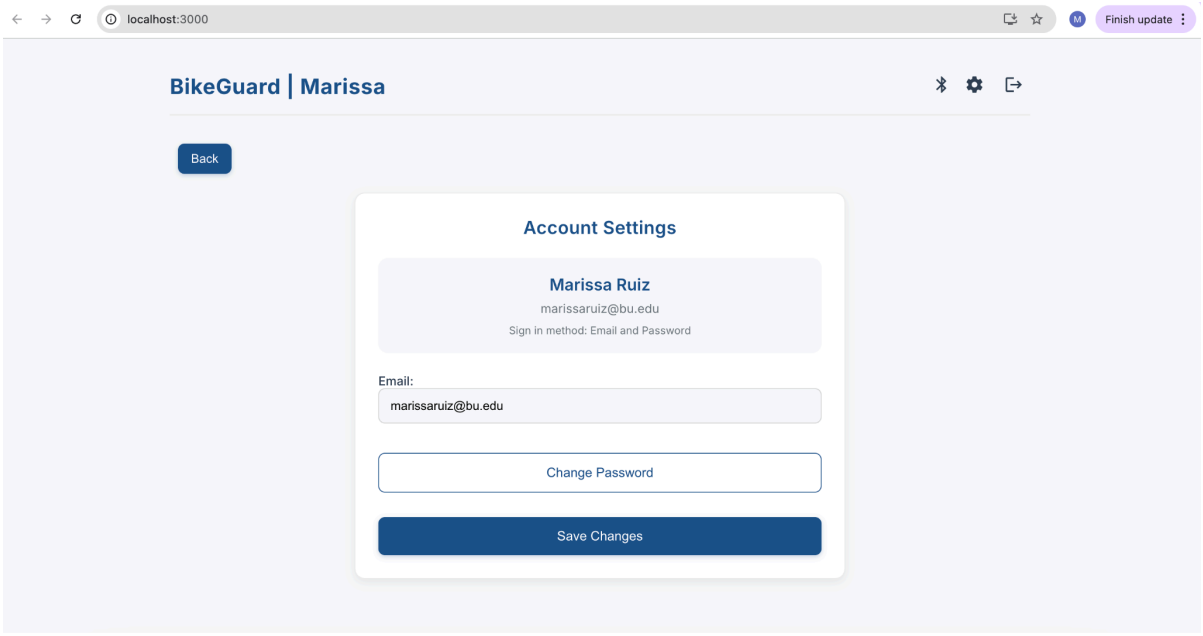

Figure 2.2.5 Account settings page

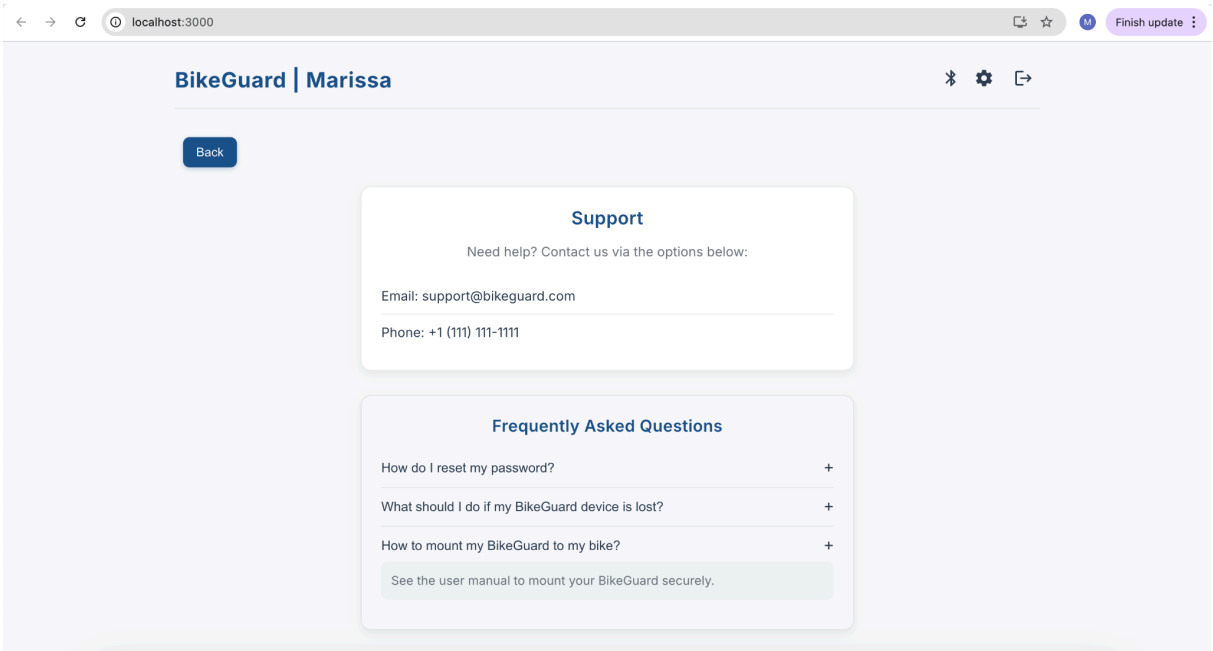

Figure 2.2.6 User support page

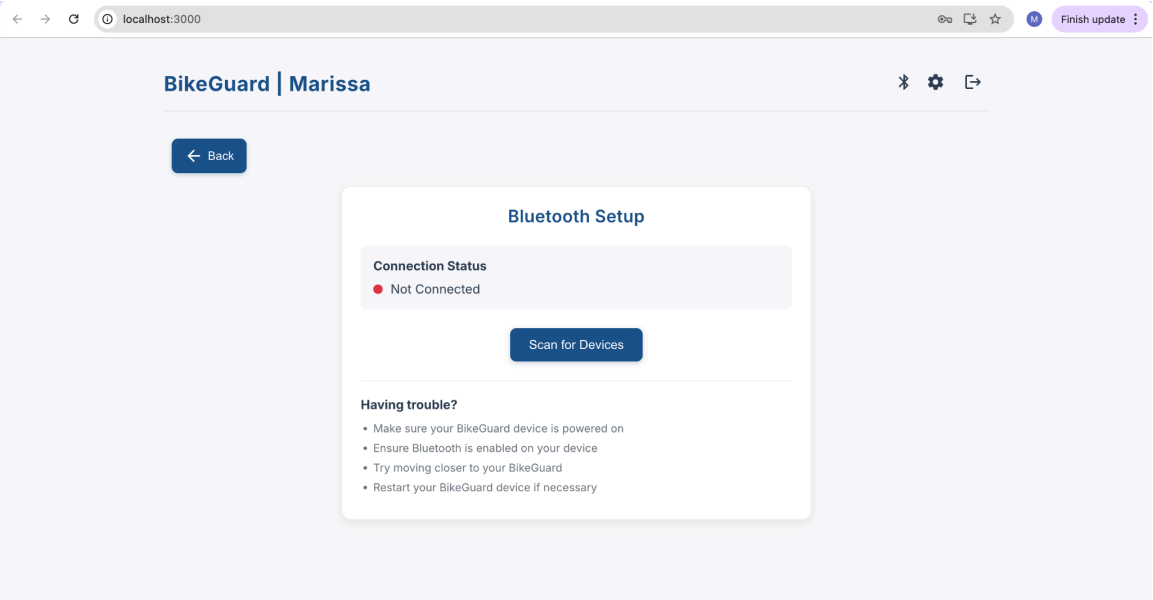

Figure 2.2.7 Bluetooth Setup and Connection

The website is currently being adapted to work as a progressive mobile application, as shown in Figure 2.2.8.

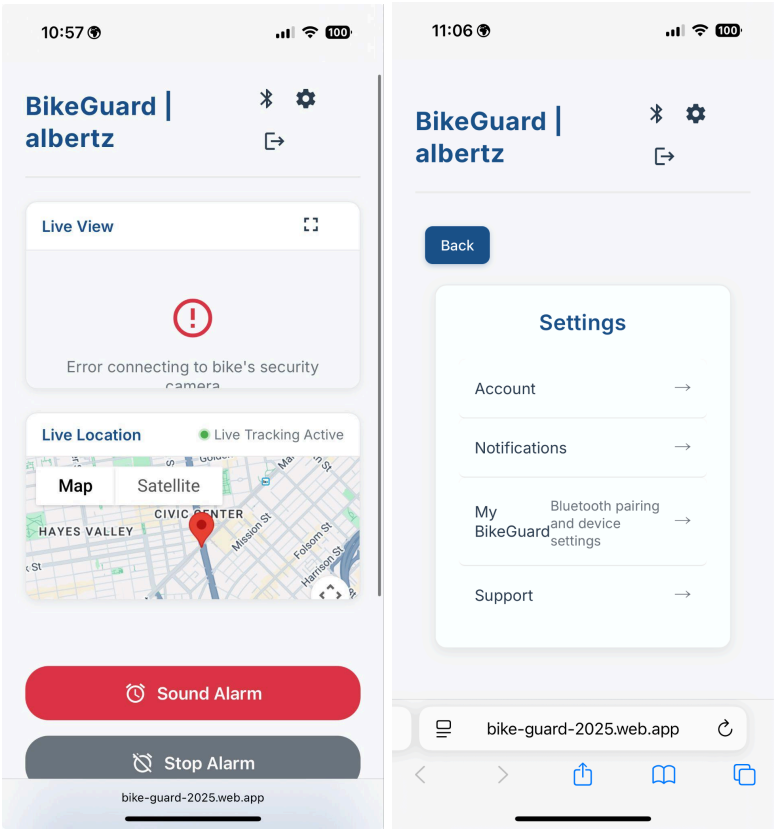

Figure 2.2.8 Progressive Mobile Application

Here is a display of the switch in the ON and OFF positions for reference

Figure 2.2.8: Switch in ON and OFF position

2.3 Physical Description

Figure 2.3.1 shows the hardware that is housed in the BikeGuard as well as the wiring connections.

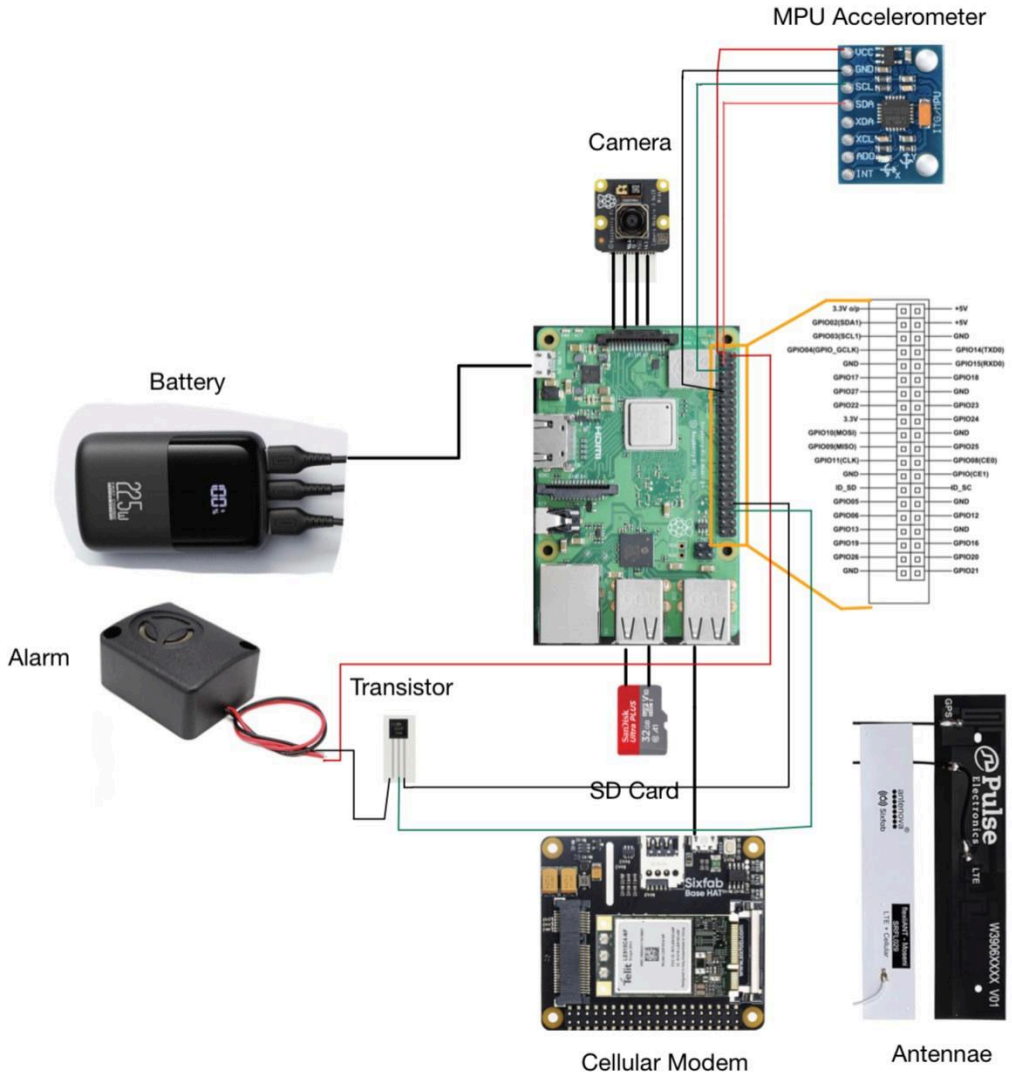

Figure 2.3.1 Hardware components and wiring

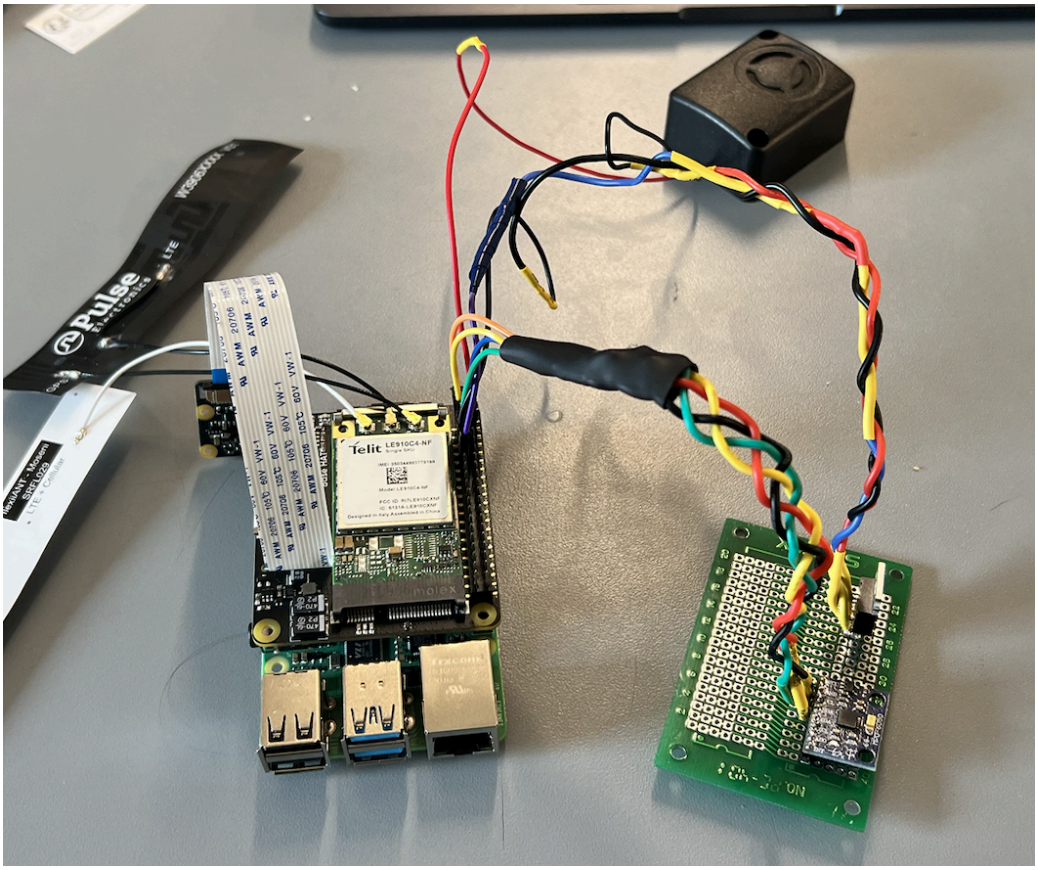

Figure 2.3.2 Completed circuitry before placing in enclosure

Figure 2.3.2 shows the completed circuit before we inserted it into the enclosure.

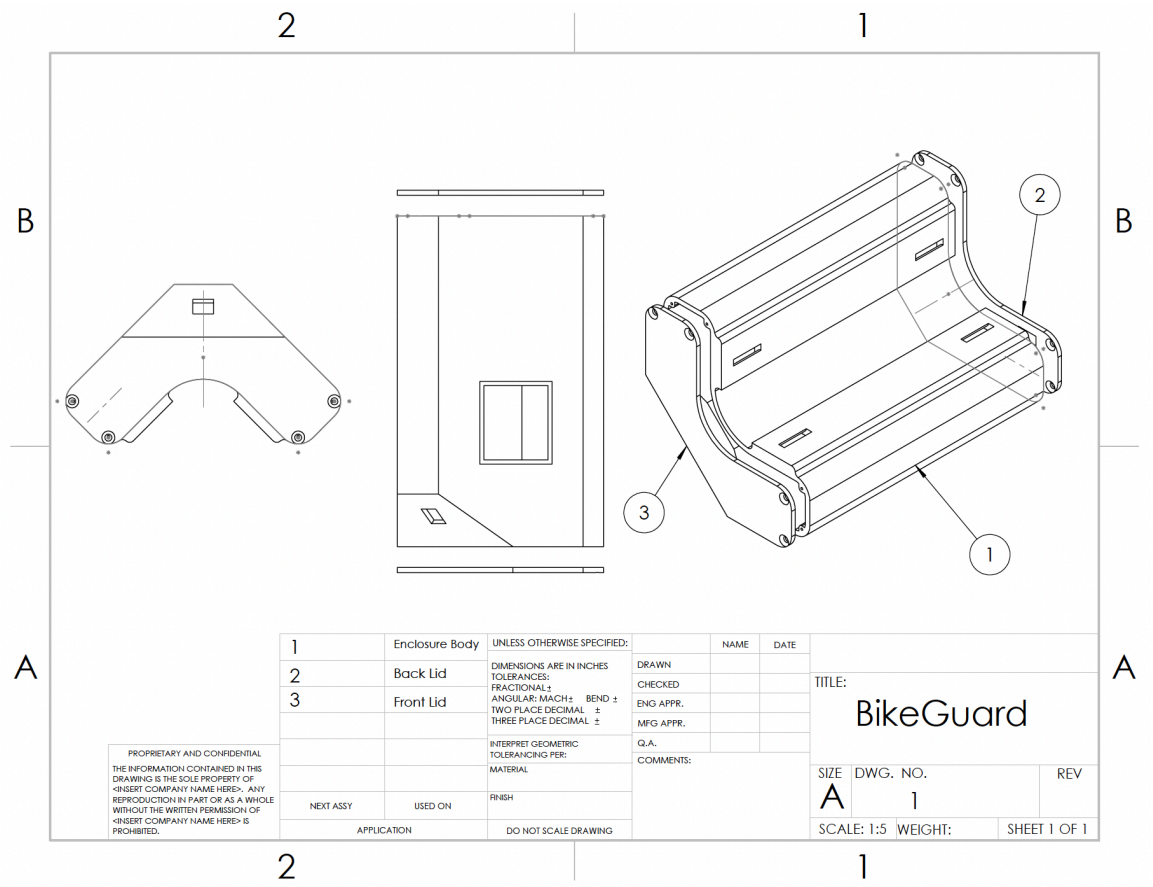

Figure 2.3.3 BikeGuard Enclosure

Figure 2.3.3 shows the technical drawings of the BikeGuard enclosure. This includes the main body (1), back lid (2), and front lid (3). The drawing shows a number of important features that the enclosure has to support the device. Starting with the body, there are a total of 8 M2 screw holes for fastening the back and front lid to their respective sides. Another feature of the main body is the four holes on the bottom of the enclosure, which are used for attaching the enclosure to the main bike structure via hose clamps. This provides a secure, removable, and easy way to attach the device to a user's bike. The front of the enclosure body has a small hole angled upwards, which is where the camera is placed. The last feature of the body is the viewing window for the battery charge, located on the side of the device. The front lid is a standard lid with only 4 M2 screw holes. The back lid has a port for charging the battery and a slot for the key lock switch, used to manually activate the alarm system.

2.4 Installation, Setup, and Support

Installation:
We recommend that users install the device on the top tube of the bike (indicated by the red circle in Figure 2.4.1). We also recommend orientating the device so the key switch faces the front of the bike. This allows for easier use and better camera visibility.

1. Use the clamps to mount the device onto the top tube of the bike.
2. Tighten the clamps using any flathead screwdriver.
3. Ensure that the clamps are snug but not too tight to avoid damage to the bike frame.

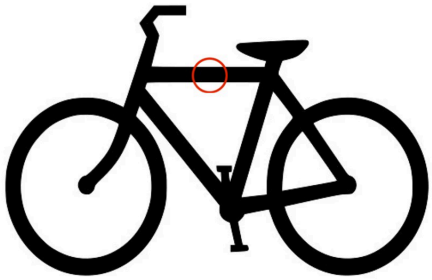

Figure 2.4.1 BikeGuard Placement

Setup:
1. Insert the provided key into the key switch and rotate it to the "ON" position. Then take the key out and keep it somewhere safe. When you want to turn off the device, use the key to rotate the switch to the OFF position.
2. Make sure that your smartphone or computer is connected to a stable Wi-Fi or cellular network, as internet access is required for account creation and device linking.
3. Create an account using the web interface or app. You can create an account either through Google or email. Save your login credentials!
4. Go to the settings page and link your device via Bluetooth.
5. Once the device is linked, you should see the device’s live video stream and bike’s live location via the website/app.

Support:
Support can be found on the settings page of the user interface. There, we address frequently asked questions and provide contact information for further aid. More support for installation and setup can be found on our GitHub page: https://github.com/alberttzhao/bike-guard

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

Hardware and Installation

For BikeGuard to work properly, the user has to install the device in the recommended position to get the best monitoring result and obtain the desired normal operation.

1. Locate the top tube of your bike as the recommended mounting position
2. Position the BikeGuard device on the tube with the key switch facing the front of the bike for optimal camera visibility
3. Secure the device using the provided hose clamps
4. Use a flathead screwdriver to tighten the clamps until the device is snug
5. Ensure the clamps are secure but not overtightened to prevent damage to the bike frame

Software and System Activation

Once the device is mounted properly as instructed above, the user can turn the device on. The user needs to ensure they are connected to wifi or cellular for normal operation.

1. Insert the provided key into the key switch
2. Rotate the key to the "ON" position to activate BikeGuard
3. Remove the key and store it in a safe location
4. The system will now be actively monitoring for suspicious activity

User Interface Options

The BikeGuard dashboard provides access to the following features. Make sure that all of these features listed below are visible from our BikeGuard website or mobile application.

1. Live camera feed from your bike
2. Real-time location tracking on an interactive map
3. Alarm controls
4. Notification center showing recent events

Now the device should be securely mounted at the right spot, activated, and showing all features on the software interface. Following these steps will ensure normal operation.

When the key switch is turned in the ON position, the BikeGuard device enters Theft Prevention Mode. When the key switch is turned to the OFF position, the BikeGuard device enters the Standby Mode. These modes are defined in the following table

Theft Prevention Mode	**Standby Mode**
Key switch turned to ON	Key switch turned to OFF
Accelerometer.py script running on the Raspberry Pi	Any instances of accelerometer.py on the Raspberry Pi are killed
Buzzer controlled by script	Buzzer inactive
Accelerometer and camera continuously polling data	The accelerometer and camera are idle/low power state, not collecting data
Raspberry Pi actively running	Raspberry Pi in low-power standby mode
Camera feed visible on UI	Camera feed not visible/frozen on UI
Live location available on UI	Live location unavailable/frozen on UI
Live notifications received on UI	Live notifications unavailable on UI
Buzzer control from UI	Buzzer control unavailable from UI

Table 3.1.1 Device operation modes

The user can expect the following system responses in normal operation while in Theft Prevention Mode:

Bike Monitoring from User Interface

- Live Camera Feed: The top-left panel shows a real-time video stream from your bike's camera
- Location Tracking: The top-right panel displays your bike's current location on a map
- Notifications: The bottom section shows recent alerts and events related to your bike

Available Controls from User Interface

- Sound Alarm: Triggers the alarm on your bike device
- Stop Alarm: Deactivates the currently sounding alarm
- Tracking On/Off: Enables or disables GPS tracking of your bike

Response to each user action

Action	Normal Response	Possible Issues
Sound Alarm	Bike buzzer activates. A confirmation message appears	"Cannot trigger alarm while offline" message if no internet connection
Stop Alarm	Bike buzzer stops. A confirmation message appears	"Cannot stop alarm while offline" message if no internet connection
Toggle Tracking	Map tracking status changes; button color changes	No response if the server connection is lost
Check Battery	Battery level is visible through a clear window on the device	Battery level not visible if privacy screen viewed from the wrong angle

Table 3.1.2 Device responses and issues for user actions

Error Conditions and Troubleshooting

When something is not working properly, error messages will be displayed through the web/mobile interface to tell the user what is wrong. These troubleshooting messages should display when met with certain error conditions.

- Camera Feed Error: If you see "Error connecting to bike's security camera," check that your BikeGuard device is powered on and has network connectivity
- Offline Banner: A banner will appear at the top of the screen if your device loses internet connection
- Map Loading Issues: If the map doesn't load properly, try toggling tracking off and on

- Buzzer Issues: If buzzer control is unresponsive, refresh the page and ensure you have internet connectivity
- Missing Notifications: If notifications disappear or aren't displaying, refresh the page and check your internet connection

Charging the Device

The user can monitor the battery level through the clear window on the enclosure. When the battery level is low, users can charge the battery using a USB-C cable connected to the charging port.

Exiting Normal Operation

To exit normal operation, you can either close the website/app or simply log out of your account. To log out, tap the logout icon in the top-right corner. To deactivate the system, insert the key and turn it to the "OFF" position. The user can confirm successful deactivation from the web or app interface.

3.2 Operating Mode 2: Abnormal Operations

BikeGuard has been designed with robustness in mind, but if a user encounters any abnormal states during operation, we offer this section to outline appropriate recovery procedures for abnormal operations.

Network Connectivity Issues

Network connectivity issues arise when the device is improperly configured and, as such, cannot connect to the internet. This can be due to Wi-Fi failures, cellular network failures, or both. When this happens, the “Offline” banner will appear on the interface, the camera feed will be unavailable, and the user will be unable to control the alarm remotely.

To recover:

- Check that your mobile device has internet connectivity, either through Wi-Fi or cellular connection
- Ensure your BikeGuard device is within range of your Wi-Fi network, if applicable
- Verify that your home Wi-Fi is functioning properly, if applicable
- Restart the BikeGuard device by turning the key switch to OFF, waiting 10 seconds, then turning it back ON
- If problems persist, try reconnecting the device through the settings page

Bluetooth Pairing Problems

Bluetooth pairing issues occur when users are unable to connect to BikeGuard device during initial setup. If you visit the settings/bluetooth page, it will show that your mobile device is unable to connect to BikeGuard, or that BikeGuard device is not found.

To recover:

1. Ensure Bluetooth is enabled on your mobile device
2. Make sure you are within 10 meters of the BikeGuard device
3. Restart the BikeGuard device using the key switch
4. Go to your device's Bluetooth settings, forget BikeGuard if previously paired, and try pairing again
5. If problems persist, try restarting your mobile device

Camera Feed Issues

The camera feed issue can arise when the device is improperly connected to the internet, bluetooth, etc. When it happens, a black screen appears where the camera feed should appear, and the “error connecting to the bike’s security camera” message will show.

To recover:

1. Verify the device is powered on (check battery indicator)
2. Restart the BikeGuard device using the key switch
3. Check for physical damage to the camera lens
4. If problems persist, contact support via our GitHub page

Unresponsive Buzzer

Happens when no sound is coming out of the buzzer when the alarm is triggered, or the buzzer continues after stopping the alarm.

To recover:

1. Check internet connectivity of both your mobile device and BikeGuard
2. Try manually triggering and stopping the alarm through the interface
3. Restart the BikeGuard device using the key switch
4. If problems persist, the buzzer may be physically damaged and require service

3.3 Safety Issues

BikeGuard has been designed with safety in mind, but users should be aware of several important considerations:

Physical Safety

Users need to ensure proper mounting to prevent the device from becoming loose while riding. Users should avoid checking notifications or the camera feed while riding to prevent dangerous distractions. Users should also store the activation key separately from the bike to prevent unauthorized system deactivation.

Device Safety

The device should not be exposed to extreme temperatures where it might be prone to fire or explosion. If the device becomes unusually hot, emits odors, or shows signs of swelling, power it off immediately. Use only the recommended USB-C cable for charging to prevent any current surge. While our device is water-resistant, avoid submerging the device or exposing it to heavy rain for extended periods.

Data Security

BikeGuard collects location data and camera footage that could contain personal information. Use strong, unique passwords for your account and regularly review privacy settings. Be aware of local laws regarding surveillance and recording in public spaces.

4 Technical Background

Housing

Our system is easy to use and mount. It is mountable on any strut of the bike via hose clamp. Our system is detachable for user flexibility. Its dimensions measure at 14.5 cm, 10cm, 10cm and it weighs 558 grams (1 lb 3.7 oz). This size ensures that the device is unobtrusive to the bike's normal functionality. The housing has been printed using a 3D printer and PLA filament.

Electronics

Our device is controlled by a Raspberry Pi 4 Model B microcontroller. The Raspberry Pi connects to a TP-Link router at boot to communicate with servers and devices on the local network. We are currently working on adding cellular networking capabilities to our product through a Sixfab modem. The Raspberry Pi is configured to run a script called boot_try.py at boot. This script checks to see whether the key switch is in the ON or OFF position. If it is in the ON position, the script runs our driving file called accelerometer.py, which starts polling from all the peripherals. If it is in the OFF position, the script checks for any instances of accelerometer.py and eliminates them. In this manner, our system is able to save battery by polling from peripherals only when the key is switched to ON.

The main sensor implemented in our system is the accelerometer. We are using an MPU-6050, a simple but effective choice for collecting acceleration data in the x, y, and z directions. With the x, y, and z acceleration data from the accelerometer, the Raspberry Pi immediately calculates pitch and roll in the accelerometer thread. It then calculates changes in pitch and roll by keeping track of the previous values recorded. Analyzing changes in pitch and roll helps the system determine whether the bike is shaking too much. All the data for the change in pitch and role is recorded to a CSV file called "mpu_date_log.csv" that is later used in our machine learning model. With our machine learning model, we are able to pick the best threshold value that sets off the buzzer. This threshold was informed by preliminary data of shaking the device. The threshold value is 10 for change in pitch and 15 for change in roll.

The buzzer is a 110dB Piezo buzzer. It is connected to the Raspberry Pi 5V pin and is controlled via a MOSFET connected to a GPIO pin on the Raspberry Pi. We chose this buzzer for its loud noise, small size, and small power consumption. Once the buzzer is triggered due to a detected theft attempt, a notification is sent to the user, who can start or stop the buzzer using a button on the interface. Otherwise, the buzzer will automatically stop when the shaking is no longer detected.

Machine Learning Model

We have made a machine learning model coded in Python and executed in Google Colab to have a way of providing accurate and real-time security. Making use of our accelerometer to classify between theft and non-theft attempts, we created and trained a linear classification model that leverages logistic regression and learns to differentiate between normal bike interactions and potential theft attempts. Our data collection is done in three different datasets capturing static, light shaking, and hard shaking scenarios, with hard shaking specifically indicating suspicious activity. The purpose of writing the x, y, and z acceleration and pitch and roll values to a CSV file is to use this data to train our machine learning model.

The current system has 380 data samples, labeled as 0 and 1, to train our machine learning model. Through our model, we extract key features like movement magnitude, rotation magnitude, and angular displacement, so we can accurately identify potentially dangerous scenarios. Using a logistic regression model with standardized features, we've achieved an impressive 88% accuracy in detecting theft-like movements. The model demonstrates high precision, with 89% accuracy in identifying normal bike movements and 86% accuracy in detecting potential theft attempts. Our testing shows the system correctly identifies 49 normal activities with only 3 false alarms and successfully detects 18 potential theft scenarios. While not perfect, the system provides a robust first model as our base.

A snippet from our code used for training the model using the standard scaler:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model = LogisticRegression(random_state=42)
model.fit(X_train_scaled, y_train)
```

User Interface

As part of the safety suite, our device offers a live camera video feed on the user interface. The main purpose of this feature is to allow users to confirm that a bike theft is actually occurring or otherwise identify false positives. To implement this, we used the Raspberry Pi Camera module V3 NoIR Wide that is connected to the Raspberry Pi 4 via a ribbon connector. This camera has a wide lens view and a NoIR filter. This helps enhance the video stream for users by giving wider and clearer visibility. Using Python and Flask, we are able to collect video data, convert video into discrete images, and send it to a Flask server in real-time. The video feed is converted to discrete images for better compatibility with embedding into the user interface.

The Python script accelerometer.py runs a camera collection thread that runs in parallel to the accelerometer thread. This thread uses the picamera2 library for Python to start collecting video data and sending it to a Flask server hosted by the Raspberry Pi. This camera is turned off until the user turns the key lock switch to an on position to save power.

Our system is powered by a portable charger that is intended to charge tablets and phones. The portable battery supplies 5V to the Raspberry Pi. The user can see the battery level directly on the device via the battery's LCD. We accomplish this by adding a hole into the enclosure where the battery and LCD screen are located and covering the hole with clear acrylic. In this way, the device is still waterproof. The acrylic is covered by a privacy screen, which means that you can only see the battery level if you are right in front of it.

To get started with BikeGuard, users can authenticate through our React-based web interface using either email/password credentials or Google OAuth integration provided by the @react-oauth/google library. User authentication data is securely managed through Firebase Authentication, with account details and preferences stored in Firestore collections for persistent access across devices.

When the accelerometer detects suspicious movement exceeding our threshold values (10 for pitch change and 15 for roll change), the Python backend triggers the GPIO-controlled piezo buzzer and simultaneously dispatches a notification to our Flask server. This server, implemented in backend/app.py, handles both REST API endpoints and Socket.IO events for real-time communication. The React frontend, built with functional components and hooks, receives these notifications through either WebSocket connections or RESTful API polling, depending on network conditions.

The interface layout is structured in a responsive grid using CSS Flexbox and Grid. The real-time MJPEG camera feed from the Raspberry Pi Camera Module V3 is embedded at the top-left of the interface through the /api/video-feed endpoint, which leverages the picamera2 library to capture and stream images. Adjacent to this is the live Google Maps integration displaying real-time GPS coordinates, implemented using the google-maps API with custom styling and a pulsing location indicator.

Below the map, three primary control buttons are prominently displayed: "Sound Alarm" triggers a POST request to /api/trigger-alarm, "Stop Alarm" calls the /api/stop-alarm endpoint, and "Tracking On/Off" toggles the GPS tracking functionality by updating user preferences in Firestore. The notification feed at the bottom of the interface renders events stored in our SQL database, sorted chronologically and styled according to notification type.

The entire system is designed with offline resilience, using localStorage for credential caching and a service worker for Progressive Web App functionality. Error handling is implemented throughout the codebase to manage network interruptions gracefully, with user-friendly error states and automatic reconnection logic.

5 Operation of the Project

For the manufacturing of BikeGuard, we took into consideration multiple engineering standards to ensure the safety and reliability across its hardware and software.

Electrical and Safety Standards

The electrical components in our project prioritize functionality and safety. We made sure that our circuit was correctly wired to avoid any short circuiting. We also make sure to solder all components together and utilize heat shrinks to prevent short circuits. To prevent the circuitry from overheating we added head sinks to the main chips of the Raspberry Pi.

When designing our circuitry and picking components, we decided to prioritize battery efficiency to allow the user to use BikeGuard as long as possible. To power our device, we decided to use a rechargeable battery. The system's rechargeable battery architecture follows IEEE 1625, which provides guidelines for battery performance and safety. This standard ensures that the battery will not overheat and that the battery life and usage will be optimized. This battery is also isolated, meaning that it has its internal measures in place to handle battery issues that might arise.

For the safety of our electrical components in our product and considering that BikeGuard is mainly for outdoor usage, we made sure to design a waterproof housing. BikeGuard meets the IEC 60529 Ingress Protection rating requirements, ensuring resilience against dust, water, and extreme environmental conditions.

Wireless Communication and Internet Standards

The backbone of BikeGuard is connectivity; without it, our components wouldn’t be able to work together so seamlessly. Our system’s Wi-Fi network ensures secure and efficient data transmission. It also ensures a stable connection, as well as being efficient for our battery life. We also have a Bluetooth connection to pair the device with the front-end interface.

Software Development and Cybersecurity Standards

BikeGuard implements robust security using Firebase Authentication with JWT (JSON Web Tokens) for secure credential verification. Our API communications use TLS 1.3 encryption and employ CORS (Cross-Origin Resource Sharing) headers to prevent unauthorized access. The Flask backend uses parameterized SQL queries to prevent injection attacks, while React components implement proper state management with useEffect cleanup functions to prevent memory leaks. For offline functionality, we utilize the browser's IndexedDB API through localStorage, with a service worker implementing a cache-first strategy for essential assets.

Regulatory and Governmental Compliance

As a project designed and developed for a senior design project, BikeGuard is not currently subject to commercial regulatory requirements or compliance. However, our design principles anticipate future compliance needs by respecting user privacy and data protection standards. We've designed BikeGuard with ethical considerations in mind, ensuring that the device is specifically for bike security purposes rather than general surveillance. Our system only monitors the user's own property with their explicit consent, and we've implemented data minimization principles to collect only what's necessary for the security function.

6 Cost Breakdown

The total cost reported in table 6.1 is the estimated cost to produce a BikeGuard device given the current design. The majority of the cost of the device comes from the hardware components, along with some mechanical components that support enclosure. Any software or cloud computing costs as well as physical wires used for this project were negligible.

BikeGuard is a commercial product that would normally be produced at scale, for this reason actual retail price would likely be much lower to reflect economies of scale. The Raspberry Pi 4, the camera unit, and the cellular networking modem comprise most of the cost of the device. We believe in future designs we can replace these components with much lower priced alternatives. With a complete prototype, the BikeGuard team can create customized hardware that cuts out unnecessary features of current hardware components.

Item	Description	Quantity	Unit Cost	Total Cost
1	Raspberry Pi 4 Model B	1	$42.00	$42.00
2	INIU Portable Charger	1	$23.29	$23.29
3	110dB Piezo Buzzer	1	$3.00	$3.00
4	64GB SD Card	1	$9.74	$9.74
5	MPU-6050 Accelerometer	1	$5.99	$5.99
6	Raspberry Pi Camera Module V3 NoIR Wide	1	$44.18	$44.18
7	USB-C Male to USB-C Female Cable	1	$6.99	$6.99
8	Raspberry Pi Heat Sink	1	$4.97	$4.97
9	USB A Male to USB C Female Cable	1	$7.99	$7.99
10	Key Lock Switch	1	$9.99	$9.99
11	IRLZ44N Mosfet	1	$9.99	$9.99
12	3D Printed Enclosure	1	$24.17	$24.17
13	Solderable Breadboard	1	$1.60	$1.60
14	Hose Clamps	1	$1.75	$3.50
15	SixFab Raspberry Pi Cellular Modem Kit	1	$135.00	$135.00
16	Prepaid SIM card	1	$20.00	$20.00
Beta Version-Total Cost				$352.40

Table 6.1 Production Cost Estimate for BikeGuard

7 Appendices

7.1 Appendix A - Specifications

Team 8
Team Name: BikeGuard
Project Name: BikeGuard

Mechanical Requirements	
Attach securely to most bikes and scooters	✓
Resistant to 15 lbs of force	✓
Not obstructive to bike riding and scooter activities	✓
Weatherproof to moderate rain and snow	✓
< 15 x 15 x 15 cm, < 2 lbs, portable	✓
Power Requirements	
Lasts for active working time of ~ 8 hours	✓
Rechargeable batteries	✓
Onboard Software Requirements	
Correctly classify between theft and non-theft attempts 85% of the time with ML model	✓
Constantly scan for peripheral data when theft detection mode is on	✓
Communicate with remote device via wifi	✓
Communicate with remote device via cellular data	✓
Use networking capabilities to convey location to remote user	✓
Remote Software Requirements	
Mobile app interface	✓
Alert user of theft attempts	✓
Stream camera data in case of theft attempt	✓
Display map with bike location	✓
Control device remotely	✓

7.2 Appendix B - Team Information

The BikeGuard team consists of Marissa Ruiz, Margherita Piana, Bennett Taylor, Albert Zhao, and Beren Donmez. All members of the team are seniors studying computer engineering at Boston University and will be graduating in May of 2025. The device outlined in this document was built by the team to demonstrate mastery of engineering principles as a part of their senior design course (EC 463/463), where they were team 8 (BikeGuard). The team chose to build a bike theft detection system because of collective experiences with bike theft while attending college in Boston. After researching the problem of bike theft, the team decided that it was an important issue to help deter, leading to the development of BikeGuard. Throughout the project, each of our team members improved their software, hardware, and engineering design skills. The result of this development can be seen in our final product.

Bennett Taylor will be continuing his education at Boston University by pursuing a M.S. in Electrical and Computer Engineering. He has a passion for low-level programming, mathematics, and optimization which he plans on using in the healthcare or medical device industries.

Margherita Piana will be continuing her education at Northeastern University by pursuing a M.S. in Cybersecurity. She has a passion for network security, cryptography and IoT and embedded systems security.

Marissa Ruiz will enter the workforce as an Embedded Software Engineer following her graduation. She enjoys embedded systems, computer networking, and machine learning, and hopes to explore the industry and find her niche.

Albert will take a gap year to pursue potential engineering career opportunities. In the meanwhile, he wants to study for the LSAT to seek and explore new interests in law.

Beren Yağmur Dönmez will be continuing her education at the London School of Economics with an MBA. She is really passionate about machine learning, AI and business and management.

Figure 7.1 Group Picture with All Team Members
From left to right: Bennett Taylor, Marissa Ruiz, Albert Zhao, Margherita Piana, Beren Donmez