Boston University
Electrical & Computer Engineering
EC463 Senior Design Project

First Semester Report

Submitted to

Ryan Lagoy
rclagoy@bu.edu

by

Team 8
Bike Guard

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

Submitted: December 7, 2024

Table of Contents

Executive Summary

Team 8 – Bike Guard

BikeGuard offers a comprehensive theft detection system designed to protect the 80% of bike owners affected by theft. Combining hardware components such as a Raspberry Pi Zero 2 W, an MPU-6050 accelerometer, a piezo buzzer, and a Raspberry Pi Camera Module V2 in a durable, inconspicuous lockbox-mounted setup, 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 future plans to integrate cellular SIM cards for enhanced connectivity and GPS tracking. The user interface offers a web-based platform built with Flask and React, featuring live video streaming and push notifications triggered by motion-detection sensors. Early prototype tests have validated the hardware's functionality, reliable alert systems, and data logging for iterative model improvements. Our future improvement plans include embedding the ML model directly on the Raspberry Pi, transitioning to a mobile app interface, and 3D printing our enclosure design to blend in as a bike accessory such as a bike light. With its innovative approach, BikeGuard goes beyond traditional locks to offer cyclists a scalable, intelligent solution for theft prevention, reimagining bike security with real-time technology and advanced analytics.

1.0 Introduction

Bicycle theft is a pervasive issue, with an estimated 175,000 bikes reported stolen annually in the United States alone—and countless more thefts unreported (Chub). 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 integration by making use of a Raspberry Pi Zero 2 W, an accelerometer, a buzzer, and a camera housed in a durable enclosure. 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 current sample product, we achieve connectivity and interface through a web-based user interface built on Flask and React provides live updates, video feeds, and notifications to the user. 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. Future plans include GPS tracking, mobile app integration, and enhanced enclosure design. 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.

2.0 Concept Development

2.1 Concept in Terms of Engineering Understanding and Perspective

To understand the issue of bike theft through an engineering perspective, we looked at current solutions and identified what is missing. We noticed that they were unable to monitor bikes in real-time, they lacked immediate communication to the user in case of bike theft attempts, and they lacked recourse in the event of a bike theft attempt. So, from our perspective as engineers, we addressed these shortcomings in a novel device, BikeGuard. Our device offers real-time motion and tamper detection with minimal false positives, a reliable communication system to notify users immediately, and durable, lightweight design to withstand environmental conditions and disguise theft-prevention hardware as a standard accessory.

2.2 Engineering Requirements

Our engineering requirements are visible under appendix 1.

2.3 Conceptual Approach

Our innovative solution integrates advanced hardware, intelligent software, and machine learning to create a robust, proactive theft prevention system specifically designed for mobile assets like bicycles.

Our hardware is designed to detect bike lock or component tampering via shaking, cutting, lock-picking, or brute force. Shaking data is recorded from an accelerometer (MPU-6050). This data is converted to pitch and roll by our microcontroller, a Raspberry Pi Zero 2 W. When a theft attempt is detected by the machine learning model, the microcontroller sets off a buzzer to scare away thefts. It also will send a post notification to a back end server that communicates with the front end user interface. Our device also boasts a camera to stream live video and record theft attempts.

Our machine learning model is a simple logistic regression model that processes sensor data to classify activities as theft attempts or benign movements with 88.16% accuracy. It will live on the Raspberry Pi Zero 2 W and classify pitch and roll data as theft or non-theft attempts.

We currently offer a web-based user interface built using Flask and SQL (backend) and React, Javascript, and CSS (frontend) to provide live video streaming, notifications, and activity logs. This is the user's hub for monitoring their bike.

To connect our device to the internet to communicate with remote users we currently are utilizing Wi-Fi, with future plans to add cellular connectivity via SIM cards for a broader range and GPS integration for bike tracking.

Our enclosure is designed to be easily mountable to most bikes and scooters. We are currently using a pre-made enclosure purchased from Amazon, but we plan to 3D print a custom enclosure in the future. We aim to make this enclosure as small and inconspicuous as possible. We are also considering disguising the device as a bike component, such as a bike light.

2.4 Why We Chose This Concept

More than half of our team members own bikes and have had concerns about their bikes' safety. We understand the importance of the affordable and effective mode of transportation that bikes offer. The problem of bike theft was an issue that we wanted to tackle, especially living in a big city.

At the start of the semester, we had numerous ideas, such as Alzheimer's glasses and natural disaster communication. However, our initial market research revealed significant gaps in the bike lock market. It became clear that many features we would personally value in a smart lock were missing.

We toyed with the idea of adding proximity sensors to our device to capture further bike-theft activity, but in the end decided they were not necessary. We also looked into adding a motor that can sweep the camera back and forth across the housing to get a better range of view. We decided against this because it would make our device bigger than we wanted it to be. We also considered making the device an actual bike lock in addition to a bike-theft detection device. We decided against this because we lack the mechanical engineering skills required to make a competitive bike lock. We also looked into making this device mountable to the lock and not the bike itself.

3.0 System Description

Our system will be easy to use and mount. Our system will be mountable on any part of the tubing of the bike with two screws and a U shaped clasp. Our system will be detachable for user adaptability on the bike. The housing will be compact, at least less than 30cm x 30cm x 30cm, and weigh less than 8 lbs. This threshold ensures that the device is unobtrusive to the bike's normal functionality. Additionally, the housing will be durable and robust, capable of withstanding up to 15lbs of force. We plan to make the housing small and obscure such that potential thieves won't perceive the device on the bike. In this way, we can outsmart a thief who can otherwise destroy the device with brute force. We have also discussed disguising the device as a bike component, such as a bike light. Our product will utilize machine learning to detect theft-like activity by analyzing accelerometer data collected from the bike. Specifically, we will employ a linear classification approach using a logistic regression model, trained to distinguish between normal movement patterns and those indicative of theft.

Our device will be controlled by a Raspberry Pi Zero 2 W microcontroller. We picked this microcontroller for its wifi and camera capabilities. The Raspberry Pi Zero 2 W is flashed with the Raspberry Pi OS 32 Bit (Legacy) Debian Bullseye operating system. We chose this specific legacy operating system for its ease of use with the Raspberry Pi camera module. We additionally flashed the SD card with our router's SSID and password. Once the Raspberry Pi boots, it will automatically connect to our network. We are using a simple TP-Link router to connect our components to our network. The router is plugged in upstream to the BU network. In the future, we plan to add a SIM card to the microcontroller to allow for cellular data streaming, potentially using a Hardware Attached on Top (HAT) adapter. This will be investigated more next semester. To prevent overheating on the Raspberry Pi Zero 2 W, we have attached small heat sinks to the controller’s main chip components.

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. The MPU-6050 also collects gyro data in the x, y, and z, directions, but we ended up not needing these values. From the x, y, and z accelerations, the Raspberry Pi is able to calculate the MPU-6050’s pitch and roll. As it stands, we have hardcoded a pitch and roll threshold value that sets off the buzzer and sends a post request to the backend server. This threshold was informed by our preliminary data of shaking the accelerometer. The threshold value is 5 for pitch and 10 for roll. The buzzer is a 90 dB Active Piezo Electronic Buzzer. It is connected to the Raspberry Pi via GPIO connection. We chose this buzzer for its loud sounds, small size, and small power consumption. For our prototype we have not utilized the full sound potential for this buzzer. In the future we hope to utilize the full 90dB. We also plan for buzzer control to be optionally offloaded to the user when desired.

To run the onboard software, we utilized node.js and python3. The main encapsulating code is a node.js file called “log_mpu_data.js.” This code creates a file called “mpu_data_log.csv” and then calls a python script called “accelerometer.py” as a child process. “Accelerometer.py” interacts directly with the accelerometer to collect x, y, and z acceleration values and then convert them to pitch and roll values. The x, y, and z acceleration values in addition to the pitch and role values are then printed to the console in CSV format. The encapsulating “log_mpu_data.js” code then grabs these values printed to console using the “stdout.on” function. From there, it uses the “fs.write” function to write these values into the “mpu_data_log.csv” file. When the accelerometer pitch and roll values reach a certain threshold, the “accelerometer.py” code sets off the buzzer and sends a post notification to the backend flask server.

We have made a machine learning model coded in Python and executed in Google Colab in order 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 of 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)
```

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 Module V2 that is connected to the Raspberry Pi Zero 2 W via ribbon connector. Using python and flask, we are able to collect video data and send it to a flask server real time. The python script uses the picamera library for python to start collecting video data. Ideally, this camera will be off most of the time. It will only be af=ctivated when the model classifies a theft attempt. In the future, we plan to integrate two cameras for better visibility. This, however, wil be challenging because each Raspberry Pi Zero 2 W only supports one camera module. We plan to look into more solutions, but our current idea involves using two Raspberry Pi Zero 2 Ws.

Our system is currently powered by a portable charger intended to charge tablets and phones. The portable battery supplies 5V to the Raspberry Pi. In the future we hope to move to replaceable alkaline batteries. We might discuss using a lithium ion battery with our client, but given the safety issues that arise with its use we might pass.

When the buzzer goes off, a notification is posted to a back end server that is hosted using flask. The front end grabs this information and displays it to the web UI. The front end also embeds the Raspberry Pi Camera feed server. The camera feed is displayed to the top of the web UI page, followed by notifications from the device. The web UI is made using React and Javascript. It also uses CSS for formatting. The back end is implemented using Flash and SQL for data storage. In our final product we expect the front end to be a phone application so that notifications can be sent.

See Appendix 3, 4, and 5 for Software Block Diagram, Onboard Software Block Diagram, and Hardware Block Diagram respectively. See Appendix 6 for sample web UI interface without the embedded camera.

4.0 First Semester Progress

4.1 Concept

Aug 2024 - Concept Creation

- Throughout the month of August we spent a lot of time coming up with a concept for our product. We were very interested in working with machine learning and embedded devices. Our teammate Albert had bought a bike at around this time and came up with this initial idea. We wrote up a project proposal, and Albert called the project "BikeGuard"

Oct 8, 2024 - PDRR

- Our first assignment regarding our senior design project was the PDRR. We elected Marissa as our team leader and she asked everyone to write a problem statement for our project. We looked into bike-theft statistics and current competing products. Specifically, we researched the main failures of traditional bike locks such as component theft, lock picking, and pure brute force. We also discussed deliverables, engineering requirements, and budget. We then split up the PDRR assignment. Finally, Marissa came up with the product logo which can be seen on the cover page.

Oct 22, 2024- Shark Tank

- For us, Shark Tank was about getting feedback from professionals more than anything else. We reused our slides from the PDRR with the addition of our first hardware and software block diagram concept. We presented to 4 industry professionals and got some good advice. We were told to really think about the customer. Are they individuals who splurge on a bike? Or do they buy the cheapest one at a second-hand store? Are they professional cyclists? Depending on our audience, our product definition could change. We were also asked to consider if we really needed to include a machine learning model. One professional told us that this could easily be implemented without the hassle of creating a machine learning model. Because of our various components, we were told to think about our power budget early on. Finally, we were told that our project is ambitious and that we should start looking into doing various sprints to get everything done

Oct 29, 2024 - First IDR with Prof. Osama

- Met with Professor Osama and GST Nolan Vild for our first IDR. We talked about the IR sensor, accelerometer, and Raspberry Pi camera. Osama mentioned that our project has a lot going on and that it was hard for him at first to see it holistically. He also didn't understand the purpose of the IR sensor. He recommended that we think about our project more holistically to help clients understand it better. Nolan told us that we should order our enclosure on Amazon instead of trying to 3D print it. Osama agreed.

Nov 5, 2024- Second IDR with Prof. Hirsch

- Met with Prof. Hirsch for our second IDR. We talked with him about data collection, genuine or ingenuine bike theft data, and accuracy. We decided that we don't actually need to steal a bike to collect data for the machine learning model. We just need to mimic bike theft. Our teammate Beren had found some sample videos on YouTube of bike theft. We agreed to analyze these videos in order to inform our model. We decided to aim for an 80% accuracy with the machine learning model

Nov 7, 2024- Client meeting with Ryan

- This was our first meeting with our client Ryan Lagoy. After a quick introduction, he told us that we might be better off switching to just a Raspberry Pi instead of using an ESP32 and a Raspberry Pi as was originally planned. He said that we should always try to make things simplest, and having two microcontrollers didn't seem beneficial if we could do the same with one. We also talked about not implementing the IR sensors. This to us was notable because Prof Osama had told us the same thing about a week prior. Although we had already bought them, we realized they were big and probably not more informative than the accelerometer. Additionally, to get a full coverage we would need at least 4 IR sensors which would make our product bigger. Our team agreed to scrap the IR sensors for now. We also talked about camera placement. We had only planned on integrating one camera and we weren't sure how to get the most coverage. Ryan suggested putting the camera on a bendable chord that the user can adjust. Kind of like a reading lamp. We also talked about implementing two cameras. Finally, Ryan suggested that in the future we 3D print our enclosure to personalize it. Our current enclosure was too big, bulky, and noticeable.

Nov 12, 2024- Third IDR with Prof. Pisano

- This was our final IDR before our demo. We talked with Prof. Pisano about what was feasible for us on demo day. Prof. Pisano encouraged an end-to-end product and integration of back and front end. He also encouraged us to think of edge cases. What if someone smashed our product with a hammer? What if someone put aluminum foil over the product to cut off the network connection?

Nov 19, 2024 - Demo Day

- Ryan was happy with our product and impressed with our efforts. Our device was able to collect accelerometer data and save it to a CSV file. It also was streaming the camera data onto our front end. The device successfully set off the buzzer when we shook the device. The front end received alerts when the buzzer was set off. The camera stream on the front end was large with a small delay. We talked about moving away from a portable charge and using replaceable alkaline batteries. Ryan and GST Luca said that overheating would not be too much of an issue, especially with heat sinks. We also discussed moving to a smaller enclosure, perhaps one that would blend into the bike. Finally, we discussed moving away from a website and towards a mobile application.

4.2 Hardware

Oct 19, 2024 - Initial component purchase

- We made our first component purchase. First and most importantly for us, we ordered the housing. This was a pre-made bike lock-box. We also purchased a Raspberry Pi Zero w 2, a Piezo buzzer, an MPU-6050 accelerometer, a Raspberry Pi Camera Module V2, and four Sharp IR sensors.

Oct 30, 2024 - Accelerometer connection to ESP32

- We connected the accelerometer to the ESP32 using I2C and collected preliminary shaking and banging data on the accelerometer. We collected x, y, and z acceleration and pitch and roll data. This data was saved into 4 labeled CSV files: no shaking, slight shaking, heavy shaking, and heavy banging.

Nov 14, 2024 - Soldering, flashing, connection, and assembly

- This was the day where a lot of things came together. After soldering on the pins to our Raspberry Pi incorrectly, we tried to de-solder them, and we thought we fried our board. We had to place a new order for another Raspberry Pi just in case. Ryan inspected the board and found it was fine, just had some superficial burns. He helped us desolder our Raspberry Pi and showed us proper soldering techniques. The most important thing was to make sure the solder was flushed in the pins. After his tutorial, our teammate Marissa was able to solder pins onto both Raspberry Pi's and the accelerometer. We flashed the Raspberry Pi's SD card with our router's SSID and password so that it would be connected to our network. We connected our accelerometer to the Raspberry Pi using I2C communication. Finally, we spent some time trying to SSH into the Raspberry Pi. After we were successful, we wrote a quick python script for accelerometer collection.

Nov 17, 2024 - Node.js and accelerometer data

- Our teammates Margherita and Marissa were able to use Node.js to run the python script and collect the accelerometer values to write to a CSV file. We were struggling with the Raspberry Pi Camera.

Nov 18, 2024- Successful camera connection and stream and buzzer integration

- Our teammate Marissa was able to connect to the Raspberry Pi camera by flashing an older Raspberry Pi OS to the Raspberry Pi. She then used Flask to stream the video to a local server on our network. Our teammate Margherita was able to connect the buzzer to the circuit using a GPIO connection. She looked through our previous accelerometer data and wrote in a hard-coded threshold that would set off the buzzer. This was to accomplish end-to-end functionality for the demo.

4.3 Software

Oct 20- Nov 16, 2024 - Creating interface

- After purchasing the initial components, our teammates Albert and Ben got to working on the user interface. They spent about a month creating the front end and back end. They created the front end with Javascript and React and used CSS for formatting. They used Flask to connect it to the backend. The back end was created using Flask and SQL for data storage. The web UI shows the video stream and alerts sent from the embedded device.

Nov 17, 2024 - Integration of embedded device and website UI

- Our teammates Albert and Marissa integrated the embedded device and the web UI. Albert added a post notification to his Flask backend server to be sent when the buzzer goes off. This data was then picked up and displayed by the front end. He also embedded the camera stream Flask server to the front end web UI.

4.4 Machine Learning Model

Oct 30, 2024- Collecting initial data

- Our teammates Beren, Margherita, and Marissa collected initial accelerometer data from the MPU-6050 and saved it to a CSV file. We collected x, y, and z acceleration and pitch and roll data. This data was saved into 4 labeled CSV files: no shaking, slight shaking, heavy shaking, and heavy banging. Teammates Beren and Marissa analyzed this data and found that the most significant difference between each type of data was the variance of the pitch and roll.

Nov 1 - Nov 10, 2024 - Creating initial model

- Our teammate Beren spent about a week creating and tweaking our initial classification model. The model was a logistic regression model that classified theft based on accelerometer data. The model assigns labels “0” for non-theft attempt and “1” for theft attempt. She extracted some features from the data using the extract_features() method to help with classification. She then trained and tested her model. Her model yielded a classification accuracy of 88.16%. This was exceptionally higher than we had hoped for. She found there were 49 true positives, 3 false positives, 6 false negatives, and 18 true negatives.

Nov 11, 2024 - Getting model feedback

- Our teammate Beren communicated with BU ECE Professor Ashok Cutkosky to get some feedback on her model. He said everything looked great and that he only recommended adding some boosting algorithms to make our learning agent stronger. We plan to implement this next semester

5.0 Technical Plan

Task 1. Housing Prototype
A housing or container shall be designed and 3D-printed to accommodate our hardware components. It should have cutouts to allow our cameras a clear view of the bike's surroundings. It should include secure pockets for the Raspberry Pi, accelerometer, buzzer, and battery. The housing needs to be compact and easily fitted to any bike model. It should also be "hidden."
Lead: Albert Zhao, Margherita Piana, Bennett Taylor, Beren Donmez

Task 2. Cellular network integration
We aim to integrate cellular network connectivity into our system to enable its use without relying on Wi-Fi. Currently, our prototype functions exclusively when connected to Wi-Fi. To expand the system's flexibility and functionality, we plan to purchase a prepaid cellular card and assess its performance. Specifically, we will evaluate how well the cellular connection supports the system, including the video stream and front-end interface. Additionally, we will monitor data usage and check the reliability and stability of the cellular connection over time.

Task 3. Machine Learning Movement Detection Model
Our machine-learning model must be thoroughly tested to evaluate its accuracy in detecting potential bike theft movements. The testing process should include various scenarios to ensure the model reliably distinguishes between normal bike activity and theft-related movements. A

critical metric is maintaining a false-positive rate (incorrectly detecting theft movement when there is none) of ideally less than 20%.
Lead: Beren Donmez, Albert Zhao

Task 4. UI Improvements
Our website and app need to be intuitive and easy to use while preserving overall functionality. Key tasks include implementing user login and authentication features and setting up a database to securely store user information, allowing it to be retrieved as needed. Additionally, the live camera feed on the website must be optimized so it doesn't occupy the entire screen. The live notification section also requires enhancements to include animations and other interactive features and improve user experience. We also need to integrate a live location tracking feature, enabling users to monitor the real-time location of their bikes.
Lead: Albert Zhao, Ben Taylor

Task 5. Geo-location Tracking
We need to integrate a geo location tracking into our device and connect it to the Google Maps API so that users can track their stolen bike's location in real time. We plan to get geo location from cellular network data instead of from a GPS module. This is to ensure that our enclosure stays small. This feature is critical for providing users with accurate and reliable location data in the event of a theft.
Lead: Margherita Piana, Marissa Ruiz

6.0 Budget Estimate

Item	Description	Amount	Cost
1	Raspberry Pi Zero 2 W	2	$40
2	Raspberry Pi Camera Module V2	2	$30
3	90dB Active Piezo Electronic Buzzer	2	$10
4	MPU-6050 Accelerometer	2	$20
5	32 GB SD Card	2	$8
6	Enclosure	2	$20
7	Raspberry Pi Zero 2 W Heat Sinks	2	$10
8	TP-Link Router	1	$30
10	Micro USB to USB-C adapter	2	$16
11	Small Breadboard	2	$14
12	INIU BI-B61 Portable Charger 22.5W 10000mAh Power Bank	1	$20
13	Alkaline Batteries	4	$5
14	Hardware Attached on Top (HAT) SIM card adapter	2	$40
15	3 Month Prepaid SIM card for cellular data	2	$60
	Total Cost		$324

For our budget estimate we took into consideration all the required components to create two functional BikeGuard devices. The cost estimate is composed of the Cost of Goods (COGs) from our prototype, rounded up to account for variability in costs, and expected future purchases until May 2025. Of course, it is hard to foretell what our needs will look like in the coming months, but this is a rough estimate. The main expenses come from the Raspberry Pi Zero 2 W, the Raspberry Pi Camera Module V2, the HAT SIM Card adapter, and the 3 month prepaid SIM card. We are willing to splurge on the Raspberry Pi Zero 2 W because it has camera module and wifi capabilities. It is also smaller than a traditional Raspberry Pi. The camera module is a necessary evil in our budget as it is the best camera compatible for the Raspberry Pi Zero 2 W. The cellular data components ended up denting our budget more than we had anticipated. For each BikeGuard to have cellular data connectivity, we need to purchase two hardware attached on top (HAT) SIM card adapters and 2 three month prepaid SIM cards. We are willing to splurge here too because cellular data usage is a highlight of our product.

As expected, our budget estimate is subject to change in the future. Our most variable cost is that of the enclosure. The estimate is based on the current bike lock box we used in our prototype, but in the future we hope to 3D print our own enclosure, potentially changing costs.

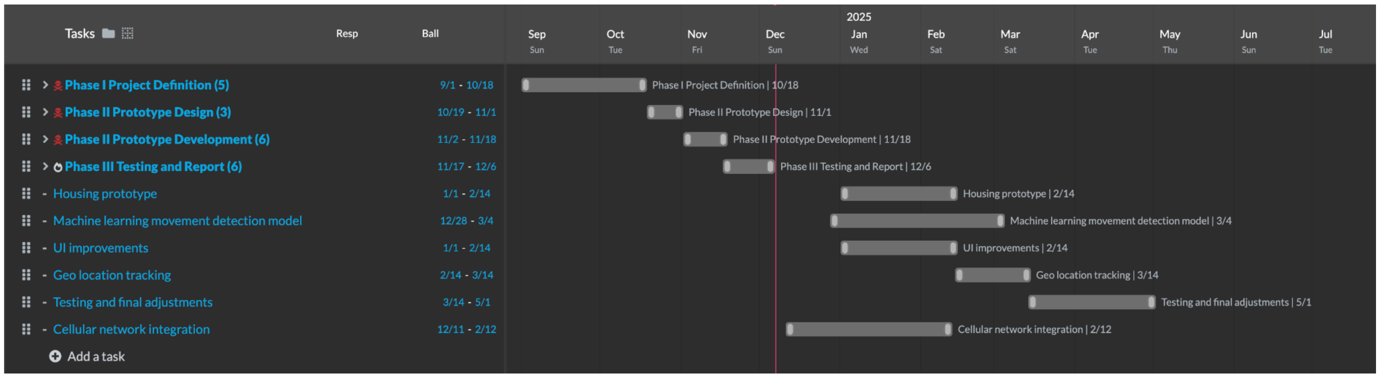
One of the constraints for our device is keeping costs competitive with similarly priced competing products. High quality bike locks often have a price around $100, so the goal for our device is to be around that price point. It is notable however that in a production environment, there would likely be lower costs, as part orders could be done in bulk.

7.0 Attachments

7.1 Appendix 1 – Engineering Requirements

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	❌
< 35 x 35 x 35 cm, < 8lbs, portable	✅
Power Requirements	
Lasts for active working time of ~ 24 hours	❓
Replaceable batteries	❌
Rechargeable batteries	✅
Onboard Software Requirements	
Correctly classify between theft and non-theft attempts 90% 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	❌

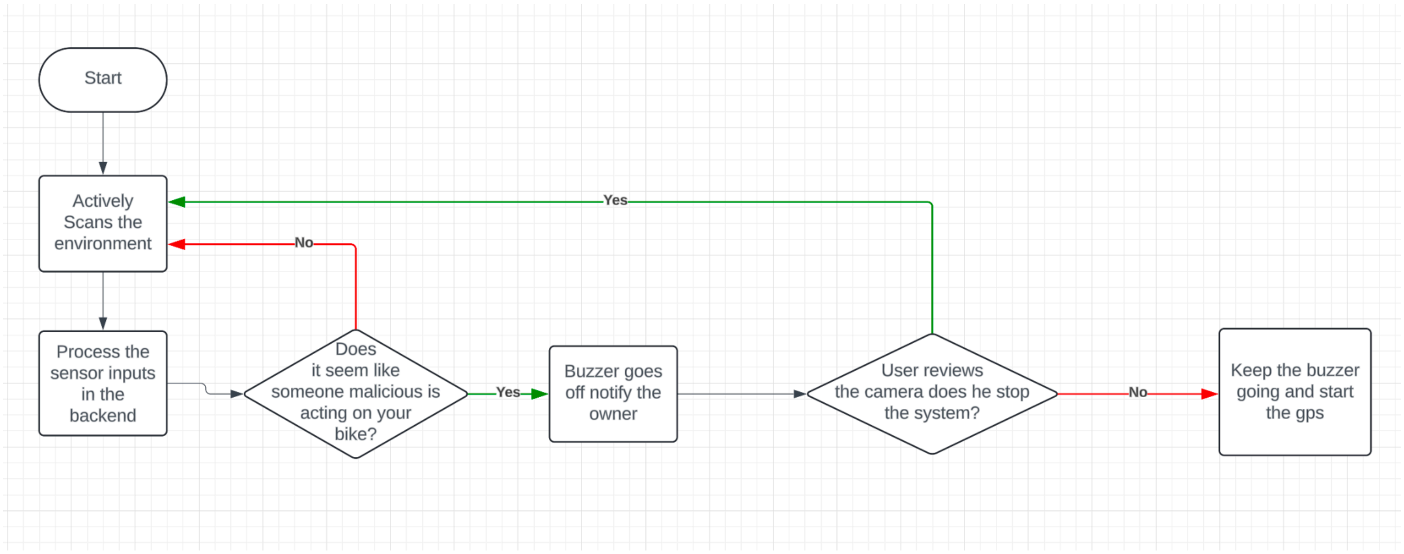
7.2 Appendix 2 – Gantt Chart



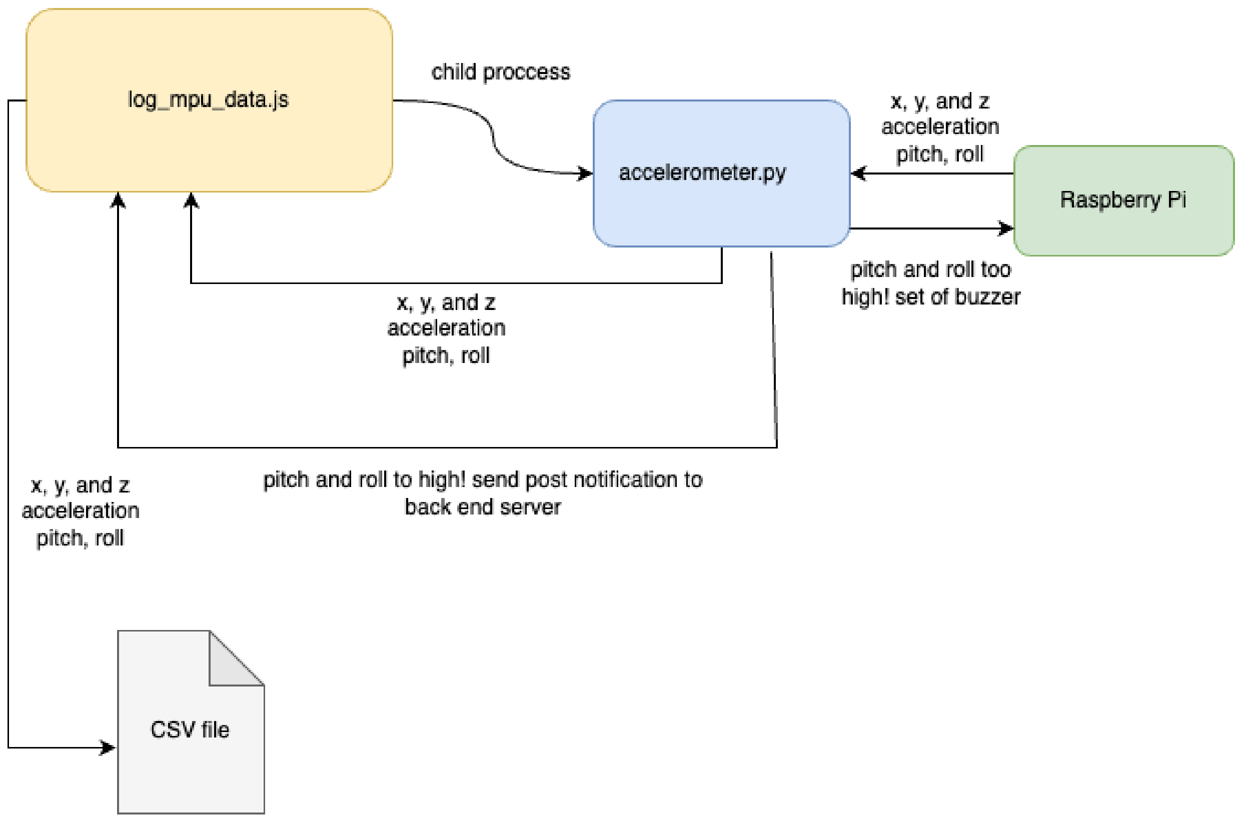
Tasks	Resp	Ball	Dates
Phase I Project Definition (5)			9/1 - 10/18
Phase II Prototype Design (3)			10/19 - 11/1
Phase II Prototype Development (6)			11/2 - 11/18
Phase III Testing and Report (6)			11/17 - 12/6
Housing prototype			1/1 - 2/14
Machine learning movement detection model			12/28 - 3/4
UI improvements			1/1 - 2/14
Geo location tracking			2/14 - 3/14
Testing and final adjustments			3/14 - 5/1
Cellular network integration			12/11 - 2/12
Add a task			

2025
Sep Sun
Oct Tue
Nov Fri
Dec Sun
Jan Wed
Feb Sat
Mar Sat
Apr Tue
May Thu
Jun Sun
Jul Tue
Phase I Project Definition | 10/18
Phase II Prototype Design | 11/1
Phase II Prototype Development | 11/18
Phase III Testing and Report | 12/6
Housing prototype | 2/14
Machine learning movement detection model | 3/4
UI improvements | 2/14
Geo location tracking | 3/14
Testing and final adjustments | 5/1
Cellular network integration | 2/12

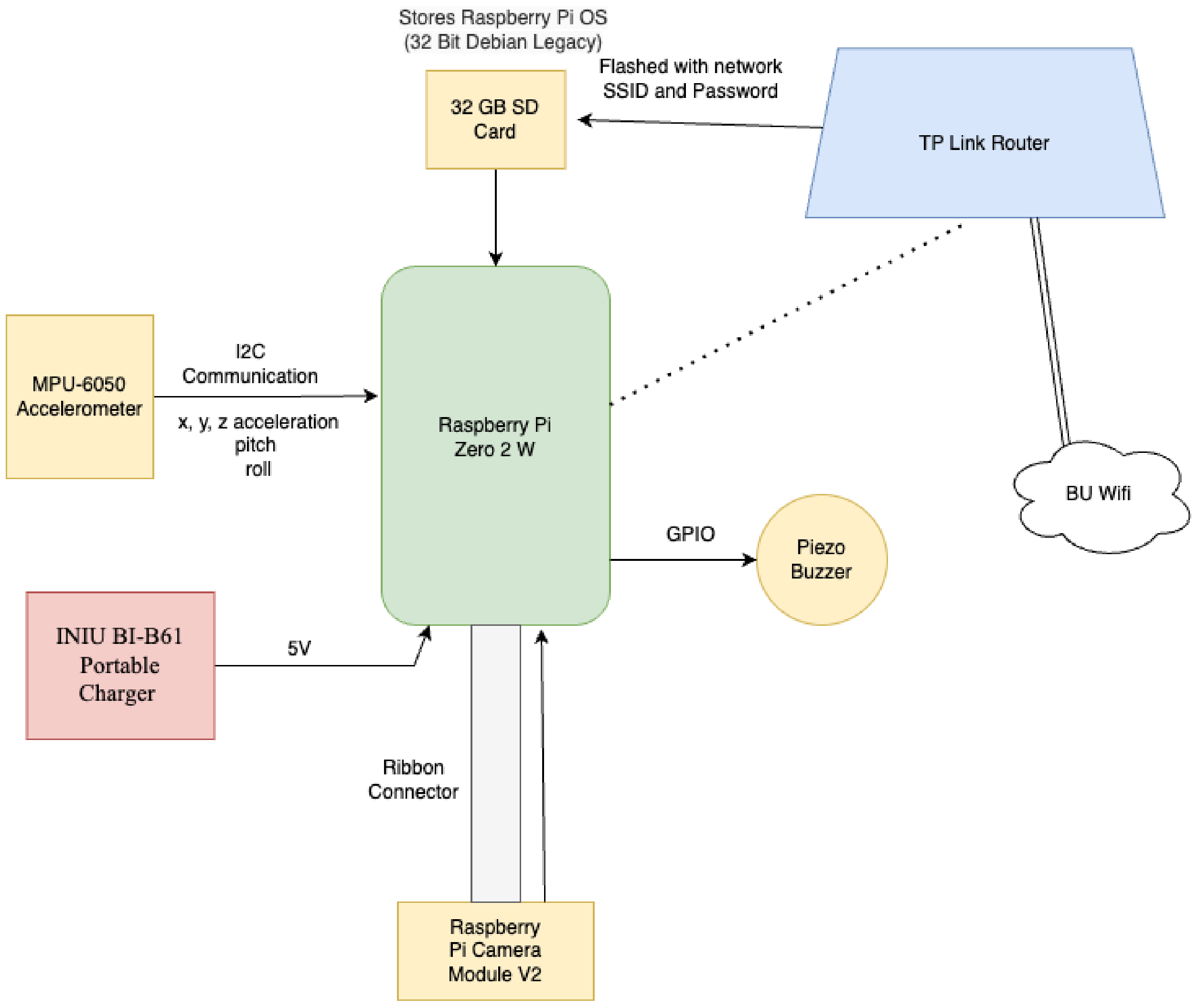
7.3 Appendix 3 – Software Block Diagram


Start
Actively Scans the environment
Process the sensor inputs in the backend
Does it seem like someone malicious is acting on your bike?
No
Yes
Buzzer goes off notify the owner
User reviews the camera does he stop the system?
Yes
No
Keep the buzzer going and start the gps

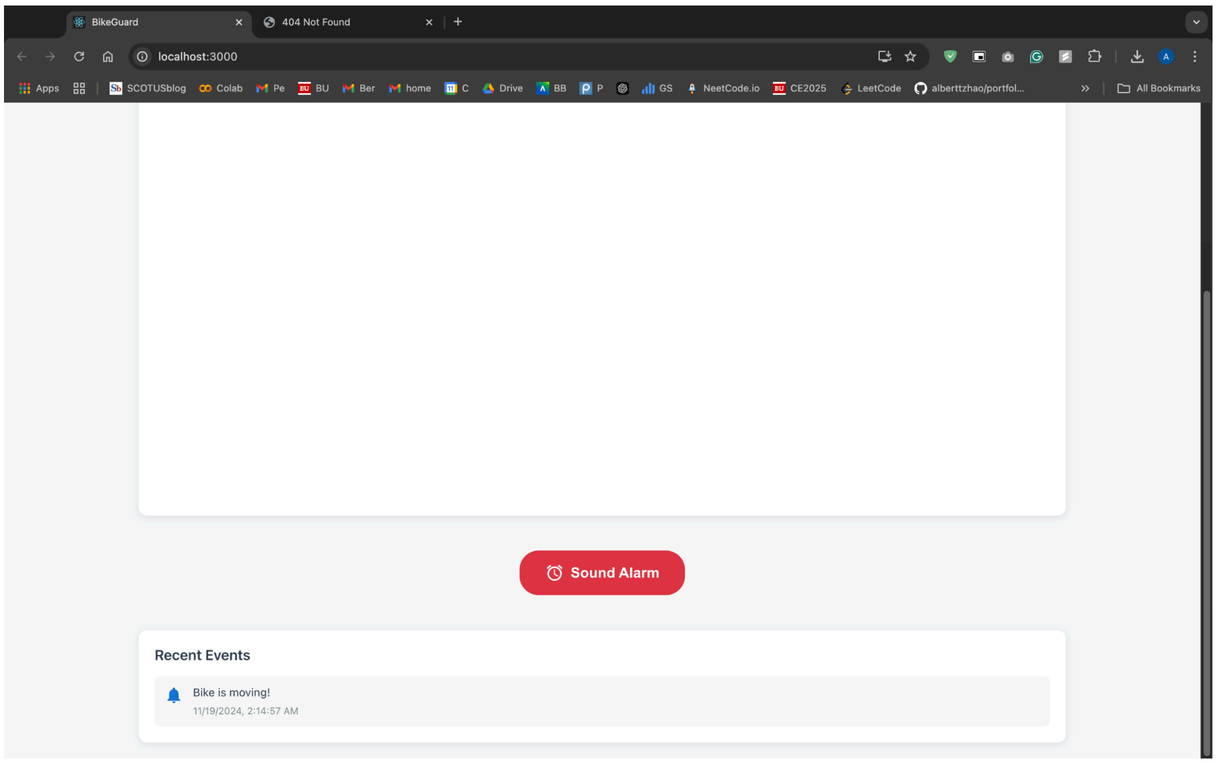
7.4 Appendix 4 – Onboard Software Block Diagram


log_mpu_data.js
child proccess
accelerometer.py
x, y, and z acceleration pitch, roll
Raspberry Pi
pitch and roll too high! set of buzzer
x, y, and z acceleration pitch, roll
pitch and roll to high! send post notification to back end server
x, y, and z acceleration pitch, roll
CSV file

7.5 Appendix 5 - Hardware block diagram


Stores Raspberry Pi OS (32 Bit Debian Legacy)
32 GB SD Card
Flashed with network SSID and Password
TP Link Router
MPU-6050 Accelerometer
I2C Communication
x, y, z acceleration pitch roll
Raspberry Pi Zero 2 W
BU Wifi
GPIO
Piezo Buzzer
INIU BI-B61 Portable Charger
5V
Ribbon Connector
Raspberry Pi Camera Module V2

7.6 Appendix 6 - Sample Web UI



Recent Events

Bike is moving!
11/19/2024, 2:14:57 AM

7.7 Appendix 7 - Required Materials

Hardware	**Software**
Onboard: ● Raspberry Pi Zero 2 W (with 32GB SanDisk SDHC Class 10 card) ● Piezo Buzzer ● MPU-6050 (Accelerometer) ● INIU BI-B61 Portable Charger 22.5W 10000mAh Power Bank ● Raspberry Pi Camera Module V2 ● TP-Link Router ● Small breadboard ● Electrical Tape ● Jumper Cable ● USB-C to micro USB cable ● Small heat sinks Remote: ● Laptop	Raspberry Pi: ● Raspberry Pi OS 32 Bit (Legacy), Debian Bullseye -> better for camera ● node.js -> child_process and fs ● Python3 -> accelerometer reading ● Flask -> pi camera stram Front End: ● Javascript and React ● CSS for formatting ● Flask for connecting to the backend Back End: ● Flask ● SQL for data storage Machine Learning Model: ● Logistic regression model ● Detect theft based on the accelerometer data that Assign labels "0" for normal and "1" for theft ● Extract_features() method ● Train_test_split() method ● StandardScaler() for better performance ● Train model with LogisticRegression. ● True Positives (TP): 49, False Positives (FP): 3, False Negatives (FN): 6, True Negatives (TN): 18 -> Accuracy = ≈0.8816

7.8 Appendix 8 - Hardware Pinout

Raspberry Pi pins	Usage/Description
GPIO 2 Serial Data(I2C) Pin# : 3	SDA pin used for I2C communication between accelerometer (MPU) and raspberry pi
GPIO 3 Serial Data(I2C) Pin# : 5	SCL pin used for I2C communication between accelerometer (MPU) and raspberry pi
Power Pin (3V3) Pin# :1	Powers the accelerometer
Ground Pin Pin# :9	Ground for accelerometer
GPIO 17 Pin# :11	Powers the buzzer, connects it to Raspberry Pi for GPIO usage
Ground Pin Pin# :6	Ground for buzzer
Ribbon Connector Pin# :N/A	Connects Camera to Raspberry Pi

7.9 Appendix 9 - Technical References

World Bicycle Day - European Parliament, www.europarl.europa.eu/RegData/etudes/ATAG/2022/729463/EPRS_ATA(2022)729463_EN.pdf. Accessed 7 Dec. 2024.

People Are Riding More than Ever—and This Is Good News!, www.bicycling.com/culture/a45325157/good-news-everyone-people-are-riding-more-than-ever/. Accessed 7 Dec. 2024.

Palma, Daniel. "The Best GPS Bike Trackers 2024: Why You Need One and How to Choose." *Cyclingweekly.Com*, Cycling Weekly, 13 May 2022, www.cyclingweekly.com/group-tests/find-your-stolen-bike-with-a-gps-tracker-165579.

Chub, Anastasiia. "E-Bike Theft - Prevention, Statistics and More!" *Whizz*, 15 Oct. 2024, getwhizz.com/blog/all-about-e-bikes/e-bike-theft/#:~:text=According%20to%20the%20FBI%2C%20more,are%20greater%20than%20a%20million.

7.10 Appendix 10 - Team Reference Sheet

Beren Donmez
bydonmez@bu.edu
Computer Engineering
Machine Learning Lead

Margherita Piana
mpiana@bu.edu
Computer Engineering
Hardware Co-Lead

Marissa Ruiz
mbruiz@bu.edu
Computer Engineering
Hardware Co-Lead

Bennett Taylor
betaylor@bu.edu
Computer Engineering
Software Co-Lead

Albert Zhao
albertz@bu.edu
Computer Engineering
Software Co-Lead