Boston University
Electrical & Computer Engineering
EC464 Senior Design Project

Final Test Report

BIKE GUARD

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: April 13, 2025

Equipment and Set Up

Equipment

Hardware	Software
Onboard: ● Raspberry Pi 4 (with 32GB SanDisk SDHC Class 10 card ● Piezo Buzzer 110 db ● IRLZ44N Mosfet ● MPU-6050 (Accelerometer) ● INIU BI-B61 Portable Charger 22.5W 10000mAh Power Bank ● Raspberry Pi Camera Module V3 NoIR Wide ● 12mm On/Off Metal Key Lock Switch ● TP-Link Router ● Small breadboard ● Electrical Tape ● Jumper Cable ● USB-C to micro USB cable ● USB C Extension Cable 1ft, 1-Pack, USB C 3.2 Male to USB C Female Cable ● 2-Pack 1.5ft Short Braided USB 3.0 A to A Cable - Male to Male ● Small heat sinks ● 3D Printed Enclosure + lid Remote: ● Laptop	Raspberry Pi: ● Raspberry Pi OS 64 Bit, Bookworm ● Python3 -> accelerometer reading, writing to CSV file, sending camera stream to flask server, sending post notification to backend, thread management, setting off buzzer ● Flask -> pi camera stream Front End: ● JavaScript and React for component-based UI ● CSS for responsive styling and theming ● WebSockets for real-time data updates ● Progressive Web App capabilities for offline functionality Back End: ● Python with Flask server ● SQLite database for local storage ● Firebase Firestore for cloud data persistence ● Firebase Authentication with email/password and Google OAuth ● Socket.IO for real-time communication 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 the model with LogisticRegression. ● True Positives (TP): 49, False Positives (FP): 3, False Negatives (FN): 6, True Negatives (TN): 18 -> Accuracy = ≈0.8816

Set Up

For testing, we mounted the device onto the main horizontal strut of the bike. This is the current most optimal position for the device. The components of the device are housed in a 3D-printed enclosure with clamps attached to the bike. Our hardware setup includes a key lock switch, a Raspberry Pi 4, a corresponding Raspberry Pi Camera module V3 NoIR Wide, an accelerometer, and a buzzer. We use a router to establish our network within the broader BU wifi. We connected the Raspberry Pi to the network and are hosting all our servers on the network. We are powering the Raspberry Pi with a portable power bank that is intended to charge phones and tablets. We additionally attached small heat sinks to our Raspberry Pi to prevent it from overheating. When the Raspberry Pi is powered on, a script called boot_try.py automatically runs. The boot_try.py script listens for GPIO signals from the key lock switch. When the switch is turned to ON, boot_try.py runs the accelerometer.py file. When the switch is turned to OFF, boot_try.py kills any existing accelerometer.py processes. This was implemented to save power. The accelerometer's data are saved in a CSV file using Python libraries. In this manner, the data can be used to train the machine learning model to recognize possible bike theft. Once the change in pitch and roll values recorded from the accelerometer reaches a certain threshold, the buzzer goes off. The buzzer will stop 5 seconds after the last theft event is detected. The user can also turn the buzzer on and off from the user interface. This video stream is embedded into the user interface. The backend file app.py receives constant information from the Raspberry Pi accelerometer.py folder. If the accelerometer detects motion greater than a pre-set threshold, it sends a push request to the backend and stores the message online in a Firebase database. Once the front end detects new changes in the database, it displays the most recent message on our website. As for the camera live view, the Raspberry Pi converts video from the Raspberry Pi camera into mpng (a different format of PNG) and streams it to its local host. Then, using Flask, we can embed the video feed directly to the front end.

Detailed Measurements Taken

Hardware and Connectivity

- Verified Power LED embedded into Raspberry Pi turns on when connected to power

- Verified green LED turns embedded into accelerometer turns on when connected to power.
- Pinged Raspberry Pi after boot to ensure network connection to router was successful.
- Confirmed successful functionality of the key switch system through the system by checking status of the process that runs boot_try.py using the command “sudo systemctl status my-script.service” in the Raspberry Pi terminal. When the switch is ON, we see a line in the output of the status check that says, "Switch ON – starting accelerometer.py..." When the switch is OFF we see either “Switch OFF – stopping accelerometer.py..." or "No process to stop" in the output of the status check.
- Further confirmed functionality of the switch by turning it to OFF, shaking the bike, and confirming that the buzzer does not go off, then turning the switch to ON, shaking the bike, and confirming the buzzer goes off. Also confirm that the video stream pauses when the switch is turned to OFF and resumes when the switch is turned to ON
- Confirmed functionality of the accelerometer by confirming 1) that the buzzer goes off once high shaking levels are achieved, 2) the buzzer stops when the excessive shaking ends, and 3) that accelerometer pitch and roll values are printing valid values (i.e. not 0)
- Checked that the connection from the Raspberry Pi to the backend is functional by pressing the start and stop buttons for the buzzer on the interface.
- Confirmed functionality and connection of the camera by 1) confirming live video was streamed to the front end and 2) checking that the camera is streaming at a wide frame angle with good resolution.

Data Recording and Software

- Confirmed that boot_try.py runs at boot by running the command: “ps aux | grep python” in the Raspberry Pi terminal after boot (shows all running python processes, we check for boot_try.py)
- Confirmed that boot_try.py runs accelerometer.py when the switch is switched to ON by running the command “ps aux | grep python” and looking for accelerometer.py
- Tested accelerometer data logging by running accelerometer.py and confirmed that motion is recorded in CSV.
- Verified the bike triggers the buzzer when the threshold is reached and sends a message to the front-end

- Observed that the user interface updates in real-time when the accelerometer detects excessive shaking, confirming software notification functionality
- Observed real-time camera stream on website, confirming camera software functionality
- Observed the functionality of the live GPS tracking, confirming the software GPS functionality
- Verified the correct functionality of the buttons to alternate the buzzer, confirming buzzer software functionality.
- Verified correct functionality of Google authentication and sign in, as well as the change of password option.
- Verified user profile creations by checking Firebase
- Verified log-ins by checking Firebase

Machine Learning and Classification

- Trained the logistic regression model on pre-collected accelerometer data
- Assessed model performance using the confusion matrix:
  - True Positives: 49
  - False Positives: 3
  - False Negatives: 6
  - True Negatives: 18
- Calculated accuracy as approximately 88.16%

Housing

- Verified the correct dimension of housing to both house all our components as well as avoid obstructing the normal functionality of the bike.
- Tested the correct positioning of the bike strut that allows for optimal video stream angle and non-obstructive behavior.
- Ensure clamps secure the device onto the bike and can withstand reasonable riding conditions
- Ensured reasonable sturdiness (i.e., you can't break it with your hands)

Conclusions

The housing functionality was successful and in agreement with our original goals. The enclosure does not interfere with the bike's operations, and it is easy to set up on any strut of the bike utilizing adjustable clamps. The hardware functionality was also successful. Once the key lock is turned to ON, the accelerometer successfully starts collecting data, and the buzzer is operational. In the event of excessive shaking, the accelerometer threshold is successfully breached, and the buzzer goes off. The buzzer successfully stops either after the excessive shaking event ends or when the user pushes the off button on the user interface. The user is also able to turn off and on from the user interface, fulfilling one of the earliest goals for this project. When the shaking threshold is reached, the back end correctly gets notified, and the notification is printed to the front end. The camera functionality was also successful. From the user interface, the user is able to access the Raspberry Pi Camera live video feed without interruption. The GPS live tracking works as expected as well, providing a good tracking system to the users. To enhance the security of our product, we ask the user to log into the web interface either by using Google identification or by creating a new account. Once the account is made, the user can change their password anytime. User verification ended up being successful as well. The machine learning model achieves good performance, effectively identifying theft scenarios with high accuracy. In conclusion, the system meets the measurable criteria for a successful run: real-time alerts, immediate buzzer response, and reliable classification of motion.

Extra credit: Next steps

- **Housing**: We plan to design and manufacture an improved housing that includes a transparent surface over the battery compartment, allowing users to easily monitor the battery level. Additionally, we will add a clear protective cover over the camera opening for enhanced durability.
- **Hardware**: Now that we have successfully implemented the networking component with our Raspberry Pi, we plan to integrate it into our system while ensuring that both the front end and back end continue to function smoothly. Once networking is fully integrated, we will move forward with adding Bluetooth pairing capabilities to both the device and the front-end interface.
- **Software**: We plan to transition our current website into a mobile app to make the user experience easier, including the ability to receive notifications instantly.

- **Machine learning**: Incorporate the trained logistic regression model directly into the Raspberry Pi system, enabling real-time detection and classification of theft scenarios without external dependencies. We would also like to collect more data using our new housing enclosure to inform the machine learning model.