BU Memo

Senior Design
ENG EC 464

To: Professor Pisano

From: Beren Donmez, Margherita Piana, Marissa Ruiz, Bennett Taylor, Albert Zhao,

Team: 8

Date: 03/06/25

Subject: Bike Guard Second Prototype Test Report

1.0 Equipment and Setup

- **Hardware**
  - Raspberry Pi Zero 3B+ with 32GB SanDisk SDHC Class 10 card
  - 110dB Piezo buzzer
  - 5V to 12V step up converter with USB input
  - 12V Mosfet
  - MPU-6050 Accelerometer
  - INIU BI-B61 Portable Charger (22.5W, 10000mAh)
  - Raspberry Pi Camera Module V3 NoIR Wide
  - TP-Link Router
  - Small breadboard
  - Electrical Tape
  - Jumper Cables
  - USB-C to Micro USB Cable
  - Small heat sinks
  - USB C 3.2 Male to USB C Female Cable
- **Remote Equipment**
  - Laptop
- **Software**
  - Raspberry Pi OS 64-bit (Bookworm)
  - Python – accelerometer data reading and file system operations
  - Flask – to stream camera feed and handle back-end tasks
  - Front-end: JavaScript, React, and CSS
  - Back-end: Flask, SQL for data storage

- Machine Learning Model: Logistic Regression

- **Setup**
    - Hardware components placed in the printed housing mounted to the bike, Raspberry Pi Camera peaks out of the enclosure through the designed hole to capture video data
    - Raspberry Pi and all servers are connected to TP-Link Router
    - Raspberry Pi is powered by a portable power bank
    - Small heat sinks are attached to Raspberry Pi's onboard chip components
    - Raspberry Pi connected via SSH using ssh Team8-B@Raspberry_pi_IP
    - Accelerometer data collected and saved in CSV files using Python scripts (for later use by machine learning model, not currently incorporated)
    - Buzzer connected to a transistor and a step up connected Raspberry Pi, goes off when Pitch and Roll values reach hardcoded threshold that were picked using the machine learning model
    - Pi Camera data is taken using Python and streamed to local host using Python and Flask
    - Pi Camera stream and accelerometer data collection managed by Python threading library
    - When buzzer goes off, push request is sent to back-end and stored in an SQL database
    - Front-end monitors for changes in back-end, uses React to update changes on web interface (npm start)
    - Front-end embeds camera stream from Raspberry Pi's local host to the web interface (python script to connect front-end and back-end)

2.0 Measurements Taken

2.1 Hardware and Connectivity

- Verified that the Raspberry Pi boots up correctly with the portable power bank by checking that green LED on the Raspberry Pi turns on
- Pinged Raspberry Pi after boot to ensure network connection to router was successful
- COnfirmed accelerometer was powered by checking that green LED turns on

- Confirmed successful communication between peripherals (accelerometer, buzzer) and Raspberry Pi by confirming accurate data was written to CSV file as well as confirming camera stream was successful and sent to local host by running our accelerometer.py

2.2 Data Recording and Alerts

- Tested accelerometer data logging by running accelerometer.py and confirmed that motion is recorded in CSV
- Verified the bike triggers the buzzer (pitch > 5 or roll > 10) and sends a message to the front-end
- Observed that the website updates in real-time when the accelerometer detects the shaking
- Observed real-time camera stream on website

2.3 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%

3.0 Conclusions based on test data

- The hardware setup functions as expected: the enclosure does not interfere with the bike's operation, and all components operate reliably.
- The accelerometer successfully detects excessive shaking, triggers the buzzer, and logs data.
- The Raspberry Pi Camera shows a live video feed to the front-end without interruptions.
- The machine learning model achieves good performance, effectively identifying theft scenarios with high accuracy.
- The system meets the measurable criteria for a successful run: real-time alerts, immediate buzzer response, and reliable classification of motion.

4.0 Extra Credit: Our Ideas for the Future

- **Integrate Machine Learning Model to the Product:**
  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.
- **Incorporate SixFab modem for cellular connectivity:**
  We currently have the SixFab modem kit connected and ready to go for cellular connection. We even activate the prepaid SIM card from SixFab. The problem right now is that the Raspberry Pi is not detecting the modem as a USB device. It is actually not detecting any USB devices. This could signal an issue with our Raspberry Pi USB driver. Over the next month we plan on troubleshooting and fixing this problem
- **Print out new iteration of housing:**
  Our first iteration of housing has some issues. The camera hole is at a suboptimal location and it is too big. We plan on making these changes and printing a new version of our enclosure. We also plan to add some clamps to this iteration to clamp onto the bike.
- **Incorporate power saving solution:**
  During our demo Prof Hirsch gave us a really good recommendation for power saving. He recommended using a small microcontroller like a Raspberry Pi Pico to sit idle and wait for accelerometer data that will then turn on the whole system when the threshold is reached. This would save a ton of power and be helpful for individuals who don't want to charge the device that often or plan on leaving it running overnight.
- **Cloud storage and authentication:**
  Right now our only form of authentication for users is through their google account. We would like to make our own login authentication with its corresponding database stored on the cloud. This will make our device more accessible to those without google accounts.
- **Location Tracking:**
  Once our modem is implemented we want to be able to grab the location of the device and display it on the front end.