CS370 Term Project: Speeding Detection with Raspberry Pi and GPS

Anton Vulman (avulman@colostate.edu)

Ezra Hsieh (ezra.hsieh@colostate.edu)

Mitchell Barrett (mitch146@colostate.edu)

Introduction

The issue of speeding poses a significant concern, particularly among teenagers and young undergraduate students. Newly licensed individuals often exhibit a tendency to engage in reckless driving practices, surpassing designated speed limits.[1] Unfortunately, parents, guardians, and other responsible family members find themselves with limited tools to effectively monitor and address this potentially perilous behavior.

In response to this challenge, our term project endeavors to employ a Raspberry Pi as a solution. By integrating a GPS module and the Raspberry Pi, we aim to develop a system capable of tracking the precise geographical coordinates of a vehicle, calculating its velocity, and flagging instances where the speed exceeds predefined limits. The collected data will be presented through a user-friendly map GUI on a desktop, offering a visual representation of speeding violations. This initiative not only addresses the immediate safety concerns associated with speeding but also provides caregivers with a proactive means of monitoring and guiding young drivers on the road.

Problem and Technical Description

Scenario Illustration

To illustrate the primary problem our project successfully addresses, consider being a parent of a 16-year-old teenager who has recently obtained their driver's license. Concerned for their safety, you decide to deploy a Raspberry Pi mini-computer inside your teenager's car, along with a GPS sensor to track the vehicle's latitude and longitude. Employing a Python script, you calculate your child's velocity while driving and record incidents where the vehicle exceeded a speed of 75mph, for instance, on a 16 GB SanDisk USB drive. Armed with this data, you can have a constructive conversation with your teenager about their driving habits. Without this tool, you have no idea if your teenager is endangering lives on the road while driving by themselves.

Technical Challenges

The central challenge at hand is the imperative need to monitor and visually alert caregivers when a vehicle surpasses predetermined speed limits. Traditional laptops and standard computers, due to their bulkiness and unwieldy nature, present limitations in securely installing and removing them from a moving vehicle. Recognizing this constraint, we opted for the Raspberry Pi, leveraging its compact size and portability, making it well-suited for the task at hand. The unobtrusive design of the Raspberry Pi allows for convenient placement within a vehicle without compromising the spatial availability of its internal compartments. For this project, we specifically selected the Raspberry Pi 4 Model B with 8 GB of RAM, ensuring optimal functionality and meeting the requirements of our term project. It is noteworthy that other Raspberry Pi models could also suffice for this purpose, with minor adjustments in code to best fit hardware and software requirements.

In conjunction with the Raspberry Pi, a pivotal component is the sensor device responsible for collecting satellite data and transmitting it to the computer. While numerous GPS modules are available, each with reliability in collecting GPS coordinates, they vary in specifications and connection methods. The meticulous hardware selection process is integral to establishing a physical connection between the GPS module and one of the input ports of the Raspberry Pi. Our chosen sensor for monitoring global positioning activity is the VK-162 G-Mouse USB GPS Dongle Navigation Module, selected for its accurate latitude and longitude readings, quick and consistent updates, ease of connectivity with a USB 1.0 connection, extended access through a lengthy cable, and secure protective casing with a magnet for effortless positioning.

Next, a script must be written to manage and specify the data, format, rate, and form of transferring coordinate data from the GPS to the Raspberry Pi. The script for the Raspberry Pi must also calculate and record velocity based on the displacement of GPS coordinates over a unit of time. To run the script automatically upon the Pi receiving power, the configuration of a convenient auto start feature was developed to aid with the “headless” setup. To allow for writing to the drive, permissions to the flash drive were granted, coupled with a ‘.desktop’ file consisting of the terminal command to run ‘gps.py’. Collectively, this allowed for an automatic process that conveniently requires no human oversight, besides powering on the Pi.

Subsequently, the data containing coordinate and velocity information needs to be transferred to a desktop computer with a screen monitor. As previously mentioned, this is easily accomplished by simply unplugging the flash drive from the Pi, and plugging it into the desktop. The desktop runs a script that automatically pulls data from the text file on the flash drive. The specific road based on the coordinates and its corresponding legal speed limit also need to be identified and obtained. Finally, it needs to produce a web-based application displaying a map GUI, demonstrating information including the location of each speeding violation, the speed limit of the location, and its speed value.

Solution and Implementation Strategy

Hardware Selection and Preparation

To address the imperative of providing parents with peace of mind regarding their teenager's driving habits, our solution commenced with a careful selection of hardware. As previously

mentioned, the Raspberry Pi 4 Model B with 8 GB of RAM was chosen as the mini-computer for deployment in the car, powered via a USB-C connection directly from the car. With the Raspberry Pi OS at its disposal, the miniature computer is adept at managing a variety of Python scripts, facilitated by an autostart script that initiates the execution of the script upon power activation.[2]

Initially, the NEO-6M GPS module was selected based on its functionality, inexpensive price tag, and recommendations from online guides, specifically by Arijit Das.[3] However, due to the absence of a built-in cable facilitating direct connection to the Raspberry Pi, and the necessity of a soldering operation for the connection, we opted for the VK-162 GPS module. This module features a strong magnetic case and a built-in USB connector, ensuring a safe, straightforward and hassle-free connection to the Raspberry Pi.

The VK-162 GPS Module is positioned with its face up to guarantee a robust GPS connection. A 16 GB SanDisk flash drive is connected to the Raspberry Pi within the car, enabling a Python script to record flagged incidents directly to this external storage. This setup facilitates easy data transfer from the Pi to a desktop for subsequent analysis. The flash drive is pivotal for data transfer, given the "headless" setup of the Pi in the car, lacking a monitor, keyboard, and mouse.

Two alternative options were considered but deemed less viable: the first involved removing the Pi and GPS module from the car to export data via another file transfer method, which was considered too cumbersome. The second option involved setting up a local server for data transfer when the car connects to the internet in the garage, but this was ruled out due to weak internet connectivity and the need for continuous server operation or preparation beforehand. The group unanimously concluded that a flash drive was the most practical and straightforward option.

Transferring, Recording, and Saving GPS Data

The Python script, initiated upon power start, documents latitude and longitude, calculates velocity based on the change in distance over time from the last GPS update, and determines if the calculated velocity exceeds a predetermined limit. In such instances, the script captures and prints incident details, including the incident count, date, time, longitude, latitude, and recorded velocity, into a text file named 'velocity_violations.txt' directly onto the external flash drive. Only data of clear incidents that violate the predetermined velocity limit will be recorded and documented. This is helpful since it both prevents the storage of too much unnecessary data noises, as well as retaining a degree of privacy of the drivers of the vehicle. By ensuring the device remains only a recorder of speeding violation and not a tracking device, a sense of trust between the parent and the child is maintained. This shared agreement balancing privacy and the supervision of inappropriate behavior can serve as an incentive for a child, whose well-being is prioritized in maintaining speeds below the agreed-upon predetermined limit.

The Python script for the transmission of data is partially inspired by Abdullah Jirjees' guide to VK-162 GPS Dongle and the codes available on his public GitHub repository.[4] We have to modify and greatly expand upon the original code, which was intended for connecting the GPS module to a desktop computer, not a Raspberry Pi. In addition, we have to develop codes to calculate the displacement and velocity from the coordinate data and store them in the

appropriate format. During the programming phase, three crucial libraries—pyserial, pynmea2, and geopy—were indispensable. Pyserial facilitates serial communication between devices, addressing the challenge of identifying the correct serial port for the GPS module. Due to minimal preliminary knowledge of ports, identifying the correct serial port characteristics for the GPS module was not a simple task. This was overcome by researching and employing the 'ls /dev/tty*' command in the Pi terminal window to visually detect the newly connected device. Another port used is pynmea2, which plays a vital role in parsing National Marine Electronics Association (NMEA) sentences received from the GPS module, ensuring accurate extraction of latitude and longitude data. Additionally, the geopy library is employed for calculating distances between consecutive GPS coordinates, contributing to the overall functionality of the Python script. The integration of these libraries in our script allows for the effective calculation of velocity, extraction of latitude and longitude data, and forwarding of incidents to a designated file on a flash drive. In doing so, the script provides a comprehensive and efficient solution to the challenges encountered during the project.

Visualization and Analysis Process

Following the removal of the flash drive from the Raspberry Pi, the subsequent step involves inserting it into a desktop computer. In order to provide an accessible visualization of our data, our team has developed a second Python script that generates a web page application referencing an HTML template, encompassing CSS and JavaScript components. This application automatically reads the violation incident count, date, time, longitude, latitude, and recorded velocity data from the flash drive. Subsequently, it employs an embedded map API to produce a visually intuitive map, adorned with pin-like markers at the locations of each speeding violation.

The web application, crafted with the 'flask' library, leverages the Flask microframework to efficiently create and develop web pages using Python. Additional essential libraries and modules employed by our web page script include 'os,' facilitating various utilities for handling file path manipulation, and 'datetime,' containing methods for more readable formatting of date and time data.

Finally, the request library is used to send HTTP requests to communicate with the Bing Maps API, which features the “Snap to Road” function that can identify the road based on its coordinates, and return valuable information including its legal speed limits, given their appropriate parameters, including a free basic API key.[5]

Our custom "Speeding Violation Map" web application, built on the Flask framework, plays a crucial role in reformatting the data for enhanced visualization and utility. For example, the script converts the velocity unit from meters per second to miles per hour, ensuring a more comprehensible presentation for American users. The reformatted data is then sent to a specifically designed HTML template, which generates an interactive map window through the Google JavaScript API.[6] This API utilizes coordinate, velocity, speed limit, date, and time data as parameters for its markers, providing a comprehensive display of each speeding violation.

The resulting web application not only visually showcases each incident on a user-friendly map but also offers additional insightful features. It displays the highest recorded speed and provides a total count of speeding incidents (defined by any incident of velocity that is greater than

predefined the limit set in the code), offering users a holistic overview of the driving behavior being monitored. Through the incorporation of interactive elements, such as clickable markers with infoboxes, users can access specific details such as date, time, speed limit, and speed for each recorded violation, enriching their understanding and facilitating informed decision-making. All in all, our method came together very well in every category. From the hardware/software, to the scripts and short command line arguments, everything operates as desired in an effort to satisfy our goal of creating a device to discourage dangerous driving habits.

Conclusion

The pivotal role played by the Raspberry Pi forms the core of our project, serving as the linchpin in achieving our primary objectives. Its ability to record and store GPS data in environments unsuitable for most standard computers underscores its adaptability and efficiency.

Looking ahead, with additional time or opportunities, we envision refining the data export process. An enhancement proposal involves scripting an automated connection to users' home Wi-Fi, facilitating the seamless upload of data files to a server. This advancement not only streamlines the process but also eliminates the need for a USB flash drive, enhancing both security and predictability.

Furthermore, the script governing the Raspberry Pi's data recording capabilities can undergo significant improvements. While our current implementation records incidents of velocity surpassing a predefined value along with the legal speed limit of the identified road, potential enhancements include utilizing the legal speed limit of each road itself as the metric value to classify whether or not there is a speeding incident. Though implementing such changes may necessitate substantial adjustments to our code, it promises a notable boost in usability.

Exploring additional avenues for improvement, our project could extend its functionality to monitor various risky driving behaviors beyond speeding violations. This expanded scope might encompass features such as detecting abrupt braking, illegal U-turns, reckless acceleration, and more. This diversification would not only fortify the system against a broader range of potential safety concerns but also contribute to a more comprehensive and robust solution. Due to the scalability of this project, there is potential to have great real world impact as well. Already there are analogous systems that incorporate real world driving data to impact the cost of insurance, rewarding safe driving by lowering rates.

Aside from the Raspberry Pi, our web map also serves the crucial role in visually displaying the speeding violation while representing the final stage of the practical utilization of our project. The map ultimately succeeds in presenting all the relevant information to users: location, legal speed limit, and the speed of the velocity violation. While the final result is satisfactory, improvement can be made to further stylize the web page, offering a more modernized appearance and layouts. Further functions can be implemented with the web page to support a true GUI, including the ability to refresh and update data, or a method to remove data with a single click.

In conclusion, our attempt has proven successful in harnessing the potential of the Raspberry Pi, its sensor, and a desktop computer to establish the intended comprehensive system. This system

effectively records and visually displays instances of speeding violations, within the parameters set by the users. While we have achieved our initial goals, we recognize that room for further enhancement and fine-tuning exists, contingent on the availability of additional time, resources, and opportunities. This acknowledgment underscores our commitment to refinement and potential advancements in the functionality and capabilities of our devised solution.

Bibliography

1. Governors Highway Safety Association. (2021). *Teens and Speeding: Breaking the Cycle.* https://www.ghsa.org/resources/Teens-and-Speeding-Report21

2. Raspberry Pi. *Raspberry Pi OS – Raspberry Pi*. https://www.raspberrypi.com/software/

3. Arijit Das. (2019). *Make a Realtime GPS Tracker device with Raspberry Pi.* https://sparklers-the-makers.github.io/blog/robotics/realtime-gps-tracker-with-raspberry-pi/

4. Abdullah Jirjees. (2023). *VK-162 G-Mouse USB GPS Dongle Navigation.* https://github.com/AbdullahJirjees/VK-16_GP

5. Microsoft. (2022). *Snap Points to Roads*. https://learn.microsoft.com/en-us/bingmaps/rest-services/routes/snap-points-to-roads

6. Google. *Maps JavaScript API.* https://developers.google.com/maps/documentation/javascript/overviewS