BLE Inspector

Gregory Gertsen

University of Washington-Tacoma
1900 Commerce St, Tacoma, WA 98402
gertsg@uw.edu

***Abstract*— The BLE inspector project was motivated by the desire to develop a creative technological solution for monitoring the status of a parked vehicle in a limited duration free-parking zone. The solution to this problem could allow a vehicle to park for the maximum possible duration while eliminating the risk of ticketing. The approach to the problem was to utilize a Raspberry Pi (RPI) computer powered by a cell-phone power bank in coordination with an open source IoT platform called ThingsBoard for monitoring, transmission, and display of relevant Bluetooth signals within the proximity of the vehicle. All code was written in Python and Pybluez was the library used for all Bluetooth functionality. The result of this project is a functioning prototype that acts as a proof of concept.**

***Keywords*— BLE, Raspberry Pi, IoT, Cloud**

I. Introduction

The BLE Inspector project is a further development from the initial prototype that was created during the first annual UWT Raspberry Pi Hackathon on June 7, 2019. The challenge of the hackathon was to create useful applications built upon the RPI platform. The idea for the project stemmed from the demand on students to continually move their vehicles to avoid ticketing versus the longer duration of their time on campus. The inconvenience of reparking and the costly solution of paying for parking motivated me to explore alternative solutions that utilize technology. Researchers have previously used BLE to solve other parking related issues. For example, in 2019 researchers in Slovakia proposed using BLE to monitor the availability of public parking spaces to make finding parking easier [2]. To better understand the technological solution to our problem, it is important to understand how the ticketing process works. Whenever a parking enforcer passes by a vehicle on foot, they simply make a chalk mark on the tire to signify that a countdown has begun on the vehicle. The countdown is simply how much time a vehicle can remain parked in the free parking zone, e.g. three hours, without being in violation. After the free parking period has elapsed, the parking enforcer returns to the area to look for any parked vehicles with chalked tires. If a vehicle is found with a chalked tire, then the vehicle is in violation and a ticket is issued. This process implies that a parking enforcer must always pass by a vehicle twice in order to issue a ticket. In the context of my solution, this means that the device only needs to notify us of the first time at which the enforcer passes by our vehicle to chalk it. Knowing this point in time allows us to have a perfectly accurate time frame of how long we have to park without the risk of a ticket. The inconsistency of parking patrols makes this knowledge especially valuable because without it we cannot know when the countdown has begun.

II. Methodology or Approach

The first and most crucial step of the project was to identify and capture any public signals that were being emitted by the parking enforcement. In order to do this, whenever I observed a parking enforcer I would simply approach them close enough to be able to see if any of their devices were discoverable. As luck would have it, there was always a discoverable device on the parking enforcer and it always displayed the common prefix of ‘XXRAJ’. The device that was discoverable was the printer

that each parking enforcer carries around on their belt to print out parking tickets. From this point, it was clear to me that parking enforcement could be detected anytime they came within the distance required to chalk a vehicle. The BLE signal is detectable within approximately 30 feet [1].

After determining that the signal was distinct and could be captured, the next step was to utilize the Raspberry Pi's Bluetooth capabilities. The RPI 3 B+ comes standard with Bluetooth 4.2 Low Energy as well as Dual-band 802.11ac wireless LAN (2.4GHz and 5GHz). In addition to these hardware features the RPI can also run Python scripts. For detection, all that was needed was to write code that could continuously scan for the BLE device which contains the prefix of 'XXRAJ' in its name. When detected, we simply log the time-stamp as well as other relevant data for user notification.

I chose to write the code in Python because of my familiarity and the fact that the RPI comes installed with Python. The code makes use of a Bluetooth library called Pybluez to continuously scan the vicinity for any publicly discoverable devices. The results of the BLE scan return a list of tuples where the first tuple value is represented as a string of the form "xx:xx:xx:xx:xx", where each x is a hexadecimal character representing one octet of the 48-bit address. The second tuple value is the more user-friendly name of the device, e.g. 'Greg's iPhone'. For our purposes we are looking for the name containing 'XXRAJ'. The duration of the scan is customizable and I have chosen to scan for 20 seconds at a time on a loop. During each scan, the list of tuples is checked for a string match 'XXRAJ'. If a match is found, then the loop is broken and the following data is collected for transmission. The time, public IP address, latitude, longitude, and Bluetooth scan result.

After collecting the relevant data from the successful BLE scan, the next step was to process and transmit it. The local IP address is obtained by using Python's requests module. It simply returns your public address as seen by web pages by running a request to http://ip.42.pl/raw. With the public IP address we can obtain our geolocation using the Python library ipinfo. The rest of the data is already in a form that can be used. To display the data we send it to the IoT cloud platform called ThingsBoard using a python API and in JSON format. As an added measure, the data can be sent to any specified email address. To make the system work successfully in a parked vehicle, the RPI is left in the vehicle connected to a cell-phone power bank and the device must be connected to wifi. In order to run the program, the device can be connected to via SSH.

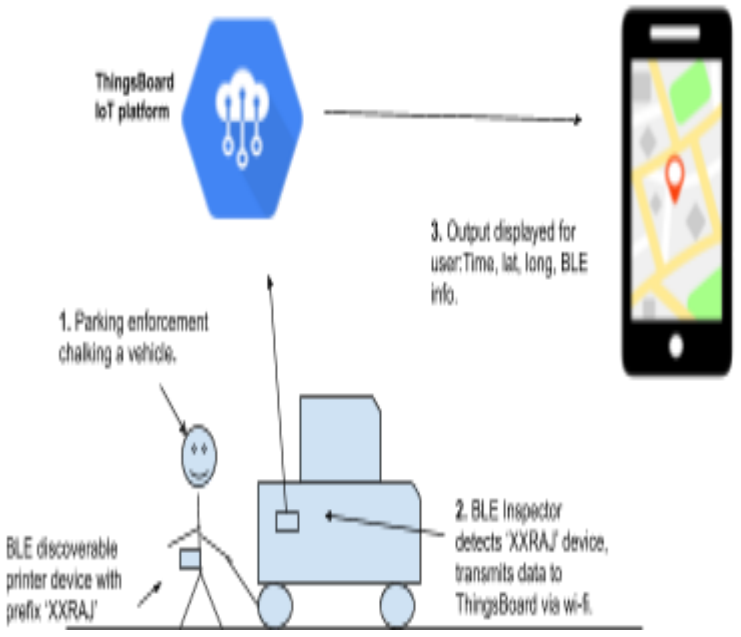

Fig. 1 Model of the BLE Inspector process.

A. Results and Discussion

The output can be viewed on the open source IoT platform called Thingsboard. ThingsBoard offers a number of data visualization options that take input from IoT telemetry data. For our purposes, the data visualization is of a street map with GPS location capability. ThingsBoard offers the administrator the option to create devices and dashboards and

then make these available to customers on a view only basis. I have created an administrator account where I have added the device that is our RPI as well as a dashboard map for viewing. Additionally, I have added a customer group that I refer to as MapCustomer. To MapCustomer, which more or less represents an organization, we add individual users of this organization by adding their email addresses to receive access links. When the user follows the link that is sent to them, they create an account and then can directly access the dashboard with the latest telemetry data sent from the RPI. The map displays a tagged location of the RPI/ vehicle. Clicking on the tag will bring up relevant information such as the latitude, longitude, BLE detected, and the time of detection. As previously mentioned, the information can be sent via email. In the current configuration, after the target BLE device is detected, the data is transmitted, displayed, and then the program is terminated, meaning no more BLE scanning is taking place. This is due to the fact that only an initial detection and transmission of data is needed to achieve the desired result.

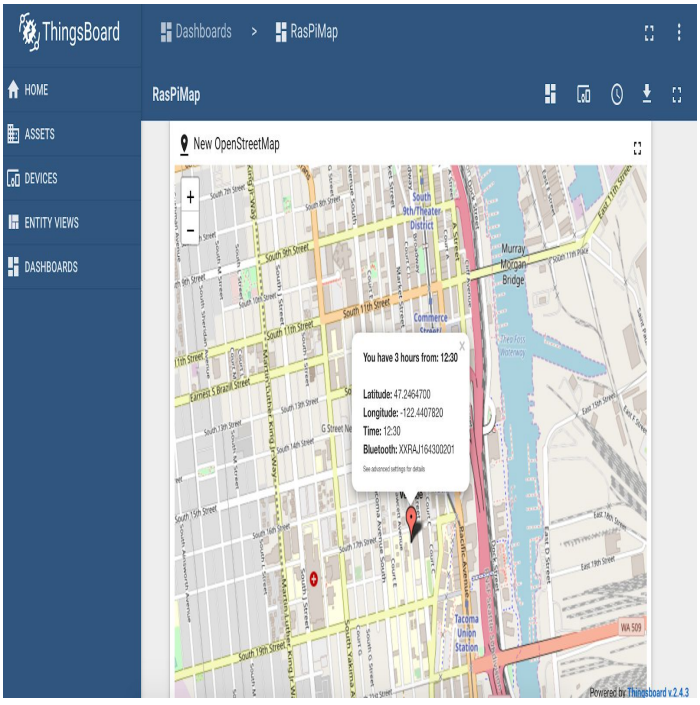

Fig. 2 User output as viewed on ThingsBoard.io

III. Conclusion

I have demonstrated proof of concept by creating a working prototype of the BLE inspector. While the technology is relatively straightforward, I believe there are a number of additional features that could be added to improve the user experience. Some of those improvements include: SMS messaging for convenience, additional information such as a live timer displayed on the map, the collection of historical data to track and even predict the movements of parking enforcement, analytics that could make parking location recommendations based on the historical data, and finally, the device could shut itself off after successful execution to save energy.

Fig. 3 The Raspberry Pi 3 B+ device and the Anker portable charger.

Acknowledgment

I would like to thank professor Eyhab Al-Masri, Ph.D. for his guidance.

References

[1] Muller, Nathan J., *Bluetooth Demystified*, 1st ed. New York, USA: McGraw-Hill Professional 2008.

[2] Karol Marso, Dominik Macko, "A New Parking-Space Detection System Using Prototyping Devices and Bluetooth Low Energy Communication", *International Journal of Engineering and Technology Innovation*, vol. 9, no. 2, pp. 108-118, 2019.