University of California, Berkeley
Mechanical Engineering Department

GPS Hardware for Vehicle to Grid Simulation
& MyGreenCar System
ME 196 Final Report

Aman Khan - SID 24767521
Dr. Karl Hedrick
Dr.Samveg Saxena & Alyssa Scheske

May 18, 2015

Contents

1 Abstract

MyGreenCar is an application in development in the Vehicle to Grid Simulation Lab, as a part of the Environmental Energies Technology Division at Lawrence Berkeley National Labs[1]. The application hopes to address a number of issues surrounding Electric and Plug in Hybrid Vehicles, including eliminating range anxiety and providing an accurate personalized fuel economy for all of its users. The application is the front end aspect of the Vehicle to Grid Simulation (V2G Sim) framework, and is currently in the Beta stage of development. The application logs GPS data from the users phone to determine realistic trip statistics. However, since the GPS data received is directly from Google Maps, it is difficult to verify its accuracy. Furthermore, for the application to be used successfully in car and bus fleets with multiple different drivers every day, a platform would have to be developed to utilize the data produced effectively.

2 Objective

The purpose of this project was to create a device to determine the accuracy of MyGreenCar by registering raw GPS data and comparing it to the data received from Google Maps on a cell phone. This would ultimately provide a sanity check to verify existing data produced by the Vehicle to Grid Simulation lab and provide a framework for its potential application in car and bus fleets. This report will cover the background of our reasoning behind selecting appropriate hardware and the methods used to construct a device that logs GPS data, which can then be passed to the MyGreenCar server, which produces a trip report. This report will also serve as a user manual for the hardware (GPS_ 1) in its current state as well as the technical difficulties I encountered while creating the device, and existing bugs. Current usage and future applications to validate data and implement in fleets will also be covered.

[1]V2G Sim Press Release - http://eetd.lbl.gov/news/article/57641/eetd-s-vehicle-to-grid-simulato

3 Procedure

3.1 Selecting the Device

To select the correct GPS we identified what aspects of the data we would be receiving were most important. Ultimately, our criteria consisted of:

- Location Accuracy : Ideally between 1-10 meters is what we were hoping to achieve. This would ensure realistic speed and location data and produce correspondingly verifiable results from V2G Sim.
- Altitude Data : Through the course of the semester altitude was a feature that was in development on the back end of the application to improve the V2G sim model. When it was clear that altitude would be playing a role in the accuracy of the trip result, we decided to incorporate as a part of our GPS criteria.
- Size : The device would have to be portable and should easily be able to fit in a car while remaining unobtrusive and unassuming.
- Robust : We should be able to recover data easily and modify the device if necessary.

After analyzing the criteria we identified as important, it became apparent that 2 options were available:

1. High End GPS : This would serve as the most accurate verification of MyGreenCars data by logging position data to within a meter of accuracy. The GPS we identified that would best fill this role was the ProMark 800. [2] It would be mounted on top of the car and the software it came with would allow us to easily compare results with phone data.
2. Low End GPS: Accurate enough to log data within a reasonable level of accuracy (5-10 meters), and far more portable than the higher end, car mounted GPS systems we identified earlier. It would also serve as the prototype for a platform that could be implemented in car and bus fleets. The hardware we selected for this purpose was the Adafruit GPS

[2] ProMark 800 Specsheet - http://www.igage.com/mp/PM800/022487-175_Spectra_ProMark800_1011_sec.pdf

Shield mounted an Arduino Uno[3]. Due to time and budget constraints, it was only feasible to develop one of these systems over the course of the semester. We selected the Low End GPS identified earlier.

3.2 Hardware

After receiving the components, I soldered the GPS Shield onto the board and produced the first iteration of an enclosure for the system in Autodesk Fusion 360[4]. We also purchased an Adafruit GPS antenna and SMA connector (See Figure 2 in Appendix for picture of antenna and connector. Antenna Datasheet : https://www.adafruit.com/products/960) to boost the signal of the GPS module slightly with the hope to produce better location data. A MicroSD card was purchased to log directly to an external memory source instead of on the device itself, as the device had both MicroSD card compatibility as well as Flash Memory capability. After writing Arduino code based off of samples provided on the adafruit website, the device successfully logged data on a serial monitor when set to Direct Connect mode. This data came in the format of NMEA[5] text which would then have to be parsed at a later stage into a usable format. We then verified that the device was operating within the desired level of accuracy (approximately 10 meters) after receiving a GPS connection. This indicated that the device was functioning correctly and was ready for Soft Serial connection, whereby it would log directly to the MicroSD card. This involved writing Arduino code to fulfill this purpose and testing to ensure the device could function accurately without being connected to a computer.

3.3 Parser Development

The GPSparser.py file can be located at http://tinyurl.com/m7pcdfg
A vital aspect of the platform is the ability to express the data coherently, both for the user as well as the server to process logically. In the format that the data is received, that is as an NMEA text file, the data cannot be easily interpreted.[6]

[3]https://learn.adafruit.com/adafruit-ultimate-gps-logger-shield/
[4]See Figure 1 in appendix
[5]NMEA Data properties can be seen here : http://www.gpsinformation.org/dale/nmea.htm
[6]An example of NMEA data is shown below in Figure 3 of the Appendix.

There are set rules for the format of NMEA data which had to be followed to parse and interpret the data correctly. After numerous attempts at finding a library to parse the data correctly, we settled on a Python library called pynmea2[7] , which we used as the framework of our parser. The script we created reads an nmea format .txt file (generated by the GPS) line by line and identifies the correct type of data to parse (RMC format). GPS_1 logs timestamps according to UTC (Coordinated Universal Time), however the program accounts for this by automatically adjusting the time zone to reflect California time. This is crucial to match up trips from GPS_1 with trips from the phone and compare the two correctly. The script then populates a .json file with the headings required by the server to produce a trip result. This json is then automatically posted to the MyGreenCar server using the python requests library[8]. The user then receives an email of the trip just as he normally would with a phone.

3.4 Difficulties Experienced

Selecting the correct hardware involved a great deal of deliberation due to the number of factors and options available. Ultimately we decided on a platform with the greatest robustness for prototyping the proof of concept design and determine the scope of MyGreenCar for fleets. This option would also provide a degree of verification of trips that have already been collected as well as future trips. Furthermore the price of more accurate modules came in the form of size constraints as well as budget constraints, which led to the decision to go with a portable, cost effective platform. Developing an effective parser was challenging, as many Python libraries were not easy to use or not well documented. After finally settling on pynmea2, it took some time to flesh out the library as well as extract the data we wanted while saving it in the correct json format. GPS_1 is far from perfect and has some bugs and areas of improvement that will be explored later in this report.

[7]https://github.com/Knio/pynmea2

[8]http://docs.python-requests.org/en/latest/

4 Theory

Speed is a crucial value that is required to generate a trip report and calculate State of Charge. To acquire speed data we had 2 options :

- Speed data sent from the GPS Satellite, which is received in the form of speed over ground in knots. The server utilizes speed in the form of meters per second. Due to this we had to convert speed using the formula:

$$Speed(knots) * 1.150779(\frac{mph}{knots}) * 0.44704(\frac{Meters/second}{mph})$$

- Haversine Formula that takes into consideration changes in position values to create a speed, using the formula:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

$$Speed = \frac{d}{\Delta t}$$

r=radius of earth in kilometers = 6378

$$\phi = Latitude$$

$$\lambda = Longitude$$

After testing we concluded that the speed provided by the satellite was more consistent with the results we were expecting, and are currently using that in the parser.

5 GPS_1 Manual : Recording a Trip

This section will cover usage of the device, running the parser and the type of results a user should expect. The device should be used outside of buildings, with the antenna magnetically mounted on the roof of a car. It can log trips until the battery dies (approximately 6-7 hours) or until the SD card runs out of space.

1. Ensure the microSD card and all the connections are correctly inserted and secure[9]. Check that the device is set to Soft Serial mode[10].

2. Before setting out on a trip, turn the device on by flipping the switch on the battery pack. At this point the red light labeled fix on the top of the device (the Adafruit shield) should be flashing at 1-second intervals. This indicates that the GPS is warming up and attempting to get a fix from the numerous GPS satellites orbiting Earth. GPS_1 can track up to approximately 22 satellites simultaneously. Once the fix light[11] stops blinking every second and begins to blink every 15 seconds, this means that the GPS is now logging data accurately as it has a strong enough connection with the satellites.

3. At this point the reset button should be clicked, which creates a new log file on the SD card. This is to ensure that the trip being logged starts fresh, as the device logs the entire time it is on, even if it does not have a satellite fix

4. The device is now actively logging data to a .txt file on the SD card. To verify this, ensure that the L13 light [12] is blinking regularly, which indicates that the device is writing to the SD card. Begin your trip and note the start time as well as any times the GPS loses a fix (fix light blinks every second instead of every 15 seconds).

5. When your trip is complete, click the reset button again. At this point your trip has been recorded.

[9] See figure 4 for device with all connections correctly inserted
[10] See figure 5 for location of Soft Serial switch
[11] See figure 6 for location of "Fix" light
[12] See figure 7 for location of "L13" light

6. Turn the device off remove the antenna from the car. Remove the microSD card and insert it into the adapter. Insert the adapter into an SD card slot on your computer and copy the second to last .txt file in chronological order from the SD card to the same directory as the parser script.

7. Call the parser titled GPSparser.py on the .txt file. The parser will automatically post the trip data to the server and generate a trip result email, as well as a file with the serialized json for reference. [13]

[13] An example trip result email is shown in figure 8.

6 Results and Improvements

The results we have generated so far indicate that the device is accurate enough to produce logical trip data. At this time we have not conducted enough trips to completely verify just how accurate GPS_1 is. At certain points during the trip, when the GPS loses signal, the speed can occasionally drop to zero. We would like to improve this in the future by incorporating the Haversine speed formula with the GPS speed to have a more accurate, non-zero speed value. The 3D printed enclosure we have developed so far holds the device but does not fully enclose it. It would be beneficial to create an enclosure that better suits the usage of the device and protects it better. The device also appears to rarely record data at an incorrect baud rate, resulting in data that is unusable. More testing is required to further improve on the reliability of the device.

7 Applications and Conclusion

GPS_1 is a prototype that functions as model for how a permanent GPS tracker could be implemented in car and bus fleets to log trip data accurately enough to generate a useful result. In the future, the framework and necessary components could be scaled down for cost and size effectiveness by printing a custom PCB.

8 Appendix

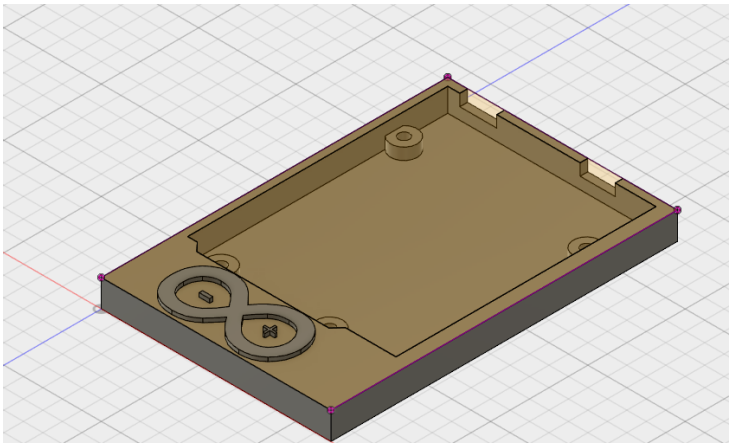


Figure 1: 3D model of the enclosure

Figure 2: GPS Antenna and SMA connector

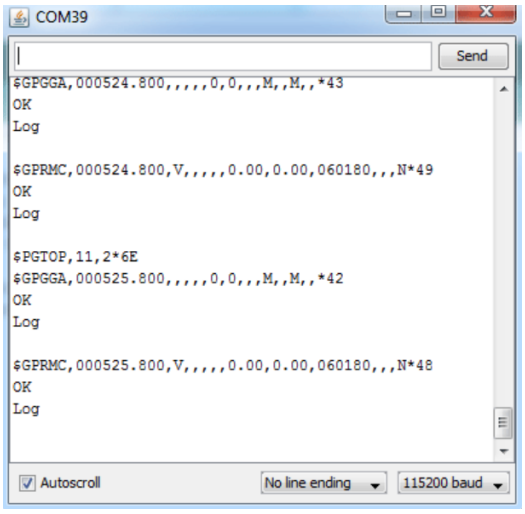


Figure 3: NMEA example data from serial monitor (taken from Adafruit site)

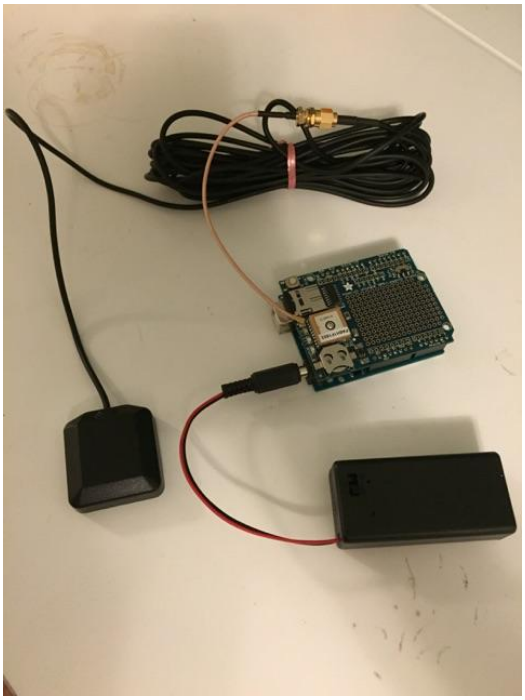


Figure 4: GPS_1 with all connections

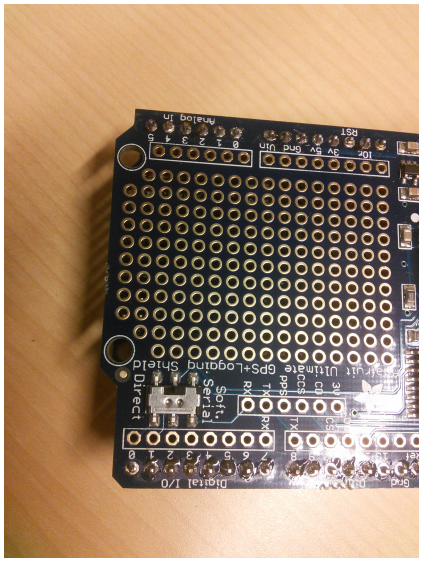


Figure 5: Soft Serial Switch

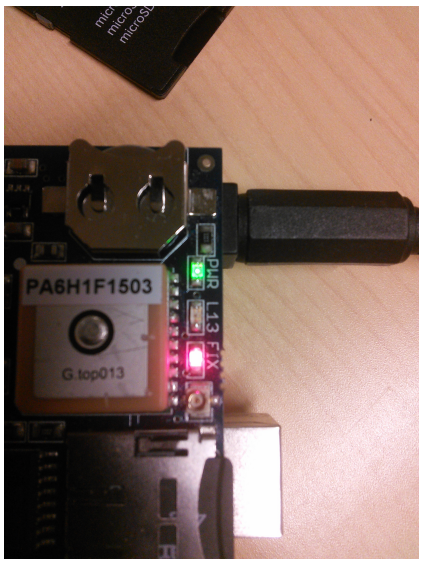


Figure 6: GPS Fix light

Figure 7: L13 light

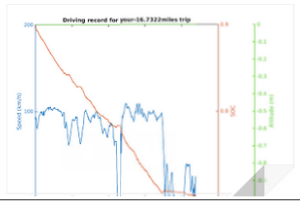
Hi GPS_1!

This is a summary report of your single driving route:

- Time duration: 1316.0 sec
- Distance: 16.7322 miles
- Initial SOC: 0.89987
- Final SOC: 0.70251

You'll find more data, generated by the MyGreenCar models, attached. More information about the process we use to generate this data is available at our site.

Thanks for using MyGreenCar!



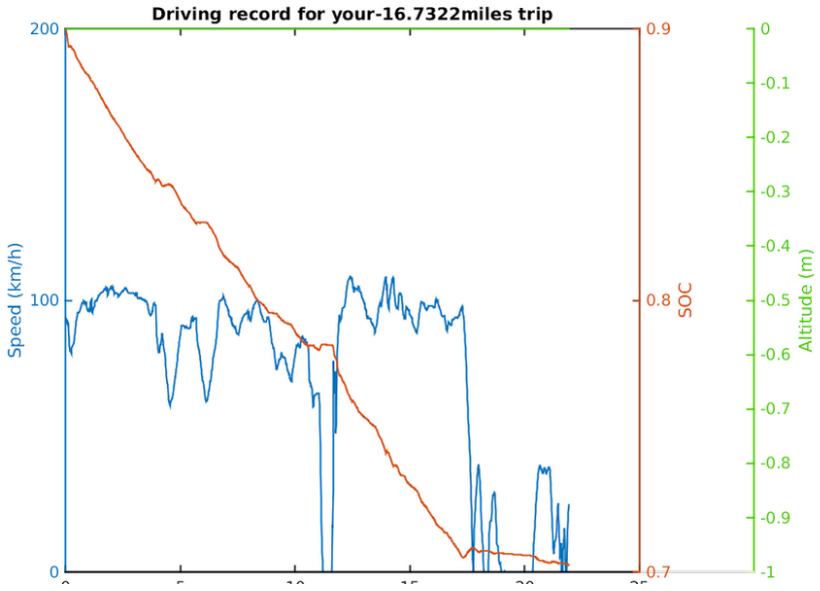


Figure 8: Example Trip Result Email