University of Strathclyde Glasgow

CS413 Embedded Systems – ACE1 Phase 1

Group 3:

Marc Ritchie

Robert Rendell

Mark Provan

Patrick Monaghan

Progressive Improvement in Motoring Proficiency for Economic Driving (PIMPED)

Contents

Table of Figures

1 Abstract

This document has been produced as part of Phase 1 of ACE1 for CS413 – Embedded Systems. It describes the idea and assess the capabilities and functionality of the device we have decided to produce. It identifies all of the components we require to build the device and the cost of each individual component. In addition, the document includes a detailed hardware and software design of the device. Finally, it outlines a plan of which team member(s) will take on each task and the time assigned to each task involved in building the hardware and software.

2 Idea

We have decided to create a device which we have named Progressive Improvement in Motoring Proficiency for Economic Driving (PIMPED). PIMPED will allow a user to connect to their cars on-board system and extract real-time data while they are driving. Most extracted data will be stored and the rest of the data - real-time data such as current miles per hour, current miles per gallon - will be displayed on a digital screen viewable to the user. A full description of the data which will be displayed to the user while driving is described in section 4.1. When the user completes their journey they will be able to remove the PIMPED device from their car and upload the data extracted from their journey onto a web server. Each user of PIMPED will be able to create their own unique user account where they can upload their journey data to and have access to all the journey data they have ever uploaded to the web server. This allows each user of PIMPED to track their driving over time. The main benefit of this is that a user will be able to assess with ease, whether they are driving more efficiently.

3 Main Components

3.1 ELM327

An ELM327 device will be an essential component of the PIMPED device. An ELM327 device supports all On-board Diagnostics II (OBDII) protocols, which are crucial for the PIMPED device to function. OBDII will be used to extract any data from the on-board system of the car that the PIMPED device is running on. An ELM327 device can connect to most cars produced after 1996 and can access data from the Engine Control Unit. Figure 1[4] shows the ELM327 device that will be used by the PIMPED device.

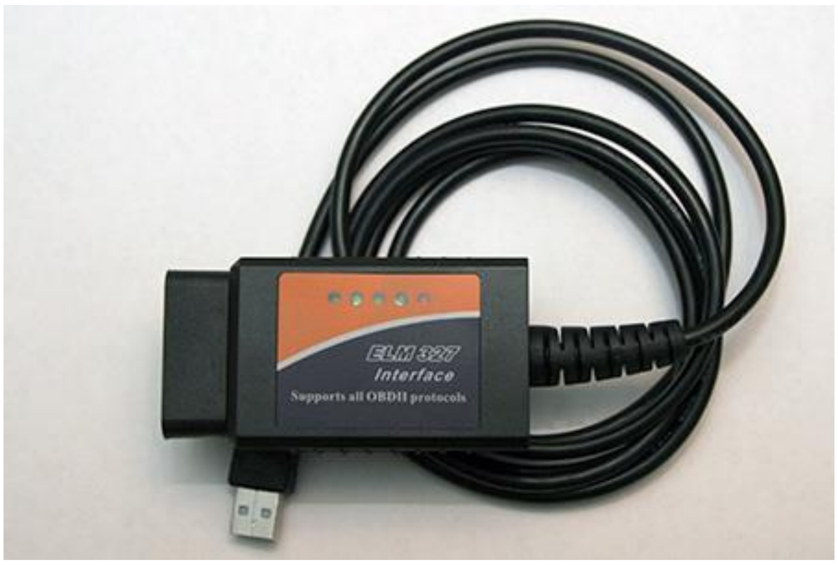


Figure 1: ELM327 - OBDII device

The full range of the data which is extractable by OBDII can be found on Wikipedia [5].

3.2 Raspberry Pi b+

A Raspberry Pi b+ will be a key component of the PIMPED device. A Raspberry Pi is a small, low-cost computer which has the standard capabilities of most desktop PC's. It can also interact with the outside world using chips and sensors which can be easily installed onto the Raspberry Pi's GPIO headers. More information on Raspberry Pi's and their capabilities can be found on the Raspberry Pi website [9]. Figure 2 shows a labelled image of a Raspberry Pi b+ with its main features identified.

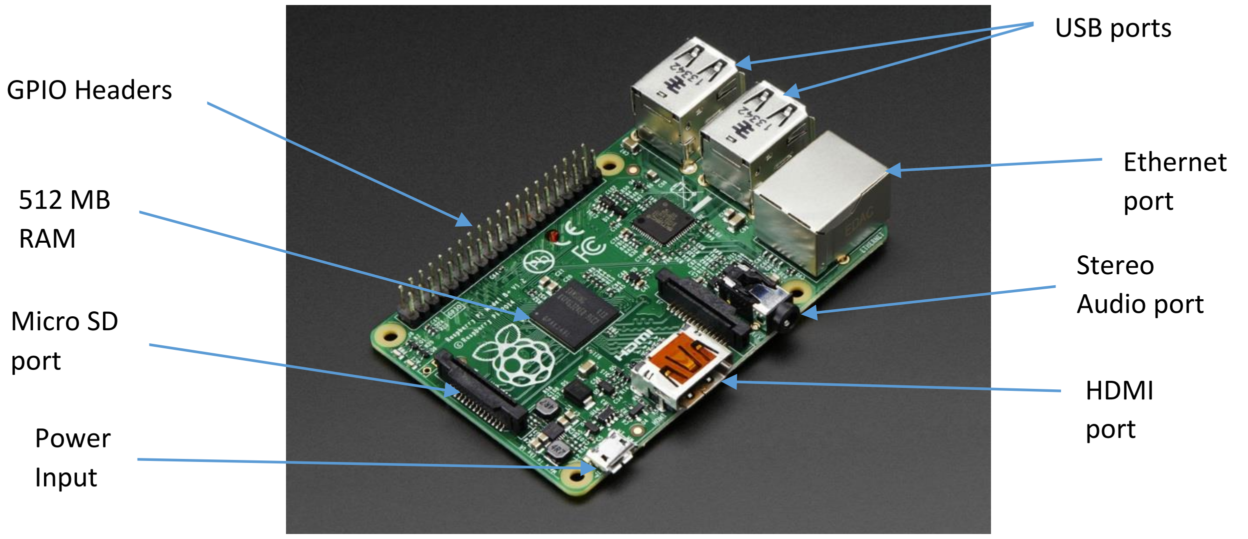


Figure 2: Raspberry Pi b+

Data will be passed back from the ELM327 device, using the OBDII protocols, to the Raspberry Pi. Some of this data will be displayed on a TFT LCD Display. All data retrieved by the Raspberry Pi will be stored on the Pi. The extracted data file will be transferrable from the Raspberry Pi via USB onto the user's computer once they have completed their journey. Once the extracted data file is on the user's computer they will be able to upload the file onto the PIMPED web server into their own unique account. Here the user will be able to view all previously uploaded data and the data from the journey they have just completed.

3.2.1 Raspberry Pi Digital Display Monitor

A 7-inch Raspberry Pi TFT LCD Digital Display will be used to display real-time data when the device is being used. The OBDII device returns data every one second, so the LCD display will be updated every one second.

3.3 GPS Receiver

PIMPED will use journey tracking. In order to allow for this, it is crucial that a GPS receiver is installed into the PIMPED device. An Adafruit Ultimate GPS Breakout chip[1] will be installed directly onto the Raspberry Pi's GPIO headers.

4 Core Functionality

The core functionality of the device includes:

1. Display of real-time information;
2. Record journeys in car with GPS;
3. Dashboard online service that allows user to upload data to web server;
4. Scoring economic driving;
5. Compare friends results;

4.1 Real-time information

We want to show real-time information as you are driving. Information should be shown on an LED display detailing:

1. Current miles per hour (MPH);
2. Current miles per gallon (MPG);
3. Miles driven in current drive;
4. GPS co-ordinates of the car;
5. Current journey time;
6. Engine load value;
7. Ambient Air temperature;
8. Throttle position;
9. Engine RPM;
10. Engine coolant temperature.

This information should be scrolled across the LED screen in sequence. This information should not be distracting.

4.2 GPS Record Journeys

We intend to have a GPS receiver component as part of our device that will be able to feed co-ordinates so that a map of the route taken can be drawn. The GPS receiver will start as soon as the device is connected to the car.

This will allow us to pinpoint certain events on a map. The colour of the route will gradually change depending on how economically you were driving. This is a similar idea to the route colouring technique used in the Nike Running app[7], displayed in Figure 3:

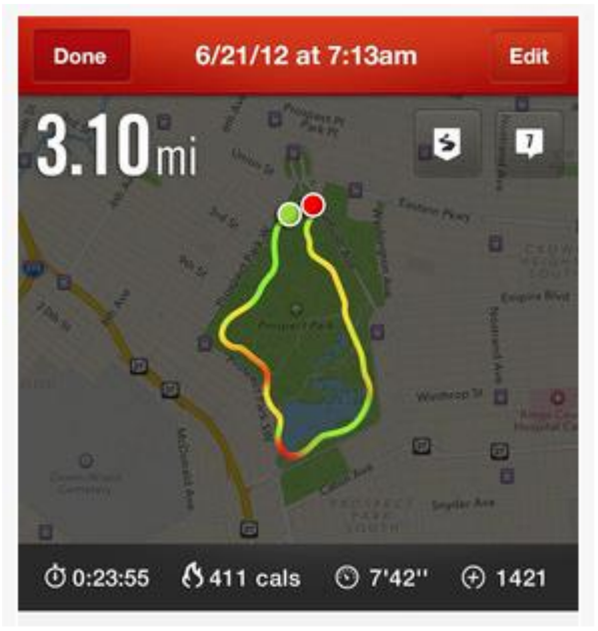


Figure 3: Example route colouring - Nike Running App

This would easily show where the speed was greatest or where you were driving most efficiently. We plan to use a digital display for the real-time information so the actual map of route taken would have to be displayed on our web interface as the digital display would not be able to handle this.

4.3 Web Application

The idea is that when journeys have been recorded, the device can be connected to the web application and upload all the information gathered. As a user you will be able to log into the web application and view the analysis of your journey or all of your journeys.

You will be able to view statistics for individual journeys and the route for each will be plotted onto a map. There will also be a screen of general statistics and averages of all your journeys.

4.4 Economic Driving

One of the main themes of the project is economic driving and we want to be able to tell the user how economic their driving is.

In our opinion there are two main factors in uneconomic driving. The first is in braking too harshly; this would be represented as a sudden decrease of speed from the data collected during a journey. The idea is that speed change should be gradual and planned. This is also the case for the second factor which is whether you are over-revving or under-revving. Over-revving and under-revving can be detected by checking the rev counter value is within the range of 800rpm-3300rpm.

We will use these factors to provide feedback to the driver on how they could drive more efficiently or tell them if they are already driving at optimum efficiency.

An Economic Driving Score will be calculated based on all the journeys by a single driver.

4.5 Compared Results

There will be a table on the Online Dashboard Service that allows you to compare your general statistics and your economic driving score to others who have used the device.

5 Software Design

In order to make sure our software is as modular and therefore maintainable as possible, we will split out our code into separate applications. The applications will be run by a series of scripts, with each script dealing with one specific job. In this section we will detail how our scripts will carry out each of their jobs and how they will communicate with one another. During its design, four primary parts of the software that need to be implemented in order for PIMPED to function were identified. These are detailed below.

5.1 Web Application

This application is responsible for receiving data from the device and displaying in a clear and attractive way to the end user of the PIMPED device. The application will require the user to login so that their driving data can be saved specifically to their account online. When the user logs in, the user will be prompted to connect their device to their computer via USB, where the file stored on the PIMPED device will be accessible. The file uploaded to the web application will contain a combination of the data about the car extracted via OBDII and the GPX file generated from the PIMPED devices GPS module, all of which will be stored on the Raspberry Pi. Figure 4 shows an example of what the user file upload screen will look like.

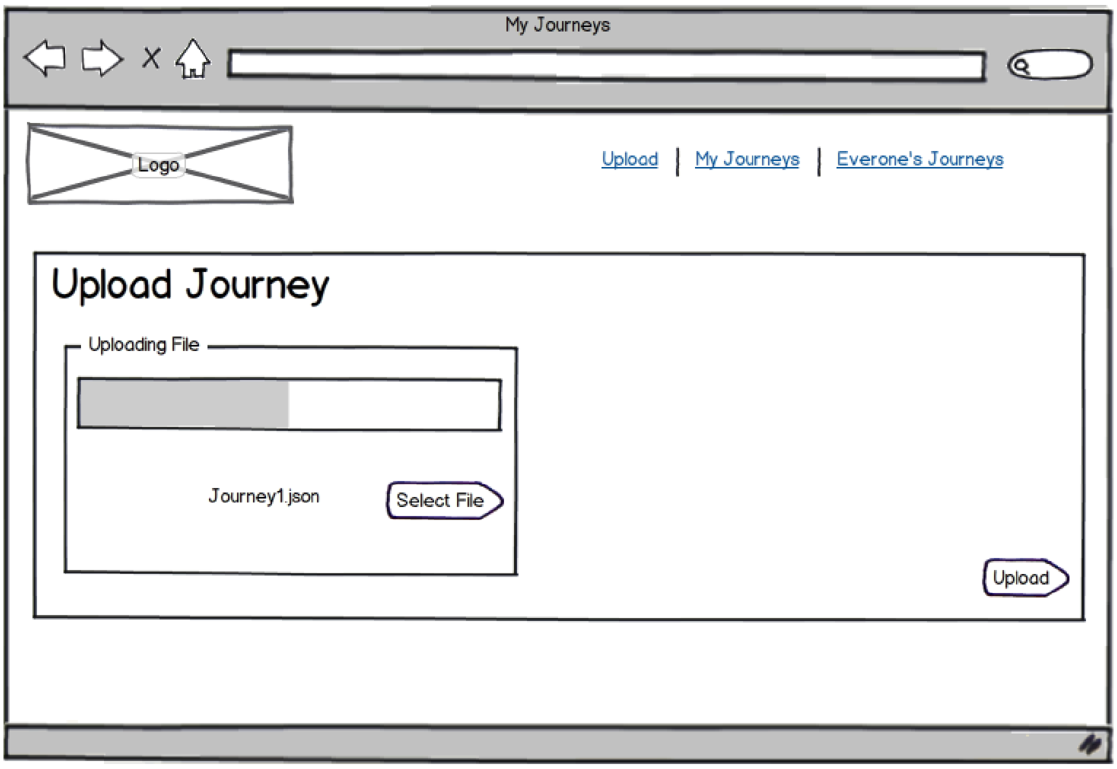
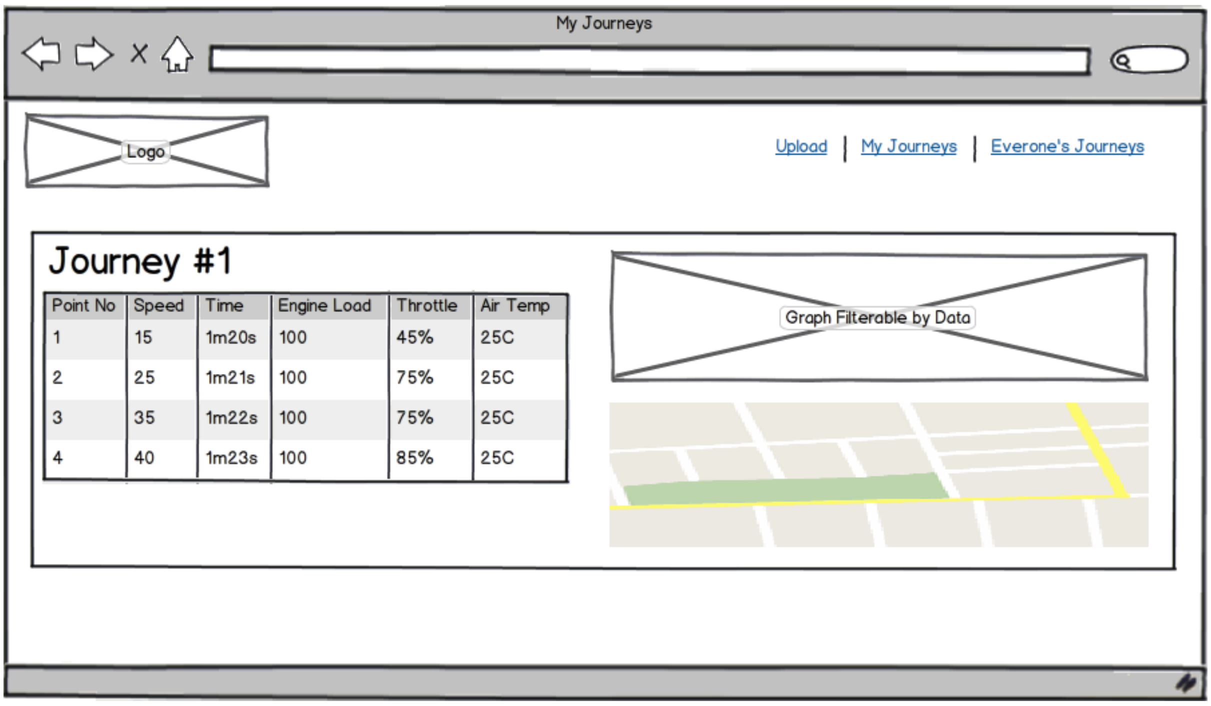


Figure 4: JSON file upload GUI

When uploaded, the data must then be stored. As all the processing of the data will be done on the Raspberry Pi and saved in JSON format, MongoDB[6] will be used to store it. MongoDB is a schema-less database and stores documents in a JSON format. This keeps data in the same format across our entire stack and makes it easy to query.

To handle the visualisation of data, we will need to make use of the Open Street Maps API v0.6[8] to display the GPS data for the journey. We will also use the ChartJS[2] JavaScript library to allow us to render basic charts in the browser, for visualising feedback and comparing the users driving to that of other users on the system. Figure 5 shows an example of how data will be displayed to the user after upload.



Point No	Speed	Time	Engine Load	Throttle	Air Temp
1	15	1m20s	100	45%	25C
2	25	1m21s	100	75%	25C
3	35	1m22s	100	75%	25C
4	40	1m23s	100	85%	25C

Figure 5: User data display GUI

The web application itself will be built using Ruby, HTML, JavaScript and CSS and will be hosted on a server running Ubuntu Server 14.04.1 LTS. DigitalOcean[3], a cloud hosting company, kindly offer free credit to students, so they will be used as the cloud service provider for the PIMPED web application.

5.2 GPS & OBD Extractor

As we are using a USB GPS unit on our device, a script will be required to extract this data and store it in a file that can be uploaded to the web later. Python is recommended as the language of choice for Raspberry Pi, so that is the language that will be used for this part of the implementation of PIMPED.

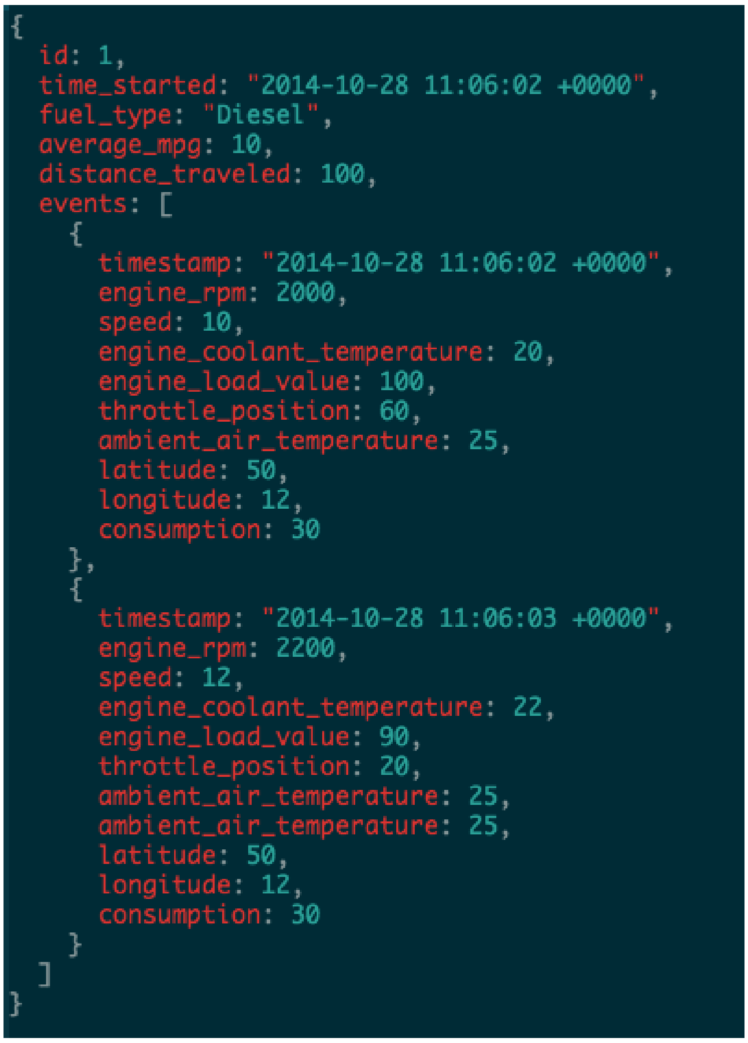
Having a script running constantly on the device would use excess power and this is undesirable. Every second, the Python script will run executing the following instructions:

- get the latitude and longitude from the GPS unit;
- store this data in a file with a timestamp;
- sleep until it wakes one second later and repeat

The same process will be followed for the OBDII unit, polling it every second to extract data from the car. Both of these data points are combined by the device and stored on the internal SD card of the Raspberry Pi.

5.3 File Format

We will use a JSON format for storing each journey and the events associated with it. This makes it easy to store in MongoDB and allows data to be stored for each journey in one document, rather than distributed across a relational database. Figure 6 shows the JSON file structure that will be used.



```
{
  id: 1,
  time_started: "2014-10-28 11:06:02 +0000",
  fuel_type: "Diesel",
  average_mpg: 10,
  distance_traveled: 100,
  events: [
    {
      timestamp: "2014-10-28 11:06:02 +0000",
      engine_rpm: 2000,
      speed: 10,
      engine_coolant_temperature: 20,
      engine_load_value: 100,
      throttle_position: 60,
      ambient_air_temperature: 25,
      latitude: 50,
      longitude: 12,
      consumption: 30
    },
    {
      timestamp: "2014-10-28 11:06:03 +0000",
      engine_rpm: 2200,
      speed: 12,
      engine_coolant_temperature: 22,
      engine_load_value: 90,
      throttle_position: 20,
      ambient_air_temperature: 25,
      ambient_air_temperature: 25,
      latitude: 50,
      longitude: 12,
      consumption: 30
    }
  ]
}
```

Figure 6: JSON file structure

5.4 Render Data to Screen

A 7-Inch LCD screen, detailed in section 3.2.1, will be used to display real time data. Figure 7 shows the GUI which will be implemented and displayed on the LCD screen.

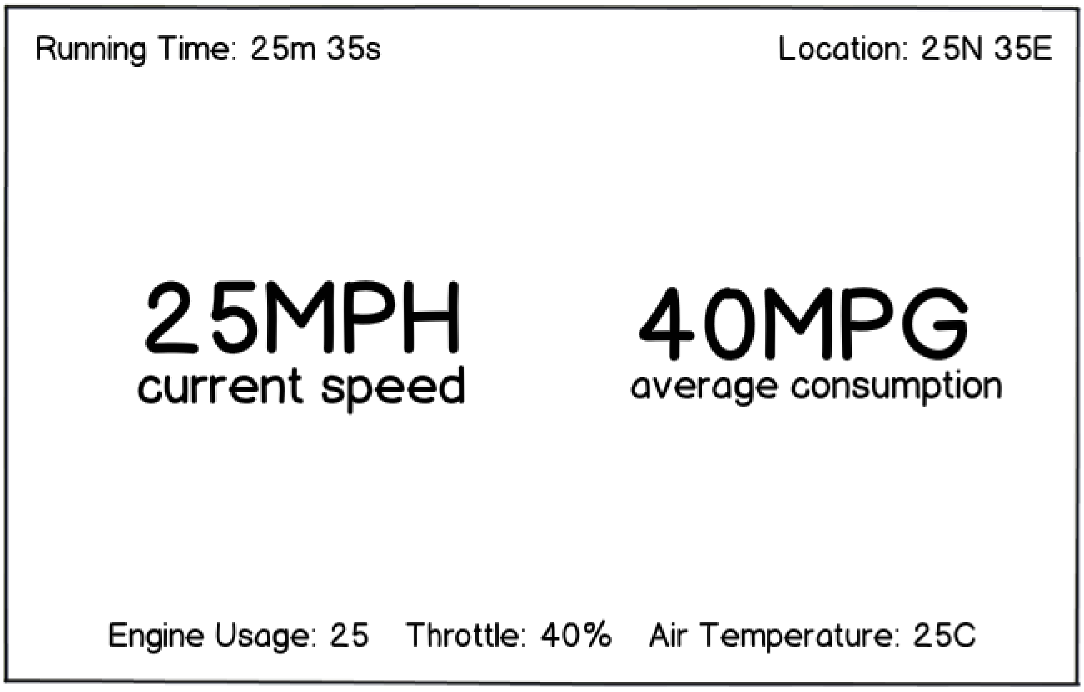


Figure 7: Real-time Data Display GUI

Python will be used to implement the GUI, using it's built in GUI framework, TKinter[10].

5.5 Development Cycle

The following describes the order in which each major part of the software will be implemented during development:

- Connect Raspberry Pi with GPS adapter and OBD cable and enable the Pi to extract data from both these devices;
- Store data from both USB devices to a file on the Pi. We will need to implement the file format detailed above;
- Build web application to upload file and show data in a meaningful way in the browser;
- Implement software that allows for extracted real-time data to be displayed on the LCD screen.

6 Hardware Design

There are four main mechanisms to the hardware design of the PIMPED device. These are described in the following sections.

6.1 Hardware Packaging

The LCD TFT display screen will occupy a separate case from the Raspberry Pi, and will connect to the Pi board directly via the GPIO headers. Although the Pi and Display will be in separate cases, both cases will be in close proximity of each other in order to connect both components.

6.2 GPS Receiver

The GPS receiver will be inside the Raspberry Pi case, interfacing with the Pi via a Breadboard. This will mean the GPS receiver will not be visible to the user.

6.3 Power Source

The Pi will draw power from the 12v lighter socket from the car. Since the Pi runs on 5v power, a transformer or convertor is required in order to safeguard the Pi from power surges, overloads and spikes. The solution decided on is to use a Cigarette-Lighter to USB power supply, which can be connected to the cigarette lighter of any car. The display will then draw its power from the Pi. When the PIMPED device is removed from the car for upload of data to the PIMPED web service, the Pi's standard 3-pin power supply will be used.

6.4 ELM327 to USB connection

To communicate with the on board systems on a vehicle, the ELM327 USB interface is introduced. This will allow the Pi to communicate with the OBDII interface on the car via USB using the ELM327. The ELM327 will draw its power from the OBDII port on the car.

6.5 PIMPED Hardware Flow Diagram

Figure 8 shows a flow diagram of the PIMPED hardware, including the devices main components identified in the sections above.

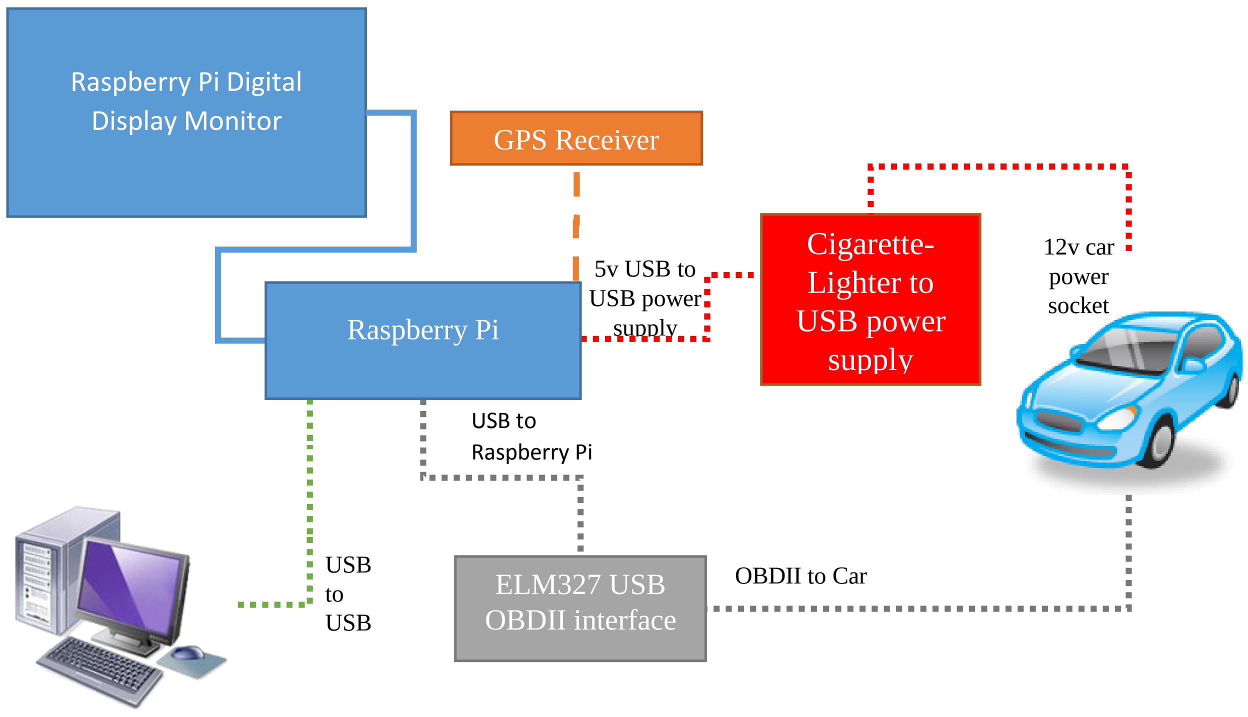
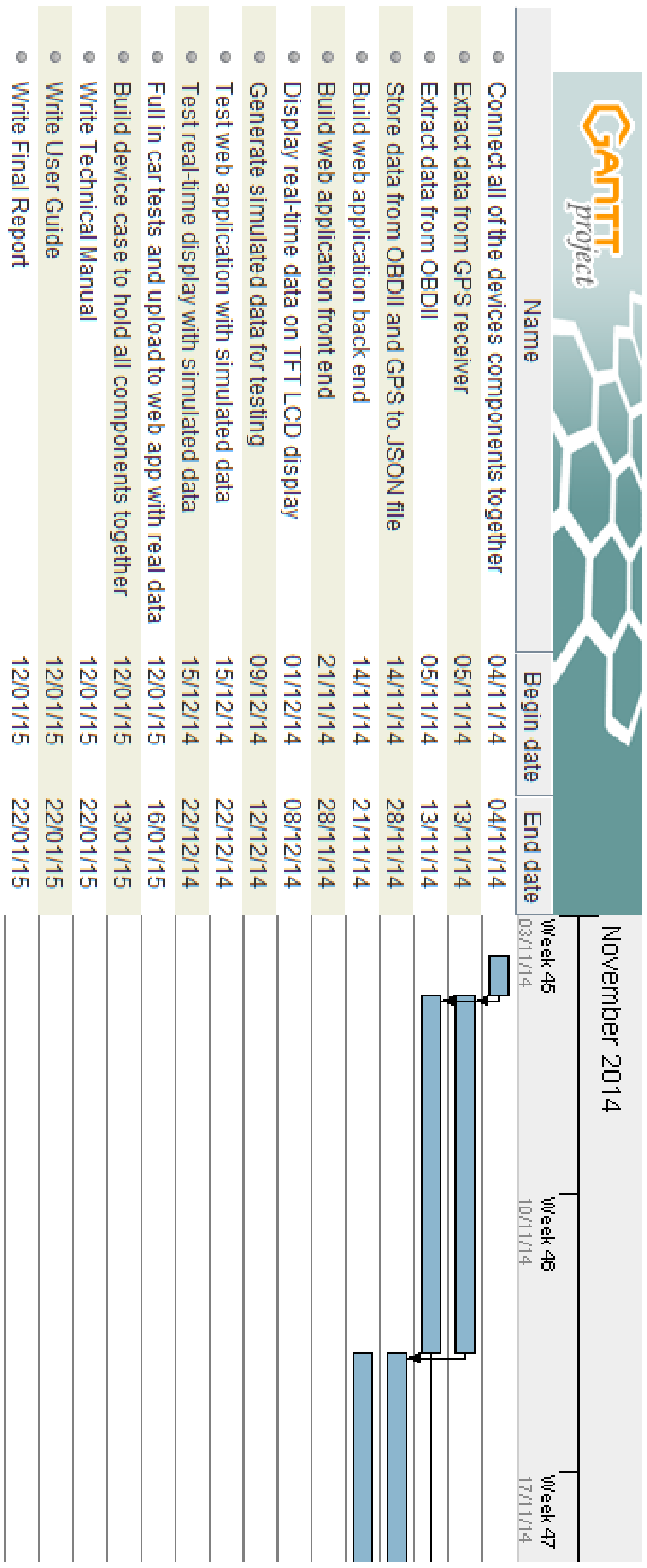

Raspberry Pi Digital Display Monitor
GPS Receiver
Cigarette-Lighter to USB power supply
12v car power socket
5v USB to USB power supply
Raspberry Pi
USB to Raspberry Pi
USB to USB
ELM327 USB OBDII interface
OBDII to Car

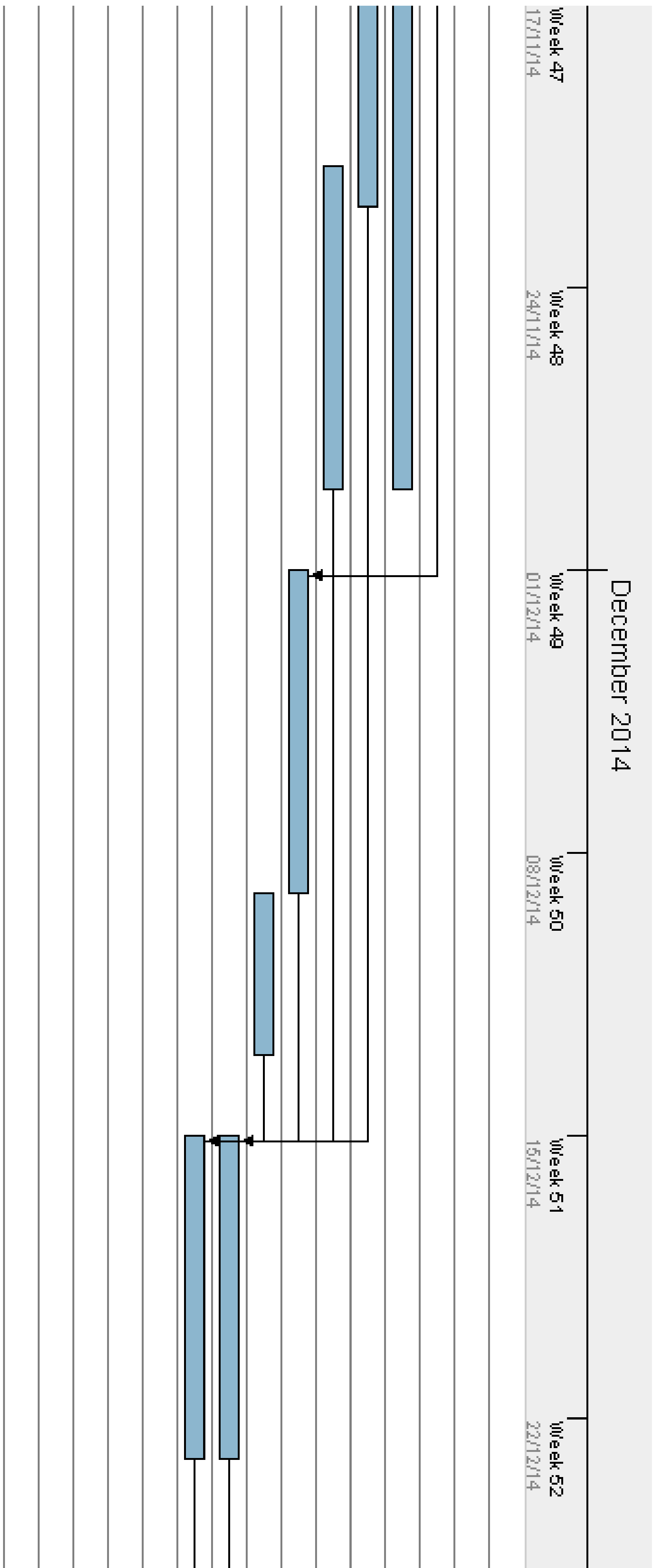
Figure 8: Device flow diagram

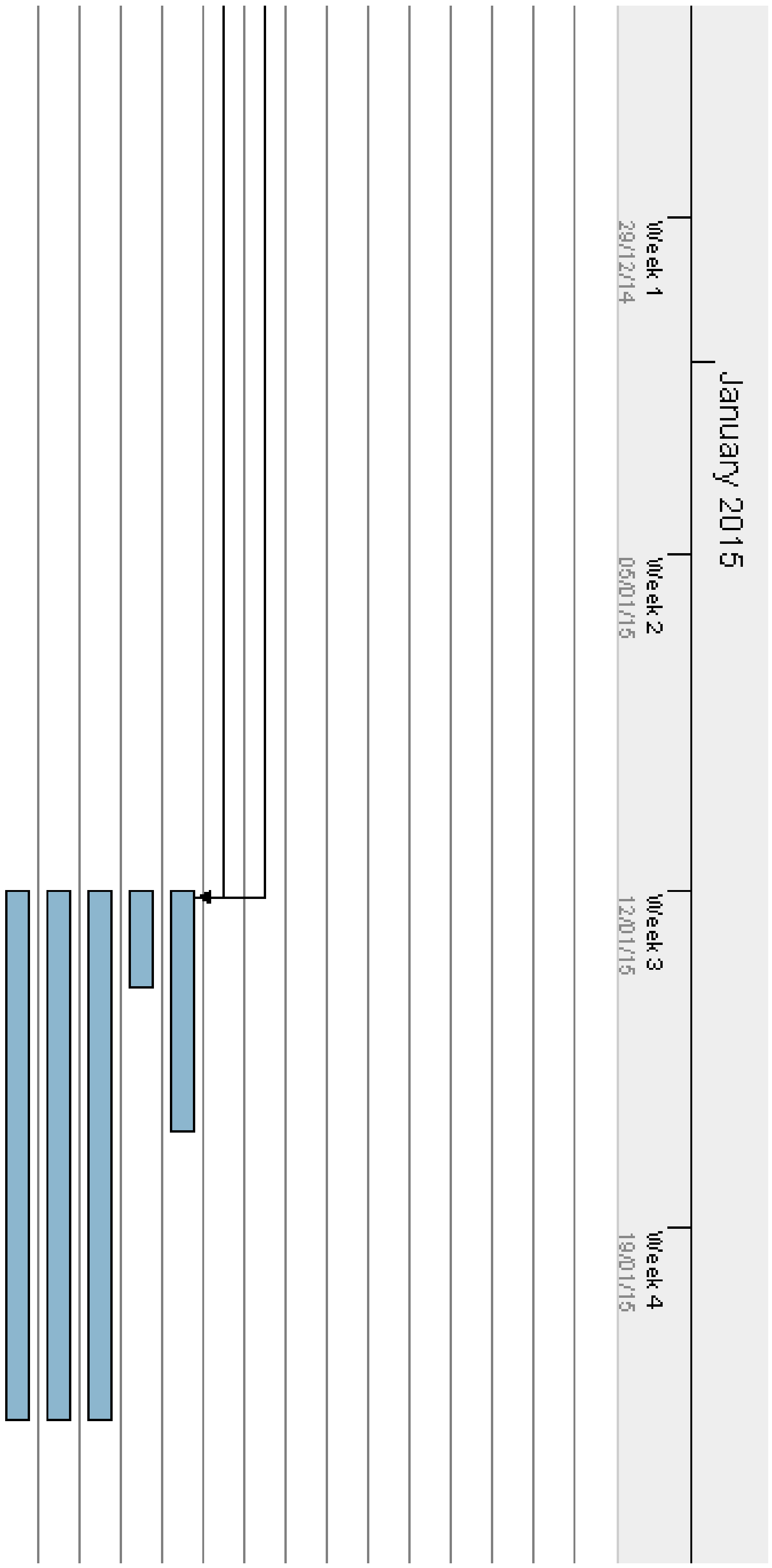
7 Project Plan

We have generated a project plan in the form of a GANTT chart, shown below.



Name	Begin date	End date
Connect all of the devices components together	04/11/14	04/11/14
Extract data from GPS receiver	05/11/14	13/11/14
Extract data from OBDII	05/11/14	13/11/14
Store data from OBDII and GPS to JSON file	14/11/14	28/11/14
Build web application back end	14/11/14	21/11/14
Build web application front end	21/11/14	28/11/14
Display real-time data on TFT LCD display	01/12/14	08/12/14
Generate simulated data for testing	09/12/14	12/12/14
Test web application with simulated data	15/12/14	22/12/14
Test real-time display with simulated data	15/12/14	22/12/14
Full in car tests and upload to web app with real data	12/01/15	16/01/15
Build device case to hold all components together	12/01/15	13/01/15
Write Technical Manual	12/01/15	22/01/15
Write User Guide	12/01/15	22/01/15
Write Final Report	12/01/15	22/01/15


December 2014
Week 47
17/11/14
Week 48
24/11/14
Week 49
01/12/14
Week 50
08/12/14
Week 51
15/12/14
Week 52
22/12/14


January 2015
Week 1
29/12/14
Week 2
05/01/15
Week 3
12/01/15
Week 4
19/01/15

8 Component Pricing List

Component	URL	Price
ELM327 USB Interface OBDII Diagnostic Auto Car Scanner Scan Tool Cable v1.5	http://www.amazon.co.uk/Bluetooth-Diagnostic-Scanner-Engine-READER/dp/B004KL0I9I	£6.99
Raspberry Pi B+	http://uk.rs-online.com/web/p/processor-microcontroller-development-kits/8111284/	£24.35
Adafruit Ultimate GPS Breakout	http://www.amazon.co.uk/Adafruit-Ultimate-GPS-Breakout/dp/B00K9M6T8G/ref=sr_1_cc_2?s=aps&ie=UTF8&qid=1414542902&sr=1-2-catcorr&keywords=Adafruit+Ultimate+GPS	£25.40
SainSmart 7 Inch TFT LCD Display Monitor For Raspberry Pi + Driver Board HDMI VGA 2AV	http://www.amazon.co.uk/SainSmart-Display-Monitor-Raspberry-Driver/dp/B00GZCHHIU/ref=sr_1_fkmr0_1?s=computers&ie=UTF8&qid=1414543157&sr=1-1-fkmr0&keywords=7-inch+Raspberry+Pi+LCD+Digital+Display	£33.00
USB to USB cable x2	http://www.amazon.co.uk/USB-Male-2m-Cable-Black/dp/B0018I97M0/ref=sr_1_1?s=computers&ie=UTF8&qid=1414544174&sr=1-1&keywords=usb+to+usb+cable	£2.34
Cigarette-Lighter to USB power supply	http://www.amazon.co.uk/Maxsimafoto%C2%AE-charger-Cigarette-Lighter-classic/dp/B003I4Q0F4	£3.99
Raspberry Pi b+ case	http://www.amazon.co.uk/Black-Raspberry-Model-Access-ports/dp/B00MQWQT0A/ref=sr_1_10?ie=UTF8&qid=1414549397&sr=8-10&keywords=raspberry+pi+case+b%2B	£4.19
New Pink Soft Silicone Cover Case	http://www.amazon.co.uk/Silicone-Cover-Android-Capacitive-Tablet/dp/B00IMWO3HO/ref=sr_1_5?ie=UTF8&qid=1414549471&sr=8-5&keywords=7+inch+tablet+case	£2.59
	Total	£102.85

9 Conclusion

In conclusion this document has detailed the PIMPED device idea and its core functionality. The main components required to build the device have been identified as well as the cost for each. All of the software and hardware designs have been noted and there is a plan with tasks and estimates which we can adhere to. We will use this document to fuel all of the future stages in our project. The next stage is implementation.

10 References

[1] Adafruit Ultimate GPS Breakout:
http://thepihut.com/products/adafruit-ultimate-gps-breakout

[2] ChartJS:
http://www.chartjs.org/

[3] DigitalOcean website:
https://www.digitalocean.com/

[4] ELM327 OBDII Device Image:
http://make.larsi.org/electronics/ELM327/ELM327v13a_box.jpg

[5] Full list of OBDII extractable data:
http://en.wikipedia.org/wiki/OBD-II_PIDs

[6] MongoDB website:
http://www.mongodb.org/

[7] Nike running app:
http://fitfeat.com/blog/wp-content/uploads/2013/08/nike-app.png

[8] Open Street Map API v0.6:
http://wiki.openstreetmap.org/wiki/API_v0.6

[9] Raspberry Pi website:
http://www.raspberrypi.org/

[10] TKinter:
https://docs.python.org/2/library/tkinter.html

End of Document