***Abstract*—SmartWheel—is a parent control automobile system, which allows to monitor the driving skills and record the driving habits based on the acceleration of the vehicle. The driving records are available both through the local web server and the smartphone application, providing data such as drivers speed, acceleration, and location. Moreover, in case of the potential emergency, the system grants a real time alert through the mobile text messaging.**

# INTRODUCTION

The US teenagers start driving from a very young age, not infrequently putting themselves in dangerous situations due to their immature age. In the US for the year of 2015, the fatal crash rate per mile for drivers, whose age was 16-19 years old is nearly 3 times the rate compare with the drivers 20 years old or older [1]. Young age and youthful maximalism force teenagers to rapidly accelerate on the local roads, slamming on the breaks right in front of the red light, putting not only themselves, but other road users in danger. Of course, parents would quickly cut such a dangerous behavior of their kids while being a witness of such driving habits. But we all know that it is impossible to have a 24 hour parent control over the child, especially when a teenager starts driving a car. SmartWheel device will allow the parents to be sure that their kids are driving safe, rather than exploiting their car as a bolide formula 1. The SmartWheel system consists of two devices: SmartWheel Box and House Box. The SmartWheel box is placed in the vehicle, registering every rapid acceleration or deceleration that is accounted for the dangerous driving behavior. The SmartWheel Box monitors the speed of the car, 

its acceleration, and location. In case of the potential emergency, the parents would receive an instant text message, containing the possible type of the emergency and the location of the car. The SmartWheel system also has a smartphone application, allowing to access the driving history of the child on the go. The desired SmartWheel specifications are listed below in Table 1.The unit price is based on the price of the components the system was built from. Since the SmartWheel box will not always have a chance to synchronize data with the Home box, it will have enough storage capacity to maintain the 72 hour driving record. The device will have convenient size with the low power consumption.

Table 1. List of Requirements and Specifications.

|  |  |
| --- | --- |
| **Requirements** | **Specification** |
| Dimensions | 11.5cm x 8.5cm x 4.5cm |
| Weight | 340 grams |
| Battery Life | 16 hours |
| Unit Price | < $120 |
| Text Message Alert Delay | <10 seconds |
| Storage Capacity | 14 days of driving |
| Server Connection Radius | >70 meters in open area; > 32.5 meters with a wall in between. |

II. DESIGN

## *Overview*

Overall Design

SmartWheel is a monitor system that helps parent to monitor teenagers driving habits. The automobile’s dynamic acceleration is the key parameter we want to measure. Originally, we decided to use OBD2 adapter to get the speed in order to get instantaneous acceleration. After rethinking the problem, we decided to use an accelerometer to measure the acceleration because it not only saves work

for building the adapter, but also could get more accurate acceleration in the 3D setting.

 Other than the accelerometer, we will use a GPS sensor to get the speed and location that related to the acceleration and a GSM module to send the acceleration and coordinate to a pre-registered phone when the acceleration is too high.

A phone app is also need to get the information from text message and plot it on the map.

Finally a Home-Box works as a data server to sort and analyse the historical data.

 

Explanation of Block Diagram

 As we can see from the block diagram, In the SmartWheel Box, the microprocessor we are using is Raspberry Pi 3 [2]. Firstly, The accelerometer cooperate with GPS sensor, when the acceleration exceed a certain point, gps will provide the corresponding location and speed. After pi get the data, pi would let GSM module to send a message to a pre-registered phone.

 On smart phone, the application would gather the data from the text message and show it onto phone user interface.

## Accelerometer

.

 ADXL345 is 3-axis accelerometer chosen for the project due to its low power consumption: 40uA in measurement mode, 0.1uA in standby@ 2.5V, extended industrial temperature range (-55°C to +105°C), and small dimensions: 25mm x 19mm x3.14mm [3]. It is used to measure the dynamic acceleration resulted from car’s motion. Since accelerometer is an important component of the system, it provides high resolution, allowing to compute car’s acceleration vector.

 Technology

 The graph 1 rough data obtained by an accelerometer and further processed by an algorithm is presented in graph 1 by two quantities: angle, measured in degrees, represent the direction of the acceleration vector (red line), and acceleration, measured in g’s, represents the magnitude of the acceleration vector (blue line).

 If we will pay attention to the angle line we can notice eight consecutive peaks with values varying from 150 to 190. This means that eight consecutive left turns were made during the test. Likewise, the angles between 80 and 100 would present the forward acceleration, the angles between 260 and 290 would present backward acceleration, the angles between 350 and 40 would present the right turn.

 The magnitude component, is also presented by eight consecutive peaks, two of which are reaching the values of 0.75 g’s. Those turns were made at a high speed, imitating the dangerous driving behavior.

Results

The high resolution 3-axis accelerometer allowed to compute the car’s acceleration vector, that is used to obtain the car’s nature of motion: rapid acceleration, rapid deceleration, possible car collision. The SmartWheel system provides the device’s misposition detection, which sends a notification message if the SmartWheel Box is not fixed in the steady state. Moreover, the SmartWheel Box is able to detect couple types of dangerous turns: sharp turns on the low speed, gradual turns on the high speed.

## GPS sensor

 Introduction

 The GPS sensor that we are using is from Adafruit[4]. It requires a 5V DC input voltage and only drawn 20mA current which is really power saving. The chip has maximum of 10 Hz for data updating. It will provide coordinates and speed parameter that we need in our project.

Technology

 Thanks for other people that already worked with this sensor and did most of the tough jobs, They developed a nice package call “gpsd” it could convert the raw data that collected from the sensor to “ready to use” data.[5] each data listed here could be retrieved by a single method.



Image 2 [8]



Image 3 [4]

Image 2 shows the raw data that directly get from the gps sensor. and image 3 shows the data that covert by gpsd (time, latitude, longitude, altitude, speed, heading, climb and heading angle from true north) (please note that the number itself does not match). In our project, We implement the latitude, longitude and speed for the location on the map and driving history. We set the data renew speed at about 2 times every second to avoid the overwhelming data that comes into the microcontroller.

Testing

 By testing the data that we gather from the GPS sensor, we got that the latitude and longitude are pretty accurate.

 However we got the speed parameter not equal to 0 while the system was at rest. The speed we got are listed below:

 0.119m/s; 0.035m/s; 0.208m/s; 0.020m/s; 0.241m/s;

 After the long-time running (20 minutes) we found that the speed we got never exceed 0.25m/s. So we concluded that the speed parameter has an error up to 0.25 meter/s which is 0.556 mph.

 Next, we should determine whether this error will have a critical effect on the result when we drive the car on the road. we took an assumption that if the car drives down the road with a speed of 30 mph, the maximum error of the speed parameter is 1.853% which will not have a huge effect on the result.

 Every time that the system reboot, the GPS need to redo the 3D fix. It takes from 45 seconds to 2 minute. It will be even longer when the satellite signal is weak. So in this case, in order to get a better performance, the device is better to be placed at a better signal location like the dashboard of the car under the windshield.

## Communication GSM

 This subsystem pertains to the communication between the SmartWheel and drive ‘s cell phone. When the car staring to moving, any cell phone number which already set up will communicate with the SmartWheel, certain alert text message will send to parent’s cell phone. Also, the parents will automatically connect to it in the future once it been set up in Raspberry pi3. If the cell phone number has not set up yet, the SmartWheel will note that there was no cell phone to communication. For achieve those requirement, the technology are software and hardware part. For the hardware part, we have to install new system document into Raspberry 3, adjust system security level, then install firmware of GSM900A, and disabling Bluetooth, because GSM model and Bluetooth part are conflict by using the same port. There are two choices for a wireless interface for our Raspberry Pi. One choice is the “Wi-Fi USB antenna for the Raspberry Pi” that has a low profile. Unfortunately, this affects distance of communication. Another The better one is by using“GSM900A hardware part”. It will help us have larger range, and the price is lower than the USB one.Python code is the core of software part, we used this language to programming our Raspberry 3, basic we programming the command which GSM900A part could read, the GSM900A part will staring process once it get command from Raspberry pi 3.

 The communication between the SmartWheel and the driver’s cell phone, I will be using SIM900 GSM model and Raspberry Pi 3 for communication. For keep the higher level of security, I will use certain python code to only allowed one cell phone number connect. Depending on the state of the car, the SmartWheel will sent the message out by using GSM model, the user will receive text message which include accurate information for speed, acceleration and location. This device will be determined by the how faster the car was and how faster of acceleration was. If a cell phone number is not registered with a SmartWheel device, it won’t work. When the car is stationary, the SmartWheel will start upload the data to the server. we find a solution to communicate with one subnet. lucky, there is an internet access on GSM model.

 Power Supply

The power supply part was easier to solved. The requirement of power supply should be able to provide 5.0V to Raspberry Pi from car, and also include a rechargeable battery keep SmartWheel working while the car turn off. The reason we need a rechargeable battery is for user to track the car while the vehicle turning off. We were using two circuit to solve this problem, but soon we realized those two circuit are too complex, and it will take more volume in the car. For the reality, we believe people don’t want to put a such big box like 30cm\*35cm\*20cm. By solving the volume problem for the power supply, we found a really efficient way, it also satisfy our requirement perfectly. It’s named Portable charger. The specification of portable charger is provide 5VDC power, the volume only 3.8 × 1.7 × 0.9 in which is smaller enough for the circuit we build. Also the way to recharge the portable charger is just plug in the USB port into the car, it will automatically recharge while the vehicle turning on, and provide the power while the vehicle turning off. Eventually, this device we choose also perfectly fit into our hardware box, and make our SmartWheel working correctly.

## Phone Application

Usage

The phone application acts as the access point for the users of the system. It is in charge of receiving data from the circuit that is to be placed in a car. After receiving the information, it will use the data supplied to plot a point on the map denoting the location at which the acceleration was too high. On the marked place, there will also be information displayed showing the information at that point. The information for each point has its corresponding date, time, acceleration, and street address. As shown in Figure 2, there will be both red points and blue points. The blue points are the points plotted using information from text messages. In cases of extreme acceleration which is abnormal it will receive a text from the box. This will be real time and gives the family members the quickest possible alert in case of emergencies such as crashes. The red points are points plotted using data from the server. The blue points are plotted automatically when the app is started. The red points are plotted based on the user’s commands. First the phone has to have access to the internet and then click on the “Get Data” button. This tells the phone application to connect to the webserver to obtain the data stored. After the data is obtained, the user can click on the “Plot Points” button to plot the points that fit the chosen parameters. The parameters are chosen by the user through the two drop down menus at the top of the application. The first one lets the user choose the acceleration threshold. Any points with accelerations under this chosen value will not be shown. The second drop down menu lets the user choose the time frame. Any points that are older than the chosen amount of days will not be shown. In figure 2, all points with acceleration greater than .3g within the last 30 days are plotted.

Technology

The phone application is built for android phones. It is developed and tested on the Android SDK and coded using java. The emulator built into Android SDK can emulate different models of android phones as well as different APIs used by them. It can also emulate text messaging making it easier to test the accuracy and functionality of the app. It is also possible to test on a physical android phone if it meets the requirements of the app. Currently, the android app is fully functional for phones running Android API levels 17-22. The most recent API level for android phones are API level 25. The mapping interface is implemented using the Google Maps API.

Techniques from Courses

The techniques used to build this block come primarily from software classes. The phone application is mostly coding so the prior experiences from coding classes have been very helpful. Data structures are an important part for this phone application. It allows the phone application to store and search through the information more efficiently. As the data stored in either the phone or web server grows, the use of data structures will become more and more important. Software program testing methods and techniques were also used. This helps minimizing the amount of errors that occur when using the app. Techniques and concepts from ECE 374, Computer Networks & Internet were also used. When the phone application connects to the internet to obtain the data from the server, it is achieved through the use of TCP connections as well as HTTP connections. Though, most of the protocols were handled automatically by the internet provider, it was still helpful information to know when implementing the connection for requests. The knowledge made it easier for me to understand what type of connection was best suited for the type of data I needed to obtain. These choices help make the application run faster and not waste unnecessary processing power.

Testing

There were many tests used to verify the functionality of the phone application. Two of the most important functions of the application are to read and plot information from a list/web server and to get access and read text messages of a phone.

Initially, the test used to test the first function was to make a list and verify the outputs were correct by controlling the data stored on the list. Currently the most recent nine points that exceed the acceleration limit are plotted onto the map. To test this, I tested what happens if there are less than nine points of data on the list. I then tested what happens to a point plotted on the map if the acceleration was lowered underneath the limit of the acceleration specified. I also tested what happens when a new points that are both above and below the threshold were inserted into the data. I also tested how the location of the marker changed and saw if it was accurate or not.

Since the MDR, we have changed from reading data from a local list to reading data from a web-server. First I ran tests similar to the ones ran on the list. One thing that was changed was that there was no longer a limit of markers put on the map instead we used a new user chosen acceleration and date threshold to limit the amount of points on the map. We would take the Smart Wheel box into the car and drive around while testing the real time alert function. Afterwards, we would connect the box to the internet and have it load the data from the test drive onto the server. The phone app was then opened and used to retrieve the data from the server. We would then set the acceleration threshold close to the one set for the real time alerts. We expected to see a cluster of points close together for each time we received a text message during the test drive. We would then plot all the points from that drive and see if a trace of the route we took on the drive was shown. After all points were plotted, we could make out the route taken during the drive and confirmed the function was working.

To test that the correct information was taken from the text messages, I first stored the information of the most recent text message on the phone and the displayed it to make sure the correct information is retrieved from the text messages. I then had it send a new message received a text message from a different number to see if the latest message from the specified number is stored. I also tested having the app read text messages with slightly altered messages to ensure it reads lines and plots it in case of a formatting issue from the sender. The team then went on a test drive with the smart wheel box inside the car. We would find relatively empty locations and make the box trigger and send a text message. We would then open up the application to confirm that a point was plotted on the exact location we were at.

## F. Home Box

 Usage

 We named it as Home Box is because this is a data server that runs on user's private PCs or laptops. The purpose that we build home box is that we want SmartWheel box could automatically connect to server when it is connected to home Wifi and also automatically transfer history data. Users could either open the phone app to see the driving course or open the web user interface to see the graph with latest driving behaviors.

 Techniques

 In order to accomplish this usage, the first problem we encountered is that how to transfer data between raspberry pi and home PCs. In order to accomplish this problem, we use socket programming to build a TCP server and client.



Image.6 socket programming block diagram[9]

 This block diagram shows how our socket program works. For the server side, create a socket system call, bind a socket onto an address and then listen for connections. As the client side, create a socket system call, connect to the server with the appropriate address, after server accept the connection, the data could start transferring. The data processing on the server side and the client side are simply the writing and reading method for text files.

 The second problem we encountered which is on the client side is that how to make SmartWheel box stop receiving data while it connecting to server, and how to make it resume working after it disconnect from server. In order to accomplish this problem, we add a secondary loop in the code to let the SmartWheel continually pings the server address. When the connection was made, SmartWheel box goes into a infinite loop. After the SmartWheel box disconnect with the server, the loop breaks and program resumes.

 We establish the server using MAMP[10]. It could install a internet server environment in seconds. MAMP will also not compromise any Apache installation already used on your system.



 We also build a simple web user interface to show the recent driving acceleration history.



image.7 the recent driving acceleration diagram

This graph shows the driving acceleration history for our test drive on the FPR demo. There are two excessive driving detected and the text messages are all received.

 The x-axes refers to the time line and the magnitude of acceleration is on the y-axes with the unit of m/s^2. For plotting graphs on html website, we use Microsoft One-Drive excel. It provide website embedded code for website programming that allows programmer to plot the graphs directly on the website without using any other API [11].

# III. PROJECT MANAGEMENT

Over the course of this project, the use of tools such as timelines and Gantt Charts were extremely helpful. These tools helped us keep on track and showed us what else was left to complete. Online editing tools such as Google Docs has also helped us keep track of each other’s progress. We can see the exact progress each team member has made and make sure no one is doing overlapping work.

Overall, the team has been able to work with each very well. Because we had an even split of EEs and CSEs we had an easy time splitting the parts between the team members. We also had someone else to go to who had a similar enough background for help in case there was a problem they were stuck on. We have meetings once a week with the team and advisor once a week to discuss the progress and readjust the goals for the next week if needed. There would also be other meetings held, normally once a week and sometimes more often as needed. Bingjun Wang was in charge of the GPS sensor as well as the home-box/server. Bingze Li was in charge of the GSM communication as well as the power supply. Jack Tam was in charge of the Android phone application as well as processing and filtering the data. Yaroslav Burdin was in charge of the accelerometer, did most of the mathematical calculations and also assumed the responsibilities of the team manager. .

# IV. CONCLUSION

Since the MDR, we have successfully integrated all the separate subsystems together to form a whole, cohesive system. The three main subsystems can now communicate with each other concurrently as needed. We now have a power supply for the system so that it can operate in the car without being plugged in. We also made an interface for the server so that some of the data can be accessed without using the phone app. We have also added a user interface to the phone application so that it is more interactive. The data is also utilized more so that the real time alerts give more information other than the acceleration. We also created a box for the parts of the system to make it more like a completed commercial product.

Overall, we have completed the project to a level that the team is satisfied with. We have implemented everything we had initially planned to implement. Throughout the course of the project new ideas of functions to add to the project were suggested. Though we have not implemented all the extra features most of the main features were added. Due to some time constraints the other functions/features that that could have made the project better and were suggested were not implemented or fully implemented. Given enough time we would have liked to added more interface for the website of the project. As of now it is minimalistic in the amount of information presented compared to what we could show with the data stored.

Acknowledgment

The SmartWheel was supported by Professor Lixin Gao who provided insight and expertise that greatly assisted the project.We would also like to show our gratitude to Professor Maciej Ciesielski and Professor William Leonard for feedback that greatly improved the SmartWheel.

We are also immensely grateful to Jesse Cafarelli, Sergio Cunha, Benjamin Miller and Ray Paradis for sharing their ideas and wisdom with us during the course of this project.

References

[1] U.S. Department of Transportation's *Fatality Analysis Reporting System*. Posted November 2016, Available at: <https://www-fars.nhtsa.dot.gov/Main/index.aspx>

[2]Raspberry Pi Ltd. “*Raspberry Pi Compute Module 3 Lite*” Revised 2016

[3]Analog Devices. *ADXL345 DataSheet.* Available at: https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf

[4] Townsend, Kevin. "Adafruit Ultimate GPS on the Raspberry Pi." *Setting Everything Up | Adafruit Ultimate GPS on the Raspberry Pi | Adafruit Learning System*. N.p., n.d. Web. 22 Dec. 2016.

[5] GlobalTop Technology Inc. “*FGPMMOPA6H GPS Standalone Module Data Sheet”* 2017

[6] Google. *Google APIs for Android. Available at:* https://developers.google.com/android/

[7] Android. *Android Developers.* Available at: <https://developer.android.com/index.html>

[8]Turpin, Matt. "*File:GPS Raw Data Stream." File:GPS Raw Data Stream - Northwestern Mechatronics Wik*i. N.p., n.d. Web. 11 May 2017.

[9]*CS330 Sockets*. N.p., n.d. Web. 11 May 2017.

[10]*GmbH, Appsolute. "MAMP & MAMP PRO." MAMP & MAMP PRO. N.p., n.d. Web. 11 May 2017*.

[11]*"In OneDrive, Right-click the Workbook, and Then Click Embed." Excel. N.p., n.d. Web. 11 May 2017.*