Road Safe Phone Case-Phase 2

Design Document

Team number: sddsn-15
Client: Christine Shea-hunt
Adviser: Mohammad Tayeb Al Qaseer
Team Members:
Scribe: Shuang, Shihab
Facilitator: Philip
Software Lead: Ali
Hardware lead: Isaac
Test lead: Chad
Team Email: (temporary)pmarkose@iastate.edu
Team Website: http://sddec20-15.sd.ece.iastate.edu

Revised: Date/Version
02/23/2020 V1.0
03/29/2020 V2.0

Executive Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the engineering standards that apply to this project that were considered.

- Circuit Schematics
- Parts list
- Software libraries
- In-line comments
- Data security and privacy
- Digital simulation results
- User's manual
- Instructions and warnings

Summary of Requirements

- Microcontroller
- Android phone or iOS phone
- OBD splitter
- Bluetooth transmitter
- PCB boards
- Circuit assembly kit
- Battery

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

- EE 230
- CprE 281
- CprE 285
- CprE 288
- EE 330
- CS 309
- SE 319

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- Python
- AutoCAD
- Purchase of hardware parts
- Bluetooth protocols
- Microcontrollers

Table of Contents

List of figures and tables

1 Introduction

1.1 Acknowledgement

ETG - Provided Gitlab Space and website.

1.2 Problem and Project Statement

The problem we are trying to solve is teenagers using their phones while driving. We are trying to limit teenager's access to their phones while driving a car so they can be less distracted and more focused on driving and thus reducing the risk of car accidents. The approach we decided to take is designing a system that will block all notifications from a teenager's phone while driving a car. We will achieve our goal by creating a system that consists of three parts; car device, driver's application (app), and parent's application. 1) The device in the car will read the car gear and send a signal to the driver's and parent's apps. 2) The driver's app will block all notifications from the driver's phone and will not allow that phone to be used while the car is driving with the exception of navigation and music. 3) The Parent's app will check if the driver's app is in use and if the car device is functioning. By making all these components work together we will make driving safer for teenagers and will reduce the number of accidents related to phone use while driving. This project will also help parents be more comfortable with their new teenage drivers driving and will reduce their worry and stress.

1.3 Operational Environment

The environment presents challenges for hardware and design of the On Board Diagnostics (OBD) unit. Since the OBD stays in the car at all times, we have to consider the car's internal temperature throughout the year. Our device has to perform when exposed to -20^{o} and 110^{o} Fahrenheit. Also, the unit should endure vehicle activity. We expect it to withstand speed bumps, hard acceleration/braking, reasonable car accidents. Finally, our product has a size constraint. The OBD plugs into the Engine Control Unit (ECU). If the OBD interferes with the driver's operations, we are putting the passengers at risk. This means the OBD can't be very big.

1.4 Requirements

Software Functional Requirement:

1. All user types shall be able to login to their account.
2. App shall enter driving mode after signal is received from the OBD sensor indicating car is started.
    a. App shall enter driving mode a set amount of time after signal is received to allow the driver to warm up the car during winter.
    b. Driving mode
        i. In driving mode, the app shall record each time the app is exited.
        ii. The app shall send a real-time notification to the Parent to notify them unexpected behavior has occurred.
3. App shall exit driving mode after signal is received from the OBD sensor indicating the car is turned off.

4. App shall track the history of child's unexpected behavior(exiting app while driving)

Software Non-Functional Requirement:

1. App shall display information clearly with decent color contrast and emphasis on important information.

Hardware Functional Requirement:

1. The OBD sensor shall connect to the driver's app within a certain range.
2. The OBD sensor shall detect the status of the car: Car in use/Car not in use
3. The OBD sensor shall detect the connection status of itself.
    a. The OBD sensor shall record each time it is unplugged from the car.
    b. The OBD sensor shall send the information of unplugging to the parent's app within a certain range.
4. The OBD sensor shall react to parent's app's ping within a certain range.

1.5 Intended Users and Uses

Our client approached us for this project with the intentions of her daughter using it while she is driving. Her daughter is 15 years old and is just beginning to drive. Our client also has an older daughter who got in a car accident while she was texting and driving. She is hoping that the device we create will be able to prevent her younger daughter from following in her sibling's footsteps and not be in a car accident. If we are able to create a product which works well with our client's daughter, she is hoping to then sell this product to other parents wanting their teenagers to be safer and more responsible drivers. The issue with teenagers using their phone while driving is a growing issue and our client recognizes there are limited options out there for parents to keep their kids safe while driving. Because of this, we will be designing our product to be used by teenagers only, with no intentions for it to be used by adults.

1.6 Assumptions and Limitations

Assumptions

- That cars were manufactured after 1996 - we will use the OBD port for our hardware bluetooth device
- That the parents are going to check if the bluetooth device is still plugged into the OBD port of the car.
- That the microcontroller will allow us to communicate with the cars main computer
- That the user is well versed with technology - the user should be able to know how to download and install Android or iOS applications
- That the user has a smartphone that will allow him/her to download the application and also access to bluetooth

Limitations

- The cost to produce the end product shall not exceed $150 - that way it makes it cheaper in case it ever goes into mass production
- The end product shall be no larger than; length 5", width 4" and height 1" - this is because it has to fit in the car's drivers compartment without causing discomfort while driving

- The user's phone must be within a few feet for the bluetooth device to connect - if a user is out of range, the phone will not connect to the bluetooth device which in turn does not activate the application

1.7 Expected End Product and Deliverables

Our deliverables consist of two products. The first is an On-Board Diagnostics (OBD) unit. The OBD plugs into the car's Engine Control Unit (ECU) and detects if the engine is started, gear status, vehicle speed, etc. This information will be transmitted to the driver's phone via Bluetooth.

The second of our deliverables is an app. The app has two account types, "child" and "parent". The driver signs into the "child" account, and the app receives the OBD data. Using this data, the app determines phone privileges. These privileges include GPS and phone audio like music, phone calls, and navigation assistance. If there's an attempt to violate privileges, the "parent" account is notified. There are exceptions for emergency protocols where the "child" is given full access to their phone.

A short document similar to an instruction manual will be included. It will instruct the user where to plug in the OBD, how to connect the OBD to the phone, and how to navigate through the app. The first prototype will be released in May 2020, and the final in December 2020.

2. Specifications and Analysis

2.1 Proposed Approach

From our team discussion, we decided that there are two possible approaches to solve the problem described in section 1.2 of the report. The first approach is to design a phone-case. This case will have a mechanism that allows either the phone or the car key inside at a time, so the driver will only be able to access the car key if the phone is secured inside the case and vise-versa. After almost three weeks of brainstorming and discussing this approach, and on the basis that teenagers might want to listen to music, use navigation, and be in an emergency situation, we decided to abandon this approach.

The second approach the team decided to adopt and implement was the approach discussed in section 1.2 of the report, which is designing a system to block notifications and apps from the teen driver's phone while driving. After agreeing to this approach, we started researching ideas to implement the system. We are now in the research phase of the project. We are trying to understand the different elements of the system and a way to connect them together to achieve our goal. Due to the many requirements of this approach (see 1.4) we decided to divide the tasks and assign each member a specific task to research. In the coming weeks, we will start with the next phase in which we will start purchasing the required elements and pair them together.

2.2 Design Analysis

We have spent most of our time brainstorming, researching, and designing our product thus far. Initially, our product was supposed to be a case which would deposit your car keys when you put your phone in it. We came up with many issues with this idea however, for example: How would you unlock your cars if your keys were in the case? How would you get your keys if you did not have your phone? What if there was an emergency and you needed your phone to call 911 while the car was running? Would the user need to carry this clunky box around with them everywhere they go? We approached the client and informed her of these concerns and asked her if she would be okay with going a different route which included an app that locks your phone instead. She obliged and we got to work brainstorming ways we could implement this. We came up with many ideas but quickly found ways that the teenager could easily work around and still be able to use their phone while the car is running. We then had an idea to implement a feature that would alert the parent if the child has used their phone while driving or if the child has disconnected the device from the car. She thought this was a good idea so this is what we will be going forward with.

Something we are going to potentially modify in the future is what components use bluetooth to communicate and what components will communicate through USB or wiring. We have concerns about bluetooth drawing too much power and draining the battery. The biggest weakness we have right now is a lack of experience working with phone applications, bluetooth, and OBD-II programming. We are all doing research to quickly familiarize ourselves with it however. The biggest strength is that the entire team has a very good idea about what our final product is going to look like and the requirements it needs to meet. Overall, I think our proposed solution is very solid and is way better than the initial phone case idea that we had.

2.3 Development Process

The development process we chose to follow is the agile development approach. The reason we chose this approach is due to multiple reasons. First, the fact that we are a team of students who usually do not have schedules that align. The agile approach allows flexibility and adaptability that we think will be very beneficial to us. Second, we would like to push out a prototype as soon as possible and the agile approach will allow fast paced development. Third, the agile approach enables frequent efficient face to face communications. This will keep our team in sync and allow us to have the same goals in the back of our minds.

2.4 Conceptual Sketch

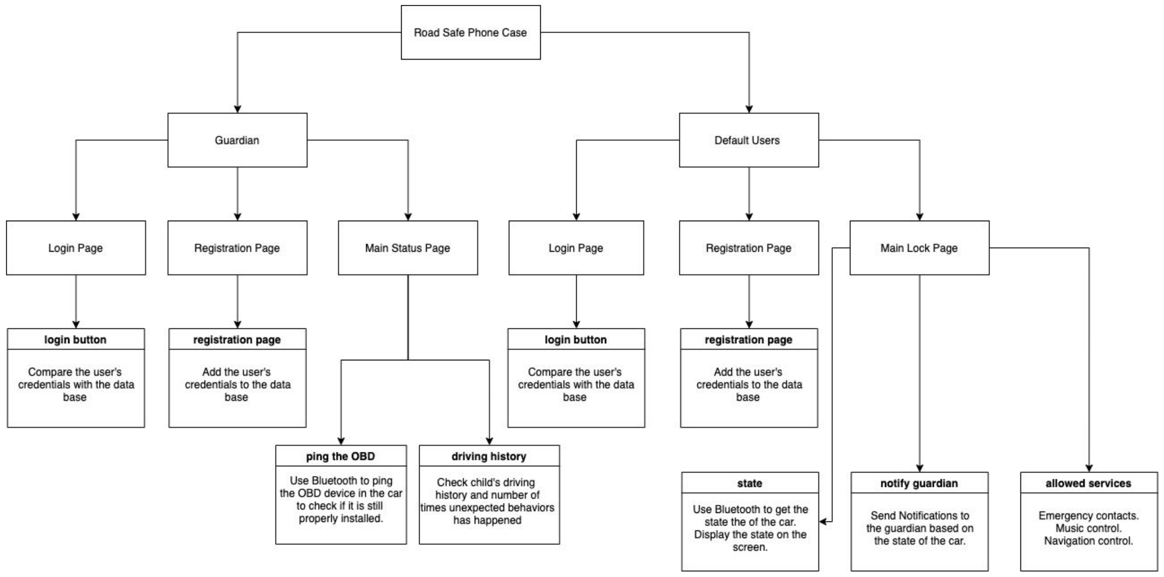

Road Safe Phone Case
Guardian
Default Users
Login Page
Registration Page
Main Status Page
Login Page
Registration Page
Main Lock Page
login button
Compare the user's credentials with the data base
registration page
Add the user's credentials to the data base
login button
Compare the user's credentials with the data base
registration page
Add the user's credentials to the data base
ping the OBD
Use Bluetooth to ping the OBD device in the car to check if it is still properly installed.
driving history
Check child's driving history and number of times unexpected behaviors has happened
state
Use Bluetooth to get the state the of the car. Display the state on the screen.
notify guardian
Send Notifications to the guardian based on the state of the car.
allowed services
Emergency contacts. Music control. Navigation control.

Figure 1 | Conceptual Sketch of Product

3. Statement of Work

3.1 Previous Work And Literature

There are a couple of similar products that are on the market currently. Most of those products rely entirely on software to implement the functionality. That is because they are intended for users who are trying to improve their driving by not using their phones, however our task is to help the parent or the guardian keep their child safe by not allowing them to use their phones while driving.

This makes existing products implemented only in software flawed for our purposes. The kid could simply disable wifi or not start the app. This is due to the fact that if you rely entirely on software there is no way to know if the user is in a car or not. Those apps also do not include any functionality for a parent for example to check how their child is doing.

There was one product that we found that is close to our design, The Shellback Safe which is a small case where you put your phone when driving (1). Even though this product implements a lot of the functionality that we were looking for, there were two concerns about the safe that we wanted to address in our product. First, similar to some of the apps that we found, the product does not get any data from the car, this means that you can not guarantee that the child will take it with them on every trip. Second, there is some functionality on the phone that we would like the user to have access to, like navigation and music.

The products that we found online are useful for their specific purposes, but we did not find a product that fits perfectly our functionality requirements.

(1) S. Cohen, "Stop Distracted Driving: Put Your Phone In The Shellback Safe," Digital Trends, 16-Feb-2017. [Online]. Available: https://www.digitaltrends.com/cars/stop-distracted-driving-put-your-phone-in-a-shellback-safe/. [Accessed: 30-Mar-2020].

3.2 Technology Considerations

Since we are going to be using a combination of both software and hardware in our project, I believe that our project will have more strengths than weaknesses. One of the strengths of our technology is that it's cheap because it's mainly dependent on software which is far cheaper than hardware. The other strength is that the Android and iOS application we are going to develop will allow us to make modifications and adjustments without necessarily changing the overall project. also, it gives our users control of the technology since it is designed for our customers.

One weakness of our technology is that we do not know how to read and decipher the codes from the vehicle's computer so we have to use an OBD-II reader which means we have to spend more money on the project.

An alternative implementation is to create an Android and iOS application that can use GPS to detect the speed the phone is travelling in and if it's above a certain speed, then it should lock the user out. Another alternative is use GPS to let the application know the location of the phone, and if the phone is on a street, then it should shut the user out.

3.3 Task Decomposition

Our entire project can be stripped to 5 major tasks:

1. OBD Scanner reads car's diagnostics and transmits input to Raspberry Pi
2. Raspberry Pi interprets data and sends encoded char to bluetooth connected device
3. Bluetooth connected device receives encoded char and relays information to mobile app
4. Mobile app decodes char and limits user's phone accessibility accordingly
5. Submit a working prototype.

3.4 Possible Risks And Risk Management

The widespread of the COVID-19 (Novel Coronavirus) in the world forced the university to take unprecedented measures to reduce the risk of the virus. Iowa State University decided to cancel all in-person classes and suspend all university activities for the remainder of the semester, which created a great deal of inconvenience for our team. Therefore, canceling all face-to-face interactions for this course.

We decided to keep having our regular team meetings virtually keep having meetings with the advisor and client virtually as well. But we realized a bigger problem regarding the hardware part of the project. Each member of the hardware team has a physical component and we cannot have face-to-face meetings. Furthermore, each member is in a different state of the country and for the hardware system to work we need all the components to work together. Therefore, we decided to divide the work so each member will do all the tasks regarding the component they possess then after finishing their part, they must send that component by mail -which will cause some delay- to the most capable member to assemble the component. For the software team, they were able to work completely virtually on their part of the project.

Another problem our team faces is the closure of all computer labs at the University. This problem is more apparent when working with the raspberry-pi since it requires two desktops to work on it with ease; one for the Raspbian software and another to research. We worked around this problem by using the Raspbian on a laptop and doing the research by phone.

Lastly, extensive testing might be needed in later stages as cars have high security and reliability requirements, which means we have to do extensive research on the safety protocols in cars.

All in all, the Novel Coronavirus is causing logistic and time problems for the team, but we are still determined to complete a working prototype by the end of the semester.

3.5 Project Proposed Milestones and Evaluation Criteria

- Stage 1: SW: Beta app complete. HW: Bluetooth is set up, OBD sensor connects to Raspberry Pi and sends data through the Pi.

  Test: SW: Install the app on testers' phones, test that user can successfully register and receive real-time notifications. HW: Test that data is sent from OBD sensor to Raspberry Pi and is successfully received.

- Stage 2: Integrate SW and HW

Test: Install the app on testers' phones, then test the app is able to connect to the Raspberry Pi through bluetooth. Make sure data is sent to the app and is successfully received.

- Stage 3: Design a chip to replace the Raspberry Pi to reduce the power needed and size for hardware.

  Test: Run the same tests on the chip as the ones ran for Raspberry Pi, make sure all functionalities are successfully implemented to replace the Raspberry Pi.

- Stage 4: Integrate SW and HW

  Test: Same as Stage 2 testing, but with the newly designed chip.

- Stage 5: Beta Testing

  Distribute the testing version of the app and chip to other testers, and receive feedback from them for future improvements.

3.6 Project Tracking Procedures

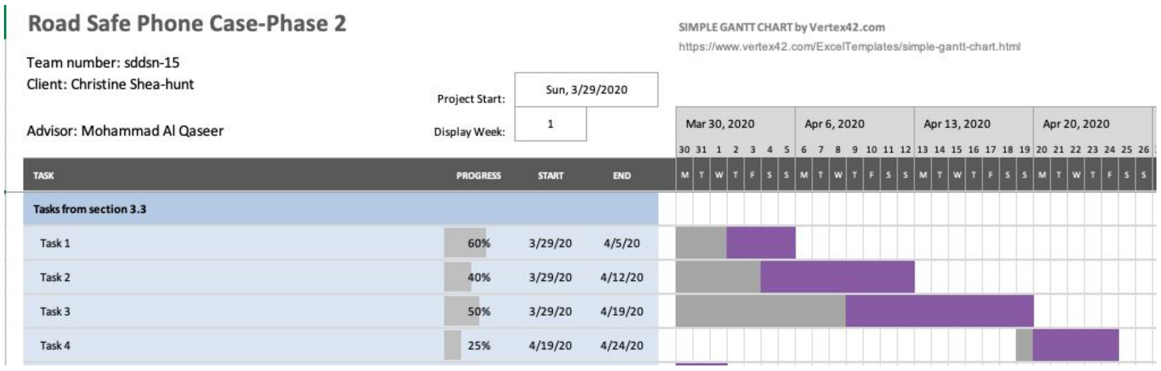
There are a couple different ways we have been tracking our progress. The first is by taking our bi-weekly reports seriously and putting effort into the content we put in them. We are able to use them to reflect on our progress and see whether or not we are on track to complete our goals on time. It also shows the rest of the team how much individuals are contributing to the project and whether someone needs help with their parts of the project. The second way we have been tracking progress is through our website. We have been uploading files to the website that clearly document what our team is currently working on, what we have accomplished, and any problems we may have run into. This is very useful to our client and advisor, since they can see the progress we are making daily. This has also been nice for individual group members to get a better understanding of the progress others in the group are making. This has allowed us to support each other better by asking questions or sharing any comments we may have about each other's work. Lastly, we have been tracking our progress through weekly meetings. At these meetings, we share what we have accomplished and what we plan on accomplishing within the next week. The notes we take during these meetings are uploaded to an ongoing entry log where everyone is able to look back and see what was discussed and where the project was at. We believe these three methods have been very effective this semester and we plan on continuing to use them the same throughout the rest of the semester and next year.

3.7 Expected Results and Validation

Our expected result for the end of the Spring 2020 semester is to have a working prototype. We expect this working prototype to consist of an application which can work on both Android and iOS platforms, and should be able to lock all functions of the phone. The app does not need to have emergency contact capabilities or have a visually appealing interface. This app should be able to read data sent over bluetooth from a Raspberry Pi. The Raspberry Pi will need to be programmed with the proper protocol to make this communication efficient and clear. The Raspberry Pi will use the PiCan2 board to decipher OBD-II readings from the OBD-II port on the car. The Raspberry Pi should be able to detect from the PiCan2 when the car is in park and when it is not in park. It will need to continuously send these messages via bluetooth to the phone. For the final product, we expect to have all features from the prototype, with a few additional functions added on, along with moving away from the Raspberry Pi to make it more energy efficient, cheaper, and smaller. The application will be visually pleasing, and have the capabilities to contact emergency services in the case of an accident. The hardware should be able to detect whether the phone is connected and locked if the car is not in park. If it is not connected and the car is not in park, it should be able to store that memory, and connect via bluetooth to the parent's phone and inform them that the car has been driven without the phone being locked. It should also be able to store the memory if the controller has been unplugged from the car, which should then inform the parent of such when they are in range of the bluetooth connection. We plan on conducting many tests with both the prototype and the final product to ensure that the product we are developing is infact one that is of high quality and ready to be put on the market.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline



Road Safe Phone Case-Phase 2

Team number: sddsn-15
Client: Christine Shea-hunt

Advisor: Mohammad Al Qaseer

Project Start: Sun, 3/29/2020
Display Week: 1

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

TASK	PROGRESS	START	END
Tasks from section 3.3			
Task 1	60%	3/29/20	4/5/20
Task 2	40%	3/29/20	4/12/20
Task 3	50%	3/29/20	4/19/20
Task 4	25%	4/19/20	4/24/20

Figure 2 | Project Timeline

From the above Gantt Chart; a working prototype must be completed before the end of the semester by 04/29/2020. Our team decided to finish a working prototype of the project this semester. The prototype will not be commercial or feasible for use by regular people. This semester we are focusing on proving our concept and making a minimal feasible product, but for the next semester, we will focus on reducing the size, cost, and complicity of the project. The following paragraphs will describe how we are intending to do this.
September 2020: in the first month of next semester, we will start researching and brainstorming ideas to cut the cost and improve the product design to make it commercial. This period will be a vetting process for the ideas we have and in it, we will try to get help from a product design engineer we know. By the end of the month, we should have a clear idea about the way we want the product to look and how much it will cost to manufacture it in a big quantity.
October 2020: This month will be the main month and we will do most of the physical work in it. our goal by the end of this month is to have submitted a commercial prototype to a manufacturer.
November 2020: This month we will start to look for cheaper ways of manufacturing and ways to market and sell the product in a big market. Depending on our client, we might want to look for investors to help with the product. On Monday, December 7th, 2020, we must be done with the project completely.

4.2 Feasibility Assessment

One foreseeable challenge is the bluetooth connection between the app and the Raspberry Pi. Team members are actively researching pre-existing examples and aim to understand how these examples are implemented. We have also found documentation for both Android and iOS regarding using bluetooth within an app, which could be helpful in our case, but we still need to figure out how to connect to the Pi or chip designed by the hardware team.

Another potential challenge is getting accurate OBD-II sensor readings. We need an OBD-II decoder to translate data from the ECU of the car to data that we can read using the Raspberry Pi which then sends the information to the mobile apps through bluetooth. Currently we cannot identify an existing OBD-II sensor that fits our needs on the market. If we still cannot find a suitable OBD-II decoder by the end of this stage, we will follow the tutorials online to design our own reader using a PCB board.

4.3 Personnel Effort Requirements

SOFTWARE

Personnel	Task	Explanation	Hours
Shuang	Software- iOS Full stack	Design iOS specific UI. Co-design backend database data structure. Build the iOS app that complies to the software functional diagram, which includes the bluetooth communication with microcontroller.	20
Ali	Software - Android Full stack	Design Android specific UI. Co-design backend database data structure. Build the Android app that complies to the software functional diagram, which includes the bluetooth communication with microcontroller.	20

HARDWARE

Personnel	Task	Explanation	Hours
Chad	OBD-II reader	Chad is responsible for understanding and deciphering the information provided by the OBD-II reader.	20
Shihab	Bluetooth Protocol	Shihab is responsible for configuring the bluetooth protocol that will be suitable	20

		for the communication between hardware and software.	
Philip	Bluetooth Protocol	Philip is also responsible for the bluetooth protocol and also responsible for the programming of the microcontroller.	20
Isaac	Python	Isaac is responsible for learning the python programming language and will be helping Philip in programming the microcontroller.	20

Figure 3 | Personnel Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be based on the projected effort required to perform the task correctly and not just "X" hours per week for the number of weeks that the task is active

4.4 Other Resource Requirements

There are two types of other resources that we require:

First, there are some software tools that we need to implement our design like Android Studio and Xcode. We also might need some other software tools in the future to create the hardware from scratch like the PCB layout.

Second, there are some parts that we need for our product. We need a device that can communicate with the car like the OBD Scanner. We also need a microcontroller that has bluetooth capabilities to communicate with the user's phone. Finally, we need the electric parts to combine the microcontroller and OBD scanner into a small device that can be used in the car.

4.5 Financial Requirements

Our client is providing a $500-$1000 budget. This exceeds our estimated financial needs. We anticipate two hardware expenses. The first is our raspberry pi. The other is an OBD scanner. We estimate these to cost less than $100.

5. Testing and Implementation

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)
2. Define the individual items to be tested
3. Define, design, and develop the actual test cases
4. Determine the anticipated test results for each test case 5. Perform the actual tests
6. Evaluate the actual test results
7. Make the necessary changes to the product being tested 8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

– Discuss any hardware/software interfacing that you are working on for testing your project

In this project, the connection between software and hardware is achieved using Bluetooth Low Energy (BLE). Testing for the interface covers the initial establishment of the connection, the stability of the connection, autoconnect, transmitting and receiving format, and the completeness of the information being transferred. All of the above uses unit and integration testing. Due to the inability to have in-person meetings because of COVID-19, we have had to change the way we test hardware to software interfacing. However, we are still able to perform tests by having the software team send the software installing packages to the hardware team and provide them with instructions for how to install and test it. Once the hardware team has performed the necessary tests and checks, they will then give feedback to the software team for next iteration improvement.

5.2 Hardware and software

– Indicate any hardware and/or software used in the testing phase

– Provide brief, simple introductions for each to explain the usefulness of each

iOS unit testing will be performed on Xcode using the built-in testing framework XCTest. This framework allows programmers to make placeholders for Systems Under Test and assess their functionalities by comparing the actual output with expected output [1]. Tests can be run either in bulk or individually.

Android testing will be performed in Android Studio, using initially the Mockito library and then later on using the Android Studio emulator. The Mockito library will be used to run behavioral tests on smaller units of the app. The Android Studio emulator will be used to test the integration of those smaller units. Finally when the interface with hardware is being tested, a physical android phone will be used to test the functionality of the app.

The two primary types of hardware tests that we will be performing are for the OBD-II reader and for the bluetooth connection between the Raspberry Pi and the user's phone. Both tests will be performed primarily using Rasbian with the Raspberry Pi. Rasbian is extremely useful because it makes coding on the Raspberry Pi very easy and straightforward. We are able to program both our bluetooth and OBD-II hardware with it, which is also why we will be using it to conduct all of our tests.

For the OBD-II tests, we will be using the PiCan2 bus which is able to connect to the OBD-II port on the car using an OBD-II to DB9 cable. The PiCan2 bus is also able to connect to the Raspberry Pi using the in/out pins on top of the board. It is specifically made to be used with a Raspberry Pi, which makes it fairly easy to use. It also translates the data from the OBD-II output into readable data that is able to be used with Rasbian.

For our hardware bluetooth testing, we will be using Raspberry Pi and Raspbian to connect to an Android and iPhone. For Android we will be using Android studios. Our ability to program bluetooth compatibility gives us better feedback when testing bluetooth operations. iOS testing requires a basic version of our future iOS application. This method reports failures, but won't report where the error occurred.

5.3 Functional Testing

Software Functional Testing:

1. Unit Testing: Testing individual functionalities with complete statement coverage for every basic block and branch coverage for every outcome of a branch. (Ex. When a Child User navigates away from the app while driving, real-time change of signal is recorded in the database.)
2. Integration Testing: Testing the interface between modules are interconnected with each other. When adding a new feature, old features are not impacted in any way. (Ex. By adding app exit detection, the app can still be connected to our bluetooth device installed on the car.)
3. System Testing: Testing that all functionalities do not interfere with one another. (Follow Section 1.4 Software Functional-requirement, test that all functionalities co-exist and do not interfere with each other.)
4. Acceptance Testing: Release Beta testing and let testers operate the device without developer interference. This testing is in place to ensure the compliance between the testing version and the initial customer needs.

Hardware Functional Testing:

1. Unit Testing: Testing individual functions, such as the connection between the hardware device and the user's phone through bluetooth or if a software application is able to detect the hardware device when the vehicle is started.
2. Integration Testing: Testing multiple functions together, for example, whether the hardware device will be able to send the OBD-II data through bluetooth to the software application. Also, making sure that the hardware and software interfaces are able to communicate through a bluetooth protocol.
3. System Testing: Full functionality testing, which will be testing if the driver of the car is able to use their phone while the car is running.
4. User Acceptance Testing: Testing whether the hardware device is able to meet the client's requirements. In this scenario, that is being able to work with an iPhone and 2015 Ford Escape.

5.4 Non-Functional Testing

Software Non-Functional Testing:

1. Security Testing: Testing that the iOS and Android applications are secure and will not be able to be hacked into by the teenager user.
2. Performance: The real-time data transmitting should have a delay of no longer than 0.5 second for practicality.
3. Compatibility Testing: Testing that the application can work with all models of iPhone and Android phones. This can also be done through Beta testing.
4. Usability Testing: Test the App's UI to make sure they are user friendly and easy to navigate for both adults and teenagers.

Hardware Non-Functional Testing:

1. Security Testing: Testing that the hardware device is not able to be hacked into or reset in any way. This testing is to ensure that the teenager is not able to get into the Raspberry Pi and change the program so they are able to not be detected while using their phone while driving.
2. Performance Testing: Testing to make sure that the hardware draws as little power from the car as possible to create as little stress on the battery as possible while also maintaining a high enough level to create optimal operating speed.
3. Compatibility Testing: Testing that the OBD-II reader is able to work with all types of vehicles.
4. Usability Testing: Making sure that the hardware we produce will be able to be installed easily by the client and possible future consumers of our product.

5.5 Process

– Explain how each method indicated in Section 2 was tested

– Flow diagram of the process if applicable (should be for most projects)

The method we came up in section 2 for hardware was pretty straight forward. As stated earlier, our main objective was to connect our hardware device to our iOS or Android application. Although we were not able to achieve that this semester, we made progress when it came to software and hardware development. The hardware tests we conducted included testing whether our OBD-II scanner was able to detect code from the vehicle's ECU. The OBD-II interface can be tested using an actual vehicle. One difficulty for testing is that the PiCAN2 only works with vehicles that use the CAN for their OBD-II, which was not standard until 2008. The next test was whether our Raspberry Pi was able to detect other bluetooth devices.

The framework XCTest and the library Mockito are being used to test various components of the apps on both platforms. After those tests are passed, the next phase is to use the emulator to test the integrations of the different components. Finally, testing is moved to actual physical hardware.

Figure 4 | Software Testing Process

5.6 Results

The project was going at a steady pace, and we were confident of our ability to deliver a working prototype by the end of the semester. The tasks for this semester are indicated in section 3.3, and Figure:2 shows the initial timeline for the tasks to be completed. The team intended to work on the project together after Spring Break (03/16 – 03/20) and join all our work to meet the deadline.

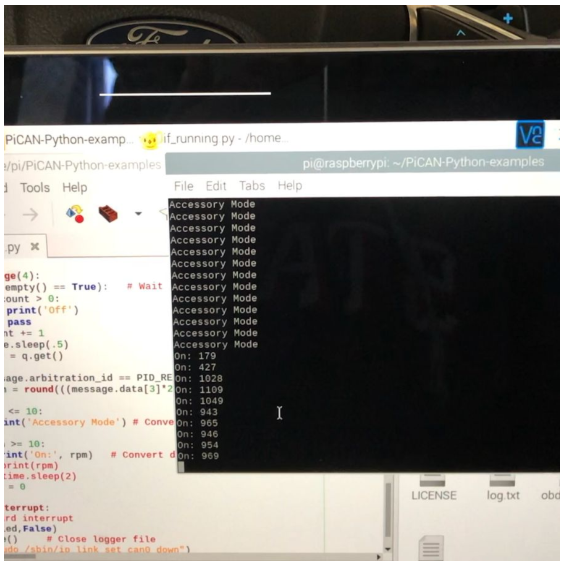
- By January 30th, news of the COVID-19 (Novel Coronavirus) broke-out all over the world, and the World Health Organization (W.H.O) declares a global health emergency [1]. Although our team was aware of the virus, we did not expect it to get worse. Thus, we continued our work at the same pace and confidence to finish the prototype by the end of the semester.
- February 29th. Level 3 travel health notice issued by the Centers for Disease Control and Prevention [2] and we started noticing more warnings, but again we did not expect the virus to affect our work.
- March 16th. Start of Spring Break.
- March 18th. All university courses moved to virtual instruction until May 9th [2]. At this point, the team recognized the severity of the virus, but we were optimistic and decided to meet virtually to discuss the project. We decided to take this time to improve our knowledge separately of the separate tasks assigned individually.
- March 23ed. We created a contingency plan to combat the virus. In the contingency plan, we expressed our confidence in our timeline and expressed our ability to submit a working demo of the project because we thought that we were going to get back to normal and will be able to meet and use the university resources. But unfortunately, everything got locked down and we were instructed to NOT meet in-person.

Not being able to meet in-person, presented us with a great challenge. Our project consists of two components; hardware and software, and the project creation process is interconnected. Hence, our progress started to slow down due to the relicense of one task to the one before. In addition, we had a limited number of hardware components and the team members were in their respective homes/states, thus sharing of components stopped completely. Furthermore, to finalize the software side of the project, the hardware must be functioning properly, so we were not able to complete the software part as well.

Although the challenge was great, we were able to finish a substantial part of the project and we were able to have a separate working task. We only need to connect the various parts together now and refine the project as needed. After we connect the parts, we will be able to finish the tasks that require advanced work. Figure 5, shows the modified timeline and the progress with each task (Section 3.3).

Figure 5 | Gantt Chart of Current Progress

Task 1: completely finished and working (see Figure 6). Initially, we had bought an OBD-II to USB scanner that was supposed to translate all of the data coming from the OBD-II output on the car into a readable format to the Raspberry Pi. However, after a long time of testing, we decided that this option was not feasible and decided to use a different approach. This approach required the use of the PiCan2 bus, which is able to translate the code output from the OBD-II scanner and translate it into code the Raspberry Pi is able to read. By using this approach, we were able to read data from the car and write a code that determines when the car is on, off, or in standby mode.



```
ge(4):
empty() == True):   # Wait
ount > 0:
 print('Off')
 pass
nt += 1
e.sleep(.5)
 = q.get()

age.arbitration_id == PID_RE
 = round(((message.data[3]*2

 <= 10:
int('Accessory Mode') # Conve

 >= 10:
int('On:', rpm)   # Convert d
rint(rpm)
ime.sleep(2)
 = 0

terrupt:
rd interrupt
ed,False)
()     # Close logger file
do /sbin/ip link set can0 down")
```

pi@raspberrypi: ~/PiCAN-Python-examples

```
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
Accessory Mode
On: 179
On: 427
On: 1028
On: 1109
On: 1049
On: 943
On: 965
On: 946
On: 954
On: 969
```

Figure 6 | OBD-II Test Result

Task 2: 80% completed. We were able to use the Bluetooth feature in the Raspberry Pi and connect it to a phone (see Figure 7). But we were not able to finalize this task because to make it work completely, we must have many components at the same time (car, Raspberry Pi, Phone). Therefore, 20% of this task is left and we will be able to finish it whenever the team gets together, and the components are available at the same time.

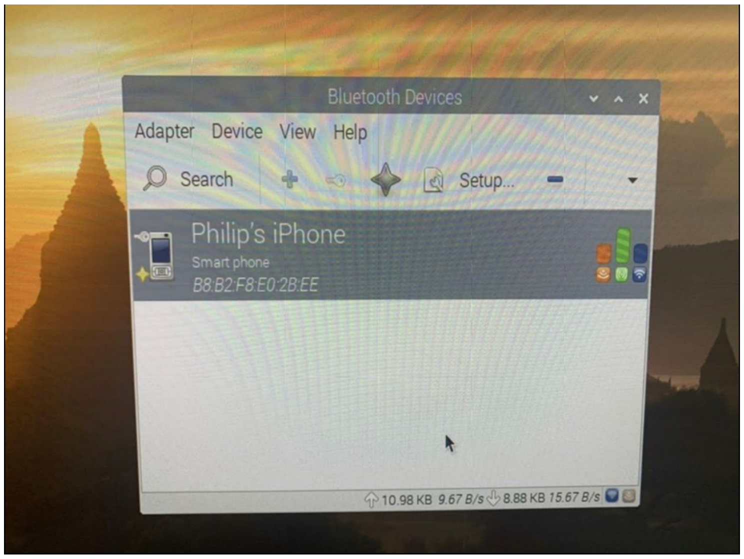


Figure 7 | Bluetooth Test Result

Task 3: 50% completed. We accomplished the first half of this task by accomplishing most of the first two tasks. The other half requires finalizing Task 2 and creating a unique ID for the Raspberry Pi so the phone will know the source of the signal it is receiving. After that we need to make sure the connection between the car, PiCan2, Raspberry Pi, and the phone is working properly, then testing the phone application to get the needed results.
Task 4: 20% completed. We learned about this task and made a thorough research about Bluetooth protocols and signals. The remaining 80% is about access blocking features that will be implemented after bluetooth connection has been established in the previous task.

6. Closing Material

6.1 Conclusion

Our goal was to create a system that reads a vehicle's gear position, encodes the position, sends the code via bluetooth to a device, and the device accessibility is limited based on the code. Currently, we have an OBD-II scanner that interprets the car's RPM into gear position. The Raspberry Pi has a bluetooth package that makes it discoverable, sends and receives signals, and completes other bluetooth tasks. Two apps, android and iOS are also in the works. While each of these components have been tested, implementing them together is our next obstacle.

6.2 References

[1] M. Katz and M. Katz, "iOS Unit Testing and UI Testing Tutorial," raywenderlich.com, 08-Apr-2019. [Online]. Available: https://www.raywenderlich.com/960290-ios-unit-testing-and-ui-testing-tutorial. [Accessed: 23-Apr-2020].

[2]A. S. Huang and L. Rudolph, in *Bluetooth essentials for programmers*, New York, NY: Cambridge University Press, 2007.

6.3 Appendices

Figure 8 shows our rough sketch of what our phone application is going to look like. It was drawn during the beginning of our design process. Next semester, we plan on focusing more on refining this design and testing the functionality of it.

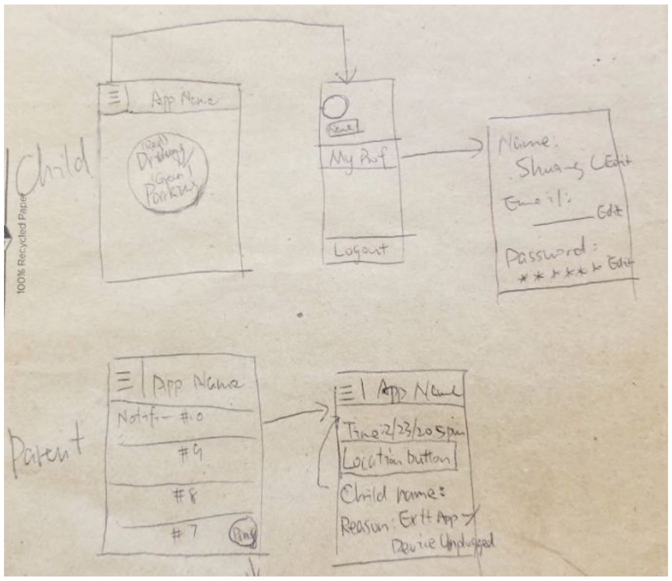


Figure 8 | Rough Sketch of Phone Application Layout Drawn During Team Meeting