Driving Management System (DMS)

Group 26

Department of Electrical Engineering & Computer Science
University of Central Florida
Dr. Samuel Richie
Senior Design II–Spring 2014
April 28, 2014

Aaron Kost
Victor Medina
Sarah Bokunic

Table of Contents

1.0Executive Summary

The goal of this project is to provide driver's with access to features that are not available on most base model vehicles. The Driving Management System (DMS) provides drivers with fuel efficiency analysis, a blind spot detection system, a front end collision sensor and a rearview camera. All of these features are meant to enhance the safety of the driver. All of the features that the DMS provides can be added to any vehicle for a large price. The DMS looks to provide driver's with a cheap and efficient alternative to expensive aftermarket products. Most importantly each feature of the DMS will be tied into an easy to use application on the driver's android phone. Smartphone applications are currently very popular. Many car manufacturers are implementing smart phone applications within their vehicles. The DMS looks to combine electronic hardware that may be offered as additional features on newer vehicles and combine them with the user's android device. This allows the user to have full control of certain features in their vehicle just by using their smartphone or tablet. The DMS is based off the OpenXC API designed by Ford. This allows the project to obtain information that is not necessarily available to the driver. The goal of Ford's OpenXC is to allow drivers and developers to make their own aftermarket features for Ford vehicles. The DMS will use this to implement aftermarket features for vehicles and combine it with an Android application.

The driver will be provided messages through their phone on how they can improve their fuel efficiency based on driving behavior. This will save the driver money by getting rid of bad habits that may be causing their vehicles fuel efficiency to drop. This also helps to protect the driver from accidents as many dangerous driving behaviors also decrease fuel efficiency. Insurance companies such as Progressive have been using technology similar to this to provide drivers with discounts and feedback on their driving behavior. Insurance companies can potentially use the DMS as another method to monitor driver's behaviors and to offer discounts to safe drivers. This could decrease the amount of accidents seen on the road by promoting safer driving habits. The DMS also provides additional safety features that communicate with the driver's android device. The blind spot detection system alerts drivers when there is another vehicle in their blind spot. This allows the driver to take appropriate action if they are changing lanes and may reduce the chance of an accident. Rear ending vehicles or other objects can lead to expensive repair costs. For this reason the front end collision system was implemented to help avoid these accidents. The DMS is also very simple to use as there is no need for a complicated installation. Each component of the DMS can be placed on the vehicle. This allows for anyone to be able to fit the DMS to their vehicle themselves. The ability of the driver to know whether or not a car is located in their blind spot or to have vision of what is directly behind their car when backing up greatly reduces the chances of an accident. This project is designed for the 2013 Ford Focus. The Vehicle Interface used is programmed to use information that is specific to the Ford Focus. This project can be used as a

means on how to improve driver's fuel efficiency and safety. It can also be seen as a starting point on how to implement a similar device on other vehicle models.

2.0 Project Description

DMS will assist drivers in driving more effectively and safely by providing expensive vehicle features for a fraction of the cost. DMS will use the driver's smartphone to provide fuel efficiency management, blind spot detection, rear view camera vision, and frontal collision detection. Each feature will consist of a self-powered stand-alone hardware module that communicates with the phone wirelessly. Using the OpenXC vehicle interface that plugs into the vehicles OBDII port, the phone will receive data from the vehicle. The received data will include the needed vehicle information to determine fuel efficiency, lane changing, and reversing. Using vehicle information from the car and signals received from the hardware peripherals DMS will assist the driver with switching lanes, driving with better fuel efficiency, and reversing safely. By having all the hardware peripherals be self-contained and powered, with wireless communications, no modifications to the vehicle will be necessary, allowing drivers with no technical knowledge to obtain expensive and advanced features post-manufacture that will improve driving habits and assist in reducing accidents.The vehicle interface is a hardware device that plugs into the vehicles diagnostic port and passively listens for a subset of CAN messages, performs required unit conversion, and makes the converted data accessible using the OpenXC android library. The host device, or smartphone, connects to the vehicle interface and uses the OpenXC android library to access the required data.

The smartphone acts as the host device, making it the driving force behind DMS. The smartphone is responsible for storing data, relaying information to the driver, and determining how DMS should respond to varying situations. The smartphone will be wirelessly receiving vehicle information from the vehicle interface via the OpenXC library, as well as receiving indication signals and data from the various hardware modules installed on the vehicle. The host device for DMS must be an android phone running android 3.0 or greater. The blind spot detection feature for DMS will consist of two standalone hardware modules that are battery powered and communicate with the host device wirelessly. The hardware will include a low power-consumption microcontroller that is Bluetooth enabled, and a low range sensor to detect objects that the vehicle may collide with when switching lanes.

The reverse assist will activate when the vehicles gear position is set to reverse. A webcam placed on the rear of the vehicle will stream video to the host device as well as provide an onscreen collision detection overlay to help driver with positioning of the vehicle.

The fuel efficiency portion of the app will be active when the car is in drive. The smartphone will receive information pertaining to fuel consumption and driving habits that affect fuel consumption while the vehicle is in motion. The fuel efficiency monitor will provide the driver with real-time feedback on their fuel consumption, and when the vehicle is stopped at a light, it will provide them with suggestions based on their driving to improve their future fuel consumption. The frontal collision detection will consist of a low power-consumption microcontroller with a narrow wave sensor. The frontal collision detector will be responsible for helping the driver park, and ensure the front of their vehicle does not collide with parking blocks and other objects out of the driver's vision.

2.1 Project Motivation

The motivation for this project stems for the need to have better gas mileage and safer driving on the roads. People who have purchased new standard package vehicles such as the 2013 Ford Focus pay anywhere from $15,000 to $21,000 dollars and most do not come with advanced features such as a backup camera, blind spot detection, or advanced fuel efficiency assistance. The 2013 Ford focus currently has a very primitive feedback system for fuel efficiency. If the vehicle is able to communicate with a drivers smartphone, a more accurate fuel efficiency monitor should be available that gives the driver detailed reports, suggestions, and real-time feedback in an easy to interpret manner.

The package on the 2013 Ford Focus that this project will be using does not come with blind spot detection because it was too expensive. If a hardware module that communicates to the smartphone wirelessly can be developed for a low cost, then the expensive feature of blind spot detection can be added using the smartphone for a much smaller price presenting money tight customers with access to luxury safety features.

There are a lot of products that offer these features independent of each other, but if the smartphone is used as a central host device, a number of hardware peripherals can be added allowing people to add many more post-manufacture features without the clutter of a bunch of standalone hardware devices.

2.2 Objectives

The projects main objectives can be categorized as follows:

1) Vehicle Interface
2) Blind spot sensing and Collision detecting circuits
3) Fuel efficiency monitoring
4) Reverse Assistance

2.2.1 Vehicle Interface Objectives

The vehicle interface is the keystone of the project, that is, for this project to work the vehicle interface must be 100% functional. For this reason the project will first use a prebuilt vehicle interface from Ford, and if time permits we hope to develop our own vehicle interface. Therefore the following objectives for the vehicle interface include:

1. Loading correct firmware for 2013 Ford Focus
2. Setting up Eclipse with OpenXC libraries
3. Setting up android enabler and running initial diagnostic test
4. Setting up vehicle emulator for testing
5. Determining if there is time to design our own project specific vehicle interface.
    a. Selecting a compatible microprocessor.
    b. Selecting appropriate diagnostic pins required for project.
    c. Selecting and integrating compatible Bluetooth.
    d. Ensuring microprocessor always receives correct voltage from diagnostic port.
    e. Passing 12v power from vehicle to separate power jack.

2.2.2 Blind spot sensing and Collision Detection Circuits

The Blind Spot sensing and collision detecting circuits are important because they represents a major portion of the features this project will offer. It is important that the circuits communicate wirelessly and are battery powered so that it meets the project criteria of being effortless to install. Therefore the following objectives for the Blind Spot sensing circuit and Collision detecting circuit include:

1. Selecting a microprocessor that has a low power mode, can be battery powered, and can handle the computations required for the blind spot detection or collision detection.
2. Selecting a sensor type and a sensor range.
3. Selecting the type of wireless connection and the wireless connection hardware
4. Selecting the battery that will power the circuit.
5. Ensuring that the battery can be recharged using the vehicles power outlets if the battery needs to be recharged.
6. Creating an effective way to easily mount on vehicle

2.2.3 Fuel Efficiency Monitoring

The fuel efficiency monitoring portion of this app utilizes the vehicle interface more than any other feature. The fuel efficiency monitoring portion of the app has

to be safe for drivers to view, user friendly so they don't have to touch it while driving, and provide accurate information and suggestions. The objectives for the fuel efficiency monitoring include:

1. Using vehicle data to accurately determine the driver's fuel efficiency.
2. Creating an accessible database to store driving data that pertains to fuel efficiency.
3. Using stored and current driving data to make suggestions on improving fuel efficiency.
4. Providing graphical breakdown of driving information to display bad driving habits in an easy to interpret manner.

2.2.4 Reverse Assistance

Reverse assistance feature includes a rear-facing camera with a graphical overlay to show the driver where it is safe to reverse. The reverse assistance feature will use the vehicle interface to tell when the care in in the reverse gear, and activate the camera feed. The objectives for the reverse assistance include:

1. Connecting the camera to the android host device, so that the host device receives live camera feed from behind the vehicle. This includes creating a java library that can handle connecting camera and receiving video frames.
2. Creating a graphical overlay that responds to vehicle wheel position.
3. Mounting the camera on the rear of the vehicle with necessary wiring.

2.3 Project Requirements and Specifications

The most important requirement of the DMS is that it does not interfere with the driver. The alerts sent to the driver must be by voice so that the driver can maintain vision on the road. Since a phone or tablet is being used to relay information back to the driver, it must be positioned to prevent it from interfering with the driver's attention on the road. Ideally the project would be integrated into the dashboard of the vehicle, but there is currently no access to altering the information displayed on the dashboard. The DMS must also be small and easy to use. Due to the fact that each component of the DMS will be communicating wirelessly this removes the hassle of a complicated installation onto ones car. It is also important that the DMS can be powered for a long amount of time. This makes it convenient for the driver since constant recharging is not needed.The following list contains specifications for each part of the DMS.

Fuel Efficiency

- Displays a graphic on screen that shows drivers how well they are currently driving.
- Alerts driver during a stop, providing the driver with tips of what they can do to improve fuel efficiency if the driver is driving poorly.
- Keeps long term data to track drivers progress over a period of time.

Blind Spot Detection

- Must be able to detect objects within 3 meters of the driver's blind spot.
- Alerts users through their phone or tablet of which side has an object within the driver's blind spot.
- Only alerts driver while turn signal is active and/or steering wheel is being turned a specific distance.

Rearview Camera

- Sends a clear video feed from behind the car to the driver's phone or tablet.
- Assists drivers while parking in reverse.

Collision Detection

- Alert driver when to begin braking to prevent a head on collision.
- Uses the speed of the vehicle and distance of the object in front to determine when to brake.

2.3.1 Hardware Requirements and Specifications

The following requirements and specifications are to be used as a guide when designing the hardware portion of the project. The requirements are meant to be achievable and as the project progresses may be subject to change.

- The microcontroller used must have low power modes to save battery life.
- The wireless communication module must be compatible with an android device.
- The sensors used must be able to function in multiple weather conditions such as rain, high and low temperatures, and foggy conditions.
- The protective casing for the sensors must not fall off of the vehicle during use.
- All of the components for the project must be able to communicate wirelessly to an android device, excluding the camera.
- The peripheral placed on the outside of the car must be small enough to not detract from the overall appearance of the car.

Providing a long battery life is a very important part of the project. A longer battery life will prevent the driver from having to constantly replace the batteries of each component within the project. The battery must also last for long periods

of time in between recharges. This will eliminate the hassle of the driver to constantly recharge the batteries in between drives.

The main device of the DMS will be an Android device. This will be the core of the project. It should be able to maintain a way to communicate with each device attached throughout the vehicle. The device can either be a tablet or a smartphone that the driver owns. Ideally the information provided by the DMS would be available through the Ford Focus' dashboard. This would allow the DMS to be an application designed specifically for the vehicle. The Android device should be able to operate while being charged in the driver's vehicle. It must also be able to save data so the driver can refer to it while not using the vehicle.

2.3.2 Software Requirements and Specifications

The following software requirements and specifications will ensure that the project meets the defined goals. They may be subject to change, as with the hardware requirements and specifications.

- The software must be compatible with an android device.
- The program must be able to take in and process data in real-time.
- The program must be able to store data for future use.
- The program must be able to adjust for different visual requirements for people who are not able to see certain colors.
- The program must automatically switch between the fuel efficiency monitoring display and the rearview camera display without the driver interacting with the phone directly.
- The program must have graphs to display fuel efficiency information to the user.
- The graphs must either display all necessary information, or allow the user to scroll to view all necessary information.
- The program must have options to allow the user to customize the program to fit their needs.
- The program must display tips to the user that are specific to the user's driving habits, rather than just displaying generic tips.

3.0 Research

3.1 Existing Products/Projects

The following products are post-manufacture hardware additions for personal vehicles. The idea of having the vehicle and hardware communicate information to an android app is a fairly new concept so there is a limited amount of products that fulfill this particular niche. The most complete product commercially available is automatic link. Automatic link plugs into the vehicles OBD-II port and communicates with the android host device via Bluetooth. Automatic link

provides driving efficiency tips based on your speed and driving habits, calls help in case of a crash, and helps perform engine diagnostics. Automatic link does not provide reverse assistance or blind spot and collision detection. Automatic link is compatible with android and iOS, and DMS will only be compatible for android. Although Automatic link provides fuel efficiency feedback, it will not be as accurate as DMS's because they do not have access to all of the specific vehicle information that OpenXC has. Although there are not many products that encompass everything DMS will do, there are a lot of standalone hardware add-ons that do only one of the specific features.

3.1.1 Blind Spot Product

GOSHERS blind spot detection system is $249.99 and requires an extensive installation. To install the blind spot detection holes have to be cut in the vehicles bumper for the sensor placement. The hardware box has to be installed in the vehicle and the LED indicators must be wired through the interior and placed on the right and left sides of the vehicle. GOSHERS blind spot detection differs from DMS in that DMS blind spot detection will alert you with an audible sound, and a heads up message on the host device if the driver goes to turn an object in in the way versus using LED indicators.

3.1.2 Fuel Efficiency Monitoring Products

Some vehicles now come with an economy driving mode that can be activated. The 2013 Honda Civic has Eco Assist™, which is activated when you press the ECON button. On the dash of the vehicle there is a strip of LED lights on either side of the speedometer that inform you of how well you are driving for fuel efficiency by changing color. The Eco Assist™ is much more effective than the DMS fuel efficiency monitor because not only does it monitor and inform you of your fuel efficiency it also modifies how the vehicle runs by slowing down the A/C and making the vehicles acceleration much slower. Eco Assist™ isa core feature of the vehicle from the manufacturer so it has the ability to control factors that affect fuel efficiency that post-manufacture additions cannot control.

3.1.3 Rearview Camera Products

There is a wide selection of rear view camera products on the market in a very wide price range ($100-$500). All of the products available however require the installation of an LCD screen on the dash, or on the rearview mirror. On the lower end of the price range spectrum there are rearview camera products such as the Safesight SC0302. The Safesight SC0302 consists of a camera that connects to an LCD screen through component cables. The Safesight uses the brake lights to determine if the car is in reverse. The benefit that the reverse feature of DMS has over this is that DMS does not need an installed LCD screen because it uses

the smartphone, the camera also does not need to sense when the brake lights are on to activate because it the smartphone will know when the vehicle is in reverse from the vehicle interface.

3.1.4 Existing OpenXC Projects

Ford has provided drivers with OpenXC, which is an API to Ford vehicles. This has allowed the Ford community to produce their own aftermarket hardware and software for vehicles. Since its release many hobbyist have done projects using information from their Ford vehicle that only can be obtained using OpenXC. The following projects are just some of the aftermarket hardware and software devices that have been designed for Ford vehicles using OpenXC.

Haptic Feedback Shift Knob -The Haptic Feedback Shift Knob is a replacement for the traditional shift knob in a Ford vehicle. The Haptic Feedback Shift Knob alerts the driver when the optimal time to shift is by vibrating. The goal of this device is to allow driver's to pay more attention to the road, while being able to shift for best performance. It uses an Android application which monitors the vehicles speed, RPM and accelerator pedal position. The Android device then sends the signal that causes the Haptic Feedback Shift Knob to vibrate via USB. There is also a seven segment display at the top of the Haptic Feedback Shift Knob which displays the current gear position.

Rearview Camera -The Rearview Camera is an Android app that provides drivers with a way to see what is behind their car. The app was designed by an intern at Ford and is intended to be used on vehicles that do not come equipped with a factory rear view camera. A simple USB webcam was used as the camera and securely placed onto the rear of the vehicle. The Android application monitors the vehicles gear position and steering wheel angle. When the vehicles gear position is placed in reverse the camera feed will be displayed to the driver. On the video displayed to the driver is an outline of the vehicles wheels. The steering wheel angle is used to determine where the path of the car will be if the driver continues to move in the same direction.

3.2 Fuel Efficiency

3.2.1 Driving Habits

The choices a driver makes behind the wheel has a direct effect how much gas the vehicle is using. These choices can range from accelerating to fast all the way to choosing to keep heavy items in the trunk. DMS aims to help the driver become more aware of the habits he/she display behind the wheel that effect fuel efficiency. If the driver becomes more aware of the bad choices he/she is making behind the wheel and how those choices effect the amount of money spent yearly on gas, then behavior choices may be made to increase their cars fuel economy. Below is a table of habits that adversely affect a vehicles fuel consumption.

- Rapidly increasing acceleration and braking frequently can lower gas mileage between five percent and 33 percent depending on the speed of the vehicle, which is equivalent to wasting about eighteen cents to one dollar and nineteen cents per gallon.
- Although it varies depending on the vehicle, gas mileage starts decreasing at a faster rate at speeds about 50 miles per hour. This can lower gas mileage between seven and fourteen percent, which is about 25 cents to 51 cents wasted per gallon.
- Having excessive weight in the vehicle can also lower gas mileage by about one to two percent per 100 pounds. This is equivalent to about four to seven cents wasted per gallon. [1]
- Idling for extended periods of time decreases gas mileage by about one to three cents per minute if the air condition is off, over two to four cents per minute if the AC is on. If the driver has to wait longer than 1 minute, it is better to just turn off the car while waiting instead of idling. [2]
- Having the air conditioner on can reduce gas mileage from between five and 25%. [1]
- In stick shift vehicles, fuel is wasted if a person is driving in a gear that is lower than the necessary gear, so the driver should switch gears as quickly as possible.
- If the gas cap isn't completely tightened after refilling gas, gas is able to evaporate and some of it is wasted depending on how long the gas cap is in this state.
- Having the windows rolled down increases drag which can hold back the vehicle and waste gas. [3]
- Coasting toward stoplights rather than braking reduces the amount of fuel wasted because braking wastes fuel.
- Constantly changing speed can increase the amount of fuel that is wasted, so using cruise control can improve fuel economy by at most 14% which is about 43 cents per gallon. [4]
- Revving the engine before turning it off reduces fuel economy. [2]

3.2.2 2013 Ford Focus Information

The vehicle being used in this project is a 2013 Ford Focus, 4-door sedan SE. The vehicle has a 2.0L I4 GDI engine with a 6-speed PowerShift automatic transmission. The PowerShift transmission is essentially a manual transmission that is shifted automatically for the driver using a special software operated clutch. The EPA estimated fuel economy for this specific vehicle is listed as 33 MPG combined city/highway with 28 MPG city and 40 MPG highway, with an estimated average fuel cost of $1,600 a year. The EPA estimated fuel economy is just an estimation and actual fuel consumption may vary depending on driving habits and factors such as temperature and terrain. The miles per gallon in the city is significantly less than on the highway mostly due to stop-and-go traffic and constant idling at stoplights. [2] The vehicle has a built in feature which monitors the tire pressure and alerts the driver if the tire pressure is at an unsafe level,

which also reduces fuel from being wasted as a result of driving with tires that are not properly inflated. [5] The goal of DMS is to help the driver get as close as possible to the EPA estimated fuel economy by helping the driver operate the vehicle more effectively.

3.2.2.1 Current Ford Focus Driving Aids

Every 2013 Ford Focus comes equipped with an ECO MODE driving aid. The ECO MODE driving aid is a system that assists the driver in driving more efficiently by monitoring characteristics of gear changing, how well the driver anticipates traffic controls (sudden breaking ext), and change in speeds while driving. The driver can view how well they are doing by viewing a flower like display on the dash that consists of five petals. The petals are filled in as the driver performs various actions with better efficiency, and made empty as the driver performs with less efficiency. That is, if the driver is getting the best possible fuel consumption, and driving efficiently all five petals will be filled. The more effectively you drive, the better the rating, and the better your overall fuel economy becomes. With the Ford ECO MODE the efficiency values that rate the driver do not directly result in a fuel consumption figure, that is, the pedal does not directly relate to the drivers fuel consumption, but is based on the idea that if the driver is driving effectively they will have a on average better fuel consumption rate, but does not necessarily mean that if they have all five petals full they will be getting the best possible fuel consumption rate because there are influencing outside factors such as cold weather, hilly terrain, long idles, or short trips. The ECO MODE criteria of gear position is important to fuel economy, however, in the model being used for this project gear position is outside of the drivers control because the transmission is an automatic power shift transmission that is controlled by software, meaning that the gear positions are preset.

Although the 2013 Ford Focus comes equipped with a way to help the driver drive more efficiently, it is not a very dynamic system, and it does not provide extensive feedback. While using the ECO MODE, it runs off current trip, so every time a driver wants to use it, the driver has to reset the current trip. After logging many miles, the ECO MODE tends to become “set” and no longer changes if the driver follows the same driving habits. The DMS aims to provide a better more dynamic feedback by offering live feedback based on in the moment fuel consumption, and driving efficiency. DMS will also provide feedback suggestions that actually help the driver drive more efficiently by providing suggestions based on the driver’s habits.

3.2.3 Outside Factors

There are factors that may not be directly caused by the driver which can affect fuel efficiency. These factors need to be taken into account when determining how much fuel is being used compared to the expected miles per gallon determined by the manufacturer of the vehicle.

- The terrain that is being driven on affects gas mileage. For example, driving on hilly terrain or unpaved roads can decrease fuel efficiency. When cars are being tested, they are expected to be on flat roads.
- Cold weather also reduces fuel efficiency because the engine needs to be warmed up to work effectively. It is recommended to not take many short trips in cold weather because it does not allow the vehicle to operate at the necessary temperature. [1]
- If the vehicle's tires are not properly inflated, fuel consumption can increase by at most 6%. [3]
- If the spark plugs in the vehicle are bad, fuel economy can decrease by 30% which costs about 94 cents per gallon.
- If the tires of the vehicle are not aligned properly, fuel efficiency can be decreased by 10% due to the tires dragging rather than rolling freely. It also wears out the tires and can result in an even greater decrease in fuel efficiency. [4]
- Vehicles with automatic transmission cannot be controlled by the driver to get better fuel consumption.
- Idling at stoplights, road intersections, or a type of lawful road stop lowers fuel efficiency but is outside of the driver's control.

There are also a number of factors outside of driving habits that influence fuel economy that can be controlled by the driver, but cannot be monitored or prevented by DMS. These factors would include things such as not using the recommended engine oil for the specific vehicle and not performing all regularly scheduled maintenance.

3.3 Microcontrollers

3.3.1 Overview

The microcontroller that will be used in the DMS will need to meet a certain number of requirements. The project will require the use of up to four microcontrollers due to the fact that each circuit will be separate from one another. This will maintain the wireless aspect of the DMS integration onto the car. Therefore the microcontroller must be very cheap since the project will require up to four. Very little power must also be consumed by the microcontroller. A low power consumption microcontroller will save battery life while it is not in use. The microcontroller must also be compatible to the addition of a Wi-Fi or Bluetooth module for wireless communications. This will allow the microcontroller to relay information to the driver's phone or tablet located inside the car. Each circuit will be communicating with the phone or tablet separately.

3.3.2 MSP430G2553

The MSP430G2553 is part of an Ultra-Low power 16-bit microcontroller family. It is manufactured by Texas Instruments and uses the MSP430 Launchpad kit as the development environment. The development environment includes free access to Texas Instruments Code Composer Studio which is a tool used to program the microcontroller. The MSP430G2553 runs off a 16 MHz clock and uses a low supply-voltage range from 1.8 V to 3.6 V. The supply-voltage range of the MSP430G2553 is not of a typical value. Most batteries provide a voltage of at least 5 V. For this reason the MSP430G2553 will require the use of voltage regulators. The voltage regulators will also help to keep a constant supply voltage to prevent the microcontroller from being damaged. It has one active mode and five selectable low-power modes. This feature allows the user to enter a low power mode during periods of inactivity, and switch to an active mode without an impact on performance. This is ideal for meeting the low power consumption requirement. The MSP430G2553 also has a UART connection which is compatible with a Bluetooth module. This will allow us to send data from each circuit attached to the vehicle wirelessly. Combining this with the low power modes allows us to only send one signal at a time when it is appropriate. Lastly, the MSP430 microcontroller's family is a very inexpensive option. This is important due to the project requiring multiple microcontrollers. Table 1 and Figure 1 contain information on the MSP430G2553 which is relevant to the project.

MSP430G2553	
Operating Voltage	1.8 -3.6V
Max I/O Pins	10
Flash Memory	16 KB
Clock Speed	16 MHz
SRAM	0.5 KB
EEPROM	256 bytes
Max DC Current per I/O Pin	6 mA
Max DC Current for Vcc and Gnd	420 µA

Table 1: Table of MSP430G2553 specifications.

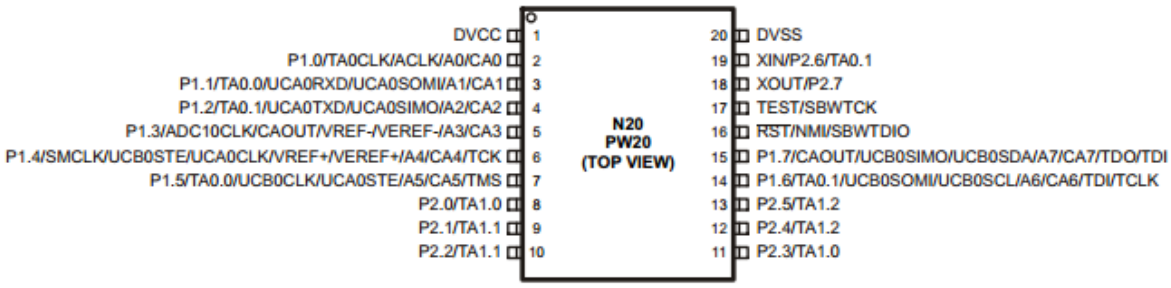

Figure 1: Pin layout of the MSP430G2553, permission provided by Texas Instruments.

3.3.3 ATmega328P

The ATmega328P is a high performance, low power microcontroller designed by Atmel. It is based off an 8-bit AVR RISC architecture which is capable of executing 20 MIPS at 20 MHz. This microcontroller provides up to 32 Kbytes of flash memory, 23 general purpose I/O pins and 1 UART port. The operating voltage of the ATmega328P ranges from 1.8V to 5.5V. Similar to the MSP430 the ATmega328P provides low power alternatives while maintaining a high level of performance. This is done by using a power save mode which allows the microcontroller to sleep until it is called upon. There are six sleep modes that are provided by Atmel, all that allow for different modes of functionality. The ATmega328P can remain in a sleep mode until an interrupt is sent to switch the microcontroller into an active state. The UART port will also allow the ATmega328P to receive and transmit data, which allows for a Bluetooth module to be added on to enable wireless communication. The following table contains specifications of the Atmega328P which are relevant to the project.

ATmega328P	
Operating Voltage	1.8 -5.5V
Max I/O Pins	23
Flash Memory	32 KB
Clock Speed	20 MHz
SRAM	2 KB
EEPROM	1 KB
DC Current per I/O Pin	40 mA
DC Current for Vcc and Gnd	200 mA

Table 2: Table of ATmega328P specifications. [6]

3.3.4 CC2541

The CC2541 is a microchip design by Texas Instruments specifically for low power Bluetooth applications. It combines a RF transceiver and an 8051 microcontroller into one module. The CC2541 can last from several months up to several years on a single coin-cell battery. Also included within the CC2541 are 2 USARTs, 23 general purpose IO pins, and a built in battery monitor. This chip provides all the necessary features to implement both the blind spot detection and collision detection features of the project, unfortunately the price of the CC2541 is a major trade off for the convenience. The development kit for this chip is $299.99 from Texas Instruments. This would defeat the purpose of providing a cheap alternative of safety features for drivers. Table 3 and Figure 2 contain information on the CC2541 which is relevant to the project.

CC2541	
Operating Voltage	2.0 -3.6V
Max I/O Pins	23
Flash Memory	128 KB
Clock Speed	32 MHz (High performance 8051)
SRAM	8 KB
EEPROM	1 KB
Transmission Current Draw	Tx - 18.2 mA
Receiving Current Draw	Rx - 17.9 mA

Table 3: Table of CC2541 specifications. [7]

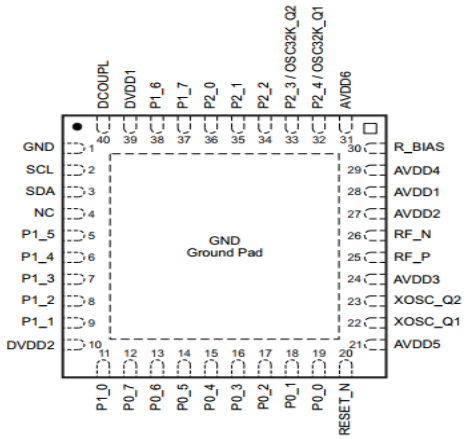

Figure 2: Pin layout of the CC2541, permission provided by Texas Instruments.

3.3.5 Summary

After comparing the three different types of microcontrollers that were researched, the group has decided to use the MSP430G2553. All of the group members have experience with the MSP430G2553 as it was used in a previous course. The MSP430G2553 also is much cheaper than the other options for microcontrollers. This helps keep the cost down due to requiring multiple microcontrollers to be placed in different places along the vehicle. Texas Instruments also provides many resources which makes the programming of the MSP430G2553 simple to learn and easy to use.

3.3.6 UART Communications

Most microcontrollers, including the MSP430, provide a UART channel which allows for wireless communication with the addition of a compatible Bluetooth or Wi-Fi module. UARTs typically use communication standards such as RS-232, RS-422 and RS-485. A UART connection does not require both devices to operate on the same clock cycle, but does require a specific set of configurations. The UART transmits a byte of data serially to another device one bit at a time. The receiving device then rebuilds the entire byte. The following list contains information on what specifications must be configured properly to enable a UART channel.

Baud Rate - is regarded as a data transmission rate. The Baud rate should be a high value, this allows the devices to send and receive signals at any time. Typically it is set to 9600Bd for UART communications.

Data bits - are the amount of bits that are sent and received by the two devices connected via UART. The length of the data bits is typically 8.

Stop bits - are used to indicate that a byte has just been transmitted. Typically a byte of data is followed by a start bit.

Parity -is another bit that is added to the end of a byte. This bit indicates whether the number of bits in the byte that are high (1). Typically parity bits are used in error detection.

3.3.7 Programming Languages

It is important to choose the correct programming languages for the project so that all of the devices are compatible with one another. It is also important to choose programming languages that all the group members are comfortable with using. The MSP430 microcontroller is compatible with C and Assembly programming languages. The development environment provided by TI allows someone to program the MSP430 in either language. The Android API primarily

uses Java and will be used to implement the Smartphone application. The following programming languages may be used within the design and prototyping stages of the project.

Assembly Language - Assembly language is a low level programming language used on programmable hardware. Every line of code is an instruction which is implemented by the device. Assembly languages are specific to particular computer architectures, such as MIPS and Intel x86. The microcontrollers used within the project can be programmed using assembly language.

C - C is a flexible language that is quite fast and places few constraints on programmers. It was originally designed for system software applications. It provides low level access to information while keeping the syntax of a high level language. The microcontrollers used within the project can also be programmed using C.

Java - Java is a programming language which is considered as an Object Oriented Language. Its syntax is a combination of C and C++ with additional features. Java programs run on a virtual machine. This allows Java applications to be ran on multiple computer architecture platforms. Since Java runs on a virtual machine this makes it easy to use on mobile applications since there is compatible with different types of phones. Java will be the main programming language used to program the DMS Smartphone application.

Extensible Markup Language - Extensible Markup Language, also known as XML, is a markup language that encodes documents in a format which is readable both by users and computers. XML is a programming language which is similar to HTML. It is intentionally designed to structure, store, and move data. It does not do anything requires a programmer to write software which can either send, receive or display the data within the XML file. Application Programming Interfaces (API) typically help programmers with processing XML files.

3.4 Application Development

3.4.1 Android Development Environment

To program an Android application, Android Developer Tools (ADT) that are compatible with the compiler of choice will be used. Java programming language is used to program Android applications. Android provides an ADT bundle for Eclipse which comes with an Eclipse ADT plugin, Android Software Development Kit (SDK) tools, Android platform-tools, the latest Android platform, and the latest Android system image for the emulator. The ADT allows the GUI to access many SDK tools and User Interface design tools which allows for quick application designing. The ADT provided by Android allows for Android project creation, building, packaging, installation, debugging, XML editors, and SDK tools all to be integrated into Eclipse. [8] In order to test Android applications, it is possible to either use an emulator or to use an actual Android device that is connected to the

computer. If an emulator is used, however, it will be likely much slower than an actual Android device so if a test needs to be run at a fast speed, it is better to use an actual Android device rather than an emulator. However, by using an emulator, it is possible to test different Android devices with different specifications in order to verify whether or not the program works on different Android devices.

3.4.2 OpenXC Application Programming Interface (API)

OpenXC API consists of hardware and software that allows developers to gather data from their vehicle in order to create android Applications for their vehicle. The vehicle interface is a small piece of hardware that reads what the vehicle is doing and transmits it to an Android application that uses the OpenXC library. OpenXC was developed to allow for people's personal vehicles to be just as easy to program as any other application. However, it cannot be used to actually modify the vehicle's behavior; it can only transmit data gathered through the vehicle interface.OpenXC contains the following signals, shown in table 4, which may be useful for making an android application for a vehicle. This isn't a complete list of all of the possible signals, it is just an example of some of the signals that may be used for the application.

Signal	**Value**
steering_wheel_angle	numerical, -600 to 600 degrees
torque_at_transmission	numerical, -500 to 1500 Nm
engine_speed	numerical, 0 to 16382 RPM
brake_pedal_status	boolean (true == pedal pressed)
transmission_gear_position	states: first through eighth, reverse, neutral
fuel_level	percentage
door_status	state: driver, passenger, rear_left, rear_right event: boolean, true == ajar
ignition_status	states: off, accessory, run, start
high_beam_status	boolean (true == on)
accelerator_pedal_position	percentage (0% == not pressed)
odometer	numerical, 0km to 16777214.000km

Table 4: Table of Some OpenXC Signal Names [9]

3.4.3 Android Connectivity and Data Storage

Android provides a rich application framework, this section will detail the different API's that DMS could possibly use to connect to other devices to receive data and to store data. [8]

3.4.3.1 Android Classic Bluetooth

Android includes support for the Bluetooth network stack allowing Android devices to communicate wirelessly with other Bluetooth devices. An Android app can gain access to the Bluetooth functionality through the Android Bluetooth APIs. Using the Android Bluetooth APIs available in the android.bluetooth package, an application can scan for other Bluetooth devices, query the local adapter for paired Bluetooth devices, establish RFCOMM channels, connect to other devices through service discovery, transfer data to and from other devices, and manage multiple connections.

Out of all of the things the Android Bluetooth APIs can do the most important for DMS will be managing multiple connections and transferring data to and from other devices. The application will be responsible for managing multiple Bluetooth connections while also receiving data from different devices. The Bluetooth classes listed in the list below are part of the android.bluetooth package and are required if the DMS is to use Bluetooth to connect to the other devices

BluetoothAdapter -The BluetoothAdapter class in the entry point for all Bluetooth interaction. In order for the android device to communicate using Bluetooth the default method BluetoothAdapter.getDefaultAdapter() must be called, and will return a BluetoothAdapter object that represents the Android devices own Bluetooth radio.

BluetoothDevice -This class represents a remote Bluetooth device and is used to request remote connection through a BluetoothSocket or to query information about the remote device.

BluetoothSocket -This class represents the connection point that will allow DMS to exchange data using input and output streams with other Bluetooth devices.

BluetoothProfile -This class contains BluetoothProfile.ServiceListener and will be important to DMS to notify other devices when they have been connected to or disconnected from service.

The various Bluetooth classes will have to be used to establish, manage and maintain connections to the other Bluetooth devices that are part of DMS. The management of the other hardware peripherals is an important part of how DMS functions.

3.4.3.2 Android Low Energy Bluetooth

Bluetooth Low Energy (BLE) has some of its own key terms and concepts that require understanding in order to use a BLE device. DMS could benefit greatly from using BLE because of the nature of its sensing devices, and for that reason the key terms and concepts are important to this project.

BLE profiles are all based off the generic attribute profile (GATT), which is a general specification for sending and receiving small packets of data known as “attributes”. Bluetooth SIG contains many pre written GATT profiles and can implement more than one profile at a time. Attributes define service and characteristics. A service is simply a collection of characteristics, and characteristics hold descriptors that are defined values.

BLE has functions with two devices having defined roles in the communication. One device has a central role, scanning for an advertisement from the device in the peripheral role. The device in the peripheral role is the device that makes the advertisement. In the case of DMS it would make sense to have the sensor act as the GATT server and the phone to be the GATT client, because the sensor should report to the phone if there is activity.

BLE uses some of the classic Bluetooth classes but also has some of its own classes for defining services, GATT, and call backs, all of which DMS will use if BLE is implemented for this project. The GATT notification class would also be of importance because the sensor would need to notify the phone of state changes.

3.4.3.3 USB Host and Accessory

Android supports USB accessories and peripherals through USB accessory or USB host mode. USB accessory mode the external hardware act as a host to the android device. Some examples of hardware that function in USB accessory mode would include devices such as docking stations or card readers. In USB host mode, the Android acts as the host, instead of the hardware acting as a host. In USB host mode a wide range of USB devices can be used if the Android application is written to correctly communicate with the device. USB host mode is the mode that DMS would use to communicate with the camera. In order for DMS to communicate with the USB camera the application must be able to establish connection and pull data from the USB camera. The USB communication between the DMS and the USB camera device will function as displayed in Figure 3. When the android device is in host mode, it powers the bus and enumerates the host device. In order for an Android powered device to function as the host device it must run Android 3.1 or later, therefore it will be assumed that users have Android 3.1 or later. In order to connect the USB camera device the android host APIs in the android.hardware.usb package will be used.

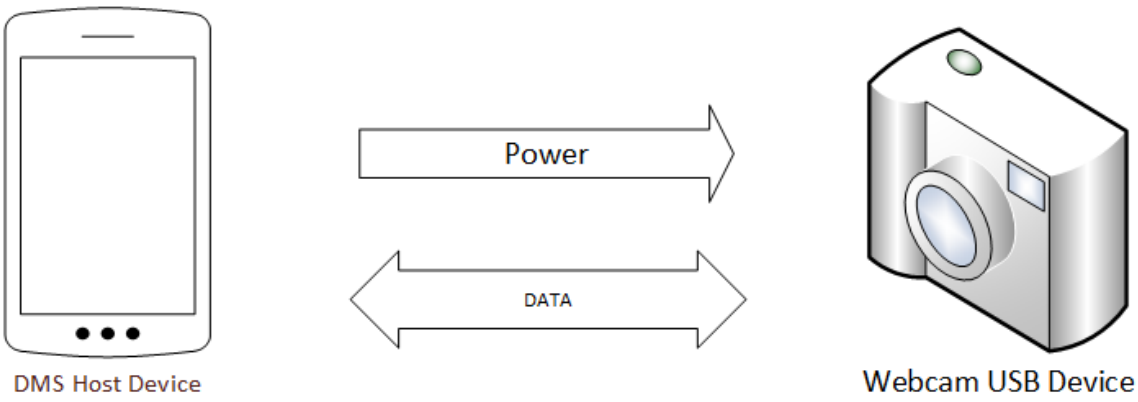

Figure 3: Communication between USB camera and device.

In order to connect the usb device and receive information using the android.hardware.usb package certain classes from the package will be required. In specific the classes listed in table 5 will be needed.

Class	Description
UsbManager	Allows enumeration and connectivity with connected USB devices.
UsbDevice	Creates a USB object that represents a USB device and contains various methods for access.
UsbInterface	An interface for the USB device that defines a set of functionality for the device.
UsbEndPoint	Will be used to establish an input/output endpoints for the communication channel.
UsbDeviceConnection	Use to transfer data on the established endpoints.

Table 5: Classes that will be used for the project.

Communication with the USB device can either be synchronous or asynchronous, and all data transmissions should be carried out on an independent thread because it is possible to interfere with other data transmissions occurring with the DMS. To establish communication the proper usb interface will be chosen, along with the appropriate USB endpoint for the chosen interface. At the endpoint a USB device connection will be opened and the function for transferring data will be called.

3.4.3.4 Android Data Storage

Android provides multiple ways to save persistent application data, which is important for DMS because long term driving data will be stored on the device.

Possible types of data storage that Android provides that will be of interest to this project include shared preferences, internal storage, and network storage.

The SharedPreferences class will allow DMS to save and retrieve persistent primitive data types. Persistent in that the data will persist across user sessions even if the application is turned off. The data types the SharedPreferences class supports are booleans, floats, integers, longs, and strings. This means that user preferences for the application can be easily saved and will persist across sessions. It is also important to note that the SharedPreferences is not exclusively just for storing user preferences, because it can store primitive data types that are static, it can also be used to store numerical information that should persist across application sessions. The methods in the SharedPreferences class that will be important to the project are detailed in table 6, the methods include how to load and save data using the SharedPreferences class.

SharedPreferences Method	**Description**
getSharedPreferences()	Allows the loading of a specified preference file.
edit()	Calling edit will instantiate a SharedPreferences editor.
putBoolean(), putString(), ext	The various put methods allow adding values to the preference file.
commit()	Commits the new values so they are saved.
getBoolean, getString, ext	Reads preference values from preference file

Table 6: Shared preferences methods that may be used.

It is important to note that getSharedPreferences() takes a file name as an input parameter because it is intended for use with multiple preferences. For DMS this function could be used to allow multiple users to use the same phone to track vehicle data.

Internal storage can be accessed by an application using OpenFileOutput() which returns a file output stream that can be used to output to an internal file on the phone device for the application. Any internal storage file is private by default, but can have its viewing rights modified by pre-defined constants such as MODE_WORLD_READABLE. If in this project internal storage is used, the methods found in Table 7A will be of particular interest.

Internal Storage Method	Method Description
openFileOutput()	returns a file output stream object that allows output to specified file
openFileInput()	returns a file input stream object that allows input to specified file
write()	used with the file output stream to write bytes to the file
read()	used with the file input stream to read bytes from file
close()	used with the file input and/or output stream
openRawResource()	Method allows the use of static file that is saved at compile time
getCacheDir()	Used to open a file if information added to file is to be cached
getDir()	creates or opens a directory in internal storage
fileList()	returns a list of files from current directory in internal storage

Table 7A: Internal storage methods that may be used.

When using cache and the device is low on internal storage space the Android device may delete the cache files. If cache files are used with DMS they will need to be maintained by the application not the device, a cache size should be set and maintained. When the user uninstalls the application the files are removed. Other storage possibilities that can be explored are cloud storage or SQlite databases.

3.4.4 Android Application User Interface

In an Android application the user interface consists of views and view groups. The view group is essentially the layout and holds other views and view groups. In this top level abstraction, every android applications UI layout is a collection of view groups and views. An example of a view hierarchy for an application such as DMS is shown in Figure 4.

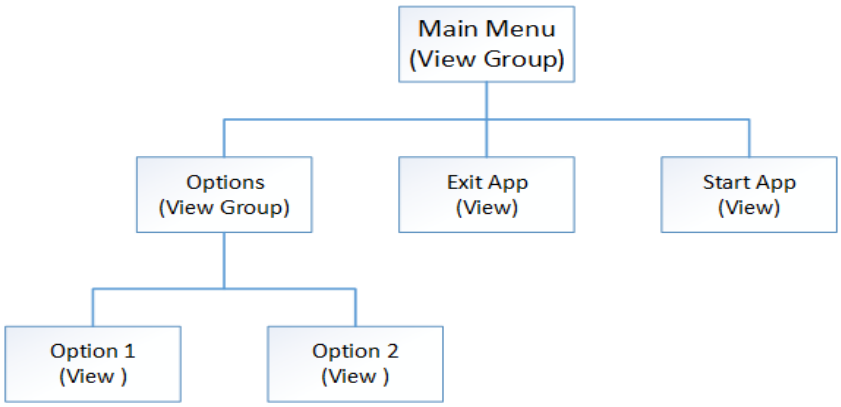

Figure 4: View Hierarchy for an Android Application

A view hierarchy tree can be as complex or as simple as a program needs but for DMS the hierarchy should be simple because a specification is to have an application that is easy to navigate. The layout of the hierarchy tree can be made by declaring view objects in code and manually building the hierarchy, but the Android developers guide suggests that the best way to create user interface layouts is with an XML document. [8]

3.4.4.1 Android UI Layouts

Android provides two main ways to create layouts for an application. The first way is to create the layout using an XML file, and the second way is to create the layout programmatically in the source code as one would with a Java application. The first option of using the XML file to define layouts has its advantages over creating the layout programmatically, the drawback is that not everyone who can program is familiar with XML layouts, and it is arguable that anyone programming in Android should be, so for this project the method of using XML will be explored in further detail. One of the main advantages of creating the application layout using XML is this allows the programmer to better separate the code that controls the behavior of the app from the portion that displays the events to the user. When using XML the UI is external to the code for the application which means that the code can be modified and adapted without extensive reworking of source code.

Android provides four major layout options: Linear, Relative, List and Grid. Linear layout allows aligns all of its components in either vertical or horizontal rows. Linear layouts can be nested into a parent-child relationship. In Figure 5 an example of this parent-child relationship is shown, the main box represents a parent linear layout that has its orientation set to vertical. With the parent having a vertical layout its children will be placed vertically but each child can also have its own linear layout. The green children have a horizontal linear layout but are placed vertically within the parent. The red child represents a larger standalone child that is not part of a nested group but simply placed within the parent in a vertical orientation. Linear layouts can be used in this way to create very complex nested layouts but is also very inefficient. For More complex layouts a

relative layout can be much more efficient than a linear layout. Linear layout also supports the weighting of child elements. The weights given to children determine how much the children pad to fill out remaining space on the screen. For example, in Figure 5 if the green children had a weight of zero and the red child had a weight of 1, it would stretch to fill out the rest of the parent because it has a higher weight value than the green children, and a weight of zero on the green children means that they will not change size at all.

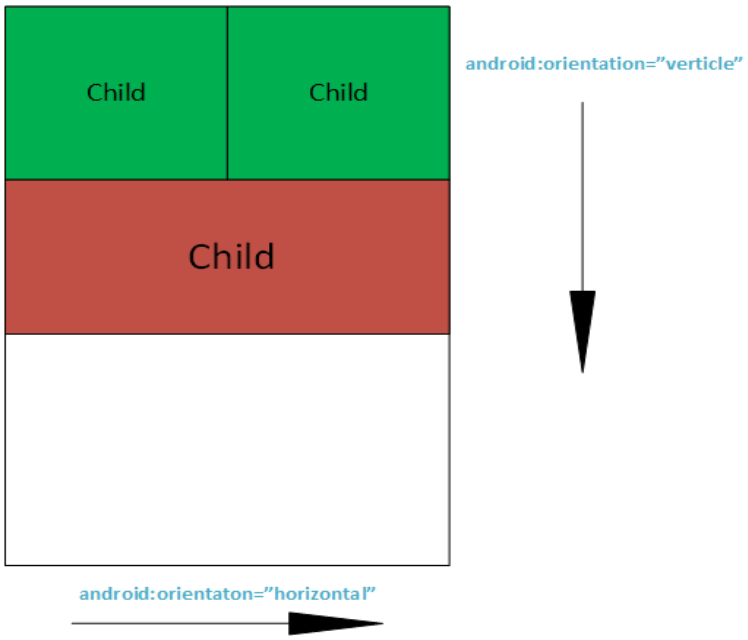

Figure 5: Linear Layout example

Relative layout is a layout that allows child views to be displayed in relative positions and can be used in combination with linear layouts to create more efficient user interfaces that do not require multiple nested linear layouts. Each child's position can be specified relative to its sibling using commands such as:

- **android:layout_below**- This can be used to place the specified view below a resource that has been identified by the resource id.
- **android:alignParentTop**- When this is set to true the view will be aligned so that its top edge will match the top edge of its parent.
- **android:alignParentBottom**- When this is set to true the view will be aligned so that its bottom edge will be aligned to the parent's bottom edge.

Relative layout has an entire library full of different assignments to adjust the location of views relative to other components, the examples above show how relative view can be used to eliminate nested linear layouts. The full list of layout properties is available in the android class RelativeLayout.LayoutParams. In Figure 6, that some layouts can be achieved more effectively. Instead of login

and cancel being part of a nested linear view each can just be a component of a relative view and do not need to be nested, just linked with a Boolean variable. Keeping the layout with as few “layers” as possible is a convention that is used to improve the performance of apps. Keeping the layout simple is not only a requirement for DMS but also keeping the layout with as few layers as possible will help the application run faster which will be important to an application that performs many calculations.

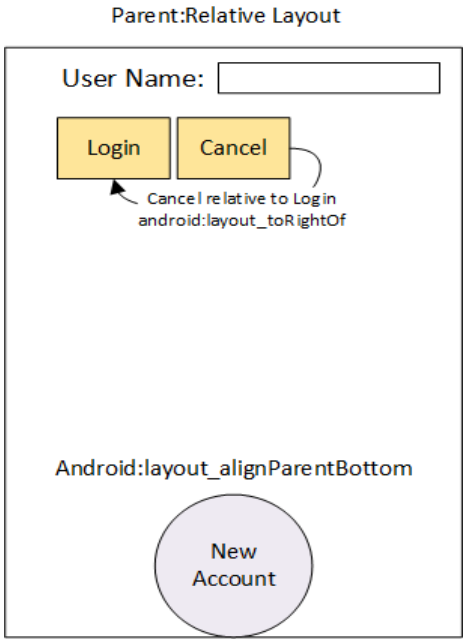

Figure 6: Relative Layout Example

For DMS a combination of linear and relative views would be the best option for creating a layout. The layout will for DMS should be coded in a XML file and stored in the layout folder of the application and simply instantiated in the OnCreate method in every android app. It is important to note that in the XML file dependencies against other views. For example in Figure 6, Login and Cancel could be declared below User Name before user name is aligned to the top of the parent. For this reason it is also more convenient to code the android layout in XML than coding the layout programmatically.

3.4.4.2 UI Input controls

Input controls are important to every application, they act as the medium to transfer input from the user, through the UI, to the program. Essentially they allow the user to tell the program what functions they would like performed, input information into the program, and can also allow the program to prompt the user for information. Android provides a variety of input controls that interface with the user interface layouts, but because of the simplicity of the layout for DMS not all of the input controls need to be researched, the input controls that will be needed in the DMS application are detailed in Table 7B.

Control Type	Class Name(s)	Description
Button	android.widget.Button	Simple push button that can be clicked to provide input options from user.
Check Box	android.widget.CheckBox	A basic on/off switch that can be used in groups that are not mutually exclusive
Radio Button	android.widget.RadioButton android.widget.RadioGroup	A basic on/off switch that can be used in groups, however only one option can be selected.
Toggle Button	android.widget.ToggleButton	A singular on/off switch that has a light indicator
Spinner	android.widget.Spinner	A dropdown set of options that allows one selection to be chosen.
Text Field	android.widget.EditText android.widget. AutoCompletTextView	An editable text file that can provide output to users and do things like an automatic text file.

Table 7B: Android input control buttons.

The android.widget class contains many other useful methods for controlling input but these are the most important to the DMS application. Each of these input controls have their own functions that allow the application to check for user input and methods that allow the programmer to create custom looks for the application. For example a button can be sized and textured with an image, or a custom shaped button can even be created for the application. The other mentioned input controls have similar versatility and customization to the button. The widget class is very versatile and will enable DMS to have professional looking input controls. The input control buttons seen in Table 7B are used by most Android applications. Android makes these input control buttons implement for an app designer. When designing the Android application layout, these will be used to interface the Android application with input functionality. This allows the user to use the buttons placed on the screen to easily navigate through the menus. Additional input control buttons may be needed to implement the DMS.

3.4.4.3 UI Event Handling

Although input controls such as buttons and handlers allow the user to interact with the application, each input control needs event handling to communicate to the application what the user wants to happen. For example if there is a mute button and the user pushes the mute button, the buttons event handling should mute the volume. The goal then for event listeners is to capture the events that are specific to the view object that the user interacts with. With Android, the View class is used to accomplish the events from a view. The methods listed in Table 8 are call back methods included in the android interface that will be important to creating event listeners for DMS. Each callback method will be called by the android framework for each implementation of a listener.

Method Call Back	**Listener**	**Description**
onClick()	View.OnClickListener	If an OnClickListener is attached to a view object such as a button, when the button is clicked the android framework will call the onClick() method defined for that OnClickListener
onKey()	View.OnKeyListener	If an OnKeyListener is attached to a view object , when the defined key or key(s) are pressed the onKey() method will be called by the android framework
onFocusChange()	View.OnFocusChange Listener	If an OnFocusChangeListener is attached to a view object, the onFocusChange() method will be called if the user navigated onto or away from the item.

Table 8: Table of method callbacks for event listeners.

There are a number of different ways to attach a listener to a view and implement the callback function to do what is desired. The programmer can choose to create an anonymous implementation of the listener and then attach the listener to the button in the application onCreate() method, or the Activity that contains the onCreate() bundle can implement the listener. The third way and the way that DMS will most likely handle listeners for view objects is to define the listener in the XML layout, then within the Activity that hosts the layout create a function with the same name that is defined in the xml layout. The function must be public, return void, and take only a view as a parameter. The relationship

between the different parts of the code are detailed in Figure 7. In the XML layout the code highlighted in red tells android that the login button will have a click listener and the corresponding onClick() method that should be used is called “Login”. In the code the function Login is contained in the Activity that implements the layout that contains the button. In this way the listener is defined in the XML layout file, defined in the code, and the two come together to create a button that has a listener attached that executes the Login function when clicked. By attaching the listener in the XML layout, the code becomes more organized and the layout becomes easier to debug if problems arise.

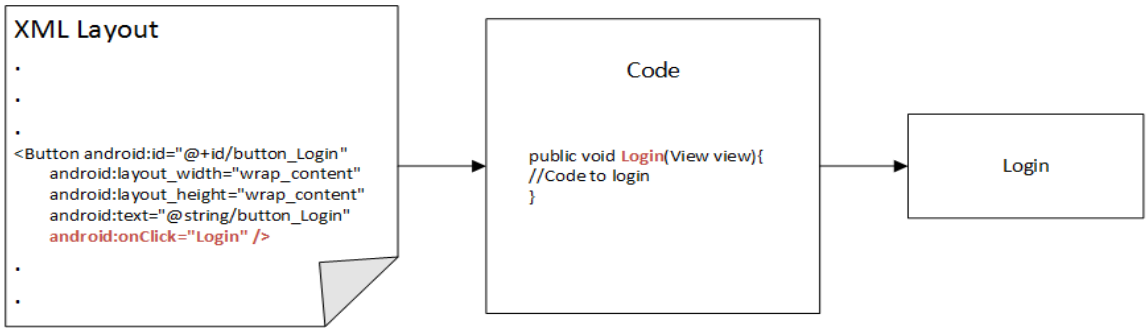

Figure 7: Process of rendering a button in layout with click listener attached.

3.4.4.4 UI Settings

DMS will need to have settings that allow the driver to customize the application for his/or her needs. For example if the driver wants to disable noise notifications, set how often data should be reset, or other various settings, they should be handled properly. The convention when programming an application for Android is to handle application settings using Android’s preference class. The reason why Android suggests that the settings be handled through the android.preference. Preference class is to maintain consistency across applications. If DMS uses a settings panel, the settings will be managed and displayed using the Preference class. The setting class does not use view objects like the rest of the user interface, instead the setting menu is generating using Preference objects. Although the settings menu is built out of Preference objects, the Preference objects are still declared using an XML file. The generation of the UI for the settings is handled by Android automatically so that the settings for the application are consistent with other Android setting menus. When a preference object is declared in the XML file it a key-value pair is also associated with the Preference object and saved in a SharedPreferences file that is default used for storing application settings. SharedPreferences data was covered in section 3.4.3.4, and details how to access these files and the types of primitive key-value pairs that can be stored in a SharedPreference data file. The Preference objects are similar to the input control view objects covered in 3.4.4.2, containing lists, check boxes, ext. Similar to declaring a layout in XML except

preferences also have their own XML attributes and are saved into a separate folder. The folder that the settings preference file should be saved in is res/xml and is traditionally named preferences.xml but can be named anything as long as it is a legal name. The XML attributes found in Table 9 are of importance to creating a XML file that contains the settings that DMS will need. There are many attributes for preference objects that allow for customization such as custom layout attributes, or attributes specific to a preference object type. However, all preference objects will have the three attributes found in Table 9.

XML Attribute	Description
android:key	The key is required for any preference that is going to store a data value in the default SharedPreferences file for settings. The key is represented by a string and is used as a unique identifier for the key-value that the preference stores.
android:title	The title attribute will specify what the name of the preference object is. This name is what will be displayed to the user.
android:defaultValue	The defaultValue sets the default key-value for a preference object. Android coding practice specifies that the default value for a preference should always be initialized.

Table 9: Required XML Preference Attributes

The Preference class contains many other advanced features for settings such as the ability to open a web address to configure settings, creating sub menus, and displaying data usage for phones. The additional features will not be used by DMS, the requirements set for DMS specify that the application should be easy to navigate and set up, therefore the settings for DMS should be easy to access and set.

3.4.4.5 UI Toasts

In order for the DMS application to supply feedback suggestions to the driver, user interface toasts will be needed. A toast allows an application to provide feedback in a simple popup. The toast will only take up as much space on the screen as is required to encapsulate the message in the toast. Another important feature of a toast is that the toast will not interrupt the current activity. In the DMS

application if a suggestion is made while the vehicle is at a standstill, the suggestion should not interrupt the current on-screen activity, therefore the user interface toasts will be the correct choice for providing feedback to the driver. To create a toast, the toast object must be instantiated with the makeText constructor, and is displayed to the screen with the toast.show() method. For the purpose DMS will use the toasts for, the toasts will need to be able to have position changes, and custom layouts. The position changes are detailed in Figure 8, will allow the DMS application to present toasts in different areas of the screen, with a visible and easy to view layout.

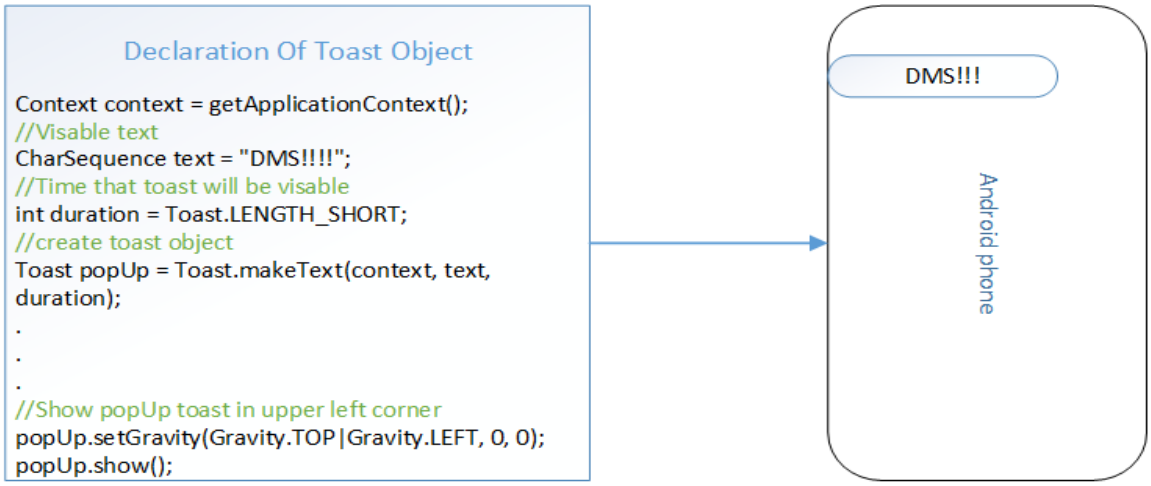

Figure 8: Position change for toast

Toasts come with a default layout and background, in the toast shown in Figure 8 if what is shown is the default layout provided by Android and DMS needed one with a different color background and a vertical layout, a custom icon image, and colored text, a custom layout for the toast can be declared in an XML layout file and applied to the toast, the process for this is detailed in Figure 9.

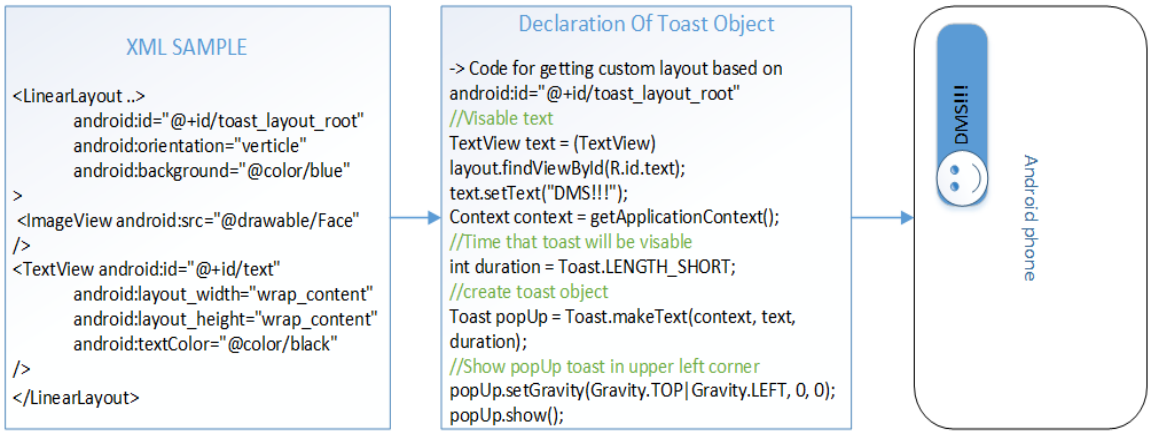

Figure 9: Using XML layout file to create custom toast.

The XML sample is not the complete layout file for the toast but contains the key values that are needed for the features of this custom toast. In the layout the toast is assigned an id so it can be found, an orientation, and a background color. The image section tells the layout what drawable image is going to be used (@drawable/Face) and the TextView formats how the text behaves within the toast. Using a XML layout file to define custom toasts will keep the toast layouts consistent with how the layouts for the rest of the application will be designed. DMS will have multiple different toasts so for each unique toast a layout file will have to be created with a unique id.

3.4.4.6 Styles and Themes

Styles and themes allow the developer to set different properties for the application in terms of font, layout, and color. They can be used to easily separate the design from the content of the program because they keep the general layout properties in an easy to manage spot, rather than mixing them in with other parts of the code. This allows the developer to keep the code organized and also to be able to easily switch between different styles and themes within different parts of the code. Styles and themes are similar but the difference between them is themes are applied to an activity rather than a view, as with styles. So, styles will only affect a single screen, but themes will affect a larger section of the program. For example, in the layout XML file, instead of defining the layout width, layout height, text color, and typeface every single time, the developer can turn that into a style and instead simply use the name of the style in the layout XML file instead. [8] An example of how the code in the layout XML file is organized is shown below in figure 10. The first box represents the original code. The second box represents the same code but in a simplified manner. The third box represents the style code which is not stored in the layout XML file, but rather in an arbitrary XML file in the res/values/ directory.

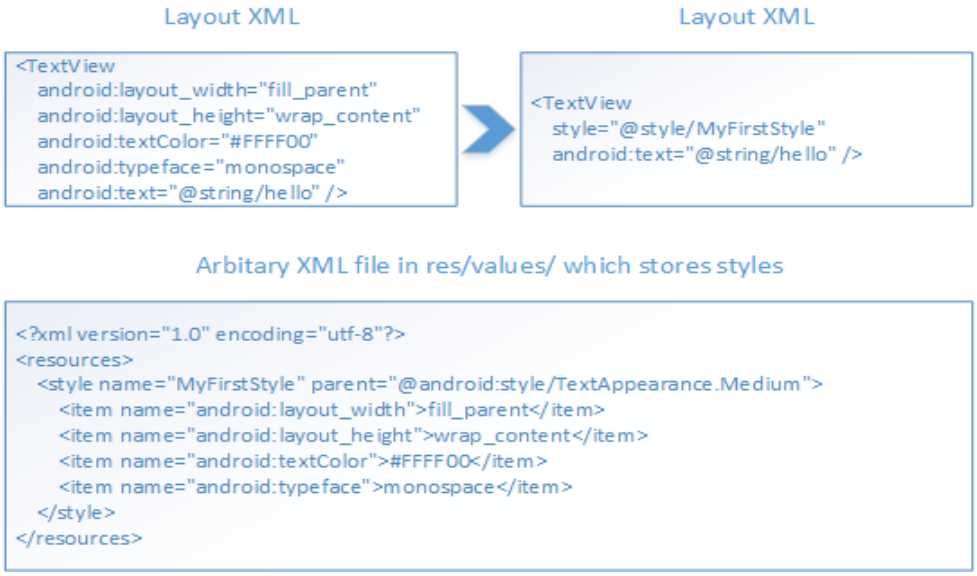

Figure 10: Styles and Themes example

Styles and/or themes will be extremely useful in making the app accessible to people who have varying levels of color blindness. Different themes can be created and made available through the settings panel allowing vision impaired users to make adjustments so they can safely see the application. Styles and themes can also be used to give users a variety of layouts that will allow the user to customize the look of the application in fun ways.

3.4.5 Graphing Tools

GraphView library is an open-source library for java programming to be used in Android applications. It will be used for this project because of its different features which will be detailed below. This library allows for the creation of both line charts and bar charts with few lines of code, so drawing the graphs will not cost a significant amount of time. This is important because the data will be gathered in real time and there may be a lot of data points so this should not cause the application to slow down or potentially harm other parts of the application. The data can be inputted into a chart and displayed in the application with different options to display the desired information. It can show multiple lines on a single graph if desired. A legend can be displayed for the user to be able to quickly understand the graph. The library allows for incomplete data to be inputted which is necessary for this project because, for the charts which will look at improvements over time, not all time slots will have data. It has the ability for the programmer or user to set the viewport so that only some of the data will be on the screen at once. In addition, it allows for easy scrolling by the user so the user will be able view past data points in an easy-to-use manner. This is necessary for this project because the user needs to be able to see how they have improved from day one until present day. The library also allows for zooming so the user can adjust exactly where they want to look on the chart and how much of the chart they want to be displayed at one time. The programmer is able to change the colors of the charts using this library, which is necessary for colorblind assistance and aesthetic or functional purposes. [10]

3.4.6 Colorblind Assistance

For an application that wishes to allow people to have a real-time knowledge of their fuel efficiency without distracting from driving, it is necessary to make adjustments to the display for people to customize based on which type of vision they have. With normal vision, it is possible to see all colors, including the traffic light colors: green, yellow, and red. Colorblind people may be able to tell which traffic light is on based on which they perceive to be brightest and the position it is in, but that is not a possibility with this project’s application due to the screen being a single, solid color.

People with monochromacy have a very uncommon condition, but they are able to see only black, white, and different shades of gray. There are two types of red-green color-blindness; people with deuteranopia color-blindness have an

absence of green retinal photoreceptors and people with protanopia have an absence of red retinal photoreceptors. They are both able to see blue and brownish yellow shades. People with tritanopia have blue-yellow color-blindness; they have an absence of blue retinal receptors. They do not have the same issues with the red, yellow, green system in comparison to people with red-green color-blindness, because for them red looks like red, green looks like light greenish blue, and yellow looks like white. [11]

3.5 Sensors

Both the blind spot sensor and the collision detection sensor will be made up of a sensor. The blind spot sensor must be able to determine when another vehicle has entered the driver's blind spot. The driver will then be alerted via their android device of which side the vehicle is located. The collision detection sensor must be able to determine whether or not a vehicle is located in front of the driver. It must also be able to calculate the distance between the vehicles in front of the driver using this information along with the speed of the driver's vehicle to determine when the best time to brake is.

3.5.1 Ultrasonic Sensors

Ultrasonic sensors work by sending out a sound wave and measuring the echo that is received. This is also known as sonar. Ultrasonic sensors use a transmitter and receiver to determine if an object is in front of it. This can be seen in Figure 11. A transmitter initially sends a pulse and a receiver looks for the returning signal. Sending and receiving a signal allows the sensor to determine the distance of an object by calculating the time elapsed between the emitting and receiving of the signal. These sensors are very good at determining the distance an object is from the sensor. One drawback that the ultrasonic sensor has is that when it is placed at an angle it may not receive the signal back or cause false triggers. Another issue is that there is no way to distinguish objects from one another.

The following equation is used to determine the distance of an object. Where *d* is the distance of the object, *c* is the speed of sound in air, theta is the angle of incidence between the signal and the object being detected, *t* is the time between the emitting of a sound wave and detecting of the echo.

$$\mathrm{d} = \frac{\mathrm{ctcos}(\theta)}{2}$$

Figure 11: Equation for ultrasonic distance

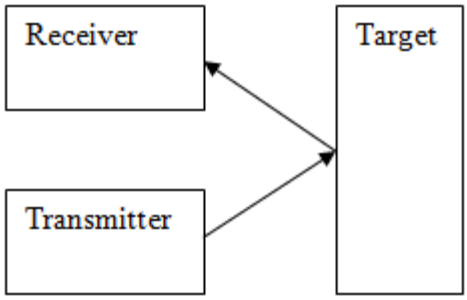

Figure 12: Visual representation of how an ultrasonic sensor works.

There are two types of ultrasonic sensors, piezoelectric transducer and electrostatic transducers. The main difference between the two are the materials that make them up. Electrostatic transducers contain a thin metal plating while piezoelectric transducers are made of a ceramic material that is bonded to a case. Electrostatic transducers are generally more sensitive than piezoelectric transducers. Although they are more sensitive electrostatic transducers cannot be sealed in a protective casing which makes them vulnerable in rougher environments.

3.5.2 Passive Infrared Sensors

Infrared sensors measure infrared light emitting from objects. Infrared light oscillates in a range of 300 GHz to 430 THz and is not visible with the human eye. These sensors are useful when detecting motion as they look for a change in infrared levels. An infrared sensor is typically made of detecting halves that both measure infrared light. There is no motion detected when each half measures the same infrared light level. When there is a moving object in front of the sensor each half will measure a different infrared light level, this allows the sensor to detect an object in its range of vision. These sensors are easy to use, last long, and consume very little power. Although one limitation includes the beam width being very small. This means that an object must be directly in front of the sensor in order to be detected.

3.5.3 Microwave Sensors

Microwave sensors act in a similar manner to ultrasonic sensors. This type of sensor uses electromagnetic pulses to detect motion. Microwave sensors send out high frequency electromagnetic pulses and measures a change in the reflected frequency that is received. When an object moves into the field of a microwave sensor, the sensor compares the measured frequency to the average measured value. These sensors are based off the basic Doppler Effect principle. The Doppler Effect is a change in frequency of the emitted waves caused by the movement of an emitting source relative to an observer.

The following equation can be used to set the sensitivity of a microwave sensor by ensuring the change in frequency is large enough to be measured. Where f^x is the observed frequency, f is the emitted frequency, c is the speed of the wave in the air, v_s is the velocity of the source and v_o is the velocity of the observer.

$$f^x = \frac{(c - Vo)}{(c - Vs)} \times f$$

Figure 13: Formula for calculating Doppler Effect. [12]

3.5.4 Light Sensor

Light sensors, or photosensors, have the ability to detect visible light, infrared light and ultraviolet light. To implement a photosensor a photodiode is typically used. Photodiodes can convert light into either current or voltage. Photodiodes work when a photon reaches the diode. The photon then excites an electron within the diode creating a hole-electron pair. The number of hole-electron pair's increases as more light is applied to the photodiode. This leads to an increase in current flowing. In the case of not being able to retrieve the turn signal status from the OBD-II port, a light sensor may be used to determine when the turn signal is being used. The light sensor must small enough to be placed inside of the tail light.

3.5.5 Alternative Options

One alternative sensor that could be used to implement the project is a lidar detector. Lidar detectors measure distances remotely using a laser instead of sound waves. A laser is reflected off a target and the reflected light is analyzed to determine the distance of the object. Unlike radar, lidar detectors reflect off all types of materials which is useful when dealing with different type of targets. Lidar detectors are very accurate over long distances, but they are also very expensive. These sensors are used in military applications, robotics, and many more areas. Lidar detectors face limitations when used in moving applications because the object must be actively targeted.

3.6 Wireless Communication

3.6.1 Bluetooth

Bluetooth is used in all sorts of electronic devices from mobile phones to watches. The Bluetooth protocols let devices find and connect to each other in an act called pairing, and securely transfer data. Bluetooth is a proprietary host-slave wireless communication protocol that uses predefined profiles that act similar to structures in coding, defining the type and names of data to be expected in data transfer. Bluetooth comes in both standard and low energy

architectures, the low energy architecture being specifically designed for sensor communication. [13]

3.6.1.1 Bluetooth Architecture

The basic unit of a Bluetooth system is known as a piconet. A piconet consists of a master node with up to seven slave nodes within ten meters. Multiple piconets can interact with each other using bridge nodes, interacting piconets are known as a scatternet. A device connected to the master can be placed into a “parked state” to save battery. In the parked state the slave cannot do anything except respond to an activation or beacon signal from the master. With Bluetooth the master device controls the conversation, determining which connected device gets to communicate. The conversation between devices in Bluetooth can only occur between master and slave, slave to slave conversation is not possible.

3.6.1.2 Bluetooth Low Energy

Bluetooth 4.0 with low energy technology can run for years on coin cell batteries. The low energy Bluetooth is good for small, discrete data transfers. With low energy Bluetooth data can be triggered by local events such as sensor, and data can be read at any time by a client. Although Bluetooth low energy does not support data streaming it is designed for sending small chunks of data for state alerts. The Bluetooth low energy is most likely the best choice for the blind spot and collision detection modules, because they only need to send a very limited amount of data to the host device. Standard Bluetooth low energy specifications can be seen in the table below.

Range	~ 150 meters open field
Output Power	~ 10mW (10dBm)
Max Current	~ 15mA
Latency	3 ms
Sleep current	~ 1µA
Modes	Broadcast, Connection, Event Data Models,Reads, Writes

Table 10: Table of Bluetooth low energy technical specifications.

The blind spot detection feature needs to have a fast data transfer time because if something is in the driver's blind spot the alert cannot be delayed for long periods of time. The latency of 3ms is fast enough to handle the data transfer for the blind spot detection feature.

3.6.2 Zigbee

Zigbee networks allow multiple devices to connect to one another wirelessly. Zigbee networks provide fast and reliable connections which is useful when constantly transferring data. Zigbee is a wireless personal area network (WPAN). It is typically used for periodic data transfer from an external device, such as a sensor. The range a Zigbee network may span can range from a few feet to approximately 70 meters, which is large enough to implement the project. The downside to using Zigbee technology is that it is not fully compatible with Smartphones. This is due to the smartphone's wireless technology operating on different frequencies and protocols from the Zigbee. This is an issue when dealing with mobile devices that may be in constant use. Using a Zigbee connection may also drain the battery rapidly, which could result in a much shorter battery life.

3.7 Power

3.7.1 Batteries

The DMS components will be powered by either a lithium ion battery or the lead acid battery located in the vehicle. The battery chosen must be able to power each circuit which will be located in different areas of the car. The battery must also be rechargeable so that it may be used over a long period of time.

Lead-acid batteries - Using a lead acid battery is a convenient and cheap option due to having one already being used by the vehicle. A lead acid battery in a vehicle provides 12 V which can be used to power each circuit attached to the vehicle. Some major benefits of the lead acid battery in a vehicle include high reliability and lifespan, can be easily replaced by the driver, and constant recharging by the alternator located in the vehicle. Powering the circuits attached to the vehicle with the lead acid battery will require a much more complicated installation process for the user.

Lithium-ion batteries - A lithium-ion battery is useful due to its compact size which will allow each circuit to be powered separately allowing for a wireless design. Most lithium-ion batteries provide about 3.6 V to 4.1 V. A charging circuit and controller will be required to maintain the batteries longevity. One drawback of the lithium-ion battery is that it ages and deteriorates while not in use. A lithium-ion battery is also sensitive to heat which will cause the battery to degrade rapidly. The charging and discharging process of a lithium-ion battery is due to the movement of energy between anode and cathode. Although the lithium-ion battery discharging process is not a chemical process, lithium-ion batteries still face performance losses over time. Lithium-ion batteries also have a self-discharge of 2-3% per month, which is low in comparison to other batteries.

Lithium-ion batteries are different from other batteries because of a higher voltage per cell. Most of these batteries tolerate a max charge to about 4.2 V per cell. The battery is no longer charged once this threshold is reached, allowing the voltage to slowly 3.6 V and 3.9 V. In typical cases a lithium-ion battery will not be fully charged. This prevents high voltages from causing stress on that battery which could lead to a decrease in performance. It is important not to overcharge past this threshold as charging past the voltage threshold can cause the lithium-ion battery to oxidize, potentially leading to the battery catching on fire and exploding. Due to this the charging circuit must be able to determine the full charge state accurately to prevent overcharging.

3.7.2 Voltage Regulator

A voltage regulator is a device which allows us to provide a constant voltage to each component of the project. Due to using multiple microcontrollers and sensors within the DMS, the voltage provided must be a constant value. The common voltage range for a microcontroller being used in the project is between 3.3V and 3.6V. Also, the sensors that will be used during the experiment will require a constant voltage in the range of 3V to 5V. The voltage regulator that will be used within the project will need to provide a max load current that can support each component, must be able to take an input from a 12V car battery or lithium-ion battery and must provide a precise output voltage. The types of voltage regulators that were researched for use within the project are linear voltage regulators and switching voltage regulators.

3.7.2.1 Linear voltage regulator

These regulators are used when stepping down a voltage. This is where the input voltage of the regulator is greater than its output voltage. Due to this the efficiency of a linear voltage regulator is limited. Linear voltage regulators use a voltage divider to set a constant output voltage. The voltage difference between the input and output is then dissipated as heat. An example of a linear voltage regulator can be seen in Figure 14.

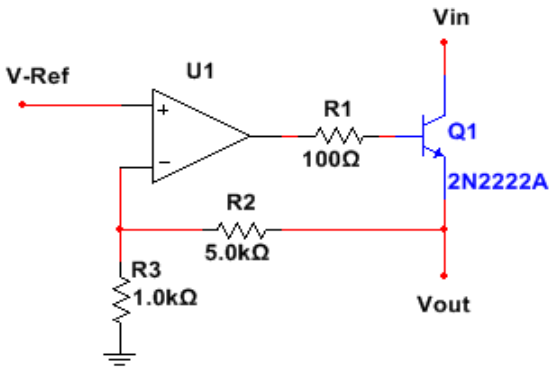

Figure 14: General schematic of a linear voltage regulator.

The op-amp's output will fluctuate to maintain an equal voltage across its positive and negative terminals. This relationship between the positive and negative terminals of the op-amp is used to regulate the output voltage. [14]

3.7.2.2 Switching voltage regulator

Switching regulators use the same model as the linear voltage regulator (Figure 14), but includes a switch to that allows it to shut off. These regulators are used to switch a device on or off. Switching voltage regulators can either step down or step up a voltage. Therefore its input voltage is not required to be higher than its output voltage. The efficiency of a switching voltage regulator is generally much greater than a linear regulator. One reason this is true is because a switching voltage regulator is either fully operating or shut off, which leads to very little heat being dissipated. One downside to switching voltage regulators is the amount of noise that is generated. The switching device within the regulator generates high levels of noise.

3.8 Rearview Camera

3.8.1 Camera

Connecting a webcam to an Android device should be able to be done via USB, Wi-Fi, or Bluetooth. Very few Bluetooth enabled webcams have been developed, so it would not be worthwhile to try to include them in the project due to the added difficulty. Wife enabled webcams tend to be expensive and would be too costly for the project and for the potential future users. In order to keep the cost low, a USB camera will likely be used, with appropriate measures taken to wire the webcam to the android phone. Logitech webcams have had success with connectivity to Android devices, so a Logitech webcam, preferably under $40, will be used. Two specific Logitech webcams, the HD Webcam C270 and HD Webcam C310, will be detailed below. These may not be the actual webcam used, but they are good options. They have the same specifications except one takes higher quality pictures so whichever one is used will be based on price rather than functionality. [15]

- 720p
- Up to 1280x720 pixels
- USB 2.0
- Automatic Light Correction
- 5-foot cable
- 3 x 8.2 x 6 inches
- Always focused

If there were more funds for the project, it would be possible to use a Wi-Fi camera instead. A possible Wi-Fi camera would be the Logitech Broadcaster Wi-Fi Webcam. It costs approximately $199. Using this camera would eliminate the

need to run a wire from the back of the vehicle to the front of the vehicle, because the device would just need to access the webcam via Wi-Fi. However, the application would need to be on an OSX device such as a mac computer, iPad, or an iPhone, because the webcam only works with OSX devices. The specifications of this webcam are shown below. [16]

- 720p
- Three times digital zoom
- Built-in illumination lamp
- Digital pan and tilt
- 2-hour battery life
- Battery charged via USB.

This would be an excellent option if the application was being developed for OSX instead of for Android. The digital zoom and pan and tilt would allow for more functionality on the device, allowing the user to change the camera angle of their backup camera. It also has the built in lamp allowing for vision when it is dark. But, the battery life, which is typical for other Wi-Fi cameras as well, would be a huge downside, as well as the cost.

4.0 Hardware and Software Design Details

4.1 Software Design

The software in this project consists of an Android application. The Android application communicates will all of the different hardware components of the project. The layout of the Android application is simplex so that it is easy to navigate. The stored data within the application is kept to a minimum in order to decrease lag while using the application. The following sections will detail the application's layout as well as different design choices such as color scheme and audio as well as functional choices such as the fuel efficiency calculations.

4.1.1 Android Application

The Android application portion of DMS consists of various menus that will either affect the program or will display information to the user. The application starts on the main menu screen and branches off into six different choices: start driving session, last driving session, driving history, your fuel economy, options, and credits. Driving history branches off into further options, either allowing the user to view one of the past driving sessions or to view an overview. This is where the driver can see all of their driving data. This data may contains a summary of the fuel efficiency score that was gathered during the history of the user using the application. Your fuel economy contains a further break down of the bad driving habits that the user frequently commits as well as indicating if the user is

improving.The options button contains different choices: audio toggle, rear view camera toggle, change colors, and test colors.It also allows the user to erase saved data if they choose. The options allow the driver to turn on or off certain features that they may want or not want. This allows the driver to customize the DMS to fit their needs. For example if the driver does not want to be bothered by the blind spot detection system, it is possible to deactivate audio notifications for their trip. This removes the hassle of needing to remove the device each time it is not wanted by the driver.The change colors sections allow the user to choose from predetermined color sets labeled normal, monochromacy, deuteranopia and protanopia, or tritanopia. These are different forms of colorblindness which help colorblind drivers use the Android application with ease.This is shown below in a view hierarchy in Figure 15.

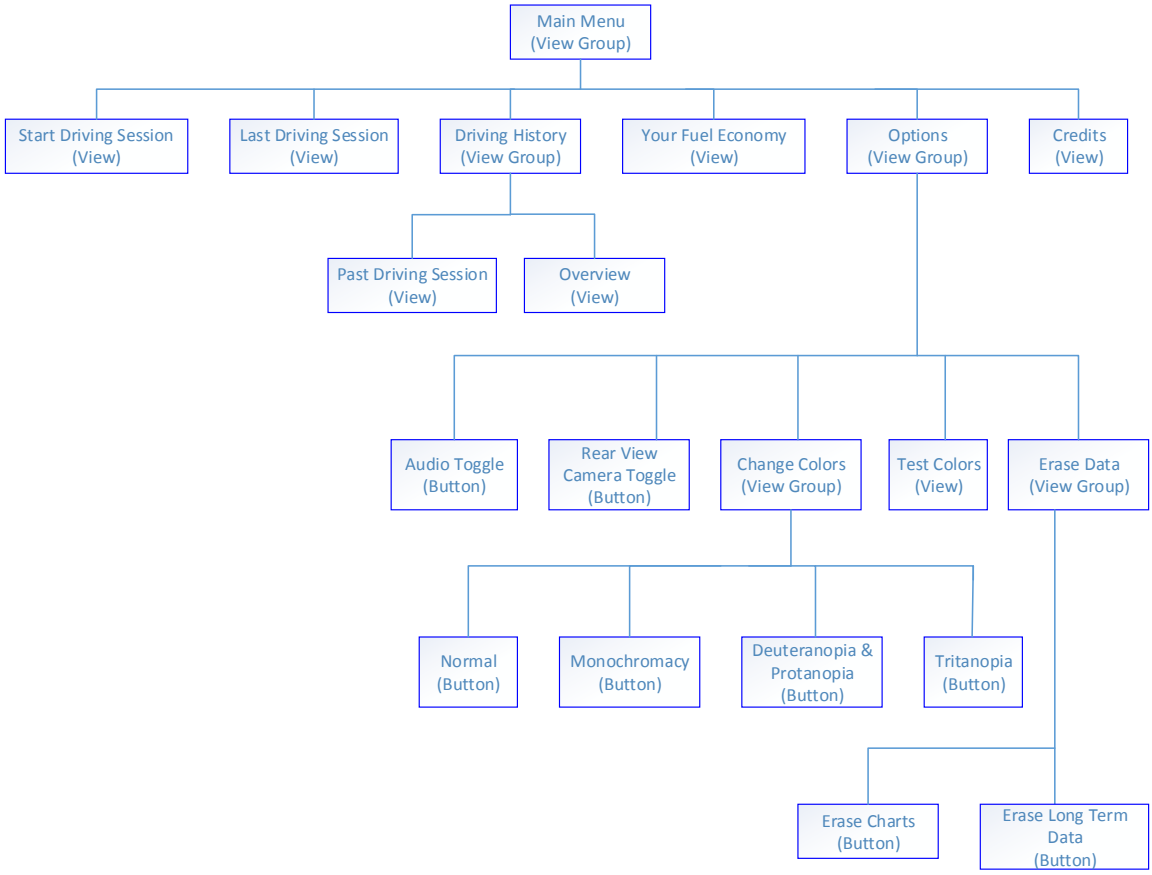

Figure 15: View Hierarchy of DMS Android Application

The Unified Modeling Language (UML) Class Diagram for the DMS Android Application is shown below in figure 16. It shows the different attributes and operations for each class. This will be used as the base of the Android application for the DMS.

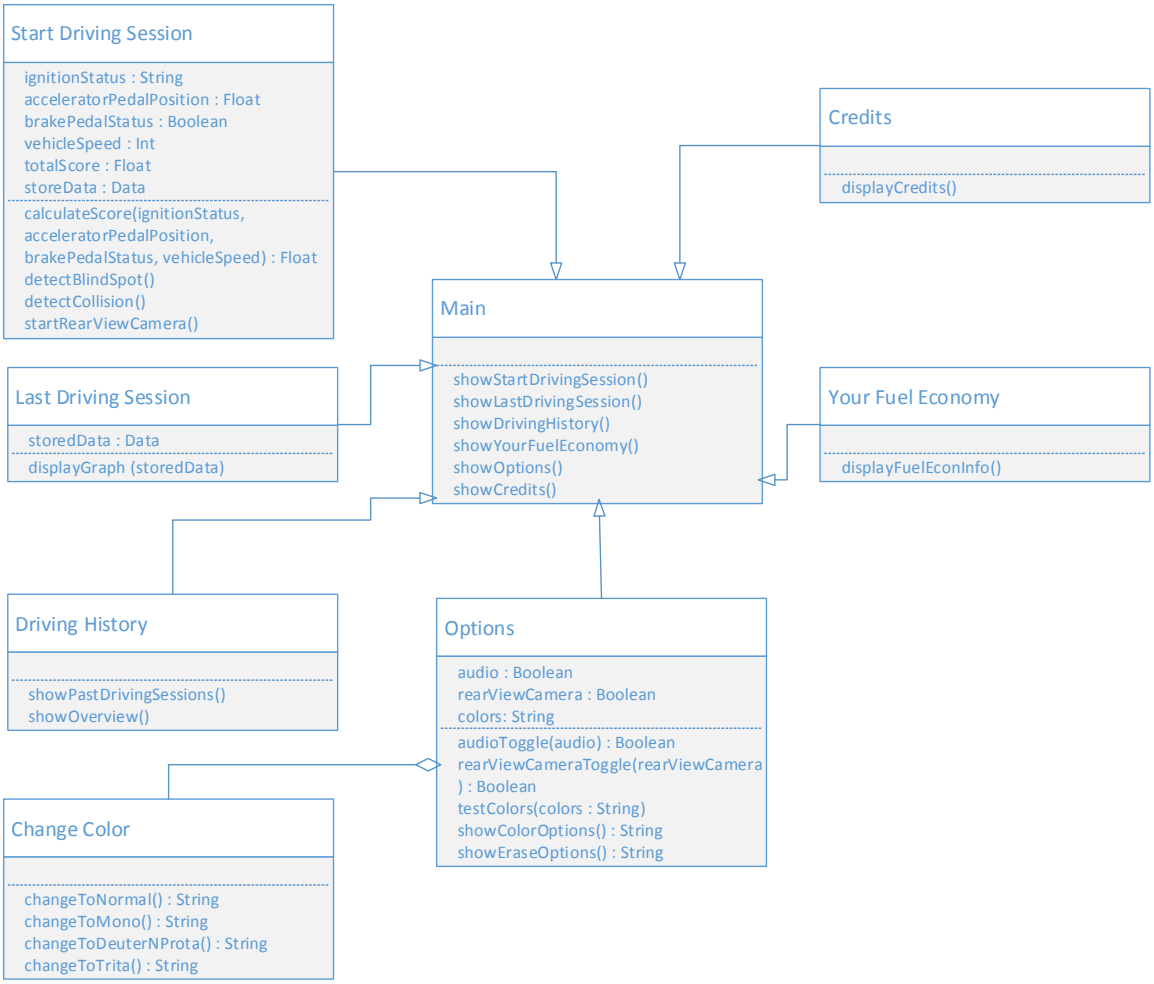

Figure 16: DMS UML Class Diagram

The UML class diagram in Figure 16 is not exhaustive, that is, the class diagram does not contain all variables and methods needed to implement all of the features in the DMS application, the UML diagram in Figure 16 is intended as a general overview of how the different classes are structured and how communication occurs using methods and instantiated objects. DMS will have service processes running in the background that perform connectivity and data transfer services in the background of the app, these background services are not included in this class diagram as they are associative. The services continue to run even when the app is downsized to the tray, in this way if the application is interrupted during operation, connectivity and data transfer do not stop.The connections between the android application and the sensors are severed when the application is returned to the tray, but the connections are reestablished immediately upon opening up the application.

The following OpenXC signals, shown on table 11, are be used in the Android application. The uses of the signals will be detailed following the table.

Signal	Value
vehicle_speed	numerical, 0 to 655 km/h
accelerator_pedal_position	percentage (0% == not pressed)
brake_pedal_status	boolean (true == pedal pressed)
gear_lever_position	states: neutral, park, reverse, drive, sport, low, first through sixth
ignition_status	states: off, accessory, run, start
steering_wheel_angle	numerical, -600 to +600 degrees
turn_signal_status	states: left, right, off

Table 11: Table of OpenXC Signal Names that will be used.

The program will not start running until ignition status is START. Although other statuses will still waste fuel, the program is supposed to assist with driving, so it does not start running until the engine has been started. The program will continue to run until the application is closed or connection with OpenXC is severed.

The turn signal status signal will be added to the OpenXC library in the future which and would have been used in this project if it were available. In the project, it would have been used to activate the left or right blind spot detection. However, because it has not been implemented yet, the steering wheel angle is used instead. A value between -110 and -45 is considered a left turn. A value between 45 and 110 is considered to be a right turn.

The gear lever position signal would have been used for the rear view camera. If the gear lever position is reverse, the rear view camera would have activated on the android device. If the user had chosen to deactivate the rear view camera, the gear lever position being reverse will result in no changes. However, because the rearview camera did not make it into the final project, this functionality was removed.

The vehicle speed signal is used to determine the user's acceleration. The past 10 accelerations are stored and a weighted average is used. The acceleration is determined by taking the current vehicle speed, subtracting it from the previous vehicle speed, and dividing by 50ms which is the clock cycle on which the application runs. It is also used to determine if the user speeds over 70mph which is stored as a bad driving habit. The vehicle speed signal is also used to determine if the user is idling, which wastes fuel. If the vehicle speed is 0, assuming that the engine is running, then the program will notify the user that they are wasting fuel.

The accelerator pedal position signal is used to determine if the user is accelerating too hard. The brake pedal position signal is similarly used to determine if the user is braking too hard. They will be used together to determine when the driver is accelerating and braking too soon afterward, which is determined by checking the brake pedal position within 10 seconds of pressing the accelerator.

All of the signals mentioned previously are necessities. Further signals could have been used to provide more data to the user. One such example would have been the fuel consumed since restart signal in conjunction with the odometer signal which could have been stored in order to tell the user their actual miles per gallon. However, for simplicity sake and to have more control over the results, this was not included. This is because the user may feel like they are not improving simply because their miles per gallon do not seem to be improving. However, it may have just been based on a circumstance that was out of the user's control that caused this to happen.

4.1.2 Real Time Fuel Efficiency Analysis

In order to determine fuel efficiency, a score between 0 (bad) and 100 (good) is given based on the driver's driving habits. At the start of each calculation cycle, which is 50ms, all necessary signals will be gathered from the vehicle interface. Before making any calculations, it will be determined if the driver is doing anything that triggers other parts of the application, such as driving too close to the person in front of them or trying to turn when there is a vehicle in their blind spot. If these occur, the appropriate sound will be played to notify the user. Afterward, orif none of those triggers have occurred, a series of if-statements will check each signal to determine if the driver is exhibiting any negative driving behaviors. These if-statements are shown in figures F1 through F6. The calculations for each different section are set on a delay to insure that a single bad driving habit does not overpower the rest of the bad driving habits in a short period of time. The actual score is based purely on the vehicle speed which is used to calculate acceleration. The score calculated this way does not change instantaneously. Instead, the score gradually counts up or down until it reaches the target score. This is to prevent the application from flashing a lot of different colors at once.Also, this will allow for the user's score to remain low if they are continually exhibiting bad driving behaviors. It will also allow the user's score to remain high if they only make a small mistake for a very short period of time.

Score: As shown in Figure (F1) the vehicle speed is gathered using past data. The weighted average is found using stored accelerations 1-9 and the newest acceleration that was calculated this clock cycle. The application checks if the user is braking or accelerating and adjusts the score to fit this, with braking having a smaller multiplier effect.

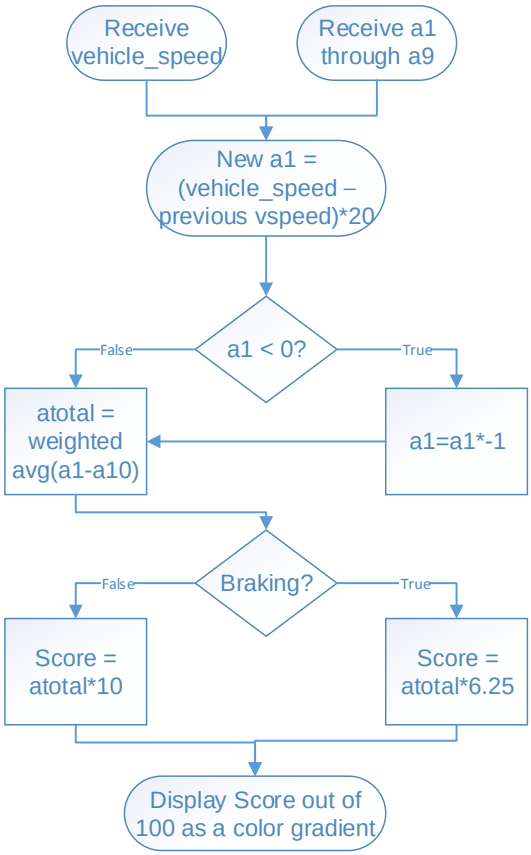

Figure (F1): Score which produces the color gradient.

Accelerating hard: The harder that the accelerator is pressed, the quicker the driver is accelerating, and the more fuel that is wasted. Figure (F2) shows how this is being taken into account. A point is added to the acceleration bad habit if the user meets the requirements, with a total of 1 point given out every 30 seconds.

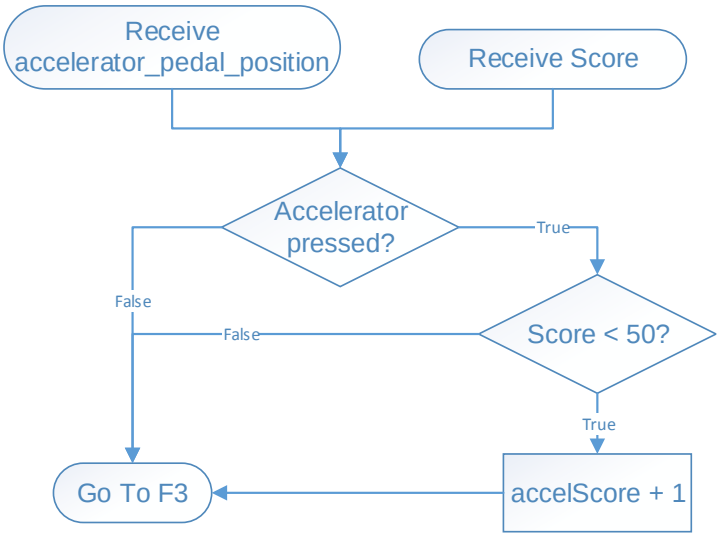

Figure (F2): Calculating acceleration score

Braking hard: Braking hard wastes fuel, so any time the driver is braking hard, a point will be added to the brake score as shown in Figure (F3). This is similar to the acceleration score from Figure (F2). As such, 1 point is given out every 30 seconds.

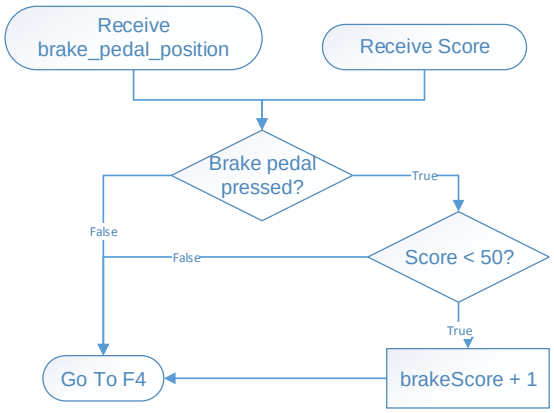

Figure (F3): Calculating brake score

Speeding and idiling:Going too fast wastes fuel and is taken into account as shown in Figure (F4). The speed calculations are done in km/h rather than mph; 81 km/h is about 50 mph and 105 km/h is about 65 mph. Speeding over 70 mph is penalized with 1 point every 30 seconds due to the maximum speed limit in Florida being 70mph. In addition, if the driver idles, which means the vehicle's speed is 0, they are also wasting fuel. Idling is only taken into account if it has occurred for longer than 1 minute, so that the user is not penalized for short stops at stop lights, stop signs, or when parking their vehicle. The idle score is only increased once every time the driver has idled for more than 1 minute. It does not increase after 2 minutes, 3 minutes, and so on.

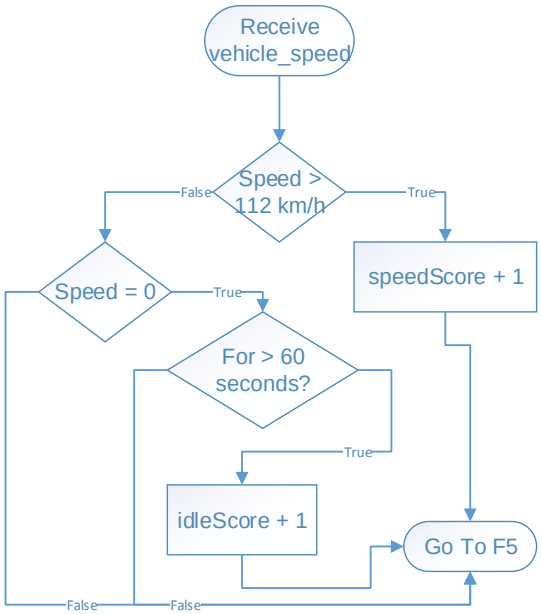

Figure (F4): Calculating speed and idle score

Idling: As shown in Figure (F5) if the driver presses the accelerator past 20% and then brakes within 10 seconds, they get a mark toward the brake and accelerate score. This is because braking soon after accelerating significantly is usually avoidable and wastes fuel unnecessarily.

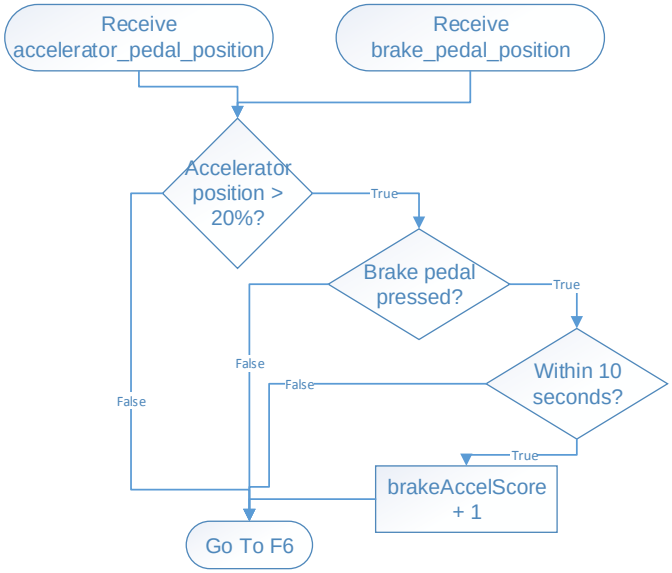

Figure (F5): Calculating idle score

Final: After all the calculations are made, they are stored away for future use within the program. If the user is idling, a message is displayed based on which of the scores is maximum. This is shown in further detail Figure (F6). Afterward, whichever message was shown has the score reset to 0 to make room for other messages to show up next time. The process should be repeated approximately every 50ms.

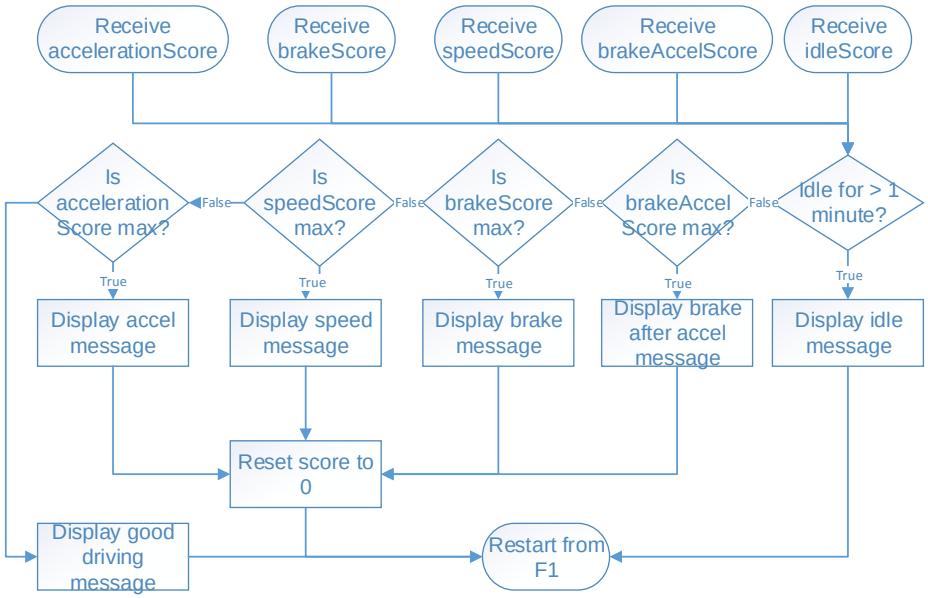

Figure (F6): Calculating total score and finishing process

4.1.3 Long Term Analysis and Messages

All of the data gathered as mentioned in the previous section is stored into shared preferences or a file on the Android application and after the driving session is over, the data needs to be analyzed to be put to use by the user. The total drive time of each driving session is divided into 10 second segments. All of the data points within each time period will be added up and averaged to find the total score for that time period. This will be placed into a new file which is read by the overview graph and displayed to the user. The original data is shown in the last driving session graphand it is also stored in the driving history section as well, until 10 new driving session has occurred, in which case it will be wiped from memory. This allows the user to view up to 10 previous driving sessions. In addition to this, as mentioned previously, all of the data will be added up and averaged over the time period. This new data point will be added to a completely separate file which will store the entire user's driving history. It will show them their improvements, even from months beforehand. If it becomes too memory intensive, the user is free to erase all graph data, however.

After analyzing the data, the program will generate messages to display to the user based on their driving habits. These messages will both be displayed on the screen while the user is at a stop light, and on another section of the program called "Your Fuel Economy." All of the messages that are possible to see in the application are shown below. The first six messages are shown while driving, and the following seven are only shown in the "Your Fuel Economy" section of the Android application.

- You have been idling for more than 1 minute. If you know you will be idling for an extended period of time, consider turning off your vehicle's engine.
- You are driving fuel efficiently.
- You are accelerating too hard. Consider accelerating slower to reach your desired speed.
- You are braking too hard. Consider slowing down by lifting off the accelerator ahead of time if you know you need to stop soon.
- You brake too soon after accelerating. Look ahead to see if you will need to brake soon, and try to not accelerate before then.
- You drive over 70 mph frequently. This decreases your fuel efficiency; you might want to slow down.
- Fuel efficiency starts decreasing at speeds over 55 mph.
- It's recommended to drive under 65 mph to maintain good fuel efficiency.
- Accelerating and braking frequently reduces your fuel efficiency.
- Idling wastes fuel, so idling for more than 1 minute should be avoided.
- Stopping quickly wastes fuel, so try to gradually reduce your speed if you know you need to stop.
- Accelerating too fast wastes fuel; it's better to only lightly press on the accelerator.
- It's a good idea to put your vehicle in neutral at stop lights.

4.1.4 User Interface

4.1.4.1 Aesthetics

The application is visually appealing while also maintaining a simplex design. The colors chosen for each color scheme option, with consideration to people with color-blindness, are non-distracting colors that contrast so that the user will not have difficulty reading charts and navigating different screens. The colors are primary and secondary colors combined with black and white allowing for a sleek display. Few images are used within the application but the charts and any displays will look professional rather than tacky. The purpose of keeping the colors and design simple is because the application is meant to be used for the user to view information that has been gathered and stored while driving and to be able to easily see how they have improved over time. The colors used while driving are intended to be easy to view at a glance, or with peripheral vision, no matter the user's color vision. This means careful consideration has to be made to make sure that the color combinations used in the gradient produces distinct colors that are not confused with other colors within the gradient. They also were chosen keeping in mind the colors typically used in the world of driving. For people with normal vision, this means green, yellow, and red. However, for people with different forms of color vision, there was consideration made to make sure that the colors were distinct but also represented what those users are able to see within the real world as well.

4.1.4.2 Layout and Accessibility

The main menu of the application has buttons for users to go to different parts of the application. The buttons are: Start Driving Session, Last Driving Session, Driving History, Your Fuel Economy, Options, and Credits. Each section after the main page has a button which allows the user to return to the previous page. This button is built into Android devices so it will not be necessary to add it to the actual program, but it will be tested to make sure it is functioning appropriately, especially at the end of driving sessions. The application is able to be used in either landscape or portrait view and the buttons are easy to view and press. However, certain screens lock into landscape mode. This happens while driving, in order to make sure everything is viewable to the user, as well as while reading some of the charts in the application.The layout is shown in figure 17. The layout in the figure is in landscape mode. This is just a demonstration for this report so that the picture does not take up too much space; it will not be required to actually be in horizontal view if the user does not wish for it. To have it in horizontal view, all that is necessary to do is to turn the Android device horizontal. It will automatically switch between a horizontal or vertical view depending on how the user has decided to hold their Android device. The same is true for the pages that will have graphs. The graphsautomatically switch between horizontal and vertical view without the user needing to press any button on their Android device.

Figure 17: Main Screen

- The “Start Driving Session” button activates the application which monitors the user’s fuel economy while driving.
- The “Last Driving Session” button displays a graph with the information gathered during the most recent driving session. It gives a general timeline showing different colored bars for the times at which the user was using good, moderate, and bad driving habits. The colors will be a gradient. Because the data is gathered in real time, the time is displayed on the x axis.
- The “Driving History” button shows a chart with a single data point for every driving recorded driving session, showing the average score for that driving session. It also allows the user to view full charts for some or all of their past driving sessions.
- The “Your Fuel Economy” button is a section that notifies the user of how they can improve their fuel economy by analyzing what has caused them to score “moderate” or “bad” during past driving sessions. It also shows them how they have improved and provides general fuel economy hints. There is a button on this screen which gives a further break down of all of the bad driving habits that have been recorded and analyzed while the user is driving.
- The “Options” button allows the user to change the color scheme of the program, test the colors they have chosen, turn on and off audio notifications, and turn on and off the rear view camera. It also lets the user erase graph data and bad driving habits data. This includes a confirmation window just in case the user presses the button on accident.
- The “Credits” button displays the names of all people who worked on the project as well as any other sources that contributed to the project.

4.1.4.3 Graphing

The GraphView library allows for data to be easily displayed in either a bar graph or a line graph. For the purposes of this project, the bar graph us used for the individual driving sessions and the line graph is used for the overall driving sessions. The bar graph allows for different colors to be visible to represent how well the user is driving and allow the user to easily see their score. The color displayed is a gradient between the three different colors representing good, moderate, and bad driving habits. The data is inputted into the graph in a series of data points. The format for the data points is (X, Y) with the number in the X place being the time and the Y place being the score. Before or after each data point is entered, it is possible to change the color of each individual point on the graph. In this project, this will be done using an algorithm that is shown below in figure 18. The score is represented on a 0 to 255 scale and no number can be a decimal. If a decimal is created by the calculation, it will be rounded to the nearest integer. Red is represented as 255|0|0, yellow is represented as 255|255|0, and green is represented as 0|255|0.

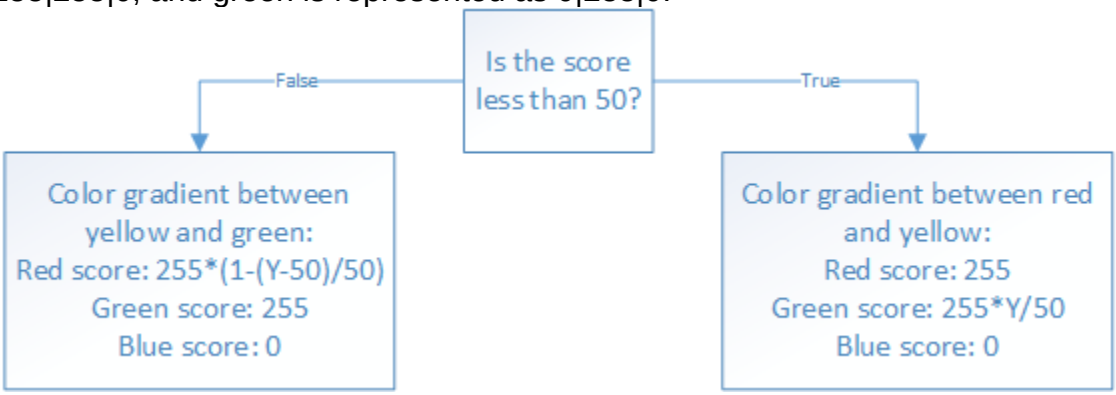

Figure 18: Algorithm for deciding color

The algorithm above produces some color between the three goal colors depending on the score. An example of how the algorithm appears in the actual program is shown in figure 19. It is possible to see how the colors fade into other colors as the score decreases. The lowest score produces a red color, but as the score gets higher it turns into different shades of orange, and then it goes to yellow, and then it changes to more of a green-yellow, before going to green. The higher shades of green are a little difficult to tell apart but they are different colors. These colors are the same colors that will appear on the actual application while driving. If the user has chosen different color settings, the color calculations will change to accommodate the new colors allowing for a gradient between those colors as well. It should be possible to come up with a generic algorithm that will produce a gradient between any three colors, but the gradient will be hard coded due to the limited amount of color options.

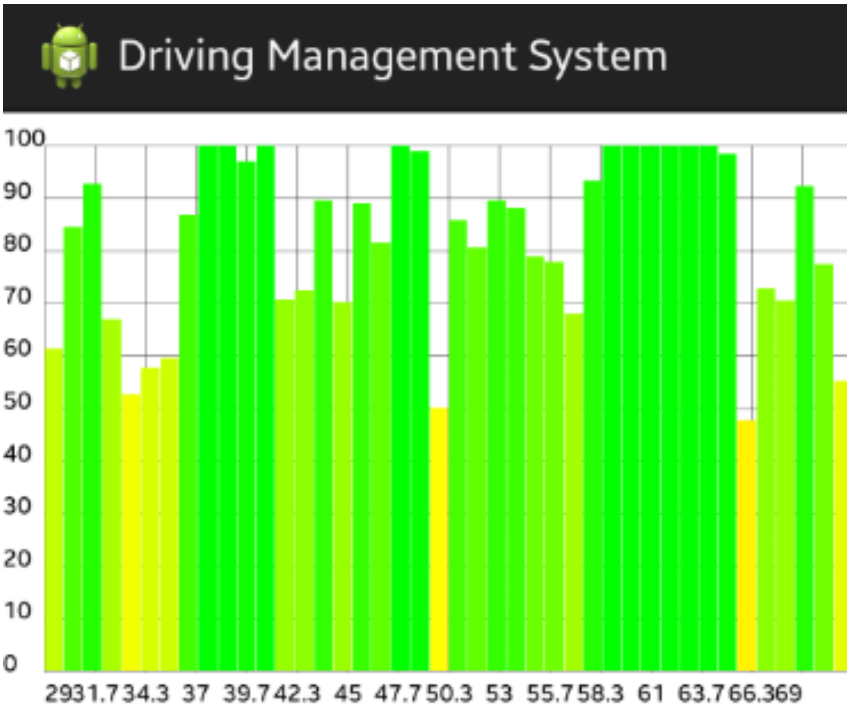

Figure 19: Example graph made using GraphView and color algorithm

4.1.4.4 Color Themes

The available color themes in the application are based on different levels of color vision. People with normal color vision have the default option: Green for good, yellow for moderate, red for bad fuel efficiency. However, different options are available for color blind people.

People with monochromacy have the option: White for good, gray for moderate, black for bad. This allows them to easily tell how they are driving because the lighter the color is, the better they are driving. This option can be used by any person no matter the amount of color they are able to see, but the graphs may not be as easy to see because the background color will have to exactly match one of the possible bar colors.

In order to allow the application to be used by people with red-green color-blindness, either deuteranopia or protanopia, there is a color option which displays: Blue for good, white for moderate, red for bad fuel efficiency. The red color looks more like a brown-yellow color to the user, but it is used to avoid confusion because the person will likely already associate that brown-yellow color with “stop” due to the prevalence of red meaning stop in society.

Even though people with tritanopia should do not have as many difficulties with telling apart green, yellow, and red, they have the option to set the display to: Green for good, white for moderate, and red for bad fuel efficiency. People with this form of color-blindness could also use the red-green colorblind option without

having any issues with the application while driving. It is still be recommended for them to choose the appropriate option on the application because the rest of the colors used in the application, such as for the charts, will be adjusted according to the options chosen. Also, using green instead of blue and white instead of yellow clears up some confusion between colors.

There is a preview option in the application so that the user can make sure they can tell the different colors apart, as well as enabling them to familiarize themselves with which colors represent good and bad driving behavior. The colors displayed are not three solid colors only. Instead, they are a gradient between the three colors. The colors can be seen in the following image.

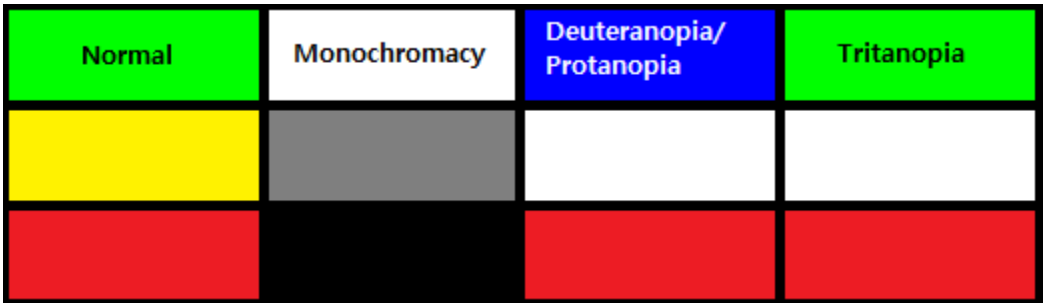

Figure 20: Different color options different types of color vision

4.1.5 Safety Features

The part of the application that deals with recording information while the user is driving purposefully will not have any buttons or any way for the user to interact with it. This is to discourage people from messing with their phone while they are driving. If the application is closed prematurely, such as if the user receives a phone call,it is reopened automatically without needing the user to interact with it. If the user is unable to establish connection with OpenXC, they are able to see if on their phone before driving so they can fix the issue when they are in a safe position. Also, if the user quits the application after pressing the “Start Driving Session” button but before actually starting their car, they will not receive any error message.

The application does not contain any flashing images or colors in order to be safe for people who have epilepsy. If the user is stable enough to drive, the colors changing on the phone while the user is driving should not be quick enough to trigger a seizure. However, there are safeguards in place to make sure that the colors do not change in rapid succession, considering that there may be a passenger who has epilepsy and may be able to see the display. This is done by changing the color gradient gradually on every clock cycle rather than having the color change automatically to the actual score that was calculated on that particular clock cycle. This also prevents false positives or negatives, because a

score being good or bad for a short period of time will not have much effect on the gradient.

The phone is intended to be mounted on a platform that holds it securely in place. It should not be prone to falling down and it should be strong enough to hold onto the phone even if the car quickly changes speed or direction. It should also be strong enough to hold onto the phone in case the user is in an accident, to avoid the phone becoming a projectile. The Android device should also be mounted in such a way that the user is able to see the device without looking directly at it or turning their head. It is up to the user on how they place the Android device, but the group is not held accountable for the user choosing a poor location. It is a similar situation to positioning the Android device for using GPS directions, however the user needs to be able to view the application with their peripheral vision or else the application cannot be used appropriately. This is because the fuel efficiency part of the application relies on the user being able to see the application without looking directly at it, which is why distinct colors are chose and fill the whole screen. This also allows the user to utilize the images for sensor detections because the images contrast greatly with the backgrounds.

The application allows for blind spot detection while driving. To activate blind spot detection, the user needs to turn their wheel to the appropriate angle as mentioned in previous sections. In order to avoid distracting the user while they are driving, blind spot detection will be done via audio notifications by default. Because it is possible to disable audio notifications, or because the user may be deaf, it will also display an arrow with a cross through it if it is not okay to change lanes. A different audio notification also plays if the user is too close to another object that is in front of their vehicle. A message saying “TOO CLOSE” will be on the screen as well, for the same reasons as mentioned previously.

4.1.6 Audio Notifications

Audio notifications are used either when the user is trying to switch lanes or when the user is driving too close to an object that is in front of their vehicle. If the user is trying to switch lanes, it will be activated as soon as the user turns their steering wheel either between -110 and -45 degrees for a left turn or 45 and 110 degrees for a right turn. In order to not be distracting to the user, it will produce a beeping noise similar to the native noise that turn signals make when they are activated. However, the noise differs enough that the user will know whether or not it is safe to turn. If nothing is detected in the blind spot, no noise will be played. However, if something is detected in the blind spot, the noise is activated in order to alert the user that it is not safe to switch lanes. The audio notification will only take into account the blind spot that is on the side of the car that the turn signal is set to. For example, if the wheel angle is between -110 and -45 degrees, it is considered “LEFT,” and the audio notification will only take into account the sensor that is on the left side of the car. If there is a car in the right

blind spot but no car in the left blind spot, the user would receive a notification indicating that it is safe to turn.

If the user has deactivated the noise notification from the options menu, then no noise will occur while the turn signal is activated or while an object is too close to the front of their vehicle. However, they are still able to make use of the blind spot detection by looking at the large picture on their phone, but they will not be able to hear any notification. Pictures only occur if the noise would be playing, which means that the user does not have to worry about mistaking a safe image for a not safe image. They can still view if it's safe out of their peripheral vision.

There are only two noises that are ever played to the user. These are the "too close" noise and the "blind spot" noise. Because the user will be able to tell through practice, or by glancing at their screen, which noise represents which message, it is not necessary to allow the user to test the sounds on the application beforehand. There also aren't any audio choices for the user because the sounds were chosen to be simple and unique. Allowing further options could confuse the user.The available sounds were chosen such that they will not potentially startle the person who is driving. They are also natural and helpful rather than loud and potentially surprising. They do not mimic actual emergency vehicle noises, so as to not confuse the user into thinking there is an emergency vehicle nearby when there is not. Due to the length of time that the sound will need to be played varying per each situation, the audio files are relatively short and will repeat for the full time that the sensors are detecting the objects. The length is actually 500ms even though the clock cycle is 50ms, but this is okay because the user needs to be notified for an extended period of time if there is something in their blind spot or if they are too close to the vehicle in front of them. Also, it prevents the noise from being too obnoxious because it is repeated less frequently.

All sounds used in the application were received from an open source location. Short sound bites were retrieved and further edited to make them even shorter than they were originally.In accordance to the terms of service in the open source location, it is not necessary to credit the location of the sounds within the application. The graphics used, however, were created originally by the group and thus no credit is needed.

4.1.7 DMS Toast Generation

DMS will generate Android toasts, detailed in section 3.4.4.5, which are essentially small pop up messages that stay for a set period of time then fade away, leaving the current activity on the device uninterrupted. The toasts that are generated for DMS will provide suggestions to the driver based on how they are doing during their current driving session. DMS will also generate toasts that accompany verbal alerts for the sensors installed on the vehicle. If there is a blind spot detection alert, then there will be a toast to the screen to alert the

driver to the situation. The toasts will have a custom layout for each type of toast so that the driver can distinguish between suggestions and alerts. The custom layouts for the two types of toasts are displayed in Figure 21. If the toast is for an alert it will have yellow background with stop signs on both sides, the text will be black and bold to alert the user. If the toast is a suggestion to improve a driver's driving habit, the toast will be gray and white with a frown face. If the toast is to give the driver positive feedback it will have a similar look to the suggestions but will be accompanied by a smiley face.

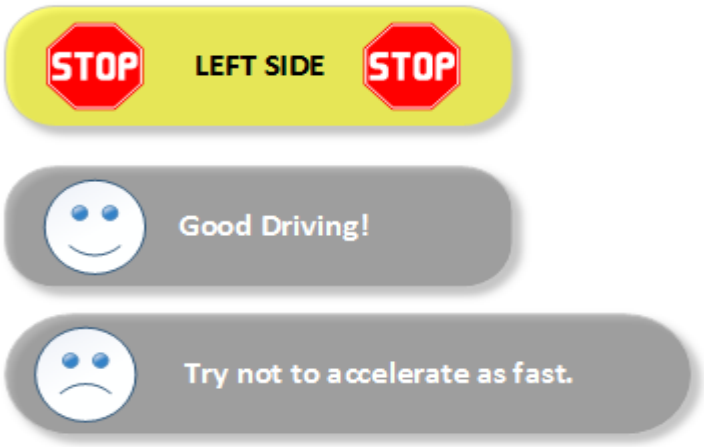

Figure 21: DMS custom layout toasts

The suggestion toasts will be generated based on the driver's habits during the current driving session. To accomplish this the driver's habits will have to be recorded. To monitor the driver's habits, each criteria will have its variables examined, if the variables lie within ranges that are determined to be detrimental to driving economy then the habit will be recorded. The different criteria and the associated vehicle attributes are shown in table 12.

Driving Habit	**Associated Vehicle Attributes**
Over Acceleration	Torque at transmission, Accelerator pedal position, Engine speed.
Breaking to often	Accelerator pedal position and Brake pedal status
Speed over 70 mph	Vehicle speed
Idling longer than 60 seconds while in park or neutral	Transmission gear position
Frequent Lane Changes	Steering wheel angle

Table 12: Driving Habit and Associated Vehicle Attributes.

The driving habits are monitored using the combination ranges and thresholds for the various vehicle attributes associated with the driving habit. If the values fall within ranges deemed unacceptable for economic driving the infraction will be stored and used to decide what suggestions will be generated. The generated suggestions for each monitored driving habit are available in Table 13. The suggestions are designed to inform the driver on how to correct the bad habits they are exhibiting often.

Driving Habit	**Suggestion**
Over Acceleration	“Try not to accelerate as fast to increase your gas mileage”
Breaking to often	“Breaking less frequently will increase your gas mileage!”
Speed over 70 MPH	“Excessive speeding costs you more in gas money.”
Idling longer than 60 seconds while in park or neutral	“Idling for longer than 60 seconds is bad, if you plan to be in your car for a while, try shutting it off to save gas.”
Frequent Lane Changes	“Try not to change lanes so frequently.”

Table 13: Driving Habit Suggestions.

In order to determine what suggestions need to be generated as toasts, driving data is collected, and compared against calculated conditions, ranges and thresholds for each bad habit, if the driver falls with any of the bad habits, the habit will be recorded in a list for that driving session. The list will be a data structure that is created upon the start of each driving session. As the user drives, if any bad habits are exhibited the list will be updated and the infraction will be recorded. The list is needed to generate the suggestions which will be chosen based on the bad habits the driver commits most often. The committed bad habits will be displayed only when the vehicle is not currently moving, and will be sent from a queue where the queue is generated from most committed offence to least committed offence, but only committed offences will be put into the queue. At the end of each driving session the list will be saved for long term analysis, but the queue will be reset. To ensure that the queue does not always bump the most committed offence to the top of the queue because of priority, if a most committed offence has been displayed in a diving session its priority will be reduced. In this way the committed offences will generate toasts each time the driver stops, and the suggestions will be cycled through so that the driver doesn't get the same suggestion over and over, but the driver may see the same suggestion more than once if he/she keeps committing the offence because it will be added to the list, and be assigned a place in the queue. The process for generating the bad habit list in detailed in Figure 22.

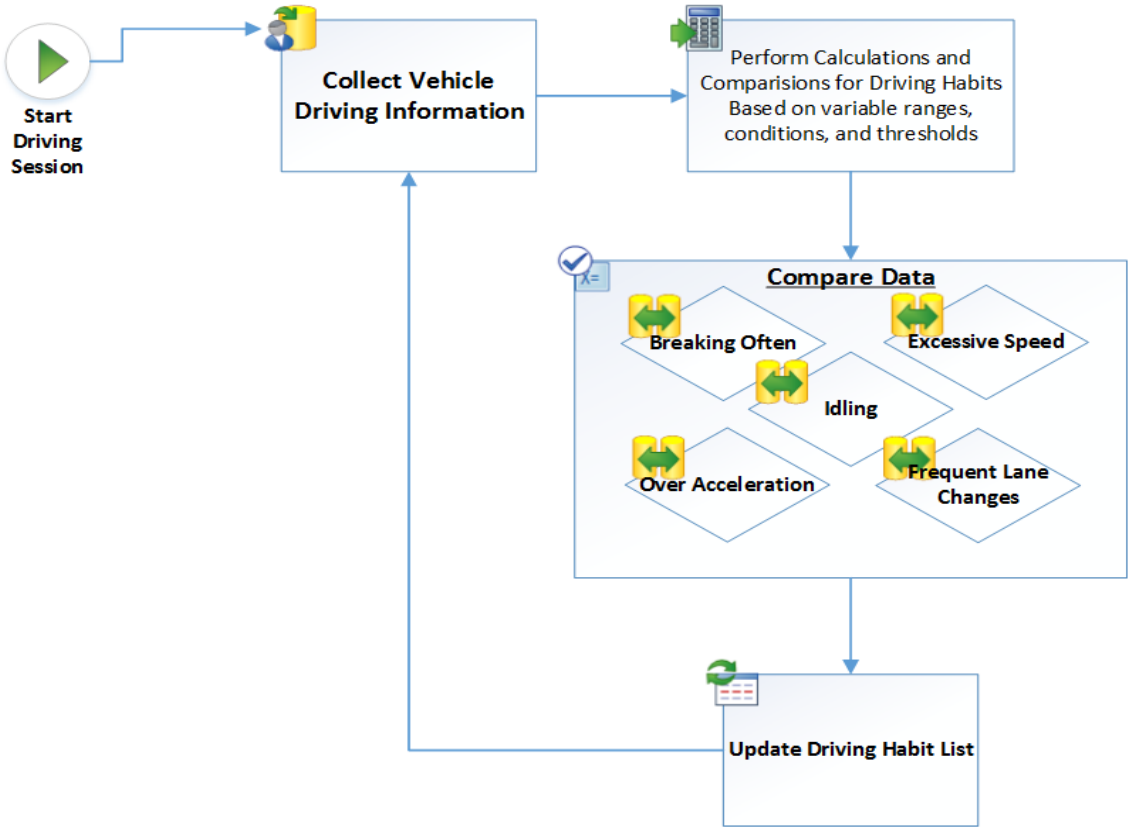

Figure 22: Process for updating Driving Habit List

As detailed earlier in this section, DMS will contain other toasts besides the driving suggestions. DMS will also provide toasts for the sensors. The toasts for the sensors will be activated when the conditions for the sensor activating are met and the sensor responds with a result indicating an object is in the way of the vehicle. The toast will simply indicate what side of the vehicle is on and warn the driver to avoid any actions in that direction. For example, if the left blinker is on and the left blind spot detection circuit sends a positive to the phone indicating that there is something in the blind spot, the toast will pop up on the screen informing the driver that there is something in the left blind spot.

4.2 Hardware Design

4.2.1 Vehicle Interface

4.2.1.1 Overview

A vehicle interface module connects a vehicles onboard diagnostic port to a computer and transfers vehicle data to the user. Information that a vehicle interface could send from the car to a computer includes, but is not limited to,

steering wheel angle, vehicle speed, brake pedal status, accelerator pedal position and fuel status.

The Ford OpenXC vehicle interface is a custom made OBD-II reader which has access to information that is specific to each Ford model. It uses a dual-channel CAN transceiver to read information from the vehicle. The vehicle interface is designed to be compact so that it can be used while driving without hitting the driver's knees. Features that are included onto the vehicle interface include Bluetooth for wireless data transfer and a 12 V port to power additional devices. The vehicle interface connects to the vehicles OBD-II port and transfers data to a computer or android device wirelessly. This eliminates the need for cable connections to the vehicle and to the computer being used. An image of the vehicle interface can be seen in the following image (Figure 23).

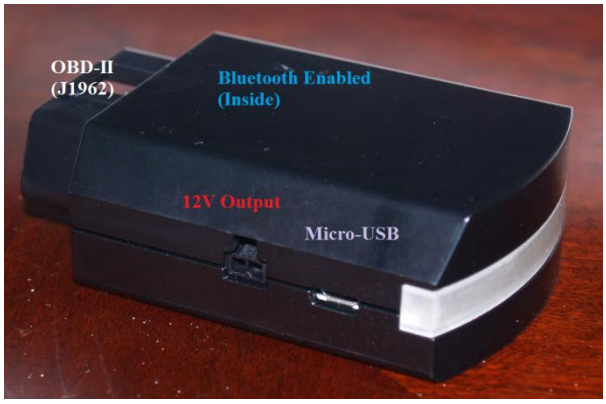

Figure 23: Vehicle interface module that was provided by Ford.

The vehicle interface used in the project was provided to us by Ford. The vehicle interface contains a Bluetooth module within it. This allows for wireless communication to the users android device or pc. This gets rid of the need for a USB cable to attach to other devices. The vehicle interface uses a NXP LPC1768 microcontroller as the core of their hardware design. Ford chose this microcontroller due to it having many available products available to the community. This allows for a large amount of community driven Ford projects. The Vehicle Interface takes in a 12V input from the vehicles battery. Two low dropout linear voltage regulators are then used to provide 5V and 3.3V for the microcontroller and peripherals within the Vehicle Interface. A push button is used to program the microcontroller. The button is held down using a pin. After a certain amount of time a green LED within the Vehicle Interface will light up. This means that the Vehicle Interface is ready to be programmed using the firmware for the Ford Focus.

4.2.1.2 OBD-II Port Specifications

The on board diagnostics system (OBD-II) allows someone to gain access to specific information about the vehicle. The OBD-II port provides power for the attached vehicle interface modules so that no additional power source is

required. The OBD-II system consists of 16 data buses, each providing unique information on how the vehicle is functioning. The following table contains a list of pins from the Ford Focus OBD-II port that will be used.

Pin	Signal	Description
3	LS CAN High	Low speed (125Kb) CAN bus
4	CGND	Chassis Ground
5	SGND	Signal Ground
6	CAN High	J-2284
7	K-Line	ISO9141-2 and ISO/DIS 14230-4
11	LS CAN Low	Low speed (125Kb) CAN bus.
14	CAN Low	J-2284
15	ISO 9141-2 L-Line	ISO9141-2 and ISO/DIS 14230-4
16	+12V	Battery power

Table 14: Table of pins from the OBD-II port that will be used by the Vehicle Interface.

The vehicle interface will be plugged into the OBD-II port to access the available information. Most of the vehicle information is available through CAN1 (pins 6 and 14) which the vehicle interfaces directly plugs into. In addition to the OpenXC signals that can be obtained from the vehicle (Table 14), the OBD-II also provides drivers with the following standard information:

- Battery Voltage
- Air Intake Temperature
- Check Engine Light Codes
- Timing Position
- Transmission Temperature

4.2.2 Power

The peripherals of DMS will run off a 9V lithium ion battery. The battery is rated at 550mAH which is enough to run each device. Since there will be three circuits that require power separately it will be necessary to purchase three batteries. The cost of these batteries range anywhere from 5-10$. These batteries are also

rechargeable in case they lose charge. Typically the 9V lithium ion batteries are found in low power consuming devices such as clocks and smoke detectors. Since all of the components used within the project consume low amounts of power it is possible to use the 9V lithium ion battery. The devices will also be in low power modes a majority of the time while the vehicle is in use which will also lead to an increased battery lifespan. To maintain a constant supply voltage for the sensors, microcontrollers and Bluetooth module a few LM1117 voltage regulators will be used. The LM1117 voltage regulator is made by Texas Instruments. It is a linear voltage regulator that is available in multiple models with different output voltages. There is also an adjustable version that can set a regulated output voltage from 1.25V to 13.8V. An Eagle schematic of the LM1117 can be seen below in Figure 24.

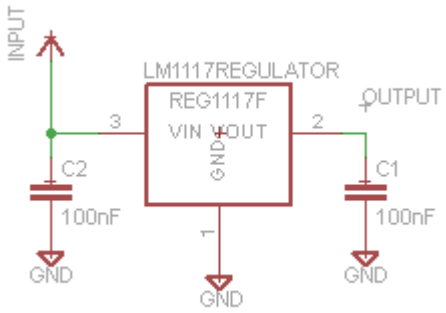

Figure 24: Eagle schematic of the LM1117 voltage regulator.

4.2.3 Microcontroller

The MSP430G2553 was chosen as the microcontroller to implement both the blind spot and collision detection systems. It is a great option for the price Due to having multiple components with separate microcontrollers the low cost of the MSP430 was a large deciding factor. The MSP430 is not powerful enough to run a Bluetooth stack while operating as normal. This will require the addition of a Bluetooth module. The Bluetooth module can be connected to the MSP430 via a UART connection provided by the MSP430. An analog to digital converter is also provided by the MSP430 which can be used to input any analog signals from the sensors that may be used. It will then convert the analog signal into a digital signal which can be used as a reference to determine what needs to be done with the data that has been received. The MSP430 will be powered by a 3.3V regulated power source. While in an active mode the MPS430 will require 330µA on average. The MSP430 offers an active mode and four low power modes which is great for the project. It can be programmed to stand by in any of the four low power modes until an interrupt signal is received. Although a majority of the time the vehicle is in use the microcontrollers used will remain in Low Power Mode 4. While in this low power state the microcontroller only draws about 0.8µA. This is incredibly low and helps to prolong battery life. The following Figure 25 shows a graph of the MSP430's current consumption in Low Power Mode 4.

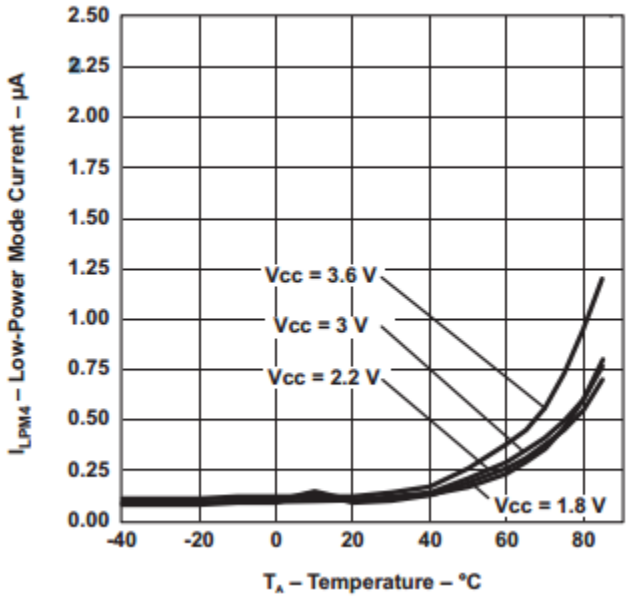

Figure 25: Graph of the supply current while the MSP430 is in Low Power Mode 4. Permission provided by Texas Instruments.

4.2.4 Bluetooth Communication

The wireless communications between the DMS peripherals and the user's android device will be done through a Bluetooth connection. The Bluetooth enabled devices will all be synced to the android device. Each microcontroller will be kept in low power mode 4 and will not be set active until an interrupt signal is received. The android device will then be used as the main medium for communication. When the android device receives the appropriate signal from the Vehicle Interface, an interrupt signal will be sent to the corresponding microcontroller. For example when the left turn signal is in use, the left blind spot detector's microcontroller will be activated. This way there will be no interference between multiple Bluetooth channels and will allow the android device to directly communicate with each component.

The Bluetooth module chosen for the project is the RN-42. The RN-42 is a low power, class-2 Bluetooth module. It allows for the wireless communication aspect of the project to be implemented in a cheap and efficient manner. The RN-42 includes a high performance on chip antenna and comes with everything needed for the Bluetooth protocol. This is useful because the MSP430 microcontroller would not be able to run a Bluetooth stack. In keeping with the low power consumption goal of the project, the RN-42 has a sleep mode. This sleep mode greatly decreases the power consumption of the device while it is not in use. The RN-42 will allow data to be transferred between the android device and the microcontroller. Both the blind spot detection circuit and the collision detection circuit will use the RN-42 for wireless communication. The following

table provides specifications of the RN-42 and the following image contains the RN-42 pinout.

RN-42 Bluetooth Module	
Bluetooth Version	Class-2 Bluetooth 2.1+ EDR
Frequency Range	2.405-2.48GHz
Supply Voltage	3.0 - 3.6V
Data Rate	Slave: 240 Kbps Master: 300 Kbps
Power Consumption	Active: 40mA Sleep: 26µA

Table 15: Table of the RN-42 Bluetooth Module specifications.

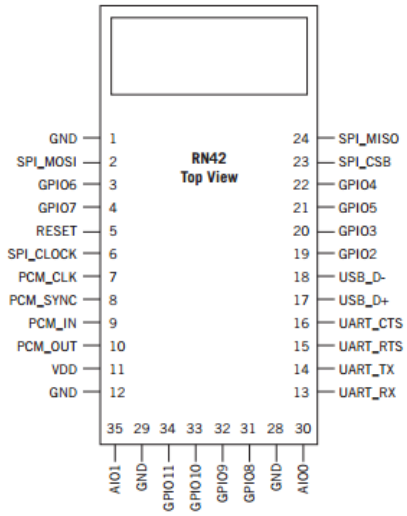

Figure 26: Pinout of the RN-42 Bluetooth module. Permission provided by Microchip.

The Bluetooth module will run off the same 3.3V regulated power source as the MSP430. While the device is active it will draw 40 mA and while it is inactive it will only draw 26µA which will improve the lifespan of the battery being used. First the Bluetooth module must be made compatible with the MSP430 and the Android device being used by the driver. To do this the baud rate will be set to 9600 using 8 data bits with 1 stop bit and no parity. A Bluetooth module will be connected onto both blind spot detectors and the collision detection systems. To connect the Bluetooth module to the MSP430, pin 14 (UART_TX) of the RN-42 will be connected to pin 3 of the MSP430. This will enable the transmit data line of the Bluetooth module. Similarly, pin 13 (UART_RX) of the RN-42 will be connected to pin 4 of the MPS430. This will enable the receive data line of the

Bluetooth module. For testing purposes a LED will be used to determine if the Bluetooth connection has been established. For the final product there will be no LED on the circuit. The Eagle schematic of the Bluetooth connection can be seen below in Figure 27.

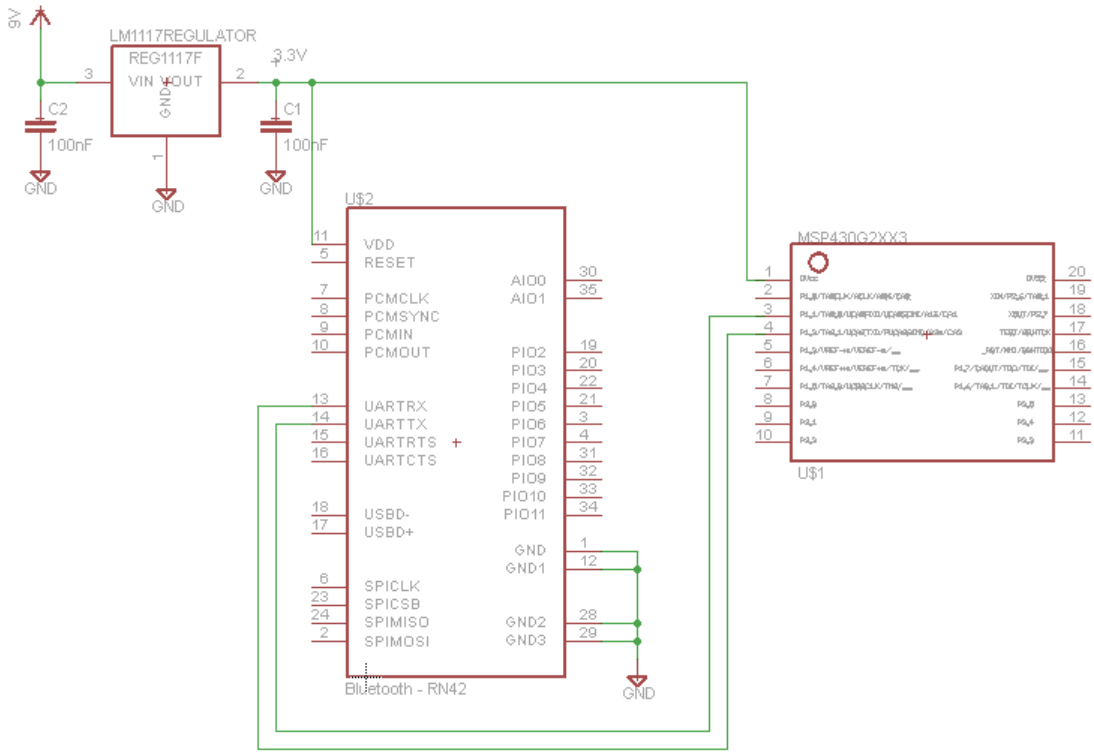

Figure 27: Eagle schematic of the Bluetooth connection between the RN-42 and MSP430.

4.2.5 Blind Spot Detection

4.2.5.1 Overview

The blind spot detection systems main goal is to alert drivers when other vehicles are located on the rear or sides of the vehicle. Blind spots are areas around a vehicle that cannot be seen by the driver in either the rear view and side view mirrors. The side-view mirrors can be adjusted to greatly reduce the area of the blind spot, although this does not eliminate the blind spot entirely. Each year there are more than 800,000 blind spot related accidents. The use of a blind spot detection system would greatly reduce the number of related accidents. Different factors may affect the size of a vehicles blind spot. These factors may include the size of the driver, the size of the vehicle, and the time of day. Since a blind spot may change in size between vehicles, the DMS blind spot detection can be moved around by the driver to fit their needs. The figure below (Figure 28) depicts the area of a driver's blind spot.

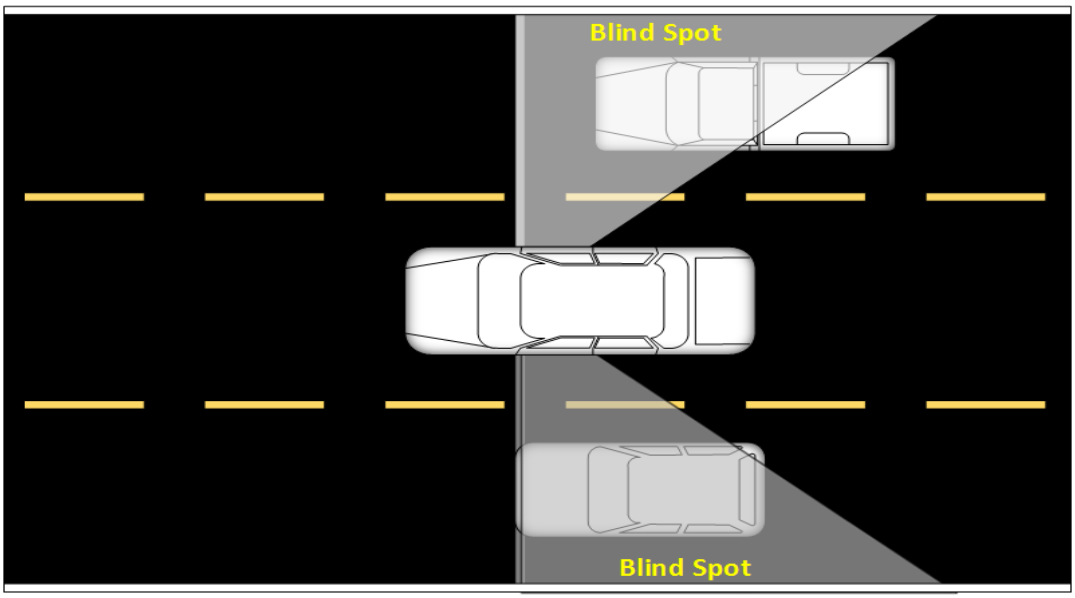

Figure 28: Diagram depicting where blind spots can occur while driving, pending permission from crimson concrete.

The blind spot detection system will work off the status of the vehicles turn signal. The blind spot detection system relies on the driver's Android device to relay information from the vehicle. When the driver uses their turn signal the android device will alert the blind spot detection that the driver is possible changing lanes. If the right turn signal is on, the blind spot sensor on the right of the car will be activated. The same is done for the left turn signal. When the blind spot sensor is activated, any motion within the blind spot will be seen by the sensor. If motion is detected the blind spot system will relay the information back to the driver's Android device. The Android device will then alert the driver will a sound alert and image.

4.2.5.2 Sensor

For the blind spot detection portion of the project, a microwave sensor was chosen due to the flexibility it offers over the other types of sensors. Microwave sensors have the ability to sense motion through objects and are not affected by weather and driving conditions, unlike ultrasonic and PIR sensors. Ultrasonic sensors were not chosen for this portion of the project because at high speeds the readings would not be accurate. Wind and noise while the vehicle is in use affects the detection range of the ultrasonic sensor. If the initial design decision of using a microwave sensor does not work, PIR sensors will be used as an alternative. Since the microwave sensor has the ability to detect motion through walls and certain objects the option is available where it could be placed within the vehicles trunk or rear bumper. This removes the need for weatherproof casing that would protect the circuit from harsh environment conditions. Having the blind spot sensors within the vehicle also keeps the DMS from affecting the look of the vehicle. If the lithium ion batteries were to fail it is still possible to power the blind spot detectors. Powering the blind spot detectors can also be

done within the vehicle, through the fuse box located in the trunk. Placing the sensors within the car will be done if the original placement of the sensors on the vehicle does not work.

The HB100 microwave sensor was chosen as the main sensor for the blind spot detection system. It is designed by ST electronics, although there is not much information available on it. The HB100 uses a 10.525GHz RF signal to determine if an object is moving. The output is very small and requires an amplifying circuit to boost the signal. Typically the output signal is also very noisy so a filtering circuit is also required to filter out frequencies that are not wanted. It was chosen as the sensor for the blind spot detection system due to its low cost. Typically each sensor falls in the range of $5-10. The transmit frequency and power is preset by the manufacturer and cannot be adjusted by the user. The output signal of the HB100 is labeled as the IF signal. This stands for Intermediate Frequency. The Intermediate Frequency is the combination of the transmitter frequency and the reflected signal from objects in motion. This allows for the Doppler Shift value to be determined. The microwave sensor uses the Doppler Shift output from the IF terminal to determine when there is motion within the range of the sensor. The HB100 is a low current consuming sensor with a long detection range. The following table contains specifications of the HB100 microwave sensor module.

Parameter	Typical Value
Frequency	10.525 GHz
Settling Time	3 µs
Supply Voltage	5 V
Current Consumption	30 mA
Receive Signal Strength	200 µVpk-pk

Table 16: Specifications of the HB100 microwave sensor module.

4.2.5.3 Amplification Circuit

The HB100 microwave sensor module does not come with an integrated amplifier. It also provides an output amplitude that is very small which usually lies in the µV range. This is why an amplification circuit with a very large gain is required. A very large gain is also required since the sensor will be used in a long range application, therefore the IF signal will not be very strong. The amplification circuit consists of two cascaded non-inverting summing amplifiers. Each stage has a total gain of 100. Total gain is calculated as the product of each stages gain, therefore this amplifier provides a total voltage gain of 10000. Due to the gain of this amplifier being so high, the supply voltage noise is visible at the output. To reduce the amount of noise seen at the output of this amplifier, linear

voltage regulators will be used instead of switching regulators. This is because the noise caused by the switching regulators is much greater than the linear regulators and may interfere with the output. The high gain of the amplifier will allow the microcontroller to receive a signal that is large enough to be used. The schematic of the amplifier can be seen in Figure 29.

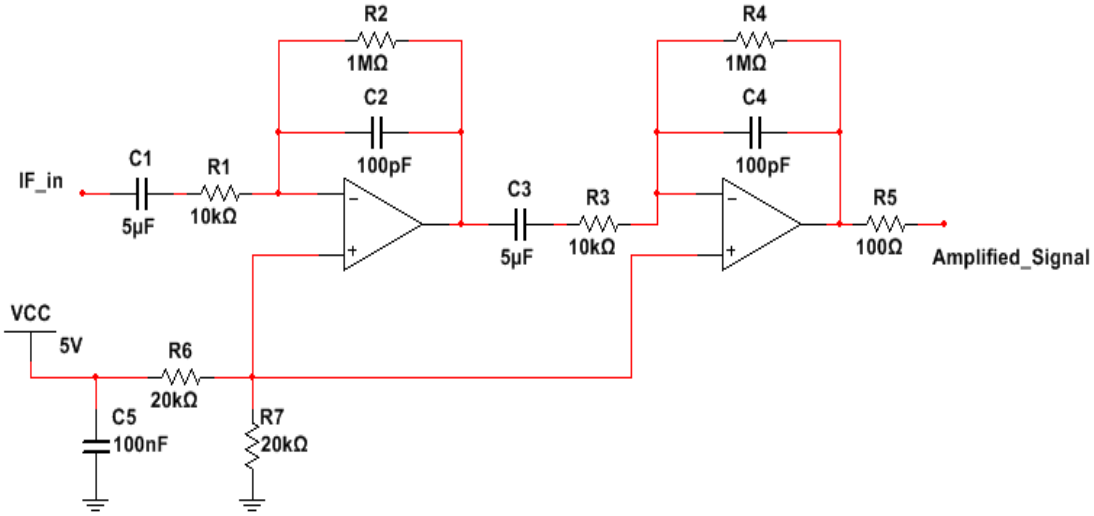

Figure 29: Schematic of the amplifier that will be used to boost the microwave sensor's output signal.

Although the microwave sensor requires an amplifier with an extremely high gain. The resistor values of the amplifier cannot exceed 1 MΩ. This is due to input bias current increasing as the resistor values increase. When the input bias current becomes too large it will affect the op-amps output voltage. This will lead to inaccurate readings on the output that is connected to the microcontroller. LMV772 op amps were chosen to be used within the amplifying circuit. The LMV772 is a low noise and low offset voltage precision op amp.

4.2.5.4 Overall Blind Spot Circuit

The overall blind spot detection system consists of a HB100 microwave sensor, MSP430 microcontroller, amplifying stage and an RN-42 Bluetooth module. Since the blind spot detection system will be looking for motion within a rather larger area of 10 meters. The microwave sensor requires that its output signal is amplified. The further a vehicle is from the sensor the smaller the signal will be from the HB100 microwave sensor. This is why the amplifying stage with a very large voltage gain of 10000 was used. It is made up of two summing amplifier stages, each with a voltage gain of 100. This will boost a 100µV signal to 1V which will allow for the MSP430 microcontroller to interpret the signal. The output of the amplifying stage is connected to pin 10 of the MSP430. A schematic of the overall circuit can be seen below in Figure 30A.

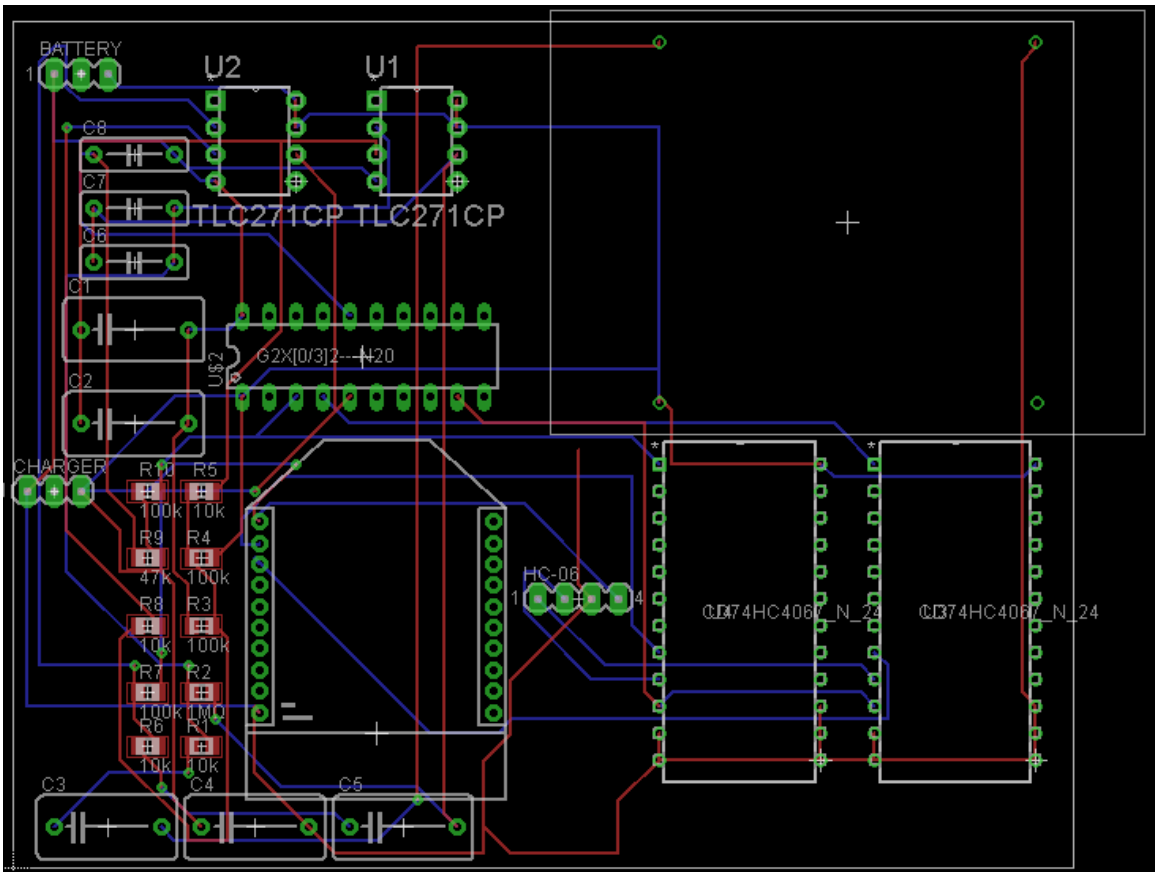

Figure 30A: Eagle schematic of the overall circuit for the blind spot detection.

4.2.6 Collision Detection

4.2.3.1 Overview

The goal of the collision detection component of the project will be to alert the driver when the optimal opportunity to begin braking occurs. This portion of the project will use a sensor located on the front of the car to determine the distance of objects located in front of the vehicle. The distance between the driver's vehicle and the object located directly to the front will then be used to determine when to begin braking. According to the National Highway Traffic Safety Administration 28% of accidents are rear-end collision accidents. Being alerted when to brake in situations where the driver may or may not be paying attention to the road could lower the number of rear-end collision accidents. By preventing a potential accident the DMS would be saving the driver money by avoiding accident costs and preventing possible accident related injuries.

4.2.3.2 Sensor

The collision detection portion of the DMS will use an ultrasonic sensor to determine the distance between the driver's vehicle and the vehicle directly in front of the driver. The ultrasonic sensor was chosen to implement this part of the

project due to its ability to accurately determine the distance between itself and another object. Since the collision detection portion of the DMS is only searching for vehicles located directly in front of the car the ultrasonic sensor will not have to be placed at an angle. This reduces the chances that the ultrasonic sensor may cause false positives. An ultrasonic sensor was chosen over a PIR sensor for this portion of the project due to needing to be able to measure distance. Since the ultrasonic sensor uses the traveling of sound waves, it is possible to determine the distance of an object based on the time it took for the signal to be transmitted and received. Since the PIR sensor uses a constant infrared beam it is convenient for motion detection but not as reliable when determining distances.

The HC-SR04 ultrasonic sensor was chosen as the sensor to implement the collision detection system. It is a popular ultrasonic sensor device that is compatible with the MSP430 microcontroller used throughout the project. The device includes the ultrasonic transmitter and receiver together in one package. The sensor works by transmitting multiple 40 kHz signals and looks for an echo. If the targeted object is located directly in front of the sensor, the distance of the targeted object can be determined by taking the time it took for the transmitted signal to return and multiplying it by the speed of sound in air. The specifications for the HC-SR04 ultrasonic sensor module can be found in the following table.

Parameter	Typical Value
Working Frequency	40 Hz
Working Distance	Max: 4 m Min: 2 cm
Supply Voltage	5 V
Current Consumption	15 mA
Measuring Angle	15 degrees

Table 17: Table of specifications for the HC SR04 Ultrasonic module.

4.2.3.3 Overall Circuit

The overall design of the collision detection system is similar to the blind spot detection system. The main difference is caused by the collision detection system measuring distances while the blind spot detection system monitors motion. It includes a 9V lithium ion battery, a low power MSP430 microcontroller, the HC-SR04 ultrasonic sensor for measuring distances and an RN-42 Bluetooth module for wireless communication between an Android device and the MSP430 microcontroller. The HC-SR04 will be directly connected to the MSP430. Pin 13 of the MSP430 will be connected to the TRIG pin of the ultrasonic sensor and pin 12 of the MSP430 will be connected to the ECHO pin of the ultrasonic sensor. In

Figure 30B an Eagle circuit schematic of the overall collision detection system can be seen below.

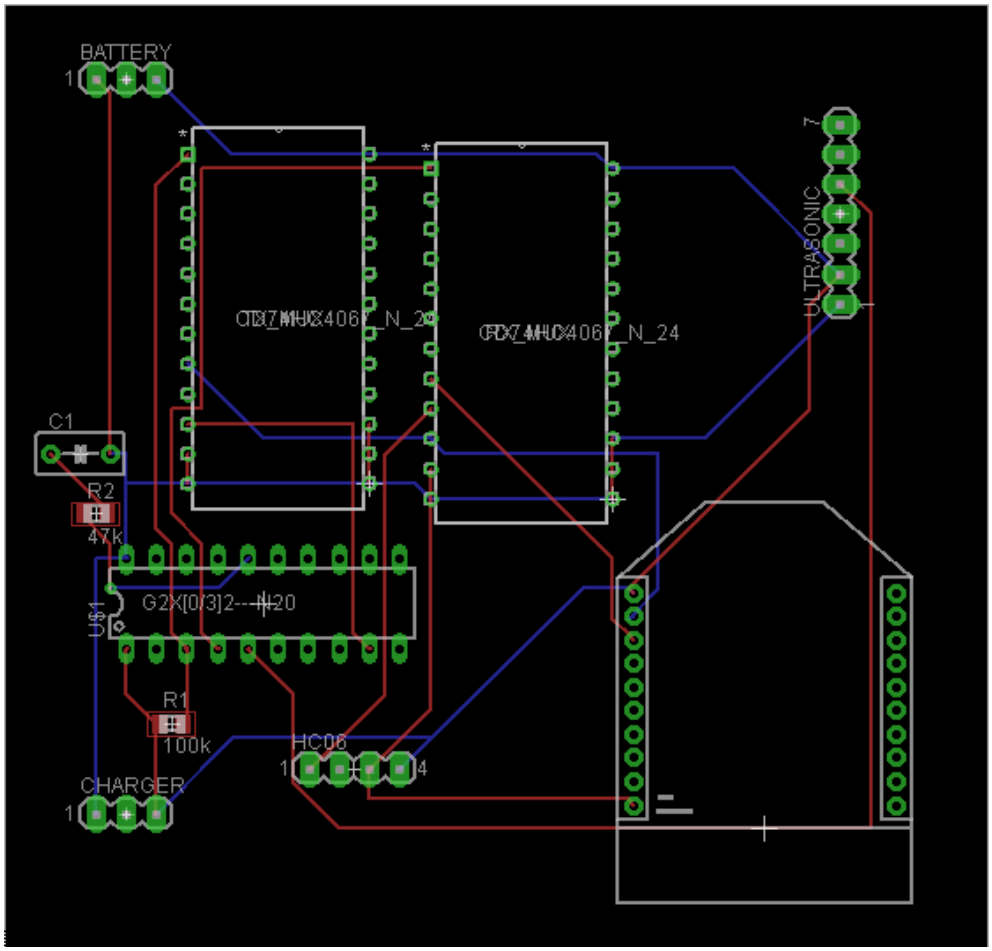

Figure 30B: Eagle schematic of the overall circuit for the collision detection.

4.2.7 Rear View Camera

4.2.4.1 Overview

A camera will be positioned at the rear of the car to assist the driver while backing up. The main goal of this will be to prevent the driver from hitting another vehicle or striking a pedestrian. Many accidents that take place while backing up occur in driveways and parking lots, with most involving children. For this reason the U.S. government has been planning on making rearview cameras mandatory in newer pedestrian vehicles. Although many options available to consumers are expensive, the rearview camera portion of the project will be designed to be affordable.

The rearview camera will be connected to the Android device at the front of the car. The vehicle interface will then read the gear position of the car to determine when to display the video to the driver. When the driver puts their car into reverse the video feed will be displayed to the driver. After the driver backs up and brings the vehicle into drive the video feed will cut off.

Although it is not ideal, for this project the camera will be connected through USB to the Android device. This will require wiring the vehicle in order to avoid having exposed wire running from the back of the vehicle to the front of the vehicle. The camera will be mounted in such a way that it is possible to view a large portion of the area behind the vehicle that is not normally visible, while insuring that the driver's vision is not obstructed. Using a USB device will allow for a reduced cost due to being able to use a USB connected camera rather than a Wi-Fi enabled camera. Because the camera will need to be connected with a wire, the rear view camera will be considered an optional component that the user can disable if they cannot or do not want to hook up the camera.

The camera that will be used will be under $40 and have decent video quality. It should work in daylight or at nighttime. It does not have to have the best quality because the main purpose for it is for the user to make sure they are not about to run into someone or something behind them, but it should be high quality enough that the driver is able to identify objects behind their vehicle. An option for the webcam is the Logitech HD Webcam C270 or Logitech HD Webcam C310. They are both approximately $30 but may be acquired for as little as $14 depending on the source and whether or not it is used or refurbished. Logitech Webcams tend to be Android compatible so it should be able to be connected via USB to the android device without much issue. The Logitech HD Webcam C270 will be used over the C310 due to a slightly cheaper price without a diminished quality.

4.2.8 Casing

Both the blind spot and frontal collision detection circuits will be placed outside of the vehicle. This will add the requirement of needing a case to protect the circuit. The circuit will be exposed to different types of weather and driving conditions such as rain, fog, and high winds. The blind spot detection system will be completely enclosed in a plastic case. This can be done due to the blind spot detection system using a microwave sensor. Since microwaves can travel through certain objects it is possible to completely enclose the circuit which will protect it and still allow it to maintain functionality. The blind spot detection system will be placed on the rear side panel of the 2013 Ford Focus. To maintain a stock vehicle appearance it is important that the case is not large enough to take away from the vehicles appearance. It must also not stand out in a way that it may take another driver's attention off of the road. To achieve this the case will have a color to match that of the vehicle being used. In this case the vehicle being driven is a silver color, therefore we will use a silver casing to cover the blind spot detection system.

The collision detection circuit differs from the blind spot detection circuit because of the sensor being used. The sensor used for this circuit is an ultrasonic sensor. It is not possible to completely encase the ultrasonic sensor, since it relies off bouncing a signal off of the target. Due to this the casing will have two holes cut

out in the front. This will allow the ultrasonic sensors transmitting and receiving ends to be outside of the case. The sensor will then be able to transmit and receive a signal. Doing this the circuit is also protected from weather or driving conditions that may affect it. The circuit will be placed onto the grill of the car. Since the grill of the car is black, a black casing will be used to enclose the collision detection system. This will prevent it from taking away from the look of the car. The rear view camera will be placed near the license plate of the vehicle. It is positioned in a way that the driver can see as much as possible behind their car. Since the camera is already encased all that is needed is a thin screen cover to protect the lens of the camera from any hazardous conditions.

After creating each casing for the circuit, a magnet will be attached to one end. The magnet will be used to attach the DMS devices throughout the car. Since a magnet is being used it is possible to move the devices wherever is needed. This also allows the driver to easily remove these devices when they are not working or they simply do not want them on their vehicle at the time. This also will prevent damages to the paint of the car, since using an adhesive type of material could rip off paint. The dimensions of each case can be seen in table 18 below.

Circuit	**Dimensions**
Blind spot detection casing	2.5 x 2.5 x 0.6 Inches
Frontal collision casing	2.5 x 2.5 x 1 Inches

Table 18: Table of case dimensions.

5.0 Design Summary of Hardware and Software

5.1 Fuel Efficiency Summary

Fuel efficiency is displayed on an Android device screen while the user is driving their vehicle. It is represented as a gradient between three different colors, depending on the settings chosen by the user, allowing for the user to have an idea of how their driving behaviors are affecting their fuel efficiency. The device should be placed in such a way that the user will be able to view it from their peripheral vision, so that they do not need to take their eyes off the road. The calculations to determine how fuel efficiently the user is driving will take into acceleration and deceleration. The program also measure how hard the user is accelerating and braking, whether or not they brake too soon after accelerating, if they speed over 70mph, and how frequently they idle over 1 minute, to display this information to the user at a later point in time. All information is gathered through the vehicle interface and analyzed in the application in real-time. New

information is gathered every 50ms in order to insure that the fuel efficiency display is up-to-date and accurately represents the current driving habits of the user. However, when the fuel efficiency information is stored and transported to graphs inside the application, the data will not display per each second. A more reasonable time frame based on how long the driving session occurred is chosen and an average of the data gathered within that time frame is shown instead.

5.2 Blind Spot Detection Summary

The blind spot detection system will be made up of two components, one for the left side and another for the right side of the car. Each circuit will consist of a microwave sensor, MSP430 microcontroller, a RN-42 Bluetooth module and a lithium-ion battery. The circuit will be placed in a case that will protect the circuit from harsh weather conditions and debris from the road. Each component will be placed onto the rear side panel of the car using a magnet. This will keep the blind spot detection system from falling off while the vehicle is in motion. A block diagram of the blind spot detection system can be seen in Figure 31.

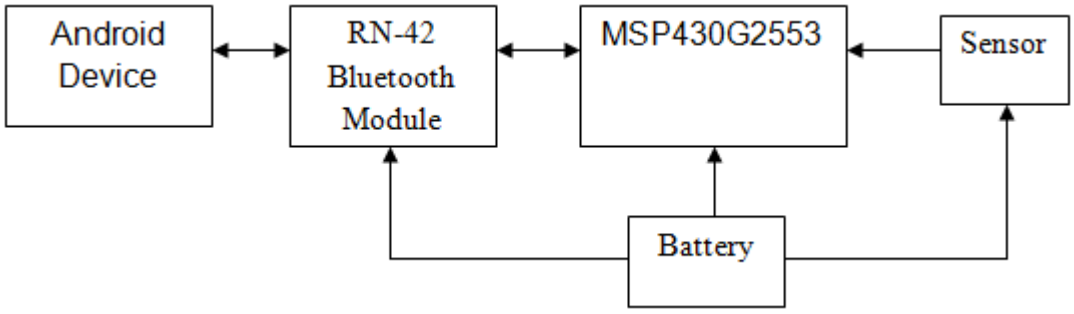

Figure 31: Block diagram of the blind spot detection system.

The blind spot detection system will not be activated until the turn signal of the car has been used. When the driver uses their turn signal, the phone will then activate the blind spot detection system. The MSP430 will be running in low power mode, to limit power consumption, until an interrupt signal is sent from the driver's phone. While the driver's turn signal is on, the microwave sensor will detect any object located in the driver's blind spot. When the turn signal is turned off the MSP430 will return to low power mode and the blind spot detection system will become inactive. This feature can be turned on and off at the drivers discretion.

5.3 Collision Detection Summary

The collision detection system will be located at the front of the car. It will be placed at the Ford Focus' grill. Similar to the blind spot detection system the collision detection circuit will consist of a MSP430 microcontroller, a RN-42 Bluetooth module, lithium ion battery and an ultrasonic sensor. The block

diagram for the collision detection circuit is similar to the one in Figure 31. The only difference is that an ultrasonic sensor is used for this portion of the project instead of a microwave sensor. The circuit will also be protected by a case that can withstand debris from the road and harsh weather conditions. Unlike the microwave sensor the signal sent from the ultrasonic sensor cannot move through certain objects. This requires the case to have openings for the ultrasonic sensors transmitter and receiver. The collision detection system will alert drivers when they are about to hit another vehicle or an object located in front of their vehicle. While the vehicle is in use the collision detection system will be awaken from a sleep mode. When the driver gets too close to the object, under 1 ft., the distance between them and the object in front of them will be displayed on the phone. If they continue to move forward they will be alerted when they are about to strike the object. This feature can also be turned on and off at the drivers discretion.

5.4 Overall Design Summary

The OpenXC API created by Ford is used to implement the project. With this and the vehicle interface that was provided to us by Ford, we are able to obtain vehicle data that is not available to most drivers. With this we are able to build the DMS. The DMS consists of four main parts. These parts are fuel efficiency analysis, blind spot detection, collision detection and a rearview camera. An Android device is used as the center of all the four parts. An application that is installed on the Android device will communicate with the peripherals attached to the vehicle. The Android device will be connected to this peripherals using Bluetooth. The Android device will receive specific vehicle data from the Vehicle interface that is connected to the OBD-II port of the car. When the turn signal is being used the Android device will be alerted about the new status of the turn signal. It will then send an interrupt signal to the respective blind spot sensor (left or right) to activate it. The blind spot sensor will then relay any information if movement is picked up within the driver's blind spot. To communicate with the collision sensor the Android device will wait for information regarding the distance between the driver's vehicle and the vehicle or object in front. When this distance becomes too small (under 1ft) the driver will be alerted of their position and how far away they are from this vehicle. The rear view camera will be connected directly to the Android device using a USB. When the driver places the car into reverse video of what's behind the car will be displayed on the Android device.

Along with all of the wireless communication done with each component placed throughout the vehicle the Android device is also the center for fuel efficiency calculations. The application on the Android device that will be controlling communications between all of the devices will also perform these calculations. There will be two calculations done which include long term and real term analysis which will help the driver improve their driving behaviors. These calculations will be done using information that is retrieved from the vehicle interface such as vehicle acceleration. To allow for all of this to be done on an

Android device, the rear view camera, and the collision detection system will have to wait for an alert from the Android device. This way each component will not interfere with one another. An overall block diagram of the DMS can be seen below in Figure 32.

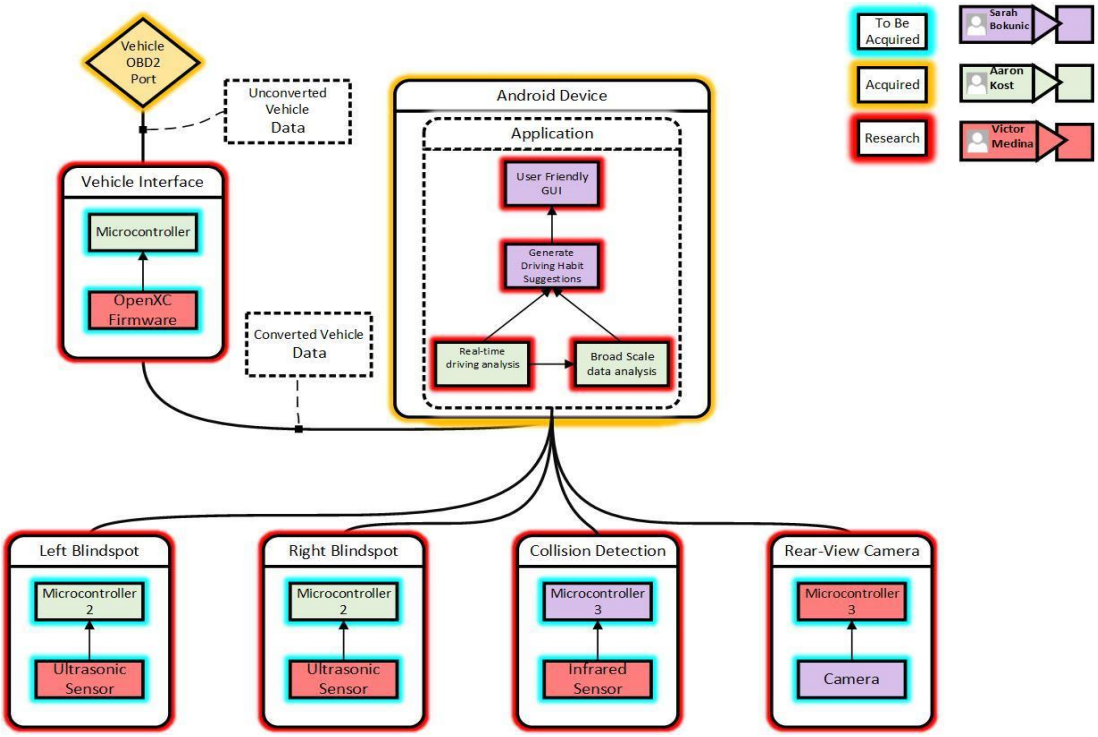

Figure 32: Overall block diagram of the Drive Management System.

6.0 Project Prototype Testing

6.1 Software Testing

In order to determine if the project is working as anticipated, a variety of tests need to be performed. These test make sure that the blind spot detectors, visual display for fuel efficiency, graphs, and other aspects of the project meet all of the specifications and requirements. Testing each functionality occurred throughout the design process, originally starting with generic data input and progressing as more features are fully implemented in the vehicle. Prior to completing the project, all tests described below must meet the expected results. If the expected results are not met, proper measures must be taken to ensure that they are met.

6.1.1 Blind Spot Detection

The blind spot detection is expected to activate when the steering wheel angle is in the appropriate range. It will continually monitor to determine if any vehicle appears in the blind spot, so long as the steering wheel angle remains in the

range. A steering wheel angle between -110 and -45 only activates the sensor on the left side of the vehicle and a steering wheel angle between 45 and 110 only activates the sensor on the right side of the vehicle. The test cases are shown in table 19. Before running the test cases in table 19, the Bluetooth communication between the Android device and each blind spot sensor had to be tested. Since the blind spot sensor is detecting motion, the communication between the Android device and the blind spot system can be done without the vehicle. One of the group members will walk across the sensor in its field of vision. The blind spot detector will then send the alert to the Android device. This was repeated at various distances until the max distance of 10 meters is tested. As long as the blind spot system communicates with the Android device appropriately it has passed this stage of testing, otherwise it will need to be modified.

Step	Procedure/Input	Expected Results
1. Left Turn Signal	The steering wheel angle is between -110 and -45	Audio notification will start which indicates if it is safe to switch into the left lane, and a display on the screen will show a left arrow with a cross if it's not okay
2. Right Turn Signal	The steering wheel angle is between 45 and 110	Audio notification will start which indicates if it is safe to switch into the right lane, and a display on the screen will show a right arrow with a cross if it's not okay
3. Default	The user returns the turn signal lever to the default position	Audio notifications stop and the display stops showing a message

Table 19: Blind Spot Detection Test Cases

6.1.2 Collision Detection

Collision detection will be activated if the driver gets too close to an object that is in front of their vehicle. It will result in an audio notification, unless disabled, and a notification on the screen. The collision detection will continuously run throughout the drive so long as the vehicle speed is over 35mph. The test cases in table 20 show how the application should react to a possible collision. First the application will determine if a possible collision can occur. This happens when the driver is too close to an object or vehicle. The driver will be alerted by an audio notification which is different form the blind spot detector notification will begin to play to alert the user that they are too close. There will also be a large image on the screen which shows "TOO CLOSE." The range for being too close

is under 250 inches. The second step is when the driver's vehicle is too close to an object in front of it. No audio notification will be played, and no image will be displayed. This prevents the battery from being wasted simply because the user is stopped at a stop light. The final step will be while the user is driving a safe distance from the car in front of theirs. No audio notifications will play and no image will appear on their device.

Step	Procedure/Input	Expected Results
1. Possible Collision in Motion	The user drives too close to an object in front of their vehicle	An audio notification which is different from the blind spot detector notification will begin to play to alert the user that they are too close. A display on the screen will show TOO CLOSE.
2. Possible Collision no Motion	The user is stopped too close to an object in front of their vehicle	No audio notification will play, and nothing will display on the screen, even if the user is within the range that was considered to be too close to other drivers in other conditions.
3. No Possible Collision	The user drives a safe distance from any object in front of their vehicle	No audio notification will play and now display will appear on their device.

Table 20: Collision Detection Test Cases

6.1.3 Visual Display for Fuel Efficiency

The visual display will be based off a gradient system that reacts to how the driver is currently performing behind the wheel. The gradient display will change based on data that is received via the vehicle inferences vehicle management service. For the gradient, only the vehicle speed is taken into consideration. However, other factors, such as accelerating, braking, speeding, braking after accelerating, and idling are taken into account for other fuel efficiency data. This data is shown to the user in the form of toasts rather than a color gradient, but is still considered part of the fuel efficiency display. If the driver exhibits bad driving habits in terms if accelerating and braking too hard, the gradient will change based on the calculated score from the received vehicle data. The vehicle speed is calculated into an acceleration and stored in the application's temporary memory and the past 10 accelerations are added together using a weighted average. The weighted average is adjusted to be a score out of 100 for the gradient to change appropriately, but acceleration is given a stronger weight than deceleration.The test cases detailed in Table 22A are the generalized expected inputs during normal operation.

Step	Procedure/Input	Expected Results
1. Starting	The user starts the driving session and starts the engine	The program starts calculating the user's score and adjusts the color display accordingly. It continually stores the data gathered into a file. The display remains on and does not allow the phone display to dim due to inactivity.
2. Starting Fails	The user starts the driving session but the Android device does not detect OpenXC data	The program shows a blank screen which allows them to see that the OpenXC has failed.
3. Acceleration	The user acceleratesand score is less than 50	The further the user pushes on the accelerator, the more their score drops, and the display changes colors to reflect this action. Also a point is added to acceleration score every 30 seconds.
4. Braking	The user brakes and the score is less than 50	The user's score drops and the display changes colors to reflect this action. Also a point is added to braking score every 30 seconds.
5. Idling	The user idles for longer than 1 minute	A message is displayed showing that the user has been idling for longer than 1 minute and a point is added to the idling score.
6. Speeding	The user drives at a speed faster than 70 mph	A point is added to the speeding score every 30 seconds that the vehicle speed is over 70 mph
7. Braking after Accelerating	The user presses the brake pedal after pressing the accelerator past 20% within 10 seconds	A point is added to the brake after accelerating score if the brake pedal is pressed within 10 seconds of the accelerator being pressed past 20%
8. Messages	Speed = 0	Messages displayed on screen.

Table 22A: Visual Display Test Cases

Once all of the testing criteria in table 22A have been met, the values of the score were tested and adjusted to provide drivers with accurate data. This part of the application required balancing points earned and lost by different driving behaviors. The effect of the total score on the gradient displayed on the Android device was also be tested. The gradient was adjusted to not change too quickly so that the driver can notice their current fuel efficiency. The gradient displayed must did not change too slow as that would no longer be a reflection on the real time calculations.

6.1.4User Interface

The user interface for the program allows the user to choose different options which bring up different displays. The different displays will either allow the user to view new information or allow the user to make changes to the program. The steps to test that this is working as anticipated are shown in table 22B.

Step	Procedure/Input	Expected Results
1. Start Driving Session	The user presses on the Start Driving Session button	The screen changes to the Visual Display for Fuel Efficiency
2. Last Driving Session	The user presses on the Last Driving Session button	The screen changes to show a graph of the most recent driving session
3. Driving History	The user presses on the Driving History button	The screen changes to show new buttons to see past driving sessions or an overview
4. Your Fuel Economy	The user presses on the Your Fuel Economy button	The screen changes to a display showing tips that are geared toward frequent mistakes the user has made while driving
5. Options	The user presses on the Options button	The screen changes to show the options: audio toggle, rear view camera toggle, change colors, test colors, and erase data
6. Credits	The user presses on the Credits button	The screen changes to show the credits for the application
7. Default	The user presses the Back button	The screen returns back to the main menu

Table 22B: User Interface Test Cases

6.1.4.1 Driving History

If the user wants to view their driving history, there are a few options to choose from. These options were tested as shown in table 23. For the first option, there is actually more than one button available to be pressed. For example, there areten different “past driving session” buttons where each stores a different data set, in order for the user to be able to view specific past driving histories. The oldest is number 10 and the newest is number 1. Once the limited amount of data is stored, the older files will be overwritten with new data.

Step	Procedure/Input	Expected Results
1. Past Driving Session	The user presses on one of the available Past Driving Session buttons	The screen changes to show a graph of the indicated past driving session
2. Overview	The user presses Overview button	The screen changes to show a graph of an overview of all driving sessions in memory

Table 23: Driving History Test Cases

6.1.4.2 Options

The options menu has different choices that can be made to alter how the program functions. This includes audio functionalities, color themes, and erase data buttons. All of the possible test cases to determine that the options menu works properly are shown in table 24.

Step	Procedure/Input	Expected Results
1. Audio Toggle	The user presses the Audio button	If the audio is not muted: Mutes the audio that plays while driving. If the audio is muted: Unmutes the audio that plays while driving
2. Color Choice	The user presses a color theme	The colors in the application change to reflect the chosen color option
3. Erase data	The user presses one of the erase data buttons	Erase all charts: Erases all data stored on the charts in the application. Erase long term data: Erases all long term data

Table 24: Options Test Cases

6.1.4.2.1 Change Colors

There are four color options to which the user can choose to switch the color theme. Each color in the theme is mapped to specific parts of the base program so there will be no layout difference depending on the colors. The only change that is made is the actual color change. The test cases to determine that these work properly are shown in table 25.

Step	Procedure/Input	Expected Results
1. Normal (Green / Yellow / Red)	The user presses the button for Normal color vision	The color theme of all aspects of the program changes to black, white, green, yellow, and red display
2. Monochromacy (White / Gray / Black)	The user presses the button for Monochromacy color vision	The color theme of all aspects of the program changes to a black, white, and gray display
3. Deuteranopia & Protanopia (Blue / White / Red)	The user presses the button for Deuteranopia & Protanopia color vision	The color theme of all aspects of the program changes to a black, blue, white, and red display
4. Tritanopia (Green / White / Red)	The user presses the button for Tritanopia color vision	The color theme of all aspects of the program changes to a black, green, white, and red display
5. Default	The user presses the Back button	The screen returns to the Options menu display

Table 25: Changing Colors Test Cases

6.2 Safety

Technological advances in vehicles bring with them inherent distractions that could possibly cause issues with safety. Extensive research has been done by National Highway Traffic Safety Administration on how distracting technology can be while still remaining relatively safe for the driver. DMSadheres to the research done, trying to minimize distraction to the driver. All features implemented in this project must be done so in a way that is completely safe, providing no extra hazard that could result in a dangerous situation for the driver. Testing was done to ensure that placement of all new objects as well as audio are not distracting or disrupt concentration. The following list details the requirements to ensure that the project is safe.

- The phone must be mounted in a position that does not block vision or get in the way of the driver's hands.
- The phone must be securely mounted so that it will not fall off of the mount while the user is driving, even in the event of a sudden stop or sudden acceleration.
- The phone mount must be securely attached to the vehicle so that it will not detach from the vehicle even in the event of a sudden stop or sudden acceleration.
- The rear view camera must not block the driver's ability to see through the rear view mirror.
- The rear view camera must not activate at any point that the user is not in reverse, so as to not provide a distraction to the driver.
- The blind spot detection must accurately detect if there is someone in the blind spot. Not detecting another vehicle being in the blind spot could result in serious harm.
- The blind spot detection must activate for the correct side of the vehicle.
- The audio notifications must begin as soon as possible after the turn signal is activated.
- The audio notifications must clearly identify whether or not it is safe to switch lanes.
- The audio notifications must not be surprising, shocking, obnoxiously loud, or imitate emergency vehicle noises.
- All audio notifications must have a visual counterpart to accommodate users who are deaf.
- The visual display must not be distracting.
- It should take under 2.0 seconds for the driver to look away from the road and at the screen, however it is preferable for the user to be able to see the screen without needing to look away at all. [17]
- The visual display must not rapidly change to different colors in a way that could trigger someone with epilepsy.

6.3 Simulations

To ensure that the fuel economy monitoring portion of the DMS is working as intended, and also for presentation purposes simulations will be created to show how the application reacts to different situations. There will be several different simulations; The first simulation will demo how the application responds to a driver who is driving with poor fuel economy in the city, the second simulation will demo a driver who is driving with good fuel economy in the city, the third simulation will demo a typical trip on the highway, and the fourth simulation will demonstrate how the application responds to driving with excessive acceleration and lane changes. The simulations will be created using trace files that represent pre-recorded data points. The data points will be generated by taking short drives in the vehicle with the VI set up to record trace data points to a file so that we have the desired data for each simulation. A data point consists of a signal tag, a value, and a timestamp. The following are some sample data points.

- {"name":"accelerator_pedal_position","value":0,"timestamp":1361454211.483000}
- {"name":"torque_at_transmission","value":1,"timestamp":1361454211.488000}
- {"name":"steering_wheel_angle","value":-46.7,"timestamp":1361454211.521000}
- {"name":"fuel_consumed_since_restart","value":0.326952,"timestamp":13614542

The first simulation that represents a driver who is driving poorly, will showcase how the algorithm for changing screen color represents the driver's choices behind the wheel. The simulation will accomplish this by choosing trace data points that represent over acceleration, poor fuel consumption, and excessive braking, and fast lane changing. This simulation should be programmed so that the color gradient change on the screen and the pop up toast suggestions are activated.

The second simulation will showcase how the application responds to a driver who is driving well according to the rules that DMS follows for driving efficiently. This means recording vehicle data for when the car is operating in ranges we have calculated to be an acceptable driving score. In this simulation the gradient should stay within the green and yellow range and never enter the orange and red ranges. There can still be toast suggestions but they should not be related to things such as acceleration and excessive braking. Toasts for this simulation could include things like excessive lane changing or idling.

The third simulation will be designed to demo how the application will function on the highway. The highway is a special case because fuel consumption tends to be more efficient on the highway because of the constant speed and generally very little acceleration. The special case to consider for the highway is excessive speeds. The variable that will affect the highway driving the most is if the vehicle reaches speeds over 70 miles per hour. For this simulation the vehicle will be at a constant speed of 60 miles per hour and then will periodically accelerate to and maintain speeds over 70 miles per hour, then decelerate back to 60 miles per hour.

The fourth simulation is designed to show the reactiveness of the DMS to "aggressive" driving. For this simulation data will be recorded while the vehicle accelerates excessively, breaks excessively and rapidly changes lanes. This simulation is intended to show how the DMS will make suggestions to help the driver improve on their driving and fuel economy, and also how fast the gradient reacts to changing vehicle conditions in rapid succession. This simulation differs from the first simulation in that it is not intended to show common driving mistakes and is intended only to showcase the most extreme of driving behavior.

6.4 Road Testing

The road tests were conducted during the final stages of testing of the design. Testing done while the vehicle is in use was the most important stage of testing. This determined the overall functionality of the project. Each component of the project was tested in different types of weather conditions. The DMS also was tested during night time to verify that it not only works during night, but is also not a distraction to the driver during the night. Since the DMS requires that one person is driving, at least two people must be present during this stage of the testing phase. This is used as a safety measure to allow the person driving to focus their attention on the road, while another person can monitor the activity of the DMS and verify that the project works correctly. The driver also can notify how easy it is to tell their driving habits without taking their eyes off the road.

Most importantly all of the components to the DMS must be attached or placed onto the vehicle securely. Different types of driving conditions that the DMS must survive through include highway driving at higher speeds, heavy traffic, and normal city traffic. They must not fall off while the vehicle is in use. If any component of the DMS were to fall off the vehicle while in use, the DMS would not be able to function properly. This would increase the cost of the project due to having to replace a component(s). It is important that the method of attaching the components to the car does not cause any scratches or dents to the vehicle. The placement of these components must not interfere with the driver's view of the road or attention. They must also not interfere with other drivers. Each sensor must be placed within a case that can withstand different types of weather conditions and driving conditions. The following lists contain testing requirements of the project which are specific to each component.

Fuel Efficiency

- Messages must be sent to the driver while the car is not in motion as to what they can do to achieve better fuel efficiency levels.
- Must test multiple driving behaviors to verify that all possible scenarios are met.
- Real time analysis on fuel economy must be displayed in a non-distracting way.
- Long term data on driving behaviors and fuel economy must be saved correctly and easy to access for the driver. The information must be displayed in a manner which is easy to read.

Blind Spot Detection

- Must be active while the vehicles turn signal is on.
- Objects located within the blind spot must be detected regardless of size.
- The driver should receive an alert as to when the blind spot detection system has sensed an object.
- While the vehicles turn signal is off no alerts should be sent to the android device.

Collision Detection

- Able to detect objects very close to the front of the car.
- Test that the driver is alerted of an object which they could potentially drive into.

Rear-view camera

- While pulling out of a parking spot in reverse the driver must be presented with a clear video.
- While backing into a parking spot the driver will be provided with a clear video feed so that they may park correctly.

6.4.1 Fuel Efficiency road testing

To accurately test that the fuel efficiency monitoring portion of the project is working properly, multiple test drives were required. The vehicle must first be driven with different driving behaviors. Initially the vehicle is driven using a reckless driving behavior where the driver is accelerating quickly and changing lanes often. This style of driving will be used to make sure that the DMS is taking the correct information from the car and using it to see how the driver is driving. The driver should then receive a low score for their fuel efficiency. While the driver is at a stop, the android devicedisplays information as to what they are doing wrong. This information includes accelerating too fast and braking too hard, as well as other factors mentioned in previous sections. Another style of driving that will be used is a safer style of driving where the driver accelerates smoothly, allows time to come to complete stop, and does not change lanes too often. When the DMS sees this style of behavior the driver will be prompted with messages that praise them on their good driving behavior. The fuel efficiency score may rise due to their more fuel friendly driving style.

After testing driving behaviors, a long distance drive was required to verify that the long term fuel consumption data is being stored correctly. This drive should have been approximately 20 miles in distance or last about an hour in duration. During the drive the driver changed their speed and driving behaviors from a less fuel efficient style to a much more efficient style. The passengers within the car then verified that the real time fuel efficiency data has changed in a similar manner to the driver’s behavior. After the drive the long term fuel economy data that is stored will be observed. This is to verify that the data stored is accurate. The Ford Focus then needed to be taken out on other small trips throughout the week to make sure that the fuel economy data for the week was also being stored accurately.

6.4.2 Blind Spot Detection road testing

When testing the blind spot detection system, another car was used to act as a test object within the driver's blind spot. This prevents the chance of an accident if the test was to be done with unsuspecting cars on the road. The driver behaved normally and changed lanes as usual. The test carthen approached the driver and pull into the driver's blind spot. The driver will then prepare to turn using the steering wheel angle in the direction of the test vehicle. The blind spot detection system then sensed the test car and communicated with the android device to alert the driver that there is an object within their blind spot. A further step into testing the blind spot detection system isto see that when the driver uses turns their steering wheel before a car enters the blind spot, that the sensor will detect the car when it enters the blind spot. This was done by first having the driver user turn their steering wheel in the proper direction. The test car then drove into the blind spot of the driver to be detected. The last portion of the blind spot testsincluded testing the system when the steering wheel is turnedwhile there are no objects within the blind spot. This was used to make sure that the blind spot detection system does not set any false positives.

6.4.3 Collision Detection road testing

The collision detection system was tested in two different ways. It is used when approaching another car on the street. When approaching a stopped vehicle the driver allowed the vehicle to coast slowly towards the vehicle in the front. When there is less than a foot of space between the front of the driver’s vehicle and the vehicle to the front the driver must be alerted through their android device. The collision detection also was tested while parking. The Ford Focus was taken to a UCF parking garage where the driver attempted to park the vehicle. As the front of the vehicle approaches the wall in front the collision sensor communicated with the android device to alert the driver when they are close enough.

7.0 Administrative Content

7.1 Budget and Finance

The project is not sponsored and will be funded by the group members. Since all the funding will be provided by the group members, keeping the overall cost of the project low is very important. The total cost was lowered by group members already owning required products and the donation of an OBD-II reader by Ford. Additional donations, sponsorships, or assistance are being sought after to lower the total cost of the project. The group is prepared to fund the project out of pocket if no additional assistance can be found. The expected expenses and

used parts are shown in table 26. The cheapest parts that met the specifications of the DMS were chosen to implement the project.

Part	***Manufacturer***	***Qty.***	***Unit Price***	***Net Price***
2013 Ford Focus	Ford	1	$16,500.0 (Donated)	$0.00
MSP430g2553 microcontroller	Texas Instruments	3	$2.25	$6.75
Maxbotix LV-EZ2	Cytron Technologies	1	$29.99	$29.99
HB100 Microwave Sensor	ST Electronics	2	$10.00	$20.00
Samsung Galaxy S4 (Android Device)	Samsung	1	$584.29 (Donated)	$0.00
RN-42 Bluetooth Module	Roving Networks	2	$15.95	$31.90
Lithium Ion Battery	Energizer	3	$7.20	$21.60
PCB (4 square inch)	4PCB	3	$33.00	$99.00
HC-06 Bluetooth Module	SMAKN	3	$10.00	$30.00
OBD-II Reader	Ford	1	$100.00 (Donated)	$0.00
Vehicle Phone Mount		1	$5.99	$5.99
Estimated Total:			$245.23	

Table 26: Table of finances for the DMS.

7.2 Milestone Chart and Discussion

The following tables provide descriptions of milestones for Senior Design 1 and Senior Design 2. Although each member has specific responsibilities, it is the group's responsibility to meet each deadline as close as possible. The time for development in Senior Design 2 is small, which means fast prototyping is required. The shorter deadlines allow room for any miscalculations to be fixed before the project is due. This will prevent the group from not being able to finish on time or overlooking a major flaw that may not be noticed before presenting the project.

Date	Task
11/7/2013	Draft document finalized and turned in.
11/11/2013	Software design concept completed.

11/20/2013	Hardware design concept completed.
12/2/2013	Senior Design 1 final paper turned in.
12/9/2013	Order parts for prototyping.
12/17/2013	Connect vehicle interface with Ford Focus.

Table 27: Table of milestones for Senior Design 1.

Date	Task
1/5/2014	Begin prototyping of blind spot detector.
1/20/2014	Finish prototype of blind spot detector.
1/21/2014	Begin Android app development.
2/15/2014	Begin front collision detection prototyping.
2/21/2014	Finish Android app development.
2/28/2014	Finish prototyping of front collision detector.
3/01/2014	Begin testing stages of DMS.
3/24/2014	Finish testing of the DMS.
4/14/2014	Final Documentation

Table 28: Table of milestones for Senior Design 2.

7.3 Work Distribution

Each group member was responsible for different portions of this report. Although different aspects were agreed upon by all members, one member was to come up with design choices for each member to look over. The responsibilities of each group member are detailed below in table 29. As the Computer Engineers of the group Aaron and Sarah are focusing on designing the application for the DMS and the fuel efficiency algorithms that will go along with it. Since the Android device is the centerpiece of the project, it is important that the application running on the Android device is programmed correctly. The fuel efficiency part of the project also requires a lot of design as many variables affect this. As the Electrical Engineer of the group Victor will be responsible for the circuit design of the components that will be used with the DMS. Although each group member has specific tasks assigned to them, it is important that they review the work done by each of their peers. As this is a team effort it is important that the group cooperates throughout all stages of the project. If a

group member runs into problems while designing and prototyping the DMS it is necessary that they look for help from other group members or faculty members at UCF. The responsibilities of each group member were decided on based on their knowledge and experiences.The writing of the paper has been split up evenly with a minimum of 30 pages per group member. The following tabledepicts what each group member has been assigned.

	Wireless Comm.	Power	Hardware	Android App
Aaron Kost	X	X	X	X
Sarah Bokunic				X
Victor Medina	X	X	X	X

Table 29: Table of the project work distribution in the group.

8.0 Project Operation

Driving Management System was intended to be a cheaper alternative to current market products for fuel efficiency monitoring as well as blind spot and collision detection sensors. It does however have some instructions for operation for the user.

8.1Hardware Peripherals

To correctly operate the Driving Management System the sensors must be turned on in a certain order so that the wireless network correctly initializes. First, the right blind spot sensor must be turned on followed by the left blind spot sensor. This allows the blind spot system to correctly initialize. After starting the blind spot system, turn on the collision avoidance at the front of the car. This allows the blind spot system to connect to the collision avoidance. Once the entire system is powered, the hardware will wait for the Android device to connect. To correctly use the Driving Management System, place the two blind spot detectors within the vehicles trunk on each side. Then place the collision avoidance at the front of the car, in the license plate area. Placement of the sensors can be seen in figure 33 below.

Figure 33: Placement of the hardware peripherals throughout the car.

8.2Android Application

1. On the main screen, as shown in Figure 34, to start the driving session, the user presses the Start Driving Session button.
2. To view the most recent driving session, the user presses the Last Driving Session button.
3. To view up to 10 previous driving sessions as well as an overview of all driving sessions, the user presses the Driving History button.
4. To view an overview of the user's fuel economy information gathered from the application, the user presses the Your Fuel Economy button, as shown in Figure 33.
5. To view options to customize the application or to erase data, the user presses the Options button.
6. To view the credits on the application, the user presses the Credits button.

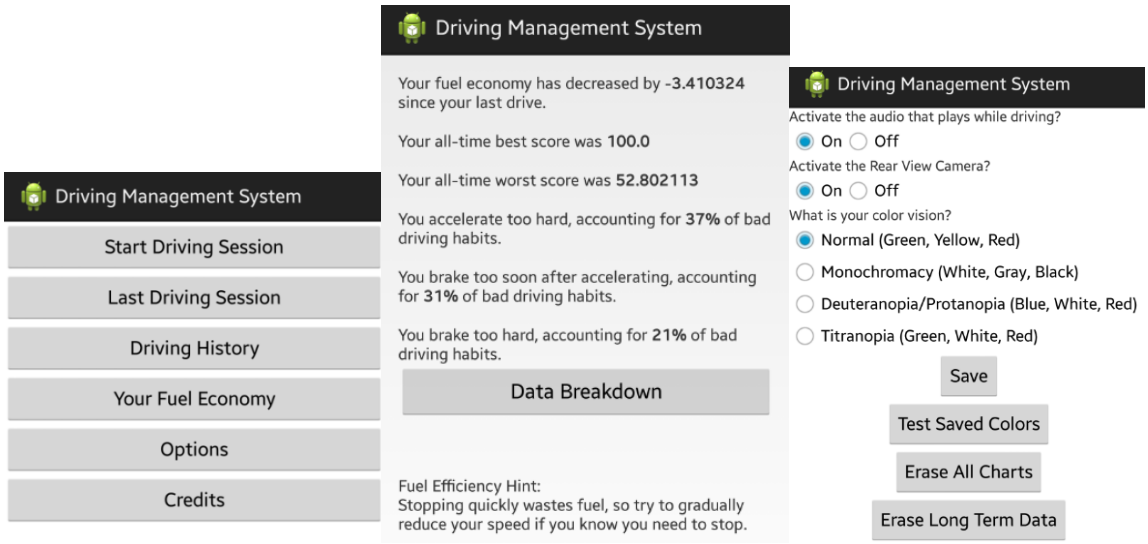

Figure 34: Screenshots from application

Appendices

Appendix A - Permissions

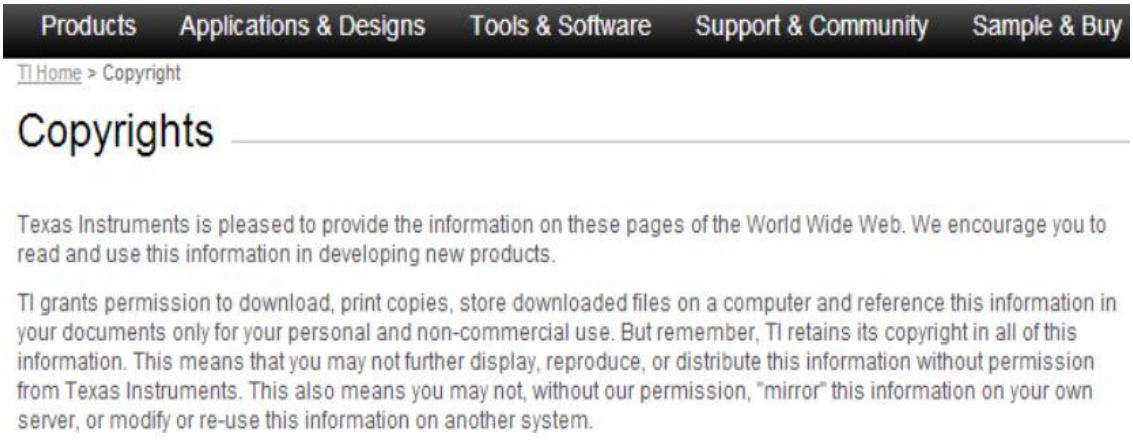

Figure 35: Texas instruments permissions.

Educational and Non-Profit Use of Copyrighted Material: If you use Microchip copyrighted material solely for educational (non-profit) purposes falling under the "fair use" exception of the U.S. Copyright Act of 1976 then you do not need Microchip's written permission. For example, Microchip's permission is not required when using copyrighted material in: (1) an academic report, thesis, or dissertation; (2) classroom handouts or textbook; or (3) a presentation or article that is solely educational in nature (e.g., technical article published in a magazine). Please note that offering Microchip copyrighted material at a trade show or industry conference for the purpose of promoting product sales does require Microchip's permission.

Figure 36: Microchip permissions.

Appendix B - References

1. "Gas Mileage Tips - Driving More Efficiently." *fueleconomy.* n.p., n.d. Web. 5 Nov. 2013. <http://www.fueleconomy.gov/feg/driveHabits.shtml>
2. Ford. *2013 Focus Owners Guide.* Web. 15 Nov. 2013. <http://www.fordservicecontent.com/Ford_Content/catalog/owner_guides/13focom1e.pdf>
3. "Fuel-efficient Driving." *eartheasy*. n.p, n.d. Web. 5 Nov. 2013. <http://eartheasy.com/move_fuel_efficient_driving.html>
4. "9 Easy Ways To Increase Your Gas Mileage." *Investopedia.* Jean Folger, 21 Feb. 2013. Web. 14 Nov. 2013. <http://www.investopedia.com/financial-edge/0211/9-easy-ways-to-increase-your-gas-mileage.aspx>
5. "2013 Ford Focus S Sedan." *Ford*. n.p, n.d. Web. 3 Nov. 2013. <http://www.ford.com/cars/focus/2013/trim/ssedan/>
6. Atmel Corporation. *8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash.* Web. 12 Nov. 2013. <http://www.atmel.com/Images/doc8161.pdf>
7. Texas Instruments Incorporated. *2.4-GHz Bluetooth low energy and Proprietary System-on-Chip.* Web. 14 Nov. 2013. <http://www.ti.com/lit/ds/swrs110d/swrs110d.pdf>

8. *Android Developers.* Android. Web. 16 Nov. 2013. <http://developer.android.com/>
9. "OpenXC Message Format Specification." *openxc*. Ford Motor Company, 2012-2013. Web. 1 Nov. 2013. <https://github.com/openxc/openxc-message-format>
10. "GraphView Library." *jjoe64 developer blog.* Jonas Gehring, 2011. Web. 15 Nov. 2013. <http://www.jjoe64.com/p/graphview-library.html>
11. "What is Color-Blindness." *Color Matters.* J.L. Morton, n.d. Web. 15 Nov. 2013. <http://www.colormatters.com/color-and-vision/what-is-color-blindness>
12. "Doppler Effect." *Wolfram Research.* Eric W. Weisstein, n.d. Web. 3 Nov. 2013. <http://scienceworld.wolfram.com/physics/DopplerEffect.html>
13. Tanenbaum, A. S., and D. J. Wetherall. *Computer networks*. Prentice Hall, 2010. Print.
14. "Understanding the Linear Regulators." *DigiKey.* Steven Keeping, 8 May 2012. Web. 10 Nov. 2013. <http://www.digikey.com/us/en/techzone/power/resources/articles/understanding-the-linear-regulators.html>
15. "Logitech HD Webcam C270." *Logitech.* Logitech, 2013. Web. 23 Nov. 2013. <http://www.logitech.com/en-us/product/hd-webcam-c270?crid=34>
16. "Logitech Broadcaster Wi-Fi Webcam." *Logitech*. Logitech, 2013. Web. 23 Nov. 2013. <http://www.logitech.com/en-us/product/broadcaster-wifi-webcam?crid=34>
17. National Highway Traffic Safety Administration. *Visual-Manual NHTSA Driver Distraction Guidelines for In-Vehicle Electronic Devices.* Web. 26 Nov. 2013. <http://www.nhtsa.gov/staticfiles/rulemaking/pdf/Distraction_NPFG-02162012.pdf>

Appendix C - Table of Tables

Table of Tables

Appendix D - Table of Figures

Table of Figures