DMS: Driving Management System

Aaron Kost, Sarah Bokunic, Victor Medina

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — A hands free vehicle system that provides drivers with fuel efficiency analysis, blind spot detection, collision avoidance, and a rearview camera. It consists of microwave sensors, an ultrasonic transducer, and a webcam to perform correctly. The DMS will interact with the driver through an Android device located in the vehicle. The MSP430G2553 was chosen as the microcontroller for the project. It will control wireless communications between an Android device and the hardware peripherals. It is also used to sample the signal from each sensor. A preamplifier stage is required to bring the microwave sensors output signal to a readable value for the analog to digital converter microcontroller. The Android application will provide real time feedback on driving performance as well as long term analysis on a drivers habits.**

***Index Terms* – Wireless communications, preamplifier, microwave sensors, ultrasonic transducer**

I. Introduction

Due to the high prices of gas, fuel efficient cars have become very popular over the past decade. Improvements have been made through efficient engines and electrical vehicles. However, these vehicles are only as efficient as the drivers maneuvering them. The DMS looks to provide drivers with information on their driving habits as well as combining this feedback system with safety features such as blind spot detection, collision avoidance, and a rearview camera.

Safety features such as blind spot detection help avoid accidents. Many drivers who have disabilities take advantage of these features to protect themselves on the road. Unfortunately obtaining these safety features when buying a car are expensive. Typically consumers who have purchased a new standard package vehicles such as the 2013 Ford Focus pay anywhere from $15,000 to $21,000 dollars without advanced features such as a backup camera, blind spot detection, or advanced fuel efficiency assistance.

The main requirement of the project is to provide drivers with a cheaper and more efficient alternative to expensive manufacturer and aftermarket products. Most importantly each feature of the DMS will be tied into an easy to use application on the driver's Android device. This allows the driver to have full control of certain features on their vehicle by simply using their smartphone or tablet. A block diagram of the system can be seen below.

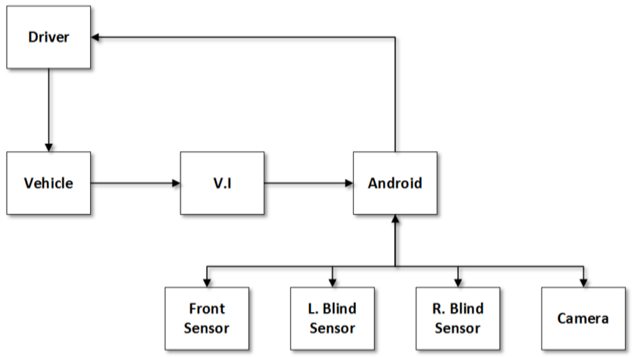


Figure 1: Flowchart of the DMS functionality.

The Android application will provide the driver with real time feedback on their fuel efficiency. The feedback is based off a color gradient which gradually shifts from green to red based how the drivers performance.

The android application will also communicate with the blind spot detection and collision avoidance systems wirelessly. When the driver is making a lane change the appropriate blind spot sensor will alert the driver whether or not there is an object within the blind spot. Based on the driver's speed, if the driver is driving too close to a vehicle in front of him, he will be alerted that additional car lengths should be left between the driver's vehicle and the vehicle to the front. A list of design specifications can be seen below:

- Display fuel efficiency and provide real time analysis
- Long term data storage to analyze driving performance
- Wireless communication between hardware systems and Android device
- Functionality in harsh weather and driving conditions
- Long battery life (5+hrs)
- Low cost (<$300.00)

The paper is organized as follows. First, an introduction to OpenXC and how data is retrieved from the car will be discussed in II. In III the wireless communication system will be presented. The specifications and design of the blind spot sensor and collision detection will be presented in IV and V respectively. In VI the rearview camera will be discussed. The software components such as the android application and microcontroller programming will be presented in VII. The power specifications and design will be presented in VIII. Last, conclusions will be drawn in IX.

II. OPENXC

A. OpenXC Architecture

The project uses Ford's OpenXC to retrieve real time data from the vehicle. The goal of OpenXC is to allow drivers and developers to make their own aftermarket features for Ford vehicles. A vehicle interface module is used to connect to an OBD-II port to transfer data from the vehicle to the user. Some of the vehicle information that is transferred includes steering wheel angle, accelerator pedal position, and brake pedal status. The vehicle interface is designed to be compact so that it can be used while driving without hitting the driver's knees. The vehicle interface module can be seen in the figure below.

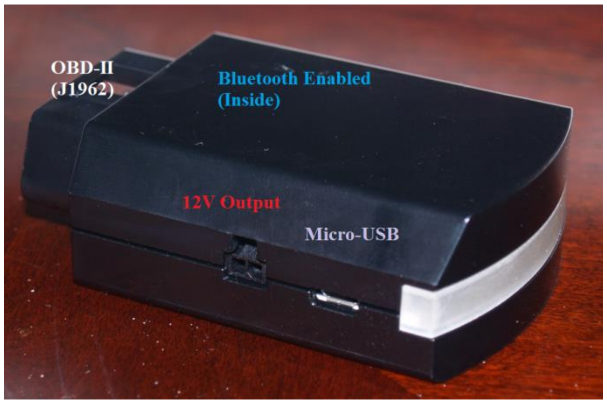


Figure 2: Vehicle Interface module and its features.

The OpenXC hardware requires vehicle specific firmware to operate correctly. For our project the firmware loaded onto the OpenXC hardware is for the 2013 Ford Focus. If the incorrect firmware is flashed onto the OpenXC, the vehicle will be rendered un-drivable as the OpenXC will interrupt the vehicles CAN network, noting the importance of having the correct firmware flashed.

To receive vehicle information on the Android device OpenXC can connect using either Micro-USB or with Bluetooth using the SPP protocol. For our project we opted to use the Micro-USB connection. The choice to use Micro-USB over the Bluetooth connection will be discussed in III.

The hardware connects to the DMS application through an Android service that maintains a connection automatically to the hardware and ensures that the phone receives an uninterrupted data stream. The DMS application binds to the OpenXC Android service using a software library. Once connected to the vehicle Android service the application will receive the vehicle data. It is important to note that as long as the phone is on the vehicle service is running, but DMS application is only bound to the vehicle service during a driving session. The architecture described above is shown below in Figure 3.

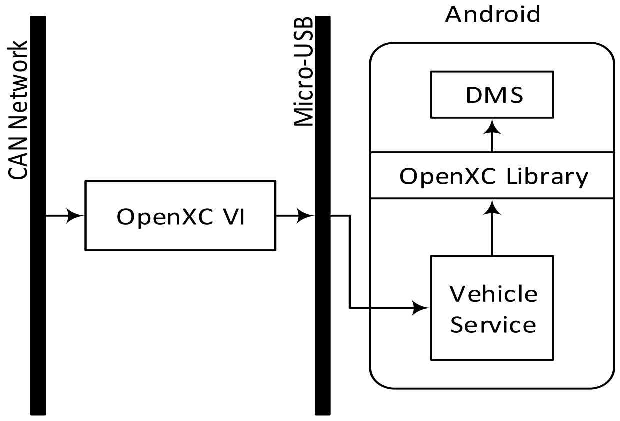


Figure 3: OpenXC Architecture for DMS project.

As shown in Figure 3 the Vehicle Interface works in unison with the vehicle manager service on the Android device to provide a consistent data pipe to any application that binds to the vehicle manager service. In this way DMS is able to receive a constant live-steam of vehicle data information.

B. Why DMS needs OpenXC

One of the most common questions about the DMS project is "Why use OpenXC when there are other diagnostic OBD-II port readers?" To avoid this confusion, the understanding of how the vehicle CAN network functions is important. The CAN messages used by DMS are fundamentally different than the Diagnostic messages on the CAN BUS used by diagnostic readers. OpenXC is not a diagnostic reader, and does not read the same messages that typical OBD-II port scanners access.

The CAN network in the vehicle is constantly sending CAN messages to inform the various mechanical parts of the engine of what needs to be done. Whenever the accelerator is pressed, a message is sent to the Engine Control Unit that informs the engine to increase RPMs. This is just one example of the messages that are being constantly sent over the network. These important messages are sent packed into a 64-bit frame with an 11-bit identifier. The format of the frame and the particular messages are sensitive information that is not readily available to the public. It is these messages that the OpenXC receives from the CAN BUS. The different "nodes" or control centers in the network maintain a diagnostic status that is standard across all vehicles. These diagnostic status are not sent into the CAN Network unless specifically asked for. The diagnostic signals are readily available to the public and are standard across all vehicles. Standard OBD-II readers only access these standard diagnostic signals and are unable to interpret the other CAN Signals constantly moving on the BUS. For these reasons, OpenXC differs from a standard OBD-II port reader in that is doesn't simply ask for standard diagnostic signals, but taps into the CAN network BUS and interprets the sensitive vehicle information, transforms it into usable data, and then sends it to the vehicle manager service for use on the Android device.

C. Adding Custom Messages to OpenXC

Although the signals traveling on the CAN network are relatively sensitive information, it is possible to add a custom message to OpenXC if the frame identifier is known. For DMS we were hoping to add turn signal status to our OpenXC version, however, we could not gain access to the identifier for turn signal status.

While developing the project, we also ran into the problem that the gear position we planned on using did not function properly with the 2013 Ford Focus that has an automatic transmission. To solve the problem we needed to get the position of the gear lever. The frame identifier for the gear lever was already in the 2013 Ford Focus firmware but was not updated in the OpenXC library. To gain access to this message we were able to add the instruction to the OpenXC library. We need this custom message to tell if the car is in reverse.

III. Wireless Communications

Wireless communication is an important aspect of DMS. One of the novelties of the project was that the capabilities we add to the vehicle are relatively easy for the user to set up. Wireless communications allow the use of battery powered sensing circuits that can be placed easily onto the vehicle. Not requiring the hardware to be wired into the vehicles electrical system allows people with no prior experience add features such as blind spot detection to a vehicle.

DMS is an Android application, and because of this fact we are limited by the wireless capabilities of the device DMS is installed on. For our purposes the only options for wireless communication is Bluetooth, or WIFI. Other alternatives are possible but require special peripherals, such as Zigbee requiring the user device to be rooted, so this was not an option for us.

We chose Bluetooth for our project because it required less hardware to be installed in the vehicle and matched the project specifications better than WIFI. Although on paper Bluetooth seemed perfect for our project, we have run into several issues with its use.

In Android to connect a Bluetooth module, the module needs a specific UUID. The default SPP UUID in Android is the same on almost all Bluetooth modules, meaning we can only have one Bluetooth device connected to the Android device using the SPP protocol. We also discovered that not all Android devices can connect multiple Bluetooth devices of the same profile. To clarify, when a device connects to an Android over Bluetooth it uses a specific profile. A keyboard uses a different profile than a Bluetooth audio device, so one could have a keyboard connected, and a headset connected, but not two headsets or two keyboards. The same applies to our hardware modules using Serial Port Profile (SPP).

To solve the issue of not being able to connect all three hardware modules to the phone at once, we changed our network topology to be more linear. The network topology is detailed in Figure 4 below.

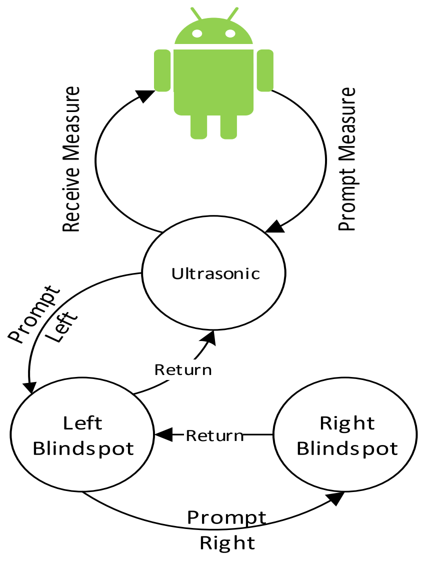


Figure 4: Wireless Network

To accommodate a linear network structure as shown in Figure 4, the hardware and software had to become more advanced. The Ultrasonic sensing circuit and the left blind spot sensor now requires two Bluetooth modules. The reason that they require two Bluetooth modules is that one has to function as a slave, and one as a master. DMS functions in real time and cannot afford the time it takes for a Bluetooth module to change roles and reconnect. The increase in Bluetooth modules also introduced the addition of Multiplexing/De-multiplexing devices. The Transfer and receive is switched between master and slave by the MCU on each sensing circuit. In this way we can pass information through the network.

Due to the Android device being limited to one Bluetooth connection at a time, we are unable to wirelessly connect the vehicle interface module. This leads to the vehicle interface module being connected to the Android device via Micro-USB.

The hardware chosen to implement the wireless network for DMS is as follows:

- HC-06 (Slave Module)
- RN-41 (Master Module)
- CD74HC4067 (Analog/Digital MUX)

The HC-06 was chosen as the slave module for this project because it is extremely affordable and requires no development environment. The module can be purchased for as little as five dollars and is not FCC certified, however for this project that is acceptable as we do not plan on selling this in the marketplace.

The RN-41 was chosen as the master module for this project because it is highly customizable and is able to perform auto-reconnects to a set MAC address. The RN-41 is more expensive than the HC-06 but is still relatively inexpensive and requires no development environment, and can be programmed over Bluetooth from the phone using serial commands. The RN-41 is set in auto-reconnect master mode, meaning it will only connect to the specified slave hc-06 and does not need to ever enter discovery mode saving on power and increasing system stability.

The CD74HC4067 Multiplexer was chosen for this project because it is a diverse device. It can function as a multiplexer or a de-multiplexer and can function on the supply voltage of 3.3 V. The ability to function on 3.3 V is important because it is powered by a MSP430 pin which supplies 3.3 V.

IV. Blind Spot Detection

The use of a blind spot detection system can reduce the probability of accidents occurring during a lane change. Although many vehicles are equipped with mirrors that may greatly reduce the size of a blind spot, most do not eliminate them completely. Due to the turn signal status being unavailable, the blind spot detection system relies on the steering wheel angle of the vehicle. When the driver attempts to make a lane change the Android device located within the vehicle will alert the driver if an object is present in their blind spot.

A. Sensor

A radar module was used to implement the blind spot detector due to it currently being used by Ford and BMW in their blind spot systems. The HB100 microwave sensor module was chosen as the sensor for blind spot detection. It consists of a DRO oscillator, patch antennas, and a frequency mixer which converts the Doppler signal into a baseband output. Some of the features are listed below:

- X-Band frequency of 10.525 GHz
- Typical EIRP of 15 dBm
- Detection range of 10-15m

The Doppler module uses the Doppler Effect to determine the velocity of an object. The module sends out a microwave signal and compares the frequency of the received signal to the frequency of the transmitted signal. If the target is moving within the range of the signal, a change in frequency will be detected. The Doppler Effect is represented by the following equation:

$$f = f'\frac{(c + vs)}{(c + vr)}$$

Where f and f' are the original and received frequencies respectively. The variables V_s represent the velocity of the transmitter; V_r is the velocity of the receiver, and c is the speed of light in medium.

B. Signal Processing

The IF output of the HB100 microwave sensor is in the range of a hundred µV and to a few mV. An overall gain of 1000 was used to obtain a suitable signal for the ADC of the microcontroller to sample. Resistor values were limited to no more than 1MΩ to reduce thermal noise. Since the output of the microwave sensor is very small, it is susceptible to small amounts of noise produced by different components within the circuit.

Objects such as trees, pedestrians, and animals may be detected by the microwave sensor as lower frequencies. A band pass filter is used to filter out these lower frequencies and higher frequencies which can produce false positives when changing lanes. It is attenuated so that vehicles are the primary object detected. The amplifier can be seen in the figure below.

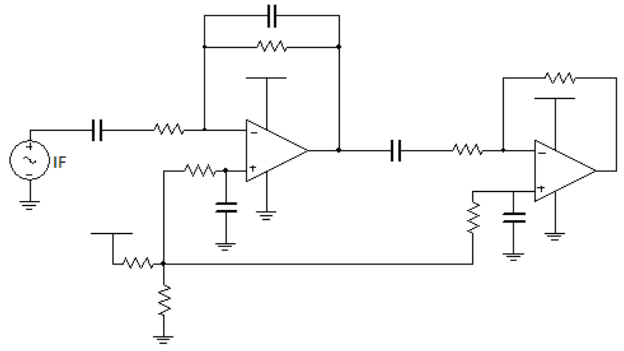


Figure 5: Two stage amplifier circuit.

V. Collision detection

Driving too close to another vehicle may lead to an accident. The collision detection feature of the DMS looks to alert the driver when they are driving too close to the rear of another vehicle. This is done by using the vehicles speed to determine a safe distance for the driver to keep between their vehicle and the vehicle in front. The collision detection system will only activate above 35 mph to conserve battery life. This also prevents the collision detection system from warning the driver that they are driving to close when at a stop. Once the vehicle reaches speeds over 35 mph the driver's Android device will constantly poll the collision detection system for data.

B. Sensor

To accurately determine distance an ultrasonic transducer was chosen as the sensor for the collision detection system. The ultrasonic transducer sends a pulse out and measures the reflected signal. To measure the distance between two objects the sensor uses the time *t* it takes for the signal to return and the speed *c* at which the pulse travels at. Since the pulse travels to and from the object the total distance travelled must be divided by two to obtain a one way measurement. The following equation shows how distance is determined.

$$d = \frac{ct}{2}$$

The Maxbotix LV-EZ2 is the ultrasonic transducer used for the collision detection system. The following list contains some of its features:

- Detection Range: 6.4 m
- Readings occur at a 20 Hz rate
- Multiple analog and digital outputs

VI. REARVIEW CAMERA

The DMS uses a rearview camera to the driver with a video feed of what is behind their vehicle. This allows the driver to have complete vision of what is behind their car while backing up. The use of a rearview camera lowers the chance of striking a pedestrian or object while backing up. Beginning in 2016, the National Highway Traffic Safety Administration will require all new vehicles to come with backup cameras. To implement this feature, a Logitech webcam is placed at the rear of the vehicle above the license plate. The positioning of the camera maximizes what the driver is able to see.

When the driver puts their car into reverse, the Android device will display the video feed to the driver. After the driver has placed their car back into drive, the app will switch back to monitoring their fuel efficiency. Due to time restrictions the rearview camera is not connected wirelessly to the driver's Android device. A USB hub is used to switch the application between the driver's rearview camera and the connection to the vehicle interface.

VII. SOFTWARE

A. Microcontroller

Since the project consists of three separate peripherals that are powered separately by a battery, a low power consumption microcontroller was needed. An onboard UART was also needed to implement wireless communications. The Texas Instruments MSP430G2553 was chosen as the microcontroller to be implemented in both the blind spot and collision detection systems. Some of its features can be seen below.

- Low supply voltage (1.8 to 3.6 V)
- Multiple power saving modes
- UART, SPI, On-Chip ADC

The On-Chip ADC that is provided with the MSP430 is used to sample analog signals from both the Doppler module and ultrasonic transducer. The ADC register holds a value between 0 and 1024 representing the voltage in steps of 3.3 mV. Since the blind spot detection system is checking for movement within the blind spot, a change in the voltage at the pin is simply used to determine if something has been detected. For the collision detection system, the ultrasonic transducers signal must be displayed to the driver as a distance. Using a supply of 3.3V the ultrasonic transducer yields 6.4 mV per inch. The following equation was used to convert the voltage seen at the pin of the ADC to a distance:

$$d = \frac{\left(\frac{ADC}{1023}\right) * Vref}{.0064}$$

Where *Vref* is the reference voltage used by the MSP430. In this case *Vref* is 3.3 V. *ADC* is the value seen at the pin of the On-chip *ADC*. The product of the reference voltage and value seen at the *ADC* is then divided by the voltage seen per inch by the ultrasonic transducer.

B. While Driving

When the user turns on their vehicle, after connecting any components necessary through USB and putting their Android device in a location that they are able to see without turning their head, they will open up the Android application and press the "Start Driving Session" button at the top of the main screen. At this point, their driving session will begin to be monitored. This monitoring occurs on a 50 millisecond clock cycle, which is a short enough period of time to allow the user to receive accurate feedback. While the vehicle is in motion, the user will only see a screen that has no text with a color gradient based on their current driving habits. By default, the colors change from green, “good,” to yellow, “mediocre,” and then to red, “bad,” but the user is able to change this in the options menu before driving.

The color changes based on the user’s acceleration and deceleration, with higher acceleration and deceleration resulting in a lower score. The acceleration and deceleration is calculated using the user’s current speed and past speed from the previous clock cycle. Then, the new acceleration is added to the past 9 accelerations and using a weighted average, the acceleration score is calculate. The raw acceleration score is multiplied by 10 if the user is accelerating and 6.25 if the user is decelerating. These numbers were chosen through testing because the user was graded too harshly while braking when the multiplication factor was equal for braking and accelerating. The acceleration score is subtracted from 100 in order to get the color gradient on a scale from 0 to 100 with a smaller acceleration score being considered better. In order to prevent the colors from flashing from one color to the next which could be a distraction to the user, a separate score is used to keep track of what the user actually sees rather than the raw score. This score is constantly trying to reach the raw score, but it can only move up or down by 1.0 point per clock cycle. This allows the user to see a clear color gradient rather than flashing colors while they are driving.

While the user is driving, more data is being taken in in order to provide the user with short term and long term suggestions for improving their driving behavior. While driving, these suggestions are shown only while the user’s velocity is zero. They are shown to the user via a toast that only appears for a few seconds, but will repeat until the user’s velocity changes. If the user accelerates so hard that their total score is below 50, they will receive 1 point every 30 seconds. The same is true for braking so much that the total score is below 50. If the user presses on the accelerator past 20% and then brakes within 10 seconds, they will receive 1 point for that factor. Every 30 seconds that the user speeds over 70 miles per hour, they receive 1 point for that bad driving behavior. They also receive 1 point every time they idle for more than 1 minute. The bad driving behavior that has the most points once the user’s velocity reaches 0 determines which message is shown to the user. If the user has not committed any bad driving behaviors, the Android application will instead tell the user that they have good driving habits.

In order to prevent the same message from repeatedly being shown to the user, after a message is shown and the user resumes driving, the short term score for that factor is reset to 0. It can show up again within the driving session, but it will only show up if the score for that factor is higher than all other factors. Also, after 1 minute of being idle, the user will always be shown a message about idling for over 1 minute, even if their idling score is less than their maximum score for the other factors.

The application also collects the steering wheel angle as well as gear lever position from the vehicle through OpenXC. The gear lever position is used to determine when the vehicle is in reverse, which will allow the rearview camera to activate. It is also used to determine when to switch back from the camera. When the steering wheel angle is between 45 and 110 degrees, the application starts continually sending the letter “R” for “right” over Bluetooth to the sensors. When the steering wheel angle is between -45 and -110, it starts sending the letter “L” for “left.” In either situation, if it receives back a “Y” then a noise starts playing to inform the user that there is someone in their blind spot. If it receives back an “N” then no noise starts to play.

In addition, if the user is going over 35 mph but the steering wheel angle is not in between those ranges, then it continually sends out an “A” to receive back the distance between the driver and the vehicle in front of the driver. If the distance is below 250 inches, then a noise plays that is different from the blind spot detection noise to warn the user that they are too close to the vehicle in front of them. If none of these conditions are met, then no signal is sent over Bluetooth.

C. Other Features

The score that the user sees via a color gradient while driving is stored into one text file per driving session, with a maximum of 10 individual driving sessions stored at any period of time, and the long term data used to give hints for fuel efficiency are stored into variables. After each driving session, the data is analyzed and a single overall score is calculated, which is stored in a separate text file. The user can view that data in the form of a line graph in the Android application, as shown in Figure 6, in order to see if they are improving their driving habits.

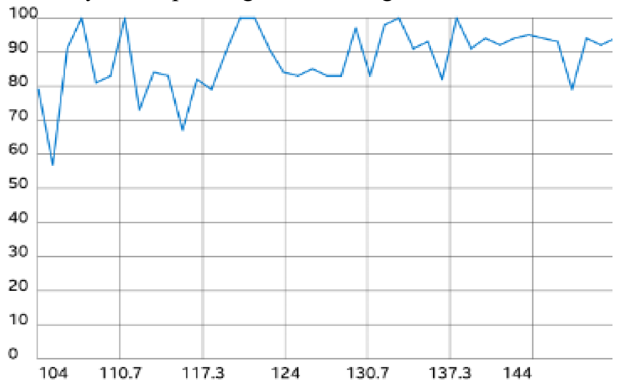


Figure 6: Overall Progress in DMS Application

They are also able to view the individual driving sessions in bar charts on the application, with each bar on the chart summarizing 10 seconds worth of data into a single score.

One section of the application, called the "Your Fuel Economy" section, allows the user to see their progress at a glance. They are able to see how they are improving and what they need to do to further improve without needing to look through their driving charts. This section will show the user how much they have improved or how much they have gotten worse compared to recent data. It will also show the highest score they have received and the lowest score they have received. In addition, this is where the user is able to receive their long term data on which specific bad driving behaviors they frequently commit. The application shows the top three bad driving behaviors, as well as a percentage which shows how frequently they commit that behavior compared to other bad behaviors. For example, someone may brake too soon after accelerating which contributes 50% to their bad driving behaviors, but they may only brake too hard for 10% of their bad driving behaviors, which means they need to focus on not accelerating when they know they need to brake soon. The user is able to view a chart showing the raw data for the bad driving behaviors via this screen as well. When the user begins a new driving session, all of the bad driving habit scores are decreased slightly to allow for the user to improve over time.

VIII. POWER MANAGEMENT

Due to the DMS being compromised of three separate components; three separate batteries are used to power the entire system. Each battery is a 3.7 V lithium ion battery. The maximum power consumption of the blind spot detection system is 621.8 mW with a current draw of 188.42 mA. For the collision avoidance system, the maximum power consumption is 529.3 mW with a current draw of 160.4 mA. The total power budget of the DMS can be seen below.

Device	Power Consumed	Max Current
MSP430	1.3mW	420 µA
HB100	99 mW	30 mA
LV-EZ2 Sensor	6.6 mW	2 mA
HC-06	26.4 mW	8 mA
RN-41	165 mW	50 mA
Multiplexer	165 mW	50 mA

Table 1: Table containing power consumption of each device used in the project.

A push button on the case of each hardware peripheral will allow the driver to turn the system on or off. This allows the driver to turn off the DMS while not in use, increasing the lifespan of the battery. The batteries can be charged through Micro USB. This allows the driver to either charge the DMS through their cigarette lighter in the vehicle or in their home. The MCP73831T charge controller charges the lithium ion battery at 500 mA.

The MSP430 microcontroller is programmed to be used in low power mode. When a signal is received from the driver's Android device the MSP430 will awaken and perform the requested action in active mode. While in its low power mode the MSP430 draws as little as 0.8 µV.

The op amps used were powered using a single-supply configuration. Single-supply operation requires the use of a reference voltage. This method was chosen to avoid the use of dc-dc converters to supply a negative voltage to the op amp. Single-supply op amps do not use a ground reference and must be biased to ensure that the output voltage swings between the correct voltages. To bias the op amps, a reference voltage of VCC/2 was used. A current limiting resistor was used to reduce the error caused by input bias current. It is also important to note that decoupling capacitors must be placed between the power supply and each component. This blocks the noise produced by other components from traveling through the power line to other components. Without correct decoupling on the blind spot detection system, the microwave sensor does not operate correctly.

IX. CONCLUSION

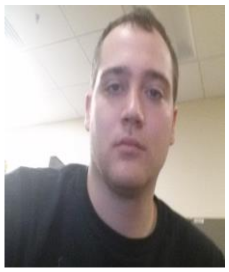
This paper explained the design process of the DMS. The prototype of the DMS was mostly successful, although changes were made from the original idea of the product. The wireless communication setup proved to be a challenging task as only one Bluetooth device can be connected to the driver's Android device at a time. Therefore the topology of the wireless networked was changed to fit the limits of an Android device. Obtaining a useable signal from the Doppler module also proved to be an issue. This was due to noise seen from LEDs located on the Bluetooth modules within the blind spot detection system. It was important to ensure that all of the devices were decoupled correctly to obtain the best signal. Due to time constraints the camera was originally intended to be done wirelessly, but was attached to the Android device via USB.

Many of the design specifications of the DMS were met such as, low power consumption, real time and long term fuel efficiency analysis, and detection of other vehicles within 10 ft. while driving. The total cost of the project came to a little over 300$ but was near the desired goal.

X. Acknowledgements

We would like to thank everyone who has assisted our group throughout the semester. A special thanks to Mr. Bill Mote for assisting us with issues regarding Android and Bluetooth connectivity. We would also like to thank Dr. Richie for his assistance and advice throughout the course of the project. Additional thanks to Dr. Kenneth Stanley, Dr. Alireza Seyedi, and Dr. Chung Yong Chan.

XI. BIOGRAPHIES



Aaron Kost is attending University of Central Florida and will receive his Bachelor's of Science in Computer Engineering in May of 2014. Aaron is currently seeking employment in related fields.

Sarah Bokunic is attending the University of Central Florida and will receive her Bachelor's of Science in Computer Engineering in May of 2014. After graduating, she plans to work as a programmer for a small company and is considering continuing her education in graduate school for Computer Engineering.

Victor Medina is a senior at the University of Central Florida studying Electrical Engineering and will graduate in May 2014.