A Mobile Application for Safer-Intelligent Driving and Vehicle Preventive Maintenance Using Vehicle OBD Data

Adnan Shaout and Abdur Rafay Mir

The Electrical and Computer Engineering Department
The University of Michigan-Dearborn
Dearborn, Michigan 48128
shaout@umich.edu; armir@umich.edu

Abstract - The purpose of this paper is to develop an app that could read the vehicle diagnostic trouble codes real time data through the on-board diagnostics (OBD-II). A regular driver might not be able to comprehend all of that data and the use of this app which could make it confined to only those with the knowledge of OBD-II. Keeping this mind, warning/notification feature was added into this smart phone app which will notify the driver about any malfunction or parameters breach with a warning sign and a beep. The goal of this paper is to achieve maximum awareness among drivers and vehicle owners. We also aim to make every driver more educated about their own vehicle and its maintenance through the use of this technology, which will not only help them in saving time and money but also makes vehicles much safer, reliable and fun to drive.

Keywords: OBD II, Mobile Application, Intelligent Driving, Vehicle Preventive Maintenance

1. INTRODUCTION

A vehicle is a complicated machine which not only consists of the mechanical components but also a large variety of electrical sensors, Integrated circuits, multiple electronic modules and a central processing unit known as ECM/ECU. This kind of combination of electrical, electronic and mechanical parts in a vehicle makes it one of the engineering marvels where all these parts are connected to each other in one way or the other. The Engine sensors reads most of the data and is major source of overall vehicle information followed by the chassis components such as suspensions, brakes etc. This information from various parts of the vehicle goes to the ECM and gets processed constantly [1] [2].

The In-vehicle information is very critical and accounts for the overall performance of any vehicle. When a sensor fails or starts malfunctioning, it comprises the vehicle's reliability and safety, and could also lead to a major mechanical component failure [2].

Therefore, it is very important to monitor this information from time to time in order to keep a vehicle in a good working condition. It not only helps in preventive maintenance reducing the risk of mechanical failure but also makes a driver more aware and educated of the working condition and day to day performance of his/her car [2].

Unfortunately, most of this information is not available to drivers at hand and it can only be accessed by scanning an OBD system [3].

The motivation behind this paper is to provide ease of access to the vehicles critical information to almost anyone and everyone without putting any price tag to the service or needing to visit any auto parts store. We tend to hear it quite often about the vehicle diagnostics mishaps by dealerships and mechanics. Sometimes simple issues or mechanical defects are misdiagnosed and good parts being replaced for being suspected as faulty, and eventually customers end up paying a huge repair bills yet the original issue remains unfixed. Here, the customer not only loses his valuable time and money but also undergoes high level of frustration.

The paper is organized as follows: section 2 will present some background on OBD, section 3 will present the design of the application, section 4 will present the OBD-II Modes and Parameter IDs (PIDs), section 5 will present DCT, sections 6, 7, 8 and 9 will present the implementation of the system and the testing result and finally section 10 will have the conclusion.

2. BACKGROUND

Going before the 1950s, most disclosure of issues in automotive was accomplished with listening to abnormal sounds, using hand tools, smelling unusual odors, some mechanical gages and the rest by hand. As vehicles became more advanced, we entered into a new era of the 1960s, where we became more reliable on advanced instrumentations and devices for testing and diagnostics purposes [4].

On turning the key into ignition mode these days, the vehicle dash should flash all lights present in the information cluster including "Check engine" light or "Service Engine Soon" light in some other vehicles. This means that the vehicle has a proper self-diagnosing system which shall notify the driver in case of any malfunctions within the vehicle systems. This early notification by the vehicle is due to the presence of OBD-II system and helps to not only warn the driver for any problems but also in protecting the environment [5].

Recently, in the mobile application market, several applications have emerged that pair the power of a mobile device with the information available through the use of an OBD-II reader. These applications tend to be directed toward auto enthusiasts, developing features that concentrate on measuring vehicle performance and troubleshoot mechanical issues.

Over the last couple years, there's been an explosion of aftermarket gadgets that allow the OBD-II connection to transmit useful stats. Tools were sold in the market targeting one feature per device such as instant fuel economy reader, engine speed reader, fault codes reader, temperature reader, vehicle speed reader etc. However, in this project, we are coupling all these different information into one single device with the help of Android smart phone technology.

There are currently few devices in the market that accomplish the function of reading trouble codes from the vehicle. They cost of these devices range from $50 to $300, but they only deal with reading and clearing the error codes. AutoZone is nation's largest chain of automotive spare parts stores; they offer free service of checking error codes from the vehicle using the same devices. There are also other smart phone applications in the market; the most popular of them is "Torque" app available for all compatible smartphones for $6. It is a commercial app with over 1 million buyers already and counting, it can also perform almost all of the functions as our application; however we still have some additional features that are completely unique from the current market as will be shown later. Table 1 shows a brief comparison of different products with their respective attributes and features.

3. DESIGN & IMPLEMENTATION

Figure 1 shows the basic workflow of the application where vehicle ECU sends the data to the OBD-II port, at this time, both vehicle's OBD-II port and OBD-II reader exchange data. In this process, OBD-II reader requests the vehicle OBD-II to transmit the data and then reader forwards it to the android software. The final output can be seen on the Smartphone device which will display the data coming from the engine ECU to the user.

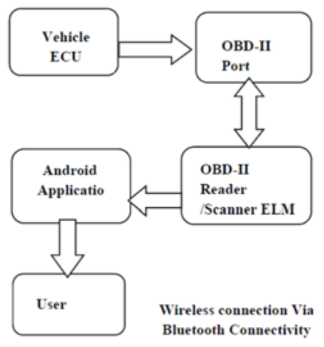

Vehicle ECU
OBD-II Port
Android Applicatio
OBD-II Reader /Scanner ELM
User
Wireless connection Via Bluetooth Connectivity

Figure 1- Block diagram of working process of this application

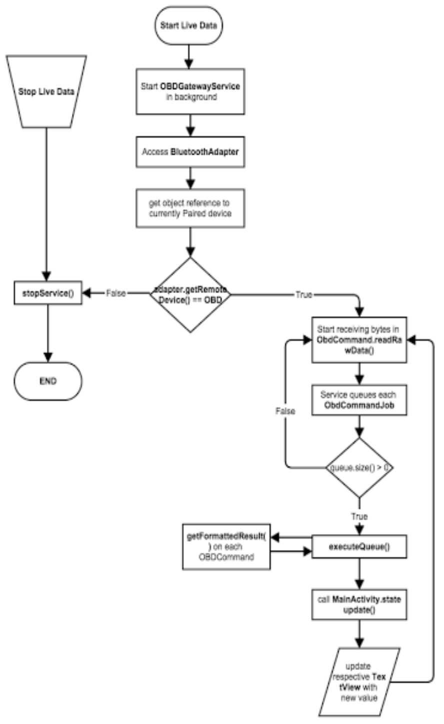

Start Live Data
Stop Live Data
Start OBDGatewayService in background
Access BluetoothAdapter
get object reference to currently Paired device
adapter.getRemoteDevice() == OBD
False
True
stopService()
END
Start receiving bytes in ObdCommand.readRawData()
Service queues each ObdCommandJob
queue.size() > 0
False
True
getFormattedResult() on each OBDCommand
executeQueue()
call MainActivity.stateupdate()
update respective TextView with new value

Figure 2- Basic design flowchart

The basic design of the project is shown in the form of a flowchart in figure 2.
The process starts with “Start Live Data” link which then triggers the OBD gateway/services on.
The application then tries to access the external hardware Bluetooth OBD scanner/adapter. If the Bluetooth adapter is already paired with the smart phone, the application continues to run, otherwise all the services are stopped and app is closed.
If pairing is already done, then the software starts receiving raw data in the form of bytes through OBD port. This raw data is organized in queue of different jobs. “***ObdCommandJob***” refers to the functions of our application such as engine speed, coolant temperature etc., all queued to be interpreted and displayed on the screen.
The data then goes to the decision zone where value of the function is confirmed, and with a right value, data goes to execute zone and gets updated on the screen.
The process can also be stopped by the external manual operation shown as “Stop Live Data” in the flowchart.
There are a huge number of sensors (nodes), for example, different engine component sensors, transmission sensors, ABS sensors etc. Each one of those speaks with a host processor of the vehicle by means of a controller area network (CAN) Controller [4] [6].

4. OBD-II Modes and Parameter IDs (PIDs)

A PID is a special kind of command/code that OBD assigns to a particular set of data. To exchange data with Engine control unit (ECU) through OBD-II, the OBD-II needs to send proper PID to the ECU, and in return ECU responds back to that information request in the form of bits and bytes. Those bytes are mostly displayed in the form of hexadecimal format [6].
The OBD-II standard does not oblige auto makers to execute all PIDs. It doesn't even give a base for a few modes, for example, Mode 1 and Mode 2 PIDS. On the other hand, many makers use the most widely recognized ones, for example, speed and RPM of vehicle [6].
Since there are diverse classes of solicitations, the OBD-II standard splits the PIDS up into various modes. First OBD protocol SAE J1979 details a record of 9 different diagnostic test modes [6]. Table 2 shows various modes of diagnostic tests.

Table 2- Diagnostic Test Modes [6]

Mode 1	PIDs in this category display current real time data such as the results of the engine RPM sensor.
Mode 2	When a fault or malfunction occurs, a snap shot of all mode 1 sensors are taken. This snap shot is known as a freeze frame. To access each individual sensor, you use the mode 2 requests
Mode 3	Sending a mode 3 request, the ECU responds with a list of DTCs stored if any.
Mode 4	Sending a mode 4 request, the ECU clears the DTCs stored and turns off the malfunction indicator lamp (MIL) if on.
Mode 5	Test results from oxygen sensor monitoring
Mode 6	Test results from other types of tests
Mode 7	Show pending Diagnostic Trouble Codes
Mode 8	Control operation of on-board system
Mode 9	Responds with the vehicles identification number (VIN).

To send a request to the ECU you must specify the mode and the PID. So for example, if you want to view the current engine RPM, you would send a *010Ch* (hexadecimal) query to the ECU. The ECU would then respond with a few bytes of data for the response. If you wanted to see the engine RPM stored value when a fault occurred last on the vehicle, you would instead send a mode 2 query, *020Ch*.
As you can see the query or command to send to an ECU is a combination of the mode and the relevant PID. All requests must adhere to this request format.

5. Interpreting Diagnostic Trouble Codes (DTCs)

“There are four main types of DTC codes defined by the SAE standards. These are as follows:
First digit will be:
• Powertrain Codes (P codes) starting with 0 - 3
• Chassis Codes (C codes) starting with 4 - 7
• Body Codes (B codes) starting with 8 - B
• Network Codes (U codes) starting with C – F [6].”

These codes distinguish about where or what framework the deficiency occurred. The powertrain codes are the most widely recognized as they are mostly generic and engine related codes [6].

Figure 9 shows the arrangement of vehicle trouble codes. They are made of 5 digits. The digits are in hexadecimal order. The first digit of any trouble codes distinguishes the sort of code like to identify whether the code is chassis code or body code or a powertrain. In figure 9, you can see the 5 digits that distinguish what class of codes it has a place with. The other 4 digits in the code distinguish other data. For instance the second digit distinguishes that it is a standard SAE characterized code while the third digit recognizes what framework created the issue [6].

.

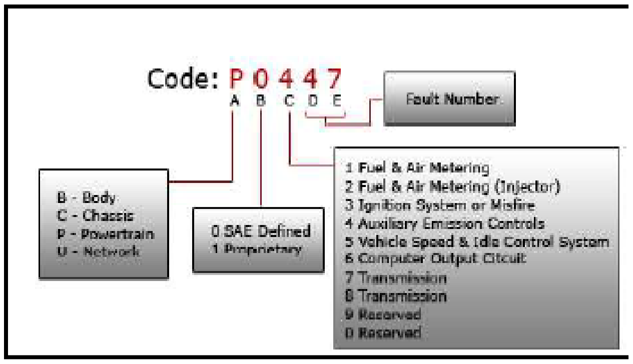


Figure 3- Example of Diagnostic Trouble Code

The code in figure 3 means "Evaporative Emission Control System Vent Control Circuit Open" which is a trouble code for malfunctioning exhaust system. We also see that it is a powertrain code as it starts with the letter "P" which is a standard code for all automobiles characterized by the SAE. Second digit is zero which is supposed to be manufacturer specific code. Then comes the third digit which as a value 4 to it and third digit is always an assigned to the list of vehicle components shown in figure 9. In this particular code for instance, it shows that the issue lies within the ***Auxiliary Emissions Control***. The ECU reacts with 4 hexadecimal bytes for each one code. The principal byte is in charge of parts A and B in the code above [6].

The list in the previous page gives the ranges of the first digit for each type of code. This concludes the most important parts of OBD-II that I needed to further research in order to gain an understanding of how to work with it. However, in this project, we are deriving our design source code from an open source android studio library that will automatically integrate all the PID and CAN data into the device. Modifications and debugging is still required in order to run the application. Also, please note that the open source library data only helps to interpret the data deriving from the vehicle OBD.

6. Implementation

Since we wanted to develop an android application, the logical language to choose was JAVA because the android SDK as all of the android APIs are built on top of the JAVA programming language.

In order to communicate with the OBD-II device we used the OBD-android API wrapped around the OBD-JAVA API. These APIs have helper classes and methods that can be accessed to enable receiving and making sense of the data from the OBD-II adapter.

Abstract class ***ObdCommand.java*** is the base class representing all incoming OBD commands. All the subclasses of ***ObdCommand*** like ***EngineLoadObdCommand.java*** are specific to different OBD commands and allow properly interpreting the OBD data.

ObdGatewayService.java is an android background service that listens for incoming OBD data and routes it to the specific handler for parsing and presenting to the user interface on ***MainActivity***.

In this paper, we used Android Studio as our platform for developing this Android application. Android Studio is an integrated development environment (IDE) for the Android platform and is free for all the developers. Based on JetBrains' IntelliJ IDEA software, the Studio is designed specifically for Android development. Major advantage of using this platform was the ease in debugging the application. Android Studio enables you to debug apps running on the emulator or on an Android device. With Android Studio, we can a) Select any device to debug our app on b) View the system log, c)Set breakpoints in your code, d) Examine variables and evaluate expressions at run time, e) Run the debugging tools from the Android SDK, f) Capture screenshots and videos of your app [7][8].

7. Project Application functions

This app is designed to read and display the following information in real time to the driver. Table 3 shows the list of all functions used in this application. All these functions have been tested in different environments and setups, which is explained in detail in the next section.

Table 3: List of functions [9]

Compass	Integrated in the app to show the direction of vehicle
Speed	Shows the speed and warns/notifies if reaches 75 mph

Engine RPM	It is a measure of crankshaft rotation of the engine per minute
Engine Coolant Temperature	It is used to determine the temperature of the engine coolant
Engine Load	Engine load is a measure of work being done by an Engine
Engine Runtime	Engine Runtime is a measure of time from the start of the engine until engine shut off
MAF	It is used to find out the mass flow rate of air entering a fuel-injected internal combustion engine
Air-Fuel Equivalence Ration	Air–fuel ratio (AFR) is the mass ratio of air to fuel present in a combustion process
Throttle Position	This function tells us the exact throttle position of the vehicle at any given time
Timing Advance	It is the measure of ignition spark fired by the spark plugs at a given time
Ambient Air Temp	Ambient air temperature is actually an outside Air Temperature
Air Intake Temp	Air intake temperature is actually an air temperature inside the intake manifold, should be very close to Ambient air temperature
Barometric Pressure	It is actually an atmospheric pressure of the vehicle's surroundings
Intake Manifold Pr	Intake Manifold Absolute Pressure (MAP) displays manifold pressure. Normally, it should always be in a state of vacuum
Fuel Economy	The fuel economy of an automobile is the fuel efficiency relationship between the distance traveled and the amount of fuel consumed by the vehicle
Fuel Level	Fuel Level Input is the percentage of fuel with 0% equaling tank is full and 100% when tank is empty
Fuel Pressure	Fuel rail pressure is the fuel pressure at the engine when reading in reference to atmospheric pressure
Trouble Codes	A trouble code is an alphanumeric value that corresponds to a particular type of fault in a vehicle

7.1 ADDITIONAL FEATURE

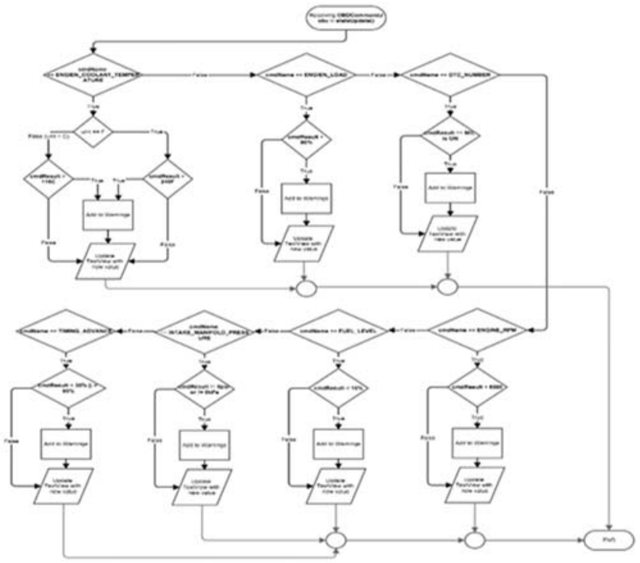


Figure 4 – The flowchart design of the warning/notification features.

In this paper, we have added some additional features that outstands it from the other similar apps available in the market today. The addition of "If- then" feature makes this app more users friendly and allows regular drivers who may not have any knowledge of automobile engines to understand the engine condition of their vehicles. This is achieved by introducing the ***Warning/notification feature/function*** that lets the driver know if any of the above listed functions exceeds the recommended values of normal operating condition. For example, if the fuel level gets below 20 percent, the driver will immediately be notified of the low fuel warning, similarly if the ignition timing advance value gets out of the set limits, the driver will be notified that the vehicle spark timing/ignition is malfunctioning and that its time for a tune up. Figure 4 shows the flowchart design of warning/notification feature which uses if-else code.

i- **Warning/Notification Function**: This function/feature of this app will read and displays the warning notification with a sound for any the following functions reading out of spec.

ii- **Engine RPM:** When the engine RPM exceeds 6000 rpm, the driver will be notified it as a warning, because it can cause wear and tear of the other mechanical parts in the vehicle.

iii- **Speed:** When the vehicle crosses 75 MPH, the driver will be notified to slow down as it could be dangerous

iv- **Engine Load**: In this app, the set limit of engine load is 90 percent, once it gets crossed, the app will pop up a warning notification. Increased engine load can cause severe wear and tear on the engine and can even damage the engine

v- **Engine Coolant temp:** Engine coolant temperature is directly associated with overall engine temperature. If this exceeds the set limit, the driver will be notified of the overheated engine and advised to stop the engine.

vi- **Timing advance:** If the ignition timing advance value gets out of the set limits, the driver will be notified that the vehicle spark timing/ignition is malfunctioning and that its time for a tune up.

vii- **Trouble Codes:** The driver shall be notified every time a trouble code pops up. This app will also show the definition of the code so the driver gets aware of the issue immediately.

8. GRAPHICAL INTERFACE

The GUI contains grouping of related fields/functions into 4 different groups and each group is named and assigned with an icon. After clicking on the icon, the list of fields/functions will appear at the bottom. Here is what it looks, see below pictures

Figure 5 shows the home screen of the application. As you can see, only 4 fields are shown here, namely:

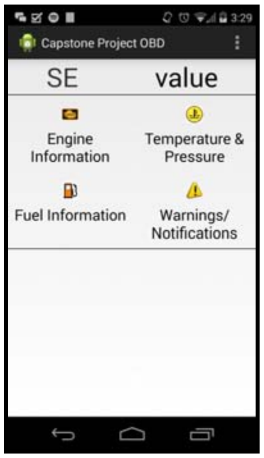
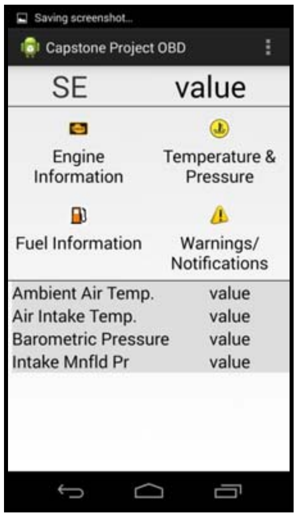


Figure 5- New Interface main screen

1) Engine Information
2) Temperatures & Pressure
3) Fuel Information
4) Warnings/Notifications

For example, figure 6 shows the real time data of different temperatures and pressures of the vehicle.

Warning/Notifications is a new feature being added into this app. This feature will use the "If then" values and will display the warning or notify the driver if any real time value exceeds the recommended value (recommended by auto manufacturer).



Ambient Air Temp.	value
Air Intake Temp.	value
Barometric Pressure	value
Intake Mnfld Pr	value

Figure 6- Temperature & Pressure Information screen

9. TESTING

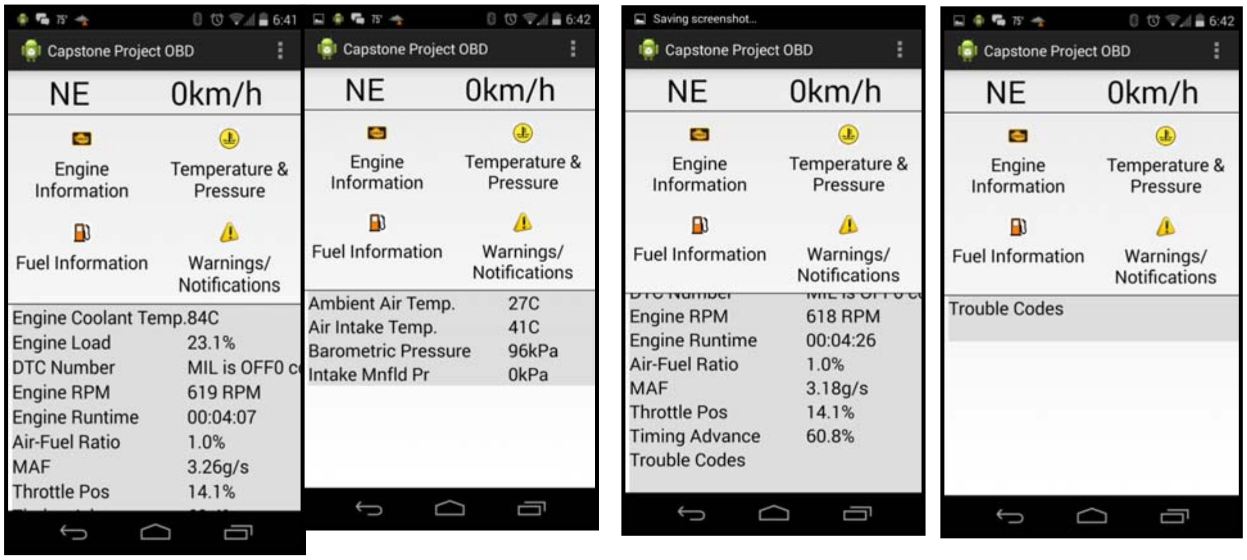
Testing is the last and most important part of any project. In this project, we have conducted testing at multiple stages with certain features of the application being tested at each stage. All features of the app have been tested multiple times with different hardware set up.

9.1- Hardware Testing

Our hardware includes smart phones, Bluetooth OBD-II scanner/adapter, and vehicle OBD-II port and test vehicles.

a- **Smart Phones**: Smart phone used in this project is a Nexus 5 Android Smart phone that runs Android software 4.4.4 Kitkat version. Two different Nexus 5 devices are used and prior to running the application, devices and their Bluetooth connectivity were tested as fully functional and OK for use

b- **Bluetooth OBD-II Scanner**: In this paper, we used an ELM327 OBD-II scanner/adapter sold by BAFX [10]. To ensure the accuracy of this device, we purchased two of these and



Capstone Project OBD

NE 0km/h

Engine Information | Temperature & Pressure | Fuel Information | Warnings/ Notifications

Engine Coolant Temp. 84C
Engine Load 23.1%
DTC Number MIL is OFF0 c
Engine RPM 619 RPM
Engine Runtime 00:04:07
Air-Fuel Ratio 1.0%
MAF 3.26g/s
Throttle Pos 14.1%

Ambient Air Temp. 27C
Air Intake Temp. 41C
Barometric Pressure 96kPa
Intake Mnfld Pr 0kPa

Engine RPM 618 RPM
Engine Runtime 00:04:26
Air-Fuel Ratio 1.0%
MAF 3.18g/s
Throttle Pos 14.1%
Timing Advance 60.8%
Trouble Codes

Trouble Codes

No Codes found

Figure 7- Test results of Ford Taurus

compared the results of both in order to verify the accuracy of the scanner. Our scanners were tested and found to be OK.

c- **Test Vehicles**: Since the goal was to develop an app that is robust enough to be run on different platforms and different vehicles, multiple test vehicles were used as part of the testing plan. These vehicles included 2012 Ford Taurus, 2012 Nissan Altima, 2002 Toyota Camry, and 1998 Honda Accord.

d- **OBD-II Ports**: Prior to final tests on the above vehicles, their OBD-II ports were test by visual inspection and electrical functionality. These ports and pins needs to be assured as functional, as sometimes ports do get clogged with debris and carbon. In addition to that, I have also visually inspected the back of the connector to make sure that all of the wires running from the contacts to their respective destinations are intact. Before proceeding for additional tests, they were confirmed to be OK and fully functional.

9.2 - Software Testing

Android Application testing: The application has been tested on multiple devices and vehicles. In this section, we will present all our test results conducted on different vehicles

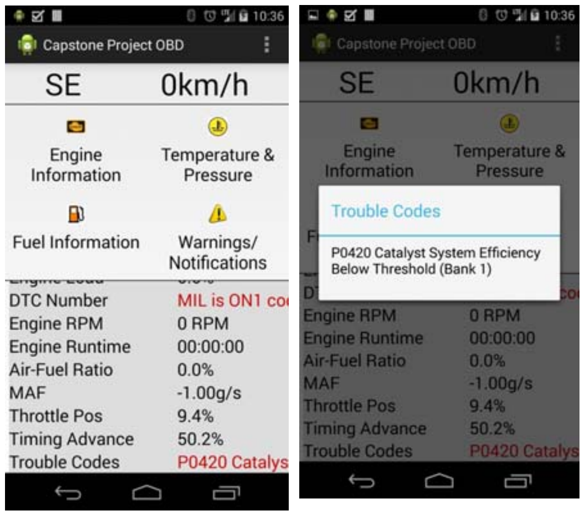
Test Results

a) Figure 7 shows the test results a **2011 Ford Taurus** SEL (3.5 l Engine/ Front wheel drive)

Test result conclusion for Ford Taurus: All functions are tested OK and showed accurate readings. Since there was nothing wrong with this particular vehicle, no function value exceeded the assigned parameters and therefore no warnings/notification or trouble codes were found in this vehicle.

b) Figure 8 shows test result of a **2000 Honda Accord**

Test results conclusion for Honda Accord:



Capstone Project OBD

SE 0km/h

Engine Information | Temperature & Pressure | Fuel Information | Warnings/ Notifications

DTC Number MIL is ON1 co
Engine RPM 0 RPM
Engine Runtime 00:00:00
Air-Fuel Ratio 0.0%
MAF -1.00g/s
Throttle Pos 9.4%
Timing Advance 50.2%
Trouble Codes P0420 Catalys

Trouble Codes

P0420 Catalyst System Efficiency Below Threshold (Bank 1)

Figure 8- Test results of Honda Accord

As we can see in the figure 8, this vehicle has a trouble code P0420. DTC number reads MIL is ON1 Code, which means Malfunction indicator lamp is ON and it has 1 number of code. Then once you click on the code at the bottom, a dialog box pops up and it reads the code definition, as ***"P0420 – Catalyst System Efficiency Below Threshold".***

10. CONCLUSION

The main goal of this paper was to develop an app that could read the vehicle diagnostic trouble codes, later additional goals were added, i.e. to include the OBD-II real time data. Thereafter, it is realized that a regular driver might not be able to comprehend all of that data and the use of this app could be confined to only those with the knowledge of OBD-II. Keeping this mind, warning/notification feature was added into this app which will notify the driver about any malfunction or parameters breach with a warning sign and a beep.
This paper has demonstrated a very unique and high level design from connecting an Android Bluetooth with OBD-II Bluetooth adapter to reading the critical vehicle data.
Future work on this app will include integration of GPS into the real time data, so that a driver can see the vehicle's performance at specific time and location using GPS coordinates. Also, we would like to add the gas station tracker into this app, so when the fuel level is low, the app shall locate the nearest gas station and notify the driver.

REFERENCES

[1] Engine control unit- Wikipedia, the free encyclopedia Source: http://en.wikipedia.org/wiki/Engine_control_unit
[2] List of sensors- Wikipedia, the free encyclopedia Source: http://en.wikipedia.org/wiki/List_of_sensors
[3] On-board diagnostics - Wikipedia, the free encyclopedia Source: en.wikipedia.org/wiki/On-board_diagnostics
[4] Creating a Wireless OBDII Scanner- A Major Qualifying Report Source: https://www.wpi.edu/.../FinalPaper.pdf [8] OBD by EPA Source: http://www.epa.gov/obd/questions.htm
[5] OBD-II published journal Source: eecs.ucf.edu/.../g09/.../originalreport.pdf
[6] An Embedded Automotive Monitoring Device Source: http://automon.donaloconnor.net/files/fypreport.pdf
[7] Android API open source library for OBD Source: https://github.com/pires/android-obd-reader
[8] Android Studio | Android Developers Source: https://developer.android.com/sdk/installing/studio.html
[9] OBD2 PID Reader Source: http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1089&context=cpesp
[10] ELM327 Bluetooth scanner/reader by BAFX Source: http://www.bafxpro.com/bafx-products-tm-pic18f2480-bluetooth-obd2-scan-tool-for-check-engine-light-and-other-diagnostics-for-android-only