

Design and Implementation of Android-Based Vehicle Diagnostic System

Solomon Oluwole Akinola

Computer and Electrical/Electronic Engineering
Department, Afe Babalola University
Ado-Ekiti, Ekiti State, Nigeria
soakinola@abuad.edu.ng

Damilare Ogunmola

Computer and Electrical/Electronic Engineering
Department, Afe Babalola University
Ado-Ekiti, Ekiti State, Nigeria
dogunmola0@gmail.com

Abstract—an On-Board Diagnostic standard was developed and integrated with an Android-Based Vehicle Diagnostic System (ABVDS). ABVDS was installed on an Android smartphone and tested on a 2012 Hyundai Accent and 2000 Lexus RX300 vehicle models. The result established that low cost application can be developed for automobile mechanics using android smartphone. This will help avoid wrong diagnosis of vehicles and also reduce the cost of repairs.

Keywords—Android; automobile; diagnostics; Bluetooth; smartphone; software

I. INTRODUCTION

In addition to traditional engine parts, modern cars are equipped with gears for identifying automobile faults. Embedded systems are designed to control many of the components in most modern cars today. These computers control everything from air/fuel mixture ratio, ignition timing to air bags and the anti-lock brakes.

These embedded computers are collectively called the Engine Control Unit (ECU). It uses the information gathered from dozens of sensors placed strategically in the automobiles and performs millions of calculations before a decision is made [1].

The ECU became a very powerful tool for the delivery of physical and diagnostic data to automobile technicians. This led to the development of a standard called On-board Diagnostics (OBD). OBD was necessary to encourage reliable emission control systems by automobile manufacturers.

In the mid-'90s, a new standard was introduced that provides almost complete engine control and also monitors parts of the chassis, body and accessory devices, as well as the diagnostic control network of the car. The new standard is called OBD-II.

The ECU gathers information through a network of sensors in the automobile. The data collected are used for decision making by the ECU. The information logged by the ECU is used to trace faults in the automobile's operation. The OBD standard provides an interface for the user to access the information gathered by the ECU. Access to the sensors and diagnostic data gathered by the ECU via the OBD requires special hardware and software.

This paper presents a low cost and simplified diagnostic system compatible with most vehicles made after 1996. The interface is an Android

application which connects to a Bluetooth adapter attached to the vehicle. It uses a Bluetooth network to send requests and receives data which is converted by the application to a human readable format.

II. SYSTEM OVERVIEW

The Android-based application reads data from a vehicle's ECU by connecting to the ECU via a Bluetooth-enabled adapter. See Fig. 1 for the application illustration of the diagnostic system.



Fig. 1. Android-based OBD-II System

A. On-Board Diagnostics (OBD)

The OBD version II (OBD-II) was used for the implementation due to the added features. It is also the next generation standard (designed to reduce the time between fault occurrences). It detects and thereby repairs faults that can cause more hydrocarbon emissions. All modern vehicles must have the ECU system as it is required by law to be built into the vehicles. This legislation was done to maintain low hydrocarbon emissions in vehicles for both light and medium duty vehicles. The OBD is built into the automobile's ECU while the ECU controls everything from the spark timing to the lights and exhaust monitoring.

The OBD standard enables vehicle owners retrieve useful data about their automobile from the ECU. The

information provided describes the state of the various parts of the vehicle. This allows technicians and automobile users identify faults with the automobile and thereby commence with maintenance. The state of all the sensors can be monitored in real-time and the diagnostic trouble codes in the ECU logs can be accessed and possibly removed.

In 1960, the California Motor Vehicle Pollution Control Board (CMVPCB) was created to implement regulations for air quality control standards. [2]. In 1966, the US federal government and the state of California also used a cycle to certify vehicles. In 1970, the US congress passed the Clean Air Act which regulates air emissions [3]. This led to the creation of the OBD standard by the California Air Resources Board (CARB). Hence all vehicles sold in California from 1988 have an on-board diagnostic system to detect emission failures. As at the time of implementation, the embedded system used was then a simple ECU for the monitor of various sensors in the engine. [4].

However, the first version had many problems such as:

- It provided no clues about what led to the faults detected.
- Also, troubleshooting was expensive.

These shortcomings led to the development of a new version called the OBD-II in 1994. In 1996, all automobiles in California had the OBD II device installed during the manufacturing process.

More recent OBD-II systems have standard data link connector (DLC) with specific pin arrangements. The communication protocols also became standardized. The terminologies and diagnostic trouble codes (DTCs) were also standardized. However, automobile manufacturers were allowed to customize the device with specific codes.

There are presently five types of signalling protocols used with OBD-II. Although, a wide range of vehicles support only one standard. The protocols signify how the low level signalling are done in the automobile. These signals are [5]:

- SAE J1850 PWM (Pulse-Width Modulation – 41.6 kB/s standard of Ford Motor Company)
- SAE J1850 VPW (Variable Pulse Width – 10.4/41.6 kB/s, standard of General Motors)
- ISO 9141-2. This is used in Chrysler, European and Asian vehicles. It has an asynchronous serial data rate of 10.4 kbit/s.
- ISO 14230 KWP2000 (Keyword Protocol 2000). Electrically the same with ISO 9141.
- ISO 15765 CAN (250 kbit/s or 500kbit/s). This is the OBD requirement for all automobiles in the U.S. from 2008 onwards.

The OBD-II recommends a standard connector called the SAE J1962 connector [6]. The connector

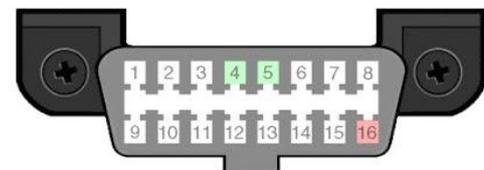
required should be around 3 feet from the driver at all times. The pinout is showed in Fig. 2.

B. Bluetooth OBD Adapters

The Bluetooth adapter acts as a bridge between the ECU and the Android application. The adapter plugs into the SAE J1962 port that comes with every OBD compliant vehicle. The port is reachable from the driver's seat. This is in accordance with the specification standard. Several adapters exist in the market with Bluetooth capability.

Majority of the adapters are built on a chip known as ELM327. The ELM327 was chosen for this project due to its support for all OBD protocols and low power consumption ca. The ELM327 Programming can be implemented using AT command instructions. The Android application can send Parameter ID (PID) requests (usually in two bytes) to the adapter (ELM327). The adapter then transfers the received data the ECU which replies with a set of 7-bytes data.

Data Link Connector (vehicle OBDII port)



- 1 Make/Model Specific
- 2 SAE J1850-PWM POS(+) or SAE J1850-VPW POS(+)
- 3 Make/Model Specific
- 4 Chassis Ground (all protocols)
- 5 Signal Ground (all protocols)
- 6 ISO15765-4 CAN-Bus High
- 7 ISO9141-2 K-Line or ISO14230-4 KWP2000 K-Line
- 8 Make/Model Specific
- 9 Make/Model Specific
- 10 SAE J1850-PWM NEG(-)
- 11 Make/Model Specific
- 12 Make/Model Specific
- 13 Make/Model Specific
- 14 ISO15765-4 CAN-Bus Low
- 15 ISO9141-2 L-Line or ISO14230-4 KWP2000 L-Line
- 16 +12v (always on) (all protocols)

Fig. 2. Data Link Connector (SAE J1962) (Source: [6])

C. Parameter IDs and ECU responses

The PID request is a two (2) bytes string. The first byte is usually the mode. This is represented as 01 to 09. The second byte is the PID request. To check the speed of the automobile, the system application will send the bytes "01 0D". This suggests that the PID "0D" is in mode 01.

The OBD then responds with a 7-bytes hexadecimal data. The first byte indicates the mode (40 must be subtracted from the value to get the actual mode). The second byte is the PID sent. The third byte contains the data received. Most PID responses use two (2) bytes only which are the third and fourth bytes. The other bytes will be padded by zeros.

This data acquired from the ECU will be converted to the actual values using the methods provided by the OBD standard.

The data acquired from the OBD can either be a DTC, real-time data or freeze frame data (snapshot of real-time data) from when an error occurred. We focused only on the DTCs and real-time data. Real-time data refers to the data acquired from the sensors in real-time such as speed, engine temperature etc. DTCs are the error codes caused by malfunction in the vehicle, e.g. to get the engine coolant temperature from the ECU, the PID 05 in mode 01 is sent to the ECU i.e. "0105". The response will be in the order 41 05 78 00 00 00 00.

To get the mode, i.e. 41- 40 = 01 computed i.e. mode 01. To calculate the coolant temperature will require converting "78" hexadecimal to decimal which is "120". This is then used to determine the conversion temperature in degree Celsius e.g. $120-40=80^{\circ}\text{C}$.

III. ANDROID APPLICATION SOFTWARE

There are several mobile operating systems (OSs) such as the Google Android, Apple iOS, Windows, Blackberry, Mozilla Firefox OS, Tizen etc. However, the Google Android OS has established itself as the market leader. Google Android has the largest market share of 76.6% followed by iOS 19.7% as at the fourth quarter of 2014 [7].

Google Android OS was chosen for this project with regard to the following benefits:

- It has the largest market share.
- It is supported by a wider range of devices as compared with iOS which runs only on Apple devices (iPhone, iPod and iPad).
- The hardware is accessible due to its open-source status.

Android is a mobile OS based on the Linux kernel and currently under development at Google Inc. The user interface is based on direct manipulation. It was designed primarily for touchscreen interfaces and used in a variety of devices such as smartphones, tablets, TVs, automobiles, wrist watches, game consoles, digital cameras and PCs.

The application developed for this project is known as Android-Based Vehicle Diagnostic System (ABVDS) ABVDS is tailored for smartphones and tablets devices. The user interface of an Android smartphone or tablet is based on direct manipulation by touch inputs. The user can swipe, tap, pinch and reverse-pinch to manipulate screen objects. These gestures are designed to mimic real life manipulation.

The application performs the following tasks:

- Connect to the Bluetooth adapter.
- Send PID requests to the ECU via the adapter.
- Receive data from the OBD.
- Convert the data to human-readable format.
- Display responses in an intuitive manner.

This sequence is shown in Fig. 3.

A. Android Application Development for ABVDS

The application was developed on a Windows 8.1 OS with the Android Studio. The Android Studio is an integrated development environment (IDE) for developing applications for the Android OS. Android Studio is the official IDE for Android application development. The IDE is based on the IntelliJ IDEA which is an IDE for Java programming. The Android Studio offers features such as a Gradle-based system for project development automation; application package (.apk) file generation and many templates for common application features.

B. Software Testing

The ABVDS application was tested and debugged for logical errors. Errors detected by the compiler were rectified prior to program compilation with the Android IDE. The Graphics User Interface (GUI) of the application was inspected during development by uploading the application onto an Android emulator on a computer with a Windows based OS.

The Android emulator is not designed to emulate Bluetooth device communication. To test Bluetooth connection, the ABVDS application was installed on a compatible Android smartphone and tablet.

Two vehicles were used as the test environment for the onboard diagnosis system. The first vehicle was a 2012 Hyundai Accent and the second was a 2000 Lexus RX300. The same Android device and OBD Bluetooth adapter were used for the vehicles.

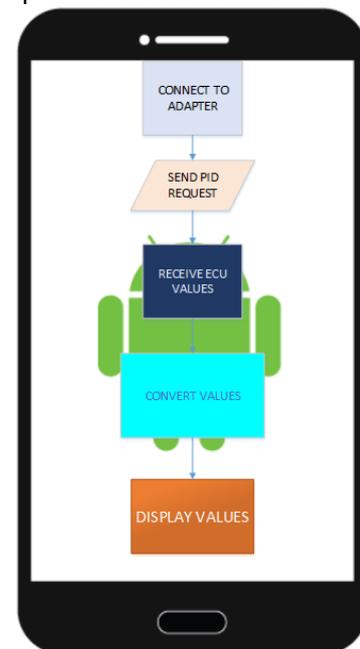


Fig. 3. Android application operational flowchart

1) *Bluetooth Connectivity Testing:* The ABVDS was tested on the Android IDE and the Bluetooth functionality of the application was tested with the adapter connected to the vehicle OBD II port. Upon cranking the engine of the vehicle, the connect button on the ABVDS application menu is selected. The ABVDS application will request to turn the Bluetooth device on (search for the ELM327 adapter). The

adapter will be selected from a list of Bluetooth devices. The ABVDS application adapter device name is OBDII. Once the device connects, the status on the GUI will change from not connected to connected. The ABVDS application connected to the adapter without a force stopped error for the two vehicles tested. Fig. 4 shows the ABVDS application before the Bluetooth device is connected.

2) *Real-Time Functions Testing:* Once connection to the Bluetooth adapter is established, the ABVDS application protocol updates the ABVDS application for real-time monitor and the diagnostic trouble codes in the ECU logs. In the two vehicles tested, the parameters speed in Kilometers per hour (Km/h) and frequency of rotation in (revolution per minute) 100xRPM were updated in real-time. However, there was few seconds delay in the out response of the ABVDS application to the vehicle on-board mechanical meters. This was due to delay in ODB data sent and received via the Bluetooth adapter and processing the result obtained.

The Engine Load function displayed variation as expected when tested with 2012 Hyundai Accent but when tested with the 2000 Lexus RX300, the Engine Load function displayed 0%. Also, the battery voltage remained constant throughout at 0 V for both two vehicles. This was due to the vehicle torque application compatibility with the ELM327 adapter used for the project.



Fig. 4. The ABVDS application user interface

3) *DTC Function testing:* Testing for the number of DTCs will require selecting the ABVDS application Trouble Codes button. When tested with the 2012 Hyundai Accent, the DTC returned 0 which suggests that there are no DTCs i.e. no error codes found. In the second vehicle i.e. the 2000 Lexus RX300 displayed 22 DTCs available. This suggests that 22 error codes were found. The results were verified using the Torque application commercially available software for onboard diagnostic system). The result correlated with ABVDS application.

TABLE I. SUMMARY OF ABVDS APPLICATION DIAGNOSIS REPORT

| S/N | TABLE COLUMN SUB HEAD | 2012 HYUNDAI ACCENT | 2000 LEXUS RX300 |
|-----|-----------------------|---------------------|------------------|
| 1 | BLUETOOTH ADAPTER | CONNECT | CONNECT |
| 2 | PARAMETER SPEED | 0 -180 KM/H | 0 -180 KM/H |
| 3 | FREQUENCY OF ROTATION | 100 x RPM | 100 x RPM |
| 4 | ENGINE LOAD FUNCTION | CHANGED | NO ACTION |
| 5 | BATTERY VOLTAGE | NO ACTION | NO ACTION |
| 6 | TROUBLE CODES | NO DTCs | 22 DTCs |

IV. CONCLUSIONS

The project demonstrates the combination of low-cost hardware and a user-friendly ABVDS application to create a real-time OBD standard Android application.

The ABVDS application demonstrated a functional mobile application. The ABVDS application was developed to support the ELM327 and STN1110 integrated chips. This can provide useful functionality to automobile mechanics and technicians. The ABVDS application will help reduce false diagnosis and reporting which is peculiar to automobile mechanics. Also, a demo version ABVDS application will be available on the Google play store. This will allow feedback and further improvement on the ABVDS application.

ACKNOWLEDGMENT

The authors would like to thank Dr. Tunde Adegbola of African Language Technology Initiative (Alt-i) for his support in the development of this project work.

REFERENCES

- [1] K. Nice, "How Car Computers Work," 11 April 2001. [Online]. Available: <http://auto.howstuffworks.com/under-the-hood/trends-innovations/car-computer1.htm>.
- [2] Ganesan, "Introduction to On Board Diagnostics (II)," 2009. [Online]. Available: http://groups.engin.umd.umich.edu/vi/w2_works_hops/OBD_ganesan_w2.pdf.
- [3] United State Environmental Protection Agency (EPA), "Summary of the Clean Air Act," 12 November 2014. [Online]. Available: <http://www2.epa.gov/laws-regulations/summary-clean-air-act>.
- [4] Clear Vehicle Diagnostics, "Short History of On-Board Diagnostics," 28 December 2012. [Online]. Available: <http://www.clearvehiclediagnosics.com/news/21/short-history-of-on-board-diagnostics.aspx>.
- [5] SAE International, "On-Board Diagnostics for Light and Medium Duty Vehicles Standards Manual, Pennsylvania: Society of Automotive

Engineers Inc, 2003.

[6] ScanTool, "Can't connect OBDLink LX to JOBD vehicle!!!" 2015. [Online]. Available: <https://www.scantool.net/forum/index.php?topic=9560.0>.

[7] International Data Corporation, "Smartphone OS Market Share 2014, 2013, 2012 and 2011," 2015. [Online]. Available: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>.