WIRELESS REAL-TIME VEHICLE MONITORING BASED ON ANDROID MOBILE DEVICE

Martin ČABALA[*], Ján GAMEC[**]
[*]Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, e-mail: martin.cabala@student.tuke.sk
[**]Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Park Komenského 13, 042 00 Košice, Slovak Republic, e-mail: jan.gamec@tuke.sk

ABSTRACT

The subject of this article is to bring the problem analysis of vehicle communication through diagnostic interface OBDII with diagnostic tools used in automotive diagnostics with an emphasis on Bluetooth communication. Based on the analysis, this paper deals with the description of software design and its implementation to mobile devices on purpose of communication with CAN OBDII interface. For this type of communication is usually used diagnostically-oriented communication processor ELM 327. Aim of this solution is to bring an application provided for mobile devices with Android operating system, which can obtain real time data from engine control unit through ELM 327.

Keywords: *Android, Bluetooth, CAN, car diagnostics, ELM 327, OBDII*

1. INTRODUCTION

In early 60-ies vehicles were built of few simple mechanical parts and only simple electrical circuits. Nowadays only easy quick look at the vehicle was worth to discover defect. As the time was passing by, more complicated vehicles started to be and more time it took to discover the defect. Need for faster and more accurate defect detection was growing. Current vehicles include buses and without proper diagnostic tools it would be impossible to detect and locate a specific fault. To profile, design and implement software for mobile device for such a fault detection and real time data collection is essential knowledge of diagnostic tools, CAN OBD II and chosen communication technology required.

2. AUTOMOTIVE DIAGNOSTICS

Establishment of automotive diagnostics is dated back to the end of 60-ies of the 20th century, when the State of California has mandated a law that required emission control systems on cars to solve the smog problem. The US federal government first extended and expanded a version of California's controls nationwide in 1968. The State of California took this system a step further in 1985, decreeing that a more detailed system to be installed, beginning with 1988 model-year cars and light trucks sold in California. This new system was known as On-Board Diagnostics, or OBD. In 1996, California and the SAE enhanced OBD once more, and this became On-Board Diagnostics, Generation Two, or OBD-II, first federally required for vehicles assembled by January 1, 1996. Today, OBD II is the international standard for communication between automobiles and diagnostic testers. The OBD II standard defines hardware, electrical signaling, and message formats for communicating with a vehicle. In addition, the OBD II protocol also specifies certain data that must be accessible in a vehicle's Engine Control Unit (ECU).

2.1. Diagnostic tools

Automotive diagnostics can be divided into general and specialized. Specialized diagnostics can be accomplished only with the appropriate, often not a cheap tool in conjunction with the official database of the manufacturer. This method is limited only to a domain-knowledge expert. On the other hand, the general diagnostics can be accomplished with standardized codes, understandable for almost laymen and offers plenty of information about a vehicle that can be practical also for experts.

2.2. Communication protocols

For clear communication with the engine control unit there were designed diagnosis communication protocols (description of the signals, the speed of communication...). There are five signalling protocols that are mostly used in the OBD-II interface. Many resources are talking about the number 9, or even more. This is because variants of protocols are mistakenly counted to be different communication protocols [1]. Most vehicles implement only one of these protocols. It is often possible to deduce protocol on the pins that are used for standardized connector. However, it should be mentioned that if vehicle supports one of the communication protocols it does not have to support OBD II interface. Having the two different vehicles, which support communication on the same communication protocol, one may support the OBD-II standard (emission diagnostics), but the second may not [3].

2.3. Diagnostic modes

The standards require that each OBD command or request that is sent to the vehicle must adhere to a set format. The first byte sent (known as the 'mode') describes the type of data being requested, while the second byte (and possibly a third or more) specifies the actual information required. The bytes which follow after the mode byte are known as the 'parameter identification' or PID number bytes. The modes and PIDs are described in detail in documents such as the SAE J1979, or ISO 15031-5 standards, and may also be defined by the vehicle manufacturers [2].

Vehicles are not required to support all of the modes, and within modes, they are not required to support all possible PIDs. Within each mode, PID 00 is reserved to show which PIDs are supported by that mode. Mode 01, PID 00 must be supported by all vehicles.

Table 1 OBD II communication protocols

Communication protocol	Communication speed (kb/s)
• SAE J1850 PWM	41.6
• SAE J1850 VPW	10.4
• ISO9141-2	10.4
• ISO14230-4 (KWP2000)	10.4
• ISO 15765-4/SAE J2480	250 - 500

Table 2 SAE 1979 standards currently possible diagnostic test modes

Mode	Description
01	Show current data
02	Show freeze frame data
03	Show diagnostic trouble codes
04	Clear trouble codes and stored values
05	Test results, oxygen sensors
06	Test results, non-continuously monitored
07	Show 'pending' trouble codes
08	Special control mode
09	Request vehicle information
0A	Request permanent trouble codes

3. ELM327

ELM 327 belongs to general OBD Interpreter interfaces that are designed for usage with vehicles that use one of the standard OBDII (On Board Diagnostics) protocols [2]. Interpreter is queried by AT or OBD commands. AT commands are recognized as internal commands and are targeted to the ELM 327, OBD commands are routed through the gateway to the engine control unit or other control units.

3.1. AT Commands

ELM327 recognizes the AT command as each command beginning with "AT" and ending with a newline (\n). If the command had adjusting nature, the ELM327 sends "OK" reply if successful. Some commands require a number as argument. These numbers are always hexadecimal and must be written in pairs. Normally there is no need to change any of AT adjustments before communication is established with the vehicle. AT commands for ELM 327 is described in detail in ELM manufacturer official website - data sheets.

3.2. OBD II commands

All commands that do not start with the letters "AT" are considered to be OBD commands for the vehicle. Each pair of bytes will be tested if it is really a hexadecimal digit, and then will be sent. OBD commands are composed into data packets and are sent to the vehicle. Most of the standards require 3 bytes as a header and byte error (checksum) as message format [2]. ELM327 adds these bytes to the message according to used protocol and user does not have to care about it.

3.3. DTC interpretation

A very common usage of ELM327 is the detection of diagnostic fault codes. Diagnostic fault codes can specify the exact fault in the vehicle if MIL (Malfunction Indicator Lamp) is currently on. Some of the DTC (Diagnostic Trouble Code) are standardized but there are many codes known only by certain manufacturer. Diagnostic mode 03 is used to obtain this kind of information. Firstly, user can retrieve number of stored DTC's. Using the mode 01 PID 01 is needed to process this requirement. If number of stored DTC's is clear, user can retrieve these codes with mode 03. Typical answer could look like:

```
43 01 33 00 00 00 00
>
```

43 indicate the response to the mode 03. Remaining 6 bytes are read in pairs to obtain the DTC (in this case would be interpreted as 0133, 0000 and 0000). The answer was concatenated by zeros but zeros according to SAE standard are not recognized as a DTC. In each received code, the most significant bit identifies more information. The easiest way to obtain extra bit is to use the table below:

Table 3 MSB representation in DTC retrieved code

MSB	MSB repl.	DTC Description
0	P0	Powertrain Codes – SAE defined
1	P1	" " - manufacturer defined
2	P2	" " - SAE defined
3	P3	" " - jointly defined
4	C0	Chassis Codes – SAE defined
5	C1	" " - manufacturer defined
6	C2	" " - manufacturer defined
7	C3	" " - reserved for future
8	B0	Body Codes – SAE defined
9	B1	" " - manufacturer defined
A	B2	" " - manufacturer defined
B	B3	" " - reserved for future
C	U0	Network Codes – SAE defined
D	U1	" " - manufacturer defined
E	U2	" " - manufacturer defined
F	U3	" " - reserved for future

MSB: Most Significant Bit
MSB repl.: Most Significant Bit replacement
DTC: Diagnostic Trouble Code

3.4. Communication interfaces

There are number of different possible interfaces that can be used in order to communicate with ELM 327. In principle, converters can be divided into data transfer via cable or wireless transmitters. The oldest are converters with RS 232 interface, which is now obsolete.

Table 4 Comparison of transmission of data between the diagnostic equipment and other equipment

	Speed	Wireless	Energy
RS232	< 19 Kb/s	No	Normal
USB 2.0	480 Mb/s	No	Low
Bluetooth	2,1 Mb/s	Yes	Normal
WIFI	22 Mb/s	Yes	High

Bluetooth: Bluetooth 2.0 + enhanced data rate
WIFI: WIFI IEEE 802.11g

4. MOBILE DEVICE SOFTWARE FOR COMMUNICATION WITH DIAGNOSTIC INTERFACE ELM 327

Android is now dominating operating system in a lot of industries and not only in smartphones and tablets. There can be found many topics related to Android usage in other gadgets like watches, belts, and even mirrors. Since Android is becoming more famous in other industries, it is also finding a place in the car industry. First car concept which included Android OS was Roewe 350 N1 concept in 2009. Its usage in a car brings the possibility of features such a street view based on sat-nav, voice activated calling and live traffic. The 350 N1 is marketed as the one bringing high end features normally found in luxury cars to a more affordable vehicle. The software allows the driver to use navigation avoiding traffic with live traffic updates over the air, as well as finding business places and places with live up to date information on them (such as current offers available at business's for example). You can also browse the internet from the car [5].

Recently there was also presented first Android powered OEM car stereo. This Android powered gadget can access internet, install updates, and it contains navigation, audio player, accepts SD cards, USB devices, and will even connect to devices over Bluetooth [6]. The main advantage is that it can be extended with many other applications like a mobile device.

4.1. Existing solutions

The problem of communication between mobile devices supporting Android operating system and diagnostic module ELM 327 was attempted to be dealt with in the past. The applications that are trying to solve this problem include either applications with rich user interface, visualization of data in different fashion, or simple terminal applications.

The most known terminal applications are:

- ELM 327 Terminal
- alOBD Terminal

The most successful applications with rich user interface are:

- Torque Pro (OBD II & Car)
- Car Gauge Pro

Based on analysis of mentioned solutions, the main disadvantages of these solutions are:

- Absence of terminal view in rich user interface applications
- Absence of or not intuitive creation of data records and viewing them
- Very sophisticated user interface – annoying for most of users

Modern application for diagnostic usage for mobile device should contain following features:

- Obtaining real time values of OBD II standard
- Displaying actual values in more display forms
- Save the values obtained in a CSV formatted file or any other simple format
- Viewing the recorded files in graph representation
- Sending commands via the console interface
- Editing database
- Viewing the OBD II parameters
- Customizing the user interface
- Simple expandability for the future

4.2. New solution ELM Diag application

Diagnostics application ElmDiag allows wireless (Bluetooth) communication with the ELM 327 diagnostic adapter engaged in self-diagnostic car socket supporting OBD II standard. The application can read the real time values of the OBD II standard parameters and display them in graphically modifiable user interface or store the values in external memory of mobile device in CSV format. The user interface is designed to look like main screen of the Android OS. It consists of a variable number of screens on which user can add / remove widgets. Each "widget" is a different display form of the variable (analog indicator, digital, graph). Variables of OBD II standard are stored in a SQLite database. Implemented application consists of eight main activities and three auxiliary activities. The application removes all the drawbacks of previous solutions while maintaining all the advantages.

Application supports up to 40 OBD II values and can be easily extended with more. It is based on customized version of JAVA programming language. Communication with ELM 327 is request-response based and using the fastest car protocol for communication. Mobile device can obtain up to 6 responses in a second. This communication speed could be probably higher using native development kit (NDK) for Android.

4.3. User interface

User interface is highly customizable – user can regroup or delete views with drag and drop action and add with long press action. When the drag and drop state is initialized the trash icon is shown at the bottom of the screen instead of the navigation bar. There are 4 sizes of widgets that can be added to the user screen – the screen containing 4x6 grid of same sized cells:

- Tiny - 1x1 cell
- Small - 2x2 cells
- Normal - 3x3 cells
- Large - 4x4 cells

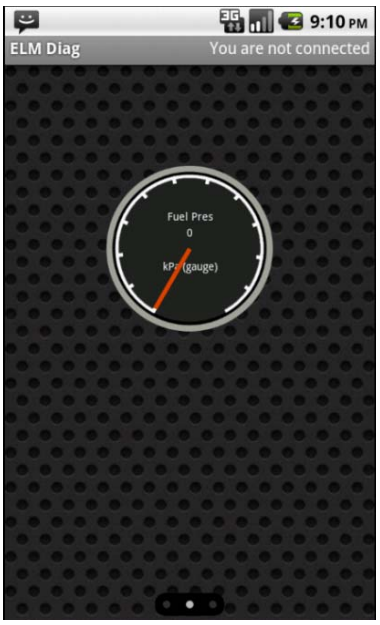

Fig. 1 Customizable user interface

Current solution offers three types of widget form – analog, digital or graph. Solution is also highly extensible for these widgets.

Analog widget

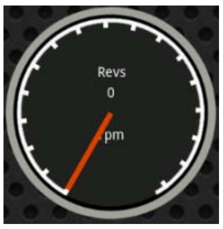

Fig. 2 Analog widget

This is a classic "clock" view similar to such that can be seen in most cars. In this view, there is a digital representation of what currently shows clock pointer.

Digital widget

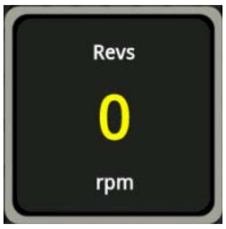

Fig. 3 Digital widget

This type of widget contains actual value of OBD II value parameter in digital form.

Graph widget

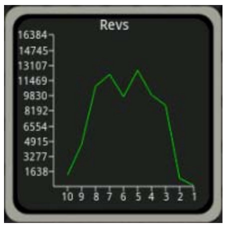

Fig. 4 Graph widget

This type of widget contains x and y-axis and the line which shows the last ten values in order they came. The x-axis therefore does not track time. Incoming values are stacked from right side of the display.

Log records

Solution offers recording obtained data into file. File is formatted to CSV format so it is easy to import to most of text editors, database editors or other programs for manipulating and storing data. User can choose from up to 40 OBD II parameters that can be recorded. Recording is not interrupted even while receiving a phone call, sms or other activity taking place on the mobile device. Recorded values can be accessed by simple file manager built in application. In the file manager user can see detail, rename, delete or see graphical representation of recorded data. Graphical representation offers to see at maximum of two different recorded parameters data in a same graph containing two different y-axes. Graphical representation can be controlled by simple gestures to zoom in, zoom out or move graph around. It can be practical to see the dependency of two recorded data streams in a single view.

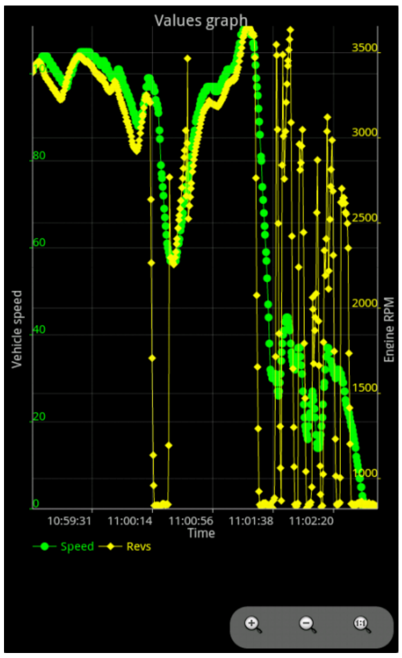

Fig. 5 Graphical activity showing two input data recorded streams at same graph using two y-axes

X-axis contains time data from both selected recorded parameters. Each of them contains time information of the data arrival.

5. CONCLUSION

The main contribution was creation of console interface, along with helper in writing commands and OBD II commands library with explanation. This step makes console interface also clear to the layman. Expert can benefit from the possibility of sending any command, even one that is not in the application helper library. Console interface can even be used to communicate with any serial device that includes a Bluetooth module.

The principal improvement over other solutions is not only improved production of records but view of the recorded files directly in the application. Overview of files is possible in a simple file manager that was also added to the application. The manager can view the recorded file in the representation of graphs, with a maximum of two y-axes.

Since automotive diagnostics is a broad issue, there are still many improvements that could be added to the application. The application could in the future work with maps and link the records to the current position of the car. Recorded data could be viewed as travelled route on the map.

ACKNOWLEDGEMENT

We support research activities in Slovakia / Project is co-financed from EU funds. This paper was developed within the Project "Centrum excelentnosti integrovaného výskumu a využitia progresívnych materiálov a technológií v oblasti automobilovej elektroniky", ITMS 26220120055.

REFERENCES

[1] SECONS s.r.o.: OBD-II protocols [online] [cit. 2012-04-21] Available online: <http://www.obdtester.com/obd2_protocols>

[2] ELM Electronics: ELM 327 OBD to RS232 Interpreter Datasheet, [online] [cit. 2012-04-21]. Available online: <http://www.elmelectronics.com/DSheets/ELM327DS.pdf>

[3] TOLÁK, J.: Úvod do diagnostiky – OBD/OBD II [online] [cit. 2012-04-21]. Available online: <http://www.autodiagnostika.jantolak.sk/?%E8l%E1nok-5-%FAvod-do-diagnostiky-obd-obd2-eobd-zariadenia,259>

[4] BALANI, R.: Energy Consumption Analysis for Bluetooth, WiFi and Cellular Networks, [online] [cit. 2012-04-21]. Available online: <http://nesl.ee.ucla.edu/fw/documents/reports/2007/PowerAnalysis.pdf>

[5] CORNISH, T.: Roewe 350 concept [online] [cit.2012-05-13]. Available online: <http://www.freewebs.com/roewe/roewe350.htm>

[6] CARTER, S.: World's first Android powered OEM car stereo [online] [cit 2012-05-13]. Available online: <http://www.androidauthority.com/android-powered-oem-car-stereo-68660/>

Received July 19, 2012, accepted December 21, 2012

BIOGRAPHIES

Martin Čabala was born in 1989 in Slovakia. In 2012 he graduated at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University (TUKE) in Košice.

Ján Gamec was born in Stuľany, Slovakia in 1960. He graduated from the Technical University in Košice with specialization in Radiotechnics, Summa cum laude in 1985. He reached a Ph.D. degree in radioelectronics at the Technical University, Košice, Slovakia, in 1995. Currently he is an associated professor of Electronics and Multimedia Communication Department of Faculty of Electrical engineering and Informatics of Technical University Košice. His main area of scientific research is digital image processing and UWB radar signal processing.