IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 5, Issue 2, February 2018

# Based Development of a PC and Smartphone Based Wireless Automobile Diagnostic System

**Selvaraj D[1], Arulkumar D[2], Dhinakaran D[3]**

Professor, Department of Electronics and Communication Engineering, Panimalar Engineering College, Poonamallee, Chennai, India[1]

Assistant Professor (G-I), Department of Electronics and Communication Engineering, Panimalar Institute of Technology, Poonamallee, Chennai, India[2]

Assistant Professor, Department of Computer Science and Engineering, Peri Institute of Technology, Manivakkam, Chennai, India[3]

**Abstract:** The design, implementation and testing of an automobile Engine Control Unit (ECU) data acquisition system is presented. The ECU parameters are graphically analyzed on a PC (Windows/Linux) or a smart phone. The graphical analysis software for real-time data monitoring are developed on Qt framework for PC use and android platform for smart phone use. The use of open source software Qt framework and android SDK for GUI development is a highlight of this system. The wireless communication features of the system using Zigbee technology allows remote real-time monitoring of the vehicle on test track.

**Keywords:** Android, Engine Control Unit, OBD protocol, Qt Framework, Zigbee.

## I. INTRODUCTION

The microprocessor based system responsible for the synchronization of various sub systems of an automobile. The ECU collects sensor values from different parts of the engine and performs the appropriate actions [1]. The ECU logs will allow easy identification of errors and subsequent repairs. The commercially available automobile diagnostic equipment suffer from numerous draw backs like non-support of hardware up gradation, most of them have sluggish operation which cannot match the needs of real-time diagnostics and inadequate network capability. Hu et.al [2] has designed a diagnostic system based on Freescale16-bit microcontroller with the graphical software developed on Microsoft Visual Basic 6.0. PC based diagnostic system with internet updates has been proposed by Jim et al. [3] which exemplify the prevailing trend in this domain.

In this paper, the development of low cost, fast, wireless and network capable automobile diagnostic system is proposed. The system is based on the industry standard Onboard Diagnostic -II (OBD-II) standard. The PC based analysis software is platform independent and can run on both Windows and Linux systems. The forth coming sections discuss the different aspects of the wireless automobile diagnostic system.

## II. ONBOARD DIAGNOSTIC STANDARD

OBD system was originally conceived as mechanism to monitor efficiency of engine and thus control vehicle exhaust emission. Different car manufacturers came up with their own proprietary Data Link Connecter (DLC), fault code and communication protocols for their cars. OBD-II provides a common platform for automobile fault detection. Different automobile manufacturers use signaling protocols to communicate with the ECU.

TABLE I. DEFINITION OF OBD – II STANDARD [1]

| | | | | |
|---|---|---|---|---|
| Physical Layer | ISO9141-2 | SAE J1850 | ISO14230-1 | ISO 11898<br>ISO 15765-4 |
| Data link Layer | ISO9141-2 | SAE J1850 | ISO14230-2 | ISO 11898<br>ISO 15765-4 |
| Network Layer | | | | ISO 15765-2<br>ISO 15765-4 |
| Transport Layer | | | | |
| Session Layer | | | | |
| Presentation Layer | | | | ISO 15765-4 |
| Application Layer | SAE J1979<br>ISO 15031-5 | SAE J1979<br>ISO 15031-5 | SAE J1979<br>ISO 15031-5 | SAE J1979/ ISO 15031-5 |

There are five widely used signaling protocols namely SAEJ1850 (PWM), SAEJ1850 (PVM), ISO14230 (KWP-2000), ISO9141 and ISO15765-4/SAEJ2480 (CAN). The OBD–II standard integrates these protocols at the application layer even though they are different at the physical and data link layer.
The OBD –II assigns a unique parameter ID (PID) to specific data request type. The user has to send the appropriate PID to the ECU using OBD –II to retrieve particular information and the ECU will respond with a sequence of bytes. The OBD-II standard groups the PID's into 9 modes each corresponding to different category of requests. For instance the mode 1 display current real time data such as the results of engine RPM sensor and mode 7 displays pending Diagnostic Trouble Codes (DTC). The diagnostic scan tools based on OBD-II send a message to the ECU to retrieve information. The footer contains a check sum field. The ECU responds to this request with a series of bytes. The response can either be bit encoded or simply value based bytes.

## III.SYSTEM DESIGN

The hardware and software design of the wireless automobile diagnostic system is presented in the upcoming paragraphs.

### *A. Hardware design and integration*

The automobile diagnostic system is built around the chip STN1110 [4]. The chip communicates with engine ECU through OBD-II protocol and returns requested vehicle parameters. The conversion of data from OBD II to serial format is also carried out by STN1110. A Zigbee module is directly connected to the serial bus of STN1110 to transfer the data wirelessly to the PC. The Bluetooth module provides connectivity to the PDA. At the receiver side, to interface the Zigbee module to the PC, a serial to USB converter built around FTDI23RL chip is used, as presently most laptops/PC's lack serial ports. The overall block diagram of the system hardware on the transmitter and receiver side is shown in the figures 1 and 2.

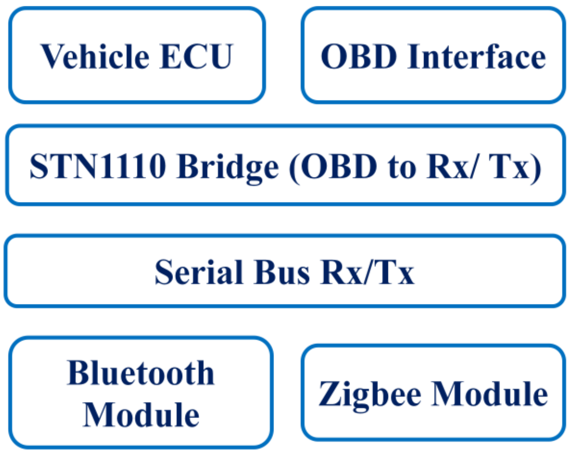

Fig.1 Block diagram of the system hardware – Transmitter side (vehicle)

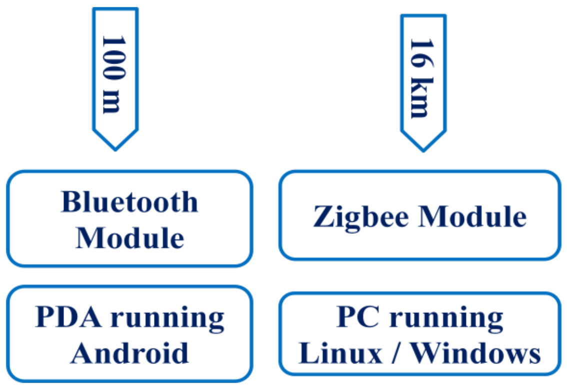

Fig.2 Block diagram of the system hardware – Receiver side

OBD-II compliant vehicles have a standardized diagnostic link connector (DLC) allowing a scan tools to be used on all vehicles. The DLC is placed under the dashboard. The vehicle-side hardware STN1110 connects to the DLC by means of an OBD -II to DB-9 cable[5]. To debug the connection errors, the board is provided with four indicator LED's which

indicate OBDRx, OBDTx, Rx and Tx. The Blue LINK Bluetooth Module [6] has built-in Voltage regulator and 3V3 to 5V level converter that can be easily interfaced with STN1110 OBD to UART board.
The Zigbee module used for the vehicle end and the PC end used was XBee OEM RF Module [8]. The module is compatible with the IEEE 802.15.4 (Zigbee). The Rx-Tx voltage levels of XBee module is 3V3. The Zigbee module can be configured Digi's X-CTU software utility [7]. The Transmitting end hardware assembly is shown in figure 3.

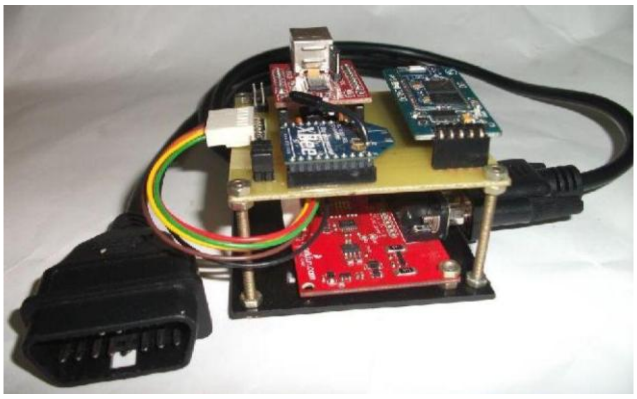

Fig.3 The Transmitting end hardware assembly: The OBD to UART board integrated with the zigbee module, Bluetooth module and USB to RS232 breakout module.

At the PC receiving side the XBee zigbee module receives data from the vehicle-side XBee zigbee module. The XBee module is connected to PC using XBee-Explorer board [8].

## B. Software design

The analysis software for the automobile diagnostic system has been developed for the PC/Laptop based operation and PDA/Smartphone based operation. The PC based analysis software was intended to be platform independent; the software development was hence carried out using Qt framework [9]. Android application has been developed to allow the analysis of ECU data in real-time on smart phones.[9].
The architecture of the PC based application software is shown in figure 4. The kernel accomplishes various tasks such as listening to OBD and collecting data from OBD. The GUI will look after the logic in how the user can use these functions. The GUI layer handles the display of acquired data in attractive way and GUI is rendered to screen. To display the graphs and dials QWT library (Qt widgets for technical applications) is used. For serial communication between software and STN1110 hardware QExtSerialPort library is used [10 -12].

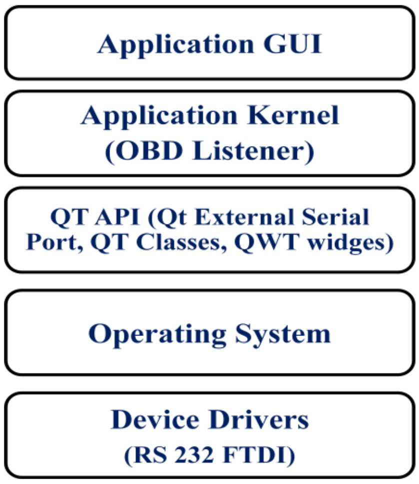
Application GUI
Application Kernel (OBD Listener)
QT API (Qt External Serial Port, QT Classes, QWT widges)
Operating System
Device Drivers (RS 232 FTDI)

Fig.4 Architecture of application software based on Qt

The real time monitoring of data from STN1110 is done using threading functionality of Qt. The serial communication requires to be run in its own thread; otherwise the GUI will hang by waiting the communication to finish first. Qt provides a useful class for threading, the QThread class. In order to implement threading the class must inherit the QThread class, and add run () method to derived class. Whenever the run () method is called, the thread will start running. Once this run () method is called, it simply create an instance of the derived class and executes start () method.

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
ISO 3297:2007 Certified
Vol. 5, Issue 2, February 2018

A SerialPortReader class is created to implement the real- time monitoring of sensor data from STN1110. SerialPortReader subclasses Qthread and implements the run () method for the continuous monitoring of sensors. Once off calls to the ECU do not require to run in their own thread, so these are just handled using a normal method. Every time the application is executed it begins monitoring sensors, by simply starting the SerialPortReader thread. Within the run of this, it loops around different sensor PIDs and communicates with the ECU in order to update each of these sensors values continuously. The sensors then emit signals as their values change so that slots get the updated value. QOBJECT macro along with Meta Object Compiler provides the signals and slot functionality in Qt. The signal is emitted during each reception of sensor value, and the run () method emits the signal. QWT library was used to implement the widgets in GUI. The dials, graphs are created by inheriting the QWT classes QWTPlot, QWTDial, etc [10]. All the widgets are created as separate objects, and they are arranged using QT class QHBoxLayout, QVBoxLayout. The different widgets are tabbed and displayed using QTabWidget class of Qt in the main event loop. The plots and dials are updated whenever the sensor value is changed. The slots provided on each plot and dials is connected to the corresponding signals using connect method. Each individual widget is separate object but thread is called and created once in the main event loop. The GUI contains two different tabs, Graphical view, and Cluster view[13].
The Android application running on the smart phone initially loads up the GUI. It then waits to establish a Bluetooth connection with the vehicle end hardware discussed earlier. Using the inflator menu the user can connect to any Bluetooth device that is already paired. After establishing the connection, the application can now acquire data from vehicle through Bluetooth data exchange. The values obtained from the STN1110 are displayed on graph using the a chart engine library. The GUI is coded in XML, the java based UI declarative language.

## IV. SYSTEM TESTING AND RESULTS

This section presents the testing of the hardware units of system followed by testing of the completely integrated system. The hyper terminal and the OBD-II-UART will return the battery voltage. The OBD-II-UART has an auto-detect feature that can determine which protocol should be used, this feature can be invoked by issuing the command 'ATSP0'. Once, the OBD-II-UART has detected the proper protocol, an 'OK' response seen in the terminal window.
Once the OBD-II-UART has been configured for the proper OBD protocol we can start sending OBD commands to the board. The OBD commands consist of hexadecimal codes written in ASCII. In the serial terminal, issue the command '0100' to determine the PIDs supported in mode 01.

### *A. Vehicle end hardware connectivity testing*

This section describes the hardware vehicles end connectivity testing procedure with the ECU. Connect the DB9 OBD cable to the vehicle DLC and connect the bridge to DB9 STN1110. FTDI USB to connect the serial converter to the output of the 1110 STN bridge then connect the FTDI card to the USB port of the laptop. Open a serial terminal on the laptop. Configure the settings for the terminal to use the serial port to which the user is connecting the USB cable, the transmission settings must be configured for 9600 bps, 8 data bits, 1 stop bit, no parity. Make sure the OBDII-UART is connected to the vehicle with the OBDII cable. Then, to restart the OBD-II-UART card, enter the characters of the ATZ 'in the terminal and press Enter. This command will cause the OBD-II-UART board to restart. On the plate, LEDs flash and the start-up message is received in the terminal window. If the missing characters are visible, the serial port is not correctly configured. Check all settings and try again. Once the communication is established with the card, send the command to read the battery voltage "ATRV"

### *B. Full system Testing*

The track testing of the fully integrated system was carried out on a Hyundai motors Santro car. The hardware is connected to the DLC through OBD to DB9 cable. On starting the cat a beep sound was heard and the OBD indicator was seen on the dash board. When theSTN1110 board is initialized the OBDRx, OBTx, Rx, Tx LEDs blink for a while. Then the car was driven on an actual track .The complete hardware set

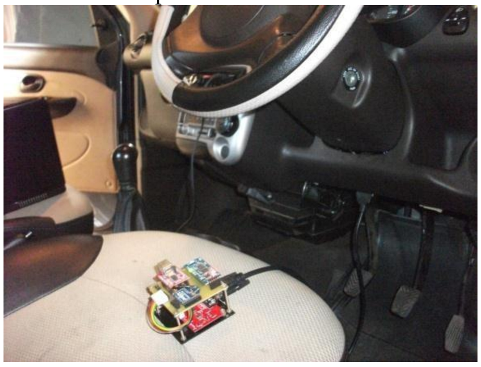
Fig.5. The complete hardware set up in the vehicle end for testing

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

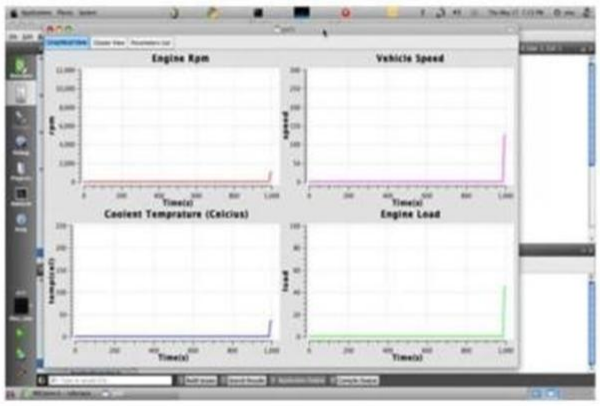

Fig.6. The PC end testing set up and screenshot of the GUI displaying data

The track testing of the android application for display the ECU data was carried out using, the android mobile phone (HTC Wildfire S, Android 2.3 Gingerbread). The mobile phone was placed within the Bluetooth range and Bluetooth connection was established between the vehicle end device and the mobile phone. The vehicle was driven on track and various parameters were observed graphically on the android app. The screenshot of the android phone app testing is shown in figure 7.

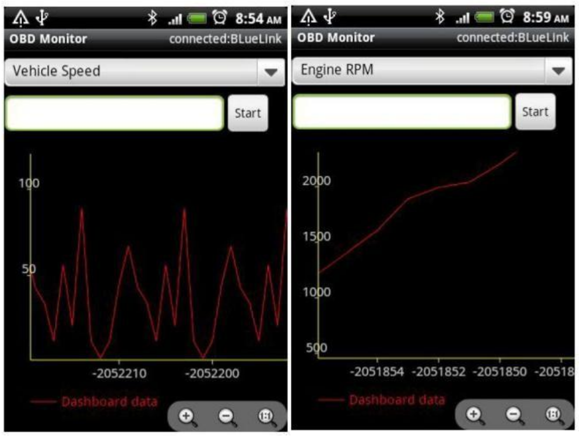

Fig.7. The Android app running on HTC Wildfire S phone (Android 2.3 Gingerbread) displaying real-time ECU data (Vehicle Speed, Engine RPM)

up in vehicle end during testing is shown in figure 5.The real time data acquired from the ECU by the hardware was sent to the laptop via the zigbee wireless ling and displayed on the GUI (Figure 6). The PC/laptop based analysis software was tested on Windows and Linux platform and data from ECU was displayed on the GUI

## V. CONCLUSION

A wireless automobile diagnostic system was developed using commercial off-the-shelf components, the system is based on the industry standard OBD-II protocol. The system has wireless operation using Bluetooth and Zigbee technology, thus enabling real-time track testing of the vehicle. The PC based analysis software is platform independent and can run on both Windows and Linux systems. The Android application allows analysis of ECU data in real-time on smart phones. The test results indicate that diagnostic system can provide the actual working status of engine or vehicle in real time thus forming a valuable tool for vehicle maintenance and development of new vehicle models.

## REFERENCES

[1] California Air Resources Board ―Technical status update and proposed revisions to malfunction and diagnostics system requirements application to 1994 and subsequent California passenger cars, light-duty trucks, and medium-duty vehicles-(OBDII)‖. CARB Staff Report, 1991.
[2] Hu Jie , ―Developing PC-Based Automobile Diagnostic System Based on OBD System‖, Power and Energy Engineering Conference (APPEEC ), 2010 Asia-Pacific, pp 1-5.
[3] Jim Luyckx, Mohan Sethi, Mike Clancy. ― Completely integrateddiagnostic platform using industry standard PCs and internet update‖. SAE 2004-01-0674H.
[4] STN1110 multiprotocol OBD to UART interpreter IC datasheet, http://www.scantool.net/scantool/downloads / 97/ stn1110-ds.pdf,accessed on April 4, 2013.
[5] ELM 327 - OBD to RS232 Interpreter data sheet, http://www.elmelectronics.com/dsheets.html accessed on April 4, 2013.
[6] BlueLINK- Bluetooth Module datasheet, www.rhydolabz.com/documents /.../BlueLINK_User_Manual.pdf, accessed on April 4, 2013.

[7] FT232R is a USB to serial UART interface IC datasheet, http://www.ftdichip.com/Products/ICs/FT232R.htm, accessed on April 4, 2013.
[8] XBee™/XBee-PRO™ OEM RF Modules datasheet, ftp://ftp1.digi.com/support/documentation/ 90000982_A.pdf, accessed on April 4, 2016.
[9] Jasmin Blanchette, Mark Summerfield, Qt Programming, Prentice Hall,2006.
[10] QTCross platform API and UI framework, http://qt.digia.com/Product/Learning/Education/elearning/onlinetraining/ accessed on April 4, 2016.
[11] QExtSerialPort Cross-platform Library for serial communication, http://code.google.com/p/qextserialport/, accessed on April 4, 2016
[12] QWT (Qt library for technical applications), http://qwt.sourceforge.net/, accessed on April 4, 2013.
[13] Android Developer, http://developer.android.com/, accessed on April 4,2013.
[14] Wang Quanqi, ―Design of Vehicle Bus Data Acquisition and Fault Diagnosis System‖, Conference of Consumer Electronics, Communications and Networks (CECNet), 2011, pp 245-248.
[15] Global OBD Vehicle Communication Software Manual, Snap-on Incorporated GlobalOBD Vehicle Communication Software Manual_EAZ0025B43B.pdf accessed on April 4, 2016.