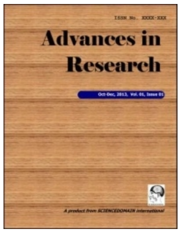


Advances in Research
2(1): 01-11, 2014, Article no. AIR.2014.001

SCIENCEDOMAIN *international*
www.sciencedomain.org

Implementation of Ethernet in the OBD Port of Automobiles

Gulshan Mandhyan[1], Vijay Charan[1], G. J. Vinod Kumar[1] and Gerardine Immaculate Mary[1*]

[1]School of Electronics Engineering, Vellore Institute of Technology, Vellore, 632014, India.

Authors' contributions

This work was carried out in collaboration between all authors. Author GM designed the study, performed the statistical analysis, wrote the protocol, and wrote the first draft of the manuscript. Authors VC and GJVK managed the analyses of the study. Author GIM managed the literature searches. All authors read and approved the final manuscript.

Original Research Article

Received 8th August 2013
Accepted 27th October 2013
Published 16th November 2013

ABSTRACT

Aims: To build an appended Ethernet port in automobile and display the diagnostic data on the web browser.
Study Design: Ethernet port enables the automobiles for real time performance testing, diagnosing, data logging, vehicle monitoring, and remote diagnosing over IP.
Place and Duration of Study: Embedded Systems Lab, Vellore Institute of Technology, Vellore, between June 2012 and July 2013.
Methodology: The numerous diagnostic data available from On Board Diagnostic (OBD) port of modern automobiles is extrapolated to promising network technologies like Ethernet.
Results: Acquired engine parameters from the OBD port of an automobile and displayed them on to an LCD, these real time engine parameter readings were transferred to the internet and accessed from the web browser.
Conclusion: The diagnostics of a car normally requires a person to take his car to an authorized service center where the mechanic uses a Scan Tool to diagnose the car which is usually time consuming. A more profound way to diagnose a car is using your laptop which is connected via Ethernet to your car. This allows you to diagnose your car at your own convenience, so that whenever the Malfunction Indicator Light of your car goes ON one can know and fix the fault as it occurs.

**Corresponding author: E-mail: gerardine@vit.ac.in;*

Keywords: Implementation of ethernet; remote diagnosing over IP; OBD port; automotive application.

1. INTRODUCTION

For the ‘24/7/365’ world we live in, the user now demands the same services in the car as the office or home; emails, music, gaming and video, along with increased safety. A single IP-based network is the key, enabling the car to enter the World Wide Web. Introducing Ethernet port enables the automobiles for real time performance testing, diagnosing, data logging, vehicle monitoring, and remote diagnosing over IP. This project is targeted to build an appended Ethernet port in automobile and display the diagnostic data on the web browser.

In automotive electronics, electronic control unit (ECU) is a generic term for any embedded system that controls one or more of the electrical system or subsystems in a motor vehicle. A current vehicle consists of up to 70 electronic control units (ECUs) with different functionalities interconnected by specific automotive bus systems [1]. Modern engines are controlled electronically using real time software in a device known as the engine control unit (ECU) [2] - not to be confused with electronic control unit, the generic term for all these devices. The engine control unit controls many other sub systems of the engine such as, for example, the antilocking braking system (ABS). All decisions made by the engine control unit are based on the state of sensors that are placed at various places throughout the vehicle primarily around the engine bay. As years went on, the engine control unit became more capable of supplying diagnostic and sensor data to help mechanics identify the source of problems that arise in the engine management system. Eventually a standard was created that all manufacturers were encouraged to follow. The standard became commonly known as On Board Diagnostics (OBD) [3]. OBD-II is an enhancement of the OBD standard that was introduced later and made mandatory [4]. Types of Electronic control unit (ECU) include many units/modules and one such is engine control unit (ECU). Despite this ambiguity in abbreviation, we will refer ECU to Engine Control Unit henceforth. And the reason is OBD-II standards gives access to information only from Engine Control Unit.

The diagnostics of a car normally requires a person to take his car to an authorized service center where the mechanic uses a Scan Tool to diagnose the car which is usually time consuming. A more profound way to diagnose a car is using your laptop which is connected via Ethernet to your car. This allows you to diagnose your car at your own convenience, so that whenever the Malfunction Indicator Light of your car goes ON one can know and fix the fault as it occurs.

2. METHODOLOGY

The aim of this project is to build an appended Ethernet port that facilitates the diagnosis of a car over IP. It should be capable of extracting the necessary data from the vehicle's engine control unit (ECU) in order to use it in a meaningful and useful way. The software that will run on the device will have to be able to work with the hexadecimal replies that the ECU returns on requests of data. The communication with the ECU will have to be handled using a polling type method as data interrupts or automatic updating of data from the ECU cannot be done sporadically. Instead a cyclic process or thread will have to run continuously to do a query to the ECU followed by the reading of the reply. The software will have to work with the returned hexadecimal data in a way that provides the user or driver useful functionality.

The following is the methodology that is used in this work and Fig. 1 shows the block diagram.

1. To communicate with the ECU of a vehicle indirectly with the help of an integrated circuit, the ELM327 [5] which will handle all the low level bus communication with the ECU using whatever signalling method the vehicle uses. There are five OBD-II signalling protocols in total and the ELM327 supports all five including CAN.
2. To retrieve the sensor data and the diagnostic trouble codes from the ECU by sending commands to the ECU and retrieving and interpreting the responses via ELM327 IC.
3. Configuring the Arduino Ethernet Shield to host a web page where all the diagnostic data is visible.

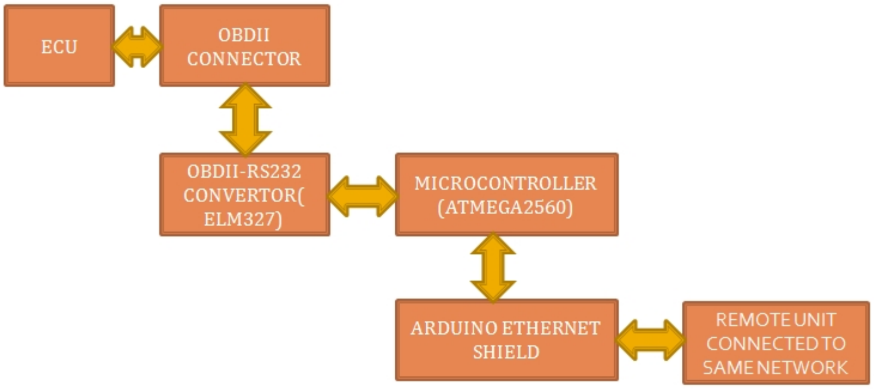

ECU
OBDII CONNECTOR
OBDII-RS232 CONVERTOR(ELM327)
MICROCONTROLLER (ATMEGA2560)
ARDUINO ETHERNET SHIELD
REMOTE UNIT CONNECTED TO SAME NETWORK

Fig. 1. Block Diagram of the project

In order to connect to the OBD-II port and interpret the signals, one needs to convert these signals to RS232 format. The IC widely used for this purpose is the ELM327. It is an OBD-II to RS232 interpreter. This IC is available individually as well as packaged in form of an adapter (OBD-II adapter). The adapter one is preferred as it avoids power or interfacing problems.

ELM327 is just an interpreter, so a microcontroller is required to send, retrieve and interpret commands and responses from the ECU. The Arduino Mega which has an ATMEGA2560 microcontroller is quite suitable for this purpose.

On retrieving the sensor data and the diagnostic trouble codes (DTC) from the ECU, the next task is to view these on a web page via an Ethernet network. This is possible with Arduino compatible Ethernet Shields. This shield also has a micro SD card slot which is used to store a database for all DTC’s and their meaning. For creating a web page, HTML is used.

3. BACKGROUND THEORY

OBD is a technology that is embedded within an engine control unit (ECU). The ECU is the heart of a vehicle’s engine management system. It is the computer that controls everything from when the brakes of a vehicle are briefly disabled to prevent locking to the exact timing of when a spark occurs in the engine.

All modern vehicles manufactured in United States must implement the OBD-II [6] technology in their vehicles by law. And many other countries have started establishing such mandatory laws. The original OBD standard was developed in an effort to encourage manufacturers to produce highly efficient vehicles that produced minimum emissions while maintaining optimum fuel economy. However the newer version, OBD-II was made mandatory on all vehicles.

The OBD technology benefits motorists, technicians and mechanics by providing them with useful information, such as the state of certain parts of the engine management system. This allows them to quickly identify the sources of problems and guide them on the correct path to repairing. Several different methods of diagnosing are available [7]. If the ECU discovers a fault in some system, it logs a diagnostic trouble code. If the mechanic wants to monitor sensors in real time it can be done by looking up the relevant sensor value using scan tools [8].

A parameter ID (PID) [9] is a unique code or command that OBD assigns to a specific data request type. So in order to communicate with an ECU using OBD-II, you must first send the appropriate PID for the type of information you want and the ECU will then respond with a sequence of bytes. The bytes are usually expressed in hexadecimal format.

The OBD-II standard does not require vehicle manufacturers to implement all PIDs. In fact, it doesn't even give a minimum for some modes such as Mode 1 and Mode 2 PIDs. However, most manufacturers implement the most common ones such as vehicle speed and engine RPM.

Since there are different categories of requests, the OBD-II standard breaks the PIDs up into groups, known as modes. In the original J1979 specification document of the SAE, it listed 9 diagnostic test modes [9]. They are as follows:

- 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 snapshot 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).

Since the original specification, other modes have been added on and a lot are manufacturer specific.

To send a request to the ECU you must specify the mode and the PID. So for example, if we want to view the current engine RPM, we would send a 010Ch (hexadecimal) query to the ECU. The ECU would then respond with a few bytes of data for the response. If we wanted to see the engine RPM stored value when a fault occurred last on the vehicle, we 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.

There are four main types of Diagnostic Trouble Codes (DTC) defined by the SAE standards, Fig. 2. These are the following:

	First digit will be:
Powertrain Codes (P codes)	0 - 3
Chassis Codes (C codes)	4 - 7
Body Codes (B codes)	8 - B
Network Codes (U codes)	C - F

These codes identify where or what system the fault occurred. The powertrain codes are the most common and represent codes that occur in the engine management system. Diagnostic trouble codes are made up of 5 digits. The digits are in hexadecimal format. The first digit always identifies the type code whether it a powertrain code, body code etc. In the above list, you can see the range of digits that identify what category of codes it belongs to. The other 4 digits in the code identify other information. For example the second digit identifies if it is a standard SAE defined code or a proprietary while the third digit identifies what system caused the fault. Below is a diagram that illustrates the format of a code.

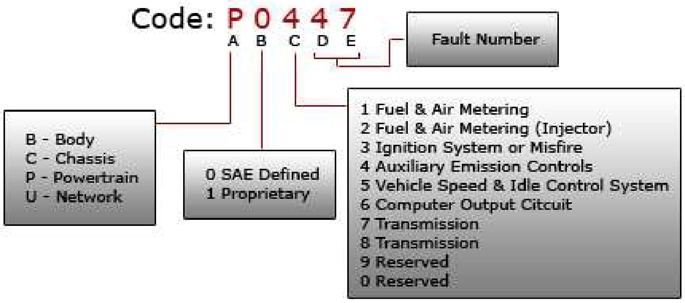


Fig. 2. Format of diagnostics trouble code

4. RESULTS AND DISCUSSION

The first stage was to acquire engine parameters and display them to an LCD shield. This initial testing helped in understanding the conversion of OBDII format data from ECU of a car into RS232 signal. Since the serial format is a universal format of data for transferring, reading, operating by almost all microcontrollers, this data was displayed to a 48x84 size LCD display. Fig. 4 shows snapshots of eight engine parameters and their real time values that is measured on Toyota ETIOS car. Fig. 3 is the port of Toyota ETIOS that is used for testing from the Service Centre. The ELM327 Adapter is to be connected to this port.

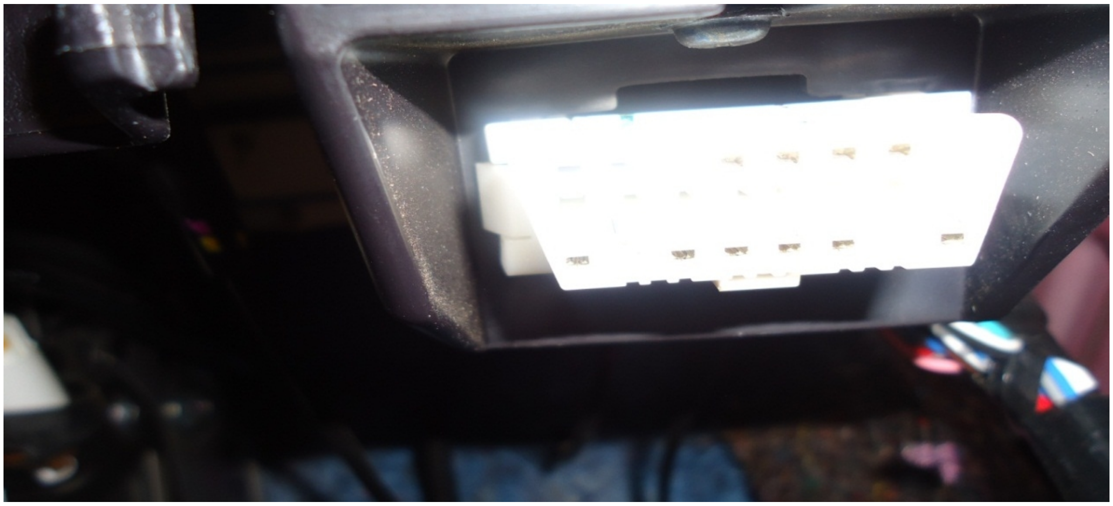


Fig. 3. OBD port of Toyota ETIOS car

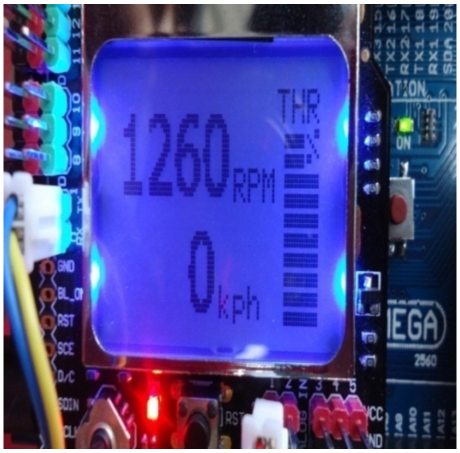


Fig. 4a. The display as the car engine is on

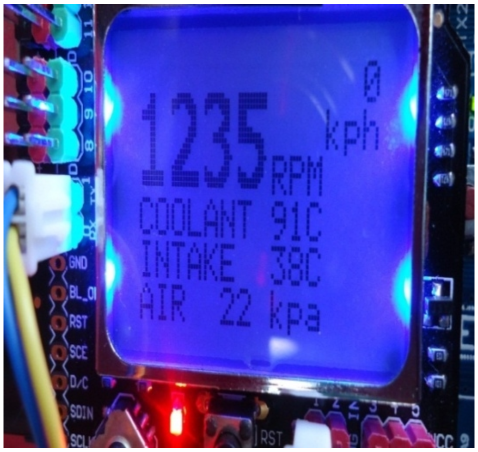


Fig. 4b. The next set of parameters

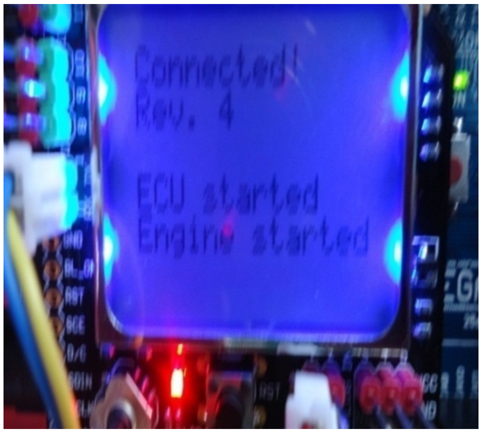


Fig. 4c. The initial display

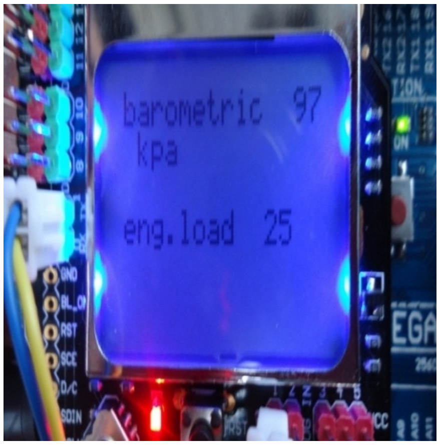


Fig. 4d. Other parameters

Fig. 4c is the initial display of the device as the adapter is connected to the OBD port. Primarily, here the adapter checks whether the device is connected properly and if everything is fine, displays the version of the adapter. Then it waits for the ignition to ON and if it is ON, its then displays “Engine Started”.

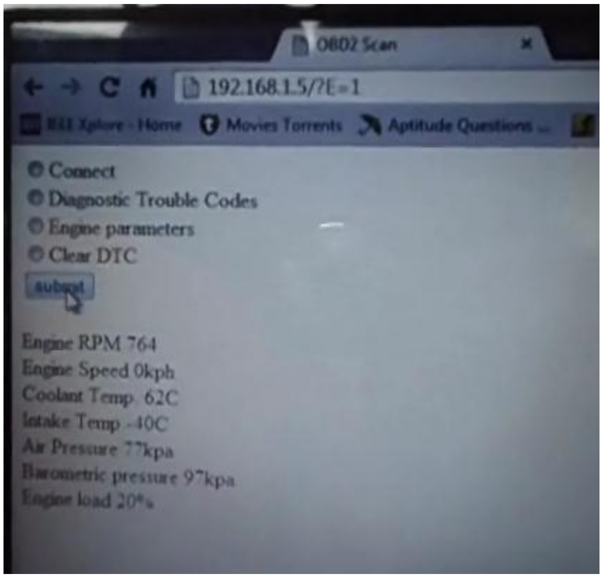
Fig. 4a shows the display as the car engine is on. The parameters displayed here are Engine RPM, Engine speed (it is ‘0’ as the car is not mobile) and throttle position. For attaining the other parameters, joystick button provided on the LCD shield is operated appropriately.

Fig. 4b shows the next set of parameters along with Engine RPM and Speed. Coolant and inlet temperatures and air pressure are other parameter showed on this screen.

Fig. 4d shows the biometric pressure and engine load.

The second stage of the project was to transfer these real time engine parameter readings to the internet and hence can be accessed from the web browser. Apart from the engine parameters, it shows the DTC of the automobiles, if any. The DTCs can be cleared from the device. This stage uses the serial data previously obtained and using the formatting devices (Ethernet shield), the data is converted to TCP/IP format. The sample web page created and hosted on the Arduino board, has functions for showing the engine parameters, showing DTC and for clearing the DTC.

Showing engine parameters is done through the Mode 1 request and since these set of data can be accessed without complication or threat to the automobile, it can be readily availed with proper PID queries to the ECU. The snapshot in Fig. 5 shows the engine parameter when selecting the *Engine parameters* button. The various real time parameters are Engine RPM, Speed, Coolant and Intake temperature, Air pressure, Barometric pressure and engine load. It can be verified that the same parameters were obtained on LCD display during the first stage of testing.



Engine RPM 764
Engine Speed 0kph
Coolant Temp. 62C
Intake Temp -40C
Air Pressure 77kpa
Barometric pressure 97kpa
Engine load 20%

Fig. 5. Engine Parameters shown on web page

In this part of the work the diagnostic trouble codes of the car was displayed on the webpage. This information of the car are obtained using Mode 3 request. DTCs are shown only if there are any troubles associated with engine and each fault in the hardware is encoded with unique code. On request, the mechanics at the Service Centre created a malfunction deliberately with the automobile. The malfunction was created with *Air Flow Circuit* unit. The mechanics later on showed the DTC on their professional Scan Tool, *Intelligent Tester* II. To authenticate the new device it was then connected to the OBD port and it showed the same DTC associated with the fault created. Fig. 6 shows the Malfunction Indicator Light On as fault is deliberately generated.

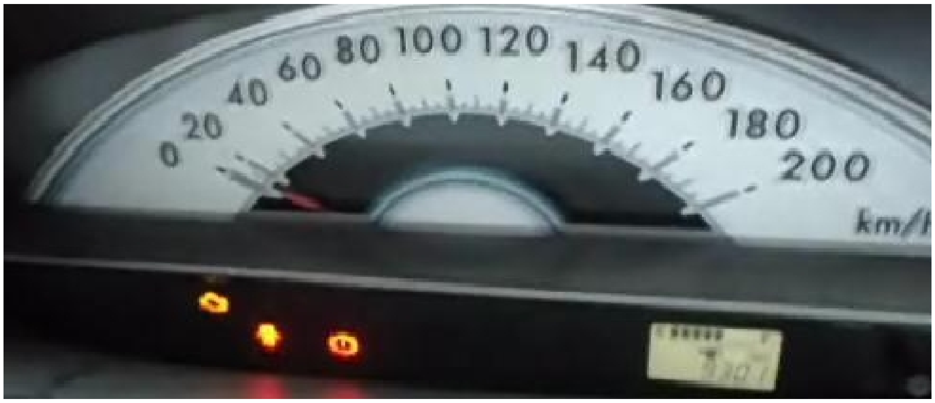
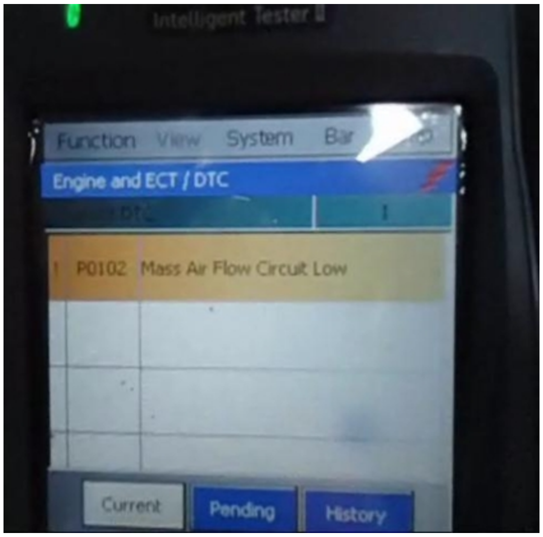


Fig. 6. MIL is ON as fault is created.

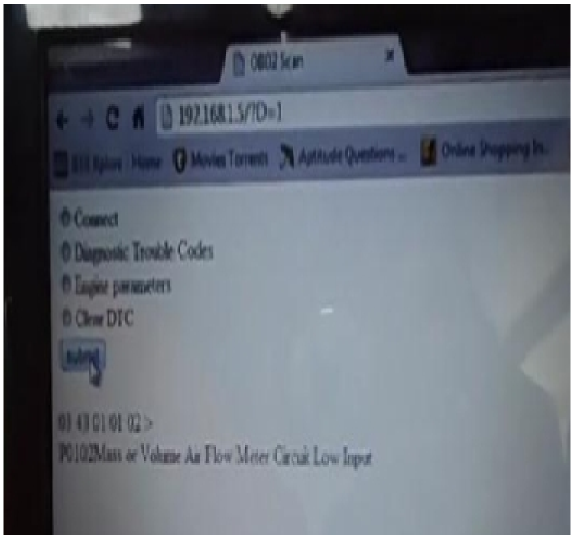
The snapshot of DTCs shown on the commercial Scan Tool of Service Centre and the one shown on web browser through the new device is shown in Figs. 7 and 8.



Engine and ECT / DTC

1 P0102 Mass Air Flow Circuit Low

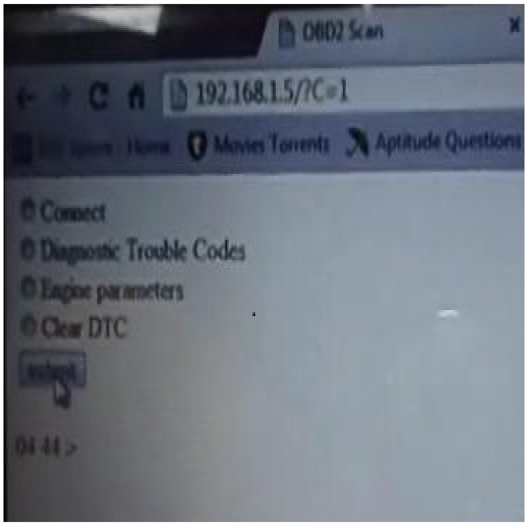
Fig. 7a. DTC shown on Scan Tool



```
03 43 01 01 02 >
P0102Mass or Volume Air Flow Meter Circuit Low Input
```

Fig. 7b. DTC shown on web browser

Clearing DTC means software deletion of trouble codes and it is done with a Mode 4 request. Unless the real hardware malfunction is rectified the DTC would reappear even after it is cleared. Dealing with DTCs, both viewing and clearing them are sensitive operations. To authenticate that the new device was able to successfully clear the DTC, it was verified again with the MIL On/Off light and with the commercial scan tool. As one clears the DTCs, the ECU would simply respond with string of '44' and it means that the troubles are successfully cleared. Later, the mechanics were requested to check for any DTCs present by the commercial Scan Tool. As no DTC was found in its list, it is concluded that the new device cleared the fault codes. It can also be verified with MIL Off on the car deck, Fig. 9.



04 44 >

Fig. 8a. Clear DTC via browser

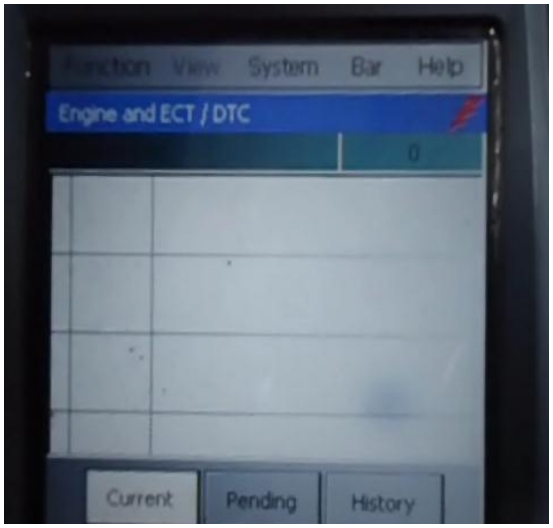


Fig. 8b. Checking for DTC on Scan tool

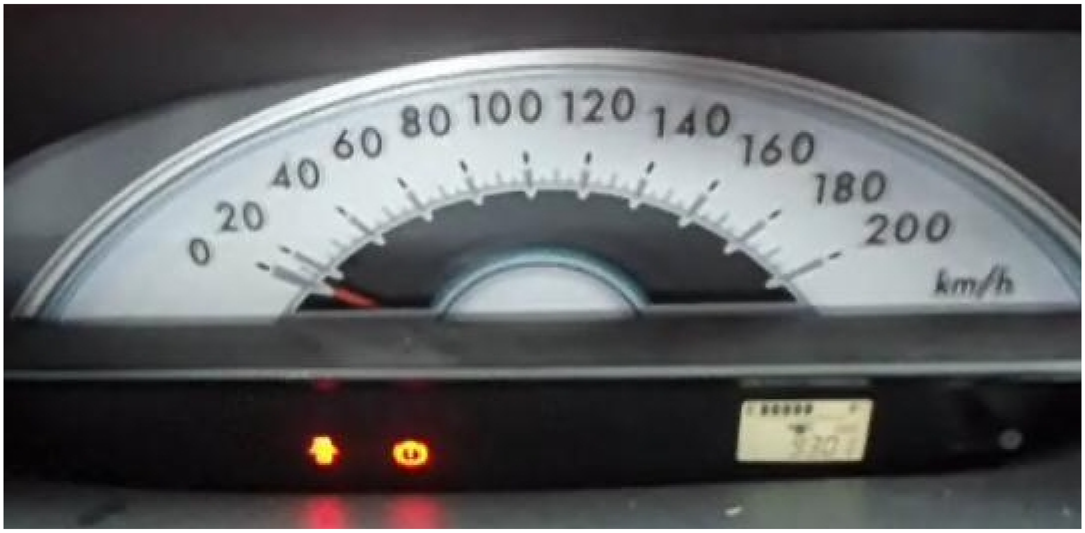


Fig. 9. MIL is off as fault code is cleared.

Thus the engine parameters were showed on the web browser just as shown on a LCD display. Further through a series of snapshots the process of acquiring DTCs in Mode 3 request and also clearing the same using Mode 4 request were shown. To validate and authenticate the proper working of the new device, the results obtained from a commercial Scan Tool is shown alongside in Figs. 7 and 8.

4. CONCLUSION

This work has successfully demonstrated the building of an automotive embedded device from the scratch. The implementation of Ethernet port in automobiles required the very basic understanding of any simple automotive system, the evolution of improvements, the technical problems associated with new models, solutions proposed to such problems, the attributes of different types of automobiles, etc. Besides this work was a solution for high speed diagnoses of automobiles over internet by a common man without much support from Service stations. Set for a low-budget the new device, received positive feedbacks from the experts of the Service station as it almost completely mimicked the highly priced commercial Scan Tool.

Although the concept of diagnosing over internet was proved through this work, there is always room for improvement. When a fault is detected in the engine management system by the ECU, diagnostic trouble codes (DTCs) are logged and a copy of the state of all sensors is taken and stored in the form of a freeze frame. Freeze frame data is essentially just a snap shot of all available sensors. This recorded state of the engine management system when a fault occurred can be very useful to mechanics on identifying the root cause of the fault. Implementing such functionality as this is very feasible. It is only a matter of doing a query of all sensors using Mode 2 requests rather than Mode 1. This function can be included with some slight modifications in the web page. As several sensor data will be produced onto the screen, sufficient amount of memory space management would be required.

In the context of this work, data logging could be a background process that runs during sensor monitoring sessions. That is, during the Mode 1 request, as the device reads the real-time sensor values, it can be logged on to a file and this history of data can retrieve for further analysis. Any mathematical operation can be done on this stored data and can be presented in different formats. It can be easily said that data logging would serve as a black box of information for future use.

ACKNOWLEDGEMENTS

This work is funded by Department of Science and Technology (DST), Government of India, and supported by Vellore Institute of Technology, together with Lanson Toyota Service Center at Vellore.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Hyung-Taek Lim D. Herrscher L. Volker MJ. Walt, “IEEE 802.1AS Time Synchronization in a switched Ethernet based In-Car Network”, 2011 IEEE Vehicular Networking Conference (VNC). November 14-16, 2011, Amsterdam.
2. Nice, Karim. “How Car Computers Work. 2001. HowStuffWorks.com http://auto.howstuffworks.com/car-computer.htm.
3. Wikipedia. Onboard Diagnostics; 2009. http://en.wikipedia.org/wiki/On-Board_Diagnostics
4. B&B Electronics. OBD-II Background Information. 2008. http://www.obdii.com/background.html
5. ELM Electronics (Unknown), LM327 Datasheet; 2012. www.elmelectronics.com/DSheets/ELM327DS.pdf
6. Wikipedia Clean Air Act. 1970;2009. http://en.wikipedia.org/wiki/Clean_Air_Act_(1970).
7. Daoud RM, Amer HH, Elsayed HM, Sallez Y. Ethernet-Based Car Control Network, IEEE CCECE/CCGEI, Ottawa, May 2006.
8. Navet N, Song Y, Simonot-Lion F, Wilwert C. Trends in Automotive Communication Systems. Proceedings of the IEEE. 2005;93(6).
9. Wikipedia. OBD-II PID.; 2013. http://en.wikipedia.org/wiki/OBD-II_PIDs

---

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sciencedomain.org/review-history.php?iid=322&id=31&aid=2523