System for Remote Diagnostic of Vehicle Defects

Lejla Ferhatović, Almir Lipjankić, Asja Handžić i Novica Nosović

***Abstract* — This paper presents implementation of simple system for remote diagnostic of vehicle defects. Implementation process includes diagnostic trouble codes defined in existing standards. Presented application is client/server application and the paper presents functionality and algorithms. The communication link between the client and server is established using mobile phone.**

***Keywords* — Data Link Connection (DLC), Diagnostic Trouble Codes (DTC), diagnostic, OBDII, remote diagnostic, vehicle**

I. Introduction

Today, the concept of vehicle navigation and location is used often and science invests a lot in development of this idea. However, it is seldom spoken about the problem of diagnoses of defects and errors on a remote vehicle

A diagnostic system is a system for detection of defects on a vehicle, correct diagnoses of a defect and sending instructions and commands to the driver on how to proceed in that situation. If the system has data about the location and type of error, the system can send the vehicle to the nearest car service.

The goal of this work is the execution of diagnoses on a remote vehicle using internationally agreed data trouble codes. We exposed the implementation of a simple diagnostic system which is available for everybody because the system offers understandable information for drivers and for specialists.

II. Communication Standards for Diagnostic

Electronic Control Unit - ECU collects data from sensors and controls work of different systems in a vehicle. For a successful communication with the ECU a certain communication standard, which can be used for effective sending of commands to the ECU and safe reception of data, is necessary. The well-known diagnostic standards for communication with the ECU in a vehicle are OBDII and EOBD. The OBDII standard was published in 1996 by the California Air Resource Board (CARB) and the United States Environmental Protection Agency (US EPA). The standard requires that all vehicle manufacturers use similar embedded diagnostic standards (On-Board Diagnostic, OBD). This includes standardized connectors, standardized data trouble codes, terminology, system monitoring etc. In most situations manufacturers extend this standard and add own specific parts. EOBD is an acronym for European On-Board Diagnostics.

By connecting a specific electronic device - Data Link Connector (DLC) on the OBD connector, the connection is established and receiving data from electronic control unit in a vehicle becomes possible. It is possible to receive information about defects, additional information about the vehicle such as voltage for the injection system, state of the ABS system, etc. by using DLC and Diagnostic Trouble Codes (DTC).

Diagnostic Trouble Codes are registered only when a defect in the vehicle exists and they stay saved in the permanent memory of an electronic control unit.

Every diagnostic trouble code includes one letter and four numbers. Examples of correct DTC codes are: P0171, P0201, and P0272...

Communication between a computer in a vehicle and an external device for diagnostic is conducted through messages. EOBD/OBDII protocols use a same format of messages, defined with standard (Fig. 1).

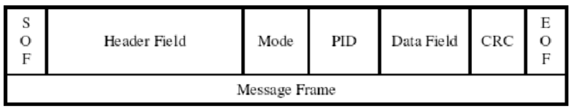


Fig. 1. SAE J1850 message

An external device sends a data request to the computer and the computer from the vehicle answers this request.

III. Structure of OBDII-GSM Hardware Module

The OBDII-GSM module is an electronic microprocessor device. It connects the OBDII port on the vehicle, the GSM modem and the computer with a purpose of sending information about the vehicle to the user of the device using a GSM network. The structure of the module provides that the OBDII-GSM system, when it receives an SMS message with a command for a vehicle sent from the server, receives data, does software processing in the microcontroller and sends a command to the OBDII port on the vehicle. The response from the vehicle is processed in the microcontroller and the microcontroller sends the answer to the server via a GSM modem.

L. Ferhatović, Faculty of Electrical Engineering Sarajevo, Zmaja od Bosne bb, University Campus, Sarajevo, Bosnia and Herzegovina; (phone: +387 61 826 000; e-mail: lejla.ferhatovic@gmail.com)

A. Lipjankić, Faculty of Electrical Engineering Sarajevo, Zmaja od Bosne bb, University Campus, Sarajevo, Bosnia and Herzegovina; (phone: +387 61 637 577; e-mail: almir.lipjankic@gmail.com)

A. Handžić, Faculty of Electrical Engineering Sarajevo, Zmaja od Bosne bb, University Campus, Sarajevo, Bosnia and Herzegovina; (phone: +387 61 477 675; e-mail: asja.handzic@gmail.com

N. Nosović, Faculty of Electrical Engineering Sarajevo, Zmaja od Bosne bb, University Campus, Sarajevo, Bosnia and Herzegovina; (e-mail: nnosovic@etf.unsa.ba)

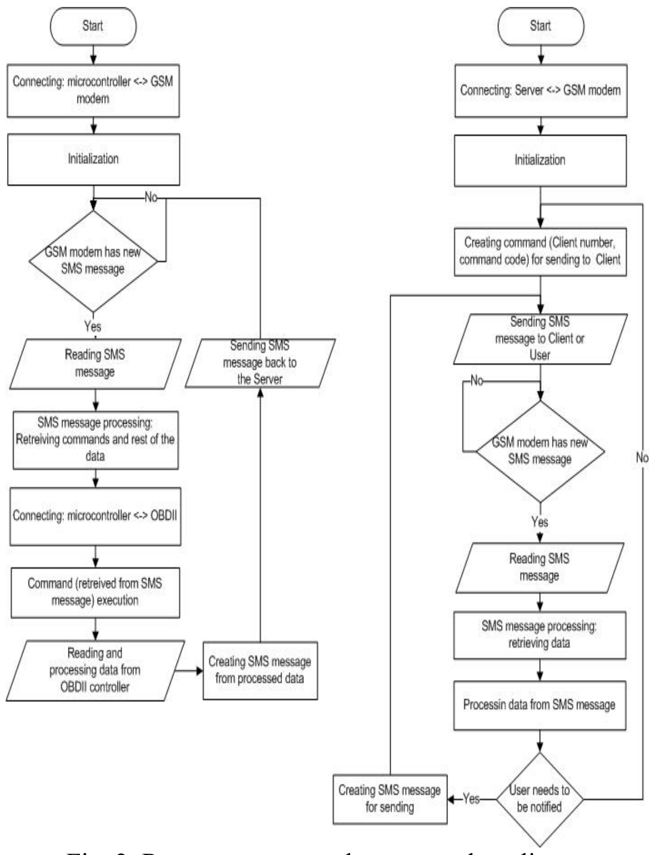

Start
Connecting: microcontroller <-> GSM modem
Initialization
GSM modem has new SMS message
No
Yes
Reading SMS message
SMS message processing: Retreiving commands and rest of the data
Connecting: microcontroller <-> OBDII
Command (retreived from SMS message) execution
Reading and processing data from OBDII controller
Creating SMS message from processed data
Sending SMS message back to the Server
Start
Connecting: Server <-> GSM modem
Initialization
Creating command (Client number, command code) for sending to Client
Sending SMS message to Client or User
GSM modem has new SMS message
No
Yes
Reading SMS message
SMS message processing: retrieving data
Processin data from SMS message
User needs to be notified
Yes
Creating SMS message for sending
No

Fig. 2. Remote server sends command to client

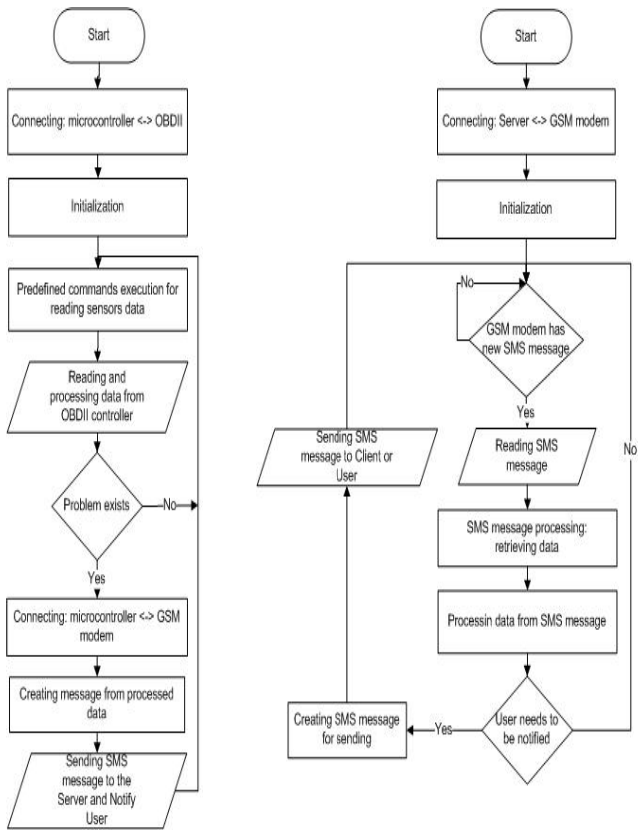

Start
Connecting: microcontroller <-> OBDII
Initialization
Predefined commands execution for reading sensors data
Reading and processing data from OBDII controller
Problem exists
No
Yes
Connecting: microcontroller <-> GSM modem
Creating message from processed data
Sending SMS message to the Server and Notify User
Start
Connecting: Server <-> GSM modem
Initialization
GSM modem has new SMS message
No
Yes
Reading SMS message
SMS message processing: retrieving data
Processin data from SMS message
User needs to be notified
Yes
Creating SMS message for sending
Sending SMS message to Client or User
No

Fig. 3. Client reports problem to server without request from server

Communication with the OBDII port is conducted through OBD commands while communication with the GSM modem is conducted through AT commands specified in the technical documentation of the modem.

Implemented methods of sending commands to the remote client and the client's reporting of a problem to the server are presented on Fig. 2 and Fig. 3.

IV. Presentation of Functionalities of Implemented OBD-GSM System

When basics of the communication standard are known, a system for remote diagnostic on a vehicle can be developed. Below is a description of the functionality for this system for remote diagnostic on a vehicle.

The application communicates with the vehicle and with a remote server. The application reads data from the vehicle, from the vehicle's ECU, and sends this data to the server. The server receives data from the client (vehicle), compares received data with data in the database, which contains data trouble codes, and sends answers and commands to the client.

Communication between the client (vehicle) and the server is conducted through SMS messages. A message in this communication has the following design:

- Unique Identification Number (ID)
- Phone number of sender
- Command (message content)
- Type of message (OBDII command, answer from the client, warning from the client, warning/notification for the client)
- Time

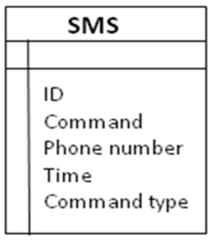


Fig. 4. Design of SMS message content

On the client side there is a computer or a device with similar abilities, a device for connection and communication with the vehicle and a device for communication with the server. An ELM323 adapter is used as a device for communication with the vehicle. The ELM323 is a 14-pin integrated circuit that, with only a few external components, is able to convert the OBD date format to the standard RS323 serial data format.

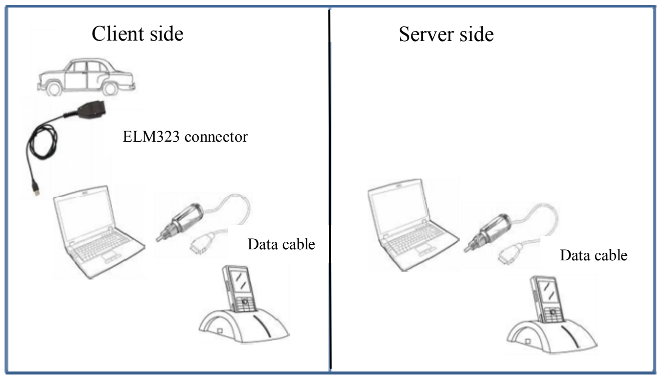


Fig. 5. OBDII-GSM system

A mobile phone is used for communication with the server. The computer has Windows XP operative system and .NET Framework 2.0 installed, drivers for serial port of a mobile phone and serial port of the ELM323 device.

The implemented application (called OBDII-GSM Box) is a client-server application.

A. *OBDII GSM Box – Client Side*

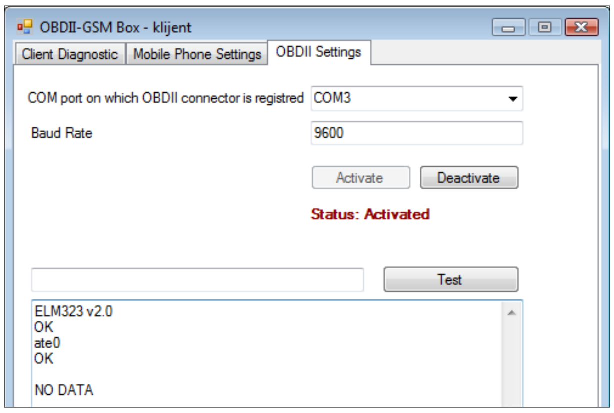
Client side of the application consists of three parts:

1. Part for the OBDII connector settings
2. Part for mobile phone settings
3. Part for communication between the client, vehicle and server

1) *Part for OBDII connector settings*

Part for the OBDII connector settings (Fig. 6) serves for setting the connection parameters and the connection between the computer and the vehicle using the ELM323 adapter. Following parameters should be defined:

- COM port on which the OBDII connector is registered
- Baud rate for port. By default, baud rate is 9600.



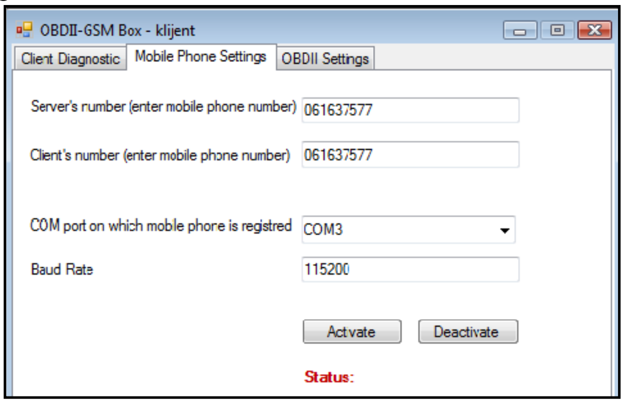
COM port on which OBDII connector is registred COM3
Baud Rate 9600
Status: Activated

```
ELM323 v2.0
OK
ate0
OK

NO DATA
```

Fig. 6. ODBII settings and connection establishing

After parameter settings, the connection will be established with a click on the command *Activate*. Now, initial communication with the ELM323 adapter is established. The program sends AT commands: ATZ, ATE0 and AT10. If communication is successful, a specific text should be visible in the textbox. An example of a successfully established communication is visible on Fig. 6.



Server's number (enter mobile phone number) 061637577
Client's number (enter mobile phone number) 061637577
COM port on which mobile phone is registred COM3
Baud Rate 115200
Status:

Fig. 7. Mobile phone settings

2) *Part for mobile phone settings*

Part for mobile phone settings (Fig. 7) serves for setting the connection parameters and the connection between the computer and the mobile phone itself. It is necessary to enter the telephone number of the server and client, COM port on which the mobile phone is registered and baud rate of COM port. After entering the settings, the connection will be established with a click on the command *Activate*.

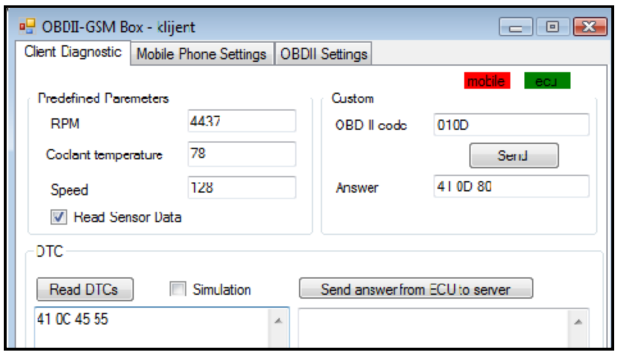
3) *Part for communication between client, vehicle and server*

Part for communication between the client (client's computer), vehicle and server serves for execution of OBDII commands and AT commands, sending queries to the server and receiving answers from the server.

First part communicates with the remote server using a mobile phone and a GSM network. This part does not require action from the user and communication (sending messages) with the server is automatic.

Second part communicates with the OBDII connector which is from the other side connected with a specific connector on the vehicle. This subpart has a few functionalities.

Reading data from sensors (Fig. 8). The application can read data from any sensor on the vehicle, but in this demo application, sensors and data for a vehicle's speed, coolant temperature and number of revolutions per minute (RPM) were chosen.

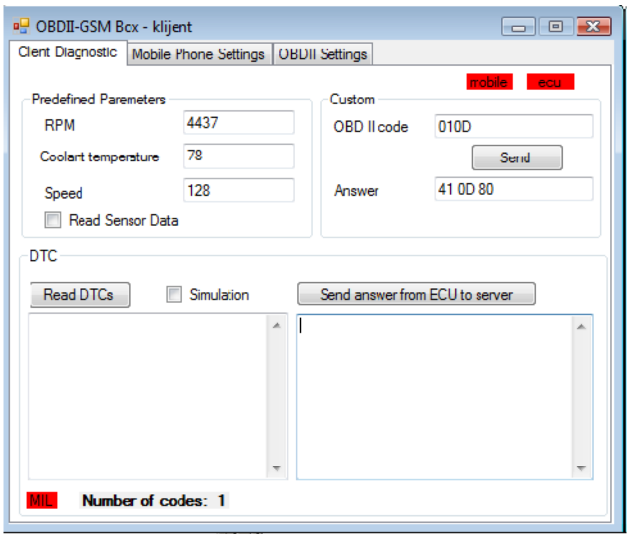


RPM 4437
Coolant temperature 78
Speed 128
[✓] Read Sensor Data
OBD II code 010D
Answer 41 0D 80

```
41 0C 45 55
```

Fig. 8. Reading data from sensor

Reading DTCs (Fig. 9). The application can read the MIL (Malfunction Indicator Lamp) lamp state, number of trouble codes and print registered trouble codes.

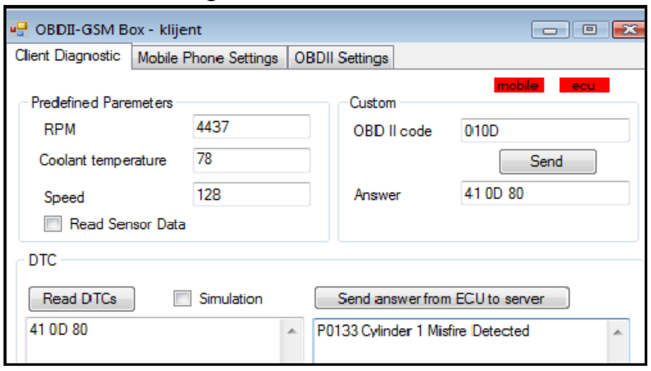


RPM 4437
Coolant temperature 78
Speed 128
Read Sensor Data
OBD II code 010D
Answer 41 0D 80
Number of codes: 1

Fig. 9. Reading DTCs

Sending code to server, receiving answer from server and displaying answer

Fig. 10. shows an example of sending a code to the server and receiving an answer from the server.



Predefined Paremeters

RPM	4437
Coolant temperature	78
Speed	128

☐ Read Sensor Data

Custom

OBD II code	010D
Answer	41 0D 80

DTC

☐ Simulation

```
41 0D 80
```

```
P0133 Cylinder 1 Misfire Detected
```

Fig. 10. Sending of error code to server

B. *OBDII GSM Box – Server*

Server part of application includes three parts:

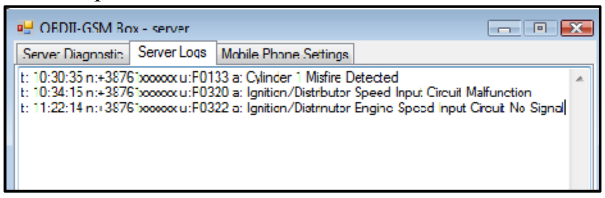
1. Part for mobile phone settings
2. Part for server logs
3. Part for communication between server and client

1) *Part for mobile phone settings*

Part for mobile phone settings is the same as this part on the client side of the application.

2) *Part for server logs*

Part for server logs (Fig. 11) shows all servers' answers to client requests.



```
t: 10:30:35 n:+38761xxxxxx u:F0133 a: Cylinder 1 Misfire Detected
t: 10:34:15 n:+38761xxxxxx u:F0320 a: Ignition/Distributor Speed Input Circuit Malfunction
t: 11:22:14 n:+38761xxxxxx u:F0322 a: Ignition/Distributor Engine Speed Input Circuit No Signal
```

Fig. 11. Display of server logs

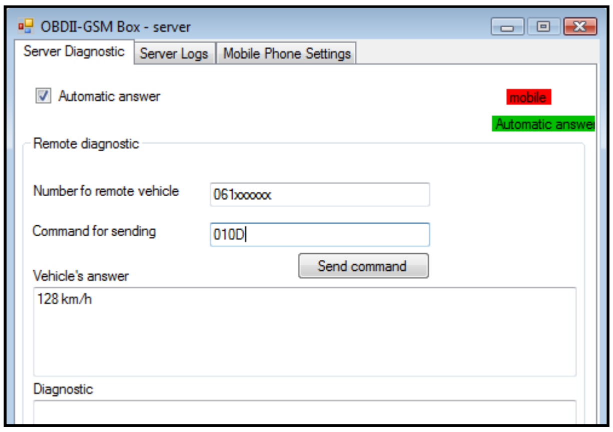
Every log message contains these parts

- t – time when answer was sent
- n – telephone number to which the answer was sent
- u – request received from the client
- a – answer sent to the client

In this application, answers are hard coded in application source code. Complete application holds these answers in database.

3) *Part for communication between client and server*

This part serves to execute remote diagnostic on a vehicle (Fig. 12). On the server side there is a possibility to enter a wanted command in the specific text box and send this command to a specific phone number of the client. When the client receives this command from the server, the client sends a command to a vehicle and reads an answer from the vehicle. Now, the client sends back this vehicle's answer to the server and the server displays the vehicle's answer.



☑ Automatic answer

Remote diagnostic

Number fo remote vehicle	061xxxxxx
Command for sending	010D

Vehicle's answer

```
128 km/h
```

Fig. 12. Display of client's answer on sever request

V. How To Develop The Proposed system in The Future

This paper presents the implementation of a simple system for remote diagnostic on vehicle which can be used by drivers or professionals in service centers. In current implementation, hard coded messages and data trouble codes are used in source code. One of the next steps can be developing of a complete data base on the server side. With a complete database application in order to send correct messages with the correct details about decoded errors and defects. The database should hold all information about the vehicle because this information can be needed later for detailed diagnostic and should have all information about services, their possibilities and locations. With this information the server can send a driver to the right address if a defect is registered.

Current message sending using SMS messages can be changed with sending data using GPRS. With this change the system will be cheaper; because GPRS is cheaper than SMS service and can send more data.

Another step is the integration of the proposed system for remote diagnostic with the system for navigation and location. With the diagnostic of the defect and current vehicle's location, the system will be able to search all service centers in the neighborhood where it is possible to remove the existing defect. If priority and severity of the defect are known, the driver will receive instructions where the nearest service center is.

The whole system is cheap and simple to use. Practical use of this system can be in continuing monitoring of a vehicle and its sensors, sending of warnings and notifications about defects to the driver and service center in near-real time. The advantage is work in real time so driver security will not be affected.

VI. References

[1] OBD II: Trouble Codes and Diagnostics: Quick Reference Guide,

[2] David P., OBD II Fault Codes Reference Guide, Kotzig Publishing

[3] http://www.onstar.com/us_english/jsp/explore/onstar_basics/technology.jsp

[4] The OBD-II Home Page http://www.obdii.com/

[5] http://www.developershome.com/sms/

[6] http://en.wikipedia.org/wiki/Engine_Control_Unit

[7] Carley L., Understanding OBDII: Past, Present & Future, 2005. http://www.aa1car.com/library/us796obd.html