THE DEVELOPMENT OF A REMOTE DIAGNOSTIC SYSTEM FOR VEHICLES

Attila Trohák, Máté Kolozsi-Tóth, Péter Rádi, László Méhes, Zoltán Biró

University of Miskolc, Hungary

Abstract: In the paper we would like to introduce the results of a research project with a public transportation company. We had to develop a system which is able to transfer the CAN data communication of a vehicle placed at a remote site into the central office of the company to make possible the diagnostics of the engine and the gearbox system. We have a lot of measurement data and experience at this field.

Keywords: remote diagnostics, vehicle, CAN bus

Introduction

A public transportation company has a lot of vehicles. The diagnostics of these vehicles is very difficult, because there are only a few diagnostic tools and skilled people. The company has several sites and the vehicles can fail also on route. The diagnostic software connects directly to the vehicle's internal CAN based network through a communication interface. Our task was to find solutions to transfer the data of the CAN communication from the vehicle to a remote location where the diagnostic software runs and the communication interface is placed.

1. CAN data conversion

If we would like to diagnose a vehicle on route we have to use wireless communication. The coverage of the broadband wireless Internet is not good enough in Hungary, so we tried to transmit the data on the GSM network [1.].
At first we tried to use an RS-232 – GSM/GPRS modem. In this case the vehicle's CAN based communication had to be converted to RS-232 serial communication. The result was that, the buffer of the gateway device overflowed and could not convert all the data.
After the CAN/RS-232 conversion failure we had to look for a new method to solve the problem. The amount of the received data from the CAN bus was too much to be converted by the RS-232 converter and was also too much to transmit with a GPRS connection.

1.1. CAN data filtering

To reduce the amount of data to be transmitted we started to analyze the traffic on the CAN network and we tried to filter it. The filter is a device which must have a CAN interface

to receive data from the CAN bus and it has to be able to do the further data processing, like filtering and transmitting to the PC or to the GPRS modem through RS-232.

1.2. CAN messages

First we had to get more details about the communication on the vehicle and we performed measurements to determine the actual data rate. The communication is defined by SAE J1939 standard [2.]. This is the vehicle bus standard used for communication and diagnostics between vehicle components, that are used in heavy-duty vehicles. SAE J1939 defines five layers in the 7-layer OSI network model, and this includes the CAN 2.0b specification (using only the 29-bit "extended" identifier) for the physical and data-link layers. SAE J1939 specifies exactly how the information (e.g. engine RPM) is exchanged between electronic control units (ECUs) on a vehicle. It defines the data's priority, size, scaling, and offset. For example, J1939 specifies engine RPM to have a default priority of 3, to have a size of 16-bits, a resolution of 0.125 rpm/bit, and an offset of 0. The standard defines many other aspects, including message timeouts, how large messages are fragmented and reassembled, the network speed, the physical layer, and how applications acquire network addresses.

2. Measurement system

We started to develop a measurement system based on a programmable controller which transmits the data to a computer.

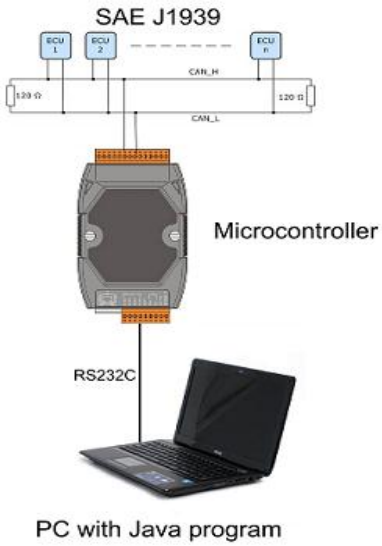

Figure 1. Measurement system

We used the C programming language to create an application onto the controller. The controller receives the CAN messages, processes them and transmits the useful information via serial port. The program has two functions.
The first prepares a list of identification data. This list is important to see how many messages are transmitted on the bus. Every new message is examined, and if an ID is already included in the list, then its number increases, otherwise the program adds the ID to the list. On the PC we can choose between different functions. After selecting a function, the length

of the measurement must be specified in seconds. The next value is the update time, this is set in milliseconds. The controller sends us the ID list with the selected intervals. If each new message would send the list, then there would be too much data, what could not be followed by the serial communication, so the measurement result would be incorrect. After this the measurement starts and transmits the data, when the time is up, then it goes back to the menu.
The second function is the filtering, when only those messages are sent to the computer which is specified by their ID. The filtering starts after the IDs had been entered. The PC can easily and quickly evaluate the received data from the controller.

3. The PC program

During the project a PC program was developed in Java language. The advantage of this is the program's platform independency. The application has a graphical user interface to make the interaction and the monitoring progress easier. The program implements the Java Communication API 3.0 from Oracle to make the communication possible between the programmable controller and the PC via RS-232. To store the received data on PC in convenient format, our Java program also implements the Apache POI 3.8 beta, which can handle Microsoft document formats and makes Microsoft Excel type spreadsheets from the received and calculated data in formatted style with additional diagrams. This API is also used to get the base data on start.

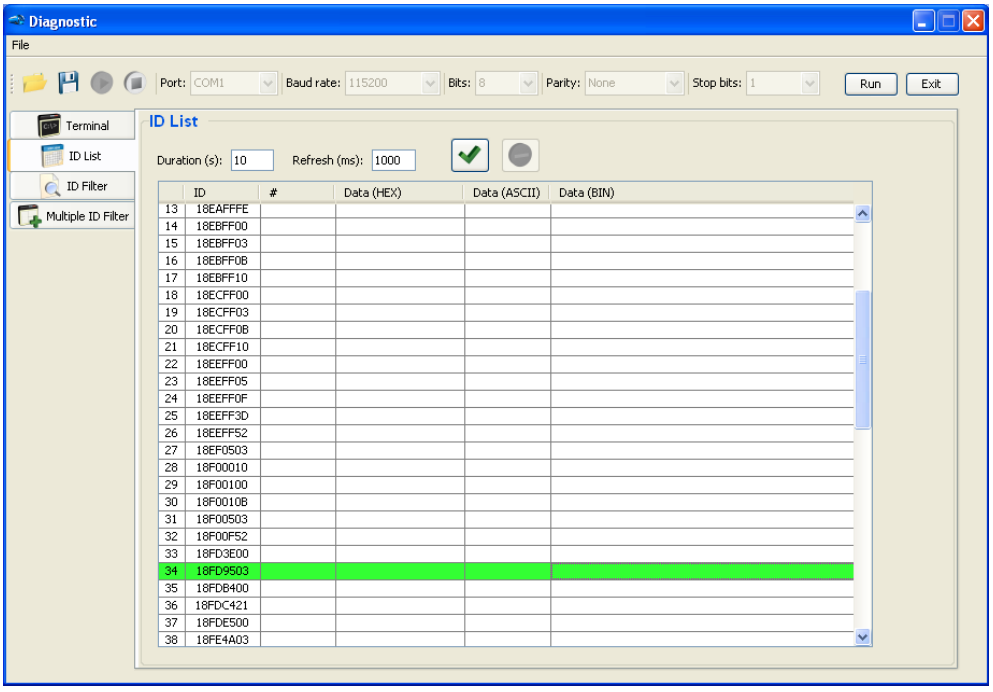

Figure 2. The PC application

This application has 3 main functions:

- Terminal: This function provides a terminal window between the programmable controller and the user over the RS-232 bus, like the operation system's default terminal (HyperTerminal, Putty). In this mode the user can use the programmable controller functions in text mode.

- ID Listing: It interacts with the controller ID listing function. In this mode a table shows the received data in formatted style. In this table appears the ID of the CAN message, how many message received with this ID, the data of the CAN message in hexadecimal, ASCII, and binary format. After this measurement the summary of the messages will appear. Before using this function we have to set the duration of the measurement and the refresh time.
- ID Filtering: This function does real time data monitoring on one or more selected IDs. Because of the low bandwidth, the number of the parallel monitored IDs is limited to 5. On the screen a continuously expanding table shows immediately the received and the additional data like ID, the time of arrival in millisecond and the bytes of the data from 7 to 0 in hexadecimal format. If there is no element in the list we want to monitor, we can add new IDs to the list manually.

The application can store the results of ID filtering, and ID listing in Microsoft Excel spreadsheets.

4. Measurement data

Seven 60-second long measurements were made in various situations. First we connected our system to the CAN network of the vehicle, we started the measurement and after it the ignition was switched on. In this case we received 74 message IDs, the message count was 27,872.
In the next two cases the measurement started after turning on the ignition. In both cases 58 IDs appeared on the CAN bus. The message count was 28,303 and 28,306.
At the first measurement there were 16 more IDs on the CAN bus comparing to the second and third measurement. Most of these messages were Address Claimed Message, which used when the ECU's are setting up their addresses. The message count at the first measurement was less, because we switched on the ignition after the measurement started, so less messages have arrived.
During the fourth measurement we connected the diagnostic tool to the CAN network. Then 67 ID and 28,437 messages were received. At this measurement 9 more identifiers appeared. These messages are used for the communication between the diagnostic software and the devices on the CAN network.
The last three measurements were made with the engine running, in this cases 58 IDs were received and the message count varied between 28,311 and 28,317. We can say there is no difference in the data rate between the engine running and the under ignition cases.

Table 1. Measurement data

Case	**Number of IDs**	**Message Count**
switching on the ignition	74	27872
with ignition 1	58	28306
with ignition 2	58	28303
with diagnostics	67	28347
engine running 1	58	28317
engine running 2	58	28313
engine running 3	58	28311

After this we examined the data of the messages. A complete CAN message contains several parts. The next picture shows how the J1939 standard uses the CAN 2.0b specification.

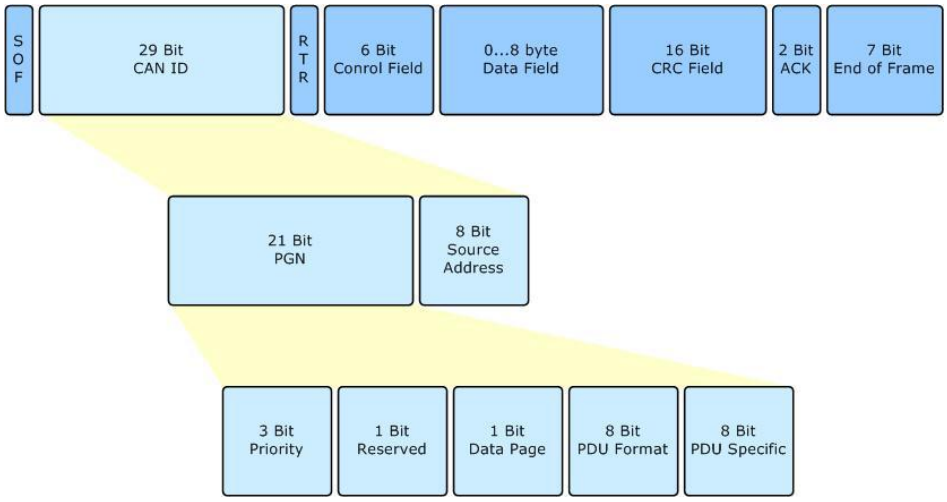

Figure 3. The structure of a CAN message

The PGN (Parameter Group Number) is an important parameter, as this allows us to determine what information is included in the Data Field. Just have to look after it in the J1939 standard part 71. Our aim is to query, transmit and delete fault codes. We can do these with Diagnostic Messages (DM) according to the J1939 standard part 73. This standard defines 13 DMs with different PGNs. The measurement to determine these Diagnostic Messages is under preparation.

5. Summary

In the future we will make development on the PC application, like binding the physical meaning and value to the received data and show it graphically. We intend to do further development on the other side of communication with the CAN bus and send messages to it. We will finalize the filter system and we will be able to reduce the CAN traffic and in this case it can be converted to RS-232 and can be transmitted through a GPRS connection.
For a transportation company it is useful, because they can easily diagnose their vehicles on remote sites or on route and they can reduce costs by making good decisions when a vehicle goes wrong. This system also can be useful for transportation companies for advanced fleet management. And of course in the future we are planning more measurements on different kind of vehicles with different kind of diagnostics tools and interfaces.

Acknowledgements
This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 and TAMOP-4.2.2/B-10/1-2010-0008 projects with support by the European Union, co-financed by the European Social Fund.

References

[1.] AJTONYI, I.: **Ipari kommunikációs rendszerek IV**. , AUT-INFO Kft., 2011
[2.] J1939 standard