Vol. 11, No. 1/2021

Journal
of Danubian
Studies
and Research

Development of a Device for Displaying on an Additional Screen the Parameters Read from the Vehicle's on Board Computer

Barothi Laszlo[1], Voicu Daniela[2], Stoica Ramona-Monica[3], Deleanu Lorena[4]

Abstract: Present paper aims to present the steps of developing a prototype used for the display of several functional parameters which does not normally appear on the dashboard of an automobile. Data is collected from the vehicle CAN bus, follow-up processed with a MCP2515 microcontroller and a ATMEGA328P and finally displayed on a color touchscreen. The prototype is a Plug&Play device, which can easily be connected to the vehicle's OBD2 socket, from where the data is collected. The parameters are shown in an interactive manner, by displaying one or multiple parameters at a time, on a single page.

Keywords: engine temperature; battery voltage; ambient temperature; MCP2515

1. Introduction

Nowadays, automobiles are electronically controlled and actuated for two reasons: to reduce the exhaust gases like NOx and at the same time, to increase engine performances and vehicle dynamics.

To that end, each vehicle is equipped with microcontrollers or microcomputers which ensure the proper functioning of different subassemblies. Some vehicles, of higher performances or belonging to the premium class can have up to 120

[1] PhD, Military Technical Academy "Ferdinand I", Romania, Address: 39-49 George Cosbuc blvd., Bucharest, Romania, E-mail: arothi.laszlo@mta.ro.
[2] PhD, "Dunărea de Jos" University of Galati, Romania, Address: 47 Domneasca Street, Galați Romania, Corresponding author: daniela.voicu@mta.ro.
[3] PhD, Military Technical Academy "Ferdinand I", Romania, Address: 39-49 George Cosbuc blvd., Bucharest, Romania, E-mail: ramona.stoica@mta.ro.
[4] PhD, "Dunărea de Jos" University of Galati, Romania, Address: 47 Domneasca Street, Galați, Romania, E-mail: lorena.Deleanu@ugal.ro.

microcontrollers. On such computers, every milisecond there is plenty of information travelling between end points. From all data travelling through the CAN bus, few are available for the user/driver to see. Some drivers find very useful some particular parameters which are not displayed on-board. Such examples, in case of Dacia Logan II automobile, Sandero II and Sandero Steppwey II are engine temperature, charging voltage etc.

Present paper presents the steps to create a prototype, which can be placed on a phone stand, able to display different parameters on a LCD, such as engine or ambient temperature, battery voltage etc. The data is received from the CAN bus, through the two existing wires from every modern vehicle: CAN LOW and CAN HIGH. It is also used a Microchip MCP2515 microcontroller. Data is further transfered to another microcontroller, ATmega328p, which is embedded within an Arduino NANO and it has the role to process input data in order to display the values on a LCD Nextion NX3224K024.

2. Experimental Research

2.1. CAN bus

The first step when performing the experimental research is to undertand the data package which travels on the CAN bus (through CAN LOW and CAN HIGH), for every Dacia vehicles. According to standard ISO 11898 each package contains four key elements regardless of the vehicle make or model:

- arbitration ID: is a broadcast message which identifies the device ID of the one trying to communicate, even if every device can send multiple arbitration ID. If there are send two packages at the same time, the winner is the one with the lowest arbitration ID;
- identifier extension bit (IDE): this bit is always zero for standard CAN. Actually, it represents a space between two information;
- data length code (DLC): this is the data dimension, which varies from 0 to 8 bytes;
- data: information which is about to be transmitted.

In figure 1 it is presented the structure of such a message along with voltage variations between CAN LOW and CAN HIGH. There is also observed that the two signals are always of opposing voltages.

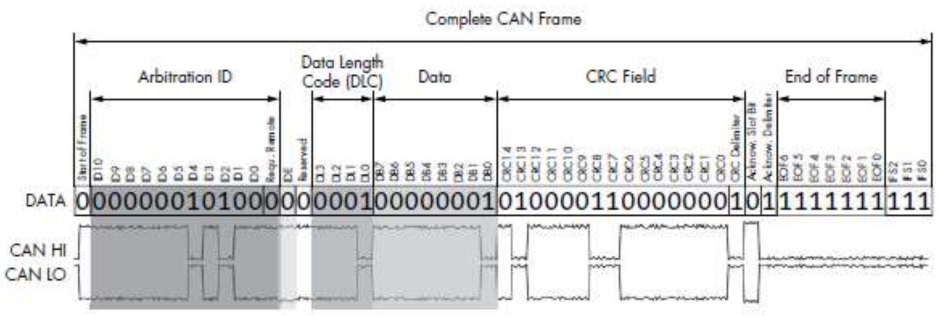


Figure 1. Message Structure on CAN

2.2. Data Read from the CAN Bus

According to [0], each package of data present on CAN bus, has a specific ID, identical for each vehicle manufacturer, which is established according to standard ISO 15765-2. The ID provides information regarding the parameters which are to be send.

In table 1 there are also presented the PIDs (Parameter IDs) specific to the Dacia automobile. Currently, it is quite a challenge to decipher these kind of codes, for each model of vehicle, unless there are indicated by the manufacturer.

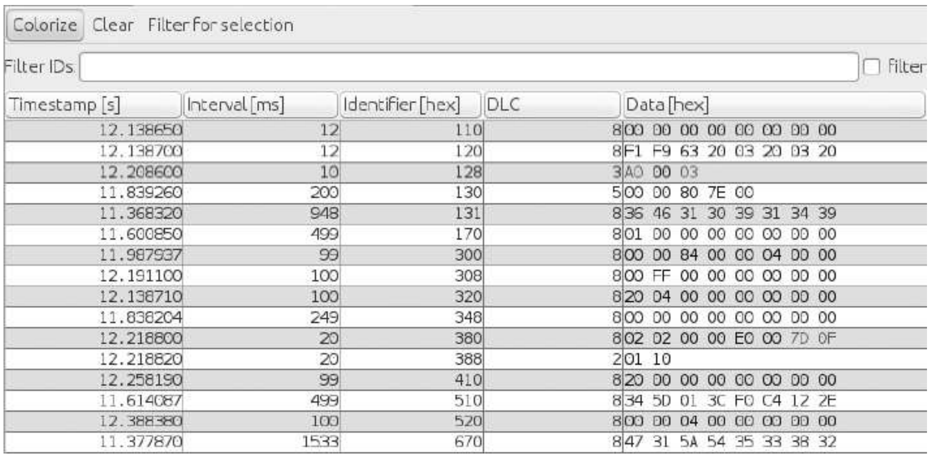
Table 1. PID's of Important Parameters.

Parameter	PID Hexadecimal	PID Decimal	PID Specific to DACIA	Calculus manner
Coolant temperature	05	5	0x05DA	A-40
Ambient temperature	46	70	0x3B7	A-40
Battery voltage	42	66	0x4AC	A/10

The read of data can basically be done with the use of devices such as ”CAN Sniffer”, which are able to provide information without any changes.

A very usefull CAN Sniffer is the KAYAK free software which can be installed on Linux Kernel. As a prerequisite, there should also be installed the Can-utils, which enables Linux to communicate with the CAN.

In figure 2 it is presented a part of the graphic interface of Kayak software, namely the part containing CAN messages. It is observed the interval between messages, the message ID, the length of information transmitted, expressed in bits and actual data. In order to understand the designation of each frame, first it is performed a numerical identification of recorded parameters. To this end, it is needed to know the constituents of the communication network for the specific vehicle. Follow-up, from the established interval (after the messages are repeated), it is performed an analysis to determine primary messages. Hence, a message (with the same ID) which appears at short time intervals, is categorized as important message, meaning that it could come from a subassembly critical to vehicle functioning. An example of such assembly is the vehicle engine.



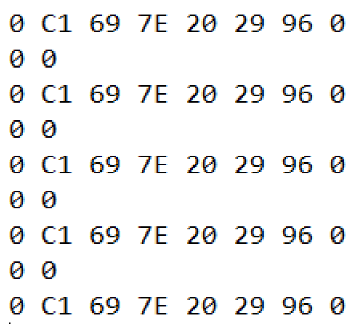
Timestamp [s]	Interval [ms]	Identifier [hex]	DLC	Data [hex]
12.138650	12	110	8	00 00 00 00 00 00 00 00
12.138700	12	120	8	F1 F9 63 20 03 20 03 20
12.208600	10	128	3	A0 00 03
11.839260	200	130	5	00 00 80 7E 00
11.368320	948	131	8	36 46 31 30 39 31 34 39
11.600850	499	170	8	01 00 00 00 00 00 00 00
11.987937	99	300	8	00 00 84 00 00 04 00 00
12.191100	100	308	8	00 FF 00 00 00 00 00 00
12.138710	100	320	8	20 04 00 00 00 00 00 00
11.838204	249	348	8	00 00 00 00 00 00 00 00
12.218800	20	380	8	02 02 00 00 E0 00 7D 0F
12.218820	20	388	2	01 10
12.258190	99	410	8	20 00 00 00 00 00 00 00
11.614087	499	510	8	34 5D 01 3C F0 C4 12 2E
12.388380	100	520	8	00 00 04 00 00 00 00 00
11.377870	1533	670	8	47 31 5A 54 35 33 38 32

Figure 1. KAYAK Interface, CAN Sniffer

Based on the identifiers presented in Table 1, the following step is to determine the groups of bits which vary with the signal coming from a particular sensor. This may lead to the conclusion that the bit is the information carrier for the parameter. If the corresponding bit cannot be determined from the first attempt, the process is iterative until it is obtained the result, according to specialty literature [0].

In order to determine the specific identifiers, it was developed a device based on an Arduino NANO and a Microcontroller MCP 2515 (a device which is also used to design the prototype, still, the difference consists in the software uploaded in the memory of the Arduino platform) which shows the information travelling on the CAN LOW and CAN HIGH of the Dacia Sandero Stepway automobile, thus allowing to verify the results.

Therefore, in figure 3 it can be observed a small sequence, which was already selected with the use of the identifier specific to engine temperature in case of Sandero Stepway vehicle. During the recording of data, engine temperature was 56 ^{o}C. The question which arises is why the 56 and not the 96 value and why does it appeared on the 7th position.



```
0 C1 69 7E 20 29 96 0
0 0
0 C1 69 7E 20 29 96 0
0 0
0 C1 69 7E 20 29 96 0
0 0
0 C1 69 7E 20 29 96 0
0 0
0 C1 69 7E 20 29 96 0
```

Figure 2. Data Recorded and Presented in Serial

According to [0], it is known that a byte contains eight bits. In this case, the byte is from group A.

Messages on CAN can be structured on four or eight bits. Group A means the first byte, the second group is noted B and the last group si noted D. Each bit from group A is noted from 0 to 7, in the following order: A7, A6, A5, A4, A3, A2, A1 and A0. As a consequence, the bit noted A1 represents the value of engine coolant temperature, and from the value read on the CAN bus (decimal), it is subtracted 40, meaning A-40, where A represents a byte. Also, in this case the A7 bit represents the ambiental temperature value, from which it is subtracted 40. These 40 units are artificially added because in hexadecimal only natural numbers can be expressed and not the negative too.

As an example, during winter season, the coolant temperature while vehicle start can have negative values, which is why the minimum temperature that can be recorded by the vehicle's on-board computer is -40ºC.

The three afore-mentioned identifiers determined for the three analysed parameters are:

- 0x5DA for engine temperature;
- 0x3B7 for ambient temperature;
- 0x4AC for battery voltage or working voltage of the computer's chassis (EBM-Electronic Body Module).

With the data thus obtained, the next step is programming the Arduino microcontoller for data analysis and initialization of Nextion screen for data display.

3. Final Product

The final product, without integrating the constituents into a case, is presented in figure 4 (while running) and in figure 5 (while shut down). The components are as follows: the OBD2 socket, power cable, cables for the two CAN wires, prototype test cable, DC-DC voltage reducer from 12-14.8 V to 5.75 V, Arduino Nano, Microcontroller Microchip MCP 2515 and the touchscreen display type Nextion NX3224K024.

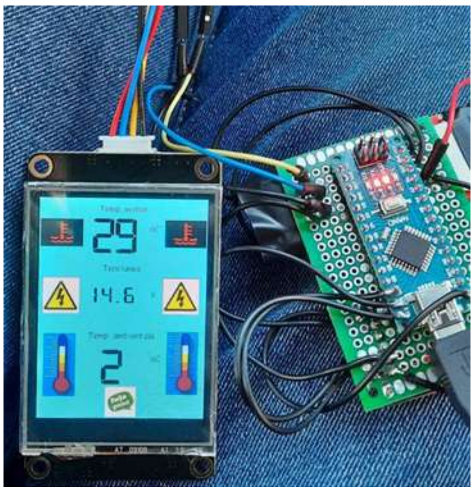


Figure 3. The Prototype during Functioning

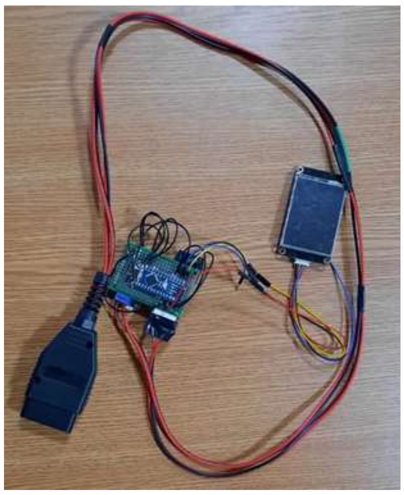


Figure 4. The Prototype Constituents

The data transfer between the Arduino Nano microcontroller and the one of the Nextion screen, the latter being responsible for the image display of usefull information received from the Arduino, it is performed through the serial interface, at a Baud rate of 11520 Hz. There are used the RX (receive) and TX (transmission) pins. This frequency is provided by the MCP2515 microcontroller because it has the same frequency as the setting Baud rate.

Initial programming setting for the MCP2515, for CAN are as follows: (CAN1.begin(CAN_500KBPS, MCP_8MHz)) 500KBPS - data transfer frequency on CAN protocol and 8 MHz for the working frequency of MCP2515.

Future improvements of the prototype may include to integrate the hardware into a casing, for a better user experience in terms of value reading. The casing can be 3D printed, according to the .STL file, from a CAD software. Also, the prototype can be placed as the user finds usefull, whether it is on a phone mount or in other variants.

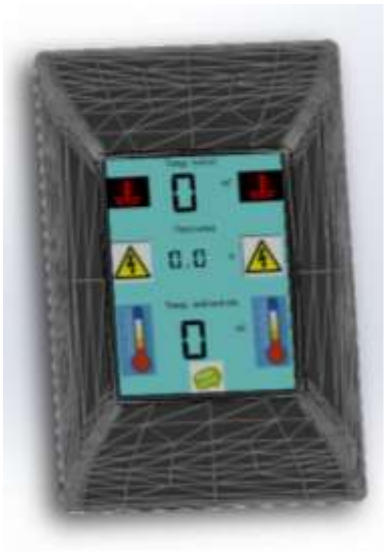


Figure 6. Final Display of Parameters

4. Conclusions

The created device has the disadvantage that being a home-made prototype, the current consumption is relatively high: 0.105 A, which means that on the long term, it should not be left plugged in to the OBD2, despite the fact that the vehicle is stopped. Such a situation can lead to the battery discharge. In order to improve the system, it can be added another source to the existing one on the OBD2 socket and it can be changed the DC-DC voltage reducer to a top range solution, thus decreasing the current consumption.

Also, the disadvantage of understanding the messages from the CAN consists in the easy accessing by hackers. In this manner, wrong information can be introduced in the CAN bus, by knowing the message structure.

On the other hand, the advantage of understanding the messages on the CAN bus can lead to understanding the architecture of the entire vehicle. Follow-up, the user can intervene on th vehicle electronic part, at a higher level.

5. Acknowledgement

“This work is supported by the project ANTREPRENORDOC, in the framework of Human Resources Development Operational Programme 2014-2020, financed from the European Social Fund under the contract number 36355/23.05.2019 HRD OP /380/6/13 – SMIS Code: 123847.”

References

Smith, Craig (2016). *The car Hacker’s handbook.* San Francisco: Ed. No Strach.

https://nextion.tech/instruction-set/.

https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00.

http://www.can232.com/docs/canusb_manual.pdf.