IOT DIAGNOSTICS FOR CONNECTED CARS

Gheorghe PANGA, Sorin ZAMFIR, Titus BĂLAN, Ovidiu POPA
"Transilvania" University, Braşov, Romania (titus.balan@unitbv.ro)

DOI: 10.19062/2247-3173.2016.18.1.39

Abstract: *Internet of things (IoT) is a relatively new concept identifying various computing devices connected over the internet, differentiated by unique identifiers (IP). Considering the IoT protocols (CoAP, MQTT) as becoming more mature, there is still no generally accepted solution for intelligent management of these devices, though several initiatives are present. Our aim is to demonstrate the versatility of these technologies by implementing a special use-case related to the automotive world, emphasizing the connection between IoT and the connected car. We have chosen the field of automotive diagnostics for our demonstrator considering it represents a crucial path for future intelligent transportation systems.*

Keywords: *IoT, automotive diagnostics, MQTT,Mosquito, Raspberry PI*

1. INTRODUCTION

The idea behind IoT resides in the "thing" term, which includes any natural or artificial object that can be assigned an IP address and provided with the ability to transfer data over a network.[1] As a result we can encounter all sorts of connected devices that are part of this ecosystem.

Even if we talk about an automobile built with fuel sensor capable of alerting the driver when the tank is empty, a pet tracking device or even a heart monitor system implanted inside the body of a patient, the IoT protocols[2] will allow monitoring and controlling remotely [3], over an existing network resulting in improved accuracy, efficiency and security.

In the past decade, because of the new features introduced by the car manufacturers, the vehicles have become increasingly more software driven and at the same time dependent. When it comes to connecting drivers and technology, the auto industry has a longer and richer track record than any other sector by offering a series of improvements aimed not only at safer driving (e.g. ABS, ESP , airbags, cruise control) but also at improving the driving experience overall.

However, diagnostics and maintenance have always been something reserved to the specialized repair shops using custom tools and specific methods, and ultimately these procedures are always related to physical connectivity to the car. It is under this status quo that we take initiativeinto a research project that aims to assure and enhance the car-to-driver and car-to-car communication by studying how IoT can eliminate the need to be in the proximity of a car to perform diagnostics.

By monitoring all the aspects of the car is easier to detect any problem in advance by sending all sensor readings to a certified center where technicians and engineers will apply their expertise to find and predict imminent failures of key systems integrated in the vehicle.

FIG. 1. The Connected Car concept

Besides the diagnostic part, this system would help drivers to keep track of the yearly maintenance, programming service appointments or analyzing the fuel consumption and giving advices for a more eco-friendly driving style.

In a difficult situation like a traffic accident, such a system would be capable to inform the emergency rescue service when the driver is incapacitated, improving the chances for the wounded to live by reducing the overall time of the rescue mission.For the business sector, it can provide an enhanced way to monitor the entire fleet of company cars using open protocols and reducing the costs derived from the already custom solutions on the market.

2. THE IOT PROTOCOL

Our technology choice is MQTT [1] which is alight weight, messaging interchanger protocol based on publish-subscribe model.This pattern is an alternative to the traditional client-server model and requires a message broker (server) which is responsible for distributing messages, over the network to interested clients (devices) based on the topic of a message they are associated with.

The publisher is the client who will send a particular message to a broker and the subscriber is the client who will receive the message from the broker.The publisher and subscriber don’t know about the existence of one another and only way for them to communicate is through the broker, known by both the publisher and subscriber, which filters all incoming messages and distributes them accordingly.

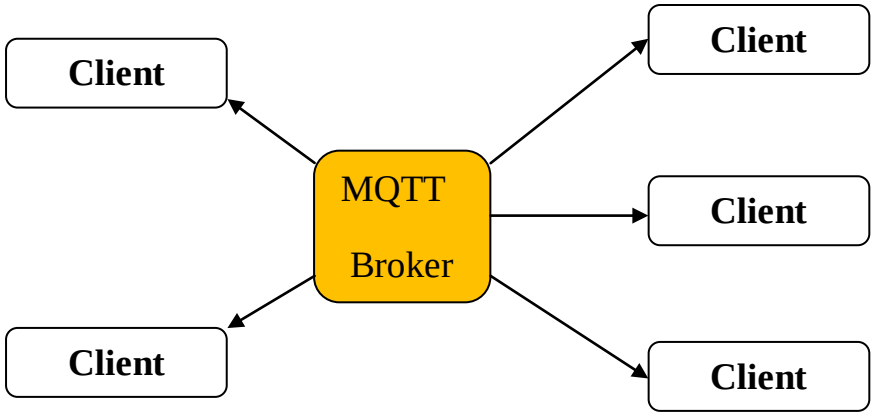

FIG. 2. "Client-Broker" structure of MQTT protocol

One of the most relevant MQTT [2]aspects for this project are decoupling of publisher and subscriber, which can be differentiated in three dimensions:

- *Space decoupling*: It's not necessary that the publisher and subscriber are at same physical location (however a client can be both a publisher and a subscriber at the same time)
- *Time decoupling*: Publisher and subscriber do not need to run at the same time (this is useful for situations when the connection between these two entities is not stable)
- *Synchronization decoupling*: The publishing or receiving operations will not halt any other device operation currently running.

MQTT protocol is designed for interconnecting remote locations where a message is required to have a small footprint or where network bandwidth is very limited. The implemented demonstrator uses Mosquitto[3] as an open source (EPL/EDL licensed) server/broker which implements MQTT protocol.The main advantage and the reason it was chosen for this project was the cross platform capability, Mosquitto offering support for Windows, UNIX kernel based Linux distribution, IOS and Android.

3. THE MONITORING AND CONTROL UNIT

The MCU (monitoring and controlling unit)representsthe car mounted device that connects to the vehicle through OBD-II[4] port and is used for performing the actual diagnostics of the vehicle.

The MCU collects the messages interchanged between ECUs(electronic control unit) and sensor, actuators or any other computers, on the CAN – Bus network, processes them and sends the raw data to a local MQTT client through a serial type communication (RS232/ USB or Bluetooth).By implementing the ISO 9141-2 and SAE J1939 standards of the OBD II protocol, MCU is compatible with a large variety of car manufacturers.

As you can observe in Fig. 3 both standards use a two line communication:

- *K line:* bidirectional line used for initializing the communication between ECU and MCU.
- *L line:* unidirectional line used for data transfer between ECU and MCU.

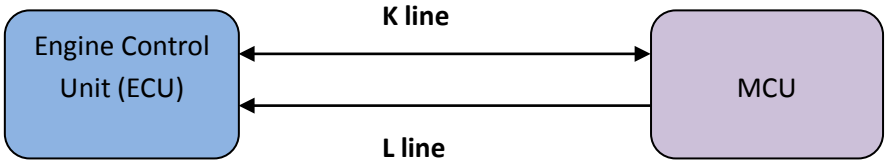

FIG. 3. ISO 9141-2 and SAE J1939communication

Because ISO 9141-2 and SAE J1939 [5] standards are used in theautomotive industry, the reference voltage for the logic levels is 12 V (V_B).

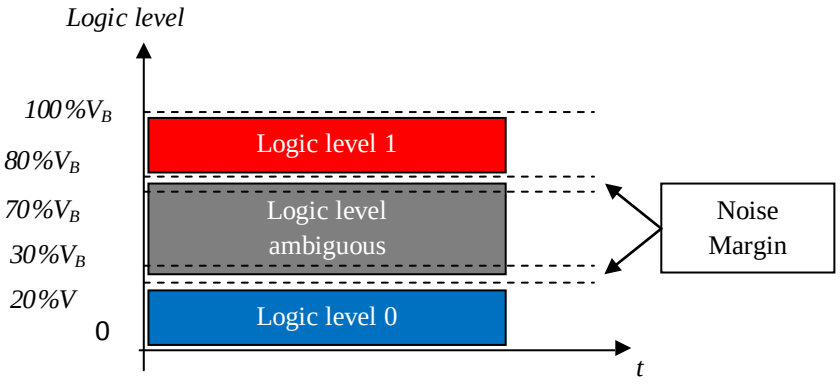

FIG. 4. Logic levels for K line transitions

4. DEMONSTRATOR FOR CONNECTED CAR CONCEPT

4.1 ECU diagnostic stack

The MCU is based on the ELM327[6] chipset capable of interpreting the two CAN based protocols and is triggered by means of an UART interface using AT commands. The following code snippets illustrate how the diagnostic queries are passed to the car:

Table 1. Initialization of the UART interface

```
void uart_init (unsgned int baudrate_init)
 {
       UBRRH=(unsigned char)(baudrate_init>>8);
       UBRRL=(unsigned char) baudrate_init;
       UCSRB|=(1<<TXEN)|(1<<RXEN);
       UCSRC|=(1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);

 }
```

After the UART interface has been properly set up the AT commands are sent through the following two methods:

Table 2. Sending and receiving diagnostic through UART

```
void uart_transmit_string(char
*send_string)
 {
while(*send_string)
       {
uart_transmit(*send_string++);

       }
 }
```

```
void uart_recieve_string(unsigned
char *buffer)
{
int s = 0;
while (UCSRA & (1 << RXC))
{
buffer[s] = UDR;
s++;
}
UCSRA |= (1 << RXC);
}
```

4.2 MosquittoMQTT Client

The MQTT client is used mainly as a publisher for the data received from the MCU device. All the sensor related valuesare sent over a mobile network to a broker and from there to a web application where it can be presented in a human readable form.

For the hardware platform a Raspberry PI v3 B is considered to be suitable to sustain the processing power necessary for the system and also because it offers stable development platform.Also,the platform is easily extensible andcould provide real time location through an additional GPS module incorporated in the system for more accurate measurements, real time fleet monitoring for business and companies or for car-to-car signaling of different traffic events like accidents or road blocks.

Another aspect that was considered in the choice of hardware was the limited size and the low power consumption constrains that are usually specific to the automotive sector. The entire client system is car mounted alongside MCU which represents a simple to implement solution for car producers or for any driver who wants to make his car smarter.

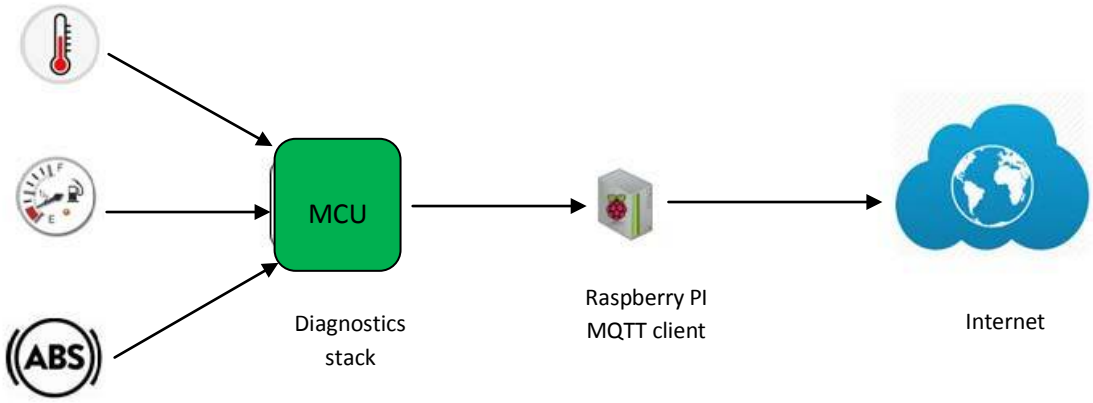

FIG. 5. Connected car client side (car mounted)

Mosquitto MQTT Server

The backend side is represented by theMQTT Server which both the broker and a Web application used for administration purposes. It is used for data interchanging between the publishers (multiple clients)and subscriber and resides in the Internet behaving like a data aggregator.

While a future endeavor would be to create a powerful backend with a rich user interfaces,our focus for the time being revolves on the communication between the car and the internet and more specifically the transmission of diagnostic information by means of an IoT protocol.

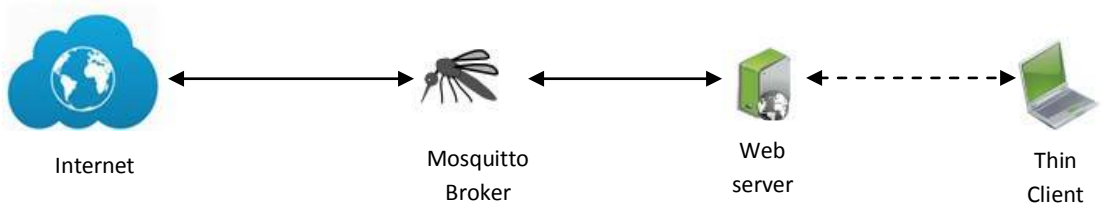

FIG. 6. Connected car server side

5. INTEGRATION AND TESTING SCENARIO

For illustrative purpose we have chosen to present a vehicle use-case signaling an error that was reported by an ECU, but the actual scenario can be carried for several vehicle parameters used in monitoring and diagnostics.

In a real testscenario wehave specifically chosen a vehicle with a known intake manifold problem.This problem is reported by the car ECU as P0103 (Mass or Volume Air Flow Circuit Low Input) error code.

On the CAN-Bus network, data between ECU, sensors and others control modules ispassed using a hexadecimal format. This means that our vehicle under test the intake manifold error message is transmitted to the ECU as: `44000066` which can be translated in a more human readable form as follows:

- first 2 octets represent the mode of operation (in our case mode 03-Diagnostic stored trouble code [7])
- the next 2 octets indicate the power train assembly
- the last 4 octets are the error code, so in our case, this error code has the value 0103

The MCU is connected to CAN-Bus network through OBD-II port placed, usually, under the steering wheel and will interpret the ISO 9141-2 and SAE J1939 specificframe and extract the data of interest.The value P0103extracted from the error message is then sent to Raspberry PI which hosts the Mosquitto client, through RS232 serial communication.

FIG. 7. Client assembly

```
pi@raspberrypi:~ $ mosquitto_pub -t "car/events" -m "P0103" -q 1 -r
pi@raspberrypi:~ $
```

FIG. 8. Mosquitto client publishing the error code message intercepted from CAN-Bus Network

The client will publish (see Fig. 9), using MQTT protocol, the value on /car/events topic on which it is associated and then the broker will assure that the message is delivered to a web application which serves as a remote interface between machine and human factor.

FIG. 9. Mosquitto broker receiving P0103 error code form Moquitto client over the IP network.

Then the web application searches through a local data base in which we store the mapping between the error codes and the human readable format and sends back a full description of the failure message as follows.

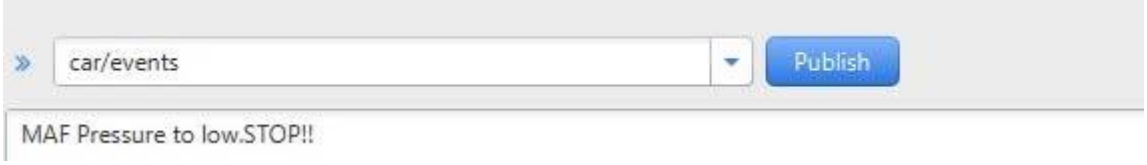

FIG. 10. Mosquitto broker publishing the error description to the topic /car/events.

```
pi@raspberrypi:~ $ mosquitto_sub -t car/events -h 192.168.1.137
MAF Pressure to low.STOP!!
```

FIG. 11. Error alarming via Mosquitto client in subscribe mode

Subsequently, a different client can be used for notification purposes thus informing the person of interest of the available events (in our case the car error code). This notification mechanism could be employed using a smart phone or even a smart watch.

Throughout testing phase we encounter a limitation of our solution: because of fact that the ECU firmware is proprietary we are not able to modify or take actions on the car itself (e.g. resetting the error code, running an extended diagnostic session etc.). Nevertheless the reporting itself is useful for long term analysis and more accurate diagnostics when it comes to sporadic errors, since a mechanic can also take into considerations factors that are not easily reproducible in the repair shop (e.g. temperature, humidity, elevation).

CONCLUSIONS

We were able to illustrate a scenario for IoT Connected Car using MQTT andthe Mosquitto broker where a technical problem, unnoticed by most of the drivers, can represent a serious risk of mechanical failure or even human live threatening situation.Our IoT integrated solution will provide the necessary technical background in order to avoid similar scenarios and even offer a way for remote monitoring the vehicle or traffic events.

Further, our solution can be enhanced; taking benefit of the OMA defined management protocols, by implementing a remote management solution of the MCU allowing functional updates and backend triggered actions.

Due to the factthat the ECU firmware is proprietary we were not able to modify or take actions on the car itself. In order to completely implement the Connected Car concept, including also a reaction loop for remote or self-healing procedures, the implemented solution should be synchronized with the car manufacturer, by creating a trust zone between Engine Control Unit and IoT management devices.

REFERENCES

[1] Giusto D., Iera A., Morabito G., Atzori L. (Eds.): *The Internet of Things*, Springer, 2010
[2] Gubbi J. et al.: Internet of Things (IoT) - *A vision, architectural elements, and future directions, in Future Generation Computer Systems*, 29, pp 1645-1660, (2013)
[3] L. Atzori et al.: The Internet of Things - *A survey, in Computer and Telecommunications Networking 54*, (2010)
[4] Message Queuing Telemetry Transport - v3.1.1. OASIS MQTT TC, http://docs.oasisopen.org/mqtt (2014)
[5] Message Queuing Telemetry Transport Org, http://mqtt.org (2016)
[6] Mosquitto. *Open source MQTT broker*, http://mosquitto.org/
[7] Huan Pan, ISO 9141-2 and J1939 Protocols on OBDII, University of Changaua, 2009.
[8]The OBD home page, http://www.obdii.com/background.html
[9] ELM327 datasheet, http://elmelectronics.com/DSheets/ELM327DS.pdf
[10] Engine codes, http://www.engine –codes.com