Application of OBDII Technology in the IoT Conception

J. Rogowski
Institute of Information Technology
Lodz University of Technology
Lodz, Poland
jan.rogowski@p.lodz.pl

Застосування Технології OBDII в Концепції IoT

Я. Роговський
Інститут інформаційних технологій
Технологічний університет м. Лодзь
Лодзь, Польща
jan.rogowski@uni.lodz.pl

Abstract— **This paper describes a system built on the basis of the Internet of Things concept enabling the monitoring of vehicle location and operation parameters. The system reads the vehicle operation parameters and its location, transmits this data from the vehicle to the server using a secure connection, displays vehicle operating parameters in the user interface in a real time and creates reports using software operating in a distributed environment. The deployment of the system allows for complete monitoring of the work of professional drivers in a large transport company or globally.**

Анотація— **У цій статті представлена система, побудована на основі концепції Інтернету речей, яка дозволяє контролювати місцезнаходження та параметри роботи транспортного засобу. Система читає параметри роботи транспортного засобу та його місцезнаходження, передає ці дані з транспортного засобу на сервер за допомогою безпечного з'єднання, відображає робочі параметри транспортного засобу в користувальницькому інтерфейсі в режимі реального часу та створює звіти за допомогою програмного забезпечення, що працює в розподіленому середовищі. Розгортання системи дозволить здійснювати повний моніторинг роботи професійних водіїв у великій транспортній компанії або у всьому світі.**

Keywords— Internet of Things; OBD II; ELM327; complete vehicle monitoring

Ключові слова— Інтернет речей; OBD II; ELM327; повний моніторинг автомобіля

I. Introduction

The Internet of Things concept involves the combination of devices equipped with sensors that allow one to interact with the physical world in a common network. Such connected items are referred to as intelligent. They can communicate with each other, surroundings and other devices. The idea also includes mutual communication between a man and a machine and between two machines. Some of the benefits of the Internet of Things include increased situational awareness, decision analytics controlled by sensors, immediate control and reaction. The Internet of Things is developing in many areas, including, among others: intelligent buildings, vehicles, cities, industry, environmental monitoring, threats and health. It can be classified into three categories: detection, processing and communication. Detection requires the use of sensors and includes, for example, the measurement of atmospheric pressure, temperature or speed. Data collected in this way can be processed by processors or microcontrollers. The devices themselves are connected using technologies such as OBD (On-Board Diagnostics), Wi-Fi, Bluetooth, GPS or RFID.

The work presents a system that allows one to register any number of vehicles and monitor the work parameters and location of these vehicles in real time using a distributed environment.

System acquires vehicle operating parameters using OBDII. This data via the ELM327 adapter connected to the DLC port is transferred using Bluetooth technology to the device operating on the Android platform. The location of the vehicle is obtained using a positioning module of the Android device. Then the device transfers the collected data to the server module that operates in a distributed environment. The server module allows one to monitor the operating parameters and the location of any number of vehicles registered in the system.

For a system, a collection of monitored vehicles is a collection of things, so the system is inherently associated with the concept of the Internet of Things. Presented system can be integrated with other solutions to create a larger intelligent transport system.

II. On-Board Diagnostic level 2

OBD in the automotive industry is the definition of vehicle self-diagnosis possibilities. It allows access to data and status of some vehicle systems. The available number of diagnostic data is constantly increasing, modern OBD implementations offer standardized digital communication by providing vehicle data over time and a set of error codes that allow one to identify potential problems with a vehicle.

OBDII (On-Board Diagnostic level 2) is a unified on-board diagnostic system complied with ISO and SEA standards. Using the diagnostic device one can connect and read the necessary data from any vehicle that is compliant with this standard regardless of the brand or car model. In addition, the OBDII system defines a set of diagnostic procedures that allow for early detection of a fault, which in turn may affect the emissions of toxic compounds. Initially, OBDII was created for the purpose of monitoring emissions exhaust. Currently, vehicle manufacturers use it to monitor components not affecting the exhaust. The OBDII system is a mandatory standard in vehicles: sold after January 1, 1996 in the USA and sold after January 1, 2001 in the European Union.

ECU (Engine Control Unit) can mean one or more modules that have a task to monitor and control vehicle functions. ECU parameter tuning can allow to change the levels of engine performance and its economy. The most popular ECU types are:

- Engine control module, ECM, controls the engine, including things like ignition, fuel mixture and idling speed,
- Transmission control module, TCM, manages things like fluid temperature, clutch position and speed,
- Electronic brake control module, EBCM, the module reads data and controls the wheel lock prevention system during braking (ABS, Anti-Lock Braking System),
- Body control module, BCM, the module controls body parts, such as electric windows and mirrors, heated seats.

The monitor in the context of the OBD system means the diagnostic procedure that is carried out using equipment and software. The purpose is checking the correct operation of the vehicle system or component. An additional task of monitors is to store the test results and notifications about detecting fault. These types of diagnostic procedures are mainly focused on the power and emission systems.

A single monitor oversees the work of one element. In connection with this, the number of the monitors in vehicles is very diverse. It depends mainly on the vehicle engine and on emission control system. Monitors are divided into: continuous and conditional. Continuous monitors are diagnostic procedures that control components while driving, and their testing can be started and disabled without affecting operation of other monitors. Conditional monitors are used in elements and subassemblies in which the identification of the damage requires longer observation, what translates into a longer reaction time. This is because conditional monitors use data provided from other elements, which in turn are supervised by other monitors.

The detailed conditions for activating a given monitor depend on many factors such as thermal and dynamic condition of the engine. The vehicle manufacturer is responsible for the proper working conditions of the monitor.

The OBDII system allows to distinguish two types of faults:

- Type A. These are mainly errors related to exhaust emissions. With this type of fault, an error code and a frozen frame are remembered and a MIL indicator light is lighted.
- Type B. As in the case of type A errors, an error code and a frozen frame are remembered, but only after two occurrences of the malfunction during a single driving cycle. In a case when the fault does not occur again, the error code may be removed.

One of the required OBDII functions is to remember the errors associated with faults that affect the exhaust composition and signaling this using indicator Check-Engine.

The DTC (Diagnostic Trouble Code) error code is standardized fault code consisting of one letter and four digits. If a fault is detected by the motor controller, it is remembered together with so-called frozen frame. This is done by saving parameters from some sensors that are in the car. Thanks to this treatment it is possible to precisely determine the conditions in which the fault occurred. These data can be overwritten by a higher priority error. Data that can be found in such a frame are, for example, engine speed, vehicle speed, engine load or fuel pressure.

The Data Link Connector (DLC) is a 16-pin diagnostic connector through which all information from the vehicle system covered by the OBDII standard is available. In order to read the information, a scan tool is needed, which is attached to the DLC connector. Using DLC it is possible to read such information as current engine parameters and error codes.

The ELM327 adapter - a microcontroller manufactured by ELM Electronics that can be used to scan the car's operation [1]. It is one of the most popular diagnostic interfaces that is able to connect to the majority of car engines on the CAN bus. This scanner only needs to be connected to the DLC diagnostic connector and it should be paired via bluetooth with the selected device with the Android system installed. The scanner does not require a battery, because the power supply is from the OBD2 connector of the vehicle.

III. Obtaining location information using mobile devices

Mobile devices enable obtaining locations in multiple ways using the following systems: GPS system, A-GPS system and via wireless Wi-Fi network.

The most popular and the most accurate solution is the GPS system, or rather GPS-NAVSTAR (Global Positioning System - Navigation Signal Timing and Ranging). It is a satellite navigation system whose task is to provide the user with precise information about his location and to facilitate navigation around the site. It consists of three main segments, namely the space segment, the ground segment and the user segment. The most important segment are satellites orbiting the Earth on a medium orbit, because thanks to them, the user gets information about his location. Accuracy of the determined location is estimated at about 1-3 m. In addition, the use of this system is completely free. The disadvantages of this approach in the case of mobile devices are relatively high battery consumption and a relatively long waiting time for accurate location information [2].

Another way to get information about the location is the A-GPS system. It is a solution supporting the standard method of GPS, which consists in retrieving information from the mobile phone network operator about the location of the nearest satellite and transferring them to the GPS module in the phone, which allows to reduce battery consumption and speed up the acquisition of user position. This solution may be particularly useful in places where there are a lot of tall buildings, because in such conditions propagation of the radio signal is difficult. The disadvantage of this approach is dependence on the network operator to which the user belongs.

The last solution is the determination of the location using wireless networks. Interestingly, to get the location using this solution, user does not need to connect to any network, because the location is calculated using nearby networks and their signal strength. Thanks to this information, it is possible to determine the approximate location. The biggest disadvantage is the need to have several networks nearby and the accuracy of the location, with error limit of 200 m.

IV. SYSTEM PROJECT

The system was designed in a client-server architecture. The server application should consist of two modules: the user interface (presentation layer) and business logic layer responsible for data processing. The data should be stored in a database that is independent of the server application. The presentation layer is implemented as a Single Page Application. In addition, the server application must provide the API (Application Programming Interface) compatible with the REST architecture.

The client application is designed to read measurements from the OBDII interface together with the current location of the vehicle on the globe and send this data via the Internet to the server application. Using API on the server side allows for the different implementations of the client application, ensuring a high level of flexibility of the solution.

A. Server application

The functional requirements of the server application are identified as follows.

- Only registered user can posess the access to the full functionality of the application.
- The user can create, modify and delete things, constituting the representation of a physical vehicle.
- The user can create, modify and delete drivers. The driver has a list of vehicles assigned to him and routes traveled.
- The user can define, modify and delete values for individual things alarms.
- The user can create, delete and modify the reports based on the collected data for single routes or for a selected historical period of measurements.
- The user can view the current measurements from the selected thing.
- The user can view the generated reports.
- Things can send measurements to the application.

The non-functional requirements of the server application are identified as follows.

- Incoming data from things should update the user interface state without the need of manual refresh.
- The user interface of the application should be responcible.

The users of the server application are *Authorized user*, an entity that has access to all application functions and a *Thing*, that is a vehicle with a client application installed.

B. Client application

The functional requirements of the client application are identified as follows.

- Acquiring vehicle self-diagnosis data at cyclical intervals.
- Acquiring the current position using the GPS module.
- Sending collected data to a server application.

The non-functional requirements of the client application are identified as follows.

- Operation on a device that supports the Bluetooth standard and operates under control of Android, version at least 4.0.
- Ability to act as a service without the need to keep the device screen on.

The users of the client application are *User*, a person who has a device operating under the control of the Android system with installed application and internet access.

V. IMPLEMENTATION DETAILS

Java programming language, Android Studio and Android Software Development Kit [3] were used to implement the client application.

AngularJS was used to implement the server application. AngularJS is a JavaScript-based programming framework created for building internet applications. It was created based on the idea of using declarative programming for building user interface and connecting individual components, and using imperative programming to create business logic of the application. It allows to extend the functionality offered by the HTML language for dynamic content presentation using two-way data binding, which ensures automatic view and model synchronization. AngularJS implements the MVC pattern to separate the presentation layer, data and application logic [4].

In order to ensure the secure transmission of information, all communication between clients and server are carried out using the HTTPS protocol [5]. This protects against man in the middle attacks, however, due to the fact that the service receiving messages from customers is publicly available, the additional security should be implemented, whose task is to ensure that the received message was actually sent by the customer with the given identifier. For this purpose, the password mechanism is used. Each message that is sent must have a secret key that acts as a password. Secret key for a given thing is created when the thing is created or modified. In order to use it later, it is stored in a database. In a case a secret key sent does not match the one assigned to a specific thing, the *403 Forbidden* error code is returned in response. If the message received does not contain any secret key value, the error code *400 Bad Request* is returned.

One of the requirements that were put on the system was checking the received messages for defined rules indicating the occurrence of an alarm value. If an alarm value is detected, information about such an occurrence is stored in a database.

Another requirement for the server application was to generate reports on historical database. Considering the potentially very large collections of those gathered data, the processing of which would be necessary to generate a report, Apache Spark distributed computing engine was used in conjunction with MapReduce.

MapReduce is a style of computing that has been implemented in several systems. It enables to manage many large-scale computations in a way that is tolerant of hardware faults. All is needed to write are two functions, called Map and Reduce, while the system manages the parallel execution, coordination of tasks that execute Map or Reduce, and also deals with the possibility that one of these tasks will fail to execute [6]. MapReduce consists of two phases: mapping and reduction. Mapping is the first calculations phase, it consists of executing a Map function that accepts one of the data pairs as an argument. The input must be in the form of a list of key and value pairs. After completing the mapping phase, the data is sorted, grouped in relation to the keys and passed to the Reduce function, whose task is to combine the received data and prepare the final value. The result of the reduction is the final result of the calculation.

Apache Spark is an open source general computing engine used for processing and analysis of large data sets. The solution is written in Scala programming language operating on the basis of a Java Virtual Machine. Spark allows to disperse data on the computer cluster and performs calculations in parallel using Resilient Distributed Datasets as a data structure. It operates based on the master/slave architecture, one main coordinator and many distributed processes that process the data [7].

Apache Cassandra was used as a database that stores received messages that ensured high performance IO operations. Apache Cassandra is a scalable, non-relational database offering high availability, linear scaling and data distribution between multiple data centers. Cassandra is a general-purpose database that can be successfully used for many different applications, it works particularly well in applications, where quick storage of large amounts of incoming data is required [8].

During the operation of the server application the new tasks are added to the fixed-volume thread pool. Using this solution the resources consumed on the machine on which the server application works are significantly limited in the case of a large number of tasks. The report execution task transfers the calculation to the Apache Spark driver. The query is parameterized, thanks to that it is protected against SQL injection attacks. It is also necessary to transfer the implementation of the Map and Reduce functions. These functions define calculations to be performed on the Apache Spark cluster. The reducing function is designed to calculate the minimum, maximum and average values for each parameter passed for the entire defined period for historical data. The calculations are passed to the Apache Spark driver whose job is to distribute the entire work to be done among all available executors. Data needed to execute the report are downloaded automatically from the database, without the need to engage the server application. Thanks to this, the necessary physical resources of the machine are reduced and efficiency is increased.

Conclusion

The system built on the basis of the Internet of Things concept is present. A system enables the monitoring of the vehicle location and operation parameters.

The number of vehicles whose parameters can be tracked by the system is not fundamentally limited. The system uses the Apache Spark distributed computing engine and the MapReduce algorithm to process large amounts of incoming data. For this reason system can be used for complete monitoring of the work of professional drivers in a large transport company or globally.

References

[1] ELM327 product web page. [Online], Available: https://www.elmelectronics.com/ic/elm327/ [Accessed: Apr. 10, 2018].

[2] GPS technical documentation. [Online], Available: http://www.gps.gov/technical/ps/ [Accessed: Apr. 10, 2018].

[3] Android Studio web page. [Online], Available:https://developer.android.com/studio/index.html [Accessed: Apr. 10, 2018].

[4] AngularJS developer guide. [Online], Available: https://docs.angularjs.org/guide [Accessed: Apr. 10, 2018].

[5] HTTPS protocol specification. [Online], Available: https://tools.ietf.org/html/rfc2818 [Accessed: Apr. 10, 2018].

[6] J. Leskovec, A. Rajaraman, and J. Ullman, *Mining of Massive Datasets*. Cambridge: Cambridge University Press, 2014.

[7] Apache Spark product web page. [Online], Available: https://spark.apache.org/ [Accessed: Apr. 10, 2018].

[8] Apache Cassandra product web page. [Online], Available: http://cassandra.apache.org/ [Accessed: Apr. 10, 2018].