A Real-World Implementation of IoT Automobiles

Milković, Hrvoje; Ivković, Nikola; Konecki, Mario

A Real-World Implementation of IoT Automobiles

Hrvoje Milković, Nikola Ivković, and Mario Konecki

***Abstract*—Modern automobiles have many built-in sensors and electronic control units responsible for increasing safety, optimizing performance and improving occupant comfort. Along with this well-established uses of computing devices inside automobiles, there are two main research directions in further computerization of automobiles. One direction is aimed at creating vehicles that communicate with road infrastructure and other vehicles in ad hoc manner forming vehicular ad hoc network (VANET). The other research direction is expected to produce self-driving automobiles that can drive safely on the safe road together with regular automobiles. The research presented in this paper is focused on making regular automobiles connected to the Internet and thus becoming a part of the Internet of Things (IoT), with some motivating applications under consideration. In this paper a developed general architecture that enables automobiles to communicate with user applications by the help of an axillary computational infrastructure is presented and elaborated. A prototype based on the proposed architecture has been developed by putting embedded system inside an automobile. The developed system is supported by a suitable software and infrastructure that have also been developed as a part of the research. Driving tests with the prototype on real roads confirmed that proposed approach is feasible.**

***Index Terms*—Automobile, embedded system, Internet of Things, distributed system.**

I. Introduction

Inside modern automobiles there are many embedded systems known as electronic control units (ECUs). They are responsible for controlling and monitoring various elements and subsystems of the automobile. Even low-end models often use more than 10 different microcontrollers, while mid-range automobile might have around 50 microcontrollers and high-end automobiles often have more than 100. Many of these ECUs need data from sensors that are connected to other ECUs.

A communication between ECUs is achieved by using Controller Area Network (CAN) that is present in almost any modern automobile [1]. Although CAN buses use a form of random media access control mechanism, they can guarantee that the messages will be delivered within specified time [2].

Along with CAN, some additional automotive communication busses like FlexRay, Local Interconnect Network (LIN) and Media Oriented Systems Transport (MOST) are implemented, filling differed engineering and economical demounts. FlexRay [3] allows higher speed communication for safe-critical applications and uses deterministic TDMA to fulfill real-time requirements. LIN bus [4] is used for connecting simple sensors and actuators, and MOST [5] is intended for infotainment applications with audio and video data.

The communication of external system with buses and ECUs inside an automobile is possible through On-Board Diagnostics II (OBD-II). On-Board Diagnostics II [6], [7] is standard and communication protocol required by law legislative in many countries including EU, USA, etc. By using the connector provided by ODB-II it is possible to connect not only external computer for diagnostic purposes inside automobile repair shop but also a custom-build embedded system.

With the advance of the Internet of Things (IoT) many hardware components became readily available and with affordable prices. Mobile network access is widely available and improvements with new 5G technology are expected [8], [9]. Therefore it is promising to upgrade standard automobiles so they can become the part of the Internet of Things. By using the mobile link between the automobile and the rest of the Internet the automobiles can communicate with remote servers, desktop workstations, laptops, smartphones, and even with other automobiles.

There are many motivating examples of applications that can be used in such situations.

It is possible to collect various statistical data from sensors and ECUs build in the automobile, often referred as telemetry, to improve monitoring, planning and managing of a single automobile or of an entire fleet of automobiles. Based on acquired data it is posable to observe and analyze average fuel consumption, driving style, dynamics of automobile movements through a geographical area, etc.

Remotely checking if doors or windows are left open would be practical for forgetful drivers. Automatic emergency call in the case of an accident could save human lives. Recordings of automobile positions and behavior could help in resolving traffic accidents.

Remote troubleshooting on road could solve some failures that might otherwise require arrival of the technician on site or transportation of the automobile to a mechanical shop.

Applications for infotainment like automatic synchronization of favorite songs between an automobile and other devices are also possible.

There are some examples of previous works were researchers have used OBD-II for reading automobiles data or they have created applications for which automobile data where simulated. Teng et al. have implemented an Android application that directly connects to OBD-II and graphically

Manuscript received October 15, 2016; revised December 30, 2016.

Hrvoje Milković is with the Kraken Systems, Ulica hrvatskih branitelja 3, 10090 Zagreb, Croatia (e-mail: hmilkovi@gmail.com).

Nikola Ivković is with the Department of Computing and Technology on the Faculty of Organization and Informatics, University of Zagreb, 42000 Varaždin, Croatia (e-mail: nikola.ivkovic@foi.hr).

Mario Konecki is with the Department of Theoretical and Applied Foundations of Information Sciences on the Faculty of Organization and Informatics, University of Zagreb, 42000 Varaždin, Croatia (e-mail: mario.konecki@foi.hr).

displays automobile data as a virtual instrument [10]. Jhou et. al. used a simulator to test an application for sending data from automobile to a computer cloud by using 3.5G wireless communication [11]. Ćurguz et al. used a simulator that generates driving events and implemented an Android application that sends an e-mail message to the parents of a young driver in the case that speed limit is violated on a section of the road [12].

There is a considerable ongoing research of Vehicle to Vehicle (V2V) communication where vehicles on road communicate with each other in an ad hoc manner forming what is known as Vehicular Ad Hoc Network (VANET). This approach requires development of specific physical and data link layer protocols that can allow automobiles that are passing by each other for a very short period of time (e.g. driving in opposite direction on the highway) to exchange messages in the environment where moving obstacles are usual appearance [13], [14]. This kind of communication is mainly applicable in urban and densely inhabited areas with enough automobiles to form VANET. Security and especially trust are particularly challenging for VANETs [15].

Research in the area of self-driving and computer-assisted automobiles has produced some competition within the industry. Companies like Tesla, Google, and Uber are experimenting with prototypes that have different approaches and capabilities [16].

The research presented in this paper is focused on plain old automobiles that are currently in use on roads and on how to update them with simple and affordable embedded systems in order for them to become connected to the Internet. In this paper design requirements for such applications are analyzed based on motivating examples presented in this section. A general architecture that can suitably facilitate these applications is also proposed. Finally, before conclusions, a real-world prototype that implements a subset of possible applications is presented.

II. Application Requirements

Vehicles are dynamic things that move through space and thus have variable Internet connectivity, ranging from possibly high throughput and excellent signal quality to slow or low quality links towards area without Internet connectivity. Proposed applications have to adapt to all of these conditions to perform in a best possible way.

Inside of automobile the embedded system collects data of interest and transfers them to the user applications that are normally running at remote locations.

Every second automobile as a "thing" on the Internet generates a large amount of data from internal sources and external sensors. Because of a limited memory capacity of embedded system inside the automobile, and also because of reliability, availability and safety reasons data have to be transferred to external storage.

For some applications it would be sufficient to record only aggregate statistical data, but for others like analyzing events in the case of traffic accidents the complete data should be stored. Some applications, like optimization of dynamic pickup and delivery by a fleet of vehicles, require communication with real-time properties.

While communication with the Internet is unavailable it is possible to save a smaller amount of data on the automobile embedded system and send it to an external storage when the connectivity is restored.

Establishing direct communication between the user application and the embedded system in the manner of Peer-to-Peer (P2P) paradigm would require dealing with various challenges. Both user application and embedded system would normally have private IP addresses, hidden behind Network Address Translation (NAT) mechanism, and publicly available addresses that are dynamically allocated by Internet Service Providers (ISPs). This makes initial communication establishing hard to accomplish without axillary meeting server [17]. Another problem with P2P approach would be availability. User applications are not always turned on or connected to the Internet and embedded systems have variable Internet connectivity with gaps without connectivity. Also, the embedded systems and sometimes devices for user applications have limited processing and memory capabilities.

Because of issues regarding P2P organization, the proposed distributed systems have to be composed of at least three principle parts: automobile embedded system, auxiliary Background Computational Infrastructure (BCI) and user application. The simplified system architecture is depicted in Fig. 1.

Many automobiles, i.e. embedded systems can share common BCI, and on the other side, many different user applications can use the same BCI.

Automobiles that share common BCI might be part of one vehicle fleet or originate from different fleets and individual owners.

Although most applications require data flow from automobile embedded system over BCI to user application, some applications, like synchronizing songs or fixing malfunctioning automobile remotely, require the opposite direction of data too.

Automatic emergency call in the case of an automobile accident, detected by crash sensors, would notify BCI, giving available data about the position, timing, etc. Simultaneously, automobile embedded system and BCI would try to reach an emergency center. Both attempts are aimed at improving robustness of the system regarding that the Internet communication might break and automobile embedded system might fail any time after the crash is detected.

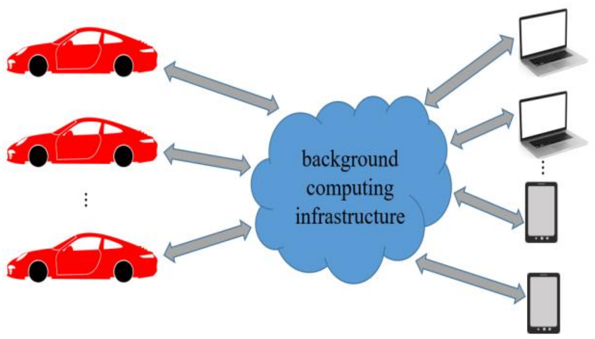


Fig. 1. Principle parts of the system architecture.

The safety of proposed system requires special approach, particularly in the case of data incoming to automobile embedded system. Also the architecture has to be highly modular enabling only applications that are selected for the particular vehicle.

To accommodate various application cases and to meet high safety and privacy requirements the overall architecture has to be flexible and modular, allowing end users to use only parts of the system they find necessary or desirable.

III. System Architecture

The principle parts of system architecture, presented in the previous section and depicted in Fig. 1, internally consist of different modules to accomplish different subtasks whose detailed architecture are depicted in Fig. 2 and Fig. 3.

A. *The Automobile Embedded System*

The regular automobile can be transformed into a "thing" that is a part of the Internet of Things by incorporating automobile embedded system. The embedded system is a single board computer or a system build from one or many microcontrollers that is able to communicate with automobile internal sensors, ECUs and busses (most notably the CAN bus). This communication is possible through OBD-II interface. The embedded system typically only reads data from the automobile, but it is also possible to control the automobile by sending messages. Since controlling automobile is a sensitive operation, if this possibility is allowed by the embedded system then exceptional security treatment is obligatory. The internal structure of the automobile embedded system is shown in Fig. 2.

Vehicle to embedded system communication module needs to interpret messages received by the automobile, filter them and prepare them in a suitable form before giving them to the mobile communication subsystem. This module also needs to send messages or exchange whole conversation of messages with the vehicle if the automobile controlling is enabled.

To upgrade automobile capabilities it is possible to attach additional external sensors to the embedded system if the suitable sensor is not part of internal automobile equipment.

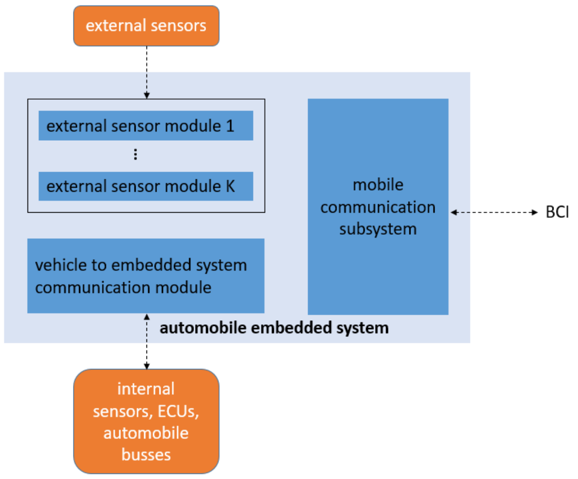

external sensors
external sensor module 1
⋮
external sensor module K
mobile communication subsystem
BCI
vehicle to embedded system communication module
automobile embedded system
internal sensors, ECUs, automobile busses

Fig. 2. Automobile embedded system.

For example, global positioning system (GPS) sensor is of vital importance for enabling automobile tracking or locating and it is useful for dynamic routes planning. The crash sensors are needed, together with GPS sensor, for automatic emergency calls, and can be useful in forensic analysis of traffic accidents. Each external sensor requires software module that can take data from the sensor using sensor specific protocols and prepare data into a form suitable for the rest of the system. These data are sent through mobile communication subsystem to external storage located in BCI.

The mobile communication subsystem is responsible for establishing communication between the automobile embedded system and BCI. This module requires scheduling mechanisms with priorities and policies how to handle data. Depending on the available data rate and quality of communication channel this subsystem needs to decide which messages to send, which one to queue, and which one to discard.

Messages that are queued because of the poor communication channel can be sent later when communication conditions improve or need to be discarded when queuing capacity is overfilled or data become outdated. In the case of a single board computer with relatively large memory it is possible to temporary store data for quite long time without network connectivity. Then it is of vital importance to decide in which order these history data and fresh data from automobile will be scheduled for sending. For safety and reliability reasons mobile communication subsystem is composed of outgoing data module and incoming data module. The direction of data refers to the flow of application data, while both modules can exchange control messages like acknowledgements in both directions.

B. *Background Computational Infrastructure*

The purpose of the background computational infrastructure is to facilitate information exchange between automobile embedded system and user application, to increase reliability and robustness, to provide storage for a large amount of data, and to assist in simple realization of user application on versatile computational devices. The internal structure of BCI is shown in Fig. 3.

The communication module inside background computational infrastructure is responsible for communicating with the automobile embedded system and preparing data to be stored in the time series database. The communication module also needs priority scheduling and traffic policy.

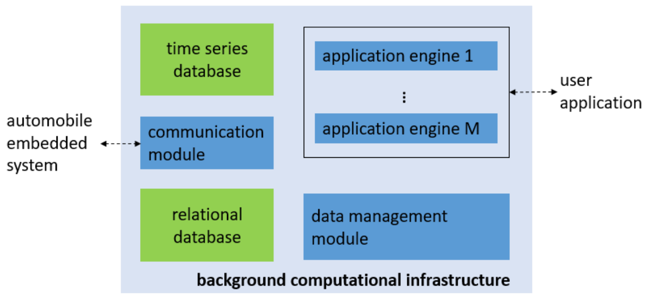


Fig. 3. Background Computational Infrastructure (BCI).

In the case of infotainment data, e.g. sending multimedia files with a song to the embedded system, the communication module needs to take into consideration available throughput and negotiate communication parameters with the mobile communication subsystem that reside inside automobile embedded system.

Data coming from sensors do not have complex relational structure. These data are incoming fast and have simple fields, with time of creation and automobile identification as always present attributes.

Examples of automobile data include vehicle geolocation, vehicle speed, engine rotational speed, diagnostic trouble codes, etc. Storing only aggregate data is not possible since some application cases require complete data. To enable scalability, time series database component of BCI is generally implemented with multiple servers and load balancer that allow a simple view of multiple machines and storages as logically one database.

Data related to general information about automobiles, their owners, user application profiles and settings, data for planning, and aggregate statistical history data are saved in the relational database.

Considering that data from many sensors of many automobiles that are running for a long time could generate an enormous amount of data, although possibly abundant storage capacity of time series database must be guarded against exaction. Data Management Module (DMM) is responsible for removing old data from time series when appropriate. Before deleting data DMM processes these data and produces aggregated statistical data that are saved in the relational database.

The set of application engines inside BCI is intended to implement particular application cases (where they act as server side of application) together with user application that is running on the end user device (as a client side). These application engines make use of data from time series database, relation database and services of the module for communication with automobiles.

Using web application paradigm enables simple devices to use application. Web application choice removes the need for separate application development for different platforms. This also makes easier for end users to start using application since web browsers are present on most devices of interest and there is no need for installing a standalone client application. For some application cases, like locking/unlocking doors and turning lights on/off, standalone user application with only essential elements might be better alternative than web application. In that case the application engine inside BCI is a server that interacts with the user application.

C. *Safety, Security and Privacy Aspects*

Whenever automobile controlling, tracking and monitoring is in action it is vitally important to maintain safety, security, and privacy.

By introducing embedded system inside an automobile it is of vital importance to guard against an adversary that might interfere with the automobile, but also against unintended errors and program bugs.

Modularity of automobile embedded system makes easier to control security and safety. The embedded system in the particular automobile uses only modules that are necessary. For example, if controlling automobile is not used then incoming communication module is not present in the system. If such module is in use, then only one transport protocol port is enabled and all other communication protocols and ports are disabled by default.

Because of modularity and isolation, malfunction of one module, e.g. an external sensor for GPS, does not cause entire automobile embedded system to malfunction.

Messages that come from outside cannot be forwarded to the rest of the automobile embedded system. For controlling automobile only a predefined set of commands are issued by incoming communication module to the vehicle to embedded system communication module. For the most critical parts of the automobile embedded system prescribed safety properties can be verified by using formal method techniques.

Communication between the automobile embedded system and BCI have to be encrypted and both sides have to be authenticated. The mean to achieve this is transport layer security (TLS) with certificates for both sides. To preserve privacy in a case of security breach forward secrecy mechanizes should be chosen. Communication between BCI and users application needs similar measures to preserve security and privacy.

The users of data stored in BCI can require privacy so that owners and administrators of BCI and the other users cannot see their content. This is achievable by using symmetrical encryption to protect data. The administrators of BCI infrastructure need to ensure availability, robustness and security by the standard set of technical precautions and measures.

IV. Prototype Implementation

For the purpose of testing we have built a prototype with few years old vehicle Ford Fiesta 1.4 TDCI (50 kW) model JD3. The embedded system is based on a single board computer Beaglebone Black, extended with external modules. For the BCI, the servers running Linux operating system have been used. The user applications have been built in a form of web applications allowing the user to track the vehicle movement on the map in real time and to analyze telemetric data sampled in real test drives.

Further details regarding implementation of the prototype are presented in following subsections of this paper.

A. *The Implementation of the Embedded System*

The tested Ford Fiesta has available OBD-II interface that uses CAN protocol with 11bit message identifiers and data rate of 500 kb/s. The automobile embedded system used in the Ford Fiesta is a single board computer Beaglebone Black with 1 GHz, 2000 MIPS ARM processor, 512 MB DD3L 6606 Hz, of RAM, 4GB on-board flash memory and 65 GPIO. As an operating system Ubuntu for embedded systems with a customized kernel has been installed.

Communication between the automobile and the embedded system has been established by OBD-II interpreter ELM327 that is implemented in the form of OBD-II to USB adapter.

Mobile modem in a form of external 3G dongle Huawei e3131 was attached to Beaglebone Black to allow data exchange with BCI.

Since selected automobile does not have internal GPS sensor, an external GPS module with -165 dBm sensitivity, 10 Hz updates and 66 channels was connected to Beaglebone

Black. The GPS module uses NMEA 0183 protocol.

External sensor module and vehicle to embedded system communication module has been implemented with Python programming language because of its suitability for prototyping. Each model was written as a standalone program in according to proposed architecture plans, thus if one module fails the other can continue to work.

GPS sensor module uses underlying GPSD daemon service that watches GPS sensor on POSIX level via UART port so Python script can just pull off the data from it. Messages of the vehicle to embedded system communication module ELM327 must be parsed, which requires dealing with implementation peculiarities. E.g. for requesting data about frequency on engine rotation RPM (Rotation per Minute) the AT command 01 0C is sent. As the response to this message another message like 41 0C 1A F8 would be received, where last two bytes is RPM scaled by 4.

To deal with connectivity losses a bash script was implemented to automatically reconnect automobile embedded system with BCI.

The communication with BCI was implemented by Message Queuing Telemetry Transport (MQTT) protocol, invented by IBM company and later standardized by ISO/IEC [18].

MQTT is a lightweight messaging protocol for small sensors and mobile devices, optimized for high-latency or unreliable networks. MQTT supports publisher-subscriber messaging pattern and therefore promotes scalability. MQTT does not force any message format and in proposed implementation JSON was used. Furthermore, it implements client and broker side caching that is needed to handle disconnections from the Internet when the vehicle moves through tunnels or otherwise becomes unreachable.

B. The Implementation of BCI

To solve scalability issues, VerneMQ implementation of MQTT broker was selected which supports clustering, traffic control, authorization, and authentication.

As a scalable implementation of time series database a third party database InfluxDB was installed in BCI. InfluxDB supports messages with format of time, value and multiple tags thus all messages needed to be prepared in that format. For the implementation of JSON parser, Node.js was selected as appropriate because it has highly optimized JSON parser with good performance, originating from underneath C and C++ implementations of their internals.

As an application engine inside BCI a web application was implemented with Python web application framework Django. It uses relation database PostreSQL to store and retrieve general data about users and their vehicles information. To present vehicle telemetry data, Django application also reads from InfluxDB computing statistics at database level regarding the engine data and to render vehicle GPS data to client devices on the map. In this way proposed prototype of the user application can show the history of the vehicle movement. To demonstrate the real-time capabilities of the system, web sockets to MQTT has been employed, which is natively supported by VerneMQ. Web socket transmits published GPS data in 500 ms intervals of real-time vehicle geological location and that data is rendered on the map as real-time tracking

C. Security Measures

In our prototype logging has been used in order to analyze the system behavior and to identify potential problems. The logging was used on each side of the system at process level with Linux daemon tool named Supervisor. Supervisor has its own system log and permits configuring for each process parameters like automatic restarts, log location, log file, etc.

In order to select necessary security measures for the prototype Threat risk modeling technique has been used. For both, BCI and automotive embedded system various security measures have been employed. Publicly reachable ports of BCI are made available through load balancer HAproxy. In order to increase process separation Linux Containers (LXC) technology was used. LXC container has its own firewall and is communicating with HAproxy via VPN connection that is implemented by OpenVPN with certificate level authentication.

The automobile embedded system was specially configured to disable all non-used ports. The public IP addresses were dynamically allocated and the private IP addresses were hidden behind NAT, thus minimalizing the chance of an unauthorized access from the Internet.

Communication level security is implemented in MQTT protocol in a form of certificate authorization, authentication and TLS encryption. MQTT is further secured on MQTT broker as each vehicle has its own user account to authenticate and each user has its own topic for which the user can publish and subscribe to data. Each topic is identified by vehicles VIN number. Presentation layer known as Django application is secured with Nginx web server reverse proxy that uses TLS version 1.2 certificate. Django application layer was secured with cross origin tokens, basic authentication and automatic IP banning the denial-of-service (DoS) attempts.

D. Scalability

Vertical scaling is trivial and not most cost effective so system parts that all support horizontal scaling as cluster of servers on every level of the system have been used.

Therefore, VerneMQ implements configuration how many parallel connections can be opened in server node, and how many messages can be published per client.

InfluxDB implements clustering and data sharding on multiple hosts where PostgreSQL implements only replication and sharding data.

All webservers have configured static content cache and Django web application has its own database cache implemented as Redis in-memory key value database.

To distribute requests from client in-Vehicle services and end user client devices, a load balancer HAproxy was used.

V. Prototype Testing

Before a real world testing of proposed prototype, the modules of BCI were installed and necessary parameter setup was performed. Required accounts in BCI were created. One account is used for the mobile communication module of the Ford Fiesta. The other account is used by the user who is allowed to access the data regarding Ford Fiesta from relational and time series databases. The data about the automobile were filled in relational database.

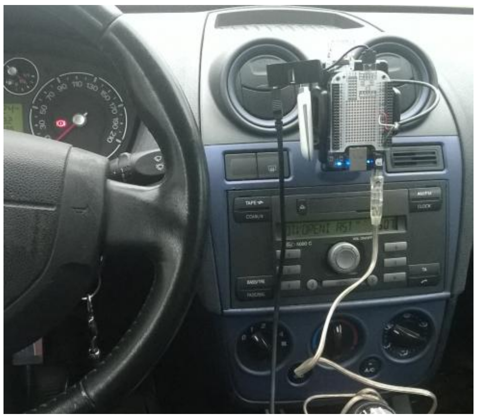
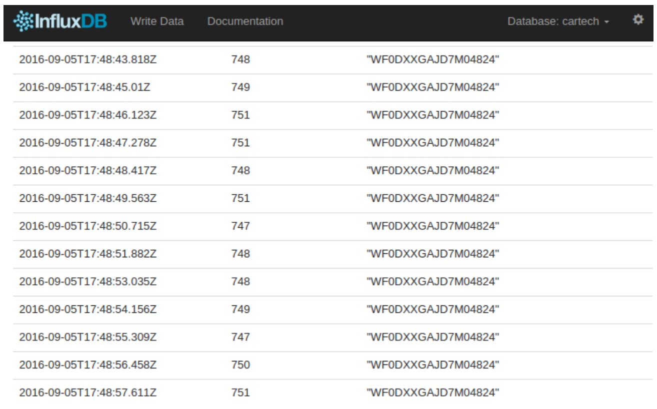


Fig. 4. The automobile embedded system with attached GPS module and mobile Internet module, connected to the automobile OBD-II interface.

The automobile embedded system was inserted in the Ford Fiesta and connected to OBD-II. The photography of the embedded system inside the test vehicle is shown in Fig. 4. After powering up the automobile embedded system, Linux boots up and the implemented scripts and programs automatically start available modules.



2016-09-05T17:48:43.818Z	748	"WF0DXXGAJD7M04824"
2016-09-05T17:48:45.01Z	749	"WF0DXXGAJD7M04824"
2016-09-05T17:48:46.123Z	751	"WF0DXXGAJD7M04824"
2016-09-05T17:48:47.278Z	751	"WF0DXXGAJD7M04824"
2016-09-05T17:48:48.417Z	748	"WF0DXXGAJD7M04824"
2016-09-05T17:48:49.563Z	751	"WF0DXXGAJD7M04824"
2016-09-05T17:48:50.715Z	747	"WF0DXXGAJD7M04824"
2016-09-05T17:48:51.882Z	748	"WF0DXXGAJD7M04824"
2016-09-05T17:48:53.035Z	748	"WF0DXXGAJD7M04824"
2016-09-05T17:48:54.156Z	749	"WF0DXXGAJD7M04824"
2016-09-05T17:48:55.309Z	747	"WF0DXXGAJD7M04824"
2016-09-05T17:48:56.458Z	750	"WF0DXXGAJD7M04824"
2016-09-05T17:48:57.611Z	751	"WF0DXXGAJD7M04824"

Fig. 5. A sample of data from time series database.

In the first phase of testing the messages between OBD-II port, vehicle to embedded system module, and MQTT broker inside mobile communication subsystem were recorded. The content and timing of messages were analyzed. An example of message recording is shown in Fig. 6. It can be observed that MQTT publish messages are following immediately after OBD-II messages: Vehicle speed, Car RPM, and Car MAF (Mass Air Flow) which indicates the mass of air that is entering an internal combustion engine.

In the second phase of testing a filling of time series database was performed. With the help of web application a sample of data is presented in Fig. 5. These records were obtained when automobile was in the neutral gear without pressuring accelerator pedal and contain time when data sample was created, RPM value, and vehicle identification number (VIN).

The third phase of testing was performed by driving the automobile on Croatian roads with the real-time monitoring and afterword by historical data analysis by using the implemented web applications. Fig. 7 shows the parameters of driving that are remotely monitored by the developed user application. The screenshot from the user application that shows tracking of the vehicle in a test drive is shown in Fig. 8.

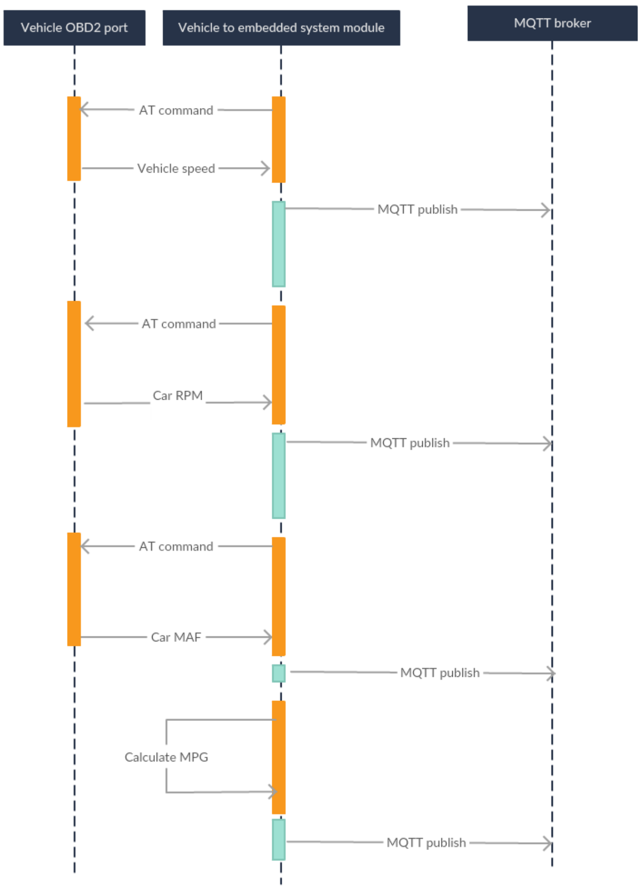


Fig. 6. Recorded messages between components inside the automobile embedded system.

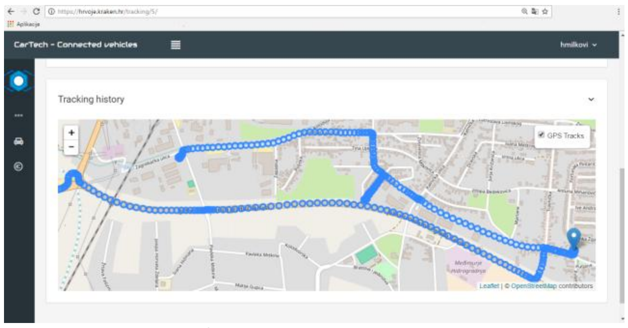

https://hrvoje.kraken.hr/tracking/5/
CarTech - Connected vehicles
hmilkovi
Tracking history
GPS Tracks
Leaflet | © OpenStreetMap contributors

Fig. 7. Display of automobile movements on one test drive.

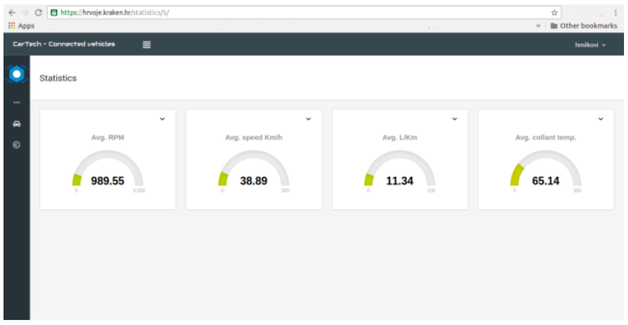


Fig. 8. Remote monitoring of statistical data about automobile on a test drive inside web browser on the user device.

VI. Conclusion

The aim of conducted project was to research how to upgrade regular automobiles and make them connected to the Internet. In this paper different application scenarios have been identified and based on requirement analyses proposed general system architecture has been presented. The system architecture consists of principle parts of the automobile embedded system, the Background Computational Infrastructure and the user applications. The modular organization of architecture subsystems allows flexibility and promotes security. For the testing purposes we have developed the prototype and its functionality has been tested in real-world applications. The results of tests confirmed that proposed approach is feasible.

REFERENCES

[1] Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signaling, International Organization for Standardization, ISO 11898-1:2003.

[2] N. Ivković, D. Kresic, K.-S. Hielscher, and R. German, "Verifying worst case delays in controller area network," *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, pp. 91-105, March 19-21, 2012.

[3] *FlexRay Communications System Protocol Specification*, Ver. 2.1, FLEXRAY Consortium, 2005.

[4] *LIN Specification Package Revision 2.2A*, LIN Consortium, December 31, 2010.

[5] *MOST Media Oriented Systems Transport*, Rev 2.4, MOST Cooperation, 2005.

[6] R. Cox, *Introduction to On-board Diagnostics II (OBDII),* 1st edition, Cengage Learning, Cengage Learning US, 2005.

[7] A. Santini, *OBD-II: Functions, Monitors and Diagnostic Techniques*, 1st edition, Cengage Learning, 2010.

[8] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617-1655, 2016.

[9] X. Ge, H. Cheng, G. Mao, Y. Yang, and S. Tu, "Vehicular communications for 5G cooperative small-cell networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 7882-7894, October, 2016.

[10] H.-F. Teng, M.-J. Wang, and C.-M. Lin, "An Implementation of Android-Based Mobile Virtual Instrument for Telematics Applications," in *Proc. the 2011 Second International Conference on Innovations in Bio-inspired Computing and Applications (IBICA '11)*, Shenzhan, Guangdong, December 16-18, 2011, pp. 306-308.

[11] J.-S. Jhou, S.-H. Chen, W.-D. Tsay, and M.-C. Lai, "The implementation of OBD-II vehicle diagnosis system integrated with cloud computation technology," in *Proc. the 2013 Second International Conference on Robot, Vision and Signal Processing (RVSP '13),* Kitakyushu, Japan, December 10-12, 2013, pp. 9-12

[12] A. Ćurguz, T. Maruna, B. Kovačević, and M. Z. Bjelica, "Android application as parental control service in car," in *Proc. the 2015 23rd Telecommunications Forum Telfor (TELFOR),* Belgrade, Serbia, November, 2015, pp. 934-937

[13] A. Ghosh, V. V. Paranthaman, G. Mapp, O. Gemikonakli, and J. Loo, "Enabling seamless V2I communications: Toward developing cooperative automotive applications in VANET systems," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 80-86, December 2015.

[14] H. Wang *et al*., "VANET modeling and clustering design under practical trafic, channel and mobility conditions," *IEEE Transactions on Communications*, vol. 63, no. 3, pp. 870-881, March, 2015.

[15] Q. Li, A. Malip, K. M. Martin, S.-L. Ng, and Z. Jie, "A reputation-based announcement scheme for VANETs," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 9, November, 2012.

[16] B. Paden, M. Čap, S. Z. Yong, D. Yersho, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, March, 2016

[17] N. Ivković, I. Magdalenić, and L. Milić, "An ad-hoc smartphone-to-smartphone live multimedia streaming application with real-time constraints," *Journal of Advances in Computer Networks*, vol. 4, no. 1, pp. 6-12, March, 2016.

[18] *Information technology – Message Queuing Telemetry Transport (MQTT)*, v3.1.1, ISO/IEC 20922, 2016.

Hrvoje Milković received the bachelor degree in information systems at Faculty of Organization and Informatics, University of Zagreb. He is a full-time software engineer at company Kraken Systems and first time part-time student at Faculty of Organization and Informatics, University of Zagreb, study program Databases and Knowledge Bases. His interests include design and implementation of complex embedded and backend systems.

Nikola Ivković received the MS degree in computer engineering and the PhD degree in computer science from the Faculty of Electrical Engineering and Computing, University of Zagreb. He is a member of the research and teaching staff at the Department of Information Technologies and Computing at the Faculty of Organization and Informatics, University of Zagreb. His research interests include computer networks, formal methods, computational intelligence and optimization, operating systems, and parallel programming.

Mario Konecki is an assistant professor at the Faculty of Organization and Informatics in Varaždin, Croatia. During his former scientific work, he has published over 60 scientific papers and has been actively involved in 8 scientific projects. He is also active in professional work in the field of programming, design and education. His main scientific interests are: intelligent systems, development of programming languages, education in the area of programming, design of user interfaces and web technologies. His main research projects are: "Determining the possibility of including visually impaired in the activities of graphical user interfaces design" and "New methods of teaching programming with an emphasis on teaching visually impaired students".