Archives of
Transport System
Telematics

Volume 11

Issue 1

February 2018

Mobile Driver Assistance System Based on Data from the Diagnostic Port of Vehicle

S. IWAN[a], K. MAŁECKI[b], Ł. ZABOROWSKI[b], M. NÜRNBERG[a]
[a] MARITIME UNIVERSITY OF SZCZECIN, Faculty of Economics and Engineering of Transport, Pobożnego 11, 70-515 Szczecin, Poland
[b] WEST POMERANIAN UNIVERSITY OF TECHNOLOGY, Faculty of Computer Science, Piastów 17, 70-310 Szczecin, Poland
EMAIL: s.iwan@am.szczecin.pl

ABSTRACT
The article presents one application from the ADAS (Advanced Driver Assistance Systems) group of systems, which enables the reading of parameters from the module connected to the diagnostic port in the vehicle. The developed application enables better control of engine operation and supports the driver in the field of, among others indication of currently running gear and suggestion of switching on the higher or lower gear depending on the engine parameters read. The suggestion of changing gears is shown graphically and sonically. The application is designed for mobile devices working under the control of Android operating system.

KEYWORDS: Advanced Driver Assistance Systems (ADAS), diagnostic port, mobile application

1. Introduction

Every year, more and more vehicles are equipped with driver support systems based on numerous sensors, control modules and cameras. In a modern vehicle the driver is supported by various information about the driving style, recognized traffic signs and vehicle data. Many models also offer support in closing the door, opening and closing the tailgate, parking and many other activities, while increasing the safety of driving.

However, according to Samar [1], the average age of passenger cars traveling on Polish roads is around 13 years. The average age for cars in Europe is according to ACEA [2] about 10 years. Many of these vehicles are not equipped with on-board computers at all, and if they already have them, the information displayed is very limited. The reason is certainly that the number of sensors and modules installed in is much smaller than in modern cars. However, the vast majority of these vehicles have a diagnostic connector allowing to read parameters from the engine management module.

Considering the increasing popularity and functionality of smartphones that accompany us every day, the aim of this work is to develop a mobile application analyzing vehicle parameters based on data from a module connected to the diagnostic port which is located in the vehicle. The parameters available from the engine module, so far invisible to the driver, will be analyzed. The application will present data in a way that is understandable for the vehicle user in accordance with the selected driving mode. In addition, the application will visually and audibly signal to the driver the need to change gear to a higher or lower depending on the registered parameters. It will also enable the measurement of acceleration time to set speeds and current and average fuel consumption.

2. Related work

Advanced driver assistance systems (ADAS) help drivers react to a situation on the road or in a vehicle and thus improve driving safety [3].

The most popular systems of this type are road sign recognition systems [4-8], commonly implemented in modern vehicles. There are also known systems and methods of vehicle recognition [9-11], detection of brake lights [12-15], systems of surface condition control [16, 17], applications downloading and processing data from ITS systems [18-20] and communication systems between vehicle and road infrastructure that allow for optimal switching of traffic lights [21-25].

3. Communication with the vehicle

The appearance of the first on-board diagnostic (OBD) systems [26] was closely related to the continuous monitoring of vehicle failure and exhaust emissions. Since the introduction of the solution, the information available has been very diverse and limited in scope. At the end of the 1980s, in the United States, thanks to the California Air Resources Board (CARB) [27], it was decided that all vehicles must be equipped with basic OBD capabilities. These decisions and an attempt to continuously reduce and control the exhaust emissions caused that in 1994 the OBD-II specification was issued.

This standard has become so popular that over time it also appeared in Europe. The existing European equivalent is denoted by the abbreviation EOBD (European On Board Diagnostic) [26] and it has the same technical specification as OBD-II. Since 2000, it has been mandatory to install it in vehicles with a gasoline engine and since 2003 in vehicles with a diesel engine.

3.1. The OBD-II system

OBD-II system appearing on the market significantly increased the ability of vehicles to self-diagnosis and facilitated a communication with the vehicle using external devices. The most important assumptions of the system are: control of all devices affecting the final emission of the vehicle, protection of the exhaust gas catalytic converter against damage, optical warning indications when the devices affecting the final emission from the vehicle exhibit functional faults and error memory.

The introduction of the system has brought new standards, which define, among others, the format of sent messages and available data transmission protocols used in the standardized Data Link Connector (DLC). Additionally, the related standards provide the On-board Diagnostics Parameter IDs (OBD-II PIDs) with possible monitoring parameters along with their detailed description. However, it is not an obligation for producers to implement all of them. They can add their own proprietary parameter identification numbers (PIDs) to standard items.

The specification of the diagnostic interface is specified in detail in SAE J1962 standard [28]. It contains information about the location in the cockpit and an exact construction indicating a 16-pin standard female socket with specifically spaced communication protocols lines (Fig. 1).

Vehicle manufacturers have several communication ports at their disposal, but in most cases only one is used. Each of them has a position defined in the norm, which means that some of the 16 pins are reserved. The remaining pins, not specified in the aforementioned standard, allow for authorized use by producers [29].

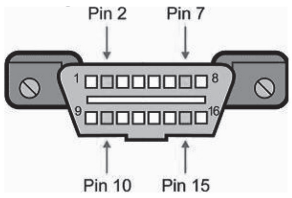


Fig. 1. Diagram of the diagnostic connector in the vehicle [28]

The on-board diagnostic system after making the connection provides 9 operating modes defined in the SAE J1979 standard [30]. Each of the possible modes is responsible for providing a different type of data (Table 1).

Table 1. Working modes of OBD-II [30] [own study]

Mode	Description
01	Current drive diagnostic data and system information
02	Frozen frame information (saved information at the time of failure)
03	Error codes
04	Erase diagnostic information
05	Request for oxygen sensor monitor test results
06	The results of discontinuous monitors tests
07	The results of continuous monitors tests
08	The control over the diagnostic system
09	Read information about the vehicle

Considering the scope and objectives of the article, the most important one to be discussed is the first mode, because it allows access to the data of the propulsion system in real time.

The defined list of identification numbers used for data exchange provides software developers with some important information that allows them to interpret the received values appropriately. The first of the available identification numbers specified by the "00" code allows obtaining information on the numbers that can be used. The answer returned by the OBD-II system does not have a defined formula for decoding the result, because it requires a different interpretation of the value obtained. The number of bytes returned by the system in the case of such a query is always 4. By decoding the received bytes to the hexadecimal form, we get a string that once again should be decoded. Each subsequent character of the hexadecimal value stored in binary form allows to determine whether the given identification number, starting from the first, is available for reading.

It is good practice to start diagnostics from checking the availability of identification numbers. This allows determining whether the app is correctly connected to the vehicle and do not send unnecessary commands to the system, thus increasing the expectations of others. For most of the other parameters included in the list, short formulas are defined that do not require such in-depth decoding. The exact number of returned bytes only makes it easier to properly decode values and substitute them for the formula. Clearly defined descriptions and values allow users to create their own prototype libraries to communicate with the system.

3.2. The ELM327 interface

There are several types of interfaces available to connect with the vehicle. However, the most popular is the specified ELM327 interface created by ELM Electronics [31]. It gained its popularity thanks to the ability to communicate with all protocols defined for the OBD-II system. The first available interfaces communicated with the software via the RS232 port, which was located on older computers. However, the constantly evolving technology and high demand led to the creation of three more versions: via USB port, Bluetooth module and WiFi. The number and type of possible configuration types allow for data exchange with virtually any device available on the market.

The big advantage of the ELM327 interface is the fact that in the absence of any interaction it can go into a low-power sleep mode. In addition, communication interfaces connected to the port and using the Bluetooth and WiFi module due to their ergonomic design do not disturb the driver in driving a car.

4. Driver support application

This application is designed for mobile devices working under the control of the Android operating system [32]. First time, this system was presented in October 2008. Thanks to the Google Inc. Android has become, over time, the most popular operating system for mobile devices, with a market share of 82%. The last release of Android is version 7 called Nougat, and Marshmallow and Lollipop are the most popular so far.

4.1. The project assumptions

To correctly design the application, functional requirements (Table 2) and non-functional requirements (Table 3) were initially defined.

Table 2. Functional requirements [own study]

No.	Description
FR1	Data exchange between vehicle and smartphone
FR2	Establishing a connection with the vehicle
FR3	Analysis of selected parameters
FR4	Display of parsed and received parameters on a user friendly interface, with the possibility of change
FR5	A possibility of turning off audible notifications for gearshifting
FR6	Saving vehicle profile to database
FR7	Establishing a re-connection at unexpected disconnection of devices

Table 3. Non-functional requirements [own study]

No.	Description
NFR1	Device running Android 4.2 or later
NFR2	Device equipped with Bluetooth module
NFR3	ELM327 interface connected to diagnostic port on the vehicle

For proper data exchange between the vehicle and the mobile device, the OBD-II Java API [33] library was used. The library also provides functionalities that allow for proper configuration of the connection with the ELM327 interface, thus enabling the establishment of appropriate forms of messages transferred from the interface to the application.

Unfortunately, some data downloaded from the diagnostic port are not legible for the user of the vehicle. An example of it is the parameter returning the value of the air mass flow sensor. However, this value together with other available values allow to obtain knowledge about the current fuel consumption in the vehicle and allow to calculate the average combustion [34]:

$$MPG = \frac{VSS * 7.107}{MAF} \quad \textbf{(1)}$$

where: VSS - the current vehicle speed obtained from the diagnostic port, MAF - the value of the air mass flow sensor from the diagnostic port. The calculated MPG value shows the amount of fuel burned in gallons per mile. In order to adjust the format to European requirements, the value of 235.21 is divided by the MPG, thanks to which we get a result in liters per 100 km.

The application also allows indicating the current gear while the vehicle is moving. The application uses two parameters for this: vehicle speed and engine speed. Then, in the process of calibration the application, a constant value for each gear is calculated independently. The ability of the gears' calibration allows using the application in many vehicles. This functionality also allows monitoring the time of moving in a given gear, as well as suggesting a change of gear to a higher or lower, at a specific engine speed. In the application, it was assumed to suggest a lower gear if the engine speed for a given gear drops below 1,400 rpm or 1,200 rpm for second gear. The suggestion of switching on the higher gear always takes place at the same level, 2,500 rpm. All parameters provided by the application are arranged on several screens, thus ensuring readability and the ability to change the interface depending on the driving mode.

Each suggestion of changing the gear beyond a graphic notification carries with it a sound signal, made using the ToneGenerator class included in Android [32]. This functionality has been implemented so that the driver does not have to pay much attention to the device while driving, but still responds appropriately to the delivered messages. However, it is possible to disable this notification in case it becomes too persistent for the user.

The application allows saving the profile of a given vehicle to the local database [35] in the phone's memory. A database contains one table that stores data of the VIN number of the vehicle, type of fuel, number of gears and calibration of individual gears. The main key of the table is the column containing the name of the vehicle profile entered by the user.

The last of the defined functional requirements is the possibility of automatic connection with the module plugged into the diagnostic port, in case the connection was interrupted without user intervention.

4.2. The application structure

The application structure is based on eight packages with 21 classes:

- Activity – contains one class, the main activity of the whole application,
- Database – there is a class responsible for exchanging data with the database,
- Fragments – a package containing all classes that provide particular views in the application,
- Interfaces – a package containing interfaces that ensure communication between fragments and the main Activity class,
- Model – here is a class containing the object of vehicle,
- OBD2 – the package responsible for establishing a connection and exchanging data with the vehicle,
- Services – there is a class responsible for the Bluetooth module,
- Utils – package of classes' auxiliary in the application operation.

The only class included in the Activity package is MainActivity. This class has an implemented SectionsPagerAdapter inheriting from FragmentPageAdapter responsible for loading individual fragments into subsequent application screens. In addition, the main task of the MainActivity class is to provide fragments, through the implemented BroadcastReceiver, with information received from the service class and processing diagnostic data.

The next package is Database. The DatabaseAdapter class is included in it and it is responsible for creating a database in case it does not exist and for opening or closing a connection to a database. This class also provides methods used to remove or add new vehicle profiles.

The next package is the largest one. Fragments included in it define each of the available screens in the application. There are 10 classes that describe the views in the application.

The project also includes Interfaces and Model packages. The first one contains the ComunicateFragment interface used to communicate between the Fragments extending it and the MainActivity class. The Model package includes the Vehicle class, in which, at the moment of establishing the connection, data on available identification numbers from the diagnostic port, type of fuel, constants calculated for each gear, as well as distance traveled and average fuel consumption are stored.

A few of the most important classes are in the OBD2 package. The package contains 4 classes: two of them (CustomCommandOBD and PidOBD) are not currently used, but they were created for further development of the project. The most important classes are TransferConfigOBD and TransferDataOBD. Both of them act on a separate thread at the time of the operation, thus causes that the thread of the main application is not overloaded. They are started from the level of the BluetoothService service class. The first one is responsible for correctly configuring the diagnostic interface connected to the vehicle and establishing a connection with it, while the second one is activated when the first connection is successful. TransferDataOBD is a class that is responsible for sending, receiving and analyzing received data.

Another of the most important elements is the Services package which contains the BluetoothService class. This class is responsible for initiating and supervising the connection to the diagnostic device. This class allows the application to react properly when it is minimized. The service class informs the application when the status of the Bluetooth module changes.

Utils is the last package. The MessageUtil and StaticValues classes included in it support the operation of the application. The first of these classes allows sending messages to the service class from anywhere in the application code. The second one defines constant values used during application operation.

4.3. The application's interface

The main screen of the developed application is shown in Fig. 2. The icons on the right side of the screen are the buttons corresponding to the following functions: profile editing, availability of Bluetooth module and connection with the module. The next program options (shown at the top of the main screen) are available on the next screens, visible by moving the current screen to the left (it is mobile application, so user can operate it using screen of device).

Editing the vehicle profile (Fig. 3) allows inserting or changing such data as: the fuel type of our vehicle, the number of available gears and the revolutions thresholds for each gear. In the right part of the screen there is a list of gears marked in green when the gear is set up and in red when the gear is not defined yet. From this screen user can also move to the vehicle's profile screen (Fig. 4) by clicking the button in the upper right corner which is managed by the SaveVehicleProfileFragment class.

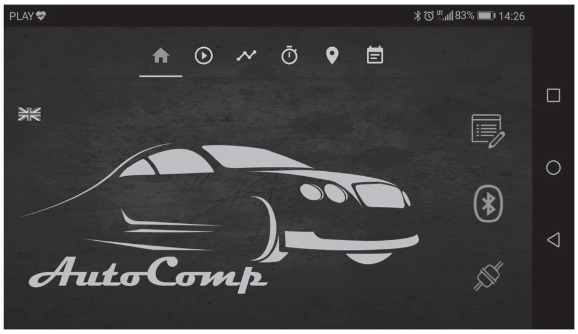
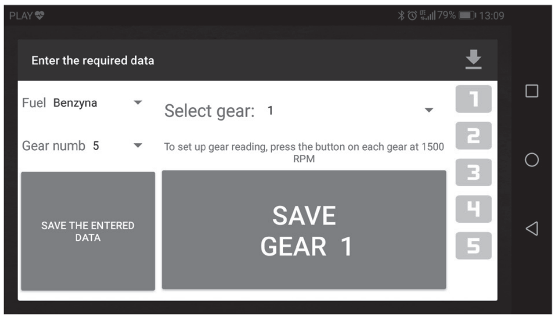


Fig. 2. The main window of developed application [own study]

Another option available on the main screen of the application is a possibility of informing the user about the status of the Bluetooth module together with the possibility of searching and establishing a connection with a new device. The screen that appears for this purpose contains a list of devices already paired and currently searched. The DeviceListFragment dialog deals with the management of available devices as well as the search. When the user selects the device, the dialog notifies the service class that automatically establishes the connection. This button has also an informative role. Through possible visible states it indicates the connection status - when it is not connected to the vehicle (yellow), establishes a connection (circular ProgressBar), and green when the connection has been established.


Enter the required data

Fuel Benzyna

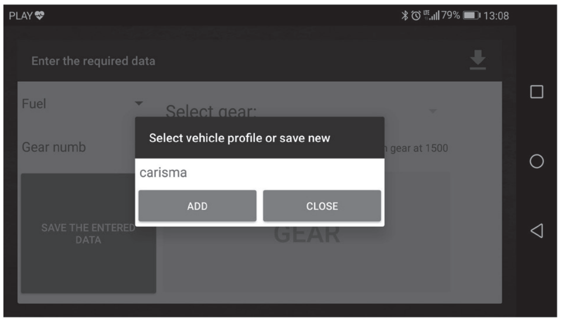
Gear numb 5

Select gear: 1

To set up gear reading, press the button on each gear at 1500 RPM

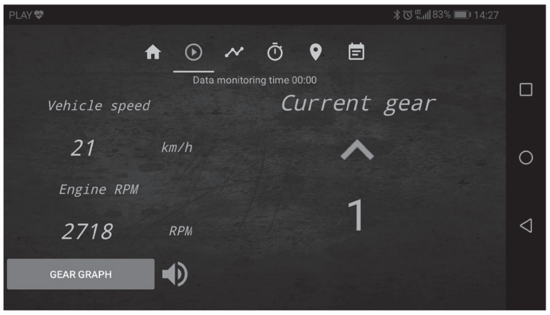
Fig. 3. The profile editing [own study]

The second available screen (by moving the screens to the left) is the view defined by the EcoLiveFragment class, which presents information about the current vehicle speed, the number of engine revolutions and the current gear and indications regarding its change (Fig. 5). At the bottom of the screen there is a green button for displaying the graphs of the percentage of gears used during the trip and the number of indications for the suggested gear (Fig. 6).


Select vehicle profile or save new

carisma

Fig. 4. Saving the profile [own study]


Data monitoring time 00:00

Vehicle speed 21 km/h

Engine RPM 2718 RPM

Current gear 1

Fig. 5. The second screen of the application [own study]

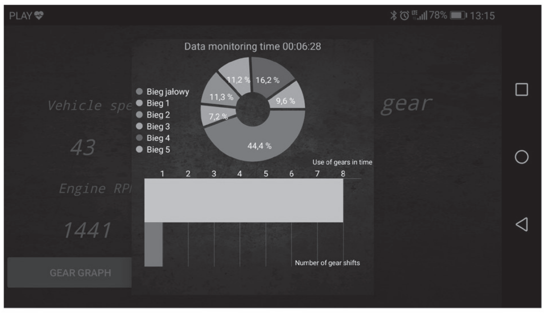

Data monitoring time 00:06:28

Fig. 6. The screen with graphs of gears used during the trip [own study]

The third screen of the application (Fig. 7) is implemented by the GraphLiveFragment class and enables the display of a line graph containing actual data on current, instantaneous combustion, average combustion and engine speed.

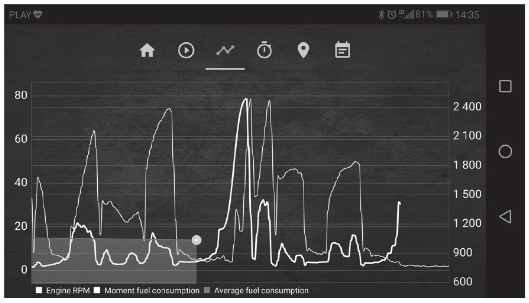
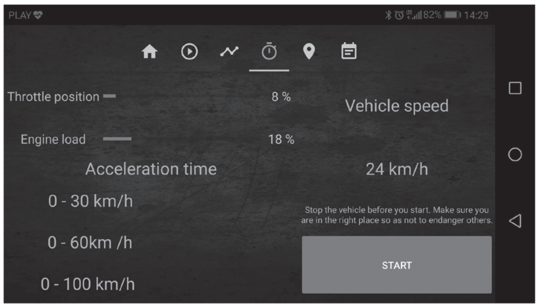


Fig. 7. The third screen of the application [own study]

Another available screen there is a view representing the SportLiveFragment class, displaying information about the current throttle position and engine load. At this point, the possibility of measuring the acceleration time on 3 sections of the road appears (Fig. 8).


Throttle position 8 %

Engine load 18 %

Vehicle speed 24 km/h

Acceleration time

0 - 30 km/h

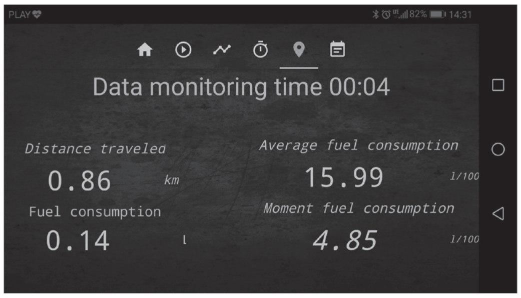
0 - 60km /h

0 - 100 km/h

Stop the vehicle before you start. Make sure you are in the right place so as not to endanger others.

Fig. 8. The screen with engine data [own study]

Numerical monitoring of the connection time, distance traveled, fuel consumption and average and instantaneous combustion of the vehicle is presented on the fifth screen of the application (Fig. 9).


Data monitoring time 00:04

Distance traveled 0.86 km

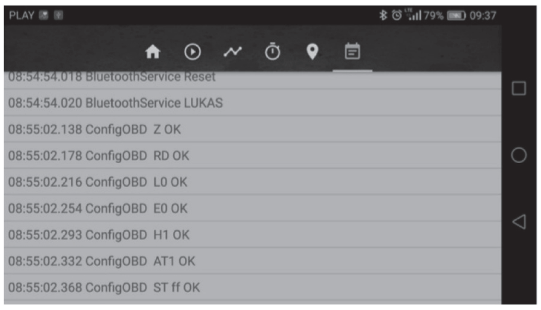
Average fuel consumption 15.99 l/100

Fuel consumption 0.14 l

Moment fuel consumption 4.85 l/100

Fig. 9. Trip data [own study]

All activities related to the operation of the application are recorded in the application and they are presented on the next screen (Fig. 10).



```
08:54:54.018 BluetoothService Reset
08:54:54.020 BluetoothService LUKAS
08:55:02.138 ConfigOBD  Z OK
08:55:02.178 ConfigOBD  RD OK
08:55:02.216 ConfigOBD  L0 OK
08:55:02.254 ConfigOBD  E0 OK
08:55:02.293 ConfigOBD  H1 OK
08:55:02.332 ConfigOBD  AT1 OK
08:55:02.368 ConfigOBD  ST ff OK
```

Fig. 10. The activity of the application [own study]

4.4. Testing of the application

Three stages of testing the developed application were performed: checking the correctness of the operation of defined functional requirements, checking the demand for system resources and verification of the application's usefulness.

The first stage of testing was based on the developed test scenarios:

- Scenario 1 – checking the correctness of establishing a connection with the ELM327 interface and the diagnostic port,
- Scenario 2 – checking the gear shift signaling,
- Scenario 3 – saving the vehicle profile to the database.

The application has been tested on three mobile devices: Huawei P9Lite with Android version 6.0, Samsung Galaxy J5 with version 6.0.1 and HTC One S with Android 4.2.

The first test scenario was carried out twice on each of the above mentioned devices. Of the six attempts to establish a connection, five went well and achieved the intended result in the scenario. One of the trials for the Huawei P9Lite was unsuccessful. The error did not result from the way the application or smartphone works. The only diagnostic problem was the diagnostic interface returning the wrong value for the query received. Another attempt was successful.

The second test scenario was successful on each of the tested mobile devices. The application running on various hardware configurations and system versions behaved correctly, respectively signaling graphically and audibly suggested gears.

The last test was also successful. The application correctly created a database and saved the vehicle profile for each tested smartphone.

The second stage of testing was to check the system's demand for system resources. Four properties were analyzed: RAM usage, CPU usage, network connection usage, and GPU utilization. At the time of data exchange, the application used the processor of the device within 5%, which is a low value, not slowing down the device. The use of RAM during application operation was within 31-33 MB. These values mean that the developed software can work in the background, without interfering with other enabled applications.

The last stage of testing was making the application available to a group of 15 users who determined the usability and quality of the application. Each participant of the study had the opportunity to use the application throughout the day, after which he answered the questionnaire with a few questions with a scale from 1 (very bad) to 5 (very good). Among the surveyed users were people with long-term driving experience and those who have recently obtained a driving license. There were women and men in the group. The results, being the average of all ratings, are shown in Table 4.

Table 4. Evaluation of the usability of the application by users [own study]

Question	Rating
How do you rate the refresh rate of data? (Speed response to change in driving value)	4.3
How do you rate the performance of the application while recording and analyzing data?	4.7
How do you rate ease of use in the app?	4.9
How do you rate the readability of information?	4.8
How do you assess the implemented display functionality and suggestion of gearshift?	4.6
How do you assess the implemented functionality of current and medium fuel combustion?	4.0
Overall app rating	4.5

5. Conclusion

A mobile application to support the driver of the vehicle, in which there is no on-board computer installed was developed. The application analyzes vehicle parameters based on data from the module connected to the diagnostic port. The "AutoComp" mobile application is designed for devices with Android system. It is characterized by a large simplicity of interface, intuitive operation and functionality supporting the driver.

The application correctly informs the user about currently running gear and suggests the driver change to a higher or lower one, depending on the engine parameters read. The application correctly calculates current and average fuel consumption in the vehicle, thus allowing greater control of driving economy.

The developed application is characterized by a modular structure, which allows for further development. It is possible, for example, to implement the functionality regarding the deletion of diagnostic errors.

Bibliography

[1] Car park in Poland based on the Central Statistical Office http://www.samar.pl/__/1001/1001.rep/26/.html?locale=pl_PL [in Polish], [date of ccess: 12.10.2017]
[2] Average Vehicle Age http://www.acea.be/statistics/article/average-vehicle-age [date of ccess: 12.10.2017]
[3] GERONIMO D., LOPEZ A.M., SAPPA A.D., GRAF T.: Survey of Pedestrian Detection for Advanced Driver Assistance Systems, IEEE Trans. on Pattern Analysis and Machine Intell, vol. 32, 2010, pp. 1239-1258
[4] LOY G., BARNES N.: Fast shape-based road sign detection for a driver assistance system, Proc. Intelligent Robots and Systems (IROS 2004), vol. 1, 2004, pp. 70-75
[5] MALDONADO-BASCON S., et al.: Road-sign detection and recognition based on support vector machines, IEEE Trans. on Intell. Transp. Syst., vol. 8, 2007, pp. 264-278
[6] LOPEZ L.D., FUENTES O.: Color-based road sign detection and tracking, Proc. Image Analysis and Recognition, Proceedings, vol. 4633, Springer-Verlag Berlin, 2007, pp. 1138-1147

[7] MOGELMOSE A., TRIVEDI M.M., MOESLUND T.B.: Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey, IEEE Trans. on Intell. Transp. Syst., vol. 13, pp. 1484-1497, 2012

[8] FORCZMAŃSKI P., MAŁECKI K.: Selected Aspects of Traffic Signs Recognition: Visual versus RFID Approach, in Mikulski J. (ed) Activities of Transport Telematics, Springer Verlag, Berlin Heidelberg, CCIS 395 (2013), pp. 268-274

[9] FREJLICHOWSKI D., et al.: Application of Cascades of Classifiers in the Vehicle Detection Scenario for the 'SM4Public' System. Proc. 16th Int. Conf. on Intelligent Data Engineering and Automated Learning (IDEAL 2015), LNCS vol. 9375, 2015, pp. 207-215

[10] FREJLICHOWSKI D., et al.: Applying Image Features and AdaBoost Classification for Vehicle Detection in the 'SM4Public' System. Proc. Int. Conf. Image Processing and Communications (IP&C 2015), Image Processing and Communications Challenges 7, AISC, vol. 389, 2016, pp. 81-88

[11] FORCZMANSKI P., NOWOSIELSKI A.: Deep Learning Approach to Detection of Preceding Vehicle in Advanced Driver Assistance, in Mikulski J. (ed) Challenge of Transport Telematics, Springer Verlag, Berlin Heidelberg, CCIS 640 (2016), pp. 293-304

[12] SU J., et al.: Design and research on condition monitoring system of brake light of cars, IEEE World Automation Congress, 2012

[13] ALPAR O.: Corona segmentation for nighttime brake light detection, IET Intelligent Transport Systems, vol.10(2), 2016, pp. 97-105

[14] WANG J.G., et al.: Appearance-based Brake-Lights recognition using deep learning and vehicle detection, IEEE Intelligent Vehicles Symp, 2016, pp. 815-820

[15] MAŁECKI K., WĄTRÓBSKI J.: Mobile System of Decision-Making on Road Threats, Procedia Computer Science, vol. 112 2017, pp. 1737-1746

[16] STANIEK M.: Stereo vision method application to road inspection, Baltic J. of Road and Bridge Eng., vol 12(1), 2017, pp. 38–47. doi:10.3846/bjrbe.2017.05

[17] STANIEK M.: Road Pavement Condition as a Determinant of Travelling Comfort, in: G. Sierpiński (Ed.), Intell. Transp. Syst. and Travel Behav., AISC, 2017, pp. 99–107. doi:10.1007/978-3-319-43991-4_9

[18] IWAN S., MAŁECKI K., STALMACH D.: Utilization of Mobile Applications for the Improvement of Traffic Management Systems, in Mikulski J. (ed) Telematics - Support for Transport, Springer Verlag, Berlin Heidelberg, CCIS 471, (2014), pp. 48-58

[19] OSKARBSKI J., ZAWISZA M., ŻARSKI K.: Automatic Incident Detection at Intersections with Use of Telematics, Transportation Research Procedia, vol. 14, 2016, pp. 3466-3475

[20] OSKARBSKI J., BIRR K., MISZEWSKI M., ŻARSKI K.: Estimating the average speed of public transport vehicles based on traffic control system data, In: Int. Conf. Models and Techn. for Intell. Transp. Syst. (MT-ITS 2015), 2015, pp. 287-293

[21] MAŁECKI K., PIETRUSZKA P.: Comparative Analysis of Chosen Adaptive Traffic Control Algorithms, In: Macioszek E., Sierpiński G. (eds) Recent Advances in Traffic Engineering for Transport Networks and Systems (TSTP 2017). Springer, Cham, LNNS, vol 21, 2018

[22] MAŁECKI K., PIETRUSZKA P., IWAN S.: Comparative Analysis of Selected Algorithms in the Process of Optimization of Traffic Lights, In: Nguyen N., Tojo S., Nguyen L., Trawiński B. (eds) Intelligent Information and Database Systems. ACIIDS 2017. LNCS, vol. 10192. Springer, Cham, 2017

[23] MAŁECKI K.: The Importance of Automatic Traffic Lights time Algorithms to Reduce the Negative Impact of Transport on the Urban Environment, Transportation Research Procedia, vol. 16, 2016, pp. 329-342

[24] CELIŃSKI I., SIERPIŃSKI G.: Traffic Signal Control System with Extended Logic in the Context of the Modal Split, IERI Procedia, Elsevier, vol. 4, 2013, pp. 148 – 154. http://dx.doi.org/10.1016/j.ieri.2013.11.022

[25] SIERPIŃSKI G., CELIŃSKI I., STANIEK M.: The Model of Modal Split Organisation in Wide Urban Areas using Contemporary Telematic Systems, Proc. of The 3rd Int. Conf. on Transp. Inf. and Safety, 2015, pp. 277-283. doi 10.1109/ICTIS.2015.7232125

[26] MYSZKOWSKI S.: Diagnostyka pokładowa standard OBD II/EOBD. Poradnik serwisowy, vol. 5, 2003 [in Polish]

[27] On-Board Diagnostic II (OBD II) Systems - Fact Sheet / FAQs https://www. arb.ca.gov/msprog/obdprog/obdfaq.htm. [date of access: 17.02.2017]

[28] ELCOCK A., KISTER T.: U.S. Patent Application No. 11/321, 562, 2005

[29] Ports in OBD II http://www.samochodowka.koszalin.pl/warsztaty/dzialy/bielicki/ obd/systemobd2-eobd.htm. [date of access: 17.07.2017]

[30] DUAN J., XIAO J., ZHANG M.: Framework of CANopen protocol for a hybrid electric vehicle, In Intelligent Vehicles Symposium, IEEE, 2007, pp. 906-911

[31] ELM327, O. B. D. to RS232 Interpreter. ELM Electronics, 2012

[32] COLLINS C., GALPIN M.D., KÄPPLER M.: Android w praktyce: najlepsze techniki programowania na Androida w zasięgu ręki, Helion [in Polish]

[33] OBD-II Java API https://github.com/pires/obd-java-api. [date of access: 17.01.2017]

[34] AVR-Based Fuel Consumption Gauge, Deriving Miles per Gallon http://www.lightner.net/lightner/bruce/Lightner-183.pdf. [date of access: 25.03.2017].

[35] Saving Data in SQL Databases https://developer.android.com/training/ basics/data-storage/databases.html. [date of access: 25.07.2017].