Journal of Engineering Science and Technology
Vol. 15, No. 6 (2020) 3757 - 3771

INTEGRATED CAR TELEMETRY SYSTEM BASED ON INTERNET OF THINGS: APPLICATION AND CHALLENGES

FAHMI FAHMI[1], ERWIN SUTANTO[2,*],
MUHAMMAD YAZID[3], MUHAMMAD AZIZ[4]

[1]Faculty of Engineering, Universitas Sumatera Utara, Jalan
Dr. T. Mansur No. 9, Medan 20155, Sumatera Utara, Indonesia
[2]Faculty of Science and Technology, Universitas Airlangga,
Kampus C Mulyorejo, Surabaya 60115, Jawa Timur, Indonesia
[3]Faculty of Intelligent Electrical and Informatics Technology, Institut Teknologi Sepuluh
Nopember, Kampus ITS Sukolilo, Surabaya 60111, Jawa Timur, Indonesia
[4]Institute of Industrial Science, The University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
*Corresponding Author: erwin_sutanto@fst.unair.ac.id; fahmimn@usu.ac.id

Abstract

The number of cars is increasing every day, and there is a need to monitor each of them digitally as well as remotely. It does not only collect the data, send them remotely, save them in the database, and display the results through the website or mobile apps, but also analyse the performance of the car, plan the maintenance schedule, and evaluate the drivers' behaviour. A car monitoring system has been developed in this study. The development was divided into three main parts, including an on-board hardware module, an android application (app), and web servers as the user interface. The hardware was used for collecting the data from the car and simultaneously sent it to the server. Then, the app was developed to display the possible analysis results for the user. Finally, the webserver with database system became a data center for data collection. The stored data consist of GPS location and engine operating conditions, such as engine load, coolant temperature, intake manifold pressure, revolution per minute, speed, forward timing, intake air, and throttle position. However, the main focus of this work is on the analysis of data transmission speed. The results show that more than 90% of the total 611 testing records were transmitted with no delay or delay less than 10 s. Furthermore, for the car condition test, eight engine operating conditions were summarized from two different cars, which each of them has 20 h of driving test. It was found that each car had its own characteristics.

Keywords: Car monitoring, Data transmission speed, On board diagnostic, Position tracking, Web communication.

1. Introduction

In order to implement Industry 4.0, the existence of digital infrastructures becomes the main supporting factor, in which the internet of things (IoT) is one of the essential components. It refers to the whole connected network of physical devices, including vehicles, electrical appliances, and other items. For example, sensors, actuators, and their controls can be integrally connected. The existing internet network can be used to collect and exchange the data.

As a transportation tool, a car is requested to be safer and more comfortable fulfilling the demand and standards [1]. Nowadays, the preference for private vehicles over public transportations, especially in developing countries for daily trips, is increasing. Conducting supervision or monitoring is essential to maintain the car performance and driving comfort [2]. However, recent technologies and methods still cannot facilitate all of them. Therefore, it is necessary to implement a new technology to improve and replace the existing technology and method with the adoption of IoT.

Battery electric vehicles (BEVs) [3] and existing advanced internal combustion engine vehicles (ICEVs), including both hybrid and plug-in hybrid, have become technological breakthroughs in our transportation system, which requires supports from many different aspects. Among them, the development of a car monitoring system is also urgently required as it is potential to simultaneously optimize the engine efficiency, durability, and also operational performance, especially during a long trip. Avdagic et al. [4] has studied the possibility to monitor and control the fuel consumption using the adaptive neuro-fuzzy inference system (ANFIS) [4], which can help the car owner in maintaining its consumption efficiency. In addition, Rajendran and Smith [5] have developed a monitoring app for the solar-powered unmanned aerial vehicle (SUAV). Overall, a car monitoring system might be one of the critical research areas for all of BEVs, and existing and advanced ICEVs. This work deals with the effort to realize the monitoring system through the developed system using wireless sensors based on an on-board diagnostic (OBD) kit for the existing ICEVs [6]. In addition, it also tries to effectively send the data regularly to a server through available digital infrastructure. Finally, at the server, those data are processed and analysed. The communication system becomes the basis for diagnosing the condition for both types of vehicles.

The system developed in this study is divided into three main parts. The first is the on-board hardware module, which collects the data from the ICEV, such as a gasoline car, and sends these values to the server. The second is an android application (app) that has been developed as the user interface to facilitate an analysis of the results obtained from the server and also as an auxiliary display, instead of using an Android-based vehicle diagnostic app to directly display the data read by the OBD [7]. The third is a web server with a database system that functions as storage for the data sent by the hardware module. The first objective of this study is to measure the data transmission speed. This can be done by using GPS tracking device and analysing the tracking speed as in the previous study which was carried out using an iPhone [8]. Moreover, this study also evaluates the possible application of these technologies and analyses the results toward the implementation. Based on this configuration, the system can be categorized as GPS and OBD integration (GOI) [9].

2. Material and Method

There are several steps before implementing the system, including hardware module design and development, server configuration, and mobile application development.

The hardware module design and development focus on how to read the diagnostic data from the car. The next step is the development of the monitoring app, which provides access to the server interactively. The designed application was temporarily provided only for the Android smartphone. The final step is to design the server, which was performed by configuring the web server and its database system.

2.1. Hardware module

The developed hardware module is based on an OBD. It is an automotive tool commonly used by mechanics in order to diagnose the car engine. It could provide the report of the car status digitally. In this study, the ELM327 OBD-II (ELM Electronic Inc.), as shown in Fig. 1(a), was used to translate the OBD interface on the most modern car vehicles. ELM327 is one of the families of OBD translators from ELM Electronics Inc., while other variants only implement a part of the OBD protocol. ELM327 has a low-level protocol and presents a simple interface that can be called via universal asynchronous receiver/transmitter (UART), such a hand-held diagnostic tool or computer program connected by USB, RS-232, Bluetooth, and Wi-Fi. Some functions can be performed by the OBD-II ELM327 scanner module, which makes it possible to read diagnostic problem codes, both generic and factory-specific, and display their meaning with more than 3,000 standard code definitions in the database [10]. The OBD module can also erase the trouble code and turn off the damage indicator light (DIL), better known as check engine light. It can also display current sensor data, including engine rotation, calculated load value, coolant temperature, fuel system status, vehicle speed, intake manifold pressure, inlet air temperature, airflow rate, absolute throttle position, fuel pressure, etc. However, only eight of these engine operating conditions are recorded in this study as evidence that the concept can be developed further for advanced implementation.

A microcontroller is used for combining those available devices [11]. It is a low-cost microcontroller system that already includes a low-power communication module, such as an integrated Wi-Fi and dual-mode Bluetooth. A simple microcontroller was developed using Arduino Mega 2560, as shown in Fig. 1(b), with two batteries and a standard power plug available in the car. For the communication module, a general packet radio service (GPRS) module of the global system for mobile (GSM) SIM 800L, as shown in Fig. 1c, was installed to make it possible for internet access through the 2G network infrastructure [12]. For communication modules, Bluetooth HC-05 and GSM SIM 800L were adopted. The HC-05 module is an easy-to-use Bluetooth serial port protocol (SPP) module, designed for transparent wireless serial connection settings. The Bluetooth HC-05 module can be used in a master or slave configurations. This Bluetooth module is fully compliant with Bluetooth V2.0 + enhanced data rate (EDR) 3 Mbps modulation complete with 2.4 GHz radio transceiver and 2.4 GHz baseband. It uses CSR Bluecore 04 - external single-chip Bluetooth system with CMOS technology and adaptive frequency hopping (AFH) feature.

Arduino Mega 2560 controls the telecommunication module, which is SIM800L GSM, for data transferring. This module is one of the most popular types of GSM/GPRS serial modules used for various remote control purposes. Currently, there are several types of breakout boards available globally, however, the most available one in Indonesian market is the mini version with a GSM type having micro SIM card. The design of the developed IoT-based car vehicle monitor system uses this 800L SIM.

Furthermore, in this module, data collection and processing were also designed locally in case there is a problem in the communication to the server. The integrated hardware for this purpose was carried out by using the microSD card, as shown in Fig. 1€. The card is possible for reading and writing (R/W) modes. The microcontroller was able to read and write data on a microSD memory card. This module can handle memory card operations such as creating, writing files, changing, or deleting data in the form of files or folders. Working at a voltage of 5 V with 80 mA current consumption, this module can be controlled using a microcontroller using SPI communication lines with a microSD memory capacity up to 32 GB.

(a) ELM OBDII. **(b) Arduino Mega.** **(c) SIM800L.** **(d) HC-05.** **(e) MicroSD.**

Fig. 1. Hardware components used in the car monitoring system.

2.2. Android app

The purpose of the development of this Android-based mobile app is to provide an extended interface of the web server. The data presented on the monitoring interface were obtained based on the reading of the hardware module, which has been discussed before. The app is able to show advanced analysis results from the server rather than just real-time data monitoring of the engine operating conditions. However, the developed app should not be limited with the purpose because there are many features that are available in the Android smartphone. For example, there is an assisted-global positioning system (A-GPS) in most phones. It could be used for position tracking. Thus, this app was tried to focus on this capability as a tracking device apart from the system function.

Overall, the usage of the system can be seen in Fig. 2. It is a use case diagram that shows all functions of the system in this work [13]. It shows that the system has at least two functions. The first is to record the engine diagnostic data by activating the hardware module. The second is to track the car position by utilizing the Android app. However, in this work, the app was utilized only for evaluating the possible data transmission speed. Hence, the usage analysis will be included as the further study.

The layout design of the application followed the existing research [14]. It was added with a position tracking feature as an additional of common indicators, such as the engine condition, tire pressure, and battery voltage. The software used to develop the application is an Android Studio. This is an integrated development environment (IDE), which has also become a trend, according to recommendations issued by

Google [15]. The data direction, as shown in Fig. 3 was only one direction to the server. Our communication method includes three different stages. The first stage is communication from the hardware module with the microcontroller module. The second one is transmission of all data from the app to the server, including the GPS locations from the mobile device. Finally, the server will be the final stage to save and analyse the data for the car conditions monitoring.

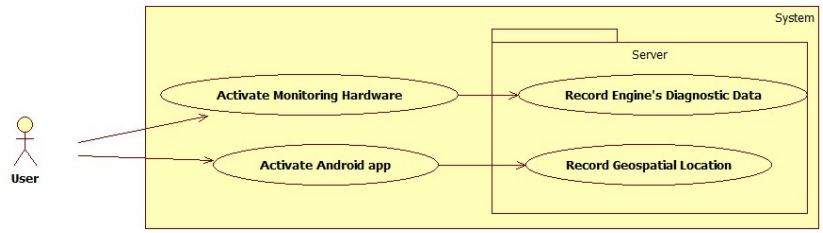


Fig. 2. Use case diagram.

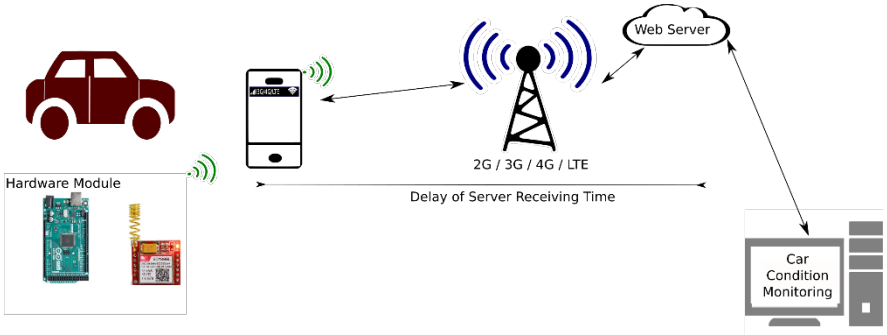


Fig. 3. Conceptual diagram of the whole developed system.

The location data can also be used for the calculation of data transmission speed. This is conducted by finding out the interval time within each transmitted data. These intervals can be calculated by having the difference between two records as they are sequentially inserted into the database (DB). Thus, the calculation of the interval can be presented in Eq. (1).

$$t_{inerval}[n] = t[n] - t[n-1] \tag{1}$$

where $t[n]$ is the time recorded when the n-th data was entered and $t[n\text{-}1]$ is the previous data. Hence, the time interval can be obtained. This interval value for nth-data, $t_{interval}[n]$, does not show directly the length of processing time for this HTTP protocol. The time here is also affected by the logic of the Android app program, which is waiting for the changes in geo-location data before sending it to the server. This means that the acquired interval is not always the same. It really depends on the road situation and the condition of the GPS signal, apart from the HTTP protocol time through the GSM network infrastructure. A traffic jam on the road causes a longer interval time compared with a situation where the car can run fast without any traffic. The GPS signal itself depends on GSM infrastructure, such as how far the phone position is located away from the antenna of the base transceiver station (BTS).

2.3. Webserver

In the implementation of the IoT-based car monitoring system, the server and database subsystem function as the central hub for data storage, analysis and distribution. Data from the on-board hardware module is received using HTTP transmission and stored in the database. This information can be accessed by user online using a web browser or smartphone application. An artificial intelligence based on the analysis of these data can be used to get useful information such as car usage patterns or car engine health conditions which can also be provided to the user via the web browser or smartphone application. In this subsystem, the Poisson distribution equation as shown in Eq. (2) can be used to find the occurrence probability of data measurement delay time.

$$P[X = t] = \frac{\lambda^t e^{-\lambda}}{t!} \tag{2}$$

where P is the probability of t as delay time, λ is the number of occurrences, and e is the Euler number. This server also consists of a web server composed of a data interface for the on-board hardware module from the user car, and a user interface for both gadget-based display (mobile device) and PC browser owned by the user. It is then could find a module with a computing capability for real-time intelligent data analysis once the transmission speed is confirmed.

3. Results and Discussion

The results obtained could be observed in each of the following sub-sections. They were the hardware module, web server, Android app, system testing result, and its challenges.

3.1. Hardware module

The prototype of the car monitoring system was built based on a design which has been proposed before, consisting of OBD-II scanner module, which was available on the market and an on-board controller board module. The on-board controller board module could control the OBD-II module via a Bluetooth close range wireless network to read data from sensors on the vehicle and send the readings through the GSM/GPRS internet network provided by cellular communication service providers. The sensors based on the car manufacture were the percentage of engine load, coolant temperature, intake manifold pressure, engine rotation, car speed, engine forward timing, air intake temperature, and throttle position. The on-board controller board module was composed of a battery and a battery control module as a power supply. Then, it also had an HC-05 module as a Bluetooth communication module. Furthermore, SIM800L GSM had been installed to provide internet communication via GSM/GPRS networks. Finally, the Arduino Mega 2560 microcontroller module was used as the controller of the overall system. The prototype, as shown in Fig. 4, was implanted in the cars to record the necessary data.

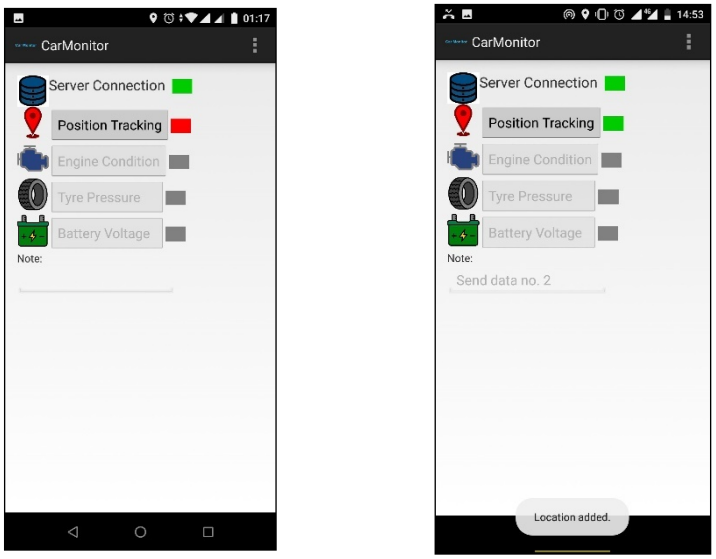
3.2. Android app

The first appearance of the Android app is shown in Fig. 5. This app has an indicator of connectivity status with the server. It also has an indicator and an activation button for GPS tracking. Finally, it also has indicators from the web server, such as

engine condition, tire pressure, and battery voltage. Once the tracking is activated, the indicator color changes to green. The display is shown in Fig. 5(b). Besides the indicator, a message will also pop up if there are any data locations added to the server. The message is shown at the bottom part of the display. The version of this app development is only limited to this function because the analysis part in the server is only up to the running test.

(a) OBD and hardware. **(b) Hardware placement.** **(c) Inside hardware box.**

Fig. 4. Implementation of the car monitoring system installed in the car.



(a) Before tracking. **(b) Tracking activated.**

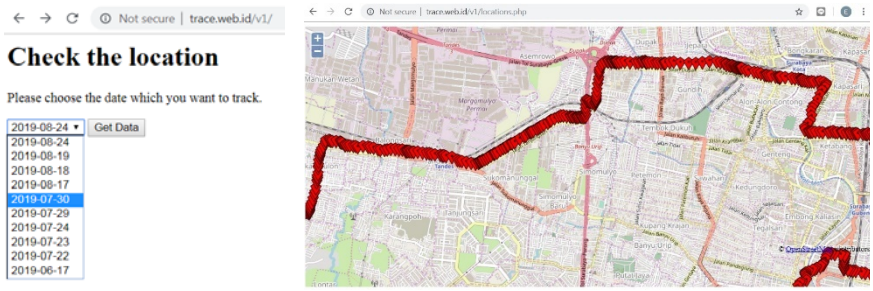
Fig. 5. Car tracking system.

3.3. Webserver

In developing the server for data analysis, a web server was developed, including a database, an interface that receives data from an on-board monitor and stores it in a database, as well as an interface for a web browser to read data from a database and display it in a graphical form that could be accessed by users. In order to store

data sent by an on-board monitor, a web server was prepared with a PostgreSQL as a database server. A user interface had been developed using a standard web browser to facilitate data monitoring.

Overall, this system could also be used for position tracking, as shown in Fig. 6(b). It shows the recorded positions with a sequence of red position marks during the driving test in Surabaya, Indonesia, on July 30, 2019. Here, in addition to location data, inserted time was also stored in the database. By using that time, the interval time could be calculated by using Eq. (1). That was by looking for the time difference between the records. It was merely by subtracting the time of the existing location data with the time of previous location data. The results from the overall distribution of the traveling test could be seen in Table 1.



(a) Search by date. **(b) Recorded positions.**

Fig. 6. Webserver display.

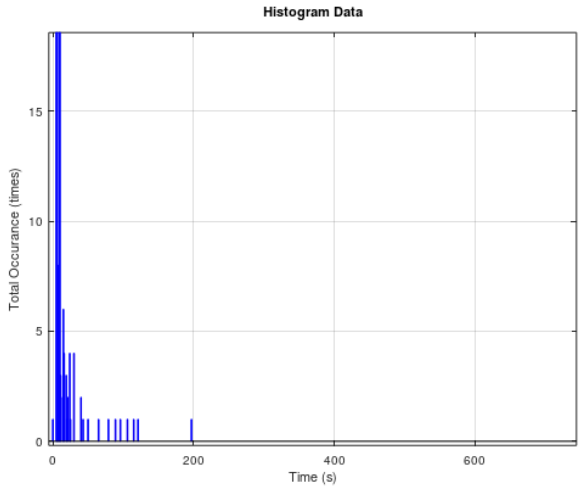
Table 1. Distribution of interval time.

Interval time	Count	Interval time	Count	Interval time	Count
0:00:05	285	0:00:19	3	0:01:19	1
0:00:06	83	0:00:20	1	0:01:29	1
0:00:07	1	0:00:21	2	0:01:36	1
0:00:08	8	0:00:24	4	0:01:46	1
0:00:09	69	0:00:25	1	0:01:55	1
0:00:10	121	0:00:30	4	0:02:01	1
0:00:11	3	0:00:40	2	0:03:17	1
0:00:14	2	0:00:43	1	2:12:54	1
0:00:15	6	0:00:50	1		
0:00:16	4	0:01:05	1		
Sub Total	582	Sub total	20	Sub total	8
		Total	610		

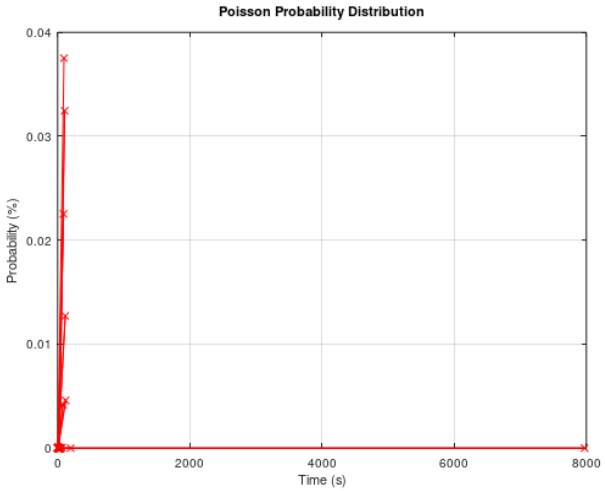
Table 1 was made from 611 dots, which results in 610 of interval times. From this table, it could be observed that the minimum and the most dominant interval was 5 s, with the total data of 285 dots. This can be assumed to be the fastest time in the process of recording the data in real-time. This can further be illustrated in the situation of vehicles that are moving fast, and there is no problem with the GPS signal. The second most common interval was 10 s, with the total number of 121 dots, less than half the first interval. This can be assumed to be a delay from the recording system. Furthermore, the interval is followed by two other intervals of 6 and 9 s. With a difference of 1 s from the two mentioned intervals, this can be assumed that the car was not in any traffic jams, but there is a gap about 5 s from the previous time. The total number of data with intervals less than 10 s was 558

dots, which is 91.48% of all data. Furthermore, the majority of the time interval is in the range of less than 30 s with a total number of 597 dots, which is 97.87% from the aggregated data with total variations. The rest is the interval time longer than 30 s, which has the total number of 13 dots, or only 2.13% of the whole data number. This can be assumed that the car was in a traffic jam, and a new location data was sent just after the position had been changed.

The described data statistic can be justified first with its histogram to see the distribution of the delay time in Fig. 7(a). That distribution is a histogram representation of Table 1. To measure the most probable delay time, Poisson probability density function was calculated using Eq. (2) with λ =1. The result was shown in Fig. 7(b). From the distribution, it could be observed that the probability of the occurrence is from the highest of five red-cross indicators, which were originated at x-axis values close to zero, but with *y*-axis values of 0.037, 0.032, 0,022, 0,012, and 0.004.



(a) Histogram distribution.



(b) Probability density function.

Fig. 7. Time delay distribution.

Once we know the range of delay time, it is possible to plan a real-time analysis at the server. However, the first application that had been developed is only to help the user on monitoring their car position. This could be used as an initial development for the autonomous driving option where the position tracking has been successfully made. It is left with the vehicle control, which should be developed in the future [16]. However, if only manual driving is needed, this application is considered sufficient, and it has more possible features just for a simple car monitoring app. At least three operating conditions of the car have been planned in the developed app.

The problem with recording data from this tracking system was only that the time had not been recorded in detail. Hence, it has not been sure which one has made the interval time vary. It was only based on the driver estimation on the situations of the passing roads, such as the perceived congestion conditions in order to justify the interval components. By those assumptions, the components of interval time can be summarized in Eq. (3).

$$t_{inerval} = t_{GPS} + t_{delay} + t_{waiting} \tag{3}$$

where t_{GPS} is the time for acquiring GPS data from the BTS, t_{delay} is the time for delay of sending the data, and $t_{waiting}$ is the waiting time for changes in GPS location data. Therefore, the overall difference in the data could be distributed. When the road was not jammed, the changes of position can be acquired continuously. Then the first dominant time of 5 s can be achieved as t_{GPS}, and while t_{delay} can be assumed in a variation between 1 to 5 s, where the portion of data has reached more than 91%. The remaining about 8% was the change in time from waiting time as there were a number of positions where the road was rather busy, and there were some positions at a traffic light waiting for the turns. Thus, the next challenge is how to ensure the time interval component through a more detailed recording using the Android program.

3.4. Condition running test

The procedure of sending data to the server is described in Algorithm 1. It requires hardware, which has been discussed previously in sub-section 2.1. They are ELM 327 OBD-II Scanner, Arduino Mega 2560, Shield SIM800L for GSM/GPRS communication, HC-05 for Bluetooth communication, and microSD card as an internal data storage.

Algorithm 1. Sending Data to Server

Require: OBD-II, Arduino (Serial), SIM800L (GSM/GPRS Comm), HC-05 (Bluetooth Comm.), SD card
procedureSENDDATA
serial ← *Initialize Serial Comm.*
    gsm ← *Initialize GSM Comm.*
    bluetooth ← *Initialize Bluetooth Comm.*
while (*serial***AND***gsm***AND***bluetooth*) = 1 **do**
success ← *Connecting_to_OBD-II*
if*success* = 1 **then**
        Request PID Data
        Processing PID Response
        Display LCD: OBD Data
        Send Data to Server
else
        Display LCD: Initialization Status Failed
end if
end while
end procedure

Basically, the objective of this procedure is to ensure the initialization and establishment of each communication module. Firstly, the communication between the Arduino and its shields is established. The purpose of this step is to activate all these serial, GSM, and Bluetooth shields. Secondly, after the shield is activated, the communication with the OBD-II scanner is tried. Thirdly, if this communication is successful, then, the PID data will be requested. It is required in order to get the operating conditions from ODB data. That data will then be displayed in the LCD, which is finally sent to the server. Otherwise, in a case that there is a failed communication with the OBD scanner, a failure indication is shown in the LCD.

The obtained data are used for a condition running test. The test scheme was designed using two different cars, in which each of them employs the same developed hardware monitoring. Both cars were representing typical ICEV-type of cars in Indonesia. The first is Honda Stream 2010 (with car ID: Car#01), and the second is Toyota Rush 2017 (with car ID: Car#02).

The difference between Car#01 and Car#02 is in the initialization of the bus (BUS INIT) between the OBD ELM327 and the Car ECU. BUS INIT on Car#01 was done automatically when installing the unit, READER OBD ELM327, while on Car#02, it was done manually by giving several AT COMMAND to initialize the BUS. Some of these requests were ATZ, ATSP0, and ATE0. Based on the standard Hayes AT command, the commands were ATZ for module reset, ATSP0 for automatic protocol detection, and ATE0 for deactivating the echo-back mode of the module. After initialization was completed, the machine would respond according to the requested parameter ID (PID) request. Some more differences are the accepted data formats. Examples of data obtained are shown in Table 2.

Table 2. Example of obtained data.

Honda Stream (Car#01)	Toyota Rush (Car#02)
48 6B 0A 41 04 62	7E8 03 41 04 3A
48 6B 0A 41 05 80	7E8 03 41 05 7D
48 6B 0A 41 0B 26	7E8 03 41 0B 1E
48 6B 0A 41 0C 0B C5	7E8 03 41 0B 32
48 6B 0A 41 0D 00	7E8 04 41 0C 0E A7
48 6B 0A 41 0E 90	7E8 03 41 0D 00
48 6B 0A 41 0F 5A	7E8 03 41 0E 92
48 6B 0A 41 11 17	7E8 03 41 0F 74
	7E8 03 41 11 19

It could then be translated back into readable information using CAN 11-bit standard. The standard data was divided into several parts. As shown in Table 3, they were identifier, length of data, mode, parameter ID, and data.

The running test was conducted for about 5 months from July 2019 to November 2019 for more than 20 h of driving test. Car#01 recorded 63 data cycles with the data listed in Table 4. The data was equal to approximately 1.5 h of nonstop driving test. Meanwhile, Car#02 was able to record more than 545 data cycles with the reading result described in Table 5. These data equal to approximately 6.5 h of nonstop driving test. The car condition is slightly different compared to Car#01. Each recorded operating condition consists of the min, max, and average values. A comparison of this statistical data can then be summarized in the next part, challenge.

Table 3. Data format.

Function	Translation
IDENTIFIER	7E8 means response message
LENGTH OF DATA	03 is the length of data obtained, which means providing three data, such as 41 0D 00
MODE	41 is the service request mode 01 where 0 is replaced by 4 according to SAE J1979 OBD2, 01 means the latest data request
PID	Parameter ID, e.g., 0D is the parameter ID for vehicle speed
DATA	The data of obtained bytes in Hexa, e.g., 00, means 0 km/h. But to get the original data in decimal form, several formulas have been standardized to convert the data

Table 4. Statistical data of Car#01.

Operating conditions	Min value	Recorded time	Max value	Recorded time	Average
Engine load (%)	0	22/07/2019 21:17:02	100	02/08/2019 05:14:14	53.77
Coolant temperature (˚C)	50	26/07/2019 21:57:00	169	26/07/2019 23:37:49	49.3
Intake manifold pressure (kPa)	64	26/07/2019 23:37:54	238	26/07/2019 23:35:21	69.78
Revolution per minute (rpm)	100	02/08/2019 05:08:01	15,254	26/07/2019 23:35:21	3,223.78
Speed (km/h)	0	23/07/2019 18:24:06	121	27/07/2019 23:59:30	93.25
Forward timing (deg)	-60	26/07/2019 23:35:21	110	02/08/2019 04:44:50	71.16
Air intake (˚C)	-20	26/07/2019 23:37:49	200	02/08/2019 04:59:23	130.67
Throttle position (%)	0	23/07/2019 19:17:21	100	02/08/2019 05:14:14	37.72

Table 5. Statistical data of Car#02.

Operating conditions	Min value	Recorded time	Max value	Recorded time	Average
Engine load (%)	0	22/10/2019 22:55:12	76	31/10/2019 20:42:12	23.11
Coolant temperature (˚C)	40	28/10/2019 06:36:08	95	22/10/2019 23:00:18	82.94
Intake manifold pressure (kPa)	24	04/11/2019 20:34:37	128	22/10/2019 22:57:39	42.5
Revolution per minute (rpm)	684	31/10/2019 20:24:15	4032	05/11/2019 19:22:10	1,152.68
Speed (km/h)	1	28/10/2019 06:27:26	121	22/10/2019 23:05:36	14.1
Forward timing (deg)	-64	22/10/2019 22:56:20	34	04/11/2019 19:28:10	9.47
Air intake (˚C)	-40	31/10/2019 20:37:06	114	04/11/2019 19:31:23	65.04
Throttle position (%)	0	22/10/2019 22:56:53	43	28/10/2019 06:39:43	10.45

3.5. Limitations and constraints of the study

The use of a Bluetooth-based OBD-II scanner enabled straightforward and cheaper systems because there is no need to develop unique hardware to access the data from the car computer system, but only using a simple Bluetooth transmission module. This allows a greater focus on the processing and analysis of data. The recorded results can be viewed using an internet web browser on a map. The results of this position tracking system were analysed by looking at the interval time distribution.

After successfully developed the prototype of the monitoring system, there were at least two challenges that were found and had to be resolved toward the implementation. The first is related to the challenge of how to perform data retrieval continuously for a longer period. Although at this work, the on-board monitoring system and web server had been able to operate, the data collection still had a risk of data loss due to network infrastructure. It could also be seen from the proposed time interval equation. There was time for delay and waiting due to the same data connection problem. The second one was about how to analyse the data, and how to make a proper conclusion after that.

The last issue for the development of analysis might depend on the car manufacturer. The analysis would be from them as well as the maintenance service which could be scheduled based on the analysis. It would be based on the real engine condition rather than period of time or distance approach. Thus, the development of the analysis could be from the expert who could make an interpretation of the data. This kind of diagnostics could also be detected from the abnormalities. It could then be developed also by utilizing artificial intelligence (AI) to sense those operating conditions and additional operating conditions for unmanned vehicles.

In this study, two different cars have been evaluated and been compared in the effort to understand the possible different characteristics of each car. It was found that there was a significant difference between them when applying the monitoring system. Starting from the system initialization on how to activate the OBD device until the processing stage on how to obtain the data, they might have different formats from each other or using different AT commands. It implied from the test result that different car engine had unique electronic data. Thus, this study also emphasized the need for standardization of car monitoring design in the future.

4. Conclusion

An IoT-based condition monitoring system that has been developed in this study had shown the potential to support the development of long-distance vehicle condition monitoring systems. The data that was successfully read by the OBD-II scanner was quite a detail, including information about the car condition. It might be difficult to obtain without accessing the internal sensors placed by the car manufacturer.

From the obtained results, it can be observed that there was a time for GPS retrieval for 5s. By calculating the total of interval time group, it also can be observed that the total was up to more than 91%, with a situation of the road were not in any traffic jam. Thus, the change in delay time could also be estimated. The value was around 1–5 s. The rest was the waiting time, which had a duration of more than 10 s.

From the system running test, a comparison of two different cars by using this car monitoring system showed significant differences. This might indicate that there was the uniqueness of each car. Future work will include the efforts to observe the behaviour of each car individually for further data analysis, and also the application of a machine learning technique to find out its characteristic, therefore, the standardization for car monitoring can be clearly justified.

Acknowledgments

This work has been done with the coordination of three Universities in Indonesia. Each researcher from the university is taking responsibility for each part from the mentioned three parts. Firstly, the hardware module was developed by Fahmi from Universitas Sumatera Utara. Secondly, the android app and its map were organized by E. Sutanto from Universitas Airlangga. Thirdly, a possible AI module development in the webserver was carried out by M. Yazid from Institut Teknologi Sepuluh November. Finally, the overall system was reviewed by M. Aziz as an expert from the University of Tokyo. Authors would like to have many thanks to the Ministry of Research, Technology, and Higher Education (Kemenristekdikti) of Indonesia who provided funding under the “Program Penelitian Kolaborasi Indonesia" (PPKI) scheme of 2019.

References

1. Dobrzyński, M.; Lijewski, P.; Ziólkowski, A.; and Daszkiewicz, P. (2019). Analysis of driving simulation capabilities car vehicle on the engine brake stand. *Proceedings of the 15th Conference on Computational Technologies in Engineering*. Jora Wielka, Poland, 2078(1), 1-5.
2. Ruta, M.; Scioscia, F.; Gramegna, F.; and Sciascio, E.D. (2010). A mobile knowledge-based system for on-board diagnostics and car driving assistance. *Proceedings of the Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. Florence, Italy, 91–96.
3. Aziz, M.; Huda, M.; Budiman, B.A.; Sutanto, E.; and Sambegoro, P.L. (2018). Implementation of electric vehicle and grid integration. *Proceedings of the 5th International Conference on Electric Vehicular Technology*. Surakarta, Indonesia, 9–13.
4. Avdagic, Z.; Cernica, E.; and Omanovic, S. (2019). Adaptive neuro-fuzzy inference system based modelling of vehicle guidance. *Journal of Engineering Science and Technology*, 14(4), 2116–2131.
5. Rajendran, P.; and Smith, H. (2018). Sensitivity analysis of design parameters of a small solar-powered electric unmanned aerial vehicle. *Journal of Engineering Science and Technology*, 13(12), 3922–3931.
6. Rimpas, D.; Papadakis, A.; and Samarakou, M. (2020). OBD-II sensor diagnostics for monitoring vehicle operation and consumption. *Energy Reports*, 6(3), 55-63.
7. Tahat, A.; Said, A.; Jaouni, F.; and Qadamani, W. (2012). Android-based universal vehicle diagnostic and tracking system. *Proceedings of the 16th International Symposium on Consumer Electronics*. Harrisburg, Pennsylvania, 137–143.

8. Menard, T.; and Miller, J. (2011). Comparing the GPS capabilities of the Iphone 4 and Iphone 3G for vehicle tracking using FreeSim_Mobile. *Proceedings of the Intelligent Vehicles Symposium (IV)*. Baden-Baden, Germany, 278–283.
9. Xiao, Z.; Li, P.; Havyarimana, V.; Hassana, G.M.; Wang, D.; and Li, K. (2018). GOI: A novel design for vehicle positioning and trajectory prediction under urban environments. *IEEE Sensors Journal*, 18(13), 5586–5594.
10. SAE J1979/ISO 15031-5. (2017). E/E Diagnostic Test Modes. *SAE International*, USA.
11. Fahmi, F.; Nurmayadi, F.; Siregar, B.; Yazid, M.; and Susanto, E. (2019). Design of hardware module for the vehicle condition monitoring system based on the internet of things. *Proceedings of the International Conference on Information Technology and Engineering Management*. Belitung, Indonesia.
12. Sugiyanti, I. (2019). Design of ATM crime monitoring system based on MQTT protocol using SIM800L and Arduino Mega 2560. *INA-Rxiv*, 1-5.
13. Sutanto, E.; Romadhon., C.M.; Kamil, F.R.; and Rahman., I.M. (2018). Android application for baby immunization schedule. *Proceedings of the 2nd International Conference Postgraduate School*. Surabaya, Indonesia, 511–514.
14. Srinivasan, A. (2018). IoT cloud based real time automobile monitoring system. Proceedings of the *3rd IEEE International Conference on Intelligent Transportation Engineering*. Singapore, 231–235.
15. Nugroho, K.; and Murdowo, S. (2019). Mobile cloud learning system using laravel framework and android studio web view. *2019 International Seminar on Application for Technology of Information and Communication (iSemantic)*. Semarang, Indonesia, 141-144.
16. Arena, F.; and Ticali, D. (2018). The development of autonomous driving vehicles in tomorrow's smart cities mobility. *Proceedings of the International Conference of Computational Methods in Science and Engineering*. Thessaloniki, Greece.

Appendix A

Computer Programme

- Android app could be downloaded from Google Playstore at https://play.google.com/store/apps/details?id=id.web.trace.carmonitor
- URL for accessing recorded tracking data could be accessed from http://trace.web.id/v1/