# Smart Rash Driver System via Internet of Things (IoT)

*Kong* Sheau Tong[1], *Mohd Natashah* Norizan[1,2,3,*], and *Ili Salwani* Mohamad[1,2,3]

[1]School of Microelectronic Engineering, Universiti Malaysia Perlis, Pauh Putra Campus, 02600 Arau, Perlis, Malaysia.
[2]Centre of Excellence Geopolymer and Green Technology (CeGeoGTech), Universiti Malaysia Perlis, Perlis, Malaysia.
[3]Advanced Multi-Disciplinary MEMS-Based Integrated NCER Centre of Excellence (AMBIENCE), Universiti Malaysia Perlis, Perlis, Malaysia.

**Abstract.** Nearly half a million accidents on Malaysians road occur in 2015. The aim of this research is to detect car speed, capture the photo of the speeding car and then transfer the data like car speed, date and time, location and lane number to an online database. A distance sensor is used to measure the distance range between two points on the road. The ESP8266 NodeMCU will be the control unit to process the data and calculate the speed with the formula of speed equal to distance over time. The ESP8266 NodeMCU is also a Wi-Fi module to help in transferring data via IoT to an online database. The Google spreadsheet acted as an online database and will receive all the data if detected a speeding car. In conclusion, the Smart Rash Driver System is successfully invented and able to detect vehicle speed, capture the photo of over speed vehicle and save it to the SD card and lastly transfer all data via IoT to the Google Spreadsheet. This invention will be able to help to decrease the road accident rate efficiently.

## 1 Introduction

In 2015, nearly half a million accidents on Malaysians road were recorded [1, 2]. Most of the cases happened were caused by the human error, faulty vehicle, driving under the influence of alcohol, drug effect and speeding [3-7]. Moreover, according to the cause of death record of World Health Ranking which is updated on 2014, road accident is rank 5 in Malaysia and rank 34 in worldwide [8-10].

The government has figured out a solution for this problem by placing Automated Enforcement System (AES) speed tracker near major roads in Malaysia [11]. According to Jabatan Pengangkutan Jalan Malaysia, after the first eight months since the AES was implemented, the accident rates at 14 locations nationwide have dropped by up to 30% [12]. Therefore, more speed trackers need to be installed to decrease the rate of road accident as mentioned by Malaysian Institute of RoadSafety (Miros) Chairman [13]. Road users might not be stopped from speeding in camera-free section, but they will be more aware on the speed limit all along the road. However, a huge cost needed to install those speed monitoring system such as AES in whole Malaysia, therefore human intervention as well is needed to detect the speed of moving vehicle [11]. The drawback of human intervention is human unable to work 24 hours and besides, human error might occur [14-17].

The objective of this project is to develop a low cost speed monitoring system, which can capture the photo for car that are speeding. Besides, this research also will provide real-time wireless reporting data speed, location, date and time, location and lane to the centralized database via Internet of Things (IoT). NodeMCU which is an open source IoT platform will be used for the data transfer as it promotes a breakthrough in wireless sensor networks [19] and very popular among various applications [20-23].

## 2 Proposed architecture

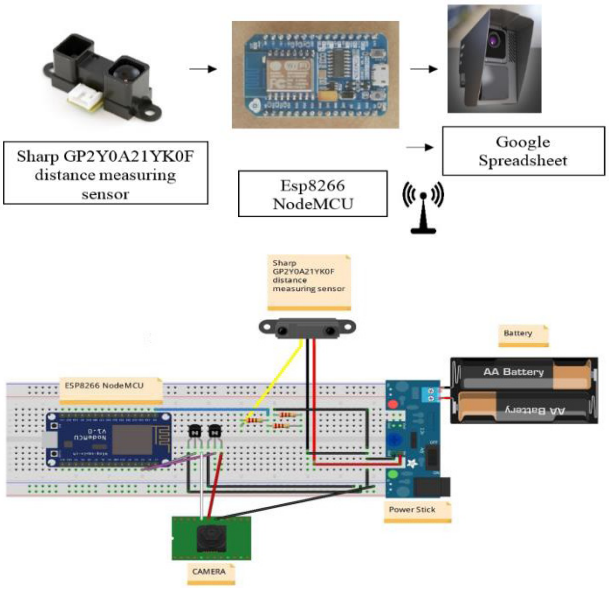


**Fig. 1.** Overview of system block design and schematic diagram.

The smart rash driver system is a speed monitoring system, which able to capture car speed, date, time, lane,

* Corresponding author: mohdnatashah@unimap.edu.my

location and then transfers the data to a database via IoT. Besides, when the system detected a speeding car, it will trigger the camera automatically to capture the car's photo. Therefore, the input of the ESP8266 NodeMCU is a Sharp GP2Y0A21YK0F distance measuring sensor and the output of the ESP8266 NodeMCU will be the camera and Google spreadsheet. Figure 1 shows the overview of the system block diagram and the schematic diagram of the design.

The main input for the speed monitoring system is the Sharp laser sensor and the main control unit will be the ESP8266 NodeMCU. ESP8266 NodeMCU will act as a control unit and Wi-Fi module at the same time. After processing the data, ESP8266 NodeMCU will send it via IoT to Google spreadsheet. Lastly, the authorities will able to observe the data from the Google spreadsheet.

Resistor in the circuit act as the voltage divider of the circuit. It can change the input voltage of the Sharp distance sensor from 5 to 3.3 V. Therefore, it will not burn out the ESP 8266 NodeMCU because of the high voltage. ESP 8266 NodeMCU only able to support the input voltage that less than 3.3 V. Moreover, the transistor used in the circuit is to push the input voltage to the maximum so that it can provide enough power to trigger the camera. The breadboard power stick can be powered by two 3.6 V batteries. Therefore, it makes the system become portable.

### 2.1. System design

Figure 2 shows the system design flowchart. At the beginning, the ESP8266 NodeMCU will check the Wi-Fi connection with the Internet. Then, the Sharp GP2Y0A21YK0F distance sensor will start to detect the distance range. The distance between two points will be calculated when the vehicle passed through the sensor. The system on road diagram in Figure 3 shows the measurement of the distance while speeding.

The system will calculate the speed with the distance range that is received from the sensor and the time range that provided by the Arduino IDE in the unit of a millisecond. The data of distance1, time1, distance2, and time2 are then collected. The formula of calculating the speed is:

$$Speed = \frac{Distance1 - Distance2}{Time2 - Time2} \quad (1)$$

After calculating the speed, the unit of the speed will be cm/ms, therefore a conversion needed for changing the unit to km/hour. For getting the timing, time will be provided from the Arduino IDE with the coding of "millis ()". The timing will be count in the unit of milliseconds. The reason of using millisecond instead of a second because it is the smaller unit of time and able to increase the accuracy of calculating the speed of the moving vehicle.

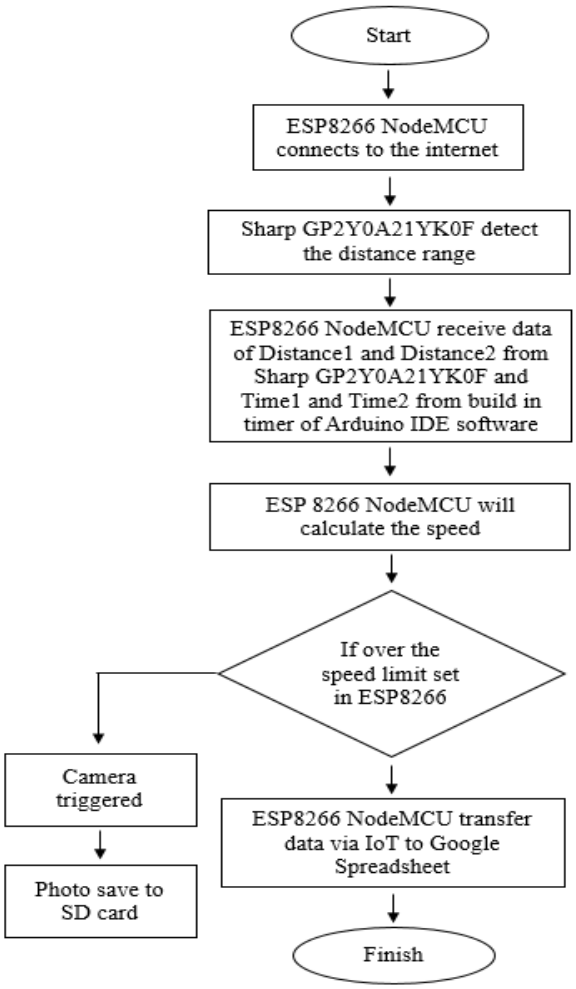


**Fig. 2.** System design flowchart.

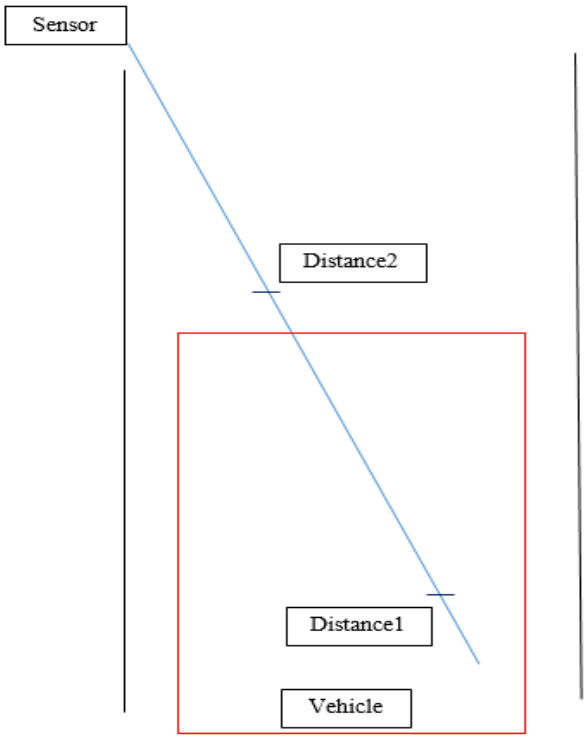


**Fig. 3.** System on road diagram.

After the speed is calculated, the ESP8266 NodeMCU will begin to process the data. If the speed of the vehicle are over the speed limit that being set, it will trigger the camera to capture the photo of the speeding vehicle. The photo captured will be saved on an SD card. Besides, the ESP8266 NodeMCU will also provide the Wi-Fi connection ability for sending the data wirelessly to an online database.

### 2.2. Internet of Things (IoT) design

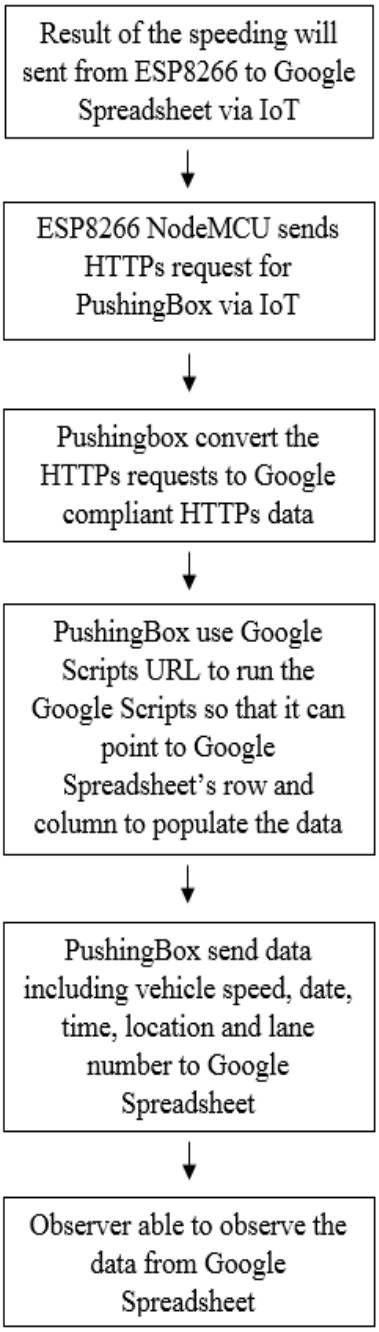


**Fig. 4.** IoT process flow chart.

Figure 4 shows the IoT process flow chart. Before uploading the data to the Google spreadsheet, there are a few steps need to be done. Firstly is creating and deploying a Google spreadsheet. Log into the free Google account and create a new "Google Sheet". This is the spreadsheet which will be populated with all the data of speed, location, date and time happened, lane number and location via Wi-Fi. After creating the Google spreadsheet, the spreadsheets URL key is copied and saved. This key is listed in the URL between the "/d/" and the "/edit" of the new spreadsheet. Lastly, the title of the spreadsheet is renamed. Figure 5 shows the example of Google spreadsheet.

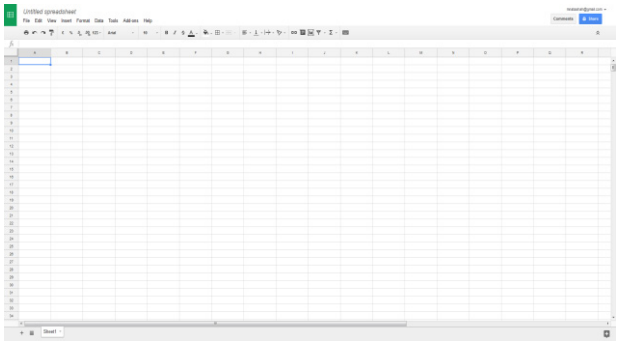

**Fig. 5.** Google spreadsheet.

Next, a Google app script (similar to JavaScript) is created which will process the data and populate to the specific spreadsheet correctly. To insert the Google app script, first navigate from the spreadsheet to the script editor. Then, the coding that prepared for the Google spreadsheet is keyed in to Google script and then the Google spreadsheet URL key is inserted to make sure the data point to the right spreadsheet as shown in Figure 6. Users need to program the Google Scripts so that user able to program a build in date and time, all the data are then populated into specific rows and columns.

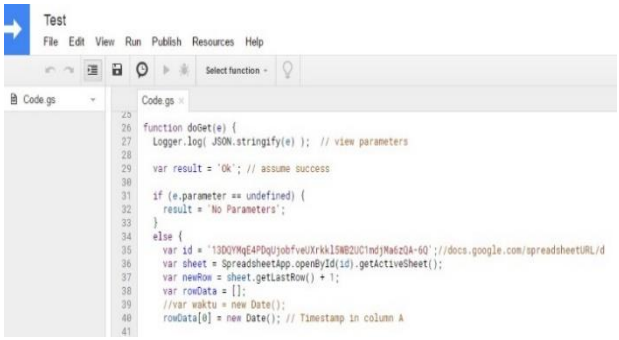


**Fig. 6.** Google script.

Next, the Google spreadsheet is saved in other to receive the web app URL to be used in the next step. Then the Web app URL need to be saved.

The current web app URL that is shown in Figure 7 will be used for PushingBox so that the PushingBox able to communicate with the Google app scripts. For the last step of the IoT is configuring PushingBox. The need for using the PushingBox API is to convert the HTTP transmitted data into Google compliant HTTPS encrypted data. The web app URL will be used now for PushingBox to send the data to a right Google sheet. First, a service must be created in PushingBox to let the user interface with any API of any product. Figure 8 shows how to create a service in PushingBox.

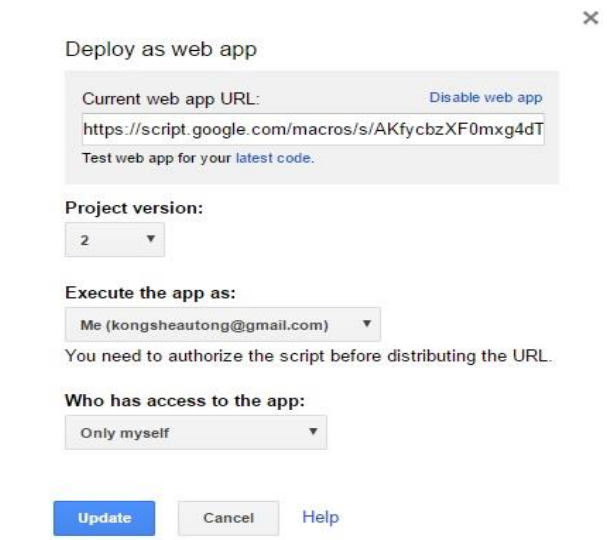


**Fig. 7.** Save the web app URL.


**Fig. 8.** PushingBox services.

Next, the web app URL that is shown in Figure 7 will now be used in the PushingBox while creating the services. Figure 9 shows the web app URL of Google scripts will be filled in the column of Root URL. There are two options able to choose in the column of method, which is Get or Post. For this project, the PushingBox need to fetch the data from the ESP8266, therefore, the option that chosen need to be Get.

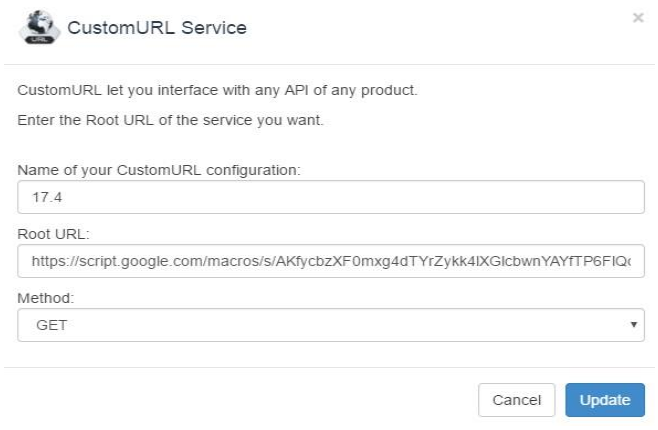


**Fig. 9.** Key in the web app URL in the Root URL.

Then a scenario is created and user able to receive a Device ID as shown in Figure 11. Device ID will be used in the program code of Arduino IDE as shown as Figure 10. The device ID is used to locate the right service of PushingBox device. The yellow highlight section in Figure 10 is the place that placing the Device ID. The Device ID can be changed for other different services.

```
#include <ESP8266WiFi.h>
#define D1 5
const char* ssid = "Tong";
const char* password = "58648851";

const char WEBSITE[] = "api.pushingbox.com"; //pushingbox API server
const String devid = "v262EAE71022AF18"; //device ID from Pushingbox
```

**Fig. 10.** Device ID.


**Fig. 11.** Device ID with command.

Figure 11 is the place that placing the command string to tell the PushingBox which data it should fetch and what data should be posted. The data that with the symbol of $Distance$ is means that the data is getting from the microcontroller. User also able to set a constant data that desire to be posted to the Google spreadsheet, the data must be wrote without the symbol of $. Lastly, through those processes, the data is able to transfer wirelessly to the Google spreadsheet successfully and accurately.

## 3 Results and discussion

Figure 12 shows the result of the prototype that is made for this project. The prototype contains a Sharp GP2Y0A21YK0F distance sensor, ESP8266 NodeMCU, camera, resistor and transistor.

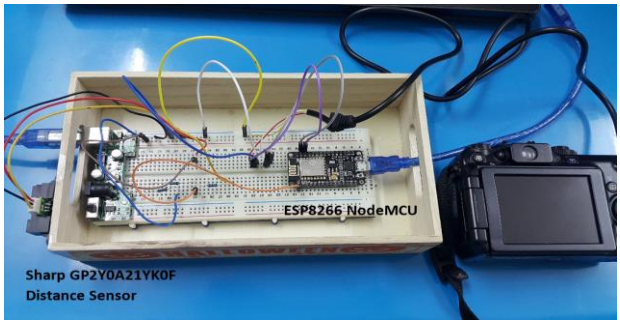


**Fig. 12.** Prototype of the smart rash driver system.

Figure 13 shows the result that the ESP 8266 NodeMCU has connected to the Internet. It will show "Credentials accepted! Connected to wifi" at the serial communication of Arduino IDE. After the successful connection, the data will be able to be transferred wirelessly.

Figure 14 shows the condition for the serial communication of Arduino IDE when the ESP 8266 NodeMCU unable to connect to a Wi-Fi Server. If the ESP 8266 NodeMCU did not connect to an Internet, the serial communication will keep showing "……." until it connected to an Internet.


**Fig. 13.** ESP 8266 NodeMCU connected to the Internet.

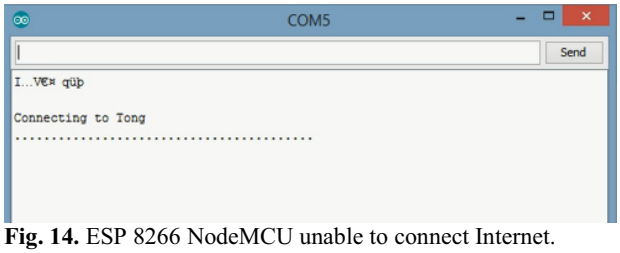


**Fig. 14.** ESP 8266 NodeMCU unable to connect Internet.

Figure 15 shows the result of the data that are uploaded to the Google spreadsheet. The data uploaded to the Google spreadsheet consist of vehicle speed, date and time, lane number and the location of speeding. As for testing purpose, the system has been set to capture the vehicle with speed over 0.5 km/h.

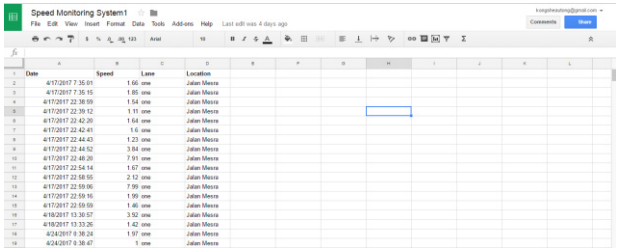


**Fig. 15.** Result of speeding upload to Google spreadsheet.

## 4 Conclusions

This work demonstrated a smart rash driver system that manage to capture of the speeding car, calculating the speed and transfer the data to online database via IoT through PushingBox service. Compared to the existing AES, this system does not need LAN cable to transfer the data besides of it is a portable system (easy to install and operate). Based on the evaluation, the data and photo collected were in real time condition and successfully transferred to the database when speeding car triggered the sensors. Therefore, this system is expected to help the authorities to reduce the road accidents in the future with an efficient monitoring system.

## References

1. Bernama, Malay Mail Online (2017). URL: http://www.themalaymailonline.com/malaysia/article/average-of-18-people-killed-daily-in-road-accidents-in-2015-says-liow [accessed: 2017-01-13]
2. T. Yi Liang, The Star Online (2016). URL: http://www.thestar.com.my/news/nation/2016/04/18/transport-ministry-almost-half-a-million-road-accidents-in-2015/ [accessed: 2017-01-15]
3. N. H. Rodzi, The Star Online (2015). URL: http://www.thestar.com.my/news/nation/2015/05/02/accident-family-death/ [accessed: 2017-01-15]
4. R. Krishnan and R. U. R. Sohadi, J. Heal. Transl. Med. **2**, 39 (1997).
5. A. Kareem, Malaysian J. Med. Sci. **10**, 31 (2003).
6. L. Abdullah N. Zamri, J. Emerg. Trends Comput. Inf. Sci. **3**, 225 (2012).
7. M. G. Masuri, K. A. M. Isa, M. P. M. Tahir, Procedia - Soc. Behav. Sci. **38**, 213 (2012).
8. WHO, HEALTH PROFILE : MALAYSIA (2014). URL: http://www.worldlifeexpectancy.com/country-health-profile/Malaysia [accessed: 2017-01-17]
9. WHO, Global Status Report on Road Safety 2015 (2015). URL: http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/ [accessed: 2017-01-17]
10. E. Zarulazam, H. Evdorides, IATSS Res. (2017)
11. H. Sivanandam, The Star Online (2015). URL: http://www.thestar.com.my/news/nation/2015/11/16/parliament-contract-aes/ [accessed: 2017-01-21]
12. M. Carvalho, Y. Meikeng, H. Sivanandam, The Star Online (2014). URL: http://www.thestar.com.my/news/nation/2014/10/30/aes-helps-reduce-accidents-says-liow-cases-at-14-locations-dropped-30-since-implementation/ [accessed: 2017-01-21]
13. Bernama, TheStar Online (2016). URL: http://www.thestar.com.my/news/nation/2016/04/29/more-aes-cameras-necessary-to-nab-speedsters/ [accessed: 2017-01-21]
14. J. M. Harrington, Occup. Environ. Med. **58**, 68 (2001).
15. A.G. Foord, W.G. Gulland, Process Saf. Environ. Prot. **84**, 3, 171-173 (2006).
16. J. M. Haight and R. G. Caringi, Int. J. Risk Assess. Manag. **7**, 708 (2007).
17. J. M. Haight, V. Kecojevic, Process Saf. Prog. **24**, 45 (2005).
18. S. Borsay, Transmit ESP8266 Data to Google Sheets (2016). URL: https://www.hackster.io/detox/transmit-esp8266-data-to-google-sheets-8fc617 [accessed: 2017-01-25]
19. M. Mehta, Int. J. Electron. Commun. Eng. Technol. **6**, 7 (2015).
20. R. L. Tirupatamma, A. Pulagam, Int. J. Eng. Sci. Res. Technol. **5**, 634 (2016).
21. K. Chooruang, P. Mangkalakeeree, Procedia Comput. Sci. **86**, 160 (2016).
22. T. A. Abdulrahman, O. H. Isiwekpeni, N. T. Surajudeen-Bakinde, A. O. Otuoze, Procedia Comput. Sci. **94**, 473 (2016).
23. B. Yong, Z. Xu, X. Wang, L. Cheng, X. Li, X. Wu, Q. Zhou, J. Parallel Distrib. Comput. (2017).