Journal of Multimedia Information System VOL. 6, NO. 4, December 2019 (pp. 239-244): ISSN 2383-7632(Online)
http://doi.org/10.33851/JMIS.2019.6.4.239

Design and Implementation of a CAN Data Analysis Test Bench based on Raspberry Pi

Sudarshan Pant[1], Sangdon Lee[2*]

Abstract

With the development of Cyber-Physical Systems(CPS), several technologies such as automation control, automotive and intelligent house systems have been developed. To enable communication among various components of such systems, several wired and wireless communication protocols are used. The Controller Area Network(CAN) is one of such wired communication protocols that is popularly used for communication in automobiles and other machinery in the industry. In this paper, we designed and implemented a response time analysis system for CAN communication. The reliable data transfer among various electronic components in a significant time is crucial for the smooth operation of an electric vehicle. Therefore, this system is designed to conveniently analyze the response time of various electronic components of a CAN enabled system. The priority for transmission of the messages in the CAN bus is determined by the message identifier. As the number of nodes increases the transmission of low priority messages is delayed due to the existence of higher priority messages on the bus. We used Raspberry Pi3 and PiCAN2 board to simulate the data transfer for studying the comparative delay in low priority nodes.

Key Words: CAN, Raspberry Pi, Response Time.

I. INTRODUCTION

Easy to use test bench for analysis of CAN communication is essential for performing experiments without the use of actual vehicles. This study focuses on developing a cost-effective test bench for studying CAN communication using an easily available card-sized mini-computer. In the systems where the specific data sources continuously sense the environment and transmit data to other components of the system. The CAN bus is one of such communication solutions. CAN is a widely-used communication protocol, originally developed for use in the automotive industry. However, its use is not only limited to use in automobiles. It provides high-speed serial communication in various domains such as manufacturing industry, medicine, and agriculture. The CAN protocol is a popular choice for such systems due to its safety, real-time capabilities, and the automatic control of access to the data bus. The CAN is a peer-to-peer network where there is no master node and the individual nodes can send and receive data through a can bus. With the highest speed of 1MB, CAN bus is a message-oriented protocol designed to operate at speeds from 20kb/s to 1Mb/s. CAN has built-in support for error detection and re-transmission of data.

This paper is organized as follows: Section II briefly describes the CAN standard and the Raspberry Pi and PiCAN2 modules used in this study for CAN communication. Section III presents the overall design of the proposed test bench for analyzing response time. Section IV summarizes the experiments and observations. Finally, Section V concludes the paper.

II. RELATED WORK

Different techniques and hardware have been used to study CAN data. Meseguer et al. designed a platform for assisting drivers to ensure safe and economical driving style [1]. They developed an Android application to communicate with the CAN data scanner, ELM327, for accessing speed, acceleration and throttle position

Manuscript received December 11, 2019; Revised December 20, 2019; Accepted December 22, 2019. (ID No. JMIS-19M-12-060)
Corresponding Author (*): Sangdon Lee, Address: 58554 Youngsan-ro 1666, Cheonggye-myun, Muan-gun, Jeonnam, Korea, +82-61-450-2357, sdlee@mokpo.ac.kr
[1]Dept. of Multimedia Engineering, Graduate School, Mokpo National University, darshanz@mokpo.ac.kr
[2]Dept. of Multimedia Engineering, College of Engineering, Mokpo National University

information which was transmitted to the server in XML format for offline analysis. Similarly, the Android-based systems were developed by [2, 3] to communicate with ELM327 to collect the vehicle information. These studies use a commercial On-Board-Diagnosis(OBD) scanner which scans the vehicle CAN data through OBD port and transmits using Bluetooth or WiFi. Rhyu et al. analyzed the transmission delay of lower priority messages according to the number of higher priority messages using the signaling model as the delay model [4]. Baur et al. implemented a proof-of-concept prototype based on Raspberry Pi and PiCAN for evaluating technical and privacy aspects [5]. They focused on the privacy issues in the agricultural domain during collaborative tasks. Similarly, Nguyen et al. also proposed the ECU design based on commercial Raspberry Pi and PiCAN devices and used it for testing automotive security.

In most studies related to accessing vehicular CAN data, the OBD port has been used. The OBD port is suitable for diagnostic purposes hence the important factors such as actual polling rate and communication delay have not been considered. In order to study the latency in communication, we designed a test bench using Raspberry Pi3 and a companion CAN shield.

2.1. Controller Area Network

CAN is a serial communication protocol standardized by ISO for communication among various electronic components in automobiles. Being a Low-cost, a lightweight network having priority management and error handling capabilities CAN networks have been intensively used for carrying periodic messages [7].

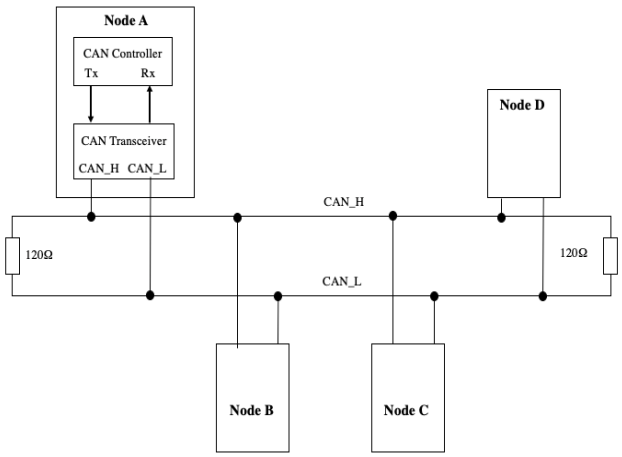


Fig. 1. A CAN Bus.

CAN is one of the commonly used communication protocols in vehicles [8]. The protocol was originally developed in the 1980s by Bosch GmbH [9]. CAN network is formed by a pair of wires indicated as CAN-H and CAN-L constituting a CAN Bus. The Electronic components or the Electronic Control Units (ECUs) are connected to the pair of wires forming a communication network with a multi-master serial bus. Such a multi-master system does not require a central supervisory unit and thus all the connected nodes have the same work priority and the frames broadcasted by a node are received by all the nodes. The CAN operates on CSMA/CD (Carrier Sense Multiple Access/Collision Detection) principle, where a single carrier is shared among a number of components. All the components are connected to each other through a bus formed by a pair of wires with an impedance of 120 ohms as shown in Fig. 1.

There are two types of CAN messages; CAN2.0A and CAN2.0B with 11-bit and 29-bit identifier respectively. The numerical identifier of the message determines the priority of the transmitted message by resolving the collision between two transmitting nodes. The frame format for an 11-bit CAN message is shown in Fig. 2.

Fig. 2. Standard CAN message frame.

In the standard CAN frame, the CAN arbitration ID is an 11-bit field and it is used to identify different devices and to prioritize message. Another important field is the CAN data field having length up to 8 bytes, holding the actual data to be transmitted by the device. Based on the functionality, the data frame can be of different types:

- General frame: a frame used for data transfer
- Remote frame: a frame requesting the transmission of a specific identifier
- Error frame: a frame sent by a node on detection of an error
- Overload frame: a frame used to inject a delay between data.

2.2. Raspberry Pi with PiCAN2

Raspberry is a small-sized general-purpose computer created by Raspberry Pi foundation and can perform general computing tasks using optimized Linux distributions like Raspbian or Ubuntu. It features a powerful BCM2837 64 bit ARM Cortes A53 quad-core processor and 1GB RAM. Raspberry Pi has support for numerous devices and interfaces for peripherals. It has built-in wireless and Bluetooth modules which makes the system connect to the external system with ease. PiCAN2 is a HAT (Hardware Attached on Top) interface for Raspberry Pi. The HAT interfaces use the GPIO slots of

the Raspberry Pi for connection. PiCAN2, produced by SK Pant Electronics Ltd enables CAN communication on a Raspberry Pi computer. The main components of PiCAN2 include the Microchip MCP2515 CAN controller with MCP2551 CAN transceiver, enabling the Raspberry Pi to operate on CAN standards with speed of up to 1 Mbps. The connection to a bus is done through a three-way screw terminal or 9-pin D-SUB connector.

For our experiments, we connected the PiCAN2 Module to a CAN bus using a twisted wire connected to its screw terminal as it allows convenient connection without soldering or additional peripheral device. A PiCAN2 module is equipped with a 12-ohm resistor for terminating the network. It is necessary to terminate the network with 120-ohm resistors on each of its sides in order to avoid reflection from the ends of a CAN bus [9].

III. PROPOSED SYSTEM

In order to perform the experiments, we designed a CAN network test bench instead of conducting experiments on real vehicles, which is not very practical due to the high cost and risk of interference with the vehicle functionality during experiments. The experiment test bench included all the necessary components required for simulating the CAN communication among various ECUs of real vehicles. Fig. 4 and 5 show a basic overview of the system which involves a CAN network formed by four Raspberry Pi 3 nodes. Each node is equipped with PiCAN module mounted for CAN communication. In Fig. 3. the Raspberry Pi board is colored in green and the Pi-CAN2 board, colored blue, is mounted on the Raspberry pi board as a shield. With MCP2515 as a CAN controller and MCP2551 as a CAN transceiver, PiCAN2 board enables CAN communication on a Raspberry Pi. Each node is powered with a micro-USB cable connected to the Raspberry Board and PiCAN board shares the power through GPIO pins.

Besides the CAN wiring, the nodes in this system are connected to the same Local Area Network for data extraction process as the data is analyzed in the server. The messages in the CAN bus are transmitted based on the message identifier field in the CAN message. The lower the identifier value the higher is the priority. We set up speed and battery nodes for transmitting messages with different identifiers. The transfer time was logged and sent to the server for analysis.

For the simulation of the dashboard, one of the Raspberry Pi devices is equipped with a display that shows the data received from other nodes in the CAN Network. The dashboard facilitates access to the CAN data transmitted among the nodes as shown in Fig. 6. The graphical user interface was created using Python language with TKinter package. The battery and speed information obtained from the respective nodes are displayed graphically in real-time. For direction indicators, the controls are placed on the dashboard which can be accessed through the touch display.

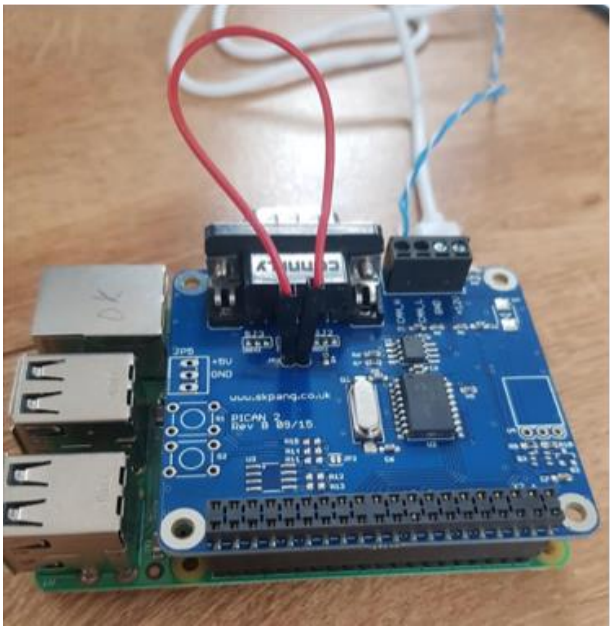


Fig. 3. A CAN node with Raspberry Pi and PiCAN module.

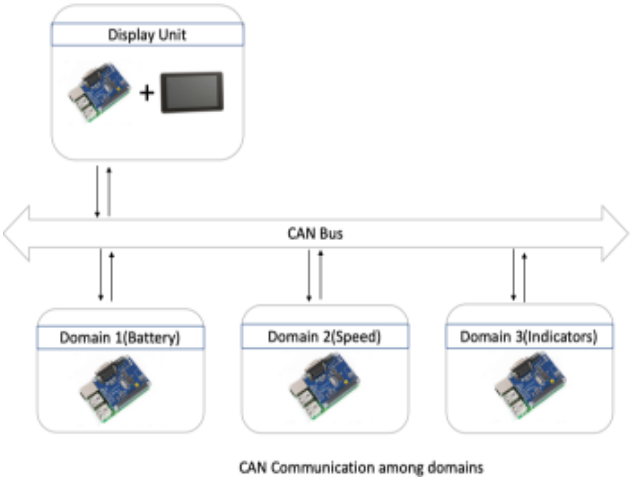


Fig. 4. Design of an experimental test bench for CAN communication.

The software implementation includes three major modules, Network module, ECU and a visual display unit. The Network module used the python-can library for CAN communication. This module handles the communication between the visual display unit and the ECUs. A visual display unit displays data received from various ECUs and

provides an interface for controlling the ECU actions. Analysis of the CAN data is done on the remote host, the response time logs are sent from Raspberry Pi nodes to the remote server for processing. The logs form different nodes are combined and the delay is calculated. Python packages such as Pandas and Matplotlib are used for the extraction of the log and visualization of the response times.

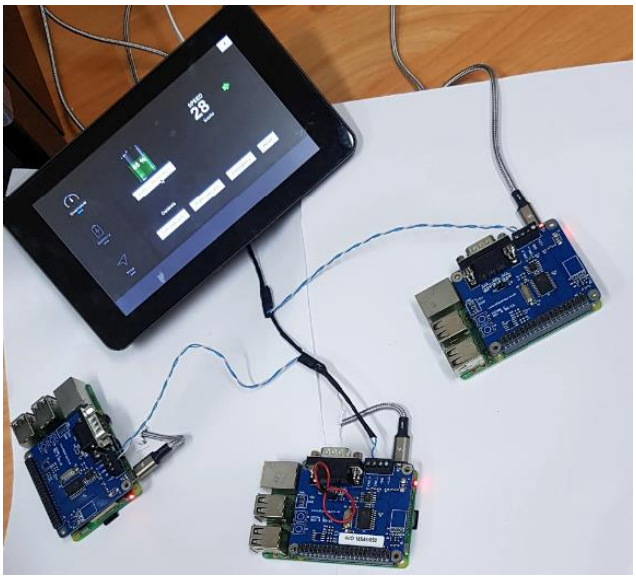


Fig. 5. The Experiment test bench with 4 CAN nodes.

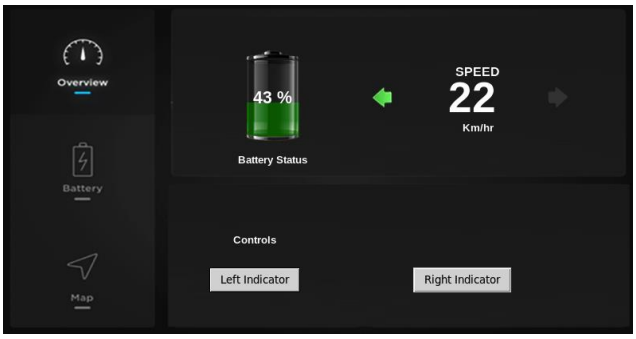
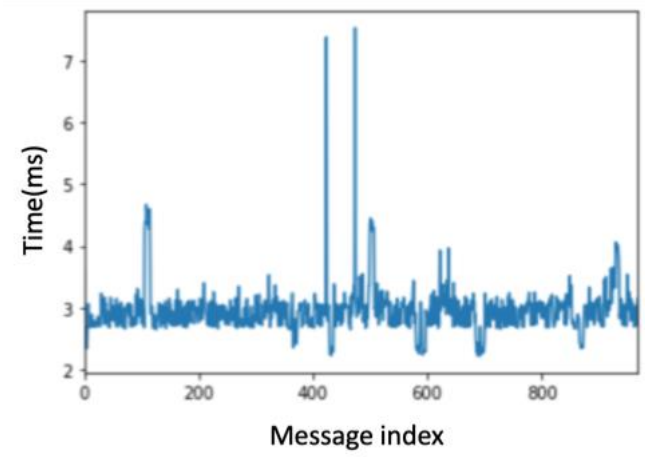


Fig. 6. The Dashboard Interface.

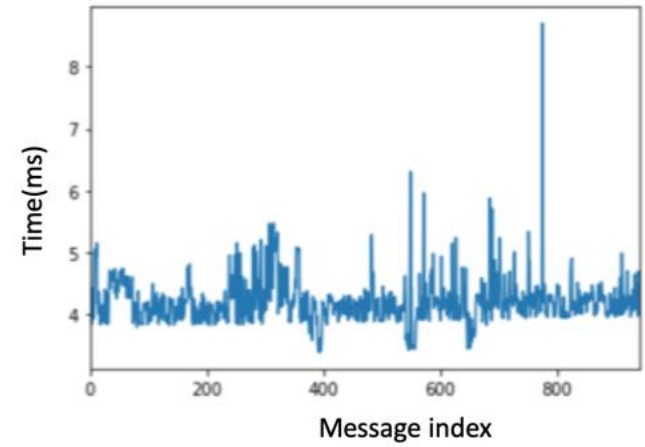
The development of the system was done on a Linux environment using a 64-bit Ubuntu 18.04 desktop distribution.

IV. EXPERIMENTS AND RESULTS

Using the proposed test bench, we measured the response time of the data transmitted from one device to another. The data transfer time was calculated for a speed node having high priority and a battery node having low priority. The data transferred between the ECUs involved the speed and battery percentage values. The payload with uniform size was used to study the delay in transmission of the messages based on priority only. The experiment results show that the existence of high priority nodes affects the transmission of low priority messages.



(a) Time taken by high priority messages.



(b) Time taken by low priority messages.

Fig. 7. Time taken by the messages with different priorities.

Fig. 7 (a) and (b) show the time taken by the speed having a higher priority on the bus and the battery information with lower priority. It was observed that the average time taken by the messages to reach from speed ECU to the dashboard was 2.95ms, where the minimum delay was measured to be 2.21ms and the maximum delay was up to 7.53ms for some messages. On the other hand, the low priority messages experienced the average delay of 4.20ms with the minimum of 3.39ms and the maximum of 8.69ms as shown in the table 1.

Table 1. The delay details of battery and speed nodes.

	Minimum	Maximum	Average
Battery	**3.39ms**	**8.69ms**	**4.20ms**
Speed	**7.53ms**	**2.21ms**	**2.95ms**

V. CONCLUSION

In this paper, we designed a test bench for measuring the response time of the data transmitted from one device to another using RaspberryPi3 and PiCAN2 CAN shield. The proposed system can not only be used to extract raw CAN data from the CAN bus but also displays selected features such as battery and speed information visually. This system was designed for analyzing response time for CAN communication in different conditions such as varying messages and varying numbers of CAN nodes. Furthermore, the system is extensible to perform various experiments using additional nodes and different data sizes. The future work includes the development of a customizable test bench to analyze various types of CAN data. In this study, the log collection and analysis were automated using Linux shell scripts with user intervention for visualization. In the future, such tasks can be processed with an easier graphical interface with more user-friendly interaction.

Acknowledgement

Following are results of a study on the "Leaders in INdustry-university Cooperation +" Project, supported by the Ministry of Education and National Research Foundation of Korea.

REFERENCES

[1] J. E. Meseguer, C. T. Calafate, J. C. Cano, P. Manzoni, "DrivingStyles: a smartphone application to assess driver behavior," in *Proceedings of the International Symposium on Computers and Communications*, pp. 535–540, 1994.

[2] M. Reininger, S. Miller, Y. Zhuang, J. Cappos, "A first look at vehicle data collection via smartphone sensors," in *SAS 2015 - 2015 Proceedings of IEEE Sensors Applications Symposium*, April 2015.

[3] Y. Yang, B. Chen, L. Su, D, Qin, "Research and Development of Hybrid Electric Vehicles CAN-Bus Data Monitor and Diagnostic System through OBD-II and Android-Based Smartphones," *Advances in Mechanical Engineering*, vol. 2013 pp. 1-9, 2013.

[4] S. Ryu, J. Bang, J. Han, "Transmission Delay in a CAN System," in *Proceedings of Symposium of the Korean Institute of communications and Information Sciences*, pp. 419–420, 2019.

[5] J. Bauer, R. Helmke, A. Bothe, N. Aschenbruck, "CAN't track us: Adaptable privacy for ISOBUS controller area networks," *Computer Standards & Interfaces*, vol. 66, October 2019.

[6] H. N. Nguyen, S. Tavakoli, S. A.Shaikh, O. Maynard, "Developing a QRNG ECU for automotive security: Experience of testing in the real-world," in International Conference on Software Testing, Verification and Validation Workshops, April 2019.

[7] M. Farsi, K. Ratcliff, M. Barbosa, "An overview of controller area network," *Computing & Control Engineering Journal*, vol. 10, no. 3, pp. 113-120, Jun. 1999.

[8] G. Leen, D. Heffernan, "Expanding automotive electronic systems," *Computer*, vol. 35, no. 1, pp. 88-93, Jan. 2002.

[9] Bosch, *CAN Specification, Version 2.0*, Robert Bosch GmbH, 1991.

Authors

Sudarshan Pant received the B.S., and M.S., in Multimedia Engineering in Mokpo National University, Korea in 2010, and 2012, respectively. He worked at Arihant Technologies, Nepal and TekTak Nepal Pvt Ltd, Nepal as Software Engineer and Mobile Application Architect respectively from 2012 to 2016. He is currently a Ph.D. student in department of Multimedia Engineering at Mokpo National University. His research interests include Data Science, Computer Vision, and Machine Learning.

Sangdon Lee received his B.S., M.S., and Ph.D. degrees in Computer Engineering from the department of Computer Engineering, Seoul National University, Korea in 1984, 1986 and 1996 respectively. He worked at Research and Development Group in Korea Telecom, Seoul, Korea for 10 years before joining Mokpo National University. He is currently a professor of the Department of Multimedia Engineering. His research interests include big data management, multimedia services and intelligent data processing.