Received December 10, 2021, accepted January 24, 2022, date of publication March 10, 2022, date of current version March 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3158269

Effective In-Vehicle Network Training Strategy for Automotive Engineers

JIN YOUNG BYUN, JAE WAN PARK, DO YEON KIM, CHANG YOUNG JO, AND JAE WOOK JEON, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea

Corresponding author: Jae Wook Jeon (jwjeon@yurim.skku.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT), Korea Government, under Grant 2020R1A2C3011286; and in part by Hyundai NGV (Next Generation Vehicle).

ABSTRACT This study describes a hands-on in-vehicle network (IVN) training course for automotive engineers, particularly for learners with different prior knowledge. We collected essential content to educate those with different backgrounds on IVN-related information, which is essential in practice. The course aims within a short period to effectively educate engineers working in the industry on IVN systems, and explains how to employ embedded systems and graphical user interfaces (GUIs) for this purpose. At the end of the training, the level of knowledge before and after training was compared to evaluate the training results, and engineers' responses were recorded.

INDEX TERMS Automotive engineering, engineering curriculum, engineering technology, CAN, ethernet, experiential learning, in-vehicle network, teaching skills.

I. INTRODUCTION

A. MOTIVATION

This course was initiated by a training request from a company to organize a course to help students understand the fundamentals of the IVN. All vehicle information comes and goes through the IVN, and this structure is widely applied to the internal combustion engine, electric, and hydrogen vehicles. Even if the students are not engineers designing the IVN directly, there are common key elements that all engineers involved in vehicle control need to know, and we were tasked with organizing a training course for that. Therefore, the design purpose of this curriculum is to effectively understand the basic core contents of IVNs centered on controller area network (CAN) and Ethernet, and the target audience is engineers engaged in vehicle-related development. The theory course has been in operation for over 10 years, and at the beginning, the customers made the point that it was difficult to apply the theory in the field, because the course consisted only of theory classes. To address this point, we have organized a hands-on course that is similar to the conceptual structure of IVNs. This paper mainly introduces a practical training course, which is configured to be as similar to the actual IVN as possible, and the hands-on experience for vehicles can be obtained by directly using the toolkit with the Aurix TC275 microcontroller unit. However, because of the wide range of networks covered in the training, the process of practice should not be complicated. As a result, rather than manually writing the code, the format of using the modules configured in the toolkit is adopted. Various communications can be easily implemented using the display, actuator, and sensor included in this toolkit, to effectively understand the mechanism of the electronic control unit (ECU).

B. RELATED WORK

An ECU is an embedded device used in vehicles to control the internal operation of an engine in various ways, and they are used in automotive systems to control engines, as well as transmissions, steering units, and sensors. Early ECUs were independent, with each having a specific function. They formed 1:1 connections with the part of the vehicle they were controlling. Later, ECUs gradually began to cooperate with each other; for example, body, comfort, chassis, powertrain, and infotainment ECUs began to exchange information. This created 3-4 independent networks within a single vehicle, with wires between them to exchange data. Recently, universal ECUs have also been developed. They are structured to allow stronger collaboration from a central gateway and

The associate editor coordinating the review of this manuscript and approving it for publication was John Mitchell.

cross-functional connections. As hardware has improved, these structures have become possible; therefore, more complex functions can be implemented [1]. With functions such as advanced driver assistance system being added to vehicles, the number of ECUs installed in automobiles has continued to increase. Each ECU exchanges information with the others in the IVN. The first IVNs were built using point-to-point wiring systems; however, as more ECUs are installed in vehicles, the weight of the wiring harness, and space and cost requirements have become major disadvantages. To overcome these issues, the CAN, FlexRay, and local interconnect network (LIN) have been developed. These networks exhibit excellent performance in terms of safety, real-time response, and topology, depending on the characteristics of the vehicle. As engineers need to implement network interfaces and develop firmware or software to send and receive data from the embedded systems, while knowledge of the networks is vital, knowledge of the embedded systems development is also important to successfully implement an IVN. Motivated by this, this paper presents an IVN training course that includes content of the automotive embedded system. This course can be used to train automotive engineers working in the field to increase their understanding of IVNs.

Many embedded systems and networking training courses for students and engineers already exist [2]–[10], but most of these training courses are designed for university students. University students use embedded systems provided in class to test input/output and simple peripherals, such as serial communication devices. The advantage of these courses taken by university students is that they are well-structured, because grades are awarded according to the completion of assignments. However, traditional embedded systems and networking courses do not allow students who are unfamiliar with computer architecture and language to attend class, and can only be taken after completing the prerequisite courses. With these types of training courses, automotive engineers in other majors (especially mechanical) may have difficulty understanding the guidelines for embedded systems.

Therefore, courses have been developed to enable students and non-experts who are unfamiliar with electronic, electrical, and computer engineering to understand and develop embedded systems [11]–[13]. Vanderbilt University developed an embedded software and systems curriculum that combines elements from traditional electrical, mechanical, and computer engineering courses [11]. The most exciting aspect of this course is that it is based on a visual modeling language, and works on the Web. The Vanderbilt University curriculum provides students with a high-quality course that covers microcontrollers, real-time systems, and distributed systems. In addition, the curriculum includes modules on embedded systems for non-engineers and new students [12], [13]. Sungkyunkwan University uses Lego Mindstorms to teach the concepts behind embedded systems to new students who do not have a fundamental understanding of engineering. The experience of working with embedded systems in toys prepares students for classes on more advanced subjects, and helps them to understand the practical applications of embedded systems. This course also uses smartphones and social networking services to enhance communication with students in the class, increase student participation, and improve class outcomes [14]. The more direct the participation of students, the better their understanding of embedded systems.

Most engineering undergraduate students take classes to improve their understanding of networks. However, few courses cover larger networks and experiments. To improve this situation, some classes have experimented with networks that consist of multiple nodes. Analyzing messages from multiple nodes using Ethernet switches helps students understand the characteristics of networks [7]. Using simulation, students can construct and experiment with large networks that would be difficult to actually construct in class [8]. However, there are very few training courses that deal with specialized networks, such as IVNs. Although there are network training courses to analyze CAN frames using CAN FD [15], these are limited to the control bus of CAN FD and do not include IVN content.

C. INNOVATIONS FROM THIS PAPER

1) OPTIMIZED TRAINING FOR AUTOMOTIVE MANUFACTURER ENGINEERS

It is reasonably straightforward to develop a course outline to teach university students who all have similar proficiency and knowledge in one class. In contrast, creating courses for students with different majors and different levels of understanding is a hard task.

At first, we prepared our curriculum assuming the students would have similar proficiency and knowledge. Contrary to the perception that automotive engineers are skilled in every single area of vehicles, we noticed that they actually tend to have different majors and levels of programming skill, and only the few automotive engineers who are good at C and understand embedded systems could successfully carry out the experiments. Most of the engineers taking the course struggled to implement the IVNs that required programming embedded systems using C. As such, we felt the need to create a curriculum that automotive engineers, especially those from automobile manufacturing companies, could readily understand.

Automobile manufacturers usually do not work on every detail directly; instead, they often collaborate with parts manufacturers to produce the necessary components. As such, it is not necessary for engineers to grasp every single detail, only the information on request to parts manufacturers and the information to assemble the parts received from the manufacturer of the parts are needed. The engineers do not necessarily need to know all the theories underpinning how a component works; it is sufficient to teach them just what they need to know when designing in the field. We first found the common denominators in basic knowledge about IVNs held by these automotive engineers. Based on this, our course uses a board that is configured to have a communication structure similar to those found in real vehicles, but their need to directly code

FIGURE 1. IVN training at Hyundai motor group.

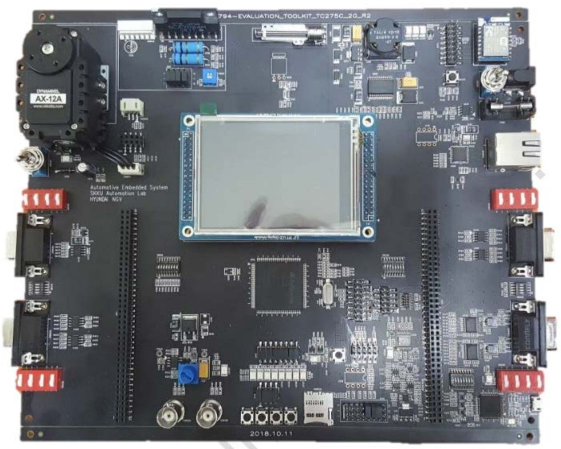


FIGURE 2. The automotive embedded system for IVN training.

in C language was minimized, and replaced by an easy-to-use GUI. We had students practice using the GUI by changing parameters and modules.

After the course, there was a lot of feedback on how helpful this was for their actual work, which led to more engineers taking the lectures, allowing us to iteratively revise the lecture curriculum from their feedback. Figure 1 shows our IVN training course for the Hyundai Motor Group.

2) A TRAINING APPROACH THAT INTRODUCES TOOLKITS AND GUIs

The proposed IVN training course provides automotive engineers with embedded systems and basic software knowledge, enabling them to implement IVN. We also provided software with an easy-to-use GUI that can be controlled from a PC. Each automotive engineer taking the course was provided with a toolkit to configure one network node, and several people worked together in a group to configure the entire network. This allowed the students to configure the necessary network topology so that they could move on to studying how to analyze the bus status. This approach significantly helped the automotive engineers' understanding of IVNs. By allowing automotive engineers students to work with the embedded system's firmware using a GUI, they were able to modify the software and network drivers in the embedded system without knowledge of C.

In the first part of the course, they were guided by testing the communication status according to the physical state of the bus by changing the CAN settings, CAN-FD bit timing, ID filtering, and terminal resistors. By making it easy to change the configuration of the IVN, automotive engineers have come to understand the characteristics of the CAN. This process was conducted in the same way for Ethernet. In addition, because there was no need for direct coding, it was possible to design various communication structures, and they implemented the various communication technologies required for IVNs in a simple manner, helping them understand the meaning of key parameters. This GUI-based training approach enhanced the completeness of the experiments conducted compared with traditional C-language-based training, where a large amount of effort is directed toward learning and writing correct code. However, with this approach, the level of difficulty was too low, which meant that they concentrated less. To overcome this problem, the proposed final IVN course combines coding in C with GUI-based training, and involves the design of a gateway through group experiments to increase achievable possibilities. They designed gateways in the IVN, and came to understand network efficiency and the flow of data by creating their own databases and routing tables.

This paper details our approach to teaching engineers IVN concepts through experiments using embedded systems. Furthermore, the course includes content and training methods to improve the quality of the IVN training course. Section II provides an overview of the embedded systems used in the IVN course, the software used, and the training sequence. Section III details what the IVN course covers; Section IV describes the CAN, Ethernet, and Gateway practices, respectively; and Section V describes the methodologies of IVN training, and discusses the advantages and disadvantages of code-based and GUI-based training and group experiments. Section VI describes the evaluation methods and evaluation results, while Section VII describes the responses of automotive engineers and companies to this IVN training. Finally, Section VIII presents our conclusions.

II. THE IVN TRAINING TOOLKIT

In this paper, we proposed an IVN training course that provides automotive engineers with a toolkit that includes an Infineon AURIX series TriCore TC275 based embedded system and basic software to allow easy experimentation with network functions. This toolkit allows students to conduct experiments involving CAN, CAN FD, Ethernet, and gateway applications. Figure 2 shows the toolkit, which combines embedded systems and peripheral tools for use with IVNs.

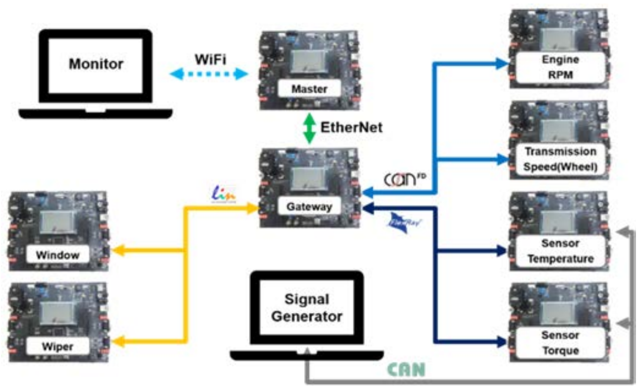


FIGURE 3. IVN demonstration of the virtual vehicle network.

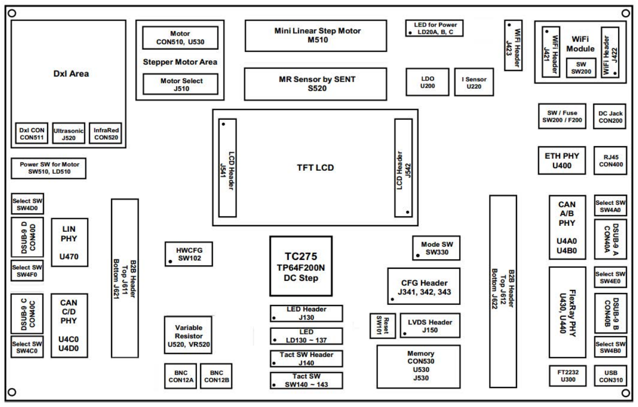


FIGURE 4. IVN training toolkit structure.

A. SYSTEM OF IVN TRAINING TOOLKIT

The IVN toolkit consists of an embedded system, cables, a universal serial bus (USB) to the CAN device (USB2CAN), basic software, and user software. The hardware is used to represent ECUs found in a vehicle. It can be used to configure CAN, CAN FD 4 channels, FlexRay 2 channels, LINs, and Ethernet 1 channels, in addition to being able to implement multichannel ECUs and gateways without performance degradation using a multicore processor. Consequently, various IVN topologies can be configured, and Figure 3 shows that a virtual automotive system can be realized by assigning each ECU to several individual embedded systems. In addition, for the convenience of automotive engineers taking the course, a GUI was configured for the embedded system so that the status of the system could be easily checked. The IVN training described in this study is mainly based on using CANs, CAN FD, and Ethernet. Therefore, the toolkit includes Ethernet cables, CAN cables that can be adjusted at 1 meter intervals, and USB2CAN devices to transmit CAN messages from a PC. Figure 4 shows the structure of the IVN training toolkit.

1) COMMUNICATIONS

The communication part of this toolkit includes four-channel CAN/CAN-FD, one-channel FlexRay, one-channel EtherNet, one-channel LIN, one-channel WiFi, and one-channel UART. In terms of physical connections, a one-channel USB connector for UART and a one-channel RJ45 connector for EtherNet are included. A four-channel D-sub connector for CAN/CAN-FD, FlexRay, and LIN is also included. The D-sub connector is designed to select the correct line using a Y-Selector, and so that resistors can be easily detached/attached, and the case considered where during CAN/CAN-FD communication, multiple nodes are connected to the BUS.

2) DIGITAL IO

IO tests can be performed using eight LEDs and four push-buttons; the board in the toolkit is configured to connect all digital IO pins using a header socket, if necessary.

3) ANALOG INPUT

The toolkit can detect various analog signals, and includes a 2-channel BNC connector, 1-channel variable resistor, 1-channel infrared sensor, and 1-channel current sensor to measure the total current consumption of the device.

4) DISPLAY

It is possible to output images or text using the TFT LCD screen, and the display can also be configured for any touch-based GUI using its touch screen capability. The LCD screen had a resolution of 320×240 pixels.

5) ACTUATORS

The servo motor and mini linear stepper motor actuators can be operated, while unipolar stepper motors can be operated by the additional drivers and connectors that are included.

6) SENSORS

The toolkit includes 1 infrared sensor and 1 ultrasonic sensor. The infrared sensor outputs an analog signal, whereas the ultrasonic sensor outputs a pulsed signal.

B. TOOLS USED IN IVN TRAINING

Automotive engineers sent and received CAN, CAN FD, and Ethernet messages from their PCs, and programmed the embedded system using our GUI or compilers. Programs for the embedded system were coded and compiled using HighTec's toolchain (a free TriCore entry toolchain), which was downloaded using the included programmer. Moreover, we provided basic software for the embedded system programming. The basic software was divided into an application layer, system library layer, and device driver layer to help students code conveniently in the application layer as much as possible. Then, a PC was used to provide the GUI SW for Ethernet transmission and reception, while Wireshark was used to easily analyze the Ethernet frames. During CAN training, they could send/receive CAN messages using the provided USB2CAN interface using software from PEAK (PCAN view). The engineers used these tools to complete

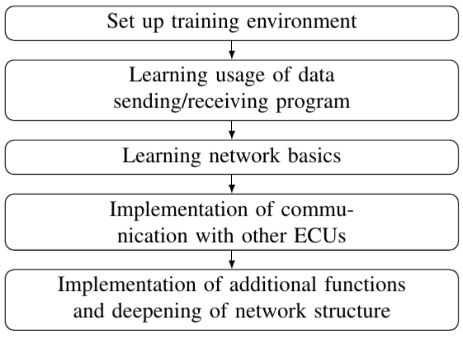

Set up training environment
Learning usage of data sending/receiving program
Learning network basics
Implementation of communication with other ECUs
Implementation of additional functions and deepening of network structure

FIGURE 5. The architecture of the training flow.

IVN training experiments without requiring extensive prior knowledge of programming languages.

C. IVN TRAINING COURSE CURRICULUM

The IVN training consisted of three parts: CAN, Ethernet, and gateway. The training course was conducted in the following order: CAN → Ethernet → gateway. If students want knowledge of only one course, there is no problem in taking it separately, but for the purpose of this education, it is recommended to take all three courses. These practices first teach students how to control the toolkit using the GUI. Next, training on the basic usage of data sending and receiving the programs necessary for network control is conducted, and training on the basic structure of each network is performed. After that, the course consists of an approximate order of training on the implementation of communication with the outside using the network, and training on the use of additional functions and in-depth content on the network structure. Figure 5 shows the general flow of courses. Tables 1, 2, and 3 show the details of each course.

1) CAN

On the first day of theory class, the CAN course introduced the role of CANs in IVNs, while on the second and third days, it introduced the transceiver, frame, bit timing, and synchronization elements. On the fourth day, engineers completed their theoretical education by analyzing the CAN FD protocol, and comparing LINs with CANs. They then used CANs directly in the experiments to better understand their characteristics. In practices, they studied the meaning of key parameters, such as the bit rate, bit timing, and loop delay offset through experiments aimed at regulating a CAN controller embedded in the USB2CAN, and used the microcontroller unit to implement a CAN between a PC and the embedded system. In the second part of the course, the CAN training emphasizes that all CAN controllers on the bus can choose which messages to receive by ID filtering. In the third part of the course, to understand the concept of signals (data units in CAN databases) in CAN messages, they read sensor values and performed signal transmit/receive experiments to control the motor. In the fourth part of the course, they analyzed the relationship between communication cycles, speed, and ID priority by experimenting with starvation that can occur according to the CAN busload. Finally, they verified the flexible data rate of CAN FD through an experiment that compared the speeds of CAN and CAN FD.

TABLE 1. IVN training course: CAN.

	CAN
Practice 1	Download practice files
Practice 2	How to use PCAN-View
Practice 3	TC275 programming
Practice 4	TC275 CAN I/F
Practice 5	Communication between PC <-> ECU
Practice 6	Communication between ECU <-> ECU
Practice 7	Multi-communication between ECU <-> ECU
Practice 8	CAN ID filter
Practice 9	Get ADC value
Practice 10	Set servo motor position
Practice 11	Starvation state
Practice 12	Starvation test
Practice 13	Multi-Bus
Practice 14	CAN communication & CAN FD speed comparison

TABLE 2. IVN training course: Ethernet.

	Ethernet
Practice 1	Download practice files
Practice 2	How to use Wireshark
Practice 3	TC275 Ethernet I/F
Practice 4	Ethernet-CAN conversion
Practice 5	Ethernet switch
Practice 6	VLAN security
Practice 7	Ethernet backbone

2) ETHERNET

On the first day of the Ethernet course, the role of Ethernet in IVNs, in contrast to fieldbuses such as CAN, CAN FD, and FlexRay, was introduced. The second day covered the data link layer, data frames, and error checks in the OSI 7 layers. On the third day, the physical layer and IEEE standards were covered. On the fourth day, they learned about the synchronization method and scalable service-oriented middleware over IP (SOME/IP), effectively completing their theoretical education. They then experimentally investigated the characteristics of Ethernet. In practices, students used Wireshark to analyze Ethernet frames, and understand the Ethernet interface through the experience of controlling the Ethernet module embedded in the microcontroller unit. In the second part, they learned how to use switches for multiple Ethernet connections, to understand the roles of the destination MAC and source MAC. In the third part, they learned the importance of time synchronization between nodes during Ethernet communication, and tested synchronization using the precision time protocol (PTP). In the fourth part, they investigated the characteristics of virtual local area networks (VLANs), and tested them for security. Finally, an experiment was conducted in which the students connect all the embedded systems in the class through an Ethernet backbone structure, and reorganize the IVN structure.

TABLE 3. IVN training course: Gateway.

	Gateway
Practice 1	Download practice files
Practice 2	Frame Based Routing
Practice 3	Signal Based Routing
Practice 4	PDU Based Routing
Practice 5	ECU Reprogramming

3) GATEWAY

The gateway course consisted of one day of theoretical education, and one day of experimental training. In the theory part, they studied the structure of gateways, ECU reprogramming, and VLANs. Subsequently, they constructed a gateway, and investigated its characteristics. First, they studied the concept of routing through a frame-based routing experiment. And they designed routing tables and signal tables on their own to understand the structure of the gateway and the role of the domain controller unit in signal-based routing and protocol data unit (PDU)-based routing experiments. Finally, they conducted an ECU reprogramming task to understand the relationship between data transfer time, timing, and stability.

III. WHAT IVN TRAINING COVERS

This section introduces some of the most important experiments per components, and explains how each was conducted.

A. CAN

First, the students learned how to use a USB2CAN to send and receive CAN messages from a PC. There are several components of CAN, and their differences are described below. This helps the students understand how to exchange CAN messages between PCs and embedded systems.

1) CAN TRANSCEIVER

The transceiver circuit was explained to automotive engineers to enable them to understand CAN hardware configurations. The embedded system was designed such that when constructing various topologies, terminal resistors could easily be attached and detached.

2) CAN CONTROLLER

Students learned how to configure and use CAN controllers in the PC and microcontroller units, and came to understand the key parameters and features of the CAN. In this experiment, they studied the relationship between the cable length and bit timing. Second, through CAN filtering experiments, they learned the concept of receiver selection in the CANs. This helped them understand that when nodes are added or removed, they do not need to modify the databases of other ECUs, as opposed to instances of schedule-based communication (FlexRay and LIN).

3) CAN DATABASE

Students learn about CANs through practical interactions with a CAN database. Understanding the concept of signals in CAN databases is crucial for practical training. Therefore, the concepts of a start bit, length, factor, and offset of signal in CAN data frames were introduced, and experiments were conducted to update data using a signal unit.

4) CAN BUS

Students studied the topology and domain concepts of IVNs by describing the CAN bus configuration. The experiment was performed in groups of four people, who intentionally caused starvation by sending messages with different IDs to a bus in a short cycle. This helps students recognize the importance of CAN ID priority and database design.

B. ETHERNET

The Ethernet experiments focused on the Ethernet frame structure and changes in communication according to the MAC address and ether type.

1) ETHERNET CONTROLLER

Students learned how to configure the Ethernet and embedded system on a PC to send and receive Ethernet messages. We provided data link layer software that could send and receive Ethernet messages from a PC. Additionally, we described the main parameters for configuring the Ethernet module of the embedded system. Then, an Ethernet message was exchanged between the PC and the embedded system, and the frame was analyzed using Wireshark. This lesson helped them understand and classify MAC address, ether type, and data frames.

2) SYNCHRONIZATION

Synchronization between nodes during communication is crucial in real-time systems. Therefore, we introduced automotive engineers to IEEE1588 PTP, and highlighted the need for synchronization. Additionally, the students conducted experiments where they synchronize between two nodes using a timer and a receive interrupt.

3) ETHERNET SWITCH

We conducted experiments to configure the Ethernet topology in various ways, and observed how changes to the MAC address and Ethernet type affected the system. Students formed a team of seven people, and performed the Ethernet switch experiments. These improved their understanding of MAC addresses, and their proficiency in sending and receiving messages across domains according to a given MAC address.

C. GATEWAY

The gateway experiments improved the students' understanding of each routing method and message database. In addition, through reprogramming experiments using CAN, they became familiar with the factors to consider in ECU reprogramming. To learn CAN and Ethernet independently, the aforementioned CAN and Ethernet courses may be sufficient. However, since our purpose was to learn about real

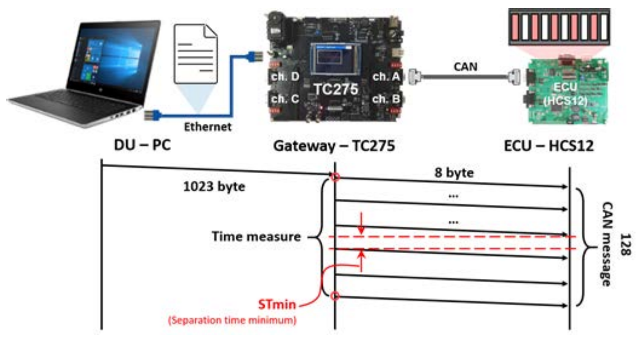


FIGURE 6. Signal-based routing in gateway training.

vehicle development, we also included the process of constructing and testing an environment similar to a real vehicle by dealing with the gateway.

1) ROUTING

The students were assigned to three-person teams, and assigned roles for gateways ECU 1 and ECU 2. In this experiment, frame-based routing, signal-based routing, and PDU-based routing were carried out in that order, and the gateway function of domain controller units in IVNs was understood. Figure 6 shows a diagram of their signal-based routing experiment.

2) ECU REPROGRAMMING

In this experiment, emphasizes the importance of the CAN transport layer's (CAN TP) structure for reprogramming and communication stability based on the minimum separation time (STmin). Engineers then see how the STmin values can affect the reprogramming success. The total reprogramming time for this process was calculated by measuring the start and end times of the transmission. Furthermore, we added the minimum waiting time in STmin after transmitting the CAN message. In this experiment, the students compared reprogramming time and success in terms of STmin duration. The students used a PC to send data (a program for the target embedded system) to the gateway via Ethernet, and the gateway downloaded the SW to the ECU via the CAN. The students learned that the longer the STmin duration, the longer the reprogramming time. The shorter the STmin duration, the lower the reprogramming stability.

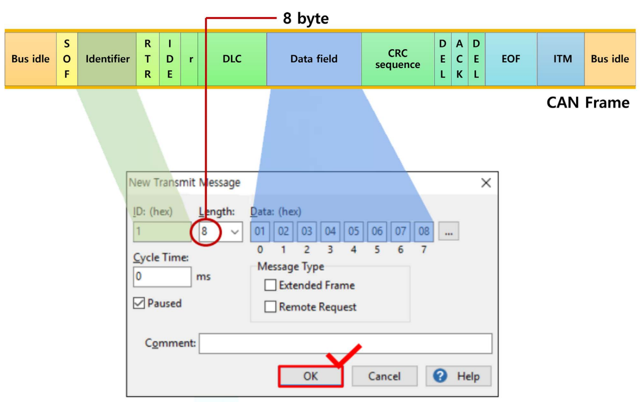
IV. PRACTICES

This section introduces the contents of each practice. The curriculum proceeds in the order of CAN networks → Ethernet networks → Gateway management.

A. PRACTICES FOR CAN NETWORKS

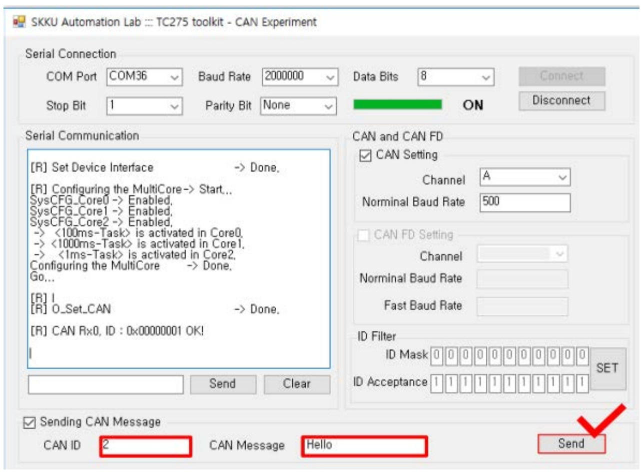
1) PRACTICE 1: DOWNLOAD PRACTICE FILES

This file briefly informs how to set up the practice environment. It provides a basic explanation of using the toolkit, and teaches students how to use the practice files.



ID: (hex) 1, Length: 8, Data: (hex) 01 02 03 04 05 06 07 08 (0 1 2 3 4 5 6 7), Cycle Time: 0 ms, ☑ Paused, ☐ Extended Frame, ☐ Remote Request

FIGURE 7. Generating CAN message.



COM Port: COM36, Baud Rate: 2000000, Data Bits: 8, Stop Bit: 1, Parity Bit: None, ON

```
[R] Set Device Interface              -> Done.

[R] Configuring the MultiCore-> Start...
SysCFG_Core0 -> Enabled.
SysCFG_Core1 -> Enabled.
SysCFG_Core2 -> Enabled.
 ->  <100ms-Task> is activated in Core0.
 -> <1000ms-Task> is activated in Core1.
 ->   <1ms-Task> is activated in Core2.
Configuring the MultiCore      -> Done.
Go...

[R] I
[R] O_Set_CAN                         -> Done.

[R] CAN Rx0, ID : 0x00000001 OK!
```

☑ CAN Setting — Channel: A, Norminal Baud Rate: 500

☐ CAN FD Setting — Channel, Norminal Baud Rate, Fast Baud Rate

ID Mask: 0 0 0 0 0 0 0 0 0 0 0

ID Acceptance: 1 1 1 1 1 1 1 1 1 1 1

☑ Sending CAN Message — CAN ID: 2, CAN Message: Hello

FIGURE 8. GUI to send CAN ID and message.

2) PRACTICE 2: HOW TO USE PCAN-VIEW

Students practice how to control sending and receiving CAN messages. This is a practice to receive the CAN message from TC275 again after inputting the CAN ID and the message to be sent using the GUI, as shown in Figs. 7 and 8. Messages are received in ASCII and hexadecimal formats.

3) PRACTICE 3: TC275 PROGRAMMING

Students practice brief description and execution of firmware files in the toolkit for CPUs and IOs.

4) PRACTICE 4: TC275 CAN I/F

In this practice, students learn about the CAN interface firmware file prepared in advance using the toolkit including the channel setting, message ID, CAN data size, FD flexible bitrate, CAN FD setting, ID mask, ID filter, etc., which are initially set through firmware. Figure 9 shows the CAN interface structure of the TC275 board.

5) PRACTICE 5: COMMUNICATION BETWEEN PC AND ECU

After connecting to PCAN-View using the PCAN-USB FD interface, students download the practice file to the board

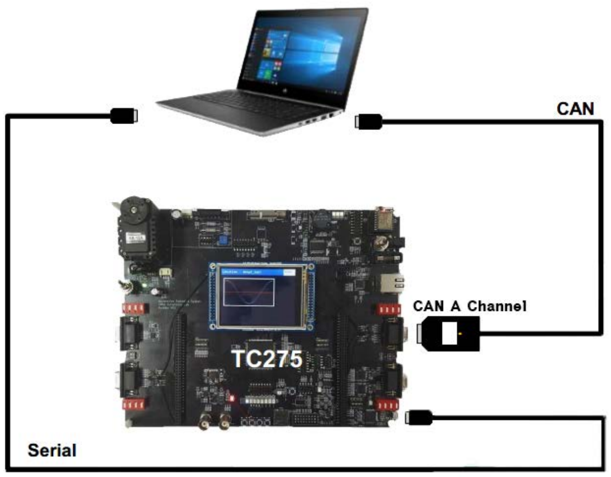
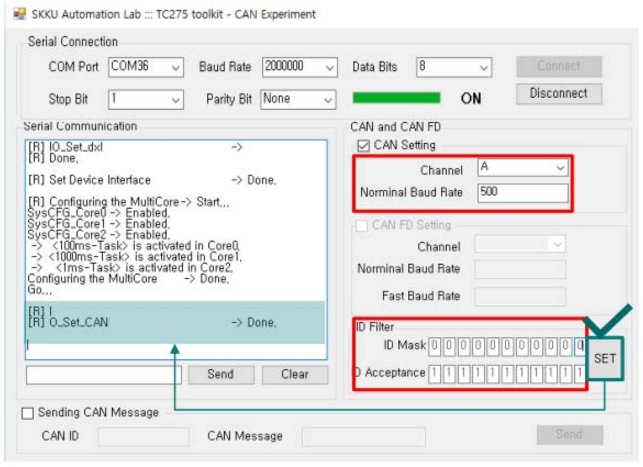


FIGURE 9. CAN interface structure of TC275.



COM Port: COM36
Baud Rate: 2000000
Data Bits: 8
Stop Bit: 1
Parity Bit: None

```
[R] IO_Set_dxl                ->
[R] Done.

[R] Set Device Interface      -> Done.

[R] Configuring the MultiCore-> Start...
SysCFG_Core0 -> Enabled.
SysCFG_Core1 -> Enabled.
SysCFG_Core2 -> Enabled.
 ->  <100ms-Task> is activated in Core0.
 -> <1000ms-Task> is activated in Core1.
 ->   <1ms-Task> is activated in Core2.
Configuring the MultiCore      -> Done.
Go...

[R] I
[R] O_Set_CAN                 -> Done.
```

☑ CAN Setting
Channel: A
Norminal Baud Rate: 500

☐ CAN FD Setting
Channel:
Norminal Baud Rate:
Fast Baud Rate:

ID Filter
ID Mask: 0 0 0 0 0 0 0 0 0 0 0
D Acceptance: 1 1 1 1 1 1 1 1 1 1 1

☐ Sending CAN Message
CAN ID:
CAN Message:

FIGURE 10. GUI to set the channels, nominal baud rate, ID mask, and ID acceptance.

using UDE STK, and set the channel, nominal baud rate, ID mask, ID acceptance, and ECU ID using the GUI, as shown in Fig. 10. Subsequently, they attempt to generate a CAN message to check if the setting is correct. The received message is displayed on the ECU LCD.

6) PRACTICE 6: COMMUNICATION BETWEEN ECU AND ECU

After setting up by connecting each ECU with a PC, communication is practiced by connecting the two ECUs with a CAN BUS, as shown in Fig. 11. Students connect two ECUs, and connect them to PCAN-View using the BUS and PCAN-USB FD interface. Subsequently, ECU A sends a message, and confirms the receipt of the message through ECU B and the trace window.

7) PRACTICE 7: MULTICOMMUNICATION BETWEEN ECUs AND ECUs

Students practice communication by connecting three or more ECUs with CAN BUS. Students connect each ECU

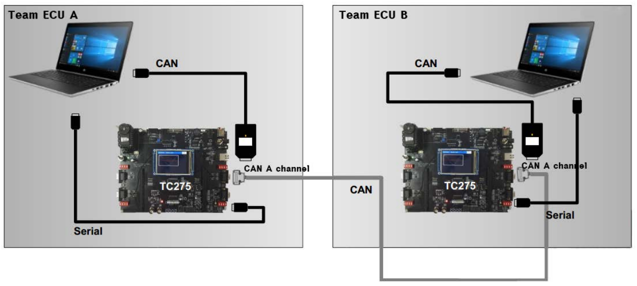


FIGURE 11. ECU-to-ECU communication practice environment.

with CAN BUS, and connect to PCAN-View using the BUS and PCAN-USB FD interface. After that, ECU A sends a message, and confirms the receipt of the message through each ECU and the trace window.

8) PRACTICE 8: CAN ID FILTER

Students practice applying ID filter through a GUI, and checking whether the filtering is successful. In this practice, the ID filter is set to allow only ID 16 to ID 31 messages in the GUI, and ID 99 messages are sent from PCAN-View. Students then check whether the corresponding ID 99 message is filtered in each ECU to which the ID filter is applied, and repeat this process for other messages.

9) PRACTICE 9: OBTAINING THE ADC VALUE

Students practice measuring the ADC value of the variable resistor in the ECU, converting it into ASCII code, and transmitting it. In this practice, they connect to PCAN-View using the PCAN-USB FD interface. They send ‘gADC_A00’ message from PC to ECU A. Then they check that the received message is displayed on the ECU LCD, which measures the ADC value of the variable resistor, converts the ADC value into a 4-digit ASCII code, and then transmits it. For example, if the ADC value is 1234, message ‘ADC_1234’ is transmitted.

10) PRACTICE 10: SET THE SERVOMOTOR POSITION

Students practice preparing and moving the servo motor position according to the message sent to the ECU. In this practice, students connect to PCAN-View using the PCAN-USB FD interface, and send ‘sDxl_A’ message from PC to ECU A to prepare to change the servo motor position of ECU, and transmit ‘A0000’ message from PC to ECU A. Students check that the received message is displayed on the ECU LCD, and that the position of the servo motor connected to the ECU has moved to position 0000. They then repeat this process for other positions.

11) PRACTICE 11: STARVATION STATE

Students practice checking the starvation phenomenon of low-priority messages through the CAN bus occupancy frequency. In this practice, an ID 100 message is generated, and the CAN message of the ID set by each ECU is transmitted

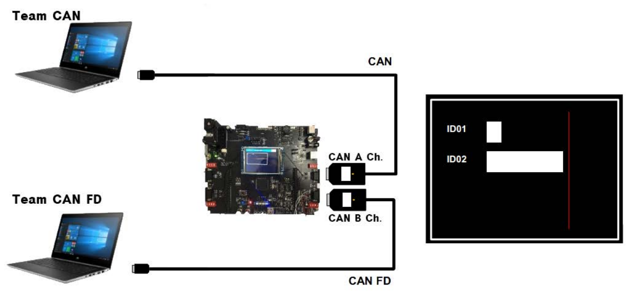


FIGURE 12. Structure of CAN and CAN FD.

in a 1 ms cycle. Students can check the current CAN BUS traffic in the BUS load window. (before sending ID 100 message -> Bus Load = 0%) By using PCAN-View, the message of ID 100 is transmitted, and the message of the ID set by each ECU is transmitted and received. In the trace window, students can check the frequency of CAN bus occupancy according to each CAN ID. The phenomenon in which the CAN BUS cannot be occupied is starvation.

12) PRACTICE 12: STARVATION TEST

Students practice control of the ECU LED in a starvation state. In this practice, students try to control the LED of each ECU using PCAN-View in situations in which starvation conditions occur. They check the result that the message of ID 99 falls into starvation, and cannot be received.

13) PRACTICE 13: MULTI-BUS

Students practice the simultaneous control of ECU using the double line topology of CAN BUS. By connecting two CAN BUSs, BUS A sends and receives messages between ECUs, and connects to the CAN Monitor with BUS B. PCAN-View uses BUS B to control the LED of each ECU, and checks the result of controlling the LEDs.

14) PRACTICE 14: CAN COMMUNICATION AND CAN FD COMMUNICATION SPEED COMPARISON

In this practice, students compare CAN and CAN FD based on the received data size in real time. The received data size is displayed in the form of a bar graph on the LCD of the ECU, as shown in Fig. 12.

B. PRACTICES FOR ETHERNET NETWORKS

1) PRACTICE 1: DOWNLOAD PRACTICE FILES

This section briefly informs how to set up the practice environment.

2) PRACTICE 2: HOW TO USE WIRESHARK

Students practice how to control the sending and receiving of messages in Ethernet. In this practice, students send a message through a GUI, and use Wireshark to check that the message is sent and received.

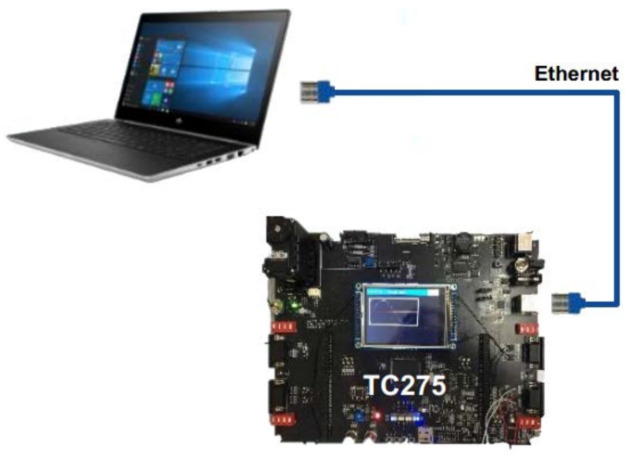


FIGURE 13. Ethernet interface structure of TC275.

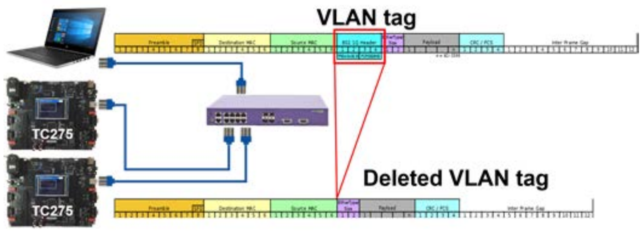


FIGURE 14. Delete VLAN tag.

3) PRACTICE 3: TC275 ETHERNET I/F

In this practice, students briefly learn about the Ethernet interface firmware file prepared in advance using the toolkit. Figure 13 shows the Ethernet interface structure of TC275.

4) PRACTICE 4: ETHERNET-CAN CONVERSION

Sequential transmission is achieved by converting the Ethernet payload into a CAN message. In this practice, students convert an Ethernet payload into a CAN message, and send it sequentially. Transmission Ethernet messages can be checked through Wireshark, and the received CAN messages can be checked through PCAN-view. The Ethernet payload is generally larger than the CAN Data Field, so the Ethernet payload is cut into 8 byte units, and transmitted sequentially.

5) PRACTICE 5: ETHERNET SWITCH

Serial communication practice through an Ethernet switch. In this practice, students confirm that serial communication works through an Ethernet switch.

6) PRACTICE 6: VLAN SECURITY

Students practice the checking of the VLAN security operation using an Ethernet switch. In this practice, a VLAN-based Ethernet payload is transmitted to the switch, and the switch deletes the VLAN tag, as shown in Fig. 14, and transmits it to the ECU connected to the transmission port assigned through VLAN. Students configure a structure in which one PC and

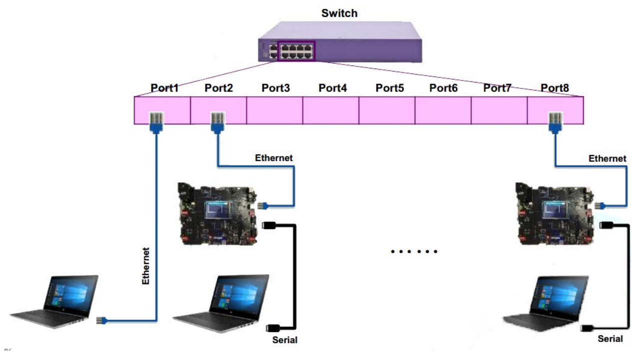


FIGURE 15. **VLAN security practice environment.**

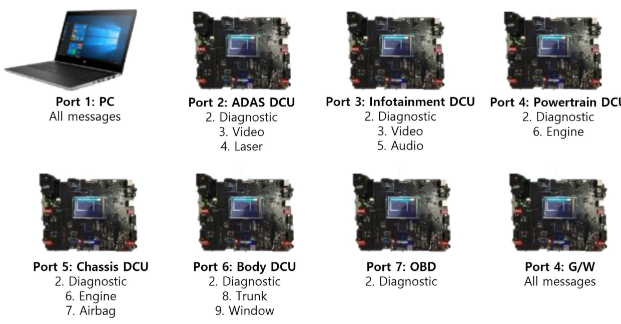


FIGURE 16. **Arrangement according to each port.**

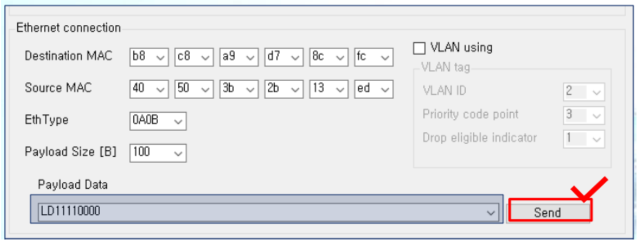
several ECUs are connected to the switch, as shown in Fig. 15. Assuming an actual vehicle development environment, each part is assigned to each ECU.

- Part number 2 (Diagnostic): 1, 2, 3, 4, 5, 6, 7, 8.
- Part number 3 (Video): 1, 2, 3, 8.
- Part number 4 (Laser): 1, 2, 8.
- Part number 5 (Audio): 1, 3, 8.
- Part number 6 (Engine): 1, 4, 5, 8.
- Part number 7 (Airbag): 1, 5, 8.
- Part number 8 (Trunk): 1, 6, 8.
- Part number 9 (Window): 1, 6, 8.

In this practice, if students want to control TC275 in part number 6 (Engine) group, they send a message from TC275 (port 4) to the switch, and from the switch to part number 6 Group TC275. If so, only TC275 connected to ports 5 and 8 will operate the motor at position 150. The students practice the same for the other groups. Figure 16 shows the arrangement according to each port, and the sorting by part number can be arranged as follows.

7) PRACTICE 7: ETHERNET BACKBONE

Students practice control after configuring a backbone network by connecting dozens of ECUs through Ethernet. The backbone architecture is based on a domain controller unit, and it is an architecture that is proposed owing to the introduction of functional domains. The backbone structure experiment is divided into three teams: No. 1 ECU to No. 7 ECU, No. 8 ECU to No. 14 ECU, and No. 15 ECU to No. 21 ECU.


Ethernet connection

Destination MAC: b8, c8, a9, d7, 8c, fc

Source MAC: 40, 50, 3b, 2b, 13, ed

EthType: 0A0B

Payload Size [B]: 100

Payload Data: LD11110000

VLAN using

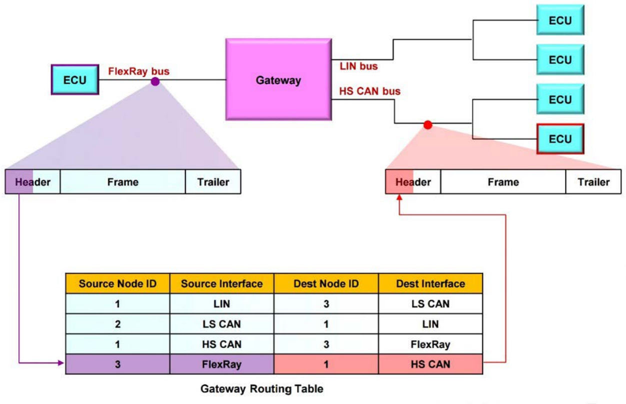
VLAN tag

VLAN ID: 2

Priority code point: 3

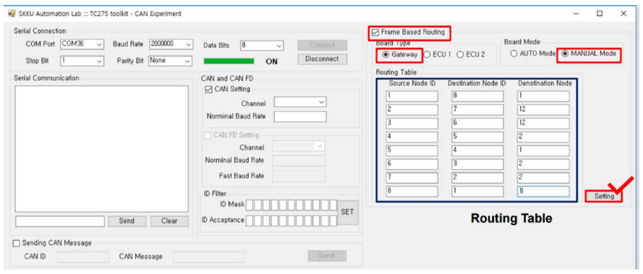
Drop eligible indicator: 1

FIGURE 17. **Transmission of ethernet message containing control message input using GUI in payload.**



Source Node ID	Source Interface	Dest Node ID	Dest Interface
1	LIN	3	LS CAN
2	LS CAN	1	LIN
1	HS CAN	3	FlexRay
3	FlexRay	1	HS CAN

FIGURE 18. **Structure of frame-based routing.**


SKKU Automation Lab ::: TC275 toolkit - CAN Experiment

Serial Connection: COM Port COM36; Baud Rate 2000000; Data Bits 8; Stop Bit 1; Parity Bit None; ON

Serial Communication

CAN and CAN FD: ☑ CAN Setting — Channel, Norminal Baud Rate; ☐ CAN FD Setting — Channel, Norminal Baud Rate, Fast Baud Rate

ID Filter: ID Mask, ID Acceptance

☐ Sending CAN Message: CAN ID, CAN Message

☑ Frame Based Routing

Board Type: [✓ Gateway] ECU 1 ECU 2

Board Mode: AUTO Mode [✓ MANUAL Mode]

Routing Table

Source Node ID	Destination Node ID	Denstination Node
1	8	1
2	7	12
3	6	12
4	5	2
5	4	1
6	3	2
7	2	2
8	1	1

FIGURE 19. **GUI for entering the routing table.**

If one team wants to control the TC275 Toolkit of another team, they set the source MAC address and destination MAC address, and input the control message. The Ethernet messages are transmitted using the GUI, as shown in Fig. 17.

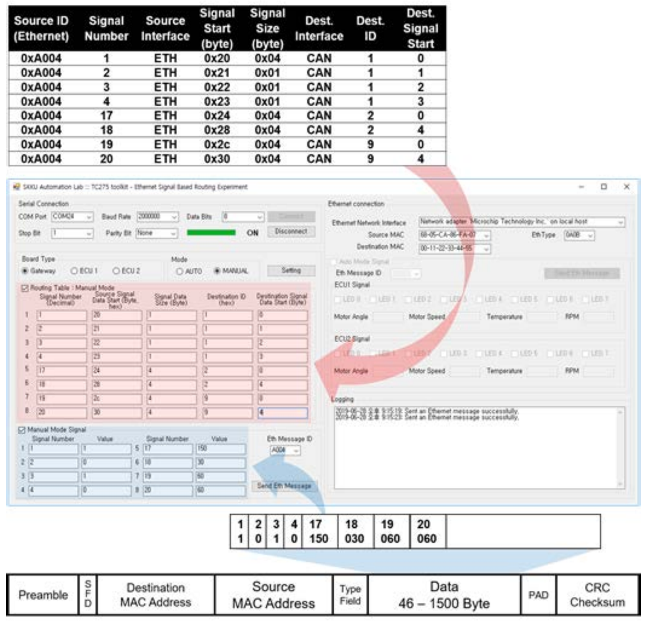
C. PRACTICES FOR GATEWAY MANAGEMENT

1) PRACTICE 1: DOWNLOAD PRACTICE FILES

A brief description of how to set up the practice environment is provided.

2) PRACTICE 2: FRAME-BASED ROUTING

Students practice routing frames from source to destination networks. Figure 18 shows that this is 1:1 frame forwarding, and Figure 19 shows that the routing table is set up through the GUI. This method is primarily used for routing between the same protocols. In the case of routing between different protocols, there is the possibility of inefficiency. A frame is a



Source ID (Ethernet)	Signal Number	Source Interface	Signal Start (byte)	Signal Size (byte)	Dest. Interface	Dest. ID	Dest. Signal Start
0xA004	1	ETH	0x20	0x04	CAN	1	0
0xA004	2	ETH	0x21	0x01	CAN	1	1
0xA004	3	ETH	0x22	0x01	CAN	1	2
0xA004	4	ETH	0x23	0x01	CAN	1	3
0xA004	17	ETH	0x24	0x04	CAN	2	0
0xA004	18	ETH	0x28	0x04	CAN	2	4
0xA004	19	ETH	0x2c	0x04	CAN	9	0
0xA004	20	ETH	0x30	0x04	CAN	9	4

FIGURE 20. GUI SW for signal-based routing.

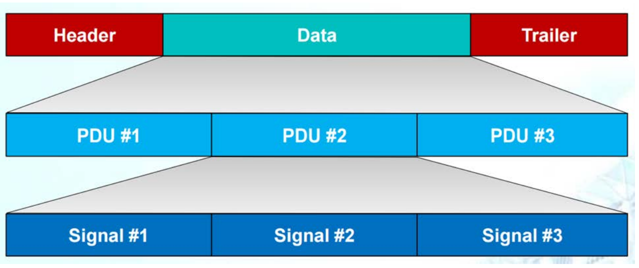


FIGURE 21. Data structure of the frame.

protocol-dependent unit, and additional conversion processing of the frame is required. Therefore, this method is mainly used for diagnostic routing between CAN-based ECUs and CAN-to-CAN gateway routing.

3) PRACTICE 3: SIGNAL-BASED ROUTING

Students practice routing signals from the source network to the destination network. This method is used when frames are composed of multiple signals. Students compose a frame with an LED control signal (1 byte), motor control signal (4 byte), and LCD output control signal (4 byte). The signal from the application is filtered by a COM module. Figure 20 presents a method for creating routing tables using GUI SW in a signal-based routing experiment.

4) PRACTICE 4: PDU-BASED ROUTING

Students practice by placing PDUs with routing using a user-defined routing table to configure Ethernet messages. PDU-based routing is useful for protocols that support frame lengths longer than CAN (8-bytes). Ethernet is 64-byte, and the Ethernet frame can contain multiple PDUs, as shown in Fig. 21.

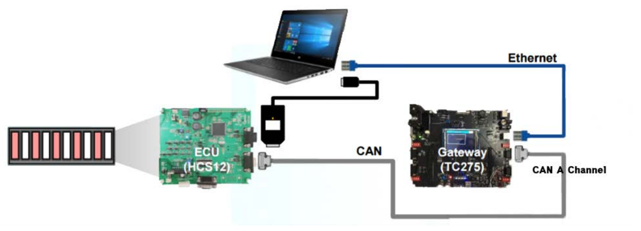
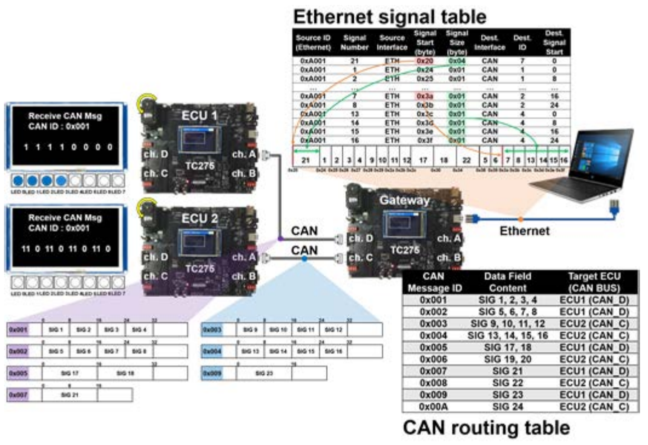


FIGURE 22. Structure of ECU reprogramming.



Ethernet signal table

Source ID (Ethernet)	Signal Number	Source Interface	Signal Start (byte)	Signal Size (byte)	Dest. Interface	Dest. ID	Dest. Signal Start
0xA001	21	ETH	0x20	0x04	CAN	7	0
0xA001	1	ETH	0x24	0x01	CAN	1	0
0xA001	2	ETH	0x25	0x01	CAN	1	8
...	...	...	...	...	...	...	...
0xA001	7	ETH	0x3a	0x01	CAN	2	16
0xA001	8	ETH	0x3b	0x01	CAN	2	24
0xA001	13	ETH	0x3c	0x01	CAN	4	0
0xA001	14	ETH	0x3d	0x01	CAN	4	8
0xA001	15	ETH	0x3e	0x01	CAN	4	16
0xA001	16	ETH	0x3f	0x01	CAN	4	24

CAN Message ID	Data Field Content	Target ECU (CAN BUS)
0x001	SIG 1, 2, 3, 4	ECU1 (CAN_D)
0x002	SIG 5, 6, 7, 8	ECU1 (CAN_D)
0x003	SIG 9, 10, 11, 12	ECU2 (CAN_C)
0x004	SIG 13, 14, 15, 16	ECU2 (CAN_C)
0x005	SIG 17, 18	ECU1 (CAN_D)
0x006	SIG 19, 20	ECU2 (CAN_C)
0x007	SIG 21	ECU1 (CAN_D)
0x008	SIG 22	ECU2 (CAN_C)
0x009	SIG 23	ECU1 (CAN_D)
0x00A	SIG 24	ECU2 (CAN_C)

CAN routing table

FIGURE 23. Reprogramming experiment.

5) PRACTICE 5: ECU REPROGRAMMING

This practice sends reprogramming data through the Ethernet-CAN gateway. Figures 22 and 23 show the structure and experimental process of the ECU reprogramming, respectively. This involves learning how to transmit with the Ethernet backbone in mind. When the PC transmits Ethernet-based reprogramming data, which turns the LED on the board into odd units, the gateway sets the minimum separation time value using the GUI, analyzes the received frame type, and converts it into a CAN payload. The students measure the time spent in reprogramming. Subsequently, the CAN reprogramming data are transmitted to the ECU to check the LED operation and the reprogramming result.

V. METHODOLOGIES FOR IVN TRAINING

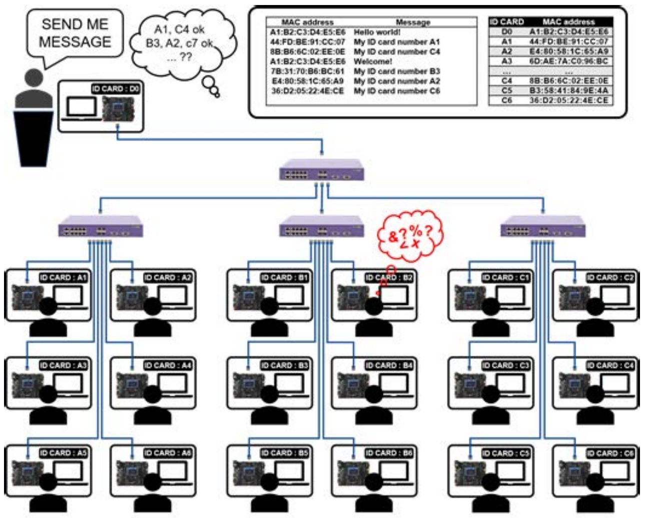
This course allowed learners with different prior knowledge to gain hands-on experience by employing toolkits and GUIs to make it easier to understand IVNs.

A. TWO-WAY CONFIGURATION

Using the GUI alone helped students in their experiments, but the experiments became very easy, so there were problems relating to poor concentration. Therefore, simple programming was introduced through C coding, and GUI was introduced for complex parts. In the proposed IVN training course, the teaching assistant prepared a program file (.elf) for all complicated applications in advance, before the engineers were tasked with changing the configuration using the GUI SW on a PC. Except for practices of observing the firmware default settings for each interface, all the other experiments were GUI-based experiments. To allow students to perform complicated experiments in a limited time, they used the GUI SW to change the configuration, operation, and parameters of the applications. Configuring networks and systems using the GUI without having to handwrite code for complex applications, such as gateways, meant that engineers did not need to spend a lot of time dealing with errors that arose from a lack of programming knowledge that is not relevant to what they were trying to learn. Additionally, the effect of conducting the experiments in this manner was positive, because the students could focus on concepts.

B. GROUP EXPERIMENTS

Unlike undergraduate classes, IVN training courses for engineers do not set exams at the end of class, and because they are active in the field, participation in the training is low. To increase the participation of these students, we decided to divide the groups, and projected the results of the experiment on a screen in front of the classroom. Figure 24 shows an embedded system of automotive engineers connected to the same network in an Ethernet backbone experiment as an example. In the group experiments, they were pre-assigned a random ID card, and a MAC address was assigned based on that ID card number. They experimented by sending a message to the teaching assistant, and if successful, they showed their efforts on the screen in front of the classroom. Experiments of this kind were performed for all practices in which groups were divided: CAN ID filters, Ethernet switches, VLAN, Ethernet backbones, etc.



MAC address	Message
A1:B2:C3:D4:E5:E6	Hello world!
44:FD:BE:91:CC:07	My ID card number A1
8B:B6:6C:02:EE:0E	My ID card number C4
A1:B2:C3:D4:E5:E6	Welcome!
7B:31:70:B6:BC:61	My ID card number B3
E4:80:58:1C:65:A9	My ID card number A2
36:D2:05:22:4E:CE	My ID card number C6

ID CARD	MAC address
D0	A1:B2:C3:D4:E5:E6
A1	44:FD:BE:91:CC:07
A2	E4:80:58:1C:65:A9
A3	6D:AE:7A:C0:96:BC
...	...
C4	8B:B6:6C:02:EE:0E
C5	B3:58:41:84:9E:4A
C6	36:D2:05:22:4E:CE

FIGURE 24. Group experiment during IVN training.

VI. EVALUATION

A. EVALUATION METHOD

For qualitative evaluation, a self-diagnosis evaluation was conducted after the course education. The questionnaire was constructed to answer each of the 22 knowledge items before and after education at six levels (0-5), corresponding to the level of knowledge. Tables 4 and 5 show examples of the two knowledge items in the questionnaire.

In the case of the quiz for quantitative evaluation, 10 questions of a multiple-choice quiz were prepared and if they obtained 70 points or more (i.e., 7 questions or more), students were evaluated as completing the course.

B. EVALUATION RESULT

Quantitative evaluation results through quizzes are confidential information of the company, and cannot be prepared in detail. Therefore, only the case of qualitative evaluation is described below. Qualitative assessments were conducted during the last four independent training sessions. In the case of qualitative evaluation of CAN education, compared to before training, the level of knowledge increased by 6 times on average, showing an increase of 2.99 levels in Training 1, 3.42 in Training 2, 2.71 in Training 3, and 2.98 in Training 4, compared to before training. Figure 25 shows the qualitative evaluation results of CAN education. In the case of the qualitative evaluation of Ethernet education, the level of knowledge increased by 4 times on average compared to before the education, showing an increase of 2.8 levels in Training 1, 2.5 in Training 2, 2.3 in Training 3, and 2.6 in Training 4. Figure 26 shows the qualitative evaluation results of Ethernet education. The content that had the highest educational effect in the evaluation was 'the operation principle of CAN', and in the case of CAN and Ethernet, 'the operation principle of Ethernet'.

VII. RESPONSE

A. DIFFERENT APPROACHES TO THE EXPERIMENTS

While training engineers in the field, we conducted sequential training using the three different approaches discussed in the following section. We reviewed the responses to these experimental training methods approximately seven times a year over the past three years. The first two years of training consisted solely of C-based experiments. In the third year, the experiments conducted were all GUI-based. We then collected responses from automotive engineers on this approach, and also examined the completion rates of the experiments. Subsequently, we again changed the training approach, by combining the two methods, and adding group experiments. The courses described in Tables 1, 2, and 3 were compiled from this process.

1) RESPONSE TO C-BASED EXPERIMENTS ONLY

Experiments that require automotive engineers to write code are difficult, particularly for those who are unaccustomed to programming (mainly those from mechanical engineering backgrounds). Therefore, engineers had to spend hours on the experiments to understand the programming part, before even considering the IVN content. Those who completed these experiments gained a strong understanding of IVNs and embedded systems. However, some engineers were unable to complete these experiments. We added three teaching

TABLE 4. Knowledge items: What level are you on CAN communication principles?

Level	behavior indicator	Before	After
5	I can suggest a new method to avoid collisions when simultaneously sending information.	5	5
4	I can configure a CAN communication system.	4	4
3	I can set an appropriate identifier to exchange information in CAN communication.	3	3
2	I can explain the roles of the CAN controller and CAN transceiver of the CAN communication ECU.	2	2
1	I can explain the need for an in-vehicle communication system.	1	1
0	I don't know at all	0	0

TABLE 5. Knowledge items: What level are you on the automotive Ethernet physical layer principles?

Level	behavior indicator	Before	After
5	I can design for a new automotive Ethernet physical layer chip architecture.	5	5
4	I can explain the need for Ethernet at a speed of 1 Gbps in the vehicle.	4	4
3	I can explain the advantages of using the PoDL method for transmitting and powering at a vehicle camera signal.	3	3
2	I can explain why people use a pair of car Ethernet lines twisted together.	2	2
1	I can explain the difference between UTP and STP wires.	1	1
0	I don't know at all	0	0

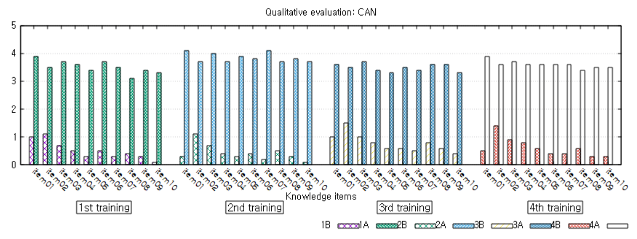


FIGURE 25. Qualitative evaluation before and after CAN training. The pre-training level of the first training is 1B, the post-training level is 1A, and the other training sessions are the same. The pre-training levels were 0.52, 0.43, 0.78, and 0.62, respectively, while the post-training levels were 3.51, 3.85, 3.49, and 3.6, respectively.

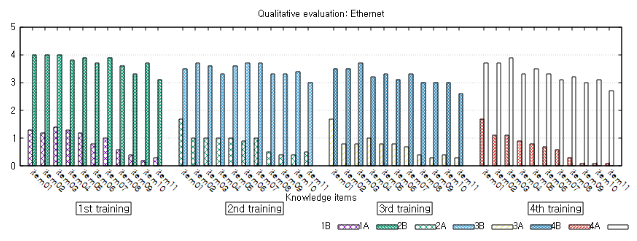


FIGURE 26. Qualitative evaluation before and after Ethernet training. The pre-training level of the first training is 1B, the post-training level is 1A, and the other training sessions are the same. The pre-training levels were 0.88, 0.85, 0.73, and 0.68, respectively, while the post-training levels were 3.73, 3.46, 3.20 and 3.32, respectively.

assistants to the class to help struggling students with the training; However, we were still not satisfied with this approach. In surveys conducted after completing the course, students answered that the education was not realistic, they could not concentrate, and it was difficult because there was a lot of theoretical content that seemed unrelated to their everyday work in the field.

2) RESPONSE TO GUI-BASED EXPERIMENTS ONLY

The theoretical content was reduced as much as possible, and the engineers used only a GUI to complete their experiments and improve their understanding of IVNs. This type of training significantly reduces the number of engineers who cannot complete their experiments. However, this also meant that the engineers did not have to concentrate on experiments to any great extent. Some engineers completed the experiments at the pace of the classes, while others did not conduct any experiments, because they thought they could conduct them at any time.

3) RESPONSE OF COMBINED EXPERIMENTS

This was the final form of the IVN training course. Because understanding of the theory was insufficient when using only the GUI, in this approach, an appropriate amount of theory training was followed by C-based and GUI-based practice. We also used Twitter to publish experimental results, which increased student participation. After the group experiment, the results were displayed on a screen at the front of the classroom, so all they became more enthusiastically involved. The results showed that the students had a high level of understanding of CAN and Ethernet, and clearly understood the important factors in implementing communications. Even in complex experiments, such as those on gateways, the students seemed to have understood the concepts of routing and databases well; they even showed good knowledge of the practical, nitty-gritty aspects of the course, such as how the timing of bits depended on the length of the CAN cable.

B. RESPONSE OF ENGINEERS

Many engineers were satisfied with this curriculum. Training took place in the company, and the curriculum was improved iteratively based on the students' feedback. During the course of learning, the engineers stated that they learned how to control single-core and dual-core systems, and as a result of the class, understood how dual-core systems are applied in the ECU networks of real vehicles. They also said that they learned both theoretical and practical aspects of CAN, Ethernet, and gateway communication that are used in ECUs in parallel. One thing in common that the engineers expressed was that, in contrast to most existing training courses that involved theory-oriented lectures, in this course, they were able to learn how the theoretical content was actually applied and implemented in a way that was very similar to real environments. Many of the students thought that the onboard APIs provided in the practice visually represented the current state of the network very well, which is a great advantage in making the course content easier to understand. Above all, they said that because a GUI was provided, even the students whose majors did not involve programming could easily follow the course; they also enjoyed the fact that they could immediately check their results. Some of them said it was nice to control the various peripherals on the board, but it added a lot of extra learning. Many students asked additional questions regarding the theory after experiments. After the training was completed and time passed, the engineers stated that practical training on CAN, Ethernet, and Gateway, which are often used in IVNs, were very helpful in their work, because what they learned could be directly applied without the need for additional study.

C. RESPONSE OF THE COMMISSIONING EMPLOYER

This course has been in operation for more than 10 years. The initial form of the course consisted of only theory, and there were various revision requirements, such as the correction of points, which were difficult to apply immediately, and strategies for problems that occurred. The employer responded that the part was adequately supplemented and modified. In particular, after applying the training course using the toolkit, we immediately received a response from the employer that they fully understood the basic core contents of the vehicle network. Therefore, the number of students taking this course at the company has continued to increase, and now every year, 500 to 800 engineers voluntarily apply for and complete this course.

VIII. CONCLUSION

This paper introduces some new teaching methods that provide effective training on IVNs to automotive engineers who are unfamiliar with IVNs, or computer programming in general, through the clever use of a training toolkit. Automotive engineers were taught about the embedded systems that can be used to build IVNs, their configuration, and the tools that can be used with these devices. Different types of experiments and their effects were conducted by students, and their feedback was gathered. When teaching using the developed toolkit, automotive engineers who were not familiar with programming were able to complete the experiments and understand important elements of the theory of IVNs. This experimental approach significantly increased the students' rate of completion, and engineers acquired a deep understanding of the most important factors of this topic relatively easily. In addition, the group experiments were effective in increasing their concentration. Engineers were able to learn about security in VLAN environments through ECU reprogramming/switch using frame, signal, and PDU-based routing/gateway in the gateway and Ethernet learning processes. In the CAN content of the course, the students learned about communication using the CAN protocol from PC to ECU or ECU to ECU, checking and controlling the board status using CAN messages, the difference between CAN and CAN-FD, priority according to the message ID, and the effect of priority on the ECU. In the multi-core part of the course, students were able to learn about task processing using the advantages of multi-core and possible problems and solutions during multicore workflows. The automotive engineers were found to have accurately understood the key elements of communication implementation in the control methods and practices of CAN, CAN FD, and Ethernet controllers. In addition, even in the complex experiments that covered content such as gateways, it appeared that the concepts related to routing and databases were effectively understood, in contrast to when students followed the original curriculum. In addition, all automotive engineers were able to actively participate in the class, as the results of the experiment were projected onto a screen at the front of the class. Based on what we have learned about implementing our IVN course using a variety of approaches, we can conclude that for effective training, it is very important to have a class structured in such a way that participants can build their understanding through active participation.

REFERENCES

[1] O. Burkacky, J. Deichmann, and J. P. Stein. (2019). *Automotive Software and Electronics 2030: Mapping the Sector's Future Landscape*. [Online]. Available: https://www.mckinsey.com

[2] J. W. Jeon, "A microprocessor course: Designing and implementing personal microcomputers," *IEEE Trans. Educ.*, vol. 43, no. 4, pp. 426–433, Nov. 2000.

[3] W. Wolf and J. Madsen, "Embedded systems education for the future," *Proc. IEEE*, vol. 88, no. 1, pp. 23–30, Jan. 2000.

[4] S. Nooshabadi and J. Garside, "Modernization of teaching in embedded systems design—An international collaborative project," *IEEE Trans. Educ.*, vol. 49, no. 2, pp. 254–262, May 2006.

[5] C.-M. Hsu and H.-M. Chao, "Constructing μ-controller teaching tool with the integration of hardware and software technology," in *Proc. Int. Conf. New Trends Inf. Service Sci.*, Jun. 2009, pp. 269–276.

[6] E. A. Qaralleh and K. A. Darabkh, "A new method for teaching microprocessors course using emulation," *Comput. Appl. Eng. Educ.*, vol. 23, no. 3, pp. 455–463, May 2015.

[7] K. A. Gotsis, S. K. Goudos, and J. N. Sahalos, "A test lab for the performance analysis of TCP over Ethernet LAN on windows operating system," *IEEE Trans. Educ.*, vol. 48, no. 2, pp. 318–328, May 2005.

[8] B. Momeni and M. Kharrazi, "Improving a computer networks course using the Partov simulation engine," *IEEE Trans. Educ.*, vol. 55, no. 3, pp. 436–443, Aug. 2012.

[9] S. Wang, R. Gu, Y. Ji, Y. Hu, Q. Guo, Y. Tan, and C. Meng, "Unified software-defined online network experiment platform for campus education," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Jul. 2016, pp. 299–302.

[10] H. Chung, S. Long, S. C. Han, S. Sarker, L. Ellis, and B. H. Kang, "A comparative study of online and face-to-face embedded systems learning course," in *Proc. 20th Australas. Comput. Educ. Conf. (ACE)*, 2018, pp. 63–72.

[11] J. Sztipanovits, G. Biswas, K. Frampton, A. Gokhale, L. Howard, G. Karsai, T. J. Koo, X. Koutsoukos, and D. C. Schmidt, "Introducing embedded software and systems education and advanced learning technology in an engineering curriculum," *ACM Trans. Embedded Comput. Syst.*, vol. 4, no. 3, pp. 549–568, Aug. 2005.

[12] S. H. Kim and J. W. Jeon, "Introduction for freshmen to embedded systems using LEGO mindstorms," *IEEE Trans. Educ.*, vol. 52, no. 1, pp. 99–108, Feb. 2009.

[13] M. Winzker and A. Schwandt, "Teaching embedded system concepts for technological literacy," *IEEE Trans. Educ.*, vol. 54, no. 2, pp. 210–215, May 2011.

[14] Y. Kim, S. Jeong, Y. Ji, S. Lee, K. H. Kwon, and J. W. Jeon, "Smartphone response system using Twitter to enable effective interaction and improve engagement in large classrooms," *IEEE Trans. Educ.*, vol. 58, no. 2, pp. 98–103, May 2015.

[15] R. de Andrade, A. A. M. Lagana, M. A. C. Guedes, J. F. Justo, A. A. M. L. N. Hodel, and F. Serafim Albaladejo, "A didactic platform to study of CAN FD bus," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Mar. 2013, pp. 318–323.

JIN YOUNG BYUN received the B.S. degree in electrical and information engineering from the Seoul National University of Science and Technology, Seoul, South Korea, in 2014. She is currently pursuing the Ph.D. degree in electrical and computer engineering with Sungkyunkwan University, Suwon, South Korea.

Her research interests include computer vision, embedded systems, and system-on-chip.

JAE WAN PARK received the B.S. degree in mechatronics engineering from Korea Polytechnic University, Siheung, South Korea, in 2016, and the Ph.D. degree in electrical and computer engineering from the School of Information and Communication Engineering, Sungkyunkwan University, Suwon, South Korea, in 2022. He is currently working with Samsung Electronics. His research interests include embedded systems, automation systems, signal processing, and real-time applications.

DO YEON KIM received the B.S. degree from Korea University, Sejong, South Korea, in 2017, and the M.S. degree in electronic electrical and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2021. She is currently working with Hyundai Motor Company. Her research interests include multi-core programming, automotive gateway, and real-time embedded systems.

CHANG YOUNG JO received the B.S. degree in electronic engineering from the University of Suwon, South Korea, in 2019, and the M.S. degree in electrical engineering from Sungkyunkwan University, South Korea, in 2021. He is currently working with Mando. His research interests include in-vehicle network (IVN) and multi-core embedded systems.

JAE WOOK JEON (Senior Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1990.

From 1990 to 1994, he was a Senior Researcher with Samsung Electronics, Suwon, South Korea. Since 1994, he has been with Sungkyunkwan University, Suwon, where he was an Assistant Professor with the School of Electrical and Computer Engineering and is currently a Professor with the School of Information and Communication Engineering. His research interests include robotics, embedded systems, and factory automation.