Design of driving interface device for electric vehicle

Thiết kế thiết bị thu thập tín hiệu điều khiển cho xe ô tô điện

Thanh Vo-Duy*, Tran Thi Minh Trang, Minh C.Ta***
(*) Center for Technology Innovation, Hanoi University of Science and Technology
(**) University of Manchester
e-Mail: thanh.voduy@hust.edu.vn, minhtrang.blue@gmail.com, minh.tacao@hust.edu.vn

Abstract

Currently, most vehicles are controlled by human interactions with steering wheel, brake and accelerator pedals. Thus it is required to establish an acquisition system to gather and process these signals from the driver; and then send them to the vehicle's controllers. In this paper, we propose a design of the driving interface device for the electric car named i-MiEV by Mitsubishi Motors. This device is constructed based on the digital signal microcontroller dsPIC30F4011 from Microchip, and communicates with the vehicle control system through Controller-Area-Network (CAN) and Serial-Peripheral-Interface (SPI) protocols.

Keywords: Electric vehicle, dsPIC30F4011, driving interface device, CAN, SPI**.**

Tóm tắt

Hiện nay, đa số phương tiện di chuyển đều do con người điều khiển thông qua tay lái, chân phanh và chân ga. Do vậy, cần phải có một bộ thu nhận và xử lí những thông tin này từ người lái và chuyển đến bộ điều khiển động cơ xe. Bài báo này đề xuất một giải pháp thiết kế công cụ nói trên cho xe ô tô điện i-MiEV của hãng Mitsubishi Motors. Thiết bị này được xây dựng trên nền tảng vi điều khiển tín hiệu số dsPIC30F4011 của Microchip, và giao tiếp với hệ thống điều khiển của xe thông qua giao thức CAN và SPI.

Từ khóa: Xe ô tô điện, dsPIC30F4011, thiết bị giao tiếp hệ thống lái, CAN, SPI**.**

1. Introduction

In the past decades, reverse engineering has been playing a vital role in developing technologies for many countries, such as Japan and China. Reverse engineering is a process of recreating a design by analyzing and partly replacing an existing product. It has also been recognized as a beneficial method for developing our domestic industries, resulting in numerous policies and investments from the Government. Automotive industry, especially electric vehicle manufacturing is one of the major application areas. Replacing electric drive system and its controller has captured the interest of many researchers since these components are the main parts of an electric vehicle. Fig.1 shows the modified power block diagram of i-MiEV by Mitsubishi Motor. Originally, this vehicle has a group of batteries, an inverter and a motor for electric drive system. However, in order to optimize the energy consumption, two ultra-capacitors 62F/125V are added to the design. The energy stored in the capacitors (when implementing regenerative braking) would then be fedbacked to the motor when the vehicle speeds up alongside with energy provided by the battery. Therefore, a bi-directional DC/DC module is required for voltage as well as energy conditioning between capacitors and DC-link of the system. It is extremely important to make the modified system compatible with the existing driving system when the driver accelerates/decelerates the vehicle.

In this paper, we propose a design of a driving interface device that collects states of accelerator pedal, brake pedal and other useful information from i-MiEV's CAN bus system which has been decrypted with a CAN bus data reader in [9]. These data are then processed and sent to Inverter and DC/DC converter as driving commands. Other information from CAN bus would be used for further display applications.

This paper is composed of 6 main parts. It begins by discussing the architecture of the system, followed by a brief overview of CAN and SPI protocols, which are the 2 communication protocols employed in this device. Section 4 and 5 proceed to describe the device's design; testing of a prototype of the device will then be discussed in section 6. The remaining part of the paper will then give a brief conclusion.

2. System configuration

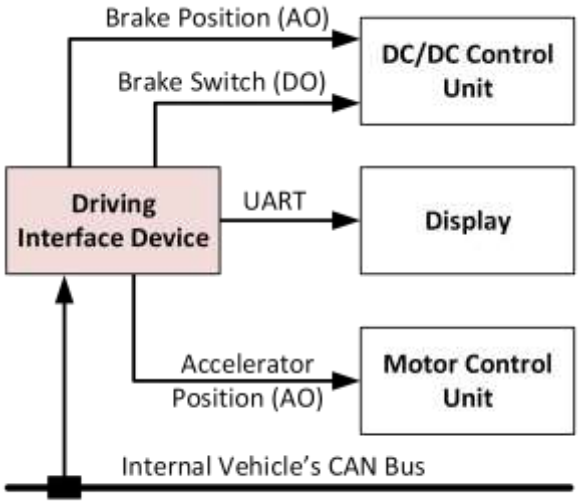


Fig. 2. Driving interface system architecture

As mentioned above, this device has to be able to communicate with the vehicle's communication link (using CAN protocol) and extract brake and accelerator data from it. It also has to be able to provide brake position analogue signal and brake switching digital signal for the DC/DC control unit, and accelerator position analogue signal for the motor control unit. For further applications, a display screen should also be implemented; for this screen, the following data is required from the CAN bus: hand brake, gear position, speed, odometer count and lights switching status.

From the above specifications, the device is designed with the configuration shown on Fig.2. The device consists of a microcontroller and 4 main blocks: CAN module block, digital to analogue converter (DAC) block, brake switching signal block and display block. The CAN block would receive the data from the vehicle's CAN bus and send them to the microcontroller for processing. These processed data would then be transmitted to the DAC block to give analogue outputs and to brake switching signal block to give digital output. Additional data regarding the vehicle's status would be sent to the display screen. The microcontroller communicates with the DAC block using SPI protocol, and communicates with the display using UART protocol. More details regarding CAN and SPI communication will be discussed on the next section.

3. Communication protocols

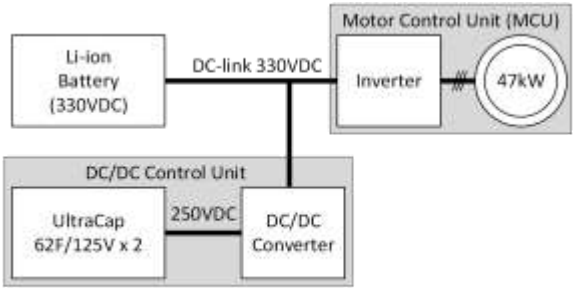


Fig. 1. Vehicle system block diagram

3.1 Serial Peripheral Interface

SPI (or Serial Peripheral Interface) is a synchronous serial communication protocol developed by Motorola. It is a master-slave protocol, using four signal lines:

- SCLK line: the clock signal sent by the master to the slaves to synchronize the transmission
- SS line: chip select signal sent by the master to choose which slaves would receive the data
- MOSI: the data line transmitted from the master to the slave
- MISO: the data line transmitted from the slave to the master

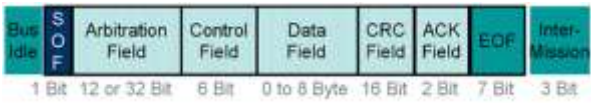


Fig. 3. CAN data frame

To initiate a transmission, the slave has to be selected by pulling the SS signal of the corresponding slave low. At each clock pulse of the SCLK signal, one bit is transmitted from the master to the slave on the MOSI line, and one bit is transmitted from the slave to the master on the MISO line [2].

Each slave requires a SS line from the master. In the situation when there are multiple slaves and there are not sufficient SS lines from the master to drive all the slaves, daisy-chain configuration is required. In this type of configuration, only one SS line is required from the master, and this line would drive all the slaves. However, only one slave would be connected directly to the MOSI line of the master, all the subsequent slaves would be connected to form a chain and would receive input from its preceding slave [3].

3.2 Controller Area Network

CAN (or Controller Area Network) is a serial communication protocol first introduced in 1986 by BOSCH. CAN differs from other protocols by its data routing method: in a CAN bus, messages are content-addressed rather than node-addressed. Every message has an identifier that is unique throughout the entire network. Nevertheless, the value of the identifier does not necessarily indicate the destination of the messages but the meaning of the data they carry [8]. CAN is a multicast system, messages travel to all nodes in the network; therefore message identifier filtering is used so as the nodes would only respond to the messages that are of interest. Moreover, the message identifier also determines the priority of that message transmission: messages with lower ID would have higher priority than those with higher ID; or in other words, among the pending messages, the message with the lowest ID value wins the bus.

A CAN node consists of four layers: physical layer, transfer layer, object layer and application layer. In this particular vehicle, the physical layer is designed using high speed CAN standard (ISO 11898-

2) with two wires (CAN_H and CAN_L) terminated with a resistor of 120Ω to prevent signal reflection [1]. All nodes connected to the CAN bus must have the same baud rate in order to communicate with other nodes. The maximum baud rate specified by CAN specification is 1Mbit/sec; however, for better stability and reliability, the common baud rate in automotive is 500kbit/sec. In CAN protocol, the nominal bit time is divided into 4 smaller segments: synchronization time, propagation time, phase segment 1 and phase segment 2. Each segment again consists of a programmable number of time quanta (T_q), except for synchronization time, which is fixed at one time quantum.

$$T_q = \frac{1/\text{baudrate}}{number\ of\ time\ quanta\ in\ a\ bit\ time} \quad (1)$$

CAN messages are sent in four types of frames (data frame, remote frame, error frame, overload frame) and two formats (standard and extended). The structure of a CAN standard message in data frame (which is the type of CAN message studied in this paper) is shown in Fig.3 [10].

A complete data frame message comprises of the following bit fields:

- Start of frame: marks the beginning of a message by a dominant bit
- Arbitration field: contains the message's 11-bit identifier and remote transmission request bit
- Control field: contains a 4-bit wide data length code
- Data field: contains data to be transmitted, maximum 8-byte wide
- CRC field: contains a CRC sequence followed by a CRC delimiter
- ACK field: contains ACK slot bit and ACK delimiter bit
- End of frame: marks the end of a message by 7 recessive bits [8]

Having discussed about the communication protocols used in the device, the next sections will move on to describe the design of the device's hardware.

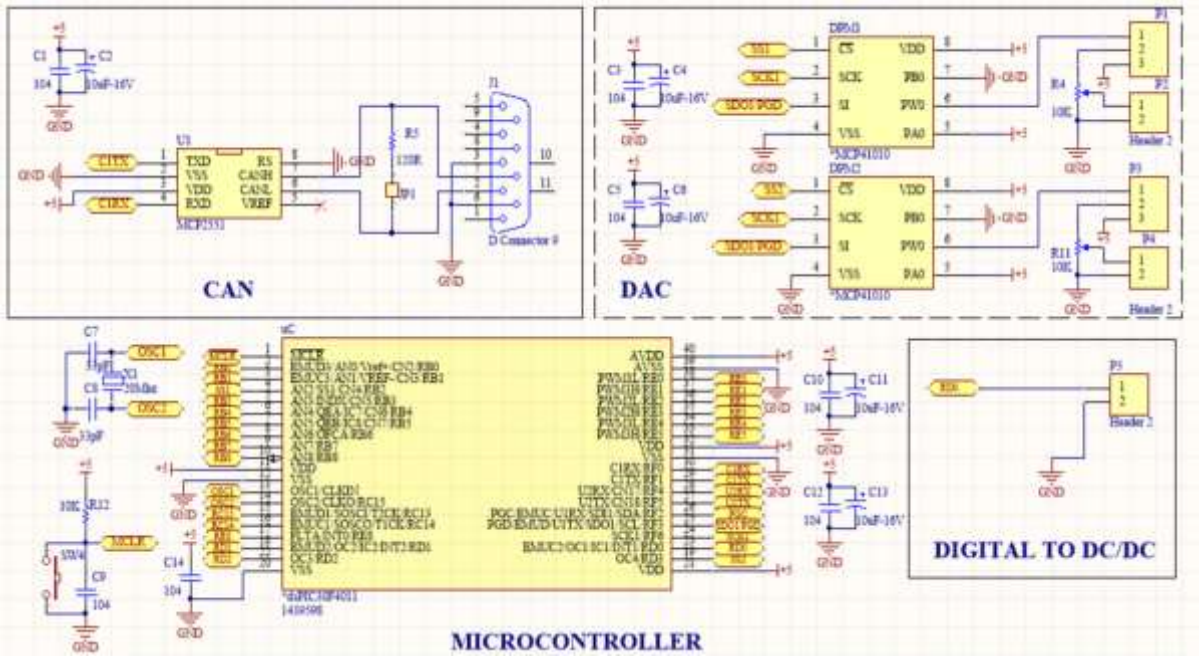


Fig. 4. Driving interface schematic

4. Hardware design

Criteria for the interface device's hardware are as follow

- Be able to communicate with the car's CAN bus via ODB2 port
- Give continuous analogue outputs (voltage) indicating the brake and accelerator positions using SPI protocol
- Give digital output (voltage) indicating the state of the brake (pressed or released) to the DC/DC

- control unit
- Transmit data received from CAN bus (speed, lights, gear, handbrake etc.) to a display screen via UART for further applications

From these criteria, the device is designed as shown in Fig.4 with 4 main blocks. The digital signal microcontroller dsPIC30F4011 from Microchip is chosen because it has a CAN interface that supports CAN 2.0A and CAN 2.0B with programmable baud rate up to 1Mbit/sec, 2 receive buffers with a total of 6 acceptance filters; it also has one SPI and two UART modules [5][6]. A transceiver is required for the dsPIC to communicate with the CAN bus, and the IC MCP2551 is selected for this purpose [7]. The transceiver is connected to the dsPIC via pin C1TX (transmitting pin) and pin C1RX (receiving pin); it is also connected to the car's ODB2 port using a DB9 connector [6].

For the DAC block, two digital potentiometers MCP41010 are chosen because they support SPI communication [4]. The processed accelerator and brake data are transmitted to two digital potentiometers via SDO1 pin and the clock signal is provided via SCK1 pin. As there is only one SPI chip select pin available on the dsPIC and daisy-chain connection is not preferred, another general purpose output pin of the PIC is configured to be the chip select pin for the second potentiometer. The brake switching signal is output on another general IO pin.

The device can be connected to a display screen using another DB9 connector as shown in Fig.5. The data collected from CAN bus can be transmitted to the screen using UART communication. This display screen can serve as a dashboard indicator. In addition, for debugging purpose, 3 LEDs and 3 switch buttons are added to the board as shown in Fig.6.

5. Software design

The embedded code for the module is written in C using MPLAB XC16 compiler. Besides the criteria listed above for the hardware, the software has to be able to read data from selected messages containing information regarding brake, accelerator position, speed, odometer count, lights, gear position, battery and hand brake. On the question of deciphering the CAN messages, the study "Design of CAN bus data reader" by V.D. et al found that only messages corresponding to 7 identifiers contain these data [9]; hence only these messages were collected in the receive buffers. The software flowchart is shown on Fig.7.

Prior to receiving any messages, the CAN module has to be initialized. During this initialization process, the operation mode, baud rate, masks and filters are configured. As the module is not required to transmit anything to the CAN bus, it is set in Listen Only mode, with a baud rate of 500kbit/sec. The number of time quanta per bit time is chosen to be 20, with synchronization time of T_q, propagation time of $3T_q$,

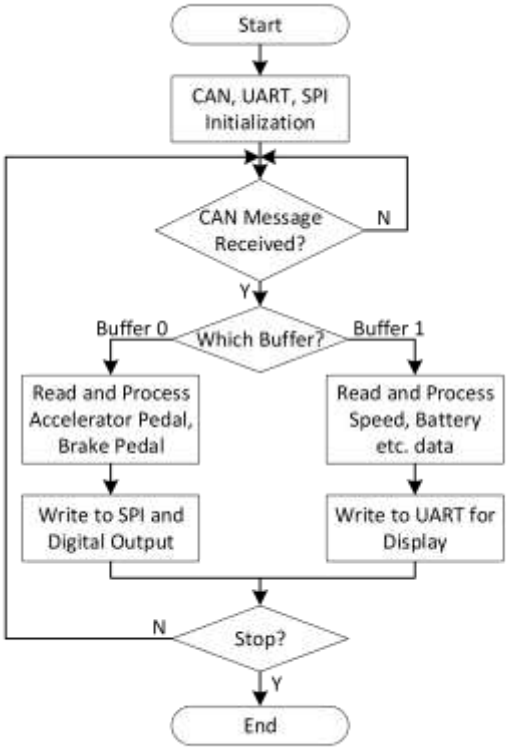


Fig. 7. Software flowchart

phase segment 1 and phase segment 2 both of $8T_q$.

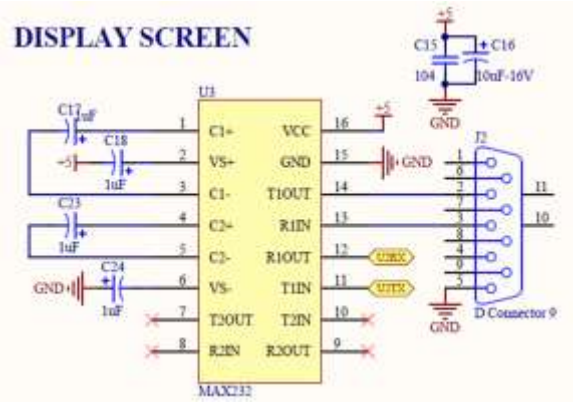


Fig. 5. Display screen schematic

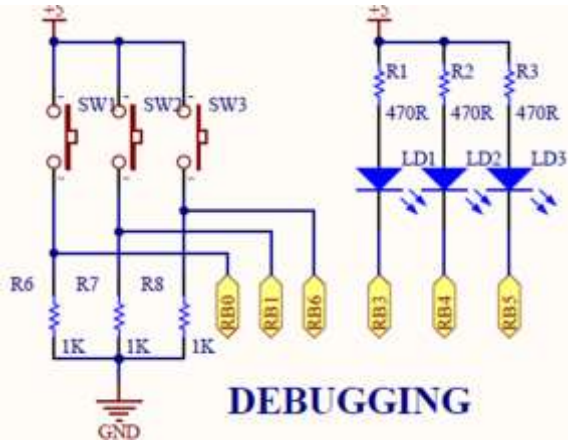


Fig. 6. Debugging block schematic

From equation (1), T_q is deduced to be 100ns. Additionally, according to the dsPIC30F4011 datasheet [5],

$$T_q = \frac{2(BRP + 1)}{F_{CAN}} \quad (2)$$

F_{CAN} is chosen to be equal to crystal frequency and BRP is the baud rate prescaler, the value written to the BRP register is determined to be 0. The masks and filters of the module also set up so that it only receives messages of interest. Buffer 0 is reserved for brake and accelerator related messages only, whereas buffer 1 is set up to receive messages for the display screen.

Regarding the SPI module initialization, it is configured to 16-bit transmission mode to compatible with the potentiometers, two digital outputs from the dsPIC are chosen as chip selecting, overflow flag is cleared and the module is then enabled.

Fig. 8. The prototype

Once a valid message is received in the buffers, the data is read and appropriate data handling is carried out for each type of data. Data received in buffer 1 was placed in an array byte by byte and then transmitted to the screen using UART.

Finally, the buffer status bit, which would be set whenever a buffer receives a message, is required to be cleared by software upon data processing. Once the status bit is cleared, the module continues to check if any valid messages are received without any delay. This guarantees the outputs to be continuous and instantaneous.

6. Experimental validations

6.1 Validation scenarios

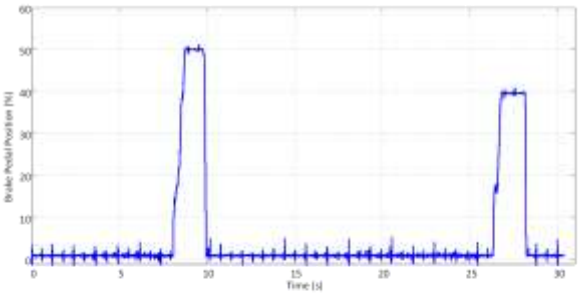
Fig.8 illustrates a prototype of the device and its input/output connections. Two tests were adopted to evaluate the functionality a prototype of the device. The first test would evaluate the accuracy, sensitivity and stability of the prototype in response to the pressing force on the pedals. Because practically, the brake and accelerator pedals are not allowed to be pressed simultaneously, hence each output signal was observed independently in this test. The device was connected only to the CAN bus using an ODB2 connector and the outputs were observed in an oscilloscope. At first, the pedals were pressed to its maximum position and held there for a few seconds. The pedals were then pressed and released repetitively for several times, however with smaller displacements in order to test the signal response to the driver's impact.

In the second test (or the online test), both signals were connected to the Motor Control Unit and DC/DC Control Unit as shown in Fig.8 to check the validity of the output signal against the motor speed. The accelerator was pressed so that the motor speed reached a certain value; after that an immediate braking followed. This procedure was then repeated for one more time.

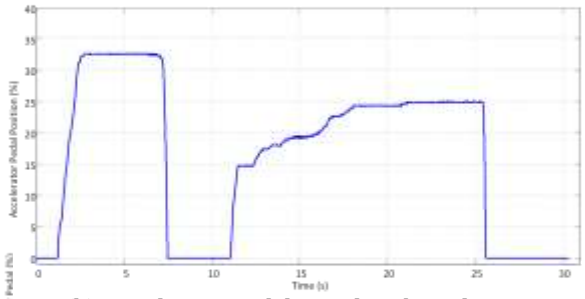
6.2 Results

The offline test's results are illustrated in Fig. 9. As can be seen from both figures, there are spikes in the signals corresponding to the times at which the pedals were pressed. The first spike peaked at approximately 90% of position range of the pedals, indicating that the pedals were fully pressed. The subsequent spikes only reach about 40% of the range, indicating that the pedals were not pressed as hard. These responses match the variations in the pressing force.

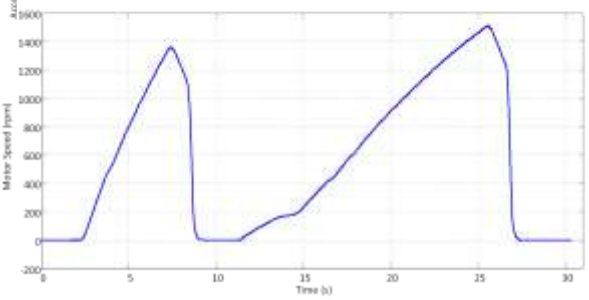
As can be seen from Fig.10(b) of the online test's results that are shown in Fig. 10, there is a spike in the accelerator signal at the beginning of the sample time that corresponds to an increase in motor speed from 0



(a) Brake pedal signal in the online test



(b) Accelerator pedal signal in the online test



(c) Motor speed in the online test

Fig. 10. Online test

to about 1400 rpm shown in Fig.10c. Similarly, there is a sudden drop in motor speed to 0 rpm corresponding to a rise in the brake signal. Hence the device worked exactly as stated in the specifications.

7. Conclusions

This paper has proposed a design of a device that obtains driving system data from the CAN bus of i-MiEV and transmits the processed signals to the Motor Control Unit and the DC/DC Control unit using CAN and SPI communication protocols. It is evident from multiple test results above that the device operates excellently and has fulfilled the design specifications. This device would be used for further development of the vehicle's dashboard indicator.

Acknowledgement

This study was supported by the State granted Project KC03.08/11-15: "Design of Control System and Drive for Electric Vehicles".

References

[1] Kvaser, "The CAN Protocol Tour - CAN physical layers", *www.kvaser.com*, 2014.
[2] Leens, F., 2009. "An introduction to I2C and SPI protocols". *IEEE Instrum. Meas. Mag*, vol. 12, pp.8-13, Feb.2009.
[3] Maxim Integrated, "Daisy-chaining SPI devices - Application note", www.maximintegrated.com, 2015.
[4] Microchip Technology Inc., "MCP41XXX/ 42XXX Datasheet", *www.microchip.com*, 2003.
[5] Microchip Technology Inc., "dsPIC30F4011/4012 Data Sheet", *www.microchip.com*, 2005.
[6] Microchip Technology Inc., "dsPIC30F Family Reference Manual", *www.microchip.com*, 2005.
[7] Microchip Technology Inc., "MCP2551 Datasheet", *www.microchip.com*, 2010.
[8] Robert Bosch GmbH, "CAN Specifications Version 2.0", *www.kvaser.com*, 1991.
[9] Võ Duy Thành, Lê Tiến Sự, Nguyễn Hà Thành Long, Tạ Cao Minh, "Design of CAN bus data reader", *Hội nghị toàn quốc lần thứ 7 về Cơ Điện tử - VCM*, 2014.
[10] Digikey Inc., "CAN data frame format", *www.digikey.com*, 2012.

Thanh Vo-Duy was born in 1982. He graduated from Hanoi University of Science and Technology (HUST) in 2004, majoring in Automation Engineering. He spent one year to work as a robotics researcher in Faculty of Electric Engineering, Meijo University, Nagoya, Japan. After that, he got his Master Degree of Control and Automation at HUST in 2007. Until now, he is working as a lecturer at Department of Industrial Automation, School of Electric Engineering, HUST. Recently, his research is focused on motion control and parameters estimation of electric vehicle.

Tran Thi Minh Trang was born in 1995. She is currently a second-year undergraduate student in Electrical and Electronics Engineering at University of Manchester, United Kingdom. Her interests include embedded systems and robotics.

Ta Cao Minh graduated from Czech Republic in 1986 and got Ph.D. at Laval University, Canada 1997. He had 6 years working in industrial and academic environments in Japan (1998 – 2004) and visiting professor in Taiwan (2010), Australia (2012) and France (2015). He became Assoc. Prof. in the Department of Industrial Automation, HUST in 2009. Besides, he is the Director of Center for Technology Innovation, HUST. His research interest is focused on Control of Electric Drive, Power Electronics, applications for electric vehicles and renewable energy. Being the author of 27 papers published in international journals and proceedings, 14 Japanese granted patents, Assoc. Prof. Minh was awarded the 2nd Prize for best paper of the IEEE-IAS Conference in 2000 and the C Prize for the inventions in NSK Co. (Japan) in 2012. He was the IEEE Vietnam Section Chair (2008 – 2011) and presently the General Secretary of the Vietnam Automation Association (2014 – 2019).