Arduino-Based ECU Simulator and OBD-II Scanner for Electric Vehicle Diagnostics with CAN BUS

Tole Sutikno[1*], Mulyadi[1], Son Aliakbar[1], Tri Wahono[2]

[1]*Faculty of Industrial Technology, Universitas Ahmad Dahlan (UAD), Yogyakarta, Indonesia*

[2]*Embedded Systems and Power Electronics Research Group (ESPERG), Yogyakarta, Indonesia*

[*]*Corresponding author. Email: tole@uad.ac.id*

Abstract— Electric vehicles (EVs) require reliable and accurate diagnostic systems to ensure the performance, safety, and efficiency of their electrical and electronic components. This study presents the design and development of an Arduino-based Electronic Control Unit (ECU) simulator and an On-Board Diagnostics II (OBD-II) scanner for electric vehicle diagnostics, both using the Controller Area Network (CAN) communication protocol. The ECU simulator is built using an Arduino Mega 2560 and an MCP2515 CAN controller, equipped with potentiometers and push buttons to simulate various EV operating parameters such as motor speed, vehicle speed, battery voltage, temperature, state of charge, and regenerative braking conditions, as well as to generate Diagnostic Trouble Codes (DTCs). The OBD-II scanner is developed using an Arduino Uno, MCP2515 module, and a Nextion HMI LCD to support live data monitoring, DTC reading, and DTC clearing in accordance with SAE J1979 standards. System performance was evaluated using a CAN analyzer to verify frame structure, arbitration behaviors, data integrity, and communication reliability. The experimental results show that all transmitted CAN frames conform to the ISO 11898 standard and are correctly interpreted by the OBD-II scanner. The system achieved data accuracy of approximately 97% with an average response time of ±100 ms, indicating stable, reliable communication. The developed platform provides a low-cost, flexible, and effective solution for electric vehicle diagnostics, education, and laboratory testing without requiring real vehicles. It can serve as a foundation for future research involving multi-ECU communication and additional OBD protocols.

Keywords— Arduino; DTC; Electrical Vehicle; LCD HMI; OBD

I. INTRODUCTION

Four-wheeled vehicles have continued to evolve continuously since they were first mass-produced in 1913 by Ford, which significantly changed the manufacturing philosophy as well as operational principles of modern vehicles. As technology develops, innovation in the automotive sector not only focuses on mechanical aspects but also extends to the integration of electronic and computing systems. Improved combustion efficiency, aerodynamic design, and the adoption of increasingly complex electronic control systems are the main characteristics of modern vehicles. These various electronic subsystems collaborate and integrate in a centralised control unit, known as an Electronic Control Unit (ECU), to ensure that the vehicle can operate optimally, safely, and efficiently [1], [2].

In addition to conventional petroleum-fuelled vehicles, Electric Vehicles (EVs) are emerging as a more environmentally friendly transportation alternative, in line with increasing global awareness of climate change and efforts to reduce greenhouse gas emissions. EVs offer advantages such as zero emissions at the point of use, higher energy efficiency, and the potential for integration with renewable energy sources [3], [4]. Nevertheless, the fundamentally different characteristics of electric drive systems compared to internal combustion engines demand new approaches to vehicle monitoring, control, and diagnostics. Therefore, a reliable diagnostic system is needed to ensure the performance, safety, and efficiency of the main components of the EV, such as traction batteries, electric motors, inverters, Battery Management Systems (BMS), and charging systems [1], [5], [6].

In the field of automotive electronics, diagnostics is defined as the process of reading data, analysing operational parameters, and identifying faults or anomalies in vehicle subsystems. This process is carried out automatically by the ECU, which reads signals from various sensors attached to the vehicle [7]. If the sensor value is outside the predetermined range, the ECU will identify the condition as a fault and store it in a specific code form. This error information is known as a Diagnostic Trouble Code (DTC), which can further trigger a warning indicator on the instrument panel, such as a check engine light or Malfunction Indicator Lamp (MIL) [8]. DTC data and vehicle operational parameters can be accessed through the On-Board Diagnostics (OBD) terminal using the Controller Area Network (CAN) communication protocol, which operates over two differential channels, namely CAN-High (CAN-H) and CAN-Low (CAN-L) [9].

OBD systems began to be widely implemented in the 1990s, with the introduction of the On-Board Diagnostics II (OBD-II) standard, which aimed to standardise vehicle diagnostic methods [10]. This standard allows the reading of various operational parameters, such as engine Revolutions Per Minute (RPM), engine temperature, throttle position, and exhaust emissions. However, in electric vehicles, the OBD system has different characteristics due to the absence of an internal combustion engine. The diagnostic focus on EVs is primarily on electrical and electronic power parameters, including battery voltage and current, battery cell temperature, State of Charge

Received 27 January 2026, Revised 15 April 2026, Accepted 17 April 2026.
DOI: https://doi.org/10.15294/jte.v18i1.41900

(SoC), State of Health (SoH), electric motor performance, and inverter efficiency [11]. However, some supporting systems, such as the Anti-lock Braking System (ABS), steering system (electric power steering), Supplementary Restraint System (SRS), and active suspension, still have architectural similarities with conventional vehicles [12].

Although OBD has become the standard for vehicle diagnostics, its application to electric vehicles still faces several challenges. One of the main obstacles is the limited availability of devices capable of accurately quantifying the ECU signal of electric vehicles, especially for education, research, and laboratory testing [13]. In addition, the availability of open, flexible, and low-cost diagnostic tools for reading and removing DTC EVs remains relatively limited [14]–[16]. This condition underscores the need to develop OBD systems based on microcontroller platforms, such as Arduino, combined with ECU simulators and CAN-BUS protocols, as an economical, easy-to-develop alternative [17], [18].

Previous studies on Arduino-based OBD-II systems and ECU simulators have mainly focused on conventional vehicles or on limited diagnostic functions, often lacking support for electric-vehicle-specific parameters such as regenerative braking, battery state of charge, and multi-DTC management. Furthermore, most existing works do not provide a low-cost, flexible platform for education and laboratory testing. This research addresses these gaps by developing an Arduino-based ECU simulator specifically tailored for electric vehicles, integrating CAN-BUS communication, regenerative braking simulation, and real-time diagnostic functions. Compared to earlier systems, the proposed platform offers improved accuracy, faster response times, and enhanced usability for EV diagnostics and training.

II. METHOD

A. System Architecture

The object of this study is the design and development of OBD devices consisting of ECU simulators and OBD-II scanner diagnostic tools. The ECU simulator is designed to represent the ECU's operational characteristics in electric vehicles, particularly in generating and transmitting operational parameter data and DTC error codes in accordance with the OBD-II standard. Meanwhile, the OBD-II scanner diagnostic tool was developed to read, interpret, and manage data transmitted by the simulator ECU via the OBD-II terminal, using the standard OBD-II communication protocol based on CAN [19]. Both devices communicate using the CAN-BUS ISO 15765-4 protocol at 500 kbps over CAN-High and CAN-Low lines, in accordance with modern OBD-II communication standards [20].

This research was conducted using an experimental method within an engineering and technology development approach. The research stages include hardware and software design, integrated system implementation, and system performance testing. The test process was carried out to evaluate the functionality and reliability of CAN-BUS communication, data compliance with OBD-II standards, and the system's ability to accurately and measurably simulate electric vehicle operating conditions.

B. ECU Simulator Design

The system consists of several main components that are integrated and have specific functions. The Arduino Mega 2560 acts as the main control unit, serving as the data processing centre and controlling the entire subsystem. This microcontroller is responsible for receiving input signals from various simulation sensors, processing data according to the designed programming logic, and transmitting the processed data to the output devices and CAN-BUS communication networks. The use of microcontrollers as the main controller is a common approach in the development of ECU prototype systems and vehicle diagnostic devices due to their flexibility and ease of integration [12], [20].

On the ECU side of the simulator, local display media show the simulated data stream, allowing the system's operational condition to be monitored directly without the need for additional diagnostic devices. Presenting data directly on the visual interface facilitates the observation and validation of system parameters during the testing phase [5]. User interaction with the system is facilitated through several push buttons as a digital input device. The T1 button is used to move between display pages, while the T2-T4 buttons serve as DTC simulation inputs. In addition, the T5 button is used to simulate the brake pedal sensor, which, when activated, will trigger the function of the regenerative braking system, which is the change of the working mode of the electric motor to a generator to return electrical energy to the battery, as is the basic principle of the regenerative braking system in electric vehicles [21].

To simulate various operational parameters of electric vehicles, the system is equipped with six VR1-VR6 potentiometers that serve as analogue signal sources. The potentiometers are each used to simulate the rotation of the motor, the speed of the vehicle, the position of the gas pedal, the battery temperature, the battery voltage, and the capacity of the battery or SoC. This simulation approach using variable analogue signals is commonly used in the research and development of simulator ECU systems to represent the condition of vehicle sensors in a controlled, measured manner [12].

Data communication between devices in this system is facilitated by an MCP2515 module that functions as a CAN-BUS communication controller. This module allows the Arduino Mega 2560 to send and receive data according to the CAN-BUS protocol standard used in OBD-II systems. As a standard interface with external diagnostic devices, a Data Link Connector (DLC) port or 16-pin OBD-II terminal is used, which serves as a communication medium between the simulator ECU and other devices, such as the OBD-II scanner, so that the reading process of operational parameters and DTC can be carried out in accordance with vehicle diagnostic standards. Figure 1 shows the simulator ECU block diagram.

C. OBD-II Scanner Design

The OBD-II scanner is designed to be integrated with the Human Machine Interface (HMI) to support vehicle diagnostic functions more interactively. The Arduino Mega 2560 serves as the main component and control centre of the system.

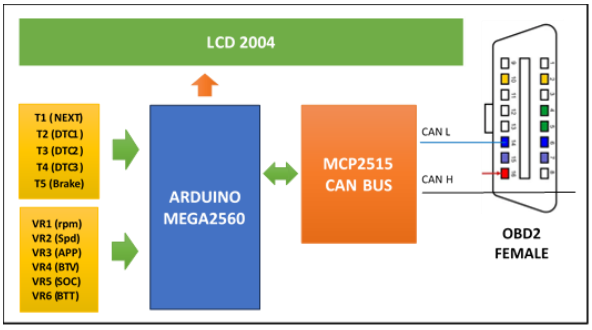

Figure 1. Simulator ECU block diagram

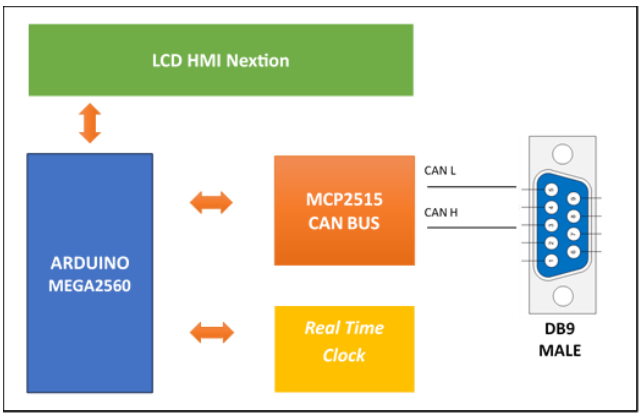

Figure 2. OBD scanner block diagram

This microcontroller is responsible for processing input data, controlling system logic, and sending and receiving data through the CAN-BUS communication network. The use of microcontrollers as the main controller is a common approach in the development of ECU prototype systems and vehicle diagnostic devices due to their flexibility and ease of integration [1].

As the main input and output interface, Nextion's HMI LCD is used to display operational parameters and DTC information from the data stream and to provide a means of user interaction [22]. Through this HMI, users can not only visually monitor the system's condition but also issue diagnostic commands, including DTC-clearing commands, similar to those in commercial OBD-II scanners. The use of touchscreen-based HMIs aims to improve the ease of operation and the effectiveness of user interaction with diagnostic systems [9].

Data communication on this system is facilitated by an MCP2515 module that serves as a CAN-BUS controller. This module enables the Arduino Mega 2560 to communicate with other devices on the CAN-BUS network in accordance with the OBD-II protocol standards. The presence of this CAN controller module is essential to ensure reliable, synchronous data exchange in accordance with the applicable CAN message format [20].

In addition, this system is equipped with a Real Time Clock (RTC) module that provides real-time time information. RTC is used to support the process of recording time when reading diagnostic data and error events, so that the resulting data can be analysed chronologically and more structured. The application of time stamping in diagnostic systems is a commonly used practice to improve the accuracy of system analysis and evaluation [12].

As an external communication interface, the DB9 connector is used, which functions as a communication terminal between the OBD-II scanner and the OBD simulator. These connectors provide a physical path for CAN-BUS data transmission, allowing external diagnostic devices to be connected directly to the simulator system via a stable, standardised communication medium, as used in a wide range of CAN analyser devices and automotive diagnostic tools [23]. The OBD-II scanner was developed using an Arduino Uno, a nan MCP2515 module, and a Nextion 3.5-inch HMI LCD. This device supports Modes 01 (live data), 03 (read DTC), and 04 according to SAE J1979 standards. Figure 2 shows the block diagram of the OBD-II scanner.

The electric vehicle simulator ECU was developed using the Arduino Mega2560 microcontroller and successfully implemented with the CAN-BUS MCP2515 communication module operating at 500 kbps, in accordance with the vehicle's OBD-II communication standard. The system is designed to simulate the basic functions of the electric vehicle ECU, including sensor signal simulation via a potentiometer, a DTC simulation button, a regenerative braking feature, and an LCD display interface divided into two pages to display operational data alternately.

The OBD ECU simulator device developed can generate and receive data via the DLC or OBD-II port using the CAN-BUS communication protocol, enabling it to act like an ECU in a real electric vehicle. The data generated by the simulator can then be accessed and analysed using an OBD-II scanner, a diagnostic tool [24], [25]. Thus, the system enables live data reading, DTC identification, and integrated communication testing between the simulator ECU and the diagnostic device. The physical appearance of the simulator ECU device and the OBD-II scanner are shown in Figure 3, respectively.

D. Experiment Procedure

The experimental procedure was organised into three main stages to validate the functionality and performance of the developed ECU simulator and OBD-II scanner:

1) Hardware Validation

The ECU simulator and the OBD-II scanner are connected via a CAN-BUS communication module, MCP2515 operating at 500 kbps. A CAN analyser is used to observe frame integrity, arbitration behaviour, and error handling. Verified parameters include frame identifier (ID), Data Length Code (DLC), payload structure, Cyclic Redundancy Check (CRC) field, and recognition signal. This stage ensures that the hardware and communication protocols are compliant with ISO 11898 and SAE J1979 standards.

2) Functional Testing of ECU Simulator

Potentiometers (VR1–VR6) are used to simulate EV parameters: motor speed, vehicle speed, throttle position, battery voltage, battery temperature, and state of charge (SoC). The push button simulates DTCs and regenerative braking conditions. The LCD simulator displays real-time values to validate the accuracy of the sensor simulation. Each parameter is systematically varied to observe the generation and transmission of the corresponding CAN frames.

3) Diagnostic Testing of OBD-II Scanner

The scanner is tested in three standard OBD-II modes: Mode 01 (Live Data), which verifies real-time monitoring of simulation parameters. Mode 03 (DTC Reading): validates single and multi-DTC DTC transmissions (e.g., P0A80, P0B3B, P0C78). Mode 04 (DTC Deletion): ensures proper reset of active and pending code. Accuracy is calculated as the ratio of correctly interpreted CAN frames to the total frames transmitted. Response time is measured by timestamping parameter ID (PID) requests and scanner output, resulting in an average latency of ±100 ms. Comparison with CAN analyser data serves as a basic truth to confirm reliability.

E. Test Method

CAN-BUS communication testing in the early stages was performed using a CAN adapter connected to the CAN analyser software. This test aims to make direct observations of the CAN data traffic generated by the simulator ECU during the system's operation. The parameters observed include the ID, DLC, and the data payload sent to each CAN frame. In addition, this test is used to evaluate the bus arbitration mechanism, communication synchronisation, and the system's ability to handle error conditions, such as bit errors, stuffing errors, and acknowledgement errors, that can occur during CAN communications.

Figure 3. ECU simulator and OBD-II scanner

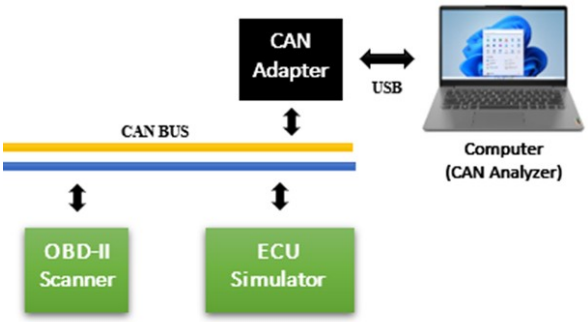

Figure 4. Test block diagram with CAN Analyzer

Using CAN analyser devices, every transmitted and received CAN frame can be recorded, displayed, and analysed in detail, enabling verification of the data format's compliance with the CAN-BUS protocol and the OBD-II standard. The results of these tests serve as the basis for ensuring that the simulator's ECU system can generate valid, stable, and correctly interpretable CAN data traffic for external diagnostic devices. The block diagram of the CAN-BUS communication test configuration using the CAN adapter is shown in Figure 4.

CAN analyser is a hardware and software used to monitor, record, and analyse data traffic on the CAN-BUS network. In this test, the CAN analyser serves as a measurement tool, allowing researchers to observe data communication at the CAN frame level without affecting the system-under-test's performance. Thus, the CAN analyser can provide an objective picture of the CAN communication characteristics generated by the simulator ECU.

Functionally, the CAN analyser can display detailed information for every CAN frame transmitted and received on the network, including frame identifiers, DLC, and data payloads in hexadecimal and decimal formats. This information is critical for verifying the suitability of the data format with the CAN-BUS protocol specifications and OBD-II standards, particularly to ensure that the data IDs and structures transmitted by the simulator ECU are correctly recognised and interpreted by external diagnostic devices, such as OBD-II scanners.

In addition, the CAN analyser has an important role in testing the bus arbitration mechanism. In the CAN-BUS network, multiple nodes can transmit data simultaneously, so ID-priority-based arbitration is crucial to ensure reliable communication. Using the CAN analyser, this arbitrage process can be observed directly, including the order in which frames are sent and the priority of the IDs used, ensuring the system follows the correct CAN communication principles.

CAN analysers are also used to detect and analyse fault conditions in CAN-BUS communications. Some types of errors that can be observed include bit errors, stuffing errors, CRC errors, acknowledgement errors, and frame errors. This error detection is important for evaluating the system's reliability, particularly in ensuring that the simulator ECU and the CAN controller (MCP2515) can handle communication errors in accordance with the error-handling mechanism established in the CAN-BUS standard. The error information displayed by the CAN analyser is the basis for assessing the stability and robustness of the developed communication system.

In this study, the CAN analyser also serves as a comparison tool to validate the OBD-II scanner's data readings. The CAN data displayed on the CAN analyser is compared with the data received and displayed by the OBD-II scanner to evaluate the accuracy and consistency of the diagnostic process quantitatively. With this approach, the CAN analyser serves as ground truth for testing CAN-BUS communication and OBD-II services implemented on the simulator ECU [26], [27].

Overall, the use of CAN analysers in this test provides real-time data analysis, high precision, and flexibility for observing various CAN-BUS communication parameters. Therefore, the CAN analyser is an important component in ensuring that the ECU system of the simulator and the OBD-II scanner developed meet the requirements of functionality, reliability, and compliance with automotive communication standards.

III. Results and Discussion

The test results showed that every CAN frame delivered by the simulator ECU was consistently detected by the CAN analyser, with a frame structure that conformed to ISO 11898. Based on the observation results, the recorded CAN frame consists of several main parts, namely the identifier field, control field, data field, CRC field, and acknowledgement field. The presence of all these elements indicates that the simulator ECU has correctly formed and transmitted the CAN frame, in accordance with the CAN-BUS communication protocol specifications.

In addition, the test results also show that the CAN arbitration mechanism is functioning as it should. In conditions where more than one frame is simultaneously transmitted to the CAN-BUS network, frames with smaller identifier values gain higher transmission priority than frames with larger identifier values. This phenomenon aligns with the principle of non-destructive bitwise arbitration in the CAN protocol, where arbitration occurs at the bit level without causing data loss or communication interruptions to other nodes. With this mechanism, nodes that lose the arbitration process will automatically delay transmission and resend the data once the bus returns to a free state.

TABLE I. Simulator ECU and OBD-II Scanner Test Result

No	Mode	Parameter	Test Results	Status
1	CAN Communications	Structure frame ISO 11898	Frame valid (ID, CRC, ACK)	Successful
2	Mode 01	Live data stream (PID B1)	Data as per potentiometer input	Successful
3	Mode 03	1 DTC Reading	P0A80 reads correctly	Successful
4	Mode 03	Multi-DTC readout	P0C78, P0B3B, P0A80 read	Successful
5	Mode 04	DTC Removal	All DTCs are deleted	Successful
6	System performance	Data reading accuracy	97%	Excellent
7	System response	ECU response time	±100 ms	Stable

These results show that implementing the CAN-BUS protocol on the simulator ECU not only allows the ECU to form frames according to the standard but also correctly supports arbitration and message prioritisation. This is an important indicator that the developed system has high compatibility with the automotive CAN-BUS network and is suitable as a test and simulation platform for the OBD-II diagnostic system.

The test results summarised in Table I show that the ECU simulator and OBD-II scanner developed can effectively replicate the basic functions of the electric vehicle diagnostic system. Every aspect of testing, from CAN communication to DTC management, delivers consistent results and complies with OBD-II standards. CAN communication testing shows that all frames delivered by the simulator ECU have a valid structure per ISO 11898, including identifiers, control fields, data fields, CRC fields, and acknowledgement fields. This indicates that the MCP2515 module and the software implementation on the Arduino can correctly form the CAN frame. These findings align with the characteristics of the CAN protocol described by [13], which state that a valid frame structure is a key requirement for successful communication between nodes. In addition, the CAN arbitrage mechanism during the test showed that frames with smaller identifier values had a higher transmission priority, without causing data loss. This is in accordance with the principle of non-destructive arbitration in the CAN protocol, as reported in [20], and demonstrates that the simulator system can replicate the behaviour of the CAN network in real vehicles.

Mode 01, the simulator ECU successfully responds to standard PID requests and displays live stream data in real time through the OBD-II scanner. The displayed data correspond to the potentiometer input values used to simulate electric vehicle parameters, such as motor speed and battery voltage. The data reading accuracy rate of 97% indicates that the difference between input and output values is very small, allowing the system to be reliably used for testing and learning. These results are in line with research [1], [28], which demonstrates the effectiveness of Arduino- and CAN-BUS-based OBD-II systems for reading vehicle operational parameters.

Mode 03 testing showed that the simulator's ECU could transmit one or more DTCs in a single CAN frame. The resulting DTCs, such as P0A80, P0B3B, and P0C78, are related to electrical systems and electric vehicle batteries. This is in accordance with the diagnostic characteristics of electric vehicles, which place greater emphasis on electric power systems than those of fossil-fuel vehicles [12]. The successful delivery of multi-DTC in a single frame also conforms to the SAE J1979 standard format, as described by [29].

In Mode 04, the simulator ECU successfully receives the DTC removal command and provides an appropriate response, characterised by the loss of the entire DTC on the next reading. This behaviour is identical to the real vehicle ECU function based on the SAE J1979 standard, where all active and pending DTCs are deleted without affecting the live data Mode 01 [23]. The success of Mode 04 demonstrates that the simulator system can mimic the diagnostic reset process in modern vehicles.

The system achieves a data accuracy rate of 97%, which is calculated as the ratio of correctly interpreted CAN frames to the total frames transmitted across various parameters (motor speed, battery voltage, SoC). Accuracy determinants include ADC (10-bit) resolution, CAN-BUS synchronisation reliability, and frame arbitration latency. The average response time of ±100 ms is measured from the time of the PID request to the display of scanner data. Compared with previous Arduino-based OBD systems, which reported accuracy rates of 85–92% [29], [30], the proposed system demonstrated improved reliability and responsiveness. This improvement is due to the integration of MCP2515 CAN controllers and software routines optimised for error handling.

Overall, the results of this study show that the ECU simulator and OBD-II scanner developed can replicate the basic behaviour of the electric vehicle diagnostic system effectively and can be used as a learning medium and research platform [12], [31]. However, this system is still limited to the CAN-based OBD-II protocol and does not yet support other protocols such as ISO 9141-2 or SAE J1850. In addition, the simulator has not yet fully implemented multi-ECU communication, so that further development can focus on simulations of more complex vehicle networks closer to real vehicle conditions.

Future system upgrades will include support for additional OBD protocols (ISO 9141-2, SAE J1850), multi-ECU communication to simulate complex EV networks, and IoT-based remote diagnostic integration. Simulation of fault conditions such as overheating, battery degradation, and inverter failure will also be included to expand the diagnostic scope. In addition, higher-resolution ADCs and advanced error-detection algorithms can be implemented to improve accuracy further and reduce false readings.

IV. Conclusion

This study successfully designed and implemented an Arduino-based electric vehicle ECU simulator and an OBD-II diagnostic scanner using the CAN-BUS communication protocol. The developed system complies with ISO 11898 and SAE J1979 standards and operates at a communication speed of 500 kbps, confirming its suitability for modern automotive diagnostic applications. The ECU simulator was able to replicate key electric vehicle operational parameters, including motor speed, battery voltage, temperature, and state of charge, through analogue and digital inputs. It also supported regenerative braking simulation and real-time data visualisation via an LCD interface. CAN-BUS testing using a CAN analyser verified that all transmitted frames had valid structures, correct arbitration behaviour, and stable communication without data loss. The OBD-II scanner demonstrated reliable diagnostic functionality, including live data reading (Mode 01), DTC detection (Mode 03), and DTC clearing (Mode 04). The system achieved a data accuracy rate of approximately 97% with an average response time of ±100 ms, indicating high reliability and responsiveness. Overall, the proposed system provides a valid, low-cost, and flexible platform for electric vehicle diagnostics, education, and research. It enables realistic simulation and testing without requiring real vehicles. Future development will expand its application to a broader range of multi-ECU environments and diagnostic protocols, reinforcing its role as a flexible, scalable tool for intelligent electric vehicle systems.

Acknowledgement

The authors would like to express their sincere gratitude to Ahmad Dahlan University for the institutional support and facilities provided throughout this research. Special appreciation is extended to the supervisors and academic staff of the Faculty of Industrial Technology for their guidance, constructive feedback, and continuous support during the design, implementation, and documentation stages of this study. The authors also acknowledge the assistance of the laboratory staff and the Embedded Systems and Power Electronics Research Group (ESPERG) for providing technical resources

and a conducive research environment, particularly in testing the CAN-BUS communication system, ECU simulator, and OBD-II diagnostic tools. Furthermore, the authors would like to thank colleagues and peers for their valuable discussions and technical support. Finally, sincere appreciation is conveyed to the authors' families for their encouragement, patience, and unwavering support throughout this research.

REFERENCES

[1] D. Rimpas, A. Papadakis, and M. Samarakou, "OBD-II sensor diagnostics for monitoring vehicle operation and consumption," *Energy Reports*, vol. 6, pp. 55–63, Feb. 2020, doi: 10.1016/j.egyr.2019.10.018.

[2] M. Al-Flehawee and A. Al-Mayyahi, "Building A Control Unit of A Series-Parallel Hybrid Electric Vehicle by Using A Nonlinear Model Predictive Control (NMPC) Strategy," Iraqi J. *Electr. Electron. Eng*., vol. 18, no. 1, pp. 93–102, Jun. 2022, doi: 10.37917/ijeee.18.1.11.

[3] A. Rahim, S. Agarwal, G. Gupta, R. Singh, and P. K. Malik, "Efficient Use of Renewable Solar Energy Resource for Electric Vehicles: Opportunities and Challenges," *Eng. Reports*, vol. 7, no. 2, p. e70007, 2025..

[4] H. Aydogan, "Electric Vehicles and Renewable Energy," in *Journal of Physics: Conference Series*, 2024, vol. 2777, no. 1, p. 12007.

[5] Nurudin, A. A. Sukmandhani, and M. Zarlis, "Monitoring Applications for Vehicle based on Internet of Things (IoT) using the MQTT Protocol," *Procedia Comput. Sci*., vol. 227, pp. 73–82, 2023, doi: 10.1016/j.procs.2023.10.504.

[6] C. Armenta-Deu and T. Coulaud, "Control Unit for Battery Charge Management in Electric Vehicles (EVs)," *Futur. Transp.*, vol. 4, no. 2, pp. 429–449, Apr. 2024, doi: 10.3390/futuretransp4020021.

[7] N. A. Jasim, H. T. Salim AlRikabi, and M. S. Farhan, "Internet of Things (IoT) application in the assessment of learning process," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1184, no. 1, p. 012002, Sep. 2021, doi: 10.1088/1757-899X/1184/1/012002.

[8] A. Efendi et al., "IoT-Based Elderly Health Monitoring System Using Firebase Cloud Computing," *Heal. Sci. Reports*, vol. 8, no. 3, Mar. 2025, doi: 10.1002/hsr2.70498.

[9] Y. Wang and J. Xu, "Vehicle Real-time Condition Monitoring Based on CAN Bus," *J. Phys. Conf. Ser*., vol. 2405, no. 1, p. 012004, Dec. 2022, doi: 10.1088/1742-6596/2405/1/012004.

[10] A. Ali et al., "An Innovative IoT and Edge Intelligence Framework for Monitoring Elderly People Using Anomaly Detection on Data from Non-Wearable Sensors," *Sensors*, vol. 25, no. 6, p. 1735, Mar. 2025, doi: 10.3390/s25061735.

[11] B. D. Soyoye, I. Bhattacharya, M. V. Anthony Dhason, and T. Banik, "State of Charge and State of Health Estimation in Electric Vehicles: Challenges, Approaches and Future Directions," *Batteries*, vol. 11, no. 1, p. 32, Jan. 2025, doi: 10.3390/batteries11010032.

[12] L. Wang, X. Zou, H. Qin, and P. Geng, "Design of OBD function test on production vehicle (PVE)," *E3S Web Conf*., vol. 268, p. 01047, Jun. 2021, doi: 10.1051/e3sconf/202126801047.

[13] B. Baráth, G. Sütheö, and L. Pekk, "Development of a Battery Diagnostic Method Based on CAN Data: Examining the Accuracy of Data Received via a Communication Network," *Energies*, vol. 17, no. 22, p. 5808, Nov. 2024, doi: 10.3390/en17225808.

[14] U. Orbe Deusto, "Design and Implementation of CAN communication for Owasys IoT devices," 2024.

[15] A. D. S. Roque, L. M. D. S. Alves, and E. P. de Freitas, "CAN-Modes: In-vehicle datasets generation and analysis in different driving situations," in *2024 Workshop on Communication Networks and Power Systems (WCNPS)*, 2024, pp. 1–7.

[16] X. Wang, J. Zhao, P. Liu, N. Yao, and Z. Xu, "Can bus intrusion detection based on deep learning with data augmentation for connected autonomous vehicles," *IEEE Trans. Veh. Technol.*, 2025.

[17] O. M. Ramadan, I. L. Enaya, A. E. Al-Zoul, and B. H. Sababha, "ML-Based CAN Bus Message Sniffing and Classification," in *2024 International Jordanian Cybersecurity Conference (IJCC)*, 2024, pp. 105–110.

[18] R. Gesteira-Miñarro, I. Gutiérrez, R. Palacios, and G. López, "pwnobd: Offensive cybersecurity toolkit for vulnerability analysis and penetration testing of OBD-II devices," *IEEE Access*, 2025.

[19] E. T. Michailidis, A. Panagiotopoulou, and A. Papadakis, "A Review of OBD-II-Based Machine Learning Applications for Sustainable, Efficient, Secure, and Safe Vehicle Driving," *Sensors*, vol. 25, no. 13, p. 4057, Jun. 2025, doi: 10.3390/s25134057.

[20] O. Rybitskyi, V. Golian, N. Golian, Z. Dudar, O. Kalynychenko, and D. Nikitin, "Using OBD2 Technology For Vehicle Diagnostic And Using It In The Information System," *Bull. Natl. Tech. Univ. "KhPI". Ser. Syst. Anal. Control Inf. Technol*., no. 1 (9), pp. 97–103, Jul. 2023, doi: 10.20998/2079-0023.2023.01.15.

[21] M. Haghani, F. Sprei, K. Kazemzadeh, Z. Shahhoseini, and J. Aghaei, "Trends in electric vehicles research," *Transp. Res. Part D Transp. Environ*., vol. 123, p. 103881, Oct. 2023, doi: 10.1016/j.trd.2023.103881.

[22] O. Zubkov, I. Svyd, and O. Vorgul, "Study of the Effectiveness of Using Nextion Displays in Projects Based on STM32 Microcontrollers," in *Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs 2023*, 2023, pp. 7–10, doi: 10.35598/mcfpga.2023.001.

[23] A. W. Wardhana, A. Mubyarto, and A. Taryana, "A Controller Area Network (CAN) Bus Temperature and Humidity Data Monitoring System," *Transm. J. Ilm. Tek. Elektro*, vol. 25, no. 3, pp. 115–125, Jul. 2023, doi: 10.14710/transmisi.25.3.115-125.

[24] K. Sevdari, L. Calearo, B. H. Bakken, P. B. Andersen, and M. Marinelli, "Experimental validation of onboard electric vehicle chargers to improve the efficiency of smart charging operation," *Sustain. Energy Technol. Assessments*, vol. 60, p. 103512, Dec. 2023, doi: 10.1016/j.seta.2023.103512.

[25] J. Li, D. Li, and Y. Xu, "Design of Dual MCU Vehicle Control Unit Based on Electric Vehicles to Respond to Control Failure," *IOP Conf. Ser. Earth Environ. Sci*., vol. 512, no. 1, p. 012132, Jun. 2020, doi: 10.1088/1755-1315/512/1/012132.

[26] T. Bianchi, A. Brighente, M. Conti, and A. Valori, "Your Car Tells Me Where You Drove: A Novel Path Inference Attack via CAN Bus and OBD-II Data," in *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P),* Jun. 2025, pp. 113–132, doi: 10.1109/EuroSP63326.2025.00016

[27] K. Iskandar, A. Tambayong, M. R. F. Mulya, S. C. Elfanlie, and M. G. Herlina, "Mobile-Based Car Diagnostic Application Using Onboard Diagnostic-II Scanner," *ComTech Comput. Math. Eng. Appl*., vol. 14, no. 2, pp. 129–141, Dec. 2023, doi: 10.21512/comtech.v14i2.9138.

[28] M. Ammar, H. Janjua, A. S. Thangarajan, B. Crispo, and D. Hughes, "Securing the On-Board Diagnostics Port (OBD-II) in Vehicles," *SAE Int. J. Transp. Cybersecurity Priv*., vol. 02, no. 2, pp. 11-02-02–0009, Aug. 2020, doi: 10.4271/11-02-02-0009.

[29] M. A. C. Din, M. T. A. Rahman, H. A. Munir, A. Rahman, and A. F. A. Hamid, "Development of CAN Bus Converter for On Board Diagnostic (OBD-II) System," *IOP Conf. Ser. Mater. Sci. Eng*., vol. 705, no. 1, p. 012011, Nov. 2019, doi: 10.1088/1757-899X/705/1/012011.

[30] A. Rodríguez Rodríguez, J. R. Vento Álvarez, and R. Inouye Rodriguez, "Implementation of an OBD-II diagnostics tool over CAN-BUS with Arduino," *Sist. y Telemática*, vol. 16, no. 45, Apr. 2018, doi: 10.18046/syt.v16i45.2747.

[31] A. Małek and R. Taccani, "Innovative approach to electric vehicle diagnostics," *Arch. Automot. Eng. – Arch. Motoryz*., vol. 92, no. 2, pp. 49–67, Jun. 2021, doi: 10.14669/AM.VOL92.ART4.