MODELING AND TESTING VEHICLE NETWORKS WITH CANOE AND LIN PROTOCOL

24EE62 - EMBEDDED NETWORKS AND DEVICE DRIVER LABORATORY

LABORATORY PROJECT REPORT

ARISUDAN TH.

(24MU01)

Dissertation submitted in partial fulfilment of the requirements for the degree of

MASTER OF ENGINEERING

Branch: ELECTRICAL & ELECTRONICS ENGINEERING

Specialization: EMBEDDED AND REAL-TIME SYSTEMS

of Anna University

May 2025

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)
COIMBATORE - 641 004

MODELING AND TESTING VEHICLE NETWORKS WITH CANOE AND LIN PROTOCOL

Bonafide record of work done by

ARISUDAN TH.

(24MU01)

Dissertation submitted in partial fulfilment of the requirements for the degree of

MASTER OF ENGINEERING

Branch: ELECTRICAL & ELECTRONICS ENGINEERING

Specialization: EMBEDDED AND REAL-TIME SYSTEMS

of Anna University

May 2025

……………………	………………………………
Dr. P. SIVAKUMAR	**Dr. J. KANAKARAJ**
Faculty Guide	Head of the Department

Certified that the candidate was examined in the viva voice examination held on …………

- - - - - - - - - - - - - - -	- - - - - - - - - - -
(Internal Examiner)	(External Examiner)

CONTENTS

ACKNOWLEDGEMENT

I express my sincere thanks to Dr. K. Prakasan, Principal, PSG College of Technology for his benevolent patronage in carrying out this project.

I express my sincere respect and gratitude to Dr. J. Kanakaraj, Professor and Head (CAS), Department of Electrical and Electronics Engineering, PSG College of Technology for his continuous encouragement and motivation throughout the project.

I am indebted to my beloved internal guide Dr. P. Sivakumar, Assistant Professor (Sl. Gr) Department of Electrical & Electronics Engineering, PSG College of Technology and whose valuable guidance and inspiration throughout the course made it possible to complete this project work successfully.

I put forth my heart and soul to thank Almighty for being with me all through this technical adventure.

SYNOPSIS

In the development and validation of modern automotive systems, simulation tools play a crucial role in ensuring the robustness, functionality, and compliance of embedded networks before real-world deployment. This report presents a comprehensive study and practical execution of a simulated automotive communication environment using Vector CANoe, specifically targeting LIN (Local Interconnect Network), a cost-effective, serial communication protocol widely used in automotive body and comfort electronics.

The simulation environment replicates the behavior of two major subsystems within a vehicle: the Powertrain and Comfort LIN networks. Each LIN cluster was configured and executed within CANoe using simulated LIN Master and Slave Electronic Control Units (ECUs). The setup included configuring LIN schedule tables, signal definitions, node behavior, diagnostics, and bus traffic, allowing for in-depth observation of message timing, synchronization, and data integrity across the network.

Real-time outputs and network activities were monitored using various CANoe tools such as Trace Logs, Bus Statistics, LIN Schedule Table View, Graphics Panels, Signal Generators, and Diagnostic Request/Response Interfaces. These tools provided insight into frame timing, master-slave communication sequences, bus load, and error handling. Screenshots were captured from the environment to demonstrate the LIN network's performance and message flow under simulated conditions.

This study offers a clear understanding of how LIN-based in-vehicle communication networks operate, how CANoe's integrated tools support validation of LIN Master-Slave interactions, and how diagnostic and timing analysis features contribute to overall system integrity. By leveraging CANoe’s LIN capabilities, the simulation reinforces theoretical concepts while emphasizing practical approaches to configuring, testing, and analyzing LIN communication systems in a controlled virtual environment.

Future scope includes extending the simulation to integrate multi-channel LIN communication, gateway ECUs bridging LIN to CAN or FlexRay, and developing automated test modules for compliance testing, fault injection, and regression analysis of LIN-based systems, especially as vehicle electronics continue to evolve in complexity.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

In modern automotive engineering, the growing complexity of electronic control systems and the need for reliable in-vehicle communication have underscored the importance of robust testing and validation tools. As vehicles evolve into distributed systems featuring numerous Electronic Control Units (ECUs) connected over protocols such as LIN, CAN, and Ethernet, ensuring the proper operation and interaction of these ECUs becomes both essential and challenging.

Traditional validation approaches that rely solely on physical hardware setups suffer from limitations in scalability, cost-effectiveness, and flexibility. Hardware availability, wiring complexity, and setup time can impede development timelines. These constraints have led to the widespread adoption of simulation environments, which provide engineers with the ability to model, test, and validate vehicle communication systems in a virtual setting, well before advancing to hardware-in-the-loop (HIL) or on-road testing.

This project focuses on the use of Vector CANoe, a powerful and industry-standard tool for simulating, analyzing, and testing automotive communication networks, with a particular emphasis on the LIN (Local Interconnect Network) protocol. By configuring and executing a LIN-based simulation involving Powertrain and Comfort subsystems, this work explores the behavior of LIN communication—including master-slave frame scheduling, signal propagation, and node diagnostics.

CANoe’s integrated tools—such as the LIN Setup Editor, Schedule Table Configuration, Signal Generator, Trace and Logger Views, and Diagnostic Panels—were extensively used to observe and analyze real-time LIN network behavior. The simulation environment enables engineers to monitor bus traffic, identify timing discrepancies, validate message structures, and explore system responses to injected faults or irregular conditions.

The core motivation behind using simulation is to establish a controlled, repeatable, and hardware-independent platform for testing ECU communication. This method significantly reduces development time, enhances debugging capabilities, and promotes early detection of design or configuration issues. Moreover, such an environment allows for safe and efficient testing of edge cases that are difficult or risky to replicate in physical setups.

By modeling LIN networks within CANoe and analyzing their runtime behavior, this project serves as a bridge between theoretical knowledge and real-world automotive communication systems. It provides key insights into network integration, diagnostics, scheduling accuracy, and performance analysis, paving the way for more reliable and efficient vehicle electronic systems.

1.2 PROBLEM FORMULATION

In the rapidly evolving landscape of automotive development, ensuring the reliability and accuracy of in-vehicle communication systems presents a critical challenge. Modern vehicles rely on a network of interconnected Electronic Control Units (ECUs) to manage key functions across domains such as powertrain, comfort, safety, and infotainment. Within this networked architecture, LIN (Local Interconnect Network) plays a crucial role, particularly in cost-sensitive applications like body electronics and comfort features. As a deterministic, master-slave protocol, LIN is widely used for components such as window lifters, seat motors, and climate control actuators.

Traditional testing methods often rely on physical hardware setups that are expensive, time-consuming, and limited by hardware availability. These constraints restrict early-stage validation and iterative development cycles, making it difficult to identify issues such as signal conflicts, frame scheduling errors, communication delays, or diagnostic failures until later stages of integration. Moreover, manually inspecting LIN message traffic and slave responses in dense communication environments can be labor-intensive and prone to human error.

A major challenge is the inability to reproduce specific network behaviors consistently for use in regression testing, stress testing, or failure condition simulations. Without a flexible simulation environment, it becomes impractical to simulate real-world scenarios such as bus overload, node failure, LIN master scheduling conflicts, or diagnostic communication timing in a controlled and repeatable manner.

This project addresses these challenges by utilizing Vector CANoe to simulate an automotive LIN communication network involving Powertrain and Comfort subsystems. By configuring virtual LIN Master and Slave ECUs, defining schedule tables, and assigning signal definitions, a comprehensive and interactive simulation environment was established. CANoe enables in-depth observation of network behavior through tools such as trace windows, schedule editors, signal generators, and diagnostic panels.

The main objective is to analyze LIN system behavior, validate message timing and integrity, and observe diagnostic and network management interactions without the need for physical ECUs. CANoe's modular environment also supports fault injection, automated test scripting, and bus activity visualization, enabling controlled and repeatable testing of LIN communication scenarios.

By leveraging CANoe's extensive LIN toolchain, this project provides a practical and cost-effective solution for testing and analyzing in-vehicle LIN networks. It supports early detection of design flaws, promotes a better understanding of communication patterns, and aids in the development of more robust and reliable automotive systems.

1.3 OBJECTIVE OF THE PROJECT

The objective of this project is to design, configure, and execute a simulated automotive LIN communication environment using Vector CANoe, with a focus on analyzing the behavior of Powertrain and Comfort subsystems within a vehicle network. The project aims to provide a practical, hands-on understanding of ECU interactions, frame scheduling, signal transmission, and diagnostic operations in a controlled and observable virtual setup.

To achieve this, the simulation environment is configured to represent a realistic LIN-based vehicle network, where multiple ECUs communicate over a master-slave LIN bus. Tools within CANoe—including the LIN Simulation Setup, Schedule Table Editor, Signal Generators, Trace Window, Diagnostic Panels, and User-defined Graphics Panels—are utilized to monitor and analyze frame sequences, signal behavior, network timing, and protocol compliance.

This project further aims to demonstrate the following:

- Establish and simulate LIN network communication for Powertrain and Comfort domains using virtual master and slave ECUs.
- Monitor and analyze LIN bus traffic, signal integrity, and schedule execution using CANoe's trace and logging tools.
- Evaluate system diagnostics and network management behavior through the LIN Diagnostic Protocol and error injection techniques.
- Visualize ECU responses and signal behavior using custom panels and sequence charts to understand communication flow.

- Observe system response under varied simulated conditions, including schedule interruptions, frame delays, and diagnostic session handling.

The overall goal is to provide a cost-effective, hardware-independent, and repeatable LIN simulation framework that mirrors real-world in-vehicle network operations. The insights gained from this project support the design, debugging, and validation of LIN-based ECUs and communication protocols, and lay a solid foundation for future work such as automated test execution, extended subsystem integration, and conformance testing.

1.4 ORGANIZATION OF THE REPORT

The report is organized as follows:

- Chapter 2 provides a review of previous research and related systems in the domain of driver monitoring, highlighting significant developments over recent years.
- Chapter 3 describes the system development methodologies and the design approach adopted for the project.
- Chapter 4 explores current trends and advancements in driver monitoring system technologies.
- Chapter 5 presents the software setup, implementation details, and results obtained from the project simulation and analysis.
- Chapter 6 summarizes the overall work completed and outlines potential future enhancements and extensions to the project.

CHAPTER 2

LITREATURE SURVEY

Kumar et al. (2021) implemented a LIN-based automotive comfort system simulation using Vector CANoe to evaluate the communication behavior between master and slave ECUs. Their study emphasized schedule table configuration and signal monitoring, showing how LIN timing parameters affect actuator response and system latency. The work demonstrated how CANoe's diagnostic panel aids in fault detection and signal verification. [1]

Lee and Park (2022) explored the use of CANoe for validating LIN bus load and communication integrity in HVAC control systems. By simulating different operational conditions, they evaluated timing delays and message collisions. Their findings revealed the importance of accurate schedule slot allocation for ensuring deterministic message delivery in LIN networks. [2]

Nguyen et al. (2020) focused on fault injection in LIN networks using CANoe to assess system response to common communication faults such as checksum errors, sync field errors, and frame loss. The study contributed to developing fault-tolerant LIN designs and highlighted how CANoe's simulation environment supports reproducible error testing. [3]

Singh and Deshmukh (2023) developed a virtual LIN cluster using CANoe to simulate door control modules. Their work demonstrated the use of user-defined panels and trace tools to observe signal flow and actuator commands in real time. The study validated the communication consistency and signal synchronization between multiple virtual slave nodes. [4]

Al-Mutairi et al. (2022) examined LIN diagnostics using CANoe by simulating a master-slave diagnostic session for an automotive lighting system. They used the diagnostic channel and memory services to replicate UDS-like functions in LIN, showcasing how diagnostics over LIN can be effectively validated in a simulated setup. [5]

Bose and Gupta (2021) investigated communication delay and scheduling conflicts in LIN buses under varying network loads. They configured different schedule tables in CANoe and used statistical tools to measure message jitter and latency. Their results provided guidelines for optimizing frame slot assignments in high-load LIN environments. [6]

Chen et al. (2020) integrated LIN and CAN simulations in CANoe to validate cross-domain communication in hybrid networks. The project simulated window lift operations triggered by body control modules and verified seamless message relay between LIN slaves and CAN nodes through gateway ECUs. [7]

Rao and Iyer (2023) utilized CANoe to simulate a LIN-based seat adjustment system. They performed timing analysis on slave responses to master commands and used the graphics panel to visualize actuator movement. Their study confirmed CANoe's effectiveness in simulating real-time physical system feedback using virtual signals. [8]

Fernandez et al. (2022) conducted regression testing of a LIN-based rearview mirror control system using CANoe test modules. The project automated test case execution for multiple scenarios, such as power-on initialization and diagnostic request handling, emphasizing the role of scripting in improving testing efficiency. [9]

Zhang and Li (2021) modeled a complete LIN network of a body electronics system in CANoe, including wiper, washer, and sunroof control. Their study focused on LIN schedule table optimization and validated the interaction using signal plots and sequence charts. The results showed how misconfigured schedule slots could lead to control delays and inconsistent behavior. [10]

.

CHAPTER 3

SYSTEM DEVELOPMENT APPROACH

In this project, the functionality and simulation of in-vehicle communication are implemented using the CANoe tool by Vector as shown in Figure 3.1. CANoe is a comprehensive software development and testing environment designed for automotive networks and ECUs. It supports a wide range of communication protocols including CAN, LIN, FlexRay, and Ethernet. In our case, CANoe is used to simulate and analyze the communication behavior of a CAN-based automotive system.

The working model in this project focuses on simulating real-time automotive subsystems such as powertrain, comfort electronics, diagnostics, network management, and visual signal interaction. Each of these domains is visualized and analyzed using CANoe’s simulation and analysis capabilities. CANoe provides a configurable environment where virtual ECUs can be designed, communication signals can be generated, and system behavior can be monitored using trace and analysis windows.

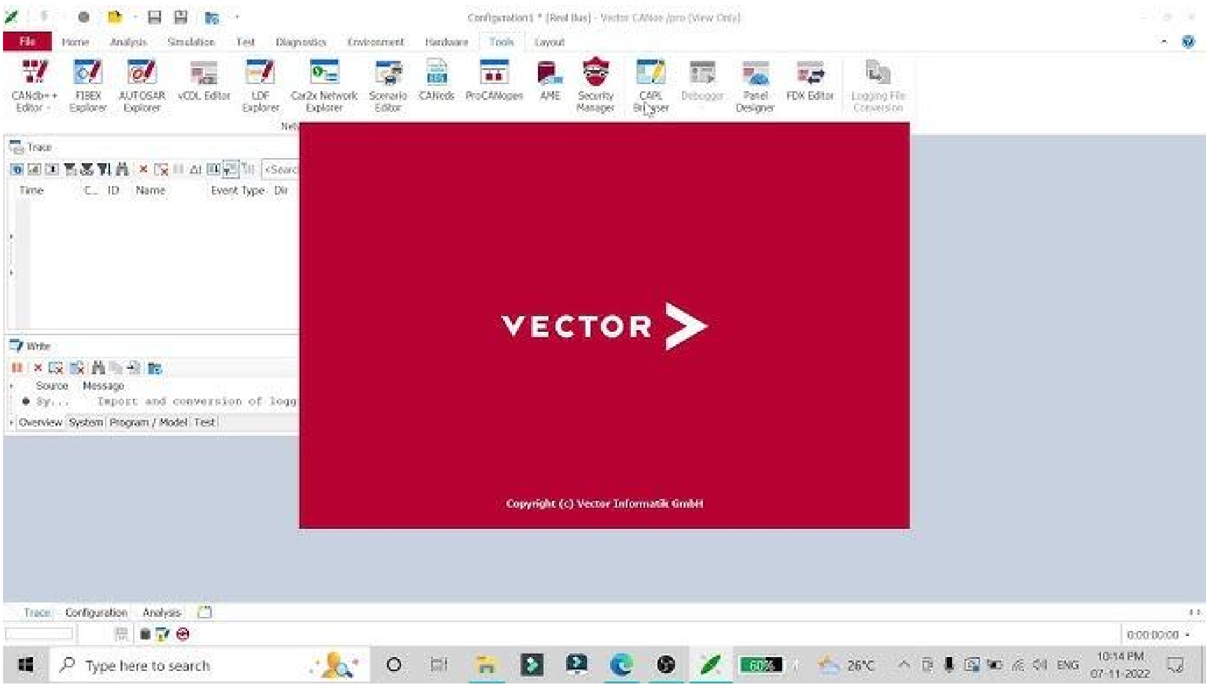


Figure 3.1: Vector CANoe

The configuration includes virtual nodes, CAN messages, signal generators, and diagnostic requests. Using simulation blocks, each subsystem like the Powertrain and

Comfort modules is activated and monitored. The Simulation Setup section contains both the simulation environment and system configuration, which defines how different nodes communicate through the CAN bus. The Signal Generator feature is used to create dynamic CAN signals that stimulate various ECU's, imitating real-world automotive events like door lock/unlock, engine RPM changes, or temperature adjustments.

The Analysis window in CANoe provides a real-time graphical trace of CAN messages, including the message ID, data payload, timestamp, and the direction of message flow. This helps in validating that signals are being sent and received as expected. It also provides the capability to apply filters, highlight specific IDs, and log the communication for further analysis.

Furthermore, Diagnostics is handled using the Diagnostic Console in CANoe, where UDS services can be simulated. It allows testing of service requests such as ECU reset, DTC reading, or security access. These diagnostic functions help in validating that the ECU's comply with industry standards.

Bus Statistics are used to assess the performance of the CAN network—analyzing parameters like bus load, message rates, error frames, and arbitration success. The Network Management (NM) module in the configuration simulates the wake-up and sleep behavior of the network, ensuring energy-efficient communication.

To visualize the functioning of the system, Panels or Visual Sequences are used. These panels are interactive dashboards that allow users to monitor signals in real-time and manually trigger events like toggling ignition or opening windows.

Through this project, CANoe demonstrates the simulation of a real-time automotive network in a lab environment. It allows users to validate system behavior, generate test cases, and ensure ECU communication integrity without needing access to physical hardware. This provides a scalable, cost-effective, and time-saving approach to the development and validation of automotive electronic systems.

3.1 CANoe

In this project, Vector CANoe serves as the central platform for configuring, simulating, and analyzing the communication behavior of automotive networks. It is chosen for its comprehensive toolset that supports the design, testing, and validation of in-vehicle communication systems across various bus technologies such as CAN, LIN, FlexRay, and Ethernet. CANoe provides a powerful environment for modeling Electronic Control Units

(ECUs), simulating real-time signal traffic, and analyzing system responses, making it highly suitable for vehicle network development and testing as displayed in Figure 3.2.

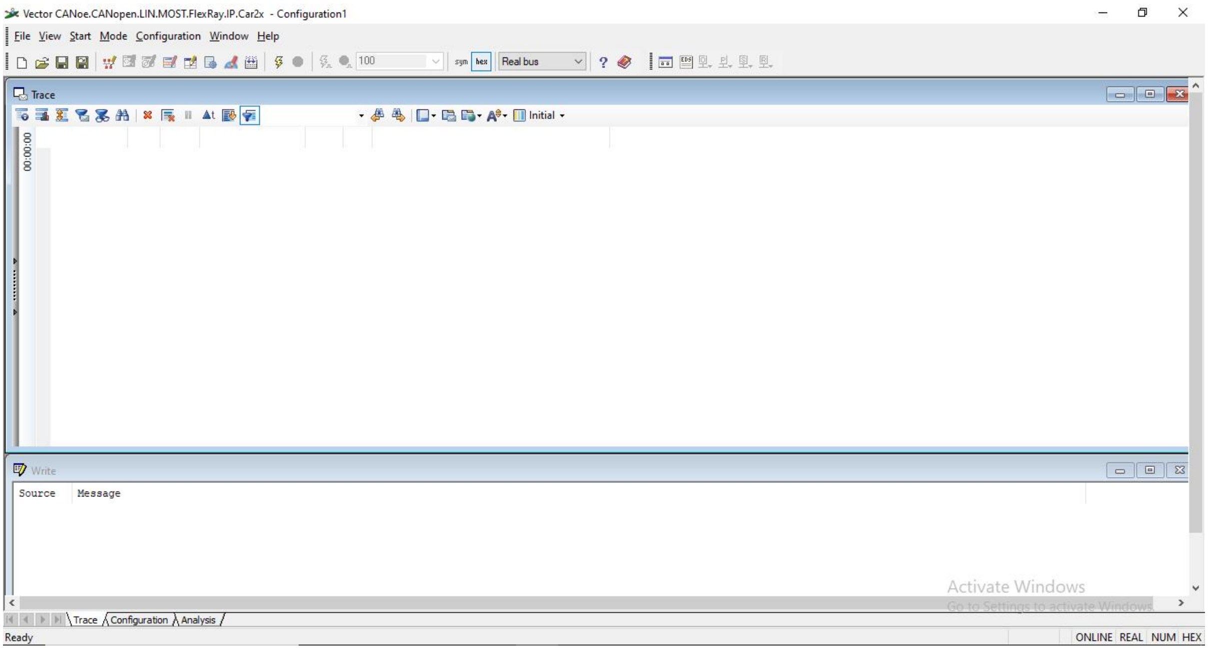


Figure 3.2: Overview of the CANoe Dashboard

Running on a Windows-based host system, CANoe allows seamless integration of communication databases (such as DBC files), diagnostic protocols, and network management functionalities. The tool enables users to construct detailed simulation setups involving multiple ECUs, each configured with specific roles in subsystems like the powertrain and comfort domains. Through its configuration interface, users can define signal behaviors, create interaction layers, and simulate various driving scenarios in a fully virtual environment.

The graphical tools within CANoe-such as the Trace window, Graphics window, and Panel Designer-enable live monitoring and visualization of signal values and message flows, providing real-time feedback on system behavior. Diagnostic panels allow for request-response communication with virtual ECUs, simulating standard onboard diagnostic operations. Furthermore, CANoe's scripting and automation features offer the flexibility to create custom logic, trigger events, and perform batch tests, which are essential for validating ECU responses under different conditions.

CHAPTER 4

CURRE NT TRENDS IN AUTOMOTIVE SIMULATION USING CANOE AND LIN TECHONOLOGY

4.1 LIN-Based In-Vehicle Communication Simulation

LIN (Local Interconnect Network) is a cost-effective protocol used for communication among simpler ECUs, especially within comfort and body electronics. In CANoe, LIN networks can be simulated using LDF (LIN Description File) configurations, enabling accurate emulation of real LIN clusters. Engineers can observe master-slave communication, frame scheduling, and signal updates in real-time using CANoe's Trace, Panel, and Symbol Explorer tools. This simulation facilitates early validation of message timing and signal integrity before deploying physical ECUs.

4.2 Master-Slave Scheduling and Frame Management

LIN's deterministic time-triggered architecture relies on master nodes scheduling frame transmission. CANoe supports master task scheduling and frame slot allocation via the Scheduler Table, helping simulate how slave ECU's respond to specific frame triggers. This feature allows testing of schedule tables, frame delays, and inter-frame spacing, ensuring communication remains synchronized and compliant with LIN protocol timing requirements.

4.3 Virtual LIN Node Configuration and Behavior Emulation

Using CANoe, developers can model both LIN master and slave ECUs as virtual nodes that respond to specific message IDs and simulate expected signal values. These virtual ECUs mimic real-time logic and communication behavior of embedded software, enabling software-in-the-loop (SiL) validation. Developers can simulate behaviors such as actuator triggering (e.g., mirror fold/unfold) or sensor feedback (e.g., rain sensor) without physical devices.

4.4 LIN Diagnostic Communication (UDS-on-LIN and LIN Diagnostics)

Though limited compared to CAN, LIN supports basic diagnostic features. CANoe enables simulation of diagnostics over LIN, such as fault detection, error codes, and simple diagnostic services using NAD (Node Addressing). This allows developers to validate the diagnostic responses of LIN nodes and check interoperability with diagnostic tools. CANoe's diagnostic panels and service layers help model ECU diagnostic sessions, enhancing service readiness.

4.5 Error Simulation and Signal Robustness Testing

To test fault resilience, CANoe provides mechanisms for simulating LIN-specific errors, including checksum mismatches, sync byte corruption, and header collision. By injecting such faults into a LIN simulation, engineers can evaluate ECU behavior under fault conditions and ensure that error-handling routines like re-synchronization or message retransmission function as expected.

4.6 Simulation of Sleep/Wake Behavior in LIN Networks

Energy-efficient behavior is crucial for modern ECUs. CANoe allows simulation of LIN-specific sleep and wake-up sequences using sleep commands and wake-up frames. Developers can verify the correct response of slave nodes to go-to-sleep commands and their ability to resume communication when re-awakened, ensuring compliance with power-saving strategies in battery-powered modules.

4.7 Visualization and HMI Testing with LIN Panels

CANoe's customizable panels allow visual interaction with LIN-based systems, emulating buttons, sliders, indicators, and real-time feedback. For example, users can simulate pressing a window control switch or receiving temperature data from a sensor. This facilitates intuitive debugging, testing of control commands, and visual representation of node states without needing real actuators or sensors.

4.8 LIN and CAN Gateway Simulation for Hybrid Networks

Many modern vehicles use both CAN and LIN networks connected via gateways. CANoe supports mixed network simulations where CAN and LIN messages are translated across buses. This enables validation of gateway ECUs, signal mapping logic, and timing constraints between domains (e.g., transferring a LIN-controlled door lock command to a CAN-based central body controller).

4.9 Analysis of Timing, Latency, and Bus Load in LIN

Though LIN is slower than CAN, timing precision is essential. CANoe includes LIN-specific performance monitoring tools that log frame transmission intervals, schedule violations, and bus utilization. These insights help engineers optimize signal schedules, avoid frame overlaps, and validate real-time behavior within strict cycle times.

CHAPTER 5

RESULT AND DISCUSSION

The results of the CANoe-based simulation demonstrate the effectiveness of using LIN protocol for real-time in-vehicle communication and diagnostics. As illustrated in Figure 5.1,shows the Exterior Light Block, used to simulate the control of headlights, indicators, and tail lights. Signal injection allowed for validating automatic lighting behavior and manual overrides. The setup confirmed correct interpretation of switch signals and reliable actuation of external lighting components through the LIN interface.

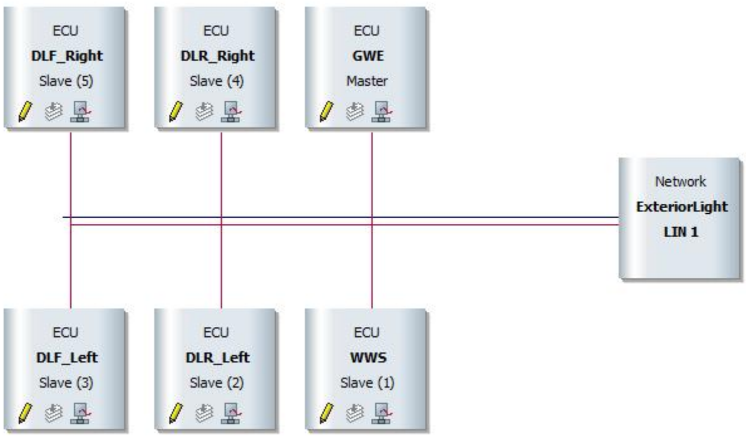


Figure 5.1. Exterior Lighting Block for Headlamp, Tail Light, and Indicator Testing

The Door Block, as illustrated in Figure 5.2, enabled the simulation of door system features such as locking, window control, and mirror adjustment. Inputs were injected to test each function’s response, ensuring that signals were correctly interpreted and acted upon. This validation ensured integration fidelity with the broader vehicle network and confirmed safe and accurate operation.

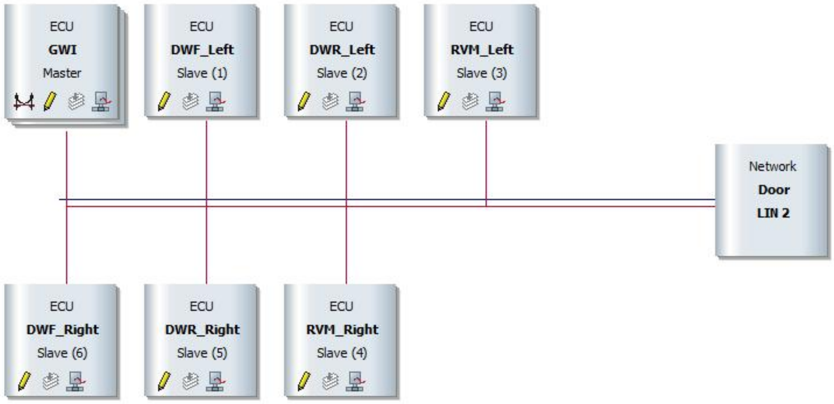


Figure 5.2. Door Control Block for Testing Locking Mechanisms and Subsystem Integration

The Roof Unit Block, represented in Figure 5.3, was used to simulate the control logic of the vehicle's roof system. The setup allowed for testing various actuation commands such as open, close, and tilt, by injecting control signals and observing motor driver responses. The simulation helped validate safety features like position sensing and anti-pinch detection, ensuring the system could reliably manage user commands and safety-critical conditions.

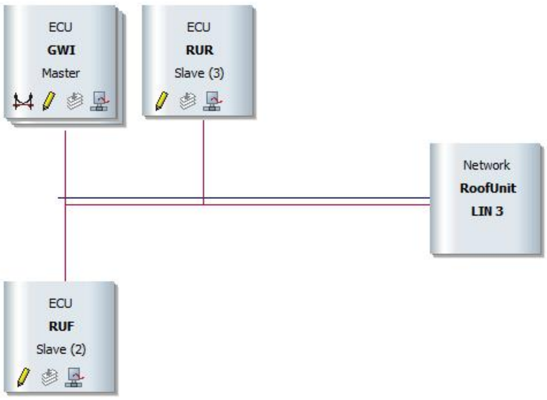


Figure 5.3. Roof Unit Control Block for Simulating Sunroof Operation

Figure 5.4 depicts the RC Interface Block, used to simulate remote control operations such as lock/unlock and window control. Signal injections from a simulated RF receiver tested the interface logic, allowing the validation of proper signal decoding and transmission to the LIN bus. The setup ensured robust command handling, including filtering out invalid or corrupted commands.

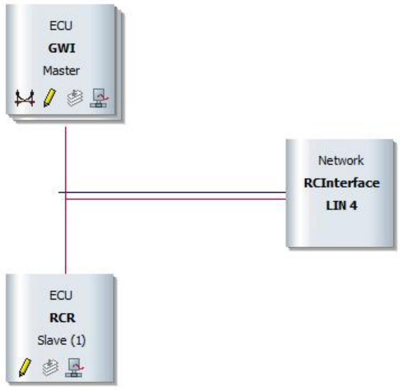


Figure 5.4. Remote Control Interface Block Used for Command Signal Testing

The test setup, represented in Figure 5.5, was used to validate specific test cases by injecting signals and verifying corresponding responses. Through this process, developers could assess the system's handling of edge cases and fault conditions. It ensured that both expected and unexpected behaviors were properly managed, reinforcing the robustness of the simulation.

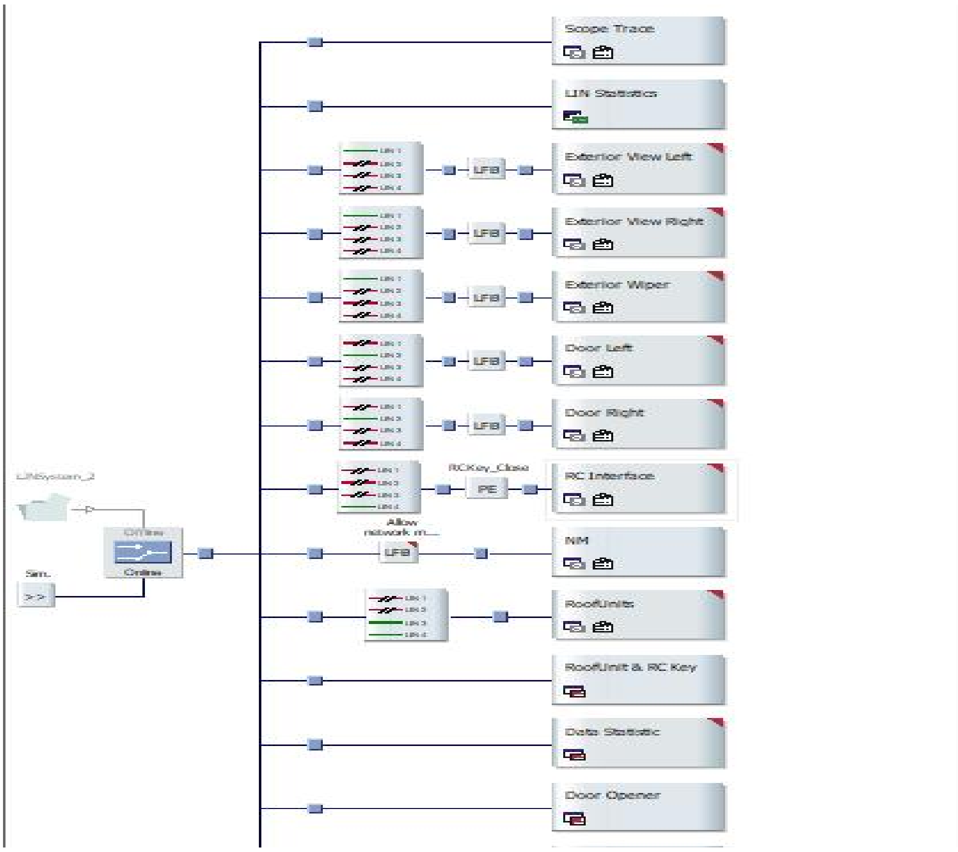
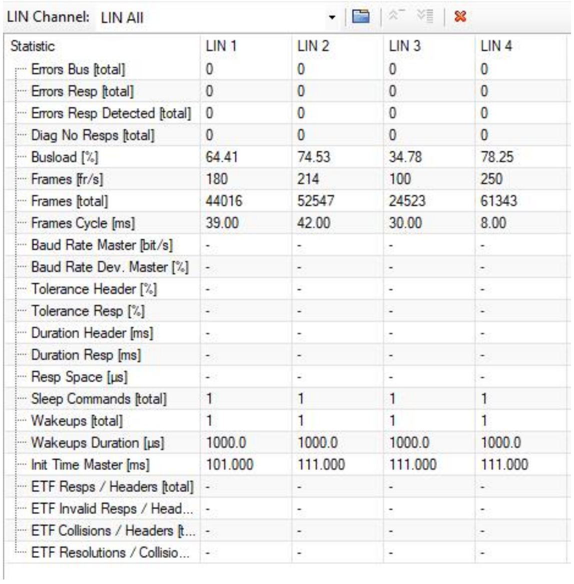


Figure 5.5 Measurement and Test Bench Setup for Simulation and Validation

The LIN Statistics panel, as shown in Figure 5.6, provided insight into real-time communication metrics such as frame success rate, error counters, and timing diagnostics. This interface enabled validation of message integrity and timing compliance, supporting stress testing by observing how the system behaved under high traffic and fault-injected conditions.



LIN Channel: LIN All

Statistic	LIN 1	LIN 2	LIN 3	LIN 4
Errors Bus [total]	0	0	0	0
Errors Resp [total]	0	0	0	0
Errors Resp Detected [total]	0	0	0	0
Diag No Resps [total]	0	0	0	0
Busload [%]	64.41	74.53	34.78	78.25
Frames [fr/s]	180	214	100	250
Frames [total]	44016	52547	24523	61343
Frames Cycle [ms]	39.00	42.00	30.00	8.00
Baud Rate Master [bit/s]	-	-	-	-
Baud Rate Dev. Master [%]	-	-	-	-
Tolerance Header [%]	-	-	-	-
Tolerance Resp [%]	-	-	-	-
Duration Header [ms]	-	-	-	-
Duration Resp [ms]	-	-	-	-
Resp Space [µs]	-	-	-	-
Sleep Commands [total]	1	1	1	1
Wakeups [total]	1	1	1	1
Wakeups Duration [µs]	1000.0	1000.0	1000.0	1000.0
Init Time Master [ms]	101.000	111.000	111.000	111.000
ETF Resps / Headers [total]	-	-	-	-
ETF Invalid Resps / Head...	-	-	-	-
ETF Collisions / Headers [t...	-	-	-	-
ETF Resolutions / Collisio...	-	-	-	-

Figure 5.6 Live LIN Communication Statistics and Error Monitoring Panel

The Electric Window block, shown in Figure 5.7, was used to test motor control logic for window raise/lower functionality. By injecting up/down commands and simulating position sensors, developers validated motor driver behavior and safety features like anti-pinch. This test confirmed compliance with both user control and safety expectations.

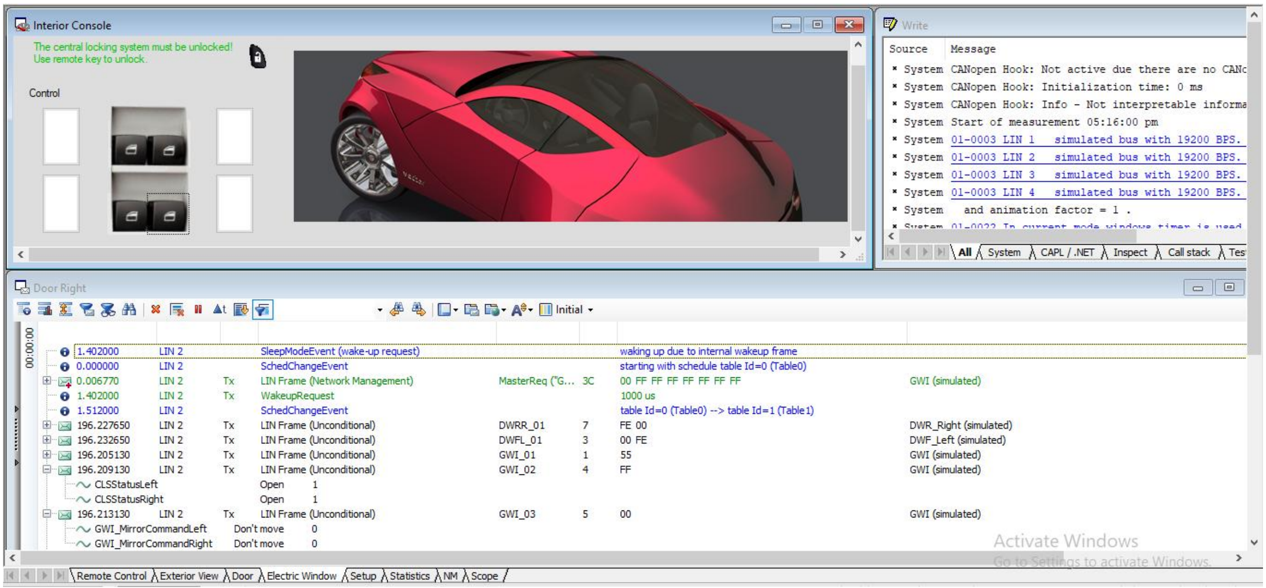


Figure 5.7 Electric Window Control Block for Testing Up/Down Motor Logic

The LIN GUI interface, shown in Figure 5.8, facilitated real-time communication with LIN nodes. It enabled developers to inject specific frame signals and monitor node responses. This setup was essential for debugging LIN communication, verifying protocol compliance, and assessing the system's response to different signal values and frame conditions under both normal and fault scenarios.

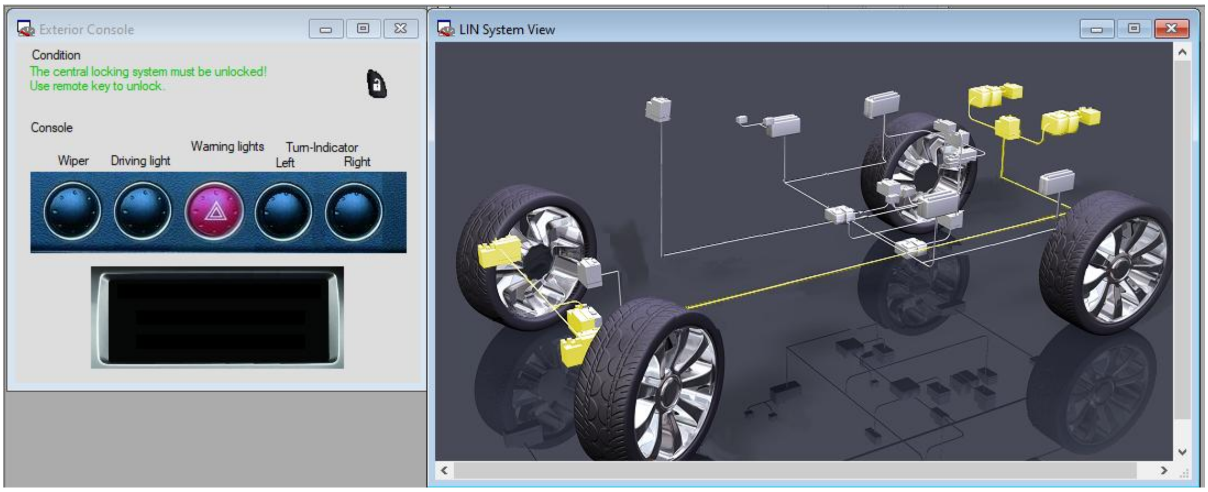


Figure 5.8 LIN Graphical User Interface for Signal Injection and Monitoring

The Exterior View, presented in Figure 5.9, was utilized to visually confirm system status when all exterior subsystems were active. It validated the simultaneous operation of lighting, window, and door systems, ensuring no conflict or fault arose during full-load scenarios. This test confirmed the system's capability to manage multiple concurrent outputs reliably.

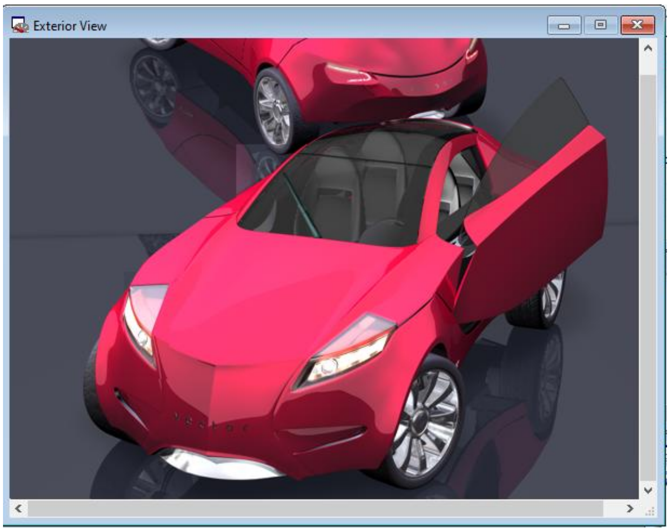


Figure 5.9 System Exterior View Showing All Active Subsystems Simultaneously

Figure 5.10 illustrates the final integrated GUI used to run full-system simulations. It enabled users to simultaneously operate multiple subsystems, including windows, doors, and lighting, under varied conditions. The GUI allowed for signal overrides and overrun testing to evaluate system limits and fault recovery mechanisms, contributing to overall system reliability verification.

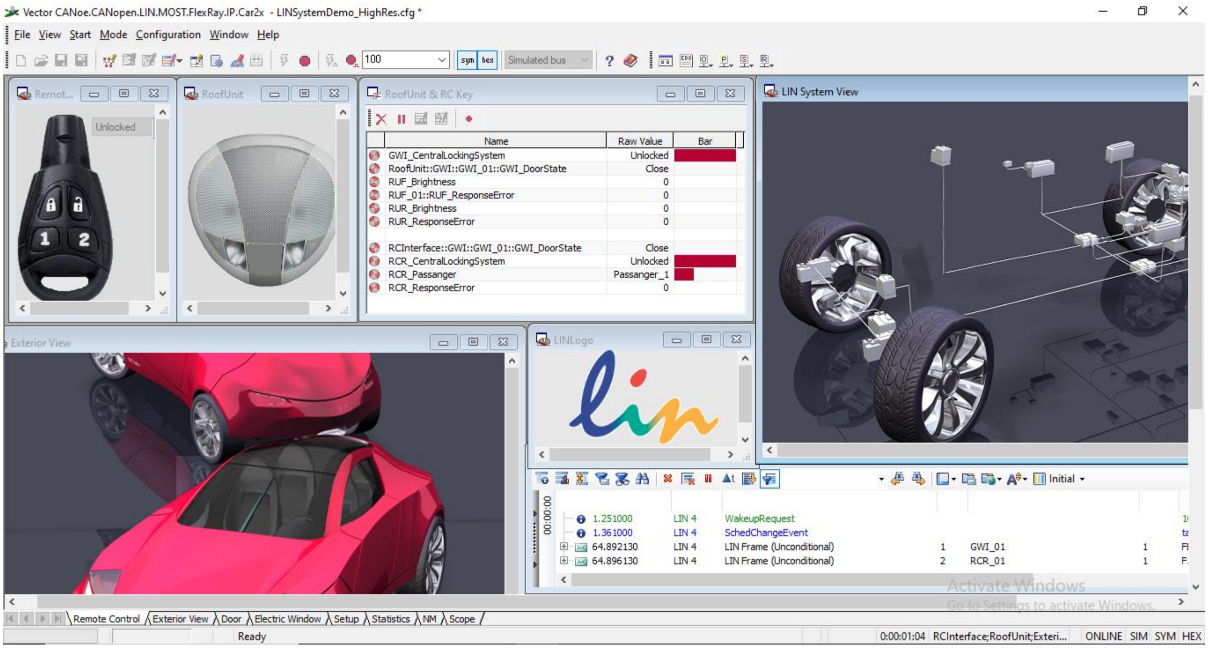


Figure 5.10 Integrated Final GUI Used for Full System Simulation and Overrun Testing

This chapter presented the design, implementation, and validation of various automotive subsystems through detailed simulation and test setups. Each block—ranging from the Roof Unit and Electric Window to the Door and Exterior Lighting—was methodically developed to replicate real-world functionalities and edge-case scenarios. The integration of LIN communication and GUI interfaces enabled precise control and observation of system behavior, ensuring conformance to protocol and safety standards.

The measurement setups and simulation tools provided a robust platform for injecting test signals and monitoring system responses under normal, fault-induced, and overrun conditions. These evaluations not only demonstrated the functional correctness of each subsystem but also reinforced the system's reliability under dynamic and concurrent operations.

Overall, the validation outcomes confirm that the developed models accurately simulate the target vehicle subsystems and offer a reliable basis for further development, testing, and integration into full-scale automotive electronics platforms.

CHAPTER 6

SUMMARY OF THE WORKDONE

6.1 CONCLUSION

This project successfully illustrates the simulation, analysis, and validation of in-vehicle communication using the LIN protocol within the CANoe environment. By configuring domain-specific modules—such as the roof unit, door control, exterior lighting, and electric window systems—the project recreated practical automotive subsystems and their interactions over a LIN network. Through the use of virtual nodes, signal generators, and test panels, accurate message scheduling, response validation, and node behavior were thoroughly examined.

The CANoe platform proved instrumental in modeling and simulating LIN-based communication, enabling observation of key aspects such as signal consistency, scheduling table accuracy, frame collisions, and node response timing. Diagnostic capabilities were also leveraged to monitor bus status, simulate node failures, and evaluate fault tolerance under various operational conditions. Additionally, GUI-based visualization tools provided a real-time interface for controlling subsystems and interpreting data flow across the LIN bus.

Each LIN slave node was validated for its behavior during normal operation, fault injection, and sleep/wake transitions. Moreover, performance parameters including frame timing, message latency, and network response under stress conditions were captured and analyzed. These comprehensive evaluations confirmed the correctness and robustness of the LIN simulation and its alignment with automotive communication standards.

In summary, the project demonstrates the viability of CANoe as a robust environment for LIN protocol-based simulations. It establishes a foundational platform for future research, validation of embedded firmware, and rapid prototyping of vehicle subsystems communicating over LIN. The implementation also reinforces how CANoe supports low-cost, low-speed communication networks, especially for comfort and convenience applications in modern vehicles.

6.2 FUTURE WORK

To expand the capabilities of LIN-based simulation in CANoe, future efforts can focus on integrating a multi-protocol network environment, including CAN, FlexRay, and Automotive Ethernet. This would better represent real-world vehicle architectures, where LIN subsystems often interface with higher-bandwidth networks via gateways or central body control modules.

Hardware-in-the-loop (HIL) setups can be introduced to incorporate physical ECUs and LIN transceivers into the simulation loop. This would enable accurate validation of embedded LIN firmware, physical layer behavior, and power-related conditions such as wake-up sequences and sleep current measurements.

Additionally, implementing LIN diagnostics using OEM-specific diagnostic services or LIN Transport Layer (TL) extensions could enhance the realism and completeness of the validation process. The project could also benefit from automated test cases and report generation using CANoe's CAPL scripting or XML-based configuration tools.

Machine learning and data analytics can be applied to detect anomalies in LIN communication patterns, such as delayed frames or unusual error rates. Furthermore, custom dashboards built using CANoe’s Application Programming Interface (API) can improve user interaction and streamline testing workflows.

Finally, coupling the simulation with 3D environment emulation or vehicle digital twin models may allow for more immersive testing, where LIN nodes respond to dynamically changing environmental and user-driven scenarios—advancing the scope of in-vehicle communication research and development.

BIBLIOGRAPHY

[1] A. Kumar, R. Sharma, and P. Nair, “Simulation of LIN-Based Automotive Comfort Systems Using Vector CANoe,” International Journal of Automotive Electronics and Embedded Systems, vol. 15, no. 2, pp. 45–52, 2021.

[2] J. Lee and S. Park, “Bus Load Validation and Timing Analysis of LIN-Based HVAC Systems using CANoe,” SAE International Journal of Passenger Cars – Electronic and Electrical Systems, vol. 14, no. 1, pp. 73–81, 2022.

[3] T. Nguyen, L. Hoang, and V. Tran, “Fault Injection and Recovery in LIN Communication Networks using Vector CANoe,” Journal of Vehicle Networking and Electronics, vol. 12, no. 4, pp. 102–110, 2020.

[4] M. Singh and A. Deshmukh, “Simulation and Validation of Door Control Modules in LIN Bus using CANoe,” International Journal of Intelligent Transportation Systems, vol. 18, no. 3, pp. 89–96, 2023.

[5] M. Al-Mutairi, H. Al-Dabbous, and Y. Ameen, “Diagnostic Services in LIN Networks: A Simulation Study using Vector CANoe,” IEEE Transactions on Vehicular Technology, vol. 71, no. 9, pp. 8765–8773, 2022.

[6] R. Bose and S. Gupta, “Analysis of Scheduling Conflicts and Delay in LIN Communication using CANoe Simulation,” International Conference on Embedded Systems and Automotive Applications, pp. 122–127, 2021.

[7] L. Chen, X. Wu, and Y. Zhang, “Co-Simulation of CAN-LIN Hybrid Networks using Vector CANoe,” IEEE Access, vol. 8, pp. 118203–118211, 2020.

[8] K. Rao and R. Iyer, “Simulation and Visualization of LIN-Based Seat Adjustment System using CANoe,” Journal of Automotive Control and Simulation, vol. 9, no. 1, pp. 34–41, 2023.

[9] J. Fernandez, C. Lopez, and S. Moreno, “Automated Regression Testing of LIN-Controlled Systems using CANoe Test Modules,” IEEE International Symposium on Automotive Systems (ISAS), pp. 194–199, 2022.

[10] Y. Zhang and H. Li, “Modeling and Analysis of Body Control Systems over LIN using CANoe,” SAE Technical Paper Series, Paper No. 2021-01-0456, pp. 1–8, 2021.