People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

UNIVERSITY OF KASDI MERBAH OUARGLA

Faculty of New Information and Communication Technologies

Department of Electronics and Communications



Academic Master Thesis

Domain: Science and Technology

Branch: Electronics

Specialty: Electronics of Embedded Systems



TOPIC

Design and Implementation of a Communication Protocol for Electric Vehicle Systems

---

Presented by

- Faical BELAID
- Soufiane BOUZIANE

Presented on June 9th, 2026, in front of the jury composed of:

Mr. Kaled BENATMANE	MCB at UKM OUARGLA	President
Ms. Samia GAMOUH	MCA at UKM OUARGLA	Examiner
Ms. Nadjla BETTAYEB	MCB at UKM OUARGLA	Supervisor
Ms. Salma TCHOKETCH KEBIR	MRB at CRTI ALGERIA	Co-Supervisor

University Year: 2025/2026

Acknowledgements

This thesis marks the culmination of a long and enriching academic journey. We express our deepest gratitude to Almighty ***God*** for His guidance and support, which enabled us to complete this work. By His grace, we overcame difficulties and reached this important milestone in our studies.

We extend our sincere thanks to our thesis supervisor, ***Dr. Nadjla BETTAYEB***, for her constant availability, invaluable advice, and unwavering support throughout the preparation of this thesis.

We also express our heartfelt gratitude to ***Dr. Selma TCHOKETCH-KEBIR***, from the ***Research Center in Industrial Technology (CRTI)***, for her assistance, wise counsel, and continuous encouragement during this period.

We wish to express our sincere gratitude to the jury members, ***Dr. Samia GAMOUH*** and ***Dr. Khaled BENATHMANE***, for kindly agreeing to evaluate our work.

We are honored by their presence and grateful for the time they dedicated to reviewing this thesis, as well as for their valuable comments and suggestions. Thanks to their guidance, we were able to carry out this work with greater confidence and renewed enthusiasm.

We are also deeply grateful to all the professors and administrative staff of the ***Department of Electronics*** of our University ***(UKMO),*** for their support over the years, as well as to the team of ***Embedded System Laboratory*** at ***CRTI***.

Personal Dedication

I dedicate this work with all my love and gratitude:

First and foremost, to **GOD Almighty**, for His guidance, protection, and support throughout this work.

To my parents, for their unwavering love, patience, and all the sacrifices they have made for me.

To my uncles and aunts, who have always been a great source of support and encouragement.

To my brothers and sisters, for their constant presence, encouragement, and unwavering support.

To my friends, for their companionship and motivation.

To our supervisor, for her trust, guidance, and continuous support throughout this work. This work is also a reflection of her invaluable contribution.

Finally, to **BOUZIANE Soufiane**, my partner in this project, for his wonderful collaboration, patience, trust, and all the unforgettable moments we have shared during this work.

Finally, a special thank you to myself for my perseverance and dedication.

Faycal

Personal Dedication

I dedicate this thesis with all my love and gratitude:

To GOD Almighty, for His guidance, support, and protection throughout this journey.

To my parents, for their immense love, patience, and all the sacrifices they have made for me.

To my siblings, for their unwavering support, presence, and encouragement.

To my grandparents, whose prayers and love are always with me.

To my cousins, **for their joy**, support, and kind words.

To my friends, **for their presence**, humor, and encouragement, even during the most difficult times.

To our supervisor, for her trust, constant presence, and wise guidance.

Soufiane

ملخص

يُشكّل تطوير المركبات الكهربائية تحديات جديدة لأنظمة الاتصالات المتكاملة. وينبع هذا من تزايد عدد المكونات الإلكترونية واعتمادها على بروتوكولات اتصـــال متعددة، بما في ذلك SPI وLIN وCAN وFlexRay وMOST، لضـمان تبادل البيانات بكفاءة وموثوقية بين الأنظمة الفرعية. يعرض هذا العمل تصـميم وتنفيذ نموذج أولي مُدمج مع نظام اتصـال خاص به لمركبة كهربائية، مثل نظام مسـاحات الزجاج الأمامي. يُمكّن هذا النموذج الأولي من التواصـل بين المكونات الإلكترونية الرئيسية، بما في ذلك محرك المساحات، وأجهزة الاستشعار، والمضخة. والهدف هو تحسين السلامة، وتقليل إجهاد السائق، وعرض حالة تشغيل محرك المساحات للمستخدم على شاشة. في هذه الدراسة، قمنا بتقييم أوقات استجابة مستشعر المطر، ومستشعر الزر، والمضخة، والمحرك باستخدام بروتوكول CAN. كما قمنا بتحسين كفاءة النظام من خلال تصميم لوحة دوائر مطبوعة للنموذج الأولي.

الكلمات المفتاحية:

المركبات الكهربائية، الأنظمة المدمجة، ماســحات الزجاج الأمامي، بروتوكولات الاتصـــال، ناقل CAN، تصـــميم لوحات الدوائر المطبوعة.

Résumé

Le développement des véhicules électriques (VE) pose de nouveaux défis aux systèmes de communication intégrés. Ceci résulte du nombre croissant de composants électroniques et de leur dépendance à de multiples protocoles de communication, notamment SPI, LIN, CAN, FlexRay et MOST, afin de garantir un échange de données efficace et fiable entre les sous-systèmes. Ce travail présente la conception et la mise en œuvre d'un prototype embarqué muni de son système de communication pour un VE, tel que le système à essuie-glaces permettant le nettoyage de vitres du VE. Ce prototype permet la liaison de principaux composants électroniques notamment un moteur à essuie-glace, des capteurs et une pompe. L'objectif est d'améliorer la sécurité, de réduire la fatigue du conducteur et d'indiquer à l'utilisateur l'état de fonctionnement du moteur à essuie-glace sur un écran. Dans cette étude, nous avons évalué les temps de réponse du capteur de pluie, d'un bouton, de la pompe et du moteur à l'aide du protocole CAN. Nous avons également optimisé l'efficacité du système en concevant un circuit imprimé (PCB) pour le prototype.

Mots clés :

Véhicules électriques, Systèmes embarqués, Essuie-glaces, Protocoles de communications, Bus CAN, Conception PCB.

Abstract:

The development of electric vehicles (EVs) poses new challenges to integrated communication systems. This stems from the increasing number of electronic components and their reliance on multiple communication protocols, including SPI, LIN, CAN, FlexRay, and MOST, to ensure efficient and reliable data exchange between subsystems. This work presents the design and implementation of an embedded prototype with its communication system for an EV, such as the windshield wipers system for cleaning the EV's windows. This prototype enables the communication of key electronic components, including a wiper motor, sensors, and a pump. The objective is to improve safety, reduce driver fatigue, and display the wiper motor's operating states to the user on a screen. In this study, we evaluated the response times of the rain sensor, the button sensor, the pump, and the motor using the CAN protocol. We also optimized the system's efficiency by designing a printed circuit board (PCB) for the prototype.

Keywords:

Electric vehicles, Embedded Systems, Windshield wipers, Communication protocols, CAN bus, PCB design.

CONTENTS

Acknowledgements **i**

Dedication **ii**

Abstract **iv**

List of Tables **viii**

List of figures **ix**

List of Abbreviations **xi**

General Introduction **1**

1 Embedded Systems(ES) in Electric Vehicles(EV) **2**

1.1 Introduction 3

1.2 Electric Vehicles 3

1.3 Embedded Systems in EVs 4

1.3.1 Types of ECUs in EV 5

1.3.2 Embedded On-Board Diagnosis (OBD) tool 9

1.4 Conclusion 12

2 Networking in EV and Communications Protocols **13**

2.1 Introduction 14

2.2 CAN 14
2.2.1 History 14
2.2.2 Principal 15
2.2.3 Constitution of a CAN Node 16
2.2.4 CAN Message Structure 17
2.2.5 CAN's features 19
2.2.6 CAN's Advantages 19
2.2.7 Applications 20
2.3 LIN 21
2.3.1 History 21
2.3.2 Protocol principle 21
2.3.3 Communication System (Master–Slave Architecture) 22
2.3.4 Physical Layer 23
2.3.5 LIN BUS Threshold Voltage Level 24
2.3.6 Message frame format 25
2.3.7 Advantages and Disadvantages 27
2.3.8 Applications 28
2.3.9 Comparison between CAN and LIN networks 29
2.4 FlexRay Protocol 30
2.4.1 FlexRay Topology and Layout 31
2.4.2 Applications of FlexRay 33
2.5 MOST Protocol 34
2.5.1 MOST Protocol Architecture 34
2.6 Conclusion 35

3 Design and Implementation of the Prototype 36
3.1 Introduction 37
3.2 System Architecture 37
3.3 Description of materials used 37
3.3.1 STM32F103C8T6 (Blue Pill): 37
3.3.2 MCP2515 39
3.3.3 Breadboard (Hitchman): 40
3.3.4 LCD I2C: 41
3.3.5 Relay Module 5V 42
3.3.6 Button: 43

3.3.7 Pump: . . . 43
3.3.8 Power source: . . . 44
3.3.9 Rain Sensor Module: . . . 45
3.3.10 Motor DC: . . . 46
3.3.11 L298N Motor Driver: . . . 46
3.4 Development tools . . . 47
3.5 Electronic Connection Diagram . . . 49
3.6 Detailed description of the enhanced integrated systems . . . 50
3.6.1 Windshield cleaning system: . . . 50
3.6.1.1 Connecting the button to the STM32 . . . 51
3.6.1.2 Connecting the rain sensor to the STM32 . . . 51
3.6.1.3 Connecting an LCD screen to an STM32 . . . 52
3.6.1.4 Connecting an electric motor to an STM32 controller . 52
3.6.2 Rear window cleaning system: . . . 53
3.6.2.1 Connecting an LCD screen to an STM32 . . . 54
3.6.3 Discussion of the results obtained: . . . 55
3.7 PCB design and manufacturing . . . 58
3.7.1 Node 1 . . . 59
3.7.1.1 Parts 1 . . . 59
3.7.1.2 Parts 2 . . . 61
3.7.2 Node 2 . . . 62
3.7.2.1 Parts 1 . . . 62
3.7.2.2 Parts 2 . . . 63
3.8 Conclusion . . . 65

General Conclusion **66**

Bibliography **67**

LIST OF TABLES

2.1 Difference between CAN and LIN bus protocols[22] 30

3.1 Connection between STM32 and CAN Module 49

LIST OF FIGURES

1.1 Essential components of an EV (Motor, Battery, and Power-Electronics). 3
1.2 Types of Embedded Systems Application in Automobile[3]. 4
1.3 Constitution of Embedded Systems in EV 5
1.4 Essential units constituting the architecture of the EV 5
1.5 Example of an EV motor equipped with its control unit (MCU) [4] . . . 6
1.6 Block diagram of a BMU-based architecture in EVs. 7
1.7 Example of a BMS [5]. 7
1.8 Synoptic diagram of PDU [6]. 8
1.9 Synoptic diagram of ECUs with communication protocols. 9
1.10 Representation of the embedded on-board diagnosis in EV 9
1.11 Diagnosis communication architecture of an EV using CAN-based bus and UDS protocol. 10
1.12 Diagnosis methods in communication protocols. 11

2.1 Automotive CAN BUS networking of different ECUs [9]. 14
2.2 Typical CAN Network [11]. 16
2.3 Basic constitution elements of a CAN node [12]. 17
2.4 The CAN standard frame format [13]. 18
2.5 Reduction of Wiring in EV through CAN Networks [14] 20
2.6 LIN Network with Master Slave Communication [19] 22
2.7 Master–Slave Architecture of LIN bus [20]. 23

2.8 Lin Physical Layer Design [21]. 24
2.9 LIN Bus-based Threshold Level 24
2.10 LIN bus frame. 25
2.11 LIN bus frame Sync Break Field (SBF). 25
2.12 LIN bus frame Protected ID Field (SYNC). 26
2.13 LIN bus frame: Protected ID Field (PID) 26
2.14 LIN bus frame Data Field 27
2.15 LIN bus frame Checksum Field. 27
2.16 In-Vehicle LIN Protocol Network.. 29
2.17 Multi-drop bus network. 31
2.18 Star network. 32
2.19 Hybrid network. 33
2.20 MOST Network Protocol. 34

3.1 A diagram of an integrated manual and automated control system for glass spaces using the CAN protocol. 37
3.2 STM32F103C8T6 (the blue pill). 38
3.3 CAN (MCP2515). 40
3.4 Breadboard. 41
3.5 LCD I2C. 41
3.6 relay5V 42
3.7 Button. 43
3.8 pump 44
3.9 power source. 44
3.10 Rain Sensor Module. 45
3.11 motor. 46
3.12 Motor Driver. 47
3.13 FT232RL chip 48
3.14 Connecting FT232RL to a CAN network (MCP12515). 48
3.15 Interfacing between CAN and STM32. 49
3.16 Windshield wiper structure via CAN system. 50
3.17 Connecting the button to the STM32. 51
3.18 Connecting the rain sensor to the STM32. 52
3.19 Connecting an LCD screen to an STM32. 52
3.20 Connecting an electric motor to an STM32 controller. 53

3.21 Rear window wiper structure via CAN system.. 54
3.22 Connecting the button to the STM32. 54
3.23 Display the status of the pump and motor. 55
3.24 Result of pressing the button from the sender. 55
3.25 Result of pressing the button on the receiver. 56
3.26 Display the status of the motor and rain sensor 56
3.27 Rain sensor result from receiver control unit. 57
3.28 The system in normal mode. 57
3.29 Normal condition. 58
3.30 EasyEDA program interface [25] 59
3.31 Windshield wiper control system. 60
3.32 PCB wiring. 61
3.33 3D drawing of the pcb for node 1. 62
3.34 Rear window wiper control system. 63
3.35 PCB wiring. 64
3.36 3D drawing of the pcb for node 2. 64

List of acronyms and abbreviations

ABS: Anti-lock Braking System

ACK: Acknowledgement

AC: Alternating Current

ADAS: Advanced Driver Assistance Systems

BMS: Battery Management System

BMU: Battery Management Unit

CAN: Controller Area Network

CRC: Cyclic Redundancy Check

DC: Direct Current

DC-AC: Direct Current to Alternating Current Converter

DC-DC: Direct_Current-to-Direct_Current Converter

DLC: Data Length Code

ECU: Electronic Control Unit

ECUs: Electronic Control Units

EES: Embedded Electronic Systems

ESP: Electronic Stability Program

ESs: Embedded Systems

EVs: Electric Vehicles

Flexray: Flexible Ray

FTDI: Future Technology Devices International

GPS: Global Positioning System

HMI: Human-Machine Interface

ID: Identifies

IDE: Identifier Extension

LIN: Local Interconnect Network

MCU: Motor Control Unit

MOST: Media Oriented Systems Transport

OBC: On-Board Charger

OBD: On-Board Diagnosis

OBD-II: On-Board Diagnosis II

PDU: Power Distribution Unit

PID: Protected ID Field

PLC: Programmable Logic Controllers

POF: Plastic Optical Fiber

RTR: Remote Transmission Request

RX: Receive

SBF: Sync Break Field

SCI: Serial Communication Interface

SOC: State of Charge

SOF: Start of Frame

SOH: State of Health

STM32: STMicroelectronics 32-bit Microcontroller

TCU: Telemetry Control Unit

TTP: Time Triggered Protocol

TX: Transmit

UART: Universal Asynchronous Receiver Transmitter

UDS: Unified Diagnostic Services

USB : Universal Serial Bus

VCU : Vehicle Control Unit

General introduction

The automotive industry has undergone a radical transformation in recent years with the rapid adoption of electric vehicles (EVs), which have emerged as a reliable and sustainable alternative to conventional internal combustion engine vehicles. Thanks to their high energy efficiency and reduced environmental impact, EVs are considered a cornerstone of sustainable mobility in the future.

With the rapid development of modern EVs, reliance on integrated in-vehicle systems has become essential to optimal performance, safety, and energy efficiency. Modern EVs contain numerous electronic control units (ECUs) that manage vital functions such as the battery management system, the electric motor, the braking system, and various sensors. To ensure these units operate harmoniously, reliable and fast communication protocols for real-time data exchange are necessary. Among the most prominent protocols currently in use are CAN (Controller Area Network), LIN (Local Interconnect Network), and Ethernet.

This project aimed to design and implement a communication system primarily based on the CAN bus to simulate a miniature electric vehicle prototype . This was achieved using STM32 microcontrollers (Blue Pill-F103C8T6), connected via a CAN network using two MCP2515 modules. The windshield wiper system was used as a practical application.

Our thesis includes the following chapters:

- **The first chapter :** Introduces basic concepts about electric vehicles.
- **The second chapter :** Describes Networking in EV and Communications Protocols.
- **The third chapter :** Explains the system design, code writing, and establishment of the CAN network between nodes.

CHAPTER

1

EMBEDDED SYSTEMS(ES) IN ELECTRIC VEHICLES(EV)

1.1 Introduction

Recently, the automotive industry has shifted toward electric vehicles (EVs), which rely on embedded systems (ES) at the core of their electronic architecture for energy management, motor regulation, and safety.This chapter aims to introduce the fundamental concepts of embedded systems in EVs. It also discusses the different types of ESs that constitute the EV, along with a communication-based bus for their networking.

1.2 Electric Vehicles

The automotive industry has undergone a significant transition with the emergence of electric vehicles (EVs), driven by the need to reduce greenhouse gas emissions and reliance on fossil fuels EVs, powered by rechargeable batteries, offering an eco-friendly alternative to conventional internal combustion engine vehicles [1]. However, their design relies on a complex architecture of electronic and electromechanical subsystems interconnected through advanced communication protocols, which play a critical role in monitoring energy management, battery supervision, motor speed regulation, and real-time passenger safety [2]. In the era of smart mobility, embedded and electronic systems have become essential components for the development and performance optimization of electric vehicles (EVs), as illustrated in Figure 1.1

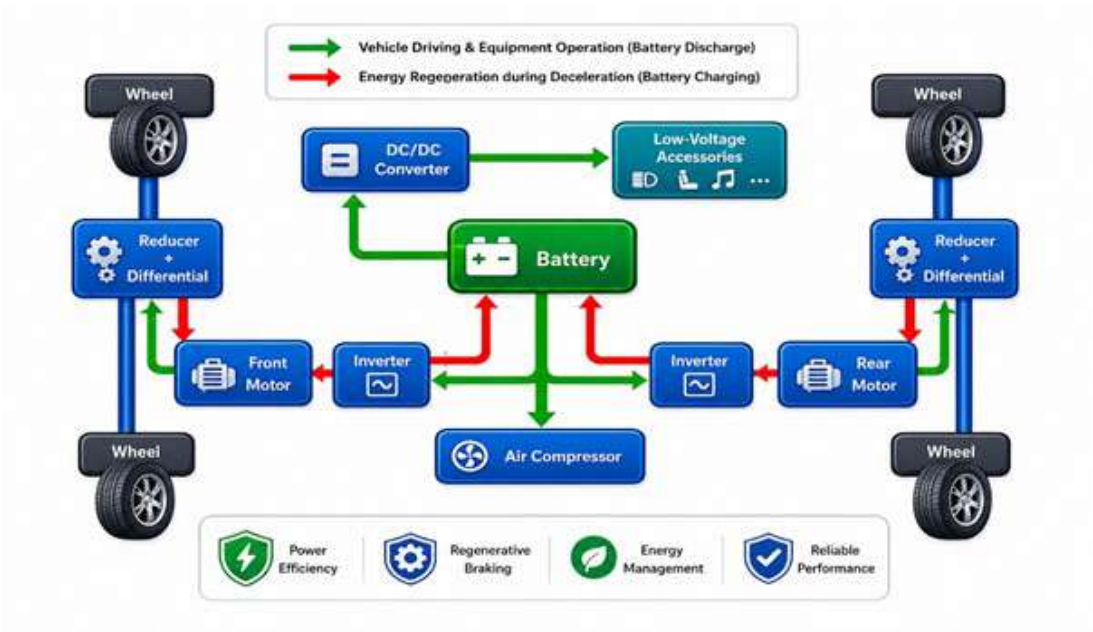


Figure 1.1: Essential components of an EV (Motor, Battery, and Power-Electronics).

1.3 Embedded Systems in EVs

In the EVs' constitution, embedded systems (ES) have become indispensable. They are widely used in various applications, including engine management, assisted braking, electronic charging systems, and advanced safety mechanisms such as the Anti-lock Braking System (ABS) and the Electronic Stability Program (ESP).

An ES is a self-contained electronic, computer-based system that fits within a hardware device. It is designed to do just one main job, often needing to respond quickly (guaranteed real-time). These systems usually have a small processor, memory, ways to connect with other devices, and parts like sensors and motors Figure 1.2. Additionally, they support driver interaction through dashboard displays, commonly referred to as Human-Machine Interfaces (HMIs), as well as navigation systems based on the Global Positioning System (GPS). Modern EVs integrate a large number of ESs, often exceeding 70 Electronic Control Units (ECUs).



Figure 1.2: Types of Embedded Systems Application in Automobile[3].

The development of EVs has amplified the integration and use of embedded electronic systems (EES). Specialized Electronic Control Units (ECUs), requiring advanced control and communications protocols, managing each main component (motor, battery, inverter, and charging system). An EV is built around a specific architecture integrating several electronic subsystems controlled by embedded software (sensors, calculators, actuators), as illustrated in Figure 1.3.

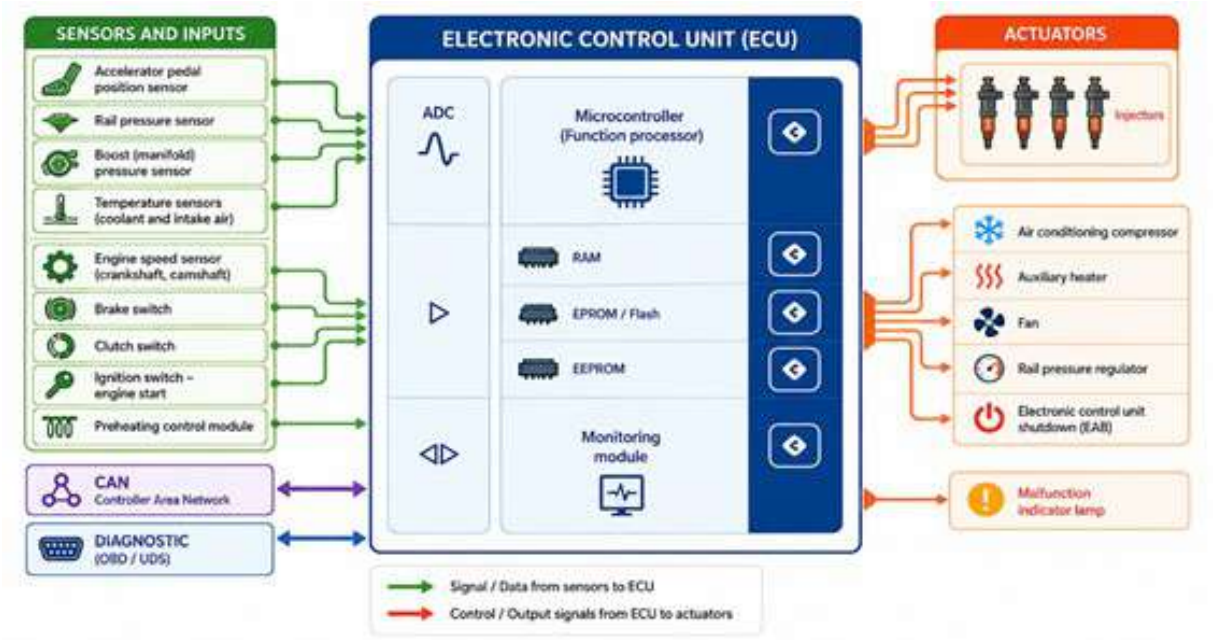


Figure 1.3: Constitution of Embedded Systems in EV .

The ESs, such as in Figure 1.3, face varying constraints based on their specific applications. In the automotive sector, real-time processing is a critical requirement, directly affecting both system performance and, more importantly, safety. Whereas a few minutes of latency on a conventional computer might merely inconvenience the user, even a few seconds of delay in an automotive ES can lead to accidents with potentially devastating consequences.

1.3.1 Types of ECUs in EV

The elementary units constituting the EV Figure 1.4 can be divided into four main categories, explained as follows.



Figure 1.4: Essential units constituting the architecture of the EV

A. **Motor Control Unit (MCU):**The electric motor of the EV allows the conversion of electrical energy into mechanical energy. Unlike an internal combustion engine, it is mechanically simpler but requires

more precise electronic control (speed, torque, frequency, and temperature). The EV's motor control unit (MCU) integrates several other key components to ensure efficient motor managing and safety, as illustrated in Figure 1.5.

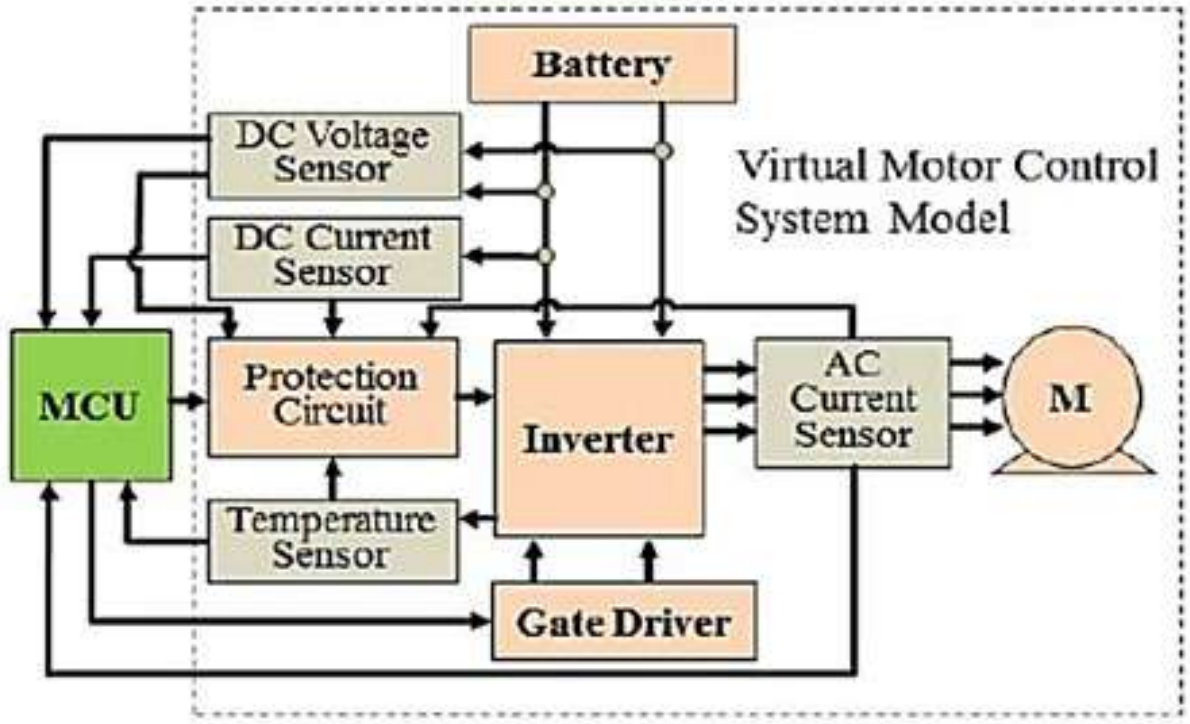


Figure 1.5: Example of an EV motor equipped with its control unit (MCU) [4]

B. **Battery Management Unit (BMU):** The energy storage unit in EVs is considered an essential element to ensure their energy efficiency and driving range. This storage is typically provided by a Battery Management Unit (BMU), which consists of a set of battery cells monitored and balanced by embedded control electronics (Battery Management System (BMS)). The BMU stores electrical energy and delivers it to the electric motor via power electronic converters (e.g., DC-DC converter and inverter), as shown schematically in Figure 1.6.

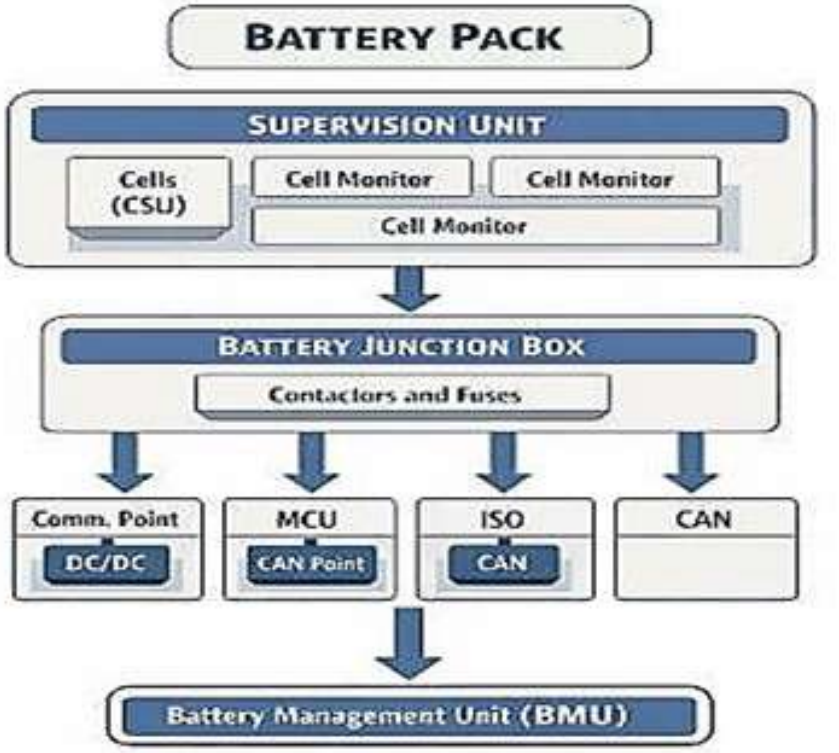


Figure 1.6: Block diagram of a BMU-based architecture in EVs.

- **Surveillance BMS**

The battery management system (Battery Management System - BMS) allows browsing more device parameters such as temperature, voltage, current, State Of Charge (SOC), and the State Of Health (SOH). An example of this BMS is represented in Figure 1.7.

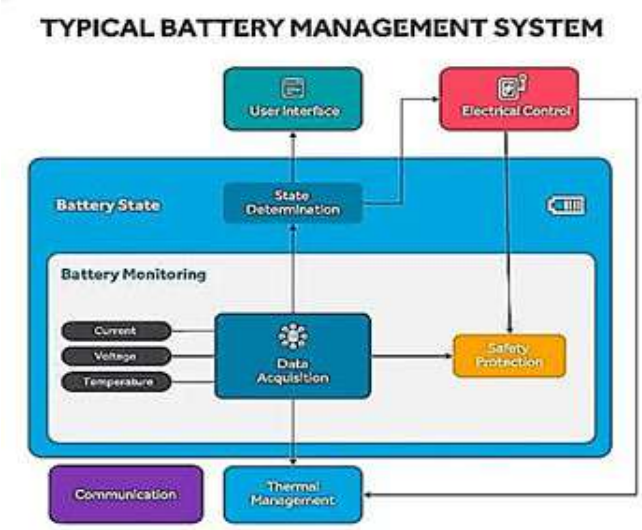


Figure 1.7: Example of a BMS [5].

C. **Power Distribution Unit (PDU):** This electronic part, known as Power-Distribution-Unit (PDU), ensures conversion and energy management enters the battery, the motor, and other parts Figure 1.8. The inverter (DC/AC), DC/DC converters, and embedded on-board charger (On-Board-Charger (OBC)).

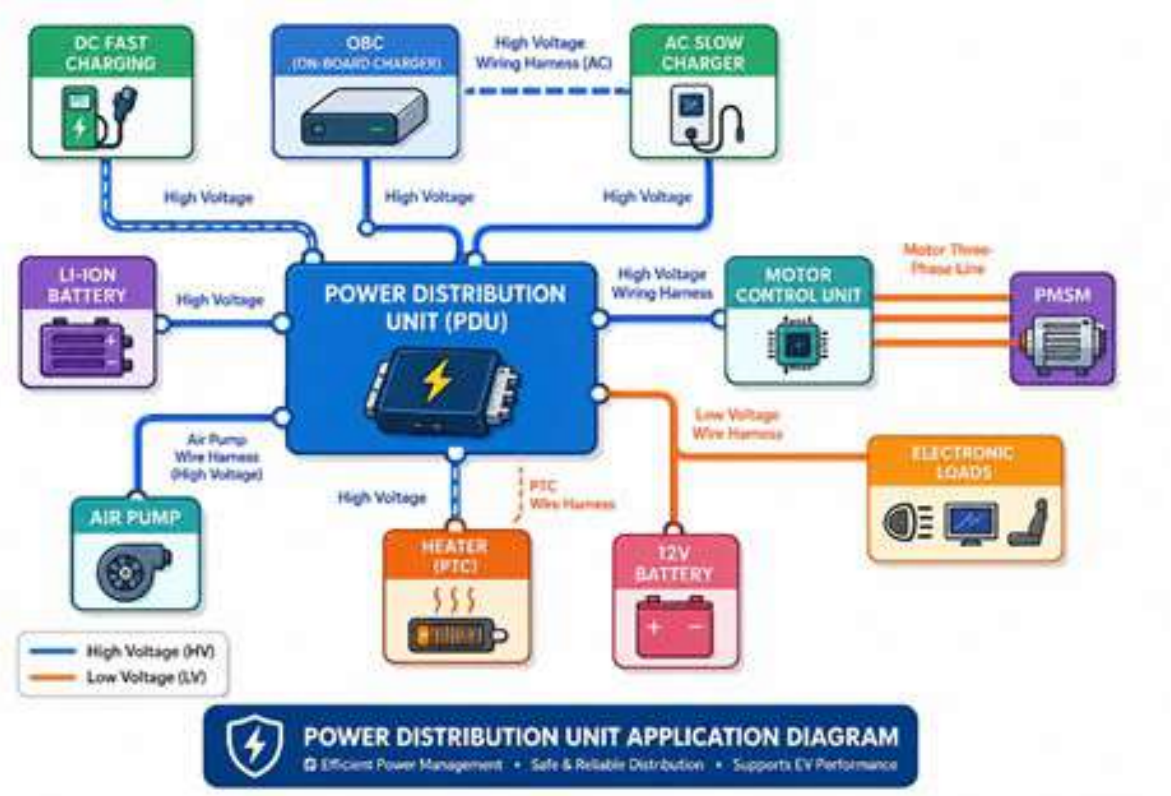


Figure 1.8: Synoptic diagram of PDU [6].

D. **Electronic Control Units (ECU):** Each subsystem of the EV is controlled by one or more ECUs. These ECUs communicate with each other via an internal on-board network, as schematically illustrated in Figure 1.9.

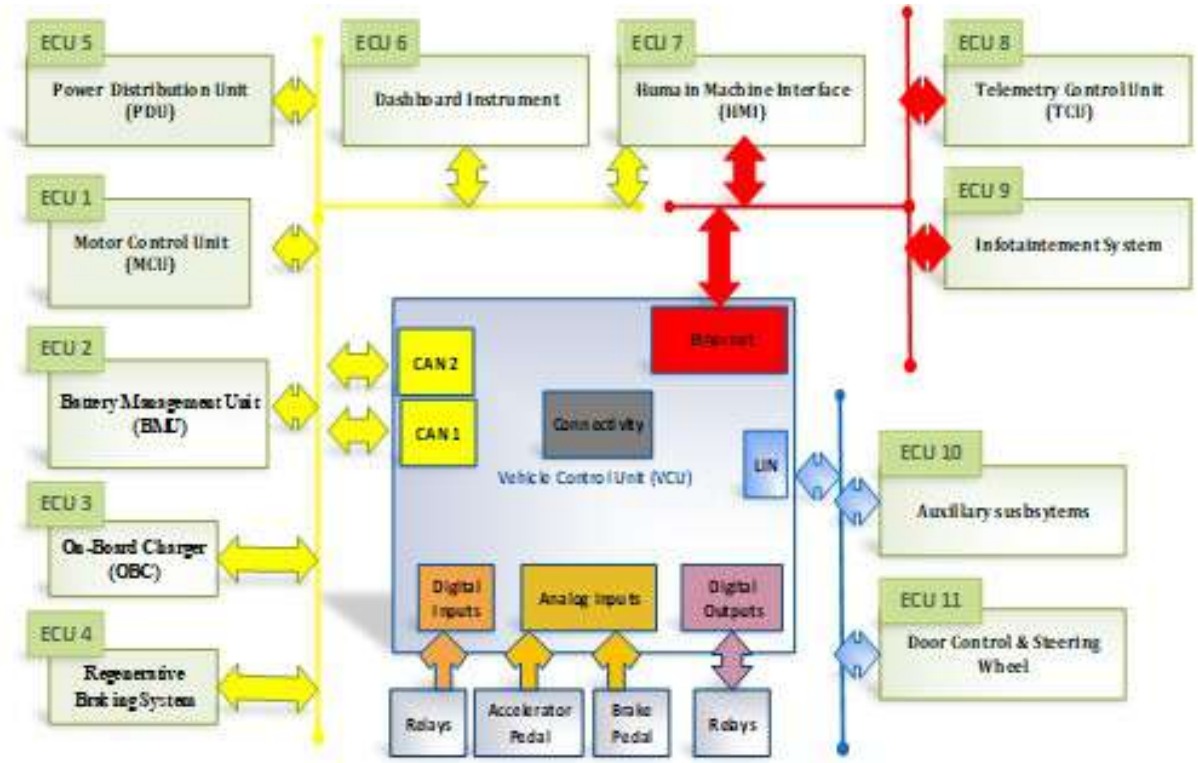


Figure 1.9: Synoptic diagram of ECUs with communication protocols.

The networking in EV of its ECUs is ensured by different communication protocols such as Controller Area Network (CAN), Local Interconnect Network (LIN), Ethernet as illustrated in Figure 1.9.

1.3.2 Embedded On-Board Diagnosis (OBD) tool

On-Board Diagnosis (OBD) tools aims to monitor in real-time the health states of various EV components, as illustrated in Figure 1.10. It enables the detection, localization, and interpretation of potential or actual faults. This capability is essential for maintaining vehicle performance, preventing critical failures, and facilitating predictive maintenance operations

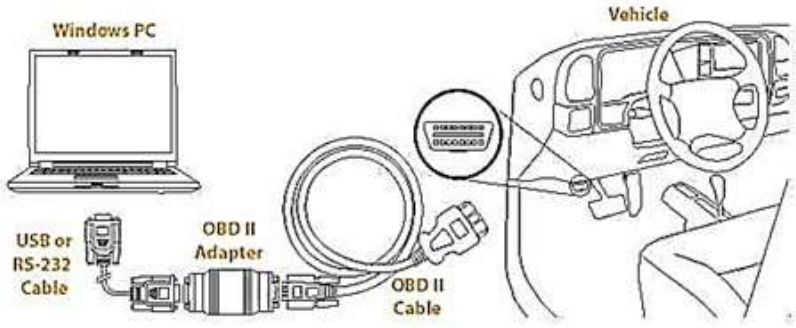


Figure 1.10: Representation of the embedded on-board diagnosis in EV

Embedded On-board Diagnosis (OBD) is fundamental to contemporary integrated electronic systems in EVs. OBD encompasses techniques and technologies that facilitate real-time monitoring, analysis, and fault diagnosis of various vehicle components. This functionality is essential in EVs, as it ensures the optimal operation of critical subsystems, including the electric motor, battery, power electronics, and communication networks. Failures within these subsystems can substantially impact vehicle performance, safety, and reliability. Diagnosis is applied across multiple disciplines, each relying on logic, analysis, and experience to establish cause-and-effect relationships. In automotive engineering, diagnosis serves to identify the sources of malfunctions and recommend appropriate solutions. Automotive fault diagnosis Figure 1.11 may be conducted using either traditional or modern approaches. Traditional methods depend on a technician's observation of visible symptoms [7]. These approaches are typically slower, less accurate, and demand significant technical expertise. Conversely, modern methods employ integrated diagnostics, such as electronic diagnostic tools, which utilize communication protocols between electronic control units to automatically detect faults. Information is transmitted via the Controller Area Network (CAN) protocol among the vehicle's electronic control units and is accessible through the OBD interface, enabling interaction with integrated diagnostics. These advanced methods provide greater sophistication and enhanced security [8].

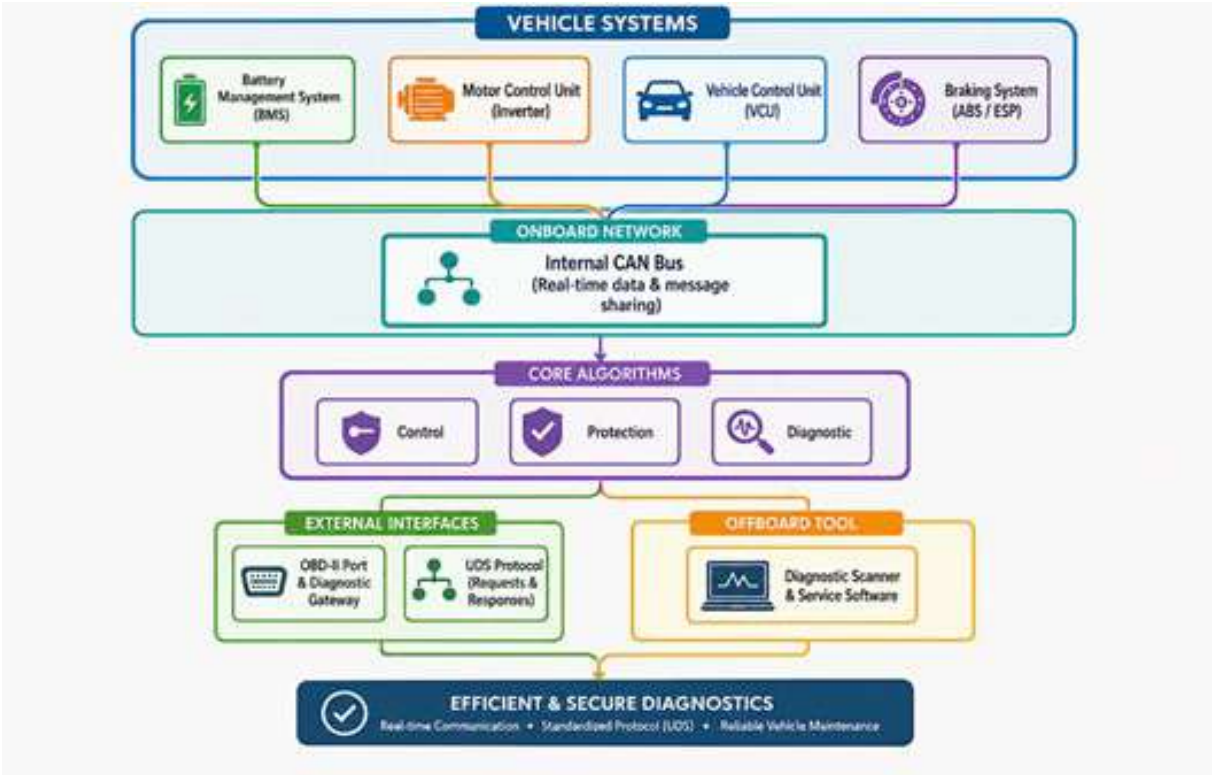


Figure 1.11: Diagnosis communication architecture of an EV using CAN-based bus and UDS protocol.

1. **Diagnosis in Communication Protocols**

   EVs rely on developed communication protocols such as CAN and LIN to coordinate its various subsystems, where communication faults can disrupt vehicle operation. Therefore, diagnosis in these protocols is categorized into three groups, according to Figure 1.12.

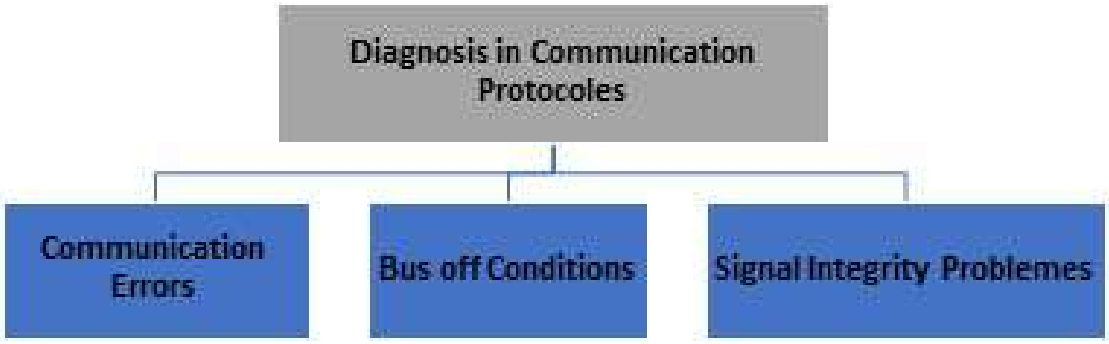


Figure 1.12: Diagnosis methods in communication protocols.

Below a presentation about the faults EV protocols for identifying them, with their diagnosis approaches.

- **Communication errors**

  Detection: Monitored using error-checking protocols and message integrity checks.

  Diagnosis: These errors may indicate network issues or faulty components.

- **Bus off conditions**

  Detection: Monitoring of the CAN network status

  Diagnosis: A"bus off" condition signals serious errors preventing normal communication.

- **Signal integrity issues**

  Detection: Analysis through signal monitoring and diagnostic tools.

  Diagnosis: Integrity issues may indicate wiring or component faults Thus, there exist advanced techniques for diagnosing protocols, listed below:

A. **Network monitoring tools:** Continuously monitor network traffic and diagnosis. This enables real-time visibility into the communication network status.

B. **Fault-tolerant communication protocoles:** Integrate redundancy and error correction into protocols. This enhances communication reliability and robustness.

1.4 Conclusion

This chapter has provided an overview of ES and networking with bus for EVs. It presented the main subsystems (MCU, BMU, PDU, ECUS with communication protocols) and classified communications protocols, including CAN, LIN, Ethernet, Bluetooth, and MOST. Despite significant progress, limitations such as lack of standardization, system complexity, data quality issues, and dependency on specialized tools remain. These challenges drive the need for smarter, faster, and more flexible diagnosis solutions.

CHAPTER

2

NETWORKING IN EV AND COMMUNICATIONS PROTOCOLS

2.1 Introduction

Modern electric vehicles depend on onboard electronic systems (ES) for efficient operation and improved safety. Specialized communication protocols (CAN, LIN, MOST, TTP, and FlexRay) enable reliable data exchange between these systems In what follows, a presentation with details about each protocol, particularly CAN and LIN, is provided.

2.2 CAN

The Controller-Area-Network (CAN) is considered as one of the most critical communication protocols in EVs, as it provides a reliable and efficient network for exchanging information among various electronic units, as revealed in Figure 2.1. Developed by Bosch in the 1980s, it was designed to reduce wiring complexity and facilitate communication in-vehicle. Given its importance in vehicle networking, this chapter examines the CAN protocol in detail, including its operating principles, architecture, and applications.

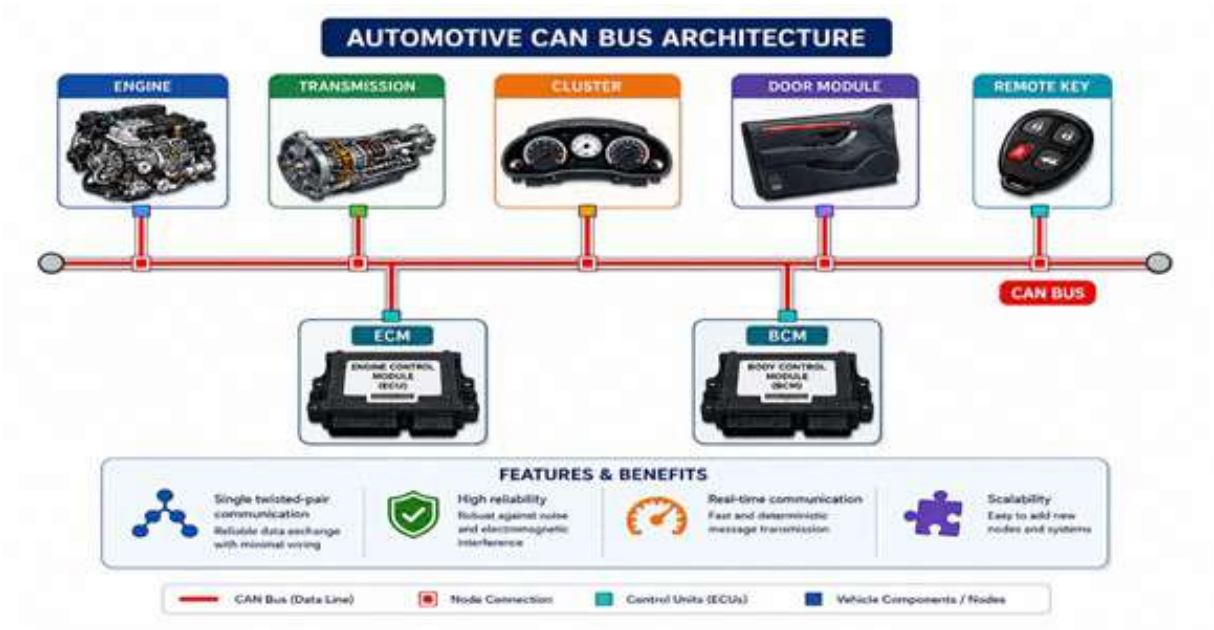


Figure 2.1: Automotive CAN BUS networking of different ECUs [9].

2.2.1 History

Bosch developed the CAN protocol in 1985, as a high-integrity serial communication-based bus designed to connect intelligent devices and replace the traditional point-

to-point wiring system used in automobiles. As vehicle electronics became more widespread, the automotive industry adopted CAN to eliminate heavy, expensive, and complex wire harnesses. In 1993, CAN was standardized internationally as ISO 11898. Later, several higher-level protocols, such as CAN open and Device Net, were standardized on top of CAN and became widely used in industrial communication systems. In 1996, the On-Board-Diagnosis (OBD-II) standard, which incorporates CAN, became mandatory for all cars and light trucks sold in the United States. Today, CAN is used in many fields beyond the automotive sector, including surface transportation, industrial automation, maritime systems, and avionics. It is also integrated into many microcontrollers, making it a common and practical communication solution in (ES) [10].

2.2.2 Principal

The CAN protocol serves as a communication protocol in EV. Specifically designed for microcontrollers and devices, it facilitates communication within a networked environment without requiring a host computer. The CAN protocol is based on a message-based communication model where each device on the network can send and receive messages. It uses a two-wire bus, consisting of a CAN High (CAN-H) and a CAN Low (CAN-L) wire, to transmit and receive data between devices. To ensure signal integrity and prevent reflections on the bus, a 120-ohm termination resistor is placed between the CAN-H and CAN-L lines at each physical end of the bus (i.e., at least two resistors). This resistor matches the characteristic impedance of the twisted pair wires, preserving the stability of the physical layer and enabling high-speed, reliable data transmission. The CAN protocol provides a highly reliable and robust communication system, capable of transmitting data at high speeds over long distances. CAN is widely used in diverse applications, such as engine control, body control, powertrain control, and more , as shown in Figure 2.2. Where each application is considered as a CAN node.

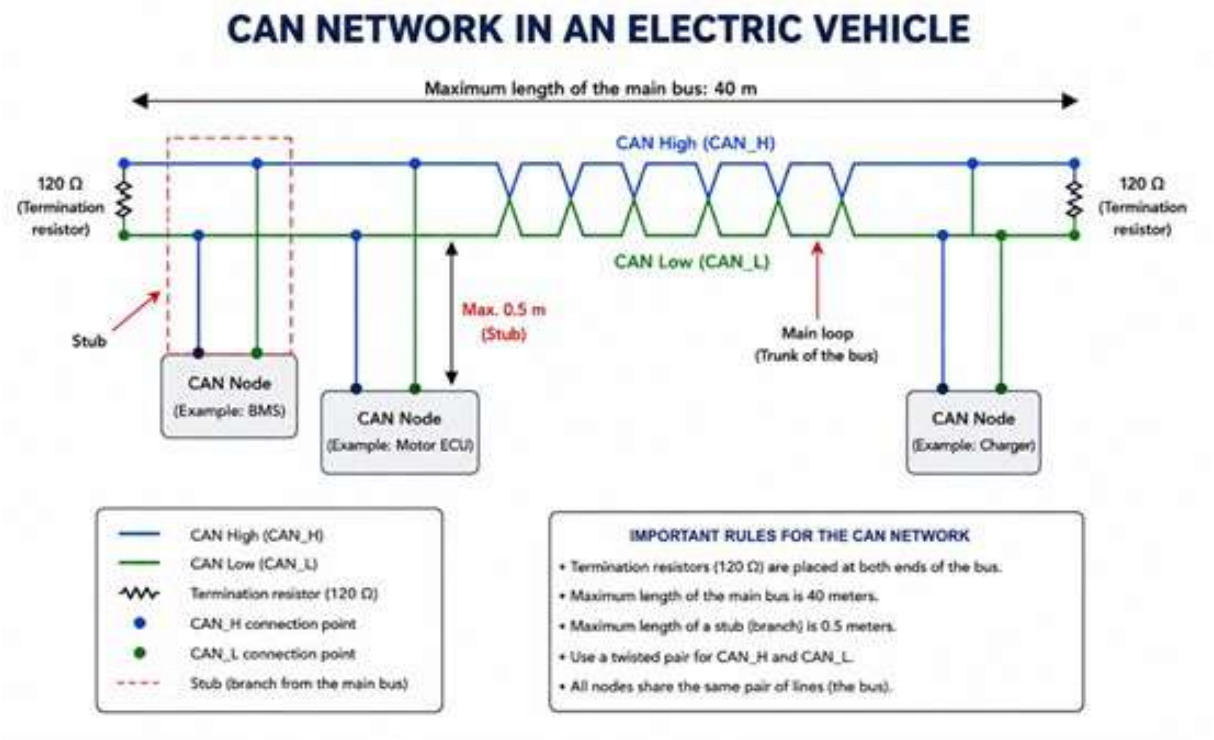


Figure 2.2: Typical CAN Network [11].

The communication speed of CAN protocol versions are described as follow:

- 125 kbps (long-distance network, e.g., heavy goods vehicles, industry). (Class A)
- 500 kbps (conventional vehicles, embedded systems). (Class B)
- 1 Mbps (short distances, critical applications). (Class C)
- 8 Mbps (CAN-FD (Flexible Data-Rate)). (Class D)

2.2.3 Constitution of a CAN Node

As shown in Figure 2.3, a CAN node is constituted basically from 3 elements, such as the microcontroller, CAN-Controller, and CAN-Transceiver.

- **Microcontroller (Host):** It decides what the received messages mean and what messages it wants to transmit.
- **CAN-Controller:** They are often an integral part of the microcontroller that handles framing, CRC, etc, like the Data Frame, Remote Frame, Error Frame, Overload Frame, and Inter-Frame Space generation.
- **CAN-Transceiver:** The CAN transceiver transforms data between the CAN controller and bus levels. It converts data from the CAN controller to CAN

bus levels and vice versa. In essence, the CAN transceiver's role is to convert Microcontroller TTL voltage levels (High/Low) into CAN Bus protocol voltage levels (recessive/dominant) and vice versa.

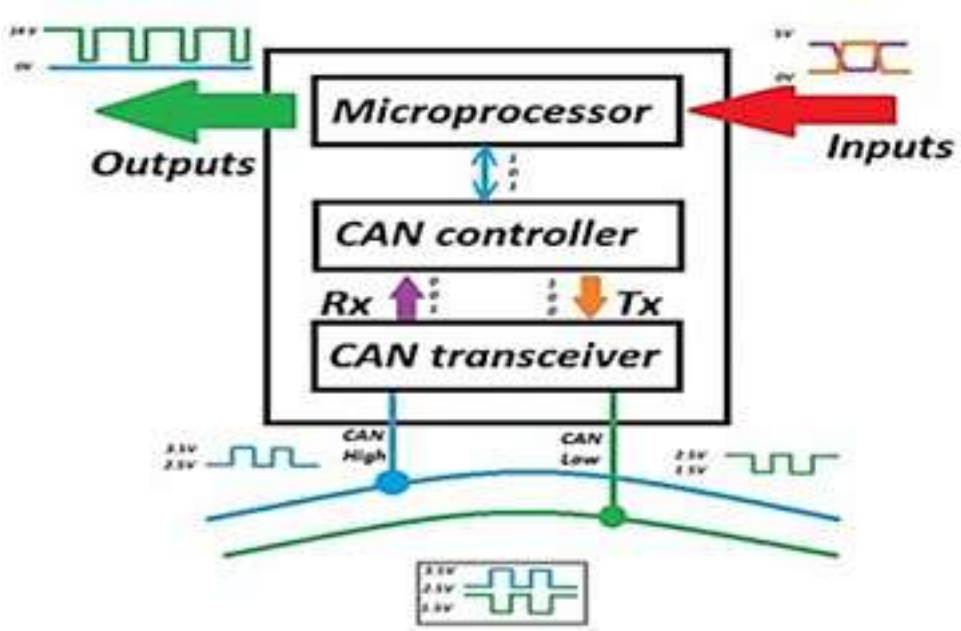


Figure 2.3: Basic constitution elements of a CAN node [12].

2.2.4 CAN Message Structure

The CAN protocol relies on a well-defined and structured communication format known as data frames. Each message transmitted over the network is organized into multiple fields, with every field serving a specific role in ensuring reliable and efficient data exchange between nodes. This structured approach enhances data integrity, synchronization, and overall system performance, as illustrated in the following Figure 2.4.

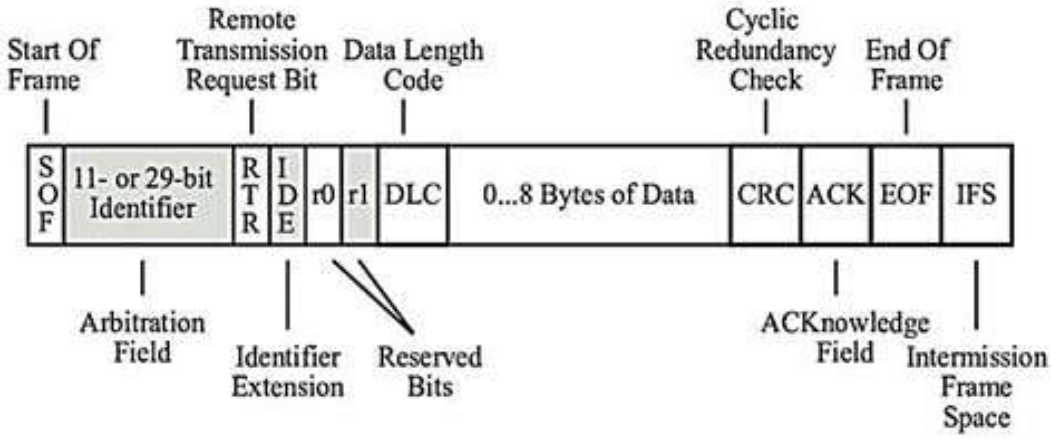


Figure 2.4: The CAN standard frame format [13].

The CAN Frame is an entire CAN transmission: arbitration ID, data bytes, acknowledge bit, and so on. Frames are also referred to as messages.

- **SOF (start-of-frame) bit:** indicates the beginning of a message with a dominant (logic 0) bit.
- **Arbitration ID:** identifies the message and indicates the message's priority. Frames come in two formats: standard, which uses an 11-bit arbitration ID, and extended, which uses a 29-bit arbitration ID.
- **IDE (identifier extension) bit:** allows differentiation between standard and extended frames.
- **RTR (remote transmission request) bit:** serves to differentiate a remote frame from a data frame. a dominant (logic 0) RTR bit indicates a data frame. a recessive (logic 1) RTR bit indicates a remote frame.
- **DLC (data length code):** indicates the number of bytes the data field contains.
- **Data Field:** contains 0 to 8 bytes of data.
- **CRC (cyclic redundancy check):** contains a 15-bit cyclic redundancy check code and a recessive delimiter bit. The CRC field is used for error detection.
- **ACK (ACKnowledgement) slot:** any CAN controller that correctly receives the message sends an ACK bit at the end of the message. The transmitting node checks for the presence of the ACK bit on the bus and reattempts transmission if no acknowledge is detected. National Instruments Series 2 CAN interfaces have the capability of listen-only mode. Herein, the transmission of an ACK bit by

the monitoring hardware is suppressed to prevent it from affecting the behavior of the bus.

- **CAN Signal:** an individual piece of data contained within the CAN frame data field. You also can refer to CAN signals as channels. Because the data field can contain up to 8 bytes of data, a single CAN frame can contain 0 to 64 individual signals (for 64 channels, they would all be binary).

2.2.5 CAN's features

The CAN protocol is a reliable communication system that connects multiple devices within a network without a central controller. It broadcasts messages to all devices, allowing each to determine if a message is relevant based on its unique ID. This approach offers flexibility and simplifies device expansion. CAN uses a priority system to ensure critical messages are transmitted first and supports multi-master functionality, enabling any device to initiate communication. Error detection methods such as (CRC) and automatic retransmission maintain data integrity, while faulty devices can be isolated to preserve network stability. The protocol operates over a single channel with two noise-resistant signals (CAN H and CAN L), includes (ACK) for message confirmation, and supports a power-saving sleep mode. These features make CAN well-suited for applications in automotive and industrial automation.

2.2.6 CAN's Advantages

A. **Low Cost, Lightweight Advantages:** CAN provides an inexpensive, durable network that allows multiple CAN devices to communicate. An advantage is that (ECUs) can use a single CAN interface instead of analog and digital inputs for every device. This reduces overall cost and weight in automobiles, as shown in Figure 2.5.

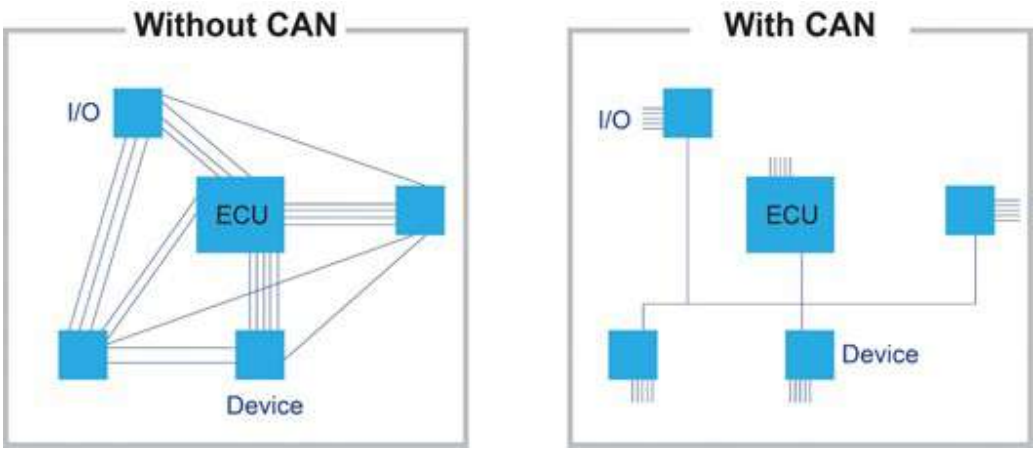


Figure 2.5: Reduction of Wiring in EV through CAN Networks [14] .

B. **Broadcast Communication:** Each device on the network has a CAN controller chip and is therefore intelligent. All devices see all transmitted messages. Each device decides if a message is relevant or should be filtered. This structure allows modifications to CAN networks with minimal impact. Additional non-transmitting nodes can be added without changing the network.

C. **Priority:** Every message has a priority, so if two nodes try to send messages simultaneously, the one with the higher priority is transmitted, and the one with the lower priority is postponed. This arbitration is non-destructive and results in non-interrupted transmission of the highest priority message. This also allows networks to meet deterministic timing constraints.

D. **Error Capabilities:** The CAN specification includes a CRC to perform error checking on each frame's contents. All nodes disregard frames with errors, and an error frame can be transmitted to signal the error to the network. The controller differentiates global and local errors, and if too many errors are detected, individual nodes can stop transmitting errors or disconnect themselves from the network completely[15].

2.2.7 Applications

The CAN protocol finds its way into a wide range of industries, powering different uses and features in each one, as shown in the examples below.

- Vehicles: Engine management, Battery, ABS, airbags, transmission, ECU communication, OBD-II diagnostics.

- Industrial: Machine control, industrial programmable logic controllers (PLC), robots, smart sensors.
- Medical: Diagnostic devices, radiology, MRI, patient monitoring.
- Heavy machinery, Such as tractors and excavators, uses CAN to connect various systems, including the engine, hydraulic system, and instruments. The J1939 protocol, a standard for heavy vehicles, is often used [16].

2.3 LIN

The LIN is a CAN and SAE-J1850 compliant application protocol that is not time-critical and does not need extreme fault tolerance since the LIN is not as reliable as the CAN protocol. The objective of the LIN protocol was to use where time and fault-tolerant is not a priority and to implement at a lower cost than the CAN protocol. [17]

2.3.1 History

The LIN stands for Local Interconnect Network. Whenever the Automotive OEMs started increasing the use of a huge amount of features in the vehicle with more sensors and other smart intelligent devices in the vehicle, the price increased, as it was more costly for the customers. So to make a vehicle with low cost, the OEM's as BMW, Volkswagen Group, Audi Group, Volvo Cars, and Mercedes-Benz, are 5 automakers that developed the LIN protocol in 1990 by Volcano Automotive Group and Motorola. The first LIN version, 1.0, was not stable enough for use in vehicles. The LIN 1.3 version, published in November 2002, began to be used in vehicles. In September 2003, the LIN 3.0 version was introduced to expand features and add diagnostics. One advantage of LIN is its ability to serve as a DC power line for automotive vehicles, and it is being standardized as ISO/AWI 17987-8.

2.3.2 Protocol principle

The LIN is a broadcasting communication protocol that uses a single master and up to 16 slave nodes. It is SCI/UART-based, serial, byte-oriented, and time-triggered. LIN is designed to support automotive networks together with Controller Area Network (CAN). It enables cost-effective communication with sensors and actuators when all the features of CAN are not needed. The main features of this protocol are low cost

and low speed compared to CAN. It is used for short-distance networks. Usually, the LIN protocol is used in Automotive vehicles to support the CAN network for cost-effective connection between the sensors and actuators with an ECU, which is often a gateway with a CAN bus. In LIN protocol, there is no collision detection capability; therefore, in LIN, all the messages are initiated by the Master node, and at most one slave will reply with the transmitted message identifier. The master is typically a moderately powerful microcontroller, whereas the slaves can be less powerful, cheaper micro-controllers or dedicated ASICs, etc. The LIN is a single-wire 12V bus networking communication protocol based on the ISO-9141 NRZ-based standard protocol. The main feature of the LIN network is the synchronization mechanism, which allows the clock recovery by the slave nodes without a quartz or ceramic resonator. Only the Master node consists of the clock or the oscillator. Like the CAN protocol, the LIN nodes can also be added to the LIN network without any hardware or software changes in other slave nodes[18] , as shown in Figure 2.6.

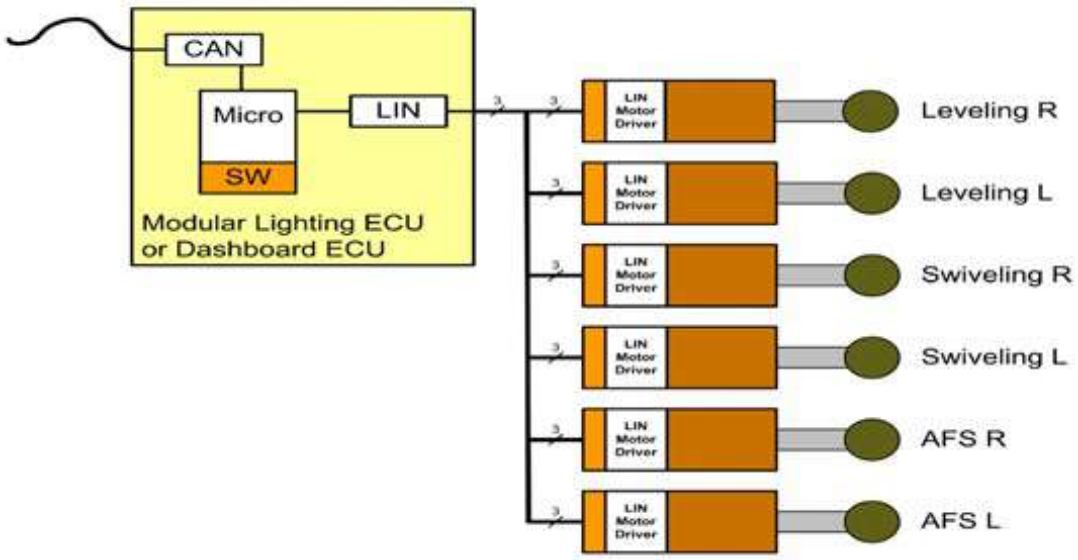


Figure 2.6: LIN Network with Master Slave Communication [19]

2.3.3 Communication System (Master–Slave Architecture)

A LIN bus has a single LIN master and up to 15 LIN slave nodes (up to 16 nodes in total), forming a LIN cluster. Every LIN node has a 'slave task' (for ignoring, sending, and receiving data), while the LIN master also has a 'master task' responsible for scheduling all communication on the bus, ensuring deterministic and collision-free operation. The LIN master often acts as a gateway between a LIN cluster and a vehicle CAN bus. As in CAN, ECUs, each including a microcontroller, serial interface

(SCI), and LIN transceiver, handle all LIN node roles [17]. As shown in Figure2.7.

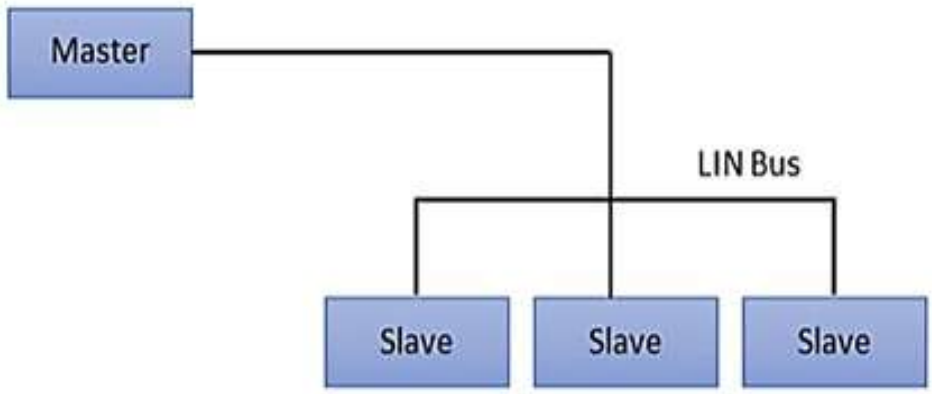


Figure 2.7: Master–Slave Architecture of LIN bus [20].

2.3.4 Physical Layer

The LIN-bus transceiver, a modified version of the LIN transceiver used in the ISO-9141 standard, facilitates bidirectional communication in automotive applications by connecting to a node transceiver and utilizing a termination resistor and diode to Vbat for reliable data transfer between Master and Slave. It converts microcontroller bit logic into higher voltage levels for transmission along the bus and translates incoming signals back to microcontroller-compatible levels. The TxD pin connects to the microcontroller's LIN controller, enabling message broadcasting on the LIN bus, while the RxD pin monitors the bus for incoming messages. Typical operating voltages for the LIN bus and transceivers range from 9 V to 18 V, with some applications supporting up to 30 V, aligning with standard vehicle battery systems (usually 12 V or occasionally 24 V) as shown in Figure 2.8.

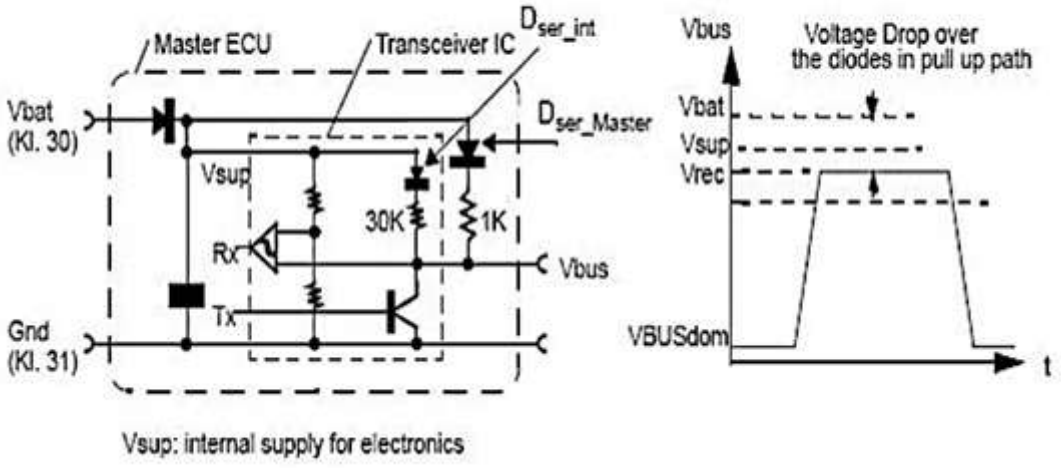


Figure 2.8: Lin Physical Layer Design [21].

2.3.5 LIN BUS Threshold Voltage Level

In a LIN network, data transmission occurs via voltage-level variations on the LIN bus. The voltage thresholds for dominant (low) and recessive (high) states differ between the transmitter (TX) and receiver (Rx) to account for potential variations in supply voltage and signal integrity, as shown in Figure 2.9

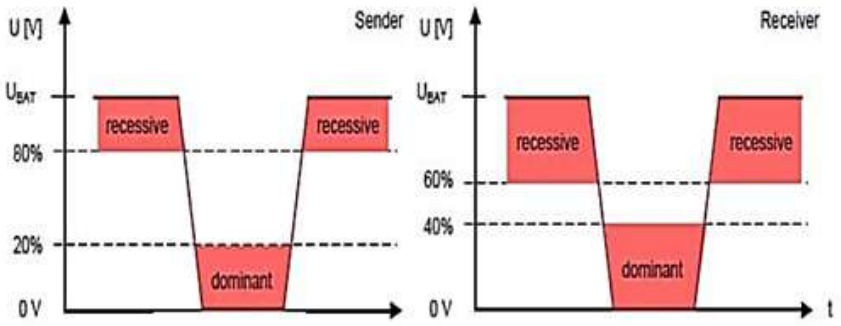


Figure 2.9: LIN Bus-based Threshold Level .

A. **Dominant State (Logical LOW – Active Signal)**

• **Transceiver (TX):** The sender drives the bus voltage down to 20 percent of the battery voltage (VBAT).

• **Receiver (RX):** A dominant bit is detected when the bus voltage drops to 40 percent of the battery voltage.

B. **Recessive State (Logical HIGH – Idle State)**

• **Transceiver (TX):** The sender raises the bus voltage to 80 percent of the battery voltage.

• **Receiver (RX):** A recessive bit is interpreted when the voltage reaches 60 percent of the battery voltage.

2.3.6 Message frame format

A message frame in the LIN protocol consists of two main parts: a header and a response. The communication process is initiated by the LIN master, which transmits the header onto the bus to address a specific slave node. Upon receiving this header, the corresponding slave responds by sending up to 8 bytes of data. This structured exchange ensures organized and deterministic communication within the network, as illustrated in Figure 2.10.

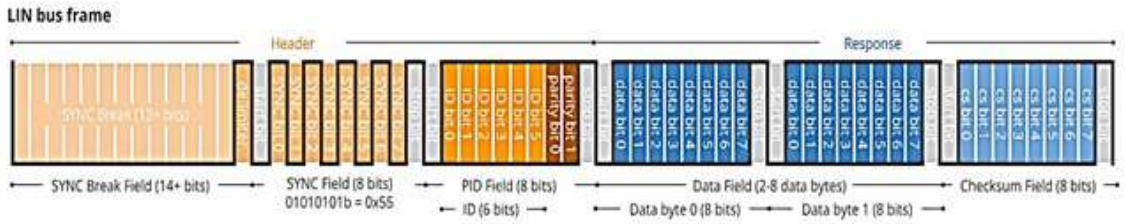


Figure 2.10: LIN bus frame.

A. **Sync Break Field (SBF):** The Sync Break Field (SBF) is a minimum 13 + 1 bit long, as shown in Figure 2.11, with the last bit reflecting a delimiter. The field acts as a 'start of frame' notice to all LIN nodes on the bus.

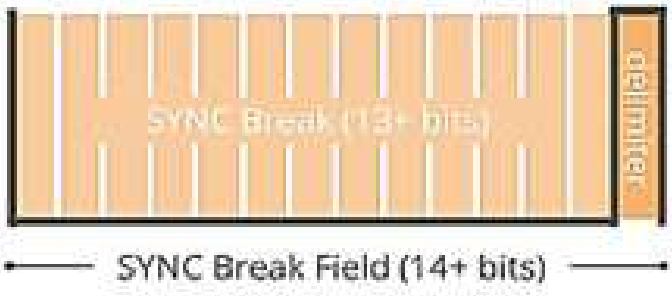


Figure 2.11: LIN bus frame Sync Break Field (SBF).

B. **SYNC Field:** The 8-bit SYNC field has a fixed data value of 0x55 (binary 01010101) as shown in Figure 2.12. This allows the LIN slaves to determine the time between rising/falling edges and thus the baud rate used by the LIN master, allowing them to stay in sync.

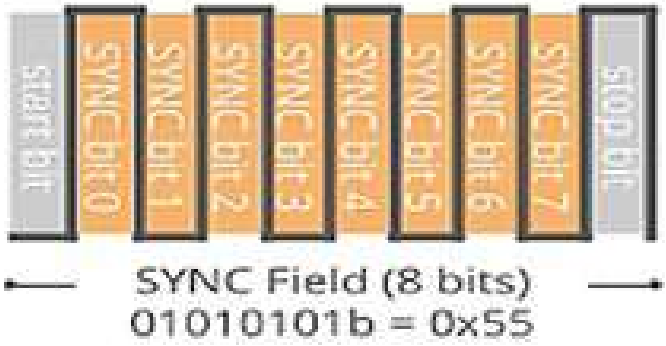


Figure 2.12: LIN bus frame Protected ID Field (SYNC).

C. **Protected ID Field (PID):** The Protected ID Field (PID) contains the frame ID (6 bits, allowing for 64 unique IDs), followed by 2 parity bits as shown in Figure 2.13.

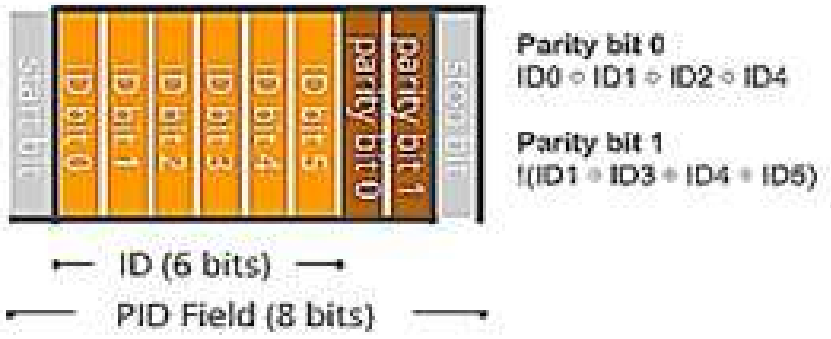


Figure 2.13: LIN bus frame: Protected ID Field (PID)

This acts as an identifier for each LIN message sent. LIN slaves verify the validity of the ID based on the parity bits and react as follows:

- Ignore the subsequent data transmission
- Listen to the data transmitted from another node
- Publish data in response to the header

D. **Data Field:** When a LIN slave is polled by the master, it can respond by transmitting 2 to 8 bytes of data. The data bytes contain the actual information being

communicated in the form of LIN signals using Intel byte order as shown in Figure 2.14. The data length can be customized, but it is typically linked to the ID range:

- ID 0-31: 2 data bytes
- ID 32-47: 4 data bytes
- ID 48-63: 8 data bytes

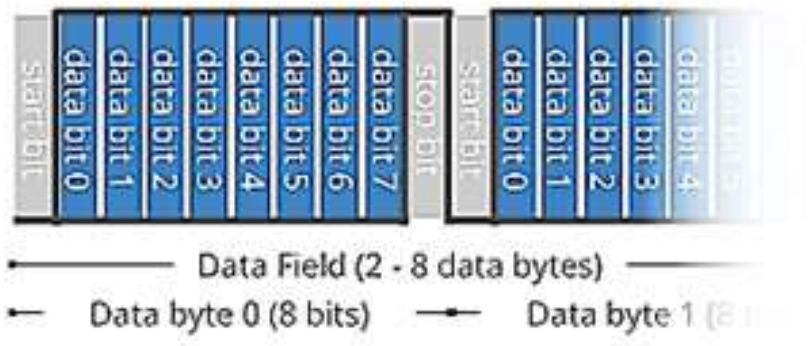


Figure 2.14: LIN bus frame Data Field

E. **Checksum Field:** As in CAN, a checksum field ensures the validity of the LIN frame. The classic 8-bit checksum is based on summing the data bytes only (LIN 1.x), while the enhanced checksum algorithm also includes the identifier field (LIN 2.x). IDs 60-61 always use the classic checksum as shown in Figure 2.15.

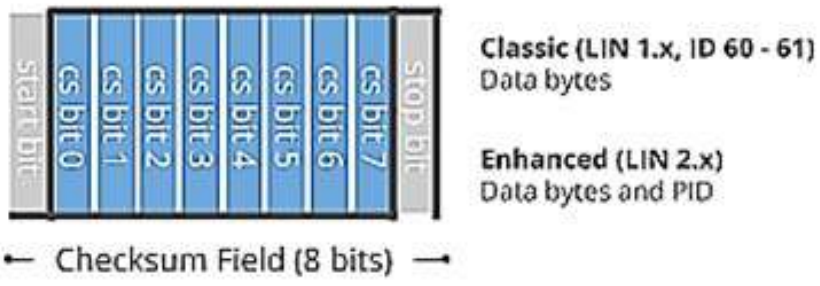


Figure 2.15: LIN bus frame Checksum Field.

2.3.7 Advantages and Disadvantages

The advantages and disadvantages of the LIN protocol are as follows:

A. **Advantages**

- Low cost: LIN is cheaper than more advanced in-vehicle networks because it uses a simple architecture and minimal hardware.
- Simple implementation: Its master–slave structure makes it easy to design, integrate, and maintain.
- Reduced wiring complexity: LIN supports communication over a single wire, which helps reduce cabling inside the vehicle.
- Deterministic communication: Since the master controls message scheduling, LIN offers predictable timing.
- Suitable for non-critical functions: It is widely used for comfort and body electronics such as windows, mirrors, and seat control.

B. **Disadvantages**

- Low data rate: LIN is much slower than CAN, so it is not suitable for high-speed communication.
- Not suitable for safety-critical systems: It is mainly used for simple applications, not for functions requiring high reliability or fast response.
- Master dependency: If the master node fails, communication on the network can be affected.
- Limited scalability: LIN is designed for small, simple networks, not large or complex ones.
- Less flexibility: It does not provide the same level of robustness and performance as CAN.

2.3.8 Applications

The LIN protocol bus is widely used for networking mechatronic systems in motor vehicles, particularly in applications where the high data rate and flexibility of the CAN bus are not required. Due to its low cost and simplicity, LIN is well-suited for non-critical control functions within the vehicle. Typical applications of LIN include door modules (such as door locking, power windows, and mirror adjustment), control of the power sunroof, windshield wiper motor control, rain and light sensors,

air-conditioning systems (including signal transmission and blower control), headlight electronics, seat adjustment motors, theft deterrence systems, and garage door openers, as illustrated in Figure 2.16.



Figure 2.16: In-Vehicle LIN Protocol Network..

2.3.9 Comparison between CAN and LIN networks

Table 2.1 highlights the key differences between CAN bus and LIN bus.

After exploring the characteristics and applications of CAN protocol and LIN protocol, it becomes clear that in-vehicle communication systems require different protocols to meet varying performance and functionality needs. While CAN and LIN are well-suited for control and low-cost applications, more advanced or specialized systems demand higher data rates, deterministic timing, or efficient multimedia data transmission. For this reason, additional communication protocols such as FlexRay protocol and MOST protocol have been developed. These protocols are designed to address specific requirements, such as real-time safety-critical communication in the case of FlexRay, and high-speed multimedia data transfer for in-vehicle infotainment systems in the case of MOST.

Parameters	**CAN**	**LIN**
Full form	Controller Area Network	Local Interconnect Network
Communication type	Broadcast type	Master-slave
Communication network	Multi master, peer to peer	Single master and multiple slaves
No. of lines required	2 wires	1 wire
Maximum communication speed	1 Mbps	20 kbps
Triggering technique	Event triggered	Time triggered
Message length	Longer messages, extensive data payload	Shorter messages, limited data payload
Message transmission latency	Depends on message priority	Guaranteed latency
Bus conflicts	Uses arbitration process to resolve	No bus conflict
Implementation cost	Higher	Comparatively lower
Implementation complexity	Higher	Comparatively lower
Error handling	Advanced error detection and correction	Basic error handling
Electromagnetic interference (EMI)	Resilient to EMI	Less immune to EMI compared to CAN
Applications	Safety critical	Non-safety critical

Table 2.1: Difference between CAN and LIN bus protocols[22] .

2.4 FlexRay Protocol

The Flexible Ray (FlexRay) is a deterministic, fault-tolerant, and high-speed automotive networking communication system protocol developed by the FlexRay Consortium to control the automotive ECUs. The deterministic terminology is used here for Time-division multiple access technologies. Whereas the CAN protocol is used Frequency division multiple access technology. It is standardized in ISO174518-1 to ISO17458-5 for use in different industries. It is faster and more reliable than the CAN protocol, but due to its expense, most OEMs do not prefer to use it. Still, in some modules, some of the OEM's are using for high-speed communication. FlexRay has a maximum of 10 Mbps speed, whereas CAN has a maximum of 1mbps only. The

FlexRay is used in Power Train Module, Safety suspension, Adaptive Cruise Control, etc. This technical article will provide a full flesh of Flexray protocol tutorial from basic to a deep understanding.

2.4.1 FlexRay Topology and Layout

One of the things that distinguishes FlexRay, CAN, and LIN from more traditional networks such as Ethernet is their topology, or network layout. FlexRay supports simple multi-drop passive connections as well as active star connections for more complex networks. Depending on a vehicle's layout and level of FlexRay usage, selecting the right topology helps designers optimize cost, performance, and reliability for a given design [23].

A. **Multi-drop Bus:**FlexRay is commonly used in a simple multi-drop bus topology that features a single network cable run that connects multiple ECUs together, as shown in Figure 2.17. This is the same topology used by CAN and LIN and is familiar to OEMs, making it a popular topology in first-generation FlexRay vehicles. Each ECU can "branch" up to a small distance from the core "trunk" of the bus. The ends of the network have termination resistors installed that eliminate problems with signal reflections. Because FlexRay operates at high frequencies, up to 10 Mbit/s compared to CAN's 1 Mbit, FlexRay designers must take care to correctly terminate and lay out networks to avoid signal integrity problems. The multi-drop format also fits nicely with vehicle harnesses that commonly share a similar type of layout, simplifying installation and reducing wiring throughout the vehicle.

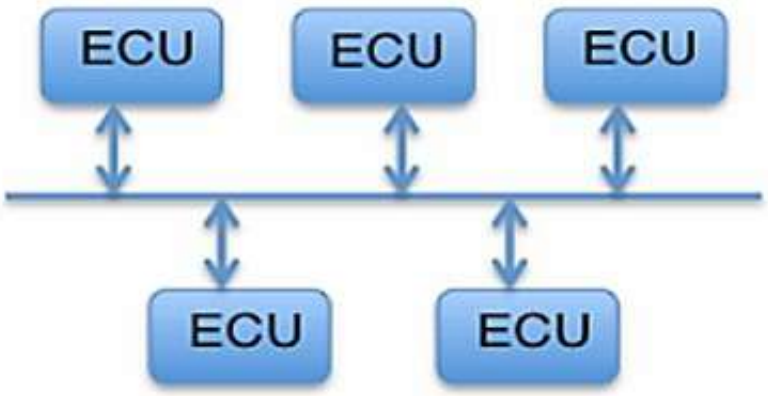


Figure 2.17: Multi-drop bus network.

B. **Star Network:** The FlexRay standard supports "Star" configurations, which consist of individual links that connect to a central active node. This node is functionally similar to a hub found in PC Ethernet networks, as shown in Figure 2.18. The active star configuration makes it possible to run FlexRay networks over longer distances or to segment the network in such a way that makes it more reliable should a portion of the network fail. If one of the branches of the star is cut or shortened, the other legs continue functioning. Since long runs of wires tend to conduct more environmental noise, such as electromagnetic emissions from large electric motors, using multiple legs reduces the amount of exposed wire for a segment and can help increase noise immunity.

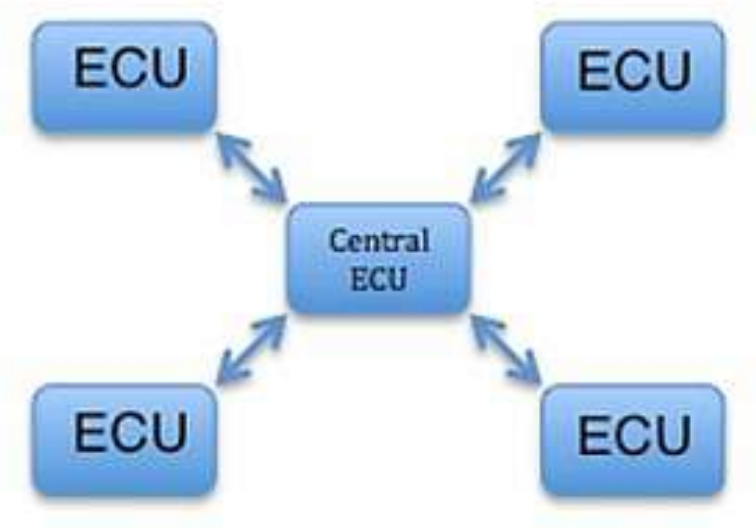


Figure 2.18: Star network.

C. **Hybrid Network:** The bus and star topologies can be combined to form a hybrid topology. Future FlexRay networks will likely consist of hybrid networks to take advantage of the ease-of-use and cost advantages of the bus topology while applying the performance and reliability of star networks where needed in a vehicle, as shown in Figure 2.19.

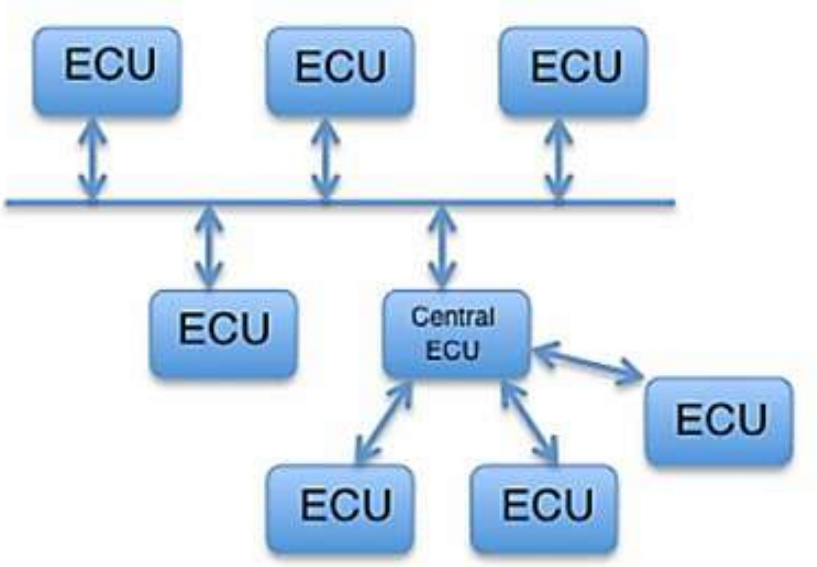


Figure 2.19: Hybrid network.

2.4.2 Applications of FlexRay

FlexRay is primarily used for safety-critical, high-performance applications in premium and performance vehicles where deterministic timing, fault tolerance, and high bandwidth are essential. Its dual-channel architecture and time-triggered communication make it ideal for x-by-wire systems and advanced driver assistance systems. Key Automotive Applications :

- X-by-wire systems: Brake-by-wire, steer-by-wire, and drive-by-wire replacing mechanical/hydraulic connections.
- Chassis and stability control: ABS, ESP, TCS, active suspension, and adaptive damping systems.
- Powertrain management: Engine control, transmission control, and torque vectoring.
- Advanced driver assistance (ADAS): Collision avoidance, adaptive cruise control, and pre-crash sensors.
- Active safety systems: Airbag deployment, restraint systems, and rollover protection.

2.5 MOST Protocol

The MOST (Media Oriented Systems Transport) is a high-speed multimedia network technology, serial bus-based daisy-chain topology or ring topology, and synchronous data communication protocol to transport the audio, video, voice, and data signals via plastic optical fiber (POF) (MOST25, MOST150) or electrical conductor (MOST50, MOST150) physical layers as shown in Figure 2.20.

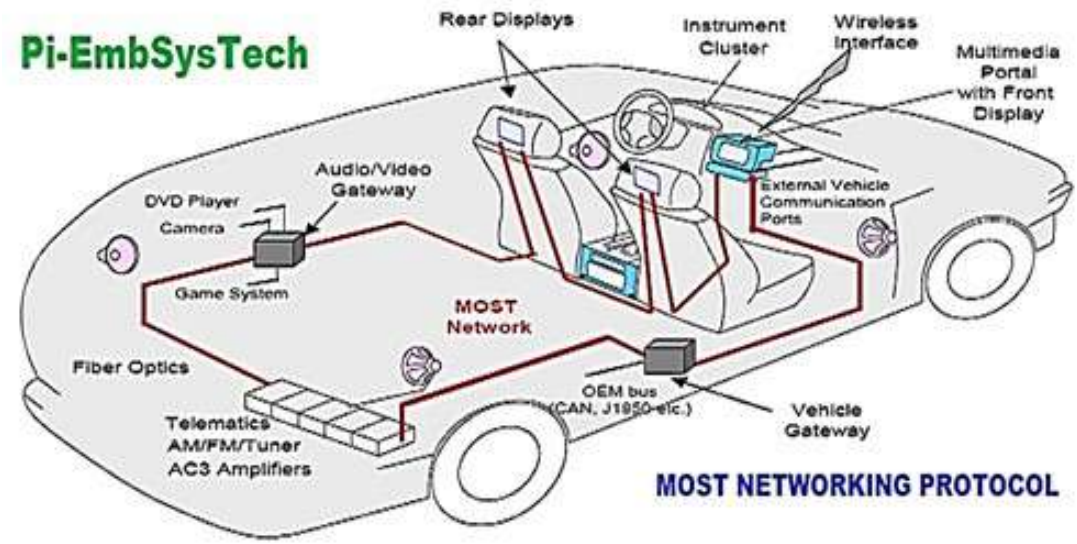


Figure 2.20: MOST Network Protocol.

2.5.1 MOST Protocol Architecture

MOST (Media Oriented Systems Transport) uses a ring topology with plastic optical fiber (POF) or electrical conductors to transmit high-bandwidth multimedia data (audio, video, packet data) at speeds up to 150 Mbps. The architecture follows the OSI model with specialized layers optimized for automotive infotainment [24].

- Key Architectural Components

A. **Physical Layer (Layer 1)**

- POF: 1mm core diameter with red LED transceivers
- Electrical options: Shielded/unshielded twisted pair for MOST50/150
- Ring topology:Daisy-chain connection forms a closed loop

B. **Data Link Layer (Layer 2)**

- Fixed frame rate: 48 kHz (one frame every 20.8 μs)
- Three transport channels per frame:
- Synchronous: Isochronous audio/video streaming
- Control: Real-time command/response messages
- Asynchronous: Packet data (IP, Ethernet in MOST150)

C. **Network Services (Layers 3-7)**

- MOST Network Services: Standardized protocol stack
- Basic Layer Services: Network/transport/session layers
- Application Socket Services: Presentation/application layers
- Timing Master: One node generates timing reference frames
- Frame Structure: Scalable bytes per 48 kHz frame, dynamically assigned to transport channels, enabling flexible bandwidth allocation for mixed multimedia traffic.
- Network Management: Up to 64 nodes with plug-and-play functionality and redundant double-ring option for safety-critical applications.

2.6 Conclusion

This chapter presented the CAN and LIN protocols as essential communication systems in modern vehicles. CAN ensures reliable high-speed communication for critical functions, while LIN provides a low-cost solution for secondary applications. With the growth of automotive electronic systems, advanced protocols such as FlexRay and MOST have emerged to support safety-critical and multimedia applications. Together, these protocols form a hierarchical communication architecture that balances performance, safety, and cost, and will continue evolving with the development of electric and autonomous vehicles.

CHAPTER

3

DESIGN AND IMPLEMENTATION OF THE PROTOTYPE

3.1 Introduction

In the previous chapters, we discussed ECUs as well as the CAN and LIN protocols. This chapter aims to provide a practical application of these protocols by designing a communication system between electronic nodes, in order to understand how data is exchanged within a vehicle and to compare their actual performance in terms of speed, reliability, and communication management.

3.2 System Architecture

The glass scanning system consists of three parts: data sensing, data processing and transmission via CAN bus, and data display as shown in the figure3.1.

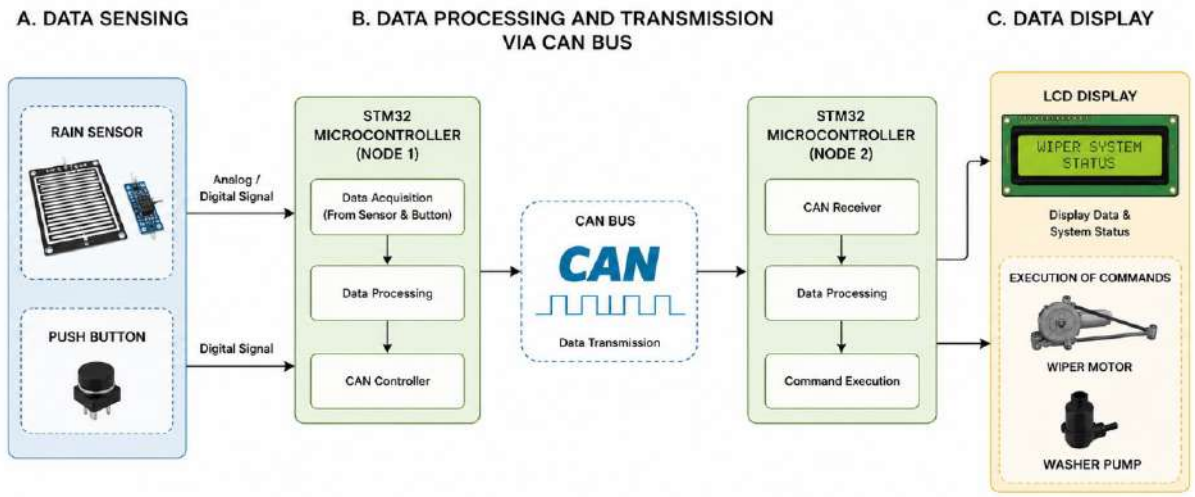


Figure 3.1: A diagram of an integrated manual and automated control system for glass spaces using the CAN protocol.

3.3 Description of materials used

3.3.1 STM32F103C8T6 (Blue Pill):

The STM32F103C8T6 "Blue Pill" is a popular, low-cost ARM Cortex-M3 development board featuring a 72MHz processor, 64KB Flash, and 20KB RAM. It is widely used as a high-performance alternative to Arduino for IoT and robotics, offering 37 GPIOs, USB connectivity as shown in the figure 3.2.

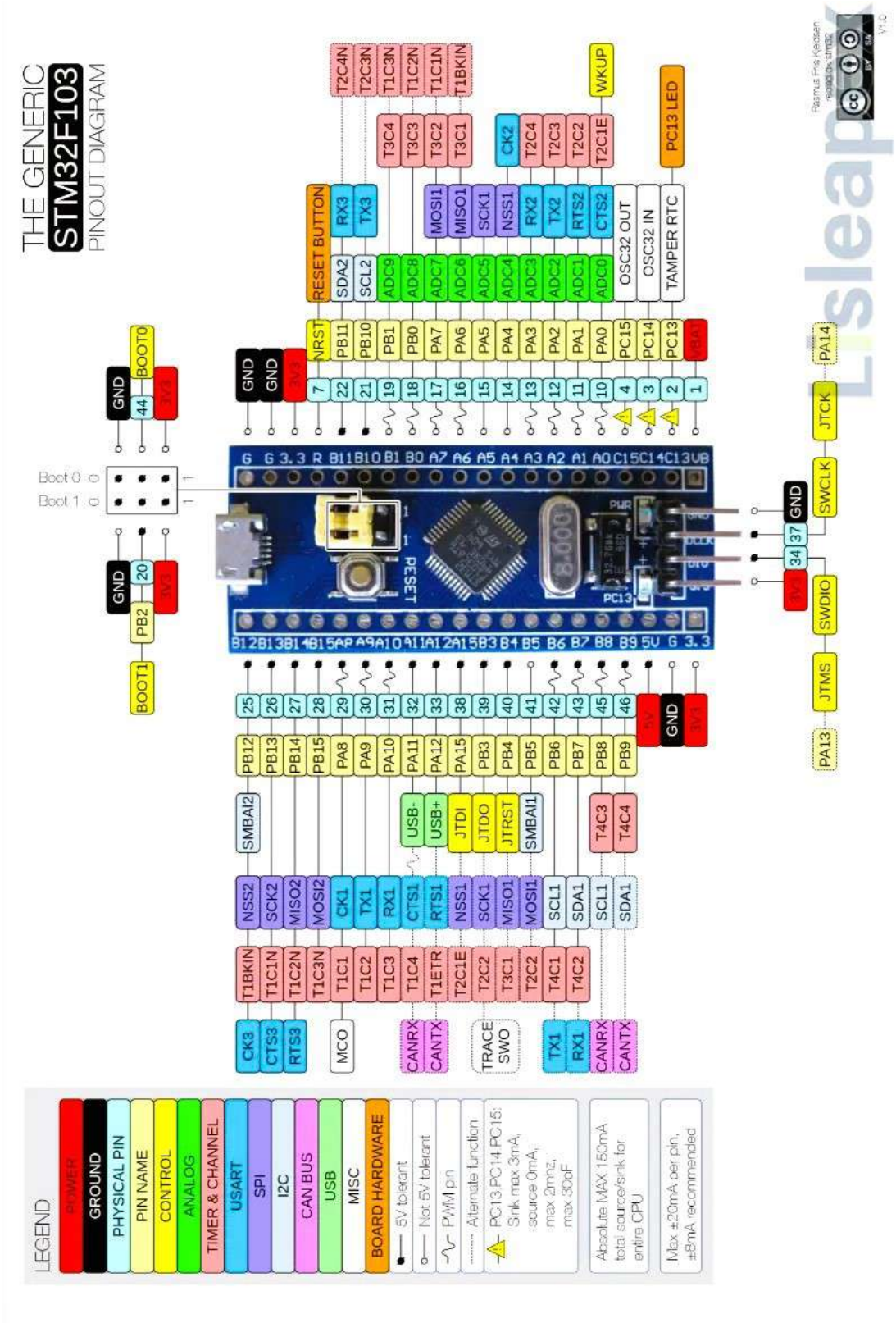


Figure 3.2: STM32F103C8T6 (the blue pill).

A. **Characteristics:**

- The STM32F103C8T6 (Blue Pill) is based on an ARM Cortex-M3 processor running at 72 MHz
- It features 64 KB of Flash memory and 20 KB of RAM.
- It operates at a voltage range of 2–3.6 V, with an internal 3.3 V regulator capable of delivering 300 mA.
- It includes 37 GPIO pins, along with support for USB 2.0, 3 USART interfaces, 2 SPI interfaces, and two I2C interfaces.
- Its dimensions are 53 x 23 mm, and it is priced under dollar5.

B. **Role:**The STM32 microcontroller acts as the system's central processing unit, managing all monitoring, control, and communication operations. At the transmitting node, it collects data from various sensors, such as the control button and rain sensor, via an analog-to-digital converter (ADC). It then identifies the information source and transmits it via the CAN bus. At the receiving node, it receives the incoming messages via the CAN network and processes them before displaying them on the LCD screen connected via the I2C protocol. In this way, the microcontroller ensures continuous and reliable data exchange between the different system components, leveraging the high reliability that the CAN protocol provides in embedded systems applications.

3.3.2 MCP2515

The CAN communication module illustrated in Figure is based on the MCP2515 controller, which is used to enable communication between microcontrollers (CAN nodes) via its CAN transceiver (TJA1050) as shown in the figure3.3.

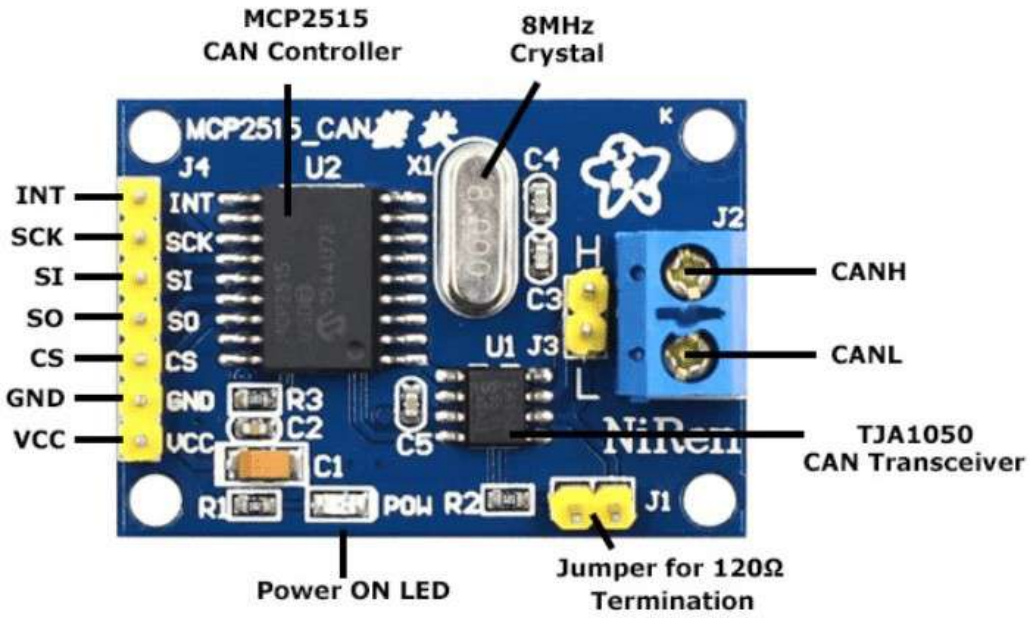


Figure 3.3: CAN (MCP2515).

A. **Characteristics**

- High communication reliability
- Real-time data transmission
- Error detection and correction
- Reduced wiring complexity
- Supports multiple nodes on the same bus
- Data rate up to 1 Mbps

B. **Role:**The CAN (Controller Area Network) protocol enables communication between electronic control units (ECUs) in automotive and embedded systems with high reliability and real-time data exchange.

3.3.3 Breadboard (Hitchman):

A breadboard, illustrated in Figure 3.4, is an essential tool in electronics for creating circuit prototypes without soldering components. This tool allows for the rapid testing and validation of electronic circuit designs before proceeding to the fabrication phase on a printed circuit board (PCB).

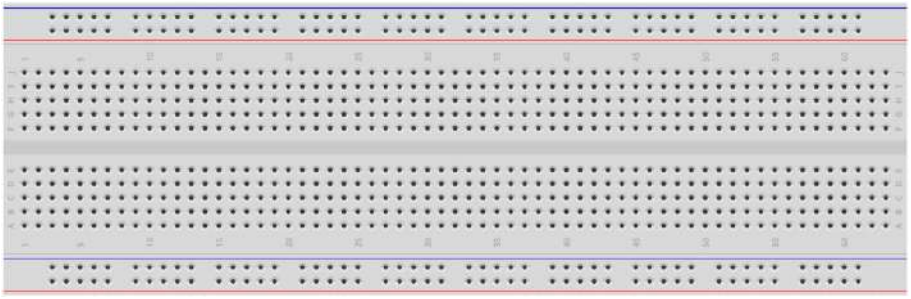


Figure 3.4: Breadboard.

A. **Characteristics:**

- Plastic base
- **Rows of interconnected holes:** These are generally organized into three zones: Vertical power rails Central zone Horizontal connection rails

B. **Role:**Fast modeling, flexibility, and time-saving

3.3.4 LCD I2C:

It is a text display (typically 16x2) integrated with an I2C adapter (often a PCF8574 chip), allowing the display to be controlled using just two wires instead of eight as shown in the figure 3.5.



Figure 3.5: LCD I2C.

A. **Characteristics**

- A 16×2 text display (2 lines, 16 characters per line).
- Operates via the I2C protocol (requires only two wires: SDA and SCL).
- Features a specific I2C address (e.g., 0x27 or 0x3F).

- Includes a controllable backlight.
- Low power consumption, making it suitable for embedded systems.
- Easy to interface with microcontrollers, such as STM32 and Arduino.

B. **Role:** The display screen is used to show texts and measured values such as temperature and speed, as well as the system's operating status, whether it is on (ON) or off (OFF).

3.3.5 Relay Module 5V

It is an electronic unit containing a 5 volt relay, used to enable microcontrollers such as Arduino or STM32 to control devices that operate at high voltages such as 220 volts as shown in the figure 3.6.



Figure 3.6: relay5V .

A. **Characteristics:**

- Operates at 5V (suitable for Arduino and STM32)
- Includes SRD-05VDC-SL-C relay
- Withstand voltage: Approximately 10A/250V AC or 10A/30V DC
- One control input (IN) + VCC and GND
- Features an LED indicator for status (ON/OFF)
- Often includes a transistor and a protection diode (sometimes with photo-conductivity)

B. **Role:** It acts as an electrical switch that allows the microcontroller to safely turn the pump on or off, while providing electrical isolation between the control circuit and the power circuit, which helps to protect the electronic components and increase the reliability of the system.

3.3.6 Button:

This is a push button/tactile switch used to input manual commands in electronic circuits as shown in the figure 3.7.



Figure 3.7: Button.

A. **Characteristics:**

- Functions as a momentary switch
- Typically has 4 pins (each pair is internally connected)
- Low operating voltage (3.3V or 5V)
- Compact size and easy to install on a breadboard
- Various colors (for identification only, no electrical effect)

B. **Role:** The push button is used to send an on/off signal to the controller. When pressed, the electrical circuit closes, generating a logic signal (high or low depending on the connection method). When released, the circuit returns to its original state and opens again, stopping the signal.

3.3.7 Pump:

It is a small DC water pump as shown in the figure3.8.

A. **Characteristics:**

- It usually works with 3V to 5V DC, depending on the model.

B. **Role:**The pump's function is to pump windshield washer fluid (water or a special liquid) from the reservoir to the spray nozzles on the windshield. When activated, it sprays the fluid onto the glass to remove dirt and improve visibility, helping the wipers to work efficiently while driving.



Figure 3.8: pump .

3.3.8 Power source:

This component is a Switching Power Supply (SMPS). Its main role is to convert the AC mains voltage (220V/110V) into a stable DC voltage such as 12V, 24V, or 5V used to power electronic circuits and embedded systems as shown in the figure3.9.



Figure 3.9: power source.

A. **Characteristics:**

- Output Voltage: Generally 12V DC or 24V DC depending on the model.
- Input Voltage: 110V–220V AC.
- High Efficiency: Reduces energy loss during conversion.
- Compact Design: Small size with metallic protection case.

B. **Role:**The power supply provides power to the DC motors of the windshield wiper system. It also ensures a stable and continuous flow of power, guaranteeing the reliable operation of the entire integrated system.

3.3.9 Rain Sensor Module:

This component is a Rain Sensor Module used to detect the presence and intensity of water droplets. Its main role in the project is to automatically activate the windshield wiper system when rain is detected on the sensor surface as shown in the figure 3.10.



Figure 3.10: Rain Sensor Module.

A. **Characteristics:**

- Operating Voltage: Typically 3.3V to 5V DC.
- Water Detection: Detects water droplets through conductive tracks on the sensing plate.
- Easy Interface: Compatible with Arduino, STM32, and other microcontrollers.

B. **Role:**The role of the rain sensor in the project is to detect the presence of water droplets or the intensity of rainfall on the windshield, and then send a signal

to the control unit (STM32) to make the appropriate decision, such as automatically operating the windshield wipers or adjusting their speed according to the amount of rain, which helps improve visibility and increase driving safety.

3.3.10 Motor DC:

A car windshield wiper motor is an electric motor (DC motor) used to move the wiper arms back and forth to clean the windshield or rear window of a car from rain and dirt as shown in the figure 3.11.



Figure 3.11: motor.

A. **Characteristics:**

- It typically operates on 12V DC.
- It provides sufficient torque to move the wipers.
- It is controlled by an electronic unit, relay, or driver such as the BTS7960.

B. **Role:**The role of the windshield wiper is to remove water and dirt from the windshield or rear window of a vehicle to improve visibility while driving. It works by moving blades back and forth to wipe the glass surface, and can be operated manually or automatically depending on the system used.

3.3.11 L298N Motor Driver:

The L298N Motor Driver is a motor control module that allows a microcontroller to regulate the speed and direction of DC motors. It supports PWM speed control and

bidirectional rotation, making it widely used in embedded, robotic, and automation systems as shown in the figure 3.12.

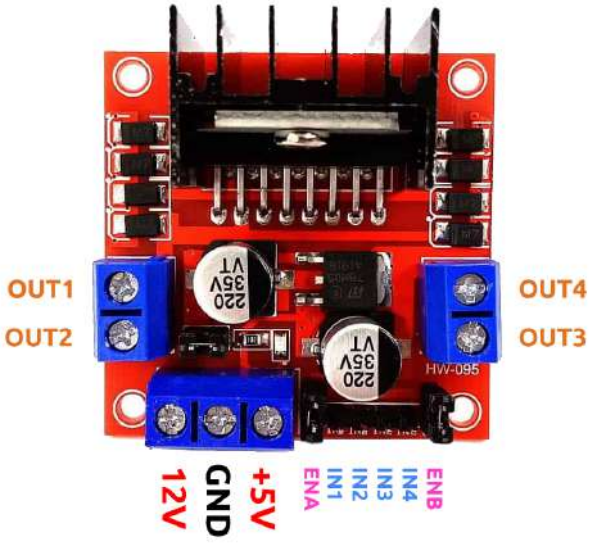


Figure 3.12: Motor Driver.

A. **Characteristics:**

- Controls up to two DC motors independently.
- Operating voltage range from 5 V to 35 V.
- Supports PWM signals for speed control.

B. **Role:** The L298N motor driver is used as an interface between the microcontroller and the DC motor. It enables the control of the motor's speed and rotation direction while supplying the higher current required by the motor, which cannot be provided directly by the microcontroller.

3.4 Development tools

The FT232RL chip is one of the most popular electronic chips used in embedded systems, as shown in Figure 3.13. It converts the connection from a USB port to a UART serial connection, allowing communication between the computer and microcontrollers such as the STM32 and Arduino.

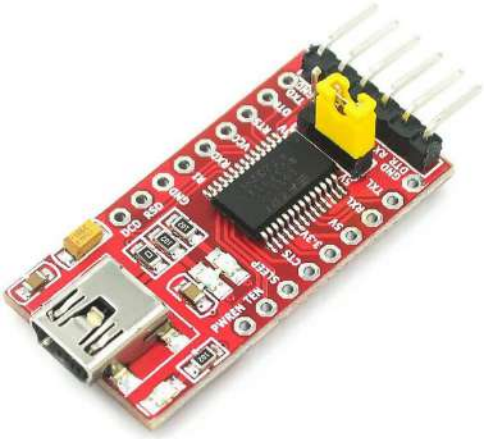


Figure 3.13: FT232RL chip .

The FT232RL is connected to the CAN (MCP12515) as shown in the figure3.14.

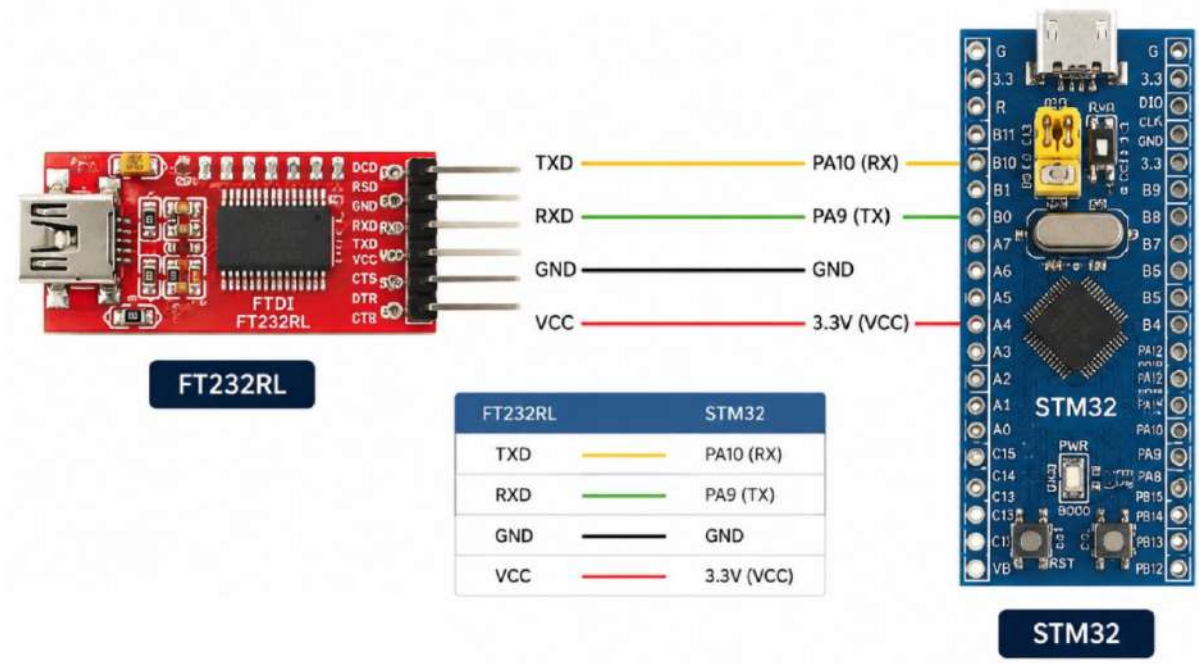


Figure 3.14: Connecting FT232RL to a CAN network (MCP12515).

After connecting the power supply, we transferred the code to the STM32 Blue Pill board using an FT232RL adapter, which provides a UART (Unified Serial Transport) connection between the computer and the microcontroller. After connecting the adapter to the board and setting the boot pins to BOOT0=1 to enter programming mode, the program was loaded via the Arduino Integrated Development Environment (IDE) through the serial port. The FT232RL adapter converted the data received from the USB port into UART signals, which were received by the microcontroller's built-in bootloader. The bootloader then stored the program in flash memory. Once

the programming process was complete, BOOT0=0 was reset, and the board was restarted, allowing the microcontroller to begin executing the stored code.

3.5 Electronic Connection Diagram

The electronic connection diagram includes the following connections between two units: CAN and STM32. The STM32 is the backbone of the message exchange and transmission process between the sender and receiver, as shown in the figure3.15.

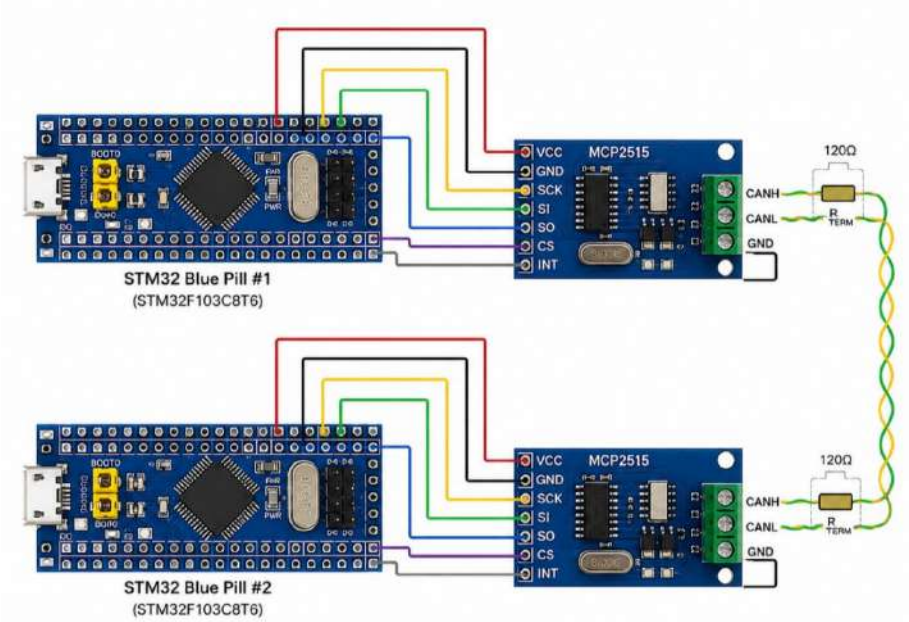


Figure 3.15: Interfacing between CAN and STM32.

Table 3.1: Connection between STM32 and CAN Module

STM32 Pin	**CAN Module Pin**
VCC	VCC
GND	GND
A4	CS
A5	SCK
A6	SO
A7	SI

3.6 Detailed description of the enhanced integrated systems

3.6.1 Windshield cleaning system:

A windshield cleaning system is essential to ensure clear driver visibility in adverse weather conditions, especially during rainfall, thus reducing the risk of accidents. In this project, this subsystem was designed to operate automatically using sensors, microcontrollers, CAN modules, a 12V DC windshield wiper motor, a power supply unit (L298N or BTS7960) to control the motor, and a water pump. The entire system is managed by a CAN network architecture, enabling communication between the various components of the electric vehicle, as illustrated in Figure 3.16.

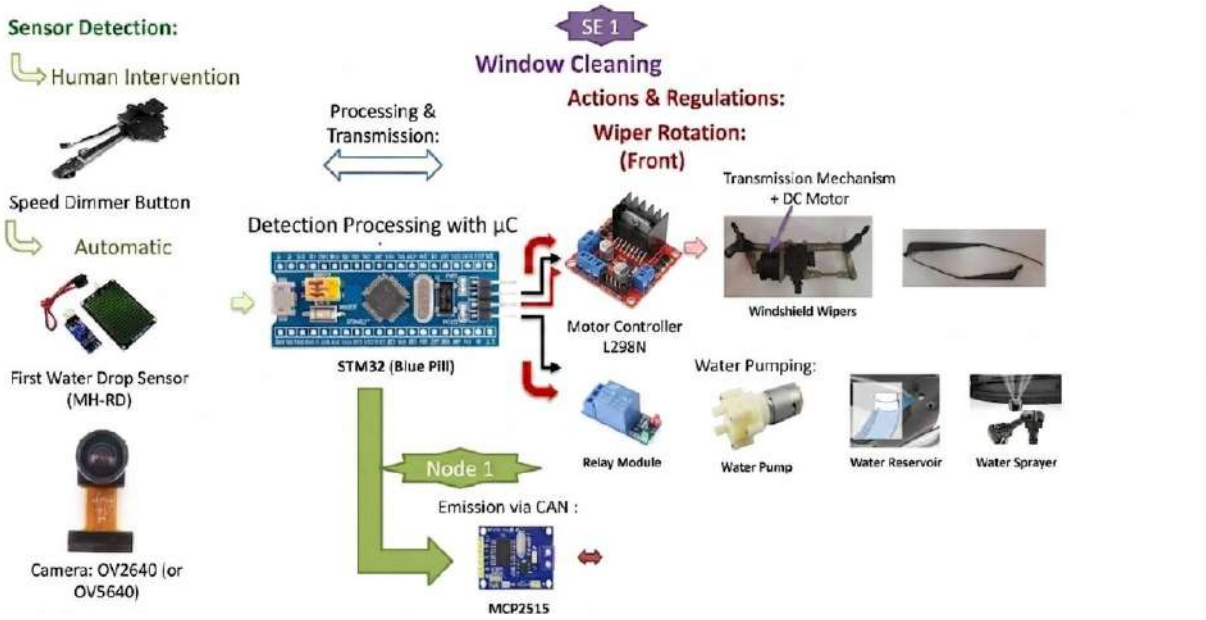


Figure 3.16: Windshield wiper structure via CAN system.

In the front windshield, two sensing units were implemented: a push button and a rain sensor. The collected data is transmitted through the CAN network to the microcontroller, which processes the received signal and activates the motor accordingly. The entire process is monitored using an LCD screen, allowing the display of the motor status and the type of sensor event, whether the windshield is cleaned manually by pressing the button or automatically in response to rainfall detected by the rain sensor. In addition, the wiper motor speed increases in real time as the rainfall intensity increases, ensuring an appropriate response to changing weather conditions. When the button is pressed, the pump is also activated to spray water onto the wind-

shield, since this mode is intended to give the vehicle owner the option to clean the glass when it contains dirt or when washing the car. The electronic components are connected as follows:

3.6.1.1 Connecting the button to the STM32

The following figure 3.17 shows how to connect the button to an STM32 device.

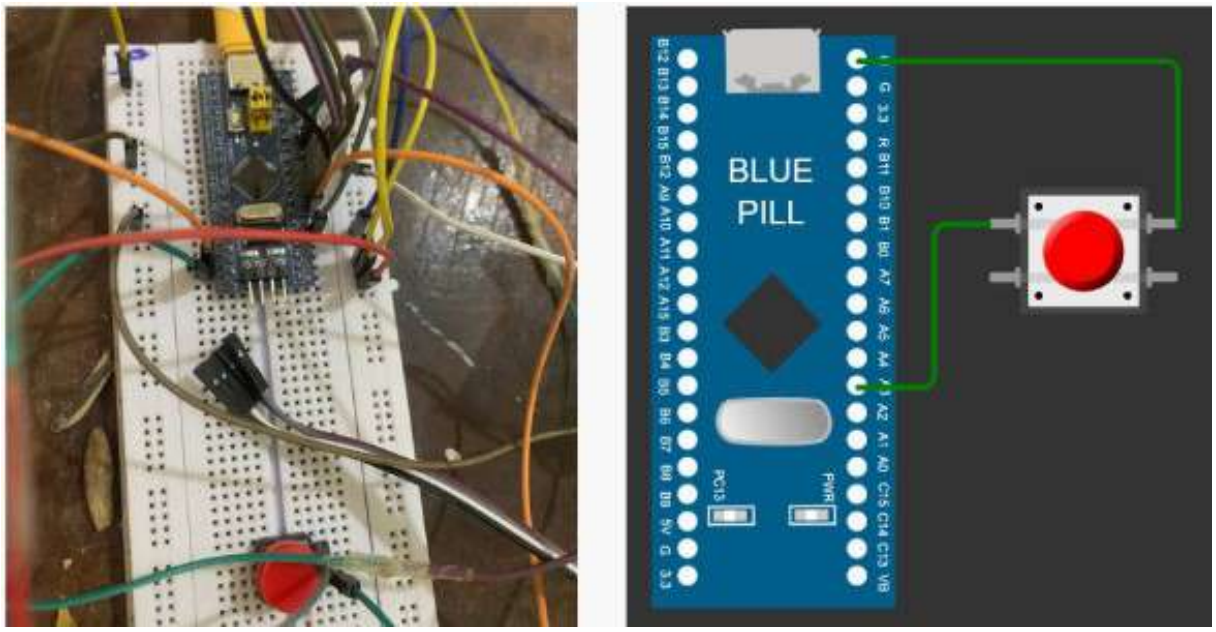


Figure 3.17: Connecting the button to the STM32.

- STM32 $(A3) \rightarrow Firstendofthebutton$
- STM32 $(GND) \rightarrow Thesecondendofthebutton$

Where GND must be common between them

3.6.1.2 Connecting the rain sensor to the STM32

- STM32 $(VCC) \rightarrow (VCC)$ rain sensor
- STM32 $(GND) \rightarrow (GND)$ rain sensor
- STM32 $(A0) \rightarrow (A0)$ rain sensor

The following figure 3.18 illustrates how to connect

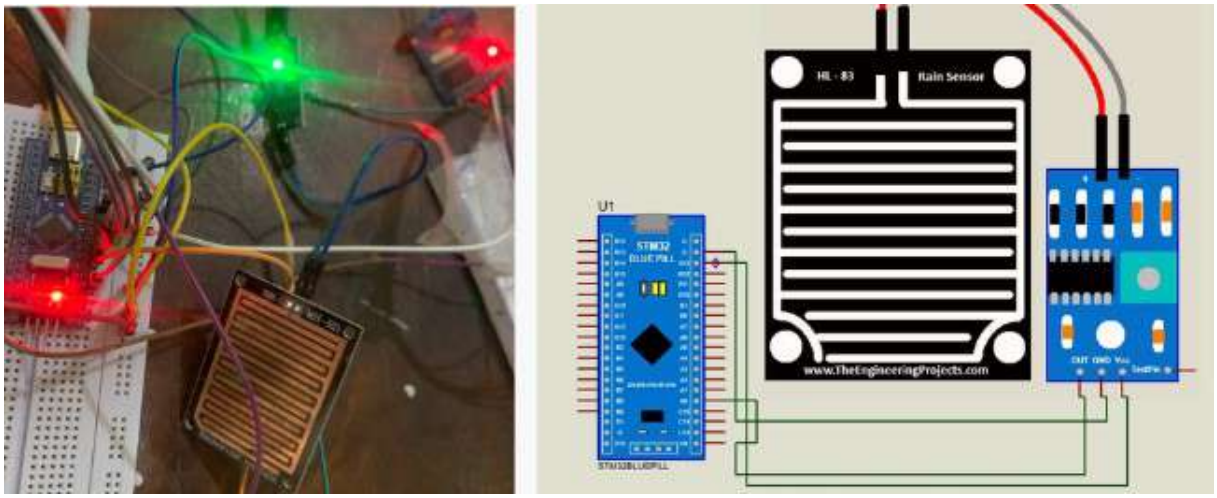


Figure 3.18: Connecting the rain sensor to the STM32.

Where GND must be common between them

3.6.1.3 Connecting an LCD screen to an STM32

The following figure 3.19 illustrates how to connect

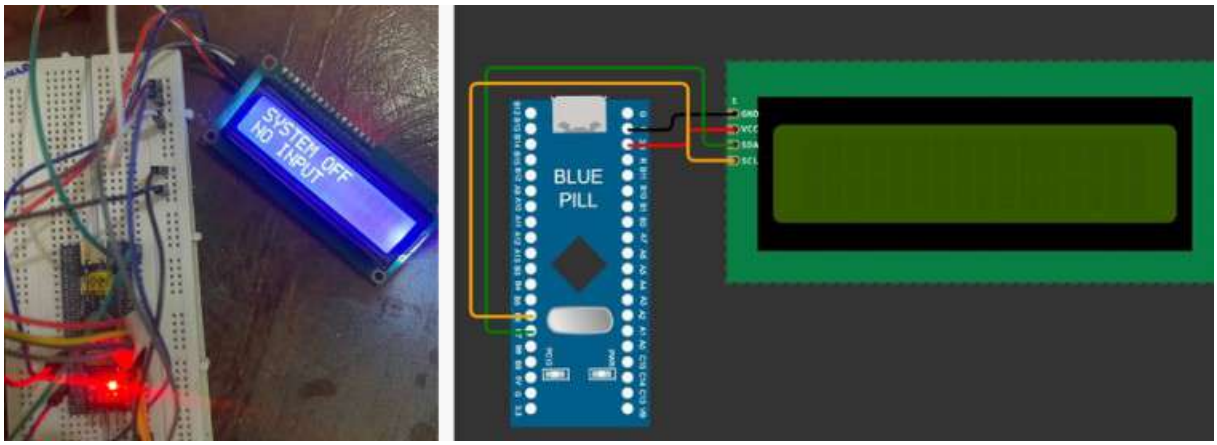


Figure 3.19: Connecting an LCD screen to an STM32.

- STM32 $(VCC) \rightarrow (VCC)$ LCD
- STM32 $(GND) \rightarrow (GND)$ LCD
- STM32 $(SDA) \rightarrow (B7)$ LCD
- STM32 $(SCL) \rightarrow (B6)$ LCD

3.6.1.4 Connecting an electric motor to an STM32 controller

The following figure 3.20 illustrates how to connect

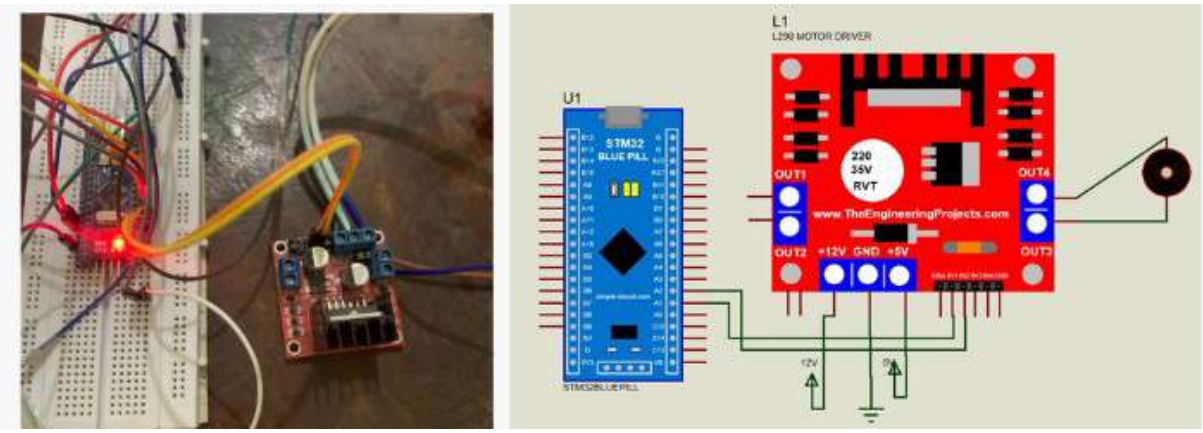


Figure 3.20: Connecting an electric motor to an STM32 controller.

- STM32 $(A1) \rightarrow (N1)$ motor
- STM32 $(A2) \rightarrow (N2)$ motor
- STM32 $(5V) \rightarrow (5V)$ motor

3.6.2 Rear window cleaning system:

A single sensor unit, the button, is mounted in the rear window. The collected data is transmitted via a CAN bus to the microcontroller, which processes the received signal and activates the motor accordingly.

The entire process is monitored by an LCD screen, which displays the motor status, the type of sensor event, and whether the windshield is being cleaned. Pressing the button also activates the pump to spray water onto the windshield, allowing the car owner to clean the glass when it gets dirty or during a car wash. The electronic components are connected as shown in the figure 3.21:



Figure 3.21: Rear window wiper structure via CAN system..

The connection remains the same, but this time we connect it to the second node using the second STM32 microcontroller. The button connection differs; we add another button for the rear motor, in addition to a motor controller. The following figure 3.23 illustrates the equipment used to create this ecosystem:

3.6.2.1 Connecting an LCD screen to an STM32

Figure 3.22 shows how to connect the button to the control unit.

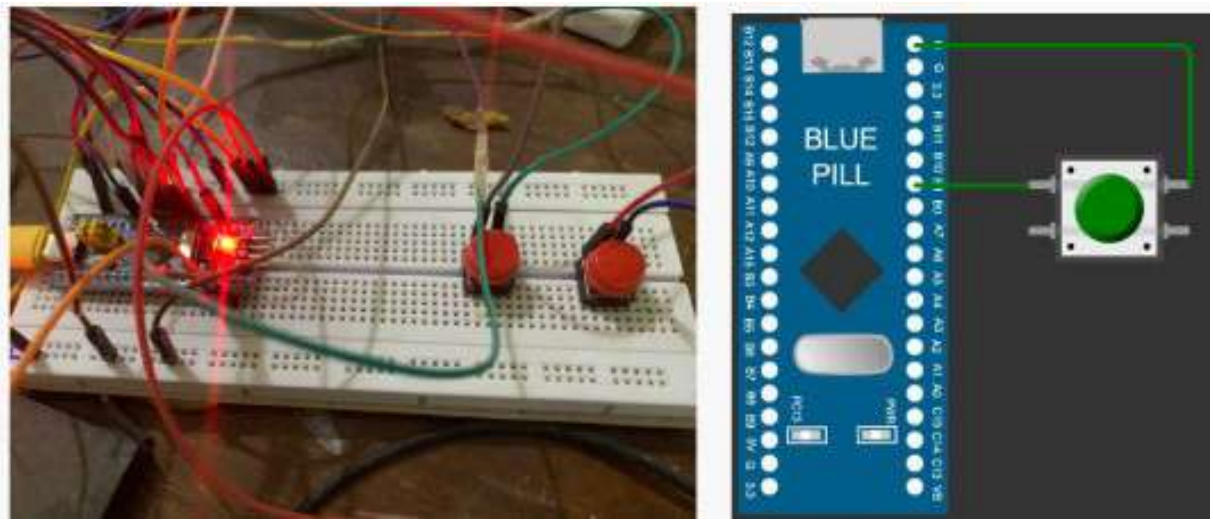


Figure 3.22: Connecting the button to the STM32.

- STM32 $(B0) \rightarrow Firstendofthebutton$
- STM32 $(GND) \rightarrow Thesecondendofthebutton$

Where GND must be common between them

3.6.3 Discussion of the results obtained:

A. Pressing the button on the first node activates the windshield wiper motor and pump, which then spray water intermittently. This is displayed on the LCD screen as shown in the figure 3.23.

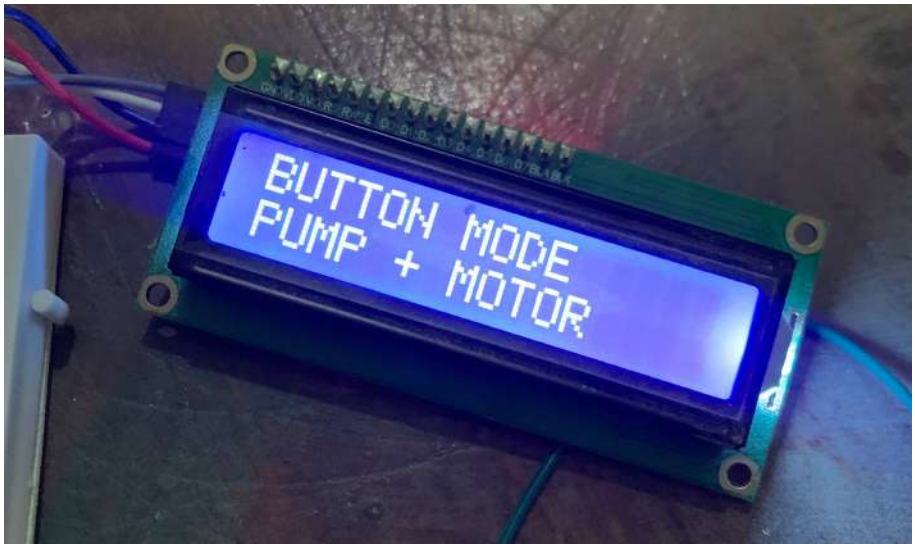


Figure 3.23: Display the status of the pump and motor.

The following results are displayed on the serial port monitoring screen in the Arduino IDE (sender's side) as shown in the figure 3.24.

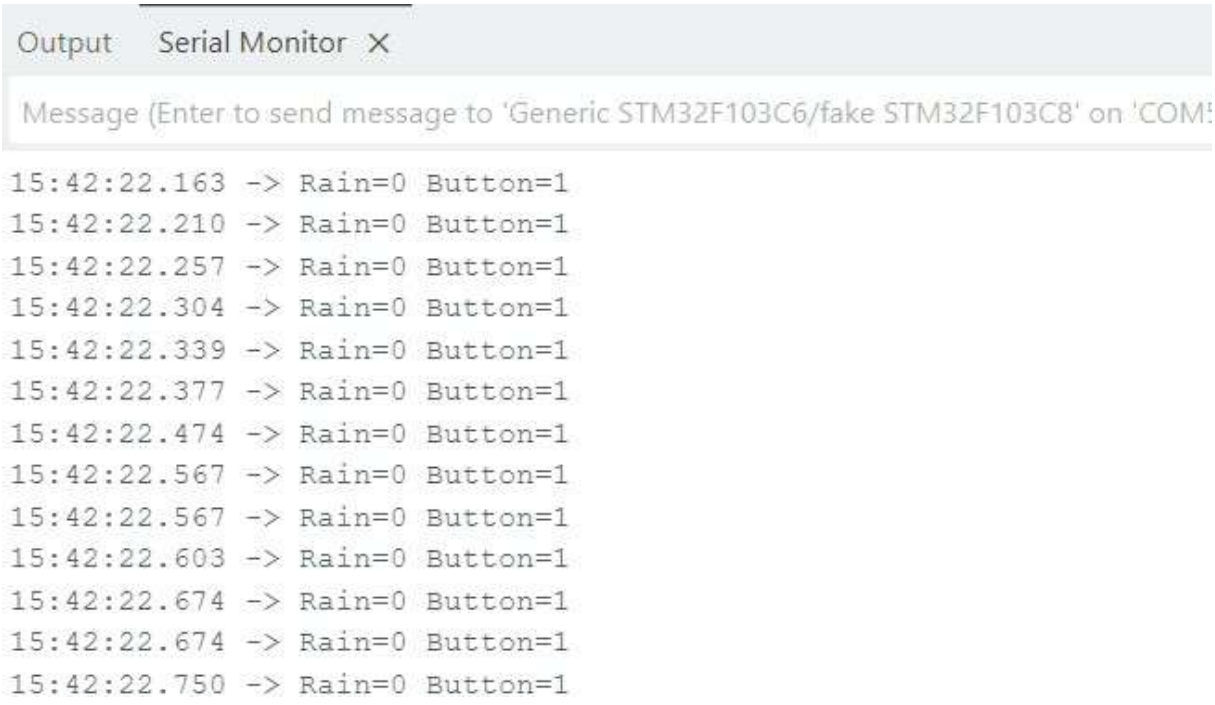


Figure 3.24: Result of pressing the button from the sender.

The following results are displayed on the serial port monitoring screen in the Arduino IDE (receiver's side) as shown in the figure 3.25.

Output Serial Monitor ×

Message (Enter to send message to 'Generic STM32F103C6/fake STM32F103C8' on 'COM

```
15:42:22.163 -> Rain=0 Button=1
15:42:22.210 -> Rain=0 Button=1
15:42:22.257 -> Rain=0 Button=1
15:42:22.304 -> Rain=0 Button=1
15:42:22.339 -> Rain=0 Button=1
15:42:22.377 -> Rain=0 Button=1
15:42:22.474 -> Rain=0 Button=1
15:42:22.567 -> Rain=0 Button=1
15:42:22.567 -> Rain=0 Button=1
15:42:22.603 -> Rain=0 Button=1
15:42:22.674 -> Rain=0 Button=1
15:42:22.674 -> Rain=0 Button=1
15:42:22.750 -> Rain=0 Button=1
```

Figure 3.25: Result of pressing the button on the receiver.

We observe a high response speed when the button is pressed, as the results show, indicating that we have met the required condition. The CAN (MCP2515) device confirms its speed at 1 Mbps.

B. When the rain sensor detects rainfall, the motor starts automatically, but the pump does not operate. This is displayed on the LCD screen as shown in the figure 3.26.



Figure 3.26: Display the status of the motor and rain sensor .

The following results are displayed on the serial port monitoring screen in the Arduino IDE (sender's side) as shown in the figure 3.27.

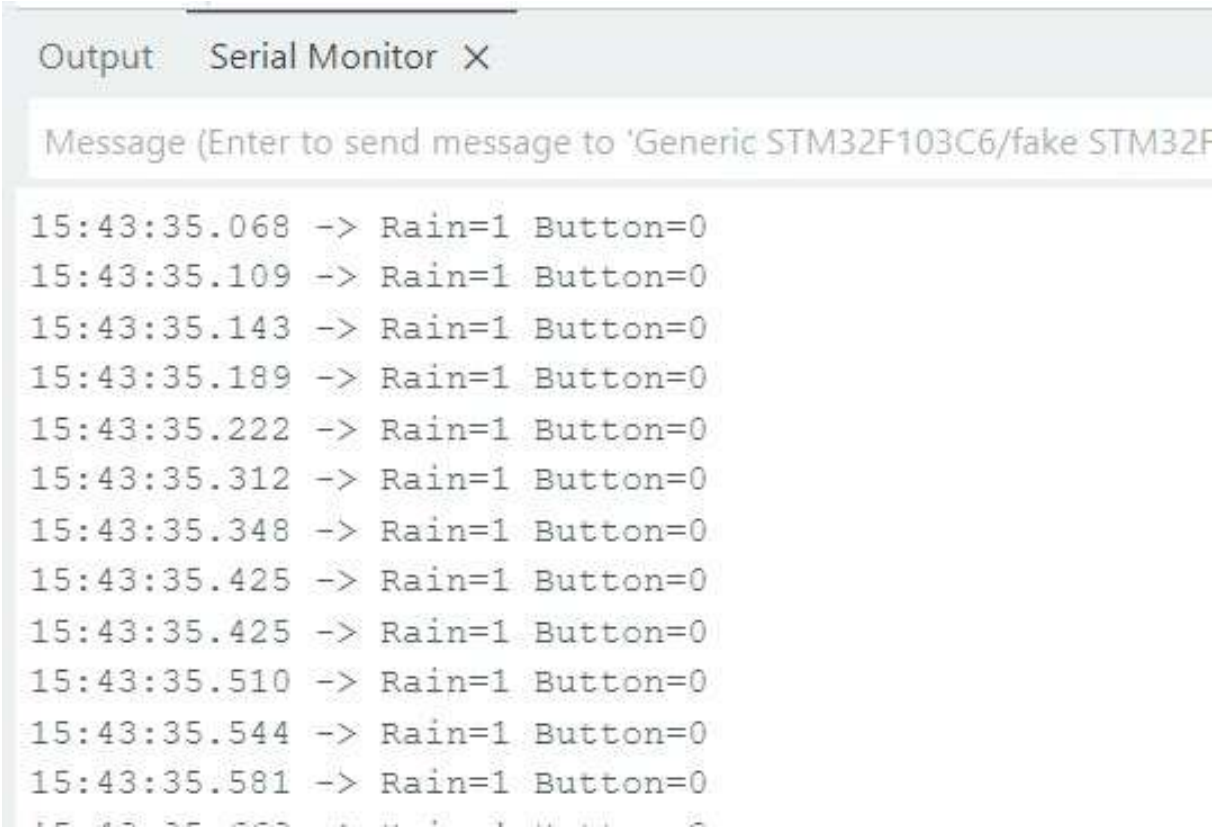


Figure 3.27: Rain sensor result from receiver control unit.

We observed a high response speed during rainfall, specifically from the MCP2515 sensor, as the results show, indicating that the required conditions were met. The CAN (MCP2515) device confirms its speed at 1 Mbps.

C. Under normal circumstances, the system remains inactive, as shown in the following figure3.28.

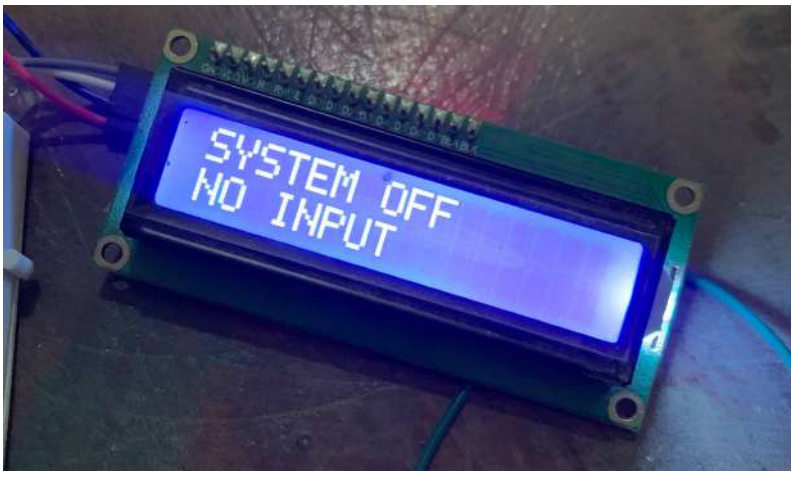


Figure 3.28: The system in normal mode.

The following results are displayed on the serial port monitoring screen in the Arduino IDE as shown in the figure 3.29.

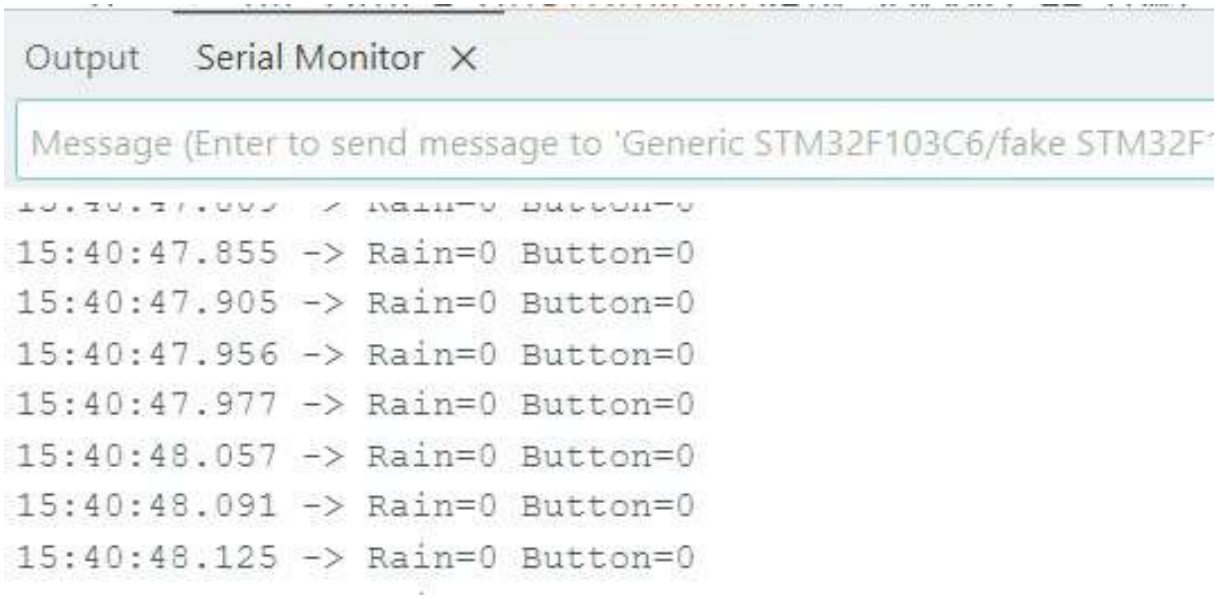


Figure 3.29: Normal condition.

3.7 PCB design and manufacturing

After completing the design of the windshield wiper project using the STM32 Blue Pill microcontroller and CAN bus modules, along with the electronic components, we aimed to improve the system's efficiency in terms of organization and practicality. Therefore, a printed circuit board (PCB) was designed. The component layout was optimized, random wiring was reduced, and EasyEDA software (Figure 3.30) was used for this purpose.

EasyEDA is an Electronic Design Automation (EDA) tool used for schematic capture and Printed Circuit Board (PCB) design. It enables users to create electronic circuits, design PCB layouts, route electrical connections, and generate manufacturing files for PCB fabrication. Its integrated tools and user-friendly interface simplify the design process and support the development of reliable electronic systems.

In these two nodes, the first node was dedicated to the windshield wiper system, and the second to the rear wiper system. We first created a file for each project and named them Node 1 and Node 2. Then we added all the necessary components by searching for them in the search box, placed them in their designated locations, and divided them into several units to make it easier for me in the next stage to place each component next to its associated component.

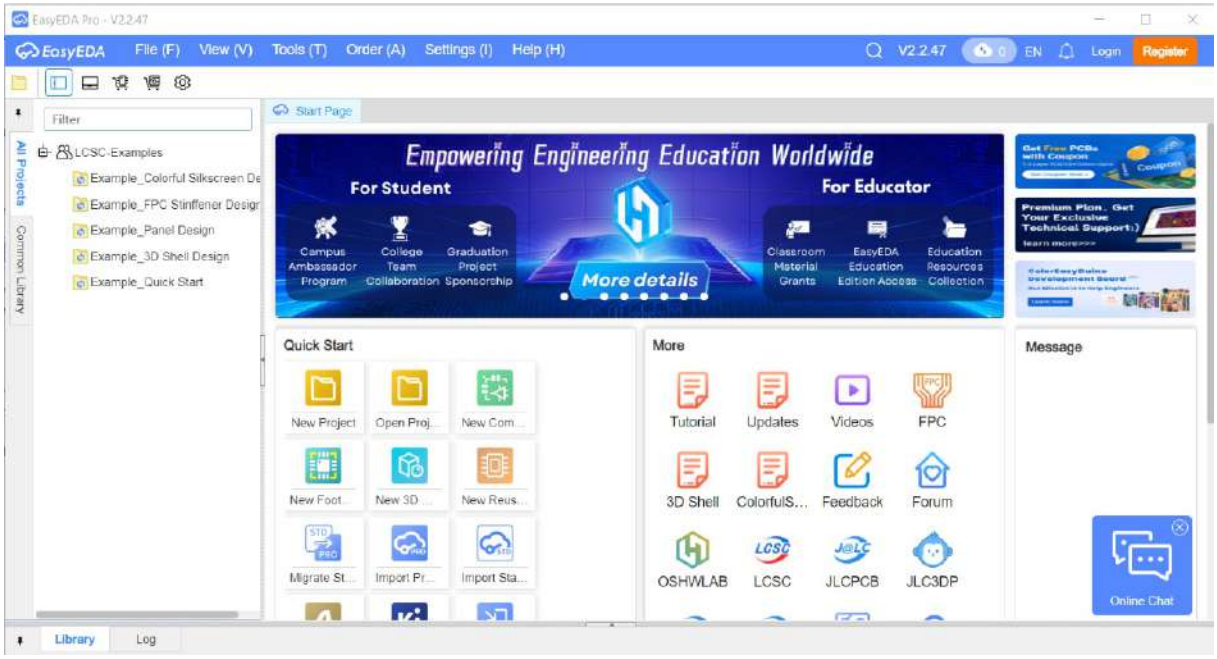


Figure 3.30: EasyEDA program interface [25]

3.7.1 Node 1

The node is divided into 2 parts.

3.7.1.1 Parts 1

As I mentioned earlier, I divided the project into several units, as shown in the figure3.31.

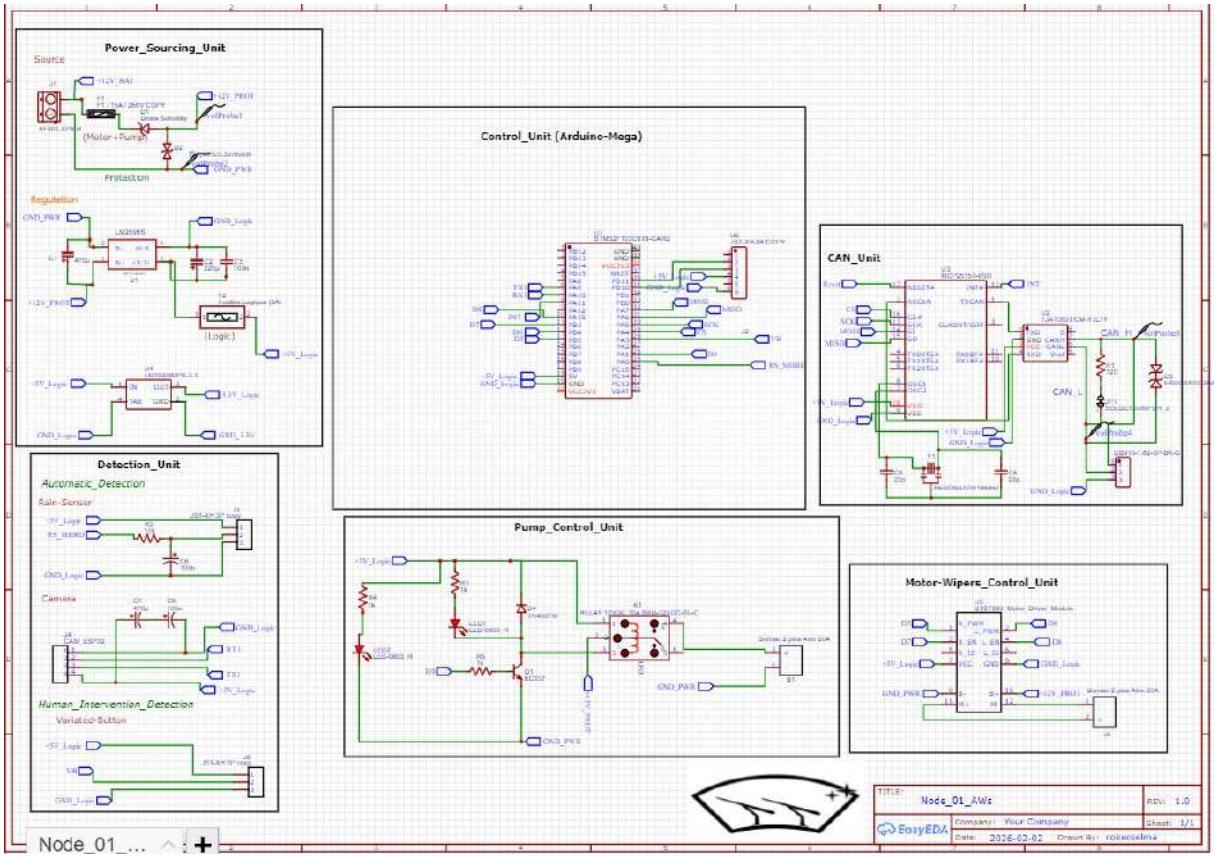


Figure 3.31: Windshield wiper control system.

A. **Power Supply Unit:** In this unit, I supplied the motor with the necessary power (12V). It was connected to the motor controller to control it and to power the STM32 microcontroller via a USB port. The microcontroller then sends commands to the motor's dedicated ports.

B. **Control Unit:** This unit consists of the STM32 microcontroller, which acts as a control center and interface, connecting to all peripherals and sending and receiving data.

C. **CAN Bus Unit:** This is the core of the project, responsible for sending and receiving information. I connected it to the microcontroller via its dedicated ports.

D. **Detection Unit:** This unit includes a rain sensor, which acts as a transmitter, and a button for sending commands.

E. **Pump Control Unit:** I connected this unit to the pump via its dedicated port.

F. **Windshield Wiper Control Unit** I connected the wiper motor to the motor control unit.

3.7.1.2 Parts 2

In the final stage, the electronic schematic was converted into a printed circuit board (PCB). The electronic components were arranged systematically, and the copper traces were connected in a way that minimizes interference and improves the quality of the electrical signals. The red lines represent the traces on the top layer, while the blue lines represent the traces on the bottom layer. The gray circles are the metal holes (connectors) that allow signal transfer between the two layers. Practical aspects, such as ease of installation and maintenance, were also considered, as illustrated in the following figures3.32.

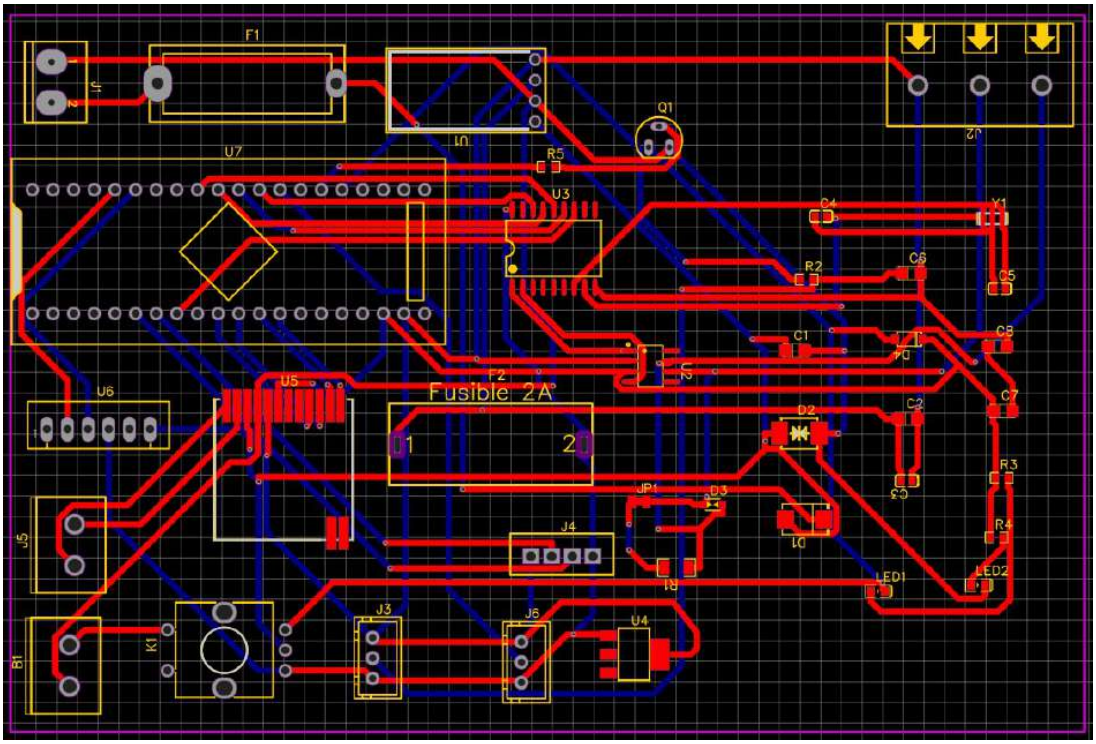


Figure 3.32: PCB wiring.

This also allowed me to see the final part of the 3D design, making it easier for me to see if I had placed the electronic components correctly or if there was an error in the way any of the components were installed, as shown in the figure3.33.

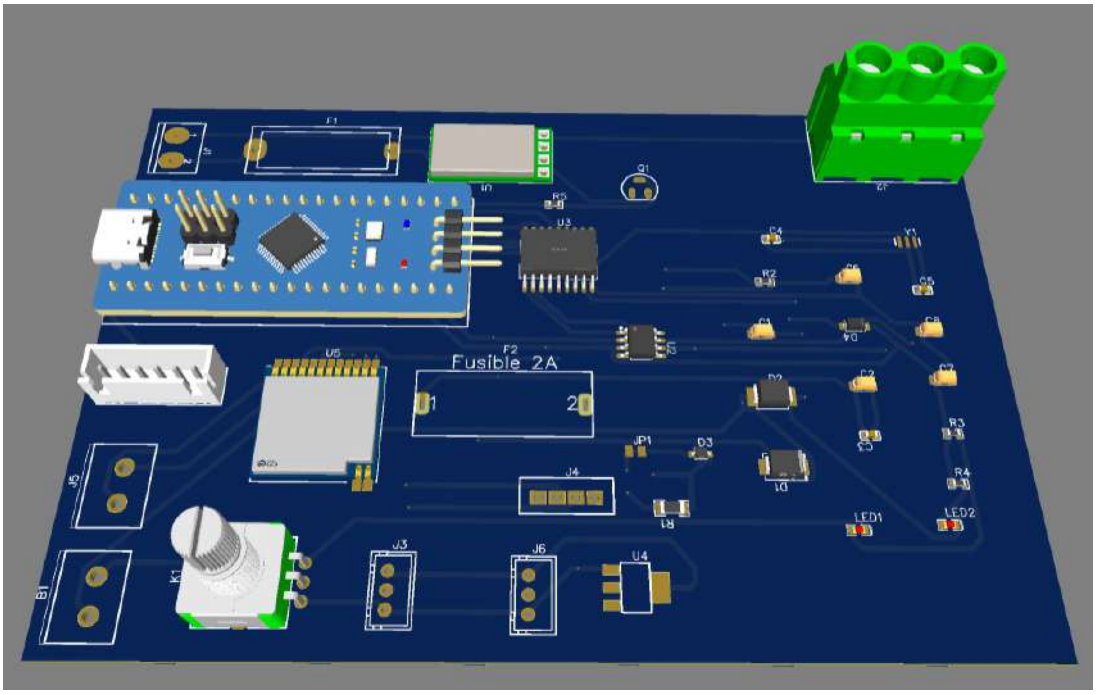


Figure 3.33: 3D drawing of the pcb for node 1.

3.7.2 Node 2

The node is divided into 2 parts.

3.7.2.1 Parts 1

As I did in the first node 1 divided the project into several units, as shown in the figure3.34.

A. **Power Supply Unit:** In this unit, I supplied the motor with the necessary power (12V). It was connected to the motor controller to control it and to power the STM32 microcontroller via a USB port. The microcontroller then sends commands to the motor's dedicated ports.

B. **Control Unit:** This unit consists of the STM32 microcontroller, which acts as a control center and interface, connecting to all peripherals and sending and receiving data.

C. **CAN Bus Unit:** This is the core of the project, responsible for sending and receiving information. I connected it to the microcontroller via its dedicated ports.

D. **Detection Unit:** This module includes a motor start button connected to the STM32 microcontroller.

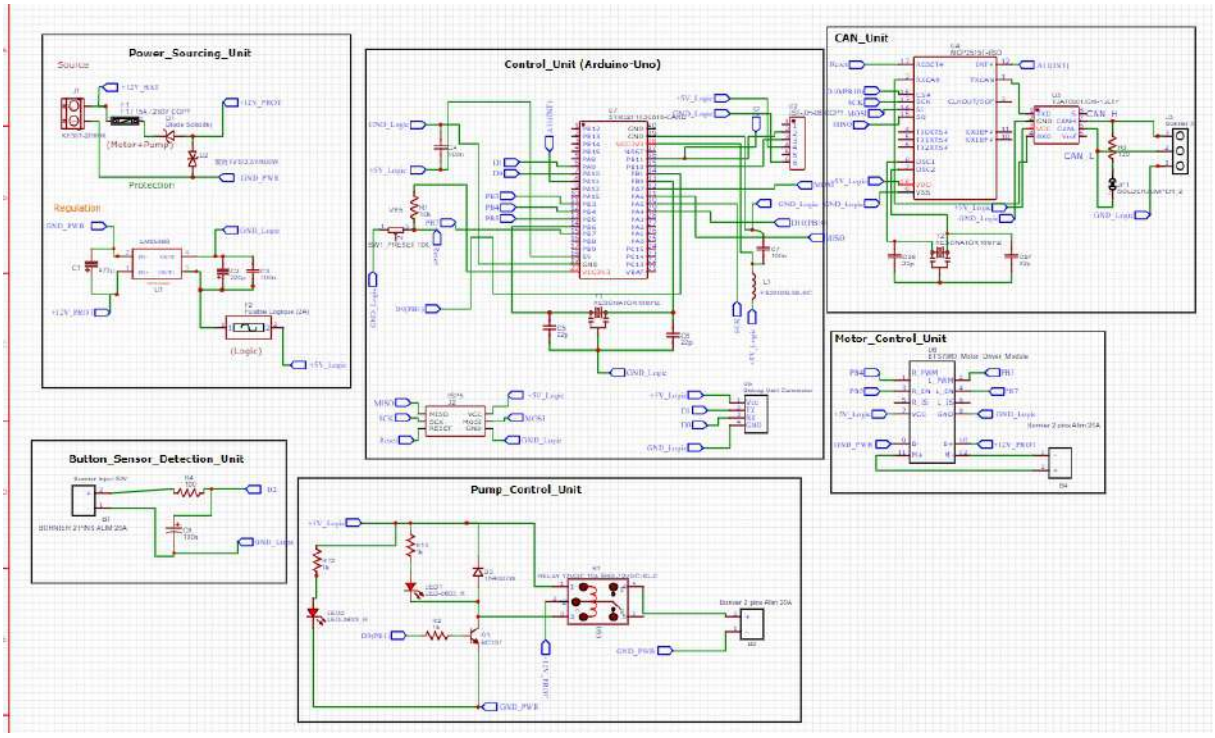


Figure 3.34: Rear window wiper control system.

E. **Pump Control Unit:** I connected this unit to the pump via its dedicated port.

3.7.2.2 Parts 2

In the final stage, the electronic schematic was converted into a printed circuit board (PCB). The electronic components were arranged systematically, and the copper traces were connected in a way that minimizes interference and improves the quality of the electrical signals. The red lines represent the traces on the top layer, while the blue lines represent the traces on the bottom layer. The gray circles are the metal holes (connectors) that allow signal transfer between the two layers. Practical aspects, such as ease of installation and maintenance, were also considered, as illustrated in the following figures3.35.

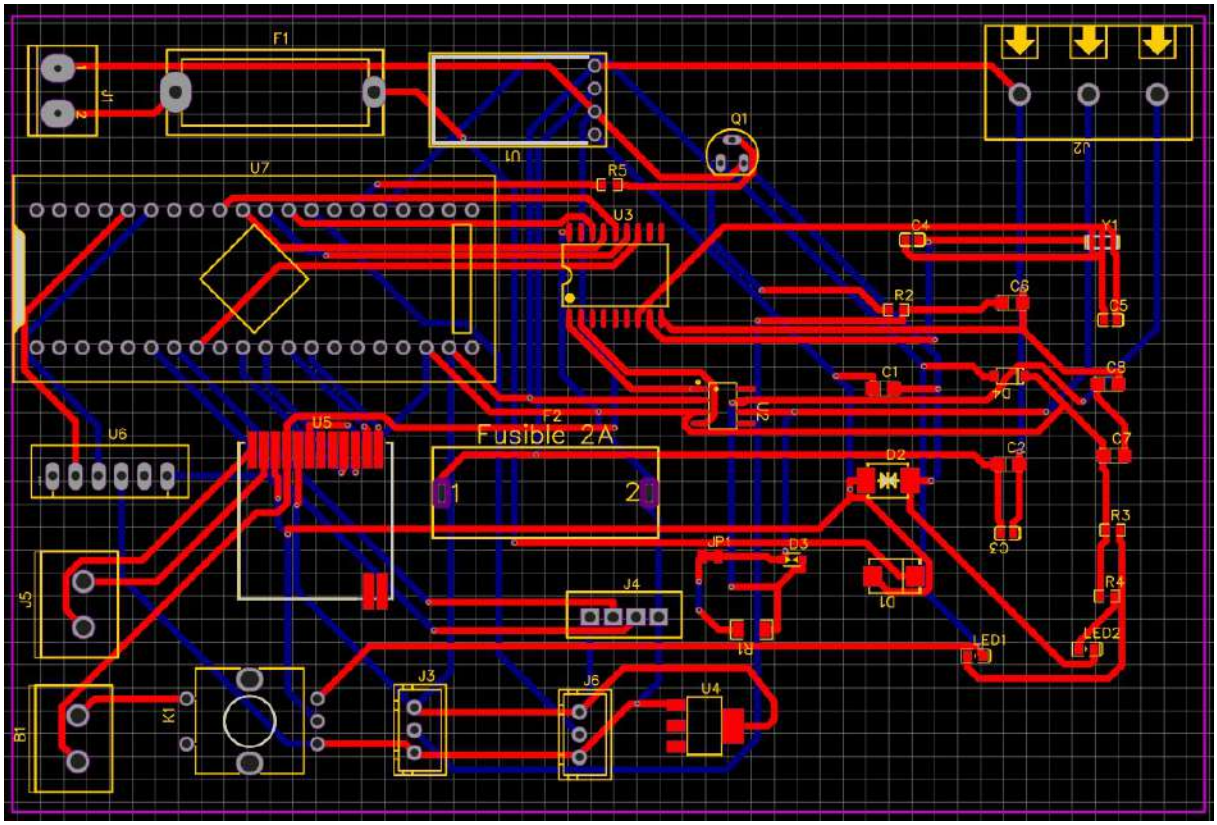


Figure 3.35: PCB wiring.

This also allowed me to see the final part of the 3D design, making it easier for me to see if I had placed the electronic components correctly or if there was an error in the way any of the components were installed, as shown in the figure3.36.

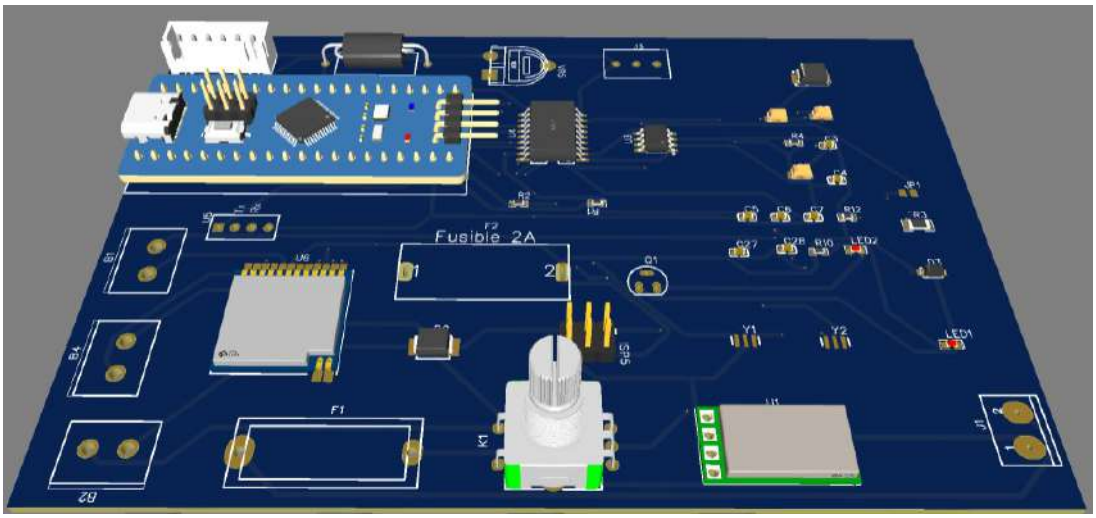


Figure 3.36: 3D drawing of the pcb for node 2.

3.8 Conclusion

This chapter discusses the implementation of an intelligent vehicle windshield scanning system using STM32 and CAN bus. The system successfully exchanged data reliably and quickly between units. Furthermore, the design was optimized by creating a PCB using EasyEDA, which increased system reliability and simplified installation and maintenance.

General Conclusion

This project successfully designed and implemented a communication system based on the CAN protocol for managing embedded systems in a miniature electric vehicle prototype. The main objective was to demonstrate the feasibility of reliable, fast, and prioritized communication between different electronic nodes using accessible components such as STM32 microcontrollers (Blue Pill) and MCP2515 modules.

Two operating modes were successfully implemented. The first is manual mode, triggered by a push button, which activates the wiper motor and water pump. The second is automatic mode, activated by a raindrop sensor (MH-RD), which simulates rain detection and automatically triggers the windshield cleaning system. In both cases, the system responded correctly, and the LCD screen displayed the corresponding mode in real time, showing "BUTTON MODE" or "RAIN MODE".

The CAN communication between the transmitter node and the receiver node was successfully established, validating the proposed distributed architecture. This work demonstrates that a low-cost, open, and flexible diagnostic and control solution can be built using standard electronic components, making it accessible for laboratories, research centers, and students.

Several perspectives can be considered for future work. These include integrating the LIN protocol for less critical subsystems such as electric windows and mirrors, adding an OBD-II module for advanced onboard diagnostic functions, expanding the system with additional sensors (temperature, humidity, CO2), and migrating to more powerful microcontrollers like STM32F4 or STM32H7 to support higher CAN traffic.

In conclusion, this project not only validated the use of the CAN protocol for managing embedded systems in a miniature electric vehicle but also provided a solid, educational, and low-cost foundation for future research and development in the field of electric vehicles and embedded networks.

BIBLIOGRAPHY

[1] S. Fu, Software-defined vehicles drive next-gen auto architectures, Electric Vehicles ResearchLast accessed on: [06/02/2026] (2025).
URL https://www.electricvehiclesresearch.com/articles/33582/software-defined-vehicles-drive-nextgen-auto-architectures

[2] K. Keshyagol, S. Kulkarni, Artificial intelligence embedded platform for battery management and autonomous navigation in a prototype electric vehicle, Discover Sustainability 6 (1) (2025) 1474. doi:10.1007/s43621-025-02310-6.

[3] M. G. Mostafa, et al., Energy management and power distribution for battery/ultracapacitor hybrid energy storage system in electric vehicles with regenerative braking control, Mathematical and Computational Applications 30 (4) (2025) 82. doi:10.3390/mca30040082.

[4] C.-Y. Lin, P. L. Li, C.-H. Li, W.-S. Chiang, Investigation on igbt failure effects of ev/hev inverter using fault insertion hil testing, 2015.

[5] ISO/SAE, Road vehicles — on-board diagnostic communication — part 3: Zero emissions vehicle propulsion systems on uds (zevonuds), last accessed on: [06/02/2026] (2025).
URL https://www.iso.org/standard/86344.html

[6] T. Agarwal, Embedded systems role in automobiles and their working, ElProCus - Electronic Projects for Engineering Students (Sep. 2019).

[7] Conquer Electronics, Fuse application field - pdu, Conquer Electronics Co., Ltd. (2022).

[8] x engineer, On-board diagnostics (obd) – introduction to the modes of operation (diagnostic services), x-engineer.org, last accessed on: [06/02/2026] (2016).
URL https://x-engineer.org/on-board-diagnostics-obd-modes-operation-diagnostic-services/

[9] PiEmbSysTech, Can protocol, PiEmbSysTech - Embedded Systems & VLSI Lab, last accessed on: [06/02/2026] (2018).
URL https://piembsystech.com/can-protocol/

[10] Ditex, Can bus (controller area network), AutoDitex, last accessed on: [06/02/2026] (2020).
URL https://autoditex.com/page/can-bus--controller-area-network-34-1.html

[11] CSS Electronics, Can bus explained - a simple intro, last accessed on: [06/02/2026] (Oct. 2021).
URL https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial

[12] T. Alferink, Practical tips: CAN-Bus, KMP Drivetrain Solutions, last accessed on: [06/02/2026] (Aug. 2017).
URL https://www.kmpdrivetrain.com/paddleshift/practical-tips-can-bus/

[13] D. Coleman, Automotive communication networks, part II: CAN bus, Snap-on Diagnostics News Center, last accessed on: [06/02/2026] (2025).
URL https://www.snapon.com/EN/US/Diagnostics/News-Center/CAN-Bus

[14] Renesas Electronics Corporation, ISL7202xSEH Using CAN Bus Serial Communications in Space Flight Applications White Paper, White Paper 1427152, Sekorm, last accessed on: [06/02/2026] (2019).
URL https://en.sekorm.com/doc/1427152.html

[15] I. P. S. J, Can standard data frame format, Medium, last accessed on: [06/02/2026] (Jun. 2024).
URL https://medium.com/@sjindhirapooja/can-standard-data-frame-format-846b8f9fc749

[16] Design World, Can bus cleared for space flight, Design World Online, last accessed on: [06/02/2026] (Jan. 2016).
URL https://www.designworldonline.com/can-bus-cleared-for-space-flight/

[17] M. Falch, Lin bus explained - a simple intro [2026], last accessed on: [06/02/2026] (January 2025).
URL https://www.csselectronics.com/pages/lin-bus-protocol-intro-basics

[18] PiEmbSysTech, Lin protocol, PiEmbSysTech - Embedded Systems VLSI Lab, last accessed on: [06/02/2026] (Jul. 2018).
URL https://piembsystech.com/lin-protocol/

[19] Beytoote Portal, Vehicle LIN protocol: One of the communication foundations of today's cars, https://www.beytoote.com/car-news/car-knowledge/vehicle-lin-protocol.html, last accessed on: [06/02/2026] (2023).

[20] Typhoon HIL, Lin bus protocol, Typhoon HIL Documentation, last accessed on: [06/02/2026] (2026).
URL https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/lin_bus_protocol.html

[21] Kvaser AB, Introduction to the lin bus, last accessed on: [06/02/2026] (2014).
URL https://kvaser.com/about-can/can-standards/linbus/

[22] RF Wireless World, CAN vs LIN bus: Key differences and applications, last accessed on: [06/02/2026] (2026).
URL https://www.rfwireless-world.com/terminology/can-vs-lin-bus

[23] National Instruments, FlexRay automotive communication bus overview, last accessed on: [06/02/2026] (July 2022).
URL https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/flexray-automotive-communication-bus-overview.html

[24] PiEmbSysTech, Most protocol, PiEmbSysTech – Embedded Systems VLSI Lab, last accessed on [06/02/2026] (Jul. 2018).
URL https://piembsystech.com/most-protocol/

[25] EasyEDA, Sign in to EasyEDA, EasyEDA (n.d.).
URL https://passport.easyeda.com/