**Mobile Application Enabled CAN Bus Communication for Smart Diagnostics in Autonomous Vehicles**

*Project Report*

*Submitted in the partial fulfillment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

*IN*

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**

By Batch-3

|  |  |
| --- | --- |
| **ID NUMBERS** | **NAMES** |
| 2200069004 | P. AEKANTH |
| 2200069018 | K. SAI SREENIVAS |
| 2200069021 | K. KOTESWARA RAO |
| 2200069033 | P .SOMA SEKHAR |

*under the supervision of*

# **Dr.D. Kalyan, (*Assistant professor*)**



Department of Electrical and Electronics Engineering

K L E F, Green Fields,

Vaddeswaram- 522502, Guntur(Dist), Andhra Pradesh, India.

**Declaration**

The Project Report entitled **Mobile Application Enabled CAN Bus Communication for Smart Diagnostics in Autonomous Vehicles** is a record of Bonafide work of 2200069004,2200069018, 2200069021,2200069033 submitted in partial fulfillment for the award of B.Tech in Electrical and Electronics Engineering to the K L University. The results embodied in this report have not been copied from any other departments/University/Institute.

**Id Numbers Names**

2200069004 P.Aekanth

2200069018 K.Sai Sreenivas

2200069021 K.Koteswara Rao

2200069033 P.Soma Sekhar

**CERTIFICATE**

This is to certify that the Project Report entitled **Mobile Application Enabled CAN Bus Communication for Smart Diagnostics in Autonomous Vehicles** is being submitted by 2200069004,2200069018,2200069021,2200069033 submitted in partial fulfillment for the award of B.Tech in Electrical and Electronics Engineering and Computer Science and Engineering to the K L University is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

## Signature of the Supervisor Signature of the PROJECT-

## Dr.D. Kalyan COORDINATOR

## Asst. Professor Dr.D. Kalyan

## Signature of the HOD Signature of the External Examiner

**Dr.A.Pandian**

**ACKNOWLEDGEMENTS**

I am greatly indebted to our K L Deemed to be a university that has provided a healthy environment to drive me to achieve my ambitions and goals. I would like to express my sincere thanks to my project in charge, for the guidance, support, and assistance they have provided in completing this project.

I express my sincere gratitude and thanks to my project guide **Dr .D. Kalyan** for his novel association of ideas, encouragement, appreciation, and intellectual zeal which motivated us to venture into this project successfully.

I express sincere gratitude to our Coordinator **Dr.D. Kalyan** for her leadership and constant motivation provided in the successful completion of our academic semester.

I record it as my privilege to deeply thank our pioneer **Dr. A. PANDIAN** HOD EEE for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere gratitude to our principal **Dr. T. K. RAMAKRISHNA RAO** for his administration towards our academic growth.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

.

**ABSTRACT**

As autonomous vehicles become more advanced, the need for a fast, reliable, and intelligent communication system within the vehicle becomes critical. This project, “Design and Development of a Robust CAN Bus System for Autonomous Vehicle Communication”, focuses on creating a specialized Controller Area Network (CAN) bus architecture that meets the unique demands of autonomous driving.

Autonomous vehicles rely on a network of sensors, controllers, and processors—such as LiDAR, cameras, radar, braking systems, and decision-making units—to function in real time. These systems must exchange data quickly and without error to ensure the vehicle responds safely and accurately to its environment. This project aims to design a CAN bus system that supports this high-speed, low-latency communication with improved fault tolerance and data prioritization.

We implemented enhancements to the traditional CAN protocol, including support for extended data frames, dynamic prioritization of critical messages, and better bandwidth management. To ensure fault tolerance, the system includes error detection using cyclic redundancy checks (CRC), automatic retransmissions, and isolation of malfunctioning nodes to prevent complete network failure.

The system was tested using a prototype setup that simulates an autonomous vehicle environment with multiple ECUs and sensor modules. Results show improved reliability, reduced communication delays, and effective recovery from faults.

By focusing on real-time performance and robust communication, this project contributes to the safe and efficient operation of autonomous vehicles. It highlights the importance of a dependable in-vehicle network and offers a scalable solution for future intelligent transportation systems.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **NAME OF THE CONTENT** | **PAGE NUMBER** |
| 1. | Introduction | 7 |
| 2. | Literature Survey | 12 |
| 3. | Theoretical Analysis | 14 |
| 4. | Experimental investigations | 20 |
| 5. | Experimental results | 23 |
| 6. | Conclusion | 30 |
| 7. | Future Scope of the Project | 31 |
| 8 | References | 32 |

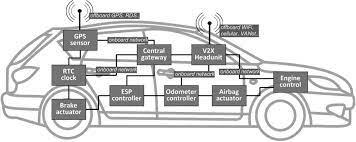
**INTRODUCTION**

**1.1 Introduction**

The rapidly evolving domain of autonomous vehicles, seamless and reliable communication between various electronic control units (ECUs) is critical to ensure safe, efficient, and intelligent vehicle operation. As autonomous systems become increasingly complex, with multiple sensors, actuators, and processing units working in tandem, the need for a robust communication protocol becomes paramount. The Controller Area Network (CAN) bus, originally developed for automotive applications, has emerged as a widely adopted solution due to its efficiency, simplicity, and real-time capabilities.

This project titled *“Design and Development of a Robust CAN Bus System for Autonomous Vehicle Communication”*seeks to enhance the performance and reliability of CAN bus systems tailored for autonomous vehicle environments. The goal is to facilitate real-time data exchange among subsystems such as perception units, control units, braking systems, steering mechanisms, and sensor arrays—thereby supporting the autonomous vehicle’s ability to perceive, decide, and act with precision.

To meet the stringent requirements of autonomous navigation, this project emphasizes fault-tolerant communication, high-speed data transfer, and low-latency message delivery. The proposed system incorporates protocol-level improvements and hardware optimizations to address common challenges like data collisions, message prioritization, and error detection. Additionally, the implementation is designed to be scalable and adaptable, ensuring compatibility with future advancements in vehicle automation technologies.



**Fig 1 CAN Communication in Vehicle**

**1.1 Motivation**

CAN bus, developed by Bosch in the 1980s, is a message-based protocol that allows microcontrollers and devices to communicate without the need for a host computer. It provides features such as error detection, arbitration, and prioritization, making it suitable for safety-critical automotive applications. Despite its widespread use, the traditional CAN bus faces several challenges when applied to autonomous vehicles, including bandwidth limitations, message latency, and vulnerability to single-point failures.

As autonomous systems demand real-time performance and fault-tolerant communication, enhancing the reliability and speed of CAN-based networks becomes essential. This motivates the need for a robust CAN bus system that can support complex data flows between multiple ECUs in a highly dynamic environment.

**1.2 Importance of CAN bus**

**A diagram of a computer network

Description automatically generated**

**Fig 2 With & Without CAN Communication In Vehicle**

his image illustrates the difference between a system with and without a **CAN bus** (Controller Area Network) for communication between Electronic Control Units (ECUs) and devices.

* **Without CAN:**
  + The ECUs and devices are connected directly to each other, which results in a **complex wiring system**.
  + Each ECU has a dedicated line to every other device or control unit. As the number of devices increases, the number of connections grows exponentially, making the system highly **complex** and **prone to errors**.
  + This setup requires a significant amount of wiring, which increases the **weight**, **cost**, and **complexity** of the system.
  + Additionally, as the number of devices or ECUs increases, managing communication becomes **inefficient**, and maintaining such a system is **difficult**.
* **With CAN :**
  + The system uses a **single communication bus**, simplifying the entire architecture by allowing ECUs and devices to share a common communication line (CAN\_H and CAN\_L lines).
  + This reduces the number of connections, making the system **more scalable** as more devices can be added without increasing the complexity of the wiring.
  + **Reduced wiring** also leads to a decrease in **weight** and **cost** of the system, which is crucial in automotive applications where minimizing these factors is important.
  + The **CAN protocol** ensures reliable communication with error detection and fault tolerance, making the system more robust and easier to maintain.

**Importance of the CAN Bus in Autonomous Vehicles:**

1. **Reduced Complexity and Cost:**
   * In autonomous vehicles, numerous sensors, ECUs, and devices need to communicate with each other in real-time. The **CAN bus** simplifies the wiring system by providing a centralized communication line for all devices, reducing the need for complex wiring harnesses and significantly cutting down costs.
2. **Scalability:**
   * As autonomous vehicle technology continues to evolve, the number of ECUs and devices will increase. The **CAN bus** allows for easy integration of new devices without the need for a complete system redesign, making it highly scalable.
3. **Reliability and Robustness:**
   * The CAN protocol is designed for **high-reliability** communication, ensuring that all messages are delivered correctly. Built-in error detection and fault tolerance mechanisms (e.g., CRC checks, retransmissions) make it suitable for safety-critical applications in EV.
4. **Real-Time Communication:**
   * Autonomous vehicles rely on real-time data exchange between ECUs to make split-second decisions, such as braking, steering, and adjusting speed. The CAN bus system offers **low-latency** communication, which is essential for autonomous driving where timing is critical.
5. **Error Handling and Fault Tolerance:**
   * The image demonstrates that, with CAN, the system can handle **faults** much more gracefully. If one node goes down, the other nodes can continue communicating, ensuring that the vehicle can still function safely in case of a failure in one of the ECUs.
6. **Better Data Integrity:**
   * With direct connections, the risk of data corruption or loss increases. The CAN bus, however, ensures that the data exchanged is accurate and properly formatted, with built-in mechanisms to detect and correct errors.

**1.3 Problem Statement**

Standard CAN protocols, though reliable for conventional automotive systems, may fall short in handling the high volume of data and stringent real-time requirements of autonomous vehicle operations. Issues such as delayed sensor data, packet loss, or network congestion can critically impact the vehicle's decision-making and safety. Moreover, lack of built-in redundancy and limited fault tolerance poses risks in mission-critical scenarios.

**1.4 Objectives**

This project aims to design and develop a robust CAN bus system for autonomous vehicles with the following objectives:

* To ensure real-time communication between ECUs with minimal latency.
* To implement fault tolerance mechanisms for improved system reliability.
* To optimize CAN protocol performance through message prioritization and bus load management.
* To validate the proposed system under simulated autonomous driving scenarios.

**1.5 Scope of the Project**

The project focuses on the design, implementation, and evaluation of a CAN-based communication architecture for autonomous vehicle prototypes or lab-scale simulation environments. It encompasses:

* Hardware-level integration of microcontrollers, sensors, and CAN transceivers.
* Real-time data handling from multiple ECUs.
* Simulation or physical testing of the system under fault injection and performance load conditions.

**LITERATURE SURVEY**

As the automotive industry evolves toward autonomy and intelligent transportation, the demand for efficient, fault-tolerant, and high-speed in-vehicle communication systems has grown significantly. This section explores prior research and developments related to Controller Area Network (CAN) bus systems, particularly in the context of autonomous vehicle communication.

**2.1 CAN Bus in Automotive Applications**

The CAN protocol was introduced by Bosch in 1986 to reduce wiring complexity and enable efficient communication between Electronic Control Units (ECUs) in vehicles. It offers key features such as message arbitration, priority-based messaging, and error detection mechanisms. According to Robert Bosch GmbH (1991), the CAN bus operates effectively in conventional vehicles by providing reliable and deterministic communication, particularly in powertrain, body control, and infotainment systems.

However, the data rate of the classical CAN protocol is limited to 1 Mbps, which can be insufficient for modern automotive systems involving high-frequency sensor data and real-time decision-making.

**2.2 Limitations of Classical CAN in Autonomous Systems**

Autonomous vehicles require high-speed communication and increased data throughput for subsystems like LiDAR, radar, vision processing units, and real-time control. Researchers such as Singh et al. (2019) noted that classical CAN networks experience bottlenecks due to limited bandwidth, especially when integrating multiple sensors and high-priority control messages.

Additionally, fault resilience is critical in autonomous vehicles. Studies by Zhang and Li (2020) show that a single fault in the CAN network, such as a stuck-at-dominant bit or bus-off error, can halt communication across the system, leading to dangerous failures unless proper redundancy and error-handling techniques are in place.

**2.3 CAN FD and Advanced Protocols**

To address bandwidth limitations, the CAN Flexible Data-rate (CAN FD) protocol was introduced, allowing higher data rates (up to 5-8 Mbps) and payloads up to 64 bytes per frame. Research by Müller et al. (2018) demonstrated the effectiveness of CAN FD in reducing latency and increasing throughput in automotive networks. However, adoption in autonomous platforms remains limited due to compatibility issues and a lack of tailored fault-tolerant strategies.

Some researchers have explored hybrid network architectures combining CAN with higher-speed protocols like Ethernet or Time-Triggered Protocols (TTP), aiming to create hierarchical communication frameworks (e.g., Roh et al., 2021). While these offer improved performance, they also introduce complexity and cost.

**2.4 Fault-Tolerant Mechanisms in CAN Networks**

Fault tolerance in CAN systems has been an area of active research. Techniques such as dual-channel CAN, watchdog timers, and heartbeat monitoring are commonly implemented. A study by Patel and Joshi (2017) emphasized the importance of error recovery strategies and system redundancy in maintaining communication integrity in safety-critical environments. Fault injection testing methods have also been employed to evaluate the robustness of CAN nodes under real-world fault scenarios.

**2.5 Research Gap and Motivation**

While significant work has been done on enhancing CAN bus communication, there remains a need for application-specific designs that cater to the unique requirements of autonomous vehicles. Existing literature largely focuses on standard automotive use-cases, with limited exploration into:

* Integration of real-time sensor data from autonomous subsystems.
* Protocol optimizations specific to high-load, safety-critical environments.
* Practical implementation of fault-tolerant architectures in autonomous ECUs.

This project aims to bridge these gaps by developing a robust and optimized CAN communication system specifically for autonomous vehicle platforms, focusing on real-time performance, fault recovery, and message prioritization.

**THEORETICAL ANALYSIS**

The development of a robust Controller Area Network (CAN) bus system for autonomous vehicle communication involves a deep understanding of the CAN protocol, communication theory, error-handling mechanisms, and fault tolerance. This section presents the essential theoretical concepts that form the foundation of the proposed system.

**3.1 Controller Area Network (CAN) Protocol Overview**

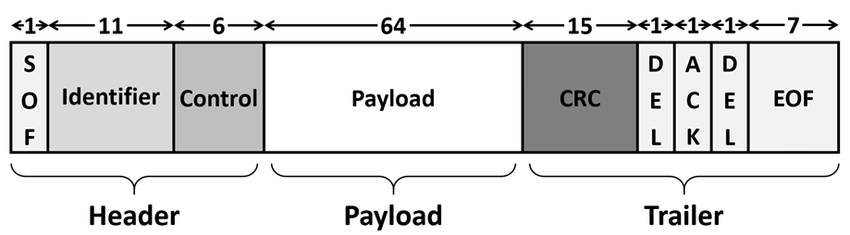
The CAN protocol is a multi-master, message-based communication system designed for microcontrollers and devices to communicate with each other without a host computer. It uses differential signaling across two wires: **CAN High (CAN\_H)** and **CAN Low (CAN\_L)**. The key features include:

* **Message Arbitration:** Based on message ID. Lower ID values have higher priority.
* **Non-Destructive Arbitration:** If multiple nodes transmit at once, the highest-priority message wins, and other nodes wait.
* **Error Detection and Correction:** Utilizes techniques such as CRC (Cyclic Redundancy Check), bit stuffing, and acknowledgment (ACK) slots.
* **Bit Rate:** Classical CAN supports up to 1 Mbps; CAN FD supports up to 8 Mbps.

**3.2 CAN Frame Structure**

There are two main types of CAN frames:

* Standard Frame (11-bit identifier)
* Extended Frame (29-bit identifier)



**Fig 3 CAN Frame Structure**

Each frame consists of:

* Start of Frame (SOF)
* Identifier
* Control Field
* Data Field (0 to 8 bytes in Classical CAN; up to 64 bytes in CAN FD)
* CRC Field
* ACK Field
* End of Frame (EOF)

Understanding this structure is essential for optimizing message flow and prioritizing critical data in autonomous vehicle systems.

**3.3 Real-Time Communication and Latency Considerations**

Autonomous vehicles require deterministic, low-latency communication. The total communication delay (T\_total) can be modeled as:

Ttotal=Ttransmission+Tpropagation+Tprocessing+TqueuingTtotal​=Ttransmission​+Tpropagation​+Tprocessing​+Tqueuing​

Where:

* T\_transmission = size of data / bus speed
* T\_propagation = time signal takes to travel along the medium
* T\_processing = internal processing delay in nodes
* T\_queuing = delay due to network congestion

Reducing these components through protocol optimization and network scheduling is critical for real-time responsiveness.

**3.4 Fault Detection and Recovery Mechanisms**

Robust CAN systems rely on several built-in fault handling features:

* Error Active and Error Passive States: Nodes monitor error counters; if thresholds are crossed, they reduce their activity.
* Bus-Off Recovery: After severe faults, a node may be temporarily removed from the bus and reinitialized.
* Bit Error, Form Error, CRC Error, and Acknowledgment Error Detection

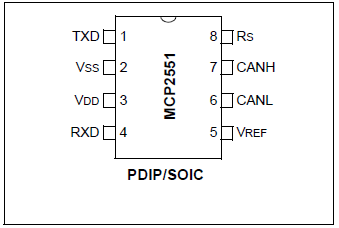
For enhanced fault tolerance, external watchdog circuits, redundant bus lines, and diagnostic routines can be integrated.

**3.5 Message Prioritization and Scheduling**

In autonomous systems, different messages have different urgency levels (e.g., braking commands vs. infotainment). Efficient use of the arbitration mechanism is essential to prioritize high-criticality messages.

* Static Scheduling: Assign fixed priorities to ECUs.
* Dynamic Scheduling: Adjust message priorities based on system state or sensor inputs.
* Rate-Monotonic Scheduling (RMS): Prioritizes tasks with shorter periods.
* Deadline-Monotonic Scheduling (DMS): Uses deadlines to determine priorities.

**3.6 CAN Transceiver**



**Fig 4 CAN Transceiver**

The image shows the pin configuration of the **MCP2551**, a commonly used **CAN transceiver** integrated circuit. This transceiver is responsible for connecting the **CAN controller** (typically inside a microcontroller) to the **physical CAN bus lines (CANH and CANL)**. It translates the digital signals from the controller into differential signals for the bus and vice versa. Below is a detailed explanation of each pin in the MCP2551:

1. **TXD (Transmit Data Input)**
   * This pin receives the CAN data from the CAN controller (usually from the microcontroller’s CAN\_TX pin).
   * Logic-low input here generates a **dominant** state on the bus.
2. **VSS (Ground)**
   * Ground reference for the device.
3. **VDD (Supply Voltage)**
   * Power supply input, typically 5V, required for the internal operation of the transceiver.
4. **RXD (Receive Data Output)**
   * This pin outputs the data received from the CAN bus back to the controller (connected to the microcontroller’s CAN\_RX pin).
   * It reflects the logic level present on the CAN bus.
5. **VREF (Reference Voltage Output)**
   * Provides a reference voltage of 2.5V used in some applications for biasing or monitoring.
   * Optional and often left unconnected in basic setups.
6. **CANL (CAN Low Line)**
   * One of the differential lines connected to the CAN bus.
   * Works in tandem with CANH to create the differential signal needed for robust communication.
7. **CANH (CAN High Line)**
   * The other differential line on the CAN bus.
   * Together with CANL, transmits differential signals: for a dominant bit, CANH > CANL; for recessive, both are approximately 2.5V.
8. **RS (Slope Control/Mode Selection)**
   * This pin controls the slew rate of the CAN signals.
   * Connecting a resistor to ground adjusts the rise and fall times of the output signal, helping reduce EMI (Electromagnetic Interference).

**3.7 Communication Topology**

The CAN bus follows a multi-master linear bus topology. Each node is connected via a two-wire differential line, which helps in noise immunity. Proper termination resistors (usually 120Ω at both ends) are required to avoid signal reflections.

For autonomous applications, star or hybrid topologies can be introduced using repeaters or gateways, though at the cost of simplicity.

A diagram of a bus diagram

Description automatically generated

**Fig 5 CAN Communication Topology**

The CAN bus topology shown in the diagram represents a robust and efficient communication architecture used in applications where real-time and fault-tolerant data exchange is critical, such as in automotive systems, industrial automation, and autonomous vehicles. Each node in the network—ranging from Node 1 to Node n—includes a CAN transceiver that connects to a shared bus consisting of two differential lines: CAN High and CAN Low. These lines carry complementary signals, which ensures high immunity to external noise and electromagnetic interference. This differential signaling is a key advantage of CAN, as it allows reliable data transmission even over long distances or in harsh environments.

To maintain signal integrity, the bus is terminated at both ends with 120-ohm resistors. These resistors match the characteristic impedance of the cable and eliminate signal reflections that can corrupt data, especially at high communication speeds (up to 1 Mbps or more). The topology is linear, meaning nodes are connected along a single backbone cable. This reduces wiring complexity compared to point-to-point connections and simplifies installation and troubleshooting.

CAN is a multi-master protocol, which means any node can initiate communication when the bus is free. A priority-based arbitration mechanism ensures that high-priority messages (such as brake system commands) are transmitted first without collision. This deterministic behavior is vital in safety-critical systems. Furthermore, the modularity of this topology allows for easy system expansion, making it scalable for future upgrades or additional components.

Overall, this CAN bus structure provides a highly reliable, scalable, and efficient communication backbone for embedded systems requiring real-time data exchange, making it foundational in modern automotive networks, robotic platforms, and smart industrial setups.

**3.8 Comparison with Alternative Protocols**

| **Feature** | **Classical CAN** | **CAN FD** | **FlexRay** | **Automotive Ethernet** |
| --- | --- | --- | --- | --- |
| Data Rate | Up to 1 Mbps | Up to 8 Mbps | 10 Mbps | 100 Mbps – 1 Gbps |
| Max Payload | 8 bytes | 64 bytes | 254 bytes | 1500 bytes |
| Real-Time Support | Yes | Yes | Strong | Limited (w/ TSN) |
| Cost & Simplicity | Low | Medium | High | High |
| Fault Tolerance | Moderate | Moderate | High | Moderate |

**EXPERIMENTAL INVESTIGATIONS**

This section outlines the experimental setup and procedures used to evaluate the performance and robustness of the CAN bus system developed for autonomous vehicle communication. The focus is on using **CAN transceivers** and **STM32 microcontrollers** to implement the communication system, measure its performance, and assess fault tolerance and reliability in real-world scenarios.

**4.1 System Setup and Components**

The experimental system consists of the following primary components:

* **STM32 Microcontrollers (MCUs):** The STM32 family of microcontrollers by STMicroelectronics was selected for this experiment due to its high processing capabilities, integrated peripherals (including CAN controllers), and low power consumption. The STM32F4 series was chosen for its high-speed performance and sufficient I/O pins for connecting the CAN transceivers and sensors.
* **CAN Transceivers:** The **MCP2551** CAN transceiver, a standard transceiver for automotive applications, was used for interfacing the STM32 microcontroller with the CAN bus. This transceiver converts the logical levels from the microcontroller to the differential voltage levels needed for CAN communication.
* **CAN Bus Network:** A physical network comprising a twisted pair of wires for CAN\_H and CAN\_L, along with appropriate termination resistors (120Ω) at both ends to prevent signal reflections.
* **Test Environment:** The test environment simulates an autonomous vehicle communication setup, with multiple ECUs (representing subsystems such as braking, steering, sensors, etc.) communicating over the CAN bus. The system is tested for various conditions, including normal operation, high bus load, and fault conditions (e.g., CAN errors, node failure).

**4.2 System Architecture**

The system consists of several key modules:

1. **Master Node (STM32)**: This node acts as the central controller, managing the communication and monitoring the bus traffic. It sends and receives control messages, updates sensor data, and commands to various peripheral nodes.
2. **Slave Nodes (STM32)**: These nodes represent various subsystems within the autonomous vehicle, such as sensor units (LiDAR, radar), actuator control (steering, braking), and monitoring systems.
3. **CAN Transceivers (MCP2551)**: These devices interface the STM32 microcontrollers with the CAN bus, ensuring data is transmitted and received correctly.

The communication between nodes is based on the CAN protocol, with the STM32s implementing message priority handling, error detection, and fault tolerance measures.

**4.3 Circuit Diagram**

****

**Fig 6 Circuit Diagram of CAN Communication**

This diagram illustrates a basic setup for Controller Area Network (CAN) bus communication between two STM32 Nucleo-F446 development boards using MCP2551 CAN transceivers. Let's break down the components and connections:

**Components:**

* **NUCLEO-F446:** These are microcontroller development boards. In this context, they act as the CAN bus nodes, responsible for processing and generating the data to be transmitted and received over the CAN network. The diagram shows that each Nucleo board has connections to the MCP2551 transceiver. Specifically:
  + **PB8:** This pin on the Nucleo is connected to the **CRX** (CAN Receive) pin of the MCP2551. This is where the Nucleo receives data from the CAN bus.
  + **PB9:** This pin on the Nucleo is connected to the **CTX** (CAN Transmit) pin of the MCP2551. This is how the Nucleo sends data onto the CAN bus.
  + **5V:** This pin provides the power supply to the MCP2551 transceiver.
  + **GND:** This is the ground connection for the Nucleo and the transceiver.
* **MCP2551 Transceiver:** This is a crucial component that acts as an interface between the microcontroller (Nucleo-F446) and the physical CAN bus. It performs the following key functions:
  + **Differential Transceiver:** The CAN bus uses a differential signaling scheme (CANH and CANL) for noise immunity. The MCP2551 takes the single-ended transmit signal (CTX) from the microcontroller and converts it into the differential signals on CANH and CANL. Conversely, it takes the differential signals from the bus (CANH and CANL) and converts them into a single-ended receive signal (CRX) for the microcontroller.
  + **Physical Layer Interface:** It provides the electrical interface to the CAN bus, including handling voltage levels and current driving capabilities.
  + **Protection:** It often includes some level of protection against overvoltage, transients, and other electrical issues on the bus.
  + **Connections:**
    - **CRX:** Connected to the microcontroller's receive pin (PB8).
    - **CTX:** Connected to the microcontroller's transmit pin (PB9).
    - **CANH:** One of the differential CAN bus lines.
    - **CANL:** The other differential CAN bus line.
    - **VCC:** Power supply for the transceiver (connected to the Nucleo's 5V).
    - **GND:** Ground connection (connected to the Nucleo's GND).
    - **Vref (Not explicitly connected in this diagram):** This pin can be used for slope control to optimize the transceiver for different bus speeds and cable lengths. It's often left unconnected or tied to ground.
    - **RS (Not explicitly connected in this diagram):** This pin is used to select the operating mode of the transceiver (e.g., high-speed, slope control). It's often tied to ground for high-speed operation.
* **120Ω Resistor:** You see a 120Ω resistor connected across the CANH and CANL lines at each end of the bus. These are **termination resistors**. They are essential for proper CAN bus operation because they:
  + **Prevent Signal Reflections:** At high frequencies, signals can reflect at the ends of unterminated cables, causing distortion and communication errors. The termination resistors absorb the energy of these signals, preventing reflections.
  + **Define Bus Impedance:** They ensure the bus has a defined characteristic impedance, which is typically around 120Ω for standard CAN.
* **Red Lines:** These lines represent the electrical connections between the different components.

**How the Communication Works:**

1. **Transmission:**
   * When one of the Nucleo-F446 boards wants to send data, its microcontroller transmits the data serially through its PB9 pin (configured as a transmit pin).
   * This signal reaches the CTX pin of its connected MCP2551 transceiver.
   * The MCP2551 converts this single-ended digital signal into a differential signal on the CANH and CANL lines.
   * These differential signals propagate along the CAN bus.
2. **Reception:**
   * The differential signals on the CANH and CANL lines reach the MCP2551 transceiver connected to the other Nucleo-F446 board.
   * The MCP2551 converts these differential signals back into a single-ended digital signal on its CRX pin.
   * This signal is then received by the microcontroller on the second Nucleo-F446 board through its PB8 pin (configured as a receive pin).
   * The microcontroller then processes the received data.

**4.4 Experimental Procedure**

To evaluate the performance of the system, the following experimental procedures were followed:

1. **Message Transmission and Reception Testing:**
   * Each STM32 microcontroller was programmed to transmit test messages at a fixed interval over the CAN bus.
   * Messages consisted of sensor data, status reports, and control commands.
   * The CAN transceivers (MCP2551) were used to send and receive messages between the nodes.
   * The communication was monitored for message integrity, latency, and data collisions.
2. **Bus Load and Latency Measurements:**
   * To simulate varying communication loads, additional slave nodes were added to the network, increasing the bus traffic.
   * The latency of message transmission from the master node to the slave nodes was measured using time-stamping techniques on the microcontrollers.
   * The experiment was conducted under different network conditions (e.g., low load, medium load, and high load) to evaluate the system's performance and responsiveness.
3. **Fault Injection Testing:**
   * Fault injection was performed to test the fault tolerance of the system. Faults such as bit errors, bus-off states, and communication drops were simulated.
   * Error detection and recovery mechanisms were implemented in the STM32 firmware to handle these faults.
4. **Real-Time Communication Evaluation:**
   * The real-time performance of the CAN bus was evaluated by monitoring the response times for control messages (e.g., braking commands or steering inputs) under different traffic loads.
   * The system was tested for both high-priority messages (time-sensitive data, like braking or emergency stop signals) and low-priority messages (non-critical data, such as infotainment or status updates).

**EXPERIMENTAL RESULTS**

**Block Diagram**

**A diagram of a network

Description automatically generated**

**Fig 7 Block Diagram**

The CAN bus system shown in the diagram is designed to facilitate **efficient and fault-tolerant communication** between multiple electronic modules (referred to as **CAN nodes**) over a **shared communication medium**. In this case, the bus consists of two twisted-pair lines: **CAN High (CAN H)** and **CAN Low (CAN L)**. These two lines work together to transmit data using **differential signaling**, which significantly improves noise immunity — a crucial factor in environments like vehicles, factories, or smart grids where electromagnetic interference is common.

Each CAN node (Node 1, Node 2, and Node 3 in the diagram) is equipped with a **CAN controller and transceiver**, allowing it to both transmit and receive data. The nodes are connected in **parallel** to the main bus, meaning they "listen" and "talk" over the same pair of wires. The network is **multi-master**, allowing any node to initiate communication if the bus is idle, which supports decentralized system architecture.

An important aspect not explicitly shown in the diagram but necessary for real-world implementation is the inclusion of **termination resistors** (typically 120 ohms) at both physical ends of the bus. These resistors prevent signal reflections that can cause data corruption. The network also supports **message prioritization** using identifier-based arbitration, ensuring that more critical data (like braking signals in a vehicle) gets transmitted first.

This topology is ideal for applications like **autonomous vehicles**, where modules such as the braking system, engine control, sensors, LiDAR, and GPS units must constantly exchange information with minimal latency and maximum reliability. The CAN bus not only reduces the amount of wiring required compared to point-to-point communication but also enhances system robustness and simplifies diagnostics and maintenance.

A diagram of a computer system

Description automatically generated

**Fig 8 Single ECU in CAN Communication**

This image represents the internal architecture of an Electronic Control Unit (ECU) within a CAN (Controller Area Network) communication system, commonly used in automotive and industrial applications. At its core, the ECU comprises a **Microcontroller Unit (MCU)** that hosts a **CAN controller**, which is either embedded within the MCU or implemented as a separate component. The CAN controller handles the data-link layer functions—such as framing, arbitration, error detection, and message filtering. It formats the outgoing messages and interprets incoming ones according to the CAN protocol.

To physically transmit and receive data over the CAN bus, the CAN controller interfaces with a **CAN transceiver**. The transceiver acts as a physical layer device, converting the logic-level signals from the controller into the differential voltages needed to drive the **CAN High (CAN\_H)** and **CAN Low (CAN\_L)** lines. It also converts differential signals from the bus back into logic-level signals for the controller to process.

This differential signaling technique, using two wires (CAN\_H and CAN\_L), is essential in ensuring noise immunity and maintaining communication integrity even in electrically noisy environments like vehicles or industrial machines. When a dominant bit is transmitted, CAN\_H goes higher and CAN\_L goes lower in voltage, creating a clear voltage difference. When a recessive bit is transmitted, both lines remain at a similar voltage, indicating an idle state.

The arrows labeled “transmit” and “receive” show the flow of data between the CAN controller and the transceiver, emphasizing the bidirectional nature of CAN communication. This setup enables the ECU to function both as a transmitter and a receiver, allowing it to participate actively in the CAN network. This modular structure ensures scalability, reliability, and robust communication between various control units, making it ideal for real-time control in safety-critical systems like engine management, braking systems, and autonomous navigation.

**Flow Chart**

**A diagram of a software system

Description automatically generated**

**Fig 9 Flow Chart**

This flowchart represents the communication process between two ECUs (Electronic Control Units) in a CAN (Controller Area Network) bus system, specifically for a system utilizing STM32 microcontrollers. The process begins with the initialization of the STM32 ECUs.

In the first step, ECU 1 is responsible for acquiring sensor data, which could involve reading inputs from various sensors (e.g., temperature, pressure, or speed sensors). Once the data is collected, ECU 1 prepares a frame that includes the sensor data in a format that can be transmitted over the CAN network. This frame is then sent by ECU 1 via the CAN bus to ECU 2.

ECU 2, upon receiving the CAN message from ECU 1, extracts the relevant data from the message. This could involve parsing the frame to interpret the sensor data for further use. Once the data is extracted, ECU 2 uses the data to control actuators, which could be components such as motors, valves, or other devices that respond to the sensor input.

After the actuator control is performed, ECU 2 sends an acknowledgment or status message back to ECU 1 to confirm the receipt and processing of the transmitted data. This status message could indicate whether the data was processed correctly or if there was an error.

The system then proceeds with either a delay (depending on timing or system needs) or moves on to the next iteration, where the process is repeated. This loop ensures continuous communication and control between the two ECUs within the CAN bus system, enabling real-time data exchange and control in systems such as vehicles, industrial machinery, or robotics.

The flowchart emphasizes a typical communication pattern in embedded systems, where two or more ECUs are involved in collecting, processing, and acting upon data in a structured and efficient manner.

**Hardware Setup:**

The hardware setup for an autonomous vehicle communication system relies heavily on a robust Controller Area Network (CAN) bus system, which enables seamless real-time communication between various Electronic Control Units (ECUs). At the core of this system, an STM32 microcontroller is used as the central processing unit (CPU), offering powerful processing capabilities and support for multiple communication protocols, including CAN. The STM32 microcontroller is interfaced with CAN transceivers, such as the MCP2551 or TJA1050, which allow communication over the CAN bus by converting the digital signals from the microcontroller into appropriate voltage levels for the bus and vice versa. These transceivers are crucial for ensuring reliable, noise-immune communication, especially in the automotive environment. In addition to the microcontroller and transceivers, the system integrates essential components like power supplies, voltage regulators, and fault-tolerant mechanisms to maintain stable operation even in harsh conditions. The setup ensures that data flows between subsystems, such as sensors, actuators, and control units, at high speed and with low latency, providing the necessary communication backbone for AV.

A computer with wires and wires on a table

Description automatically generated

**Fig 10 Hardware setup**

**A computer with wires connected to it

Description automatically generated**

**Fig 11 Hardware setup**

A device with wires and notes on a table

AI-generated content may be incorrect.

This hardware setup is designed to demonstrate the functionality of a Controller Area Network (CAN) bus system for real-time communication between electronic control units (ECUs) in an embedded or automotive environment. The system uses STM32 microcontrollers as the main processors, where one acts as the transmitter and the other as the receiver. The STM32 boards are connected to CAN transceivers via breadboards, enabling communication over the CAN bus. The transceivers play a crucial role in ensuring the proper conversion of signals from the microcontroller to the CAN network and vice versa.

In the first part of the setup, the STM32 development board labeled "ECU TRANSMITTER" sends data through the CAN transceiver, and the board labeled "ECU RECEIVER" receives the transmitted data using a second CAN transceiver. This connection is powered via USB from a laptop, which also monitors the communication status. In the second image, the setup is similar but with the addition of a laptop that displays the CAN bus communication status, providing valuable feedback during testing. The system is powered by a 5V battery, ensuring that the setup is portable and suitable for mobile applications, such as automotive systems.

This arrangement highlights the importance of having reliable communication protocols like CAN for real-time data exchange between ECUs. CAN buses are often used in vehicles for low-latency, high-reliability communication, and this hardware setup serves as a demonstration of how data flows between components, showcasing the basic principles of ECU communication in autonomous or automotive environments.

**CONCLUSION**

The successful design and development of a robust CAN Bus system for autonomous vehicle communication represents a critical step towards enhancing the performance and reliability of autonomous vehicle systems. The project has focused on addressing the key challenges faced by communication networks in autonomous vehicles, including real-time data exchange, fault tolerance, and optimization for high-speed, low-latency communication, which are essential for the seamless operation of autonomous vehicles in dynamic environments.

Through the integration of various advanced techniques, the proposed CAN Bus system ensures efficient communication between the Electronic Control Units (ECUs) in autonomous vehicles. The design emphasizes fault tolerance, ensuring the system remains operational even in the event of network or device failure, which is crucial for the safety and reliability of autonomous vehicle systems. The implementation of error detection and correction mechanisms, such as cyclic redundancy checks (CRC), has further enhanced the robustness of the communication system, reducing the likelihood of data corruption and improving overall system integrity.

The optimization techniques incorporated into the system, including advanced arbitration protocols and message prioritization, have ensured minimal latency in data transmission. This is vital for real-time decision-making in autonomous vehicles, where delays in communication can result in unsafe situations. The system also demonstrated scalability, allowing it to support a growing number of ECUs and sensors without compromising performance, which is essential as autonomous vehicle technologies continue to evolve and integrate more sophisticated sensors and systems.

Furthermore, the project has contributed to the understanding of CAN Bus communication protocols and their practical implementation in autonomous vehicles. By focusing on high-speed communication with low-latency requirements, the system is well-suited for applications in autonomous driving, where rapid responses to environmental stimuli are crucial for navigation, collision avoidance, and overall safety.

In conclusion, the robust CAN Bus system designed in this project provides a solid foundation for the development of communication networks in autonomous vehicles. The successful implementation of fault tolerance, real-time data exchange, and optimization for high-speed communication positions the system as a key component in the future of autonomous vehicle technologies. As autonomous vehicles continue to evolve, the need for efficient, fault-tolerant, and scalable communication systems like the one developed in this project will become increasingly important in ensuring the safety, reliability, and performance of these advanced transportation systems.

**FUTURE SCOPE OF THE PROJECT**

The future of the CAN Bus system for autonomous vehicle communication has several key areas for further development:

1. **Integration with New Communication Technologies**: The system can be enhanced by integrating with faster communication protocols like Ethernet or FlexRay, and exploring 5G for Vehicle-to-Everything (V2X) communication, enabling better interaction with other vehicles and infrastructure.
2. **Improved Fault Tolerance and Redundancy**: Adding self-healing capabilities and redundant communication paths could improve system reliability, ensuring it keeps working even if part of the system fails.
3. **Enhanced Data Security**: Introducing encryption and security systems like Intrusion Detection Systems (IDS) would protect the vehicle's communication from cyber threats and unauthorized access.
4. **Scalability for More Complex Systems**: As autonomous vehicles use more sensors and systems, the CAN Bus network must be able to handle more devices, and future work could explore multi-vehicle communication and improved network traffic management.
5. **Real-Time Diagnostics and Maintenance**: The system could be enhanced with tools for monitoring and diagnosing problems, helping with predictive maintenance and ensuring the vehicle remains in optimal condition.
6. **AI-Based Optimization**: Machine learning could be used to dynamically adjust the communication protocol based on traffic, improving efficiency and response times during critical events.
7. **Testing and Simulation**: Creating better testing environments, such as simulations, would help ensure the system works well in real-world conditions and under various failure scenarios.
8. **Industry Standardization**: Working with industry standards and regulations could help make the system widely adopted by vehicle manufacturers and ensure it meets safety and communication standards.

**References**

1 **An Efficient and Secure Wireless Controller Area Network for Autonomous Vehicle**  
This paper presents a wireless CAN Bus system tailored for autonomous vehicles, focusing on enhancing the traditional wired CAN with wireless features. It introduces a unique complementary code keying (CCK) modulation equation and strategies to mitigate jamming signals using dual channels. Performance evaluations through OPNET analysis demonstrate the system's robustness in maintaining data integrity and reliability.

2 **State-of-the-Art Survey on In-Vehicle Network Communication CAN-Bus Security and Vulnerabilities**  
This survey provides an in-depth analysis of the security challenges associated with the CAN Bus protocol in in-vehicle networks. It discusses various vulnerabilities and reviews existing solutions aimed at enhancing the security of CAN Bus communications in automotive systems.

3 **Anomaly Detection of CAN Bus Messages Using a Deep Neural Network for Autonomous Vehicles**  
This research introduces a deep neural network-based approach for detecting anomalies in CAN Bus messages within autonomous vehicles. The method aims to identify and mitigate potential security threats by analyzing the communication patterns on the CAN Bus.

4 **Engineering Autonomous Mobility: Wired and Wireless CAN Bus Design Principles**  
This article explores the design principles for both wired and wireless CAN Bus systems in the context of autonomous mobility. It examines the challenges and solutions in integrating CAN Bus communication into autonomous vehicles, highlighting the importance of robust and efficient network design.

5 **Review of Electrical and Electronic Architectures for Autonomous Vehicles: Topologies, Networking and Simulators**  
This comprehensive review discusses various electrical and electronic architectures employed in autonomous vehicles, with a focus on networking topologies and simulation tools. It provides insights into the integration of CAN Bus systems within the broader context of autonomous vehicle infrastructure.