Project Report

On

Implementation of CAN In Vehicle Safety

Submitted

For the award of the Degree of

PG-Diploma in Embedded Systems and Design (PG-DESD)

Sunbeam Institute of Information Technology,Hinjewadi

Guided By:

Mr.Vrishabh Patil

Submitted By:

240844230057	Phansalkar Omkar Atul
240844230061	Pranita Dilip Songire
240844230018	Chetna Sahu
240844230037	Karishma Kishor Ghadge

"Sunbeam IT Park", Second Floor, Phase 2 of Rajiv Gandhi Infotech Park,Hinjawadi, Pune 411057

ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mr. Devendra Dhande (Course Coordinator, SIIT, Pune) .

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

SIIT Pune

240844230057	Phansalkar Omkar Atul
240844230061	Pranita Dilip Songire
240844230018	Chetna Sahu
240844230037	Karishma Kishor Ghadge

CERTIFICATE

This is to certify that the project work under the title '**Impementation of CAN in Vehicle Safety'** is done by Phansalkar Omkar Atul, Pranita Dilip Songire, Chetna Sahu,Karishma Kishore Ghadge in partial fulfillment of the requirement for award of Diploma in Embedded System Design.

Project Guide **Mr.Devendra Dhande**

Date: 12-02-2025

Table of Contents

Abstract

Vehicle safety is a crucial aspect of modern transportation systems. This project focuses on implementing the Controller Area Network (CAN) protocol for vehicle safety applications using STM32F407. Various sensors, including an ultrasonic sensor for obstacle detection, a rain sensor for water drop detection and a DHT22 sensor for temperature and humidity monitoring, are integrated into the system. The project also includes CAN-to-CAN communication for efficient data transmission, an I2C-based LCD for real-time data display, and UART for debugging and external communication. The proposed system enhances vehicle safety by providing real-time alerts and automated decision-making based on sensor inputs.

Introduction

With the increasing number of vehicles on roads, ensuring safety through automation has become essential. The CAN protocol provides an efficient and robust communication mechanism between electronic control units (ECUs) in a vehicle. This project integrates multiple sensors and communication protocols with STM32F407 to enhance vehicle safety features. By monitoring environmental conditions and detecting obstacles, the system can take preventive measures and alert the driver accordingly.

Methodology

Hardware Setup

- Utilize the STM32F407 microcontroller to manage data collection and processing.
- Connect temperature, Ultrasonic, Rain sensors to the STM32 board for data acquisition.
- Initialize the communication interface used and STM32 board to capture data.
- Ensure proper connectivity and configuration of all hardware components for seamless data acquisition.

Sensor Data Acquisition

- Implement sensor data acquisition routines on the STM32 board.
- Configure the STM32 board to read temperature sensor data, Motion sensor data.
- Continuously monitor sensor readings in real-time to capture dynamic changes in environmental conditions and industrial parameters.
- Data Transmission

- Establish communication protocols between the STM32 board and peripheral devices using CAN.

- Utilize CAN communication to transmit sensor data from the STM32 board to the other module.
- Ensure reliable and efficient data transmission to peripheral devices for further processing.

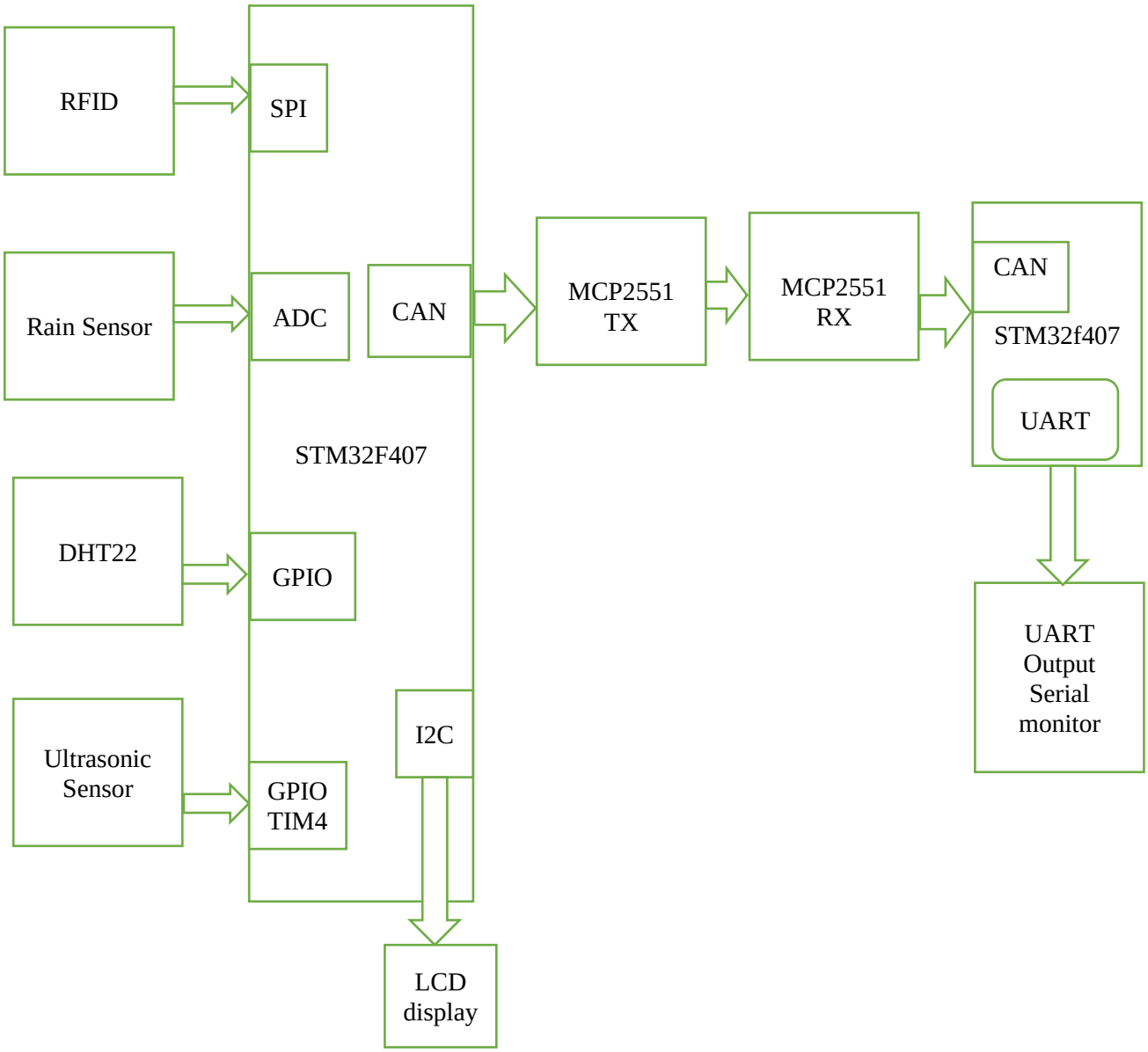


Figure: Block Diagram for proposed System

- Core Processing: The STM32F407 microcontroller serves as the central unit, managing sensor data acquisition and communication.
- Temperature Sensor (DHT22): Measures atmospheric temperature and Humidity, crucial for environmental monitoring.
- Ultrasonic Sensor: Captures movement, providing data for security or activity tracking.
- Rain Sensor: Monitors for the presence of water drops and start wiper in car
- Rfid is used for authentication in car

Hardware and Components

STM32F407G-DISC1

The STM32F407G-DISC1 microcontroller is a high-performance ARM Cortex-M4- based microcontroller unit (MCU) manufactured by STMicroelectronics. It offers a wide range of features and peripherals suitable for various embedded applications, including industrial control systems, consumer electronics, and IoT devices.

- The STM32F407G-DISC1 MCU serves as the central processing unit for the system, handling data acquisition, processing, and communication tasks.
- It features a powerful ARM Cortex-M4 core running at up to 168 MHz, providing sufficient processing power for real-time sensor data processing and control algorithms.
- Peripherals and Interfaces:
- The STM32F407G-DISC1 MCU is equipped with a rich set of peripherals and interfaces, including multiple UART, SPI, I2C, and CAN interfaces.
- These interfaces facilitate communication with external sensors, modules, and devices, enabling seamless integration into the system architecture.

Data Acquisition: The STM32F407G-DISC1 MCU interfaces with various sensors, such as temperature sensors, speed sensors, NO2 sensors, and others, to collect real time data.

- It employs dedicated ADC (Analog-to-Digital Converter) channels to digitize analog sensor readings, ensuring accurate and reliable data acquisition.
- The MCU supports various communication protocols, such as UART, SPI, and

I2C, for serial communication with peripheral devices.

- It facilitates communication with external modules, including the Bluetooth HC-05 module, ESP8266 module, LCD display, and others, enabling seamless data
- transmission and control.
- Firmware development for the STM32F407G-DISC1 MCU is carried out using integrated development environments (IDEs) such as STM32CubeIDE.

Ultrasonic Sensor

It is a device that uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. High-frequency sound waves reflect across boundaries to produce distinct echo patterns. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. This process is a key aspect of ultrasonic sensor working. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. This process is a key aspect of ultrasonic sensor working.

It all starts when the trigger pin is set HIGH for 10µs. In response, the sensor transmits an ultrasonic burst of eight pulses at 40 kHz. This 8-pulse pattern is specially designed so that the receiver can distinguish the transmitted pulses from ambient ultrasonic noise.

These eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the echo pin goes HIGH to initiate the echo-back signal.

Rain Sensor

A rain sensor or rain switch is a switching device activated by rainfall. There are two main applications for rain sensors. The first is a water conservation device connected to an automatic irrigation system that causes the system to shut down in the event of rainfall. The second is a device used to protect the interior of an automobile from rain and to support the automatic mode of windscreen wipers.
Rain sensor is an advanced driver-assistance system that detects water on a car's windscreen and automatically triggers programmed actions. The main function of this system is activating windscreen wipers in the rain. But the triggered actions might also include closing the car's windows and sunroof.

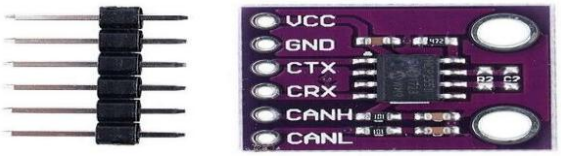
The rain sensor is located behind the windscreen near the rearview mirror. It consists of LEDs that beam infrared light and a central photodiode that measures the amount of light that lands on it. The infrared light is beamed on the windscreen at a 45-degree angle where it reflects back onto the photodiode. When the windscreen is dry, all of the light is reflected. The more drops of water there are on the windscreen, the less light the photodiode receives. This information is sent to an electronic control unit which turns on the wipers and adjusts their speed accordingly.
The purpose of rain sensors is to make driving in wet conditions safer and more comfortable. The driver does not have to shift their focus off the road to turn on the wipers or close the windows. The system does it for them and they can focus on the road. Since the automated system can react faster than a human, it is especially important in situations where the driver may experience sudden visual distractions, for example in the case of a thunderstorm.

RFID Sensor

An RFID or radio frequency identification system consists of two main components, a tag attached to the object to be identified, and a reader that reads the tag.
A reader consists of a radio frequency module and an antenna that generates a high frequency electromagnetic field. Whereas the tag is usually a passive device (it does not have a battery). It consists of a microchip that stores and processes information, and an antenna for receiving and transmitting a signal.
When the tag is brought close to the reader, the reader generates an electromagnetic field. This causes electrons to move through the tag's antenna and subsequently powers the chip.
The chip then responds by sending its stored information back to the reader in the form of another radio signal. This is called a backscatter. The reader detects and interprets this backscatter and sends the data to a computer or microcontroller.
Hardware Overview
The RC522 RFID module based on the MFRC522 IC from NXP is one of the cheapest RFID options you can get online for less than four dollars. It usually comes with an RFID card tag and a key fob tag with 1KB of memory. And the best part is that it can write a tag that means you can store any message in it.
The reader can communicate with a microcontroller over a 4-pin SPI with a maximum data rate of 10 Mbps. It also supports communication over I2C and UART protocols.
The RC522 RFID module can be programmed to generate an interrupt, allowing the module to alert us when a tag approaches it, instead of constantly asking the module "Is there a card nearby?".
The module's operating voltage ranges from 2.5 to 3.3V, but the good news is that the logic pins are

5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using a logic level converter.
Technical Specifications
Here are the specifications:
Frequency Range 13.56 MHz ISM Band
Host Interface SPI / I2C / UART
Operating Supply Voltage 2.5 V to 3.3 V
Max. Operating Current 13-26mA
Min. Current (Power down) 10µA
Logic Inputs 5V Tolerant
Read Range 5 cm.

MCP2551 CAN-BUS TRANSCEIVER



- Utilize the STM32F407VGt6 microcontroller as the core processing unit to handle CAN communication and sensor data acquisition.
- Integrate the MCP2551 CAN-BUS transceiver with the STM32 board to enable reliable CAN network communication.
- Ensure proper wiring and connections between the MCP2551 transceiver and the STM32 board for effective CAN signal transmission and reception.
- Configure the MCP2551 CAN-BUS transceiver to match the STM32's CAN peripheral settings for optimal performance and compatibility.

- Develop data acquisition routines on the STM32 board to interface with sensors and prepare data for CAN transmission.
- Leverage the ESP32 module for wireless communication to complement the CAN network by providing remote data access and control capabilities.
- Implement real-time CAN data transmission protocols on the STM32 to send sensor data via the MCP2551 transceiver and handle incoming CAN messages.
- Ensure accurate CAN message encoding and decoding to maintain data integrity and facilitate communication between the STM32 and other CAN nodes.
- Monitor CAN network traffic and sensor data on the STM32, and use the ESP32 for remote monitoring and system diagnostics.
- Configure the STM32 board to manage CAN message filtering and prioritize messages based on application needs for efficient network operation.

DHT22 Temperature and Humidity Sensor

The DHT22 is a digital temperature and humidity sensor commonly used in weather monitoring and environmental control applications. It provides accurate readings of both temperature and relative humidity with a digital output.

Key Features:

- Temperature Range: -40°C to 80°C
- Temperature Accuracy: ±0.5°C
- Humidity Range: 0% to 100% RH
- Humidity Accuracy: ±2–5% RH
- Operating Voltage: 3.3V to 6V
- Current Consumption: ~1.5mA (measuring), ~60µA (standby)
- Output Signal: Digital (Single-wire communication)
- Sampling Rate: 0.5 Hz (one reading every 2 seconds)

Working Principle:

- The temperature is measured using a thermistor (a resistive temperature device).
- The humidity is measured using a capacitive humidity sensor that detects moisture in the air.
- A built-in microcontroller processes the raw data and transmits it as a digital signal using a single-wire protocol.

Advantages:

- High accuracy and stability
- Low power consumption
- Easy interfacing with microcontrollers like Arduino, ESP32, STM32, and Raspberry Pi
- Fully calibrated and provides direct digital output

Disadvantages:

- Slower response time (due to the 2-second refresh rate)
- Limited data transmission speed
- Slightly larger size compared to other sensors like DHT11

Software

STM32CubeIDE

STM32CubeIDE, an integrated development environment (IDE) crafted by STMicroelectronics, serves as a pivotal tool in the project's software development journey targeting STM32 microcontrollers. This environment provides a holistic platform with an array of tools and features aimed at expediting firmware development. Its seamless integration with the STM32CubeMX configuration tool streamlines the process of configuring STM32 peripherals, pin assignments, and middleware components. By visually configuring the MCU's parameters through STM32CubeMX, developers can swiftly generate initialization code, significantly reducing the workload associated with peripheral setup. Furthermore, STM32CubeIDE offers robust project management capabilities, facilitating efficient organization of source files, libraries, and resources within projects. With built-in debugging and testing functionalities, including support for hardware debugging using ST-LINK or JTAG/SWD debug probes, developers can effectively debug firmware code using features like breakpoints and watchpoints. Additionally, STM32CubeIDE comes bundled with the GNU Arm Embedded Toolchain,

providing a robust compiler and toolchain optimized for ARM Cortex-M-based microcontrollers. This toolchain supports advanced compiler optimizations and debugging features, enhancing code efficiency and reliability. Integrated seamlessly with STM32Cube middleware components and software libraries, the IDE enables developers to easily incorporate middleware functionalities into their projects, further accelerating the development process. Through STM32CubeIDE's comprehensive suite of features, developers can efficiently develop, debug, and deploy firmware for STM32 microcontroller-based projects, including the envisioned wireless data transmission system.

STM32Cube Programmer

STM32Cube Programmer stands as an essential tool within the STM32 ecosystem, offering crucial functionalities for programming STM32 microcontrollers and configuring their embedded memories. This versatile tool streamlines the process of flashing firmware onto STM32 devices and managing their memory configurations. By supporting various programming modes, including UART, USB, and CAN, STM32Cube Programmer accommodates diverse deployment scenarios and ensures compatibility with a wide range of STM32 microcontrollers. Its intuitive user interface simplifies the task of selecting programming options and configuring device settings, enabling efficient and error-free programming operations. Moreover, STM32Cube Programmer integrates seamlessly with STM32CubeIDE and other development environments, providing a seamless workflow for firmware development and device programming. With support for batch programming and scripting capabilities, the tool enhances productivity and scalability, enabling developers to streamline production processes and automate repetitive tasks. STM32 Cube Programmer is a tool for programming and configuring STM32 microcontrollers, supporting firmware updates and memory operations via JTAG, SWD, and UART interfaces Overall, STM32Cube Programmer plays a crucial role in the development lifecycle of STM32based embedded systems, offering robust programming capabilities and streamlined device management functionalities.

Controller Area Network (CAN) Protocol

The Controller Area Network (CAN) protocol is used for robust communication between the STM32F407VGt6 microcontroller and external devices, such as sensors and other CAN nodes. CAN communication offers a reliable and efficient method for data exchange in automotive and industrial applications, ensuring high data integrity and error detection.

- The STM32F407VGt6 microcontroller features built-in CAN peripherals for seamless integration with CAN networks.
- CAN interfaces are configured with specific baud rates and communication settings to ensure compatibility and reliable data transfer between devices.
- CAN communication operates in a multi-master, multi-slave configuration, allowing multiple nodes to communicate over the same bus with built-in error detection and handling mechanisms.
- Data transmission occurs synchronously, with the microcontroller and CAN nodes exchanging messages framed with identifiers, data bytes, and control information.
- The microcontroller transmits and receives data packets via the CAN TX and RX pins, with the CAN transceiver handling the physical layer of communication.

- CAN communication employs a standardized message format, including identifiers, data length codes, and data payloads, to ensure accurate data exchange between nodes.
- The baud rate determines the speed of data transmission on the CAN bus and must be set identically on all nodes to ensure proper communication.
- Baud rate selection depends on the network requirements and the maximum speed supported by the CAN peripherals and transceivers.

In the project, CAN1 and CAN2 are utilized on the STM32F407VGt6 microcontroller for different communication purposes:

CAN1 Configuration (Sensor Data):

- CAN1 is configured with a baud rate of 500 kbps for communication with various sensors integrated into the system.
- The microcontroller sends sensor data packets via CAN1, with data formatted according to the CAN protocol specifications.
- The CAN transceiver converts the data into CAN-compatible signals, allowing it to be transmitted over the CAN bus to other devices.

CAN2 Configuration (External Devices):

- CAN2 is configured with a baud rate of 1 Mbps for high-speed communication with external devices or other CAN nodes.
- The microcontroller transmits control and status information through CAN2, interfacing with devices that support CAN communication.
- The data sent via CAN2 is processed by external CAN nodes or devices connected to the same bus, ensuring timely and reliable data exchange**.**

Conclusion

The CAN protocol has revolutionized vehicle safety by enabling fast, reliable, and efficient communication between safety-critical systems. Its real-time data exchange, fault tolerance, and scalability make it a preferred choice for automotive applications. With the evolution of CAN FD (Flexible Data Rate) and integration with AI-based safety systems, future vehicles will achieve even higher levels of autonomy, accident prevention, and driver assistance. Thus, CAN remains a backbone technology in modern vehicle safety and automation.

Future Scope

The future scope of a vehicle safety project using the CAN (Controller Area Network) protocol is quite promising as the automotive industry continues to evolve toward smarter and more autonomous vehicles. Here are some key areas of future development:

1. Integration with ADAS (Advanced Driver Assistance Systems)
    - CAN-based safety systems can be integrated with ADAS features like collision avoidance, lane departure warning, automatic emergency braking, and adaptive cruise control.
    - Enhancing vehicle safety by reducing human errors and improving real-time decision-making.
2. Autonomous Vehicle Communication
    - Future autonomous and semi-autonomous vehicles will require robust and real-time communication between Electronic Control Units (ECUs), which CAN networks can facilitate.
    - Integration with Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication will enhance traffic safety.
3. Real-time Vehicle Health Monitoring & Predictive Maintenance
    - CAN can enable real-time diagnostics and fault detection, improving vehicle reliability.
    - Predictive maintenance algorithms can be developed to prevent failures before they occur using CAN data logs.
4. Enhanced Cybersecurity for Vehicle Networks
    - As vehicles become more connected, CAN networks will need stronger security measures to prevent hacking and unauthorized access.
    - Implementing secure CAN protocols (such as CAN FD with encryption) will be a major focus.
5. CAN FD (Flexible Data-Rate) for High-Speed Data Transmission
    - Upgrading to CAN FD will allow higher data rates and larger payload sizes, improving the efficiency of vehicle safety systems.
    - Enables faster response times for safety-critical applications.
6. Integration with IoT and Cloud-based Monitoring

- Cloud-based vehicle monitoring using CAN data can enhance fleet management and road safety.
- IoT-enabled CAN modules can send real-time alerts about accidents, vehicle failures, and driver behavior to emergency services.

7. Energy-Efficient and Electric Vehicle (EV) Safety Enhancements
    - CAN networks will play a crucial role in Battery Management Systems (BMS) for EVs.
    - Advanced safety mechanisms for fire prevention, thermal runaway detection, and energy efficiency optimizations.
8. Smart Traffic Management Systems
    - CAN data can be used in smart city infrastructure to optimize traffic flow, reduce congestion, and improve road safety.
    - Connected vehicles sharing CAN-based telemetry can help authorities predict and prevent accidents.
9. Adaptive Safety Mechanisms for Different Road Conditions
    - CAN-based sensors and AI integration can adapt safety measures based on weather conditions, terrain, and traffic density.
    - Example: Dynamic braking systems adjusting according to road grip and weather.
10. Emergency Vehicle Communication and Crash Detection
    - Automatic crash detection and SOS alert systems using CAN-based sensor data.
    - Vehicles can communicate with emergency services for faster response times during accidents.