A
Project Report
On

"CAN BASED COLLISION AVOIDANCE SYSTEM"

Submitted in the partial fulfillment of the requirements
for the PG Diploma in

EMBEDDED SYSTEMS & DESIGN
(PG - DESD)

by

Saumya Mishra	240844230111
Gayatri Ramnath Kaje	240844230034
Sayali Pandurang Jadhav	240844230075
Harshali Kiran Patil	240844230106

Sunbeam Institute of Information Technology, Pune & Karad. Year : 2024-25

Sunbeam Institute of Information Technology, Pune & Karad.

CERTIFICATE

This is to certify,

Saumya Mishra	240844230111
Gayatri Ramnath Kaje	240844230034
Sayali Pandurang Jadhav	240844230075
Harshali Kiran Patil	240844230106

have satisfactorily completed Project and presented a report on topic titled "CAN based collision avoidance system" at Sunbeam Institute of Information Technology in partial fulfilment of requirement of PG Diploma in Embedded Systems & Design (PG-DESD) academic year 2024-2025.

Date : 12/02/2025

TABLE OF CONTENTS

1. Introduction

This system is designed to facilitate efficient communication between transmitter and receiver modules using the CAN protocol, allowing multiple sensors to monitor various vehicle parameters and display them to the driver via an LCD screen. The system utilizes a range of sensors, including a temperature sensor to monitor engine heat and an ultrasonic sensor to measure the distance between obstacles. It is crucial that the driver can simultaneously monitor vital parameters, such as engine temperature and obstacle distance, while driving.

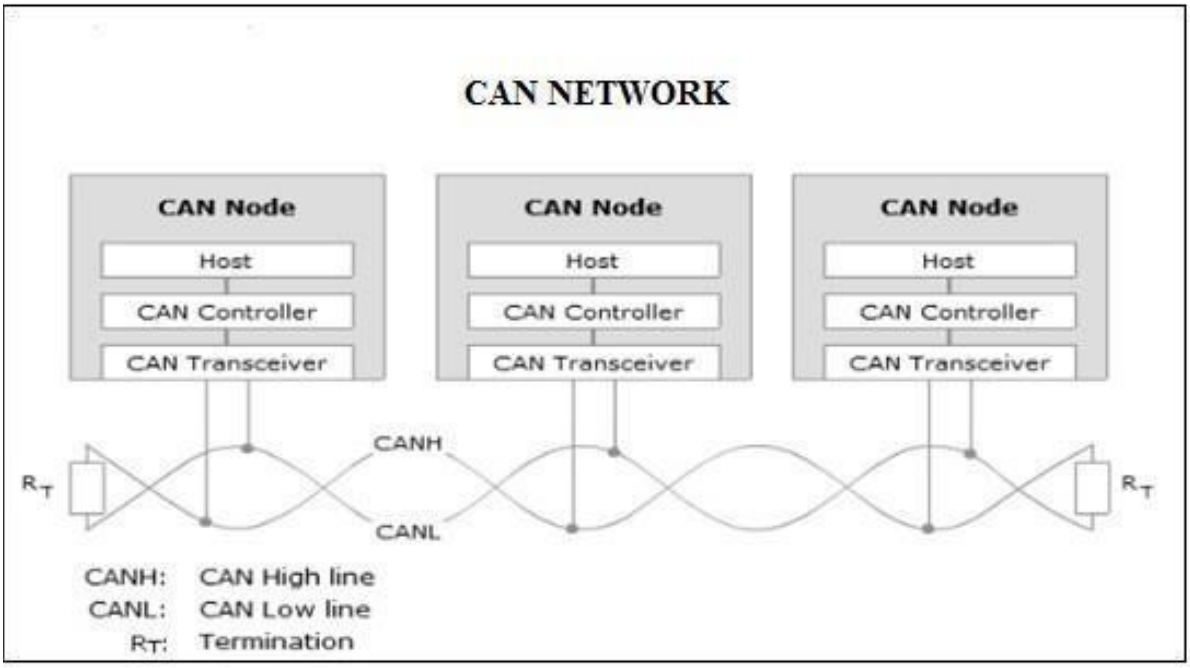
The CAN protocol (bus) is employed for seamless data transmission between the various components. To manage this communication, a CPU is required to handle tasks such as bus arbitration, message prioritization, and address identification. The STM32F407 microcontroller is chosen as the CPU, thanks to its built-in CAN controller hardware, which simplifies the implementation of CAN bus-based designs. This microcontroller is capable of efficiently managing the entire communication process.

In addition to the temperature and ultrasonic sensors, the system also incorporates two IR sensors, which are used to calculate the time taken for an object to travel between them. By measuring the time difference and knowing the distance between the IR sensors, the system is able to calculate the speed of the object, providing additional valuable data to the driver.

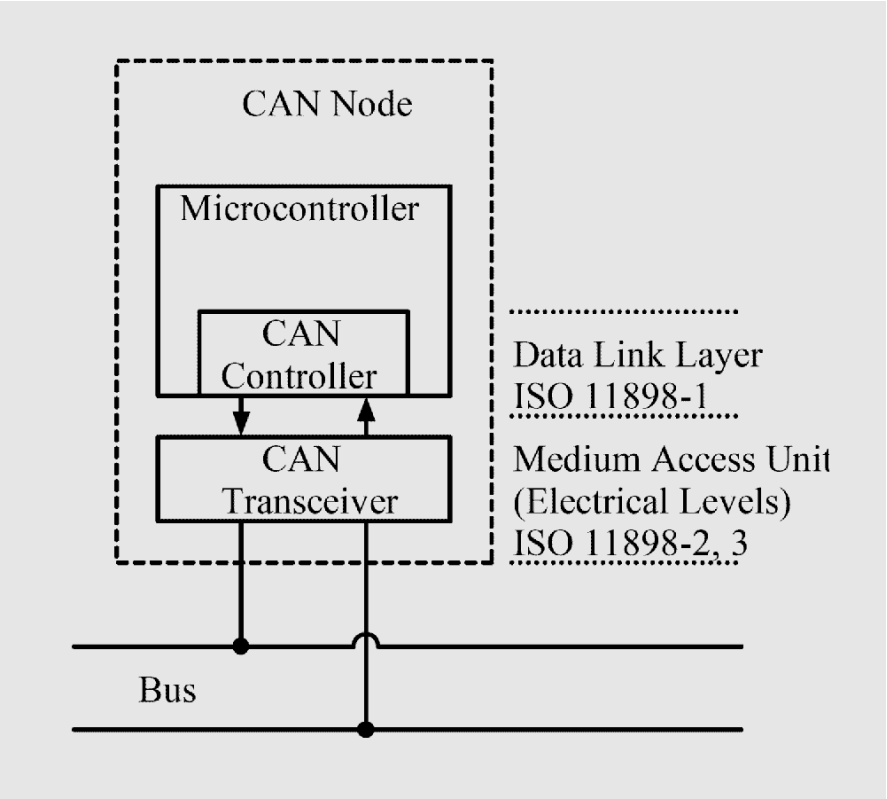
The core components of the system include a microcontroller (STM32F407) to control the data flow, a temperature sensor for engine heat detection, ultrasonic sensors for obstacle distance measurement, IR sensors for speed calculation, and a power supply to drive the entire system. Together, these elements work to enhance the driver's ability to monitor and respond to vehicle conditions in real time.

2. Controller Area Network

2.1. CAN NETWORK



2.2. CAN BUS



In a CAN network, signal transmission relies on differential voltages, which effectively minimize the impact of interference caused by sources such as motors, ignition systems, and switch contacts. As a result, the CAN bus uses two wires: CAN High and CAN Low.

To reduce electromagnetic interference further, the two wires are twisted together, which helps minimize the magnetic field. This is why twisted pair cables are commonly employed as the physical medium for signal transmission in CAN networks. However, because signals propagate at a finite speed, reflections caused by transient phenomena (such as signal bouncing) can become more problematic as the data rate increases or the bus length extends.

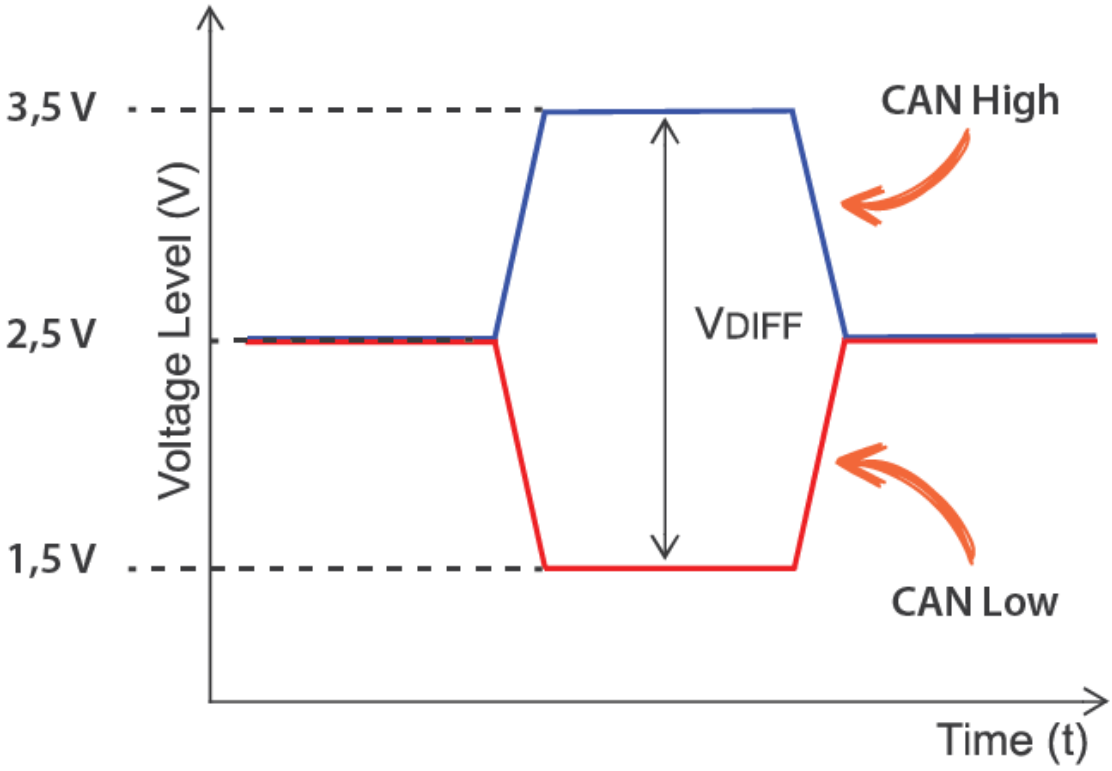
To avoid these reflections and maintain reliable communication, termination resistors are placed at both ends of the bus. These resistors simulate the electrical characteristics of the transmission medium, ensuring that the signal integrity is preserved in high-speed CAN networks.

The value of the bus termination resistor is crucial and is typically set to match the characteristic impedance of the transmission line, which is 120 Ohms. In contrast, the ISO 11898-3 standard for low-speed CAN networks does not require termination resistors, as the lower data rate of 125 kbit/s reduces the risk of signal reflections.

2.3. CAN bus levels

Physical signal transmission in a CAN network is based on differential signal transmission. The specific differential voltages depend on the bus interface that is used. A distinction is made here between the high-speed CAN bus interface (ISO 11898-2) and the low-speed bus interface (ISO 11898-3).

ISO 11898-2 assigns logical “1” to a typical differential voltage of 0 Volt. The logical “0” is assigned with a typical differential voltage of 2 Volt. High-speed CAN transceivers interpret a differential voltage of more than 0.9 Volt as a dominant level within the common mode operating range, typically between 12 Volt and -12 Volts. Below 0.5 Volt, however, the differential voltage is interpreted as a recessive level. A hysteresis circuit increases immunity to interference voltages. ISO 11898-3 assigns a typical differential voltage of 5 Volt to logical “1”, and a typical differential voltage of 2 Volt corresponds to logical “0”. The figure “High-Speed CAN Bus Levels” and the figure “Low-Speed CAN Bus Levels” depict the different voltage relationships on the CAN bus.



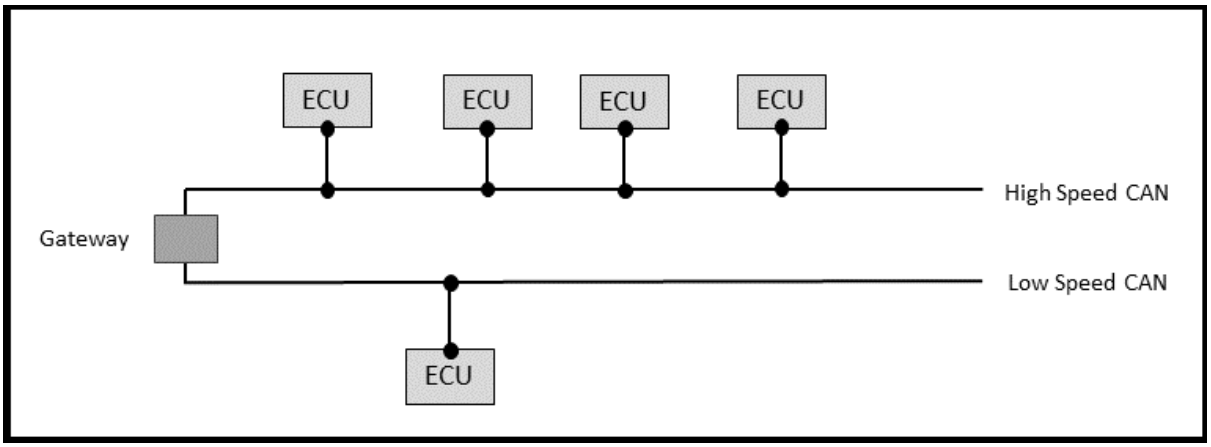
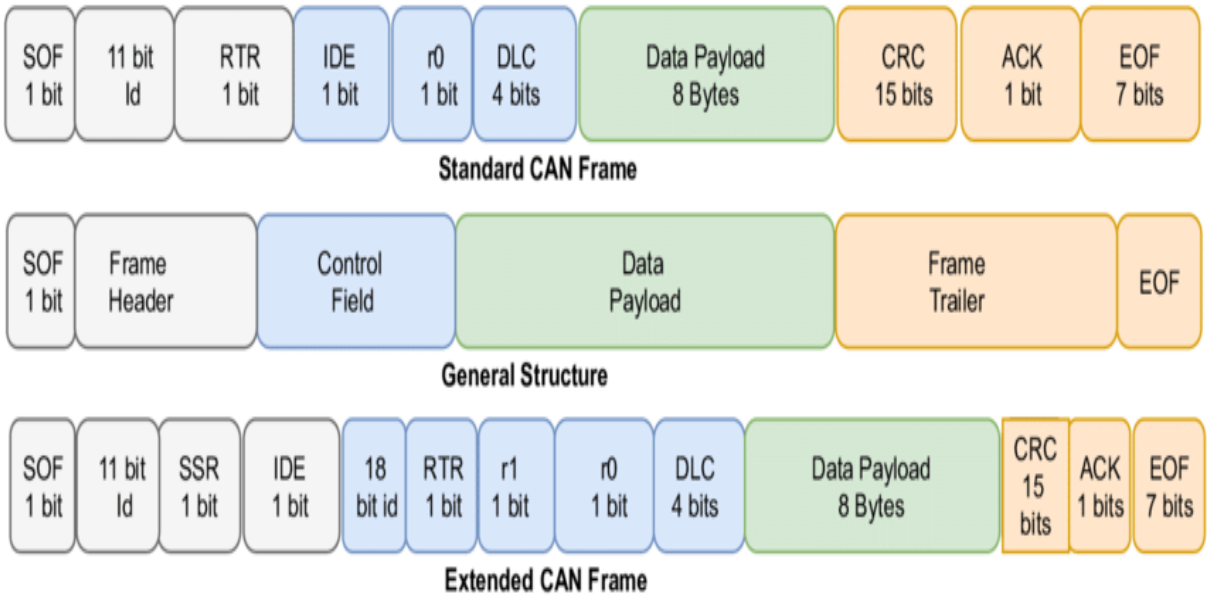


Fig.(a) HIGH AND LOW SPEED CAN

2.4. CAN Frames

A) DATA FRAME



CAN Protocol Frame Format

The following figure illustrates the structure of a CAN protocol frame. Each part of the frame has a specific function in the communication process between nodes on a CAN bus.

1. SOF (Start of Frame)

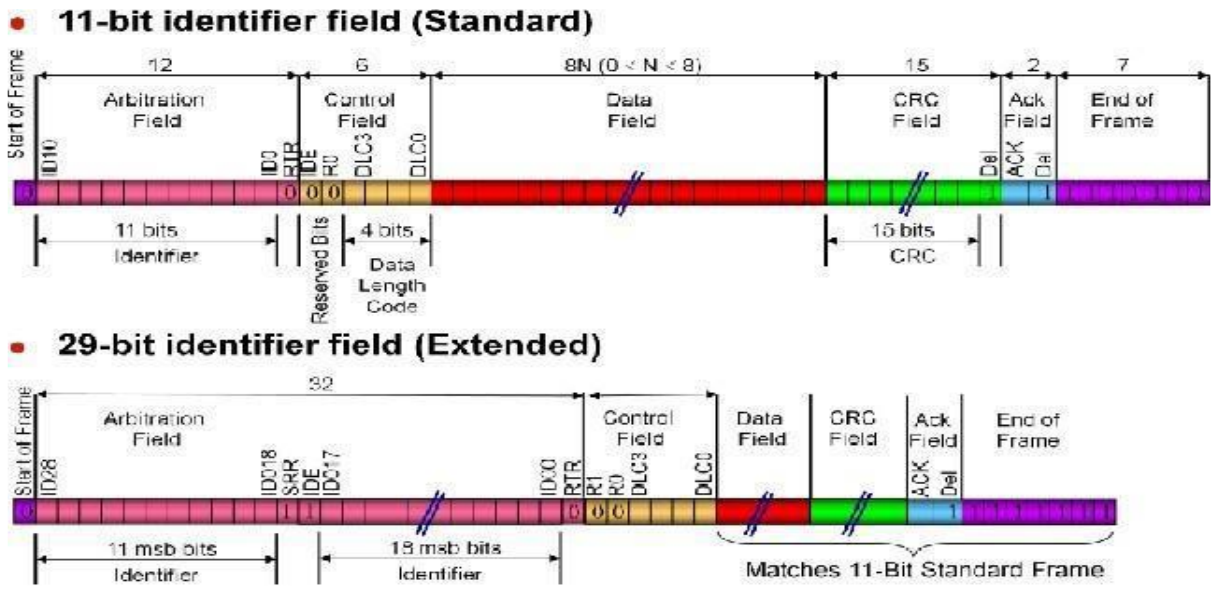
- The SOF is a single dominant bit that marks the beginning of a message. It helps synchronize all the nodes on the bus, especially after the bus has been idle. This bit is crucial for ensuring that all devices are ready to receive the incoming message.

2. Identifier

- The Identifier is an 11-bit field in the standard CAN frame. This identifier determines the priority of the message on the bus. A lower binary value of the identifier indicates a higher priority. Messages with lower identifiers are processed first when multiple nodes try to send data simultaneously.

3. RTR (Remote Transmission Request)

- The RTR bit is a single dominant bit that indicates whether the message is a data frame or a remote frame. If the bit is dominant, it signals that the node is requesting information from another node. All nodes on the network receive this request, but the node with the matching identifier responds with the requested data. This ensures that all data on the network is consistent and



shared across the system.

4. IDE (Identifier Extension)

- The IDE bit, if dominant, indicates that a standard CAN identifier is being used with no extension. In extended frames, the IDE bit signals that there are additional identifier bits following the standard 11-bit identifier, making it a 29-bit identifier.

5. r0 (Reserved Bit)

- The r0 is a reserved bit that is set aside for future expansions of the CAN standard. It currently has no defined function but may be used in future versions of the protocol.

6. DLC (Data Length Code)

- The DLC is a 4-bit field that specifies the number of bytes of data being transmitted in the message. It allows the receiving node to know how much data to expect, ranging from 0 to 8 bytes.

7. Data

- The Data field can hold up to 64 bits of application data. This is where the actual information being communicated is stored, such as sensor readings, control commands, or other data being shared across the network.

8. CRC (Cyclic Redundancy Check)

- The CRC is a 16-bit field (15 bits plus a delimiter) that is used for error detection. It contains a checksum of the data transmitted in the frame, allowing nodes to detect errors in the received message. If any error is detected, the message is discarded, and the sender must retransmit it.

9. ACK (Acknowledgment)

- The ACK field consists of two bits. The first bit is the acknowledgment bit, which is overwritten by any node that successfully receives the message. This bit becomes dominant if the message is error-free. If a node detects an error, it leaves the acknowledgment bit recessive, indicating that the message was not properly received. The second bit is a delimiter.

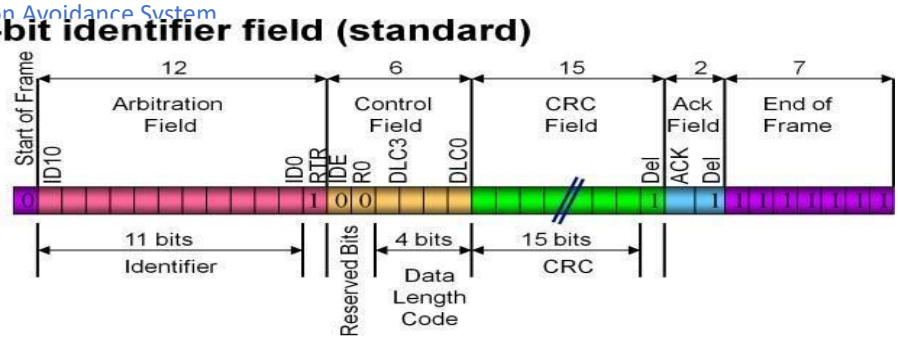
10. EOF (End of Frame)

- The EOF is a 7-bit field marking the end of a CAN frame. It indicates the completion of the message transmission. Additionally, the EOF field disables bit-stuffing, which is a method used to insert a bit of the opposite logic level into the message to prevent long sequences of identical bits. The appearance of five consecutive bits of the same level triggers bit-stuffing.

11. IFS (Inter-Frame Space)

- The IFS is a 7-bit field that defines the minimum time interval between the end of one message and the start of the next. This time is necessary for the CAN controller to move the received frame to its buffer, ensuring that data is processed in the correct sequence.

CAN Extended Frame Format



In addition to the standard CAN frame, the Extended CAN Frame includes extra fields to support longer identifiers and additional reserved bits:

1. SRR (Substitute Remote Request)

- The SRR bit replaces the RTR bit in the extended format. It acts as a placeholder for the remote transmission request in the extended frame structure.

2. IDE-A (Identifier Extension A)

- The IDE-A is a recessive bit in the identifier extension field. It indicates that more bits follow in the identifier extension. The extended identifier uses an 18-bit extension, which allows for more identifiers than the standard 11-bit format.

3. r1 (Reserved Bit 1)

- The r1 bit is an additional reserved bit included in the extended frame format. It comes after the RTR and r0 bits, serving as a placeholder for possible future extensions to the protocol.

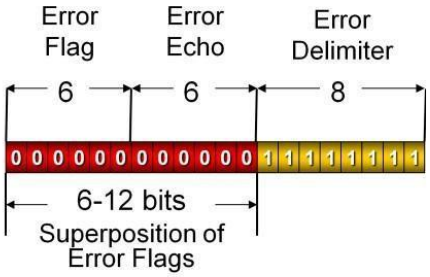
B)Remote frame

The intended purpose of the remote frame is to solicit the transmission of data from another node. The remote frame is similar to the data frame, with two important differences. First, this type of message is explicitly marked as a remote frame by a recessive RTR bit in the arbitration field, and secondly, there is no data.

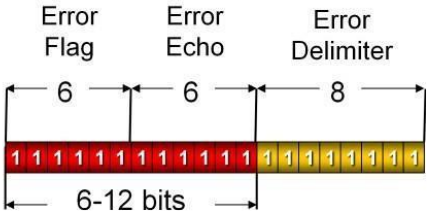
C) Error Frame

The error frame is a special message that violates the formatting rules of a CAN message. It is transmitted when a node detects an error in a message, and causes all other nodes in the network to send an error frame as well. The original transmitter then automatically retransmits the message. An elaborate system of error counters in the CAN controller ensures that a node cannot tie up a bus by repeatedly transmitting error frames.

- **Active Error Frame**



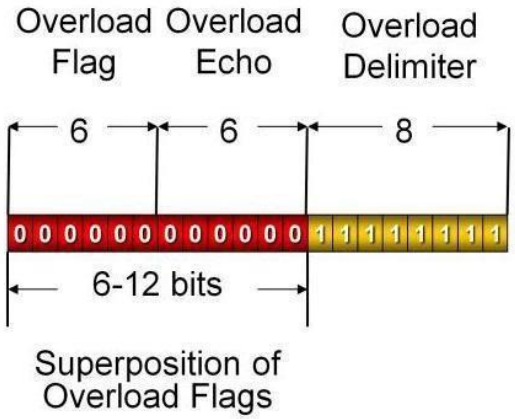
- **Passive Error Frame**



D) Overload frame

The overload frame is mentioned for completeness. It is similar to the error frame with regard to the format, and it is transmitted by a node that becomes too busy. It is primarily used to provide for an extra delay between messages.

Overload Frame

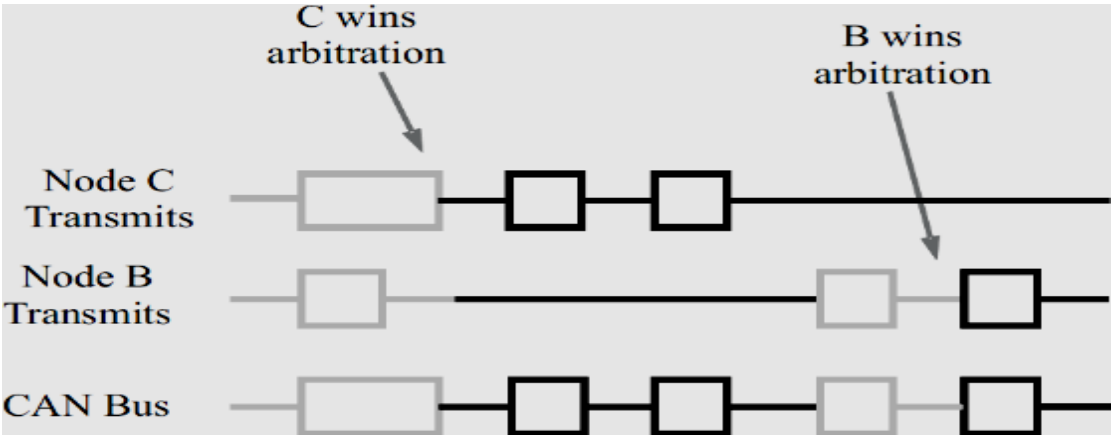


2.5. Arbitration in CAN Communication

CAN arbitration is non-destructive and relies on the identifier in the message frame to determine the priority of each message. The node with the highest priority (lowest identifier) wins the arbitration and gains access to the bus to send its message.

When multiple CAN nodes want to transmit messages simultaneously, they do not rely on any central master controller for permission to send their messages. Instead, arbitration is **distributed** among all nodes, and it proceeds as follows:

- **Step 1: Start of transmission** Each node that wants to send a message begins transmitting the **SOF (Start of Frame)** and continues to send the identifier and other parts of the message. All nodes can detect the bus state and know that the transmission is happening.
- **Step 2: Bit-by-bit comparison** While transmitting the identifier field, each node monitors the bus and compares the bits it is transmitting with the actual bits being sent on the bus. CAN uses **dominant** (0) and **recessive** (1) bits for communication.
    - A **dominant bit (0)** overpowers a **recessive bit (1)**, so if a node sends a recessive bit (1) but detects a dominant bit (0) on the bus, it knows that another node is sending a higher-priority message. This node **loses the arbitration** and immediately stops transmitting.
    -



- The node with the **lowest identifier** (dominant bits during transmission) will continue to transmit its message, as its bits will be the dominant ones during the comparison.

- **Step 3: Winning node**
  - The node with the highest priority identifier (lowest binary value) continues sending its message.
  - All other nodes that were trying to transmit a message **back off** and wait for the bus to become available again.
- **Step 4: Retransmission**
  - The nodes that lost arbitration will wait until the bus is free again and then attempt to retransmit their message. Since the CAN protocol is based on **non-destructive arbitration**, the message that "lost" the arbitration doesn't corrupt or delete the higher-priority message; it simply waits for its turn.
  - The lost messages are retransmitted after the current transmission has completed, ensuring data integrity.

3) Requirements

3.1. Hardware Requirements

A) STM32F407 Discovery Board:-The STM32F407xx family is based on the high-performance ARM® Cortex®-M4 32-bit RISC core with FPU operating at a frequency of up to 72 MHz, and embedding a floating point unit (FPU), a memory protection unit (MPU) and an embedded trace macro cell (ETM). The family incorporates high-speed embedded memories (up to 1 Mbytes of Flash memory, up to 192 Kbytes of RAM) and an extensive range of enhanced I/Os and peripherals connected to two APB buses. They also feature standard and advanced communication interfaces: up to two I2Cs, up to three SPIs (two SPIs are with multiplexed full-duplex I2Ss), three USARTs, up to two UARTs, CAN and USB. To achieve audio class accuracy, the I2S peripherals can be clocked via an external PLL. The STM32F303xB/STM32F303xC family operates in the -40 to +85°C and -40 to +105 °C temperature ranges from a 2.0 to 3.6 V power supply. A comprehensive set of power-saving mode allows the design of low-power applications.

B) LM35 Temperature Sensor

The **LM35** is a high-precision temperature sensor designed to measure environmental temperature and display the readings on an LCD. It provides accurate results with a linear voltage output corresponding to temperature in Celsius.

Key Features:

- Pre-calibrated for direct temperature readings in **Celsius**
- Generates a **linear output** with a scale factor of **10 mV per °C**
- Ensures **±0.5°C accuracy** at **25°C**
- Capable of measuring temperatures in the range of **-55°C to 150°C**
- Suitable for **remote temperature monitoring applications**
- **Cost-effective** due to wafer-level trimming
- Operates efficiently with a voltage supply between **4V and 30V**
- Consumes **less than 60 µA** of current, making it energy efficient
- Produces minimal self-heating, only **0.08°C in still air**
- Features a **low non-linearity of ±¼°C (typical)**
- Offers a **low-impedance output** (0.1 Ω for a 1-mA load), ensuring stable performance

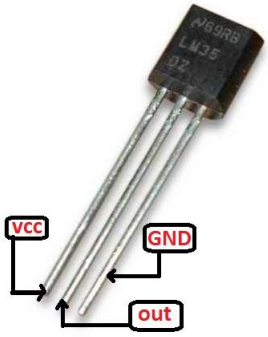


Fig b. LM35

C) HC-SR04 Ultrasonic Sensor

The **HC-SR04** ultrasonic sensor operates by emitting high-frequency sound waves and measuring the time taken for the echo to return after hitting an object. This time interval is used to calculate the distance of the object from the sensor.

Features and Working Principle:

- Capable of **non-contact distance measurement** ranging from **2 cm to 400 cm** with an accuracy of **±3 mm**.
- The sensor module consists of an **ultrasonic transmitter, receiver, and control circuit**.
- The measurement process follows these steps:
    1. A **trigger signal** of at least **10 microseconds** is sent through the I/O pin.
    2. The module transmits **eight pulses of 40 kHz ultrasonic waves** and detects if an echo is received.
    3. If the signal returns, the duration of the **high-level pulse** on the output pin represents the time taken for the wave to travel to the object and back.
    4. The distance is calculated using the formula: Distance=High-level time×Speed of sound (340 m/s)2\text{Distance} = \frac{\text{High-level time} \times \text{Speed of sound (340 m/s)}}{2}Distance=2High-level time×Speed of sound (340 m/s)

Fig c. ultrasonic sensor

D) IR Sensor

An **Infrared (IR) sensor** is an electronic device that detects infrared radiation from objects and determines their presence, distance, or motion. It is commonly used in automation, obstacle detection, and communication systems.

Features and Working Principle:

- Capable of detecting objects based on **infrared reflection and absorption**.
- Operates within a specific **wavelength range** of infrared light.
- Consists of an **IR transmitter (LED) and an IR receiver (photodiode or phototransistor)**.
- The working process involves the following steps:
    1. The **IR LED emits infrared rays** toward the target object.
    2. If an object is present, the **infrared waves are reflected** back to the receiver.
    3. The **IR receiver detects the reflected rays** and converts them into an electrical signal.
    4. The signal is processed to determine the **object's presence or distance**.

IR sensors are widely used in **proximity sensing, line-following robots, motion detection, and security systems** due to their **high sensitivity and fast response time.**

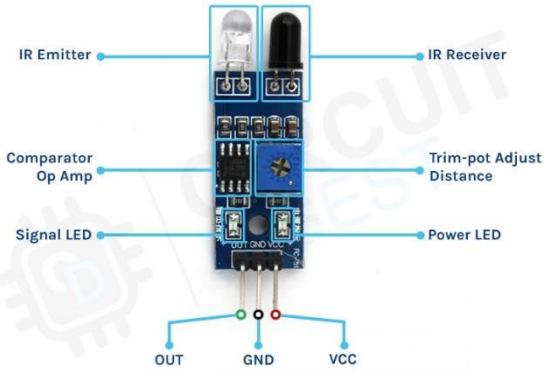


Fig d.IR sensor

E) I2C serial interface 20 x 4 LCD module :

- This is I2C interface 20x4 LCD display module, a new high-quality 4 line 20 character LCD module with on-board contrast control adjustment, backlight and I2C communication interface.

Features :

- Display Type: Black on yellow green backlight. 20 x 4 display format .
- I2C Address:0x38-0x3F (0x3F default)
- Supply voltage: 5V
- Interface: I2C to 4bits LCD data and control lines.
- Contrast Adjustment : built-in Potentiometer.

Fig. e. LCD display with I2C interface

F) MCP2551 CAN Transceiver

The MCP2551 is a high-performance CAN transceiver that implements the ISO-11898 standard physical layer requirements. It is designed to support reliable communication in automotive and industrial applications. Below are its key features:

Key Features:

- **ISO-11898 Compliance**: Implements standard physical layer requirements.
- **High-Speed Operation**: Supports up to 1 Mb/s data transmission.
- **Wide Voltage Compatibility**: Suitable for both 12V and 24V systems.
- **Reduced RFI Emissions**: Externally-controlled slope for minimized radio frequency interference.
- **Fault Detection**: Detects ground faults (permanent dominant state) on the TXD input.
- **Power Management**: Includes power-on reset and voltage brown-out protection.
- **Bus Stability**: Ensures an unpowered node or brown-out event does not disturb the CAN bus.
- **Low-Power Operation**: Supports standby mode with minimal current consumption.
- **Robust Protection Features**:
    - Protection against short-circuit conditions (both positive and negative battery voltage).
    - High-voltage transient protection for enhanced durability.
    - Automatic thermal shutdown protection to prevent overheating.
- **Scalability**: Supports up to 112 nodes on a single CAN network.
- **High Noise Immunity**: Utilizes differential bus implementation for robust signal integrity.

- **Wide Temperature Range**:
    - **Industrial (I) Grade**: -40°C to +85°C
    - **Extended (E) Grade**: -40°C to +125°C

Fig f. MCP2251 Module

G) Buzzer

The buzzer is an audio signaling device used on the receiver side of the system in this project.

Features:

- **Physical Dimensions**: 38mm diameter, 22mm height, with 2 mounting holes at a 40mm distance.
- **Wiring**: Comes with two wires for easy connectivity.
- **Rated Voltage**: 12V DC.
- **Operating Voltage**: Ranges from 3V to 24V for flexible usage.
- **Buzzer Type**: Piezoelectric for efficient and reliable performance.
- **Sound Pressure Level**: 95dB for clear and audible alerts.
- **Low Power Consumption**: Ensures energy efficiency.
- **Durability**: Designed for long-term operation with stable performance.
- **Application**: Suitable for alarms, notifications, and emergency alerts in various systems.

fig. g. Buzzer

3.2. Software Requirements

A) STM32CubeIDE

STM32CubeIDE is a comprehensive development tool that supports multiple operating systems and is a key part of the STM32Cube software suite. It serves as an advanced C/C++ programming environment designed for STM32 microcontrollers and microprocessors. This platform offers capabilities such as peripheral configuration, automated code generation, compilation, and debugging.

Built upon the Eclipse®/CDT™ framework and GCC toolchain, STM32CubeIDE employs GDB for debugging and allows seamless integration of a wide range of plugins that enhance the functionality of the Eclipse® IDE. It incorporates STM32CubeMX features, enabling streamlined STM32 device configuration and project setup within a single interface, reducing both installation complexity and development time.

Users can initiate a project by selecting an STM32 microcontroller or microprocessor, choosing from a variety of boards, or using predefined example configurations. The software then generates the necessary initialization code, making the development process more efficient and user-friendly.

At any time during the development, the user can return to the initialization and configuration of the peripherals or middleware and regenerate the initialization code with no impact on the user code.

STM32CubeIDE includes build and stack analysers that provide the user with useful information about project status and memory requirements. STM32CubeIDE also includes standard and advanced debugging features including views of CPU core registers, memories, and peripheral registers, as well as live variable watch, Serial Wire Viewer interface, or fault analyser.

Features :

- Integration of services from STM32CubeMX:STM32 microcontroller, microprocessor, development platform and example project selection Pinout, clock, peripheral, and middleware configuration Project creation and generation

of the initialization code Software and middleware completed with enhanced STM32Cube Expansion Packages.

- Based on Eclipse®/CDT™, with support for Eclipse® add-ons, GNU C/C++ for Arm® toolchain and GDB debugger
- STM32MP1 Series: Support for Open ST Linux projects: LinuxSupport for Linux
- Additional advanced debug features including:CPU core, peripheral register, and memory viewsLive variable watch viewSystem analysis and real-time tracing (SWV)CPU fault analysis tool RTOS-aware debug support including Azure.
- Support for ST-LINK (STMicroelectronics) and J-Link (SEGGER) debug probes
- Multi-OS support: Windows®, Linux®, and macOS®, 64-bit versions only

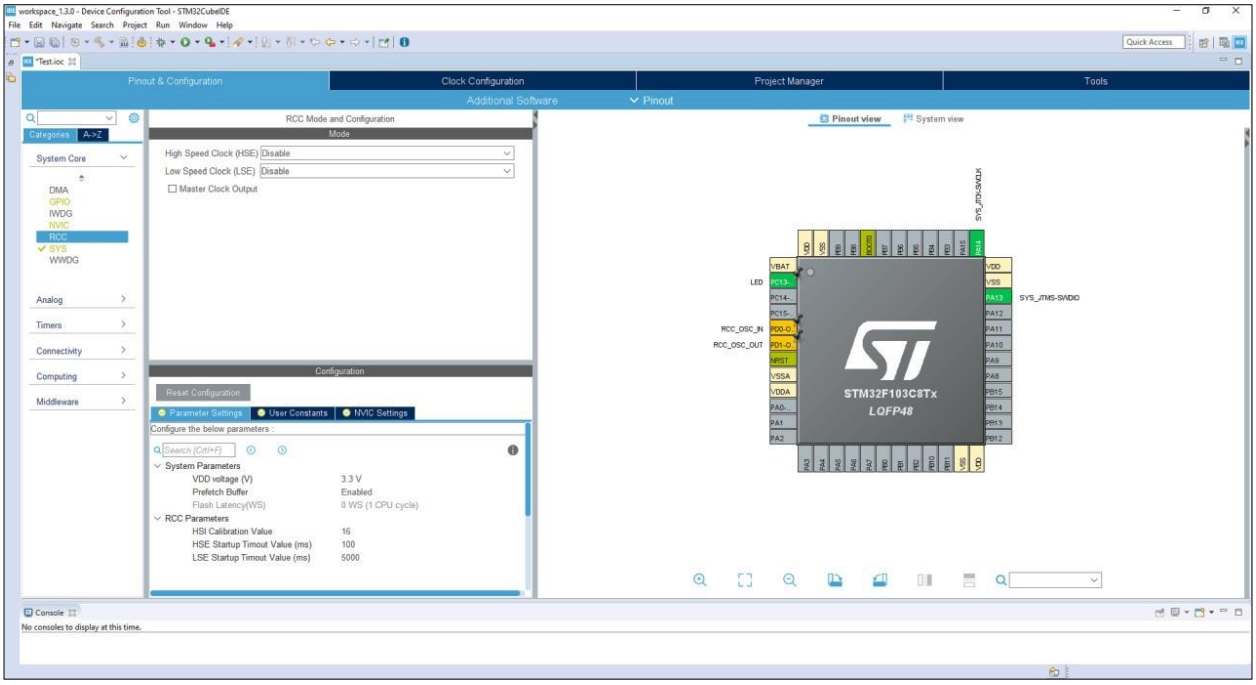


fig h. STM32 cube IDE interface

4.Project Architecture And Flowchart

4.1 Block Diagram

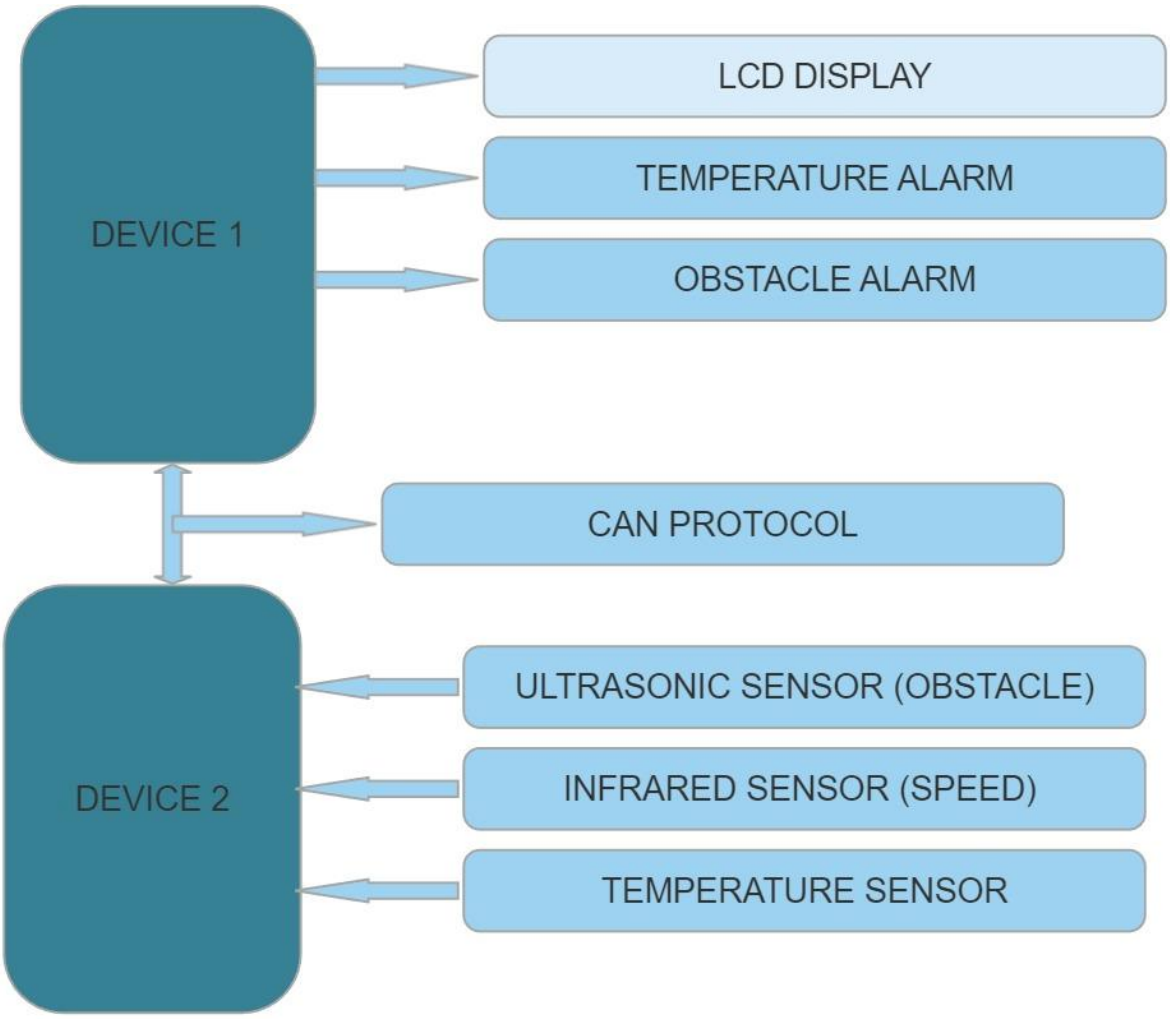


Fig i. CAN Based Collision Avoidance System

4.2. Description

As per block diagram here we are using two STM32F407 discovery board, one is used as a transmitter and another one is used as a receiver.

It uses a temperature sensor to detect engine heat and ultrasonic sensor to detect the distance between object and vehicle using STM32F407 MicroController.

On transmitting end STM32F407 MicroController take data of temperature sensor and ultrasonic sensor then this data will be transmitted to the receiver's end.

This system uses CAN protocol for communication between sensors and display /alert system at the other end.

Driver alert system transmitting and receiving end. Transmitter end consists of inbuilt CAN module in discovery board and transreceiver (MCP2551). LM35 (temperature sensor) is connected to A0 pin at receivers side and PA0 and PA1 pins are connected to ultrasonic sensor (HCSR407). PB9 and PB8 (CAN tx and CAN rx pins) are connected to MCP 2551 Rx and Tx end. Both transreceivers are connected together with connection of CAN high and CAN low connections. At the receivers end we I2C module of LCD is connected with PB6 for SDL signal and PB7 for SDA signal. For buzzer we have used PB14 as output pin.

4.3. Flowchart :

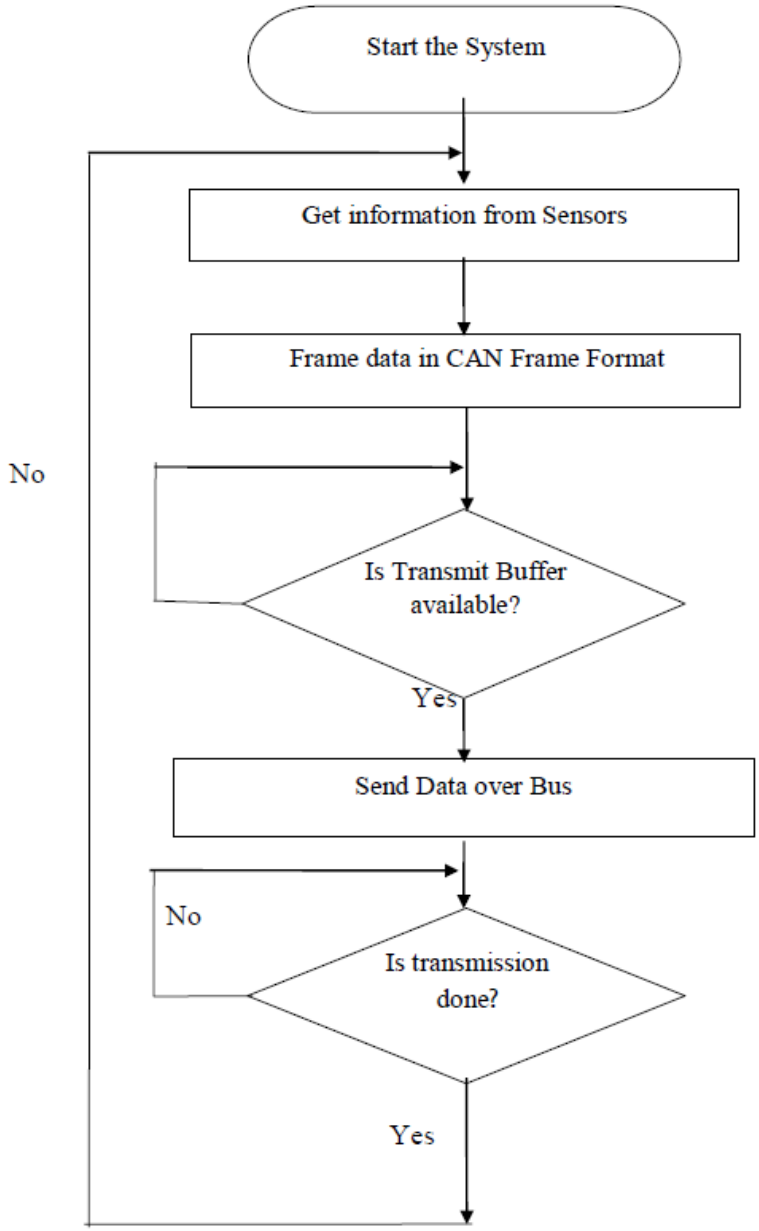


Fig j. Node A flowchart

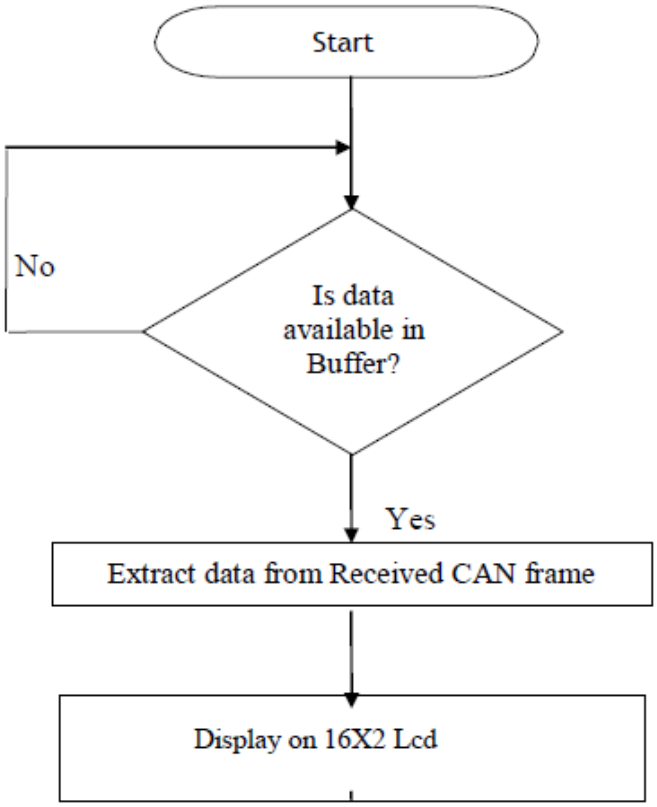
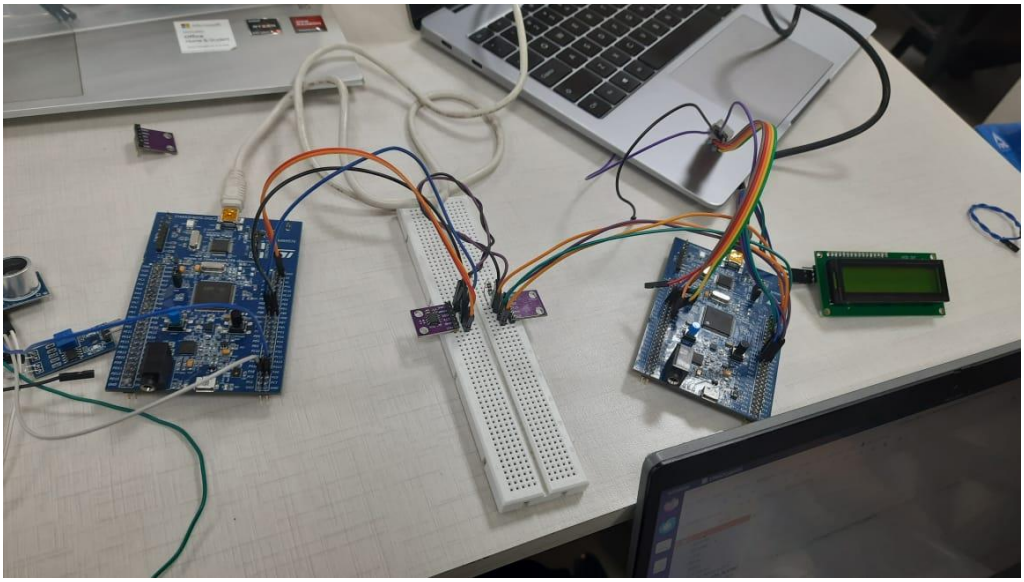


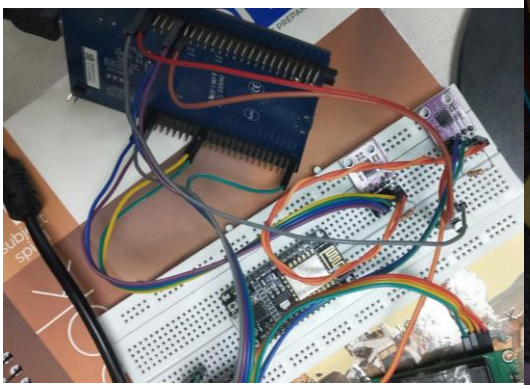
Figure 4.3 Node B Flowchart

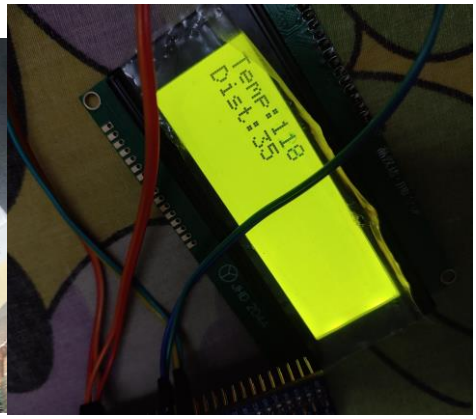
5.Testing images

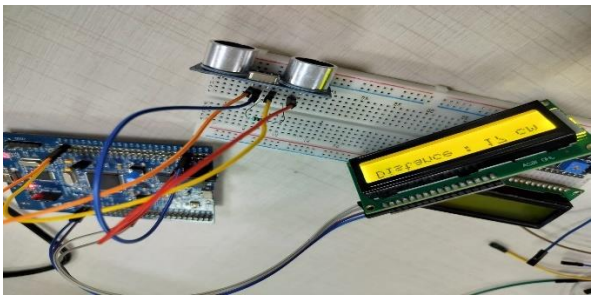
5.1. CAN COMMUNICATON :



5.2. HC-SR04 Ultrasonic Sensor and LM35:







6.Conclusion

This system is designed to implement a CAN bus-based communication network for an intelligent driver alert system. It displays critical vehicle status parameters such as fuel level, vehicle speed, obstacle detection, and engine temperature digitally on an LCD. The controller processes and transmits signal information to alert the driver accordingly.

The proposed high-speed CAN bus system enhances automotive applications by ensuring efficient data transfer between different nodes, improving both vehicle safety and driver awareness in real-world scenarios.

7.Future Scope

We have successfully implemented a system based on the CAN protocol that can detect and provide alerts to prevent excessive heating of a car and measure the distance between an obstacle and the vehicle. This has been achieved using a temperature sensor and an ultrasonic sensor mounted on an STM32F407 microcontroller. The system consists of two STM32F407 boards communicating with each other, where the sensor data is transmitted from one board to another. The received data is then displayed on an LCD mounted on the second hardware unit.

As a future enhancement, the system can be extended to allow remote monitoring and control. For example, users could remotely turn on the air conditioning, activate a buzzer, or roll up/down windows based on the sensor data. A similar application has been utilized by TATA Motors for remote AC control.

Additionally, for enhanced user accessibility, the collected sensor data can be uploaded to the cloud using IoT protocols such as MQTT, enabling real-time monitoring and data analysis.

8.References

1) Jadhav Snehal Dnyandeol, Taware Tejashree Brahmadeo2, and Jadhav Shamal popatrao3, VEHICLE CONTROL SYSTEM USING CAN PROTOCOL. International Journal of Engineering Research and General Science Volume 3, Issue 3, May-June, 2015.

2) Manoj Prasanth.R, Raja.S, and Saranya.L, VEHICLE CONTROL USING CAN PROTOCOL FOR IMPLEMENTING THE INTELLIGENT BRAKING SYSTEM (IBS). International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. Vol. 3, Issue 3, March 2014.

3) Muhammad Ali Mazidi, Rolin D. McKinlay, and Danny Causey: PIC MICROCONTROLLER AND EMBEDDED SYSTEMS, PE India, 01-Sep-2008.

4) https://elearning.vector.com/vl_can_introduction_en.html

5)https://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php