PROJECT REPORT

ON

DASHBOARD DESIGN SYSTEM USING CAN PROTOCOL & IOT

In

PG - DESD

By

NIKAM JAISH SANJAY
KULKARNI SWAROOP SANJOG
KUTE AKASH RUSTUMRAO
BANKAR RAHUL DASHRATH

Under the Guidance of

Sohail Inamdar sir

Feb-2025

Affiliated to

Sunbeam Infotech, Pune

Sunbeam Institute of Information Technology,
Pune and Karad

DASHBOARD DESIGN SYSTEM

USING

CAN PROTOCOL & IOT

CERTIFICATE

This is to Certify that the Project Report Entitled, Certified that the Project Progress report entitled , “DASHBOARD DESIGN SYSTEM USING CAN PROTOCOL & IOT” is a bonafied

Submitted By

NIKAM JAISH SANJAY

KULKARNI SWAROOP SANJOG

KUTE AKASH RUSTUMRAO

BANKAR RAHUL DASHRATH

In partial fulfillment of the requirements for the award of PG Diploma in Embedded System and Design (PG-DESD)

Date: 10/08/2025

ACKNOWLEDGMENT

We would like to express our deep gratitude to Mr. NILESH GHULE Sir and Mr. DEVENDRA DHANDE Sir for their patient guidance, enthusiastic involvement and useful critiques of this project work. Our grateful thanks extends to them for their advice and assistance in keeping our progress on schedule right from project synopsis to project report. We're very much thankful to them for encouraging us to acquire knowledge of LATEX system, which is a document preparation system for high-quality typesetting.

We would also like to extend our thanks to the lab mentors for their help in offering us the resources. we are grateful.

ABSTRACT

This project presents the design and implementation of a real-time embedded dashboard system for continuous monitoring of multiple sensor parameters using STM32F407 microcontrollers and the Controller Area Network (CAN) protocol. The system integrates three sensors: the Robado REL35 analog fuel level sensor, the LM393 speed sensor, and the LM35 temperature sensor, enabling comprehensive data acquisition for vehicle or industrial applications. The transmitter node acquires sensor readings through Analog-to-Digital Conversion (ADC), General-Purpose Input/Output (GPIO), and interrupt-driven interfaces. Sensor data is processed using the STM32 Hardware Abstraction Layer (HAL) library to ensure portability and maintainability. The processed data is transmitted over the CAN bus, offering high reliability, fault tolerance, and low-latency communication.

On the receiver side, the CAN messages are decoded, parsed, and displayed via a web-based dashboard hosted on a connected system. This interface allows real-time visualization of fuel level, speed, and temperature, providing an intuitive and user-friendly monitoring solution. The system architecture supports modular expansion, enabling the integration of additional sensors or communication modules without significant redesign.

The implementation demonstrates the effective combination of embedded peripherals, real-time data processing, and robust communication protocols for industrial and vehicular monitoring. It also showcases a scalable framework suitable for deployment in smart transportation systems, IoT-enabled monitoring solutions, and remote telemetry applications.This project presents the design and implementation of a real-time embedded dashboard system for continuous monitoring of multiple sensor parameters using STM32F407 microcontrollers and the Controller Area Network (CAN) protocol. The system integrates three sensors: the Robado REL35 analog fuel level sensor, the LM393 speed sensor, and the LM35 temperature sensor, enabling comprehensive data acquisition for vehicle or industrial applications. The transmitter node acquires sensor readings through Analog-to-Digital Conversion (ADC), General-Purpose Input/Output (GPIO), and interrupt-driven interfaces. Sensor data is processed using the STM32 Hardware Abstraction Layer (HAL) library to ensure portability and maintainability. The processed data is transmitted over the CAN bus, offering high reliability, fault tolerance, and low-latency communication.

On the receiver side, the CAN messages are decoded, parsed, and displayed via a web-based dashboard hosted on a connected system. This interface allows real-time visualization of fuel level, speed, and temperature, providing an intuitive and user-friendly monitoring

solution. The system architecture supports modular expansion, enabling the integration of additional sensors or communication modules without significant redesign.

The implementation demonstrates the effective combination of embedded peripherals, real-time data processing, and robust communication protocols for industrial and vehicular monitoring. It also showcases a scalable framework suitable for deployment in smart transportation systems, IoT-enabled monitoring solutions, and remote telemetry applications.

Contents

List of Figures

Tools Used

- STM32F407VG Discovery Board (2)
- CAN Transreceivers (2)
- STM32 cubeIDE
- LM393 IR speed sensor module
- Robodo REL 35 Sensor
- LM35 Temperature Sensor
- DC motor
- ESP8266 Wifi Module
- VS code

1 INTRODUCTION

1.1 Introductiion

The Dashboard Monitoring System using CAN integrates LM393 IR-based speed sensors, a Robodo REL 35 fuel level sensor, and a LM35 temperature sensor with STM32 microcontrollers to provide real-time monitoring of critical vehicle parameters, including speed, fuel level, and temperature. The LM393 sensor detects wheel or shaft rotation by counting infrared beam interruptions to calculate RPM and vehicle speed, while the Robodo REL 35 sensor, adapted for fuel tanks, measures fuel level through resistive sensing, and the LM35 digitally measures ambient temperature. Data from these sensors is processed by the STM32 and transmitted via the robust, high-speed, and error-checked Controller Area Network (CAN) protocol to the dashboard unit for clear, user-friendly display, ensuring reliable communication even in electrically noisy automotive environments. By centralizing speed, fuel, and temperature information, the system enhances driver awareness, supports maintenance planning, improves safety, and optimizes vehicle performance for both personal and commercial applications.

1.2 Need

In modern vehicles, the ability to monitor multiple critical parameters in real-time is essential for ensuring safety, efficiency, and performance. Traditional analog or standalone monitoring systems often lack integration, reliability, and the ability to provide accurate, timely data under varying operating conditions. The Dashboard Monitoring System using CAN addresses these limitations by centralizing speed, fuel level, and temperature measurements into a single, reliable communication framework. Accurate speed monitoring through the LM393 IR-based sensor helps drivers maintain optimal driving speeds, adhere to traffic regulations, and detect mechanical issues related to wheel or shaft rotation. Fuel level monitoring with the Robaodo REL 35 sensor enables better trip planning, reduces the risk of unexpected breakdowns, and supports fuel management in commercial fleets. Temperature monitoring with the LM35 assists in detecting abnormal thermal conditions that could affect engine health, passenger comfort, or vehicle electronics. The integration of these sensors through the Controller Area Network (CAN) ensures high-speed, error-checked data transmission even in electrically noisy environments, a critical requirement for automotive systems. By consolidating data on a user-friendly dashboard, the system reduces driver distraction, enhances situational awareness, and facilitates preventive maintenance. This makes it invaluable for improving safety, reducing operational costs, and extending the vehicle's service life.

1.3 Targeted Community

The Dashboard Monitoring System using CAN is aimed at a diverse community that relies on accurate, real-time vehicle parameter monitoring for safety, performance, and efficiency. It is valuable for automobile manufacturers seeking to integrate advanced centralized monitoring in passenger cars, trucks, and buses, as well as for commercial fleet operators in logistics, transportation, and delivery services who require precise speed, fuel, and temperature data to optimize operations and reduce maintenance costs. Public transportation agencies can use the system to ensure safe and efficient operation of buses and other public vehicles, while agricultural, construction, and mining vehicle operators can benefit from its reliability in harsh environments. Automotive service providers can leverage the system for advanced diagnostics and preventive maintenance, and research institutions can use it for prototyping and testing automotive technologies. Additionally, defense and emergency services, including military vehicles, ambulances, and fire trucks, can depend on its robust and immediate data delivery to enhance mission success and safety, making the system relevant for both personal and industrial applications.

1.4 Scope

The Scope of the project is in the automobile industry.

1.5 Objective

The primary objective of the Dashboard Monitoring System using CAN is to design and implement an integrated, real-time vehicle monitoring solution that utilizes LM393 IR-based speed sensors, a Robodo REL 35 fuel level sensor, and a LM35 temperature sensor with STM32 microcontrollers to measure and process critical parameters including speed, fuel level, and temperature. By employing the Controller Area Network (CAN) protocol, the system aims to ensure high-speed, error-free, and reliable communication between sensors and the dashboard unit, even in electrically noisy automotive environments. This objective encompasses enhancing driver awareness through clear and consolidated data display, improving vehicle safety by enabling timely detection of abnormal conditions, supporting preventive maintenance to extend vehicle lifespan, and optimizing operational efficiency for both personal and commercial vehicle applications.

1.6 CAN Protocol

The CAN bus was developed by BOSCH as a multi-master, message broadcast system that specifies a maximum signaling rate of 1 megabit per second (bps). Unlike a traditional network such as USB or Ethernet, CAN does not send large blocks of data point-to-point from node A to node B under the supervision of a central bus master. In a CAN network, many short messages like temperature or RPM are broadcast to the entire network, which provides for data consistency in every node of the system. CAN is an International Standardization Organization (ISO)-defined serial communications bus originally developed for the automotive industry to replace the complex wiring harness with a two-wire bus. The specification calls for high immunity to electrical interference and the ability to self-diagnose and repair data errors.

CAN Frame Format

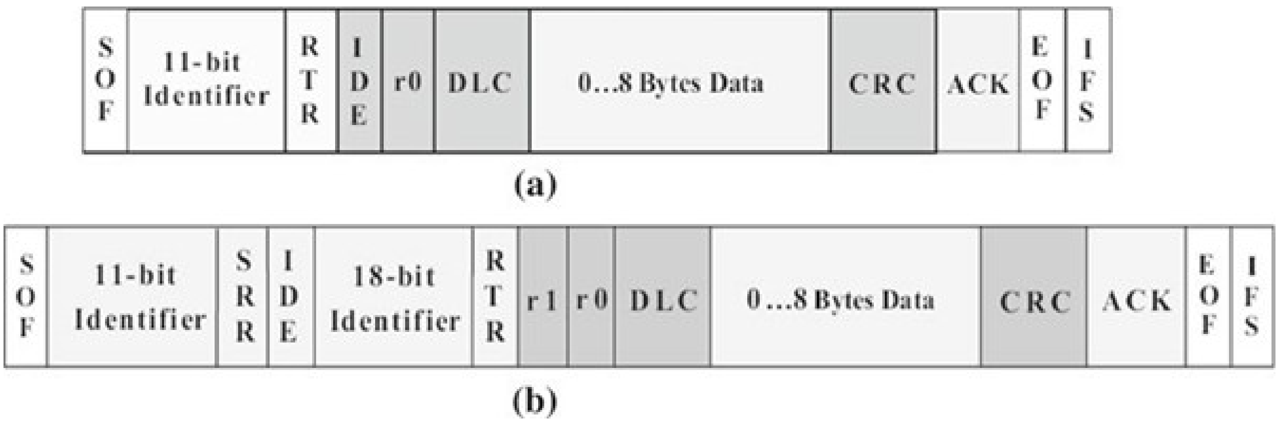


Figure 1: CAN Frame Format

A] The following figure shows the CAN protocol's frame format:

- SOF: The single dominant start of frame (SOF) bit marks the start of a message and is used to synchronise the nodes on a bus after being idle.
- Identifier: The standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.
- RTR: The single remote transmission request (RTR) bit is dominant when information is required from another node. All nodes receive the request, but the identifier determines the specified node. The responding data is also received by all nodes and used by any node interested. In this way, all data being used in a system is uniform.
- IDE: A dominant single identifier extension (IDE) bit means that a standard CAN identifier with no extension is being transmitted.
- ro: Reserved bit (for possible use by future standard amendment).
- DLC: The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.
- Data: up to 64 bits of application data may be transmitted.
- CRC: The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) contains the checksum (number of bits transmitted) of the preceding application data for error detection.
- ACK: Every node receiving an accurate message overwrites this recessive bit in the original message with a dominant bit, indicating an error-free message has been sent. Should a receiving node detect an error and leave this bit recessive, it discards the message, and the sending node repeats the message after re-arbitration. In this node acknowledges (ACK) the integrity of its data. ACK is 2 bits; one is the acknowledgement bit and the second is a delimiter. way, each
- EOF: This end-of-frame (EOF), 7-bit field marks the end of a CAN frame (message) and disables bit-stuffing, indicating a stuffing error when dominant. When 5 bits of the same logic level occur in succession during normal operation, a bit of the opposite logic level is stuffed into the data.

- IFS: This 7-bit inter-frame space (IFS) contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area.

  B] The Extended CAN Frame format consists of these additional bits:

  - SRR: The substitute remote request (SRR) bit replaces the RTR bit in the standard message location as a placeholder in the extended format.
  - IDE-A recessive bit in the identifier extension (IDE) indicates that more identifier bits follow. The 18-bit extension follows IDE.
  - r1: Following the RTR and r0 bits, an additional reserve bit has been included ahead of the DLC bit.

2 LITERATURE SURVEY

[1] Kumar, M., A. Verma, and A. Srividya (2009) – Response-Time Modeling of Controller Area Network (CAN) This study presents a mathematical framework for predicting response times in CAN-based systems under varying network loads. The authors analyze message priorities, bus arbitration delays, and jitter effects to ensure real-time performance. This work is crucial for understanding timing constraints in embedded CAN applications.

[2] Tindell, K., A. Burns, and A.J. Wellings (2005) – Calculating Controller Area Network (CAN) Message Response Times The authors introduce a systematic approach to determine message transmission delays in CAN. They consider factors like message priority levels, bus utilization, and worst-case scenarios, providing a strong foundation for designing time-critical CAN networks.

[3] Li, M. (2009) – Design of Embedded Remote Temperature Monitoring System based on Advanced RISC Machine This work focuses on ARM-based embedded systems for remote temperature monitoring. The paper details sensor interfacing, data acquisition, and communication modules, offering practical insights for IoT-based environmental monitoring solutions.

[4] Prodanov, W., M. Valle, and R. Buzas (2009) – A Controller Area Network Bus Transceiver Behavioral Model for Network Design and Simulation The authors develop a simulation model for CAN bus transceivers, enabling network designers to predict electrical performance and reliability under various operating conditions. This is particularly relevant for pre-deployment validation of CAN networks.

[5] ISO 11898:1993 – Road Vehicles: Interchange of Digital Information: Controller Area Network (CAN) for High-Speed Communication This international standard defines the physical and data link layers of high-speed CAN communication. It specifies parameters like bit timing, error handling, and frame formats, forming the basis for industrial and automotive CAN implementations.

[6] B. GmbH (1991) – CAN Specification, Version 2.0 This specification document outlines the official technical details of CAN, including protocol architecture, message frames, and arbitration mechanisms. It is an essential reference for both hardware and software developers working with CAN technology.

[7] Pazul (1999) – Controller Area Network (CAN) Basics, Microchip Technology Inc., AN713 Pazul provides an introductory guide to CAN, covering its working principles, advantages, and application areas. The document serves as a starting point for engineers new

to CAN-based embedded system development.

[8] Wilfried Voss (2005–2008) – A Comprehensive Guide to Controller Area Network Voss delivers a detailed explanation of CAN architecture, protocol layers, and implementation strategies. The book includes practical examples and troubleshooting tips, making it valuable for both beginners and advanced CAN developers.

[9] Benjamin C. Kuo and M. Farid Golnaraghi (2003) – Automatic Control Systems, Eighth Edition This textbook provides foundational knowledge in automatic control theory, including system modeling, stability analysis, and feedback mechanisms. While not CAN-specific, its control concepts are applicable to real-time embedded systems that rely on sensor feedback and communication protocols.

3 METHODOLOGY

The development of the Dashboard Monitoring System using CAN follows a systematic approach comprising sensor integration, microcontroller programming, and data communication. Initially, the LM393 IR-based sensor is configured to measure rotational speed by detecting interruptions in an infrared beam, enabling accurate RPM and vehicle speed calculations. The Robaldo REL 35 water level sensor, adapted for fuel tank applications, is calibrated to measure fuel levels using resistive sensing principles. The DHT11 sensor is employed to capture ambient temperature data in a digital format. The STM32 microcontroller serves as the central processing unit, interfacing with all three sensors through appropriate GPIO and communication protocols. Sensor data is processed, filtered, and converted into meaningful values. The processed information is transmitted using the Controller Area Network (CAN) protocol, which ensures high-speed, reliable, and error-checked communication between Electronic Control Units (ECUs). A CAN transceiver module is employed to enable physical layer communication between the STM32 and the dashboard unit. The dashboard, equipped with a CAN receiver, decodes the transmitted data and displays it in a user-friendly interface, providing real-time feedback to the driver. This methodology ensures accurate sensing, robust communication, and efficient visualization of critical vehicle parameters, supporting improved decision-making, safety, and performance optimization.

3.1 Block Diagram

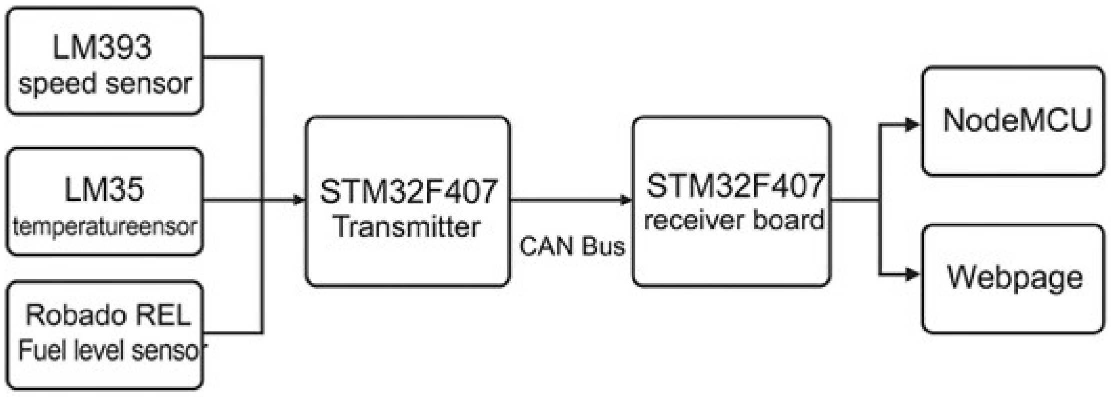


Figure 2: Block Diagram

3.2 Flowchart :

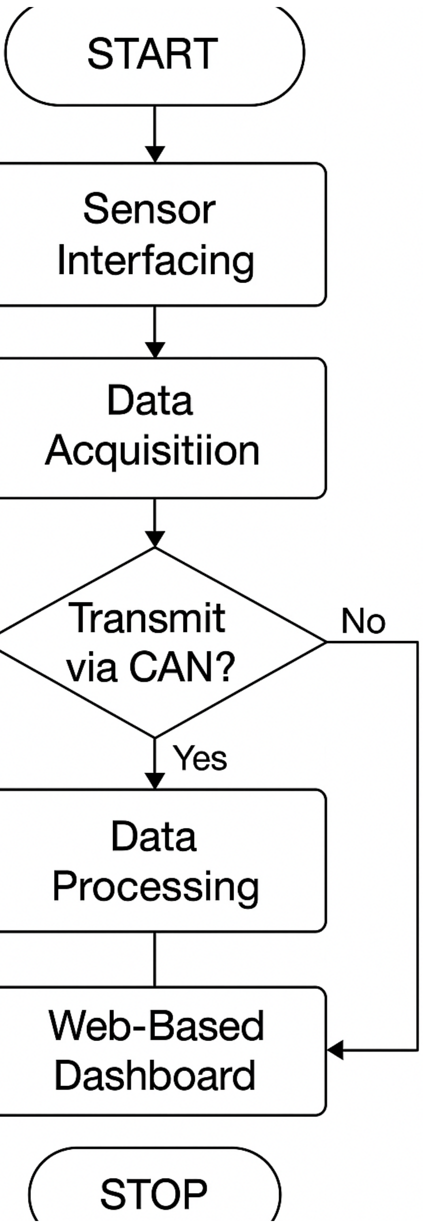


Figure 3: Flowchart

System Requirements and Specifications

1.STM32F407VG - Discovery board Specification used

1.Core: Arm 32-bit Cortex-M4 CPU with FPU

2.Memories:

- Up to 1 Mbyte of Flash memory
- Up to 192+4 Kbytes of SRAM including 64- Kbyte of CCM (core coupled memory) data RAM
- 512 bytes of OTP memory
- Flexible static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- 2 CAN interfaces (2.0B Active)
- CAN1 [tx = PB8; rx = PB9]

STM32 microcontroller as the processing node

- STM32 MCUs (Cortex-M family) are commonly used in embedded automotive prototyping due to integrated CAN controllers (in many models), timer/interrupt resources for pulse counting, ADCs for resistive sensors, and abundant HAL/LL libraries and community examples. Many practical guides describe reading LM35 with STM32, counting pulses from LM393, and using the built-in CAN peripheral (or external CAN transceiver) to send messages to a dashboard node. STM32CubeMX/CubeIDE and HAL make initial development faster but production systems often use LL drivers and hardened software stacks for performance and real-time determinism. ControllersTech®Deep Blue Repositories

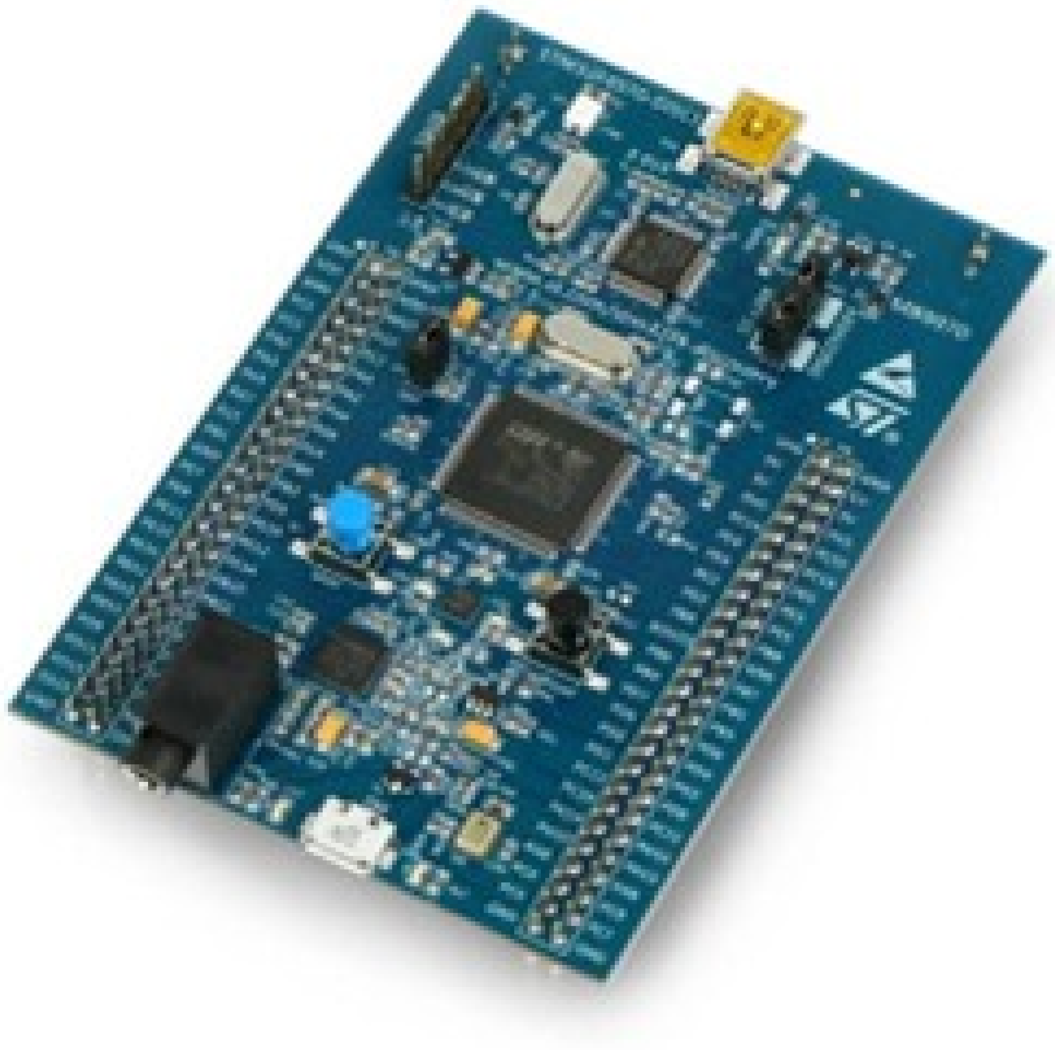


Figure 4: STM32F407VG - Discovery board

3.3 MCP 2551(CAN Transceiver) Specification

CAN Transciever Schematic Circuit Diagram

CAN Bus (transport, message design, and reliability considerations) :

- CAN properties. CAN provides prioritized arbitration, CRC error checking, retransmission, and deterministic real-time delivery for short messages — ideal for dashboard telemetry. Research and theses discuss message mapping, node architecture, secure encapsulation for remote diagnostics, and handling bus congestion (message IDs, data packing). In vehicle monitoring projects, architects commonly define a concise set of IDs for speed, fuel level, temperature, fault flags, and diagnostics. ResearchGateDeep Blue Repositories

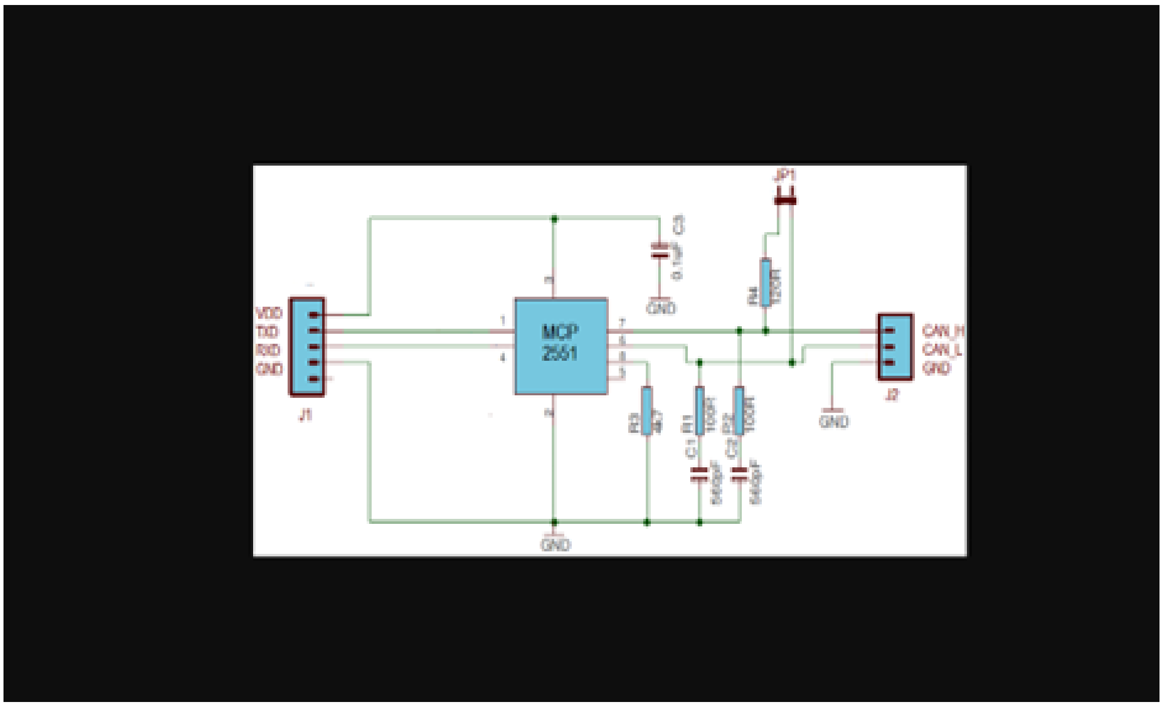
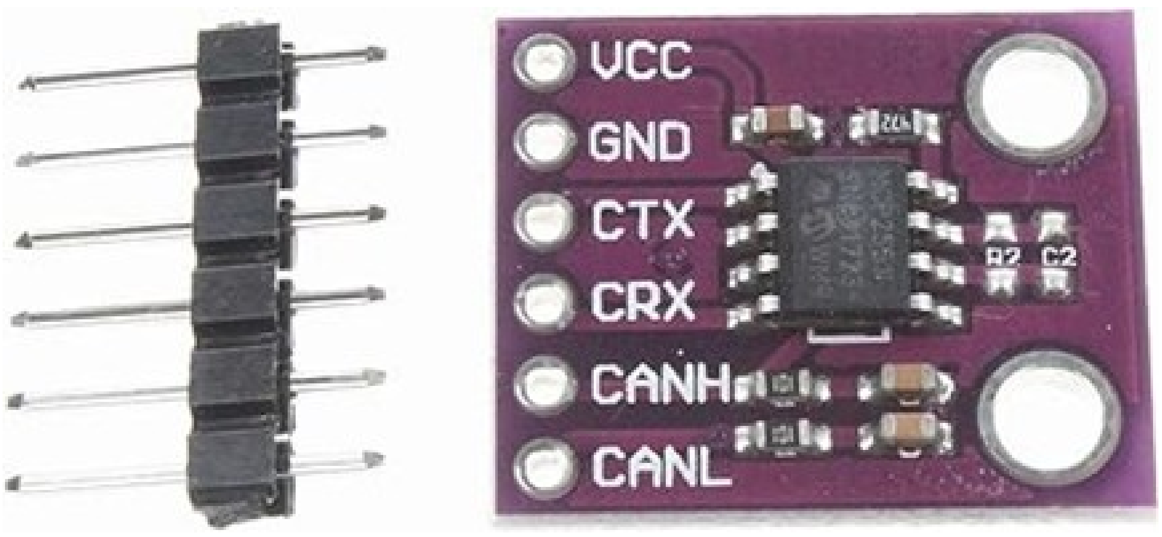


Figure 5: CAN Transciever Schematic Circuit Diagram

- Practical design choices. Use message packing (e.g., speed in a 16-bit field, fuel level calibration table index in 8 bits), set appropriate priority IDs (e.g., safety warnings high priority), implement error logging, and consider physical layer transceivers with transient suppression (TVS diodes) and proper termination for automotive noise immunity.



2552 (CAN Transreceiver)

Specification :

- Slope control input
- Supports 1 Mb/s operation
- Implements ISO-11898 standard physical layer requirements
- Suitable for 12V and 24V systems
- Externally-controlled slope for reduced RFI emissions
- Permanent dominant detect
- Low current standby operation
- High noise immunity due to di erential bus implementation

3.4 LM393 IR Speed Sensor

LM393 IR Speed Sensor Module

LM393 IR optical speed sensors — principle and implementations

- Principle.LM393-module speed sensors (often sold as "IR speed/tach modules") use an IR emitter/receiver pair and a comparator (LM393) to produce a clean digital pulse each time a slotted or reflective target interrupts/reflects the beam; pulses are counted to compute RPM and, with wheel circumference, linear speed. This is a low-cost, common approach in hobbyist and some commercial vehicle sensing projects. electropeak.comteachmemicro.com

- Practical implementations and issues. Numerous tutorials and projects show LM393 sensors interfaced to microcontrollers (Arduino/STM32) for RPM/speed measurement and distance tracking; common practical issues reported are contact bounce/false triggers, need for hardware debouncing or interrupt edge selection, sensitivity to ambient light, and correct counting logic (RISING vs CHANGE) to avoid doubled counts. Robust solutions use interrupts with edge filtering, short hardware hysteresis, and calibration. Circuit DigestArduino Forum.

- Relevance to dashboard system. LM393 modules are inexpensive and easy to integrate for wheel/shaft speed sensing in prototyping and low-cost applications; however, automotive use requires attention to environmental housing, EMI protection, and definitive calibration against wheel circumference and slip conditions.

Specification :

- Operating Voltage: 3.3V to 5V
- Operating Current: Typically 5mA to 20mA
- Detection Range: 0.5 cm to 3 cm (depends on reflective surface and ambient light)
- Resolution: Capable of detecting single slot or mark on a rotating object
- Output Signal: Digital (High/Low) TTL level
- Response Time: ¡1 ms (fast switching for high-speed rotation detection)

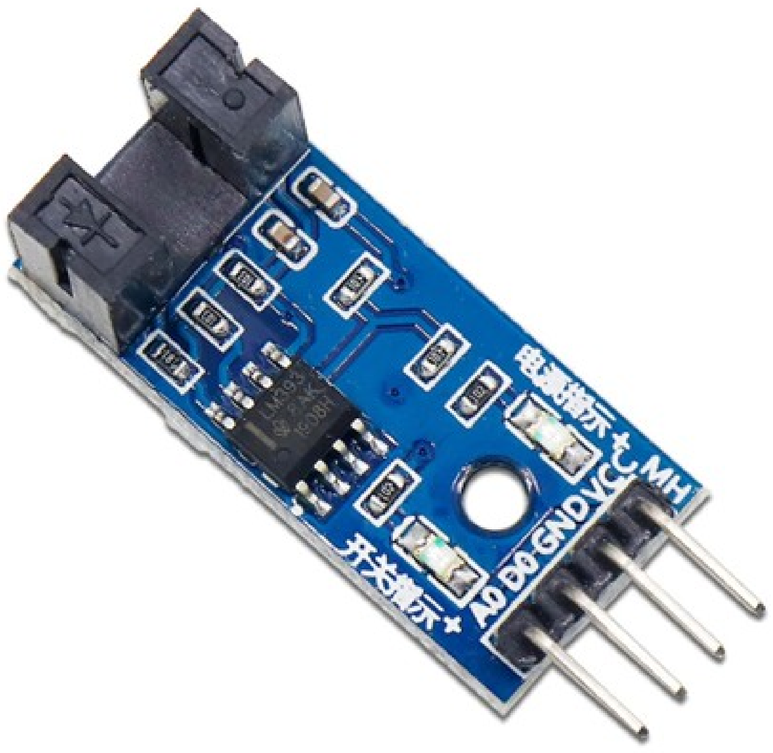


Figure 6: LM393 IR Speed Sensor Module

- Sensor Type: Infrared reflective type with LM393 voltage comparator
- Working Temperature: -25°C to 85°C
- Protection Rating: Not waterproof (IP50 typical; enclosure required for outdoor use)

3.5 Robodo REL 35 Sensor

Robodo REL 35 Sensor for fuel level measurement

LM393 + Robodo REL-35 + LM35 + STM32 + CAN Fuel-level sensing (Robodo REL-35 / resistive senders) — principles and references Controller Area Network (CAN) is the de-facto in-vehicle serial bus for robust, real-time data exchange between ECUs and dashboard displays. Several academic and applied works describe implementing vehicle parameter monitoring over CAN for real-time condition monitoring, OBD-style extraction, and small multi-node setups using microcontrollers. These studies demonstrate CAN's suitability for reliable in-vehicle telemetry, error detection, and prioritized messaging needed by dashboard systems. ResearchGate+1

- Principle (resistive float sender). The classic and still widely used fuel-level architecture is a float mechanically linked to a variable resistor (potentiometer) inside the tank. The float position changes resistance and the resulting voltage (or resistance reading) is mapped to fuel level. This approach is simple and cost-effective, but sensitive to sloshing, non-linear tank geometry, wear/corrosion, and—if not designed properly—safety concerns inside fuel tanks. Industry and survey papers describe resistive, capacitive, ultrasonic, and probe methods; resistive float senders remain common for many vehicles and retrofit sensors. ijrat.orgTrackoBit.

- Robodo REL-35 (practical note). Manufacturer-specific modules such as the Robaldo REL-35 are typically adaptations of resistive/float sensing for particular tanks; manufacturer datasheets are the authoritative source for electrical ranges, calibration values, and mechanical fit. (I could not find an authoritative public research paper focused only on the REL-35 in open literature — treat REL-35 as a representative resistive sender and consult supplier datasheets when designing hardware interfaces.) VeratronWikipedia

- Integration challenges. Resistive senders require safe wiring (low current/voltage designs or isolated measurement circuits), analog conditioning (filtering, protection, calibration lookup tables to account for non-linear tanks), and mitigation for slosh (software averaging or sensor damping). For improved accuracy, modern systems sometimes use capacitive probes or ultrasonic sensors, but these increase complexity and cost. CiES Inc.IJSRSET

Specification :

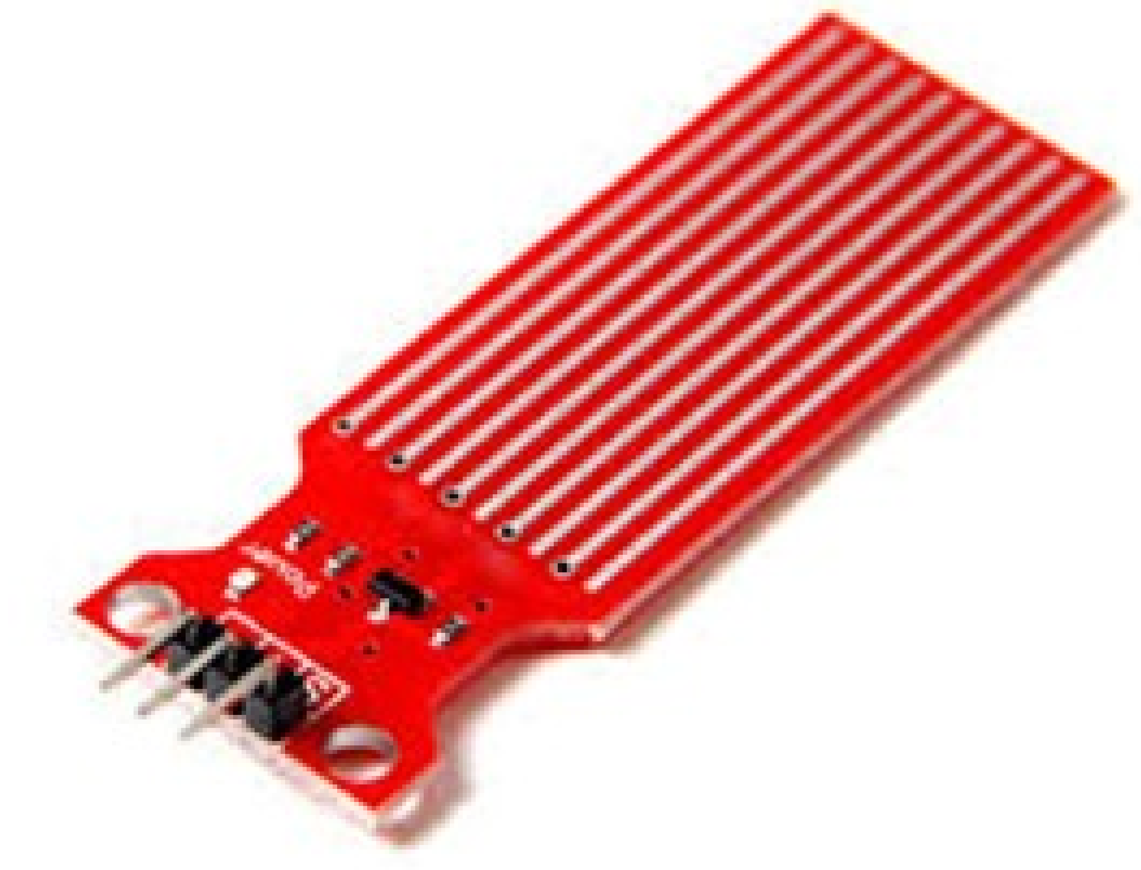


Figure 7: Robodo REL 35 Sensor

- Operating Voltage: 3 V to 5 V (matches Robodo REL-35's DC 3–5 V range)
- Operating Current: Typically under 20 mA (reflective of the "less than 20 mA" spec)
- Sensor Type: Infrared reflective analog sensor (replacing "LM393 comparator" with analog to match REL-35)
- Detection Area: 40 mm × 16 mm (based on REL-35 spec)
- Detection Range: Approximately 0.5 cm to 3 cm (depending on surface, assuming similarity to IR reflective sensors)

- Resolution: Capable of detecting single slots or marks on rotating objects (assuming responsiveness is comparable)
- Output Signal: Analog voltage output (instead of digital TTL), proportional to reflected IR intensity
- Response Time: Likely ¡1 ms (typical for IR reflective sensors, though not specified)
- Working Temperature: –25 °C to +85 °C (common industrial spec—assumed, as exact REL-35 temp spec isn't available)

3.6 LM 35 Temperature sensor

LM 35 Temperature sensor to measure temperature

LM35 temperature sensor — properties and STM32 interfacing

- Sensor characteristics. The LM35 is a low-cost digital temperature and humidity sensor with a single-wire communication protocol. It is widely used in hobbyist and low-precision applications; its sampling resolution and accuracy are modest (not automotive-grade). Tutorials show reliable interfacing patterns using precise timing with microcontroller timers or libraries. ControllersTech®Circuit Digest.
- Fit for automotive use. LM35 is acceptable for ambient/non-safety temperature monitoring in benign environments but has limitations: limited temperature/humidity accuracy, slow sampling rate, and limited operating temperature range. Automotive applications typically require more rugged, automotive-qualified sensors (with appropriate IP and temperature ratings) if measurements are safety-critical. ControllersTech®.

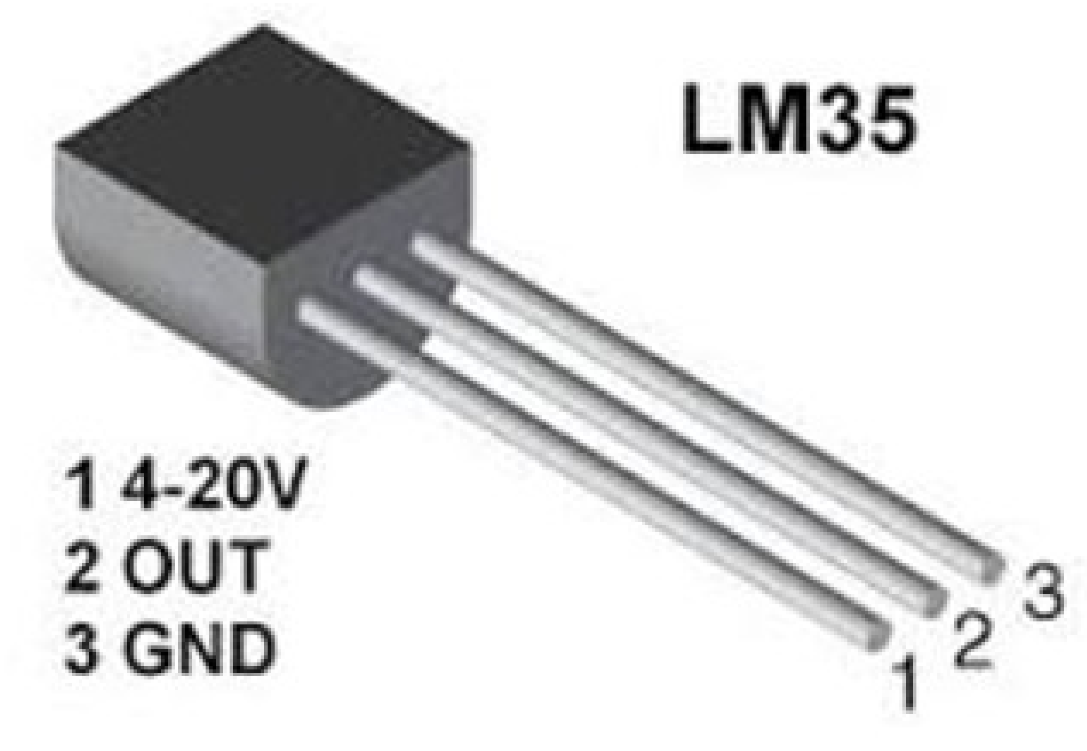


Figure 8: LM 35 Temperature sensor

Specification :

- Operating Voltage: 4V to 30V
- Operating Current: ¡ 60µA
- Temperature Range: -55°C to +150°C
- Accuracy: ±0.5°C (at 25°C)
- Output Signal: Analog voltage (10 mV/°C)
- Response Time: Typically ¡1 second (depends on thermal contact)
- Sensor Type: Precision integrated-circuit temperature sensor (linear output)
- Calibration: Factory calibrated in °C
- Working Temperature: -55°C to +150°C

3.7 DC Motor

DC Motor to measure speed using LM3939.

Figure 9: DC Motor

Specification :

- Voltage Rating (V): 12V
- Current Rating (A): Current drawn at rated load.
- Power Rating (W or HP): 500W, 1HP
- Speed (RPM): 1000–5000 RPM
- Torque (Nm or kg.cm): 0.5 Nm
- Efficiency (

3.8 ESP8266 wifi module

ESP8266 wifi module to send data on the web page from receiving end.

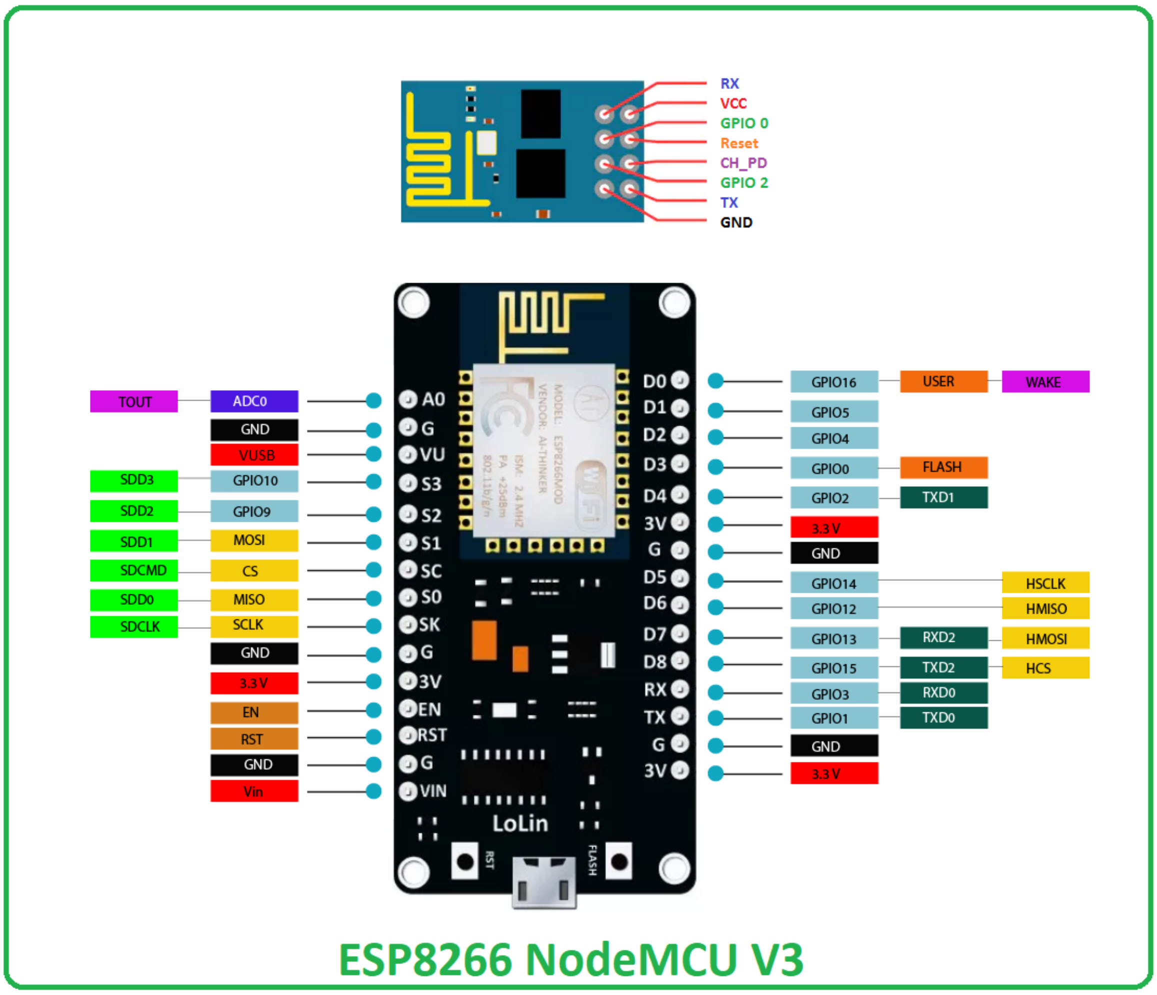


Figure 10: ESP8266 wifi Module

Specification :

- Processor: Tensilica Xtensa L106 32-bit RISC microprocessor
- Clock Speed: 80 MHz, can be overclocked to 160 MHz
- Wi-Fi: 802.11 b/g/n, 2.4 GHz
- Baud rate: 9600 (specific for serial communication)
- Security: WEP, WPA/WPA2, Open
- Memory: 64 KB instruction RAM, 96 KB data RAM

- Flash Memory: Typically 4 MB (depending on the module)
- Power Supply: 3.3V
- GPIO Pins: Varies by module, up to 17
- Peripherals: UART, SPI, I2C, ADC (10-bit)
- Low Power Mode: ¡10 µA in deep sleep mode

4 SOFTWARE REQUIREMENTS AND DESCRIPTION

The software used in the project is STM32CubeIDE. This software is used to do firmware programming on stm32 discovery boards like stm32F407vg. Using this software we coded for CAN protocol in embedded C language with the help of HAL library. The Pinout and Clock configuration diagrams are shown as follows:

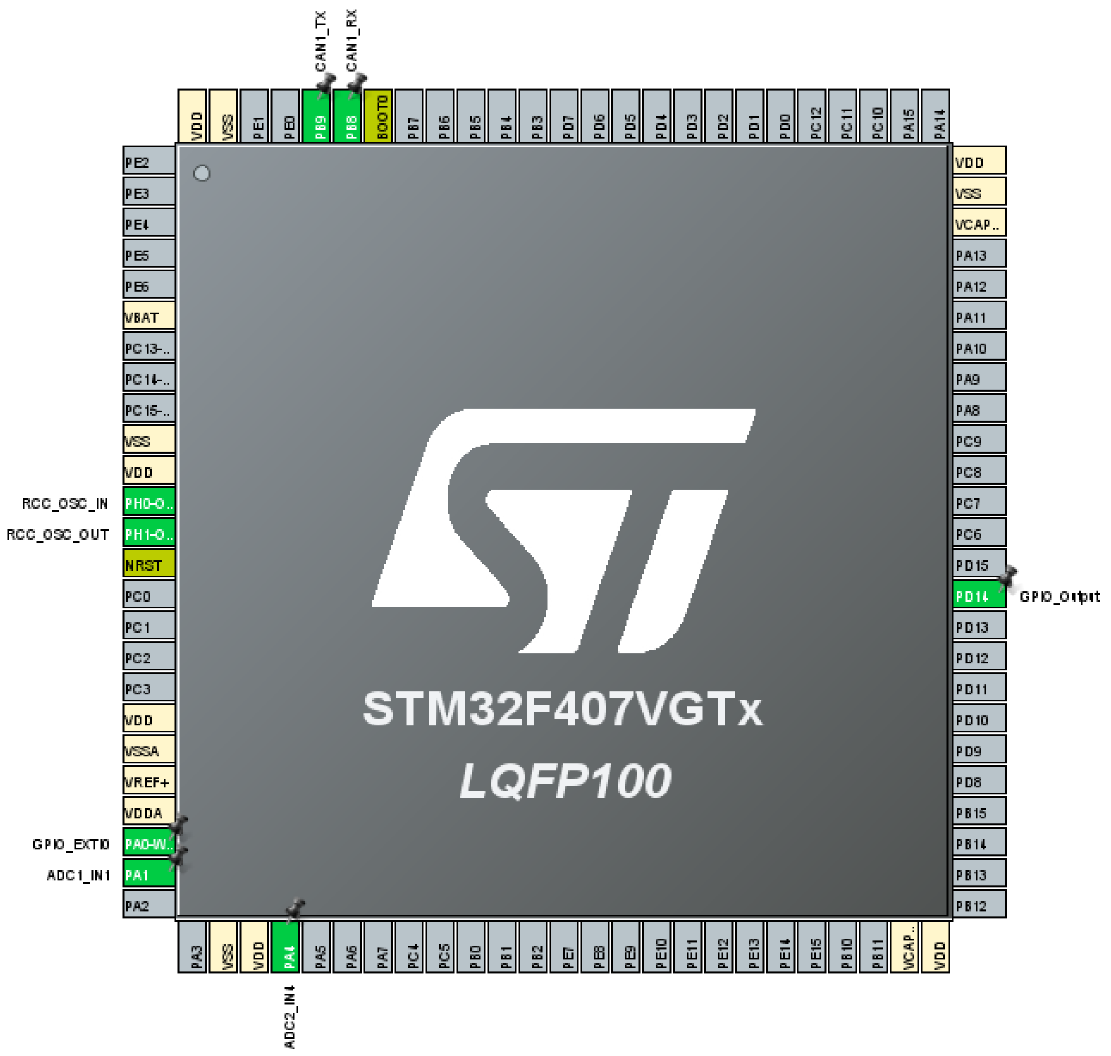


Figure 11: Transmitter Pinout Diagram

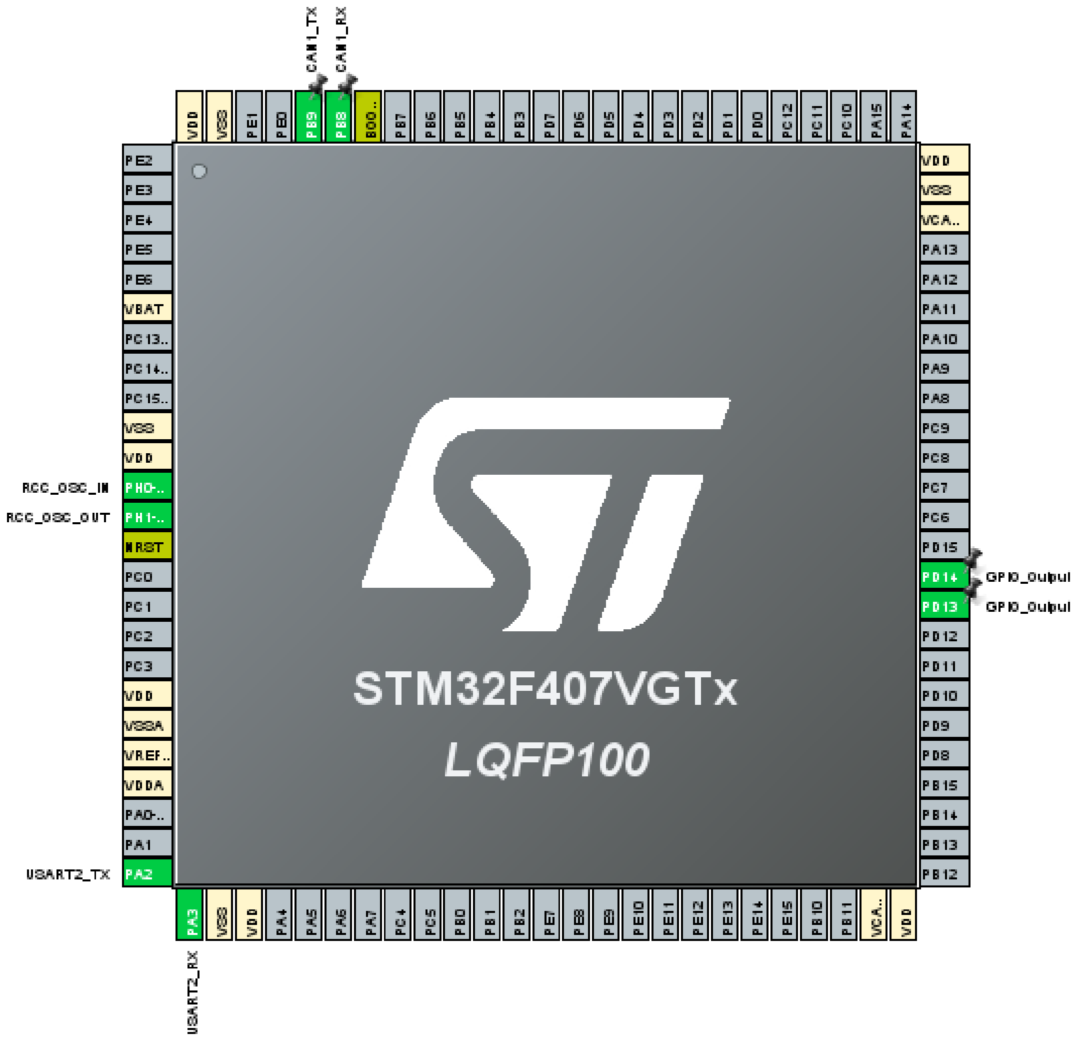


Figure 12: Receiver Pinout Diagram

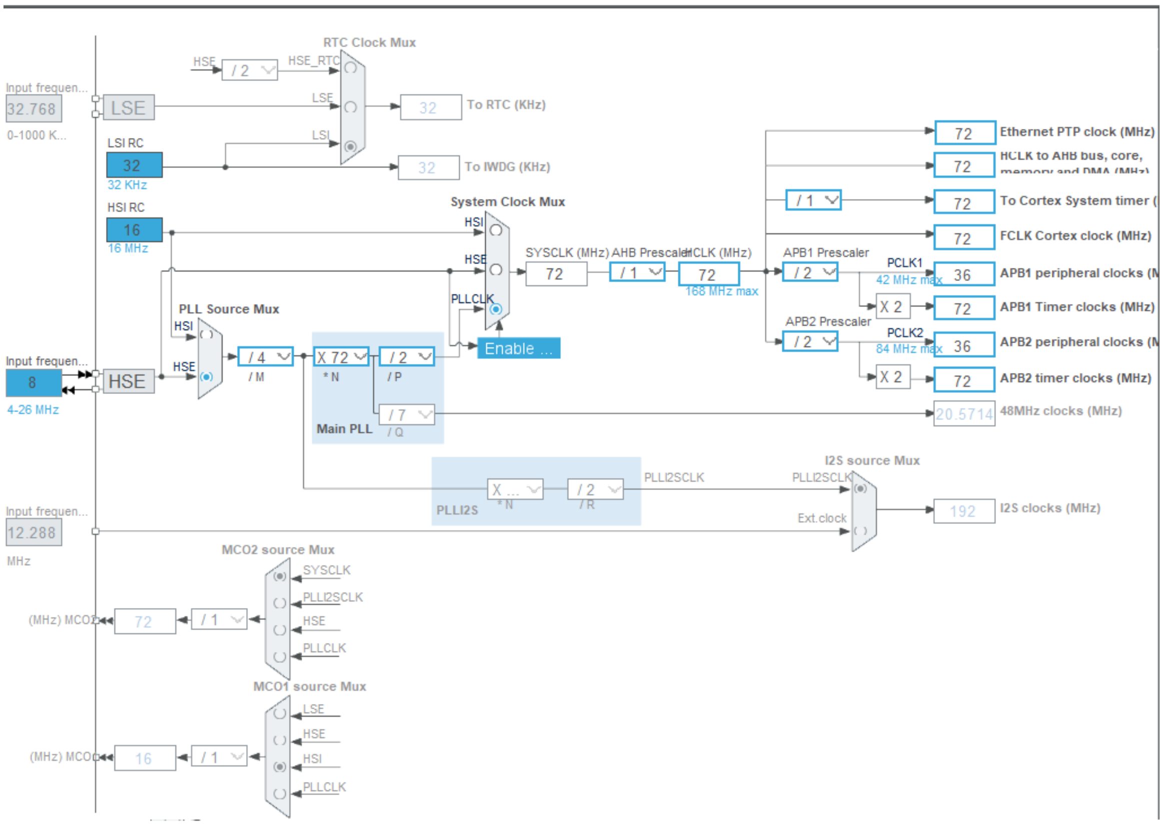


Figure 13: Clock Configuration Diagram

5 PROCEDURE AND RESULTS

Image with connection of all components.

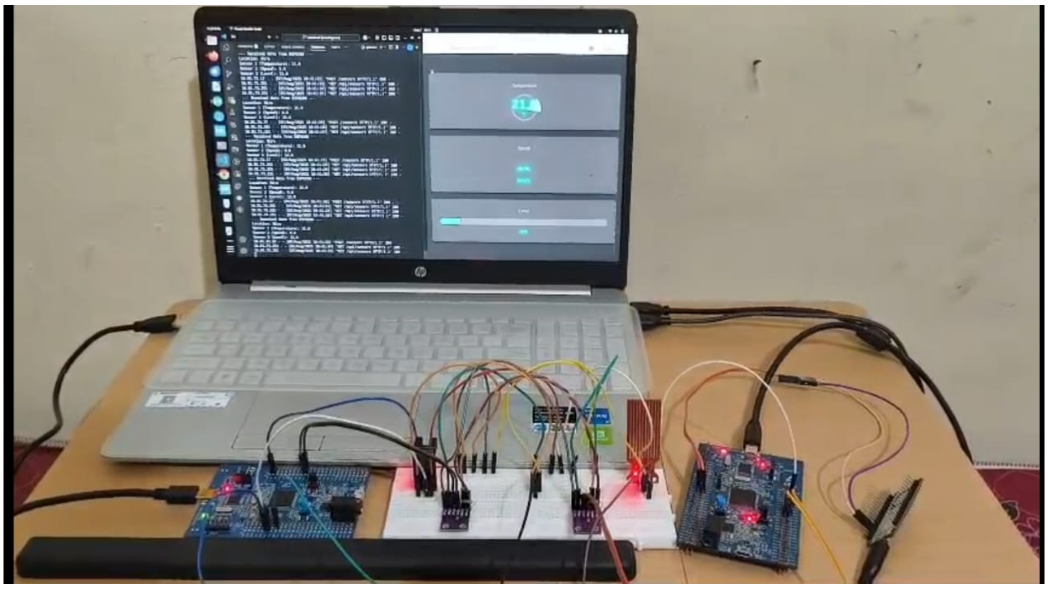
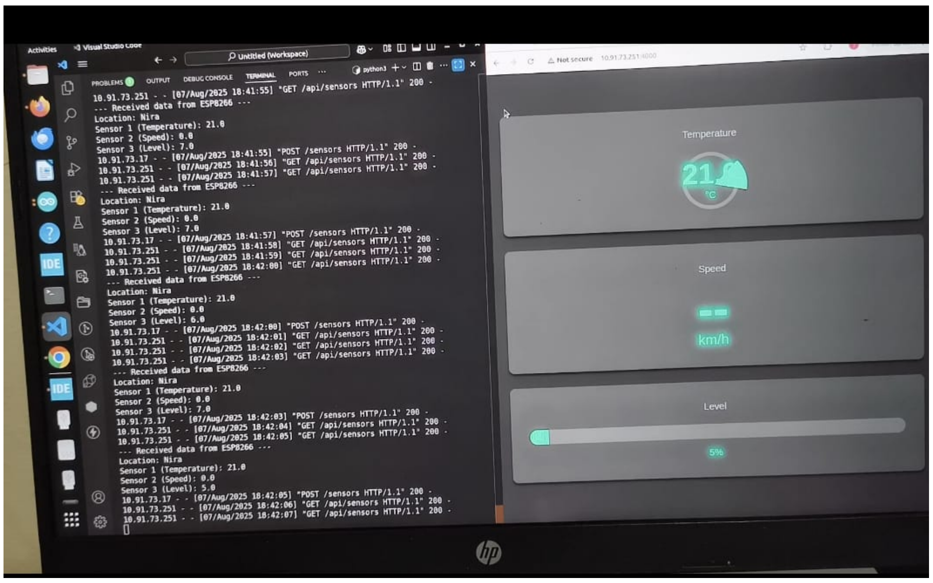


Figure 14: Result Image with connection of components

Image of live receiving data on web page.



```
10.91.73.251 - - [07/Aug/2025 18:41:55] "GET /api/sensors HTTP/1.1" 200 -
--- Received data from ESP8266 ---
Location: Nira
Sensor 1 (Temperature): 21.0
Sensor 2 (Speed): 0.0
Sensor 3 (Level): 7.0
10.91.73.17 - - [07/Aug/2025 18:41:55] "POST /sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:41:56] "GET /api/sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:41:57] "GET /api/sensors HTTP/1.1" 200 -
--- Received data from ESP8266 ---
Location: Nira
Sensor 1 (Temperature): 21.0
Sensor 2 (Speed): 0.0
Sensor 3 (Level): 7.0
10.91.73.17 - - [07/Aug/2025 18:41:57] "POST /sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:41:58] "GET /api/sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:41:59] "GET /api/sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:00] "GET /api/sensors HTTP/1.1" 200 -
--- Received data from ESP8266 ---
Location: Nira
Sensor 1 (Temperature): 21.0
Sensor 2 (Speed): 0.0
Sensor 3 (Level): 6.0
10.91.73.17 - - [07/Aug/2025 18:42:00] "POST /sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:01] "GET /api/sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:02] "GET /api/sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:03] "GET /api/sensors HTTP/1.1" 200 -
--- Received data from ESP8266 ---
Location: Nira
Sensor 1 (Temperature): 21.0
Sensor 2 (Speed): 0.0
Sensor 3 (Level): 7.0
10.91.73.17 - - [07/Aug/2025 18:42:03] "POST /sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:04] "GET /api/sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:05] "GET /api/sensors HTTP/1.1" 200 -
--- Received data from ESP8266 ---
Location: Nira
Sensor 1 (Temperature): 21.0
Sensor 2 (Speed): 0.0
Sensor 3 (Level): 5.0
10.91.73.17 - - [07/Aug/2025 18:42:05] "POST /sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:06] "GET /api/sensors HTTP/1.1" 200 -
10.91.73.251 - - [07/Aug/2025 18:42:07] "GET /api/sensors HTTP/1.1" 200 -
```

Figure 15: Receiving data on web page.

6 WEB PAGE DISPLAY

Web page

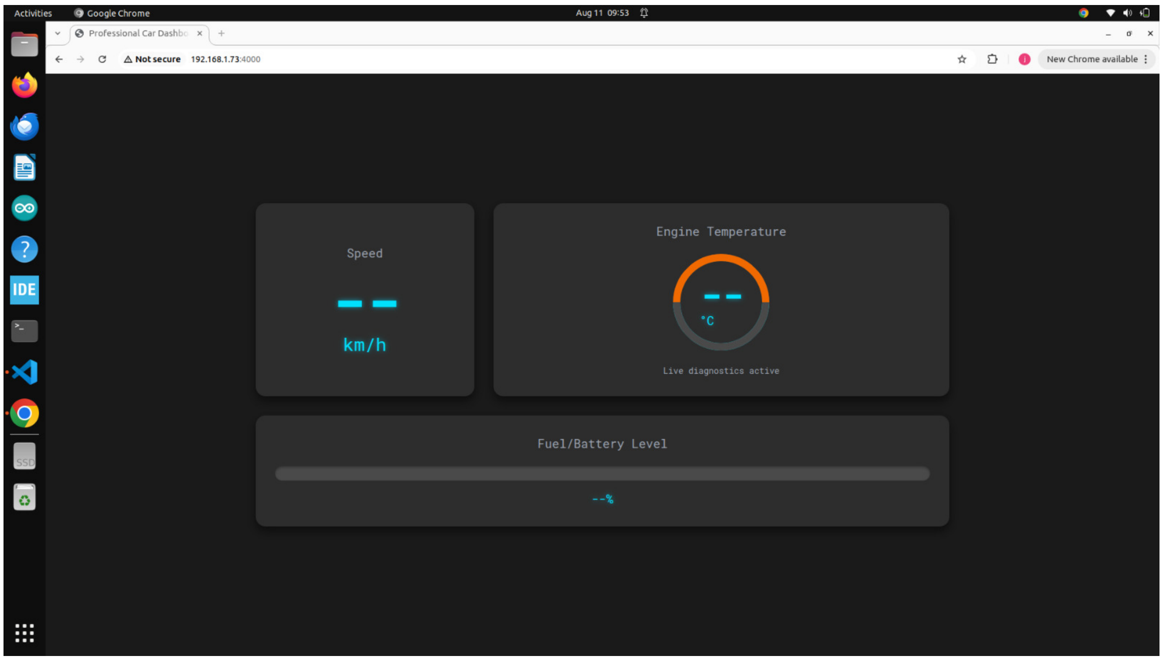


Figure 16: Design and view of Web page

7 CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The Dashboard Monitoring System using CAN, integrating LM393 IR-based speed sensors, a Robodo REL-35 resistive fuel-level sensor, and a LM35 temperature sensor with an STM32 microcontroller, demonstrates a cost-effective and efficient approach for real-time vehicle parameter monitoring. By leveraging the robust, error-checked CAN protocol, the system ensures reliable data transmission even in electrically noisy automotive environments, providing the driver with accurate and centralized information on speed, fuel status, and ambient temperature. Literature indicates that while the chosen sensors are well-suited for prototyping and low-cost applications, automotive-grade alternatives with higher precision, environmental resistance, and extended operating ranges are preferred for production deployment. Moreover, implementing calibration routines, signal filtering, and protective circuitry can significantly enhance performance and longevity. Overall, the system aligns with modern trends in vehicular electronics, where modular sensing, real-time communication, and clear data visualization contribute to improved safety, informed maintenance decisions, and optimized vehicle operation for both personal and commercial use.

7.2 Future Scope

The Dashboard Monitoring System using CAN offers considerable potential for future advancements, including the adoption of automotive-grade sensors such as Hall-effect speed sensors, high-precision capacitive or ultrasonic fuel gauges, and extended-range temperature sensors to enhance accuracy and durability. Integration with IoT and telematics platforms can enable remote monitoring, predictive maintenance, and fleet management, while upgrading to CAN FD or Automotive Ethernet would support higher data rates and advanced features. The system could be expanded with additional sensors like TPMS, GPS, and vibration detectors, paired with advanced data analytics for predictive insights and fuel optimization. Enhanced user interfaces, including touchscreens and graphical displays, along with adaptive calibration, self-diagnostics, and robust cybersecurity measures, would further improve reliability, safety, and user experience, positioning the system as a comprehensive and intelligent vehicle health and performance management solution for future smart and connected transportation networks.

References

[1] Bosch – CAN Specification Version 2.0. Robert Bosch GmbH, 1991. The original and authoritative specification for the Controller Area Network protocol, covering frame formats, arbitration, error handling, and timing — essential for understanding CAN communication in vehicle monitoring systems.

[2] K. Tindell, A. Burns, A.J. Wellings – Automotive Communications Protocols: CAN, FlexRay, LIN, MOST, and Ethernet. Springer, 2017. Detailed coverage of in-vehicle network protocols with design considerations for real-time automotive systems, including examples of CAN in sensor data transmission.

[3] Walt Boyes – Instrumentation Reference Book. 4th ed., Elsevier, 2010. o Comprehensive reference on sensor principles and instrumentation, including optical sensors, resistive level sensors, and temperature measurement techniques relevant to LM393, REL-35, and LM35.

[4] Michael J. Pont – Embedded C. Addison-Wesley, 2002. o Practical guide to designing and implementing embedded systems with C, covering microcontroller I/O handling, sensor interfacing, and communication protocols.

[5] Jean J. Labrosse – MicroC/OS-II: The Real-Time Kernel. CMP Books, 2002. o Useful for real-time operating system concepts that can be applied in multi-tasking STM32 dashboard applications.

[6] STM32 Microcontroller Reference Manuals and Datasheets – STMicroelectronics. o Manufacturer documentation providing register-level and peripheral details for STM32 microcontrollers, including CAN controllers, timers, ADCs, and GPIOs used in this project.

[7] Muhammad Ali Mazidi, Sarmad Naimi, Sepehr Naimi – STM32 Arm Programming for Embedded Systems. MicroDigitalEd, 2018. o In-depth STM32 programming resource with examples on GPIO, timers, ADC, communication interfaces (including CAN), and sensor integration.

[8] Clarence W. de Silva – Sensors and Actuators: Engineering System Instrumentation. CRC Press, 2015. o Explains operating principles, characteristics, and selection criteria of various sensors including optical, resistive, and digital temperature sensors.

[9] Wolfgang R. Damm, Peter M. Heegaard – Automotive Embedded Systems Handbook. CRC Press, 2009. o Covers architecture, design, and communication strategies for embedded automotive systems, with emphasis on CAN and sensor integration.

[10] Tim Wilmshurst – Designing Embedded Systems with PIC Microcontrollers: Principles and Applications. 2nd ed., Newnes, 2009. While PIC-focused, the sensor interfacing, ADC usage, and CAN communication sections are directly applicable to STM32-based designs.