Project Report

On

CNG Gas Leakage Detection Using CAN

Submitted

In partial fulfilment

For the award of the Degree of

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

C-DAC, Sunbeam Institute Of Information Technology,

Pune

Submitted By:

240844230050 Pankaj Ghorpade

240844230097 Vrushabh Urkudkar

240844230096 Vrishal Chandawarkar

240844230060 Pradyumna Jadhav

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, Mr. Vrushabh Patil , Sunbeam, Pune for his constant guidance and helpful suggestion for preparing this project CNG Leakage Detection using CAN Interfacing. We express our deep gratitude towards her for inspiration, personal involvement, constructive criticism that he provided us along with technical guidance during this project.

It is our great pleasure in expressing sincere and deep gratitude towards our Technical Director Mr. Nilesh Ghule for his valuable guidance and constant support throughout this work and help to pursue additional studies.

We take this opportunity to thank our Course Co-ordinator Mr. Devendra Dhande for providing us such a great infrastructure and environment for our overall development. We express sincere thanks to Mr. Ankush Tembhurnikar and Ms. Netra Shirke for their kind cooperation and extendible support towards the completion of our project.

Also, our warm thanks to Sunbeam Pune, which provided us this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

240844230050 Pankaj Ghorpade

240844230097 Vrushabh Urkudkar

240844230096 Vrishal Chandawarkar

240844230060 Pradyumna Jadhav

(Pune- 411008)

----------ABSTRACT----------

The CNG Gas Leakage Detection System Using CAN is a smart and efficient safety solution that integrates Controller Area Network (CAN) technology with Internet of Things (IoT) capabilities to provide real-time monitoring and quick response to gas leaks. This system is designed to enhance safety in industrial, residential, and automotive environments by detecting gas leaks with high accuracy and triggering immediate alerts and preventive actions.

The system consists of an ESP8266 microcontroller, MQ-9 gas sensor, flame sensor, MCP2551 CAN transceiver, GSM800L module, piezo buzzer, servo motor, LED indicators, and a relay module. The MQ-9 sensor continuously monitors gas levels, while the flame sensor detects fire hazards. Upon detecting a leak, the system activates alarms, engages safety mechanisms, and sends real-time alerts via IoT platforms like Blynk and Thingspeak. Additionally, the GSM800L module enables SMS-based alert notifications for instant communication in emergency situations. The MCP2551 CAN transceiver ensures reliable and interference-free communication between components, allowing for fast and efficient data transmission.

By combining IoT-based remote monitoring with high-speed CAN communication and GSM-based emergency alerts, this project offers a secure, scalable, and effective gas leakage detection system. It can be deployed in industrial plants, smart homes, CNG-powered vehicles, and smart city infrastructure to prevent potential hazards and ensure public safety.

This project aims to revolutionize gas leakage detection by providing real-time alerts, automated safety measures, and seamless IoT integration, making environments safer and more secure.

.----------

Contents

1.Introduction

This project presents the development of a CNG Gas Leakage Detection System Using CAN, designed for real-time monitoring and rapid response to gas leaks. By leveraging the Controller Area Network (CAN) protocol, known for its reliable and interference-free communication, alongside a variety of sensors, the system ensures efficient gas leak detection and safety mechanisms. The integration of IoT technologies and GSM-based alerts further enhances the system's capability to provide instant notifications and preventive actions, making it a robust and secure solution for industrial, residential, and automotive applications.

1.1. History :

1.1.1. Early Efforts

- Initial gas detection systems were basic, relying on wired sensors with limited communication capabilities.
- Traditional gas leak detection methods required manual inspections, making early detection difficult.
- The need for automated, real-time detection systems led to research in sensor-based monitoring solutions.

1.1.2. Advancements and Expansion

- The development of wireless gas sensors improved flexibility and scalability, reducing the dependency on manual monitoring.
- The MQ-9 gas sensor was introduced, offering high sensitivity to carbon monoxide, methane, and liquefied petroleum gases.
- The adoption of the CAN protocol provided real-time, robust, and interference-free communication between sensors and controllers.
- IoT integration enabled remote monitoring and data visualization on platforms like Blynk and Thingspeak.

1.1.3. Mainstream Adoption

- Automated gas leakage detection systems became widely used in industries, homes, and vehicles to ensure safety.
- The combination of sensor networks with IoT and GSM-based alerts enhanced real-time notification and smart industry applications.
- CAN-based networks expanded into automotive safety, smart cities, and industrial plants, making gas leakage detection more efficient and accessible.

- CAN-based networks expanded into smart cities and infrastructure, becoming vital for modern monitoring systems.

1.2. Problem Statement

Current gas leakage detection systems often struggle to provide real-time, accurate data across various environments, leading to:

- Inefficient monitoring: Gas leaks may go undetected due to delayed or unreliable data transmission, increasing the risk of hazardous incidents.
- High operational risks: Lack of real-time insights prevents immediate corrective actions, raising the likelihood of accidents and potential disasters.
- Limited scalability: Traditional gas detection systems are often expensive and difficult to integrate into larger industrial, residential, or automotive applications.
- Inadequate data for decision-makers: Fragmented or outdated gas level readings hinder proactive safety measures and efficient management of gas-related risks.

1.3. Objective and Specification

- **Flexibility**:

The modular design of the CNG gas leakage detection system allows for easy integration across industrial, residential, and automotive environments. This flexibility ensures adaptability to various setups and monitoring requirements**.**

- **Cost-Effectiveness**:

The system utilizes cost-efficient components, including the MQ-9 gas sensor and ESP8266, making it an affordable alternative to traditional gas detection solutions. Its modular nature allows for redeployment and expansion without significant additional costs.

- **Ease of Installation**:

The sensor modules and communication units are designed for simple and quick installation, reducing downtime and minimizing labor costs in industrial and automotive environments.

- **Scalability**:

The CAN-based architecture allows seamless expansion by integrating additional sensors or extending the network to cover larger areas, making it suitable for growing industries or smart infrastructure projects.

- **Data Collection**:

The system continuously gathers real-time data on gas concentration and fire hazards, ensuring proactive safety measures. This data is processed and displayed through IoT platforms like Blynk and Thingspeak for easy monitoring.

- **Remote Access**:

IoT integration enables facility managers and vehicle operators to access real-time gas level data remotely. In case of a leak, alerts are sent via GSM800L, allowing for immediate action, even from distant locations.

- **Reliability:**

The CAN protocol ensures robust and interference-free communication between sensors and controllers, providing continuous and accurate data transmission even in harsh industrial conditions. This reliability is crucial for maintaining consistent monitoring and preventing hazardous situations.

.

2. Literature Review

- Researchers have extensively explored the integration of sensor networks and CAN protocols for safety and environmental monitoring, a field that has gained significant attention in industrial, automotive, and residential sectors.

- Sharma, Patel, and Desai (2016) examined the application of CAN-based sensor networks for gas leakage detection in industrial plants. Their study demonstrated the effectiveness of these systems in improving workplace safety by providing real-time gas concentration monitoring and automated emergency responses. The authors emphasized the role of CAN in ensuring reliable communication in harsh industrial environments.

- Gupta, Reddy, and Mehta (2017) analyzed the scalability of IoT-integrated gas detection systems in industrial and residential settings. Their research focused on how modular sensor systems, like those utilizing MQ-9 gas sensors, can be easily scaled to cover larger areas, improving safety and hazard prevention strategies.

- Kumar, Singh, and Iyer (2018) presented a comprehensive review of remote gas leakage monitoring systems utilizing CAN-based sensor networks. Their study explored integration with cloud-based IoT platforms for real-time data access and emergency alerts. The authors illustrated how remote monitoring enhances response times and minimizes risks by providing instant notifications through GSM-based messaging systems.

- Verma, Choudhary, and Das (2019) investigated the role of sensor networks in predictive safety management for industrial and automotive applications. Their research demonstrated how CAN-based networks could collect and analyze gas leakage data, enabling proactive safety measures and reducing accident risks. The authors highlighted the cost savings and operational efficiency gained through real-time alerts and automated safety mechanisms.

- Bhattacharya, Nair, and Kulkarni (2020) studied the security challenges of CAN-based gas detection systems in industrial environments. Their study focused on potential vulnerabilities to cyber-attacks and emphasized the importance of implementing robust encryption and authentication protocols to ensure secure data transmission.

2.1. Inferences drawn from Literature Review

- CAN-based sensor networks, with their reliability, scalability, and real-time data capabilities, are transforming gas leakage detection and safety management. Their impact extends beyond basic monitoring, influencing predictive hazard prevention, remote operations, and overall operational efficiency.

- **Enhancing Safety and Hazard Prevention**:

Studies like Sharma et al. (2016) highlight how real-time data from CAN-based networks improve safety by providing instant alerts and activating emergency protocols in case of gas leaks. This capability extends to broader industrial, residential, and automotive applications, ensuring proactive safety measures.

- **Advancing Predictive Safety Management**:

As demonstrated by Verma et al. (2019), CAN-based gas leakage detection systems enable predictive safety management by continuously monitoring gas concentrations and triggering early warnings. Future advancements could further integrate AI-based analytics for more precise and automated safety interventions.

- **Improving Security and Data Integrity**:

Research by Bhattacharya et al. (2020) emphasizes the importance of securing CAN-based gas detection networks against cyber threats. As these networks become more integrated into critical safety systems, implementing strong encryption and authentication measures is crucial to ensuring data reliability and system security.

3. Methodology

3.1 Hardware Setup

- Utilize the ESP8266 microcontroller to manage data collection and processing.
- Connect MQ-9 gas sensor, flame sensor, and motion sensor to the ESP8266 board for data acquisition.
- Integrate supporting components, including a piezo buzzer (12V), relay (12V), LED indicators (green and red), and a servo motor (SG90) for emergency response actions.
- Use MCP2551 as the CAN transceiver for reliable communication between modules.
- Ensure proper connectivity and configuration of all hardware components for seamless operation..

3.2 Sensor Data Acquisition

- Implement sensor data acquisition routines on the ESP8266 board.
- Configure the ESP8266 board to read real-time data from the MQ-9 gas sensor, flame sensor, and motion sensor.
- Continuously monitor sensor readings to detect CNG leakage and fire hazards promptly..

3.3 Data Transmission

- Establish communication between the ESP8266 and other modules using the CAN protocol via MCP2551.

- Utilize CAN communication to transmit sensor data efficiently, ensuring low-latency and real-time monitoring.
- Ensure stable data transfer between ESP8266 and other monitoring units for accurate hazard detection..

3.4. Alert Mechanism

- Implement an automated alert system that triggers a buzzer and LED indicators upon detecting gas leakage or fire.
- Use a relay to activate ventilation or safety mechanisms if dangerous gas levels are detected.
- Control a servo motor (SG90) to close the gas valve in case of a critical leakage scenario.
- Send emergency alert messages using GSM800L, notifying designated personnel for immediate action.

3.5 Data Processing and Transmission to Cloud

- Implement firmware on the ESP8266 module to process and analyze sensor data.
- Develop algorithms to filter and validate sensor readings, reducing false alarms.
- Utilize Wi-Fi connectivity on the ESP8266 to establish a connection with IoT platforms.
- Upload processed data to cloud platforms like Blynk or Thingspeak for real-time monitoring and analysis.
- Ensure data integrity and security during transmission to maintain reliability and confidentiality.

3.6 Cloud-based Data Processing:

- Scalability: Cloud-based platforms provide flexible resource management based on data volume.
- Remote monitoring: Real-time data access enables monitoring from anywhere, ensuring quick responses to potential hazards.

Reliability:
Cloud platforms offer high availability and redundancy, minimizing risks of data loss or system failures.

- **Security**:
  Encryption and authentication mechanisms protect sensor data from cyber threats.

- **Automatic updates**:
  Ensures continuous operation with minimal manual intervention.

- **Advanced analytics**:
  Cloud integration supports data analytics and predictive safety mechanisms.

- **Disaster recovery**:
  Provides data backup and retrieval in case of system failures or unforeseen incidents..

3.1. Block Diagram

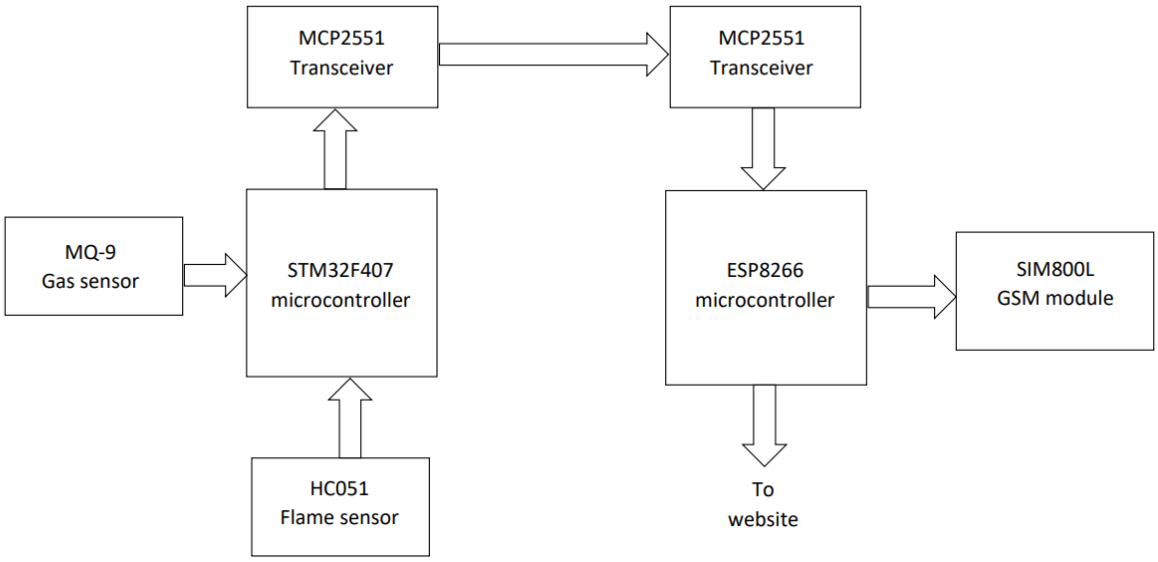


Figure: Block Diagram for proposed System

- Core Processing: The STM32F4 microcontroller serves as the central unit, managing sensor data acquisition and communication.
- Communication Interfaces: Utilizes both IoT (via ESP8266 Wi-Fi board) for data transmission and CAN protocol for robust, real-time communication.
- Temperature Sensor (BMP180): Measures atmospheric temperature and pressure, crucial for environmental monitoring.
- Gas Sensor: Detects harmful gases, enhancing safety in the monitored environment.
- Motion Sensor: Captures movement, providing data for security or activity tracking.
- Heat Flame Sensor: Monitors for the presence of flames, essential for fire detection

4. Proposed System

4.1. STM32F407VGT6 Pin Configuration

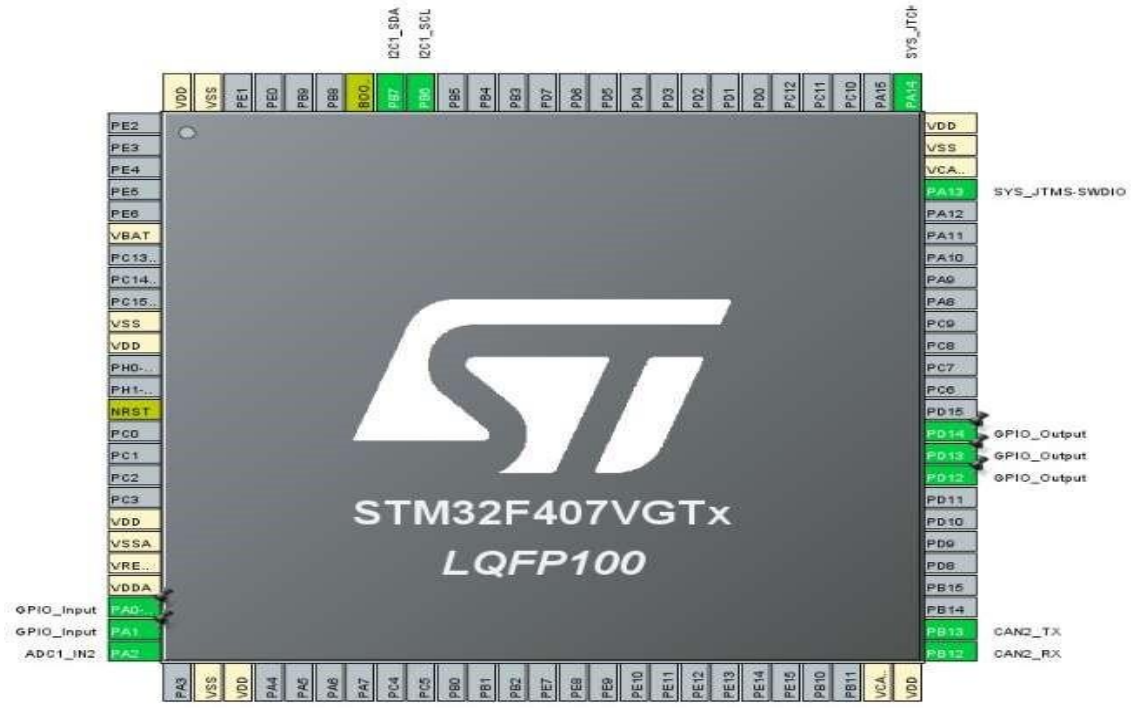


Figure: Pin Configuration of STM32F407VGT6

4.2. Clock Configuration

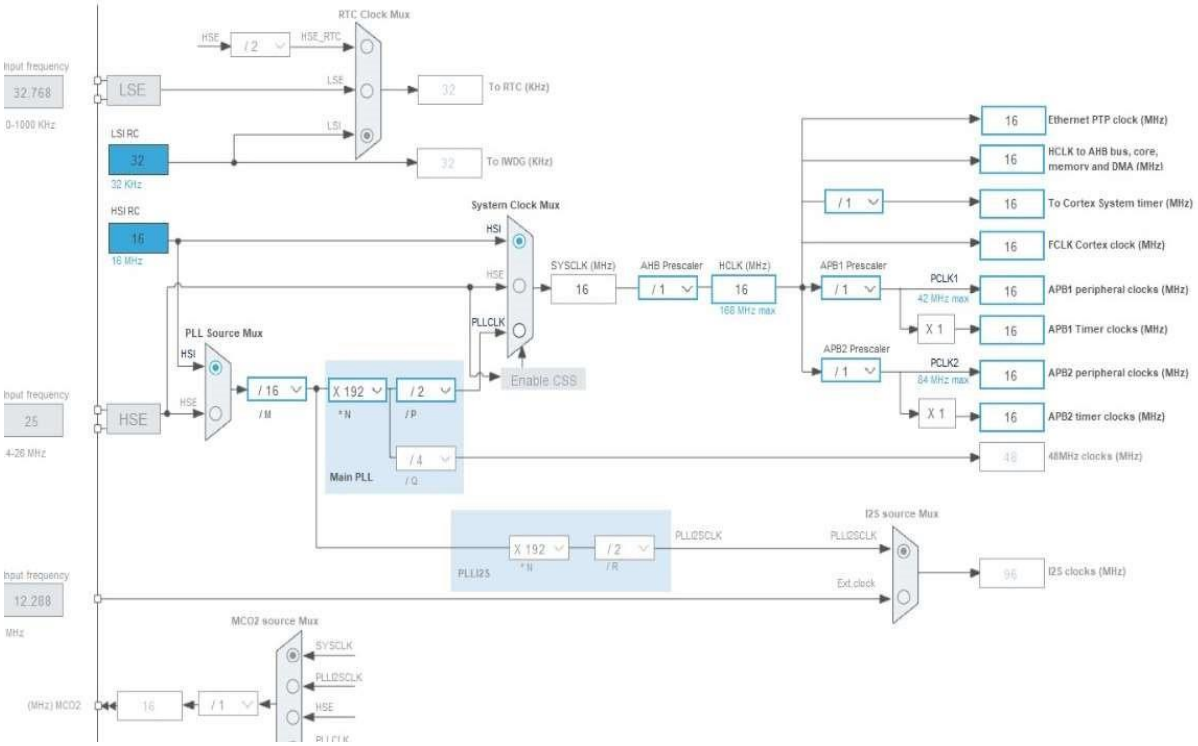
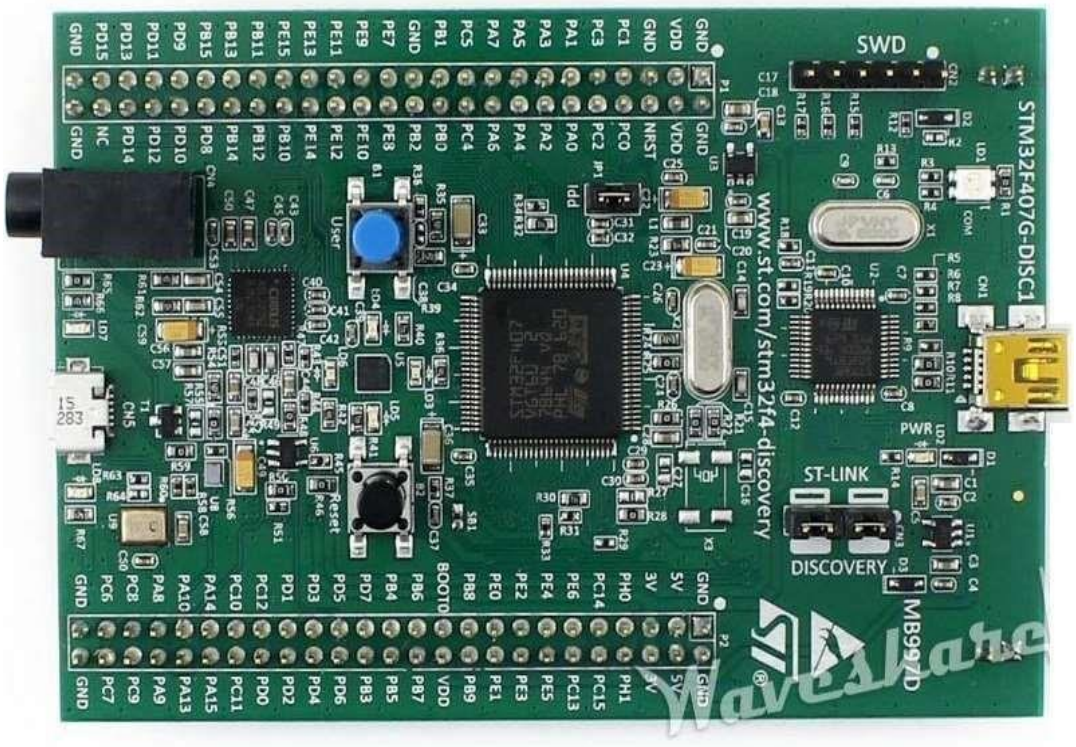


Figure: Clock Configuration

4.3. Hardware and Components

4.3.1. STM32F407G-DISC1

The STM32F407G-DISC1 microcontroller is a high-performance ARM Cortex-M4based 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.

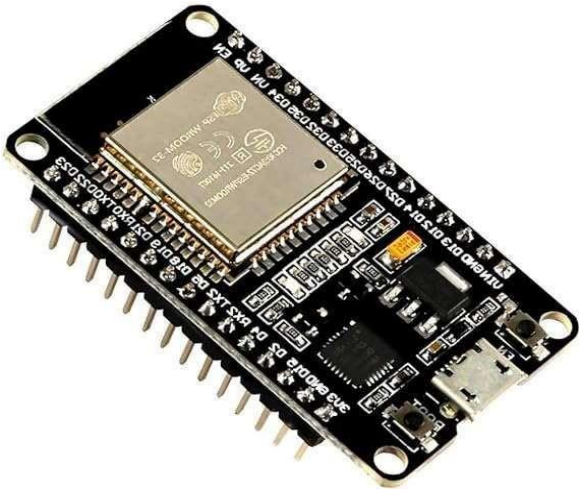
Data Acquisition: The STM32F407GDISC1 MCU interfaces with various sensors, such as temperature sensors, speed sensors, NO2 sensors, and others, to collect real time data.

-

- 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 HC05 module, ESP8266 module, LCD display, and others, enabling seamless data transmission and control.
- Firmware development for the STM32F407GDISC1 MCU is carried out using integrated development environments (IDEs) such as STM32CubeIDE or Keil µVision.

4.3.2. ESP8266

- Integrate an ESP8266 microcontroller module into the system architecture for additional processing power and wireless connectivity capabilities.
- Utilize the ESP8266 module to facilitate communication between the STM32 board and the cloud server or other IoT devices.

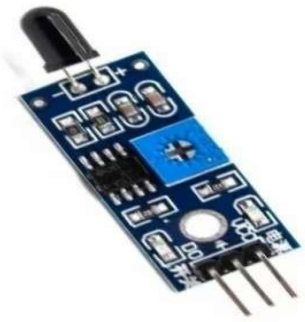


- Configure the ESP8266 module to establish Wi-Fi or Bluetooth connections, enabling wireless data transmission and remote-control functionality. Develop firmware on the ESP8266 module to receive sensor data from the STM32 board via CAN Protocol.
- Implement protocols for securely uploading sensor data to the cloud server for storage and further analysis.
- Leverage the capabilities of the ESP8266 module to offload certain processing tasks from the STM32 board, optimizing system performance and efficiency.
- Integrate the ESP8266 module into the overall system architecture, ensuring seamless interoperability with other hardware components and firmware functionalities.

4.3.4. GAS Sensor

- Integrate the gas sensor for detecting specific gas concentration levels.
- Ensure proper wiring and connections between the gas sensor and the STM32 board.
- Verify compatibility and calibration of the gas sensor with the STM32 board for accurate gas concentration measurements.
- Develop sensor data acquisition routines on the STM32 board tailored for the gas sensor.
- Configure the STM32 board to sample gas concentration readings from the sensor at regular intervals.
- Implement algorithms to process raw sensor data and convert it into gas concentration values.
- Continuously monitor gas concentration levels in real-time to detect variations in air quality.

4.3.6. Heat Flame Sensor



- Utilize the STM32F407G DISC-1 microcontroller as the central processing unit.
  Integrate the heat flame sensor for detecting flame and heat levels.

Ensure proper wiring and connections between the heat flame sensor and the STM32 board.

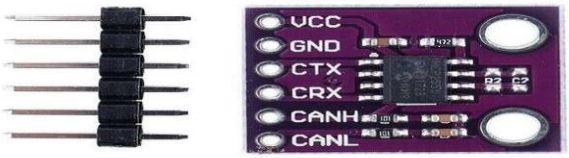
Verify compatibility and configuration of the heat flame sensor with the STM32 board for accurate flame detection.

- Develop sensor data acquisition routines on the STM32 board to interface with the heat flame sensor.
- Utilize the heat flame sensor to monitor flame and heat levels with the STM32 board.
- Configure the STM32 board to transmit sensor data related to flame detection via CAN or ESP8266.

• Integrate the STM32 board with the ESP8266 module for wireless transmission of flame detection data.

• Implement real-time alerts on the STM32 board for dangerous flame or heat detection levels.

4.3.8. MCP2551 CAN-BUS TRANSCEIVER



- Utilize the STM32F407G-DISC1 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.

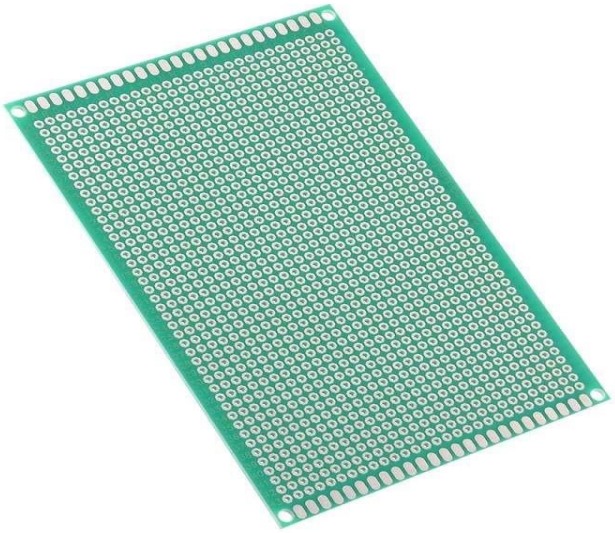
Leverage the ESP8266 module for wireless communication to complement the CAN network by providing remote data access and control capabilities.

- 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 ESP8266 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

4.3.9. PCB Board

- Physically mount and electrically connect various components, such as microcontrollers, sensors, and transceivers.
- Provide a structured layout for routing electrical signals and power between components.
- Ensure reliable and stable operation of electronic circuits through proper design and grounding.
- Facilitate ease of assembly, testing, and troubleshooting of the electronic system.



4.3.10. Rainbow and Jumper Wires

Utilize rainbow and jumper wires to establish connections between the STM32 board, CAN transreceiver MCP2551, sensors, and other peripherals.

Select appropriate resistor values for sensor interfacing and CAN signal conditioning, ensuring compatibility with the STM32 board and sensor specifications.

Integrate resistors into the circuit design to regulate current flow, protect components from overvoltage, and maintain signal integrity.

Employ rainbow and jumper wires to connect components on the breadboard, facilitating flexible and modular circuit assembly.

Implement proper resistor placement and wiring configurations using rainbow and jumper wires to optimize circuit performance and enhance system reliability

5. Software

5.1 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 STLINK 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.

5.1.1. 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.

5.2. **Arduino IDE**

The Arduino IDE serves as a fundamental software tool for programming Arduino microcontroller boards, providing an accessible and user-friendly platform for developing embedded projects. Designed with simplicity in mind, the IDE offers a straightforward integrated development environment suitable for both beginners and experienced developers. With its intuitive interface and extensive library support, the Arduino IDE simplifies the process of writing, compiling, and uploading code to Arduino boards. Developers can leverage the IDE's built-in code editor, which features syntax highlighting, auto-completion, and error checking functionalities to facilitate code development. Additionally, the IDE offers a diverse selection of prebuilt libraries and example code, enabling developers to easily integrate complex functionalities into their projects without the need for extensive programming knowledge. Furthermore, the Arduino IDE supports a wide range of Arduino compatible boards, allowing developers to choose the most suitable hardware platform for their applications. Overall, the Arduino IDE plays a pivotal role in the Arduino ecosystem, empowering developers to unleash their creativity and bring innovative ideas to life through embedded programming.

5.3. Cloud Platform

5.3.1. ThingSpeak

The ThingSpeak cloud platform stands as a robust and versatile solution for building and deploying IoT applications, offering a comprehensive suite of features and functionalities for device management, data visualization, and application development. As a highly scalable and customizable platform, ThingSpeak facilitates the seamless integration of IoT devices, sensors, and gateways, enabling users to collect, process, and analyze real-time data from connected devices. With its intuitive dashboard builder and drag-and-drop interface, ThingSpeak empowers users to create dynamic and interactive dashboards to visualize sensor data, monitor device performance, and track key metrics in real-time. Moreover, the platform offers advanced data processing capabilities, including rule chains, complex event processing, and integration with external systems, enabling users to implement custom business logic and automate decision-making processes based on incoming data streams. Additionally, ThingSpeak provides robust security features, including rolebased access control, encryption, and device authentication, ensuring the confidentiality, integrity, and availability of IoT data throughout the entire lifecycle. Furthermore, ThingSpeak supports seamless integration with thirdparty services and applications through its extensive set of APIs and connectors, enabling users to leverage existing infrastructure and tools within their IoT ecosystem. Overall, ThingSpeak serves as a powerful and flexible cloud platform for building and deploying scalable and feature-rich IoT solutions across a wide range of industries and use cases.

6. Communication Protocols

6.1 Serial Peripheral Bus

SPI (Serial Peripheral Interface) is utilized for fetching onboard accelerometer data from the STM32F407VGt6 microcontroller. The SPI protocol is chosen for its ability to facilitate high-speed serial communication between the microcontroller and the accelerometer sensor. Specifically, SPI is employed to retrieve data from the accelerometer's internal registers representing the x, y, and z-axis measurements of acceleration.

- The SPI interface on the STM32 microcontroller is configured and initialized to establish communication with the accelerometer sensor.
- Parameters such as clock frequency, clock polarity, clock phase, and data format are set according to the specifications of the accelerometer device.
- The STM32 microcontroller initiates a read operation by sending a command or address byte to the accelerometer via SPI.
- The command byte specifies the register address from which the accelerometer data is to be read.
- Following the command byte transmission, the STM32 microcontroller sends dummy data bytes to the accelerometer to clock out the response data.
- Simultaneously, the accelerometer responds with the requested data bytes containing the x, y, and z-axis acceleration measurements.
- Upon receiving the response data from the accelerometer, the STM32 microcontroller interprets the data bytes to extract the x, y, and z-axis acceleration values.
- The extracted acceleration values are typically represented in digital form and may require additional processing, such as scaling or conversion, to obtain meaningful acceleration measurements.
- The fetched accelerometer data, representing the x, y, and z-axis accelerations, can be processed and utilized for various applications within the project. 6.2 Controller Area Network (CAN) Protocol

6.2 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.

6.2.1 Integration with ESP8266 and CAN

The STM32F407G-DISC1 microcontroller uses CAN communication to interface with the ESP8266 module for further processing or wireless transmission:

- The CAN interface on the STM32 is configured to transmit sensor data packets to the ESP8266 module.
- The CAN RX pin of the STM32 is connected to the CAN TX pin of the ESP8266 module, establishing a serial communication link.
- Upon receiving data, the ESP8266 processes the CAN messages for further analysis, storage, or transmission via its wireless capabilities.

6.3. One-Wire Protocol

The One-Wire protocol provides a simple and efficient method for communication between the STM32F407G-DISC1 microcontroller and external devices, such as temperature sensors or other One-Wire compatible peripherals. The protocol enables data exchange over a single data line, making it ideal for low-pin-count applications and easy integration.

- The STM32F407G-DISC1 microcontroller is configured to support One-Wire communication with compatible devices through a single GPIO pin.

- One-Wire devices, such as temperature sensors, are connected to the STM32 via the dedicated data line, which handles both data transmission and reception.
- One-Wire communication operates with a unique addressing scheme, allowing multiple devices to be connected to the same data line and addressed individually.
- Data transmission occurs asynchronously, with the microcontroller sending and receiving data packets framed according to the One-Wire protocol specifications.
- The microcontroller initiates communication by sending a reset pulse, followed by command and data sequences to interact with One-Wire devices.
- One-Wire communication employs a simple data format, including command bytes and data bytes, to facilitate device interactions and data exchange.

In the project, the One-Wire protocol is utilized for interfacing with temperature sensors and other peripherals:

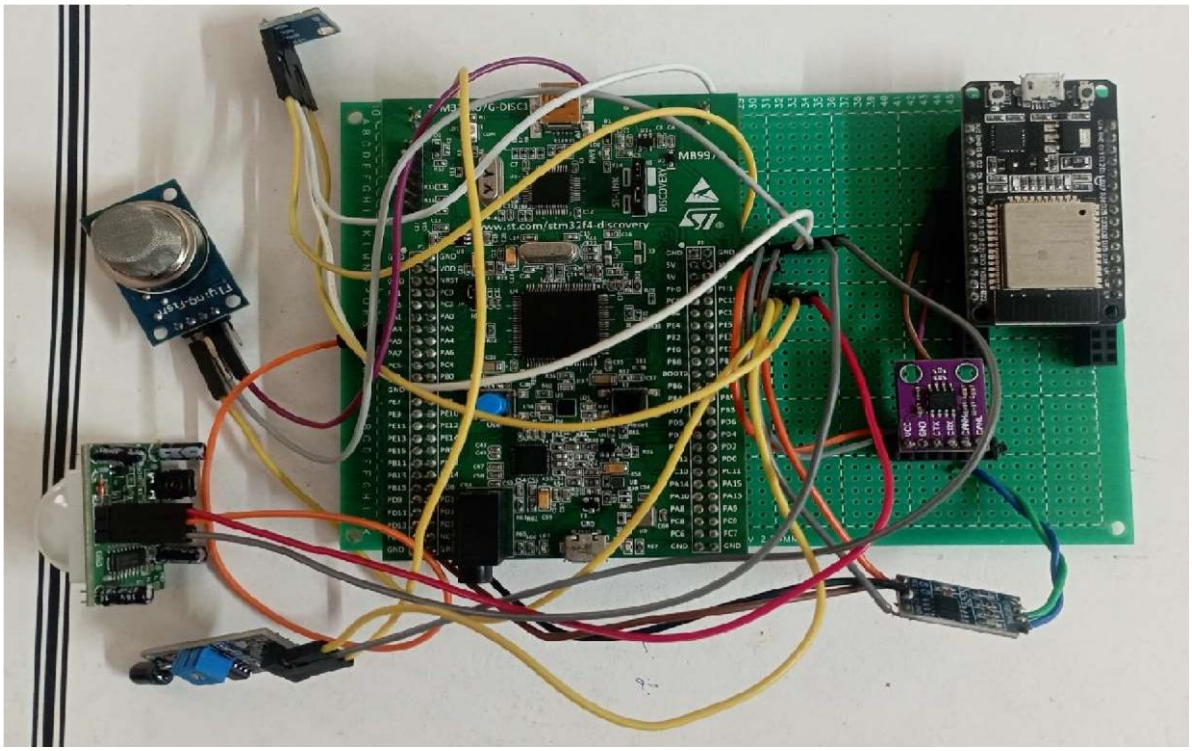
One-Wire Sensor Integration:

- Connect the One-Wire temperature sensors to the STM32F407G-DISC1 microcontroller using a single GPIO pin for data communication.
- Configure the STM32 to send and receive data packets to and from the One-Wire sensors, retrieving temperature measurements or other sensor data.
- Implement routines on the STM32 to handle the One-Wire protocol, including device addressing, command sequences, and data processing.

Data Handling and Communication:

- Process the sensor data retrieved via the One-Wire protocol and integrate it into the overall system for further analysis or transmission.
- Use the data acquired from One-Wire sensors for system monitoring, control, or display purposes, depending on project requirements.

7. Output



8. Conclusion :

Connected telemetry devices and sensors play a crucial role in our CNG Gas Leakage Detection System, enabling real-time data monitoring, collection, and transmission to ensure safety and efficiency.

Here are some key aspects of telemetry in our project:

- **Real-time monitoring:**

  - The telemetry system provides continuous insights into the performance of the STM32F407VG microcontroller, CAN communication, and connected sensors such as the MQ-9 gas sensor and HC-015 flame sensor.

  - This enables proactive detection of CNG gas leaks and fire hazards, allowing immediate interventions.

- **Data collection and analysis**:

  - The system gathers sensor data, including gas concentration levels, flame detection status, and system alerts via the CAN bus.

  - By analyzing this data, the system can detect abnormal conditions, trigger alarms, and take necessary safety measures.

- **Remote connectivity**:

  - Wireless communication is integrated using the ESP8266 module and GSM800L, enabling remote monitoring of gas leak and fire alerts.

  - SMS alerts are sent in case of emergencies, ensuring timely notifications to users and authorities.

- **Scalability and flexibility:**

  - The system can be easily expanded to include additional sensors or modules, making it adaptable to different industrial or residential environments.

  - The CAN-based network ensures reliable communication among multiple sensors and controllers.

- **Enhanced efficiency and safety:**

  - By continuously monitoring gas leakage and fire hazards, the system improves operational safety and prevents potential disasters.

- In case of dangerous gas levels or fire, the system triggers an alarm, activates the servo motor for safety measures, and sends notifications to ensure a quick response.

9. Future Scope

The future scope of the CNG Gas Leakage Detection System with connected telemetry devices is vast, offering opportunities for innovation and improvement. Here are some key areas for future advancements:

- **Expansion of IoT ecosystem**:
  - The system can be enhanced by integrating more IoT-enabled devices, sensors, and modules into the CAN network and STM32-based platform.
  - This will lead to a more interconnected system, improving data-driven decision-making and real-time monitoring.
- **Integration with AI and machine learning:**
  - Future enhancements could involve AI and machine learning algorithms to analyze sensor data and CAN bus communication.
  - AI-based predictive maintenance can help detect faults or irregularities before they lead to critical failures.
  - Machine learning models can improve anomaly detection, optimizing system performance and reliability.
- **Advancements in connectivity**:
  - The adoption of emerging wireless communication technologies like 5G and low-power wide-area networks (LPWAN) can improve real-time data transmission.
  - This will enable remote monitoring with higher reliability and lower latency, even in industrial or hazardous environments.
- **Enhanced data security**:
  - As telemetry devices become more integral to safety systems, cybersecurity measures will need to be strengthened.
  - Implementing advanced encryption and authentication protocols will protect sensitive data transmitted over the CAN network and other communication channels.

- **Energy efficiency and sustainability**:
  - The use of energy-efficient components and low-power sensors will extend battery life and reduce power consumption.
  - Future designs can incorporate solar-powered or energy-harvesting modules to ensure long-term sustainability in remote or off-grid locations.
- **Integration with smart city and industrial automation**:
  - The system can be scaled and integrated into larger industrial automation frameworks and smart city infrastructure.
  - This would allow seamless coordination with fire safety systems, gas monitoring networks, and emergency response mechanisms.

By leveraging these advancements, the CNG Gas Leakage Detection System can become more efficient, intelligent, and adaptable to evolving technological landscapes, ensuring greater safety and reliability in industrial and residential applications.