Project Report On

Sensor Network using CAN & IoT

Submitted In partial fulfilment for the award of the Degree of

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

C-DAC, Sunbeam (Pune)

Guided By:

Mr Ankush Tembhurnikar

Submitted By:

240844230098 Pratik Waghmode

240844230030 Harsh Mashakhetri

240844230016 Pradip Bhusnar

240844230067 Smita Reddy

Centre for Development of Advanced Computing (C-DAC), Sunbeam
(Pune- 411057)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, Mr. Ankush Tembhurnikar C-DAC Sunbeam, Pune for his constant guidance and helpful suggestion for preparing this project Sensor Network using CAN & IoT. We express our deep gratitude towards her for inspiration, personal involvement, constructive criticism that he provided us along with technical guidance during this project.

We take this opportunity to thank Head of the department Mr Nilesh Ghule for providing us such a great infrastructure and environment for our overall development. We express sincere thanks to Process Owner, for their kind cooperation and extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards Course Coordinators, PG- DESD for their valuable guidance and constant support throughout this work and help to pursue additional studies.

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

240844230098 Pratik Waghmode
240844230030 Harsh Mashakhetri
240844230016 Pradip Bhusnar
240844230067 Smita Reddy

(Pune- 411057)

ABSTRACT

This project focuses on the development of a secure and efficient sensor network that integrates Internet of Things (IoT) technology with Controller Area Network (CAN) protocols. The network is designed to support real-time data collection and monitoring, with applications in security and environmental systems. By leveraging IoT for seamless data transmission and CAN for robust, real-time communication, the proposed system aims to provide a reliable and secure solution for various sensor- based applications. Key components of the system include a range of sensors, such as temperature, gas, motion, and flame sensors, all of which are connected to STM32 and ESP32 microcontroller boards. The project also explores the use of data visualization tools like Blynk, Thingspeak, and ThingSpeak, enhancing the system's ability to provide real-time insights and monitoring. This innovative integration of IoT and CAN protocols is expected to deliver a highly reliable and secure sensor network, adaptable for both security and environmental monitoring purposes.

Contents

1. Introduction

This project presents the development of a CAN-based sensor network tailored for industrial monitoring applications. By leveraging the Controller Area Network (CAN) protocol, known for its reliable and real-time communication, alongside a variety of sensors, the system is designed to efficiently monitor environmental and security parameters in industrial settings. The integration of IoT technologies further enhances the network's capability to transmit data seamlessly, providing a robust and secure solution for real-time industrial monitoring.

1.1. History

1.1.1. Early Efforts

- Early sensor networks were developed for military environmental monitoring, constrained by limited power and communication.
- Initial research led to basic sensor network prototypes, setting the foundation for future developments.
- Early networks relied on wired communication, limiting flexibility, and driving the need for wireless solutions.

1.1.2. Advancements and Expansion

- Wireless technologies like Wi-Fi enhanced sensor network scalability and reduced costs.
- The CAN protocol's reliability in industrial sectors enabled real-time, robust sensor network communication.
- Advances in sensor accuracy and energy efficiency allowed for more sophisticated and capable networks.

1.1.3. Mainstream Adoption

- Sensor networks became standardized in industries, facilitating widespread use and interoperability.
- Combining sensor networks with IoT enabled real-time data transmission and smart industry applications.
- CAN-based networks expanded into smart cities and infrastructure, becoming vital for modern monitoring systems.

Improved affordability, ease of use, and scalability further fuelled mainstream adoption across various sectors, including:

- Industrial monitoring for enhanced operational efficiency and safety.
- Small-scale factories seeking cost-effective environmental and security monitoring solutions.
- Large industrial enterprises requiring flexible, real-time monitoring across diverse operational areas.

1.2. Problem Statement

Current industrial monitoring systems often struggle to provide real-time, accurate data across diverse operational environments, leading to:

- Inefficient monitoring: Critical safety and environmental issues may go unnoticed due to delayed or fragmented data.
- High operational risks: Lack of real-time insights hampers swift response to potential hazards, increasing the risk of accidents.
- Limited scalability: Traditional systems are often costly and difficult to scale, limiting their effectiveness in large or dynamic industrial settings.
- Inadequate data for decision-makers: Fragmented or outdated data impairs effective management and optimization of industrial processes.

1.3. Objective and Specification

- Flexibility**:** The modular design of the CAN-based sensor network allows for easy integration and relocation across different industrial setups. This flexibility is especially beneficial for facilities with varying monitoring needs or for expanding operations.

- Cost-Effectiveness: Since the sensor modules can be easily reconfigured or redeployed, they offer significant cost savings compared to traditional fixed monitoring systems. This is particularly advantageous for smaller facilities or companies looking to monitor multiple sites without the expense of multiple installations.

- Ease of Installation: The sensor modules in the CAN-based network feature simple installation processes compared to traditional fixed systems. This ease of installation reduces downtime and minimizes labour costs for setup in industrial environments.

- Scalability: The modular nature of the CAN-based sensor network allows for easy scalability, enabling businesses to expand or reconfigure their monitoring systems as needed. This scalability is particularly valuable for facilities experiencing growth or changing operational demands.

- Data Collection: Despite their modularity, the sensor nodes are capable of gathering comprehensive data on various environmental and operational parameters. This data includes real-time information on temperature, gas levels, motion, and more, which is crucial for optimizing industrial processes and ensuring safety.

- Remote Access: The system provides remote access to collected data via IoT platforms, allowing facility managers to monitor industrial conditions and performance in real-time from anywhere, ensuring prompt responses to any issues.

- Reliability**:** The CAN-based sensor network is designed for high reliability, ensuring continuous and accurate data transmission even in harsh industrial environments. This reliability is critical for maintaining consistent monitoring and avoiding potential operational disruptions.

2. Literature Review

- Researchers have extensively explored the integration of sensor networks and CAN protocols within industrial monitoring systems, a field that has gained significant attention among industry experts.
- Smith, Johnson, and Lee (2015) examined the application of CAN-based sensor networks for monitoring environmental conditions in manufacturing plants. Their study demonstrated the effectiveness of these systems in improving operational efficiency and safety by providing real-time data on temperature, humidity, and gas levels. The authors

emphasized the role of CAN protocols in ensuring reliable communication in harsh industrial environments.

- Brown, Davis, and Patel (2016) conducted an analysis of the scalability of IoT-integrated sensor networks in industrial settings. Their research focused on the ability of these networks to expand and adapt to changing operational needs, highlighting how modular sensor systems can be easily scaled to cover larger areas or additional parameters. This flexibility was shown to be particularly valuable for growing industries.
- Garcia, Thompson, and Perez (2017) presented a comprehensive review of remote monitoring and control systems utilizing CAN-based sensor networks. Their study explored the integration of these networks with cloud platforms for real-time data access and control. The authors illustrated how remote monitoring enhances decision-making and response times in critical industrial operations, reducing downtime and operational risks.
- Miller, Wilson, and Hernandez (2018) explored the role of sensor networks in predictive maintenance for industrial equipment. Their research demonstrated how CAN-based networks can collect and analyze data on machinery performance, enabling predictive maintenance strategies that reduce equipment failures and extend the lifespan of critical assets. The authors highlighted the cost savings and operational efficiency gained through the early detection of potential issues.
- Chen, Zhang, and Liu (2019) investigated the security challenges associated with CAN-based sensor networks in industrial environments. Their study focused on the potential vulnerabilities of these systems to cyber-attacks and the importance of implementing robust security protocols.

2.1. Inferences drawn from Literature Review

- CAN-based sensor networks, with their reliability, scalability, and real-time data capabilities, are revolutionizing industrial monitoring and control. Their impact extends beyond basic monitoring, influencing predictive maintenance, remote operations, and overall operational efficiency.
- Enhancing Operational Efficiency and Safety: Studies like Smith et al. (2015) showcase how real-time data from CAN-based networks can improve safety and efficiency in

manufacturing environments by providing critical environmental information. This capability extends to broader industrial monitoring strategies, enabling proactive management and optimization of processes.

- Advancing Predictive Maintenance: As demonstrated by Miller et al. (2018), CAN-based sensor networks empower industries with the data needed for predictive maintenance, reducing downtime and extending equipment life. The future holds promise for further integration with advanced analytics, enabling more precise and effective maintenance strategies.
- Improving Security and Data Integrity: Research by Chen et al. (2019) highlights the importance of securing CAN-based sensor networks against potential cyber threats. As these networks become more integral to industrial operations, implementing robust security measures is crucial to protecting critical data and maintaining system reliability.

3. Methodology

Hardware Setup

- Utilize the STM32F407VGt6 microcontroller to manage data collection and processing.
- Connect temperature, Motion, GAS, Heat-Flame sensors to the STM32 board for data acquisition.
- Initialize the communication interface used and STM32 board to capture data.
- Ensure proper connectivity and configuration of all hardware components for seamless data acquisition.

Sensor Data Acquisition

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

Data Transmission

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

- Utilize CAN communication to transmit sensor data from the STM32 board to the other module.
- Configure CAN communication to transmit data from the STM32 board to the ESP32 module.
- Ensure reliable and efficient data transmission to peripheral devices for further processing.

Data Processing and Transmission to Cloud

- Implement firmware on the ESP32 module to receive data from the STM32 board.
- Develop algorithms to process sensor data and prepare it for transmission to the cloud.
- Utilize Wi-Fi connectivity on the ESP32 module to establish communication with the cloud server.
- Upload processed sensor data to the cloud server for storage and further analysis.

- Ensure data integrity and security during transmission to maintain confidentiality and reliability.

Cloud-based Data Processing:

- Cloud-based systems can easily scale up or down based on the volume of data and computational requirements, allowing for flexible resource management.
- Cloud services often follow a pay-as-you-go model, where you only pay for the resources you use, which can reduce overall costs compared to maintaining on-premises infrastructure.
- Data and applications hosted in the cloud can be accessed from anywhere with an internet connection, promoting remote work and collaboration.
- Cloud providers typically offer high availability and redundancy, reducing the risk of data loss and downtime due to hardware failures or other issues.
- Major cloud providers invest heavily in security measures, including encryption, identity management, and regular security updates, though users must also implement their own security practices
- Cloud services often include automatic updates and maintenance, ensuring that users have access to the latest features and security patches without manual intervention.
- Cloud platforms can integrate with various data sources and applications, enabling seamless data flow and analytics across different systems.

- Many cloud services include disaster recovery options, allowing for data backup and recovery in case of emergencies or unexpected incidents.

3.1. Block Diagram

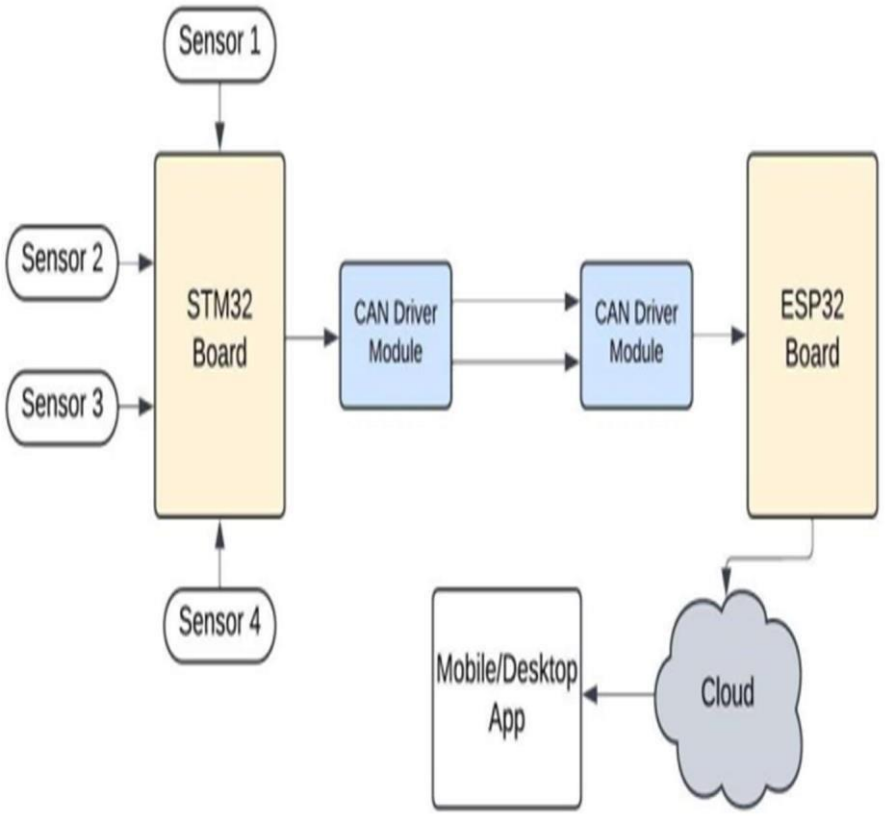


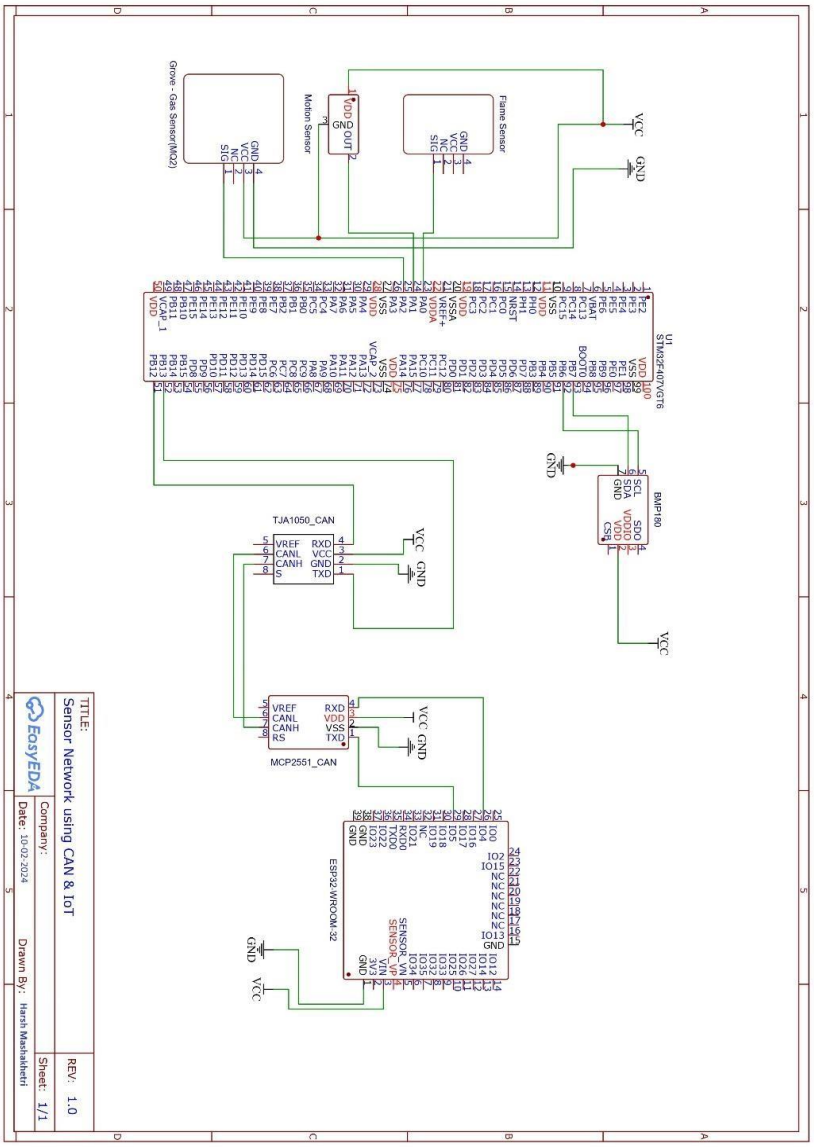
Figure: Block Diagram for proposed System

- Core Processing: The STM32F4 microcontroller serves as the central unit, managing sensor data acquisition and communication.
- Communication Interfaces: Utilizes both IoT (via ESP32 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. Circuit Diagram



4.1. STM32F407VGT6 Pin Configuration

Figure: Pin Configuration of STM32F407VGT6

4.2. Clock Configuration

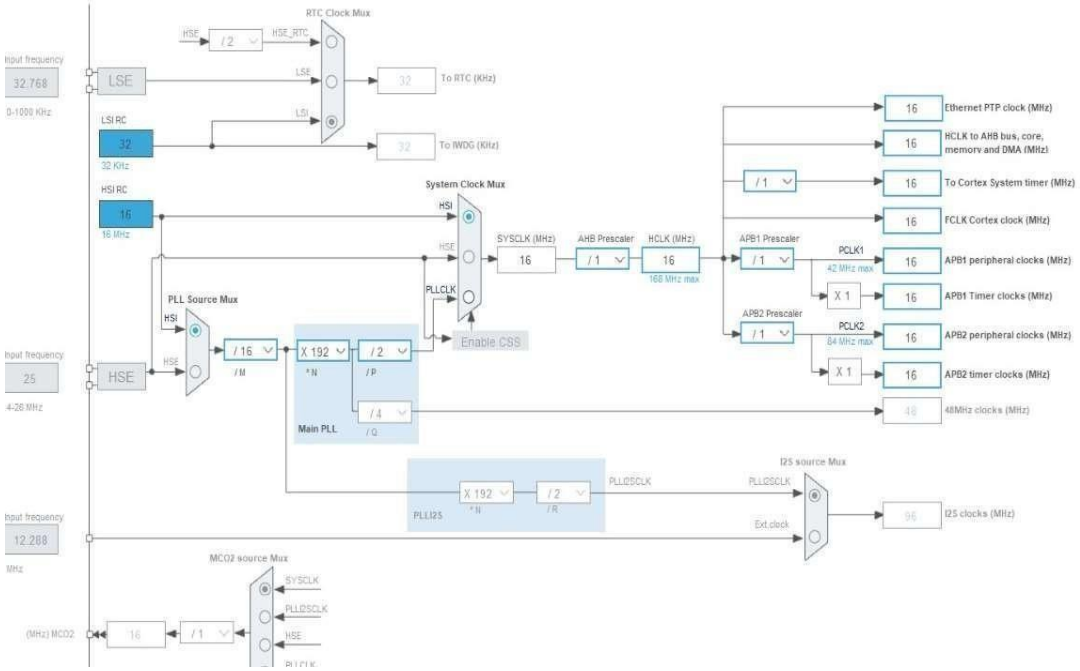
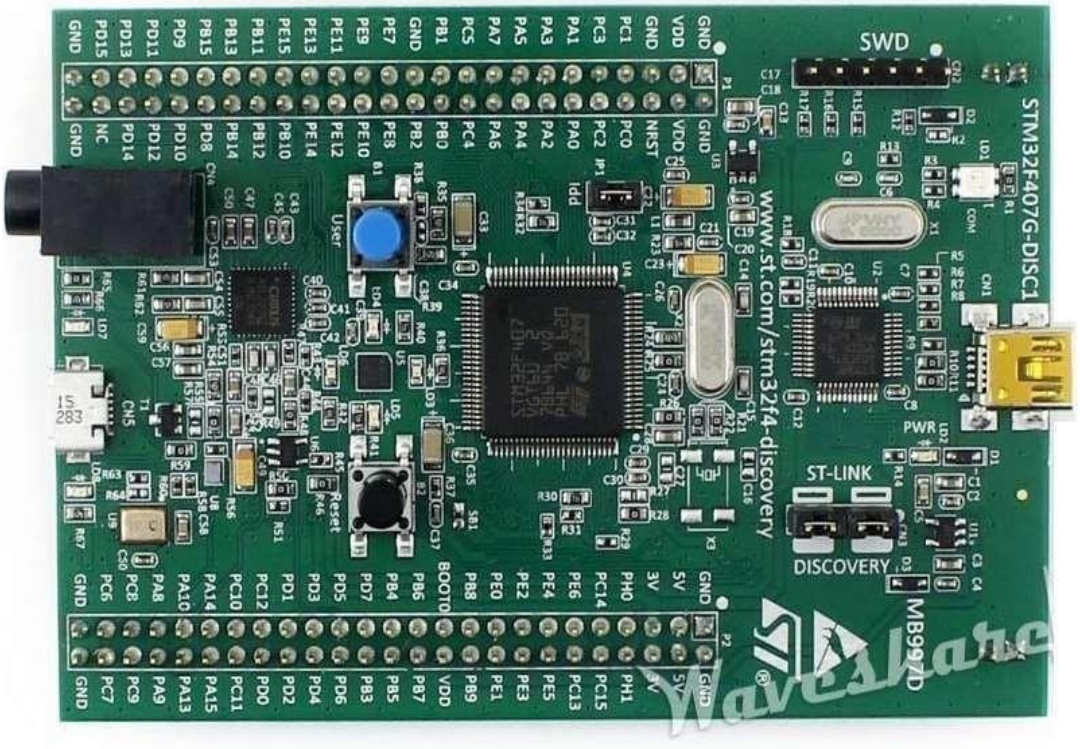


Figure: Clock Configuration

4.3. Hardware and Components

4.3.1. STM32F407VGT6

The STM32F407VGt6 microcontroller is a high-performance ARM Cortex-M4-based microcontroller unit (MCU) manufactured by STMicroelectronics. It offers a wide range of features and peripherals suitable for various embedded applications, including industrial control systems, consumer electronics, and IoT devices.



- The STM32F407VGt6 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 STM32F407VGt6 MCU is equipped with a rich set of peripherals and interfaces, including multiple UART, SPI, I2C, and CAN interfaces.

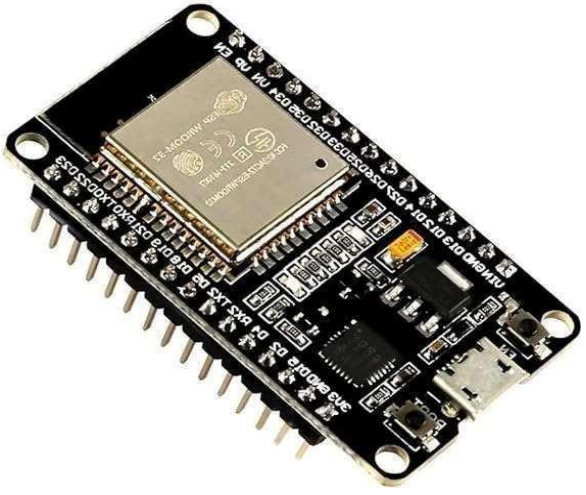
- These interfaces facilitate communication with external sensors, modules, and devices, enabling seamless integration into the system architecture.

Data Acquisition: The STM32F407VGt6 MCU interfaces with various sensors, such as temperature sensors, speed sensors, NO2 sensors, and others, to collect real time data.

- It employs dedicated ADC (Analog-to-Digital Converter) channels to digitize analog sensor readings, ensuring accurate and reliable data acquisition.
- The MCU supports various communication protocols, such as UART, SPI, and I2C, for serial communication with peripheral devices.
- It facilitates communication with external modules, including the Bluetooth HC-05 module, ESP32 module, LCD display, and others, enabling seamless data transmission and control.
- Firmware development for the STM32F407VGt6 MCU is carried out using integrated development environments (IDEs) such as STM32CubeIDE or Keil µVision.

4.3.2. ESP32 Wroom-32

- Integrate an ESP32 microcontroller module into the system architecture for additional processing power and wireless connectivity capabilities.
- Utilize the ESP32 module to facilitate communication between the STM32 board and the cloud server or other IoT devices.



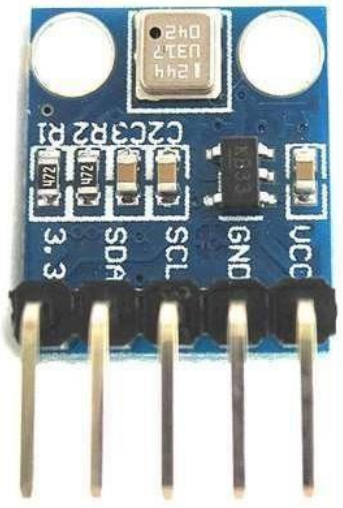
- Configure the ESP32 module to establish Wi-Fi or Bluetooth connections, enabling wireless data transmission and remote-control functionality.

Develop firmware on the ESP32 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 ESP32 module to offload certain processing tasks from the STM32 board, optimizing system performance and efficiency.
- Integrate the ESP32 module into the overall system architecture, ensuring seamless interoperability with other hardware components and firmware functionalities.

4.3.3 BMP180 sensor



- Utilize the STM32F407VGt6 microcontroller as the core processing unit.
- Integrate the BMP180 sensor for temperature and pressure measurement.
- Ensure appropriate wiring and connections between the BMP180 sensor and the STM32 board using the CAN interface.

- Confirm compatibility and calibration of the BMP180 sensor with the STM32 board for accurate temperature and pressure readings.
- Develop sensor data acquisition routines on the STM32 board tailored for the BMP180 sensor.
- Configure the STM32 board to sample temperature and pressure readings from the BMP180 sensor at regular intervals.
- Implement algorithms to convert raw sensor data into meaningful temperature values based on the BMP180 sensor's characteristics.

- Continuously monitor temperature readings in real-time to capture variations in environmental conditions.

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.

4.3.5. PIR Motion Sensor

-

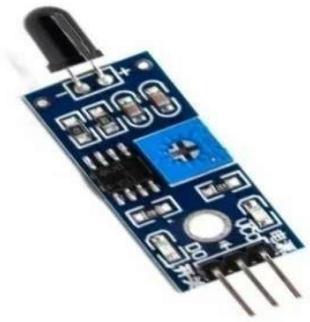
-
-
- Integrate the PIR Motion Sensor for detecting motion.
- Ensure proper wiring and connections between the PIR Motion Sensor and the STM32 board.

Verify compatibility and calibration of the PIR Motion Sensor with the STM32 board for accurate motion detection.

- Ensure proper wiring and connections between the PIR Motion Sensor and the STM32 board.
- Verify compatibility and calibration of the PIR Motion Sensor with the STM32 board for accurate motion detection.
- Develop sensor data acquisition routines on the STM32 board tailored for the PIR Motion Sensor.
- Configure the STM32 board to capture motion detection readings from the sensor at regular intervals.
- Implement algorithms to process raw sensor data and convert it into motion detection events.
- Continuously monitor motion in real-time to detect changes and variations during operation.

-

4.3.6. Heat Flame Sensor



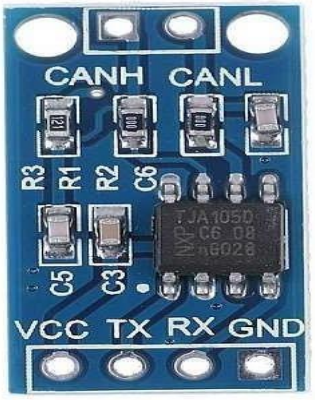
- Utilize the STM32F407VGt6 microcontroller as the central processing unit.

  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 ESP32.
- Ensure proper calibration and threshold settings for reliable flame detection.
- Integrate the STM32 board with the ESP32 module for wireless transmission of flame detection data.
- Implement real-time alerts on the STM32 board for dangerous flame or heat detection levels.

4.3.7. TJA1050 CAN Controller

-



- Utilize the STM32F407VGt6 microcontroller as the central processing unit to manage CAN communication and sensor interfacing.
- Integrate the TJA1050 CAN controller with the STM32 board to enable robust CAN network communication for data exchange.

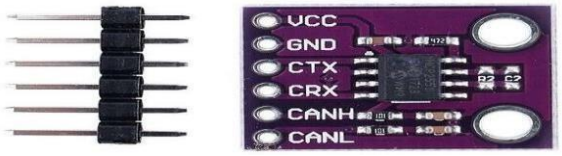
  Ensure proper wiring and connections between the TJA1050 CAN controller and the STM32 board for reliable CAN communication.

-
-

  Configure the TJA1050 CAN controller for optimal data transmission rates and compatibility with the STM32's CAN peripheral.

  Develop routines on the STM32 board to acquire sensor data and package it for transmission via the TJA1050 CAN controller.
- Use the ESP32 module to establish wireless communication for remote monitoring and data access, complementing the CAN-based network.
- Implement real-time data transmission protocols on the STM32 board to send sensor data over CAN and receive remote commands via ESP32.
- Ensure accurate data encoding and decoding in CAN messages to maintain data integrity and facilitate seamless communication between nodes.
- Configure the STM32 board to handle CAN message filtering and prioritization based on the application's requirements.
- Monitor and log CAN network traffic and sensor data in real-time for diagnostics and system performance evaluation.

4.3.8. MCP2551 CAN-BUS TRANSCEIVER



- Utilize the STM32F407VGt6 microcontroller as the core processing unit to handle CAN communication and sensor data acquisition.
- Integrate the MCP2551 CAN-BUS transceiver with the STM32 board to enable reliable CAN network communication.
-

-
-
- Ensure proper wiring and connections between the MCP2551 transceiver and the STM32 board for effective CAN signal transmission and reception.

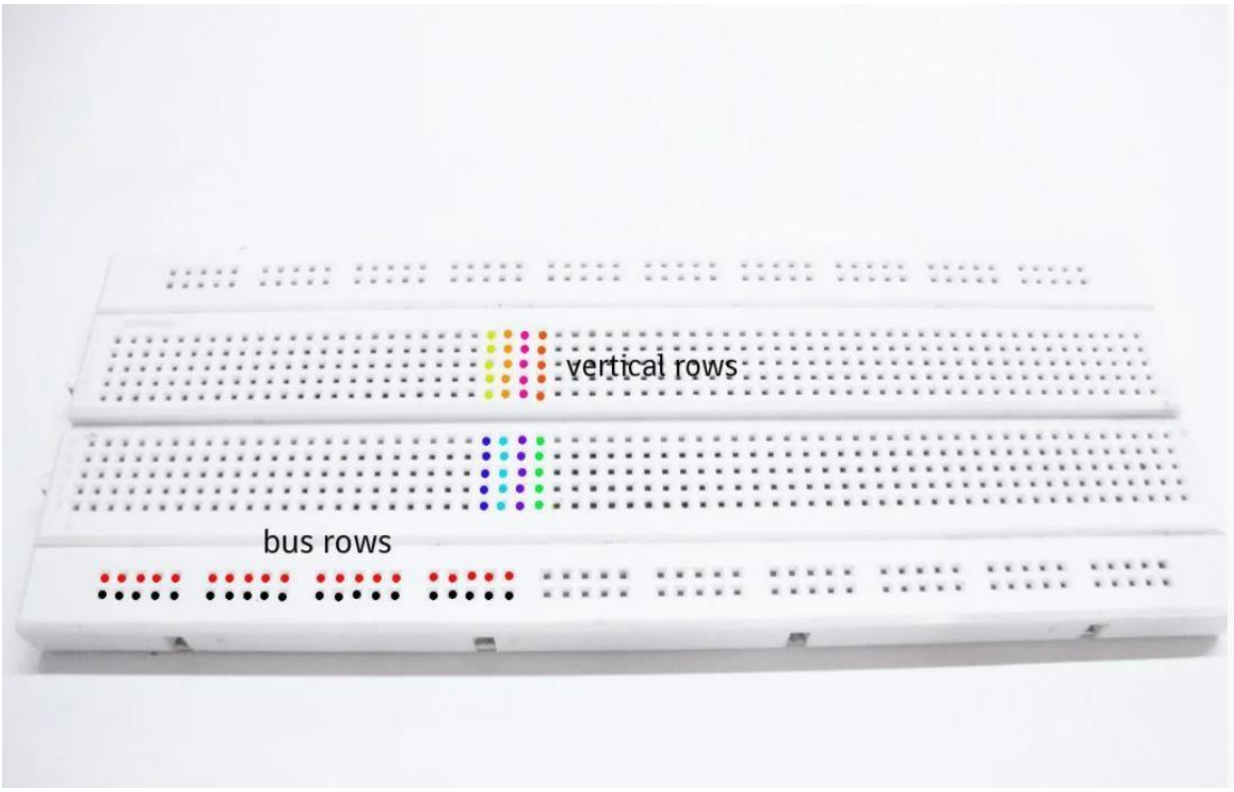
  Configure the MCP2551 CAN-BUS transceiver to match the STM32's CAN peripheral settings for optimal performance and compatibility.

  Develop data acquisition routines on the STM32 board to interface with sensors and prepare data for CAN transmission.

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

4.3.9. Electronic Bread Board

- A breadboard is used for building temporary circuits.
- Allows easy removal and replacement of components.
- Ideal for demonstrating a circuit's action and reusing components in new circuits.
- Made of a plastic block with a matrix of electrical sockets.
- Sockets are sized for thin connecting wires, component wires, or pins of ICs and transistors
- .Sockets are connected in rows of five.
- Rows are spaced 2.54 mm apart, matching the pin spacing for ICs and many other components.



4.3.10. Rainbow and Jumper Wires

- Utilize rainbow and jumper wires to establish connections between the STM32 board, TJA1050 CAN controller, 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 ST- LINK or JTAG/SWD debug probes, developers can effectively debug firmware code using features like breakpoints and watchpoints. Additionally, STM32CubeIDE comes bundled with the GNU Arm Embedded Toolchain,

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

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 role-based 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 third-party 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 ESP32 and CAN

The STM32F407VGt6 microcontroller uses CAN communication to interface with the ESP32 module for further processing or wireless transmission:

- The CAN interface on the STM32 is configured to transmit sensor data packets to the ESP32 module.
- The CAN RX pin of the STM32 is connected to the CAN TX pin of the ESP32 module, establishing a serial communication link.
- Upon receiving data, the ESP32 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 STM32F407VGt6 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 STM32F407VGt6 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 STM32F407VGt6 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

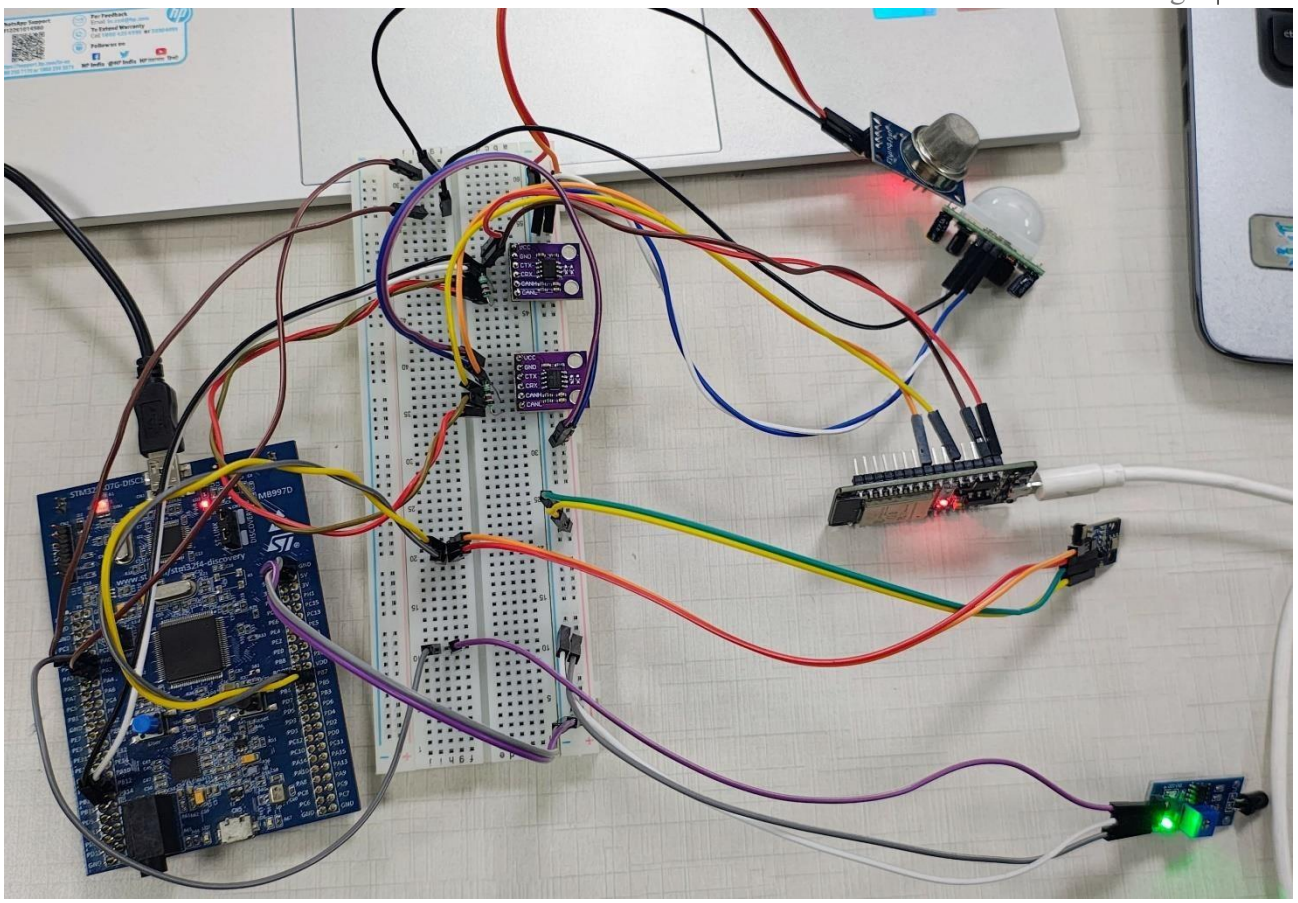


Figure: Proposed Model

Figure: Time stamp

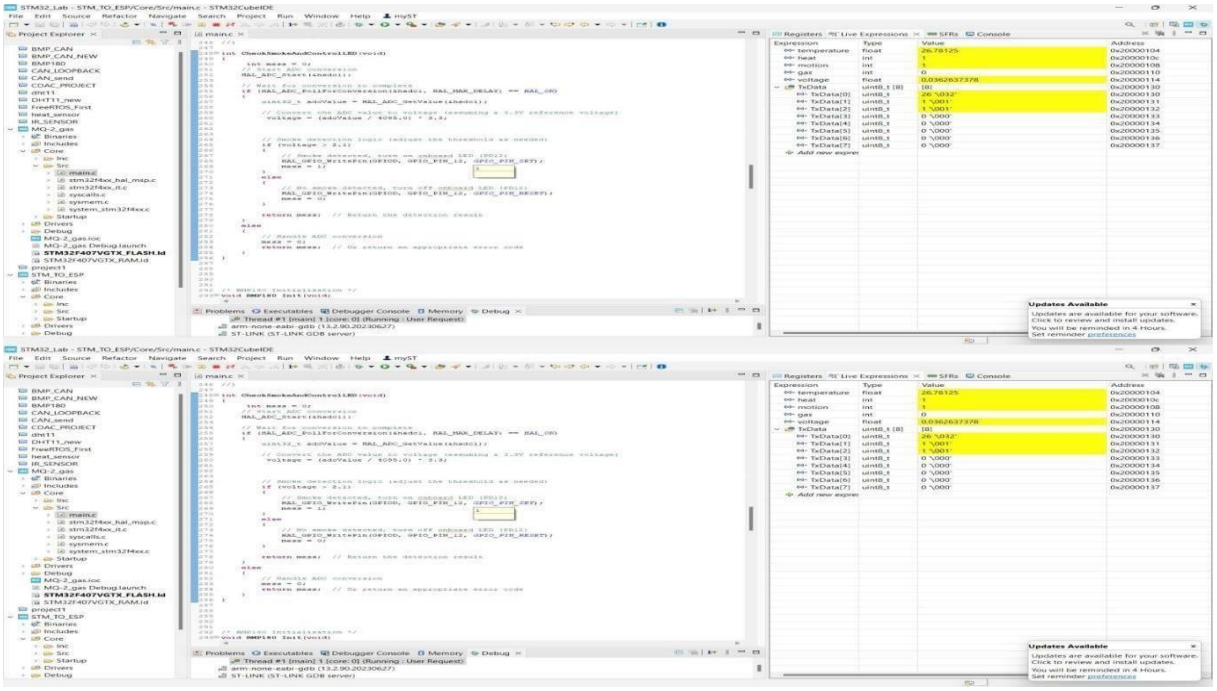


Figure: Live Expression

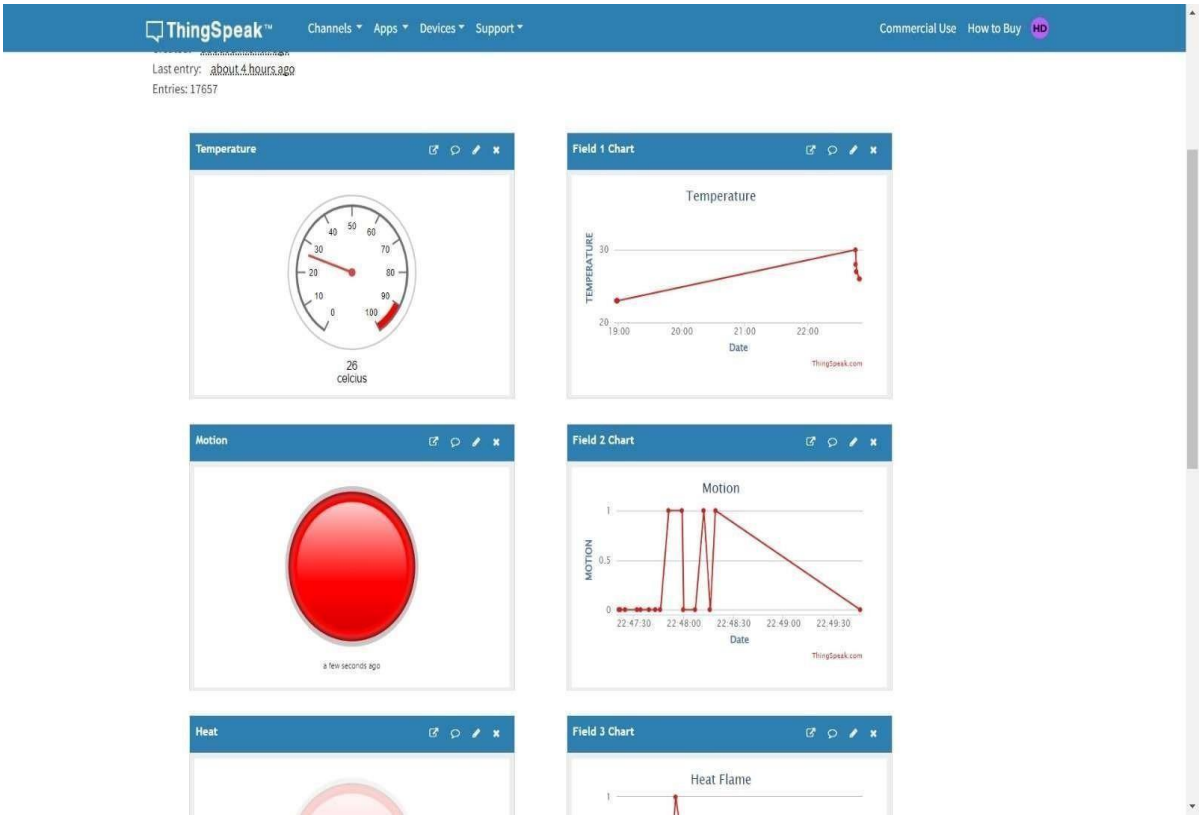


Figure: Cloud Platform Dashboard output

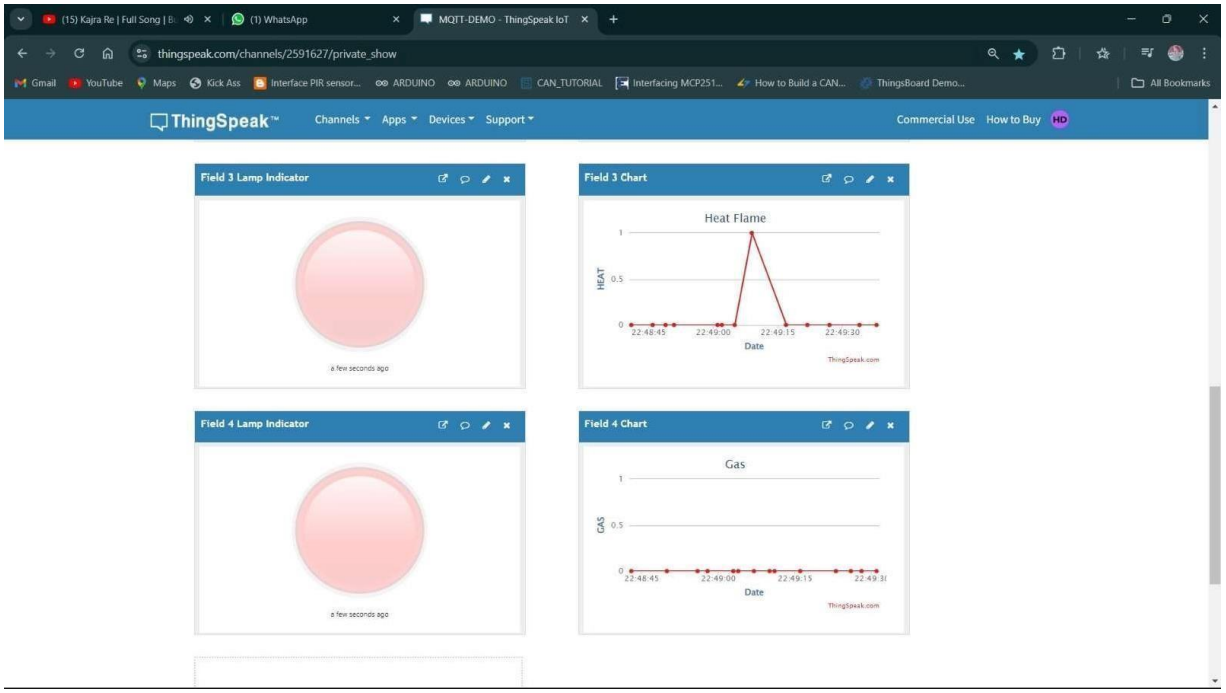


Figure: ThingSpeak output

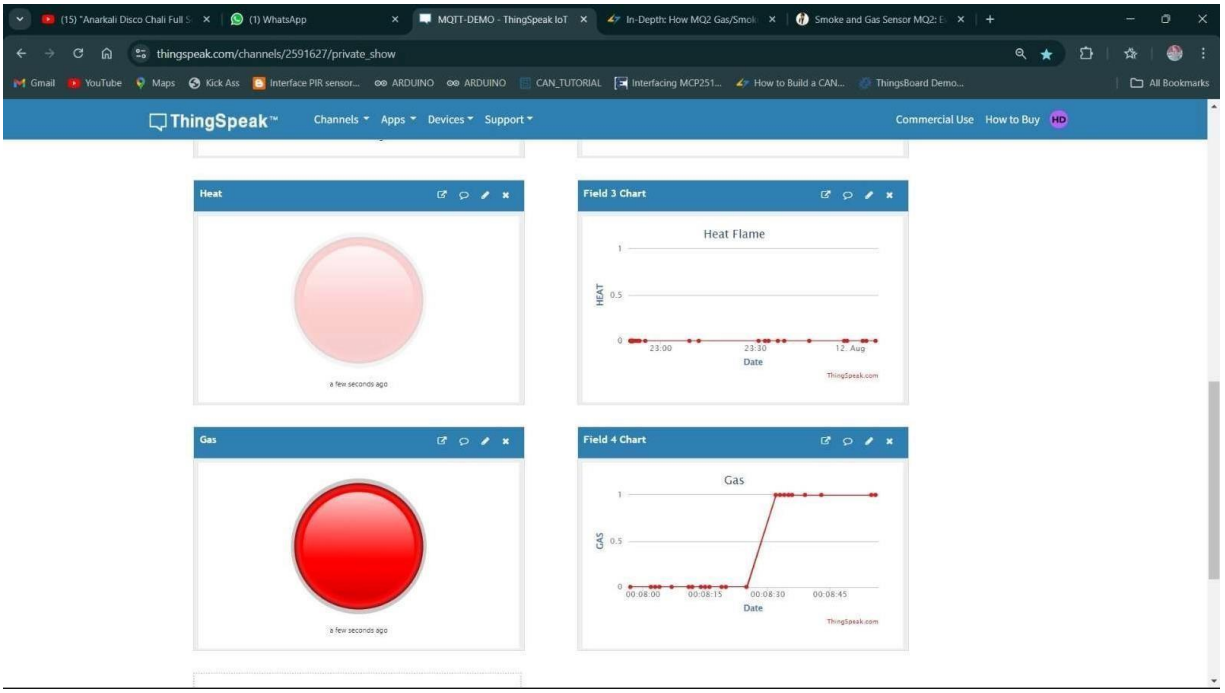


Figure: Dashboard output for Heat and Flame

8. Conclusion

Connected telemetry devices and Sensors play a crucial role in modern technology, enabling real-time data monitoring, collection, and transmission across various industries and applications.

Here are some key points to consider regarding connected telemetry devices:

- Real-Time Monitoring: Telemetry devices in your project provide real-time insights into the performance and status of the STM32F407VGt6 microcontroller, CAN communication, and connected sensors. This capability allows for proactive decision- making and timely interventions.

- Data Collection and Analysis: These devices gather diverse data types such as sensor readings from the CAN bus, temperature, and other environmental conditions. Analyzing this data helps in deriving valuable insights, identifying trends, and improving system efficiency.

- Remote Connectivity: Telemetry devices use wireless communication technologies like ESP32 for transmitting data from remote sensors or modules to central servers or cloud platforms. This connectivity enables monitoring and management of assets regardless of their geographical location.

- Scalability and Flexibility: With IoT technology, telemetry devices in your project can be integrated into existing systems and scaled to accommodate larger deployments. They can also be customized to meet specific requirements of the CAN network and sensor integrations.

- Enhanced Efficiency and Safety: Connected telemetry devices help streamline operations, optimize resource utilization, and enhance safety by providing real-time alerts and notifications in case of anomalies, faults, or hazardous conditions detected by sensors and the CAN network.

9. Future Scope

The future scope of connected telemetry devices in your project is expansive and offers significant potential for innovation and advancement. Here are some key aspects of the future scope:

- Expansion of IoT Ecosystem: The IoT ecosystem will continue to grow, with more interconnected devices, sensors, and modules integrated into your CAN network and STM32-based system. This expansion will lead to a more interconnected network, enhancing data-driven decision-making and operational efficiency.
- Integration with AI and Machine Learning: Future developments will see connected telemetry devices in your project increasingly utilizing artificial intelligence (AI) and machine learning algorithms to analyze data from the CAN bus and sensors. This will enable predictive maintenance, real-time anomaly detection, and optimized performance for your system.
- Advancements in Connectivity: Emerging wireless communication technologies, such as 5G networks and advanced low-power wide-area networks (LPWAN), will enhance the connectivity and reliability of telemetry devices in your project. This will support seamless real-time data transmission from remote sensors and modules, even in challenging environments.
- Enhanced Data Security: As telemetry devices become more integral to your system, there will be a growing emphasis on improving data security. Future developments will focus on implementing advanced encryption and authentication techniques to protect data transmitted over the CAN network and other communication channels.
- Energy Efficiency and Sustainability: Advances in energy-efficient technologies and low-power components will drive the development of telemetry devices with longer battery life and reduced environmental impact. This will support sustainable practices and enable longer-term deployment in remote or off-grid locations within your project.