Group Number – 7

Vehicle Automation using CAN and IOT

Submitted in partial fulfilment for the award of

Post Graduate Diploma in Embedded Systems and Design

From: C-DAC ACTS Pune

Guided By:

Mr. Tarun Bharani

Presented By:

Gurram Chandu	-	**240340130018**
Oindrilla Chakraborti	-	**240340130030**
Shubham Pujari	-	**240340130047**
Suhail Ok	-	**240340130048**
Vipul Surte	-	**240340130052**

Certificate

To whomsoever it may concern

This is to certify that

Mr. Gurram Chandu

Ms. Oindrilla Chakraborti

Mr. Shubham Pujari

Mr. Suhail Ok

Mr. Vipul Surte

Have successfully completed their project on

Vehicle Automation using CAN and IOT

Under the guidance of

Mr. Tarun Bharani

Project Guide

Project Supervisor

Acknowledgement

The success and final outcome of this project “Vehicle Automation using Can and IOT” required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along with the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I am highly indebted to Mr. Tarun Bharani for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We take this opportunity to thank Ms. Risha P.R. and all staff members for the cooperation provided by them in many ways.

It is our great pleasure in expressing sincere and deep gratitude toward Ms. Namrata Ailawar for her valuable guidance and constant support throughout this work.

I would like to express my gratitude towards my parents & member of CDAC ACTS, Pune for their kind co-operation and encouragement which help me in the completion of this project.

My most heartfelt thanks go to Ms. Srujana Bhamidi (Course co-ordinator, PG-DESD) who gave all the required support to me. My thanks and appreciations also go to my batchmates in developing the project and people who have willingly helped me out with their abilities.

From:

Gurram Chandu

Oindrilla Chakraborti

Shubham Pujari

Suhail OK

Vipul Surte

Contents

Abstract

This project explores the integration of Controller Area Network(CAN) and Internet of Things(IOT) technology for advancing vehicle automation. The CAN bus facilitates efficient in-vehicle communication between Electronic Control Units(ECUs), enabling real-time data exchange for critical vehicle functions. By incorporating IoT, the vehicle gains connectivity to external networks, facilitating remote monitoring, diagnostics, and advanced features. This project investigates the architecture, implementation, and potential applications of a CAN-IOT based vehicle automation system. Key focus areas include data acquisition, processing, and transmission, as well as the development of intelligent algorithms for decision making and control. The integration of these technologies hold promise for enhancing vehicle safety, efficiency, and user experience while paving the way for autonomous driving capabilities.

Project Overview

1. Introduction

In the realm of modern automotive engineering, vehicle automation is rapidly evolving with the integration of advanced technologies like Controller Area Network (CAN) and the Internet of Things (IoT). This project aims to enhance vehicle automation by leveraging CAN for communication between different vehicle components and IoT for data collection, and remote monitoring.

2. Objectives

- **Implement a CAN-based communication system** to enable effective and efficient data exchange between various vehicle modules.
- **Integrate IoT devices** to enable real-time monitoring, data logging, and remote management of vehicle systems.
- **Develop automation features** that enhance vehicle performance, safety, and user convenience through intelligent decision-making and control.

3. Components and Technologies

a. Controller Area Network (CAN)

- **CAN Protocol**: A robust vehicle bus standard designed to facilitate communication among various vehicle control units without needing a host computer.
- **CAN Nodes**: Electronic Control Units (ECUs) such as Engine Control Unit (ECU), Transmission Control Unit (TCU), Body Control Module (BCM), and others.
- **CAN Transceivers**: Interface between the CAN protocol and the physical data transmission layer.

b. Internet of Things (IoT)

- **IoT Sensors**: Devices for monitoring vehicle parameters (e.g., temperature, speed, fuel level).
- **IoT Gateways**: Devices that connect the vehicle's CAN network to the internet, enabling remote data access and control.
- **Cloud Platform**: For data storage, analysis, and visualization.
- **Mobile/ Web Application**: User interface for interacting with the vehicle's automated features and monitoring system status.

Literature Survey

The intersection of Controller Area Network (CAN) and the Internet of Things (IoT) in vehicle automation represents a significant technological advancement. This literature survey examines existing research and developments relevant to the integration of CAN and IoT for vehicle automation, highlighting key findings, methodologies, and trends in the field.

CAN (Controller Area Network) in Vehicle Automation

a. Overview and Evolution

- **Introduction to CAN Protocol**: The CAN protocol, introduced by Bosch in 1986, is a robust vehicle bus standard that facilitates communication among various electronic control units (ECUs) within a vehicle. [1][2]
- **CAN Protocol Features**: It is known for its fault tolerance, high-speed data transmission, and real-time capabilities. These features make it suitable for automotive applications where reliability and speed are critical.

b. Applications and Advancements

- **ECU Communication**: Recent advancements focus on improving the efficiency of CAN communication and expanding its applications to new vehicle systems. Studies like those by Nascimento et al. (2020) discuss the evolution of CAN and its adaptation to modern automotive systems.
- **CAN FD (Flexible Data-rate)**: The introduction of CAN FD allows for larger data payloads and faster data rates, addressing the limitations of the traditional CAN protocol. Research by Cobo et al. (2019) highlights its advantages in modern vehicle automation.

IoT (Internet of Things) in Vehicle Automation

a. IoT Framework and Technologies

- **IoT Architecture**: IoT in vehicles involves a framework where sensors and actuators are connected through IoT gateways to cloud-based platforms. This setup enables real-time data collection and remote management.
- **Communication Protocols**: Protocols like MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) are commonly used in IoT for efficient data transmission.

b. IoT Applications in Vehicles

- **Telematics and Fleet Management**: IoT technology is extensively used in telematics for monitoring vehicle performance and fleet management. Research by Wang et al. (2018) examines how IoT enhances fleet management through real-time data and analytics.
- **Predictive Maintenance**: IoT enables predictive maintenance by analysing data trends to forecast potential vehicle issues before they occur. Studies by Lee et al. (2020) illustrate the application of machine learning algorithms in predictive maintenance.

Integration of CAN and IoT for Vehicle Automation

a. Synergy Between CAN and IoT

- **Data Integration**: Combining CAN with IoT involves integrating CAN-based vehicle data with IoT platforms for enhanced monitoring and control. Studies such as those by Su et al. (2021) explore methods for integrating CAN data into IoT systems to improve vehicle automation.
- **Remote Diagnostics and Control**: IoT-enabled remote diagnostics and control systems use CAN data to provide real-time vehicle status updates and remote management capabilities. Research by Zhao et al. (2022) discusses advancements in remote vehicle control and diagnostics through IoT integration.

Future Trends and Research Directions

a. AI and Machine Learning

- **Advanced Algorithms**: Future research is likely to focus on integrating artificial intelligence and machine learning algorithms with CAN and IoT systems to enhance decision-making and automation. Studies by Garcia et al. (2024) explore the potential of AI in vehicle automation and predictive analytics.

b. V2X Communication

- **Vehicle-to-Everything (V2X)**: The development of V2X communication systems is expected to play a significant role in future vehicle automation by enabling vehicles to communicate with infrastructure, other vehicles, and pedestrians. Research by Liu et al. (2024) provides insights into V2X technologies and their impact on vehicle automation.

System Requirements

STM32F407VGT6 :

Figure 1: STM32F407VGT6 Microcontroller Board

The STM32F4DISCOVERY Discovery kit allows users to easily develop applications with the STM32F407VG high-performance microcontroller with the Arm® Cortex®-M4 32-bit core. It includes everything required either for beginners or experienced users to get started quickly. Based on STM32F407VG, it includes an ST-LINK/V2-A embedded debug tool, one ST MEMS digital accelerometer, one digital microphone, one audio DAC with integrated class D speaker driver, LEDs, push-buttons and a USB OTG Micro-AB connector. Specialized add on boards can be connected by means of the extension header connectors. The STM32F4DISCOVERY Discovery kit comes with the STM32 comprehensive free software libraries and examples available with the STM32CubeF4 MCU Package.

Features :

The STM32F4DISCOVERY offers the following features:

• STM32F407VGT6 microcontroller featuring 32-bit Arm®(a) Cortex®-M4 with FPU core, 1-Mbyte Flash memory, 192-Kbyte RAM in an LQFP100 package

• USB OTG FS

• ST MEMS 3-axis accelerometer

• ST-MEMS audio sensor omni-directional digital microphone

• Audio DAC with integrated class D speaker driver

• User and reset push-buttons

• Eight LEDs:– LD1 (red/green) for USB communication– LD2 (red) for 3.3 V power on– Four user LEDs, LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)– Two USB OTG LEDs, LD7 (green) VBUS and LD8 (red) over-current

• Board connectors:– USB with Micro-AB– Stereo headphone output jack– 2.54 mm pitch extension header for all LQFP100 I/Os for quick connection to prototyping board and easy probing

• Flexible power-supply options: ST-LINK, USB VBUS, or external sources

• External application power supply: 3 V and 5 V

• Comprehensive free software including a variety of examples, part of STM32CubeF4 MCU Package, or STSW-STM32068 for using legacy standard libraries

• On-board ST-LINK/V2-A debugger/programmer with USB re-enumeration capability: mass storage, Virtual COM port, and debug port

• Support of a wide choice of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE.

SN65HVD230 CAN Transceivers:

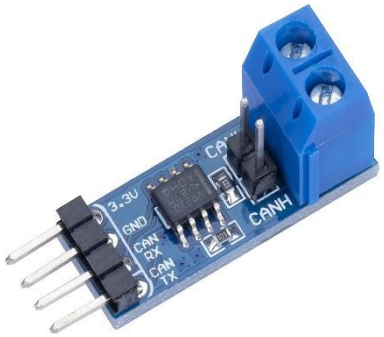

Figure 2: CAN Transceiver

The SN65HVD230is a high-speed CAN, fault-tolerant device that serves as the interface between a CAN protocol controller and the physical bus. The SN65HVD230device provides differential transmit and receive capability for the CAN protocol controller and is fully compatible with the ISO-11898 standard, including 24V requirements. It will operate at speeds of up to 1 Mb/s. Typically, each node in a CAN system must have a device to convert the digital signals generated by a CAN controller to signals suitable for transmission over the bus cabling (differential output). It also provides a buffer between the CAN controller and the high-voltage spikes that can be generated on the CAN bus by outside sources.

Some of its features are:

• Supports1Mb/s operation

• ImplementsISO-11898standardphysical layer requirements

• Suitablefor12Vand24Vsystems

• Detection of ground fault(permanent dominant) on TXD input

• Power-on Reset and voltage brown-out protection

• An unpowered node or brown-out event will not disturb the CAN bus

• Low current standby operation

• Protection against damage due to short-circuit conditions (positive or negative battery voltage)

• Upto112nodescanbeconnected

• High-noise immunity due to differential bus implementation

• Temperature ranges:-Industrial (I):-400C to +850C- Extended (E):-400C to +1250C

Communication between two STM32 boards using the CAN protocol. CAN (Controlled Area Network) Protocol is a way of communication between different devices but under certain rules. These rules must be followed when a message is transmitted over the CAN bus.

Shown below is the Standard CAN Frame

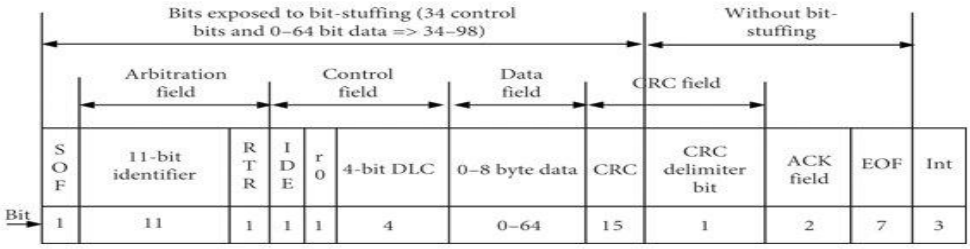

Figure 3: CAN Standard Data Frame

- The **CRC** and **ACK** will be handled by the **HAL Library**.
- **Identifier** is the ID of the transmitting Device,
- **RTR** (Remote Transmission Request) specifies if the data is Remote frame or Data frame,
- **IDE** specifies if we are using Standard ID or Extended ID,
- **'r'** is the Reserved bit DLC specifies the data length in Bytes,
- **Data Field** is where we can send the data, which should be up to 8 bytes,
- **Checksum and DEL** are the CRC data and it's Delimiter,
- **ACK and DEL** is the acknowledgment bit and its Delimiter.

Step Motor 28BYJ-48 :

Figure 4: Step Motor(used for wipers)

The 28BYJ-48 is a popular and widely used stepper motor, often found in hobbyist and educational projects. It is a unipolar stepper motor that is known for its low cost and ease of use. Here's a detailed overview of the 28BYJ-48 stepper motor:

Specifications

- **Type**: Unipolar Stepper Motor
- **Step Angle**: 5.625 degrees per step (64 steps per revolution)

- **Voltage**: Typically 5V (common in many applications)
- **Current**: About 340mA per phase
- **Number of Phases**: 4
- **Torque**: Approximately 34 oz-in (240 m Nm)
- **Mounting Type**: Typically comes with a 5mm shaft

Operating Principle: The 28BYJ-48 is a unipolar stepper motor, which means it has a centre tap on each of its windings. The motor's movement is controlled by energizing different combinations of the four coils to achieve stepping motion.

Rain Sensors :

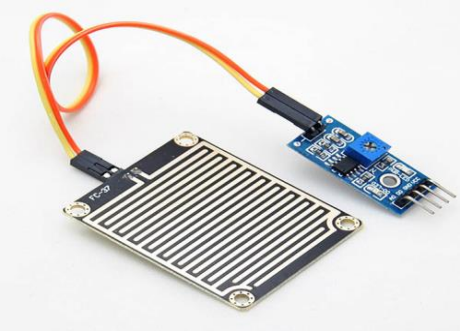

Figure 5: Rain Sensor

A rain sensor typically consists of a rain detection plate (the sensor) and a control module that interfaces with a microcontroller like the STM32.

Components of the Rain Sensor Module

- **Rain Detection Plate**: This is the main sensor that detects the presence of water droplets. It is usually made of a PCB with exposed conductive paths.
- **Control Module**: This module processes the signal from the detection plate. It typically includes:
    - **Comparator**: Compares the signal from the sensor with a reference voltage to determine if rain is detected.
    - **Digital Output**: Provides a high or low signal depending on whether rain is detected.
    - **Analog Output**: Provides a variable voltage proportional to the amount of rain detected.
    - **Potentiometer**: Adjusts the sensitivity of the rain detection.

Working Principle

- When raindrops fall on the detection plate, the resistance between the conductive paths decreases.
- This change in resistance is detected by the control module, which converts it into a voltage level.
- The module provides both analog and digital outputs:
    - **Digital Output (DO)**: It goes LOW when rain is detected and HIGH when no rain is detected.

- **Analog Output (AO)**: It provides a varying voltage corresponding to the amount of rain detected.

DHT11 :

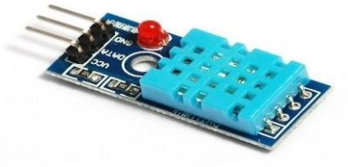

Figure 6: DHT11(Temperature and Humidity)

The DHT11 is a popular digital temperature and humidity sensor. It's often used in DIY electronics and hobbyist projects due to its low cost and ease of use.

Specifications

- **Temperature Range**: 0°C to 50°C with ±2°C accuracy
- **Humidity Range**: 20% to 80% relative humidity with ±5% RH accuracy
- **Operating Voltage**: 3.3V to 5.5V
- **Communication Protocol**: Single-wire digital interface
- **Response Time**: Approximately 1 second
- **Power Consumption**: Low (typically less than 0.5mA during measurement)

Working Principle

The DHT11 sensor uses a capacitive humidity sensor and a thermistor to measure the humidity and temperature, respectively. Here's how it works:

- **Humidity Measurement**: The sensor uses a capacitive hygrometer, which changes capacitance based on the humidity level. This change in capacitance is converted into a digital signal.
- **Temperature Measurement**: The sensor uses a thermistor, which changes resistance with temperature. This resistance is measured and converted into a digital signal.

The sensor outputs a digital signal that represents both temperature and humidity values, which can be read by a microcontroller or other digital device.

ESP32 :

Figure 7: ESP32 Microcontroller

The ESP32 is a powerful and versatile microcontroller with integrated Wi-Fi and Bluetooth capabilities, making it an excellent choice for IoT (Internet of Things) applications.

Wi-Fi Connectivity:

- **IoT Applications:** The ESP32's Wi-Fi capability makes it ideal for IoT applications where it needs to send data to the cloud or interact with other Wi-Fi-enabled devices.
- **Access Point and Client Modes:** The ESP32 can function as a Wi-Fi access point, a client, or both, which provides flexibility in how it connects to a network.

GPIO Pins:

- **Digital and Analog I/O:** The ESP32 has a rich set of General-Purpose Input/Output (GPIO) pins, many of which support PWM, ADC, DAC, I2C, SPI, and other protocols. This allows it to interface with a variety of sensors, actuators, and other peripherals.

Development Support:

- **Programming Environments:** The ESP32 can be programmed using various environments, including Arduino IDE, Espressif's own ESP-IDF, and even MicroPython. This makes it accessible to a wide range of developers, from hobbyists to professionals.

Wi-Fi Connection:

- The ESP32 uses its Wi-Fi capability to connect to a local Wi-Fi network and then communicates with the ThingSpeak platform over the internet.

Data Upload:

- The data received from the STM32F407VGT6 is uploaded to ThingSpeak, where it can be stored, visualized, and analyzed. ThingSpeak offers tools to create real-time charts, alerts, and triggers based on the incoming data.

UART :

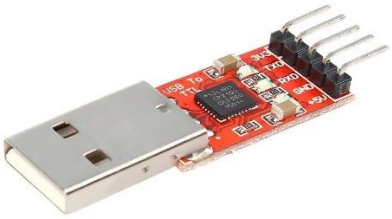

Figure 8: UART Bus

UART stands for **Universal Asynchronous Receiver/Transmitter**. It is a widely used hardware communication protocol that facilitates serial communication between devices. UART is fundamental in embedded systems for transmitting and receiving data between microcontrollers and peripherals like sensors, displays, and communication modules.

Asynchronous Communication:

- **No Clock Signal**: UART does not use a shared clock signal. Instead, it relies on a pre-agreed baud rate (speed of communication) to synchronize data transmission and reception.
- **Start and Stop Bits**: Each data frame begins with a start bit and ends with one or more stop bits, allowing the receiver to identify the beginning and end of each byte of data.
- **Baud Rate**: The speed of data transmission is determined by the baud rate, which must be the same for both the transmitter and receiver. Common baud rates include 9600, 115200, etc.

Data Framing:

- **Data Bits**: Typically, UART data frames contain 8 data bits, although 7 or 9 bits are also possible.
- **Parity Bit**: An optional bit used for basic error checking, adding either an even or odd parity bit to ensure the total number of 1s is either even or odd.
- **Error Detection**: While UART includes simple error detection through parity bits, it lacks complex error handling features like those found in more advanced protocols.

Simple, Point-to-Point Communication:

- **TX and RX Lines**: UART uses two primary lines for communication—TX (Transmit) and RX (Receive). Data sent from the TX line of one device is received by the RX line of the other.
- **Full-Duplex**: UART allows full-duplex communication, meaning data can be sent and **received simultaneously.**

Working Principle:

 Transmitter Side:

- The transmitter converts parallel data (e.g., an 8-bit byte) into a serial stream of bits. It appends a start bit, possibly a parity bit, and a stop bit to frame the data.
- This serial data is then sent out via the TX line at the agreed baud rate**.**

 Receiver Side:

- The receiver listens on its RX line. When it detects a start bit, it begins reading the incoming serial data.
- The received serial data is then converted back into parallel data (typically an 8-bit byte), and the start, parity, and stop bits are stripped away.

ThingSpeak :

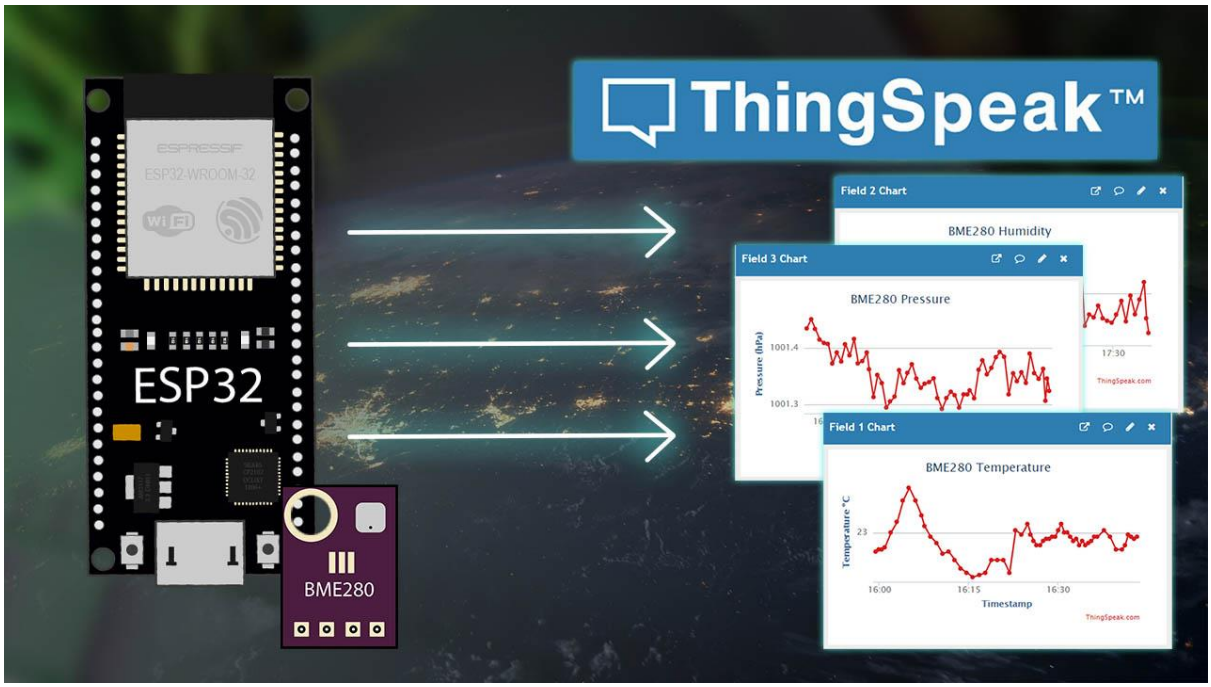

Figure 9: ThingSpeak Cloud Platform

ThingSpeak is an IoT (Internet of Things) analytics platform service that allows users to aggregate, visualize, and analyze live data streams in the cloud. It is widely used for collecting sensor data, performing real-time analytics, and triggering actions based on the data.

Data Collection:

- **Channels:** ThingSpeak organizes data into channels, each of which can store multiple data streams (fields). Each field in a channel can represent a different type of data, such as temperature, humidity, or any other sensor data.
- **API Keys:** To interact with ThingSpeak channels, users need to use API keys, ensuring secure data transmission. Each channel has a unique Write API Key for sending data and a Read API Key for retrieving data.

Real-Time Data Visualization:

- **Charts and Graphs:** ThingSpeak provides built-in tools for creating real-time charts and graphs, making it easy to visualize incoming data without needing external tools.
- **Public and Private Views:** Channels can be set to public or private, allowing data to be shared openly or restricted to specific users.

Triggers and Alerts:

- **Event-Based Triggers:** ThingSpeak can trigger actions based on the data received. For example, if the temperature exceeds a certain threshold, an alert can be sent via email, SMS, or social media.
- **Integration with Other Services:** ThingSpeak can connect with other services like IFTTT (If This Then That), allowing users to create complex workflows based on their data.

Cloud Storage:

- **Historical Data:** ThingSpeak stores data over time, allowing users to access historical data for long-term analysis and trend detection.
- **Exporting Data:** Users can export their data in various formats (e.g., CSV) for offline analysis or integration with other tools.

Why Use ThingSpeak?

- **Ease of Use:** ThingSpeak is user-friendly, with a straightforward interface for setting up channels, visualizing data, and creating triggers.
- **Built-In Tools:** It offers a variety of built-in tools for data visualization, processing, and analysis, which can be customized to meet specific needs.
- **Cloud-Based:** As a cloud service, ThingSpeak allows for remote monitoring and control, making it ideal for IoT applications where users need to access data from anywhere.
- **Community and Support:** ThingSpeak has an active community and extensive documentation, making it easy to find resources and support.

BLOCK DIAGRAM

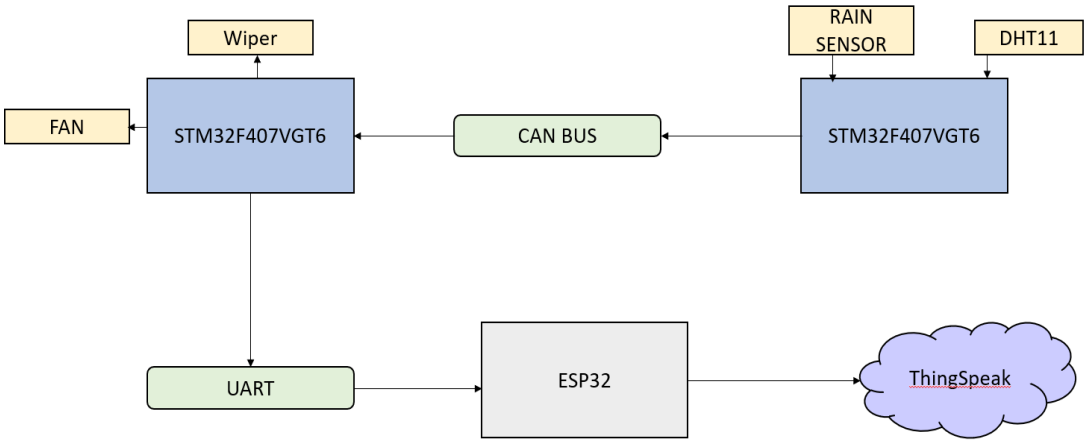

Figure 10: Block Diagram of Vehicle Automation

Let's go through the block diagram step-by-step, analysing the interactions between the components in more detail:

1. STM32F407VGT6 Microcontrollers

- **Microcontroller 1 (Left Side)**:
  - **Connected Devices:**
    - **Wiper**: This microcontroller controls the wiper. It might adjust the speed or activation of the wiper based on data received from the other microcontroller via the CAN bus.
    - **FAN**: Similarly, this microcontroller controls the operation of the fan, likely based on environmental conditions like temperature and humidity, or even the detection of rain.
  - **Communication**:
    - **CAN Bus**: This microcontroller receives data from Microcontroller 2 (on the right side) over the CAN bus. This data could include information from the rain sensor and DHT11, which would then be used to control the wiper and fan.
- **Microcontroller 2 (Right Side)**:
  - **Connected Devices:**
    - **RAIN SENSOR**: This sensor detects the presence of rain. The microcontroller processes this data to determine whether it is raining and the intensity of the rain, if applicable.

        - **DHT11**: This is a temperature and humidity sensor. It provides data on the current environmental conditions, which the microcontroller processes to determine the appropriate actions (e.g., whether the fan needs to be activated).
    - **Communication**:
        - **CAN Bus**: This microcontroller sends processed data from the rain sensor and DHT11 to Microcontroller 1 via the CAN bus. This communication ensures that the appropriate actions (e.g., turning on the wiper or fan) are taken based on real-time environmental data.

2. CAN Bus

- **Purpose**:
    - The CAN bus connects the two STM32F407VGT6 microcontrollers, enabling them to share data and coordinate their actions.
- **Data Flow**:
    - Microcontroller 2 sends sensor data (from the rain sensor and DHT11) over the CAN bus to Microcontroller 1.
    - Microcontroller 1 uses this data to make decisions about controlling the wiper and fan.
- **Advantages**:
    - This setup allows for real-time, robust communication between the microcontrollers, which is essential for synchronized control of the wiper and fan based on the environmental conditions.

3. UART Communication

- **Purpose**:
    - UART is used for communication between Microcontroller 1 (which also controls the actuators) and the ESP32. This serial communication method is straightforward and effective for transferring data between devices at a lower speed compared to CAN.
- **Data Flow**:
    - Microcontroller 1 sends data (possibly a combination of sensor readings, system status, and control commands) to the ESP32. This data might be consolidated before transmission to reduce the overhead and ensure efficient use of the communication channel.

4. ESP32

- **Purpose**:
    - The ESP32 microcontroller is responsible for sending the data it receives from Microcontroller 1 (via UART) to the cloud platform ThingSpeak. The ESP32 has built-in Wi-Fi capabilities, making it ideal for IoT applications where remote monitoring and control are required.
- **Data Flow**:
    - Upon receiving data from the STM32F407VGT6 via UART, the ESP32 processes this data and uploads it to ThingSpeak. This might include sensor readings (e.g.,

temperature, humidity, rain status) and actuator status (e.g., whether the wiper or fan is on).

- **Cloud Integration**:
    - ThingSpeak can be used to visualize this data in real-time, create alerts, or even trigger remote actions if necessary.

5. ThingSpeak

- **Purpose**:
    - ThingSpeak is a cloud-based platform that allows you to collect, visualize, and analyze data from IoT devices.
- **Functionality**:
    - The data uploaded by the ESP32 can be monitored remotely via ThingSpeak's dashboards.
    - It can also trigger alerts or actions based on specific conditions, such as turning off the fan if the temperature drops below a certain threshold.
    - Data from ThingSpeak can be used for further analysis or integrated into other systems.

Connections:

Transmitter side:

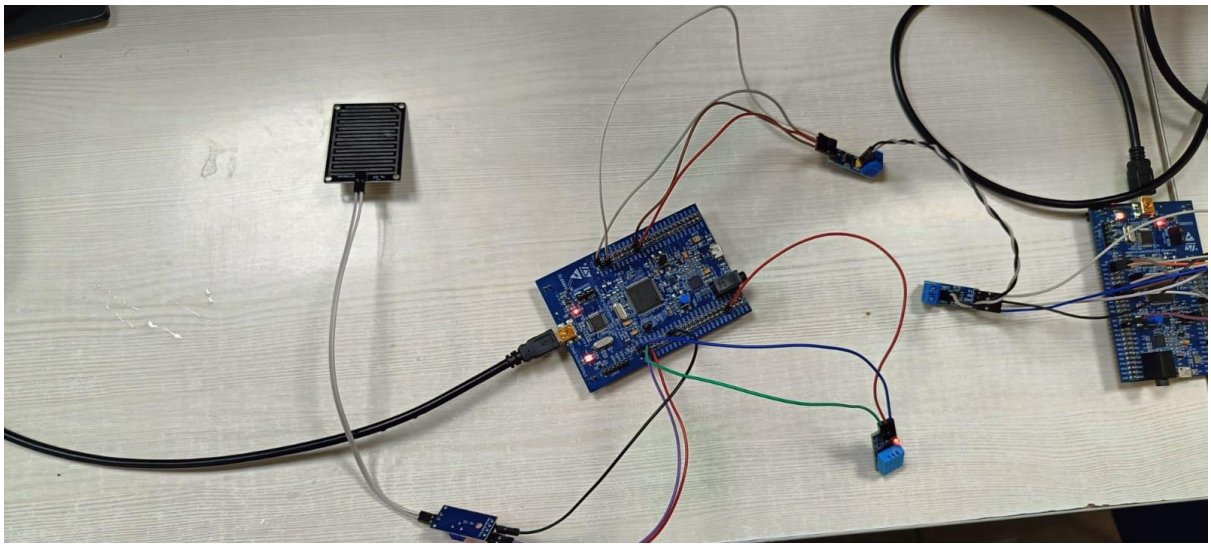

Figure 11: Rain sensor, DHT11, STM Microcontroller, CAN Bus

Receiver side:

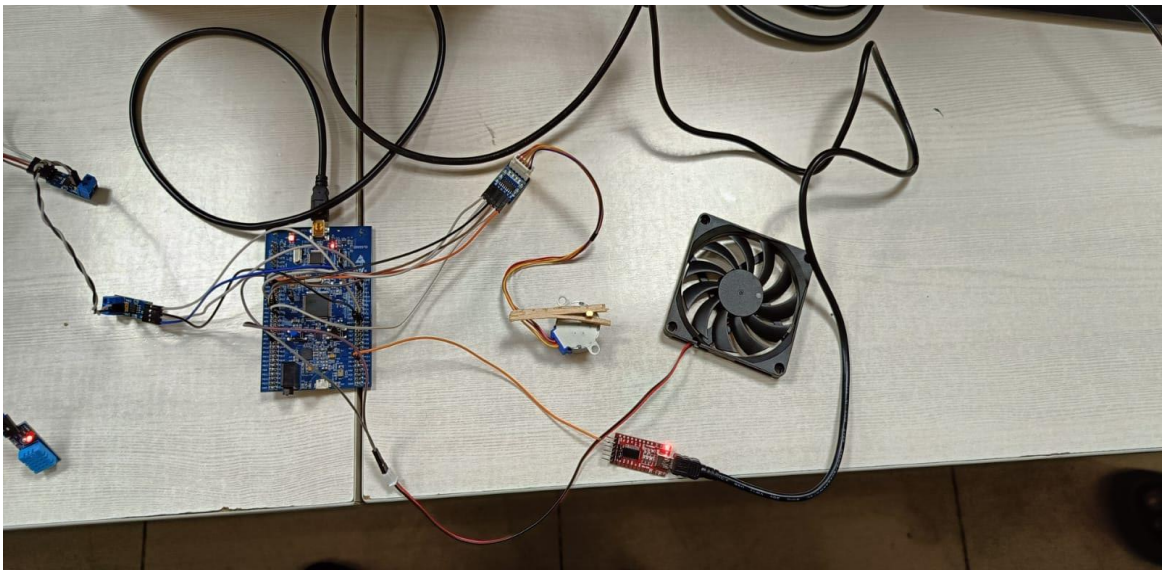

Figure 12: CAN Bus, STM Microcontroller, Step Motor(Wiper), Fan, UART

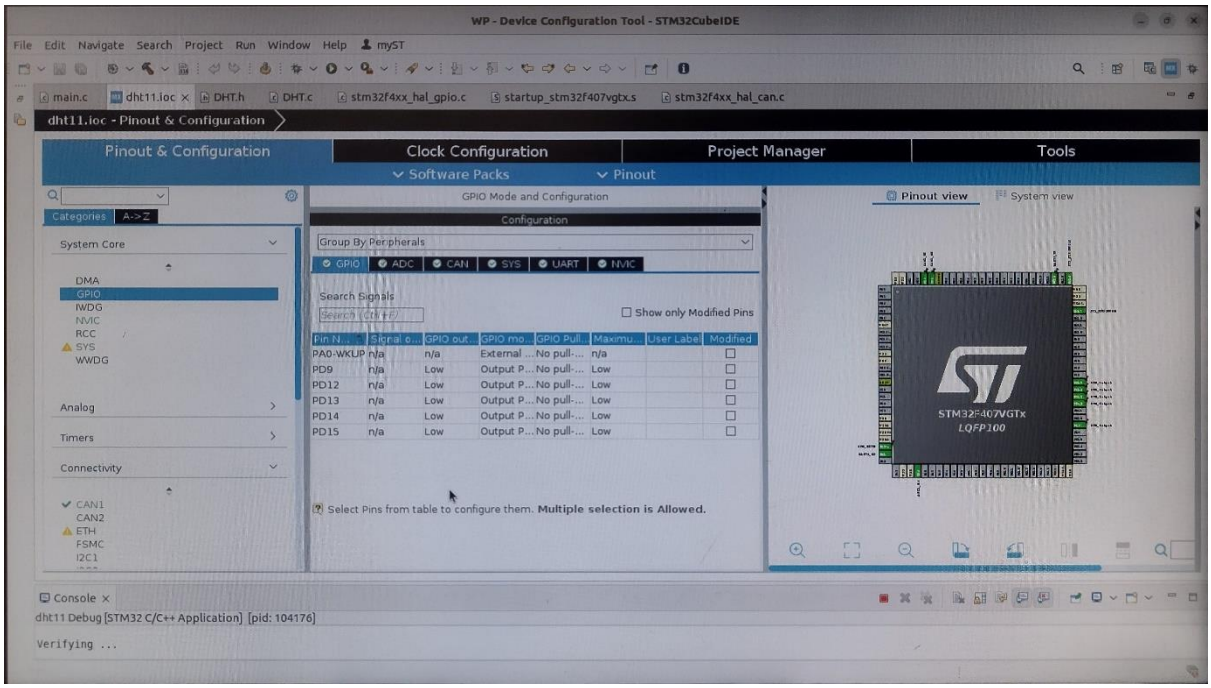

Figure 13: tx.ioc

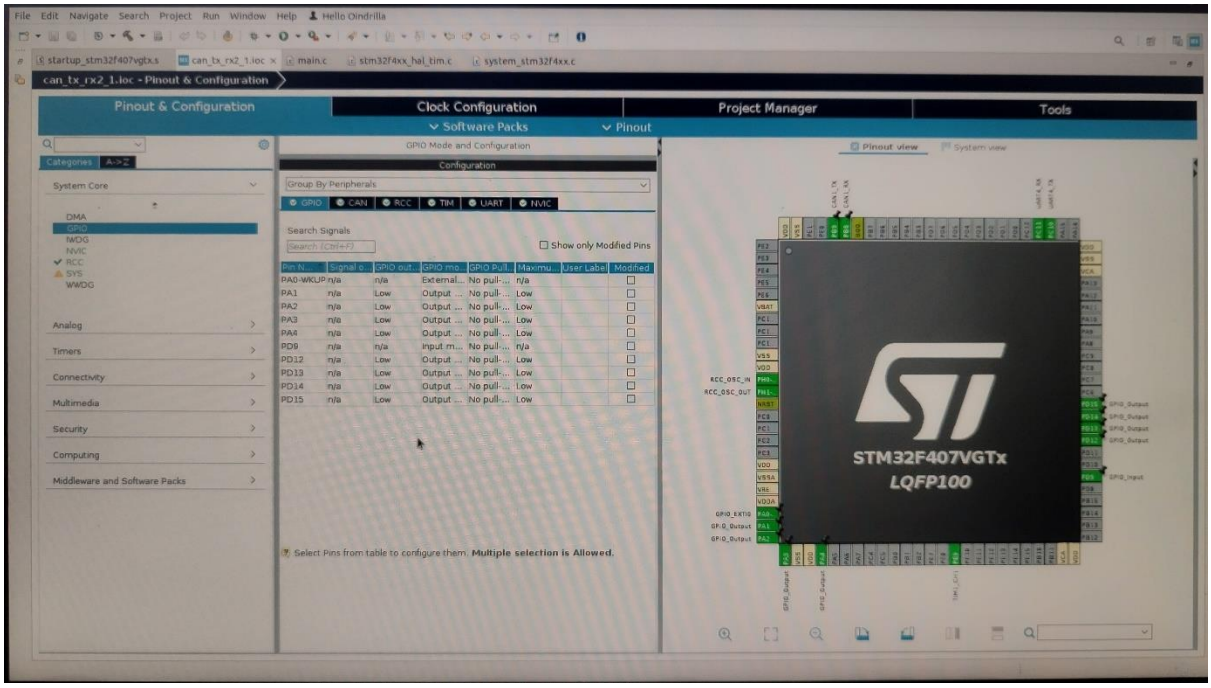

Figure 14: rx.ioc

RESULT

In this project, we designed and implemented an IoT-based environmental monitoring and control system. The system uses the STM32F407VGT6 microcontroller to gather data from various sensors and control actuators. The ESP32 microcontroller facilitates wireless data transmission to the ThingSpeak cloud platform, enabling real-time monitoring and data analysis. This system provides an efficient solution for monitoring environmental conditions such as temperature, humidity, and rainfall, with the ability to trigger responses based on sensor data.

The system successfully achieved the following objectives:

- **Real-Time Monitoring:**
    - Environmental data (temperature, humidity, and rainfall) was successfully transmitted to ThingSpeak in real-time and visualized through its dashboard interface.
- **Automated Control:**
    - The system could automatically trigger the wiper when rain was detected and control the fan based on temperature thresholds.
- **Remote Accessibility:**
    - Users could monitor and interact with the system remotely through the ThingSpeak platform, accessing real-time data and historical trends from any internet-enabled device.
- **Data Analytics:**
    - MATLAB was integrated into ThingSpeak for advanced data processing. The system could calculate averages, detect anomalies, and trigger alerts (e.g., email notifications) based on sensor data.

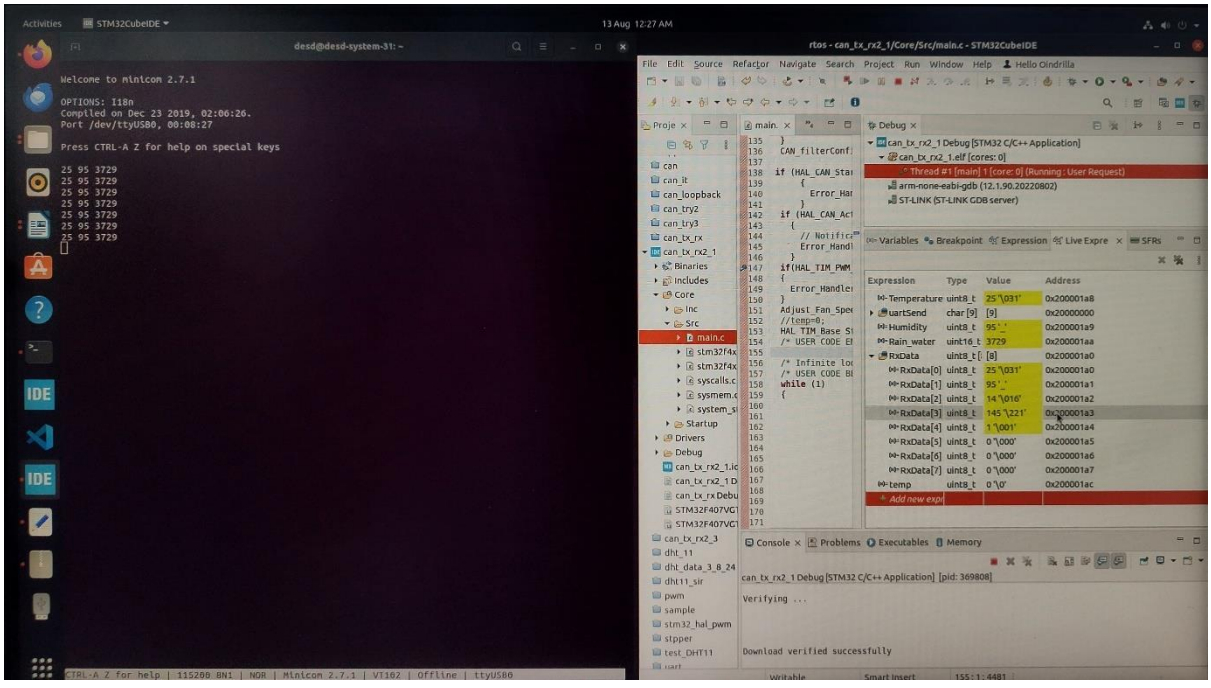

Figure 15: Result 1(STM 32 board)

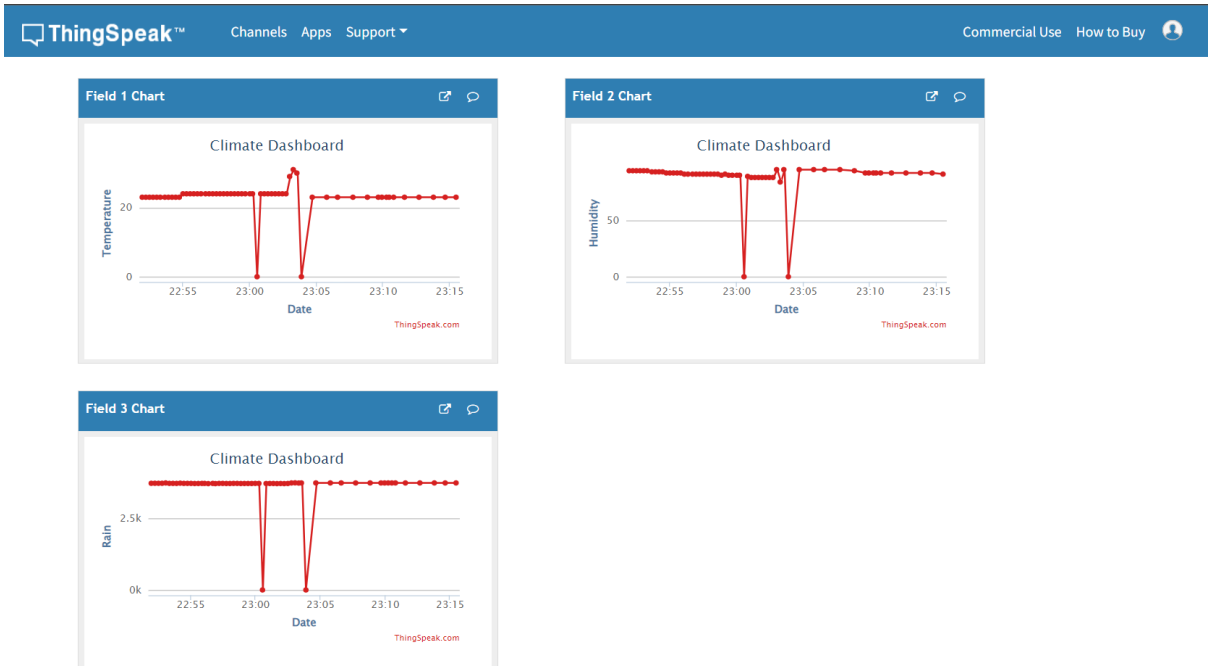

Figure 16: Output(ThingSpeak)

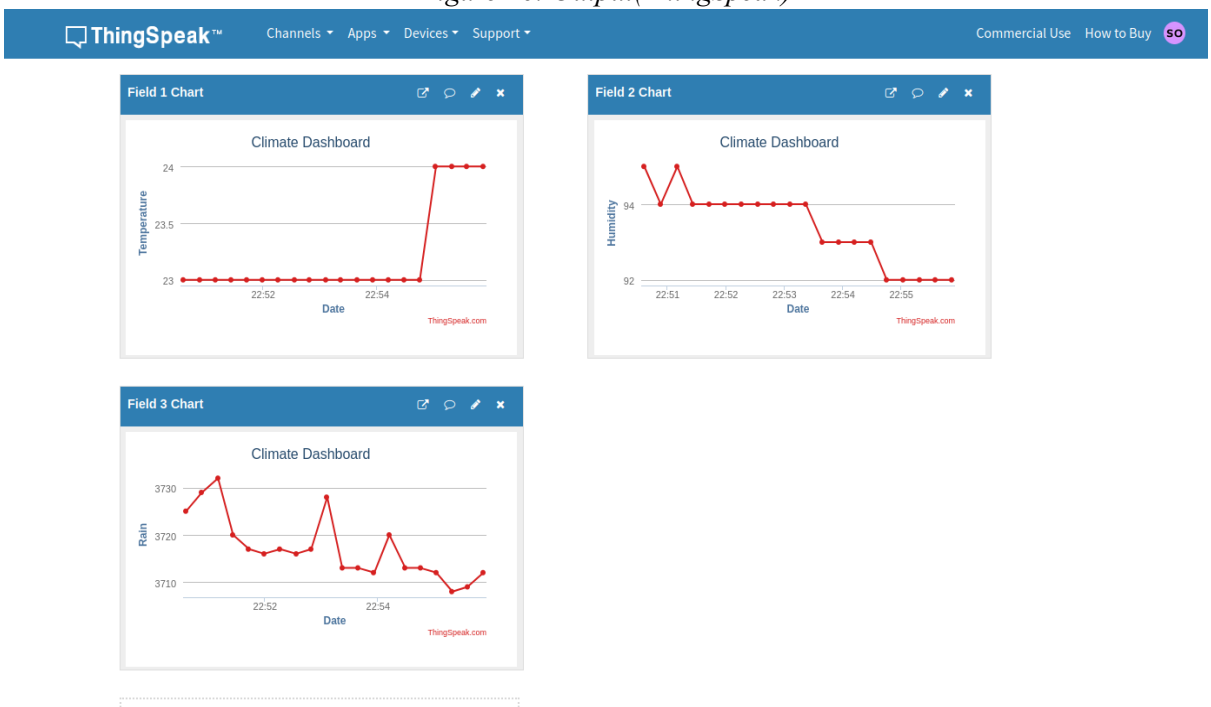

Figure 17:Output2(ThigSpeak)

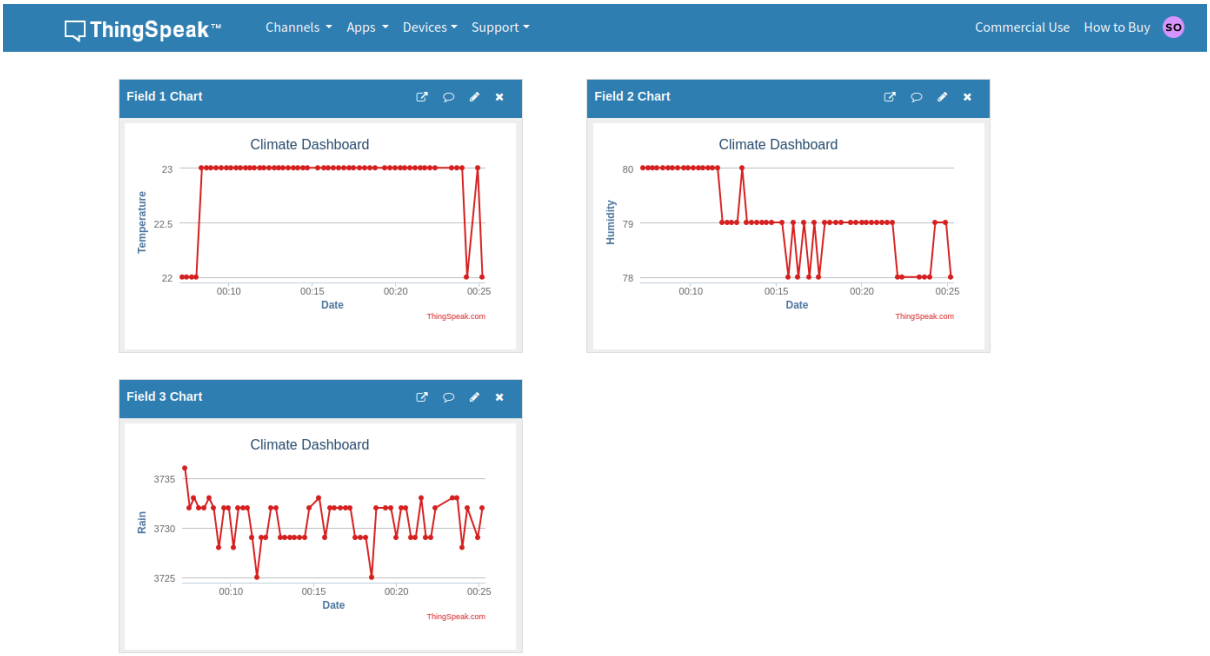

Figure 18: Output 3(ThingSpeak)

Conclusion

The project successfully implemented an IoT-based environmental monitoring and control system utilizing STM32F407VGT6, ESP32, and ThingSpeak. The system effectively gathers environmental data, processes it, and takes appropriate actions in real-time while providing users with remote access and control capabilities. This solution demonstrates the potential for deploying IoT technology in environmental monitoring and smart home applications, offering both real-time responsiveness and long-term data analysis.

Future Work

To enhance the system's capabilities, the following future improvements are proposed:

- **Integration of Additional Sensors:** To monitor more environmental parameters like air quality or soil moisture.
- **Improved Data Analytics:** Implementation of machine learning models within ThingSpeak for predictive analytics and more intelligent decision-making.
- **Mobile Application:** Developing a dedicated mobile app for better accessibility and control on-the-go.

References

- STM32F407VGT6 Microcontroller Datasheet
- ESP32 Technical Reference Manual
- ThingSpeak Documentation and API Reference
- DHT11 Sensor Datasheet
- MATLAB Integration with ThingSpeak for IoT Analytics