DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

EC-310: Microprocessors and Microcontrollers Based Design

PROJECT REPORT

SUBMITTED TO:

Asst Prof Sagheer Khan

LE Ayesha Khanum

SUBMITTED BY:

Sr No.	Name	Reg No.
1	Abeera Shahid	417494
2	Areesha Batool	413055
3	Khadija Zahra	432294
4	Maria Butt	414065
5	Shafla Rehman	412780
6	Shaheer Mukhtiar	432017

DE-44 Syn B

Submission Date: 30/12/2024

Table of Contents

Contents

Abstract

This project involves the development of a Weather Station with Thingspeak Cloud Integration using the ESP32-WROOM-32 microcontroller and a LoRa communication module. The system consists of two parts: the sender side and the receiver side. The sender side, built around the ESP32-WROOM-32 microcontroller, collects environmental data from rain, moisture, and temperature sensors. This data is transmitted to the receiver side through a LoRa module and also uploaded to the cloud for remote monitoring. The receiver side retrieves the data from the cloud, processes it, and uses it to control actuators based on predefined conditions, implementing edge computing.

The project aims to demonstrate the integration of wireless communication, sensor data collection, and cloud technologies to build a scalable, reliable, and energy-efficient system. By using the ESP32-WROOM-32, the project benefits from its wireless connectivity options, including Wi-Fi and LoRa, which facilitate long-range data transmission. The cloud-based approach allows for real-time data access and control of actuators, making the system suitable for a variety of applications such as smart agriculture, environmental monitoring, and IoT-based automation.

1 Introduction

The aim of this project is to develop an IoT enabled wireless communication system that combines cloud storage, actuator control, long-range communication, and sensor data collection. Through a network of sensors, the system gathers environmental data such as temperature, humidity, and barometric pressure. LoRa (Long Range) technology is used to wirelessly send the data to a receiver module that has an ESP32 microcontroller installed in it. The data is further communicated to a cloud platform, ThingSpeak, via the ESP32's dual capability and integrated Wi-Fi. Real-time environmental monitoring and analysis are made possible by cloud storage. Additionally, by using the gathered data to control associated actuators including fans, water pumps, and motorized curtains through the same ESP32 module, this system exhibits an advanced degree of automation.

This closed-loop system offers a scalable and effective way to improve water management, minimize manual intervention, and maintain ideal environmental conditions. Agricultural irrigation is optimized by automated water management to maintain industrial efficiency and increase crop yields. Through water resource conservation and reducing water wastage, it also advances global sustainability goals.

2 Components

- ESP32-WROOM-32 x2
- Rain Drop Moisture Detection sensor x1
- Digital Humidity and Temperature Sensor x1
- Servo Motor x1
- Micro DC Motor x1
- LORA XL1278-SMT (1x transmitter, 1x receiver)
- 2V Two Channel 10A Relay Module
- Breadboard

- Jumper wires
    - **ESP32-WROOM-32 Microcontroller**

Figure 1: ESP32-WROOM-32 microcontroller

The ESP32-WROOM-32 is a powerful and versatile microcontroller developed by Espressif Systems. It is based on the dual-core Xtensa LX6 processor and operates at a clock speed of up to 240 MHz, making it suitable for a wide range of applications. This microcontroller is equipped with 4 MB of flash memory, integrated Wi-Fi, and Bluetooth capabilities, enabling seamless wireless communication. Its compact size, low power consumption, and robust performance make it a popular choice for IoT and embedded system projects.

One of the key features of the ESP32-WROOM-32 is its support for multiple communication protocols, including UART, SPI, I2C, and CAN, which facilitate easy integration with various sensors and modules. Additionally, its built-in Wi-Fi module allows the ESP32 to connect to cloud platforms for real-time data upload and remote monitoring, while its Bluetooth functionality supports short-range data transmission and device pairing. The ESP32 also offers GPIO pins, PWM, ADC, and DAC capabilities, making it highly versatile for interfacing with actuators and other peripherals.

The ESP32-WROOM-32 was selected for our project due to its capacity to manage two communication requirements efficiently. On the sender side, it acts as the main controller, connecting to cloud services for data uploading and interacting with

LoRa modules to convey data over great distances. Its dependable connectivity features and great performance guarantee smooth and effective operation, making it the perfect option for projects needing real-time data processing and strong wireless communication.

Schematic:

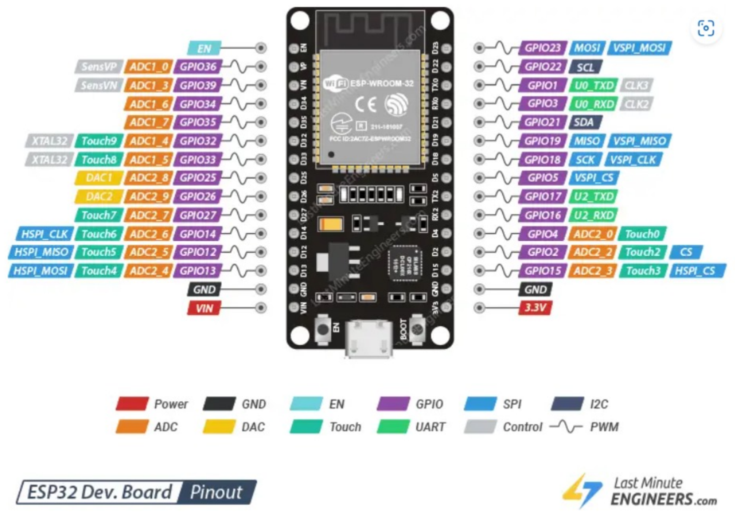


Figure 2: ESP32-WROOM-32 pinouts

- **LORA Module XL1278-SMT**

LoRa (Long Range) technology is a wireless communication protocol designed for long-distance, low-power data transmission. It operates on unlicensed ISM (Industrial, Scientific, and Medical) frequency bands, such as 433 MHz, 868 MHz, and 915 MHz, making it ideal for IoT applications. LoRa uses Chirp Spread Spectrum (CSS) modulation, which provides excellent resistance to interference and multipath fading, enabling robust communication even in challenging environments.

LoRa modules, such as the XL1278-SMT used in our project, are specifically designed to support long-range communication with minimal power consumption. They can achieve communication distances of several kilometers in open areas, making them suitable for applications requiring remote monitoring and control.

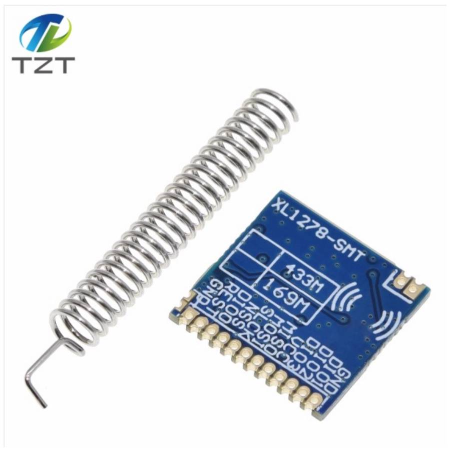


Figure 3: XL1278-SMT LoraWAN

These modules support LoRaWAN (LoRa Wide Area Network) protocols, which enable scalable networks consisting of multiple nodes and gateways, further enhancing their usability in IoT ecosystems.

Connections:

- VCC(LORA) → 3.3V (ESP32)
- GND(LORA) → GND (ESP32)
- NSS (CS) (LORA) → GPIO5 (ESP32)
- SCK(LORA) → GPIO18 (ESP32)
- MISO(LORA) → GPIO1 (ESP32)
- MOSI(LORA) → GPIO23 (ESP32)
- DIO0(LORA) → any GPIO pin, e.g., GPIO2 (ESP32)
- DIO2(LORA) → GPIO4 (ESP32)
- RESET(LORA) → any GPIO, e.g., GPIO14 (ESP32)

Schematic:

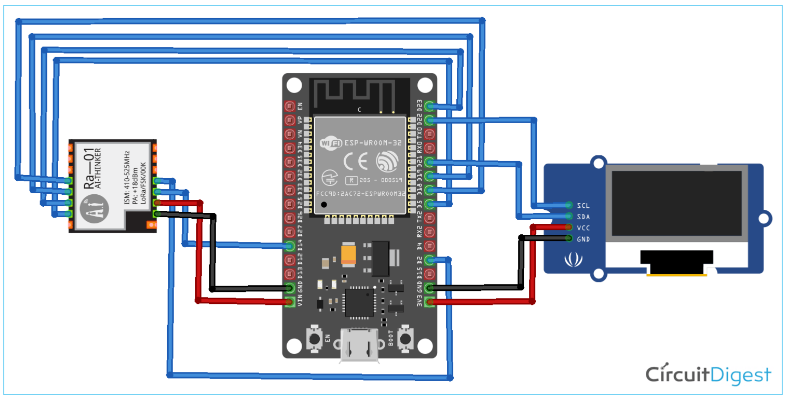
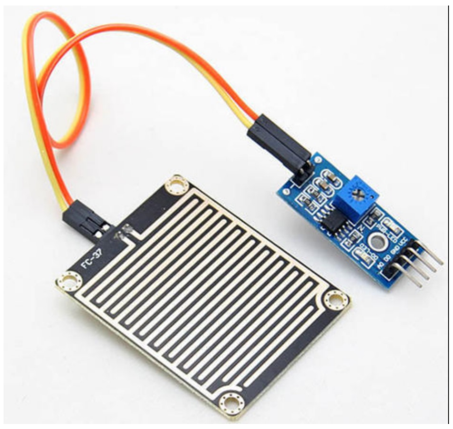


Figure 4: Interfacing LoRa with ESP32

- **Rain Drop Moisture Detection Sensor**

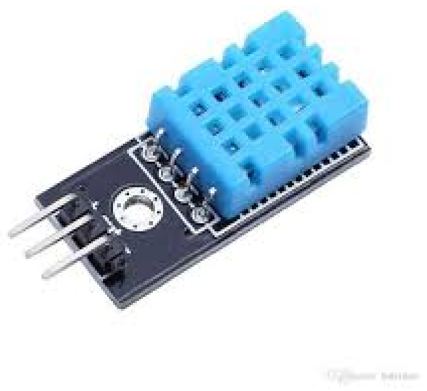


It detects the amount of water on a surface or the intensity of rainfall. It is frequently used in irrigation systems and weather monitoring.

Connections:

- VCC Pin (Rain Drop Sensor) → 3V Pin (ESP32)
- GND Pin (Rain Drop Sensor) → GND Pin (ESP32)
- Analog Output (A0) Pin (Rain Drop Sensor) → GPIO Pin (ESP32)

- **Digital Humidity and Temperature Sensor DHT-11**

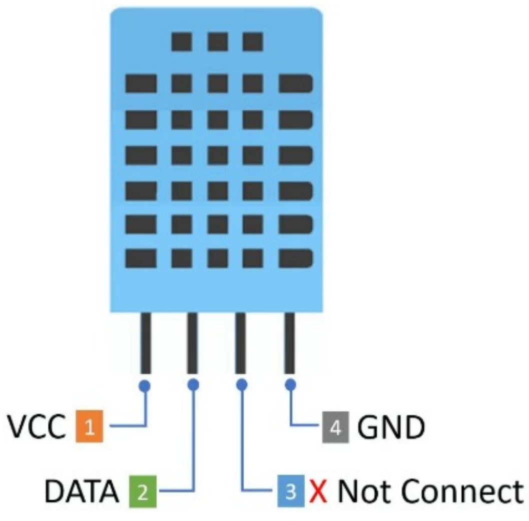


A temperature sensor is a device that measures and monitors temperature changes in its environment, converting thermal data into electrical signals. It is commonly used in weather monitoring, industrial systems, and IoT applications for accurate temperature detection and control.

Connections:

- VCC (Pin 1) – Connect to 3.3V or 5V on the ESP32 (depending on sensor variant)
- DATA (Pin 2) – Connect to any GPIO pin on the ESP32 (e.g., GPIO4). Use a 10k pull-up resistor between DATA and VCC for stable communication
- NC (Pin 3) – Not connected
- GND (Pin 4) – Connect to GND on the ESP32

Schematic:



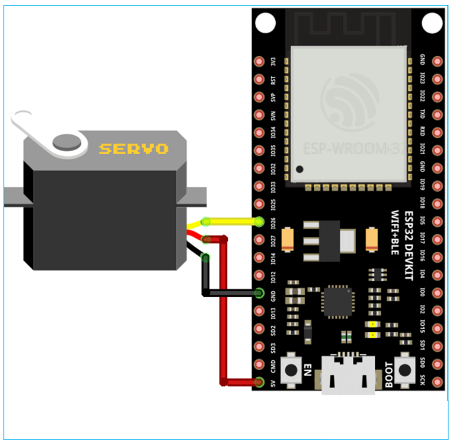
DHT11 Pin Diagram

- **Servo Motor**

Provides precise rotational movement, often used in robotic arms, automated gates, and curtain mechanisms.

Connections:

- VCC(Servo) → 5V pin (ESP32)
- GND(Servo) → GND pin (ESP32)
- Signal(Servo) → GPIO 15 (or any PWM-capable GPIO pin on ESP32)



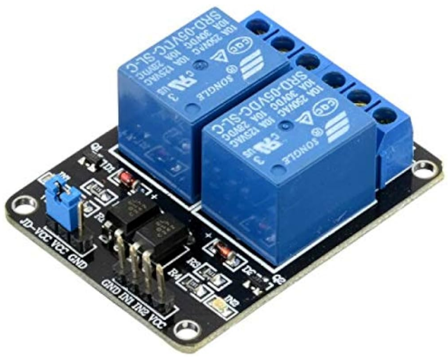
- **2A503 3 Blade Propeller Fan for DC Motor**

Frequently combined with DC motors, this device is used for cooling and airflow applications.

- **Micro DC 3V-6V *7000-7300RPM High-Speed Motor**

A DC motor is an electrical device that converts direct current (DC) electrical energy into mechanical motion. It operates based on the interaction between magnetic fields and current-carrying conductors, producing rotational motion. DC motors are widely used in automation, robotics, and control systems due to their simple design, precise speed control, and high efficiency.

- **5V Two Channel 10A Relay Module**



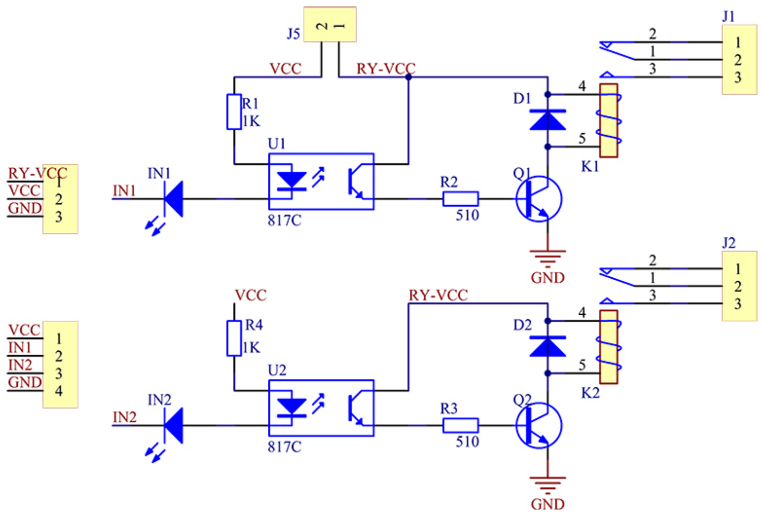
A relay is an electrically operated switch that uses a small control signal to activate or deactivate a larger electrical circuit. It is commonly used for isolating low-power control systems from high-power devices, ensuring safety and efficient operation.

Connections:

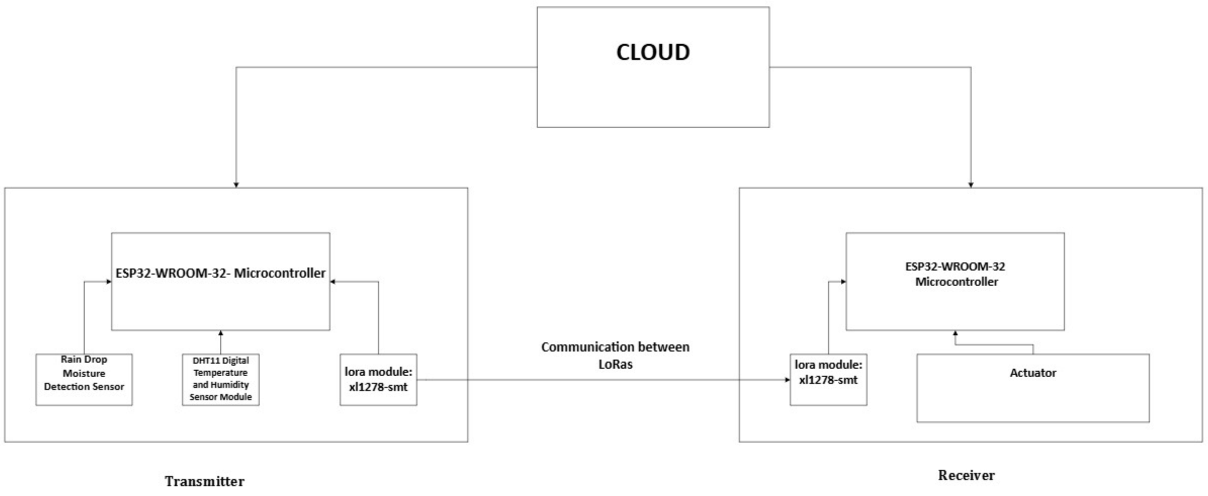
- VCC (Relay Module) → 3.3V (ESP32)
- GND (Relay Module) → GND (ESP32)

- IN1 (Relay Channel 1) → any available GPIO pin (ESP32)
- IN2 (Relay Channel 2) → another GPIO pin (ESP32)
- External 5V Power Supply → 5V pin (for powering the relay coil)

Schematic:



3 Block Diagram


CLOUD
ESP32-WROOM-32- Microcontroller
Rain Drop Moisture Detection Sensor
DHT11 Digital Temperature and Humidity Sensor Module
lora module: xl1278-smt
Communication between LoRas
ESP32-WROOM-32 Microcontroller
lora module: xl1278-smt
Actuator
Transmitter
Receiver

4 Software Implementation

4.1 Softwares used

- Arduino IDE
- ThingSpeak
- MIT App Inventor:

4.1.1 Arduino IDE:

Writing, debugging, and uploading code to microcontrollers such as the ESP32 is made possible via the open-source Arduino IDE. It offers a comprehensive library environment for integrating with sensors and communication modules, including WiFi and LoRa, as well as an intuitive sketching interface in C/C++. The Arduino IDE was used in this project for writing and debugging the code to read temperature, humidity, and barometer sensor data. LoRa module configuration for wireless data transfer. ESP32 programming for actuator control, cloud integration, and data recording. Additionally, serial monitoring is supported by the IDE, which was essential for testing and confirming the data logging and transmission procedures.

4.1.2 ThingSpeak:

It is an IoT analytics platform that enables cloud-based data processing, visualization, and aggregation. It is perfect for applications on the Internet of Things, as it offers real-time data logging and analytical capabilities. ThingSpeak was utilized in this project for recording sensor data that the ESP32 WiFi module has received. preserving information in the form of charts and logging information into database created in the form of CSV file for later use and examination. allowing for integration with outside systems so that decisions can be made using the recorded data. Using this platform we are predicting future values using the sensor data. To control relays and actuators for operations like running a water pump, fan, and curtains, stored data was retrieved using the ThingSpeak API.

4.1.3 MIT App Inventor:

For creating Android apps, MIT App Inventor offers a user-friendly, visual programming environment. Its block-based UI makes it easier to create apps and makes interacting with IoT devices simpler. In this project, MIT App Inventor was used to create an application for: Remote observation of ThingSpeak-logged sensor data. Displaying the environmental parameters' current state as well as their past patterns. Supplying an intuitive user interface for communicating with and keeping an eye on the wireless communication system. By enabling users to remotely monitor and manage the system using their cellphones, the software improved the system's usability and accessibility.

4.2 Transmitter End

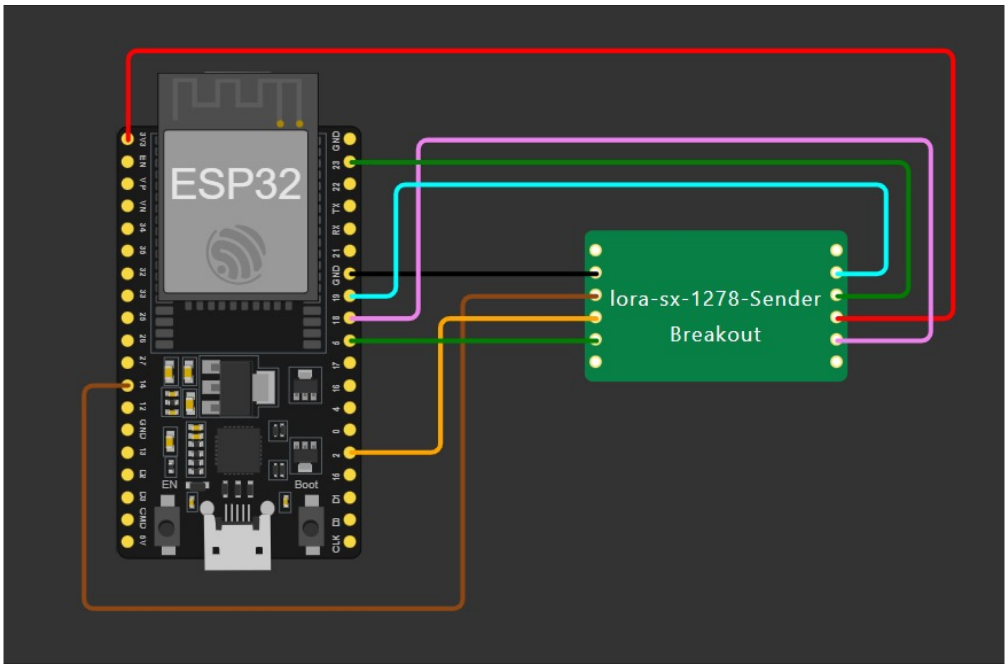
The transmitter end is responsible for collecting environmental data from the sensors connected to the ESP32-WROOM-32 microcontroller. This data includes measurements from rain, moisture, and temperature sensors. The data is then processed and transmitted via the LoRa module to the receiver end. Additionally, it uploads the collected sensor data to the ThingSpeak cloud for remote monitoring.

- Sensor data collection (rain, moisture, temperature)
- Data transmission using LoRaWAN
- Cloud integration with ThingSpeak

Below is the Wokwi simulation for the transmitter end:

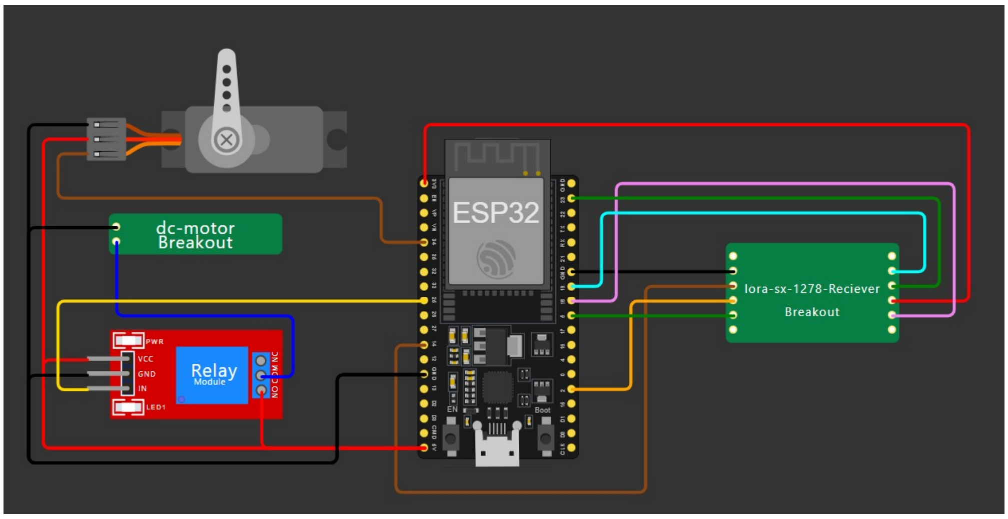
4.3 Receiver End

The receiver end retrieves data sent via LoRa from the transmitter. It also fetches data from the ThingSpeak cloud, ensuring redundancy and allowing real-time access to environmental data. Based on predefined conditions, the receiver controls actuators like relays or pumps for automated responses.



- Data reception via LoRa
- Cloud data fetching from ThingSpeak
- Actuator control based on predefined logic

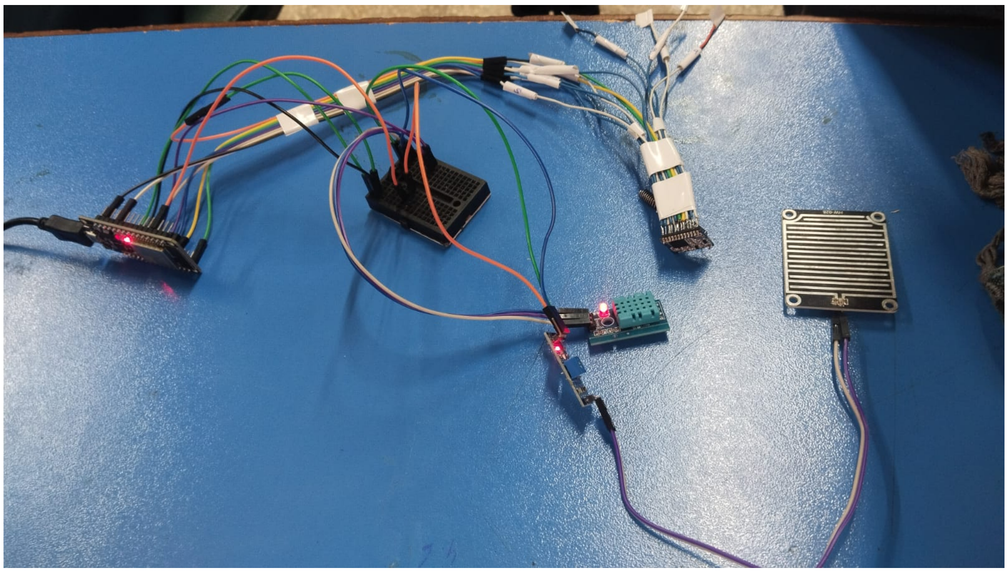
Below is the Wokwi simulation for the receiver end:



5 Hardware Implementation

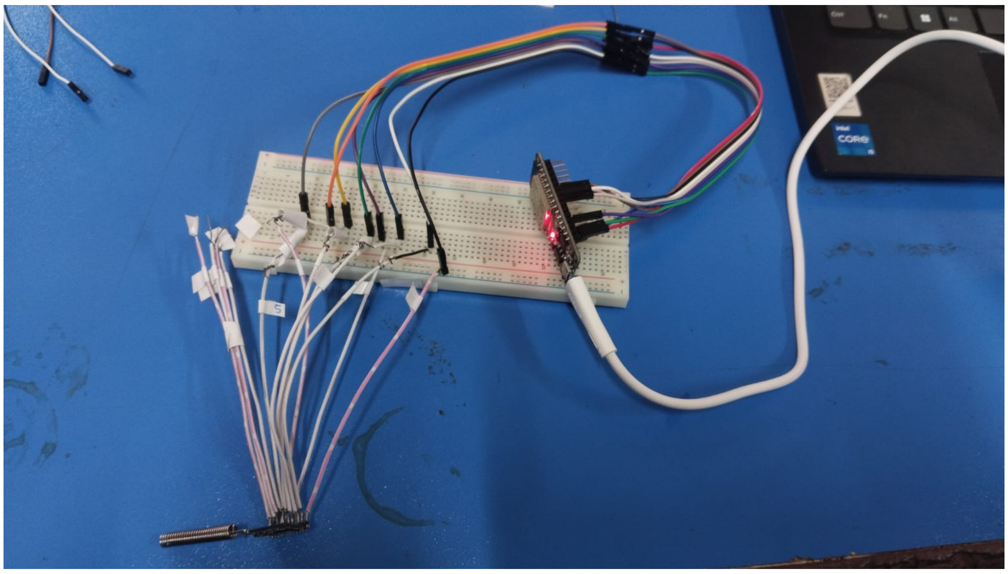
5.1 Transmitter End

The transmitter end consists of the ESP32-WROOM-32 microcontroller, which collects data from rain, moisture, and temperature sensors. The data is then transmitted to the receiver end via a LoRa module. The transmitter setup is compact and energy-efficient, designed for long-range communication.The sender esp32 uploads data to Thingspeak cloud as well.



5.2 Receiver End

The receiver end includes an ESP32-WROOM-32 microcontroller connected to a LoRa module, enabling it to receive data from the transmitter. Additionally, it retrieves data from the ThingSpeak cloud and controls actuators like relays based on predefined conditions. The setup ensures reliable operation and real-time monitoring.



6 Program Code

6.1 Transmitter End

6.1.1 ThingsSpeak

The code for the sender side implements an IoT-based weather monitoring system using an ESP32 microcontroller and the ThingSpeak cloud platform for real-time data logging and visualization. It collects data from a DHT11 sensor (temperature and humidity) and a raindrop sensor (rain level). The data is uploaded to ThingSpeak every 20 seconds for remote monitoring and analysis.

The system uses the following formulas to fetch sensor data:

- **Temperature:** `temperature = dht.readTemperature();`
- **Humidity:** `humidity = dht.readHumidity();`
- **Rain Detection:** `rainValue = analogRead(RAINDROP_PIN);`

Key Features:

- **Temperature and Humidity:** Readings are captured using the DHT11 sensor.

- **Rain Detection:** Analog data from the raindrop sensor detects moisture levels.
- **Wi-Fi Connectivity:** Data is sent over Wi-Fi to ThingSpeak.
- **Cloud Storage:** Sensor values are uploaded to Fields 1, 2, and 3 of the ThingSpeak channel.

The collected data is visualized on ThingSpeak for trend analysis and weather predictions. Additionally, it supports real-time updates with error handling for failed uploads, ensuring reliability in data transmission.

6.1.2 LoRa Code

Code for Transmitter End: This code demonstrates the integration of a DHT11 sensor, a raindrop sensor, and a LoRa communication module for environmental monitoring and data transmission.

- **Libraries Used:**

```
#include <SPI.h>
#include <LoRa.h>
#include <DHT.h>
```

These libraries are included to enable SPI communication, LoRa functionality, and DHT11 sensor operations. The LoRa module and sensors are assigned specific GPIO pins.

- **Sensor and LoRa Initialization:**

```
DHT dht(DHTPIN, DHTTYPE);
LoRa.setPins(ss, rst, dio0);
LoRa.begin(433E6);
```

The DHT11 sensor is initialized with its pin and type, and the LoRa module is set to operate at 433 MHz frequency with defined pins. A sync word (0xF3) is configured to prevent interference.

- **Sensor Readings:**

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
int rainValue = analogRead(RAINDROP_PIN);
```

The DHT11 sensor provides temperature and humidity, while the raindrop sensor gives an analog value to detect water levels. The readings are checked for errors to ensure valid data.

- **LoRa Transmission:**

```
LoRa.beginPacket();
LoRa.print("Temperature: ");
LoRa.print(temperature);
LoRa.print("C, Humidity: ");
LoRa.print(humidity);
LoRa.print("%, Rain Value: ");
LoRa.print(rainValue);
LoRa.endPacket();
```

Data is formatted into a readable string and sent via LoRa packets to the receiver.

- **Delay for Periodic Transmission:**

```
delay(5000);
```

The loop is programmed to transmit data every 5 seconds, ensuring regular updates for monitoring.

6.2 Receiver End

6.2.1 ThingsSpeak

The code for the receiver side implements an IoT-based data retrieval system using an ESP32 microcontroller to fetch and display weather data stored on ThingSpeak. It collects temperature, humidity, and rain sensor values uploaded from remote sensors, enabling real-time monitoring and analysis.

The data is fetched using the following formulas:

- **Humidity:** `humidity = ThingSpeak.readFloatField(channelID, 1, readAPIKey);`
- **Temperature:** `temperature = ThingSpeak.readFloatField(channelID, 2, readAPIKey);`
- **Rain Detection:** `rainValue = ThingSpeak.readFloatField(channelID, 3, readAPIKey);`

Key Features:

- **Wi-Fi Connectivity:** Connects to a local Wi-Fi network for internet access.
- **Cloud-Based Retrieval:** Fetches sensor values stored in ThingSpeak Fields 1, 2, and 3.
- **Real-Time Updates:** Reads data every 20 seconds for continuous monitoring.
- **Validation and Error Handling:** Checks for invalid readings and notifies errors.

The retrieved data can be visualized on a serial monitor for quick debugging or used for further processing, such as weather prediction algorithms or mobile app integrations.

6.2.2 LoRa Code

Code for Receiver End: This code configures a LoRa receiver to process environmental data transmitted by a LoRa transmitter. It receives and parses data and displays the results along with the RSSI (Received Signal Strength Indicator).

- **Libraries Used:**

```
#include <SPI.h>
#include <LoRa.h>
```

The SPI and LoRa libraries are included to support LoRa communication.

- **LoRa Initialization:**

```
LoRa.setPins(ss, rst, dio0);
while (!LoRa.begin(433E6)) {
    Serial.println("LoRa initialization failed. Retrying...");
    delay(500);
}
LoRa.setSyncWord(0xF3);
Serial.println("LoRa Initialized Successfully!");
```

The LoRa module is configured to operate at 433 MHz with a sync word (0xF3), ensuring communication only with devices using the same sync word. It retries initialization until successful.

- **Packet Reception and Parsing:**

```
int packetSize = LoRa.parsePacket();
if (packetSize) {
    char receivedData[256];
    int index = 0;
    while (LoRa.available() && index < 255) {
        receivedData[index++] = (char)LoRa.read();
    }
    receivedData[index] = '\0';
```

The receiver continuously listens for incoming LoRa packets. When data is received, it is stored in a character array and null-terminated to form a proper string for further processing.

- **Data Extraction and Validation:**

```
float temperature, humidity;
int raindrop;
sscanf(receivedData, "Temperature: %f C, Humidity: %f %%, Rain Value: %d
Serial.printf("Temp: %.1f C, Hum: %.1f%%, Rain: %d\n", temperature, humi
```

The received data is parsed into temperature, humidity, and rain level values using the sscanf() function. If parsing succeeds, the values are displayed; otherwise, an error message is shown.

- **Signal Strength Monitoring:**

```
int rssi = LoRa.packetRssi();
Serial.print("Signal␣Strength␣(RSSI):␣");
Serial.println(rssi);
```

The RSSI value is displayed to evaluate signal strength, providing insights into link quality.

- **Delay for Readability:**

```
delay(5000);
```

A 5-second delay is added to make the output more readable.

6.3 Weather Detection Algorithm

7 Troubleshooting

During the development of our weather station project, we encountered synchronization issues between the LoRa sender and receiver modules, leading to mismatched data timestamps and occasional packet loss. We resolved these problems by carefully adjusting the timing intervals for data transmission and reception, ensuring proper synchronization. Additionally, the rain drop sensor required thorough calibration as initial readings were inconsistent, which we addressed by fine-tuning the sensor sensitivity and verifying calibration against known water levels. Delays in actuation may occur due to processing overhead, necessitating optimization of data processing algorithms. These steps ensured accurate data collection and reliable communication between the sender and receiver, enabling smooth integration with the cloud for real-time monitoring and control.

8 Conclusions

The Weather Station with ThingSpeak Cloud Integration project demonstrates the potential of combining IoT, wireless communication, and cloud technologies to create a scalable and efficient environmental monitoring system. By leveraging the ESP32-WROOM-32 microcontroller and LoRa communication module, the project effectively collects, transmits, and processes environmental data, ensuring real-time monitoring and automated control of actuators. The integration of cloud storage with ThingSpeak enhances accessibility and scalability, making the system versatile for various applications, such as smart agriculture, environmental monitoring, and IoT-based automation.

This project highlights the benefits of utilizing edge computing to optimize actuator responses, reducing manual intervention, and promoting resource efficiency, particularly in water management. By enabling automated processes and supporting sustainable practices, the system contributes to global sustainability goals, such as water conservation and improved agricultural productivity.

9 Future Work

To predict changes in the environment and maximize actuator responses, the system can be extended in further work by including machine learning models and advanced analytics. For smart home or agricultural applications, additional sensors that measure things like light intensity and soil moisture might improve system performance. For sustainable operation, the project can also investigate energy-efficient designs, including solar-powered modules. Lastly, to cover wider areas, the system might integrate multi-node LoRa networks, allowing for scalable deployments of IoT solutions for urban, agricultural, or industrial settings.

10 References

11 Appendices