National Institute of Technology Rourkela

PROJECT NAME: IoT Climate Monitor

Guided by : Prof. Santos Kumar Das

Team Members:

Satyajit Barik	122EC0123
Yash Raj Dash	122EC0126
Dibyaranjan Dehury	122EC0128

Temperature, Pressure and Humidity Measurement with Arduino and Adafruit ESP8266 and displaying it on 128*64 OLED screen

Objective:

Create a comprehensive monitoring system using Arduino for collecting and transmitting data on temperature, pressure and humidity. Display the real-time results on an Adafruit IO. This project provides valuable insights into environmental conditions and location tracking.

Hardware Requirements:

ESP8266

Pressure Sensor: (BMP280 or BMP180)

Humidity and Temperature Sensor: (DHT22 or DHT11)

OLED Display(128*64)

Power Supply

Brief Description:

ESP8266:-

The ESP8266 is a versatile and low-cost Wi-Fi module that has gained widespread popularity in the field of Internet of Things (IoT) and embedded systems. It is designed to provide wireless connectivity to microcontrollers and other embedded devices. Here's a brief overview of the ESP8266 and its connection capabilities:

Microcontroller and Wi-Fi Module:

The ESP8266 integrates a microcontroller unit (MCU) along with a Wi-Fi module on a single chip. This MCU allows you to program and control the ESP8266, and the Wi-Fi module enables wireless communication.

Wireless Connectivity:

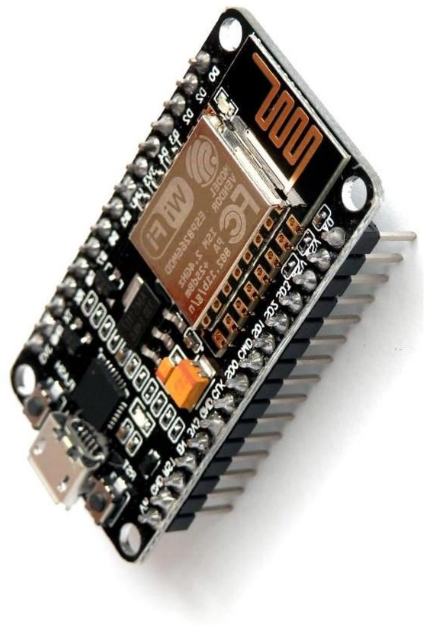
The primary feature of the ESP8266 is its ability to connect to Wi-Fi networks, allowing your embedded devices to communicate over the internet or local networks.

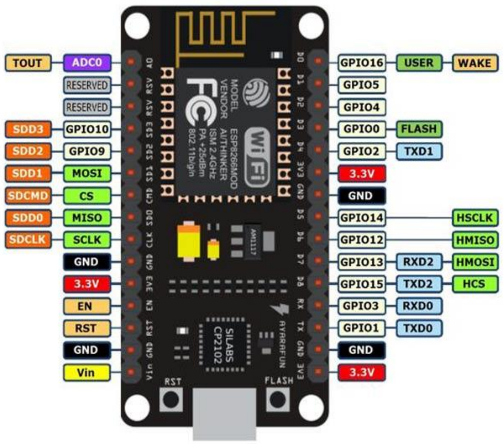
Programming:

The ESP8266 can be programmed using the Arduino IDE, which makes it user-friendly and accessible to a wide range of developers. There are also other programming options, including using the native Espressif SDK or MicroPython.

GPIO Pins:

The ESP8266 module typically has a set of General Purpose Input/Output (GPIO) pins that can be used to interface with sensors, actuators, and other electronic components.





BMP180:-

BMP180 used in weather stations, altitude measurement, and other applications where accurate pressure data is required. While they have some differences in terms of features and specifications, their basic connection principles are quite similar.A general guide on connecting a pressure sensor (BMP280 or BMP180) to a microcontroller like the ESP8266:

1. Power Supply:

Connect the VCC pin of the BMP280/BMP180 to a 3.3V power source. Both sensors operate at low voltage and are compatible with the typical 3.3V used in microcontroller systems.

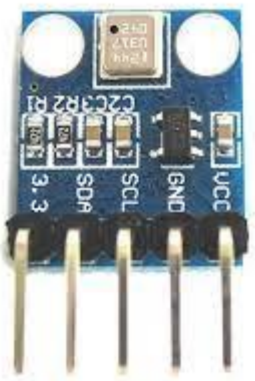
2. Ground Connection:

Connect the GND (ground) pin of the BMP280/BMP180 to the ground (GND) of your microcontroller.

3. I2C Communication:

Both BMP280 and BMP180 support I2C communication. Connect the SDA (data line) and SCL (clock line) pins of the sensor to the corresponding SDA and SCL pins on your microcontroller.

On the ESP8266, the default I2C pins are often labeled as D1 (GPIO5) for SDA and D2 (GPIO4) for SCL



Humidity and Temperature Sensor: (DHT22 or DHT11)

The DHT11 is a low-cost digital sensor that is commonly used to measure temperature and humidity in various electronic projects. It comes with a built-in thermistor and a humidity sensor. The sensor provides accurate readings and is easy to interface with microcontrollers like Arduino or ESP8266.

Key Features:

Humidity Range: 20% to 90% RH (Relative Humidity)

Temperature Range: 0°C to 50°C

Accuracy:

Humidity: ±5% RH

Temperature: ±2°C

Operating Voltage: 3.3V to 5V

Communication: Digital signal output with a single-wire serial interface.

Connection to ESP8266:

In the code you provided, the DHT11 sensor is connected to the ESP8266 microcontroller. Here's a general guide on how you might connect the DHT11 to an ESP8266 board (e.g., NodeMCU):

DHT11 Pinout:

Pin 1 (leftmost): VCC (3.3V to 5V)

Pin 2: Data (Signal)

Pin 3: Not Connected

Pin 4 (rightmost): GND

ESP8266 Pin Connection:

Connect the VCC pin of DHT11 to a 3.3V pin on the ESP8266 board.

Connect the GND pin of DHT11 to a GND pin on the ESP8266 board.

Connect the Data pin of DHT11 to a digital pin on the ESP8266 (e.g., D5 in your code).

OLED Display(128 * 64):-

An OLED (Organic Light-Emitting Diode) display with a resolution of 128x64 pixels is a type of display technology that utilizes organic compounds to emit light when an electric current is applied. OLED displays offer several advantages over traditional LCD (Liquid Crystal Display) technology, including better contrast, faster response times, wider viewing angles, and the ability to produce deeper blacks.

Here's a brief description of the OLED display in your project and how it's connected:

OLED Display (128x64):

Size and Resolution:

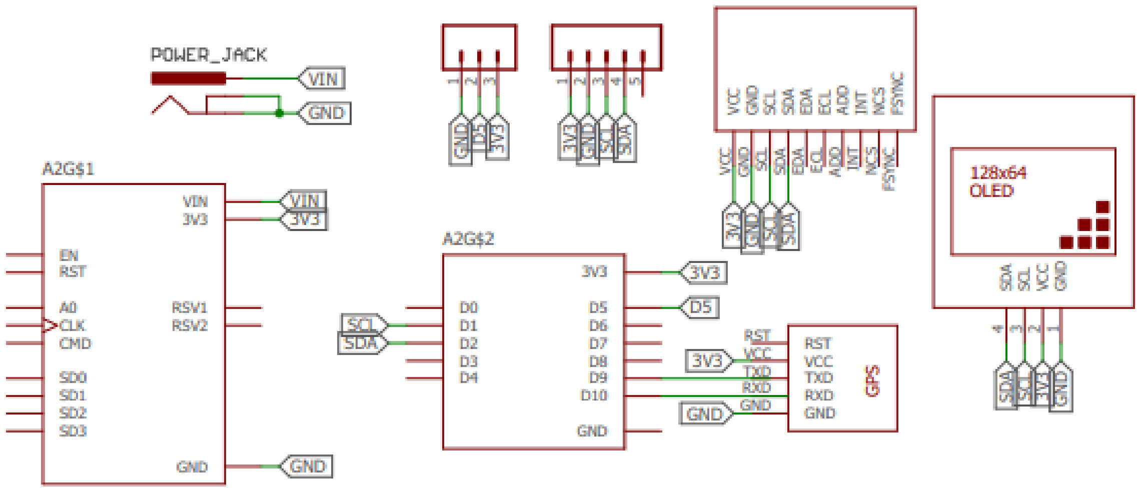
The OLED display in your project has a resolution of 128x64 pixels. This means it can display 128 pixels horizontally and 64 pixels vertically.

Communication Interface:

OLED displays commonly use the I2C (Inter-Integrated Circuit) communication protocol. In your code, you are using the Wire library to communicate with the display, indicating an I2C connection.

1.3 Inch 128×64 OLED Display

Circuit Diagram-:



Software Requirements:

Arduino IDE: https://www.arduino.cc/en/software

Adafruit IO Account: https://io.adafruit.com/

Adafruit MQTT Library: Install from the Arduino Library Manager

Libraries for Sensors: Install libraries for BMP280/BMP180, DHT22/DHT11, SimpleDHT.h

ESP8266WiFi.h,Adafruit_MQTT.h and Adafruit_MQTT_Client.h,Adafruit_SSD1306

Create an Adafruit IO Account:

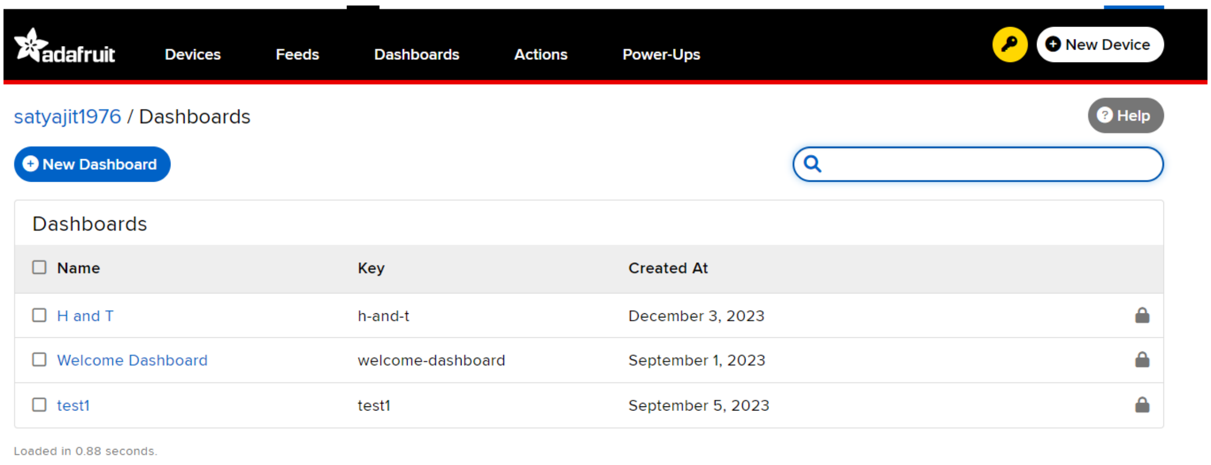
If you don't have an account, go to Adafruit IO, sign up, and log in.

Create a Dashboard:

Once logged in, click on "Dashboards" in the left menu.

Click on the "Actions" button (three dots) and select "Create Dashboard."

Give your dashboard a meaningful name (e.g., Environmental Monitoring) and click "Create."



satyajit1976 / Dashboards

Dashboards

Name	Key	Created At
H and T	h-and-t	December 3, 2023
Welcome Dashboard	welcome-dashboard	September 1, 2023
test1	test1	September 5, 2023

Loaded in 0.88 seconds.

Add Feeds:

Click on the "Feeds" tab in the left menu.

Create four feeds for temperature, pressure, humidity, and location:

Click "Create a New Feed."

Enter a unique name for the feed (e.g. Temperature).

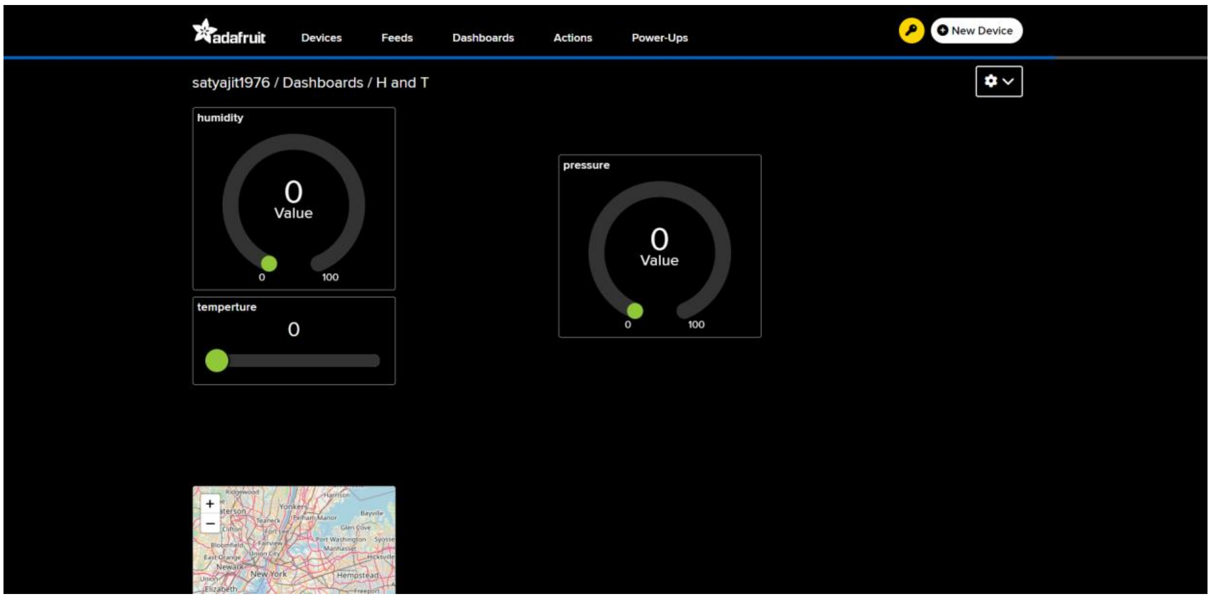
Repeat this process for pressure, humidity, and location.

Build Blocks on the Dashboard:

Go back to your dashboard.

Click on the "Blocks" tab in the left menu.

Click the "+" button to add a new block.



Temperature Block:

Choose the "Line Chart" block type.

Select the temperature feed.

Configure other settings like the title, colour, and data range.

Click "Create Block."

Pressure Block:

Follow the same steps as for the temperature block but select the pressure feed.

Humidity Block:

Follow the same steps as for the temperature block but select the humidity feed.

Location Block:

Choose the "Map" block type.

Select the location feed.

Configure other settings like the title.

Click "Create Block."

Arrange Blocks:

Click and drag the blocks to arrange them on your dashboard.

Resize or customize the appearance of each block as needed.

Save and View Dashboard:

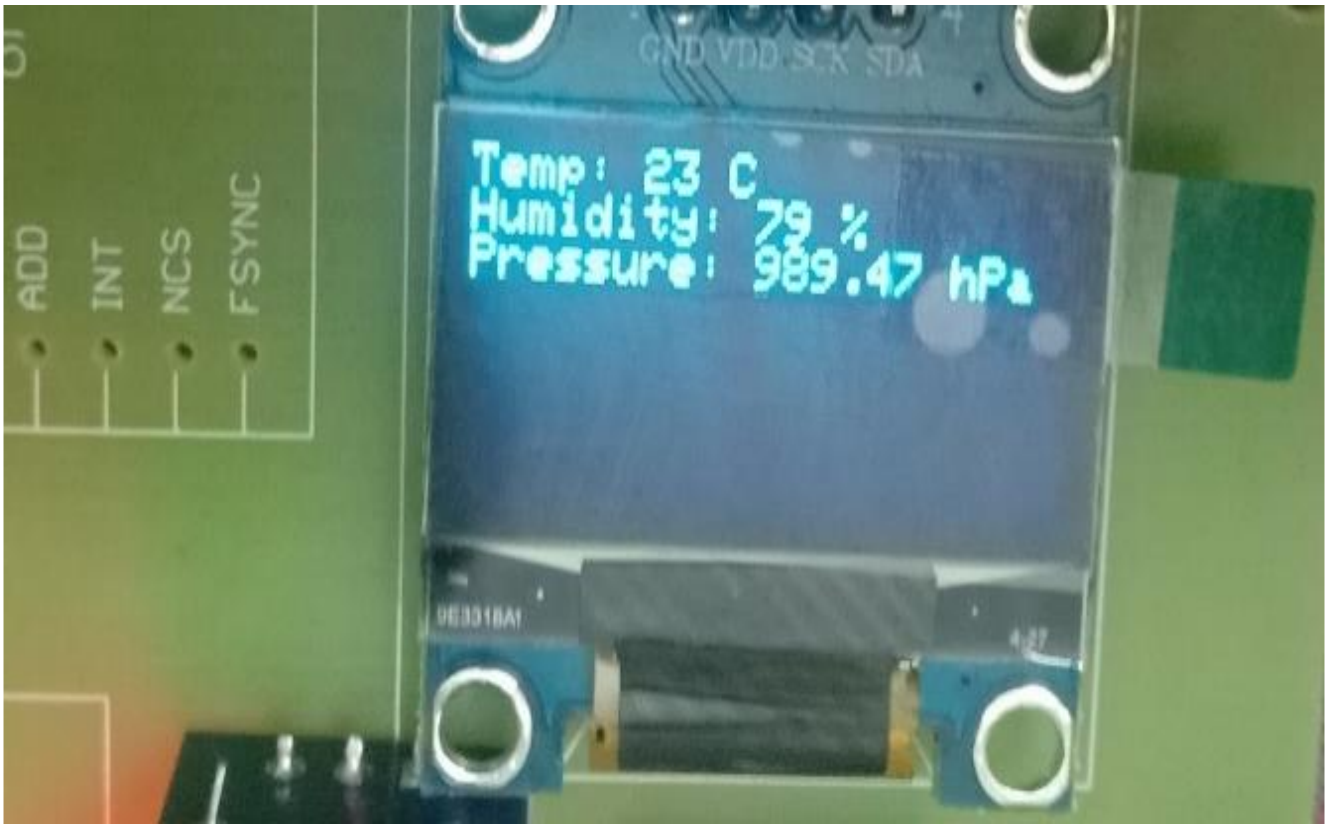
Click "Save" to save your dashboard.

Click "View Dashboard" to see the real-time data visualizations.

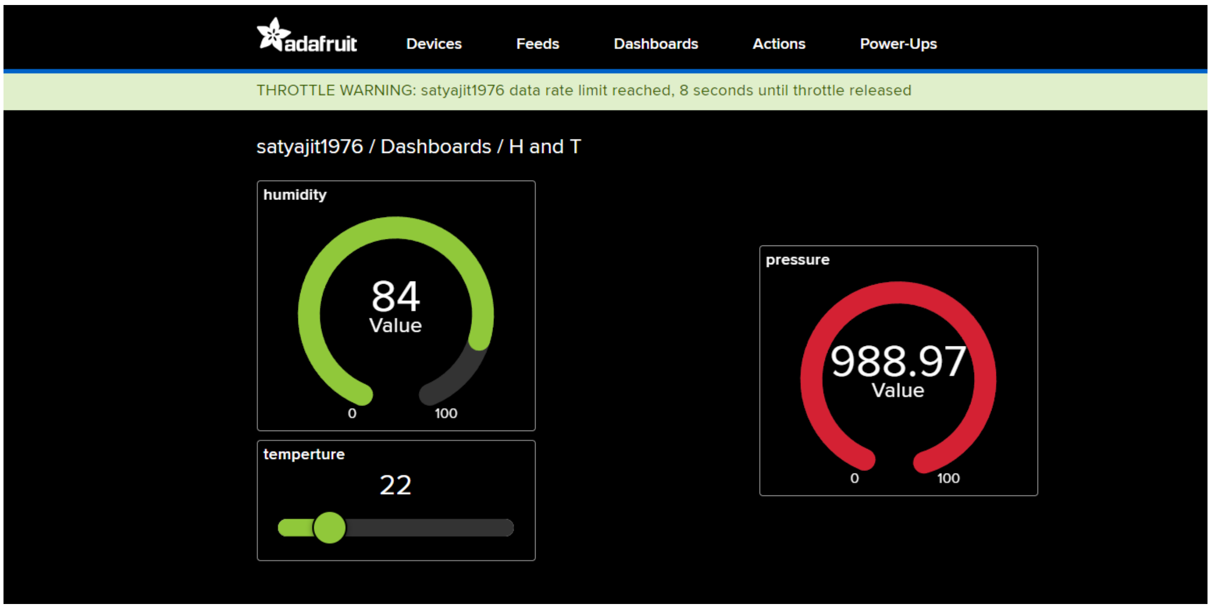
Code for all the sensors-:

```
#include <SimpleDHT.h>
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085.h>
#include <Adafruit_SSD1306.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

#define WLAN_SSID       "V2065"
#define WLAN_PASS       "satya2005"
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "satyajit1976"
#define AIO_KEY         "aio_UwBx987sQIfSiUSkdJ4tD2yDrV6V"

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME,
AIO_KEY);
Adafruit_MQTT_Publish temperature = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/temperature");
Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/humidity");
Adafruit_BMP085 bmp;

int pinDHT11 = D5;
SimpleDHT11 dht11;

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  Serial.begin(115200);
  Wire.begin();
  bmp.begin();
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) { delay(500); }
  connect();
}

void connect() {
  while (mqtt.connect() != 0) { delay(10000); }
}

void loop() {
  if (!mqtt.ping(3)) {
    if (!mqtt.connected()) connect();
  }

  byte temperature_data = 0;
  byte humidity_data = 0;
  float pressure_data = bmp.readPressure() / 100.0F; // convert to hPa

  if (dht11.read(pinDHT11, &temperature_data, &humidity_data, NULL) == SimpleDHTErrSuccess) {
    temperature.publish((int)temperature_data);
    humidity.publish((int)humidity_data);
    // Publish atmospheric pressure
    Adafruit_MQTT_Publish pressure = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/pressure");
    pressure.publish(pressure_data);

    // Display on OLED screen
    // Display on OLED screen
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0, 0);
display.print("Temp: ");
display.print((int)temperature_data);
display.println(" C");
display.print("Humidity: ");
display.print((int)humidity_data);
display.println(" %");
display.print("Pressure: ");
display.print(pressure_data);
display.println(" hPa");
display.display();

    delay(5000);
  }
}
```

Output in OLED Display:



Output in Adafruit:



THROTTLE WARNING: satyajit1976 data rate limit reached, 8 seconds until throttle released

satyajit1976 / Dashboards / H and T

humidity

84
Value

0 100

temperture

22

pressure

988.97
Value

0 100

Conclusion-:

The provided Arduino code constitutes a robust environmental monitoring project, leveraging an ESP8266 microcontroller, DHT11, BMP085 sensors, and an OLED display. The code initializes necessary libraries, connects to Wi-Fi, and establishes MQTT communication with Adafruit IO. The DHT11 captures temperature and humidity data, while the BMP085 records atmospheric pressure. These readings are then published to corresponding Adafruit IO feeds through MQTT, facilitating remote monitoring. The SSD1306-based OLED display enhances user interaction by presenting real-time information in a clear format.

The project excels in its integration of sensor data, wireless communication, and cloud-based storage. Adafruit_MQTT handles the connection to Adafruit IO, ensuring seamless data transmission. The OLED display provides a tangible interface, offering insights into temperature, humidity, and pressure. The code includes robust error-handling mechanisms, checking Wi-Fi connectivity and implementing MQTT reconnection strategies. This comprehensive solution serves as an effective foundation for environmental sensing and IoT applications. Opportunities for expansion and customization abound, allowing users to tailor the system to specific needs, making it an educational and versatile project for students exploring sensor integration, data communication, and display technologies.