Instrumentation & Measurements

IoT Based Weather Monitoring System w/ a Live Server

Submitted To:

Prof. Dr. Aamir Shareef

Submitted By:

Muhammad Haider Ali (2022-MC-45)

Department of Mechatronics & Control Engineering, University of Engineering & Technology, Lahore

May 12, 2025

I. Introduction:

The purpose of this project is to design and implement an **IoT-based Weather Monitoring System** capable of measuring and displaying key atmospheric parameters: temperature, altitude, humidity, pressure, and rainfall detection. The system is developed using three different types of sensors: **DHT11**, **BMP180**, and a **Rain Sensor**, interfaced with an **Arduino Mega 2560** microcontroller. Real-time readings are displayed via the Arduino Serial Monitor and data is sent to a cloud server through ThingSpeak API which is done using a NodeMCU ESP8266 WiFi Module.

This project fulfills the criteria of a **Complex Engineering Problem (CEP)** and **Complex Engineering Activity (CEA)** by integrating multiple sensors, performing signal conditioning, and ensuring system-level design and verification.

II. Objective:

- To measure environmental parameters (temperature, altitude, humidity, pressure, rainfall) using different sensors.
- To perform basic signal conditioning where applicable.
- To simulate, implement, and test a working weather monitoring system.

III. Components Used:

- Arduino Mega 2560.
- DHT11 Temperature and Humidity Sensor.
- BMP180 Pressure Sensor.
- Rain Sensor (Yl-83 Module).
- ESP8266 for real-time data communication.
- 16x2 Generic LCD for Data Display.
- PCF8574 I2C Serial Interface Adapter Module.
- Breadboard and jumper wires.
- USB cable for Arduino Programming.

IV. Sensor Details and Signal Conditioning”

1. DHT11 – Temperature & Humidity Sensor

- **Output**: Digital signal
- **Signal Conditioning**: No external conditioning required. The sensor outputs a calibrated digital signal. Data is read using a DHT library and displayed directly.
- **Notes**: Can be improved with averaging filters in software to smooth out jittery readings.

2. BMP180 – Pressure and Altitude Sensor

- **Output**: I2C Digital Interface
- **Signal Conditioning**: No hardware signal conditioning required. Calibration is built-in. The software performs compensation and unit conversion.
- **Notes**: Use floating-point scaling in code for proper unit conversion (e.g., Pa to hPa or meters to feet).

3. Yl-83 Rain Sensor

- **Output**: Analog signal (via AO pin) or Digital signal (via DO pin through comparator)
- **Signal Conditioning**:
    - **Amplification**: Not needed, signal strength is sufficient for Arduino's analog input.
    - **Filtration**: A basic RC low-pass filter can be added to reduce noise if necessary.
    - **Scaling**: Analog reading scaled in code to reflect rain intensity.
    - **Conversion**: Analog signal converted to digital using ADC of Arduino Mega.

V. Implementation and Working:

The system is constructed on a breadboard and interfaced to an Arduino Mega 2560. Sensor readings are acquired periodically and displayed on a 16x2 LCD screen using I2C communication for reduced wiring. Simultaneously, sensor data is printed to the Arduino Serial Monitor for real-time debugging and verification.

VI. Real-Time Communication Using ESP8266 and ThingSpeak:

The ESP8266 module is programmed to establish a WiFi connection and upload sensor data to the ThingSpeak API. The Arduino Mega communicates with ESP8266 via serial commands (AT commands or SoftwareSerial). Data is formatted in HTTP GET requests and sent to ThingSpeak every 15 seconds.

Implementation Steps:

1. ESP8266 is powered through 3.3V and connected via SoftwareSerial to Arduino Mega.

2. WiFi credentials and ThingSpeak API key are defined in the code.
3. Arduino sends collected sensor data to ESP8266.
4. ESP8266 formats and sends HTTP GET requests to ThingSpeak server.

Note: The code responsible for data collection, processing, and transmission is provided in the Appendix.

VII. Proteus Simulation:

The entire system was simulated using Proteus before hardware implementation. Sensors were simulated using virtual models or potentiometers for variable analog readings. The ESP8266 and LCD were tested with serial terminals and simulated I2C protocol.

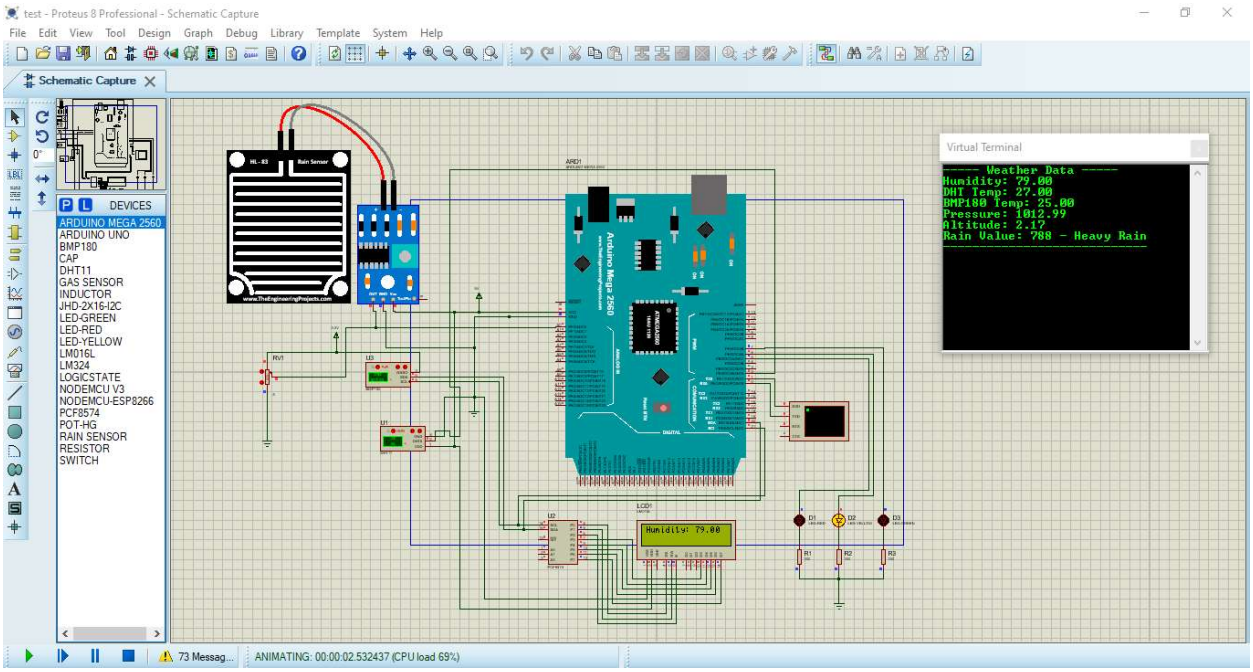

Fig1.1: IoT Based Weather Monitoring Proteus Simulation

VIII. Hardware Implementation:

The final hardware implementation is based on the following wiring:

- **DHT11**: Connected to Digital Pin 2
- **BMP180**: Connected via I2C (A4 - SDA, A5 - SCL)
- **Rain Sensor**: AO to A0 (Analog Input)
- **LCD**: Connected via I2C to SDA/SCL

- **ESP8266**: TX/RX via SoftwareSerial (Pin 10 & 11)

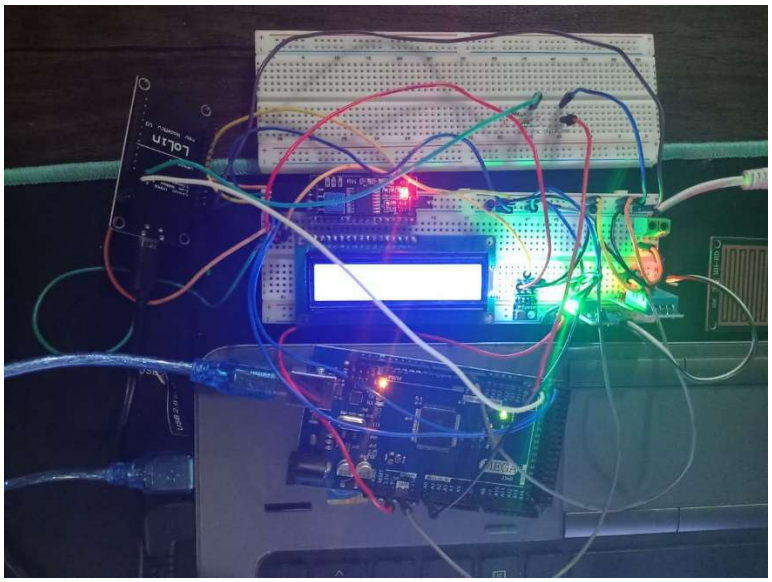

Fig1.2: IoT Based Weather Monitoring System Hardware

IX. Serial Monitor Results:

Data output from the Arduino Serial Monitor shows real-time readings for temperature, humidity, pressure, altitude, and rain sensor status.

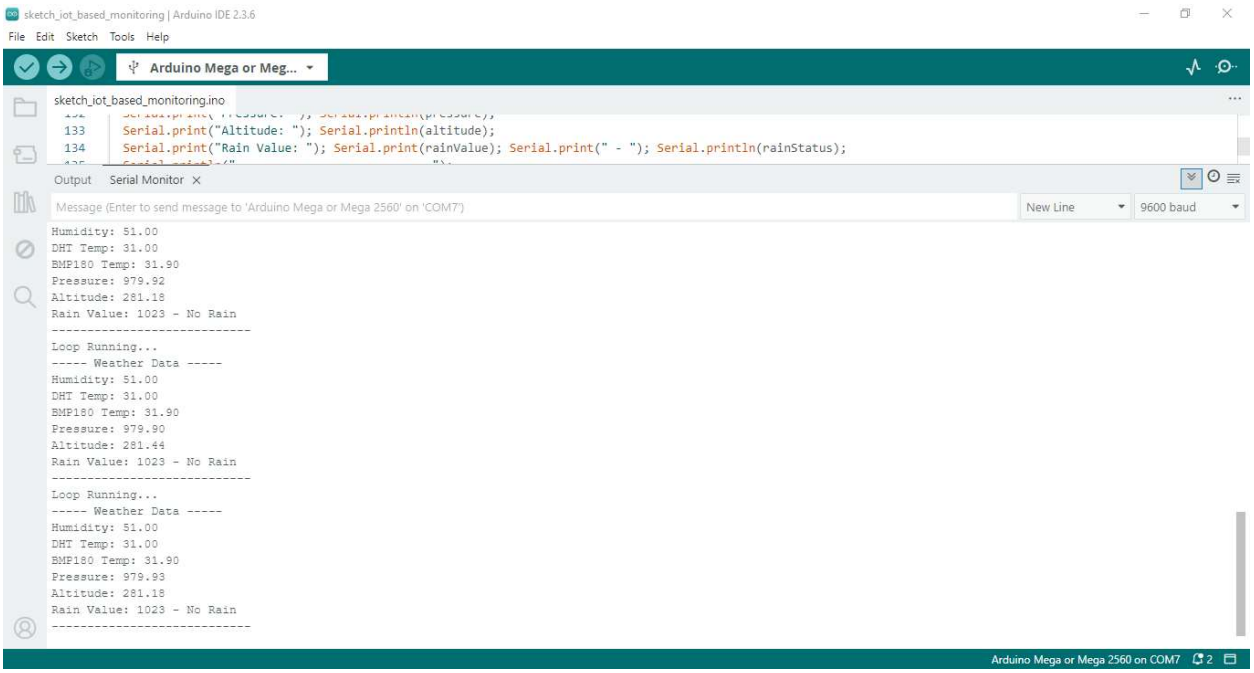

Fig1.3: Arduino IDE Serial Monitor Results

X. ThingSpeak IoT Cloud Graphs:

The data collected from the weather monitoring system is uploaded in real-time to the ThingSpeak cloud platform using the ESP8266 module. Each sensor value is assigned to a specific field on ThingSpeak for organized data visualization:

- **Field 1:** Humidity (from DHT11)
- **Field 2:** Temperature (from DHT11)
- **Field 3:** Temperature (from BMP180)
- **Field 4:** Pressure (from BMP180)
- **Field 5:** Altitude (from BMP180)
- **Field 6:** Rain Detection (from YL-83 sensor)

Graphs are automatically generated by ThingSpeak to visualize trends over time. These visualizations provide insights into changing weather conditions and can support applications like environmental monitoring, irrigation control, and early warning systems.

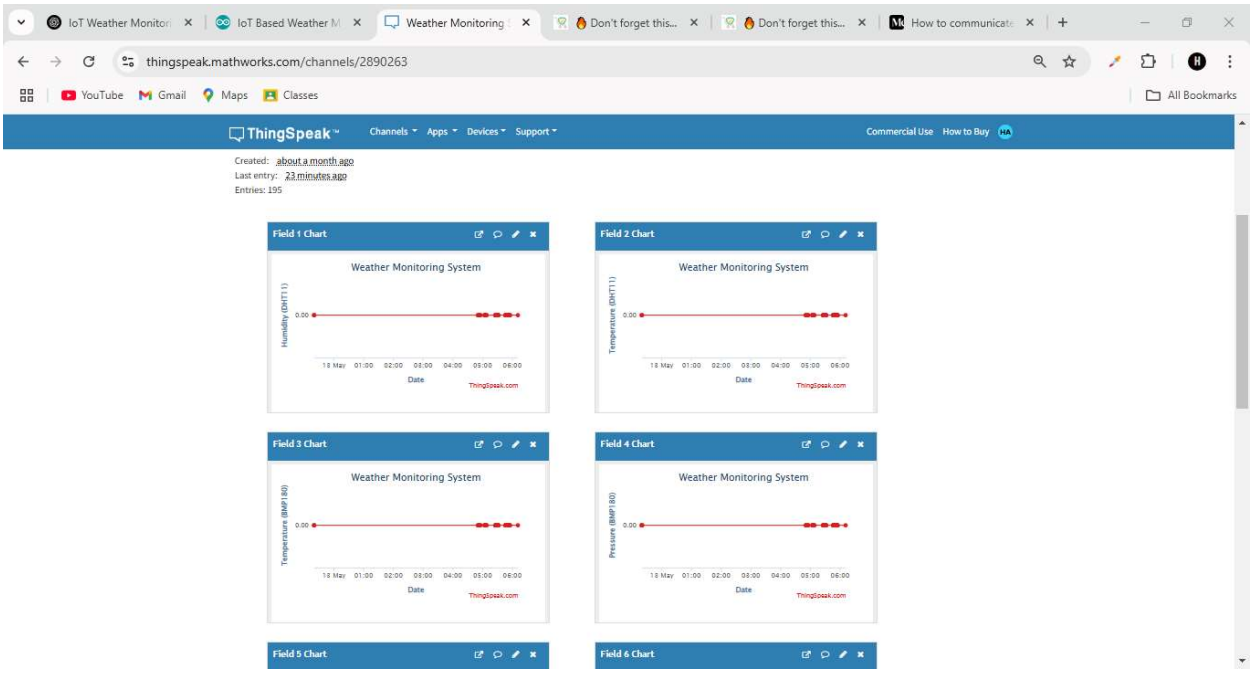

Fig1.4: ThingSpeak API Graphs for Weather Monitoring System

A node.js server is also built to display data offline. Following is the server which is open at port 3000 using localhost.

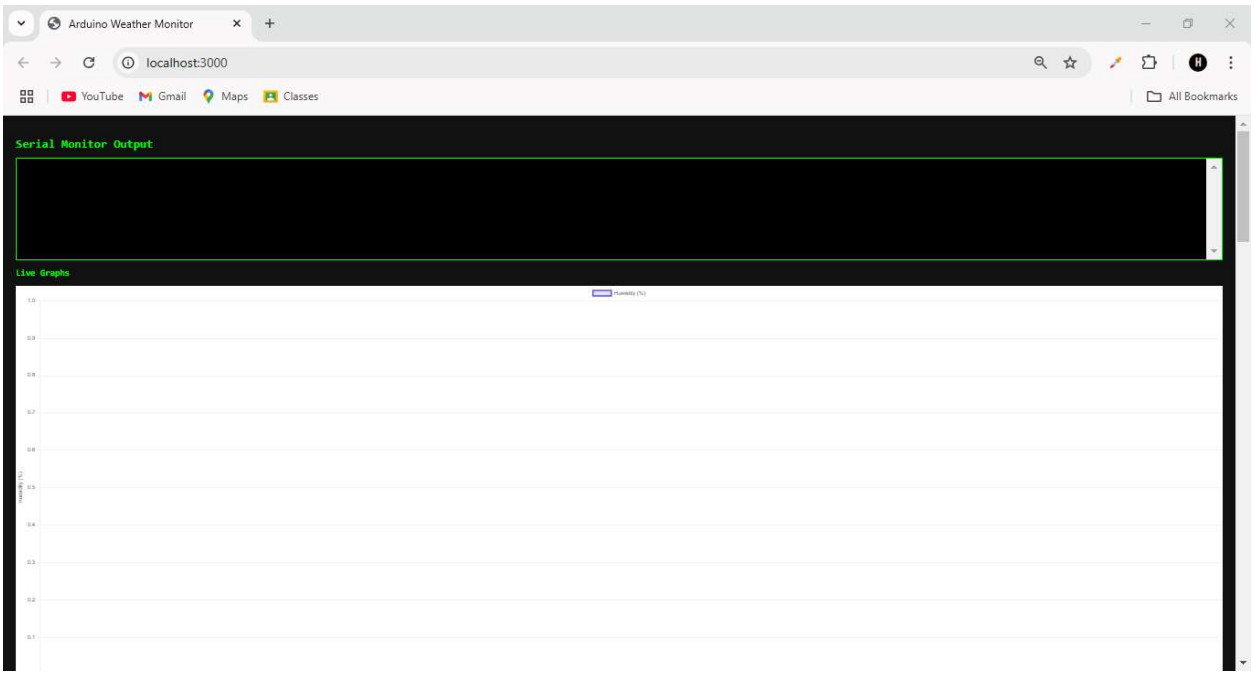

Fig1.5: Node.js Offline Web Server

XI. Conclusion:

This project demonstrates an effective and scalable weather monitoring system integrating multiple sensors and real-time cloud communication. It can be extended for real-time alerts, remote automation, and smart city infrastructure. The system emphasizes sensor integration, real-time IoT communication, and basic signal conditioning—making it an excellent representation of a Complex Engineering Problem.

BOM (Bill of Materials):

Sr.	NAME	Description	Quantity	Price per piece	Total price	Contact info of vendor

01	Arduino Mega 2560	Arduino Mega 2560 R3 In Pakistan SKU: B187, KRT15, TMD50, TH25, A	1	3,700	3,700	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
02	DHT11	DHT11 Temperature Humidity Sensor Module for Arduino With LED Light Indication	1	230	230	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
03	HW-596 / GY-68	HW-596 / GY-68 Arduino BMP180 Barometric Pressure Sensor Module in Pakistan	1	140	140	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221

04	YL-83 Rain Sensor Module	YL-83 Rain Drops Detection Sensor Module in Pakistan	1	150	150	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
05	Generic 16x2 LCD	Blue 16x2 LCD for Arduino Display	1	289	280	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
06	PCF8574	PCF8574 I2C Serial Interface Adapter Module	1	176	175	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
07	NodeMCU ESP8266 CH340 LoLin V3 WiFi Development Board	NodeMCU ESP8266 CH340 LoLin V3 WiFi Development Board for Serial Communication b/w Arduino and ESP8266	1	600	600	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
08	Power Supply	MB102 Breadboard Power Supply Module 3.3V/5V for	1	150	150	https://digilog.pk/

	Module MB102	Arduno Solderless Breadboard				**Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
09	MB102 Solderless Prototyping Breadboard	MB102 Solderless 830 Points Prototyping Breadboard High Quality	2	270	540	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
10	Jumper Wires (Male-Male)	65pcs Jumper Wire Cable Male-Male for Arduino 1Bag	1	220	220	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
11	20cm Micro USB Cable	20cm Micro USB Cable for Programming NodeMCU ESP32	1	40	40	https://digilog.pk/ **Address:** 13Th Regal chowk, Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221
12	2.54mm Pitch 40	2.54mm Pitch 40 Pin Male Header in Pakistan	5	12	60	https://digilog.pk/ **Address:** 13Th Regal chowk,

	Pin Male Header					Shahrah-e-Quaid-e-Azam, Mozang Chungi, Lahore, +92 312 4002221+92 323 4373141
13					**Total =** **Rs.** 6785 /- (Including Delivery Fees)	

APPENDIX A

Arduino Mega 2560 Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <dht.h>
#include <Adafruit_BMP085.h>

#define RED_LED_PIN 5      // Red LED pin (Failure)
#define YELLOW_LED_PIN 6   // Yellow LED pin (Caution)
#define GREEN_LED_PIN 7    // Green LED pin (OK)

#define DHTPIN 2                      // DHT11 sensor pin
#define RAIN_SENSOR_PIN A0             // Rain sensor pin

LiquidCrystal_I2C lcd(0x20, 16, 2);   // Initialize I2C LCD
dht DHT;                              // Create DHT object
Adafruit_BMP085 bmp;                  // Create BMP180 object

float humidity;
float temperature;
float pressure;
float altitude;
int rainValue = 0;
String rainStatus = "No Rain";

// Define threshold values
float HUMIDITY_THRESHOLD = 85.0;    // Humidity threshold for warning
float TEMPERATURE_THRESHOLD = 35.0; // Temperature threshold for warning
float PRESSURE_THRESHOLD = 2980.0;   // Pressure threshold for warning (in hPa)
int RAIN_THRESHOLD = 2600;           // Rain sensor threshold for heavy rain

void setup() {
  Serial.begin(9600);   // Start Serial communication
  Serial.println("Serial Communication Initialized");
  lcd.begin(16, 2);     // Initialize LCD
  lcd.backlight();      // Turn ON LCD backlight

  lcd.setCursor(4, 0);
  lcd.print("Hello!");
  lcd.setCursor(0, 1);
  lcd.print(" Weather Monitor ");
  delay(1000);
  lcd.clear();

  // Initialize BMP180 sensor
  if (!bmp.begin()) {
    Serial.println("BMP180 sensor not found!");
```

```
    lcd.setCursor(0, 0);
    lcd.print("BMP180 Error!");
    while (1);
  }

    // Initialize LED pins
  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(YELLOW_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
}

void loop() {
  Serial.println("Loop Running...");
  delay(1000);
  // Read temperature and humidity from DHT11
  DHT.read11(DHTPIN);
  humidity = DHT.humidity;
  temperature = DHT.temperature;

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("DHT Sensor Error!");
    delay(1000);
    return;
  }

  // Read values from BMP180
  float bmpTemp = bmp.readTemperature();
  pressure = bmp.readPressure() / 100.0; // Convert to hPa
  altitude = bmp.readAltitude();

  // Read rain sensor
  // Read analog value (0-1023)
   // Read and average rain sensor
  int rainSum = 0;
  for (int i = 0; i < 10; i++) {
    rainSum += analogRead(RAIN_SENSOR_PIN);
    delay(5);
  }
  rainValue = rainSum / 10;

    // Determine rain status
  if (rainValue > 600) {
```

```
    rainStatus = "No Rain";
  } else if (rainValue > 300) {
    rainStatus = "Light Rain";
  } else {
    rainStatus = "Heavy Rain";
  }

    // Check if any sensor value exceeds the threshold
  if (humidity > HUMIDITY_THRESHOLD || temperature > TEMPERATURE_THRESHOLD ||
pressure > PRESSURE_THRESHOLD || rainValue > RAIN_THRESHOLD) {
    // System in warning state or failure
    digitalWrite(RED_LED_PIN, HIGH);   // Turn on Red LED (Failure)
    digitalWrite(YELLOW_LED_PIN, LOW); // Turn off Yellow LED
    digitalWrite(GREEN_LED_PIN, LOW);  // Turn off Green LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Warning! Check System");
  }
  else if (rainValue > 300 || humidity > 60.0 || temperature > 30.0) {
    // System in caution state
    digitalWrite(RED_LED_PIN, LOW);    // Turn off Red LED
    digitalWrite(YELLOW_LED_PIN, HIGH); // Turn on Yellow LED (Caution)
    digitalWrite(GREEN_LED_PIN, LOW);  // Turn off Green LED
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Caution! Abnormal");
  } else {
    // System is OK
    digitalWrite(RED_LED_PIN, LOW);    // Turn off Red LED
    digitalWrite(YELLOW_LED_PIN, LOW); // Turn off Yellow LED
    digitalWrite(GREEN_LED_PIN, HIGH); // Turn on Green LED (OK)
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("System OK");
  }


  // Print values to Serial Monitor
  Serial.println("----- Weather Data -----");
  Serial.print("Humidity: "); Serial.println(humidity);
  Serial.print("DHT Temp: "); Serial.println(temperature);
  Serial.print("BMP180 Temp: "); Serial.println(bmpTemp);
  Serial.print("Pressure: "); Serial.println(pressure);
  Serial.print("Altitude: "); Serial.println(altitude);
```

```
  Serial.print("Rain Value: "); Serial.print(rainValue); Serial.print(" - ");
Serial.println(rainStatus);
  Serial.println("----------------------------");

  // Display on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Humidity: ");
  lcd.print(humidity);
  lcd.print(" %");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("DHT Temp: ");
  lcd.print(temperature);
  lcd.print((char)223);
  lcd.print("C");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("BMP Temp: ");
  lcd.print(bmpTemp);
  lcd.print((char)223);
  lcd.print("C");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Pressure: ");
  lcd.print(pressure);
  lcd.print(" hPa");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Altitude: ");
  lcd.print(altitude);
  lcd.print(" m");
  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Rain: ");
```

```
  lcd.print(rainStatus);
  delay(1000);

  lcd.clear();
  delay(1000);
}
```

ESP8266 Code:

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>

//-------  WI-FI details ----------//
char ssid[] = "SSID"; //SSID here
char pass[]  = "PASSWORD"; // Password here
//--------------------------------//

//-----------  Channel details ----------------//
unsigned long Channel_ID = 123456; // Your Channel ID
const char * myWriteAPIKey = "ABC123CDE456"; //Your write API key
//----------------------------------------------//

const int Field_Humidity = 1;
const int Field_DHT_Temp = 2;
const int Field_BMP_Temp = 3;
const int Field_Pressure = 4;
const int Field_Altitude = 5;
const int Field_Rain = 6;

WiFiClient client;

float humidity = 0;
float dhtTemp = 0;
float bmpTemp = 0;
float pressure = 0;
float altitude = 0;
int rainValue = 0;

void setup()
{
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
  connectWiFi();
}
```

```
void loop()
{
  connectWiFi(); // Ensure WiFi is connected

  if (Serial.available() > 0)
  {
    delay(100);
    while (Serial.available() > 0)
    {
      String receivedData = Serial.readStringUntil('\n'); // Read full serial
line

      if (receivedData.startsWith("*") && receivedData.endsWith("#"))
      {
        receivedData.remove(0, 1); // Remove '*' at the start
        receivedData.remove(receivedData.length() - 1, 1); // Remove '#' at the
end

        // Split values from received data
        sscanf(receivedData.c_str(), "%f,%f,%f,%f,%f,%d", &humidity, &dhtTemp,
&bmpTemp, &pressure, &altitude, &rainValue);

        // Debugging output
        Serial.println("Received Data:");
        Serial.print("Humidity: "); Serial.println(humidity);
        Serial.print("DHT Temp: "); Serial.println(dhtTemp);
        Serial.print("BMP Temp: "); Serial.println(bmpTemp);
        Serial.print("Pressure: "); Serial.println(pressure);
        Serial.print("Altitude: "); Serial.println(altitude);
        Serial.print("Rain Sensor Value: "); Serial.println(rainValue);
      }
    }
  }

  uploadToThingSpeak();
}

void connectWiFi()
{
  if (WiFi.status() != WL_CONNECTED)
  {
    Serial.print("Connecting to WiFi");
    WiFi.begin(ssid, pass);
```

```
    int retries = 20;
    while (WiFi.status() != WL_CONNECTED && retries-- > 0)
    {
      delay(1000);
      Serial.print(".");
    }

    if (WiFi.status() == WL_CONNECTED)
      Serial.println("\nWiFi Connected!");
    else
      Serial.println("\nWiFi Connection Failed!");
  }
}

void uploadToThingSpeak()
{
  Serial.println("Uploading Data to ThingSpeak...");

  ThingSpeak.setField(Field_Humidity, humidity);
  ThingSpeak.setField(Field_DHT_Temp, dhtTemp);
  ThingSpeak.setField(Field_BMP_Temp, bmpTemp);
  ThingSpeak.setField(Field_Pressure, pressure);
  ThingSpeak.setField(Field_Altitude, altitude);
  ThingSpeak.setField(Field_Rain, rainValue);

  int response = ThingSpeak.writeFields(Channel_ID, myWriteAPIKey);

  if (response == 200)
    Serial.println("Upload Successful!");
  else
    Serial.print("Upload Failed. HTTP Error Code: "), Serial.println(response);

  delay(15000); // ThingSpeak rate limit (must wait at least 15s before next
update)
}
```

Node.js Code:

```
const express = require('express');

const http = require('http');

const { Server } = require('socket.io');

const { SerialPort } = require('serialport');
```

```
const { ReadlineParser } = require('@serialport/parser-readline');


const app = express();
const server = http.createServer(app);
const io = new Server(server);


const port = new SerialPort({ path: 'COM7', baudRate: 9600 });
const parser = port.pipe(new ReadlineParser({ delimiter: '\n' }));


app.use(express.static('public'));


parser.on('data', (line) => {
   console.log('Received:', line);
   io.emit('serialData', line);


   if (line.includes("Humidity:") && !line.includes("BMP")) {
      const match = line.match(/Humidity:\s*([\d.]+)/);
      if (match) io.emit('humidity', parseFloat(match[1]));
   } else if (line.includes("DHT Temp:")) {
      const match = line.match(/DHT Temp:\s*([\d.]+)/);
      if (match) io.emit('dhtTemp', parseFloat(match[1]));
   } else if (line.includes("BMP180 Temp:")) {
      const match = line.match(/BMP180 Temp:\s*([\d.]+)/);
      if (match) io.emit('bmpTemp', parseFloat(match[1]));
   } else if (line.includes("Pressure:")) {
      const match = line.match(/Pressure:\s*([\d.]+)/);
      if (match) io.emit('pressure', parseFloat(match[1]));
```

```
    } else if (line.includes("Altitude:")) {
        const match = line.match(/Altitude:\s*([\d.]+)/);
        if (match) io.emit('altitude', parseFloat(match[1]));
    } else if (line.includes("Rain Value:")) {
        const match = line.match(/Rain Value:\s*([\d.]+)/);
        if (match) io.emit('rain', parseFloat(match[1]));
    }
});

server.listen(3000, () => {
    console.log('Server listening on http://localhost:3000');
});
```