IoT-Based Weather Monitoring System using node mcu

MUHAMMED MIDHILAJ
S3 ER, Roll No: 41 Department of Electronics and Communication Engineering
https://github.com/muhammedmidhilaj1212-del/WEATHER-MONITORING-SYSTEM.git

Abstract

This project presents an IoT-based weather monitoring system that measures temperature, humidity, rain detection, and ambient light intensity in real time. Using sensors such as DHT11, rain sensor, and LDR, the system collects environmental data and transmits it via Wi-Fi using the ESP8266 NodeMCU. The data is logged to an IoT platform for remote monitoring through a smartphone or web dashboard. This solution is cost-effective, scalable, and useful for applications like smart agriculture and environmental monitoring.

1 Intro duction

Weather monitoring plays a vital role in agriculture, disaster management, and daily activities. Traditional manual observations are often time-consuming and inaccurate. The advent of IoT technology enables automatic measurement, recording, and transmission of weather parameters. This project designs and implements an IoT-enabled weather monitoring system to measure temperature, humidity, rain, and light intensity in real time.

2 ObjectivesoftheProject

1. Design a real-time IoT-enabled weather monitoring system.
2. Measure temperature and humidity using DHT11.
3. Detect rain using a rain sensor.
4. Measure light intensity using an LDR.
5. Transmit collected data to an IoT cloud platform.
6. Provide remote access via mobile/PC dashboard.
7. Offer a cost-effective monitoring solution.

3 Explanation

The IoT-based weather monitoring system uses a NodeMCU ESP8266 microcontroller as the main controller, interfacing with multiple sensors including the DHT11 for temperature and humidity, a rain sensor module for precipitation detection, and an LDR for measuring ambient light intensity. The ESP8266 connects to a Wi-Fi network and uploads sensor data to an IoT cloud platform such as ThingSpeak or Blynk for storage and visualization.

The DHT11 sensor outputs temperature and humidity readings in digital form. The rain sensor detects the presence of water droplets on its sensing surface, outputting an analog or digital signal depending on moisture. The LDR changes resistance with light intensity, allowing the system to determine day/night or light level variations. All these readings are processed by the NodeMCU, displayed on an LCD, and periodically transmitted to the cloud.

Users can access the real-time data remotely through a web or mobile dashboard. The system can also trigger alerts for significant weather events like rain detection or high temperature. This makes it suitable for smart agriculture, weather stations, and environmental monitoring.

4 Block Diagram of the System

Figure 1: Block Diagram of IoT Weather Monitoring System

5 Components Required

- ESP8266 NodeMCU – Wi-Fi microcontroller
- DHT11 Sensor – Temperature & Humidity
- Rain Sensor Module – Rain Detection
- LDR Sensor – Ambient Light
- Breadboard, Jumper Wires, 10kΩ Resistor, Power Supply

6 Circuit Diagram

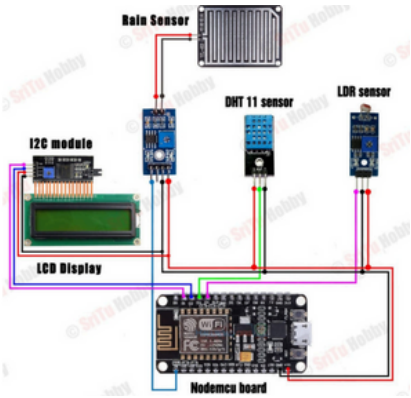

Figure 2: Circuit Diagram of Weather Monitoring System

7 WorkingPrinciple

The system continuously reads data from the sensors: temperature and humidity from the DHT11, rain detection from the rain sensor, and light intensity from the LDR. The ESP8266 NodeMCU processes this data, connects to Wi-Fi, and uploads it to a cloud platform such as ThingSpeak or Blynk using HTTP or MQTT. Users can then monitor the data in real time from a smartphone or web dashboard.

8 Expected Outputs/Outcomes

1. Real-time measurement of temperature, humidity, rain status, and light intensity.
2. Remote monitoring via IoT dashboard.
3. Automated alerts for weather changes.
4. Long-term data logging and trend analysis.
5. Cost-effective, scalable design for agriculture and environment.

9 Conclusion

The proposed IoT-based weather monitoring system using ESP8266, DHT11, rain sensor, and LDR demonstrates a low-cost, reliable, and scalable approach for real-time weather monitoring and data accessibility.

Code

```
// IoT-Based Weather Monitoring System Code
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ThingSpeak.h>


// Wi-Fi and ThingSpeak setup
const char* ssid = "your␣ssid";
const char* password = "password";
unsigned long myChannelNumber = channel_id;
const char* myWriteAPIKey = "write_api_key";
WiFiClient client;

// DHT11 setup
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// Sensor pins
#define LDR_PIN A0
#define RAIN_DIGITAL_PIN D5

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  Serial.begin(115200);
  Wire.begin(D2, D1);
  lcd.init(); lcd.backlight();
  dht.begin();
  pinMode(LDR_PIN, INPUT);
  pinMode(RAIN_DIGITAL_PIN, INPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { delay(500); }
  ThingSpeak.begin(client);
}

void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  int ldrValue = analogRead(LDR_PIN);
  int rainDigital = digitalRead(RAIN_DIGITAL_PIN);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("T:"); lcd.print(temp); lcd.print("C␣H:"); lcd.print(hum);
  lcd.setCursor(0,1);
  lcd.print((ldrValue<800)?"Bright":"Dark");
  lcd.print("␣"); lcd.print((rainDigital==0)?"Rainy":"Dry");

  ThingSpeak.setField(1, temp);
  ThingSpeak.setField(2, hum);
  ThingSpeak.setField(3, ldrValue);
  ThingSpeak.setField(4, rainDigital);
  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  delay(20000);
}
```

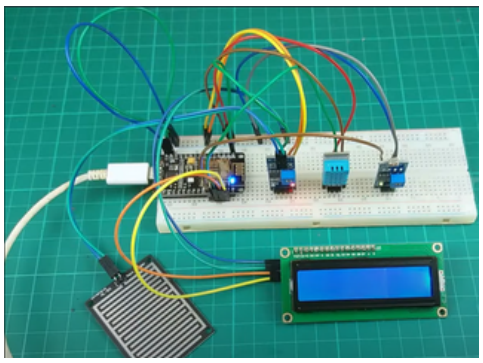

Figure 3: Working model image