Wireless Temperature and Humidity Monitoring System Using Node MCU and DHT11 Sensor

Rahul R. Avhad[1], Sham R. Avhad[2]
[1]*Electrical Engineer, AISSMS's Institute of Information Technology, Pune*
[2]*Computer Engineer, D. Y. Patil Institute of Engineering and Technology, Pune*

***Abstract*—This paper presents the design and implementation of a cost-effective, wireless temperature and humidity monitoring system using a Node MCU board (ESP8266) and a DHT11 sensor. The system captures real-time environmental data and uploads it to the thingspeak cloud platform, enabling remote monitoring through internet-connected devices. Battery power and a custom casing enhance portability, allowing flexible deployment in residential and industrial environments. The data can also be utilized for further analysis or to control other processes and devices.**

***Index Terms*—Node MCU board, DHT11 sensor, Temperature, humidity, thingspeak cloud platform.**

I. INTRODUCTION

In recent years, the proliferation of the Internet of Things (IoT) has paved the way for innovative solutions in environmental monitoring, where real-time data acquisition and remote accessibility are crucial. Temperature and humidity are fundamental parameters that affect various aspects of both residential and industrial environments, influencing everything from indoor air quality to the stability of manufacturing processes. Despite the availability of sophisticated monitoring systems, many are either prohibitively expensive or lack the flexibility required for diverse applications.

This paper presents the design and implementation of a cost-effective, wireless temperature and humidity monitoring system, utilizing a Node MCU board (ESP8266) and a DHT11 sensor. The system is engineered to provide real-time environmental data that is uploaded to the thingspeak cloud platform, enabling remote monitoring via any internet-connected device. The proposed solution combines the affordability and ease of use associated with the DHT11 sensor with the versatility of the Node MCU's built-in Wi-Fi capabilities, making it an ideal candidate for deployment in both residential and industrial settings.

To further enhance portability and user convenience, the system can be powered by a battery and housed in a custom casing, transforming it into a compact, handheld device. This feature allows the device to be easily relocated or used in various locations without dependence on a fixed power source, making it suitable for dynamic environments. Users can conveniently monitor temperature and humidity levels from anywhere, whether for maintaining optimal living conditions at home or ensuring the integrity of sensitive industrial processes. The application of this system spans various domains, including smart homes, greenhouses, server rooms, and production facilities, where maintaining precise environmental conditions is paramount.

This work aims to demonstrate that a low-cost, flexible, and user-friendly IoT solution can effectively meet the needs of diverse environments, offering a practical alternative to more complex and expensive monitoring systems. The following sections will detail the system's architecture, implementation process, and performance evaluation, illustrating its potential for widespread adoption.

II. RELATED WORK

The advancement of IoT has led to various systems for environmental monitoring, often using microcontrollers and sensors to collect and analyze data. Early systems frequently used Arduino boards with sensors like the DHT11 or DHT22 to monitor temperature and humidity, displaying data locally or on cloud platforms like thingspeak. These solutions, while effective, often required manual setup and lacked portability.

More advanced systems have utilized Raspberry Pi for greater processing power, enabling local data processing and integration with cloud services such as AWS IoT or Google Cloud IoT Core. However, these systems can be complex and costly for basic monitoring tasks.

The Node MCU (ESP8266) has emerged as a cost-effective alternative, integrating Wi-Fi capabilities with sensors like the DHT11 for compact, affordable monitoring solutions. Existing projects have successfully used Node MCU with cloud platforms like thingspeak, though many lack portability and user-friendly features.

This project enhances previous systems by combining the Node MCU's affordability with battery power and a custom casing, offering a portable and flexible solution. Additionally, it extends functionality by enabling data analysis and control of other devices through thingspeak addressing limitations of earlier systems.

III. METHODOLOGY

The system is designed to collect environmental data and upload it to the thingspeak cloud platform for remote monitoring. The following steps outline the approach taken to develop and deploy this system:

A) System Components

Node MCU Board (ESP8266): Acts as the microcontroller with integrated Wi-Fi capabilities. It handles data acquisition from the DHT11 sensor and communicates with the thingspeak cloud platform.

DHT11 Sensor: Measures temperature and relative humidity (RH). It provides digital output that the Node MCU processes.

Battery Power Supply: Provides power to the Node MCU and DHT11 sensor, enabling the system to be portable and independent of a fixed power source.

Custom Casing: Encloses the electronic components, protecting them and making the device more portable and user-friendly.

B) System Design

Circuit Diagram: The DHT11 sensor is connected to the Node MCU via three jumper wires. The sensor's data pin is connected to a digital input pin on the Node MCU, while power and ground pins are connected accordingly.

For circuit diagram refer to Appendix II Fig. 1, 2.

Software Development:

Programming Environment: The Node MCU is programmed using the Arduino Integrated Development Environment (IDE).

For source program refer to Appendix I.

Sensor Integration: The Arduino code reads data from the DHT11 sensor, including temperature and humidity values. The DHT library is used to facilitate communication with the sensor.

Data Upload: The collected data is formatted and sent to the thingspeak platform via HTTP requests. The thingspeak API is used to update channels with temperature and humidity data.

Code Structure: The code includes initialization routines for the sensor, data acquisition loops, and functions for connecting to Wi-Fi and sending data to thingspeak.

C) thingspeak Configuration:

Channel Setup: A thingspeak channel is created to store and visualize the data. Two fields are defined: one for temperature and one for humidity.

API Integration: The API key from the thingspeak channel is used in the Node MCU code to authenticate and upload data.

D) Testing and Calibration

Initial Testing: The system is tested in various environments to ensure accurate data collection and reliable data transmission to thingspeak.

Calibration: The DHT11 sensor is calibrated against known temperature and humidity standards to verify accuracy and adjust any discrepancies.

E) Deployment

Battery Integration: The system is powered using a 5V battery pack, allowing it to operate independently of a fixed power source.

Enclosure: The Node MCU and DHT11 sensor are housed in a custom-designed casing to protect the components and facilitate portability.

F) Data Utilization

Remote Monitoring: Users can access real-time data from the thingspeak dashboard via web or mobile applications.

Further Analysis: The data can be analyzed for trends, and triggers can be set up to control other devices or processes based on the environmental conditions recorded.

IV. RESULTS AND DISCUSSION

The developed wireless temperature and humidity monitoring system was tested to evaluate its

performance, accuracy, and portability. The system consistently transmitted real-time data to the thingspeak cloud platform, which was accurately visualized on separate graphs for temperature and humidity. The Node MCU maintained a stable Wi-Fi connection, ensuring reliable data transmission.

The DHT11 sensor provided temperature readings with an accuracy of ±2°C and humidity readings with ±5% RH, which is adequate for general residential and industrial use. However, the sensor may not meet the precision requirements of more specialized applications.

The addition of a battery power supply and custom casing enhanced the device's portability, allowing it to operate independently from a fixed power source and be easily relocated to different environments. This flexibility makes it suitable for dynamic monitoring scenarios, such as in various rooms of a home or industrial facility.

Furthermore, the data uploaded to thingspeak can be used for further analysis or integrated into automation systems, enabling the control of other devices based on environmental conditions. This extends the system's utility beyond basic monitoring.

While the system performs well for its intended purpose, future improvements could include using more accurate sensors and adding other environmental parameters, like air quality, to broaden its application scope.

For results refer to Appendix III Fig. 3, 4, 5.

V. CONCLUSION

This project successfully developed a cost-effective, portable system for monitoring temperature and humidity using a Node MCU board (ESP8266) and a DHT11 sensor. The system reliably uploads real-time data to thingspeak, allowing remote monitoring via any internet-connected device. With battery power and a custom casing, the device is easily deployable in various locations without needing a fixed power source.

The system's performance is suitable for residential and industrial applications, providing valuable environmental insights.

APPENDIX I

/*source code*/

```
/*upload temperature and humidity values to Thing speak
received through the DHT11 sensor*/
#include<ESP8266WiFi.h>
#include<WiFiClient.h>
#include<ESP8266HTTPClient.h>
#include <DHT11.h>
WiFiClient client;
HTTPClient http;
DHT11 dht11(2);
String url;
String API = "YOUR_API_KEY";
String FieldNo;
String FieldNo1 = "1";
String FieldNo2 = "2";
int httpcode;
String ssid = "YOUR_WIFI_SSID";
String passwrd = "YOUR_WIFI_PASSWORD";
void connectToWiFi(void);
void sendGETRequest(int data, String FieldNo);
void setup()
{
 Serial.begin(9600);
 connectToWiFi();
 pinMode(A0, INPUT);
}
void loop()
{
 int temperature = 0;
 int humidity = 0;
 // Attempt to read the temperature and humidity values from the DHT11 sensor.
 int result = dht11.readTemperatureHumidity(temperature, humidity);
```

```
  sendGETRequest(temperature, FieldNo1);
  delay(16000);
  sendGETRequest(humidity, FieldNo2);
  delay(16000);
}
void sendGETRequest(int data, String FieldNo)
{
  /*link*/
  /*GET
https://api.thingspeak.com/update?api_key=9YDA0
CJKNS7SELCX&field1=0*/
  url = "http://api.thingspeak.com/update?api_key=";
  url = url + API;
  url = url + "&field";
  url = url + FieldNo;
  url = url + "=";
  url = url + data;
  http.begin(client, url);
  Serial.println("waiting for response..");
  httpcode = http.GET();
  if(httpcode>0)
  {
    if(FieldNo=="1")
    {
      Serial.print(data);
      Serial.println(" - temperature data sent
successfully!!");
    }
    if(FieldNo=="2")
    {
      Serial.print(data);
      Serial.println(" - humidity data sent
successfully!!");
    }
  }
  else
  {
    Serial.println("Error in sending");
  }
  http.end();
}
void connectToWiFi(void)
{
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, passwrd);
  Serial.print("connect to wifi");
  if(WiFi.status()!=WL_CONNECTED)
  {
    Serial.print('.');
    delay(250);
  }
  Serial.println("connected!!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("mac address: ");
  Serial.println(WiFi.macAddress());
}
```

APPENDIX II

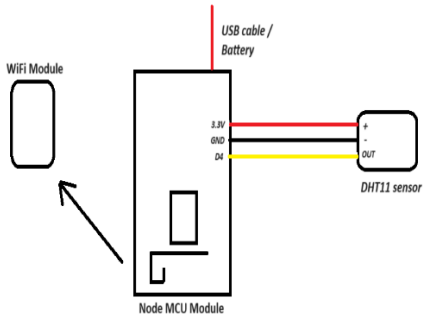

Fig1. Block diagram of device

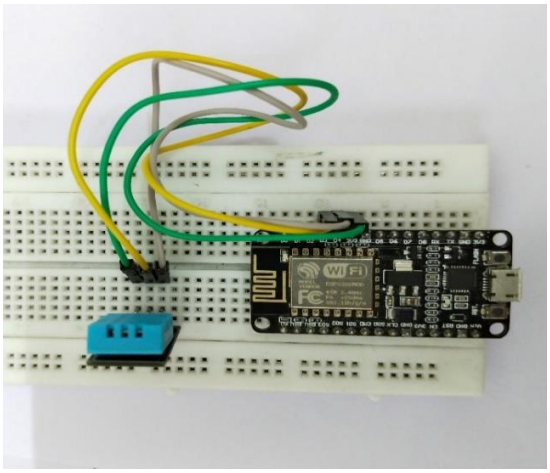

Fig2. Actual connections for device

APPENDIX III

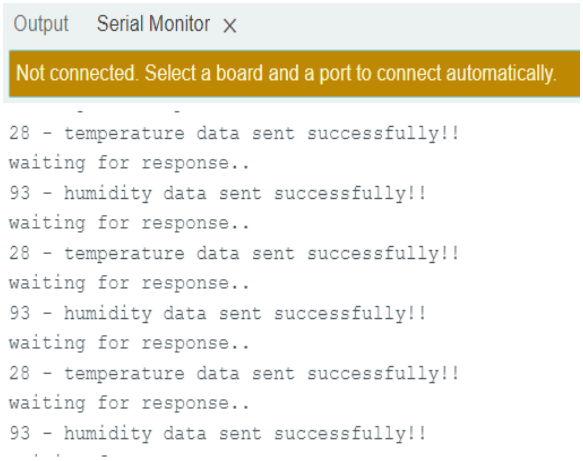

Fig3. Data sent from Node MCU

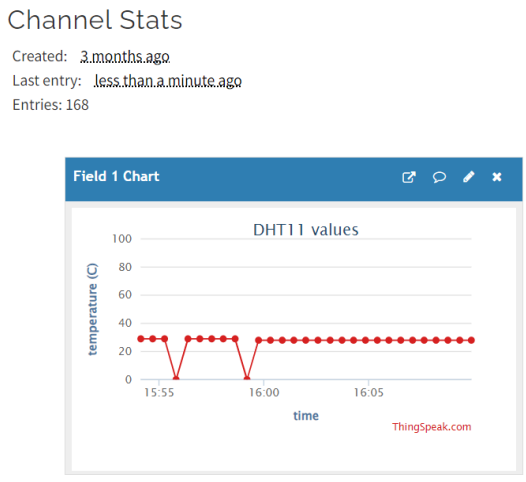

Fig4. Temperature data received on thingspeak portal

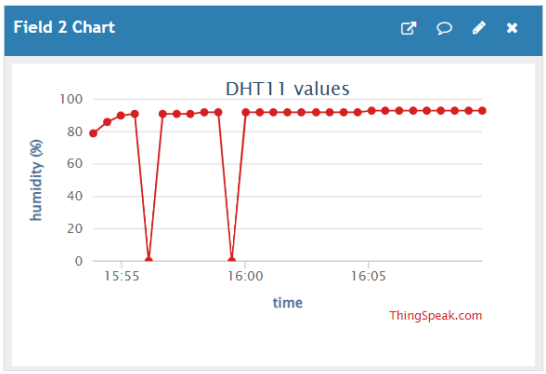

Fig5. Humidity data received on thingspeak portal

REFERENCES

[1] Y. Kanetkar, *Let Us C*, 16thed.BPB publications, India,2018.

[2] N. Rai, *Arduino Projects for Engineers*, 1sted.BPB Publications, India, 2023, ch.18.

[3] Y. Kanetkar, Shrirang Korde, *21 IoT Experiments*, 1sted.BPB Publications, India, 2023, ch.14.

[4] https://www.arduino.cc/education

[5] https://docs.arduino.cc/learn/starting-guide

[6] https://www.arduino.cc/en/software

[7] https://www.espressif.com/en/products/socs

[8]https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf

[9] https://docs.espressif.com/projects/esp8266-rtos-sdk/en/latest/get-started/index.html

[10]https://www.arduino.cc/reference/en/libraries/dht-sensor-library/

[11] https://github.com/ekstrand/ESP8266wifi

[12]https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiClient.h

[13]https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266HTTPClient/src/ESP8266HTTPClient.h

[14]https://www.tme.eu/Document/7a4fd48d400b8c4c8309ef1e2b13cdd4/MR003-005-1.pdf