Remote Environmental Monitoring System Using DHT11

Hajar Zrioual[1]; Isogun Toluwalase Adewale[2*]; Cherkaoui Bahaa Eddine[3]
[1]School of Electronic and Information Engineering, Nanjing University of Information Science & Technology, Nanjing 210044, China
[2]Jiangsu Key Laboratory of Meteorological Observation and Information Processing, School of Electronic and Information Engineering, Nanjing University of Information Science & Technology, Nanjing 210044, China

Abstract:- This document proposes the implementation of a remote environmental monitoring system using the DHT11 sensor, NodeMCU microcontroller, and ThingSpeak IoT analytics platform. It emphasizes the importance of temperature and humidity measurements in industrial applications and the value of real-time monitoring for safety and process optimization. The proposed system leverages the DHT11 sensor's capabilities to provide precise and continuous data, which is transmitted to ThingSpeak for visualization and analysis. The methodology section details the system's structural design, analytical approach, and code analysis, highlighting its adherence to IoT principles. The results and discussion section presents the system's performance, demonstrating its accuracy and potential for data-driven decision-making. However, it also acknowledges the limitations of the DHT11 sensor and suggests areas for further analysis and improvement. Overall, the document provides a comprehensive overview of the system's architecture, methodology, and results, showcasing its potential for environmental monitoring in industrial and IoT applications.

Keywords:- *DHT11, NodeMcu, IOT, Wireless Communication.*

I. INTRODUCTION

In the realm of environmental monitoring, the accurate measurement and analysis of temperature and humidity are paramount. These two variables are critical in determining the state and condition of substances or objects, making their monitoring essential in various sectors, particularly in industrial applications. The ability to track changes in temperature and humidity remotely not only enhances safety but also provides valuable insights for process optimization and decision-making.

One of the key tools for achieving remote monitoring of these environmental conditions is the DHT11 sensor. This compact and sensitive device is capable of detecting minute changes in temperature and humidity levels, providing real-time data that can be crucial for timely interventions and adjustments. The DHT11's ability to offer precise readings every second makes it an ideal choice for applications where continuous and immediate data updates are necessary.

To harness the capabilities of the DHT11 sensor for remote monitoring, a system centered around the NodeMCU microcontroller has been proposed. The NodeMCU, built on the ESP8266 Wi-Fi chip, serves as the control center of the system, interfacing with the DHT11 sensor and managing the transmission of collected data. This microcontroller is particularly well-suited for Internet of Things (IoT) applications due to its Wi-Fi connectivity, which allows for seamless data transfer over the internet.

The proposed system leverages the ThingSpeak IoT analytics platform to log and visualize the data collected by the DHT11 sensor. ThingSpeak provides a robust infrastructure for aggregating, visualizing, and analyzing live data streams in the cloud, making it an excellent choice for real-time monitoring applications. By integrating ThingSpeak into the system, users can easily access and analyze temperature and humidity data from anywhere in the world, enabling them to make informed decisions and take necessary actions promptly.

The system's architecture involves the creation of a ThingSpeak channel dedicated to storing the temperature and humidity data. The NodeMCU, programmed using the Arduino IDE, reads the sensor data and sends it to the ThingSpeak channel over a Wi-Fi network. The ThingSpeak channel then displays the data in real-time, allowing users to monitor environmental conditions and set up alerts or automated responses based on predefined thresholds.

This innovative approach to remote monitoring not only enhances the safety and efficiency of industrial processes but also opens up possibilities for data-driven decision-making. By integrating the DHT11 sensor with the NodeMCU and ThingSpeak, the system offers a cost-effective and reliable solution for real-time environmental monitoring, making it a valuable tool in various industrial and environmental applications.

II. LITERATURE REVIEW

Monitoring is an essential practice across many fields, involving the continuous collection of data to evaluate environmental or operational conditions and ensure adherence to desired standards. In environmental science and industrial management, especially within IoT (Internet of Things) applications, monitoring allows for real-time oversight, which is crucial for maintaining safety, optimizing processes, and ensuring timely interventions. Effective monitoring requires high-frequency data collection and analysis, helping stakeholders identify trends, detect anomalies, and assess the performance of systems over time [1]. In IoT-based environmental monitoring, real-time data can lead to actionable insights, particularly in remote and resource-constrained environments where manual checks are impractical.

The NodeMCU ESP8266 module, an IoT-friendly microcontroller platform, is known for its Wi-Fi capabilities, which make it suitable for remote data transmission. Developed around the ESP8266 chip, NodeMCU integrates a range of I/O pins and can be programmed through the Arduino IDE, thus making it accessible and highly flexible for IoT projects. This module is particularly valuable for applications requiring wireless connectivity, where data needs to be communicated to cloud-based systems for further processing and analysis. Unlike traditional microcontrollers such as Arduino, the NodeMCU's built-in Wi-Fi facilitates seamless internet integration, enabling efficient and scalable IoT applications [2].

Sensors are critical components of IoT systems, particularly in environmental monitoring and robotics, where they serve as the primary sources of data. By translating physical conditions, such as temperature, humidity, and pressure, into digital signals, sensors provide real-time inputs to the microcontroller, which then processes and transmits the data for analysis. In control systems and robotics, sensors enable automation by providing feedback that informs decision-making processes [3]. Environmental sensors, such as those used in IoT applications, must deliver consistent, reliable data to support accurate environmental analysis and timely responses.

The DHT11 is a commonly used digital sensor for measuring temperature and humidity, known for its affordability and ease of integration with microcontrollers like the NodeMCU. The DHT11 sensor provides stable readings, which makes it useful in non-critical applications, including indoor environmental monitoring. The sensor's one-time programmable (OTP) memory enables calibrated measurements, enhancing its accuracy and stability. However, it is limited by a lower precision compared to advanced sensors, which should be considered when applying it in contexts that demand higher accuracy. This limitation positions the DHT11 as more suited to IoT projects where relative rather than absolute accuracy suffices.

ThingSpeak is a cloud-based IoT analytics platform that allows users to aggregate, visualize, and analyze real-time data. With support for MATLAB, ThingSpeak enables advanced data processing and custom visualizations, which are essential for drawing insights from continuous data streams. ThingSpeak's capability to create custom channels for data logging and trigger actions based on specified conditions adds to its appeal for remote monitoring applications [4]. By integrating with devices such as the NodeMCU, ThingSpeak facilitates seamless data collection and enhances the IoT system's ability to analyze and react to environmental changes remotely[5].

III. METHODOLOGY

➢ *Structural Design and Analytical Approach*

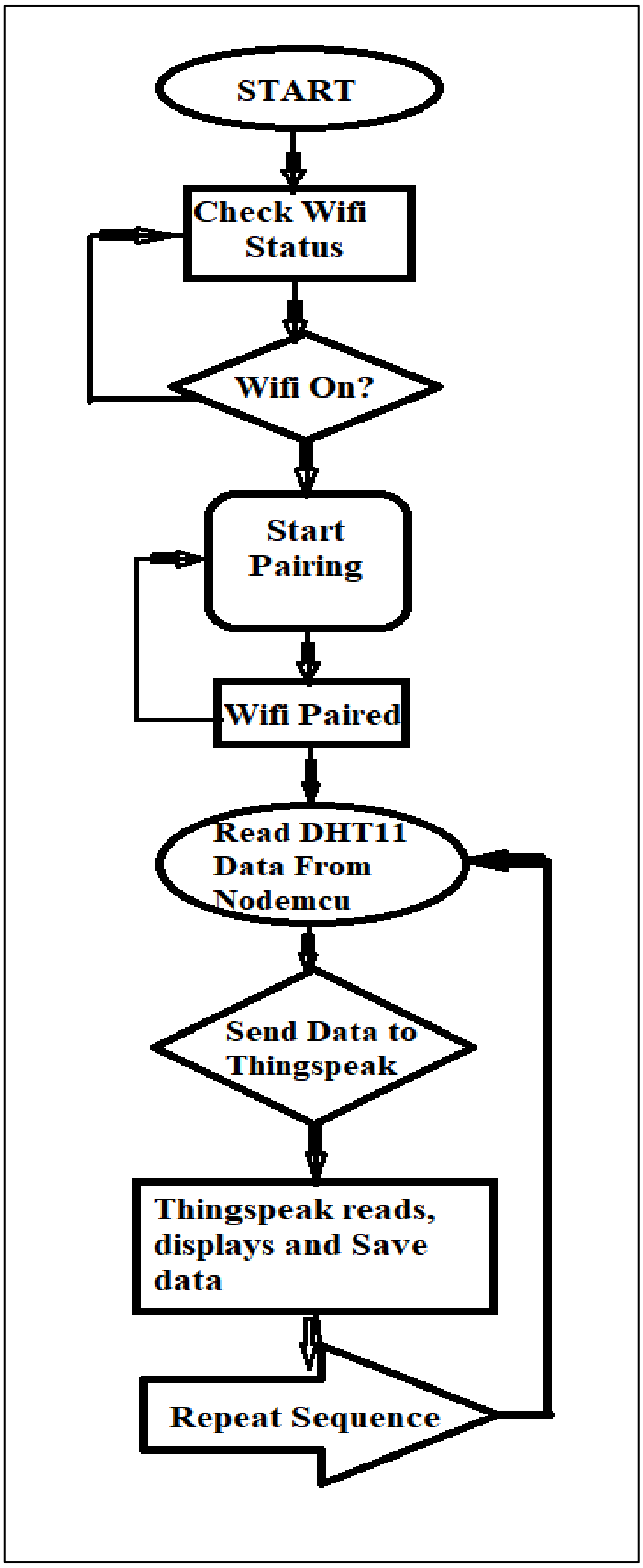


Fig 1 FlowChart

The provided flowchart details the operations of a NodeMCU DHT11 temperature and humidity monitoring system, illustrating the entire process from initialization to data transmission and repetition. This flowchart serves as a blueprint for understanding how the system operates in a sequential manner to ensure continuous monitoring and data handling.

The process begins with the system being initialized, as indicated by the "Start" node. Immediately, the system checks for the availability of Wi-Fi, a crucial component for data transmission to the cloud. If the Wi-Fi is found to be unavailable, the system loops back, continually checking until a connection is established. This loop ensures that the system does not proceed without a stable internet connection.

Upon confirming that Wi-Fi is available, the system then verifies whether the Wi-Fi is turned on. This step is critical as it prevents the system from attempting to pair with a network that is not active. If the Wi-Fi is not turned on, the system rechecks the Wi-Fi status, creating a cycle that ensures the system only proceeds under optimal conditions.

Once the Wi-Fi is confirmed to be on, the system initiates the pairing process. This step involves connecting the NodeMCU device to the Wi-Fi network. The flowchart highlights that this is not always a one-time process; if the pairing fails, the system will retry until a successful connection is established. This iterative process is vital for ensuring a reliable connection to the network which is necessary for subsequent data transmission. Following a successful Wi-Fi pairing, the system proceeds to read data from the NodeMCU device. The NodeMCU, equipped with a DHT11 sensor, collects temperature and humidity data. This data is then prepared for transmission to the ThingSpeak platform, a cloud-based service that allows for the storage, analysis, and visualization of sensor data.

The flowchart indicates that the next step involves sending the collected data to ThingSpeak. This step is crucial as it enables remote monitoring and analysis. Once the data is successfully sent, ThingSpeak reads, displays, and saves the data. This ensures that the collected data is not only stored but also readily accessible for real-time monitoring and historical analysis.

Finally, the system is designed to repeat this entire sequence, starting again with checking the Wi-Fi status. This repetition ensures continuous monitoring, making the system robust and reliable for long-term data collection and analysis.

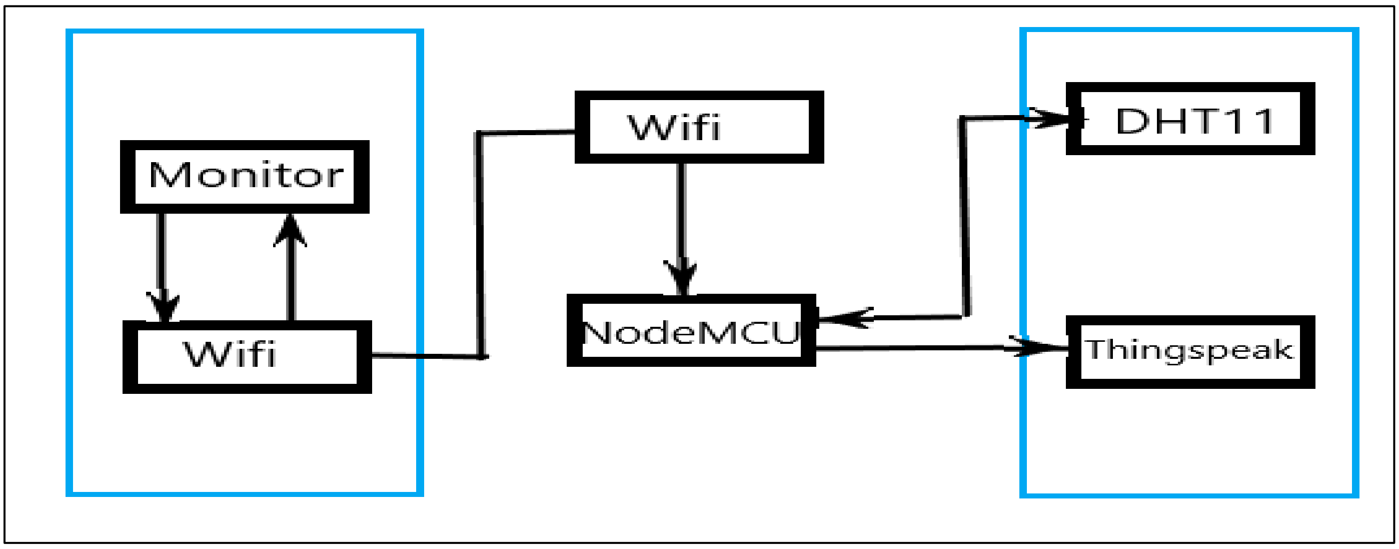


Fig 2 Circuit Block Diagram

In the design phase, a block diagram of the system is created, which can be viewed in the accompanying figure. Hardware design is crucial in tool creation, as it involves analyzing the components to ensure the tool functions as intended. To achieve optimal performance, it's essential to start with a solid design that considers the properties and characteristics of each component, thus preventing damage and facilitating the work process. In this particular design, the DHT11 temperature and humidity sensor is utilized, which provides an analog voltage output that can be further processed by the NodeMCU microcontroller development board. This sensor is known for its accurate calibration of temperature and humidity readings. The circuit diagram and pictorial diagram for the NodeMCU with the DHT11 sensor is depicted in the figure below.

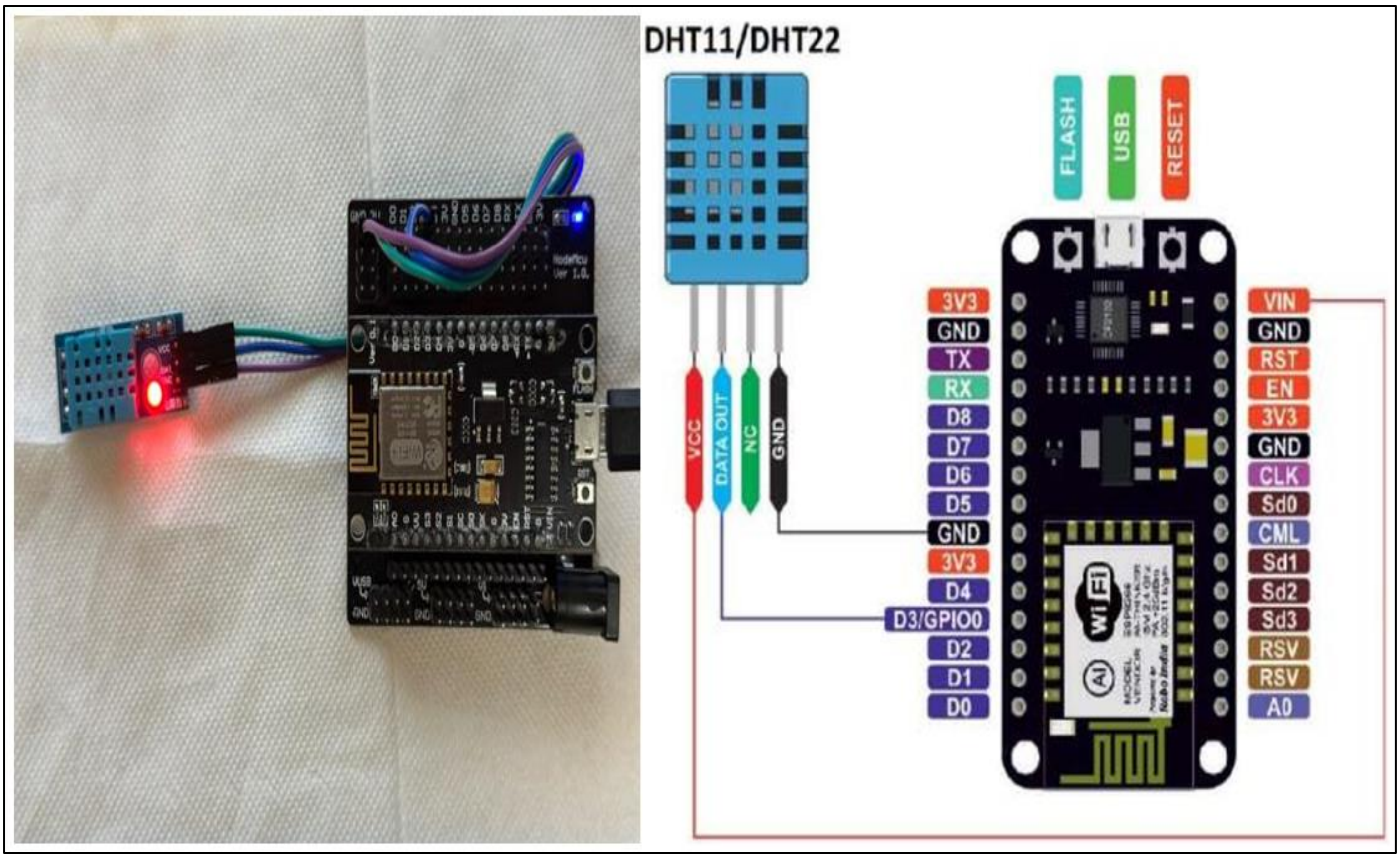


Fig 3 Circuit Schematic Diagram

The provided Arduino code is designed to read temperature and humidity data from a DHT11 sensor and send this data to a ThingSpeak channel for remote monitoring and data logging. This code is particularly suited for IoT applications due to its use of Wi-Fi connectivity and cloud-based data storage.

```
#include <DHT.h>  // Including library for dht
#include <ESP8266WiFi.h>
String apiKey = "C1BL6C6EW2TXRJD2";     // Enter your
Write API key from ThingSpeak
const char *ssid =  "MDT";     // replace with your wifi ssid
and wpa2 key
const char *pass =  "mdtolu306";
const char* server = "api.thingspeak.com";
#define DHTPIN D3          //pin where the dht11 is
connected
DHT dht(DHTPIN, DHT11);
WiFiClient client;
void setup()
{
Serial.begin(115200);
delay(10);
dht.begin();
Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);
while (WiFi.status() != WL_CONNECTED)
{
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
}
void loop()
{
float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t))
{
Serial.println("Failed to read from DHT sensor!");
return;
}
if (client.connect(server,80))    //    "184.106.153.149" or
api.thingspeak.com
{
String postStr = apiKey;
postStr +="&field1=";
postStr += String(t);
postStr +="&field2=";
postStr += String(h);
postStr += "\r\n\r\n";

client.print("POST /update HTTP/1.1\n");
client.print("Host: api.thingspeak.com\n");
client.print("Connection: close\n");
client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
client.print("Content-Type:       application/x-www-form-
urlencoded\n");
client.print("Content-Length: ");
client.print(postStr.length());
client.print("\n\n");
client.print(postStr);
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" degrees Celcius, Humidity: ");
Serial.print(h);
Serial.println("%. Send to Thingspeak.");
}
client.stop();
Serial.println("Waiting...");
// thingspeak needs minimum 15 sec delay between updates
delay(1000);
}
```

Let's break down the code and its underlying principles:

#include <DHT.h>: This library is necessary for interfacing with the DHT11 sensor. It handles the communication protocol required to read data from the sensor.

#include <ESP8266WiFi.h>: This library is used for connecting the NodeMCU (based on ESP8266) to a Wi-Fi network. It's essential for enabling the IoT functionality of the device.

String apiKey = "C1BL6C6EW2TXRJD2";: This is the Write API key from ThingSpeak, which is used to authenticate and send data to a specific ThingSpeak channel.

const char *ssid = "MDT"; and ***const char *pass = "mdtolu306";***: These are the credentials for the Wi-Fi network that the NodeMCU will connect to.

const char* server = "api.thingspeak.com";: This is the server address for ThingSpeak's API.
#define DHTPIN D3: This defines the pin on the NodeMCU where the DHT11 sensor is connected.

DHT dht(DHTPIN, DHT11);: This creates an instance of the DHT class, specifying the pin and the type of sensor (DHT11).

The ***setup()*** function initializes the serial communication, sets up the DHT11 sensor, and connects the NodeMCU to the specified Wi-Fi network. It prints feedback to the serial monitor during the connection process.

The ***loop()*** function reads the temperature and humidity values from the DHT11 sensor.

It checks if the readings are valid (not ***NaN***, which stands for "Not a Number", indicating an error).

If the readings are valid, it attempts to connect to the ThingSpeak server.

Upon successful connection, it prepares a POST request to send the data to ThingSpeak. The request includes the API key, temperature ***(field1)***, and humidity ***(field2)*** values.

The data is sent in the form of a URL-encoded HTTP body, and the length of the data is also specified in the request headers.

After sending the data, the client connection is closed, and the function prints a confirmation message to the serial monitor.

The loop includes a delay to comply with ThingSpeak's API rate limits (15 seconds minimum between updates).

Connectivity: The code leverages Wi-Fi connectivity to send data to the cloud, which is a fundamental aspect of IoT.

Data Logging: By using ThingSpeak, the code enables real-time data logging and monitoring, which is valuable for analyzing environmental conditions over time.

Scalability: ThingSpeak supports multiple devices and can scale to accommodate large amounts of data, making this approach suitable for expanding IoT networks.

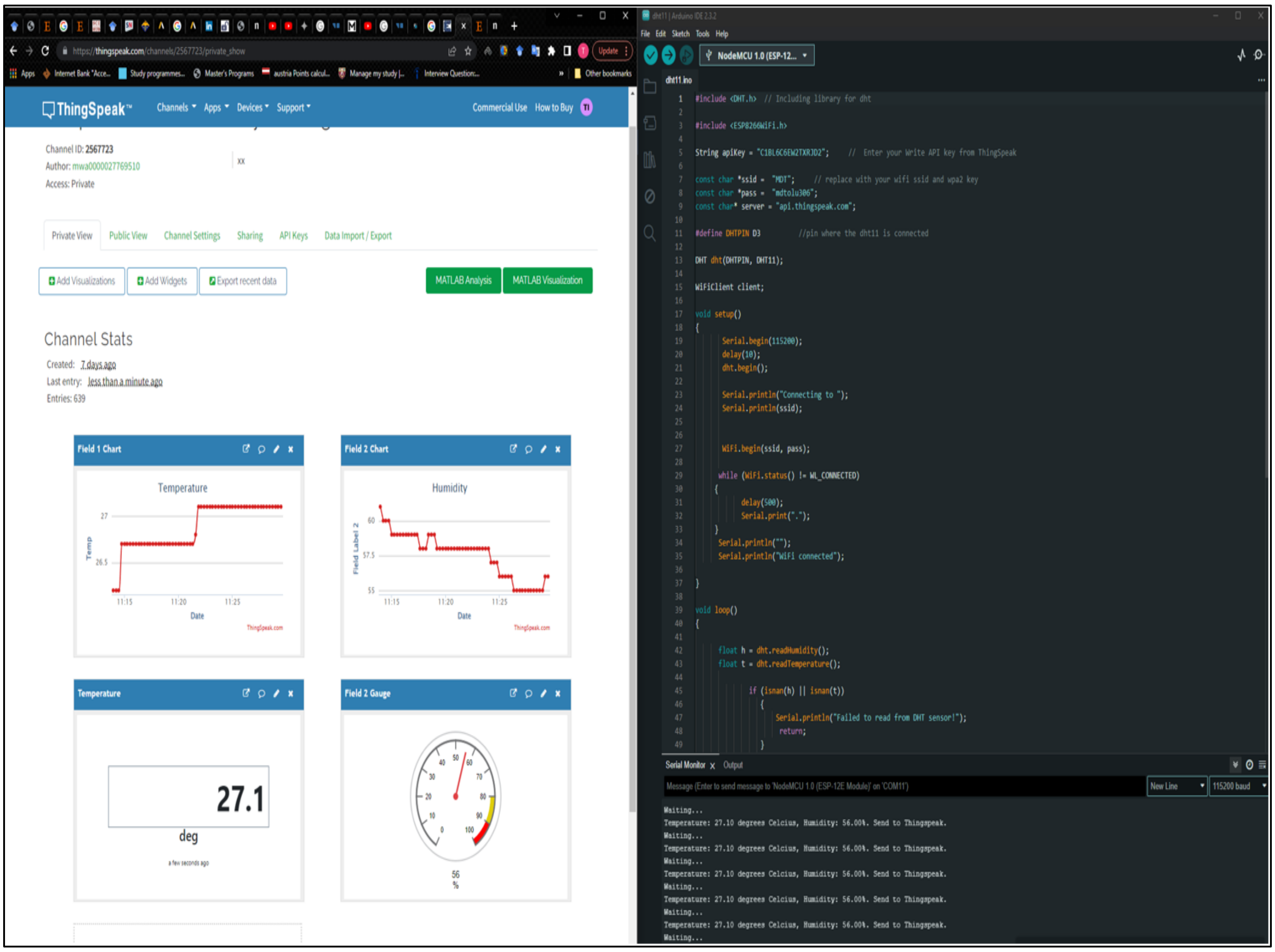
Accessibility: The data is stored in the cloud, making it accessible from anywhere, which is beneficial for remote monitoring and global access.

Ease of Use: The Arduino IDE and the libraries used in the code are well-documented and widely used, making the development process relatively straightforward for developers familiar with the Arduino ecosystem.

Overall, this code exemplifies the core principles of IoT and Wireless communication by integrating sensors, connectivity, and cloud services to create a functional remote monitoring system.

IV. RESULTS AND DISCUSSION

The figure in question appears to display the output of a temperature and humidity monitoring system based on a DHT11 sensor interfaced with a NodeMCU board, with data being transmitted to ThingSpeak for visualization and Arduino IDE's serial monitor for debugging and immediate feedback.



Channel ID: 2567723
Author: mwa0000027769510
Access: Private

Channel Stats
Created: 7 days ago
Last entry: less than a minute ago
Entries: 639

dht11 | Arduino IDE 2.3.2

NodeMCU 1.0 (ESP-12...

dht11.ino

```
#include <DHT.h>  // Including library for dht

#include <ESP8266WiFi.h>

String apiKey = "C1BL6C6EW2TXRJD2";    //  Enter your Write API key from ThingSpeak

const char *ssid =  "MDT";     // replace with your wifi ssid and wpa2 key
const char *pass =  "mdtolu306";
const char* server = "api.thingspeak.com";

#define DHTPIN D3          //pin where the dht11 is connected

DHT dht(DHTPIN, DHT11);

WiFiClient client;

void setup()
{
       Serial.begin(115200);
       delay(10);
       dht.begin();

       Serial.println("Connecting to ");
       Serial.println(ssid);


       WiFi.begin(ssid, pass);

      while (WiFi.status() != WL_CONNECTED)
     {
            delay(500);
            Serial.print(".");
     }
      Serial.println("");
      Serial.println("WiFi connected");

}

void loop()
{

      float h = dht.readHumidity();
      float t = dht.readTemperature();

              if (isnan(h) || isnan(t))
                 {
                      Serial.println("Failed to read from DHT sensor!");
                       return;
                 }
```

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM11')

```
Waiting...
Temperature: 27.10 degrees Celcius, Humidity: 56.00%. Send to Thingspeak.
Waiting...
Temperature: 27.10 degrees Celcius, Humidity: 56.00%. Send to Thingspeak.
Waiting...
Temperature: 27.10 degrees Celcius, Humidity: 56.00%. Send to Thingspeak.
Waiting...
Temperature: 27.10 degrees Celcius, Humidity: 56.00%. Send to Thingspeak.
Waiting...
Temperature: 27.10 degrees Celcius, Humidity: 56.00%. Send to Thingspeak.
Waiting...
```

Fig 4 Thingspeak and Arduino IDE Live Interface

On the left side of the figure, we can see the ThingSpeak web interface, which is displaying two main types of visual data representation: line charts and gauges for temperature and humidity. The temperature line chart shows a small increase over a short period of time, with the temperature rising from approximately 26.5 degrees to 27 degrees Celsius. The corresponding gauge below the line chart reflects the latest temperature reading, which is 27.1 degrees Celsius, labeled as "a few seconds ago," indicating a near real-time update of the data.

The humidity data is similarly represented, with its own line chart and gauge. The humidity chart shows a slight decrease over the same period, dropping from about 57.5% to just below 56%. The gauge for humidity indicates a current reading of 56%, consistent with the latest data point on the line chart.

On the right side of the figure, the Arduino IDE's serial monitor is open, displaying a log of operations performed by the NodeMCU board. The code snippet above the serial monitor shows a portion of the Arduino sketch used to read data from the DHT11 sensor and then send it to ThingSpeak using Wi-Fi. We can observe the repetitive log entries on the serial monitor confirming the temperature and humidity values, 27.10 degrees Celsius and 56.00% humidity, which are successfully being sent to ThingSpeak. The log messages indicate that the system is operating as expected, consistently reading from the sensor and transmitting the data.

The consistency between the readings on the serial monitor and the ThingSpeak visualizations suggests that the system is accurately capturing and relaying the sensor data. The slight variations in temperature and humidity over the brief time window displayed on ThingSpeak's charts suggest normal fluctuations in environmental conditions.

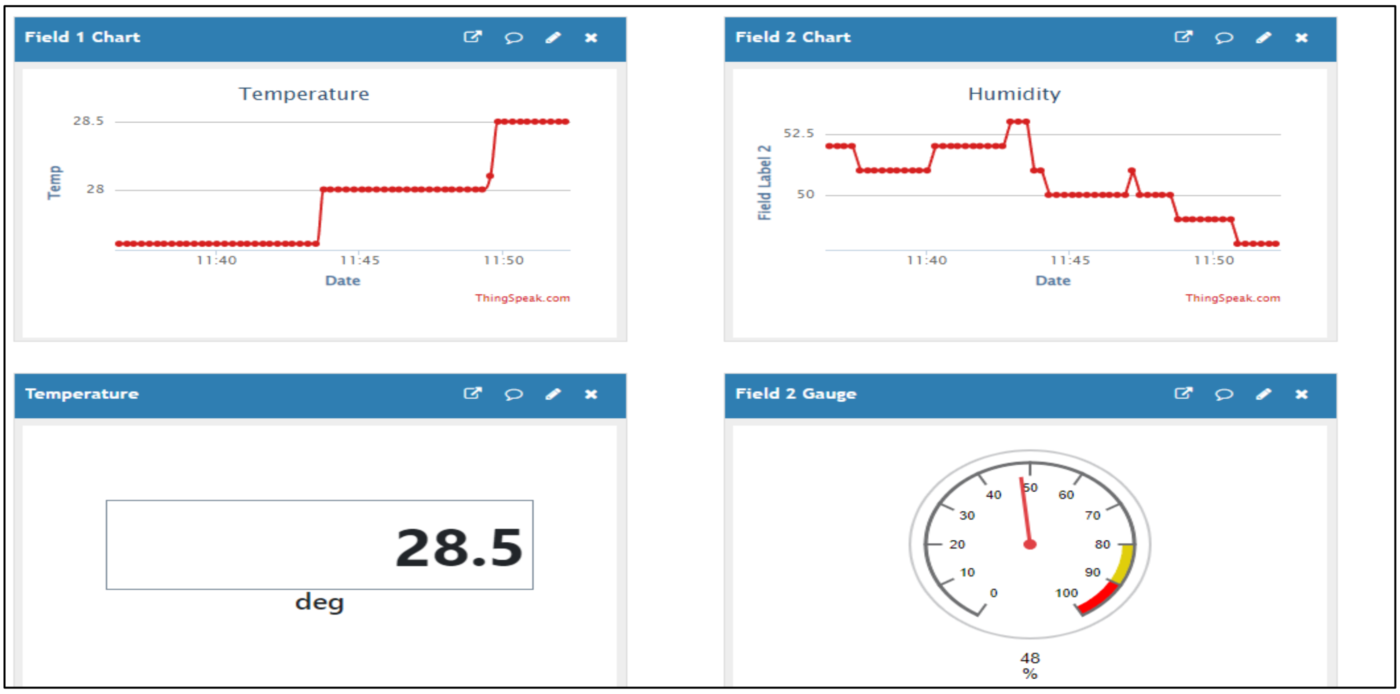


Fig 5 Thingspeak Live Data Interface

In discussing the results, it is important to note that the DHT11 sensor has certain accuracy limitations and is generally considered suitable for hobbyist projects or non-critical applications. The sensor's performance and the system's reliability can be influenced by factors such as sensor placement, ambient conditions, and the stability of the Wi-Fi connection used for data transmission.

This monitoring system provides a useful example of how IoT devices can be used for environmental monitoring, with the ability to remotely track and analyze data over time. For further analysis, one could look at longer-term data trends, examine data from multiple sensors for comparison, or explore the use of alerts or triggers based on certain thresholds or conditions detected by the system.

V. CONCLUSION

The document presents a comprehensive exploration of remote environmental monitoring using the DHT11 sensor, NodeMCU microcontroller, and ThingSpeak IoT analytics platform. It emphasizes the significance of temperature and humidity measurements in various industrial applications and the critical role of monitoring in decision-making and process optimization. The proposed system's architecture, methodology, and results demonstrate the successful integration of these technologies to enable real-time data collection, visualization, and analysis. The code analysis and IoT principles highlight the system's adherence to fundamental IoT concepts, making it a valuable tool for environmental monitoring in industrial and IoT applications. The results and discussion section provides insights into the system's performance, emphasizing its accuracy and potential for data-driven decision-making. However, it also acknowledges the limitations of the DHT11 sensor and suggests areas for further analysis and improvement.

REFERENCES

[1]. Leigh, C., Alsibai, O., Hyndman, R., Kandanaarachchi, S., King, O., McGree, J., Neelamraju, C., Strauss, J., Talagala, P., Turner, R., Mengersen, K., & Peterson, E. (2018). A framework for automated anomaly detection in high frequency water-quality data from in situ sensors.. The Science of the total environment, 664, 885-898 . https://doi.org/10.1016/j.scitotenv.2019.02.085.

[2]. Shakthidhar, S., Srikrishnan, P., Santhosh, S., & Sandhya, M. (2019). Arduino and NodeMcu based Ingenious Household Objects Monitoring and Control Environment. 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), 1, 119-124. https://doi.org/10.1109/ICONSTEM.2019.8918730.

[3]. Kondratenko, Y., Atamanyuk, I., Sidenko, I., Kondratenko, G., & Sichevskyi, S. (2022). Machine Learning Techniques for Increasing Efficiency of the Robot's Sensor and Control Information Processing. Sensors (Basel, Switzerland), 22. https://doi.org/10.3390/s22031062.

[4]. Susheela, K., Harshitha, E., Rohitha, M., & Reddy, S. (2019). Home Automation and E-Monitoring Over ThingSpeak and Android App. Lecture Notes in Networks and Systems. https://doi.org/10.1007/978-981-13-3765-9_14.

[5]. Lin J-Y, Tu H-L, Lee W-H. An Integrated Wireless Multi-Sensor System for Monitoring the Water Quality of Aquaculture. Sensors. 2021;21(12):4038. doi:10.3390/s21124038