# 

# 

# **Home Temperature Monitoring System**

# 

**Table of Contents**

[Home Temperature Monitoring System 1](#_Toc1)

[Internet of Things 1](#_Toc3)

[List of Figures 2](#_Toc4)

[Section 1: Introduction 3](#_Toc5)

[Section 2: Analysis 3](#_Toc6)

[Section 3: Description of the Solution 4](#_Toc7)

[Section 4: Functional Requirements 4](#_Toc8)

[Section 5: Resources and Components 5](#_Toc9)

[5.1 Arduino Uno R3 6](#_Toc10)

[5.2 WiFi Module (ESP8266) 6](#_Toc11)

[5.3 Temperature Sensor (TMP36) 6](#_Toc12)

[5.4 1 KΩ Resistor 7](#_Toc13)

[Section 6: Architectural Design 7](#_Toc14)

[6.1 Client 7](#_Toc15)

[6.2 Server (Platform for Thingspeak) 8](#_Toc16)

[Section 7: System Diagram 8](#_Toc17)

[Section 8: Protocols 9](#_Toc18)

[Section 9: Programming Codes 10](#_Toc19)

[9.1 Explanation of the Codes 11](#_Toc20)

[Section 10: Running of the Codes 12](#_Toc21)

[Section 11: The Operation of the System 13](#_Toc22)

[Section 12: Issues And Application Areas Of The System 16](#_Toc23)

[Section 13: Conclusion 17](#_Toc24)

[Section 14: References 18](#_Toc25)

## **List of Figures**

[Figure 1 Architectural Design 8](#_Toc1)

[Figure 2 Circuit Diagram 9](#_Toc2)

[Figure 3 Serial Monitor Output 14](#_Toc3)

[Figure 4 ThingSpeak Temperature readings 15](#_Toc4)

[Figure 5 Temperature Data Visualization 15](#_Toc5)

[Figure 6 Temperature Data Analysis 16](#_Toc6)

## **Section 1: Introduction**

This documentation's primary goal is to give a thorough overview of the system being created with the help of the available source code. The device collects temperature information from an analog pin and transmits it wirelessly to the ThingSpeak platform using the ESP8266 microcontroller. The system's analysis, solution description, functional requirements, components, architectural design, protocols, programming codes, running instructions, system operation, and prospective application domains are all covered in this documentation.

## **Section 2: Analysis**

It is crucial to comprehend the study carried out before the system was developed before getting into the technical specifics of the system. Real-time temperature monitoring and recording is the system's goal, which can be helpful in several applications, including home automation, industrial process control, and environmental monitoring. The system offers a low-cost and effective solution for temperature monitoring by employing the ESP8266 microcontroller and the ThingSpeak platform.

In the analysis phase, the system requirements were evaluated while considering data correctness, dependability, scalability, and ease of implementation. The connection with the ThingSpeak platform was also assessed to verify compatibility and effective data transmission. The investigation also considered the limitations of the ESP8266 microcontroller, such as its small memory and processing abilities needed for code optimization to bring about efficiency.

## **Section 3: Description of the Solution**

The created system uses the ThingSpeak platform and ESP8266 microcontroller to provide a reliable temperature monitoring solution. The fix is connecting via WiFi, gathering temperature information from an analog pin, and sending it to the ThingSpeak platform for archival and display. Every 10 seconds, the system sends temperature data in a loop-continuous operation.

The microcontroller is programmed using the Arduino framework to implement the solution. The program sets up a TCP connection with the ThingSpeak server, initializes the ESP8266, connects it to the specified WiFi network, and sends the temperature information as an HTTP request. Before being included in the HTTP request, the analog temperature value is mapped to the required range. The system uses the ESP8266 microcontroller's capabilities, making it the perfect option for IoT-based temperature monitoring applications.

## **Section 4: Functional Requirements**

The functional requirements are defined based on the system's expected behavior and intended purpose. The following are the main applicable requirements:

WiFi connection: The system should connect to the specified WiFi network using the provided SSID and password. Therefore this makes it possible for the ESP8266 microcontroller to connect to the ThingSpeak server online.

Gather Temperature Information: The system must read temperature information from an analog pin and map it to a specified range. This guarantees that the temperature measurements accurately reflect actual environmental conditions and are simple for the ThingSpeak platform to understand.

Data transmission to ThingSpeak: The system should transmit the temperature data to the ThingSpeak platform using an HTTP request. This allows temperature data to be stored and displayed on the ThingSpeak server, allowing users to view and examine the gathered information from a distance.

Periodic Data Transmission: The system should regularly repeat the data transmission procedure to ensure ongoing temperature monitoring. The technology provides real-time updates and guarantees a steady stream of data to the ThingSpeak platform by sending data every 10 seconds.

## **Section 5: Resources and Components**

The following components and resources make up the system:

|  |  |  |
| --- | --- | --- |
| **Name** | **Quantity** | **Component** |
| U1 | 1 | Arduino Uno R3 |
| U2 | 1 | Wifi Module(ESP8266) |
| U3 | 1 | Temperature Sensor[TMP36] |
| R1  R2 | 2 | 1 KΩ Resistor |

*Table 1 components*

### 5.1 Arduino Uno R3

The Arduino Uno R3 is an ATmega328P-based microcontroller board. It acts as the system's primary control component. The system's overall functionality is coordinated, the coding logic is carried out, and other features are interfaced with the Arduino Uno R3. The Arduino Uno R3 board connects serially to the ESP8266 WiFi module, reads analog data from the temperature sensor, and issues instructions for sending the data to the ThingSpeak server (Nahid & Khan, 2021).

### 5.2 WiFi Module (ESP8266)

The affordable ESP8266 WiFi module provides the system's wireless connectivity. It serves as the system's internet connection and as the conduit between the Arduino Uno R3 and the ThingSpeak server, allowing the system to join a WiFi network. The ESP8266 module is configured to connect via TCP to the ThingSpeak server and send the temperature information that the Arduino Uno R3 has collected, just as stated by Balakrishna and Thirumaran (2019). It is essential to the system's ability to support remote monitoring and data transmission.

### 5.3 Temperature Sensor (TMP36)

The TMP36 is an analog temperature sensor to gauge the surrounding air's temperature (Kalai Selvi et al., 2022). It is a reasonably priced sensor that offers precise temperature readings across a broad range. The Arduino Uno R3's analog input pins are linked to one of the system's TMP36 sensor's inputs. The map function on the Arduino Uno R3 reads the analog voltage output from the temperature sensor and translates it into a temperature measurement. The HTTP request is then built using the temperature information gathered from the TMP36 sensor, and it is sent to the ThingSpeak server for storage and viewing.

### 5.4 1 KΩ Resistor

The temperature sensor (TMP36) and the 1 K resistor work together as a voltage divider. Between the output pin of the temperature sensor and the ground, a resistor is attached. In order to keep the voltage levels within the permitted range for the analog input pin of the Arduino Uno R3, this resistor aids in scaling down the voltage output from the temperature sensor. The voltage divider circuit, which includes a resistor, enables the temperature sensor to give the system precise temperature values (Aniebiet & Fidelis, 2020).

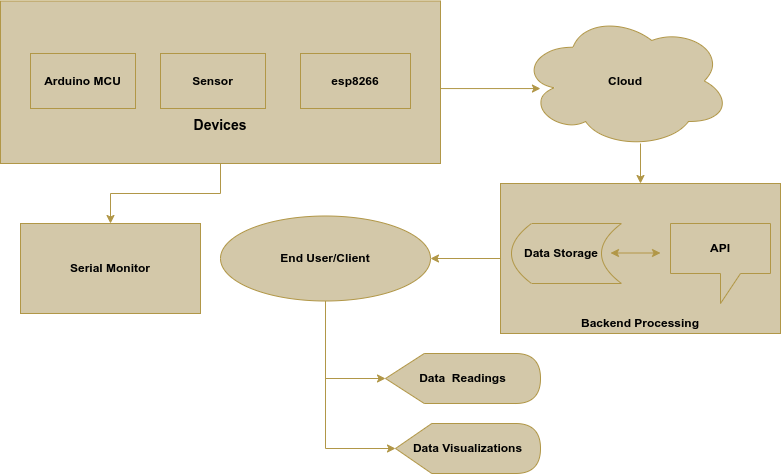
## **Section 6: Architectural Design**

The ESP8266 microcontroller is the client in the system's client-server architecture, while the ThingSpeak platform is the Server. The building's layout can be summed up as follows:

### 6.1 Client

The ESP8266 is responsible for obtaining temperature information from the analog pin.

a WiFi connection is made with the chosen network. It transmits the temperature information by requesting an HTTP to the ThingSpeak server. The figure below can better demonstrate the understanding of the architecture:



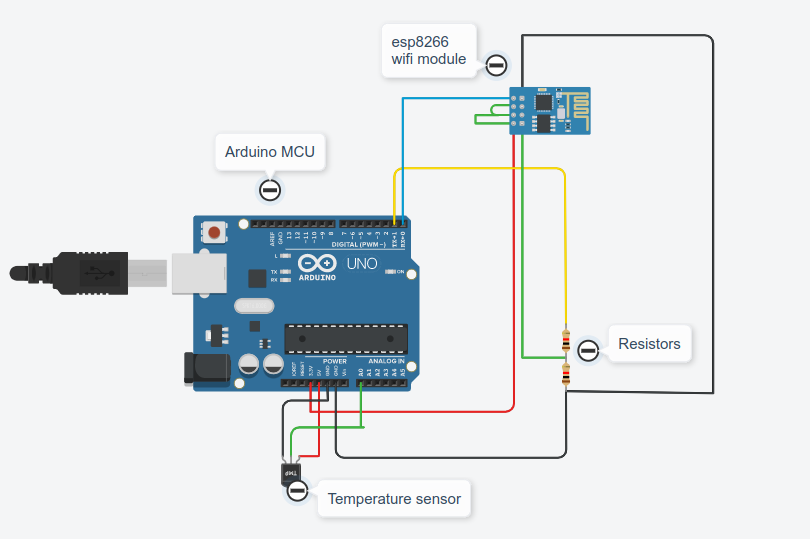
*Figure 1 Architectural Design*

### 6.2 Server (Platform for Thingspeak)

The Server receives the HTTP request with temperature data. It records the temperature information for future use in its database. It allows users to monitor and examine the gathered data using visualization tools and APIs. The ThingSpeak platform manages data storage, analysis, and visualization, while the microcontroller concentrates on data gathering and transmission (Antonova et al., 2021). This architecture allows for flexibility and scalability because more clients can be added to the system to monitor several temperature sensors simultaneously.

## **Section 7: System Diagram**

The diagram below shows the system circuit diagram:



*Figure 2 Circuit Diagram*

## **Section 8: Protocols**

The system uses the following protocols for communication:

WiFi (802.11): The system connects to the Internet and sends data to the ThingSpeak server using a WiFi connection established by the ESP8266 microcontroller. Wireless connectivity is guaranteed by the WiFi protocol, which also allows for seamless communication between the client and server components.

The system uses the HTTP (Hypertext Transfer Protocol) protocol to deliver temperature information to the ThingSpeak server. The temperature value, host information, and the API endpoint are all included in an HTTP request containing the temperature data. The ESP8266 microcontroller may be integrated with the ThingSpeak platform thanks to the HTTP protocol, making data transmission over the Internet more accessible. WiFi and HTTP protocols enable reliable and effective communication between system components. These protocols offer a standardized method for IoT-based data transfer and integration and are widely adopted.

## **Section 9: Programming Codes**

The system has been programmed in C programming, and the code is as shown below:

**#include <ESP8266WiFi.h> *// optional in the simulator***

**#include <wire.h> *// optional in the simulator***

**const** **char**\* wifiSSID = "Simulator Wifi"; *// Wi-Fi network SSID*

**const** **char**\* wifiPassword = ""; *// Wi-Fi network password*

**const** **char**\* host = "api.thingspeak.com"; *// ThingSpeak API server*

**const** **int** httpPort = 80; *// HTTP port for communication*

**const** **char**\* apiEndpoint = "/update?api\_key=RZVOOQ9M4FFAGTZA&amp;field1="; *// ThingSpeak API endpoint*

**int** **setupESP8266**() {

Serial.begin(115200); *// Starting the serial communication*

Serial.println("AT"); *// Sending AT command to ESP8266*

delay(10);

**if** (!Serial.find("OK")) **return** 1; *// Checking if "OK" response is received*

Serial.println("AT+CWJAP=\"" + String(wifiSSID) + "\",\"" + String(wifiPassword) + "\""); *// Connect to Wi-Fi network*

delay(10);

**if** (!Serial.find("OK")) **return** 2; *// Checking if "OK" response is received*

Serial.println("AT+CIPSTART=\"TCP\",\"" + String(host) + "\"," + String(httpPort)); *// Establish TCP connection with ThingSpeak server*

delay(50);

**if** (!Serial.find("OK")) **return** 3; *// Checking if "OK" response is received*

**return** 0; *// Return success*

}

**void** **sendDataToThingSpeak**() {

**int** temperature = map(analogRead(A0), 20, 358, -40, 125); *// Reading temperature from analog pin and mapping the values*

String httpRequest = "GET " + String(apiEndpoint) + String(temperature) + " HTTP/1.1\r\nHost: " + String(host) + "\r\n\r\n"; *// making an HTTP request*

**int** length = httpRequest.length();

Serial.print("AT+CIPSEND="); *// Sending the length of the HTTP request to ESP8266*

Serial.println(length);

delay(10);

**if** (!Serial.find(">")) **return**; *// Checking if ">" response is received*

Serial.print(httpRequest); *// Sending HTTP request to ThingSpeak server*

delay(10);

**if** (!Serial.find("SEND OK\r\n")) **return**; *// Checking if "SEND OK" response is received*

}

**void** **setup**() {

setupESP8266(); *// Initializing ESP8266*

}

**void** **loop**() {

sendDataToThingSpeak(); *// Sending data to ThingSpeak*

delay(10000); *// Delaying for 10 seconds before looping again*

}

### 9.1 Explanation of the Codes

The programming code was created using the Arduino framework, a popular platform for microcontroller programming. The code is broken up into various functions and follows a methodical approach. A summary of the code's principal sections is provided below:

The setupESP8266() function opens the program and initializes the ESP8266 microcontroller by establishing a serial connection and transmitting AT commands to it for configuration. It establishes a TCP connection with the ThingSpeak server and joins the microcontroller to the selected WiFi network. To obtain the temperature data from the analog pin, map it to the desired range, and create an HTTP request, use the sendDataToThingSpeak() function. The function transmits the temperature data to the ThingSpeak server via an HTTP request using AT instructions to store and display it.

The sendDataToThingSpeak() function is continuously called by the program's main loop at regular intervals of 10 seconds. This guarantees that temperature data is periodically transmitted to the ThingSpeak server for real-time monitoring. The code uses the Wire library for analog pin communication and the ESP8266 WiFi library for WiFi connectivity. It uses the ESP8266 microcontroller's capabilities to connect to networks, read analog data, and send HTTP requests. The code is organized to be clear and compelling, using the microcontroller's capabilities well.

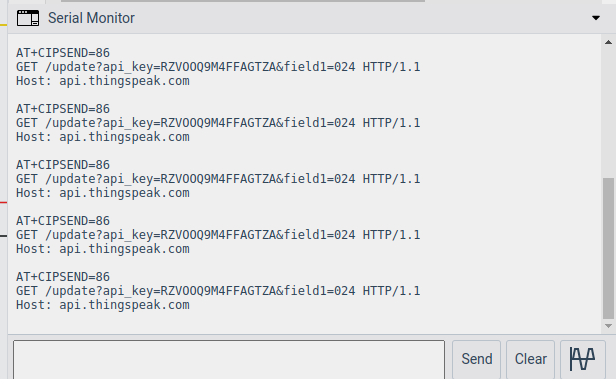
## **Section 10: Running of the Codes**

The development environment should be set up. Make sure the ESP8266 board is correctly incorporated into the Arduino IDE after installing the Arduino IDE. Install the necessary libraries, such as the Wire and ESP8266 WiFi libraries. Ensure the analog temperature sensor is powered and connected to the ESP8266 microcontroller. Then change the code by entering the SSID and password for the WiFi network. Update the ThingSpeak API endpoint and API key as necessary to reflect the details of the ThingSpeak account.

To upload the code to the ESP8266 microcontroller, connect the device to the computer, choose the Arduino IDE port, and click the upload button. AFTER THE CODE IS SUCCESSFULLY UPLOADED, the ESP8266 microcontroller will gather temperature data and transmit it to the ThingSpeak platform. By logging into the ThingSpeak account and checking the data visualization, we can keep track of the temperature data.

## **Section 11: The Operation of the System**

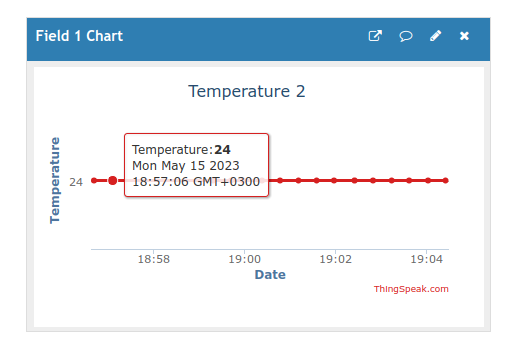
The ESP8266 microcontroller is initialized by setting up serial connectivity and defining the relevant parameters when it first powers on. It gains access to the Internet by establishing a connection with the authorized WiFi network. The microcontroller reads the temperature data via the analog pin attached to the temperature sensor. The appropriate temperature range is transferred to the analog value to ensure correct depiction. An HTTP request is subsequently made with the mapped temperature value. The microcontroller connects over TCP to the ThingSpeak server and sends the temperature-related HTTP request. The figure below shows a successful connection to the Internet and the Server:



*Figure 3 Serial Monitor Output*

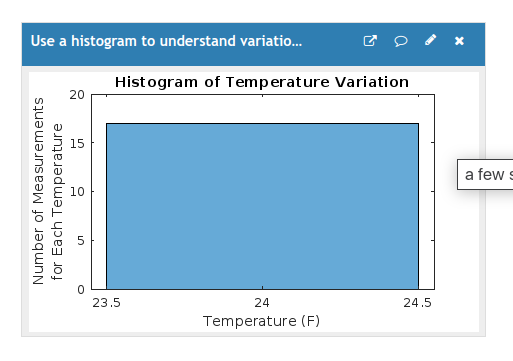
The HTTP server receives the ThingSpeak request, which extracts the temperature information. It offers visualization tools to track and examine the gathered data and maintains the temperature data in its database for future use. After every 10 seconds, the system will repeat the data transmission process. Guaranteeing a constant temperature data supply makes real-time monitoring and analysis possible.

Temperature data is consistently gathered and supplied for remote monitoring and analysis thanks to the continuous loop. The temperature sensor and the ThingSpeak platform are connected by the ESP8266 microcontroller, which also makes for easy data transmission over the Internet. The figures below demonstrate the temperature readings and data visualization of the ThingSpeak platform:



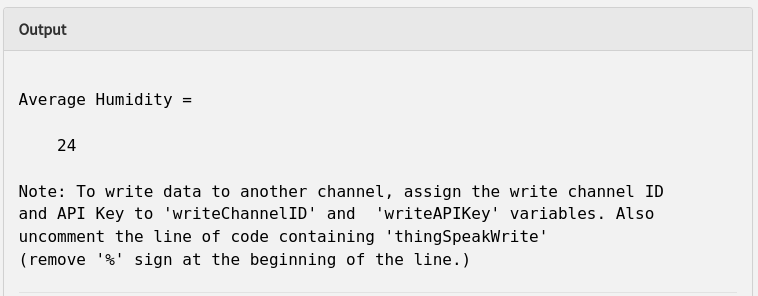
*Figure 4 ThingSpeak Temperature readings*

This temperature data can be further visualized as shown in the figure below:



*Figure 5 Temperature Data Visualization*

The collected data or temperature readings can also be analyzed into some meaningful information just as shown in the figure below:



*Figure 6 Temperature Data Analysis*

## **Section 12: Issues And Application Areas Of The System**

The system created using the provided has several advantages and finds use in various contexts. A few things to remember, though: The reliance on a solid WiFi connection could be problematic. The system's capacity to reliably communicate temperature data to the ThingSpeak server may be impacted by WiFi network outages or connectivity problems. To ensure uninterrupted use, setting up a reliable and stable internet connection is essential.

The system has uses in many different industries. Environmental monitoring is an application where the system can gather and monitor temperature data in various situations. In order to maintain ideal conditions for various activities, it can be used to monitor temperature differences in agricultural settings, warehouses, server rooms, and more. Home automation is another use for the system, where it can be set up as part of an intelligent home. Users can automate heating and cooling systems by monitoring the temperature, assuring energy efficiency and cozy living conditions. The management and optimization of energy use can benefit from the collected temperature data.

The system can also be used for research and development tasks that call for continuous temperature monitoring. It allows for the rapid and accurate collection of temperature data for analysis and decision-making by researchers (Astill et al., 2020).

## **Section 13: Conclusion**

The created system uses a microcontroller and the ThingSpeak platform to deliver a workable solution for tracking temperature data. It allows for the remote collection, transmission, and visualization of temperature data, enabling real-time observation and study. The computer code and instructions for running the system have all been thoroughly detailed, together with the system's functional requirements, components, and architectural design. The system's uses in environmental monitoring, home automation, and research further highlight its adaptability and potential influence across various industries.

## 

## **Section 14: References**

Nahid, S.I. and Khan, M.M., 2021, December. Toxic Gas Sensor and Temperature Monitoring in Industries using the Internet of Things (IoT). In 2021 24th International Conference on Computer and Information Technology (ICCIT) (pp. 1-6). IEEE.

Ngebani, I., Nfanyana, K., Sheikh, S.M., and Ishmael, Z., Design of a Microcontroller Based Internet of Things Weather Station.

Balakrishna, S. and Thirumaran, M., 2019. Programming paradigms for IoT applications: an exploratory study. In Handbook of IoT and Big Data (pp. 23-57). CRC Press.

Kalai Selvi, K., Gowthaman, N., Shirly Janet, A., Rithick, R., Rohith, P. and Sabariesh, S. (2022). Novel TMP36GT9Z-Based Industrial Temperature Measurement and Analysis for Industry 4.0. In Technology, Business, Innovation, and Entrepreneurship in Industry 4.0 (pp. 155-170). Cham: Springer International Publishing.

Aniebiet, I. and Fidelis, I.S., 2020. Design and Implementation of Gsm Enabled Remote Sensor for Monitoring Power Transformer Operation. American Journal of Electrical and Computer Engineering, 4(2), pp.62-71.

Antonova, V.M., Malikova, E.E., Panov, A.E., Spichek, I.V. and Malikov, A.Y., 2021. Implementation of IoT technology for data monitoring via cloud services. T-Comm-Телекоммуникации и Транспорт, 15(2), pp.46-53.

Astill, J., Dara, R.A., Fraser, E.D., Roberts, B. and Sharif, S., 2020. Smart poultry management: Smart sensors, big data, and the Internet of things. Computers and Electronics in Agriculture, 170, p.105291.