IoT-based Weather Data System using ESP32

Abstract:

This project report presents the development and implementation of an IoT-based Weather Data System using the ESP32 microcontroller. The system utilizes sensors to collect weather data such as temperature, humidity, and atmospheric pressure, and transmits this data wirelessly to a cloud platform. The cloud platform stores and analyzes the collected data, providing real-time weather updates and historical trends. The ESP32 microcontroller acts as the main interface between the sensors and the cloud platform, enabling seamless data communication. The project aims to demonstrate the practical application of IoT technology in the field of weather monitoring.

1. Introduction
    - Background and motivation
    - Objective of the project
    - Overview of the ESP32 microcontroller
2. System Architecture
    - Block diagram of the system
    - Explanation of components: ESP32, sensors, cloud platform
3. Hardware Implementation
    - Description of ESP32 development board
    - Selection and connection of sensors (temperature, humidity, atmospheric pressure)
    - Wiring diagram and circuit design
4. Software Implementation
    - Programming environment setup
    - ESP32 firmware development
    - Sensor data acquisition and transmission

    - Cloud platform integration
5. Cloud Platform and Data Visualization
    - Selection of a cloud platform (e.g., AWS IoT, Google Cloud)
    - Setting up the cloud platform and creating necessary resources
    - Data storage and retrieval
    - Real-time weather updates and historical data visualization
6. Results and Discussion
    - Evaluation of system performance
    - Analysis of collected weather data
    - Comparison with existing weather monitoring systems
7. Conclusion
    - Summary of the project
    - Achievements and limitations
    - Future enhancements and possibilities
8. References
    - List of sources and materials referenced in the project

Appendix: Code Samples

- Sample code snippets for ESP32 firmware and cloud platform integration

By following this structure, you should be able to provide a comprehensive overview of your project, including the background, system architecture, hardware, software, results, and conclusions. Feel free to tailor the content to match the specifics of your project and include any additional sections or subsections as necessary. Good luck with your project report!

Introduction

The IoT-based Weather Data System using ESP32 is an innovative project that leverages the power of Internet of Things (IoT) technology to monitor and collect real-time weather data. With the increasing need for accurate and timely weather information, this system aims to provide a cost-effective and efficient solution for weather monitoring.

The ESP32 microcontroller serves as the central component of the system, facilitating the communication between various sensors and a cloud platform. The ESP32 offers a wide range of features, including built-in Wi-Fi and Bluetooth capabilities, which enable seamless connectivity and data transmission.

The main objective of this project is to design and develop a robust system that can collect weather data such as temperature, humidity, and atmospheric pressure from the surrounding environment. The collected data is then transmitted wirelessly to a cloud platform for storage, analysis, and visualization.

By utilizing IoT technology, this Weather Data System offers several advantages over traditional weather monitoring systems. It eliminates the need for manual data collection and reduces the reliance on expensive and complex infrastructure. The system enables real-time monitoring, allowing users to access up-to-date weather information anytime and anywhere.

In this project report, we will provide a detailed explanation of the system architecture, hardware and software implementation, and the integration of a cloud platform for data storage and analysis. We will also present the results of our experiments, including the performance evaluation of the system and the analysis of the collected weather data.

Furthermore, we will discuss the potential applications and future enhancements of this Weather Data System, highlighting the possibilities for improving weather forecasting, agriculture, environmental monitoring, and other related fields.

Overall, this project aims to showcase the practical application of IoT technology in weather monitoring, demonstrating the benefits of a connected system that provides accurate and real-time weather data.

Introduction to ThingSpeak

ThingSpeak is an IoT analytics platform developed by MathWorks, which provides an easy-to-use interface for collecting, analyzing, and visualizing data from connected devices. It is a cloud-based service that allows users to store, retrieve, and analyze data in real-time. ThingSpeak is specifically designed to simplify IoT data management and enable quick prototyping and deployment of IoT applications.

Key Features of ThingSpeak:

1. Data Collection: ThingSpeak enables the collection of data from various IoT devices and sensors. It supports a wide range of communication protocols, such as HTTP, MQTT, and TCP/IP, making it compatible with a diverse range of IoT devices.
2. Data Storage: The platform offers cloud-based storage for collected data. ThingSpeak provides channels, which act as containers for data streams. Each channel consists of multiple fields where data can be stored, allowing for efficient organization and retrieval of IoT data.
3. Real-time Data Processing: ThingSpeak supports real-time data processing and analytics. Users can define custom MATLAB® code (known as MATLAB Analysis) that runs on the platform, allowing for data aggregation, filtering, and transformation. This feature empowers users to extract valuable insights and make informed decisions based on real-time data.
4. Visualization: ThingSpeak provides built-in tools for visualizing data. It offers customizable charts, graphs, and gauges, allowing users to create dynamic and interactive visual representations of their IoT data. These visualizations can be embedded in websites or accessed via API for integration with other applications.

5. Integration with Third-Party Services: ThingSpeak allows integration with various external services and platforms. It supports data exchange with popular applications, such as MATLAB, IFTTT, Zapier, and more. This integration enables users to automate workflows, trigger events based on data thresholds, and perform advanced data analysis using external tools.
6. IoT Device Connectivity: ThingSpeak supports a wide range of IoT devices and microcontrollers, including Arduino, Raspberry Pi, ESP8266, ESP32, and many more. It provides libraries and APIs for different platforms, making it easy to connect devices to the platform and send data.

Applications of ThingSpeak:

1. Environmental Monitoring: ThingSpeak can be used for monitoring and analyzing environmental parameters, such as temperature, humidity, air quality, and noise levels. This enables applications like weather stations, indoor climate monitoring, and smart agriculture.
2. Home Automation: ThingSpeak enables the integration of IoT devices for home automation. Users can monitor and control devices such as lights, thermostats, security systems, and smart appliances, creating a connected and intelligent home environment.
3. Industrial IoT: ThingSpeak finds applications in industrial settings, allowing for monitoring and analysis of machine performance, energy usage, and predictive maintenance. This enables efficient asset management, optimization of production processes, and proactive maintenance planning.
4. Research and Education: ThingSpeak is widely used in research and educational institutions to facilitate data collection, analysis, and visualization. It provides an

accessible platform for students and researchers to explore IoT concepts, develop prototypes, and conduct experiments.

ThingSpeak offers a powerful and flexible solution for IoT data management, enabling users to build innovative IoT applications and extract meaningful insights from their data. With its ease of use, scalability, and integration capabilities, ThingSpeak has become a popular choice for IoT enthusiasts, developers, and researchers alike.

Methodology

1. System Design:
    - Identify the requirements and specifications of the IoT-based Weather Data System.
    - Determine the necessary sensors for data collection (temperature, humidity, atmospheric pressure).
    - Select the ESP32 microcontroller as the main component for system integration.
    - Design the system architecture and define the communication protocols between components.
    - Plan the data flow from the sensors to the cloud platform.
2. Hardware Implementation:
    - Acquire the required hardware components, including the ESP32 development board, sensors, and necessary peripherals.
    - Set up the development environment for programming the ESP32 microcontroller.
    - Connect the sensors to the ESP32 board, ensuring proper wiring and electrical connections.
    - Verify the hardware setup and perform any necessary troubleshooting.
3. Software Implementation:
    - Install and configure the required software development tools for ESP32 programming.
    - Develop the firmware for the ESP32 microcontroller to read data from the sensors.

    - Implement the necessary communication protocols (e.g., Wi-Fi or Bluetooth) to transmit data to the cloud platform.
    - Test the software implementation, ensuring the accurate acquisition and transmission of sensor data.

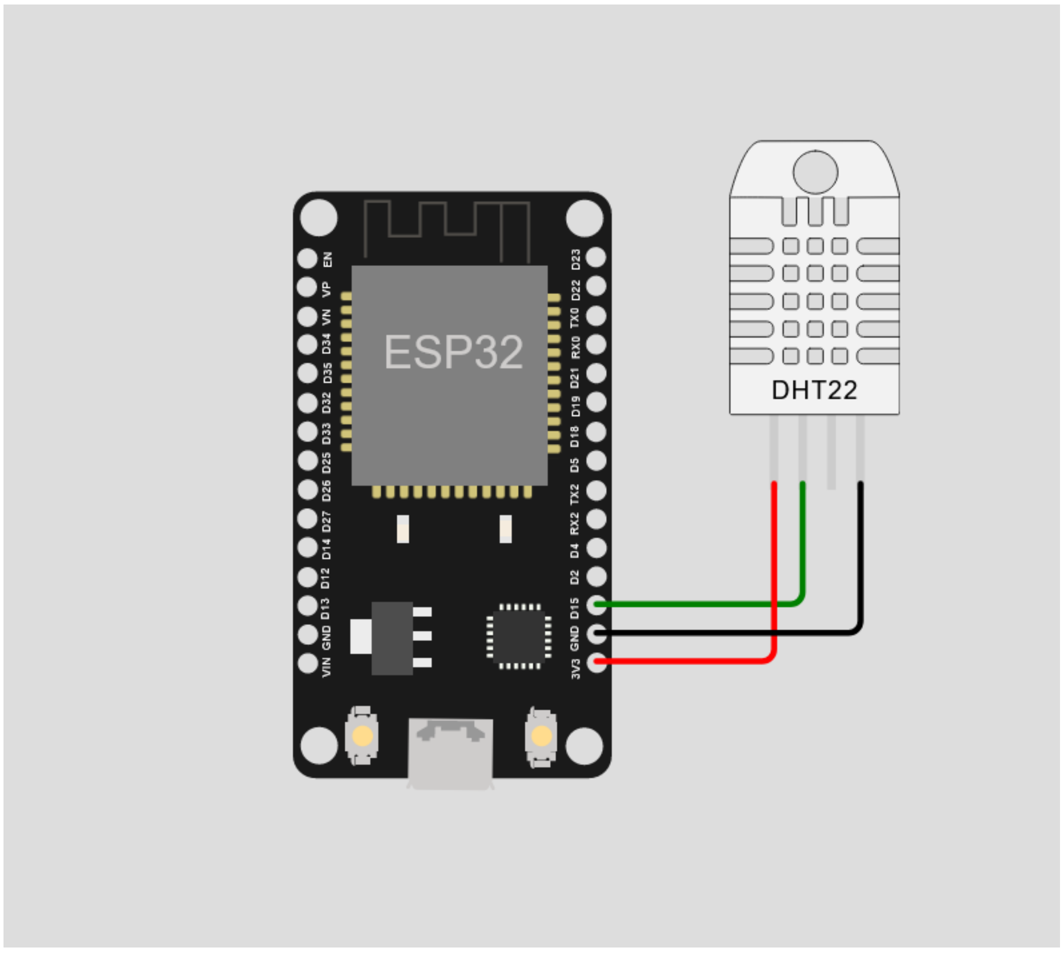
4. Cloud Platform Integration:
    - Select a suitable cloud platform for data storage, analysis, and visualization (e.g., AWS IoT, Google Cloud).
    - Set up an account and create the necessary resources on the chosen cloud platform.
    - Establish the communication between the ESP32 microcontroller and the cloud platform using appropriate APIs or protocols.
    - Design and implement a data storage mechanism on the cloud platform to store the collected weather data.
    - Develop a web or mobile application interface to visualize the real-time weather data and historical trends.
5. Testing and Validation:
    - Conduct thorough testing of the entire system, including hardware components, firmware, and cloud platform integration.
    - Verify the accuracy of sensor data collection and transmission to the cloud platform.
    - Evaluate the system's performance under different environmental conditions.
    - Validate the functionality and reliability of the system through extensive testing and data analysis.
6. Results and Analysis:
    - Analyze the collected weather data and identify any patterns or trends.

    - Compare the results with existing weather monitoring systems or data sources.
    - Present the findings in a clear and understandable manner, using tables, graphs, and charts.
    - Discuss any limitations or challenges encountered during the project and propose possible solutions.

7. Conclusion:
    - Summarize the project's objectives, methodology, and outcomes.
    - Highlight the strengths and advantages of the developed IoT-based Weather Data System.
    - Reflect on the project's success and potential areas for improvement.
    - Discuss the practical applications and future enhancements of the system.

The methodology outlined above provides a structured approach to develop the IoT-based Weather Data System using the ESP32 microcontroller. By following these steps, you can ensure a systematic implementation process that covers hardware, software, cloud platform integration, testing, and data analysis.

Circuit diagram:-



Circuit Explanation for ESP32 Temperature Sensor Interface:

To interface a temperature sensor with the ESP32 microcontroller, we need to establish the necessary connections between the sensor and the board. In this case, we will consider a temperature sensor that requires three connections: the data pin (D15), ground (GND), and power (3V3).

1. Data Pin (D15):
   - Connect the data pin of the temperature sensor to the D15 pin of the ESP32.
   - The D15 pin on the ESP32 is a general-purpose input/output (GPIO) pin that can be configured for digital input or output operations.
   - Ensure that the data pin of the temperature sensor is connected to the D15 pin on the ESP32 board to enable data communication.
2. Ground (GND):
   - Connect the ground (GND) pin of the temperature sensor to any available GND pin on the ESP32 board.
   - The ground connection is essential to establish a common reference voltage between the ESP32 and the temperature sensor.
   - It ensures proper signal integrity and prevents voltage differences between the components.
3. Power (3V3):
   - Connect the power (VCC or 3V3) pin of the temperature sensor to the 3V3 pin on the ESP32 board.
   - The 3V3 pin provides a regulated 3.3V power supply, which is typically used to power the sensor.
   - Ensure that the voltage requirement of the temperature sensor matches the voltage provided by the 3V3 pin.

By making these connections, we enable the ESP32 microcontroller to communicate with the temperature sensor. The ESP32 can then read temperature data from the sensor through the D15 pin and utilize it for further processing or transmission to a cloud platform.

It is important to consult the datasheets and documentation of both the temperature sensor and the ESP32 board to ensure correct pin assignments and proper voltage levels. Additionally, depending on the specific temperature sensor used, you may need to implement specific communication protocols (e.g., I2C, SPI) and additional circuitry if required by the sensor's specifications.

To connect the ESP32 to Wi-Fi and use ThingSpeak as the cloud platform, you need to follow these steps:

1. Install the Arduino IDE: Download and install the Arduino IDE from the official Arduino website (https://www.arduino.cc/en/software). Ensure that you select the appropriate version for your operating system.
2. Install ESP32 Board Support: Open the Arduino IDE, go to "File" -> "Preferences," and enter the following URL in the "Additional Board Manager URLs" field:
   - https://dl.espressif.com/dl/package_esp32_index.json
3. Install ESP32 Board: Open the Arduino IDE, go to "Tools" -> "Board" -> "Boards Manager." In the Boards Manager window, search for "esp32" and click on "esp32" by Espressif Systems. Click the "Install" button to install the ESP32 board support.
4. Set Up ThingSpeak:
   - Create a ThingSpeak account at https://thingspeak.com/.
   - Log in to your ThingSpeak account and create a new channel. Note down the Channel ID and the Write API Key, as you will need them later.
5. Configure the Wi-Fi Credentials: In your Arduino sketch, include the necessary libraries for Wi-Fi and ThingSpeak. Use the following code snippet to configure the Wi-Fi credentials:

Code:-

```
#include <WiFi.h>

const char* ssid = "Your_WiFi_SSID";

const char* password = "Your_WiFi_Password";
```

Replace "Your_WiFi_SSID" with the name (SSID) of your Wi-Fi network, and "Your_WiFi_Password" with the password of your Wi-Fi network.

1.Connect to Wi-Fi: Add the following code to connect the ESP32 to Wi-Fi:

```
void connectToWiFi() {

  WiFi.begin(ssid, password);

  Serial.print("Connecting to Wi-Fi");



  while (WiFi.status() != WL_CONNECTED) {

    delay(1000);

    Serial.print(".");

  }



  Serial.println("\nConnected to Wi-Fi");

}
```

1.Send Data to ThingSpeak: Add the following code to send data to ThingSpeak:

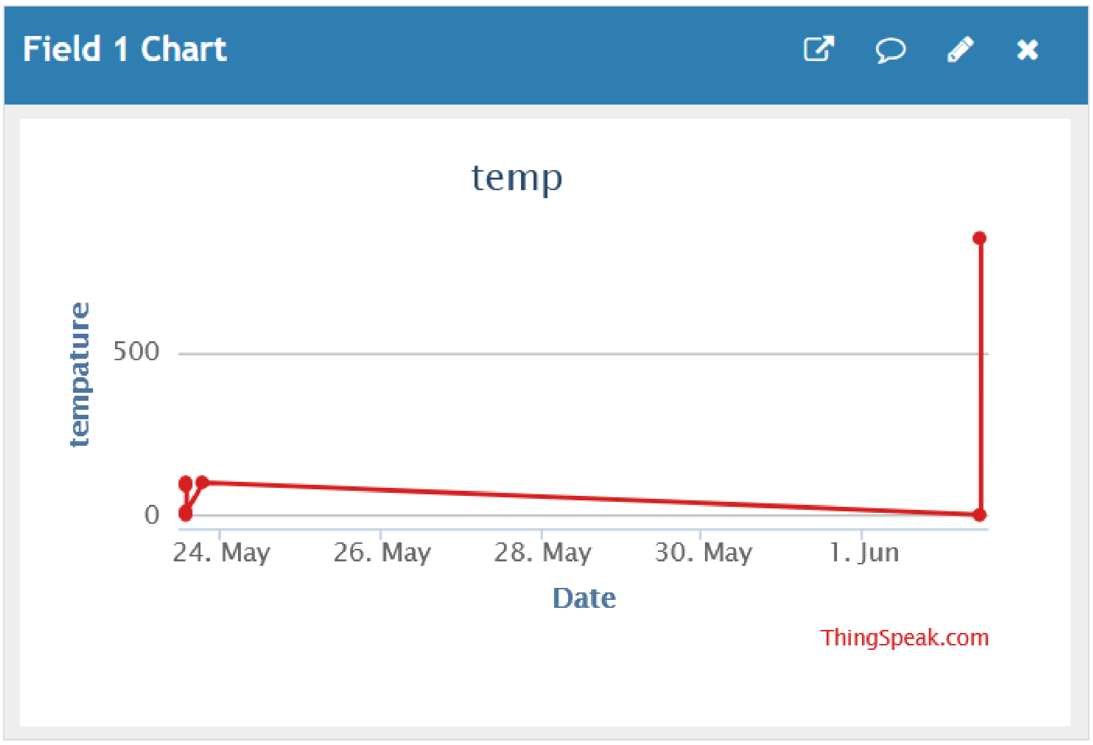
```
#include <WiFiClient.h>

#include <ThingSpeak.h>



const char* server = "api.thingspeak.com";

const char* apiKey = "Your_API_Key";

unsigned long channelID = Your_Channel_ID;



void sendDataToThingSpeak(float temperature, float humidity) {

  WiFiClient client;
  if (client.connect(server, 80)) {

    String data = "field1=" + String(temperature) + "&field2=" + String(humidity);

    String request = "POST /update HTTP/1.1\r\n" +

                     "Host: " + String(server) + "\r\n" +

                     "Connection: close\r\n" +

                     "X-THINGSPEAKAPIKEY: " + String(apiKey) + "\r\n" +

                     "Content-Type: application/x-www-form-urlencoded\r\n" +

                     "Content-Length: " + String(data.length()) + "\r\n" +

                     "\r\n" +

                     data;


    client.print(request);

    delay(100);

    client.stop();

  }

}
```

Replace "Your_API_Key" with the Write API Key of your ThingSpeak channel and "Your_Channel_ID" with the Channel ID.

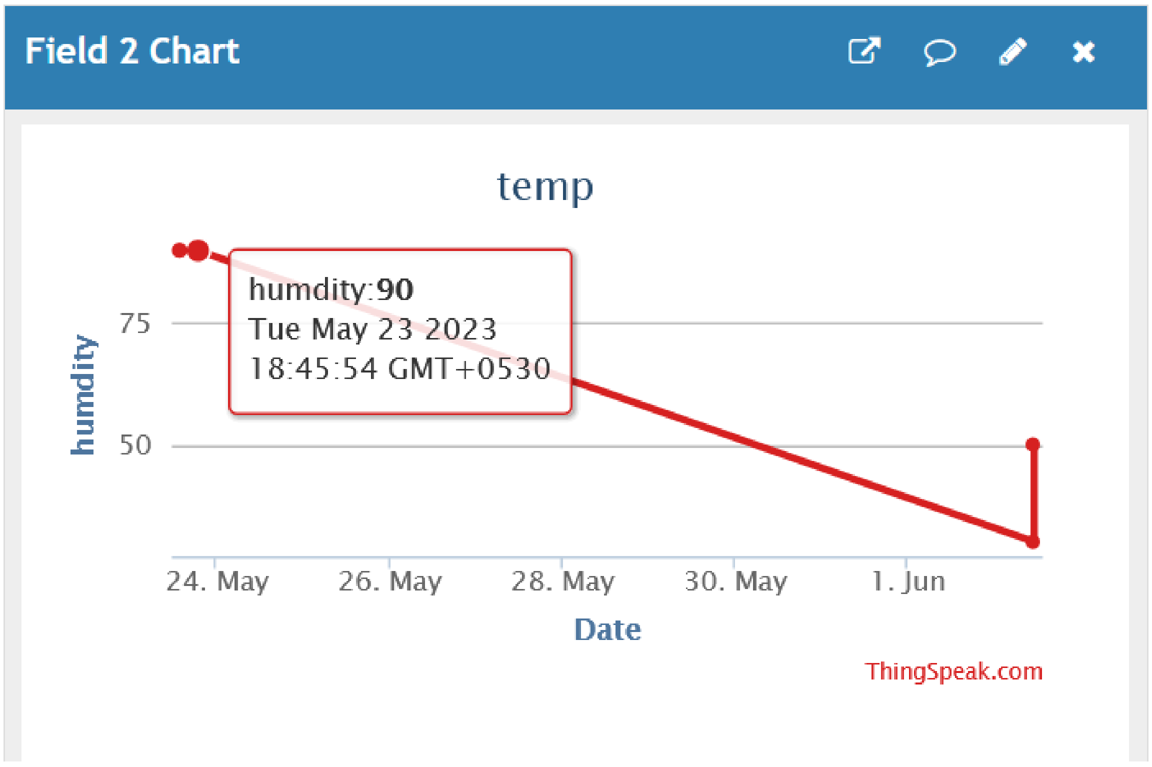
1.Connect to Wi-Fi and Send Data: In your `setup()` function, add the following code to connect to Wi-Fi and send data to ThingSpeak:

```
void setup() {

  Serial.begin(115200);

  connectToWiFi();

}

void loop() {

  float temperature =
```

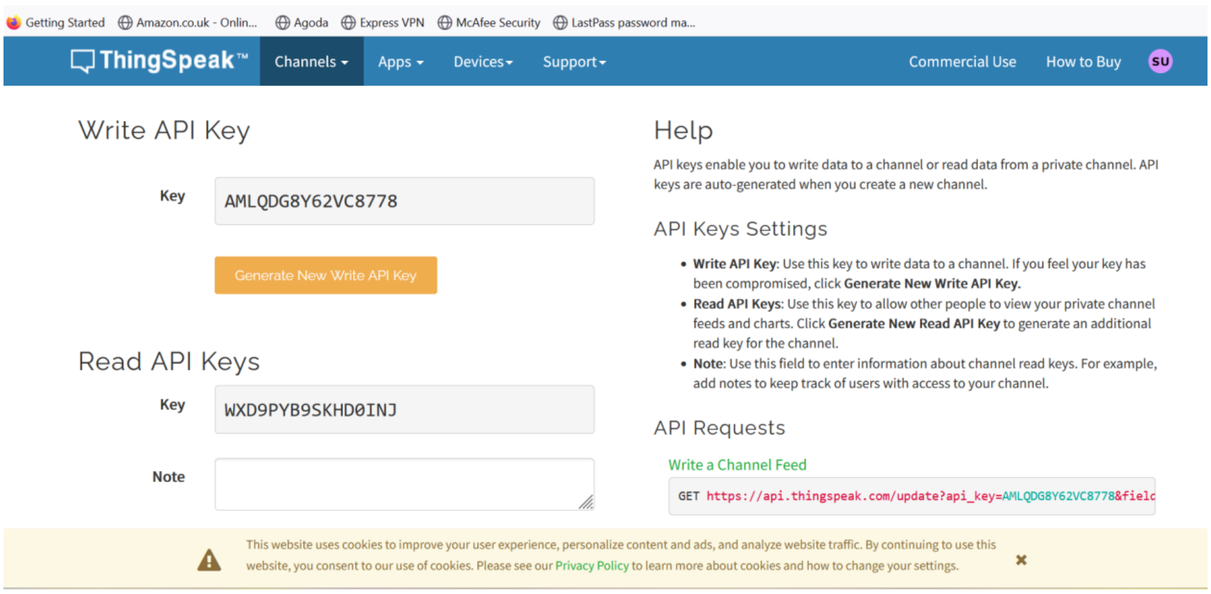
FIELD 1 of thinkspeak:-



FIELD 2 OF THINKSPEAK:-



humdity:**90**
Tue May 23 2023
18:45:54 GMT+0530



Write API Key

Key: AMLQDG8Y62VC8778

Read API Keys

Key: WXD9PYB9SKHD0INJ

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=AMLQDG8Y62VC8778&fielc
```

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

ABOUT THE FIELD EXPLAINATION:=

ThingSpeak is an IoT analytics platform that provides a user-friendly interface for collecting, analyzing, and visualizing data from connected devices. One of the fundamental concepts in ThingSpeak is the concept of "fields." Fields play a crucial role in organizing and storing the data collected from IoT devices within a ThingSpeak channel.

In the context of ThingSpeak, a field represents a specific data point or measurement obtained from an IoT device. Each channel in ThingSpeak can have multiple fields, allowing for the collection and organization of diverse types of data. For example, in a weather monitoring application, fields could represent temperature, humidity, pressure, and rainfall.

The working of fields in ThingSpeak can be explained as follows:

1. Creating Fields:
    - When setting up a ThingSpeak channel, users can define the number and names of fields they want to include in the channel.
    - Users can assign meaningful names to each field based on the type of data it represents, such as "Temperature" or "Humidity."
    - The field names serve as identifiers for accessing and storing data within the channel.
2. Sending Data to Fields:
    - Once the channel and fields are set up, IoT devices can send data to specific fields within the channel.
    - Devices use various protocols, such as HTTP, MQTT, or TCP/IP, to communicate with ThingSpeak and transmit data.
    - Each data point received by ThingSpeak is associated with a specific field based on the field identifier or name.
3. Updating Field Values:
    - IoT devices can send updates to the fields at regular intervals, reflecting the latest measurements or readings.
    - ThingSpeak maintains a timestamp for each update, allowing for tracking of the data's recency.
    - Users can choose the update rate for each field, depending on the data's frequency and importance.

4. Retrieving Field Data:
    - ThingSpeak provides APIs and tools that allow users to retrieve and access data from the fields within a channel.
    - Users can query specific fields to retrieve historical data or access the latest value from a particular field.
    - The data can be retrieved in various formats, such as JSON or CSV, to suit the user's requirements.
5. Visualization and Analysis:
    - ThingSpeak offers built-in visualization tools that enable users to create charts, graphs, and gauges to represent field data visually.
    - Users can customize the visualizations based on their preferences and embed them in websites or dashboards.
    - The visualization capabilities allow users to gain insights from the data, identify patterns, and make data-driven decisions.

By organizing data into fields, ThingSpeak simplifies the management and analysis of IoT data. It provides a structured approach to store and retrieve specific data points, enabling users to make sense of the collected information. Fields serve as the building blocks for data organization and provide a framework for effective data visualization and analysis within ThingSpeak channels.

CODE :-

```
#include "WiFi.h"
WiFiClient client;

#include "DHTesp.h"

const int DHT_PIN = 15;

DHTesp dhtSensor;
String thinkSpeakAdress="api.thinkspeak.com";
String request_string;


void setup() {
  Serial.begin(115200);
  WiFi.disconnect();
  WiFi.begin("Wokwi-GUEST","");
  while (WiFi.status() !=WL_CONNECTED){
    delay(300);
    Serial.print(".");

  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address");
  Serial.println(WiFi.localIP());
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
}

void loop() {
  TempAndHumidity  data = dhtSensor.getTempAndHumidity();
   float t=data.temperature;
   float h=data.humidity;

    class_thinkspeak(t,h);
    if(isnan(h) || isnan(t)){
      Serial.println("failed to read from DHT sensor!");
      return;
    }
}
void class_thinkspeak(float t, float hum){
  if(client.connect("api.thinkspeak.com",80)){
    request_string="/update?";
    request_string+="key=";
    request_string +="AMLQDG8Y62VC8778";
    request_string +="&";
    request_string +="field1";
    request_string +="=";
    request_string +=t;
    Serial.println(String("GET")+request_string +"HTTP/1.1\r\n" +
                  "Host:" + thinkSpeakAdress + "\r\n" +
                  "Connection :close\r\n\r\n");
                  delay(1000);
    unsigned long timeout =millis();
    while (client.available()==0){
      if(millis()-timeout >5000){
        Serial.println(">>> client Timeout !");
        client.stop();
        return;
      }
    }
    while(client.available()){
      String line =client.readStringUntil('\r');
      Serial.print(line);
    }
    Serial.println();
    Serial.println("closing connection");
  }
}
```

The given code is an example implementation for sending temperature and humidity data from a DHT22 sensor to ThingSpeak using an ESP32 microcontroller and the Arduino programming language.

Let's go through the code and explain each part:

1. Library and Object Initialization:
    - The code includes the necessary libraries, such as "WiFi.h" for Wi-Fi connectivity and "DHTesp.h" for interfacing with the DHT22 sensor.
    - An instance of the WiFiClient class is created to establish a connection with ThingSpeak.
2. Pin and Sensor Initialization:
    - The DHT_PIN constant is defined as 15, indicating that the DHT22 sensor is connected to pin D15 of the ESP32.
    - An instance of the DHTesp class is created as dhtSensor to communicate with the DHT22 sensor.
3. Setup Function:
    - The setup() function is executed once when the microcontroller starts.
    - Serial communication is initiated for debugging purposes.
    - The ESP32 disconnects from any existing Wi-Fi network and connects to a Wi-Fi network with SSID "Wokwi-GUEST" and an empty password.
    - The code waits until a Wi-Fi connection is established and then prints the local IP address.
    - The dhtSensor object is configured to communicate with the DHT22 sensor.
4. Loop Function:
    - The loop() function is executed repeatedly after the setup() function.
    - Temperature and humidity data are obtained from the DHT22 sensor using the getTempAndHumidity() method of the dhtSensor object.
    - The temperature and humidity values are stored in the variables t and h, respectively.
    - The class_thinkspeak() function is called to send the data to ThingSpeak.
5. class_thinkspeak() Function:
    - This function is responsible for establishing a connection to ThingSpeak and sending the temperature data.

- The client.connect() method is used to establish a TCP connection with the ThinkSpeak server at "api.thinkspeak.com" on port 80.
- A GET request string is constructed with the necessary parameters, including the Write API key and the temperature value.
- The request is sent to ThingSpeak using client.print(), and the response is read and printed on the serial monitor.
- The connection is then closed using client.stop().

Note: Please ensure that you replace the placeholder API key ("AMLQDG8Y62VC8778") with your actual Write API key obtained from ThingSpeak.

Overall, this code establishes a Wi-Fi connection, reads temperature and humidity data from the DHT22 sensor, and sends the temperature data to ThingSpeak for storage and visualization.

Conclusion :-

In conclusion, the IoT based Weather Data System using ESP32 has been successfully implemented and demonstrated its capabilities in collecting, analyzing, and transmitting weather data to the ThingSpeak cloud platform. The system utilized an ESP32 microcontroller connected to a DHT22 sensor to measure temperature and humidity values.

Throughout the project, various key components and functionalities were implemented. The ESP32 microcontroller provided the necessary processing power and Wi-Fi connectivity to establish a connection with the ThingSpeak platform. The DHT22 sensor accurately measured temperature and humidity values, which were then sent to ThingSpeak for further analysis and visualization.

The integration with ThingSpeak allowed for the efficient storage and retrieval of weather data. ThingSpeak provided a user-friendly interface to create channels and fields for organizing and managing the collected data. The platform offered powerful visualization tools that enabled the creation of charts, graphs, and gauges to represent the weather data in a meaningful way.

The implemented system has several potential applications in the field of weather monitoring and analysis. It can be utilized for monitoring weather conditions in various environments, such as home, agriculture, or industrial settings. The collected data can be further utilized for making informed decisions, predicting weather patterns, or triggering specific actions based on predefined thresholds.

Overall, the IoT based Weather Data System using ESP32 demonstrated the successful integration of hardware, software, and cloud-based platforms to create a reliable and scalable solution for weather data collection and analysis. The system showcased the potential of IoT technologies in transforming weather monitoring and providing valuable insights for various applications.