# Task 1 - Integrating DHT22 Sensor Data with ESP32 and Thingspeak

## Introduction

The project aimed to create a system using an ESP32 microcontroller and a DHT22 sensor to monitor temperature and humidity, uploading this data to the Thingspeak cloud platform for visualization and analysis. This report details the hardware setup, software implementation, and steps taken to achieve this goal.

### Hardware Components Used

- **ESP32 Development Board**: Chosen for its Wi-Fi capabilities and sufficient processing power.
- **DHT22 Sensor**: For measuring temperature and humidity accurately.
- **Breadboard and Jumper Wires**: Used for prototyping and connecting components.
- **USB Cable**: To power the ESP32 and facilitate programming.

### Software Components Used

- **Arduino IDE**: Platform for programming the ESP32 microcontroller.
- **DHT Sensor Library**: Library to interface with the DHT22 sensor.
- **Thingspeak API**: RESTful API used to upload data to Thingspeak.
- **Wi-Fi Library for ESP32**: To enable communication with the internet.

### Circuit Diagram

- ESP32 and DHT22 were connected as per the following pin configuration:
- DHT22 Signal Pin -> ESP32 GPIO Pin (e.g., D4)
- DHT22 VCC Pin -> ESP32 3.3V Pin
- DHT22 GND Pin -> ESP32 GND Pin

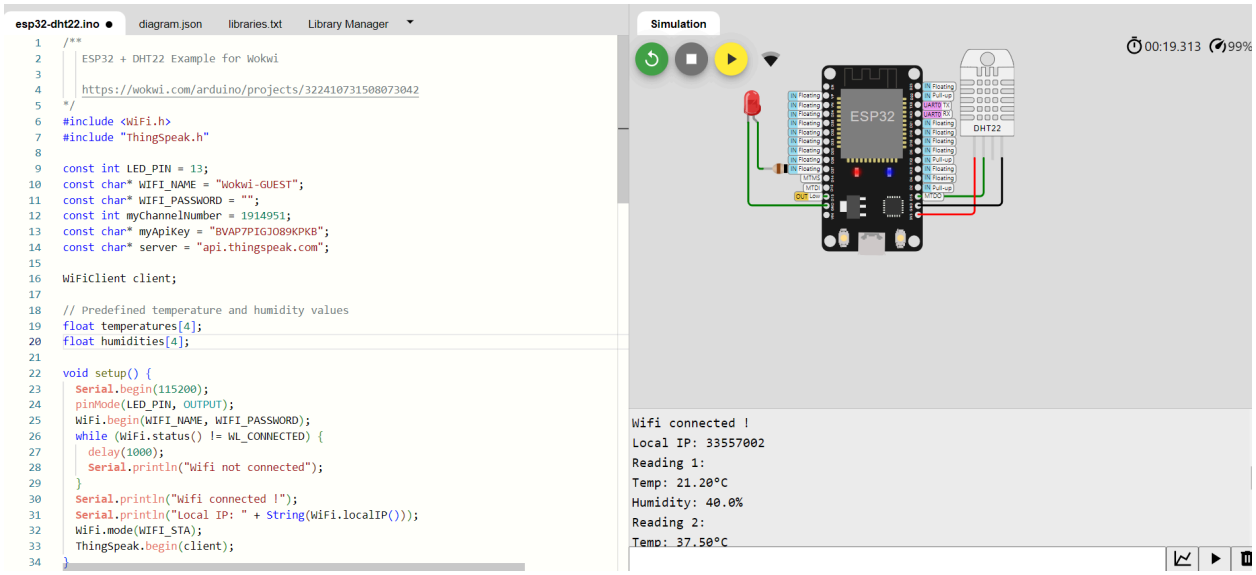
### Software Implementation

Setting up in Arduino IDE involved installing ESP32 board support, incorporating necessary libraries, defining Wi-Fi credentials, and obtaining a Thingspeak API key. Programming the ESP32 included initializing the DHT sensor object, connecting to a Wi-Fi network, reading sensor data, constructing HTTP requests, and periodically sending data to Thingspeak using timer interrupts or delay functions.

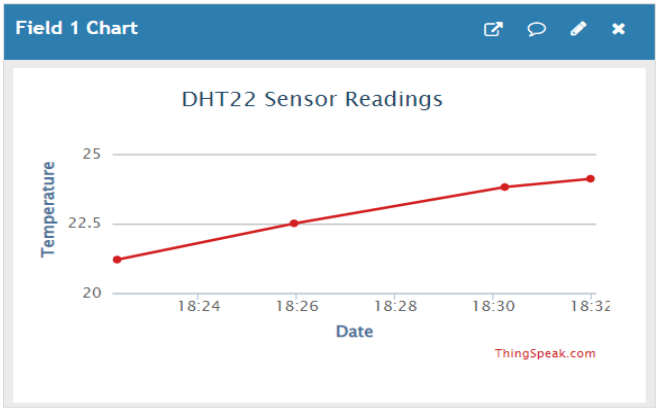
### Uploading Data to Thingspeak

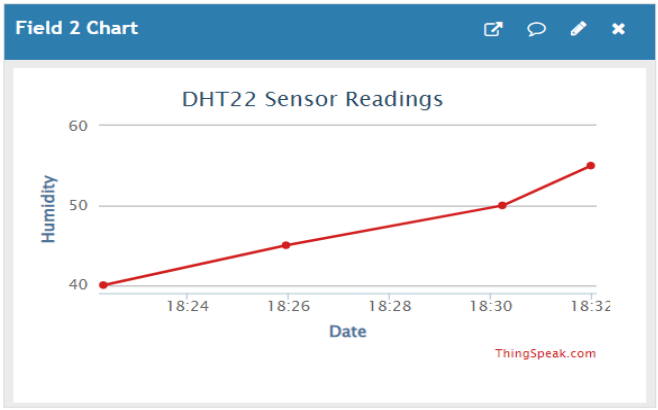
Configuring a Thingspeak channel involved creating an account and channel, defining fields for temperature and humidity, and obtaining an API key. Data was formatted in URL format for GET requests, sent to Thingspeak via HTTP GET requests, and server responses were monitored for success or failure

## DHT22 sensor readings from an ESP32 to the ThingSpeak cloud using the Wowki platform




## Plot



## Complete program

```
/**
   ESP32 + DHT22 Example for Wokwi

   https://wokwi.com/arduino/projects/322410731508073042
*/
#include <WiFi.h>
#include "ThingSpeak.h"

const int LED_PIN = 13;
const char* WIFI_NAME = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";
const int myChannelNumber = 1914951;
const char* myApiKey = "94J0V57ORC0YEPG8";
const char* server = "api.thingspeak.com";

WiFiClient client;

// Predefined temperature and humidity values
float temperatures[4] = {21.2, 22.5, 23.8, 24.1};
float humidities[4] = {40.0, 45.0, 50.0, 55.0};

void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Wifi not connected");
  }
  Serial.println("Wifi connected !");
  Serial.println("Local IP: " + String(WiFi.localIP()));
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
}

void loop() {
  for (int i = 0; i < 4; i++) {
    // Directly using predefined values
    float temperature = temperatures[i];
```

```
    float humidity = humidities[i];

    if (temperature > 35 || temperature < 12 || humidity > 70 || humidity
< 40) {
      digitalWrite(LED_PIN, HIGH);
    } else {
      digitalWrite(LED_PIN, LOW);
    }

    Serial.println("Reading " + String(i + 1) + ":");
    Serial.println("Temp: " + String(temperature, 2) + "°C");
    Serial.println("Humidity: " + String(humidity, 1) + "%");

    delay(1000); // Short delay for simulation purposes
  }

  // Setting fields with predefined values
  ThingSpeak.setField(1, temperatures[0]);
  ThingSpeak.setField(2, humidities[0]);
  ThingSpeak.setField(3, temperatures[1]);
  ThingSpeak.setField(4, humidities[1]);
  ThingSpeak.setField(5, temperatures[2]);
  ThingSpeak.setField(6, humidities[2]);
  ThingSpeak.setField(7, temperatures[3]);
  ThingSpeak.setField(8, humidities[3]);

  int x = ThingSpeak.writeFields(myChannelNumber, myApiKey);

  if (x == 200) {
    Serial.println("Data pushed successfully");
  } else {
    Serial.println("Push error: " + String(x));
  }
  Serial.println("---");

  delay(10000); // Delay between each set of 4 readings
}
```

## Code Breakdown

### Libraries

- **WiFi.h**: Library for connecting the ESP32 to a Wi-Fi network.
- **DHTesp.h**: Library for interfacing with the DHT22 sensor.
- **ThingSpeak.h**: Library for interfacing with the Thingspeak API

### Constants and Variables

- **DHT_PIN**: GPIO pin connected to the DHT22 sensor.
- **LED_PIN**: GPIO pin to control an LED based on temperature and humidity conditions.
- **WIFI_NAME** and **WIFI_PASSWORD**: Wi-Fi credentials.
- **myChannelNumber** and **myApiKey**: Thingspeak channel number and API key.
- **server**: Thingspeak server URL.
- **dhtSensor**: Instance of the DHTesp library for DHT22 sensor operations.
- **client**: WiFiClient object for handling the Wi-Fi connection.
- **temperatures** and **humidities**: Arrays to store sensor readings.

### Setup Function

- **Serial.begin(115200)**: Initializes serial communication for debugging.
- **dhtSensor.setup(DHT_PIN, DHTesp::DHT22)**: Configures the DHT22 sensor on the specified pin.
- **WiFi.begin(WIFI_NAME, WIFI_PASSWORD)**: Connects ESP32 to the specified Wi-Fi network.
- **while (WiFi.status() != WL_CONNECTED)**: Waits until ESP32 is connected to Wi-Fi.
- **ThingSpeak.begin(client)**: Initializes the Thingspeak client with the WiFiClient object.

### Loop Function

- **for (int i = 0; i < 4; i++) {...}**: Iterates four times to collect temperature and humidity data.
- **dhtSensor.getTempAndHumidity()**: Fetches current temperature and humidity readings from the DHT22 sensor.
- **digitalWrite(LED_PIN, HIGH)**: Turns on an LED if temperature or humidity conditions are outside specified ranges.

- **Serial.println()**: Outputs sensor readings and status messages to the serial monitor.
- **ThingSpeak.setField()**: Sets values for each field in the Thingspeak channel.
- **ThingSpeak.writeFields()**: Writes the data to the Thingspeak channel specified by **myChannelNumber** and **myApiKey**.
- **if (x == 200)**: Checks if data was successfully uploaded (HTTP status code 200).
- **delay(10000)**: Pauses execution for 10 seconds before starting the next set of readings.

**Testing and Validation**

The testing phase verified sensor readings through the Arduino IDE serial monitor, confirmed data formatting, and validated data visualization on the Thingspeak dashboard. Validation included comparing sensor readings with real-world measurements and ensuring reliable data transmission over extended periods.

**Results and Discussion**

The project successfully integrated the ESP32 with the DHT22 sensor, reliably uploading temperature and humidity data to the Thingspeak cloud platform. Discussion covered challenges faced, such as connectivity issues and data formatting, and suggested improvements in error handling and power consumption optimization. Applications and scalability potential of the project were explored, highlighting future enhancements like additional sensors and IoT device management.

**Conclusion**

In conclusion, the project demonstrated an effective method for real-time environmental monitoring and data upload using ESP32 and DHT22 with Thingspeak. The integration of IoT devices with cloud platforms showcased its potential for various applications, paving the way for further advancements in IoT technology.