

TASK 2

Major IoT Project using ESP32 along with DHT22 (temperature and humidity sensor) and Photoresistor (LDR) using WOKWI, Thingspeak, and Zapier

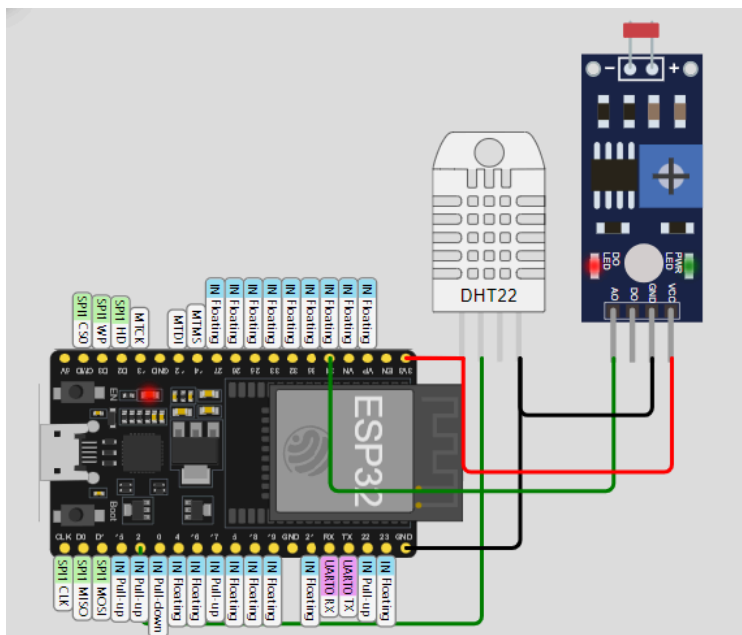
Introduction

The goal of this project was to develop a Smart Environment Monitoring System utilizing the ESP32 microcontroller, DHT22 sensor (for temperature and humidity), and a photoresistor (LDR) to monitor environmental conditions. The project also involved sending data to the ThingSpeak cloud platform and setting up automated alerts using Zapier. WOKWI was used for simulating the project components.

Components Used

- **ESP32 Microcontroller:** Serves as the main controller for the project.
- **DHT22 Sensor:** Measures temperature and humidity levels.
- **Photoresistor (LDR):** Measures light intensity.
- **WOKWI:** Online simulator for testing the circuit and code.
- **ThingSpeak:** Cloud platform for storing and visualizing sensor data.
- **Zapier:** Service for setting up automated alerts based on sensor data.

Circuit diagram



Overview of the Program

The program is designed to read environmental data (temperature, humidity, and light levels) using an ESP32 microcontroller, a DHT22 sensor for temperature and humidity, and a Photoresistor (LDR) for light intensity. The data is then sent to ThingSpeak, a platform for data logging and visualization.

Program

```
#include <WiFi.h>
#include <DHT.h>
#include <HTTPClient.h>
// Replace with your network credentials
const char* WIFI_NAME = "Wokwi-GUEST";
const char* WIFI_PASSWORD = "";
// Replace with your ThingSpeak API key and Channel ID
const char* server = "http://api.thingspeak.com/update";
const char* apiKey = "8LZ46D2443YINRC5";
// Replace with your Zapier Webhook URL
const char* zapierWebhookUrl =
"https://hooks.zapier.com/hooks/catch/19441308/23v0umo/";
// Pin configuration
const int DHTPin = 15;
const int LDRPin = 34;
// DHT sensor type
#define DHTTYPE DHT22
DHT dht(DHTPin, DHTTYPE);
// Arrays to store constant values for temperature, humidity, and LDR
const float temperatures[]
const float humidities[]
const int ldrValues[]
int arrayIndex = 0; // Index to iterate through the arrays
void setup() {
    Serial.begin(115200);
    WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }
    Serial.println("Connected to WiFi");

    dht.begin();
}
```

```

void loop() {
    // Read constant values from the arrays
    float temperature = temperatures[arrayIndex];
    float humidity = humidities[arrayIndex];
    int ldrValue = ldrValues[arrayIndex];

    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.print("%, Temperature: ");
    Serial.print(temperature);
    Serial.print("C, LDR Value: ");
    Serial.println(ldrValue);

    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;

        // Send data to ThingSpeak
        String url = String(server) + "?api_key=" + apiKey + "&field1=" +
String(humidity) + "&field2=" + String(temperature) + "&field3=" +
String(ldrValue);
        http.begin(url.c_str());
        int httpStatusCode = http.GET();
        if (httpStatusCode > 0) {
            Serial.print("HTTP Response code: ");
            Serial.println(httpStatusCode);
            Serial.println("Data pushed to ThingSpeak successfully");
        } else {
            Serial.print("Error code: ");
            Serial.println(httpStatusCode);
            Serial.println("Failed to push data to ThingSpeak");
        }
        http.end();
    } else {
        Serial.println("WiFi Disconnected");
    }

    if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;

        String url = String(zapierWebhookUrl) + "?humidity=" + String(humidity) +
"&temperature=" + String(temperature) + "&ldrValue=" + String(ldrValue);
        http.begin(url.c_str());
        int httpStatusCode = http.GET();
        if (httpStatusCode > 0) {
            Serial.println("Data sent to Zapier successfully");
        } else {
            Serial.println("Failed to send data to Zapier");
        }
    }
}

```

```

    http.end();
} else {
    Serial.println("WiFi Disconnected");
}

delay(20000); // Upload every 20 seconds
arrayIndex = (arrayIndex + 1) % 4; // Move to the next set of values
delay(20000); // Upload every 20 seconds
}

```

Code Explanation

The Arduino sketch includes libraries for WiFi, DHT sensor, and HTTP client functionality. It establishes a WiFi connection and periodically reads sensor values. The data is then formatted into HTTP GET requests for both ThingSpeak and Zapier. This dual functionality allows for real-time data visualization and automated alerts.

Code Breakdown

- **Libraries Included:**
 - `WiFi.h`: Allows the ESP32 to connect to Wi-Fi networks.
 - `DHT.h`: Provides functions to interface with the DHT22 sensor.
 - `HttpClient.h`: Used to send HTTP requests.
- **Wi-Fi Credentials:**
 - `WIFI_NAME`: The SSID of the Wi-Fi network.
 - `WIFI_PASSWORD`: The password for the Wi-Fi network (empty in this case, as Wokwi's guest network doesn't require one).
- **API Configuration:**
 - `server`: The ThingSpeak endpoint for data updates.
 - `apiKey`: Your unique ThingSpeak Write API Key.
 - `zapierWebhookUrl`: The Zapier webhook URL for sending data.
- **Pin Configuration:**
 - `DHTPin`: GPIO pin connected to the DHT22 sensor.
 - `LDRPin`: GPIO pin connected to the LDR.
- **DHT Sensor Type Definition:**
 - `DHTTYPE`: Specifies the type of DHT sensor (DHT22).
 - `dht`: Creates an instance of the DHT class for the specified pin
- **Setup Function:**
 - Initializes serial communication for debugging.
 - Connects to the Wi-Fi network and waits until connected.
 - Initializes the DHT22 sensor

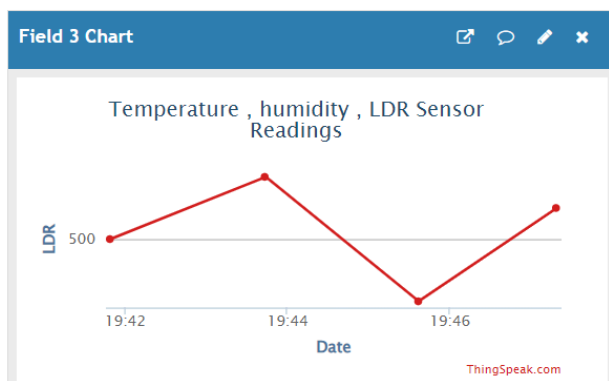
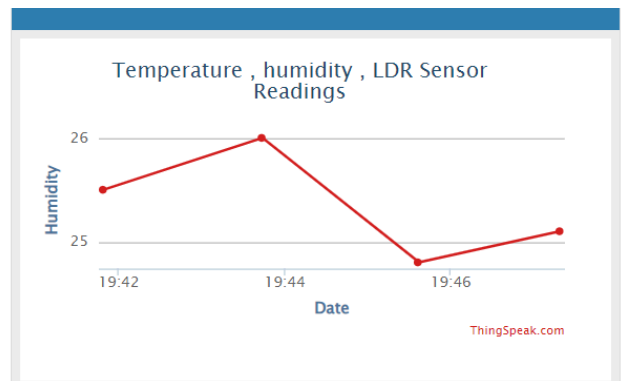
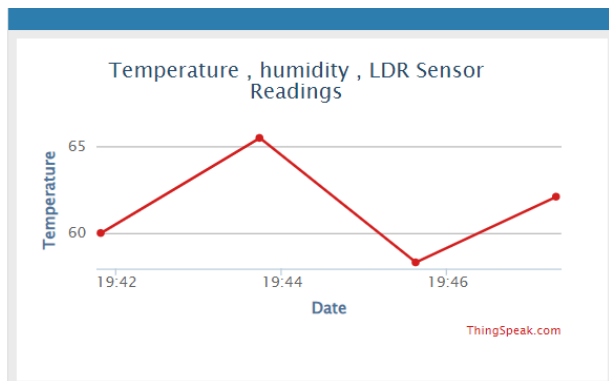
- **Sending Data to ThingSpeak:**
 - Checks for a Wi-Fi connection.
 - Constructs the URL with sensor data for ThingSpeak.
 - Sends a GET request to update the data.
 - Prints response codes for debugging
- **Sending Data to Zapier:**
 - Similar structure as the ThingSpeak request, but it sends data to the Zapier webhook.
 - Constructs the URL with the same sensor values.
 - Handles the response for success or failure

Data Visualization in ThingSpeak

After uploading the data:

- The "Visualize" tab in ThingSpeak was used to create line graphs for humidity, temperature, and LDR values.
- Graphs automatically update as new data is received, providing an effective way to monitor environmental changes over time.

Thingspeak output



Zapier Integration

- A Zapier account was set up, and a new Zap was created with a Webhook trigger to receive data from the ESP32.
- Email notifications were configured to alert users when specific conditions, such as high temperature, are met. This feature enhances the project by ensuring timely awareness of environmental changes.

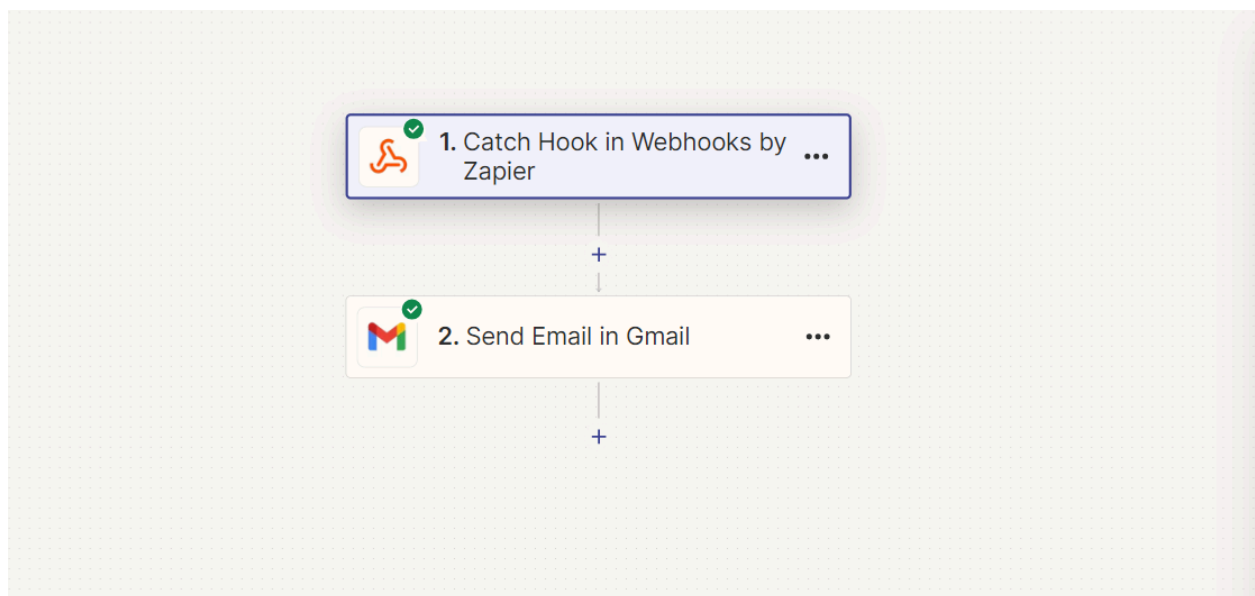
Setting Up a New Zap: In the Zapier dashboard, a new "Zap" was created. The first step involved selecting "Webhooks by Zapier" as the trigger application. The specific trigger event chosen was "Catch Hook," which allows Zapier to receive data sent from the ESP32.

Webhook URL Configuration: Upon setting up the trigger, Zapier generated a unique Webhook URL. This URL was incorporated into the Arduino code, specifically in the section that sends sensor data to Zapier. The Arduino code was updated to ensure that humidity, temperature, and LDR values are transmitted correctly.




Testing and Activation: A test was conducted to ensure that when the ESP32 sends data, an email notification is received as expected. The data populated correctly in the email, confirming successful integration. Once verified, the Zap was activated, allowing for real-time email alerts based on environmental conditions.


Outputs from zapier


Overview





Data received from ESP32

 1. Catch Hook in Webhooks by Zapier  

App & event 

Trigger 

Test 

 We found records in your Webhooks by Zapier account. We will load up to 3 most recent records, that have not appeared previously.

[Learn more about test records](#)

☒ request B
original record pulled on Jul 12, 2024

querystring


Querystring Humidity 60.00

Querystring Temperature 25.50




Querystring Ldr Value 500


☐ request A
original record pulled on Jul 12, 2024


Continue with selected record





Sending email alert



 2. Send Email in Gmail  

App & event 

Account 


Action 

Test 

 → 

Send Email to Gmail

We'll use this as a sample for setting up the rest of your Zap.

 A Send Email was sent to Gmail about 2 minutes ago

Data in

Data out

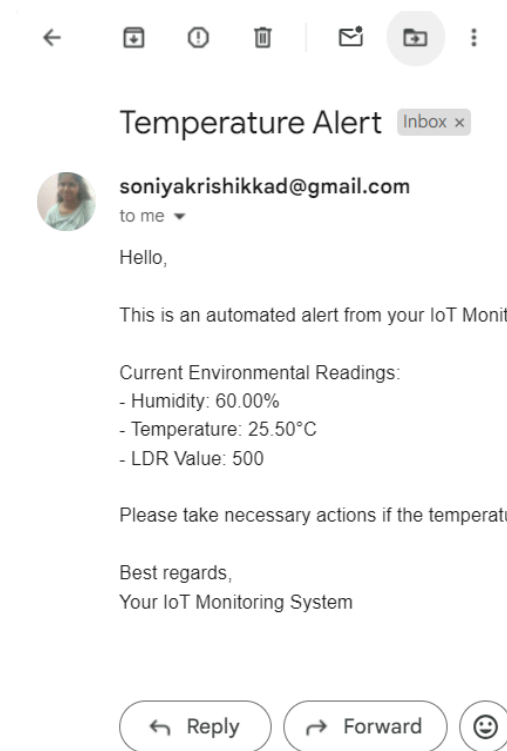
ID 190a72fd49989b07

Thread Id 190a72fd49989b07

Label Ids

1 SENT

Received Email Alert



Links

WOWKI - <https://wokwi.com/projects/403215109719309313>

THINGSPEAK - <https://thingspeak.com/channels/2598475>

ZAPIER - <https://zapier.com/shared/9ab4c625feb3bb567aff0681958d88e3297192fe>

Conclusion

This IoT project successfully demonstrates the integration of hardware and cloud services to create an efficient environmental monitoring system. Future improvements could include integrating additional sensors or expanding notification capabilities. Overall, the project exemplifies the potential of IoT technologies in real-world applications.