**Station 1:-**

In developing the IoT system, the following steps were followed to collect data from the Wokwi platform and transmit it to ThingSpeak:

Link:- https://wokwi.com/projects/393386061810673665

**1. Integration with Wokwi:**

- The virtual station data was collected from Wokwi, an online platform for simulating Arduino projects.

- Within the Wokwi environment, sensors for temperature and humidity were interfaced with the DHT22 sensor, while for the CO2 sensor, a random number generator was implemented.

- The virtual station was set up to periodically generate sensor readings.

**2. Arduino Code Development:**

- Arduino code was developed to interface with the sensors and Wokwi platform.

- Libraries such as `WiFi.h`, `DHTesp.h`, and `ThingSpeak.h` were included in the code to enable Wi-Fi connectivity, DHT22 sensor functionality, and communication with the ThingSpeak IoT platform, respectively.

- Constants were defined for the DHT22 sensor GPIO pin, Wi-Fi credentials, ThingSpeak channel number, API key, and server address.

- A function `generateCO2Value()` was implemented to simulate CO2 values within the specified range using a random number generator.

- The `setup()` function initialized serial communication, configured the DHT22 sensor, connected to the Wi-Fi network, and initialized the ThingSpeak library.

- The `loop()` function continuously read temperature and humidity data from the DHT22 sensor, generated a random CO2 value, and transmitted this data to ThingSpeak.

- Sensor data (temperature, humidity, and CO2) was set as fields in the ThingSpeak channel using the `ThingSpeak.setField()` function.

- The data was then written to the ThingSpeak channel using the `ThingSpeak.writeFields()` function, and the status of the data transmission was checked.

- Serial print statements were used to display the sensor readings and the status of the data transmission.

**3. Data Transmission to ThingSpeak:**

- Within the `loop()` function, sensor data (temperature, humidity, and CO2) was collected and transmitted to the specified ThingSpeak channel.

- Each sensor reading was assigned to a specific field in the ThingSpeak channel to ensure accurate data storage.

- The status of the data transmission was monitored, and appropriate messages were printed to the serial monitor.

**4. Data for Station 1:**

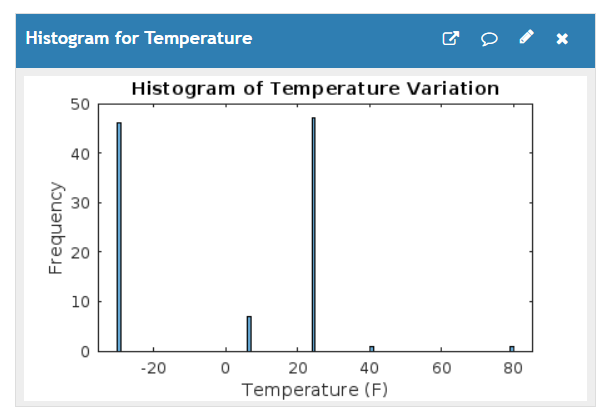
The data transmitted to ThingSpeak for Station 1 included:

1. Temperature

A graph with red lines

Description automatically generated

Visualization:



2. Humidity

A screen shot of a graph

Description automatically generated

Visualization:

A screenshot of a graph

Description automatically generated

3. CO2 concentration

A graph with red dots

Description automatically generated

Visualization:

A graph of co2 variation

Description automatically generated

In summary, the development process involved integrating virtual sensors from Wokwi, implementing Arduino code to collect sensor data, and transmitting this data to ThingSpeak for storage and visualization. The code ensured periodic data collection and transmission, providing real-time monitoring of environmental parameters for Station 1.

**Station 2:-**

This Python code is designed to simulate the generation of random sensor readings for temperature, humidity, and CO2 levels. The generated data is then published to a ThingSpeak channel using MQTT (Message Queuing Telemetry Transport) protocol. It's worth noting that this designed code was executed in a Google Colab environment, leveraging its computational resources and collaborative features for IoT development.

Let's break down the code:

**Setup:**

The code imports necessary libraries, including paho.mqtt.client for MQTT communication, random for generating random numbers, and time for adding delays.

Variables such as station\_id, channel\_id, write\_api\_key, and MQTT settings (mqtt\_host, mqtt\_user, mqtt\_password, mqtt\_topic) are defined. These parameters are used for connecting to the ThingSpeak channel and publishing data.

**Functions:**

generate\_sensor\_data(): This function generates random sensor readings within specified ranges for temperature, humidity, and CO2 levels.

on\_connect(): Callback function for MQTT client connection. It prints a message indicating successful or failed connection to the MQTT broker.

on\_publish(): Callback function called when a message is published. It prints the message ID of the published message.

publish\_sensor\_data(): This function constructs the payload with sensor data and publishes it to the ThingSpeak channel using MQTT.

**MQTT Client Setup:**

An MQTT client instance is created with callback API version VERSION1.

MQTT client username and password are set using username\_pw\_set() method.

Callback functions on\_connect and on\_publish are assigned to the MQTT client.

The client connects to the MQTT broker (mqtt\_host) on port 1883.

**Main Loop:**

Inside the main loop, sensor data (temperature, humidity, and CO2 levels) is generated using generate\_sensor\_data() function.

The generated data is printed to the console for monitoring.

Sensor data is published to the ThingSpeak channel using publish\_sensor\_data() function.

A delay of 20 seconds is added to adhere to the ThingSpeak update limits.

The loop continues until it's interrupted by the user (KeyboardInterrupt).

**Exception Handling:**

The code includes exception handling to gracefully handle keyboard interrupt (Ctrl+C) by disconnecting the MQTT client.

Overall, this code demonstrates a simple IoT simulation where random sensor data is generated and published to a ThingSpeak channel using MQTT protocol, allowing for real-time monitoring and visualization of sensor readings.

**4. Data for Station 1:**

- The data transmitted to ThingSpeak for Station 1 included:

1. Temperature

A graph with red dots

Description automatically generated

Visualization:-

A graph of a temperature variation

Description automatically generated

2. Humidity

A screen shot of a graph

Description automatically generated

Visualization:-

A graph of a person with blue lines

Description automatically generated with medium confidence

3. CO2 concentration

A graph with red lines

Description automatically generated

Visualization:-

A graph of a graph

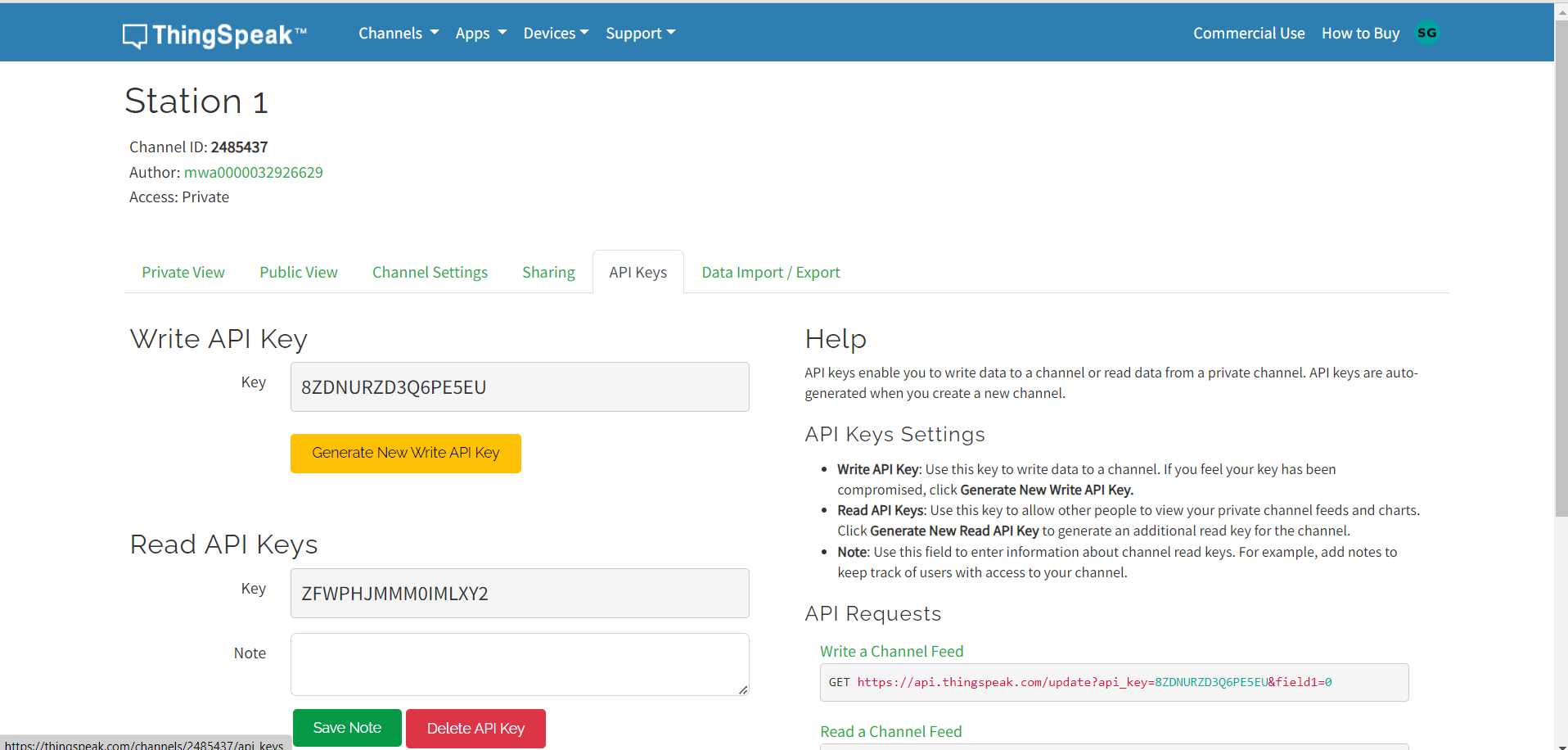
Description automatically generated with medium confidence

Channel creations:-

A screenshot of a computer

Description automatically generated

Station 1:-



Station 2:

A screenshot of a computer

Description automatically generated

Device created for station 2:

A white background with black text

Description automatically generated