Name:Vansh Rahate
SUID: 547161826

IOT- Assignment 3

STEPS:

To develop the cloud-based IoT system, I used the ThingSpeak platform with Python to simulate virtual environmental stations. Here's a breakdown of the steps:

ThingSpeak Channel Setup

- A ThingSpeak channel contained three data fields for Temperature, Humidity and CO2 measurements.
- The system required both the Channel ID and Write API key for MQTT or HTTP data push functionality.

Virtual Sensor Creation

- Developed a Python script to simulate random sensor data (Temperature: -50 to 50°C, Humidity: 0–100%, CO2: 300–2000 ppm).
- The paho-mqtt library proved initially useful for ThingSpeak data publication yet I had to switch to ThingSpeak's HTTP API with requests due to Colab restrictions.

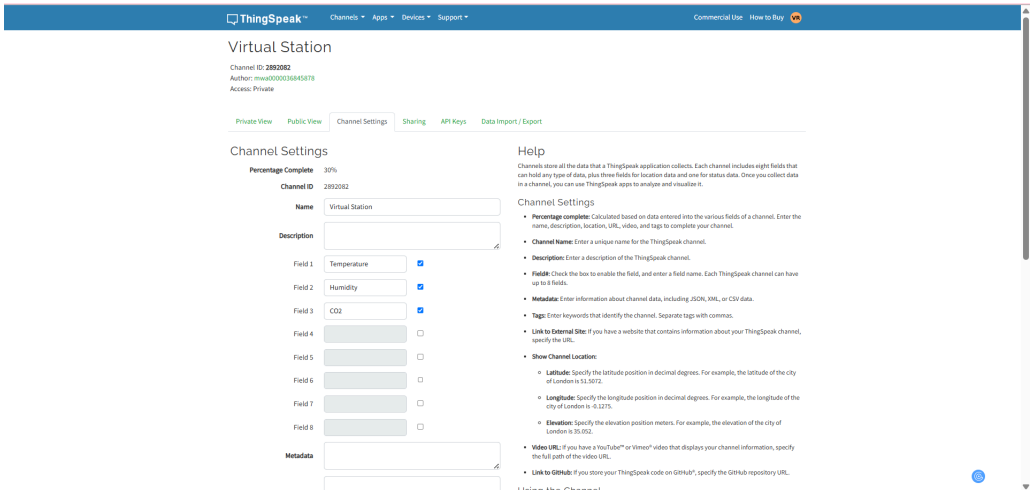
Data Collection and Visualization

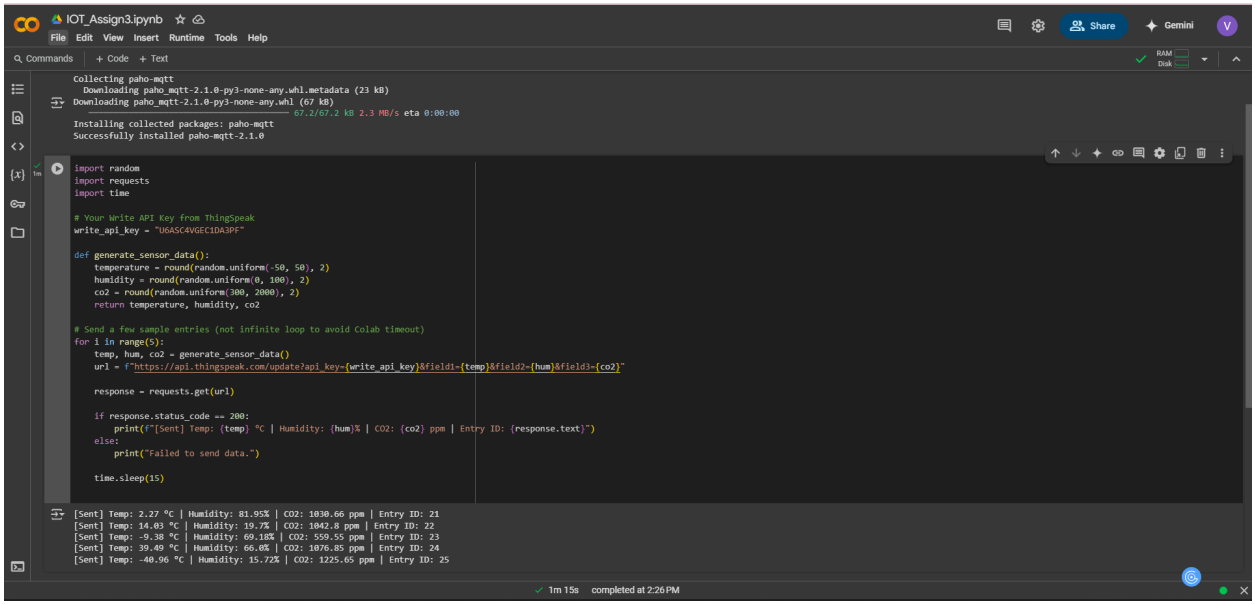
- Sent multiple data points using the API.
- Retrieved sensor values via ThingSpeak's REST API.
- Visualized the data using matplotlib in Colab.

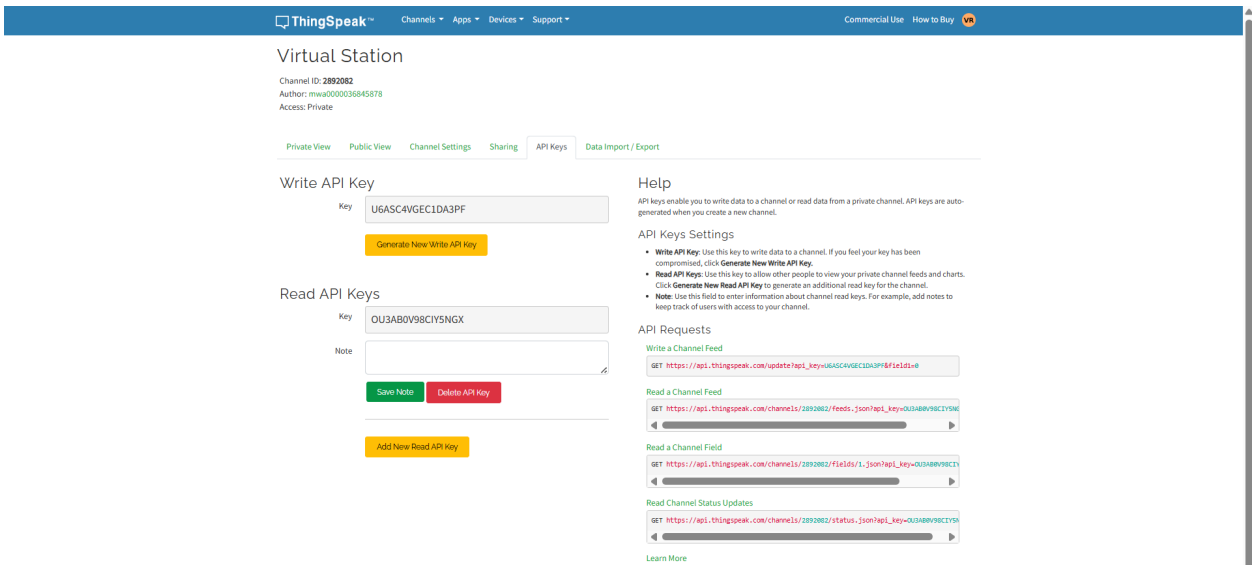
Output & GitHub Submission

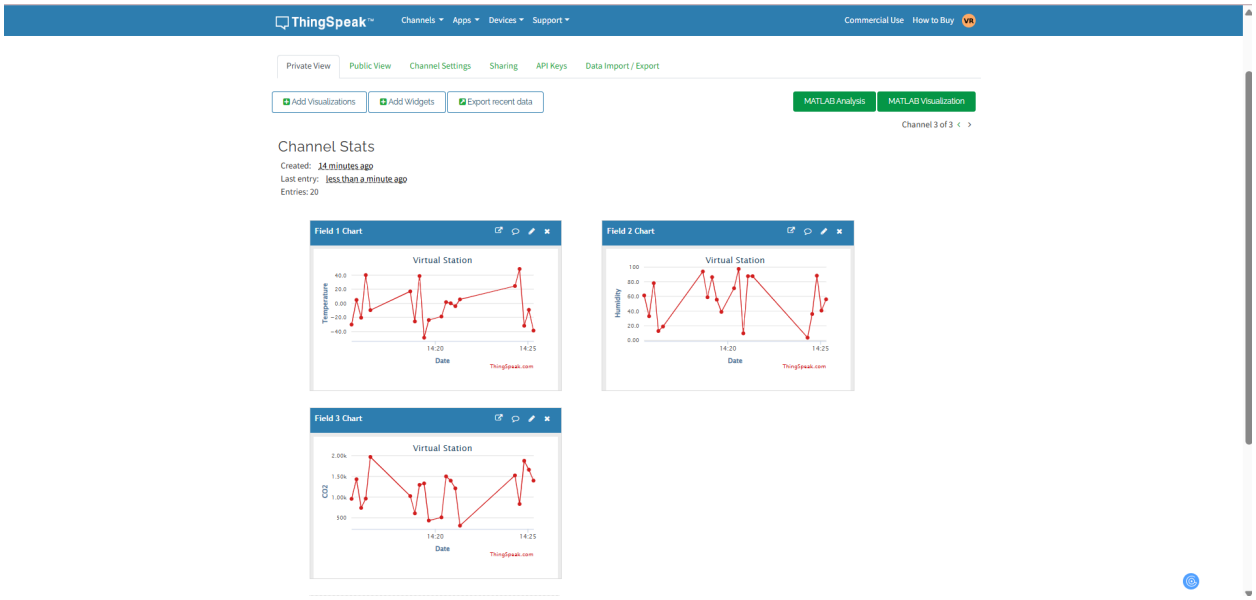
- Captured screenshots of channel output and plots.
- Uploaded code and notebook to GitHub with a structured README.

Screenshots




IOT_Assign3.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Collecting paho-mqtt
Downloading paho_mqtt-2.1.0-py3-none-any.whl.metadata (23 kB)
Downloading paho_mqtt-2.1.0-py3-none-any.whl (67 kB)
67.2/67.2 kB 2.3 MB/s eta 0:00:00
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-2.1.0
import random
import requests
import time
# Your Write API Key from ThingSpeak
write_api_key = "U6ASC4VGEC1DA3PF"
def generate_sensor_data():
temperature = round(random.uniform(-50, 50), 2)
humidity = round(random.uniform(0, 100), 2)
co2 = round(random.uniform(300, 2000), 2)
return temperature, humidity, co2
# Send a few sample entries (not infinite loop to avoid Colab timeout)
for i in range(5):
temp, hum, co2 = generate_sensor_data()
url = f"https://api.thingspeak.com/update?api_key={write_api_key}&field1={temp}&field2={hum}&field3={co2}"
response = requests.get(url)
if response.status_code == 200:
print(f"[Sent] Temp: {temp} °C | Humidity: {hum}% | CO2: {co2} ppm | Entry ID: {response.text}")
else:
print("Failed to send data.")
time.sleep(15)
[Sent] Temp: 2.27 °C | Humidity: 81.95% | CO2: 1030.66 ppm | Entry ID: 21
[Sent] Temp: 14.03 °C | Humidity: 19.7% | CO2: 1042.8 ppm | Entry ID: 22
[Sent] Temp: -9.38 °C | Humidity: 69.18% | CO2: 559.55 ppm | Entry ID: 23
[Sent] Temp: 39.49 °C | Humidity: 66.6% | CO2: 1076.85 ppm | Entry ID: 24
[Sent] Temp: -40.96 °C | Humidity: 15.72% | CO2: 1225.65 ppm | Entry ID: 25
1m 15s completed at 2:26 PM


ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Virtual Station
Channel ID: 2892082
Author: mwa0000036845878
Access: Private
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Write API Key
Key
U6ASC4VGEC1DA3PF
Generate New Write API Key
Read API Keys
Key
OU3AB0V98CIY5NGX
Note
Save Note
Delete API Key
Add New Read API Key
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=U6ASC4VGEC1DA3PF&field1=0
Read a Channel Feed
GET https://api.thingspeak.com/channels/2892082/feeds.json?api_key=OU3AB0V98CIY5NG
Read a Channel Field
GET https://api.thingspeak.com/channels/2892082/fields/1.json?api_key=OU3AB0V98CIY
Read Channel Status Updates
GET https://api.thingspeak.com/channels/2892082/status.json?api_key=OU3AB0V98CIY5N
Learn More



Reflection:

This assignment was a great hands-on experience that helped me better understand how IoT systems work with cloud platforms. I initially tried using MQTT with the paho-mqtt library, but ran into issues running it on Google Colab. After some trial and error, I switched to using HTTP requests with ThingSpeak's API, which worked much better in that environment.

One thing I learned is how important it is to adapt when things don't go as planned. Debugging API errors and figuring out how to structure the requests correctly helped me understand how cloud-based communication really works. Visualizing the data in Colab was also really satisfying, it made the whole system feel real.

Overall, I feel more confident working with IoT data, APIs, and cloud platforms now, and I enjoyed the process of solving real-world problems during this project.

Github:

https://github.com/Vansh3117/iot-environmental-sensor