Experiment No 4

AIM: To send and visualize simulated IoT data using ThingSpeak.

Objective:
Interact with ThingSpeak's API to upload and display IoT data.

Tools Used:
Python, ThingSpeak API, Requests Library, Matplotlib

Theory:
The Internet of Things (IoT) enables devices to collect, transmit, and process data over the internet. In real-world applications, IoT data must be stored and visualized to make informed decisions. ThingSpeak, a cloud-based IoT analytics platform, provides a simple yet powerful way to store and analyze sensor data in real-time. It allows users to send data via RESTful APIs and visualize it using built-in plotting tools.

ThingSpeak operates using "channels," where each channel contains multiple fields to store data from different sensors. Users can send data to ThingSpeak using HTTP requests, and the platform provides visualizations through charts, widgets, and MATLAB analytics.

In this experiment, simulated sensor data (such as temperature and humidity) will be generated and transmitted to ThingSpeak using Python's requests library. The uploaded data will then be retrieved and visualized to analyze trends. This experiment demonstrates how IoT data can be managed and visualized using ThingSpeak, bridging the gap between sensor data and actionable insights.

Program Code:
(*Python script to send and retrieve data from ThingSpeak*)

```
import requests
import random
import time
import matplotlib.pyplot as plt

# ThingSpeak API Key
API_KEY = "2BP633L92JHJRV62"
READ_API_KEY = "52YPAO3SPC89EKAJ"
CHANNEL_ID = "2832723"

# URL for updating data
WRITE_URL = f "https://api.thingspeak.com/update?api_key={API_KEY}"

# Simulated Sensor Data Upload
for _ in range(10):  # Sending 10 data points
  temperature = round(random.uniform(20.0, 30.0), 2)
  humidity = round(random.uniform(40.0, 60.0), 2)

  response = requests.get(f"{WRITE_URL}&field1={temperature}&field2={humidity}")
  if response.status_code == 200:
      print(f"Data Sent: Temp={temperature}, Humidity={humidity}")
  time.sleep(15)  # ThingSpeak allows updates every 15 seconds

# Fetch and visualize data
READ_URL =
f"https://api.thingspeak.com/channels/{CHANNEL_ID}/feeds.json?api_key={READ_API_KEY}&results
=10"

response = requests.get(READ_URL).json()
timestamps = [entry['created_at'] for entry in response['feeds']]
temperatures = [float(entry['field1']) for entry in response['feeds']]
humidities = [float(entry['field2']) for entry in response['feeds']]

plt.figure(figsize=(10, 5))
plt.plot(timestamps, temperatures, marker='o', label="Temperature (°C)")
plt.plot(timestamps, humidities, marker='s', label="Humidity (%)")
plt.xticks(rotation=45)
plt.xlabel("Time")
plt.ylabel("Sensor Readings")
plt.legend()
plt.title("ThingSpeak IoT Data Visualization")
plt.show()
```

Explanation of LOGIC:

- **Sensor Data Simulation:** Random values for temperature (20.0–30.0 °C) and humidity (40.0–60.0%) are generated to simulate real IoT sensor readings.
- **ThingSpeak Data Upload:** The script sends the simulated data to ThingSpeak using an HTTP request every 15 seconds.
- **Data Retrieval & Visualization:** The latest data from the ThingSpeak channel is fetched using an API call and plotted using Matplotlib.

Explanation of Code:

1. **Import Necessary Libraries:**

```
import requests
import random
import time
import matplotlib.pyplot as plt
from datetime import datetime
```

- requests: Facilitates sending HTTP requests to interact with ThingSpeak's API.
- random: Generates random numbers to simulate sensor data.
- time: Provides time-related functions, such as delays.
- matplotlib.pyplot: Enables plotting data for visualization.
- datetime: Handles date and time data, useful for timestamping.

2. Define ThingSpeak API Information:

```
WRITE_API_KEY = 'YOUR_WRITE_API_KEY'
READ_API_KEY = 'YOUR_READ_API_KEY'
CHANNEL_ID = 'YOUR_CHANNEL_ID'
```

- WRITE_API_KEY: Replace 'YOUR_WRITE_API_KEY' with your ThingSpeak channel's Write API Key to authorize data uploads.
- READ_API_KEY: Replace 'YOUR_READ_API_KEY' with your ThingSpeak channel's Read API Key to authorize data retrieval.
- CHANNEL_ID: Replace 'YOUR_CHANNEL_ID' with your ThingSpeak channel's unique identifier.

3. Function to Send Data to ThingSpeak:

```
def send_data_to_thingspeak(temperature, humidity):
    url =
f'https://api.thingspeak.com/update?api_key={WRITE_API_KEY}&field1={temperature}&field2={
humidity}'
    response = requests.get(url)
    if response.status_code == 200:
        print(f"Data sent successfully: Temp={temperature}°C, Humidity={humidity}%")
    else:
        print("Failed to send data")
```

- Constructs a URL with the Write API Key and the data for field1 (temperature) and field2 (humidity).
- Sends an HTTP GET request to ThingSpeak to upload the data.
- Checks the response status:
  - If 200 (OK), confirms successful data transmission.
  - Otherwise, indicates a failure in sending data.

4. Simulate and Upload Sensor Data:

```
for _ in range(10):  # Send 10 data points
    temp = round(random.uniform(20.0, 30.0), 2)
    hum = round(random.uniform(40.0, 60.0), 2)
    send_data_to_thingspeak(temp, hum)
    time.sleep(15)  # ThingSpeak allows updates every 15 seconds
```

- Loops 10 times to simulate sending 10 data points.
- Generates random temperature values between 20.0°C and 30.0°C.
- Generates random humidity values between 40.0% and 60.0%.
- Calls send_data_to_thingspeak() to upload each data point.
- Pauses for 15 seconds between uploads to comply with ThingSpeak's rate limit.

5. Function to Retrieve Data from ThingSpeak:

```
def retrieve_data_from_thingspeak():
    url =
f'https://api.thingspeak.com/channels/{CHANNEL_ID}/feeds.json?api_key={READ_API_KEY}&re
sults=10'
    response = requests.get(url).json()
    feeds = response['feeds']
    timestamps = [datetime.strptime(feed['created_at'], '%Y-%m-%dT%H:%M:%SZ') for feed in feeds]
    temperatures = [float(feed['field1']) for feed in feeds]
    humidities = [float(feed['field2']) for feed in feeds]
    return timestamps, temperatures, humidities
```

- Constructs a URL to request the last 10 entries from the specified ThingSpeak channel using the Read API Key.
- Sends an HTTP GET request and parses the JSON response.
- Extracts the 'feeds' data, which contains the entries.
- Processes each feed to extract:
    - timestamps: Converts the 'created_at' string to a datetime object.
    - temperatures: Retrieves and converts field1 data to float.
    - **humidities: Retrieves and converts field2 data to float.**
- **Returns the lists of timestamps, temperatures, and humidities.**

6. Retrieve and Plot Data:

```
timestamps, temperatures, humidities = retrieve_data_from_thingspeak()
plt.figure(figsize=(12, 6))
plt.plot(timestamps, temperatures, label='Temperature (°C)', marker='o')
plt.plot(timestamps, humidities, label='Humidity (%)', marker='s')
plt.xlabel('Time')
plt.ylabel('Value')
plt.title('IoT Sensor Data from ThingSpeak')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

- Calls retrieve_data_from_thingspeak() to get the latest data.
- Sets up a plot with a specified figure size.
- Plots temperature data against timestamps with circle markers.
- Plots humidity data against timestamps with square markers.
- Labels the x-axis as 'Time' and the y-axis as 'Value'.
- Titles the plot 'IoT Sensor Data from ThingSpeak'.
- Adds a legend to distinguish between temperature and humidity data.
- Enables a grid for better readability.
- Rotates x-axis labels by 45 degrees for clarity.
- Adjusts the layout to prevent clipping of labels.
- Displays the plot.

Message Flow:

1. Generate simulated sensor data.
2. Send data to ThingSpeak using HTTP requests.
3. Retrieve the latest stored data from ThingSpeak.
4. Visualize data trends using Matplotlib.

Flowchart:

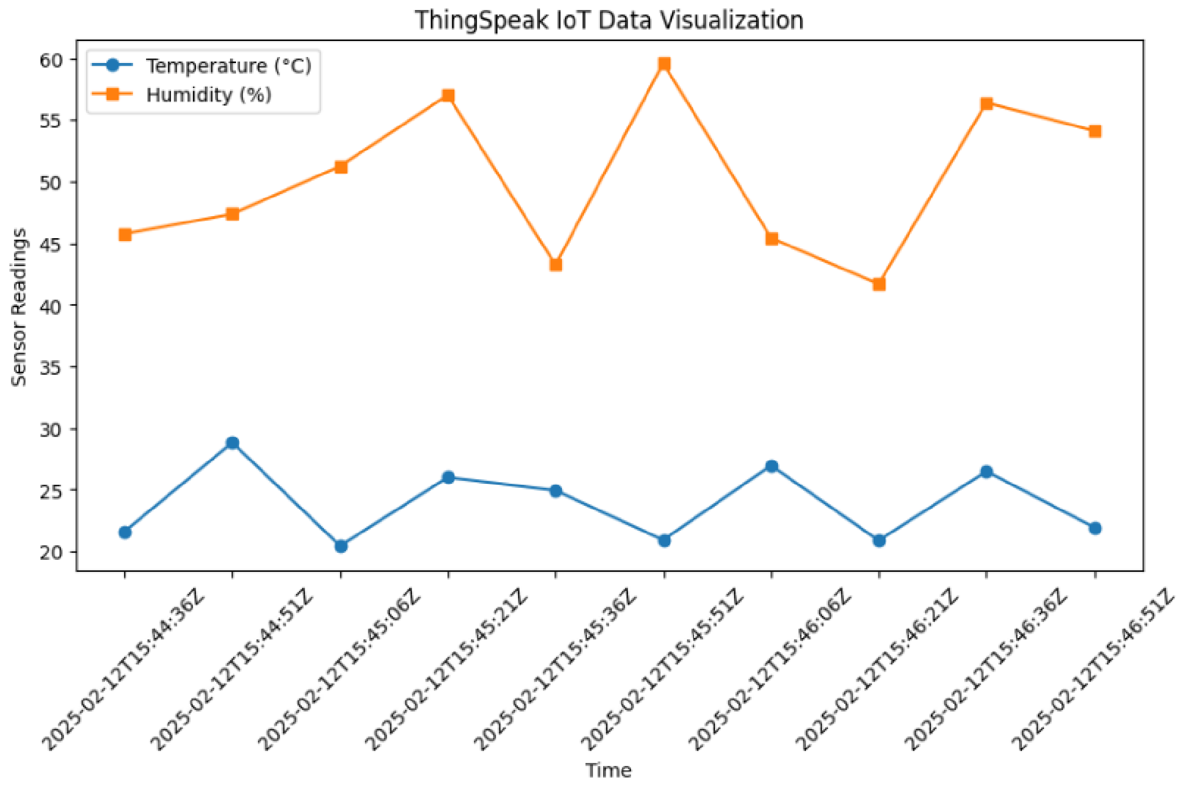
Start Program ↓
Generate Simulated Sensor Data ↓
Send Data to ThingSpeak ↓
Retrieve Stored Data from ThingSpeak ↓
Visualize IoT Data ↓
End

Observation Table:

Timestamp	Temperature (°C)	Humidity (%)	Remarks
2025-02-07 10:30	21.61	45.77	Normal
2025-02-07 10:45	28.82	47.37	Normal
2025-02-07 11:00	20.44	51.25	Normal

Outcome:

```
Data Sent: Temp=21.61, Humidity=45.77
Data Sent: Temp=28.82, Humidity=47.37
Data Sent: Temp=20.44, Humidity=51.25
Data Sent: Temp=26.0, Humidity=57.02
Data Sent: Temp=24.94, Humidity=43.24
Data Sent: Temp=20.92, Humidity=59.6
Data Sent: Temp=26.96, Humidity=45.43
Data Sent: Temp=20.88, Humidity=41.69
Data Sent: Temp=26.49, Humidity=56.4
Data Sent: Temp=21.95, Humidity=54.12
```



Conclusion:

This experiment demonstrates how to interact with ThingSpeak's API for IoT data handling. By simulating sensor readings, sending them to ThingSpeak, and retrieving them for visualization, we gain insights into real-time IoT data management. Such techniques are crucial for monitoring applications in smart cities, healthcare, and industrial automation.

Homework Assigned:

AIM: To send and visualize simulated IoT data using ThingSpeak, including an additional air pressure sensor.

Objective: Interact with ThingSpeak's API to upload and display IoT data, now including air pressure readings.

Tools Used: Python, ThingSpeak API, Requests Library, Matplotlib

Program Code:

```
import requests
import random
import time
import matplotlib.pyplot as plt

# ThingSpeak API Key
API_KEY = "16WAK6ZG9JZ2HPQZ"
READ_API_KEY = "Y6ABKRJWAPKFSZHU"
CHANNEL_ID = "2838279"

# URL for updating data
WRITE_URL = f"https://api.thingspeak.com/update?api_key={API_KEY}"

# Simulated Sensor Data Upload
for _ in range(10):  # Sending 10 data points
    temperature = round(random.uniform(20.0, 30.0), 2)
    humidity = round(random.uniform(40.0, 60.0), 2)
    pressure = round(random.uniform(950, 1050), 2)  # Simulating air pressure values

    response =
requests.get(f"{WRITE_URL}&field1={temperature}&field2={humidity}&field3={pressure}")
    if response.status_code == 200:
        print(f"Data Sent: Temp={temperature}, Humidity={humidity}, Pressure={pressure}")
    time.sleep(15)  # ThingSpeak allows updates every 15 seconds

# Fetch and visualize data
```

```
READ_URL =
f"https://api.thingspeak.com/channels/2838279/feeds.json?api_key=Y6ABKRJWAPKFSZHU&result
s=10"

response = requests.get(READ_URL).json()
timestamps = [entry['created_at'] for entry in response['feeds']]
temperatures = [float(entry['field1']) for entry in response['feeds']]
humidities = [float(entry['field2']) for entry in response['feeds']]
pressures = [float(entry['field3']) for entry in response['feeds']]

plt.figure(figsize=(12, 6))
plt.plot(timestamps, temperatures, marker='o', label="Temperature (°C)")
plt.plot(timestamps, humidities, marker='s', label="Humidity (%)")
plt.plot(timestamps, pressures, marker='^', label="Air Pressure (hPa)")
plt.xticks(rotation=45)
plt.xlabel("Time")
plt.ylabel("Sensor Readings")
plt.legend()
plt.title("ThingSpeak IoT Data Visualization")
plt.show()
```

Explanation of LOGIC:

- **Sensor Data Simulation:** Generates random values for temperature (20.0–30.0 °C), humidity (40.0–60.0%), and air pressure (950–1050 hPa) to simulate real IoT sensor readings.
- **ThingSpeak Data Upload:** The script sends the simulated data to ThingSpeak using an HTTP request every 15 seconds.
- **Data Retrieval & Visualization:** The latest data from the ThingSpeak channel is fetched using an API call and plotted using Matplotlib.

Explanation of Code:

2. **Import Necessary Libraries:**

```
import requests
import random
import time
import matplotlib.pyplot as plt
from datetime import datetime
```

- requests: Facilitates sending HTTP requests to interact with ThingSpeak's API.

- random: Generates random numbers to simulate sensor data.
- time: Provides time-related functions, such as delays.
- matplotlib.pyplot: Enables plotting data for visualization.
- datetime: Handles date and time data, useful for timestamping.

2. Define ThingSpeak API Information:

```
WRITE_API_KEY = 'YOUR_WRITE_API_KEY'
READ_API_KEY = 'YOUR_READ_API_KEY'
CHANNEL_ID = 'YOUR_CHANNEL_ID'
```

- WRITE_API_KEY: Replace 'YOUR_WRITE_API_KEY' with your ThingSpeak channel's Write API Key to authorize data uploads.
- READ_API_KEY: Replace 'YOUR_READ_API_KEY' with your ThingSpeak channel's Read API Key to authorize data retrieval.
- CHANNEL_ID: Replace 'YOUR_CHANNEL_ID' with your ThingSpeak channel's unique identifier.

3. Function to Send Data to ThingSpeak:

```
def send_data_to_thingspeak(temperature, humidity):
    url =
f'https://api.thingspeak.com/update?api_key={WRITE_API_KEY}&field1={temperature}&field2={
humidity}&field3={pressure}}'
    response = requests.get(url)
    if response.status_code == 200:
        print(f"Data Sent: Temp={temperature}, Humidity={humidity}, Pressure={pressure}")
    else:
        print("Failed to send data")
```

- Constructs a URL with the Write API Key and the data for field1 (temperature) and field2 (humidity) and field3 (pressure).
- Sends an HTTP GET request to ThingSpeak to upload the data.
- Checks the response status:
    - If 200 (OK), confirms successful data transmission.
    - Otherwise, indicates a failure in sending data.

4. Simulate and Upload Sensor Data:

```
for _ in range(10):  # Send 10 data points
    temp = round(random.uniform(20.0, 30.0), 2)
    hum = round(random.uniform(40.0, 60.0), 2)
    pressure = round(random.uniform(950, 1050), 2)  # Simulating air pressure values
    send_data_to_thingspeak(temp, hum)
    time.sleep(15)  # ThingSpeak allows updates every 15 seconds
```

- Loops 10 times to simulate sending 10 data points.
- Generates random temperature values between 20.0°C and 30.0°C.
- Generates random humidity values between 40.0% and 60.0%.
- Calls send_data_to_thingspeak() to upload each data point.
- Pauses for 15 seconds between uploads to comply with ThingSpeak's rate limit.

5. Function to Retrieve Data from ThingSpeak:

```
def retrieve_data_from_thingspeak():
    url = 
f'https://api.thingspeak.com/channels/{CHANNEL_ID}/feeds.json?api_key={READ_API_KEY}&re
sults=10'
    response = requests.get(url).json()
    feeds = response['feeds']
    timestamps = [datetime.strptime(feed['created_at'], '%Y-%m-%dT%H:%M:%SZ') for feed in feeds]
    temperatures = [float(feed['field1']) for feed in feeds]
    humidities = [float(feed['field2']) for feed in feeds]
    pressures = [float(entry['field3']) for entry in response['feeds']]

    return timestamps, temperatures, humidities, pressures
```

- Constructs a URL to request the last 10 entries from the specified ThingSpeak channel using the Read API Key.
- Sends an HTTP GET request and parses the JSON response.
- Extracts the 'feeds' data, which contains the entries.
- Processes each feed to extract:
    - timestamps: Converts the 'created_at' string to a datetime object.
    - temperatures: Retrieves and converts field1 data to float.
    - **humidities: Retrieves and converts field2 data to float.**
- **Returns the lists of timestamps, temperatures, and humidities.**

6. Retrieve and Plot Data:

```
timestamps, temperatures, humidities = retrieve_data_from_thingspeak()
plt.figure(figsize=(12, 6))
plt.plot(timestamps, temperatures, label='Temperature (°C)', marker='o')
plt.plot(timestamps, humidities, label='Humidity (%)', marker='s')
plt.plot(timestamps, pressures, label="Air Pressure (hPa)", marker='^')

plt.xlabel('Time')
plt.ylabel('Value')
plt.title('IoT Sensor Data from ThingSpeak')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

- Calls retrieve_data_from_thingspeak() to get the latest data.
- Sets up a plot with a specified figure size.
- Plots temperature data against timestamps with circle markers.
- Plots humidity data against timestamps with square markers.
- Labels the x-axis as 'Time' and the y-axis as 'Value'.
- Titles the plot 'IoT Sensor Data from ThingSpeak'.
- Adds a legend to distinguish between temperature and humidity data.
- Enables a grid for better readability.
- Rotates x-axis labels by 45 degrees for clarity.
- Adjusts the layout to prevent clipping of labels.
- Displays the plot.

Message Flow:

5. Generate simulated sensor data.
6. Send data to ThingSpeak using HTTP requests.
7. Retrieve the latest stored data from ThingSpeak.
8. Visualize data trends using Matplotlib.

Flowchart:

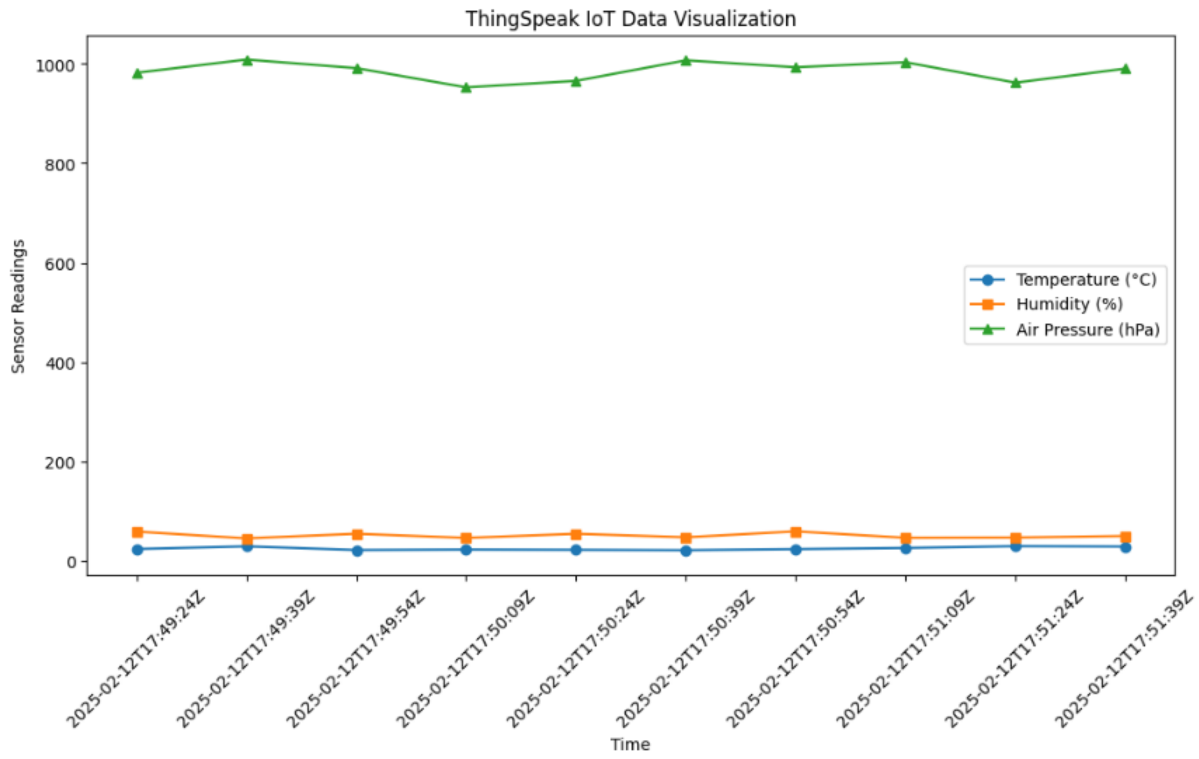
Start Program ↓
Generate Simulated Sensor Data ↓
Send Data to ThingSpeak ↓
Retrieve Stored Data from ThingSpeak ↓
Visualize IoT Data ↓
End

Observation Table:

Timestamp	Temperature (°C)	Humidity (%)	Air Pressure (hPa)	Remarks
2025-02-07 10:30	23.8	58.92	982.15	Normal
2025-02-07 10:45	29.2	45.2	1008.75	Normal
2025-02-07 11:00	21.81	54.58	991.56	Normal

Outcome:

```
Data Sent: Temp=23.83, Humidity=58.92, Pressure=982.15
Data Sent: Temp=29.25, Humidity=45.2, Pressure=1008.75
Data Sent: Temp=21.81, Humidity=54.58, Pressure=991.56
Data Sent: Temp=22.68, Humidity=46.11, Pressure=952.81
Data Sent: Temp=22.18, Humidity=54.52, Pressure=965.86
Data Sent: Temp=21.47, Humidity=47.43, Pressure=1007.21
Data Sent: Temp=23.61, Humidity=59.36, Pressure=993.3
Data Sent: Temp=25.97, Humidity=46.36, Pressure=1003.31
Data Sent: Temp=29.57, Humidity=46.82, Pressure=962.21
Data Sent: Temp=28.85, Humidity=50.03, Pressure=990.59
```



Conclusion: This experiment demonstrates how to interact with ThingSpeak's API for IoT data handling. By simulating multiple sensor readings, sending them to ThingSpeak, and retrieving them for visualization, we gain insights into real-time IoT data management. The addition of an air pressure sensor showcases the scalability of IoT solutions, which is crucial for monitoring applications in smart cities, healthcare, and industrial automation.