Internet Of Things

Basics of HTTP and ThingSpeak

Prepared by:

Dr. Murad Yaghi
Eng. Malek Al-Louzi

School of Computing and Informatics – Al Hussein Technical University

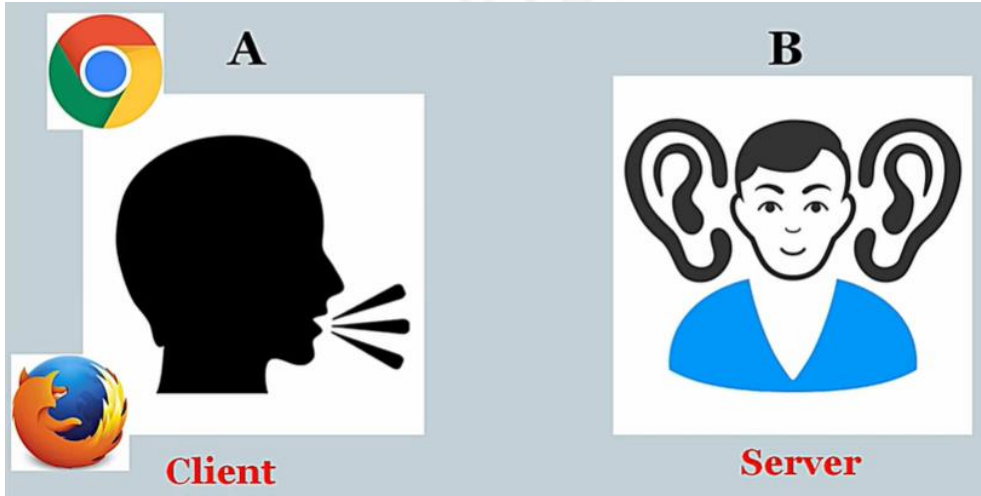
Fall 2024/2025

Introduction

- HTTP stands for Hyper Text Transfer Protocol
- Most widely used by web services for communication over the internet
- It defines:
  - How messages are formatted and transmitted
  - How web servers and browsers should respond to various commands

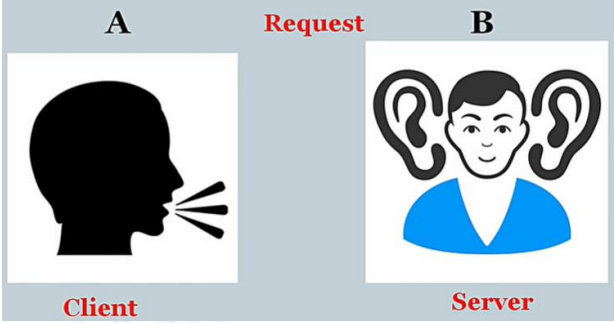
Introduction

- To communicate from one node to another node in internet using HTTP
- Node A called web client and usually it’s a web browser that initiates the communication
- Node B called web server that waits a client request

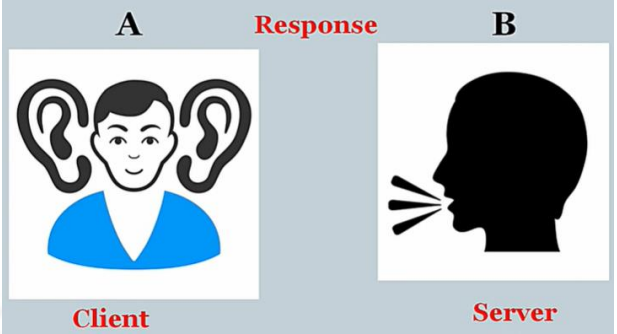


Introduction

- Request
  - Client sends a request: The browser requests resources like HTML pages, images, or videos from the server



- Response
  - Server responds: The server sends back the requested resource or an error message



Introduction

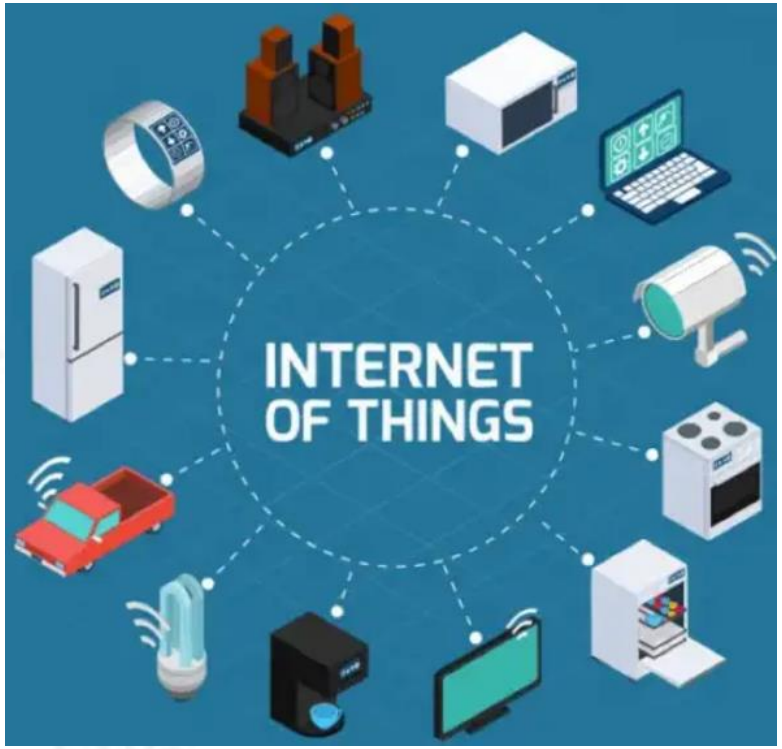
- For each request from the client, the server will send a response to the client
- Number of requests equals to number of responses

Introduction

- **HTTP Methods:** Common HTTP methods include
  - **GET:** Retrieve data (e.g., loading a webpage)
  - **POST**: Send data to the server (e.g., submitting a form)
  - **PUT**: Update data on the server
  - **DELETE**: Remove data from the server

ThingSpeak

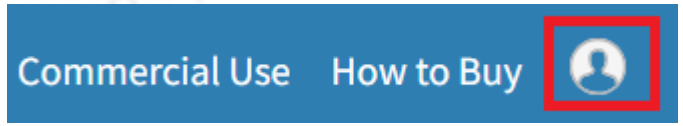
- In the below figure, you can see various devices (car, laptop, camera, etc)
- The common thing for all theses devices (things) is the internet
- That’s why we called it internet of things

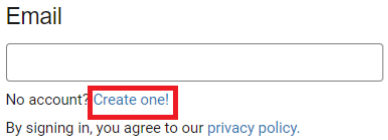


ThingSpeak

- To establish the IoT concept, we need a very important component called cloud server
- The server that we will use in this class is **ThingSpeak** IoT cloud platform
- ThingSpeak is an IoT platform created by MathWorks for implementing IoT concepts

ThingSpeak

- To access ThingSpeak, follow the following steps:
  - Open a web browser and go to https://thingspeak.com
  - Click on the Sign In button on the top right, 

    then create an account 

  - Provide your email address and other required details
  - Activate your account through the confirmation email sent to your inbox

ThingSpeak Channels

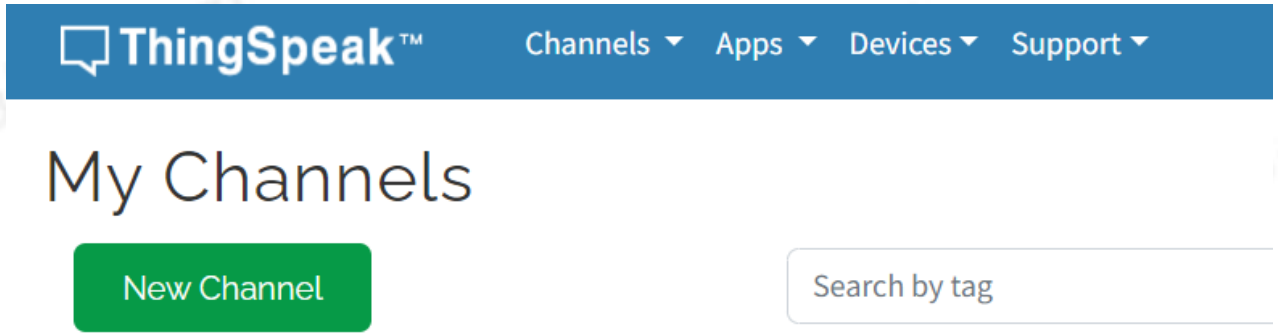
- Navigate to the **Channels** tab
- Channels are data containers where you can store, organize, and share IoT data
- Each channel can be configured as either Private or Public
  - Public: data in public channels is visible to anyone on the internet
  - Private: accessible only by channel owner (and authorized users with API keys)
- You can create up to 4 channels with your free account

ThingSpeak Fields

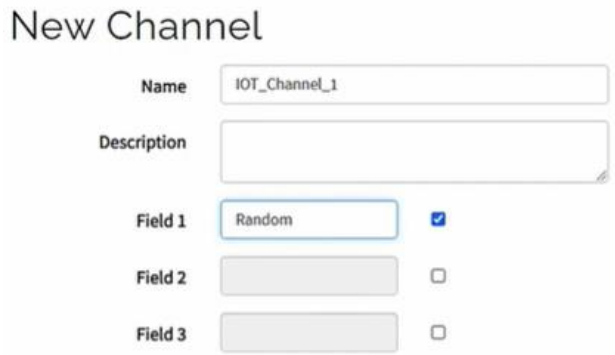
- In ThingSpeak, a field is a specific data storage space within a channel used to record and organize a particular type of data
- Each field corresponds to one type of sensor reading or data point you want to track
- Fields help structure your data, for example, if you're monitoring weather, you might have:
  - Field 1: Temperature
  - Field 2: Humidity
  - Field 3: Wind Speed

ThingSpeak
Writing Data Using WebBrowser

- After creating your account and logging in, go to **My Channels**

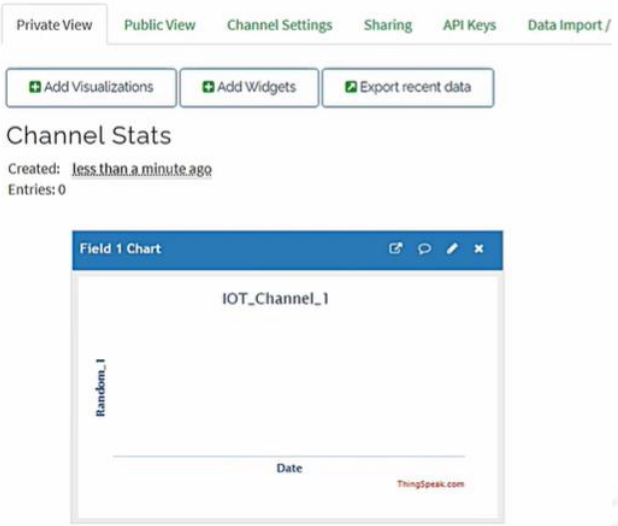


- Click on **New Channel** to create a new channel
- Give a name for the channel and a name for the field, all other options are optional



ThingSpeak
Writing Data Using WebBrowser

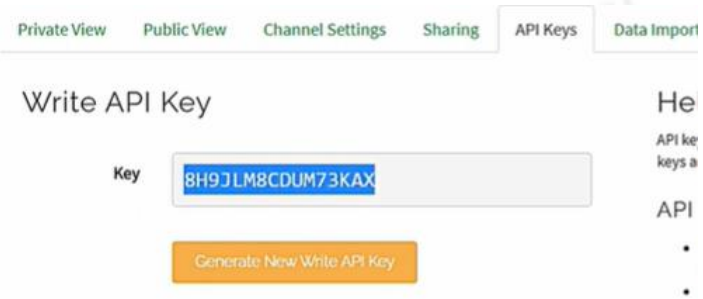
- You can create up to 8 fields in the channel
- This is the field within the channel that we created



- Any data that we are sending from IoT device will be showed on the created field

ThingSpeak
Writing Data Using WebBrowser

- We can send data and read data from a field using API keys
- Go to API Keys tab



- API Keys are very important when reading from or writing on a particular field in a channel
- Write API Key: will be used when writing on a particular field in a channel
- Reda API Keys: will be used when reading from a particular filed in a channel

ThingSpeak
Writing Data Using WebBrowser

- For every channel that we are creating, there will be an ID allocated for it
- We will use HTTP to read and write data on fields
- The request format that we will use to read and write data is GET request
- On the right of the API Keys webpage, you will see several GET requests

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=8H9JLM8CDUM73KAX&field
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/1400595/feeds.json?api_key=
```

Read a Channel Field

```
GET https://api.thingspeak.com/channels/1400595/fields/1.json?api_ke
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/1400595/status.json?api_key=
```

ThingSpeak
Writing Data Using WebBrowser

- You can write data to specific fields in your channel by sending a GET request to the ThingSpeak API using the following URL

Host name — Values

```
https://api.thingspeak.com/update?api_key=YOUR_WRITE_API_KEY&field1=value1&field2=value2
```

process that we want to do

- Replace `YOUR_WRITE_API_KEY` with your channel's **Write API Key**
- Replace `field1`, `field2` ,etc., with the data you want to send
- Example:

```
https://api.thingspeak.com/update?api_key=ABC123XYZ&field1=25&field2=60
```

ThingSpeak
Writing Data Using WebBrowser

- You can test the previous GET request using your browser
- Browsers are capable to send GET requests
- To test it, open your browser and put the previous URL with desired fields values, keep in mind the number of fields created in your channel

ThingSpeak
Writing Data Using HTTP in Python

- Now we are going to do the same process, but we will use Python code to send the GET request from Raspberry Pi to write data to a field in your ThigSpeak channel
- You need to install **requests** library if its not already installed

```
pip install requests --break-system-packages
```

ThingSpeak
Writing Data Using HTTP in Python

- Here’s an example Python code that writes data to a ThingSpeak field

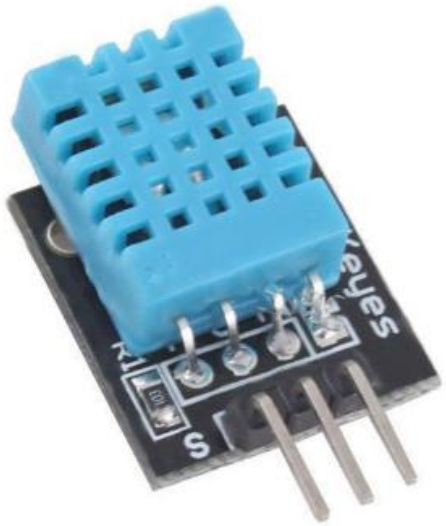
```python
import requests

# ThingSpeak API details
WRITE_API_KEY = "YOUR_WRITE_API_KEY"  # Replace with your Write API Key
URL = "https://api.thingspeak.com/update"

# Data to send
payload = {
    "api_key": WRITE_API_KEY,
    "field1": 25,  # Example: Temperature value
    "field2": 60   # Example: Humidity value
}

# Send data to ThingSpeak
response = requests.get(URL, params=payload)

if response.status_code == 200:
    print(f"Data written successfully. Entry ID: {response.text}")
else:
    print(f"Failed to write data. HTTP Code: {response.status_code}")
```

ThingSpeak
Writing Data Using HTTP in Python

- **Explanation of the Code:**
- Replace `YOUR_API_KEY` with your actual Write API key from ThingSpeak
- `URL` This is the ThingSpeak API endpoint for updating a channel
- `field1` This is the field you want to update, you can modify values for field1, field2, etc., depending on which field you want to write to in your ThingSpeak channel
- `requests.get()` The data is sent via a GET request with the **params** argument containing the payload, including the API key and the data for the field

ThingSpeak Exercise

- Read Temperature and Humidity from DHT11 Sensor and Send Data to ThingSpeak



ThingSpeak Exercise

- **Hints:**

1. Connect the DHT11 sensor to the Raspberry Pi
   - VCC -> 3.3V
   - GND -> GND
   - DATA -> GPIO pin (e.g., GPIO17)
2. Install required libraries
   - **Adafruit CircuitPython DHTlibrary** and **Adafruit Blinka** to interact with the DHT11 sensor

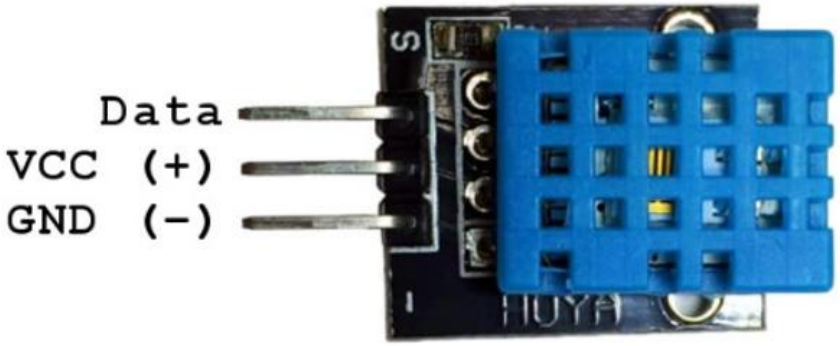
```
pip3 install adafruit-circuitpython-dht
```

3. Now start writing your code

ThingSpeak Exercise

- **Hints:**

4. The pinout for DHT11 sensor:



5. Wiring the DHT11 to the Raspberry Pi:
   - VCC → 3.3V (Pin 1 on the Raspberry Pi)
   - GND → Ground (Pin 6 on the Raspberry Pi)
   - DATA → GPIO Pin (For example, GPIO4, which is Pin 7 on the Raspberry Pi)

ThingSpeak Exercise

- **Hints:**

6. The code to read temperature and humidity from DHT11 sensor:

```
import time
import board
import adafruit_dht

# Sensor data pin is connected to GPIO 4
sensor = adafruit_dht.DHT11(board.D4)
while True:
    try:
        temperature = sensor.temperature
        humidity = sensor.humidity
    except RuntimeError as error:
        # Errors happen fairly often, DHT's are hard to read, just keep going
        print(error.args[0])
        time.sleep(2.0)
```

Any Questions???