IoT Interns: MQTT Communication

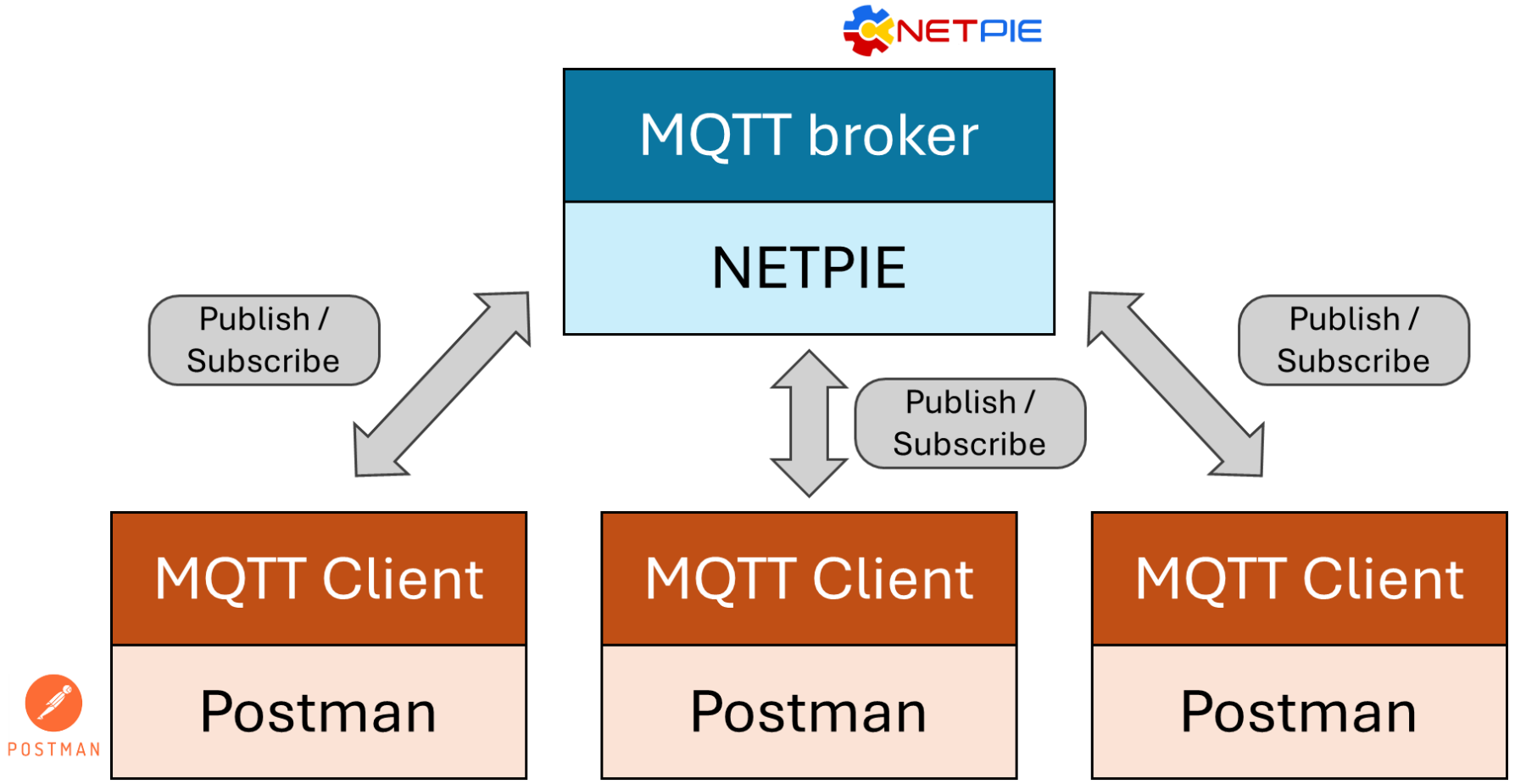
28 May 2025

While waiting…

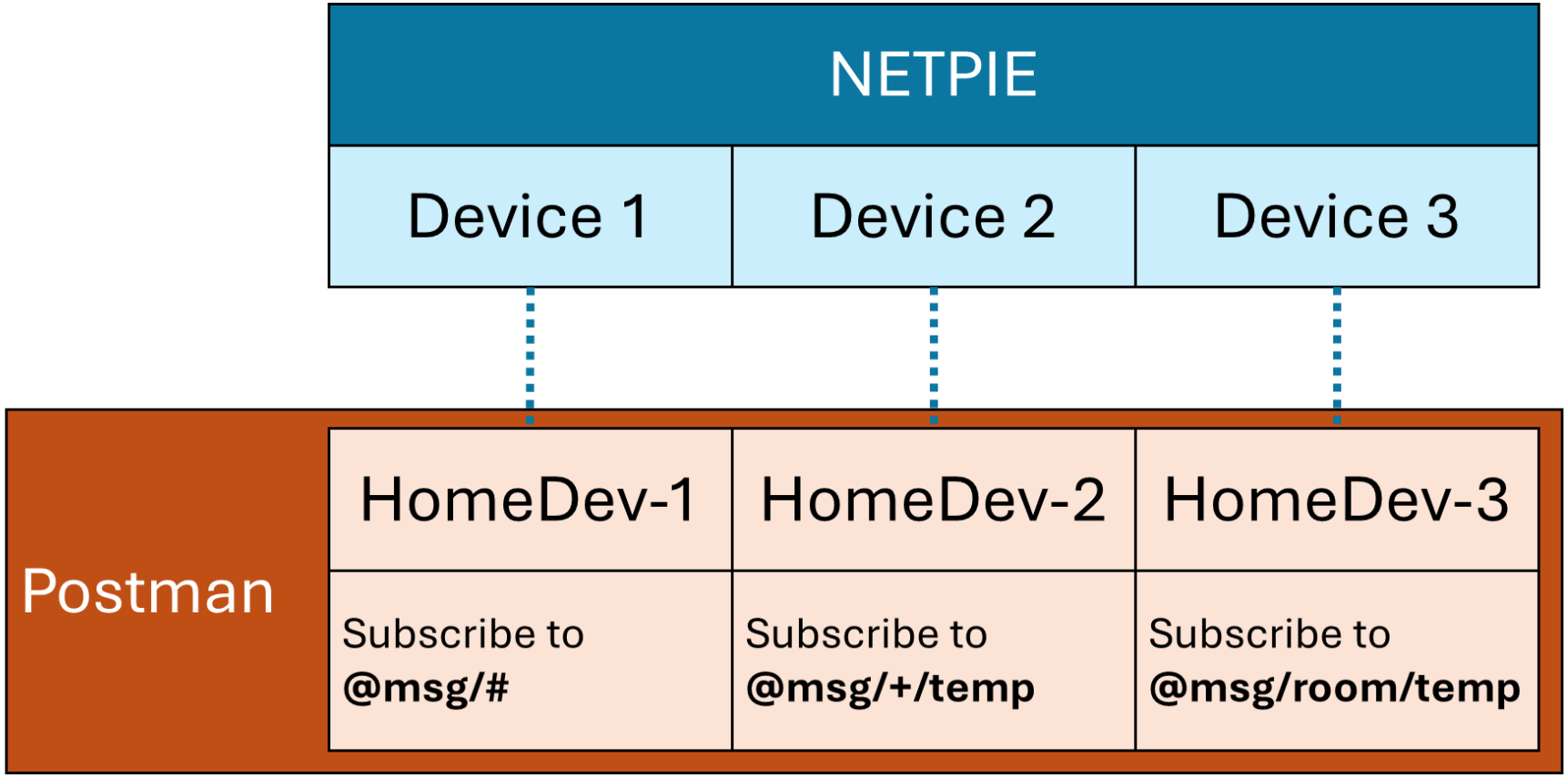
- Go to the directory that you want
- Right click -> Open in terminal
- Run the command:

git clone https://github.com/PatchapongKul/CEPT_IoT_Intern.git

MQTT Communication

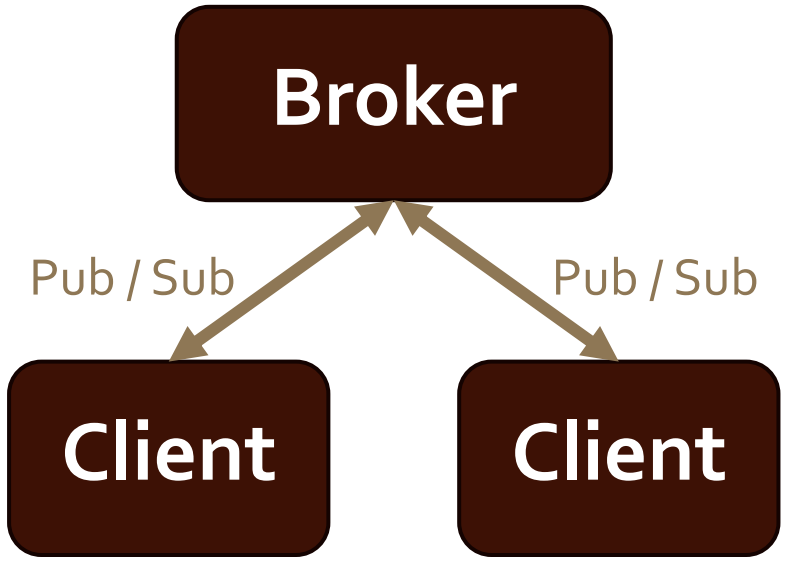


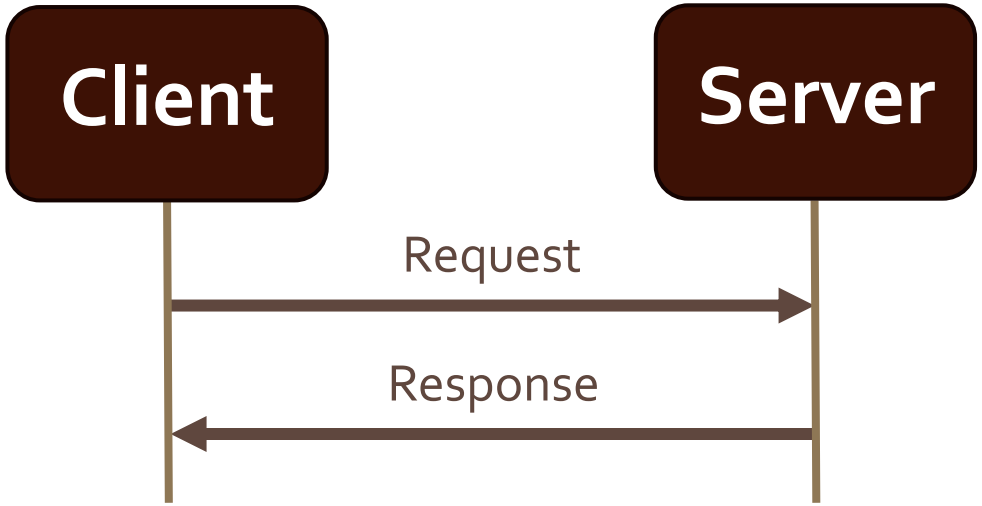
What have you found out about MQTT?

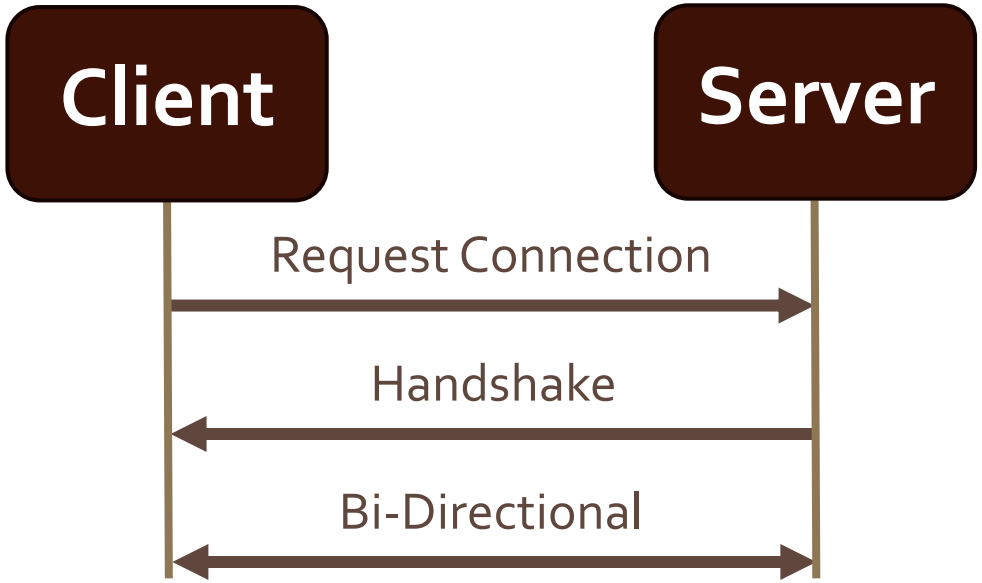


MQTT

- **Publish / Subscribe model**
  What else can it be?



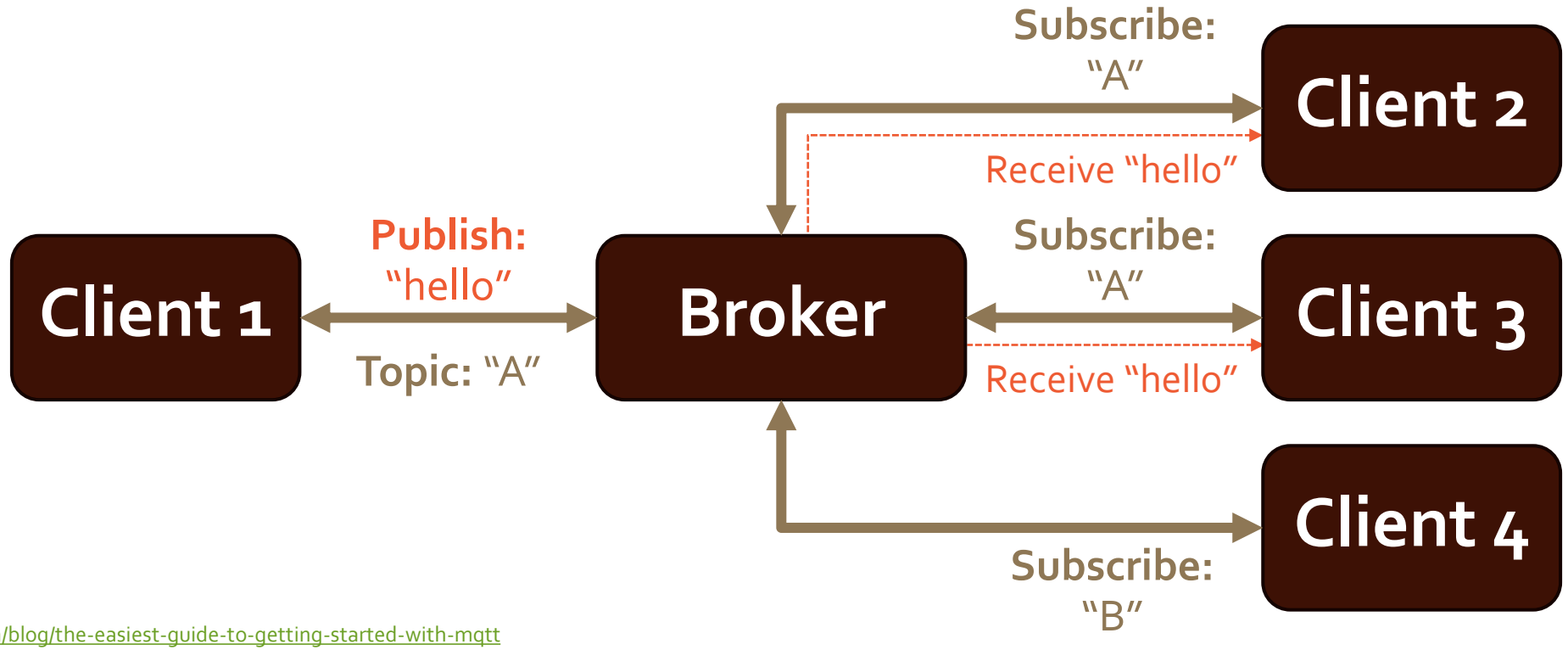




https://www.wallarm.com/what/websocket-vs-http-how-are-these-2-different

MQTT

Only subscribed clients receive the published messages



https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt

MQTT Topic

Topics are structured using a hierarchical format separated by **slashes /**. For example,

- *Room/Sensor/Temperature*
- *Room/Sensor/Humidity*
- *Room/Actuator/Light*

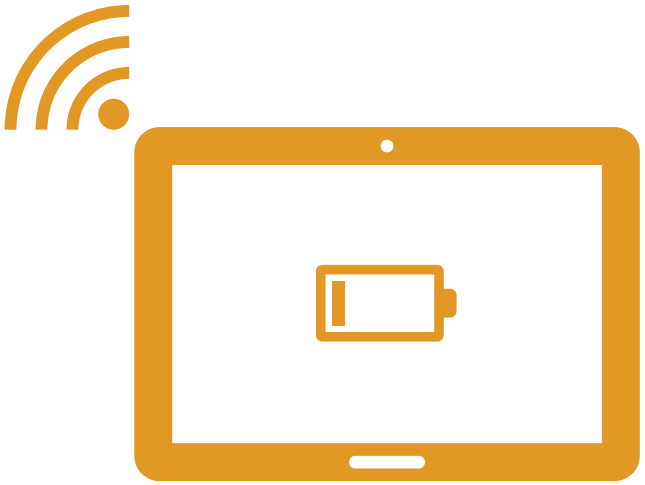
Or using wildcards:

- *Room/#* to subscribe all subtopics under room
- *Room/+* to subscribe one level under room
  (e.g., ***Room/Sensor*** but not ***Room/Sensor/Temperature***)

Why MQTT?

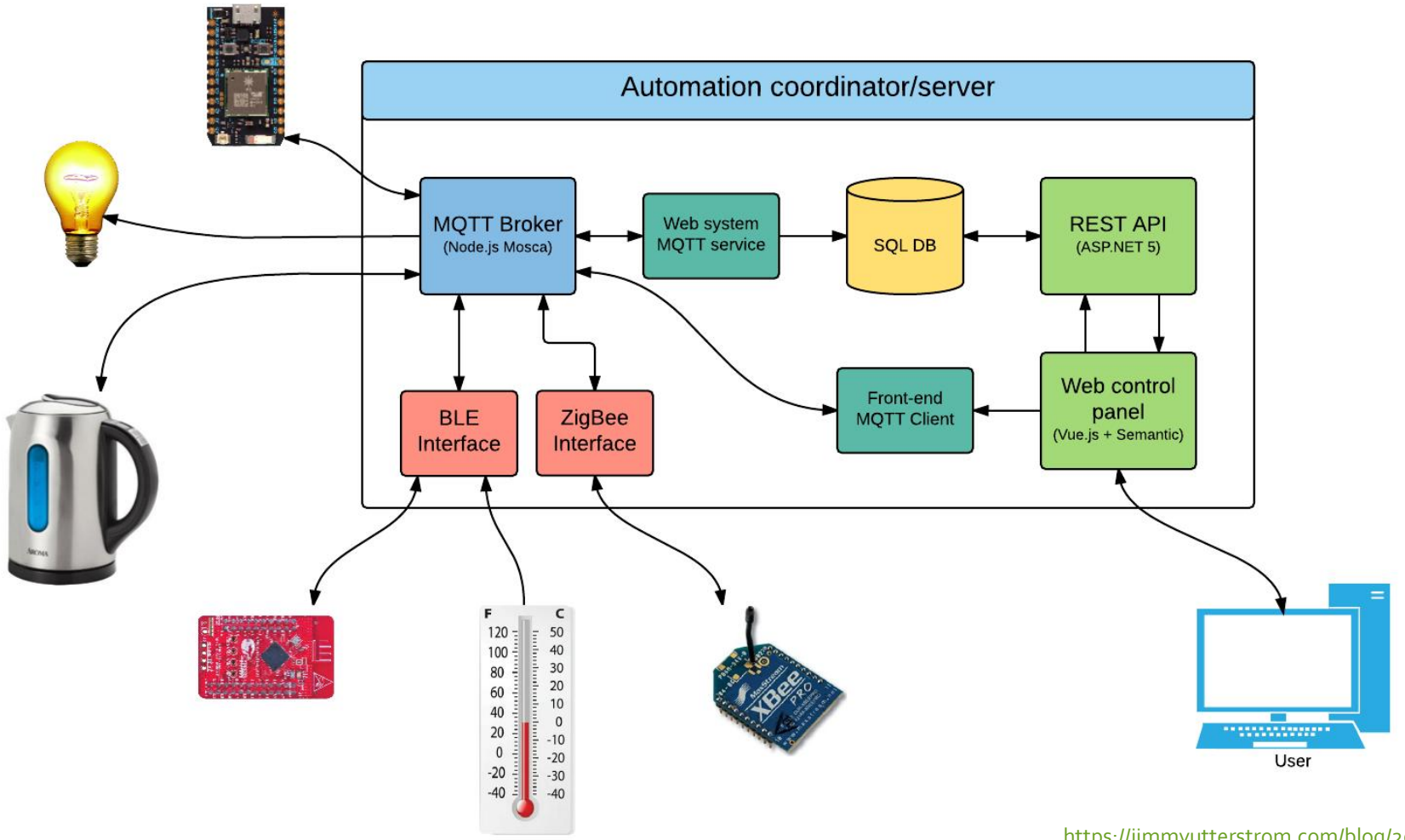
Lightweight Protocol

- Minimal protocol overhead and small packet size
- Less data to transmit and process
- Lower Power Consumption
- Good for IoT Environment



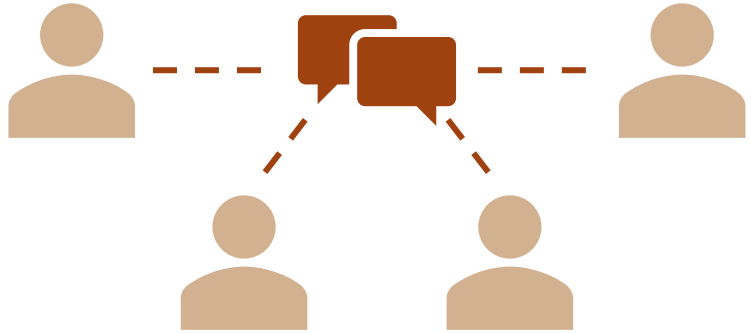
https://www.emqx.com/en/blog/the-easiest-guide-to-getting-started-with-mqtt

MQTT Example: Home Automation



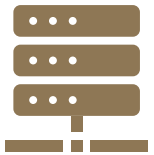
https://jimmyutterstrom.com/blog/2016/03/08/home-automation-plans/

MQTT Example: Facebook Messenger

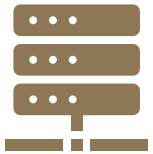


Topic Director

Broker 1



Broker 2



Broker 3

Note: Information from 2011

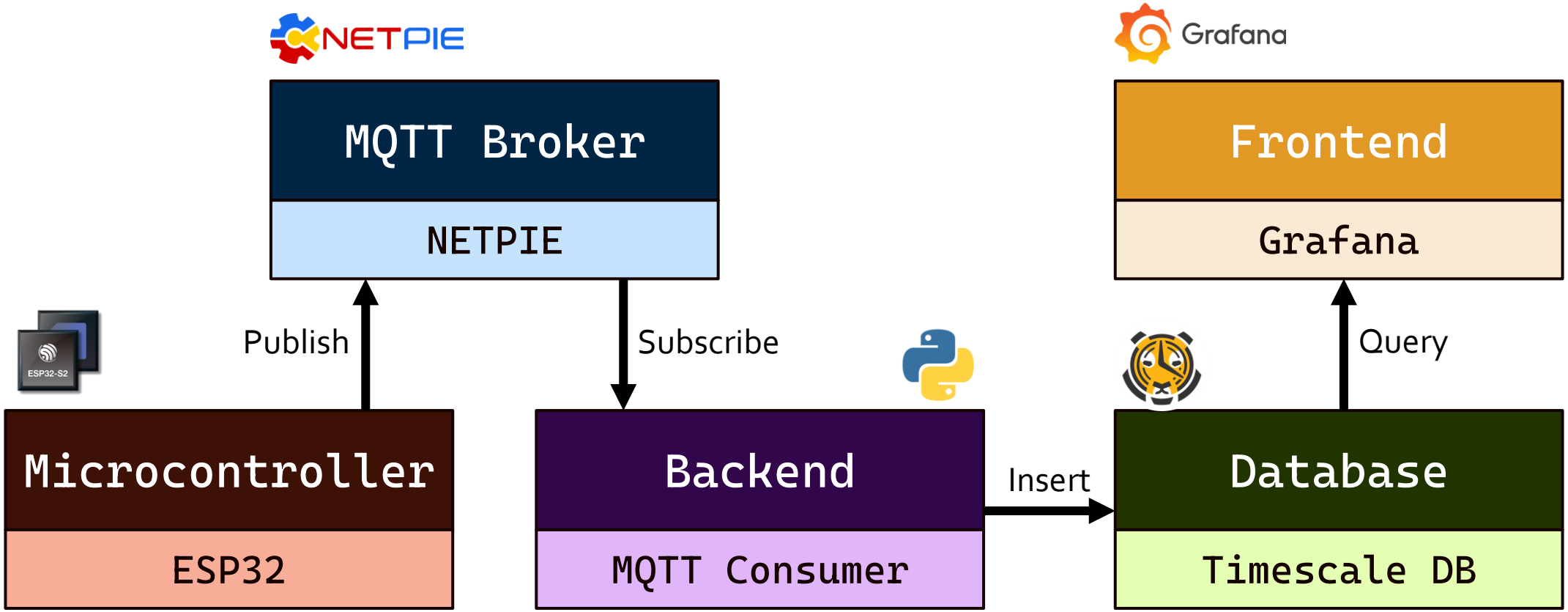
https://engineering.fb.com/2011/08/12/android/building-facebook-messenger/

https://stackoverflow.com/questions/61507894/how-does-facebook-utilize-mqtt-when-it-is-topic-based

IoT Interns: 1st Tutorial Session

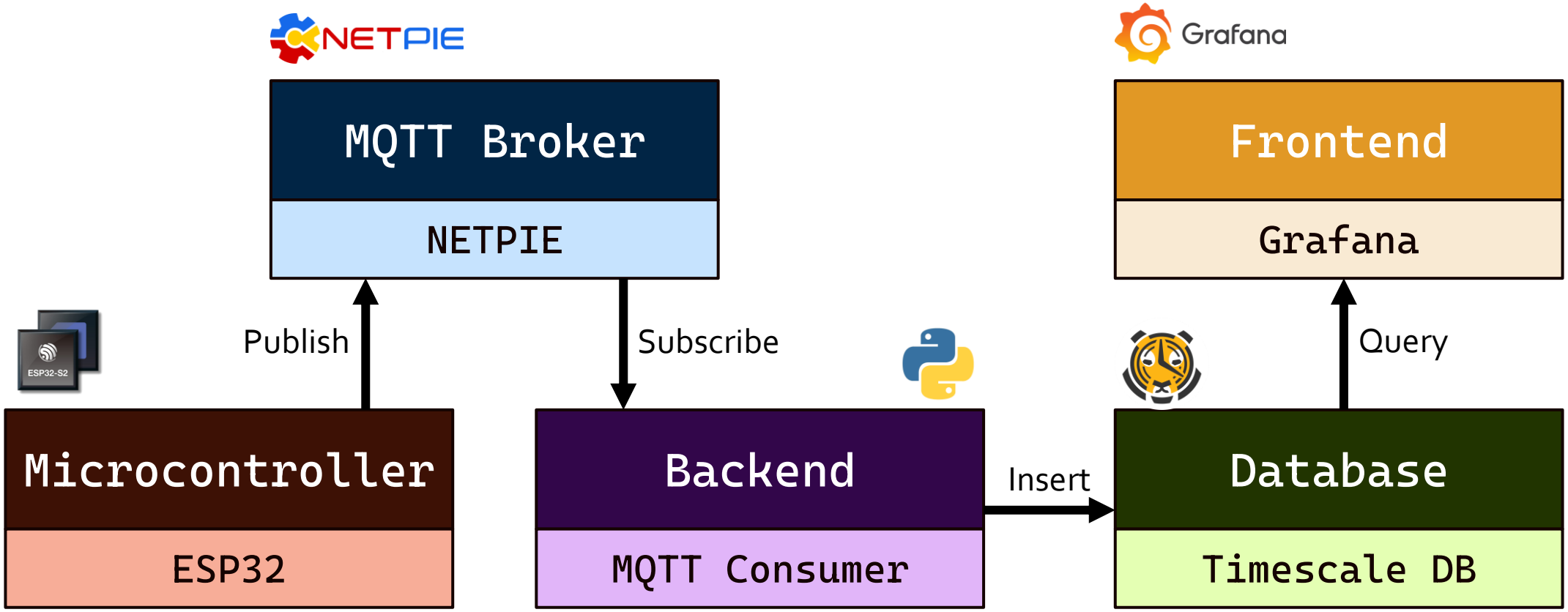
28 May 2025

System Overview



Today's objective

To gain a basic understanding of how data flows — not aiming to become an expert



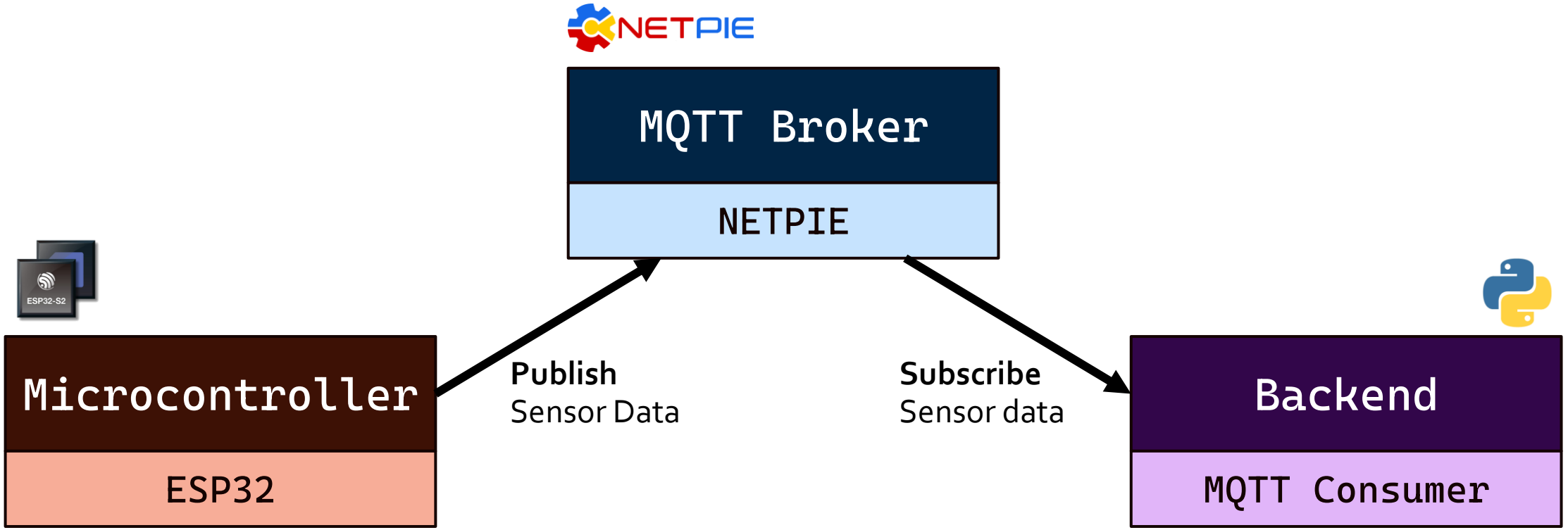
Expected Outcome

We should see the sensor data from ESP32 visualized on the dashboard

ESP32 Programming

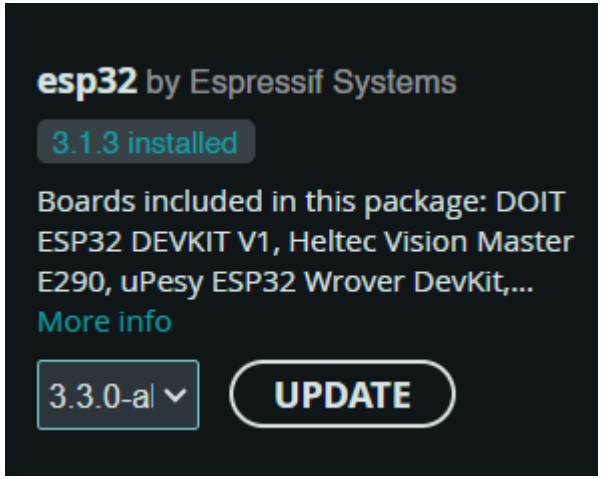
Arduino IDE

ESP32 Programming Overview

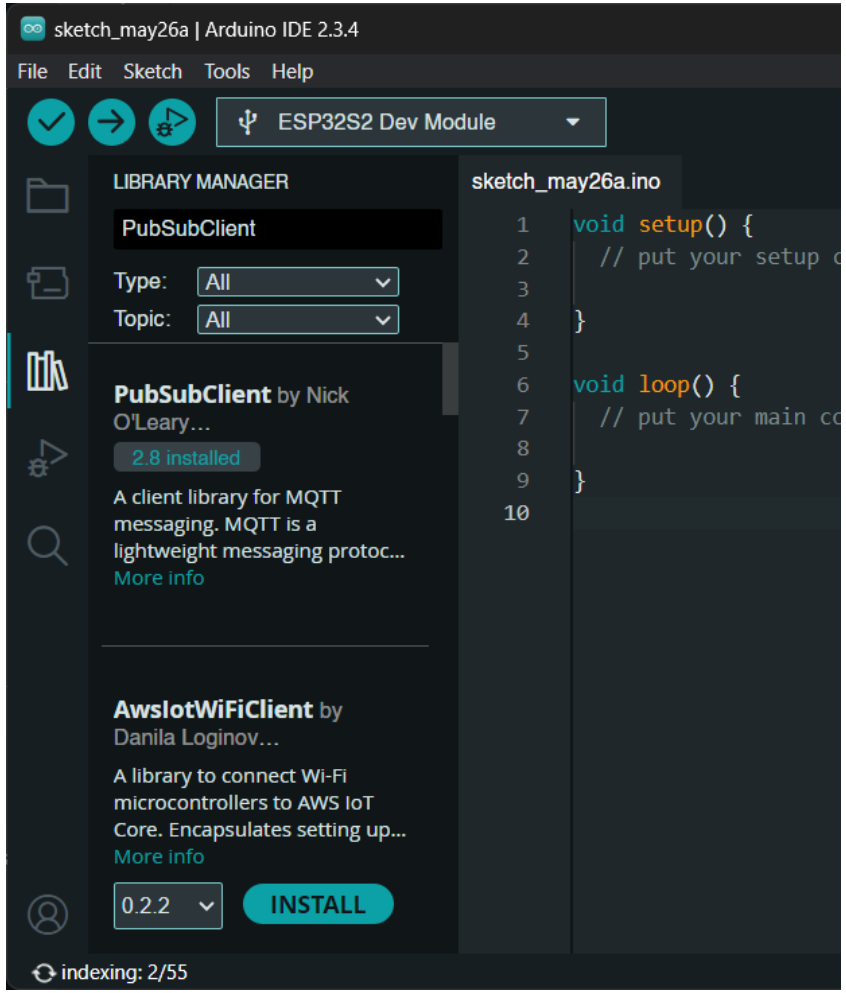


Install Arduino Core for ESP32

- Open Arduino IDE > File > Preferences
- Add this URL to Additional Board Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
- Tools > Board > Board Manager > Install ESP32

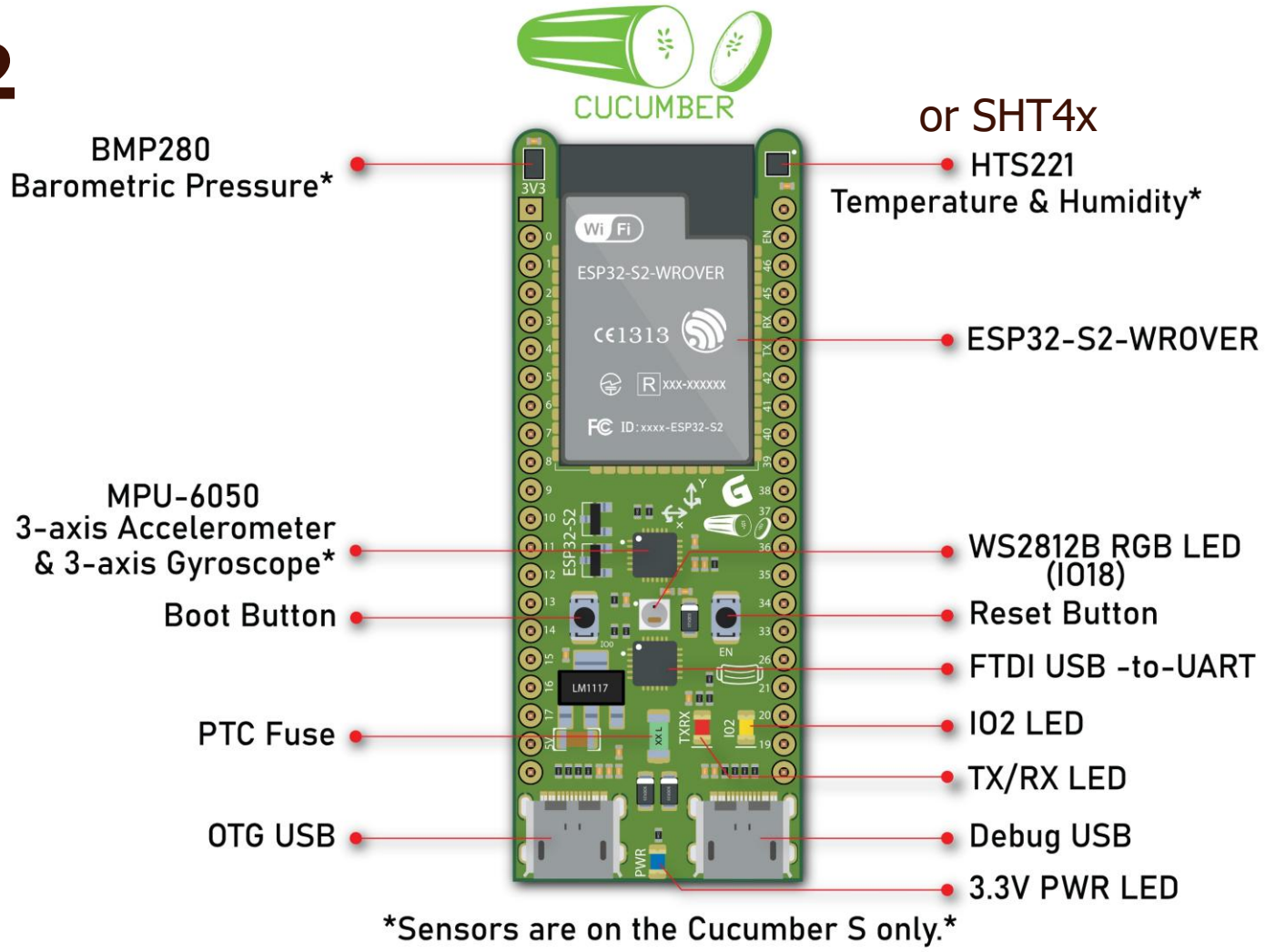


Install Required Libraries



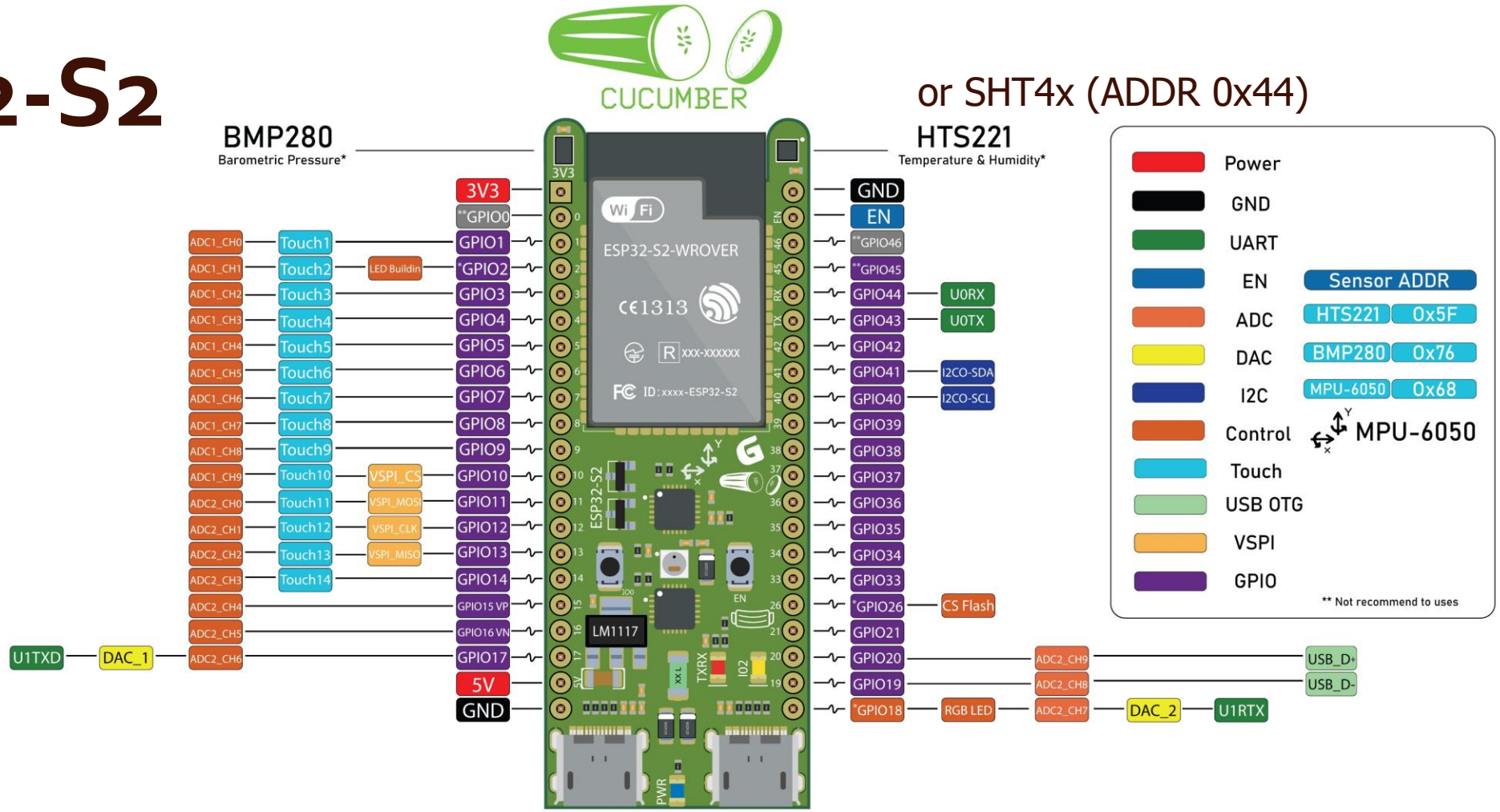
- Adafruit HTS221
- Adafruit SHT4x Library
- PubSubClient

ESP32-S2



https://www.gravitechthai.com/product/cucumber-rs-gravitech-esp32-s2-wifi-dev-board-with-sensors/11001005073001437

ESP32-S2



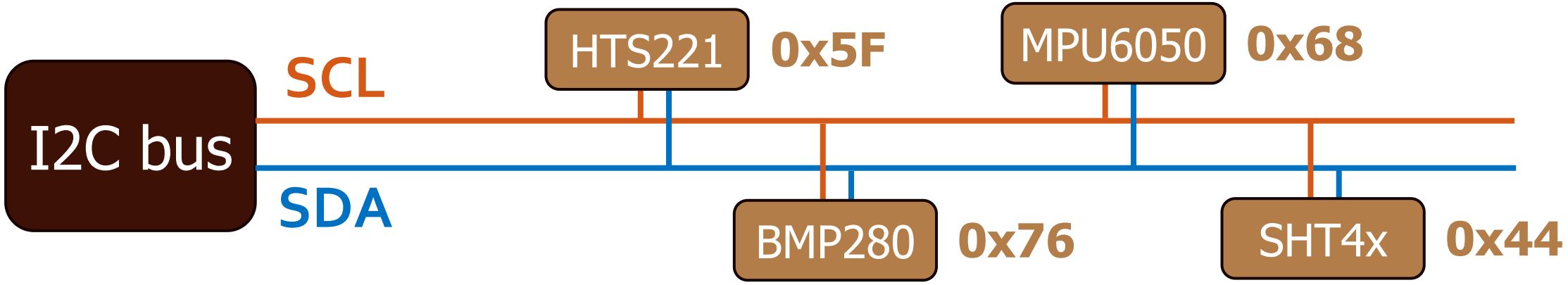
https://www.gravitechthai.com/product/cucumber-rs-gravitech-esp32-s2-wifi-dev-board-with-sensors/11001005073001437

CUCUMBER PINOUT

GRAVITECH

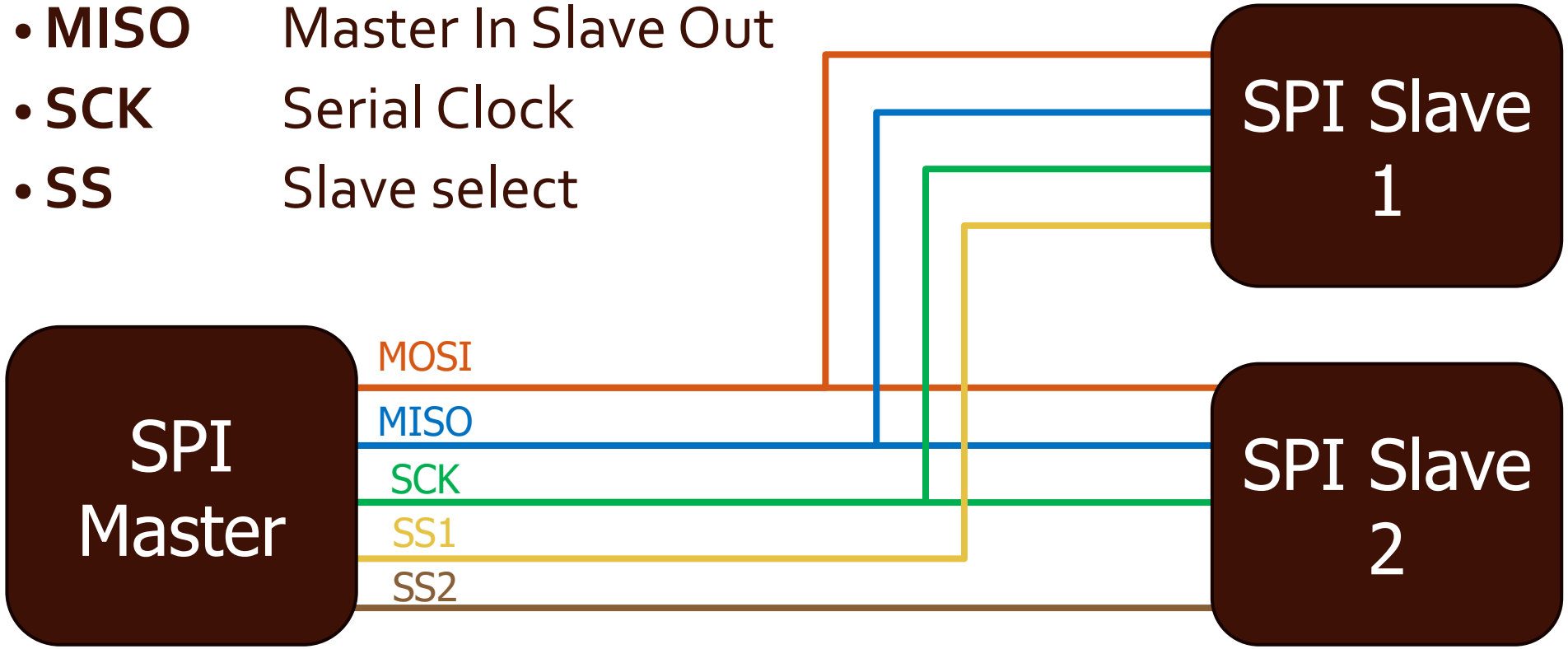
I2C (Inter-Integrated Circuit)

- **two-wire** communication protocol (half-duplex)
  - **SCL** carries the clock signal
  - **SDA** carries the data
- Support multiple devices on those two wires (only 1 master)
- Each device has a unique address

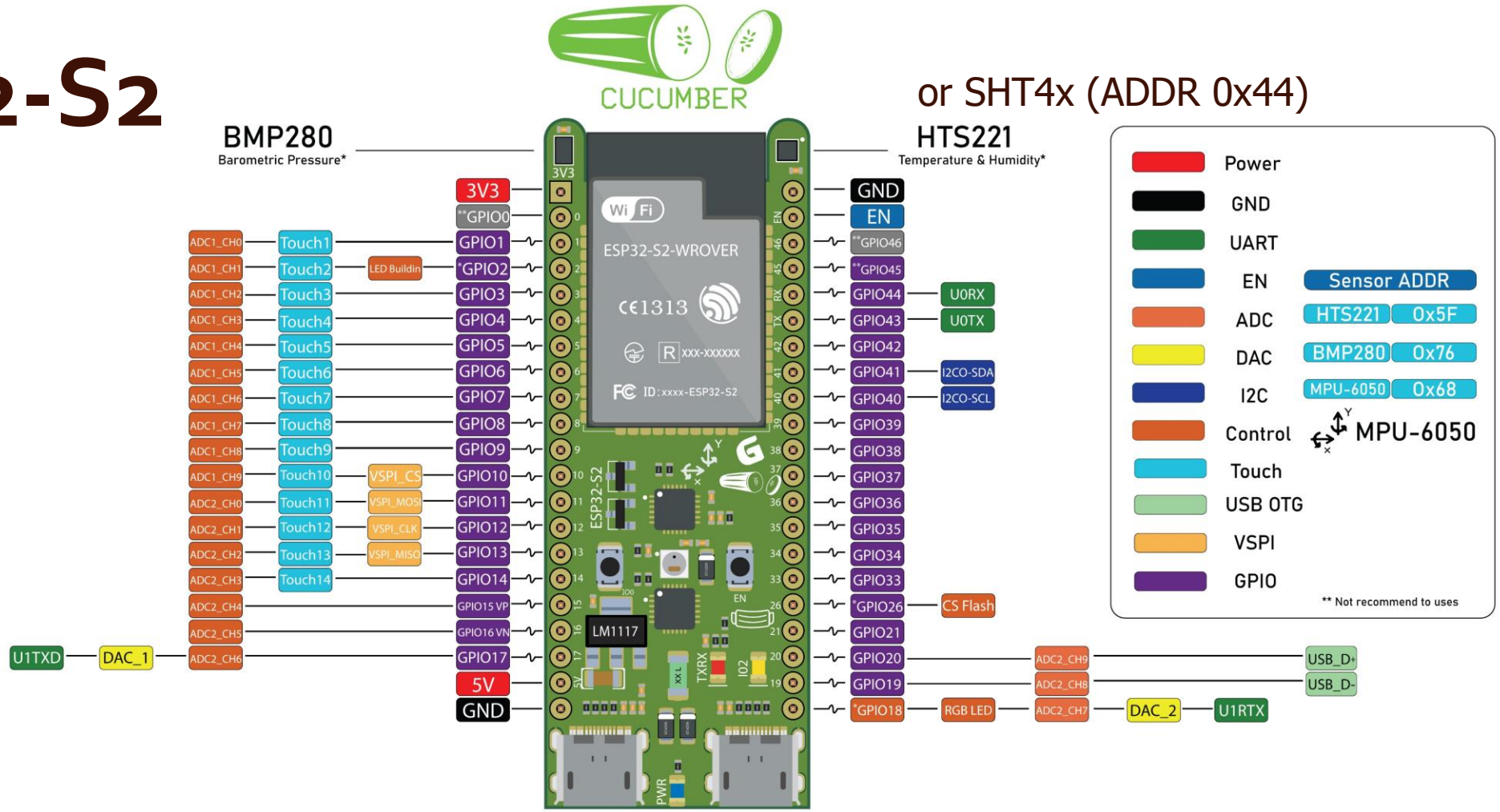


SPI (Serial Peripheral Interface)

- **four-wire** communication protocol (full-duplex)
  - **MOSI** Master Out Slave In
  - **MISO** Master In Slave Out
  - **SCK** Serial Clock
  - **SS** Slave select



ESP32-S2



Sensors are on the Cucumber S only.

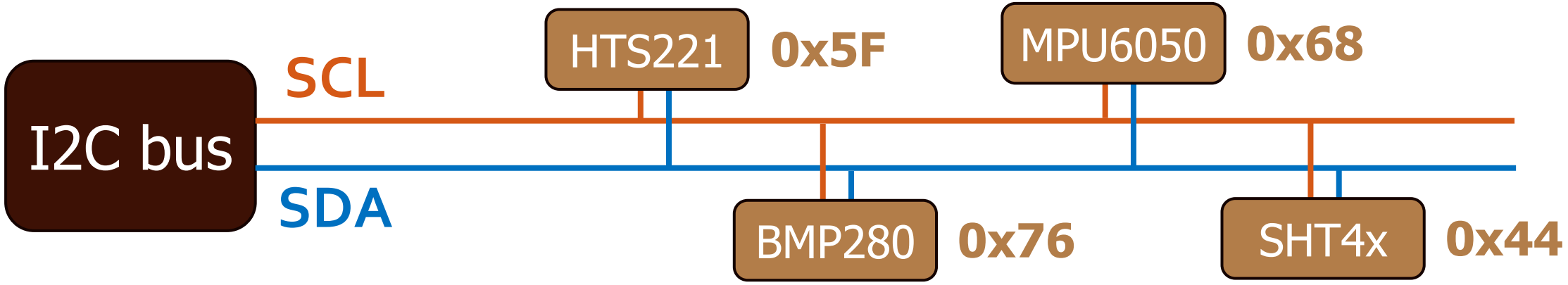
https://www.gravitechthai.com/product/cucumber-rs-gravitech-esp32-s2-wifi-dev-board-with-sensors/11001005073001437

CUCUMBER PINOUT

GRAVITECH

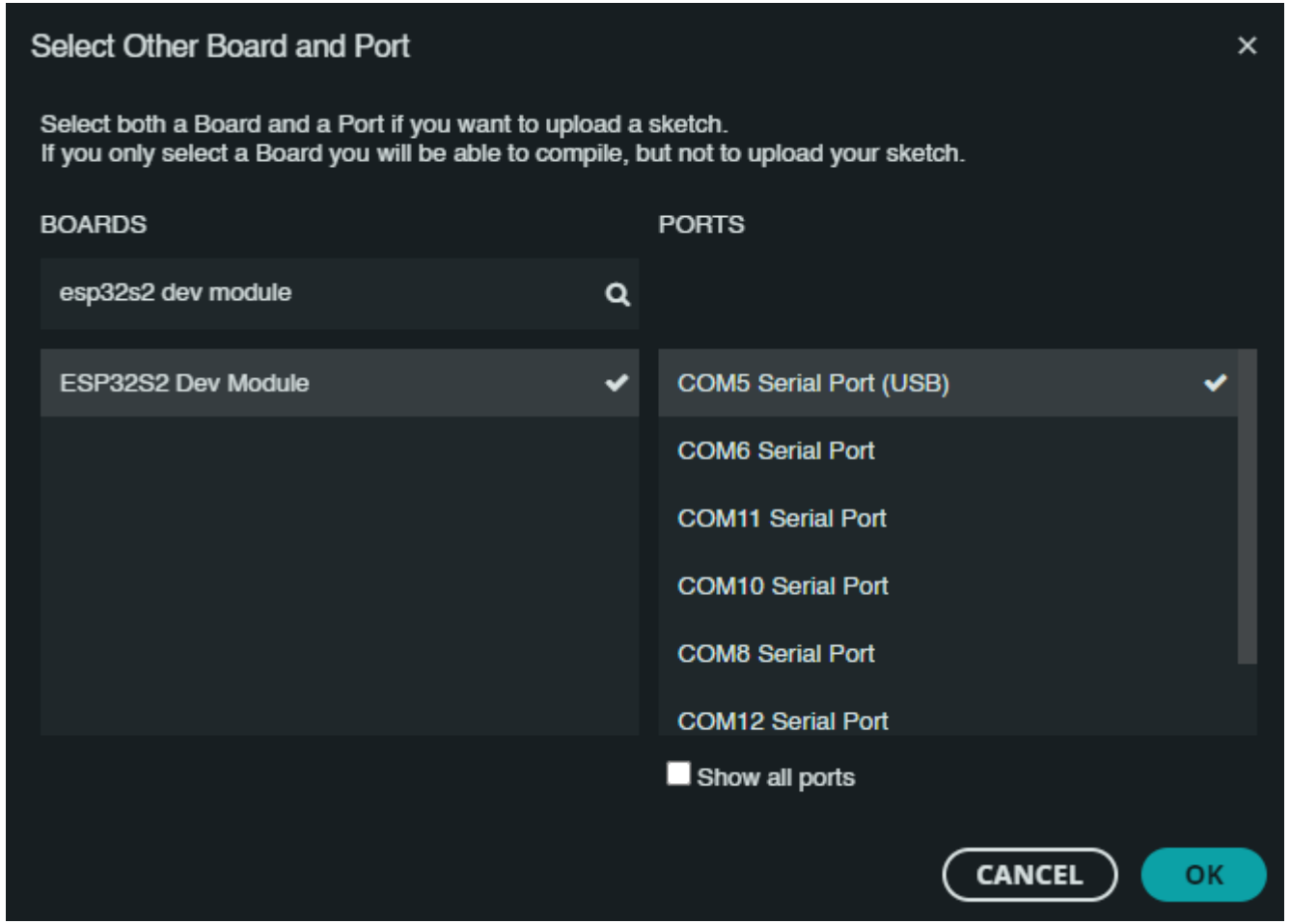
Let's check the sensors' address

- Arduino IDE > Open this file: *CEPT_IoT_Intern\Training\Tutorial-Session-1\esp32\I2C_scanner**I2C_scanner.ino***
- To scan all the I2C devices connected to ESP32



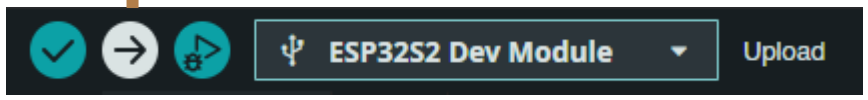
Connect board to your Laptop

- Use USB-C Data cable to connect ESP32 (debug port) with your laptop.
- It will show the detected com port
- Choose **ESP32S2 Dev Module** and **detected COM port**



I2C Scanner

Upload code to your board



I2C_scanner.ino ci.json

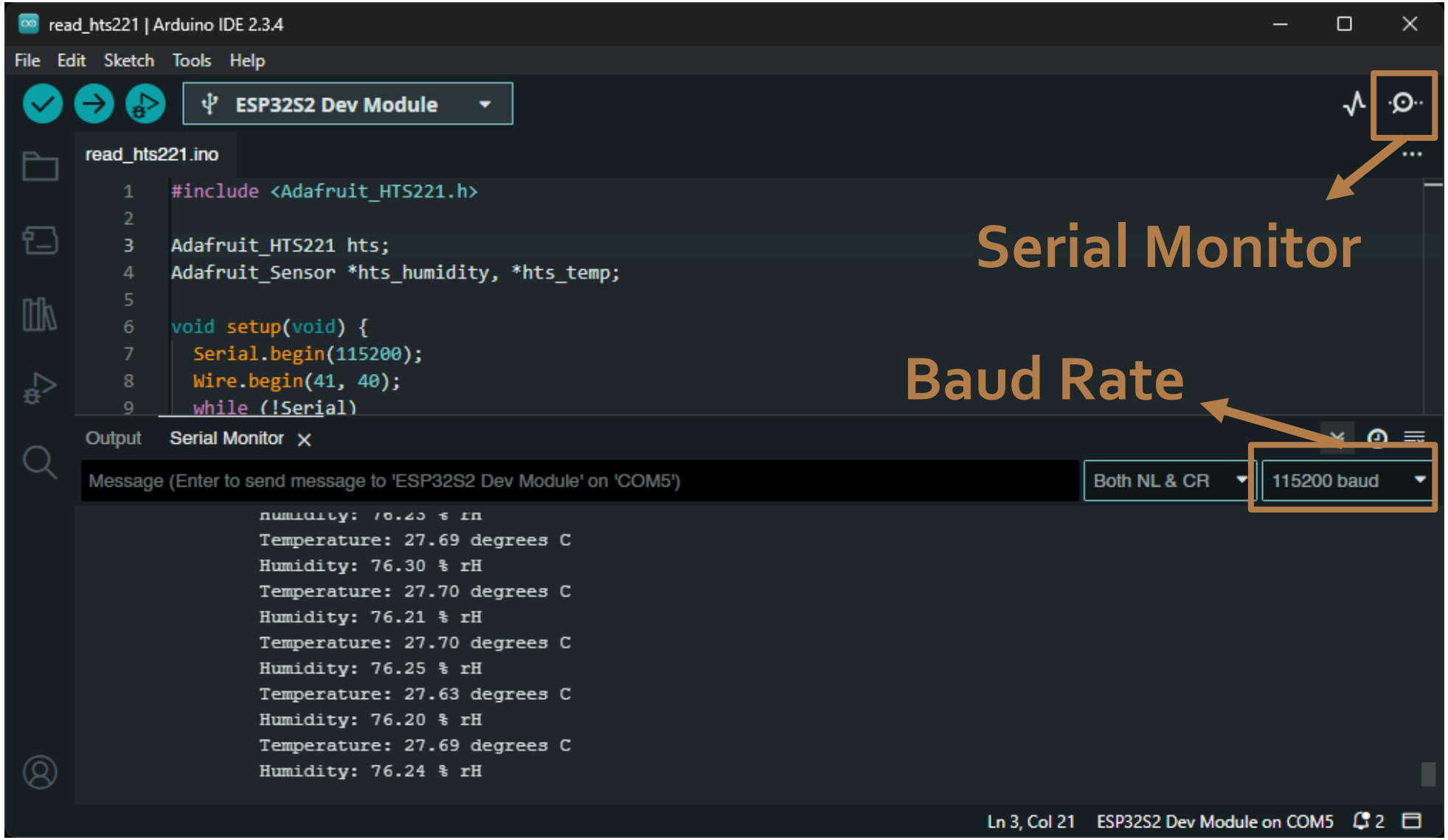
```ino
#include "Wire.h"

void setup() {
  Serial.begin(115200);
  Wire.begin(41, 40);
}

void loop() {
  byte error, address;
  int nDevices = 0;

  delay(5000);

  Serial.println("Scanning for I2C devices ...");
  for (address = 0x01; address < 0x7f; address++) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0) {
      Serial.printf("I2C device found at address 0x%02X\n", address);
      nDevices++;
    } else if (error != 2) {
      Serial.printf("Error %d at address 0x%02X\n", error, address);
    }
  }
  if (nDevices == 0) {
    Serial.println("No I2C devices found");
  }
}
```

Temperature & Humidity Sensor

- If you found **0x5F**, you go with the **HTS221** sensor
- If you found **0x44**, you go with the **SHT4x** sensor

- To read sensor value, open this file:
  *CEPT_IoT_Intern\Training\Tutorial-Session-1\esp32\read_sonsors\read_xxxx**read_xxxx.ino***

Read sensor data



Connect to Wi-Fi, MQTT, NTP

- We need **Wi-Fi** because we need internet
- We need **MQTT** because we need to transmit data
- We need **NTP** because we need the correct time

- To connect those services, open this file: *CEPT_IoT_Intern\Training\Tutorial-Session-1\esp32\wifi-mqtt-ntp**wifi-mqtt-ntp.ino***

Change credentials

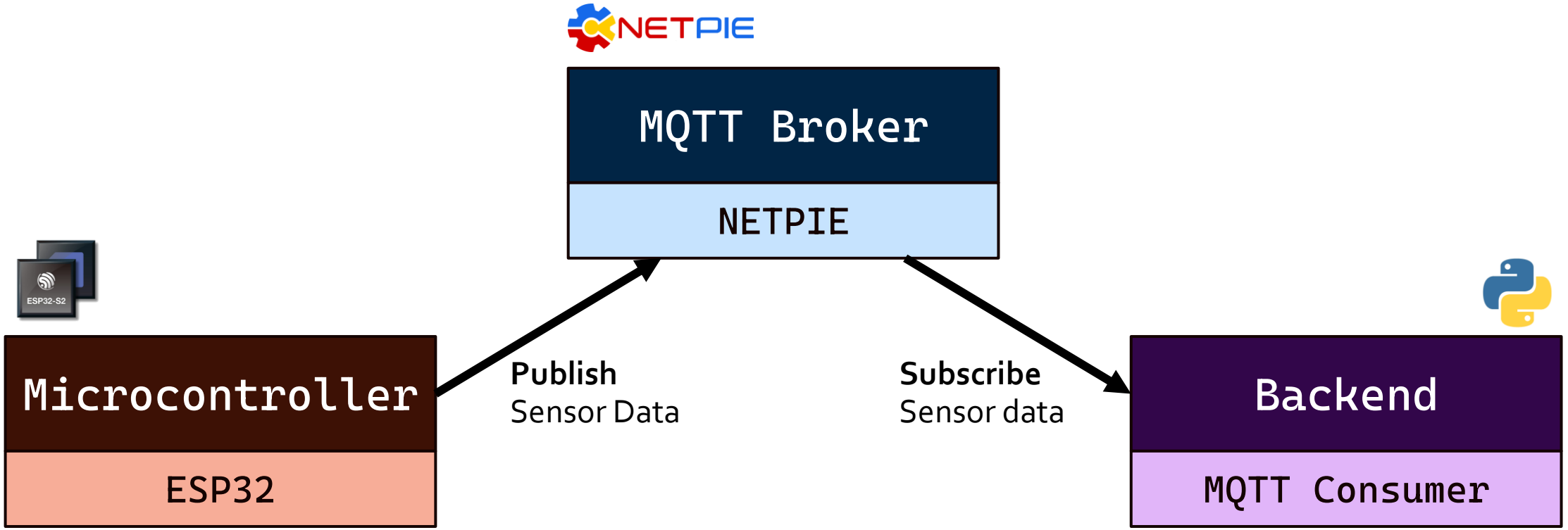
```
// WiFi Credentials
const char* ssid     = "WIFI SSID";
const char* password = "Password";

  // MQTT Broker
const char* mqtt_server = "broker.netpie.io";
const int   mqtt_port   = 1883;
const char* mqtt_client = "REPLACE_WITH_YOUR_CLIENT_ID"; // Replace with your NETPIE clientID
const char* mqtt_user   = "REPLACE_WITH_YOUR_TOKEN"; // Replace with your NETPIE Token
const char* mqtt_pass   = "REPLACE_WITH_YOUR_SECRET"; // Replace with your NETPIE Secret
// MQTT Topic
const char* mqtt_topic  = "@msg/sensor";
```

How to verify if MQTT is connected?

ESP32 continuously sends the generated data to NETPIE

Try receiving data from NETPIE!

ESP32 Programming Overview



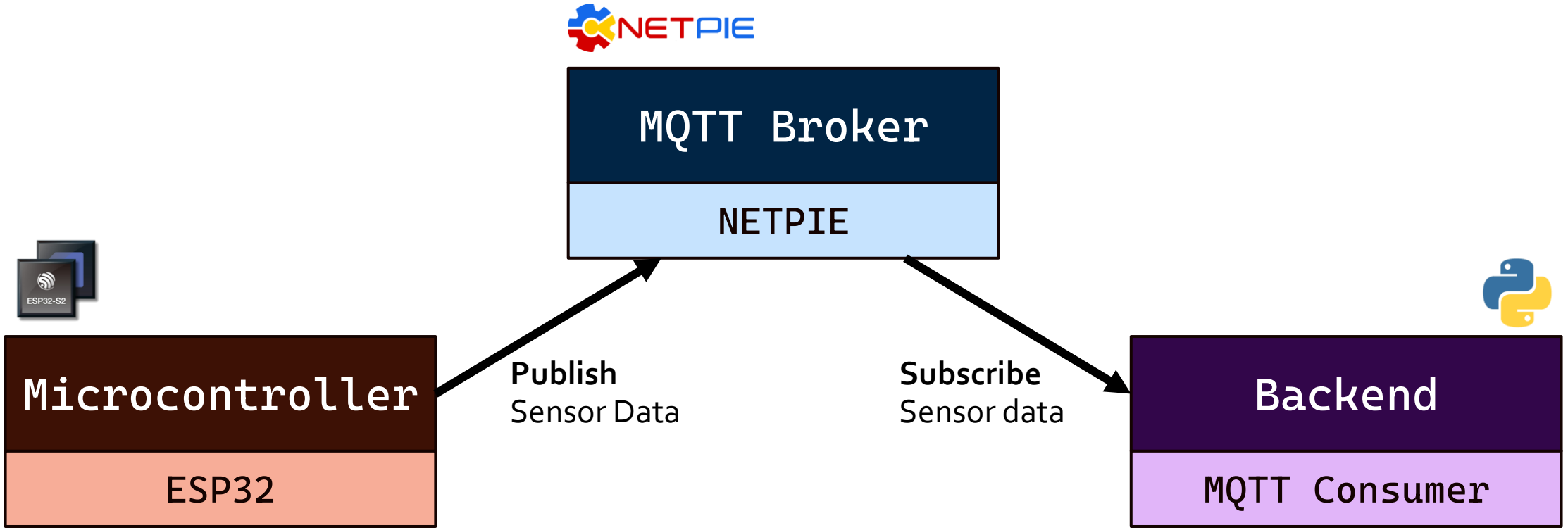
Your turn!

- Now you can read the sensor data
- Also, you can connect to NETPIE and send the generated data

Then, try sending the sensor data to NETPIE and check if it was sent successfully using another MQTT client

You have time until 10:25

ESP32 Programming Overview



How about using python as MQTT client?

- pip install paho-mqtt
- Go to file:
  *CEPT_IoT_Intern\Training\Tutorial-Session-1\esp32**subscribe_mqtt.py***

```
# Replace these with your credentials
CLIENT_ID = "REPLACE_WITH_YOUR_CLIENT_ID"; # Replace with your NETPIE client ID
TOKEN  = "REPLACE_WITH_YOUR_USERNAME"      # Replace with your NETPIE token
SECRET = "REPLACE_WITH_YOUR_PASSWORD";     # Replace with your NETPIE secret
```

Run the MQTT Client Python Script

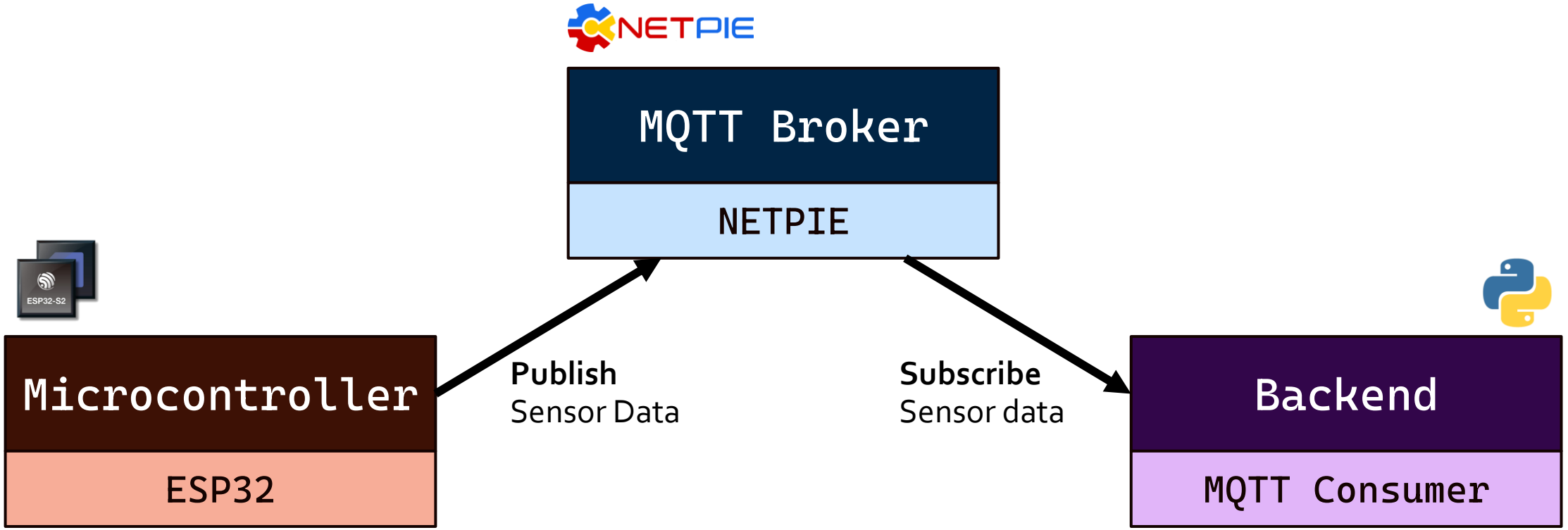
To run a Python script from the terminal:

> *python [file path]*

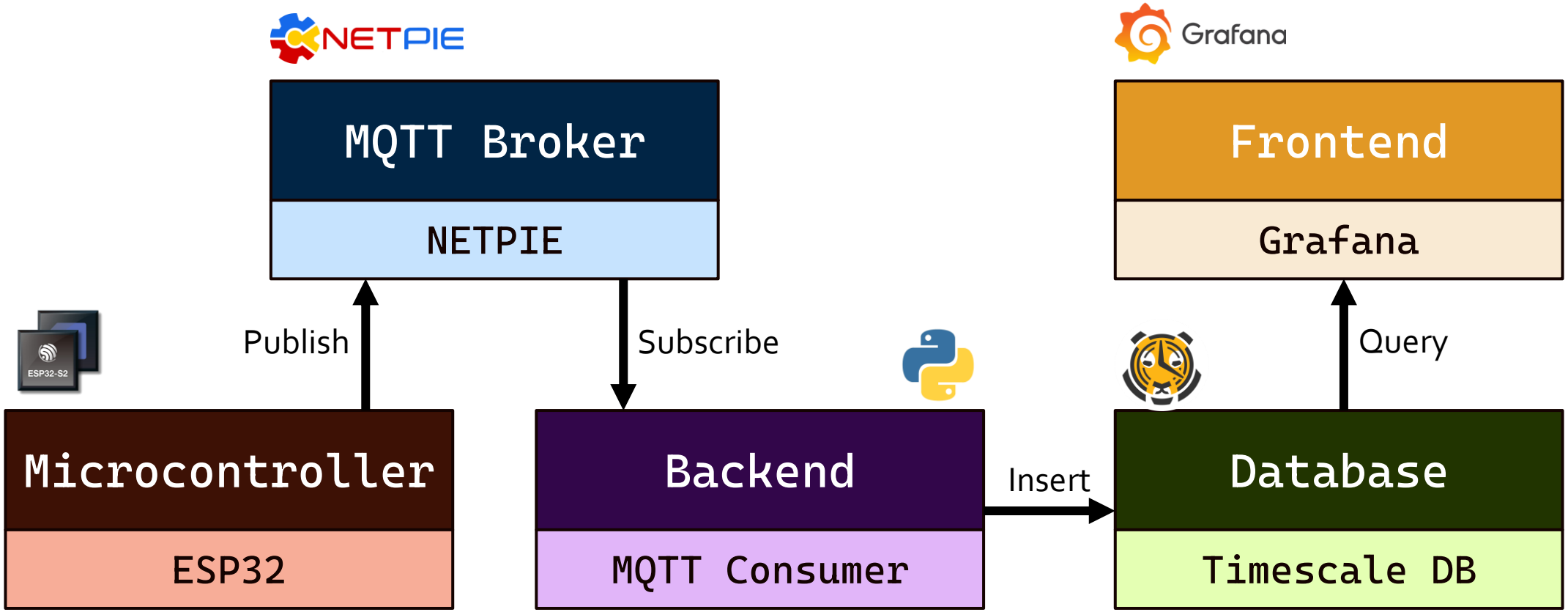
> *Example: python Training\Tutorial-Session-1\esp32\subscribe_mqtt.py*

Keep sending data from ESP32 and verify the incoming message from the subscribed topic!

ESP32 Programming Overview

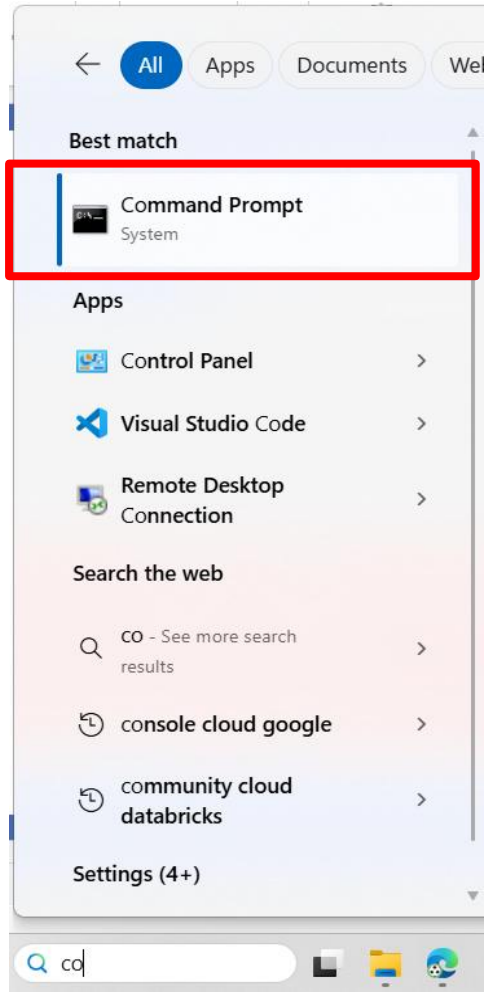


After Break!


NETPIE
Grafana
MQTT Broker
NETPIE
Frontend
Grafana
Publish
Subscribe
Query
Microcontroller
ESP32
Backend
MQTT Consumer
Insert
Database
Timescale DB

DURING BREAK

Install images and library for next section



Open command prompt and run following command one by one

```
docker pull timescale/timescaledb:2.20.0-pg17

docker pull grafana/grafana-oss

pip install psycopg2
```

10-MINUTE BREAK

SEE YOU AT .....