

GUIDELINE FOR RND PROJECT

1. Docker

Link: <https://www.docker.com/>

- This is an open platform for developing, shipping, and running applications.
- **Step 1:** git clone <https://github.com/questdb/questdb-mock-power-sensor-mock-sensor>
- **Step 2:** docker network create tutorial

2. MQTT Eclipse

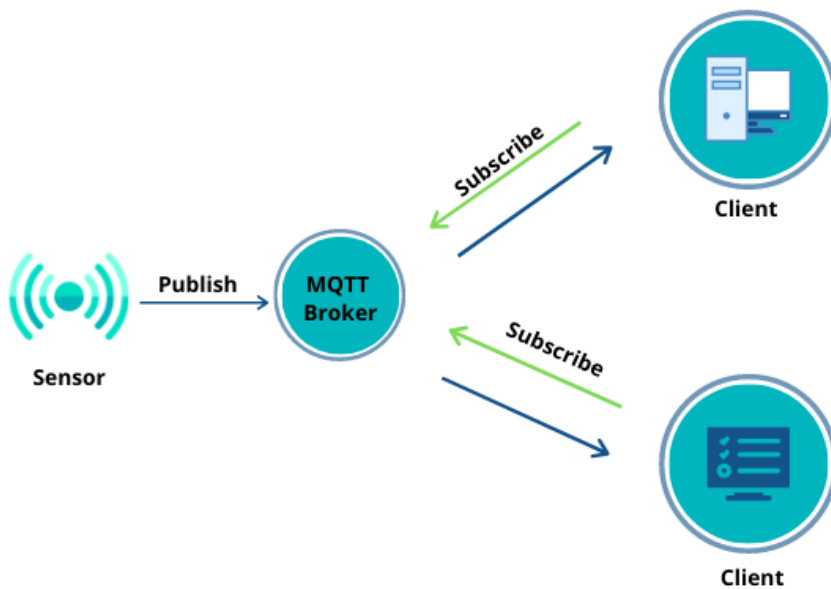
Link: <https://mosquitto.org/download/> (for both window and mac)

<https://mosquitto.org/download/#:~:text=mosquitto%2D2.0.20%2Dinstall%2Dwindows%2Dx64.exe>

- **Step 3:** Override the mosquitto file by conf/mosquitto/mosquitto.conf to fix unauthenticated clients allowed by default config for Mosquitto.
- **Step 4:** after setting up, dashboard appear: Docker engine stopped, then go to:
C:\Users\<username>\AppData\Roaming\Docker\settings.json, and set "wslEngineEnabled": true (or just restart the application)
- **Step 5: Start MQTT Eclipse using Docker**
- docker run --rm -dit -p 1883:1883 -p 9001:9001 ^-v "%cd%\conf\mosquitto\mosquitto.conf\mosquitto.conf:/mosquitto/config/mosquitto.conf" ^--network=tutorial --name=mosquitto eclipse-mosquitto (for window)
- docker run --rm -dit -p 1883:1883 -p 9001:9001 \ -v "\$ (pwd)"/conf/mosquitto/mosquitto.conf:/mosquitto/config/mosquitto

.conf \ --network=tutorial --name=mosquitto eclipse-mosquitto (for mac)

⇒ "%cd%\conf\mosquitto\mosquitto.conf\mosquitto.conf:/mosquitto/config/mosquitto.conf" is the link contain file conf we cloned before



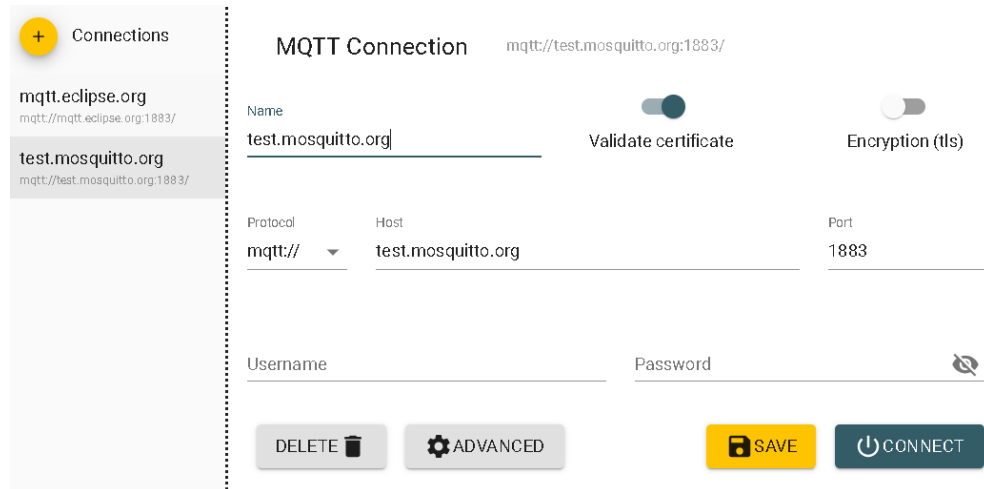
Simple communication

3. MQTT explorer

Link: <https://mqtt-explorer.com/>

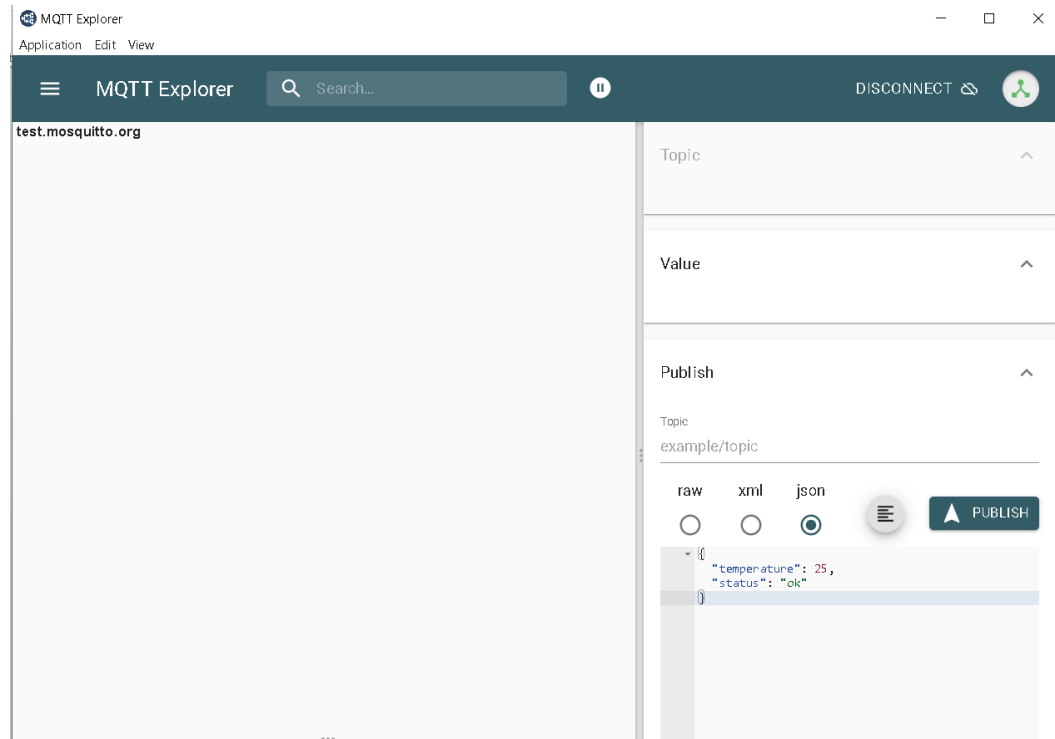
- This is a comprehensive MQTT client.
- Working with devices/services on broker.

⇒ **This is a tool for interacting with MQTT broker(mosquitto eclipse).**
In other terms, it allows user to message, monitor message traffic, visualize the mqtt's structure, publish and subscribe to messages.



The image shows the MQTT Explorer connection configuration dashboard. On the left, there is a sidebar with a yellow '+' icon and the text 'Connections'. Below this, two connection profiles are listed: 'mqtt.eclipse.org' and 'test.mosquitto.org'. The 'test.mosquitto.org' profile is selected and highlighted. The main area is titled 'MQTT Connection' and shows the URL 'mqtt://test.mosquitto.org:1883/'. Below this, there are fields for 'Name' (test.mosquitto.org), 'Protocol' (mqtt://), 'Host' (test.mosquitto.org), and 'Port' (1883). There are also toggle switches for 'Validate certificate' and 'Encryption (tls)'. At the bottom, there are buttons for 'DELETE', 'ADVANCED', 'SAVE', and 'CONNECT'.

MQTT Explorer connection Dashboard



The image shows the MQTT Explorer dashboard interface. The top bar includes the application name 'MQTT Explorer' and a search bar. Below the top bar, the main area is divided into two sections. The left section is titled 'test.mosquitto.org' and contains a large empty space for displaying messages. The right section is titled 'Publish' and contains a form for publishing messages. The form includes a 'Topic' field with the value 'example/topic', a 'Value' field with a JSON object, and a 'PUBLISH' button. The JSON object is:

```
{  "temperature": 25,  "status": "ok"}
```

MQTT Explorer Dashboard

- Set up Host address as the machine address (IPv4 address in this tutorial as my computer address)

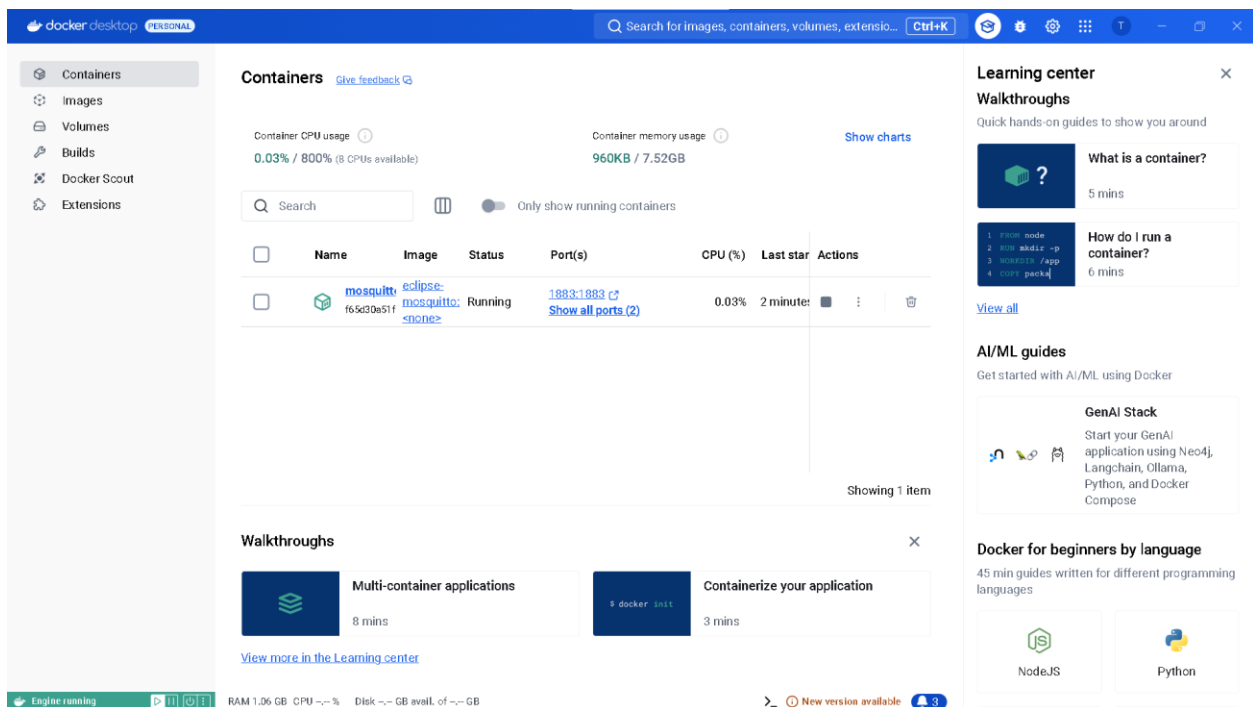
4. Go

- Link: <https://go.dev/dl/>

- This is an open-source programming language supported by Google.
- In this project, it used for continuously publishes sensor data to the MQTT broker, ensuring a stable flow of time-series data.
- ⇒ Allow QuestDB to later store and Grafana to visualize the data.
- **Step 6:** cd script

go get

go run ./main.go (Tunneling data into mosquito)



5. QuestDB

- This is a time-series database.
- Has SQL coding.
- This database is designed for horizontal scaling, enabling to distribute data and queries across multiple nodes for increased performance and availability.

- **Step 7: start QuestDB using docker**

```
C:\Users\ASUS>docker run --rm -dit -p 8812:8812 -p 9000:9000 ^ -p 9009:9009 -e QDB_PG_READONLY_USER_ENABLED=true ^ --network=tutorial --name=questdb questdb/questdb:latest
Unable to find image 'questdb/questdb:latest' locally
latest: Pulling from questdb/questdb
a480a496ba95: Pull complete
a54db233e77d: Pull complete
668fc9ea515d: Pull complete
21adc6b64d4f: Pull complete
67b45f7aa4a0: Pull complete
4f4fb700ef54: Pull complete
d37538f879bf: Pull complete
Digest: sha256:8b74dfae9ca40e5e7d746ea9bf5f092c47e1caf713819e44316ea8162eef793
Status: Downloaded newer image for questdb/questdb:latest
e5121a496abca27a9c164c03c0dbf33aa2f3ef1d7936861a9ca087a80aa867254
```

Command Prompt















- docker run --rm -dit -p 8812:8812 -p 9000:9000 ^ -p 9009:9009 -e QDB_PG_READONLY_USER_ENABLED=true ^ --network=tutorial --name=questdb questdb/questdb **(for window)**
- docker run --rm -dit -p 8812:8812 -p 9000:9000 \ -p 9009:9009 -e QDB_PG_READONLY_USER_ENABLED=true \ --network=tutorial --name=questdb questdb/questdb **(for mac)**
- Check on <http://localhost:9000/> to access **questDB** platform

6. Telegraf

- Telegraf transfers data between Mosquitto and QuestDB using **QuestDB's ILP (Influx Line Protocol)** interface(this can handle large volumes of data sent from telegraf).
- **Step 8: Start telegraf container using Docker**
- docker run --rm -it ^ -v C:\Users\ASUS\mock-sensor\conf\telegraf\telegraf.conf:/etc/telegraf/telegraf.conf ^ --network=tutorial --name=telegraf telegraf **(for window)**
- docker run --rm -it \ -v "\$ (pwd)"/conf/telegraf/telegraf.conf:/etc/telegraf/telegraf.conf \ --network=tutorial --name=telegraf telegraf**(for mac)**

```
C:\Users\ASUS>docker run --rm -it ^ -v C:\Users\ASUS\mock-sensor\conf\telegraf\telegraf.conf:/etc/telegraf/telegraf.conf ^ --network=tutorial --name=telegraf telegraf
2024-10-26T18:31:38Z I! Loading config: /etc/telegraf/telegraf.conf
2024-10-26T18:31:38Z I! Starting Telegraf 1.32.1 brought to you by InfluxData the makers of InfluxDB
2024-10-26T18:31:38Z I! Available plugins: 235 inputs, 9 aggregators, 32 processors, 26 parsers, 62 outputs, 6 secret-stores
2024-10-26T18:31:38Z I! Loaded inputs: mqtt_consumer
2024-10-26T18:31:38Z I! Loaded aggregators:
2024-10-26T18:31:38Z I! Loaded processors:
2024-10-26T18:31:38Z I! Loaded secretstores:
2024-10-26T18:31:38Z I! Loaded outputs: socket_writer
2024-10-26T18:31:38Z I! Tags enabled: host=df9123bb9ddc
2024-10-26T18:31:38Z I! [agent] Config: Interval:1s, Quiet:false, Hostname:"df9123bb9ddc", Flush Interval:10s
2024-10-26T18:31:38Z I! [inputs.mqtt_consumer] Connected [tcp://mosquitto:1883]
```

Command Prompt

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last star	Actions
<input type="checkbox"/>	 mosquitto f65d30a51f	eclipse-mosquitto <none>	Running	1883:1883  Show all ports (2)	0.03%	1 hour ago	  
<input type="checkbox"/>	 questdb e5121a496d	questdb/questdb <none>	Running	8812:8812  Show all ports (3)	21.55%	36 minutes	  
<input type="checkbox"/>	 telegraf df9123bb9dc	telegraf <none>	Running		0.06%	1 minute	  

Docker Dashboard

- Now the sensor data is automatically tunneled into QuestDB.

```
# Configuration for Telegraf agent
[agent]
  ## Default data collection interval for all inputs
  interval = "1s"

[[inputs.mqtt_consumer]]
  servers = ["tcp://mosquitto:1883"]
  topics = ["sensor"]
  data_format = "json"
  client_id = "telegraf"
  data_type = "string"
  tag_keys = [
    "country",
    "status"
  ]

# Write results to QuestDB
[[outputs.socket_writer]]
  # Write metrics to a local QuestDB instance over TCP
  address = "tcp://questdb:9009"
[[outputs.file]]
  files = ["stdout"]
```

- **Step 9: Change data_format to "json" in telegraf.conf file**
- ***tag_keys:** key will be added as tag

7. Grafana

- **Step 10: Run Grafana via Docker**
- `docker run --rm -dit -p 3000:3000 ^ --network=tutorial --name=grafana grafana/grafana (for window)`
- `docker run --rm -dit -p 3000:3000 \ --network=tutorial --name=grafana grafana/grafana (for mac)`
- **Step 11:** login at <http://localhost:3000/login> using **Username:** admin

Password: admin

- **Step 12:** choose Connections/data sources -> add data source -> scroll down to the end and click on find more. Search Questdb -> install -> add new data source
- **Step 13: Config questdb-questdb-datasource**
 - **Server address:** host.docker.internal
 - **Server port:** 8812
 - **Username:** user
 - **Password:** quest
 - **TLS/SSL mode:** disable

8. Publish testing

On testing, I send a json package with sensor topic from mqtt explorer as a client to mqtt eclipse broker. The data then will be sent to questDB via telegraf.

The screenshot shows the MQTT Explorer interface. At the top, the topic 'sensor' is entered. Below the topic, there are three radio buttons for message format: 'raw', 'xml', and 'json'. The 'json' radio button is selected. To the right of the radio buttons is a menu icon and a 'PUBLISH' button. Below these controls is a text area containing a JSON payload:

```
{  "temperature": 25,  "humidity": 10,  "country": "VN"}
```

JSON file

- **Step 14: Finish on dashboard, move to code and paste SQL:**

```
SELECT

    date_trunc('minute', timestamp) AS minute,

    AVG(temperature) AS avg_temperature,

    AVG(humidity) AS avg_humidity,

    AVG(uv_value) AS uv_value,

    AVG(totalBee) AS Bee

FROM mqtt_consumer

WHERE place = 'Dong Nai'

GROUP BY minute

ORDER BY minute ASC

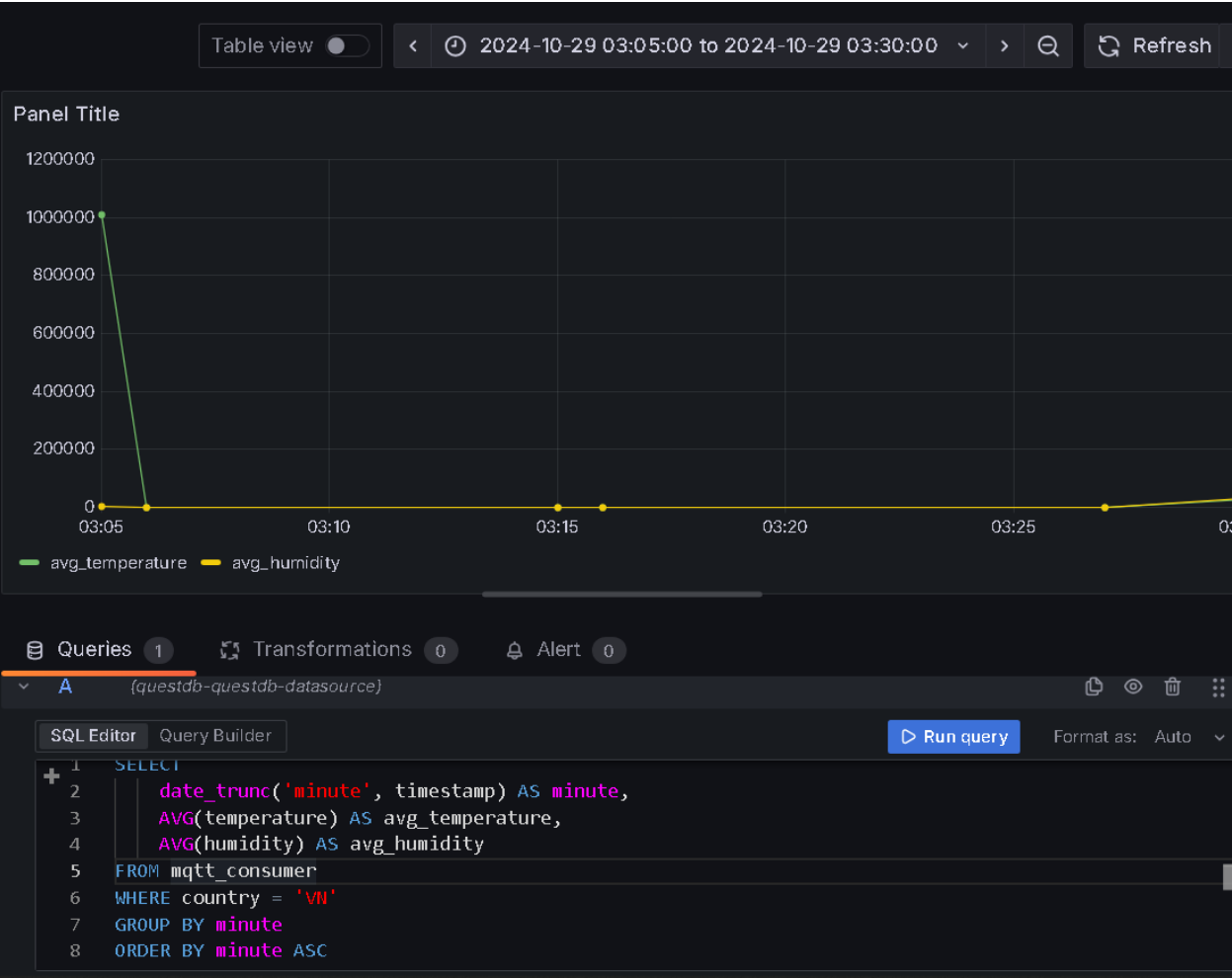
***
```

```
SELECT
    date_trunc('minute', timestamp) AS minute,
    AVG(temperature) AS avg_temperature,
    AVG(humidity) AS avg_humidity
FROM mqtt_consumer
WHERE country = 'VN'
GROUP BY minute
ORDER BY minute ASC
```

```
SELECT
avg(uv_value),
avg(uv_index),
avg(sound) as sound,
avg(weight) as weight,
avg(temperature_inside) as temperature_inside,
avg(humidity_inside) as humidity_inside,
avg(temperature_outside) as temperature_outside,
avg(humidity_outside) as humidity_outside,
date_trunc('minute', vn_time) AS minute
```

```
from "R&D_database.csv"  
WHERE place = 'Dong Nai'  
GROUP BY minute  
ORDER BY minute ASC
```

🕒 minute	📊 avg_temperature	📊 avg_humidity
2024-10-29 03:05:00	1011210	2020
2024-10-29 03:06:00	10	10
2024-10-29 03:15:00	12	12
2024-10-29 03:16:00	19.7	25
2024-10-29 03:27:00	252	325
2024-10-29 03:30:00	25200	32225
2024-10-29 03:32:00	252030	3222115



Grafana Visualization Dashboard

Increase vm.max_map_count limit:

```
Command Prompt - wsl -d docker-desktop

Microsoft Windows [Version 10.0.19045.5131]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>wsl -d docker-desktop
LAPTOP-CC9P1FJB:/tmp/docker-desktop-root/run/desktop/mnt/host/c/Users/ASUS# sysctl -w vm.max_map_count=1048576
vm.max_map_count = 1048576
LAPTOP-CC9P1FJB:/tmp/docker-desktop-root/run/desktop/mnt/host/c/Users/ASUS#
```

Edit Grafana etc/grafana.ini file on Docker:

[auth.anonymous]

Enabled = true

org_name = Main Org.

org_role = Viewer

allow_embedding = true

Web Console

SQL: `select * from mqtt_consumer`

Log [11:19:50 PM GMT+07:00] 231 rows in 15ms. Executes: 4.28ms Network: 10.72ms Total: 15ms Count: 0 Authentication: 4.8µs Compile: 0

host	topic	totalBee	temperature	humidity	place	uv_value	uv_index	timestamp
device	symbol	double	double	double	varchar	double	double	timestamp
e5e018a1faca	beehive	0	31.180000381469	68.880003051757	Dang Nai	12	0	2025-02-23T15:56:43.177205Z
e5e018a1faca	beehive	0	31.29999923786	68.199996948242	Dang Nai	12	0	2025-02-23T15:56:49.189686Z
e5e018a1faca	beehive	0	31.39999961853	68.199996948242	Dang Nai	12	0	2025-02-23T15:56:55.200497Z
e5e018a1faca	beehive	0	31.29999923786	68.199996948242	Dang Nai	12	0	2025-02-23T15:57:01.212343Z
e5e018a1faca	beehive	0	31.29999923786	68.380003051757	Dang Nai	12	0	2025-02-23T15:57:07.224937Z
e5e018a1faca	beehive	0	31.29999923786	68.480001525878	Dang Nai	5	0	2025-02-23T15:57:13.237649Z
e5e018a1faca	beehive	1	31.39999961853	68.480001525878	Dang Nai	14	0	2025-02-23T15:57:19.251899Z
e5e018a1faca	beehive	1	31.29999923786	68.5	Dang Nai	8	0	2025-02-23T15:57:25.263452Z
e5e018a1faca	beehive	1	31.39999961853	68.5	Dang Nai	8	0	2025-02-23T15:57:31.273565Z
e5e018a1faca	beehive	1	31.29999923786	68.5	Dang Nai	8	0	2025-02-23T15:57:37.285242Z
e5e018a1faca	beehive	1	31.39999961853	68.5	Dang Nai	8	0	2025-02-23T15:57:43.297782Z
e5e018a1faca	beehive	2	31.39999961853	68.599998474121	Dang Nai	8	0	2025-02-23T15:57:49.308611Z
e5e018a1faca	beehive	2	31.29999923786	68.599998474121	Dang Nai	2	0	2025-02-23T15:57:55.319521Z
e5e018a1faca	beehive	2	31.39999961853	68.880003051757	Dang Nai	8	0	2025-02-23T15:58:01.334020Z
e5e018a1faca	beehive	3	31.38000061853	68.880003051757	Dang Nai	8	0	2025-02-23T15:58:07.344037Z

Link:

https://drive.google.com/drive/folders/1DXZFzizVYcct8oy0EXTtIIBykgITZiCj?usp=s_haring

Part 2

Esp8266 connect wifi with 3 mode:

Accession point mode(phát wifi): `WiFi.mode(WIFI_AP);`

Station mode(kết nối wifi): `WiFi.mode(WIFI_STA);`

Accession point and Station mode(vừa phát vừa thu): `WiFi.mode(WIFI_AP_STA);`

Turn off wifi mode: `WiFi.mode(WIFI_OFF);`

- On this project we focus on the station mode. In this mode, we need to collect data for: user name/ password/ ip gateway / subnet mask (of Access Point)

A . Sao e dùng điện thoại mà sai 4G vô trình duyệt nhập IP của Esp8266 lại ko vô được.mà mở wifi trên điện thoại lên là lại vô được .có khi cùng wifi ms vô dk. Mong a trả lời dùm e

 1   **Phản hồi**

  **4 phản hồi**

 **@dienthongminhesmart** 2 năm trước

Webserver trên esp8266 là web cục bộ, phải chung mạng mới truy cập được

-
- Check SSID is the current connected wifi on cmd: `Netsh WLAN show interfaces`
- Check password of the SSID: `netsh wlan show profile name= "Tên Wi-Fi" key=clear`

- Demo connect esp8266 to wifi AP:

sketch_nov10a.ino ●

```
1  #include <ESP8266WiFi.h>
2
3  const char* ssid = "Nhung";
4  const char* password = "khimvanhung";
5
6
7  void setup() {
8      // put your setup code here, to run once:
9      Serial.begin(9600);
10     WiFi.mode(WIFI_STA);
11
12     WiFi.begin(ssid,password);
13
14     while(WiFi.status() !=WL_CONNECTED){
15         delay(500);
16         Serial.print(".");
17     }
18     Serial.println("WiFi connected");
19     Serial.print("IP HOST: ");
20     Serial.println(WiFi.localIP());
21 }
22
23 void loop(){
24
25 }
```

Output Serial Monitor ✕

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM4')

```
17:59:19.252 -> .....WiFi connected
17:59:21.812 -> IP HOST: 192.168.1.138
```

- Config for esp wifi(thiết lập cấu hình wifi): `WiFi.config(staticip, gateway, subnet, dns1, dns2);`

- **Staticip:** static ip we want to assign for esp (ip tĩnh muốn gán cho esp)
- **Gateway:** IP of gateway (router) to connect the outside network (ip của gateway của router để kết nối với mạng bên ngoài)
- **Subnet:** subnet for ip range of local network (subnet xác định phạm vi ip của mạng nội bộ)
- **Dns1, dns2:** tham số tùy chọn của máy chủ phân giải tên miền

- **Use cmd** `Serial.println(WiFi.gatewayIP());`

`Serial.println(WiFi.subnetMask());`

- **To check the current AP gateway and subnet mask.**

```
Serial.println(WiFi.gatewayIP());
Serial.println(WiFi.subnetMask());
```

```
}
```

Serial Monitor ✕

(Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM4')

```
7.900 -> n? $?! ? ? ? ? OCE ? ? ? ? CAM4 ? < ? ? C1X.....WiFi connected
2.364 -> IP HOST: 192.168.1.138
2.396 -> 192.168.1.1
2.441 -> 255.255.255.0
```

-
- ****note:** the staticip should have the same subnet as AP (192.168.1.XXX)

- **Manage the connection:**

`WiFi.status();`

0 : WL_IDLE_STATUS

Đang thay đổi trạng thái

1 : WL_NO_SSID_AVAIL

SSID không được tìm thấy

3 : WL_CONNECTED

Kết nối thành công

4 : WL_CONNECT_FAILED

Sai mật khẩu kết nối

6 : WL_DISCONNECTED

Chưa thiết lập chế độ STA


```

23 }
24 Serial.println("WiFi connected");
25 Serial.print("IP HOST: ");
26 Serial.println(WiFi.localIP());
27 Serial.println(WiFi.gatewayIP());
28 Serial.println(WiFi.subnetMask());
29 Serial.println(WiFi.status());
30 }
31
32 void loop(){

```

Output Serial Monitor x

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM4')

```

18:51:22.296 -> 0~?4!~_CE~CAM4,X.....WiFi connected
18:51:29.701 -> IP HOST: 192.168.1.30
18:51:29.701 -> 192.168.1.1
18:51:29.743 -> 255.255.255.0
18:51:29.743 -> 3

```

- **Check if wifi is connected:** `WiFi.isConnected()`: true if connected, false if disconnected
- **Disconnect and Reconnected after that:** `WiFi.reconnect()`; -> esp have this in-build function
- **Take the ip address the esp connected:** `WiFi.localIP()`;
- **Mqtt_server:** mqtt broker

connect_to_wifi_and_configuration | Arduino IDE 2.3.3

File Edit Sketch Tools Help

NodeMCU 1.0 (ESP-12E Module)

connect_to_wifi_and_configuration.ino

```

1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h> //mqtt protocol
3
4
5 const char* ssid = "nhung";
6 const char* password = "khimvanhung";
7 const char* mqtt_server = "192.168.1.69"; //mqtt broker
8
9 WiFiClient espClient;
10 PubSubClient client(espClient); //use espClient for network communication
11
12 void setup_wifi() {
13
14     delay(10);
15     // We start by connecting to a WiFi network
16     Serial.println();
17     Serial.print("connecting to ");
18     Serial.println(ssid);
19
20     WiFi.mode(WIFI_STA);
21     IPAddress staticip(192,168,1,30);
22     IPAddress gateway(192,168,1,1);
23     IPAddress subnet (255,255,255,0);

```

Command Prompt

```

IPv4 Address. . . . . : 192.168.56.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 8:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix  :

Wireless LAN adapter Local Area Connection* 12:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix  :

Wireless LAN adapter Local Area Connection* 11:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix  :

Wireless LAN adapter Wi-Fi:
Connection-specific DNS Suffix  :
IPv4 Address. . . . . : 2405:4803:c852:dd70:6468:3cab:3baa:f4ae
IPv6 Address. . . . . : 2405:4803:c852:dd70:ffff:ffff:ffff:ffffd
Temporary IPv6 Address. . . . : 2405:4803:c852:dd70:7843:566b:8508:416b
Link-local IPv6 Address . . . : fe80::64e5:21f8:80f4:6f3994
IPv4 Address. . . . . : 192.168.1.69
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::134
192.168.1.1

```

- **Fail to connect:**

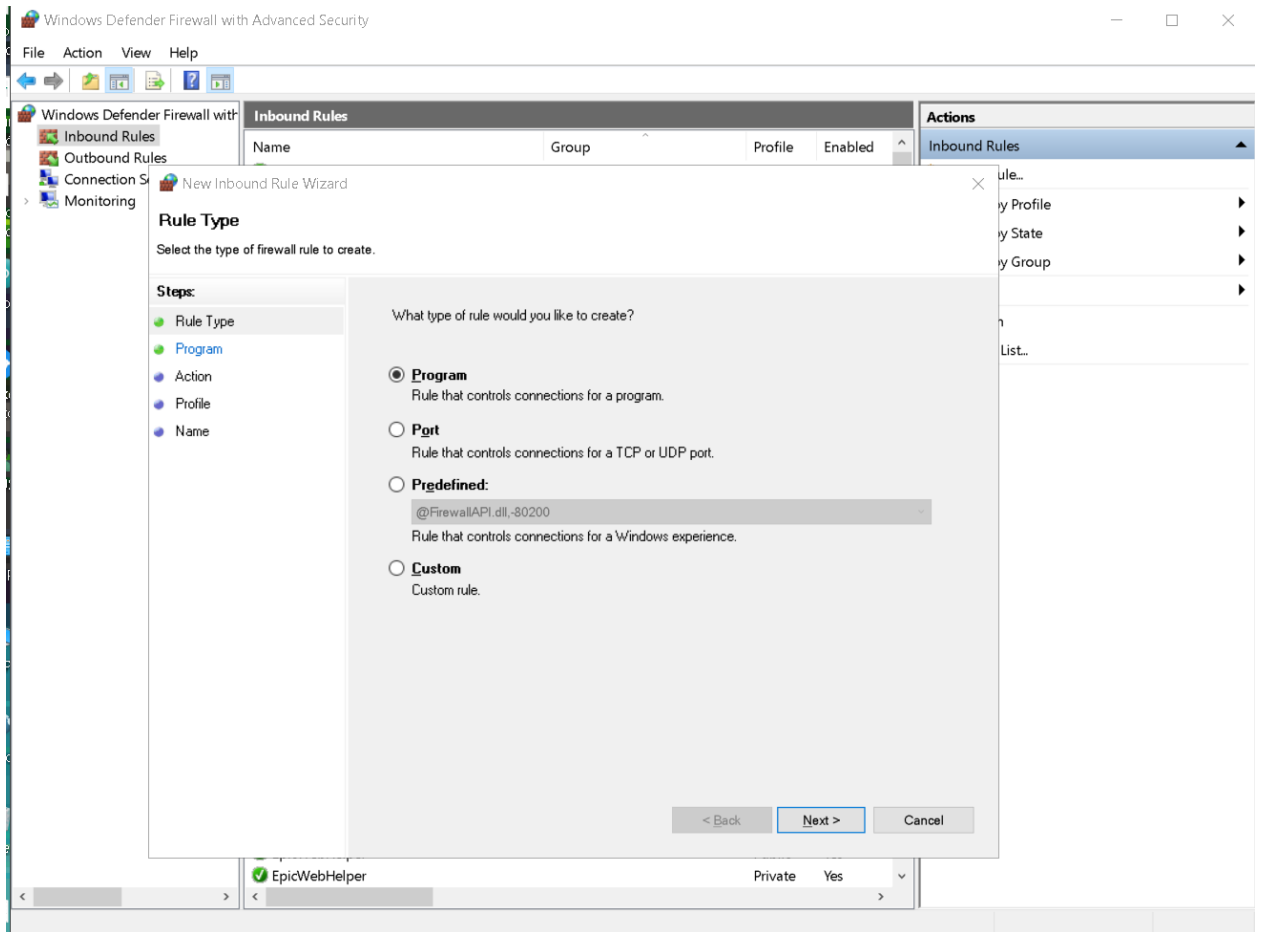
```
41 |
42 | void callback(char* topic, byte* payload, unsigned int length) {
43 | ..
44 |
45 | }
46 |
47 | void reconnect() {
48 |     // Loop until we're reconnected
49 |     while (!client.connected()) {
50 |         Serial.print("Attempting MQTT connection...");
51 |
52 |         if (client.connect("ESP8266Client")) {
53 |             Serial.println("connected");
54 |         } else {
55 |             Serial.print("failed, rc=");
56 |             Serial.print(client.state());
57 |             Serial.println(" try again in 5 seconds");
58 |             delay(5000);
59 |         }
60 |     }
61 | }
```

Output Serial Monitor x

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM4')

13:31:41.083 -> Attempting MQTT connection...failed, rc=-2 try again in 5 seconds

- Go to window defender/advance setting/inbound rules/new rules/ports



Protocol and Ports

Specify the protocols and ports to which this rule applies.

Steps:

● Rule Type

● Protocol and Ports

● Action

● Profile

● Name

Does this rule apply to TCP or UDP?

☒ **TCP**

☐ **UDP**

Does this rule apply to all local ports or specific local ports?

☐ **All local ports**

☒ **Specific local ports:**

1883

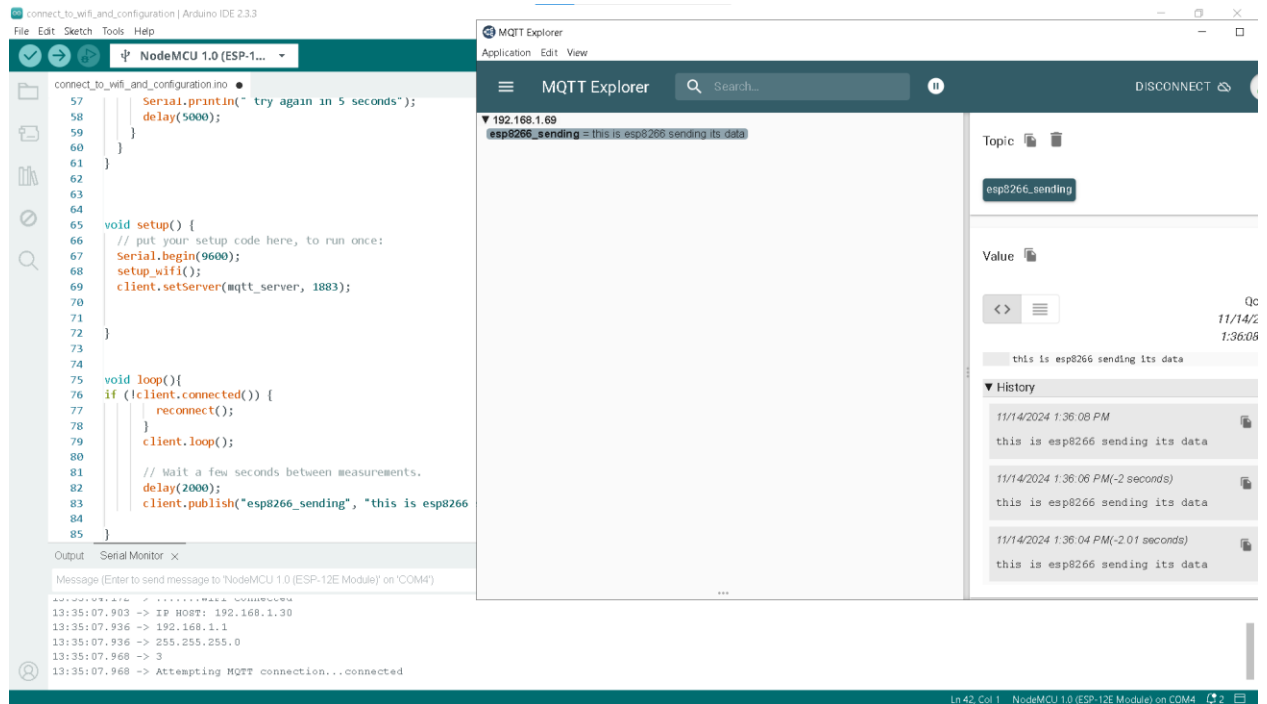
Example: 80, 443, 5000-5010

< Back

Next >

Cancel

- Finish sending data from esp8266 to device(mqtt explorer):



Subscribe:

```

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");

        if (client.connect("ESP8266Client")) {
            Serial.println("connected");
            client.subscribe("device/led");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}
  
```

```

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();

    // Switch on the LED if an 1 was received as first character
    if ((char)payload[0] == '1') {
        Serial.print("esp8266 receive message 1");
    }
}

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    setup_wifi();
    client.setServer(mqtt_server, 1883);
    client.setCallback(callback);
}

```

Output Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM4')

```

14:01:04.322 -> 192.168.1.1
14:01:04.322 -> 255.255.255.0
14:01:04.354 -> 3
14:01:04.354 -> Attempting MQTT connection...connected
14:01:32.296 -> Message arrived [device/led] 1
14:01:32.328 -> esp8266 receive message 1

```

Part 3: send data from esp8266 to questdb

```
4 #include <Arduino_JSON.h>
5 #include <Arduino_JSON.h>
6
```

Include library

```
JSONVar data;
```

Define json variable

```
data["temperature"] = getTemp("c");
data["humidity"] = getTemp("h");
data["place"] = "Dong Nai";
String jsonString = JSON.stringify(data);

client.publish("beehive", jsonString.c_str(), true);
```

Publish the converted data

Adjust refreshing time of Grafana dashboard on webapp:

```
<iframe src="http://localhost:3000/d-solo/bedrbmhdvpm9se/new-
dashboard?orgId=1&refresh=1m&timezone=browser&panelId=1&__feature.dashboardSceneS
olo" width="750" height="400" ></iframe>
```

→ With refresh = 1m equal to refreshing time will be 1 minute

```
Questa db sql: SELECT host, topic, uv_value, place, uv_index, totalBee, sound,
weight, temperature_inside, humidity_inside, temperature_outside,
humidity_outside, timestamp, timestamp + 25200000000L AS vn_time
from mqtt_consumer
```