

# PM1000H Meter with gateway

## Software:

NUM	item	software	Notes
1	Novakon Gateway	Novakon web page configuration	Download : <a href="https://mosquitto.org/files/binary/win64/mosquitto-2.0.18-install-windows-x64.exe">https://mosquitto.org/files/binary/win64/mosquitto-2.0.18-install-windows-x64.exe</a>
2	mosquittolocalserver	link	

## Hardware:

NUM	item	model
1	Novakon Gateway	GW-01

## Commnuication parameter and wiring:

vendor	Modus
interface	RS485
Baud rate	9600
parity	even
Data bits	8
Stop bits	1
address	2



←↻⚠ Not secure | im-connect.local

⚙ Running

🔌 Usage:11%

💻 Free:73%

💾 Free:2452M

👤 admin

🌐 Language: English

IM-CONNECT • PLC Setting

Communicator Port Setting

Add LinkDelete Link

Add NodeDelete Node

sta(1-to-1)

1

Node Setting

Enabled

Display\_Name:

1

Advanced Setting:

....

Command Delay:

0

Station No:

1

Set to default station

Retry:

3

Timeout:

500

←↻⚠ Not secure | im-connect.local

⚙ Running

🔌 Usage:19%

💻 Free:73%

💾 Free:2452M

👤 admin

🌐 Language: English

IM-CONNECT • Tag Setting

AddAdd many...DeleteDelete All

Modbus Setting Dialog

☐ Exact Match

🔍 Type to Search

ImportExportPaste

Id	Tag Name	Access	Connector	Data Type	Tag Address	Modbus Address	OPCUA
1	freq	Read/Write	sta	FLOAT	2-FN6HRSW3110		<input checked="" type="checkbox"/> Include
2	VLL	Read/Write	sta	FLOAT	2-FN6HRSW3026		<input checked="" type="checkbox"/> Include
3	VLN	Read/Write	sta	FLOAT	2-FN6HRSW3036		<input checked="" type="checkbox"/> Include

1

Tag Name:

VLN

Access :

Read/Write

Connector:

sta

Data Type:

FLOAT

Station No:

Tag Address:

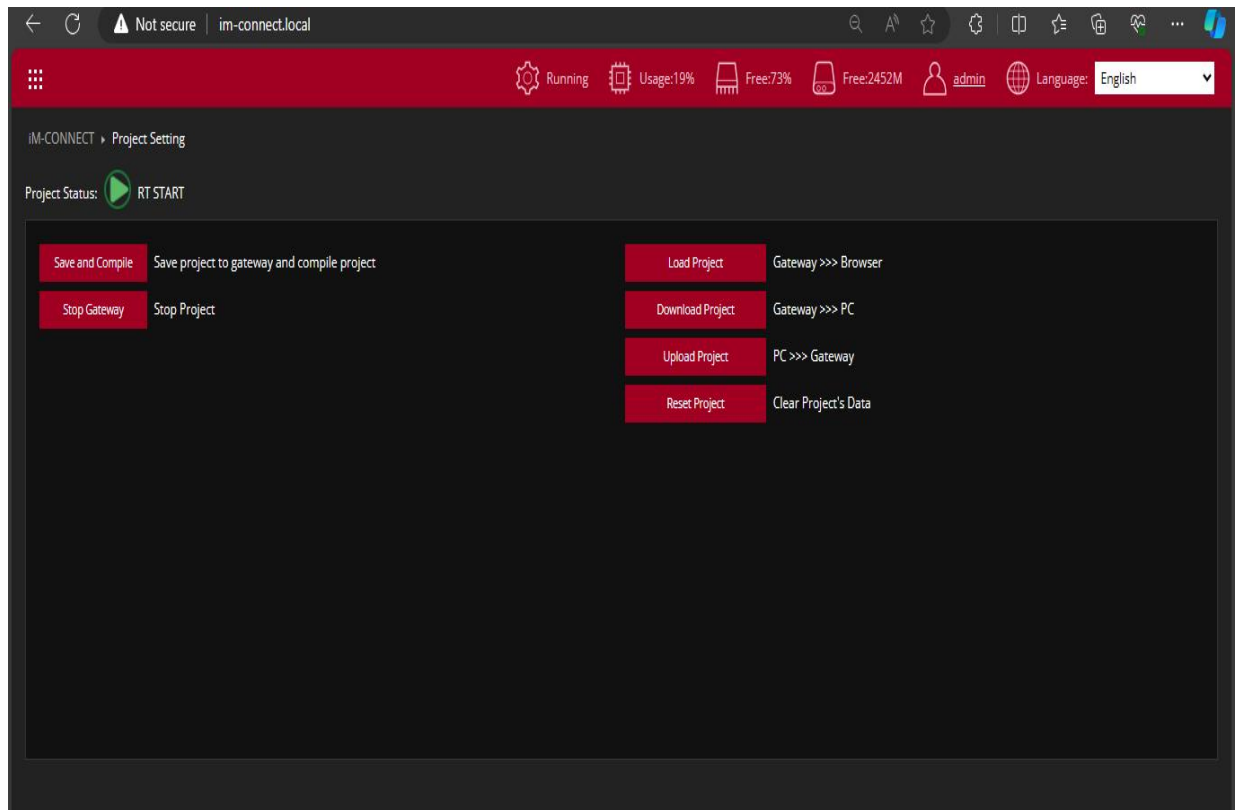
FN6HRSW

3036

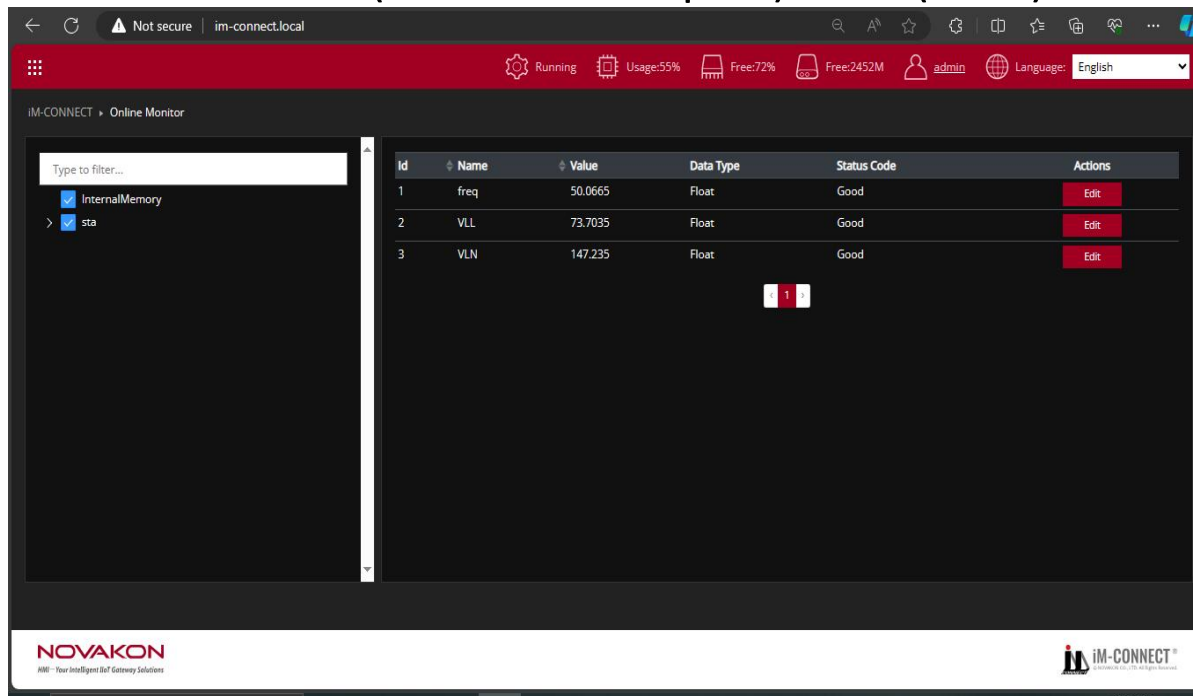
✓

Matched Format

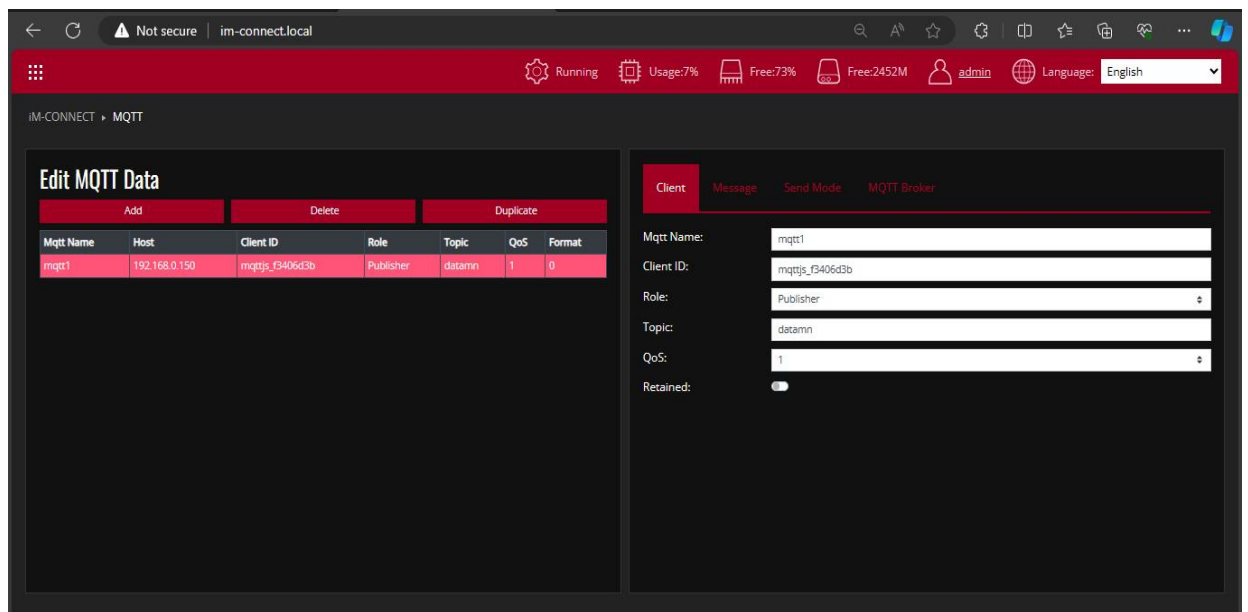
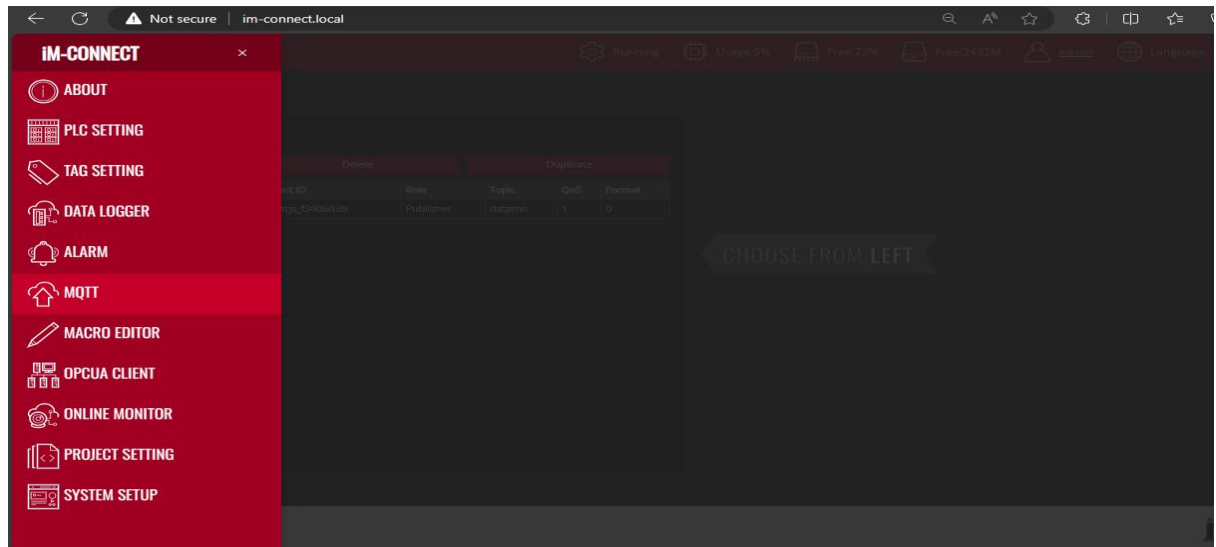
Holding Registers (read/write with function code 6, HR1-HR65536, only swap in 32b)



After enter data (save and compile ) then (start)

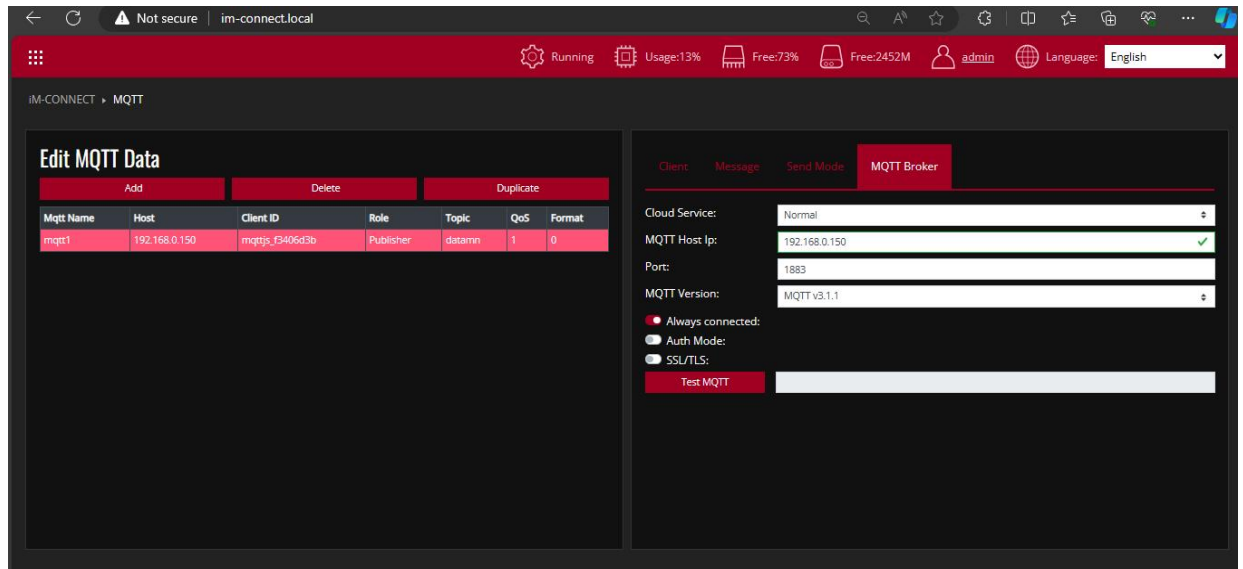


- 1-Go to online monitor to test the connection ..
- 2-If it good then go to mqtt

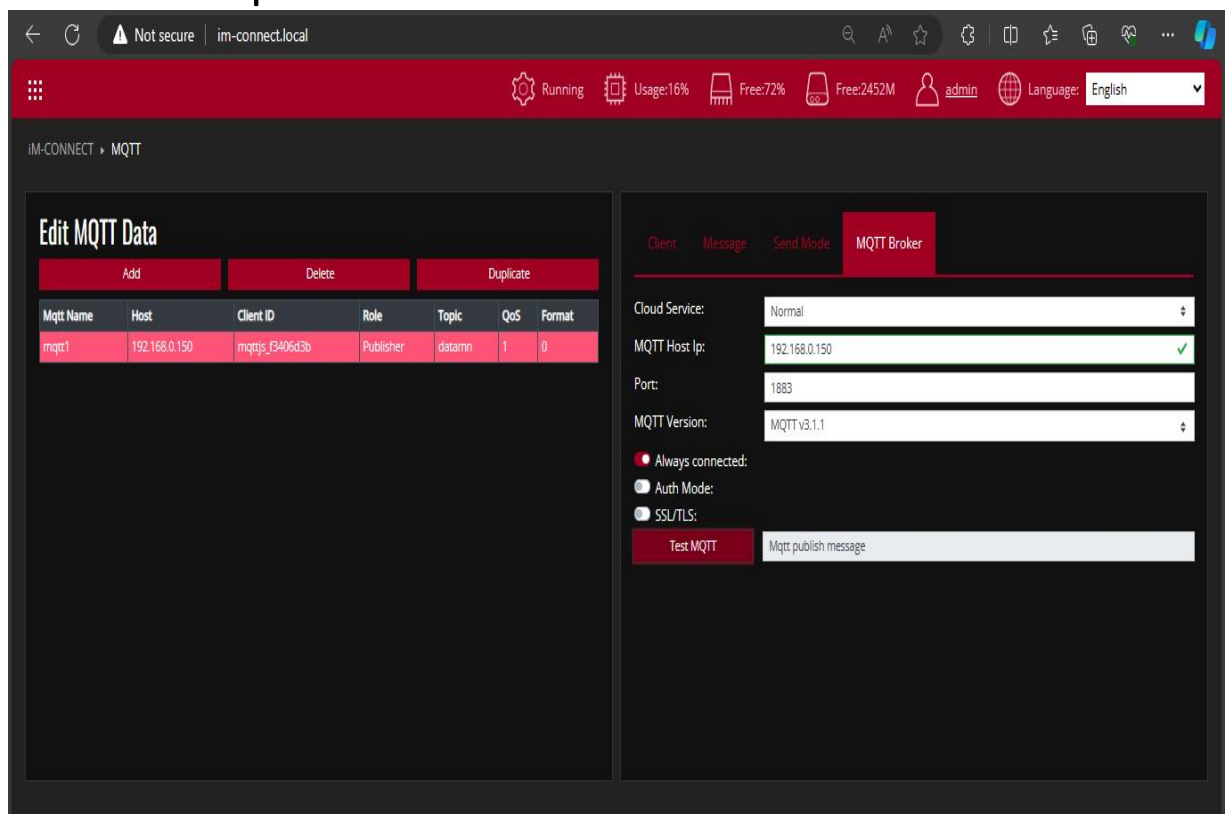


- client id is auto generated
- topic should be unique to avoid overlap





- Enter the ip address of pc
- Enter the port 1883



Open MQTTbox and fill data (host&topic&client id&username&password&qos)

MQTT Client Name: mqtt1

MQTT Client Id: mqttjs\_f3406d3b

Append timestamp to MQTT client id? ☒ Yes

Broker is MQTT v3.1.1 compliant? ☒ Yes

Protocol: mqtt / tcp

Host: 192.168.0.150

Clean Session? ☒ Yes

Auto connect on app launch? ☒ Yes

Username: sla

Password: \*\*\*\*

Reschedule Pings? ☒ Yes

Queue outgoing QoS zero messages? ☒ Yes

Reconnect Period (milliseconds): 1000

Connect Timeout (milliseconds): 30000

KeepAlive (seconds): 10

Will - Topic: datamn

Will - QoS: 1 - Atleast Once

Will - Retain: ☐ No

Will - Payload:

Save Delete

Choose subscribe and

MQTTBox

Menu Connected Add publisher Add subscriber

Topic to subscribe: datamn

QoS: 0 - Almost Once

Subscribe

mqtt - mqtt://192.168.0.150

["freq":50.05939865112305,"VLL":73.9872055053711,"VLN":148.0198516845703]

qos : 0, retain : false, cmd : publish, dup : false, topic : datamn, messageid : , length : 81, Raw payload : 1233410211410111334585348464853575157565453494950514853443486767634585551465756555048533485351554949443486767834584952564648495756534954565253554851125

["freq":50.057987213134766,"VLL":73.94178009033203,"VLN":147.93960571289062]

qos : 0, retain : false, cmd : publish, dup : false, topic : datamn, messageid : , length : 84, Raw payload : 123341021141011133458534846485355756555049514951525554544434867676345855514657524955564848

Then test mqqt and message must be (Mqtt publish message) to confirm the connection

\* if there is any problem follow this steps \*



### **local mosquitto server setup:**

1. download the local server app from the provided link before
2. install the app
3. after installation check the mosquitto services is run correctly
4. the broker ip is your device ip (wifi - ethernet)

note: if you test broker and other devices not connected to it  
add these two lines in **mosquitto.conf** file in setup file of  
mosquitto app in c drive

listener 1883

allow\_anonymous true

and then save the file

5-Check service and make sure that mosquitto is running

6-Check firewall setting by Open Windows Defender Firewall Settings:

\*Press Win + S to open the Windows search bar.

\*Type "Windows Defender Firewall" and select the corresponding result.

\*Choose "Advanced Settings" (Windows Firewall with Advanced Security):

On the left-hand side, you will see "Advanced settings." Click on it.

\*Create a New Inbound Rule:

\*In the left-hand pane, select "Inbound Rules."

\*In the right-hand pane, click on "New Rule..." to open the New Inbound Rule Wizard.

\*Select "Port" and Click "Next":

\*Choose "Port" and click "Next."

\*Specify TCP or UDP and the Port Number:

\*Choose either "TCP" or "UDP," depending on the protocol used by your MQTT broker (most likely TCP).

\*Select "Specific local ports" and enter the MQTT broker port number (e.g., 1883 for MQTT/TCP).

\*Click "Next."

\*Choose "Allow the Connection" and Click "Next":

\*

\*Select "Allow the connection" and click "Next."

\*Specify When to Apply the Rule and Click "Next":

\*Choose when to apply the rule. Generally, keeping all options selected is fine.

\*Click "Next."

\*Name the Rule and Add a Description (Optional):

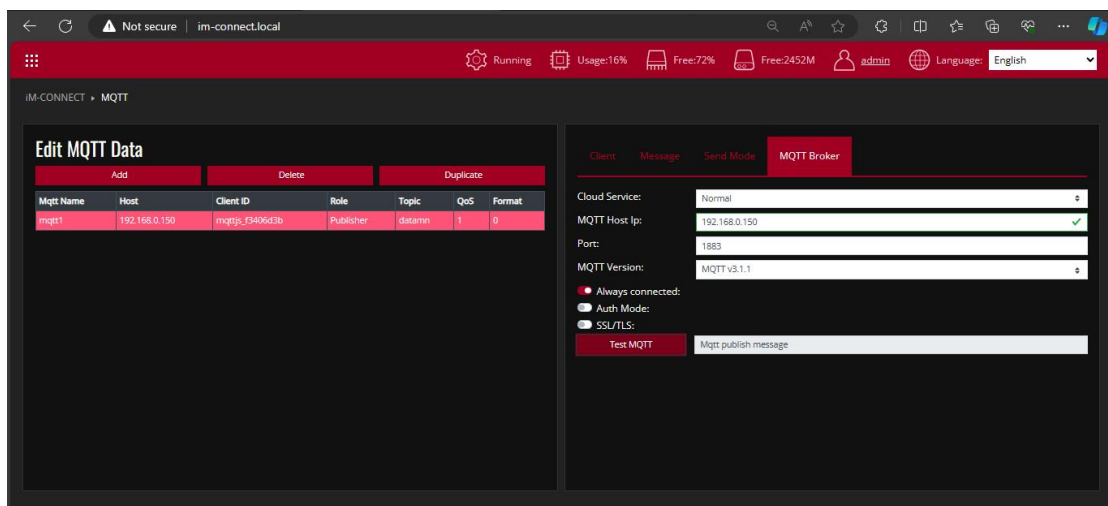
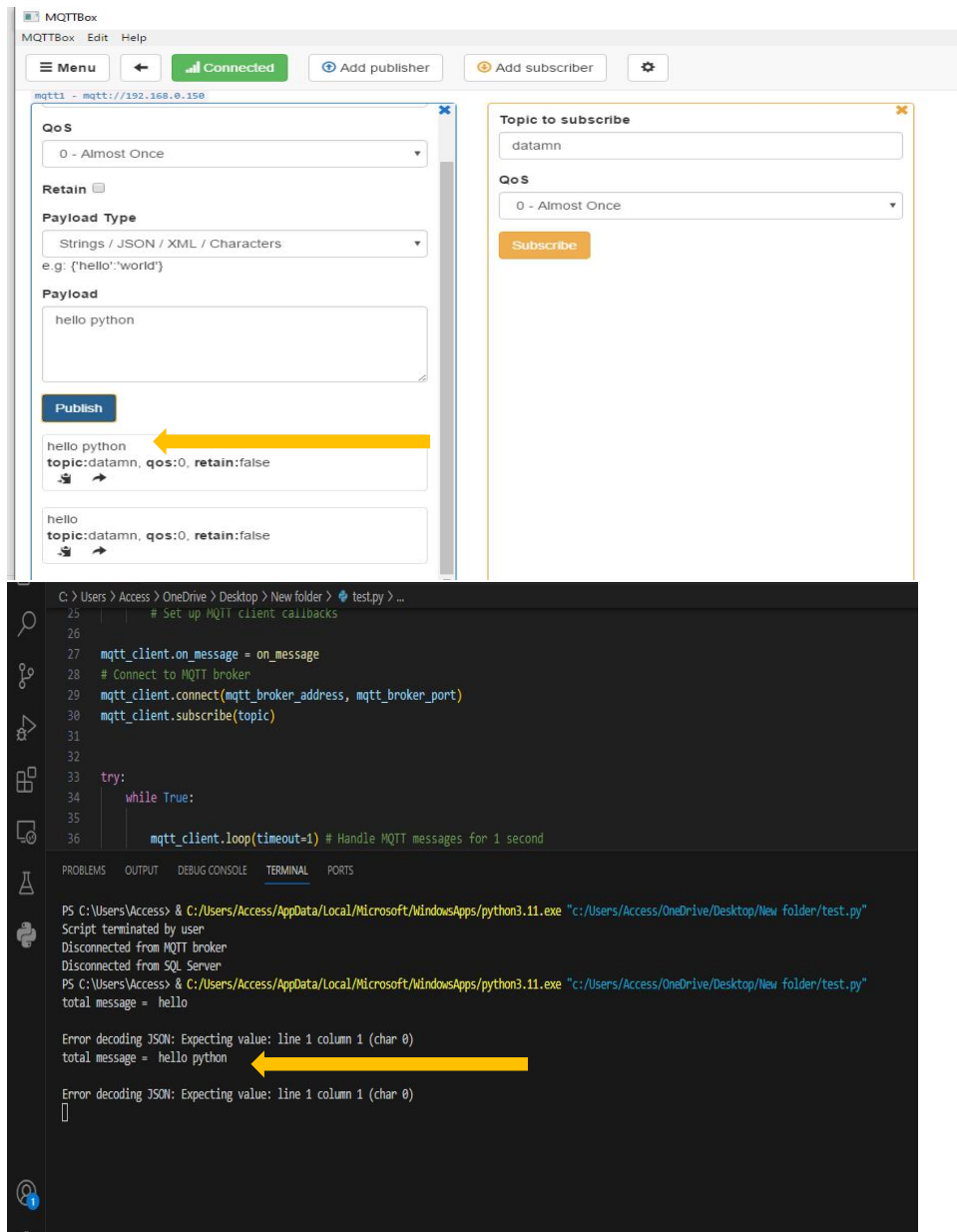
\*Provide a name and, if desired, a description for the rule.

\*Click "Finish."

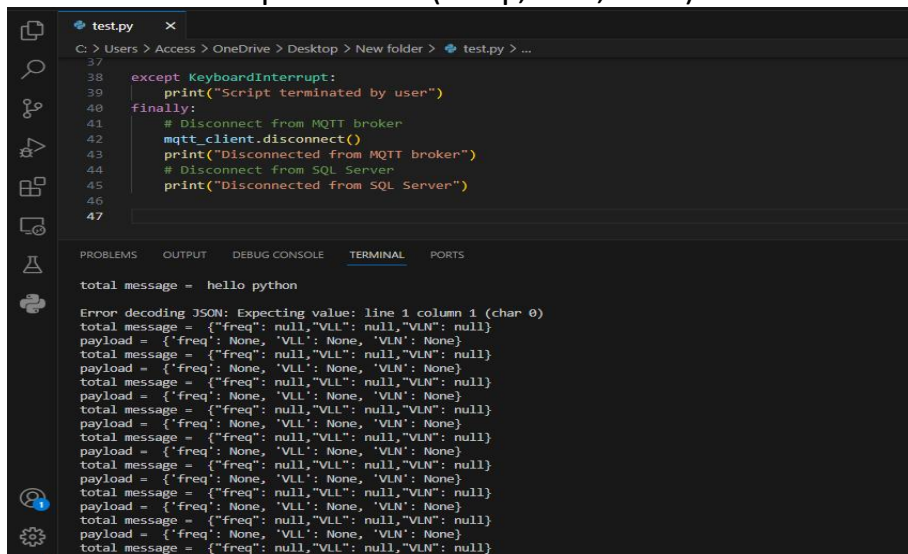
## Connect with python

```
C:\> Users > Access > OneDrive > Desktop > New folder > test.py > ...
1  import paho.mqtt.client as mqtt
2  import json
3
4  # MQTT broker connection details
5  mqtt_broker_address = "192.168.0.150"
6  mqtt_broker_port = 1883
7  topic = "datamn" # Replace with the actual MQTT topic
8
9
10 # Connect to MQTT broker
11 mqtt_client = mqtt.Client()
12
13
14 def on_message(client, userdata, message):
15     try:
16         # Decode the JSON message
17         valuet = message.payload.decode("utf-8")
18         print("total message = ",valuet)
19
20         payload = json.loads(valuet)
21         print("payload = ",payload)
22
23     except json.JSONDecodeError as e:
24         print(f"Error decoding JSON: {e}")
25     # Set up MQTT client callbacks
26
27     mqtt_client.on_message = on_message
28     # Connect to MQTT broker
29     mqtt_client.connect(mqtt_broker_address, mqtt_broker_port)
30     mqtt_client.subscribe(topic)
31
32
33 try:
34     while True:
35
36         mqtt_client.loop(timeout=1) # Handle MQTT messages for 1 second
37
38 except KeyboardInterrupt:
39     print("Script terminated by user")
40 finally:
41     # Disconnect from MQTT broker
42     mqtt_client.disconnect()
43     print("Disconnected from MQTT broker")
44     # Disconnect from SQL Server
45     print("Disconnected from SQL Server")
46
47
```

test from mqttbox



The value of the parameter (Freq , VLL , VLN ) will received on python



The screenshot shows a VS Code editor with a file named `test.py` open. The file path is `C:\Users\Access > OneDrive > Desktop > New folder > test.py > ...`. The script contains a `finally` block that disconnects from an MQTT broker and a SQL server. The terminal output shows a series of JSON messages being received, each with `freq`, `vll`, and `vln` fields set to `None` or `null`. An error message is also visible: "Error decoding JSON: Expecting value: line 1 column 1 (char 0)".

```
37
38 except KeyboardInterrupt:
39     print("Script terminated by user")
40 finally:
41     # Disconnect from MQTT broker
42     mqtt_client.disconnect()
43     print("Disconnected from MQTT broker")
44     # Disconnect from SQL Server
45     print("Disconnected from SQL Server")
46
47
```

total message = hello python

Error decoding JSON: Expecting value: line 1 column 1 (char 0)

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

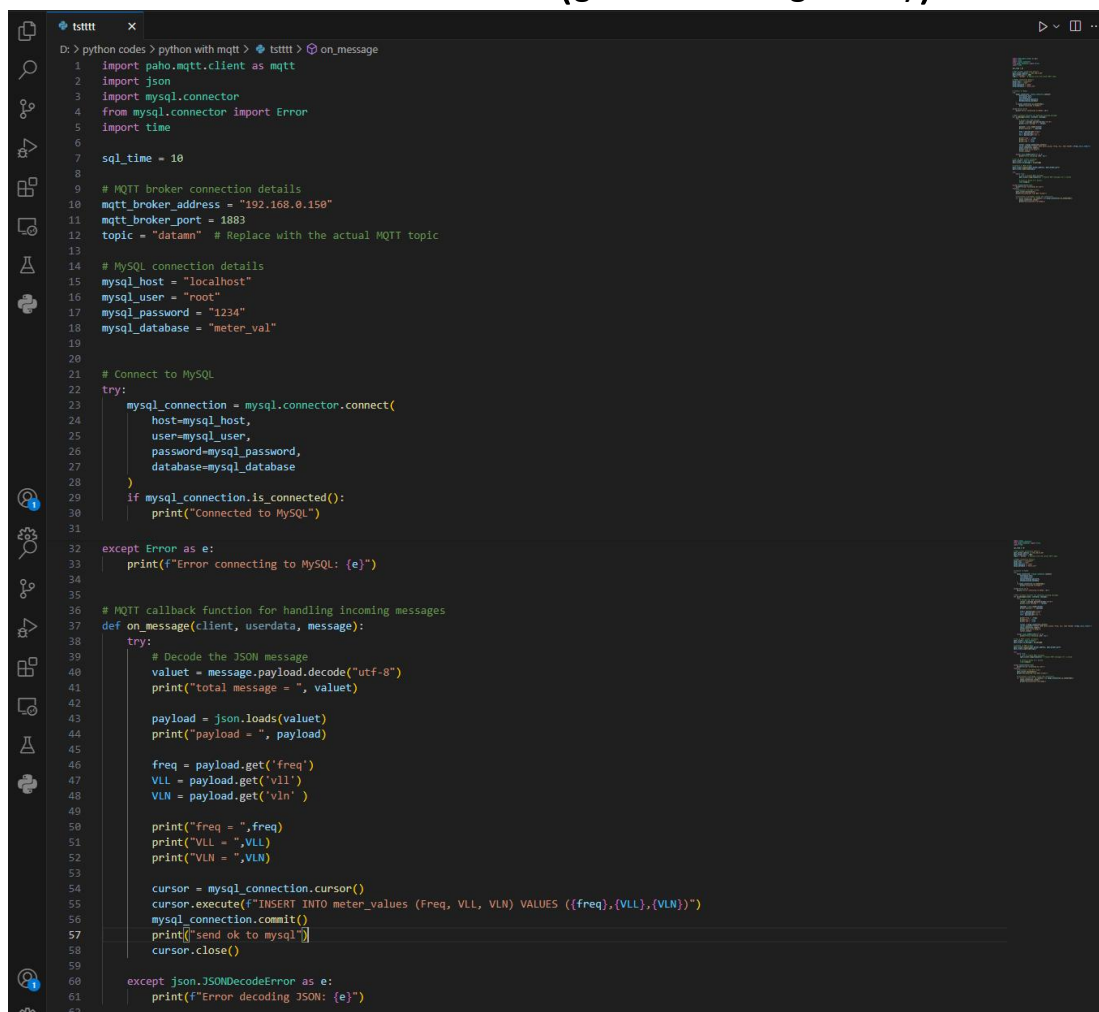
payload = {"freq": None, "vll": None, "vln": None}

total message = {"freq": null, "vll": null, "vln": null}

payload = {"freq": None, "vll": None, "vln": None}

\*note\* the value is null because of desconnecting the meter


## Connect with database (get data from gateway)





The screenshot shows a VS Code editor with a file named `tstttt.py` open. The file path is `D:\python codes > python with mqtt > tstttt > on_message`. The script imports `paho.mqtt.client`, `json`, `mysql.connector`, and `time`. It defines MQTT broker connection details and MySQL connection details. It connects to MySQL and defines an `on_message` callback function that decodes the JSON message, extracts the `freq`, `vll`, and `vln` fields, and inserts them into a MySQL database. The terminal output shows the script running successfully, connecting to MySQL, and receiving a message.


```
1 import paho.mqtt.client as mqtt
2 import json
3 import mysql.connector
4 from mysql.connector import Error
5 import time
6
7 sql_time = 10
8
9 # MQTT broker connection details
10 mqtt_broker_address = "192.168.0.150"
11 mqtt_broker_port = 1883
12 topic = "datam" # Replace with the actual MQTT topic
13
14 # MySQL connection details
15 mysql_host = "localhost"
16 mysql_user = "root"
17 mysql_password = "1234"
18 mysql_database = "meter_val"
19
20
21 # Connect to MySQL
22 try:
23     mysql_connection = mysql.connector.connect(
24         host=mysql_host,
25         user=mysql_user,
26         password=mysql_password,
27         database=mysql_database
28     )
29     if mysql_connection.is_connected():
30         print("Connected to MySQL")
31
32 except Error as e:
33     print(f"Error connecting to MySQL: {e}")
34
35
36 # MQTT callback function for handling incoming messages
37 def on_message(client, userdata, message):
38     try:
39         # Decode the JSON message
40         valuet = message.payload.decode("utf-8")
41         print("total message = ", valuet)
42
43         payload = json.loads(valuet)
44         print("payload = ", payload)
45
46         freq = payload.get('freq')
47         vll = payload.get('vll')
48         vln = payload.get('vln')
49
50         print("freq = ", freq)
51         print("vll = ", vll)
52         print("vln = ", vln)
53
54         cursor = mysql_connection.cursor()
55         cursor.execute("INSERT INTO meter_values (Freq, VLL, VLN) VALUES ({freq},{vll},{vln})")
56         mysql_connection.commit()
57         print("send ok to mysql")
58         cursor.close()
59
60 except json.JSONDecodeError as e:
61     print(f"Error decoding JSON: {e}")
62
```


```
D:\> python codes > python with mqtt > tsmttt > on_message
60     except json.JSONDecodeError as e:
61         print(f"Error decoding JSON: {e}")
62
63 # Set up MQTT client callbacks
64 mqtt_client = mqtt.Client()
65 mqtt_client.on_message = on_message
66
67 # Connect to MQTT broker
68 mqtt_client.connect(mqtt_broker_address, mqtt_broker_port)
69 mqtt_client.subscribe(topic)
70
71 try:
72     while True:
73         # Loop to handle MQTT messages
74         mqtt_client.loop(timeout=1) # Handle MQTT messages for 1 second
75
76         # Insert a delay of 1 minute
77         time.sleep(2)
78
79 except KeyboardInterrupt:
80     print("Script terminated by user")
81 finally:
82     # Disconnect from MQTT broker
83     mqtt_client.disconnect()
84     print("Disconnected from MQTT broker")
85
86     # Disconnect from MySQL (close the connection)
87     if 'mysql_connection' in locals() and mysql_connection.is_connected():
88         mysql_connection.close()
89     print("Disconnected from MySQL")
```


 Running

 Usage:12%

 Free:72%

 Free:2443M

 admin

 Language: English

Id	Name	Value	Data Type	Status Code	Actions
1	freq	50.025	Float	Good	Edit
2	vll	400	Float	Good	Edit
3	vln	220	Float	Good	Edit

1

Read from database (mysql)

File Edit View Query Database Server Tools Scripting Help

Navigation: products meter\_values meter\_values products meter\_values meter\_values Administration - Users and Privil... meter\_values

SCHEMAS

Filter objects

- meter\_val
  - Tables
    - meter\_values
      - Columns
        - Freq
        - VLN
        - VLL
      - Indexes
      - Foreign Keys
      - Triggers
    - Views
    - Stored Procedures
    - Functions
  - mydatabase
  - prod
  - produ
  - products
  - sys

1 • SELECT \* FROM meter\_val.meter\_values;

Result Grid

Freq	VLN	VLL
50.025	220	380
50.025	220	380
50.025	220	380
50.025	220	380
50.025	220	400
50.025	220	400
50.025	220	400
50.025	220	400

meter\_values 4 x

Read Only Context Help Snippets

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Information

Table: meter\_values

Columns:

Column	Type
Freq	float
VLN	float
VLL	float

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 18	12:15:41	SELECT * FROM meter_val.meter_values LIMIT 0, 1000	170 row(s) returned	0.000 sec / 0.000 sec
✓ 19	12:15:51	SELECT * FROM meter_val.meter_values LIMIT 0, 1000	179 row(s) returned	0.000 sec / 0.000 sec
✓ 20	12:22:26	SELECT * FROM meter_val.meter_values LIMIT 0, 1000	515 row(s) returned	0.000 sec / 0.000 sec
✓ 21	12:37:52	SELECT * FROM meter_val.meter_values LIMIT 0, 1000	856 row(s) returned	0.015 sec / 0.000 sec
✓ 22	12:37:50	SELECT * FROM meter_val.meter_values LIMIT 0, 1000	859 row(s) returned	0.000 sec / 0.000 sec
✓ 23	12:38:28	SELECT * FROM meter_val.meter_values LIMIT 0, 1000	862 row(s) returned	0.000 sec / 0.000 sec

Object Info Session