

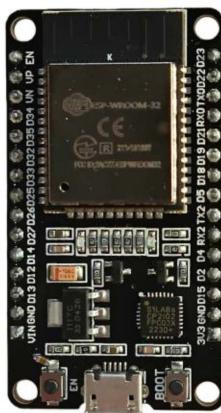
## การทดลองที่ 7 การใช้งาน MQTT และ Node-RED

### วัตถุประสงค์

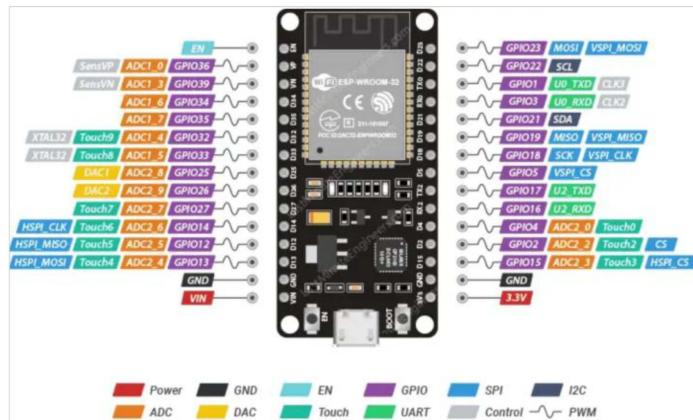
การทดลองนี้มีวัตถุประสงค์เพื่อให้นักศึกษา

- ศึกษาและเข้าใจหลักการทำงานของโพรโทคอล Message Queuing Telemetry Transport (MQTT)
- ฝึกการเขียนโปรแกรมสำหรับการเชื่อมต่อ MQTT และการรับ/ส่งข้อมูลระหว่างอุปกรณ์
- ศึกษาและประยุกต์ใช้งาน Node-RED กับระบบอินเทอร์เน็ตของสรรพสิ่ง
- พัฒนาโปรแกรมสำหรับการเชื่อมต่อระหว่าง Node-RED และ MQTT และอุปกรณ์อินเทอร์เน็ตของสรรพสิ่ง
- สามารถสร้าง Dashboard บน Node-RED เพื่อแสดงข้อมูลของอุปกรณ์อินเทอร์เน็ตของสรรพสิ่งได้ เช่น แสดงสถานะการปิด-เปิดสวิตช์ แสดงอุณหภูมิและความชื้น

การทดลองนี้ใช้บอร์ด ESP32 (ESP-WROOM-32 ภาพที่ 1 (ก) แสดงบอร์ด ESP-WROOM-32 และภาพที่ 1 (ข) แสดงแผนผังขา รายละเอียดของบอร์ด WROOM ESP32 ได้กล่าวไว้แล้วในปฏิบัติการทดลองที่ 5



ก. บอร์ด



ข. แผนผังขา

ภาพที่ 1 บอร์ด ESP-WROOM-32

ที่มา (ข) <https://lastminuteengineers.com/esp32-pinout-reference/>

การทดลองนี้ศึกษาการใช้งานโพรโทคอล MQTT และ Node-RED สำหรับระบบอินเทอร์เน็ตของสรรพสิ่ง โดยแบ่งการทดลองออกเป็น 2 ส่วนคือ การใช้งานโพรโทคอล MQTT และการใช้งาน Node-RED รายละเอียดดังต่อไปนี้

### การทดลองที่ 7.1 การใช้งานโพรโทคอล MQTT

MQTT เป็นโพรโทคอลที่ใช้ในการส่งข้อความระหว่างอุปกรณ์ การทำงานของ MQTT เป็นแบบ Publish/Subscribe (ทำให้เครื่องที่เป็น Publish ส่งข้อความ และเครื่องที่เป็น Subscribe รับข้อความได้โดยผ่านตัวกลางที่เรียกว่า broker) MQTT เป็นโพรโทคอลมาตรฐานสำหรับส่งข้อความของระบบอินเทอร์เน็ตของสรรพสิ่งนี้จากมีขนาดเล็ก รองรับการทำงานแบบเรียลไทม์ และไม่จำกัดแพลตฟอร์ม (รายละเอียดเพิ่มเติม <https://mqtt.org/> และที่ <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf>)

MQTT ประกอบด้วย 3 ส่วนที่สำคัญคือ MQTT Broker, MQTT Client (Publisher/Subscriber) และ Topic รายละเอียดดังนี้ 1) MQTT Broker เป็นตัวกลางในการรับส่งข้อมูลระหว่าง MQTT Client ซึ่งจะมีทั้งแบบจากผู้ให้บริการ เช่น CloudMQTT หรือ Mosquitto หรือแบบสร้างขึ้นมาเองโดยใช้ซอฟต์แวร์ เช่น Mosquitto หรือ HivaMQ ติดตั้งบนเซิร์ฟเวอร์ส่วนตัว หรือแบบบริการพรี เช่น public test brokers ของ Eclipse Mosquitto 2) MQTT Client แบ่งเป็น 2 ประเภทคือแบบ Publisher และ Subscriber โดย Publisher ทำหน้าที่ส่งข้อมูลไปยัง Topic (Publishing) และ Subscriber ทำหน้าที่รับข้อมูลจาก Topic (Subscribing) โดยที่อุปกรณ์ใดๆ สามารถเป็นได้ทั้ง Publisher และ Subscriber และ 3) Topic เป็นหัวข้อในระบบ MQTT สำหรับจัดหมวดหมู่ ซึ่งอาจแบ่งเป็นหัวข้อย่อยได้ โดยใช้เครื่องหมาย “/” ในการแบ่งระดับ หากเป็นหัวข้อในระดับเดียวกันจะใช้เครื่องหมาย “+” เช่น หากต้องการรับข้อมูลของอุณหภูมิทุกห้องภายในบ้านจะใช้ home+temperature หากเป็นหัวข้อหลายระดับจะใช้เครื่องหมาย “#” วางแผนสุดท้ายเพื่อ subscribe ทุกหัวข้อในระดับนั้น เช่น หากต้องการรับข้อมูลทั้งหมดภายในห้อง bedroom ก็จะเป็น home/bedroom/#

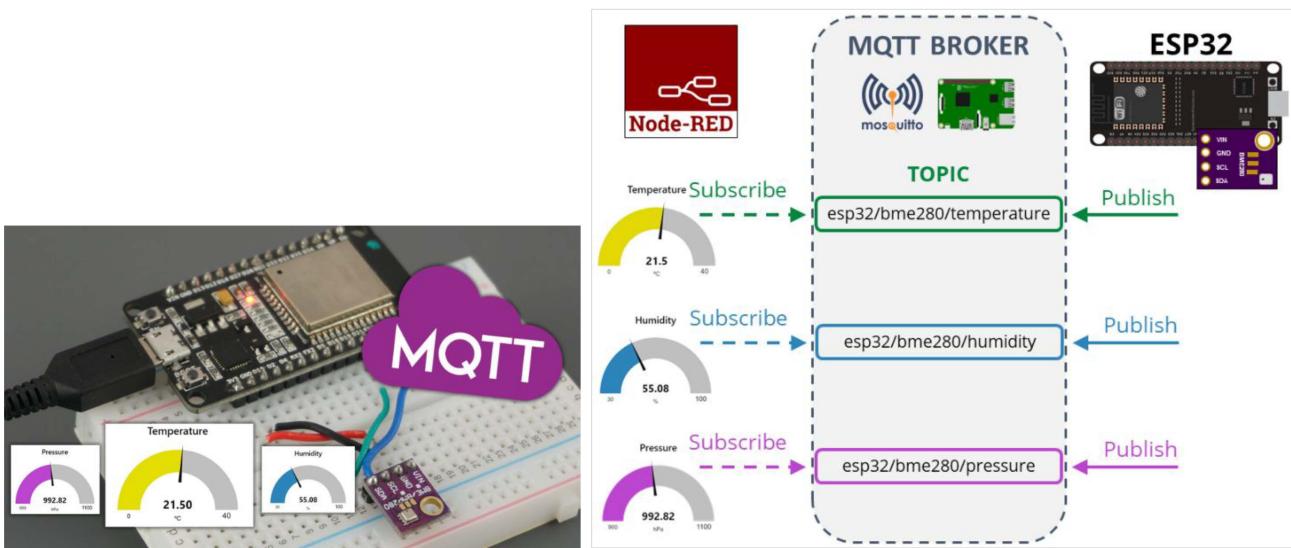
ตัวอย่างสถาปัตยกรรมของ MQTT Publish/Subscribe แสดงดังภาพที่ 3 ในภาพที่ 3 MQTT Client ที่เป็นเซนเซอร์วัดอุณหภูมิเป็น Publisher และมี MQTT Client ที่เป็น Subscriber สำหรับ topic: temperature MQTT Client ที่เป็น Publisher จะส่งค่าอุณหภูมิภายใน topic: temperature และ MQTT Client ที่เป็น Subscriber จะรับข้อมูลค่าอุณหภูมิภายใน topic: temperature



ภาพที่ 3 สถาปัตยกรรมของ MQTT Publish/Subscribe

ที่มา <https://mqtt.org/>

ตัวอย่างการทำงานของ ESP32 MQTT (Publish BME280 Sensor) แสดงดังภาพที่ 4 พิจารณาภาพที่ 4 จากผู้ช่วยที่ 4 ที่มีเซ็นเซอร์ BME280 ใช้วัดอุณหภูมิ ความชื้น และความกดอากาศ ESP32 อ่านค่าเหล่านี้แล้ว Publish ภายใต้ 3 หัวข้อ คือ esp32/bme280/temperature, esp32/bme280/humidity และ esp32/bme280/pressure ขณะที่ Node-RED อยู่ผู้ช่วยที่ 4 สามารถ Subscribe ทั้งสามหัวข้อ ซึ่งการรับส่งข้อมูลนี้จะผ่าน MQTT Broker ภายใต้หัวข้อเหล่านั้น



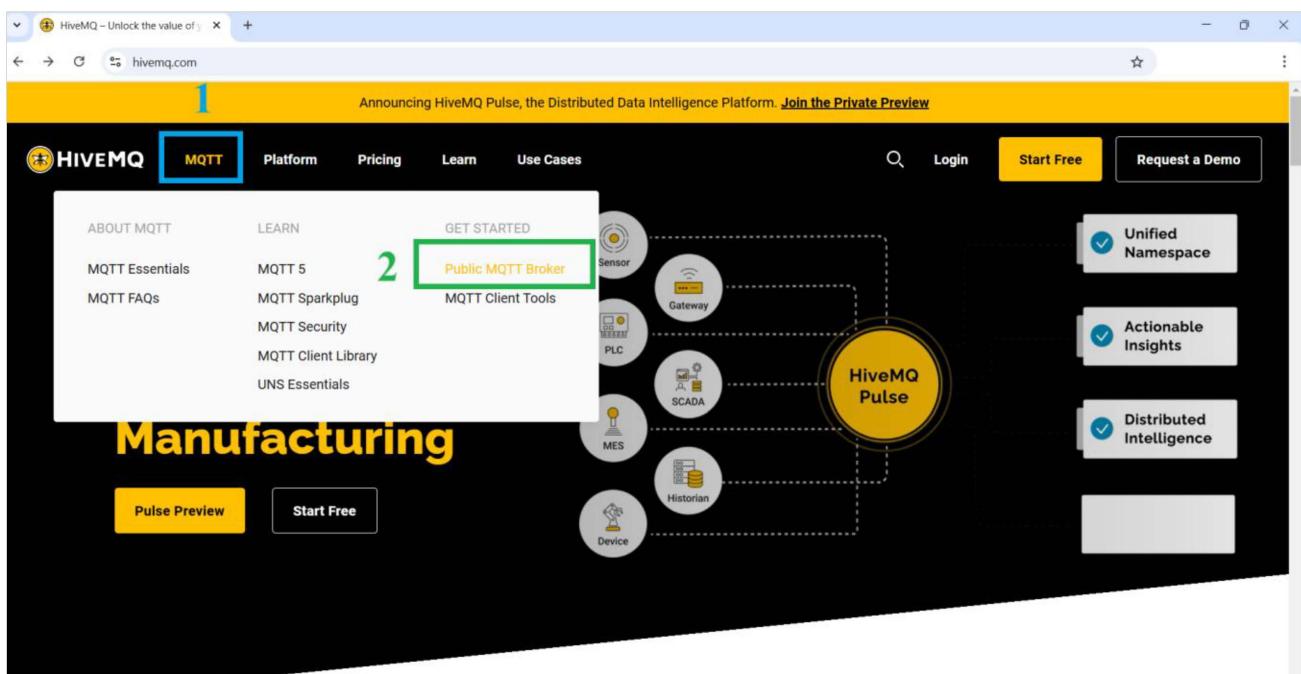
ภาพที่ 4 ESP32 MQTT (Publish BME280 Sensor)

ที่มา <https://randomnerdtutorials.com/esp32-mqtt-publish-bme280-arduino/>

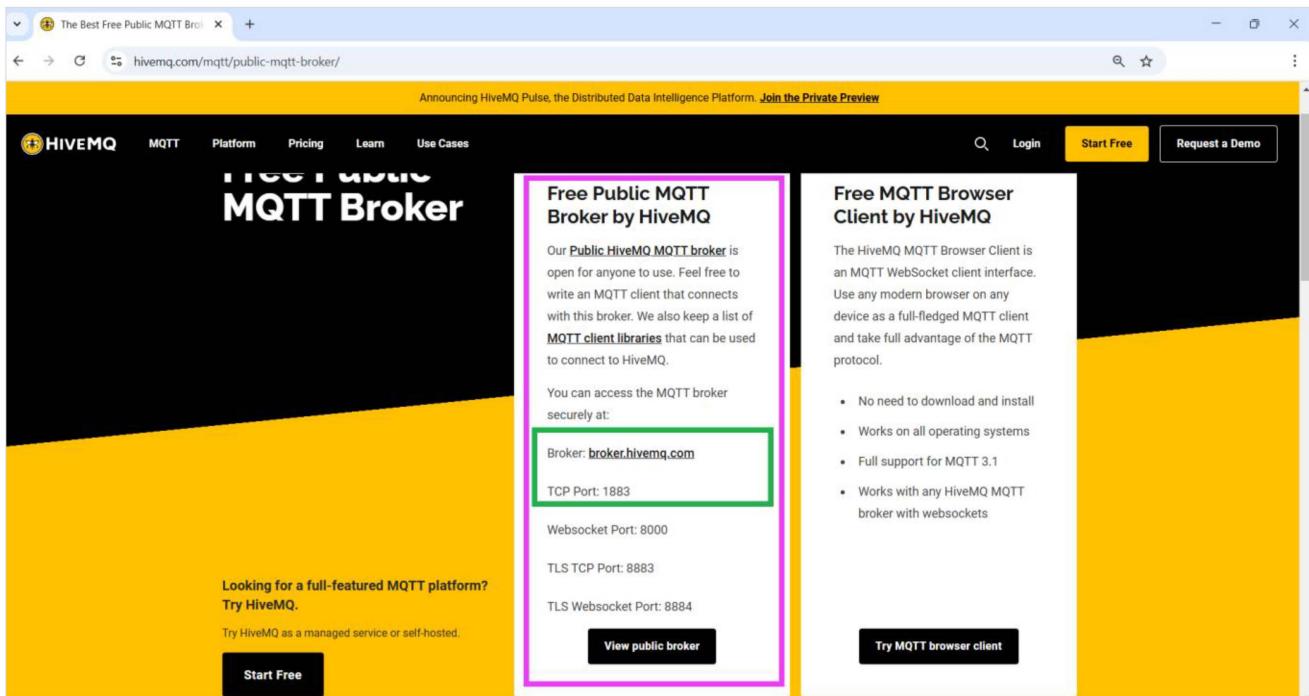
### ทดสอบ MQTT

ในการทดลองนี้ใช้ HiveMQ หรือ Mosquitto ที่เป็น Public Broker ซึ่งมีบริการฟรีสำหรับการทดสอบโดยไม่ต้องลงทะเบียน มีขั้นตอนดังนี้

1. เข้าสู่เว็บไซต์ <https://www.hivemq.com/> ดังภาพที่ 5 จากนั้นคลิกที่ MQTT (กรอบสีฟ้า) จะปรากฏหน้าต่างให้คลิกที่ Public MQTT Broker (กรอบสีเขียว) จะปรากฏดังภาพที่ 6 ซึ่งจะมีข้อมูล MQTT Broker ที่จะใช้งาน (กรอบสีเขียว)

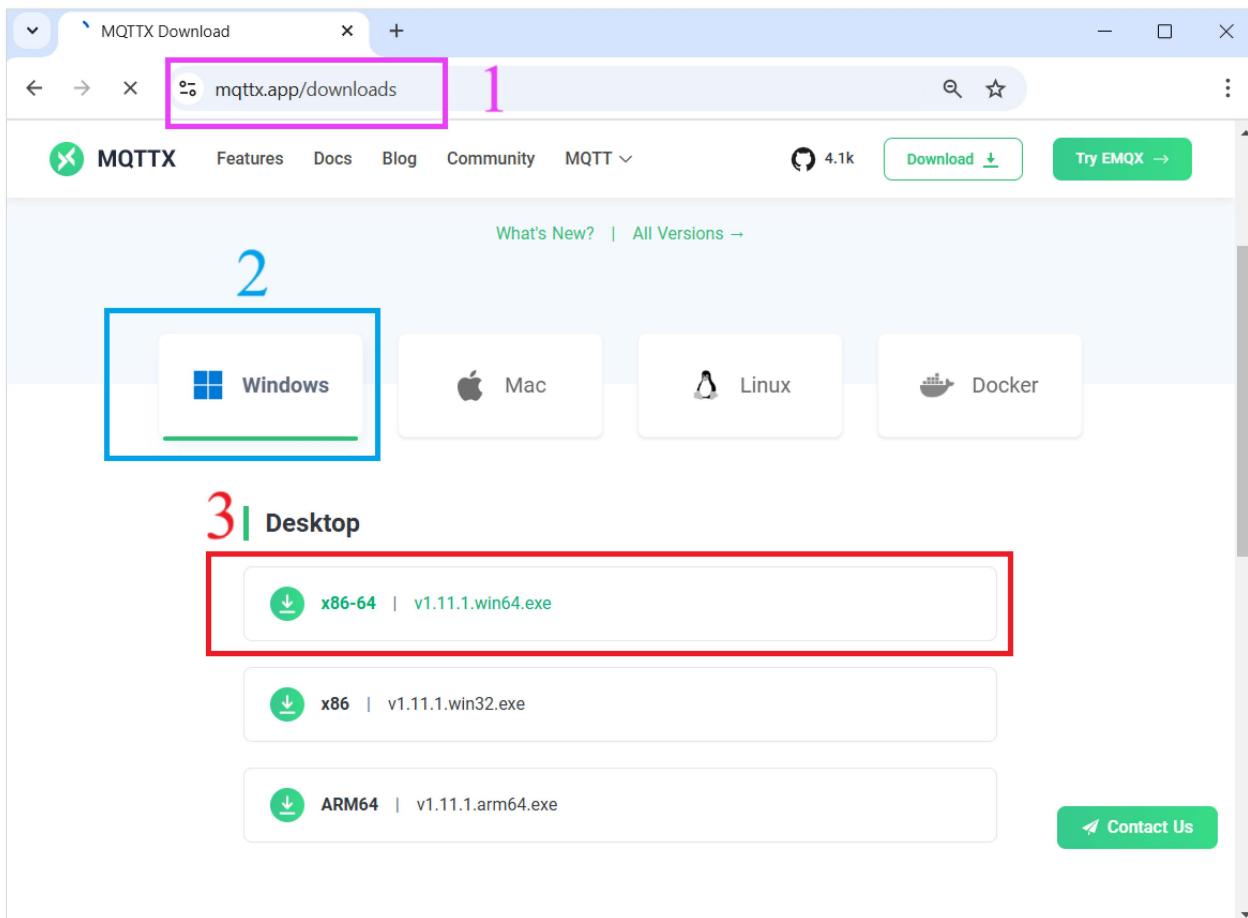


ภาพที่ 5 เว็บไซต์ [Hivemq.com](https://www.hivemq.com/)

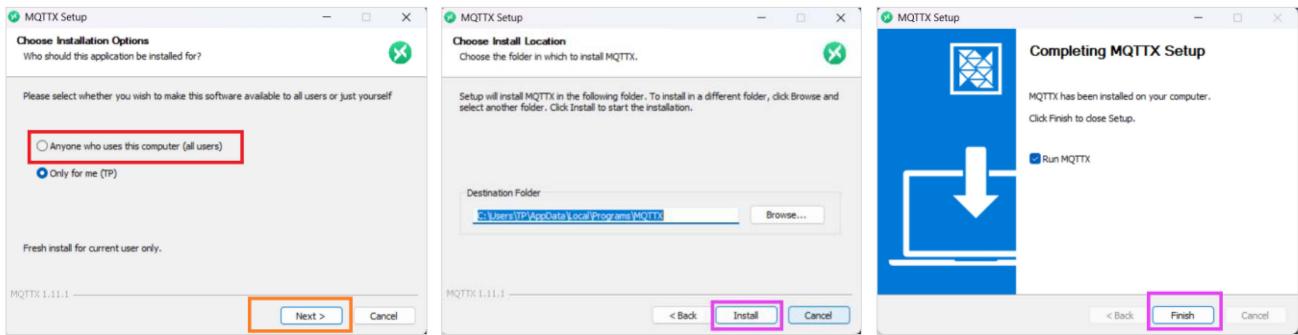


ภาพที่ 6 ข้อมูล MQTT Broker

2. ดาวน์โหลดและติดตั้ง MQTTX จากเว็บไซต์ <https://mqtx.app/downloads> MQTTX เป็นเครื่องมือสำหรับทดสอบการทำงานของ MQTT (ภาพที่ 7) ติดตั้ง MQTTX ดังภาพที่ 8

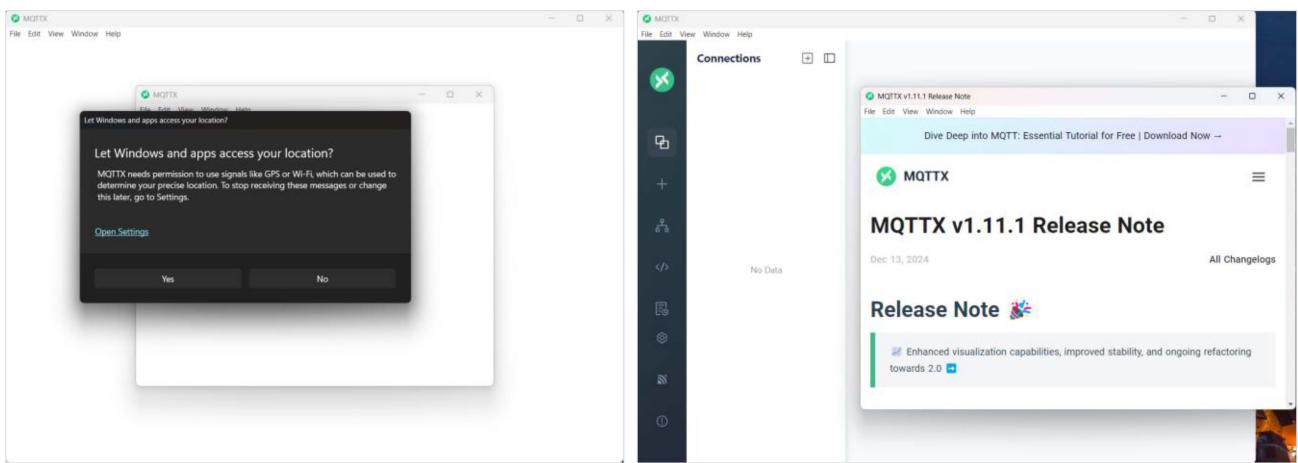


ภาพที่ 7 เว็บไซต์ MQTTX



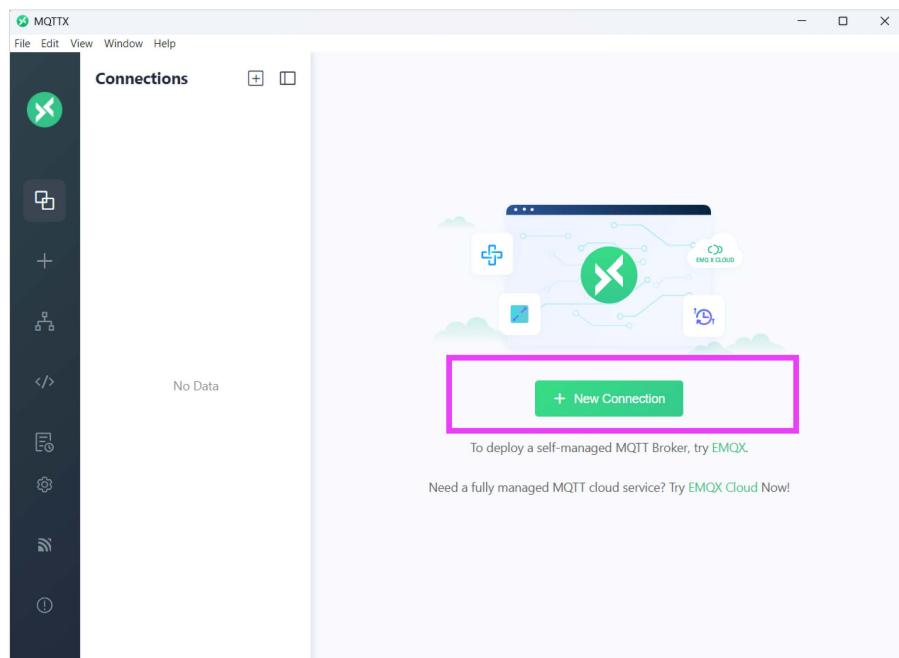
ภาพที่ 8 ติดตั้ง MQTTX

เมื่อติดตั้งเสร็จแล้วเปิด MQTTX ครั้งแรกจะปรากฏดังภาพที่ 9

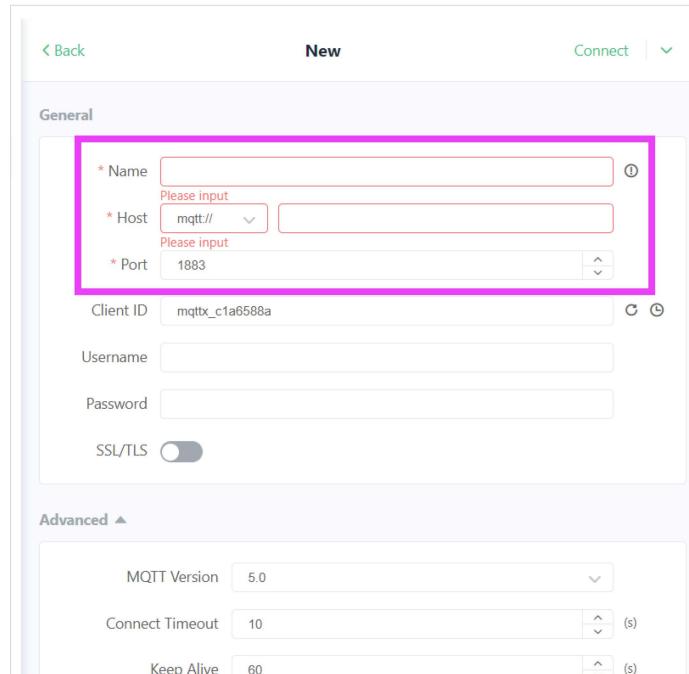


ภาพที่ 9 โปรแกรม MQTTX

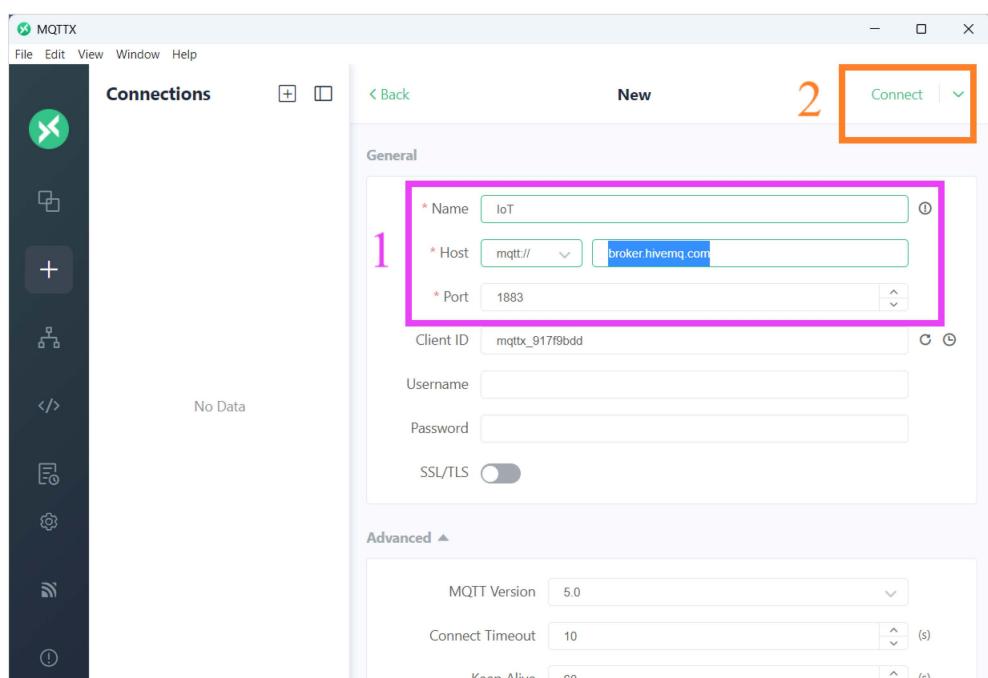
3. ทดลองใช้งาน MQTTX โดยคลิกที่ + New Connection ดังภาพที่ 10 จะปรากฏหน้าจอของ MQTTX ดังภาพที่ 11 จากนั้นกรอกรายละเอียดให้ครบถ้วน Name, Host และ Port ดังภาพที่ 12 จากนั้นคลิก Connect



ภาพที่ 10 + New Connection

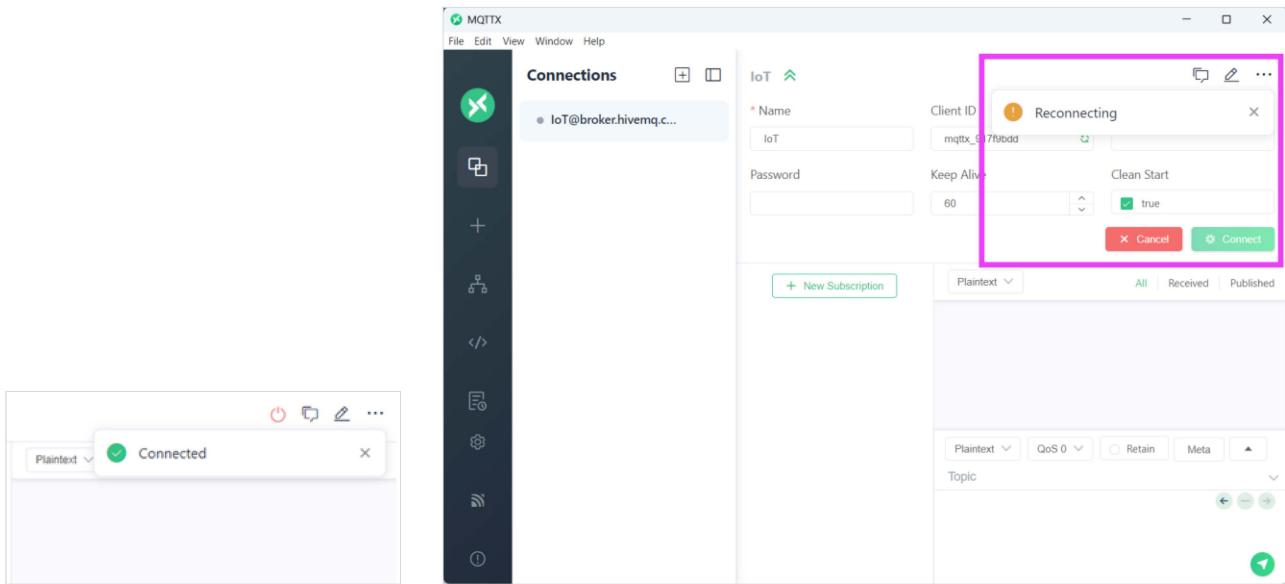


ภาพที่ 11 กรอกรายละเอียดเพื่อเพิ่ม + New Connection



ภาพที่ 12 กรอกรายละเอียดและเชื่อมต่อ (broker.hivemq.com)

หากเชื่อมต่อสำเร็จจะปรากฏคำว่า Connected ดังภาพที่ 13 (ก) แต่ถ้าเชื่อมต่อไม่สำเร็จระบบจะพยายามเชื่อมต่อ ดังภาพที่ 13 (ข)

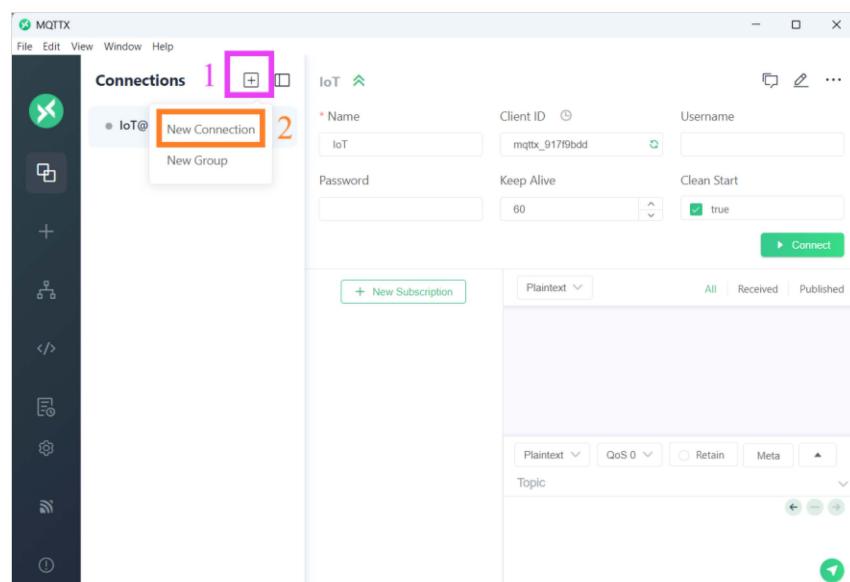


ก. สำเร็จ

ข. ไม่สำเร็จ

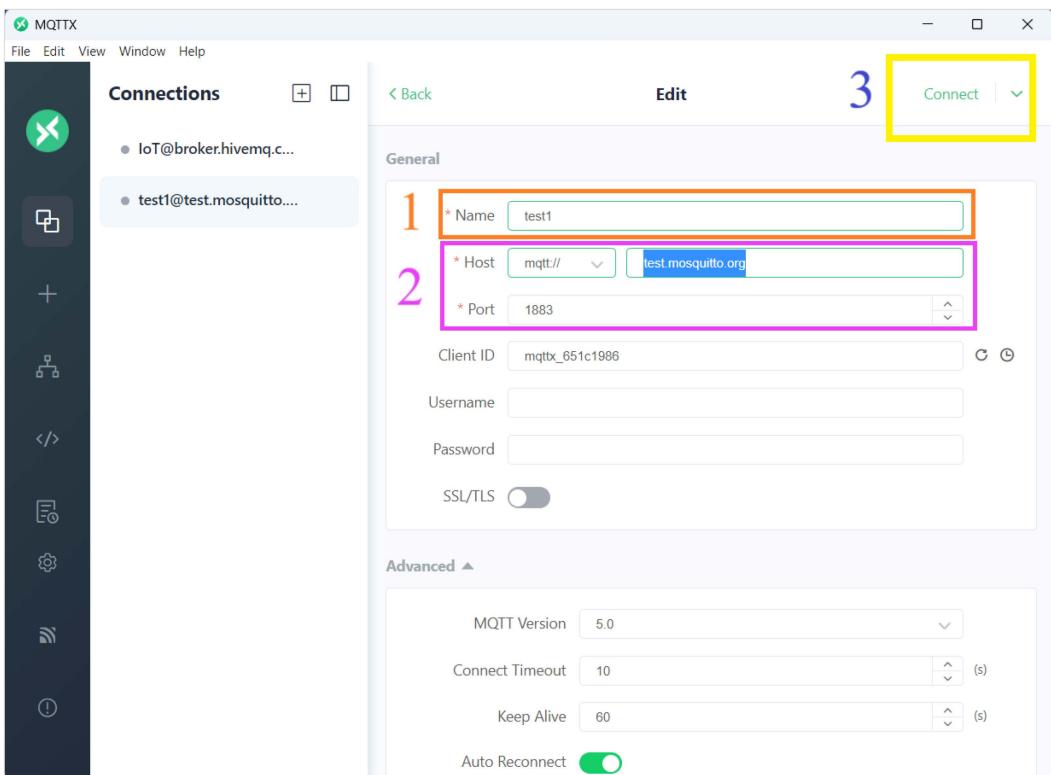
ภาพที่ 13 สถานะการเชื่อมต่อ

หากไม่สามารถเชื่อมต่อกับ Hivemq ได้ให้เปลี่ยนไปใช้ Mosquitto แทน โดยการเพิ่ม Connection ดังภาพที่ 14

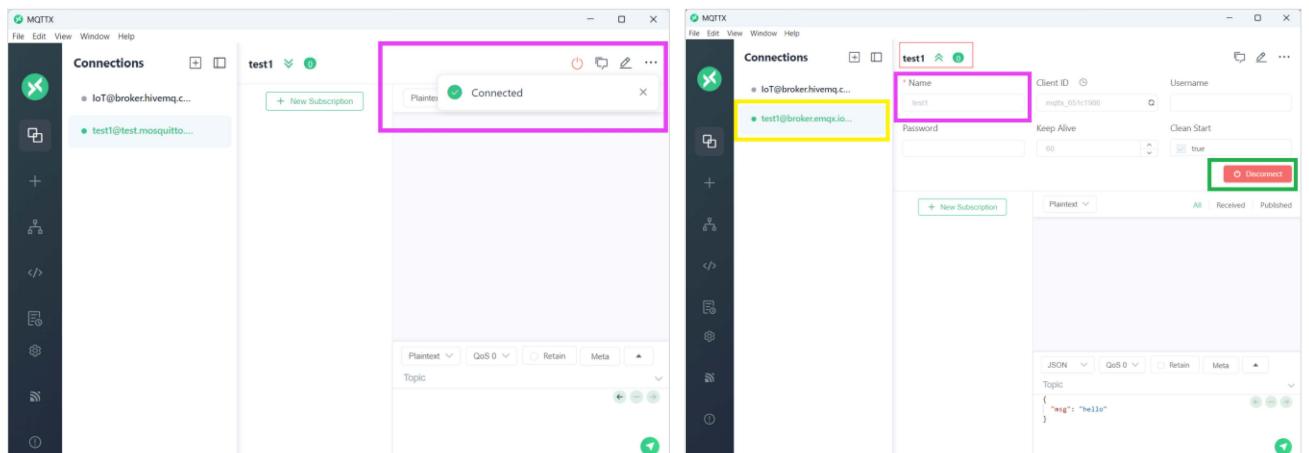


ภาพที่ 14 การเพิ่ม Connection

กรอกรายละเอียดให้ครบถ้วน Name, Host และ Port ดังภาพที่ 15 จากนั้นคลิก Connect และหากสำเร็จจะแสดง Connected ดังภาพที่ 16



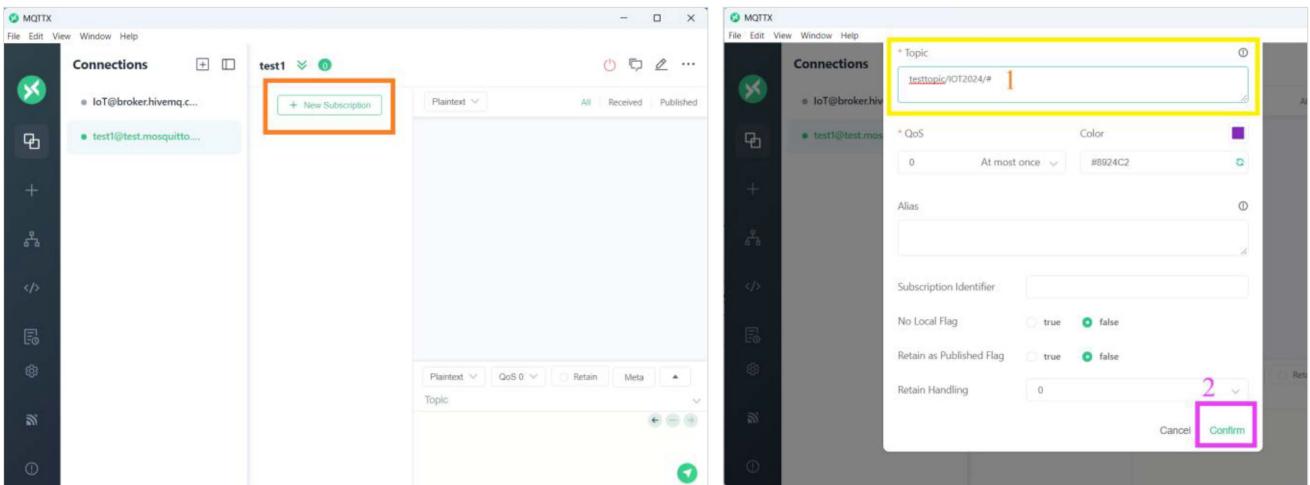
ภาพที่ 15 กรอกรายละเอียดและเชื่อมต่อ (test.mosquitto.org)



ภาพที่ 16 สถานะการเชื่อมต่อสำเร็จ

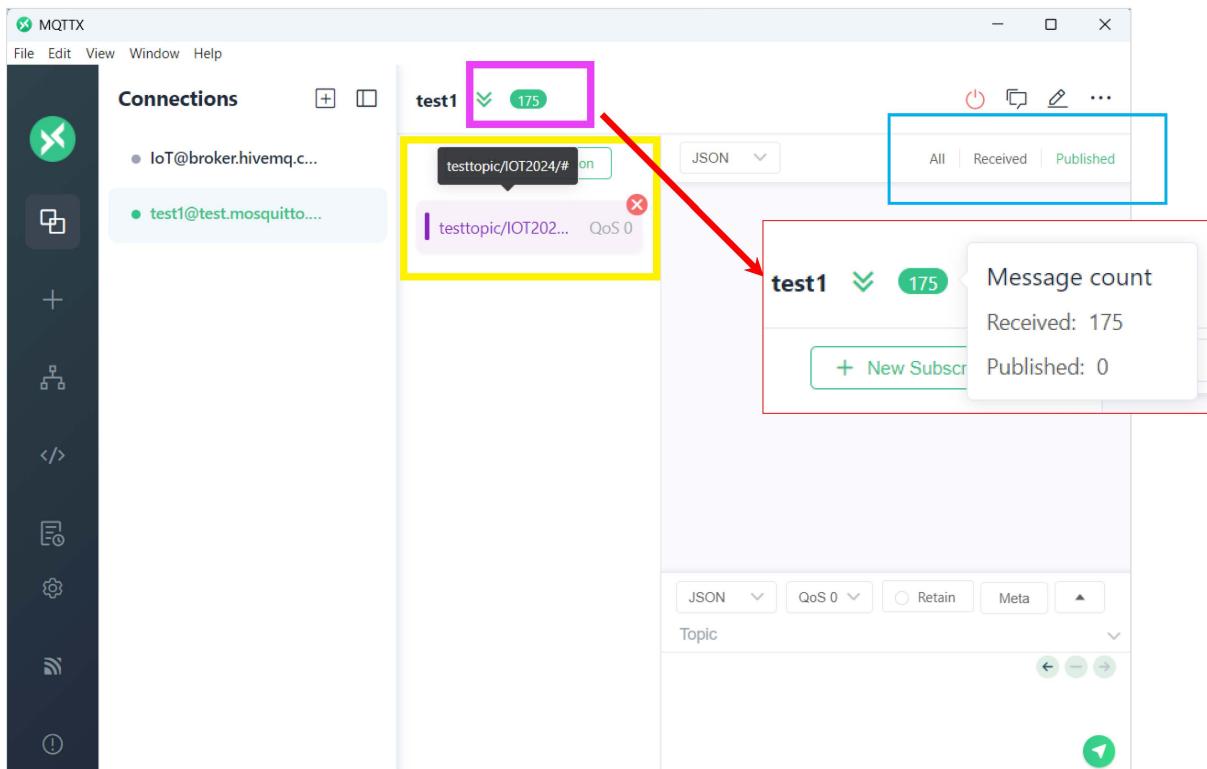
4. การทดลองนี้จะใช้ test.mosquitto.org

5. ทดสอบการรับส่งข้อมูลระหว่าง Publisher และ Subscriber โดยสร้าง New Subscription และตั้งชื่อ Topic เป็น testtopic/IOT2024/# คลิกที่ ดังภาพที่ 17 จากนั้นกรอกรายละเอียดของหัวข้อที่ต้องการ subscribe



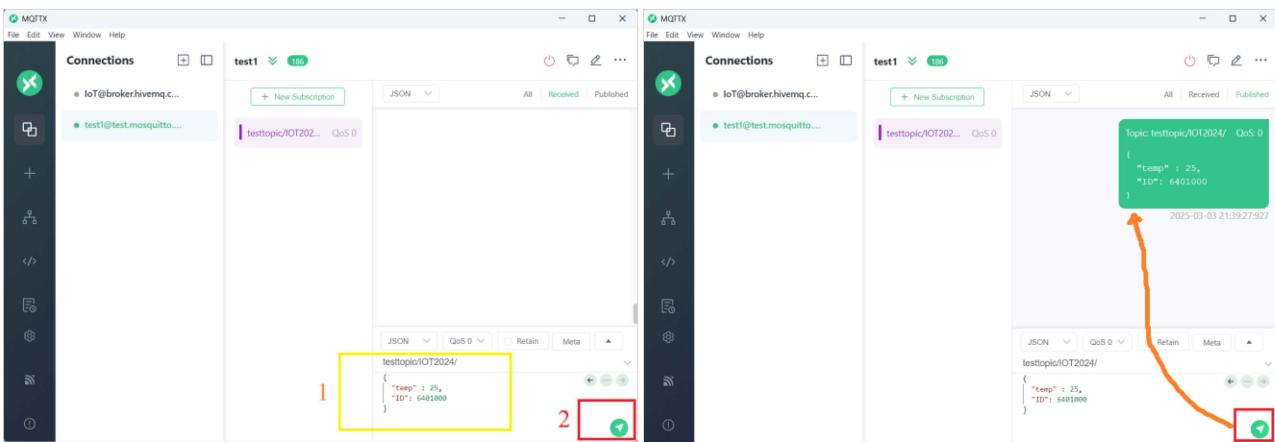
ภาพที่ 17 สร้าง New Subscription

เมื่อสร้างหัวข้อเสร็จจะปรากฏหัวข้อที่ subscribe ดังภาพที่ 18 และด้านบนหัวข้อจะปรากฏตัวเลขที่แสดงจำนวนข้อความที่รับและส่ง และด้านขวาเมื่อ สามารถเลือกได้จะดูข้อมูลทั้งหมด หรือจาก Received หรือจาก Published

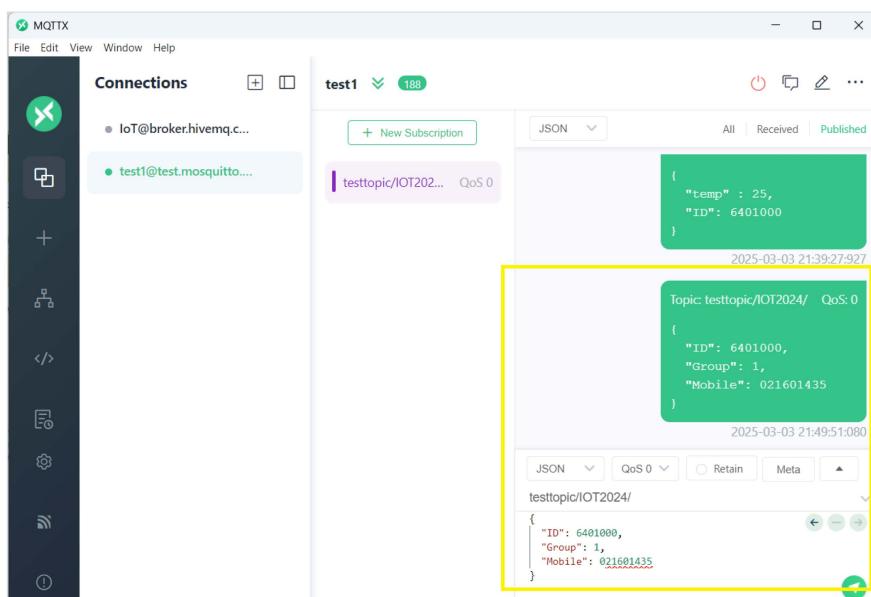


ภาพที่ 18 สร้างหัวข้อ testtopic/IOT2024/#

จากนั้นทดลอง publish ข้อมูลภายใต้หัวข้อ testtopic/IOT2024/ ดังภาพที่ 19 และภาพที่ 20 แสดงการ publish รหัสนักศึกษา ชื่อกลุ่ม และเบอร์โทรศัพท์

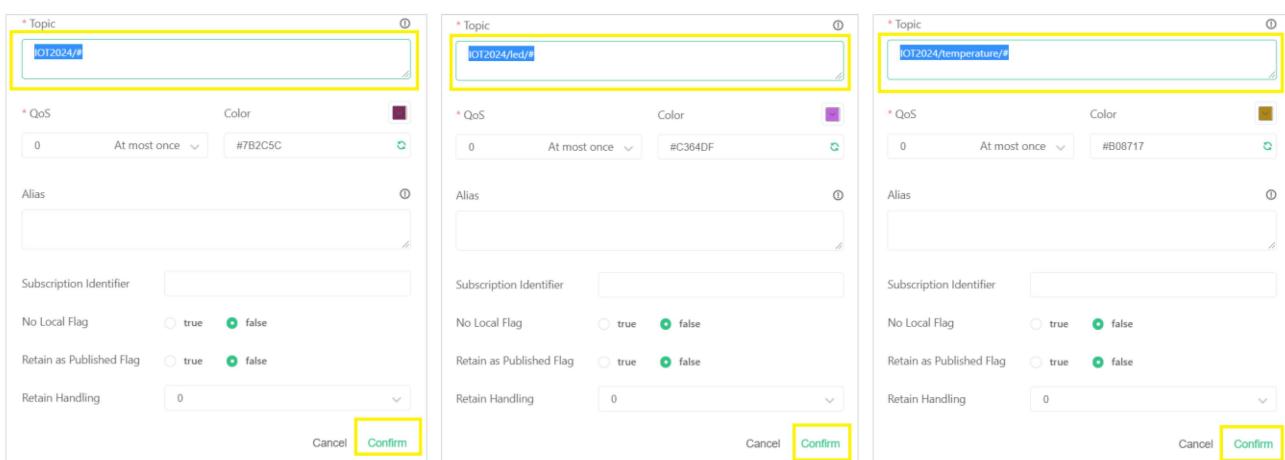


ภาพที่ 19 การ publish ข้อมูลในหัวข้อ testtopic/IOT2024/

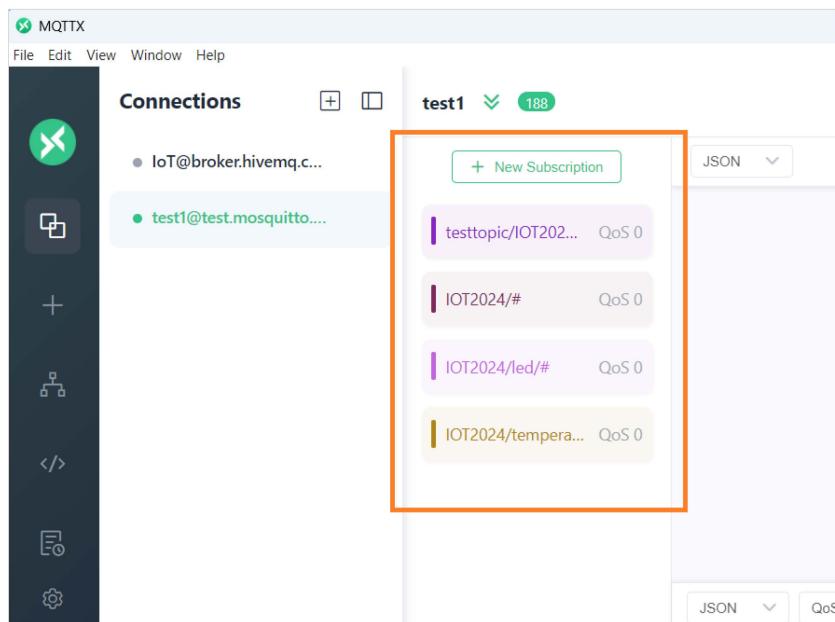


ภาพที่ 20 การ publish ข้อมูลและหน้าต่างของ published

6. 既然สร้าง 3 หัวข้อ ได้แก่ IOT2024 (สำหรับดูทั้งหมด), IOT2024/led (สำหรับดูเฉพาะ led) และ IOT2024/temperature (สำหรับดูเฉพาะ temperature) ตั้งภาพที่ 21 既然หน้าจอของ MQTTX จะแสดงดังภาพที่ 22

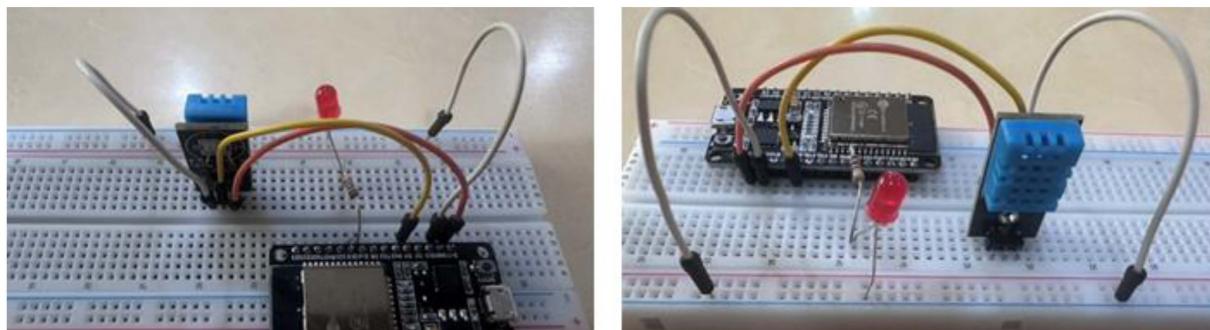


ภาพที่ 21 สร้างหัวข้อทั้งสาม



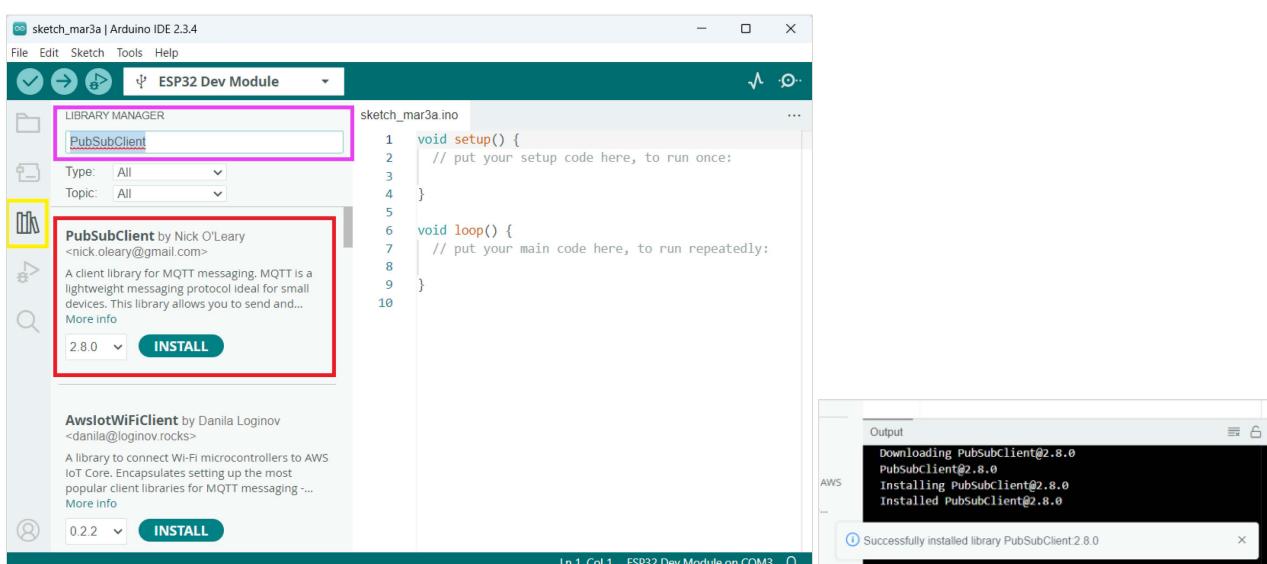
ภาพที่ 22 MQTTX ที่สร้างหัวข้อทั้งสาม

7. วงจรสำหรับการทดลอง แสดงดังภาพที่ 23 (สายสีขาว GND, สายสีเหลือง OUT ขา 4 และสายสีส้ม 3.3 V)



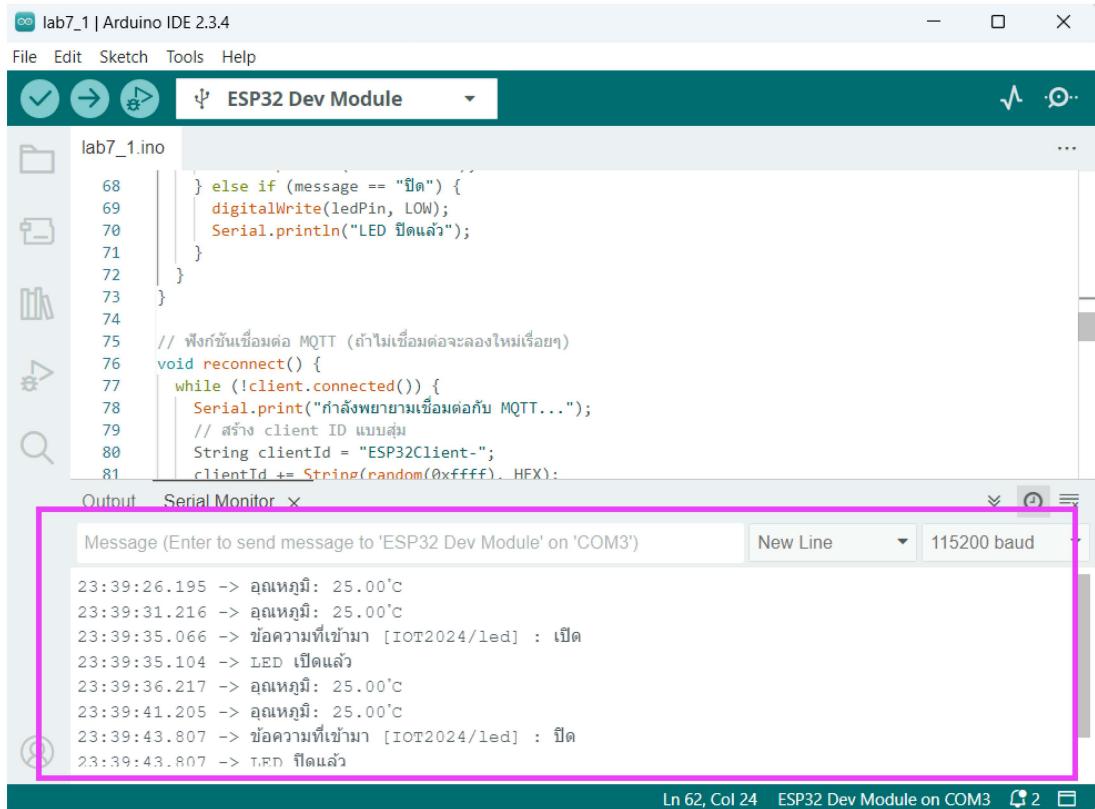
ภาพที่ 23 วงจรสำหรับการทดลอง

8. ติดตั้งไลบรารี PubSubClient ดังภาพที่ 24



ภาพที่ 24 ติดตั้งไลบรารี PubSubClient

9. เขียนโปรแกรมการรับส่งข้อมูลระหว่าง ESP32 ผ่าน MQTT ไปยัง MQTTOX เพื่อทดสอบผลลัพธ์ (ดูภาคผนวก) ตัวอย่างผลการทดลองหน้าจอ serial monitor แสดงบนภาพที่ 25 (บันทึกผลการทดลอง ใช้รูปร่วมกันได้ รายงานเดียวทำส่วนบุคคล)



```

lab7_1.ino

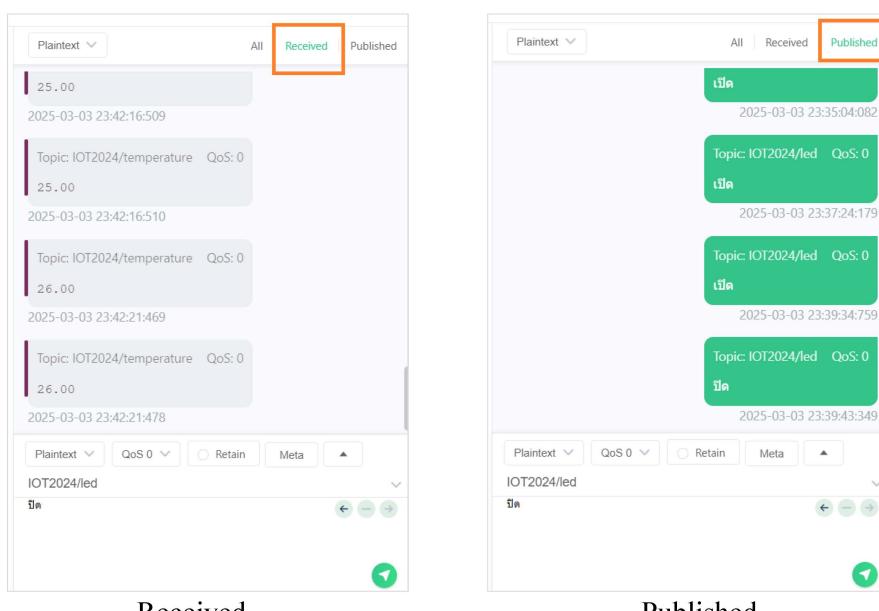
68 } else if (message == "ปิด") {
69     digitalWrite(ledPin, LOW);
70     Serial.println("LED ปิดแล้ว");
71 }
72 }
73 }
74
75 // ฟังก์ชันเชื่อมต่อ MQTT (ถ้าไม่เชื่อมต่อจะลองใหม่เรื่อยๆ)
76 void reconnect() {
77     while (!client.connected()) {
78         Serial.print("กำลังพยายามเชื่อมต่อกับ MQTT...");
79         // สร้าง client ID แบบสุ่ม
80         String clientId = "ESP32Client-";
81         clientId += String(random(0xffff), HEX);

```

Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')  
23:39:26.195 -> อุณหภูมิ: 25.00°C  
23:39:31.216 -> อุณหภูมิ: 25.00°C  
23:39:35.066 -> ข้อความที่เข้ามา [IOT2024/led] : เปิด  
23:39:35.104 -> LED เปิดแล้ว  
23:39:36.217 -> อุณหภูมิ: 25.00°C  
23:39:41.205 -> อุณหภูมิ: 25.00°C  
23:39:43.807 -> ข้อความที่เข้ามา [IOT2024/led] : ปิด  
23:39:43.807 -> LED ปิดแล้ว

ภาพที่ 25 หน้าจอ serial monitor

10. ส่งข้อมูลจาก MQTTOX ไปยัง ESP32 เพื่อทดสอบการควบคุม LED (ทดสอบการรับข้อมูลของ ESP32) เขียนที่ Plaintext และ topic: IOT2024/led ดังภาพที่ 26



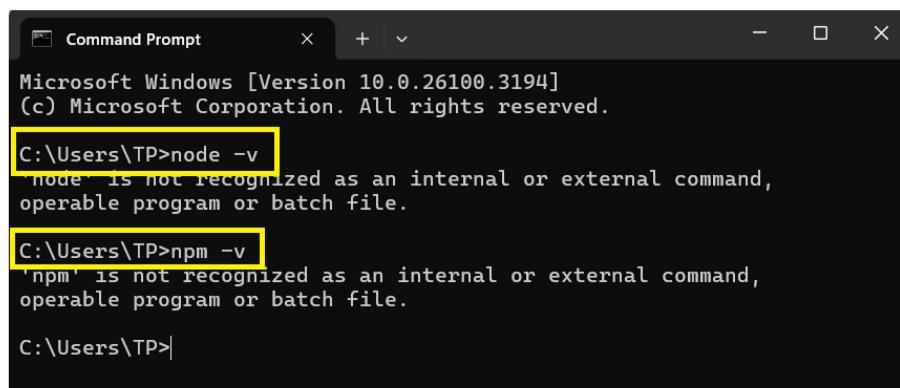
ภาพที่ 26 ส่งข้อความจาก MQTTOX ไปยัง ESP32

11. บันทึกผลการทดลอง หน้าจอจากผลการทดลองทั้งหมด รวมถึงจดบันทึกในการทดลอง (ความสว่างของ LED) รูปไปใช้กลุ่ม รายงานเดียว

## การทดลองที่ 7.2 การติดตั้งและใช้งาน Node-RED

Node-RED เป็นแพลตฟอร์มที่พัฒนาโดย IBM ให้บริการข้อมูลในรูปแบบเว็บไซต์ โดยออกแบบและพัฒนาที่ผู้ใช้สามารถสร้าง workflow ได้ง่ายโดยการลากและวางการเชื่อมต่อหนึ่ง ในการทดลองนี้ประกอบด้วยสองส่วนที่สำคัญคือ การติดตั้งและการใช้งาน Node-RED รายละเอียดดังต่อไปนี้

- ก่อนติดตั้ง Node-RED จะตรวจสอบว่า Node.js และ npm บนเครื่องหรือไม่ โดยเปิด Command prompt (cmd) หรือ PowerShell และใช้คำสั่งดังในภาพที่ 27



```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

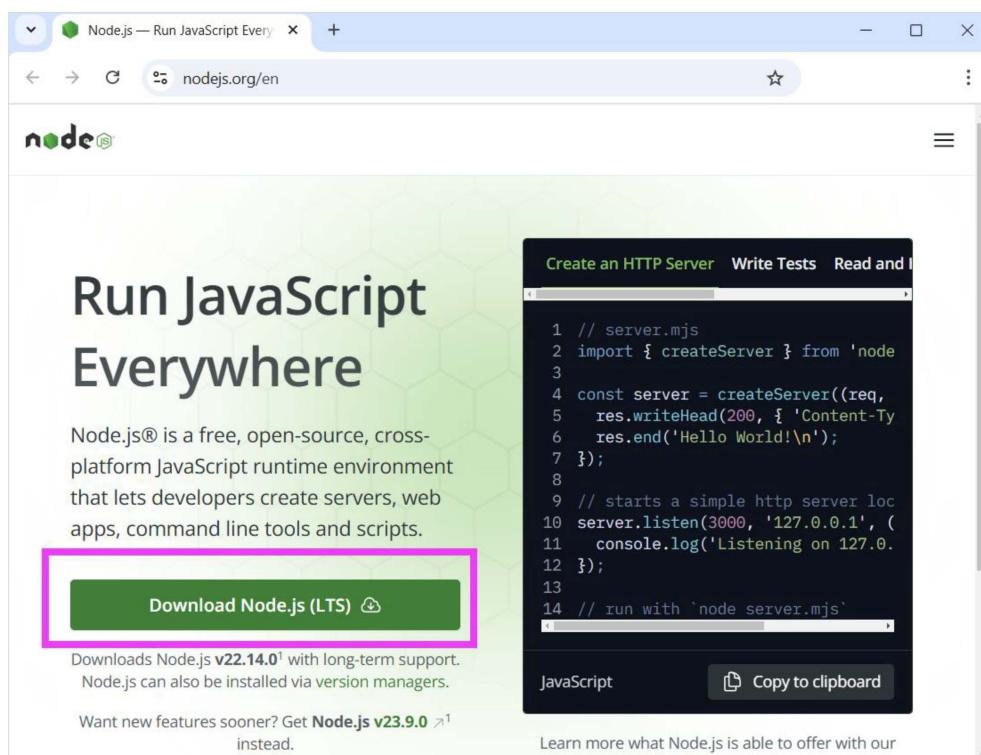
C:\Users\TP>node -v
'node' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\TP>npm -v
'npm' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\TP>
```

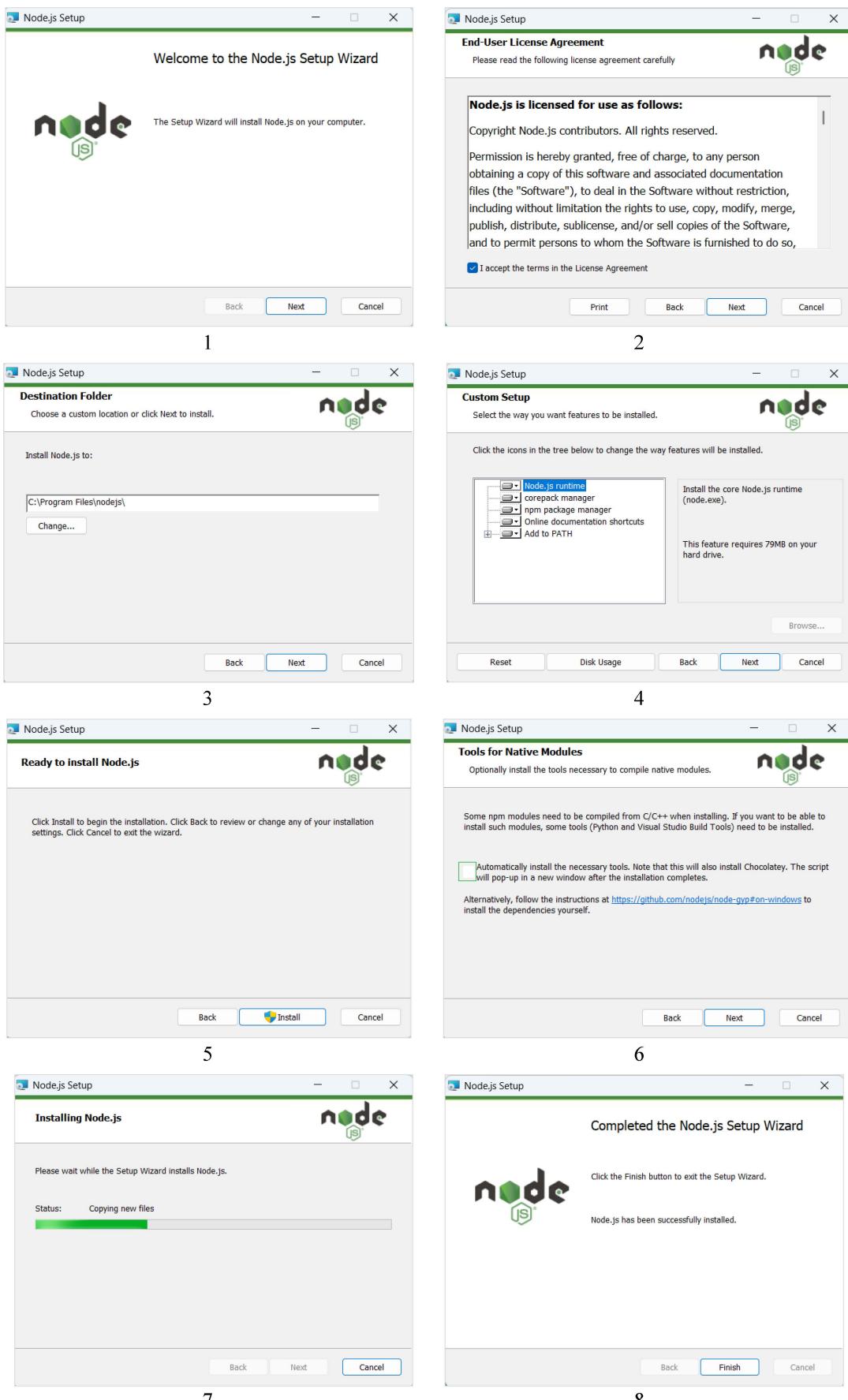
ภาพที่ 27 ตรวจสอบการติดตั้งโปรแกรม Node.js

ถ้าไม่พบให้ติดตั้ง Node.js โดยดาวน์โหลดจากเว็บไซต์ <https://nodejs.org/en> ดังภาพที่ 28



ภาพที่ 28 ดาวน์โหลด Node.js

## ติดตั้ง Node.js ตาม Wizard ดังภาพที่ 29



2. เปิด Command prompt ตรวจสอบการติดตั้ง Node.js และ npm (กรอบสีแดง ภาพที่ 30) จากนั้นติดตั้ง node-red ผ่าน npm (กรอบสีเหลือง ภาพที่ 30)

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TP>node -v
v22.14.0

C:\Users\TP>npm -v
10.9.2

C:\Users\TP>npm install -g --unsafe-perm node-red
/|
```

ภาพที่ 30 ติดตั้ง Node.js ผ่าน npm

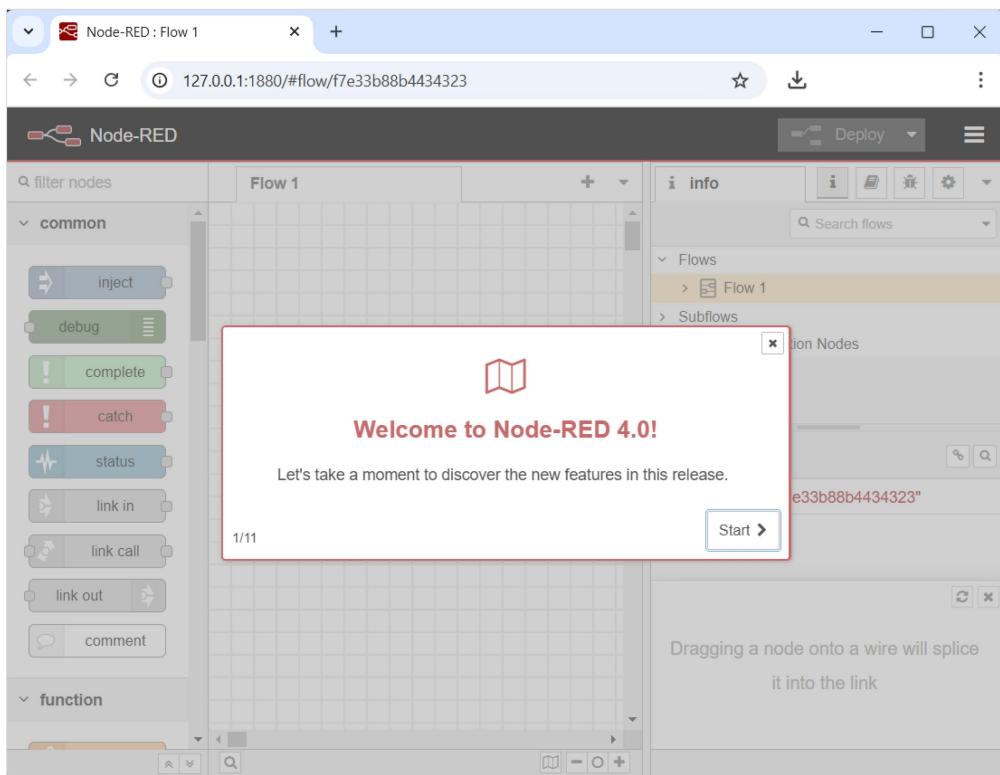
3. ตรวจสอบการติดตั้งและเปิดใช้งาน Node-RED โดยไปที่ Command prompt และกรอก node-red (กรอบสีส้ม ภาพที่ 31) และใช้ลิ้งในกรอบสีเขียวไปเปิดที่เบราว์เซอร์เพื่อใช้งาน Node-RED (ภาพที่ 32) และภาพที่ 33 แสดงภาพรวมของ Node-RED

```
C:\Users\TP>node-red
4 Mar 01:00:27 [info] Welcome to Node-RED
=====
4 Mar 01:00:27 - [info] Node-RED version: v4.0.9
4 Mar 01:00:27 - [info] Node.js version: v22.14.0
4 Mar 01:00:27 - [info] Windows_NT 10.0.26100 x64 LE
4 Mar 01:00:28 - [info] Loading palette nodes
4 Mar 01:00:30 - [info] Settings file : C:\Users\TP\.node-red\settings.js
4 Mar 01:00:30 - [info] Context store : 'default' [module=memory]
4 Mar 01:00:30 - [info] User directory : C:\Users\TP\.node-red
4 Mar 01:00:30 - [warn] Projects disabled : editorTheme.projects.enabled=false
4 Mar 01:00:30 - [info] Flows file : C:\Users\TP\.node-red\flows.json
4 Mar 01:00:30 - [info] Creating new flow file
4 Mar 01:00:30 - [warn]
-----
Your flow credentials file is encrypted using a system-generated key.

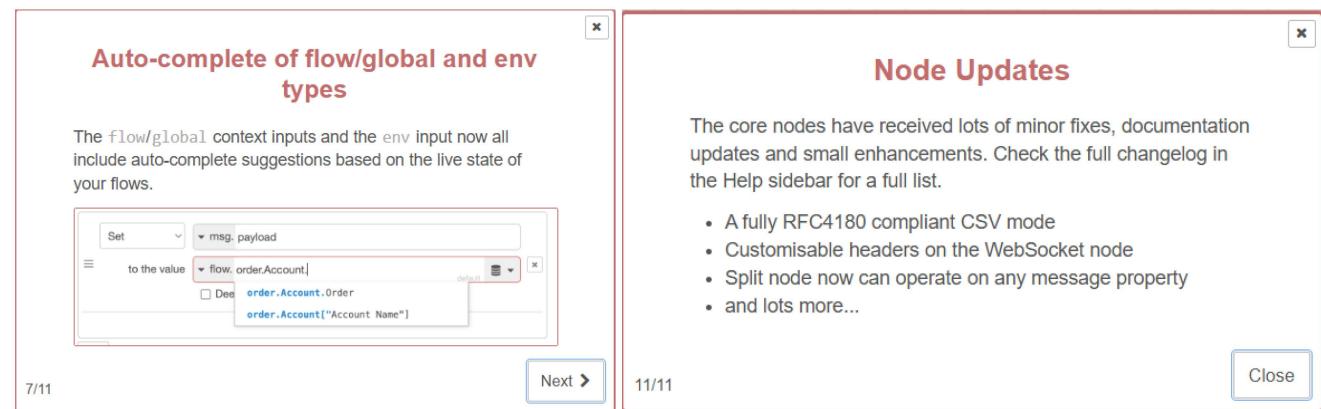
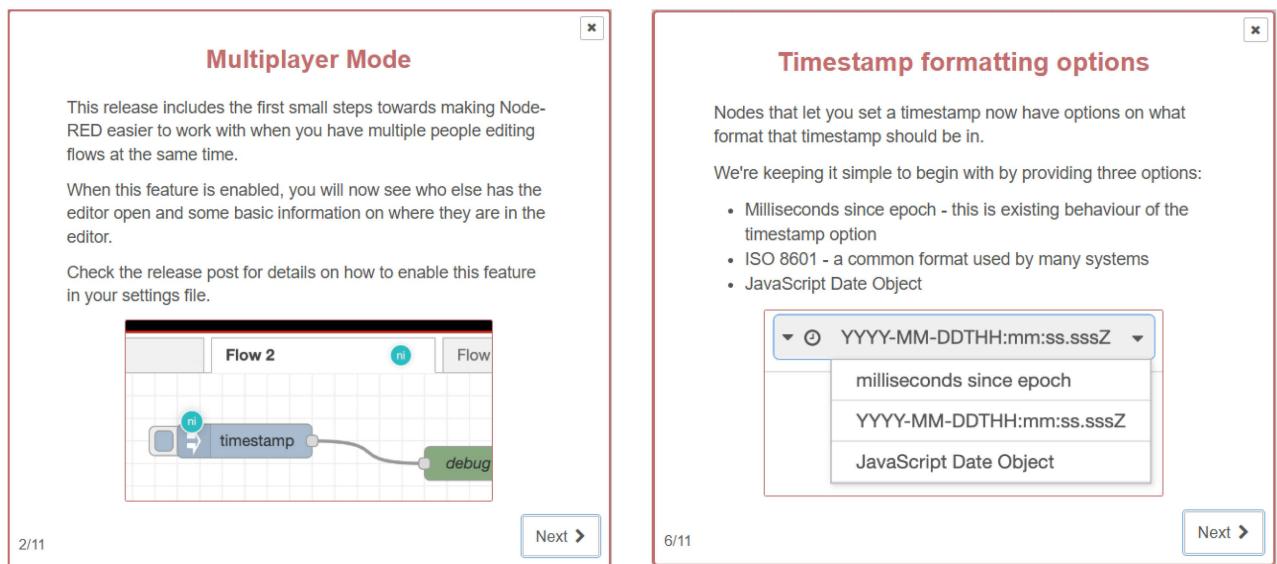
If the system-generated key is lost for any reason, your credentials file will not be recoverable, you will have to delete it and re-enter your credentials.

You should set your own key using the 'credentialSecret' option in your settings file. Node-RED will then re-encrypt your credentials file using your chosen key the next time you deploy a change.
-----
4 Mar 01:00:30 - [info] Server now running at http://127.0.0.1:1880/
4 Mar 01:00:30 - [warn] Encrypted credentials not found
4 Mar 01:00:30 - [info] Starting flows
4 Mar 01:00:30 - [info] Started flows
|
```

ภาพที่ 31 ตรวจสอบการติดตั้งและเปิดใช้งาน Node-RED

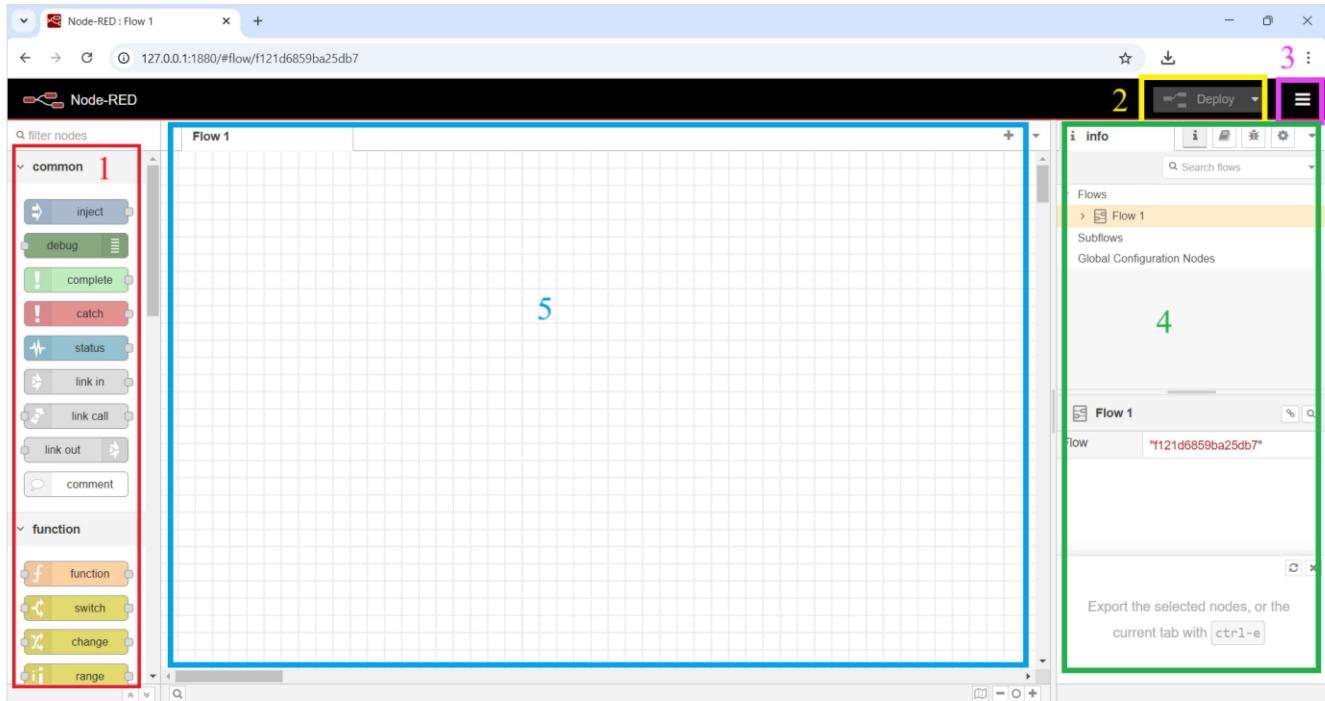


ภาพที่ 32 Node-RED



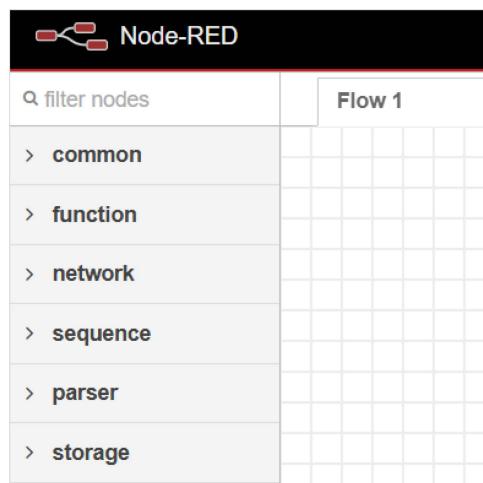
ภาพที่ 33 Node-RED

Node-RED ประกอบด้วย 5 ส่วนที่สำคัญคือ 1) Development tools 2) Deploy tools 3) System menu 4) System palettes และ 5) Flows แสดงในภาพที่ 34



ภาพที่ 34 Node-RED

Development tools เป็นเมนูเครื่องมือด้านซ้ายมือของ Node-RED เรียกว่า โหนด ในแบบเครื่องมือนี้ประกอบไปด้วย แบบค้นหา (filter nodes อยู่ด้านบนสุด สำหรับค้นหาโหนด) common, function, network, sequence, parser และ storage



ภาพที่ 35 Development tools

ในการทดลองนี้จะอธิบาย 3 tools ที่สำคัญ คือ common, function และ network (จะอธิบาย tools อื่นเพิ่มเติมในการทดลองต่อไป) ส่วนประกอบ (โหนด) ของ tools ทั้งสามแสดงดังภาพที่ 36

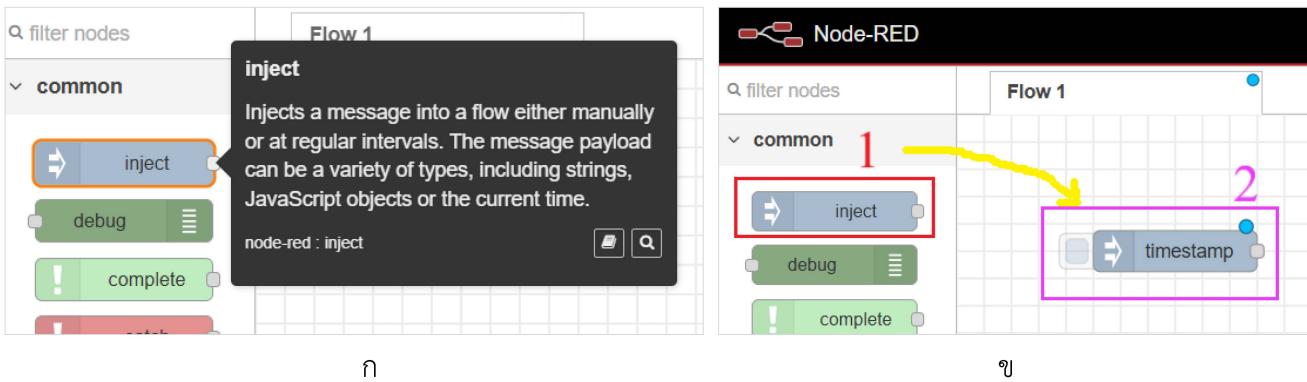


ภาพที่ 36 โหนดของ Tools ต่างๆ

Common เป็นโหนดคำสั่งพื้นฐานเพื่อควบคุมการทำงานและ debug การทำงานของ Flow

- inject ใช้กำหนดค่าเริ่มต้นเข้า Flow ตามช่วงเวลาหรือแบบกดปุ่ม
- debug ใช้แสดงค่าข้อมูลที่เหล่าน Flow ใน Debug Window
- complete ใช้ติดตามสถานะของ Node ที่กำลังทำงาน
- catch ใช้จับข้อผิดพลาดของ Flow และจัดการ Error Handling
- status ใช้ติดตามสถานะของ Node และแสดงผลลัพธ์ที่เปลี่ยนไป
- link in / link out ใช้เชื่อมต่อระหว่าง Flow ต่าง ๆ
- link call ใช้เรียก Flow อื่น ๆ แบบ synchronous
  - comment ใช้เพิ่มข้อความอธิบายใน Flow

การสร้างโหนดบน Flow สามารถทำได้โดยการลากแล้ววาง (รวมถึงการเคลื่อนย้ายตำแหน่งหลังจากวางไปแล้ว) ตัวอย่าง เช่น ที่ common หากต้องการนำโหนด inject มาวางบน Flow 1 (ภาพที่ 37 ก) ทำได้โดยการคลิกเลือกที่ inject (1, ภาพที่ 37 ข) จากนั้นลากมาวางที่ flow และปล่อย (2, ภาพที่ 37 ข)



ภาพที่ 37 Development tools

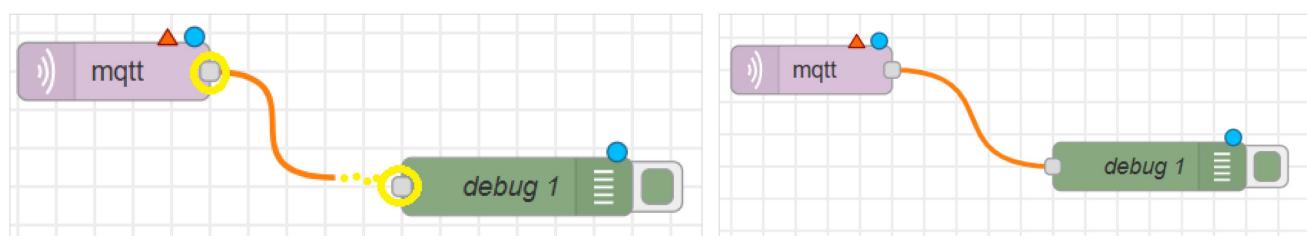
Network เป็นเทคโนโลยีที่งานในการสื่อสารและเชื่อมต่ออุปกรณ์ IoT และเครือข่ายคอมพิวเตอร์

- mqtt in/mqtt out ใช้รับและส่งข้อมูลผ่านโปรโตคอล MQTT สำหรับ IoT
- http in/http response/http request ใช้สร้าง API เพื่อรับส่งข้อมูลผ่าน HTTP
- websocket in/websocket out ใช้เชื่อมต่อ WebSocket สำหรับรับส่งข้อมูลแบบ Real-time
- tcp in/tcp out/tcp request ใช้สื่อสารผ่านโปรโตคอล TCP
- udp in/udp out ใช้สื่อสารผ่านโปรโตคอล UDP

Function เป็นหนึ่งในเครื่องมือในการประมวลผลและควบคุมการดำเนินงานของระบบ

- function ใช้เขียนโค้ด JavaScript เพื่อประมวลผลข้อมูล
- switch ใช้ควบคุมเส้นทางของข้อมูลตามเงื่อนไขที่กำหนด
- change ใช้เปลี่ยนค่าใน message เช่น เปลี่ยน msg.payload
- range ใช้ปรับค่าของข้อมูลจากช่วงหนึ่งไปยังอีกช่วงหนึ่ง
- template ใช้สร้างข้อความหรือ HTML ตามข้อมูลที่รับเข้า
- delay ใช้หน่วงเวลาหรือควบคุมอัตราการส่งข้อมูล
- trigger ใช้สร้างการทำงานตามเงื่อนไข เช่น ส่งค่าอกมาหนึ่งครั้งหลังจากได้รับอินพุต
- exec ใช้รันคำสั่ง Shell หรือ Terminal
- filter ใช้คัดกรองข้อมูลที่ผ่าน Flow

ตัวอย่างการวางโนําติดใน Flow แสดงดังภาพที่ 38

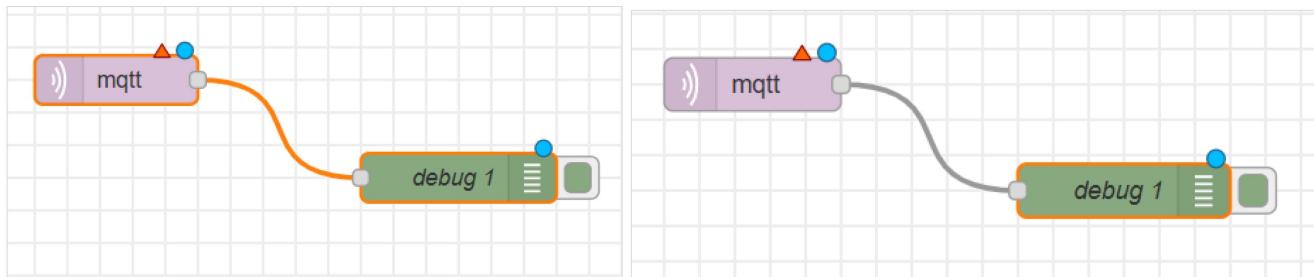


ก ลากจากจุดเริ่มต้นไปวางจุดสุดท้าย

ภาพที่ 38 การเชื่อมต่อโนําติด

ข เสร็จสิ้น

ภาพที่ 39 แสดงการเลือกโหนดและลิ้งใน Flow โดยภาพที่ 39 (ก) เลือกทั้งหมดทำได้โดยใช้เม้าส์ลากให้ครอบคลุมโหนดและลิ้งทั้งหมด (จะปรากฏว่ามีกรอบ) ภาพที่ 39 (ข) เลือกเฉพาะโหนด debug 1 เมื่อเลือกโหนดและลิ้งแล้วสามารถทำการคัดลอก วาง หรือตัดได้สะดวก



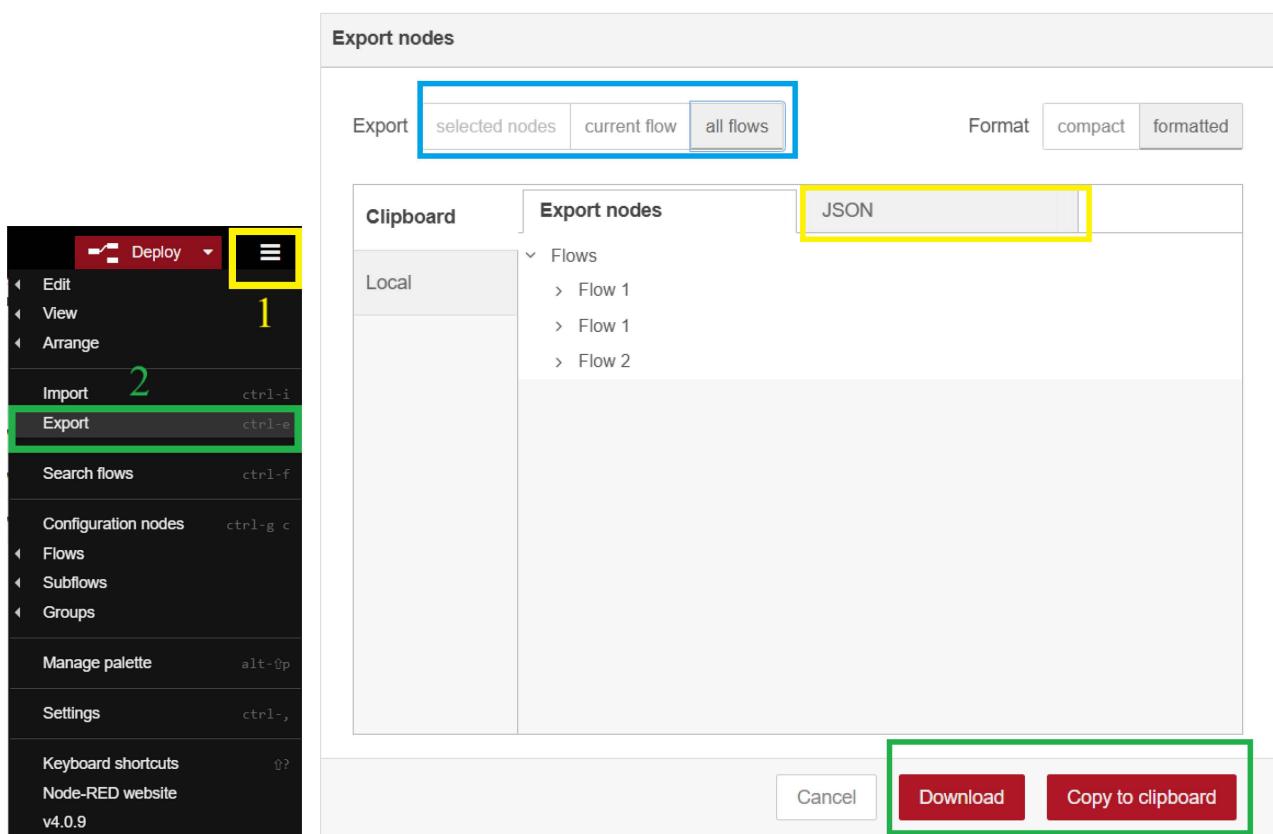
ก เลือกทั้งหมด

ข เลือก debug 1

ภาพที่ 39 การวางแผน

4. ให้นักศึกษาเขียนอธิบายผลการศึกษาการใช้งาน Node-RED (งานเดี่ยว รายบุคคล) เนื้อหาในการทดลองนี้ และทำรายงานส่ง

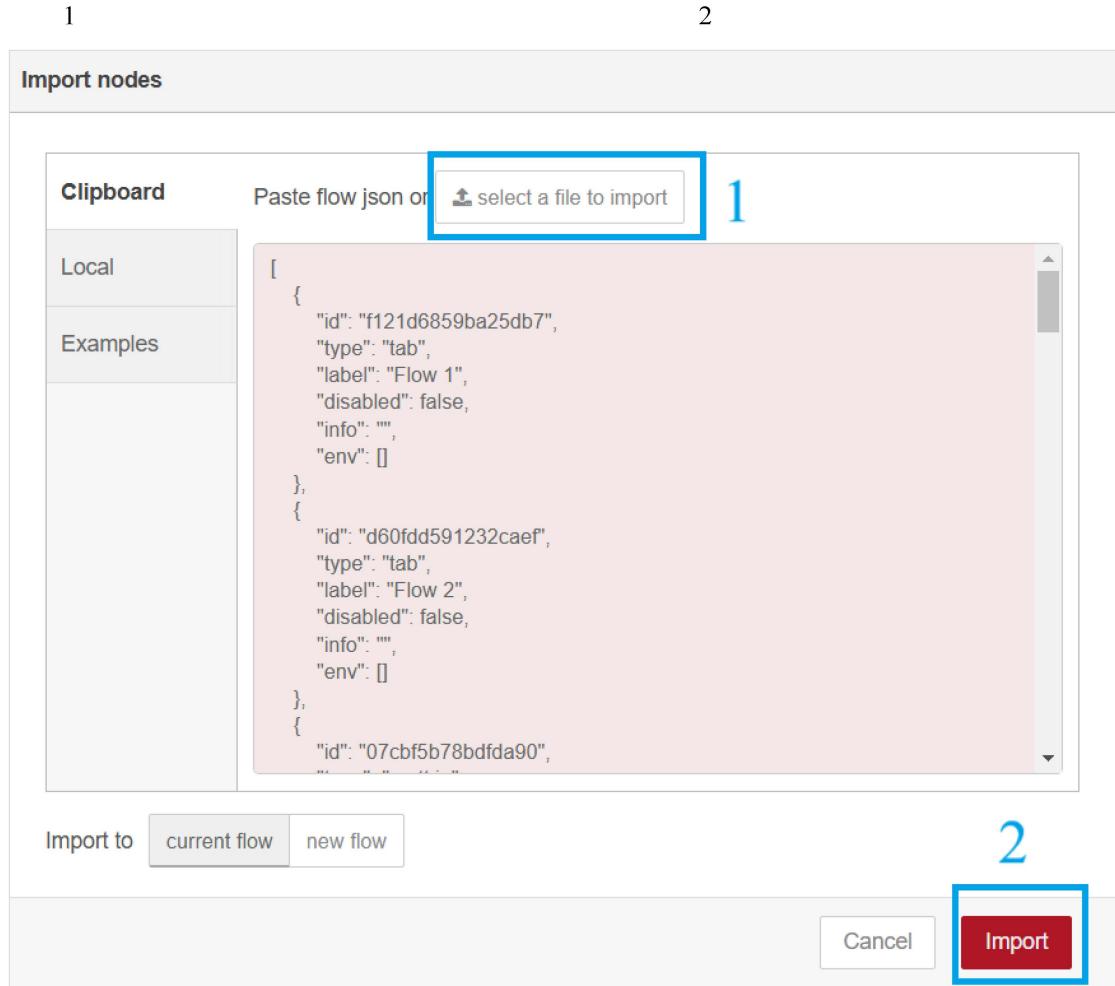
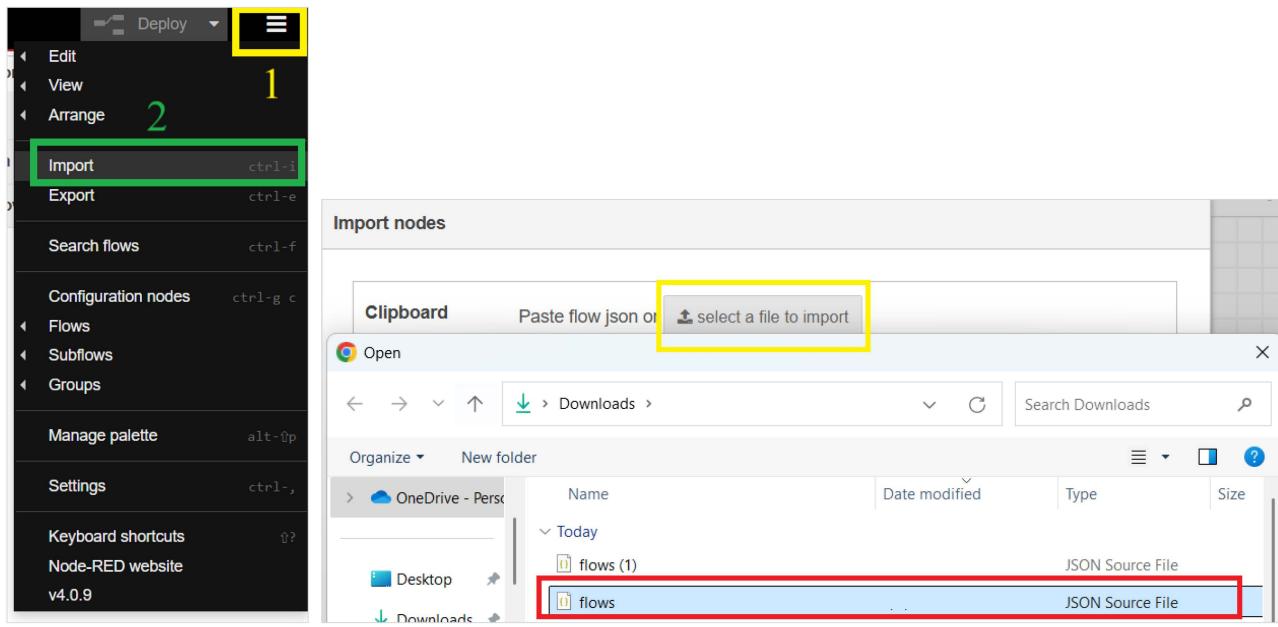
5. การดาวน์โหลดไฟล์และอัปโหลดไฟล์ จะใช้ System menu โดยเลือก สำหรับดาวน์โหลด และเลือก สำหรับอัปโหลด รายละเอียดดังภาพที่ 40



1

2

ภาพที่ 40 การดาวน์โหลดไฟล์



ภาพที่ 41 การอัปโหลดไฟล์

6. มอบหมายให้ไปศึกษาการใช้งาน Node-RED อีก 1 ครั้ง ก่อนทำการทดลองครั้งถัดไป

## ภาคผนวก

โปรแกรมสำหรับการทดลองที่ 7.1 แสดงในภาพที่ ผ.1

lab7\_1.ino

```
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3  #include <DHT.h>
4
5  // กำหนดค่า WiFi
6  const char* ssid = "CSD3204";
7  const char* password = "csd3204y25";
8
9  // กำหนดค่า MQTT Broker
10 const char* mqtt_server = "test.mosquitto.org";
11 const int mqtt_port = 1883;           // พอร์ต MQTT โดยปกติคือ 1883
12 const char* mqtt_user = "";         // ถ้าใช้ authentication
13 const char* mqtt_password = "";     // ถ้าใช้ authentication
14
15 // กำหนดพินสำหรับ sensor DHT11 และ LED
16 #define DHTPIN 4                  // ขาสำหรับเชื่อมต่อ DHT11
17 #define DHTTYPE DHT11            // กำหนดชนิดของ sensor เป็น DHT11
18 DHT dht(DHTPIN, DHTTYPE);
19
20 const int ledPin = 19;           // ขา LED สำหรับควบคุม
21
22 WiFiClient espClient;
23 PubSubClient client(espClient);
24
25 // สำหรับการส่งข้อมูลทุก 5 วินาที
26 unsigned long lastMsg = 0;
27 const long interval = 5000;    // หน่วงเวลา 5,000 ms = 5 วินาที
28
29 // พิงก์ชันเชื่อมต่อ WiFi
30 void setup_wifi() {
31     delay(10);
32     Serial.println();
33     Serial.print("กำลังเชื่อมต่อ WiFi: ");
34     Serial.println(ssid);
35
36     WiFi.begin(ssid, password);
37
38     while (WiFi.status() != WL_CONNECTED) {
39         delay(500);
40         Serial.print(".");
41     }
}
```

ภาพที่ ผ.1 โปรแกรมสำหรับการทดลองที่ 7.1

## lab7\_1.ino

```
42
43     Serial.println();
44     Serial.println("เชื่อมต่อ WiFi สำเร็จแล้ว");
45     Serial.print("ที่อยู่ IP: ");
46     Serial.println(WiFi.localIP());
47 }
48
49 // ฟังก์ชัน callback สำหรับรับข้อความจาก MQTT
50 void callback(char* topic, byte* payload, unsigned int length) {
51     Serial.print("ข้อความที่เข้ามา [");
52     Serial.print(topic);
53     Serial.print("] : ");
54
55     // สร้าง String จาก payload
56     String message;
57     for (unsigned int i = 0; i < length; i++) {
58         message += (char)payload[i];
59     }
60     Serial.println(message);
61
62     // ตรวจสอบว่าข้อความมาจาก topic ที่ควบคุม LED หรือไม่
63     if (String(topic) == "IOT2024/led") {
64         if (message == "เปิด") {
65             digitalWrite(ledPin, HIGH);
66             Serial.println("LED เปิดแล้ว");
67         } else if (message == "ปิด") {
68             digitalWrite(ledPin, LOW);
69             Serial.println("LED ปิดแล้ว");
70         }
71     }
72 }
73
74 // ฟังก์ชันเชื่อมต่อ MQTT (ถ้าไม่เชื่อมต่อจะลองใหม่เรื่อยๆ)
75 void reconnect() {
76     while (!client.connected()) {
77         Serial.print("กำลังพยายามเชื่อมตอกับ MQTT...");
78         // สร้าง client ID แบบสุ่ม
79         String clientId = "ESP32Client-";
80         clientId += String(random(0xffff), HEX);
81         if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
82             Serial.println("เชื่อมตอกับ MQTT สำเร็จแล้ว");
83             // subscribe topic สำหรับควบคุม LED
```

ภาพที่ ผ.1 โปรแกรมสำหรับการทดลองที่ 7.1 (ต่อ)

### lab7\_1.ino

```
81     if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
82         Serial.println("เชื่อมต่อ MQTT สำเร็จแล้ว");
83         // subscribe topic สำหรับควบคุม LED
84         client.subscribe("IOT2024/led");
85     } else {
86         Serial.print("ไม่สามารถเชื่อมต่อ, รหัสข้อผิดพลาด = ");
87         Serial.print(client.state());
88         Serial.println(" - ลองใหม่ใน 5 วินาที");
89         delay(5000);
90     }
91 }
92 }

93 void setup() {
94     // กำหนดพิน LED เป็น OUTPUT
95     pinMode(ledPin, OUTPUT);
96     digitalWrite(ledPin, LOW);
97
98     // เริ่ม Serial สำหรับ debug
99     Serial.begin(115200);
100
101    // เริ่มเชื่อมต่อ WiFi
102    setup_wifi();
103
104    // เริ่มใช้งาน DHT sensor
105    dht.begin();
106
107    // กำหนด MQTT Broker และ callback
108    client.setServer(mqtt_server, mqtt_port);
109    client.setCallback(callback);
110 }
111

112 void loop() {
113     // ตรวจสอบการเชื่อมต่อ MQTT
114     if (!client.connected()) {
115         reconnect();
116     }
117     client.loop();
118
119     // ตรวจสอบเวลาเพื่อส่งข้อมูลทุก 5 วินาที
120     unsigned long now = millis();
121     if (now - lastMsg > interval) {
```

ภาพที่ ผ.1 โปรแกรมสำหรับการทดลองที่ 7.1 (ต่อ)

### lab7\_1.ino

```
102     setup_wifi(),
103
104     // เริ่มใช้งาน DHT sensor
105     dht.begin();
106
107     // กำหนด MQTT Broker และ callback
108     client.setServer(mqtt_server, mqtt_port);
109     client.setCallback(callback);
110
111 }
112
113 void loop() {
114     // ตรวจสอบการเชื่อมต่อ MQTT
115     if (!client.connected()) {
116         reconnect();
117     }
118     client.loop();
119
120     // ตรวจสอบเวลาเพื่อส่งข้อมูลทุก 5 วินาที
121     unsigned long now = millis();
122     if (now - lastMsg > interval) {
123         lastMsg = now;
124
125         // อ่านค่าอุณหภูมิจาก DHT11
126         float temperature = dht.readTemperature();
127
128         // ตรวจสอบว่าอ่านค่าได้หรือไม่
129         if (isnan(temperature)) {
130             Serial.println("ไม่สามารถอ่านค่าอุณหภูมิจาก DHT11");
131             return;
132         }
133
134         // แปลงค่าอุณหภูมิเป็น String (2 ตำแหน่งหลังทศนิยม)
135         String tempStr = String(temperature, 2);
136         Serial.print("อุณหภูมิ: ");
137         Serial.print(tempStr);
138         Serial.println(" °C");
139
140         // ส่งค่าอุณหภูมิไปยัง MQTT Topic "IOT2024/temperature"
141         client.publish("IOT2024/temperature", tempStr.c_str());
142     }
143 }
144 }
```

ภาพที่ พ.1 โปรแกรมสำหรับการทดลองที่ 7.1 (ต่อ)

## รายงานผลปฏิบัติการทดลอง

ให้นักศึกษาส่งรายงานผลปฏิบัติการทดลองเดี่ยว โดยรายงานประกอบด้วย

1. ผลการทดลองที่ให้บันทึกทั้งหมด
2. วิเคราะห์และสรุปผลการทดลอง
3. ส่งรายงานด้วยไฟล์ pdf
4. จัดรูปแบบรายงานกำหนดไว้ในแลป 1

**กดส่ง submit ภายในเวลา 11:40 + แนบไฟล์(อัพโหลด) ไม่แนบลิ้ง**