

การสร้าง MQTT Server บน Raspberry Pi เพื่อใช้งาน Chatbot LINE ในฟาร์มอัจฉริยะ Chatbot LINE from Raspberry Pi MQTT Server for Smart Farming

3/4 – IoTs with Node-RED on Raspberry Pi

- การใช้งาน ESP32 เพื่อแสดงค่าและควบคุมผ่าน Blynk
- การใช้งาน Raspberry Pi เพื่อแสดงค่าและควบคุมผ่าน Blynk
- การใช้งาน ESP32 เพื่อแสดงค่าและควบคุมผ่าน Public MQTT Server
- การใช้งาน Raspberry Pi เพื่อแสดงค่าและควบคุมผ่าน Public MQTT Server
- การติดตั้ง Private MQTT Server บน Raspberry Pi และการทดสอบ
- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

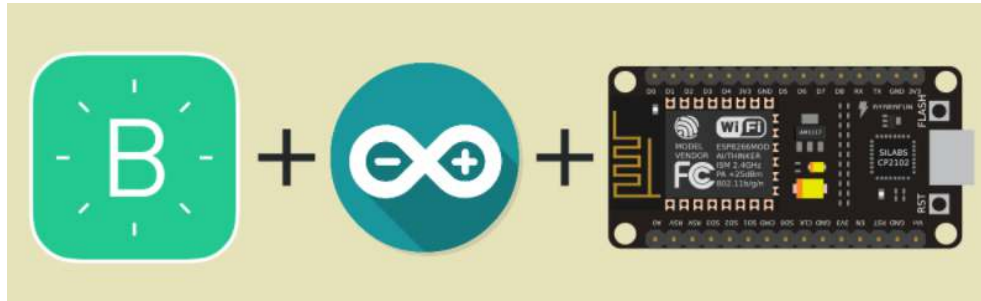


1/6 - การใช้งาน ESP32 เพื่อแสดงค่าและควบคุมผ่าน Blynk

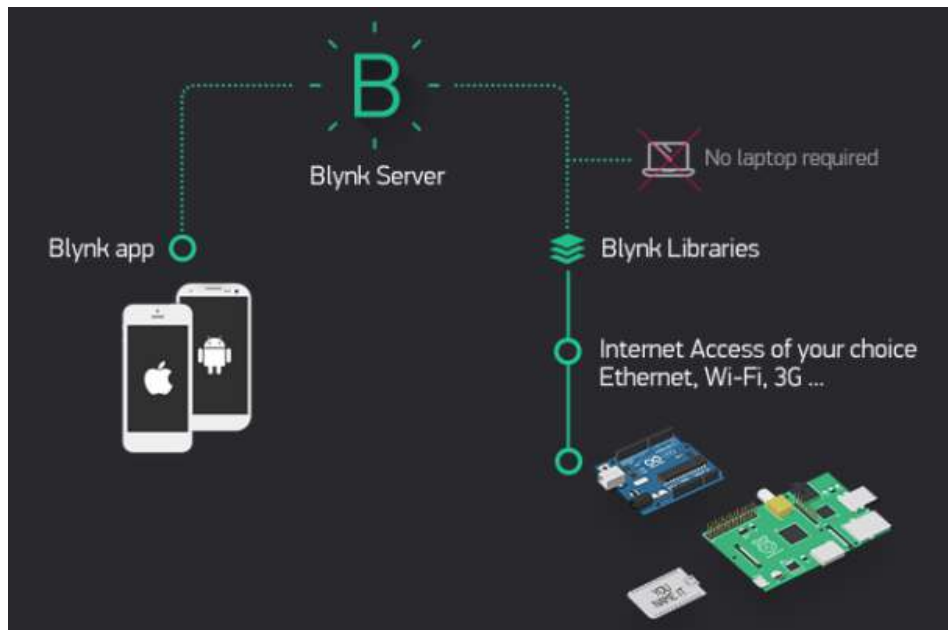
<http://help.blynk.cc/getting-started-library-auth-token-code-examples/blynk-basics/how-to-display-any-sensor-data-in-blynk-app>



Blynk เป็น cloud platform ที่ให้บริการฟรี สำหรับ IOT, Blynk เป็น Application สำเร็จรูปสำหรับงาน IOT มีความน่าสนใจคือการเขียนโปรแกรมที่ง่าย ไม่ต้องเขียน App เองสามารถใช้งานได้จริง Real time สามารถเชื่อมต่อ Device ต่างๆเข้ากับ Internet ได้อย่างง่ายดาย ไม่ว่าจะเป็น Arduino, ESP-8266, ESP-32, Node-MCU, Raspberry Pi นำมาแสดงบน Application ได้อย่างง่ายดาย แล้วที่สำคัญ Application Blynk ยังฟรี และ รองรับในระบบ IOS และ Android อีกด้วย



Blynk สามารถเชื่อมต่ออุปกรณ์ Device ของเราเข้ากับ internet ได้อย่างง่ายดาย ไม่ว่าจะเป็น Arduino, ESP8266, Raspberry pi หรือแม้แต่อื่นๆ ที่รวมเอา widget ต่างๆมาควบคุมแทนการเขียน code ยากๆ ไม่เพียงเท่านั้น ทางเลือกในการเชื่อมต่อเข้ากับ Blynk server เรายังสามารถใช้ได้ทั้ง WiFi และเครือข่ายมือถือ โดยสามารถ Download application นี้ได้ฟรีทั้งระบบ IOS และ Android

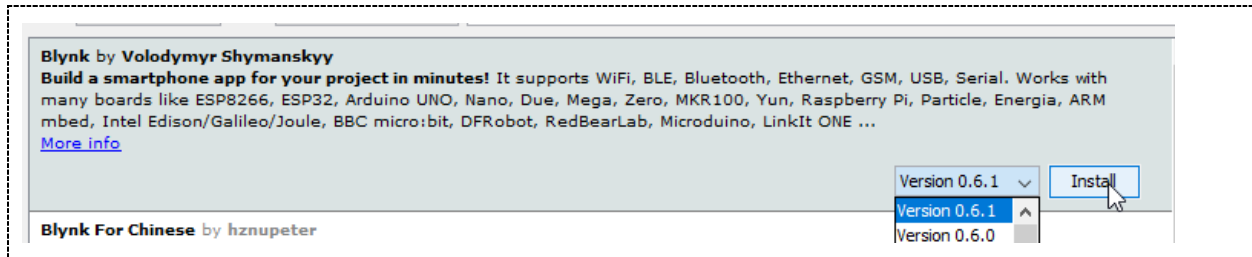


อ่านเพิ่มเติม

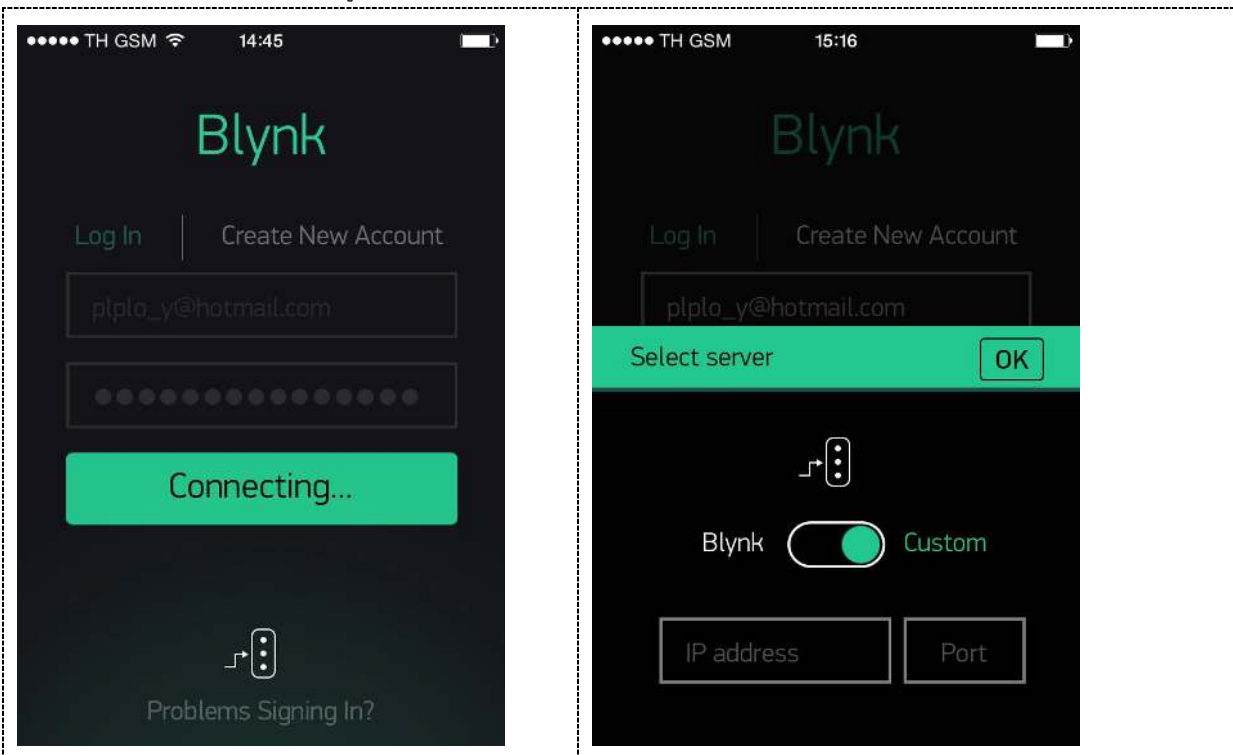
- <https://www.blynk.cc/>, <http://docs.blynk.cc/>
- <http://help.blynk.cc/getting-started-library-auth-token-code-examples/blynk-basics/how-to-display-any-sensor-data-in-blynk-app>
- <https://www.9arduino.com/article/59/app-สำเร็จรูป-blynk-nodemcu-esp8266-ตอนที่-1-blynk-คืออะไร>
- <http://www.ayarafun.com/2015/08/easy-iot-play-with-blynk/>
- <https://github.com/blynkkk/blynk-server>
- <http://thaiopensource.org/มาเล่น-blynk-กับ-esp8266-กัน/>
- <http://suwitkiravittaya.eng.chula.ac.th/B2i2019BookWeb/blynkapp1.html>

Lab301A – Blynk Control ESP32 Output

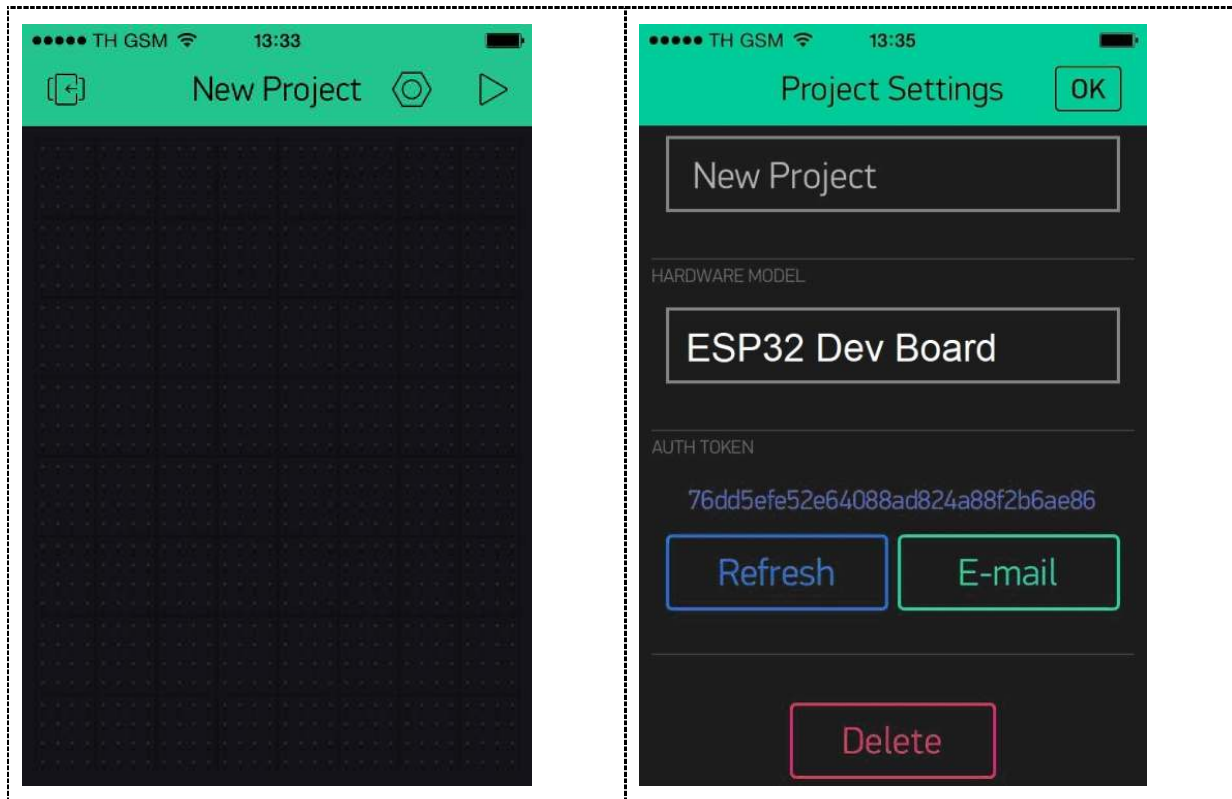
1. ติดตั้ง Arduino IDE และ เพิ่ม ESP32 ใน Arduino IDE ตามเอกสาร “LNE-D31 -- Arduino ESP32 Start.pdf”
1. ติดตั้ง Blynk Application บนมือถือ
2. ติดตั้ง library Blynk เพื่อใช้งานกับ ESP32 บน Arduino IDE เลือกใช้ **Version 0.6.1** หรือติดตั้งจาก <https://github.com/blynkkk/blynk-library/archive/master.zip>



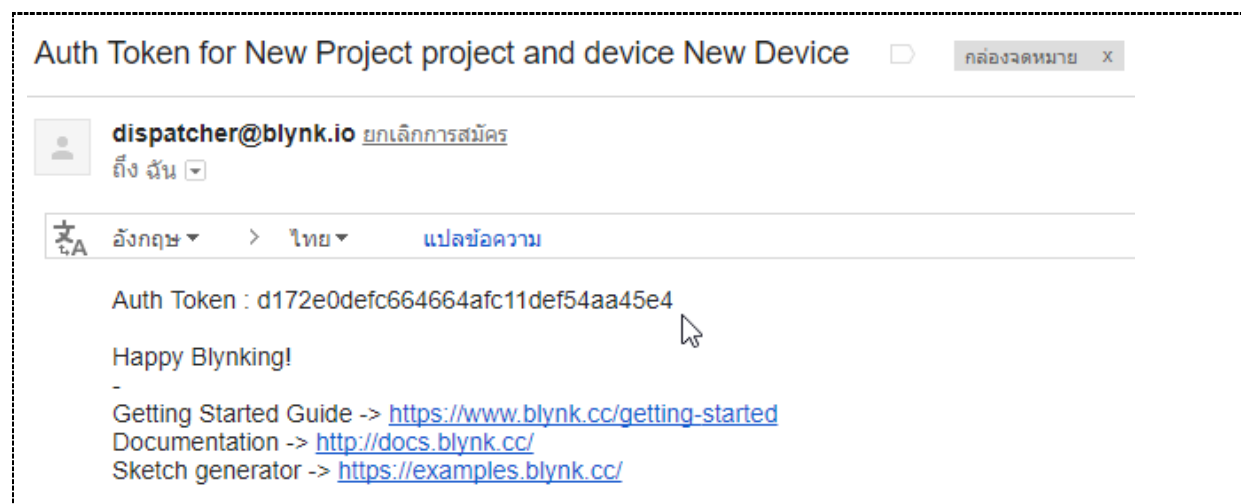
3. เริ่มต้นใช้งาน Blynk เราต้องสมัคร ลงทะเบียน เพื่อใช้งานก่อน ให้เลือกที่คำว่า Create New account เพื่อสร้างการเชื่อมต่อกับ application กับ Email ของผู้ใช้งาน
4. Login เข้า Blynk
5. เลือก Connect เข้ากับ server ของ Blynk (Blynk ให้ผู้ใช้เลือก Custom Server ได้โดยใส่ IP Address ของ Server เราเอง โดยกดที่รูป Problems Signing In แล้วเลื่อน scroll จาก Blynk ไป Custom)



6. ต่อมาเป็นการสร้าง Project ของเราด้วย Blynk ให้กดที่สัญลักษณ์หกเหลี่ยมมุมขวบนเพื่อตั้งค่า
7. โดยในหน้านี้เราสามารถตั้งชื่อ Project ของเราและเลือกรูปแบบ Hardware ที่เราจะใช้ได้ Hardware ให้เลือก ESP32 Dev Board



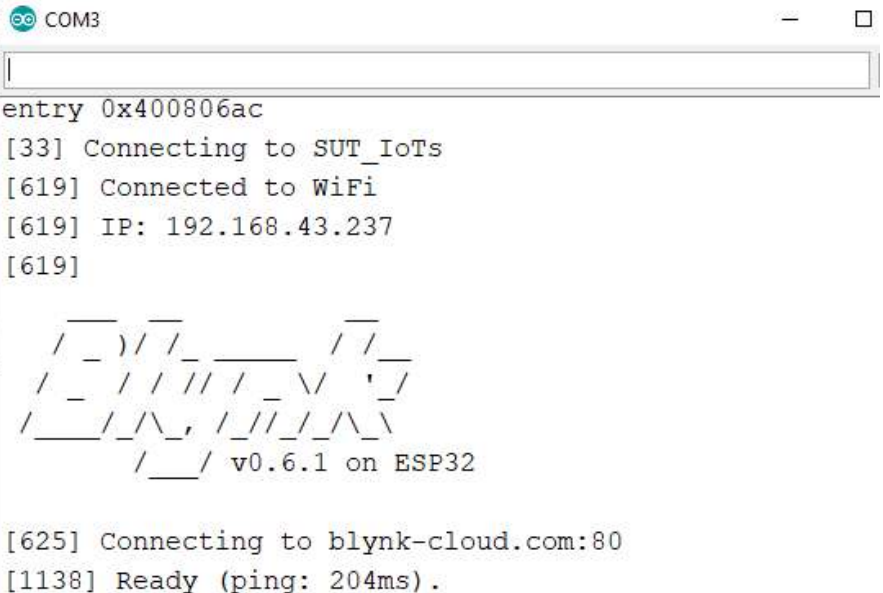
8. โดยทุกๆครั้งที่เริ่มสร้างโปรเจคใหม่ AUTH TOKEN จะถูกเปลี่ยนใหม่เสมอ ซึ่ง KEY นี้เองที่เป็นเสมือนกุญแจสำหรับเชื่อมต่อ โดยที่เราไม่ต้องใช้ user, password เราสามารถกดที่คำว่า” E-mail” เพื่อส่ง KEY นี้เข้าเมลเราได้



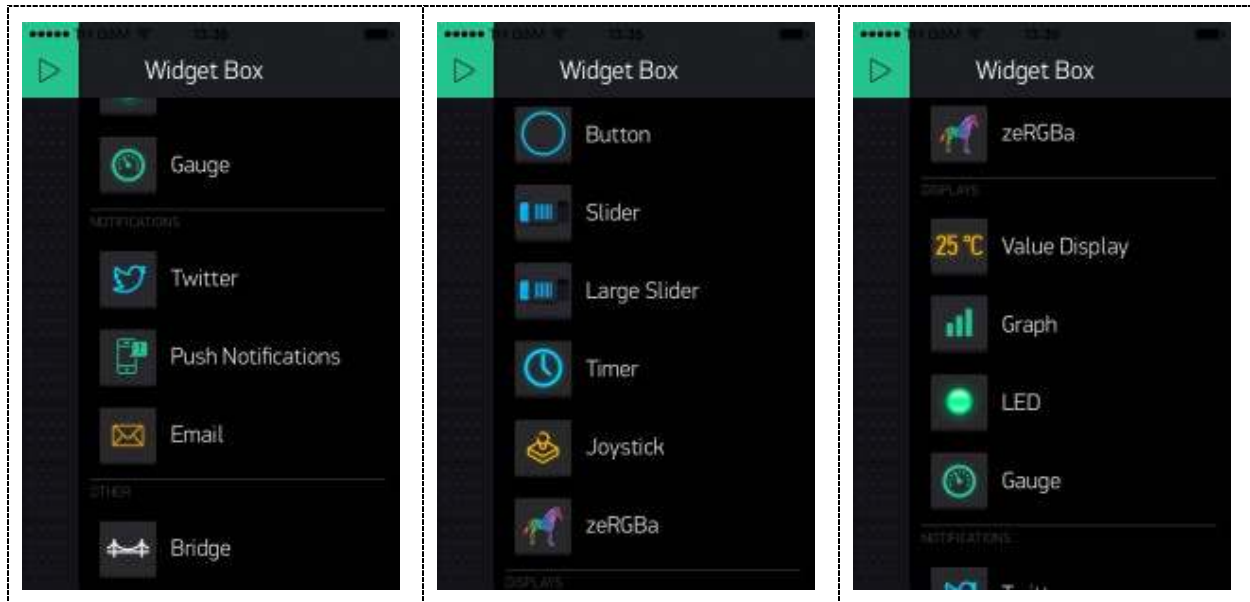
9. การใช้งานบน Arduino IDE ด้วยโปรแกรม < อย่าลืมแก้ไข Key, SSID, Password >

<pre> ESP32_WIFI\$ 1 2 #define BLYNK_PRINT Serial 3 4 #include <WiFi.h> 5 #include <WiFiClient.h> 6 #include <BlynkSimpleEsp32.h> 7 8 char auth[] = "YourAuthToken"; 9 char ssid[] = "YourNetworkName"; 10 char pass[] = "YourPassword"; 11 12 void setup() 13 { Serial.begin(9600); 14 Blynk.begin(auth, ssid, pass); 15 } 16 17 void loop() 18 { Blynk.run(); 19 } 20 </pre>	<pre> #define BLYNK_PRINT Serial #include <WiFi.h> #include <WiFiClient.h> #include <BlynkSimpleEsp32.h> char auth[] = "YourAuthToken"; char ssid[] = "YourNetworkName"; char pass[] = "YourPassword"; void setup() { Serial.begin(115200); Blynk.begin(auth, ssid, pass); } void loop() { Blynk.run(); } </pre>
--	--

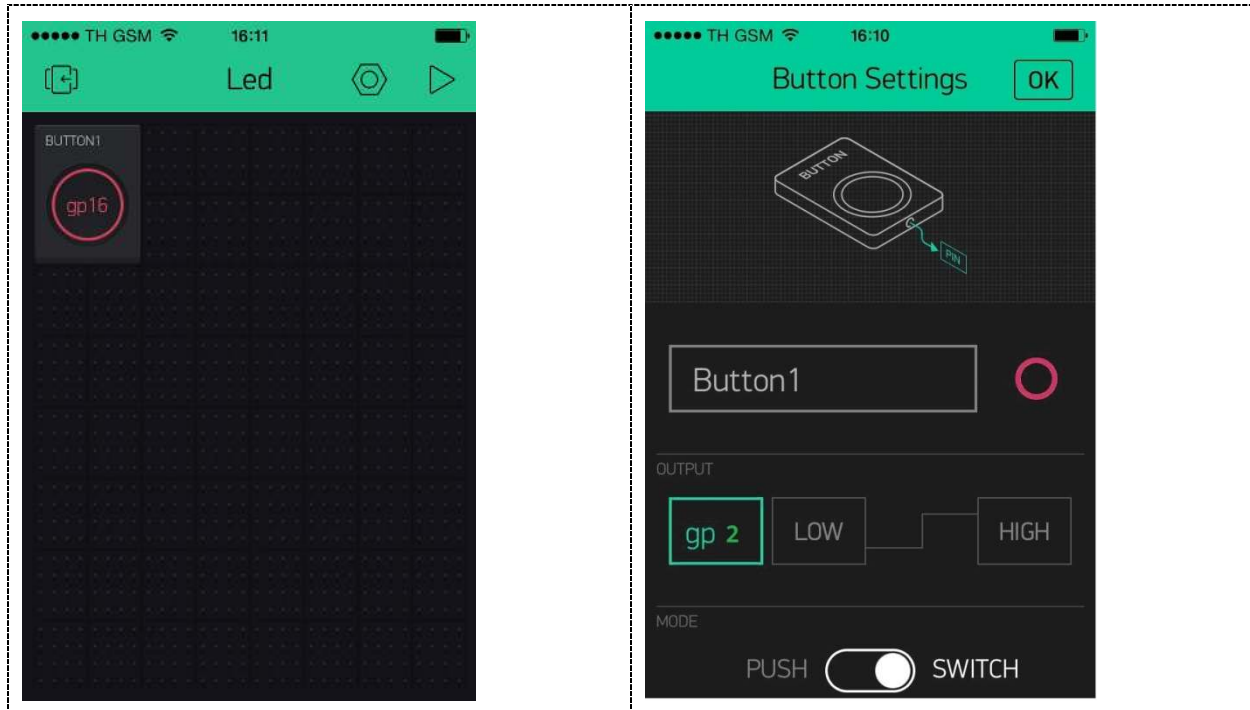
10. จากนั้นให้ทำการ Upload ลง board แล้วเปิด Serial Monitor จนกระทั่งมีข้อความขึ้นแบบนี้

<pre> 4 #include <WiFi.h> 5 #include <WiFiClient.h> 6 #include <BlynkSimpleEsp32.h> 7 8 char auth[] = "YourAuthToken"; 9 char ssid[] = "YourNetworkName"; 10 char pass[] = "YourPassword"; 11 12 void setup() 13 { Serial.begin(9600); 14 Blynk.begin(auth, ssid, pass); 15 } 16 17 void loop() 18 { Blynk.run(); 19 } 20 </pre>	 <pre> entry 0x400806ac [33] Connecting to SUT_IoTs [619] Connected to WiFi [619] IP: 192.168.43.237 [619] v0.6.1 on ESP32 [625] Connecting to blynk-cloud.com:80 [1138] Ready (ping: 204ms) </pre>
--	--

11. สร้างโปรเจกของเรบบน Blynk โดยการกดที่พื้นที่ว่างเปล่าตรงไหนก็ได้ในหน้า New project จะปรากฏหน้าต่างของ Widget ให้เราเลือกขึ้นมา

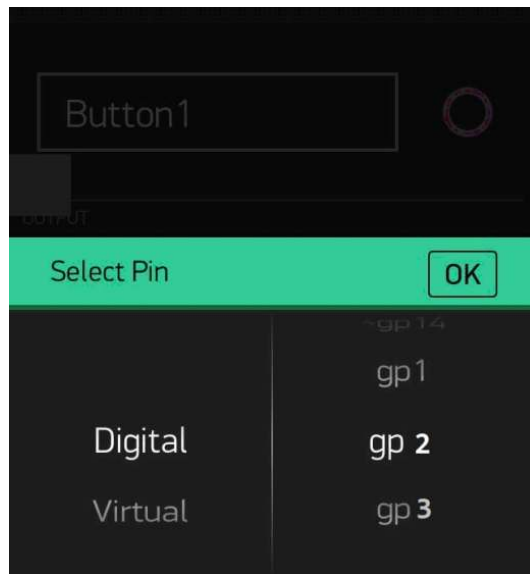


12. ในบทความรอบนี้เราจะลองให้ดูอะไรที่ง่ายๆ ก่อน ให้ลองเลือก Button widget มาลงบนพื้นที่ว่างเปล่ามา 1 อัน จากนั้นเราจะมาตั้งค่าการใช้งานปุ่ม Button กันโดยกดไปที่รูป Button ที่เราเลือกจะปรากฏหน้าต่างแบบนี้ ซึ่งในหน้านี้เราสามารถเปลี่ยนชื่อปุ่มได้ และเลือกโหมด output pin ที่ต่อกับอุปกรณ์จาก board ของเราได้



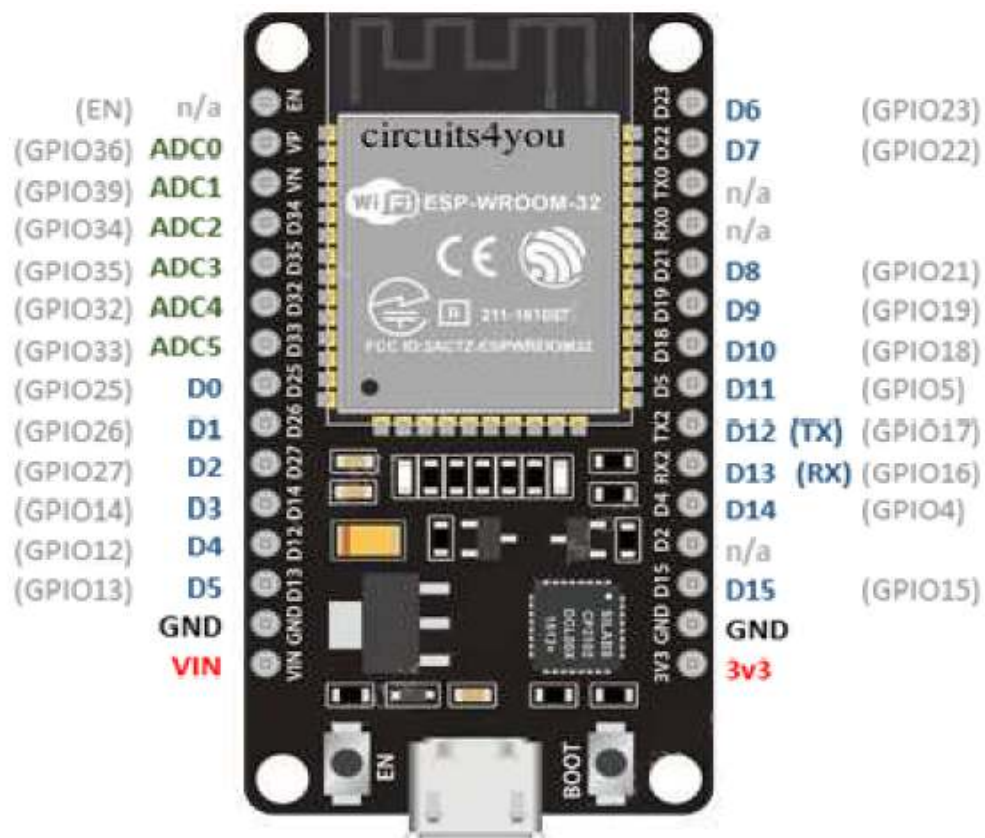
13. เลือกรูปแบบ pin จะให้เป็นขา Digital หรือ Virtual ก็ได้ ซึ่งรูปแบบ Virtual จะไม่ใช้การรับค่าจากขาจริงๆ เป็นเหมือนการสร้างตัวแปรมาเก็บค่าอีกที และเลือกขา GPIO ให้ตรงกับ อุปกรณ์ที่เราจะต่อ เมื่อเสร็จแล้วกด ▷ เพื่อให้ App เริ่มทำงาน

14. ทดสอบการทำงานโดยการคุม LED GPIO2 หรือ On Board [DOIT ESP32 Kit Ver1](#)



- Button1
- SWITCH
- Select Pin
 - # Virtual → Variable
 - # Digital → Real Port
 - # gp2 < ESP32 On Board LED >
- กด ▷ เพื่อให้ App เริ่มทำงาน

ESP32 PINOUT



Lab301B – ESP-32 Control and Monitor by Blynk

15. การทดสอบอ่านสวิตช์ DO แล้วแสดงผลที่ LED Port V5 บน Blynk

- จาก Web <http://help.blynk.cc/getting-started-library-auth-token-code-examples/blynk-basics/how-to-display-any-sensor-data-in-blynk-app>
- จาก Web <https://community.blynk.cc/t/how-to-turn-on-widget-leds/643>
- ทดสอบโปรแกรม ให้แก้ไข (1/3)Auth, (2/3)SSID และ (3/3>Password

```
// ESP 32

#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

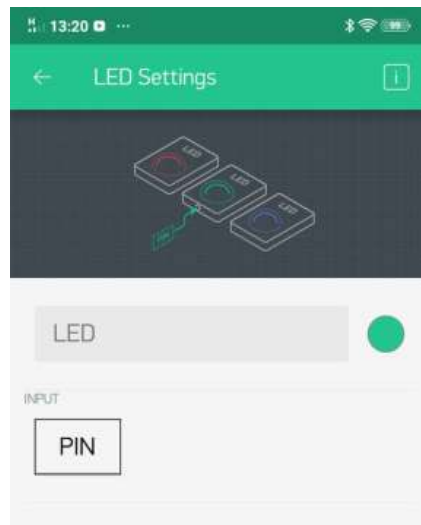
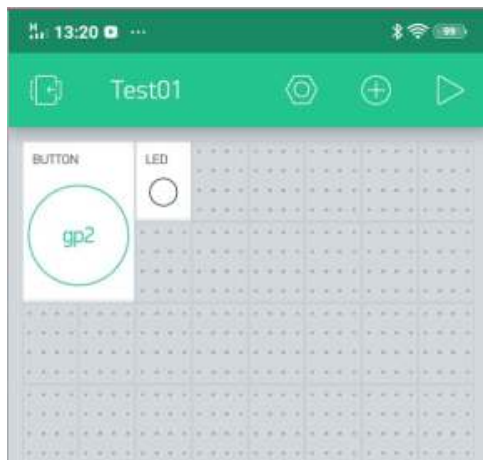
const int btnPin = 0; // D0
boolean btnState = false;
WidgetLED blynk_LED(V5);
BlynkTimer timer; // Announcing the timer

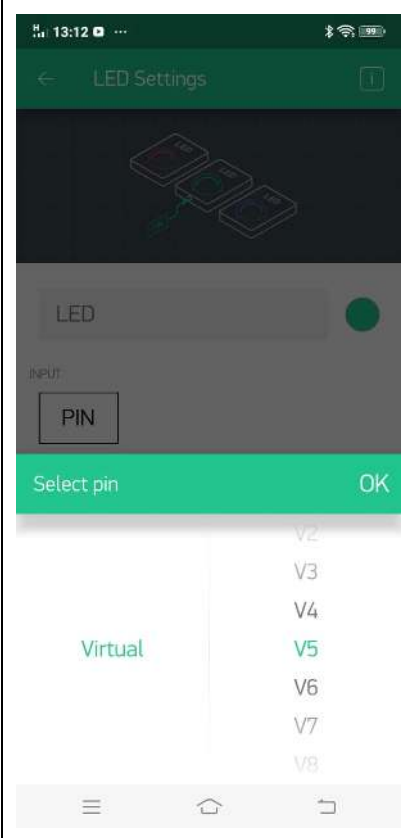
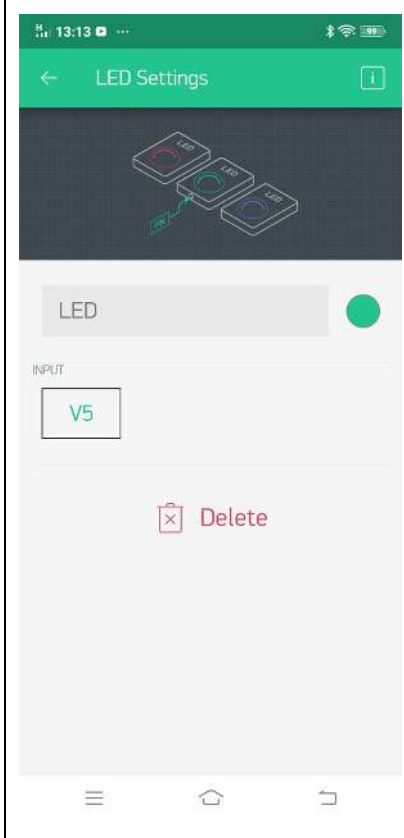
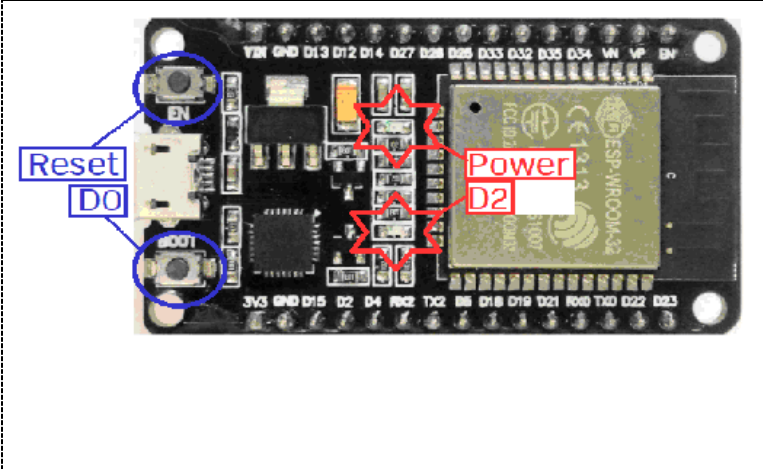
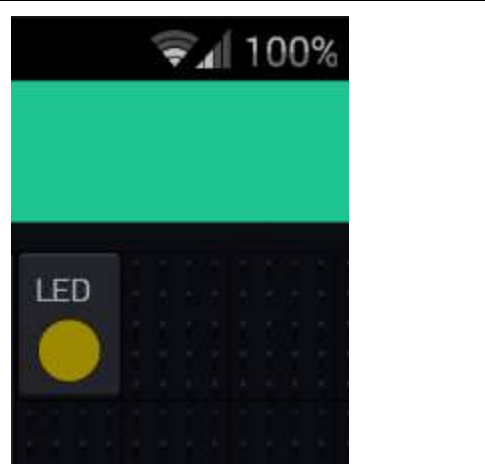
char auth[] = "f311e6ae4aa94210f29b53061b305b";
char ssid[] = "SUT_IoTs";
char pass[] = "MaiMeeJingJing";

void myTimerEvent()
{
  boolean isPressed = (digitalRead(btnPin) == LOW);
  if (isPressed != btnState)
  {
    if (isPressed)
      blynk_LED.on();
    else
      blynk_LED.off();
    btnState = isPressed;
    Serial.print(" LED Status = ");
    Serial.println(btnState);
  }
}

void setup()
{
  Serial.begin(115200);
  pinMode(btnPin, INPUT_PULLUP);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(250L, myTimerEvent);
}

void loop()
{
  Blynk.run();
  timer.run(); // running timer every 250ms
}
```



		<pre> /___/ v0.6.1 on ESP32 [4678] Connecting to blynk-cl [4892] Ready (ping: 177ms). LED Status = 1 LED Status = 0 LED Status = 1 LED Status = 0 [37614] Heartbeat timeout [39617] Connecting to blynk-cl [42631] Login timeout [44631] Connecting to blynk-cl [46263] Ready (ping: 1621ms).</pre>
		

Lab301c – Blynk LED Control and Sensor Monitor

17. การทดสอบอ่านอุณหภูมิด้วย DHT-22 แล้วแสดงผลที่ Blynk

- จาก Web <http://help.blynk.cc/getting-started-library-auth-token-code-examples/blynk-basics/how-to-display-any-sensor-data-in-blynk-app>
- Install **DHT22 Library** เลือก **DHT Sensor library for ESPx V1.17.0** แล้วทำการติดตั้ง

DHT sensor library for ESPx

by beegee_tokyo Version 1.17.0 **INSTALLED**

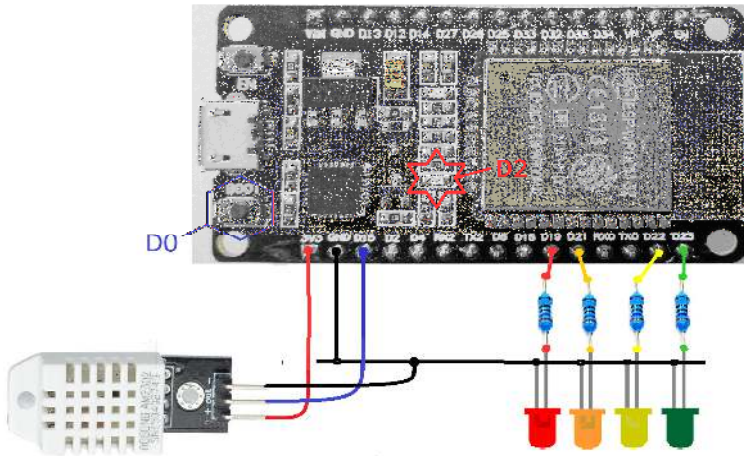
Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors Optimized library to match ESP32 requirements. Last changes: Reduce CPU usage and add decimal part for DHT11

[More info](#)

Select version ▾

Install

18. ต่อวงจร DHT-22 เข้าที่ขา D15 และทดสอบการทำงานของโปรแกรม



```
#include "DHTesp.h"
DHTesp dht;
const int pinDHT_22 = 15;

void setup()
{ Serial.begin(115200);
  dht.setup(pinDHT_22, DHTesp::DHT22);
}

void loop()
{ float temperature = dht.getTemperature();
  float humidity = dht.getHumidity();
  Serial.print(" Temp('C) >> "); Serial.print(temperature, 1);
  Serial.print(", Humidity(%) >> "); Serial.println(humidity, 1);
  delay(2000);
}
```

19. ทดสอบการทำงานของโปรแกรม ให้แก้ไข (1/3)Auth, (2/3)SSID และ (3/3>Password

```
// ESP-32

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "DHTesp.h"

BlynkTimer timer; // Announcing the timer
DHTesp dht;

char auth[] = "f311e6ae4aa9421cb0f29b53061b3b";
char ssid[] = "SUT_IoTs";
char pass[] = "MaiMeeJingJing";

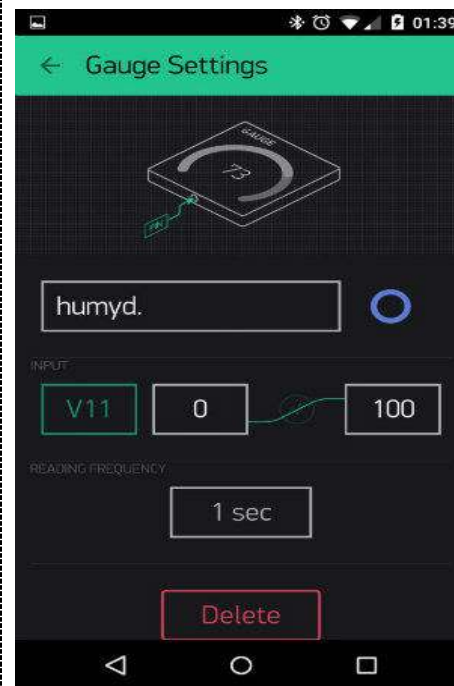
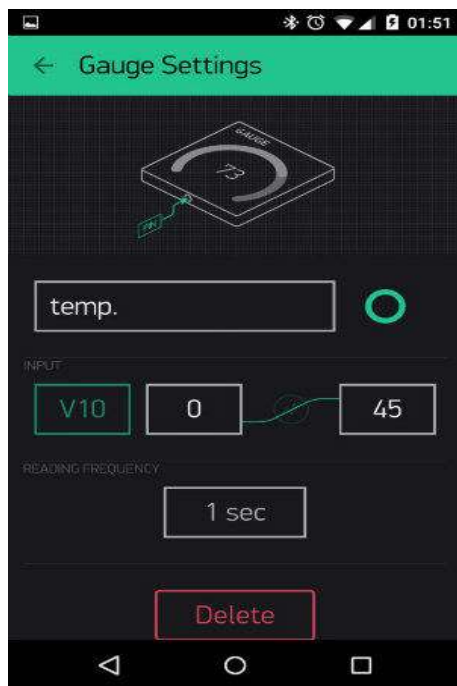
const int pinDHT_22 = 15; // D15
float temperature = 12.34, humidity = 56.78;

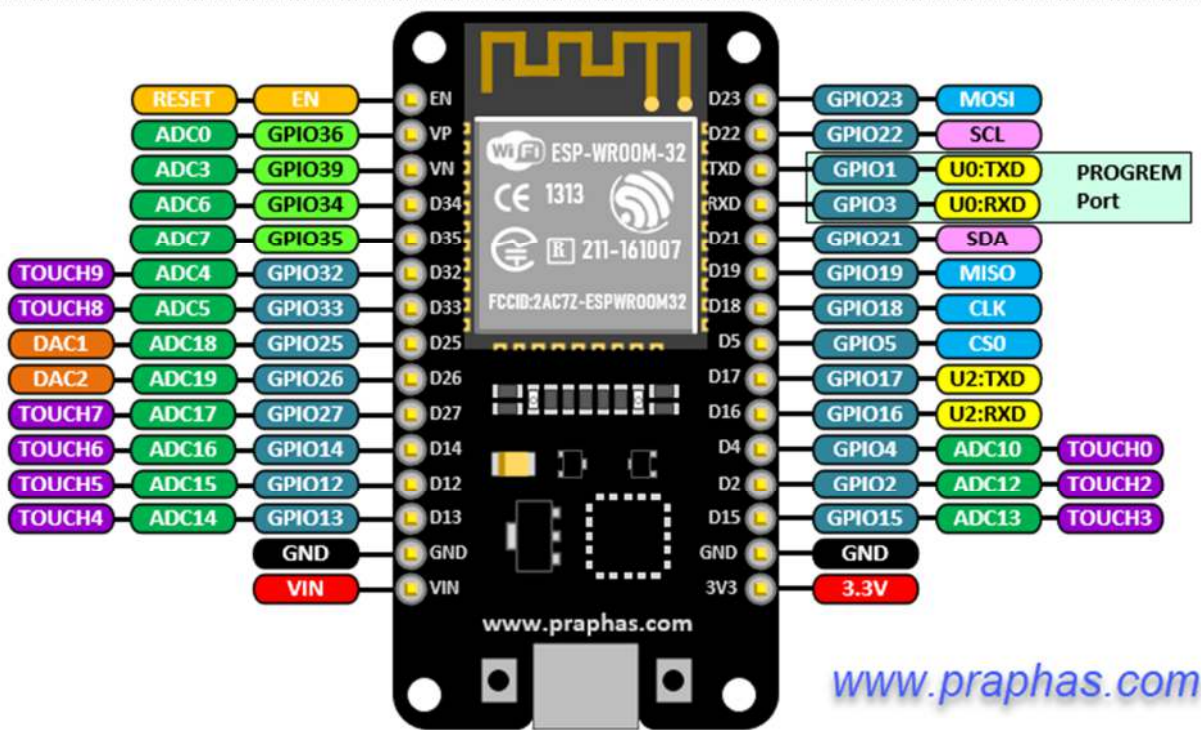
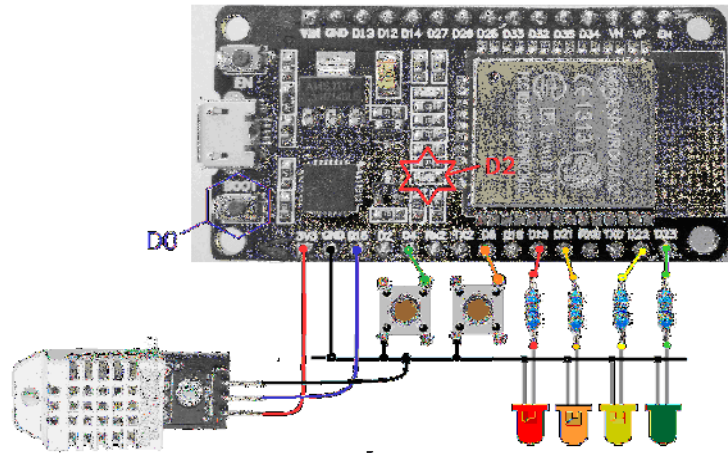
void myTimerEvent()
{
  temperature = dht.getTemperature();
  humidity = dht.getHumidity();
  Blynk.virtualWrite(V10, temperature);
  Blynk.virtualWrite(V11, humidity);
  Serial.print(" Temp('C) >> ");      Serial.print(temperature, 1);
  Serial.print(", Humidity(%) >> ");  Serial.println(humidity, 1);
}

void setup()
{
  Serial.begin(115200);
  dht.setup(pinDHT_22, DHTesp::DHT22);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(1000L, myTimerEvent);
}

void loop()
{
  Blynk.run();
  timer.run(); // running timer every 250ms
}
```

20. ที่ Blynk ให้ใช้ Gauge และ Port V10 และ Port V11





2/6 - การใช้งาน Raspberry Pi เพื่อแสดงค่าและควบคุมผ่าน Blynk

<http://nrc-intelligentsystems.com/nd/node-red-คืออะไร/>

Lab302 – Raspberry Pi to Blynk with Node-RED

1. ตรวจสอบและ Upgrade npm

```
wget https://www.npmjs.com/install.sh && sudo sh ./install.sh
```

```
npm -v
```

```
nodejs -v
```

```
found 0 vulnerabilities
successfully installed npm@latest
pi@raspberrypi:~ $ npm -v
7.20.6
pi@raspberrypi:~ $ nodejs -v
v10.24.0
```

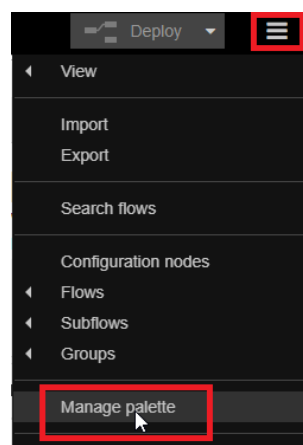
```
pi@raspberrypi:~ $ npm -v
```

```
7.20.5
```

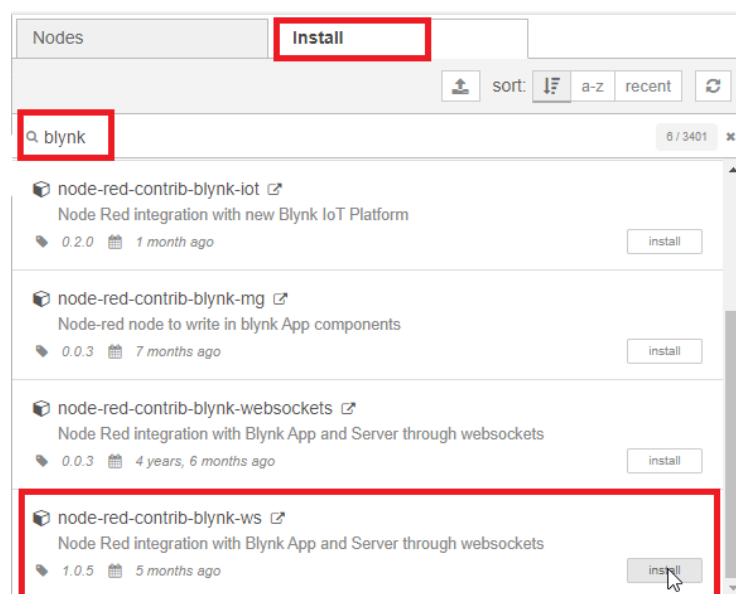
```
pi@raspberrypi:~ $ nodejs -v
```

```
v10.24.0
```

2. เพิ่มโหนด Blynk ให้กับ Node-RED



Manage Palette

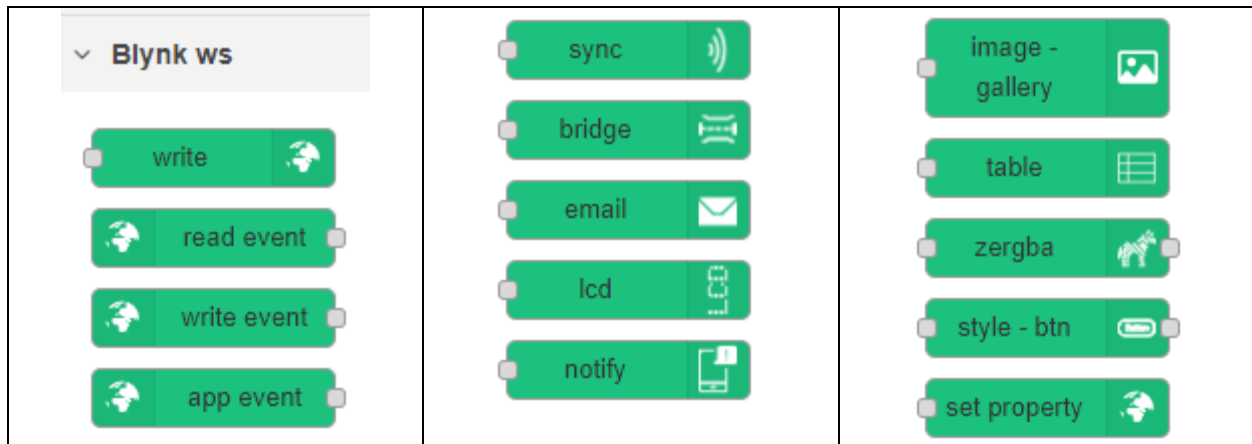


Install

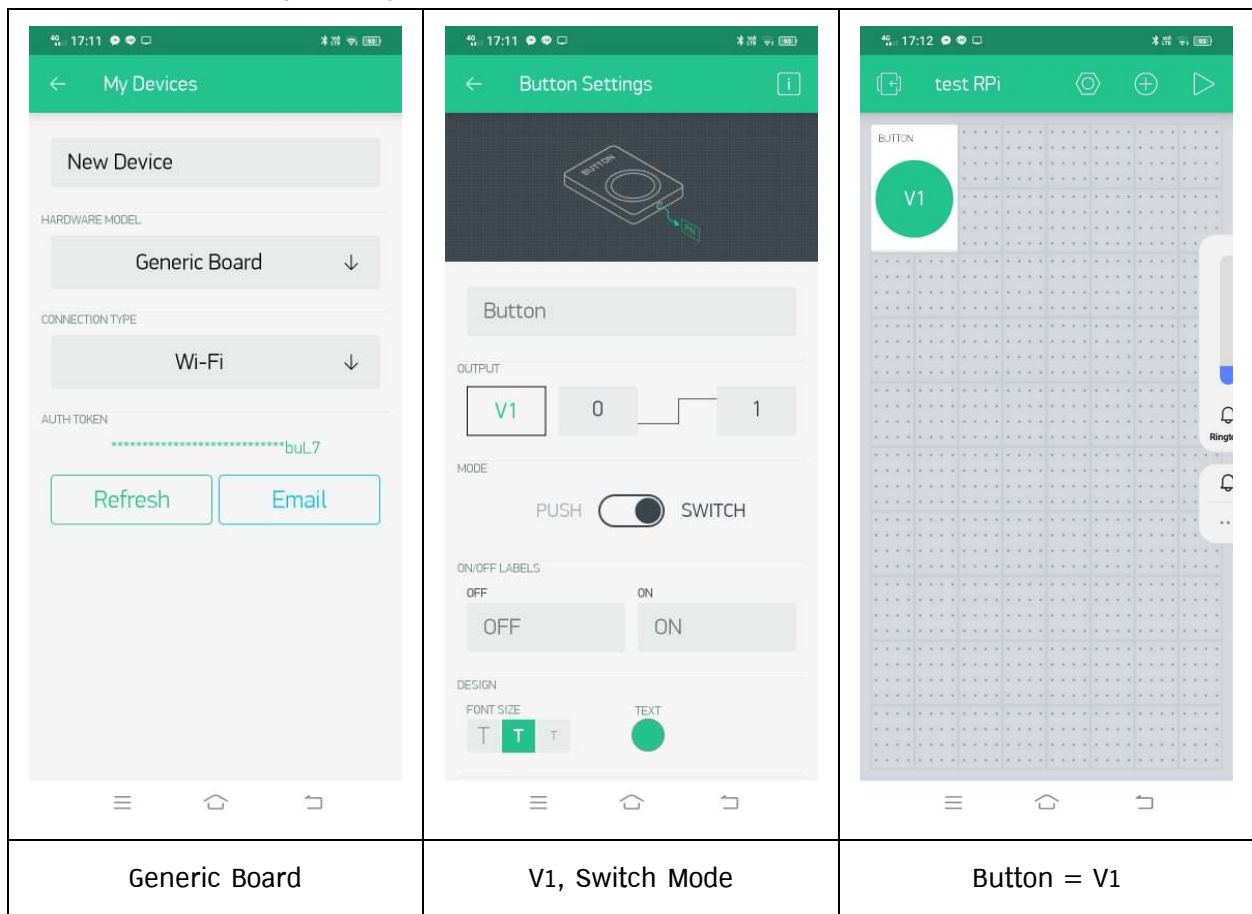
blynk

node-red-contrib-blynk-ws
install
confirm


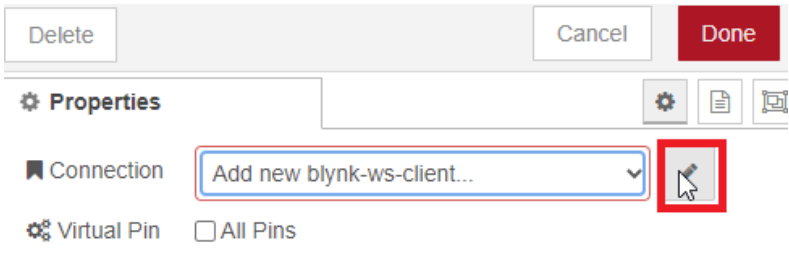
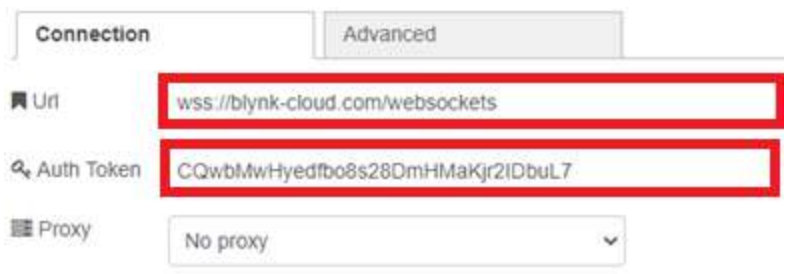

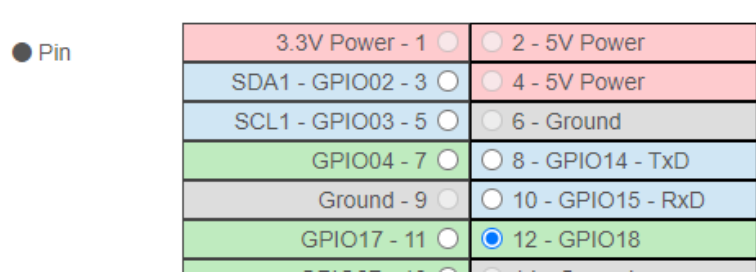
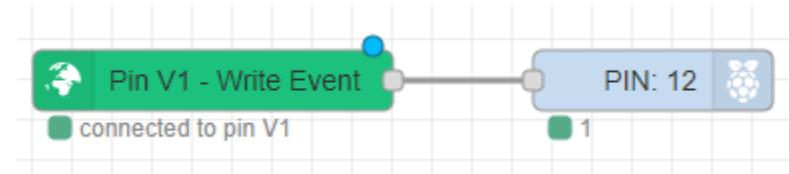
3. จะได้โหนด Blynk WebSocket เพิ่มเข้ามา



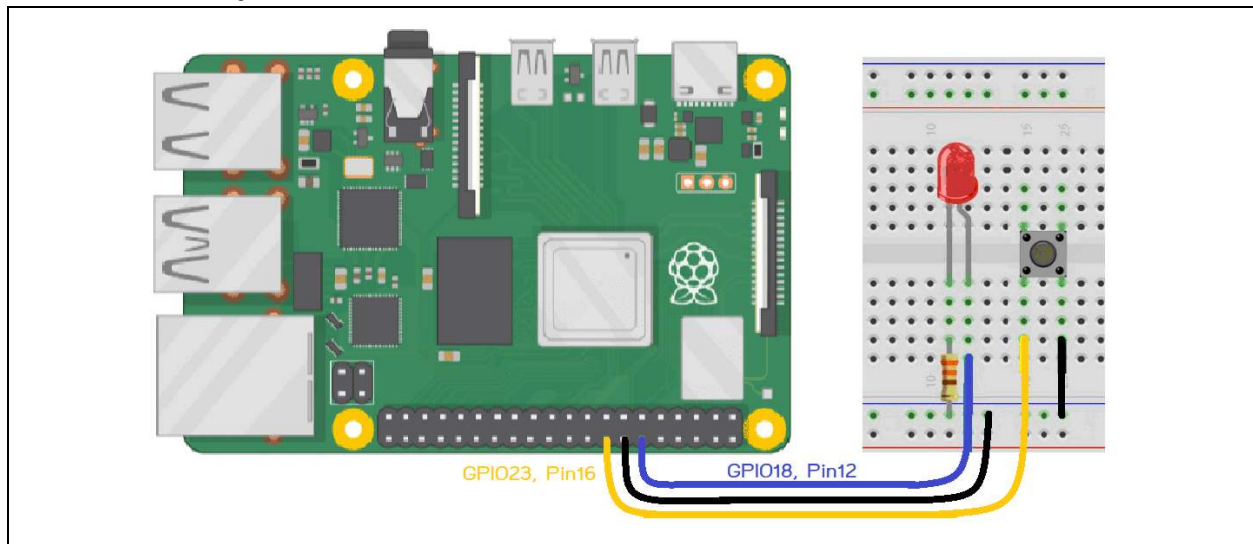
4. Create New Project ที่ Blynk



5. สร้าง Node Flow และ Configure

	Write event Gpio-out
	Add new blynk
	wss://blynk- cloud.com/websockets Token Key
	Set virtual pin
	Pin12 - GPIO18
	Deploy

6. ทดสอบตามรูป LED at Board.Pin12

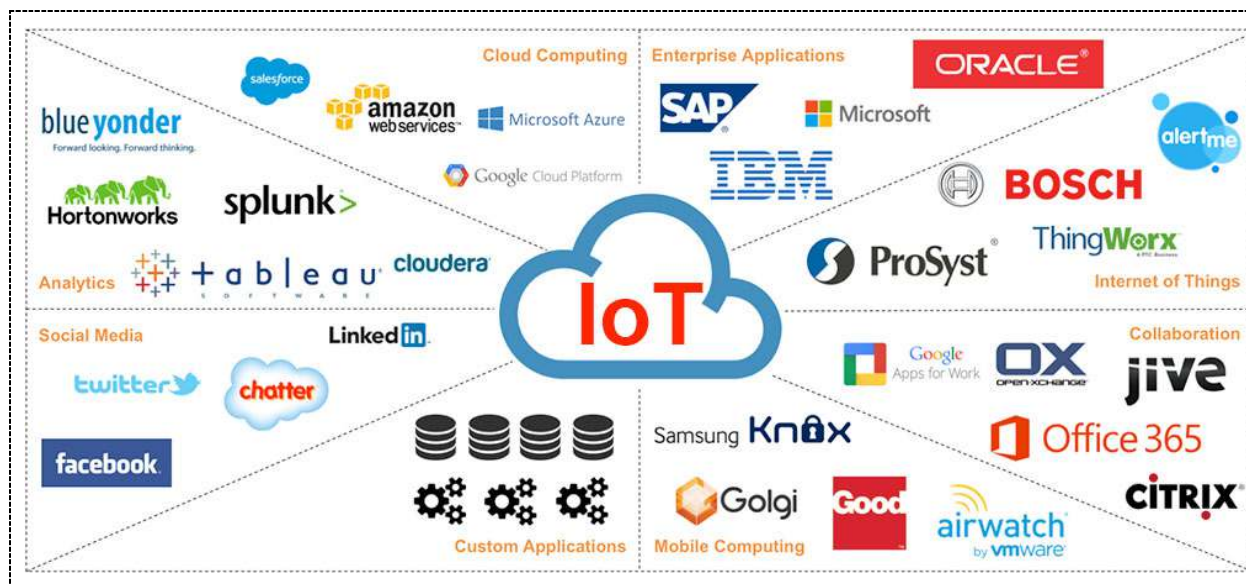


7. เพิ่มเต็มNode Flow และ Configure

	<p>X1rpi-gpio in node change node Write node Write node</p>
	<p>Pin Board = 16</p> <p>Pull up</p>
	<p>Change msg.payload for number = 0 replace number = 0</p> <p>Add msg.payload for number = 1 replace number = 255</p>

3/6 - การใช้งาน ESP32 เพื่อแสดงค่าและควบคุมผ่าน Public MQTT Server

<https://mntolia.com/10-free-public-private-mqtt-brokers-for-testing-prototyping/>
<https://www.hivemq.com/blog/mqtt-toolbox-mqtt-lens/>

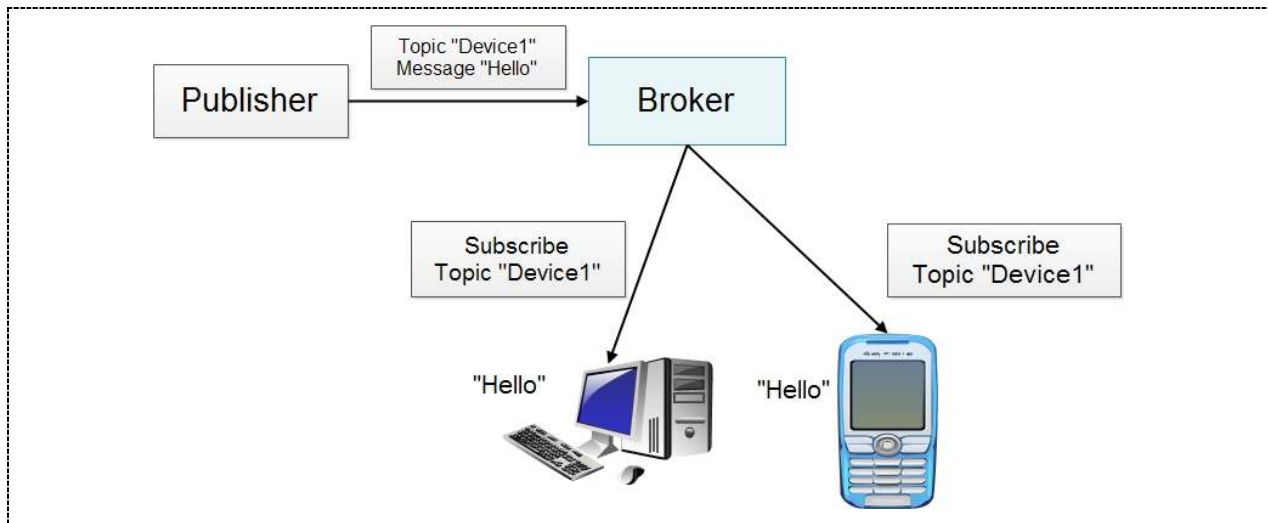


3.1 IoT Concept

ปัจจุบันเทคโนโลยีที่กำลังมาแรงสำหรับนักพัฒนาด้าน Embedded (ไม่ได้แค่เฉพาะ Embedded อย่างเดียวครับ) เป็นเทคโนโลยีที่กำลังมาแรงมากจนจะหนีไม่พ้น IoT ซึ่งเป็นเทคโนโลยีใหม่ในยุคนี้เลยก็ว่าได้ แรงขนาดที่ Microsoft เอง ก็ยังพอร์ต Windows 10 มาวิ่งเล่นบน Raspberry Pi แถมยังเจตนา IoT มาให้ด้วย ซึ่งผมยังไม่ได้ตามลงไปดูว่าใช้ Broker ตัวไหน และมี Library ให้ใช้งานมาด้วยหรือไม่? หรือไม่ผมก็เข้าใจผิดเกี่ยวกับมันครับถ้าผิดพลาดก็ขออภัยมานะที่นี้ด้วยครับ.

IoT มันคืออะไร พอค้นดูมีหลายสิ่งอธิบายไว้มากมาย เช่น [Internet of Things เมื่อคอมพิวเตอร์เริ่มคุยกันเองได้](#) , [โลกแห่ง IoT มาถึงแล้ว](#) , [IoT เทคโนโลยีที่ธุรกิจต้องรู้](#). ลองนึกภาพดูครับว่าถ้าหากอุปกรณ์สามารถส่งงานไปมาหากันได้ผ่าน www ไม่ว่าจะเป็น PC, Smart Phone หรือแม้แต่อุปกรณ์ขนาดเล็กพวก Micro-Controller, PLC, HUB, Switch หรืออะไรก็แล้วแต่ที่มันสามารถต่อระบบ Network ได้ว่ามันจะอยู่ที่บ้าน ที่โรงงาน ไร่ นา ฟาร์ม โรงเรือน โรงงานอุตสาหกรรมหรือที่อื่นๆที่มีระบบเน็ตเวิร์กที่เข้าถึง www ได้เราจะสามารถควบคุมมันได้ทั้งหมดที่ไหนก็ได้ในโลกใบนี้

องค์ประกอบหลักของ IoT จะมี 3 ส่วนคือ Broker, Publisher และ Subscriber. ซึ่งการรับและส่งข้อมูลนั้น มันจะส่งข้อมูลไปมาหากันนั้นจะส่งผ่านตัวกลางนั้นก็คือ Broker Server โดยตัวส่งนี้จะเรียกว่า Publisher ส่งข้อมูลขึ้นไปยัง Broker พร้อมระบุหัวข้อ (Topic) ที่ต้องการส่งข้อมูลออกไป จากนั้นตัวรับซึ่งเรียกว่า Subscriber ถ้าหากตัวรับต้องการรับข้อมูลจากตัวส่งจะต้องทำการ Subscribe หัวข้อ Topic ของ Publisher นั้นๆ ผ่าน Broker เช่นกัน



จากรูปภาพด้านบนจะมีตัว Publisher ทำการ Publish ข้อความ “Hello” ใน Topic Device1 เมื่อและถ้าหากมีคอมพิวเตอร์ หรืออุปกรณ์อื่นๆทำการ Subscribe หัวข้อ Topic Device1 เมื่อ Publisher ทำการส่งข้อมูลไปยัง Topic อุปกรณ์ Subscribe จะได้ข้อความ “Hello” เช่นเดียวกัน. ก็คล้ายๆกับที่ใช้งานไลน์ที่คุยกันเป็นกลุ่มนั่นแหละครับ. ซึ่งจะเห็นข้อความ “Hello” ในเวลาเดียวกันนั้นหมายความว่าอุปกรณ์ใดๆที่ทำการ Subscribe Topic เดียวกันก็จะได้รับข้อความเดียวกันครับ

3.2 MQTT-Message Queue Telemetry Transport

โปรโตคอลที่ใช้สำหรับรับและส่งข้อมูลนั้นคือ MQTT ปัจจุบันถึง Version 3.1 ในที่นี้จะมาทำการทดลองส่งข้อมูลกันตัว Server จะมีอยู่ด้วยกันหลายค่ายครับสำหรับที่ลิสมาด้านล่างนี้ครับ

Open Source MQTT Broker Server

- Mosquitto
- RSMB
- ActiveMQ
- Apollo
- Moquette
- Mosca
- RabbitMQ

Client

- Paho
- Xenqtt
- mqtt.js
- node_mqtt_client
- Ascoltatori
- Arduino MQTT Client

สำหรับ MQTT Broker Server ฟรีที่ผมพอค้นได้ก็มีดังนี้ครับ

- test.mosquitto.org
- mqtt.eclipse.org
- broker.mqttdashboard.com

3.3 IoT และ MQTT คืออะไร?

MQTT ย่อมาจาก Message Queue Telemetry Transport เป็นโปรโตคอลประยุกต์ที่ใช้โปรโตคอล TCP เป็นรากฐาน ออกแบบมาสำหรับงานที่ต้องการ ๑ สื่อสารแบบเรียลไทม์แบบไม่จำกัดแพลตฟอร์ม หมายถึงอุปกรณ์ทุกชิ้นสามารถสื่อสารกันผ่าน MQTT

MQTT จะแบ่งเป็น 2 ฝ่าย คือฝั่งเซิร์ฟเวอร์มักจะเรียกว่า MQTT Broker ส่วนฝั่งผู้ใช้งานจะเรียกว่า MQTT Client ในการใช้งานด้าน IoT จะเกี่ยวข้องกับ MQTT Client เป็นหลัก โดยจะมี MQTT Broker ทั้งแบบฟรี และเสียเงินไว้รองรับอยู่แล้ว ทำให้การสื่อสารข้อมูลผ่าน MQTT จะใช้เซิร์ฟเวอร์ฟรี หรือ MQTT Broker ฟรี เหล่านั้นเป็นตัวกลาง

ลักษณะการใช้งาน MQTT อาจจะเปรียบเสมือนได้กับการใช้งานห้องแชท Line สำหรับอุปกรณ์ โดยอุปกรณ์แต่ละตัวจะมีชื่อเป็นของตนเอง มี Username Password เป็นของตัวเอง และอาจจะมีห้องลับเฉพาะของตนเอง ดังนั้นการใช้งาน MQTT ผู้เขียนจึงจะขอยกตัวอย่างของ MQTT เทียบกับห้องแชทได้ดังนี้

กลุ่มผู้ใช้ (User)

ใน MQTT จะแบ่งกลุ่มของผู้ใช้งานออกเป็น 2 ระดับ คือ

- ระดับสูงสุด – สามารถที่จะรับ-ส่งข้อมูลกับอุปกรณ์ หรือช่องทางใด ๆ ก็ได้ในระบบ หรือเปรียบได้กับแอดมินที่สามารถเข้าไปดูข้อความได้ทุกห้องแม้จะเป็นห้องลับก็ตาม
- ระดับทั่วไป – สามารถรับ-ส่งข้อมูลกับอุปกรณ์หรือช่องทางที่กำหนดไว้เฉพาะเท่านั้น เปรียบได้กับผู้ใช้งาน Line ที่สามารถแชทในห้องที่ตัวเองสร้างได้ หรือเป็นสมาชิกในห้อง แต่ไม่สามารถเข้าไปแชทในห้องที่ไม่ได้เป็นสมาชิก

ในการใช้งานจริง ในอุปกรณ์ต่าง ๆ ควรจะใช้งานในระดับทั่วไป เพื่อความปลอดภัยกรณีอุปกรณ์เหล่านั้นถูกแฮกแล้วไม่สร้างความเสียหายไปยังอุปกรณ์อื่น ๆ ที่อยู่ในช่องทางเฉพาะของแต่ละอุปกรณ์

เส้นทาง (Topic)

เส้นทาง เปรียบเหมือนกับหัวข้อ หรือห้องแชทที่ต้องการจะคุย และการคุยกันจะมีเฉพาะอุปกรณ์ที่อยู่ในห้องนั้น ๆ (Subscribe) ถึงจะสามารถได้รับข้อมูลที่มีการส่งไปในห้องนั้น ๆ ที่ถูกเรียกว่าเส้นทางเนื่องจากการใช้งานส่งข้อมูลและรับข้อมูลจะเหมือนกับเส้นทางในระบบไฟล์ เช่น /Room1/LED ซึ่งระบบเส้นทางนี้นอกจากอุปกรณ์จะสามารถรอการสนทนาในห้องตามเส้นทาง /Room1/LED ได้แล้ว ยังสามารถรอสนทนาเส้นทาง /Room1 ได้ด้วย หากเป็นการรอฟังในเส้นทาง(Subscribe) /Room1 จะหมายถึงการส่งข้อมูลใด ๆ ที่นำหน้าด้วย /Room1 เช่น /Room1/LED , /Room1/Value ผู้ที่รอฟัง (Subscribe) /Room1 อยู่จะได้รับข้อมูลเหล่านั้นด้วย

คุณภาพข้อมูล (QoS)

แบ่งออกเป็น 3 ระดับดังนี้

- QoS0 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่
- QoS1 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่ แต่ให้จำค่าที่ส่งล่าสุดไว้ เมื่อมีการเชื่อมต่อใหม่จะได้รับข้อมูลครั้งล่าสุดอีกครั้ง
- QoS2 – ส่งข้อมูลหลาย ๆ ครั้งจนกว่าปลายทางจะได้รับข้อมูล มีข้อเสียที่สามารถทำงานได้ช้ากว่า QoS0 และ QoS1

การส่งข้อมูล (Publish)

การส่งข้อมูลในแต่ละครั้งจะต้องประกอบไปด้วยเส้นทางข้อมูล และคุณภาพข้อมูล ซึ่งการส่งข้อมูลจะเรียกว่า Publish

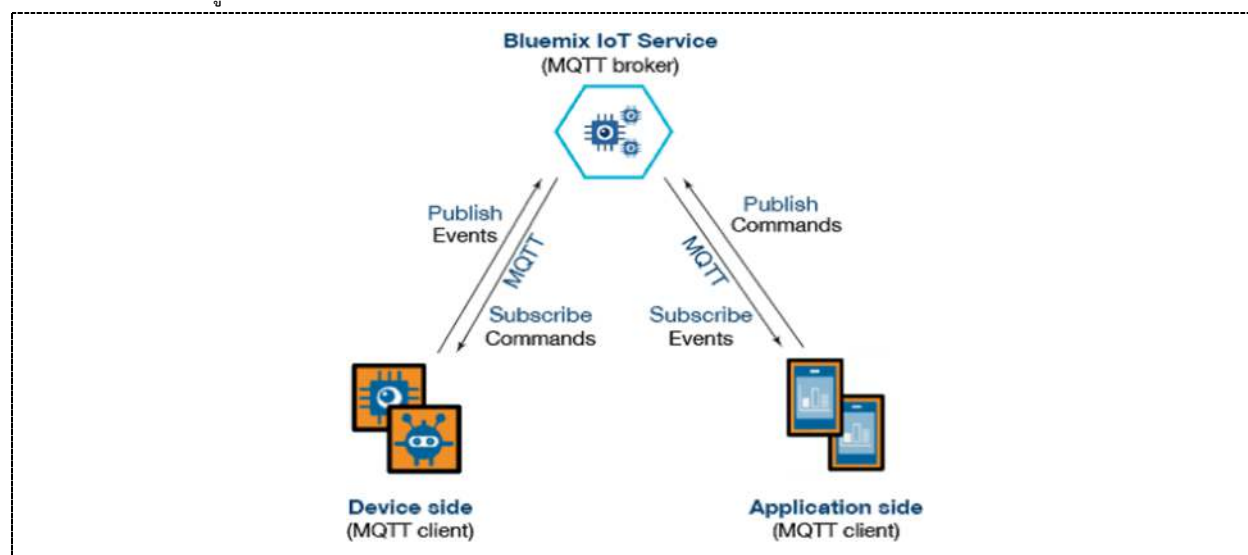
การรับข้อมูล (Subscribe)

การรับข้อมูลในระบบ MQTT จะรับข้อมูลได้เฉพาะเมื่อมีการเรียกใช้การ Subscribe ไปยัง Topic ที่กำหนด อาจเปรียบได้กับการ Subscribe คือการเข้าไปนั่งรอเพื่อนในกลุ่ม Line ส่งแชทมาหา เมื่อมีการส่งข้อมูลเข้ามาจะเกิดสิ่งที่เรียกว่าเหตุการณ์ (Event) ให้เรากดเข้าไปดูข้อความที่เพื่อน ๆ ส่งเข้ามา

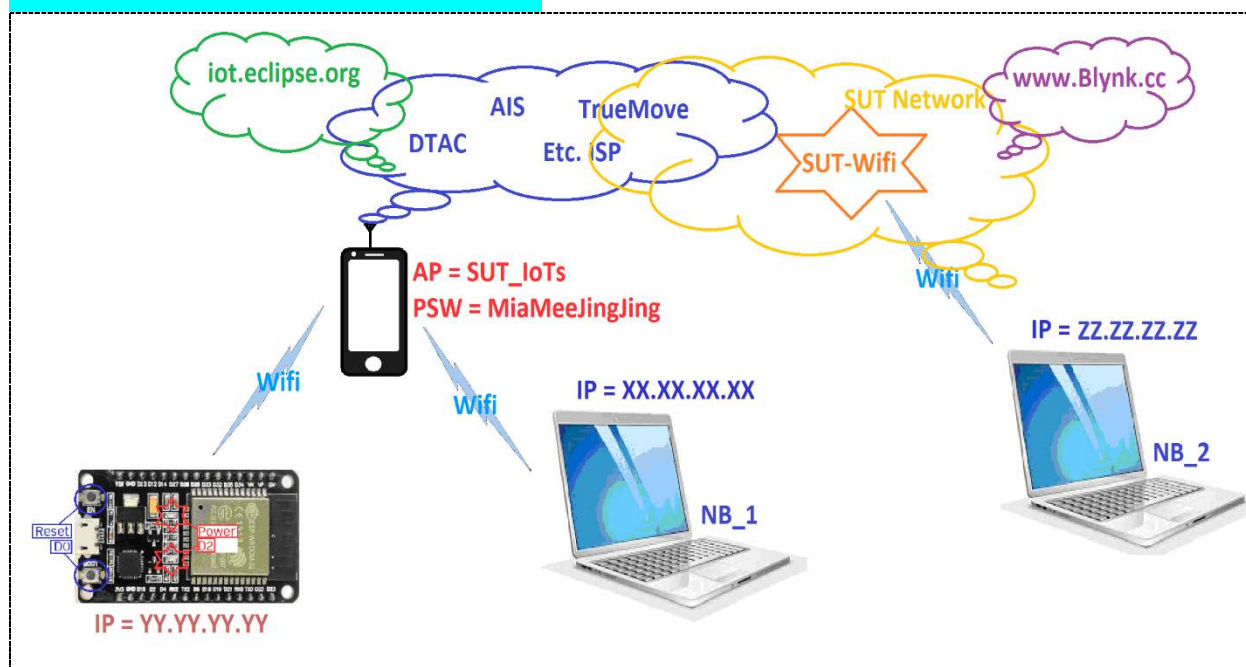
จะเห็นว่า MQTT ก็เปลี่ยนเสมือนห้องแชทของอุปกรณ์ที่จะสนทนาแลกเปลี่ยนข้อมูลกันแบบเรียลไทม์ผ่านเครือข่ายอินเทอร์เน็ต

3.4 IoT มีวิธีการทำงานอย่างไร?

องค์ประกอบหลักของ IoT จะมี 3 ส่วนคือ **Broker**, **Publisher** และ **Subscriber**. ซึ่งการรับและส่งข้อมูลนั้น มันจะส่งข้อมูลไปมาหากันนั้นจะส่งผ่านตัวกลางนั้นก็คือ Broker Server โดยตัวส่งนี้จะเรียกว่า Publisher ส่งข้อมูลขึ้นไปยัง Broker พร้อมระบุหัวข้อ (Topic) ที่ต้องการส่งข้อมูลออกไป จากนั้นตัวรับซึ่งเรียกว่า Subscriber ถ้าหากตัวรับต้องการรับข้อมูลจากตัวส่งจะต้องทำการ Subscribe หัวข้อ Topic ของ Publisher นั้นๆ ผ่าน Broker เช่นกัน ลองดูความสัมพันธ์ ตามรูป



3.5 ข้อแตกต่างระหว่าง IoT กับ Over Internet



การทดลองก่อนหน้านี้เป็นการควบคุมผ่านอินเทอร์เน็ต จำเป็นต้องรู้ IP ของอุปกรณ์ปลายทาง และระบบต้องอยู่ในวงเครือข่ายเดียวกัน เช่น จากรูปเราไม่สามารถใช้ NB_2 เข้ามาควบคุม ESP32 ได้โดยตรงเพราะ IP=ZZ.ZZ.ZZ.ZZ และ IP=YY.YY.YY.YY อยู่คนละเครือข่าย หากต้องการสามารถกำหนดเส้นทางจาก NB_2 ผ่านเครือข่ายของมหาวิทยาลัย ไปยังผู้ให้บริการมือถือ วนมาที่มือถือ เข้ามายัง ESP32 การควบคุมสั่งการแบบนี้จำเป็นต้องรู้เลขปลายทางซึ่งเป็นไอพีของอุปกรณ์

กรณีของ IoTs กระบวนการข้างต้นจะปรับเปลี่ยน คือ ไม่จำเป็นต้องรู้เลขไอพีของอุปกรณ์ปลายทางแต่ให้อุปกรณ์วิ่งไปรับคำสั่งที่ตัวกลาง (Broker) แทน จากรูป NB_2 จะส่งคำสั่ง (Publish) ไปยังตัวกลาง ตัวอุปกรณ์ปลายทางต้องแจ้งรับข้อความ (Subscribe) จากตัวกลาง เมื่อมีคำสั่งเข้ามา และตัวอุปกรณ์ปลายทางเข้ามารับข้อมูลอุปกรณ์ปลายทางค่อยทำงานตามคำสั่งที่ได้รับ

เห็นได้ว่าแบบแรกจำเป็นต้องเข้าให้ถึง ESP32 แต่แบบหลังใช้วิธีนี้รับข้อความที่ตัวกลางที่ทั้งสองฝั่งตกลงกันได้

Lab303 – ESP32 to Public/Subscribe MQTT Server

1. Add Library → **PubSubClient Version 2.8.0** ของ **Nick O'Leary** แล้วสามารถกดปุ่ม Install เพื่อติดตั้ง

PubSubClient
by **Nick O'Leary** Version **2.8.0** **INSTALLED**
A client library for MQTT messaging. MQTT is a lightweight messaging protocol ideal for small devices. This library allows you to send and receive MQTT messages. It supports the latest MQTT 3.1.1 protocol and can be configured to use the older MQTT 3.1 if needed. It supports all Arduino Ethernet Client compatible hardware, including the Intel Galileo/Edison, ESP8266 and TI CC3000.
[Library information](#)
[Select version](#) [Install](#)

- โปรแกรมจะ Publish ไปยัง **test.mosquitto.org** ในหัวข้อ **myHome1234**
- คำแนะนำการติดตั้งใช้งานตามนี้ <http://www.steves-internet-guide.com/using-mqtt-lens/>
- เปิด MQTT Lens แล้ว ตั้งค่าโปรคเกอร์




MQTT Lens

- Name = อะไรก็ได้
- Host tcp:// = **test.mosquitto.org**
- Port = **1883**
- Topic, Subscribe, Publish = **myHome1234**

Connection Details

Connection name
pk007

Connection color scheme


Hostname
tcp:// **test.mosquitto.org**

Port
1883

Connections

pk007



Connection: Pk007

Subscribe
myHome1234 0 - at most once SUBSCRIBE

Publish
myHome1234 0 - at most once ☐ Retained PUBLISH

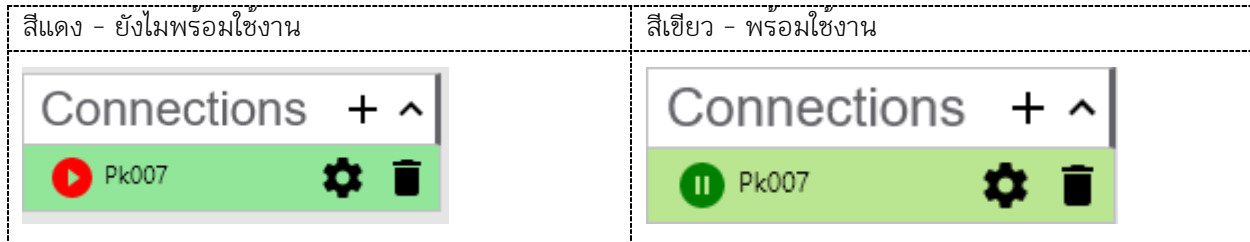
Message
hello

Subscriptions

Topic: "myHome1234" Showing the last 5 messages — +

#	Time	Topic	QoS
0	7:00:19	myHome1234	0

Message: hello



5. โปรแกรมทดสอบ Publish ไปยัง **test.mosquitto.org** ในหัวข้อ **myHome1234**
6. โหลดโปรแกรมไปที่ ESP-32 < อย่าลืมแก้ไข SSID, PASSWORD >

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Mue.Home";
const char* password = "pk1212312121";
const char* mqtt_server = "test.mosquitto.org";
const char* myTopic = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);

unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32_Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.subscribe(myTopic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  Serial.print("\n Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\n WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, 1883);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    float RanValue = random(6000, 9999) / 100;
    snprintf(msg, MSG_BUFFER_SIZE, "Random Data = %0.2f", RanValue);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(myTopic, msg);
  }
}
```

7. ผลการทำงานดูที่ Serial Monitor ว่าส่งอะไรออกไป และดูที่ MQTTLens ว่าได้รับอะไรมาบ้าง

The top image is a screenshot of a Serial Monitor window connected to COM3. It shows a list of published messages: "Publish message: Random Data = 71.00" through "Publish message: Random Data = 85.00". The left sidebar shows the "Subscriptions" section with a topic "myHome1234" and a message "Message: Random Data = 86.00". The bottom status bar indicates "56 10:42:44 myHome1234 0" and "Message: Random Data = 85.00".

The bottom image is a network diagram illustrating the MQTT setup. A Raspberry Pi (labeled "NB_1") and a laptop (labeled "NB_2") are connected via Wi-Fi to a central "SUT-WiFi" router. The router is connected to a cloud network containing various services: "AIS", "TrueMove", "DTAC", "Etc. SP", "SUT Network", and "www.Blynt.co". The diagram also shows a smartphone connected to the network with the text "AP = SUT_IoT" and "PSW = MisMeeljing".

8. โหลดโปรแกรมที่ทำงาน รับค่าแล้วควบคุม LED D2

```

#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "Mue.Home";
const char* password = "pk1212312121";
const char* mqtt_server = "test.mosquitto.org";
const char* myTopic = "myHome1234";

#define MSG_BUFFER_SIZE (50)
#define TestLED 2

WiFiClient espClient;
PubSubClient client(espClient);

unsigned long lastMsg = 0;
char msg[MSG_BUFFER_SIZE];

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32_Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.subscribe(myTopic);
    } 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)
{ char myPayload[50];
  Serial.print("Message arrived [");
  Serial.print(myTopic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
    myPayload[i] = payload[i];
    myPayload[i + 1] = '\0'; // End of String
  }
  Serial.print("\n ---> "); Serial.println(myPayload);
  myPayload[4] = '\0'; // String less than 4 Charector
  if ((String)myPayload == "ON") digitalWrite(TestLED, HIGH);
  if ((String)myPayload == "OFF") digitalWrite(TestLED, LOW);
}

void setup() {
  Serial.begin(115200);
  pinMode(TestLED, OUTPUT);
  Serial.print("\n Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500); Serial.print(".");
  }
  Serial.println("\n WiFi connected");
  Serial.println("IP address: "); Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }

  client.loop();

  unsigned long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    float RanValue = random(6000, 9999) / 100;
    snprintf(msg, MSG_BUFFER_SIZE, "Random Data = %0.2f", RanValue);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(myTopic, msg);
  }
}

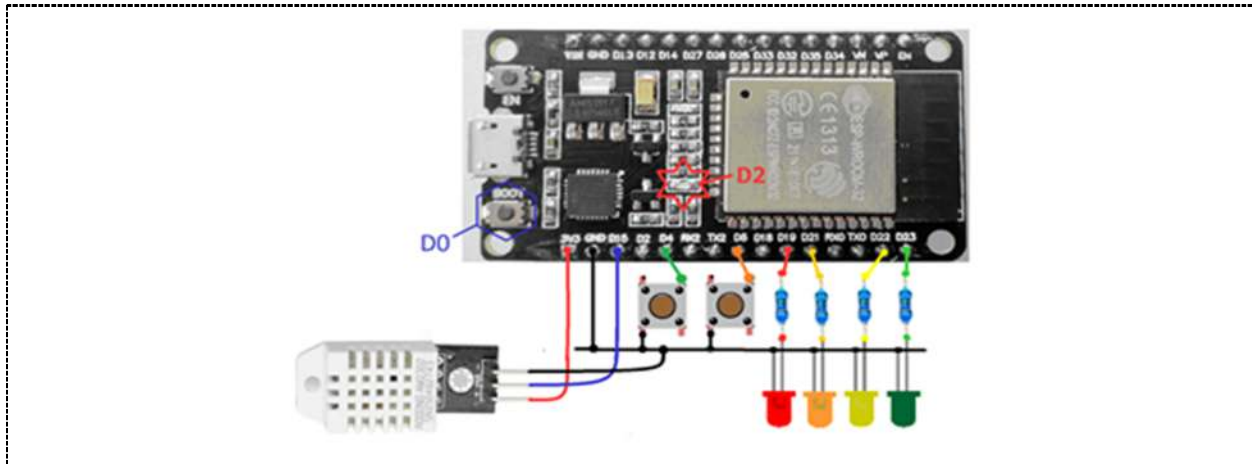
```


9. ทดสอบ publish “OFF” → Off LED และ “ON” → On LED

<p>Connection: Pk007</p> <p>Subscribe</p> <p>myHome1234 0 - at most once SUBSCRIBE</p> <p>Publish</p> <p>myHome1234</p> <p>0 - at most once <input type="checkbox"/> Retained PUBLISH</p> <p>Message</p> <p>ON</p>	
<p>Connection: Pk007</p> <p>Subscribe</p> <p>myHome1234 0 - at most once SUBSCRIBE</p> <p>Publish</p> <p>myHome1234</p> <p>0 - at most once <input type="checkbox"/> Retained PUBLISH</p> <p>Message</p> <p>OFF</p>	
<pre> ---> Random Data = 91.00 Message arrived [myHome1234] Random Data = 66.00 ---> Random Data = 66.00 Message arrived [myHome1234] ON ---> ON Publish message: Random Data = 74.00 Publish message: Random Data = 84.00 Message arrived [myHome1234] Random Data = 74.00 ---> Random Data = 74.00 Message arrived [myHome1234] OFF ---> OFF Message arrived [myHome1234] Random Data = 84.00 ---> Random Data = 84.00 Publish message: Random Data = 73.00 </pre>	

10. ออกแบบโปรแกรมเพื่อทดสอบ Pub/Sub Data from (DHT22 + 4 LED + 2 Switch)

- อ่านค่า DHT-22 แล้วส่งไปยัง MQTT Broker ทุกๆ 5 วินาที
- กำหนดให้ใช้ mqtt.eclipse.org เป็น Broker
- ควบคุมการปิดเปิด 4 LED
- รับคำสั่งที่กำหนด SW1 แจ้ง Overheat Alarm, SW2 แจ้ง Intruders Alarm



4/6 - การใช้งาน Raspberry Pi เพื่อแสดงค่าและควบคุมผ่าน Public MQTT Server

<https://www.youtube.com/watch?v=DEU4nhCD8Xw>

<https://medium.com/mmp-li/เริ่มต้นใช้งาน-node-red-ฉบับปี-2018-3fca5ed140f9>

Message Queuing Telemetry Transport (MQTT) เป็น Protocol ที่ออกแบบมาเพื่อการเชื่อมต่อแบบ M2M (machine-to-machine) ออกแบบมาให้สนับสนุน IoT (Internet of Thing) คือเทคโนโลยีที่อินเทอร์เน็ตเชื่อมต่อกับอุปกรณ์ต่างๆ เช่น Smart phone, Raspberry pi เป็นต้น ทำให้เราสามารถเชื่อมต่ออุปกรณ์เข้าด้วยกันผ่านระบบอินเทอร์เน็ตได้ และยังออกแบบมาเพื่อใช้งานกับอุปกรณ์อิเล็กทรอนิกส์ขนาดเล็ก การรับส่งข้อมูลในเครือข่ายที่มีขนาดเล็ก แบบวิธีต่ำ ใช้หลักการแบบ Publisher / Subscriber และมีตัวกลางคือ Broker ซึ่งจะคอยจัดการกับข้อมูลจาก Publisher และ Subscriber

สรุปองค์ประกอบของ MQTT Protocol


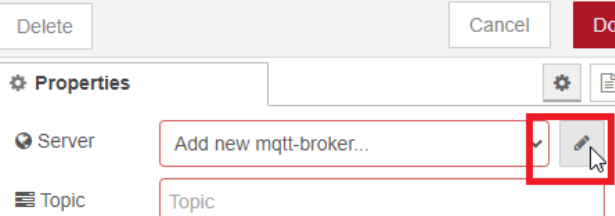
- Broker ทำหน้าที่เป็นตัวกลางคอยจัดการกับ ข้อความโดย อ้างอิงจาก Topic
- Publisher ทำหน้าที่คอยส่งข้อมูลไปยัง Topic
- Subscriber ทำหน้าที่คอยดูการเปลี่ยนแปลงของ message ที่อ้างอิงด้วย Topic เช่นถ้ามีหัวข้อที่น่าสนใจและมีการเปลี่ยนแปลงก็จะทำการดึงข้อมูลนั้นๆ มาใช้งาน
- QoS คือ คุณภาพหรือความสำคัญของการส่งข้อมูล QoS0 คือการส่งข้อมูลเพียงครั้งเดียว ปลายทางอาจจะได้รับหรือไม่ได้รับก็ได้ QoS1 คือการส่งข้อมูลเพียงครั้งเดียว ปลายทางอาจจะได้รับหรือไม่ได้รับก็ได้ แต่จะมีการจำข้อมูลที่ส่งล่าสุดไว้ เมื่อมีการเชื่อมต่อเข้ามาก็จะได้รับข้อมูลล่าสุดที่ส่งไป QoS2 คือการส่งข้อมูลไปจนการปลายทางจะได้รับ

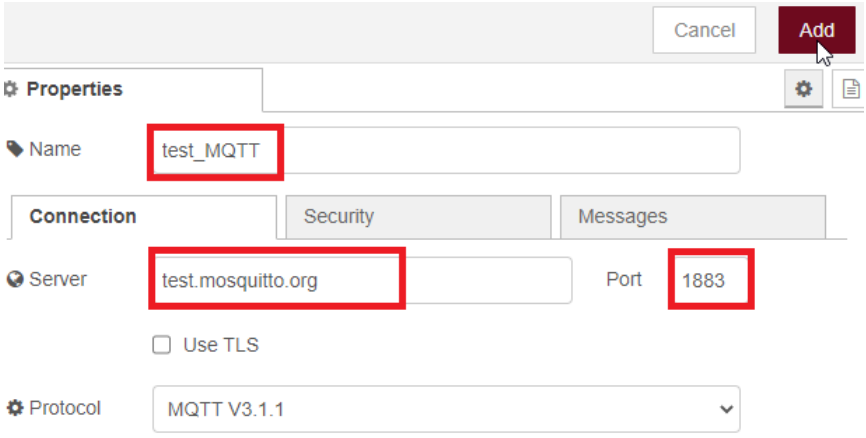
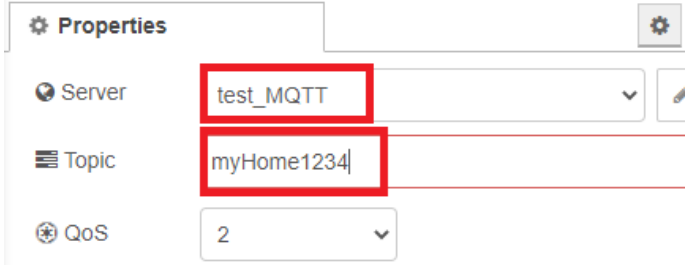
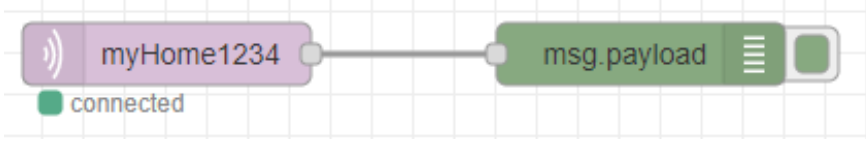
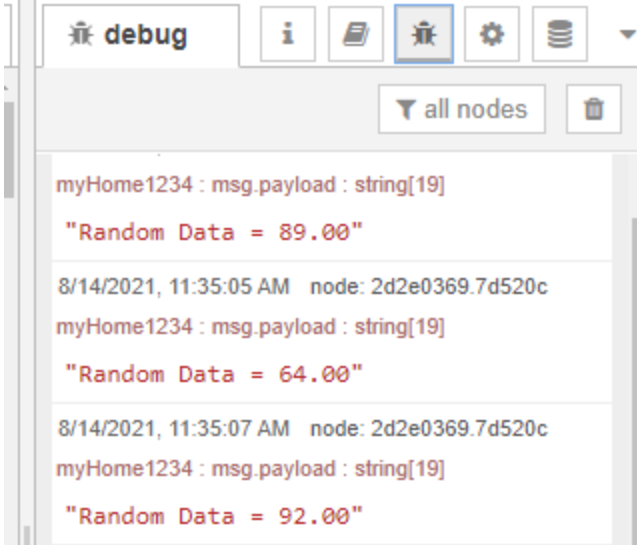
แนะนำ Link ของ Broker

- test.mosquitto.org Port 1883 (เป็น Broker ที่ใช้ในบทความนี้)
- mqtt.eclipseprojects.io Port 1883
- broker.hivemq.com Port 1883

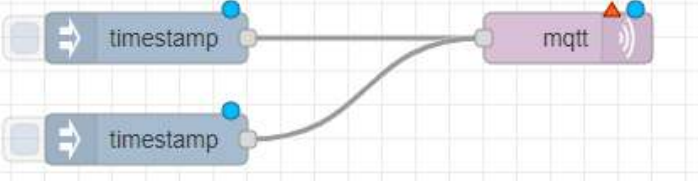
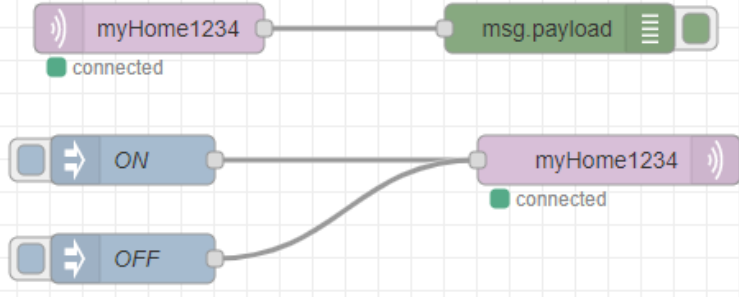
Lab304 – Raspberry Pi to Public MQTT Server

1. วาง 2 โหนดและกำหนดค่า เพื่อทดสอบการรับค่าจาก Broker < Subscribe Test >

	Node = mqtt-in Node = debug
	Add Broker

	Name = test_MQTT Server = test.mosquitto.org Port = 1883
	Server = test_MQTT Topic = myHome1234
	Deploy
	Data from Topic "myHome1234"

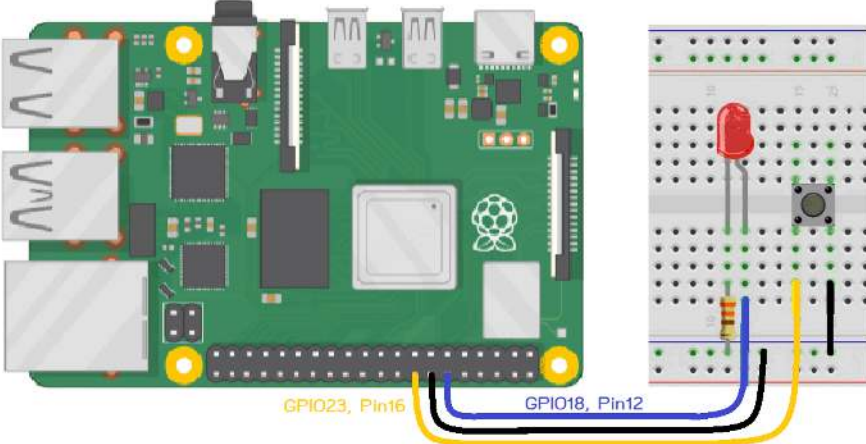
2. ทดสอบส่งค่า On/Off เพื่อควบคุม LED D2 บน ESP-32 ผ่าน myHome1234 Topic < Publish Test >

	
<p>Properties</p> <p>Name ON</p> <p>msg. payload = ON</p> <p>msg. topic = a_z</p>	
<p>Properties</p> <p>Name OFF</p> <p>msg. payload = OFF</p> <p>msg. topic = a_z</p>	
<p>Properties</p> <p>Server test_MQTT</p> <p>Topic myHome1234</p> <p>QoS 1 Retain <input type="checkbox"/></p> <p>Name Name</p>	
	

<pre> 8/14/2021, 11:50:36 AM node: 2d2e0369.7d520c myHome1234 : msg.payload : string[3] "OFF" </pre>	
<pre> 8/14/2021, 11:50:37 AM node: 2d2e0369.7d520c myHome1234 : msg.payload : string[19] "Random Data = 68.00" </pre>	
<pre> 8/14/2021, 11:50:40 AM node: 2d2e0369.7d520c myHome1234 : msg.payload : string[19] "Random Data = 65.00" </pre>	
<pre> 8/14/2021, 11:50:40 AM node: 2d2e0369.7d520c myHome1234 : msg.payload : string[2] "ON" </pre>	
<pre> 8/14/2021, 11:50:51 AM node: 2d2e0369.7d520c </pre>	


3. ทดสอบใช้งาน Broker “mqtt.eclipseprojects.io:1883 (Topic = myHome1428)เพื่อรับค่าการนับสวิตช์และควบคุม LED ที่ RPi

3.1 วงจรทดสอบ

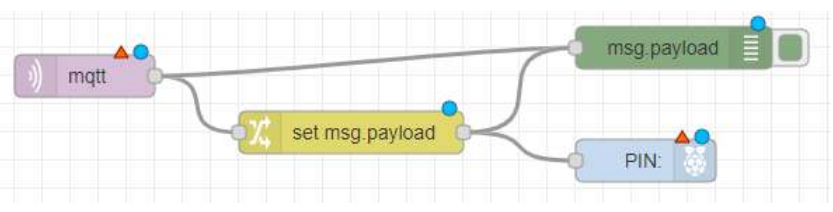
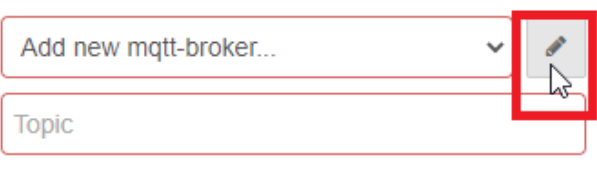
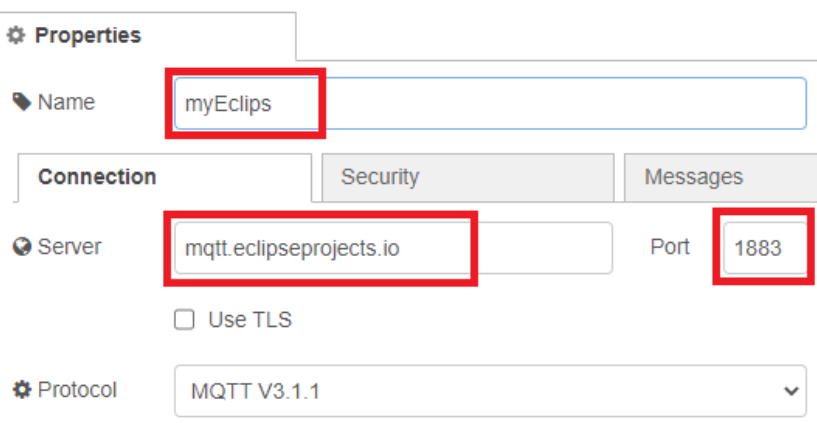
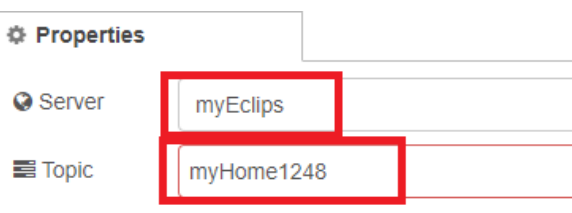
	<p>LED = GPIO18, Pin12</p> <p>SW = GPIO23, Pin16</p>
--	--

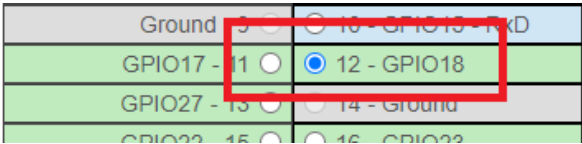
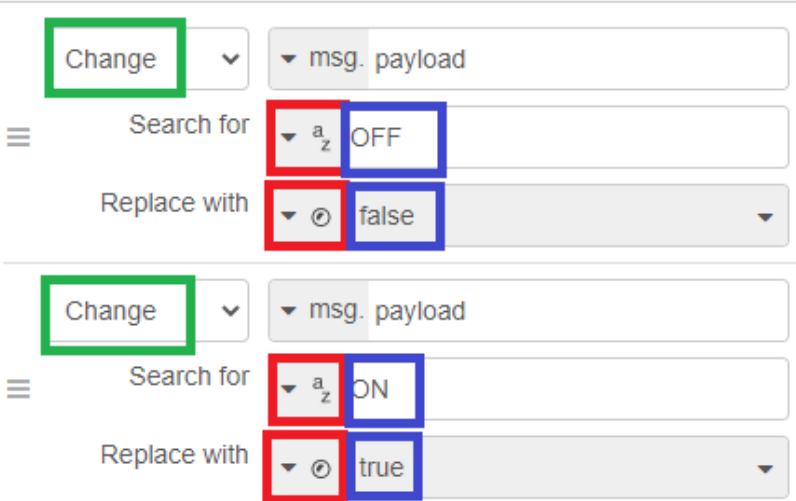
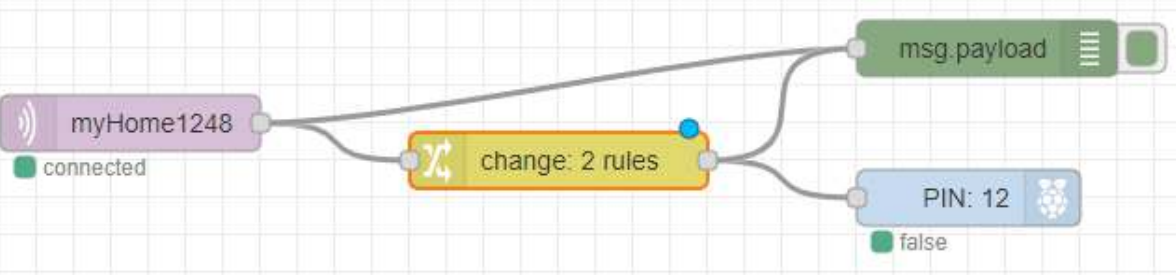
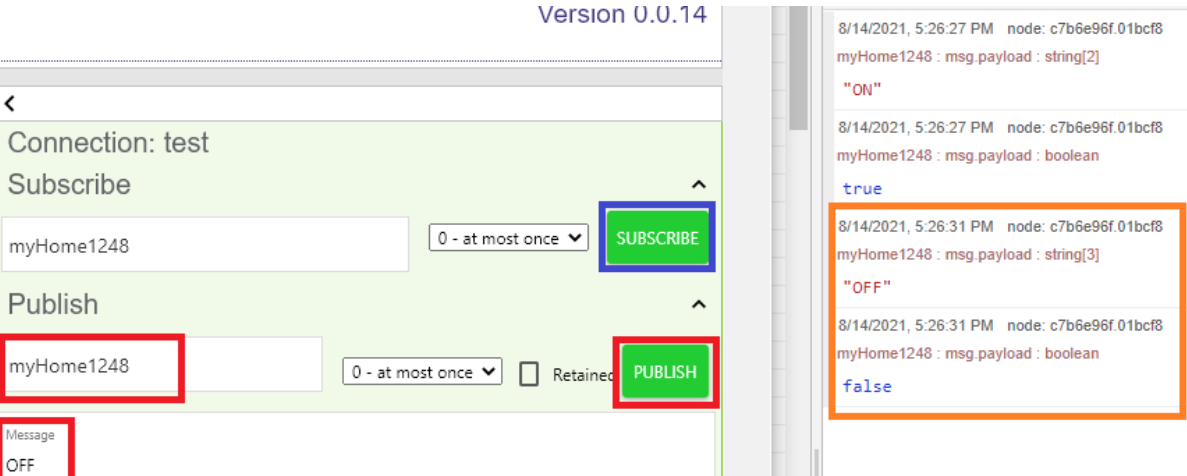
3.2 กำหนดค่าที่ MQTTLens

<p>Connection name</p> <p>test</p> <p>Connection</p> <p>Hostname</p> <p>tcp:// mqtt.eclipseprojects.io</p> <p>Port</p> <p>1883</p> <p>Client ID</p> <p>lens_THbnJVA4wWiUn9LHeBA5EIs0nvJ</p> <p>Ge</p>	<p>mqtt.eclipseprojects.io</p> <p>1883</p>
---	--

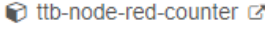
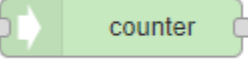
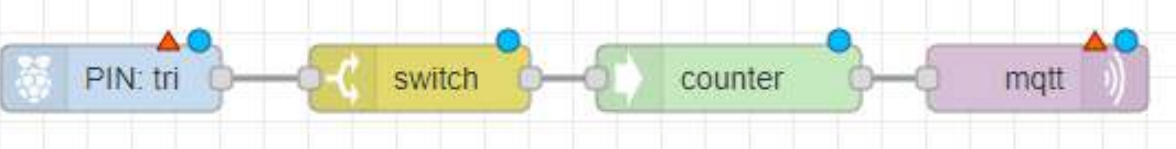
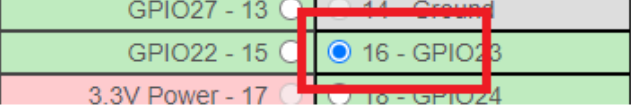
	<p>Topic = myHome1248</p>
--	---------------------------

3.3 ทดสอบการรับค่าจาก MQTTLens เพื่อควบคุม LED

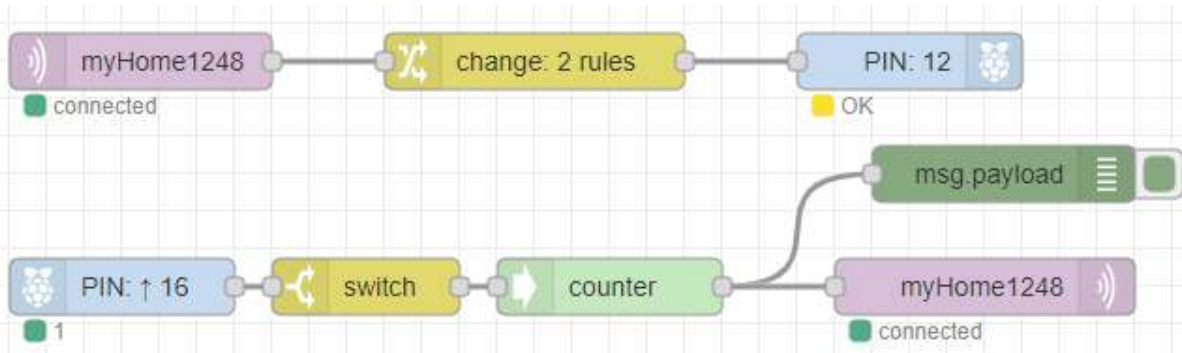
	<p>Node: mqtt-in Node: rpi-gpio out Node: change Node: Debug</p>
	<p>Add Broker</p>
	<p>N = myEclips S = mqtt.eclipseprojects.io P = 1883</p>
	<p>Set Server Topic = myHome1248</p>

	<p>Rpi-gpio-out = Pin12</p>
	<p>Get</p> <p>OFF Change to false ON Change to true</p>
<p>Deploy</p> 	
<p>MQTTLens "ON"/"OFF"Publish</p> <p>Version 0.0.14</p> 	

3.4 ทดสอบนับจำนวนการกดสวิตช์แล้วแสดงค่าที่ MQTTLens

 <p>ttb-node-red-counter 🔗</p> <p>A node-red node to increment or decrement. You can easily create a daily tweets counter with it for example.</p> <p>0.1.0 6 years ago</p> <p>install</p>	<p>Add Node</p> <p>>> ttb-node-red-counter, V0.1.0</p>
	<p>Counter node</p>
<p>Add Flow</p> 	
 <p>GPIO27 - 13 44 - Ground</p> <p>GPIO22 - 15 16 - GPIO23</p> <p>3.3V Power - 17 18 - GPIO24</p> <p>16</p> <p>pullup Debounce 25 mS</p>	<p>rpi-gpio out</p> <p>>> pin 16</p> <p>>> pullup</p> <p>>> 25ms</p>
<p>Property msg. payload</p> <p>== a_z 1 → 1 ✕</p>	<p>Switch</p> <p>>> == 1 then pass</p> <p>< ปุ่มกด ></p>
<p>Properties</p> <p>Increment 1</p> <p>Name Name</p>	<p>Counter</p> <p>>> Ince = 1</p>
<p>Server myEclips</p> <p>Topic myHome1248</p>	<p>Mqtt-out</p> <p>>> server = myEclips</p> <p>>> Topic = myHome1248</p>

Deploy



กดสวิตช์ เพื่อบันทึก

Connection: test

Subscribe

myHome1248 0 - at most once SUBSCRIBE

Publish

myHome1248 0 - at most once Retained PUBLISH

Message

OFF

Subscriptions

Topic: "myHome1248" Showing the last 1 messages

#	Time	Topic	QoS
149	6:13:27	myHome1248	0

Message: 5

8/14/2021, 6:13:11 PM
pi/16 : msg.payload : 1

2

8/14/2021, 6:13:12 PM
msg : string[16]
"Invalid input:"

8/14/2021, 6:13:21 PM
pi/16 : msg.payload : 1

3

8/14/2021, 6:13:22 PM
msg : string[16]
"Invalid input:"

8/14/2021, 6:13:22 PM
pi/16 : msg.payload : 1

4

8/14/2021, 6:13:22 PM
msg : string[16]
"Invalid input:"

8/14/2021, 6:13:27 PM
pi/16 : msg.payload : 1

5

Property

msg. payload

== a_z 0 → 1 x

แก้ไข Switch

>> == 0 then pass

< กดปุ่ม >

5/6 - การติดตั้ง Private MQTT Server บน Raspberry Pi และการทดสอบ

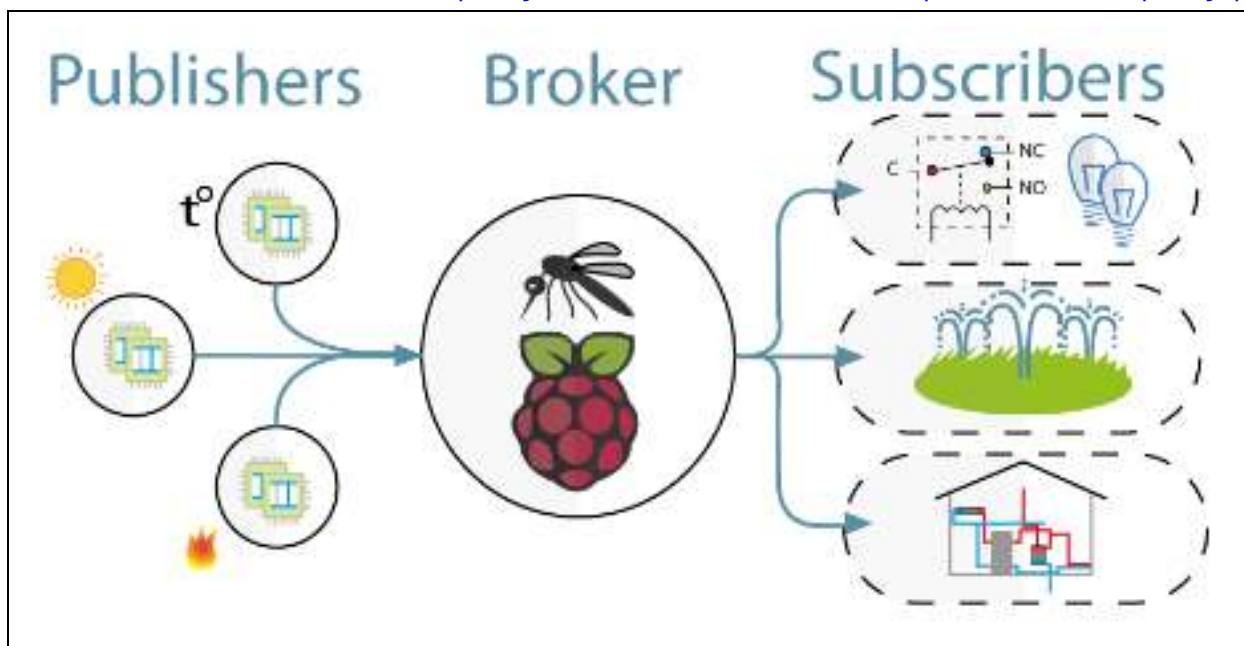
<https://www.hackster.io/dhairya-parikh/running-a-mqtt-broker-on-raspberry-pi-63c348>

<https://raspberry-valley.azurewebsites.net/MQTT-on-Raspberry-Pi/>

<https://iotbytes.wordpress.com/mosquitto-mqtt-broker-on-raspberry-pi/>

http://openfog.net/testboardesp8266_doc_3.html

www.raspberrypihome.net/article/5/การติดตั้ง-mqtt-broker-บน-raspberry-pi



การพัฒนาทางด้าน [IoT](#) หรือ Internet of things นั้น คงจะหนีไม่พ้นเรื่อง protocol หรือวิธีการส่งข้อมูลนะครับ และแน่นอนครับ วันนี้เราจะใช้ MQTT Protocol ในการรับส่งข้อมูล และบทความนี้จะเป็นการเริ่มต้นและสร้างความคุ้นเคย กับงานพัฒนาทางด้าน IoT (Internet of things) กันด้วยการติดตั้ง MQTT Broker & MQTT Web Client บน Raspberry Pi

MQTT Protocol

MQTT เป็น Application Protocol ที่วิ่งอยู่บน TCP/IP ซึ่งถูกสร้างขึ้นโดย Dr Andy Stanford-Clark of IBM, และ Arlen Nipper of Arcom (now Eurotech) เมื่อนานมาแล้ว ตั้งแต่ปี 1999 หรือประมาณ 17 ปีมาแล้ว ซึ่งก็ออกแบบมาสำหรับ Device ที่มีพลังงานประมวลผลต่ำ ต้องการประหยัดพลังงาน และสภาพอินเทอร์เน็ตที่แย่มากๆ โดยกำหนดให้มี QoS ถึง 3 ระดับ คือ

- QoS0 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่
- QoS1 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่ แต่ให้จำค่าที่ส่งล่าสุดไว้ เมื่อมีการเชื่อมต่อใหม่จะได้รับข้อมูลครั้งล่าสุดอีกครั้ง
- QoS2 – ส่งข้อมูลหลาย ๆ ครั้งจนกว่าปลายทางจะได้รับข้อมูล มีข้อเสียที่สามารถทำงานได้ช้ากว่า QoS0 และ QoS1

MQTT คืออะไร

MQTT ย่อมาจาก Message Queuing Telemetry Transport เป็นโปรโตคอล machine-to-machine (M2M) ใช้ในงาน IOT (Internet of Things) ออกแบบมาให้มีขนาด package ที่มีขนาดเล็กมาก ให้ทำการ publish (ส่งข้อมูล) และ Subscribe (ติดตามข้อมูล) ไปยังตัว Server ที่เรียกกันว่า MQTT Broker ซึ่งตัว Server จะทำหน้าที่จัดคิว ของข้อมูลต่างๆ ให้มีประสิทธิภาพสูงสุด

Mosquitto MQTT

สาเหตุที่เราเลือกใช้ Mosquitto MQTT และทำ Server เองนั้นก็เนื่องจากว่า จริงๆแล้วมีการให้บริการ MQTT Server ฟรีอย่างมากมายในปัจจุบันทั้งในรูปแบบฟรีและ ไม่ฟรี ซึ่งปัญหาที่ตามมา ก็คือ บาง Server มีที่ตั้งของ Server อยู่ไกลมากอีกฟากหนึ่งของโลก การทำการ Publish ข้อมูล และ Subscribe นั้นทำให้เกิดการ Delay เกิดขึ้น และ อีกสาเหตุ ก็คือ บางผู้ให้บริการ จะมีการจำกัด จำนวน Client จำกัดจำนวนการส่งข้อมูล หรือไม่ก็จำกัด การใช้งาน Application ซึ่งจะทำให้เราขยายและต่อยอด ออกไปในอนาคตไม่ได้

ส่วนสาเหตุที่เลือก ตัว Raspberry Pi เป็น Server ก็เนื่องมาจากว่า ตัว MQTT Broker นั้นทำหน้าที่คอยจัดการ message ที่มีขนาดเล็กดังนั้นตัว Raspberry Pi จึงสามารถเชื่อมต่ออุปกรณ์ Client ได้เป็น พันๆ Client ซึ่งนับว่าค่อนข้างเยอะเลยทีเดียว ซึ่งถ้าเทียบกับการที่เราเอามาทดลองและเอามาใช้งานเองนั้น ก็ถือว่าเพียงพอต่อความต้องการของเราแล้ว



การติดตั้ง MQTT Broker บน Raspberry Pi

MQTT (Message Queuing Telemetry Transport) เป็น โพรโตคอล (Protocol) สำหรับการติดต่อสื่อสารและรับส่งข้อมูลระหว่างอุปกรณ์ต่างๆ ที่เชื่อมต่อกันอยู่บนระบบเครือข่ายภายใน(LAN)และระบบเครือข่ายอินเทอร์เน็ต อาทิ เช่น คอมพิวเตอร์ สมาร์ทโฟน เซ็นเซอร์อุณหภูมิ เซ็นเซอร์ตรวจจับการเคลื่อนไหว สวิตช์และปลั๊กไฟอัจฉริยะ อุปกรณ์ระบบกันขโมย อุปกรณ์ระบบควบคุมอัตโนมัติ และอุปกรณ์อื่นๆ เป็นต้น การรับส่งข้อมูลด้วย MQTT นั้นรวดเร็วเนื่องจากข้อมูลมีขนาดเล็กมากสามารถส่งไปยังหรือรับข้อมูลจากอุปกรณ์จำนวนมากได้พร้อมๆ กัน โดยใช้ระบบ Broker, Publisher, Subscriber และ Topic

โบรกเกอร์นั้นก็คือ MQTT Server ที่เรากำลังจะติดตั้ง ส่วนพับลิเชอร์และซับสไคร์เบอร์นั้นคืออุปกรณ์ต่างๆ ที่เชื่อมต่อกันอยู่ในระบบเน็ตเวิร์ค อุปกรณ์ที่ทำหน้าที่เป็นพับลิเชอร์จะส่งข้อมูลตามหัวข้อ Topic ไปยังโบรกเกอร์ หากอุปกรณ์ใดๆ ที่สมัคร (Subscribe) ในหัวข้อ Topic นั้นไว้ โบรกเกอร์จะส่งข้อมูลตามหัวข้อ Topic นั้นไปยังอุปกรณ์ต่างๆ ทั้งหมดที่สมัคร (Subscribe) ไว้ทั้งหมด ส่วนอุปกรณ์ใดๆ ไม่ได้สมัคร (Subscribe) ในหัวข้อ Topic นั้นไว้ ก็จะไม่ได้รับข้อมูล เพราะโบรกเกอร์ก็จะไม่ส่งข้อมูลในหัวข้อ Topic นั้นไปให้

อนึ่ง อุปกรณ์ทุกชิ้นในระบบเน็ตเวิร์คสามารถส่ง(Publish)หรือรับ(Subscribe)ข้อมูลในทุกหัวข้อ Topic ได้ตามที่กำหนด เราจะเห็นได้ว่าในปัจจุบัน มีการนำ MQTT มาใช้อย่างแพร่หลาย เพราะข้อมูลมีขนาดเล็กกะทัดรัด สามารถกระจาย(Distribute)ข้อมูลในหัวข้อ Topic ต่างๆ ไปยังอุปกรณ์ (Subscribers)จำนวนมากได้อย่างรวดเร็ว ไม่ว่าอุปกรณ์เหล่านั้นจะอยู่ห่างไกลจากกันเพียงใดก็ตาม ข้อดีของ MQTT ก็คือไม่ต้องเสียเวลาในการฟอร์เวิร์ดพอร์ต(Forward port)ให้กับอุปกรณ์ทุกชิ้นในระบบเน็ตเวิร์ค (ยกเว้นตัว โบรกเกอร์ที่ต้อง Forward port) เพื่อเปิดการสื่อสารทางไกลผ่านอินเทอร์เน็ต MQTT Server หรือโบรกเกอร์(Broker) ยังทำหน้าที่เป็นตัวกลาง ซึ่งจะคอยตรวจสอบการรับส่งข้อมูลระหว่างอุปกรณ์ต่างๆ ที่เชื่อมต่อกันในระบบ เพื่อให้แน่ใจว่าอุปกรณ์ซับสไคร์เบอร์ทุกชิ้นในระบบได้รับข้อมูลที่ส่งมาจากพับลิเชอร์อย่างแน่นอน ถึงแม้ว่าในขณะที่พับลิเชอร์กำลังส่งข้อมูลตามหัวข้อ Topic อยู่ นั้น อุปกรณ์ที่เป็นซับสไคร์เบอร์ของหัวข้อ Topic นั้นจะถูกปิด(Offline)อยู่ก็ตาม แต่เมื่ออุปกรณ์ที่เป็นซับสไคร์เบอร์นั้นถูกเปิด(Online)กลับมาใช้งานอีกครั้ง โบรกเกอร์(Broker)ก็จะส่งข้อมูลนั้นในหัวข้อ Topic นั้นไปให้กับอุปกรณ์ที่เป็นซับสไคร์เบอร์ ซึ่งทำให้การสื่อสารรับส่งข้อมูลครบถ้วนสมบูรณ์ โดยมีขั้นตอนการติดตั้ง MQTT Server เพื่อทำหน้าที่เป็น Broker โดยมีขั้นตอนดังต่อไปนี้

Lab305 – Install Private MQTT Server

1. ป้อนคำสั่งเพื่อ Update/ Upgrade Raspbian และ เริ่มบูทเครื่องใหม่

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
Sudo reboot
```

2. เริ่มติดตั้งโปรแกรม Mosquitto (MQTT Server), และ MQTT Clients

2.1 SSH into Raspberry Pi and create a new directory for temp files –

```
mkdir mosquitto
```

```
cd mosquitto
```

2.2. Import the repository package signing key –

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
```

```
pi@raspberrypi:~/mosquitto $ wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
--2021-08-14 15:20:54-- http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
Resolving repo.mosquitto.org (repo.mosquitto.org)... 2001:ba8:1f1:f271::2, 85.19.83.194
Connecting to repo.mosquitto.org (repo.mosquitto.org)|2001:ba8:1f1:f271::2|:80.
. connected.
HTTP request sent, awaiting response... 200 OK
Length: 3167 (3.1K) [application/octet-stream]
Saving to: 'mosquitto-repo.gpg.key'

mosquitto-repo.gpg. 100%[=====>] 3.09K --.-KB/s in 0s
2021-08-14 15:20:55 (35.6 MB/s) - 'mosquitto-repo.gpg.key' saved [3167/3167]
```

```
sudo apt-key add mosquitto-repo.gpg.key
```

```
pi@raspberrypi:~/mosquitto $ sudo apt-key add mosquitto-repo.gpg.key
OK
```

2.3. Make the repository available to apt –

```
cd /etc/apt/sources.list.d/
```

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list
```

```
pi@raspberrypi:/etc/apt/sources.list.d $ sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list
--2021-08-14 15:22:45-- http://repo.mosquitto.org/debian/mosquitto-stretch.list
Resolving repo.mosquitto.org (repo.mosquitto.org)... 85.119.83.194, 2001:ba8:1f:f271::2
Connecting to repo.mosquitto.org (repo.mosquitto.org)|85.119.83.194|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 51 [application/octet-stream]
Saving to: 'mosquitto-stretch.list'

mosquitto-stretch.l 100%[=====>]          51 --.-KB/s    in 0s
2021-08-14 15:22:46 (2.03 MB/s) - 'mosquitto-stretch.list' saved [51/51]
```

```
sudo apt-get update
```

```
pi@raspberrypi:/etc/apt/sources.list.d $ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian buster InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian buster InRelease
Get:3 https://repo.mosquitto.org/debian stretch InRelease [12.4 kB]
Get:4 https://repo.mosquitto.org/debian stretch/main all Packages [5,223 B]
Get:5 https://repo.mosquitto.org/debian stretch/main armhf Packages [44.6 kB]
Fetched 62.2 kB in 3s (19.6 kB/s)
Reading package lists... Done
```

```
sudo apt-cache search mosquitto
```

2.4. Install Mosquitto MQTT Broker and MQTT Clients –

```
sudo apt-get install mosquitto
```

```
sudo apt-get install mosquitto-clients
```

```
Selecting previously unselected package libmosquitto1:armhf.
(Reading database ... 167168 files and directories currently installed.)
Preparing to unpack .../libmosquitto1_2.0.11-0mosquitto1~stretch1_armhf.deb ...
Unpacking libmosquitto1:armhf (2.0.11-0mosquitto1~stretch1) ...
Selecting previously unselected package mosquitto-clients.
Preparing to unpack .../mosquitto-clients_2.0.11-0mosquitto1~stretch1_armhf.deb ...
Unpacking mosquitto-clients (2.0.11-0mosquitto1~stretch1) ...
Setting up libmosquitto1:armhf (2.0.11-0mosquitto1~stretch1) ...
Setting up mosquitto-clients (2.0.11-0mosquitto1~stretch1) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10+rpi1) ...
pi@raspberrypi:~ $
```

2.5. Check Mosquitto Service Status, Process and Default Port (1883) –

service mosquitto status

```

pi@raspberrypi:/etc/apt/sources.list.d $ service mosquitto status
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor p
   Active: active (running) since Sat 2021-08-14 15:24:41 BST; 1min 6s ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
   Main PID: 2440 (mosquitto)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/mosquitto.service
           └─2440 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Aug 14 15:24:40 raspberrypi systemd[1]: Starting Mosquitto MQTT Broker...
Aug 14 15:24:41 raspberrypi systemd[1]: Started Mosquitto MQTT Broker.
pi@raspberrypi:/etc/apt/sources.list.d $

```

ps -ef | grep mosq

```

pi@raspberrypi:/etc/apt/sources.list.d $ ps -ef | grep mosq
mosquit+ 2440      1  0 15:24 ?        00:00:00 /usr/sbin/mosquitto -c /etc/mo
uitto/mosquitto.conf
pi        2608    1677  0 15:26 pts/0    00:00:00 grep --color=auto mosq

```

netstat -tln | grep 1883

```

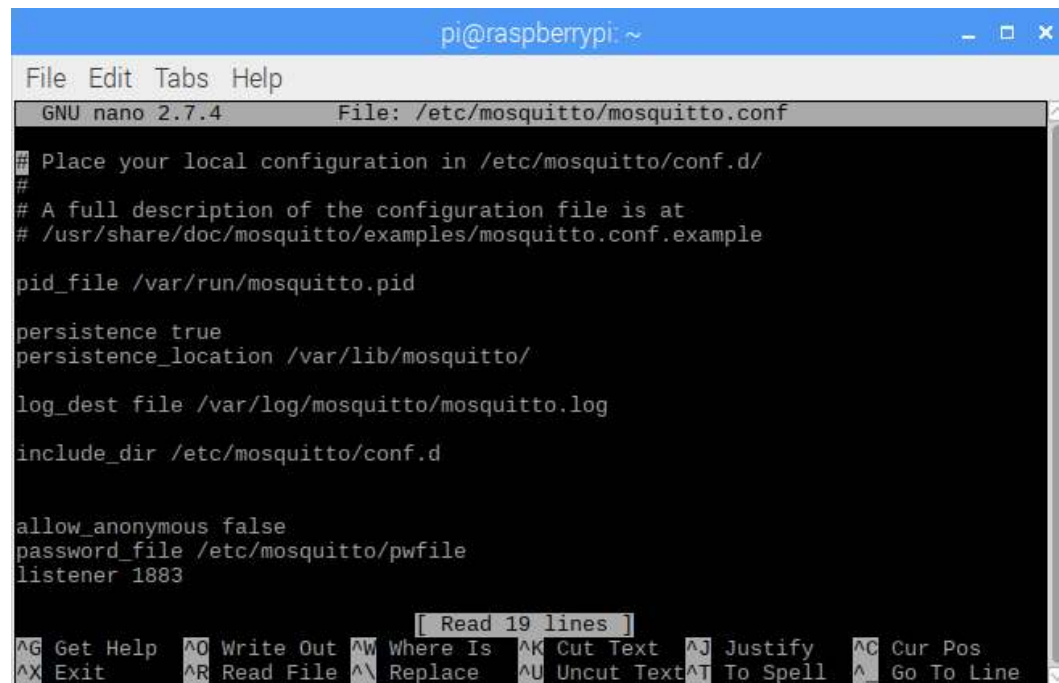
pi@raspberrypi:/etc/apt/sources.list.d $ netstat -tln | grep 1883
tcp        0      0 127.0.0.1:1883      0.0.0.0:*           LISTEN
tcp6       0      0 :::1883             :::*               LISTEN
pi@raspberrypi:/etc/apt/sources.list.d $

```

If you see Mosquitto service running and listening to TCP Port 1883, you have a functional MQTT Broker.

3. แก้ไขไฟล์คอนฟิกเกอร์เรชั่น โดยเพิ่มบรรทัดใหม่ 3 บรรทัดที่ด้านล่างสุดของไฟล์ แล้วบันทึกการเปลี่ยนแปลงในไฟล์ โดยกดปุ่ม Ctrl+O เมื่อปรากฏชื่อไฟล์ ให้กดปุ่ม Enter เพื่อยอมรับ แล้วออกจากเทกซ์อีดิเตอร์ โดยกดปุ่ม Ctrl+X

`sudo nano /etc/mosquitto/mosquitto.conf`



```

pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.7.4 File: /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example
pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

allow_anonymous false
password_file /etc/mosquitto/pwfile
listener 1883

[ Read 19 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^_ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

```

`allow_anonymous false`

`password_file /etc/mosquitto/pwfile`

`listener 1883`

4. สร้างยูสเซอร์ชื่อ mymqtt และกำหนดพาสเวิร์ดเป็น myraspi หรือพาสเวิร์ดอื่นๆ เก็บไว้ใน pwfile โดยป้อนคำสั่ง

`sudo mosquitto_passwd -c /etc/mosquitto/pwfile mymqtt`

Password: ป้อนพาสเวิร์ด `myraspi`

Reenter password: ป้อนพาสเวิร์ดยืนยัน `myraspi`

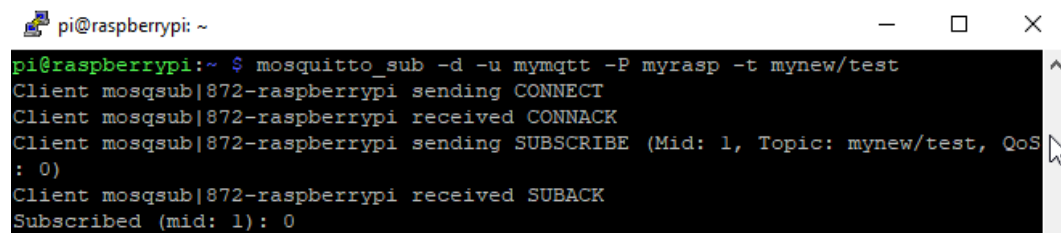
5. ในขณะนี้ เราได้ติดตั้ง Mosquitto เสร็จเรียบร้อยแล้ว แต่จะต้องบูทเครื่องใหม่ เพื่อให้ Mosquitto เริ่มทำงาน

sudo reboot

6. เริ่มทดสอบการทำงานของ MQTT Server หรือ Mosquitto โดยเราจะทดสอบด้วยการเรียกหน้าต่าง PuTTY ออกมา 2 หน้าต่าง เพื่อจำลองว่าเรามีอุปกรณ์ 2 ชิ้น ซึ่งหน้าต่างหนึ่งจะทำหน้าที่เป็นซับสไครเบอร์หรือตัวรับข้อมูล (Subscriber) อีกหน้าต่างหนึ่งจะทำหน้าที่เป็นพับบลิชเชอร์หรือตัวส่งข้อมูล (Publisher) เริ่มเรียกโปรแกรม PuTTY แล้ว Connect ใหม่ โดยใช้หมายเลข IP Address เดิม

6.1 การรับข้อมูล (Subscriber) → ป้อนซับสไครเบอร์ Topic ชื่อ mynew/test

mosquitto_sub -d -u mymqtt -P myraspi -t mynew/test



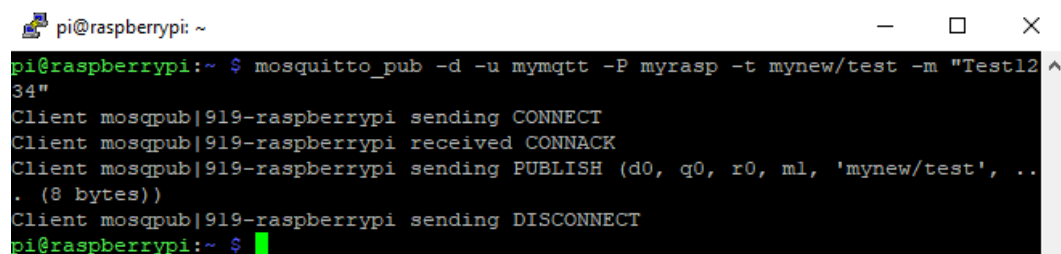
```

pi@raspberrypi: ~
pi@raspberrypi:~ $ mosquitto_sub -d -u mymqtt -P myraspi -t mynew/test
Client mosqsub|872-raspberrypi sending CONNECT
Client mosqsub|872-raspberrypi received CONNACK
Client mosqsub|872-raspberrypi sending SUBSCRIBE (Mid: 1, Topic: mynew/test, QoS: 0)
Client mosqsub|872-raspberrypi received SUBACK
Subscribed (mid: 1): 0

```

6.2 การส่งข้อมูล (Publisher) → ป้อนพับบลิชเชอร์ "ทดสอบ" ใน Topic ชื่อ mynew/test

mosquitto_pub -d -u mymqtt -P myraspi -t mynew/test -m "Test1234"



```

pi@raspberrypi: ~
pi@raspberrypi:~ $ mosquitto_pub -d -u mymqtt -P myraspi -t mynew/test -m "Test1234"
Client mosqpub|919-raspberrypi sending CONNECT
Client mosqpub|919-raspberrypi received CONNACK
Client mosqpub|919-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'mynew/test', ..
. (8 bytes))
Client mosqpub|919-raspberrypi sending DISCONNECT
pi@raspberrypi:~ $

```

7. ทดลองใหม่อีกครั้ง โดย MQTT Lens

<p>Connection Details</p> <p>Connection name <input type="text" value="XXX"/></p> <p>Connection color scheme <input type="color" value="#00FF00"/></p> <p>Hostname <input type="text" value="tcp://"/> <input type="text" value="192.168.1.111"/></p> <p>Port <input type="text" value="1883"/></p> <p>Client ID <input type="text" value="lens_1Twp6xeqDnWAOqgE433wB44eE9s"/> <input type="button" value="Generate a random ID"/></p> <p>Session <input checked="" type="checkbox"/> Clean Session</p> <p>Automatic Connection <input checked="" type="checkbox"/> Automatic Connection</p> <p>Keep Alive <input type="text" value="120"/> seconds</p> <p>Credentials</p> <p>Username <input type="text" value="mymqtt"/></p> <p>Password <input type="password" value="*****"/></p> <p>Last-Will <input type="text"/></p> <p><input type="button" value="CANCEL"/> <input type="button" value="SAVE CHANGES"/></p>	<p>1. name = อะไรก็ได้</p> <p>2. RPi IP,Port = 1883</p> <p>3. UName = mymqtt, Pass=raspi</p>																				
<p>Connection: XXX</p> <p>Subscribe <input type="text" value="mynew/test"/> <input type="button" value="SUBSCRIBE"/></p> <p>Publish <input type="text" value="mynew/test"/> <input type="button" value="PUBLISH"/></p> <p>Message <input type="text" value="Haha"/></p> <p>Subscriptions Topic: "mynew/test" Showing the last 5 messages — +</p> <table border="1"> <thead> <tr> <th>#</th> <th>Time</th> <th>Topic</th> <th>QoS</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>11:43:52</td> <td>mynew/test</td> <td>0</td> </tr> <tr> <td colspan="4">Message: Haha</td> </tr> <tr> <td>1</td> <td>11:45:04</td> <td>mynew/test</td> <td>0</td> </tr> <tr> <td colspan="4">Message: SUT12345678</td> </tr> </tbody> </table>	#	Time	Topic	QoS	0	11:43:52	mynew/test	0	Message: Haha				1	11:45:04	mynew/test	0	Message: SUT12345678				<p>4. Topic = mynew/test, and Subscribe</p> <p>5. Topic = mynew/test, Message = "Haha" and Publish</p> <p>8. Show "SUT12345678"</p>
#	Time	Topic	QoS																		
0	11:43:52	mynew/test	0																		
Message: Haha																					
1	11:45:04	mynew/test	0																		
Message: SUT12345678																					

```

pi@raspberrypi: ~
pi@raspberrypi:~$ mosquitto_sub -d -u mymqtt -P myrasp -t mynew/test
Client mosqsub|926-raspberrypi sending CONNECT
Client mosqsub|926-raspberrypi received CONNACK
Client mosqsub|926-raspberrypi sending SUBSCRIBE (Mid: 1, Topic: mynew/test, QoS
: 0)
Client mosqsub|926-raspberrypi received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|926-raspberrypi received PUBLISH (d0, q0, r0, m0, 'mynew/test', .
.. (8 bytes))
Test1234
Client mosqsub|926-raspberrypi sending PINGREQ
Client mosqsub|926-raspberrypi received PINGRESP
Client mosqsub|926-raspberrypi sending PINGREQ
Client mosqsub|926-raspberrypi received PINGRESP
Client mosqsub|926-raspberrypi received PUBLISH (d0, q0, r0, m0, 'mynew/test', .
.. (4 bytes))
Haha
Client mosqsub|926-raspberrypi sending PINGREQ
Client mosqsub|926-raspberrypi received PINGRESP

```

6. Show "Haha"

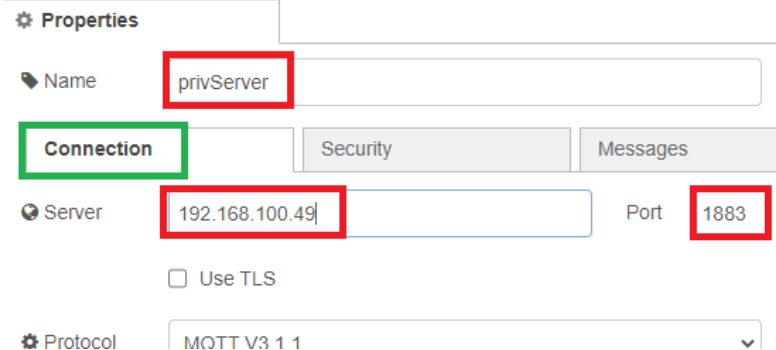
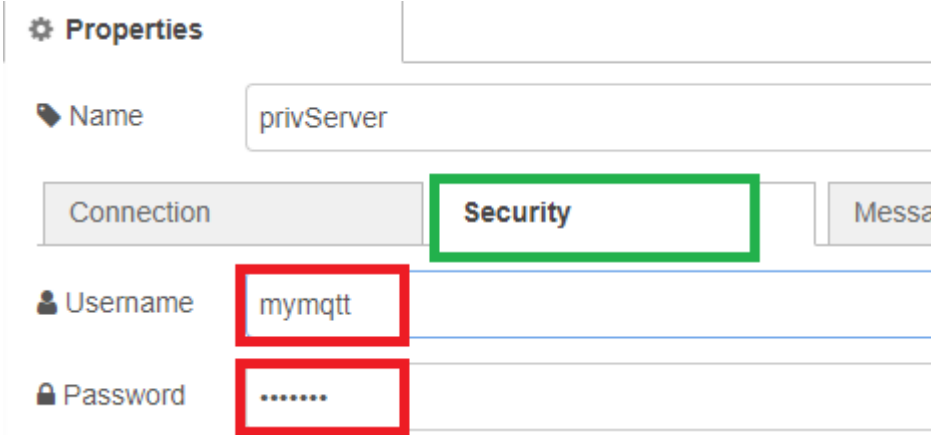
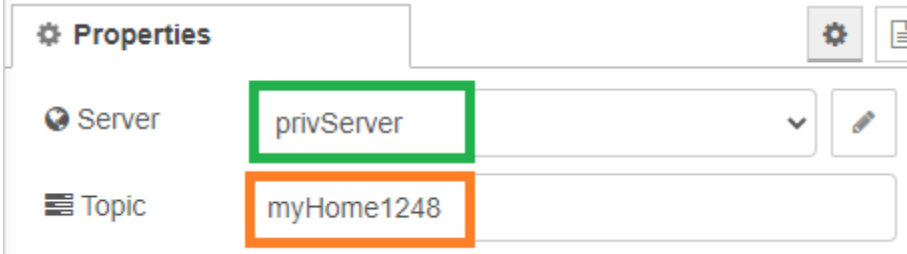
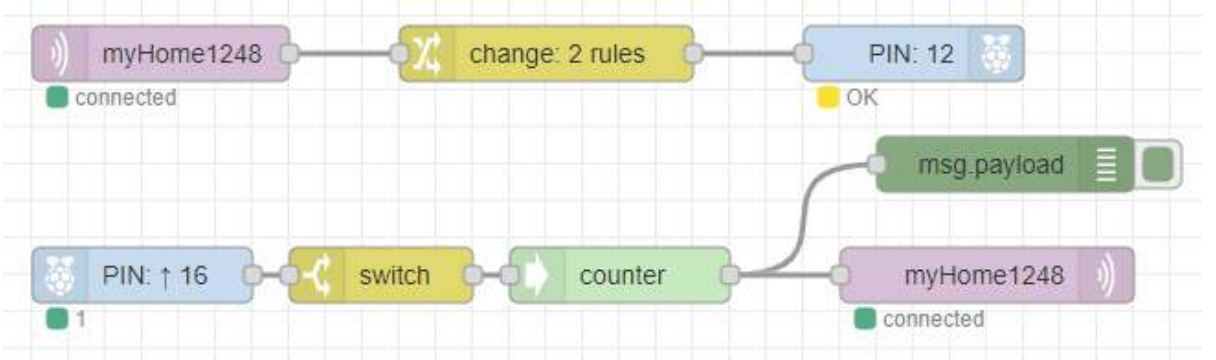
```

pi@raspberrypi: ~
pi@raspberrypi:~$ mosquitto_pub -d -u mymqtt -P myrasp -t mynew/test -m "Test12
34"
Client mosqpub|919-raspberrypi sending CONNECT
Client mosqpub|919-raspberrypi received CONNACK
Client mosqpub|919-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'mynew/test', ..
. (8 bytes))
Client mosqpub|919-raspberrypi sending DISCONNECT
pi@raspberrypi:~$ mosquitto_pub -d -u mymqtt -P myrasp -t mynew/test -m "Test12
34"
Client mosqpub|927-raspberrypi sending CONNECT
Client mosqpub|927-raspberrypi received CONNACK
Client mosqpub|927-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'mynew/test', ..
. (8 bytes))
Client mosqpub|927-raspberrypi sending DISCONNECT
pi@raspberrypi:~$ mosquitto_pub -d -u mymqtt -P myrasp -t mynew/test -m "SUT123
45678"
Client mosqpub|936-raspberrypi sending CONNECT
Client mosqpub|936-raspberrypi received CONNACK
Client mosqpub|936-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'mynew/test', ..
. (11 bytes))
Client mosqpub|936-raspberrypi sending DISCONNECT
pi@raspberrypi:~$

```

7. Publish "SUT12345678"

8. ทดสอบด้วย NodeRED

	<p>Add New Server Connection</p> <ul style="list-style-type: none"> • IP • Port
	<p>Set Security</p> <ul style="list-style-type: none"> • Name • Password
	<p>Select Server Select Topic</p>
	

Subscribe

0 - at most once

SUBSCRIBE

Publish

0 - at most once

☐ Retained

PUBLISH

Message

ON

Subscriptions

Topic: "myHome1248" Showing the last 5 messages — +

#

Time

Topic

QoS

0

11:54:20

myHome1248

0

Message: OFF

9. ทดลองด้วย Paho (more data <http://www.steves-internet-guide.com/client-connections-python-mqtt/>)

For testing you can use any MQTT Client. However, if you have Python 2.7 Installed on your machine, you can test it with following sample Python scripts. To execute these Scripts, you must have Paho MQTT Client installed on your machine. You can install it with pip command –

pip install paho-mqtt

```

pi@raspberrypi:~$ pip install paho-mqtt
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading https://files.pythonhosted.org/packages/32/d3/6dcb8fd14746fcde6a556f932b9fedcb85b3a092e0e986372c0e7/paho-mqtt-1.5.1.tar.gz (101kB)
    100% |#####| 102kB 779kB/s
Building wheels for collected packages: paho-mqtt
  Running setup.py bdist_wheel for paho-mqtt ... done
  Stored in directory: /home/pi/.cache/pip/wheels/75/e2/f5/78942b19b4d135605e58dfe85fb4d636aabf76904b
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.5.1
pi@raspberrypi:~$

```

Once Paho Client Library is installed, you can download and execute following Python scripts (Don't forget to change "MQTT_BROKER" IP Address) –

[Publisher.py](#)

```
MQTT_BROKER = "192.168.100.49"
MQTT_PORT = 1883
MQTT_KEEPALIVE_INTERVAL = 45
MQTT_TOPIC = "testTopic"
MQTT_MSG = "Hello MQTT"
MQTT_User = "mymqtt"
MQTT_Pass = "myraspi"

import paho.mqtt.client as mqtt
import time

# Define on_connect event Handler
def on_connect(mosq, obj, rc):
    print "Connected to MQTT Broker"

# Define on_publish event Handler
def on_publish(client, userdata, mid):
    print "Message Published..."

# Initiate MQTT Client
mqttc = mqtt.Client()

# Register Event Handlers
mqttc.on_publish = on_publish
mqttc.on_connect = on_connect

# Using Authentication
mqttc.username_pw_set(username=MQTT_User,password=MQTT_Pass)

# Connect with MQTT Broker
mqttc.connect(MQTT_BROKER, MQTT_PORT, MQTT_KEEPALIVE_INTERVAL)

# Publish message to MQTT Topic
for i in range(5):
    sendData = MQTT_MSG + " --> " + str(i)
    print 'sendData = ',sendData
    mqttc.publish(MQTT_TOPIC,sendData)
    time.sleep(10)

# Disconnect from MQTT_Broker
mqttc.disconnect()
```

The image displays a Raspberry Pi MQTT server setup. The top half shows a Geany IDE window titled '*Publisher.py - /home/pi - Geany'. The code defines MQTT broker details and a connection handler.

```
1 MQTT_BROKER = "192.168.100.49"
2 MQTT_PORT = 1883
3 MQTT_KEEPALIVE_INTERVAL = 45
4 MQTT_TOPIC = "testTopic"
5 MQTT_MSG = "Hello MQTT"
6 MQTT_User = "mymqtt"
7 MQTT_Pass = "myraspi"
8
9
10 import paho.mqtt.client as mqtt
11 import time
12
13 # Define on_connect event Handler
14 def on_connect(mosq, obj, rc):
15     print "Connected to MQTT Broker"
```

The bottom half shows a MQTT client interface with a green background. It displays the topic "testTopic" and shows the last 1 message. The message content is "Hello MQTT-->4".

Subscriptions

Topic: "testTopic" Showing the last 1 messages — + Messages: 0/18

#	Time	Topic	QoS
17	11:25:03	testTopic	0

Message: Hello MQTT-->4

[Subscriber.py](#)

```

MQTT_BROKER = "192.168.100.49"
MQTT_PORT = 1883
MQTT_KEEPALIVE_INTERVAL = 45
MQTT_TOPIC = "testTopic"
MQTT_User = "mymqtt"
MQTT_Pass = "myraspi"

import paho.mqtt.client as mqtt
import time
# Define on_connect event Handler
def on_connect(self, mosq, obj, rc):
    #Subscribe to a the Topic
    mqttc.subscribe(MQTT_TOPIC, 0)

# Define on_subscribe event Handler
def on_subscribe(mosq, obj, mid, granted_qos):
    print "Subscribed to MQTT Topic"

# Define on_message event Handler
def on_message(mosq, obj, msg):
    print msg.payload

# Initiate MQTT Client
mqttc = mqtt.Client()

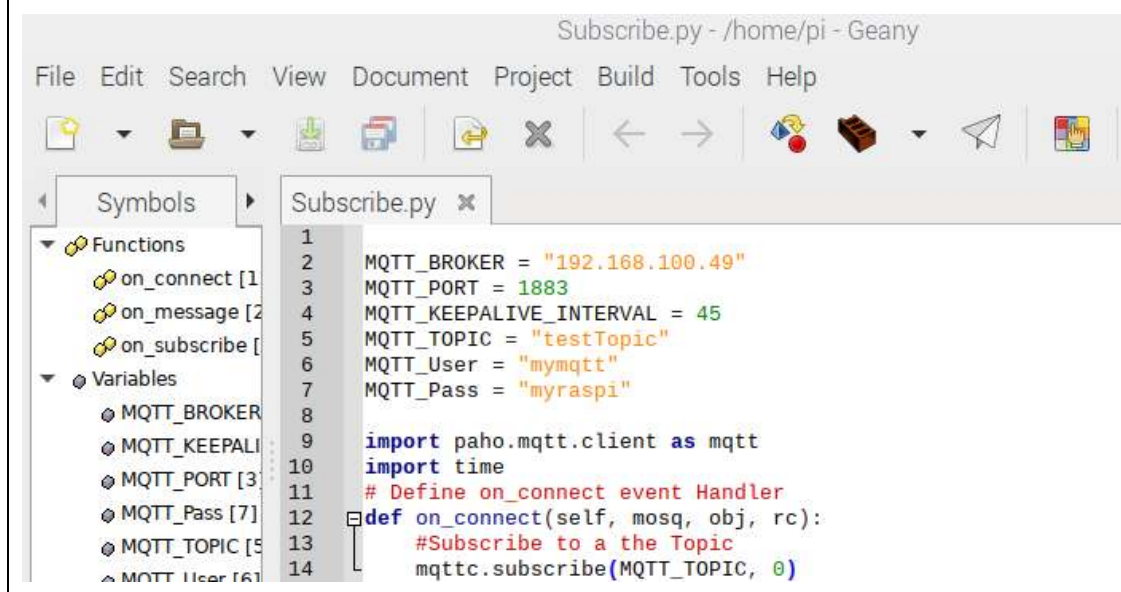
# Register Event Handlers
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe

# Using Authentication
mqttc.username_pw_set(username=MQTT_User,password=MQTT_Pass)

# Connect with MQTT Broker
mqttc.connect(MQTT_BROKER, MQTT_PORT, MQTT_KEEPALIVE_INTERVAL)

# Continue the network loop
mqttc.loop_forever()

```



Publish

0 - at most once ▼

Message

ASDFGH

Subscriptions

Topic: "testTopic" Showing the last 5 messages — +

#	Time	Topic	QoS
0	12:03:19	testTopic	0

Message: ASDFGH

geany_run_script_UKN270.sh ▼

File Edit Tabs Help

```

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to
set a new password.

Subscribed to MQTT Topic
1234
ASDFGH
ASDFGH

```

10. ถอนการติดตั้ง Mosquitto MQTT Broker

To uninstall Mosquitto you can use following command –

```
sudo apt-get purge mosquitto
```

If you want to completely remove Mosquitto with it's associated configuration files, use following command –

```
sudo apt-get --purge remove mosquitto
```


การสร้าง MQTT Server บน Raspberry Pi เพื่อใช้งาน Chatbot LINE ในฟาร์มอัจฉริยะ
Chatbot LINE from Raspberry Pi MQTT Server for Smart Farming

ชื่อ-สกุล :

6/6 - คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz_301 – Blynk with Node-RED

- ทดสอบควบคุม 4 LED ด้วย Raspberry Pi ผ่าน Node-RED
- เพิ่มเติมให้ส่งข้อมูล Temperature ไปแสดงที่ Gauge บน Blynk

Node-RED Code

Node-RED Flow

หน้าจอ Blynk

รูปการทดสอบ 1

รูปการทดสอบ 2

Quiz_302 – MQTT Client with Node-RED

- ทดสอบควบคุม 4 LED ด้วย Raspberry Pi ผ่าน Node-RED
- เพิ่มเติมให้ส่งข้อมูล Temperature ไปแสดงที่ บน Server

Node-RED Code

Node-RED Flow

หน้าจอ MQTT Lens

รูปการทดสอบ 1

รูปการทดสอบ 2

Quiz_303 – MQTT Server on Raspberry Pi

- ทดสอบควบคุม 4 LED ด้วย Raspberry Pi ผ่าน Node-RED ไปยัง private server
- เพิ่มเติมให้ส่งข้อมูล Temperature ไปแสดงที่ บน private Server

โปรแกรมที่ใช้ทดสอบ
รูปถ่ายหน้า Web Browser
รูปการทดสอบ 1
รูปการทดสอบ 2