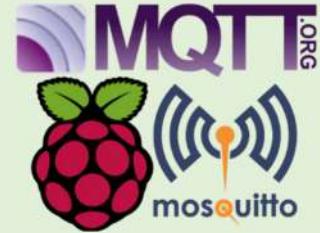


การสร้าง MQTT Server บน Raspberry Pi เพื่อใช้งาน Chatbot LINE ในฟาร์มอัจฉริยะ  
Chatbot LINE from Raspberry Pi MQTT Server for Smart Farming

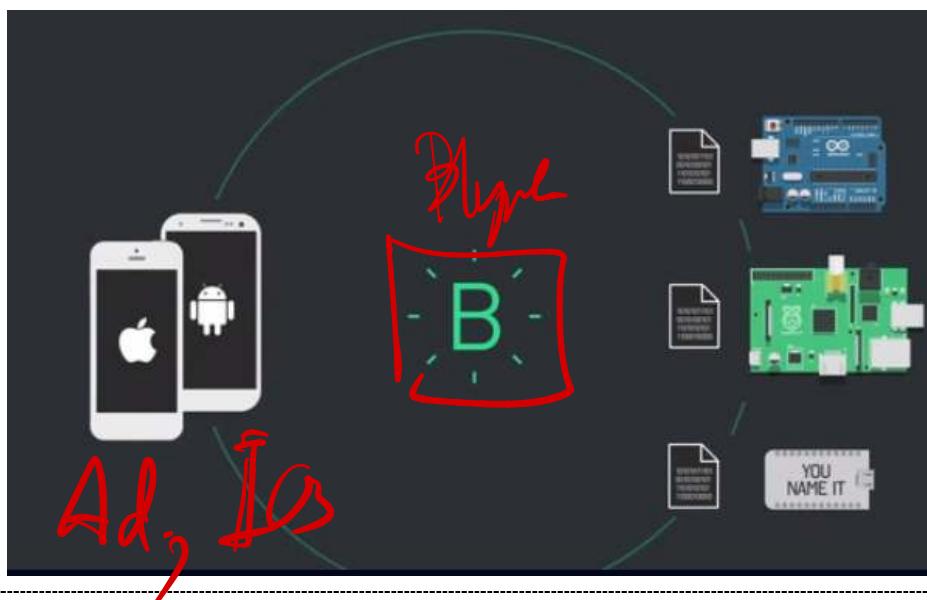
## 3/4 – IoTs with Node-RED on Raspberry Pi

- 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

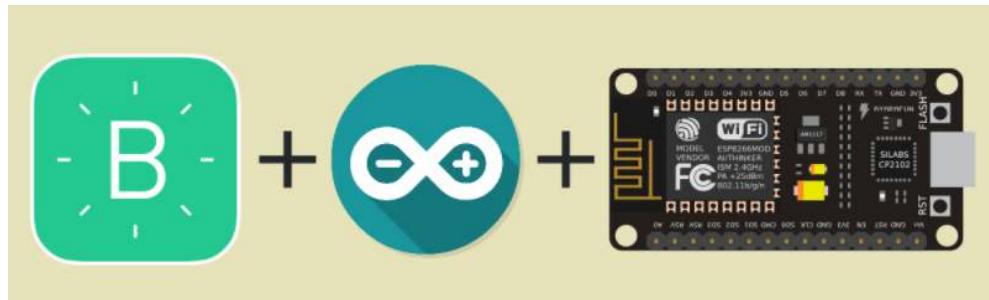


## 1/6 - การใช้งาน ESP32 เพื่อแสดงค่าและควบคุมผ่าน Blynk

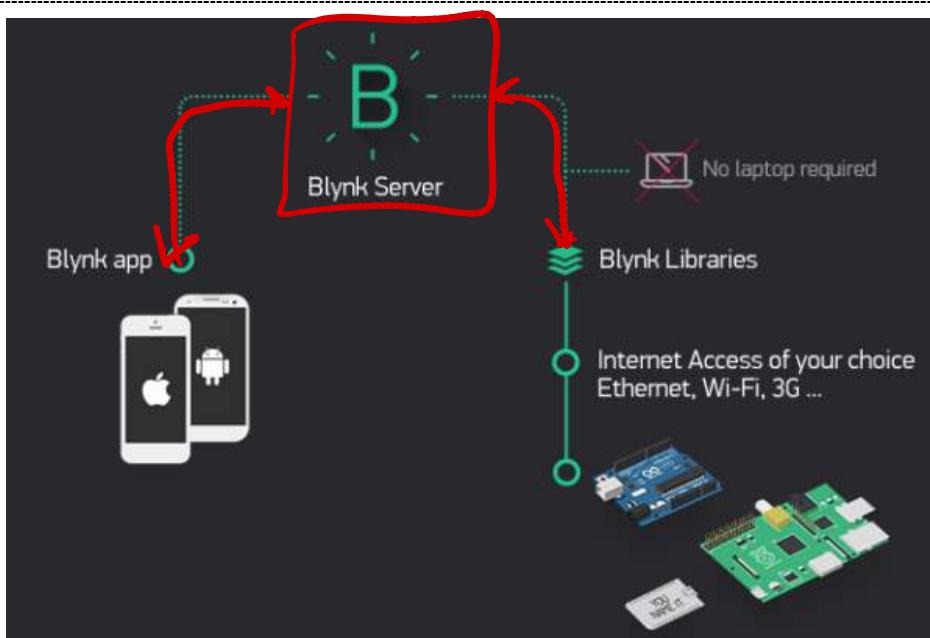
<http://help.blynk.cc/getting-started-library-auth-token-code-examples/blynk-basics/how-to-display-any-sensor-data-in-blynk-app>



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



Blynk สามารถเชื่อมต่ออุปกรณ์ Device ของเราเข้ากับ internet โดยง่ายดาย ไม่ว่าจะเป็น Arduino, ESP8266, Raspberry pi หรือแม้แต่เครื่องที่ร่วมเอา widget ต่างๆ มาควบคุมแทนการเขียน code ยากๆ ไม่เพียงเท่านั้น ทางเลือกในการเชื่อมต่อเข้ากับ Blynk server เราสามารถใช้ได้ทั้ง WiFi และเครือข่ายมือถือ โดยสามารถ Download application นี้ได้ฟรีทั้งระบบ IOS และ Android



#### อ่านเพิ่มเติม

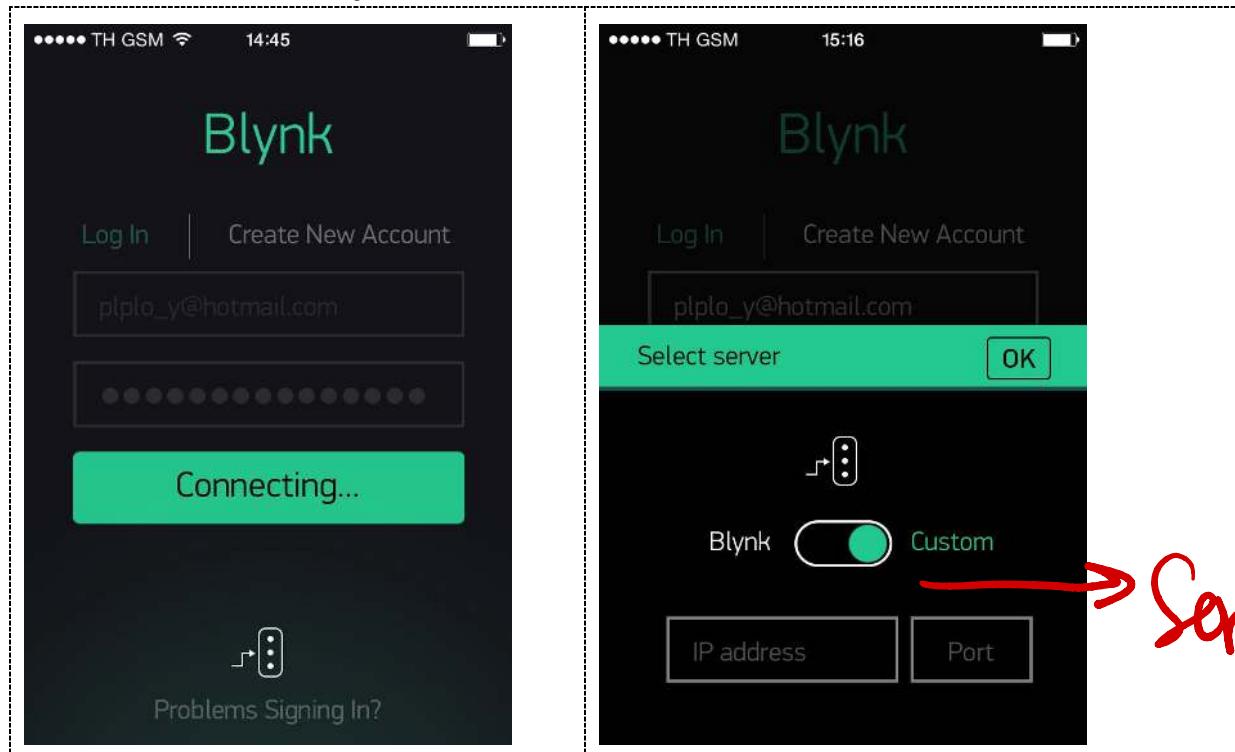
- <https://www.blynk.cc/>, <http://docs.blynk.cc/>
- <http://help.blynk.cc/getting-started-library-auth-token-code-examples/blynk-basics/how-to-display-any-sensor-data-in-blynk-app>
- <https://www.9arduino.com/article/59/app-สำเร็จรูป-blynk-nodemcu-esp8266-ตอนที่-1-blynk-คืออะไร>
- <http://www.ayarafun.com/2015/08/easy-iot-play-with-blynk/>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

### Lab301A – Blynk Control ESP32 Output

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

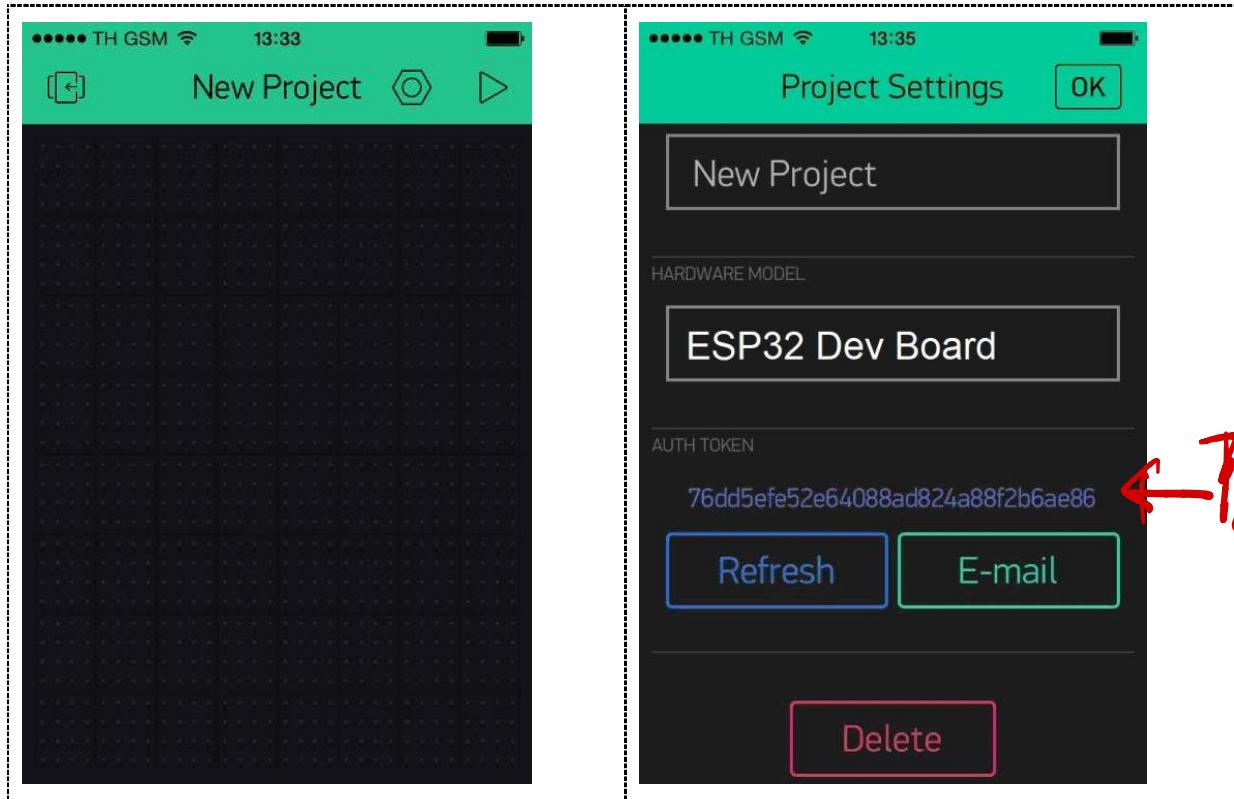


- เริ่มต้นใช้งาน Blynk เราต้องสมัคร ลงทะเบียน เพื่อใช้งานก่อน ให้เลือกที่คำว่า Create New account เพื่อสร้างการเชื่อมต่อกับ application กับ Email ของผู้ใช้งาน
- Login เข้า Blynk
- เลือก Connect เข้ากับ server ของ Blynk ( Blynk ให้ผู้ใช้เลือก Custom Server ได้โดยใส่ IP Address ของ Server เราเอง โดยกดที่รูป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

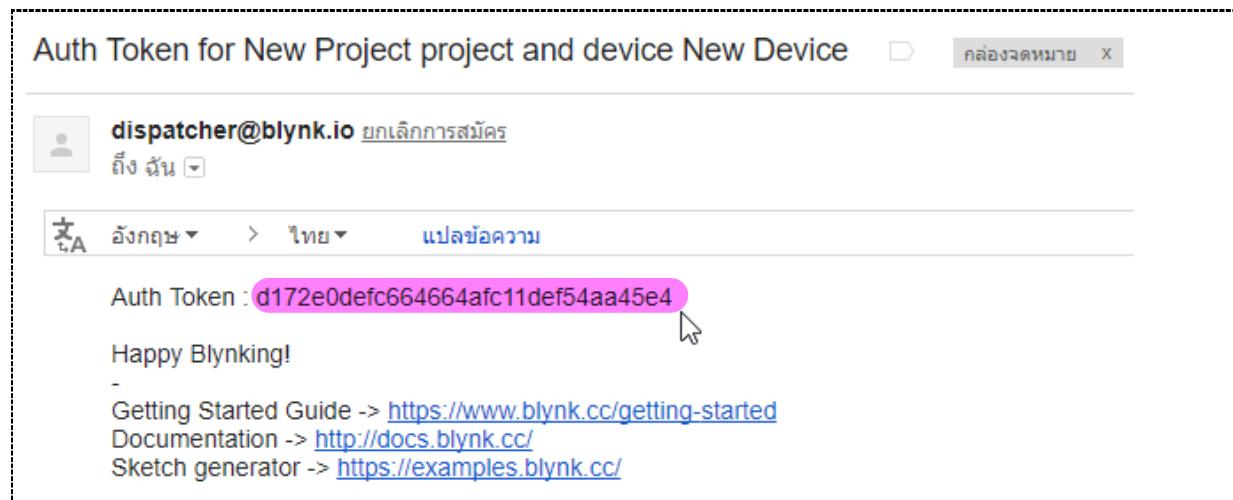


⌚ Blynk

6. ต่อมาเป็นการสร้าง Project ของเราร่วมกับ Blynk ให้กดที่สัญลักษณ์หกเหลี่ยมมุมขวาบนเพื่อตั้งค่า
7. โดยในหน้านี้เราจะสามารถตั้งค่า Project ของเราและเลือกรูปแบบ Hardware ที่เราจะใช้ได้ Hardware ให้เลือก ESP32 Dev Board



8. โดยทุกครั้งที่เริ่มสร้างโปรเจคใหม่ AUTH TOKEN จะถูกเปลี่ยนใหม่เสมอ ซึ่ง KEY นี้เองที่เป็นเสมือนกุญแจสำหรับเขื่อมต่อ โดยที่เราไม่ต้องใช้ user, password เราสามารถติดต่อผ่าน "E-mail" เพื่อส่ง KEY นี้เข้าเมล เราได้



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

```
ESP32_WiFi$ #define BLYNK_PRINT Serial

1 #define BLYNK_PRINT Serial
2
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 }

#define BLYNK_PRINT Serial

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

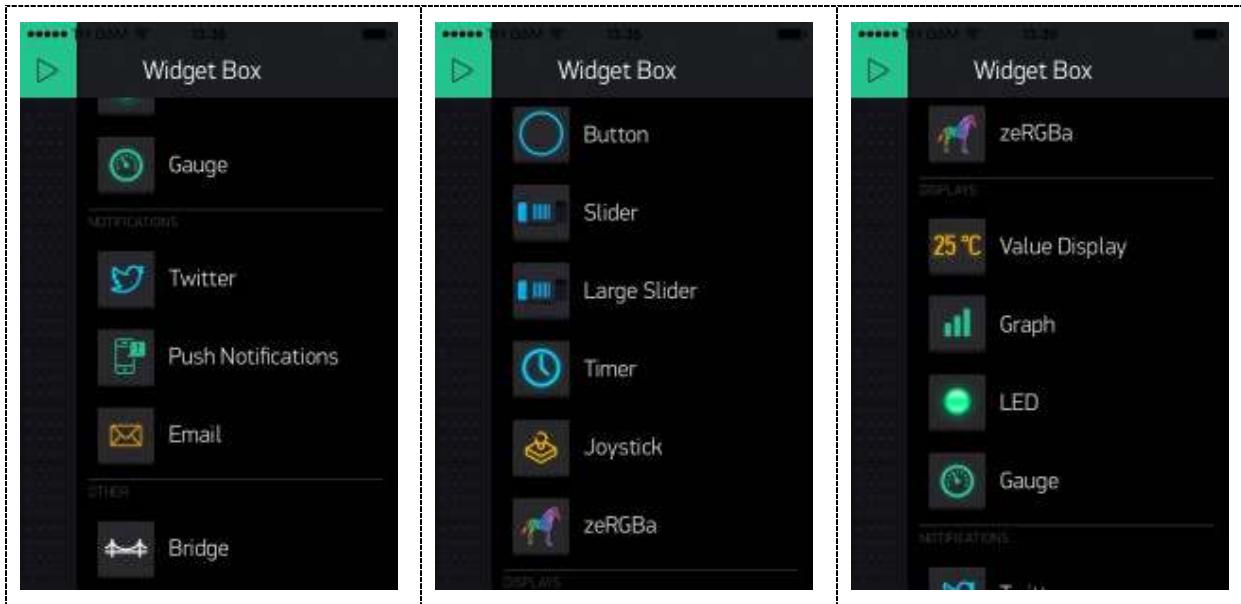
char auth[] = "YourAuthToken";
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

void setup()
{ Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
}

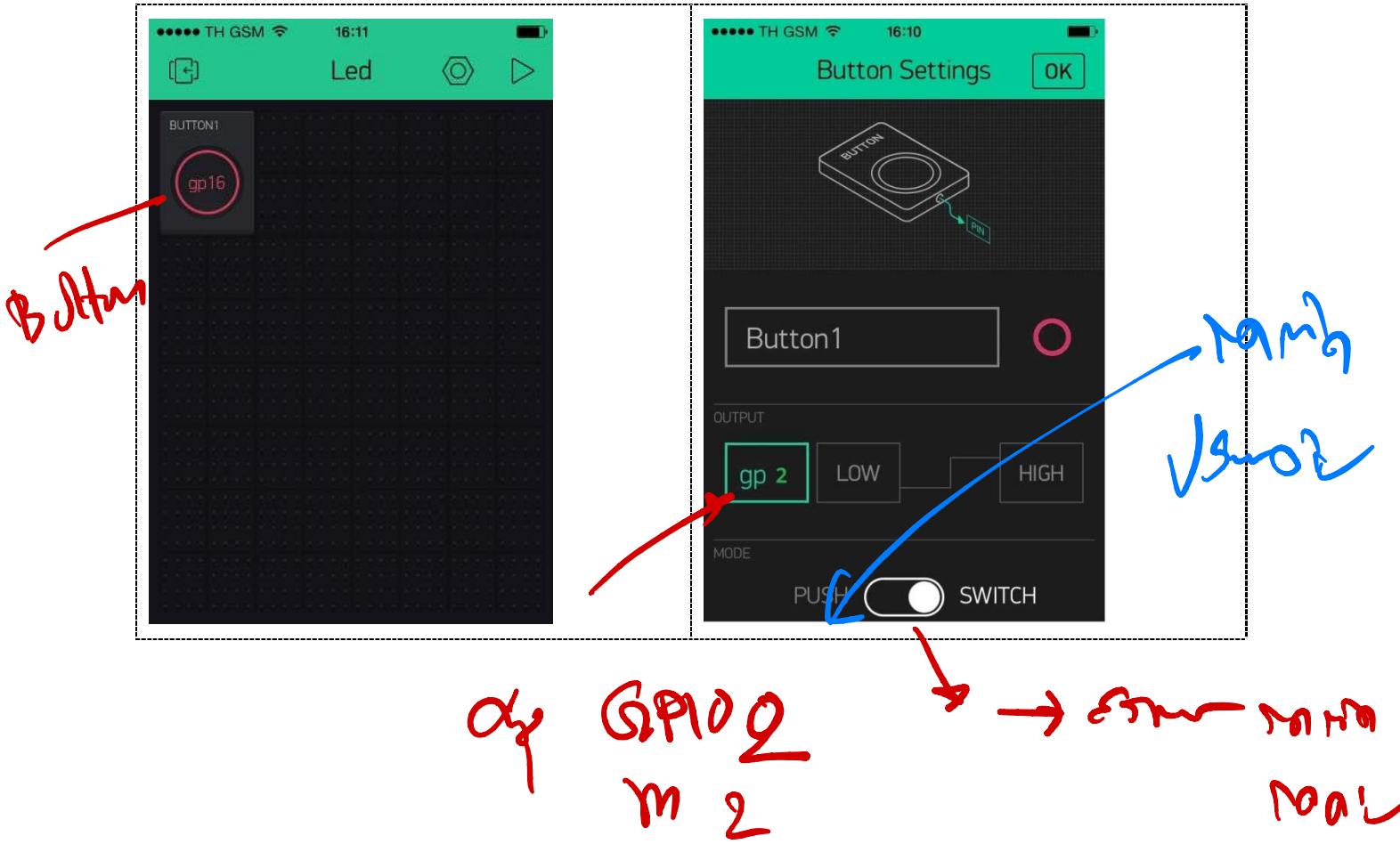
void loop()
{ Blynk.run();
}
```

10. จากนั้นให้ทำการ Upload ลง board และเปิด Serial Monitor จะกระทิ้งเมื่อความเร็วแบบนี้

11. สร้างโปรเจคของเราน Blynk โดยการกดที่พื้นที่ว่างเปล่าตรงไหนก็ได้ในหน้า New project จะปรากฏหน้าต่างของ Widget ให้เราเลือกขึ้นมา

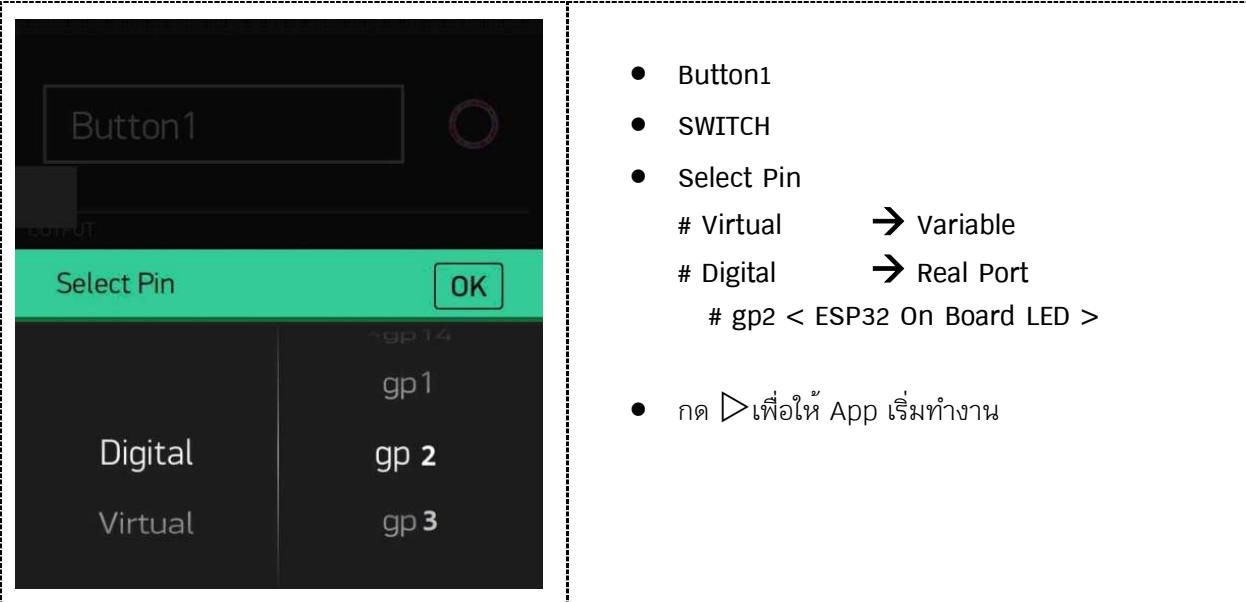


12. ໃນบทความรอบนี้เราจะลองให้ดูว่าไรที่ง่ายๆ ก่อน ให้ลองเลือก Button widget มาลงบนพื้นที่ว่างเปล่ามา 1 อัน จากนั้นเราจะตั้งค่าการใช้งานปุ่ม Button กันโดยกดไปที่รูป Button ที่เราเลือกจะปรากฏหน้าต่างแบบนี้ ซึ่งในหน้านี้เราจะสามารถเปลี่ยนชื่อปุ่มได้ และเลือกโหมด output pin ที่ต่อ กับ อุปกรณ์จาก board ของเราได้

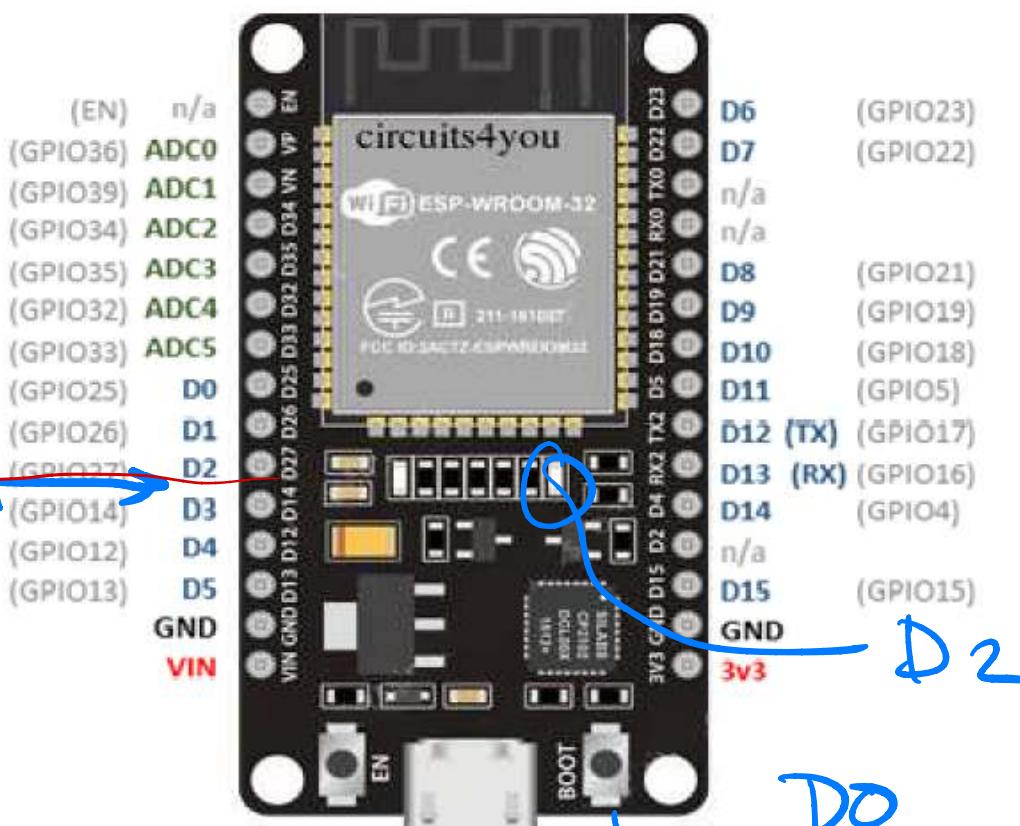


13. เลือกรูปแบบ pin จะให้เป็นขา Digital หรือ Virtual ก็ได้ ซึ่งรูปแบบ Virtual จะไม่ใช่การรับค่าจากขาต่างๆ เป็นเหมือนการสร้างตัวแปรมาเก็บค่าอีกที และเลือกขา GPIO ให้ตรงกับ อุปกรณ์ที่เราจะต่อ เมื่อเสร็จแล้วกด ▶ เพื่อให้ App เริ่มทำงาน

14. ทดสอบการทำงานโดยการคุม LED GPIO2 หรือ On Board DOIT ESP32 Kit Ver1



ESP32 PINOUT



## Lab301B – ESP-32 Control and Monitor by Blink

15. การทดสอบอ่านสวิตช์ DO และแสดงผลที่ 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>
- ทดลองโปรแกรม ให้แก่ไข (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";

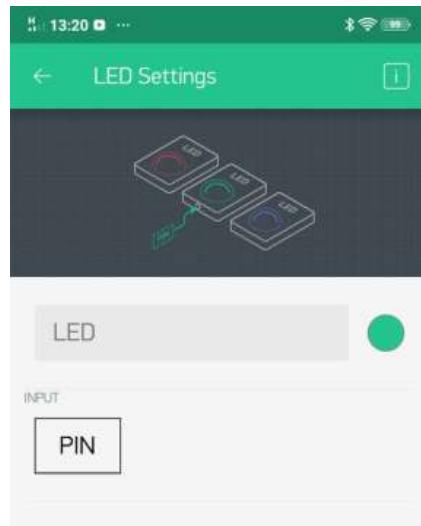
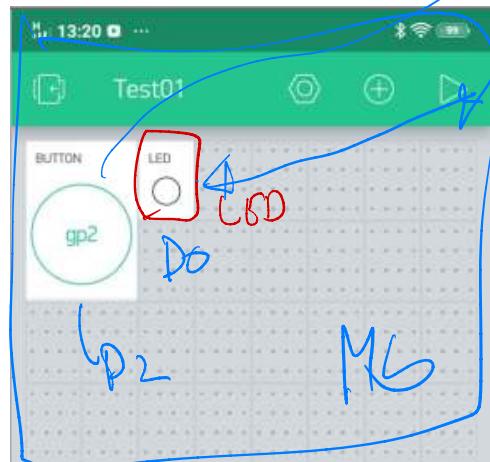
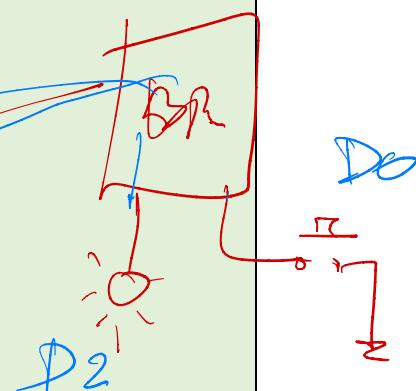
void myTimerEvent()
{
    boolean isPressed = (digitalRead(btnPin) == LOW);
    if (isPressed != btnState)
    {
        if (isPressed)
            blynk_LED.on();
        else
            blynk_LED.off();
        btnState = isPressed;
        Serial.print(" LED Status = ");
        Serial.println(btnState);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

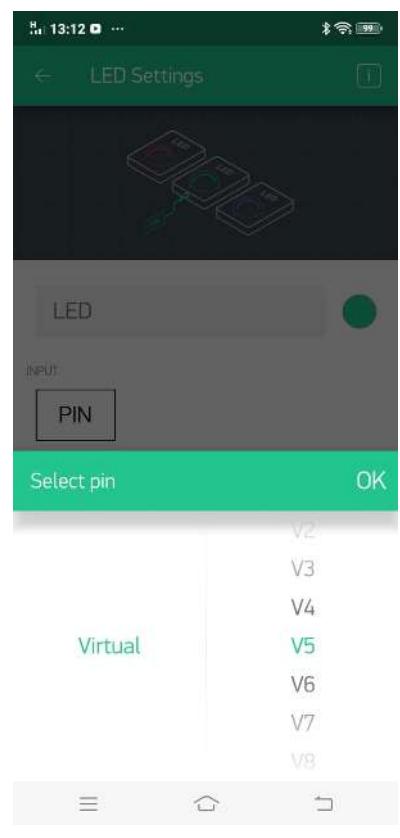
void setup()
{
    Serial.begin(115200);
    pinMode(btnPin, INPUT_PULLUP);
    Blynk.begin(auth, ssid, pass);
    timer.setInterval(250L, myTimerEvent);
}

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

DO

Blynk





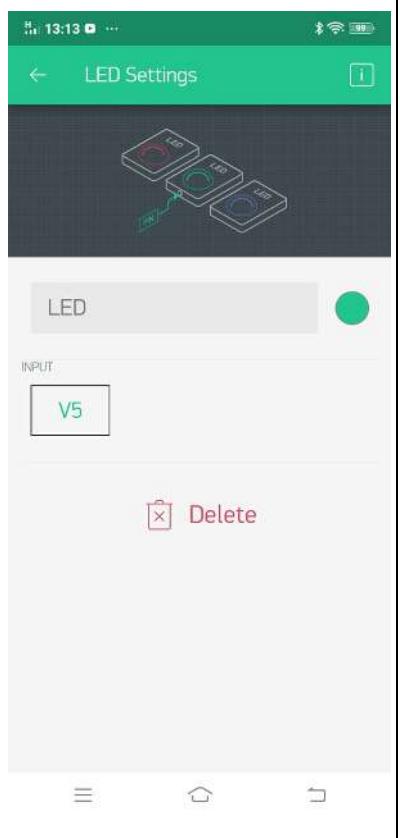
LED Settings

LED

INPUT: PIN

Select pin OK

V2
V3
V4
<b>V5</b>
V6
V7
V8



LED Settings

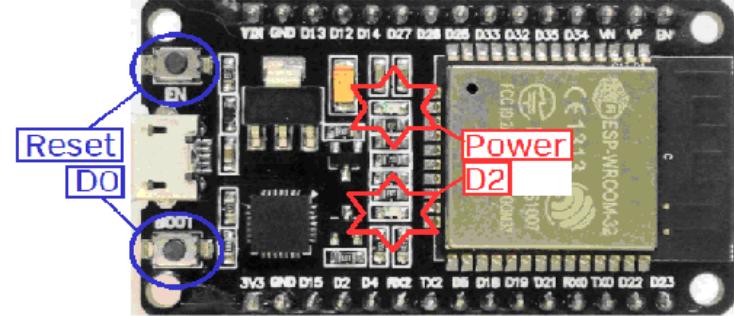
LED

INPUT: V5

Delete

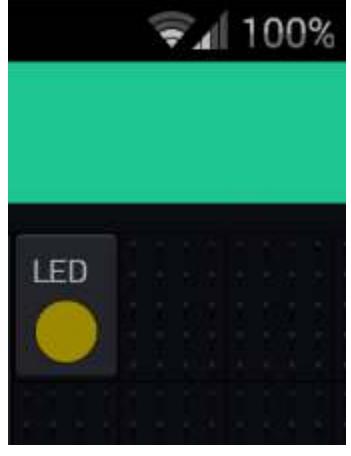
```
/ _ ) / / _ _ / / _ / 
/ _ / / / / _ \ \ ' / 
/ _ / / \ _ , / / / _ / \ 
/ _ / v0.6.1 on ESP32
```

[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).



Reset D0

Power D2



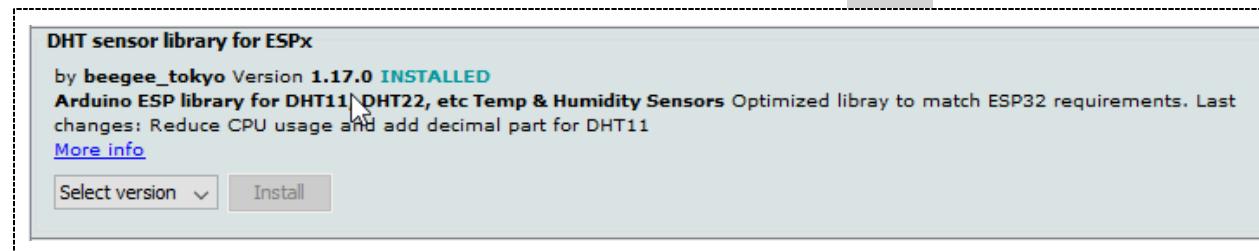
100%

LED

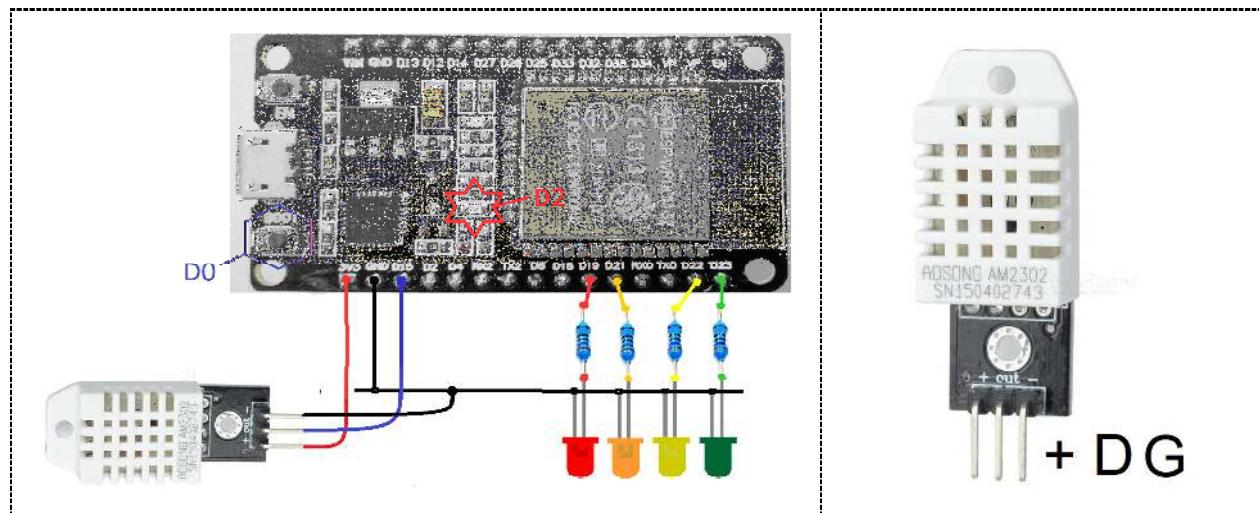
## Lab301c – Blynk LED Control and Sensor Monitor

17. การทดสอบอ่านอุณหภูมิด้วย DHT-22 และแสดงผลที่ 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** และทำการติดตั้ง



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 ให้แก่ไข (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;

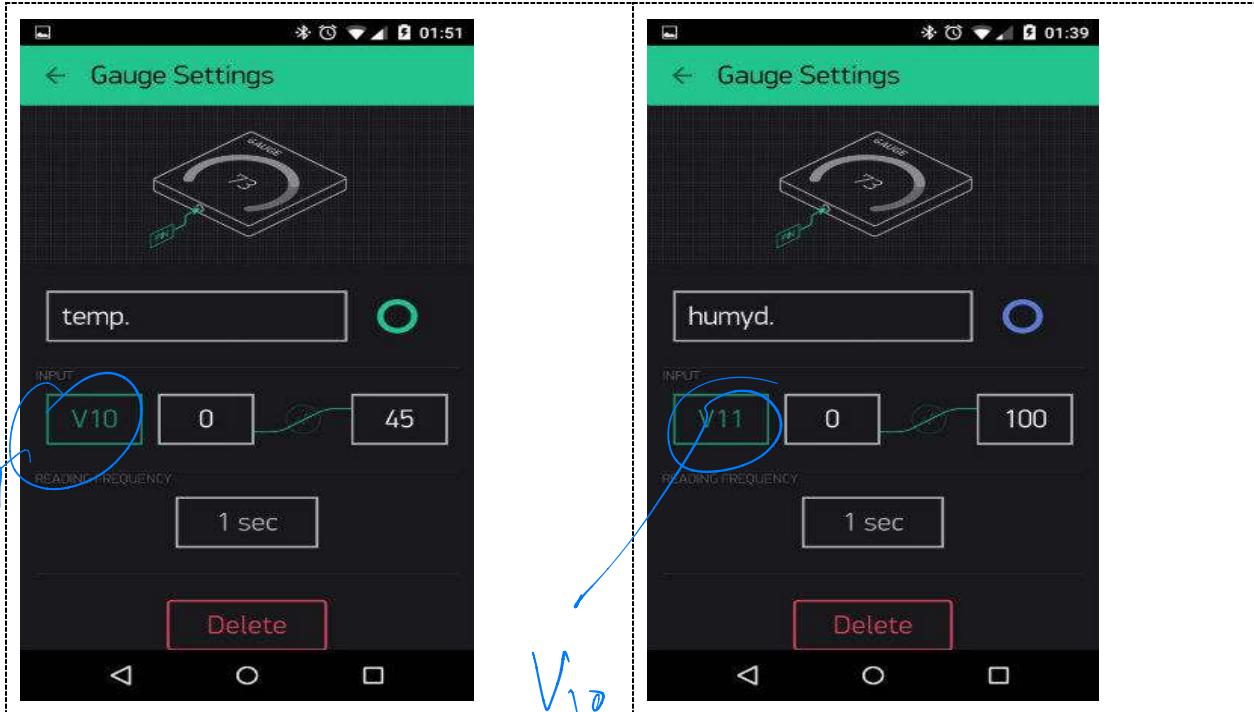
void myTimerEvent()
{
    temperature = dht.getTemperature();
    humidity = dht.getHumidity();
    Blynk.virtualWrite(V10, temperature);
    Blynk.virtualWrite(V11, humidity);
    Serial.print("Temp(°C) > ");
    Serial.print(temperature, 1);
    Serial.print(", Humidity(%) > ");
    Serial.println(humidity,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

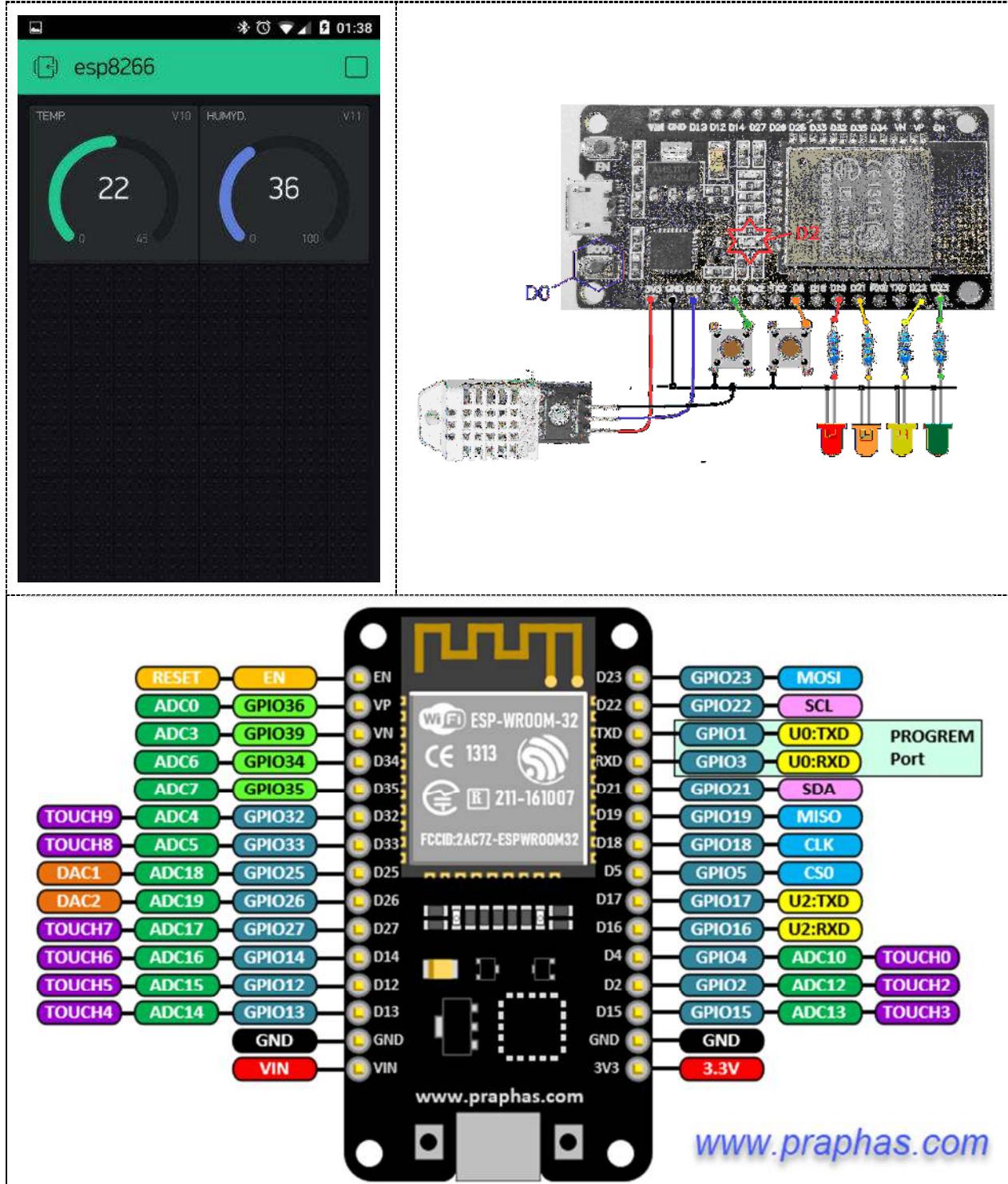
void setup()
{
    Serial.begin(115200);
    dht.setup(pinDHT_22, DHTesp::DHT22);
    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, myTimerEvent);
}

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

Virtual Port V10  
V11

20. ที่ Blynk ให้ใช 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
```

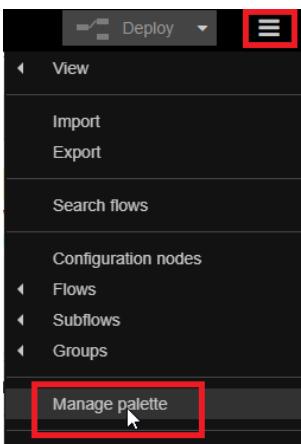
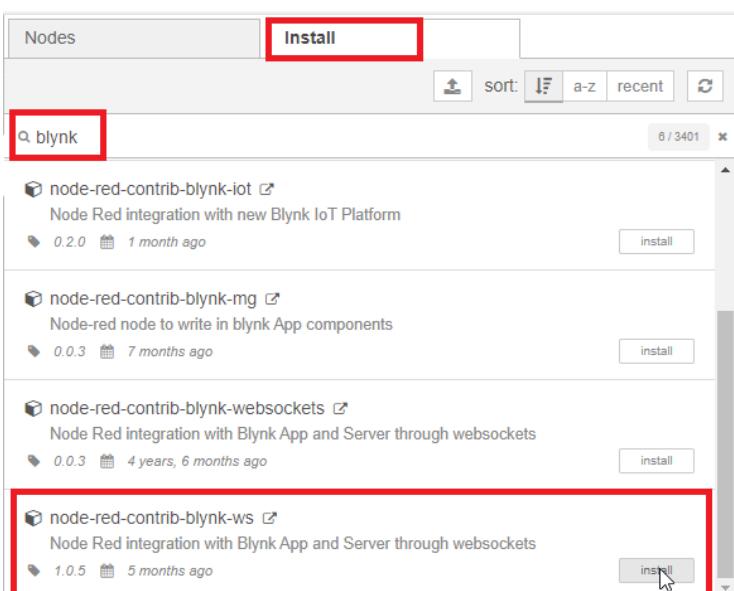
```
npm -v
```

```
nodejs -v
```

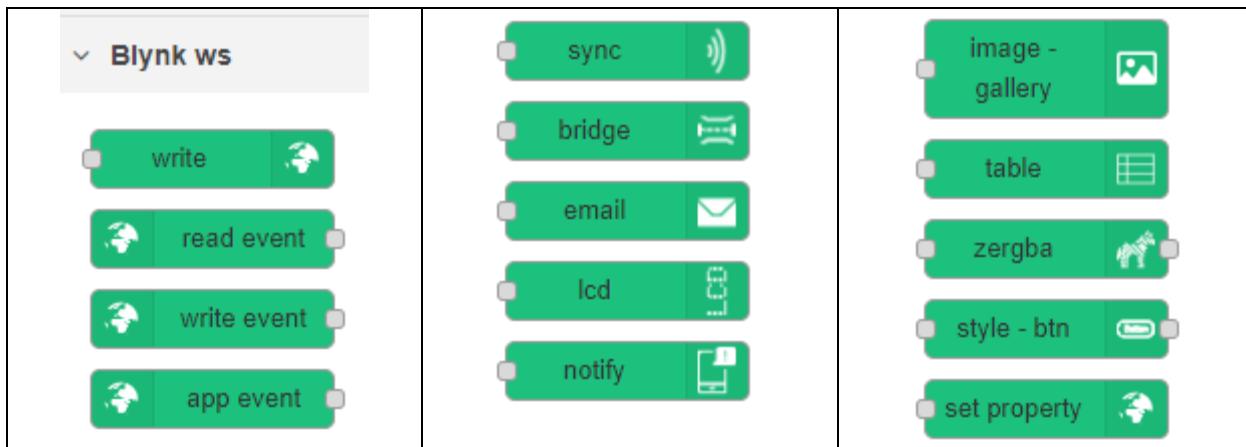
```
found 0 vulnerabilities
successfully installed npm@latest
pi@raspberrypi:~ $ npm -v
7.20.6
pi@raspberrypi:~ $ nodejs -v
v10.24.0
```

```
pi@raspberrypi:~ $ npm -v
7.20.5
pi@raspberrypi:~ $ nodejs -v
v10.24.0
```

## 2. เพิ่มหนด Blynk ให้กับ Node-RED

 <p>The screenshot shows the Node-RED interface with the sidebar open. The 'Manage palette' option under 'Configuration nodes' is highlighted with a red box.</p>	Manage Palette
 <p>The screenshot shows the 'Manage Palette' search results for 'blynk'. The search bar contains 'blynk', and the results list several nodes. One result, 'node-red-contrib-blynk-ws', is highlighted with a red box and its 'install' button is also highlighted.</p>	<p>Install blynk  node-red-contrib-blynk-ws install confirm</p>

## 3. จะได้หน้า Blynk WebSocket เพิ่มเข้ามา



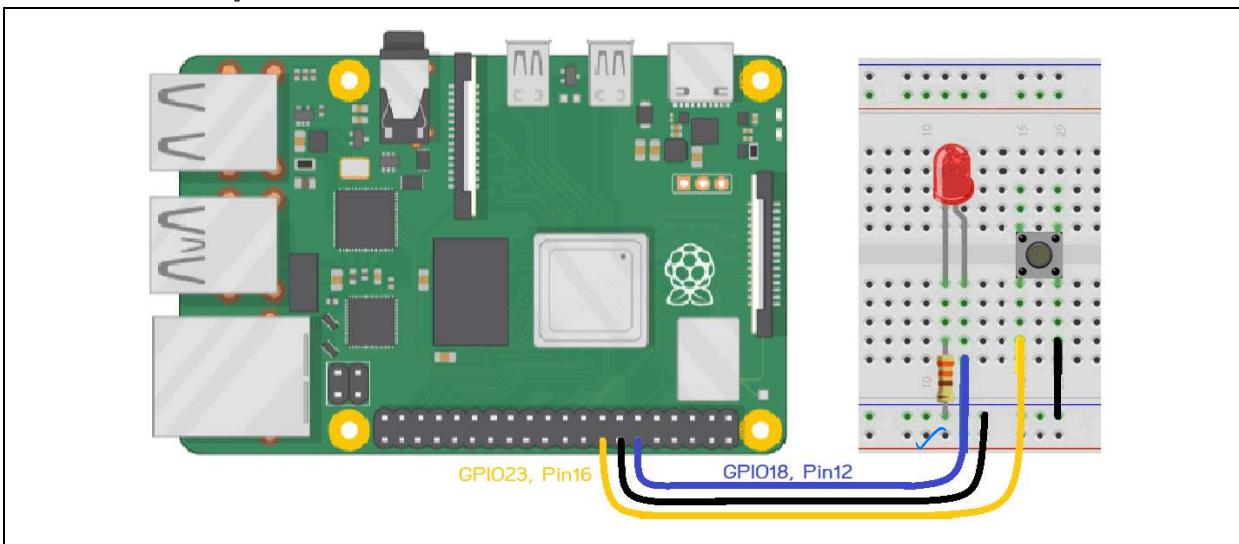
## 4. Create New Project ที่ Blynk

<p>New Device</p> <p>HARDWARE MODEL: Generic Board</p> <p>CONNECTION TYPE: Wi-Fi</p> <p>AUTH TOKEN: ****buL7</p> <p>Refresh Email</p>	<p>Button Settings</p> <p>Button</p> <p>OUTPUT: V1</p> <p>MODE: PUSH SWITCH</p> <p>ON/OFF LABELS: OFF ON</p> <p>DESIGN: FONT SIZE T T T TEXT</p>	<p>test RPi</p> <p>BUTTON V1</p>
Generic Board	V1, Switch Mode	Button = V1

## 5. สร้าง Node Flow และ Configure

	Write event Gpio-out																		
<p>Properties</p> <p>Connection: Add new blynk-ws-client... <input style="border: 1px solid red; padding: 2px 10px; border-radius: 5px;" type="button" value="..."/></p> <p>Virtual Pin: <input type="checkbox"/> All Pins</p>	Add new blynk <i>Blynk</i> <i>Add. Serv</i>																		
<p>Connection</p> <p>Url: wss://blynk-cloud.com/websockets</p> <p>Auth Token: CQwbMwHyedfb08s28DmHMaKjr2IDbuL7</p> <p>Proxy: No proxy</p>	wss://blynk-cloud.com/websockets Token Key																		
<p>Properties</p> <p>Connection: Blynk device-DbuL7</p> <p>Virtual Pin: <input type="radio"/> All Pins</p> <p>Virtual Pin: <input checked="" type="radio"/> 1</p> <p>Name: Name</p>	Set virtual pin																		
<p>● Pin</p> <table border="1"> <tbody> <tr><td>3.3V Power - 1</td><td><input type="radio"/></td><td>2 - 5V Power</td></tr> <tr><td>SDA1 - GPIO02 - 3</td><td><input type="radio"/></td><td>4 - 5V Power</td></tr> <tr><td>SCL1 - GPIO03 - 5</td><td><input type="radio"/></td><td>6 - Ground</td></tr> <tr><td>GPIO04 - 7</td><td><input type="radio"/></td><td>8 - GPIO14 - TxD</td></tr> <tr><td>Ground - 9</td><td><input type="radio"/></td><td>10 - GPIO15 - RxD</td></tr> <tr><td>GPIO17 - 11</td><td><input type="radio"/></td><td>12 - GPIO18</td></tr> </tbody> </table>	3.3V Power - 1	<input type="radio"/>	2 - 5V Power	SDA1 - GPIO02 - 3	<input type="radio"/>	4 - 5V Power	SCL1 - GPIO03 - 5	<input type="radio"/>	6 - Ground	GPIO04 - 7	<input type="radio"/>	8 - GPIO14 - TxD	Ground - 9	<input type="radio"/>	10 - GPIO15 - RxD	GPIO17 - 11	<input type="radio"/>	12 - GPIO18	Pin12 - GPIO18
3.3V Power - 1	<input type="radio"/>	2 - 5V Power																	
SDA1 - GPIO02 - 3	<input type="radio"/>	4 - 5V Power																	
SCL1 - GPIO03 - 5	<input type="radio"/>	6 - Ground																	
GPIO04 - 7	<input type="radio"/>	8 - GPIO14 - TxD																	
Ground - 9	<input type="radio"/>	10 - GPIO15 - RxD																	
GPIO17 - 11	<input type="radio"/>	12 - GPIO18																	
	Deploy																		

## 6. ทดสอบตามรูป LED at Board.Pin12



## 7. เพิ่มเติม Node Flow และ Configure

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

LED on → 1  
off → 0

<p>Connection: Blynk device-DbuL7</p> <p>Pin Mode: Fixed</p> <p>Virtual Pin: 2</p>	Set virtual pin 2
<p>Connection: Blynk device-DbuL7</p> <p>Pin Mode: Fixed</p> <p>Virtual Pin: 3</p>	Set virtual pin 3
<pre> graph LR     PIN16[PIN: ↑ 16] --&gt; P1[Pin V1 - Write Event]     P1 --&gt; C[change: 2 rules]     C --&gt; P3_1[Pin V3 - Write]     C --&gt; P3_2[Pin V2 - Write]     P3_1 --&gt; V3_1[PIN: 12]     P3_2 --&gt; V3_2[PIN: 12]   </pre>	

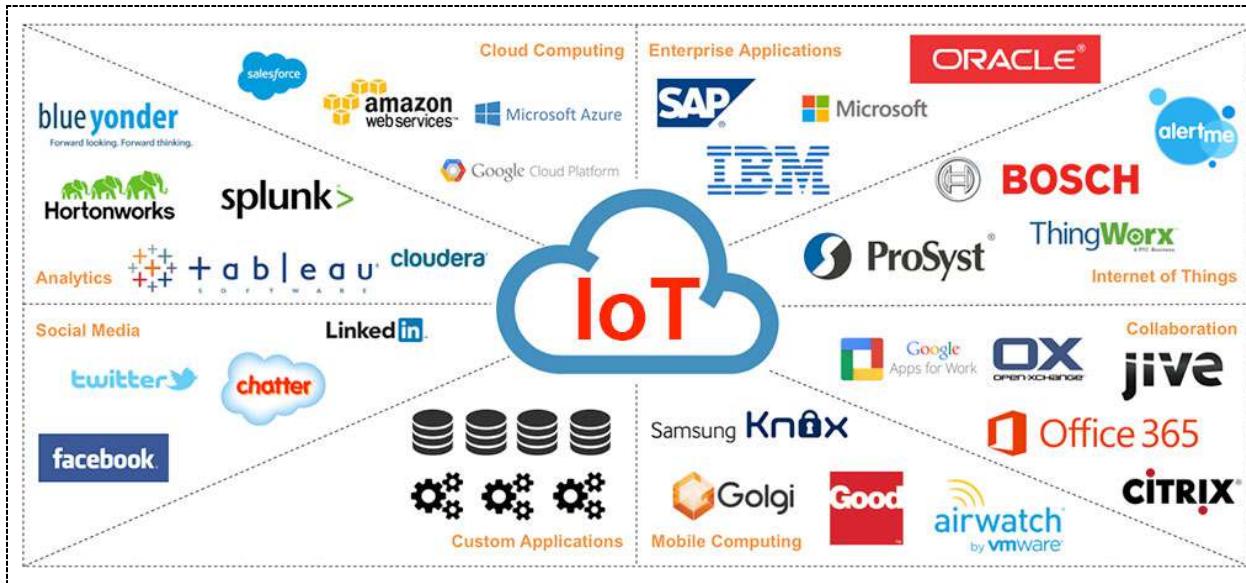
## 8. เพิ่มเติม Widgets และตั้งค่าใน Blynk

<p>Button = V1 LED = V3 &lt; LED on = 255, Off = 0 &gt; Label V-Setting = V2, set range 0, 1</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/>

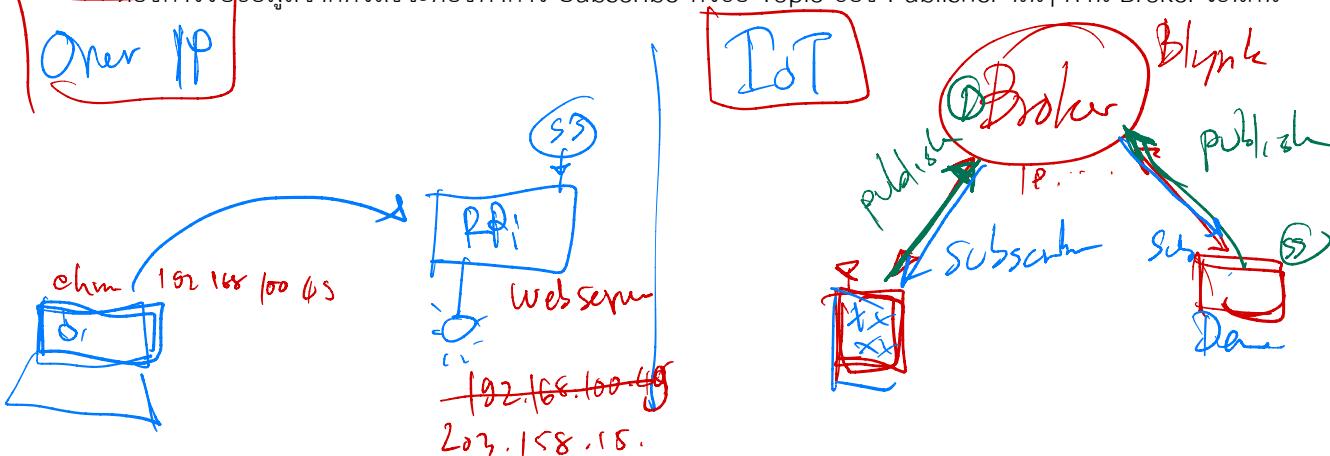


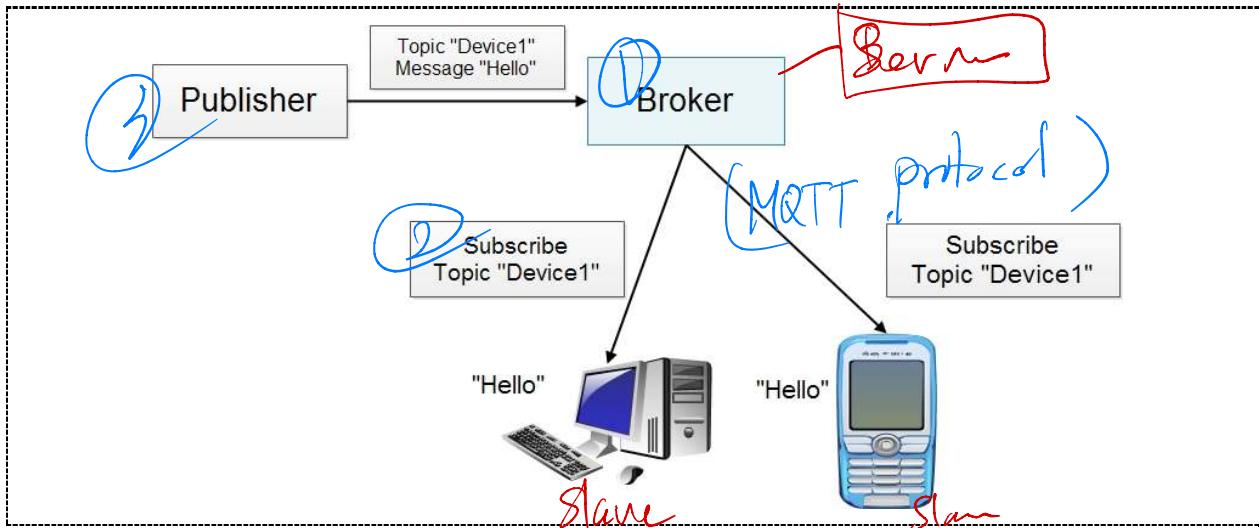
## 3.1 IoT Concept

บัญชีนเทคโนโลยีที่กำลังมาแรงสำหรับนักพัฒนาด้าน Embedded (ไม่ได้แค่เฉพาะ Embedded อย่างเดียว ครอบคลุม) เป็นเทคโนโลยีที่ก่อตัวกันมากคงจะหนีไม่พ้น IoT ซึ่งเป็นเทคโนโลยีใหม่ในยุคนี้เลยกว่าได้ แรงขนาดที่ว่า Microsoft เอง ก็ยังพอร์ต Windows 10 มาไว้บน Raspberry Pi และยังใจดีติด IoT มาให้ด้วย ซึ่งผมยังไม่ได้ตามลงไปดูว่าใช้ Broker ตัวไหน และมี Library ให้ใช้งานมาด้วยหรือไม่? หรือไม่苟กเข้าใจผิดเกี่ยวกับมันครับถ้าผิดพลาดก็ขออภัยมานะที่นี้ด้วยครับ.

IoT มันคืออะไร พอกันดูมีหลายลิงค์อธิบายไว้มากมาย เช่น [Internet of Things เมื่อคอมพิวเตอร์เริ่มคุยกันเองได้](#), [โลกแห่ง IoT มาถึงแล้ว](#), [IoT เทคโนโลยีที่ธุรกิจต้องรู้](#). ลองนึกภาพดูครับว่าถ้าหากอุปกรณ์สามารถสื่อสารไปมาหากันได้ผ่าน www ไม่ว่าจะเป็น PC, Smart Phone หรือแม้แต่ อุปกรณ์ขนาดเล็กพวก Micro-Controller, PLC, HUB, Switch หรืออะไรก็แล้วแต่ที่มันสามารถต่อระบบ Network ไม่ว่ามันจะอยู่ที่บ้าน ที่โรงงาน ไร่ นา ฟาร์ม โรงแรม โรงพยาบาล ฯลฯ ได้เราสามารถควบคุมมันได้ทั้งหมดที่ไหนก็ได้ในโลกใบนี้

องค์ประกอบหลักของ IoT จะมี 3 ส่วนคือ Broker, Publisher และ Subscriber. ซึ่งการรับและส่งข้อมูลนั้น มันจะส่งข้อมูลไปมหา埸กันนั้นจะส่งผ่านตัวกลางนั้นก็คือ Broker Server โดยตัวส่งนี้จะเรียกว่า Publisher ส่งข้อมูลเข้าไปยัง Broker พร้อมระบุหัวข้อ (Topic) ที่ต้องการส่งข้อมูลไป จากนั้นตัวรับซึ่งเรียกว่า Subscriber ถ้าหากตัวรับต้องการรับข้อมูลจากตัวส่งจะต้องทำการ Subscribe หัวข้อ Topic ของ Publisher นั้นๆ ผ่าน Broker เช่นกัน





จากขั้นตอนด้านบนจะมีตัว Publisher ทำการ Publish ข้อความ “Hello” ใน Topic Device1 เมื่อแล้วหากมีคอมพิวเตอร์ หรืออุปกรณ์อื่นๆทำการ Subscribe หัวข้อ Topic Device1 เมื่อ Publisher ทำการส่งข้อมูลไปยัง Topic อุปกรณ์ Subscribe จะได้ข้อความ “Hello” เชนเดียวกัน ก็คือทุกๆที่ใช้งานออนไลน์ที่คุยกันเป็นกลุ่มนั้นเลยครับ ซึ่งจะเห็นข้อความ “Hello” ในเวลาเดียวกันนั้นหมายความว่าอุปกรณ์ใดๆที่ทำการ Subscribe Topic เดียวกันก็จะได้ข้อความเดียวกันครับ

### 3.2 MQTT-Message Queue Telemetry Transport

โปรโตคอลที่ใช้สำหรับรับและส่งข้อมูลนั้นคือ MQTT ปัจจุบันถึง Version 3.1 ในที่นี้จะมาทำการทดลองส่งข้อมูลกันด้วย Server จะมีอยู่ด้วยกันหลายรายค่ายครับสำหรับที่ลิสตามาด้านล่างนี้ครับ

#### 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 ฟรีที่ผมพอกันได้ก็มีดังนี้ครับ

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

fr

### 3.3 IoT และ MQTT คือ อะไร?

MQTT ย่อมาจาก Message Queue Telemetry Transport เป็นโปรโตคอลประยุกต์ที่ใช้โปรโตคอล TCP เป็นรากฐาน ออกแบบมาสำหรับงานที่ต้องการ วิธีสารแบบเรียลไทม์แบบไม่จำกัดแพลตฟอร์ม หมายถึงอุปกรณ์ทุกชิ้นสามารถสื่อสารกันได้ผ่าน MQTT

MQTT จะแบ่งเป็น 2 ฝั่ง คือฝั่งเซิร์ฟเวอร์มักจะเรียกว่า **MQTT Broker** ส่วนฝั่งผู้ใช้งานจะเรียกว่า **MQTT Client** ใน การใช้งานด้าน IOT จะเกี่ยวข้องกับ MQTT Client เป็นหลัก โดยจะมี MQTT Broker ทั้งแบบฟรี และเสียเงิน ไว้รองรับอยู่แล้ว ทำให้การสื่อสารข้อมูลผ่าน MQTT จะใช้เซิร์ฟเวอร์ฟรี หรือ MQTT Broker ฟรี เหล่านี้เป็นตัวกลาง

ลักษณะการใช้งาน MQTT อาจจะเบรียบเสื่อมไปได้กับการใช้งานห้องเช่า Line สำหรับอุปกรณ์ โดยอุปกรณ์แต่ละตัวจะมีชื่อเป็นของตนเอง มี Username Password เป็นของตัวเอง และอาจจะมีห้องลับเฉพาะของตนเอง ดังนั้น การใช้งาน MQTT ผู้ใช้ในจึงจะขอตัวอย่างของ MQTT เพียงกับห้องเช่าได้ดังนี้

#### กลุ่มผู้ใช้ (User)

ใน MQTT จะแบ่งกลุ่มของผู้ใช้งานออกเป็น 2 ระดับ คือ

- ระดับสูงสุด – สามารถที่จะรับ-ส่งข้อมูลกับอุปกรณ์ หรือช่องทางใด ๆ ก็ได้ในระบบ หรือเบรียบได้กับแอ็อด มินที่สามารถเข้าไปดูข้อมูลได้ทุกห้องแม้จะเป็นห้องลับก็ตาม
- ระดับทั่วไป – สามารถรับ-ส่งข้อมูลกับอุปกรณ์หรือช่องทางที่กำหนดไว้เฉพาะเท่านั้น เบรียบได้กับ ผู้ใช้งาน Line ที่สามารถแซทในห้องที่ตัวเองสร้างได้ หรือเป็นสมาชิกในห้อง แต่ไม่สามารถเข้าไปแซทในห้องที่ไม่ได้เป็นสมาชิก

ในการใช้งานจริง ในอุปกรณ์ต่าง ๆ ควรจะใช้งานในระดับทั่วไป เพื่อความปลอดภัยกรณีอุปกรณ์เหล่านั้นถูก แฮกแล้วไม่สร้างความเสียหายไปยังอุปกรณ์อื่น ๆ ที่อยู่ในช่องทางเฉพาะของแต่ละอุปกรณ์

#### เส้นทาง (Topic)

เส้นทาง เบรียบเหมือนกับหัวข้อ หรือห้องแซทที่ต้องการจะคุยกับ และการคุยกันจะมีเฉพาะอุปกรณ์ที่อยู่ในห้องนั้น ๆ (Subscribe) ถึงจะสามารถได้รับข้อมูลที่มีการส่งไปในห้องนั้น ๆ ที่ถูกเรียกว่าเส้นทาง เนื่องจากการใช้งานส่ง ข้อมูลและรับข้อมูลจะเหมือนกับเส้นทางในระบบไฟล์ เช่น /Room1/LED ซึ่งระบบเส้นทางนี้นอกจากอุปกรณ์จะสามารถ รอการสนทนainห้องตามเส้นทาง /Room1/LED ได้แล้ว ยังสามารถอ่อนหนาเส้นทาง /Room1 ได้ด้วย หากเป็นการ รอพังในเส้นทาง(Subscribe) /Room1 จะหมายถึงการส่งข้อมูลได้ ๆ ที่นำหน้าด้วย /Room1 เช่น /Room1/LED , /Room1/Value ผู้ที่รอพัง (Subscribe) /Room1 อยู่จะได้รับข้อมูลเหล่านั้นด้วย

① MQTT กะ

### คุณภาพข้อมูล (QoS)

แบ่งออกเป็น 3 ระดับดังนี้

② QoS

- QoS 0 - ส่งข้อมูลเพียงครั้งเดียว ไม่ส่งให้ผู้รับจะได้รับหรือไม่
- QoS 1 - ส่งข้อมูลเพียงครั้งเดียว ไม่ส่งให้ผู้รับจะได้รับหรือไม่ แต่ให้จำค่าที่ส่งล่าสุดไว้ เมื่อมต่อใหม่จะได้รับข้อมูลครั้งล่าสุดอีกครั้ง
- QoS 2 - ส่งข้อมูลหลาย ๆ ครั้งจนกว่าปลายทางจะได้รับข้อมูล มีข้อเสียที่สามารถทำงานได้ช้ากว่า QoS0 และ QoS1

### การส่งข้อมูล (Publish)

การส่งข้อมูลในแต่ละครั้งจะต้องประกอบไปด้วยสิ่งที่ต้องมี คือ การส่งข้อมูล และคุณภาพข้อมูล ซึ่งการส่งข้อมูลจะเรียกว่า Publish

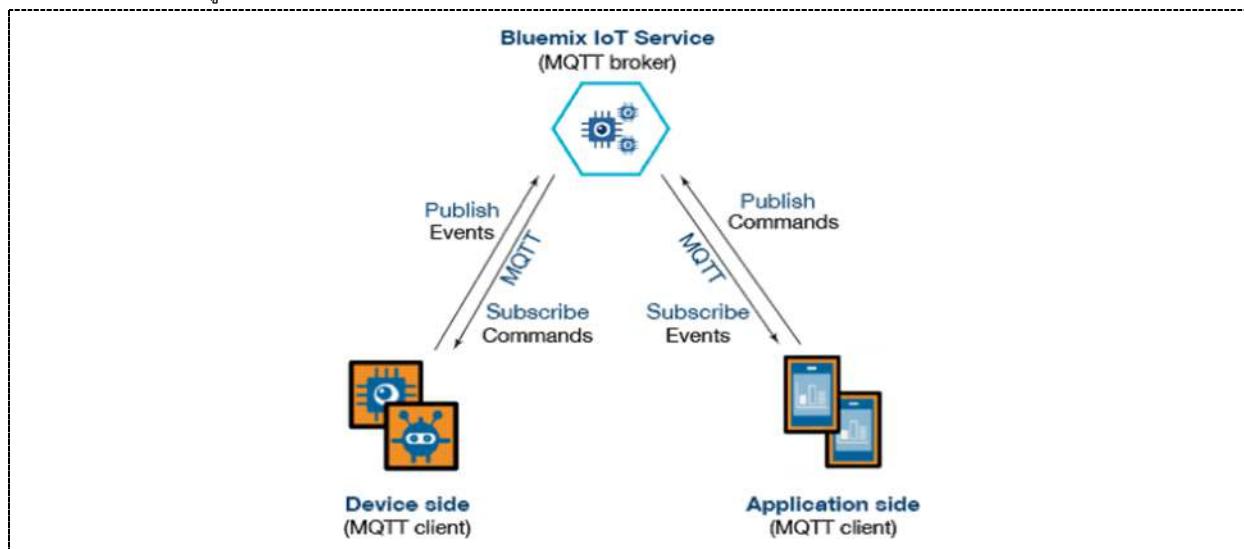
### การรับข้อมูล (Subscribe)

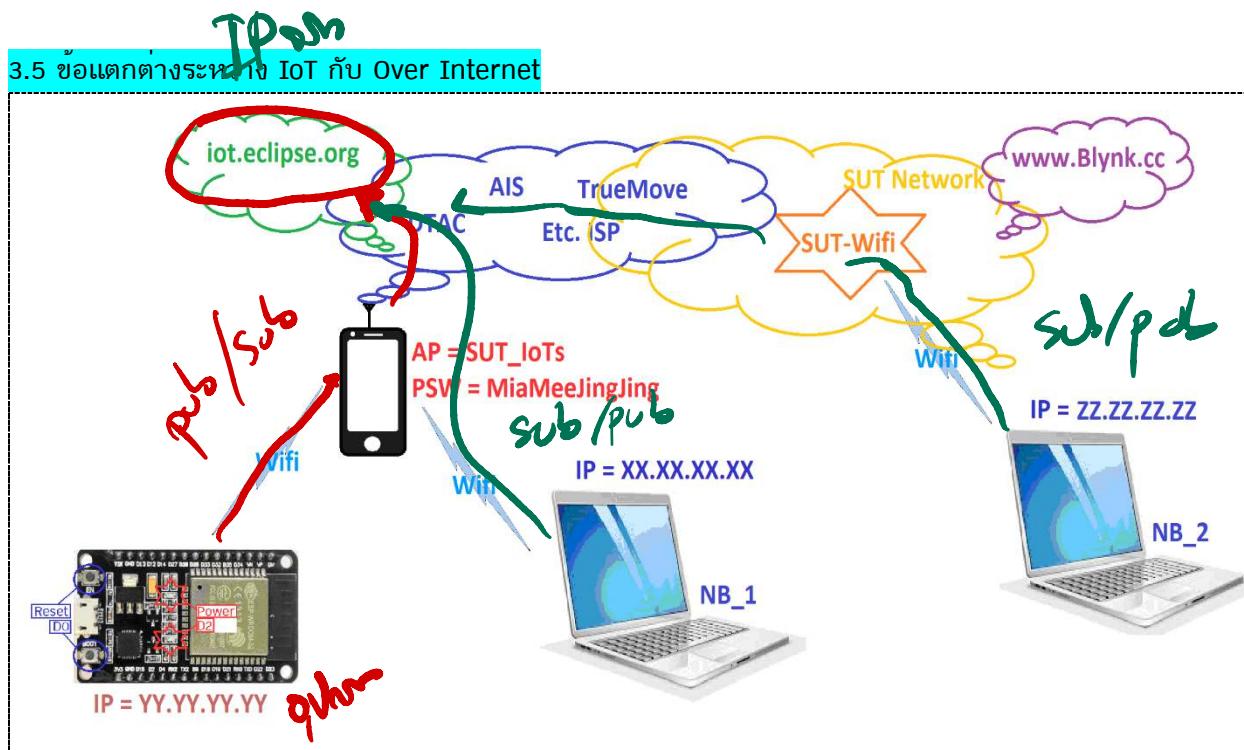
การรับข้อมูลในระบบ MQTT จะรับข้อมูลได้เฉพาะเมื่อมีการเรียกใช้การ Subscribe ไปยัง Topic ที่กำหนด อาจเปรียบได้กับการ Subscribe คือการเข้าไปนั่งรอเพื่อนในกลุ่ม Line ส่งแซทมาหา เมื่อมีการส่งข้อมูลเข้ามาจะเกิดสิ่งที่เรียกว่าเหตุการณ์ (Event) ให้เราดูเข้าไปดูข้อความที่เพื่อน ๆ ส่งเข้ามา

จะเห็นได้ว่า MQTT ที่เปลี่ยนสภาพห้องแข็งของอุปกรณ์ที่จะสนทนากันแลกเปลี่ยนข้อมูลกันแบบเรียลไทม์ผ่านเครือข่ายอินเตอร์เน็ต

### 3.4 IoT มีวิธีการทำงานอย่างไร?

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





การทดลองก่อนหน้าเป็นการควบคุมผ่านอินเตอร์เน็ต จำเป็นต้องรู้ IP ของอุปกรณ์ปลายทาง และระบบต้องอยู่ในวงเครือข่ายเดียวกัน เช่น จากฐานเราไม่สามารถใช้ NB\_2 เข้ามาควบคุม ESP32 ได้โดยตรง เพราะ IP=ZZ.ZZ.ZZ.ZZ และ IP=YY.YY.YY.YY อยู่คุณลักษณะเครือข่าย หากต้องการสามารถกำหนดเส้นทางจาก NB\_2 ผ่านเครือข่ายของมหาวิทยาลัย ไปยังผู้ให้บริการมือถือ วนมาที่มือถือ เข้ามายัง ESP32 การควบคุมสั่งการแบบนี้จะเป็นต้องรู้เลขปลายทางซึ่งเป็นไอพีของอุปกรณ์

กรณีของ IOTs กระบวนการข้างต้นจะปรับใหม่ คือ ไม่จำเป็นต้องรู้เลขไอพีของอุปกรณ์ปลายทางแต่ให้อุปกรณ์ร่วงไปรับคำสั่งที่ตัวกลาง (Broker) แทน จากรูป NB\_2 จะส่งคำสั่ง (Publish) ไปยังตัวกลาง ตัวอุปกรณ์ปลายทางต้องแจ้งรับข้อมูล (Subscribe) จากตัวกลาง เมื่อมีคำสั่งเข้ามาและตัวอุปกรณ์ปลายทางเข้ามารับข้อมูลอุปกรณ์ปลายทางค่อยทำงานตามคำสั่งที่ได้รับ

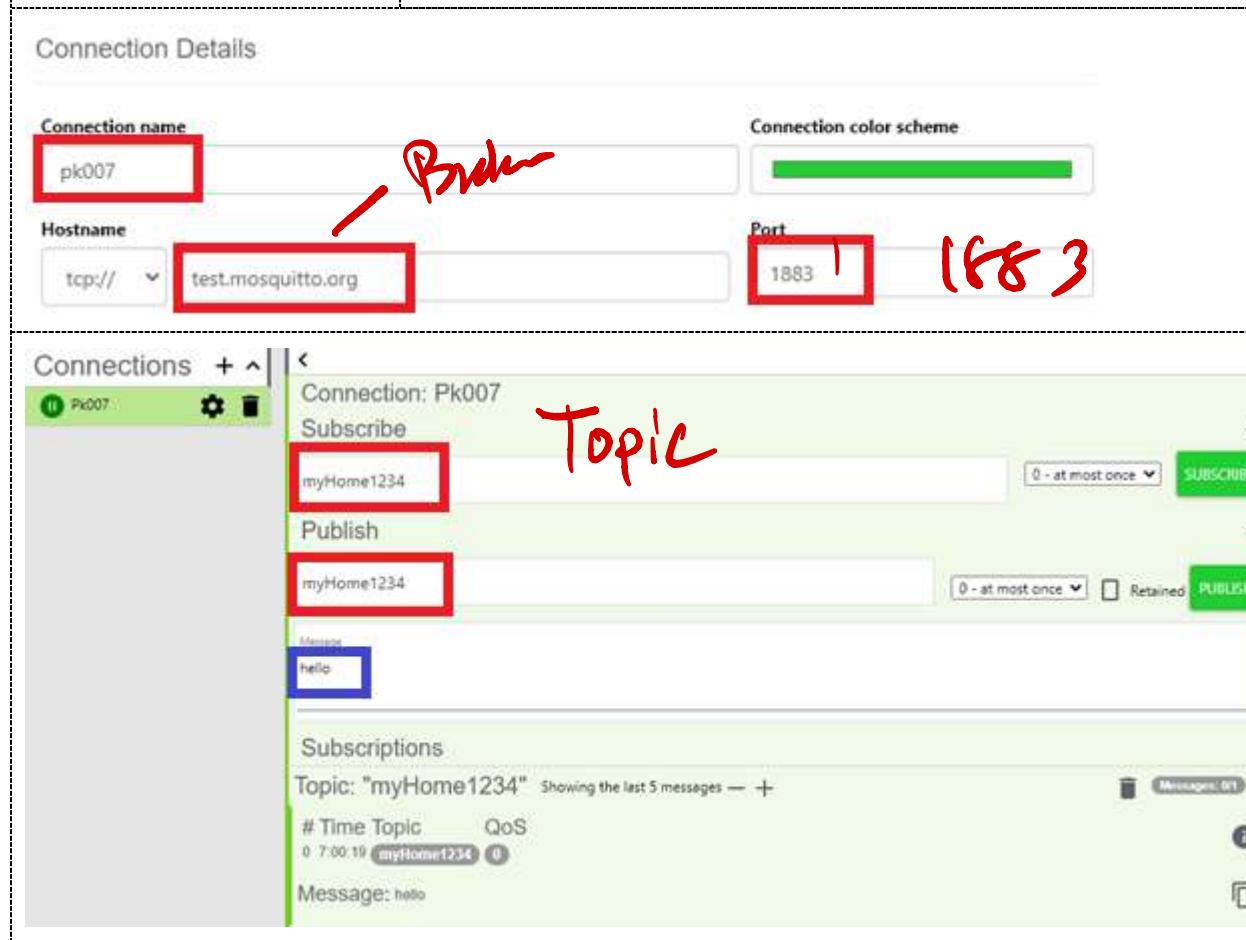
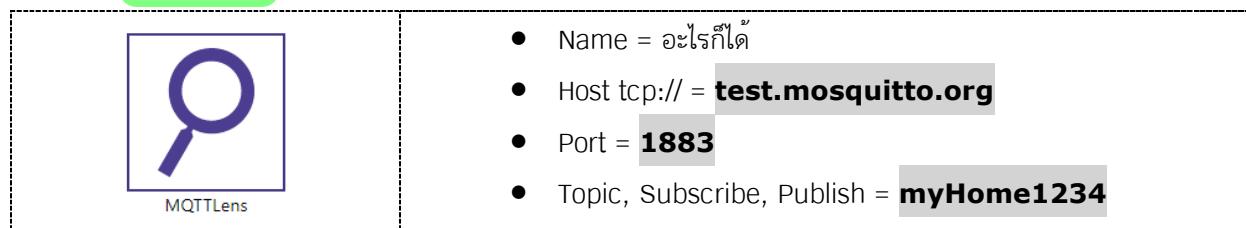
เห็นได้ว่าแบบแรกจำเป็นต้องเข้าให้ถึง ESP32 แต่แบบหลังใช้วิธีนัดรับข้อมูลที่ตัวกลางที่ทั้งสองฝ่ายตกลงกันไว้

## Lab303 – ESP32 to Public/Subscribe MQTT Server

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



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





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);
}

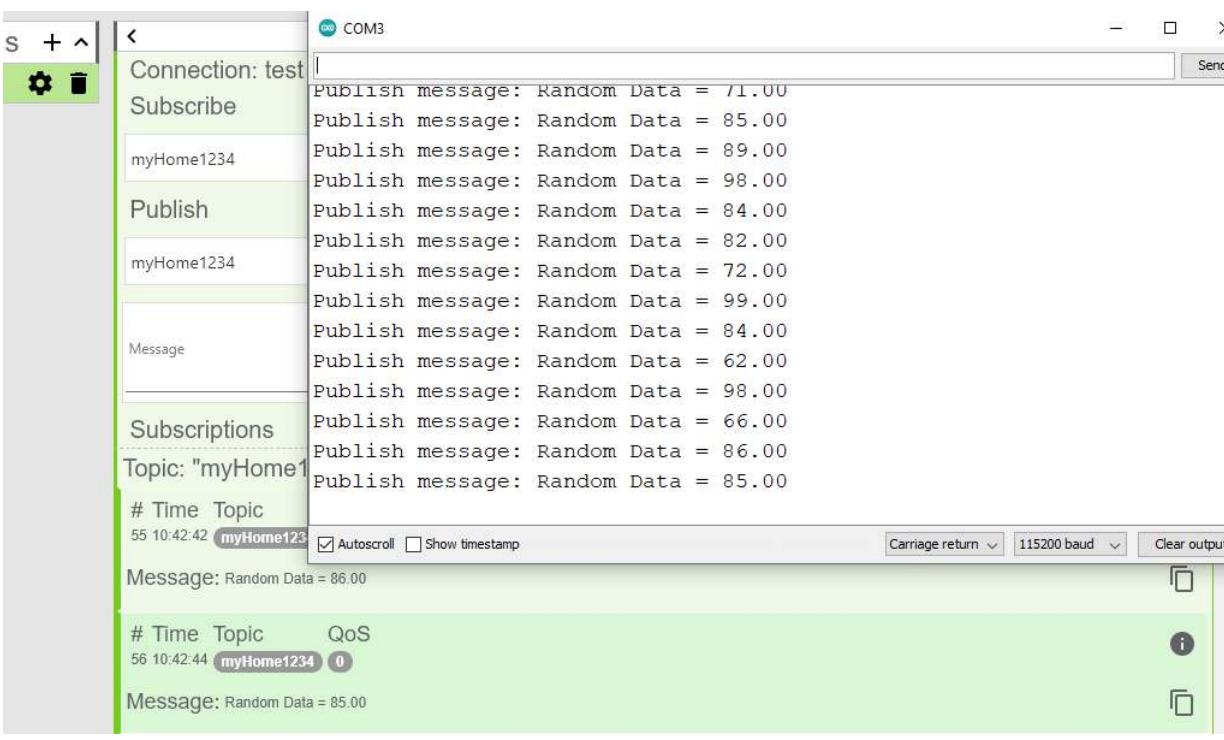
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  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);
  }
}
```

Handwritten annotations on the code:

- A red arrow points from the word "WiFi" in the first line to the "WiFi" section of the code.
- A blue arrow points from the word "SUB" to the "client.subscribe(myTopic);" line.
- A blue arrow points from the word "publis" to the "client.publish(myTopic, msg);" line.

7. ผลการทำงานดูที่ Serial Monitor ว่าส่งอะไรออกไป และดูที่ MQTTLens ว่าตีรับอะไรมาบ้าง



The Serial Monitor window shows the following publish messages:

```

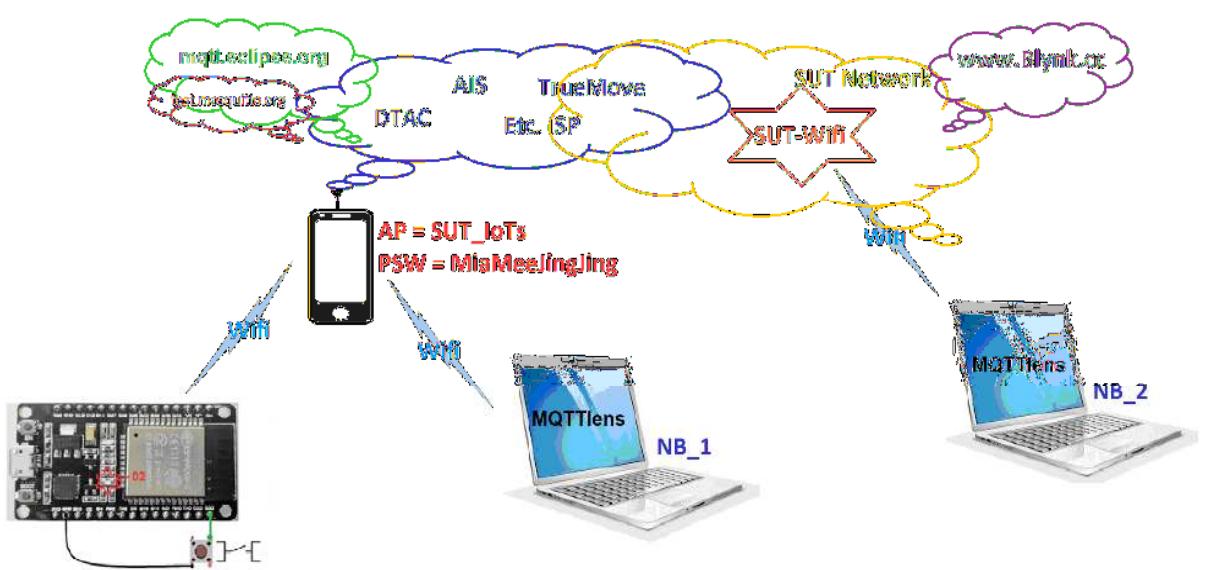
COM3
Publish message: Random Data = 71.00
Publish message: Random Data = 85.00
Publish message: Random Data = 89.00
Publish message: Random Data = 98.00
Publish message: Random Data = 84.00
Publish message: Random Data = 82.00
Publish message: Random Data = 72.00
Publish message: Random Data = 99.00
Publish message: Random Data = 84.00
Publish message: Random Data = 62.00
Publish message: Random Data = 98.00
Publish message: Random Data = 66.00
Publish message: Random Data = 86.00
Publish message: Random Data = 85.00

```

Message: Random Data = 86.00

#	Time	Topic	QoS
55	10:42:42	myHome1234	0

Message: Random Data = 85.00



The MQTTLens interface shows the following network traffic:

- AP = SUT\_IoTs**: A mobile device connected to a WiFi network.
- PSW = MisMeetingJing**: A laptop connected to a WiFi network.
- NB\_1**: A laptop connected to a WiFi network.
- NB\_2**: A laptop connected to a WiFi network.
- SUT Network**: A star-shaped network node.
- DTAC**, **AIS**, **TrueMove**, **Etc. (ISP)**: Internet Service Provider nodes.
- whatver.Blynk.cc**: A cloud-based service node.
- nodeclipse.org**, **lunarpush.org**: Other network nodes.

## 8. ໂຫລດໂປຣແກຣມທີ່ທຳງານ ຮັບຄໍາແລ້ວຈະບຸນ 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(topic);
    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);
    }
}

```

SUB

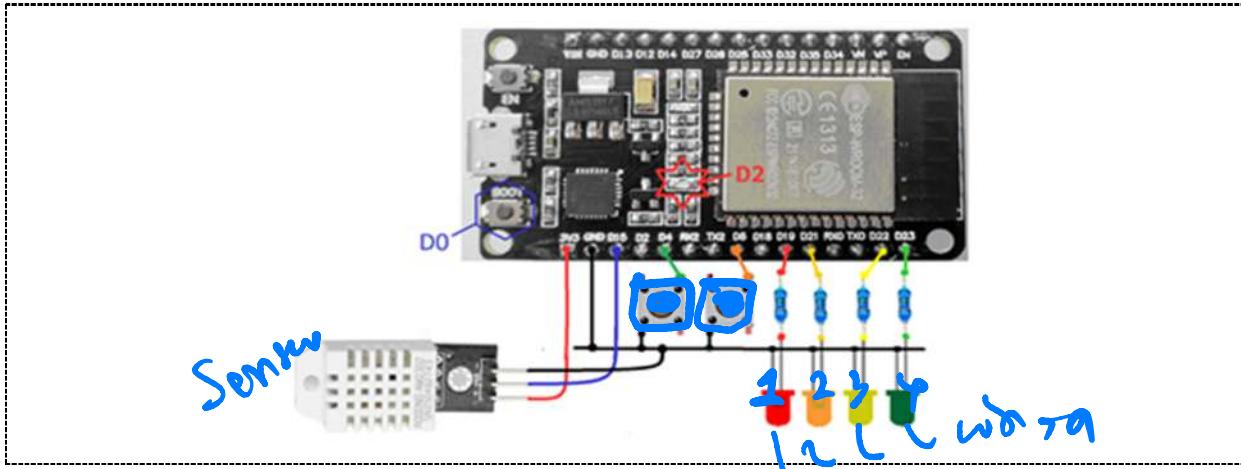
Payload 0 = off  
1 = on

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

<pre> -----&gt; Random Data = 91.00 Message arrived [myHome1234] Random Data = 66.00 -----&gt; Random Data = 66.00 Message arrived [myHome1234] ON -----&gt; ON Publish message: Random Data = 74.00 Publish message: Random Data = 84.00 Message arrived [myHome1234] Random Data = 74.00 -----&gt; Random Data = 74.00 Message arrived [myHome1234] OFF -----&gt; OFF Message arrived [myHome1234] Random Data = 84.00 -----&gt; 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 และส่งไปยัง MQTT Broker ทุกๆ 5 วินาที
- กำหนดให้ใช้ mqtt.eclipse.org เป็น Broker
- ควบคุมการเปิดปิด 4 LED
- รับค่าสวิตซ์กำหนด 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/%E0%B8%95%E0%B8%97%E0%B8%92%E0%B8%A1%E0%B8%9A-node-red-%E0%B8%81%E0%B8%A1-%E0%B8%81-3fcfa5ed140f9>

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

#### สรุปองค์ประกอบของ MQTT Protocol

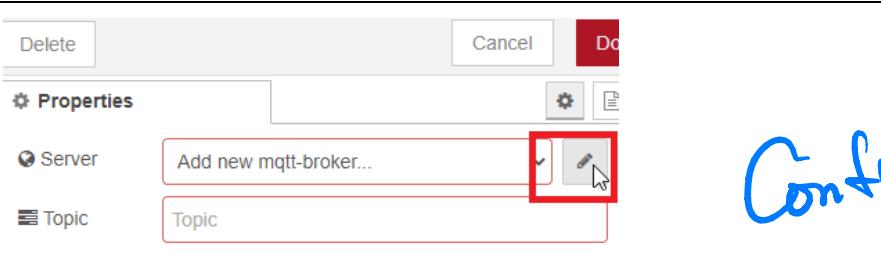
- Broker ทำหน้าที่เป็นตัวกลางคอยจัดการกับ ข้อความโดย อ้างอิงจาก Topic
- Publisher ทำหน้าที่ค่อยส่งข้อมูลไปยัง Topic
- Subscriber ทำหน้าที่คอยดูการเปลี่ยนแปลงของ message ที่อ้างอิงด้วยTopic เช่นถ้ามีหัวข้อหน้าสนใจและมีการเปลี่ยนแปลงก็จะทำการดึงข้อมูลนั้นๆ มาใช้งาน
- QoS คือ คุณภาพหรือความสำคัญของการส่งข้อมูล QOS0 คือการส่งข้อมูลเพียงครั้งเดียว ปลายทางอาจจะได้รับหรือไม่ได้รับก็ได้ QOS1 คือการส่งข้อมูลเพียงครั้งเดียว ปลายทางอาจจะได้รับหรือไม่ได้รับก็ได้ แต่จะมีการจำข้อมูลที่ล่าสุดไว เมื่อมีการเชื่อมต่อเข้ามาก็จะได้รับข้อมูลล่าสุดที่ส่งไป QOS2 คือการส่งข้อมูลไปจนการปลายทางจะได้รับ

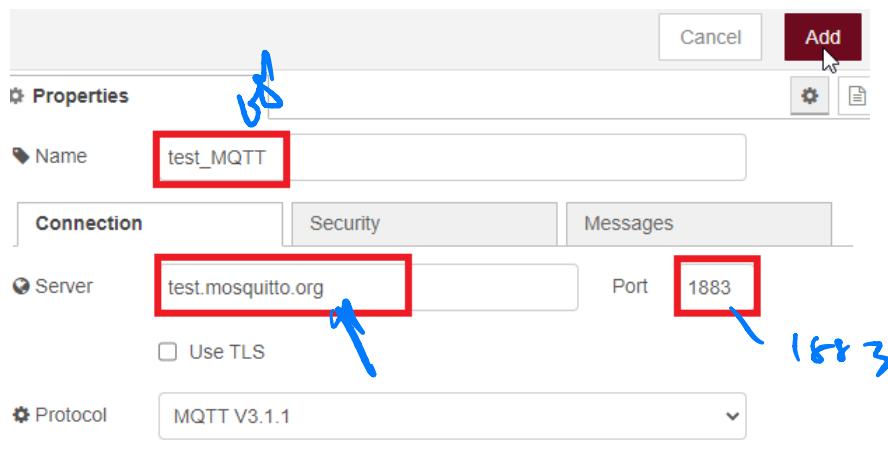
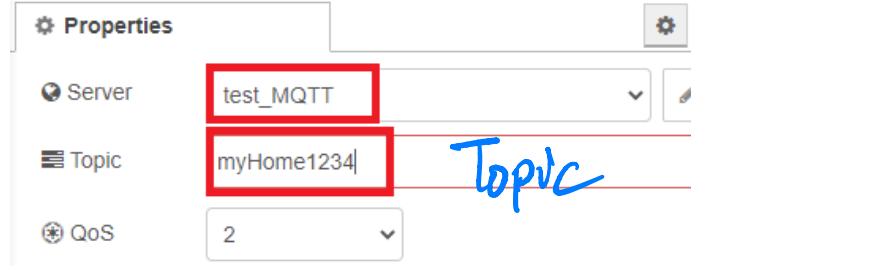
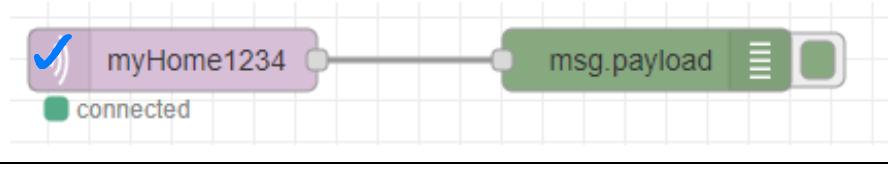
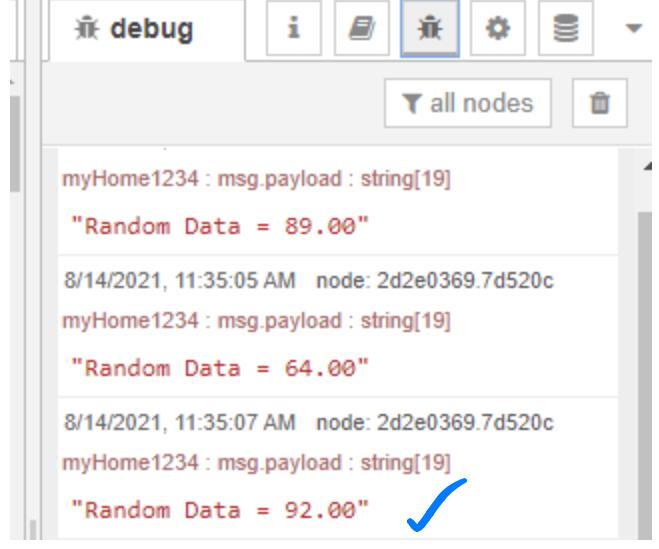
#### แนะนำ Link ของ Broker

- [test.mosquitto.org](http://test.mosquitto.org) Port 1883 (เป็น Broker ที่ใช้ในบทความนี้)
- [mqtt.eclipseprojects.io](http://mqtt.eclipseprojects.io) Port 1883
- [broker.hivemq.com](http://broker.hivemq.com) Port 1883

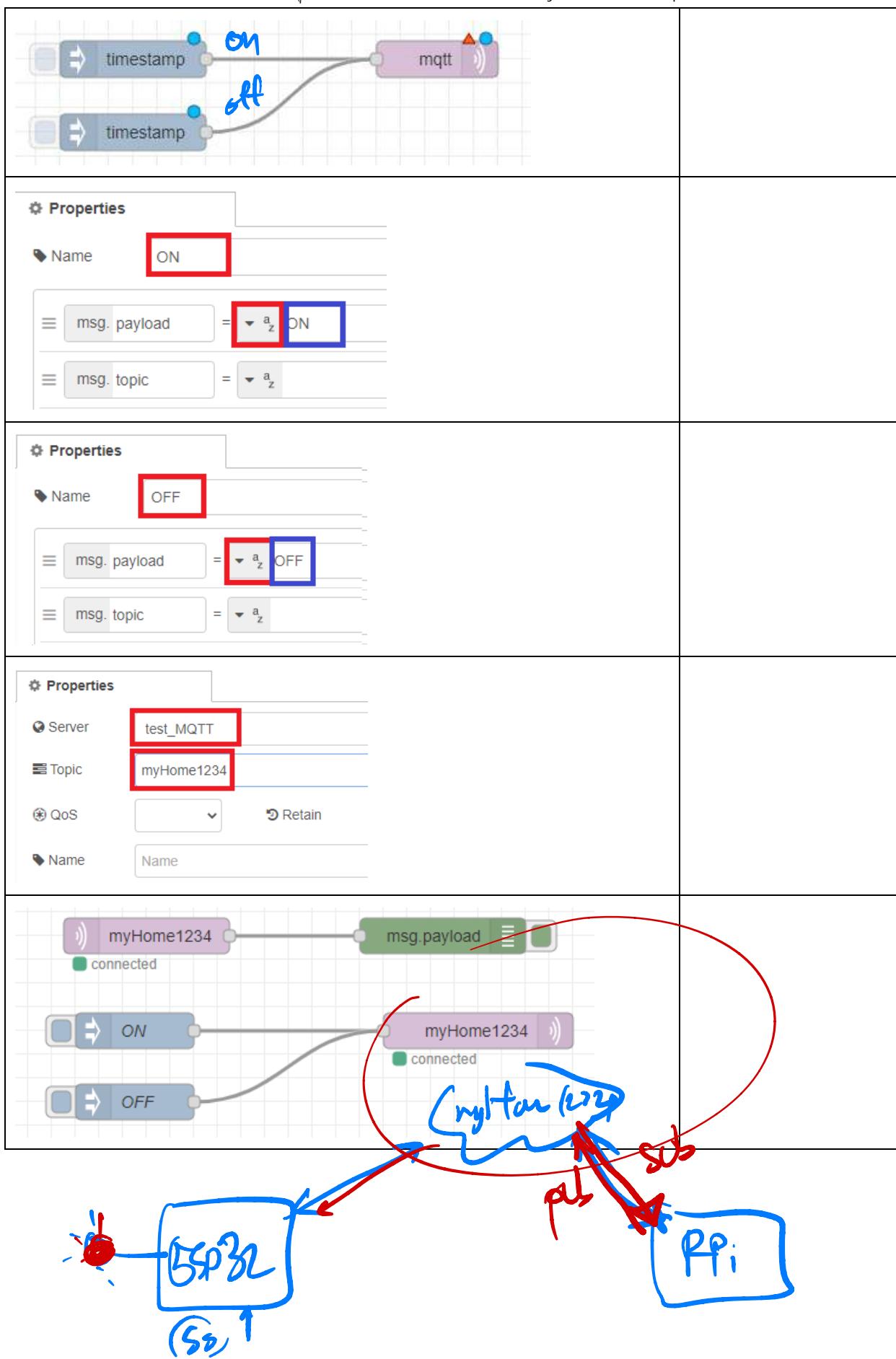
#### Lab304 – Raspberry Pi to Public MQTT Server

1. วางแผนและกำหนดค่า 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. ทดสอบสั่งค่า On/Off เพื่อควบคุม LED D2 บน ESP-32 ผ่าน myHome1234 Topic < Publish Test >



```

8/14/2021, 11:50:36 AM node: 2d2e0369.7d520c
myHome1234 : msg.payload : string[3]
"OFF"

8/14/2021, 11:50:37 AM node: 2d2e0369.7d520c
myHome1234 : msg.payload : string[19]
"Random Data = 68.00"

8/14/2021, 11:50:40 AM node: 2d2e0369.7d520c
myHome1234 : msg.payload : string[19]
"Random Data = 65.00"

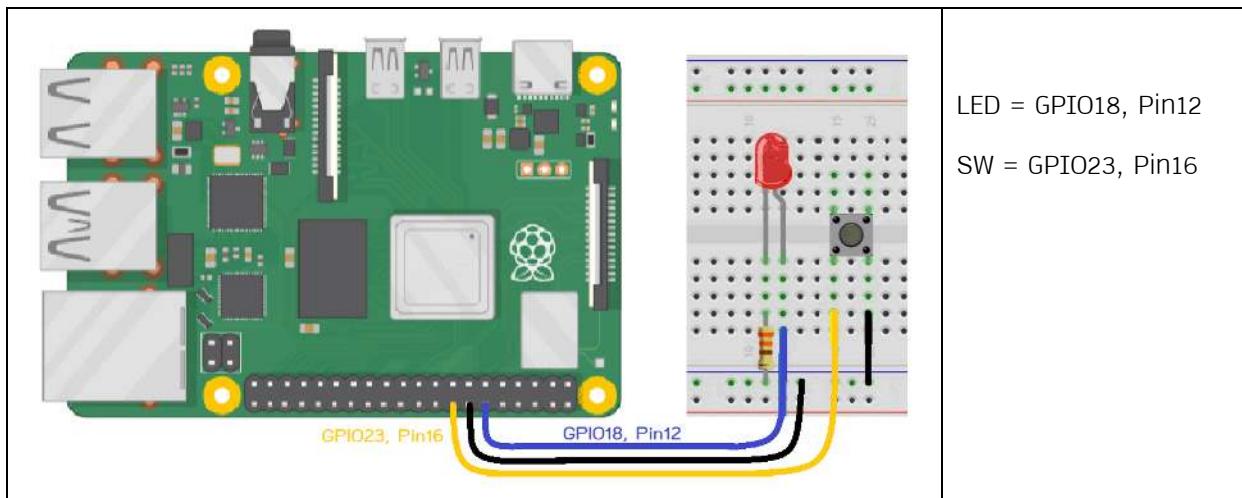
8/14/2021, 11:50:40 AM node: 2d2e0369.7d520c
myHome1234 : msg.payload : string[2]
"ON"

8/14/2021, 11:50:51 AM node: 2d2e0369.7d520c

```

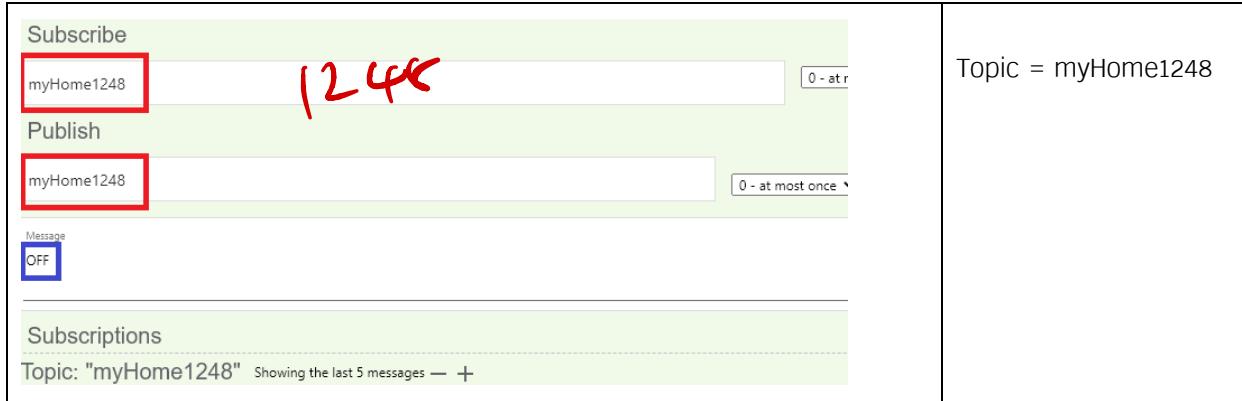
3. ทดสอบปั๊บใช้งาน Broker “mqtt.eclipseprojects.io:1883” ( Topic = myHome1428) เพื่อรับค่าการผับสวิตซ์และควบคุม LED ที่ RPi

### 3.1 วงจรทดสอบ



### 3.2 กำหนดค่าที่ MQTTlens

Connection name	test	Connection	mqtt.eclipseprojects.io 1883
Hostname	tcp:// mqtt.eclipseprojects.io	Port	1883
Client ID	lens_THbnJVA4wWiUn9LHeBA5El0nvJ		

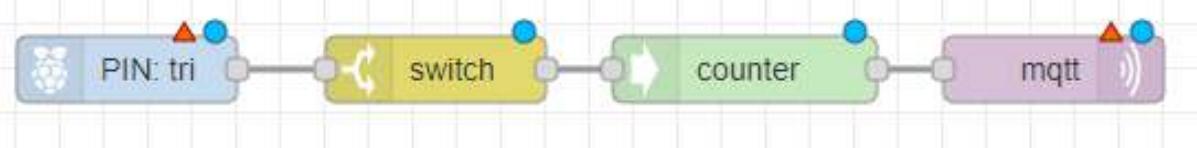


## 3.3 ทดสอบการรับค่าจาก MQTTLens เพื่อควบคุม LED

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

	Rpi-gpio-out = Pin12
	Get OFF Change to false ON Change to true
<p>Deploy</p>	
<p>MQTTLens "ON"/"OFF" Publish</p>	<p>Version 0.0.14</p> <pre> 8/14/2021, 5:26:27 PM node: c7b6e96f.01bcf8 myHome1248 : msg.payload : string[2] "ON"  8/14/2021, 5:26:27 PM node: c7b6e96f.01bcf8 myHome1248 : msg.payload : boolean true  8/14/2021, 5:26:31 PM node: c7b6e96f.01bcf8 myHome1248 : msg.payload : string[3] "OFF"  8/14/2021, 5:26:31 PM node: c7b6e96f.01bcf8 myHome1248 : msg.payload : boolean false </pre>

## 3.4 ทดสอบนับจำนวนการกดสวิตช์แล้วแสดงค่าที่ MQTTLens

<p><b>ttb-node-red-counter</b></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 style="text-align: right;"><a href="#">install</a></p> <p style="color: red; font-size: 2em; margin-left: 10%;">Add node</p>	Add Node >> ttb-node-red-counter, V0.1.0
	Counter node
<p>Add Flow</p> 	
 <p>16</p> <p><b>pullup</b>    Debounce <b>25 ms</b></p>	rpi-gpio out >> pin 16 >> pullup >> 25ms
<p>Property msg. payload</p> 	Switch >> == 1 then pass < ปล่อยนับ >
<p><b>Properties</b></p> <p>Increment <b>1</b></p> <p>Name Name</p>	Counter >> Ince = 1
<p>Server myEclips</p> <p>Topic myHome1248</p>	Mqtt-out >> server = myEclips >> Topic = myHome1248

Deploy

```

graph LR
    A[myHome1248] --> B{change: 2 rules}
    B --> C[PIN: 12]
    C -- OK --> D[msg.payload]
    D --> E[PIN: ↑ 16]
    E --> F{switch}
    F --> G[counter]
    G --> H[myHome1248]
    H -- connected --> I
  
```

กดสวิตช์ เพื่อแนบ

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      - +      Messages: 0/150

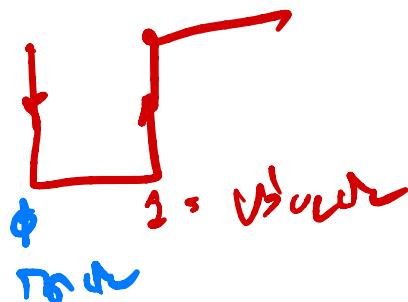
#	Time	Topic	QoS	Message
149	6:13:27	myHome1248	0	5,6,7,8

Message: 5,6,7,8

แก้ไข Switch  
>> == 0 then pass  
< กดแนบ >

Property      msg. payload

== a\_z 0      → 1 x



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

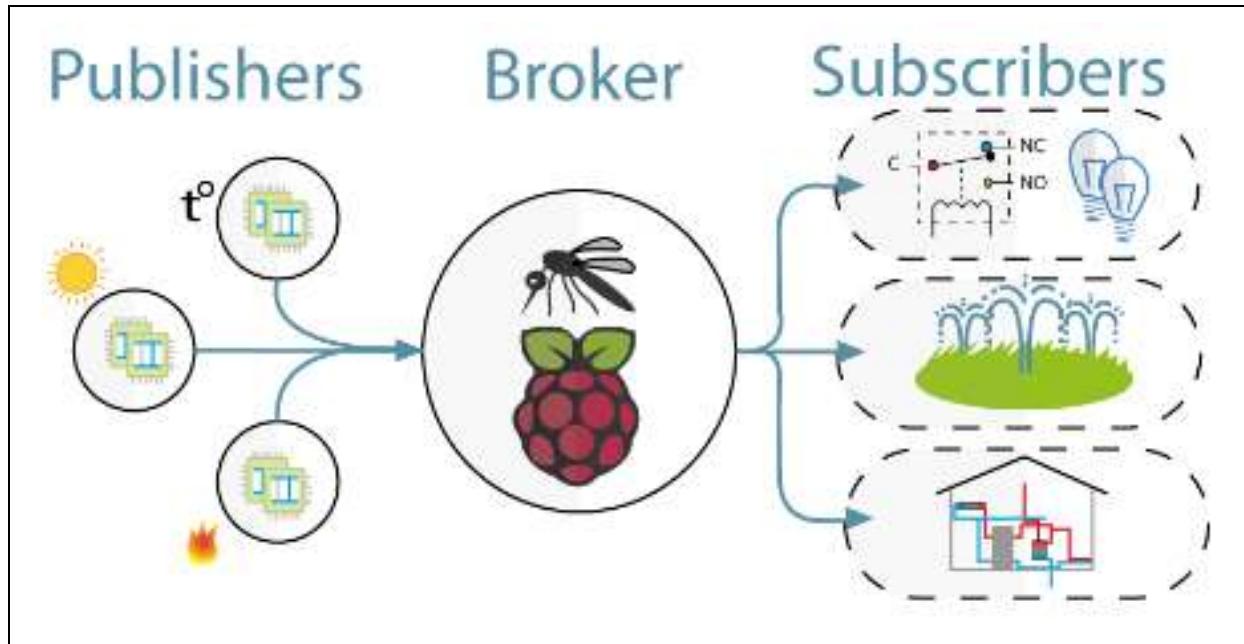
<https://www.hackster.io/dhairya-parikh/running-a-mqtt-broker-on-raspberry-pi-63c348>

<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](http://openfog.net/testboardesp8266_doc_3.html)

[www.raspberryhome.net/article/5/การติดตั้ง-mqtt-broker-บน-raspberry-pi](http://www.raspberryhome.net/article/5/การติดตั้ง-mqtt-broker-บน-raspberry-pi)



การพัฒนาทางด้าน IoT หรือ Internet of things นั้น คงจะหนีไม่พ้นเรื่อง protocol หรือ วิธีการส่งข้อมูลระหว่างเครื่อง 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 ซึ่งถูกสร้างขึ้นมาโดย Dr Andy Stanford-Clark of IBM, และ Arlen Nipper of Arcom (now Eurotech) เมื่อนานมาแล้ว ตั้งแต่ปี 1999 หรือประมาณ 17 ปีมาแล้ว ซึ่งก็ออกแบบมาสำหรับ Device ที่มีพลังงานประมวลผลต่ำ ต้องการประหยัดพลังงาน และสภาพอินเตอร์เน็ตที่แย่มากๆ โดยกำหนดให้มี QoS ถึง 3 ระดับ คือ

- QoS0 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่
- QoS1 – ส่งข้อมูลเพียงครั้งเดียว ไม่สนใจว่าผู้รับจะได้รับหรือไม่ แต่ให้จำนวนที่ส่งล่าสุดไว้ เมื่อมีการเชื่อมต่อใหม่จะได้รับข้อมูลครั้งล่าสุดอีกรอบ
- 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 ที่เรียกว่า MQTT Broker ซึ่งตัว Server จะทำหน้าที่จัดคิว ของข้อมูลต่างๆ ให้มีประสิทธิภาพสูงสุด

### Mosquitto MQTT

สาเหตุที่เราเลือกใช้ Mosquitto MQTT และทำ Server เองนั้นก็เนื่องจากว่า จริงๆแล้วมีการให้บริการ MQTT Server พรีอย่างมากมายในปัจจุบันทั้งในรูปแบบฟรีและไม่ฟรี ซึ่งปัญหาที่ตามมาก็คือ บาง Server มีที่ตั้งของ Server อยู่ไกลมากอีกหากนึงของโลก 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 นั้นทำหน้าที่ค่อยจัดการ message ที่มีขนาดเล็กดังนั้นตัว Raspberry Pi จึงสามารถเชื่อมต่ออุปกรณ์ Client ได้เป็น พันๆ Client ซึ่งนับว่าค่อนข้างเยอะเลยทีเดียว ซึ่งถ้าเทียบกับการที่เราเอามาทดลองและนำมาใช้งานเองนั้น ก็ถือว่าเพียงพอต่อความต้องการของเราแล้ว



### การติดตั้ง MQTT Broker บน Raspberry Pi

MQTT (Message Queuing Telemetry Transport) เป็น โปรโตคอล (Protocol) สำหรับการติดต่อสื่อสารและรับส่งข้อมูลระหว่างอุปกรณ์ต่างๆ ที่เชื่อมต่อกันอยู่บนระบบเครือข่ายภายใน(LAN) และระบบเครือข่ายอินเทอร์เน็ต อาทิ เช่น คอมพิวเตอร์ สมาร์ทโฟน เช่นเซอร์อุณหภูมิ เช่นเซอร์ตรวจจับการเคลื่อนไหว สวิทช์และปลั๊กไฟอัจฉริยะ อุปกรณ์ระบบกันขโมย อุปกรณ์ระบบควบคุมอัตโนมัติ และอุปกรณ์อื่นๆ เป็นต้น การรับส่งข้อมูลด้วย MQTT นั้นรวดเร็วเนื่องจากข้อมูลมีขนาดเล็กมากสามารถส่งไปยังหรือรับข้อมูลจากอุปกรณ์จำนวนมากได้พร้อมๆ กัน โดยใช้ระบบ Broker, Publisher, Subscriber และ Topic

โบรคเกอร์นั้นคือ MQTT Server ที่เรากำลังจะติดตั้ง ส่วนพับบลิเชอร์และซับส์ไครบ์เบอร์นั้นคืออุปกรณ์ต่างๆ ที่เขื่อมต่อกันอยู่ในระบบเนตเวิร์ค อุปกรณ์ที่ทำหน้าที่เป็นพับบลิเชอร์จะส่งข้อมูลตามหัวข้อ Topic ไปยังโบรคเกอร์ หากอุปกรณ์ใดๆ ที่สมัคร (Subscribe) ในหัวข้อ Topic นั้นไว้ โบรคเกอร์ จะส่งข้อมูลตามหัวข้อ Topic นั้นไปยังอุปกรณ์ต่างๆ ทั้งหมดที่สมัคร (Subscribe) ไว้ทั้งหมด ส่วนอุปกรณ์ใดๆ ไม่ได้สมัคร (Subscribe) ในหัวข้อ Topic นั้นไว้ ก็จะไม่ได้รับข้อมูล เพราะโบรคเกอร์ก็จะไม่ส่งข้อมูลในหัวข้อ Topic นั้นไปให้

อนึ่ง อุปกรณ์ทุกชนิดในระบบเนตเวิร์คสามารถถอด(Publish)หรือรับ(Subscribe)ข้อมูลในทุกหัวข้อ Topic ได้ตามที่กำหนด เราจะเห็นได้ว่าในปัจจุบัน มีการนำ MQTT มาใช้อย่างแพร่หลาย เพราะข้อมูลมีขนาดเล็กกะทัดรัด สามารถกระจาย(Distribute)ข้อมูลในหัวข้อ Topic ต่างๆ ไปยังอุปกรณ์(Subscribers)จำนวนมากได้อย่างรวดเร็ว ไม่ว่าอุปกรณ์เหล่านั้นจะอยู่ห่างไกลจากกันเพียงใดก็ตาม ข้อดีของ MQTT ก็คือไม่ต้องเสียเวลาในการฟอร์เวิร์ดพอร์ท(Forward port)ให้กับอุปกรณ์ทุกชนิดในระบบเนตเวิร์ค (ยกเว้นตัว โบรคเกอร์ที่ต้อง Forward port) เพื่อเปิดการสื่อสารทางไกลผ่านอินเทอร์เน็ต MQTT Server หรือโบรคเกอร์(Broker) ยังทำหน้าที่เป็นตัวกลาง ซึ่งจะคอยตรวจสอบการรับส่งข้อมูลระหว่างอุปกรณ์ต่างๆ ที่เชื่อมต่อกันในระบบ เพื่อให้แน่ใจว่าอุปกรณ์ซับส์ไครบ์เบอร์ทุกชนิดในระบบได้รับข้อมูลที่ส่งมาจากพับบลิเชอร์อย่างแน่นอน ถึงแม้ว่าในขณะที่พับบลิเชอร์กำลังส่งข้อมูลตามหัวข้อ Topic อยู่นั้น อุปกรณ์ที่เป็นซับส์ไครบ์เบอร์ของหัวข้อ Topic นั้นจะถูกปิด(Offline)อยู่ก็ตาม แต่เมื่ออุปกรณ์ที่เป็นซับส์ไครบ์เบอร์นั้นถูกเปิด(Online)กลับมาใช้งานอีกครั้ง โบรคเกอร์(Broker)ก็จะส่งข้อมูลนั้นให้หัวข้อ Topic นั้นไปให้กับอุปกรณ์ที่เป็นซับส์ไครบ์เบอร์ ซึ่งทำให้การสื่อสารรับส่งข้อมูลครบถ้วนสมบูรณ์ โดยมีขั้นตอนการติดตั้ง MQTT Server เพื่อทำหน้าที่เป็น Broker โดยมีขั้นตอนดังต่อไปนี้

## Lab305 – Install Private MQTT Server

- ป้อนคำสั่งเพื่อ Update/ Upgrade Raspbian และ เริ่มบูตเครื่องใหม่

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
Sudo reboot
```

- 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, 85.199.83.194
Connecting to repo.mosquitto.org (repo.mosquitto.org)|2001:ba8:1f1:f271::2|:80...
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.list 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
```

*Servr* ↗  
*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/mosquitto/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 ::1:1883               ::::*                      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. แก้ไขไฟล์คอนฟิกเกอเรชัน โดยเพิ่มบรรทัดใหม่ 3 บรรทัดที่ด้านล่างสุดของไฟล์ แล้วบันทึกการเปลี่ยนแปลงในไฟล์ โดยกดปุ่ม Ctrl+O เมื่อปรากฏชื่อไฟล์ ให้กดปุ่ม Enter เพื่อยอมรับ และออกจากเทกซ์ติดิทเตอร์ 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

Un - my mqtt

pas - my raspi

5. ในขณะนี้ เราได้ติดตั้ง Mosquitto เสร็จเรียบร้อยแล้ว และจะต้องบูทเครื่องใหม่ เพื่อให้ Mosquitto เริ่มทำงาน

```
sudo reboot
```

6. เริ่มทดสอบการทำงานของ MQTT Server หรือ Mosquitto โดยเราจะทดสอบด้วยการเรียกหน้าต่าง PuTTY ออกมา 2 หน้าต่าง เพื่อจำลองว่าเรา มีอุปกรณ์ 2 ชิ้น ซึ่งหน้าต่างหนึ่งจะทำหน้าที่เป็นชับล์ไครบ์เบอร์หรือตัวรับข้อมูล (Subscriber) อีกหน้าต่างหนึ่งจะทำหน้าที่เป็นพับบลิเชอร์หรือตัวส่งข้อมูล (Publisher) เริ่มเรียกโปรแกรม PuTTY แล้ว Connect ใหม่ โดยใช้หมายเลข IP Address เดิม

- 6.1 การรับข้อมูล (Subscriber) → ป้อนชับล์ไครบ์ Topic ชื่อ mynew/test

**mosquitto\_sub -d -u mymqtt -P myraspi -t mynew/test**

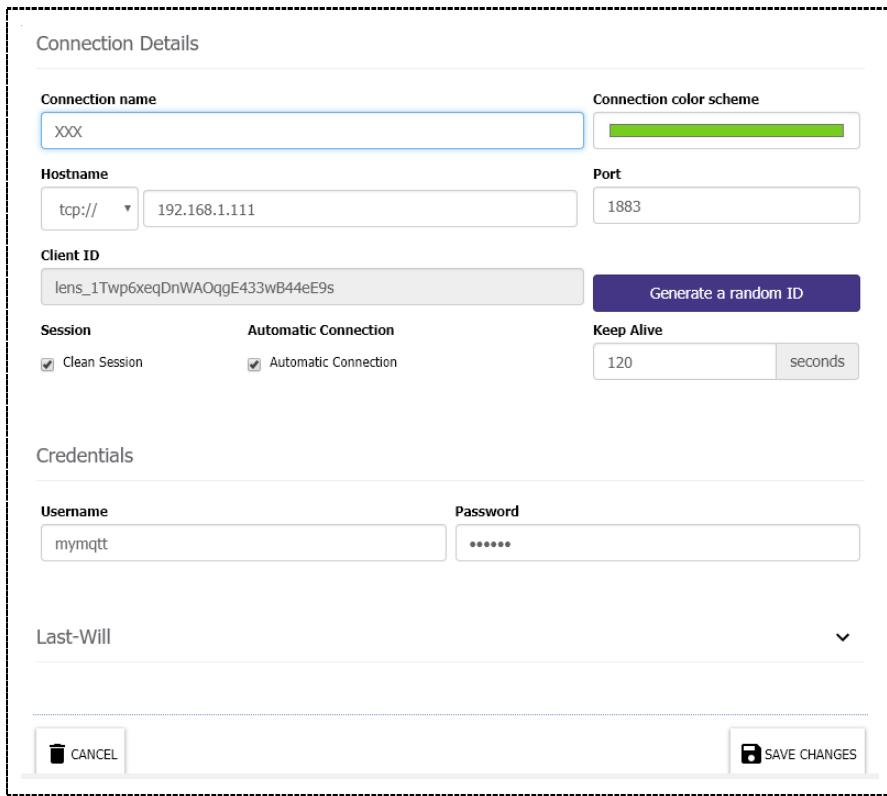
```
pi@raspberrypi: ~
pi@raspberrypi: ~ $ mosquitto_sub -d -u mymqtt -P myraspi -t mynew/test
Client mosqsub|872-raspberrypi sending CONNECT
Client mosqsub|872-raspberrypi received CONNACK
Client mosqsub|872-raspberrypi sending SUBSCRIBE (Mid: 1, Topic: mynew/test, QoS : 0)
Client mosqsub|872-raspberrypi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

- 6.2 การส่งข้อมูล (Publisher) → ป้อนพับล์ไครบ์ข้อมูล "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><b>Connection Details</b></p> <p><b>Connection name:</b> XXX</p> <p><b>Hostname:</b> tcp:// 192.168.1.111</p> <p><b>Port:</b> 1883</p> <p><b>Client ID:</b> lens_1Twp6xeqDnWAQgE433wB44eE9s</p> <p><b>Session:</b> <input checked="" type="checkbox"/> Clean Session    <b>Automatic Connection:</b> <input checked="" type="checkbox"/></p> <p><b>Keep Alive:</b> 120 seconds</p> <p><b>Credentials</b></p> <p><b>Username:</b> mymqtt    <b>Password:</b> *****</p> <p><b>Last-Will</b></p> <p><b>SAVE CHANGES</b></p>	<ol style="list-style-type: none"> <li>1. name = อะไรก็ได้</li> <li>2. RPi IP,Port = 1883</li> <li>3. UName = mymqtt, Pass=raspi</li> </ol>																
 <p><b>Connection: XXX</b></p> <p><b>Subscribe:</b> mynew/test</p> <p><b>Publish:</b> mynew/test</p> <p><b>Message:</b> Haha</p> <p><b>Subscriptions:</b></p> <p>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> </tbody> </table> <p><b>Message:</b> Haha</p> <table border="1"> <thead> <tr> <th>#</th> <th>Time</th> <th>Topic</th> <th>QoS</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>11:45:04</td> <td>mynew/test</td> <td>0</td> </tr> </tbody> </table> <p><b>Message:</b> SUT12345678</p>	#	Time	Topic	QoS	0	11:43:52	mynew/test	0	#	Time	Topic	QoS	1	11:45:04	mynew/test	0	<ol style="list-style-type: none"> <li>4. Topic = mynew/test, and Subscribe</li> <li>5. Topic = mynew/test, Massage = "Haha" and Publish</li> <li>8. Show "SUT12345678"</li> </ol>
#	Time	Topic	QoS														
0	11:43:52	mynew/test	0														
#	Time	Topic	QoS														
1	11:45:04	mynew/test	0														

```
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”

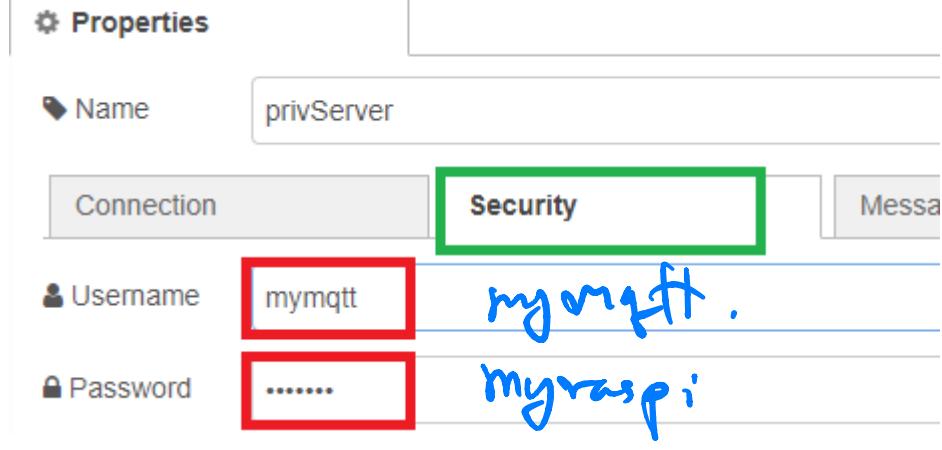
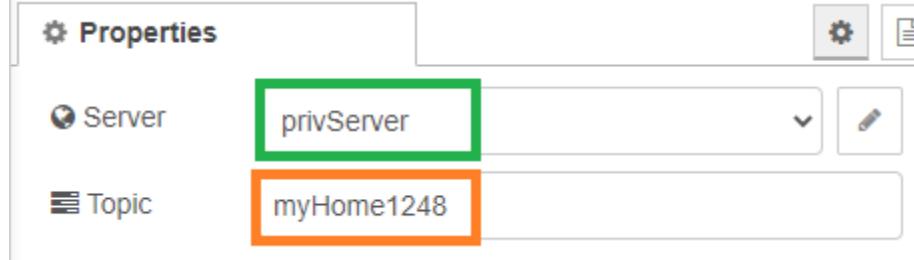
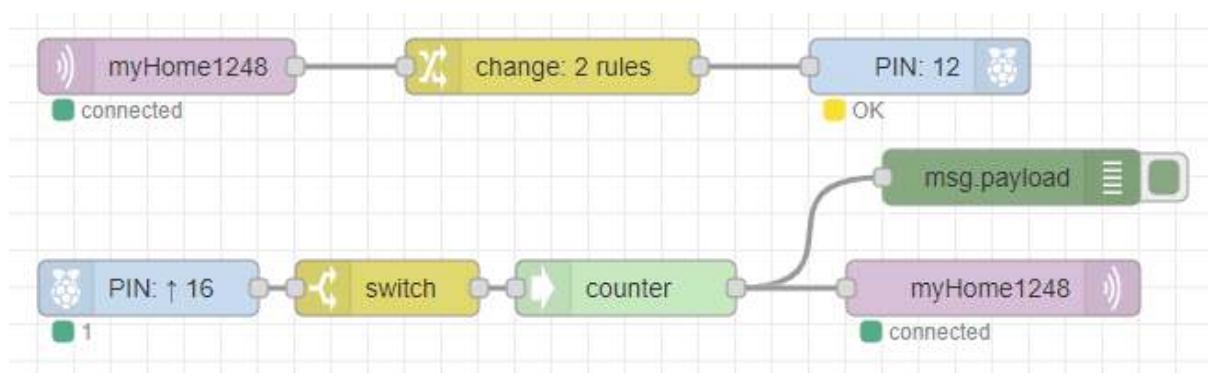
```
pi@raspberrypi:~ $ mosquitto_pub -d -u mymqtt -P myrasp -t mynew/test -m "Test12
34"
Client mosqpub|919-raspberrypi sending CONNECT
Client mosqpub|919-raspberrypi received CONNACK
Client mosqpub|919-raspberrypi sending PUBLISH (d0, q0, r0, m1, 'mynew/test', ..
.. (8 bytes))
Client mosqpub|919-raspberrypi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

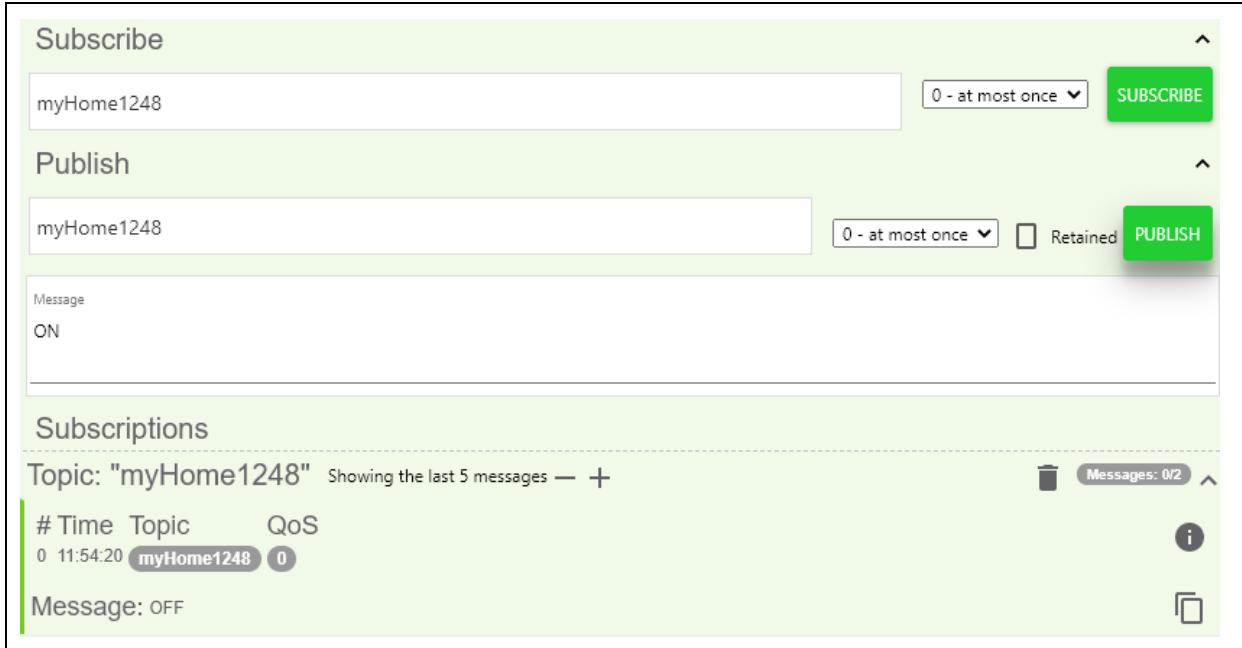
7. Publish “SUT12345678”

*Solve*

## 8. ทดสอบด้วย NodeRED

*Add Broker*

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



9. ทดสอบด้วย 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/d3/6dc8fd14746fcde6a556f932b5fedcb85b3a092e0e986372c0e7/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 screenshot shows the Geany IDE interface with the file `Publisher.py` open. The code defines constants for MQTT broker, port, keepalive interval, topic, message, user, and password, and implements an `on_connect` handler. Below the editor is a window titled "Subscriptions" showing a message received on the topic "testTopic".

```
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 import paho.mqtt.client as mqtt
10 import time
11
12 # Define on_connect event Handler
13 def on_connect(mosq, obj, rc):
14     print "Connected to MQTT Broker"
15
```

Subscriptions

Topic: "testTopic" Showing the last 1 messages — +

#	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()

```

Subscribe.py - /home/pi - Geany

File Edit Search View Document Project Build Tools Help

Subscribe.py

```

1 MQTT_BROKER = "192.168.100.49"
2 MQTT_PORT = 1883
3 MQTT_KEEPALIVE_INTERVAL = 45
4 MQTT_TOPIC = "testTopic"
5 MQTT_User = "mymqtt"
6 MQTT_Pass = "myraspi"
7
8 import paho.mqtt.client as mqtt
9 import time
10 # Define on_connect event Handler
11 def on_connect(self, mosq, obj, rc):
12     #Subscribe to a the Topic
13     mqttc.subscribe(MQTT_TOPIC, 0)
14

```

### Publish

testTopic 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 ในfarm อัจฉริยะ  
Chatbot LINE from Raspberry Pi MQTT Server for Smart Farming

ข้อ-สกุล :

6/6 - คำถ้ามทัยบทเพื่อทดสอบความเข้าใจ

Quiz\_301 – Blynk with Node-RED

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

random

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

iot.eclipse

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 ไปแสดงที่บน private Server

โปรแกรมที่ใช้ทดสอบ

รูปถ่ายหน้า Web Browser

รูปการทดสอบ 1

รูปการทดสอบ 2