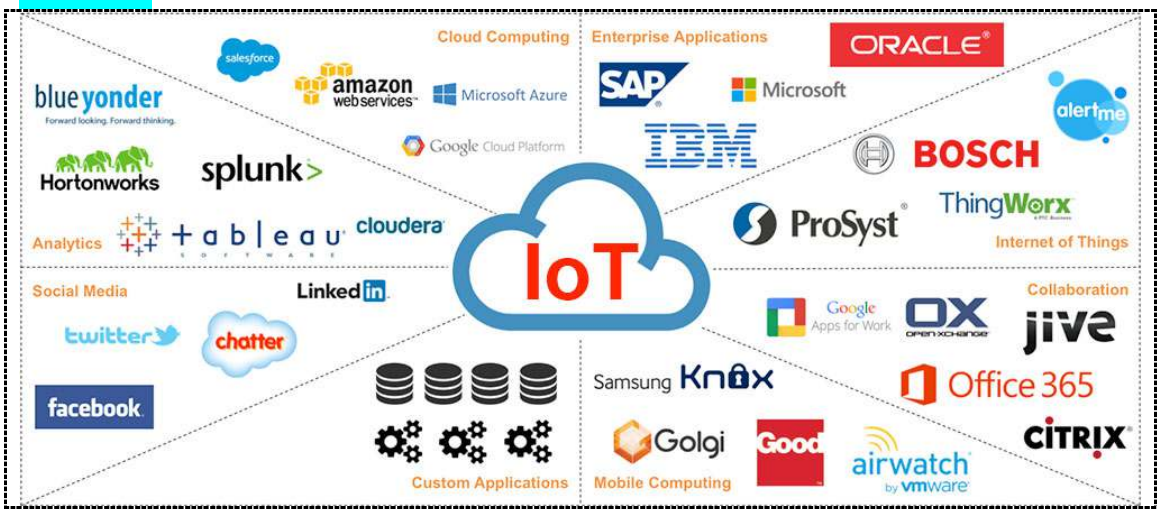
Part03b - ESP32 IoTs and MQTT Protocols

https://techtutorialsx.com/2017/04/24/esp32-publishing-messages-to-mqtt-topic/
https://www.ioxhop.com/article/74/esp32-เบื้องต้น-บทที่-13-เชื่อมต่อกับ-mqtt

1a. เนื้อหา

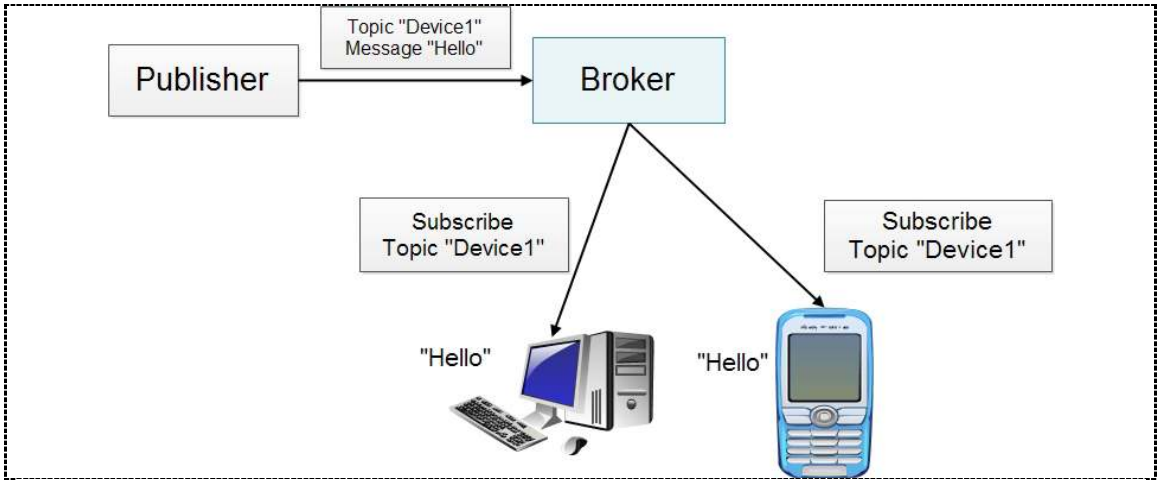


1.1 IoT Concept

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

IoT มันคืออะไร พอค้นดูมีหลายลิงค์อธิบายไว้มากมาย เช่น Internet of Things เมื่อคอมพิวเตอร์เริ่มคุยกันเองได้ , โลกแห่ง IoT มาถึงแล้ว , IoT เทคโนโลยีที่ธุรกิจต้องรู้. ลองนึกภาพดูครับว่าถ้าหากอุปกรณ์สามารถสั่งงานไปมาหากันได้ผ่าน www ไม่ว่าจะเป็น PC, Smart Phone หรือแม้แต่อุปกรณ์ขนาดเล็กพวก Micro-Controller, PLC, HUB, Switch หรืออะไรก็แล้วแต่ที่มันสามารถต่อระบบ Network ไม่ว่ามันจะอยู่ที่บ้าน ที่โรงงาน ไร่ นา ฟาร์มโรงเรือน โรงงานอุตสาหกรรมหรือที่อื่นๆที่มีระบบเน็ตเวอร์ที่เข้าถึง www ได้เราจะสามารถควบคุมมันได้ทั้งหมดที่ไหนก็ได้ในโลกใบนี้

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



จากรูปภาพด้านบนจะมีตัว Publisher ทำการ Public ข้อความ “Hello” ใน Topic Device1 เมื่อและถ้าหากมีคอมพิวเตอร์ หรืออุปกรณ์อื่นๆทำการ Subscribe หัวข้อ Topic Device1 เมื่อ Publisher ทำการส่งข้อมูลไปยัง Topic อุปกรณ์ Subscribe จะได้ข้อความ “Hello” เช่นเดียวกัน. ก็คล้ายๆกับที่ใช้งานไลน์ที่คุยกันเป็นกลุ่มนั้นเลยครับ. ซึ่งจะเห็นข้อความ “Hello” ในเวลาเดียวกันนั้นหมายความว่าอุปกรณ์ใดๆที่ทำการ Subscribe Topic เดียวกันก็จะได้ข้อความเดียวกันครับ

1.2 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

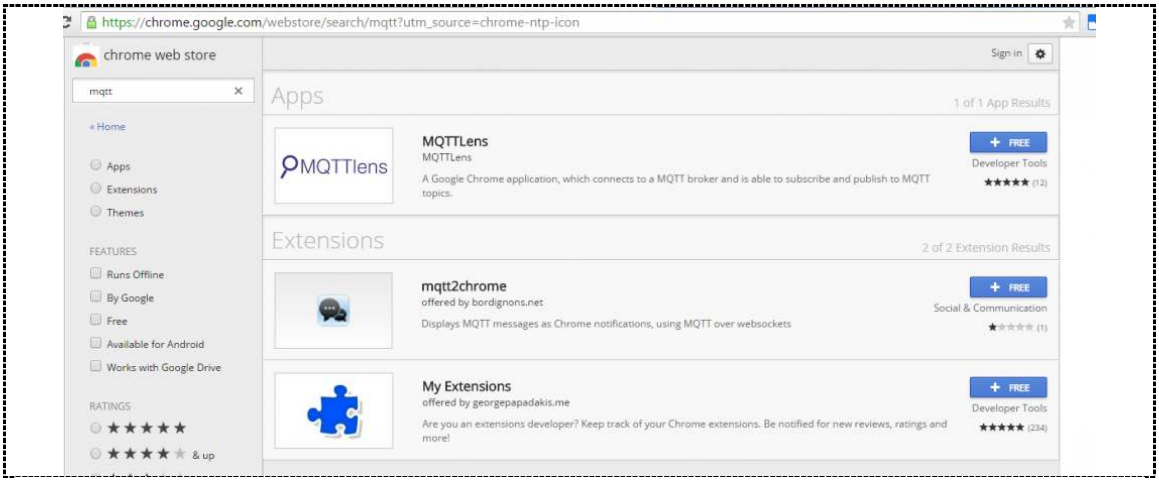
- Paho
- Xenqtt
- mqtt.js
- node_mqtt_client
- Ascoltatori
- Arduino MQTT Client

สำหรับ MQTT Broker Server ฟรีที่ผมพอค้นได้ก็มีดังนี้ครับ

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

Step1a/3 กำหนดตัว Subscribe

สำหรับเครื่องมีสำหรับทดสอบที่จะทำการส่งข้อมูล(pub) และรับข้อมูล(sub) ก็มีอยู่ด้วยกันหลายตัวครับเช่น แต่ละเลือกมาใช้งานสักตัวหนึ่ง ในที่นี้ผมเลือกเป็น plugin สำหรับ chrome คือ MQTTLens

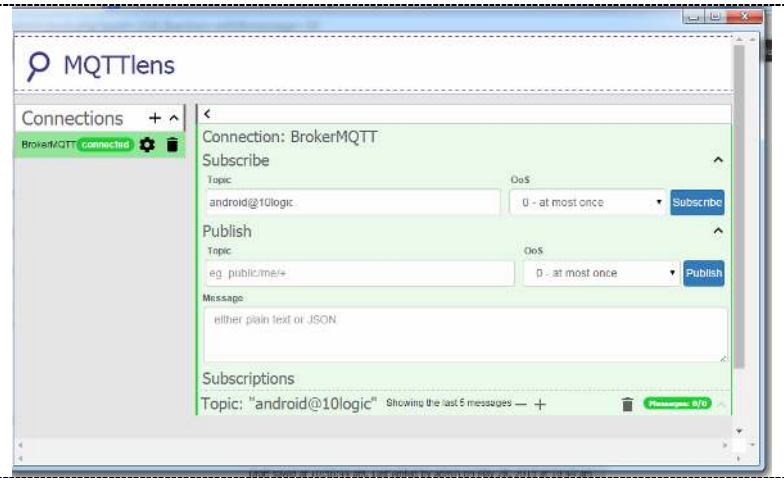


Mqttlens

mqttlens

เปิด MQTTLens ขึ้นมาจากนั้นป้อนรายละเอียด เมื่อป้อนรายละเอียดครบให้คลิกที่ CREATE CONNECTION

- Connection Name: test_MQTT ← อะไรก็ได้
- Hostname: test.mosquitto.org
- Port: 1883 ← default = 1883
- Client ID: RXL77Nb ← ตามที่ MQTTLens ให้มา



mqttlens

- Subscribe: android@10logic

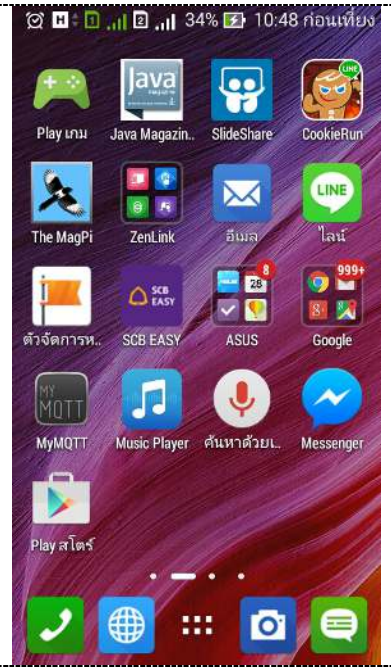
ในที่นี้ผมจะทำการ Subscriber ที่ Topic ชื่อว่า android@10logic

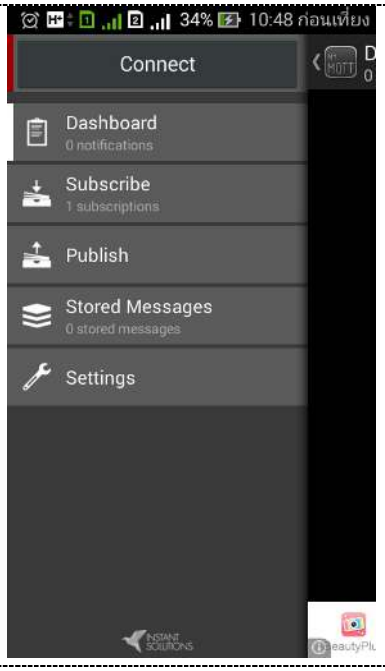
Step2a/3: กำหนดตัว Publisher

Publisher ซึ่งเป็น App สำหรับ Android ทำการ Public ข้อความ Hello 10logic ไปยัง Topic android@10logic เข้าไปใน play store และค้นคำว่า MyMQTT แล้วติดตั้งลงบน Smart Phone ของเราครับ

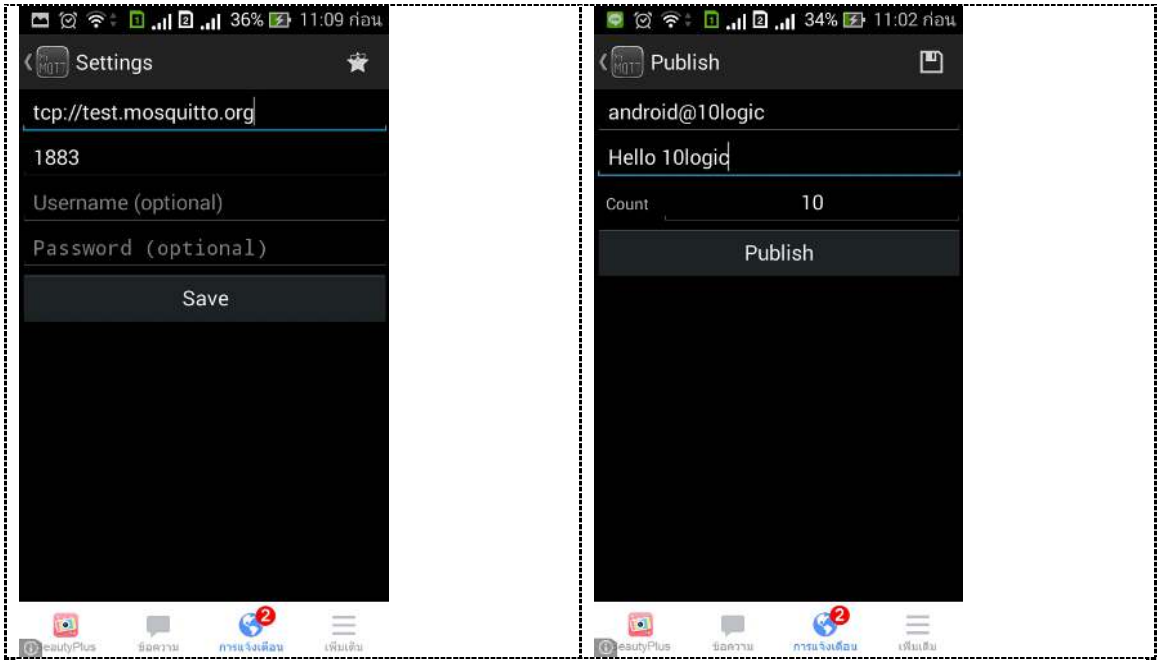
MyMQTT

เปิด MyMQTT และเข้าไปยังเมนู Settings





MyMQTT *MyMQTT*



Step3a/3: ทดสอบการทำงาน

MyMQTT

เมื่อผมใช้ Smart Phone ที่มีอยู่ทำการ Public ข้อความ “Hello 10logic” ผ่าน MyMQTT แสดงผลตามรูป

mqttlens

จะเห็นว่าสามารถรับข้อความ Hello 10logic ได้ตามตัวอย่างดังภาพ เห็นภาพกันแล้วใช่ไหมครับ ทีนี้เมื่อนักพัฒนาต้องการส่งข้อมูลจากอุปกรณ์ embedded สามารถส่งข้อมูลขึ้นมาได้เช่นกัน

1.3 IOT และ MQTT คือ อะไร?

MQTT ย่อมาจาก Message Queue Telemetry Transport เป็นโปรโตคอลประยุกต์ที่ใช้โปรโตคอล TCP เป็นรากฐาน ออกแบบมาสำหรับงานที่ต้องการ ๆ สื่อสารแบบเรียลไทม์แบบไม่จำกัดแพลตฟอร์ม หมายถึงอุปกรณ์ทุกชิ้นสามารถสื่อสารกันได้ผ่าน MQTT

MQTT จะแบ่งเป็น 2 ฝั่ง 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 หรือเปรียบได้กับแอดมินที่สามารถเข้าไปดูข้อความได้ทุดห้องแม้จะเป็นห้องลับก็ตาม
- 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 ที่ถูกเรียกว่าเส้นทาง เนื่องจากการใช้งานส่งข้อมูลและรับข้อมูลจะเหมือนกับเส้นทางในระบบไฟล์ 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)

การส่งข้อมลในแต่ละครั้งจะต้องประกอบไปด้วยเส้นทาง)Topic) ข้อมูล และคุณภาพข้อมูล ซึ่งการส่งข้อมูลจะเรียกว่า Publish

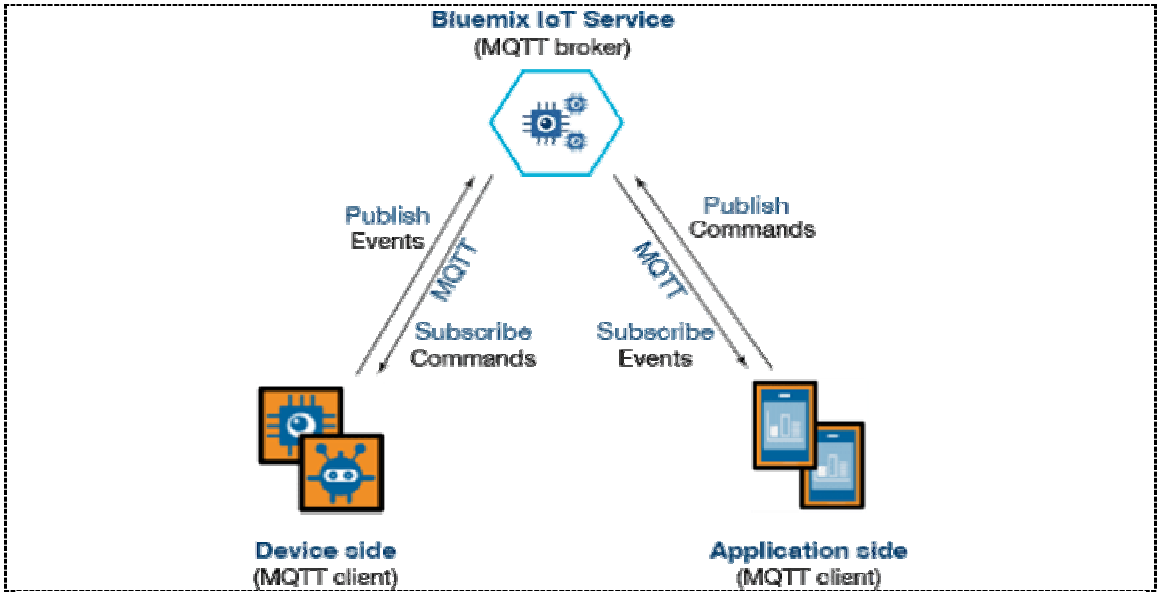
การรับข้อมูล (Subscribe)

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

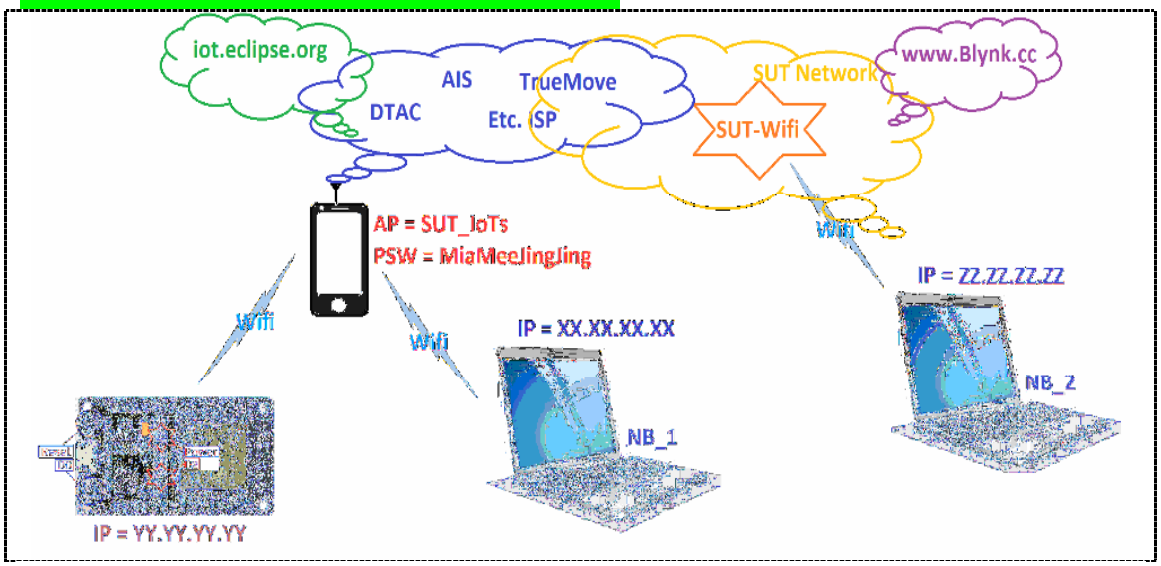
จะเห็นได้ว่า MQTT ก็เปลี่ยนเสมือนห้องแชทของอุปกรณ์ที่จะสนทนาแลกเปลี่ยนข้อมูลกันแบบเรียลไทม์ผ่านเครือข่ายอินเตอร์เน็ต

1.4 IoT มีวิธีการทำงานอย่างไร?

องค์ประกอบหลักของ IoT จะมี 3 ส่วนคือ **Broker**, **Publisher** และ **Subscriber**. ซึ่งการรับและส่งข้อมูลนั้นมันจะส่งข้อมูลไปมาหากันนั้นจะส่งผ่านตัวกลางนั้นก็คือ Broker Server โดยตัวส่งนี้จะเรียกว่า Publisher ส่งข้อมูลขึ้นไปยัง Broker พร้อมระบุหัวข้อ)Topic) ที่ต้องการส่งข้อออกไป จากนั้นตัวรับซึ่งเรียกว่า Subscriber ถ้าหากตัวรับต้องการรับข้อมูลจากตัวส่งจะต้องทาการ Subscribe หัวข้อ Topic ของ Publisher นั้นๆ ผ่าน Broker เช่นกัน ลองดูความสัมพันธ์ ตามรูป



1.5 –ข้อแตกต่างระหว่าง IoT กับ Over Internet



การทดลองก่อนหน้าเป็นการควบคุมผ่านอินเตอร์เน็ต จำเป็นต้องรู้ IP ของอุปกรณ์ปลายทางและระบบต้องอยู่ในวงเครือข่ายเดียวกัน 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 กระบวนการข้างต้นจะปรับใหม่ คือ ไม่จำเป็นต้องรู้เลขไอพีของอุปกรณ์ปลายทางแต่ให้อุปกรณ์วิ่งไปรับคำสั่งที่ตัวกลาง (Broker) แทน จากรูป NB_2 จะส่งคำสั่ง (Publish) ไปยังตัวกลาง ตัวอุปกรณ์ปลายทางต้องแจ้งรับข้อความ (Subscribe) จากตัวกลางเมื่อมีคำสั่งเข้ามาและตัวอุปกรณ์ปลายทางเข้ามารับข้อมูลอุปกรณ์ปลายทางค่อยทำงานตามคำสั่งที่ได้รับ

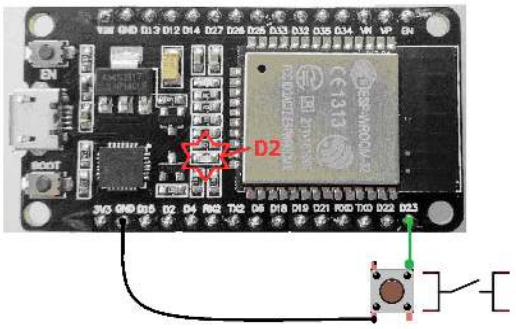
เห็นได้ว่าแบบแรกจำเป็นต้องเข้าให้ถึง ESP32 แต่แบบหลังใช้วิธีนัดรับข้อความที่ตัวกลางที่ทั้งสองฝั่งตกลงกันไว้

2a. การทดลอง

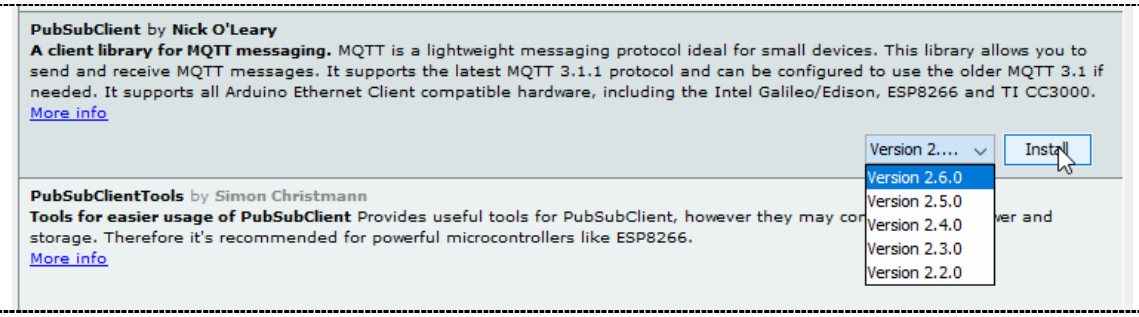
การทดลองที่ 1/4 Publish

1. ทดสอบโปรแกรมนับจำนวนครั้งการกดสวิตซ์ แสดงผลที่ Serial Monitor ด้วย baud rate 115200

```
int pushButton = 23;
int Counter = 0;

void setup() {
  Serial.begin(115200);
  pinMode(pushButton, INPUT_PULLUP);
  Serial.print(" Counter = ");
  Serial.println(Counter);
}

void loop() {
  if (digitalRead(pushButton) == 0)
  { Counter++;
    Serial.print(" Counter = ");
    Serial.println(Counter);
    while (digitalRead(pushButton) == 0);
    delay(100);
  }
}
```



2. Add Library → การใช้งาน MQTT บน ESP32 จะใช้งานผ่านไลบารี่ PubSubClient.h จะต้องติดตั้งเพิ่มเติมโดยใช้ Library Manager ค้นหาคำว่า PubSubClient เลือก PubSubClient Version 2.6.0 แล้วสามารถกดปุ่ม Install เพื่อติดตั้ง



3. โปรแกรมจะ Publish ไปยัง **iot.eclipse.org** ในหัวข้อ **myHome1234**

- Test Broker = iot.eclipse.org
- Port = 1883
- Topic, Subscribe, Publish = **myHome1234**

4. โปรแกรมทดสอบ Publish ไปยัง **iot.eclipse.org** ในหัวข้อ **myHome1234**

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

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
const char* mqtt_server = "iot.eclipse.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect()
{ while (!client.connected())              // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32 Client-";
    clientId += String(random(0xffff), HEX);  // Create a random client ID
    if (client.connect(clientId.c_str()))     // Attempt to connect
    { Serial.println("connected");           // Once connected, publish an announcement...
      client.publish(topic1, "Hello World Pk007");      // ... and resubscribe
      client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup()
{ Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

void loop()
{ if (!client.connected())  reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 2000)
  { lastMsg = now;
    ++value;
    snprintf (msg, 75, "hello world #%ld", value);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
  }
}
```

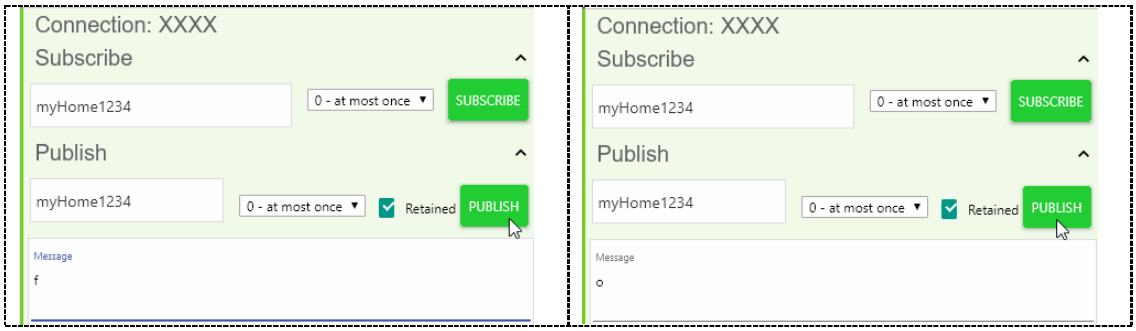
5. ทดสอบโปรแกรมนับจำนวนครั้งการกดสวิตซ์ ส่งค่าไปยัง MQTT Broker

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

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
const char* mqtt_server = "iot.eclipse.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];

int pushButton = 23;
int Counter = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");       Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("");                   Serial.println("WiFi connected");
  Serial.println("IP address: ");       Serial.println(WiFi.localIP());
}

void reconnect()
{ while (!client.connected())                          // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32 Client-";
    clientId += String(random(0xffff), HEX);           // Create a random client ID
    if (client.connect(clientId.c_str()))              // Attempt to connect
    { Serial.println("connected");                     // Once connected, publish an announcement...
      client.publish(topic1, "Hello World Pk007");     // ... and resubscribe
      client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup()
{ Serial.begin(115200);
  pinMode(pushButton, INPUT_PULLUP);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

void loop()
{ if (digitalRead(pushButton) == 0)
  { Counter++;
    if (!client.connected())  reconnect();
    client.loop();
    snprintf (msg, 75, "Counter =  #%ld", Counter);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
    while (digitalRead(pushButton) == 0);
    delay(100);
  }
}
```

Q-A1. ปรับแก้โปรแกรมให้อ่านค่า DHT22 แล้วส่งไปยัง MQTT Broker ทุกๆ 2 วินาที

การทดลองที่ 2/6 Publish and Subscribe

6. เมื่อโหลดโปรแกรมแล้วทำการทดสอบ publish “f”` ➔ Off LED และ “o” ➔ On LED



7. โปรแกรมการ Publish และ Subscribe หน่วงเวลาครั้งละ 2 วินาที

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

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
const char* mqtt_server = "iot.eclipse.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int TestLED = 2;
int value = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);     Serial.print(".");
  }
  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  pinMode(TestLED, OUTPUT);
}

void callback(char* topic, byte* payload, unsigned int length)
{ Serial.print("Message arrived [");
  Serial.print(topic1);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
  }
  Serial.println();
  Serial.print(" ---> ");
  Serial.println((char)payload[0]);
  if ((char)payload[0] == 'o')  digitalWrite(TestLED, HIGH);
  if ((char)payload[0] == 'f')  digitalWrite(TestLED, LOW);
}

void reconnect()
{ while (!client.connected())              // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);  // Create a random client ID
    if (client.connect(clientId.c_str()))     // Attempt to connect
```

```
    { Serial.println("connected");          // Once connected, publish an announcement...
     client.publish(topic1, "Hello World Pk007");     // ... and resubscribe
     client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
     Serial.print(client.state());
     Serial.println(" try again in 5 seconds");
     delay(5000);
    }
  }
}

void setup()
{ Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(TestLED, OUTPUT);
}

void loop()
{ if (!client.connected())  reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 2000)
  { lastMsg = now;
    ++value;
    snprintf (msg, 75, "hello world #%ld", value);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
  }
}
```

8. ทดสอบโปรแกรมนับจำนวนครั้งการกดสวิตช์ ส่งค่าไปยัง MQTT Broker และควบคุมการปิด-เปิด BUILD IN LED

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

const char* ssid = "SUT_IoTs";
const char* password = "MaiMeeJingJing";
const char* mqtt_server = "iot.eclipse.org";
const char* topic1 = "myHome1234";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];

int TestLED = 2;
int pushButton = 23;
int Counter = 0;

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
void callback(char* topic, byte* payload, unsigned int length)
{ Serial.print("Message arrived [");
  Serial.print(topic1);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  { Serial.print((char)payload[i]);
  }
  Serial.println();
  Serial.print(" ---> ");
  Serial.println((char)payload[0]);
  if ((char)payload[0] == 'o')  digitalWrite(TestLED, HIGH);
  if ((char)payload[0] == 'f')  digitalWrite(TestLED, LOW);
}

void reconnect()
{ while (!client.connected())              // Loop until we're reconnected
  { Serial.print("Attempting MQTT connection...");
    String clientId = "ESP32 Client-";
    clientId += String(random(0xffff), HEX);  // Create a random client ID
    if (client.connect(clientId.c_str()))     // Attempt to connect
    { Serial.println("connected");           // Once connected, publish an announcement...
      client.publish(topic1, "Hello World Pk007");      // ... and resubscribe
      client.subscribe(topic1);
    } else
    { Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup()
{ Serial.begin(115200);
  pinMode(pushButton, INPUT_PULLUP);
  pinMode(TestLED, OUTPUT);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop()
{ if (!client.connected())  reconnect();
  client.loop();
  if (digitalRead(pushButton) == 0)
  { Counter++;
    client.loop();
    snprintf (msg, 75, "Counter =  #%ld", Counter);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topic1, msg);
    while (digitalRead(pushButton) == 0);
    delay(100);
  }
}
```

Q-A2. อ่านค่า DHT-22 แล้วส่งไปยัง MQTT Broker ทุกๆ 2 วินาที และควบคุมการปิด-เปิด BUILD IN LED

PCเลขที่ _____รหัส_____ ชื่อ-สกุล _____

3. คำถามท้ายการทดลอง- ให้เขียนโปรแกรมเพื่อทำงานต่อไปนี้

Q-A1. อ่านค่า DHT-11 แล้วส่งไปยัง MQTT Broker ทุกๆ 2 วินาที

Q-A2. อ่านค่า DHT-11 แล้วส่งไปยัง MQTT Broker ทุกๆ 2 วินาที และควบคุม BUILD IN LED