

Week11 – MCS51: Hardware Design, ESP32: Grafana and Influx DB, Robot: SUMO Robot

PC 28 รหัส B6401887 ชื่อ-สกุล นันท์นภัส ศรีมาตรภิรมย์

- จัดข้อมูลใน Word ให้เรียบร้อยสวยงาม
 - ให้เป็นคะແນນ໌ອງ Class 5 ຄະແນນ (5 ຄະແນນຕັດເກຣດ)
 - Font ເດີວກັນທີ່ຈຳບັບຮາຍງານ, ກາຣລຳດັບເລຂ້າຂ້ອຕ້ອງຕ່ອນເນື່ອງກັນ
 - ໄກສໍເໜືອນກັນກັບເພື່ອນ ຈະຖຸກຫາຮະແນນທາມຈຳນວນທີ່ໜ້າກັນ
- ທຳກາຣທດລອງແລະທຳຮາຍງານຂັ້ນຕອນກາຣທດລອງ – ESP32 to Google Spreadsheet
- ທຳກາຣທດລອງແລະທຳຮາຍງານຂັ້ນຕອນກາຣທດລອງ – ESP32 to Influx DB with Grafana Dashboard Display
- ສ່າງຈານທີ່ໜຳດັກອນ 0600-20231013 ປີ <https://forms.gle/YZXeevkv9XB8MYGr9>

Read More

- <https://grafana.com/blog/2021/03/08/how-i-built-a-monitoring-system-for-my-avocado-plant-with-arduino-and-grafana-cloud/>
- <https://www.metricfire.com/blog/iot-dashboards-with-grafana-and-prometheus/>
- <https://gabrieltanner.org/blog/grafana-sensor-visualization>

1/3 ให้จัดเรียบเรียงข้อมูล

1a/3: ESP8266 / ESP32 & Mesh Network

- ESP Mesh Network 1 - <https://meetjoeblog.com/2018/03/25/esp8266-esp32-mesh-network-ep1/>

ตอนที่ 1: Mesh Network กับ Introduction to Painlessmesh โดยใช้ ESP8266 / ESP32

Mesh Network คือโครงสร้างของเครือข่ายที่มีลักษณะเป็น Node หลายๆ ตัวที่เชื่อมต่อถึงกันและทำงานร่วมกันในการส่งผ่านข้อมูลจาก Node หนึ่งไปยัง Node อีกด้วย โดยไม่จำเป็นต้องต่อสายแบบต่อกันแบบตัวต่อตัว นี่คือคุณสมบัติหลักของ Mesh Network ที่ช่วยให้เครือข่ายเป็นไปตามรูปแบบของการติดตั้งและการเชื่อมต่อ อุปกรณ์ได้อย่างอิสระและเจ้งที่เมื่อขั้นอุปกรณ์ Home Automation อย่าง Zigbee และ Z-Wave ที่ใช้เครือข่าย Mesh Network.

คุณสมบัติสำคัญของ Mesh Network รวมถึงการทำงานแบบ Self-Organize และ Self-Configure ซึ่งหมายความว่าเครือข่าย Mesh Network สามารถปรับเปลี่ยนโครงสร้างได้ตลอดเวลา โดยตัวอุปกรณ์สามารถคุยกันเพื่อหาเส้นทางที่เหมาะสมในการส่งข้อมูลจากจุด A ไปยังจุด B แม้ว่าเส้นทางเดิมจาก A->B->C->D อาจต้องเปลี่ยนเมื่อ Node B ล่ม ในกรณีนี้เครือข่าย Mesh Network จะหาเส้นทางแทนที่เพื่อให้ข้อมูลถึงปลายทางได้.

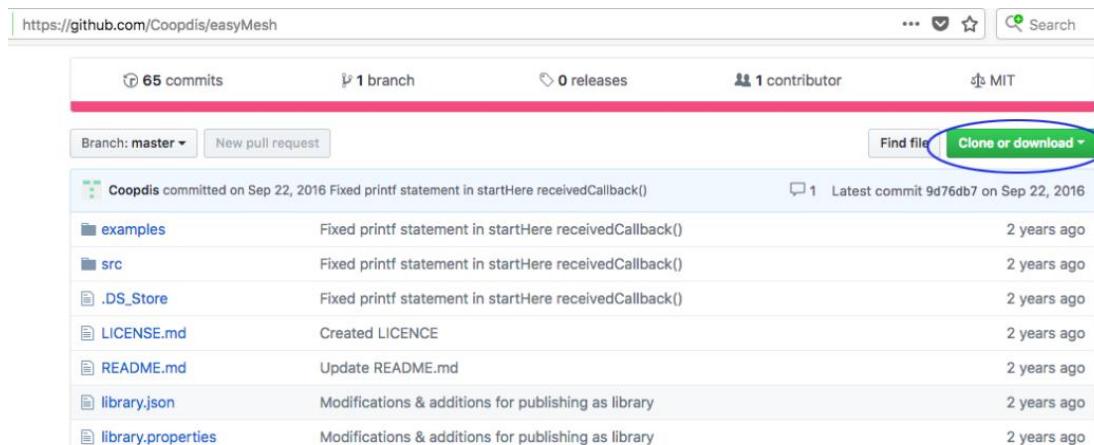
สำหรับการทำงานแบบ Mesh Network ในบ้านหรือสำหรับโปรเจกต์ IoT อย่าง ESP8266 และ ESP32 จาก Espressif, มีการพัฒนา ESP-Mesh Protocol ซึ่งหมายความว่าเราสามารถใช้งานในระบบ Mesh Network. อย่างไรก็ตาม, การใช้งาน ESP-Mesh Protocol อาจจะมีความซับซ้อนกว่าการใช้งานแบบ传统协议 แต่เราพบว่า มีหลายไลบรารีที่ช่วยในการสร้าง Mesh Network โดยง่ายและไม่ซับซ้อน เช่น EasyMesh และ Painlessmesh.

Painlessmesh เป็นไลบรารีที่พัฒนาขึ้นจาก EasyMesh และสามารถใช้งานได้ทั้งกับ ESP8266 และ ESP32 โดยที่การใช้งานมีความเข้าใจและไม่ซับซ้อนมาก ทำให้หมายความว่าเราสามารถสร้าง Mesh Network ในโปรเจกต์ IoT และเราสามารถเริ่มต้นกับ ESP8266 หรือ ESP32 ของเราได้.

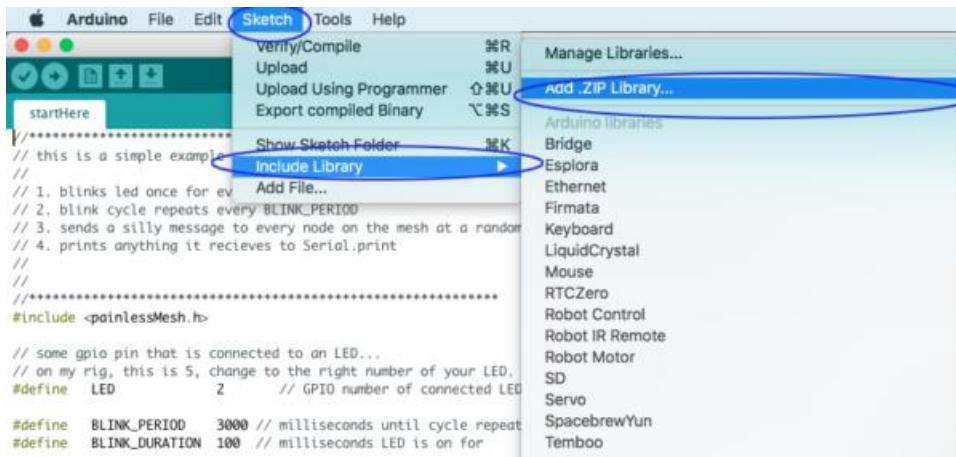
หลังจากการศึกษาเนื้อหาข้างต้น เราสามารถเริ่มต้นการสร้าง Mesh Network โดยใช้ ESP8266 และ ESP32 โดยการใช้ Painlessmesh หรือไลบรารีอื่น ๆ ที่ช่วยในการสร้างระบบเครือข่าย Mesh Network ในโปรเจกต์ IoT ของเรา. การสร้าง Mesh Network นี้จะช่วยให้เราควบคุมและเชื่อมต่ออุปกรณ์ได้อย่างอิสระและเจ้งที่ในโปรเจกต์ของเรา

เตรียมความพร้อม

หลังจากได้ติดตั้ง board esp32 และ esp8266 ไว้เรียบร้อยแล้ว ขั้นแรกก็ Download/Clone Library จาก Github มาไว้ที่เครื่องก่อน โดยเข้าไปที่ Painlessmesh ที่ Github



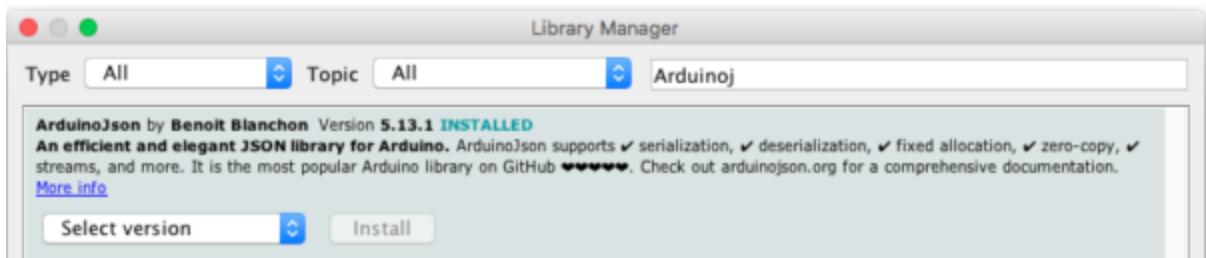
จากนั้นก็ติดตั้งลงใน Arduino IDE โดยเข้าไปที่ Sketch -> Include Library -> Add .zip library และเลือก zip file ที่เรา Download มา



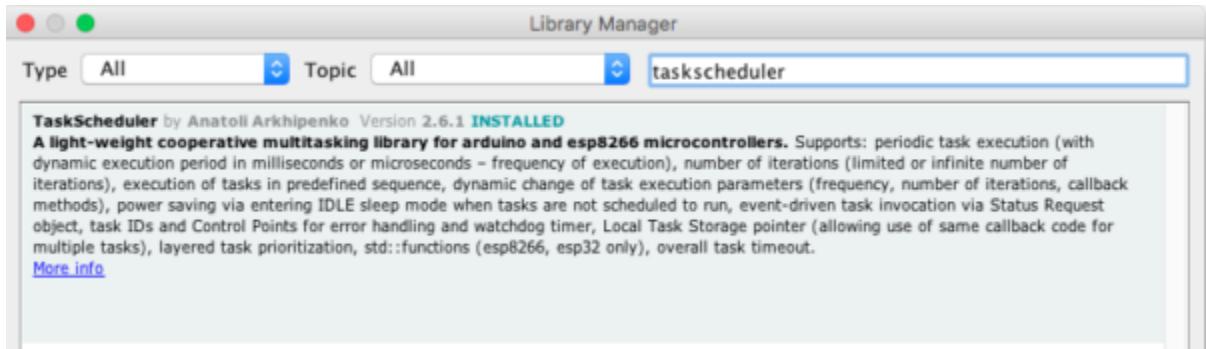
ติดตั้ง Library Dependencies ซึ่งจำเป็นต่อการใช้ Painlessmesh

ก่อนการใช้งานเรายังต้องใช้ Library อีกสองตัวนั้นคือ ArduinoJson และ TaskScheduler ซึ่งการ Install Library ก็สามารถทำได้ไม่ยากเลย สำหรับสองตัวนี้ เข้าไปที่ Sketch -> Include Library -> Manage Libraries จากนั้นในช่อง Filter Your Search ก็สามารถใส่ชื่อของ Library ทั้งสองตัวนี้ได้เลย

ArduinoJson



TaskScheduler



สิ่งที่ต้องรู้และข้อจำกัด

หากผู้พัฒนา Painlessmesh เค้าเคลมเอาไว้ว่า library ชุดนี้นั้นเป็น True Adhoc Networking นั่นก็คือไม่ต้องมีการออกแบบ node ให้_node_ เชื่อมต่อ node ให้_node_ โกรงสร้างของ mesh จะเป็นยังไง ไม่ต้องมี router ศูนย์กลาง ซึ่งแค่ 2 node ก็สามารถทำงานได้แล้ว ทำให้คนพัฒนาไม่ต้องมาอยุ่งวุ่นวายในเรื่องของการจัดการกับ Self-Organize / Self-Configure ไปสนใจในส่วนของ Business Logic / Operation Logic ว่าจะใช้งาน Mesh นื้อย่างไรดีกว่า

ซึ่งคำถามของคนใช้ที่มักจะถามบ่อยๆ เรียกได้ว่าเป็น FAQ ของ Painlessmesh เลยก็ว่าได้นั้นก็คือ จำนวน Maximum Node ที่รองรับ โดยที่ทางทีมผู้พัฒนา ก็บอกไว้ว่า

“The maximum size of the mesh is limited (we think) by the amount of memory in the heap that can be allocated to the sub-connections buffer and so should be really quite high.”

นั้นก็คือ ขึ้นอยู่กับ memory ของแต่ละ node นั้นเอง เพราะมันจะต้องใช้ในการเก็บสถานะของแต่ละ node เพื่อที่จะได้ทำ routing config ต่างๆ ฉะนั้นถ้าแต่ละ Node อ่านค่าจาก Sensor ตามรอบเวลาที่กำหนด แล้วส่งผ่าน Mesh Network ไปเก็บไว้ที่ Server จำนวน memory ที่ใช้ก็ไม่น่ามาก ซึ่งเดียวเราจะมาดูกันว่า memory นั้นจะถูกใช้ไปเท่าไหร่เมื่อเพิ่มนодูเข้าไปใน Mesh Network

non-ip networking: ในระบบ Network เราสามารถคุ้นชินกับระบบ IP หรืออย่างน้อยก็ mac address แต่ว่า Painlessmesh นั้นไม่ใช้ tcp/ip ใน การสื่อสารและไม่ได้อ้างอิงโดยใช้ ip หรือ mac address แต่จะใช้ chipid ซึ่งก็เป็นหมายเลขเฉพาะในแต่ละชิปของ esp8266/esp32 โดยใน library จะถูกเรียกว่า nodeid แทน

JSON Based: painlessmesh ใช้ระบบการส่งผ่านข้อมูลในรูปแบบของ JSON ถ้าใครไม่คุ้นก็อาจจะงงๆหน่อย แต่รับรองไม่ยากแน่นอน ซึ่งข้อดีของมันก็คือ Node ที่รับข้อมูลแล้วต้องการส่งต่อไปยัง webserver หรือระบบงานอื่นที่รองรับ json อยู่แล้วก็จะง่ายเลยที่เดียว

Delay: เว็บเข้าสู่เครื่องข้อจำกัด ข้อจำกัดแรกเลยก็คือเรื่องของ delay ซึ่งผู้พัฒนา Painlessmesh เลือกแนะนำให้หลีกเลี่ยง เพราะภายในตัว library เองจะมีวงรอบในการคุ้นชินกับ node ต่างๆ เพื่ออัพเดทสถานะของ mesh ตลอดเวลา node ไหนอยู่ node ไหนหลุดไป ถ้าเราไม่การใช้งาน delay ภายใน loop อาจทำให้ mesh ของเราไม่เสถียร ซึ่งทางแก้เค้าก็แนะนำให้ใช้ task scheduler แทน

อัตราการส่งข้อมูล: ทางผู้พัฒนาเค้าแนะนำให้คิดคำนวณอัตราการส่งข้อมูลแบบ conservative อะหน่อย ต้องดูให้เหมาะสมกับ application ด้วย เช่นถ้าจะ monitor temp/humid ในдинแต่ส่งข้อมูลทุก 5 วินาทีก็อาจจะได้ไม่ค่อยสมเหตุสมผลเท่าไหร่ เพราะจะทำให้ตัว esp8266/esp32 ที่มีข้อจำกัดเรื่องของการประมวลผลและหน่วยความจำอาจต้องทำงานหนักขึ้น

Message Dropped: ถึงแม้ระบบ Mesh Network มันจะเป็นอะไรที่เจ้มากจนดูเหมือนเป็น fault tolerance / high availability network แต่ก็มีโอกาสทำงานผิดพลาดได้ ฉะนั้น message ที่ส่งผ่านก็อาจจะหายไปได้บ้างในบางโอกาส ฉะนั้นก็ขึ้นอยู่กับ logic ของ application ที่เราพัฒนา ว่าจะ handle ยังไง

ไอ้ข้อกำหนดข้างบนหนะ ผูกก็ไม่ได้เก่งเขียนขึ้นมาเองหรอกครับ ทีมพัฒนา Painlessmesh เนี่ยแหละแนะนำมา ผู้ว่าก็ยังได้ไว้เป็น best practice บ้างก็ได้ครับ เวลาจะหาข้อผิดพลาดจะได้มีตัววนหาตั้งแต่ศูนย์ จะได้มาเข้าคิวได้ว่าເວີ້ທີ່ມັນສ่งข้อมูลໄໝຝ່ານເປັນພຣະອ່າຣ໌ memory overflow ມັຍ หรือເປັນປຸ່ງຫາຈາກກາຍກາພທີ່ໄໝ join เข้า mesh หรือວ່າເຮົາມີເຂົ້າມີ delay ໄວ

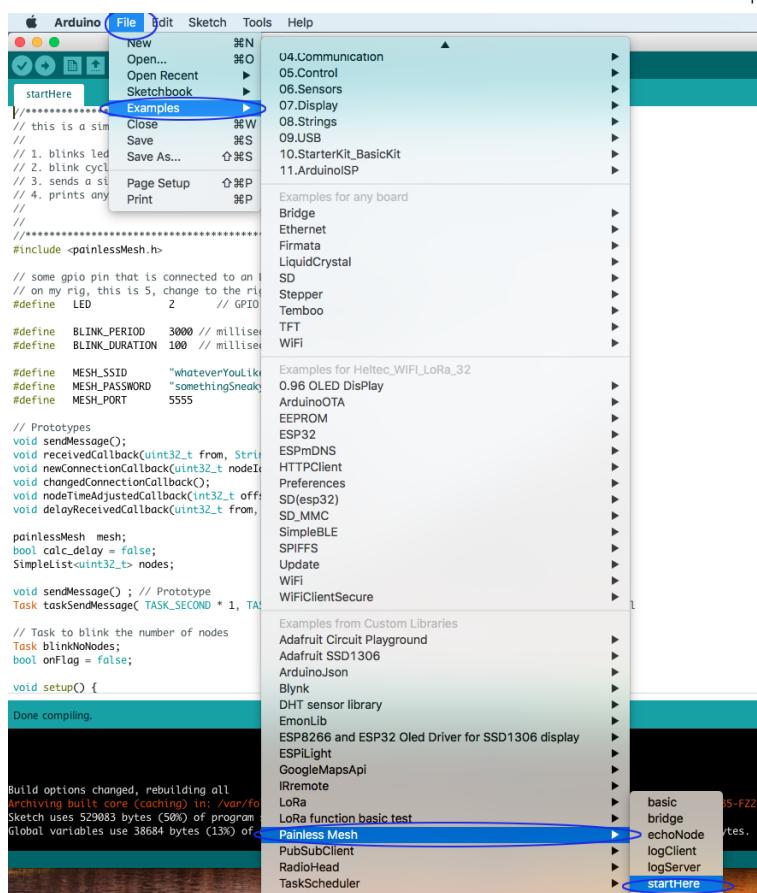
Hello Mesh Network

หลังจากเตรียม environment ในการทำงานเรียบร้อยแล้วเรา ก็จะมาเริ่มทดลองสร้าง Mesh Network แรกของเรางานนี้ขอเจ้า เขียนร่ายมาซั้ง ยาว เริ่มซัก 5 node ก่อนลงกัน โดยที่ Node ของเราจะประกอบด้วย



Wemos D1 Mini, ESP32 Wroom, ESP-01, แล้วก็ Nodemcu อีกสองตัว

ขั้นแรกเลยก็เปิดไฟล์ startHere กันก่อนเลยครับ ที่ Arduino IDE ก็เข้าไปที่ File->Examples->Painless Mesh-> startHere



มาดูกำตั้งค่าคร่าวๆ ของ Mesh Network เรา กัน

```
#define MESH_SSID "HelloMyMesh"
#define MESH_PASSWORD "hellomymeshnetwork"
#define MESH_PORT 5555
```

จากไฟล์ startHere ที่มากับตัวอย่าง เราสามารถที่จะແຍກງของ Mesh รวมถึงเพิ่ม Security ในส่วนของ Mesh Network เราได้จาก 3 บรรทัด
บนนี้เลยครับ ซึ่ง help ของ painlessmesh รวมถึงคำอธิบายที่อยู่ใน code นั้นค่อนข้างดีเลยที่เดียว ซึ่งการทำงาน Mesh Network โดย^{ใช้} Painlessmesh นั้นจะแบ่งการทำงานเป็นลักษณะของ task โดยการกำหนดจาก task scheduler อย่างเช่น task ที่ใช้ในการส่ง message
mesh.scheduler.addTask(taskSendMessage);

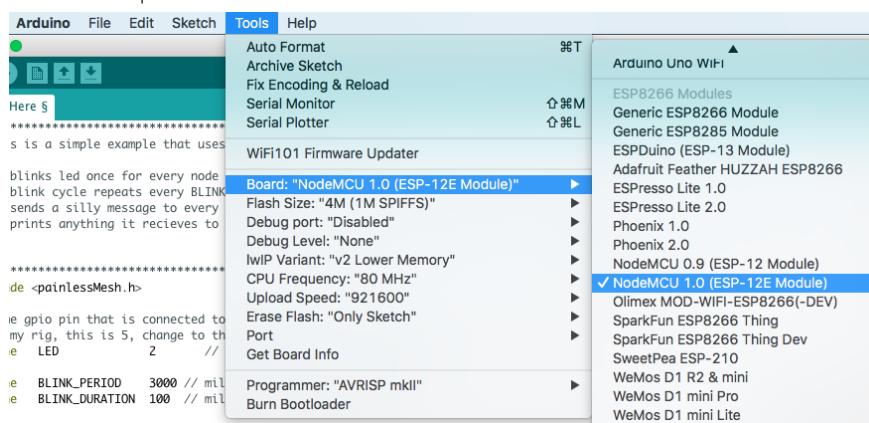
```
taskSendMessage.enable();
```

และในส่วนของ loop ก็จะมีแค่ การ update mesh เป็นหลัก พร้อม logic ต่างๆจะไปอยู่ที่ task scheduler หมวด mesh.update();

ในส่วนของ CallBack นั้นหลักที่ใช้งานจริงๆเลยก็คือ receivedCallback เพื่อที่จะดูว่า message ที่เราได้รับมา ไม่ว่าจะมาจาก Broadcast หรือมาแบบระบุ NodeID นั้นจะนำไปทำอะไรต่อซึ่งในตัวอย่างนี้ก็คือ Print ออกมาผ่านทาง Serial Port

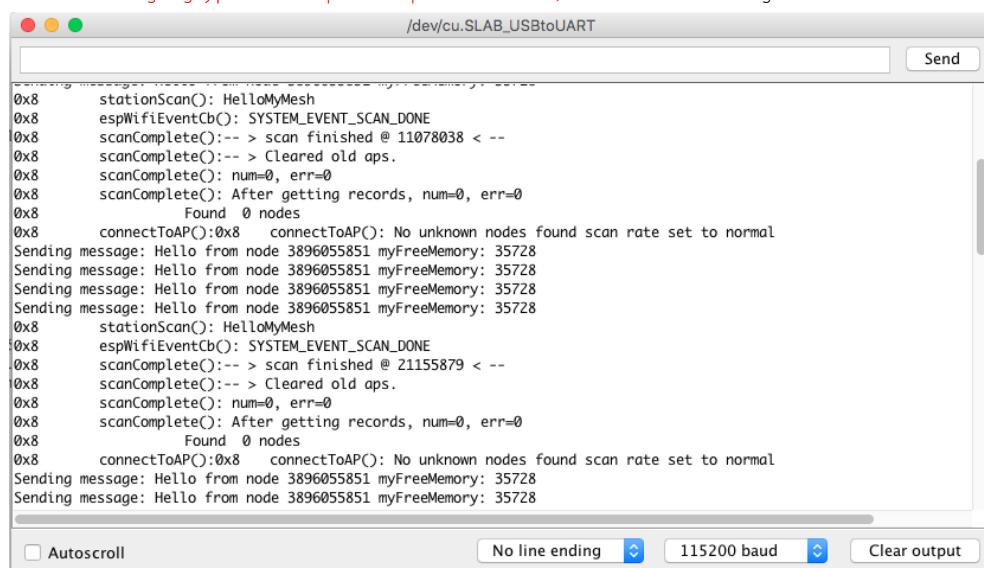
```
void receivedCallback(uint32_t from, String & msg) {
    Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}
```

ซึ่งที่ผมแก้หลักๆเลยก็คือตัว SSID และ Password แค่นั้นจากนั้นก็ทำการ Flash เลยครับ เลือก Board ให้ตรงรุ่นแค่นั้น



```
// 1. blinks led once for every node on the mesh
// 2. blink cycle repeats every BLINK_PERIOD
// 3. sends a silly message to every node on the mesh at a random time between 1 and 5 seconds
// 4. prints anything it receives to Serial.print
```

เมื่อทำการ flash nodemcu ตัวแรกไปแล้วเปิดที่ Serial Monitor ดูข้อมูลมันก็จะ酵ะๆหน่อย เพราะมีการ enable debug ไว้ในตอน setup
mesh.setDebugMsgTypes(ERROR | DEBUG | CONNECTION); ซึ่งจะตบของการ debug ก็มีหลายแบบให้ลองเลือกใช้งานตามที่เหมาะสม



ซึ่งเมื่อ flash เสร็จและเริ่มทำงาน nodemcu ตัวแรกในวง Mesh Network (มีตัวเดียวจะเรียก Mesh ทำนายยย) ก็จะเริ่มหา AP ที่เชื่อมต่อ กับบ้านใกล้ๆเรือนเคียงของ node อื่นๆๆ เพื่อที่จะตั้งมาเข้าสู่ Mesh นี้โดยข้อมูลจากการ Debug ก็จะได้ nodeid รวมถึง memory ที่เหลืออยู่ล่ะ

คราวนี้เราflash nodemcu อีกสองตัวเข้าไปดูบ้างว่าผลลัพธ์ที่ได้เป็นอย่างไร (note ไว้ก่อนว่า nodeid ของตัวแรกคือ 5851 เอาแค่ 4 ตัวสุดท้ายพอจะได้ไม่ต้องจำเยอะ และ memory ที่เหลือคือ 35728 byte)

```

0x00      .const = "HelloMyMesh", _ZL11HelloMyMesh
0x8      Found 2 nodes
0x8      connectToAP():0x8      connectToAP(): No unknown nodes found scan rate set to normal
startHere: Received from 3896664414 msg=Hello from node 3896664414 myFreeMemory: 31880
startHere: Received from 3896055851 msg=Hello from node 3896055851 myFreeMemory: 33872
Sending message: Hello from node 3893419003 myFreeMemory: 27088
startHere: Received from 3896664414 msg=Hello from node 3896664414 myFreeMemory: 31880
startHere: Received from 3896055851 msg=Hello from node 3896055851 myFreeMemory: 33872
startHere: Received from 3896664414 msg=Hello from node 3896664414 myFreeMemory: 31208
startHere: Received from 3896055851 msg=Hello from node 3896055851 myFreeMemory: 33872
Sending message: Hello from node 3893419003 myFreeMemory: 27088
startHere: Received from 3896664414 msg=Hello from node 3896664414 myFreeMemory: 31880
startScan(): HelloMyMesh
0x8      espWiFiEventCb(): SYSTEM_EVENT_SCAN_DONE
0x8      scanComplete():-- > scan finished @ 260926004 < --
0x8      scanComplete():-- > Cleared old ops.
0x8      scanComplete(): num=2, err=0
0x8      scanComplete(): After getting records, num=2, err=0
0x8      found : HelloMyMesh, -29dBm
0x8      found : HelloMyMesh, -15dBm
0x8      Found 2 nodes
0x8      connectToAP():0x8      connectToAP(): No unknown nodes found scan rate set to normal
startHere: Received from 3896055851 msg=Hello from node 3896055851 myFreeMemory: 33872

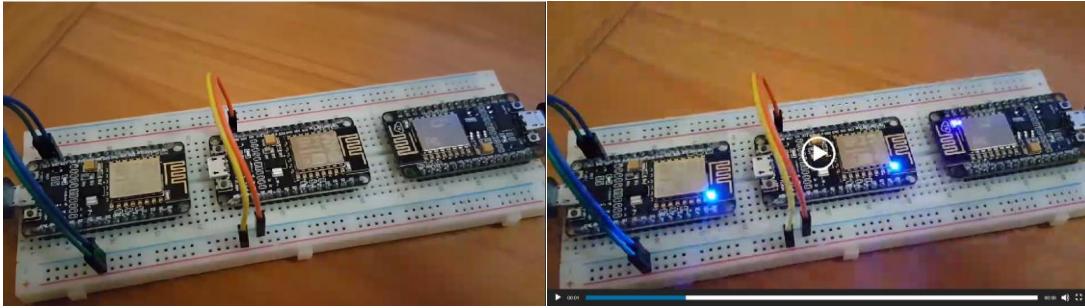
```

คราวนี้เรามี Node ที่อยู่ใน Mesh นี้ 3 ตัวลงนั่นคือ 5851, 4414 และ 9003 ซึ่งแต่ละตัวก็จะ Broadcast Message ออกมานานาเวลา ที่ถูกใจได้ในแต่ละตัว ดำเนินขั้นตอนการกระพริบไฟ Led นั้นจะมีการตั้ง time synchronize เมื่อมี node ใหม่เพิ่มเข้ามาใน mesh ซึ่งจะมี CallBack ไว้วางรับและทำงานร่วมกับ MeshUpdate ที่วน loop หัวว่ามี Node ไหน drop connection ไปหรือ มี node ไหน join เข้ามาและมีค่าความแรงของสัญญาณทำให้หรือ เพื่อใช้ในการจัดสียนทางในการส่งข้อมูล

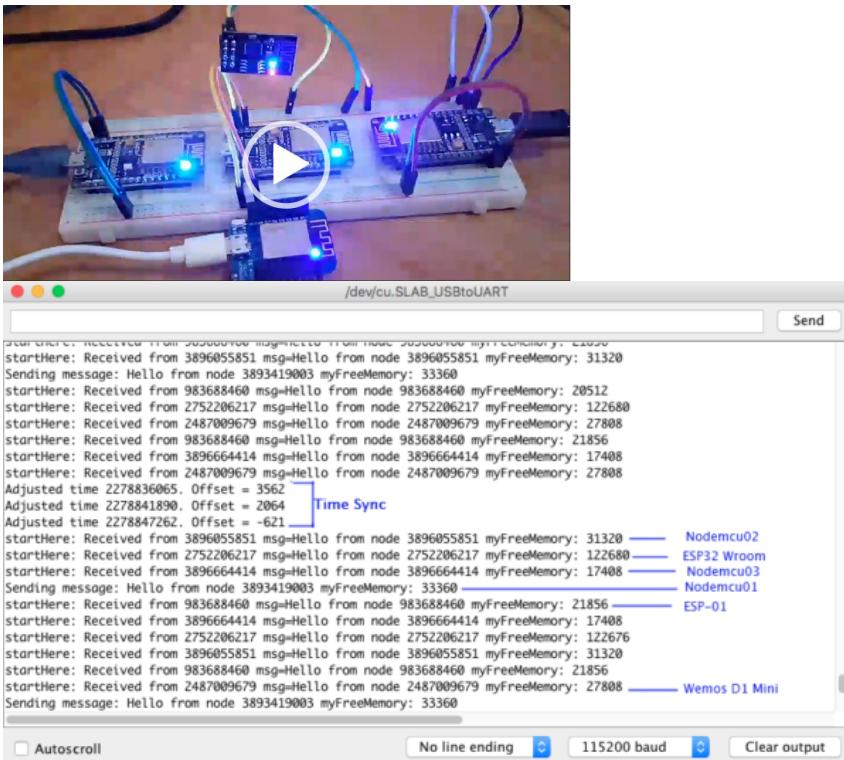
```

0x00      .const = "HelloMyMesh", _ZL11HelloMyMesh
0x8      espWiFiEventCb(): SYSTEM_EVENT_AP_STACONNECTED
startHere: Received from 3896664414 msg=Hello from node 3896664414 myFreeMemory: 28656
0x8      stationScan(): HelloMyMesh
0x8      espWiFiEventCb(): SYSTEM_EVENT_SCAN_DONE
0x8      scanComplete():-- > scan finished @ 583618487 < --
0x8      scanComplete():-- > Cleared old ops.
0x8      scanComplete(): num=1, err=0
0x8      scanComplete(): After getting records, num=1, err=0
0x8      found : HelloMyMesh, -28dBm
0x8      Found 1 nodes
0x8      connectToAP():0x8      connectToAP(): No unknown nodes found scan rate set to normal
startHere: Received from 3896055851 msg=Hello from node 3896055851 myFreeMemory: 33336
Sending message: Hello from node 3893419003 myFreeMemory: 27872
Changed connections [{"nodeId":3896664414, "subs":[]}]
Num nodes: 2
Connection list: 3896664414 3896055851
Sending message: Hello from node 3893419003 myFreeMemory: 27856
Delay to node 3896664414 is 8120 us
Delay to node 3896055851 is 12122 us
startHere: Received from 3896055851 msg=Hello from node 3896055851 myFreeMemory: 33824
startHere: Received from 3896664414 msg=Hello from node 3896664414 myFreeMemory: 31872
Sending message: Hello from node 3893419003 myFreeMemory: 27856
startHere: Received from 3896055851 msg=Hello from node 3896055851 myFreeMemory: 33808

```



แค่ 3 Node ยังดูน้อยไป เรามาเพิ่ม Node เข้าไปใน Mesh Network เราต้องอีกตึกกว่า ถ้าดูจากตอนนี้ 3 Node Memory ของ Node ที่ memory เหลือน้อยสุดจะอยู่ที่ 27856 Byte เท่านั้น โดยผู้จะเพิ่ม ESP32 Wroom, Wemos D1 Mini และ ESP-01 เข้าไปแล้วมาดูว่า Memory จะเหลือกันมากน้อยขนาดไหนในเมื่อขนาดของ Mesh Network โตขึ้น (Built-in LED pin ของ ESP-01 อยู่ที่ Pin 1)

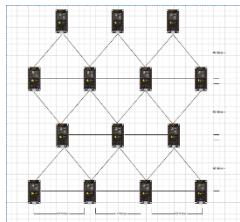


จากรูปด้านบนเมื่อทำงานไปได้ซักพักถ้า Memory ที่เหลืออยู่ต่ำสุด จะเป็นของ Nodemcu03 ซึ่งเหลืออยู่ 17408 Byte ส่วนที่เหลือมากสุด ก็เดินไม่ยากเลย ย่อมเป็น ESP32 MCU ตัวใหม่แน่นอนอยู่แล้ว ดังที่ทางทีมผู้พัฒนาค่าแจ้งเอาไว้ว่าตัว mesh network นั้นจะรองรับได้กี่ node นั้นอยู่ที่ memory เป็นหลักเลย ไว้มีเวลาตอนที่ 2 จะเพิ่ม node เข้ามาใน mesh network พ้อกับเพิ่มการส่งข้อมูลที่อ่านได้จากเซนเซอร์ต่างๆ แล้วมาดูกันว่า ขนาดที่เพิ่มขึ้นพร้อมกับ operation logic ที่เพิ่มขึ้น จะทำให้โครงข่าย mesh นี้เปลี่ยนไปยังไงกันบ้าง รับรองว่าตอนต่อไปจะได้นำ mesh network มาใช้งานในการส่งข้อมูลกันจริงๆ และ ส่วนตอนต่อๆไปอีกจะมีอีกเรื่องก็ลิสต์ขึ้นมาให้ดูกัน เรียกแขกกันซะหน่อย

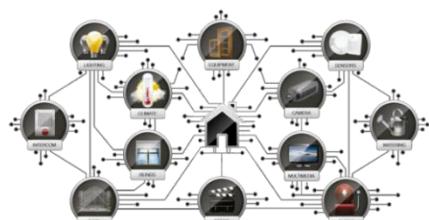
- ESP Mesh Network 2 - <https://meetjoeblog.com/2018/03/27/esp8266-esp32-mesh-network-painlessmesh-client-server-ep2/>

ESP8266 / ESP32 & Mesh Network ตอนที่ 2: ภาคต่อของ Painlessmesh Client/Server

High Availability / Fault Tolerance



ขยายความจากตอนเดิม Mesh Network นั้นขึ้นต่อไปเรื่อยๆ ของความเสถียร สะดวก แต่ไม่ต้อง setup อะไรเลย ถ้าดูด้วยตาจะเห็นว่ามี Node ใด Node หนึ่งร่วงไปหรืออาจจะมากกว่านี้ตัวใดได้ สัมภានในการส่งข้อมูลก็ยังคงไม่ถูกตัดขาด เพราะด้วยคุณสมบัติของ Self-Organize / Self-Configure ก็จะทำให้สามารถส่งข้อมูลไปยังปลายทาง หรือ Node ที่ยัง Active อยู่ได้ โดยไม่ต้องมีอุปกรณ์เป็นศูนย์กลางเหมือนระบบ WiFi ที่มีการทำงานแบบ Star ซึ่งถ้า WiFi Router หรือ AP ล่ม ทุก Node



Coverage Area

จากรูปด้านบน ถ้าคิด “เฉลี่ย” ที่ hop ละ 40 เมตร นี่คิดแบบ Conservative สุดๆไปเลยนะ เพียงแค่ 14 Node ก็สามารถทำให้มีการสื่อสารครอบคลุมพื้นที่ได้ถึง 14,400 ตารางเมตร หรือ 9 ไร่กันเลยทีเดียว ถูกกว่าซื้อ AP หรือ Repeter มาใช้จะอีกถูกกว่า การทำงานของ Mesh Network นั้นมักจะเป็นการใช้งานแบบ local network อย่างที่เราเห็นกันในระบบ home automation หรือแม้กระทั่ง smart meter บางรุ่น ฉะนั้นถ้าจะทำให้ Home

Automation หรือ Local Mesh Network ที่เราสร้างขึ้นมาต้องส่งข้อมูลออกไปข้างนอกได้ ก็จะมีอีก Node หนึ่งขึ้นมาเพื่อเป็นสะพาน (Bridge) และใช้เป็นประตู (Gateway) ในการ forward ข้อมูลหรือใช้ในการสื่อสารกับ Network ภายนอก ในระบบ Home Automation ที่ใช้ Zigbee หรือ Z-Wave หรือบางระบบซึ่งรองรับการสื่อสารทั้งสองแบบก็จะมีอุปกรณ์ที่เรียกว่า Gateway ไว้เป็นตัว Bridge ระหว่าง Local Mesh Network กับการสื่อสารผ่านโครงข่ายข้างนอก ไม่ว่าจะเป็น WiFi, Internet หรือแม้กระทั่งข้ามไปมาระหว่าง Zigbee กับ Z-Wave ซึ่งเดียวเราจะมาต่อ กันในบทที่ 3 เรื่อง Bridge ระหว่าง Mesh Network กับ Network ภายนอกกัน

Power Consumption

ถ้าจุดที่เราติดตั้งไฟฟ้าใช้งานตลอดเวลา หรือมีความสะดวกสบายในการเดินสายไฟ ประเด็นเรื่อง Power Consumption อาจจะไม่ต้องคิดมาก นัก แต่ในระบบ home automation มักจะมีอุปกรณ์ประเภทที่ติดตั้งไว้โดยเดียว หรือตอนที่บ้านสร้างเสร็จแล้ว อาจไม่สะดวกในเรื่องของการเดินสายไฟ ซึ่งอุปกรณ์เหล่านี้ก็จะเป็นพวง

PIR ไสวตรวจจับความเคลื่อนไหวต่างๆ

Leak Sensor ไสวว่ามีน้ำรั่วใต้ชั้นล่างงาน หรือใต้หลังคาบ้าน

IR Remote ซึ่งทำให้เราสามารถที่จะควบคุมอุปกรณ์ต่างๆ เช่น TV และ พัดลม

ที่ยกตัวอย่างไป ถ้าบ้านที่สร้างไว้แล้ว และต้องมาเดินสายไฟ อาจไม่สวยงามนัก Zigbee กับ Z-wave จึงมาตอบโจทย์ตรงนี้ ในการเชื่อมต่อ กับ เน็ตเวิร์ก Mesh Network ไม่ต้องเดินสาย แต่กินไฟต่ำ ใส่ถ่าน A23 ไปสองก้อนอยู่ได้เป็นปี ทำให้สะดวกมากๆ แต่ อุปกรณ์เหล่านี้ก็มีราคาแพง ถ้าเทียบกับ ESP8266/ESP32

ซึ่งถ้าเราจะใช้ ESP8266/ESP32 แล้วหลังก็ สิ่งหนึ่งที่ต้องคำนึงถึงเลยก็คือเรื่องของการจ่ายไฟให้ Node เหล่านี้ทำงานอยู่ตลอดเวลา ไม่สามารถที่จะ Sleep เพื่อที่จะประหยัดไฟได้ เรา มาลองถูกกันว่า ในสถานการณ์ต่างๆ เจ้า nodemcu ของเรากินกระแสมากน้อยขนาดไหน ซึ่งลองวัดกันแบบหยาบๆ ผ่าน usb tester กัน ซึ่งถ้าใครมี Oscilloscope หรือ Multi-meter อยู่ใกล้ตัวก็จะรู้ดีล่ะเอียงดู



Scenario1: rom เปล่าๆ วนลูปไปเรื่อย ค่าที่ได้ 80mA

Scenario2: ต่อ WiFi ทิ้งไว้ตามปกติ ค่าที่ได้อよู่ที่ 80mA

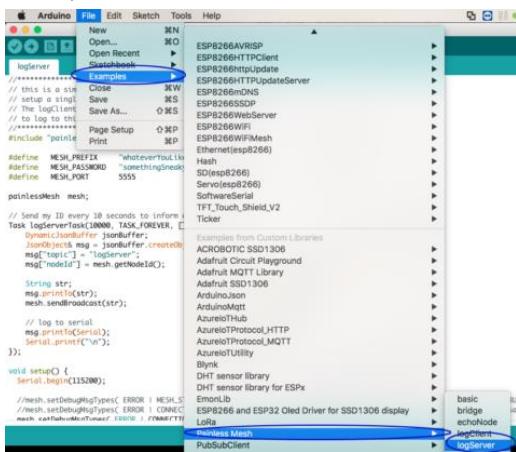
Scenario3: Join Mesh Network (ใช้ Painlessmesh) 2-6 Nodes Broadcast Message ค่าที่สังเกตได้อยู่ที่ประมาณ 80-90mA มีค่า Peak อยู่ที่ประมาณ 150mA ซึ่งน่าจะมาจากตอน Led กระพริบและตอนที่ Join/Update Mesh Network ซึ่งคาดว่าใช้งานจริงเพิ่ม Node เข้าไป น่าจะทำให้อัตราการกินกระแสสูงขึ้น ถ้าอย่างให้กระแต่ถ้าต้องการให้มีการเชื่อมต่อ

ฉะนั้นถ้าจะเอา ESP8266/ESP32 มาทำ Mesh Network เพื่อรับส่งข้อมูล หรือ Control อุปกรณ์ต่างๆแล้ว น่าจะต้องคำนึงในเรื่องของภาคจ่ายไฟด้วย ในโรงงานหรืออาคารที่ไม่มีปัญหาเรื่องการจ่ายไฟ น่าจะเหมาะสม ส่วนงานประเภท Outdoor / Open Field อาจต้องพิจารณาในเรื่องของการนำพาไฟไว้ให้เหมาะสมสมได้ (อย่าลืมวัดจากการใช้งานใน application จริงด้วย เพราะเซนเซอร์บางอย่างกินกระแสไฟหนักมาก)

ในตอนที่ผ่านมาได้รับมาตัวเดียวที่ผ่านมาอ่านแล้วได้ทดลองทำตามด้วยอย่างไฟไป อาจเริ่มตัวอย่างที่สองในส่วนของการทำงานลักษณะที่เป็น Client/Server ไปแล้วก็ได้ เพราะตัวอย่างที่ให้นั้นค่อนข้างดีเลยที่เดียว แต่ถ้าใครยังไม่ได้รีบ ก็มาเริ่มพร้อมๆกันเลยครับ เกริ่นซะยะว่าอีกแล้ว

เริ่มใช้งาน Painlessmesh แบบ Client/Server

ตัวอย่างประกอบสำหรับการใช้งานแบบ Client/Server จะใช้อ่ายุ่งด้วยกันคือ logServer และ logClient โดยสามารถเลือกเปิดได้จาก File->Examples->Painless Mesh-> logServer หรือ logClient



```
#define MESH_PREFIX      "HelloMyMesh"
#define MESH_PASSWORD   "hellomymeshnetwork"
```

```
#define MESH_PORT      5555
```

โดยเราจะเริ่มที่ logServer กันก่อนครับ ซึ่ง concept ก็ยังคงเหมือนเดิม ให้แก้ไขในส่วนของการตั้งค่า Mesh Network ของเราระดับชั้นที่ 3 บรรทัดต้นบนนี้ครับ หรือใครจะคงเดิมไว้เหมือนตัวอย่างที่ได้มาก็ได้แล้วก็ Flash ลง ESP8266/ESP32 ของเราได้เลย

แต่ในส่วนของคนที่ใช้ ESP32 ต้องแก้ไขนิดนึงครับ เพราะการ Declare ในส่วนของ Authentication Mode นั้นไม่เหมือนกัน แต่ ESP8266 จะแก้ให้เป็นแบบ ESP32 ก็สามารถใช้งานได้เหมือนกันครับ

```
mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, STA_AP, WIFI_AUTH_WPA2_PSK, 6 ); // <-- ต้องเพิ่มคำว่า WIFI
```

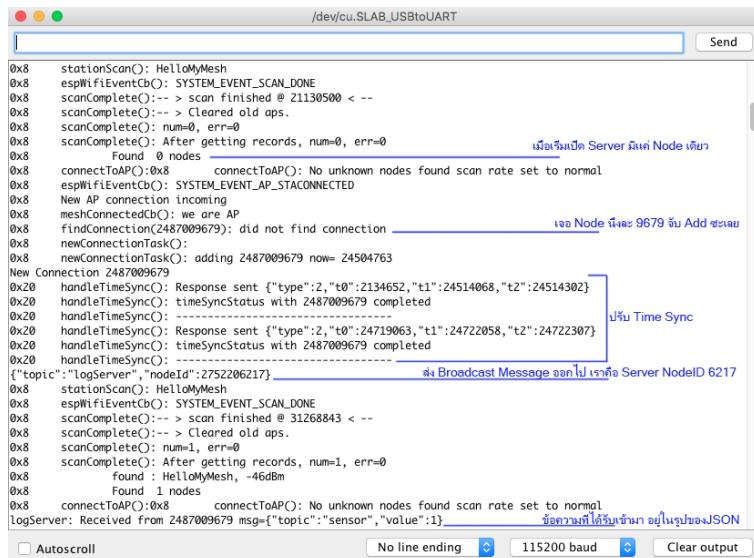
Concept ของตัวอย่าง logServer นั้นคล้ายๆกับตอนที่ 1 ครับเพียงแต่ว่า Broadcast Message ที่ส่งออกไปบันทึกประสมค์ในการส่งออกไปเพื่อบอกให้ node ที่อยู่ใน Mesh Network นี้รู้ว่า ฉัน NodeID นี้จะเป็น Server Node นะ ซึ่งก็จะทำงานด้วย task scheduler ทุกๆ 10 วินาทีนี้เมื่อมี node ใหม่เข้ามา join ใน Mesh Network ก็จะรู้แล้วว่าจะต้องส่งข้อมูลให้เครื่องรับแบบ Message ที่ส่งออกไปเก็บอยู่ในฟอร์แมทของ JSON ครับ โดยในส่วนของ receivedCallback ก็ยังเหมือนเดิมคือ ได้รับ message มาที่ Print ออกทาง Serial Port ไม่มีเปลี่ยนแปลง

```
Task logServerTask(10000, TASK_FOREVER, []){
    DynamicJsonBuffer jsonBuffer;
    JsonObject& msg = jsonBuffer.createObject();

    msg["topic"] = "logServer"; //กำหนด Topic เพื่อที่ Node ที่ join เข้ามาหาข้อมูลของ Server เจอก
    msg["nodeId"] = mesh.getNodeId(); // ใช้คำสั่ง mesh.getNodeID(); เพื่ออ่านค่า NodeID

    String str;
    msg.printTo(str);
    mesh.sendBroadcast(str); // ส่งข้อความ Broadcast ไปยังทุก Node ว่าฉันคือ Server Node นะ

    // log to serial
    msg.printTo(Serial);
    Serial.printf("\n");
}
```



หลังจาก Flash เสร็จการทำงานก็เป็นอย่างที่เห็นข้างบนครับ จะเริ่มทำการเช็คข้างเคียงว่ามี Node ไหนที่สามารถ Join ได้บ้าง จากนั้นก็ปรับ Offset Time Synchronize ให้เท่ากันเพื่อนๆในวงจร แล้วก็เข้าสู่ Task ที่ใช้ในการ Broadcast Message ออกไป่่าเราคือ Server เพื่อให้ Node ทุกด้วยที่อยู่ใน Mesh Network เดียวกันรู้ว่าจะส่งข้อมูลให้ Server ให้ส่งไปที่ NodeID หมายเลขอะไร ซึ่งในที่นี่คือ 6217

มาดูในส่วนของ logClient กันบ้าง ซึ่งในส่วนของ receivedCallback เนี่ยแหล่ะที่จะเปลี่ยนไป เพราะจะมีการเช็ค Broadcast Message จาก Node อื่นๆ ซึ่งคือ Server Node ว่า เห้ายย นี่แหล่ะ Server Node นั้น มี ID นี้นั่น จากนั้นก็จะส่งข้อความไปที่ ServerNode ให้ส่งไปที่ NodeID นี้นั่น

```
void receivedCallback( uint32_t from, String &msg ) {
    Serial.printf("logClient: Received from %u msg=%s\n", from, msg.c_str());
    // Saving logServer
    DynamicJsonBuffer jsonBuffer;
    JsonObject& root = jsonBuffer.parseObject(msg);
    if (root.containsKey("topic")) {
        if (String("logServer").equals(root["topic"].as<String>())) { //check Topic ว่าเป็น logServer หรือเปล่า
            // check for on: true or false
            logServerId = root["nodeId"]; //อ่านและบันทึกค่า nodeId ของ Server เพื่อใช้เป็นปลายทางในการส่ง
            Serial.printf("logServer detected!!!!\n");
        }
        Serial.printf("Handled from %u msg=%s\n", from, msg.c_str());
    }
}
```

สำหรับส่วนของ Task ที่ใช้ในการส่ง Message ก็จะทำงานล้อกัน คือถ้ายังไม่เจอกว่า ใน Mesh Network ที่เพิ่ง join เข้ามามี server หรือเปล่า ฉันก็ broadcast ข้อมูลของฉันเลยละกัน เมื่อจาก Server Node มีระยะเวลา 10 วินาทีถึงจะ Broadcast Message Topic “logserver” ออกไป ฉะนั้นการทำ เช่นนี้ Server Node ที่รับข้อมูลจาก Client Node ก็จะได้ข้อมูลແນ່ງๆ (รวมถึงตัวอื่นๆด้วย)

```
Task myLoggingTask(10000, TASK_FOREVER, []() {
    DynamicJsonBuffer jsonBuffer;
    JsonObject& msg = jsonBuffer.createObject();
    msg["topic"] = "sensor";
    msg["value"] = random(0, 180); //สำหรับไฟล์ตัวอย่างนี้ยังไม่ได้ต่อเข้ากับเซนเซอร์อะไร ก็สุ่มค่าส่งไปลักษณะ
    String str;
    msg.printTo(str);
    if (logServerId == 0) // If we don't know the logServer yet
        mesh.sendBroadcast(str);
    else
        mesh.sendSingle(logServerId, str);

    // log to serial
    msg.printTo(Serial);
    Serial.printf("\n");
```

});

```

/dev/cu.SLAB_USBtoUART
Send

0x2 AP tcp server established on port 5555
0x8 stationScan(): HelloMyMesh
{"topic":"sensor2","value":173}
0x8 espWifiEventCb(): SYSTEM_EVENT_SCAN_DONE
0x8 scanComplete():-- > scan finished @ 21204468 < --
0x8 scanComplete():-- > Cleared old aps.
0x8 scanComplete(): num=2, err=0
0x8 scanComplete(): After getting records, num=2, err=0
0x8     found : HelloMyMesh, -49dBm
0x8     found : HelloMyMesh, -46dBm
0x8     Found 2 nodes
0x8 findConnection(2752206217): did not find connection
0x8 findConnection(2487009679): did not find connection
0x8 connectToAP(): 0x8 connectToAP(): Best AP is 2487009679<---
0x8 connectToAP(): Trying to connect, scan rate set to 4*normal
0x8 espWifiEventCb(): SYSTEM_EVENT_STA_AUTHMODE_CHANGE
0x8 espWifiEventCb(): SYSTEM_EVENT_STA_CONNECTED
0x8 espWifiEventCb(): SYSTEM_EVENT_STA_GOT_IP
0x8 New STA connection incoming.
0x8 meshConnectedCb(): we are STA
0x8 findConnection(2487009679): did not find connection
0x8 newConnectionTask():
0x8 newConnectionTask(): adding 2487009679 now= 22385371
{"topic":"sensor2","value":63}
logClient: Received from 2752206217 msg={"topic":"logServer","nodeId":2752206217} Broadcast Message ที่ได้รับมา
logServer detected!!!
Handled from 2752206217 msg={"topic":"logServer","nodeId":2752206217} เมื่อเราต้องแค่ Topic Server ก็ทำการนัดกับ Serve NodeID
{"topic":"sensor2","value":2} คำที่ส่งออกไปในรูปแบบ JSON
logClient: Received from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
logServer detected!!!
Handled from 2752206217 msg={"topic":"logServer","nodeId":2752206217} คำที่ส่งออกไปในรูปแบบ JSON
{"topic":"sensor2","value":149} คำที่ส่งออกไปในรูปแบบ JSON

```

Autoscroll No line ending 115200 baud Clear output

คราวนี้แทนที่เราจะจำลองข้อมูลเหมือนในตัวอย่าง ก็ได้เวลาเปิดกระปองแรก เอา DHT22 มาพ่วงเข้าไป คราวนี้ Temp/Humid ที่ได้ก็จะเป็นค่าจริง แล้วโดยการปรับ Code ในส่วนของ Task ที่ใช้สำหรับการส่งข้อความ ซึ่งจะมีการอ่านค่าจาก DHT22 แล้วจัดรูปแบบให้อยู่ในฟอร์แมตของ JSON โดยที่ Mesh Network นี้จะประกอบไปด้วย 12 Node ด้วยกันดังนี้



- ESP32 Wroom ทำตัวเป็น Server
- ESP-01 เป็น Client ที่ส่งค่าสู่เซิร์ฟเวอร์
- Wemos D1 Mini 4 ตัว เป็น Client ที่ส่งค่าสู่เซิร์ฟเวอร์ (Wemos-01 ถึง Wemos-04)
- Wemos D1 Mini 1 ตัว เป็น Client ที่ส่งค่า Temp/Humid ที่อ่านได้จาก DHT22
- NodeMCU 4 ตัว เป็น Client ที่ส่งค่าสู่เซิร์ฟเวอร์ (mcu-01 ถึง mcu-04)
- NodeMCU 1 ตัว เป็น Client ที่ส่งค่า Temp/Humid ที่อ่านได้จาก DHT22

Code ที่ทำการแก้ไขสำหรับ Node ตัวที่ส่งค่า (ESP-01, Wemos-01 ถึง 04, mcu-01 ถึง 04)

```
// Send message to the logServer every 10 seconds

Task myLoggingTask(5000, TASK_FOREVER, []() {
    DynamicJsonBuffer jsonBuffer;
    JsonObject& msg = jsonBuffer.createObject();

    msg["nodename"] = "mcu3";           //เพิ่ม NodeName เข้ามาพอหลายๆตัวเริ่มงกับ เลข NodeID
    msg["NodeID"] = mesh.getNodeId();    //เพิ่ม NodeID เอาไว้อ้างอิง
    msg["random value"] = random(0, 180); //เปลี่ยนเป็น random value ให้ชัดเจนว่าเป็นค่าที่สุ่มขึ้นมา

    String str;
    msg.printTo(str);
    if (logServerId == 0) // If we don't know the logServer yet
        mesh.sendBroadcast(str);
    else
        mesh.sendSingle(logServerId, str);

    // log to serial
    msg.printTo(Serial);
    Serial.printf("\n");
});
```

Code ที่ทำการแก้ไขสำหรับ Node ตัวที่ส่งค่า Temp/Humid จาก DHT22 (Wemos-t1 และ mcu-t1)

```
#include "painlessMesh.h"
#include "DHT.h"

#define DHTPIN D4
#define DHTTYPE DHT22

#define MESH_PREFIX "HelloMyMesh"
#define MESH_PASSWORD "hellomymeshnetwork"
#define MESH_PORT 5555

DHT dht(DHTPIN, DHTTYPE);

void receivedCallback( uint32_t from, String &msg );

painlessMesh mesh;

size_t logServerId = 0;

// Send message to the logServer every 5 seconds
```

```
Task myLoggingTask(5000, TASK_FOREVER, []() {  
  
    float h = dht.readHumidity();  
  
    float t = dht.readTemperature();  
  
    DynamicJsonBuffer jsonBuffer;  
    JsonObject& msg = jsonBuffer.createObject();  
    msg["nodename"] = "Wemos-t1";  
    msg["NodeID"] = mesh.getNodeId();  
    msg["Temp"] = String(t)+"C";  
    msg["Humidity"] = String(h)+"%";  
  
    String str;  
    msg.printTo(str);  
    if (logServerId == 0) // If we don't know the logServer yet  
        mesh.sendBroadcast(str);  
    else  
        mesh.sendSingle(logServerId, str);  
  
    // log to serial  
    msg.printTo(Serial);  
    Serial.printf("\n");  
});  
  
void setup() {  
  
    Serial.begin(115200);  
    Serial.println("Begin DHT22 Mesh Network test!");  
  
    dht.begin();  
  
    mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION ); // set before init() so that you can see startup messages  
  
    mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, STA_AP, AUTH_WPA2_PSK, 6 );  
    mesh.onReceive(&receivedCallback);  
  
    // Add the task to the mesh scheduler  
    mesh.scheduler.addTask(myLoggingTask);  
    myLoggingTask.enable();
```

```

}

void loop() {
    // put your main code here, to run repeatedly:
    mesh.update();
}

void receivedCallback( uint32_t from, String &msg ) {
    Serial.printf("logClient: Received from %u msg=%s\n", from, msg.c_str());

    // Saving logServer
    DynamicJsonBuffer jsonBuffer;
    JsonObject& root = jsonBuffer.parseObject(msg);
    if (root.containsKey("topic")) {
        if (String("logServer").equals(root["topic"].as<String>())) {
            // check for on: true or false
            logServerId = root["nodeId"];
            Serial.printf("logServer detected!!!\n");
        }
        Serial.printf("Handled from %u msg=%s\n", from, msg.c_str());
    }
}

```

หลังจาก Flash ครบตัวสุดท้ายก็เปิด Serial Monitor ขึ้นมาจะเห็นได้จากรูปข้างล่างเลยครับว่า เจอ Node ข้างเคียงอยู่ทั้งหมด 10 Node พร้อมกับความแรงของสัญญาณจากแต่ละ Node ซึ่งจะใช้ในการจัดการการเชื่อมต่อของ Mesh Network ของเรา

```

/dev/cu.SLAB_USBtoUART
Send

0x8   scanComplete():--> scan finished @ 5750896 <--
0x8   scanComplete():--> Cleared old ops.
0x8   scanComplete(): num=10, err=0
0x8   scanComplete(): After getting records, num=10, err=0
0x8     found : HelloMyMesh, -41dBm
0x8     found : HelloMyMesh, -42dBm
0x8     found : HelloMyMesh, -35dBm
0x8     found : HelloMyMesh, -37dBm
0x8     found : HelloMyMesh, -50dBm
0x8     found : HelloMyMesh, -43dBm
0x8     found : HelloMyMesh, -61dBm
0x8     found : HelloMyMesh, -59dBm
0x8     found : HelloMyMesh, -56dBm
0x8     found : HelloMyMesh, -57dBm
0x8   Found 10 nodes
0x8   findConnection(983688460): did not find connection
0x8   findConnection(2488971084): did not find connection
0x8   findConnection(2487009679): did not find connection
0x8   findConnection(2136600749): did not find connection
0x8   findConnection(3163409689): did not find connection
0x8   findConnection(572025877): did not find connection
0x8   findConnection(3893419003): did not find connection
0x8   findConnection(3896713776): did not find connection
0x8   findConnection(3163408206): did not find connection
0x8   findConnection(3896055851): did not find connection
0x8   connectToAP():0x8  connectToAP(): Best AP is 2487009679<---

Autoscroll  No line ending  115200 baud  Clear output

```

อย่างที่ได้บอกไปข้างต้นครับ เมื่อเริ่มการทำงาน Server จะ Broadcast Message เพื่อบอกเพื่อนๆใน Mesh Network นี้ว่า NodeID ของ Server นั้นคือหมายเลขอะไร เพื่อที่ว่าเมื่อมา接รู้ว่า Server NodeID คือหมายเลขอะไร Client Node ก็จะ Broadcast Message ออกไปเพื่อแจ้งไปก่อน อย่างน้อยมันก็ต้องเข้า

Server Node แหล่ง ควรวนิรบามาจำลองทดสอบที่ว่ามีกันดู ระหว่างที่ยังไม่เจอ Server Node และเมื่อ Server Node Online และทุกๆ Node รับทราบกัน หมวดแล้วและส่ง Message ออกไปแบบระบุ NodeID

Scenario ที่ 1: No Server Node

ตามรูปด้านล่างนี้เลียครับ ข้อมูลที่ได้รับผ่าน receivedCallback ได้รับข้อมูลจาก Node ที่ Online ใน Mesh Network นี้ทั้งหมด เพราะว่ายังไม่ได้เปิด Server

```
/dev/cu.SLAB_USBtoUART
[REDACTED] mcu-04 ต้องกันตอนนี้ และกำลังจะรับข้อมูล
[REDACTED] ข้อมูลที่รับมาจากทั้ง 10 Node
[REDACTED]
{"nodename": "mcu-04", "NodeID": 3896664414, "random value": 87}
logClient: Received from 3163409689 msg={"nodename": "Wemos-02", "NodeID": 3163409689, "random value": 95}
logClient: Received from 3896055851 msg={"nodename": "mcu-02", "NodeID": 3896055851, "random value": 55}
logClient: Received from 3896713776 msg={"nodename": "mcu-01", "NodeID": 3896713776, "random value": 156}
logClient: Received from 3163408206 msg={"nodename": "Wemos-04", "NodeID": 3163408206, "random value": 75}
logClient: Received from 3893419003 msg={"nodename": "mcu-03", "NodeID": 3893419003, "random value": 2}
logClient: Received from 572025877 msg={"nodename": "Wemos-01", "NodeID": 572025877, "random value": 148}
logClient: Received from 2487009679 msg={"nodename": "Wemos-t1", "NodeID": 2487009679, "Temp": "25.10C", "Humidity": "44.70%"}
logClient: Received from 2136600749 msg={"nodename": "Wemos-03", "NodeID": 2136600749, "random value": 16}
logClient: Received from 2488971084 msg={"nodename": "mcu-t1", "NodeID": 2488971084, "Temp": "25.90C", "Humidity": "42.10%"}
logClient: Received from 983688460 msg={"nodename": "esp-01", "NodeID": 983688460, "random value": 96}
{"nodename": "mcu-04", "NodeID": 3896664414, "random value": 142}
logClient: Received from 3163409689 msg={"nodename": "Wemos-02", "NodeID": 3163409689, "random value": 149}
logClient: Received from 3896055851 msg={"nodename": "mcu-02", "NodeID": 3896055851, "random value": 37}
logClient: Received from 3896713776 msg={"nodename": "mcu-01", "NodeID": 3896713776, "random value": 133}
logClient: Received from 3163408206 msg={"nodename": "Wemos-04", "NodeID": 3163408206, "random value": 176}
logClient: Received from 3893419003 msg={"nodename": "mcu-03", "NodeID": 3893419003, "random value": 132}
logClient: Received from 572025877 msg={"nodename": "Wemos-01", "NodeID": 572025877, "random value": 91}
logClient: Received from 2136600749 msg={"nodename": "Wemos-03", "NodeID": 2136600749, "random value": 13}
logClient: Received from 2487009679 msg={"nodename": "Wemos-t1", "NodeID": 2487009679, "Temp": "25.40C", "Humidity": "44.80%"}
logClient: Received from 2488971084 msg={"nodename": "mcu-t1", "NodeID": 2488971084, "Temp": "25.90C", "Humidity": "42.10%"}
logClient: Received from 983688460 msg={"nodename": "esp-01", "NodeID": 983688460, "random value": 106}
{"nodename": "mcu-04", "NodeID": 3896664414, "random value": 26}
logClient: Received from 3163409689 msg={"nodename": "Wemos-02", "NodeID": 3163409689, "random value": 27}
logClient: Received from 3896055851 msg={"nodename": "mcu-02", "NodeID": 3896055851, "random value": 61}
```

Autoscroll No line ending 115200 baud Clear output

Scenario ที่ 2: Server Node join Mesh Network

เมื่อเปิด Server Node เข้ามายังจะเป็นเหมือนรูปข้างล่างนี้ครับ ได้รับ Broadcast Message จาก Server Node และในส่วนของ receivedCallback ก็จะมี logic เพื่อใช้ในการอ่านค่าว่ามี Topic ที่เป็น logServer มั้ย ถ้ามีก็บันทึกเก็บไว้ เพื่อในการส่งรอบต่อไปจะไม่ Broadcast ไปยังทุก Node ละ แต่จะระบุปลายทางเป็น NodeID ของ Server แทน

```
/dev/cu.SLAB_USBtoUART
[REDACTED] Send
[REDACTED]
0x8 espWiFiEventCb(): SYSTEM_EVENT_AP_STACONNECTED
0x8 New AP connection incoming
0x8 meshConnectedCb(): we are AP
0x8 findConnection(2752206217): did not find connection
0x8 newConnectionTask():
0x8 newConnectionTask(): adding 2752206217 now= 389582447
logClient: Received from 3893419003 msg={"nodename": "mcu-03", "NodeID": 3893419003, "random value": 163}
logClient: Received from 572025877 msg={"nodename": "Wemos-01", "NodeID": 572025877, "random value": 1}
logClient: Received from 2136600749 msg={"nodename": "Wemos-03", "NodeID": 2136600749, "random value": 49}
logClient: Received from 2487009679 msg={"nodename": "Wemos-t1", "NodeID": 2487009679, "Temp": "24.10C", "Humidity": "38.10%"}
logClient: Received from 983688460 msg={"nodename": "esp-01", "NodeID": 983688460, "random value": 20}
{"nodename": "mcu-04", "NodeID": 3896664414, "random value": 77}
logClient: Received from 3163409689 msg={"nodename": "Wemos-02", "NodeID": 3163409689, "random value": 30}
logClient: Received from 3896055851 msg={"nodename": "mcu-02", "NodeID": 3896055851, "random value": 170}
logClient: Received from 3896713776 msg={"nodename": "mcu-01", "NodeID": 3896713776, "random value": 128}
logClient: Received from 2488971084 msg={"nodename": "mcu-t1", "NodeID": 2488971084, "Temp": "24.60C", "Humidity": "36.60%"}
logClient: Received from 3163408206 msg={"nodename": "Wemos-04", "NodeID": 3163408206, "random value": 37}
logClient: Received from 3893419003 msg={"nodename": "mcu-03", "NodeID": 3893419003, "random value": 174}
logClient: Received from 572025877 msg={"nodename": "Wemos-01", "NodeID": 572025877, "random value": 8}
logClient: Received from 2136600749 msg={"nodename": "Wemos-03", "NodeID": 2136600749, "random value": 162}
logClient: Received from 2752206217 msg={"topic": "logServer", "nodeId": 2752206217}
logServer detected!!!
Handled from 2752206217 msg={"topic": "logServer", "nodeId": 2752206217}
{"nodename": "mcu-04", "NodeID": 3896664414, "random value": 81}
```

Autoscroll No line ending 115200 baud Clear output

หลังจากที่ทุก Node ได้รับ Server NodeID จาก Broadcast Message ที่ส่งมาจาก Server Node แล้ว แทนที่ทุก Node จะส่งข้อมูลออกไปยังทุกๆ Node ก็จะส่งไปยัง Server Node แทนดังจะเห็นได้จากรูปข้างล่างที่ได้รับ Broadcast Message เข้ามายังมีเฉพาะข้อความที่มาจากการ Server Node เท่านั้น เพราะตัวอื่นๆ ส่งไปที่ Server Node หมวดแล้ว

```

logServer detected!!!
Handled from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
{"nodename":"mcu-04","NodeID":3896664414,"random value":123}
{"nodename":"mcu-04","NodeID":3896664414,"random value":62}
logClient: Received from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
logServer detected!!!
Handled from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
{"nodename":"mcu-04","NodeID":3896664414,"random value":124}
{"nodename":"mcu-04","NodeID":3896664414,"random value":76}
logClient: Received from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
logServer detected!!!
Handled from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
{"nodename":"mcu-04","NodeID":3896664414,"random value":150}
{"nodename":"mcu-04","NodeID":3896664414,"random value":55}
logClient: Received from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
logServer detected!!!
Handled from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
{"nodename":"mcu-04","NodeID":3896664414,"random value":105}
{"nodename":"mcu-04","NodeID":3896664414,"random value":88}
logClient: Received from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
logServer detected!!!
Handled from 2752206217 msg={"topic":"logServer","nodeId":2752206217}
{"nodename":"mcu-04","NodeID":3896664414,"random value":24}
{"nodename":"mcu-04","NodeID":3896664414,"random value":46}
 Autoscroll No line ending 115200 baud Clear output

```

คราวนี้เรามาลองอุปในฝั่งของ Server กันบ้างว่าเมื่อทุก Node ส่งข้อมูลมาที่ Server Node แล้วข้อมูลที่ได้จะเป็นลักษณะไหนกัน จากรูปจะเห็นได้ว่า ข้อมูลจะได้ครบเมื่อตอนที่เราดูของ Node ใด Node หนึ่งตอนที่ยังไม่มี Server Node เนื่องจากคราวนี้ทุก Node ใน Mesh Network ส่งมาที่จุดเดียว ก็เลยจะได้ข้อมูลครบ และตัว Server Node เองก็ยังคง Broadcast Message ออกไปเรื่อยๆเพื่อให้ทุกคนรู้ว่า อันอยู่ที่นี่นะ ถ้าจะส่งข้อมูลมาหาให้ส่งมาที่ไหน

```

logServer: Received from 2487009679 msg={"nodename": "Wemos-t1", "NodeID": 2487009679, "Temp": "23.60C", "Humidity": "38.20%"}
logServer: Received from 3163408206 msg={"nodename": "Wemos-04", "NodeID": 3163408206, "random value":139}
logServer: Received from 3893419003 msg={"nodename": "mcu-03", "NodeID": 3893419003, "random value":126}
logServer: Received from 572025877 msg={"nodename": "Wemos-01", "NodeID": 572025877, "random value":82}
logServer: Received from 3896664414 msg={"nodename": "mcu-04", "NodeID": 3896664414, "random value":162}
{"topic":"logServer", "nodeId":2752206217}
logServer: Received from 2488971084 msg={"nodename": "mcu-t1", "NodeID": 2488971084, "Temp": "24.30C", "Humidity": "36.50%"} —————
logServer: Received from 21366080749 msg={"nodename": "Wemos-03", "NodeID": 21366080749, "random value":92}
logServer: Received from 983688460 msg={"nodename": "esp-01", "NodeID": 983688460, "random value":154}
logServer: Received from 3163409689 msg={"nodename": "Wemos-02", "NodeID": 3163409689, "random value":45}
logServer: Received from 3896055851 msg={"nodename": "mcu-02", "NodeID": 3896055851, "random value":100}
logServer: Received from 3896713776 msg={"nodename": "mcu-01", "NodeID": 3896713776, "random value":126}
logServer: Received from 3163408206 msg={"nodename": "Wemos-04", "NodeID": 3163408206, "random value":20}
logServer: Received from 3893419003 msg={"nodename": "mcu-03", "NodeID": 3893419003, "random value":41}
logServer: Received from 572025877 msg={"nodename": "Wemos-01", "NodeID": 572025877, "random value":152}
logServer: Received from 2487009679 msg={"nodename": "mcu-t1", "NodeID": 2487009679, "Temp": "23.60C", "Humidity": "38.20%"}—————
logServer: Received from 3896664414 msg={"nodename": "mcu-04", "NodeID": 3896664414, "random value":97}
logServer: Received from 21366080749 msg={"nodename": "mcu-01", "NodeID": 21366080749, "random value":67}
logServer: Received from 2488971084 msg={"nodename": "mcu-t1", "NodeID": 2488971084, "Temp": "24.40C", "Humidity": "36.50%"}—————
logServer: Received from 983688460 msg={"nodename": "esp-01", "NodeID": 983688460, "random value":153}
logServer: Received from 3163409689 msg={"nodename": "Wemos-02", "NodeID": 3163409689, "random value":45}
logServer: Received from 3896055851 msg={"nodename": "mcu-02", "NodeID": 3896055851, "random value":104}
logServer: Received from 3896713776 msg={"nodename": "mcu-01", "NodeID": 3896713776, "random value":33}
logServer: Received from 3163408206 msg={"nodename": "Wemos-04", "NodeID": 3163408206, "random value":58}
logServer: Received from 3893419003 msg={"nodename": "mcu-03", "NodeID": 3893419003, "random value":142}
logServer: Received from 572025877 msg={"nodename": "Wemos-01", "NodeID": 572025877, "random value":179}
logServer: Received from 3896664414 msg={"nodename": "mcu-04", "NodeID": 3896664414, "random value":8}—————
{"topic": "logServer", "nodeId":2752206217} —————
Server Broadcast NodeID: 2487009679
logServer: Received from 2487009679 msg={"nodename": "Wemos-t1", "NodeID": 2487009679, "Temp": "23.60C", "Humidity": "38.20%"}—————
logServer: Received from 21366080749 msg={"nodename": "Wemos-03", "NodeID": 21366080749, "random value":21}
 Autoscroll No line ending 115200 baud Clear output

```

ขยายความในส่วนของ Server Node กันอีกนิดว่าทำไม่ต้องค่อย Broadcast Server NodeID อยู่ตลอดเวลา

ข้อดีเล็กๆน้อย คือ ถ้าในอนาคตต้องมีการเปลี่ยนอุปกรณ์ในส่วนของ Server Node ที่เสียไป แต่เมื่อสลับอุปกรณ์เปลี่ยน Server Node แน่นอนว่า ChipID จะเปลี่ยนไป ละ การ Broadcast Server NodeID และในส่วนของ Client ที่มี receivedCallback คอยเช็ค Server NodeID ก็จะทำให้ແກบไม่กระทบกับการทำงานเลย เพราะในทุกสุดทุก Node ก็จะรู้ว่าต้องส่งไปที่ Server Node ใหม่หมายเลขออะไร

ซึ่งถ้าเข้าใจหลักการที่เขียนมาสองบทนี้ เราจะสามารถประยุกต์ใช้งานในการทำ Server Node ที่มีมากกว่า 1 คือ ยังทำให้ระบบ Mesh Network ที่ใช้ในการสื่อสารอุปไป Network ภายนอกนั้นเสถียรยิ่งขึ้น หรือจะแบ่งเบาภาระของ Server Node เป็นลักษณะของ load balance ก็ยังได้

ตัวอย่างการประยุกต์ใช้งาน

Read and Forward: เคสนี้จะเข้าข่ายสำหรับพากการส่งข้อมูลเข้า Server อาจจะเป็นการรับอุณหภูมิ ความชื้น หรือแม้กระทั่งเสียงบริเวณต่างๆของโรงงาน แล้วมาเก็บไว้ที่ Server เพื่อที่จะแสดงผล หรือนำข้อมูลไปวิเคราะห์เหตุต่างๆ เมื่อในตัวอย่าง logServer/logClient ที่เราเพิ่งทำกันไป เป็นการใช้งานที่ค่อนข้าง One-way จะเป็นส่วนใหญ่ และก็เป็น Application ของ IoT จะเป็นส่วนใหญ่อีกเช่นกันด้วย

เนื่องจากการส่ง message ภายใน Mesh Network นี้สามารถที่จะกำหนดปลายทางผู้รับได้ด้วย NodeID หรือจะส่ง Broadcast แล้วค่อย Filter Message เอาไว้ได้เวลาปลายทางที่ระบุใน Broadcast Message เป็นของเราวหรือเปล่า จากนั้นอ่านค่าที่ได้มา เปรียบเทียบ logic ที่ต้องการ อาจจะเป็นการสั่งให้เปิดปั๊มน้ำเพื่อรดน้ำต้นไม้ หรือสั่งให้เปิด Siren นอกอาการกรณีมีการตรวจสอบไฟได้ จะแจ้งเป็น zone หรือแจ้งเป็นจุดๆ ก็แล้วแต่หลักการเขียนข้อความที่ส่งไป และหลักการเปรียบเทียบ logic ของข้อความที่ได้รับมา ซึ่ง Painlessmesh ใช้รูปแบบการส่งแบบ JSON ก็สะดวกยิ่งขึ้น

Point to Point / Independent: เนื่องจาก Mesh Network นั้นทำให้ทุก Node นั้นเชื่อมหากันอยู่แล้ว Node ไหน อยากคุยกับ Node ไหนก็ได้ เคสลักษณะนี้ ก็อาจจะมี Node ที่ทำงานเป็นลักษณะ Control Panel สามารถที่จะเลือกได้ว่าจะส่งคำสั่งไปให้ node ตัวไหนก็ได้ โดยการอ้างอิงจาก NodeID หรืออาจจะแปลง NodeID ให้เป็นชื่อที่อ่านแล้วเข้าใจง่ายเหมือนที่ผมทำเป็นต้น

หมายเหตุ

เนื่องจาก ESP8266/ESP32 ยังใช้ WiFi ย่าน 2.4 GHz อยู่ ซึ่งเวลาใช้งานในโหมดของ Station and AP เพื่อให้ Node ต่างๆมาเก็บรวมกันเป็น Mesh Network แล้ว ถ้าเปิดโปรแกรมพวก WiFi Analyzer มาดูก็จะเห็นชื่อ AP ใน Mesh ของเรารีบันมาเปลี่ยนเลย ซึ่งอาจจะไปกว่ากับ AP อื่นหรือ AP อื่นอาจจะมากวนเราก็ได้ ยังไงก็ลองพิจารณาการใช้งานให้เหมาะสมสมดุครับ



- ESP Mesh Network 3 – <https://meetjoeblog.com/2018/03/30/esp8266-esp32-mesh-network-painlessmesh-bridge-ep3/>

ESP8266 / ESP32 & Mesh Network ตอนที่ 3: Painlessmesh Bridge

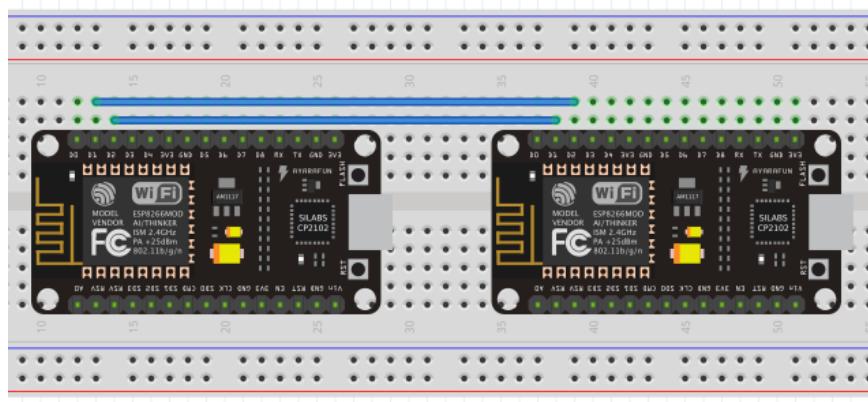
ตอนนี้จะเป็นเรื่องของการซึ่มต่อหรือ Bridge เจ้า Local Mesh Network ของเราเพื่อออกไปสู่ Internet กันครับ ซึ่งในที่นี่เราจะเพิ่ม Bridge Node ของเรามาข้างไป เพื่อต่อเข้ากับ Server Node แต่ก่อนอื่น เรามาดูเรื่องการส่งข้อมูลผ่าน Serial Port ของ Nodemcu สองตัวกันครับ เพราะสามารถประยุกต์ใช้งานอย่างอื่นได้อีก

ตัวอย่างเพื่อประยุกต์ใช้ในการส่งผ่านข้อมูลของ Nodemcu สองตัวผ่าน Serial Port

Redundant (Active-Standby): อย่างที่ทราบกันดีว่าพลังในการประมวลผลและหน่วยความจำของ ESP8266 นั้นมีจำกัด ซึ่งเราสามารถที่จะให้ Nodemcu ของเราทำงานสลับกันได้ กรณีตัวใดตัวหนึ่งตายไป อีกด้วยขั้นมาทำงานแทนแล้วก็สั่งสัญญาณไป reset Nodemcu ตัวที่ตายไป พอดีตอนปลูกขึ้นมาใหม่ก็สามารถอยู่ใน Standby Mode ค่อยฟังสัญญาณผ่านทาง Serial Port ซึ่งความสามารถทำ Heart Beat Check ผ่านทาง Serial Port ได้ โดยที่ Nodemcu สามารถทำงานแทนกันไปแทนกันมาได้ตลอด เพราะอย่างที่ทราบกันดีว่าการนี้ต้องการให้หน้าจอ死去ก็จะ hang ให้ถ้า reboot แล้วก็สามารถทำงานตามปกติได้ แต่ถ้าไม่ปักคิ้วจ้องทำให้งานที่ทำอยู่เสียหายได้ ถ้าเป็นสถานที่ที่ต้องเดินทางก็คงลำบากอีก บางคราวอาจจะคิดว่าอ้ววว ถ้าขึ้นก็ให้มันทำงานสองตัวดังแต่แรกได้สิ ทำงานพร้อมๆกันเลย อันนี้อาจต้องมองแล้วแต่เครื่องรับ เพราะบางที่เราอาจจะไม่ได้ใช้งานเป็น Sensor Node อย่างเดียว

Communication Channel: เศรษฐีจะเป็นเครื่องที่เราจะใช้กันในตอนที่ 3 นี้ครับ ซึ่งบางโปรเจคที่ทำกับ Arduino หรือ MCU บางตัวก่อนหน้านี้ที่ไม่มี WiFi รองรับแล้ววันนี้อย่างจะส่งค่าอุปกรณ์เพื่อ monitor ผ่านทาง Internet ก็สามารถทำได้โดยที่ MCU เดิมส่งค่าผ่านมาทาง Serial Port แล้วใช้ ESP8266 รับค่า จากนั้นก็ส่งข้อมูลผ่าน WiFi

น่าจะพอเห็นภาพการใช้งานเพื่อสื่อสารระหว่าง Nodemcu 2 ตัวผ่าน Serial Port กันแล้ว เรา มาดูในส่วนของ Code กันบ้างซึ่งเนื่องจากข้อจำกัดของ Nodemcu ที่มี Serial Port แค่คู่เดียว และเราใช้มันในการ Flash Code ของเราร่วมถึง Monitor ผ่าน Serial Monitor ใน Arduino IDE อีก แต่ว่าโชคดีที่ ESP8266 นั้นสามารถใช้งาน Software Serial เพื่อใช้ I/O ที่มีอยู่ในการทำงานเป็น Serial Port แทนได้ เพื่อให้เห็นภาพเรามาเริ่มที่ตัวอย่างนี้กันครับ โดยใช้ mcu2 ตัว



เริ่มจาก Copy Code ด้านล่างนี้แล้วก็ Flash ลง Nodemcu ทั้งสองตัวเลย ซึ่งใน Code ผูกกำหนดให้ D1 เป็น RX ส่วน D2 เป็น TX ละนั้น การต่อสายจะเป็นลักษณะไขว้กันระหว่าง RX<->TX ตามรูปด้านบน

```
#include <SoftwareSerial.h>
```

```
SoftwareSerial swSerial(D1, D2, false, 128); //Define hardware connections RX, TX
```

```
int i = 0;
```

```
String DataString;
```

```
void setup() {
```

```
  pinMode(D1, INPUT);
```

```
  pinMode(D2, OUTPUT);
```

```
  Serial.begin(115200); //Initialize hardware serial with baudrate of 115200
```

```
  swSerial.begin(115200); //Initialize software serial with baudrate of 115200
```

```
  Serial.println("\nSoftware serial test started");
```

```
}
```

```
void loop() {
```

```
  swSerial.print("Hello: " + String(i));
```

```
  i++;
```

```
  while (swSerial.available() > 0)
```

```
{
```

```
  char c = swSerial.read(); //gets one byte from serial buffer
```

```
  DataString += c; //makes the string DataString
```

```
}
```

```
  if (DataString != "")
```

```
{
```

```
  Serial.println(DataString);
```

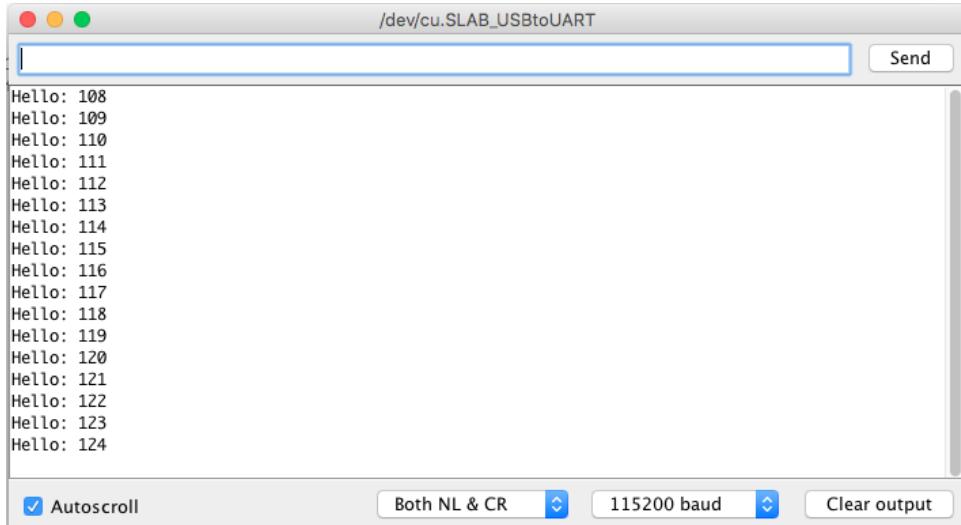
```
}
```

```
  DataString = "";
```

```
  delay(1000);
```

}

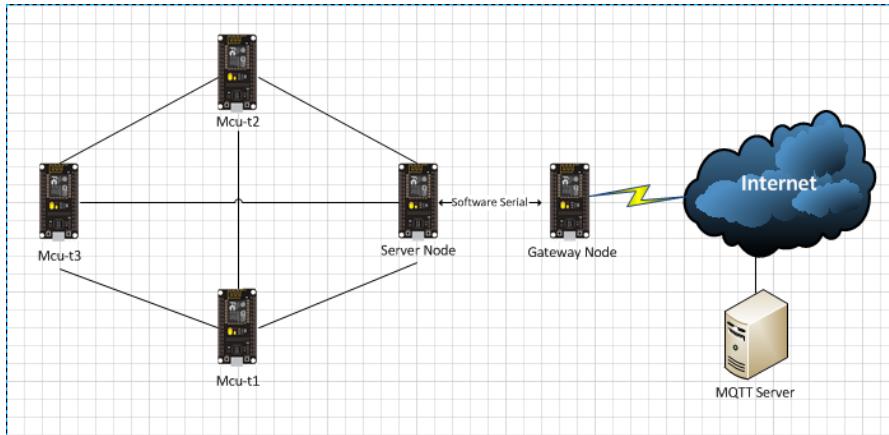
เมื่อเปิด Serial Monitor มาดูจะเห็นว่าค่าที่ print ออกมาผ่านทาง Serial Port(HW Serial) นั้นคือค่าที่เราอ่านได้จาก swSerial (Software Serial) ซึ่งวนลูปเพิ่มค่า i ไปเรื่อยๆ โดยเป็นค่าที่รับมาจาก Nodemcu อีกตัวหนึ่ง ค่าที่ได้รับมานี้ก็ขึ้นอยู่กับเราว่าจะเอาไปทำอะไรต่อ



```
>Hello: 108
Hello: 109
Hello: 110
Hello: 111
Hello: 112
Hello: 113
Hello: 114
Hello: 115
Hello: 116
Hello: 117
Hello: 118
Hello: 119
Hello: 120
Hello: 121
Hello: 122
Hello: 123
Hello: 124
```

Both NL & CR 115200 baud Clear output

มาต่อ กันที่โครงสร้างตอนนี้ของเรากันดีกว่า นั่นก็คือการ Bridge ระหว่าง Local Mesh Network ของเราเพื่อส่งข้อมูลออกไปยังโลกภายนอกกัน โดยที่ Scenario ที่เราจะจำลองขึ้นมาจะใช้ Nodemcu ทั้งหมด 5 ตัวดังรูปด้านล่างนี้



- mcu-t1 ถึง mcu-t3 เป็น Mesh Node ที่ส่งค่า Temp/Humidity ไปที่ Server Node
- Server Node ที่รับค่าจาก Node อื่นๆมา จะส่งค่าไปที่ Gateway Node ซึ่งทำการ Bridge ข้อมูลระหว่าง Mesh Network กับ WiFi Network และส่งค่าออกผ่านทาง internet
- Gateway Node ที่รับข้อมูลจาก Server Node ผ่านทาง software serial จะส่งค่าไปที่ MQTT Server

*MQTT คืออะไร โดยละเอียดໄວๆติดตามกันต่อตอนที่ 5 แต่คร่าวๆคือ messaging protocol ตัวหนึ่งซึ่งใช้รับส่งข้อความผ่าน IP Network ารมณ์คล้ายๆ Instant Message ซึ่งการรับส่งข้อความ จะใช้ลักษณะของการ Public(ส่ง)-Subscribe(รับ) กับ Topic ที่เราต้องการ ซึ่ง MQTT จะใช้มากในงานของ IoT เพราะไม่ซับซ้อน กิน resource น้อยและค่อนข้าง realtime

ในส่วนของ Code ก็ประกอบด้วย 3 ส่วนด้วยกัน Code ชุดแรกสำหรับ Nodemcu ที่ต่อ กับ DHT22 เพื่อส่งค่า Temp/Humidity ผ่านทาง Mesh Network ไปให้ Server

Code ชุดที่หนึ่ง สำหรับ mcu-t1 ถึง mcu-t3 ในการอ่านค่า Temp/Humidity

```
#include "painlessMesh.h"
#include "DHT.h"

#define DHTPIN D4
#define DHTTYPE DHT22

#define MESH_PREFIX "HelloMyMesh"
#define MESH_PASSWORD "hellomymeshnetwork"
#define MESH_PORT 5555

DHT dht(DHTPIN, DHTTYPE);

void receivedCallback( uint32_t from, String &msg );

painlessMesh mesh;

size_t logServerId = 0;

// Send message to the logServer every 5 seconds
Task myLoggingTask(5000, TASK_FOREVER, []() {

    float h = dht.readHumidity();
    float t = dht.readTemperature();

    DynamicJsonBuffer jsonBuffer;
    JsonObject& msg = jsonBuffer.createObject();
    msg["nodename"] = "mcu-t3"; //change for identify for the node that send data mcu-t1 to mcu-t3
    msg["NodeID"] = mesh.getNodeID();
    msg["Temp"] = String(t)+"C";
    msg["Humidity"] = String(h)+"%";

    String str;
    msg.printTo(str);
    if (logServerId == 0) // If we don't know the logServer yet
        mesh.sendBroadcast(str);
    else

```

```

mesh.sendSingle(logServerId, str);

// log to serial
msg.printTo(Serial);
Serial.printf("\n");
};

void setup() {

Serial.begin(115200);
Serial.println("Begin DHT22 Mesh Network test!");

dht.begin();

mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION ); // set before init() so that you can see startup messages

mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, STA_AP, AUTH_WPA2_PSK, 6 );
mesh.onReceive(&receivedCallback);

// Add the task to the mesh scheduler
mesh.scheduler.addTask(myLoggingTask);
myLoggingTask.enable();

}

void loop() {
// put your main code here, to run repeatedly:

mesh.update();

}

void receivedCallback( uint32_t from, String &msg ) {
Serial.printf("logClient: Received from %u msg=%s\n", from, msg.c_str());

// Saving logServer
DynamicJsonBuffer jsonBuffer;
JsonObject& root = jsonBuffer.parseObject(msg);
if (root.containsKey("topic")) {

```

```

if (String("logServer").equals(root["topic"].as<String>())) {
    // check for on: true or false
    logServerId = root["nodeId"];
    Serial.printf("logServer detected!!!!\n");
}
Serial.printf("Handled from %u msg=%s\n", from, msg.c_str());
}
}

```

Code ชุดที่สองสำหรับ Nodemcu ที่ทำหน้าที่เป็น Server Node เพื่อรับค่า Temp/Humidity ผ่านทาง Mesh Network จากนั้นส่งต่ออีกหนึ่งไปยัง Gateway Node โดยใช้การสื่อสารผ่านทาง Serial Port ซึ่งจุดที่จะเพิ่มก็คือการเรียกและกำหนดขาที่จะใช้งาน Software Serial และการส่งข้อมูลผ่านทาง Software Serial Port ใน receivedCallback

Code ที่ใช้สำหรับ Server Node

```

#include "painlessMesh.h"
#include <SoftwareSerial.h>

#define MESH_PREFIX      "HelloMyMesh"
#define MESH_PASSWORD   "hellomymeshnetwork"
#define MESH_PORT        5555

SoftwareSerial swSerial(D1, D2, false, 128); //Define hardware connections RX, TX

painlessMesh mesh;

// Send my ID every 10 seconds to inform others
Task logServerTask(10000, TASK_FOREVER, []() {
    DynamicJsonBuffer jsonBuffer;
    JsonObject& msg = jsonBuffer.createObject();
    msg["topic"] = "logServer";
    msg["nodeId"] = mesh.getNodeId();

    String str;
    msg.printTo(str);
    mesh.sendBroadcast(str);

    // log to serial
    msg.printTo(Serial);
    Serial.printf("\n");
}
}

```

```
});

void setup() {

    pinMode(D1, INPUT);
    pinMode(D2, OUTPUT);

    Serial.begin(115200); //Initialize hardware serial with baudrate of 115200
    swSerial.begin(115200); //Initialize software serial with baudrate of 115200


    mesh.setDebugMsgTypes( ERROR | CONNECTION | S_TIME );

    mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, STA_AP, AUTH_WPA2_PSK, 6 );
    mesh.onReceive(&receivedCallback);

    mesh.onNewConnection([](size_t nodeld) {
        Serial.printf("New Connection %u\n", nodeld);
    });

    mesh.onDroppedConnection([](size_t nodeld) {
        Serial.printf("Dropped Connection %u\n", nodeld);
    });

    // Add the task to the mesh scheduler
    mesh.scheduler.addTask(logServerTask);
    logServerTask.enable();
}

void loop() {
    mesh.update();
}

void receivedCallback( uint32_t from, String &msg ) {
    Serial.printf("logServer: Received from %u msg=%s\n", from, msg.c_str());

    swSerial.print(msg.c_str()); // Print received data in Mesh Network to Software Serial Port
}
```

Code ชุดที่สามสำหรับ Nodemcu ที่ทำหน้าที่เป็น Gateway Node เพื่อรับค่า Temp/Humidity ผ่านทาง Serial Port จาก Server Node จากนั้นส่งค่าที่ได้รับไปยัง MQTT Server ซึ่งในความเป็นจริงแล้วปลายทางของ Gateway ในการส่งผ่านข้อมูลไปจะเป็นอะไรก็ได้ อาจจะเป็น Webservice, Blynk, Firebase หรืออะไรก็ตามใช้หลักการเดียวกันเลย

Code ที่ใช้สำหรับ Gateway Node เพื่อส่งข้อมูลไปยัง MQTT server ซึ่งถ้าใครอยากระบดลองในส่วนนี้ก็ติดตามจากตอนที่ 5 ในเรื่องของการตั้ง MQTT Server บน Google Cloud หรืออาจจะหาให้บริการ Cloud MQTT ซึ่งมีส่วนให้บริการพรีเมียมได้ครับ

```
#include <SoftwareSerial.h>
```

```
#include <ESP8266WiFi.h>
```

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

```
const char* ssid = "xxx"; //wifi ssid
```

```
const char* password = "xxx"; //wifi passwd
```

```
IPAddress mqtt_server(xx, xx, xx, xx); //ip of mqtt server
```

```
WiFiClient espClient;
```

```
long lastMsg = 0;
```

```
char msg[100];
```

```
int value = 0;
```

```
String DataString;
```

```
SoftwareSerial swSerial(D1, D2, false, 256); //Define hardware connections RX, TX
```

```
void callback(char* topic, byte* payload, unsigned int length) {
```

```
    Serial.print("Message arrived [");
```

```
    Serial.print(topic);
```

```
    Serial.print("] ");
```

```
    for (int i = 0; i < length; i++) {
```

```
        Serial.print((char)payload[i]);
```

```
}
```

```
    Serial.println();
```

```
}
```

```
PubSubClient client(mqtt_server, 1883, callback, espClient);
```

```
void setup() {  
  
    pinMode(D1, INPUT);  
    pinMode(D2, OUTPUT);  
  
    Serial.begin(115200); //Initialize hardware serial with baudrate of 115200  
    swSerial.begin(115200); //Initialize software serial with baudrate of 115200  
  
    Serial.println("Bridget Node Gateway to MQTT");  
  
    Serial.println();  
    Serial.print("connecting to ");  
    Serial.println(ssid);  
    WiFi.begin(ssid, password);  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(1000);  
        Serial.print(".");  
    }  
    Serial.println("");  
    Serial.println("WiFi connected");  
    Serial.println("IP address: ");  
    Serial.println(WiFi.localIP());  
  
    client.connect("MeshGateway", "xxx", "xxx"); //MeshGateway is node name, change xxx to user/passs of your server  
    client.setCallback(callback);  
    client.subscribe("command");  
  
}  
  
void loop() {  
  
    while (swSerial.available() > 0)  
    {  
        char c = swSerial.read(); //gets one byte from serial buffer  
        DataString += c; //makes the string DataString  
    }  
}
```

```

if (DataString != "") {
{
Serial.println(DataString);

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

client.loop();

DataString.toCharArray(msg, 100);
client.publish("envMesh", msg); //publish message to mqtt server with topic envMesh

}

DataStream = "";
delay(1000);

}

void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
// Attempt to connect
if (client.connect("MeshGateway", "xxx", "xxx")) { //username of your mqtt server
Serial.println("connected");
// Once connected, publish an announcement...
client.publish("outTopic", "hello world");
// ... and resubscribe
client.subscribe("inTopic");
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(10000);
}
}
}

```

```

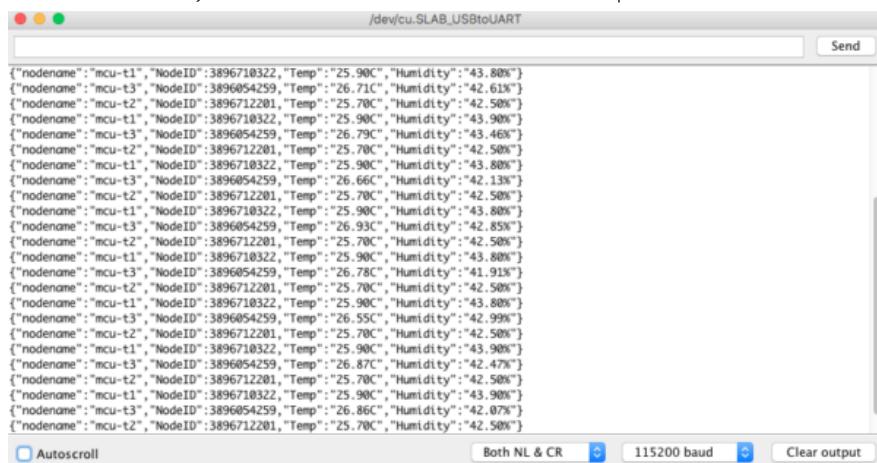
    }
}

}

```

เมื่อเปิด Nodemcu ทั้ง 5 ตัวขึ้นมา เหตุการณ์จากตอนที่ 2 ก็จะปรากฏขึ้นดัง step ต่อไปนี้

- mcu-t1 ถึง mcu-t3 จะ join เข้า mesh network แล้วส่งค่า broadcast ไปยังทุก Node และคopoly ฟังว่าใน Mesh Network นี้มี Server Node มีชื่อ ถ้ามีก็จะทำการบันทึก Server NodeID และหลังจากนั้นก็ส่งเป็นแบบรบประบ.lyทางไปที่ Server Node ละ
- Step นี้ที่จะเพิ่งเข้ามาก็คือ ที่ Server Node เมื่อได้รับค่าที่ส่งมาภายใน Mesh Network ก็จะทำการ Forward ต่อไปยัง Gateway Node ผ่านทาง software serial port ได้อะไรมาก็ส่งไปอย่างนั้น ให้ดู code ในส่วนของ receivedCallback ที่จะทำการ print ค่าออกทาง software serial swSerial.print(msg.c_str());
- ที่ Gateway Node เมื่อได้รับค่าที่ส่งมาจาก software serial port ก็จะทำการส่งต่อค่านี้ไปยัง mqtt server



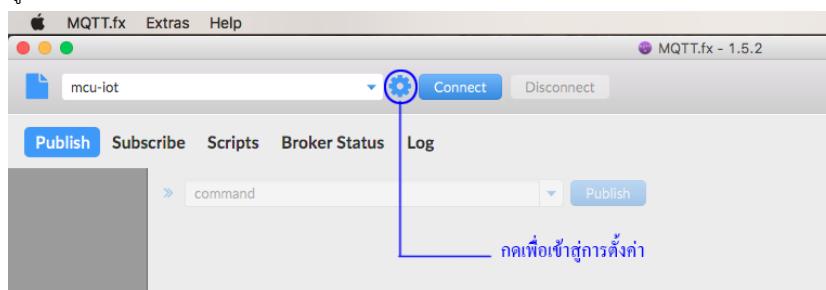
```

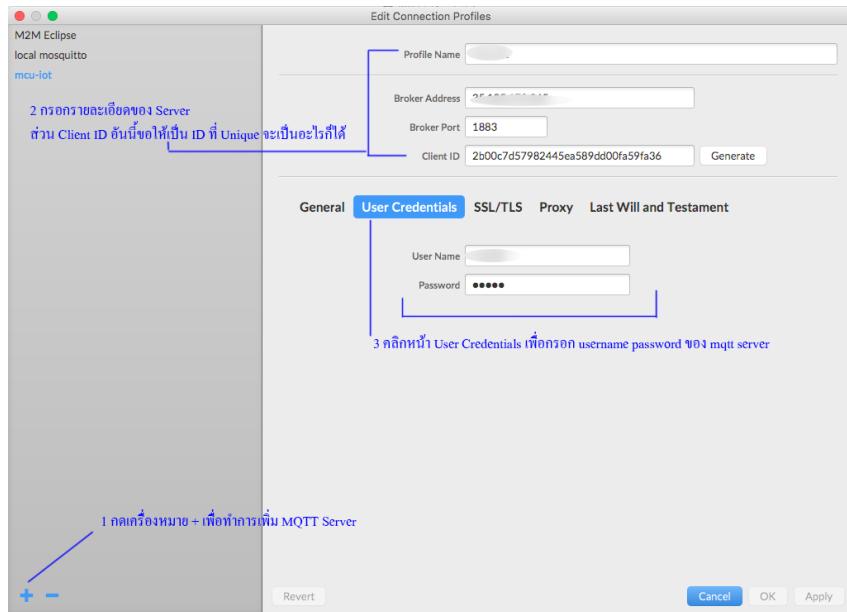
/dev/cu.SLAB_USBtoUART
Send
{"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.80%"}, {"nodeID": "mcu-t3", "Temp": "26.71C", "Humidity": "42.61%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}, {"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.90%"}, {"nodeID": "mcu-t3", "Temp": "26.79C", "Humidity": "43.46%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}, {"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.80%"}, {"nodeID": "mcu-t3", "Temp": "26.66C", "Humidity": "42.13%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}, {"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.80%"}, {"nodeID": "mcu-t3", "Temp": "26.93C", "Humidity": "42.85%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}, {"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.80%"}, {"nodeID": "mcu-t3", "Temp": "26.78C", "Humidity": "41.91%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}, {"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.80%"}, {"nodeID": "mcu-t3", "Temp": "26.55C", "Humidity": "42.99%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}, {"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.90%"}, {"nodeID": "mcu-t3", "Temp": "26.87C", "Humidity": "42.47%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}, {"nodeID": "mcu-t1", "Temp": "25.90C", "Humidity": "43.90%"}, {"nodeID": "mcu-t3", "Temp": "26.86C", "Humidity": "42.07%"}, {"nodeID": "mcu-t2", "Temp": "25.70C", "Humidity": "42.50%"}]
Autoscroll Both NL & CR 115200 baud Clear output

```

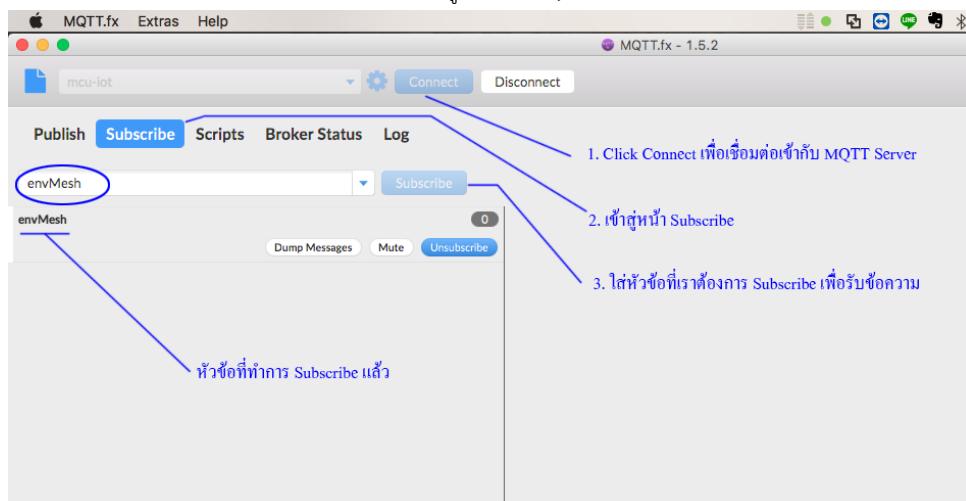
ส่วนผลลัพธ์ที่อ่านค่าได้จาก MQTT Server นั้นผมใช้โปรแกรม MQTT.fx ซึ่งเป็น Client ที่มีให้ใช้งานได้หลากหลาย platform เลยเนื่องจากว่า เป็น Java based ก็สามารถโหลดได้จากเวปของ MQTT.fx ได้โดยครับ

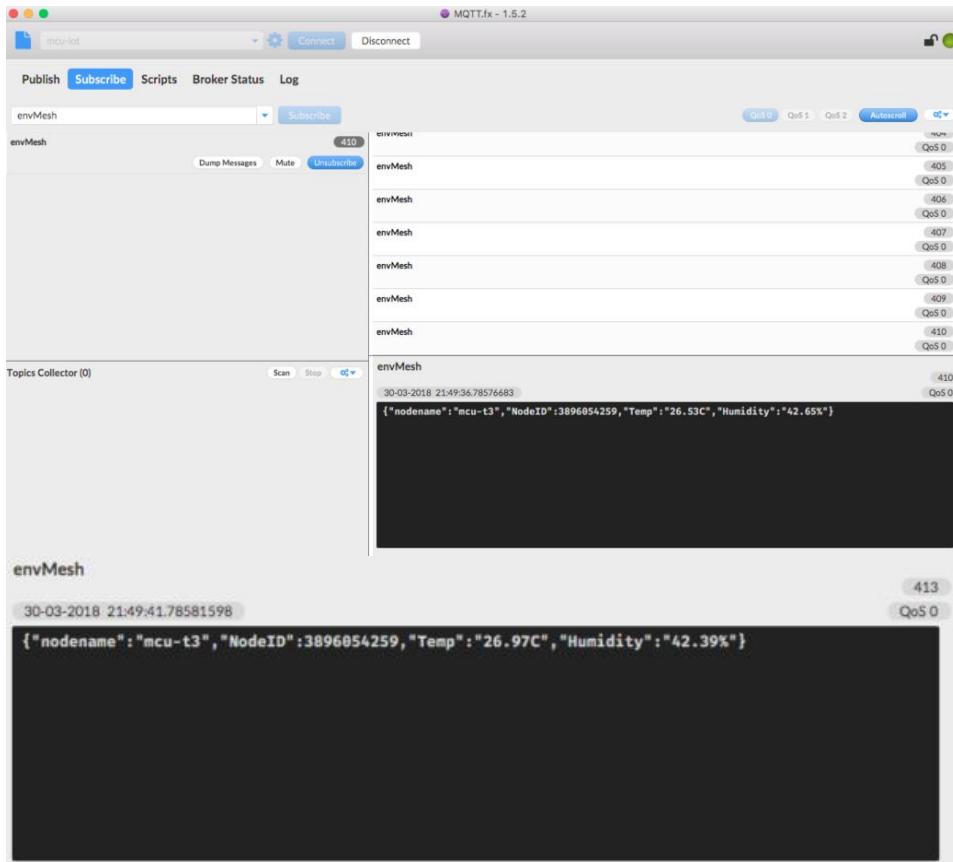
การใช้งานก็เริ่มจากการตั้งค่ากันก่อน ที่หน้าแรกให้คลิกที่รูปเปื้อง เพื่อเข้าสู่หน้าจอการตั้งค่า แล้วก็ตั้งค่าตาม MQTT Server ของแต่ละท่านที่เข้ากันอยู่โดยครับ หรือจะใช้บริการฟรีของ Cloud MQTT ก็ได้





ทำการ Subscribe หัวข้อ envMesh ตาม Code ที่อยู่ใน Gateway Node





จากกรุ๊ปที่จะเห็นข้อความที่อ่านได้จาก MQTT Server ก็จะเป็นรูปแบบ JSON โดยส่วนมากเป็นหอดๆ จาก Local Mesh Network → Server Node ภายใน Mesh Network → Gateway Node ผ่านการ Bridge โดยใช้ Software Serial จากนั้นก็ส่งต่อไปที่ MQTT Server ผ่าน WiFi/Internet → และเราที่ใช้โปรแกรม MQTT.fx เข้าไป Subscribe เพื่ออ่านค่าที่ส่งมาให้ทั่วข้อ envMesh

จริงๆ สำหรับวันนี้ใจความหลักเลยจะเป็นเรื่องของการใช้งาน Software Serial ที่ใช้ในการสื่อสารกันระหว่าง Nodemcu 2 ตัว ส่วนเรื่อง MQTT Server นั้นก็ เป็นการทึ่งpmเพื่อให้ไปค้นกันต่อ หรือจะรอถึงตอนที่ 5 ก็ได้กับการ Setup MQTT Server ขึ้นมาใช้งานเองบน Google Cloud และพบกันใหม่ตอนที่ 4 ซึ่งเรา จะ Bridge ข้อมูลจาก Local Mesh Network ของเราผ่าน LORA Network กันครับ

- ESP Mesh Network 4 – <https://meetjoeblog.com/2018/04/25/esp8266-esp32-painlessmesh-bridge-with-lora-ep4/>

ESP8266 / ESP32 & Mesh Network ตอนที่ 4: Painlessmesh Bridge with LoRa

เนื้อหาหลักๆ ในตอนนี้ก็จะเกี่ยวกับเรื่องของ LPWAN (Low Power Wide Area Network) อย่าง LoRa ทั้งการใช้งาน ข้อดีข้อเสีย กฎหมายที่ เกี่ยวข้องในการนำมาใช้งาน เพราะถ้าติดตามอ่านมาตั้งแต่ตอนที่ 1 (ESP8266/ESP32: Introduction & Painlessmesh) ใจความหลักในการใช้งาน Mesh Network ด้วย Painlessmesh นั้นจะไปตั้งแต่ตอนที่ 3 แล้วครับ ที่เหลือเป็นการประยุกต์มากกว่า ว่าเราจะใช้งาน Mesh Network ให้เหมาะสมยังไง หรือเพื่อ จุดประสงค์ไหน เช่น

- เพื่อลด Single Point of Failure
- เพื่อย้ายพื้นที่การครอบคลุมในการส่งข้อมูล
- ลดต้นทุนในการขยายจุดกระจายสัญญาณ

จากนั้นการส่งข้อมูลจาก Mesh Network ซึ่งเปรียบเหมือน Local Network ของเรา เราจะ Bridge ยังไงเพื่อให้ข้อมูลเหล่านี้ไปยัง Server, Database, Cloud, MQTT Broker หรืออะไรก็แล้วแต่ที่อยู่ Network ภายนอกหรือผ่าน Internet ซึ่งอาจจะได้ทดลองทำกันไปในตอนที่ 3 (ESP8266/ESP32: Painlessmesh Bridge) ซึ่งถ้าใครจะไม่ใช้ Software Serial ในการ Bridge ก็ยังสามารถทำได้อีกหลายวิธี ซึ่งวันนี้เราจะมา Bridge ผ่าน LoRa กัน

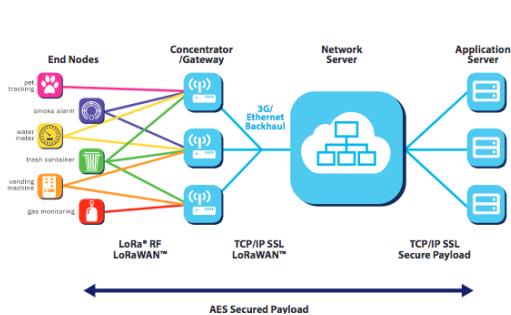
LoRa vs LoRaWan



ซึ่งทั้งสองของเรานั้นก็จะคุยกันในรูปแบบของ LoRa และการใช้งานร่วมกัน Mesh Network จะนั้นเรามาดูกันว่าแตกต่างกันระหว่าง LoRa และ LoRaWan กัน ก่อน



LoRa จะเป็นการสื่อสารในระดับ Physical Layer ซึ่งหมายความว่าจะใช้งานในลักษณะแบบ P2P ไม่มี Encryption ส่งข้อมูลกันแบบ Broadcast ออกไปเลย จะนั้นถ้ามีคืนดัง LoRa Node ขึ้นมาแล้วให้ค่า Setting เดียวกับเราที่สามารถที่จะเห็นข้อมูลได้ ซึ่ง LoRa ถูกพัฒนาเป็น Wireless Data Communication โดยบริษัท Cycleo (Grenoble, France) จากนั้นผู้ผลิตชิป Semtech ก็เข้าไปซื้อบริษัทนี้มาในปี 2012 ถ้าดูตามスペคของชิปยอดฮิตอย่าง SX1276 ที่รองรับย่านความถี่ 920-925MHz ตามประกาศของ กสทช ก็จะเห็นว่าที่ LoRa นั้นส่งได้ไกลถึงส่วนนึงก็ เพราะ High sensitivity ที่รองรับถึง -148dBm เลย (ข้อมูล Semtech SX1276) แต่ใช้งานจริงจะได้ที่เท่าไหร่ต้องดูกัน ผู้ให้บริการทางด้านโทรศัพท์มือถือในบ้านเราร่วมนี้ ทั้ง AIS (AIS NB-IoT) และ True (True IoT) ก็เปิดตัวบริการ NB-IoT ออกมาก สำนักงาน CAT ก็มาทางสาย LoRa (LoRa IoT by CAT) ซึ่งทั้งคู่ก็ออกมาเพื่อรับการสื่อสารสำหรับบริการประเภท IoT ในส่วนของ AIS และ True ที่เป็นผู้ให้บริการมือถืออยู่แล้ว มีความถี่ที่ใช้ในการจัดสรรสำหรับบริการ LTE การมาให้บริการ NB-IoT ก็จะประหดตันทุนไปได้เยอะแย่ เป็นการ Utilize ความถี่ที่ตัวเองได้รับอนุญาตมาด้วย เท่าที่เห็นนี้เป็น 3 ผู้บริการหลักๆที่เปิดตัวออกมานั้นจะอยู่ Ontop ของการสื่อสารแบบ LoRa Radio อีกที่ซึ่งจะมีในส่วนของ Protocol การรับส่งข้อมูลเข้ามา รองรับ Encryption ซึ่งใช้ AES-128 ในการเข้ารหัส สามารถทำ QOS ได้ด้วย โดยรายละเอียดของ LoRaWan นั้นจะมีอยู่ค่อนข้างเยอะพอสมควร รวมถึงในส่วนของ Device End Node ก็ยังสามารถจำแนกได้ออกเป็น Class A B C อีก และการทำงานก็จะแบ่งออกเป็น Tier ตามรูปด้านล่างนี้เลยครับ



ฉะนั้นการจะเลือกใช้ LoRa หรือ LoRaWan ก็ควรเลือกให้เหมาะสมกับ Application ที่จะใช้งานด้วย และที่สำคัญควรอยู่ตามข้อกำหนดของ กสทช ด้วยครับเนื่องจากคืนความถี่เป็นทรัพยากร่วมของทุกคนที่มีการใช้งานร่วมกัน ถ้าใช้ LoRa ก็ต้องดูในเรื่องของ Duty Cycle เองด้วยไม่ใช่ส่งข้อมูลตลอดเวลา จึงการใช้งานคืนความถี่ตลอด รวมถึงกำลังส่งที่ต่ำมากไป ก็จะไปรบกวน การใช้งานของอุปกรณ์หรือ Application ที่ใช้งานในความถี่เดียวกัน

ข้อมูลเพิ่มเติมจาก LoRa Alliance ที่น่าศึกษา ซึ่งทาง LoRa Alliance ที่เป็น Non Profit Organization ที่ผลักดันในเรื่องของการใช้ LoRa/LoRaWan เพื่อใช้ในงาน IoT

- LoRaWan What is it? (Download Link)
- LoRaWan 101: A Technical Introduction (Download Link)

Limitations

เราสามารถดูข้อดีข้อเสียของการใช้ LoRaWan กันบ้าง ไม่ใช่ว่าเทคโนโลยีที่ดีอย่าง LoRa/LoRaWan จะเหมาะสมครอบคลุมทุก Application ฉันได้กล่าวไว้ใน Mesh Network ก็จะเหมาะสมกับ Application ในบางประเภท หรือบางงาน เพราถ้าเราต้องการเอา LoRa/LoRaWan ไปส่งข้อมูลแบบ Realtime ก็คงไม่เหมาะสมแน่นอนจะกล่าวเป็นการจ่อจ่อของสัญญาตลดเวลา หรือให้ส่งข้อมูลขนาดใหญ่ๆ ก็คงไม่เหมาะสมเช่นกัน เพราะถูกจำกัดด้วยข้อมูลที่ส่งขนาดไม่เกิน byte และความเร็วในการส่งที่ต่ำ

Suitable use-cases for LoRaWAN:

- Long range – multiple kilometers
- Low power – can last months (or even years) on a battery
- Low cost – less than 20€ CAPEX per node, almost no OPEX
- Low bandwidth – something like 400 bytes per hour
- Coverage everywhere – you are the network! Just install your own gateways
- Secure – 128bit end-to-end encrypted
- Geolocation / Triangulation – you should probably use GPS for this, but we're doing our best to make it work with just LoRa. Check out Collos.

Not Suitable for LoRaWAN:

- Realtime data – you can only send small packets every couple of minutes
- Phone calls – you can do that with GPRS/3G/LTE
- Controlling lights in your house – check out ZigBee or BlueTooth
- Sending photos, watching Netflix – check out WiFi

ที่มา: (Limitations of LoRaWan:The Things Network)

Regulation & NBTC

เนื่องจากคลื่นความถี่อีเป็นทรัพยากรที่ต้องกู้จัดสรร เพราะมีอยู่อย่างจำกัด และต้องมีกำกับดูแลในเรื่องของการใช้งาน ในประเทศไทยก็จะมีองค์กรที่ทันสมัยมีวากฯ องค์กรหนึ่งซึ่งคือ กสทช (สำนักงานคณะกรรมการกิจกรรมกระจายเสียง กิจกรรมโทรทัศน์ และกิจกรรมโทรคมนาคมแห่งชาติ) ที่เดี่ยวๆ ก็จะชอบมีคนออกแบบให้เข้าร่วม เด่นชัดในอนุญาต มั่ง ปัจจุบันนี้นั่นเอง แล้วสุดท้ายเรื่องกีจีบีไป มาดูเรื่องการใช้งานความถี่ของเรากันดีกว่าครับ

จากข้อมูลที่นำมาจากเวปของหน่วยงาน กสทช เองบอกได้เลยว่าเล่นอาจงที่เดียว บางที่ค้นหาด้วยเลขความถี่ต้องใช้เลขไทย ไม่ใช้ไทย เช่น ไม่เจอก็จะมองรวมมาได้ดังนี้โดยอิงตามความถี่ย่านที่เรามักจะเห็นคุณเคยเห็นการใช้งานกันสำหรับ LoRa, RFID และ RC นั้นก็คือ EU433 ISM Band, AS923-925MHz

- ตารางกำหนดคลื่นความถี่แห่งชาติ 2560 ([Download Link](#))
- กฎความ 2560: เอกสารประกอบการรับฟังความคิดเห็นสาธารณะ ร่างประกาศเกี่ยวกับการใช้คลื่นความถี่ 920-925MHz ([Download Link](#))
- สิงหาคม 2560: ร่างประกาศเกี่ยวกับการใช้คลื่นความถี่ 920-925MHz และแนวทางการอนุญาตเพื่อประกอบกิจการ IoT ([Download Link](#))
- พฤกษา 2560: ราชกิจจานุเบกษา – หลักเกณฑ์การอนุญาตให้ใช้คลื่นความถี่ย่าน 920-925MHz ([Download Link](#))
- พฤกษา 2560: ราชกิจจานุเบกษา – มาตรฐานทางเทคนิคของเครื่องโทรศัพท์เคลื่อนที่และอุปกรณ์สำหรับเครื่องวิทยุคมนาคมที่ไม่ใช่ประเภท RFID ซึ่งใช้คลื่นความถี่ย่าน 920-925MHz ([Download Link](#))
- มกราคม 2561: ราชกิจจานุเบกษา – หลักเกณฑ์และเงื่อนไขการอนุญาตให้ใช้คลื่นความถี่สำหรับอากาศยานซึ่งมีภาระน้ำหนักสำหรับใช้งานเป็นการทั่วไป ([Download Link](#))
- กฎหมาย 2561 ราชกิจจานุเบกษา – เครื่องวิทยุคมนาคมและสถานีวิทยุคมนาคมที่ได้รับการยกเว้นไม่ต้องได้รับใบอนุญาตตามพระราชบัญญัติวิทยุคมนาคม ([Download Link](#))
- ความถี่วิทยุและใบอนุญาตวิทยุโทรศัพท์ RFID ([Download Link](#))
- การปรับปรุงกฎระเบียบด้านการบริหารคลื่นความถี่และทรัพยากรโทรศัพท์เคลื่อนที่เพื่อร่วงรับการพัฒนาของ Internet of Things ในประเทศไทย ([Download Link](#))

ฉะนั้นสำหรับส่วนของคลื่นความถี่ 433MHz ซึ่งที่เราจะทดสอบกันสำหรับ LoRa นั้น ก็จะไม่ได้เกี่ยวกับการใช้งานหัว RFID หรือการใช้งานในด้าน RC กับอากาศยานไร้คนขับ แต่จะทดสอบโดยอาศัยความต้องการของวิทยุคมนาคมและสถานีวิทยุคมนาคมที่ได้รับการยกเว้นไม่ต้องได้รับใบอนุญาต ซึ่งย่าน 433MHz นั้นห้ามส่งเกิน 10 มิลลิวัตต์

สำหรับย่าน 920-925MHz นั้นซึ่งมีข้อกำหนดในส่วนของ มาตรฐานทางเทคนิคของเครื่องโทรศัพท์เคลื่อนที่และอุปกรณ์ กสทช.มท. 1033 – 2560 ว่าด้วยเรื่องเครื่องวิทยุ คมนาคมที่ไม่ใช่ประเภท Radio Frequency Identification: RFID ซึ่งใช้คลื่นความถี่ย่าน 920-925MHz แบบเอิร์ทซ์ ก็กำหนดไว้ว่า ไม่เกิน 50 มิลลิวัตต์ ที่ได้รับยกเว้นไม่ต้องได้รับใบอนุญาตให้ทำไว้ชนิดเข้าและ 나오ออกซึ่งเครื่องวิทยุคมนาคมและใบอนุญาตให้ตั้งสถานีวิทยุคมนาคมแต่ไม่ได้รับยกเว้นใบอนุญาตให้ค้างซึ่งเครื่องวิทยุคมนาคม

AS920-923

Used in Japan, Malaysia, Singapore

Uplink:

- 1. 923.2 - SF7BW125 to SF12BW125
- 2. 923.4 - SF7BW125 to SF12BW125
- 3. 922.2 - SF7BW125 to SF12BW125
- 4. 922.4 - SF7BW125 to SF12BW125
- 5. 922.6 - SF7BW125 to SF12BW125
- 6. 922.8 - SF7BW125 to SF12BW125
- 7. 923.0 - SF7BW125 to SF12BW125
- 8. 922.0 - SF7BW125 to SF12BW125
- 9. 922.1 - SF7BW250
- 10. 921.8 - FSK

Downlink:

- Uplink channels 1-10 (RX1)
- 923.2 - SF10BW125 (RX2)

ถ้ามีดังนี้ตาม Frequency Plan และตามช่วงความถี่ที่ กสทช ปลดล็อกให้กับอุปกรณ์และบริการ IoT ก็สามารถใช้ได้สองช่วงเครือข่าย ทั้ง AS920 กับ AS923 (The Things Network Frequency Plan)

จากข้อมูลด้านบนนี้ ถ้าต้องอ่านกันดีๆ หลักๆ ก็ควรยึดตามคุณภาพความถี่ที่ใช้งานและจุดประสงค์ที่ใช้งาน จากนั้นก็ต่อตามหลักเกณฑ์และเงื่อนไขที่ประกาศในราชกิจจานุเบกษา และถ้ามีข้อจำกัดด้วยว่าในการประกาศนั้นมีรายละเอียดประกาศตัวเก่าไว้หรือไม่ หรือพวงประโยคที่ว่าเว้นแต่กำหนดเป็นอย่างอื่น พวงร่างกับการรับฟังความคิดเห็นสาธารณะก็อาจใช้ประกอบได้ กรณีที่สุดหากมีข้อสงสัยในเรื่องของการใช้งานความถี่ การผลิตหรือนำเข้าเรื่องอุปกรณ์คอมมูนิเคชันที่ใช้ความถี่ ก็ควรขอคำตอบจาก กสทช เป็นหลักซึ่งก็ควรทำเป็นหนังสือแจ้งไปเพื่อให้ กสทช ตอบกลับว่าสามารถทำได้ หรือไม่สามารถทำได้ หรือให้ยึดตามข้อกำหนดใหม่

Scenario

กรณีหนึ่งที่เราควรจะ Bridge ผ่าน LoRa ซึ่งข้อดีของ LoRa เลยก็คือเรื่องของ Power Consumption ที่ต่ำ และระยะในการส่งที่สูง ฉะนั้นด้วยคุณสมบัตินี้ แทนที่จะต้องการเก็บข้อมูล หรือควบคุมอุปกรณ์ จะต้องวาง Mesh Network ให้ครอบคลุมมาถึงจุดที่จะส่งข้อมูลออก Internet อาจจะไม่จำเป็น แต่ถ้าการส่งระยะใกล้ผ่าน LoRa มาแล้วค่อยมาออก Internet จากผู้ใช้งาน LoRa Node ที่ทำตัวเป็น Gateway ก็ได้ สำหรับตอนที่ 4 นี้จะเป็นการใช้งานผ่าน LoRa นะครับ แต่ถ้าใครอยากรายละเอียดมากกว่า แนะนำ Advance Setup LoRaWan Gateway และใช้งานผ่าน LoRaWan ก็ได้ อย่างที่บอกไว้ครับ หลักการเดียวกันกับตอนที่ 3 ลองมาคิดกันเล่นๆครับ

Scenario 1: Smart Farm

ลองนึกภาพของ Green House เรือนปลูกผัก ฟาร์มข้าวโพด หรือทุ่งนา ที่ต้องการควบคุมระบบจ่ายน้ำ หรือเก็บข้อมูลอุณหภูมิ ความชื้น ค่า PH/EC ของดินหรือสารละลาย แล้วส่งข้อมูลมาที่ส่วนกลางที่อยู่ติดกับพื้นที่ทำการว่าง Sensor Node เหล่านี้

สถานการณ์ใช้ Mesh Network เหมาะมาก เพราะได้ประโยชน์ในเรื่องของการคลอบคลุมพื้นที่ที่กว้าง จากการกระจายตัวของ Node และสร้างเป็น Mesh Network โดยที่จุดเชื่อมต่อสุดท้ายอาจอยู่ที่ชายขอบ หรือศูนย์กลางของ Mesh Network ก็จะเหมาะสมมาก สามารถลดต้นทุนในเรื่องของการสร้าง Network สำหรับการสื่อสารของแต่ละ Node ได้

AS923-925

Used in Brunei, Cambodia, Hong Kong, Indonesia, Laos, Taiwan, Thailand, Vietnam

Uplink:

- 1. 923.2 - SF7BW125 to SF12BW125
- 2. 923.4 - SF7BW125 to SF12BW125
- 3. 923.6 - SF7BW125 to SF12BW125
- 4. 923.8 - SF7BW125 to SF12BW125
- 5. 924.0 - SF7BW125 to SF12BW125
- 6. 924.2 - SF7BW125 to SF12BW125
- 7. 924.4 - SF7BW125 to SF12BW125
- 8. 924.6 - SF7BW125 to SF12BW125
- 9. 924.5 - SF7BW250
- 10. 924.8 - FSK

Downlink:

- Uplink channels 1-10 (RX1)
- 923.2 - SF10BW125 (RX2)

Scenario 2: Falling Rock Alarm

สำหรับเคสนี้เป็นเคสจริงที่ได้หัวน้ำ ที่ใช้ในการตรวจสอบหินที่หล่นจากภูเขา เนื่องจากได้หัวน้ำนี้มีสภาพภูมิประเทศเป็นภูเขา และอยู่ในเขตแผ่นดินไหว ซึ่งถนนบางเส้นนั้นตัดผ่านภูเขามีอุบัติเหตุหล่นลงมา กว่าจะทราบเหตุต้องมีคนโทรแจ้งจากที่ไปเจอกัน



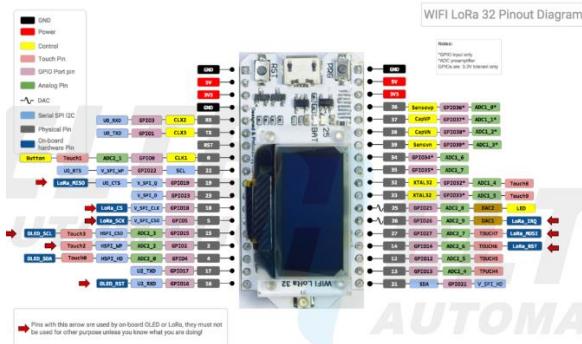
หัวน้ำหลักเลี้ยงเส้นทางได้ถูก

Scenario 3: Smart Bin

เคสนี้ก็เป็นเคสที่เกิดขึ้นจริงอีกเช่นกัน ลองคิดถึงเรื่องการเก็บขยะ ซึ่งในบางหมู่บ้าน ชุมชน ที่ต้องมีรถเก็บขยะวนหัวไปเก็บทุกวันๆ หรือบางที่วันนึงอาจจะหลายรอบก็ได้ หรือบางที่เข้าไปเก็บ ขยายในลังอาจจะไม่มีเลยก็ได้ ซึ่งเคสนี้มีคนทำเป็น Product Smart Bin ที่มีทั้งตัวบดอัดขยะเพื่อเพิ่มพื้นที่การเก็บขยะ ได้มากขึ้น มีตัว Monitor ระดับของขยะเพื่อส่งคำสั่งไปแจ้งเตือนที่ศูนย์รวมหัวรถเข็น รวมถึงจัดเส้นทางให้รถเก็บขยะได้ด้วย ซึ่งเทคโนโลยีการสื่อสารที่ใช้โดยมากก็จะเป็น LoRa หรือไม่ก็ Cellular (2G, 3G, NB-IoT) เพราะด้วยลักษณะการตั้งวางลังขยะของหมู่บ้าน หรือชุมชนใน ตปท อาจไม่เหมาะสมกับการนำ mesh มาใช้

Wiring & Diagram

ความจริงการใช้งานจะใช้งานผ่าน Module SX1278/1276 ของ Ai-Thinker ที่ได้แต่ก็ต้องต่อสายเข้ากับ Breadboard หรือในส่วนของ RFM95W ก็ต้องมาบัดกรีหากันอีก ฉะนั้นตอนนี้เนื้อหาจะยังสุดสำหรับผู้ที่ไม่สามารถเข้าใจได้ แต่สำหรับผู้ที่ต้องการรู้ว่าต้องทำอย่างไร แนะนำให้ดู wiring ไว้บน pcb เรียบร้อยแล้ว ซึ่ง pinout ก็จะเป็นดังรูปด้านล่างนี้



```
#define BAND 915.2E6 //you can set band here directly,e.g. 868E6,915E6
```

ติดตั้ง Arduino core for ESP32 WiFi chip

ขั้นตอนนี้อาจแตกต่างไปจากตอนที่เราติดตั้ง ESP8266, ATTiny85 หรือ STM32 F103C Series ไปจากตอนก่อนหน้านี้ ซึ่ง Source ที่ใช้ในการติดตั้งมีอยู่ 2 แหล่งด้วยกัน คือ

จาก Espressif ผู้ผลิตชิป ESP32

<https://github.com/espressif/arduino-esp32#installation-instructions>

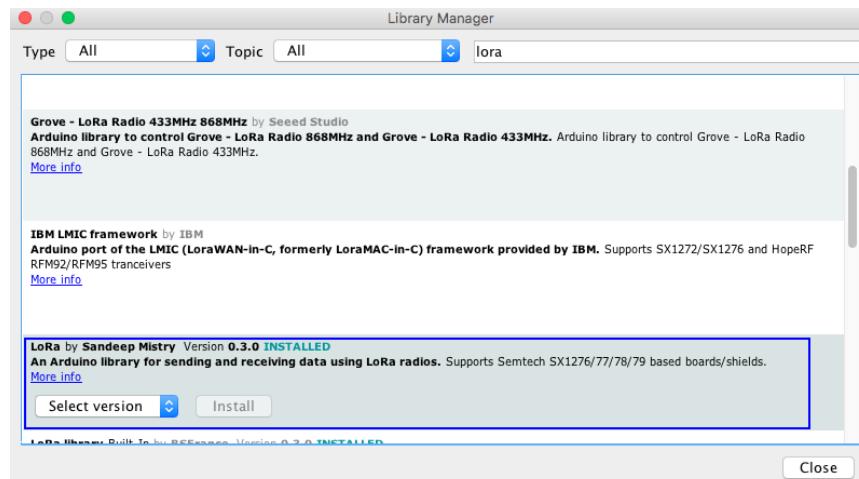
จาก Heltec ผู้ผลิตบอร์ด Heltec ESP32 LoRa

https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series#installation-instructions

ข้อแตกต่างระหว่างสองบอร์ดนี้คือ ถ้า Clone มาจาก Heltec จะมี Library LoRa มาด้วยซึ่งเป็นตัวที่ไม่มาจากของคุณ Sandeep Mistry แต่ถ้ากรณีมีการ Update ก็จะไม่ถูก Update ผ่านทาง Library manager ต้องดาวน์โหลดจากทาง Heltec เอง ผ่านเลือก Clone มาจากทาง Espressif และทำการติดตั้ง LoRa Library เอง ส่วนวิธีการติดตั้งก็ขึ้นกับ OS ของแต่ละท่านที่ใช้อยู่ หลังจากติดตั้งเสร็จแล้วก็อ่าวยลิงซั่งบนนี้และครับทำงาน Instruction ในแต่ละ OS ได้เลย

LoRa Library by Sandeep Mistry

สำหรับใครที่เลือกติดตั้ง Board โดยเลือก Source จาก Heltec ก็ขั้นตอนติดตั้ง Library นี้ได้เลยครับ แต่ถ้าใครที่ติดตั้งผ่าน Source ของ Espressif ก็ให้ติดตั้ง LoRa Library ของคุณ Sandeep Mistry โดยเข้าไปที่ Sketch -> Include Library -> Library Manager และค้นหาคำว่า LoRa ครับ จะเจอบนลับๆมากให้เลือกตามรูปด้านล่างแล้วกด Install ได้เลยครับ



มาตรฐานในส่วนของ Code ที่จะใช้ทดสอบระยะกันบ้าง ตัวโปรแกรมจะแบ่งเป็นสองชุดคือผู้ส่งและผู้รับ ผู้ส่งนี้ Count Number แล้วส่งออกไปเรื่อยๆ ส่วนผู้รับนั้นเนื่องด้วยไม่มี GPS Module ก็เลือกอัพเดต GPS Stream จาก Blynk มาใช้ในการเก็บค่าพิกัดแทน เพื่อที่จะได้มาระยะเปื้อนอย่างไรบ้าง ซึ่งถ้าใช้บอร์ดที่ใช้ชิป SX1278 ในส่วนของ Band จะกำหนดเป็น 433.175MHz และถ้าเป็นบอร์ดที่ใช้ชิป SX1276 ในส่วนของ Band จะกำหนดเป็น 923.2MHz นอกนั้นไม่มีอะไรแตกต่าง ส่วนใครที่จะทดสอบโดยที่ไม่ได้สนับใจเรื่องของ พิกัดก็ตัด code ในส่วนของ Blynk ออกได้ครับ

Code for Sender:

```
#include <SPI.h>
#include <LoRa.h>
#include "SSD1306.h"
```

```
int counter = 0;
```

```
// GPIO5 -- SX1278's SCK
// GPIO19 -- SX1278's MISO
// GPIO27 -- SX1278's MOSI
// GPIO18 -- SX1278's CS
// GPIO14 -- SX1278's RESET
// GPIO26 -- SX1278's IRQ(Interrupt Request)

//OLED pins to ESP32 0.96OLEDGPIOs via this connectin:
//OLED_SDA -- GPIO4
//OLED_SCL -- GPIO15
//OLED_RST -- GPIO16

SSD1306 display(0x3c, 4, 15);

#define SS    18
#define RST   14
#define DIO   26
#define BAND  433.175E6 //915E6

void setup() {
  Serial.begin(115200);

  pinMode(25, OUTPUT); //Send success, LED will bright 1 second

  while (!Serial);

  pinMode(16, OUTPUT);
  digitalWrite(16, LOW); // set GPIO16 low to reset OLED
  delay(50);
  digitalWrite(16, HIGH); // while OLED is running, must set GPIO16 in high

  Serial.println("LoRa Sender");

  SPI.begin(5, 19, 27, 18);
  LoRa.setPins(SS, RST, DIO);

  if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}
```

```

}

Serial.println("LoRa Initial OK!");

// Initialising the UI will init the display too.
display.init();
display.flipScreenVertically();
display.setFont(ArialMT_Plain_10);
}

void loop() {
    Serial.print("Sending packet: ");
    Serial.println(counter);

    // send packet
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();

    display.clear();
    display.setTextAlignment(TEXT_ALIGN_LEFT);
    display.drawString(10, 5, "Sending:");
    display.drawString(10, 20, "hello " + String(counter));
    // write the buffer to the display
    display.display();

    counter++;
    digitalWrite(25, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000);           // wait for a second
    digitalWrite(25, LOW); // turn the LED off by making the voltage LOW
    delay(1000);           // wait for a second

    delay(5000);
}

```

Code for Receiver:

```

#include <SPI.h>
#include <LoRa.h>
#include "SSD1306.h"

```

```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial

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

char auth[] = "xxxxxxxx"; //blynk auth code

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "xxx xxxx";
char pass[] = "xxxx";

float lat;
float lon;

// GPIO5 -- SX1278's SCK
// GPIO19 -- SX1278's MISO
// GPIO27 -- SX1278's MOSI
// GPIO18 -- SX1278's CS
// GPIO14 -- SX1278's RESET
// GPIO26 -- SX1278's IRQ(Interrupt Request)

//OLED pins to ESP32 0.96OLEDGPIOs
//OLED_SDA -- GPIO4
//OLED_SCL -- GPIO15
//OLED_RST -- GPIO16

SSD1306 display(0x3c, 4, 15);

#define SS    18
#define RST   14
#define DIO   26
#define BAND  433.175E6 //915E6

WidgetMap myMap(V1);
```

```
BLYNK_WRITE(V0) {  
    GpsParam gps(param);  
  
    // Print 6 decimal places for Lat, Lon  
  
    lat = String(gps.getLatitude(),6).toFloat();  
    lon = String(gps.getLongitude(),6).toFloat();  
  
    Serial.print("Lat: ");  
    Serial.println(lat,7);  
  
    Serial.print("Lon: ");  
    Serial.println(lon,7);  
  
    Serial.println();  
}  
  
void setup() {  
    Serial.begin(115200);  
    while (!Serial);  
  
    pinMode(16, OUTPUT);  
    digitalWrite(16, LOW); // set GPIO16 low to reset OLED  
    delay(50);  
    digitalWrite(16, HIGH); // while OLED is running, must set GPIO16 in high  
  
    Serial.println("LoRa Receiver");  
  
    SPI.begin(5, 19, 27, 18);  
    LoRa.setPins(SS, RST, DIO);  
  
    if (!LoRa.begin(BAND)) {  
        Serial.println("Starting LoRa failed!");  
        while (1);  
    }  
}
```

```
Serial.println("LoRa Initial OK!");

// Initialising the UI will init the display too.
display.init();
display.flipScreenVertically();
display.setFont(ArialMT_Plain_10);

Blynk.begin(auth, ssid, pass);

myMap.clear();

}

void loop() {
    String tmp_string, tmp_rssi;

    // try to parse packet
    int packetSize = LoRa.parsePacket();
    if (packetSize) {
        // received a packet
        Serial.print("Received packet ");

        // read packet
        while (LoRa.available()) {
            //Serial.print((char)LoRa.read());
            tmp_string += (char)LoRa.read();
        }

        Serial.print(tmp_string);

        tmp_rssi = LoRa.packetRssi();

        // print RSSI of packet

        Serial.println(" with RSSI " + tmp_rssi);

        //Serial.println(LoRa.packetRssi());
    }
}
```

```

display.clear();
display.setTextAlignment(TEXT_ALIGN_LEFT);
display.drawString(10, 0, "Received:");
display.drawString(10, 15, tmp_string+" RSSI: "+tmp_rssi);
display.drawString(10, 30, String(lat,7));
display.drawString(10, 45, String(lon,7));
// write the buffer to the display
display.display();

//BLYNK_WRITE(V0);
Blynk.syncVirtual(V0);
myMap.location(1, lat, lon, "value");

}

tmp_string ="";
tmp_rssi = "";

Blynk.run();

}

```

LoRa Range Test Result: 433.175MHz vs 923.2MHz

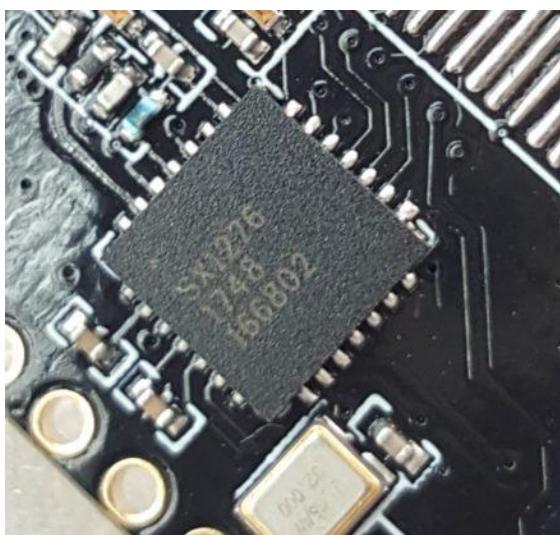


ค่าต่ำสุดที่ Heltec ESP32 LoRa SX1278 รับได้จะอยู่ที่ -108 dBm

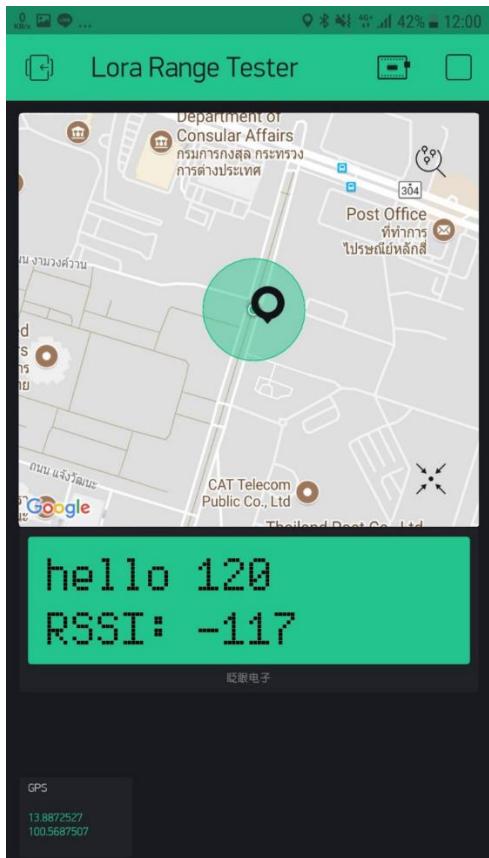
ในการทดสอบความถี่ 915MHz ใช้ชิป Heltec ESP32 LoRa SX1276 ที่ใช้งานย่าน 915MHz จะเสียแต่กีดีบอร์ดใหม่มาทันพอให้ทดสอบ ซึ่งเป็นของ TTGO เป็น ESP32 LoRa SX1276 เหมือนกันเพียงแต่แค็ปซูล OLED มาด้วยเท่านั้น ฉะนั้นการวัดระยะ และค่า RSSI ก็เลียອาร์ดอ่านค่า GPS จาก App Blynk บนโทรศัพท์มือถือแทน



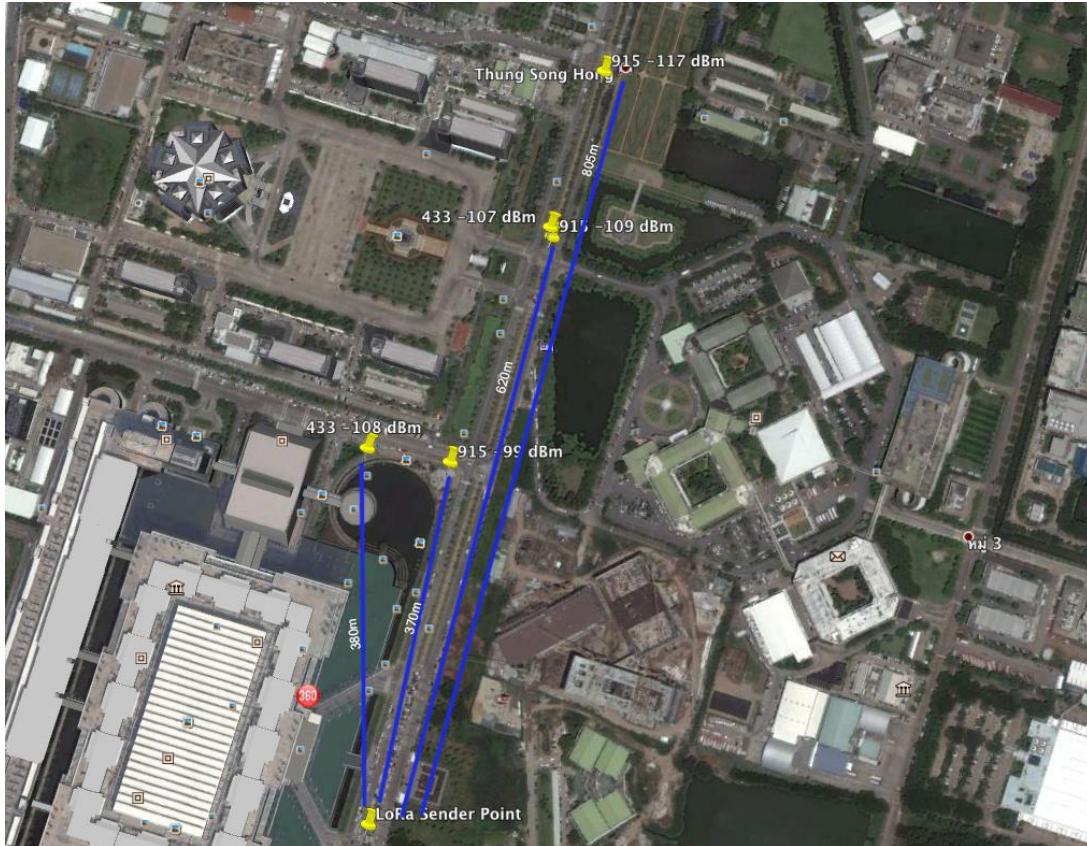
TTGO ESP32 LoRa SX1276



Semtech SX1276: LoRa ชิป ย่าน 915MHz



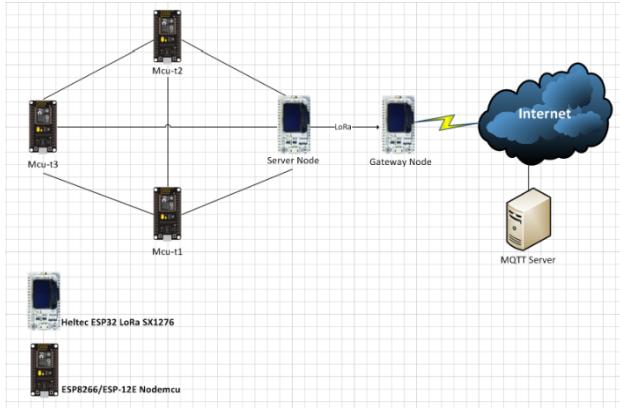
ชี้งการทดสอบสำหรับการรับส่งที่ความถี่ 923.2MHz ค่า RSSI ที่สามารถรับได้ดีสุดและรับได้ต่อเนื่องอยู่ที่ระหว่าง -110 ถึง -117 dBm



ผลการทดสอบระยะของแต่ละความถี่ที่ใช้กับรูปด้านบนเลยครับ ซึ่งถ้าเอาระยะห่างผลไกลสุดซึ่งอยู่ใน Line of Sight ก็จะอยู่ที่ประมาณ 700-750 เมตรสำหรับการรับ-ส่งที่ความถี่ 923.2MHz ส่วนคลื่น 433.175MHz ระยะห่างผลก็จะอยู่ที่ประมาณ 600 เมตร ทำให้การใช้งาน LoRa นั้นค่อนข้างสมชื่อ Long Range เลย เพราะนี่คือใช้สายอากาศที่แฉมมา ถ้าใช้สายอากาศเดียว และตั้งเสาสูงน่าจะได้ใกลกว่านี้ แต่ทั้งนี้ทั้งนั้นกำลังส่งก็ไม่ควรเกิน กษาท กำหนดน้ำหนัก ซึ่งเคสที่เขียนนี้ “สำหรับการทดลองเท่านั้น” เพราะจาก Code ได้มีการกำหนด TX Power ไว้ที่ 17dBm บวกกับเสาที่แฉมน่าจะไม่เกิน +3dBm ติกลมาก 20dBm หรือ 100 milliwatts แล้ว (แต่ส่วนตัวคิดว่าไม่น่าจะถึง น่าจะมี loss เยอะ ถ้ายากรู้ว่าริงๆคุณต้องเอาอุปกรณ์มาดู)

Painlessmesh Bridge with Lora

เกริ่นมาขอยาวสุดๆก่อนที่จะเข้าเรื่องของเรากันจริงๆในการใช้ Painlessmesh สร้าง Mesh Network และ Bridge ข้อมูลผ่าน LoRa กัน ซึ่ง Configuration ก็จะเป็นลักษณะดัง diagram ด้านล่างนี้ครับ โดยมี Heltec ESP32 Lora SX1278 เป็นตัว Bridge เพื่อส่งข้อมูลไปยัง MQTT Server



เพิ่งเห็นว่ามี Dependency Library เพิ่มเข้ามาสำหรับ Painlessmesh อีกด้วยคือ AsyncTCP สำหรับ ESP32 และ ESPAsyncTCP สำหรับ ESP8266 ถ้าใคร Update เป็น version ล่าสุดของ Painlessmesh ก็อย่าลืมติดตั้งเพิ่มเข้าไปนะครับ

สำหรับ ESP32

<https://github.com/me-no-dev/AsyncTCP>

สำหรับ ESP8266

<https://github.com/me-no-dev/ESPAsyncTCP>

Nodemcu + DHT22 (mcu-t1, mcu-t2, mcu-t3)

Code:

```
#include "painlessMesh.h"
#include "DHT.h"
#define DHTPIN D4
#define DHTTYPE DHT22
#define MESH_PREFIX    "HelloMyMesh"
#define MESH_PASSWORD  "hellomymeshnetwork"
#define MESH_PORT      5555
DHT dht(DHTPIN, DHTTYPE);

void receivedCallback( uint32_t from, String &msg );

painlessMesh mesh;
size_t logServerId = 0;
// Send message to the logServer every 10 seconds
Task myLoggingTask(10000, TASK_FOREVER, []() {

    float h = dht.readHumidity();
    float t = dht.readTemperature();

    DynamicJsonBuffer jsonBuffer;
    JsonObject& msg = jsonBuffer.createObject();
    msg["nodename"] = "mcu-t1"; //change for identify for the node that send data mcu-t1 to mcu-t3
    msg["NodeID"] = mesh.getNodeID();
    msg["Temp"] = String(t) + "C";
    msg["Humidity"] = String(h) + "%";
    String str;
    msg.printTo(str);
    if (logServerId == 0) // If we don't know the logServer yet
        mesh.sendBroadcast(str);
    else
        mesh.sendSingle(logServerId, str);
});
```

```

// log to serial
msg.printTo(Serial);
Serial.printf("\n");
});

void setup() {
    Serial.begin(115200);
    Serial.println("Begin DHT22 Mesh Network test!");
    dht.begin();
    mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION ); // set before init() so that you can see startup messages
    mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, STA_AP, AUTH_WPA2_PSK, 6 );
    mesh.onReceive(&receivedCallback);
    // Add the task to the mesh scheduler
    mesh.scheduler.addTask(myLoggingTask);
    myLoggingTask.enable();
}

void loop() {
    // put your main code here, to run repeatedly:
    mesh.update();
}

void receivedCallback( uint32_t from, String &msg ) {
    Serial.printf("logClient: Received from %u msg=%s\n", from, msg.c_str());
    // Saving logServer
    DynamicJsonBuffer jsonBuffer;
    JsonObject& root = jsonBuffer.parseObject(msg);
    if (root.containsKey("topic")) {
        if (String("logServer").equals(root["topic"].as<String>())) {
            // check for on: true or false
            logServerId = root["nodeId"];
            Serial.printf("logServer detected!!!\n");
        }
        Serial.printf("Handled from %u msg=%s\n", from, msg.c_str());
    }
}

```

Heltec ESP32 SX1276 (Server Node + LoRa)

Code:

```

#include "painlessMesh.h"
#include <SPI.h>
#include <LoRa.h>
#include "SSD1306.h"

// GPIO5 -- SX1278's SCK
// GPIO19 -- SX1278's MISO
// GPIO27 -- SX1278's MOSI
// GPIO18 -- SX1278's CS
// GPIO14 -- SX1278's RESET
// GPIO26 -- SX1278's IRQ(Interrupt Request)
//OLED pins to ESP32 0.96OLEDGPIOs :
//OLED_SDA -- GPIO4
//OLED_SCL -- GPIO15
//OLED_RST -- GPIO16

#define MESH_PREFIX    "HelloMyMesh"
#define MESH_PASSWORD  "hellomymeshnetwork"
#define MESH_PORT      5555

SSD1306 display(0x3c, 4, 15);
#define SS     18
#define RST   14
#define DIO   26
#define BAND  433.175E6 //915E6

painlessMesh mesh;
// Send my ID every 10 seconds to inform others

Task logServerTask(10000, TASK_FOREVER, []() {
    DynamicJsonBuffer jsonBuffer;
    JsonObject& msg = jsonBuffer.createObject();
    msg["topic"] = "logServer";
    msg["nodeId"] = mesh.getNodeId();
    String str;
    msg.printTo(str);
    mesh.sendBroadcast(str);
    // log to serial
    msg.printTo(Serial);
}

```

```
Serial.printf("\n");
});

void setup() {

    Serial.begin(115200);

    pinMode(25, OUTPUT); //Send success, LED will bright 1 second

    while (!Serial);

    pinMode(16, OUTPUT);
    digitalWrite(16, LOW); // set GPIO16 low to reset OLED
    delay(50);
    digitalWrite(16, HIGH); // while OLED is running, must set GPIO16 in high

    Serial.println("LoRa PainlessMesh Server");

    SPI.begin(5, 19, 27, 18);
    LoRa.setPins(SS, RST, DIO);
    if (!LoRa.begin(BAND)) {
        Serial.println("Starting LoRa failed!");
        while (1);
    }

    Serial.println("LoRa Initial OK!");
    // Initialising the UI will init the display too.
    display.init();
    display.flipScreenVertically();
    display.setFont(ArialMT_Plain_10);

    display.clear();
    display.setTextAlignment(TEXT_ALIGN_LEFT);
    display.drawString(10, 5, "Mesh Server Node:");
    display.display();
}
```

```

mesh.setDebugMsgTypes( ERROR | CONNECTION | S_TIME );
mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, STA_AP, WIFI_AUTH_WPA2_PSK, 6 );
mesh.onReceive(&receivedCallback);
mesh.onNewConnection([](size_t nodeld) {
    Serial.printf("New Connection %u\n", nodeld);
});
mesh.onDroppedConnection([](size_t nodeld) {
    Serial.printf("Dropped Connection %u\n", nodeld);
});
// Add the task to the mesh scheduler
mesh.scheduler.addTask(logServerTask);
logServerTask.enable();
}

void loop() {
mesh.update();
}

void receivedCallback( uint32_t from, String &msg ) {

String tmp_string = msg.c_str();

Serial.printf("logServer: Received from %u msg=%s\n", from, tmp_string);

Serial.println("");
Serial.println("Sending LoRa packet: "+tmp_string);

//เมื่อได้รับข้อความจากใน mesh network ก็ส่งต่อผ่านไปยัง LoRa
LoRa.beginPacket();
LoRa.print(tmp_string);
LoRa.endPacket();

display.clear();
display.setTextAlignment(TEXT_ALIGN_LEFT);
display.drawString(10, 5, "Sending: "+tmp_string.substring(13,19));
display.drawString(10, 20, "Temp: "+tmp_string.substring(49,55));
}

```

```

display.drawString(10, 35, "Humid: "+tmp_string.substring(69,75));
// write the buffer to the display
display.display();
}

Heltec ESP32 SX1276 (Gateway Node + LoRa)

```

Code:

```

#include <SPI.h>
#include <LoRa.h>
#include "SSD1306.h"

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

const char* ssid = "xxxWiFi-SSID";
const char* password = "xxxWiFi Password";
const char* mqtt_server = "xxx.xxx.xxx.xxx"; //<-- IP หรือ Domain ของ Server MQTT

long lastMsg = 0;
char msg[100];
int value = 0;

WiFiClient espClient;

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

```

```
PubSubClient client(mqtt_server, 1883, callback, espClient);

// GPIO5 -- SX1278's SCK
// GPIO19 -- SX1278's MISO
// GPIO27 -- SX1278's MOSI
// GPIO18 -- SX1278's CS
// GPIO14 -- SX1278's RESET
// GPIO26 -- SX1278's IRQ(Interrupt Request)
//OLED pins to ESP32 0.96OLEDGPIOs
//OLED_SDA -- GPIO4
//OLED_SCL -- GPIO15
//OLED_RST -- GPIO16
SSD1306 display(0x3c, 4, 15);
#define SS    18
#define RST   14
#define DIO   26
#define BAND  433.175E6 //915E6

void setup() {
  Serial.begin(115200);

  while (!Serial);
  pinMode(16, OUTPUT);
  digitalWrite(16, LOW); // set GPIO16 low to reset OLED
  delay(50);
  digitalWrite(16, HIGH); // while OLED is running, must set GPIO16 in high
  Serial.println("LoRa Receiver");

  SPI.begin(5, 19, 27, 18);
  LoRa.setPins(SS, RST, DIO);
  if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  Serial.println("LoRa Initial OK!');
```

```
// Initialising the UI will init the display too.

display.init();
display.flipScreenVertically();
display.setFont(ArialMT_Plain_10);

setup_wifi();
client.connect("ESP32Gateway", "joe1", "joe1");
client.setCallback(callback);
client.subscribe("command");

}

void setup_wifi() {
delay(10);
// We start by connecting to a WiFi network
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
// Attempt to connect
if (client.connect("ESP32Gateway")) {
Serial.println("connected");
// Once connected, publish an announcement...
```

```
client.publish("outTopic", "hello world");
// ... and resubscribe
client.subscribe("command");
} else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    // Wait 5 seconds before retrying
    delay(5000);
}
}
```

```
void loop() {
    String tmp_string, tmp_rssi;

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

    // try to parse packet
    int packetSize = LoRa.parsePacket();

    if (packetSize) {
        // received a packet
        Serial.print("Received packet ");
        // read packet
        while (LoRa.available()) {
            //Serial.print((char)LoRa.read());
            tmp_string += (char)LoRa.read();
        }
        Serial.print(tmp_string);

        tmp_rssi = LoRa.packetRssi();

        // print RSSI of packet
        Serial.println(" with RSSI " + tmp_rssi);
    }
}
```

```

display.clear();
display.setTextAlignment(TEXT_ALIGN_LEFT);
display.drawString(10, 0, "Received:");
display.drawString(10, 15, "From: " + tmp_string.substring(13, 19) + " RSSI: " + tmp_rssi);
display.drawString(10, 30, "Temp: " + tmp_string.substring(49, 55));
display.drawString(10, 45, "Humid: " + tmp_string.substring(69, 75));
// write the buffer to the display
display.display();

tmp_string.toCharArray(msg, 100);
Serial.print("Publish message: ");
Serial.println(msg);
client.publish("env", msg); //ส่งข้อมูล Temp + Humidity ออกไปที่ Topic "env"

}

tmp_string = "";
tmp_rssi = "";

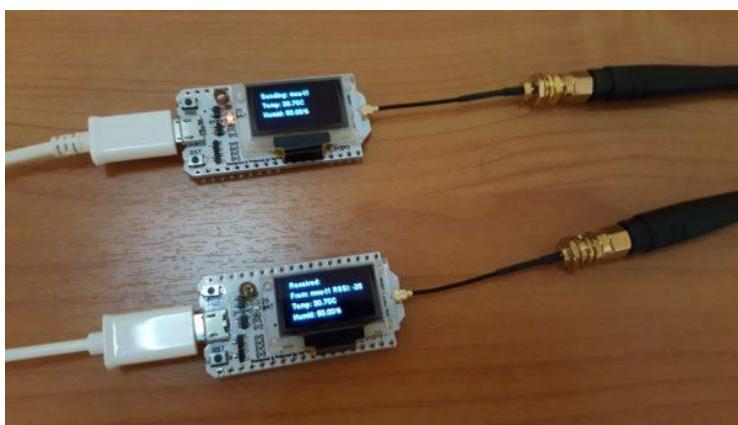
}

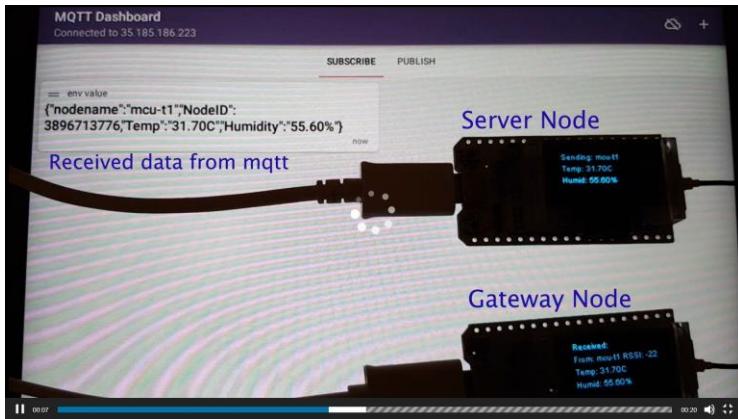
}

```

ผลลัพธ์ที่ได้

ถ้าทำตามขั้นตอนที่่าว่ามาทั้งหมดได้ ถึงบรรทัดนี้แล้วผลที่ได้ก็จะเป็นอย่างในคลิปด้านล่างนี้แหลกครับ mcu-t1, mcu-t2, mcu-t3 จะอ่านค่าจาก DHT22 และส่งผ่าน Mesh Network ไปยัง Server Node: Heltec ESP32 LoRa SX1278 จากนั้นเมื่อ Server Node ได้รับข้อมูลผ่านทาง Mesh Network ก็เอาเข้าห้องน้ำส่งต่อผ่านไปยัง LoRa





ในอีกฟากหนึ่ง Gateway Node: Heltec ESP32 LoRa SX1278 เมื่อได้รับข้อมูลมาจาก LoRa ก็จะเอาข้อมูลนั้น Publish ต่อไปยัง MQTT Server ที่เราได้สร้างกันไว้ ตั้งแต่ตอนที่ 3.5 ถ้าดูจากคลิปวิดีโอดูจะเห็นว่าระยะเวลาที่ใช้นั้นค่อนข้างเร็วมาก ซึ่งหลังจากนี้การจะนำข้อมูลจากภายใน Mesh Network ไปใช้งานก็ง่ายแล้ว จะเขียนลง Time Series Database เพื่อเอาไปแสดงผล หรือวิเคราะห์ข้อมูลต่อ ก็ไม่ยากแล้ว

สรุป

เนื้อหาในตอนนี้จะเน้นในเรื่องของ Data Communication อีกรูปแบบหนึ่งนั่นก็คือ LoRa ในการนำมาใช้งานร่วมกับ Mesh Network ซึ่งอย่างที่เกริ่นไปตอนแรกจะครับ ถ้าผ่านตอนที่ 3 มาแล้วก็ขึ้นอยู่กับเราว่าจะไป bridge กับอะไรเพื่อส่งต่อข้อมูลภายใน Mesh Network ไปยังอีก Network หนึ่ง ถ้าตัว Server Node ของเราต่อเข้ากับ Ethernet Module ENC28J60 ก็สามารถเป็น Bridge ในด้านนี้เองได้โดยถอดสิ่งข้อมูลผ่านสาย Lan ก็ได้ ดังนั้นที่ใช้บันทึกการณ์กันนี้เพื่อพยายามหาทางต่อเข้ากับตัวนี้ ที่เหลือก็จะขึ้นอยู่กับการประยุกต์ใช้ให้เหมาะสมนั้นเองครับ ใช่ว่าจำเป็นจะต้องใช้งาน Mesh Network สำหรับงาน IoT ทุกเคส หรือว่าจำเป็นต้องใช้งาน LoRa เพื่อให้ได้ระยะไกลๆ บางครั้งต้องพึ่งอุปกรณ์ของเราเข้ากับ GPRS/LTE Module ก็อาจจะทำให้ไปได้ใกลกว่าถ้าของผู้ให้บริการมีอีกด้วยด้วยซ้ำ ใกลกว่า NB-IoT ตอนนี้ด้วย

ส่วนในตอนหน้าที่น่าจะเป็นตอนสุดท้ายของ Series เรื่อง ESP8266/ESP32 กับ Painlessmesh ก็จะเป็นตอนปลีกย่อยละ เพราะในตอนที่ 3.5 ได้สอนในเรื่องของการติดตั้ง MQTT Server บน Google Cloud ไปแล้ว ก็คงปิด Series ด้วยการใช้งาน Telegraf, InfluxDB และ Grafana ไปเลย เพื่อที่จะทำให้ข้อมูลที่เหลือจาก Sensor Node ต่างๆ ไปอยู่ในรูปของ Data Visualization ที่สวยงามและเข้าใจง่าย

- ESP Mesh Network 5 - <https://meetjoeblog.com/2018/08/30/esp8266-esp32-mesh-network-ep5-influxdb-grafana/>

ตอนที่ 5: InfluxDB / Grafana

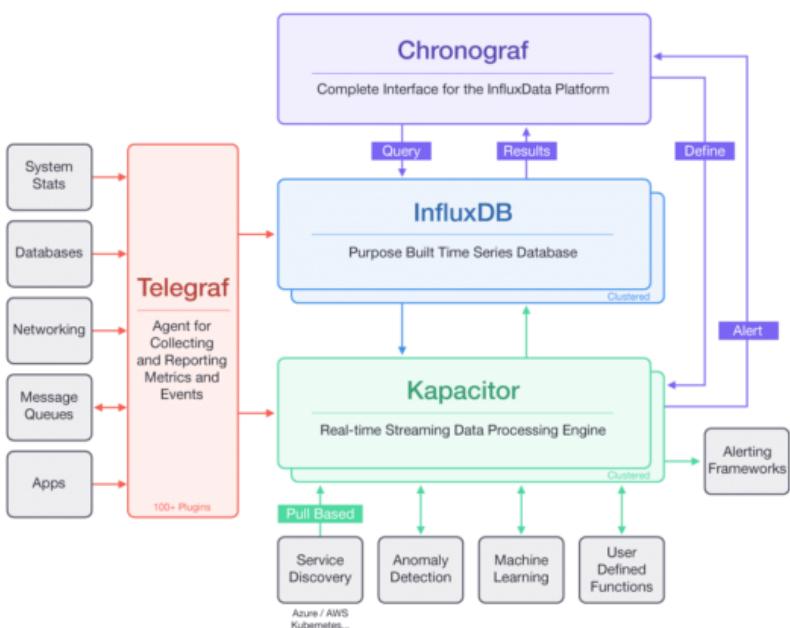
ก่อนเข้าไปสู่การติดตั้ง InfluxDB ซึ่งเป็น Time Series Database (TSDB) โดยที่เป็นการเก็บข้อมูล เมมโมนเก็บ log ที่เก็บไปเรื่อยๆ เรียงตามวันที่ และเวลาต่อๆ กันไป เช่นข้อมูลราคาหุ้น ข้อมูลอุณหภูมิ ความชื้นในแต่ละช่วงเวลา

ซึ่ง InfluxDB ก็เป็นหนึ่งใน TSDB ที่ถูก Optimize มาให้ทำงานในลักษณะนี้ โดยเน้นการทำงานที่ง่าย เร็ว ตัดพาก business logic ต่างๆไปอยู่ที่ module ข้างนอกแทน ฉะนั้นในการໂอนข้อมูลเข้ามาเพื่อเขียนลงฐานข้อมูลก็จะเร็วมากและใช้พื้นที่น้อยกว่าอีกด้วย

InfluxDB ก็มีทั้งส่วนที่เป็น Commercial และ Opensource ซึ่งข้อแตกต่างหลักๆของ version ที่เสียเงินกับ Opensource นั้นก็จะเป็นในเรื่องของการ support และที่เรื่องของ High Availability หรือการ Scale out ระบบเพื่อรับการทำ Clustering ระหว่างหลายๆ node ของ InfluxDB

InfluxCloud Great Place to Start	InfluxEnterprise Ready to Scale	TICK Stack Open Source
<ul style="list-style-type: none"> ● Open Source Core ● Extensible ● Support for Regular and Irregular Data ● High Availability (Clustering) 	<ul style="list-style-type: none"> ● Open Source Core ● Extensible ● Support for Regular and Irregular Data ● High Availability (Clustering) 	<ul style="list-style-type: none"> ● Open Source Core ● Extensible ● Support for Regular and Irregular Data ● High Availability (Clustering)

มาตรฐานส่วนของ Opensource กันซึ่งทาง Influxdata นั้นให้ชื่อว่า TICK Stack ซึ่งประกอบไปด้วย 4 Module ด้วยกันคือ Telegraf, InfluxDB, Chronograf และ Kapacitor



โดยที่การทำงานก็จะเป็นอย่างรูปที่แสดงอยู่ด้านบนเลยครับ

Telegraf จะเป็นตัวกลางในการรับข้อมูลจาก Source ต่างๆเข้ามาไม่ว่าจะเป็น Database อื่นๆหรือแม้กระทั่ง MQTT Message และรวมถึงตึงข้อมูลจาก InfluxDB และส่งกลับไปยังระบบอื่นด้วย ทำหน้าที่เหมือน ETL

InfluxDB ก็เป็นหัวใจของ TICK Stack นี้เลยซึ่งก็คือ Time Series Database ของเรานั่นเอง เน้นเก็บข้อมูลอย่างเดียว Chronograf อันนี้จะเป็น GUI ที่ใช้ในการ Manage InfluxDB รวมถึงการทำ Dashboard

Kapacitor จะเป็นส่วนที่ทำในเรื่องของ Data Processing ต่างๆ เช่นมีข้อมูลการรอสายโทรศัพท์ของลูกค้าที่โทรเข้ามาที่ Call Center เกินเท่านี้ให้ทำการ Alert แจ้งเตือนไปยังหัวหน้างาน ซึ่งก็สามารถนำ Module การวิเคราะห์ต่างๆเข้ามาใช้ได้ ไม่ว่าจะเป็นจากการตั้งค่าเอง หรือ Machine Learning ก็ได้

InfluxDB Setup โดยการติดตั้งนั้นจะเริ่มจาก InfluxDB กันก่อน โดยการเปิดหน้า SSH Command Windows ของ Google Cloud แล้วพิมพ์คำสั่งต่อตามด้านล่างนี้โดยครับโดยสเตปจะเป็น

Download & Install → Start Service → Verify

Download & Install

```
wget https://dl.influxdata.com/influxdb/releases/influxdb_1.4.0_amd64.deb
```

```
sudo dpkg -i influxdb_1.4.0_amd64.deb
```

Start InfluxDB

```
sudo systemctl start influxdb
```

Verify

```
curl "http://localhost:8086/query?q=show+databases"
```

ถ้า InfluxDB ที่เราเพิ่งติดตั้งไปทำงานได้อย่างถูกต้องแล้วหละก็ จะแสดงผลลัพธ์ตามบรรทัดล่างนี้เลย

```
{"results":[{"statement_id":0,"series":[{"name":"databases","columns":["name"],"values":[["_internal"]]}]}}}
```

Kapacitor Setup

มาถึง Module ที่สองที่เราจะทำการติดตั้งกัน เพื่อใช้ในการทำ Data Processing ซึ่งถ้าใครไม่ได้ใช้งานก็สามารถข้ามไปที่ Module ถัดไปได้

Download & Install

```
wget https://dl.influxdata.com/kapacitor/releases/kapacitor_1.4.0_amd64.deb
```

```
sudo dpkg -i kapacitor_1.4.0_amd64.deb
```

Start Kapacitor

```
sudo systemctl start kapacitor
```

Verify

```
kapacitor list tasks
```

ซึ่งถ้าติดตั้งแล้ว Service Start เรียบร้อยแล้ว ผลลัพธ์ที่ได้จะเป็นดังนี้ครับ

ID	Type	Status	Executing Databases and Retention Policies
----	------	--------	--

Telegraf Setup

Module Telegraf นี้เรียกได้ว่าเป็นอีกหัวใจหลักเลยก็ว่าได้ครับ เพราะจะเป็นตัวที่เก็บรวบรวม ดึงข้อมูลจาก Source ต่างๆเพื่อที่จะโอนเข้า InfluxDB ในตัวอย่างนี้เราจะเรียกเก็บ System Stat ของ VM Instance ของเรากันด้วยเพื่อดู load ของ CPU / RAM กัน

Download & Install

```
wget https://dl.influxdata.com/telegraf/releases/telegraf_1.4.3-1_amd64.deb
```

```
sudo dpkg -i telegraf_1.4.3-1_amd64.deb
```

Start Service

```
sudo systemctl start telegraf
```

Verify

ขั้นตอนนี้จะแตกต่างจาก Module อื่นๆ อย่างที่เกริ่นกันไว้ เราจะดึงข้อมูล System Stat ของระบบของเรา โดยการทำงานของ Telegraf จะทำงานผ่าน Plug-in ต่างๆ โดยขั้นแรกให้เปิดไฟล์ /etc/telegraf/telegraf.conf จากนั้นให้ตั้งส่วนของ Output Plugins

```
[[outputs.influxdb]]
```

```

## The full HTTP or UDP endpoint URL for your InfluxDB instance.
## Multiple urls can be specified as part of the same cluster,
## this means that only ONE of the urls will be written to each interval.
# urls = ["udp://localhost:8089"] # UDP endpoint example
urls = ["http://localhost:8086"] # required

## The target database for metrics (telegraf will create it if not exists).
database = "telegraf" # required

## Retention policy to write to. Empty string writes to the default rp.
retention_policy = ""

## Write consistency (clusters only), can be: "any", "one", "quorum", "all"
write_consistency = "any"

## Write timeout (for the InfluxDB client), formatted as a string.
## If not provided, will default to 5s. 0s means no timeout (not recommended).
timeout = "5s"

# username = "telegraf"
# password = "metricsmetricsmetricsmetrics"

## Set the user agent for HTTP POSTs (can be useful for log differentiation)
# user_agent = "telegraf"

## Set UDP payload size, defaults to InfluxDB UDP Client default (512 bytes)
# udp_payload = 512

และในส่วนของ Input Plugins นั้นก็ควรจะมีหน้าตาลักษณะแบบนี้ ในการดึงค่า System Stat ออกมายังเป็น Output ให้กับ InfluxDB

# Read metrics about cpu usage
[[inputs.cpu]]
    ## Whether to report per-cpu stats or not
    percpu = true
    ## Whether to report total system cpu stats or not
    totalcpu = true
    ## If true, collect raw CPU time metrics.
    collect_cpu_time = false

# Read metrics about disk usage by mount point
[[inputs.disk]]
    ## By default, telegraf gather stats for all mountpoints.
    ## Setting mountpoints will restrict the stats to the specified mountpoints.
    # mount_points = ["/"]
    ## Ignore some mountpoints by filesystem type. For example (dev)tmpfs (usually
    ## present on /run, /var/run, /dev/shm or /dev).
    ignore_fs = ["tmpfs", "devtmpfs"]

# Read metrics about disk IO by device
[[inputs.diskio]]
    ## By default, telegraf will gather stats for all devices including

```

```

## disk partitions.
## Setting devices will restrict the stats to the specified devices.
# devices = ["sda", "sdb"]
## Uncomment the following line if you need disk serial numbers.
# skip_serial_number = false
# Get kernel statistics from /proc/stat
[[inputs.kernel]]
# no configuration
# Read metrics about memory usage
[[inputs.mem]]
# no configuration
# Get the number of processes and group them by status
[[inputs.processes]]
# no configuration
# Read metrics about swap memory usage
[[inputs.swap]]
# no configuration
# Read metrics about system load & uptime
[[inputs.system]]
# no configuration

```

ซึ่งจริงๆแล้วทั้งในส่วนของ Output/Input Plugins นั้นก็เป็นค่า Default ที่ติดตั้งมาพร้อมกับ Telegraf อู郁แล้วครับ เสร็จแล้วก็ทดลองรันคำสั่งนี้กันดู curl "http://localhost:8086/query?q=select*+from+telegraf..cpu"

Chronograf Setup

มาถึง Module สุดท้ายของ TICK Stack กันเลยครับสำหรับหน้า GUI เพื่อใช้ในการ Config และแสดงผล สำหรับผู้ใช้งาน Grafana น่าจะสะดวกกว่า แต่ไหนๆก็ใหม่แล้ว Install กันให้ครบเลยดีกว่าครับ เป็นทางเลือก

Download & Install

```

wget https://dl.influxdata.com/chronograf/releases/chronograf_1.4.0.0_amd64.deb
sudo dpkg -i chronograf_1.4.0.0_amd64.deb

```

Start Service

```
sudo systemctl start chronograf
```

Verify

ขั้นตอนนี้ก็ให้เข้าไปที่ <http://localhost:8888>โดยเปลี่ยน local host เป็น ip หรือ hostname ของเราครับ ก็จะได้หน้า Web GUI ของ Chronograf ดังแสดงในรูปด้านล่างนี้ สำหรับใครที่ใช้ Google Cloud ก็อย่าลืมเข้าไปเช็คในส่วนของ Firewall Rule ให้ Allow Port 8888 ด้วยนะครับ วิธีการก็หาคู่ได้จากตอนที่ 3.5 ได้เลย

Add a New Source

Connection String: http://localhost:8086

Name: Influx 1

Username:

Password:

Telegraf Database: telegraf

Make this the default source

+ Add Source

หลังจากนั้นก็คลิกที่ปุ่ม +Add Source เลยครับ ค่าที่อยู่ให้คง Default ไว้ เนื่องจากในการติดตั้งครั้งแรกนั้น Default Security ของ InfluxDB นั้นจะไม่มี Username/Password ซึ่งเดียวเราจะได้มา config กันในช่วงต่อไปครับ ซึ่งหลังจาก Add Source แล้วก็จะเข้าสู่หน้าจอถัดไป ให้ทำการเลือก Host ที่เราใช้ Plugin ของ Telegraf ในการดึง System Stat มาใส่ไว้ใน InfluxDB

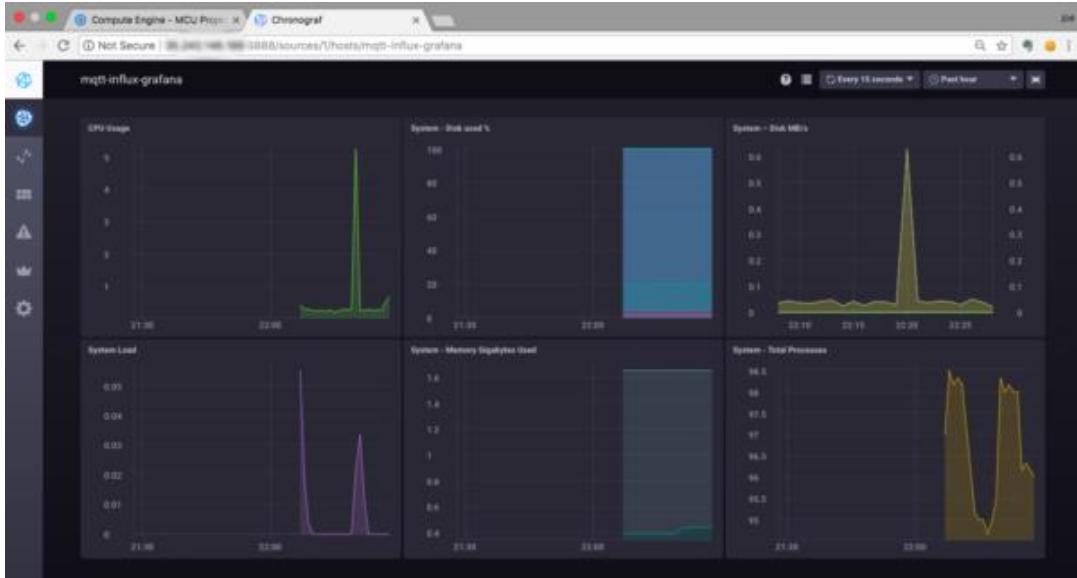
Host List

Host List 1. คลิกเพื่อเลือก Host

1 Host

Host	Status	CPU	Load	Apps
mqtt-influx-grafana	●	0.78%	0.01	system

2. เลือก Host



ถ้าทุกอย่างทำงานได้อย่างถูกต้องเราก็จะได้หน้า Dashboard ที่ใช้ในการแสดงผล System Stat ที่ได้จากการถึงข้อมูลผ่านทาง Plugin Module ของ Telegraf และโอนข้อมูลที่ได้ไปใน InfluxDB และนำมาแสดงผลบน Chronograf

แต่ยังไม่จบครับ เพราะ TICK Stack นั้นมีอยู่ด้วยกัน 4 Module นี่เราเพิ่งจะใช้งานไปแค่ 3 Module เอง ยังเหลือในส่วนของ Kapacitor ที่ใช้ในการทำ Data Processing/Analysis และ Alert ต่างๆ จากหน้า Dashboard System Stat ของเรา ก็คลิกที่ Config เพื่อ connect ไปยัง Kapacitor แต่ผมจะขอหยุดไว้เท่านี้ก่อนดีกว่า เพราะถ้าเราทำใน Grafana นั้นในส่วนของ Dashboard และ Alert เราสามารถทำได้จาก Grafana ในที่เดียวกันเลย เราจึงมักเห็นการจับคู่กันเฉพาะในส่วนของ MQTT -> Telgraf-> InfluxDB->Grafana เรียกได้ว่าเป็นสูตรสำเร็จสำหรับการเก็บข้อมูลมาแสดงผลของงาน IOT เลยครับ

แต่ก่อนอื่น รบماเซทระบบของเราให้เข้าที่เข้าทางกันก่อนดีกว่า เพราะก่อนหน้านี้ที่เรา Install InfluxDB ไปค่า Default ของระบบยังไม่มีในส่วนของ User/Password และการ Authen ต่างๆ ขึ้นตอนแรกเราจะสร้าง User Admin ของระบบกันด้วยคำสั่ง influx และตามด้วยคำสั่งด้านล่างนี้ครับ

```
CREATE USER "admin" WITH PASSWORD 'admin_passwd' WITH ALL PRIVILEGES
```

ในส่วนของ admin_passwd ก็เปลี่ยนເອົາຕາມໃຈขอบได้เลยครับ

```
Secure | https://ssh.cloud.google.com/projects/mcu-office-home-123/instances/mqtt-influx-grafana:/etc/influxdb
InfluxDB shell version: 1.4.0
Connected to http://localhost:8086 version 1.4.0
> CREATE USER "admin" WITH PASSWORD '...' WITH ALL PRIVILEGES
>
> SHOW USERS
user admin
-----
admin true
>
```

หลังจากนั้นก็ใช้คำสั่ง show users ในการเรียกดู user ทั้งหมดที่มีอยู่ ตามรูปด้านล่างนี้

โดยทำการแก้ไขไฟล์ /etc/influxdb/influxdb.conf ในส่วนของ [auth-enabled] ให้เป็น true ซึ่งหน้าตาของ config ที่แก้ไขจะเหมือนกับด้านล่างนี้

```
[http]
# Determines whether HTTP endpoint is enabled.
#enabled = true

# The bind address used by the HTTP service.
#bind-address = ":8086"

# Determines whether user authentication is enabled over HTTP/HTTPS.
#auth-enabled = true

# The default realm sent back when issuing a basic auth challenge.
#realm = "InfluxDB"

# Determines whether HTTP request logging is enabled.
#log-enabled = true

# Determines whether detailed write logging is enabled.
#write-tracing = false

# Determines whether the pprof endpoint is enabled. This endpoint is used for
# troubleshooting and monitoring.
#pprof-enabled = true

# Determines whether HTTPS is enabled.
#https-enabled = false
```

เมื่อเพิ่ม user admin และแก้ไขไฟล์ influxdb.conf แล้ว จากนั้นก็ restart service ของ influxdb ด้วยคำสั่ง sudo systemctl restart influxdb ก็เป็นอันเสร็จสิ้นการปรับแต่ง influxdb ของเราให้มีการ Authen ก่อนเข้าใช้งาน ถ้าทดลอง list user ในระบบด้วยคำสั่ง show users โดยที่ไม่มีการใส่ username/password ก็จะแสดง error แบบด้านล่างนี้ครับ

```
Secure | https://ssh.cloud.google.com/projects/mcu-office-home-123/instances/mqtt-influx-grafana:/etc/influxdb
InfluxDB shell version: 1.4.0
Connected to http://localhost:8086 version 1.4.0
> show users
ERR: unable to parse authentication credentials
Warning: It is possible this error is due to not setting a database.
Please set a database with the command "use <database>".
>
```

ตั้งนั้นการใช้งานหลังจากนี้ที่หน้า command line ให้เพิ่มคำสั่งในส่วนของ -username และ -password เข้าไปก่อนก็จะสามารถใช้งานได้ครับ

```
mqtt-influx-grafana: /etc/influxdb$ influx -username admin -password
Connected to http://localhost:8086 version 1.4.0
InfluxDB shell version: 1.4.0
> show users
user    admin
-----
admin true
>
```

คราวนี้เราจะมาแก้ไข config ในส่วนของ Telegraf กันเพื่อให้ไปดึงข้อมูลจาก MQTT Server ของเราที่ได้สร้างไว้กันตั้งแต่ตอนที่ 3.5 กัน โดยแก้ไขไฟล์ /etc/telegraf/telegraf.conf ในส่วนของ [[inputs.mqtt_consumer]] ตามด้านล่างนี้เลยครับ

[[inputs.mqtt_consumer]]

```
servers = ["xxx.xxx.xxx.xxx:1883"] #<-- IP ของ mqtt server
qos = 0
connection_timeout = "30s"
topics = ["env"] #<-- topic ที่เราต้องการดึงข้อมูลมา
username = "xxx" #<-- username ของ mqtt server
password = "xxx" #<-- password ของ mqtt server
data_format = "influx" #<-- รูปแบบของ Data ที่ต้องการดึง
```

มาตรฐานส่วนของ influx line format กันบ้าง ซึ่งรูปแบบของข้อมูลที่ Telegraf รองรับนั้นก็มีหลายรูปถึง JSON ด้วย ถ้าสนใจจะใช้ format อื่นๆจาก ลิงค์นี้ได้เลยครับ telegraf input data formats

เนื่องจากตอนนี้เราจะมีการเขียนข้อมูลลงฐานข้อมูลของ InfluxDB 2 ตัวด้วยกัน โดยตัวแรกก่อนหน้านี้คือ Default ที่มาจากการติดตั้ง Telegraf ซึ่งเป็น ข้อมูลพวก System Stat อีกอันก็จะเป็นข้อมูลที่เราได้มาจากการ Node ที่ส่งข้อมูลมาที่ MQTT Server (ในที่นี้จำลองส่งข้อมูลอุณหภูมิเข้ามา) จะนั้นเรา ต้องบอก Telegraf ว่าข้อมูลไหนจะเก็บลงฐานข้อมูลไหน โดยการใช้ namedrop, namepass ตามตัวอย่างด้านล่างนี้ครับ

[[outputs.influxdb]] #<-- ใช้เขียนข้อมูล System Stat ลง InfluxDB โดยเก็บที่นั่นข้อมูล telegraf

```
urls = ["http://localhost:8086"] # required
database = "telegraf" # required
retention_policy = ""
write_consistency = "any"
timeout = "5s"
username = "xxx" #<-- username ของ InfluxDB
password = "xxx" #<-- password ของ InfluxDB
namedrop = ["env*"] #<-- ให้ drop ข้อมูลที่ขึ้นต้นด้วย env ซึ่งเป็น measurement
[[outputs.influxdb]] #<-- ใช้เขียนข้อมูล System Stat ลง InfluxDB โดยเก็บที่นั่นข้อมูล telegraf
urls = ["http://localhost:8086"]
```

```

database = "envdb" # required
retention_policy = ""
write_consistency = "any"
timeout = "5s"
username = "xxx" #<-- username ของ InfluxDB
password = "xxx" #<-- password ของ InfluxDB
namepass = ["env*"] #<-- ข้อมูลที่เขียนด้วย env ซึ่งเป็น measurement ให้เขียนลงฐานข้อมูล
จากนั้นทำการ Restart Service
$ systemctl daemon-reload
$ sudo systemctl restart telegraf
$ sudo systemctl status telegraf
ส่วน influx line format นั้นก็ย่างที่สุดเลยครับ ยกตัวอย่าง ตามด้านล่างเป็นข้อมูลที่ Nodemcu ส่งไปที่ MQTT Server
env,location=RST temp_in=25.60,temp_out=25.40
หรือถ้าคริยังไม่มีข้อมูลส่งเข้าไปที่ MQTT Server ก็ทำการ inset ข้อมูลเองโดยใช้คำสั่งนี้ที่หน้า command line ของ influx เลยครับ INSERT
env,location=RST temp_in=25.60,temp_out=25.40
ซึ่งการเก็บข้อมูลใน InfluxDB จะแยกเป็น Measurement, TAG Key – TAG Value และ Field Key – Field Value จากตัวอย่างข้อมูลข้างบนนั้น ผม
เข้าหน้า command line แล้วใช้คำสั่งแสดงค่าดังต่อไปนี้

```

```

Secure | https://ssh.cloud.google.com/projects/mcu-office-home
instances/mqtt-influx-grafana?authuser=0&hl=en
[centos@mcu-01 ~]$ influx -username admin -password
Connected to http://localhost:8086 version 1.4.0
InfluxDB shell version: 1.4.0
> show databases
name: databases
name
name
-----
telegraf
internal
envdb
> use envdb
Using database envdb
> show measurements
name: measurements
name
name
-----
env
> show tag keys on "envdb"
name: env
tagKey
-----
host
location
topic
> show tag values on "envdb" with key ="topic"
name: env
key value
--- -----
topic env
> show tag values on "envdb" with key ="location"
name: env
key value
--- -----
location RST
> show field keys on "envdb" from "env"
name: env
fieldKey fieldType
----- -----
temp_in float
temp_out float
>

```

Measurement มีแค่ตัวเดียวคือ env

TAG Key ประกอบไปด้วย host, location, และ topic

และ temp_in, temp_out เป็น Field Key

Grafana

ขั้นตอนการติดตั้ง Grafana ซึ่งโดยปกติสามารถติดตั้งได้ผ่าน apt-get install แต่บางเครื่องนั้นอาจจะไม่ได้มีข้อมูล repository ของ Grafana อยู่ด้วย
จะนั้นเราต้องทำการ เพิ่มรายการของ Grafana เข้าไปในลิสต์ก่อนที่จะทำการติดตั้ง

curl https://packagecloud.io/gpg.key | sudo apt-key add -

จากนั้นตามด้วยคำสั่ง

```
sudo add-apt-repository "deb https://packagecloud.io/grafana/stable/debian/ stretch main"
```

```
sudo apt-get update
```

เมื่อทำการ Update List เรียบร้อยแล้วก็สามารถทำการติดตั้งได้ด้วยคำสั่ง sudo apt-get install grafana

หลังจากที่ติดตั้งเสร็จแล้วก็ทำการ Start Service และเช็คสถานะกันหน่อย

```
sudo systemctl start grafana-server
```

```
sudo systemctl status grafana-server
```

ถ้า Service ของ Grafana ทำงานเป็นปกติ ก็จะขึ้นข้อมูลแสดงสถานะดังนี้ครับ

```
mqtt-influx-grafana: /var/log/mosquitto
grafana-server.service - Grafana instance
  Loaded: loaded (/etc/systemd/system/grafana-server.service; disabled; vendor preset: enabled)
  Active: active (running) since Sun 2018-08-26 06:13:03 UTC; 37s ago
    Docs: http://docs.grafana.org
   Main PID: 17303 (grafana-server)
      Tasks: 7 (limit: 1997)
     CGroup: /system.slice/grafana-server.service
             └─17303 /usr/bin/grafana-server --config=/etc/grafana/grafana.ini --pidfile=/var/run/grafana/grafana-server.pid cfg:default.paths.logs

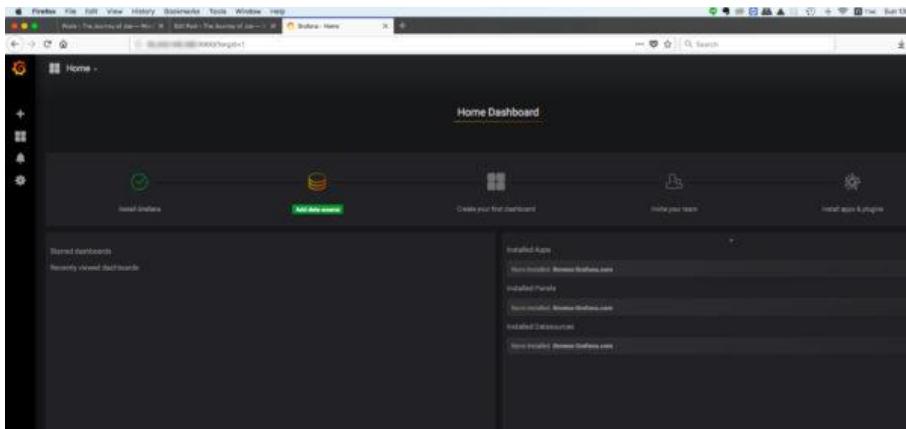
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing InternalMetricService" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing AlertingService" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing HTTPServer" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing CleanUpService" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing NotificationService" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing ProvisioningService" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing RenderingService" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing TracingService" logger=server
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Initializing Stream Manager"
Aug 26 06:13:05 mqtt-influx-grafana grafana-server[17303]: t=2018-08-26T06:13:05+0000 lvl=info msg="Server Listen" logger=http.server address=0.0.0.0:3000
lines 1-19/19 (END)
```

จากนั้นก็ตั้งค่าให้ Grafana เริ่มต้นทุกครั้งก็การรีบูตด้วยคำสั่ง sudo systemctl enable grafana-server

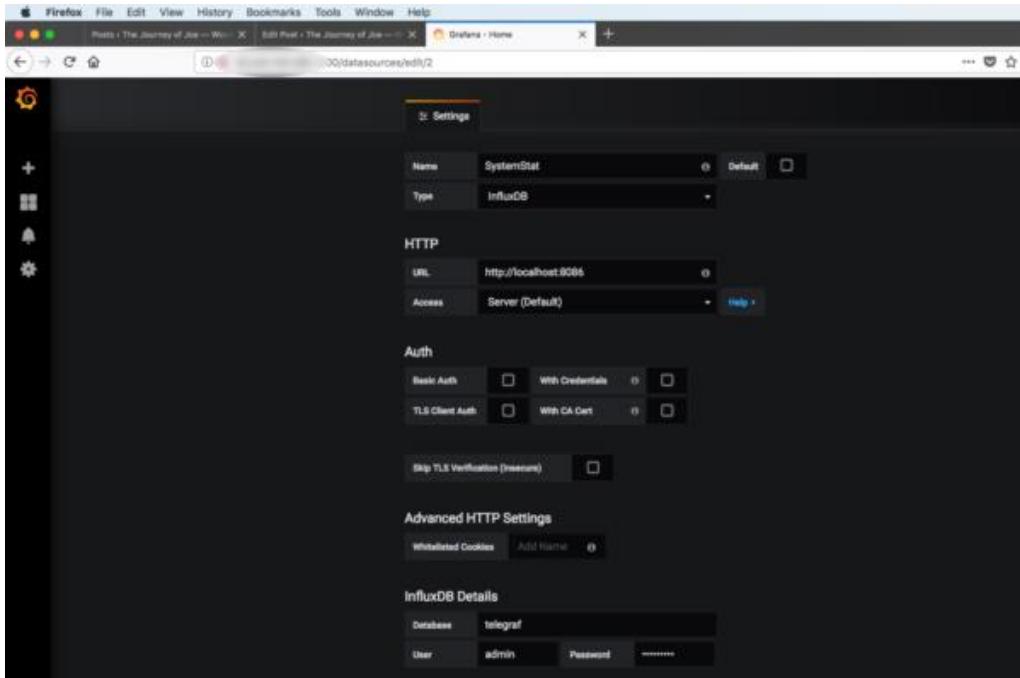
ซึ่งค่า Default Port ของ Grafana นั้นคือ port 3000 ดังนั้นถ้าหากใช้ Google Cloud ก็อย่าลืมไป allow port ให้ connection จากข้างนอกสามารถเข้าถึง VM Instance ของเราราจาก port 3000 ได้ด้วยนะครับ ซึ่งการใช้งานก็เข้าได้จากหน้า Browser เลย



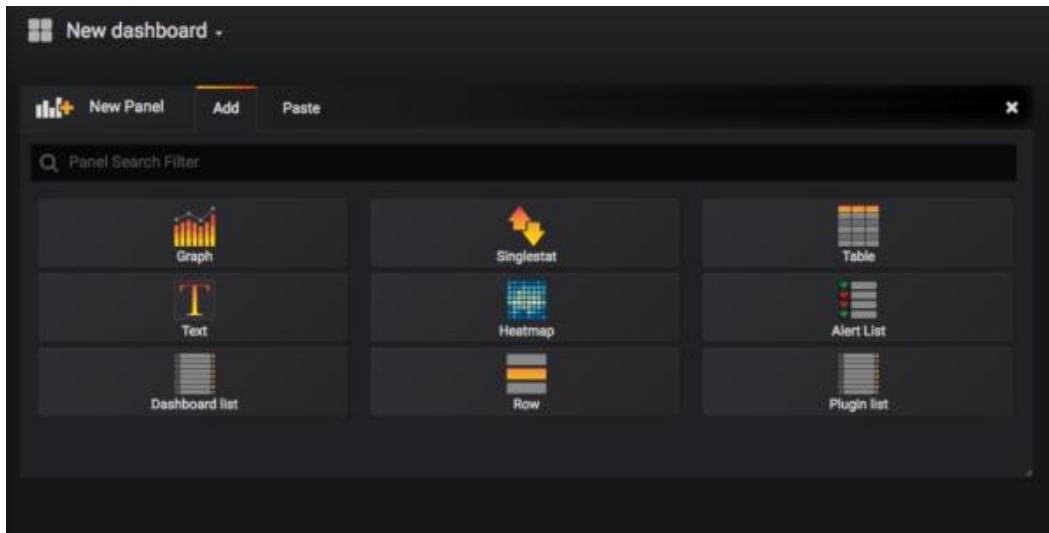
โดยที่ Default Username/Password ตั้งต้นเลยก็คือ admin/admin ครับ เมื่อ login เข้าไปแล้วระบบก็จะให้เราเปลี่ยน password ของ admin ใหม่ทันที ก็จะเข้าสู่หน้าต่อไป ซึ่ง Grafana นั้นก็ออกแบบขั้นตอนเป็นสเตปมาให้อย่างดี เมื่อติดตั้งเสร็จ ก็เพิ่ม Datasource จากนั้นก็สร้าง Dashboard ครับ จะใช้คันเดียวกันไม่ต้องกู้คืน ก็สามารถ Invite คนอื่นเข้ามาใช้งานร่วมกันได้ รวมถึงรองรับ Plugin ต่างๆด้วย



เราจะมา Add Data source และกัน ซึ่งก็คือ Data source System Stat ที่ได้มาจากการตั้งข้อมูลของ Telegraf ในตอนต้นนั้นเอง ให้คลิก Add Data Source จากนั้นก็เลือกมูลตามาที่เรา config ไว้



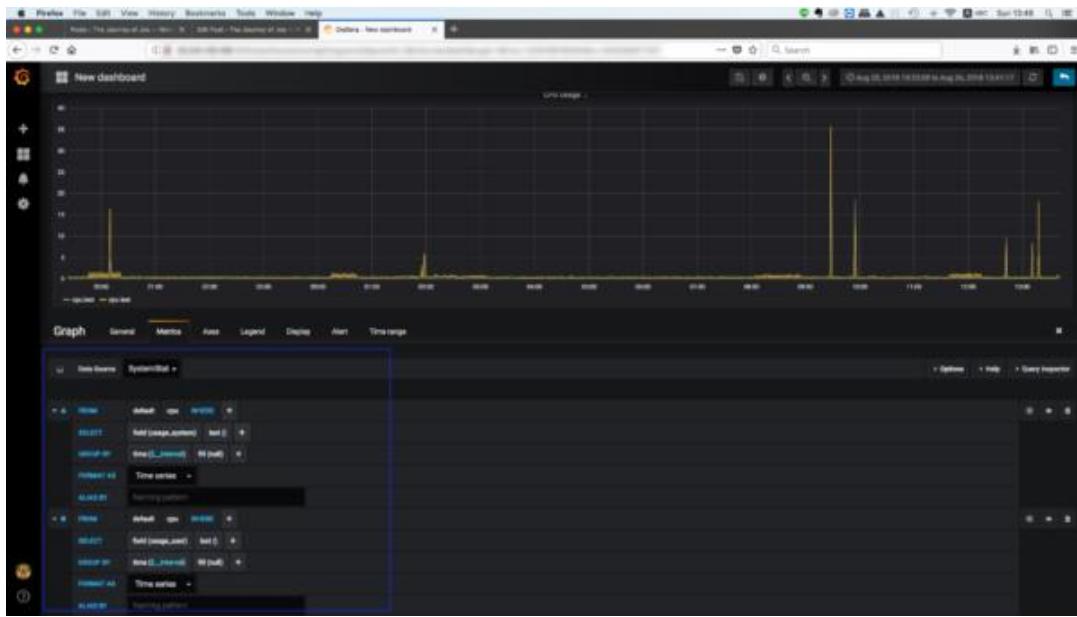
หลังจากที่เราเพิ่ม Data Source แรกของเราไปแล้ว ขั้นต่อไปก็เป็นการสร้าง Dashboard ของเรากัน โดยเข้าไปที่ New Dashboard และเพิ่ม Graph เข้าไปยัง Dashboard แรกของเรา



จากนั้นคลิกที่ชื่อ Panel แล้วเลือก Edit เพื่อทำการเลือก Data Source ว่าจะเอาข้อมูลไหนมาสร้างกราฟ



ให้เลือก Data Source System Stat ที่เราสร้างกันไว้ แล้วเลือกข้อมูลตามที่ต้องการrelayครับ จะเป็น CPU, Mem หรือ Disk i/o ก็ได้ โดยตัวอย่างผมดึง CPU Usage มาแสดงบนกราฟ



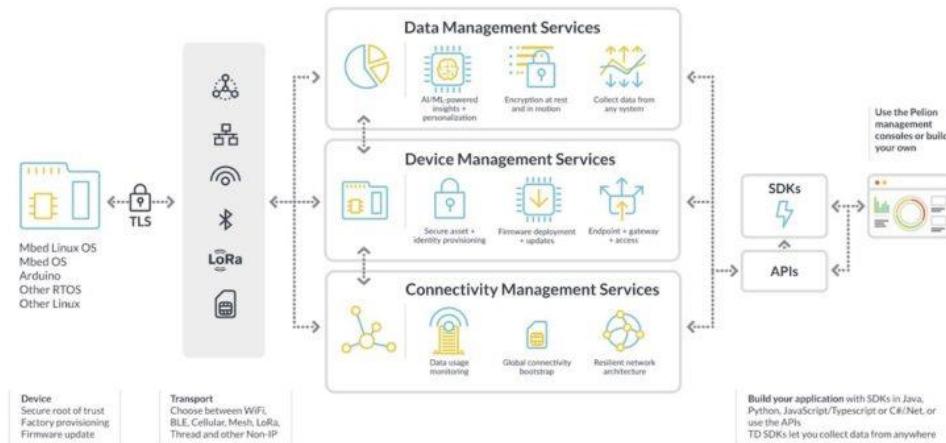
1b/3: Arm-Pelion Full Stack IoT Platform

- <https://www.techtalkthai.com/arm-pelion-full-stack-iot-platform/>

รู้จัก ARM Pelion แพลตฟอร์ม IoT จาก ARM จัดการทุกอย่างครบจบในที่เดียว เมื่อวันที่ 30 กรกฎาคมที่ผ่านมา ทาง ARM ได้จัด IoT Workshop ขึ้นเพื่อนำเสนอและมุ่งด้างๆของการนำเทคโนโลยี IoT ไปใช้ในธุรกิจ ทีมงาน TechTalkThai ได้เข้าไปร่วมงาน และทำความรู้จักกับ ARM Pelion IoT Platform ซึ่งอย่างจะมาเล่าให้ผู้อ่านฟังกันคร่าวๆว่าเจ้าแพลตฟอร์ม IoT นี้มีความสามารถอย่างไร และเหมาะสมกับธุรกิจแบบไหนบ้าง

IoT นั้นเป็นหนึ่งในเทคโนโลยีที่หลายองค์กรยกให้เป็นยุทธศาสตร์ในปี 2019 จากความสามารถในการรวบรวมข้อมูล ซึ่งนับว่าเป็นวัตถุในการทำธุรกิจที่ขาดไม่ได้เลยในยุคปัจจุบัน จนถึงตอนนี้ หลายองค์กรอาจเริ่มต้นกับ IoT กันบ้างแล้ว แต่ความท้าทายใหม่ที่ธุรกิจมักจะเผชิญกับก็คือการจัดการและสกัดระบบ IoT ให้ใช้งานเก็บข้อมูลได้จริงเต็มประสิทธิภาพ ปลอดภัย และมีระบบจัดการที่ดี ดังนั้นจึงมีการพัฒนาแพลตฟอร์ม IoT ขึ้น เพื่อช่วยธุรกิจในการแก้ปัญหานี้

ARM Pelion IoT Platform ที่เป็นหนึ่งในแพลตฟอร์ม IoT ที่จะเข้ามาช่วยลดความซับซ้อนของการนำ IoT ไปใช้งานในธุรกิจ แพลตฟอร์ม Pelion นี้แบ่งออกเป็น 3 ส่วน ตามการใช้งาน คือ Connectivity Management Services, Device Management Services, และ Data Management Services โดยทั้ง 3 ส่วนจะทำงานร่วมกันภายใต้ระบบบรักษาความปลอดภัย ซึ่งเป็นหลักสำคัญที่สุดในการพัฒนาผลิตภัณฑ์ทุกๆตัว ของ Arm



ภาพรวมของแพลตฟอร์ม Pelion ที่แบ่งการทำงานออกเป็น 3 ส่วน โดยธุรกิจสามารถเลือกใช้เพียงส่วนใดส่วนหนึ่งหรือทั้ง 3 ส่วนร่วมกันได้ (ภาพ: ARM)

Pelion จะช่วยให้ธุรกิจจัดการกับเครือข่าย อุปกรณ์ในเครือข่าย และข้อมูลที่เก็บมาได้ง่ายขึ้น โดยสามารถทำงานร่วมกับอุปกรณ์ ระบบเครือข่าย คลาวด์ และข้อมูลได้หลากหลายรูปแบบ อีกทั้งยังมีความปลอดภัย และสามารถช่วยในการนำข้อมูลไปวิเคราะห์และแสดงผลเบื้องต้นได้ด้วย

รู้จักแพลตฟอร์มนี้ไปคร่าวๆแล้ว ลองมาเจาะลึกกันว่าส่วนประกอบทั้ง 3 ส่วน คืออะไร กับ ConnectivityManager, DeviceManagement, และ DataManagement นั้นประกอบไปด้วยอะไร และมีจุดเด่นอย่างไรบ้าง

Connectivity Management

การเชื่อมต่อในเครือข่าย IoT นั้นมีอยู่หลายรูปแบบ และมีรายละเอียดปลีกย่อยที่ธุรกิจจะต้องจัดการอย่างพอดี Pelion จะช่วยให้องค์กรสามารถจัดการการเชื่อมต่อได้อย่างมีประสิทธิภาพ ปลอดภัย และพร้อมต่อการสเกลเครือข่ายขึ้นไปถึงระดับโลก โดย Connectivity Management ของ Pelion มีความสามารถที่น่าสนใจ ดังนี้

Global Cellular

Pelion จะช่วยให้อุปกรณ์ IoT สามารถเชื่อมต่อผ่านเครือข่ายได้ไม่ว่าอุปกรณ์นั้นจะอยู่ที่ใดในโลก ผ่านเวนเดอร์เพียงเจ้าเดียว โดยกลไกของ Pelion จะช่วยเชื่อมต่อสัญญาณจากชิมของอุปกรณ์ไปยังเครือข่ายท้องถิ่นที่ Pelion ได้ทำข้อตกลงไว้ ลดภาระความปวดหัวในการติดต่อกับผู้ให้บริการเครือข่ายในแต่ละประเทศ

Protocol เชื่อมต่อทั้ง IP และ Non-IP

นอกจากส่งข้อมูลผ่าน IP Network แล้ว Pelion ยังรองรับ Non-IP Network เช่น NB-IoT ด้วย โดยโปรโตคอลที่ Pelion รองรับนั้นมีได้แก่ MQTT(s), HTTPS, และ Sockets

ใช้ได้ทั้ง eSIM และชิมแบบปกติ

ธุรกิจสามารถสั่งผลิตอุปกรณ์ที่มีระบบ eSIM ผ่าน ARM ได้ตามต้องการ โดย eSIM ที่ติดมากับอุปกรณ์นั้นจะรองรับการเชื่อมต่อกับเครือข่ายกว่า 600 เครือข่ายทั่วโลก และหากต้องการเปลี่ยนเครือข่าย สามารถตั้งค่าใหม่ได้ภายในหลัง และในส่วนของชิมแบบปกติของ Pelion ก็ให้บริการชิมการ์ดในทุกขนาด อีกทั้งยังมีแผนที่จะพัฒนาไปจนถึง iSIM ที่มีขนาดเล็กกว่า eSIM มากด้วย

Network Infrastructure

Pelion ได้พัฒนาโครงสร้างพื้นฐานของเครือข่ายให้สามารถทำงานร่วมกับผู้ให้บริการเครือข่ายทั่วโลกได้อย่างมีประสิทธิภาพสูงสุด โดยมีทั้งความเสถียร ยืดหยุ่น และเป็นไปตามกฎข้อบังคับด้านข้อมูลของแต่ละประเทศ ในกรณีที่ Pelion ผู้ใช้สามารถเลือกได้ว่าจะส่งข้อมูลจากอุปกรณ์ไปยังแอปพลิเคชันผ่านเทคโนโลยีใด เช่น IPSEC, Open VPN, ผู้ให้บริการ Cloud, หรือทางเชื่อมที่ธุรกิจเข้ามาใช้โดยเฉพาะ (Leased Line)

Device Management

Device Management ของ Pelion นั้นจะช่วยให้องค์กรสามารถจัดการกับอุปกรณ์และการเชื่อมต่อกับอุปกรณ์ผ่านซอฟต์แวร์ได้โดยสะดวก ไม่ว่าจะเป็นการเขียนซอฟต์แวร์แบบ Embedded หรือว่าการเขียนแอปพลิเคชันด้านบนอย่าง Web App ก็ตาม



โดยภายในโซลูชัน Device Management ก็จะมีโมดูลในการจัดการเรื่องต่างๆให้อย่างครบถ้วน ตั้งแต่เรื่องการอัพเดท Firmware ซึ่งไม่ง่ายเลย หากมีอุปกรณ์ที่หลากหลายและมีจำนวนที่มากในเครือข่าย, Access Management ซึ่งจะช่วยจำกัดการเข้าถึงอุปกรณ์และการควบคุมแต่ละส่วน, Connector ซึ่งจัดการการเชื่อมต่อกับอุปกรณ์หลากหลายประเภท การออก Certificate เข้าใช้ระบบ การเข้ารหัสเปลี่ยนอุปกรณ์แต่ละตัว และการเก็บสถิติ, Device Directory ซึ่งช่วยในการแบ่งกลุ่ม ค้นหา เรียกดู และเช็คสถานะของอุปกรณ์แต่ละตัว, ไปจนถึงการรักษาความปลอดภัยในการส่งต่อรหัสผ่านเครือข่าย WiFi

โดยทั้งหมดนี้จะเห็นได้ว่าเป็นการจัดการ Lifecycle ของอุปกรณ์ทั้งหมด ตั้งแต่การ Onboard เข้าระบบ ไปจนถึงการใช้งานและบำรุงรักษา

Device Management นั้นสามารถพูดคุยกับอุปกรณ์ IoT ผ่านโปรโตคอลหลากหลาย โดยเฉพาะ LwM2M ซึ่งช่วยให้นักพัฒนาเขียนต่อ กับ อุปกรณ์ได้ผ่านโมเดลที่มีลักษณะคล้ายๆ REST Model และ CoAP ซึ่งจะช่วยประหยัดแบตเตอรี่ของอุปกรณ์ได้มากกว่า HTTP ระหว่าง 8-10 เท่า และในการเชื่อมต่อกับแอปพลิเคชันซึ่งเป็นปลายทางอีกด้านหนึ่ง Pelion ที่ได้เตรียม REST API และ SDK ในภาษา Java, Python, JavaScript และ .NET ไว้ให้พัฒนาแอปพลิเคชันกันได้โดยง่าย

Device Management ของ Pelion นี้รองรับการทำงานร่วมกับฮาร์ดแวร์ที่หลากหลาย ไม่ว่าจะเป็นอุปกรณ์แบบ Bare metal (มีระบบเชื่อมต่อที่เรียกว่า Edge รองรับ) และการทำงานร่วมกับระบบปฏิบัติการทั้ง Mbed OS และ Linux

Data Management

เป้าประสงค์หลักของการจัดตั้งระบบ IoT นั้นคือการสร้างระบบจัดเก็บข้อมูลที่จะช่วยให้องค์กรสามารถเรียกข้อมูลเหล่านั้นขึ้นมาวิเคราะห์เป็นความรู้ที่มีประโยชน์ต่อธุรกิจได้ แน่นอนว่า Pelion ย่อมไม่ลืมความสำคัญของส่วนนี้ จึงได้พัฒนาระบบจัดการข้อมูลครบวงจรที่จะช่วยตั้งแต่การจัดเก็บ นำข้อมูลมาใช้ตัดสินใจแบบ Real-time และรักษาความปลอดภัยและความเป็นส่วนตัวของข้อมูล โดยมีกลไกรองรับการสเกลเต็มที่ ทำให้องค์กรไม่ต้องกังวลว่าระบบจะทำงานได้แย่ลงหากมีข้อมูลหรืออุปกรณ์ในเครือข่าย IoT เพิ่มมากขึ้นเมื่อเวลาผ่านไป

โซลูชันหลักของส่วนนี้ คือ ARM Treasure Data ซึ่งเป็นซอฟต์แวร์จัดการและวิเคราะห์ข้อมูลที่เขียนต่อมาจาก Pelion ได้จับครบในตัวเดียว โดยมีเครื่องมือต่างๆพร้อมให้เลือกใช้งาน เช่น ระบบ Predictive Analytics การสร้าง Customer View 360 องศาจากข้อมูลการใช้งาน การวิเคราะห์ข้อมูลเพื่อ Cross-sell และ Upsell และการสร้างระบบ Recommendation เป็นต้น ซึ่งโซลูชันนี้หลายๆองค์กรก็ได้นำไปใช้งานพิมประสิทธิภาพให้กับการทำงานในอุตสาหกรรมมากมาย เช่น อุตสาหกรรมค้าปลีก อุตสาหกรรมพลังงาน อุตสาหกรรมการผลิต และอุตสาหกรรมอื่นๆอีกมาก

แพลตฟอร์ม Pelion นั้นปัจจุบันได้มีการนำไปใช้งานกับระบบ IoT ทั้งในประเทศขนาดใหญ่และขนาดเล็ก เช่น ระบบ IoT ในการดูแลสัตว์น้ำผ่านเข็มเจาะรับข้อมูลจากเสียง ระบบตรวจสอบสถานะการทำงานของเครื่องจักรในโรงงาน ระบบจัดการคลังสินค้า และระบบซึ่งเป็นส่วนประกอบของ Smart City เช่น ที่จอดรถอัจฉริยะ และเสาไฟฟ้าที่เปิดปิดตามความเคลื่อนไหวของคน และสามารถควบคุมได้จากระบบส่วนกลาง เป็นต้น

ท่านใดที่สนใจศึกษาเกี่ยวกับ Pelion เพิ่มเติม สามารถเข้าไปอ่านเกี่ยวกับกรณีศึกษาการใช้งานในอุตสาหกรรมได้ที่ <https://www.arm.com/products/iot/pelion-iot-platform> และหากต้องการข้อมูลเชิงเทคนิคโดยละเอียด สามารถอ่าน Document เดิมๆ ตามลิงก์นี้ <https://www.pelion.com/docs/>

สำหรับในประเทศไทย ARM ได้จับมือเป็นพาร์ทเนอร์กับ Advantech ในบริการด้านต่างๆ ท่านที่สนใจสามารถติดต่อเพื่อพูดคุยถึงโซลูชัน และผลิตภัณฑ์เกี่ยวกับ IoT ของ ARM ได้ที่อีเมล chanchai.p@advantech.com

1c/3: Grafana Dashboard

- <https://developers.ascendcorp.com/%E0%9C%A8%E0%9E%9E%E0%9E%9B%E0%9E%9E-%E0%9E%9Agrafana-dashboard-1a5efe6d170a>

Grafana คือ open source Dashboard tool เรียกว่า ฯ ก็คือเครื่องมือในการสร้าง Dashboard ฟรี นั่นเอง โดย Grafana จะทำงานร่วมกับ Datasource ต่าง ๆ เช่น Graphite, InfluxDB, OpenTSDB หรือ Elasticsearch ฯลฯ ช่วยให้ users สามารถสร้างและแก้ไข Dashboard ได้อย่างง่ายๆ ครอบคลุมรูปแบบกราฟหลากหลายประเภท

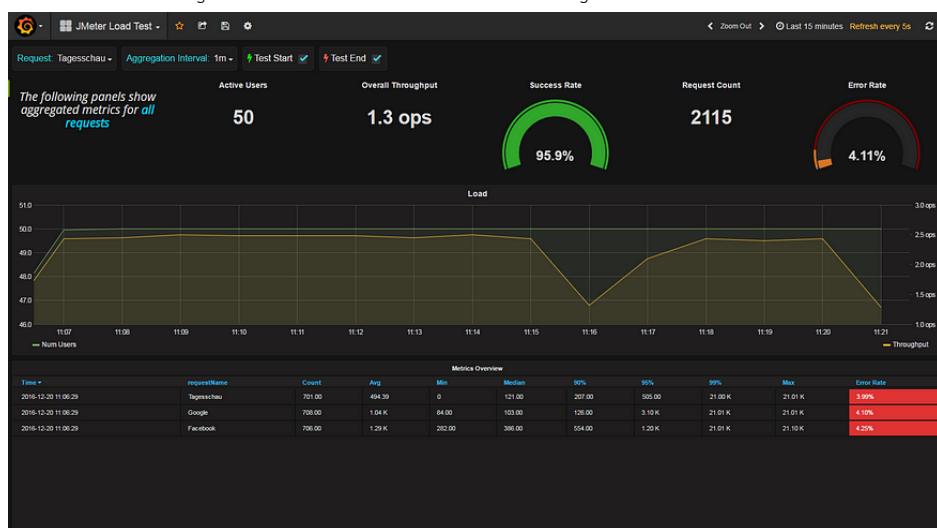
จุดเด่นของ Grafana

เน้นการนำเสนอ Metrics ที่เฉพาะเจาะจง เช่น CPU, Memory หรือ I/O ในรูปแบบของกราฟ Time series มี Role-based access ในการจัดการ user ในการเข้าใช้งานให้ในตัว ความยืดหยุ่นในการใช้งาน มี option ให้เลือกใช้จำนวนมาก รองรับ datasource ที่หลากหลายและมี query editor ที่สำหรับ datasource นั้นๆ ตัวอย่างการใช้งาน Grafana Dashboard

- Monitoring Server ใช้งานร่วมกับ Influxdb และ Telegraf

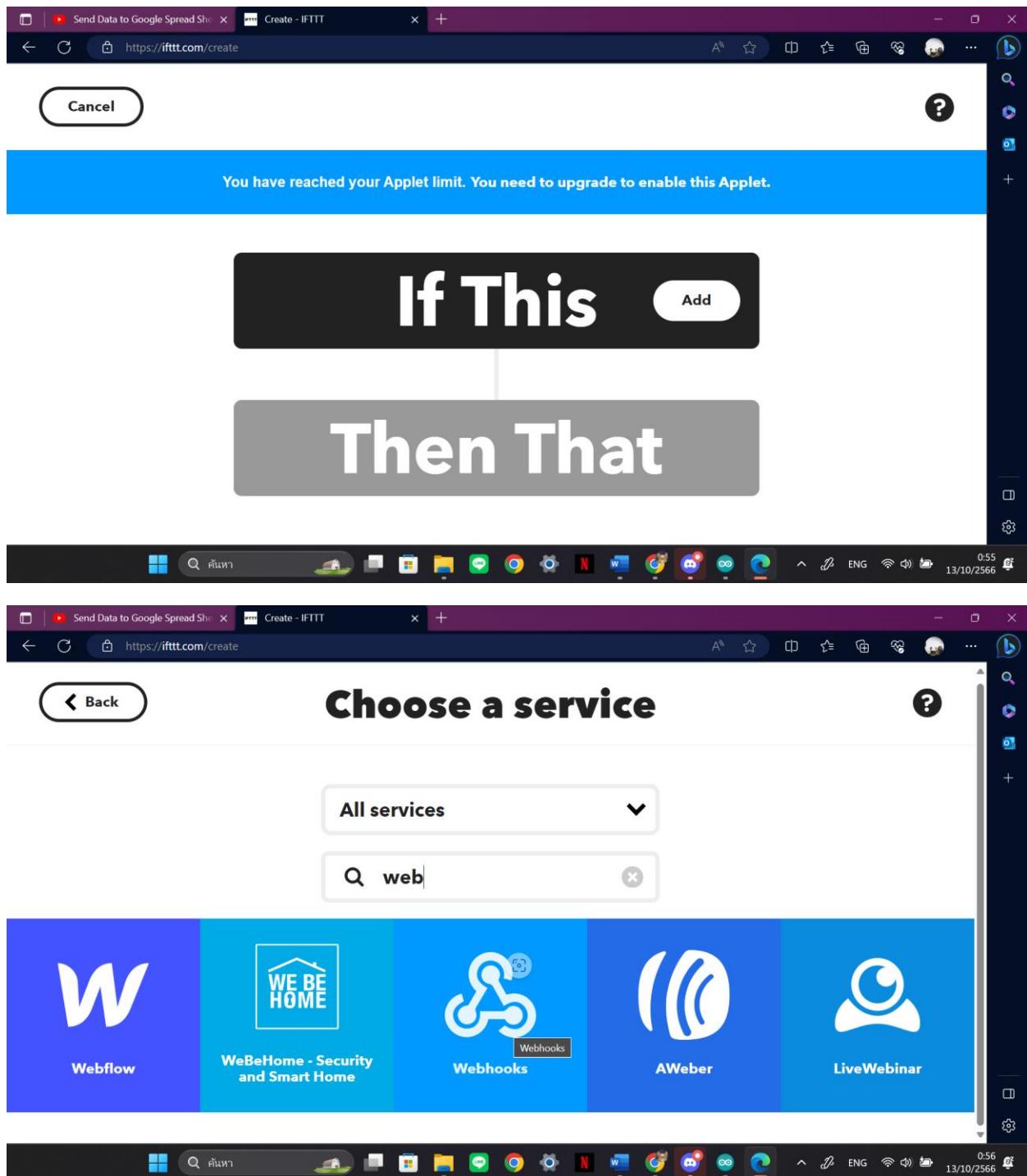


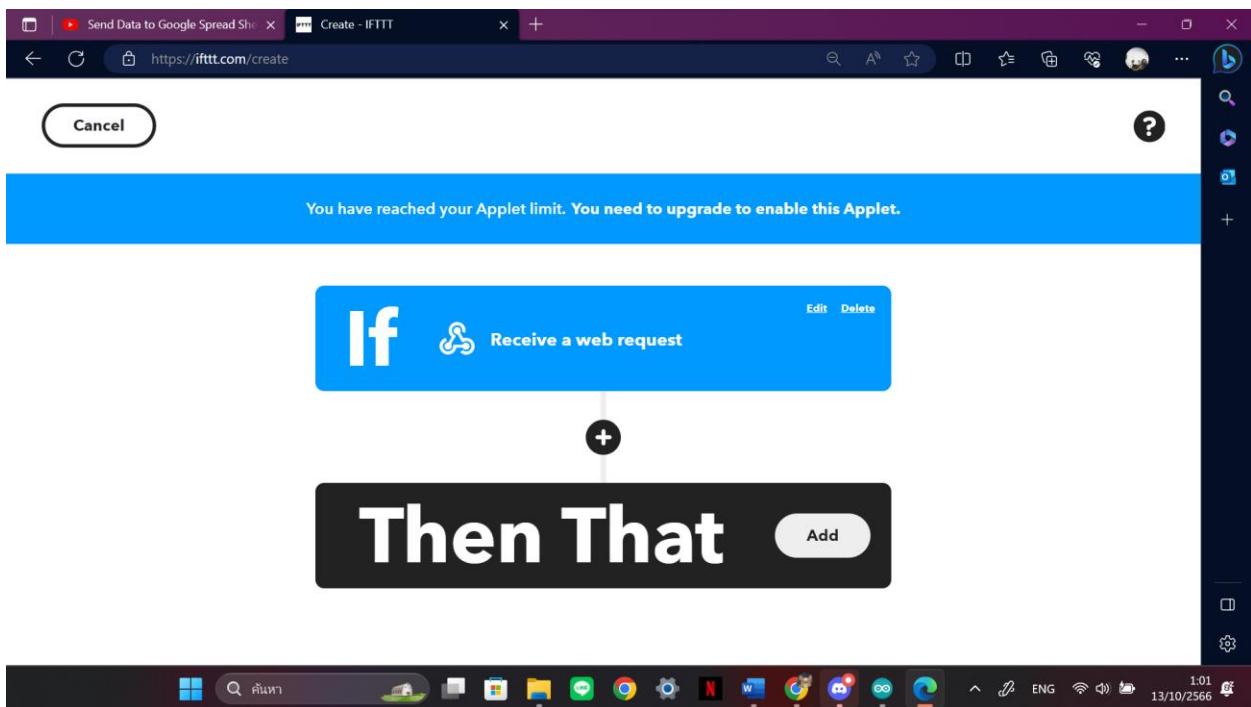
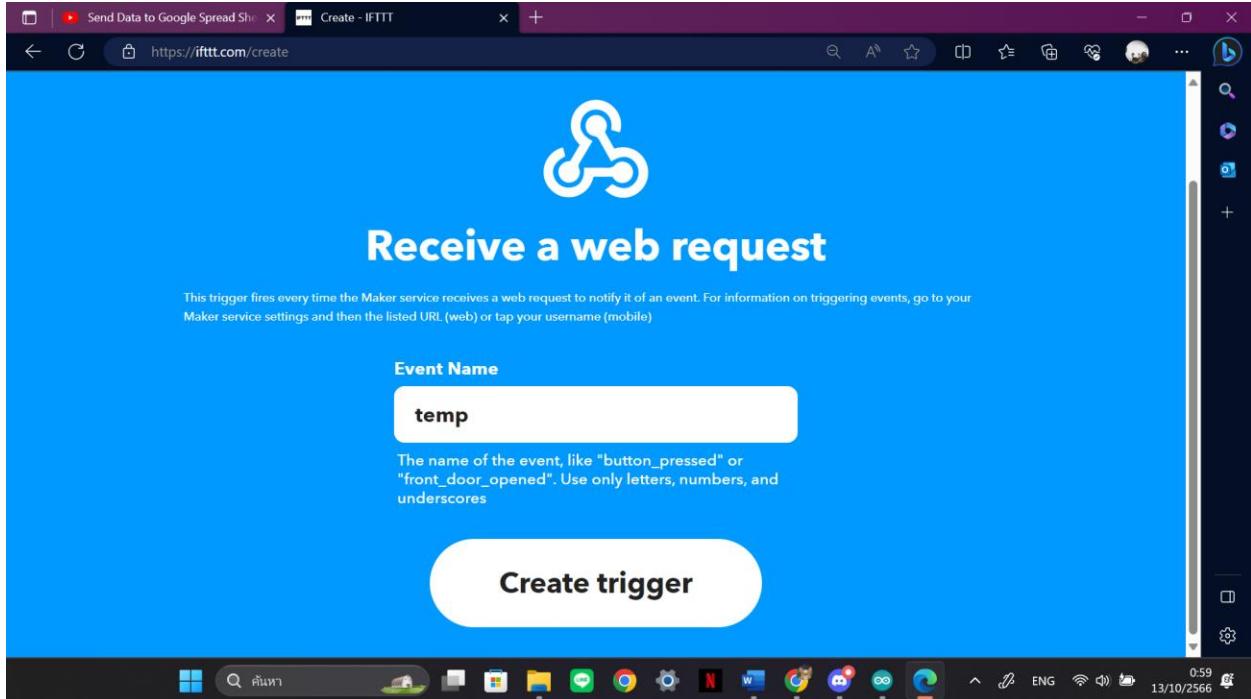
- Monitoring Realtime result สำหรับ Jmeter ใน non-gui mode

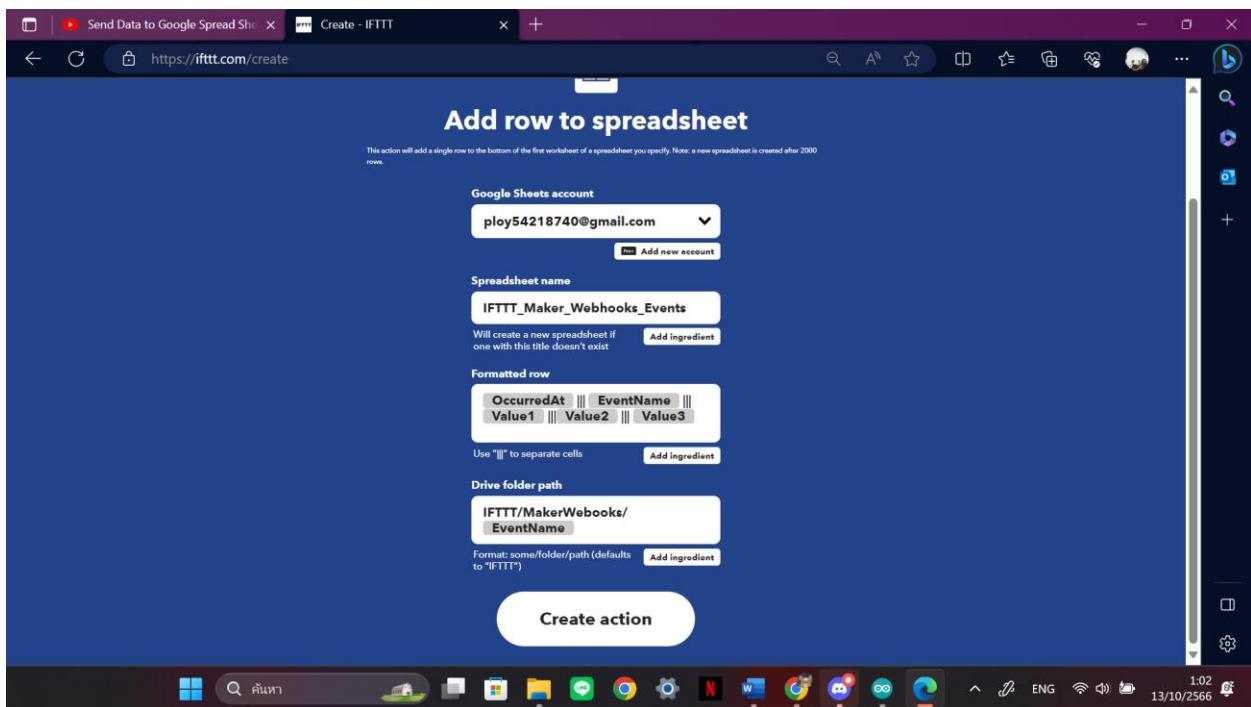
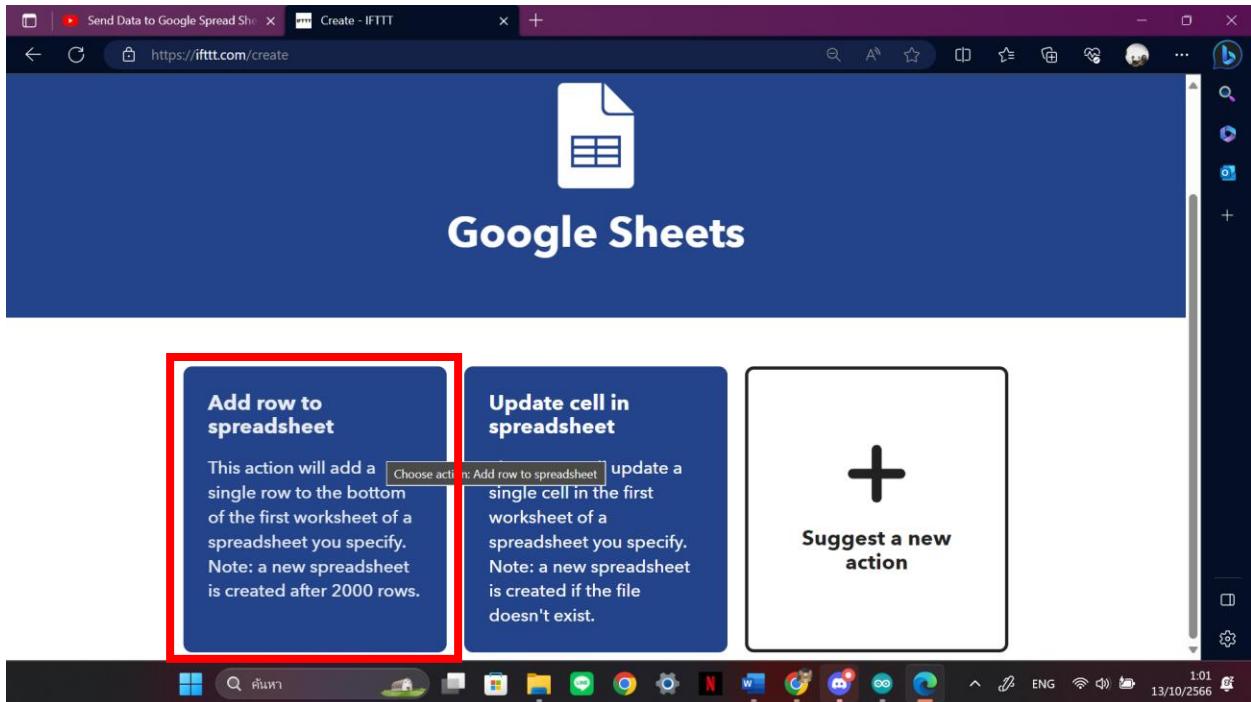


2/3 การทดสอบ -- ESP32 to Google Spreadsheet

- ให้ทำการทดสอบและเขียนขั้นตอนการทดสอบ โดยใช้ ESP32 ส่งข้อมูลไปยัง Google Spreadsheet







The image consists of two vertically stacked screenshots of the IFTTT web interface.

Screenshot 1: Review and finish screen

This screen shows the final step of creating an applet. The title of the applet is "If Maker Event "temp", then Add row to ploy54218740@gmail.com's Google Drive spreadsheet". A note indicates it was created by nannapat.srimartpirom on Oct 13, 2023. A toggle switch for "Receive notifications when this Applet runs" is turned off. A large "Finish" button is centered at the bottom.

Screenshot 2: My Applets screen

This screen displays the created applet under the heading "Connected". The title is the same as in the first screenshot. It shows the creation date (Oct 13, 2023) and run status (Never run). A "Connected" button is visible. Below the applet, there are options to "Notify me when this runs", "Check the log of your Applet runs", and a "View activity" button. The IFTTT navigation bar at the top includes "Explore", "Solutions", "Developers", "My Applets", "Create", and "Upgrade". The Windows taskbar at the bottom shows various pinned icons and the date/time (13/10/2566).

The screenshot shows the IFTTT Webhooks Integrations page. At the top, there are tabs for "Send Data to Google Spread Sheet" and "Webhooks Integrations - Conn...". The URL is https://ifttt.com/maker_webhooks. The main header is "Webhooks integrations". Below it, there is a brief description of what webhooks are and how to use them. A "Documentation" button is highlighted with a red box. Other buttons include "Create" and "Settings". Below the main header, there are sections for "Popular Webhooks workflows & automations" featuring cards like "Launch outgoing calls through Maker", "Get an email when Webhooks publishes a new", and "Get an email when a new Webhooks Applet is". The bottom of the screen shows a Windows taskbar with various icons.

The screenshot shows the IFTTT Maker Webhooks page. The URL is https://maker.ifttt.com/use/IQXbISU6TdbCZoDfiGc1jBaoNk4mmBhRL7N61EJhci. It displays the generated key: IQXbISU6TdbCZoDfiGc1jBaoNk4mmBhRL7N61EJhci. Below the key, there is a section titled "To trigger an Event with an arbitrary JSON payload". It provides instructions for making a POST or GET web request to https://maker.ifttt.com/trigger/{event}/json/with/key/IQXbISU6TdbCZoDfiGc1jBaoNk4mmBhRL7N61EJhci. It notes the extra /json path element. It also shows a sample JSON body and a curl command for triggering the event from the command line. The bottom of the screen shows a Windows taskbar with various icons.

Event has been triggered.

```
{ "value1" : " 22 ", "value2" : " 33 ", "value3" : " 44 " }
```

The data is completely optional, and you can also pass value1, value2, and value3 as query parameters or form variables. This content will be passed on to the action in your Applet.

You can also try it with curl from a command line.

```
curl -X POST -H "Content-Type: application/json" -d '{"value1":"22","value2":"33","value3":"44"}' https://maker.ifttt.com/trigger/{event}/with/key/lQXbISU6TdbCZoDfiGc1jBbaoNk4mmBhRL7N61EJhci
```

Please read our [FAQ](#) on using Webhooks for more info.

Test It

To query a web service

You can query a publicly accessible HTTP endpoint using the Webhooks service.

The "Make a web request" query requires a URL and Method as query fields. The query optionally accepts a Content Type and Request Body as query fields.

The query will always provide the Status Code returned by the endpoint as an Ingredient.



Code ໃນ Arduino

```
#include <WiFi.h>

#include <HTTPClient.h>

#include "DHTesp.h"

DHTesp dht;

#define pinDHT22 15

const char * ssid = "AndroidAP33";

const char * password = "ydvv5842";

String server = "http://maker.ifttt.com/";

String eventName = "temp";

String IFTTT_Key = "lQXbISU6TdbCZoDfiGc1jBbaoNk4mmBhRL7N61EJhci";

String IFTTTUrl="https://maker.ifttt.com/trigger/temp/with/key/lQXbISU6TdbCZoDfiGc1jBbaoNk4mmBhRL7N61EJhci";
```

```
int value1;

int value2;

void setup() {
    Serial.begin(115200);
    dht.setup(pinDHT22, DHTesp::DHT22);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Viola, Connected !!!");
}

void sendDataToSheet(void)
{
    String url = server + "/trigger/" + eventName + "/with/key/" + IFTTT_Key + "?value1=" + String((int)value1) +
    "&value2=" + String((int)value2);
    Serial.println(url);

    //Start to send data to IFTTT
    HTTPClient http;
    Serial.print("[HTTP] begin...\n");
}
```

```
http.begin(url); //HTTP

Serial.print("[HTTP] GET...\n");

// start connection and send HTTP header

int httpCode = http.GET();

// httpCode will be negative on error

if(httpCode > 0) {

    // HTTP header has been send and Server response header has been handled

    Serial.printf("[HTTP] GET... code: %d\n", httpCode);

    // file found at server

    if(httpCode == HTTP_CODE_OK) {

        String payload = http.getString();

        Serial.println(payload);

    }

} else {

    Serial.printf("[HTTP] GET... failed, error: %s\n", http.errorToString(httpCode).c_str());

}

http.end();

}

void loop() {

    value1 = dht.getHumidity();

    value2 = dht.getTemperature();
```

```
Serial.print("Values are ");  
Serial.print(value1);  
Serial.print(' ');  
Serial.print(value2);  
Serial.print(' ');\n  
sendDataToSheet();  
delay(10000);  
}
```

หน้าต่าง googleSheet

	A	B	C	D
1	October 12, 2023 at 1 temp	32	70	
2	October 12, 2023 at 1 temp	32	68	
3	October 12, 2023 at 1 temp	32	69	
4	October 12, 2023 at 1 temp	31	70	
5	October 12, 2023 at 1 temp	32	70	
6				
7				
8				

2/3 การทดสอบ -- ESP32 to Influx DB with Grafana Dashboard Display

- ให้ทำการทดสอบและเขียนขั้นตอนการทดสอบ โดยใช้ ESP32 ส่งข้อมูลไปยัง Influx DB และใช้ Grafana ในการมองเห็นข้อมูล
- อ้างอิงการทดสอบจาก <https://gabrieltanner.org/blog/grafana-sensor-visualization>
- หรือเอกสารรุ่นพี่ที่เคยทำไว้ก่อน

Step-1: Test DHT-22, GY-302 and Rotary Encoder

1. DHT-22 → <https://www.cybertice.com/p/697>
 - Read Temperature → DHT_Tempp
 - Read Humidity → DHT_Humid
2. GY-302 → <https://www.cybertice.com/p/784>
 - Ambient Light → Ambient Intensity
3. Rotary Encoder → <https://www.cybertice.com/p/2465>
 - Read Encoder → Position

Step-2: InfluxDB → Bucket Name = “pk007.sut's Bucket”

5. <https://www.influxdata.com/get-influxdb/> → Login to influxDB Cloud 2.0

The screenshot shows a web browser window with the URL <https://www.influxdata.com/get-influxdb/> in the address bar. The page features the InfluxData logo and navigation links for Products, Developers, Customers, Pricing, Contact Us, and a prominent 'Get InfluxDB' button. A dropdown menu is open under the 'Login' link, showing three options: 'Login to InfluxDB Cloud 2.0' (which is highlighted with a cursor), 'Login to InfluxDB Enterprise', and 'Login to InfluxDB Cloud 1.x'. The main heading 'Get InfluxDB' is displayed in large blue text.

6. Register InfluxDB
7. Create Bucket
8. Create Tokenkey

Step-3: ESP32 Code

9. Add Lib

10. Code write data to Influx DB

11. Influx DB Monitor



Step-4: Grafana ➔ Dashboard Name = ["Pk007"](#)

12. <https://grafana.com/> ➔ Create free account (or Sign in)

The screenshot shows the official Grafana Labs website. At the top, there's a navigation bar with links for Products, Open source, Solutions, Learn, Company, a search icon, Downloads, Contact us (in a blue button), and Sign in. Below the header, the main title is "Your K8s mo| stack" with the subtitle "Operational dashboards for your data here, there, or anywhere". To the left, there's a promotional section for the "free forever plan" which includes Grafana, 10K series, Prometheus metrics, 50GB logs, and 50GB traces. It features a "Create free account" button and a note "(No credit card required)". To the right, there's a large image of a dashboard interface showing various metrics, logs, and traces. The dashboard has three main sections: Metrics (yellow), Logs (orange), and Traces (red). A green circular icon with a square symbol is overlaid on the dashboard image.

13. Import Data
14. Create Dashboard
15. Show ➔ BMP280{Temperature, Pressure, Altitude} + DHT-11{Temperature, Humidity}
16. Show ➔ DHT-22{Temperature, Humidity} + GY-302{Ambient} + Rotary Encoder {Position}



The image shows two screenshots of the InfluxDB website. The top screenshot displays the homepage with the headline "InfluxDB. It's About Time." and a subtext about real-time insights from time series data. A purple "Get InfluxDB" button is highlighted with a red oval. The bottom screenshot shows the "Create your Free InfluxDB Cloud Serverless Account" sign-up page, featuring social login options for Google and Microsoft, and fields for First Name and Last Name.

InfluxDB Times Series Data Platf... influxdata.com

Products Solutions Developers Pricing Contact Us Log In Get InfluxDB

InfluxDB. It's About Time.

Real-time insights from any time series data with a single, purpose-built database. Run at any scale in any environment in the cloud, on-premises, or at the edge.

Get InfluxDB Find the right product →

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts. [Cookie Policy](#) [Privacy Policy](#)

Do you want to learn more, contact support or talk to a rep? Engage here:

Get InfluxDB

Cloud2.influxdata.com/signup?_gl=1*atkgay*_ga*NzQ1MTAwMjg0LjE2OTcxMjYzMDC.*_ga_CNWQ54SDD8*MTY5NzEyNjMwNy4xLjAuMTY5NzEy...

Create your Free InfluxDB Cloud Serverless Account

No credit card required

Continue with

Google Microsoft

OR

LOG IN SIGN UP

First Name* Last Name*

23:00 ENG 12/10/2566

Resource Center

What would you like to do?

- Manage Databases & Security**
Create and manage your buckets (databases) & access tokens.
- Add Data**
Write data into your database with our reporting agent, programmatically, API, CLI, or upload a CSV or Line Protocol File.
- Query Data**
Query your data with the UI, programmatically, or integrate with 3rd party tools.
- Visualize & Alert**
Integrate with 3rd party tools to visualize your data or set up alerts.

What's New

★ Announcing InfluxDB Clustered
InfluxDB Clustered is the solution for organizations needing control over their data and underlying infrastructure. Clustered turns any InfluxDB instance into a production-ready cluster that can run on-premises or in your private cloud. Keep your time series data where you need it.

[LEARN MORE HERE](#)

Source Center

What would you like to do?

- Buckets**
- Sources
- Telegraf
- API Tokens

Manage Databases & Security
Create and manage your buckets (databases) & access tokens.

Add Data
Write data into your database with our reporting agent, programmatically, API, CLI, or upload a CSV or Line Protocol File.

Query Data
Query your data with the UI, programmatically, or integrate with 3rd party tools.

Visualize & Alert
Integrate with 3rd party tools to visualize your data or set up alerts.

What's New

★ Announcing InfluxDB Clustered
InfluxDB Clustered is the solution for organizations needing control over their data and underlying infrastructure. Clustered turns any InfluxDB instance into a production-ready cluster that can run on-premises or in your private cloud. Keep your time series data where you need it.

[LEARN MORE HERE](#)

Latest Blogs

Getting Started with Infrastructure Monitoring
OCT 11 This article was originally published on The New Stack and is reposted here with permission. By taking advantage of monitoring data, companies can ensure their infrastructure is performing optimally while reducing costs. While building new features

The screenshot shows the InfluxDB Times Series Data Platform's Load Data interface. At the top, there are tabs for SOURCES, BUCKETS (which is selected), TELEGRAF, and API TOKENS. Below the tabs is a search bar labeled 'Filter buckets...' and a dropdown menu set to 'Sort by Name (A → Z)'. On the left, there's a sidebar with navigation icons and a list of buckets: 'ployP' (Retention: 7 days, Schema Type: Implicit, ID: 6ef419edd070da12), '_monitoring', and a question mark icon. The main area displays two buckets: 'ployP' and '_monitoring'. To the right of the buckets is a purple sidebar with the title 'What is a Bucket?' and a description explaining that a bucket is a named location where time series data is stored, with retention periods. Below this is a link to 'How to write data into your bucket'. At the bottom right of the main dashboard, there is a red circle highlighting the '+ CREATE BUCKET' button.

This screenshot shows the 'Create Bucket' dialog box overlaid on the main Load Data interface. The dialog has fields for 'Name*' (containing 'jj') and 'Data Retention Preferences' (set to 'DELETE OLDER THAN' and '7 days'). At the bottom right of the dialog, there are 'CANCEL' and 'CREATE' buttons, with the 'CREATE' button being highlighted by a red circle.

The image shows two screenshots of the InfluxDB Times Series Data Platform interface.

Screenshot 1: Load Data - Buckets

- Header:** InfluxDB Times Series Data Platf... | Buckets | Load Data | s | InfluxDB | +
- URL:** us-east-1.aws.cloud2.influxdata.com/orgs/825b235d3873a01e/load-data/buckets
- Toolbar:** polyja, Filter buckets..., Sort by Name (A → Z), CREATE BUCKET
- Content:**
 - Buckets List:** A list of buckets, with one named "jj" highlighted by a red box. Details: Retention: 7 days, Schema Type: Implicit, ID: 80ec55fd26859469. Buttons: ADD DATA, SETTINGS.
 - What is a Bucket?** A purple sidebar explaining what a bucket is: "A bucket is a named location where time series data is stored. All buckets have a Retention Period, a duration of time that each data point persists." It also links to "how to write data" into your bucket.
 - Need more buckets?** A section with a "GET MORE BUCKETS" button.

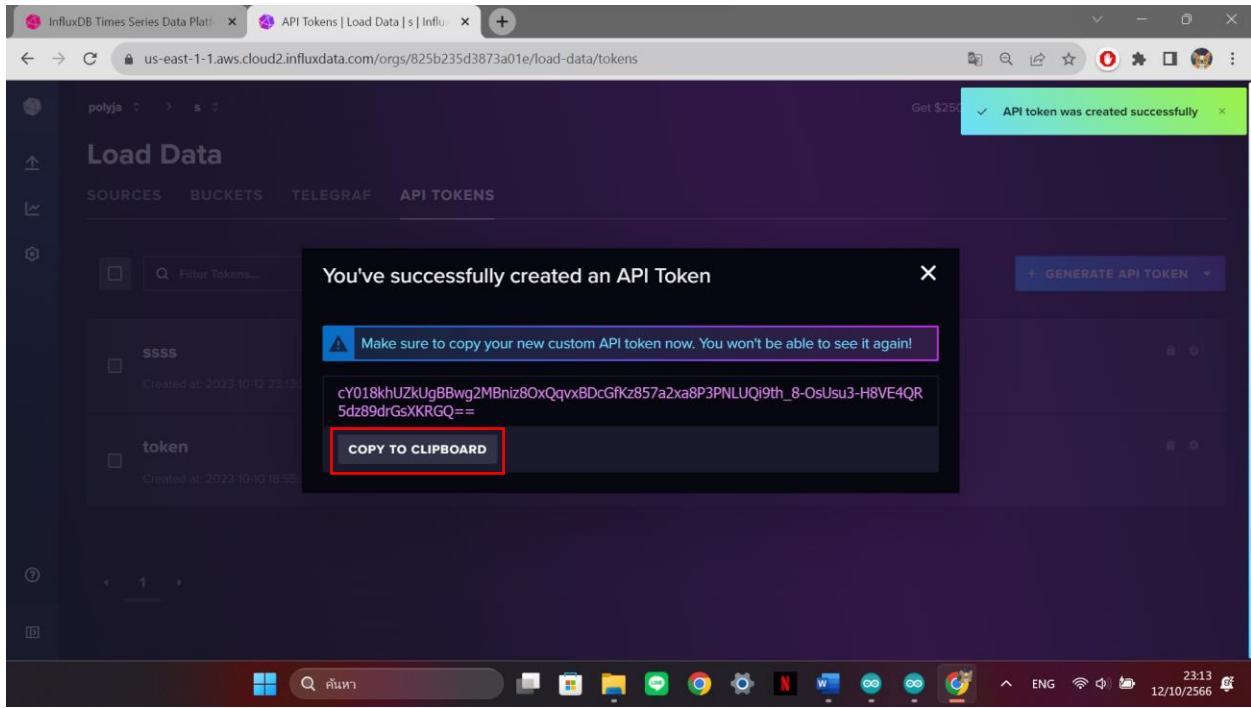
Screenshot 2: Load Data - API Tokens

- Header:** InfluxDB Times Series Data Platf... | API Tokens | Load Data | s | InfluxDB | +
- URL:** us-east-1.aws.cloud2.influxdata.com/orgs/825b235d3873a01e/load-data/tokens
- Toolbar:** polyja, Sources, Buckets, Telegraf, API Tokens (highlighted with a red arrow), Filter Tokens..., Sort by Description (A → Z), GENERATE API TOKEN
- Content:**
 - Data List:** A list of tokens, with one named "token" highlighted. Details: Created at: 2023-10-10 18:55:31, Owner: ploy54218740@gmail.com, Last Modified: 2 days ago.

The screenshot shows two consecutive screenshots of the InfluxDB Times Series Data Platform interface, specifically the 'API Tokens' section under the 'Load Data' tab.

In the first screenshot, the 'API TOKENS' tab is selected. A red arrow points from the 'GENERATE API TOKEN' button in the top right corner to a dropdown menu. The dropdown menu contains two options: 'All Access API Token' and 'Custom API Token'. The 'All Access API Token' option is highlighted.

In the second screenshot, a modal dialog titled 'Generate All Access API Token' is open. It contains a warning message: 'This token will be able to create, update, delete, read, and write to anything in this organization'. Below the message is a 'Description' input field containing the text 'พิมพ์โค้ดเรกีดี จะเอา token เดยๆ'. At the bottom of the dialog are 'CANCEL' and 'SAVE' buttons.



เปิดโปรแกรม Arduino → new file

```
#include "DHTesp.h"
#include <Wire.h>
#include <BH1750.h>
#include <ESP32Encoder.h>
#include <WiFiMulti.h>
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

WiFiMulti wifiMulti;
DHTesp dht;
BH1750 lightMeter;
ESP32Encoder encoder;
int counter = 0;
int cycle = 30;
#define DT 17
#define CLK 16
#define Pin_DHT22 15
#define DEVICE "Polyja" //ชื่ออุปกรณ์ของเรา
#define WIFI_SSID "Aimmy3flr50" //ไฟที่ใช้เชื่อมบอร์ด
#define WIFI_PASSWORD "Aimmy@2549" //ไฟที่ใช้เชื่อมบอร์ด
#define INFLUXDB_URL "https://us-east-1-1.aws.cloud2.influxdata.com"
```

```

#define INFLUXDB_TOKEN "cY018khUZkUgBBwg2MBniz8OxQqvxBDcGfKz857a2xa8P3PNLUQi9th_8-OsUsu3-
H8VE4QR5dz89drGsXKRGQ==" //tokenที่คัดลอกมาเมื่อวาน
#define INFLUXDB_ORG "ploy54218740@gmail.com" //อีเมล์ที่ใช้สมัคร
#define INFLUXDB_BUCKET "jj" //ชื่อbucketของเรา
#define TZ_INFO "UTC7"

InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACert);

Point sensor("ESP32");

void setup() {
    Serial.begin(115200);
    dht.setup(Pin_DHT22, DHTesp::DHT22);
    Wire.begin();
    lightMeter.begin();
    encoder.attachHalfQuad(DT, CLK);
    encoder.setCount(0);
    WiFi.mode(WIFI_STA);
    wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to wifi");
    while (wifiMulti.run() != WL_CONNECTED) {
        Serial.print(".");
        delay(100);
    }
    Serial.println();
    timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");
    if (client.validateConnection()) {
        Serial.print("Connected to InfluxDB: ");
        Serial.println(client.getServerUrl());
    } else {
        Serial.print("InfluxDB connection failed: ");
        Serial.println(client.getLastErrorMessage());
    }
    sensor.addTag("device", DEVICE);
    sensor.addTag("SSID", WiFi.SSID());
}

void loop() {
    sensor.clearFields();
}

```

```
float h = dht.getHumidity();
float t = dht.getTemperature();
float Luxfloat = lightMeter.readLightLevel();
Serial.print("Temperature: ");
Serial.print(t); Serial.print(" *C \t");
Serial.print("Humidity: ");
Serial.print(h); Serial.print(" % \t");
Serial.print("Light: ");
Serial.print(Luxfloat); Serial.print(" lx \t");

long newPos = encoder.getCount();
if (newPos < 0){
    counter = cycle - abs(newPos);
} else{
    counter = abs(newPos);
}
int deg = counter * (360.00/cycle);
Serial.print("Degree: ");
Serial.println(deg);
if(newPos >= cycle){
    encoder.setCount(0);
}
if(newPos <= -cycle){
    encoder.setCount(0);
}
sensor.addField("Humidity", h);
sensor.addField("Temperature", t);
sensor.addField("LightLevel", Luxfloat);
sensor.addField("Degree", deg);

Serial.print("Writing: ");
Serial.println(sensor.toLineProtocol());
if (wifiMulti.run() != WL_CONNECTED) {
    Serial.println("Wifi connection lost");
}
if (!client.writePoint(sensor)) {
```

```

Serial.print("InfluxDB write failed: ");
Serial.println(client.getLastErrorMessage());
}
}

```

กดอัพโหลด code ลงบอร์ด

ไฟสีเขียว งาน เตรียมเมื่อ ช่วงเวลาที่

```

ESP32_to_Influx_DB_with_Grafana_Dashboard_Display
Point sensor("ESP32");
void setup() {
    Serial.begin(115200);
    dht.setup(Pin_DHT22, DHTesp::DHT22);
    Wire.begin();
    lightMeter.begin();
    encoder.attachHalfQuad(DT, CLK);
    encoder.setCount(0);
    WiFi.mode(WIFI_STA);
    wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to wifi");
    while (wifiMulti.run() != WL_CONNECTED) {
        Serial.print(".");
        delay(100);
    }
    Serial.println();
    timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");
    if (client.validateConnection()) {
        // ดำเนินการเมื่อเชื่อมต่อสำเร็จ
    }
}

DHTEsp dht;
BH1750 lightMeter;
ESP32Encoder encoder;
int counter = 0;
int cycle = 30;
#define DT 17
#define CLK 16
#define Pin_DHT22 15
#define DEVICE "Polyja"
#define WIFI_SSID "AndroidAP33"
#define WIFI_PASSWORD "ydvv5842"
#define INFLUXDB_URL "https://us-east-1-1.amazonaws.com/influxdb/write?org=polyja&token=...&precision=s"
#define INFLUXDB_TOKEN "cY018khUZkUgBBwg2M"
#define INFLUXDB_ORG "ploy54218740@gmail.com"
#define INFLUXDB_BUCKET "jj"
#define TZ_INFO "UTC7"

InfluxDBClient client(INFLUXDB_URL, INFLUXDB_TOKEN);

void loop() {
    // อ่านค่าจากเซ็นเซอร์
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();
    int lux = lightMeter.readLight();
    encoder.read();

    // คำนวณค่า
    float averageLight = lux / cycle;
    float averageTemperature = temperature / cycle;
    float averageHumidity = humidity / cycle;

    // บันทึกข้อมูลไปยัง InfluxDB
    String data = "Temperature=" + String(averageTemperature) + " Humidity=" + String(averageHumidity) + " Light=" + String(averageLight) + " Degr";
    client.write("ESP32", "device=" + DEVICE, "SSID=" + WIFI_SSID, "Humidity=" + String(averageHumidity), "Temperature=" + String(averageTemperature));
    Serial.println(data);
}

```

Serial monitor output:

```

load:0x40080400,len:3028
entry 0x400805e4
Connecting to wifi
Syncing time....
Synchronized time: Thu Oct 12 09:34:00 2023
Connected to InfluxDB: https://us-east-1-1.amazonaws.com/influxdb/write?org=polyja&token=...&precision=s
Temperature: 29.60 *C  Humidity: 80.40 %  Light: 20.00 lx  Degr
Writing: ESP32,device=Polyja,SSID=AndroidAP33 Humidity=80.40,Temperature=29.
Temperature: 29.60 *C  Humidity: 79.80 %  Light: 19.17 lx  Degr
Writing: ESP32,device=Polyja,SSID=AndroidAP33 Humidity=79.80,Temperature=29.
Temperature: 29.60 *C  Humidity: 80.00 %  Light: 16.67 lx  Degr
Writing: ESP32,device=Polyja,SSID=AndroidAP33 Humidity=80.00,Temperature=29.

```

Arduino IDE status bar:

20 DOIT ESP32 DEVKIT V1, 80MHz, 921600, None, Disabled via COM8
23:34 12/10/2566

จากนั้น

The screenshot shows the 'Load Data' section of the InfluxDB platform. A red box highlights the 'jj' bucket entry in the list. The bucket details shown are: Retention: 7 days, Schema Type: Implicit, ID: 80ec55fd26859469. There are '+ ADD DATA' and 'SETTINGS' buttons. To the right, a sidebar titled 'What is a Bucket?' provides information about buckets and how to write data.

The screenshot shows the 'Data Explorer' section of the InfluxDB platform. A red box highlights the 'jj' measurement in the Schema Browser. The measurement details shown are: Measurement: Select measurement..., Value: ESP32. There is a 'SQL Sync' toggle button. The main area shows a query editor with a 'RUN' button and a results table with 0 rows. The results table has 'TABLE' and 'GRAPH' tabs.

The screenshot shows the InfluxDB Data Explorer interface. On the left, there's a schema browser with a dropdown for 'Bucket' set to 'jj' and 'Measurement' set to 'ESP32'. A search bar below it says 'Search fields and tag keys'. A red box labeled '1' highlights a list of selected fields: 'Degree', 'Humidity', 'LightLevel', and 'Temperature', all with checked checkboxes. To the right, a code editor window displays a SQL query:

```

1 SELECT *
2 FROM "ESP32"
3 WHERE

```

The status bar at the bottom of the code editor says 'Ready (282ms)'. Below the code editor is a results table with 66 rows. The first few rows of data are:

	Degree	device	Humidity	LightLevel	SSID	Temperature	time
0	Polyja	74	37.5	no group	AndroidAP33	31.6	2023
0	Polyja	80.4	37.5	no group	AndroidAP33	29.7	2023
0	Polyja	80	37.5	no group	AndroidAP33	29.6	2023
0	Polyja	80	37.5	no group	AndroidAP33	29.6	2023

At the bottom of the results table, there are navigation buttons for pages 1 through 14. A red box labeled '2' highlights the 'Past 1m' button in the top right corner of the results area.

เข้าเว็บ Grafana

The screenshot shows the Grafana Cloud homepage. At the top, there's a message: 'You have uncapped usage until 24 ตุลาคม 2566. Upgrade plans to continue using Grafana Cloud with unlimited, pay-as-you-go usage.' with a 'Upgrade now' button. Below this, a large 'Welcome to Grafana Cloud' header is followed by a 'Unlimited Usage Trial' button and a '12 days left' timer. A red box highlights the 'Connect data' button in the top right corner of the main content area. The main content area displays 'Billable usage' statistics: 1 Active user, 0 Metrics, 0 bytes for Logs, Traces, Profiles, and tests, 0 VUh, 0 IRM active users, and links to 'Upgrade plan →' and 'Keep unlimited usage'.

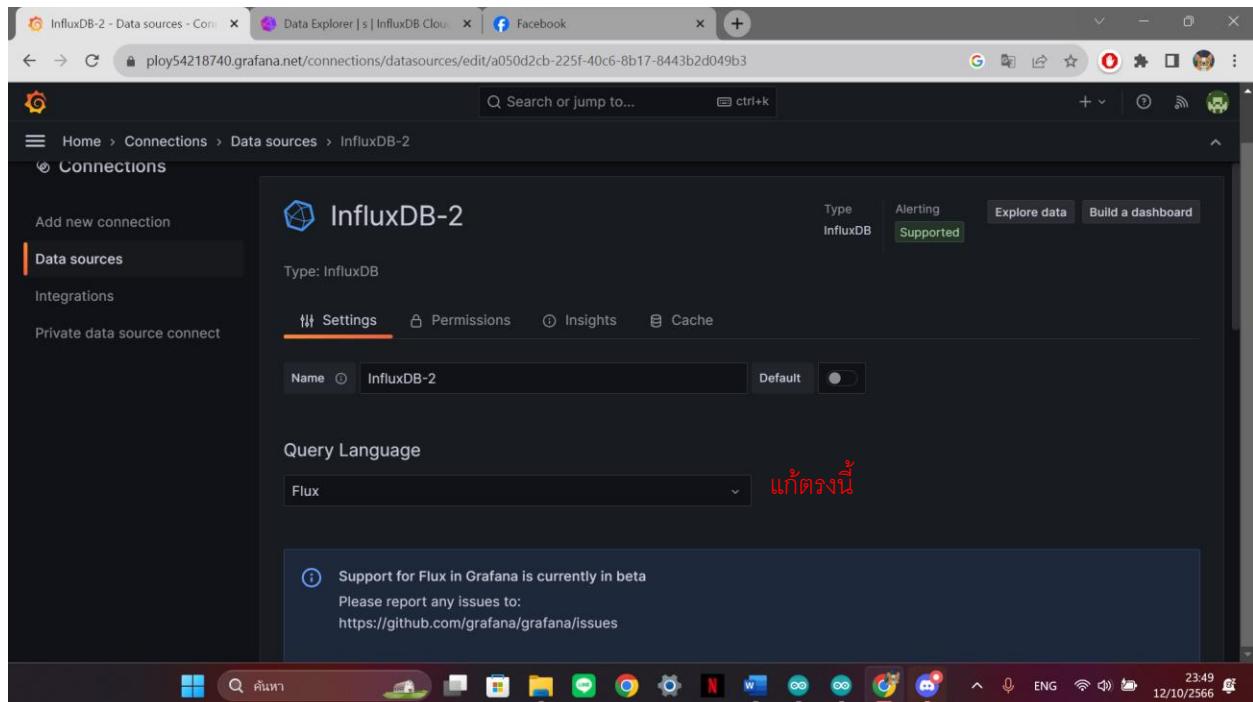
The image shows two screenshots of the Grafana web interface, illustrating the steps to add a new InfluxDB data source.

Step 1: The first screenshot shows the "Connections" page. A red box highlights the "Add new connection" button under the "Connections" sidebar. Below it, a red box highlights the "InfluxDB" data source card.

Step 2: The second screenshot shows the "InfluxDB - Add new connection" configuration page. A red box highlights the "Add new data source" button at the top right of the main content area.

Configuration Details (Visible in Step 2):

- InfluxDB Data Source:** Open source time series database
- Version:** 5.0.0
- From:** Grafana Labs
- Signature:** Core
- Buttons:** Overview, Version history, Add new data source



The screenshot shows the 'InfluxDB-2' connection configuration in Grafana. The 'Settings' tab is selected. The 'Name' field is set to 'InfluxDB-2'. The 'Query Language' dropdown is set to 'Flux'. A note indicates that support for Flux is currently in beta. The 'HTTP' section shows the URL as 'https://us-east-1.aws.cloud2.influxdata.c...'. The 'Auth' section includes options for 'Basic auth', 'TLS Client Auth', 'Skip TLS Verify', and 'Forward OAuth Identity'. The 'Custom HTTP Headers' section has a '+ Add header' button.

InfluxDB-2

Type: InfluxDB Alerting Supported Explore data Build a dashboard

Name: InfluxDB-2 Default

Query Language: Flux แก้ตัวง่าย

Support for Flux in Grafana is currently in beta
Please report any issues to:
<https://github.com/grafana/grafana/issues>

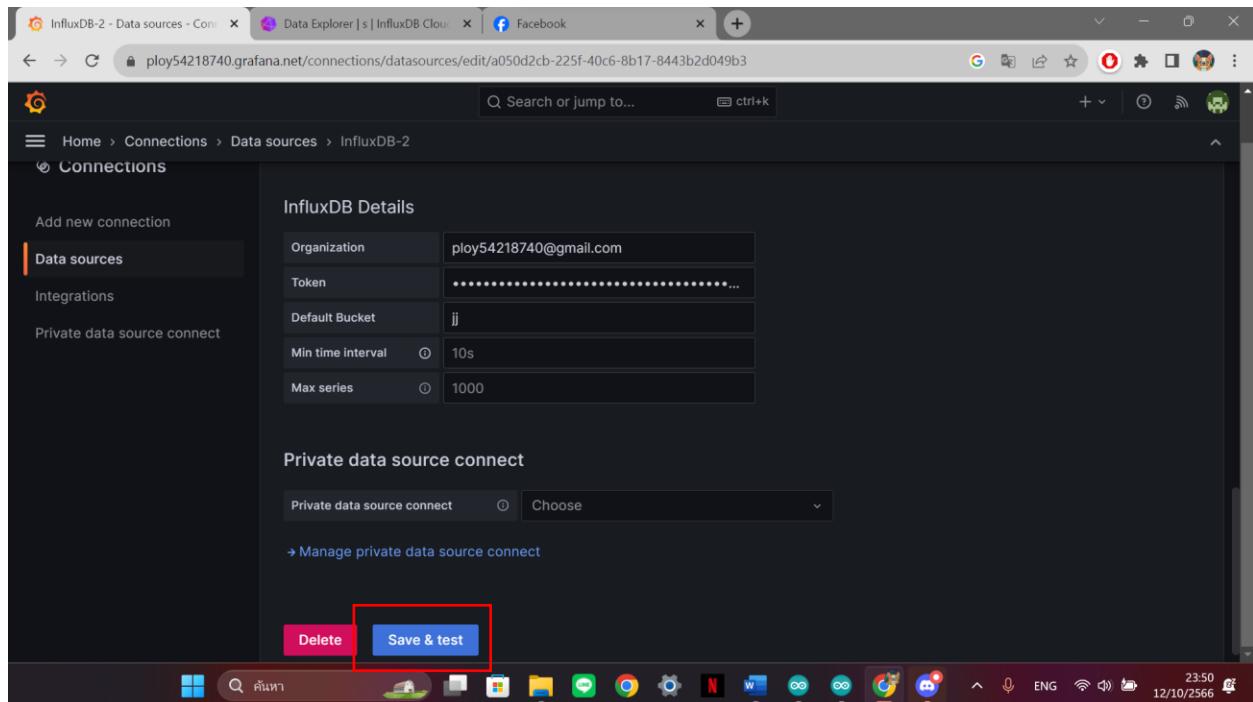
HTTP

URL: https://us-east-1.aws.cloud2.influxdata.c...
Allowed cookies: New tag (enter key to add) Add
Timeout: Timeout in seconds

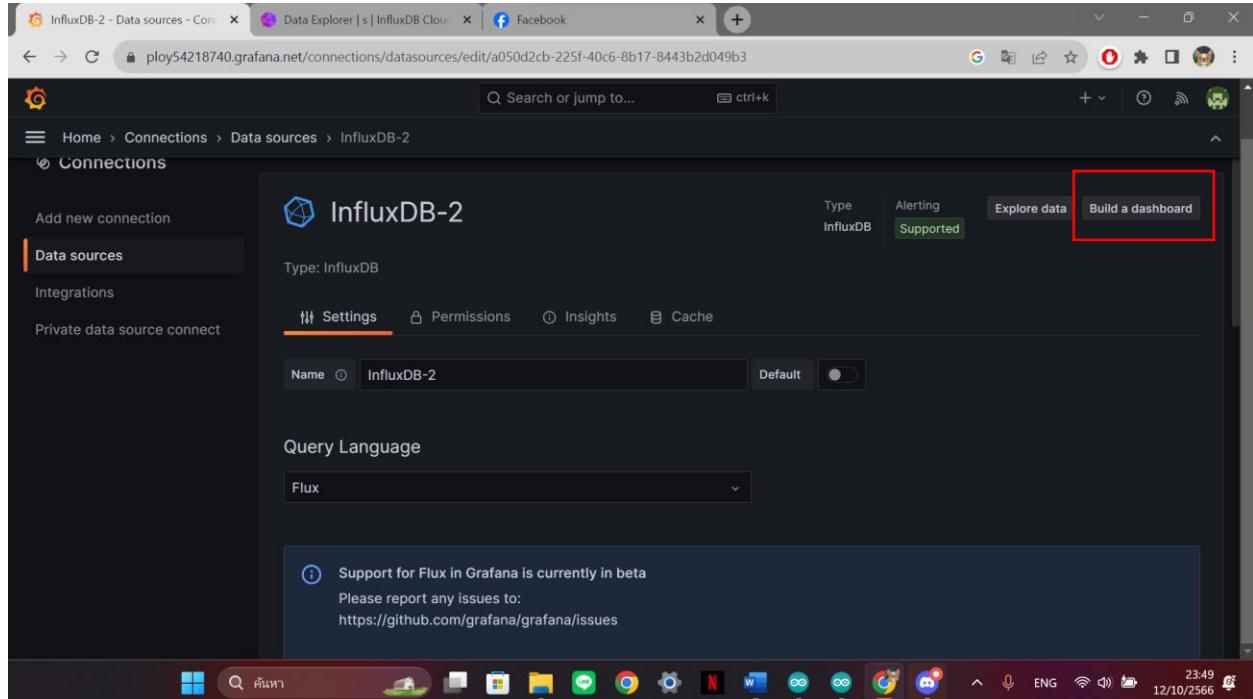
Auth

Basic auth With Credentials
TLS Client Auth With CA Cert
Skip TLS Verify
Forward OAuth Identity

+ Add header



The screenshot shows the 'Data sources' configuration page in Grafana. On the left sidebar, 'Data sources' is selected. The main area displays 'InfluxDB Details' with fields: Organization (ploy54218740@gmail.com), Token (redacted), Default Bucket (jj), Min time interval (10s), and Max series (1000). Below this is a 'Private data source connect' section with a 'Choose' dropdown and a link to 'Manage private data source connect'. At the bottom are 'Delete' and 'Save & test' buttons, with 'Save & test' highlighted by a red box.



The screenshot shows the 'InfluxDB-2' configuration page. The top bar shows 'Type: InfluxDB' and 'Alerting Supported'. Below are tabs for 'Settings' (selected), 'Permissions', 'Insights', and 'Cache'. The 'Name' field is set to 'InfluxDB-2'. Under 'Query Language', 'Flux' is selected. A note at the bottom states: 'Support for Flux in Grafana is currently in beta. Please report any issues to: https://github.com/grafana/grafana/issues'. At the top right, there are buttons for 'Explore data' and 'Build a dashboard', with 'Build a dashboard' highlighted by a red box.

The screenshot shows two consecutive steps in the Grafana interface for creating a new dashboard.

Step 1: New dashboard creation

The top window shows the 'Dashboards' section with a message: 'Start your new dashboard by adding a visualization'. A blue button labeled '+ Add visualization' is highlighted with a red box. Below it are sections for 'Add a library panel' and 'Import a dashboard'.

Step 2: Selecting a data source

The bottom window shows the 'Edit panel - New dashboard' screen. A modal dialog titled 'Select data source' is open, listing various data sources. The 'InfluxDB-2' entry is selected and highlighted with a red box. Other entries include 'grafanacloud-k6', 'grafanacloud-ploy54218740-alert-state-history', 'grafanacloud-ploy54218740-cardinality-management', 'grafanacloud-ploy54218740-graphite', 'grafanacloud-ploy54218740-logs', and 'grafanacloud-ploy54218740-profiles'. The 'Apply' button is visible in the top right corner of the modal.

The image shows two screenshots of the Grafana dashboard editor interface. Both screenshots have a dark theme.

Top Screenshot:

- Panel Title:** Panel Title
- Content:** No data
- Query Bar:** Query 1, Transform data 0, Alert 0
- Data Source:** InfluxDB-2
- Time Range:** Last 6 hours
- Inspector:** Query inspector
- Panel Options:** Title (Panel Title), Description, Transparent background

Bottom Screenshot:

- Panel Title:** Panel Title
- Content:** A code editor window containing Flux query language.
- Code:**

```

1 from(bucket: "jj")
2 |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3 |> filter(fn: (r) => r["_measurement"] == "Polyja")
4 |> filter(fn: (r) => r["_field"] == "Temperature" or r["_field"] == "Humidity" or r["_field"]
5 |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
6 |> yield(name: "mean")
    
```
- Query Bar:** Query 1, Transform data 0, Alert 0
- Data Source:** InfluxDB-2
- Time Range:** Last 6 hours
- Inspector:** Query inspector
- Panel Options:** Title (Panel Title), Description, Transparent background
- Tooltip:**



