

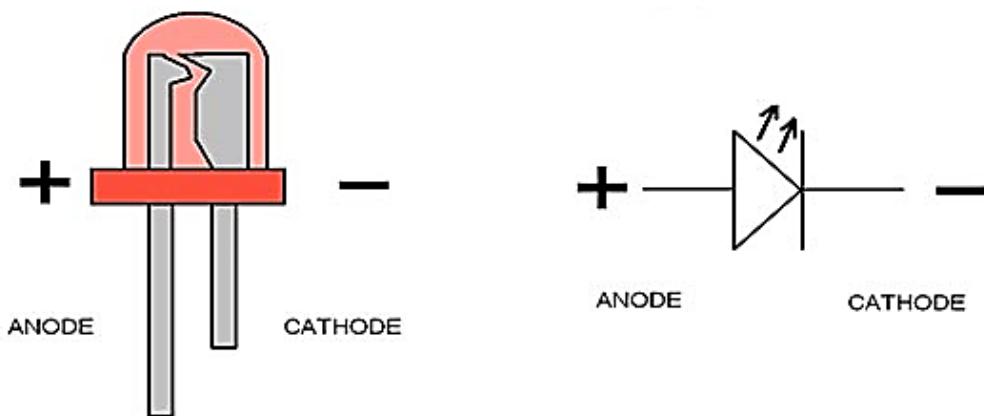
Workshop 1 สัญญาณไฟกระพริบทดสอบไมโครคอนโทรลเลอร์ ESP8266

Workshop นี้จะเป็นการทดลองการใช้ NodeMCU ESP8266 ในการควบคุมการกระพริบของหลอดไฟ LED ซึ่ง workshop จะยังไม่มีการเชื่อมต่ออินเทอร์เน็ต ทำให้ผู้ที่ไม่เคยเขียนโปรแกรมคอมพิวเตอร์มาก่อนสามารถทำความเข้าใจได้ไม่ยาก อีกทั้งยังเป็นการเรียนรู้การต่อวงจรอิเล็กทรอนิกส์เบื้องต้นไปพร้อม ๆ กันอีกด้วย

LED คืออะไร

LED ย่อมาจาก Light Emitting Diode คือ “ไดโอดชนิดเปล่งแสง”

ไดโอด (Diode) คือ อุปกรณ์กึ่งตัวนำ (Semi Conductor Device) ที่ให้กระแสไฟฟ้าไหลผ่านได้ทางเดียว ไดโอดเป็นอุปกรณ์พื้นฐานที่สำคัญในวงจรไฟฟ้า มีเชื่อมต่อไปในวงจรอิเล็กทรอนิกส์และวงจรไฟฟ้าเพื่อทำหน้าที่บังคับทิศทางการไหลของกระแสไฟฟ้า ไดโอดโดยทั่วไปแล้วไม่เปล่งแสงออกมามีสัญลักษณ์ทางวงจรคือ ➔ ส่วนไดโอดที่เปล่งแสงหรือ LED มีสัญลักษณ์ทางวงจรคือ ➔ ต่างกันนิดหน่อยตรงที่ไม่มีลูกศรแสดงการเปล่งแสงกับไม่มี



ประเภทของ LED

1. LED แบบดั้งเดิม

กำลังวัตต์ต่ำน้อย ขนาดหรือรูปร่างหรือสีขึ้นอยู่กับพลาสติกที่ใช้ทำเปลือกหุ้ม ใช้ทำไฟสัญญาณในวงจร



LED แบบดั้งเดิม

2. LED ขนาดเล็กมาก

ใช้เทคโนโลยีการเชื่อมเม็ด LED ติดลงไปกับแผงวงจรหรือที่เรียกว่า Surface Mounting Technology (SMT) ซึ่งเป็นวิธีการเดียวกับการประกอบชิ้นส่วนอิเล็กทรอนิกส์และ semi-conductor ขนาดเล็ก ในบางครั้งก็เรียก LED ชนิดนี้ว่า Surface Mounting Device LED หรือ SMD LED



SMD5050 LED Chip



SMD3528 LED Chip

LED ขนาดเล็กมากหรือ SMD LED

3. LED กำลังสูง

LED กำลังสูง หรือ Hi-power LED เป็น LED ชนิดที่ให้กำลังสูง ให้ความสว่างมาก ต้องการกระแสขับสูงถึง 100mA บางรุ่นอาจต้องการกระแสขับถึง 1A ดังนั้นการระบายน้ำร้อนสำหรับ LED ชนิดนี้จึงเป็นสิ่งสำคัญ ถ้าระบายน้ำร้อนไม่ได้อาจจะทำให้อุปกรณ์ชำรุดหรือเสียหายในภายใต้เวลาที่ LED ชนิดนี้ส่วนใหญ่จะใช้ทำอุปกรณ์ให้แสงสว่างหรือจะเข้ามาทดแทนหลอดไฟที่ใช้อยู่ในปัจจุบัน



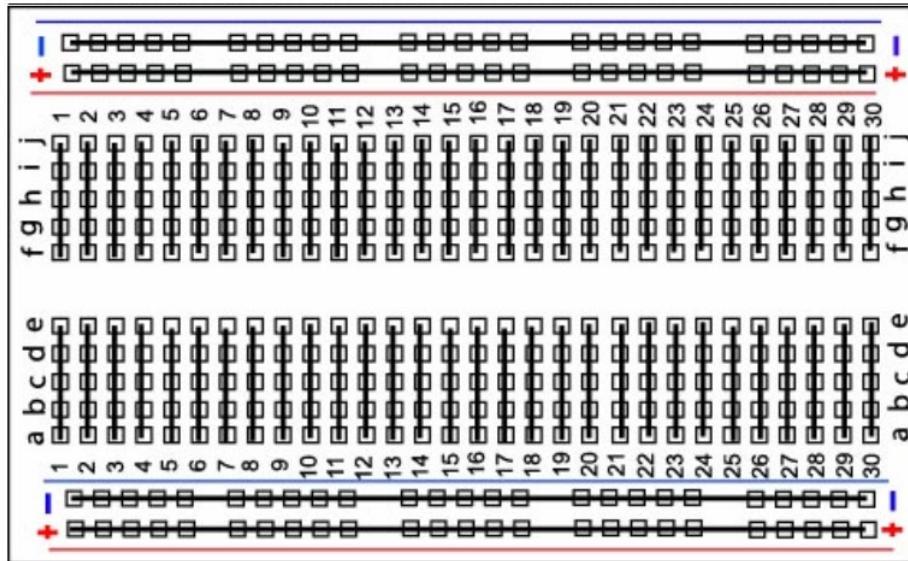
CREE XMA Series COB



Bridgelux Vero Series COB

LED กำลังสูง หรือ Hi-power LED

การต่อวงจรภายในแผ่นบอร์ดทดลอง (Breadboard)



การอ่านสีตัวต้านทาน

COLOR	1 ST BAND	2 ND BAND	3 RD BAND	MULTIPLIER	TOLERANCE	
					4-Band-Code	
Black		0	0	1Ω		
Brown	1	1	1	10Ω	± 1%	(F)
Red	2	2	2	100Ω	± 2%	(G)
Orange	3	3	3	1KΩ		
Yellow	4	4	4	10KΩ		
Green	5	5	5	100KΩ	± 0.5%	(D)
Blue	6	6	6	1MΩ	± 0.25%	(C)
Violet	7	7	7	10MΩ	± 0.10%	(B)
Grey	8	8	8	100MΩ	± 0.05%	
White	9	9	9	1GΩ		
Gold				0.1Ω	± 5%	(J)
Silver				0.01Ω	± 10%	(K)

Below the table, there is a diagram of a 5-band resistor color code. It shows a resistor with five bands: red, orange, black, black, and gold. The first four bands correspond to the values in the table, and the fifth band indicates a tolerance of ± 1%. The text "5-Band-Code" is written above the resistor.

ตารางสีในการอ่านค่าตัวต้านทาน ที่มา <https://inwfile.com/s-do/10jy0g.png>

จากรูปเป็นตารางสีในการอ่านค่าตัวต้านทาน ตัว R อ่านได้ดังต่อไปนี้

สำหรับแบบ 4 Band (หรือ 4 สี)

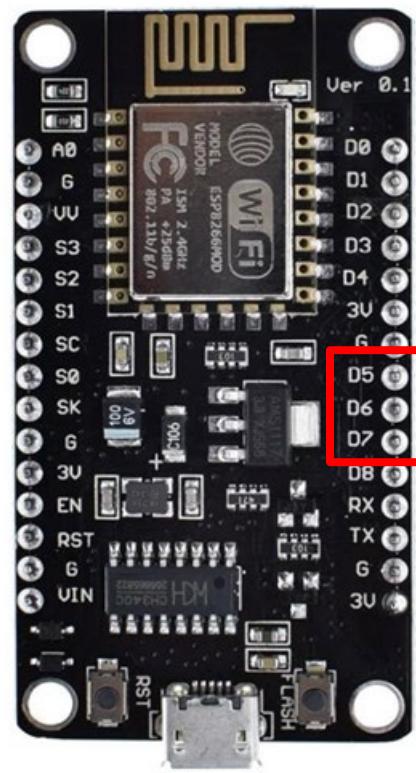
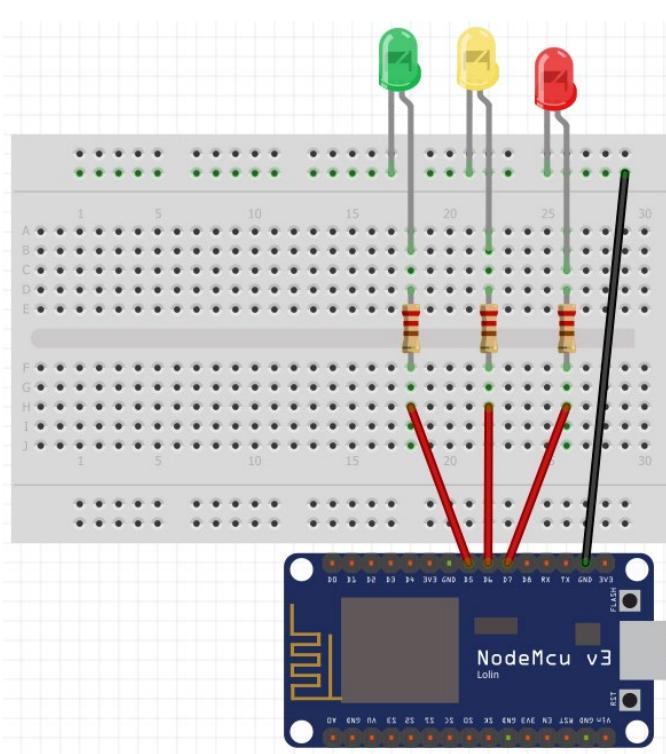
- ขั้นที่ 1 หันตัว ตัวต้านทาน (R) โดยให้ແບສີຄວາມຄລາດເຄລືອນ ສີເງິນແລະສົມອງໄປທາງຂວາ
- ແກບສີທີ່ 1 ເປັນຄ່າຕຳແໜ່ງທີ່ 1
- ແກບສີທີ່ 2 ເປັນຄ່າຕຳແໜ່ງທີ່ 2
- ແກບສີທີ່ 3 ເປັນຕົວຄຸນ
- ແກບສີທີ່ 4 ເປັນຄ່າຄວາມພິດພລາດ ບວກລບ

ອຸປະກນົນທີ່ຕ້ອງໃຊ້ຈານ

1. NodeMCU ESP8266
2. หลอดໄຟ LED ສີແດງ ,ສີເໜືອງ, ສີເຂີຍວ
3. ຕັວຕ້ານທານ 220 ໂອໜ້ມ 3 ຕັ້ງ (ແກບສີ ແດງ ແດງ ດຳ ດຳ ນຳຕາລ)
4. ສາຍໄຟ

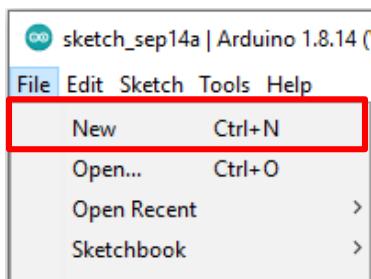
ກາຣຕ່ອວງຈຮ

PIN	ອຸປະກນົນ
D5	LED ສີເຂີຍວ
D6	LED ສີເໜືອງ
D7	LED ສີແດງ



การเขียน code

กดไปที่ File และกด New เพื่อสร้างหน้าต่างใหม่ในการเขียนโปรแกรม



จากนั้นเขียน code ตามดังนี้

Workshop_1

```
//กำหนด PIN ต่าง ๆ
int LED1 = D5;
int LED2 = D6;
int LED3 = D7;
void setup() {
    //กำหนด mode ให้แต่ละ PIN
    pinMode(D5, OUTPUT);
    pinMode(D6, OUTPUT);
    pinMode(D7, OUTPUT);
}

void loop() {
    digitalWrite(LED1, HIGH); //สั่งให้ส่งสัญญาณ High ไปที่ LED 1
    delay(500); //delay การทำงาน 0.5 วินาที
    digitalWrite(LED1, LOW); //สั่งให้ส่งสัญญาณ Low ไปที่ LED 1
    delay(500);
    digitalWrite(LED2, HIGH);
    delay(500);
    digitalWrite(LED2, LOW);
    delay(500);
    digitalWrite(LED3, HIGH);
    delay(500);
    digitalWrite(LED3, LOW);
    delay(500);
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_1/Workshop_1.ino

ผลลัพธ์ที่ได้

หลอดไฟ LED แต่ละดวงติดเป็นเวลา 0.5 วินาทีแล้วก็ดับ จากนั้นอีก 0.5 วินาที หลอดไฟ LED อีกดวงนึงติดขึ้น

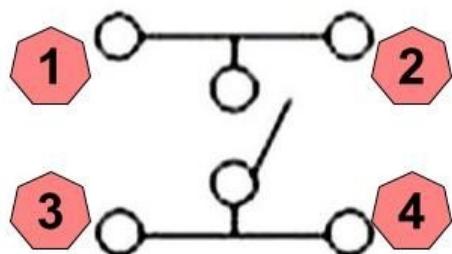
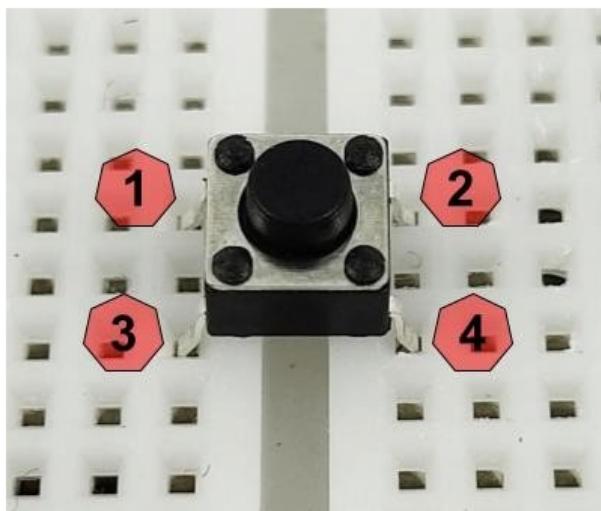
Workshop 2 การใช้งาน Push button ในการเปิด – ปิดหลอดไฟ LED

Push button

เป็นอุปกรณ์ไฟฟ้าชนิดหนึ่ง ซึ่งมีหน้าที่เป็นเหมือนสวิตซ์เปิด/ปิดให้กับวงจร โดย push button 4 pin มีการทำงานแบบ กดติดปล่อยดับหรือเรียกว่า Momentary Push Button



หลักการการทำงานของ push button 4 pin มีการทำงานโดยสังเกตตามรูปคือ เมื่อไม่มีการกด ขา 1 และ ขา 2 มีการเชื่อมกันอยู่ เช่นกันเดียวกันกับขา 3 และ ขา 4 หากสมมติให้ไฟฟ้ามีการไหลเข้าผ่านขา 1 เมื่อมีการกดปุ่มตัวไฟฟ้าก็สามารถไหลเข้าขา 3 และ 4 ได้



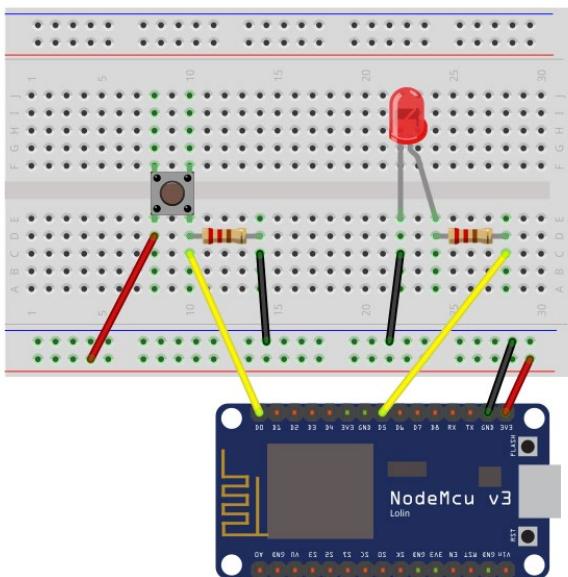
รูปจาก <https://th.cytron.io/p-6x6x1-push-button-4pin>

อุปกรณ์ที่ต้องใช้งาน

1. NodeMCU ESP8266
2. หลอดไฟ LED
3. ตัวต้านทาน 220 โอห์ม 2 ตัว (ແຄບສື ແດງ ແດງ ດຳ ດຳ ນໍາຕາລ)
4. Push button
5. สายไฟ

การต่อวงจร

PIN	อุปกรณ์
D0	Push button
D5	LED



การเขียน code

```
Workshop_2
//กำหนด PIN ต่าง ๆ
int LED1 = D5;
int button = D0;
int buttonState = 0;
void setup() {
    //กำหนด mode ให้แต่ละ PIN
    pinMode(D0, INPUT);
    pinMode(D5, OUTPUT);

    Serial.begin(9600);
}

void loop() {
    buttonState = digitalRead(button);
    Serial.println(buttonState); //แสดงสถานะของ buttonState
    if(buttonState==1) { //?ใช้ในการเช็คสถานะของ buttonState ว่ามีค่าเท่ากับ 1 หรือ High หรือไม่
        digitalWrite(LED1,HIGH);
    }
    else{
        digitalWrite(LED1, LOW);
    }
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_2/Workshop_2.ino

ผลลัพธ์ที่ได้

เมื่อทำการกดปุ่ม ก็จะทำให้หลอดไฟ LED นั้นติด และเมื่อปล่อยปุ่มกด ก็จะทำให้หลอดไฟดับ

Workshop 3 เปิดปิด หลอดไฟ LED และ pump น้ำโดยใช้ relay

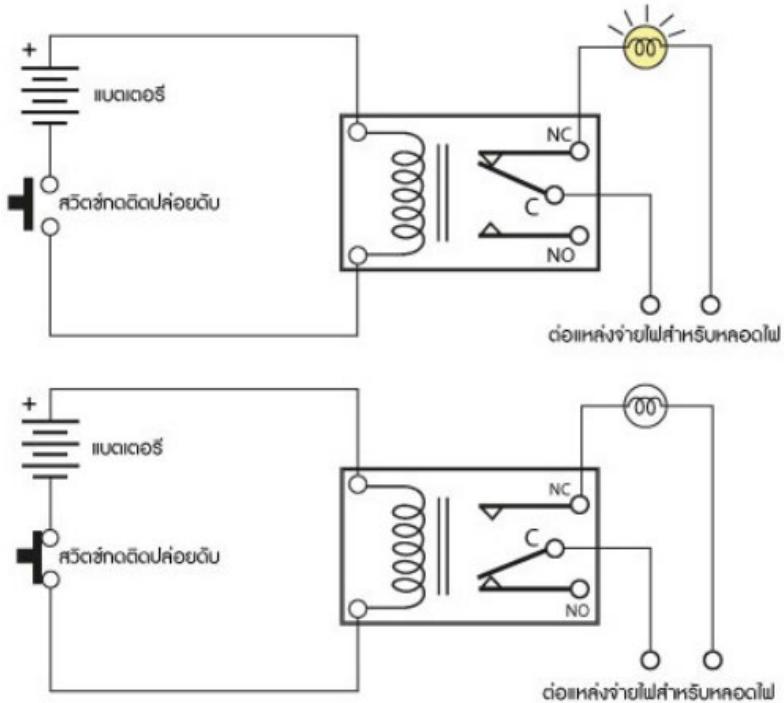
Workshop นี้จะทำการสั่งการทำงาน เปิด-ปิด หลอดไฟ LED และ pump น้ำ โดยใช้ relay เป็นตัวควบคุมการทำงาน

รีเลย์ (Relay) คืออะไร

เป็นอุปกรณ์ที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานแม่เหล็ก เพื่อใช้ในการดึงดูดหน้าสัมผัสของคอนแทคให้เปลี่ยนสภาพ โดยการป้อนกระแสไฟฟ้าให้กับขดลวด เพื่อทำการปิดหรือเปิดหน้าสัมผัสด้วยกับสวิตช์ อิเล็กทรอนิกส์ ซึ่งเราสามารถนำรีเลย์ไปประยุกต์ใช้ในการควบคุมวงจรต่าง ๆ ในงานช่างอิเล็กทรอนิกส์มากมาย

Relay ประกอบด้วยส่วนสำคัญ 2 ส่วนหลักก็คือ

- ส่วนของขดลวด (coil) เนี่ยย่วนนำกระแสแต่สำา ทำหน้าที่สร้างสนามแม่เหล็กไฟฟ้าให้แกนโลหะไปกระแทกให้หน้าสัมผัสต่อ กัน ทำงานโดยการรับแรงดันจากภายนอกต่อคร่อมที่ขดลวดเนี้ยวนำนี้ เมื่อขดลวดได้รับแรงดัน(ค่าแรงดันที่รีเลย์ต้องการขึ้นกับชนิดและรุ่นตามที่ผู้ผลิตกำหนด) จะเกิดสนามแม่เหล็กไฟฟ้าทำให้แกนโลหะด้านในไปกระแทกให้แผ่นหน้าสัมผัสดต่อ กัน
- ส่วนของหน้าสัมผัส (contact) ทำหน้าที่เมื่อไอนสวิตช์จ่ายกระแสไฟให้กับอุปกรณ์ที่เราต้องการนั่นเองจุดต่อใช้งานมาตรฐาน ประกอบด้วย
 - จุดต่อ NC ย่อมาจาก normal close หมายความว่าปกติปิด หรือ หากยังไม่จ่ายไฟให้ขดลวดเนี้ยวนำหน้าสัมผัสจะติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการให้ทำงานตลอดเวลา
 - จุดต่อ NO ย่อมาจาก normal open หมายความว่าปกติเปิด หรือหากยังไม่จ่ายไฟให้ขดลวดเนี้ยวนำหน้าสัมผัสจะไม่ติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการควบคุมการเปิด ปิด เช่น คอมไฟสนามหนีห้ามบ้าน
 - จุดต่อ C ย่อมาจาก common คือจุดร่วมที่ต่อมาจากแหล่งจ่ายไฟ



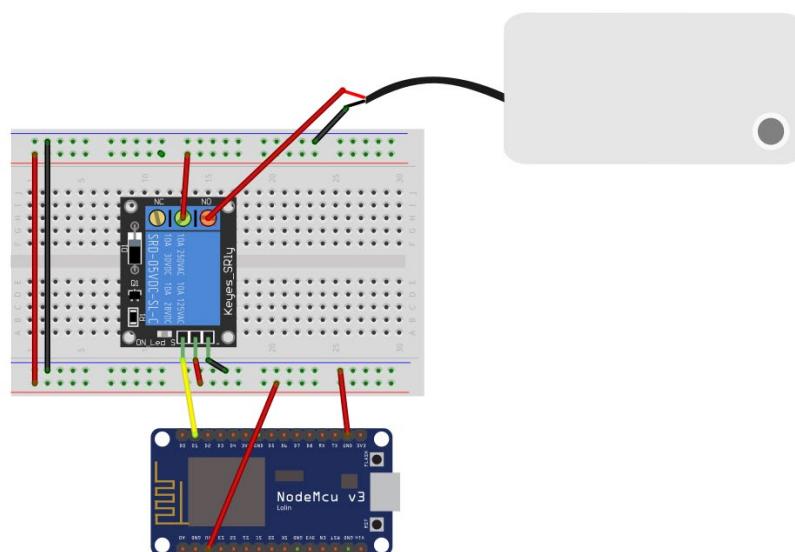
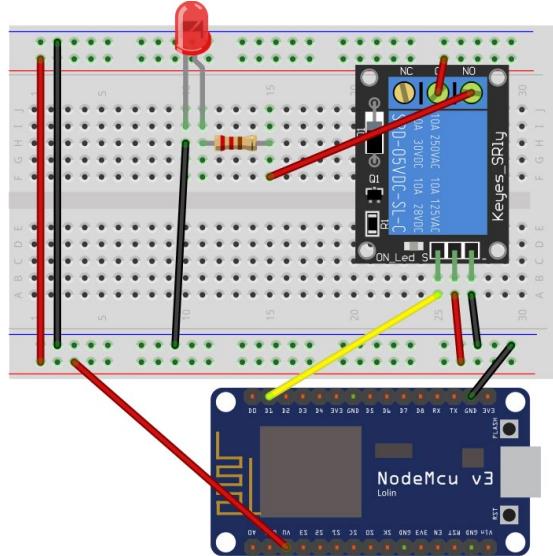
Relay Module สามารถนำมาทำ project ควบคุมเปิดปิดอุปกรณ์ไฟฟ้าภายในบ้านได้ รีเลย์ จะทำหน้าที่เหมือนสวิตซ์ทางไฟฟ้า ควบคุมการสับสะพานไฟด้วยสัญญาณ Digital 1 0 หรือ High กับ Low ปกติการเปิด ปิดไฟบ้านเราจะใช้ใน การกดสวิตซ์ แต่ถ้าเราใช้รีเลย์แทนสวิตซ์ เราสามารถนำสัญญาณดิจิตอลมาควบคุมการ เปิด-ปิดได้ สามารถนำมาประยุกต์ทำ Smart Home ควบคุม เปิด-ปิด อุปกรณ์ไฟฟ้าต่าง ๆ ภายในบ้านได้

อุปกรณ์ที่ต้องใช้งาน

1. NodeMCU ESP8266
2. รีเลย์ 5VDC แบบ 1 ช่อง จำนวน 1 ตัว
3. LED สีแดงและสีเขียว
4. มอเตอร์ปั๊มน้ำขนาดเล็ก
5. ตัวต้านทาน 220 โอห์ม 1 ตัว (แคบสี แดง แดง ดำ ดำ น้ำตาล)

การต่อวงจร

PIN	อุปกรณ์
D1	Relay



การเขียน code

Workshop_3

```
int relay1 = D1;  
void setup() {  
    pinMode(relay1, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(relay1, HIGH); // สั่งเปิดรีเลย์  
    delay(1000);  
    digitalWrite(relay1, LOW); // สั่งปิดรีเลย์  
    delay(1000);  
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_3/Workshop_3.ino

ผลลัพธ์ที่ได้

จากการเขียน code จะทำให้ relay ทำงานเป็นเวลา 3 วินาที และหยุดทำงานเป็นเวลา 3 วินาที

Workshop 4 วิจัยตรวจสอบความเข้มแสง จาก LDR Sensor และใช้เปิดไฟ LED

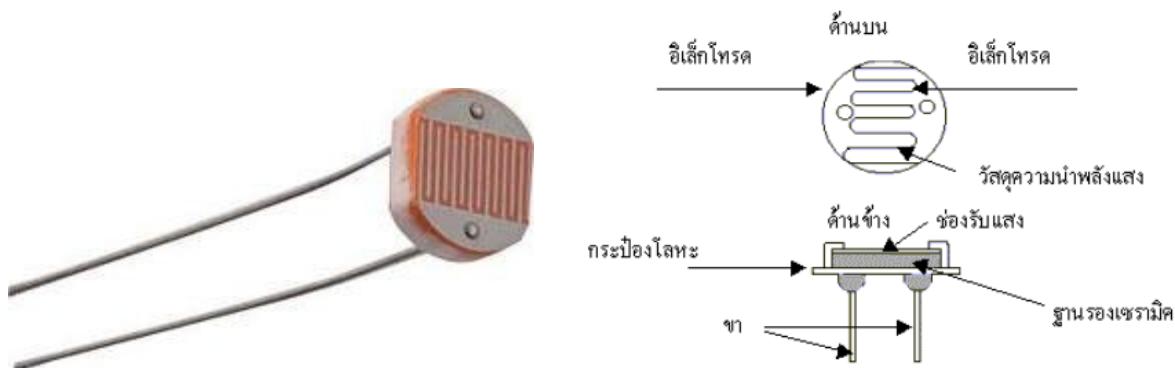
ในการทดลอง นี้จะเป็นการทำสวิตซ์แสง เมื่อมีแสงจะให้ไฟปิด เมื่อไม่มีแสงจะให้ไฟเปิด โดยการใช้ LDR (Light Dependent Resistor)

LDR คืออะไร

LDR คือ ตัวต้านทานชนิดหนึ่งหรือเรียกอีกอย่างว่าตัวต้านทานแปลค่าตามแสง

หลักการทำงานของ LDR คือ เมื่อถูกแสงตัว LDR จะมีความต้านทานลดลงและเมื่อไม่ถูกแสงตัว LDR จะมีความต้านทานมากขึ้น

LDR ย่อมาจาก Light Dependent Resistor แต่ในวงการอิเล็กทรอนิกส์จะเรียกอุปกรณ์ตัวนี้สั้นๆ ง่ายๆ ว่า LDR องค์ประกอบของ LDR จะประกอบด้วย สารกึ่งตัวนำ เช่น แคนเดเมียมแซลไฟฟ์และแคนเดเมียมมิลิโน๊ด ซึ่งเป็นสารที่มีการตอบสนองความยาวคลื่นแสง ซาบอยู่เป็นเส้นลักษณะเป็นขนาด คดเคี้ยวไปมาเป็นฐานเซรามิก LDR จะมีสองขั้ว ซึ่งมีค่าความต้านทานภายใต้ตัวเปลี่ยนแปลงค่าได้ตามแสงที่ตกลงมากรอบ

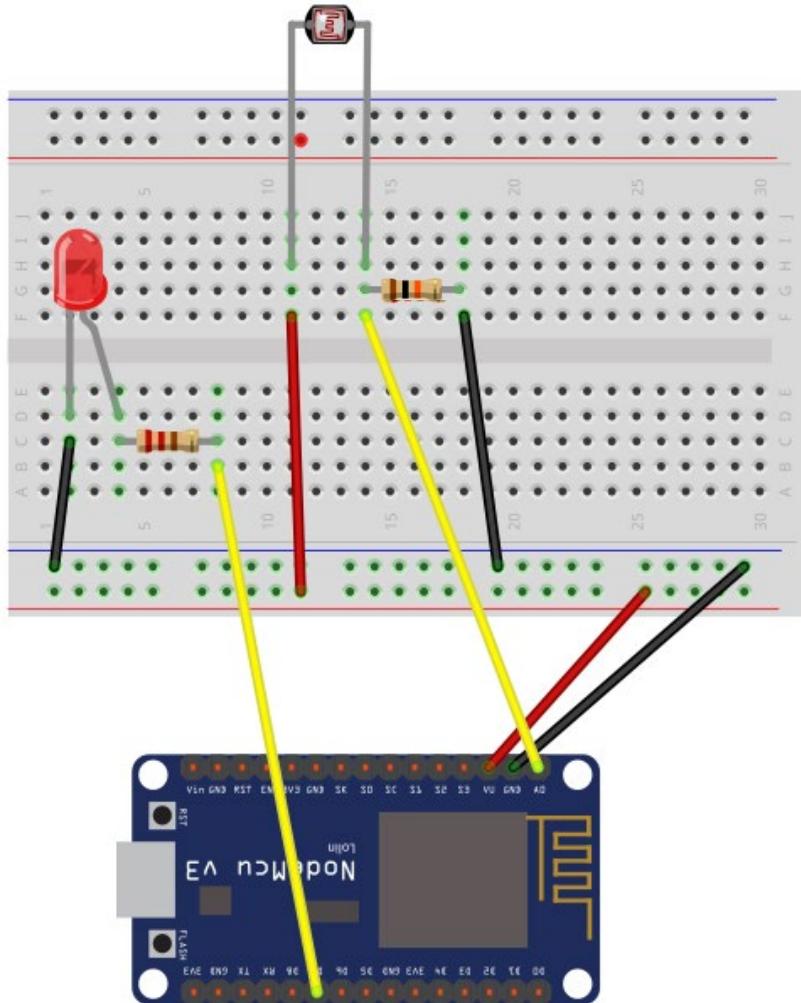


อุปกรณ์ที่ต้องใช้งาน

1. NodeMCU ESP8266
2. LDR 1 ตัว
3. LED สีแดง mm 1 ดวง
4. ความต้านทาน 220 โอห์ม 1 ตัว (ແບບສີ ແດງ ແດງ ດຳ ດຳ ນ້ຳຕາລ)
5. ความต้านทาน 10K โอห์ม 1 ตัว (ແບບສີ ນ້ຳຕາລ ດຳ ດຳ ແດງ ນ້ຳຕາລ)

การต่อวงจร

PIN	อุปกรณ์
A0	LDR
D7	LED



การเขียน code

- Code สำหรับอ่านค่า LDR

Workshop_LDR_read

```
#define analog_pin A0

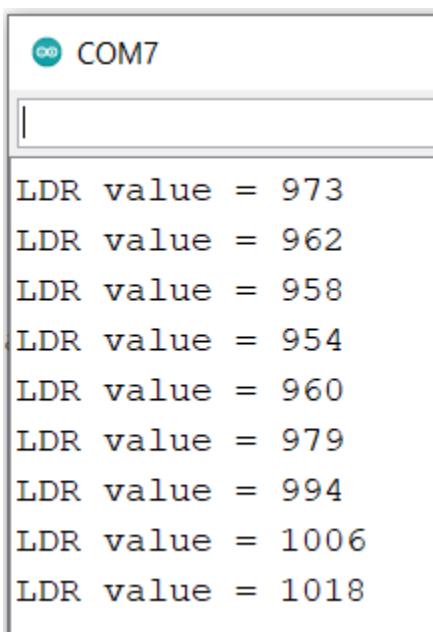
int ldr_value = 0;

void setup() {
    Serial.begin(9600);
}

void loop() {
    ldr_value = analogRead(analog_pin); // อ่านค่า analog จากตัวแปร analog_pin
    Serial.print("LDR value = ");
    Serial.println(ldr_value); // แสดงค่า ldr_value
    delay(1000);
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_4/Workshop_LDR_read/Workshop_LDR_read.ino

ผลลัพท์ที่ได้



```
LDR value = 973
LDR value = 962
LDR value = 958
LDR value = 954
LDR value = 960
LDR value = 979
LDR value = 994
LDR value = 1006
LDR value = 1018
```

- Code สำหรับอ่านค่า LDR และเปิด – ปิด LED

Workshop_LDR_LED

```
#define analog_pin A0
#define LED D7

int ldr_value = 0;
String led_state = "LOW";

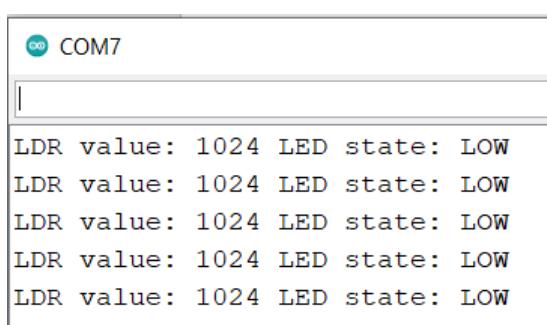
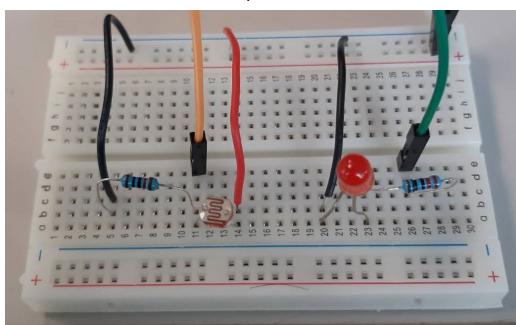
void setup() {
    //กำหนด mode ให้กับ pin
    pinMode(LED, OUTPUT);
    digitalWrite(LED, LOW); //ส่งสัญญาณ LOW ไปที่ LED
    Serial.begin(9600);
}

void loop() {
    ldr_value = analogRead(analog_pin);
    if(ldr_value > 700){ //เช็คค่าของ ldr_value ว่ามีค่ามากกว่า 700 หรือไม่
        digitalWrite(LED, LOW); //ส่งสัญญาณ LOW ไปที่ LED
        led_state = "LOW";
    }
    else{
        digitalWrite(LED, HIGH); //ส่งสัญญาณ HIGH ไปที่ LED
        led_state = "HIGH";
    }
    Serial.print("LDR value: ");
    Serial.print(ldr_value); // แสดงค่าของ ldr_value
    Serial.print("\t");
    Serial.print("LED state: ");
    Serial.println(led_state); // แสดงสถานะของ led_state
    delay(1000);
}
```

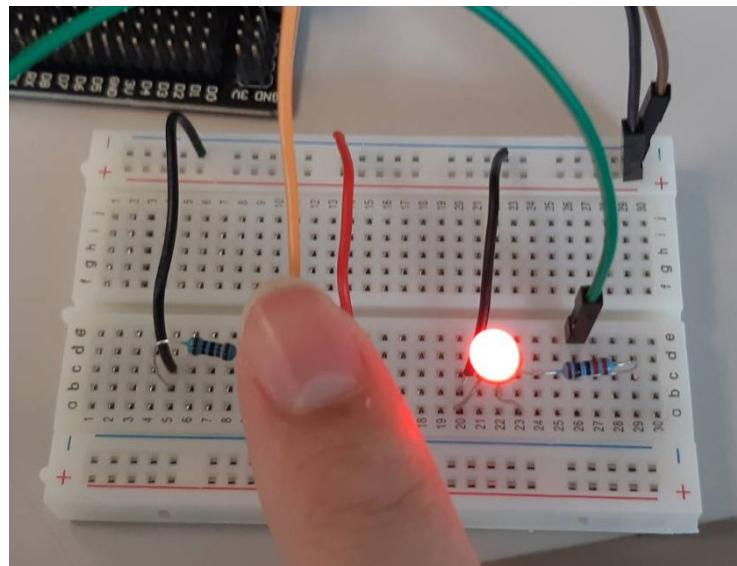
สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_4/Workshop_LDR_LED/Workshop_LDR_LED.ino

ผลลัพธ์ที่ได้

ผลลัพธ์จะมีเมื่อวัดคุณภาพ sensor



ผลลัพธ์ของที่มีวัตถุมาบัง sensor



```
xx COM7
|
| LDR value: 386  LED state: HIGH
| LDR value: 388  LED state: HIGH
| LDR value: 391  LED state: HIGH
| LDR value: 392  LED state: HIGH
| LDR value: 397  LED state: HIGH
```

Workshop 5 วัดระยะทางและตรวจจับวัตถุด้วย Ultrasonic

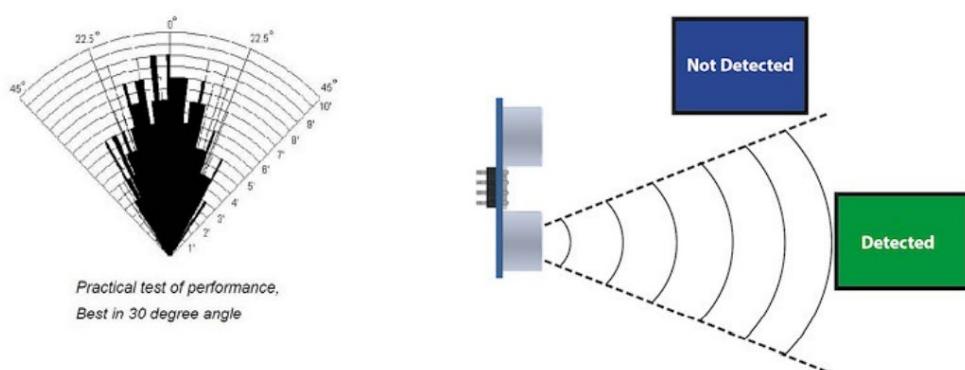
การทดลองนี้จะทำการตรวจวัดระยะทางด้วยโมดูลอัลตราโซนิก และแสดงผลออกทาง serial monitor และสามารถใช้ในการตรวจจับวัตถุ แจ้งเตือนเมื่อวัตถุเข้าใกล้เซ็นเซอร์มาก

โมดูลอัลตราโซนิก (Ultrasonic sensor)

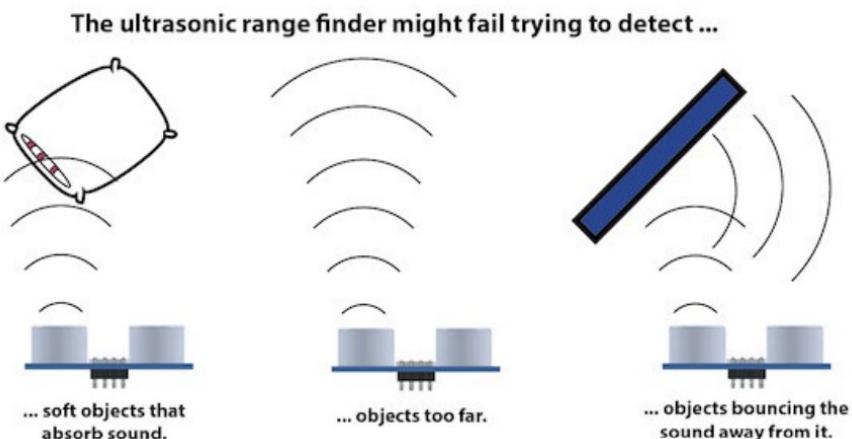
คือโมดูลที่ใช้คลื่นเสียงความถี่ในการส่ง และรับเพื่อระบุตำแหน่งระยะห่างของวัตถุนั้น ๆ โดยตัวส่งจะสร้าง คลื่นเสียงออกไป และเมื่อคลื่นกระแทกวัตถุ จะถูกสะท้อนมาให้กับตัวรับเพื่อนำไปประมวลผล โมดูล HC-SR04 วัดระยะห่างด้วยคลื่นอัลตราโซนิก (คลื่นเสียงความถี่ประมาณ 40 KHz) โดยคลื่นที่ ส่งออกไปจะเป็นรูปองศาของแสง (Beam Angle) หรือคล้าย ๆ กับแสงจากไฟฉายเมื่อเราเปิดในที่มืดนั่นเอง



ถึงคลื่นที่ส่งออกไปจะมีลักษณะเป็นรูปปีม แต่ก็ใช้ว่าจะสามารถตรวจเช็ครอบทิศได้ เพราะมีองศาในการวัดเพียง 15 องศาเท่านั้น

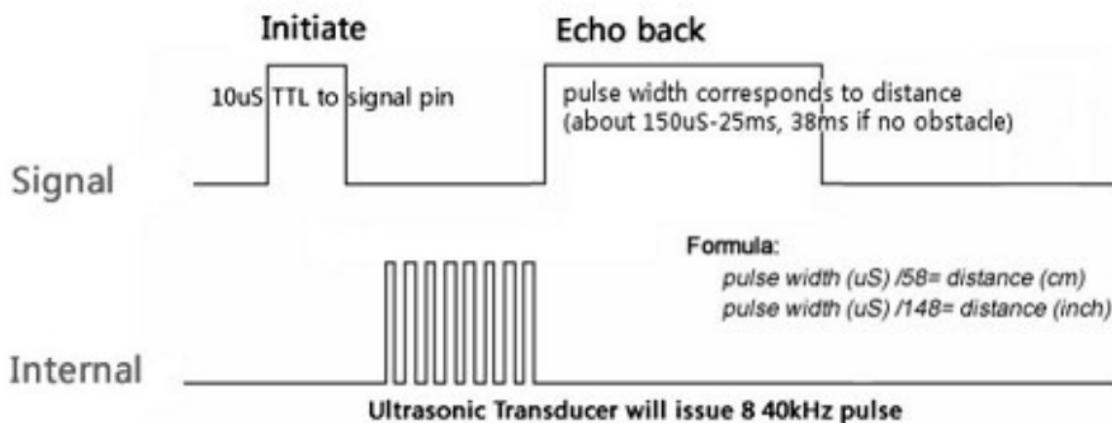


โดยโมดูล HC-SR04 มีขา TRIG (ตัวส่ง) และ ECHO (ตัวรับ) เพื่อส่งคลื่นอัลตราโซนิกในการวัดแต่ละครั้ง จะต้องสร้างสัญญาณพัลส์ (Pulse width) ที่มีความกว้างอย่างน้อย 10 ไมโครวินาที (10 microseconds) ป้อนเข้าขา Trig และวัดความกว้างของสัญญาณพัลส์ช่วงที่เป็น High จากขา Echo ประมาณ 150 ไมโครวินาที ถึง 25 มิลลิวินาที (150 microseconds - 25 milliseconds)



ข้อมูลของโมดูล HC-SR04

- จ่ายแรงดัน +5 V
- กินกระแส 15 mA
- ทำงานที่คลื่นความถี่ 40 KHZ
- สามารถวัดระยะทางประมาณ 2 cm - 4 m
- องศาในการวัด 15 องศา
- ความกว้างของสัญญาณพัลส์ที่ใช้ในการทริก 10 microseconds
- แรงดันเอาต์พุตอิจิกสำหรับขา TRIG และ ECHO ประมาณ 5 V (TTL)

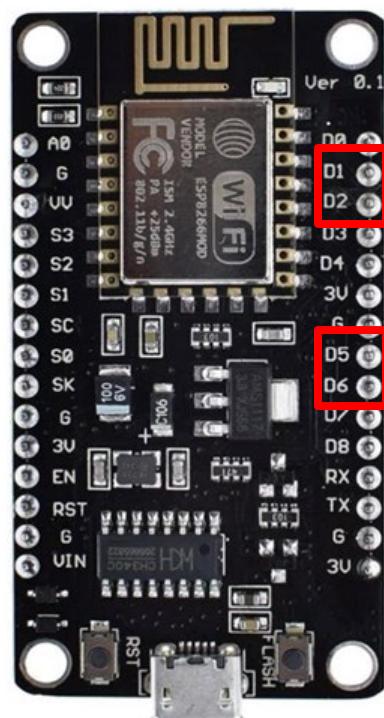
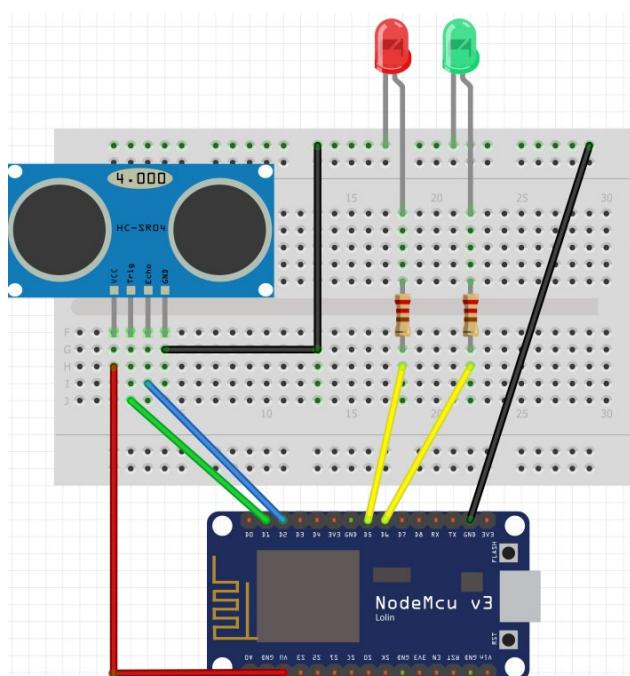


อุปกรณ์ที่ใช้

1. NodeMCU ESP8266
2. โมดูลอัลตราโซนิกเซนเซอร์
3. LED สีเขียวและสีแดง
4. ตัวต้านทานขนาด 220 โอห์ม 2 ตัว

การต่อวงจร

PIN	อุปกรณ์
D1	Ultrasonic (Trig)
D2	Ultrasonic (Echo)
D5	LED สีแดง
D6	LED สีเขียว



การเขียน code

- Code สำหรับการวัดระยะทาง

```
Workshop_HC-SR04_Distance

const int trigPin = D1;
const int echoPin = D2;

//กำหนดความเร้าของเสียงให้อยู่ในหน่วย cm/uS
#define SOUND_VELOCITY 0.034

long duration;
float distanceCm;
float distanceInch;

void setup() {
    //ตั้งค่า mode ให้ pin
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    digitalWrite(trigPin, LOW);
}

void loop() {
    // ตั้งค่า trigPin ให้มีค่า High เป็นเวลา 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // อ่านค่าจาก echoPin ซึ่งจะคือค่าเป็นเวลา sound wave travel (หน่วยเวลาเป็น micro seconds)
    duration = pulseIn(echoPin, HIGH);

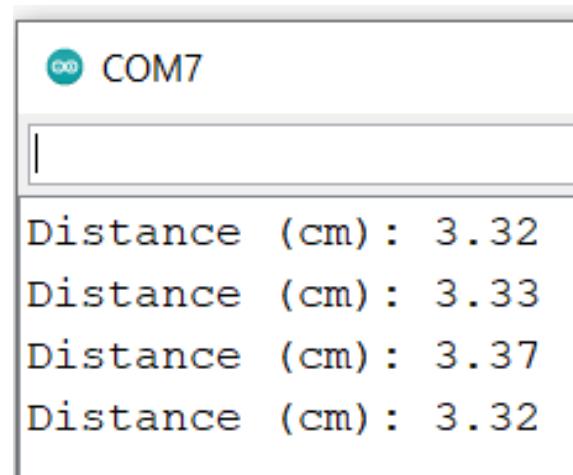
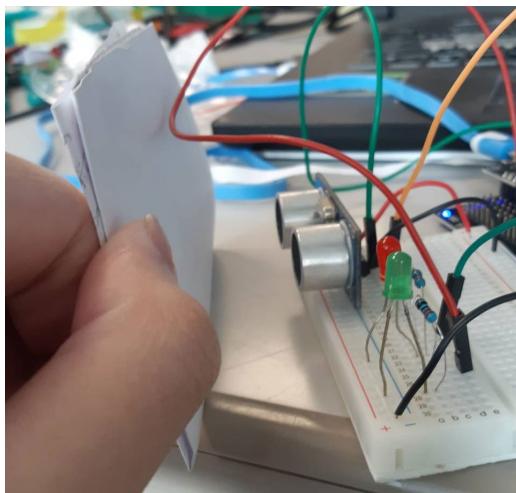
    // ค่าแทนระยะทาง
    distanceCm = duration * SOUND_VELOCITY/2;

    // แสดงระยะทาง
    Serial.print("Distance (cm): ");
    Serial.println(distanceCm);

    delay(1000);
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_5/Workshop_HC-SR04_Distance/Workshop_HC-SR04_Distance.ino

ผลลัพธ์ที่ได้



Code สำหรับการตรวจจับวัตถุ

Workshop_HC-SR04_Object_Detection

```

const int trigPin = D1;
const int echoPin = D2;

//กำหนดความเร้าของเสียงให้อยู่ในหน่วย cm/uS
#define SOUND_VELOCITY 0.034

long duration;
float distanceCm;
float distanceInch;
int LED1 = D5 ;
int LED2 = D6 ;

void setup() {
    //ตั้งค่า mode ให้ pin
    Serial.begin(9600);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    digitalWrite(trigPin, LOW);
}

```

```

void loop() {
    // ตั้งค่า trigPin ให้เป็น High เมื่อเวลา 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // จ่ายแค่จาก echoPin ซึ่งจะคืนค่าเมื่อเวลา sound wave travel (เมื่อเวลาเป็น micro seconds)
    duration = pulseIn(echoPin, HIGH);

    // ค่าหากระยะทาง
    distanceCm = duration * SOUND_VELOCITY/2;

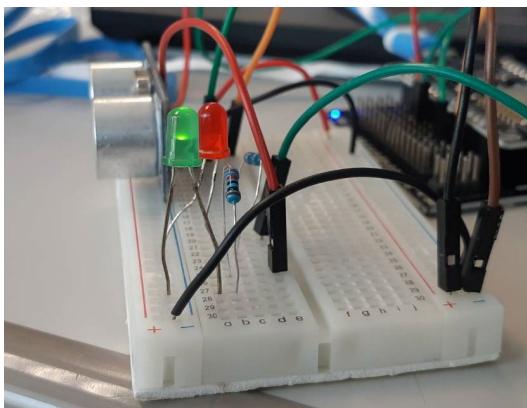
    if(distanceCm <= 10) // ใช้เกิดถ้ามีวัตถุเข้าใกล้
        digitalWrite(LED1, HIGH);
        digitalWrite(LED2, LOW);
    }
    else{
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, HIGH);
    }
    delay(1000);
}

```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_5/Workshop_HC-SR04_Object_Detection/Workshop_HC-SR04_Object_Detection.ino

ผลลัพธ์ที่ได้

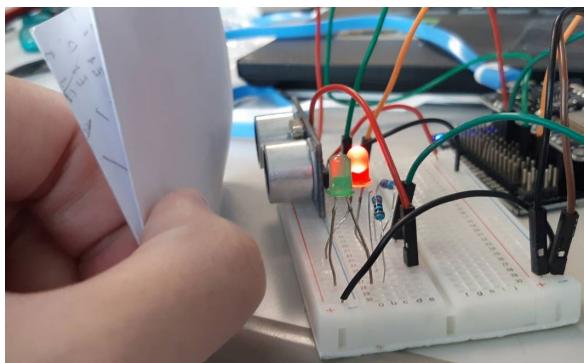
กรณีไม่มีวัตถุมาบัง sensor LED สีเขียวจะติดสว่างขึ้นมา



COM7

Distance (cm): 155.57
Distance (cm): 194.45
Distance (cm): 191.96
Distance (cm): 192.81

กรณีมีวัตถุมาบัง sensor LED สีแดงจะติดสว่างขึ้นมา



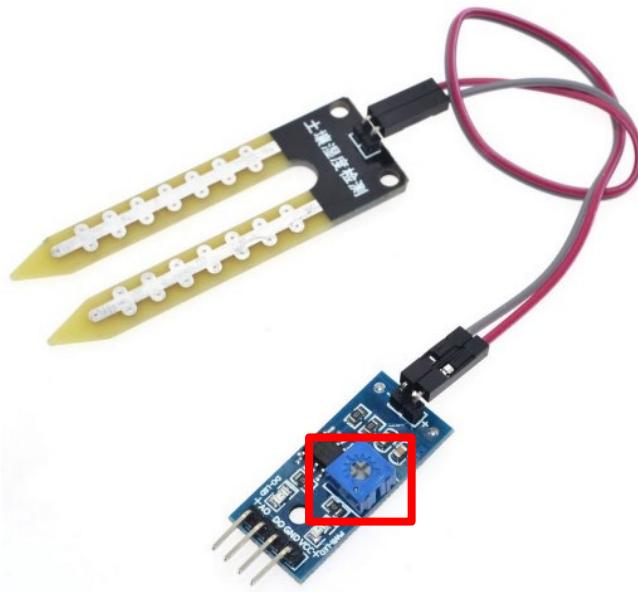
COM7

```
Distance (cm): 3.94
Distance (cm): 6.48
Distance (cm): 3.77
Distance (cm): 3.72
```

Workshop 6 วัดความชื้นในดินด้วย Soil Moisture Sensor

เช็คเซอร์วัตความชื้นในดิน Soil Moisture Sensor

สามารถต่อใช้งานกับไมโครคอนโทรลเลอร์โดยใช้อนาล็อกอินพุตอ่านค่าความชื้น หรือเลือกใช้สัญญาณดิจิตอลที่ส่งมาจากโมดูล สามารถปรับความไวได้ด้วยการปรับ Trimpot



หลักการทำงาน

เซ็นเซอร์ความชื้นในดินส่วนใหญ่ได้รับการออกแบบมาเพื่อวัดการปริมาณน้ำที่อยู่ในดินที่แพรพันตามค่าคงที่โดยอิเล็กทริก ค่าคงที่โดยอิเล็กทริกเป็นความสามารถของดินในการเป็นตัวกลางกระแสไฟฟ้า เนื่องจากน้ำเป็นตัวกลางที่ดีของกระแสไฟฟ้า ดังนั้นค่าคงที่โดยอิเล็กทริกของดินจะเพิ่มขึ้นเมื่อปริมาณน้ำในดินเพิ่มขึ้นนั่นเอง ด้วยสมมติฐานที่ว่าค่าคงที่โดยอิเล็กทริกของน้ำเป็นตัวกลางที่สะท้อนในการผ่าน กระแสไฟฟ้าเนื่องจากการวิ่งผ่านดินและอากาศในดิน ดังนั้นการวัดค่าคงที่โดยอิเล็กทริกทำให้สามารถประมาณ ค่าความชื้นได้

ตัวเซ็นเซอร์จะมี 2 ส่วน ส่วนแรกเป็นเซ็นเซอร์ร่างกายแต่จะก่ออิเล็กโอดที่ทำหน้าที่รับส่งค่ากระแสไฟฟ้าที่วิ่ง จากขาหนีงไปอีกขาหนีงโดยมีดินเป็นตัวกลาง ส่วนที่สองเป็นตัวขยายสัญญาณจากเซ็นเซอร์

หลักการวัดค่าของเซนเซอร์

ในอุปกรณ์เซนเซอร์จะมีอิซิอปแอมป์ LM393 เพื่อวัดแรงดันเบรียบเทียบกันระหว่างแรงดันที่วัดได้จากความชื้นในดิน กับแรงดันที่วัดได้จากการจราบงแรงดันปรับค่าโดยใช้ trimpot หากแรงดันที่วัดได้จากความชื้นของดิน มีมากกว่า ก็จะทำให้วงจรปล่อยล็อกจิก 1 ไปที่ขา D0 แต่หากความชื้นในดินมีน้อย ล็อกจิก 0 จะถูกปล่อยไปที่ขา D0

ขา A0 เป็นขาที่ต่อโดยตรงกับวงจรที่ใช้วัดความชื้นในดิน ซึ่งให้ค่าแรงดันออกมาตั้งแต่ 0 - 5V (ในทางอุดมคติ) โดยหากความชื้นในดินมาก แรงดันที่ปล่อยออกไปก็จะน้อย ในลักษณะของการแปรผกผัน

การนำไปใช้งาน

หากนำไปใช้งานด้านการวัดความชื้นแบบละเอียด แนะนำให้ใช้งานขา A0 ต่อเข้ากับไมโครคอนโทรลเลอร์เพื่อวัดค่าแรงดันที่ได้ ซึ่งจะได้ออกมาใช้เบรียบเทียบค่าความชื้นได้ หากมีความชื้นน้อย แรงดันจะใกล้ 5V มาก หากความชื้นมาก แรงดันก็จะลดต่ำลง

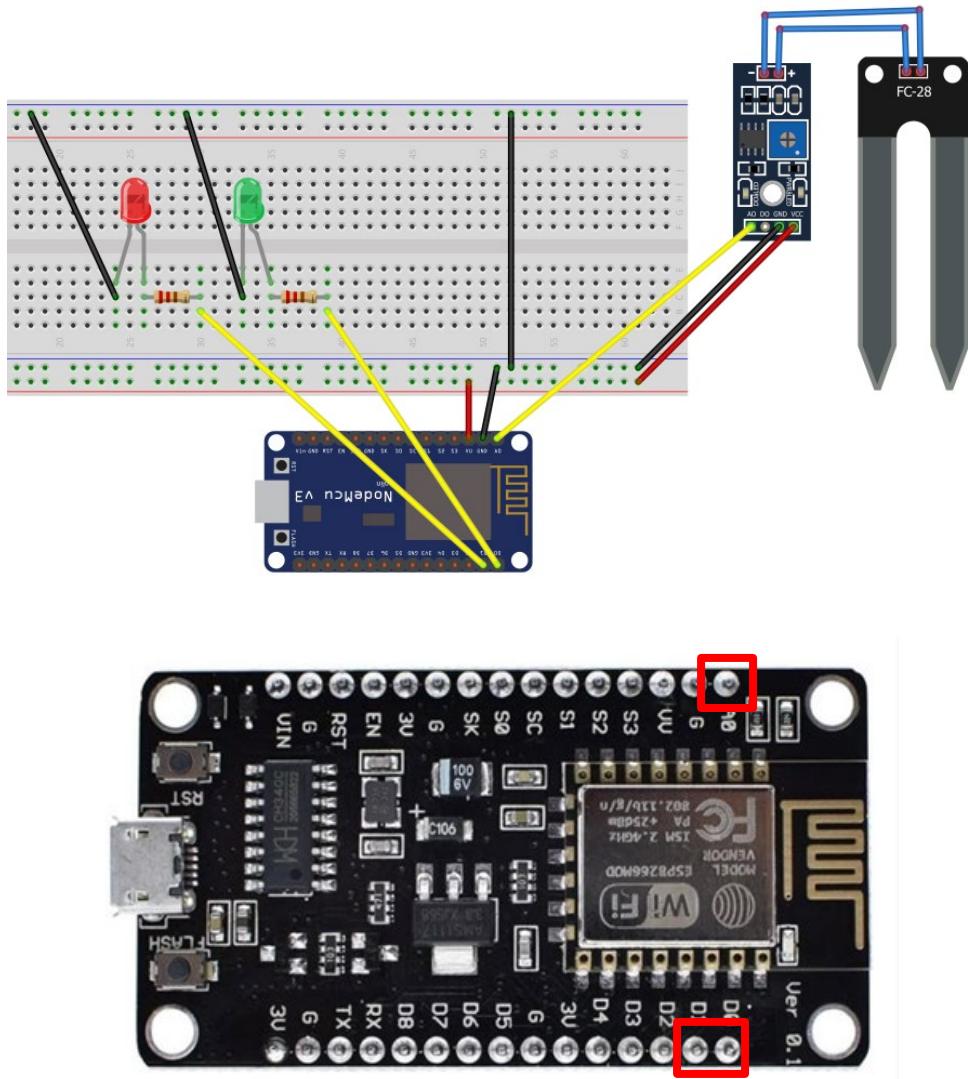
หากต้องการนำไปใช้ในโปรเจคที่ไม่ต้องใช้วัดละเอียด สามารถนำขา D0 ต่อเข้ากับทรานซิสเตอร์กำลังเพื่อส่งให้มีน้ำ หรือโซลินอยด์ทำงานเพื่อให้มีน้ำไหลมารดตันไม่ได้เลย เมื่อความชื้นในดินมีมากพอ จะปล่อยล็อกจิก 0 และทรานซิสเตอร์จะหยุดนำกระแส ทำให้มีน้ำหยุดปล่อยน้ำ

อุปกรณ์ที่ใช้

- NodeMCU ESP8266
- เซนเซอร์วัดความชื้นในดิน Soil Moisture Sensor
- LED สีแดงและสีเขียว
- ตัวต้านทานขนาด 220 โอห์ม 2 ตัว

การต่อวงจร

PIN	อุปกรณ์
A0	Soil Moisture Sensor
D0	LED สีเขียว
D1	LED สีแดง



การเขียน code

- Code สำหรับการอ่านค่า

```
Soil_Moisture_Sensor_Read

int analogPin = A0; //ประกาศตัวแปร ให้ analogPin แทนขา analog ขาที่5
int val = 0;
int map_val = 0 ;

void setup() {
    Serial.begin(9600);
}
//น้อยกว่า 500 คือมีความชื้น
void loop() {
    val = analogRead(analogPin); //อ่านค่าสัญญาณ analog ขา5 ที่ต่อกับ Soil Moisture Sensor Module ว่า
    //ปรับเปลี่ยนค่าจาก 0-1023 เป็น 0-100 เมื่อการแปลงผกผัน
    map_val = map(val, 0, 1023, 100, 0);
    Serial.print("val = "); // พิมพ์ข้อมูลความชื้น เข้าคอมพิวเตอร์ "val = "
    Serial.println(map_val); // พิมพ์ค่าของตัวแปร val
    delay(1000);
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_6/Soil_Moisture_Sensor_Read/Soil_Moisture_Sensor_Read.ino

* ค่าของ map_val คือการแปลงค่าให้อยู่ในค่าที่เราต้องการ เนื่องจากค่าที่ได้จาก sensor ถ้าดินมีความชื้นค่าที่จะได้จะเข้าใกล้ 0 แต่ถ้าดินแห้ง ค่าที่ได้จะเข้าใกล้ 1023 จึงทำการแปลงค่าให้อ่านค่าความชื้นในดินได้ง่ายขึ้น โดยที่ถ้าดินมีความชื้นมากค่าจะเข้าใกล้ 100 และถ้าดินแห้งค่าที่ได้จะเข้าใกล้ 0

ผลลัพธ์ที่ได้

ค่าความชื้นในดิน กรณี แห้ง

```
COM7

val = 1
```

ค่าความชื้นในดิน กรณี เปียก

```
COM7

val = 54
val = 54
val = 53
```

- Code สำหรับการอ่านค่า และแจ้งเตือนผ่าน LED

Soil_Moisture_Sensor_LED

```

int led_G = D0; //LED สีเขียว
int led_R = D1; //LED สีแดง
int analogPin = A0; //ประกาศตัวแปร ให้ analogPin แทนขา analog ขาที่5
int val = 0;
int map_val = 0 ;
void setup() {
    pinMode(led_G , OUTPUT); // sets the pin as output
    pinMode(led_R, OUTPUT); // sets the pin as output
    Serial.begin(9600);
}

void loop() {
    val = analogRead(analogPin); //อ่านค่าสัญญาณ analog ขา5 ที่ต่อ กับ Soil Moisture Sensor Module v1
    //ปรับเปลี่ยนค่าจาก 0-1023 เป็น 0-100 เมื่อการแปลงพอท
    map_val = map(val, 0, 1023, 100, 0);

    Serial.print("val = "); // พิมพ์ข้อมูลความชื้น เข้าคอมพิวเตอร์ "val = "
    Serial.println(map_val); // พิมพ์ค่าของตัวแปร val

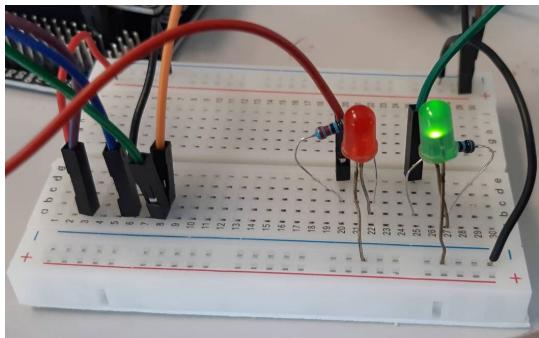
    if (map_val > 50) { //หากค่าที่ทำการแปลง map_val มีค่ามากกว่า 50 แสดงว่าดินเมื่อความชื้นเกิน 50 เมตรรัศมี
        digitalWrite(led_G , LOW); // สั่งให้ LED เขียวดับ
        digitalWrite(led_R, HIGH); // สั่งให้ LED สีแดง ติดสว่าง
    }
    else {
        digitalWrite(led_G , HIGH); // สั่งให้ LED เขียวติดสว่าง
        digitalWrite(led_R, LOW); // สั่งให้ LED สีแดง ดับ
    }
    delay(3000);
}

```

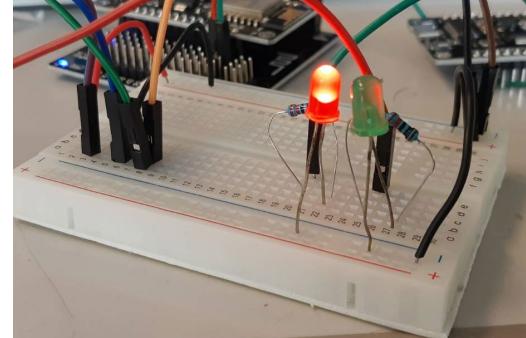
สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_6/Soil_Moisture_Sensor_LED/Soil_Moisture_Sensor_LED.ino

ผลลัพธ์ที่ได้

ค่าความชื้นในดิน กรณี แห้ง LED สีเขียวติด



ค่าความชื้นในดิน กรณี เปียก LED สีแดงติด



Workshop 7 วัดอุณหภูมิโดยใช้ DS18B20 และ แจ้งเตือนด้วยเสียงเตือน

DS18B20 คือ โมดูลเซนเซอร์อุณหภูมิแบบกันน้ำ วัดได้ทั้งอุณหภูมิห้องและแบบใส่ Probe เพื่อวัดของเหลวในท่อส่งหรือแบบจุ่มไปในของเหลวได้ ใช้ IC เบอร์ DS18B20 ผลิตโดย Dallas Semiconductor Corp. สายยาว 100 เซนติเมตร (ในท้องตลาดมีรุ่นสายยาว 2-3 เมตร) ต้องต่อตัว ต้านทาน 4.7K โอมร่วมด้วย ใช้งานง่ายและสามารถใช้งานกับ NodeMCU ESP8266 ได้ โดยการรับส่ง ข้อมูล นั้นจะใช้สายสัญญาณเส้นเดียวกันและเป็นสัญญาณแบบดิจิตอล ซึ่งเมื่อนำมาโคลคอกโนโตรเลอร์มา ใช้ต่อกับเซนเซอร์ตรวจวัดอุณหภูมิก็ สามารถเขียนคำสั่งให้มีอุณหภูมิสูงหรือต่ำกว่าค่าที่กำหนด ให้สั่ง อุปกรณ์ไฟฟ้าทำงานได้ เช่น ปั๊มน้ำ พัดลม หลอดไฟ เสียงเตือน



ข้อมูลของโมดูล DS18B20

1. แรงดันใช้งาน 3V ถึง 5V
2. กินกระแส 1mA
3. ช่วงการวัดอุณหภูมิ -55 ถึง 125 องศา
4. ความแม่นยำ ± 0.5 องศา
5. ความละเอียด 9 ถึง 12 bit (selectable)
6. เวลาในการประมวลผล น้อยกว่า 750ms
7. สายไฟ VCC สีแดง, GND สีดำ, DATA สีเหลือง
8. สัญญาณ output เป็นสัญญาณ ดิจิตอล

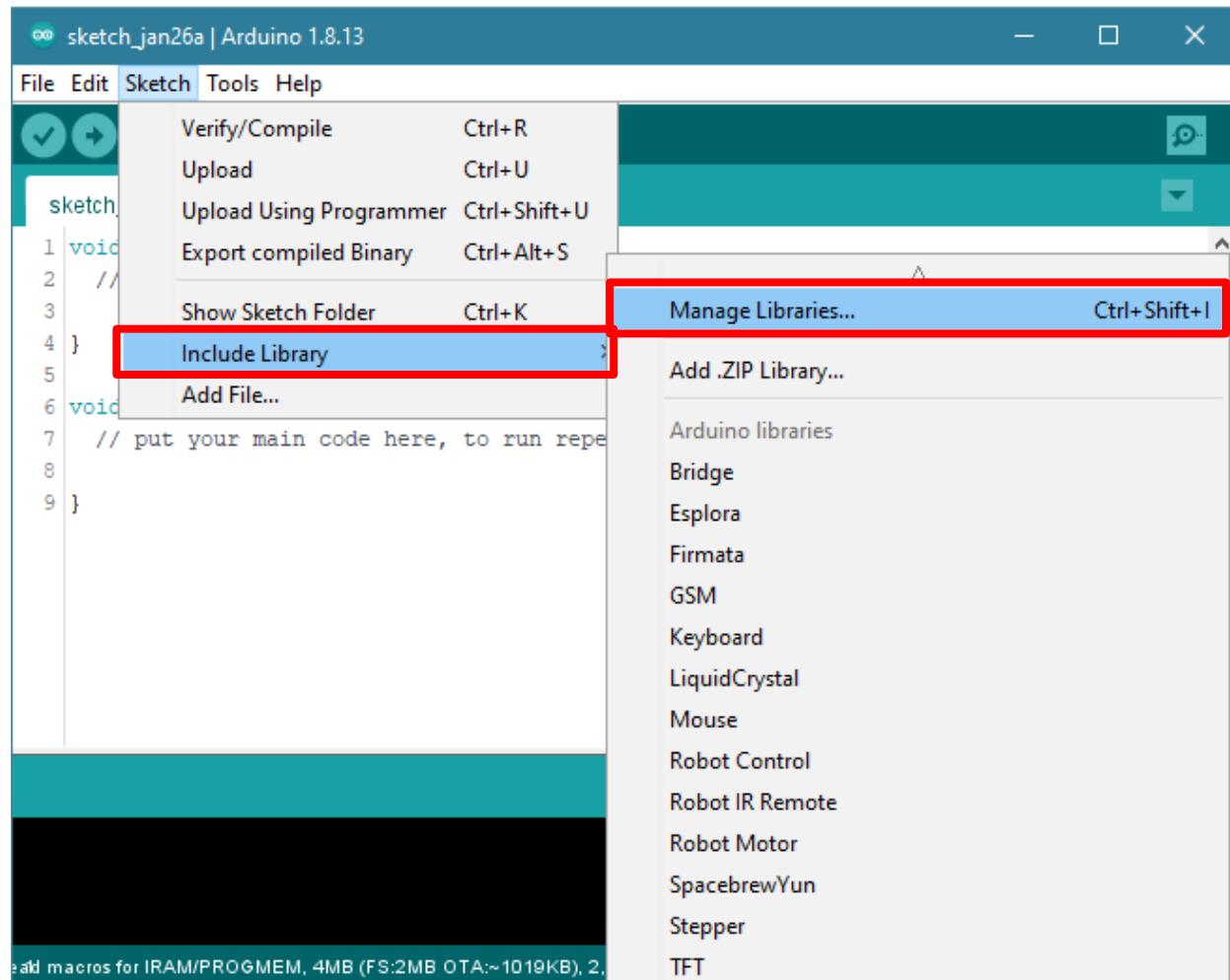
อุปกรณ์ที่ใช้

1. NodeMCU ESP8266
2. เซนเซอร์วัดอุณหภูมิ DS18B20
3. LED สีแดง
4. Buzzer
5. ตัวต้านทานขนาด 4.7k โอห์ม 1 ตัว และตัวต้านทาน 220 โอห์ม 1 ตัว

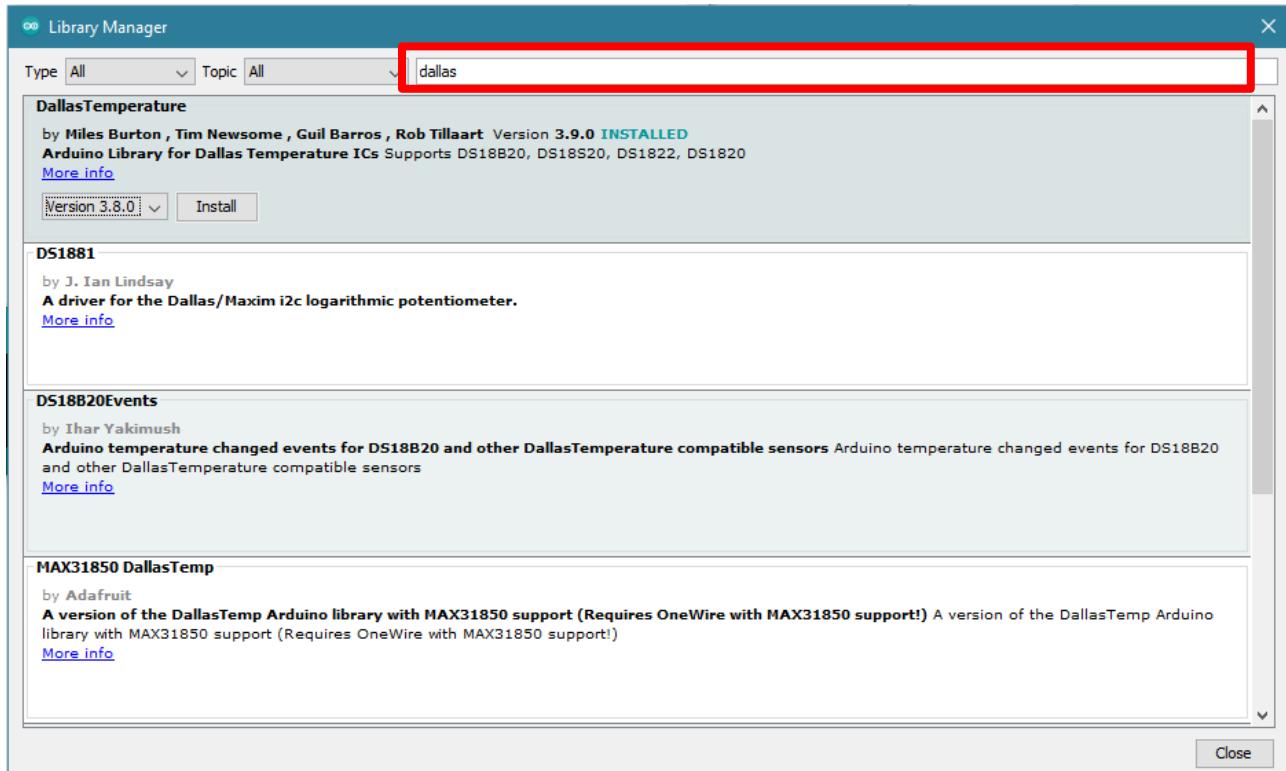
วิธีการติดตั้งโมดูล

ในการเขียนโปรแกรมเพื่อใช้งานโมดูลต่างๆ เราจะต้องศึกษารายละเอียดข้อมูลของตัวโมดูลซึ่ง อาจถูกกล่าวถึงการเลื่อนบิต (Shift bit) เพื่อความสะดวกในการใช้งานตัว Arduino IDE สามารถดาวน์โหลด Library ซึ่งเป็นการเขียนรวมคำสั่งที่โมดูลสามารถทำได้มาเขียนให้ใช้งานได้ง่ายขึ้น โดยผู้ที่เขียนอาจเป็นบุคคลหรือองค์กรก็ได้

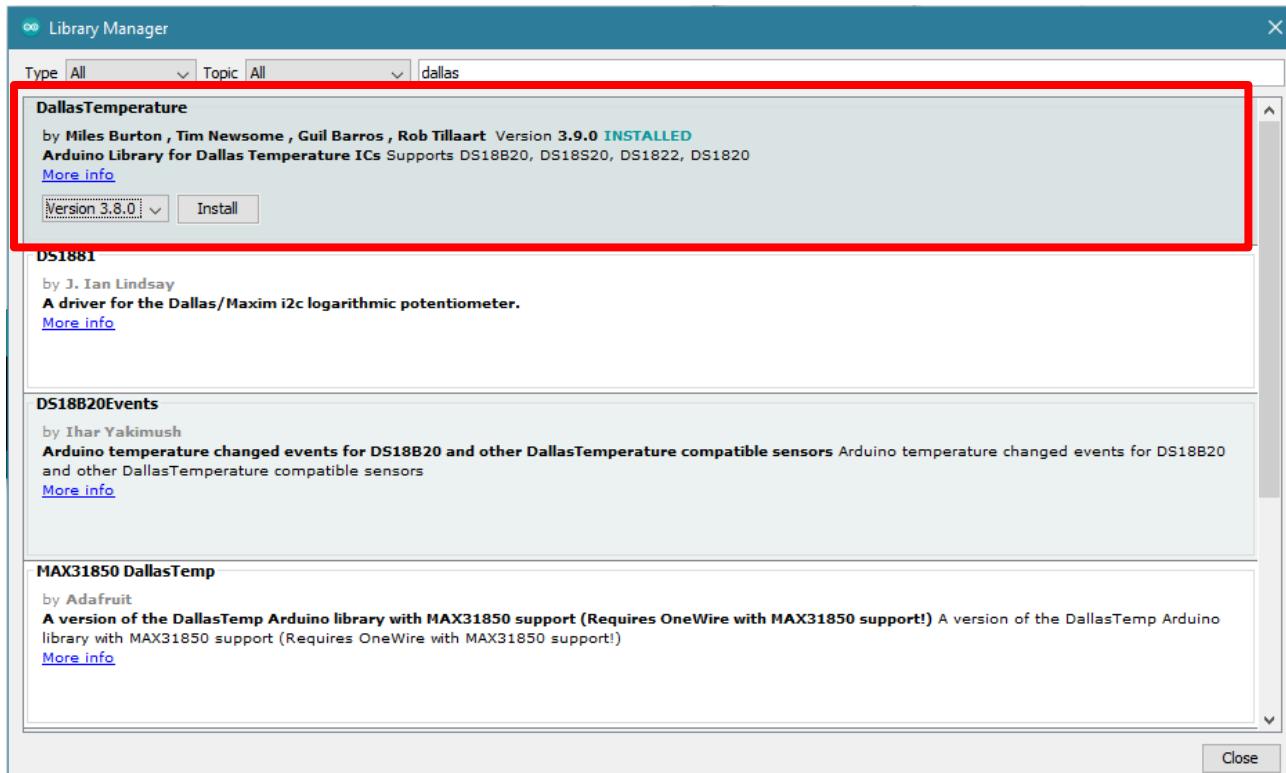
1. ในการติดตั้ง Library ให้ไปที่ Sketch -> Include Library -> Manage Libraries



2. ทำการพิมพ์ keyword ชื่อ dallas



3. กดเลือกเวอร์ชันล่าสุดและกด Install

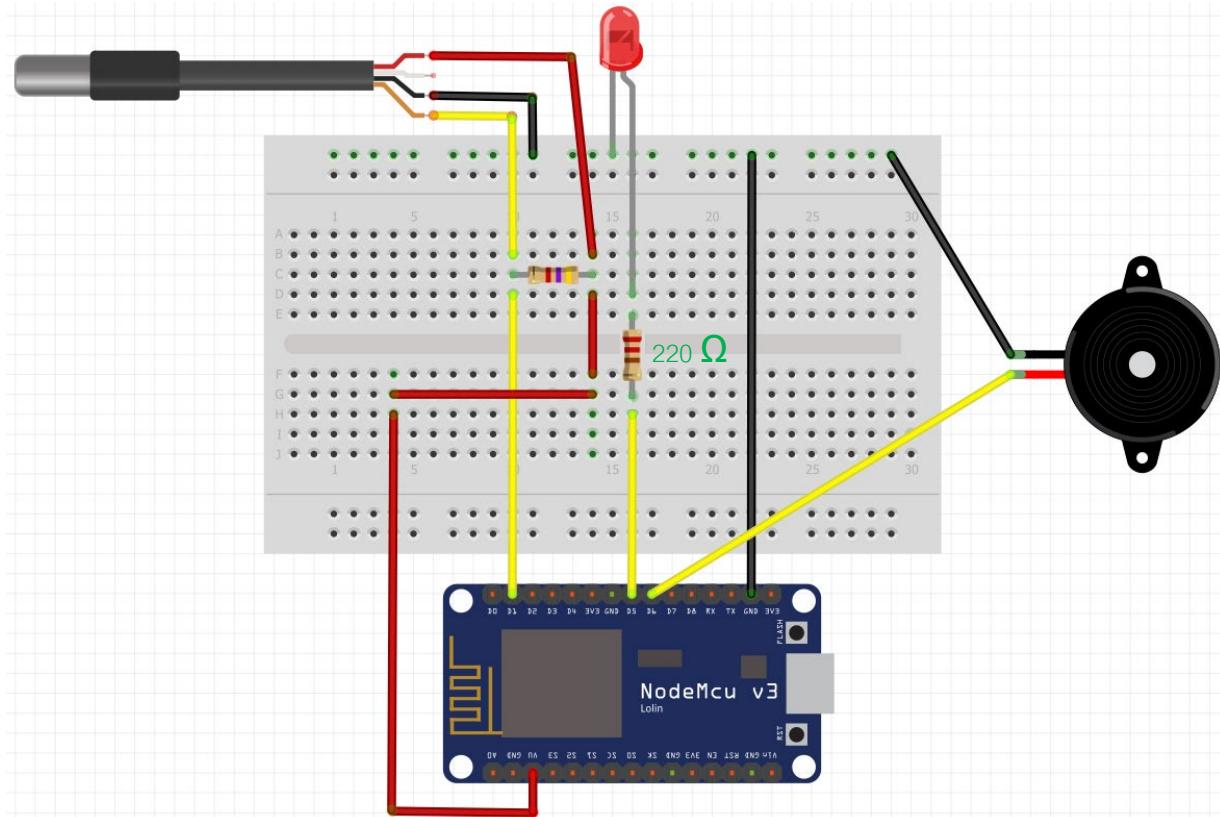


4 ทำการพิมพ์ keyword ชื่อ Onewire และเลือก Install ดังรูป



การต่อวงจร

PIN	อุปกรณ์
D1	DS18B20
D5	LED สีแดง
D6	Buzzer



การเขียน code

Workshop_7

```
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 5 //กำหนดขาที่จะเชื่อมต่อ Sensor
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
int ledPin = D5;
int Buzzer = D6;

void setup(void) {
    Serial.begin(9600);
    pinMode(ledPin, OUTPUT); // sets the pin as output
    pinMode(Buzzer, OUTPUT);
    Serial.println("Dallas Temperature IC Control Library");
    sensors.begin();
}

void loop(void) {
    Serial.println("Requesting temperatures...");
    sensors.requestTemperatures(); //อ่านข้อมูลจาก library
    Serial.print("Temperature is: ");
    Serial.print(sensors.getTempCByIndex(0)); // แสดงค่า อุณหภูมิ
    Serial.println(" *C");
    if (sensors.getTempCByIndex(0) > 32) {
        digitalWrite(ledPin, HIGH); // สั่งให้ LED สว่าง
        digitalWrite(Buzzer, HIGH); // สั่งให้ Buzzer ส่งเสียง
    }
    else {
        digitalWrite(ledPin, LOW); // สั่งให้ LED ดับ
        digitalWrite(Buzzer, LOW); // สั่งให้ Buzzer ดับ
    }
    delay(1000);
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_7/Workshop_7.ino

ผลลัพธ์ที่ได้

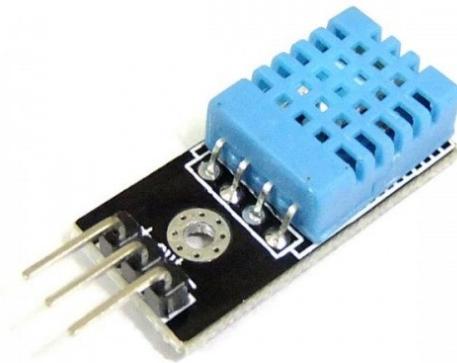
กรณีถ้า sensor ตรวจจับอุณหภูมิ ตรวจได้อุณหภูมินิมากกว่า 32 องศาตามที่เขียนไว้ในเงื่อนไข ก็จะทำให้ LED ติดและ Buzzer ส่งเสียง

```
COM7
|
Temperature is: 32.38 *C
Requesting temperatures...
Temperature is: 32.31 *C
Requesting temperatures...
Temperature is: 32.25 *C
Requesting temperatures...
Temperature is: 32.13 *C
Requesting temperatures...
Temperature is: 32.06 *C
Requesting temperatures...
Temperature is: 31.94 *C
Requesting temperatures...
Temperature is: 31.88 *C
```

Workshop 8 วัดอุณหภูมิด้วย DHT22



DHT22



DHT11

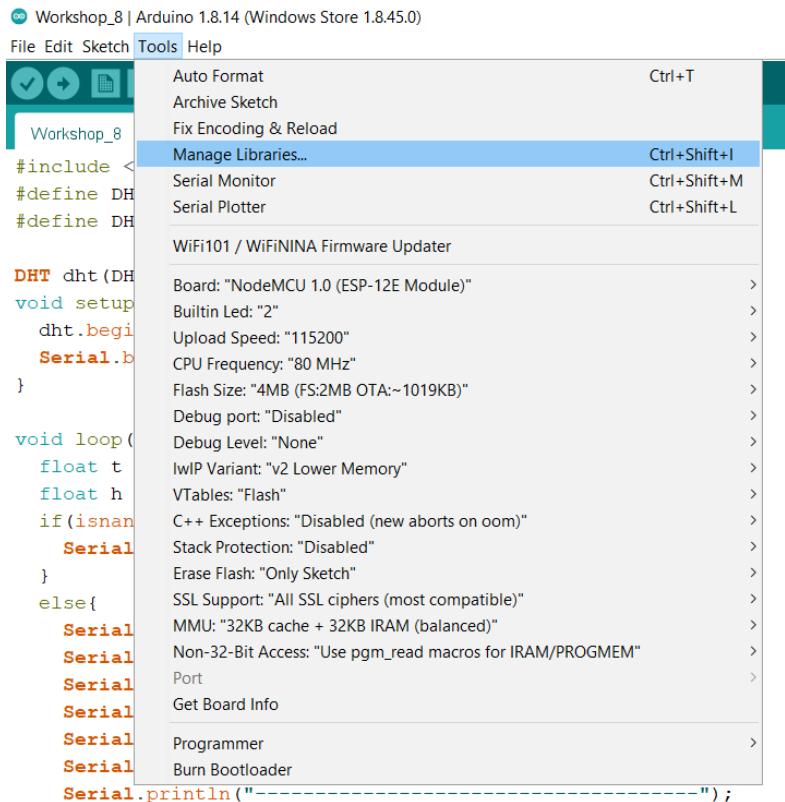
DHT22 เป็น sensor ที่ใช้ในการวัดอุณหภูมิและความชื้นในอากาศ ซึ่งออกแบบมาให้วัดได้แม่นยำกว่ารุ่น DHT11 แต่การเขียน code จะเขียนเหมือนกัน ดังนั้นเราสามารถนำ DHT22 ไปเปลี่ยนใช้งานแทน DHT11 ได้

ข้อมูลของ DHT22

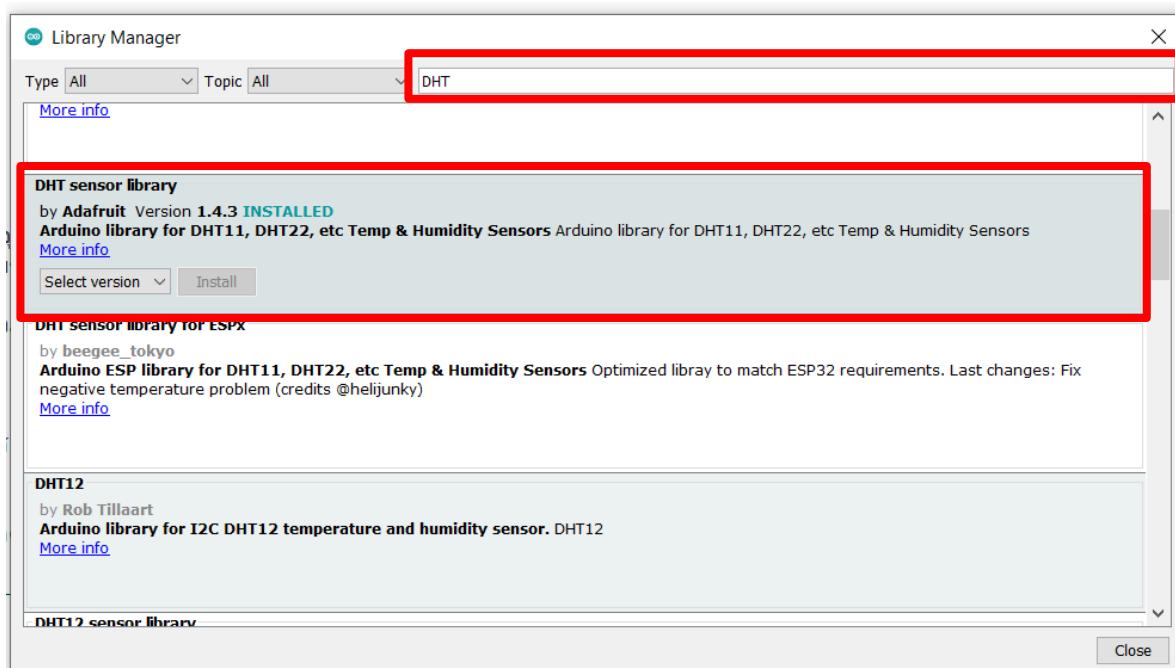
1. Power supply : 3 - 5.5V
2. วัดอุณหภูมิได้ระหว่าง 0 - 50 องศาเซลเซียส +/- 2 องศา
3. วัดความชื้นในอากาศได้ระหว่าง 20 - 90 % +/- 5%
4. เวลาที่ใช้ในการวัดค่า : 1 วินาที

ขั้นตอนการติดตั้งไลบรารี

1. กดเลือก Tool > Manage Libraries



2. พิมพ์ค้นหาว่า DHT เลือกติดตั้ง DHT sensor library by Adafruit (Version 1.4.3 หรือ Version ล่าสุด)

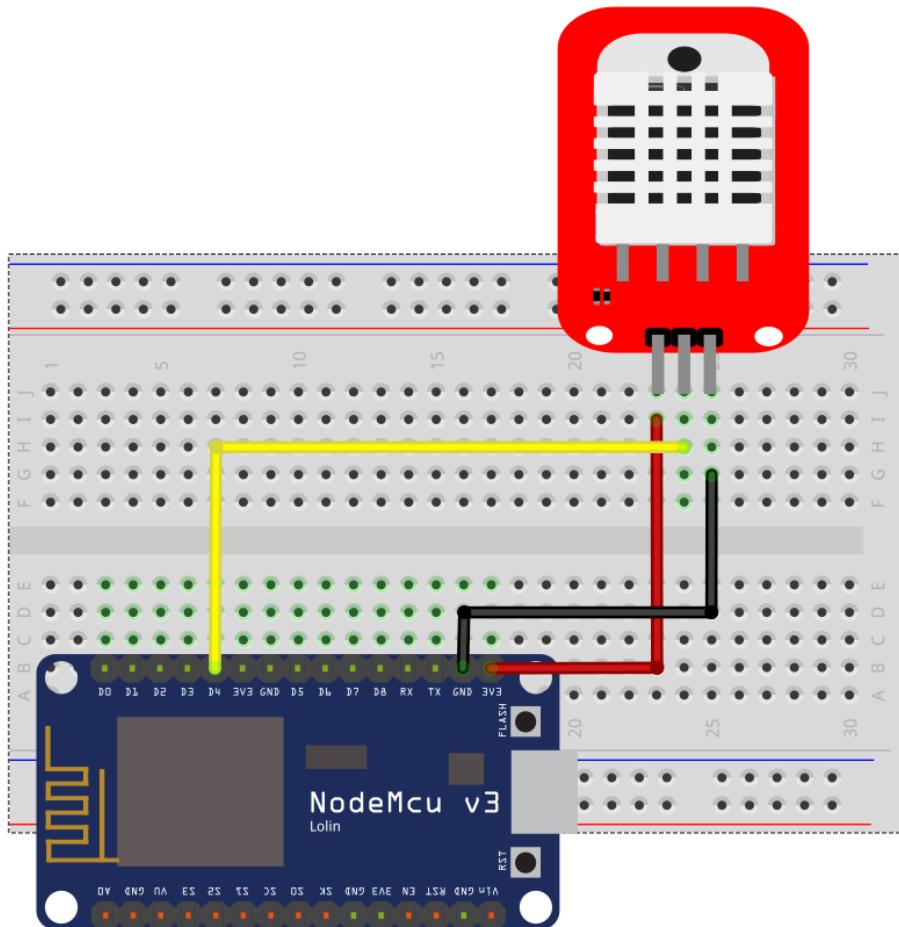


อุปกรณ์ที่ใช้

1. NodeMCU ESP8266
2. เซนเซอร์วัดอุณหภูมิ DHT22
3. สายไฟ

การต่อวงจร

PIN	อุปกรณ์
D4	DHT22



การเขียน code

Workshop_8

```
#include <DHT.h>
#define DHTPIN D4 //ใช้pin 4 สำหรับอ่านค่าจาก DHT22
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
void setup() {
    dht.begin(); //สั่งให้ DHT22 เริ่มทำงาน
    Serial.begin(115200);
}

void loop() {
    float t = dht.readTemperature(); //รับค่าอุณหภูมิในอากาศจาก DHT22
    float h = dht.readHumidity(); //รับค่าความชื้นในอากาศจาก DHT22
    if(isnan(t) || isnan(h)) { //เช็คค่าจาก DHT22 ว่ามีค่าส่งมาหรือไม่
        Serial.println("Failed!"); //ถ้าไม่มีค่าส่งมาจะขึ้นว่า failed
    }
    else{
        Serial.print("Temp :");
        Serial.print(t); //แสดงค่าอุณหภูมิในอากาศ
        Serial.println("*C");
        Serial.print("Humid : ");
        Serial.print(h); //แสดงค่าความชื้นในอากาศ
        Serial.println("%");
        Serial.println("-----");
    }
    delay(2000); //delay การทำงาน 2 วินาที
}
```

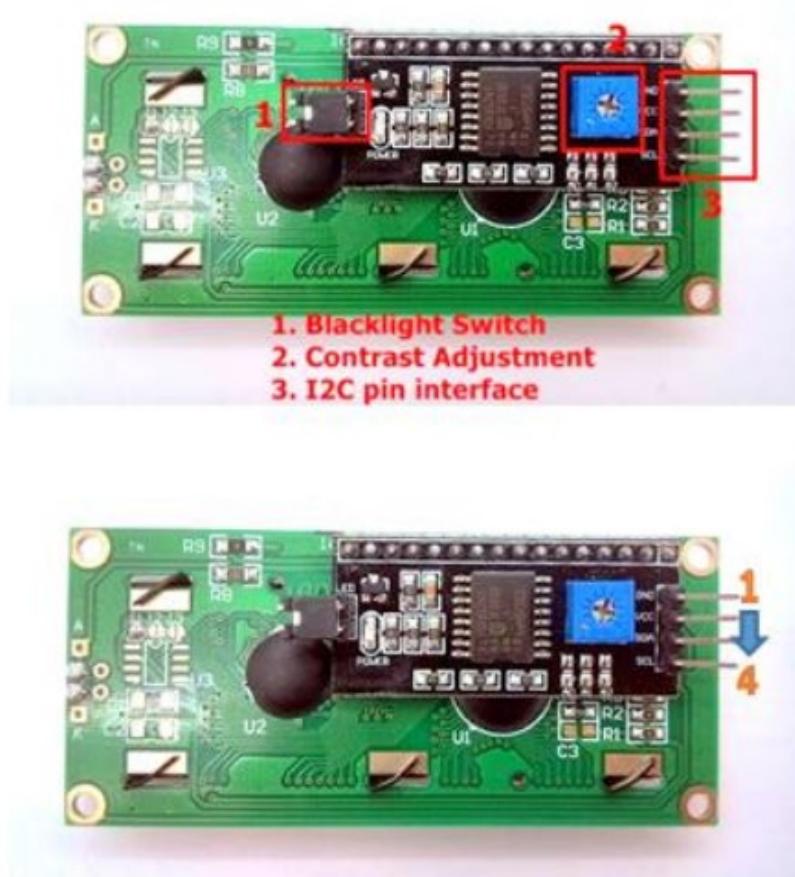
สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_8/Workshop_8.ino

ผลลัพธ์ที่ได้

```
Temp :26.90*C
Humid : 51.20%
-----
Temp :26.80*C
Humid : 51.00%
-----
Temp :26.80*C
Humid : 50.90%
-----
```

Workshop 9 การใช้ จอ LCD ในการแสดงค่าอุณหภูมิและค่าความชื้นจาก DHT22

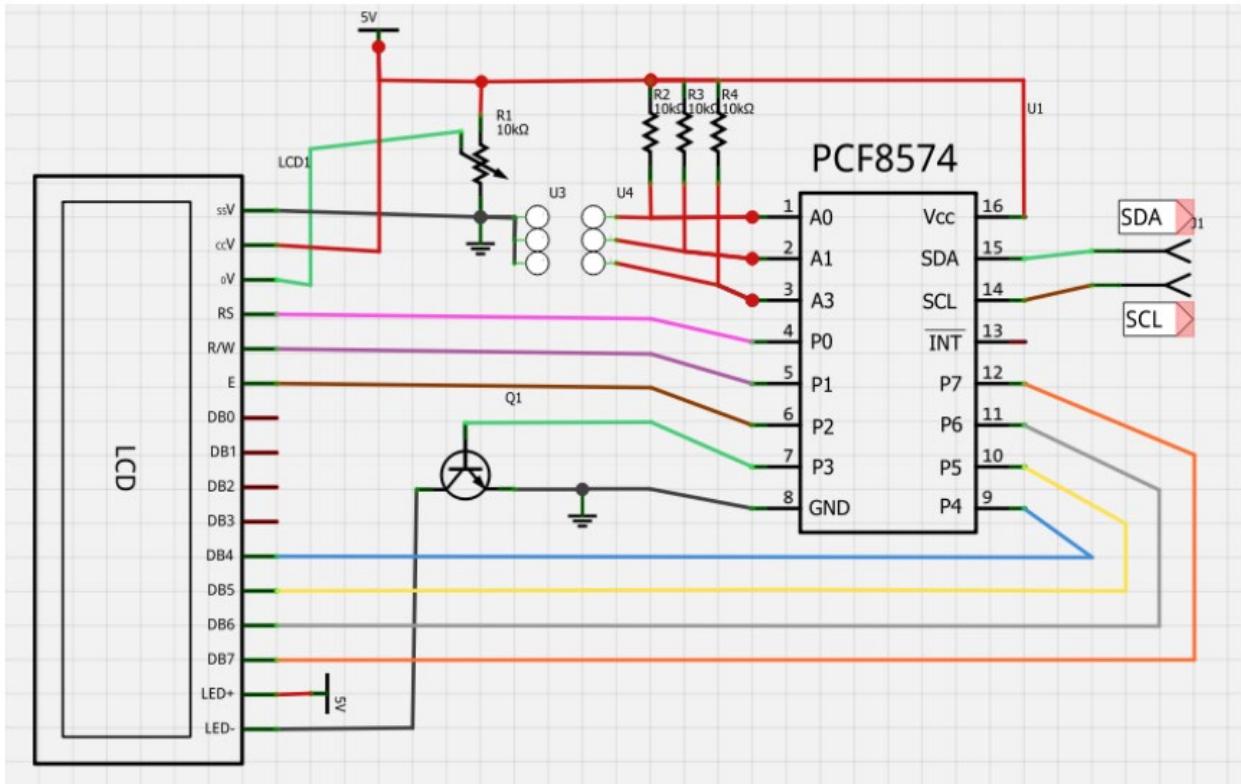
จอ LCD ที่มีการเชื่อมแบบ I2C หรือเรียกอีกอย่างว่าการเชื่อมต่อแบบ Serial เป็นจอ LCD ธรรมด้าทั่วไปที่มาพร้อมกับบอร์ด I2C Bus ที่ทำให้การใช้งานได้สะดวกยิ่งขึ้นและมาพร้อมกับ VR สำหรับปรับความเข้มของจอในรูปแบบ I2C ใช้ขาในการเชื่อมต่อ กับ Arduino เพียง 4 ขา (แบบ Parallel ใช้ 16 ขา) ซึ่งทำให้ใช้งานได้ง่ายและสะดวกมากยิ่งขึ้น



ตารางแสดงขาของจอ LCD 16x2 แบบอนุกรม (I2C)

ESP8266	LCD (I2c)
GND	GND (PIN 1)
Vin	VCC (PIN 2)
D1 (SCL)	SCL (PIN 3 serial clock)
D2 (SDA)	SDA (PIN 4 serial data)

สำหรับการเชื่อมต่อสัญญาณระหว่าง Arduino กับ LCD ที่มีบอร์ด I2C อยู่แล้วนั้นการส่งข้อมูลจาก Arduino ถูกส่งออกมาในรูปแบบ I2C ไปยังบอร์ด I2C และบอร์ดจะมีหน้าที่จัดการข้อมูลให้ออกมาในรูปแบบปกติ หรือแบบ Parallel เพื่อใช้ในการติดต่อไปยังจอ LCD โดยที่รหัสคำสั่งที่ใช้ในการสั่งงานของ LCD ยังคงไม่ต่างกับจอ LCD ที่เป็นแบบ Parallel โดยส่วนใหญ่บอร์ด I2C จะเชื่อมต่อกับตัวควบคุมของจอ LCD เพียง 4 บิตเท่านั้น วิธีการเชื่อมต่อที่ได้จะมีดังนี้



รูปการเชื่อมต่อระหว่าง Arduino กับ LCD (I2C)

(ที่มา www.loxhop.com)

จากรูป วงจรจอ LCD และบอร์ด I2C ได้มีการเชื่อมต่อขาสำหรับการรับส่งข้อมูลเป็นแบบ 4 บิต ขาที่ เชื่อมต่อไว้คือ ขา P4 > DB4, P5 > DB5, P6 > DB6, P7 > DB7 และขา P2 > E (Enable), P1 > R/W, P0 > RS รวมไปถึงตัวต้านทานสำหรับปรับค่าความเข้มของตัวอัคเซอร์ และ Switch Blacklight จากระยะห่างที่จำเป็นในการใช้งานถูกเชื่อมต่อเข้ากับตัวบอร์ด I2C และอุปกรณ์อิเล็กทรอนิกส์เรียบร้อยแล้ว



รูปโมดูล I2C Serial Interface Board Module

(ที่มา www.loxhop.com)

รายละเอียดคำสั่งในการสั่งงานระหว่าง Arduino กับ จอ LCD

คำสั่งในการควบคุมจอ LCD ของ Arduino นั้น ทาง Arduino.cc เขียนเป็น Library มาให้เพื่อสะดวกในการนำไปใช้งาน หลังจากต่อสายเสร็จเรียบร้อย ขั้นตอนแรกในการเริ่มเขียนโปรแกรมคือการเรียกใช้ Library ของ LCD จากไฟล์ชื่อ LiquidCrystal.h

ฟังก์ชัน LiquidCrystal(); ใช้ประกาศขาที่ต้องการส่งข้อมูลไปยังจอ LCD รูปแบบในการสั่งงานคือ

- LiquidCrystal lcd(rs, enable, d4, d5, d6, d7) <<< ในการนี้ใช้งานแบบ 4 บิต
- LiquidCrystal lcd(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7) <<< ในการนี้ใช้งานแบบ 8 บิต
- ใช้แบบ 4 บิต คือ LiquidCrystal lcd(12, 11, 4, 5, 6, 7); ก็หมายถึงการเชื่อมต่อ rs ที่ขา 12 , Enable ที่ขา 11 , และ DB4-DB7 ที่ขา 4-7 ของ Arduino ตามลำดับ

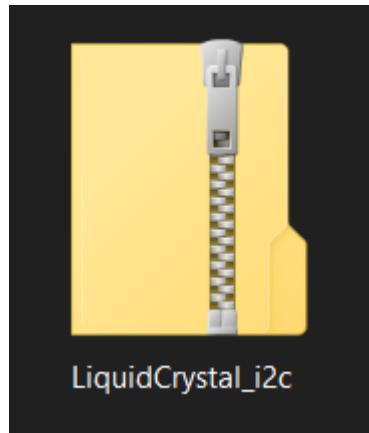
ฟังก์ชัน begin(); ใช้กำหนดขนาดของจอ ใช้ขนาด 16 ตัวอักษร 2 บรรทัด จึงประกาศเป็น lcd.begin(16, 2);

ฟังก์ชัน setCursor(); ใช้กำหนดตำแหน่งและบรรทัดของ Cursor เช่น lcd.setCursor(0, 1); คือให้เคอร์เซอร์ไปที่ตำแหน่งที่ 0 บรรทัดที่ 1 การนับตำแหน่งเริ่มจาก 0 ดังนั้น LCD 16x2 มีตำแหน่ง 0 – 15 บรรทัด คือ 0 กับ 1

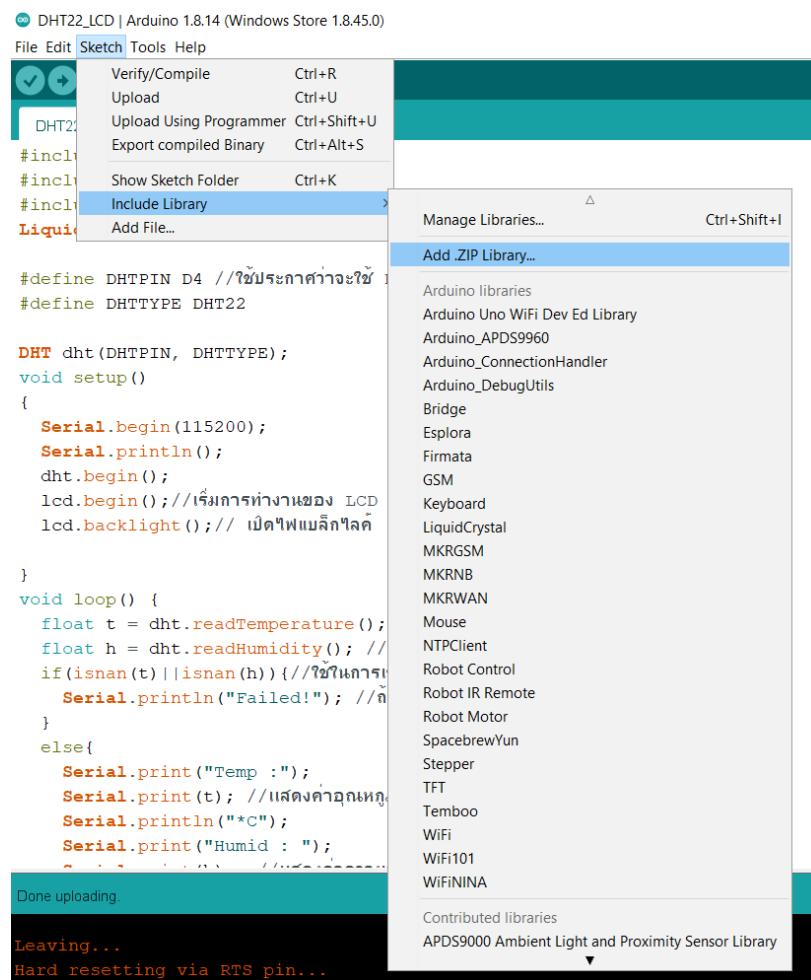
ฟังก์ชัน print(); ใช้กำหนดข้อความที่ต้องการแสดง เช่น lcd.print("TEST"); คือให้แสดงข้อความ “TEST” ออกทางหน้าจอ LCD

ขั้นตอนการติดตั้งไลบรารีจอแสดงผล LCD

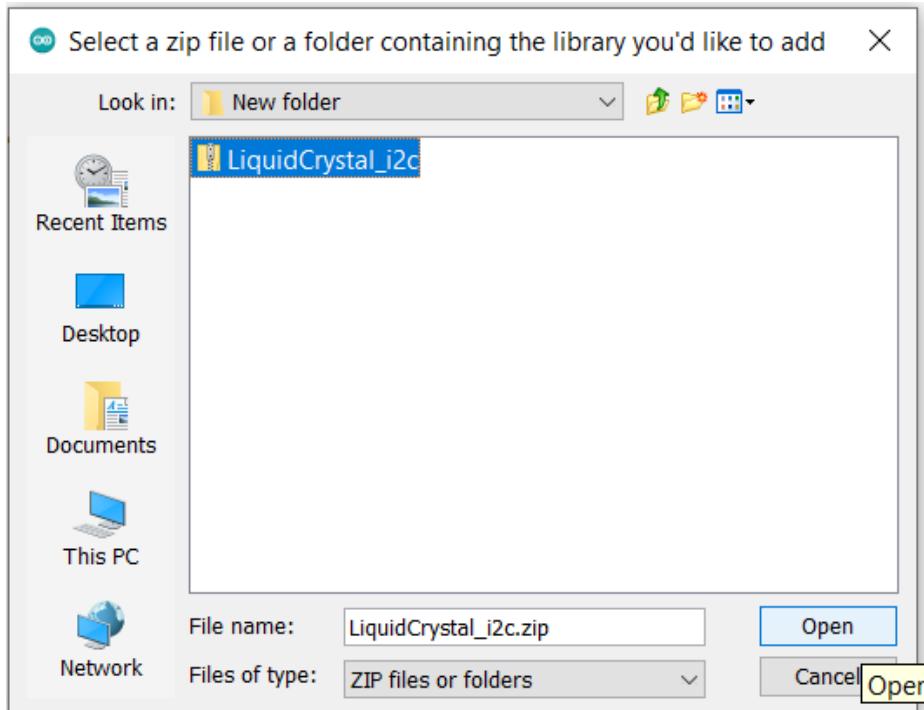
1. ดาวน์โหลดไฟล์ Zip LiquidCrystal_i2c



2. เปิดโปรแกรม Arduino IDE ขึ้นมา จากนั้นกดไปที่ Sketch > Include Library > Add .ZIP Library



3. ไปที่โฟลเดอร์ที่เราดาวน์โหลดไฟล์ Zip LiquidCrystal_i2c ไว้ จากนั้นกด open



พังก์ชันสั่งงานจอ LCD

คำสั่ง	การทำงาน
lcd.clear()	ใช้ล้างหน้าจอ เมื่อมีตัวอักษรใด ๆ อยู่บนหน้าจอ จะถูกล้างออกทั้งหมด
lcd.home()	ใช้ปรับให้เคอร์เซอร์กลับไปอยู่ที่ตำแหน่งแรกด้านซ้าย เมื่อใช้คำสั่ง lcd.print() จะไปเริ่มแสดงผลทางด้านบนซ้าย
lcd.setCursor(x,y)	x คือ ลำดับตัวอักษรนับจากทางซ้าย y คือ บรรทัด ใช้ตั้งค่าเคอร์เซอร์ เช่น lcd.setCursor(2, 0); หมายถึงเช็ตเคอร์เซอร์ไปตัวอักษรที่ 2 นับจากทางซ้ายและอยู่บรรทัดแรก เมื่อใช้คำสั่ง lcd.print() ตัวอักษรตัวแรกจะอยู่ลำดับที่ 3 นับจากทางซ้าย
lcd.write(ข้อมูลที่ต้องการเขียนออกไป)	ใช้สำหรับเขียนข้อมูลออกไปที่ลิตตัวอักษร

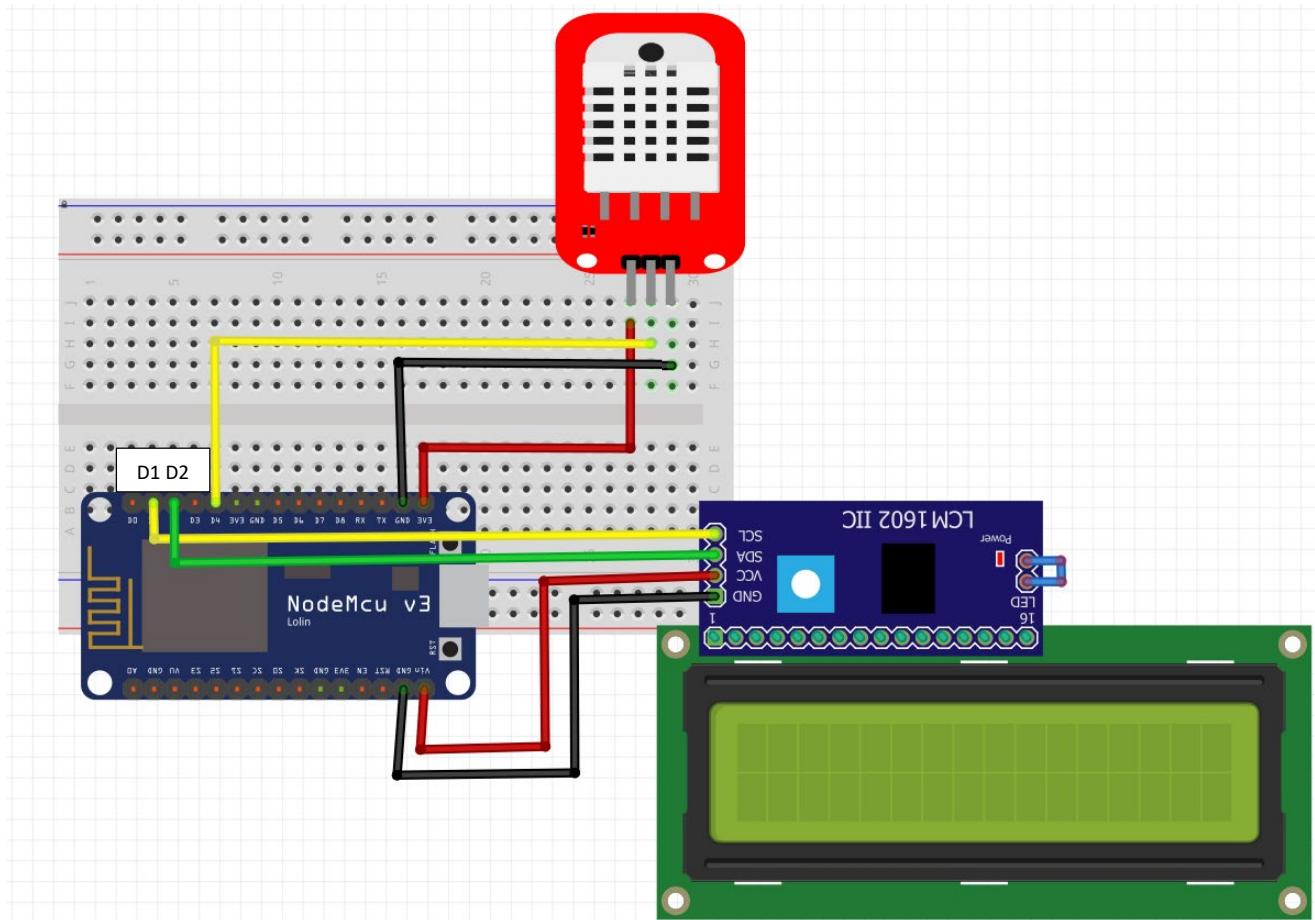
คำสั่ง	การทำงาน
lcd.print(x , y)	x คือ ข้อมูลที่ต้องการให้เขียนออกไป y คือ รูปแบบข้อมูล ใช้เขียนข้อมูลออกไปทั้งชื่อความ
lcd.cursor()	ใช้สั่งให้แสดงเครื่องเซอร์บันหน้าจอ
lcd.noCursor()	ใช้สั่งให้ไม่แสดงเครื่องเซอร์บันหน้าจอ
lcd.display()	แสดงตัวอักษรบนหน้าจอ
lcd.noDisplay()	ปิดการแสดงตัวอักษรในหน้าจอ
lcd.scrollDisplayLeft()	เลื่อนตัวอักษรไปทางซ้าย 1 ตัว
lcd.scrollDisplayRight()	เลื่อนตัวอักษรไปทางขวา 1 ตัว
lcd.autoscroll()	เลื่อนตัวอักษรไปทางขวาอัตโนมัติหากใช้คำสั่ง lcd.print() หรือ lcd.write() เมื่อตัวอักษรเต็มหน้าจอ
lcd.noAutoscroll()	ปิดการเลื่อนตัวอักษรอัตโนมัติ
lcd.leftToRight()	เมื่อใช้คำสั่ง lcd.print() หรือ lcd.write() ตัวอักษรจะเขียนจากซ้ายไปขวา
lcd.rightToLeft()	เมื่อใช้คำสั่ง lcd.print() หรือ lcd.write() ตัวอักษรจะเขียนจากขวาไปซ้าย

อุปกรณ์ที่ต้องใช้

1. NodeMCU ESP8266
2. เซนเซอร์วัดอุณหภูมิ DHT22
3. สายไฟ
4. จอ LCD

การต่อวงจร

PIN	อุปกรณ์
D1	จอ LCD (SCL)
D2	จอ LCD (SDA)
D4	DHT22



การเขียน code

Workshop_9

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

#define DHTPIN D4 //ใช้ประกาศว่าจะใช้ PIN D4 ในการรับข้อมูลจาก DHT22
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
void setup()
{
    Serial.begin(115200);
    Serial.println();
    dht.begin();
    lcd.begin(); //เริ่มการทำงานของ LCD
    lcd.backlight(); // เปิดไฟแบล็คไลท์

}

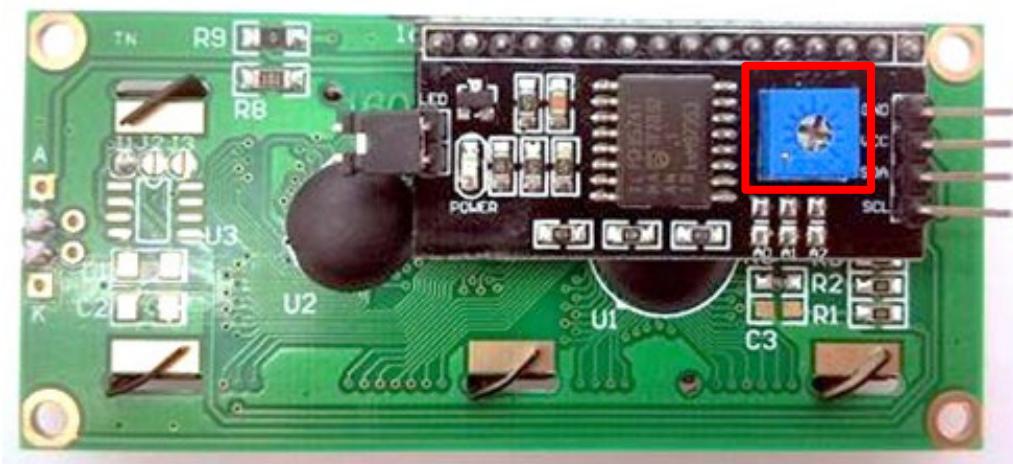
void loop() {
    float t = dht.readTemperature(); //รับค่าอุณหภูมิจาก DHT22
    float h = dht.readHumidity(); //รับค่าความชื้นจาก DHT22
    if(isnan(t) || isnan(h)) { //ใช้ในการเช็คค่าจาก DHT22 ว่ามีค่าส่งมาหรือไม่
        Serial.println("Failed!"); //ถ้าไม่มีค่าส่งมาจะขึ้นคำว่า failed
    }
    else{
        Serial.print("Temp :");
        Serial.print(t); //แสดงค่าอุณหภูมิ
        Serial.println("*C");
        Serial.print("Humid : ");
        Serial.print(h); //แสดงค่าความชื้น
        Serial.println("%");
        Serial.println("-----");
        delay(500);
        lcd.setCursor(0, 0); //เลื่อนเคอเรอร์ไปบรรทัดที่ 1 ลำดับที่ 0
        lcd.print("hum:      ");
        lcd.setCursor(4, 0); //เลื่อนเคอเรอร์ไปบรรทัดที่ 1 ลำดับที่ 4
        lcd.print(h); //แสดงค่าความชื้น
        lcd.setCursor(9, 0); //เลื่อนเคอเรอร์ไปบรรทัดที่ 1 ลำดับที่ 9
        lcd.print("%");
        lcd.setCursor(0, 1); //เลื่อนเคอเรอร์ไปบรรทัดที่ 2 ลำดับที่ 0
        lcd.print("Tem:      ");
        lcd.setCursor(4, 1); //เลื่อนเคอเรอร์ไปบรรทัดที่ 2 ลำดับที่ 4
        lcd.print(t); //แสดงค่าอุณหภูมิ
        lcd.setCursor(9, 1); //เลื่อนเคอเรอร์ไปบรรทัดที่ 2 ลำดับที่ 9
        lcd.print("C");
        delay(500);
    }
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_9/Workshop_9.ino

ผลลัพธ์ที่ได้



* ในการณีที่หน้าจอ LCD ไม่แสดงผลลัพธ์ ให้นำไขควงไปหมุนปรับการแสดงผลที่ โมดูล I2C ด้านหลังจอ LCD

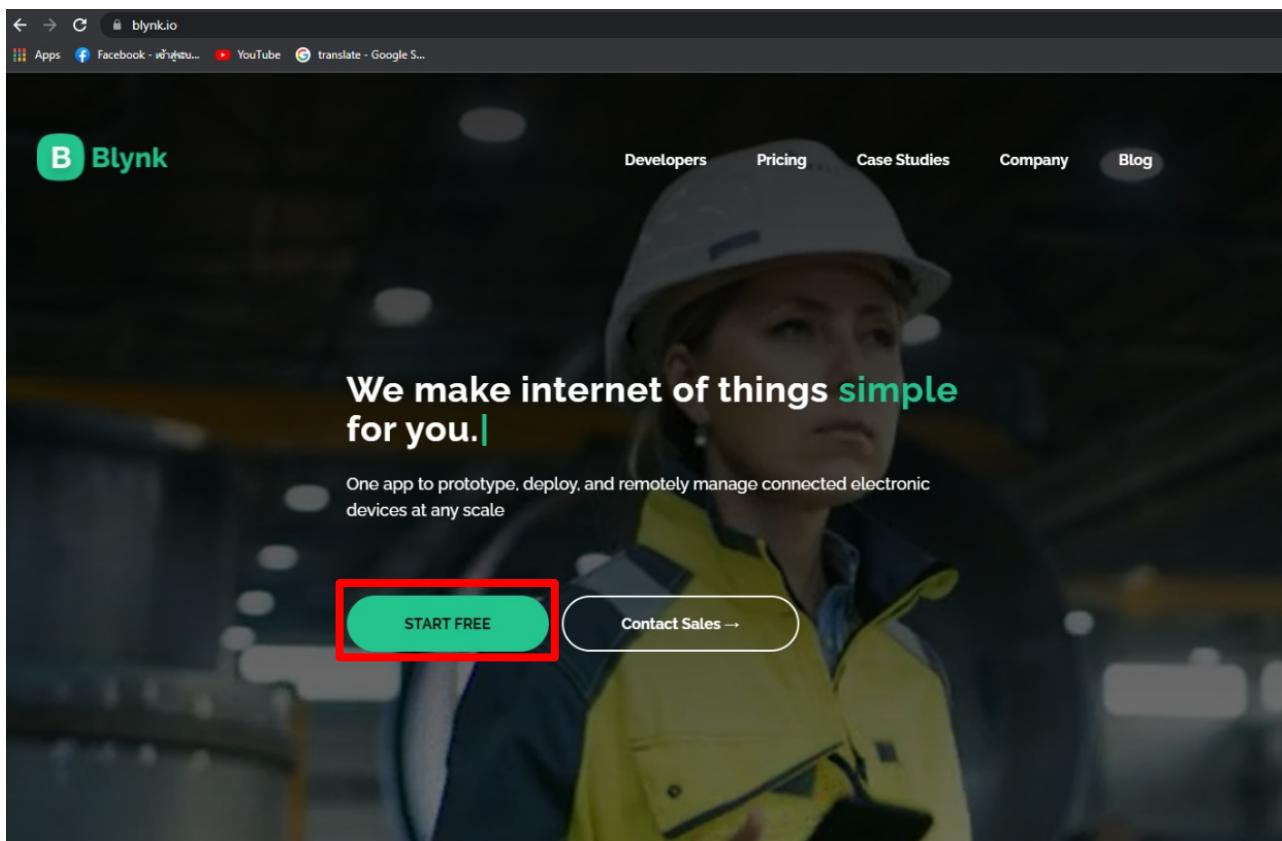


Blynk

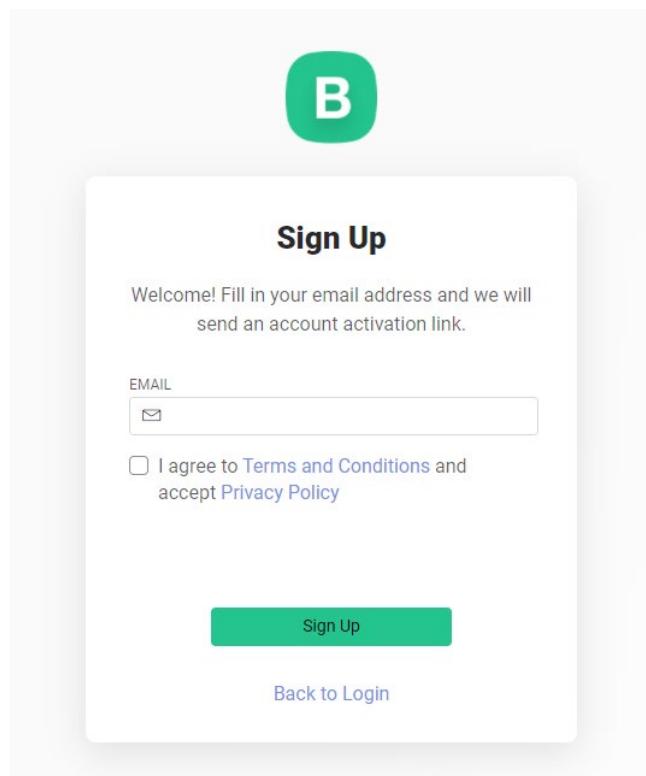
Blynk เป็น platform ที่ใช้ในการควบคุมหรือเชื่อมต่ออุปกรณ์ด้าน IoT ผ่าน Mobile Application หรือผ่าน Web Server โดยตัวซอฟต์แวร์เป็นตัวกลางระหว่างอุปกรณ์กับแอปพลิเคชัน โดยตัวแอปพลิเคชัน รองรับระบบ Android และ iOS

ขั้นตอนการติดตั้ง

1. ให้เปิดเว็บไซต์ blynk.io
2. กดปุ่ม START FREE เพื่อทำการสมัคร



3. ทำการสมัครโดยใช้ Email และทำการ log in



เมื่อ log in เรียบร้อยแล้วจะอยู่ในหน้านี้

My organization - 6526IT

My Devices

1 Device

Device name	Device owner	Status	Device model	Last updated	Organization	A	Actions
Test new Blynk	Pasawit	Offline		6:57 PM Oct 24, 2021	My organization - 6526IT	9:	

DEVICES

My Devices 1
All 1

LOCATIONS

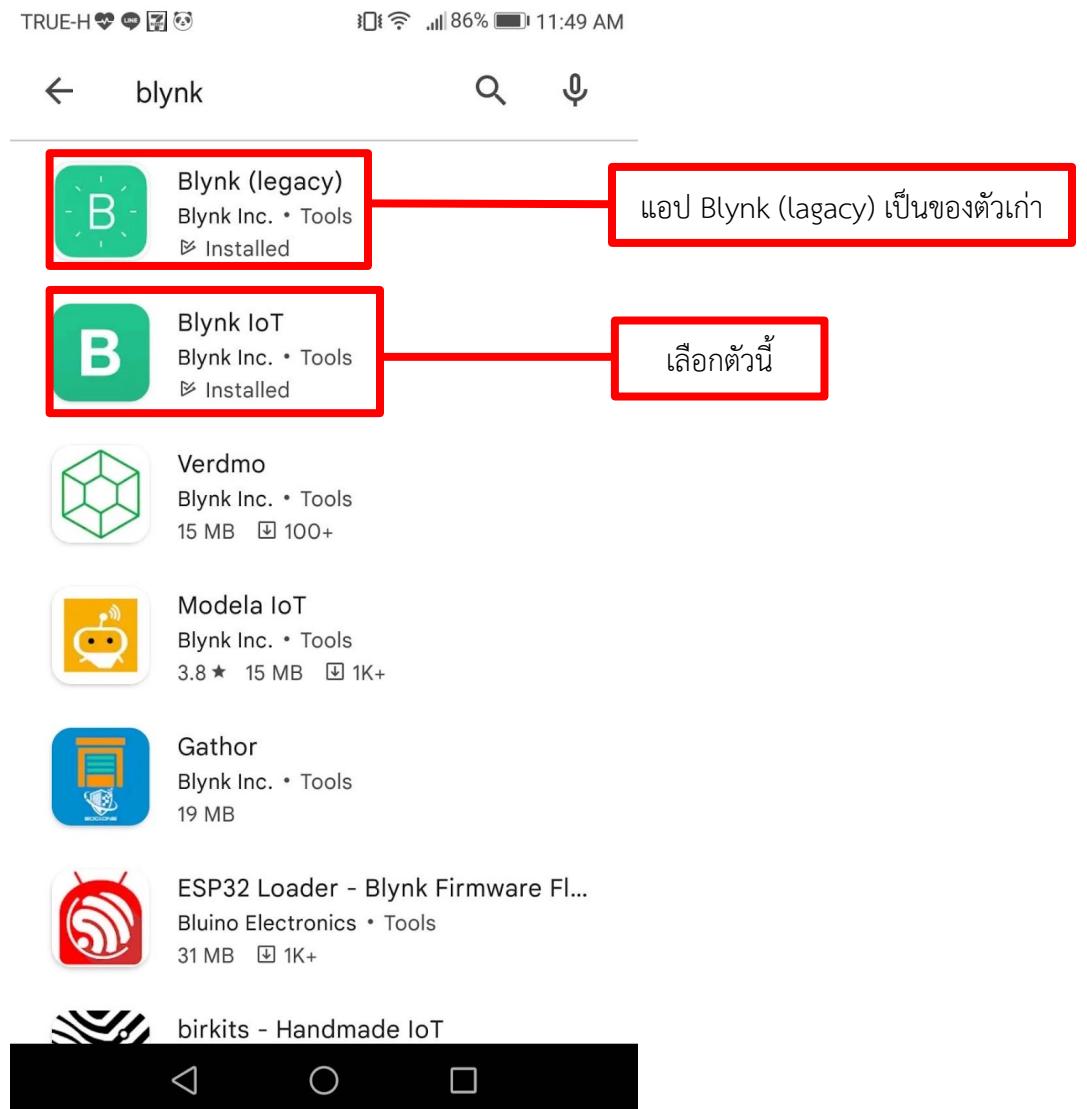
My Locations 0
All 0

USERS

My organization members 1
All 1
With no devices 0

Build Date: 11:57 18.11.2021, Commit Hash: 8ec71777b5, Commit Date: 11:55 18.11.21 logon: sgp1 | [Privacy Policy](#)

4. ในส่วนของ Application บน Smartphone ให้ค้นหาแอปพลิเคชันชื่อ Blynk IoT



5. ให้ทำการ log in เข้าระบบ



Sign Up

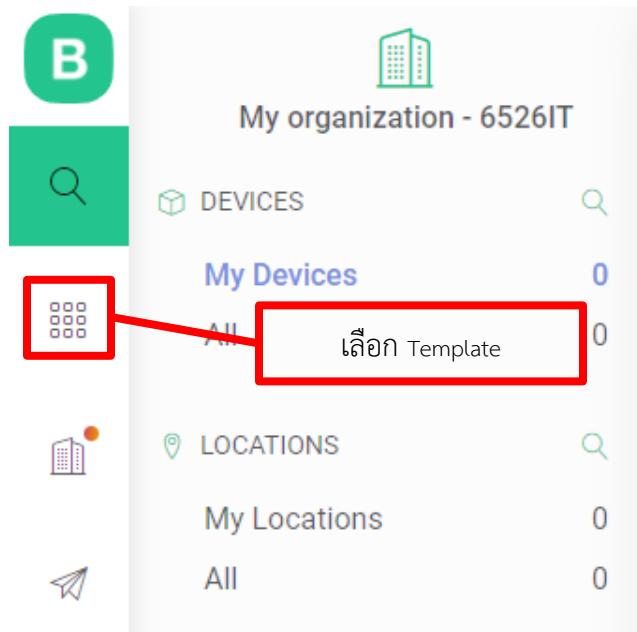
Log In

กดตรงนี่

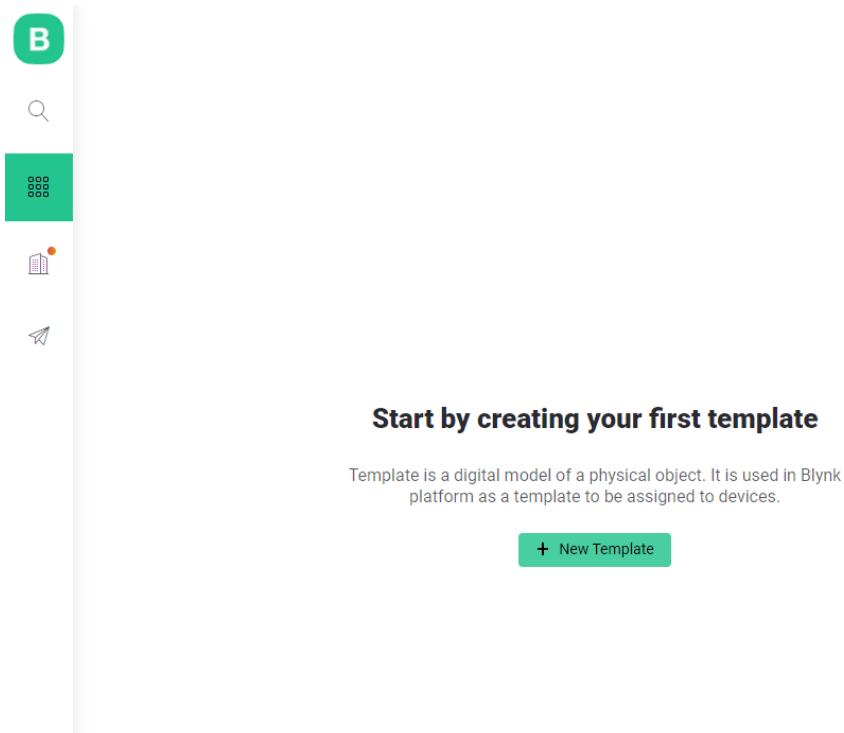


การสร้าง Project

1. ให้กดเลือกที่หัวข้อ Template



2. ทำการสร้าง Template



3. ตั้งชื่อ Template และตั้งค่าอุปกรณ์

Create New Template

NAME ตั้งชื่อ Template

HARDWARE	ประเภทอุปกรณ์
ESP8266	ให้เลือก ESP8266
CONNECTION TYPE	ประเภทอุปกรณ์
WiFi	ให้เลือก WiFi

DESCRIPTION
 This is my template
 19 / 128

Cancel Done

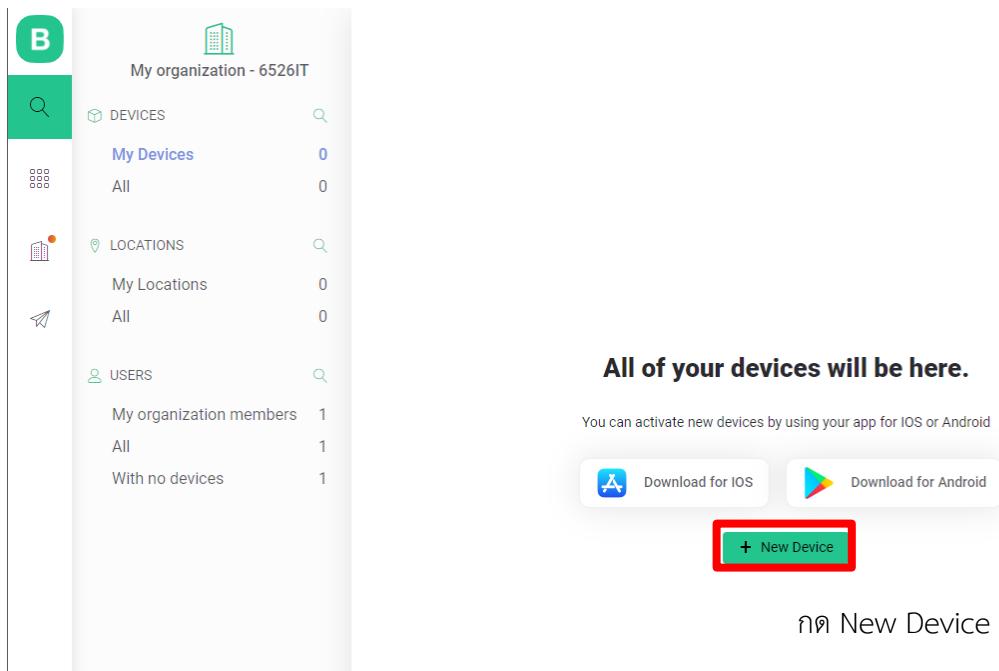
4. กด save

Test Blynk [Delete](#) [Cancel](#) Save

[Info](#) [Metadata](#) [Datastreams](#) [Events](#) [Web Dashboard](#) [Mobile Dashboard](#) ?

TEMPLATE NAME	TEMPLATE IMAGE (OPTIONAL)
<input type="text" value="Test Blynk"/>	<input type="button" value="Add image"/> Upload from computer or drag-n-drop png or .jpg, minimum width 500px
HARDWARE	CONNECTION TYPE
ESP8266	WIFI
DESCRIPTION	
<input type="text" value="This is my template"/>	
19 / 128	
FIRMWARE CONFIGURATION <pre>#define BLYNK_TEMPLATE_ID "TMPL_dwSnkr2" #define BLYNK_DEVICE_NAME "Test Blynk"</pre> <p>Template ID and Device Name should be included at the top of your main firmware</p>	
TEMPLATE ID	MANUFACTURER
<input type="text" value="TMPL_dwSnkr2"/>	<input type="text" value="My organization 6526IT"/>
CATEGORIES ?	
<input type="text" value="Other x"/>	
OFFLINE IGNORE PERIOD	
<input type="text" value="00 hrs 00 mins 00 secs"/>	
HOTSPOT PREFIX	
<input type="text" value="Hotspot Prefix"/>	
<input checked="" type="checkbox"/> Show map in device view UPGRADE	

5. กลับไปที่ Tab Search เพื่อเพิ่ม Device

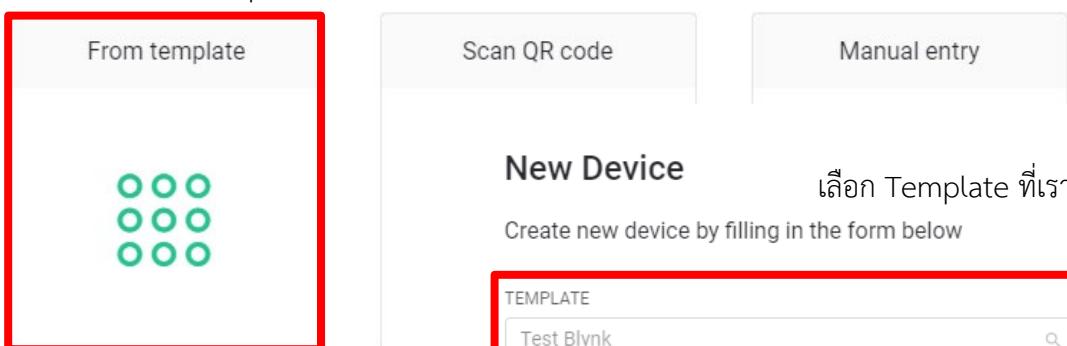


กด New Device

New Device

Choose a way to create new device

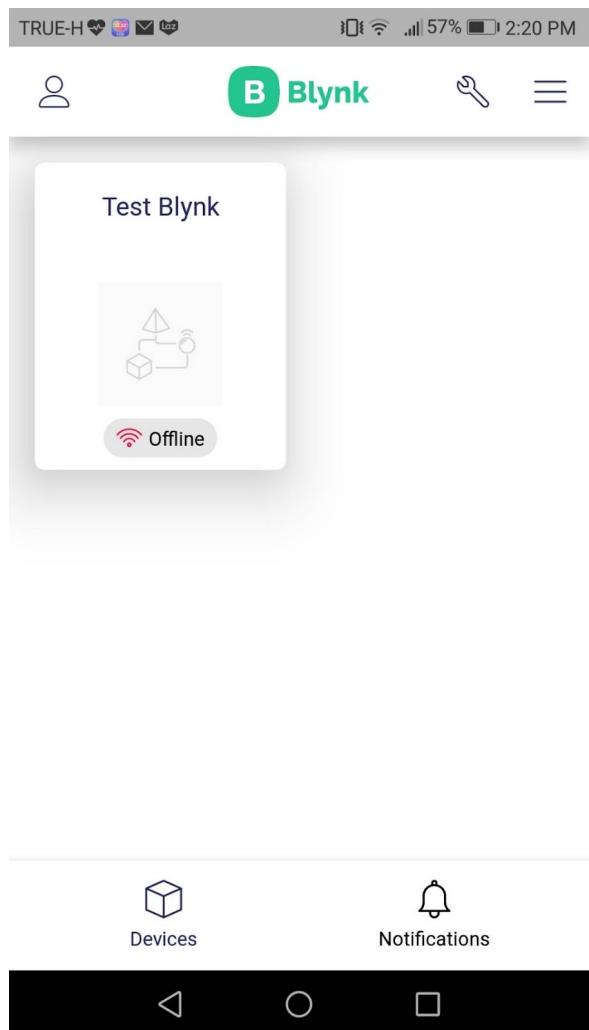
เลือก From template



Point on the cards to see instructions

กด Create เพื่อสร้าง

6. ในส่วนของ Mobile Application หากเราได้ Device แล้วหน้าจะจะปรากฏตามดังรูป



7. กลับมาที่ฝั่งคอมพิวเตอร์ ให้เลือก Device ที่เราสร้างขึ้นมาแล้วกดไปที่ Device Info

ให้ copy ส่วนนี้มาเขียนลง
ใน Arduino IDE

STATUS	LAST UPDATED
● Offline	Not updated yet
DEVICE ACTIVATED	ORGANIZATION
1:59 PM Today by pasawit55@hotmail.com	My organization - 6526IT
TOTAL ONLINE TIME	TEMPLATE NAME
Wasn't online yet	Test Blynk
AUTHTOKEN	
nSUY - **** - **** - ****	
MANUFACTURER	
My organization 6526IT	

Code ภายใน Arduino IDE

```

sketch_dec12a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_dec12a

#include <ESP8266WiFi.h>
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL_dwSnkr2"
#define BLYNK_DEVICE_NAME "Test Blynk"
#define BLYNK_AUTH_TOKEN "nSUY9Hkn70MFsLdrztNOqRyU_Y0-Qx89"
#include <BlynkSimpleEsp8266.h>
#define LED1 D5
#define LED2 D6
#define LED3 D7

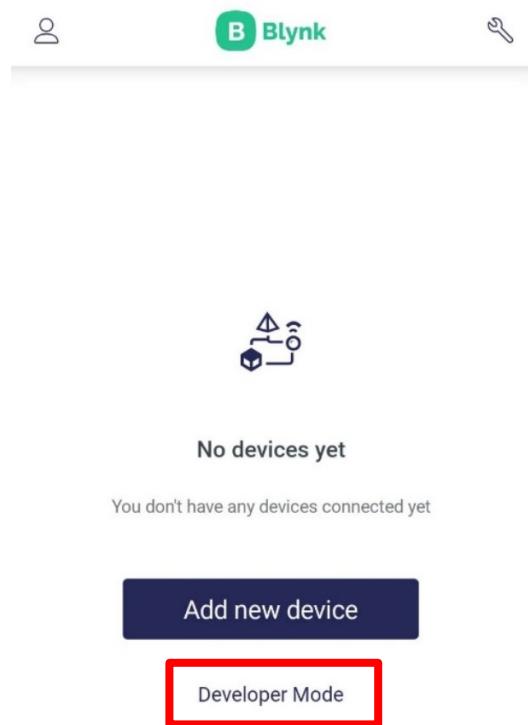
//char ssid[] = "";
//char pass[] = "";
char auth[] = BLYNK_AUTH_TOKEN;

void setup()

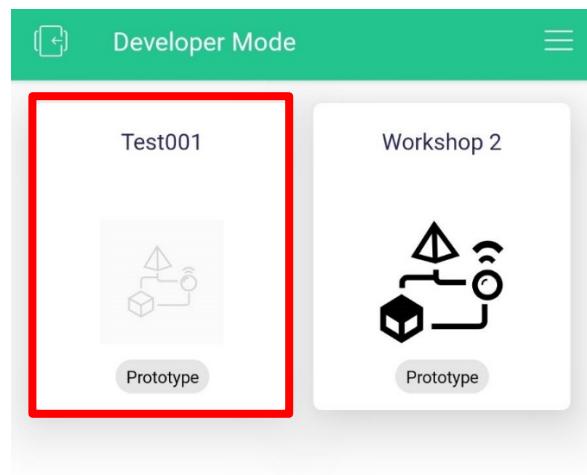
```

การจัดหน้า Dashboard บน Mobile Application

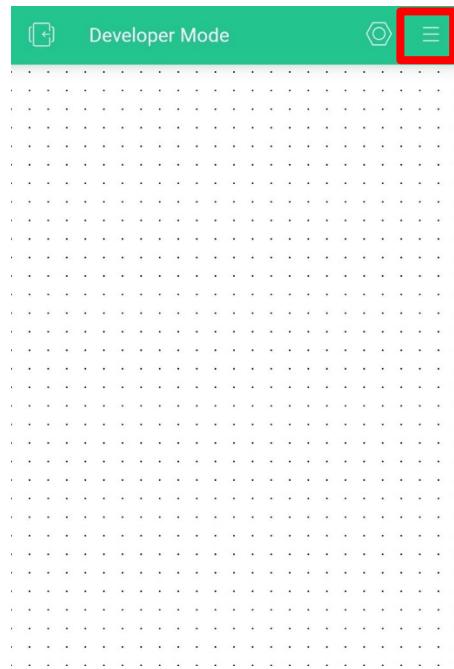
- กดเข้า Developer Mode เพื่อเลือก template



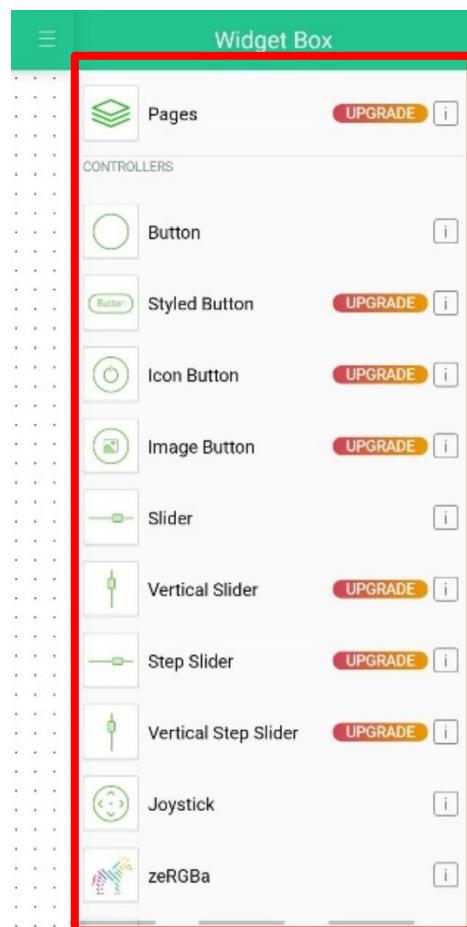
- เลือก Template ที่เราต้องการ



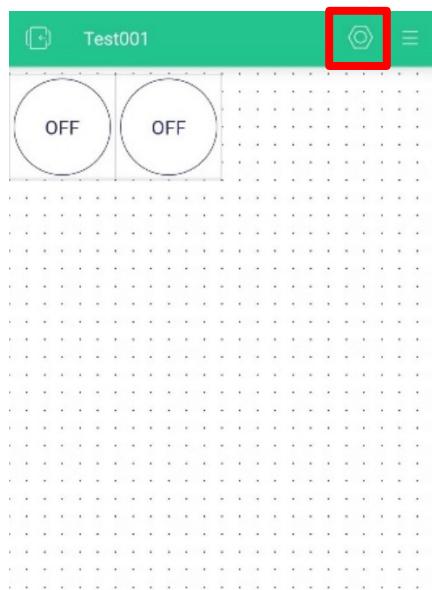
3. กดเพื่อเลือก widget ที่ต้องการในการออกแบบหน้า Dashboard



4. เลือก widget ตามที่ต้องการ



5. กดเพื่อออกจาก Developer Mode



6. หน้า dashboard พร้อมใช้งาน



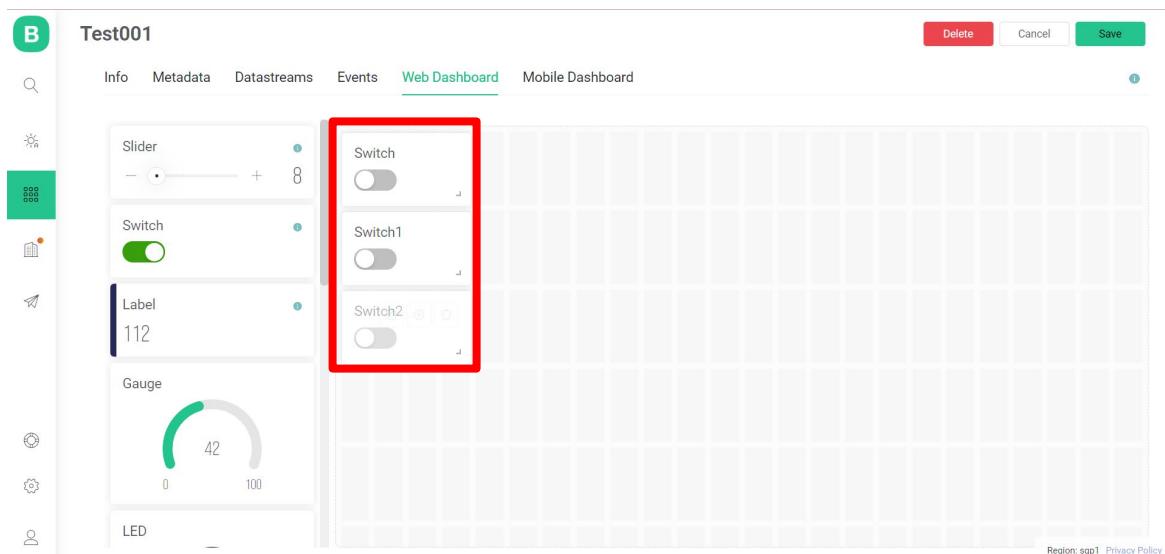
Workshop 10 การใช้งาน Blynk ควบคุมการเปิด - ปิดไฟ LED

อุปกรณ์ที่ต้องใช้

1. NodeMCU ESP8266
2. หลอดไฟ LED สีแดง ,สีเหลือง, สีเขียว
3. ตัวต้านทาน 220 โอห์ม

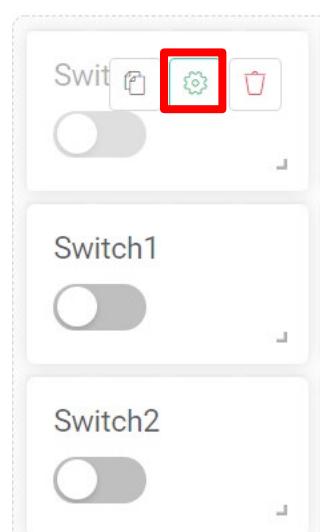
การตั้งค่า Dashboard

1. เลือก switch มาใส่ในหน้า dashboard 3 switch

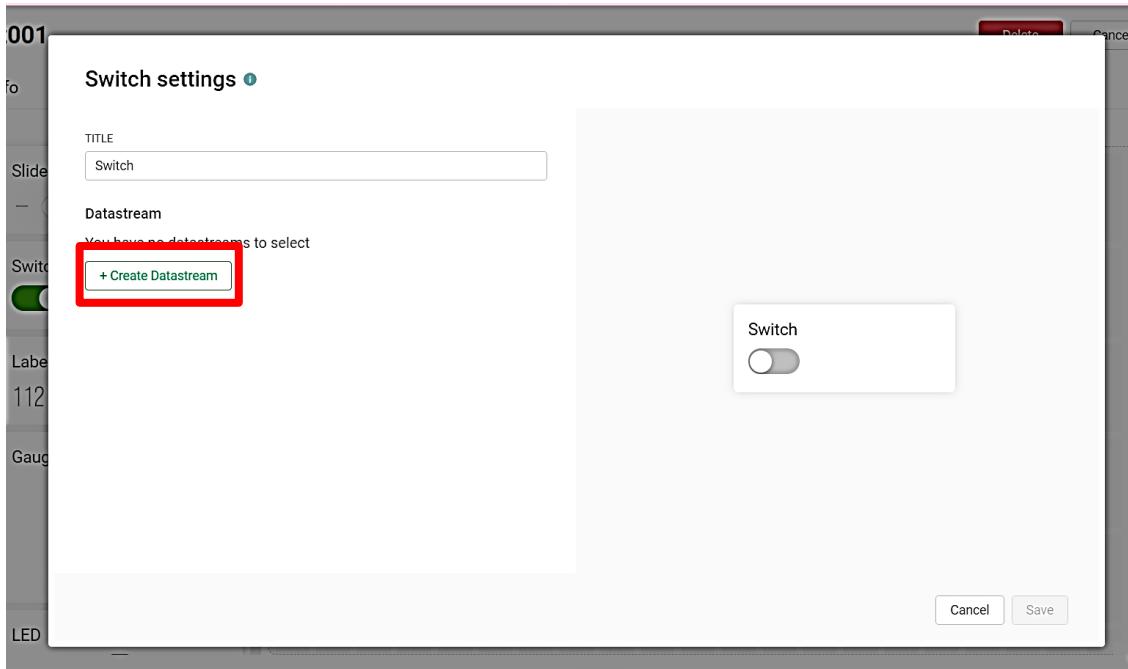


2. ตั้งค่า switch

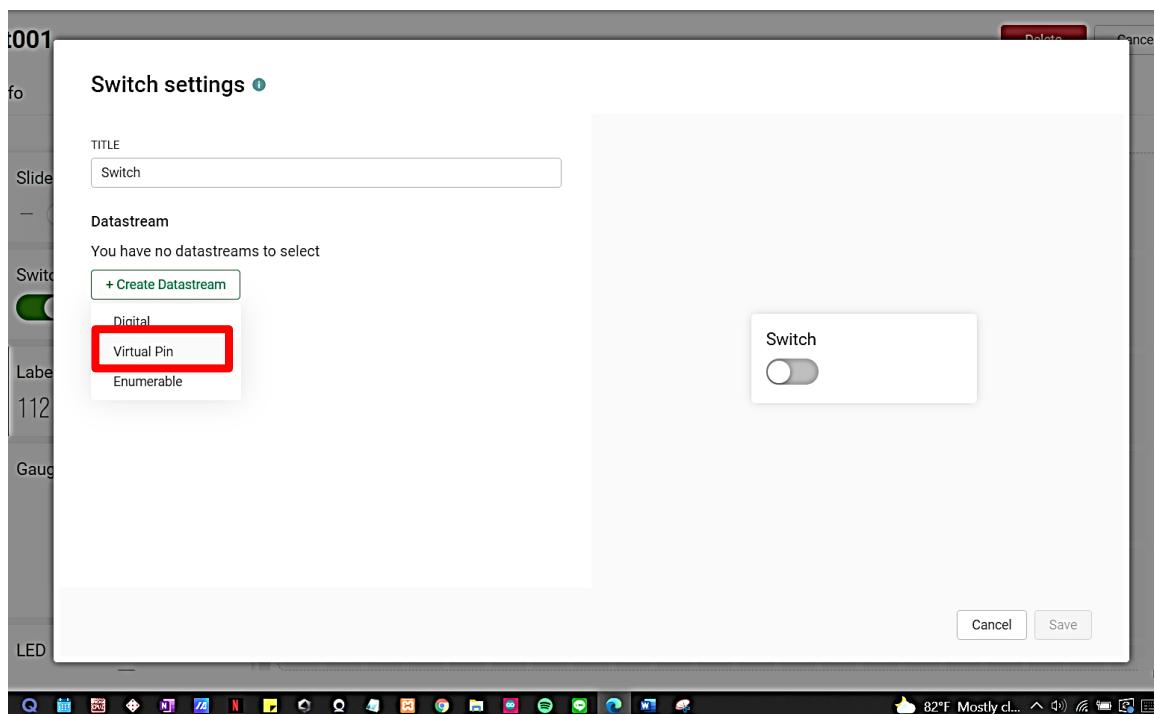
2.1 กดเพื่อตั้งค่า Switch



2.2 กดเพื่อตั้งค่าการรับส่งข้อมูล



2.3 กดเลือก virtual pin



2.4 ให้ตั้งค่า PIN เป็น V0 และกด Create

Datastream

Virtual Pin Datastream

NAME	ALIAS
 Switch	Switch 
PIN	DATA TYPE
V0 	Double 
UNITS	
None 	
<input type="button" value="Cancel"/> <input type="button" value="Create"/>	

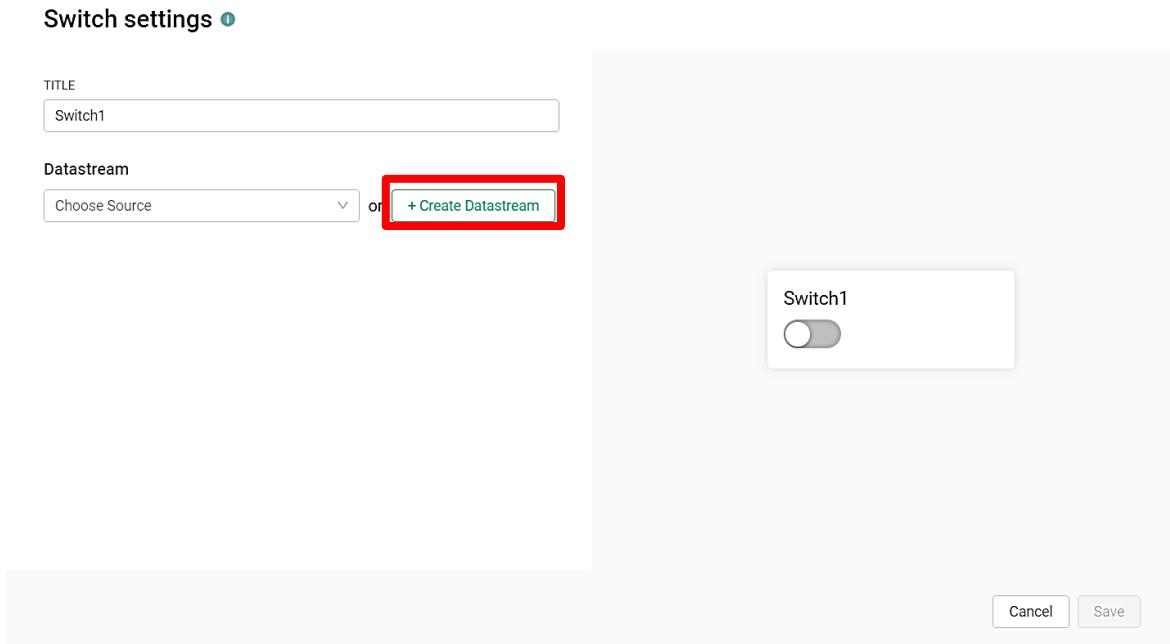
2.5 กด save เพื่อบันทึกการตั้งค่า switch

Switch settings ⓘ

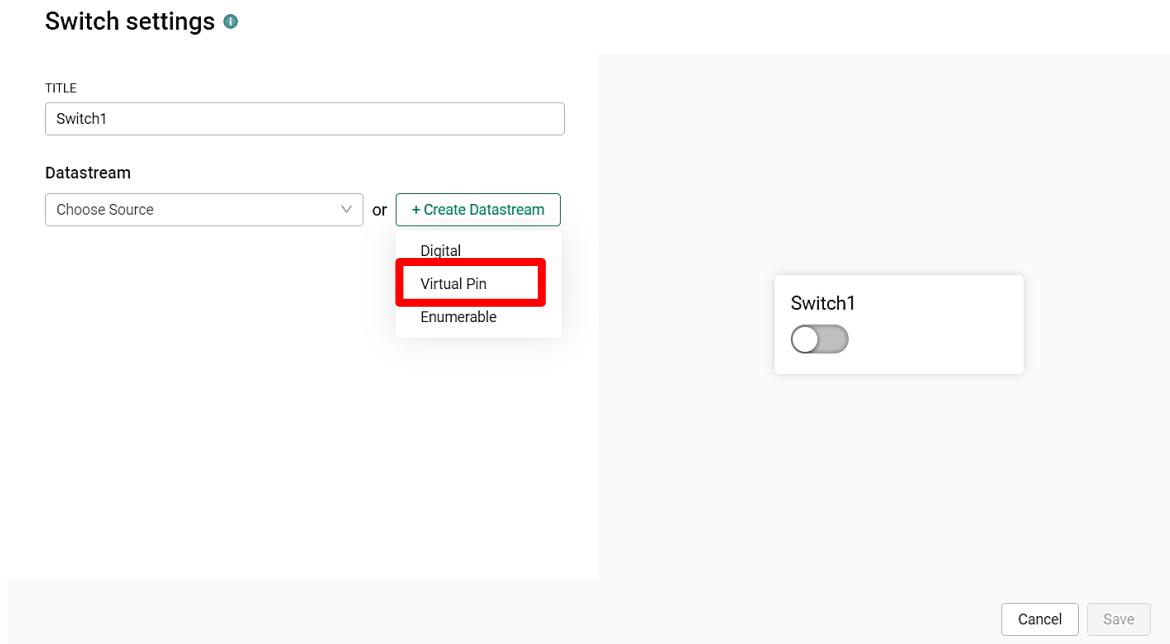
TITLE	Switch
Datastream	
Switch (V0)	
ON VALUE	OFF VALUE
1	0 
<input checked="" type="checkbox"/> Show on/off labels <input checked="" type="checkbox"/> Hide widget name	
	
<input type="button" value="Cancel"/> <input style="background-color: green; color: white; border: 2px solid red; border-radius: 5px; padding: 2px 10px; margin-left: 10px;" type="button" value="Save"/>	

3 ตั้งค่า switch1

3.1 กด create datasteam เพื่อตั้งค่า switch1



3.2 กดเลือก virtual pin



3.3 ให้ตั้งค่า PIN เป็น V1 และกด Create

Datastream

Virtual Pin Datastream

NAME	ALIAS
 Switch1	Switch1 
PIN	DATA TYPE
V1 	Double 
UNITS	
None 	
<input type="button" value="Cancel"/> <input style="border: 2px solid red; background-color: red; color: white; padding: 2px 10px;" type="button" value="Create"/>	

3.4 กด save เพื่อบันทึกการตั้งค่า switch

Switch settings ⓘ

TITLE	Switch1
Datastream	
Switch1 (V1) 	
ON VALUE	OFF VALUE
1	0 
<input checked="" type="checkbox"/> Show on/off labels <input checked="" type="checkbox"/> Hide widget name	
	
<input type="button" value="Cancel"/> <input style="border: 2px solid red; background-color: red; color: white; padding: 2px 10px;" type="button" value="Save"/>	

4 ตั้งค่า switch2

4.1 กด create datasteam เพื่อตั้งค่า switch2

Switch settings ⓘ

TITLE

Switch2

Datastream

Choose Source

or

+ Create Datastream

Switch2



Cancel

Save

4.2 เลือก Virtual Pin

Switch settings ⓘ

TITLE

Switch2

Datastream

Choose Source

or

+ Create Datastream

Digital

Virtual Pin

Enumerable

Switch2



Cancel

Save

4.3 ให้ตั้งค่า PIN เป็น V2 และกด Create

Datastream

Virtual Pin Datastream

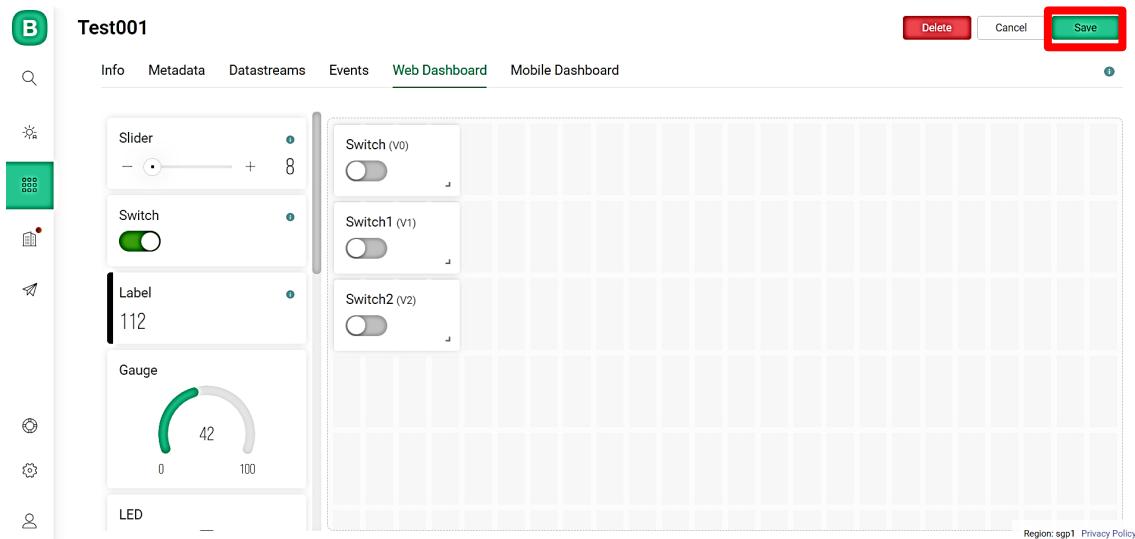
NAME	ALIAS
 Switch2	Switch2 
PIN	DATA TYPE
V2 	Double 
UNITS	
None 	
<input type="button" value="Cancel"/> <input style="border: 2px solid red; background-color: #008000; color: white; padding: 2px 10px; border-radius: 5px;" type="button" value="Create"/>	

4.4 กด save เพื่อบันทึกการตั้งค่า switch

Switch settings ⓘ

TITLE	Switch2
Datastream	
Switch2 (V2) 	
ON VALUE	OFF VALUE
1	0 
<input checked="" type="checkbox"/> Show on/off labels <input checked="" type="checkbox"/> Hide widget name	
	
<input type="button" value="Cancel"/> <input style="border: 2px solid red; background-color: #008000; color: white; padding: 2px 10px; border-radius: 5px;" type="button" value="Save"/>	

4.5 กด save เพื่อบันทึกการตั้งค่า dashboard



4.6 เลือก update 1 active device และกด continue

Apply Changes?

There is 1 active device associated with
Test001 template

How to apply changes?

Update 1 active device

Save Changes. Don't update active device

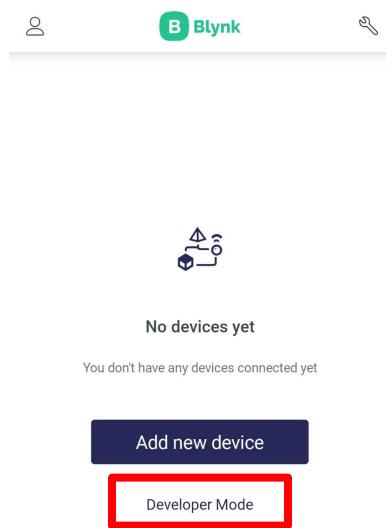
Create a clone of this Template with updated Metadata

Continue

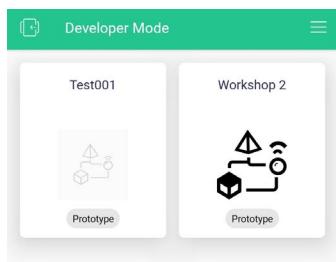
Cancel

การตั้งค่า Dashboard บนโทรศัพท์

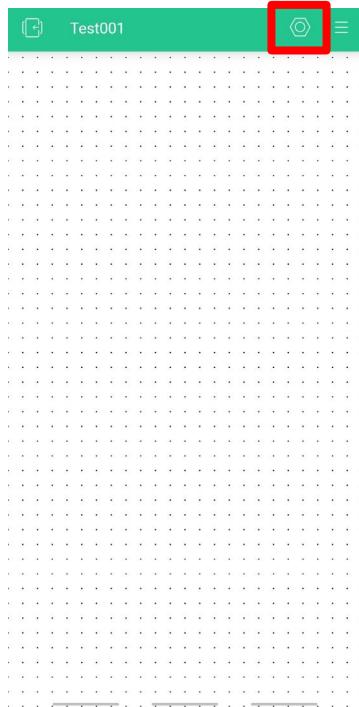
- เข้าไปที่ Developer Mode



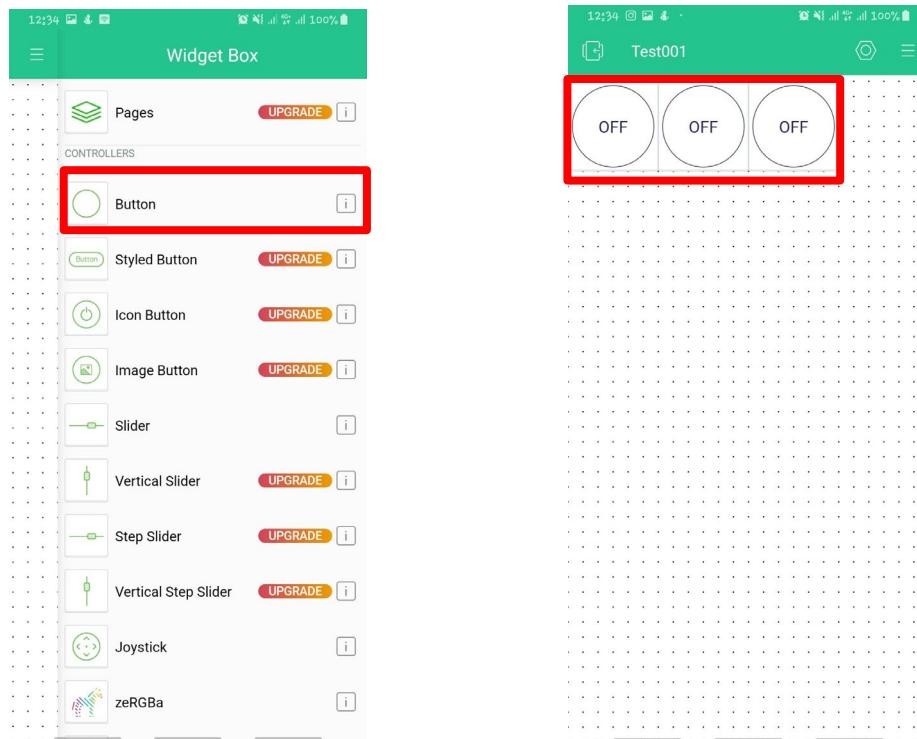
- เลือก Prototype ของตนเอง



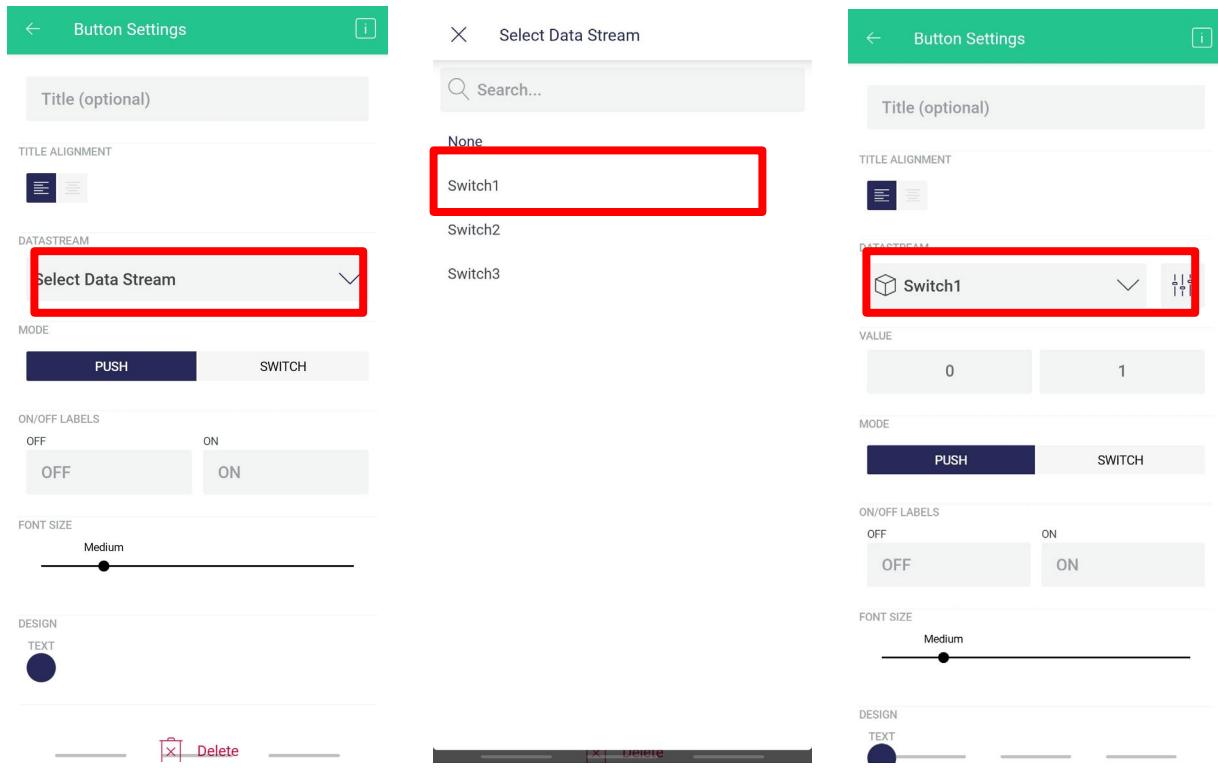
3. กดไปที่ปุ่มดังรูป



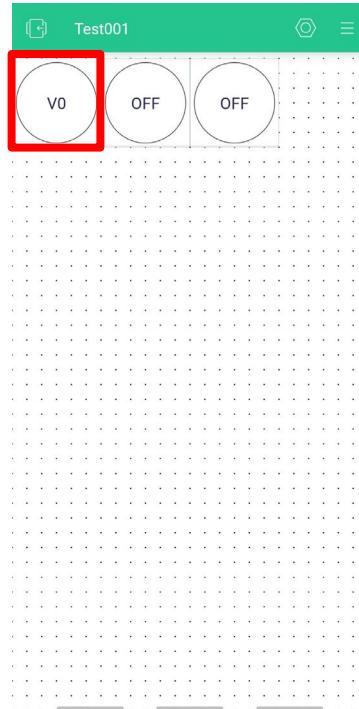
4. เลือกไปที่ Button โดยกดเลือกมาทั้งหมด 3 Button



5. กดไปที่ Button ตัวที่ 1 เพื่อทำการตั้งค่า และกดเลือก data stream ที่ต้องการ

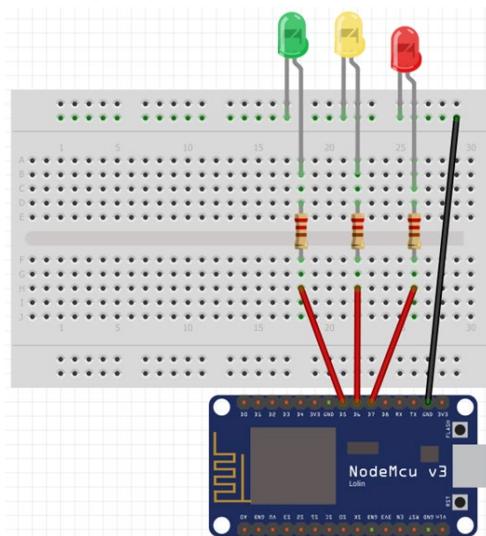


6. จะได้ผลลัพธ์ดังรูป จากนั้นนำมาขึ้นตอนที่ 5 อีก 2 Button



การต่อวงจร

PIN	อุปกรณ์
D5	LED สีเขียว
D6	LED สีเหลือง
D7	LED สีแดง



การเขียน code (กรอบสีแดงคือต้องเปลี่ยนเป็นข้อมูลของตัวเอง)

Workshop_10

```
#define BLYNK_TEMPLATE_ID "your template id"
#define BLYNK_DEVICE_NAME "your device name"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define LED1 D5
#define LED2 D6
#define LED3 D7

const char ssid[] = "Wifi_name";
const char pass[] = "password";
const char auth[] = "your token";

BlynkTimer timer;
void timerEvent();

void setup()
{
    //ทำการเชื่อมต่อไฟท์ Blynk
    Serial.begin(115200);

    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, timerEvent);

    //กำหนด mode ให้กับ pin
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
}

void loop()
{
    //เริ่มการทำงานของ Blynk
    Blynk.run();
    timer.run();
}
```

```
BLYNK_WRITE(V0) { //function การทำงานของ visualpin 0
    if (param.asInt()) { //param.asInt() ?ใช้ในการเช็คค่ามีการกดบุ๊มหรือไม่ ถ้ามีจะ return เมิน true
        digitalWrite(LED1, HIGH);
    }
    else {
        digitalWrite(LED1, LOW);
    }
}

BLYNK_WRITE(V1) { //function การทำงานของ visualpin 1
    if (param.asInt()) {
        digitalWrite(LED2, HIGH);
    }
    else {
        digitalWrite(LED2, LOW);
    }
}

BLYNK_WRITE(V2) { //function การทำงานของ visualpin 2
    if (param.asInt()) {
        digitalWrite(LED3, HIGH);
    }
    else {
        digitalWrite(LED3, LOW);
    }
}

void timerEvent() {
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_10/Workshop_10.ino

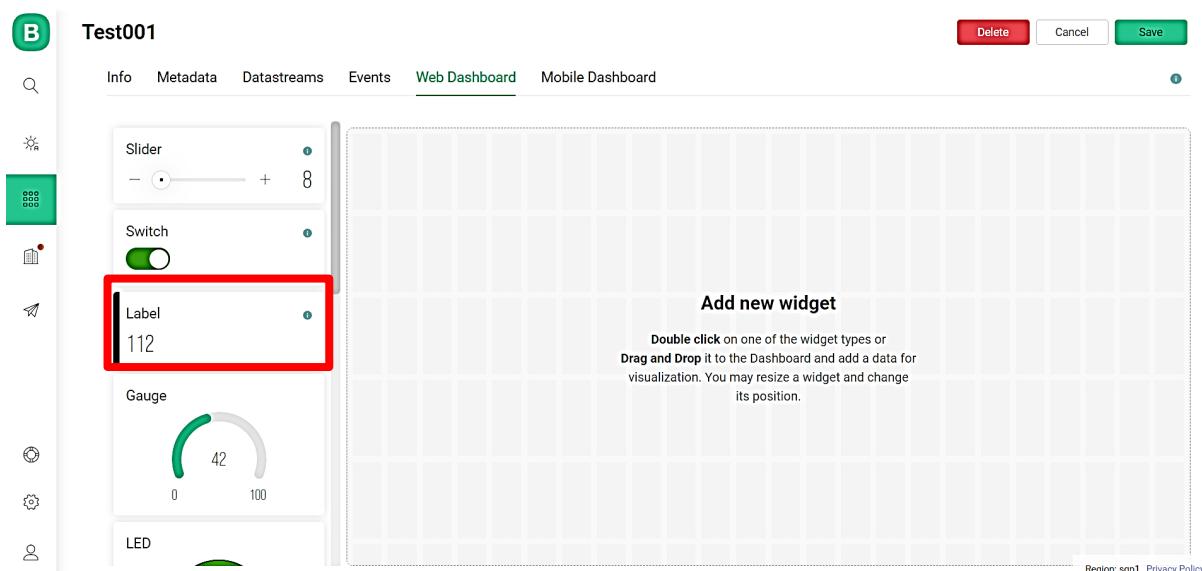
Workshop 11 Blynk กับ HC-SR04

อุปกรณ์ที่ต้องใช้

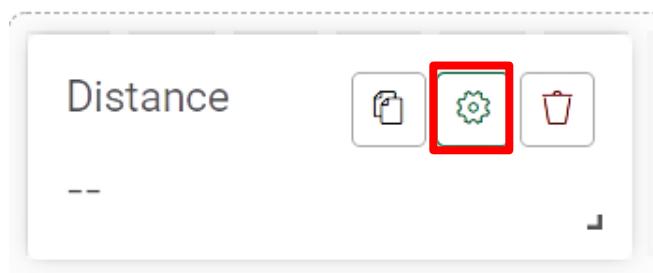
1. NodeMCU ESP8266
2. โมดูลอัลตราโซนิกเซนเซอร์
3. LED สีเขียวและแดง
4. ตัวต้านทานขนาด 220 โอห์ม 2 ตัว

การตั้งค่า Dashboard

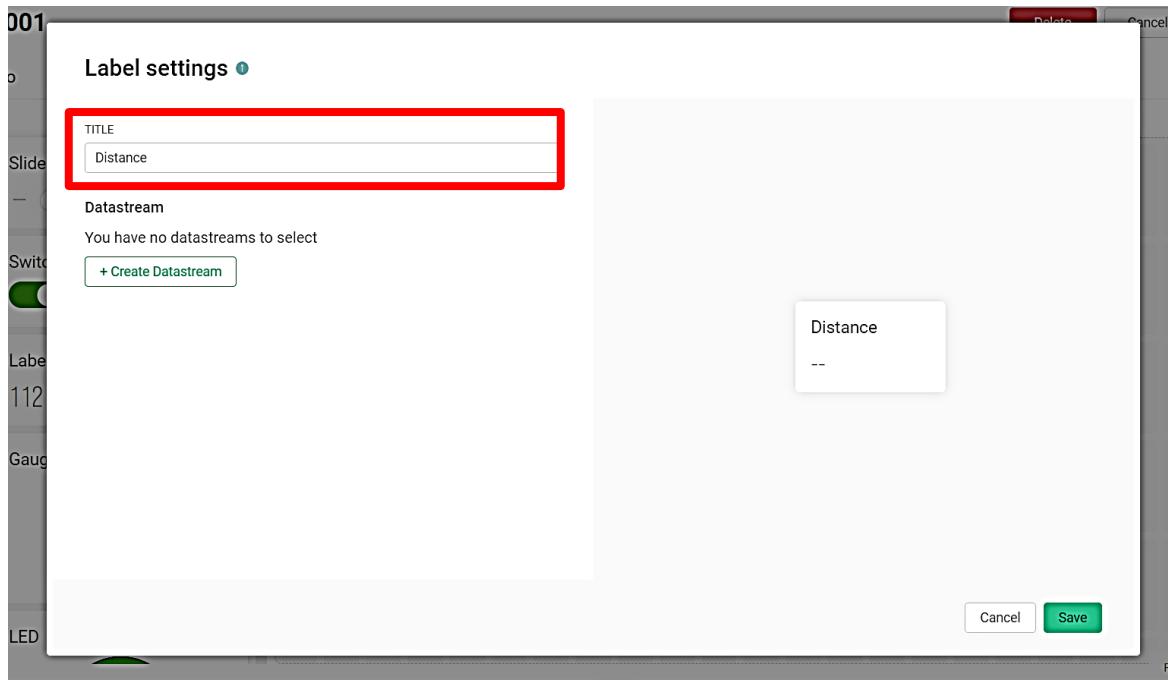
1 เลือก widget Label นำมาใส่ใน Dashboard



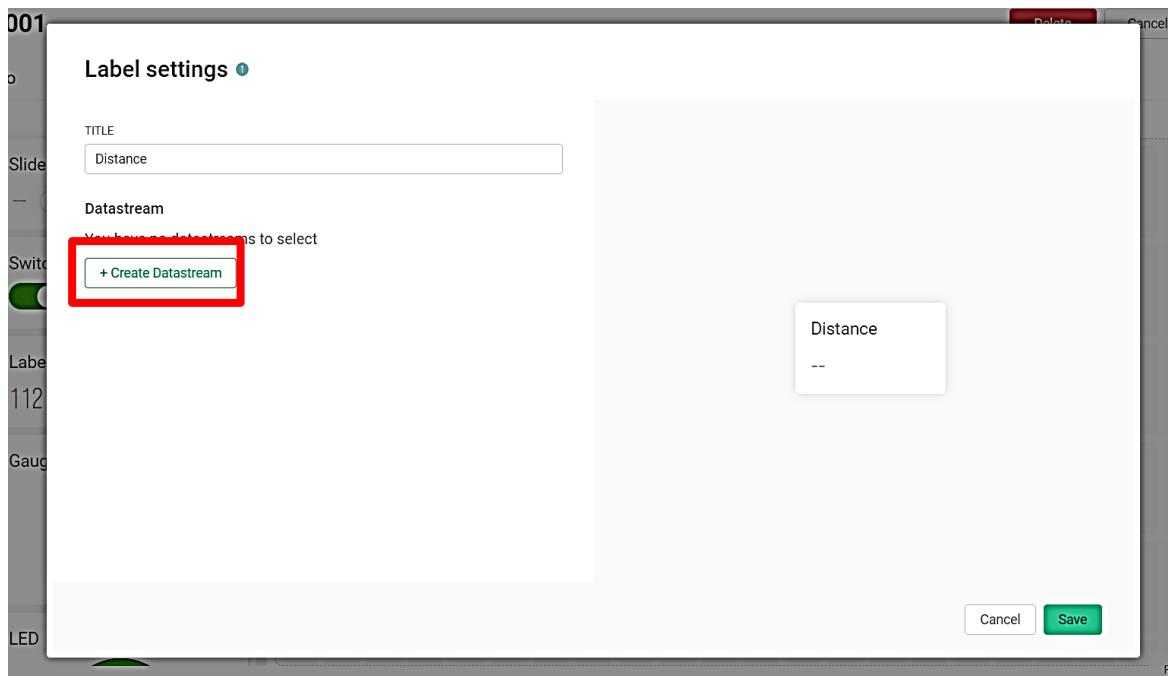
2 กดเพื่อตั้งค่าการรับส่งข้อมูล



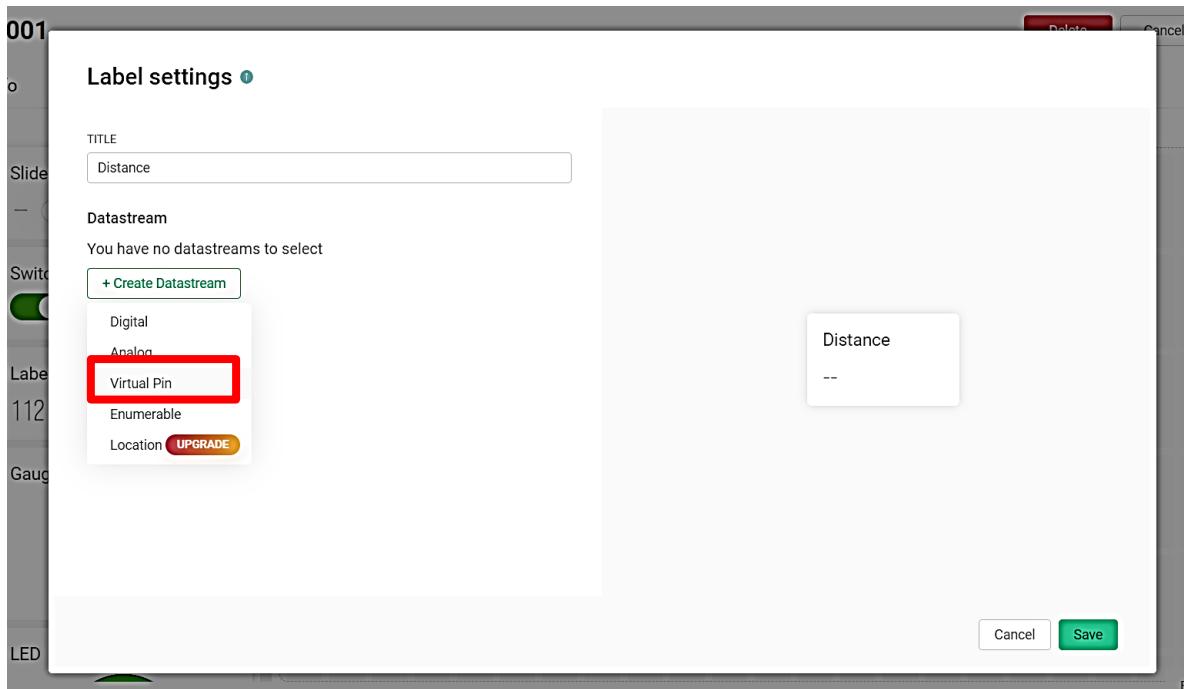
3 ตั้งชื่อ Label ว่า Distance



4 กดเพื่อสร้าง Datastream



5 กดเลือกใช้งาน Virtual Pin



6 ให้ตั้งค่า PIN เป็น V0 , ตั้งค่า Units เป็น Centimeter , ตั้งค่า max เป็น 1,000 และกด Create

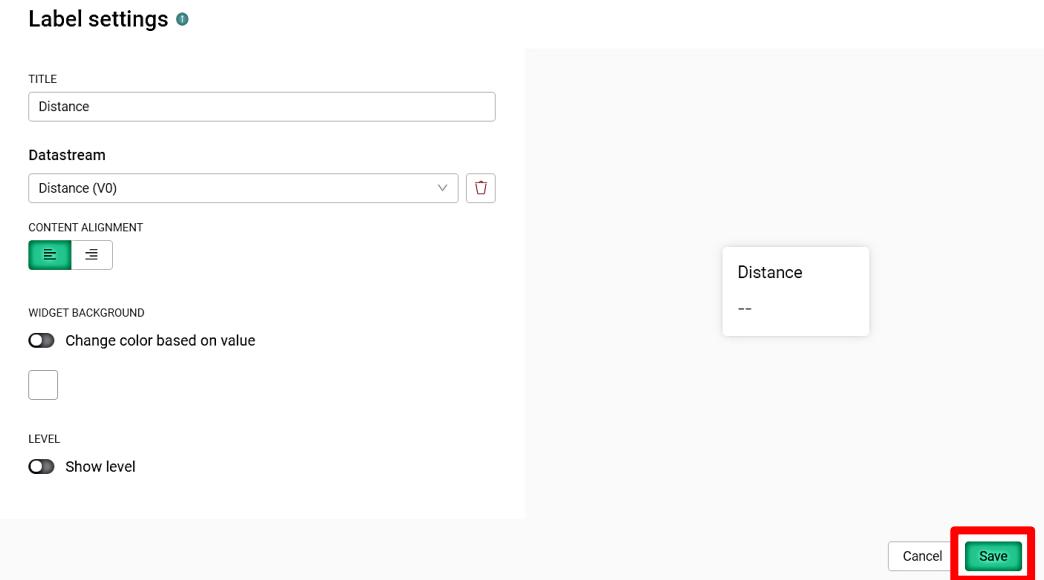
Datastream

Virtual Pin Datastream

V0	Double		
UNITS			
Centimeter			
MIN	MAX	DECIMALS	DEFAULT VALUE
0	1000	#.##	0

Create

7 กด save เพื่อบันทึก Label นี้



8 กด save เพื่อบันทึก dashboard

Test001

Info Metadata Datastreams Events Web Dashboard Mobile Dashboard

Slider
8

Switch

Label
112

Gauge
42

LED

Distance (v0)
0

Cancel **Save**

Region: sgp1 Privacy Policy

9 เลือก update 1 active device และกด continue

Apply Changes?

There is 1 active device associated with
Test001 template

How to apply changes?

Update 1 active device

Save Changes. Don't update active device

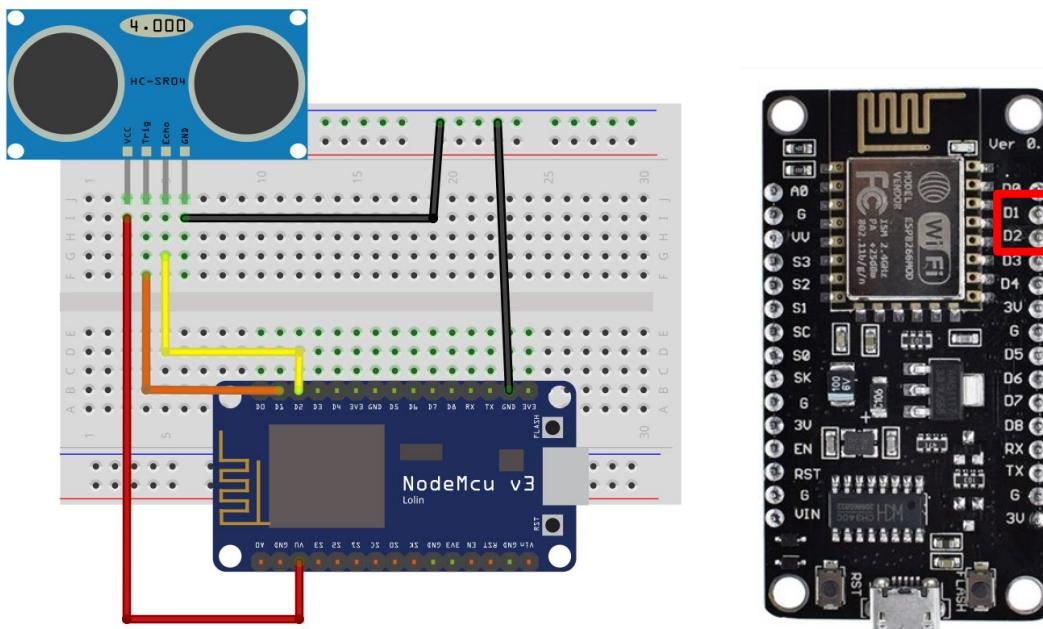
Create a clone of this Template with updated Metadata

Continue

Cancel

การต่อวงจร

PIN	อุปกรณ์
D1	Ultrasonic (Trig)
D2	Ultrasonic (Echo)



การเขียน code (กรอบสีแดงคือต้องเปลี่ยนเป็นข้อมูลของตัวเอง)

Workshop_11

```
#define BLYNK_TEMPLATE_ID "your template id"
#define BLYNK_DEVICE_NAME "your device name"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

//กำหนดความเร้าของเสียงให้อยู่ในหน่วย cm/uS
#define SOUND_VELOCITY 0.034
#define TRIG_PIN D1
#define ECHO_PIN D2

const char ssid[] = "Wifi_name";
const char pass[] = "password";
const char auth[] = "your token";

long duration;
float distanceCm;

BlynkTimer timer;
void timerEvent();
void pushDistance();

void setup() {
    Serial.begin(115200); // Starts the serial communication

    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, timerEvent);

    pinMode(TRIG_PIN, OUTPUT); // Sets the trigPin as an Output
    pinMode(ECHO_PIN, INPUT); // Sets the echoPin as an Input
    digitalWrite(TRIG_PIN, LOW); // Clears the trigPin
}

void loop() {
    // ตั้งค่า trigPin ให้เป็น High เมื่อเวลา 10 micro seconds
    Blynk.run();
    timer.run();
}
```

```

void timerEvent() {
    // แสดงระยะทาง
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    // อ่านค่าจาก echoPin ซึ่งจะคืนค่าเมื่อเวลา sound wave travel (หน่วยเวลาเป็น micro seconds)
    duration = pulseIn(ECHO_PIN, HIGH);

    // ค่าแทนระยะทาง
    distanceCm = duration * SOUND_VELOCITY / 2;
    if (distanceCm >= 200 || distanceCm <= 0) {
        Serial.println("Out of range");
    }
    else {
        Blynk.virtualWrite(v0, distanceCm); // ส่งค่าไปที่ Blynk โดยใช้ visualpin v0
        Serial.print("Distance (cm): ");
        Serial.print(distanceCm);
        Serial.println(" cm");
    }
}

```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_11/Workshop_11.ino

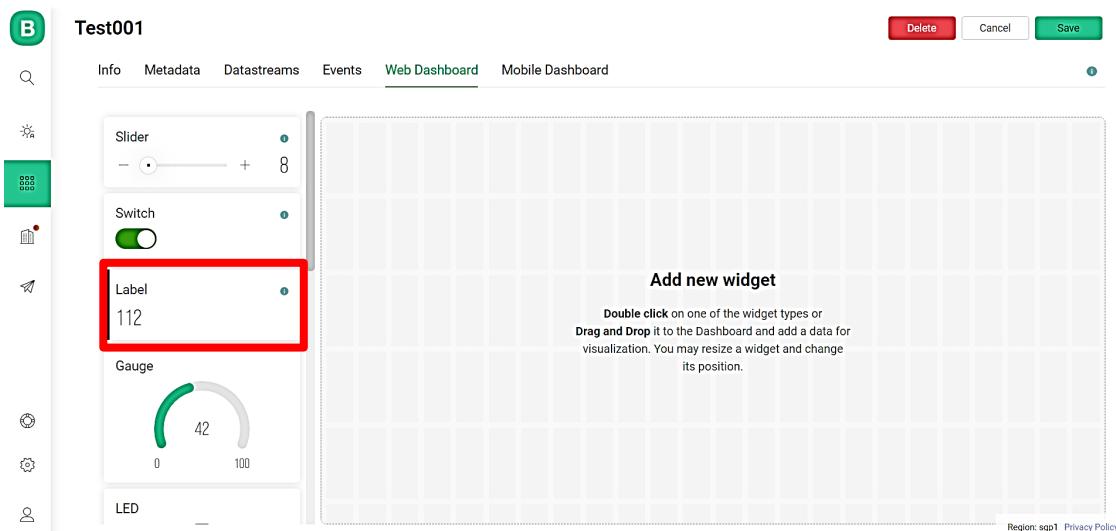
Workshop 12 Blynk and Soil_Moisture_Sensor

อุปกรณ์ที่ต้องใช้

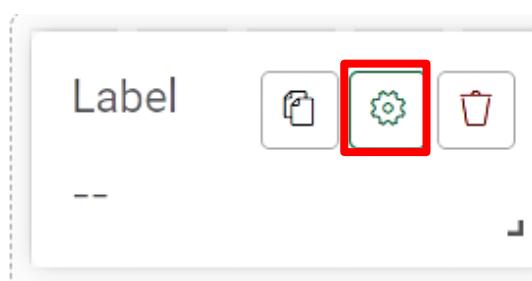
1. NodeMCU ESP8266
2. เช่นเซอร์วัตความชื้นในดิน Soil Moisture Sensor
3. LED สีแดงและสีเขียว

การตั้งค่า Dashboard

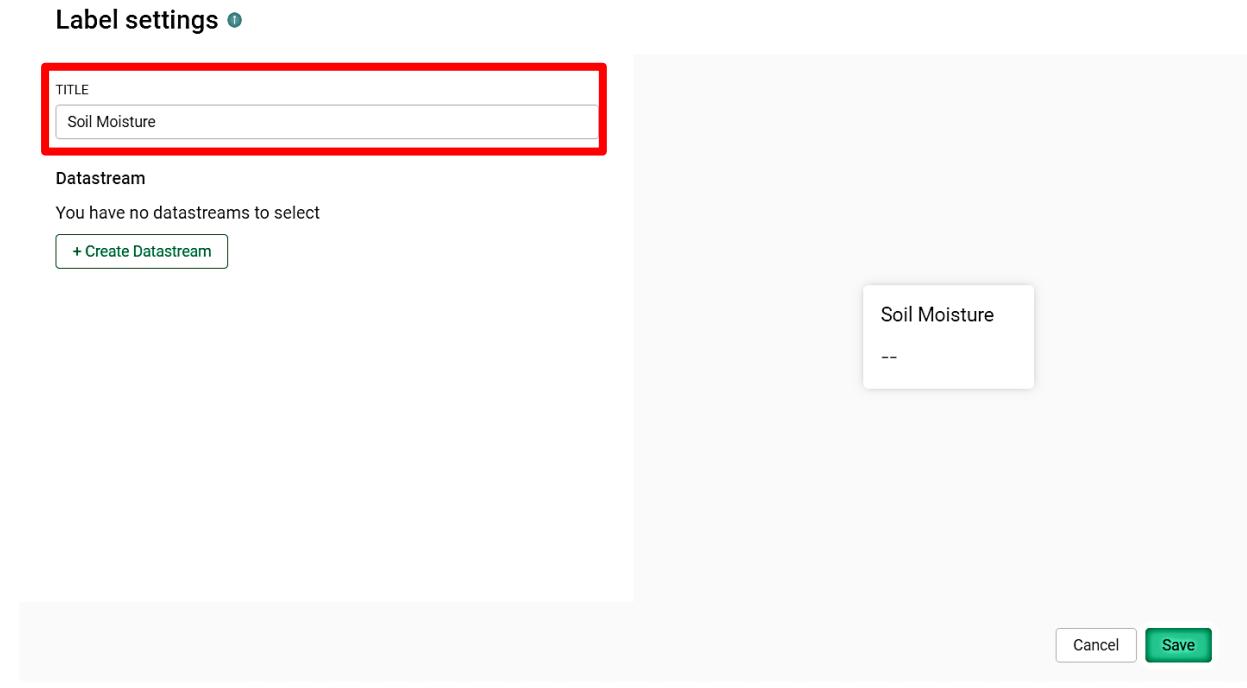
1 เลือก Label widget นำมาระบบที่ Dashboard



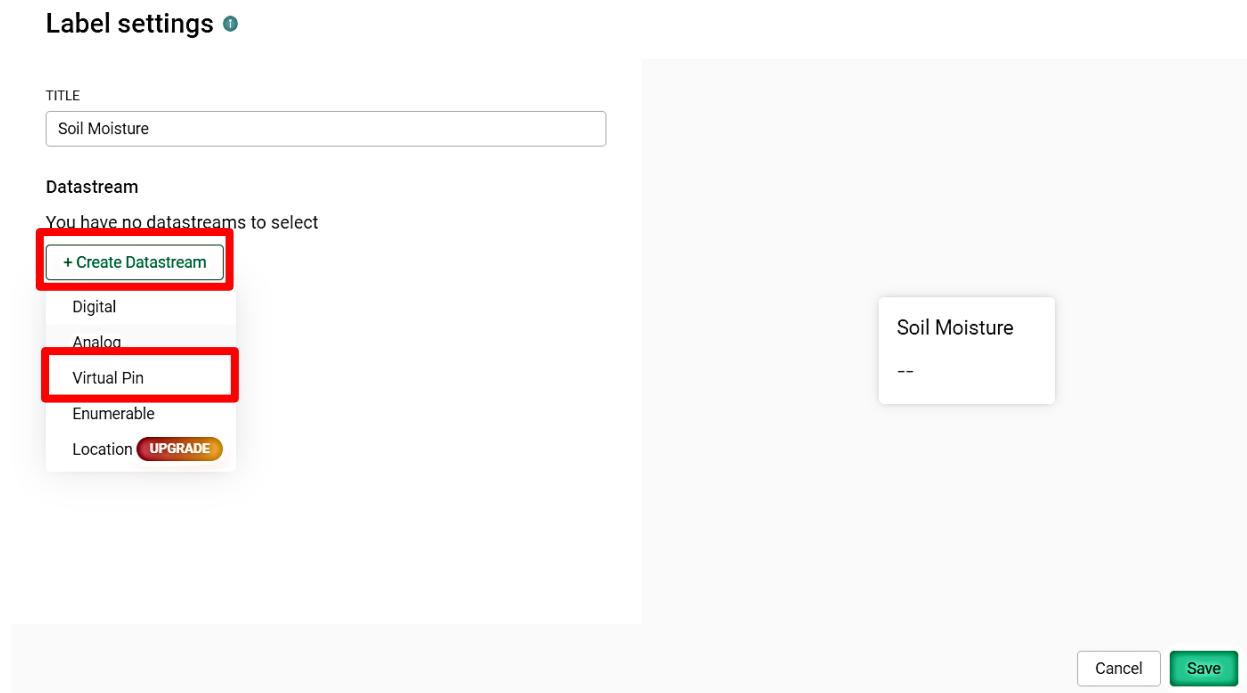
2 กดเพื่อตั้งค่าการรับส่งข้อมูล



3 ตั้งชื่อ Label เป็น Soil Moisture



4 กด Create Datastream และเลือก Virtual Pin



5 ให้ตั้งค่า PIN เป็น V0 , ตั้งค่า Units เป็น Percentage % , ตั้งค่า max เป็น 100 และกด Create

Datastream

Virtual Pin Datastream

V0	Double		
UNITS			
Percentage, %			
MIN	MAX	DECIMALS ⓘ	DEFAULT VALUE
0	100	#.##	0

Create

6 กด save เพื่อบันทึก Label นี้

Label settings ⓘ

TITLE
Soil Moisture

Datastream
Soil Moisture (V0)

CONTENT ALIGNMENT

WIDGET BACKGROUND
Change color based on value

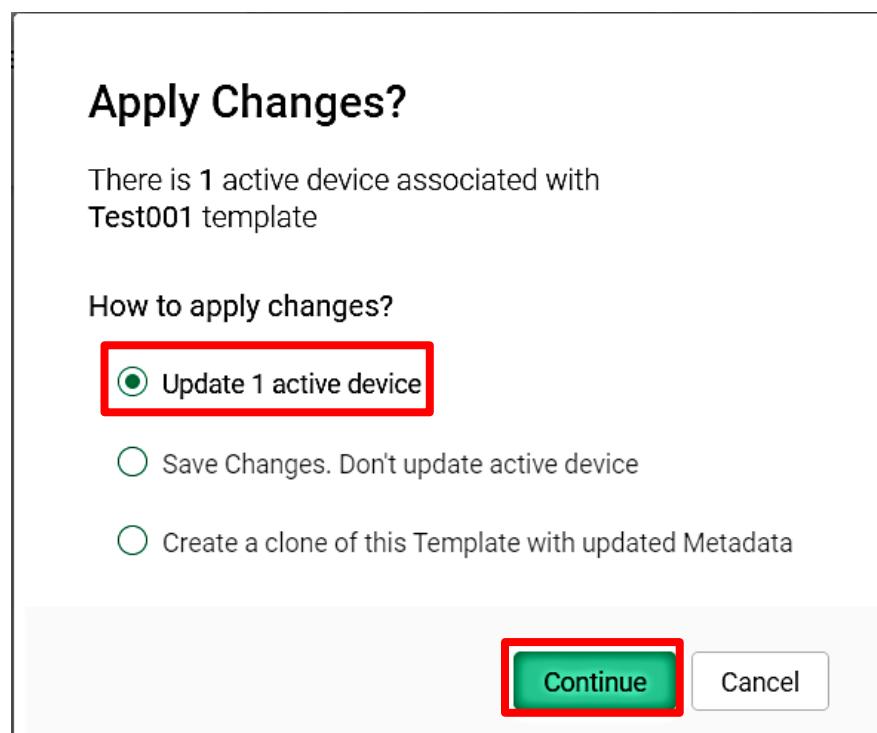
LEVEL
Show level

Soil Moisture
--

Save

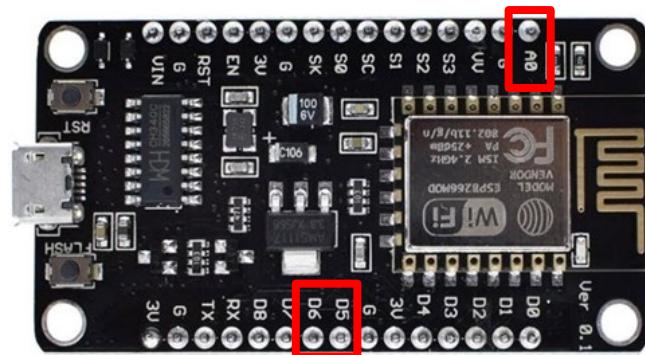
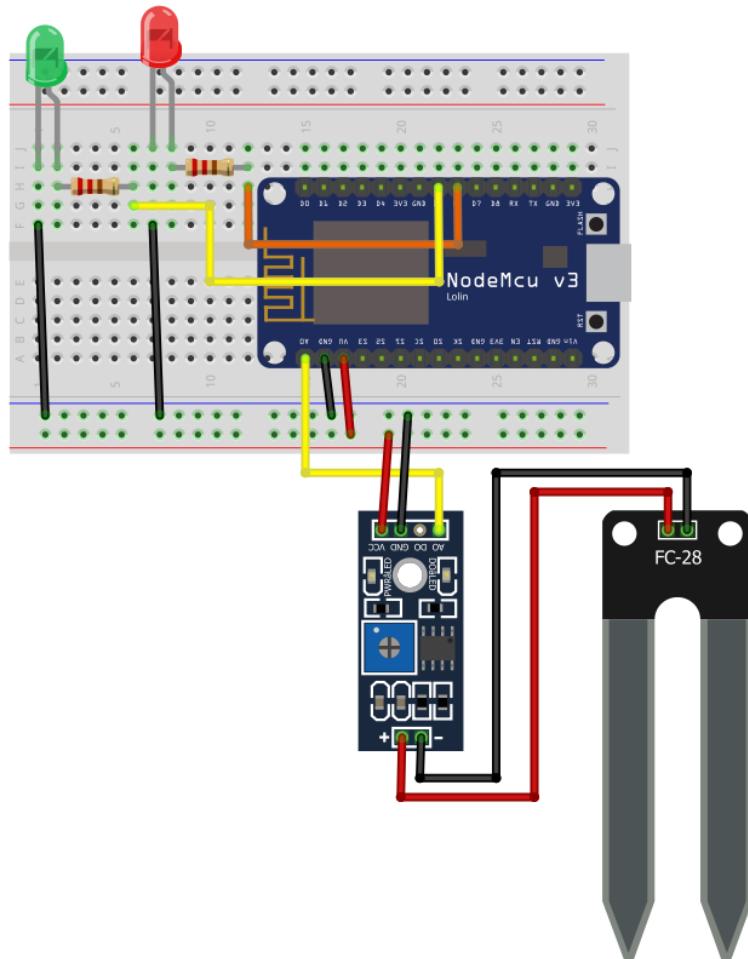
7 กด save เพื่อบันทึกหน้า dashboard

8 เลือก update 1 active device และกด continue



การต่อวงจร

PIN	อุปกรณ์
A0	Soil Moisture Sensor
D5	LED สีเขียว
D6	LED สีแดง



การเขียน code (กรอบสีแดงคือต้องเปลี่ยนเป็นข้อมูลของตัวเอง)

Workshop_12

```
#define BLYNK_TEMPLATE_ID "your template id"
#define BLYNK_DEVICE_NAME "your device name"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#define LED_G D5
#define LED_R D6
#define SOIL_MOIST A0

const char ssid[] = "Wifi_name";
const char pass[] = "password";
const char auth[] = "your token";

int raw_data = 0;
int moisture = 0 ;

BlynkTimer timer;
void timerEvent();
void setup()
{
    Serial.begin(115200);

    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, timerEvent);

    pinMode(LED_G, OUTPUT); // sets the pin as output
    pinMode(LED_R, OUTPUT); // sets the pin as output
}
```

```

void timerEvent() {
    raw_data = analogRead(SOIL_MOIST); //อ่านค่าสัญญาณ analog จาก PIN A0 ที่ต่อกับ Soil Moisture Sensor Module v1
    moisture = map(raw_data, 0, 1023, 100, 0); //ปรับเปลี่ยนค่าจาก 0-1023 เป็น 0-100
    Serial.print("Moisture = "); // พิมพ์ข้อมูลความชื้นเข้าคอมพิวเตอร์ "val = "
    Serial.println(moisture); // พิมพ์ค่าของตัวแปร val

    Blynk.virtualWrite(V0, moisture); //ส่งค่าไปที่ blynk ตามบล็อก visualpin 0

    if (moisture > 50) { //หากค่าที่ทำการแปลง map_val มีค่ามากกว่า 50 แสดงว่าดินมีความชื้นเกิน 50 เมอร์เซ่น
        digitalWrite(LED_G, LOW); // สั่งให้ LED เขียวสาม
        digitalWrite(LED_R, HIGH); // สั่งให้ LED สีแดง ติดสว่าง
    }
    else {
        digitalWrite(LED_G, HIGH); // สั่งให้ LED เขียวติดสว่าง
        digitalWrite(LED_R, LOW); // สั่งให้ LED สีแดง ตบ
    }
}

```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_12/Workshop_12.ino

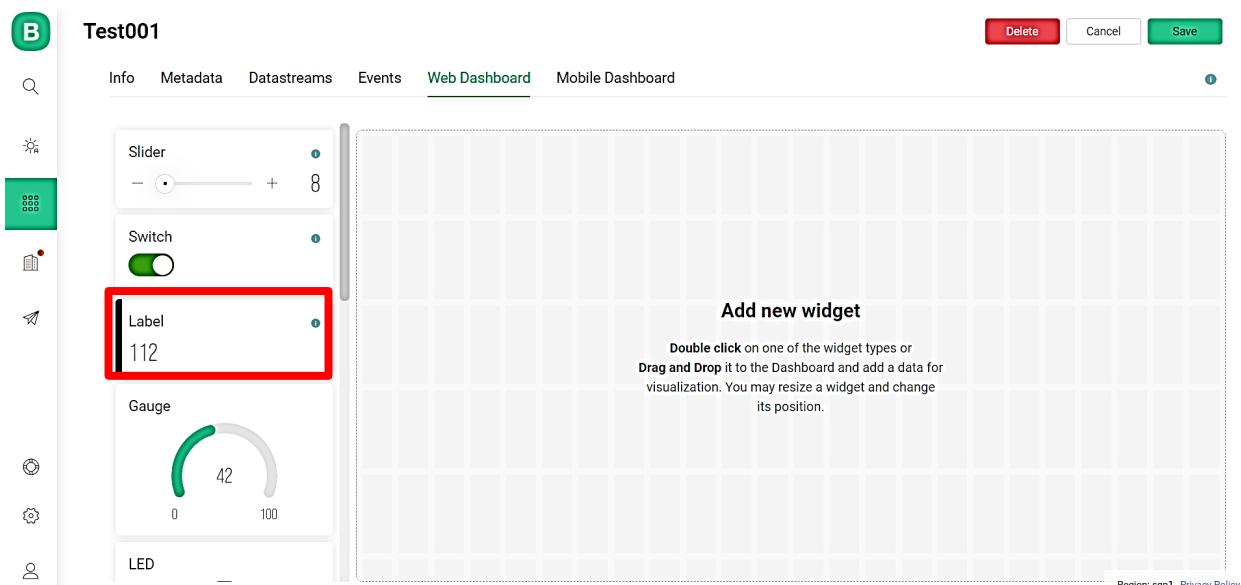
Workshop 13 Blynk and DS18B20

อุปกรณ์ที่ต้องใช้

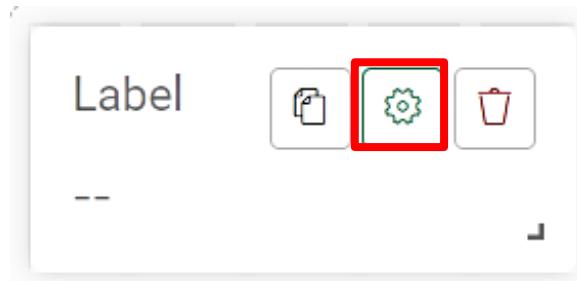
1. NodeMCU ESP8266
2. เช่นเซอร์วัสดอุณหภูมิDS18B20
3. LED สีแดง
4. Buzzer เสียงเตือน
5. ตัวต้านทานขนาด 4.7k โอห์ม 1 ตัว และตัวต้านทานขนาด 220 โอห์ม 1 ตัว

การตั้งค่า Dashboard

1 เลือก Label widget นำมาวางบนหน้า Dashboard



2 กดเพื่อตั้งค่าการรับส่งข้อมูล



3 ตั้งชื่อ Label เป็น Temperature

Label settings ⓘ

TITLE

Datastream

You have no datastreams to select

+ Create Datastream

Temperature

Cancel

Save

4 กด Create Datastream แล้วเลือก Virtual Pin

Label settings

TITLE
Temperature

Datastream
You have no datastreams to select

+ Create Datastream

Digital

Analog

Virtual Pin

Enumerable

Location UPGRADE

Temperature
--

Cancel **Save**

5 ให้ตั้งค่า PIN เป็น V0 , ตั้งค่า Units เป็น Celsius , ตั้งค่า max เป็น 100 และกด Create

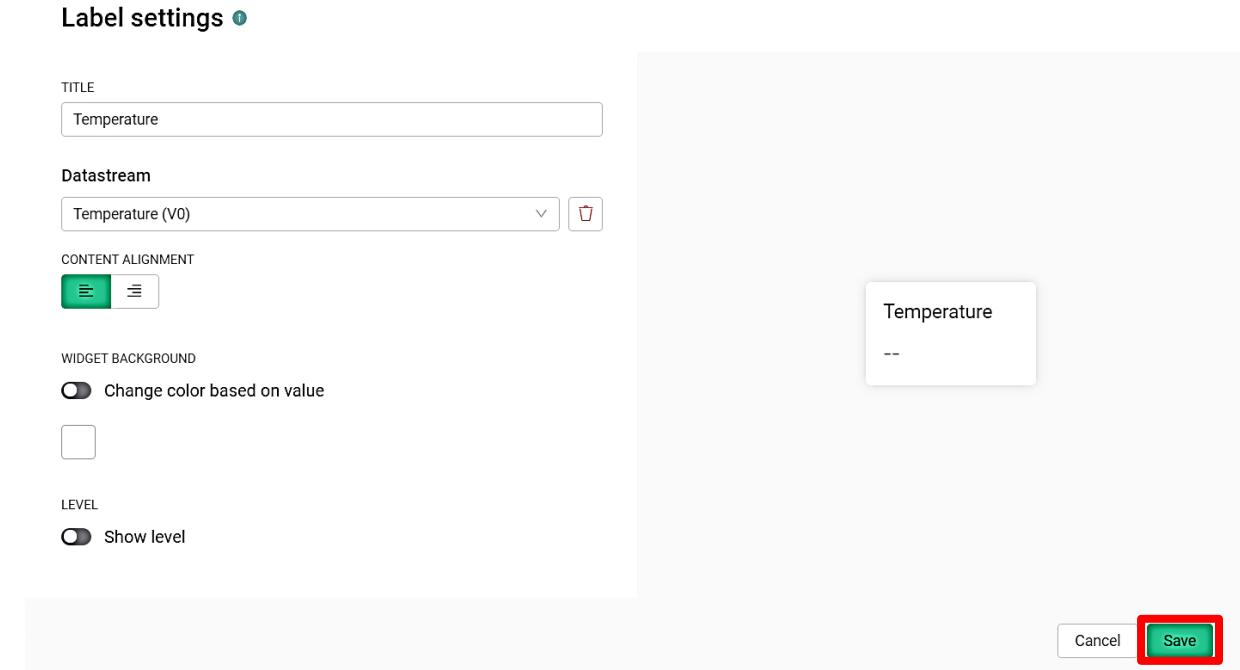
Datastream

Virtual Pin Datastream

PIN	DATA TYPE		
V0	Double		
UNITS			
Celsius			
MIN	MAX	DECIMALS	DEFAULT VALUE
0	100	#.##	0

Cancel **Create**

6 กด save เพื่อบันทึก Label



7 กด save เพื่อบันทึกหน้า dashboard

8 เลือก update 1 active device และกด continue

Apply Changes?

There is 1 active device associated with
Test001 template

How to apply changes?

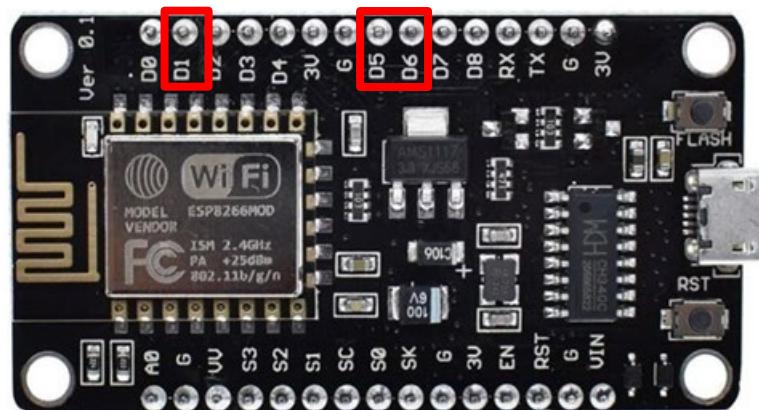
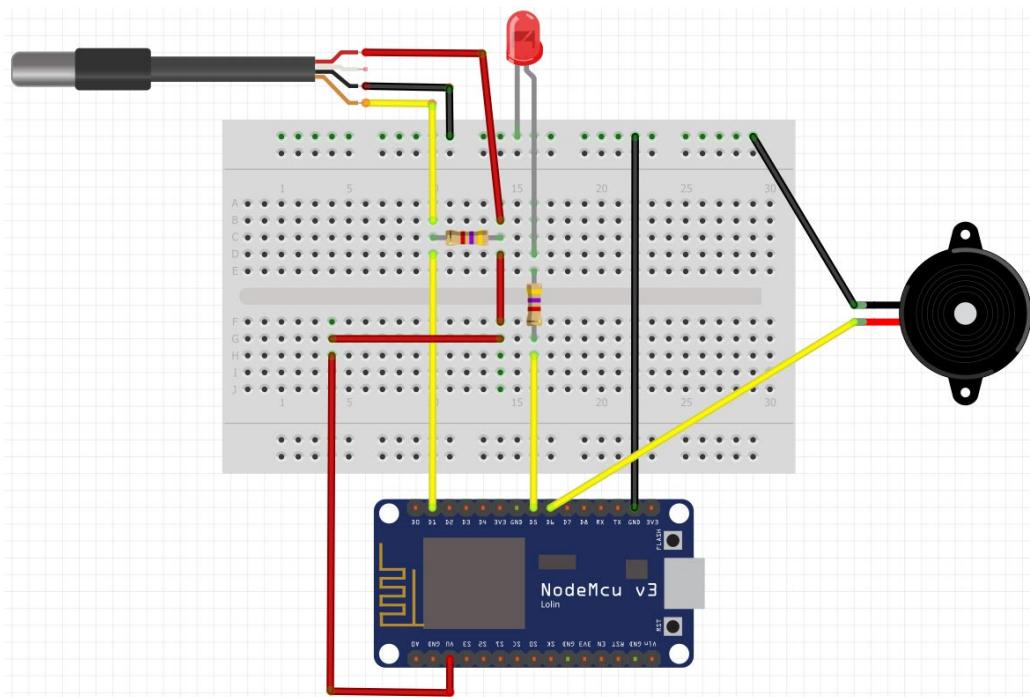
- Update 1 active device
- Save Changes. Don't update active device
- Create a clone of this Template with updated Metadata

Continue

Cancel

การต่อวงจร

PIN	อุปกรณ์
D1	DS18B20
D5	LED สีแดง
D6	Buzzer



การเขียน code (กรอบสีแดงคือต้องเปลี่ยนเป็นข้อมูลของตัวเอง)

Workshop_13

```
#define BLYNK_TEMPLATE_ID "your template id"
#define BLYNK_DEVICE_NAME "your device name"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS D1 //กำหนดขาที่จะเชื่อมต่อ Sensor
#define LED D5
#define Buzzer D6

const char ssid[] = "Wifi_name";
const char pass[] = "password";
const char auth[] = "your token";

float c = 0;

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
BlynkTimer timer;
void timerEvent();

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

    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, timerEvent);

    sensors.begin();
    pinMode(LED, OUTPUT);
    pinMode(Buzzer, OUTPUT);
}

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

```
void timerEvent() {
    sensors.requestTemperatures(); // อ่านข้อมูลจาก library
    c = sensors.getTempCByIndex(0);
    Serial.print("Temperature = ");
    Serial.print(c); // แสดงค่า อุณหภูมิ
    Serial.println(" °C");

    Blynk.virtualWrite(v0, c); // ส่งค่าไปที่ blynk โดยใช้ visualpin 0

    if (sensors.getTempCByIndex(0) > 27) {
        digitalWrite(LED, LOW); // สั่งให้ LED ดับ
        digitalWrite(Buzzer, HIGH); // สั่งให้ Buzzer ส่งเสียง
    }
    else {
        digitalWrite(LED, HIGH); // สั่งให้ LED ติดสว่าง
        digitalWrite(Buzzer, LOW); // สั่งให้ Buzzer ดับ
    }
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_13/Workshop_13.ino

Workshop 14 การใช้งาน DHT22 ร่วมกับ Blynk

ข้อมูลของ DHT22

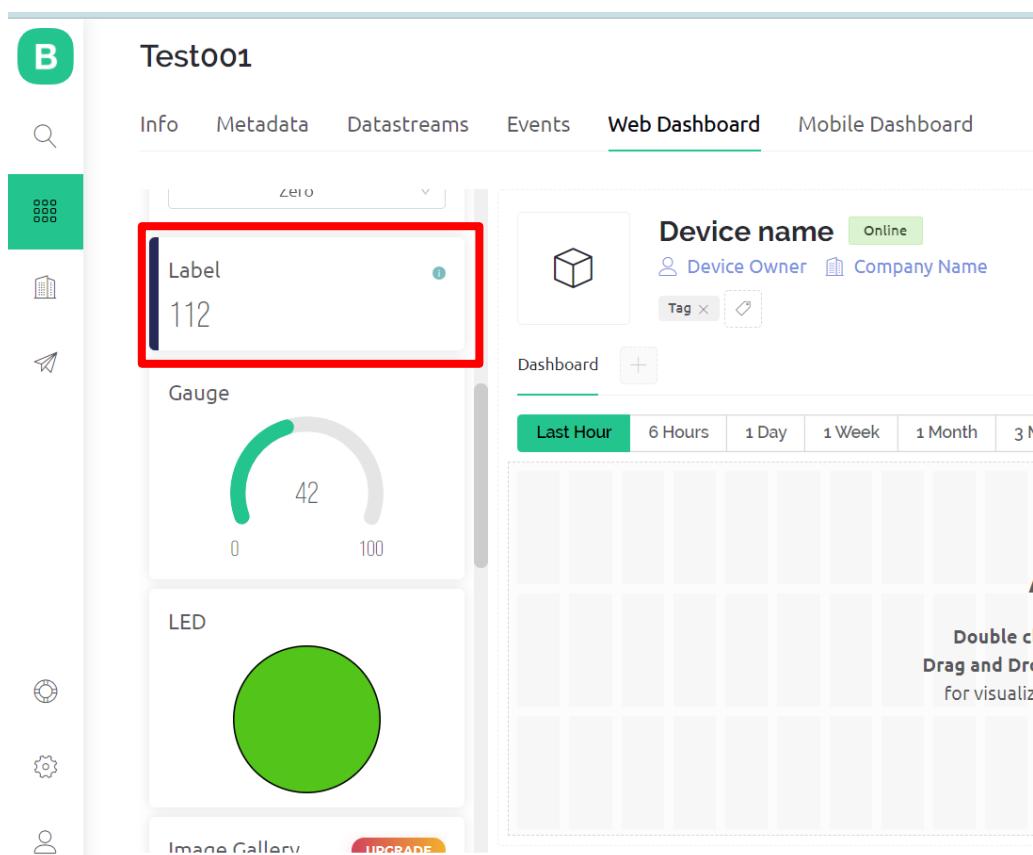
1. Power supply : 3 - 5.5V
2. วัดอุณหภูมิได้ระหว่าง 0 - 50 องศาเซลเซียส +/- 2 องศา
3. วัดความชื้นในอากาศได้ระหว่าง 20 - 90 % +/- 5%
4. เวลาที่ใช้ในการวัดค่า : 1 วินาที

อุปกรณ์ที่ใช้

1. NodeMCU ESP8266
2. เช่นเซอร์วัตอุณหภูมิ DHT22
3. สายไฟ

การตั้งค่า Dashboard

1. กดเลือกใช้ Label 2 label ลากลงหน้า dashboard



2. กดตั้งค่า Label ตัวที่หนึ่งเพื่อรับค่าอุณหภูมิ

The screenshot shows a dashboard with a 'Label' card. The card has three icons: a white square with a black outline, a green gear icon with a red border, and a red trash bin icon. The green gear icon is highlighted with a red box.

3. ตั้งชื่อ Label ใช้ชื่อว่า Temperature

Label Settings ⓘ

TITLE (OPTIONAL)

Datastream

Choose Source ▾ or [+ Create Datastream](#)

4. กด Create Datastream เลือก Virtual Pin

Label Settings ⓘ

TITLE (OPTIONAL)

Temperature

Datastream

You have no datastreams to select

[+ Create Datastream](#)

Digital

Analog

Virtual Pin

Enumerable

Location [UPGRADE](#)

5. ตั้งค่า PIN : V0 / Units : Celsius

Label Settings ⓘ

TITLE (OPTIONAL)

Temperature

Datastream

Virtual Pin Datastream



Temperature

Temperature

PIN

V0

DATA TYPE

Double

UNITS

Celsius

Cancel

Create

6. ตั้งค่า MIN : 0 / MAX : 100 และกด create

Label Settings ⓘ

TITLE (OPTIONAL)

Temperature

Datastream

Virtual Pin Datastream

MIN	MAX	DECIMALS	DEFAULT VALUE
0	100	#.##	0

Thousands separator (e.g. 10,000)

[+] ADVANCED SETTINGS

Cancel

Create

7. กด save Label

Label Settings ⓘ

TITLE (OPTIONAL)

Temperature

Datastream

Temperature (V0)



CONTENT ALIGNMENT



WIDGET BACKGROUND

Change color based on value



LEVEL

Show level

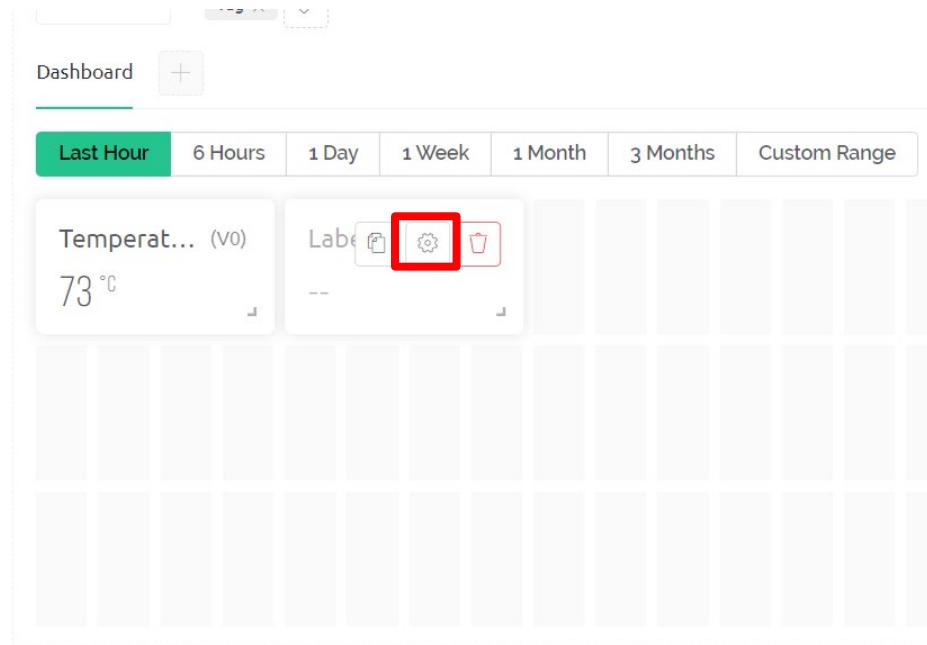
Tempera... (V0)

94 °C

Cancel

Save

8. ตั้งค่า Label ตัวที่สอง

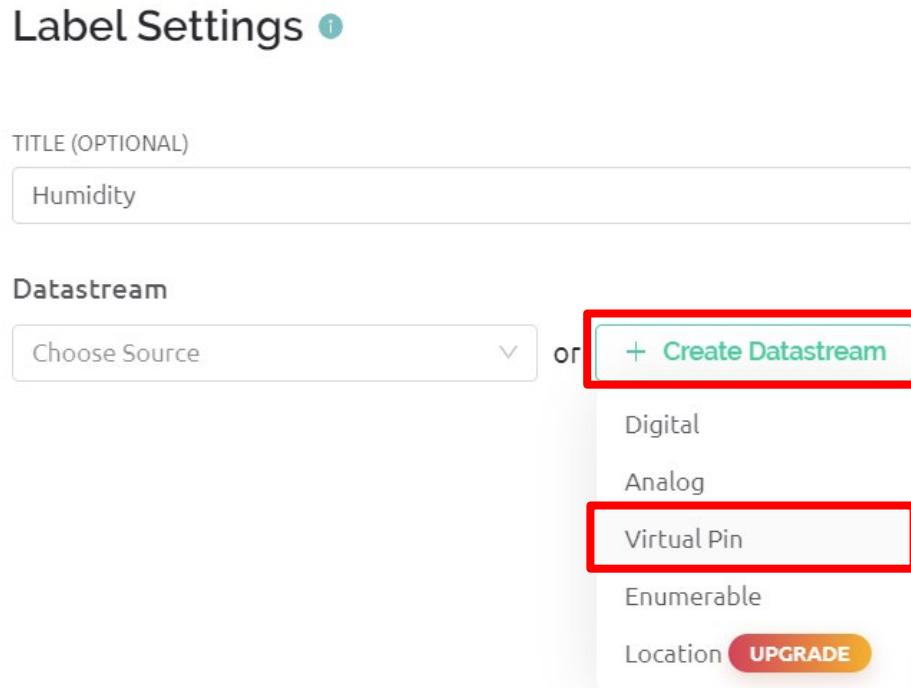


9. ตั้งชื่อ Label ใช้ชื่อว่า Humidity

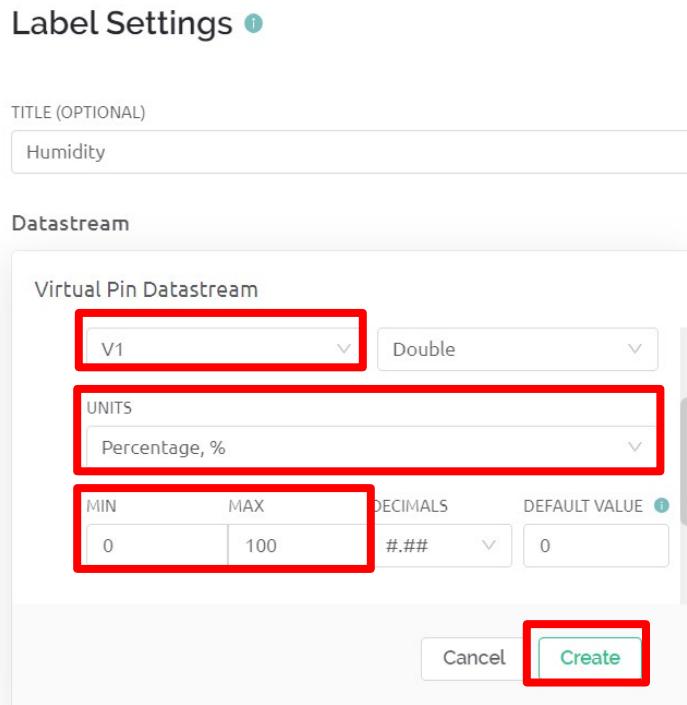
Label Settings ⓘ

The screenshot shows the 'Label Settings' dialog box. At the top, there is a 'TITLE (OPTIONAL)' field with the text 'Humidity' (which is highlighted with a red box). Below this is a 'Datastream' section. It includes a dropdown menu labeled 'Choose Source' and a button labeled '+ Create Datastream'.

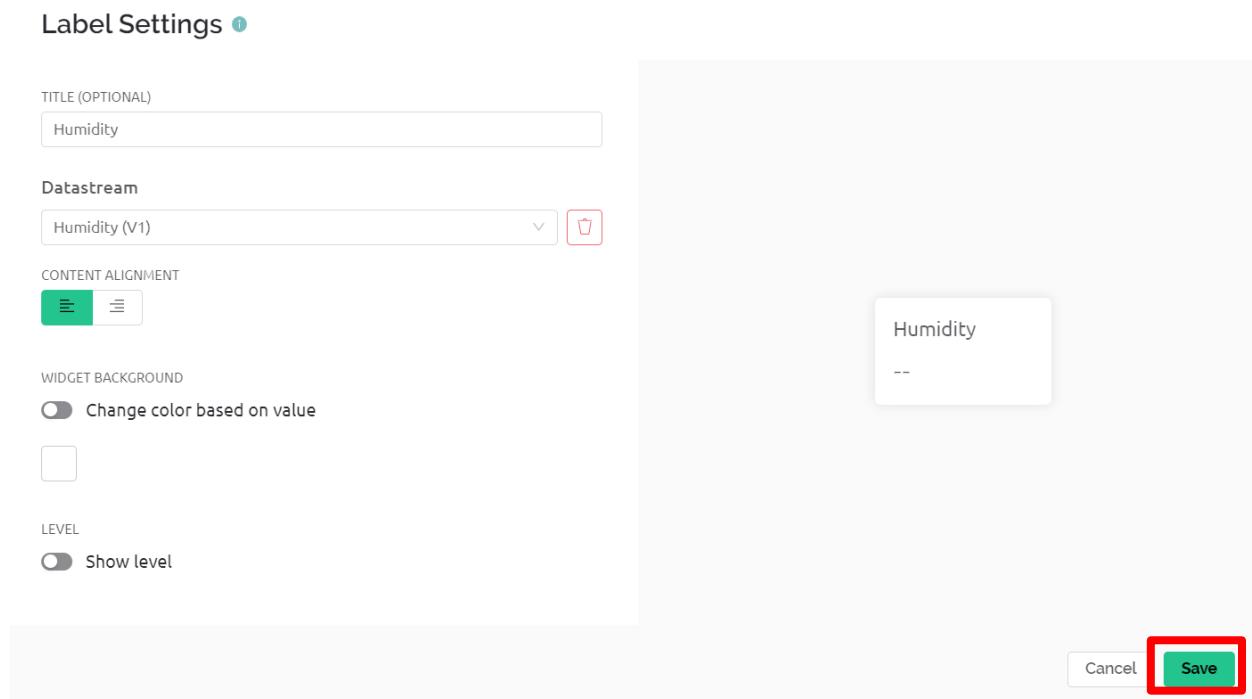
10. กด Create Datastream เลือก Virtual Pin



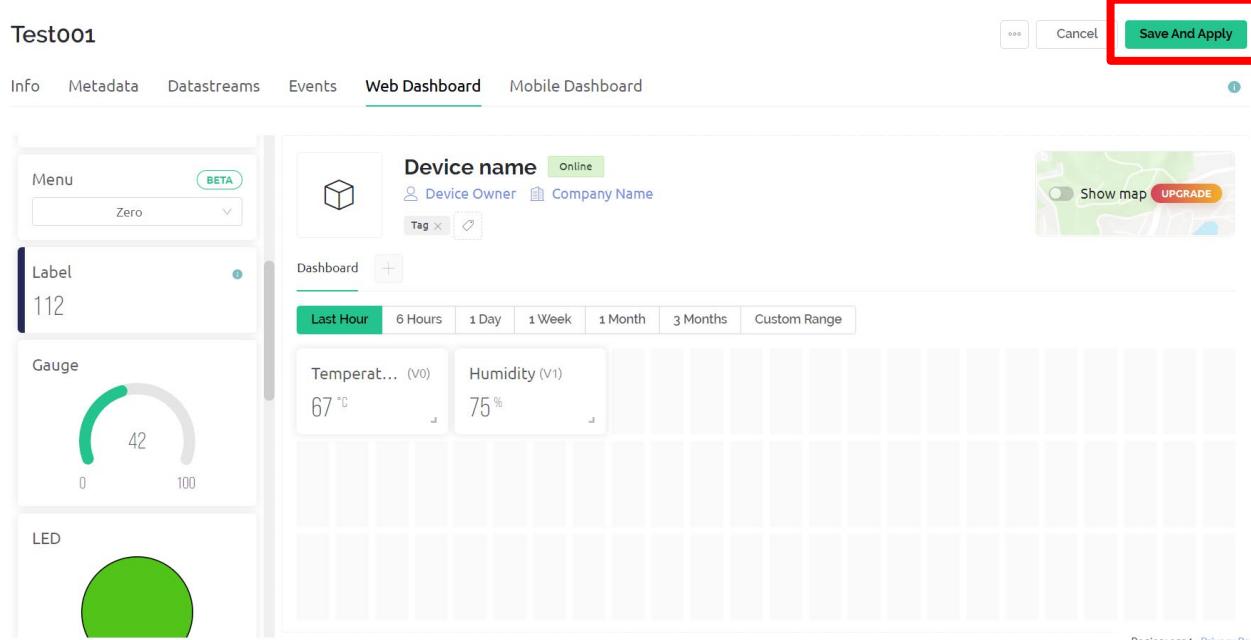
11. ตั้งค่า PIN : V1 / Units : Percentage,% / MIN : 0 / MAX : 100 และกด Create



12. ຜັນ save Label



13. ຜັນ Save And Apply



(Optional) หากต้องการดูประวัติที่ตัวเซ็นเซอร์ได้เก็บข้อมูลมาเราสามารถใช้ widget ที่ชื่อ chart ได้

14. ให้นำ Chart มา 2 ตัวลงใน dashboard

The screenshot shows the 'Web Dashboard' tab selected in the top navigation bar. On the left, there's a sidebar with various icons. A red box highlights the 'Chart' icon in the first row. The main dashboard area displays a gauge with a value of 42, a 'Device name' section showing 'online' status, and a map with a 'Show map' button.

15. ตั้งค่า Chart ตัวที่หนึ่ง

The screenshot shows the 'Dashboard' tab selected in the top navigation bar. Below it, a time range selector shows 'Last Hour' is selected. The main area contains two data cards: 'Tempera...' (97 °C) and 'Humidity (V1)' (20 %). Below these are two chart components. The first chart has a red box around its gear icon. Both charts currently show 'No Data'.

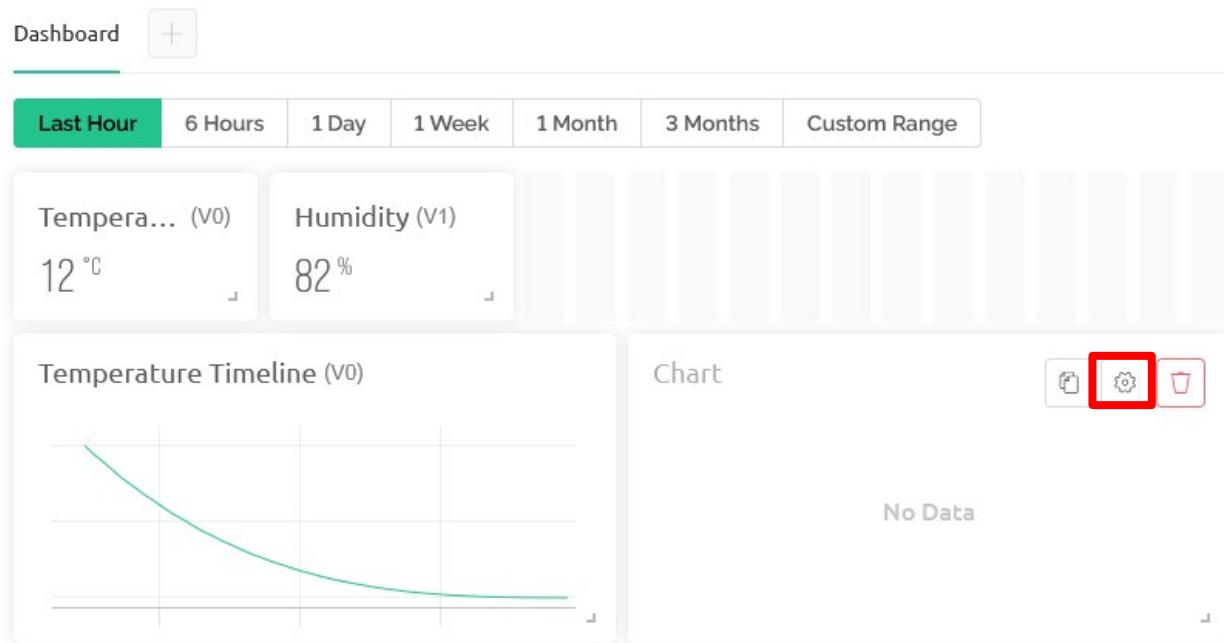
16. ตั้งชื่อ Title ว่า Temperature Timeline จากนั้นกดเลือก Add Datastream และกดเลือก Temperature

The screenshot shows the 'Chart Settings' dialog. On the left, under 'TITLE (OPTIONAL)', the text 'Temperature Timeline' is entered and highlighted with a red box. Below it, under 'Datastreams', there is a list with 'Temperature (V0)' selected and highlighted with a red box. There are buttons for '+ Add Datastream' and '+ Create New'. A 'Show legend' toggle switch is also present. On the right, a preview window titled 'Temperature Timeline' shows the text 'No Data'. At the bottom right of the dialog are 'Cancel' and 'Save' buttons.

17. เมื่อตั้งค่าเสร็จแล้วกด save

The screenshot shows the 'Chart Settings' dialog after saving. The configuration includes a title 'Temperature Timeline', a single datastream 'Temperature (V0)', and a chart type set to a line graph. The preview window on the right displays a line graph titled 'Temperature Timeline (V0)' showing a single teal line starting at approximately 200 on the Y-axis and ending at approximately 100. At the bottom right of the dialog, the 'Save' button is highlighted with a red box.

18. ตั้งค่า Chart ตัวที่สอง



19. ตั้งชื่อ Title ว่า Humidity Timeline จากนั้นกดเลือก Add Datastream และกดเลือก Humidity

The screenshot shows the 'Chart Settings' dialog box. It includes fields for 'TITLE (OPTIONAL)' containing 'Humidity Timeline' (highlighted with a red box) and a 'Datastreams' section with a list of available streams: '+ Add Datastream' (button), 'or', '+ Create New' (button), 'Humidity (V1)' (highlighted with a red box), and 'Temperature (V0)'. There is also a 'Show legend' toggle switch. On the right side of the dialog, there is a preview window titled 'Humidity Timeline' showing a graph with the message 'No Data'. At the bottom right of the dialog, there are 'Cancel' and 'Save' buttons, with 'Save' being highlighted with a red box.

20. เมื่อตั้งค่าเสร็จแล้วกด save

Chart Settings

TITLE (OPTIONAL)
Humidity Timeline

Datastreams
UPGRADE to add more datastreams

CHART TYPE

CHART COLOR

Show Y-axis

Autoscale

MIN MAX

Cancel **Save**

21. กด Save And Apply

B Test001

Info Metadata Datastreams Events Automations Web Dashboard Mobile Dashboard

Widget Box
4 of 30 widgets

CONTROL

Switch

Slider

Number Input

DISPLAY

Device name Online

Device Owner Company Name

Dashboard

Last Hour 6 Hours 1 Day 1 Week 1 Month 3 Months Custom Range

Temperature Timeline (V0) Humidity (V1)

60 °C 92 %

Temperature Timeline (V0)

LTS Temperature: 0.58 °C

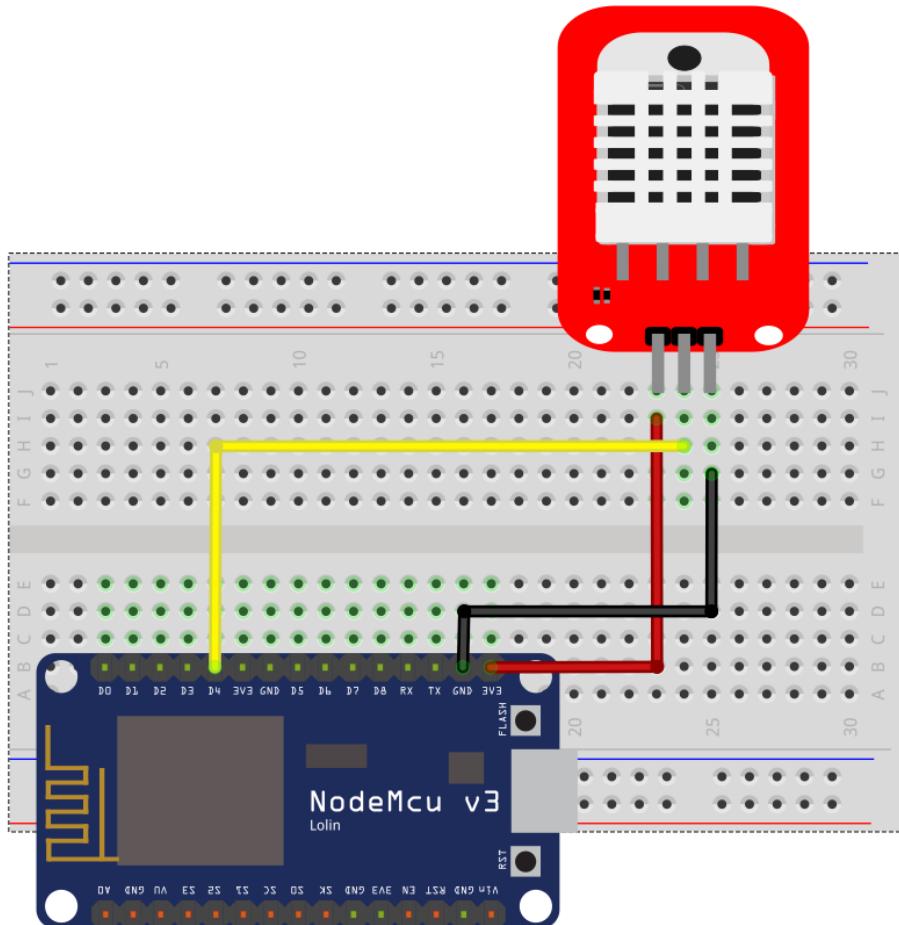
Humidity Timeline (V1)

Region: sgp1 Privacy Policy

Save And Apply

การต่อวงจร

PIN	อุปกรณ์
D4	DHT22



การเขียน code

Workshop_14

```
#define BLYNK_TEMPLATE_ID "your template id"
#define BLYNK_DEVICE_NAME "your device name"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

#define DHTPIN D4 //ใช้ปุ่มกดที่จะใช้ PIN D4 ในการรับข้อมูลจาก DHT22
#define DHTTYPE DHT22

const char ssid[] = "Wifi_name";
const char pass[] = "password";
const char auth[] = "your token";

DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;
void timerEvent();

void setup() {
    dht.begin(); //สั่งให้ DHT22 เริ่มทำงาน
    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, timerEvent);
    Serial.begin(115200);
}

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

void timerEvent(){
    float t = dht.readTemperature(); //รับค่าอุณหภูมิในอากาศจาก DHT22
    float h = dht.readHumidity(); //รับค่าความชื้นในอากาศจาก DHT22
    if(isnan(t)||isnan(h)){//ใช้ในการเช็คค่าจาก DHT22 ว่ามีค่าส่งมาหรือไม่
        Serial.println("Failed!"); //ถ้าไม่มีค่าส่งมาจะขึ้นหน้า failed
    }
    else{
        Serial.print("Temp : ");
        Serial.print(t); //แสดงค่าอุณหภูมิในอากาศ
        Serial.println("*C");
        Serial.print("Humid : ");
        Serial.print(h); //แสดงค่าความชื้นในอากาศ
        Serial.println("%");
        Serial.println("-----");
        Blynk.virtualWrite(V0,t); //ส่งค่าอุณหภูมิไปที่ blynk โดยใช้ visualpin 0
        Blynk.virtualWrite(V1,h); //ส่งค่าความชื้นไปที่ blynk โดยใช้ visualpin 1
    }
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_14/Workshop_14.ino

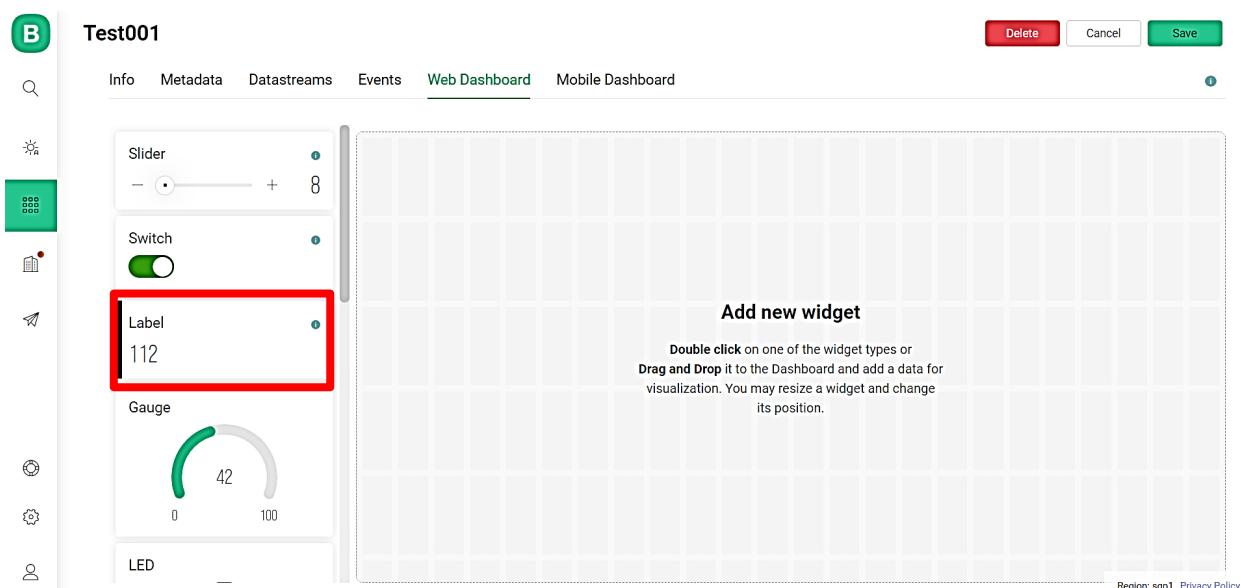
Workshop 15 Blynk and All sensor

อุปกรณ์ที่ต้องใช้

1. NodeMCU ESP8266
2. เซนเซอร์วัดอุณหภูมิ DS18B20
3. ultrasonic sensor
4. soil moisture sensor
5. LED สีแดง, สีเขียว และสีเหลือง
6. Relay
7. DC Pump 5 V

การตั้งค่า Dashboard

1 เลือก Label widget 3 Label นำมาระบบทัน Dashboard



2 เลือก switch widget นำมาระบบหน้า Dashboard

3 เลือก chart widget นำมาระบบหน้า Dashboard

4 กดเพื่อตั้งค่าการรับส่งข้อมูลของ Label แรก

5 ตั้งชื่อ Label เป็น Temperature

Label settings ⓘ

TITLE

Datastream

You have no datastreams to select

[+ Create Datastream](#)

Label

--

[Cancel](#) [Save](#)

6 กด Create Datastream แล้วเลือก Virtual Pin

Label settings ⓘ

TITLE
Temperature

Datastream
You have no datastreams to select

- + Create Datastream
- Digital
- Analog
- Virtual Pin**
- Enumerable
- Location **UPGRADE**

Temperature
--

Cancel **Save**

7 ให้ตั้งค่า PIN เป็น V0 , ตั้งค่า Units เป็น Celsius

Datastream

Virtual Pin Datastream

	Temperature	Temperature	
PIN	V0	DATA TYPE	Double
UNITS	Celsius		

Cancel **Create**

8 ตั้งค่า max เป็น 100 และกด Create

Datastream

Virtual Pin Datastream

MIN	MAX	DECIMALS	DEFAULT VALUE
0	100	#.##	0

Thousands separator (e.g. 10,000)

ADVANCED SETTINGS

Cancel **Create**

9 กด save เพื่อบันทึก Label นี้

Label settings ⓘ

TITLE
Temperature

Datastream
Temperature (V0)

CONTENT ALIGNMENT



WIDGET BACKGROUND

Change color based on value



LEVEL

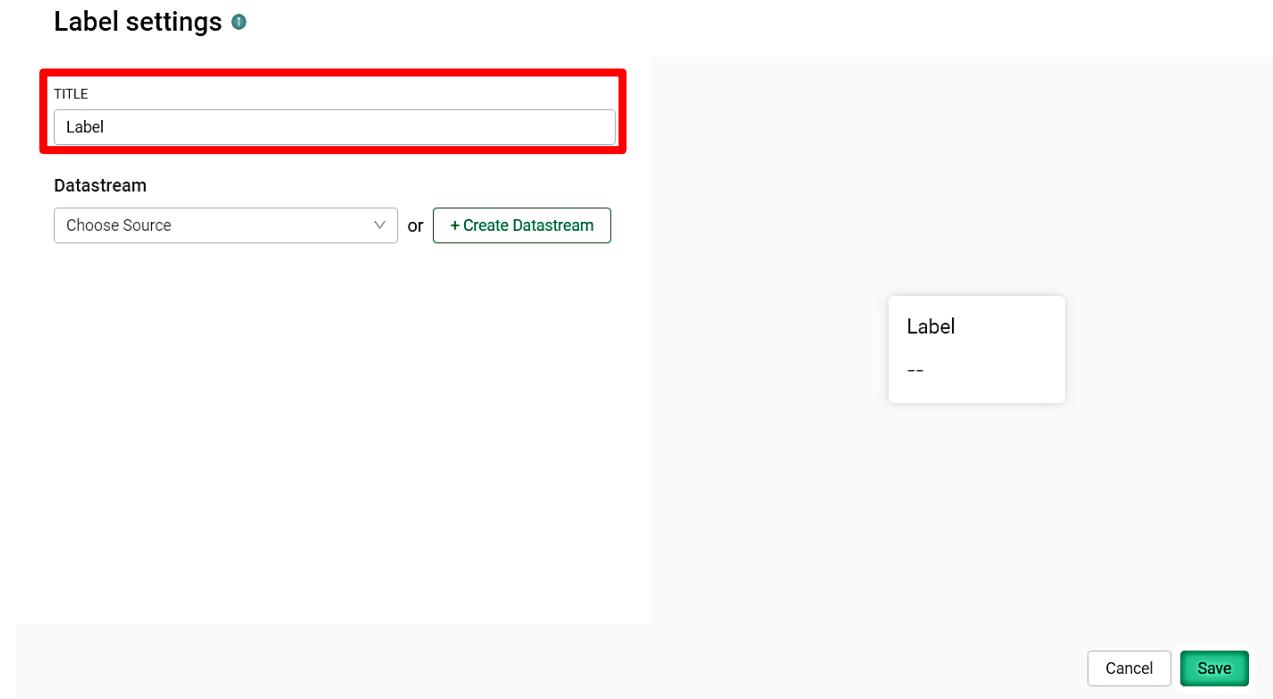
Show level

Temperature

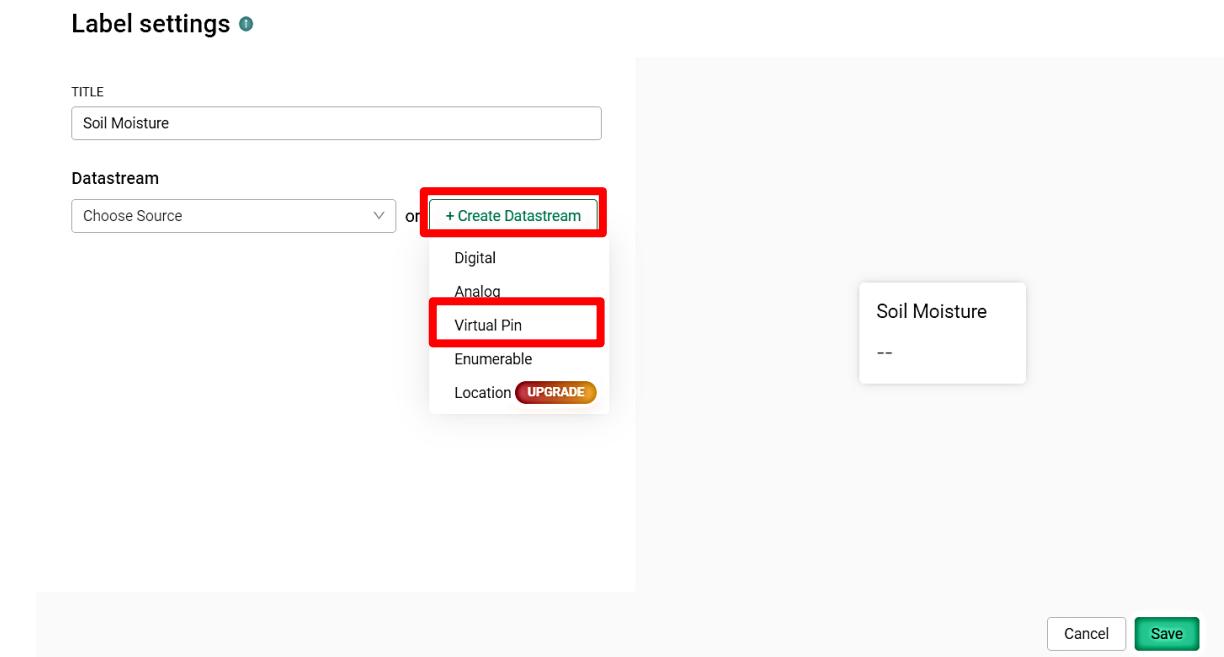
--

Cancel **Save**

10 ตั้งค่า Label ตัวที่สอง ตั้งชื่อ Soil Moisture



11 กด Create Datastream และเลือก Virtual Pin



12 ให้ตั้งค่า PIN เป็น V1 , ตั้งค่า Units เป็น Percentage %

Datastream

Virtual Pin Datastream

	Soil Moisture	Soil Moisture	
PIN	V1	DATA TYPE	Double
UNITS	Percentage, %		

Cancel Create

13 ตั้งค่า max เป็น 100 และกด Create

Datastream

Virtual Pin Datastream

MIN	MAX	DECIMALS	DEFAULT VALUE
0	100	#.##	0

Thousands separator (e.g. 10,000)

ADVANCED SETTINGS

Cancel Create

14 กด save เพื่อบันทึก Label นี้

Label settings ⓘ

TITLE
Soil Moisture

Datastream
Soil Moisture (V1) ▾ 

CONTENT ALIGNMENT



WIDGET BACKGROUND

Change color based on value



LEVEL

Show level

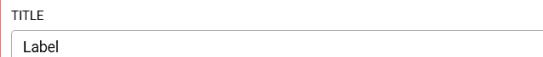
Soil Moisture

--

Cancel 

15 ตั้งค่า Label ตัวที่สาม ตั้งชื่อ Label Distance

Label settings ⓘ

TITLE
Label

Datastream

Choose Source ▾ or 

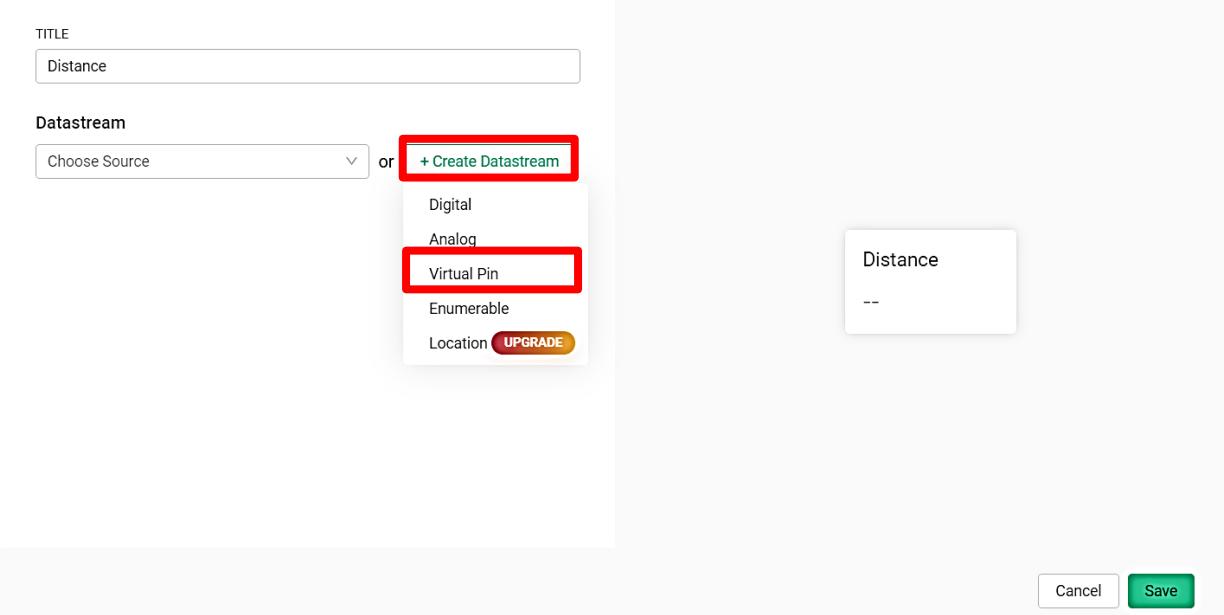
Label

--

Cancel 

16 กด Create Datastream และเลือก Virtual Pin

Label settings ⓘ



17 ให้ตั้งค่า PIN เป็น V2 , ตั้งค่า Units เป็น Centimeter

Datastream

Virtual Pin Datastream	
	Distance
Distance	
PIN	DATA TYPE
V2	Double
UNITS	
Centimeter	
<input type="button" value="Cancel"/> <input type="button" value="Create"/>	

18 ตั้งค่า max เป็น 1,000 และกด Create

Datastream

Virtual Pin Datastream

MIN	MAX	DECIMALS	DEFAULT VALUE
0	1000	#.##	0

Thousands separator (e.g. 10,000)

ADVANCED SETTINGS

19 กด save เพื่อบันทึก Label

Label settings

TITLE

Datastream

CONTENT ALIGNMENT

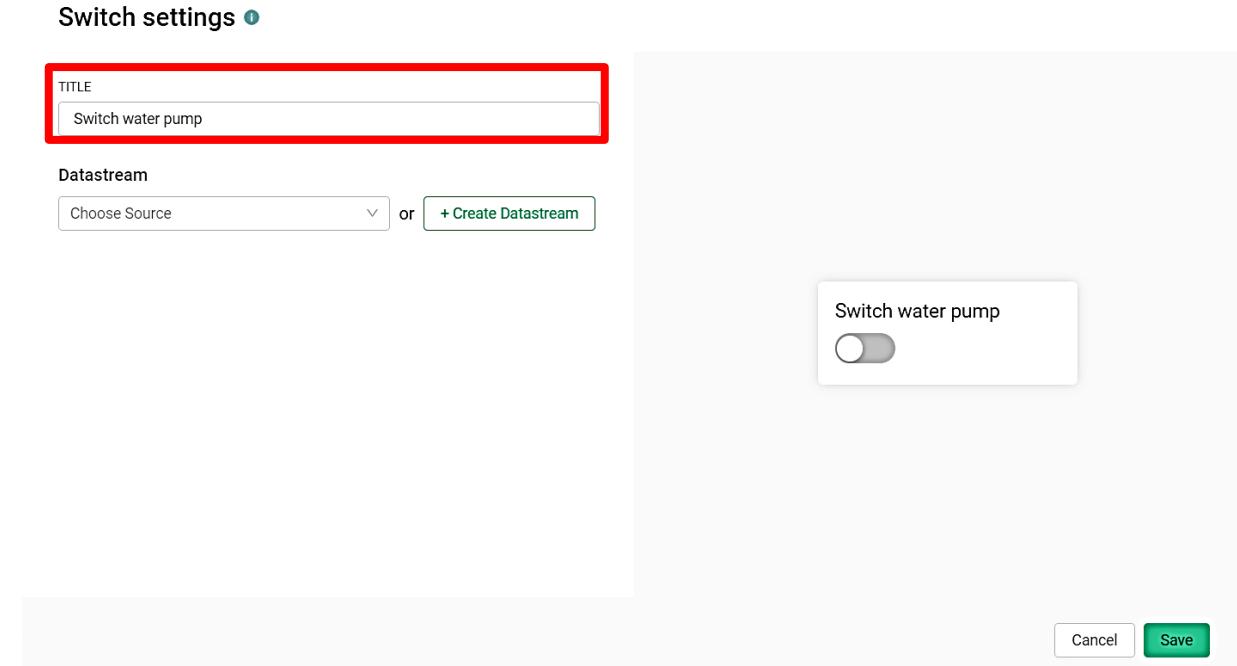
WIDGET BACKGROUND
 Change color based on value

LEVEL
 Show level

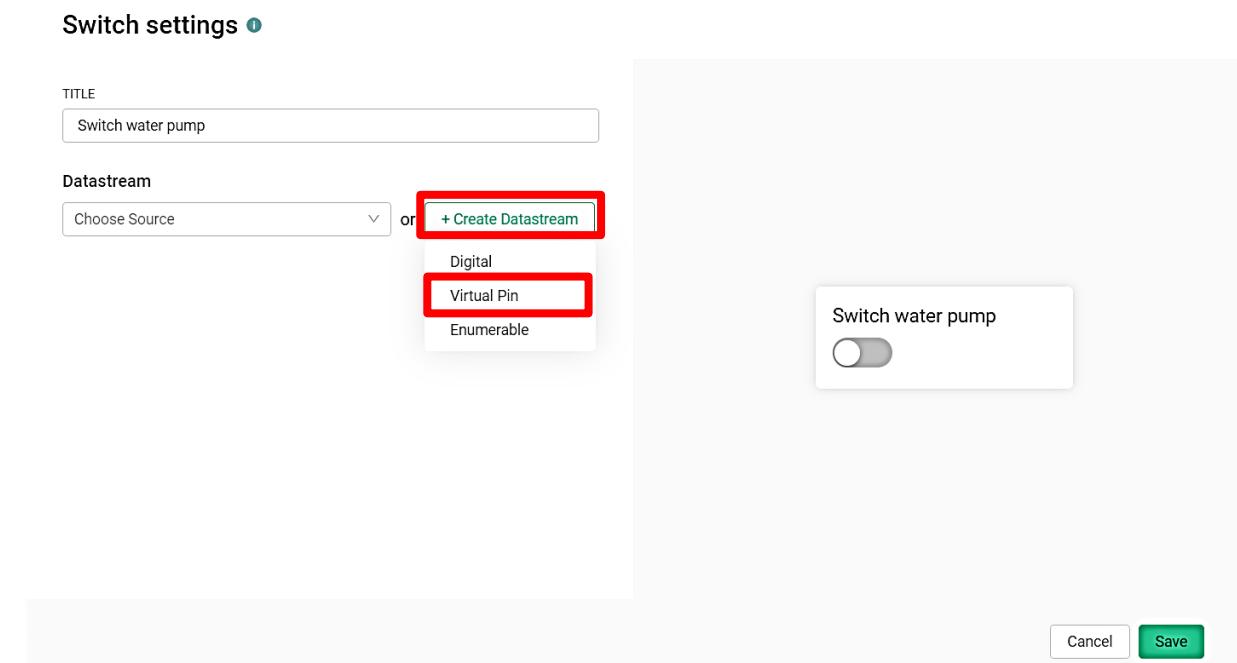
Distance

--

20 ตั้งค่า Switch ตั้งชื่อว่า Switch water pump



21 กด Create Datastream และเลือก Virtual Pin



22 ให้ตั้งค่า PIN เป็น V3 และกด Create

Datastream

Virtual Pin Datastream

NAME	ALIAS
 Switch water pump	Switch water pump 
PIN	DATA TYPE
V3	Double
UNITS	
None	
<input type="button" value="Cancel"/> <input style="border: 2px solid red;" type="button" value="Create"/>	

23 กด save เพื่อบันทึก Label นี้

Switch settings ⓘ

TITLE

Switch water pump

Datastream

Switch water pump (V3) 

ON VALUE OFF VALUE

1 0 

Show on/off labels

Hide widget name

Switch water pump 

24 ตั้งค่า chart ตั้งชื่อว่า Temperature Timeline และเลือก Add Datastream

Chart Settings

TITLE (OPTIONAL)
Temperature Timeline

Datastreams
+ Add Datastream or + Create New

Enable Zoom
 Show legend

Temperature Timeline
No Data

Cancel Save

25 กดเลือก Temperature และ กด save เพื่อบันทึก chart นี้

Chart Settings

TITLE (OPTIONAL)
Temperature Timeline

Datastreams
UPGRADE to add more datastreams
Temperature (V0) AVG of

CHART TYPE

CHART COLOR

Show Y-axis
 Autoscale

MIN 0 MAX 100

Temperature Timeline (V0)

Cancel Save

26 กด save เพื่อบันทึกหน้า dashboard

27 เลือก update 1 active device และกด continue

Apply Changes?

There is 1 active device associated with
Test001 template

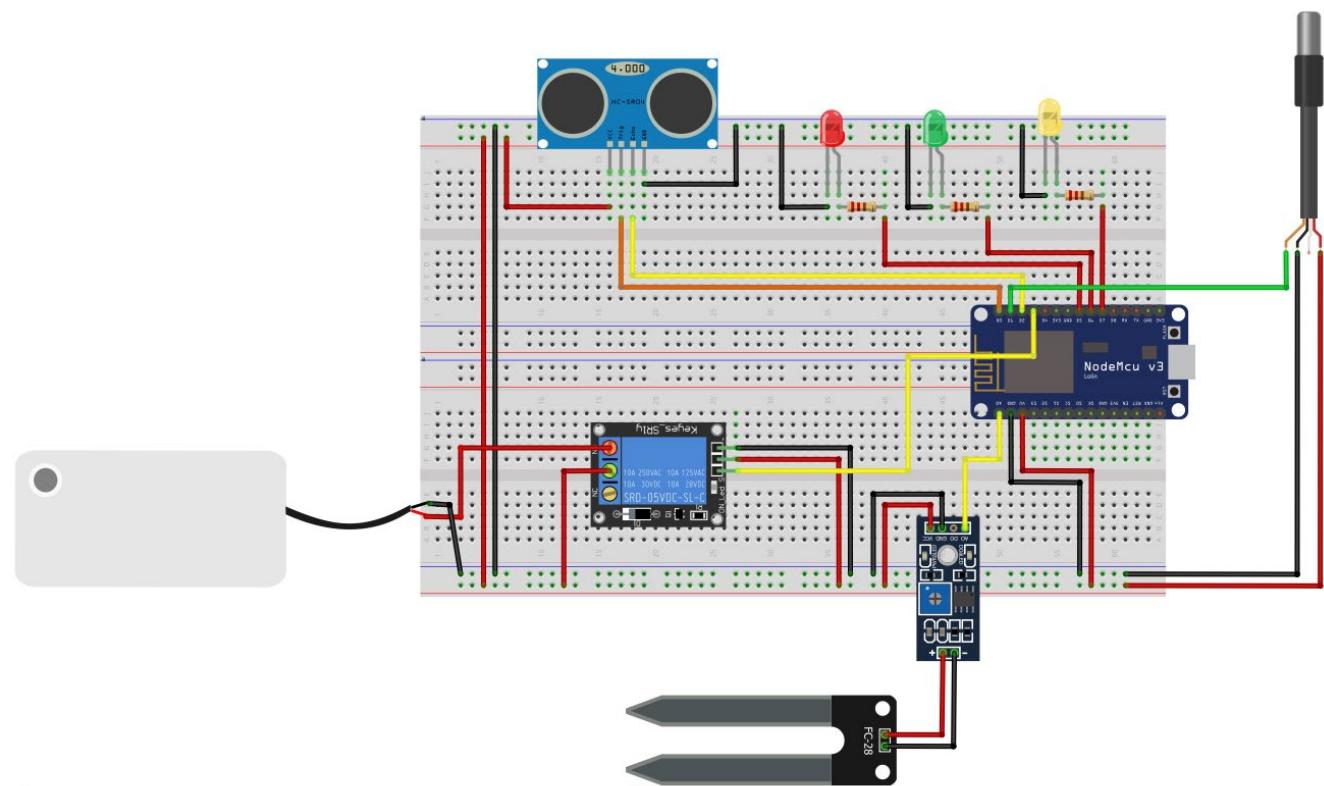
How to apply changes?

- Update 1 active device
- Save Changes. Don't update active device
- Create a clone of this Template with updated Metadata

Continue

การต่อวงจร

PIN	อุปกรณ์
A0	Soil Moisture Sensor
D0	Ultrasonic (Trig)
D1	DS18B20
D2	Ultrasonic (Echo)
D3	Relay
D5	LED สีแดง
D6	LED สีเขียว
D7	LED สีเหลือง



:zina

การเขียน code (กรอบสีแดงคือต้องเปลี่ยนเป็นข้อมูลของตัวเอง)

Workshop_15

```
#define BLYNK_TEMPLATE_ID "your template id"
#define BLYNK_DEVICE_NAME "your device name"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <OneWire.h>
#include <DallasTemperature.h>

//ประกาศตัวแปร
#define SOUND_VELOCITY 0.034

#define TRIG D0
#define ONE_WIRE_BUS D1
#define ECHO D2
#define RELAY D3
#define LED_R D5
#define LED_G D6
#define LED_Y D7

#define SOIL_MOIST A0

const char ssid[] = "Wifi_name";
const char pass[] = "password";
const char auth[] = "your token";

int raw_data = 0;
int moisture = 0;

long duration;
float distanceCm;

float c = 0;

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
BlynkTimer timer;

void timerEvent();
void pinConfig();
void pushDistance();
void pushMoisture();
void pushTemp();
```

```
void setup(void) {
    Serial.begin(115200);
    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, timerEvent);
    pinConfig();
}

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

BLYNK_WRITE(V3) { //function visualpin3 ที่ในการกดปุ่มบน blynk
    if(param.asInt()){
        digitalWrite(RELAY, LOW); }
    else{
        digitalWrite(RELAY, HIGH); }
}

void pinConfig(){
    pinMode(TRIG, OUTPUT);
    pinMode(ECHO, INPUT);
    digitalWrite(TRIG, LOW);

    pinMode(RELAY, OUTPUT);
    digitalWrite(RELAY, HIGH);

    sensors.begin();

    pinMode(LED_R, OUTPUT);
    pinMode(LED_G, OUTPUT);
    pinMode(LED_Y, OUTPUT);

    digitalWrite(LED_R, HIGH);
    digitalWrite(LED_G, HIGH);
    digitalWrite(LED_Y, HIGH);
}

void timerEvent(){
    pushTemp();
    pushDistance();
    pushMoisture();
}
```

```
void pushTemp() {
    // อ่านค่าอุณหภูมิจาก DS18B20 temperature sensor
    sensors.requestTemperatures();
    c = sensors.getTempCByIndex(0);
    Blynk.virtualWrite(V0, c);
    Serial.print("Temperature is: ");
    Serial.print(c);
    Serial.println(" °C");
    // เช็คอุณหภูมิ
    if(c > 27) {
        digitalWrite(LED_R, HIGH);
    }
    else{
        digitalWrite(LED_R, LOW);
    }
}

void pushMoisture() {
    // อ่านค่าความชื้นในดินจาก soil moisture sensor
    raw_data = analogRead(SOIL_MOIST);
    moisture = map(raw_data, 0, 1023, 100, 0);
    Blynk.virtualWrite(V1, moisture);
    Serial.print("Moisture = ");
    Serial.println(moisture);

    if (moisture > 50) {
        digitalWrite(LED_Y, LOW);
        digitalWrite(LED_G, HIGH);
    }
    else{
        digitalWrite(LED_Y, HIGH);
        digitalWrite(LED_G, LOW);
    }
}
```

```
void pushDistance() {
    //ส่งค่าระยะทางจาก ultrasonic sensor
    digitalWrite(TRIG, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);

    duration = pulseIn(ECHO, HIGH);
    distanceCm = duration * SOUND_VELOCITY/2;

    if (distanceCm >= 200 || distanceCm <= 0) {
        Serial.println("Out of range");
    }
    else {
        Blynk.virtualWrite(v2, distanceCm); //ส่งค่าไปที่ Blynk โดยใช้ visualpin v2
        Serial.print("Distance (cm): ");
        Serial.print(distanceCm);
        Serial.println(" cm");
    }
}
```

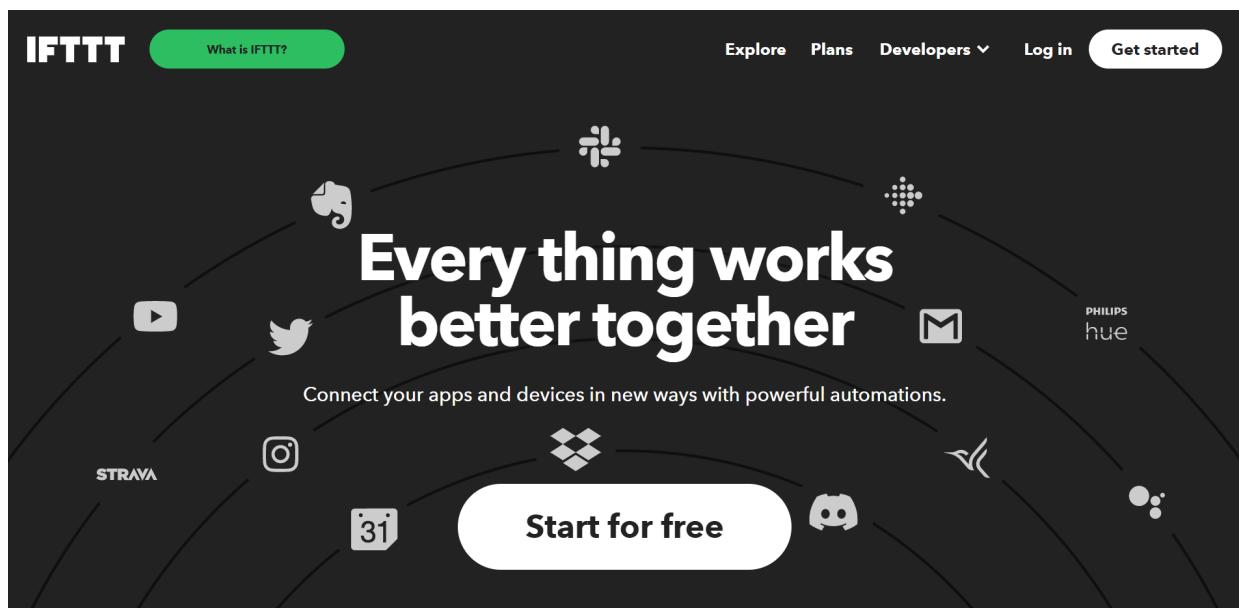
สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_15/Workshop_15.ino

IFTTT

IFTTT หรือ IF This Then That เป็นเว็บและแอปมหัศจรรย์ ที่เค้านำ API ของ Service หลายเจ้าในโลกใบนี้เข้ามาใช้ด้วยกันได้ สามารถสร้างสูตร (Recipe) ขึ้นมาได้อย่างอิสระ และนำไปเปแชร์ให้คนอื่นใช้ได้อีกด้วย

if + this then that

IF This Then That เพราะลักษณะการทำงานจะเป็น Flow เมื่อเกิดสิ่งใดสิ่งหนึ่งเกิดขึ้นตามเงื่อนไขที่เราสร้างไว้ แล้วจะเกิดสิ่งต่อไปตามมา

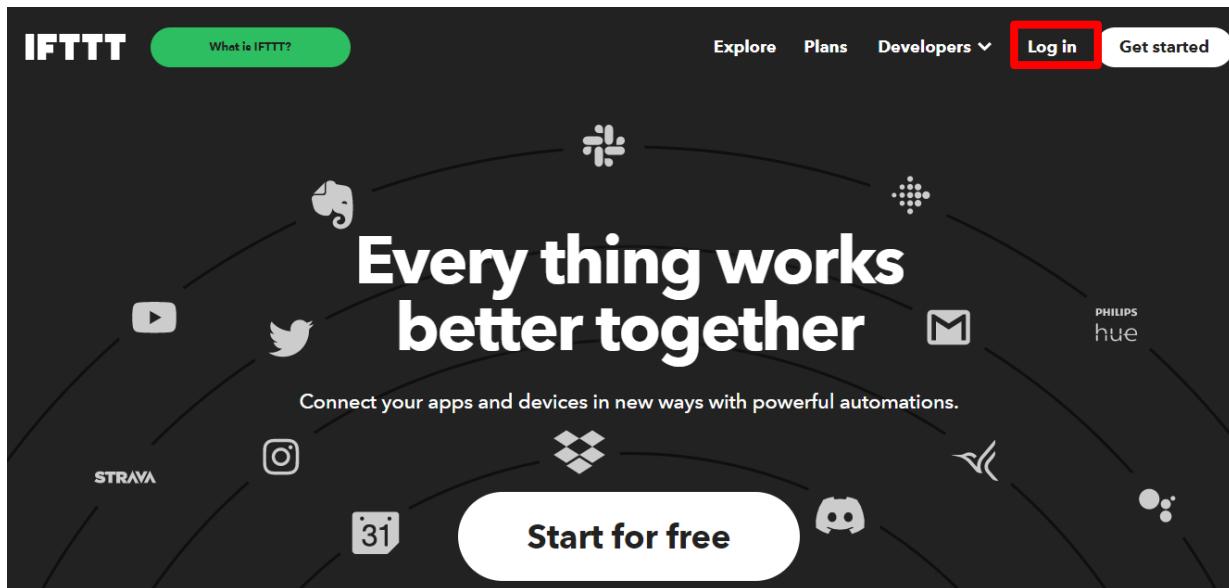


Workshop 16 การใช้งาน IFTTT ร่วมกับการบันทึกข้อมูลลงใน Google Sheet และแจ้งเตือนไปที่ Line

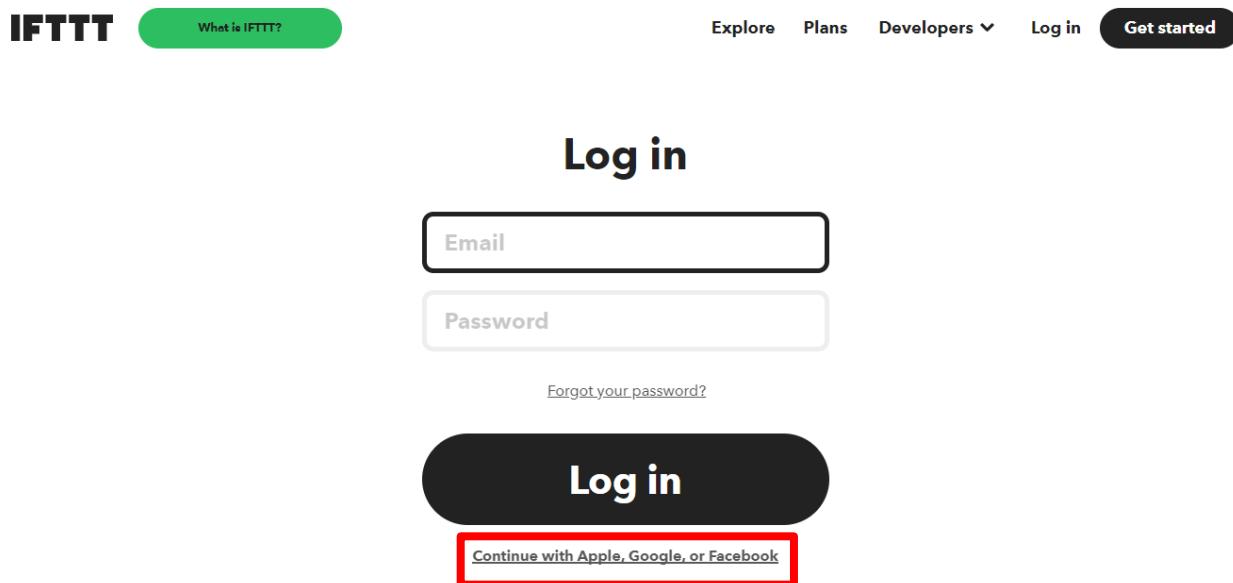
การตั้งค่า IFTTT

การสมัครและ login

ให้เข้าเว็บไซต์ <https://ifttt.com> จากนั้นให้กดปุ่ม Log in



กดที่ continue with apple, Google or Facebook



แนะนำ login ด้วย google เพื่อให้ง่ายต่อการใช้งานเชื่อมต่อกับ google drive



เข้าสู่ระบบ

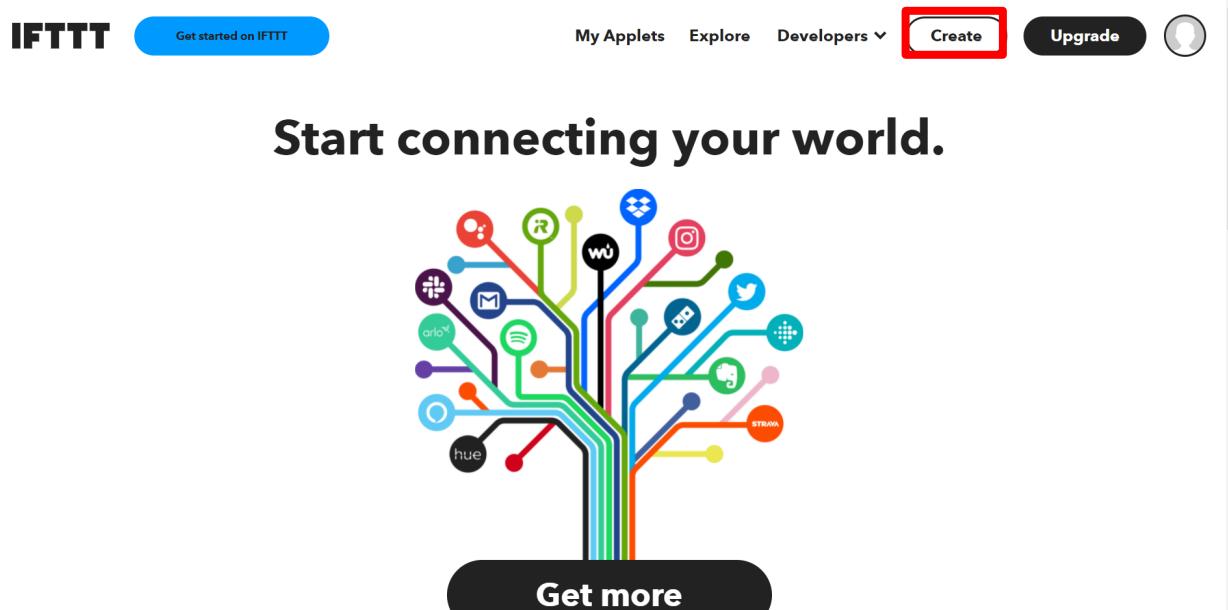
Sign up

Get started

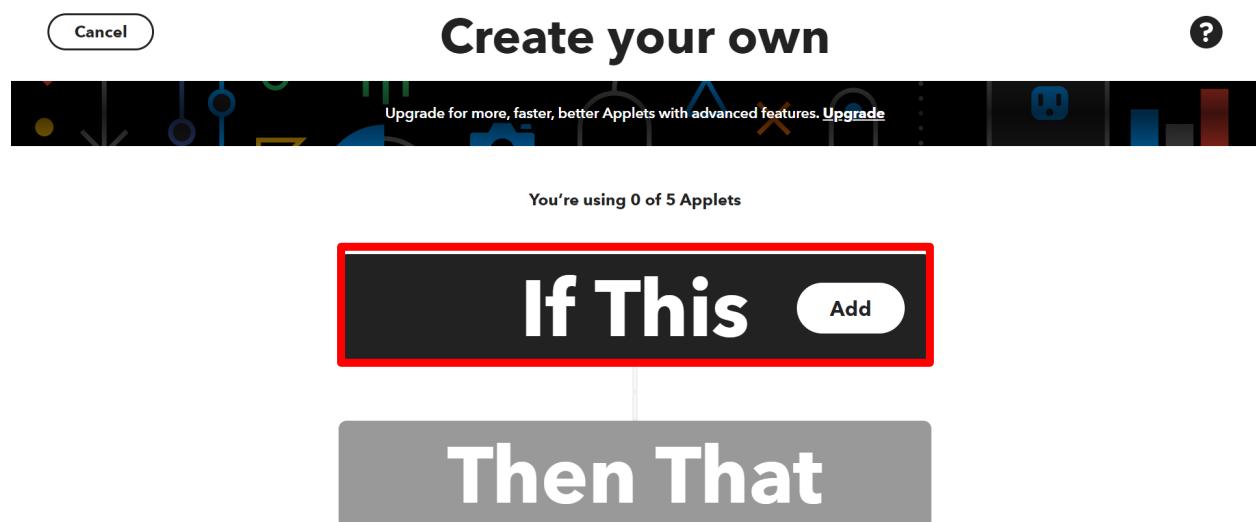
[Continue with Apple, Google, or Facebook](#)

เริ่มใช้งาน IFTTT

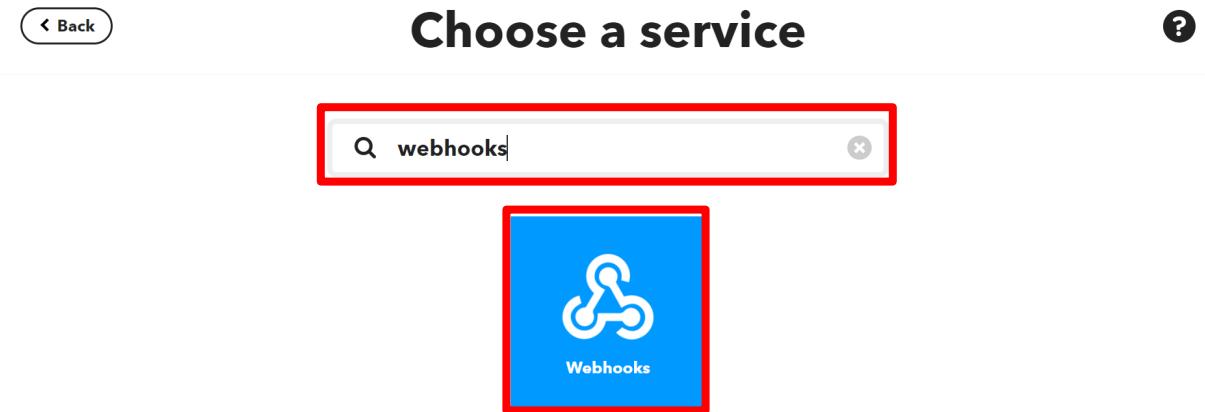
1. กดที่ Create



2. กดที่ If This



3. พิมพ์ webhooks ตรงช่องค้นหา และเลือก webhooks



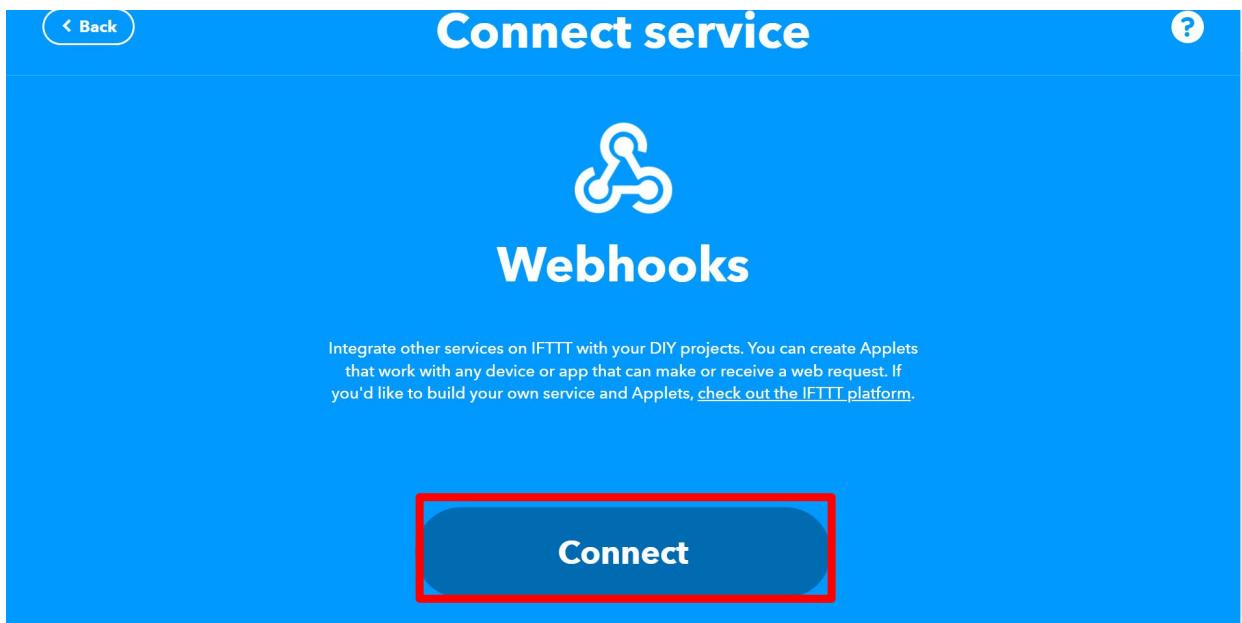
4. เลือก Recevie a web request

Webhooks

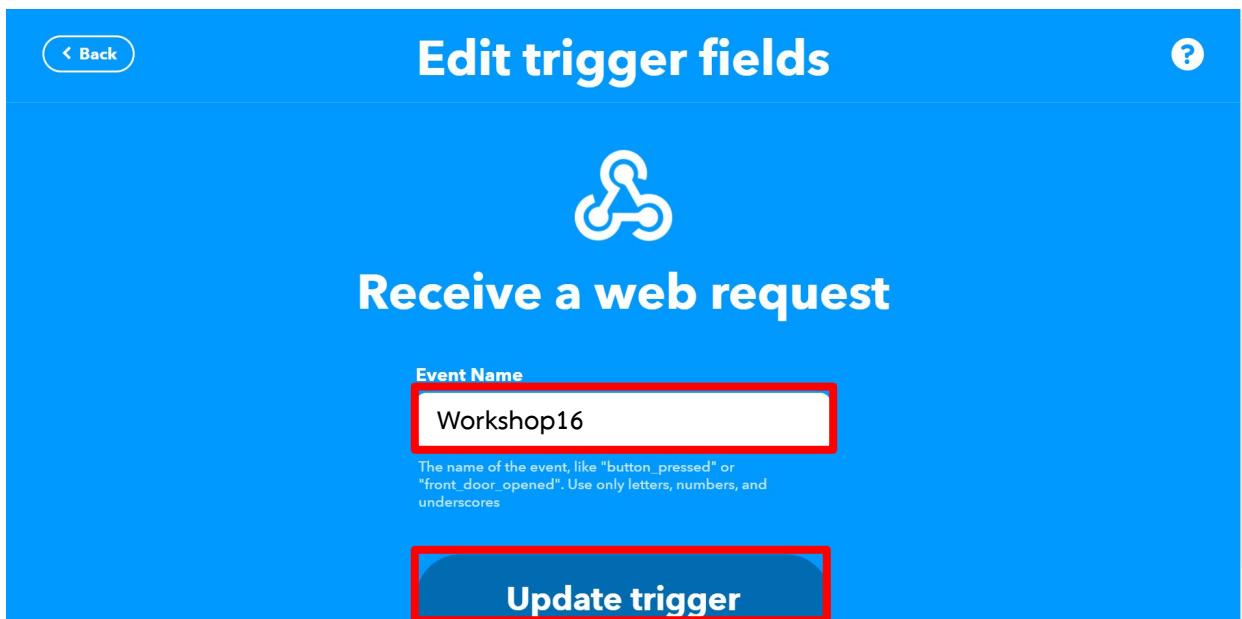
Receive a web request with a JSON payload
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Receive a web request
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

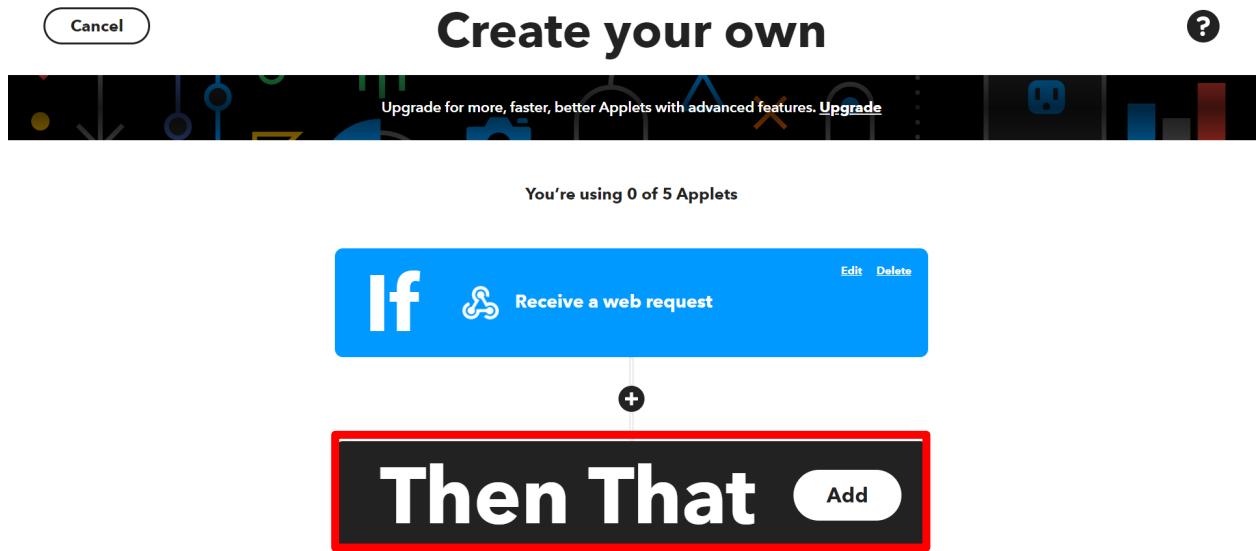
5. กด Connect เพื่อเชื่อมต่อกับ webhooks



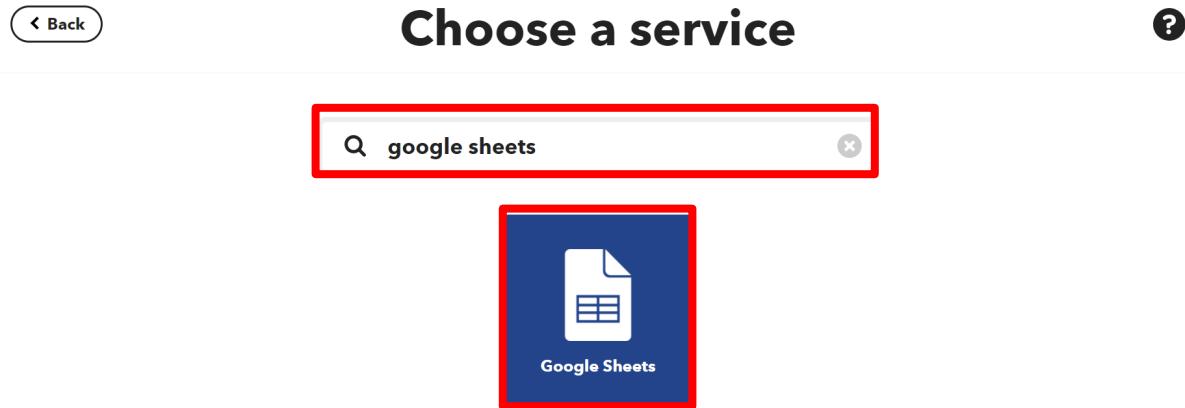
6. ตั้งชื่อ Event Name ว่า Workshop16 แล้วกด Create หรือ Update trigger



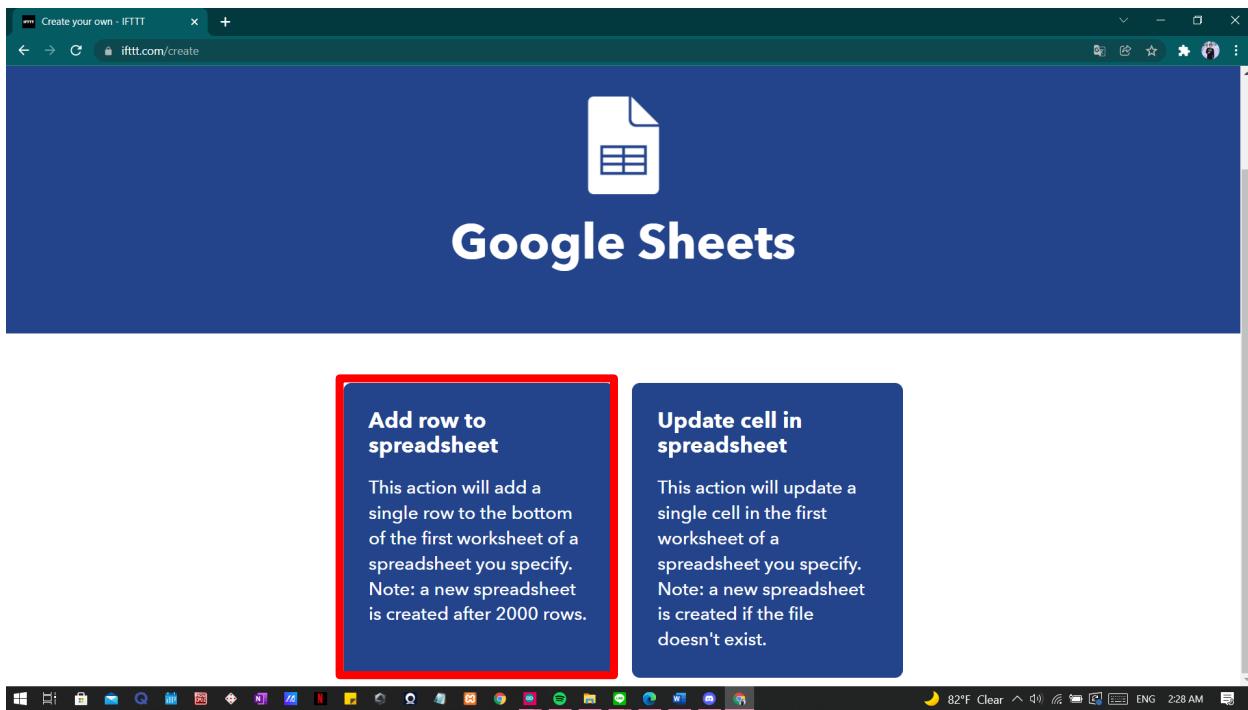
7. กด Then That เพื่อเชื่อมต่อกับ Google sheet



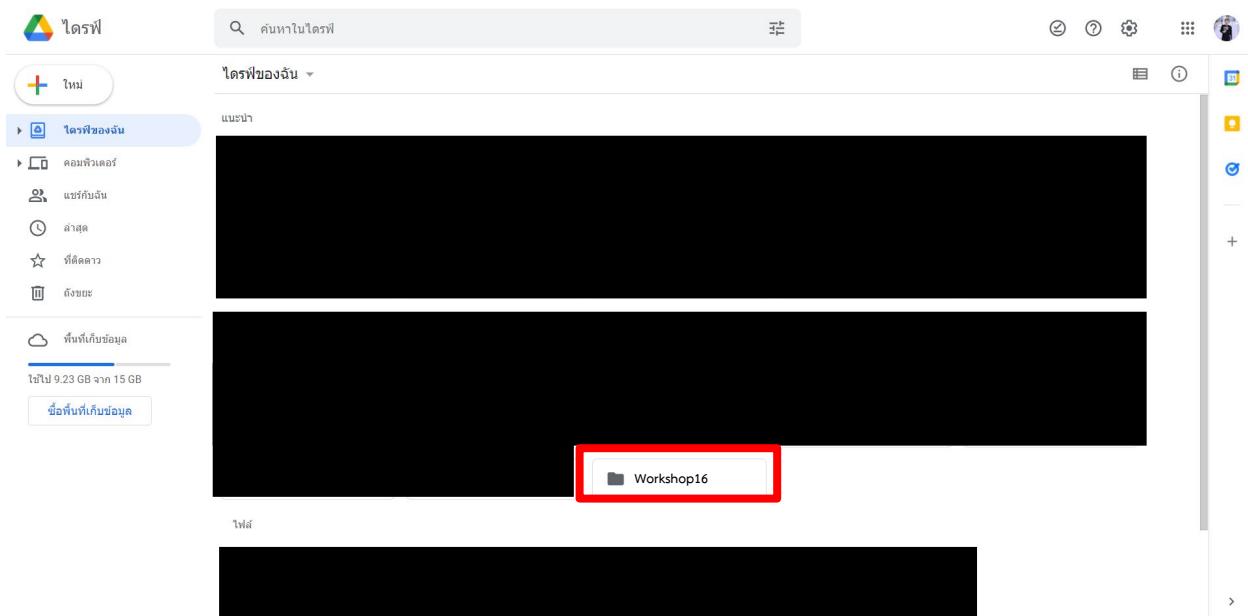
8. พิมพ์ google sheets และเลือก google sheets



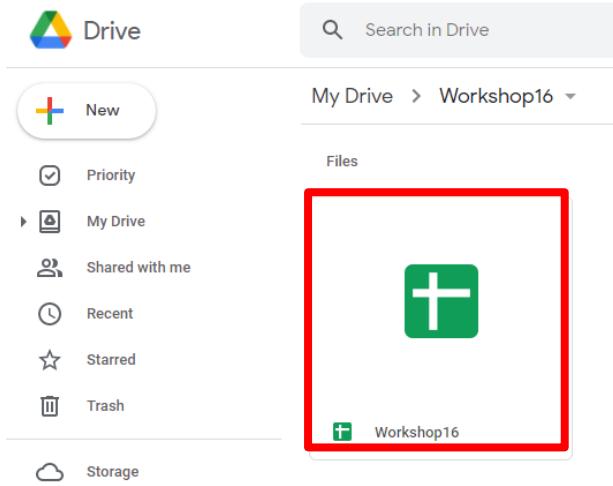
9. กดเลือก Add row to spreadsheet



10. สร้าง Folder ชื่อ Workshop16 เพื่อที่จะใช้เก็บ google sheet ใน google drive (แนะนำใช้ email
เดียวกันกับ email ที่ใช้สมัคร IFTTT)



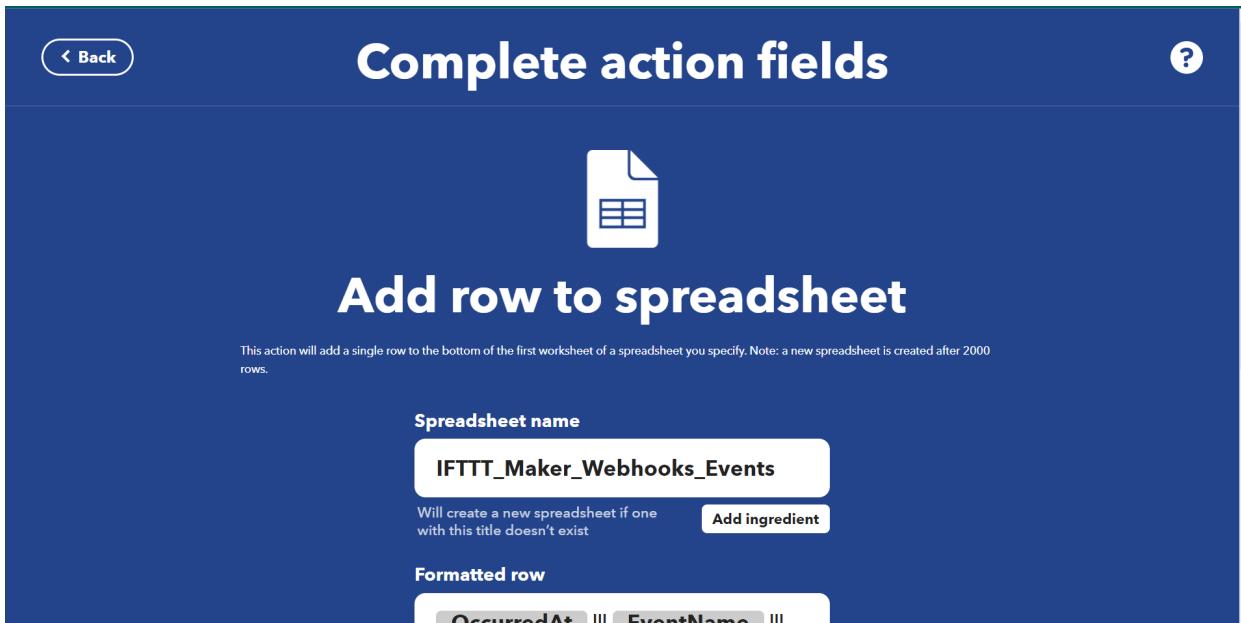
11. สร้าง google sheets Workshop16 ภายใน Folder ที่สร้างขึ้นมา



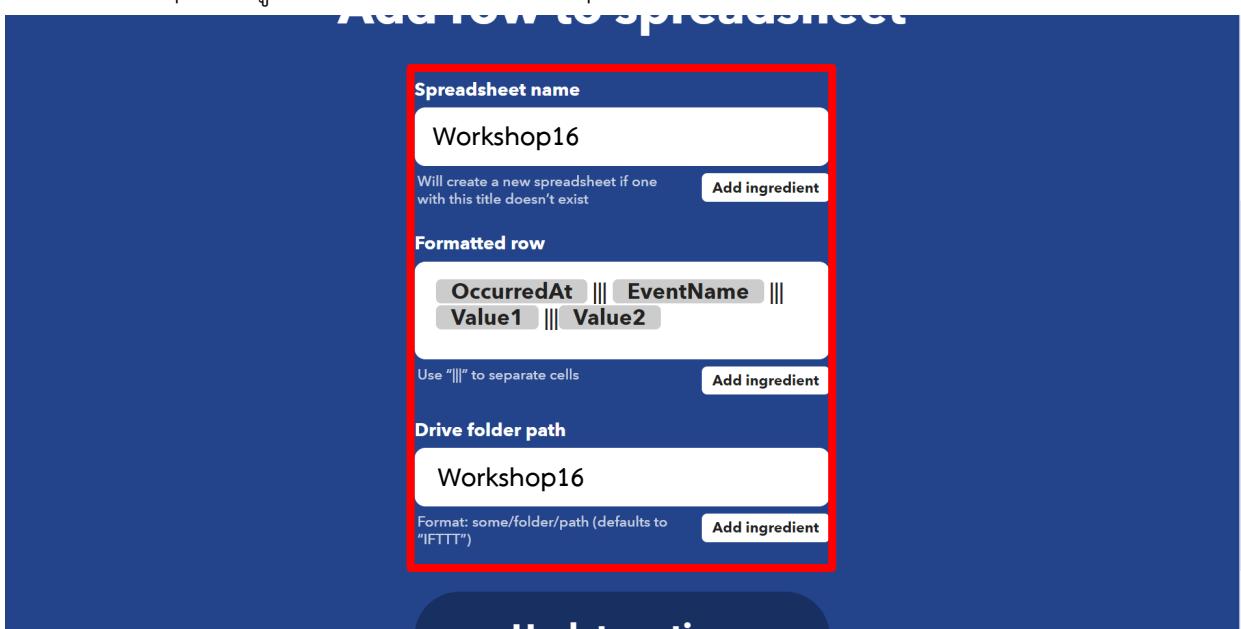
12. ทดสอบเปิด google sheets

The screenshot shows a Google Sheets document titled 'Workshop16'. The spreadsheet has 27 rows (1-27) and 14 columns (A-N). The first row (A1) contains some Thai text: 'ไฟล์ แก้ไข ดู แทรก รูปแบบ ชุดข้อมูล เครื่องมือ ស่วนขยาย ความช่วยเหลือ ແກ່ໄຟລົງຄວາສອນເພື່ອຕັ້ງຄ່າມາດ'. The rest of the cells are empty. The top menu bar includes options like 'File', 'Edit', 'View', 'Format', 'Tools', 'Data', 'Insert', 'Formulas', 'Script', and 'Help'. The bottom navigation bar shows the sheet name 'work1'.

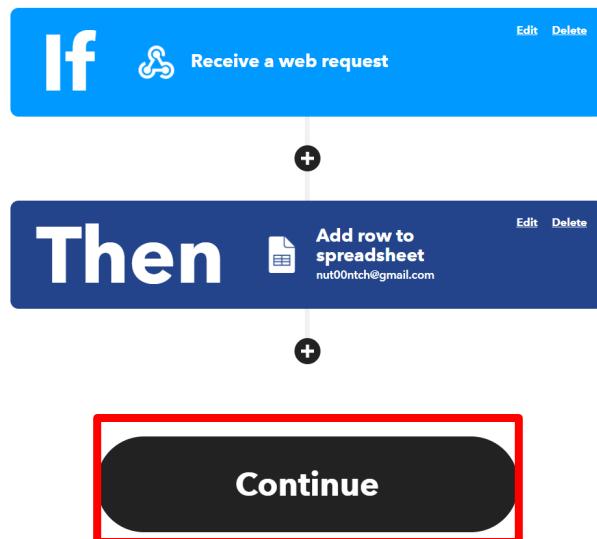
13. หน้าการตั้งค่า google sheets บน IFTTT



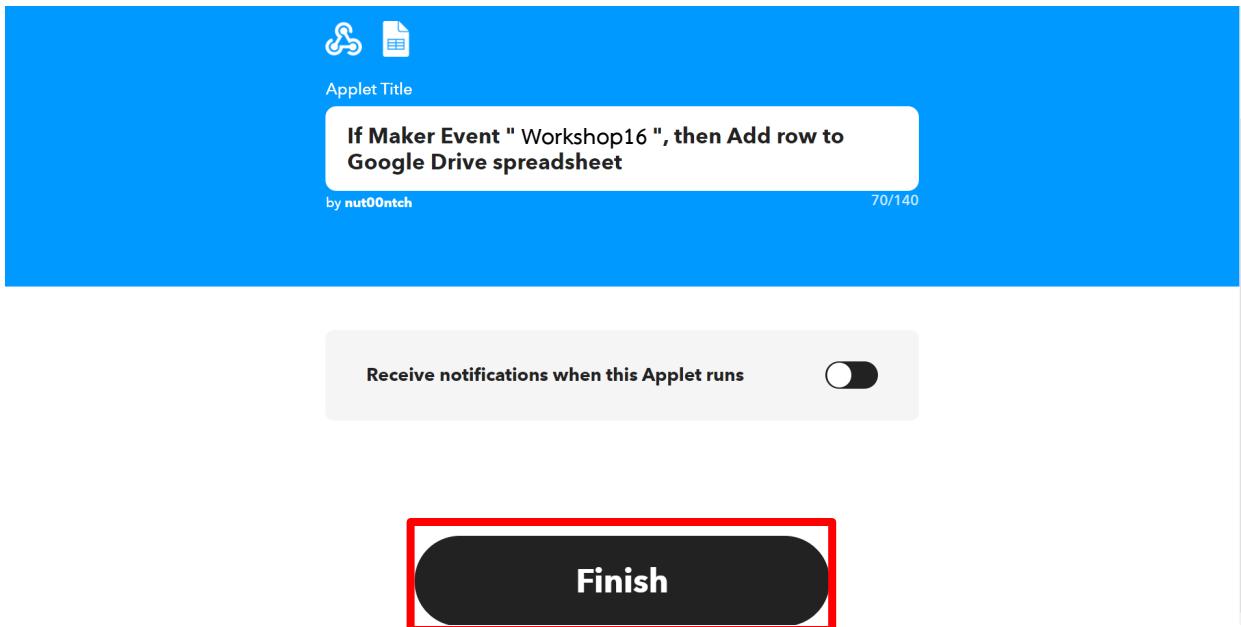
14. ตั้งค่าต่าง ๆ ตามดังรูป จากนั้นกด create หรือ update action



15. กด continue



16. กด Finish



17. หลังจากตั้งค่าเสร็จจะได้ดังรูป

The screenshot shows the IFTTT web interface for a specific applet. At the top, there are navigation links: 'Get started on IFTTT', 'My Applets', 'Explore', 'Developers', 'Create', 'Upgrade', and a user profile icon. Below these are buttons for 'Back' and 'Settings'. The main content area features two icons: a blue gear and a white document with a grid. The title of the applet is 'If Maker Event "Workshop16", then Add row to Google Drive spreadsheet'. Below the title is a link 'Edit title' and the author's name 'by nut00ntch'. A large, prominent button at the bottom center says 'Connected' with a blue circular indicator to its right.

การทดสอบ

1. กดไปที่สัญญาลักษณ์ของ webhook

This screenshot is identical to the one above, showing the same IFTTT applet details. However, there is a red rectangular box drawn over the blue gear icon, highlighting it. The rest of the interface, including the title, author, and 'Connected' status button, remains the same.

2. กด Documentation

The screenshot shows the IFTTT platform interface. At the top, there's a navigation bar with links for 'My Applets', 'Explore', 'Developers', 'Create', 'Upgrade', and a user profile icon. Below the navigation is a large blue header section with the IFTTT logo and the text 'Webhooks integrations'. A sub-instruction reads: 'Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform.' Below this, there are two buttons: 'Create' and 'Documentation', with 'Documentation' being highlighted by a red box. At the bottom of the blue header, there are tabs for 'Applets', 'My Applets', and 'Details'.

3. พิมพ์ Workshop16 แทนคำว่า Event และ กำหนดค่าลงใน value1 และ value2

The screenshot shows the configuration page for a new webhook trigger. At the top, it says 'Your key is:' followed by a redacted key. Below that, there's a section titled 'To trigger an Event' with instructions to make a POST or GET web request to a URL containing '{event}' and a key. The URL is shown as: `https://maker.ifttt.com/trigger/{event}/with/key/bWZ8fVLsCKH7Bj49Ykba_h`. There's also a JSON body example: `{"value1": "____", "value2": "____", "value3": "____"}`, where the first two fields are redacted. A note states that this data is optional and can be passed as query parameters or form variables. Below this, there's a 'Test It' button.

4. เมื่อใส่ข้อมูลครบแล้ว ให้กด Test It

Your key is: [REDACTED]

To trigger an Event

Make a POST or GET web request to:

```
https://maker.ifttt.com/trigger/ Workshop16 /with/key/bWZ8fVLsCKH7Bj49Ykba_h
```

With an optional JSON body of:

```
{ "value1": "1", "value2": "2", "value3": "" }
```

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":"1","value2":"2"}' https://maker.ifttt.com/trigger/Workshop12/with/key/bWZ8fVLsCKH7Bj49Ykba_h
```

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

Test It

5. จะมีการแจ้งเตือน Event has been triggered

Event has been triggered.

Your key is: [REDACTED]

To trigger an Event

Make a POST or GET web request to:

```
https://maker.ifttt.com/trigger/ Workshop16 /with/key/bWZ8fVLsCKH7Bj49Ykba_h
```

With an optional JSON body of:

```
{ "value1": "1", "value2": "2", "value3": "" }
```

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":"1","value2":"2"}' https://maker.ifttt.com/trigger/Workshop12/with/key/bWZ8fVLsCKH7Bj49Ykba_h
```

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

Test It

To trigger an Event with an arbitrary JSON payload

6. เชื่อ google sheet เพื่อดูการเปลี่ยนแปลง

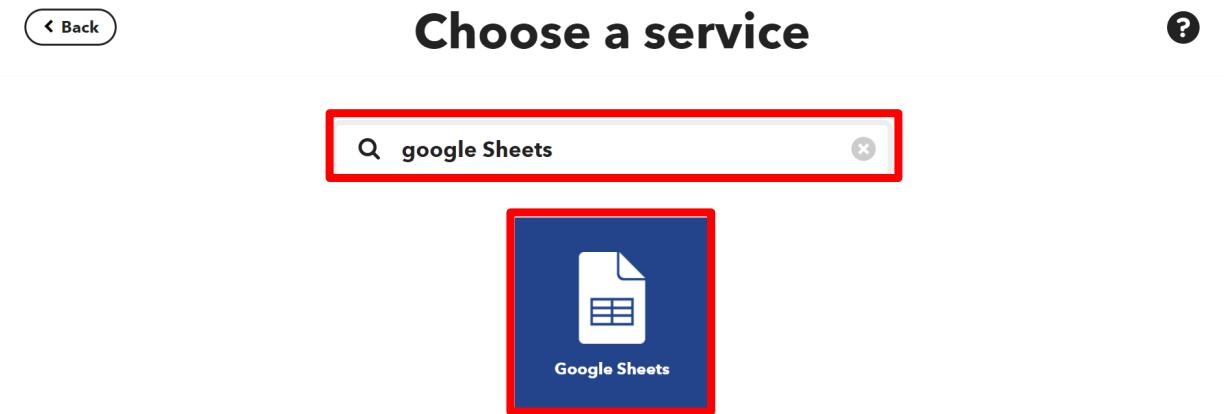
The screenshot shows a Google Sheets interface with the title 'Workshop16'. The first row has four cells: 'January 14, 2022 at 02:36AM', 'Workshop16', '1', and '2'. The '1' cell is highlighted with a red box. The rest of the sheet is blank with rows numbered from 1 to 27.

การเพิ่มการแจ้งเตือนไปที่ Line

1. กด Create

The screenshot shows the IFTTT website with the 'Create' button highlighted with a red box. The page displays the heading 'My Applets' and a search/filter bar. Below it, there are tabs for 'All (1 of 5)', 'Published', and 'Archive', along with a link to 'Get Pro to get 20 Applets'. A single applet card is visible, titled 'If Maker Event "Workshop16", then Add row to Google Drive spreadsheet' by 'nut00ntch'.

2. พิมพ์ google sheets และกดเลือก google sheet



3. เลือก New row added to spreadsheet

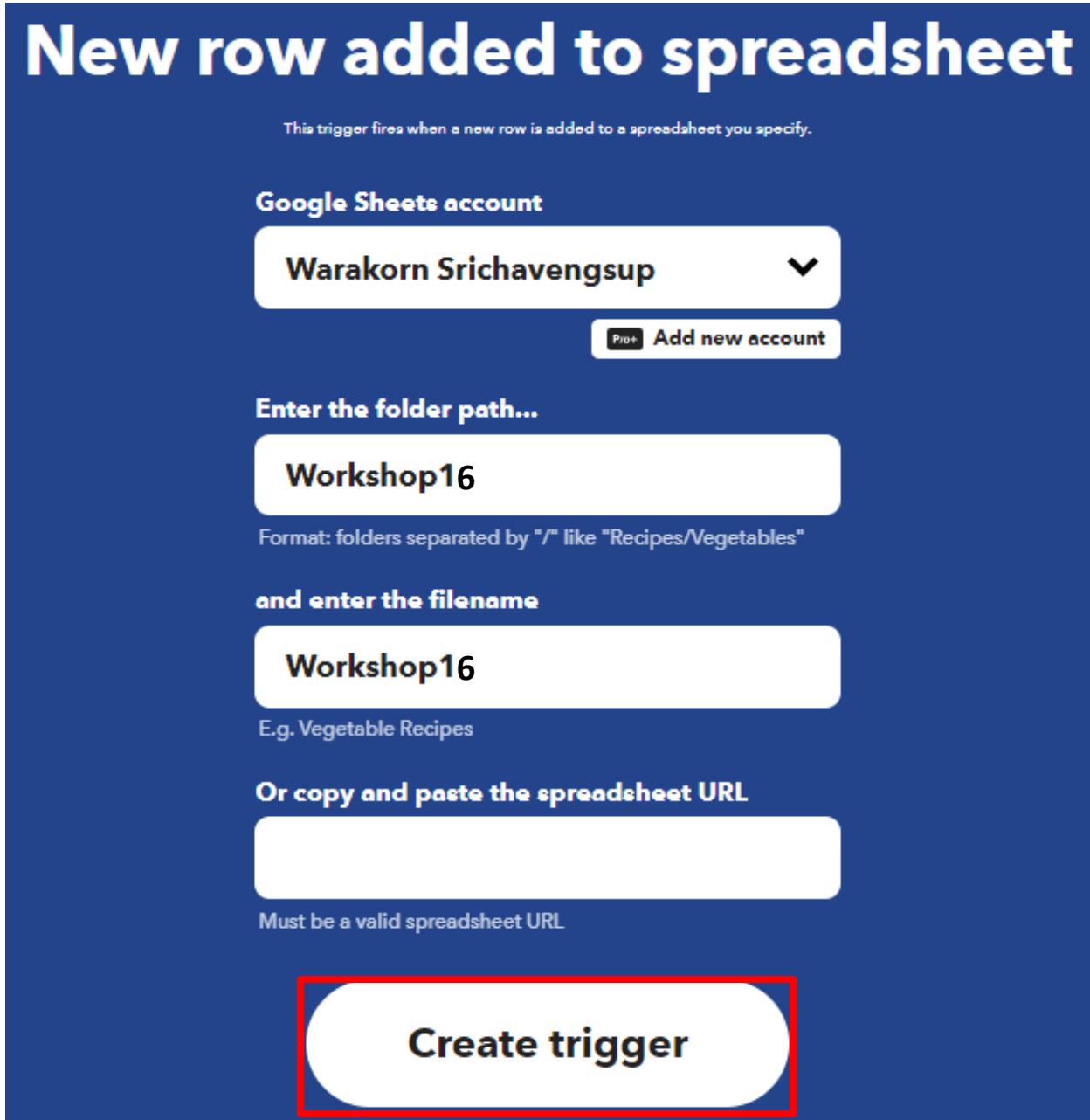
New spreadsheet added to folder
This trigger fires when a new spreadsheet is added to a Google Drive folder you specify. Note: only works for spreadsheets created after the Applet turned on.

New worksheet in spreadsheet
This trigger fires when a new worksheet is added to a spreadsheet you specify.

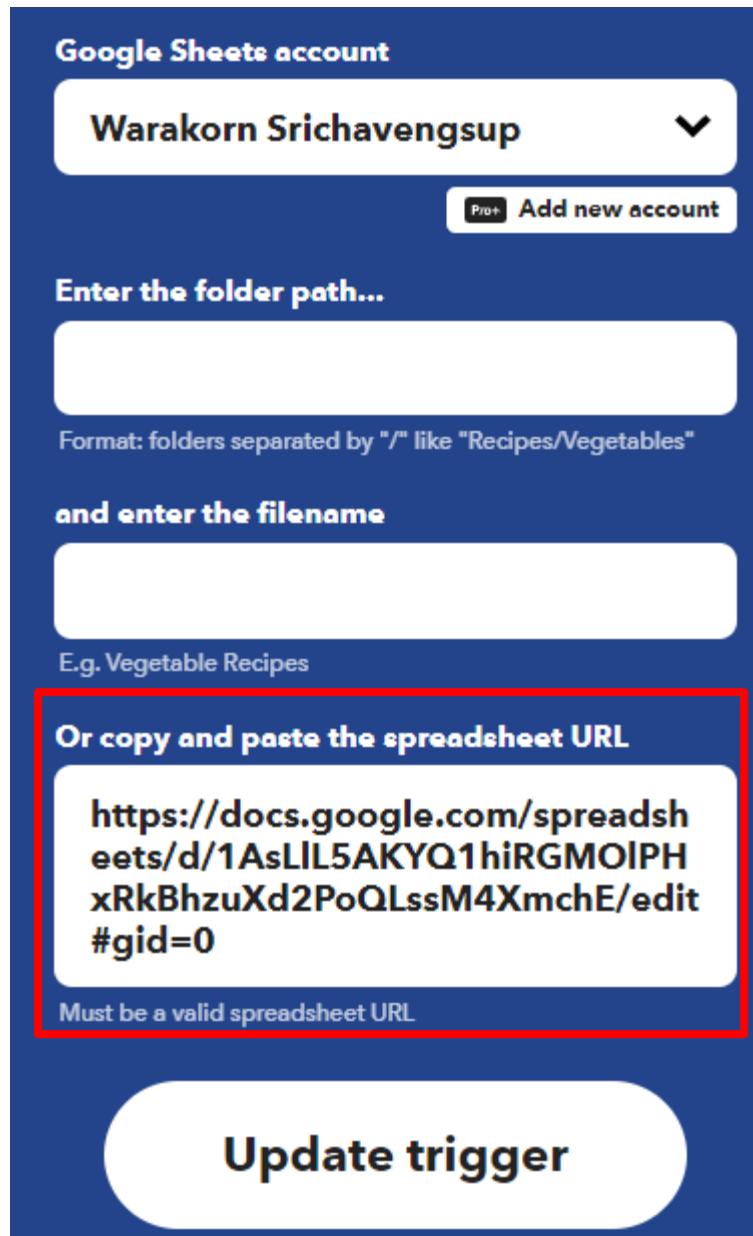
New row added to spreadsheet
This trigger fires when a new row is added to a spreadsheet you specify.

Cell updated in spreadsheet
This trigger fires when a particular cell is updated within the spreadsheet you specify.

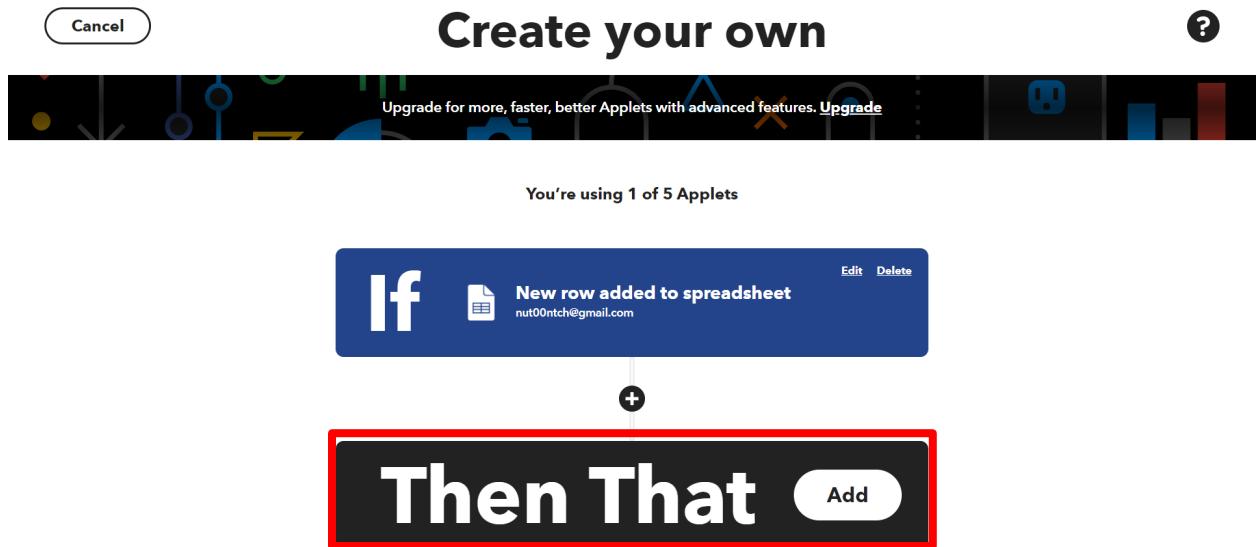
4. ใส่ชื่อ URL ไปตามดังรูป (URL ของ google sheet สามารถ copy จากหน้า sheet ได้เลย) แล้วกด Create trigger หรือ update



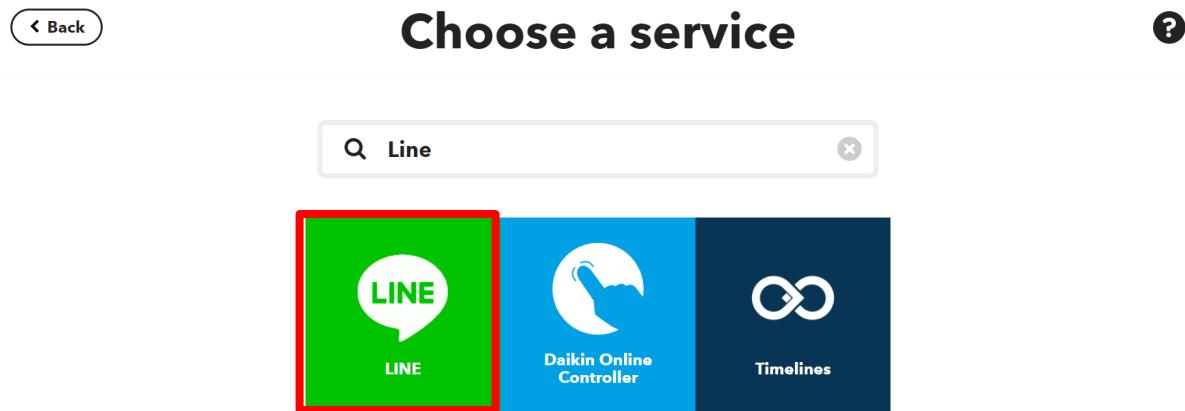
เราสามารถใส่เฉพาะ Spreadsheet URL ซึ่งจะให้ผลลัพธ์เหมือนกัน



5. กดเลือก Then That



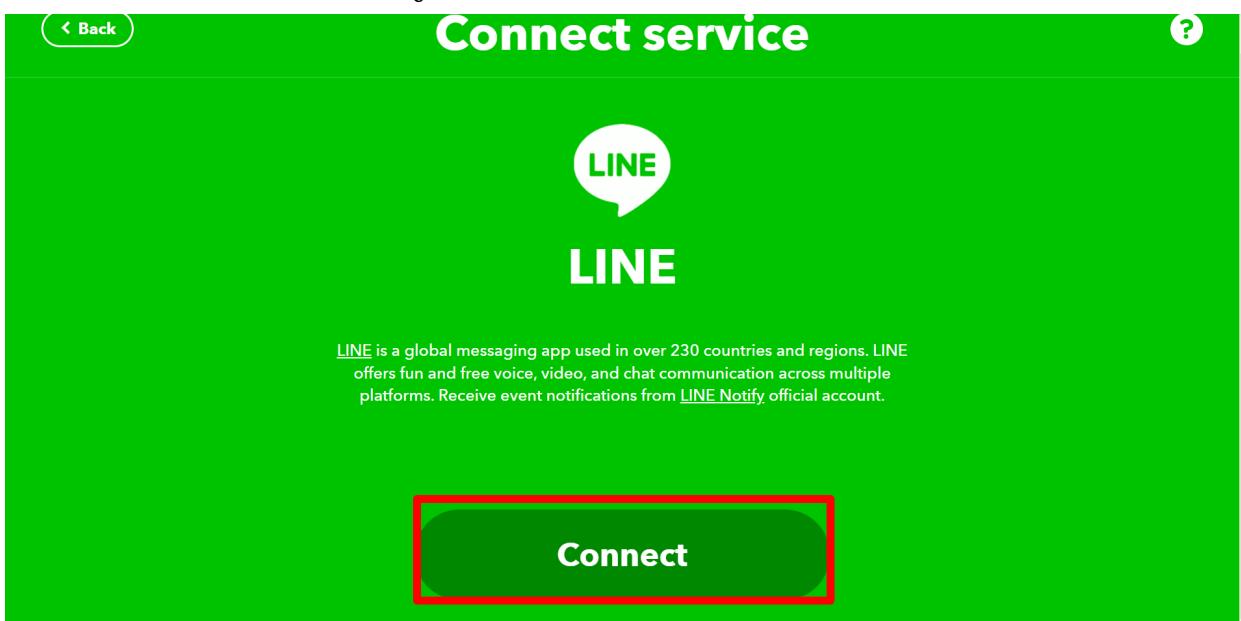
6. พิมพ์ Line แล้วกดเลือก Line



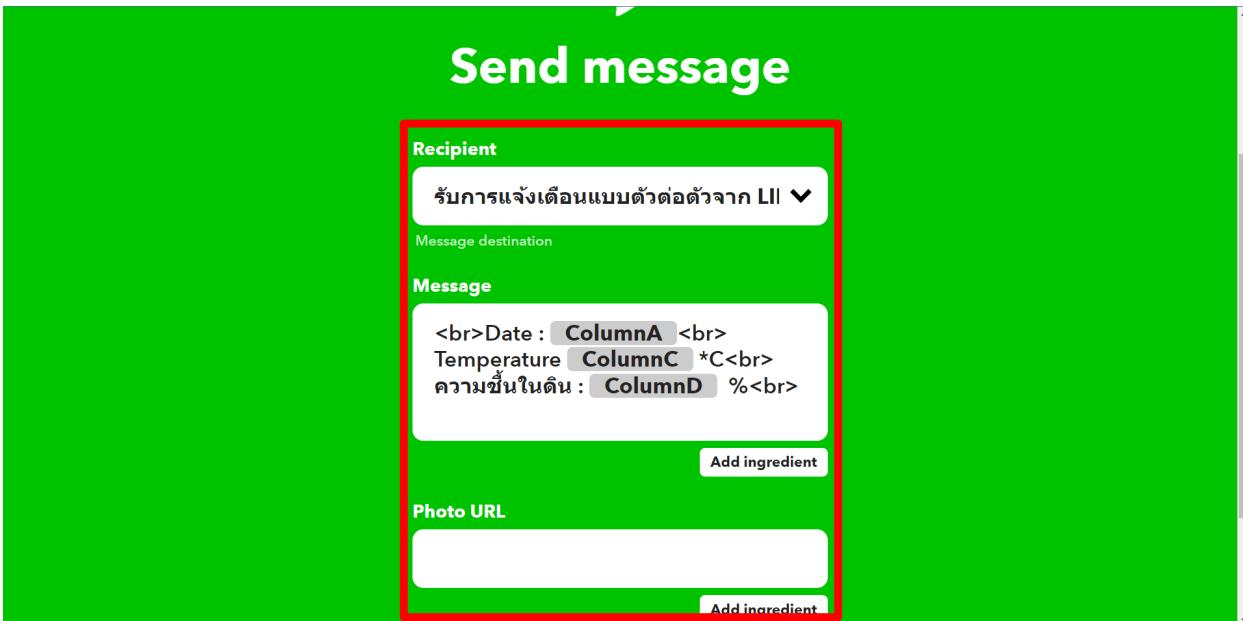
7. เลือก send message



8. กด connect เพื่อเชื่อมต่อไปที่ บัญชี LINE ของตนเอง



9. ใส่ข้อมูลง่ายตามดังรูป



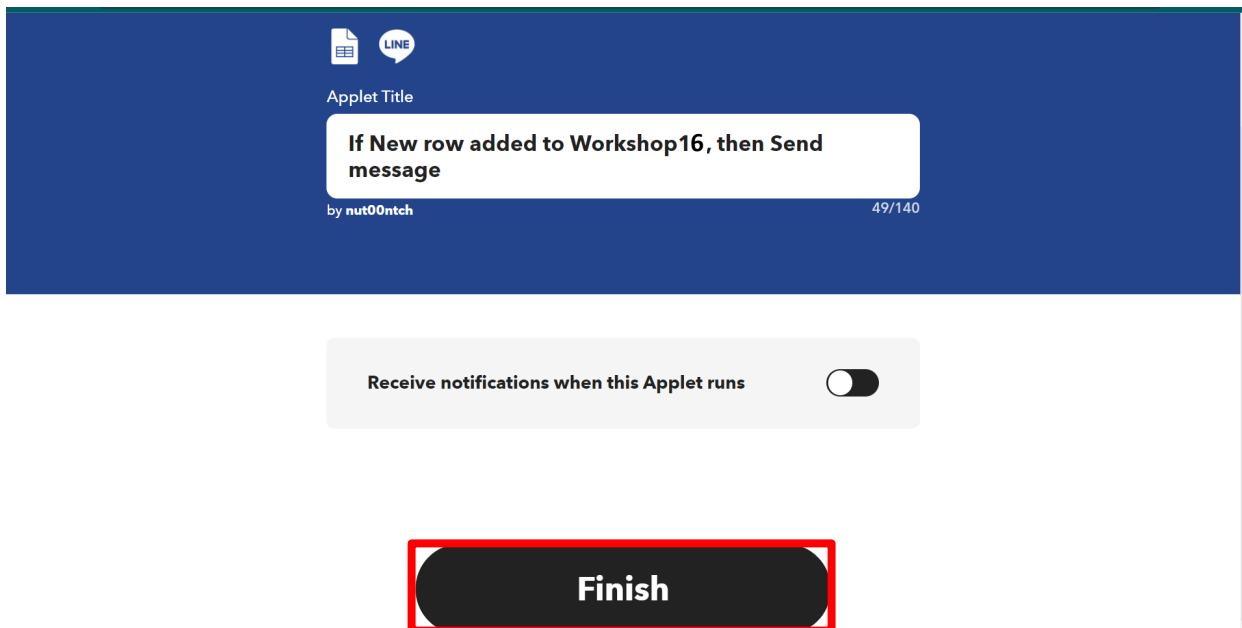
10. กด create หรือ update



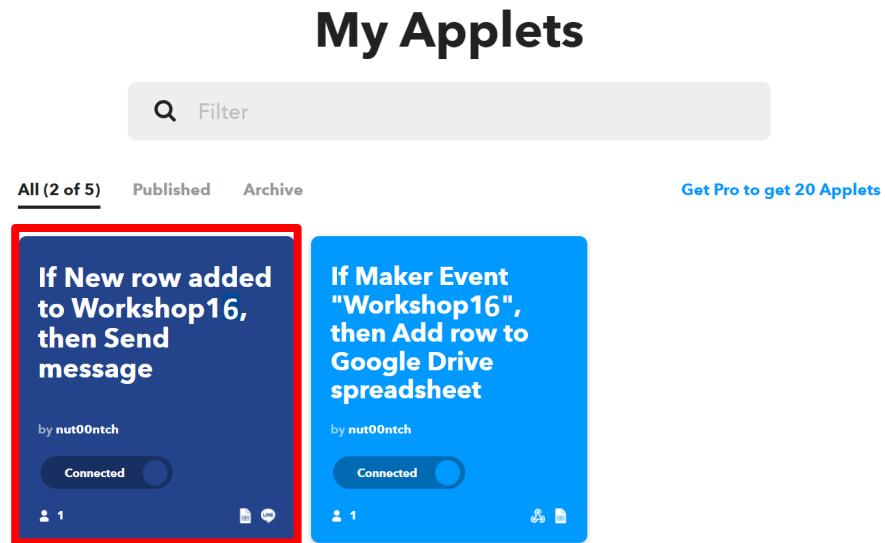
11. กด continue



12. กด Finish



13. เมื่อตั้งค่าทุกอย่างเสร็จจะได้ดังรูป



15. เพิ่ม Line Notify เป็นเพื่อนใน Line โดยการสแกน QR Code รูปด้านล่าง ก่อนนำไปใช้สำหรับการแจ้งเตือน line Workshop IFTTT



ทดสอบการทำงาน

- กดเลือก Applets ของ IFTTT เชื่อมกับ google sheet

My Applets

All (2 of 5) Published Archive Get Pro to get 20 Applets

If New row added to Workshop16, then Send message by nut00ntch Connected 1 1

If Maker Event "Workshop16", then Add row to Google Drive spreadsheet by nut00ntch Connected 1 1

- กดไปที่สัญญาลักษณ์ของ webhook

IFTTT Get started on IFTTT My Applets Explore Developers Create Upgrade

◀ Back ⚙ Settings

If Maker Event "Workshop16", then Add row to Google Drive spreadsheet by nut00ntch Edit title

Connected

<https://ifttt.com/applets/wrabZq3/edit>

3. กดเลือก Documentation

The screenshot shows the IFTTT platform's "Webhooks integrations" page. At the top, there are navigation links: "Get started on IFTTT", "My Applets", "Explore", "Developers", "Create", "Upgrade", and a user profile icon. Below these are "Back" and "Settings" buttons. The main heading is "Webhooks integrations". A sub-instruction reads: "Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform." There are two primary buttons at the bottom: "Create" and "Documentation", with "Documentation" being highlighted by a red box.

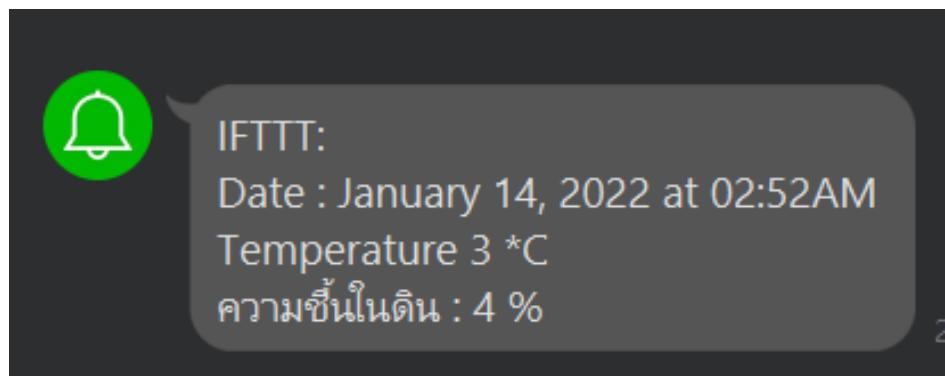
4. ใส่ข้อมูลตามดังรูป และกด Test it

The screenshot shows the "To trigger an Event" configuration page. It starts with a green header bar stating "Event has been triggered.". Below it, the "Your key is:" field contains a placeholder text followed by a blacked-out key. A "Back to service" link is visible. The main content area is titled "To trigger an Event" and includes instructions: "Make a POST or GET web request to: https://maker.ifttt.com/trigger/Workshop1/with/key/bWZ8fLsCKH7Bj49Ykba_h". An optional JSON body field contains the following code: {"value1": "3", "value2": "4", "value3": ""}. A note below states: "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." A "curl" command example is provided: curl -X POST -H "Content-Type: application/json" -d '{"value1":"3","value2":"4"}' https://maker.ifttt.com/trigger/Workshop12/with/key/bWZ8fLsCKH7Bj49Ykba_h. A "Test It" button is present, and a note at the bottom says: "Please read our FAQ on using Webhooks for more info." The "Test It" button is highlighted with a red box.

5. เช็คข้อมูลที่ Google Sheets

1	January 14, 2022 at 02:36AM	Workshop16	1	2
2	January 14, 2022 at 02:47AM	Workshop16	1	2
3	January 14, 2022 at 02:48AM	'Workshop16	3	4

6. เช็คข้อมูลที่แจ้งเตือนไปที่ Line



การเพิ่มการแจ้งเตือนไปที่ Email

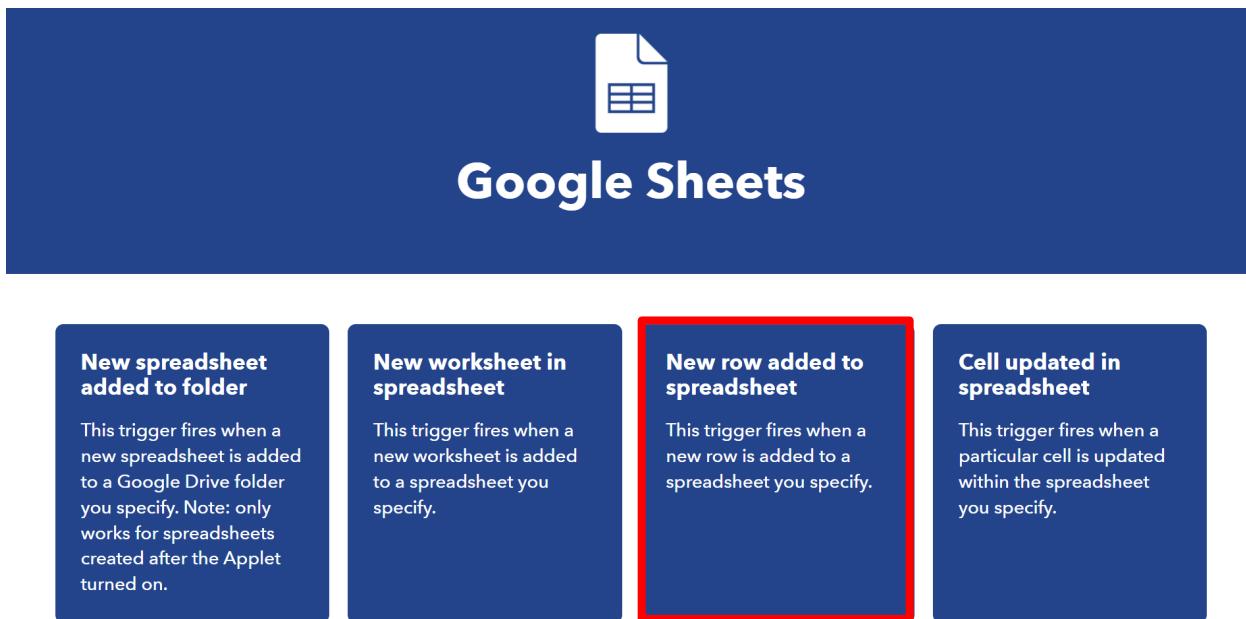
1. กด Create

The screenshot shows the IFTTT interface. At the top, there's a navigation bar with 'My Applets', 'Explore', 'Developers', a 'Create' button (which is highlighted with a red box), 'Upgrade', and a user profile icon. Below the navigation is the title 'My Applets'. Underneath it is a search bar with a magnifying glass icon and the word 'Filter'. Further down are three tabs: 'All (1 of 5)', 'Published', and 'Archive'. To the right of these tabs is a link 'Get Pro to get 20 Applets'. The main content area displays one applet card, which has a blue background and white text. The card reads: 'If Maker Event "Workshop16", then Add row to Google Drive spreadsheet' by 'nut00ntch'. There's also a 'Comment' button at the bottom of the card.

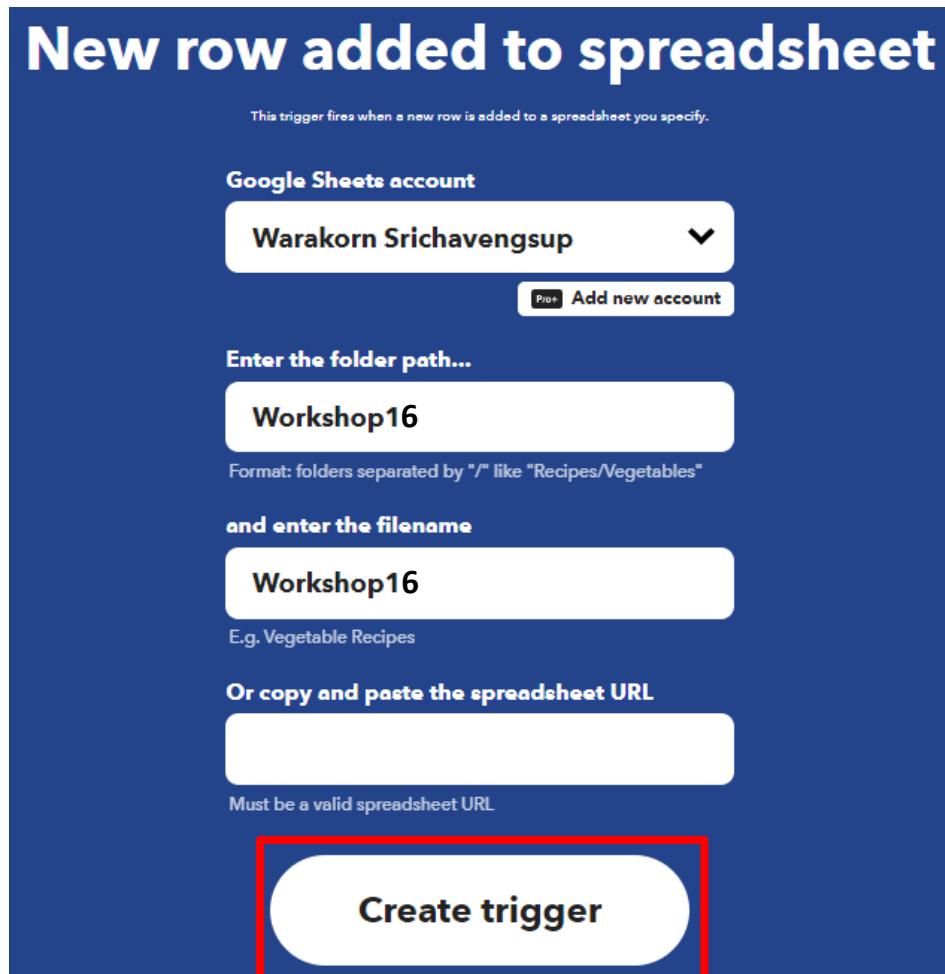
2. พิมพ์ google sheets และกดเลือก google sheet

The screenshot shows the 'Choose a service' step in IFTTT. At the top, there's a back button and a help icon. The main title is 'Choose a service'. Below the title is a search bar with the text 'google Sheets' and a clear button. Underneath the search bar is a large blue square icon for 'Google Sheets', featuring a white document icon with a grid pattern. The entire search bar area is highlighted with a red box.

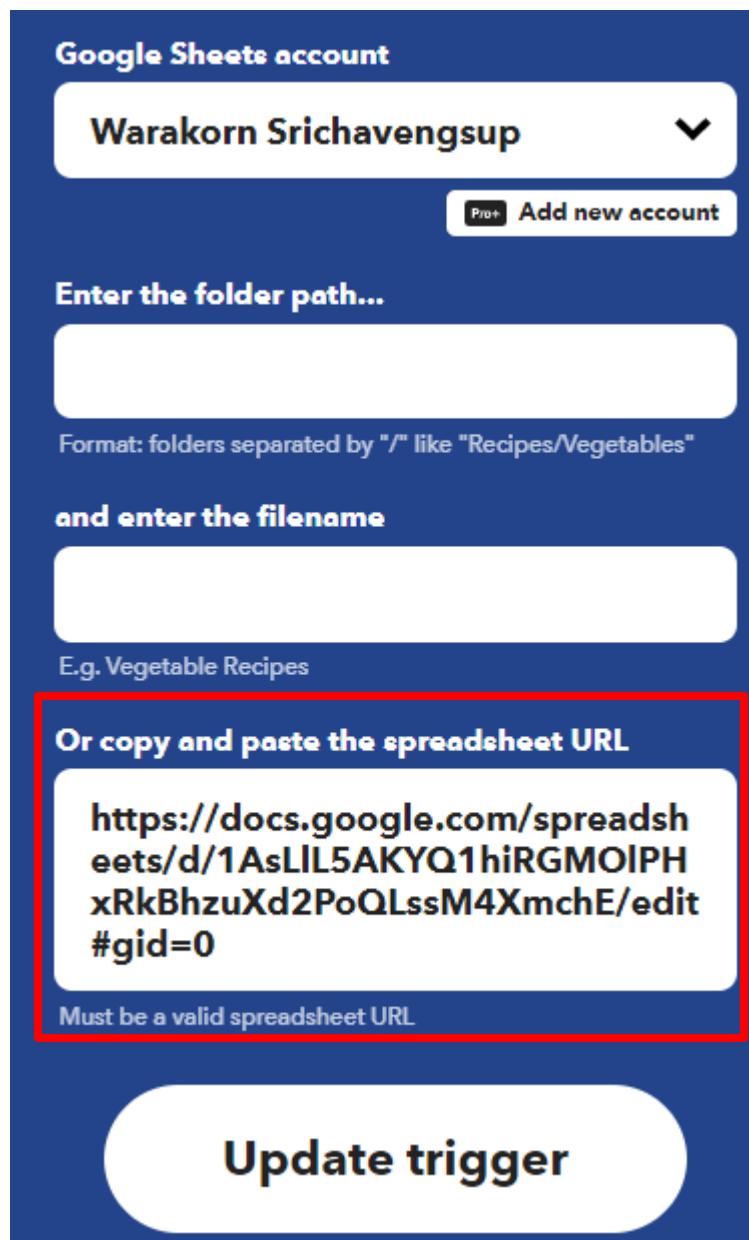
3. เลือก New row added to spreadsheet



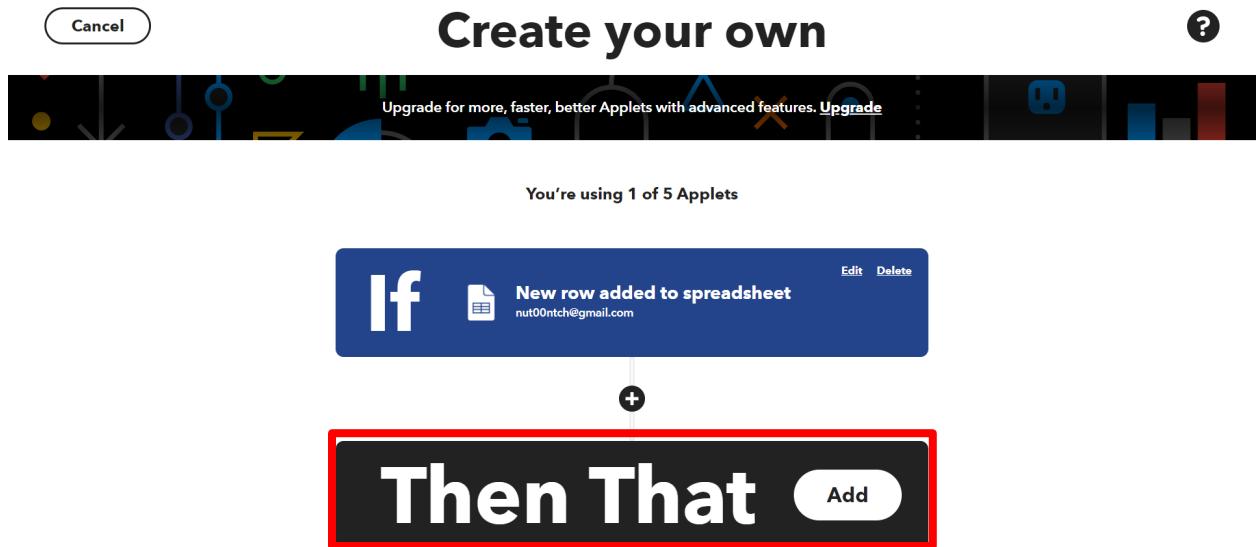
4. ใส่ชื่อคลังไปตามดังรูป แล้วกด Create trigger หรือ update



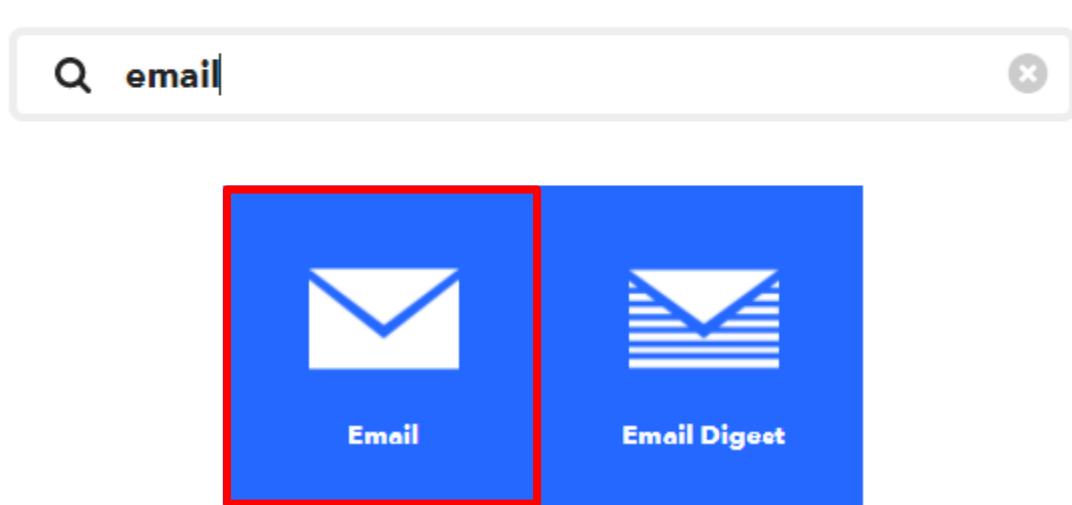
เราสามารถใส่เฉพาะ Spreadsheet URL ซึ่งจะให้ผลลัพธ์เหมือนกัน



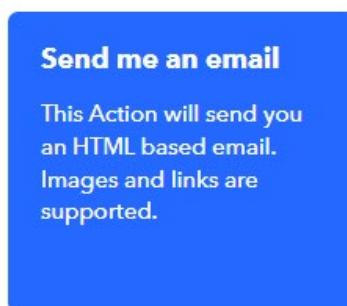
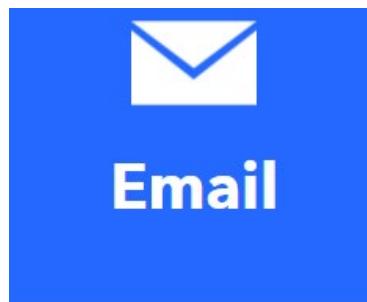
5. กดเลือก Then That



6. ค้นหา email และกดเลือก Email



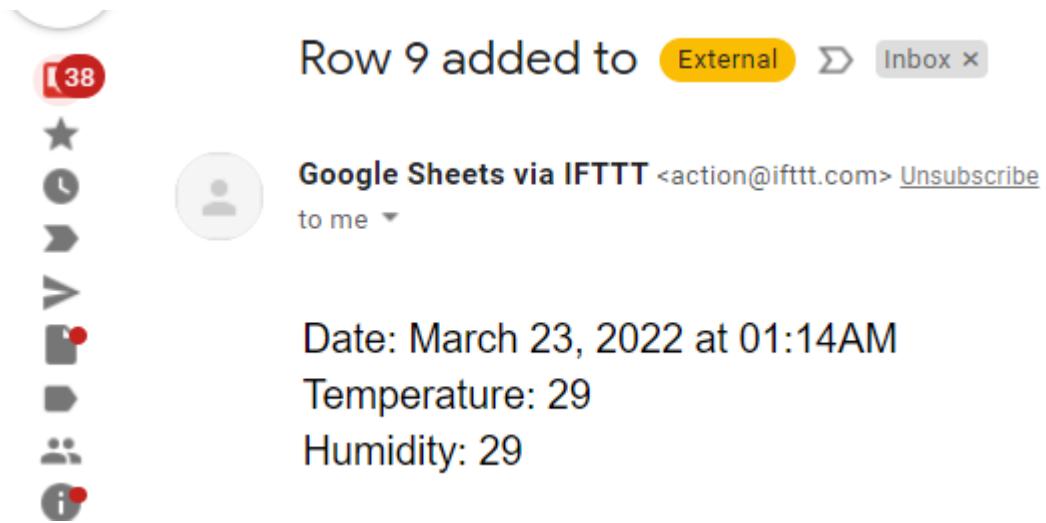
7. เลือก Send me an Email



8. กรอกข้อมูลดังภาพ



9. เมื่อมีการป้อนข้อมูลเข้า spreadsheet ให้ไปตรวจสอบที่กล่อง Inbox ของ Email



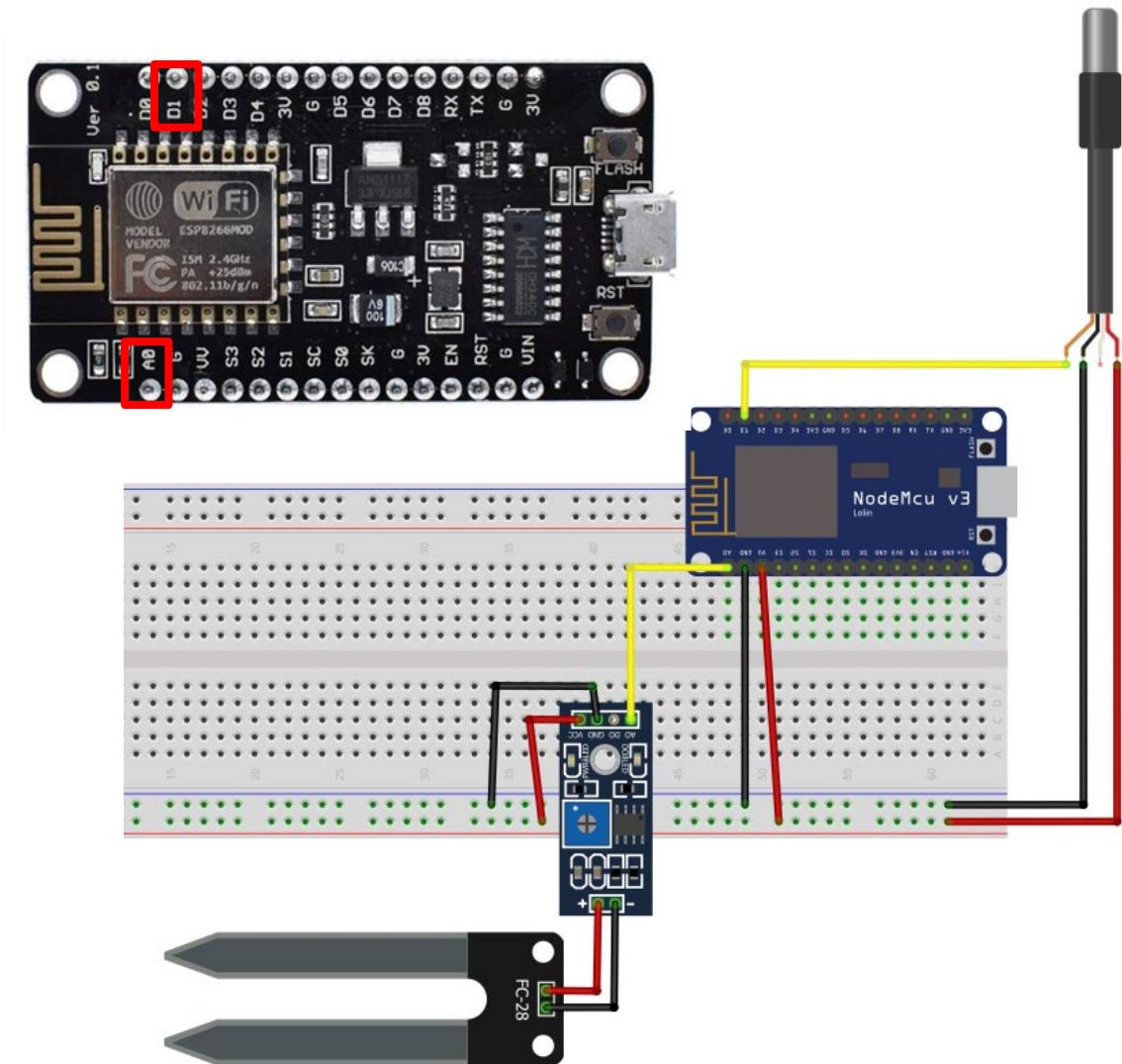
การใช้งานร่วมกับ NodeMCU esp8266

อุปกรณ์ที่ต้องใช้งาน

1. NodeMCU ESP8266
2. Soil Moisture Sensor
3. DS018B20

การต่อวงจร

PIN	อุปกรณ์
A0	Soil Moisture Sensor
D1	DS018B20



การเขียน Code (กรอบสีแดงคือต้องเปลี่ยนเป็นข้อมูลของตัวเอง)

Workshop_IFTTT

```
#include <ESP8266WiFi.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 5 //กำหนดขาที่จะเชื่อมต่อ Sensor
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// Set WiFi credentials
#define WIFI_SSID "your wifi"
#define WIFI_PASS "your password"

// Set IFTTT Webhooks event name and key
#define IFTTT_Key "your key"
#define IFTTT_Event "your event"

WiFiClient client;
int analogPin = A0; //ประกาศตัวแปร ให้ analogPin แทนขา analog ขาที่5
int val = 0;
int map_val = 0;
float temp = 0;
float IFTTT_Value1 = 0;
float IFTTT_Value2 = 0;

void setup() {
    Serial.begin(115200); // Serial output only for information, you can also remove all Serial commands
    WiFi.begin(WIFI_SSID, WIFI_PASS);
    // Connecting to WiFi...
    Serial.print("Connecting to ");
    Serial.print(WIFI_SSID);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(100);
        Serial.print(".");
    }
    // Connected to WiFi
    Serial.println();
    Serial.print("Connected! IP address: ");
    Serial.println(WiFi.localIP());
    sensors.begin();
}

}
```

```

void loop() {
    Serial.println("Requesting temperatures...");
    sensors.requestTemperatures(); // อ่านข้อมูลจาก library
    temp = sensors.getTempCByIndex(0);
    Serial.print("Temperature is: ");
    Serial.print(temp); // แสดงค่า อุณหภูมิ
    Serial.println(" *C");
    IFTTT_Value1 = temp;

    val = analogRead(analogPin); // อ่านค่าสัญญาณ analog ขา5 ที่ต่อ กับ Soil Moisture Sensor Module
    map_val = map(val, 0, 1023, 100, 0); // ปรับเปลี่ยนค่าจาก 0-1024 เป็น 0-100
    Serial.print("val = "); // พิมพ์ชื่อความสั่งเข้าคอมพิวเตอร์ "val = "
    Serial.println(map_val); // พิมพ์ค่าของตัวแปร val
    IFTTT_Value2 = map_val;

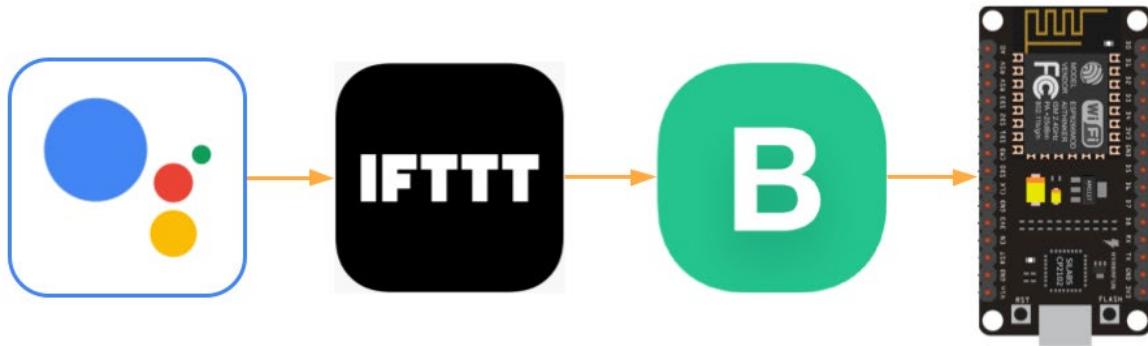
    // Send Webook to IFTTT
    send_webhook();
    delay(60000);
}

void send_webhook(){// function Send Webook to IFTTT
    // construct the JSON payload
    String jsonString = "";
    jsonString += "{\"value1\":\"";
    jsonString += IFTTT_Value1;
    jsonString += "\",\"value2\":\"";
    jsonString += IFTTT_Value2;
    jsonString += "\}";
    int jsonLength = jsonString.length();
    String lenString = String(jsonLength);
    // connect to the Maker event server
    client.connect("maker.ifttt.com", 80);
    // construct the POST request
    String postString = "";
    postString += "POST /trigger/";
    postString += IFTTT_Event;
    postString += "/with/key/";
    postString += IFTTT_Key;
    postString += " HTTP/1.1\r\n";
    postString += "Host: maker.ifttt.com\r\n";
    postString += "Content-Type: application/json\r\n";
    postString += "Content-Length: ";
    postString += lenString + "\r\n";
    postString += "\r\n";
    postString += jsonString; // combine post request and JSON
    client.print(postString);
    delay(500);
    client.stop();
}

```

สามารถเข้าไปดูได้ได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_16/Workshop_16.ino

Workshop 17 การสั่งการ ESP8266 ด้วย Google Assistant ผ่าน Blynk IoT และ IFTTT



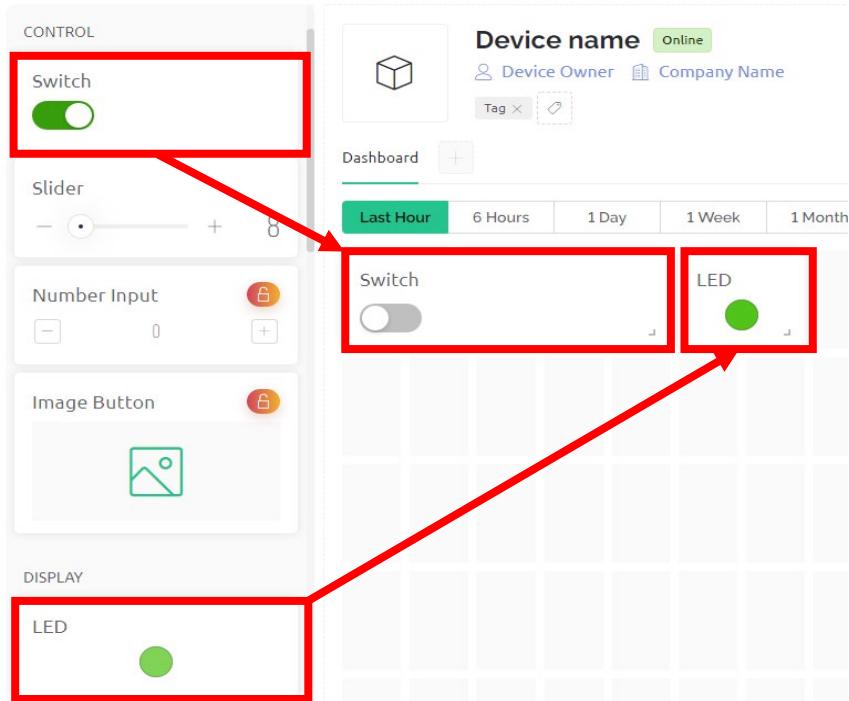
Google Assistant คือ เทคโนโลยีที่ช่วยเหลือผู้ใช้งานสมาร์ทโฟน และ อุปกรณ์อัจฉริยะต่างๆ ที่สามารถทำงานแทนผู้ใช้ได้หลายอย่างตามคำสั่งที่ตั้งไว้ และ ตอบโต้กับผู้ใช้งาน โดยสามารถใช้ได้เฉพาะกับอุปกรณ์ที่ใช้ระบบปฏิบัติการ Android เวอร์ชัน 5.0 หรือ iOS เวอร์ชัน 9.1 ขึ้นไป ในคราวนี้จะทำการประยุกต์ใช้ Google Assistant ให้สามารถควบคุมการทำงานของหลอดไฟ LED บน ESP8266 ผ่านการใช้งาน IFTTT และ Blynk IoT โดยสามารถทำได้ดังขั้นตอนต่อไปนี้

ส่วนของ Blynk IoT

- เข้าไปที่ Templates -> Web Dashboard และกด Edit เพื่อเข้าสู่การเปลี่ยนแปลง

The screenshot shows the Blynk IoT device configuration interface for an 'ESP8266 Google Assistant' device. The top navigation bar includes 'Info', 'Metadata', 'Datastreams', 'Events', 'Automations', and 'Web Dashboard' (which is highlighted with a red box). On the left, there's a sidebar with icons for 'Templates' (highlighted with a red box), 'Devices', 'Dashboards', and 'Logs'. The main area displays the device details: 'Device name' (set to 'Online'), 'Device Owner' (empty), 'Company Name' (empty), and a 'Dashboard' section with time filters: 'Last Hour' (selected), '6 Hours', '1 Day', '1 Week', '1 Month', '3 Months', and 'Custom'.

2. เพิ่ม “Switch” และ “LED” เข้ามาใน Dashboard



3. ตั้งค่า “Switch” ดังภาพต่อไปนี้ จากนั้นกด “Save”

Switch Settings ⓘ

The screenshot shows the 'Switch Settings' dialog box. At the top, there's a 'TITLE (OPTIONAL)' field containing 'Light Switch'. Below it is a 'Datastream' section for a 'Virtual Pin Datastream'. In this section, the 'NAME' is set to 'Light Switch' and the 'DATA TYPE' is 'Integer'. The 'PIN' is set to 'V0', 'UNITS' is 'None', and the 'MIN', 'MAX', and 'DEFAULT VALUE' are all set to '0'. There's also an 'ADVANCED SETTINGS' button. At the bottom of the dialog are 'Cancel' and 'Create' buttons. To the right of the dialog, a preview window shows a 'Light Switch' component with a green toggle switch icon. At the very bottom right of the screen, there's a larger 'Save' button.

4. ตั้งค่า “LED” ดังภาพต่อไปนี้ จากนั้นกด “Save”

LED Settings

The screenshot shows the configuration of a Virtual Pin Datastream named "LED". The "NAME" field is set to "LED" and the "ALIAS" field is also "LED". The "PIN" field is set to "V1" and the "DATA TYPE" is "Integer". The "UNITS" field is set to "None". The "MIN" value is 0, "MAX" value is 1, and the "DEFAULT VALUE" is 0. Below the form is a preview window titled "LED Status" showing a large green circle.

5. จากนั้นกด “Save and Apply” เพื่อบันทึกการเปลี่ยนแปลง

The screenshot shows the "Web Dashboard" section of the ESP8266 Google Assistant. It includes a control panel with a switch, slider, and number input, a device name section, and a map. The "Save And Apply" button is highlighted with a red box.

ส่วนของ ESP8266 และ Arduino IDE

Workshop_GoogleAssistant

```
#define BLYNK_TEMPLATE_ID "Enter your Blynk's template ID here"
#define BLYNK_DEVICE_NAME "Enter your Blynk's device name here"

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

const char ssid[] = "Enter your WiFi name here";
const char pass[] = "Enter your WiFi password here";
const char auth[] = "Enter your Blynk Auth Token here";

BlynkTimer timer;
void timerEvent();

void setup() {
    Blynk.begin(auth, ssid, pass);
    timer.setInterval(1000L, timerEvent);

    // sets the on-board LED pin as output
    pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
    Blynk.run();
    timer.run();

    // Update on-board LED state
    Blynk.virtualWrite(V1, !digitalRead(LED_BUILTIN));
}

BLYNK_WRITE(V0) {

    // Set incoming value from pin V0 to a variable
    int value = param.asInt();

    if(value == 0)
        digitalWrite(LED_BUILTIN, HIGH);
    else
        digitalWrite(LED_BUILTIN, LOW);
}

void timerEvent() {
    // Let it empty here. There is no use right now.
}
```

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/tree/main/Workshop-Seminar-IoT-2022/Workshop_17/Workshop_GoogleAssistant

ส่วนของ IFTTT

- สร้าง Applet บน IFTTT สำหรับเปิดไฟด้วยคำสั่งเสียง โดยกดที่ปุ่ม “Create”

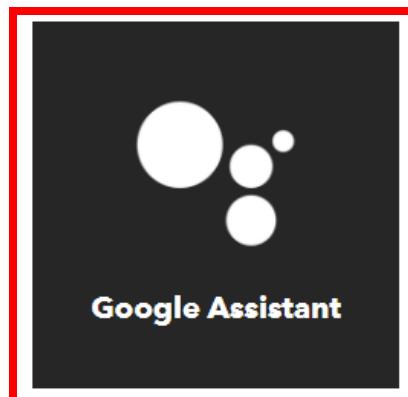


- ในส่วนของ “If This” ให้เลือกใช้บริการของ “Google Assistant”

The image shows the main creation interface of IFTTT. At the top, it says "Create your own". Below that is a banner with the text "Upgrade for more, faster, better Applets with advanced features. Upgrade". The main area has a heading "You're using 2 of 5 Applets". A large button labeled "If This" is highlighted with a red box. To its right is an "Add" button. Below this is a grey box labeled "Then That".

Choose a service

Q **google assistant** สามารถพิมพ์เพื่อค้นหาบริการที่สนใจได้



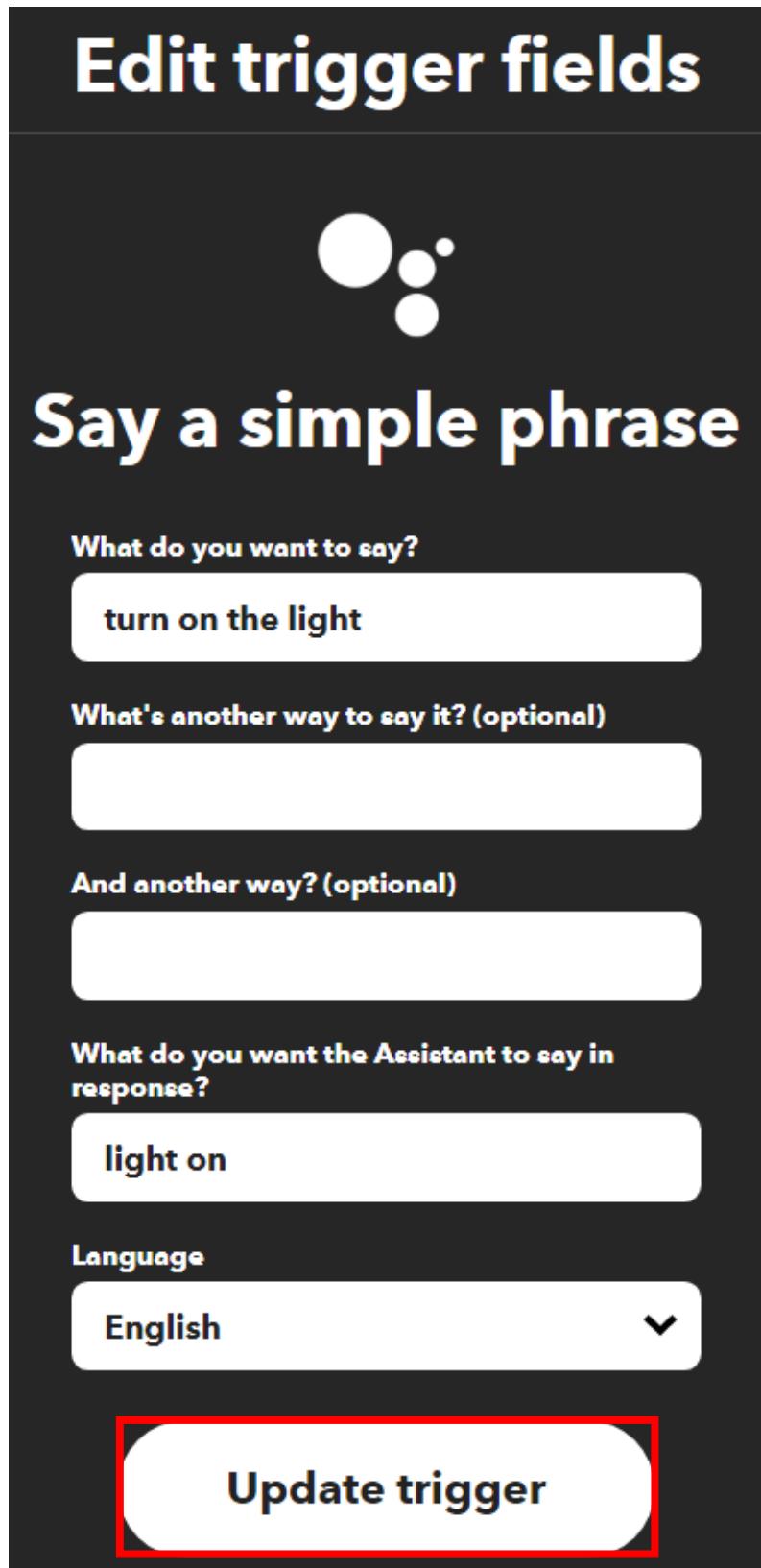
- เลือกที่ “Say a simple phrase”

Choose a trigger

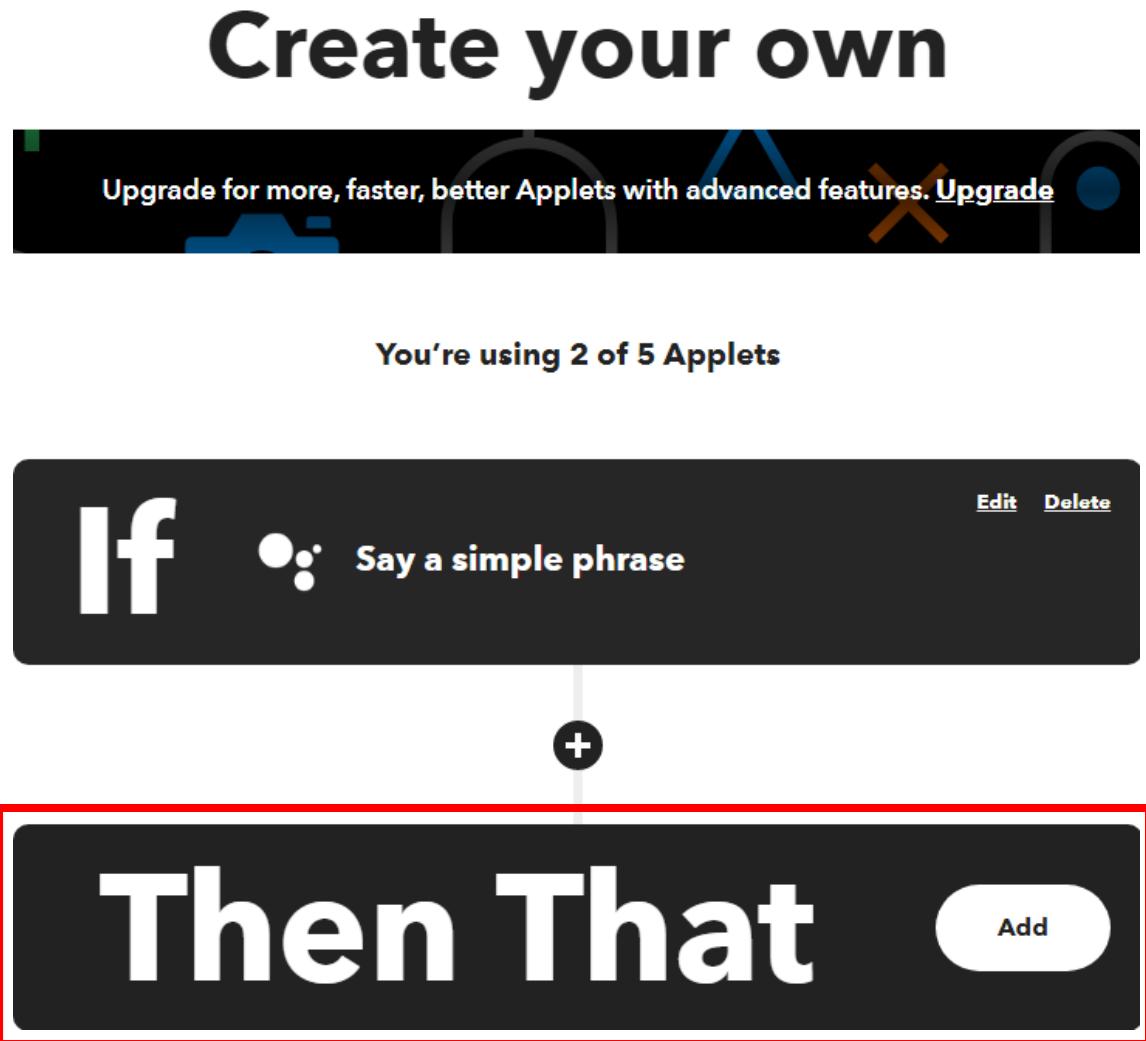
Google Assistant

<p>Say a simple phrase</p> <p>This trigger fires when you say “Ok Google” to the Google Assistant followed by a phrase you choose. For example, say “Ok Google, I’m running late” to text a family member that you’re on your way home.</p>	<p>Say a phrase with a number</p> <p>This trigger fires when you say “Ok Google” to the Google Assistant followed by a phrase like “Set Nest thermostat to 68.” **Use the # symbol to specify where you’ll say the number ingredient</p>	<p>Say a phrase with a text ingredient</p> <p>This trigger fires when you say “Ok Google” to the Google Assistant followed by a phrase like “Post a tweet saying ‘New high score.’” **Use the \$ symbol to specify where you’ll say the text ingredient</p>	<p>Say a phrase with both a number and a text ingredient</p> <p>This trigger fires when you say “Ok Google” to the Google Assistant followed by a phrase like “Block time for ‘exercise’ at 6 PM.” **Use the # symbol to specify where you’ll say the number ingredient and \$ where you’ll say the text ingredient</p>
--	---	--	--

4. ตั้งค่าบริการของ Google Assistant ดังภาพต่อไปนี้ จากนั้นกดปุ่ม “Update Trigger”



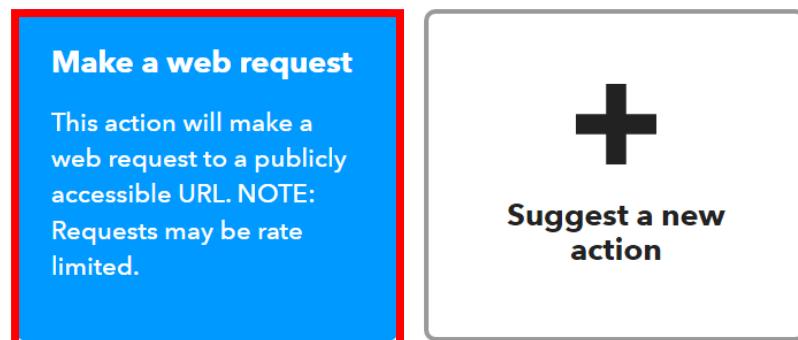
- ในส่วนของ “Then That” จากนั้นเลือกบริการ “Webhooks”



Choose a service



6. เลือกไปที่ “Make a web request”



7. ตั้งค่าบริการของ Webhooks ดังภาพต่อไปนี้ จากนั้นกดปุ่ม “Update Trigger”

หมายเหตุ: ในส่วนของ “URL” ที่ป้อนนั้นเรียกว่า “HTTPs API” ของ Blynk IoT ซึ่งมีหน้าที่ในการเรียกใช้การทำงานต่างๆ ที่ Blynk IoT สามารถทำได้ เช่น การรับข้อมูลที่อยู่บน Datastream หรือแม้แต่การสั่งการ Widget ต่างๆ ที่อยู่บนหน้า Dashboard ของ Blynk IoT โดยโครงสร้างของ HTTPs API ของ Blynk IoT คือ https://{server_address}/external/api/update?token={token}&{pin}={value} โดยที่ในส่วนของ {server_address} คือ ที่อยู่เซิฟเวอร์ของ Blynk IoT ที่ใช้ โดยสามารถดูได้ที่ภาพดังต่อไปนี้

The screenshot shows the Blynk IoT platform interface. On the left, there's a sidebar with sections for 'DEVICES', 'LOCATIONS', and 'USERS'. The main area is titled 'My devices' and shows one device listed: 'ESP8266 Google Assistant' owned by 'Anupat', status 'Offline', last updated '5:07 PM Today', and organization 'My organization - 2532GC'. A red box highlights the text 'Region: sgp1 Privacy Policy' located in the device's configuration area. Another red box highlights the URL 'Region: sgp1 Privacy Policy'.

หากรู้ที่อยู่ของเซิฟเวอร์ที่ Blynk IoT ใช้อยู่แล้ว ก็สามารถนำมาเทียบกับภาพดังต่อไปนี้เพื่อป้อน {server_address} ให้ถูกต้องได้

HTTPS API Troubleshooting

- All HTTPS API are case-sensitive. Request path and query parameters letter-case shouldn't be changed.
- Make sure you're using the correct server. Blynk currently have 2 clouds running.
The old cloud (no longer supported) with host `blynk-cloud.com` and the new cloud with host `blynk.cloud`
- In case you're getting the `Invalid token.` response from the HTTPS API, and you're sure the device auth token is correct - it could be a GEO DNS issue.

! Due to current GeoDNS settings you need to put server address with suffix manually depending on your region:

`https://fra1.blynk.cloud/` – Frankfurt

`https://lon1.blynk.cloud/` – London

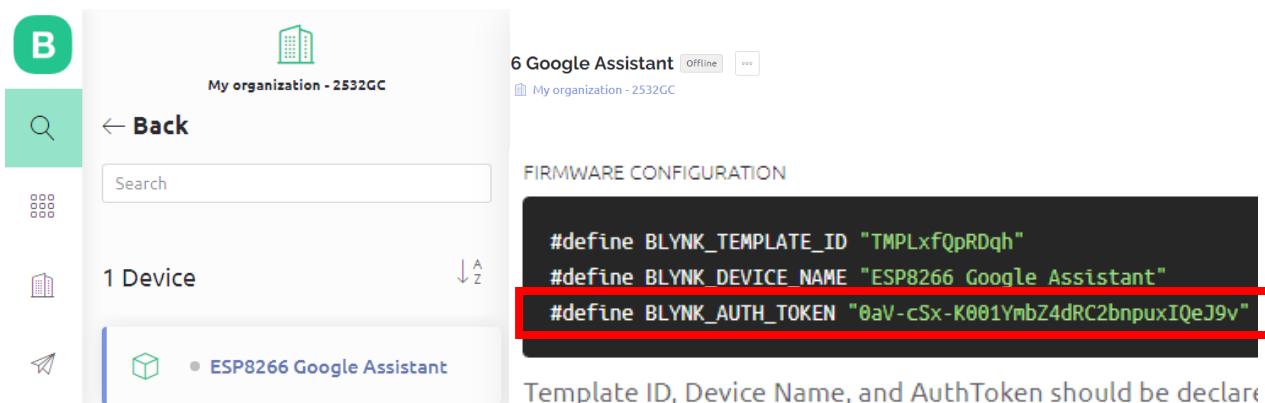
`https://ny3.blynk.cloud/` – New York

`https://sgp1.blynk.cloud/` – Singapore

`https://blr1.blynk.cloud/` – Bangalore

The server region could be found in the right bottom corner of the web interface.

ต่อมาในส่วนของ `{token}` ให้ทำการป้อน Device ที่อยู่บน Blynk IoT ลงโดยสามารถป้อน Device Token ได้ดังภาพต่อไปนี้ (Device Token ของแต่ละอุปกรณ์จะต้องไม่ซ้ำกัน เพราะจะนั่นห้ามคัดลอก Device Token ของคนอื่นมาโดยเด็ดขาด)



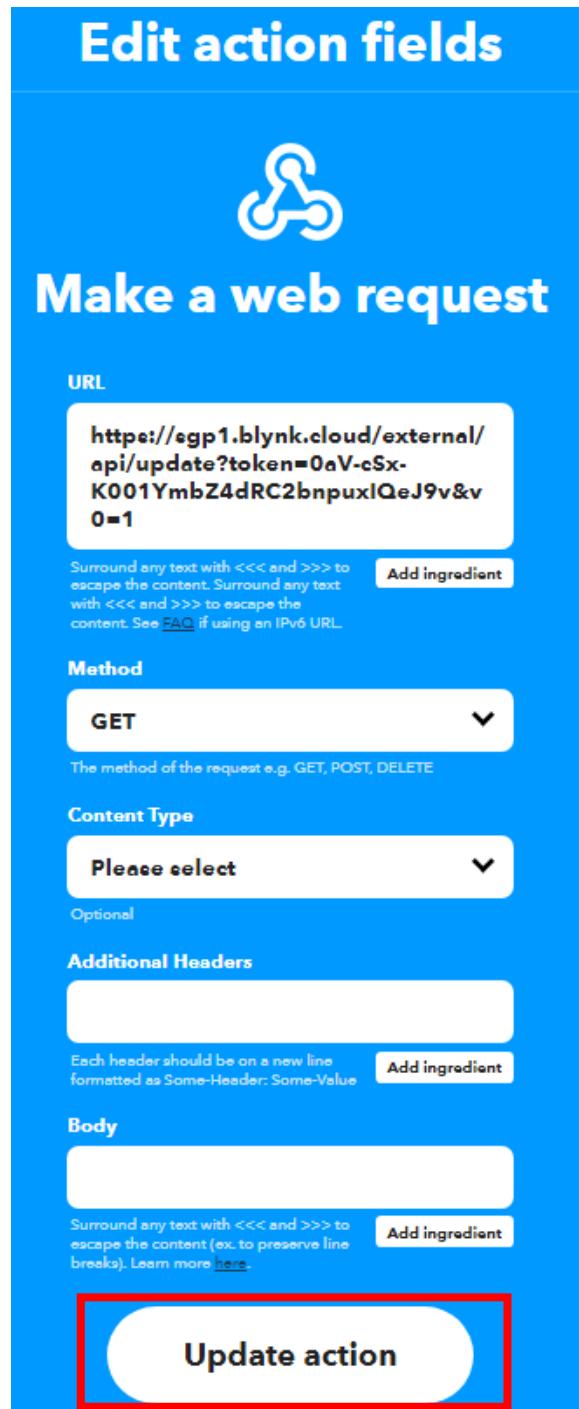
```

#define BLYNK_TEMPLATE_ID "TMPLxfQpRDqh"
#define BLYNK_DEVICE_NAME "ESP8266 Google Assistant"
#define BLYNK_AUTH_TOKEN "0aV-cSx-K001YmbZ4dRC2bnpxI0eJ9v"

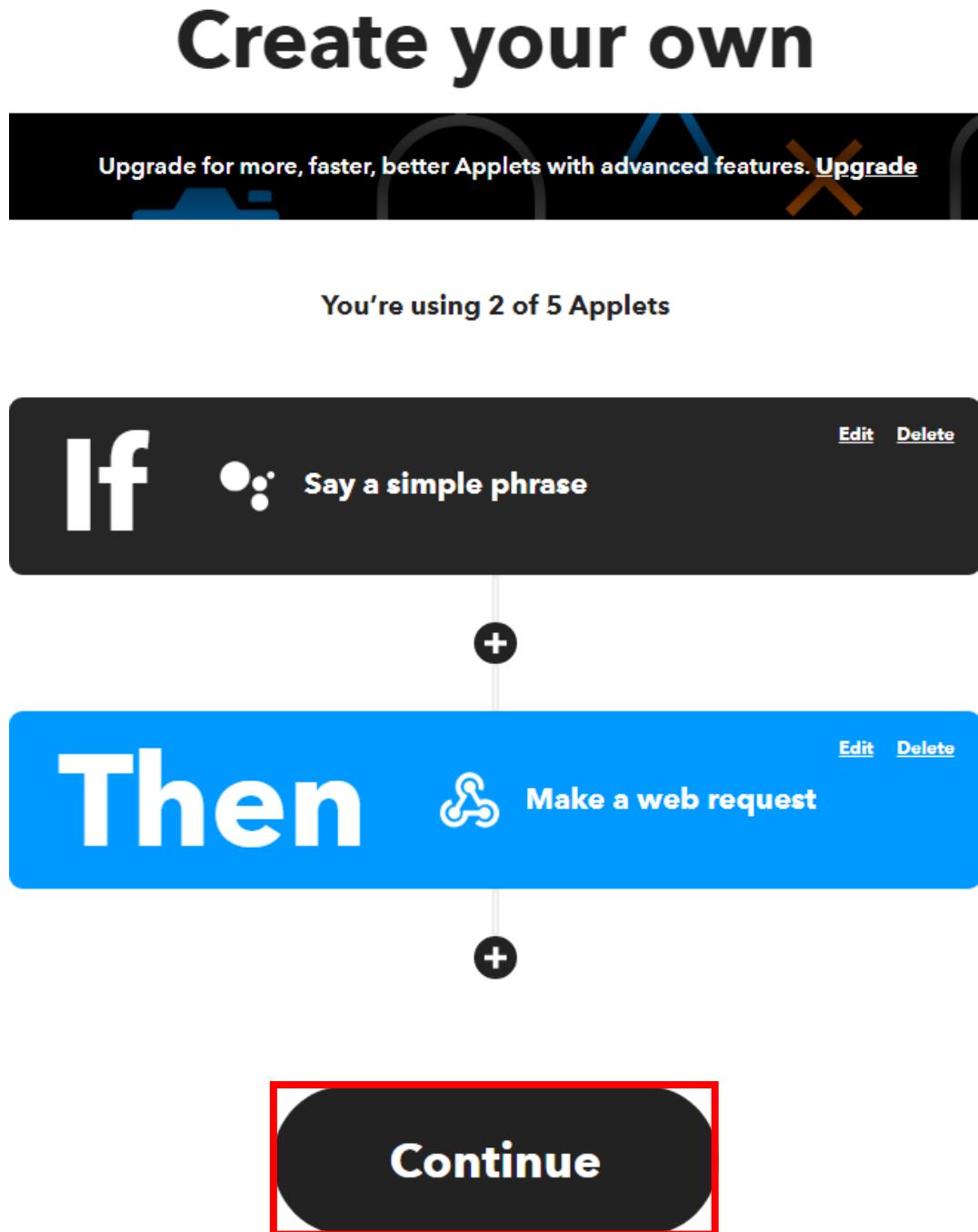
```

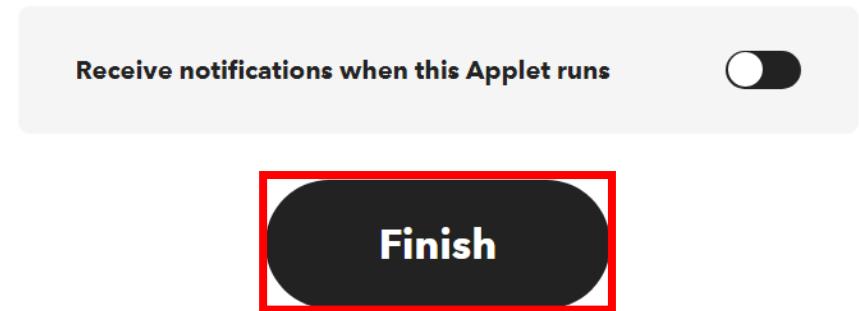
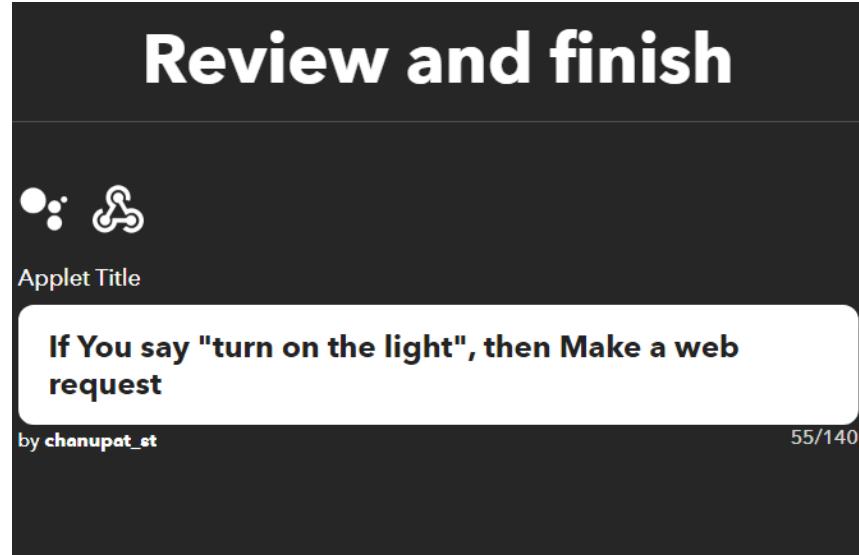
Template ID, Device Name, and AuthToken should be declared

และส่วนสุดท้ายคือ `{pin}` และ `{value}` เป็นส่วนที่ให้ระบุขาพินที่ต้องการจะเปลี่ยนแปลงค่า ซึ่งขาพินจะสอดคล้องกับ Datastream ที่ใช้ และเปลี่ยนแปลงค่าที่ต้องการในส่วนของ `{value}` ให้มีค่าเป็นไปตามที่ผู้ใช้ต้องการ



- กด “Continue” และ “Finish” เพื่อเสร็จสิ้นการสร้าง Applet





9. สร้าง Applet อีกอันหนึ่งบน IFTTT สำหรับปิดไฟด้วยคำสั่งเสียง โดยสามารถตั้งค่าส่วนของ Google Assistant และ Webhooks ได้ ดังนี้

Edit trigger fields



Say a simple phrase

What do you want to say?

turn off the light

What's another way to say it? (optional)

And another way? (optional)

What do you want the Assistant to say in response?

light off

Language

English ▾

Update trigger

Edit action fields



Make a web request

URL

`https://egp1.blynk.cloud/external/api/update?token=0aV-cSx-K001YmbZ4dRC2bnpxlQeJ9v&v=0=0`

Surround any text with <<< and >>> to escape the content. Surround any text with <<< and >>> to escape the content. See [FAQ](#) if using an IPv6 URL.

Add ingredient

Method

GET

The method of the request e.g. GET, POST, DELETE

Content Type

Please select

Optional

Additional Headers

Each header should be on a new line formatted as Some-Header: Some-Value

Add ingredient

Body

Surround any text with <<< and >>> to escape the content (ex. to preserve line breaks). Learn more [here](#).

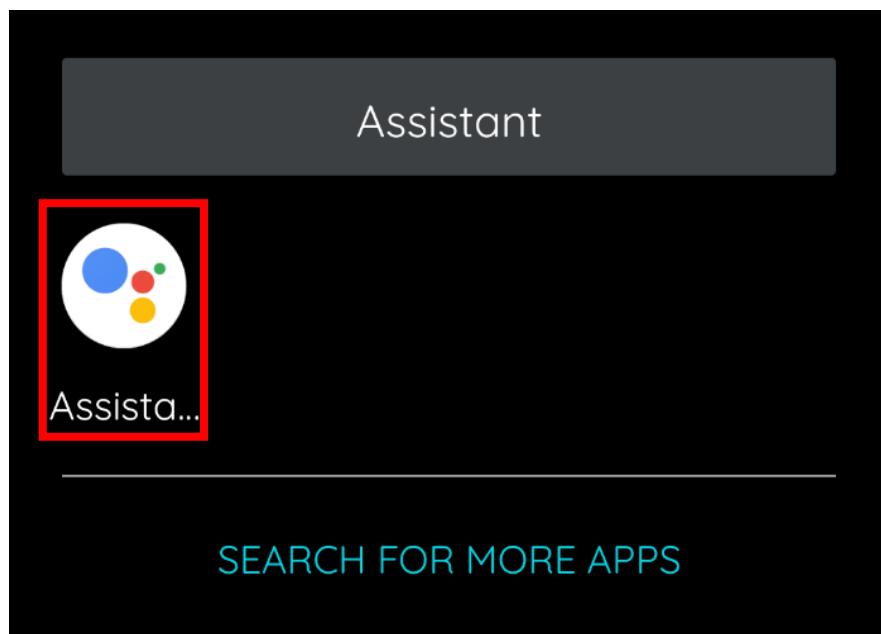
Add ingredient

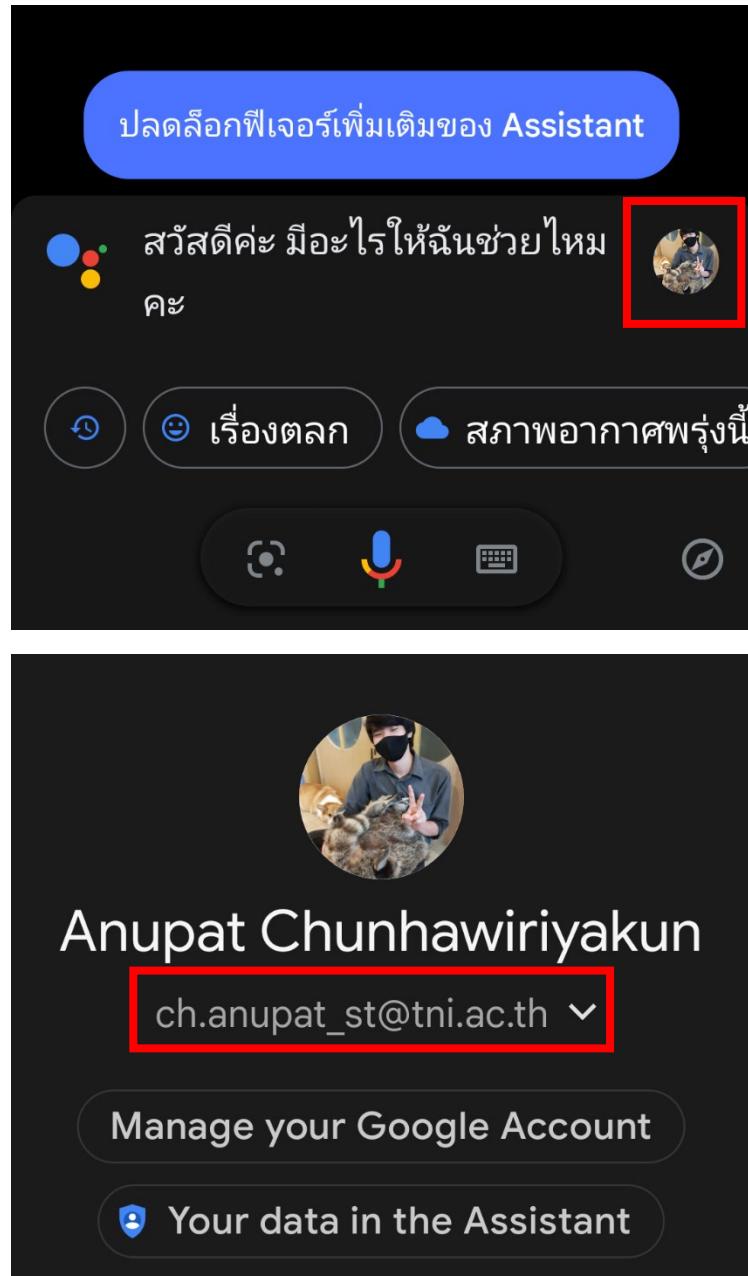
Update action

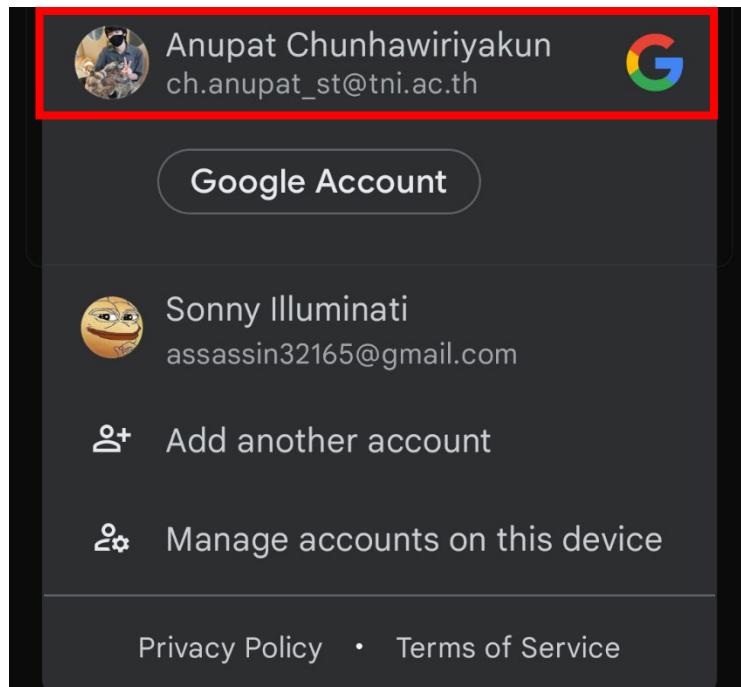
10. เมื่อสร้างเสร็จแล้ว สามารถตรวจสอบผลลัพธ์การสร้าง Applet ได้โดยไปที่ “My Applets”

The screenshot shows the IFTTT interface with the 'My Applets' tab selected. There are two applets listed under the 'All (2 of 5)' category. Both applets are titled "If You say 'turn off/on the light', then Make a web request" and were created by "chanupat_st". Each applet has a status of "Connected" and 1 action. A red box highlights these two specific applets.

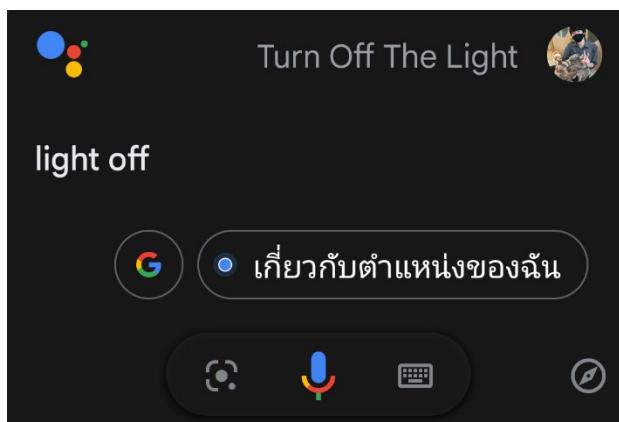
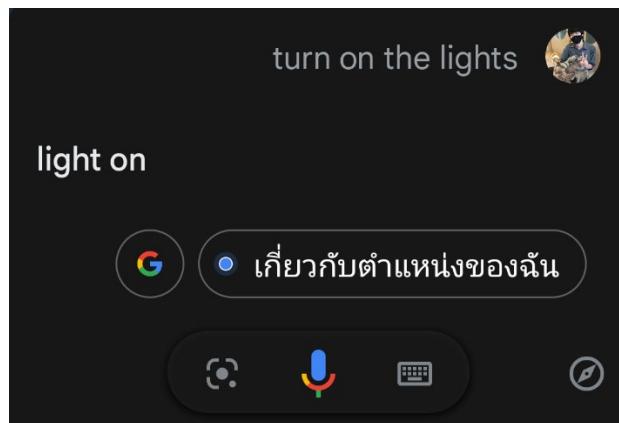
11. สำหรับผู้ใช้สมาร์ทโฟน Android จะมี Google Assistant ติดตั้งมาให้แล้ว หากเป็นผู้ใช้สมาร์ทโฟน iOS ให้ติดตั้ง Google Assistant ผ่าน App Store ก่อน จากนั้นจึงเปิด Google Assistant ขึ้นมา และเปลี่ยนบัญชีที่จะใช้งาน Google Assistant ให้ตรงกับบัญชี Google ที่ใช้สร้าง Applet โดยสามารถตั้งค่าได้ตามภาพดังต่อไปนี้





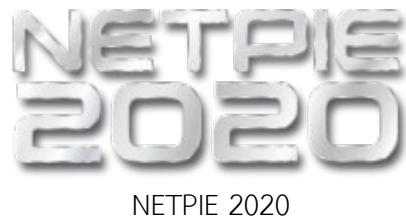


12. ทดลองสั่งการผ่าน Google Assistant โดยการพูดคำสั่งที่กำหนดไว้ใน Applet ของ IFTTT หากทำ
ถูกต้อง Google Assistant จะต้องตอบกลับมาตามที่ได้กำหนดไว้ใน Applet ของ IFTTT



NETPIE

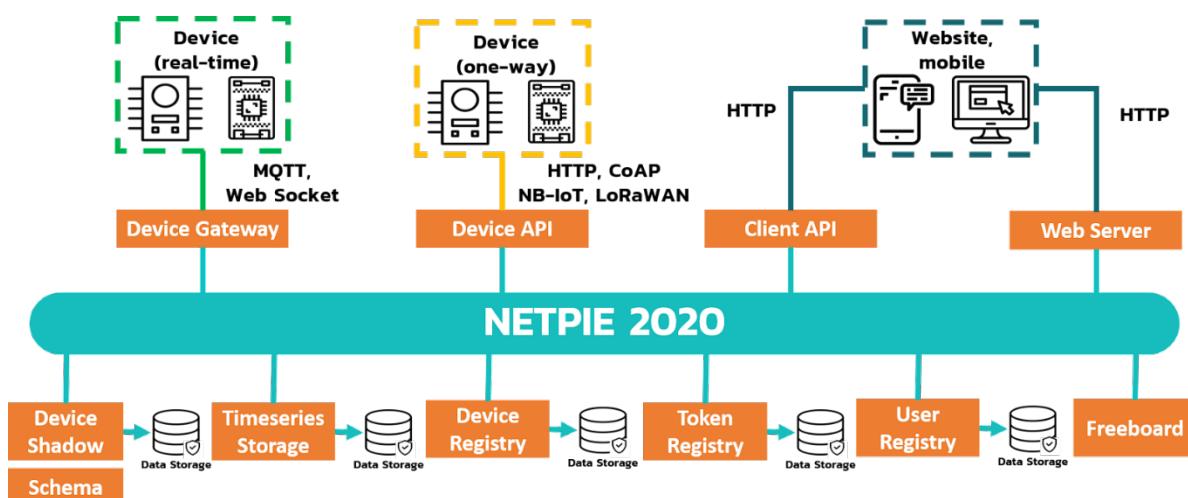
NETPIE 2020 คือแพลตฟอร์มที่ถูกพัฒนาขึ้นเพื่อตอบสนองผู้ใช้งานเชิงพาณิชย์ เช่น ผู้ผลิตอุปกรณ์ IoT, อุตสาหกรรม, โรงงาน และองค์กรที่พัฒนาสู่ยุค Digital Transformation 4.0 ซึ่งจะช่วยธุรกิจให้มีประสิทธิภาพยิ่งขึ้น ด้วยเทคโนโลยีการเชื่อมต่อทุกสรรพสิ่ง หรือ Internet of Things (IoT)



ที่มา: <https://netpie.io/compare>

คุณสมบัติหลักๆ ของ NETPIE 2020 Platform ประกอบไปด้วย

1. การแสดงค่าข้อมูลจากเซ็นเซอร์หรืออุปกรณ์แบบ Real-time (Monitoring)
2. การควบคุมการทำงานของอุปกรณ์ต่างๆ ผ่าน Cloud Platform (Controlling)
3. การเก็บค่าข้อมูลที่ได้จากเซ็นเซอร์หรืออุปกรณ์ (Data Storage)
4. การแจ้งเตือนความผิดปกติของเซ็นเซอร์หรืออุปกรณ์จากที่ได้กำหนดไว้ (Notification)
5. การแสดงผลและควบคุมการทำงานของอุปกรณ์ผ่าน Dashboard (Dashboard for monitor & control)



แผนภาพระบบของ NETPIE 2020

ที่มา: <https://docs.netpie.io/overview-netpie.html>

เริ่มต้นใช้งาน NETPIE 2020

เริ่มต้นลงที่เบียนใช้งาน และ เข้าสู่ระบบ NETPIE 2020 ผ่าน <https://netpie.io/>

Connect Everything

NETPIE is an IoT cloud-based platform-as-a-service that helps connect your IoT devices together seamlessly by pushing the complexity from the hands of application developers or device manufacturers to the cloud

GET STARTED **WATCH VIDEO**

NETPIE 2020
From Makers Nation
Toward Smart Nation

Introducing NETPIE 2020

NETPIE 2020 is the latest version of NETPIE. It aims to fulfill the needs of commercial users, especially those in the industrial sector. The platform's new version can shorten and ease the process of IoT product development considerably, from the stage of designing, prototyping, implementing right down to administering and maintaining. NETPIE 2020 comes with the newly designed architecture and many exciting new features.

ภาพหน้าหลัก NETPIE

ให้กดปุ่ม Sign up ด้านขวาเมื่อบน จะปรากฏหน้านี้

NETPIE 2020

EMAIL
 required

NAME
 required

ORGANIZATION
 required

COUNTRY CODE

MOBILE PHONE NUMBER* (NO COUNTRY CODE)
 required and number only

I agree to the [Privacy Statement](#) and [Terms of Use](#)

SIGN UP

เมื่อสมัครเสร็จ ให้ SIGN IN



The image shows the NETPIE 2020 login interface. At the top, the text "NETPIE 2020" is displayed in large, stylized letters, with "Connect Everything" below it. Below this, there are two input fields: the first for email (containing "warakorn@tni.ac.th") and the second for password (containing a series of dots). To the right of the password field is a link "Forgot your password?". A large red "SIGN IN" button is centered below the inputs. Below the sign-in area, a horizontal line separates the login form from the rest of the page. Underneath the line, the text "Don't have an account yet? [Register with NETPIE](#)" is visible. To the left of this text is the NETPIE logo, which consists of a blue and red gear-like icon followed by the word "NETPIE". To the right is a user profile icon showing a person's head and shoulders, with the name "Warakorn" next to it. A small downward arrow indicates a dropdown menu. The main content area below the line is light gray and features a central icon of a computer monitor with a speech bubble, accompanied by the text "No Data". At the bottom of this area is a large blue circular button with a white plus sign. At the very bottom of the page, a copyright notice reads "Copyright © 2018-2020 Created by NETPIE".

NETPIE 2020
Connect Everything

warakorn@tni.ac.th

.....

Forget your password?

SIGN IN

Don't have an account yet? [Register with NETPIE](#)

NETPIE

Warakorn

No Data

+

Copyright © 2018-2020 Created by NETPIE

ภาพหน้าต่าง Project

การสร้าง Project

สามารถสร้าง Project ได้โดยการกดที่ปุ่มบวกบริเวณมุมล่างซ้ายของหน้าต่าง



เมื่อกดแล้ว จะมีหน้าต่างให้กรอกข้อมูล Project โดยที่ * คือข้อมูลที่จำเป็นต้องกรอก ให้กรอกชื่อ Project “Test”

Create Project X

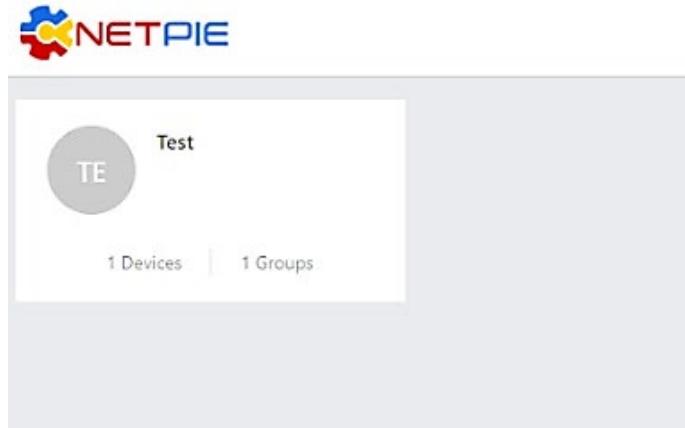
* Name:

Description:

Tag:
 + New Tag

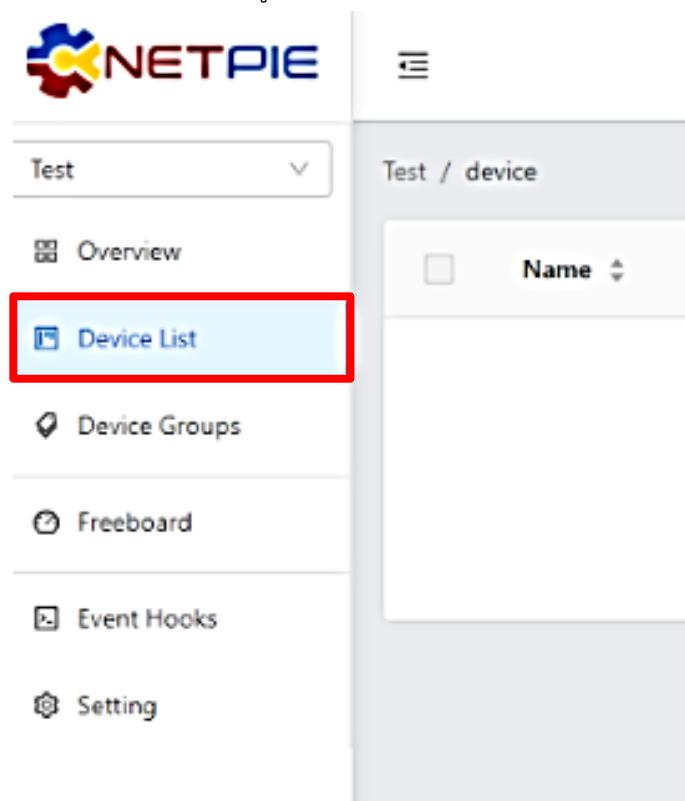
Cancel Create

เมื่อกรอกข้อมูลเสร็จแล้ว กดปุ่ม “Create” ผลลัพธ์ที่ได้จะเป็นดังนี้ สามารถกด Project เพื่อดำเนินการต่อไป



การสร้าง Device

การสร้าง Device สามารถทำได้โดยการเลือกเมนู “Device Lists” ทางด้านขวาเมื่อ



จากนั้นให้คลิกที่ปุ่ม “Create” เพื่อสร้าง Device ใหม่ โดยที่ * คือข้อมูลที่จำเป็นต้องกรอก จากนั้นกดปุ่ม “Create” เพื่อทำการสร้าง Device จากนั้นให้ระบุชื่อ Device “ESP8266” ดังภาพ

The screenshot shows the NETPIE web interface. On the left, there's a sidebar with options: Test (selected), Overview, Device List (highlighted in blue), Device Groups, Freeboard, Event Hooks, and Setting. The main content area is titled "Test / device" and shows a table with one row. The row has a checkbox, the name "ESP8266", and a "Tags" column with "No Data". In the top right of this area, there's a "Create" button with a red box around it. At the bottom of the main content area, it says "Copyright © 2018-2020 Created by NETPIE".

Create X

* Name:

Description:

Tag:

Cancel
Create

The screenshot shows the NETPIE platform's Device List interface. On the left, a sidebar menu includes options like Overview, Device List (which is selected and highlighted in blue), Device Groups, Freeboard, Event Hooks, and Setting. The main content area displays a table titled "Test / device" with one item: "ESP8266". The table columns are Name, Tags, Group, and Create Date. The "Create Date" column shows "2022-01-22 00:35". At the bottom of the table, there are navigation buttons for "1-1 of 1 items" and a page size selector set to "10 / page". A "Create" button is located in the top right corner of the main content area.

กดเข้าไปที่ Device รายละเอียดต่างๆจะปรากฏขึ้น รวมถึง Key, Token และ Secret ที่จะนำไปใช้เพื่อให้ Device สามารถเชื่อมต่อเข้ามายัง Platform ได้

The screenshot shows the NETPIE platform's Device Detail interface for the "ESP8266" device. The left sidebar is identical to the previous screenshot. The main content area is titled "Test / device / ESP8266". It contains two main sections: "Description" and "Key". The "Key" section displays Client ID, Token, and Secret values, each with a copy icon. Below this are "Status" (set to Offline) and "Enable" (set to On). At the bottom, there are tabs for Shadow, Schema, Trigger, and Feed, along with Save and Cancel buttons. A "Tree" section at the bottom allows selecting a node, showing "object {0}" and "(empty object)".

The screenshot shows the NETPIE device configuration interface for an ESP8266 device named 'Test'. The left sidebar includes options like Overview, Device List, Device Groups, Freeboard, Event Hooks, and Setting. The main panel displays device details and configuration tabs for Shadow, Schema, Trigger, and Feed. A 'Key' section lists MQTT credentials and status. A red note in the center says 'สถานะการเชื่อมต่อ Platform' and 'สถานะการเปิดใช้งานอุปกรณ์'. A 'Resync Status' button is highlighted in red.

Key	
Client ID	: d5706e49-c340-4b4f-8670-5242b5f59a47
Token	: ZFJYWwEAY8mGTmT14Wvhw14ynqPfitRr
Secret	: qQt1Shr0EeaNM(Vy5n7Fvlv4(CAQjgl1
Status	: Offline
Enable	: <input checked="" type="checkbox"/>

Resync Status

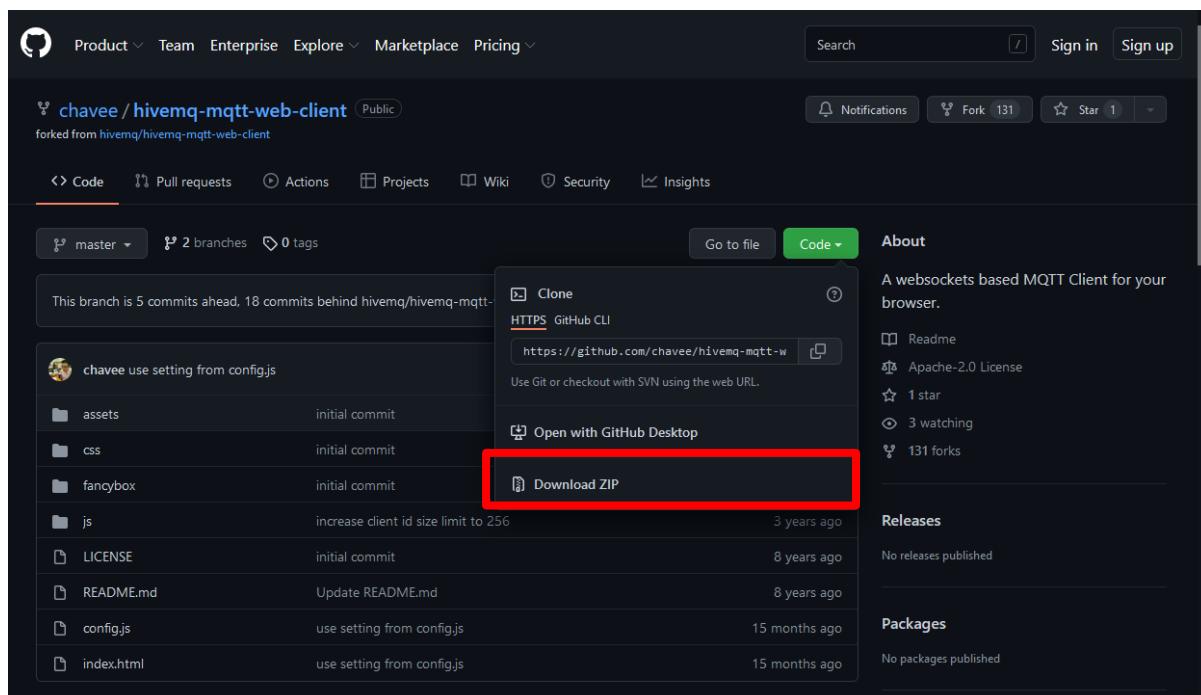
การเชื่อมต่อ Device กับ Platform

ในการเชื่อมต่ออุปกรณ์กับ Platform จะใช้ Key สำหรับการเชื่อมต่อของ Device มาบ้าง Platform กรณีเชื่อมต่อผ่าน MQTT Protocol ให้เลือกใช้งาน MQTT Client Library ที่เหมาะสมหรือรองรับกับ Device ที่จะใช้ในการเชื่อมต่อ โดยการเชื่อมต่อของ MQTT จะต้องใช้ 4 Parameters คือ Host, Client id, Username และ Password โดยดูข้อมูลที่จะนำมาใช้ สามารถระบุค่าได้ดังนี้

Host	mqtt.netpie.io
Port	1883 (mqtt), 1884 (mqqtts)
Client ID	Client ID ของ Device ที่สร้างขึ้นใน NETPIE
Username	Token ของ Device ที่สร้างขึ้นใน NETPIE
Password	ยังไม่ต้องระบุ (ใช้สำหรับกรณีที่ต้องการตรวจสอบที่เพิ่มมากขึ้น)

ทดลองเชื่อมต่อ Platform ด้วย HiveMQ โดยสามารถโหลดได้ที่

<https://github.com/chavee/hivemq-mqtt-web-client>



เมื่อดาวน์โหลดเรียบร้อยแล้ว ให้ทำการแตกไฟล์แล้วเปิดไฟล์ index.html

Name	Date modified	Type	Size
assets	2/7/2021 10:04 AM	File folder	
css	2/7/2021 10:04 AM	File folder	
fancybox	2/7/2021 10:04 AM	File folder	
js	2/7/2021 10:04 AM	File folder	
config.js	2/7/2021 10:04 AM	JavaScript File	1 KB
index.html	2/7/2021 10:04 AM	Microsoft Edge H...	14 KB
LICENSE	2/7/2021 10:04 AM	File	12 KB
README.md	2/7/2021 10:04 AM	MD File	2 KB

เมื่อเปิดไฟล์แล้ว ทำการกรอก Client ID ของ Device ที่ช่อง “ClientID” และ Token ของ Device ที่ช่อง “Username” ในส่วนของ port ใช้ “80” และให้ทำการกดปิด SSL ดังรูป

จากนั้นคลิกปุ่ม “Connect” เพื่อทำการเชื่อม Platform โดยหากเชื่อมต่อได้สถานการณ์เชื่อมต่อจะเป็น connected ดังรูป

The screenshot shows the HiveMQ Websockets Client Showcase interface. At the top left is the HiveMQ logo. To its right is the title "Websockets Client Showcase". Below the title, there is a "Connection" status bar with a green circle icon labeled "connected". The main area is divided into three sections: "Publish", "Subscriptions", and "Messages". The "Publish" section contains fields for "Topic" (set to "testtopic/1"), "QoS" (set to 0), "Retain" (unchecked), and a "Publish" button. The "Subscriptions" section has a button "Add New Topic Subscription". The "Messages" section is currently empty.

ทดสอบว่าสามารถเชื่อมต่อ Platform ได้จริงโดย Publish เข้าหาตัวเอง การเข้าค่า Topic ที่จะ Publish/Subscribe ให้ขึ้น Topic ด้วย @msg/ (คลิกปุ่ม “Subscribe” ก่อนที่จะคลิกปุ่ม “Publish”)

The screenshot shows the HiveMQ Websockets Client Showcase interface. At the top left is the HiveMQ logo. To its right is the title "Websockets Client Showcase". Below the title, there is a "Connection" status bar with a green circle icon labeled "connected". The main area is divided into three sections: "Publish", "Subscriptions", and "Messages". The "Publish" section contains fields for "Topic" (set to "@msg/test"), "QoS" (set to 0), "Retain" (unchecked), and a "Publish" button. The "Subscriptions" section has a button "Add New Topic Subscription" and a list entry for "Qos: 0 @msg/test". The "Messages" section displays a single message entry: "2022-04-28 17:12:07 Topic: @msg/test Qos: 0".

การสื่อสารระหว่าง Device

Device ที่จะสามารถสื่อสารกันได้ต้องอยู่ภายใต้ Device Group เดียวกัน โดยเริ่มจากไปที่เมนู “Device Groups”

ให้ทำการสร้าง Group โดยคลิกที่ปุ่ม “Create” และกรอกข้อมูล

Create

* Name:	Group_Test_01
Description:	
Tag:	+ New Tag
<input type="button" value="Cancel"/> <input type="button" value="Create"/>	

The screenshot shows the NETPIE web interface under the 'Test' project. On the left sidebar, 'Device Groups' is selected. The main area displays a table titled 'Test / group' with one item:

Name	Tags	Devices
Group_Test_01	-	0 Online 0 Offline

At the bottom, there are navigation buttons: '<' (left), '1' (selected), '>' (right), and '10 / page' (page size).

กลับไปที่ “Device Lists” และทำการสร้าง Device ใหม่อีก 1 ตัวเพื่อใช้ในการสื่อสารระหว่าง Device กับ Device

The screenshot shows the NETPIE web interface under the 'Test' project. On the left sidebar, 'Device List' is selected. The main area displays a table titled 'Test / device' with two items:

Name	Tags	Group
NodeMCU8266	-	-
ESP8266	-	-

Each device row has a checkbox icon on the left. At the bottom, there are navigation buttons: '<' (left), '1' (selected), '>' (right), and '10 / page' (page size).

จัด Device หั้ง 2 ตัวเข้า Group ที่สร้างไว้

The screenshot shows the NETPIE web interface under the 'Test / device' section. On the left sidebar, 'Device List' is selected. In the main area, two devices are listed: 'NodeMCU8266' and 'ESP8266'. Both devices have a blue checkmark icon to their left, indicating they are selected. A red box highlights the 'Move' button in the top right corner of the table header. The table has columns for 'Name', 'Tags', and 'Group'.

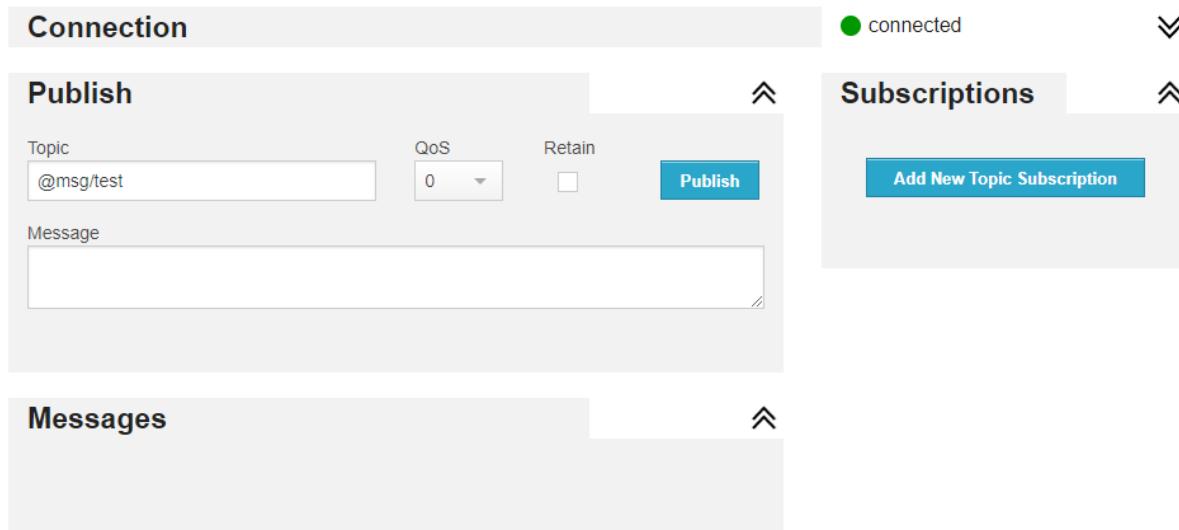
<input checked="" type="checkbox"/>	Name	Tags	Group
<input checked="" type="checkbox"/>	NodeMCU8266	-	-
<input checked="" type="checkbox"/>	ESP8266	-	-

เมื่อนำ Device เข้า Group แล้ว ในแต่ละ Device จะมี Group ปรากฏขึ้น

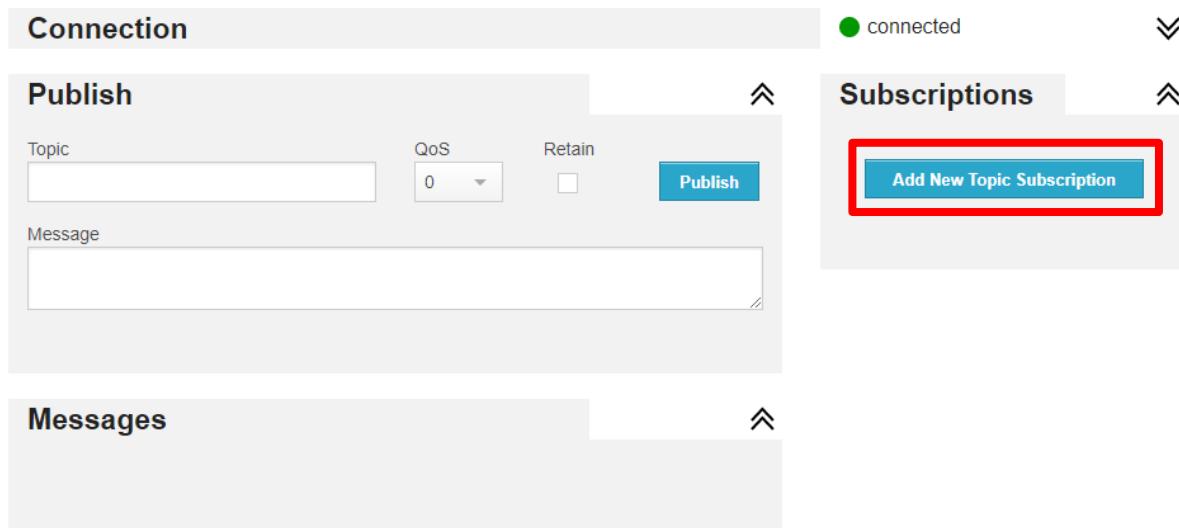
The screenshot shows the same NETPIE interface after the devices have been moved into a group. The 'Group' column now contains the value 'Group_Test_01' for both devices. A red box highlights this value for the 'ESP8266' entry. The rest of the interface remains the same, with the 'Device List' tab selected in the sidebar.

<input type="checkbox"/>	Name	Tags	Group
<input type="checkbox"/>	NodeMCU8266	-	Group_Test_01
<input type="checkbox"/>	ESP8266	-	Group_Test_01

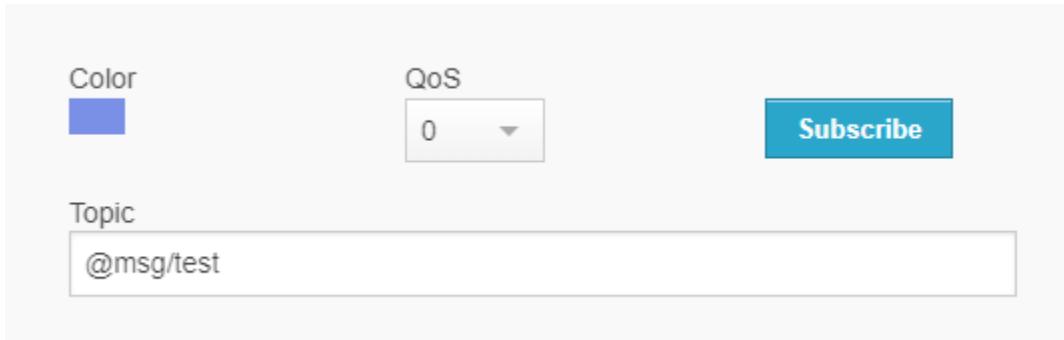
ทดลองการสื่อสารโดยการเปิด HiveMQ ใหม่ขึ้นมาอีกหนึ่งตัว โดยตั้งค่าการเชื่อมต่อตาม Device ใหม่ที่สร้างขึ้น จากนั้นจึงเริ่มสื่อสารกันโดย HiveMQ ตัวแรกจะเป็นตัว Publish และ HiveMQ ตัวที่สองจะเป็นตัว Subscribe ใน HiveMQ ตัวที่หนึ่งให้ทำการเซ็ตค่าที่ Topic ซึ่งขึ้นต้นด้วย @msg/{Topic} โดยในรูปตัวอย่างได้ใช้ “@msg/test”



ใน HiveMQ ตัวที่สองให้ทำการเซ็ตค่าที่ Subscribe ซึ่งขึ้นต้นด้วย @msg/{Topic}



กรอก Topic ชื่อ “@msg/test” จากนั้นกด Subscribe



เมื่อกรอกเสร็จแล้วผลลัพธ์จะได้ดังรูป

HiveMQ ตัวที่หนึ่ง (Publish)

HiveMQ ตัวที่สอง (Subscribe)

ทดลองการสื่อสารโดยทำการพิมพ์ข้อความลงในช่อง Message ของ HiveMQ ตัวที่หนึ่ง (Publish) และกด Publish

The screenshot shows the HiveMQ Websocket Client Showcase interface. At the top, there's a 'Connection' status indicator showing 'connected'. Below it, the 'Publish' section has a 'Topic' input set to '@msg/test', a 'QoS' dropdown set to '0', and a 'Retain' checkbox. A red box highlights the 'Publish' button. To the right, the 'Subscriptions' section has a blue 'Add New Topic Subscription' button. Below these are two empty sections: 'Messages' and another 'Subscriptions' section.

หากสื่อสารสำเร็จข้อความจะขึ้นในช่อง Messages ของ HiveMQ ตัวที่สอง (Subscribe)

This screenshot shows the same interface as above, but with the 'Messages' section highlighted by a large red box. In the 'Subscriptions' section on the right, there's a list with one item: 'Qos: 0 @msg/test'. The 'Messages' section displays a message from '2022-04-28 14:12:04' with the text 'Topic: @msg/test Qos: 0 Hi This is HiveMQ_1'.

การตั้งค่า Device

Device Shadow

Device Shadow คือ ฐานข้อมูลเสมือนของอุปกรณ์ เป็นฐานข้อมูลเล็ก ๆ ที่มีคู่อยู่กับอุปกรณ์ (Device) ทุกตัว ใช้สำหรับเก็บข้อมูลต่าง ๆ เกี่ยวกับอุปกรณ์นั้น ๆ (Device Shadow Data) เช่น ข้อมูลที่เกิดจากเซนเซอร์ ข้อมูลการกำหนดองค์ประกอบต่าง ๆ (Device Configuration) เป็นต้น

Device Schema

นิยามของ Device Schema ในที่นี้ คือ แมปผังข้อมูลที่กำหนดไว้เพื่อใช้กำกับ Device Shadow สำหรับ Device ที่ต้องมีการจัดการข้อมูล แนะนำให้สร้าง Device Schema ของข้อมูลเตรียมไว้ Device Schema เป็นส่วนหนึ่งของ Device Template ทำให้ Server สามารถ

- การตรวจสอบชนิดข้อมูล (Data Validation)
- การแปลงข้อมูล (Data Transformation) เช่น เปลี่ยนหน่วยของข้อมูล เป็นต้น
- การเก็บข้อมูลลงใน Timeseries Database

โดย Device Schema จะประกาศในรูปแบบ JSON มีลักษณะดังนี้

```
{
  "additionalProperties": false,
  "properties": {
    "temp": {
      "operation": {
        "store": {
          "ttl": "30d"
        },
        "transform": {
          "expression": "($.temp * 1.8) + 32"
        }
      },
      "type": "number"
    },
    "place": {
      "operation": {
        "store": {
          "ttl": "7m"
        }
      },
      "type": "string"
    }
  }
}
```

Device Schema จะประกอบด้วย

additionalProperties (boolean)

- สถานะการอนุญาตให้บันทึกข้อมูลลง Shadow หรือ Timeseries Database ในกรณีที่ข้อมูลไม่ตรงตามที่กำหนดใน Properties
- additionalProperties : true
 - อนุญาตให้บันทึกลง Shadow หรือ Timeseries Database
- additionalProperties : false
 - “ไม่อนุญาตให้บันทึกเฉพาะส่วนที่ไม่ตรงตาม Properties”
- ตัวอย่าง
 - Schema มีการกำหนด Properties เป็น temp, humid ข้อมูลที่ส่งมาเป็น temp, humid และ color ถ้าหาก additionalProperties เป็น true ข้อมูลของ color จะถูกบันทึกลงไปใน shadow หรือ feed แต่หากเป็น false จะมีเพียง humid และ temp เท่านั้นที่จะถูกบันทึกลง shadow หรือ Timeseries Database

properties(json)

เริ่มจากกำหนดชื่อฟิลด์ (จากตัวอย่าง คือ “temp” และ “place”) และกำหนดคุณสมบัติของแต่ละฟิลด์ซึ่งจะอยู่ในรูปแบบ JSON โดยจะแยก 2 ส่วน คือ

- operation สำหรับตั้งค่าการจัดการข้อมูลในฟิลด์นั้น ๆ ประกอบด้วย
 - store สำหรับตั้งค่าการเก็บข้อมูลลง Timeseries Database
 - ttl คือ ระยะเวลาของการเก็บข้อมูลใน Timeseries Database แต่ละจุดข้อมูลที่มีอายุการเก็บครบตามที่กำหนดจะถูกลบทิ้งอัตโนมัติ จำเป็นต้องกำหนดค่านี้ ระบบถึงจะเก็บข้อมูลลง Timeseries Database การกำหนดค่าจะระบุตัวเลขจำนวนเต็มตามด้วยหน่วยเวลา ดังนี้ ms(มิลลิวินาที), s(วินาที), m(นาที) h(ชั่วโมง), d(วัน), y(ปี) ถ้าไม่ระบุหน่วยค่า default จะเป็น ms(มิลลิวินาที) เช่น 30d หมายถึง เก็บข้อมูลนาน 30 วัน, 1y หมายถึง เก็บข้อมูลนาน 1 ปี, 3000 หมายถึง เก็บข้อมูลนาน 3 วินาที
 - transform การแปลงข้อมูล (Data Transformation) ก่อนการจัดเก็บ
 - expression คือ สูตรหรือวิธีการแปลงข้อมูล (Data Transformation) ก่อนการจัดเก็บ เช่น กำหนด expression เท่ากับ $(\$.temp * 1.8) + 32$ เป็นการแปลงหน่วยอุณหภูมิค่าที่เซนเซอร์วัดได้จากหน่วยเซลเซียสเป็นฟาเรนไฮต์ โดยนำมาคูณด้วย 1.8 และบวกด้วย 32 จะได้ค่าอุณหภูมิเป็นหน่วยฟาเรนไฮต์ ก่อนบันทึกลงใน Device Shadow หรือ Timeseries Database

- **type** คือ ชนิดของข้อมูลในฟิลด์นั้น ๆ ได้แก่ number, string, boolean, array, object

สามารถอ่านข้อมูลเพิ่มเติมเกี่ยวกับ Device Schema ได้ที่

<https://docs.netpie.io/device-config.html#device-schema>

MQTT

การเชื่อมต่อ Platform ผ่าน MQTT (Message Queuing Telemetry Transport) ซึ่งเป็น Protocol ที่มีขนาดเล็กและได้รับความนิยมสำหรับการสื่อสารแบบ M2M (Machine to Machine) โดยสามารถใช้ MQTT Library ตัวใดก็ได้ที่รองรับกับ Device ที่ใช้งานอยู่ การเชื่อมต่อของ MQTT จะต้องใช้ 4 Parameters คือ Host, Client ID, Username และ Password โดยให้ระบุแต่ละค่าดังนี้

Host	mqtt.netpie.io
Port	1883 (mqtt), 1884 (mqtts)
Client ID	Client ID ของ Device ที่สร้างขึ้นใน NETPIE
Username	Token ของ Device ที่สร้างขึ้นใน NETPIE
Password	ยังไม่ต้องระบุ (ใช้สำหรับกรณีที่ต้องการตรวจสอบที่เพิ่มมากขึ้น)

MQTT API จะมีลักษณะการใช้งานเป็นแบบ Publish / Subscribe โดย Publish จะเป็นการส่งข้อมูลไปยัง Topic ที่ต้องการ ส่วน Subscribe จะเป็นการอธิบายข้อมูลใน Topic ที่ต้องการ การสั่ง Subscribe Topic ได้ก็ตามทำเพียงครั้งเดียว ก็จะได้รับข้อมูลใน Topic นั้นไปตลอดจนกว่าจะสั่ง Unsubscribe Topic นั้น หรือการเชื่อมต่อกับ Platform หยุดลง

องค์ประกอบสำคัญที่จำเป็นต้องทราบสำหรับการใช้งาน MQTT API คือ Topic เพราะ Publish / Subscribe จำเป็นต้องระบุ Topic ที่ต้องการ Topic จะหน้าที่เหมือน Endpoint บน MQTT Broker เพื่อให้ MQTT Client มาเข้ามายังและสื่อสารกัน โดยจะรองรับคุณสมบัติดังต่อไปนี้

- **QoS (Quality of Service) 3 ระดับ** คือ ข้อตกลงระหว่างผู้ส่ง (Publisher) และผู้รับ (MQTT Broker) ในการรับประกันการส่งข้อมูล
- **QoS Level 0 :** การส่งข้อมูลที่มีการรับประกันผลในระดับต่ำสุด หรือเป็นข้อมูลที่ต้องการความรวดเร็วในการส่ง คือ ไม่ต้องการการตอบกลับว่าข้อมูลถึง MQTT Broker แล้ว

- **QoS Level 1 :** การส่งข้อมูลที่มีการรับประกันผลในระดับที่ทุกครั้งของการส่งข้อมูล ต้องมีการตอบกลับเสมอเพื่อยืนยันว่าข้อมูลได้ส่งไปถึง MQTT Broker และ ถ้าการตอบกลับสูญหาย ผู้ส่งจะทำการส่งข้อมูลไปใหม่จนกว่าจะได้รับการตอบกลับ นั่นหมายความว่า MQTT Broker จะได้รับข้อมูลอย่างน้อย 1 ครั้ง แน่นอน แต่ข้อมูลเดียวกันอาจจะได้รับมากกว่า 1 ครั้งด้วยเห็นกัน
- **QoS Level 2 :** การส่งข้อมูลที่มีการรับประกันผลระดับสูงสุด ข้อมูลมีความสำคัญมาก คือ ทุกครั้งของการส่งข้อมูลจะมีการยืนยันว่าถูกส่งไปถึง MQTT Broker อย่างแน่นอนและได้รับข้อมูลครั้งเดียว
- **Shared Subscription** คือ การส่งข้อมูลกระจายไปได้ทุกหนندที่ Subscribe Topic เดียวกัน
- **Transparent Virtual Host** คือ การ Publish / Subscribe ไปที่ Topic เดียวกัน ถ้า Device อยู่ต่างกลุ่ม ก็จะเหมือนเป็นคนละ Topic กัน

โครงสร้าง Topic ใน NETPIE Platform สามารถแยกได้เป็น 3 ประเภท ดังนี้

Message API Topic

เป็นการกำหนด Topic สำหรับ Publish / Subscribe ข้อมูลเพื่อสื่อสารระหว่าง Devices ที่อยู่ภายใต้ Group เดียวกัน รูปแบบการใช้งานให้เขียนต้น Topic ด้วย @msg ตามด้วยโครงสร้าง Topic ที่ต้องการ ดังนี้

publish	publish @msg/{any}/{topic}
subscribe	subscribe @msg/{any}/{topic}
ตัวอย่าง Topic	@msg/myhome/bedroom/lamp @msg/sensor/temp @msg/john

Shadow API Topic

เป็น Topic ที่เกี่ยวข้องกับการจัดการ Device Shadow สามารถแยกได้เป็น Publish และ Subscribe โดย Publish จะใช้กรณีที่ต้องการขอข้อมูลหรืออัพเดทข้อมูลใน Device Shadow ส่วน Subscribe จะเป็นการรับข้อมูลในกรณีที่มีการ Publish ไปขอข้อมูล หรือในกรณีที่มีการเปลี่ยนข้อมูล Device Shadow และได้ทำการ Subscribe ไว้ ซึ่งการใช้งานจะมีรายละเอียด ดังนี้

- **Private Channel Topic** คือ ช่องทางพิเศษสำหรับการตอบกลับ (Response) หรือรับข้อมูลทุกอย่างที่เกิดขึ้นกับตัวเอง เช่น Device Shadow ตัวเองมีการเปลี่ยนแปลง เป็นต้น โดยรูปแบบการใช้งานให้เขียนต้น Topic ด้วย @private มีลักษณะตามนี้ @private/{response topic} จะมีเพียงการ Subscribe เท่านั้น Response Topic สำหรับ Subscribe @private มีดังนี้

Subscribe Topic	คำอธิบาย
@private/#	การอ่านทุกข้อมูลที่ Publish มาอย่าง Topic ที่ขึ้นต้นด้วย @private/ รวมถึงข่าวสารต่างๆ ที่ Platform ต้องการ แจ้งให้ทราบก็จะถูก Publish มาอย่าง Topic นี้
@private/shadow/data/get/response	การอ่าน Device Shadow Data เมื่อมีการร้องขอ ข้อมูลไป
@private/shadow/batch/update/response	การอ่านข้อความตอบกลับ กรณีอัพเดท Shadow แบบ เป็นชุดข้อมูล (Shadow Batch Update)

- Shadow Topic คือ Topic ที่ใช้สำหรับจัดการ Device Shadow ของตัวเอง Topic ที่เกี่ยวข้องมีดังนี้

Subscribe Topic	คำอธิบาย
@shadow/data/get	เป็นการร้องขอข้อมูล Shadow Data ของตัวเองแบบ ทั้งหมด โดยการอ่านข้อมูลให้ Subscribe Topic @private/# หรือ @private/shadow/data/get/response เพื่อรับ ข้อมูล (ใช้ในกรณีที่เป็น Shadow ตัวเองท่านนั้น)
@shadow/data/update	เป็นการอัพเดทค่าใน Shadow Data โดยส่ง Payload ดังนี้ { "data":{ "field name 1": value 1, "field name 2": value 2, ..., "field name n": value n }} ถ้าต้องการได้รับข้อมูลเมื่อค่าต่าง ๆ ใน Shadow Data ถูกอัพเดทให้ Subscribe Topic @shadow/data/updated รอไว้
@shadow/batch/update	เป็นการอัพเดทค่าใน Shadow แบบเป็นชุดข้อมูล (Shadow Batch Update)

สามารถอ่านข้อมูลเพิ่มเติมเกี่ยวกับ MQTT ของ NETPIE ได้ที่ <https://docs.netpie.io/mqtt-api.html>

Dashboard

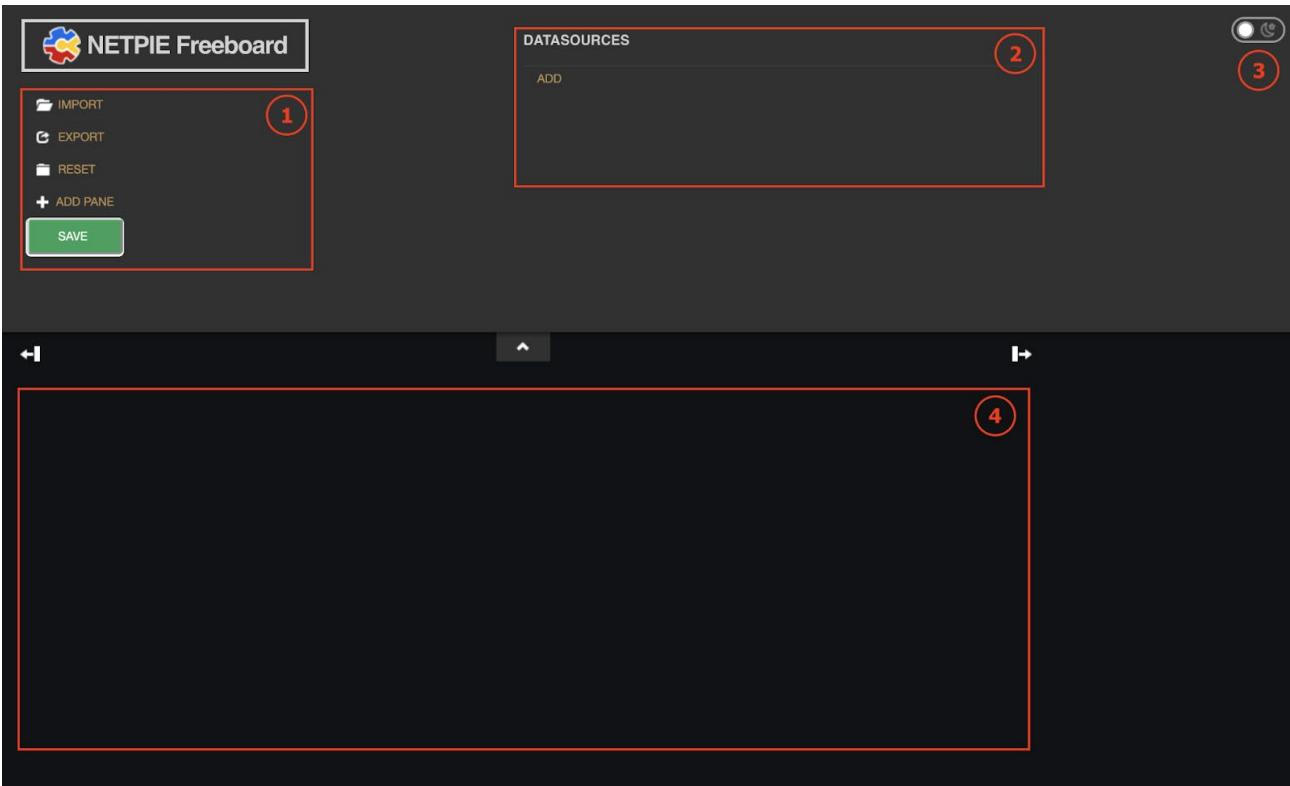
ใช้สำหรับนำข้อมูลที่เก็บอยู่ใน Platfrom มาแสดงผลในรูปแบบต่างๆ เป็นเหมือนช่องทางให้ผู้ใช้สามารถติดตามหรือควบคุมการทำงานของ Device ของตัวเอง โดย Dashboard ที่มีให้ใช้งาน ณ ปัจจุบันมีเพียงประเภทเดียว คือ Freeboard ซึ่งสามารถเข้าใช้งานได้ด้วยการคลิกที่เมนู “Freeboard” ในแท็บซ้ายมือ ก็จะปรากฏหน้าจอแสดงรายรายชื่อ Freeboard ทั้งหมดที่เคยสร้างไว้ภายใน Project นั้นๆ (ถ้ามี) ดังรูป

The screenshot shows the NETPIE Platform interface. On the left, there's a sidebar with navigation links: Test (selected), Overview, Device List, Device Groups, Freeboard (selected), Event Hooks, and Setting. The main area is titled "Test / freeboard". It has a header with a search bar and a "Create" button. Below that is a table with columns "Name" and "Create Date". A message "No Data" is displayed. The overall theme is light gray with blue highlights for selected items.

จากรูปด้านบน หลังรายชื่อ Freeboard แต่ละรายการ ถ้านำเม้าส์ไปวางไว้เหนือรายการใด จะปรากฏปุ่ม “Edit” และ “Delete” สำหรับทำการแก้ไขข้อมูลที่ว่าไปและลบ Freeboard นั้นๆ ตามลำดับ ถ้าต้องการสร้าง Freeboard ใหม่ให้คลิกที่ปุ่ม “Create” มุมบนขวาเมื่อของหน้าจอ ก็จะปรากฏหน้าจอสำหรับให้กรอกข้อมูลที่ว่าไปของ Freeboard ดังรูป

The dialog box is titled "Create Dashboard". It contains two input fields: one for "Name" with the value "ESP8266" and another for "Description". At the bottom right are "Cancel" and "Create" buttons. The background is white with a clean, modern design.

สำหรับการเข้าไปเช็ต Configuration ของ Freeboard ให้คลิกที่รายการที่ต้องการเข้าไปดำเนินการ โดยหน้าจอสำหรับจัดการ Configuration ของ Freeboard มีรายละเอียดดังนี้



จากรูปด้านบน หน้าจอสำหรับจัดการ Freeboard แบ่งเป็น 4 ส่วนใหญ่ๆ ดังนี้

1. เมนูสำหรับจัดการส่วนต่างๆ ของ Freeboard ได้แก่
 - a. เมนู “IMPORT” ใช้สำหรับนำเข้า Configuration Code ซึ่งจะอยู่ในรูปแบบ JSON File ที่ส่งออกไปจาก Freeboard ที่เคยเช็ต Configuration ไว้แล้ว เมื่อคลิกแล้วจะปรากฏ browse file ให้เลือกไฟล์ที่ต้องการนำเข้า

- b. เมนู “EXPORT” ใช้สำหรับส่งออก Configuration Code ที่เคยเซ็ตไว้แล้ว ซึ่งจะอยู่ในรูปแบบ JSON File เช่นกัน เพื่อนำเข้า (เมนู “IMPORT”) สู่ Freeboard อีน หรือเพื่อการสำรองข้อมูล (Backup)
 - c. เมนู “RESET” ใช้สำหรับล้างค่า Configuration Code ที่เคยเซ็ตไว้แล้วทั้งหมด
 - d. เมนู “ADD PANE” ใช้สำหรับสร้าง Block หรือ Panel ที่ใช้จัดกลุ่มการแสดงผลแต่ละ Widget ที่จะสร้างใน Freeboard
 - e. ปุ่ม “SAVE (สีเขียว)” ใช้สำหรับบันทึกการเปลี่ยนแปลงทุกอย่างที่มีการดำเนินการไป คลิกปุ่มนี้ ทุกครั้งก่อนออกจากหรือปิดหน้าจอเพื่อบันทึก Configuration ต่างๆ ที่ได้เปลี่ยนแปลงไป
2. ส่วนจัดการ “DATASOURCES” ใช้สำหรับสร้างการเชื่อมต่อเพื่อติดข้อมูลจาก Device ต่างๆ ใน Platform มาแสดงที่ Freeboard รายละเอียดจะอธิบายเพิ่มเติมในหัวข้อถัดไป
3. Theme ใช้สำหรับเปลี่ยนรูปสีของ Freeboard มีให้เลือก 2 โทนสี คือ Dark Color (ค่า Default) และ Light Color
4. ส่วนจัดการและแสดงผล Freeboard ใช้สำหรับจัดการ Widget ต่างๆ ที่จะนำค่ามาแสดง และยังเป็นส่วนที่ใช้ดู Freeboard (View) ที่เข็คค่าไว้แล้ว
- เมื่อกดเปลี่ยน Theme จะได้สีของ Freeboard ที่แตกต่างไป



ทดลองใช้ Freeboard กับ Device Shadow

ในส่วนนี้ จะทำการทดลองการใช้ Freeboard โดยการส่งข้อมูลข้อความเข้า Device Shadow จากนั้นจึงแสดงผลข้อมูลข้อความนั้นใน Freeboard โดยเริ่มจากการสร้าง Device Schema ดังนี้

Schema Content:

```

1 " {
2   "additionalProperties": true,
3   "properties": {
4     "message": {
5       "operation": {
6         "store": {
7           "ttl": "2d"
8         }
9       },
10      "type": "string"
11    }
12  }
13 }
```

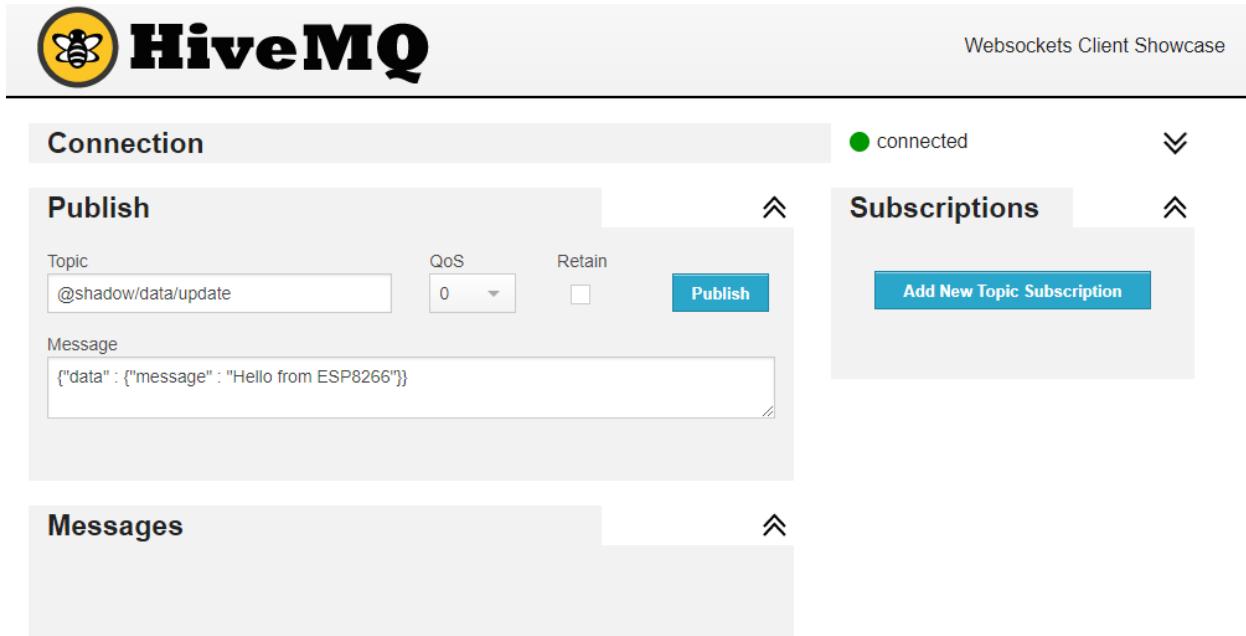
Key Configuration (Top Window):

Key		
Client ID	:	a25caa14-23fd-40f0-94c1-50dfd16cb50b
Token	:	z2hwv6TG2nNbUXg91ejbvm8C5SApo57a
Secret	:	3d4A7ocot)UIDz#(~wz8TvFJ74s#inxO
Status	:	Online
Enable	:	<input checked="" type="checkbox"/>

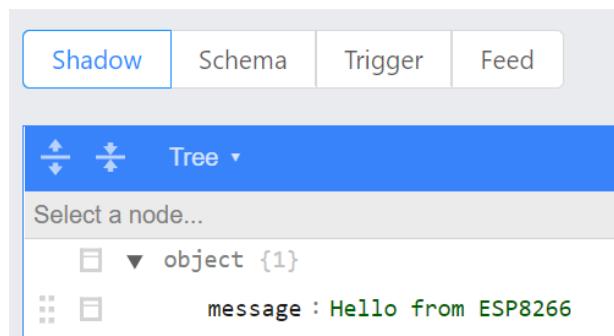
Key Configuration (Bottom Window):

Key		
Client ID	:	a25caa14-23fd-40f0-94c1-50dfd16cb50b
Token	:	z2hwv6TG2nNbUXg91ejbvm8C5SApo57a
Secret	:	3d4A7ocot)UIDz#(~wz8TvFJ74s#inxO
Status	:	Online
Enable	:	<input checked="" type="checkbox"/>

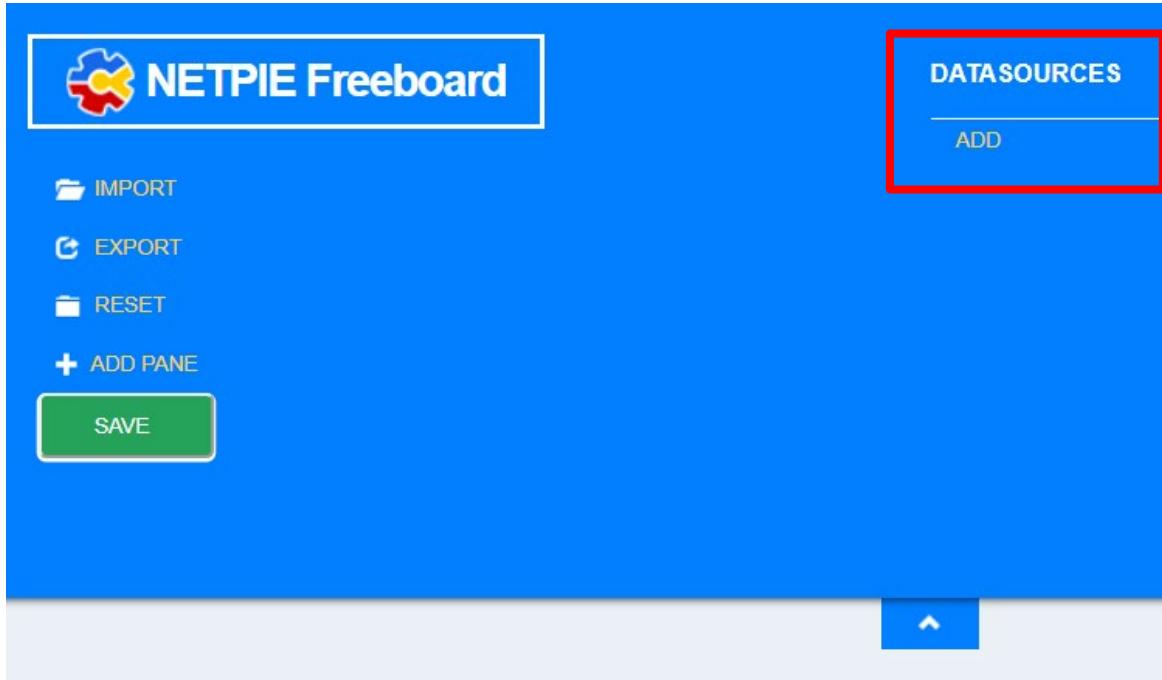
เมื่อตั้งค่า Device Schema เสร็จแล้ว ต่อไปให้ตั้งค่า Device ใน HiveMQ เพื่อให้พร้อมสำหรับส่งข้อมูลข้อความขึ้น Device Shadow โดย Topic และ Message ที่กรอกสามารถกรอกได้ดังรูป



กด Publish ใน HiveMQ จนนั้นให้ไปดูผลลัพธ์ใน Device Shadow ว่าผลลัพธ์เป็นไปดังรูปต่อไปนี้หรือไม่ หากไม่เป็นไปตามรูป ให้ลองกด Refresh หน้าต่างใหม่



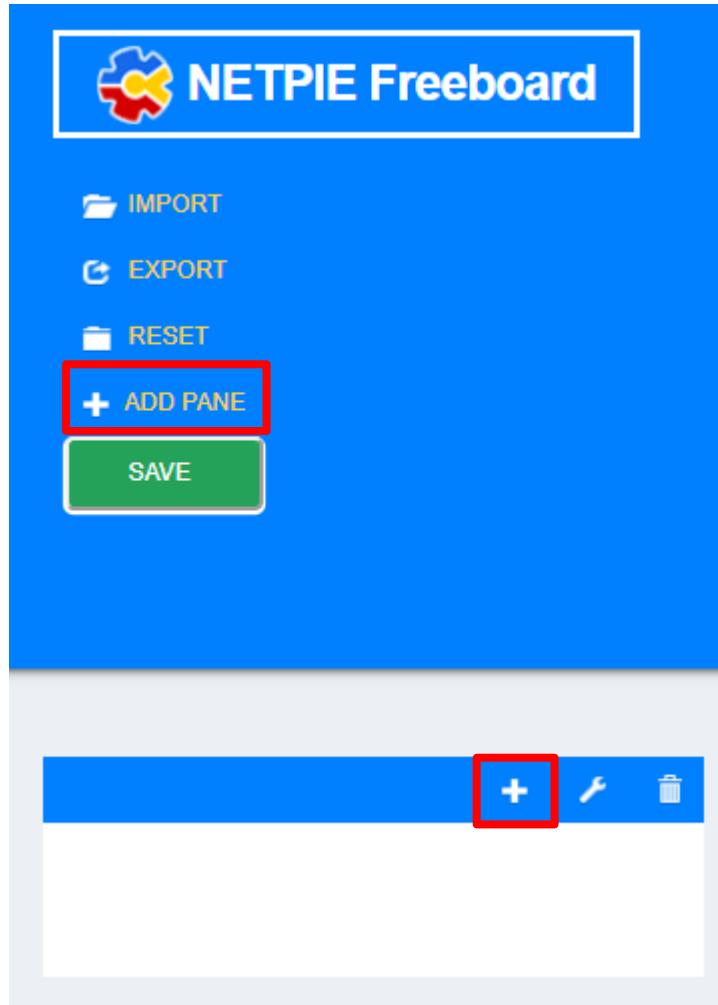
เมื่อข้อความถูกส่งเข้า Device Shadow และ ให้ทำการตั้งค่าในส่วนของ Freeboard เริ่มต้นที่การเพิ่ม Datasource โดยป้อนข้อมูลที่จำเป็นให้หมด



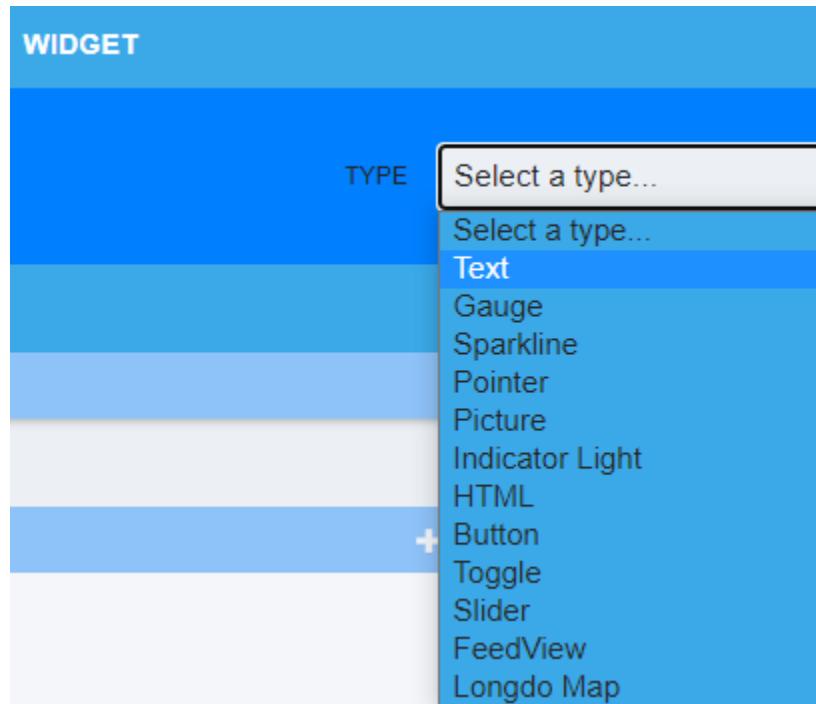
DATASOURCE

NAME	Data_ESP8266_2
DEVICE ID	03ca2f5f-b8cf-4cf4-9130-5e409219f232
Client ID ของ Device ที่ต้องการอ่านข้อมูล	
DEVICE TOKEN	5Ji3q42GWYBTcRxU4wP8Xh8XStsnWf3c
Token ของ Device ที่ต้องการอ่านข้อมูล	
SUBSCRIBED TOPICS	@shadow/data/update
Topic ที่ต้องการ Subscribe	
FEED	<input checked="" type="checkbox"/> NO
SINCE	6
Hour <input type="button" value="▼"/>	
Display data points since ... ago.	

จากนั้นกดปุ่ม Add Pane เพื่อเพิ่มช่องสำหรับใส่ Widget ที่เราต้องการ เมื่อสร้าง Pane เสร็จแล้ว ให้กด สัญลักษณ์ + ตรง Pane ที่สร้าง เพื่อเพิ่ม Widget ที่เราต้องการใช้งาน



หน้าต่างการเลือก Widget จะปรากฏให้มาให้เห็น



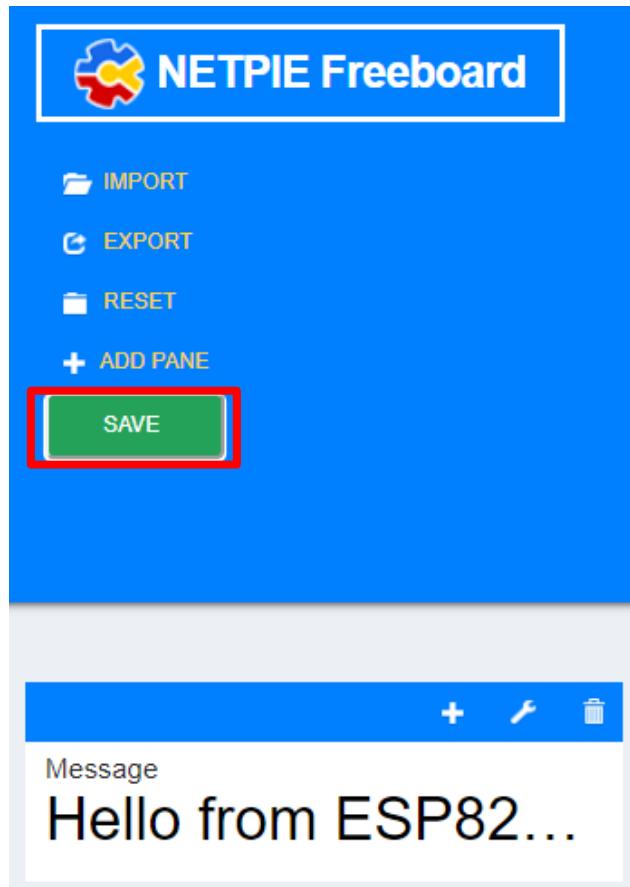
เลือก Widget ที่ต้องการใช้งาน ซึ่งในที่นี่เราจะใช้ Text เพื่อแสดงผลข้อความที่ส่งขึ้น Device Shadow เมื่อเลือกเสร็จแล้ว ให้กรอกข้อมูลในช่องต่างๆดังรูป

TYPE	Text
TITLE	Message
SIZE	Regular
INCLUDE SPARKLINE	Data_ESP8266_2
ANIMATE VALUE CHANGES	<input checked="" type="checkbox"/> TICK <input checked="" type="checkbox"/> DASH
UNITS	

+ DATASOURCE X JS EDITOR



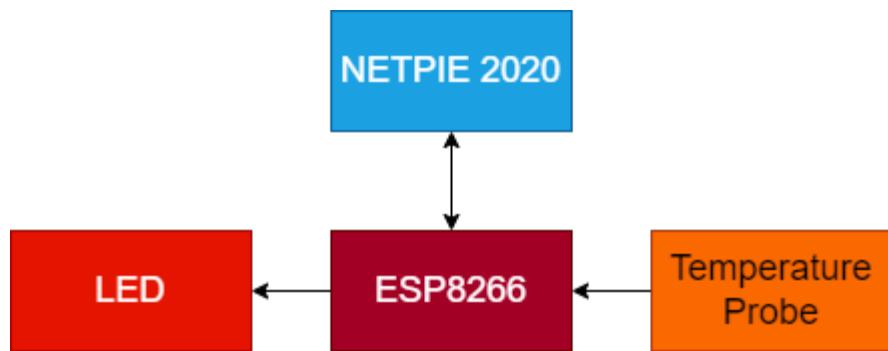
จากนั้นจึงกด Save ผลลัพธ์ที่ได้จะต้องเป็นไปตามรูป



สามารถหาข้อมูล และ ตัวอย่างการใช้งานควบคู่กับบอร์ดได้ที่ <https://netpie.io/guide>

Workshop 18 ระบบตรวจสอบอุณหภูมิโดยใช้ NETPIE 2020

ในการทดลองนี้ จะเป็นการทดลองใช้ Temperature Probe เพื่อวัดอุณหภูมิส่งไปยัง NETPIE และสามารถควบคุมหลอด LED ผ่าน NETPIE ได้



ขั้นตอนแรก จะเริ่มทำในส่วนของ NETPIE ก่อน โดยสิ่งที่เราต้องทำคือการสร้าง Project ที่ชื่อว่า “ESP8266_TEST” ดังรูป

Create Project X

* Name:

Description:

Tag:

Cancel
Create

เมื่อสร้าง Project เสร็จแล้ว ต้องมาให้สร้าง Device ใน Project เพื่อรับการเชื่อมต่อที่มาจากการ ESP8266 ของเรา โดยในที่นี่จะตั้งชื่อว่า “ESP8266”

Create

* Name:

Description:

Tag: + New Tag

หลังจากสร้าง Device ของ ESP8266 แล้ว ให้ตั้งค่าในส่วนของ Device Schema เพื่อรับข้อมูลที่ ESP8266 จะส่งมาดังนี้



```

Code ▾
1 [ "additionalProperties": true,
2   "properties": {
3     "temperature": {
4       "operation": {
5         "store": {
6           "ttl": "2d"
7         }
8       },
9       "type": "number"
10      },
11     "ledState": {
12       "operation": {
13         "store": {
14           "ttl": "2d"
15         }
16       },
17       "type": "string"
18     }
19   }
20 ]
21 ]

```

หลังจากตั้งค่า Device Schema แล้ว งานนี้จึงเริ่มเขียนโปรแกรมให้กับ ESP8266 เพื่อรับส่งข้อมูลกับ NETPIE โดยโปรแกรมสามารถเขียนได้ดังนี้

```

1 #include <ESP8266WiFi.h>
2 #include <PubSubClient.h>
3 #include <OneWire.h>
4 #include <DallasTemperature.h>
5
6 #define ONE_WIRE_BUS 4
7 #define LED 13
8
9 const char* ssid = ""; ;                                //***change
10 const char* password = "";                            //***change
11 const char* mqtt_server = "broker.netpie.io";
12 const int mqtt_port = 1883;
13 const char* mqtt_Client = "";                         //***change
14 const char* mqtt_Token = "";                          //***change
15 const char* mqtt_Secret = "";                        //***change
16
17 OneWire oneWire(ONE_WIRE_BUS);
18 DallasTemperature tempProbe(&oneWire);
19 WiFiClient espClient;
20 PubSubClient client(espClient);

```

บรรทัด ที่	คำอธิบาย
1	เพิ่ม Library ESP8266WiFi เพื่อใช้งาน WiFi บนบอร์ด ESP8266
2	เพิ่ม Library PubSubClient เพื่อใช้งานกับการสื่อสารแบบ MQTT
3	เพิ่ม Library OneWire เพื่อใช้งานกับการสื่อสารแบบ One-Wire Communication
4	เพิ่ม Library DallasTemperature เพื่อใช้งานกับ Temperature Probe Sensor
6	กำหนดค่าคงที่ชื่อ ONE_WIRE_BUS เป็น 4 เพื่อรับรู้ว่า ขารับส่งข้อมูลของ Temperature Probe Sensor อยู่ที่ GPIO4 (D2)
7	กำหนดค่าคงที่ LED เป็น 13 เพื่อรับรู้ว่า ขาควบคุม LED อยู่ที่ GPIO13 (D7)

บรรทัด ที่	คำอธิบาย
9	กำหนดค่าคงที่ชื่อ ssid เพื่อระบุชื่อของสัญญาณอินเทอร์เน็ตที่จะเชื่อมต่อ
10	กำหนดค่าคงที่ชื่อ password เพื่อระบุรหัสผ่านของสัญญาณอินเทอร์เน็ตที่จะเชื่อมต่อ
11	กำหนดค่าคงที่ชื่อ mqtt_server เพื่อระบุที่อยู่ของเซิฟเวอร์ MQTT ที่จะเชื่อมต่อ
12	กำหนดค่าคงที่ชื่อ mqtt_port เพื่อระบุพอร์ตของเซิฟเวอร์ MQTT ที่จะเชื่อมต่อ
13	กำหนดค่าคงที่ชื่อ mqtt_Client เพื่อระบุ Client ID ของอุปกรณ์ที่จะเชื่อมต่อ
14	กำหนดค่าคงที่ชื่อ mqtt_Token เพื่อระบุ Token ของอุปกรณ์ที่จะเชื่อมต่อ
15	กำหนดค่าคงที่ชื่อ mqtt_Secret เพื่อระบุ Secret ของอุปกรณ์ที่จะเชื่อมต่อ
17	สร้าง Object ของ OneWire ชื่อ onewire เพื่อสร้างการเชื่อมต่อแบบ One-Wire Communication โดยป้อนค่าคงที่ ONE_WIRE_BUS เข้าไปเพื่อระบุขาพินที่จะใช้ในการสื่อสาร
18	สร้าง Object ของ DallasTemperature ชื่อ tempProbe เพื่อสร้างชุดคำสั่งสำหรับในการอ่านค่าอุณหภูมิจาก Temperature Probe Sensor โดยป้อน onewire เข้าไปเพื่อให้ tempProbe อ่านค่าอุณหภูมิผ่านทาง onewire
19	สร้าง Object ของ WiFiClient ชื่อ espClient เพื่อสร้างชุดคำสั่งสำหรับใช้งาน WiFi บน ESP8266
20	สร้าง Object ของ PubSubClient ชื่อ client เพื่อสร้างชุดคำสั่งสำหรับสำหรับการสื่อสารผ่าน MQTT Protocol โดยป้อน espClient เพื่อให้ client ใช้งาน WiFi บน ESP8266 ผ่านทาง espClient

```

22 char msg[100];
23 float temp = 0.0;
24 bool isReady = false;
25
26 void reconnect();
27 void onoff(int ); //sent LED status to netpie
28 void callback(char* , byte* , unsigned int ); //get data from netpie
29 float readTemp();
30 void displayTemp();
31 void displayLED();

```

บรรทัด ที่	คำอธิบาย
22	สร้าง array of char ขึ้นมาชื่อ msg มีขนาด 100 elements เพื่อรับข้อมูลที่เซิฟเวอร์ MQTT จะส่งมา
23	สร้างตัวแปร float ขึ้นมาชื่อ temp เพื่อรับข้อมูลที่ได้จาก Temperature Probe Sensor โดยกำหนดค่าเริ่มต้นไว้ที่ 0.0
24	สร้างตัวแปร boolean ขึ้นมาชื่อ isReady เพื่อบอกสถานะของโปรแกรมว่าทำงานจบในส่วนของฟังก์ชัน void setup() หรือยัง โดยกำหนดให้ค่าเริ่มต้นมีค่าเป็น false
26	สร้างฟังก์ชัน void ชื่อ reconnect เพื่อเชื่อมต่อ กับเซิฟเวอร์ MQTT ในกรณีที่ขาดการเชื่อมต่อ
27	สร้างฟังก์ชัน void ชื่อ onoff เพื่อควบคุม LED และ ส่งข้อมูลกลับไปให้เซิฟเวอร์ MQTT
28	สร้างฟังก์ชัน void ชื่อ callback เพื่อรับข้อมูลที่มาจากเซิฟเวอร์ MQTT
29	สร้างฟังก์ชัน float ชื่อ readTemp เพื่อควบคุมการอ่านค่าอุณหภูมิจาก Temperature Probe Sensor
30	สร้างฟังก์ชัน void ชื่อ displayTemp เพื่อแสดงผลข้อมูลอุณหภูมิที่ได้รับอุปมาที่ Serial Monitor
31	สร้างฟังก์ชัน void ชื่อ displayLED เพื่อแสดงผลสถานะของ LED อุปมาที่ Serial Monitor

```

33 void setup() {
34   Serial.begin(115200);
35   Serial.println("Dallas Temperature IC Control Library");
36   tempProbe.begin();
37   pinMode(LED, OUTPUT);
38
39   Serial.println("Connecting to ");
40   Serial.println(ssid);
41   WiFi.mode(WIFI_STA);
42   WiFi.begin(ssid, password);           // access wifi
43   while (WiFi.status() != WL_CONNECTED) //check disconnect
44   {
45     delay(500);
46     Serial.print(".");
47   }
48   Serial.println(" WiFi connected");
49   Serial.println("IP address: ");
50   Serial.println(WiFi.localIP());
51   client.setServer(mqtt_server, mqtt_port);
52   client.setCallback(callback);
53
54   isReady = true;
55 }
```

บรรทัด ที่	คำอธิบาย
33	เข้าสู่ฟังก์ชัน void setup()
34	ตั้ง BaudRate ของ Serial Monitor ไว้ที่ 115200
35	แสดงข้อมูลความอุ่นทาง Serial Monitor โดยเมื่อจบข้อมูลให้ขึ้นบรรทัดใหม่
36	เปิดการใช้งาน Temperature Probe Sensor
37	ตั้งรูปแบบการทำงานของขา LED เป็น OUTPUT
39	แสดงข้อมูลความอุ่นทาง Serial Monitor โดยเมื่อจบข้อมูลให้ขึ้นบรรทัดใหม่

บรรทัด ที่	คำอธิบาย
40	แสดงชื่อของอินเทอร์เน็ตที่เข้ามายัง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
41	ตั้งรูปแบบการทำงานของ WiFi บน ESP8266 ให้ทำงานแบบ station
42	เปิดการใช้งาน WiFi บน ESP8266 โดยให้เข้ามายัง Serial Monitor ที่ระบุ และ ป้อนรหัสตามที่ระบุไว้
43	สร้าง while loop โดยกำหนดเงื่อนไขให้ตรวจสอบสถานะการเชื่อมต่อว่าเชื่อมต่ออินเทอร์เน็ตสำเร็จ หรือไม่
44	เข้าสู่ while loop
45	หน่วงเวลาไว้ 500 มิลลิวินาที
46	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
47	จบการทำงาน while loop
48	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
49	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
50	แสดง local IP Address ออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
51	ตั้งค่าเซิฟเวอร์ MQTT และพอร์ตปลายทางที่จะเชื่อมต่อ
52	ระบบฟังก์ชันที่จะทำงานเมื่อมีข้อมูลส่งมาจากเซิฟเวอร์ MQTT
54	กำหนดค่าให้ isReady เป็น true เพื่อระบุให้โปรแกรมรู้ว่า void setup() ทำงานเสร็จสิ้นแล้ว
55	จบการทำงานของ void setup()

```

57 void loop() {
58   if (!client.connected())
59     reconnect();
60
61   client.loop();
62
63   if (millis() % 1000 == 0 && isReady) {
64     temp = readTemp();
65     String data = "{\"data\": {\"temperature\": " + String(temp) + "}}";
66     data.toCharArray(msg, data.length() + 1);
67     client.publish("@shadow/data/update", msg); // sent data to shadow netpie
68     displayTemp();
69     displayLED();
70   }
71 }
```

บรรทัด ที่	คำอธิบาย
57	เริ่มต้นการทำงานของ void loop()
58	สร้างเงื่อนไขดักไว้ ในกรณีที่การเชื่อมต่อ กับเซิฟเวอร์ MQTT มีปัญหา ให้ทำการคำสั่งภายใน
59	เรียกใช้ฟังก์ชัน reconnect()
61	เริ่มต้นการสื่อสารกับเซิฟเวอร์ MQTT
63	สร้างเงื่อนไขการแสดงผล และ ส่งข้อมูลไปที่เซิฟเวอร์ MQTT โดยให้ทำงานทุกๆ 1 วินาที และ void setup() ต้องทำงานเสร็จสมบูรณ์แล้วเท่านั้น
64	อ่านค่าอุณหภูมิโดยใช้ฟังก์ชัน readTemp() และมาเก็บไว้ที่ตัวแปร temp
65	สร้างตัวแปร String ชื่อ data เพื่อประกอบข้อมูลให้พร้อมส่ง โดยกำหนดค่าเป็นโครงสร้างข้อมูลแบบ JSON ในรูปแบบ String
66	นำตัวแปร data ไปแปลงเป็น array of char โดยเก็บไว้ที่ msg
67	ส่งข้อมูลแบบ MQTT ออกไปที่เซิฟเวอร์ MQTT โดยข้อมูลที่ส่งไปจะถูกบันทึกที่ shadow ของ NETPIE

บรรทัด ที่	คำอธิบาย
68	แสดงผลข้อมูลอุณหภูมิทาง Serial Monitor
69	แสดงผลสถานะ LED ทาง Serial Monitor
70	จบการทำงานของเงื่อนไขการแสดงผล และ ส่งข้อมูล
71	จบการทำงานของ void loop()

```

73 void reconnect()
74 {
75     while (!client.connected()) {
76         Serial.print("Attempting MQTT connection...");
77         if (client.connect(mqtt_Client, mqtt_Token, mqtt_Secret)) {
78             Serial.println("connected");
79             client.subscribe("@msg/ledState");
80         }
81         else
82         {
83             Serial.print("failed, rc=");
84             Serial.print(client.state());
85             Serial.print("Try again in 5 sec");
86             delay(5000);
87         }
88     }
89 }
```

บรรทัด ที่	คำอธิบาย
73	ประกาศการทำงานของฟังก์ชัน void reconnect()
74	เริ่มการประกาศการทำงาน
75	สร้าง while loop เพื่อตรวจสอบการเชื่อมต่อระหว่างอุปกรณ์กับเซิฟเวอร์ MQTT จนกว่าการเชื่อมต่อจะเป็นปกติ

บรรทัด ที่	คำอธิบาย
76	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
77	เชื่อมต่อเข้าสู่เซิฟเวอร์ MQTT ปลายทาง พร้อมทั้งตรวจสอบผลการเชื่อมต่อ
78	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
79	ให้อุปกรณ์ subscribe ข้อมูลที่ส่งมาใน topic ที่ระบุ
80	จบการทำงานของ if
81	หากการเชื่อมต่อในบรรทัดที่ 77 ไม่สำเร็จ ให้ทำงานตามที่ระบุใน else
82	เริ่มต้นระบุการทำงานของ else
83	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
84	แสดงผลลัพธ์การเชื่อมต่อออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
85	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
86	หน่วงเวลาไว้ 5 วินาที
87	จบการทำงานของ else
88	จบการทำงานของ while loop
89	จบการประกาศการทำงานของฟังก์ชัน void reconnect()

```

91 void callback(char* topic, byte* payload, unsigned int length) //get data from netpie
92 {
93     Serial.print("Message arrived [");
94     Serial.print(topic);
95     Serial.print("]: ");
96     String msg;
97     for (int i = 0; i < length; i++) {
98         msg = msg + (char)payload[i];
99     }
100    Serial.println(msg);
101    if ( String(topic) == "@msg/ledState")
102        if (msg == "on") {
103            onoff(true);
104            client.publish("@shadow/data/update", "{\"data\" : {\"ledState\" : \"on\"}}");
105        }
106        else if (msg == "off") {
107            onoff(false);
108            client.publish("@shadow/data/update", "{\"data\" : {\"ledState\" : \"off\"}}");
109        }
110 }

```

บรรทัด ที่	คำอธิบาย
91	ประกาศการทำงานของฟังก์ชัน void callback(char* topic, byte* payload, unsigned int length)
92	เริ่มต้นการประกาศการทำงาน
93	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
94	แสดง topic ที่เชิฟเวอร์ MQTT ส่งมาให้ออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
95	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
96	สร้างตัวแปร String ชื่อ msg (คำลั่ตัวกับที่ใช้ใน void loop())
97	เริ่มต้น for loop เพื่อรับข้อมูลที่ส่งมาเก็บไว้ที่ msg โดยวนรอบตามความยาวของข้อมูลที่ส่งมา
98	รับข้อมูล payload แต่ละตัว มาเก็บไว้ใน msg โดยเรียงลำดับกัน
99	จบการทำงานของ for loop
100	แสดงข้อมูลที่ msg รับมาออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่

บรรทัด ที่	คำอธิบาย
101	สร้างเงื่อนไข if เพื่อตรวจสอบว่า topic ที่รับมาเป็น topic ที่ใช้สั่งการ LED หรือไม่
102	สร้างเงื่อนไข if เพื่อตรวจสอบว่าข้อมูลที่ส่งมาให้ คือการสั่งเปิด LED หรือไม่
103	เรียกใช้ฟังก์ชัน onoff() เพื่อควบคุม LED โดยป้อน true เข้าไป
104	ส่งข้อมูลแบบ MQTT ออกไปที่เซิฟเวอร์ MQTT โดยข้อมูลที่ส่งไปจะถูกบันทึกที่ shadow ของ NETPIE
105	จบการทำงานของ if ในบรรทัดที่ 102
106	สร้างเงื่อนไข else if เพื่อตรวจสอบว่าข้อมูลที่ส่งมาให้ คือการสั่งปิด LED หรือไม่
107	เรียกใช้ฟังก์ชัน onoff() เพื่อควบคุม LED โดยป้อน false เข้าไป
108	ส่งข้อมูลแบบ MQTT ออกไปที่เซิฟเวอร์ MQTT โดยข้อมูลที่ส่งไปจะถูกบันทึกที่ shadow ของ NETPIE
109	จบการทำงานของ else if ในบรรทัดที่ 106
110	จบการประกาศการทำงานของฟังก์ชัน void void callback(char* topic, byte* payload, unsigned int length)

```

112 void onoff(bool led) //sent LED status to netpie
113 {
114     if (led)
115     {
116         Serial.println("Turn on LED");
117         digitalWrite(LED, HIGH);
118     }
119     else if (!led)
120     {
121         Serial.println("Turn off LED");
122         digitalWrite(LED, LOW);
123     }
124 }
```

บรรทัดที่	คำอธิบาย
112	ประกาศการทำงานของฟังก์ชัน void onoff(bool)
113	เริ่มต้นการประกาศการทำงาน
114	สร้างเงื่อนไข if เพื่อตรวจสอบว่า led เป็น true หรือไม่
115	เริ่มประกาศการทำงานของเงื่อนไข if
116	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
117	สั่งเปิด LED
118	จบการทำงานของเงื่อนไข if
119	สร้างเงื่อนไข else if เพื่อตรวจสอบว่า led เป็น false หรือไม่
120	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
121	สั่งปิด LED

บรรทัดที่	คำอธิบาย
122	จบการทำงานของเงื่อนไข else if
123	จบการประกาศการทำงานของฟังก์ชัน void onoff(bool)

```

126 float readTemp() {
127   tempProbe.requestTemperatures();
128   return tempProbe.getTempCByIndex(0);
129 }
130
131 void displayTemp() {
132   Serial.print("temp = ");
133   Serial.print(temp);
134   Serial.println(" °C");
135 }
136
137 void displayLED() {
138   Serial.print("LED Status : ");
139   Serial.println(digitalRead(LED));
140   Serial.println();
141 }
```

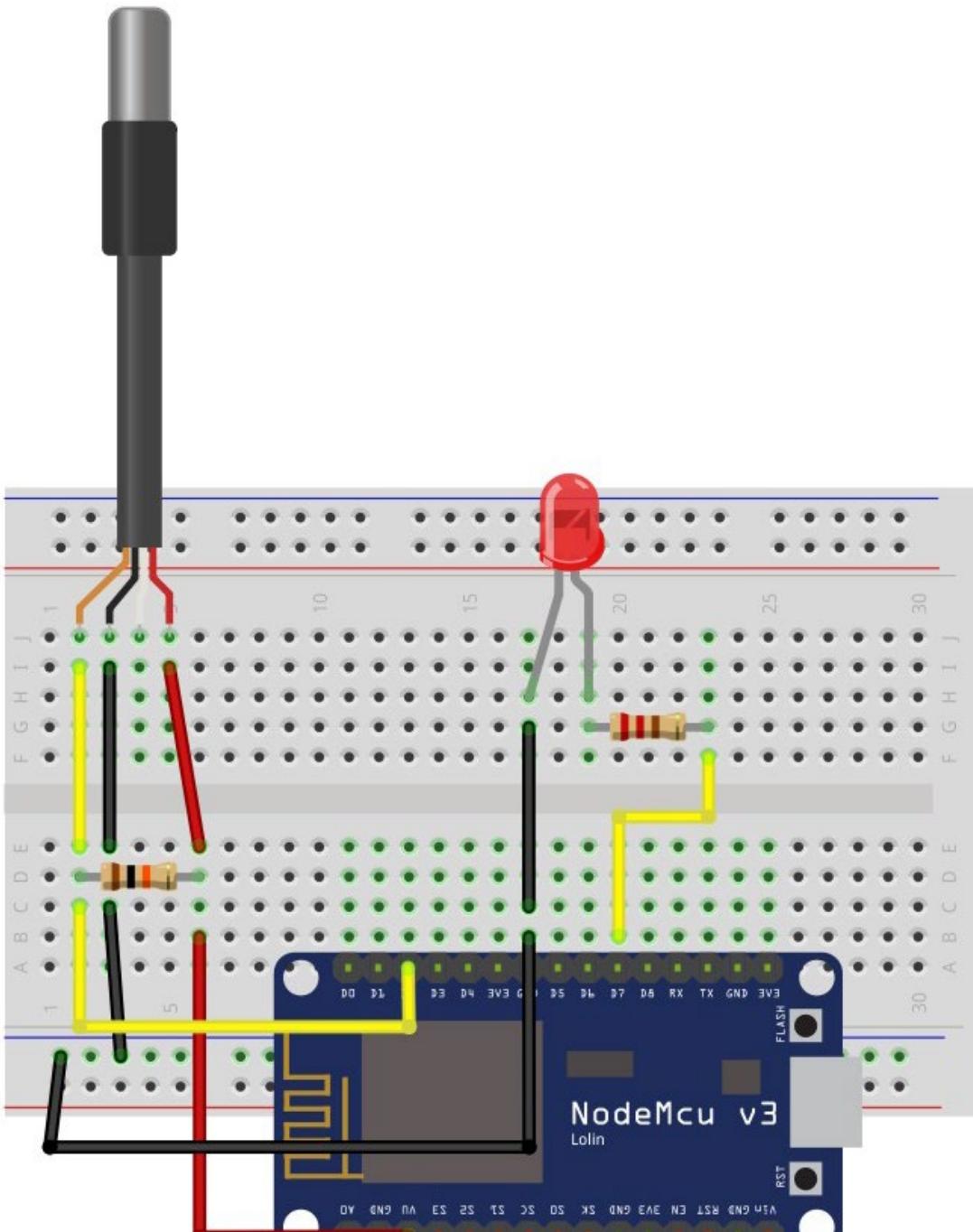
บรรทัดที่	คำอธิบาย
126	ประกาศการทำงานของฟังก์ชัน float readTemp()
127	ส่งคำสั่งเพื่อขอข้อมูลอุณหภูมิจาก Temperature Probe Sensor
128	รับ และ คืนค่าอุณหภูมิที่ได้จาก Temperature Probe Sensor
129	จบการประกาศการทำงานของฟังก์ชัน float readTemp()
131	ประกาศการทำงานของฟังก์ชัน void displayTemp()

บรรทัดที่	คำอธิบาย
132	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
133	แสดงอุณหภูมิที่อ่านได้ออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
134	แสดงข้อความออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
135	จบการประกาศการทำงานของฟังก์ชัน void displayTemp()
137	ประกาศการทำงานของฟังก์ชัน void displayLED()
138	แสดงข้อความออกทาง Serial Monitor โดยไม่ต้องขึ้นบรรทัดใหม่
139	แสดงสถานะของ LED ออกทาง Serial Monitor โดยเมื่อจบข้อความให้ขึ้นบรรทัดใหม่
140	เว้นบรรทัดใน Serial Monitor
141	จบการประกาศการทำงานของฟังก์ชัน void displayLED()

สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_18/Workshop_18.ino

การต่อวงจร

PIN	อุปกรณ์
D1	DS18B20
D7	ตัวต้านทานขนาด 220 โอห์ม



รูปจำลองการต่อวงจร

เมื่อเขียนโปรแกรม, อัปโหลดโปรแกรมลง ESP8266 และ ต่อวงจรเสร็จแล้ว ทำการสร้าง Dashboard สำหรับการแสดงผลข้อมูลโดยไปที่ Freeboard โดยตั้งชื่อว่า “ESP8266_DASHBOARD”

Create Dashboard X

* Name:	<input type="text" value="ESP8266_DASHBOARD"/>
Description: 	
<input type="button" value="Cancel"/> <input style="background-color: #007bff; color: white; border-radius: 5px; padding: 5px 10px; border: none; font-weight: bold; margin-left: 10px;" type="button" value="Create"/>	

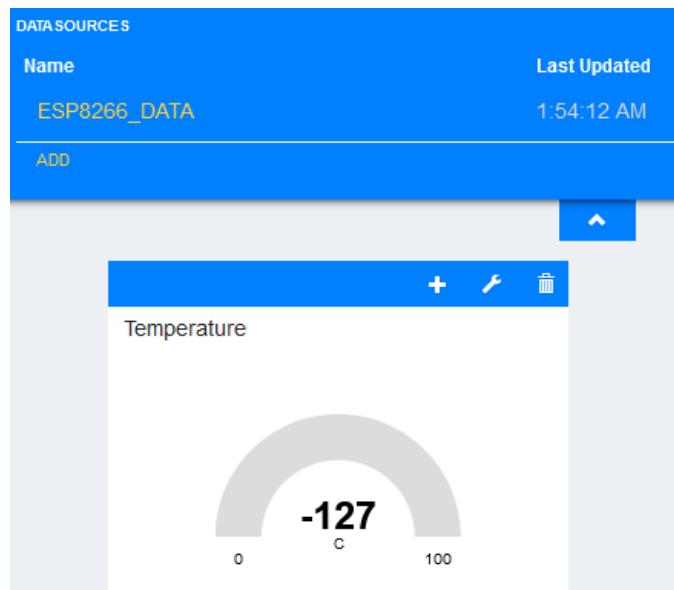
จากนั้น เข้าสู่ Freeboard และทำการเพิ่ม Data Source โดยใส่ข้อมูลที่จำเป็นให้ครบ

DATA SOURCE

NAME	<input type="text" value="ESP8266_DATA"/>
DEVICE ID	<input type="text" value="2f363130-5e0e-491e-ab44-6f971ee60fb8"/>
Client ID ของ Device ที่ต้องการอ่านข้อมูล	
DEVICE TOKEN	<input type="text" value="i36vuDQQu3frhFodxMeH3gsp9jdzRRrS"/>
Token ของ Device ที่ต้องการอ่านข้อมูล	
SUBSCRIBED TOPICS	<input type="text"/>
Topic ที่ต้องการ Subscribe	
FEED	<input checked="" type="checkbox"/> YES <input type="checkbox"/>
SINCE	<input type="text" value="6"/>
Hour	
Display data points since ... ago	

หลังจากเพิ่ม Data Source เสร็จแล้ว ทำการเพิ่ม Pane 4 ตัวเพื่อรับ Widget จะเพิ่มเข้าไปใน Dashboard โดย Widget ที่เพิ่มมี 4 ตัวได้แก่ Gauge, Toggle, Indicator Light และ FeedView โดยสามารถตั้งค่า Widget ทั้ง 4 ตัวได้ดังภาพข้างล่างต่อไปนี้

WIDGET	
TYPE	Gauge
TITLE	Temperature
VALUE	<code>datasources["ESP8266_DATA"]["shadow"]["temperature"]</code>
+ DATA SOURCE ✖ JS EDITOR	
UNITS	C
MINIMUM	0
MAXIMUM	100



WIDGET

A simple toggle widget that can perform Javascript action.

TYPE	Toggle
TOGGLE CAPTION	LED Switch
TOGGLE STATE	+ DATASOURCE Add a condition to switch a toggle state here. Otherwise it just toggle by click.
ON TEXT	ON
OFF TEXT	OFF
ONTOGGLEON ACTION	netpie["ESP8266_DATA"].publish("@msg/ledState", "on") JS code to run when a toggle is switched to ON
ONTOGGLEOFF ACTION	netpie["ESP8266_DATA"].publish("@msg/ledState", "off") JS code to run when a toggle is switched to OFF
ONCREATED ACTION	
	JS code to run after a toggle is created

WIDGET

TYPE	Indicator Light
TITLE	LED Status
DEFAULT COLOR	
	enter the color e.g. #ff0000,red or leave blank for the default color set
VALUE	datasources["ESP8266_DATA"]["shadow"]["ledState"]=="on" + DATASOURCE JS EDITOR
ON TEXT	ON + DATASOURCE JS EDITOR
OFF TEXT	OFF + DATASOURCE JS EDITOR
	SAVE

WIDGET

TYPE: FeedView

TITLE: Temperature Timeline

DATA SOURCE: datasources[{"ESP8266_DATA"}][feed]

FILTER: temperature

Type of chart: Line

X axis title:

Y axis title:

BEGIN AT 0: NO

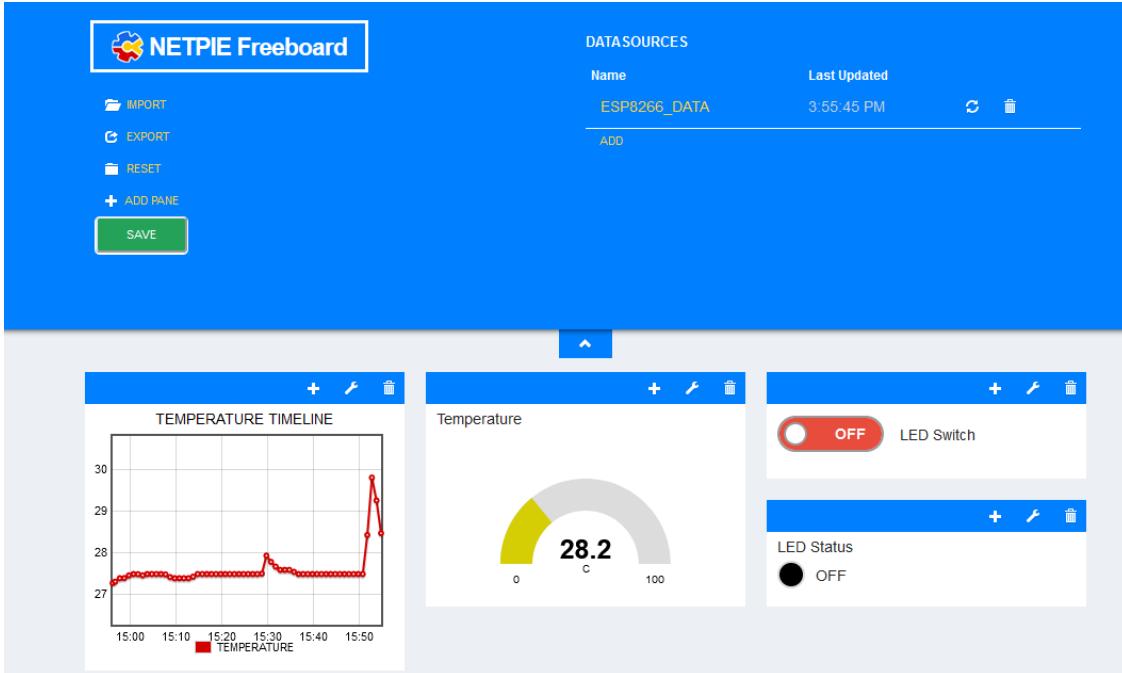
LINE COLORS: (Blank)

MAKER: YES (Orange square)

MULTIPLE AXIS: NO

HEIGHT BLOCKS: 4

SAVE CANCEL



ตัวอย่างผลลัพธ์การทำงานบน Freeboard

The screenshot shows the NETPIE Device Shadow configuration interface. On the left, a sidebar lists navigation options: Overview, Device List, Device Groups, Freeboard, Event Hooks, and Setting. The main area is titled 'ESP8266_TEST / device / ESP8266'. It contains two panels: 'Description' (empty) and 'Key' (containing Client ID, Token, Secret, Status, and Enable settings). Below these are tabs for Shadow, Schema, Trigger, and Fee, with 'Shadow' currently selected. A 'Save' and 'Cancel' button are at the bottom right. A tree view at the bottom shows a node structure: 'object {2}' with children 'ledState : off' and 'temperature : 26.25'.

ตัวอย่างผลลัพธ์การทำงานบน Device Shadow

The screenshot shows a Windows-style application window titled "COM3". The main area displays a series of text entries representing sensor data. Each entry consists of a timestamp followed by a right-pointing arrow, then a variable name, an equals sign, a value, and a unit. The data is as follows:

```
18:54:06.838 -> temp = 25.38 °C
18:54:06.838 -> LED Status : 1
18:54:06.838 ->
18:54:07.866 -> temp = 25.38 °C
18:54:07.866 -> LED Status : 1
18:54:07.866 ->
18:54:08.877 -> temp = 25.31 °C
18:54:08.877 -> LED Status : 1
18:54:08.877 ->
18:54:09.854 -> temp = 25.31 °C
18:54:09.854 -> LED Status : 1
18:54:09.854 ->
18:54:10.879 -> temp = 25.31 °C
18:54:10.879 -> LED Status : 1
18:54:10.879 ->
```

At the bottom of the window, there are several control buttons: "Autoscroll" (checked), "Show timestamp" (checked), "Newline" (dropdown menu), "115200 baud" (dropdown menu), and "Clear output".

ตัวอย่างผลลัพธ์บน Serial Monitor