

เอกสารประกอบการอบรม

เรื่องการควบคุมและแสดงผลอุปกรณ์ IoT

Internet of things

1. IoT คืออะไร

IoT หรือ Internet of thing คือ สิ่งของต่างๆ ที่ถูกเชื่อมโยงกันด้วยอินเทอร์เน็ต ซึ่งสิ่งของที่จะนำมาใช้งานก็อาจจะเป็นอะไรก็ได้ ตัวอย่างเช่น โทรศัพท์มือถือ หลอดไฟ ทีวี ปั๊มน้ำ พัดลม เครื่องปรับอากาศ เป็นต้น สิ่งต่าง ๆเหล่านี้ก็จะถูกฝังระบบควบคุมที่จะคอยตอบสนองต่อการสั่งการ และเชื่อมเข้ากับเครือข่าย และให้ติดต่อสื่อสารกันเองผ่านระบบไร้สาย ได้ด้วย อีกทั้งเรายังสามารถใช้คอมพิวเตอร์หรือสมาร์ทโฟนของเราเชื่อมต่อกับอุปกรณ์ต่าง ๆผ่านทางอินเทอร์เน็ต เพื่อเข้าควบคุม สั่งการอุปกรณ์ต่าง ๆที่เราต้องการใช้งาน หรือจะใช้ในการอ่านค่าต่าง ๆจาก sensor โดยอาศัยการส่งข้อมูลผ่านทาง cloud service เช่นฟลูอ์ร์ต่าง ๆที่เป็นตัวกลางในการรับส่งข้อมูล

1.1 ประโยชน์ของ IoT

1.1.1 ช่วยเพิ่มประสิทธิภาพในการทำงาน

IoT จะช่วยเพิ่มประสิทธิภาพในการทำงานให้แม่นยำและรวดเร็วยิ่งขึ้น เนื่องจากความสามารถในการทำงานและการส่งผ่านข้อมูลของ IoT นั้นสูงกว่าการใช้มนุษย์ทำงาน การทำงานของมนุษย์อาจจะทำให้เกิด Human Error และเกิดข้อจำกัดด้านพลังงาน, เวลา และสถานที่ได้ แต่ IoT มีความสามารถในการเก็บข้อมูล ประมวลผล ส่งผ่าน และแสดงผลได้อย่างรวดเร็วและสามารถรองรับข้อมูลได้เป็นจำนวนมากมหาศาล

1.1.2 ไร้ข้อจำกัดด้านเวลาและสถานที่

IoT สามารถทำงานได้แบบไร้พรมแดน เพราะขับเคลื่อนด้วยอินเทอร์เน็ต อย่างที่เราทราบกันดีว่า อินเทอร์เน็ตสามารถเชื่อมสิ่งที่อยู่ห่างไกล ให้ใกล้ชิดกันมากยิ่งขึ้น ยกตัวอย่างเช่นสามารถติดตามผลการดำเนินงาน และเช็คสถานะการผลิตได้ เมื่่าว่าโรงงานจะอยู่คนละจังหวัดหรือประเทศก็ตาม และ IoT ยังสามารถทำงานได้ตลอดเวลา ต่างจากมนุษย์ที่มีพลังงานจำกัด ต้องการการพักผ่อน สิ่งนี้ทำให้เห็นว่าการใช้ IoT ช่วยทำลายกำแพง ด้านเวลาและสถานที่ได้

1.1.3 ช่วยลดต้นทุนในหลาย ๆ ด้าน

เนื่องจาก IoT มีความแม่นยำและไร้ข้อจำกัดด้านเวลาและสถานที่ ทำให้ช่วยลดต้นทุนได้หลาย ๆ ด้าน อย่างเช่นต้นทุนการจ้างงาน ต้นทุนค่าเสียโอกาส หรือต้นทุนการผลิต

ตัวอย่างต้นทุนการผลิตที่ลดลง อธิบายได้ง่าย ๆ เช่น ถ้าหากว่าเราทราบข้อมูลความต้องการสินค้าอย่างละเอียด และเฉพาะเจาะจง เราจะสามารถผลิตได้ตามความต้องการของลูกค้าในทันท่วงที ตามจำนวนการสั่ง หรือที่เรียกว่า Just In Time (JIT) การผลิตแบบนี้จะช่วยลดต้นทุนในการผลิตที่เกินมาได้อย่างมีนัยสำคัญ

1.1.4 อำนวยความสะดวก ให้เวลาเหลือในการสร้างสรรค์สิ่งใหม่ๆ

สิ่งหนึ่งที่มนุษย์ทำได้ดีกว่าเทคโนโลยีคือ “ความคิดสร้างสรรค์” การให้เทคโนโลยีทำงานด้าน Routine แทนเรา จะทำให้เรามีเวลาในการทำงานสร้างสรรค์ งานนวัตกรรม ทำสิ่งที่ควรทำมากยิ่งขึ้น เพราะแท้ที่จริงแล้ว มนุษย์มีศักยภาพที่ซ่อนอยู่มากกว่าจะทำเพียงแค่งาน Routine เท่านั้น การให้เทคโนโลยีทำงานแทนและโโยก แรงงานที่ทำงาน Routine มาฝึกฝนเพื่อทำงานที่ซับซ้อนและใช้ไอเดียมากขึ้นจะทำให้เกิดความก้าวหน้ามากขึ้น

1.1.5 ยกระดับกิจการให้ Smart ในสายงานนักลงทุน

IoT เป็นอีกหนึ่งปัจจัยในการเกิดเป็น โรงงานอัจฉริยะ (Smart Factory) หรือ ธุรกิจอัจฉริยะ (Smart Business) และช่วยเสริมให้เกิดข้อดีหลาย ๆ อย่าง เช่นสร้างกำไร, ลดต้นทุน, เพิ่มรายได้และขยายกิจการ สิ่งเหล่านี้จะทำให้ผลประกอบการดีขึ้น มีหน้างบการเงินที่สวยงาม (โดยไม่ต้องตกแต่งตัวเลข) เป็นที่น่าจับตามองของนักลงทุนหรือ หุ้นส่วนทางธุรกิจต่าง ๆ ที่จะเข้ามาสนับสนุนธุรกิจ

2. การเตรียมอุปกรณ์และโปรแกรม Arduino IDE

2.1 NodeMCU ESP8266

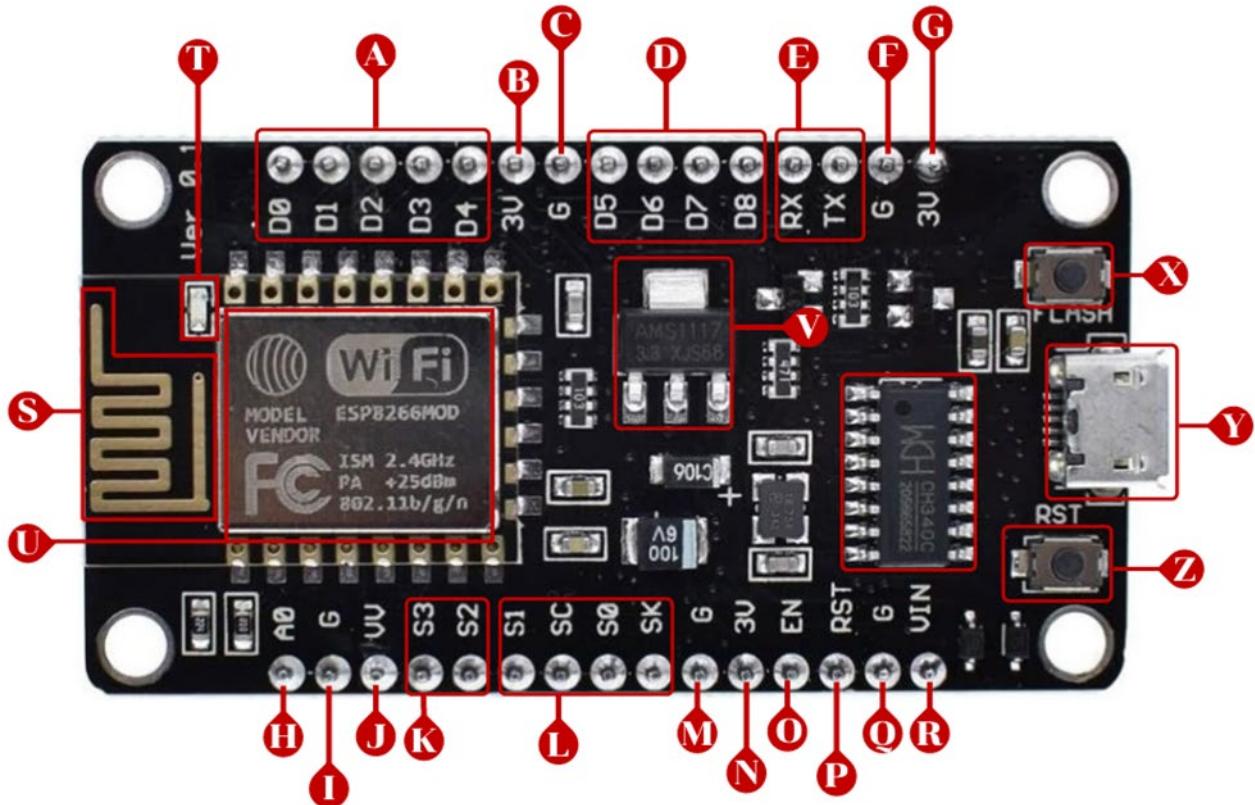
NodeMCU คือ บอร์ดคอนโทรลเลอร์ที่มีลักษณะการทำงานตามคำสั่งภาษา C คล้าย Arduino แต่มีลักษณะพิเศษกว่าตรงที่ สามารถเชื่อมต่อ กับ WiFi ได้ ในการเขียนโปรแกรมสามารถเขียนผ่าน Arduino IDE ได้

บอร์ดของ NodeMCU ประกอบไปด้วย ESP8266 พร้อมอุปกรณ์อำนวยความสะดวกต่างๆ เช่น พорт micro USB สำหรับจ่ายไฟ/อัปโหลดโปรแกรม, ชิปสำหรับอัปโหลดโปรแกรมผ่านสาย USB เป็นต้น



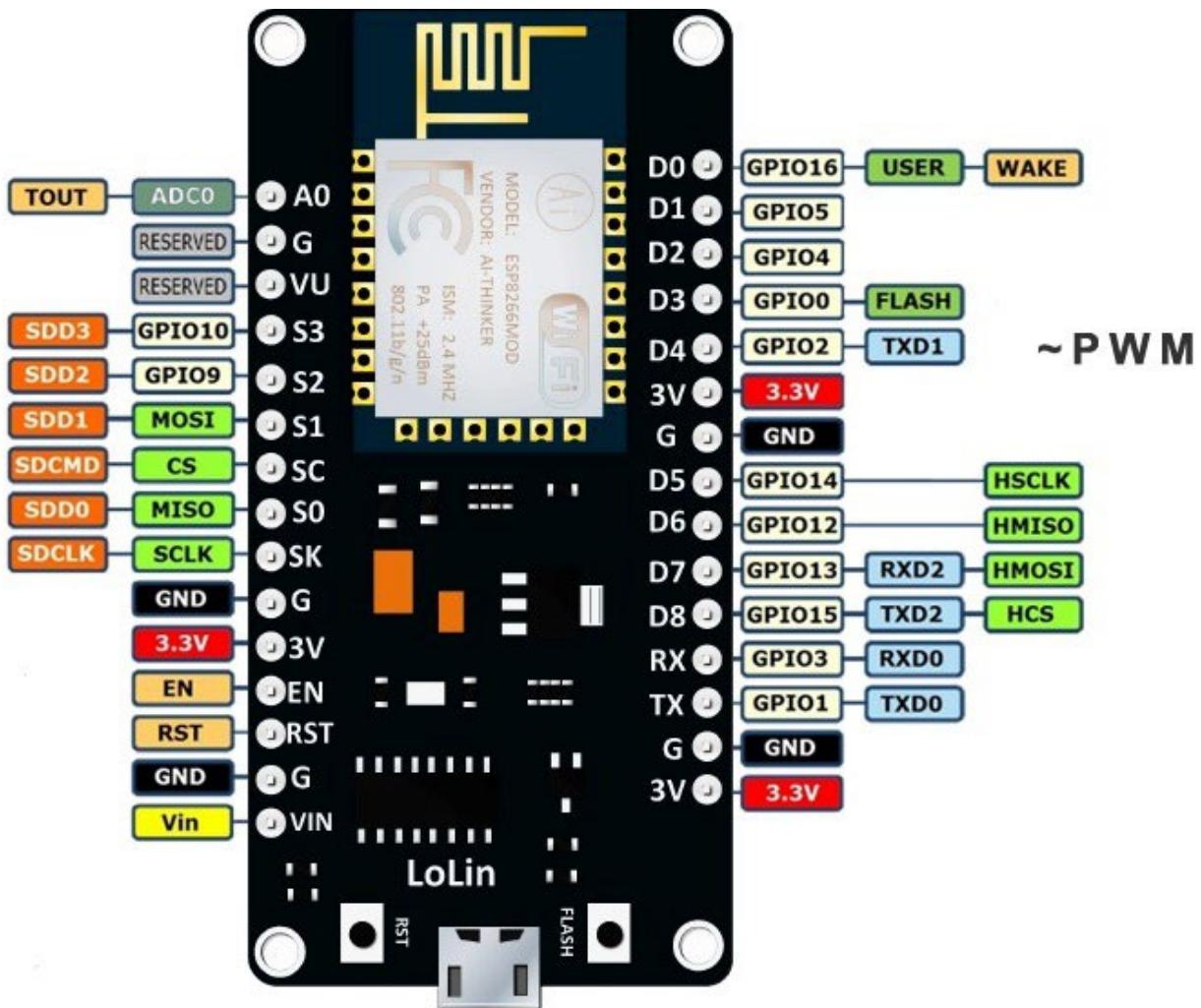
NodeMCU ESP8266 V3 ESP12E

2.2 ส่วนประกอบ NodeMCU ESP8266



- A. D0 - D4 หรือ GPIO เป็นขาดิจิตอลที่สามารถกำหนดให้ขาที่ต้องการเป็น Input หรือ Output ก็ได้
- B. 3V เป็นขาที่มีแรงดันไฟ 3.3 V
- C. G เป็นขา Ground
- D. D5 - D8 เป็นทั้งขา GPIO และขาที่ใช้ติดต่อสื่อสารกับโมดูลอื่นผ่านอินเตอร์เฟส HSPI
- E. RX/TX เป็นขา GPIO และขาที่ใช้รับส่งข้อมูลแบบ Serial กับอุปกรณ์อื่น
- F. G เป็นขา Ground
- G. 3V เป็นขาที่มีแรงดันไฟ 3.3 V
- H. A0 หรือ ADC เป็นขา Input ที่ใช้อ่านค่าจาก sensor ที่ส่งค่ามาเป็น analog มาแปลงเป็น digital
- I. G เป็นขา Ground
- J. VU หรือ V USB เป็นขาที่แรงดันไฟ 5 V
- K. S3, S2 เป็นทั้งขา GPIO และขาที่ใช้ติดต่อสื่อสารกับอุปกรณ์ SD card โดยตรงผ่านอินเตอร์เฟส SDIO (Secure Digital Input/Output)

- L. S1, SC, SO และ SK เป็นทั้งขา GPIO และขาที่ใช้ในการติดต่อสื่อสารกับอุปกรณ์ SD card โดยตรงผ่านอินเตอร์เฟส SDIO และขาที่ใช้ติดต่อสื่อสารกับโมดูลอื่นผ่านอินเตอร์เฟส SPI ที่เร็วกว่า I2C
- M. G เป็นขา Ground
- N. 3V เป็นขาที่มีแรงดันไฟ 3.3 V
- O. EN เป็นขาที่ใช้ควบคุมให้โมดูลทำงาน เมื่อมีการจ่ายไฟหรือกำหนดสถานะให้เป็น Active High
- P. RST เป็นขาที่ใช้รีเซ็ตโมดูล เมื่อต้องกับ Ground หรือ กำหนดสถานะให้เป็น Active Low
- Q. G เป็นขา Ground
- R. VIN เป็นขาที่ใช้รับไฟเข้าจากแหล่งจ่ายภายนอกเพื่อจ่ายไฟให้กับบอร์ด และอุปกรณ์เชื่อมต่ออื่น ๆ โดยตรง ซึ่งไฟที่จ่ายให้ครमีขนาดไม่เกิน 5 V
- S. 2.4GHz Antenna ตำแหน่งของสายอากาศยานความถี่ 2.4 GHz
- T. หลอดไฟ LED ใช้สถานะการอัพโหลดโปรแกรม (สำหรับบอร์ด V3 ที่ใช้ชิป USB TTL เป็น CH340 หลอดไฟนี้ยังสามารถใช้แสดงสถานะการทำงานของโปรแกรมหรือการส่งข้อมูล (TX) ที่ขา D4/GPIO2 ได้ด้วย เนื่องจากที่ขา D4/GPIO2 มีการเชื่อมต่อกับหลอดไฟ LED นี้เอาไว้ ส่วนบอร์ด V2 ที่ใช้ชิป CP2102 จะมีหลอดไฟ LED ที่ใช้แสดงสถานะการทำงานของโปรแกรมหรือการส่งข้อมูล (TX) ติดตั้งแยกมาให้ต่างหากบนบอร์ด ซึ่งจะเชื่อมต่ออยู่กับขา D0/GPIO16)
- U. ชิปหลัก ESP8266 รุ่น ESP-12E เป็นชิปที่มีทั้งหน่วยประมวลผล 32-bit ความถี่ 80 - 160 MHz, หน่วยความจำ (128 KB internal RAM + 4MB external Flash) และตัวรับส่งสัญญาณ Wi-Fi มาตรฐาน 802.11 b/g/n อยู่ในตัว ทำงานที่แรงดันไฟ 3.0-3.6V กระแสไฟโดยเฉลี่ยที่ 80mA
- V. ชิป 3.3V LDO Voltage Regulator ใช้ควบคุมและรักษากระแสไฟคงที่ที่ 3.3V
- W. ชิป USB to TTL Converter เป็นตัวกลางในการสื่อสารระหว่างอุปกรณ์ที่ใช้ port USB เพื่อส่งผ่านข้อมูลไปยังบอร์ดไมโครคอนโทรลเลอร์ เช่น การอัพโหลดโปรแกรม โดยจะทำหน้าที่แปลงข้อมูลจาก port USB ไปเป็นข้อมูลแบบอนุกรม (serial) ก่อนจะส่งไปยัง MCU ซึ่งถ้าเป็นบอร์ด V3 จะใช้เป็น CH340 USB to serial Controller ส่วน V2 จะใช้เป็น CP2102 USB to UART Bridge Controller
- X. บูม Flash มีไว้เพื่อการอัพโหลดโปรแกรม
- Y. Micro USB Connector ใช้เสียบสาย USB เพื่อจ่ายไฟ 5V ให้กับบอร์ด และยังโหลดข้อมูล
- Z. บูม RST มีไว้ใช้ในการ reset บอร์ด เพื่อเริ่มการทำงานใหม่



NodeMCU V3 ESP8266 Pinout

ที่มา <https://i1.wp.com/www.teachmemicro.com>

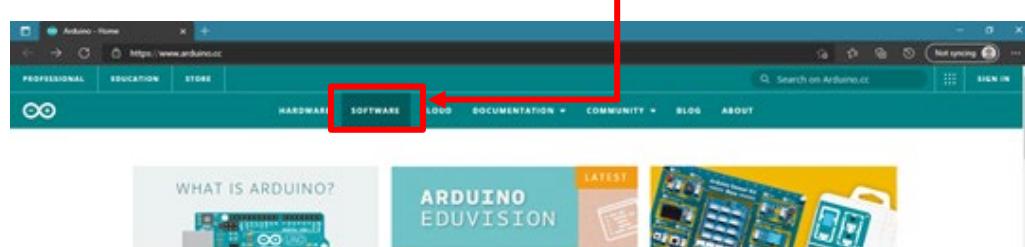
3. Arduino IDE

Arduino IDE (Arduino Integrated Development Environment) เป็นชุดซอฟต์แวร์ที่ใช้สำหรับการพัฒนา Arduino โดยเฉพาะ ซึ่งเราสามารถใช้ Arduino IDE เพื่อเขียนโค้ดคำสั่ง, คอมไฟล์ (แปลงภาษาคอมพิวเตอร์เป็นภาษาเครื่อง) และใช้เพื่ออัปโหลดซอฟต์แวร์ไปยัง Arduino รวมถึงสามารถใช้ Arduino IDE สำหรับตรวจสอบผลลัพธ์การทำงานและอื่นๆ

3.1 การติดตั้ง Arduino IDE

เริ่มต้นจากการดาวน์โหลด Arduino IDE เวอร์ชันล่าสุดได้ที่ <https://www.arduino.cc/> จากนั้นลงมือติดตั้งตามตัวอย่างต่อไปนี้

1. ไปที่หน้าเว็บ www.arduino.cc แล้วเลือกที่ software



Legacy IDE (1.8.X)

Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Arduino IDE 1.x documentation](#) for installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this gpg key](#).

DOWNLOAD OPTIONS

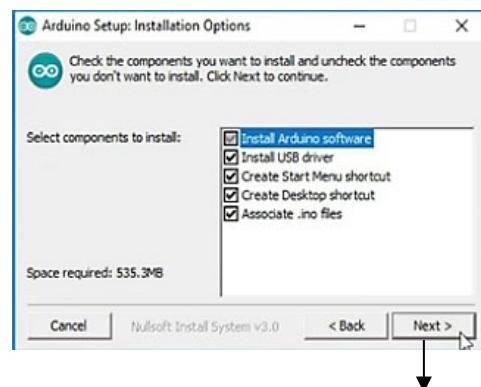
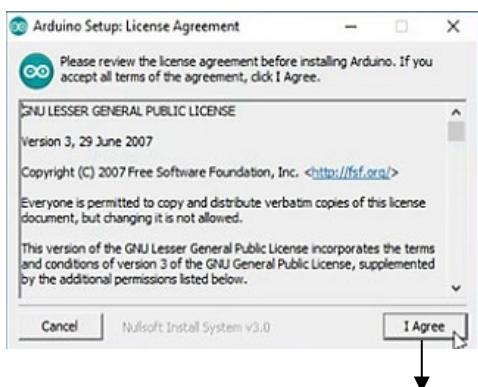
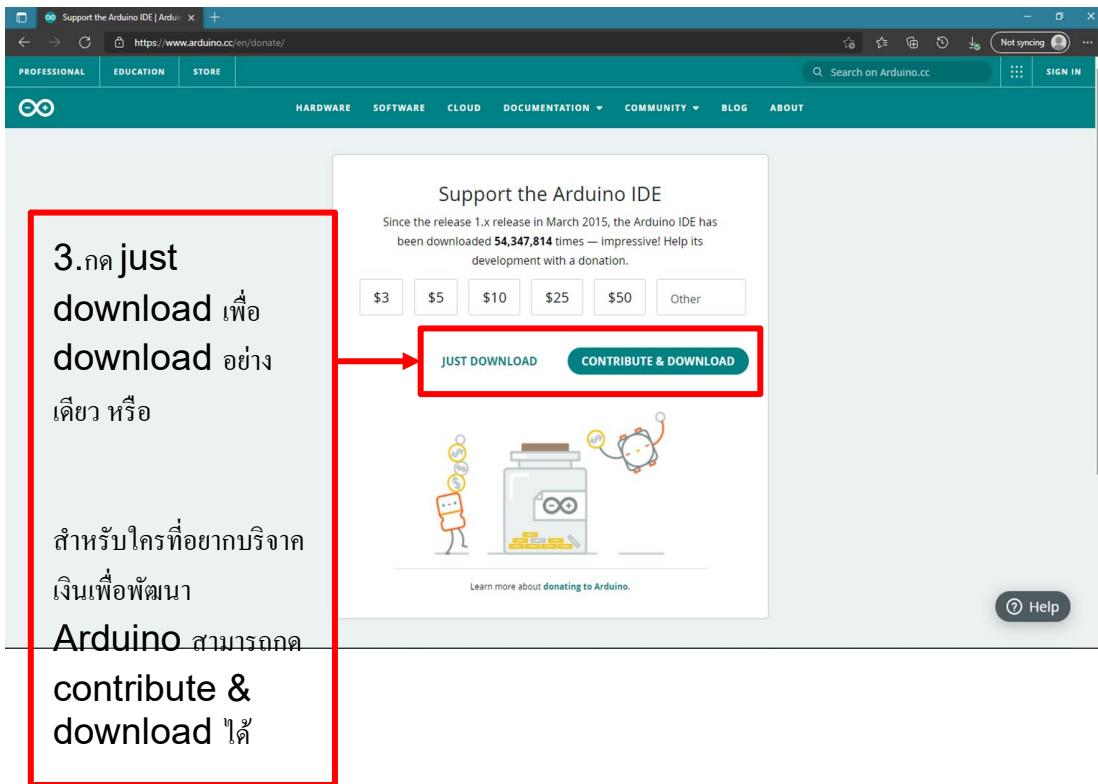
- Windows** Win 7 and newer
- Windows app** Win 8.1 or 10 [Get](#)
- Linux** 32 bits
- Linux** 64 bits
- Linux** ARM 32 bits
- Linux** ARM 64 bits
- Mac OS X** 10.10 or newer

[Release Notes](#)

[Checksums \(sha512\)](#)

2. จากนั้นเลื่อนลงมา และให้เลือก OS

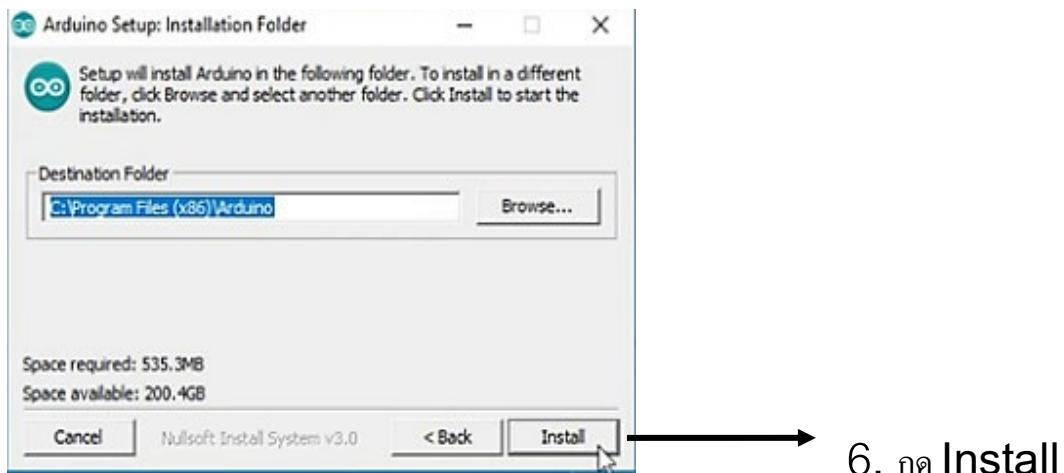
ที่ใช้อยู่ กรณี Windows ให้เลือกเวอร์ชัน 1.8.19 ซึ่งเป็นเวอร์ชันที่มีเสถียรภาพ



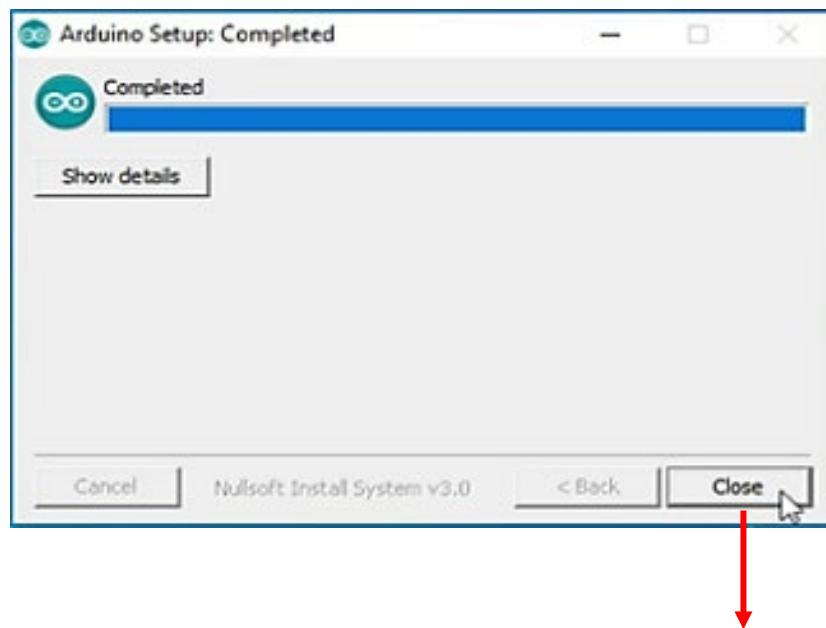
4. กด I Agree เพื่อดำเนินการต่อ

5. จะมีหน้าต่างให้เลือกตัวเลือก ให้กด

Next ต่อ โดยไม่ต้องแก้ไขตัวเลือกใดๆ



7. ในระหว่างการติดตั้ง จะมีการให้ติดตั้ง driver ต่างๆ ให้ทำการกดติดตั้ง driver ที่ไหนด



8. หลังจากติดตั้งเสร็จแล้ว ให้ทำการกด Close เป็นขั้นตอนสุดท้าย

3.2 กรณีไม่ได้ติดตั้งโปรแกรม Arduino IDE จากตัว Installer (นามสกุล .exe) แต่ติดตั้งจากไฟล์ zip แทน
จำเป็นต้องติดตั้ง driver CH340 USB to TTL Driver และ ตั้งค่าโปรแกรม สำหรับอัพโหลดโปรแกรม
สามารถทำได้ดังนี้

ให้ตรวจสอบว่า NodeMCU ESP8266 ที่มีอยู่ใช้ chip อะไร ให้ดูจากรูปร่าง หรือ ดูตัวหนังสือบน chip หาก
ไม่เห็นให้ลองพลิกให้ NodeMCU ESP8266 ดู อาจจะมีรุ่นของ chip เขียนอยู่ จากนั้นจึงเลือกติดตั้ง driver
ตามประเภท chip ที่ใช้



การติดตั้ง CH340 USB to TTL Driver

1. เข้าไปเว็บไซต์ และ download Windows CH340 Driver

https://sparks.gogo.co.nz/assets/_site_/downloads/CH34x_Install_Windows_v3_4.zip

← → ⌂ sparks.gogo.co.nz/ch340.html?srsltid=AfmBOoro28ZjpKAUq7O7iFE_A99hdjtq6jbdepVdj-zuOijqt2-ddwFz

Gogo:Tronics
Hobby Electronic Parts

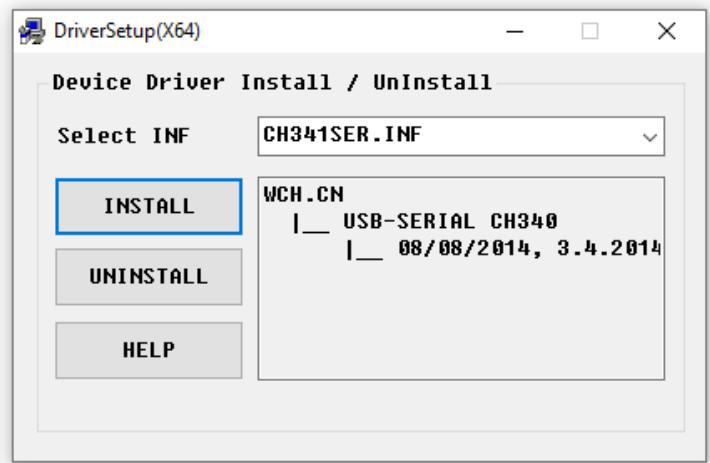
Electronic Components Online Shop Tools, Docs, Downloads Cu

Windows

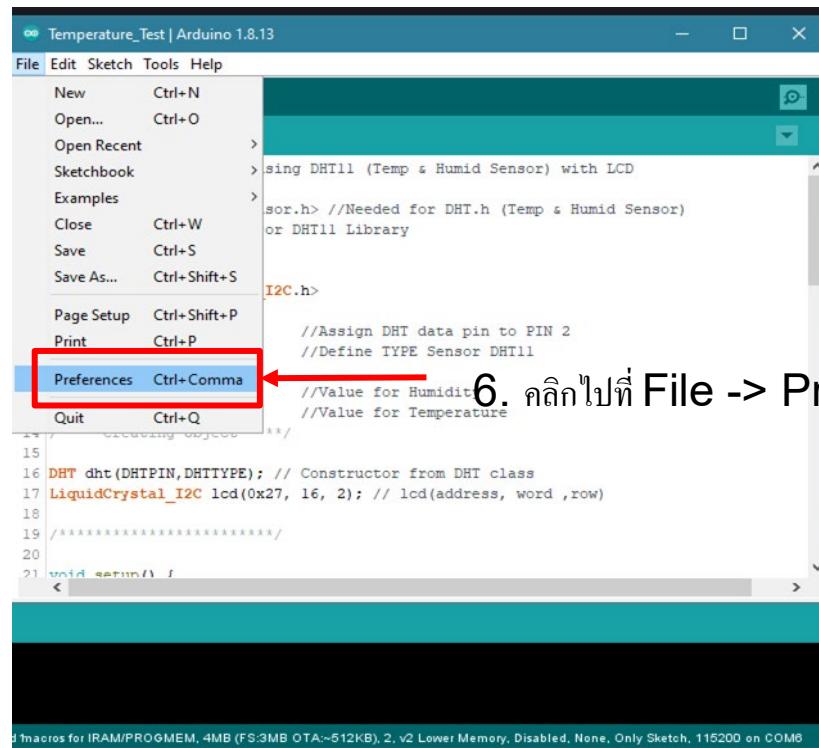
(Manufacturer's Chinese Info Link)

- Download the [Windows CH340 Driver](#)
- Unzip the file
- Run the installer which you unzipped
- In the Arduino IDE when the CH340 is connected you will see a COM Port in the Tools > Serial F depending on your system.

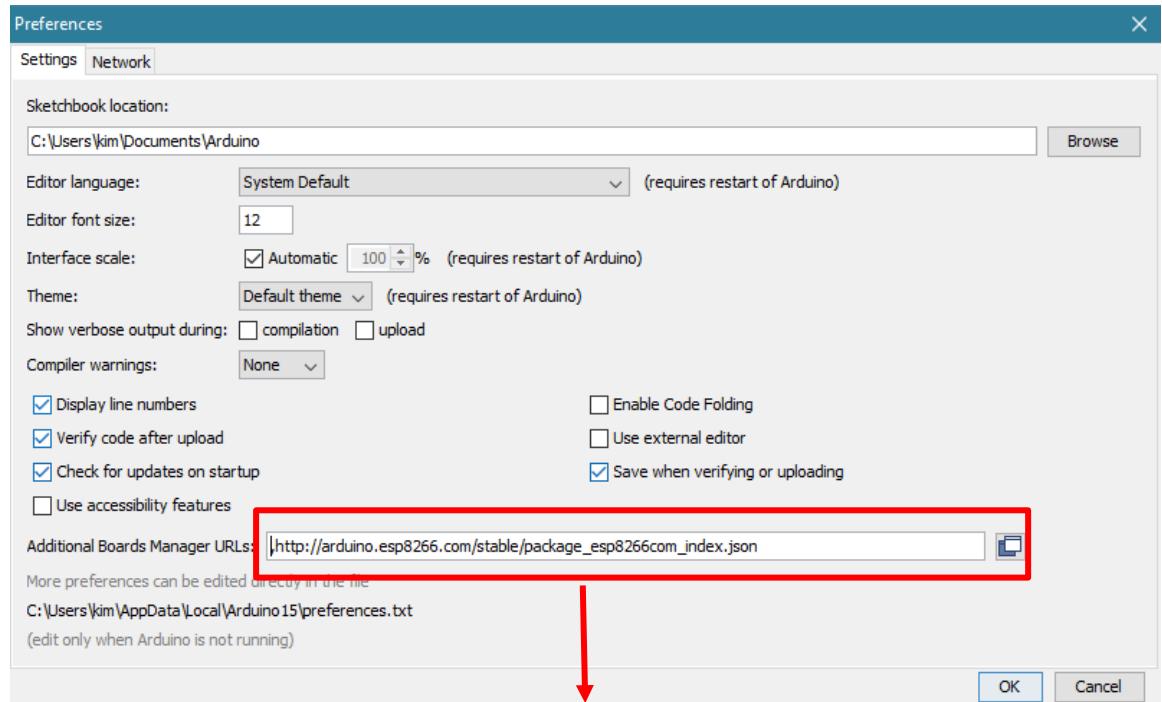
2. ให้ทำการแตกไฟล์และกด INSTALL



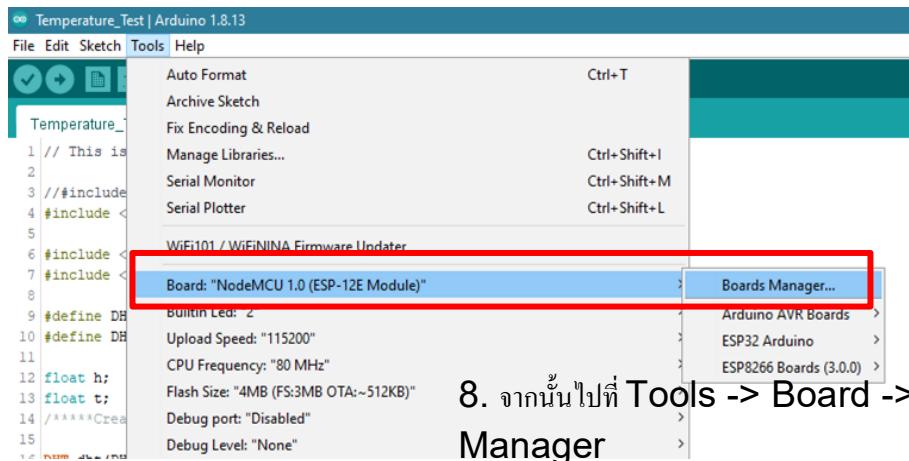
3. หลังจากติดตั้ง Driver เสร็จแล้ว ให้เปิดโปรแกรม Arduino IDE ขึ้นมา



6. คลิกไปที่ File -> Preferences



4. เพิ่ม http://arduino.esp8266.com/stable/package_esp8266com_index.json ลงในช่อง Additional Boards Manager URLs ดังภาพ หากขึ้นข้อความ Error downloading http://arduino.esp8266.com/stable/package_esp8266com_index.json ให้เพิ่ม url นี้แทน https://github.com/esp8266/Arduino/releases/download/2.3.0/package_esp8266com_index.json



8. จากนั้นไปที่ Tools -> Board -> Board Manager

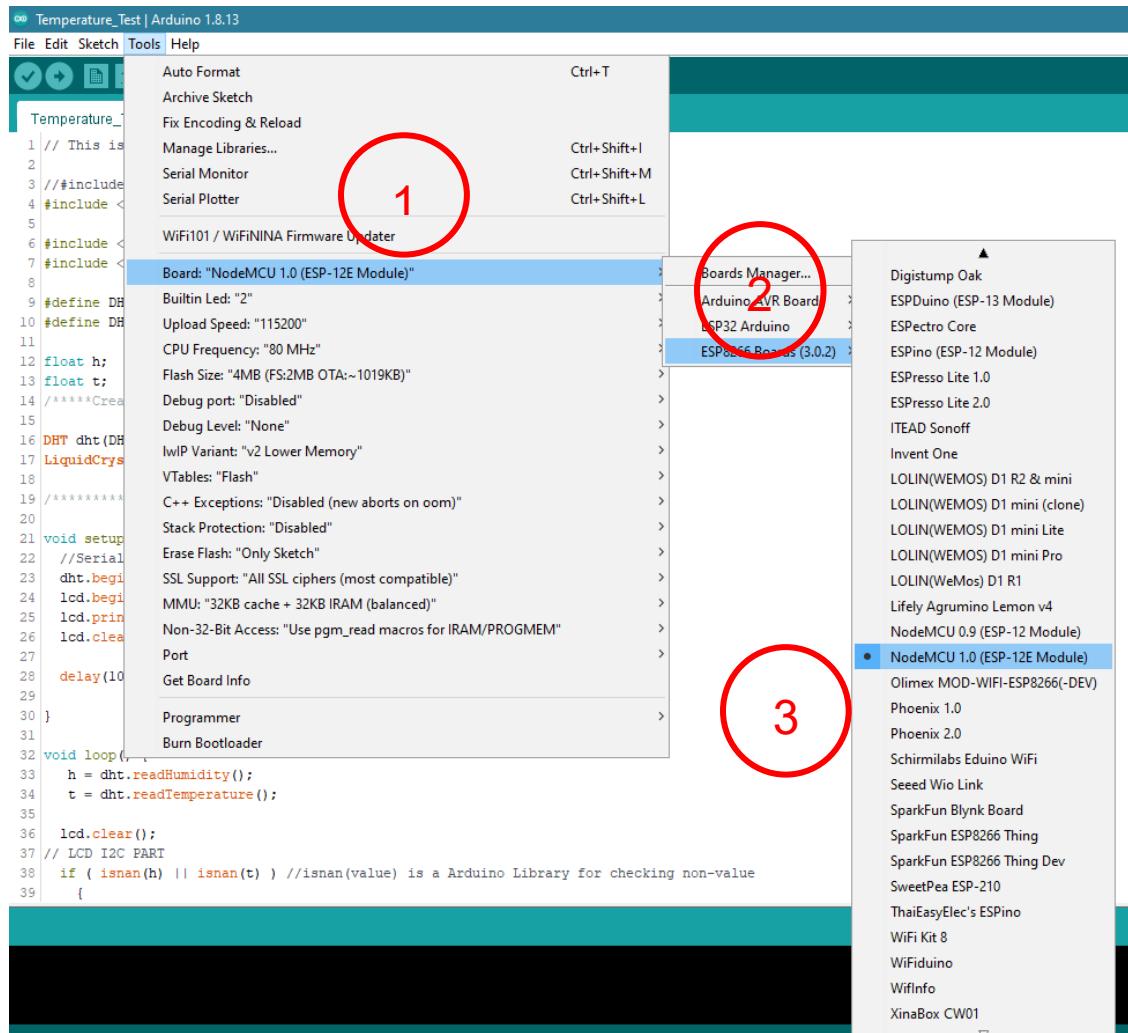
9. พิมพ์ว่า esp8266



10. เลือก version แล้วกด install

3.3 การเลือกบอร์ดและ Port

ให้ไปที่ Tools -> Board -> ESP8266 Boards -> NodeMCU1.0ESP-12E Module



4. การเขียนโปรแกรม

4.1 ชนิดข้อมูลและตัวแปร

วิธีควบคุม microcontroller ต่างๆ ให้ทำงานตามที่ต้องการจะต้องมีชุดคำสั่งเพื่อควบคุมการรับ และส่งข้อมูลระหว่างตัวบอร์ดกับอุปกรณ์ต่อพ่วงอื่นๆ ซึ่งชนิดข้อมูลและตัวแปรเป็นสิ่งแรกที่จำเป็นต้องทราบในการเขียนโปรแกรม

ในการเขียนโปรแกรม เราจะใช้ตัวแปร (Variable) ซึ่งใช้จัดเก็บข้อมูลเพื่อนำไปใช้ประมวลผลต่อ โดยตัวแปรเหล่านี้จะไปจ่องพื้นที่บางส่วนของหน่วยความจำ (Memory) ไว้จัดเก็บข้อมูลชั่วคราว ซึ่งพื้นที่หน่วยความจำของข้อมูลแต่ละชนิดจะมีขนาดที่แตกต่างกัน

ชนิด	ขนาด	ช่วงของค่า
char	8 bits	-128 to 127
unsigned char	8 bits	0 to 255
int	16 bits	-32768 to 32767
unsigned int	16 bits	0 to 65535
long	32 bits	-2147483648 to -2147483649
unsigned long	32 bits	0 to 4294967296
float	32 bits	3.4E-38 to 3.4E38 หรือ ทศนิยม 6 ตำแหน่ง
double	32 bits	1.7E-308 to 1.7E308 หรือ ทศนิยม 12 ตำแหน่ง
bool	1 bit	-

4.2 การประกาศตัวแปรและกำหนดชนิดข้อมูล

การประกาศตัวแปรเป็นการแจ้งให้คอมไพล์รู้ว่าต้องการจดพื้นที่หน่วยความจำส่วนหนึ่งไว้สำหรับใช้เก็บข้อมูลโดยกำหนดขนาดพื้นที่และวิธีดำเนินการตามชนิดข้อมูล ส่วนซึ่งตัวแปรจะเป็น

ตัวแทนตำแหน่งของหน่วยความจำที่จะองไว้

เริ่มต้นการประกาศตัวแปร

ในการประกาศตัวแปรใน C++ จะเริ่มจากการกำหนดชนิดตัวแปร เว้นวรรคแล้วตามด้วยชื่อ ตัวแปรและปิดท้ายด้วย ;

Syntax

dataType variableName;

dataType	ชนิดตัวแปร
variableName	ชื่อตัวแปรที่ใช้เก็บข้อมูล ซึ่งสามารถประกาศได้หลายตัว แต่ต้องคั่นด้วยเครื่องหมาย , (คอมม่า)

ตัวอย่าง

```
int i, Num, _int; //ประกาศตัวแปรเก็บเลขจำนวนเต็ม
char _char, ch; //ประกาศตัวแปรเก็บอักขระ
float price; //ประกาศตัวแปรเก็บตัวเลขทศนิยม
bool check; //ประกาศตัวแปรเก็บข้อมูลตรรกะ
```

4.3 การประกาศตัวแปรร่วมกับกำหนดค่าเริ่มต้น

เมื่อประกาศตัวแปรเสร็จ เราสามารถกำหนดค่าเริ่มต้นให้กับตัวแปรนั้นได้เลย โดยเพิ่มเครื่องหมาย = ตามด้วยค่าที่ต้องการ

Syntax

```
dataType variableName = value;
```

dataType	ชนิดตัวแปร
variableName	ชื่อตัวแปรที่ใช้เก็บข้อมูล
value	ข้อมูลที่กำหนดให้เป็นค่าเริ่มต้น

ตัวอย่าง

```
int i = 0, Num = 10, _int = 20;
char _char = 'A' , ch = 'a';
float price = 199.00;
bool check = true;
```

4.4 ขอบเขตของตัวแปร

ตัวแปรที่ถูกประกาศขึ้นจะมีขอบเขตการใช้งาน หากเรียกใช้ตัวแปรซึ่งอยู่นอกขอบเขตของ ตัวแปรก็ไม่สามารถใช้งานได้ ซึ่งขอบเขตตัวแปรแบ่งได้เป็น 2 ประเภท คือ ตัวแปรโกลบอล และ ตัวแปรโลคอล

1. ตัวแปรโกลบอล (Global Variable)

เป็นตัวแปรที่ประกาศอยู่นอกฟังก์ชัน ซึ่งจะมีขอบเขตครอบคลุมทั่วทั้งไฟล์โปรแกรม

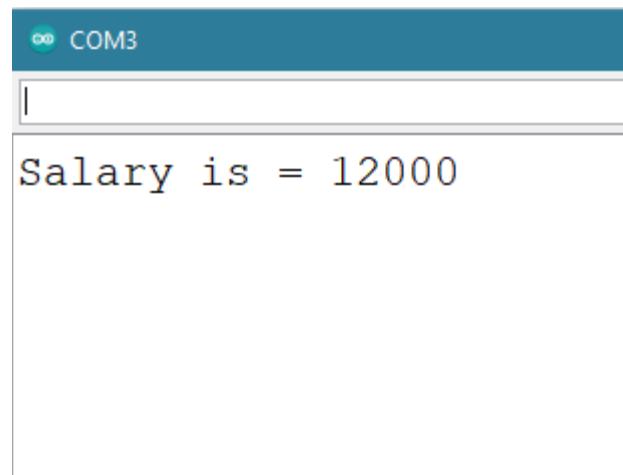
2. ตัวแปรโลคอล (Local Variable)

เป็นตัวแปรที่จะประกาศอยู่ภายในบล็อกของกลุ่มคำสั่ง (คำสั่งที่อยู่ระหว่างเครื่องหมาย {}) หรือฟังก์ชัน มีขอบเขตเข้าถึงได้เฉพาะคำสั่งที่อยู่ในพื้นที่คำสั่งหรือฟังก์ชัน เดียว กันเท่านั้น

ตัวอย่างการประมวลผลแบบทวิภาคี

```
int salary = 12000; //ประมวลผลตัวแปรคง住ล
void setup(){
Serial.begin (115200);
}
void loop(){
Serial.print("Salary is = ");
Serial.println(salary);
delay(10000);
}
```

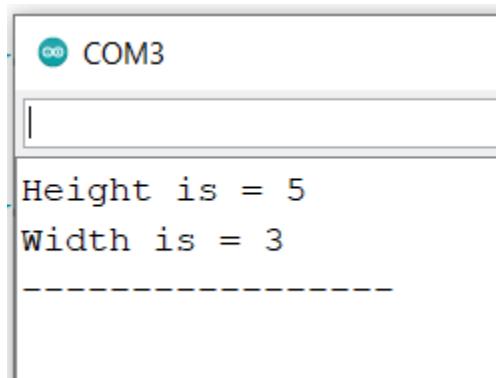
OUTPUT :



ตัวอย่างการประกาศตัวแปรแบบโลคอล

```
void setup() {
    Serial.begin (115200);
}
void loop() {
    int height, width; //ประกาศตัวแปรโลคอล
    height = 5;
    width = 3;
    Serial.print("Height is = ");
    Serial.println(height);
    Serial.print("Width is = ");
    Serial.println(width);
    Serial.println("-----");
    delay(10000);
}
```

OUTPUT :



4.5 การประกาศค่าคงที่ด้วย define

#define คือ preprocessor ที่ใช้ในการกำหนดชื่อให้กับค่าคงที่ เพื่อให้คอมไพล์รู้ว่าเมื่อเจอ ชื่อนั้นในตำแหน่งใดๆ ให้แทนที่ด้วยค่าคงที่นั้นแล้วค่อยประมวลผลโปรแกรม

Syntax

```
#define identifier value
```

identifier	ชื่อที่ต้องนิยามเป็นค่าคงที่
value	ค่าคงที่ที่ต้องการกำหนด

ตัวอย่าง

```
#define pi 3.14
#define DELAY1 1000
#define check true
```

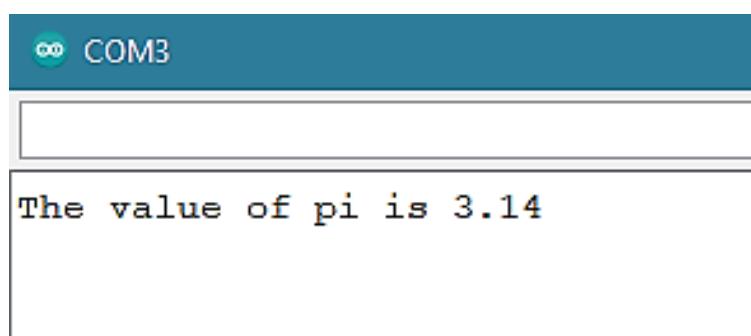
ตัวอย่าง

```
#define pi 3.14
void setup() {
    Serial.begin(9600);

    Serial.print("The value of pi is ");
    Serial.println(pi);
}

void loop() {
```

Output



```
The value of pi is 3.14
```

4.6 ตัวดำเนินการ (Operator)

ตัวดำเนินการ (Operator) เป็นสัญลักษณ์บอกให้คอมไพล์เวอร์ดำเนินการอย่างใดอย่างหนึ่งกับ ตัวแปรและค่าคงที่ต่างๆ ซึ่งใน C++ มีตัวดำเนินการหลายประเภทด้วยกัน เช่น ตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการเปรียบเทียบ ตัวดำเนินการเชิงตรรกية และอื่นๆ

ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการ	ความหมาย	ตัวอย่างการใช้
+	บวก	A+B
-	ลบ	A-B
*	คูณ	A*B
/	หาร	A/B
%	หารเอาเศษ	A%B
++	เพิ่มค่าในตัวแปรขึ้น 1	++A , A++
--	เพิ่มค่าในตัวแปรลง 1	--A , A--

ตัวอย่าง

```

int a, b, sum, sub;
void setup() {
    Serial.begin(9600);
    a = 10;
    b = 3;
    Serial.print("a = ");
    Serial.println(a);
    Serial.print("b = ");
    Serial.println(b);
    sum = a+b;
    Serial.print("Sum = ");
    Serial.println(sum);
    Serial.print("Subtract = ");
    Serial.println(sub);
}

void loop() {
}

```

Output

```
a = 10
b = 3
Sum = 13
Subtract = 0
```

4.7 การใช้ตัวดำเนินการเพิ่มค่า/ลดค่า (Prefix / Postfix)

การใช้ตัวดำเนินการเพิ่มค่า ++ ซึ่งเพิ่มค่าของตัวแปร 1 ค่า กับตัวดำเนินการลดค่า -- ซึ่งลดค่าของตัวแปร 1 ค่า สามารถใช้ตัวดำเนินการนี้ได้ 2 วิธี คือ

Prefix : วางตัวดำเนินไว้หน้าตัวแปร เช่น `++A` หรือ `--A` ซึ่งจะทำให้ตัวแปรถูกเพิ่มหรือลดก่อนถูกนำไปใช้งาน

Postfix : วางตัวดำเนินไว้หลังตัวแปร เช่น `A++` หรือ `A--` ซึ่งจะทำให้ตัวแปรถูกเพิ่มหรือลดหลังถูกนำไปใช้งาน

ตัวอย่าง

```

1 void setup() {
2   Serial.begin(9600);
3   int a = 12, b = 7;
4   Serial.print("a++ = ");
5   Serial.println(a++);
6   Serial.print("a = ");
7   Serial.println(a);
8   Serial.print("++b = ");
9   Serial.println(++b);
10 }
11
12 void loop() {
13
14 }
```

Output

```
a++ = 12
a = 13
++b = 8
```

บรรทัดที่ 5 ตัวแปร a ค่าปัจจุบันคือ 12 เมื่อถูกเรียกใช้จะทำการแสดงผลก่อน แล้วจึงค่อยเพิ่มค่าขึ้น 1

บรรทัดที่ 7 เป็นค่า a ซึ่งถูกเพิ่มค่าหลังจากถูกเรียกใช้งาน โดยปัจจุบันมีค่าเท่ากับ 13

บรรทัดที่ 9 ตัวแปร b ค่าปัจจุบันคือ 7 โดยก่อนที่จะถูกเรียกใช้ จะถูกค่าเพิ่มขึ้นก่อน 1 ค่า หลังจากนั้น
จึงถูกนำขึ้นไปแสดงผลลัพธ์

4.8 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการเปรียบเทียบ (Relation Operators) เป็นการนำข้อมูลมาเปรียบเทียบกันโดย ผลลัพธ์ที่ได้
จะออกมาเป็นค่า boolean คือ true/false โดยข้อมูลที่นำมาเปรียบเทียบต้องเป็นชนิดเดียวกัน

ตัวดำเนินการ	ความหมาย	ตัวอย่างและความหมาย
<code>==</code>	เท่ากัน	$A == B$ เป็นจริงเมื่อตัวแปร A เท่ากับตัวแปร B
<code>!=</code>	ไม่เท่ากัน	$A != B$ เป็นจริงเมื่อตัวแปร A ไม่เท่ากับตัวแปร B
<code>></code>	มากกว่า	$A > B$ เป็นจริงเมื่อค่าในตัวแปร A มากกว่าค่าในตัวแปร B
<code><</code>	น้อยกว่า	$A < B$ เป็นจริงเมื่อค่าในตัวแปร A น้อยกว่าค่าในตัวแปร B
<code>>=</code>	มากกว่าเท่ากับ	$A >= B$ เป็นจริงเมื่อค่าในตัวแปร A มากกว่าเท่ากับค่าในตัวแปร B
<code><=</code>	น้อยกว่าเท่ากับ	$A <= B$ เป็นจริงเมื่อค่าในตัวแปร A น้อยกว่าเท่ากับค่าในตัวแปร B

ตัวอย่าง

```
void setup() {  
    Serial.begin(9600);  
    int a = 12, b = 7;  
    Serial.print("a is equal to b = ");  
    Serial.println(a==b);  
    Serial.print("a is not equal to b = ");  
    Serial.println(a!=b);  
    Serial.print("a is less than b = ");  
    Serial.println(a<b);  
}
```

Output



```
COM3  
|  
a is equal to b = 0  
a is not equal to b = 1  
a is less than b = 0
```

จากตัวอย่างເອົາຕົ້ນຈະໄດ້ວ່າການແສດງຜລລັບຮູບຂອງຕັ້ງດຳເນີນການປະຕິບັດເທື່ອມີຢູ່ 2 ດ້ວຍກຳນົດ 0 ທີ່ມີຄ່າເປັນ false ແລະ 1 ທີ່ມີຄ່າເປັນ true

4.9 ตัวดำเนินการทางตรรกะ

เป็นตัวดำเนินการที่ใช้กับข้อมูลชนิด Boolean ประกอบด้วย AND, OR และ NOT ผลลัพธ์ที่ได้จะเป็นแบบ Boolean คือ จริง (true) และเท็จ (false)

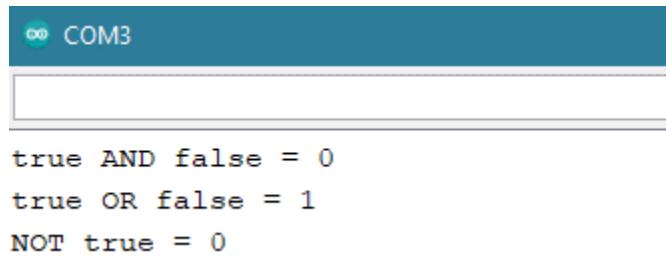
ตัวดำเนินการ	ความหมาย	ตัวอย่างและความหมาย
&&	AND	A && B เป็นจริงเมื่อตัวแปร A และ B เป็นจริงทั้งคู่
	OR	A B เป็นจริงเมื่อตัวแปร A หรือ B เป็นจริง
!	NOT	!A ให้ผลตรงข้ามกับ A ถ้า A เป็นจริงผลลัพธ์เป็นเท็จ

ตัวอย่าง

```
void setup() {
    Serial.begin(9600);
    bool a = true, b = false;
    Serial.print("true AND false = ");
    Serial.println(a && b);
    Serial.print("true OR false = ");
    Serial.println(a || b);
    Serial.print("NOT true = ");
    Serial.println(!a);
}

void loop() {
```

Output



```
COM3
true AND false = 0
true OR false = 1
NOT true = 0
```

จากตัวอย่างเราตั้งค่าให้ว่าการแสดงผลลัพธ์ของดำเนินเชิงตรรกะมีอยู่ 2 ค่า คือ 0 ซึ่งมีค่าเป็น false และ 1 ซึ่งมีค่าเป็น true

4.10 ตัวดำเนินการกำหนดค่าแบบย่อ

Compound Assignment Operators เป็นการกำหนดค่าด้วยผลลัพธ์จากการดำเนินการ ซึ่งเป็นการเขียนตัวดำเนินการให้สั้นลง

ตัวดำเนินการ	ตัวอย่าง	ความหมาย (กำหนดให้ $x = 3$)
$+=$	$x += 3$	$x = x + 3 = 6$
$-=$	$x -= 3$	$x = x - 3 = 0$
$*=$	$x *= 3$	$x = x * 3 = 9$
$/=$	$x /= 3$	$x = x / 3 = 1$
$\%=$	$x \%= 3$	$x = x \% 3 = 0$

4.11 ตัวดำเนินการแปลงชนิดข้อมูล

Casting Operator เป็นการแปลงชนิดข้อมูลชนิดหนึ่งไปเป็นชนิดอื่น

Syntax

(datatype) expression

dataType	เป็นชนิดข้อมูลที่ต้องการเปลี่ยน
expression	ข้อมูลที่ต้องการเปลี่ยน

ตัวอย่าง

```
void setup() {
    Serial.begin(9600);
    float a = 2.34;
    Serial.print("No casting = ");
    Serial.println(a);
    Serial.print("Casting = ");
    Serial.println((int) a);

}

void loop() {

}
```

Output

```
No casting = 2.34
Casting = 2
```

4.12 การเลือกทำด้วยการกำหนดเงื่อนไข (Decision Making)

การเลือกทำ (Decision Making) เป็นการเขียนโปรแกรมให้สามารถเลือกจะให้ทำงานกับโค้ด คำสั่งหรือสเตตเมนต์ ส่วนใดด้วยเงื่อนไข

การเลือกทำด้วย if

if เป็นคำสั่งกำหนดเงื่อนไขเพื่อควบคุมให้โปรแกรมทำงานเฉพาะคำสั่งที่ต้องการเมื่อเงื่อนไข นั้นเป็นจริง

Syntax

```
if(condition)
{
    statement      //ส่วนของคำสั่งที่จะทำเมื่อเงื่อนไขเป็นจริง
}
```

condition	เงื่อนไขที่กำหนด ซึ่งมีผลลัพธ์เป็นจริงหรือเท็จ
-----------	--

ตัวอย่าง

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    int score = random(0,100);      //เบินหนังก์ขันสุ่มค่าตั้งแต่ 0 ถึง 100
    Serial.print("Your score is ");
    Serial.println(score);
    if(score >= 50){
        Serial.print("You passed\n");
    }
    delay(2000);
}
```

Output

```

∞ COM3
| |
Your score is 7          Output (เงื่อนไขไม่ตรง)
Your score is 49
Your score is 73          Output (เงื่อนไขตรง)
You passed
Your score is 58
You passed

```

4.13 การเลือกทำด้วย if...else

if...else เป็นการกำหนดเงื่อนไขให้โปรแกรมเลือกคำสั่ง เป็นการเพิ่มเติมการเลือกทำอีกตัว เลือกหนึ่ง

Syntax

```

if(condition)
{
    statement      //ส่วนของคำสั่งที่จะทำเมื่อเงื่อนไขเป็นจริง
}
else
{
    statement      //ส่วนของคำสั่งที่จะทำเมื่อเงื่อนไขไม่เป็นจริง
}

```

ตัวอย่าง

```

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

void loop() {
    int score = random(0,100);      //เบินพังก์ชนสุ่มค่าตั้งแต่ 0 ถึง 100
    Serial.print("Your score is ");
    Serial.println(score);
    if(score >= 50){
        Serial.println("You passed");
    }
    else
    {
        Serial.println("You failed");
    }
    delay(2000);
}

```

Output

```

COM3

Your score is 7
You failed
Your score is 49
You failed
Your score is 73
You passed
Your score is 58
You passed

```

Output เสื่อนไขไม่ตรง

Output เสื่อนไขตรง

4.14 การเลือกทำด้วย if...else if

if...else if เป็นการเลือกทำที่มีหลายทางเลือก โดยระหว่างตรวจสอบเงื่อนไข หากมีเงื่อนไข ข้อใดตรง ก่อนก็ทำการสั่งของบล็อกนั้น หากไม่มีคำสั่งใดเป็นจริงเลย จะทำการเข้าเงื่อนไข else

Syntax

```

if(condition) {
    statement1
}
else if (condition)
{
    statement2
}
else if (condition)
{
    statement3
}
else
{
    statement4
}

```

ในตัวอย่างที่จะแสดงต่อไปนี้เป็นโปรแกรมแสดงผลการเรียนเป็นเกรด A - F โดย

- เกรด A คะแนนอยู่ในช่วง 80 - 100
- เกรด B คะแนนอยู่ในช่วง 70 - 79
- เกรด C คะแนนอยู่ในช่วง 60 - 69
- เกรด D คะแนนอยู่ในช่วง 50 - 59
- เกรด F คะแนนต่ำกว่า 50

หากไม่อยู่ในช่วง 0 - 100 จะแสดงผลลัพธ์เป็น “Invalid score”

ตัวอย่าง

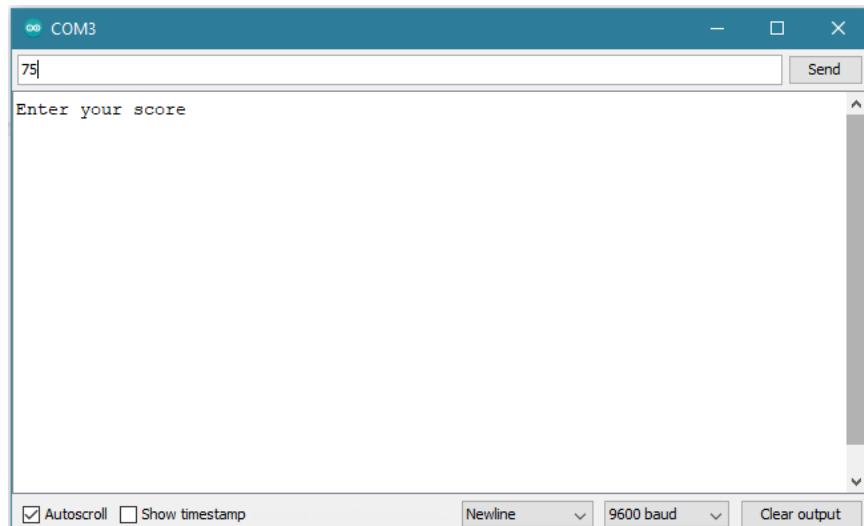
```

void setup() {
    Serial.begin(9600);
    Serial.println("Enter your score");
}
void loop() {
    String scoreText = Serial.readString();
    if(scoreText.toInt()){
        int score = scoreText.toInt();
        Serial.print("Your score is ");
        Serial.println(score);
        if(score >= 80 && score <=100)
        {
            Serial.println("Grade is A");
        }
        else if(score >= 70 && score <=79)
        {
            Serial.println("Grade is B");
        }
        else if(score >= 60 && score <=69)
        {
            Serial.println("Grade is C");
        }
        else if(score >= 50 && score <=59)
        {
            Serial.println("Grade is D");
        }
        else if(score >= 0 && score < 50)
        {
            Serial.println("Grade is F");
        }
        else
        {
            Serial.println("Invalid score");
        }
    }
}

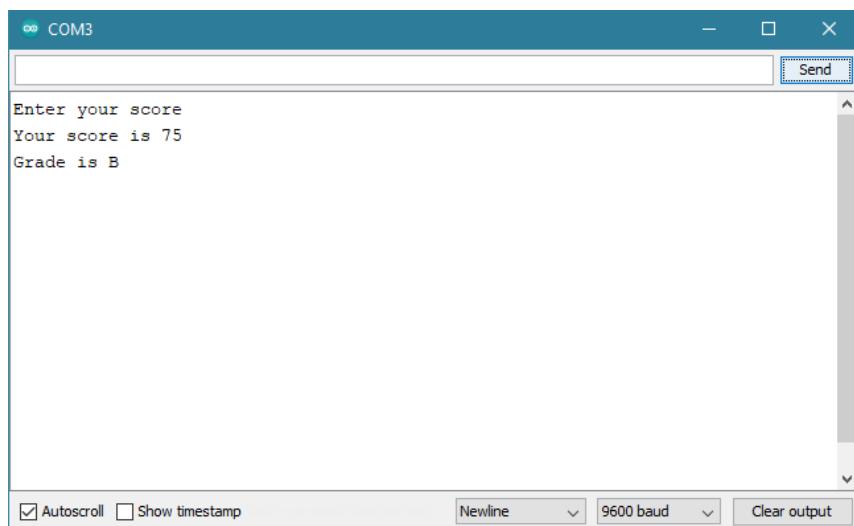
```

บรรทัดที่ 1	ฟังก์ชัน setup() ใช้เพื่อกำหนดค่าเริ่มต้น
บรรทัดที่ 5	ฟังก์ชัน loop() ใช้เพื่อการวนซ้ำไปเรื่อยๆ
บรรทัดที่ 6	นำข้อความที่กรอกผ่าน Serial Monitor ไปเก็บไว้ในตัวแปร scoreText ฟังก์ชัน Serial.readString() เป็นฟังก์ชันที่ใช้อ่านข้อความที่ส่งผ่าน Serial Monitor
บรรทัดที่ 7	เป็นการตรวจสอบว่า ข้อความที่กรอกมาสามารถแปลงเป็นตัวเลขได้หรือไม่ ส่วนฟังก์ชัน.toInt() ใช้เพื่อแปลงข้อความเป็นตัวเลข
บรรทัดที่ 8	แปลงข้อความเป็นตัวเลข และนำไปเก็บในตัวแปร score

ที่หน้าต่าง Serial Monitor กรอกตัวเลขแล้วกด Send



ตัวโปรแกรมจะนำไปเปรียบเทียบ และแสดงเกรด



4.15 การเลือกทำด้วย switch

ฟังก์ชัน switch...case เป็นการเลือกทำหลายทางเลือก มีการทำงานคล้ายกับ if...else if...else ต่างกัน ที่การตรวจสอบเงื่อนไขจะใช้การตรวจสอบการเท่ากันของตัวแปรที่ใช้ตรวจสอบ เท่านั้นโดยเมื่อตรวจสอบค่าแล้ว เท่ากับค่าที่กำหนดให้ทำฟังก์ชันที่เตรียมไว้

Syntax

```
switch (expression)
{
    case const1 :
        statement
        break;
    case const2 :
        state
        break;
    .
    .
    .
    [default : statemment]
}
```

expression:	นิพจน์ที่นำมาเปรียบเทียบกับค่าคงที่ใน case จะอยู่ในรูปแบบของตัวแปร คลาส หรือฟังก์ชัน โดยค่าได้ต้องเป็นชนิดข้อมูล
const1, const2,...:	ค่าคงที่ใน case ต้องเป็นข้อมูลเดียวกับนิพจน์ และค่าที่ในแต่ละ case ต้องไม่ซ้ำกัน เมื่อค่าของ นิพจน์ตรงกับค่าของ case ใด โปรแกรมจะทำงานในสเตตเมนต์ที่วางไว้หลังเครื่องหมาย : ของ case นั้น
break:	ตัวโปรแกรมจะໄลเปรียบเทียบ case แต่ละตัว หากตรงกับ case ไหนโปรแกรมทำงานตามส เตตเมนต์ที่อยู่หลังเครื่องหมาย : ของ case นั้น เมื่อเจอ break โปรแกรมจะออกจาก การเลือกทำแต่หากไม่มี break ปิดท้าย โปรแกรมจะทำการเปรียบเทียบ case ต่อไปเรื่อยๆจนกว่าจะเจอ break
default:	หากเปรียบเทียบค่า�ิพจน์แล้วไม่ตรงกับ case ใดเลย ตัว โปรแกรมจะมาทำสเตตเมนต์ของ default จนจบโดยไม่ต้องเขียนคำสั่ง break

ในตัวอย่างต่อไปจะเป็นการนำโปรแกรมแสดงผลการเรียนที่จากแต่เดิมใช้ if... else if ... else จะเปลี่ยนเป็นการใช้ switch มาเขียนโดยผลลัพธ์ยังคงเหมือนเดิม

ตัวอย่าง

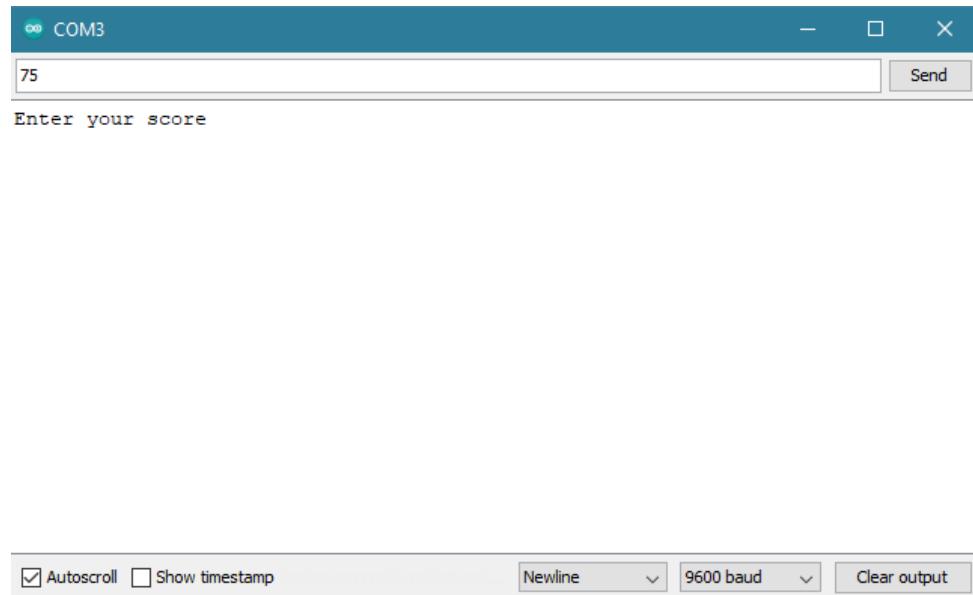
```

void setup() {
    Serial.begin(9600);
    Serial.println("Enter your score");
}

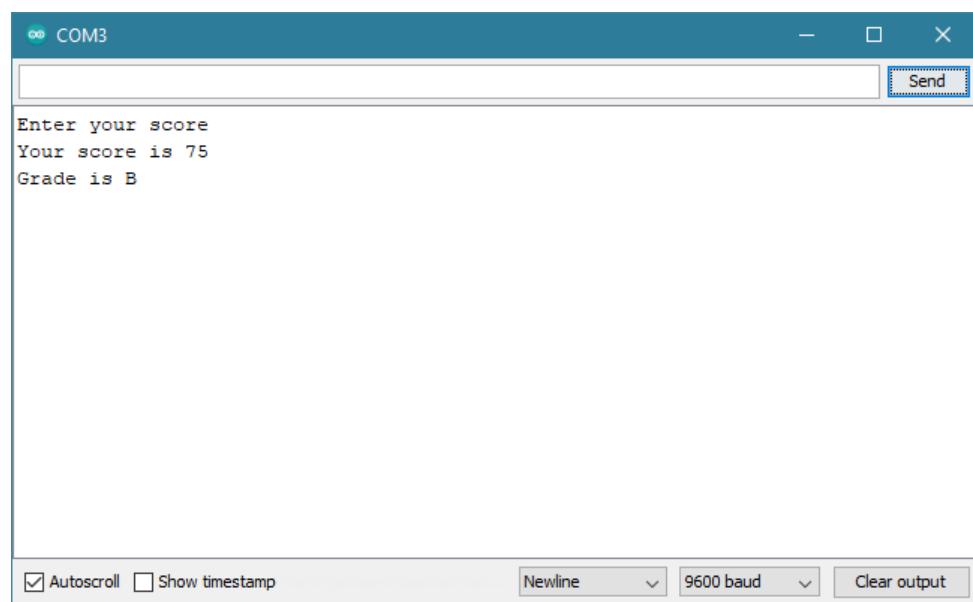
void loop() {
    String scoreText = Serial.readString();
    if(scoreText.toInt()){
        int score = scoreText.toInt();
        Serial.print("Your score is ");
        Serial.println(score);
        int num = score/10;
        switch(num){
            case 10:
            case 9:
            case 8:
                Serial.println("Grade is A");
                break;
            case 7:
                Serial.println("Grade is B");
                break;
            case 6:
                Serial.println("Grade is C");
                break;
            case 5:
                Serial.println("Grade is D");
                break;
            default:
                Serial.println("Grade is F");
        }
    }
}

```

กรอกตัวเลขแล้วกด Send



ตัวโปรแกรมจะนำไปปรีรียบเทียบ และแสดงเกรด



4.16 การวนทำซ้ำ (Loop)

หากเราต้องการให้มีการทำงานกับคำสั่งบางอย่างซ้ำๆ หลายครั้ง ก็สามารถใช้คำสั่งวนลูปซ้ำได้โดย ทำการกำหนดเงื่อนไข โดยตัวโปรแกรมจะทำงานซุดคำสั่งนั้นไปเรื่อยๆ จนกว่าเงื่อนไขจะไม่ตรงกับที่ได้กำหนดไว้

4.16.1 การวนทำซ้ำด้วยคำสั่ง while

while เป็นการให้ทำงานวนซ้ำโดยมีการตรวจสอบเงื่อนไขเป็น จริงจะทำงานตามชุดคำสั่ง ที่เขียนไว้ เมื่อทำงานเสร็จจะมีการวนกลับไป ตรวจสอบเงื่อนไขอีกโดยที่แบบนี้ไปเรื่อยๆ จนกว่า เงื่อนไขจะเป็นเท็จ จะทำการออกจากวงรอบการทำซ้ำ

Syntax

```
while (condition)
{
    statement
}
```

condition	เงื่อนไขที่ใช้ตรวจสอบก่อนเข้าทำงานเตตเมนต์ ซึ่งมีผลเป็นจริงหรือเท็จ
-----------	---

ตัวอย่างของโปรแกรมต่อไปนี้จะเป็นการให้ค่าตัว i = 10 และให้ลดค่าลงไปเรื่อยๆ จนกว่าค่า i จะเท่ากับ 1 โดยใช้คำสั่ง while

ตัวอย่าง

```
void setup() {
  Serial.begin(9600);
  int i = 10;
  while(i>0){
    Serial.print("Value of i = ");
    Serial.println(i);
    i--;
  }
}
void loop() {
```

Output

```
Value of i = 10
Value of i = 9
Value of i = 8
Value of i = 7
Value of i = 6
Value of i = 5
Value of i = 4
Value of i = 3
Value of i = 2
Value of i = 1
```

4.16.2 การวนทำข้าด้วยคำสั่ง do...while

while เป็นการให้ทำงานวนซ้ำ โดยจะทำการสั่งที่เขียนไว้ก่อน 1 ครั้ง และวิจัยตรงตรวจสอบเงื่อนไข หากเงื่อนไขยังเป็นจริงจะทำการวนซ้ำต่อ หากไม่ตรงเงื่อนไขจะออกจาก循环

Syntax

```
do
{
    statement
}while(condition);
```

condition	เงื่อนไขที่ใช้ตรวจสอบก่อนเข้าทำงานเตตเมนต์ ซึ่งมีผลเป็นจริงหรือเท็จ
------------------	---

ตัวอย่างของโปรแกรมต่อไปนี้จะเป็นการให้ตัวแปร i = 1 โดยให้นับไปจนกระทั่งค่า i มีค่า เท่ากับ 10 ซึ่งในระหว่างการนับจะมีการตรวจสอบว่าเป็นจำนวนเลขคู่หรือจำนวนเลขคี่โดยใช้ do-while

ตัวอย่าง

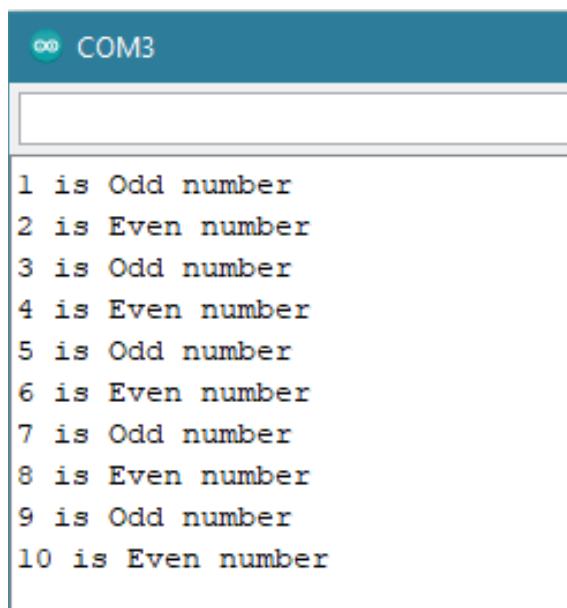
```

void setup() {
    Serial.begin(9600);
    int i = 1;
    do{
        if(i%2 == 0){           //หาก i%2 เท่ากับ 0 แสดงว่าเลขเป็นจำนวนคู่
            Serial.print(i)
            Serial.println(" is Even number");
        }
        else{                  //หากไม่ใช่ แสดงว่าเป็นเลขจำนวนคี่
            Serial.print(i)
            Serial.println(" is Odd number");
        }
        i++;
    }while(i <= 10);

}
void loop() {
}

```

Output



```

1 is Odd number
2 is Even number
3 is Odd number
4 is Even number
5 is Odd number
6 is Even number
7 is Odd number
8 is Even number
9 is Odd number
10 is Even number

```

4.16.3 การวนทำซ้ำด้วยคำสั่ง for

for เป็นการวนซ้ำในขณะที่เงื่อนไขเป็นจริงคล้ายกับ while แต่ for สามารถกำหนดจำนวนครั้งที่ แน่นอน โดยเริ่มจากค่าเริ่มต้นของตัวนับไปจนถึงค่าที่เป็นเงื่อนไขแล้วหยุดวนรอบ ขณะที่ while และ do...while ต้องกำหนดการเพิ่มหรือลดค่าที่อยู่ภาย statement เอง

Syntax

```
for (initial; condition; step)
{
    statement
}
```

initial	กำหนดค่าเริ่มต้นในการนับ
condition	กำหนดเงื่อนไขการทำซ้ำ โดยให้วนรอบทำซ้ำหากเงื่อนไขเป็นจริง และทำซ้ำ หากเงื่อนไขเป็นเท็จ
step	ค่าตัวเลขที่ใช้เพิ่มหรือลดค่าเมื่อจบการทำงานในสเตตเมนต์ในแต่ละครั้ง

ตัวอย่างของโปรแกรมต่อไปนี้จะเป็นการให้ค่าตัว i = 10 และให้ลดค่าลงไปเรื่อยๆ จนกว่าค่า i จะเท่ากับ 1 โดยใช้คำสั่ง for

ตัวอย่าง

```
void setup() {
    Serial.begin(9600);
    int i;
    for (i = 10 ; i > 0 ; i--)
    {
        Serial.print("Value of i = ");
        Serial.println(i);
    }
}
void loop() { }
```

Output

```
Value of i = 10
Value of i = 9
Value of i = 8
Value of i = 7
Value of i = 6
Value of i = 5
Value of i = 4
Value of i = 3
Value of i = 2
Value of i = 1
```

4.17 พังก์ชัน

พังก์ชันคือกลุ่มของคำสั่งที่ใช้ทำงาน และมีชื่อไว้สำหรับเรียกใช้งานผ่านโปรแกรมหลัก หรือพังก์ชันหลัก main() ช่วยให้การเขียนโปรแกรมที่มีขนาดใหญ่หรือซับซ้อนทำได้ง่ายขึ้น

ใน Arduino IDE จะมีพังก์ชันหลักคือ setup() และ loop() ที่เรียกพังก์ชันต่างๆ ที่เราสร้างขึ้นมาเอง เพื่อควบคุมการทำงานของบอร์ด

การกำหนดพังก์ชัน

การประกาศพังก์ชัน เพื่อให้พังก์ชันหลักสามารถเรียกใช้ได้ มีรูปแบบคือ

Syntax

```
return_type function_name (parameter list) {
    function body
}
```

return_type	ชนิดข้อมูลที่ต้องส่งคืนค่ากลับไปเป็นผลลัพธ์ให้กับฟังก์ชันหลัก เช่น int, char, float เป็นต้น
function_name	ตัวชื่อฟังก์ชันที่สร้าง โดยวิธีเป็นวิธีเดียวกันกับการสร้างตัวแปร
parameter list	พารามิเตอร์มีหน้าที่นำค่าจากจุดที่เรียกใช้งานเข้ามาในฟังก์ชัน โดยพารามิเตอร์ประกอบด้วย ชนิดข้อมูลและชื่อพารามิเตอร์ หากมีพารามิเตอร์หลายตัวทำมีการคั่นด้วย คอมม่า (,)
function body	เป็นกลุ่มคำสั่งหรือ statement ที่กำหนดว่าให้ทำงานอะไร โดยอาจมีการส่งผลลัพธ์กลับไปยังจุดเรียกฟังก์ชันได้

ตัวอย่าง ฟังก์ชันที่มีการคืนค่า

```
int maximum (int val1, int val2)
{
    int max;
    if (val1 > val2) //เปลี่ยนเที่ยมหาค่าสูงสุด
        max = val1;
    else
        max = val2;
    return max;      //ส่งคืนค่าสูงสุดเป็นรูปแบบ int
}
```

ตัวอย่าง ฟังก์ชันที่ไม่มีการคืนค่า

```
void maximum (int val1, int val2)
{
    int max;
    if (val1 > val2) //เปลี่ยนเที่ยมหาค่าสูงสุด
        max = val1;
    else
        max = val2;
    Serial.print("Max value is: ");
    Serial.println(max);
}
```

4.18 การประกาศฟังก์ชัน

เป็นการบอกให้คอลไฟเลอร์ทราบเกี่ยวกับชื่อ ชนิดข้อมูลในการส่งคืนค่า และพารามิเตอร์ของ ฟังก์ชัน ก่อนที่จะให้ฟังก์ชัน main() ทำงาน ซึ่งจะประกาศไว้ก่อนฟังก์ชัน main() หรือเรียกว่า ฟังก์ชัน prototype (Function Prototype)

Syntax

```
#include <Library>
return_type function_name (parameter list); //ประกาศฟังก์ชัน
void setup () {
    statement
}
void loop () {

}

return_type function_name (parameter list) //กำหนดฟังก์ชัน
{
    function_body
}
```

4.19 การเรียกใช้ฟังก์ชัน

เราสามารถเรียกใช้ฟังก์ชันได้ เมื่อกำหนดและประกาศฟังก์ชันโดยเรียกใช้ผ่านฟังก์ชัน main() พร้อมมี การส่งค่าข้อมูล หรือเรียกว่า อาร์กิวเมนต์ (argument) ไปยังพารามิเตอร์ของฟังก์ชัน หรืออาจไม่มีการส่งข้อมูลไป ก็ได้เช่นกัน จากนั้นก็นำไปประมวลผลและส่งผลลัพธ์คืนมาบังฟังก์ชัน main() เมื่อฟังก์ชันทำงานเสร็จ

Syntax

```
#include <Library>
return_type function_name (parameter list); //ประกาศฟังก์ชัน
void setup () {
    ...
    [var_return = ] function_name([arguemnt]);
    ...
}
void loop () {
```

ตัวอย่าง การเรียกใช้งานฟังก์ชันแบบส่งคืนค่า

```

int maximum (int , int);
void setup(){
    Serial.begin(9600);
    int x = 10, y = 20;
    int maxNum;
    maxNum = maximum(x,y);
    Serial.print("Max value is: ");
    Serial.println(maxNum);
}
void loop{
}
int maximum (int val1, int val2)
{
    int max;
    if (val1 > val2) //เบริญบที่ยมหาค่าสูงสุด
        max = val1;
    else
        max = val2;
    return max;      //ส่งคืนค่าสูงสุดเป็นรูปแบบ int
}

```

ตัวอย่าง การเรียกใช้งานฟังก์ชันแบบไม่ส่งคืนค่า

```

void maximum (int , int);
void setup(){
    Serial.begin(9600);
    int x = 10, y = 20;
    maximum(x,y);
}
void loop() {
}
void maximum (int val1, int val2)
{
    int max;
    if (val1 > val2) //เบริญบที่ยมหาค่าสูงสุด
        max = val1;
    else
        max = val2;
    Serial.print("Max value is: ");
    Serial.println(max);
}

```

4.20 คลาสและออบเจกต์

เป็นการเขียนโปรแกรมแบบโครงสร้างที่มีการทำงานของคำสั่งแบบเรียงลำดับกันและแยกการเขียนโปรแกรมออกจากเป็นส่วน ๆ ซึ่งจะทำงานขึ้นตรงกับข้อมูล

คลาส เป็นการกำหนดส่วนประกอบต่าง ๆ ที่จะนำไปสร้างออบเจกต์โดยคลาส จะประกอบไปด้วย ส่วนอย่างคือ

- Fields ตัวแปร ตัวแปรใช้สำหรับเก็บข้อมูลต่างๆ เกี่ยวกับออบเจกต์
- Method จะเป็นการกำหนดฟังก์ชันการทำงานของออบเจกต์

4.20.1 การสร้างคลาส

คลาสมีลักษณะเป็นเหมือนชนิดข้อมูลที่ผู้ใช้กำหนดขึ้นมาใช้งานเอง ดังนั้นการนำไปใช้งานจะต้องประกาศตัวแปรเป็น ออบเจกต์ การประกาศสร้างคลาสจะเริ่มต้นด้วยคีย์เวิร์ด class ตามด้วยชื่อของ คลาส และตัวของคลาส (Body) ที่อยู่ภายใต้เครื่องหมาย { } ส่วนรายการของสมาชิกจะอยู่ภายใต้ตัวของ คลาส

Syntax

```
class Class_name{
    Class Access Modifiers :
        member 1;
        member 2;
        member ...
};
```

Class_name	ชื่อคลาสโดยกำหนดเป็นชื่ออะไรก็ได้ที่ไม่ซ้ำกับคลาสอื่นและคีย์เวิร์ดที่ส่วนไว้
Class Access Modifier	<p>เป็นการกำหนดสิทธิ์การเข้าถึงตัวแปรข้อมูลและ Method ซึ่งมี 3 คีย์เวิร์ดได้แก่</p> <ul style="list-style-type: none"> ● private เข้าถึงได้เฉพาะภายในคลาสเดียวกัน ● protected เข้าถึงเฉพาะภายในคลาสเดียวกันหรือคลาสที่ถูกสืบทอดหรือคลาสลูก ● public เข้าได้ทุกที่ทั้งภายในและภายนอกคลาส

	<ul style="list-style-type: none"> *หากไม่มีการกำหนด Class Access Modifier จะทำให้เป็น default ซึ่งก็คือถูกตั้งให้เป็น private ทันที
member	เป็นสมาชิกคลาสมี 2 กลุ่ม คือข้อมูลตัวแปร กับ Method (หรือฟังก์ชัน)

ตัวอย่าง

```
class Box{
    public:
        double L;
        double W;
        double H;
};
```

4.20.2 การสร้าง Method

เป็นการสร้าง Method อัญญาณในคลาส โดยเราสามารถกำหนดการทำงานให้อยู่ภายใต้คลาส หรือภายนอกคลาสนอกได้ ดังตัวอย่าง

การกำหนด Method ภายใต้คลาส

ตัวอย่าง

```
class Box{
    public:
        double L;
        double W;
        double H;
        double getVol(void){
            return L * W * H;
        }
};
```

การกำหนด Method ภายในอօคลาສ

ตัวอย่าง

```
class Box{
public:
    double L;
    double W;
    double H;
    double getVol(void); //กำหนดปร๊าติใหม่ของเมธอด getVol
    double getSurf(void); //กำหนดปร๊าติใหม่ของเมธอด getSurf
};
```

4.20.3 การสร้างอօบเจ็กต์

การสร้างอօบเจ็กต์จากคลาສมีวิธีเหมือนการประกาศตัวแปรทั่วไป โดยกำหนดให้ชื่อคลาສเป็น ชนิดข้อมูล และตามด้วยอօบเจ็กต์เป็นชื่อของตัวแปร

Syntax

Class_Name Object_Var;

Class_name	ชื่อคลาສที่จะนำมาสร้างอօบเจ็กต์
Object_var	ชื่ออօบเจ็กต์ที่ถูกสร้างด้วยคลาສ

ตัวอย่าง

```
Box V1;
Box V2;
```

4.20.4 การเข้าถึงสมาชิกของคลาส

ออกแบบสามารถเข้าถึงและทำงานกับสมาชิกที่เป็นตัวแปรและ Method ในคลาสได้โดยการ พิมพ์ชื่อ ออกแบบได้ตามด้วยเครื่องหมาย . และชื่อของตัวแปรหรือ Method

ตัวอย่าง

```
double vo,su;
V1.L = 5;
V1.W = 4;
V1.H = 2;
vo = V1.getVol();
su = V1.getSurf();
```

ตัวอย่างการเขียนโปรแกรมโดยการนำคลาส Box มาใช้คำนวณหาปริมาตรและพื้นที่ของรูปทรง สี่เหลี่ยมผืนผ้าออกแบบที่ชื่อ myBox และคำนวณโดยใช้ Method getVol() หาปริมาตร และ getSurf() หาพื้นที่ผิว

ตัวอย่าง

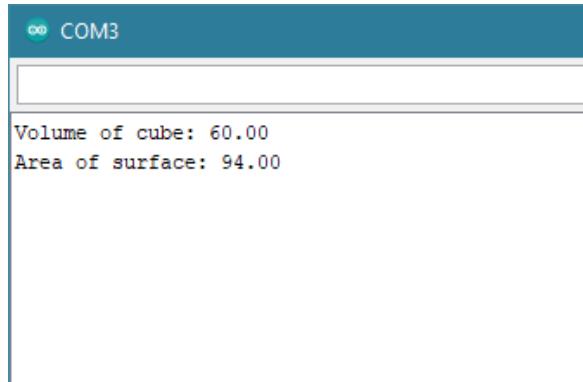
```
class Box{
public:
    double L;
    double W;
    double H;
    double getVol(void); //กำหนดค่าบอร์ดไทม์ของเมธอด getVol
    double getSurf(void); //กำหนดค่าบอร์ดไทม์ของเมธอด getSurf
};

double Box::getVol(void){
    return L * W * H;
}
double Box::getSurf(void){
    return (2*L*W)+(2*L*H)+(2*W*H);
}
void setup(){
    Serial.begin(9600);
    Box myBox;
    myBox.L = 4.0;
    myBox.W = 5.0;
    myBox.H = 3.0;
    Serial.print("Volume of cube: ");
    Serial.println(myBox.getVol());
    Serial.print("Area of surface: ");
    Serial.println(myBox.getSurf());
}

void loop(){

}
```

Output



4.21 คำสั่งพื้นฐานสำหรับการใช้งาน Arduino

`pinMode(PIN , Mode)`

- คือการตั้งค่าให้ PIN นั้นเป็น Mode ที่เราต้องการ
- PIN – PIN ต่าง ๆ ที่ต้องการใช้งาน
- Mode – INPUT , OUTPUT

`digitalWrite(PIN , status)`

- คือการส่งสัญญาณออกไปแบบ digital
- PIN – PIN ต่าง ๆ ที่ต้องการใช้งาน
- Status – HIGH , LOW หรือ 1 , 0

`digitalRead(PIN)`

- คือการรับอ่านค่าสัญญาณแบบ digital
- PIN – PIN ต่าง ๆ ที่ต้องการใช้งาน

`analogWrite(PIN)`

- คือการรับอ่านค่าสัญญาณแบบ analog
- PIN – PIN ต่าง ๆ ที่ต้องการใช้งาน

`analogRead(pin)`

- คือการรับอ่านค่าสัญญาณแบบ analog
- PIN – PIN ต่าง ๆ ที่ต้องการใช้งาน

`delay(time)`

- ใช้หน่วงเวลาทำงานก่อนทำงานคำสั่งต่อไป
- `time` – ระยะเวลาที่ต้องการให้หน่วงไว้ หน่วยเป็นมิลลิวินาที

`delayMicroseconds(time)`

- ใช้หน่วงเวลาทำงานก่อนทำงานคำสั่งต่อไป
- `time` – ระยะเวลาที่ต้องการให้หน่วงไว้ หน่วยเป็นไมโครวินาที

`Serial.begin(x)`

- ตั้งค่าเริ่มต้นเพื่อติดต่อสื่อสารกับคอมพิวเตอร์
- `x` – อัตราเร็วในการสื่อสาร หน่วยบิตต่อวินาที (ส่วนใหญ่จะกำหนดที่ 9600 หรือ 115200)

`Serial.print("sentence")`

- ใช้พิมพ์ประโยคเพื่อให้แสดงผลบนจอคอมแบบไม่เว้นบรรทัด

`Serial.println("sentence")`

- ใช้พิมพ์ประโยคเพื่อให้แสดงผลบนจอคอมแบบเว้นบรรทัด

`Serial.available()`

- ใช้ตรวจสอบว่ามีการกดคีย์บอร์ดหรือไม่

`Serial.Read()`

- ใช้อ่านค่าปุ่มคีย์บอร์ดที่กด

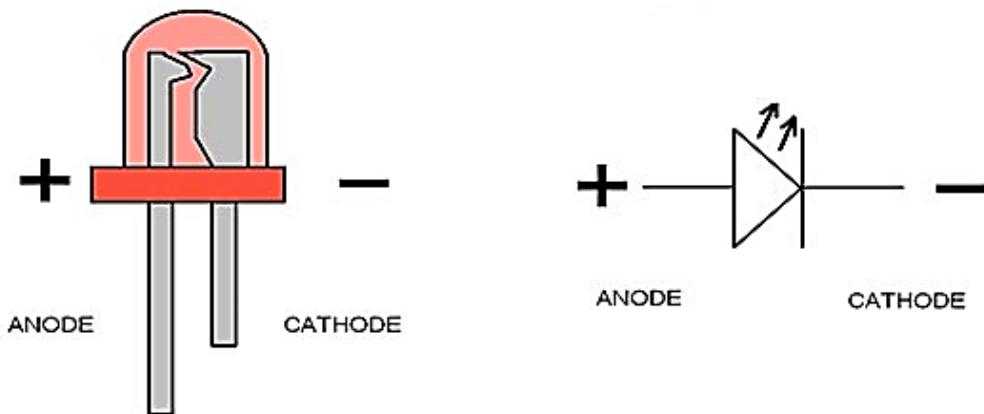
Workshop 1 สัญญาณไฟกระพริบทดสอบไมโครคอนโทรลเลอร์ ESP8266

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

LED คืออะไร

LED ย่อมาจาก Light Emitting Diode คือ ไดโอดชนิดปล่อยแสง

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



ประเภทของ LED

1. LED แบบดั้งเดิม

กำลังวัตต์น้อย ขนาดหรือรูปร่างหรือสีขึ้นอยู่กับพลาสติกที่ใช้ทำเปลือกหุ้ม ใช้ทำไฟสัญญาณในวงจร



LED แบบดั้งเดิม

2. LED ขนาดเล็กมาก

ใช้เทคโนโลยีการเชื่อมเม็ด LED ติดลงไปกับแผงวงจรหรือที่เรียกว่า Surface Mounting Technology (SMT) ซึ่งเป็นวิธีการเดียวกับการประกอบชิ้นส่วนอิเล็กทรอนิกส์และsemi-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 ถ้าระบายน้ำร้อนไม่ดีอาจจะทำให้อุปกรณ์ชำรุดหรือเสียหายในภายโน้มากวินาที LED ชนิดนี้ส่วนใหญ่จะใช้ทำอุปกรณ์ให้แสงสว่างหรือจะเข้ามาทดแทนหลอดไฟที่ใช้อยู่ในปัจจุบัน



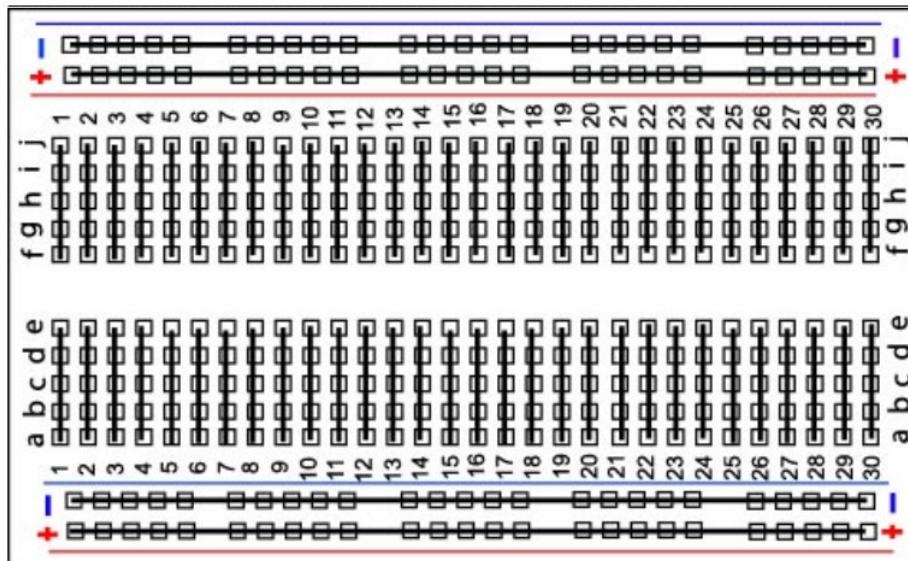
CREE XMA Series COB



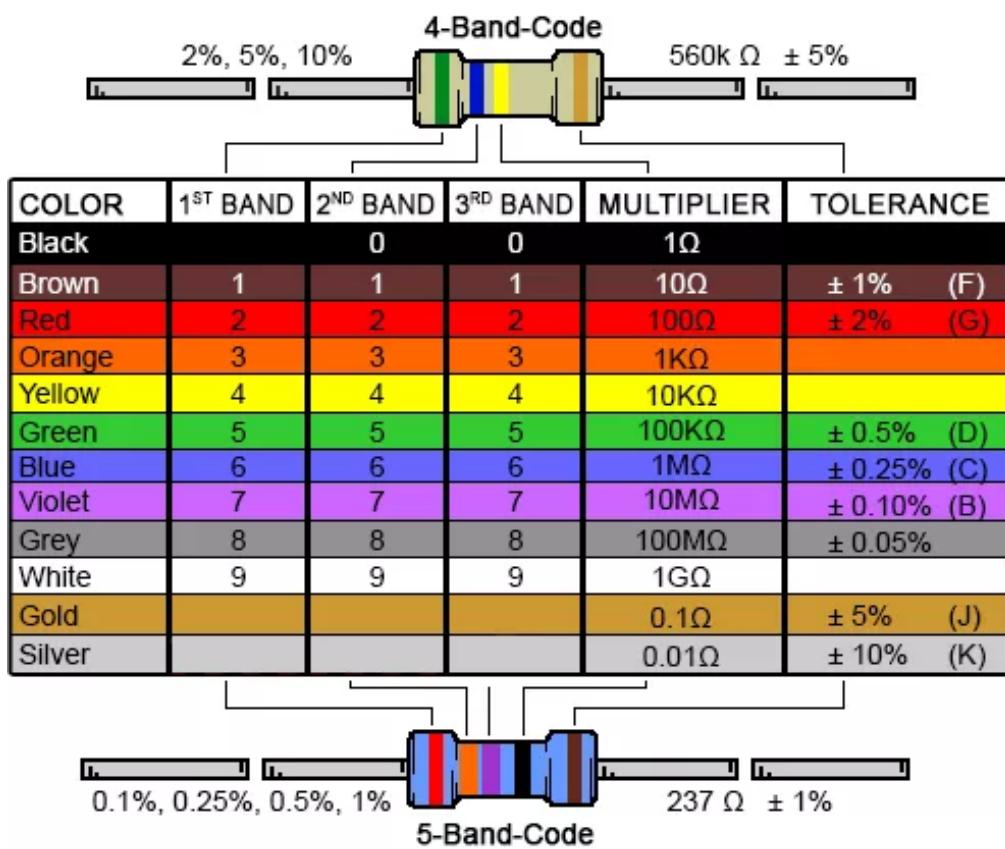
Bridgelux Vero Series COB

LED กำลังสูง หรือ Hi-power LED

การต่อวงจรภายในแผ่นบอร์ดทดลอง (Breadboard)



การอ่านสีตัวต้านทาน



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

จากภาพเป็นตารางสีในการอ่านค่าตัวต้านทาน ตัว R อ่านได้ดังต่อไปนี้

สำหรับแบบ 4 Band (หรือ 4 สี)

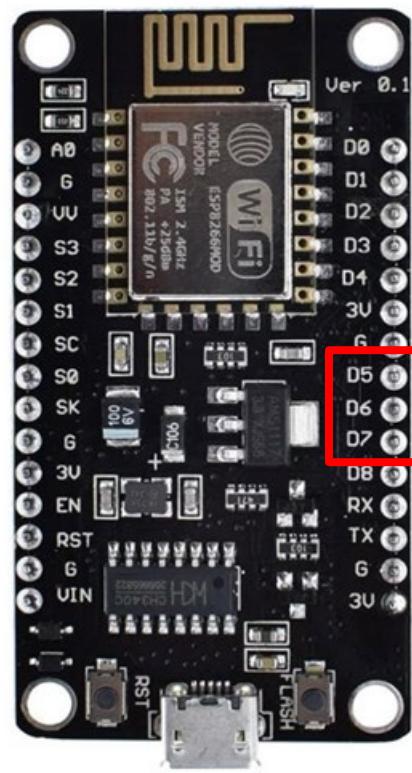
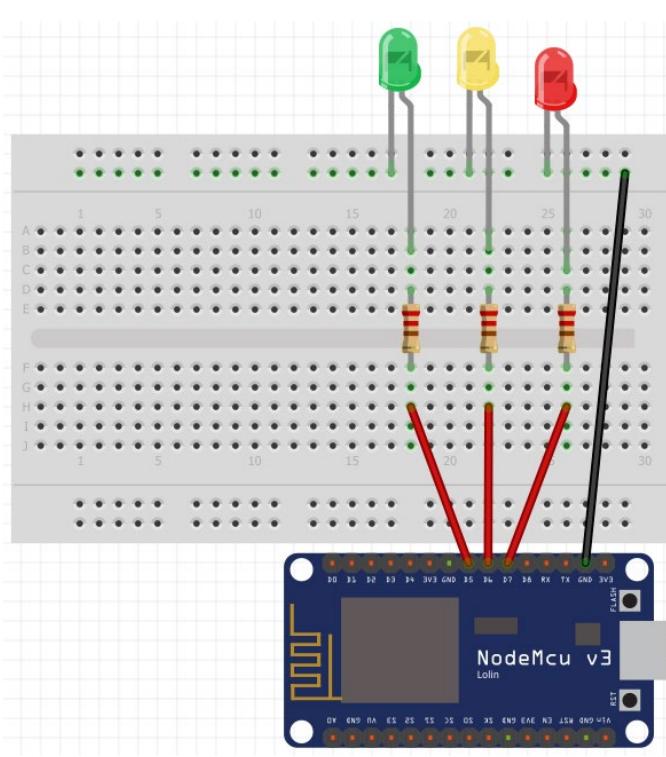
- ขั้นที่ 1 หันตัว ตัวต้านทาน (R) โดยให้แบบสีความคลาดเคลื่อน สีเงินและสีมองไปทางขวา
- แถบสีที่ 1 เป็นค่าตำแหน่งที่ 1
- แถบสีที่ 2 เป็นค่าตำแหน่งที่ 2
- แถบสีที่ 3 เป็นตัวคูณ
- แถบสีที่ 4 เป็นค่าความผิดพลาด บวกลบ

อุปกรณ์ที่ต้องใช้งาน

1. NodeMCU ESP8266
2. หลอดไฟ LED สีแดง ,สีเหลือง, สีเขียว
3. ตัวต้านทาน 220 โอห์ม 3 ตัว (แถบสี แดง แดง ดำ ดำ น้ำตาล)
4. สายไฟ

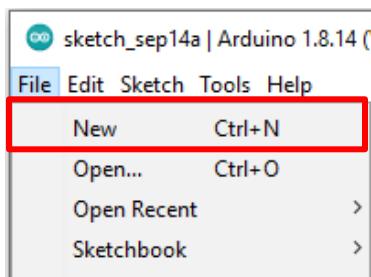
การต่อวงจร

PIN	อุปกรณ์
D5	LED สีเขียว
D6	LED สีเหลือง
D7	LED สีแดง



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 อีกดวงก็ติดขึ้น

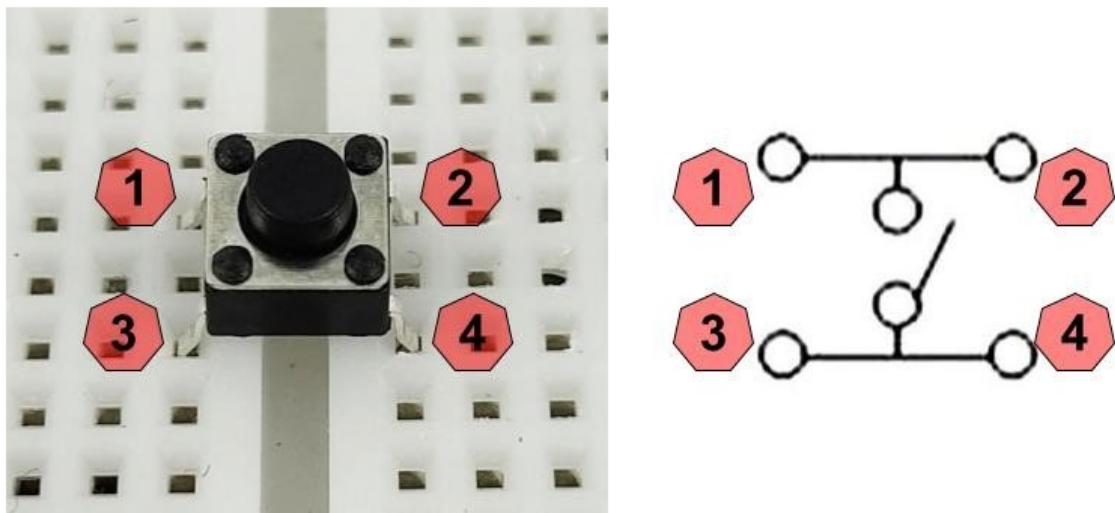
Workshop 2 การใช้งาน Push button ในการเปิด – ปิดหลอดไฟ LED

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



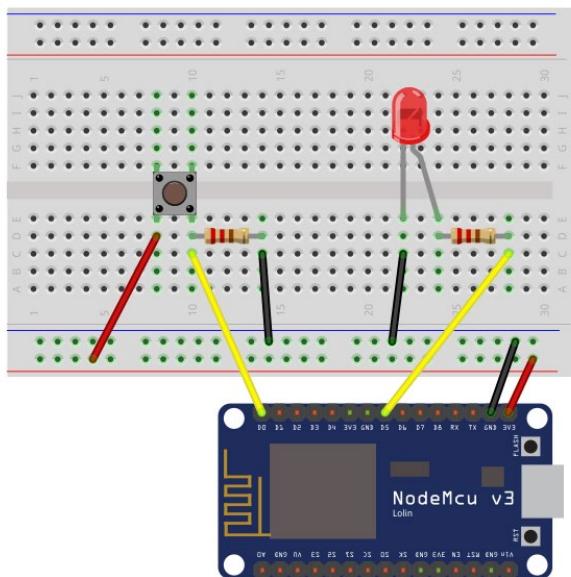
รูปจาก <https://th.cytron.io/p-6x6x1-push-button-4pin>

อุปกรณ์ที่ต้องใช้งาน

1. NodeMCU ESP8266
2. หลอดไฟ LED
3. ตัวต้านทาน 220 โอห์ม 2 ตัว (ແຄບສື ແດງ ແດງ ດຳ ດຳ ນ້ຳຕາລ)
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) { //?ขึ้นการเช็คสถานะของ 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 และเมื่อปล่อยปุ่มก็จะทำให้หลอดไฟดับ

Workshop 3 เปิดปิด หลอดไฟ LED และ pump น้ำโดยใช้ relay

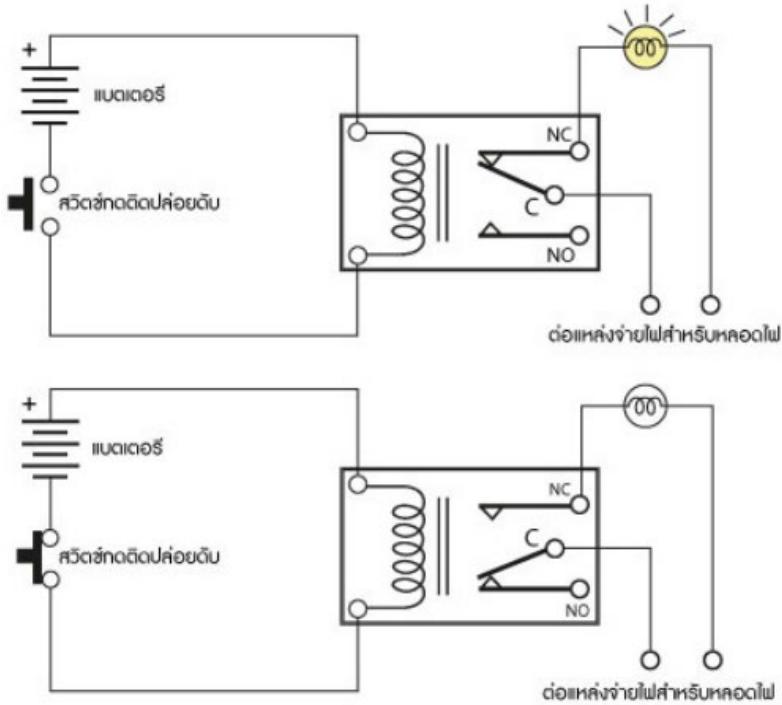
Workshop นี้จะทำการสั่งการทำงาน เปิด-ปิด หลอดไฟ LED และ pump น้ำ โดยใช้ relay เป็นตัวควบคุมการทำงาน

รีเลย์ (Relay) คืออะไร

เป็นอุปกรณ์ที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานแม่เหล็ก เพื่อใช้ในการดึงดูดหน้าสัมผัสของคอนแทคให้เปลี่ยนสภาพ โดยการป้อนกระแสไฟฟ้าให้กับขดลวด เพื่อทำการปิดหรือเปิดหน้าสัมผัคล้ายกับสวิตช์ อิเล็กทรอนิกส์ ซึ่งความสามารถนำรีเลย์ไปประยุกต์ใช้ในการควบคุมวงจรต่าง ๆ ในงานช่างอิเล็กทรอนิกส์มากมาย

Relay ประกอบด้วยส่วนสำคัญ 2 ส่วนหลักคือ

- ส่วนของขดลวด (coil) เหนี่ยวนำกระแสต่อ ทำหน้าที่สร้างสนามแม่เหล็กไฟฟ้าให้แกนโลหะไปกระแทกให้หน้าสัมผัสดังกล่าว ทำงานโดยการรับแรงดันจากภายนอกต่อครั้งที่ขดลวดเหนี่ยวนำนี้ เมื่อขดลวดได้รับแรงดัน(ค่าแรงดันที่รีเลย์ต้องการขึ้นกับชนิดและรุ่นตามที่ผู้ผลิตกำหนด) จะเกิดสนามแม่เหล็กไฟฟ้าทำให้แกนโลหะด้านในไปกระแทกให้แผ่นหน้าสัมผัสดังกล่าว
- ส่วนของหน้าสัมผัส (contact) ทำหน้าที่เมื่อนอนสวิตช์จ่ายกระแสไฟให้กับอุปกรณ์ที่เราต้องการนั่นเองจุดต่อใช้งานมาตรฐาน ประกอบด้วย
 - จุดต่อ NC ย่อมาจาก normal close หมายความว่าปกติปิด หรือ หากยังไม่จ่ายไฟให้ขดลวดเหนี่ยวนำหน้าสัมผัสจะติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการให้ทำงานตลอดเวลา
 - จุดต่อ NO ย่อมาจาก normal open หมายความว่าปกติเปิด หรือหากยังไม่จ่ายไฟให้ขดลวดเหนี่ยวนำหน้าสัมผัสจะไม่ติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการควบคุมการเปิด ปิด เช่น คอมไฟสนามหนือน้ำบ้าน
 - จุดต่อ C ย่อมาจาก common คือจุดร่วมที่ต่อมาจากแหล่งจ่ายไฟ



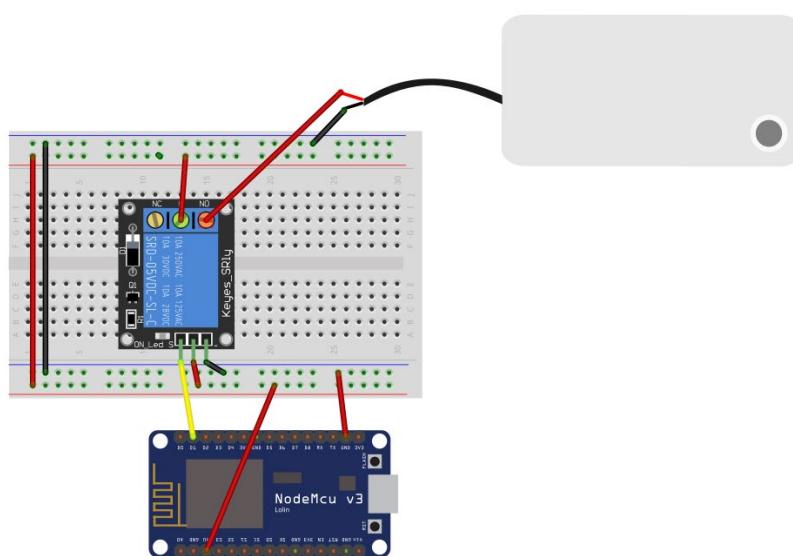
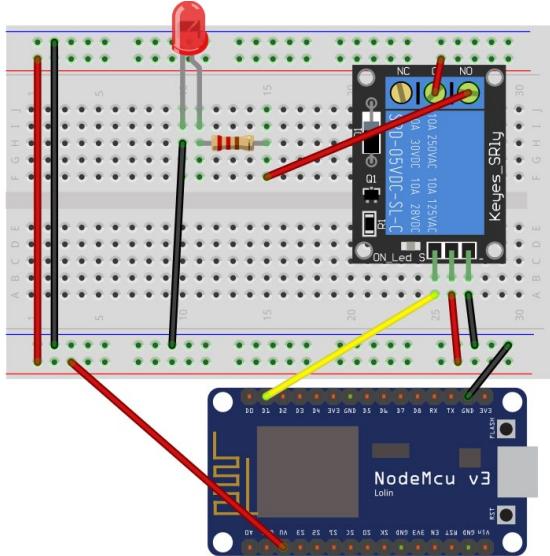
Relay Module สามารถนำมาทำ project ควบคุมเปิดปิดอุปกรณ์ไฟฟ้าภายในบ้านได้ รีเลย์ จะทำหน้าที่เหมือนสวิตซ์ทางไฟฟ้า ควบคุมการสับสะพานไฟด้วยสัญญาณ Digital 1 0 หรือ High กับ Low ปกติการเปิด ปิดไฟบ้านเราจะใช้น้ำในการกดสวิตซ์ แต่ถ้าเราใช้รีเลย์แทนสวิตซ์ เราสามารถนำสัญญาณดิจิตอลมาควบคุมการ เปิด-ปิดได้ สามารถนำมาประยุกต์ทำ 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 (ແລບສີ ແດງ ແດ ດຳ ນໍາຕາລ)

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 1 วินาที และหยุดทำงานเป็นเวลา 1 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 แคนเดเมียมแซลไฟร์ดและแคนเดเมียมชิลีโน๊ด ซึ่งเป็นสารที่มีการตอบสนองความยาวคลื่นแสง ฉบับอยู่เป็นเส้นลักษณะเป็นขนาด คดเคี้ยวไปมาเป็นฐานเซรามิก LDR จะมีสองขั้ว ซึ่งมีค่าความต้านทานภายใต้ตัวเปลี่ยนแปลงค่าได้ตามแสงที่ตกลงมากระทบ

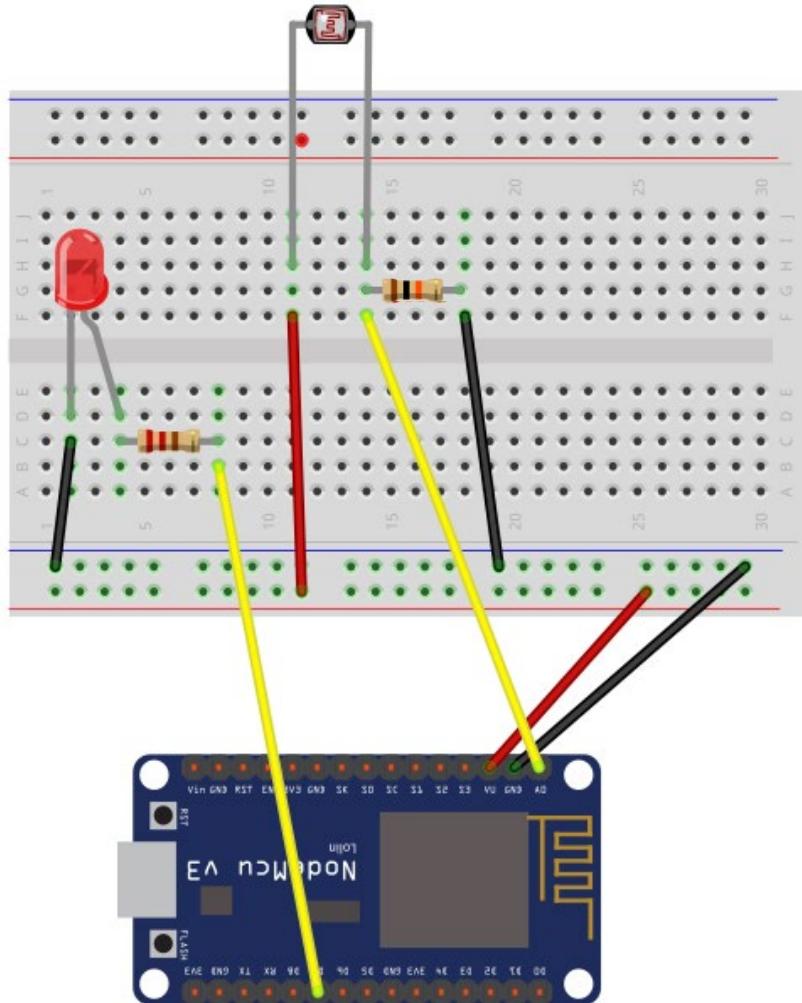


อุปกรณ์ที่ต้องใช้งาน

1. NodeMCU ESP8266
2. LDR 1 ตัว
3. LED สีแดง mm 1 ดวง
4. ความต้านทาน 220 โอห์ม 1 ตัว (ແກບສี แดง แดง ดำ ดำ น้ำตาล)
5. ความต้านทาน 10K โอห์ม 1 ตัว (ແກບສี น้ำตาล ดำ ดำ แดง น้ำตาล)

การต่อวงจร

PIN	อุปกรณ์
A0	LDR
D7	LED



การเขียน code

- Code สำหรับอ่านค่า LDR

Workshop_LDR_read

```
#define analog_pin A0

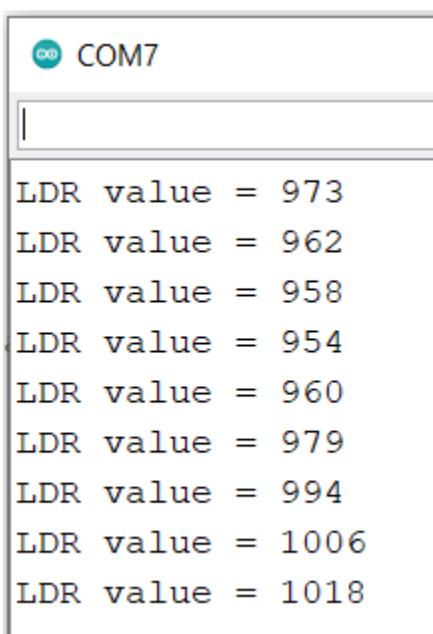
int ldr_value = 0;

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";

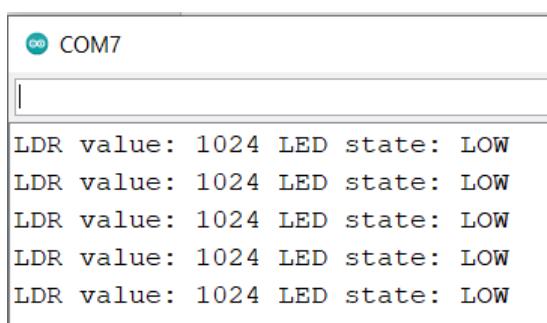
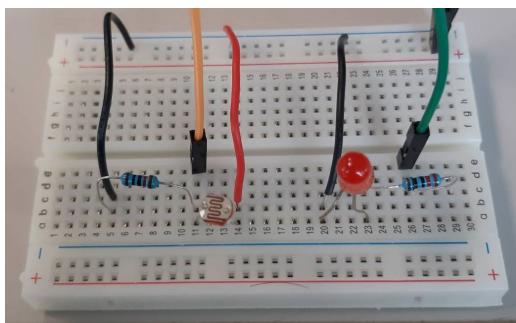
void setup() {
    //กำหนด mode ให้กับ pin
    pinMode(LED, OUTPUT);
    digitalWrite(LED, LOW); //ส่งสัญญาณ LOW ไปที่ LED
    Serial.begin(9600);
}

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);
}
```

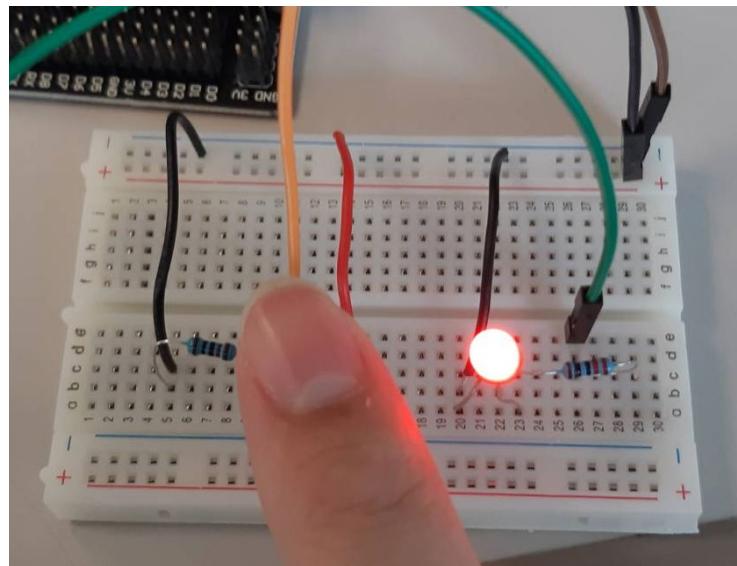
สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_4/Workshop_LDR_LED/Workshop_LDR_LED.ino

ผลลัพธ์ที่ได้

ผลลัพธ์ขณะที่ไม่มีวัตถุมาบัง 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

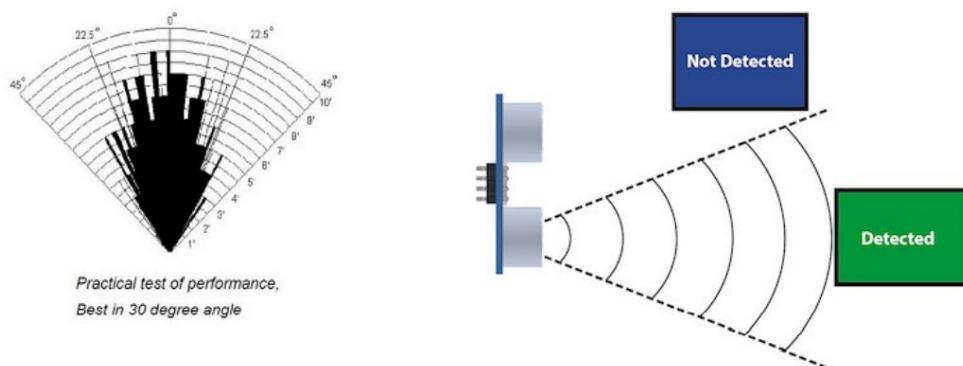
การทดลองนี้จะทำการตรวจวัดระยะทางด้วยโมดูลอัลตราโซนิก และแสดงผลออกทาง serial monitor และสามารถใช้ในการตรวจจับวัตถุ แจ้งเตือนเมื่อวัตถุเข้าใกล้เซ็นเซอร์มาก

โมดูลอัลตราโซนิก (Ultrasonic sensor)

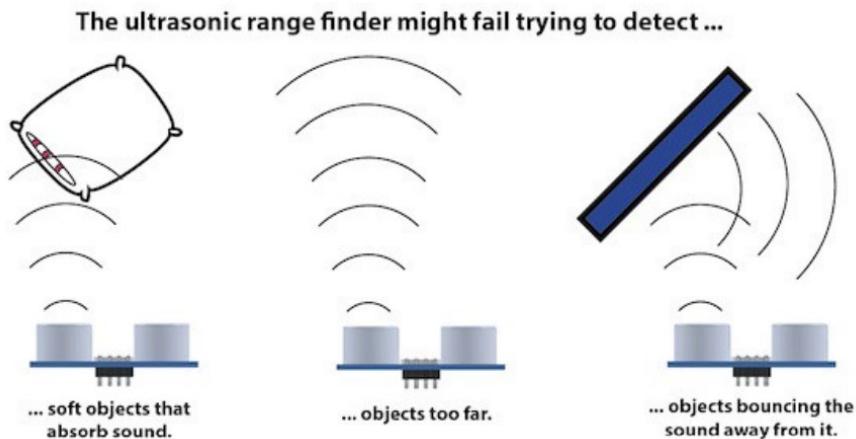
คือโมดูลที่ใช้คลื่นเสียงความถี่ในการส่ง และรับเพื่อระบุตำแหน่งระยะห่างของวัตถุนั้น ๆ โดยตัวส่งจะสร้าง คลื่นเสียงออกไป และเมื่อคลื่นกระแทกวัตถุ จะถูกสะท้อนมาให้กับตัวรับเพื่อนำไปประมวลผล โมดูล HC-SR04 วัดระยะห่างด้วยคลื่นอัลตราโซนิก (คลื่นเสียงความถี่ประมาณ 40 KHz) โดยคลื่นที่ส่งออกไปจะเป็นรูปของศากองแสง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



ถึงคลื่นที่ส่งออกไปจะมีลักษณะเป็นรูปบิม แต่ก็ใช้ว่าจะสามารถตรวจเช็ครอบทิศได้ เพราะมีองศาในการวัดเพียง 15 องศาเท่านั้น

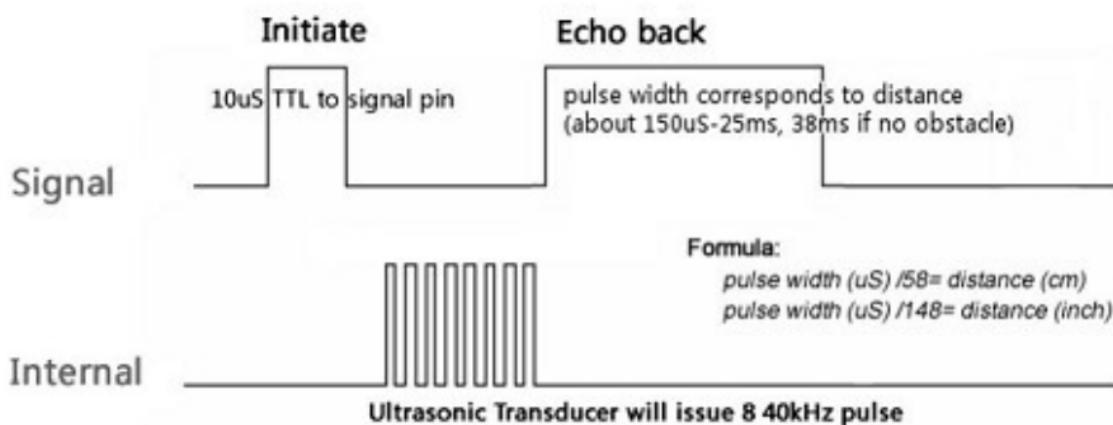


โดยโมดูล HC-SR04 มีขา TRIG (ตัวส่ง) และ ECHO (ตัวรับ) เพื่อส่งคลื่นอัลตราโซนิกในการวัดแต่ละครั้ง จะต้องสร้างสัญญาณพัลส์ (Pulse width) ที่มีความกว้างอย่างน้อย 10 ไมโครวินาที (10 microseconds) ป้อนเข้าขา Trig และวัดความกว้างของสัญญาณพัลส์ช่วงที่เป็น High จากขา Echo ประมาณ 150 ไมโครวินาที ถึง 25 มิลลิวินาที (150 microsecond - 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
- แรงดันเอาต์พุตอุจิสำหรับขา TRIG และ ECHO ประมาณ 5 V (TTL)

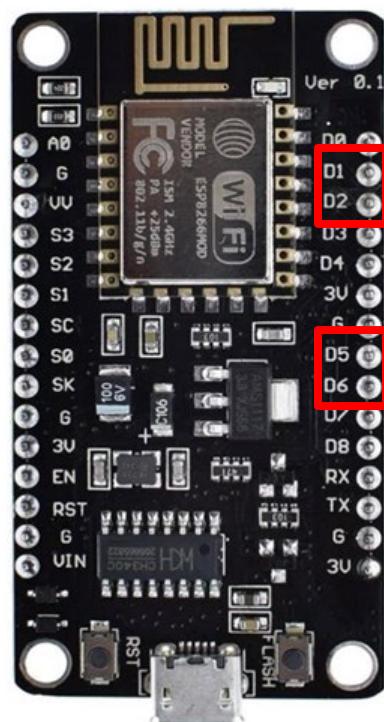
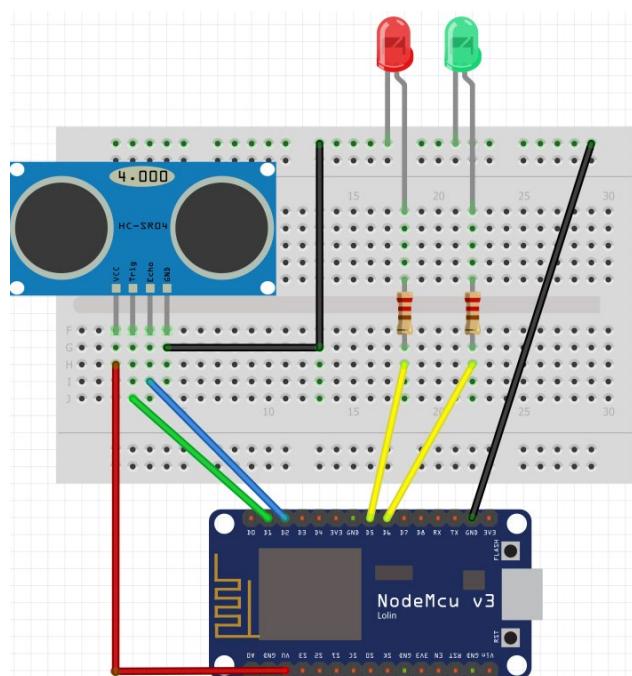


อุปกรณ์ที่ใช้

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 ให้เป็น High เป็นเวลา 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

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

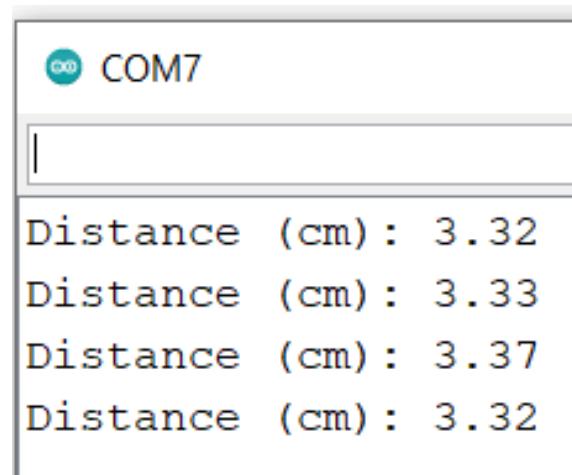
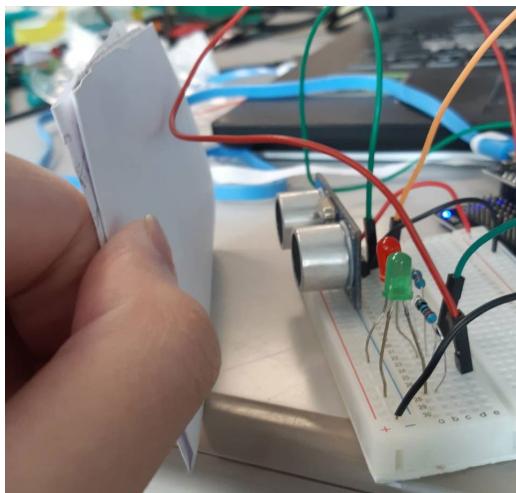
    // คำนวณระยะทาง
    distanceCm = duration * SOUND_VELOCITY/2;

    // 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);

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

    // ค่าหากระยะทาง
    distanceCm = duration * SOUND_VELOCITY/2;

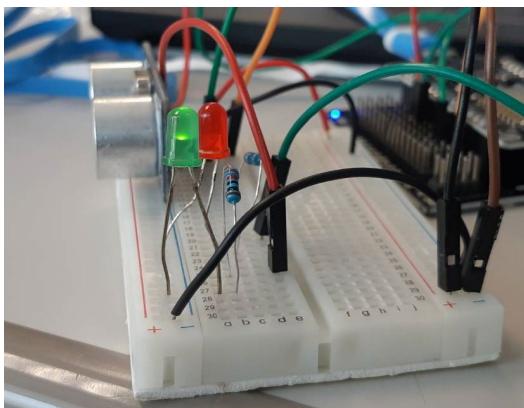
    if(distanceCm <= 10) // ใช้เกิดถ้ามีวัตถุเข้าใกล้
        digitalWrite(LED1, HIGH);
        digitalWrite(LED2, LOW);
    }
    else{
        digitalWrite(LED1, LOW);
        digitalWrite(LED2, HIGH);
    }
    delay(1000);
}

```

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

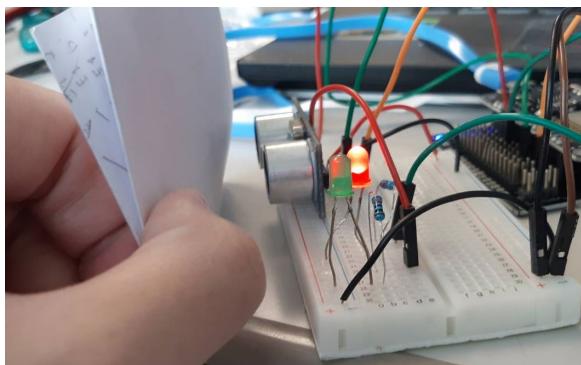
กรณีไม่มีวัตถุมาบัง sensor LED สีเขียวจะติดสว่างขึ้นมา



COM7

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

กรณีวัดถูม้าบัง sensor LED สีแดงจะติดสว่างขึ้นมา

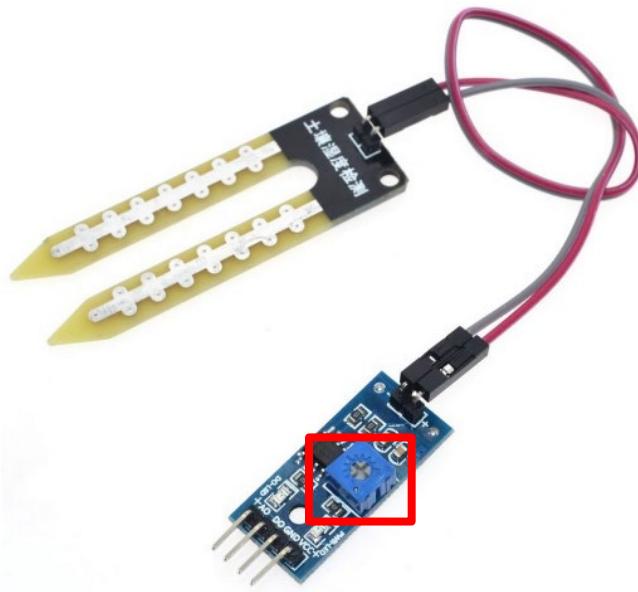


```
COM7
Distance (cm): 3.94
Distance (cm): 6.48
Distance (cm): 3.77
Distance (cm): 3.72
```

Workshop 6 วัดความชื้นในดินด้วย Soil Moisture Sensor

เช็ตเซอร์วัดความชื้นในดิน Soil Moisture Sensor

สามารถต่อใช้งานกับไมโครคอนโทรลเลอร์โดยใช้อนาล็อกอินพุตอ่านค่าความชื้น หรือเลือกใช้สัญญาณดิจิตอลที่ส่งมาจากโมดูล สามารถปรับความไวได้ด้วยการปรับ Trimpot



หลักการทำงาน

เซ็นเซอร์ความชื้นในดินส่วนใหญ่ได้รับการออกแบบมาเพื่อวัดการปริมาณน้ำที่อยู่ในดินที่แพรพันตามค่าคงที่โดยอิเล็กทริก ค่าคงที่โดยอิเล็กทริกเป็นความสามารถของดินในการเป็นตัวกลางกระแสไฟฟ้า เนื่องจากน้ำเป็นตัวกลางที่ดีของกระแสไฟฟ้า ดังนั้นค่าคงที่โดยอิเล็กทริกของดินจะเพิ่มขึ้นเมื่อปริมาณน้ำในดินเพิ่มขึ้นนั่นเอง ด้วยสมมติฐานที่ว่าค่าคงที่โดยอิเล็กทริกของน้ำเป็นตัวกลางที่สะอาดในการผ่าน กระแสไฟฟ้าเหนือการวิ่งผ่านดินและอากาศในดิน ดังนั้นการวัดค่าคงที่โดยอิเล็กทริกทำให้สามารถประมาณ ค่าความชื้นได้

ตัวเซ็นเซอร์จะมี 2 ส่วน ส่วนแรกเป็นเซ็นเซอร์ร่างกายแต่จะต้องอิเล็กโ tööd ที่ทำหน้าที่รับส่งค่ากระแสไฟฟ้าที่วิ่ง จากขาหนึ่งไปอีกขาหนึ่งโดยมีดินเป็นตัวกลาง ส่วนที่สองเป็นตัวขยายสัญญาณจากเซ็นเซอร์

หลักการวัดค่าของเซนเซอร์

ในอุปกรณ์เซนเซอร์จะมีอีซิอปแอมป์ LM393 เพื่อวัดแรงดันเปรียบเทียบกันระหว่างแรงดันที่วัดได้จากความชื้นในดิน กับแรงดันที่วัดได้จากการจราบแบ่งแรงดันปรับค่าโดยใช้ trimpot หากแรงดันที่วัดได้จากความชื้นของดิน มีมากกว่า ก็จะทำให้วงจรปล่อยลอกิจ 1 ไปที่ขา D0 แต่หากความชื้นในดินมีน้อย ลอกิจ 0 จะถูกปล่อยไปที่ขา D0

ขา A0 เป็นขาที่ต่อโดยตรงกับวงจรที่ใช้วงความชื้นในดิน ซึ่งให้ค่าแรงดันออกมาตั้งแต่ 0 - 5V (ในทางอุดมคติ) โดยหากความชื้นในดินมาก แรงดันที่ปล่อยออกไปก็จะน้อย ในลักษณะของการแพร่ภาคัน

การนำไปใช้งาน

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

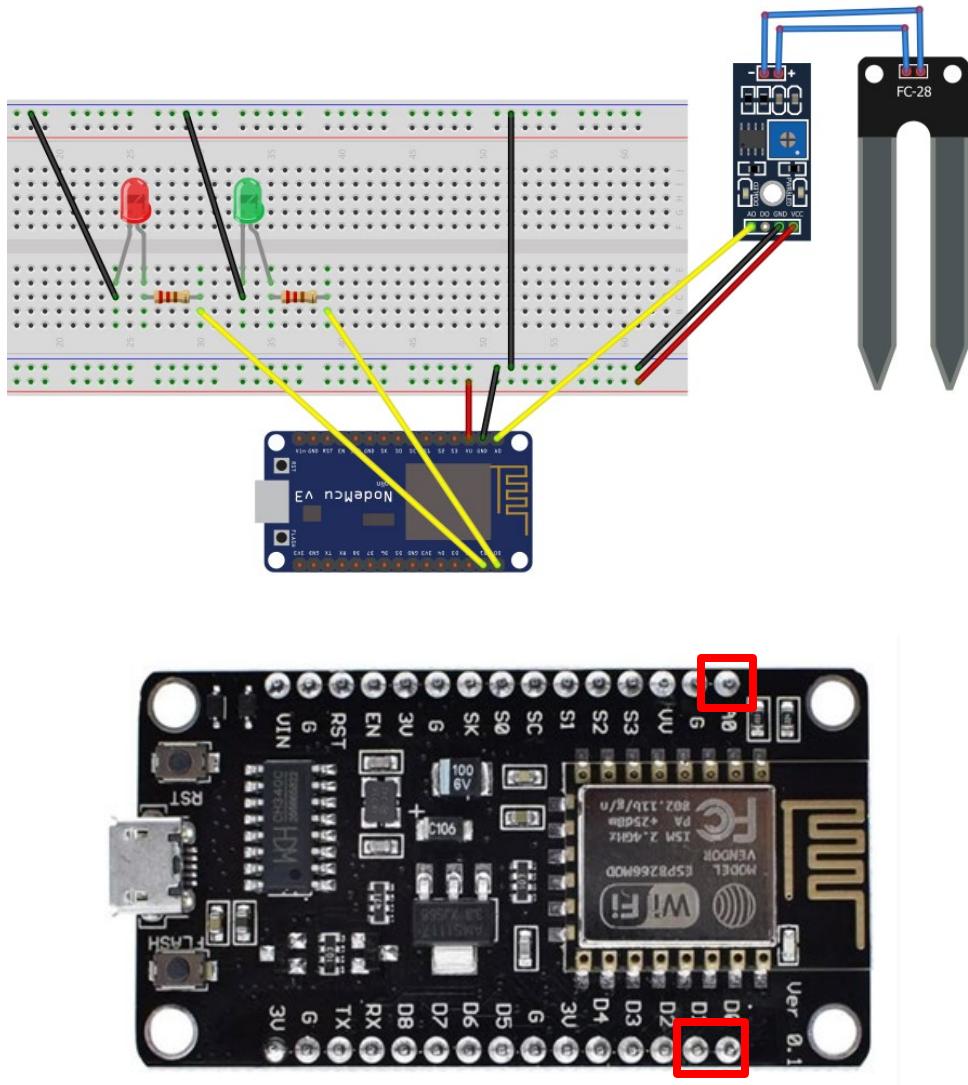
หากต้องการนำไปใช้ในโปรเจคที่ไม่ต้องใช้วัดละเอียด สามารถนำขา D0 ต่อเข้ากับทรานซิสเตอร์กำลังเพื่อส่งให้มีน้ำ หรือโซลินอยด์ทำงานเพื่อให้มีน้ำไหลมารดตันไม่ได้เลย เมื่อความชื้นในดินมีมากพอ จะปล่อยลอกิจ 0 และทรานซิสเตอร์จะหยุดนำกระแส ทำให้มีน้ำหยุดปล่อยน้ำ

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

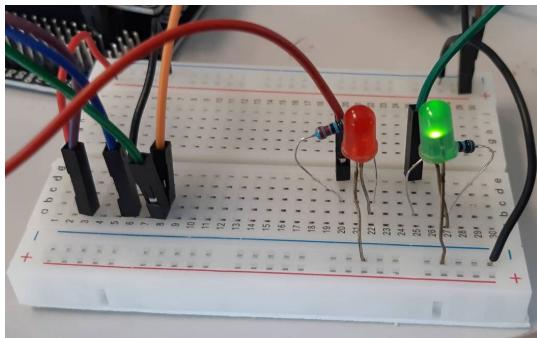
    if (map_val > 50) { //หากค่าที่ทำการแปลง map_val มีค่ามากกว่า 50 แสดงว่าคิดเมื่อความชื้นเกิน 50 เมตรเข้าไป
        digitalWrite(led_G , LOW); // สั่งให้ LED เขียวดับ
        digitalWrite(led_R, HIGH); // สั่งให้ LED สีแดง ติดสว่าง
    }
    else {
        digitalWrite(led_G , HIGH); // สั่งให้ LED เขียวติดสว่าง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

```

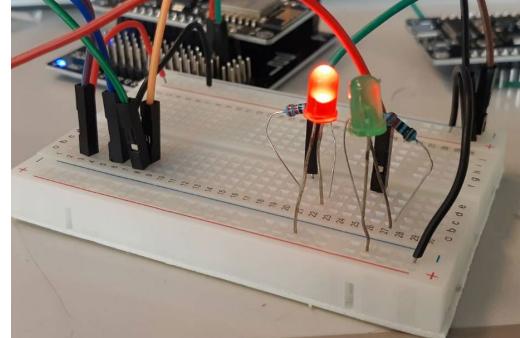
สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_6/Soil_Moisture_Sensor_LED/Soil_Moisture_Sensor_LED.ino

ผลลัพธ์ที่ได้

ค่าความชื้นในดิน กรณี แห้ง LED สีเขียวติด



ค่าความชื้นในดิน กรณี เปียก LED สีแดงติด



Workshop 7 วัดอุณหภูมิโดยใช้ DS18B20 และ แจ้งเตือนด้วยเสียงเตือน

DS18B20 คือ โมดูลเซนเซอร์อุณหภูมิแบบกันน้ำ วัดได้ทั้งอุณหภูมิห้องและแบบบีส์ Probe เพื่อวัดของเหลวในท่อส่งหรือแบบจุ่มไปในของเหลวได้ ใช้ IC เบอร์ DS18B20 ผลิตโดย Dallas Semiconductor Corp. สายยาว 100 เซนติเมตร (ในท้องตลาดมีรุ่นสายยาว 2-3 เมตร) ต้องต่อตัว ต้านทาน 4.7K โอห์มร่วมด้วยใช้งานง่ายและสามารถใช้งานกับ NodeMCU ESP8266 ได้ โดยการรับส่ง ข้อมูล นั้นจะใช้สายสัญญาณเส้นเดียวกันและเป็นสัญญาณแบบดิจิตอล ซึ่งเมื่อนำมาโคลค่อนโทรลเลอร์มา ใช้ต่อกับเซนเซอร์ตรวจวัดอุณหภูมิก็สามารถเขียนคำสั่งให้มีอุณหภูมิสูงหรือต่ำกว่าค่าที่กำหนด 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

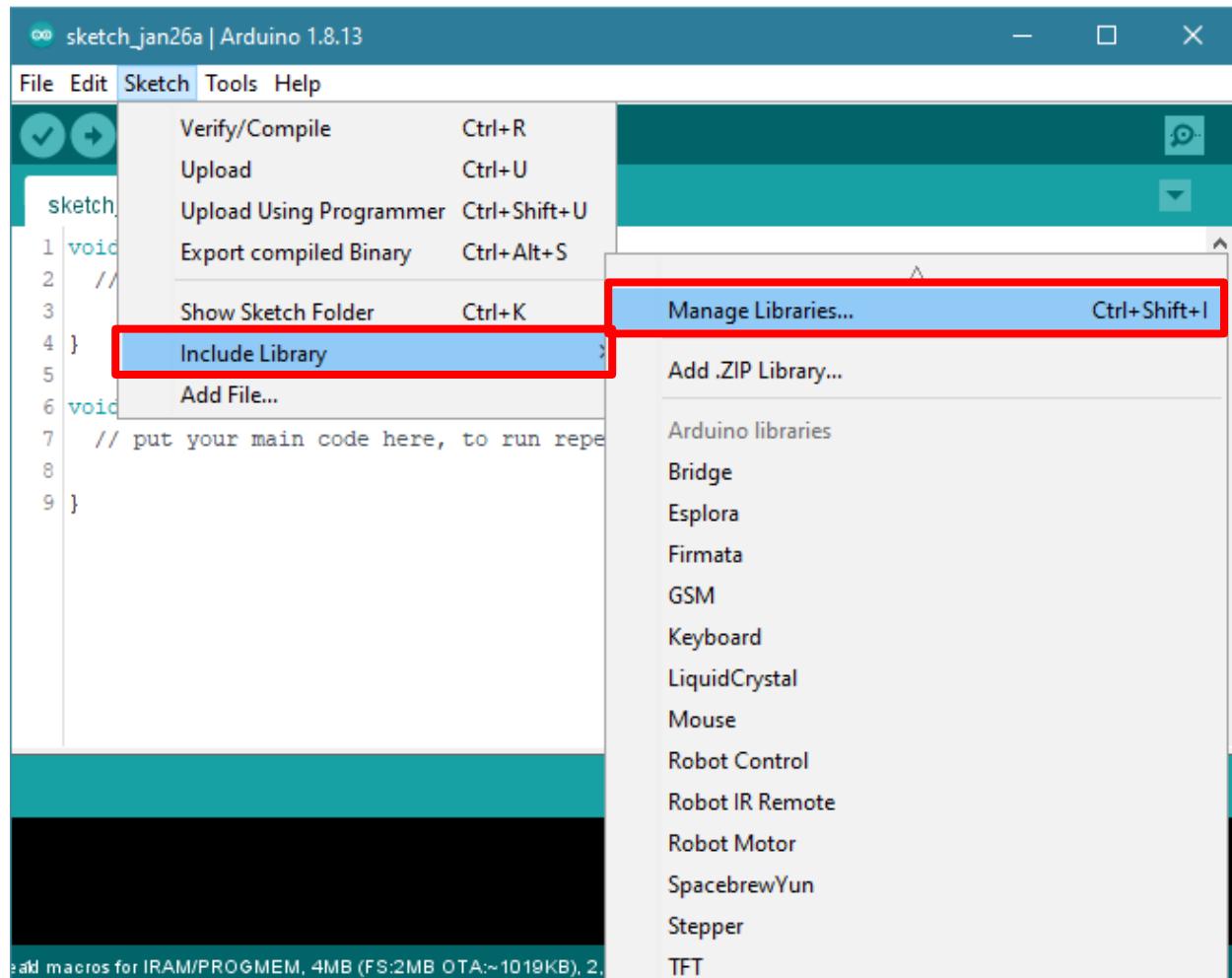
อุปกรณ์ที่ใช้

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

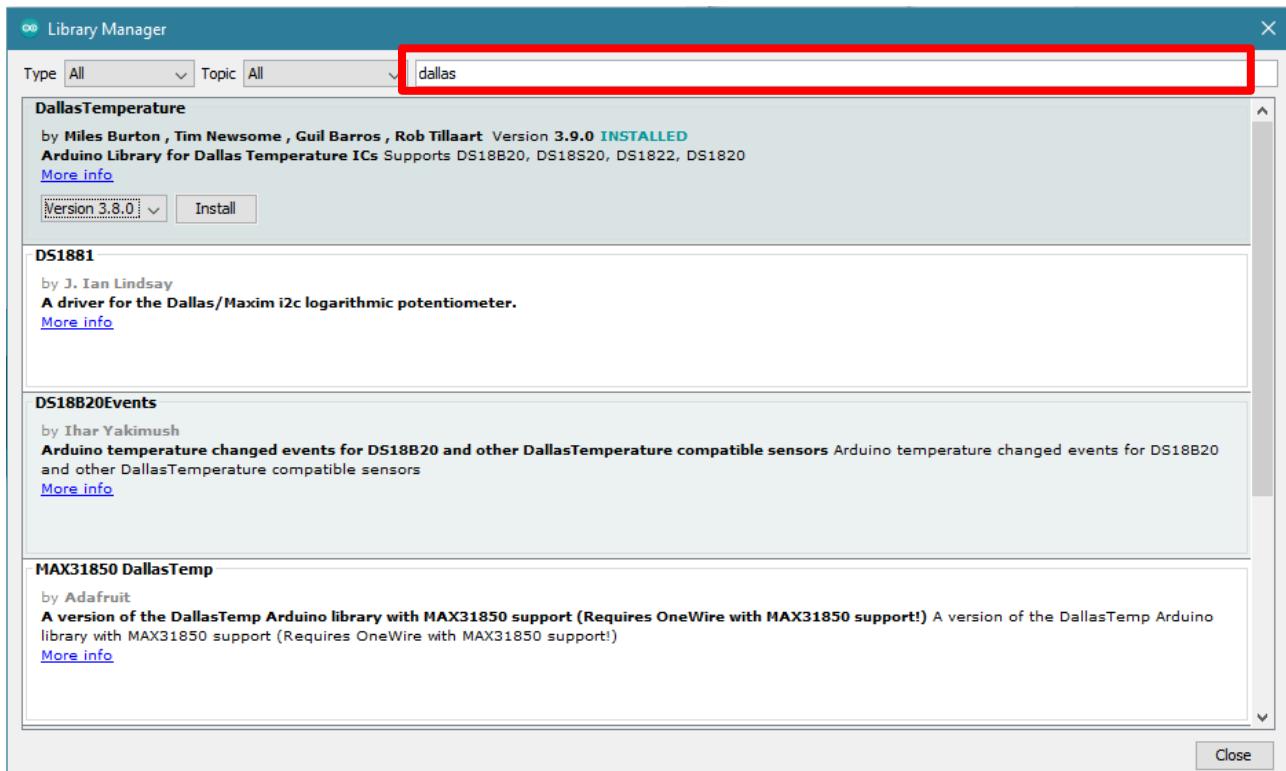
วิธีการติดตั้งโมดูล

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

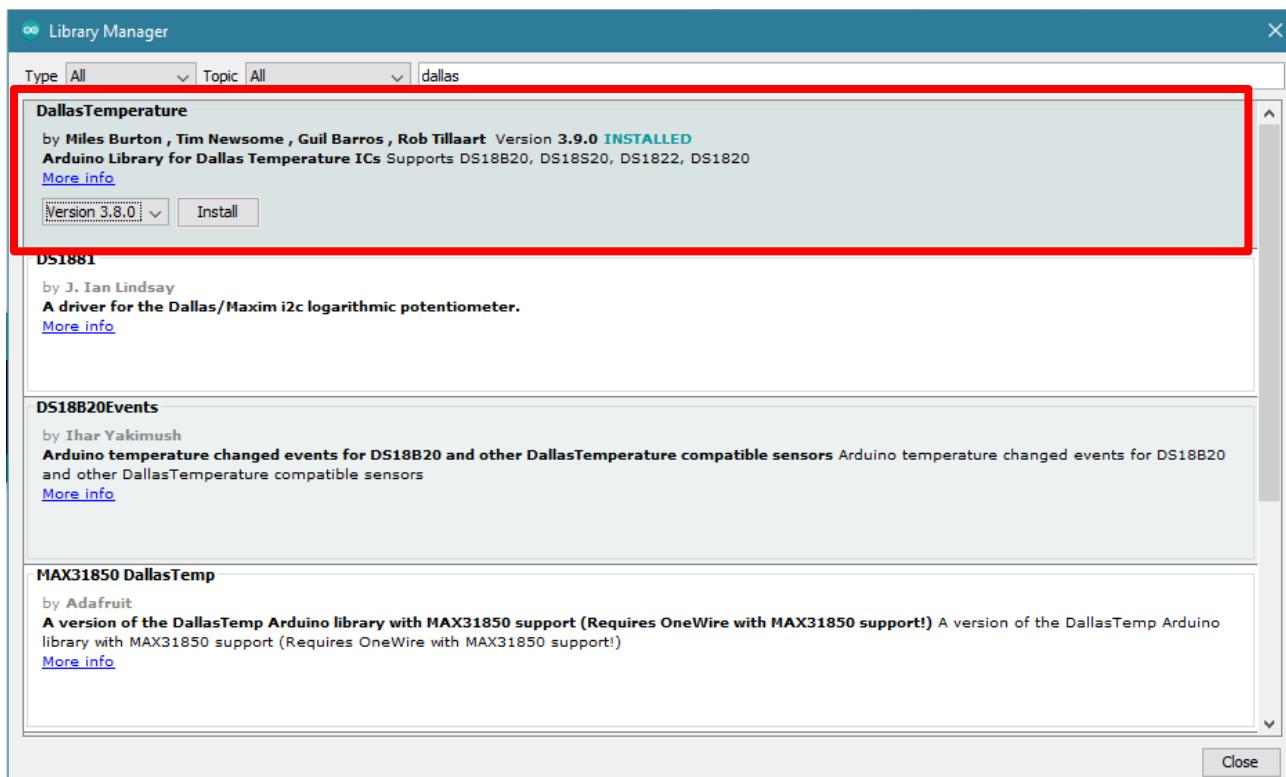
1. ใน การติดตั้ง Library ให้ไปที่ Sketch -> Include Library -> Manage Libraries



2. ทำการพิมพ์ keyword ชื่อ dallas



3. กดเลือกเวอร์ชันแล้วกด Install

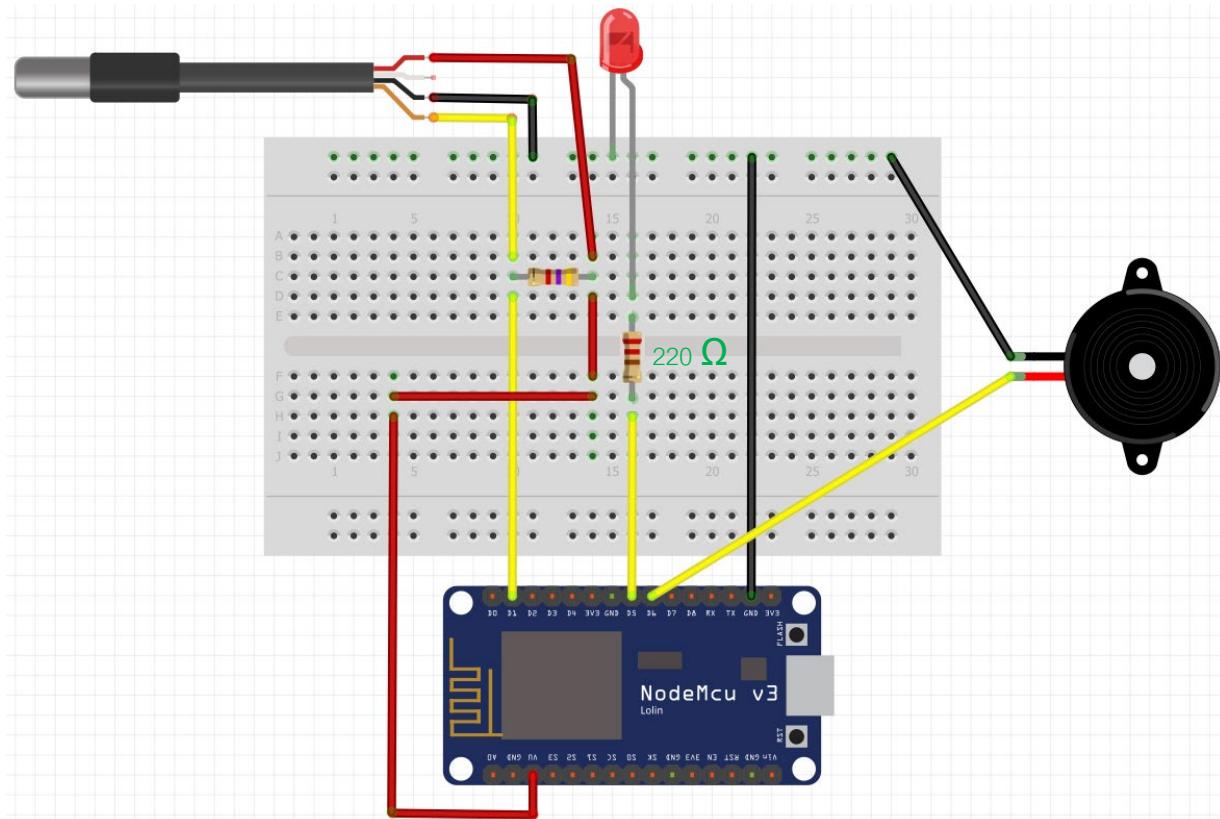


4 ทำการพิมพ์ keyword ชื่อ Onewire และเลือก Install ดังรูป



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

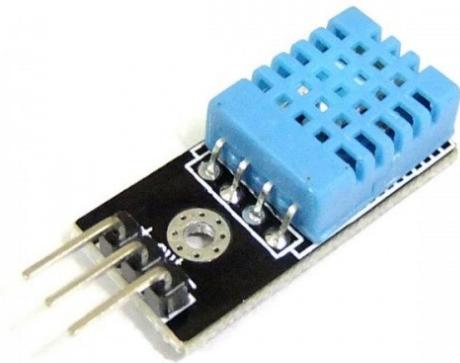
กรณีถ้า sensor ตรวจจับอุณหภูมิ ตรวจได้อุณหภูมินิมากกว่า 32 องศาตามที่เขียนไว้ในเงื่อนไข ก็จะทำให้ LED ติดและ Buzzer ส่งเสียง

```
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

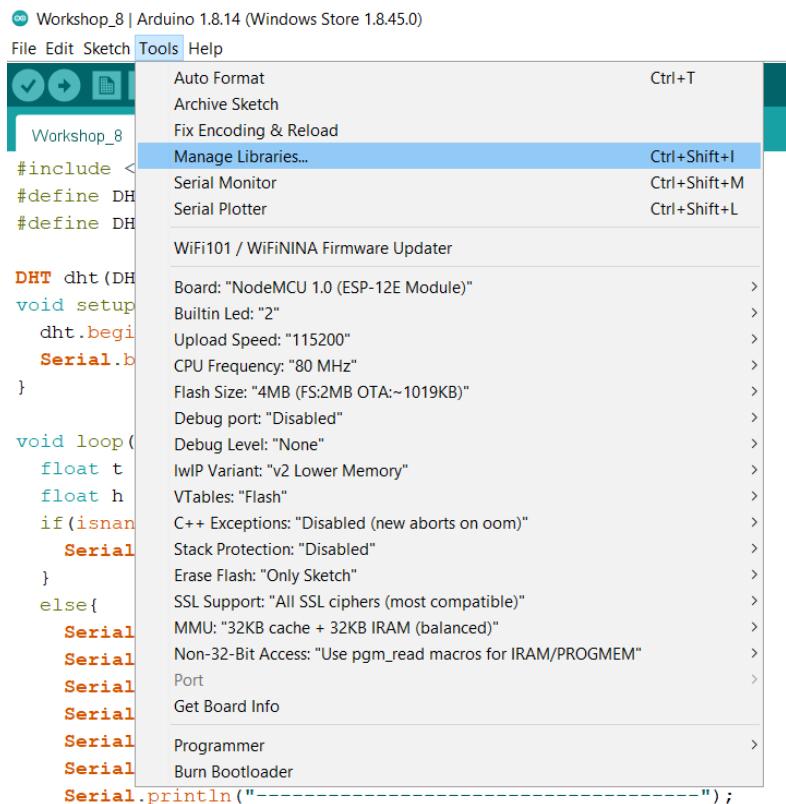
DHT22 เป็น sensor ที่ใช้ในการวัดอุณหภูมิและความชื้นในอากาศ ซึ่งออกแบบมาให้วัดได้แม่นยำกว่ารุ่น DHT11 แต่การเขียน code จะเขียนเหมือนกัน ตั้งนั้นเราสามารถนำ DHT22 ไปเปลี่ยนใช้งานแทน DHT11 ได้

ข้อมูลของ DHT22

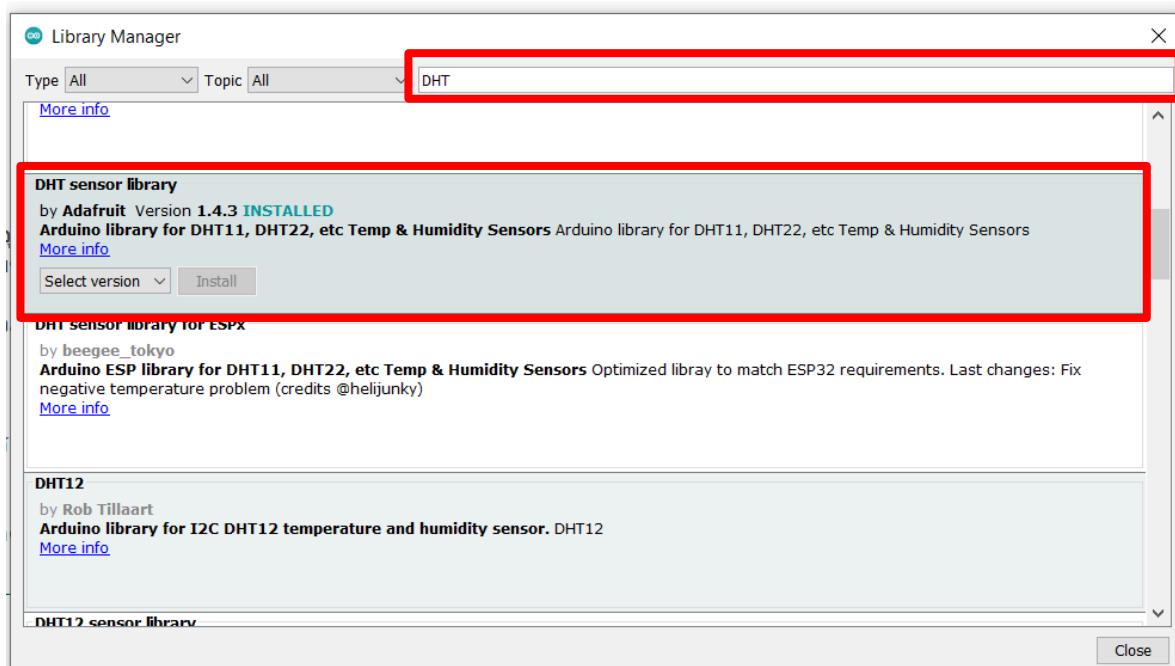
1. Power supply : 3 - 5.5V
2. วัดอุณหภูมิได้ระหว่าง 0 - 50 องศาเซลเซียส +/- 2 องศา
3. วัดความชื้นในอากาศได้ระหว่าง 20 - 90 % +/- 5%
4. เวลาที่ใช้ในการวัดค่า : 1 วินาที

ขั้นตอนการติดตั้งไลบรารี

1. กดเลือก Tool > Manage Libraries



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

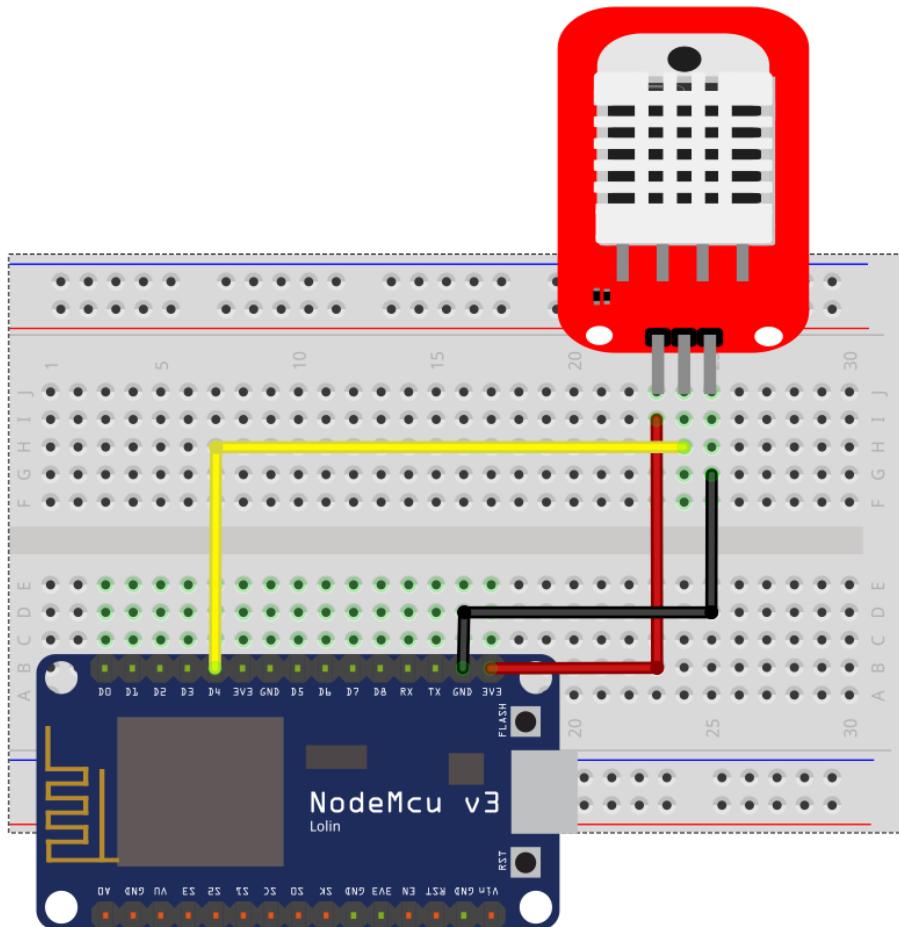


อุปกรณ์ที่ใช้

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)) ////ใช้ในการเช็คค่าจาก 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
}
```

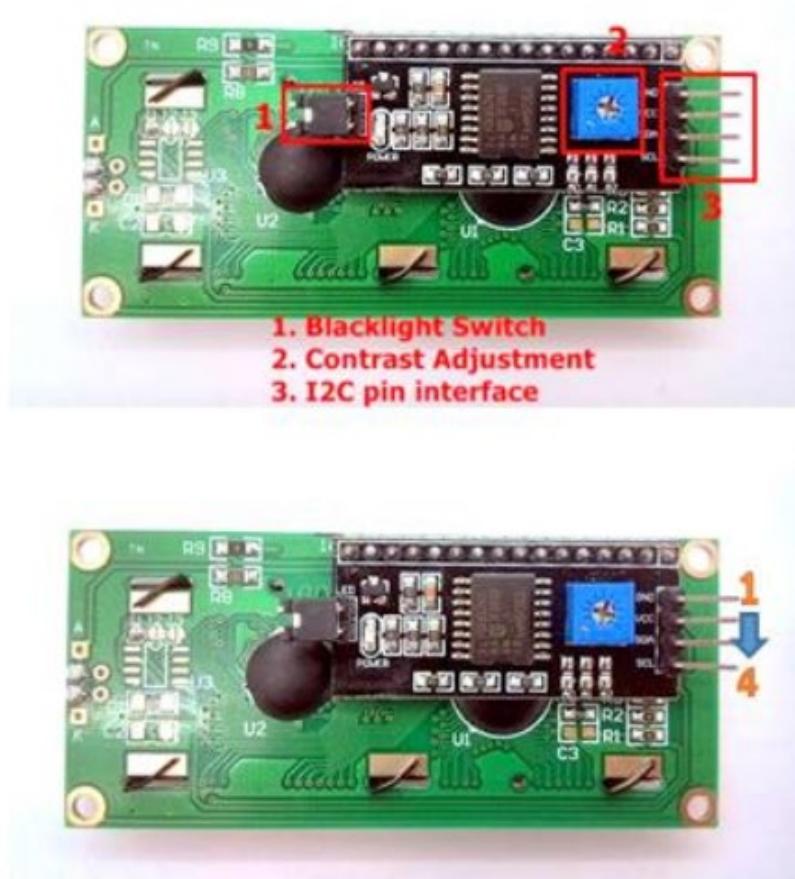
สามารถเข้าไปดูโค้ดได้ที่ https://github.com/zKim1379z/Workshop-Seminar-IoT-2022/blob/main/Workshop-Seminar-IoT-2022/Workshop_8/Workshop_8.ino

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

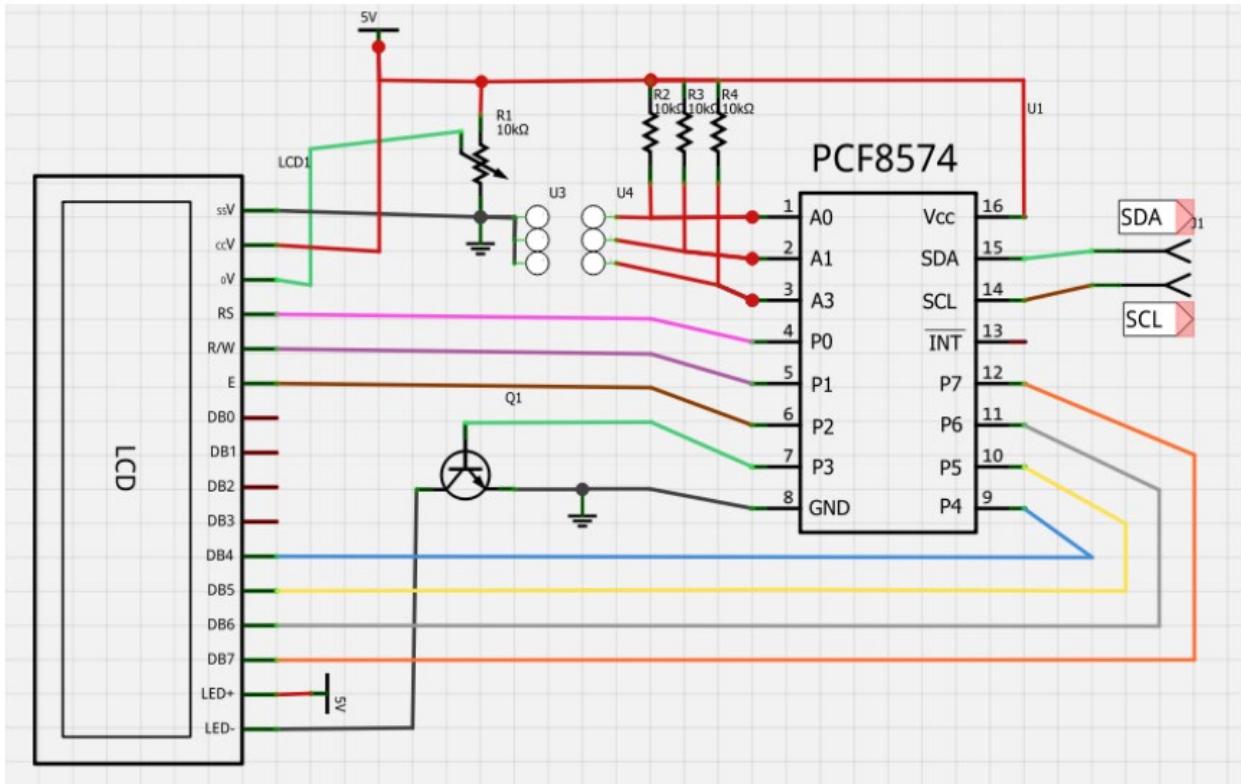
จอ LCD ที่มีการเชื่อมแบบ I2C หรือเรียกอีกอย่างว่าการเชื่อมต่อแบบ Serial เป็นจอ LCD ธรรมด้าทั่วไปที่มาพร้อมกับบอร์ด I2C Bus ที่ทำให้การใช้งานได้สะดวกยิ่งขึ้นและมาพร้อมกับ VR สำหรับปรับความเข้มของจอในรูปแบบ I2C ใช้ขาในการเชื่อมต่อกับ Arduino เพียง 4 ขา (แบบ Parallel ใช้ 16 ขา) ซึ่งทำให้ใช้งานได้ง่ายและสะดวกมากยิ่งขึ้น



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 โดยที่รหัสคำสั่งที่ใช้ในการสั่งงานจอ LCD ยังคงไม่ต่างกับจอ LCD ที่เป็นแบบ Parallel โดยส่วนใหญ่บอร์ด I2C จะเชื่อมต่อกับตัวควบคุมของจอ LCD เพียง 4 บิตเท่านั้น วิธีการเชื่อมต่อได้ดังนี้



รูปการเชื่อมต่อระหว่าง Arduino กับ LCD (I2C)

(ที่มา www.loxhop.com)

จากรูป วงจรจอ LCD และบอร์ด I2C ได้มีการเชื่อมต่อขาสำหรับการรับส่งข้อมูลเป็นแบบ 4 บิต อาทิ เชื่อมต่อไว้คือ ขา P4 > DB4, P5 > DB5, P6 > DB6, P7 > DB7 และขา P2 > E (Enable), P1 > R/W, P0 > RS รวมไปถึงตัวต้านทานสำหรับปรับค่าความเข้มของตัวอักษร และ Switch Blacklight จากการจริงๆที่จำเป็นในการใช้งานถูกเชื่อมต่อเข้ากับตัวบอร์ด 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 เอียนเป็น Library มาให้เพื่อสะดวกในการนำไปใช้งาน หลังจากต่อสายเสร็จเรียบร้อย ขั้นตอนแรกในการเริ่มเขียนโปรแกรมคือการเรียกใช้ Library ของ LCD จากไฟล์ชื่อ LiquidCrystal.h

ฟังก์ชัน LiquidCrystal(); ใช้ประกาศขาที่ต้องการส่งข้อมูลไปยังจอ LCD รูปแบบในการสั่งงานคือ

- LiquidCrystal lcd(rs, enable, d4, d5, d6, d7) <<< ในกรณีใช้งานแบบ 4 บิต
- LiquidCrystal lcd(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7) <<< ในกรณีใช้งานแบบ 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

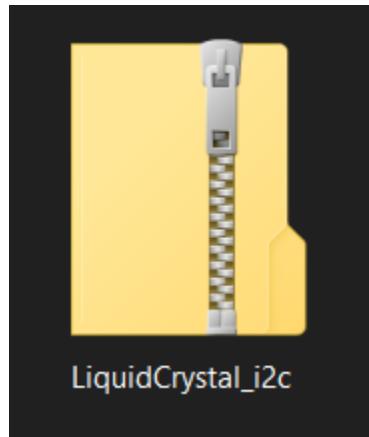
ฟังก์ชัน begin(); ใช้กำหนดขนาดของจอ ใช้ขนาด 16 ตัวอักษร 2 บรรทัด จึงประกาศเป็น lcd.begin(16, 2);

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

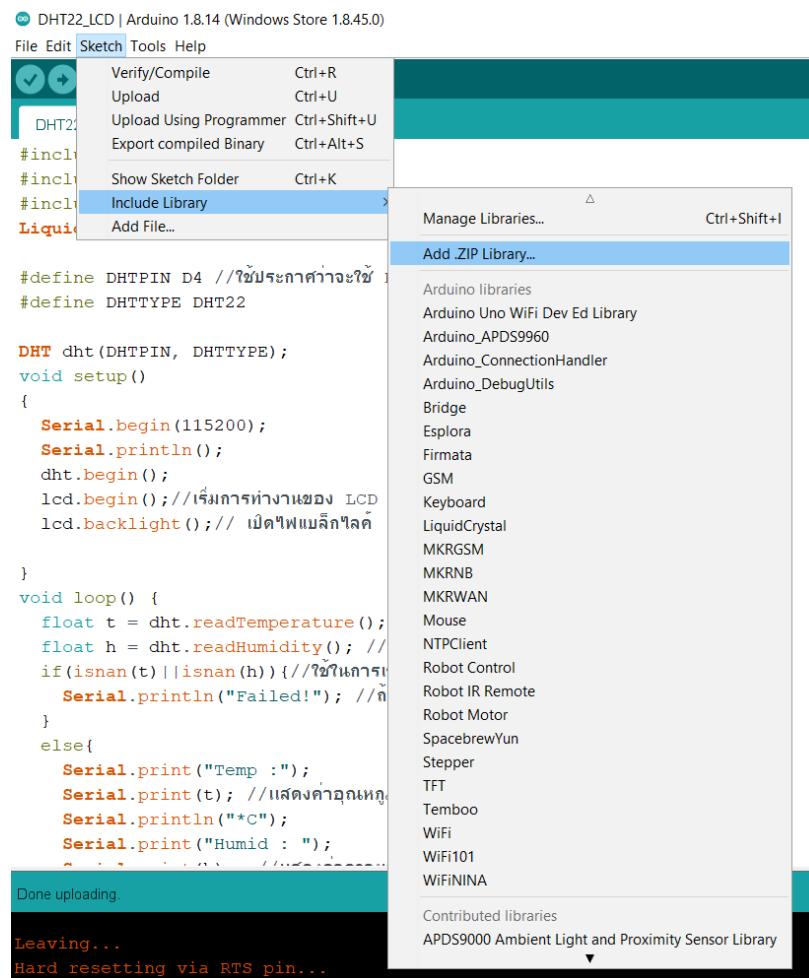
ฟังก์ชัน print(); ใช้กำหนดข้อความที่ต้องการแสดง เช่น lcd.print("TEST"); คือให้แสดงข้อความ “TEST” ออกทางหน้าจอ LCD

ขั้นตอนการติดตั้งไลบรารีจอแสดงผล LCD

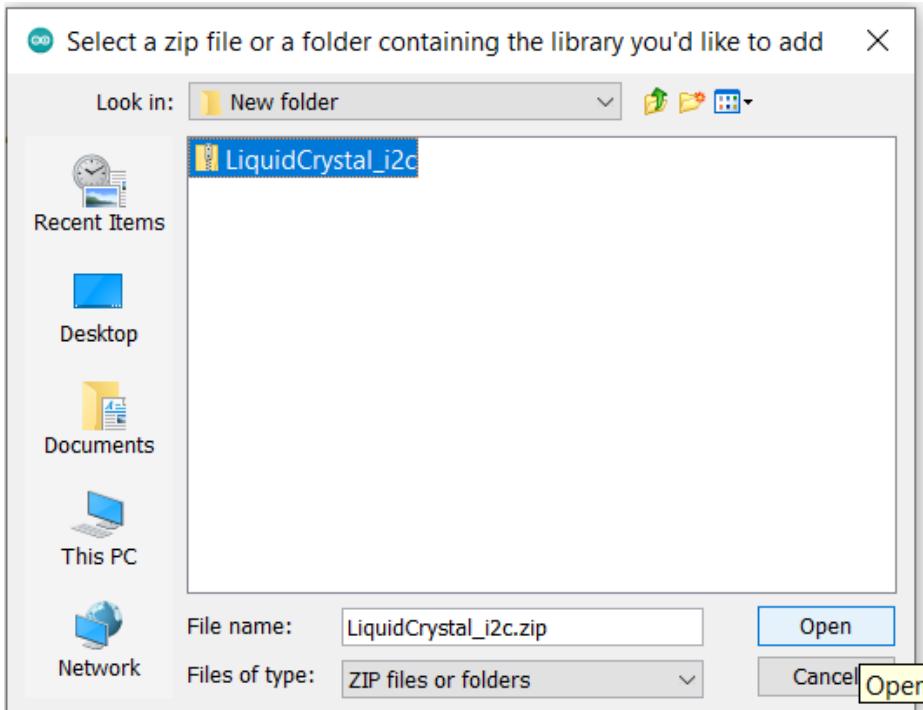
1. ดาวน์โหลดไฟล์ Zip LiquidCrystal_i2c



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



3. ไปที่ไฟล์เดอร์ที่เราดาวน์โหลดไฟล์ 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); หมายถึงเข็ตเคอร์เซอร์ไปตัวอักษรที่ 2 นับจากทางซ้ายและอยู่บรรทัดแรก เมื่อใช้คำสั่ง lcd.print() ตัวอักษรตัวแรกจะอยู่ลำดับที่ 3 นับจากทางซ้าย
lcd.write(ข้อมูลที่ต้องการเขียนออกไป)	ใช้สำหรับเขียนข้อมูลออกไปทีลีตัวอักษร

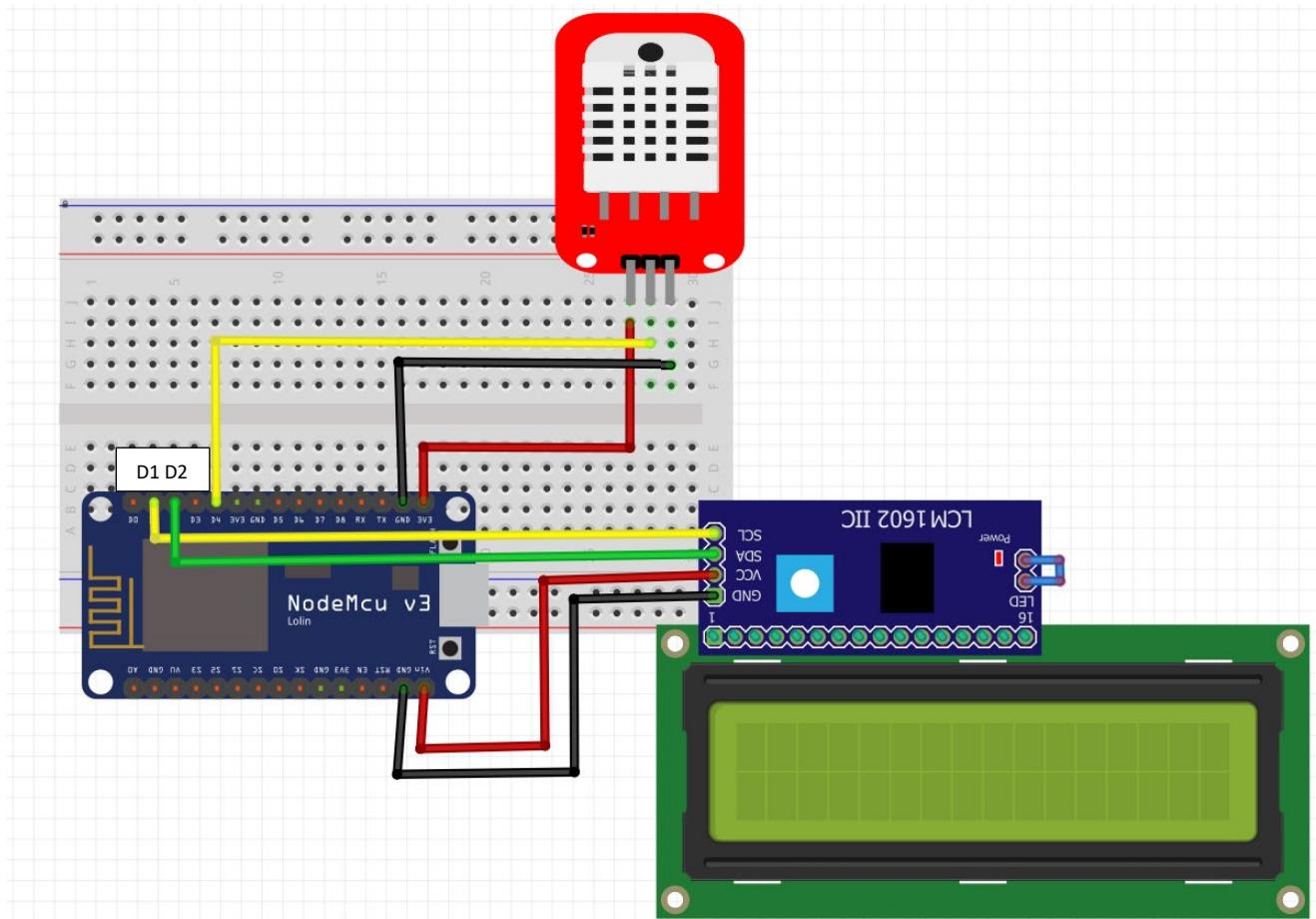
คำสั่ง	การทำงาน
lcd.print(x , y)	x คือ ข้อมูลที่ต้องการให้เขียนออกไป y คือ รูปแบบข้อมูล ใช้เขียนข้อมูลออกไปทั้งข้อความ
lcd.cursor()	ใช้สั่งให้แสดงเครื่องเขียนบนหน้าจอ
lcd.noCursor()	ใช้สั่งให้ไม่แสดงเครื่องเขียนบนหน้าจอ
lcd.display()	แสดงตัวอักษรบนหน้าจอ
lcd.noDisplay()	ปิดการแสดงตัวอักษรในหน้าจอ
lcd.scrollDisplayLeft()	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(); // เปิดไฟแบนล็อกไลด์

}

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



* ในกรณีที่หน้าจอ 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. ให้ไปที่เว็บไซต์ blynk.io
2. กดปุ่ม Sign Up Free เพื่อทำการสมัคร

blynk.io

Blynk

Log In Contact Sales Sign Up

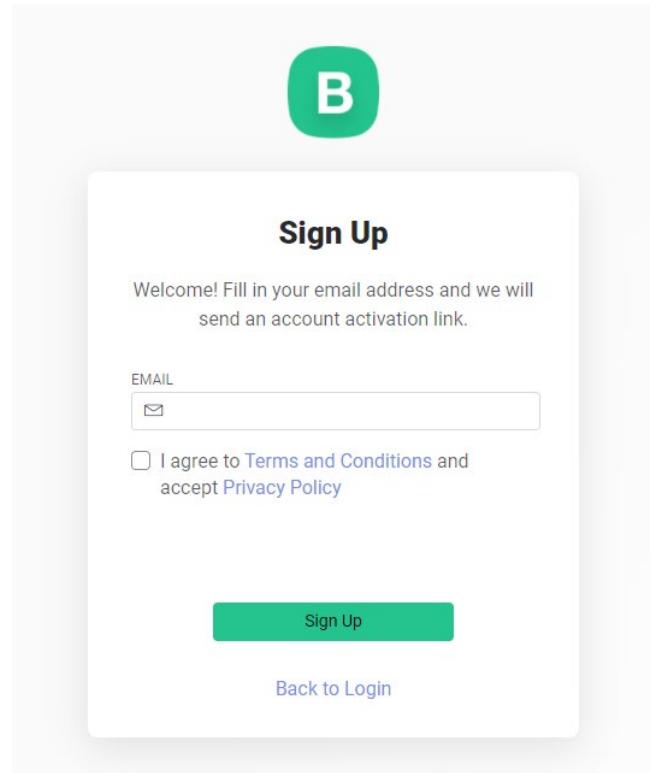
Low-code IoT cloud platform with user experience at its core

Easily build exceptional, fully customizable mobile and web IoT applications. Securely deploy and manage millions of devices worldwide.

Enterprise Solutions **Sign Up Free →**

★★★★★ G2 4.6/5 stars

3. ทำการสมัครโดยใช้ Email และทำการ log in



เมื่อ log in เรียบร้อยแล้วจะอยู่ในหน้านี้

My organization - 5265TL |

Blynk.Console

Developer Zone >

- Devices
- Users
- Organizations
- Locations

My Templates

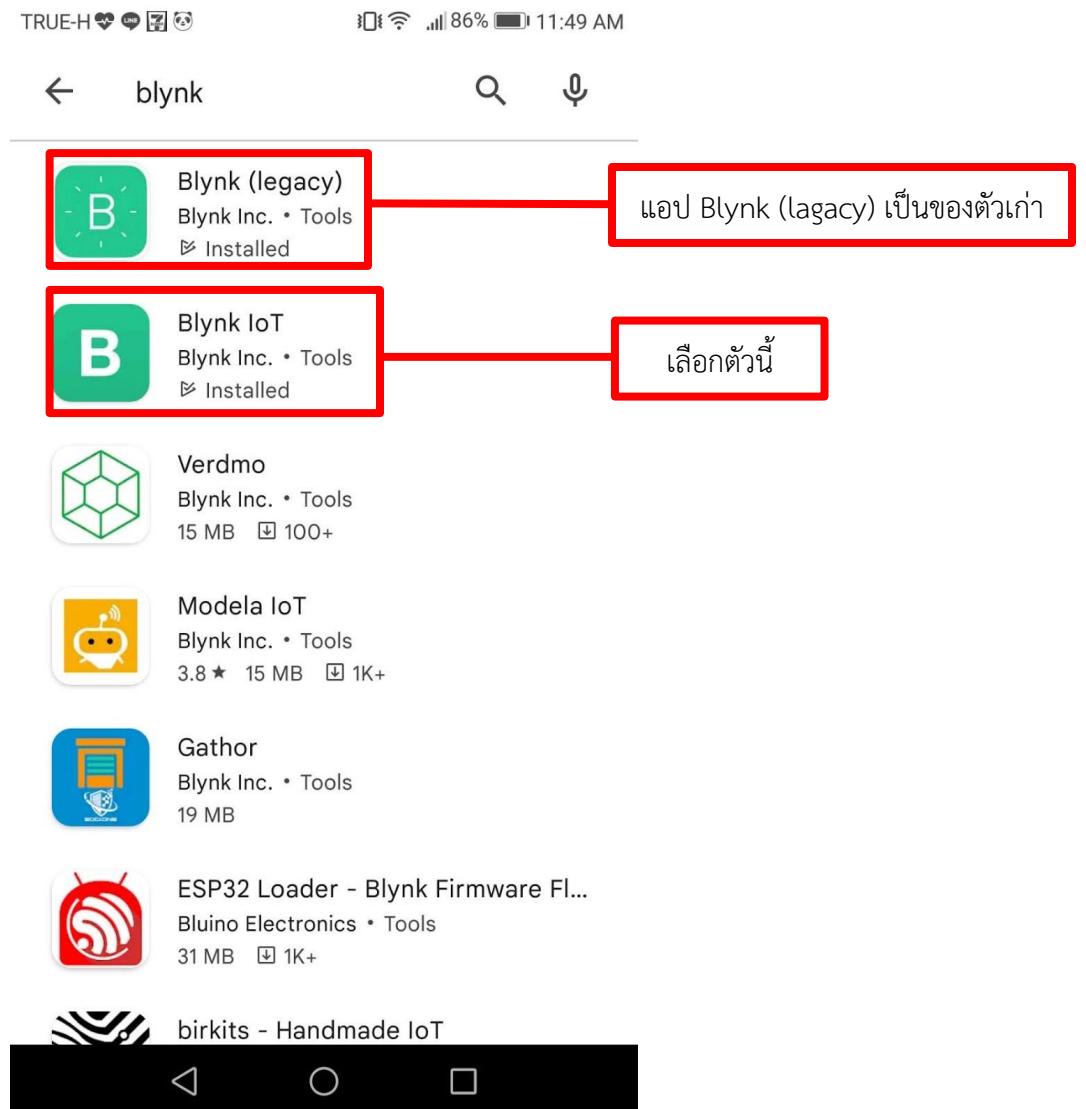
- Blueprints
- Blynk.Air (OTA)
- Webhooks
- Integrations

Start by creating your first template

Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.

+ New Template

4. ในส่วนของ Application บน Smartphone ให้ค้นหาแอปพลิเคชันชื่อ Blynk IoT



5. ให้ทำการ log in เข้าระบบ



Sign Up

Log In

กดตรงนี่



การสร้าง Project บน Web

1. ให้กดเลือกที่หัวข้อ New Template

The screenshot shows the Blynk Console interface. On the left, there's a sidebar with 'Developer Zone' selected. The main area is titled 'DEVELOPER ZONE' and contains a section for 'My Templates'. This section includes links for 'Blueprints', 'Blynk.Air (OTA)', 'Webhooks', and 'Integrations'. To the right, there's a large call-to-action button labeled '+ New Template' with a red border. Above this button, text reads 'Start by creating your first template'. Below the button, a descriptive text states: 'Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.'

2. 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

3. กด save

Blynk.Console

Developer Zone

TEST BLYNK

Home

What's next?

- Configure template
- Set Up Datastreams
- Set up the Web Dashboard
- Add first Device

Template settings

ESP8266, WiFi

Firmware configuration

Template ID and Template Name should be declared at the very top of the firmware code.

```
#define BLYNK_TEMPLATE_ID "TMPL6VhLM-3ng"
#define BLYNK_TEMPLATE_NAME "Test Blynk"
```

4. กลับไปที่ Tab Device เพื่อเพิ่ม Device

Blynk.Console

My organization - 5265TL |

Developer Zone >

Devices (highlighted with a red box)

Users

Organizations

Locations

All of your devices will be here.

You can activate new devices by using your app for iOS or Android

Download for iOS Download for Android

+ New Device (highlighted with a green box)

กด New Device

New Device

Choose a way to create new device

เลือก From template

From template (highlighted with a red box)

Scan QR code

Manual entry

New Device

เลือก Template ที่เราสร้างไว้

Create new device by filling in the form below

TEMPLATE

Test Blynk

Test Blynk

Point on the cards to see instruct

Cancel

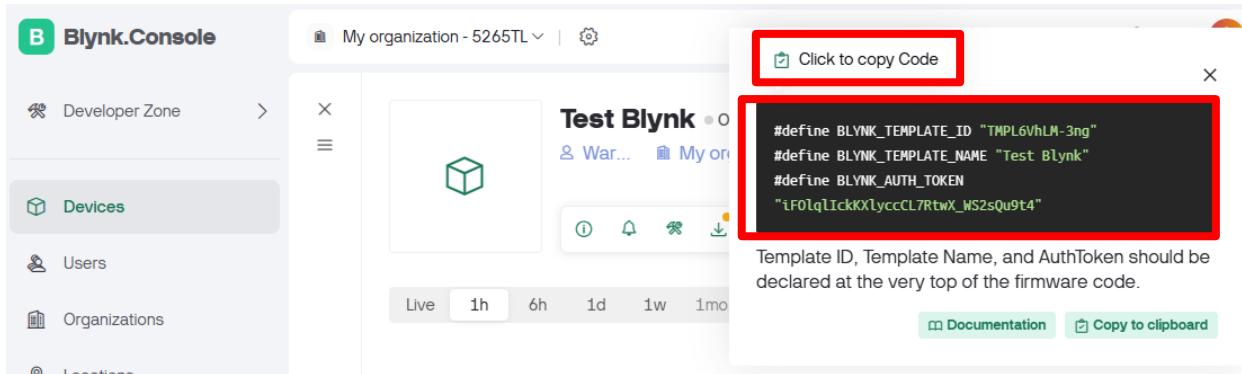
Create

กด Create เพื่อสร้าง

6. ในส่วนของ Mobile Application หากเราได้ Device แล้วหน้าจะจะปรากฏตามดังรูป



7. กลับมาที่ฝั่งคอมพิวเตอร์ ให้เลือก Device ที่เราสร้างขึ้นมาแล้วกดไปที่ Device Info



ให้ copy ส่วนนี้มาเขียนลงใน Arduino IDE

Code ภายใน Arduino IDE

```
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>

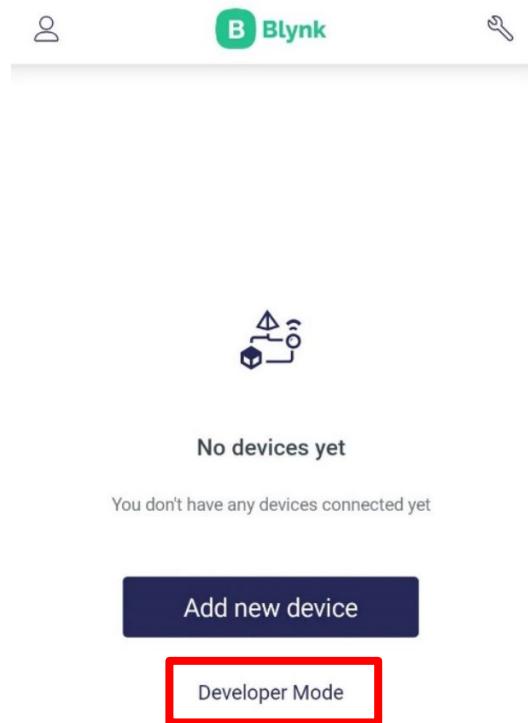
#define LED1 D5
#define LED2 D6
#define LED3 D7

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

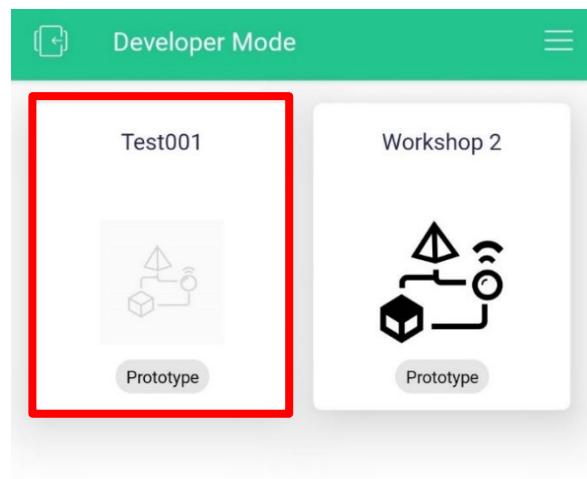
BlynkTimer timer;
void timerEvent();
```

การจัดหน้า Dashboard บน Mobile Application

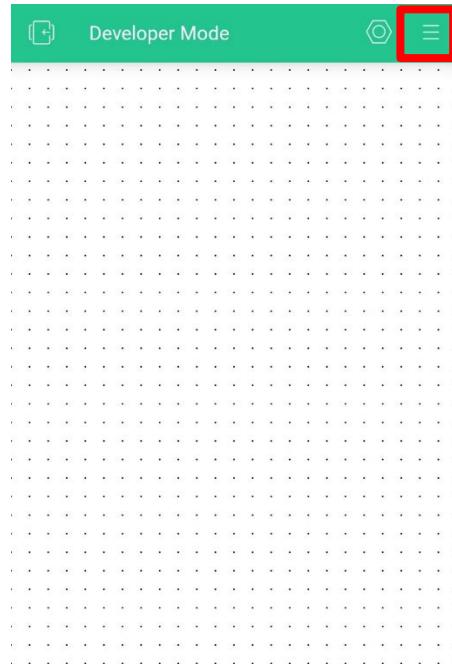
- กดเข้า Developer Mode เพื่อเลือก template



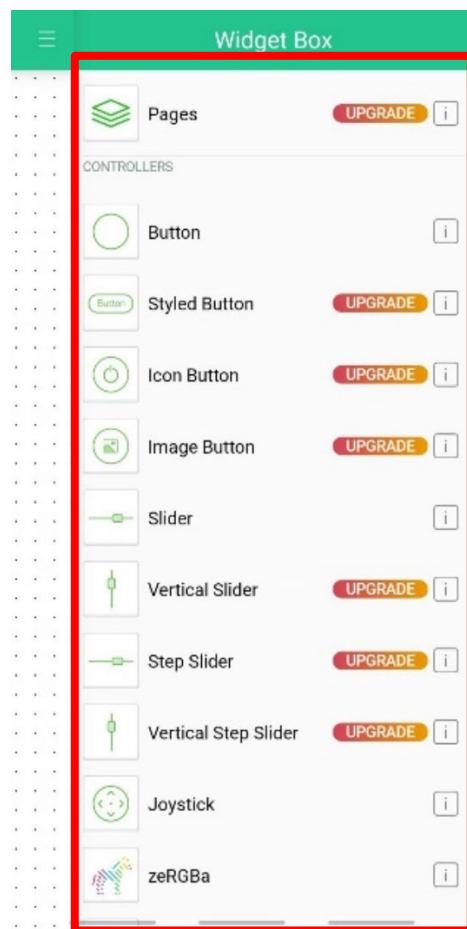
- เลือก Template ที่เราต้องการ



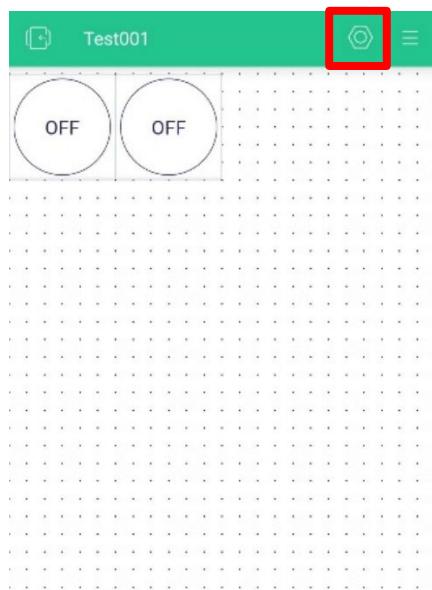
3. กดเพื่อเลือก widget ที่ต้องการในการออกแบบหน้า Dashboard



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

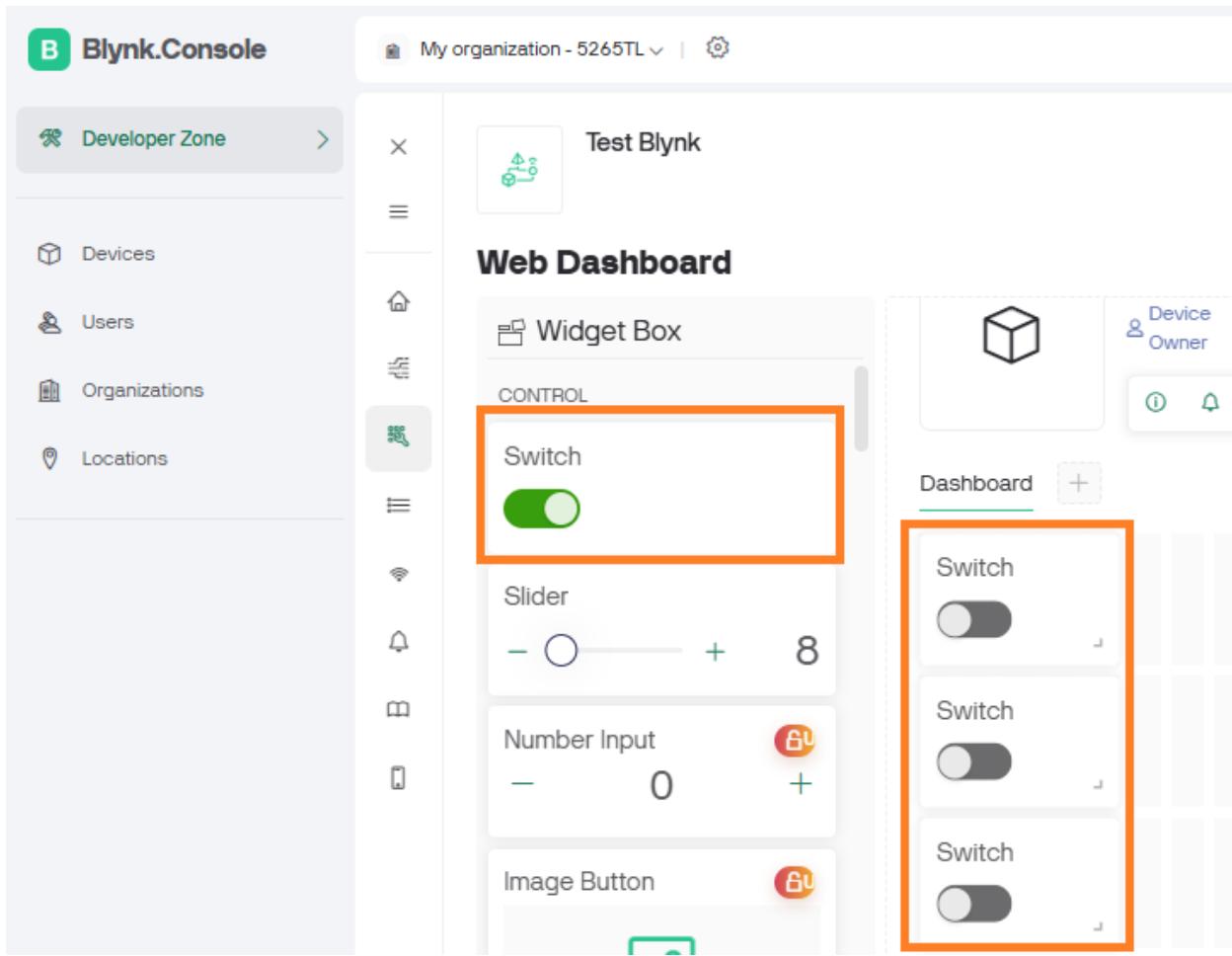
1. เลือก Device จากนั้นกดที่ device Test Blynk ตามภาพ

The screenshot shows the Blynk Console interface. On the left, there's a sidebar with 'Developer Zone' at the top, followed by 'Devices' (which is highlighted with an orange box), 'Users', 'Organizations', and 'Locations'. The main area is titled 'Devices' and contains a search bar ('Start typing') and a 'Add Filter' button. Below the search bar, there are two tabs: 'All 1 ...' (highlighted with an orange box) and 'My devices'. A table lists one device: 'Test Blynk'. The table columns include 'Name' (with a checkbox), 'Auth Token' (with a dropdown arrow), 'Device Owner' (showing 'iFOlqlckKlyccCL7RtwX_WS2sQ...' and 'warakorn@tni.ac.th [you]'), and 'Actions' (with a three-dot menu icon). There are also icons for a bell, a gear, and a person.

- ขั้นตอนมาให้เลือก Edit Dashboard

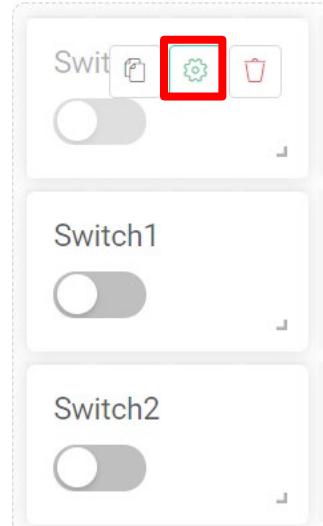
The screenshot shows the Blynk Console interface for the 'Test Blynk' device. The left sidebar is identical to the previous screenshot. The main area shows the device details: '1 DEVICE' (highlighted with an orange box), a search bar ('Search'), and a card for 'Test Blynk' which includes a cube icon, the name 'Test Blynk', the status 'Offline', and a 'My o...' link. Below the card are time filters: 'Live', '1h', '6h', '1d', '1w', '1mo', '3mo', '6mo', '1y', and a 'More' button. To the right of the card, there are several small icons: a mail icon, a bell, a gear, a downward arrow, three dots, and an 'Edit' button. At the bottom, there's a section titled 'No Dashboard widgets' with the sub-instruction 'Edit the dashboard to add widgets' and a large green 'Edit Dashboard' button (highlighted with an orange box).

- ลาก Widget Switch 3 ตัว ไปยัง Dashboard

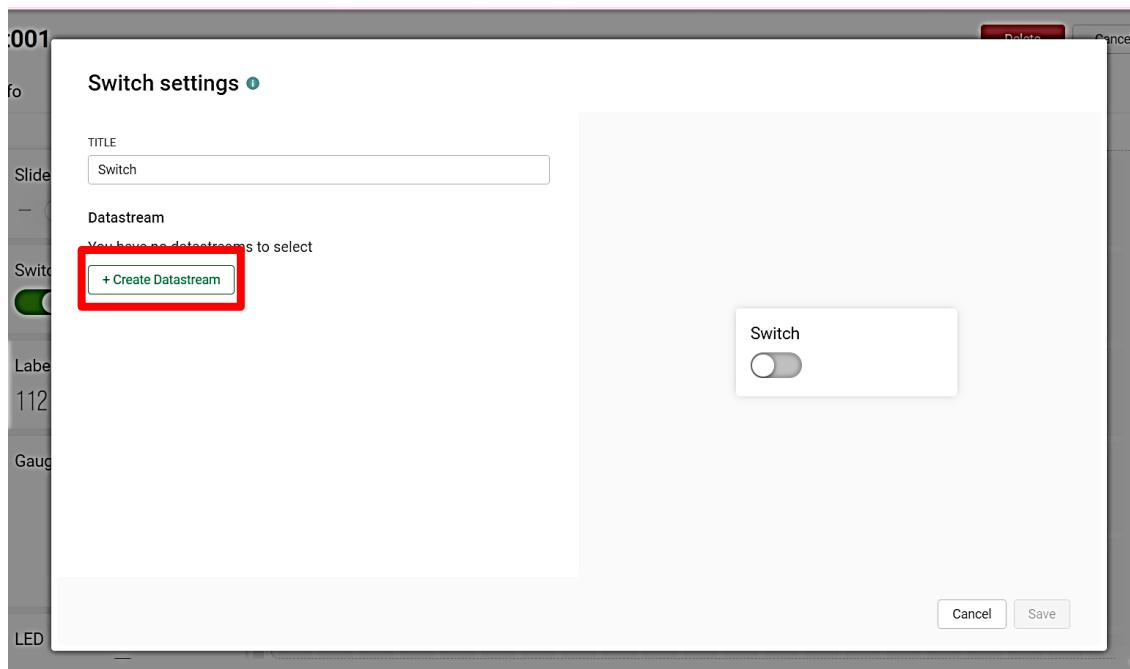


2. ตั้งค่า switch แต่ละตัว

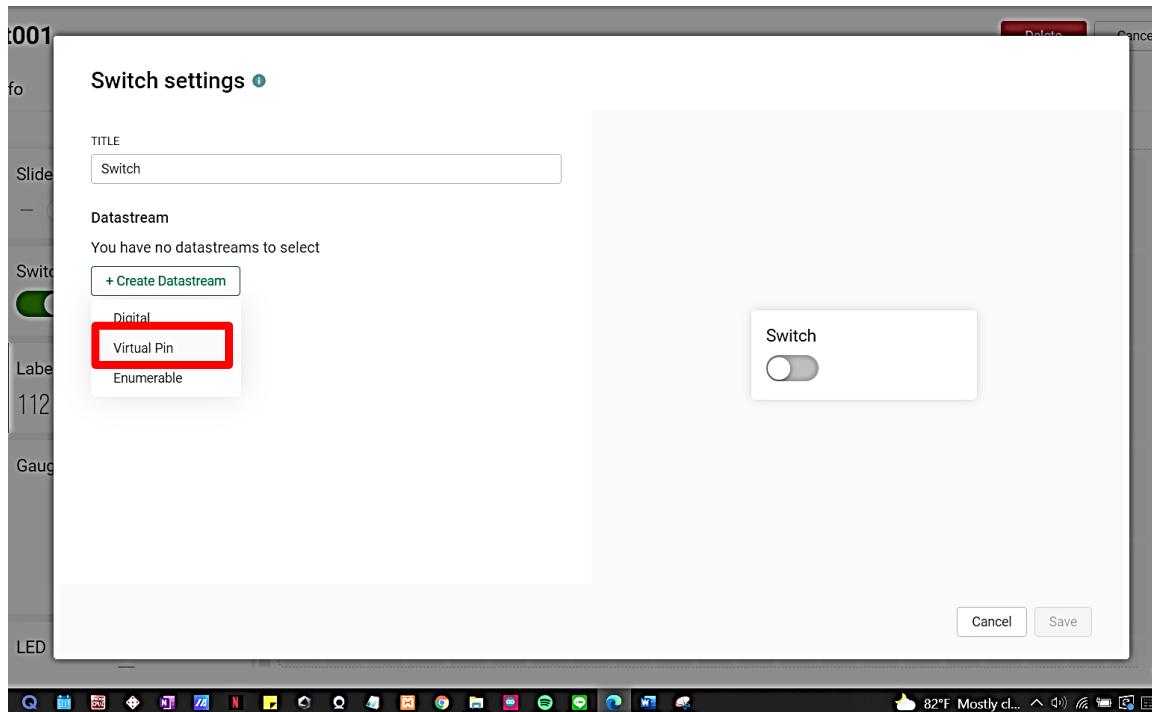
2.1 กดเพื่อตั้งค่า Switch



2.2 กดเพื่อตั้งค่าการรับส่งข้อมูล



2.3 กดเลือก virtual pin



2.4 ให้ตั้งค่า PIN เป็น V0 และกด Create

Datastream

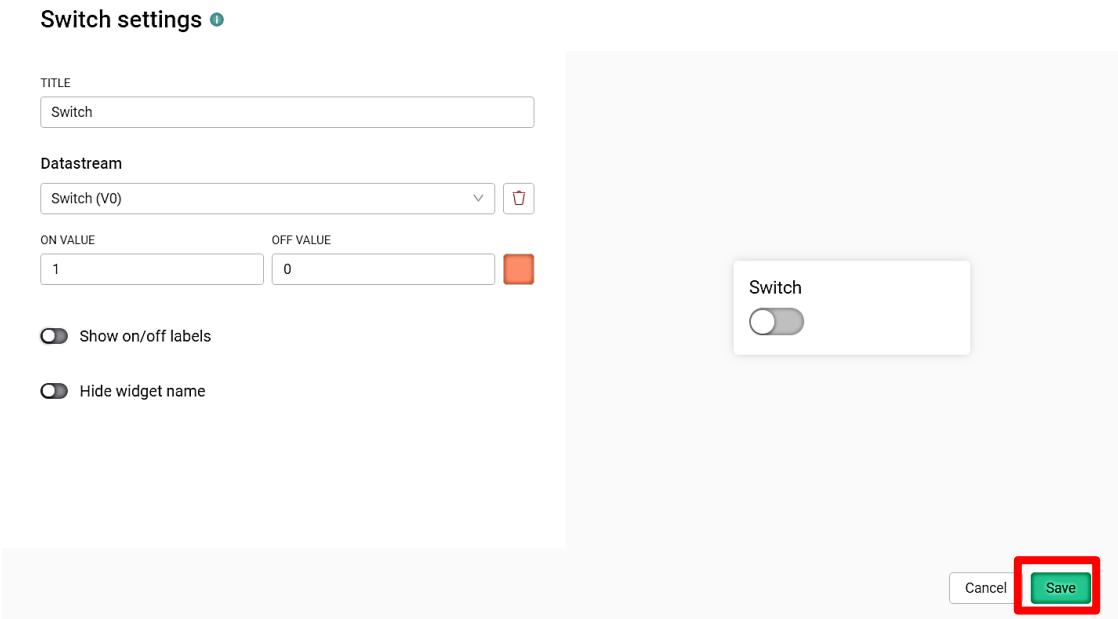
The screenshot shows the "Virtual Pin Datastream" configuration dialog. It includes fields for NAME (Switch), ALIAS (Switch), PIN (V0, highlighted with a red box), DATA TYPE (Double), and UNITS (None). At the bottom are "Cancel" and "Create" buttons, with "Create" being green.

NAME	ALIAS
Switch	Switch

PIN	DATA TYPE
V0	Double

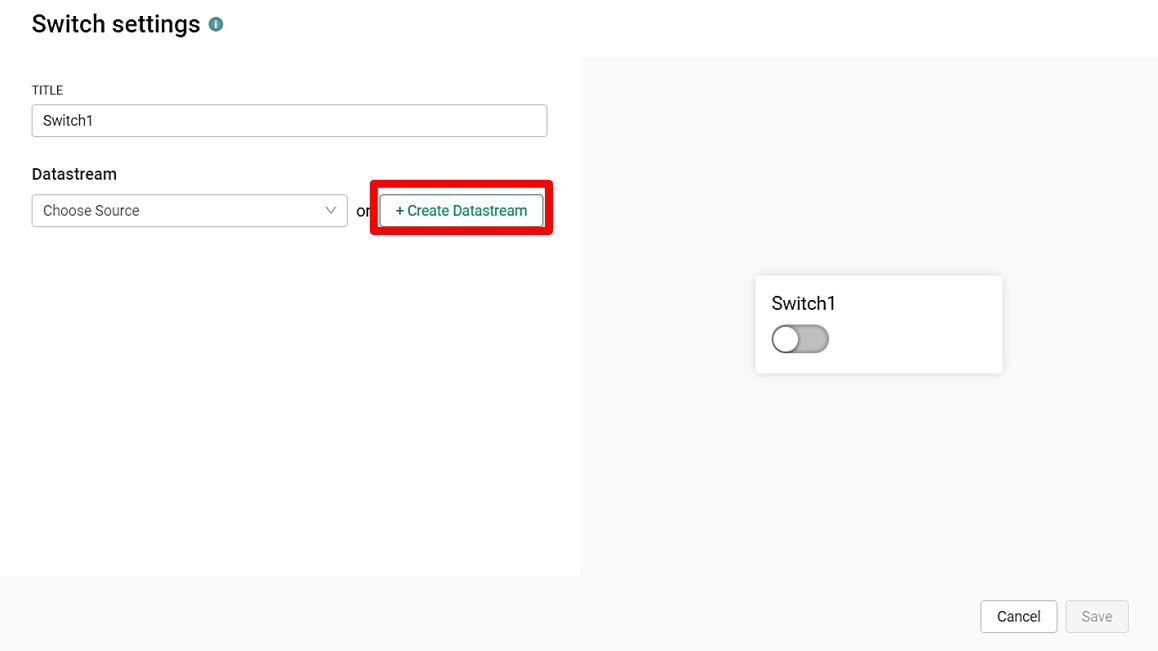
UNITS
None

2.5 กด save เพื่อบันทึกการตั้งค่า switch

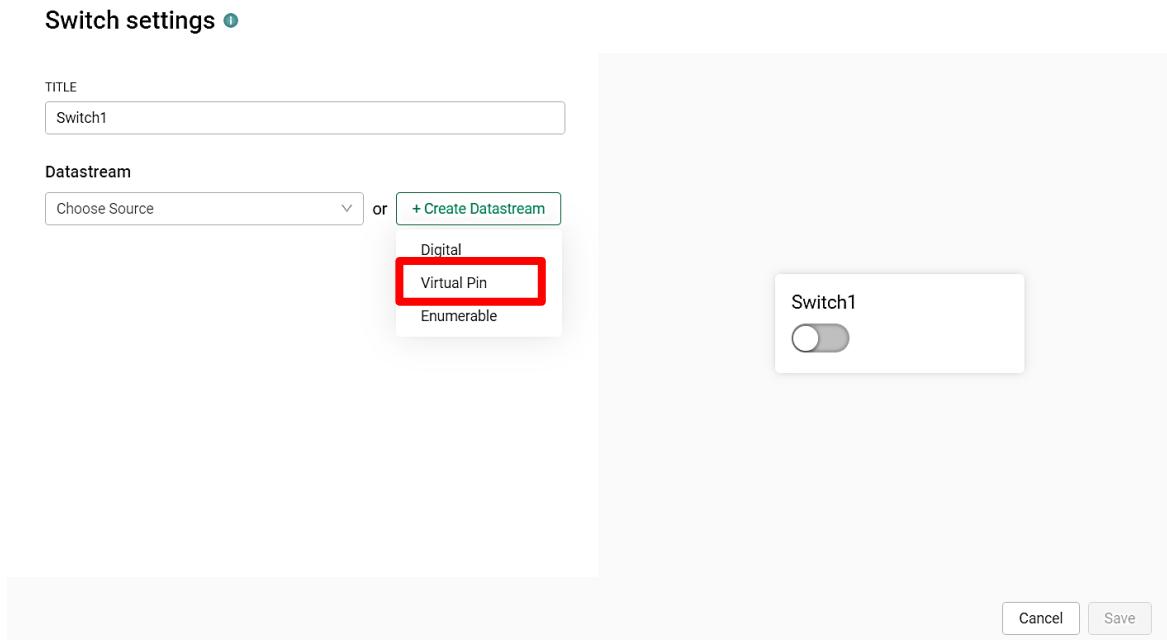


3 ตั้งค่า switch1

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	

Cancel **Create**

3.4 กด save เพื่อบันทึกการตั้งค่า switch

Switch settings ⓘ

TITLE
Switch1

Datastream
Switch1 (V1)

ON VALUE
1

OFF VALUE
0

Show on/off labels

Hide widget name

Switch1

Save

4 ตั้งค่า switch2

4.1 กด create datasteam เพื่อตั้งค่า switch2

Switch settings ⓘ

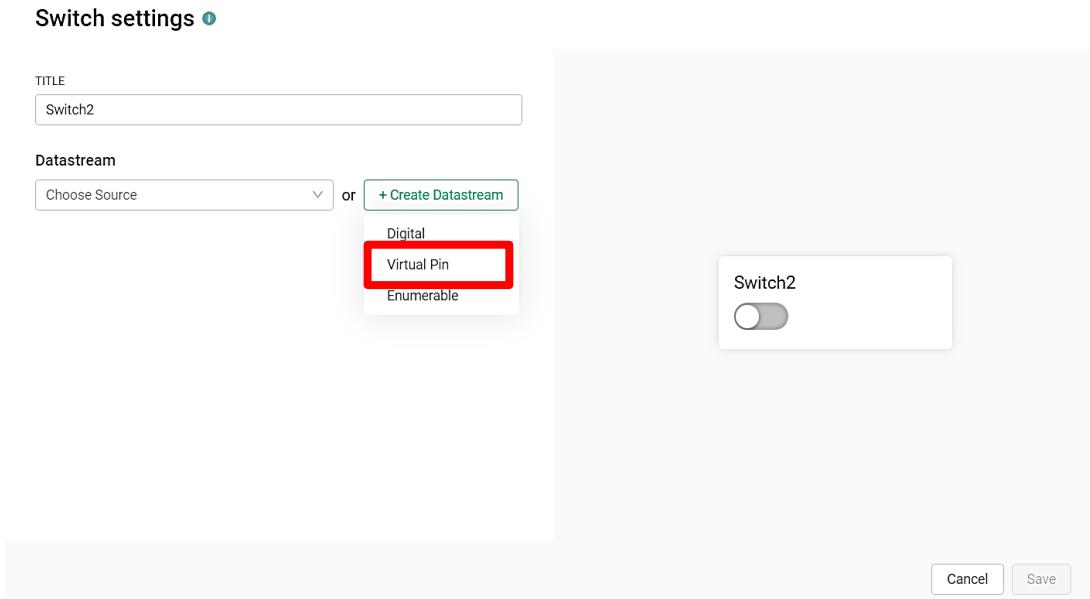
TITLE
Switch2

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

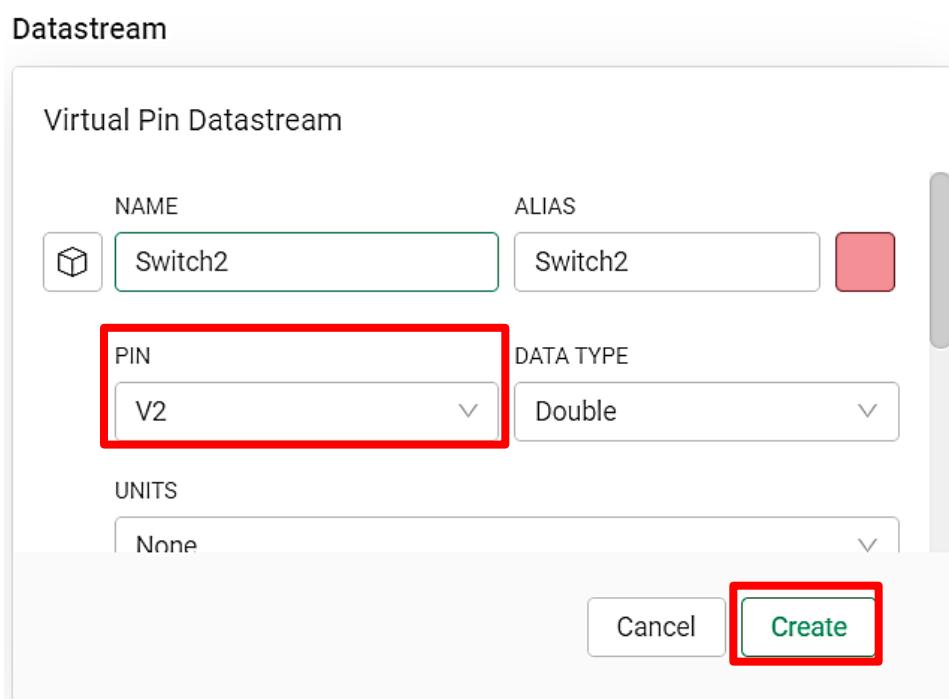
Switch2

Cancel Save

4.2 เลือก Virtual Pin

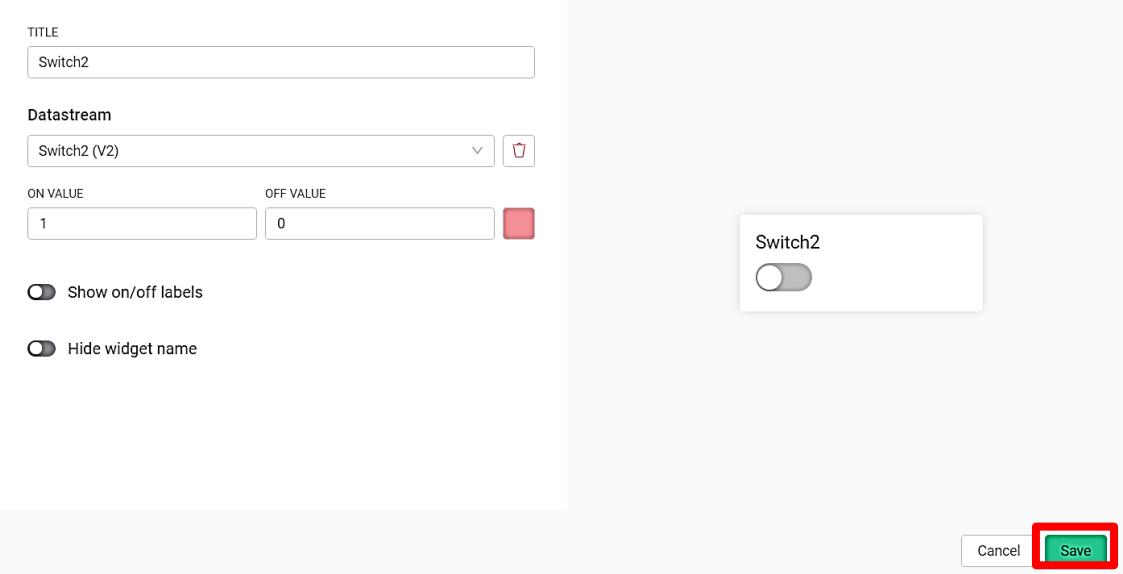


4.3 ให้ตั้งค่า PIN เป็น V2 และกด Create

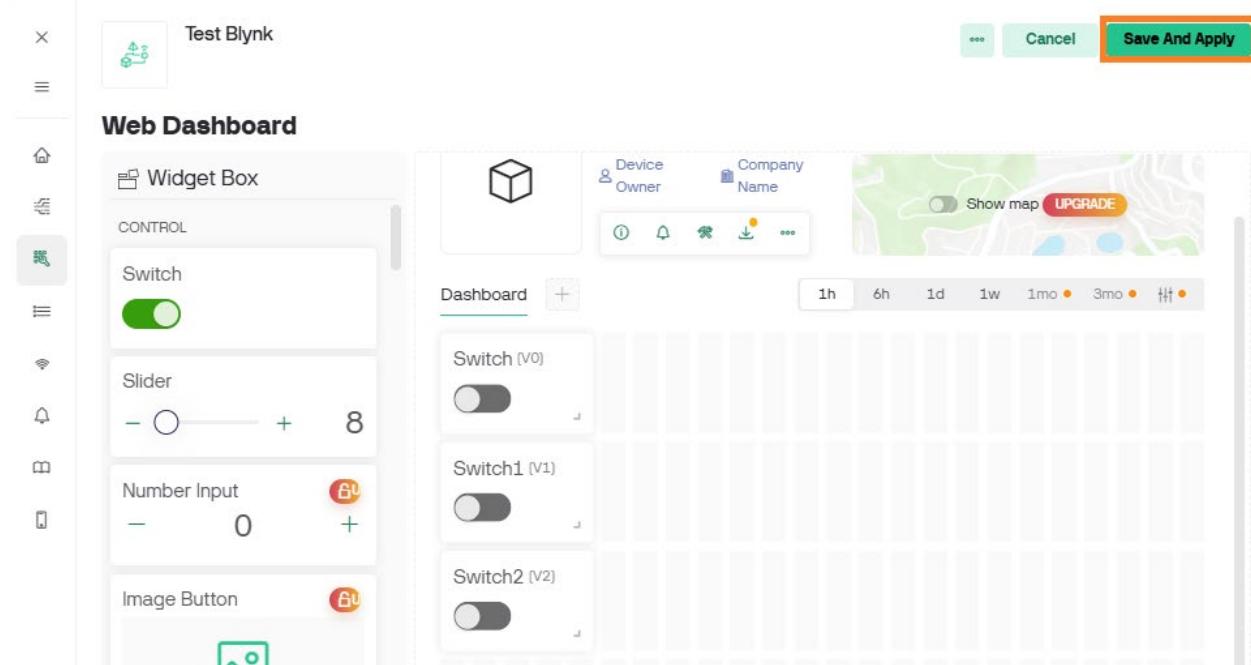


4.4 กด save เพื่อบันทึกการตั้งค่า switch

Switch settings ⓘ



4.5 กด save and Apply เพื่อบันทึกการตั้งค่า dashboard

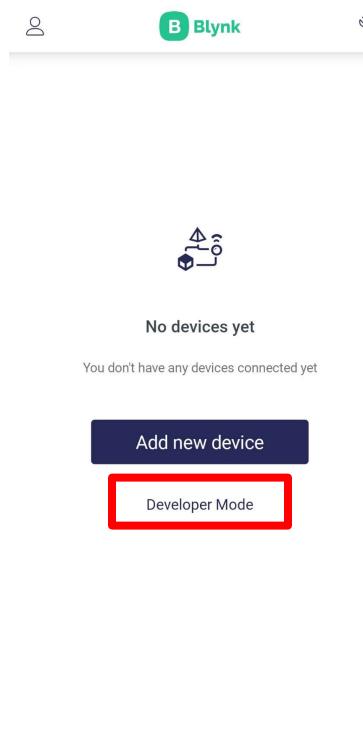


ก่อนใช้งานให้ลง library blynk

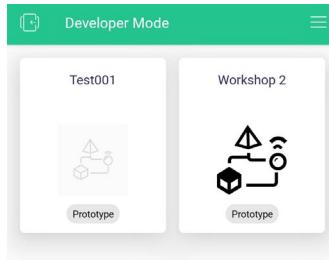


การตั้งค่า Dashboard บนโทรศัพท์

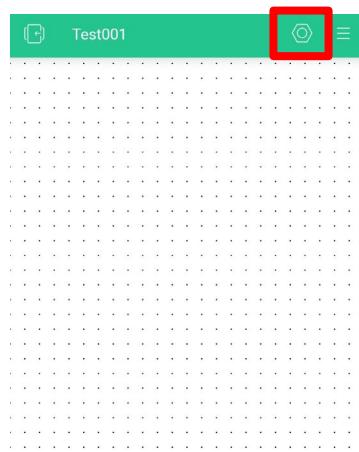
1. เข้าไปที่ Developer Mode



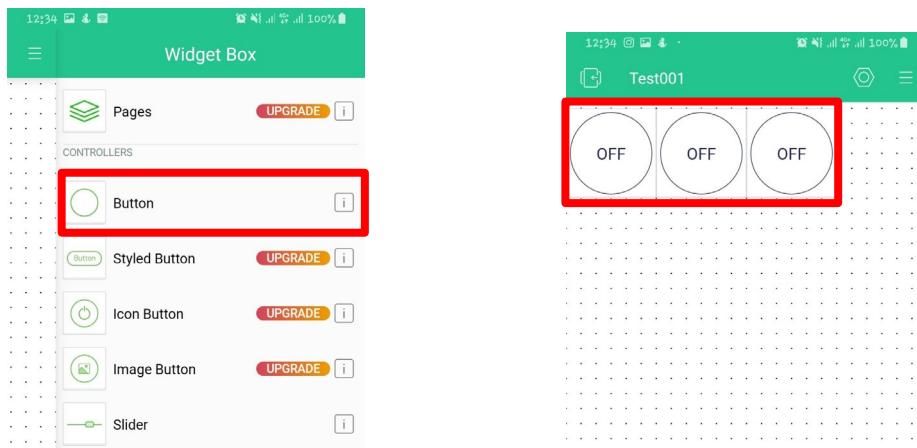
2. เลือก Prototype ของตนเอง



3. กดไปที่ปุ่มดังรูป



4. เลือกไปที่ Button โดยกดเลือกมาทั้งหมด 3 Button

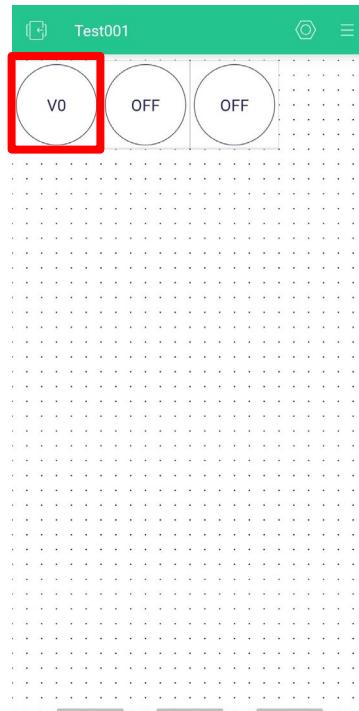


5. กดไปที่ Button ตัวที่ 1 เพื่อทำการตั้งค่า และกดเลือก data stream ที่ต้องการ

The image consists of three screenshots of a mobile application interface for button configuration:

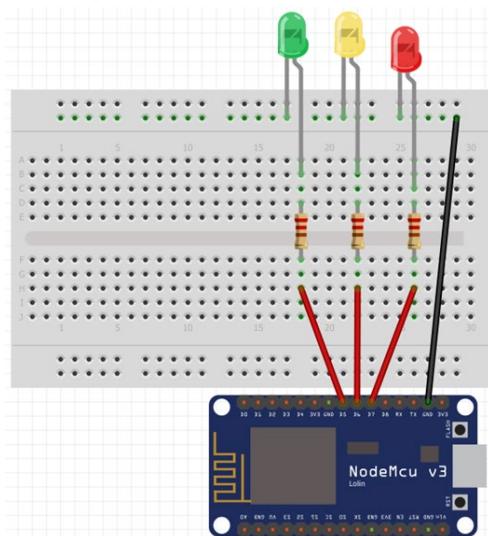
- Screenshot 1:** Shows the "Button Settings" screen. A dropdown menu labeled "Select Data Stream" is open, displaying "Switch1", "Switch2", and "Switch3". The "Switch1" option is highlighted with a red box.
- Screenshot 2:** Shows the "Select Data Stream" screen again, but now "Switch1" is selected, indicated by a checkmark icon next to it.
- Screenshot 3:** Shows the "Button Settings" screen for "Switch1". It displays the "Value" as "1". Below this, there are sections for "Mode" (Push or Switch), "On/Off Labels" (OFF and ON), "Font Size" (Medium), and "Design" (Text).

6. จะได้ผลลัพธ์ดังรูป จากนั้นนำมาขั้นตอนที่ 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() ? ขึ้นการเช็คค่ามีการกดบุ๊มหรือไม่ 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

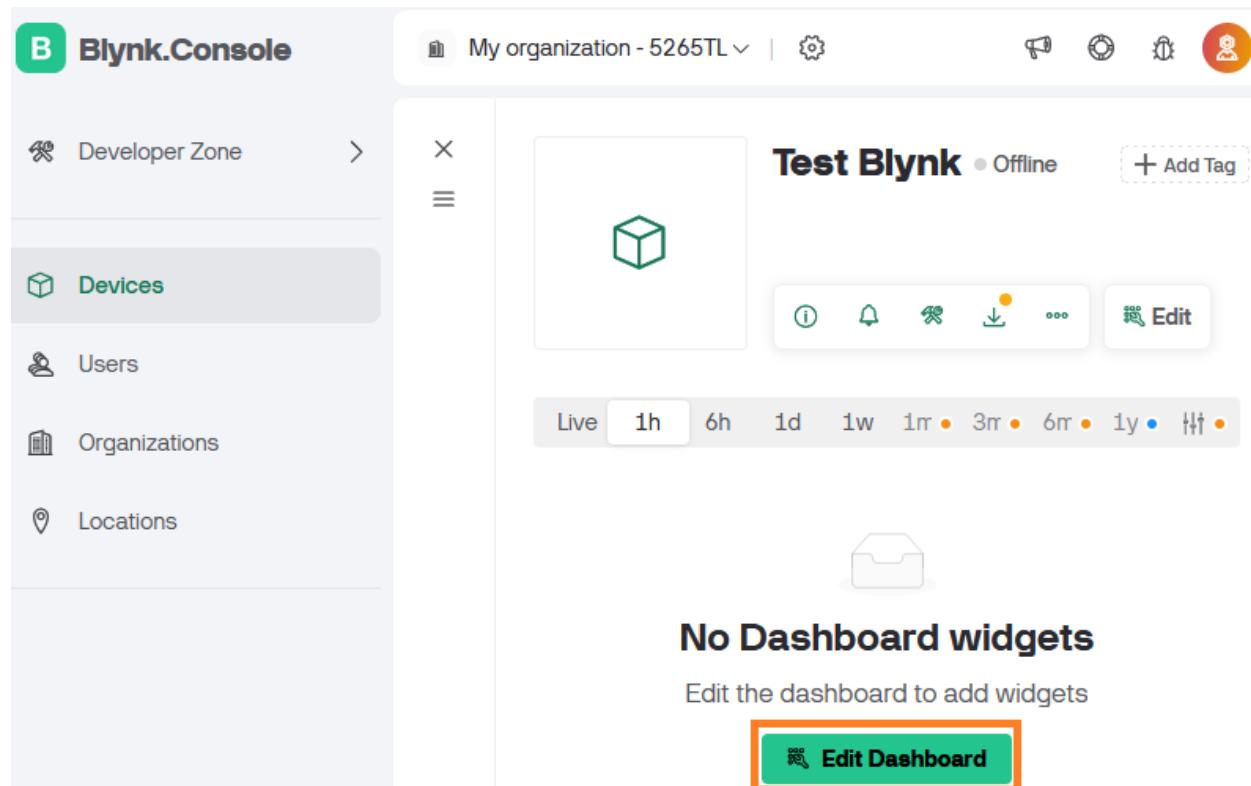
Workshop 11 Blynk กับ HC-SR04

อุปกรณ์ที่ต้องใช้

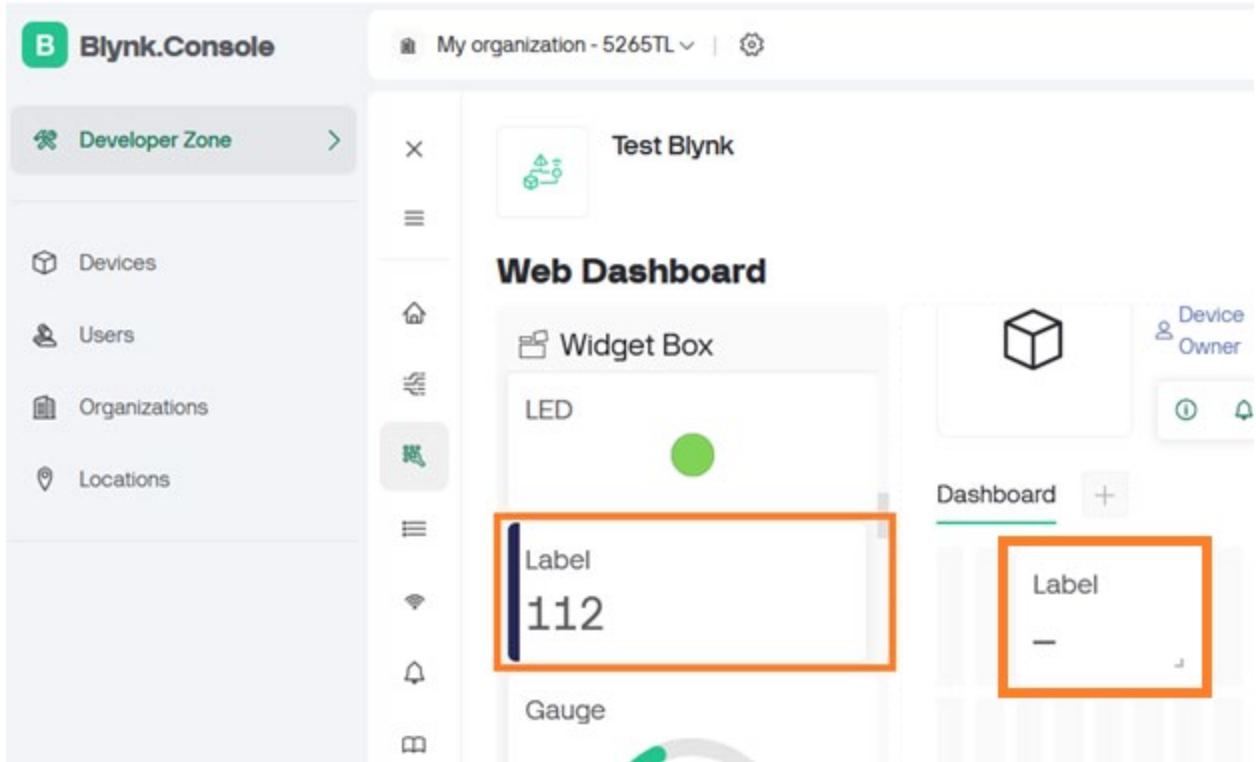
1. NodeMCU ESP8266
2. โมดูลอัลตราโซนิกเซนเซอร์
3. LED สีเขียวและแดง
4. ตัวต้านทานขนาด 220 โอห์ม 2 ตัว

การตั้งค่า Dashboard

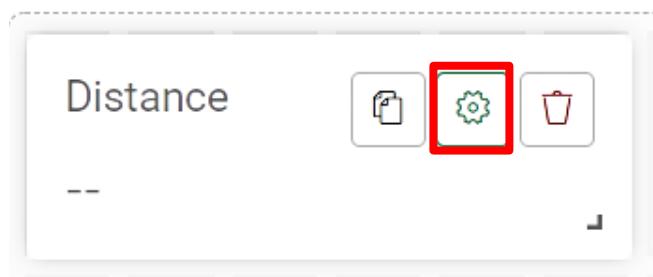
1. เลือก Edit Dashboard



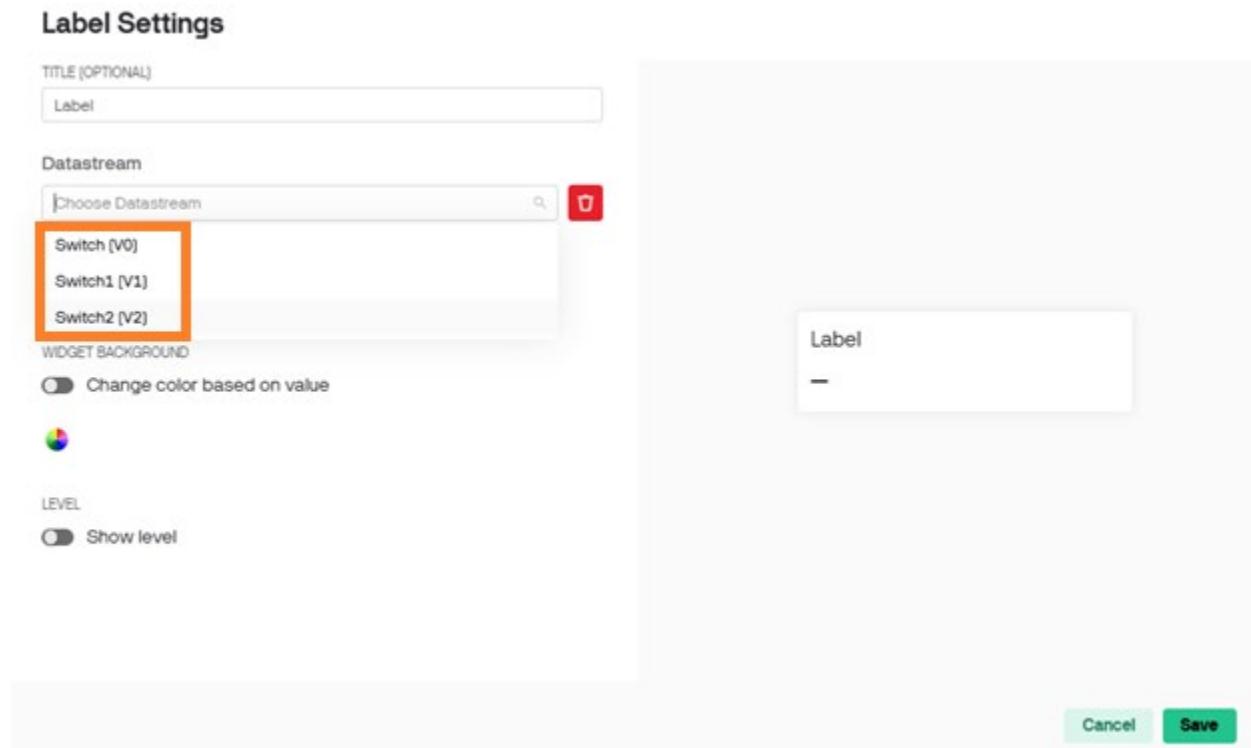
2. เลือก widget Label นำมาใส่ใน Dashboard



2 กดเพื่อตั้งค่าการรับส่งข้อมูล



3. จะสังเกตว่าตรง Datastream Virtual Pin V0, V1 และ V2 ถูกใช้ไปใน workshop ก่อนหน้านี้ ให้ทำการลบ Datastream ทั้ง 3



4. กดปุ่ม Datastreams ตามรูป

The screenshot shows the Blynk Web Dashboard interface. On the left, there's a sidebar with icons for Home, Datastreams (highlighted with an orange border), Metadata, Connection Lifecycle, Events & Notifications, User Guides, and Mobile Dashboard. The main area has a title "Test Blynk" and a subtitle "Web Dashboard". A "Widget Box" section contains three controls: a Switch (green), a Slider (value 8), and a Number Input (value 0). To the right is a "Device Name" card for "Device Name" (Online). It includes fields for "Device Owner" (with a user icon) and "Company Name" (with a company icon). Below the card is a "Dashboard" section with a "Label" field and a "1h" time indicator.

5. ลบ Datastreams ทั้ง 3 ชิ่งสร้างไว้จาก Workshop ที่แล้ว

The screenshot shows the Blynk Datastreams configuration screen. The left sidebar includes "TEST BLYNK", "Datastreams" (selected and highlighted in grey), "Web Dashboard", "Metadata", "Connection Lifecycle", "Events & Notifications", "User Guides", and "Mobile Dashboard". The main area has a title "Test Blynk" and a subtitle "Datastreams". A search bar says "Search datastream" and a green button says "+ New Datastream". Below is a table titled "3 Datastreams" with columns: Id, Name, Alias, Color, Pin, and Actions. The data is as follows:

Id	Name	Alias	Color	Pin	Actions
1	Switch	Switch	Pink	V0	
2	Switch1	Switch1	Dark Blue	V1	
3	Switch2	Switch2	Light Blue	V2	

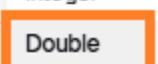
6. เมื่อlob Datastreams ทุกตัวครบแล้ว ให้กด New Datastream และเลือก Virtua Pin

The screenshot shows the Blynk application interface for managing Datastreams. On the left, there's a sidebar with options like Home, Datastreams (which is selected and highlighted in grey), Web Dashboard, Metadata, Connection Lifecycle, Events & Notifications, User Guides, and Mobile Dashboard. The main area is titled "Datastreams" and contains a brief description: "Datastreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators." Below this is a button labeled "+ New Datastream". A dropdown menu is open, showing options: "Virtual Pin" (which is highlighted with a red box), "Enum", "Location (UPGRADE)", "Digital Pin", and "Analog Pin". At the top right of the main area, there are three buttons: "...", "Cancel", and "Save And Apply".

7. ตั้งค่า Data Type ให้เป็น Double เนื่องจากต้องการให้การวัดระยะทางใน Workshop นี้ เป็นตัวเลข
ทศนิยม จากนั้นกดปุ่ม Create

Virtual Pin Datasream

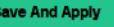
General Expose to Automations

NAME	ALIAS	
 Double V0	Double V0 	
PIN	DATA TYPE	
V0	Double 	
UNITS	Integer	
None	Double 	
MIN	MAX	DEC
0	1	#.## 
Default Value 		
<input checked="" type="checkbox"/> Enable history data		
 ADVANCED SETTINGS		
 		

My organization - 5265TL | 

TEST BLYNK

Test Blynk

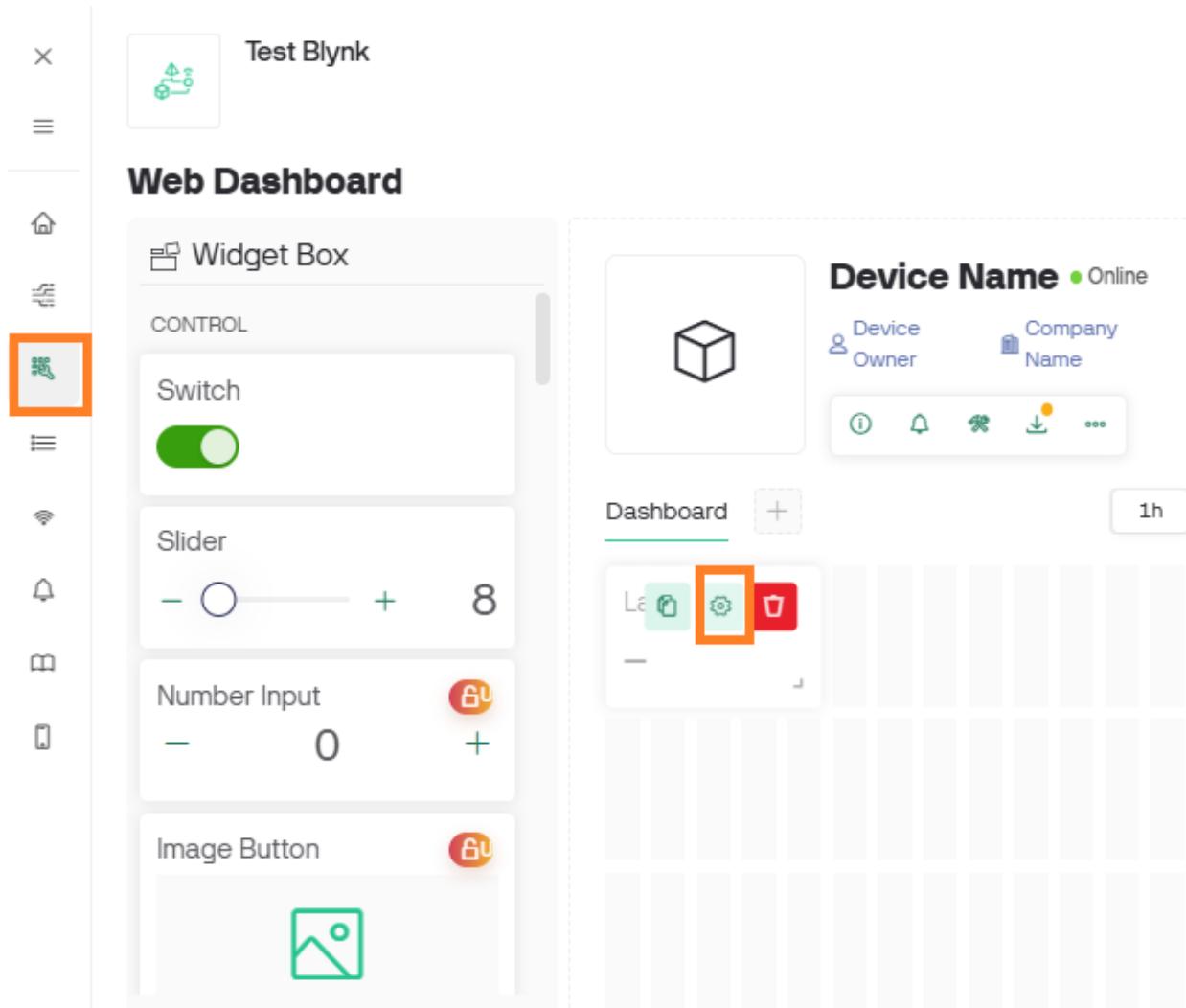
Datastreams

 Datastreams
 Search datastream


1 Datasream

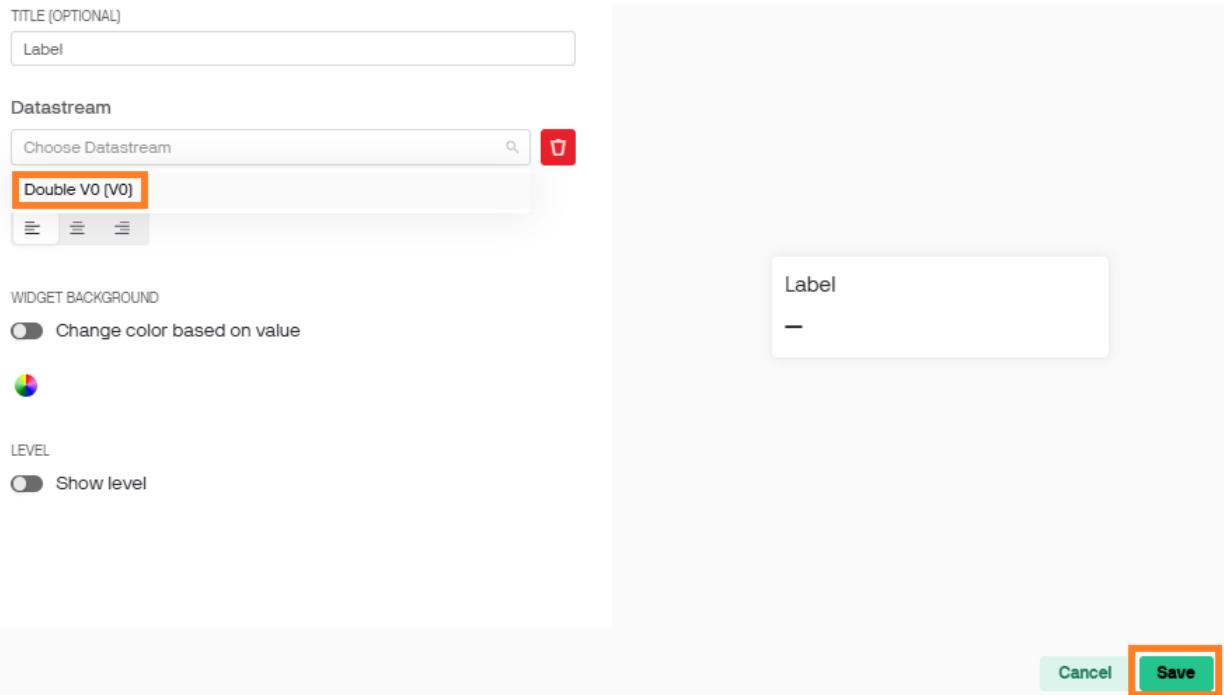
Id	Name	Alias	Color	Pin	Actions
1	Double V0	Double V0		V0	

8. กดปุ่ม Web Dashboard และกดปุ่มตั้งค่า

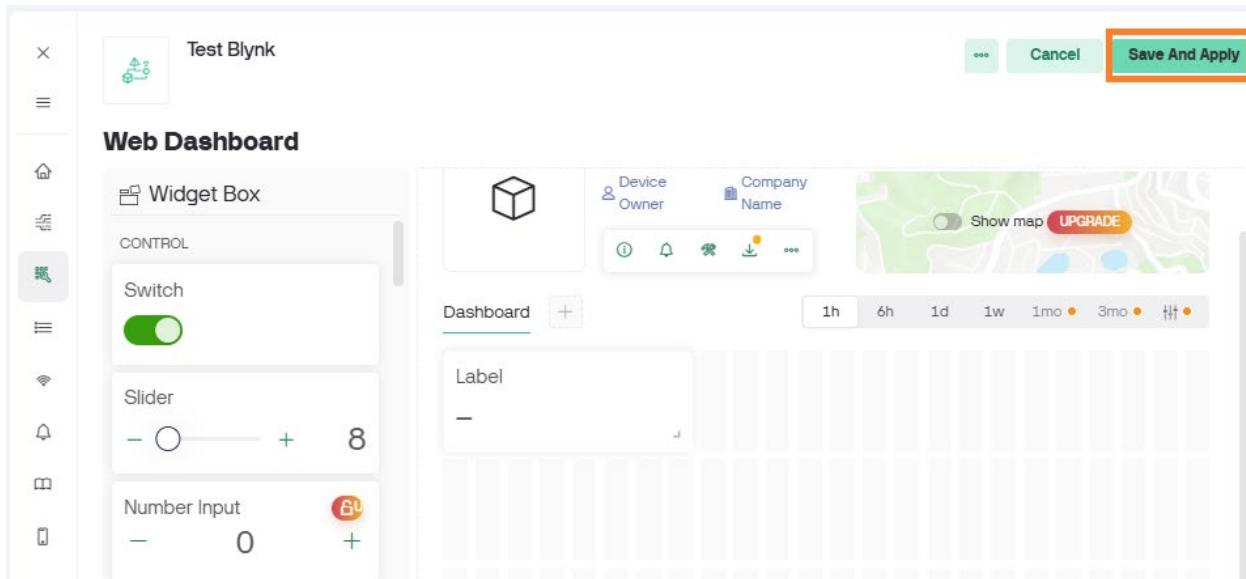


9. เลือก Datastream ที่สร้างไว้ จากนั้นกดปุ่ม Save and Apply

Label Settings

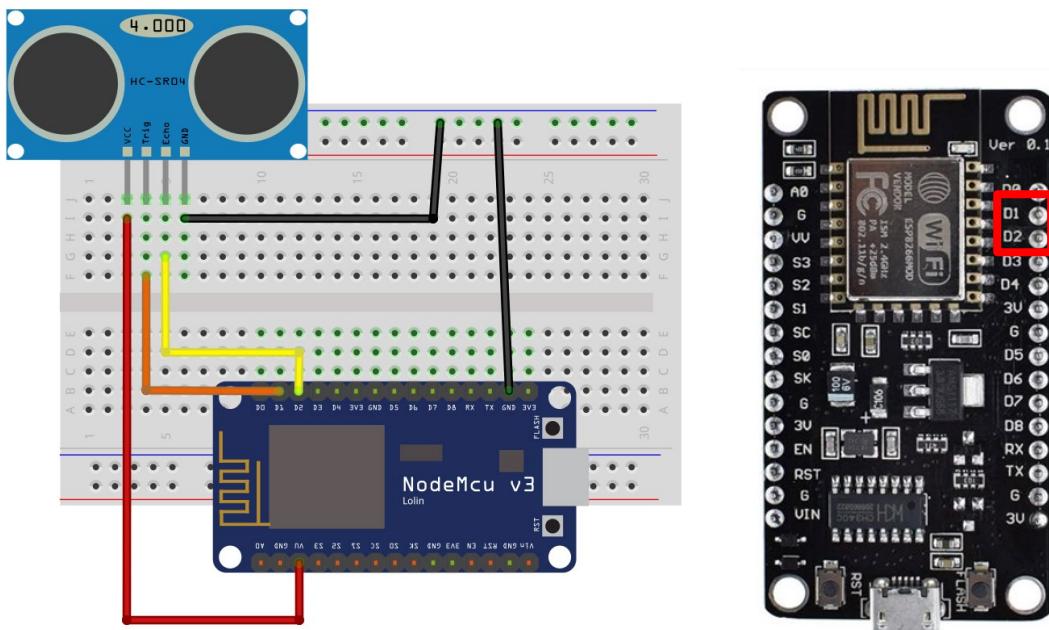


8. กด Save And Apply เพื่อบันทึก dashboard



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>

//กำหนดความเร็วของเสียงให้อยู่ในหน่วย 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 ซึ่งจะคืนค่าเป็นเวลา sound wave travel (หน่วยเวลาเป็น micro seconds)
    duration = pulseIn(ECHO_PIN, HIGH);

    // คำนวนระยะทาง
    distanceCm = duration * SOUND_VELOCITY / 2;
    if (distanceCm >= 200 || distanceCm <= 0) {
        Serial.println("Out of range");
    }
    else {
        Blynk.virtualWrite(v0, distanceCm); // ส่งค่าที่ปัดเศษไปยัง 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

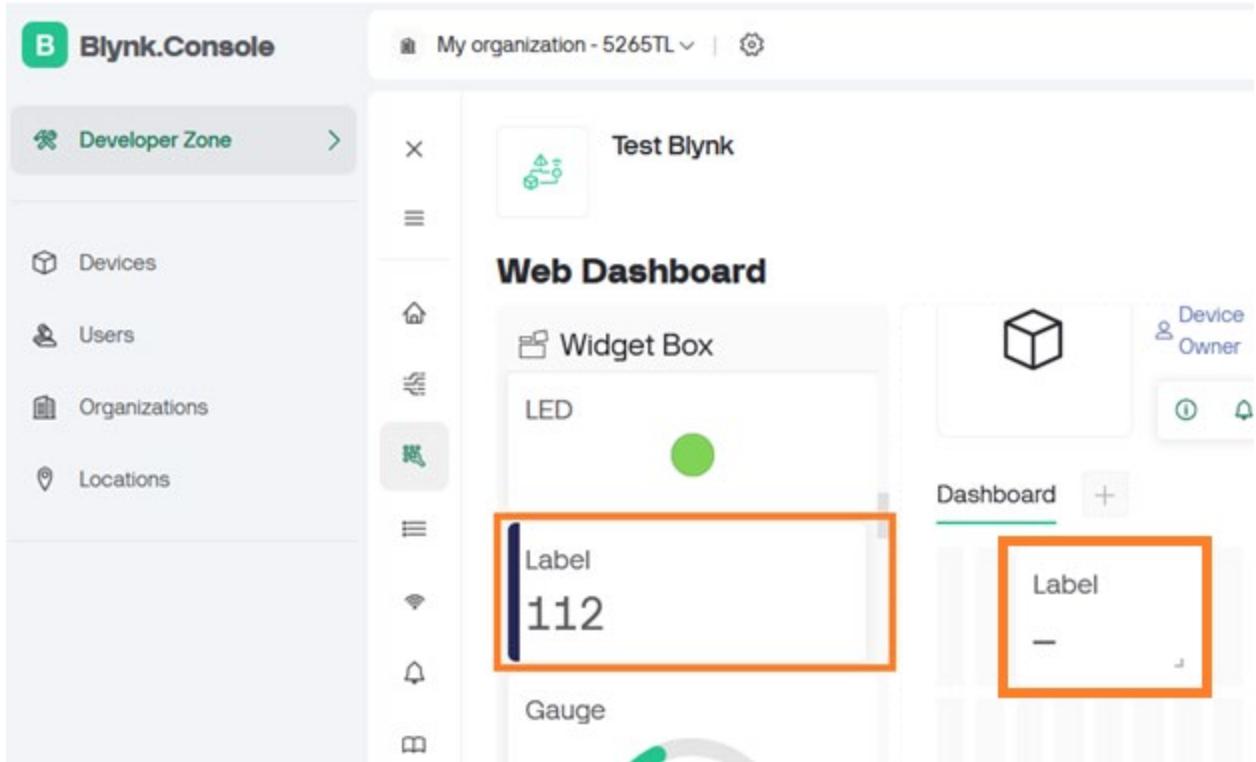
1. สร้าง Virtual Pin Datastream V0 พร้อมทั้งตั้งค่า ตามภาพ

Virtual Pin Datastream

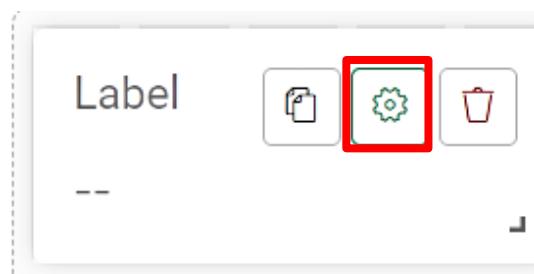
General Expose to Automations

NAME	ALIAS		
Double V0	<input type="text" value="Double V0"/>		
PIN	DATA TYPE		
<input type="text" value="V0"/>	<input type="text" value="Double"/>		
UNITS			
<input type="text" value="Percentage, %"/>			
MIN	MAX	DECIMALS	DEFAULT VALUE
<input type="text" value="0"/>	<input type="text" value="100"/>	<input type="text" value="#.##"/>	<input type="text" value="Default Value"/>
<input checked="" type="checkbox"/> Enable history data			
<input type="button" value="Cancel"/>			

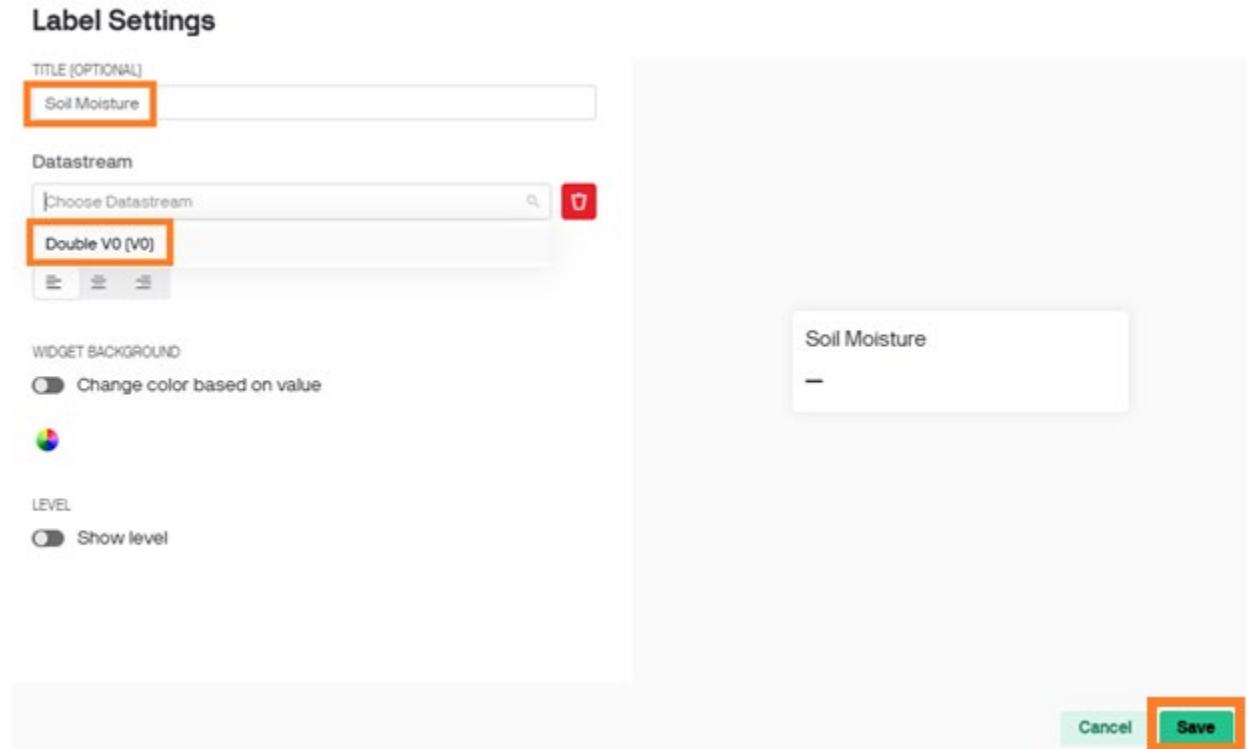
2. เลือก Label widget นำมาระบบที่ Dashboard



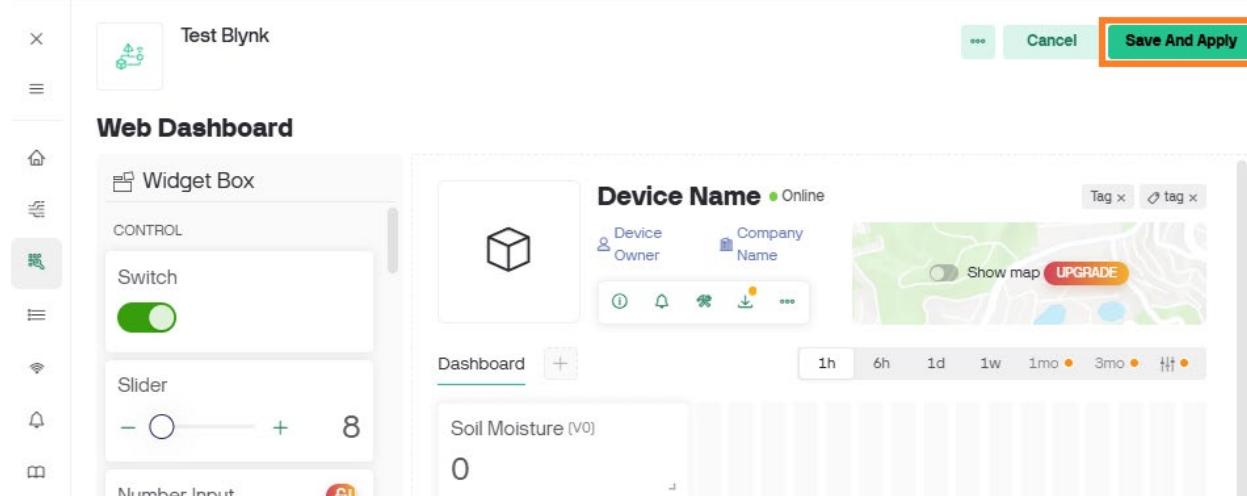
3. กดเพื่อตั้งค่าการรับส่งข้อมูล



4. ตั้งชื่อ Label เป็น Soil Moisture และเลือก Datastream Double V0 จากนั้นกดปุ่ม Save

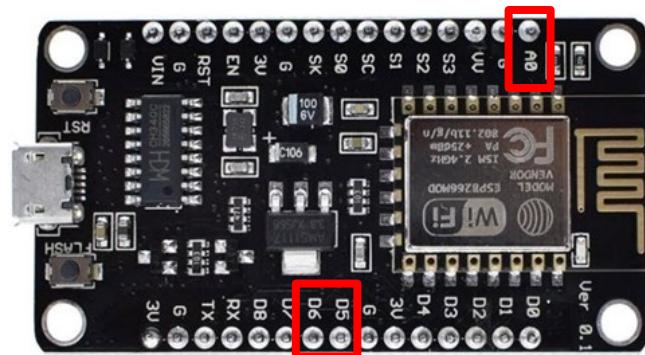
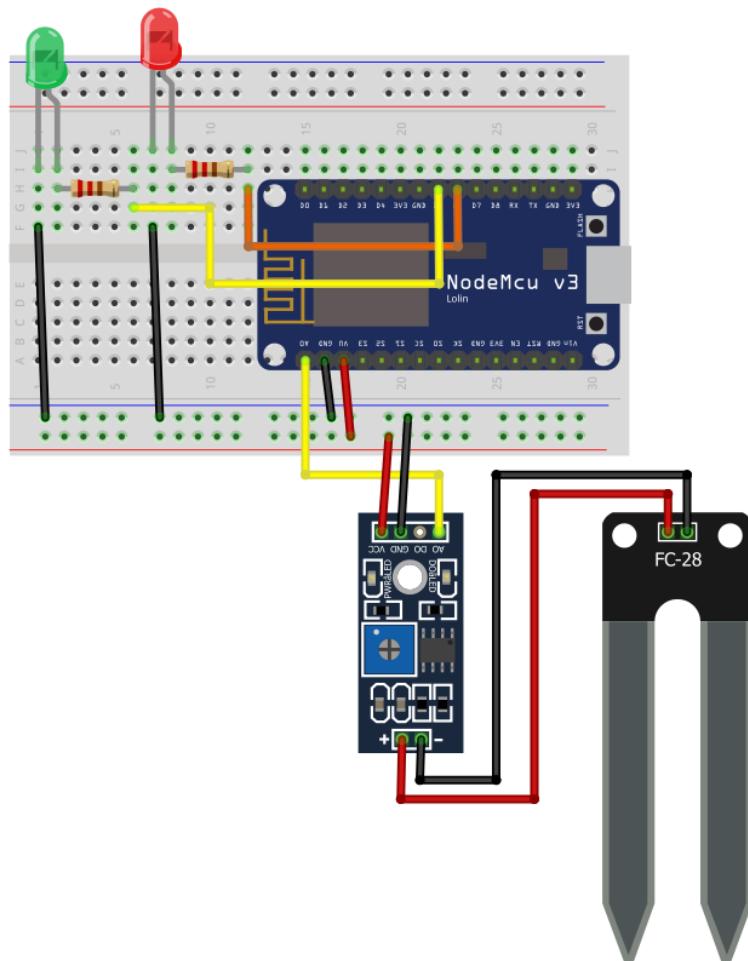


5. กด Save and Apply เพื่อบันทึกหน้า dashboard



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 จิตมิชช์ 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

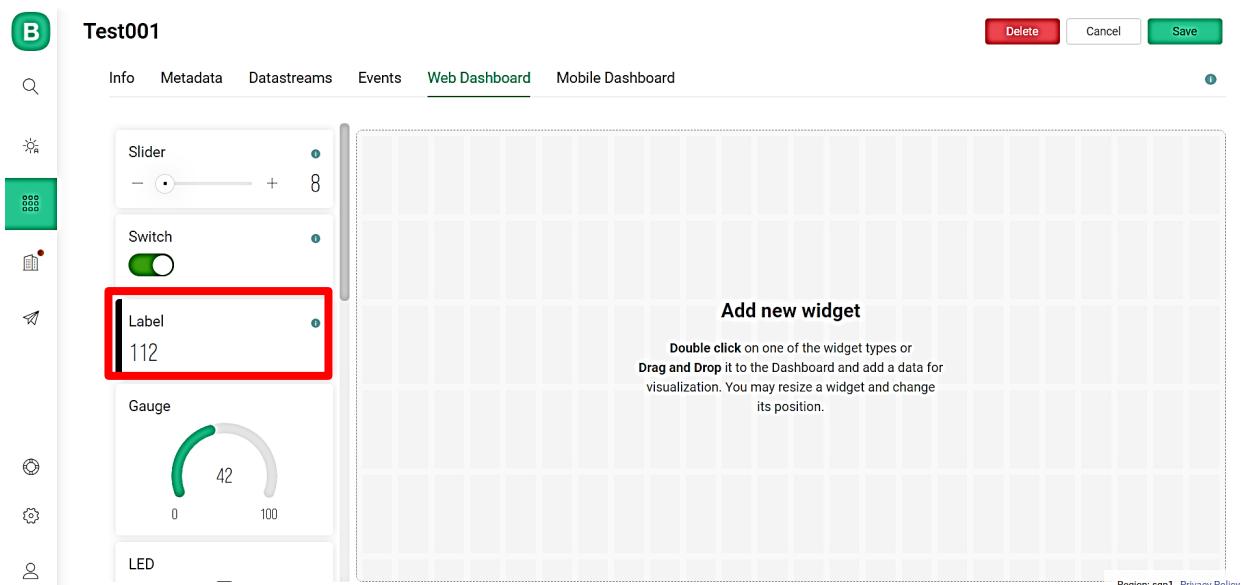
Workshop 13 Blynk and DS18B20

อุปกรณ์ที่ต้องใช้

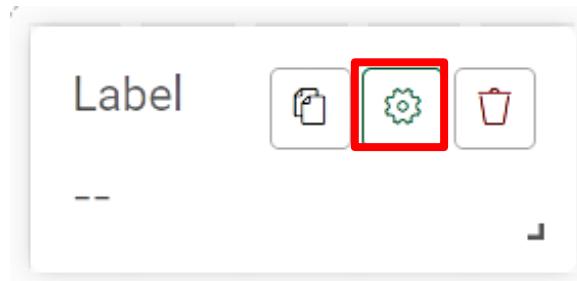
1. NodeMCU ESP8266
2. เช่นเซอร์วัสดอุณหภูมิDS18B20
3. LED สีแดง
4. Buzzer เสียงเตือน
5. ตัวต้านทานขนาด 4.7k โอห์ม 1 ตัว และตัวต้านทานขนาด 220 โอห์ม 1 ตัว

การตั้งค่า Dashboard

1 เลือก Label widget นำมาระบบหน้า Dashboard



2 กดเพื่อตั้งค่าการรับส่งข้อมูล



3 ตั้งชื่อ Label เป็น Temperature

Label settings ⓘ

TITLE

Datastream

You have no datastreams to select

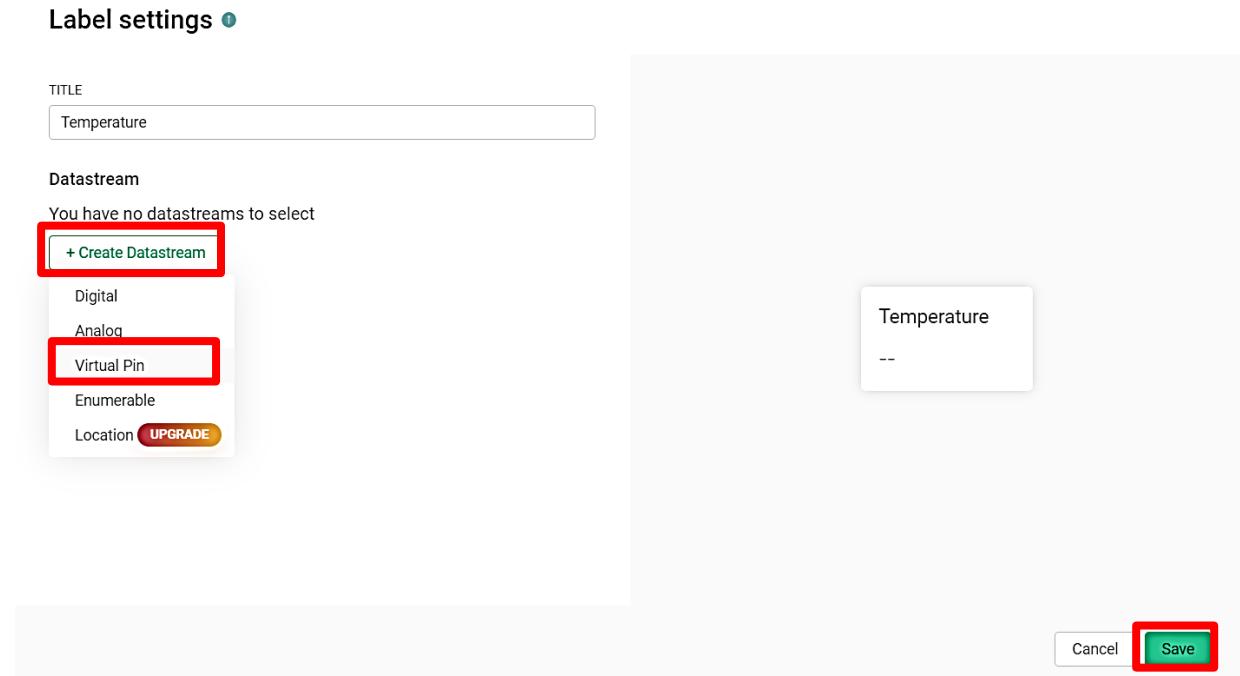
+ Create Datastream

Temperature

Cancel

Save

4 กด Create Datastream และเลือก Virtual Pin



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

Datastream

Virtual Pin Datastream

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

Cancel **Create**

6 กด save เพื่อบันทึก Label

Label settings ⓘ

TITLE

Datastream

Temperature (V0)

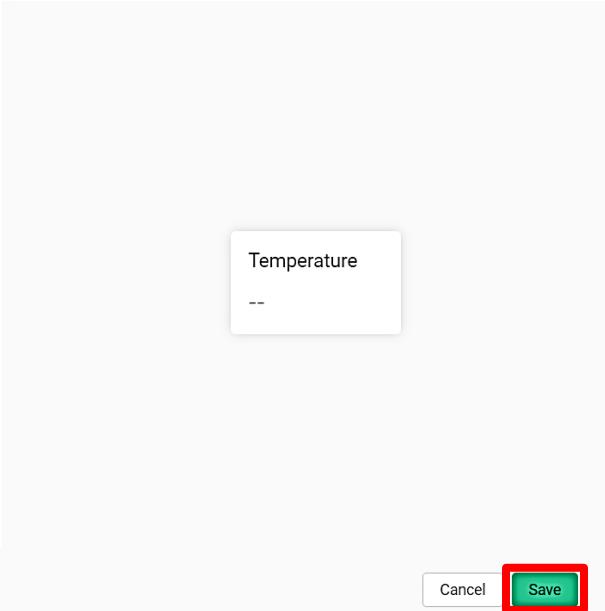
CONTENT ALIGNMENT

WIDGET BACKGROUND

Change color based on value

LEVEL

Show level



7 กด save เพื่อบันทึกหน้า dashboard

B Test001

Info Metadata Datastreams Events Web Dashboard Mobile Dashboard

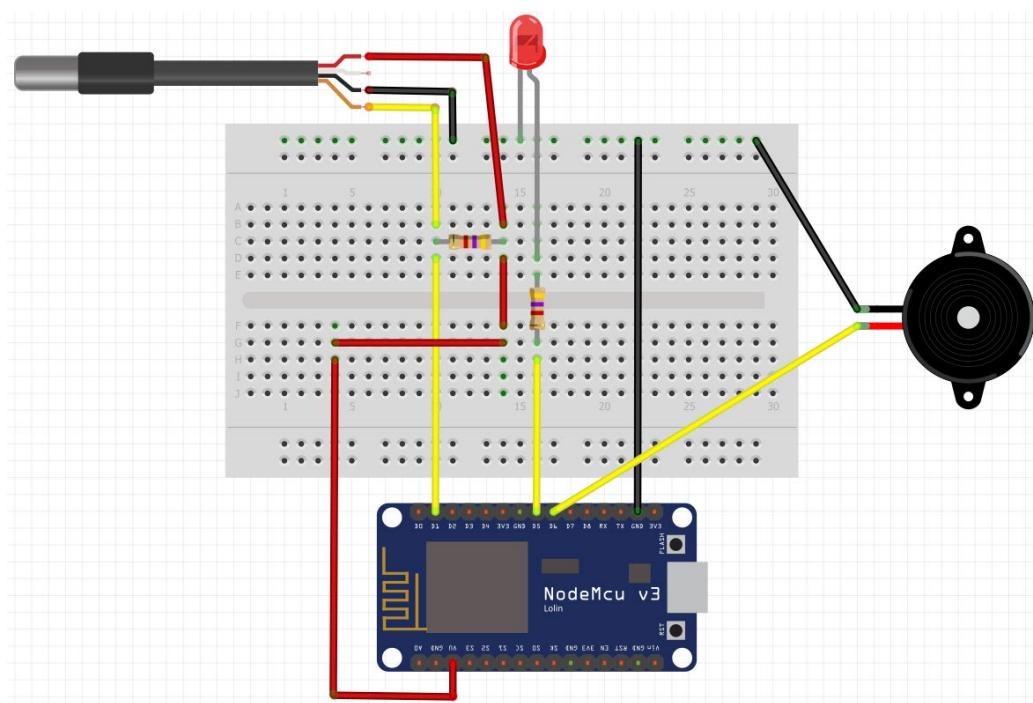
Widgets:

- Slider: Value 8
- Switch: On
- Label: Value 112
- Gauge: Value 42 (Scale 0-100)
- LED: Off

Region: sgp1 Privacy Policy

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 BLINK_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

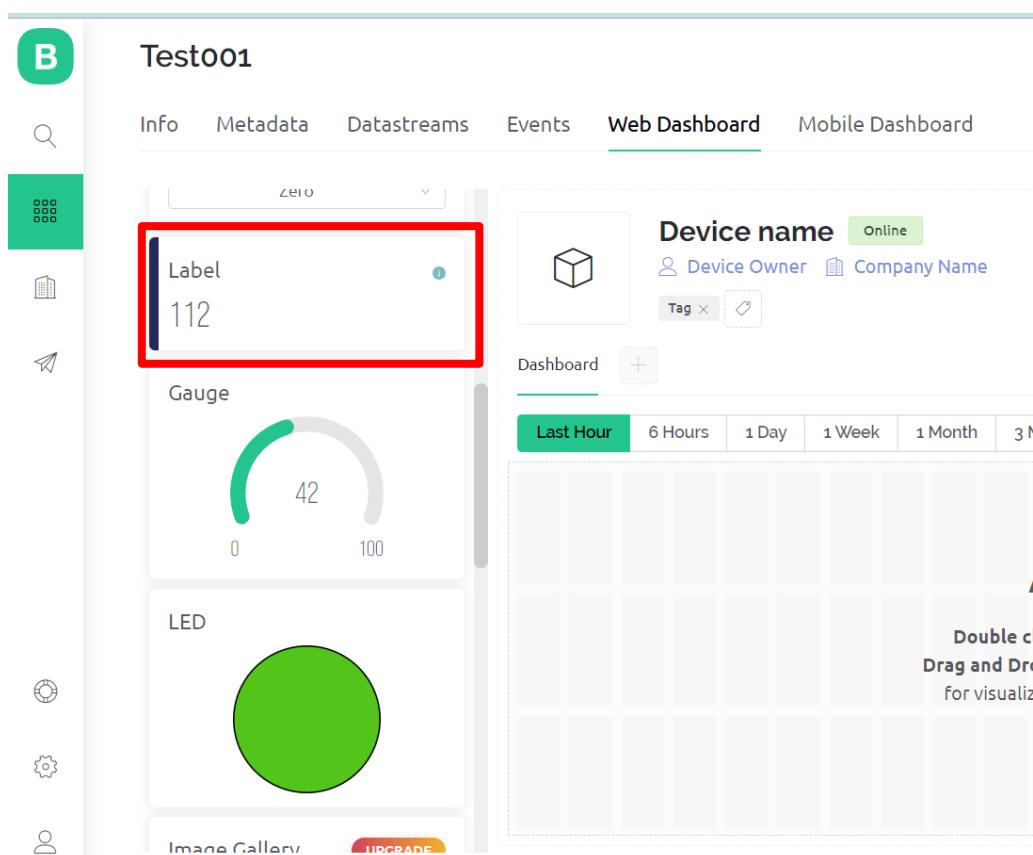
1. Power supply : 3 - 5.5V
2. วัดอุณหภูมิได้ระหว่าง 0 - 50 องศาเซลเซียส +/- 2 องศา
3. วัดความชื้นในอากาศได้ระหว่าง 20 - 90 % +/- 5%
4. เวลาที่ใช้ในการวัดค่า : 1 วินาที

อุปกรณ์ที่ใช้

1. NodeMCU ESP8266
2. เช่นเซอร์วัตอุณหภูมิ DHT22
3. สายน้ำ

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 square with a gear icon, and a red square with a trash bin icon. The green gear icon is highlighted with a red box.

3. ตั้งชื่อ Label ให้ชื่อว่า 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

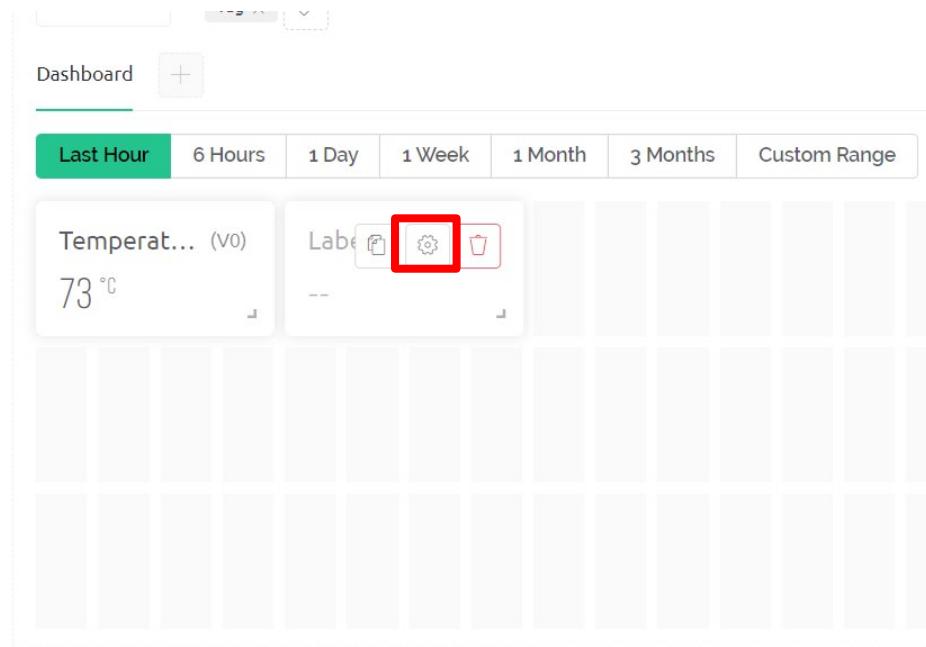
Tempera... (v0)

94 °C

Cancel

Save

8. ตั้งค่า Label ตัวที่สอง



9. ตั้งชื่อ Label ใช้ชื่อว่า Humidity

Label Settings ⓘ

The screenshot shows the 'Label Settings' page. At the top, there's a 'TITLE (OPTIONAL)' field with the word 'Humidity' entered. This entire field is highlighted with a red box. Below this is a 'Datastream' section with a 'Choose Source' dropdown and a 'Create Datastream' button.

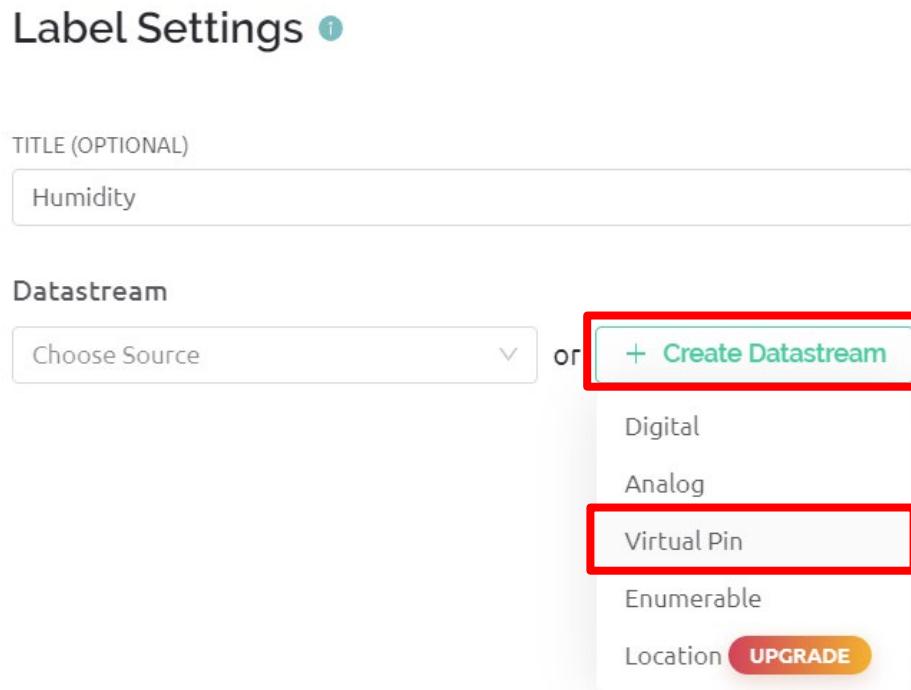
TITLE (OPTIONAL)

Humidity

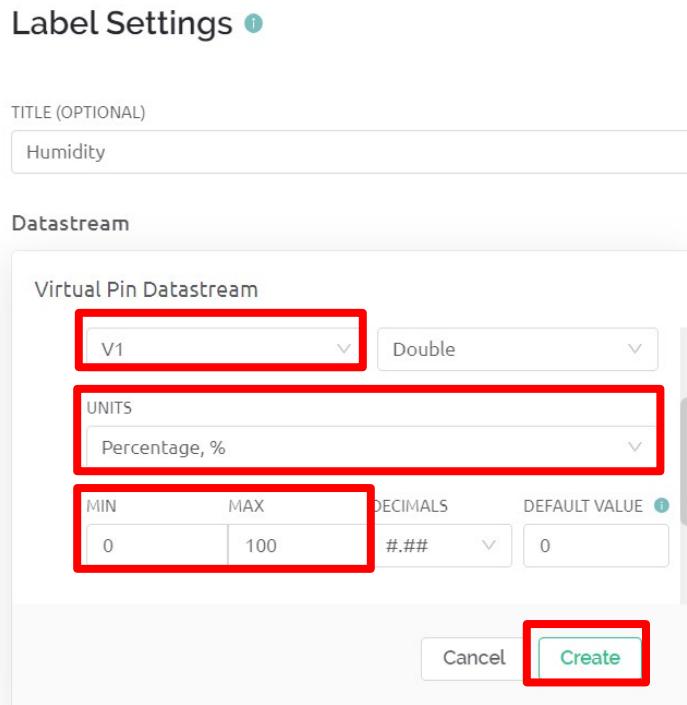
Datastream

Choose Source or + Create Datastream

10. กด Create Datastream เลือก Virtual Pin



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



12. ນັກ save Label

Label Settings

TITLE (OPTIONAL)
Humidity

Datastream
Humidity (V1)

CONTENT ALIGNMENT

WIDGET BACKGROUND Change color based on value

LEVEL Show level

Humidity

--

Cancel **Save**

13. ນັກ Save And Apply

Test001

Info Metadata Datastreams Events Web Dashboard Mobile Dashboard

Menu BETA
Zero

Label
112

Gauge
42

LED

Device name Online
Device Owner Company Name

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

Temperat... (V0) Humidity (V1)
67 °C 75 %

Show map UPGRADE

Cancel **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 area displays device information like 'Device name: online', a map, and two data cards for 'Tempera...' (84 °C) and 'Humidity (V1)' (94 %). Below these are several empty chart slots.

15. ตั้งค่า Chart ตัวที่หนึ่ง

The screenshot shows a dashboard with a 'Last Hour' time range selected. It features two data cards for temperature and humidity, and two empty chart slots labeled 'Chart'. In the second chart slot, a settings menu is open, and a red box highlights the gear icon (settings) in this menu.

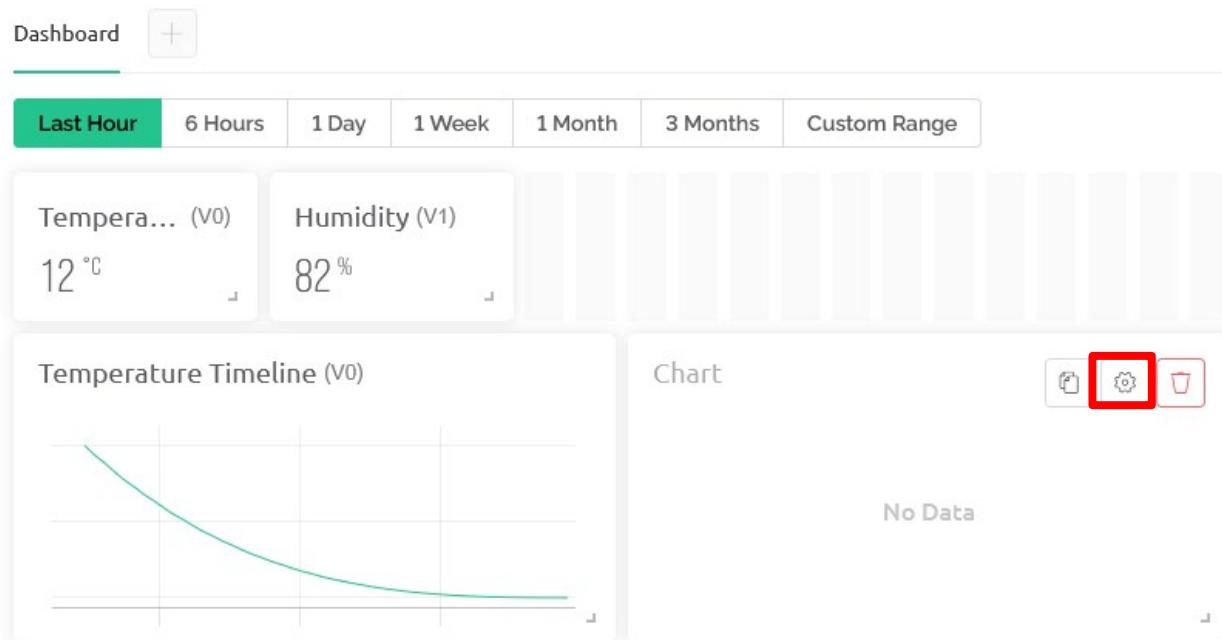
16. ตั้งชื่อ Title ว่า Temperature Timeline จากนั้นกดเลือก Add Datastream และกดเลือก Temperature

The screenshot shows the 'Chart Settings' dialog. On the left, under 'Datastreams', there is a red box around the 'Temperature (V0)' option. 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 'Datastreams' section now includes 'Temperature (V0)'. The preview window on the right displays a line graph titled 'Temperature Timeline (V0)' with a single teal line showing a downward trend from approximately 200 on the left to 100 on the right. At the bottom right of the dialog are 'Cancel' and 'Save' buttons, with the 'Save' button highlighted by a red box.

18. ตั้งค่า Chart ตัวที่สอง



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

Chart Settings

TITLE (OPTIONAL)
Humidity Timeline

Datastreams

+ Add Datastream or + Create New

Humidity (V1)

Temperature (V0)

Show legend

Humidity Timeline

No Data

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

Device name Online

Temperature Timeline (V0)

Humidity Timeline (V1)

Save And Apply

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

Temperature: 60 °C Humidity: 92 %

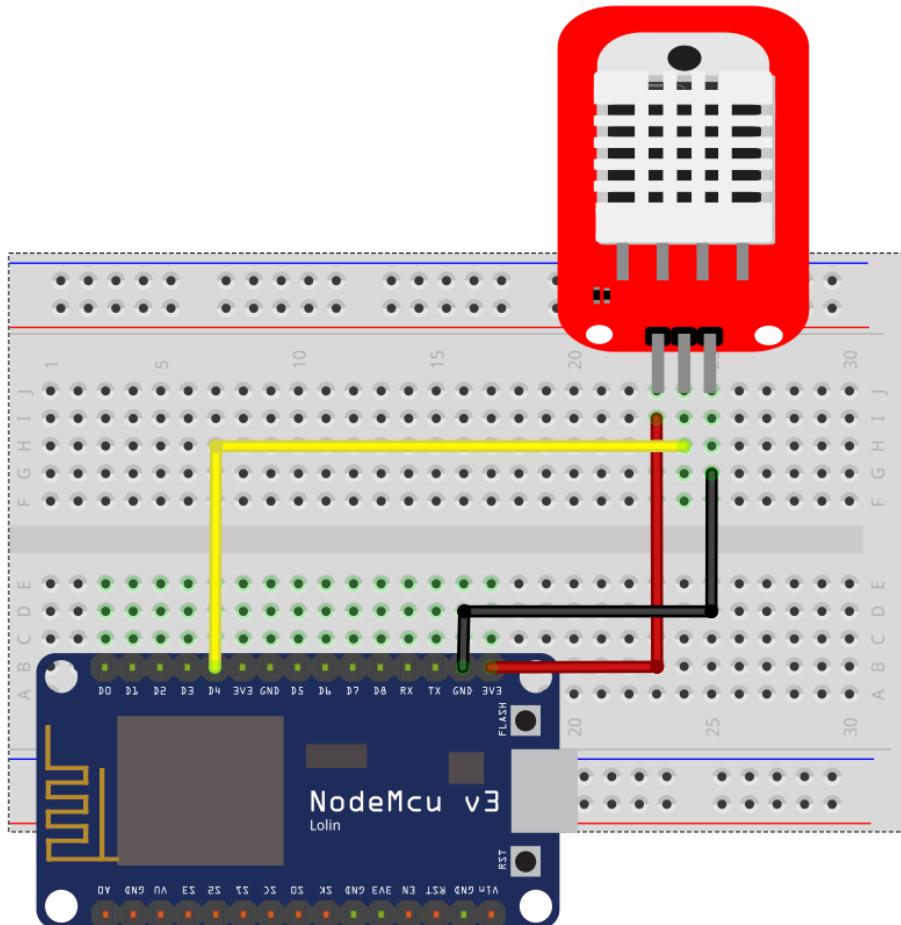
Temperature Timeline (V0)

Humidity Timeline (V1)

Region: sgp1 Privacy Policy

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 //ใช้ประกาศว่าจะใช้ 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

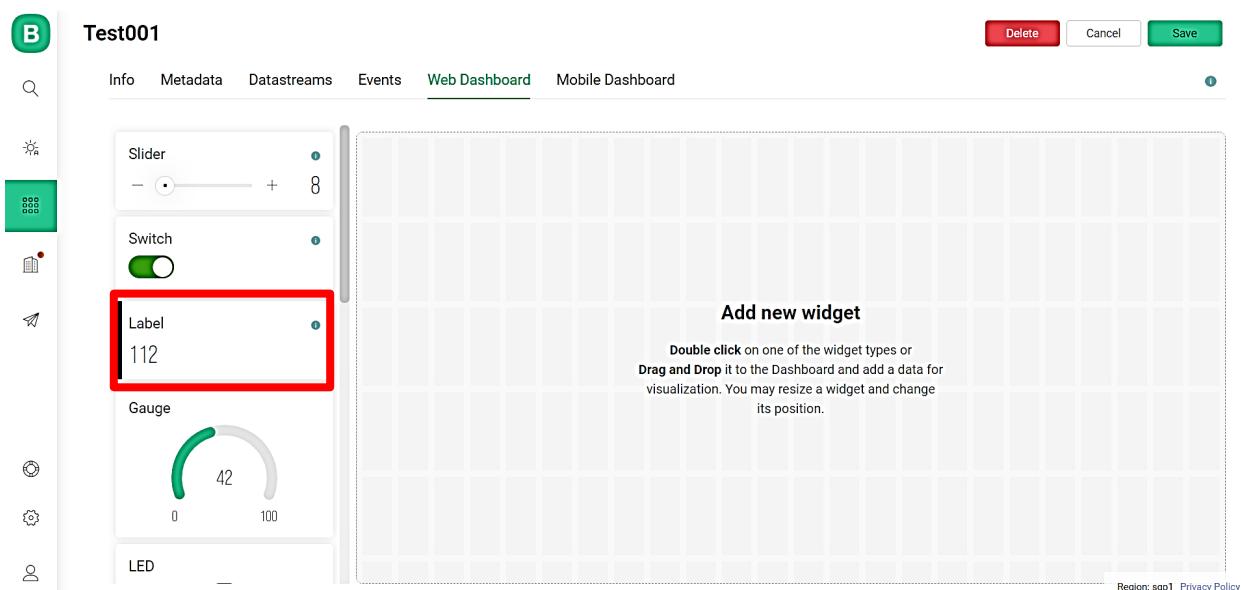
Workshop 15 Blynk and All sensor

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 นำมาระบบที่ Dashboard



2 เลือก switch widget นำมาระบบหน้า Dashboard

The screenshot shows the Grafana dashboard editor interface. The title bar says "Test001". The top navigation bar includes "Info", "Metadata", "Datastreams", "Events", "Web Dashboard" (which is underlined in green), and "Mobile Dashboard". On the right, there are "Delete", "Cancel", and "Save" buttons. The bottom right corner shows "Region: sgp1 Privacy Policy". The left sidebar has icons for search, sun, grid, list, arrow, gear, and user. The main area displays a dashboard with several widgets: a Slider (value 8), a Switch (highlighted with a red box), a Label (value 112), a Gauge (value 42), and an LED.

3 เลือก chart widget นำมาระบบหน้า Dashboard

This screenshot shows the same Grafana dashboard editor as the previous one, but the chart widget in the left sidebar is now selected and highlighted with a red box. The dashboard itself contains a Gauge (value 42), three Labels (values 112, --, and --), and a Map.

4 กดเพื่อตั้งค่าการรับส่งข้อมูลของ Label แรก

5 ตั้งชื่อ Label เป็น Temperature

Label settings ⓘ

6 กด Create Datastream แล้วเลือก Virtual Pin

Label settings ⓘ

TITLE
Temperature

Datastream
You have no datastreams to select

+ Create Datastream
Virtual Pin

Digital
Analog
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

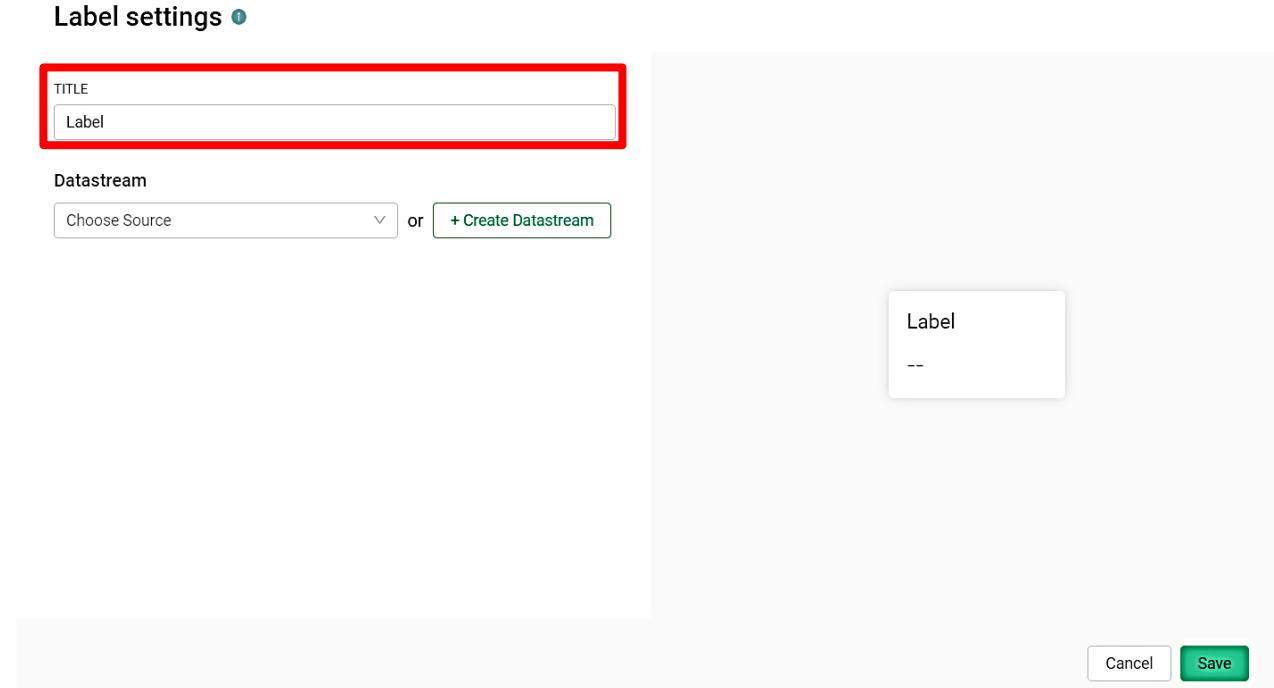
Show level

Temperature

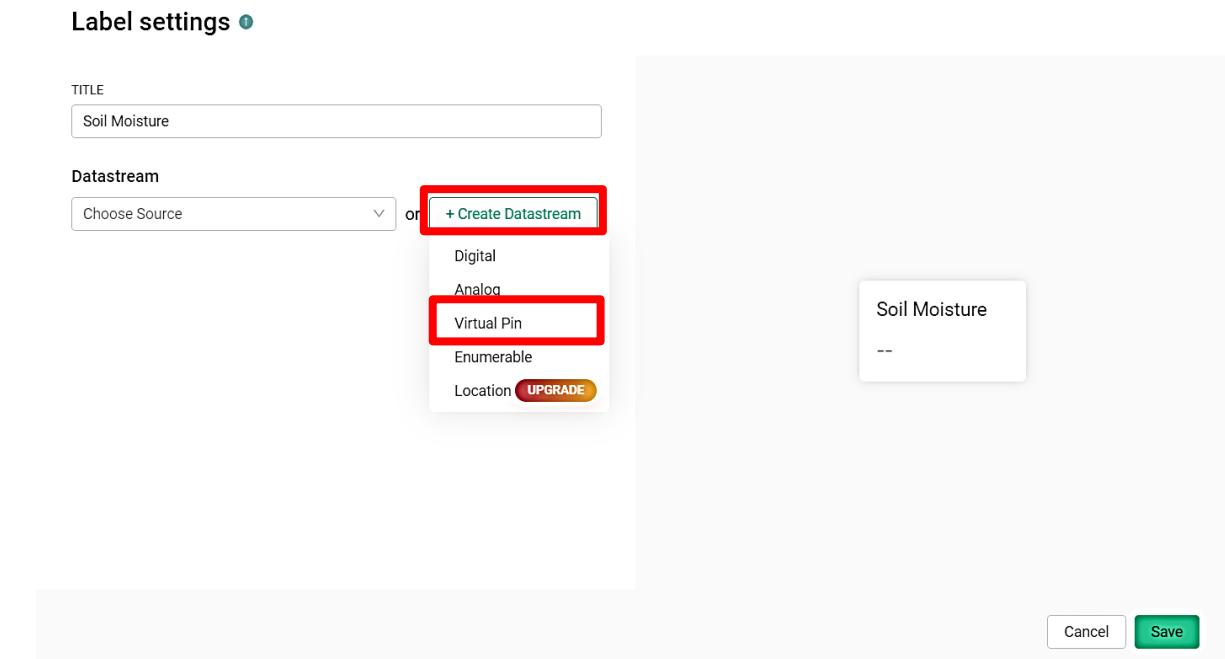
--

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

Save

15 ตั้งค่า Label ตัวที่สาม ตั้งชื่อ Label Distance

Label settings ⓘ

TITLE
Label

Datastream

Choose Source or

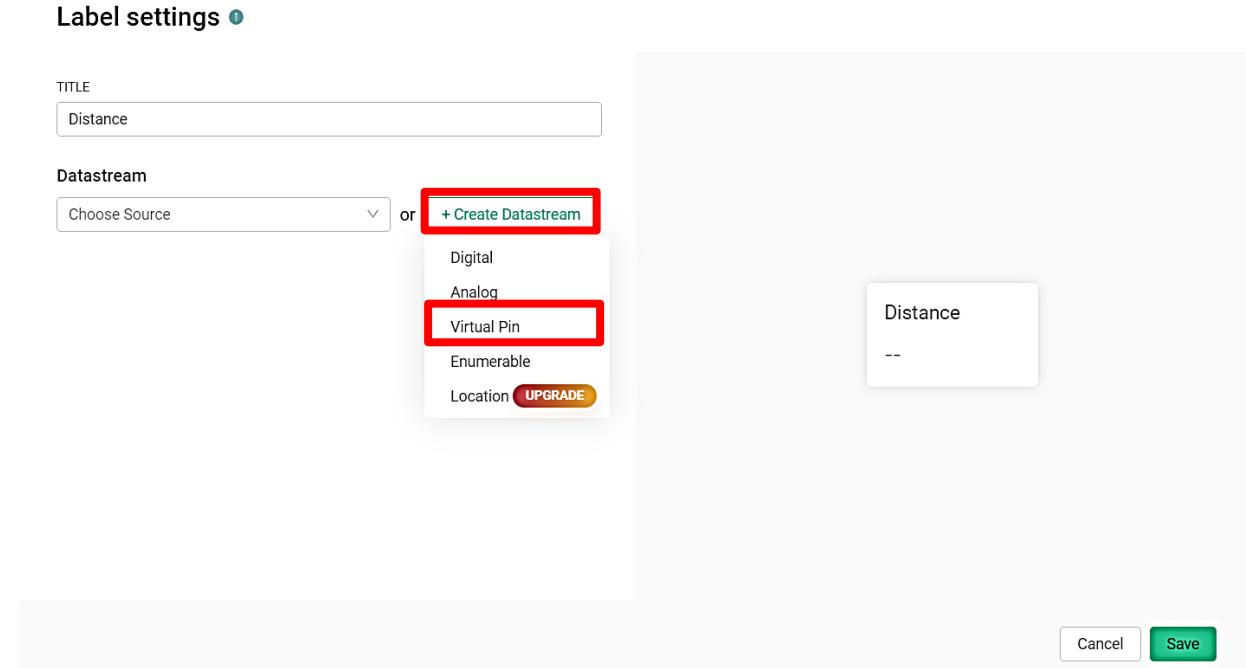
Label

--

Cancel

Save

16 กด Create Datastream และเลือก Virtual Pin



17 ให้ตั้งค่า PIN เป็น V2 , ตั้งค่า Units เป็น Centimeter

Datastream

Virtual Pin Datastream

	Distance	Distance	
PIN	V2	DATA TYPE	Double
UNITS	Centimeter		

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

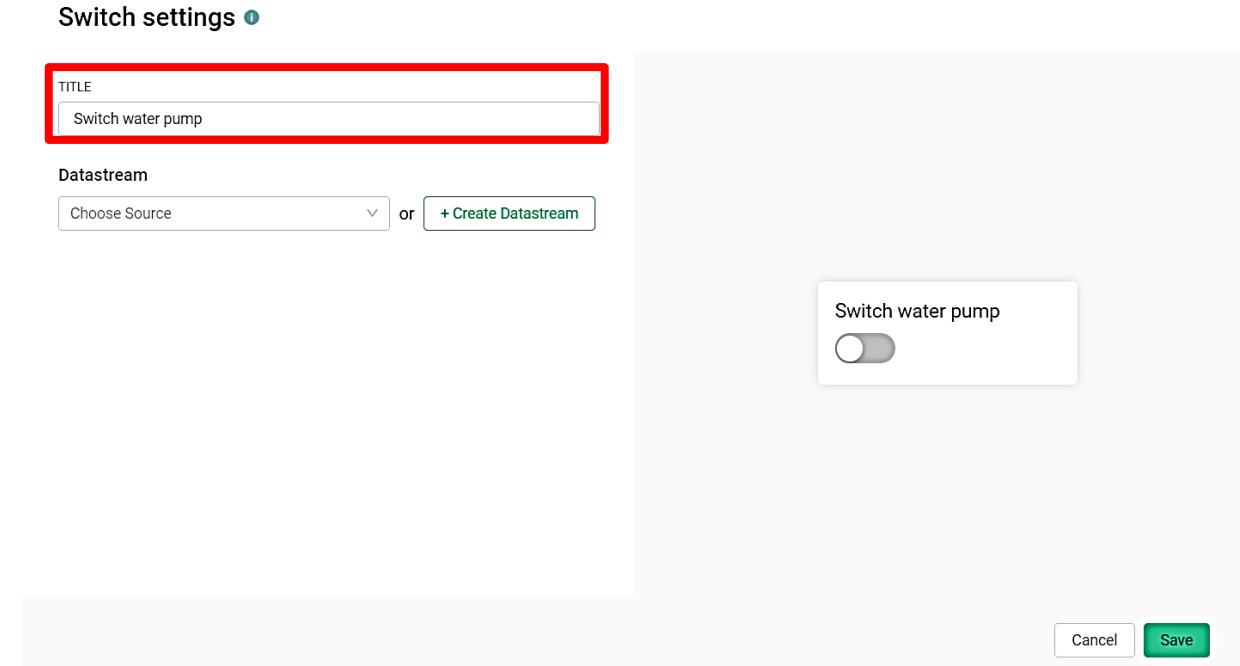
WIDGET BACKGROUND
 Change color based on value

LEVEL
 Show level

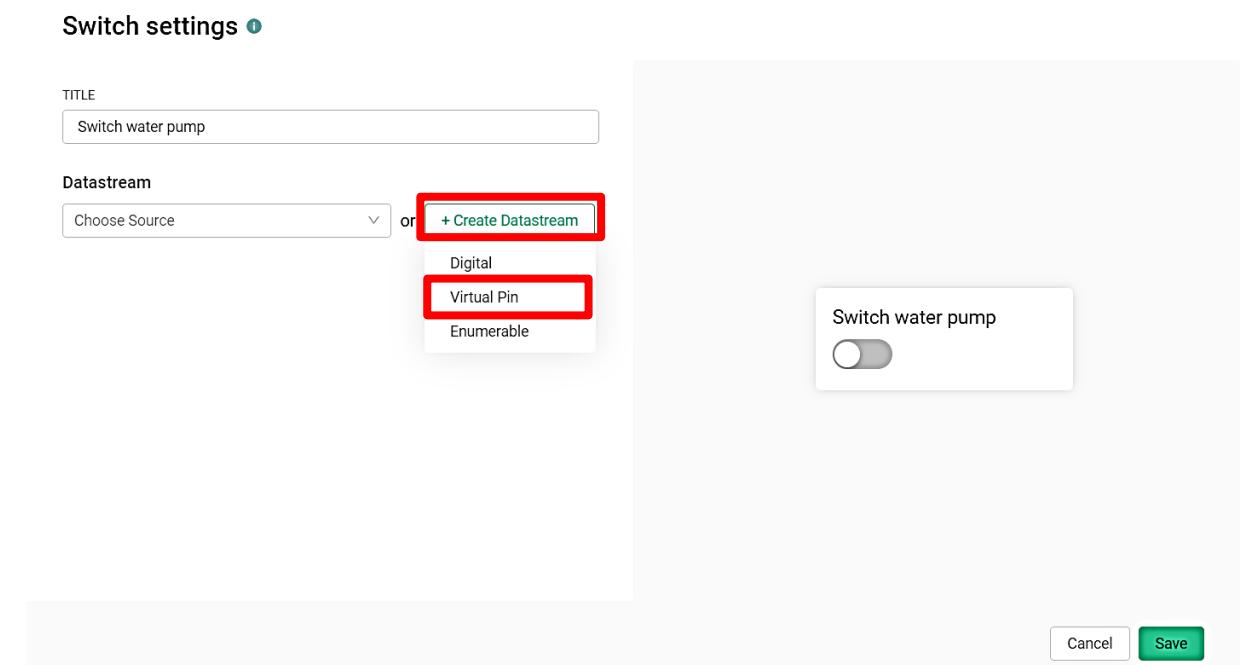
Distance

--

20 ตั้งค่า Switch ตั้งชื่อว่า Switch water pump



21 กด Create Datastream แล้วเลือก 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	<input type="text" value="Switch water pump"/>
Datastream	
Switch water pump (V3) 	
ON VALUE	OFF VALUE
1	0 
<input checked="" type="checkbox"/> Show on/off labels <input checked="" type="checkbox"/> Hide widget name	
 <input style="border: 2px solid red;" type="button" value="Save"/>	

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
Line

CHART COLOR
Green

Show Y-axis
 Autoscale

MIN 0 MAX 100

Temperature Timeline (V0)

Cancel Save

26 กด save เพื่อบันทึกหน้า dashboard

B Test001

Info Metadata Datastreams Events Web Dashboard Mobile Dashboard ⚙️

Delete Cancel Save

Slider 8

Switch

Label 112

Gauge 42

LED

Temperature (V0) 49 °C

Switch wate... (V3)

Soil Moisture (V1) 67%

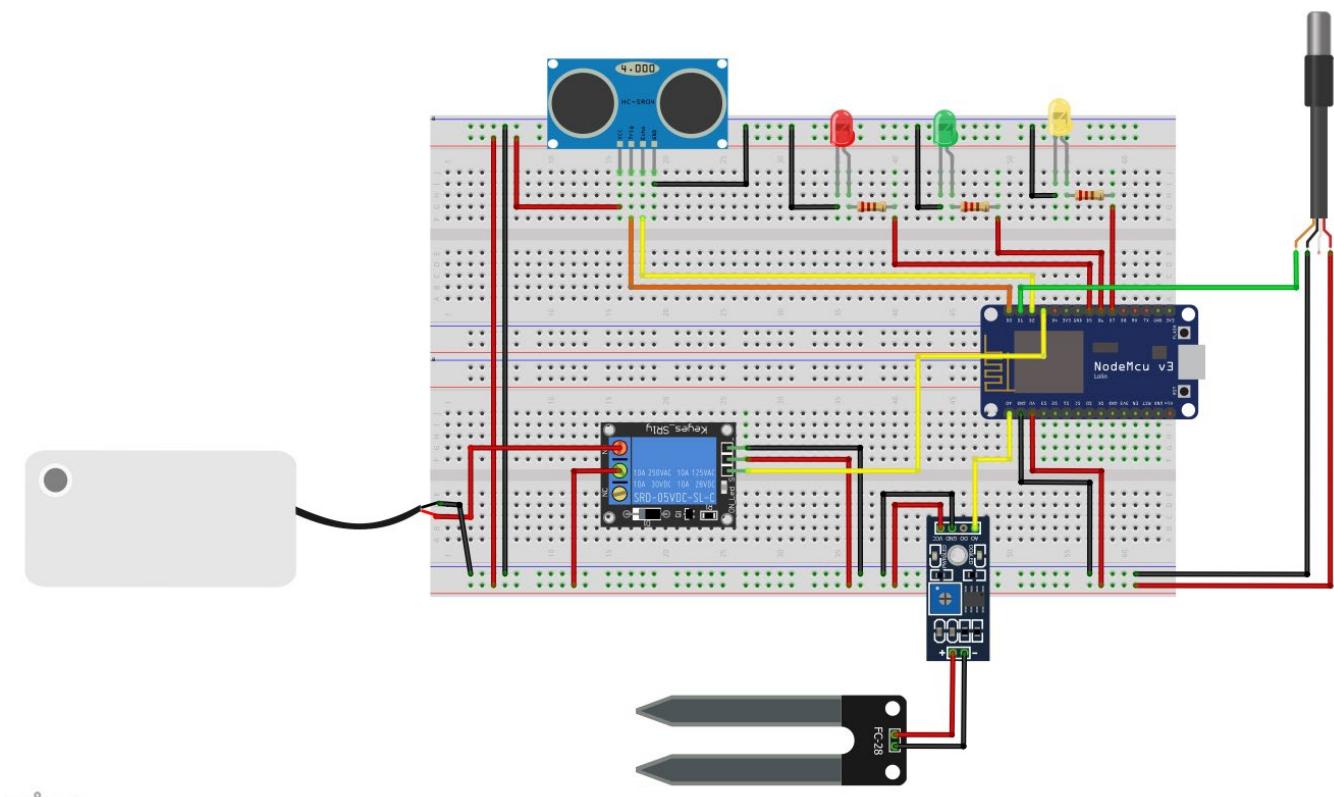
Distance (V2) 241 cm

Temperature Timeline (V0)

Region: sgp1 Privacy Policy

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



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 ที่ใช้ในการกดปุ่มบน 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() {
    //อ่านค่าระยะทางจาก 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