

เอกสารประกอบการอบรม

เรื่องการควบคุมและแสดงผลอุปกรณ์ IoT

Internet of things

1. IoT คืออะไร

IoT หรือ Internet of thing คือ สิ่งของต่างๆ ที่ถูกเชื่อมโยงถึงกันด้วยอินเทอร์เน็ต ซึ่งสิ่งของที่จะนำมาใช้งานก็อาจจะเป็นอะไรก็ได้ ตัวอย่างเช่น โทรศัพท์มือถือ หลอดไฟ ทีวี ปั๊มน้ำ พัดลม เครื่องปรับอากาศ เป็นต้น สิ่งต่าง ๆเหล่านี้ก็จะถูกฝังระบบควบคุมที่จะคอยตอบสนองต่อการสั่งการ และเซ็นเซอร์ที่จะช่วยให้รับรู้ถึงสภาพแวดล้อมต่าง ๆที่อยู่รอบ ๆตัว อีกทั้งยังสามารถเชื่อมต่อกับเครื่อข่าย และให้ติดต่อสื่อสารกันเองผ่านระบบไร้สายได้ด้วย อีกทั้งเรายังสามารถใช้คอมพิวเตอร์หรือสมาร์ทโฟนของเราเชื่อมต่อกับอุปกรณ์ต่าง ๆผ่านทางอินเทอร์เน็ต เพื่อเข้าควบคุม สั่งการอุปกรณ์ต่าง ๆที่เราต้องการใช้งาน หรือจะใช้ในการอ่านค่าต่าง ๆจาก sensor โดยอาศัยการส่งข้อมูลผ่านทาง cloud service เชิร์ฟเวอร์ต่าง 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 เราจะสามารถผลิตได้ตามความต้องการของลูกค้าในทันท่วงที่ ตามจำนวนการสั่ง หรือที่เรียกว่า Just In Time (JIT) การผลิตแบบนี้จะช่วยลดต้นทุนจนในการผลิตที่เกินมาได้อย่างมีนัยสำคัญ

1.1.4 อำนวยความสะดวก ให้กับผู้ผลิตในการสร้างสรรค์นวัตกรรม

สิ่งหนึ่งที่มนุษย์ทำได้ดีกว่าเทคโนโลยีคือ “ความคิดสร้างสรรค์” การให้เทคโนโลยีทำงานด้าน Routine แทนเรา จะทำให้เรามีเวลาในการทำงานสร้างสรรค์ งานนวัตกรรม ทำสิ่งที่ควรทำมากยิ่งขึ้น เพราะแท้ที่จริงแล้ว มนุษย์มีศักยภาพที่ซ่อนอยู่มากกว่าจะทำเพียงแค่งาน Routine เท่านั้น การให้เทคโนโลยีทำงานแทนและโหยก 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 มีหนังบการเงินที่สวยงาม (โดยไม่ต้องตกแต่งตัวเลข) เป็นที่น่าจับตามองของนักลงทุนหรือ หุ้นส่วนทางธุรกิจต่าง ๆ ที่จะเข้ามาสนับสนุนธุรกิจ

2. การเตรียมอุปกรณ์และโปรแกรม Arduino IDE

2.1 NodeMCU ESP8266

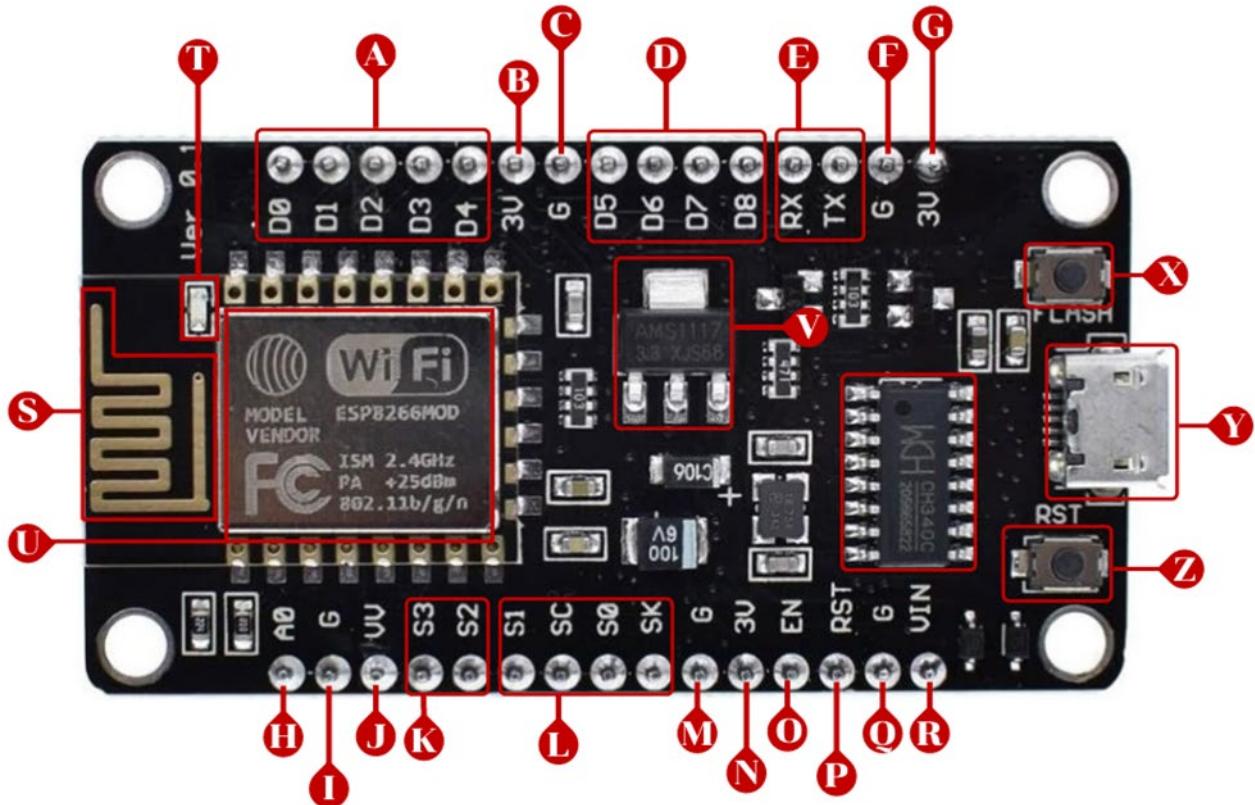
NodeMCU คือ บอร์ดคอนโทรลเลอร์ที่มีลักษณะการทำงานตามคำสั่งภาษา C คล้าย Arduino แต่มีลักษณะพิเศษกว่าตรงที่ สามารถเชื่อมต่อกับ WiFi ได้ ในการเขียนโปรแกรมสามารถเขียนผ่าน Arduino IDE ได้

บอร์ดของ NodeMCU ประกอบไปด้วย ESP8266 พร้อมอุปกรณ์อำนวยความสะดวกต่างๆ เช่น พอร์ต micro USB สำหรับจ่ายไฟ/อัปโหลดโปรแกรม, ชิพสำหรับอัปโหลดโปรแกรมผ่านสาย USB เป็นต้น



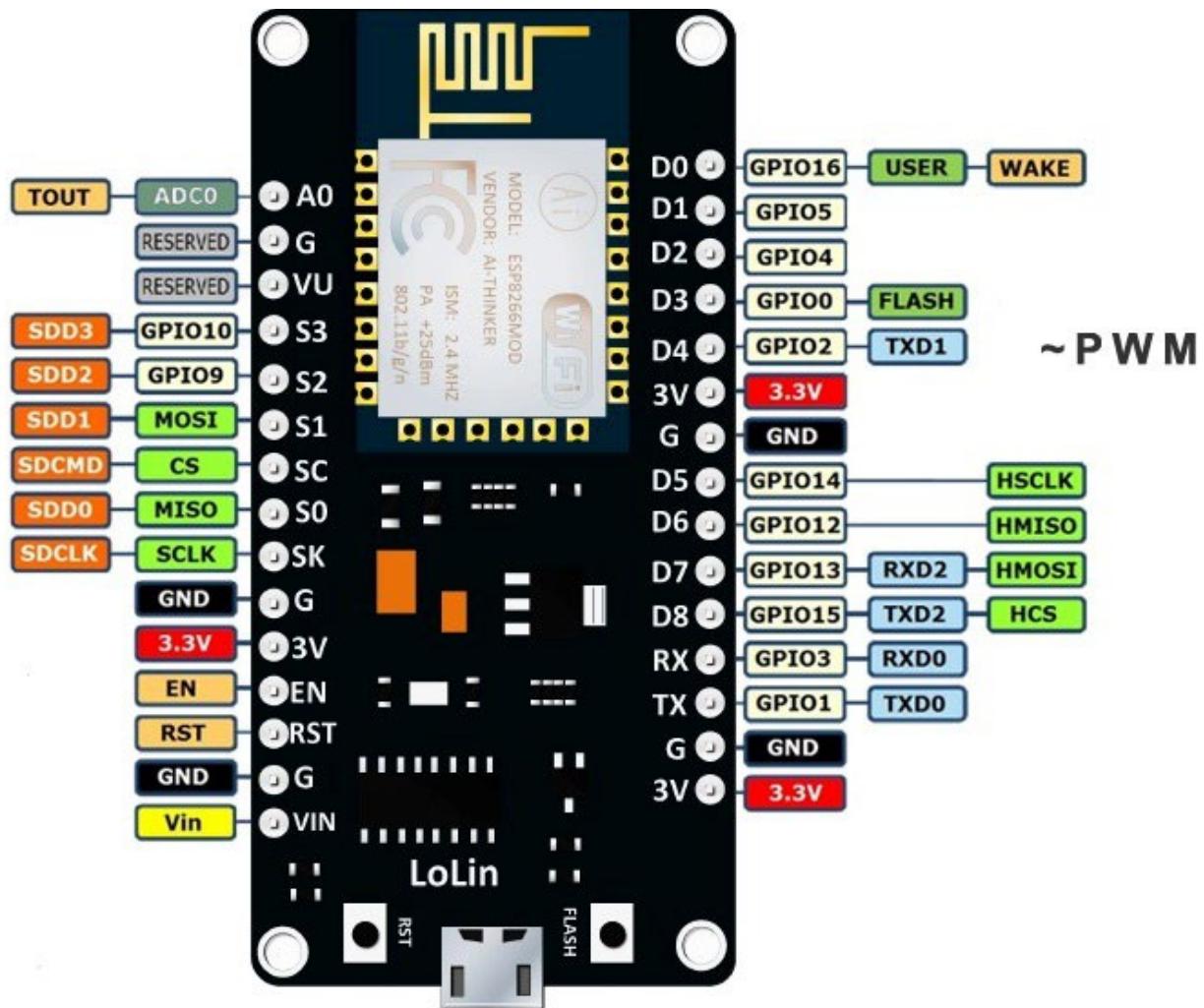
NodeMCU ESP8266 V3 ESP12E

2.2 ส่วนประกอบ NodeMCU ESP8266



- D0 - D4 หรือ GPIO เป็นขาดิจิตอลที่สามารถกำหนดให้ขาที่ต้องการเป็น Input หรือ Output ก็ได้
- 3V เป็นขาที่มีแรงดันไฟ 3.3 V
- G เป็นขา Ground
- D5 - D8 เป็นทั้งขา GPIO และขาที่ใช้ติดต่อสื่อสารกับโมดูลอินพุตอินเตอร์เฟส HSPI
- RX/TX เป็นขา GPIO และขาที่ใช้รับส่งข้อมูลแบบ Serial กับอุปกรณ์อื่น
- G เป็นขา Ground
- 3V เป็นขาที่มีแรงดันไฟ 3.3 V
- A0 หรือ ADC เป็นขา Input ที่ใช้อ่านค่าจาก sensor ที่ส่งค่ามาเป็น analog มาแปลงเป็น digital
- G เป็นขา Ground
- VU หรือ V USB เป็นขาที่แรงดันไฟ 5 V
- 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 ซึ่งไฟที่จ่ายให้คราวนีนาดไม่เกิน 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 ใช้ควบคุมและรักษากระแสตัวแปลงดันไฟให้คงที่ที่ 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 และอัพโหลดข้อมูล
- 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

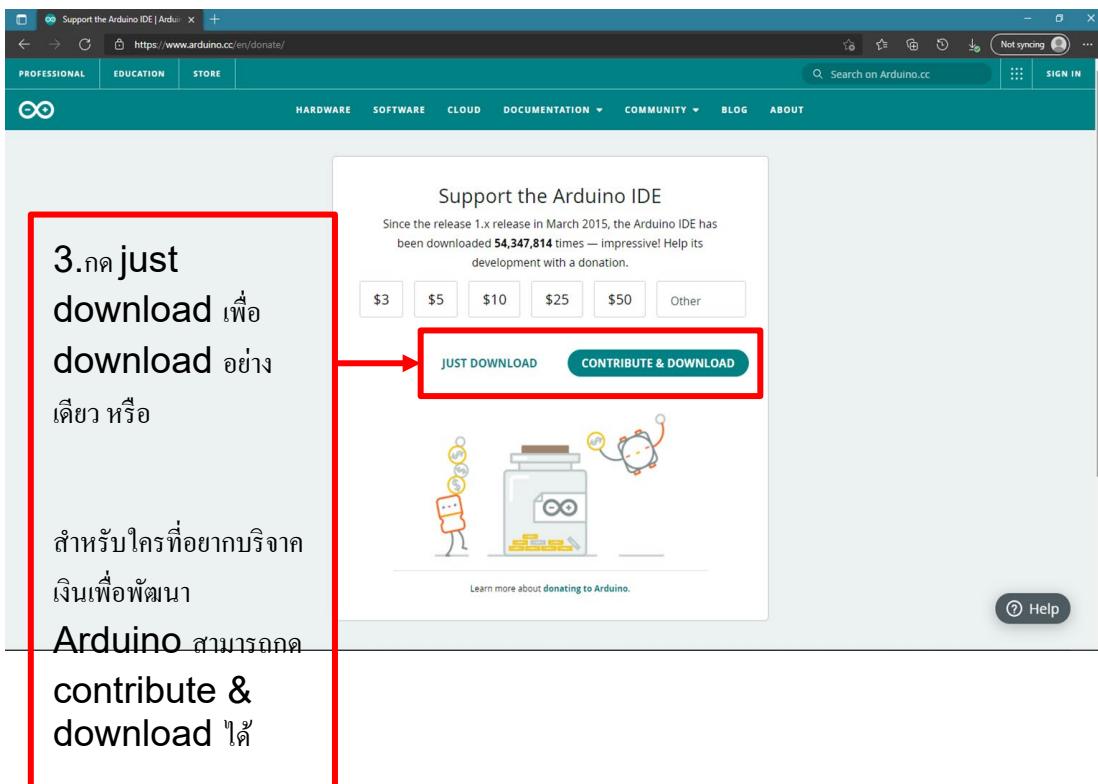


Legacy IDE (1.8.X)

A screenshot of the Arduino IDE 1.8.19 download page. At the top left is a logo of two interlocking circles. Next to it is the text "Arduino IDE 1.8.19". Below this is a brief description: "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." It also includes a link to "Arduino IDE 1.x documentation" for installation instructions. On the right side, there is a "DOWNLOAD OPTIONS" section with a red box around the "Windows" heading and a red arrow pointing to it from the text above. The "Windows" section shows "Win 7 and newer" and a "Get" button. Other options listed include "Windows app", "Linux 32 bits", "Linux 64 bits", "Linux ARM 32 bits", "Linux ARM 64 bits", and "Mac OS X 10.10 or newer". There are also links for "Release Notes" and "Checksums (sha512)".

2. จากนั้นเลื่อนลงมา และให้เลือก OS

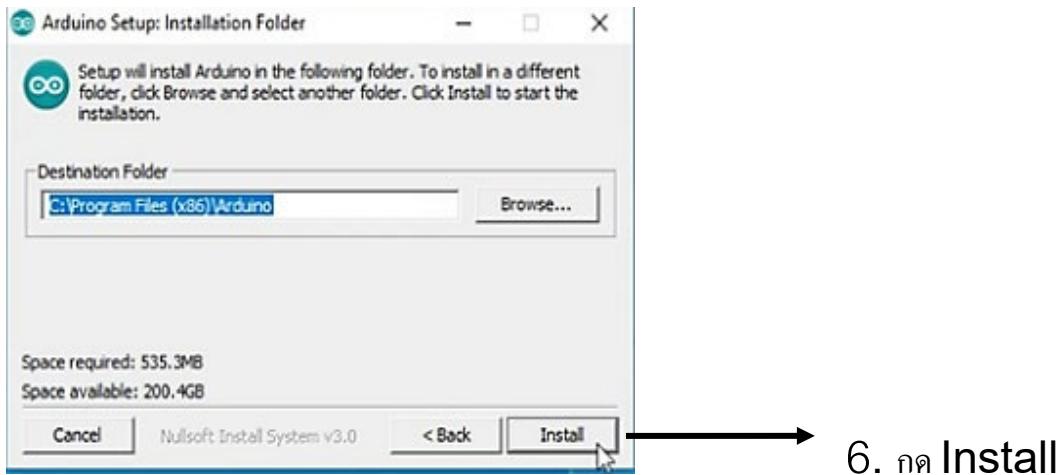
ที่ใช้อยู่ กรณี Windows ให้เลือกเวอร์ชัน 1.8.19 ซึ่งเป็นเวอร์ชันที่มีเสถียรภาพ



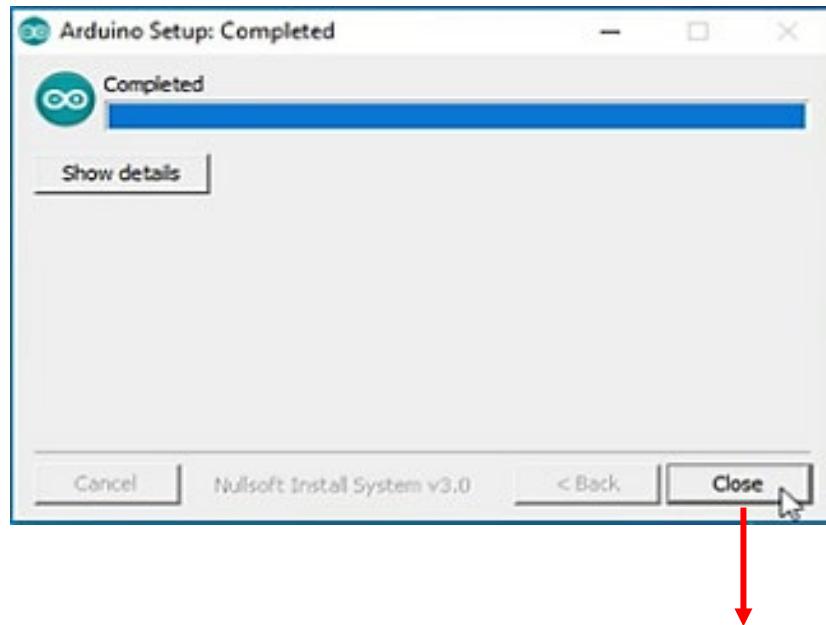
4. กด I Agree เพื่อดำเนินการต่อ

5. จะมีหน้าต่างให้เลือกตัวเลือก ให้กด

Next ต่อ โดยไม่ต้องแก้ไขตัวเลือกใดๆ



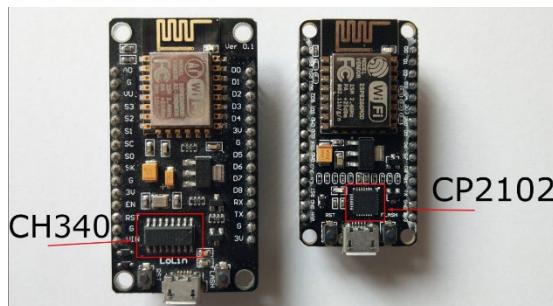
7. ในระหว่างการติดตั้ง จะมีการให้ติดตั้ง driver ต่างๆ ให้ทำการกดติดตั้ง driver ทั้งหมด



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

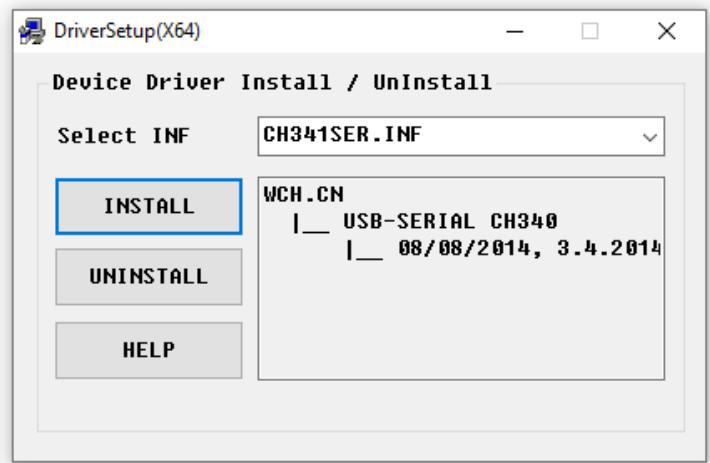
Electronic Components Online Shop Tools, Docs, Downloads Cu

Windows

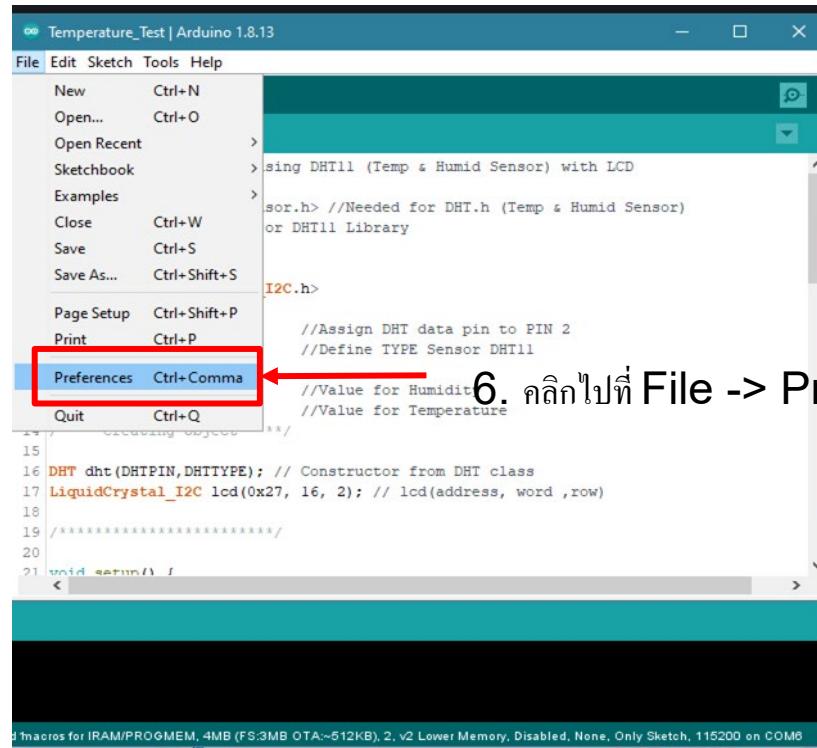
(Manufacturer's Chinese Info Link)

- 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.

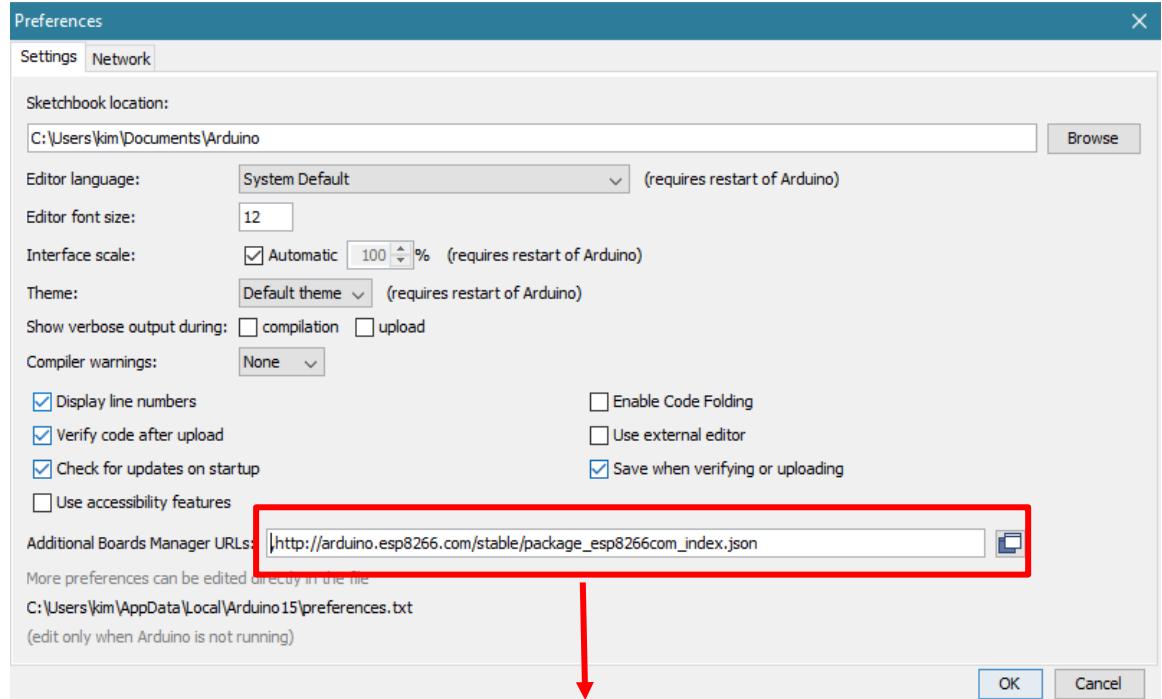
2. ให้ทำการแตกไฟล์และกด INSTALL



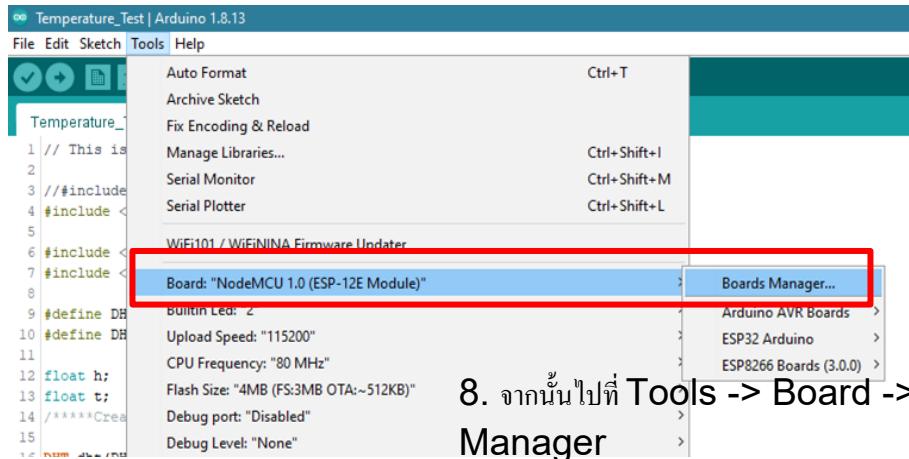
3. หลังจากติดตั้ง Driver เสร็จแล้ว ให้เปิดโปรแกรม Arduino IDE ขึ้นมา



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

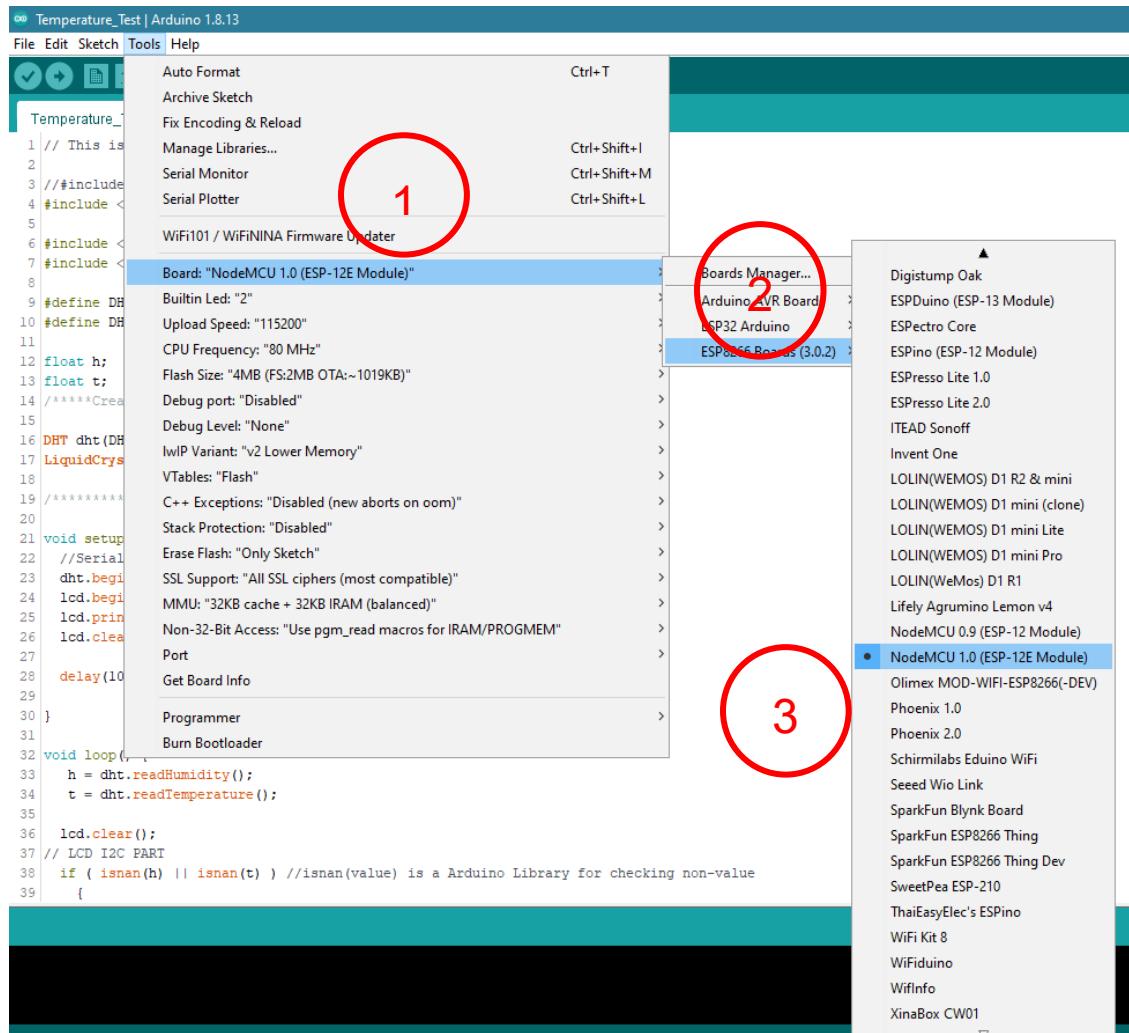
9. พิมพ์ว่า esp8266



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

การประกาศตัวแปรเป็นการแจ้งให้คอมไพล์อร์รู้ว่าต้องการจดพื้นที่หน่วยความจำส่วนหนึ่งไว้สำหรับใช้เก็บข้อมูลโดยกำหนดขนาดพื้นที่และวิธีดำเนินการตามชนิดข้อมูล 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

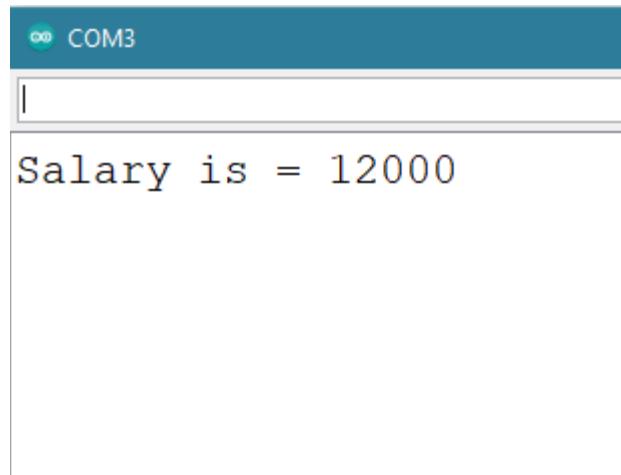
2. ตัวแปรโลคอล (Local Variable)

เป็นตัวแปรที่จะประกาศอยู่ภายในบล็อกของกลุ่มคำสั่ง (คำสั่งที่อยู่ระหว่างเครื่องหมาย {}) หรือฟังก์ชัน มีขอบเขตเข้าถึงได้เฉพาะคำสั่งที่อยู่ในพื้นที่คำสั่งหรือฟังก์ชัน เดียวกันเท่านั้น

ตัวอย่างการประมวลผลแบบทวิภาคี

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

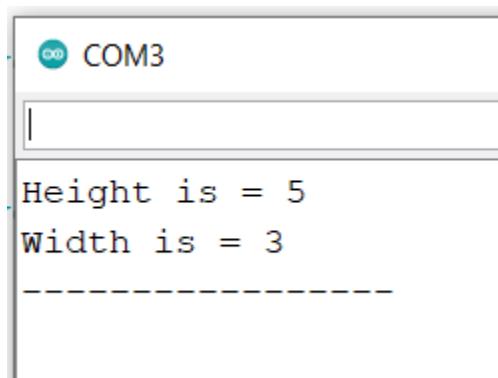
OUTPUT :



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 ชื่อนี้ในตำแหน่งใดๆ ให้แทนที่ด้วยค่าคงที่นั้นแล้วค่อยประมวลผลโปรแกรม

Syntax

```
#define identifier value
```

identifier	ชื่อที่ต้องนิยามเป็นค่าคงที่
value	ค่าคงที่ที่ต้องการกำหนด

ตัวอย่าง

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

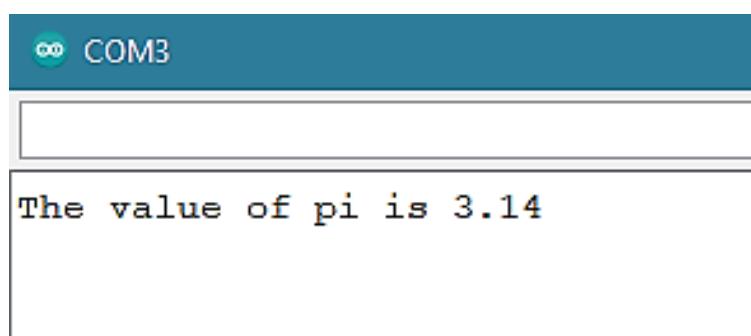
ตัวอย่าง

```
#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) เป็นสัญลักษณ์บอกให้คอมพิวเตอร์ดำเนินการอย่างใดอย่างหนึ่งกับ ตัวแปรและค่าคงที่ต่างๆ ซึ่งใน C++ มีตัวดำเนินการหลายประเภทด้วยกัน เช่น ตัวดำเนินการทางคณิตศาสตร์ ตัวดำเนินการเปรียบเทียบ ตัวดำเนินการเชิงตรรกะ 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 โดยข้อมูลที่นำมาเปรียบเทียบท้องเป็นชนิดเดียวกัน

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
```

จากตัวอย่างເອົາຕຸພູດຈະໄດ້ວ່າການແສດງຜລລັບຂອງຕົວດຳເນີນການເປີຍບໍ່ເທິຍບມື່ຢູ່ 2 ດ່າວວິກ່າວ່າມີຄ່າເປັນ false ແລະ 1 ທີ່ມີຄ່າເປັນ true

4.9 ตัวดำเนินการทางตรรกะ

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

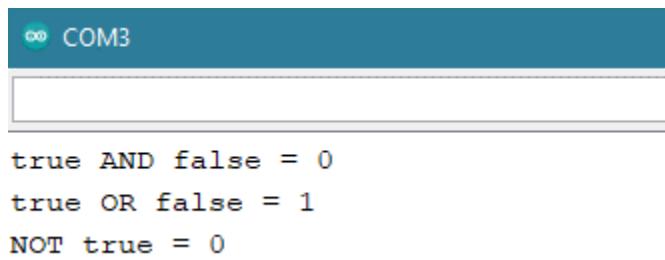
ตัวดำเนินการ	ความหมาย	ตัวอย่างและความหมาย
&&	AND	A && B เป็นจริงเมื่อตัวแปร A และ B เป็นจริงทั้งคู่
	OR	A B เป็นจริงเมื่อตัวแปร A หรือ B เป็นจริง
!	NOT	!A ให้ผลตรงข้ามกับ A ถ้า A เป็นจริงผลลัพธ์เป็นเท็จ

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
```

จากตัวอย่างเราตั้งพุตจะได้ว่าการแสดงผลลัพธ์ของดำเนินเชิงตรรกะมีอยู่ 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

```

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 เจื่อนไขไม่ตรง
Output เจื่อนไขตรง

```

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

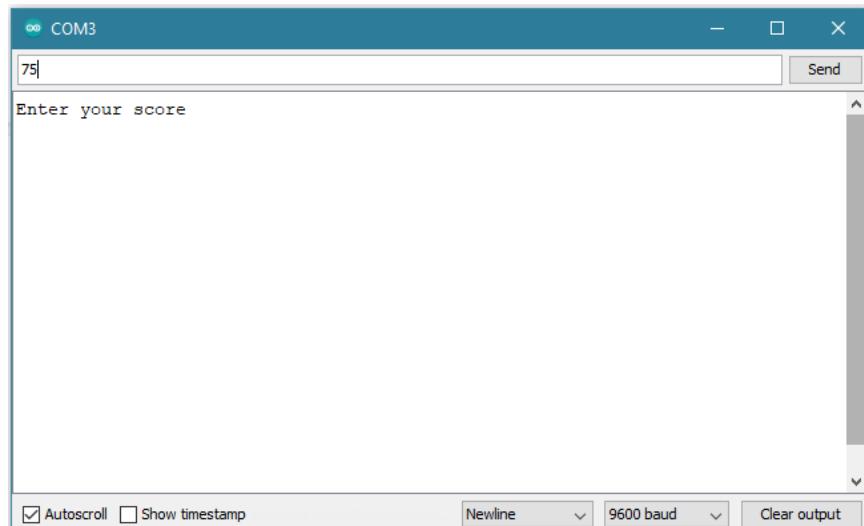
```

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");
        }
    }
}

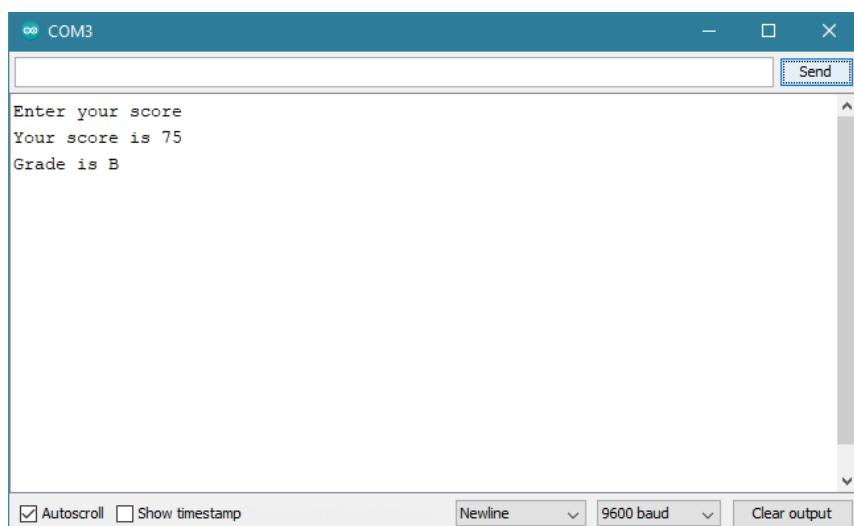
```

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



ตัวโปรแกรมจะนำไปปรับเทียบ 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]
}
```

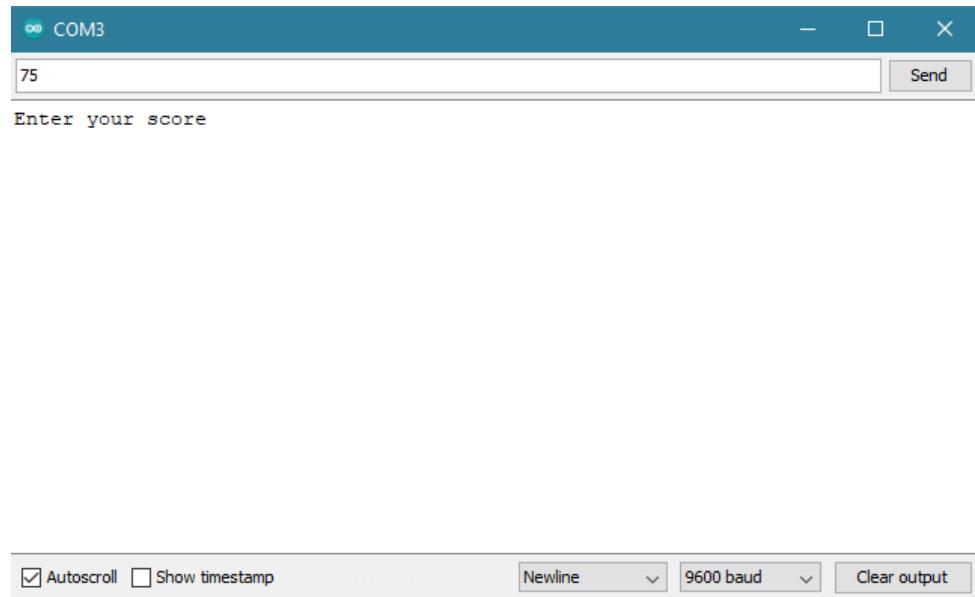
expression:	นิพจน์ที่นำมาเปรียบเทียบกับค่าคงที่ใน case จะอยู่ในรูปแบบของตัวแปร คลาส หรือฟังก์ชัน โดยค่าได้ต้องเป็นชนิดข้อมูล
const1, const2,...:	ค่าคงที่ใน case ต้องเป็นข้อมูลเดียวกับนิพจน์ และค่าที่ในแต่ละ case ต้องไม่ซ้ำกัน เมื่อค่าของนิพจน์ตรงกับค่าของ case ใด โปรแกรมจะทำงานในสเตตเมนต์ที่วางไว้หลังเครื่องหมาย : ของ case นั้น
break:	ตัวโปรแกรมจะໄล่เปรียบเทียบ case แต่ละตัว หากตรงกับ case ไหนโปรแกรมทำงานตามสเตตเมนต์ที่อยู่หลังเครื่องหมาย : ของ case นั้น เมื่อเจอ break โปรแกรมจะออกจาก การเลือกทำแต่หากไม่มี break ปิดท้าย โปรแกรมจะทำการเปรียบเทียบ case ต่อไปเรื่อยๆ จนกว่าจะเจอ break
default:	หากเปรียบเทียบค่า�ิพจน์แล้วไม่ตรงกับ case ใดเลย ตัว โปรแกรมจะมาทำสเตตเมนต์ของ default จนจบโดยไม่ต้องเขียนคำสั่ง break

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

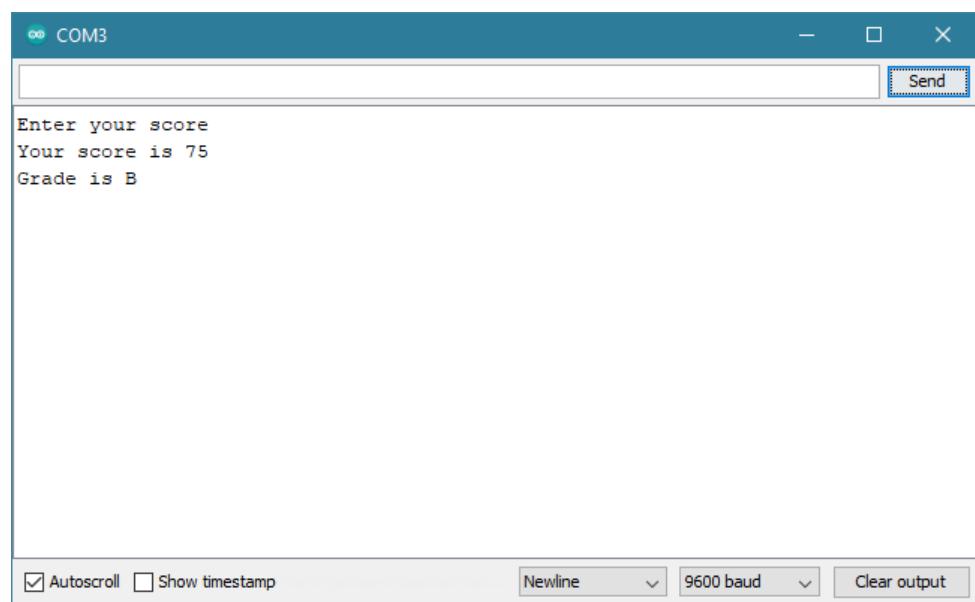
ตัวอย่าง

```
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



ตัวโปรแกรมจะนำไปเปรียบเทียบ และแสดงเกรด



4.16 การวนทำซ้ำ (Loop)

หากเราต้องการให้มีการทำงานกับคำสั่งบางอย่างซ้ำหลายครั้ง ก็สามารถใช้คำสั่งวนลูปซ้ำได้โดย ทำการกำหนดเงื่อนไข โดยตัวโปรแกรมจะทำงานชุดคำสั่งนั้นไปเรื่อยๆ 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 การวนทําซ้ำด้วยคำสั่ง do...while

while เป็นการให้ทำงานวนซ้ำ โดยจะทำคำสั่งที่เขียนไว้ก่อน 1 ครั้ง และจึงตรวจสอบเงื่อนไข หากเงื่อนไขยังเป็นจริงจะทำการวนซ้ำต่อ หากไม่ตรงเงื่อนไขจะออกจาก循环

Syntax

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

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

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

ตัวอย่าง

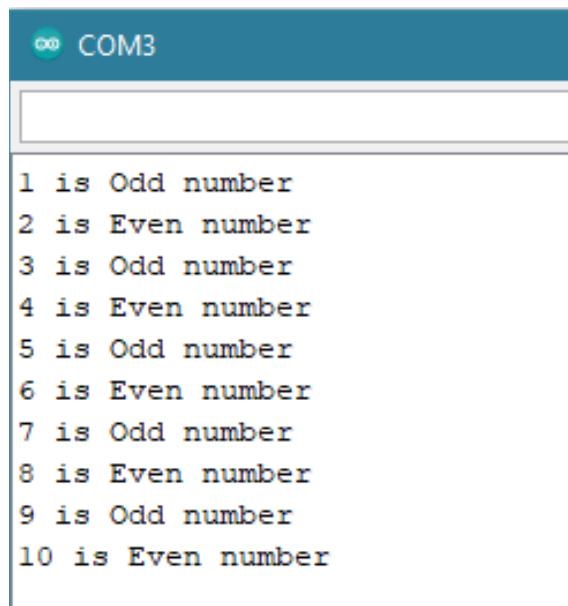
```

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 ต้อง กำหนดการเพิ่มหรือลดค่าที่อยู่ภาย 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 คอมมา (,)
function body	เป็นกลุ่มคำสั่งหรือ statement ที่กำหนดว่าให้ทำงานอะไรมาก็ได้โดยอาจมีการส่งผลลัพธ์กลับไปยังจุดเรียกฟังก์ชันได้

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

เป็นการบอกให้คอมไพล์러ทราบเกี่ยวกับชื่อ 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 อาร์กิวเม้นต์ (argument) ไปยังพารามิเตอร์ของฟังก์ชัน หรืออาจไม่มีการส่งข้อมูลไป ก็ได้เช่นกัน จากนั้นก็นำไปประมวลผลและส่งผลลัพธ์คืนมา y ังฟังก์ชัน 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 ดังนั้นการนำไปใช้งานจะ ต้องประกาศตัวแปรเป็น ออบเจกต์ การประกาศสร้างคลาสจะเริ่มต้นด้วยคีย์เวิร์ด class ตามด้วยชื่อของ คลาส และ ตัวของคลาส (Body) ที่อยู่ภายใต้เครื่องหมาย { } ส่วนรายการของสมาชิกจะอยู่ภายใต้ตัวของ คลาส

Syntax

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

Class_name	ชื่อคลาสโดยกำหนดเป็นชื่ออารก์ได้ที่ไม่ซ้ำกับคลาสอื่นและ คีย์เวิร์ดที่ 사용ไว้
Class Access Modifier	<p>เป็นการกำหนดสิทธิ์การเข้าถึงตัวแปรข้อมูลและ Method ซึ่งมี 3 คีย์เวิร์ดได้แก่</p> <ul style="list-style-type: none"> ● private เข้าถึงได้เฉพาะภายในคลาสเดียวเท่านั้น ● protected เข้าถึงเฉพาะภายในคลาสเดียวเท่านั้นหรือ คลาสที่ถูกสืบทอดหรือคลาสลูก ● 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 ออยู่ภายในคลาส โดยเรารสามารถกำหนดการทำงานให้อยู่ภายในคลาส หรือภายนอกคลาสนี้ได้ ดังตัวอย่าง

การกำหนด Method ภายในคลาส

ตัวอย่าง

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

การกำหนด Method ภายในอօกคลาส

ตัวอย่าง

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


```

4.20.3 การสร้างอօบเจ็กต์

การสร้างอօบเจ็กต์จากคลาสมีวิธีเหมือนการประกาศตัวแปรทั่วไป โดยกำหนดให้ชื่อคลาสเป็น ชนิดข้อมูล และตามด้วยอօบเจ็กต์เป็นชื่อของตัวแปร

Syntax

Class_Name Object_Var;

Class_name	ชื่อคลาสที่จะนำมาสร้างอօบเจ็กต์
Object_var	ชื่ออօบเจ็กต์ที่ถูกสร้างด้วยคลาส

ตัวอย่าง

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

4.20.4 การเข้าถึงสมาชิกของคลาส

ออกแบบต์สามารถเข้าถึงและทำงานกับสมาชิกที่เป็นตัวแปรและ Method ในคลาสได้โดยการ พิมพ์ชื่อ ออกแบบต์ตามด้วยเครื่องหมาย . และชื่อของตัวแปรหรือ Method

ตัวอย่าง

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

ตัวอย่างการเขียนโปรแกรมโดยการนำคลาส Box มาใช้คำนวณhabริมาตรและพื้นที่ของรูปทรง สี่เหลี่ยมผ่านออกแบบต์ที่ชื่อ myBox และคำนวณโดยใช้ Method getVol() หาปริมาตร และ getSurf() หาพื้นที่ผิว

ตัวอย่าง

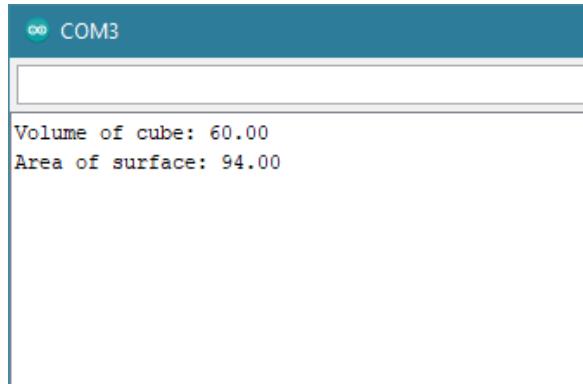
```
class Box{
public:
    double L;
    double W;
    double H;
    double getVol(void); //กำหนดค่าบาร์โค้ดที่ไม่ของเมธอด getVol
    double getSurf(void); //กำหนดค่าบาร์โค้ดที่ไม่ของเมธอด getSurf
};

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

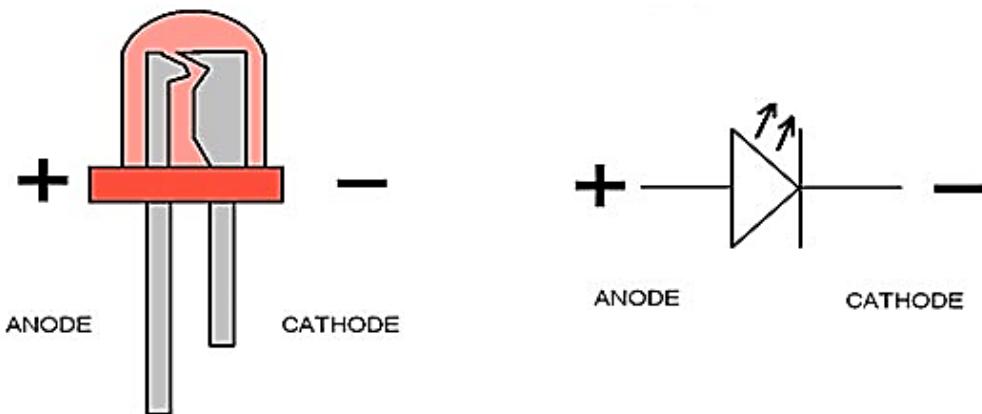
Workshop 1 สัญญาณไฟกระพริบทดสอบไมโครคอนโทรลเลอร์ ESP8266

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

LED คืออะไร

LED ย่อมาจาก Light Emitting Diode คือ “ไดโอดชนิดเปล่งแสง”

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



ประเภทของ LED

1. LED แบบดั้งเดิม

กำลังวัตต์ต่ำน้อย ขนาดหรือรูปร่างหรือสีขึ้นอยู่กับพลาสติกที่ใช้ทำเปลือกหุ้ม ใช้ทำไฟสัญญาณในวงจร



LED แบบดั้งเดิม

2. LED ขนาดเล็กมาก

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



SMD5050 LED Chip



SMD3528 LED Chip

LED ขนาดเล็กมากหรือ SMD LED

3. LED กำลังสูง

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



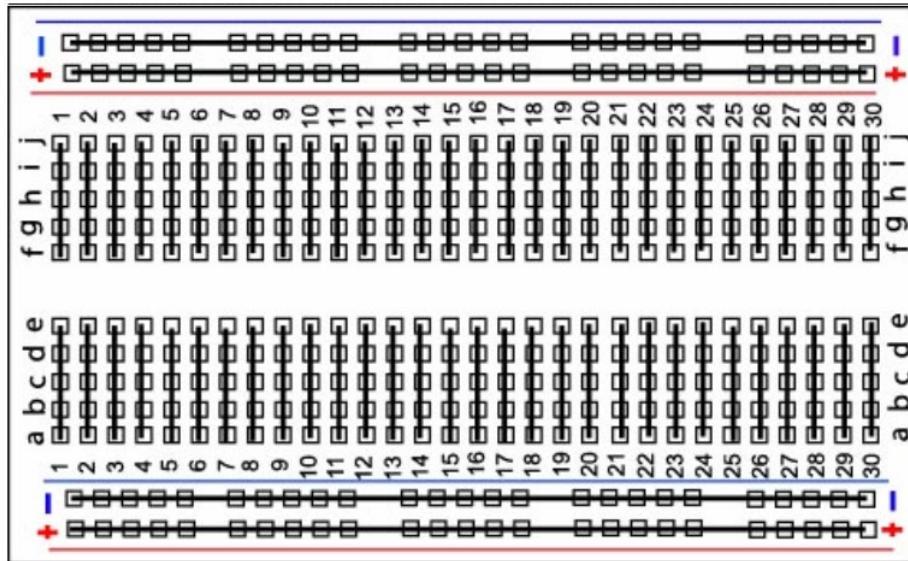
CREE XMA Series COB



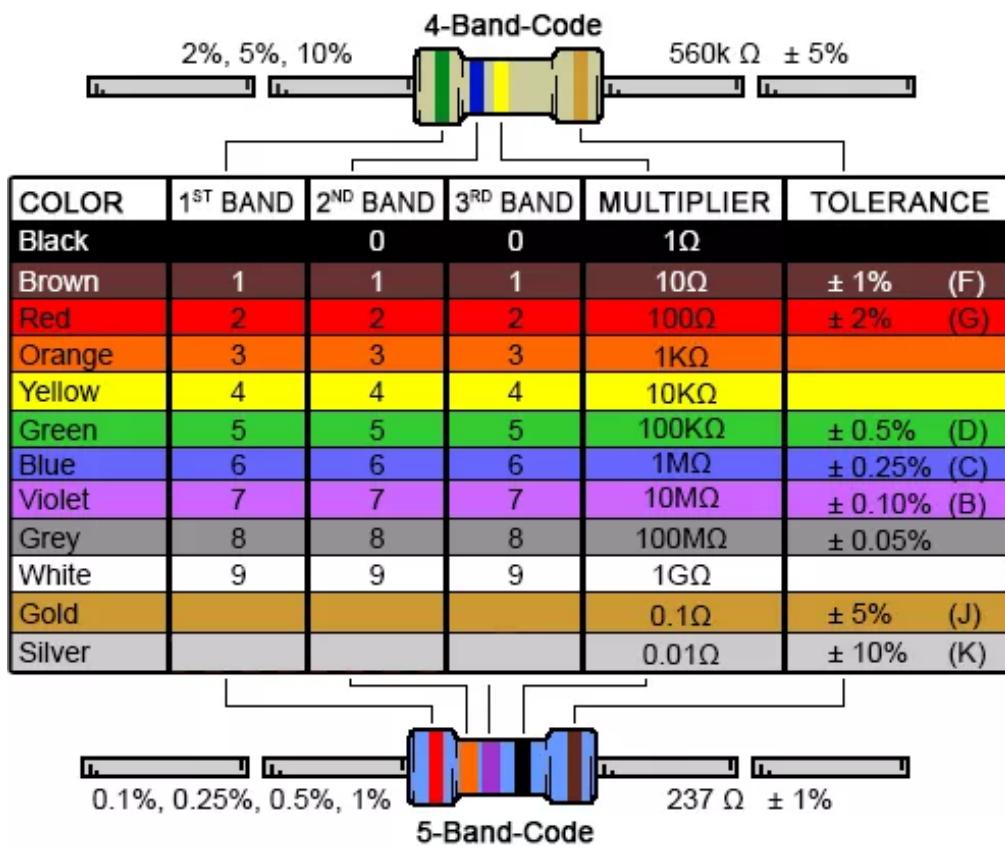
Bridgelux Vero Series COB

LED กำลังสูง หรือ Hi-power LED

การต่อวงจรภายในแผ่นบอร์ดทดลอง (Breadboard)



การอ่านสีตัวต้านทาน



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

จากรูปเป็นตารางสีในการอ่านค่าตัวต้านทาน ตัว R อ่านได้ดังต่อไปนี้

สำหรับแบบ 4 Band (หรือ 4 สี)

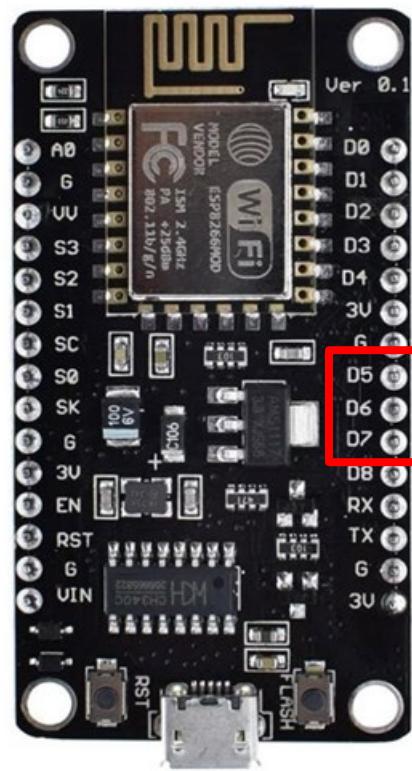
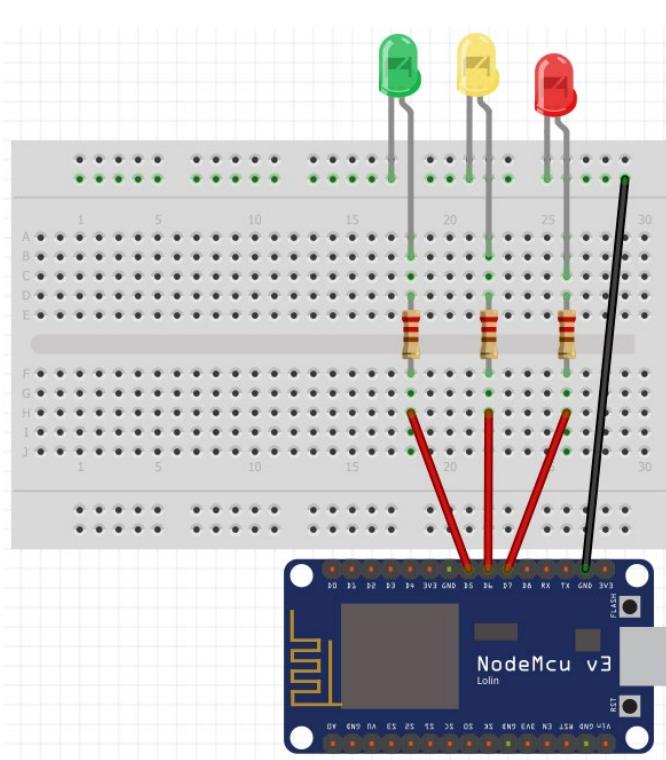
- ขั้นที่ 1 หันตัว ตัวต้านทาน (R) โดยให้ແບສີຄວາມຄລາດເຄລືອນ ສີເງິນແລະສົມອງໄປທາງຂວາ
- ແກບສີທີ່ 1 ເປັນຄ່າຕຳແໜ່ງທີ່ 1
- ແກບສີທີ່ 2 ເປັນຄ່າຕຳແໜ່ງທີ່ 2
- ແກບສີທີ່ 3 ເປັນຕົວຄຸນ
- ແກບສີທີ່ 4 ເປັນຄ່າຄວາມພິດພລາດ ບວກລບ

ອຸປະກນົນທີ່ຕ້ອງໃຊ້ຈານ

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

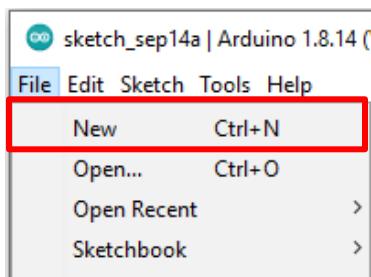
ກາຣຕ່ອວງຈຮ

PIN	ອຸປະກນົນ
D5	LED ສີເຂີຍວ
D6	LED ສີເໜືອງ
D7	LED ສີແດງ



การเขียน code

กดไปที่ File และกด New เพื่อสร้างหน้าต่างใหม่ในการเขียนโปรแกรม



จากนั้นเขียน code ตามดังนี้

Workshop_1

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

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

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

ผลลัพธ์ที่ได้

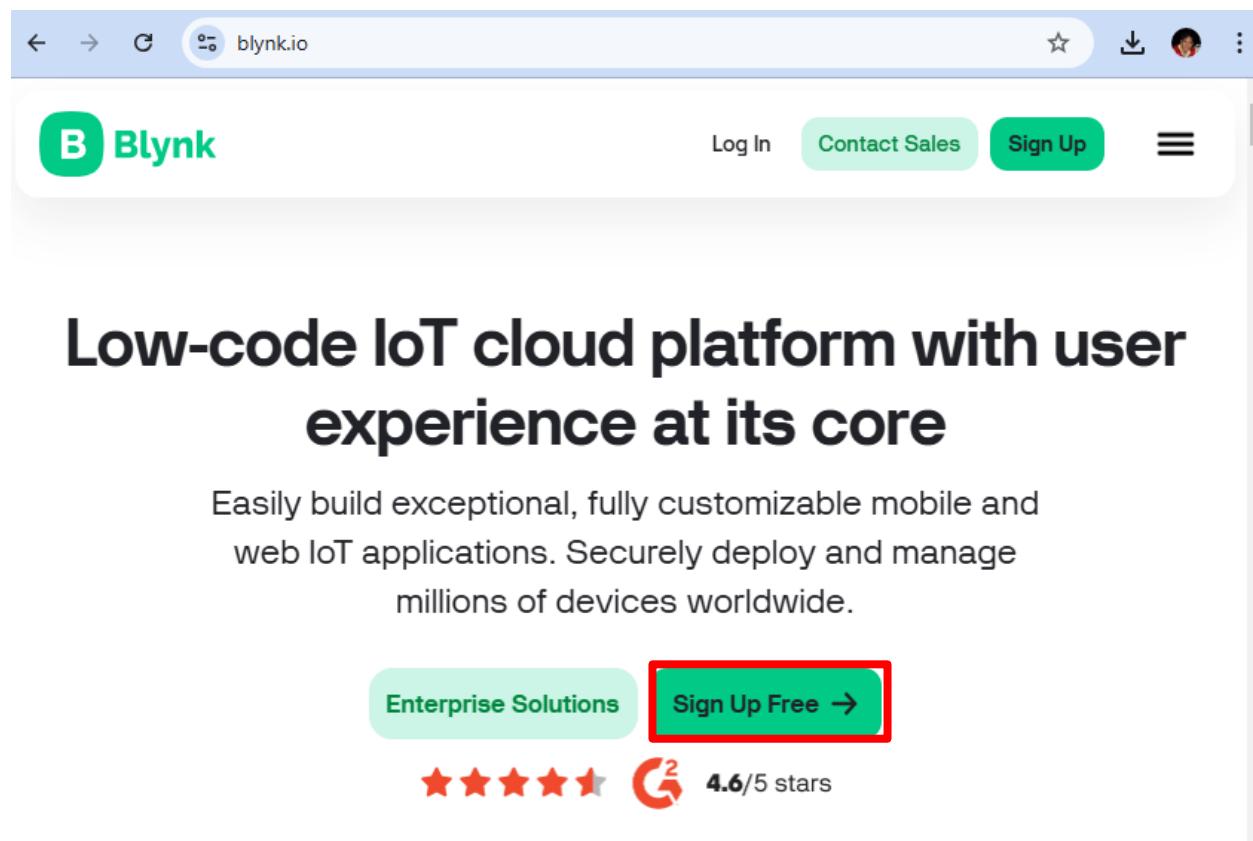
หลอดไฟ LED แต่ละตัวจะติดเป็นเวลา 0.5 วินาทีแล้วก็ดับ จากนั้นอีก 0.5 วินาที หลอดไฟ LED อีกดดวง
ติดขึ้น

Blynk

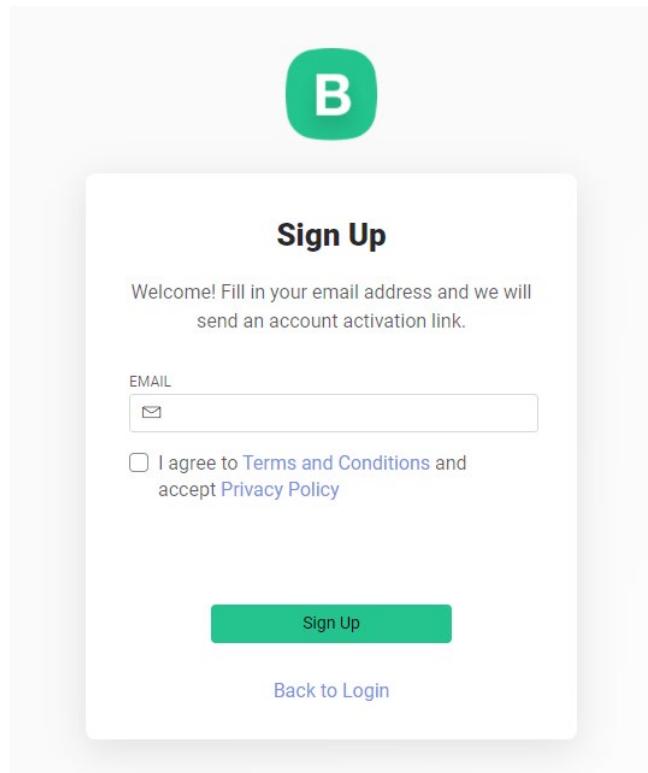
Blynk เป็น platform ที่ใช้ในการควบคุมหรือเชื่อมต่ออุปกรณ์ด้าน IoT ผ่าน Mobile Application หรือผ่าน Web Server โดยตัวซอฟต์แวร์เป็นตัวกลางระหว่างอุปกรณ์กับแอปพลิเคชัน โดยตัวแอปพลิเคชัน รองรับระบบ Android และ iOS

ขั้นตอนการติดตั้ง

1. ให้เปิดเว็บไซต์ blynk.io
2. กดปุ่ม Sign Up Free เพื่อทำการสมัคร



3. ทำการสมัครโดยใช้ Email และทำการ log in



เมื่อ log in เรียบร้อยแล้วจะอยู่ในหน้านี้

My organization - 5265TL |

Blynk.Console

Developer Zone >

- Devices
- Users
- Organizations
- Locations

My Templates

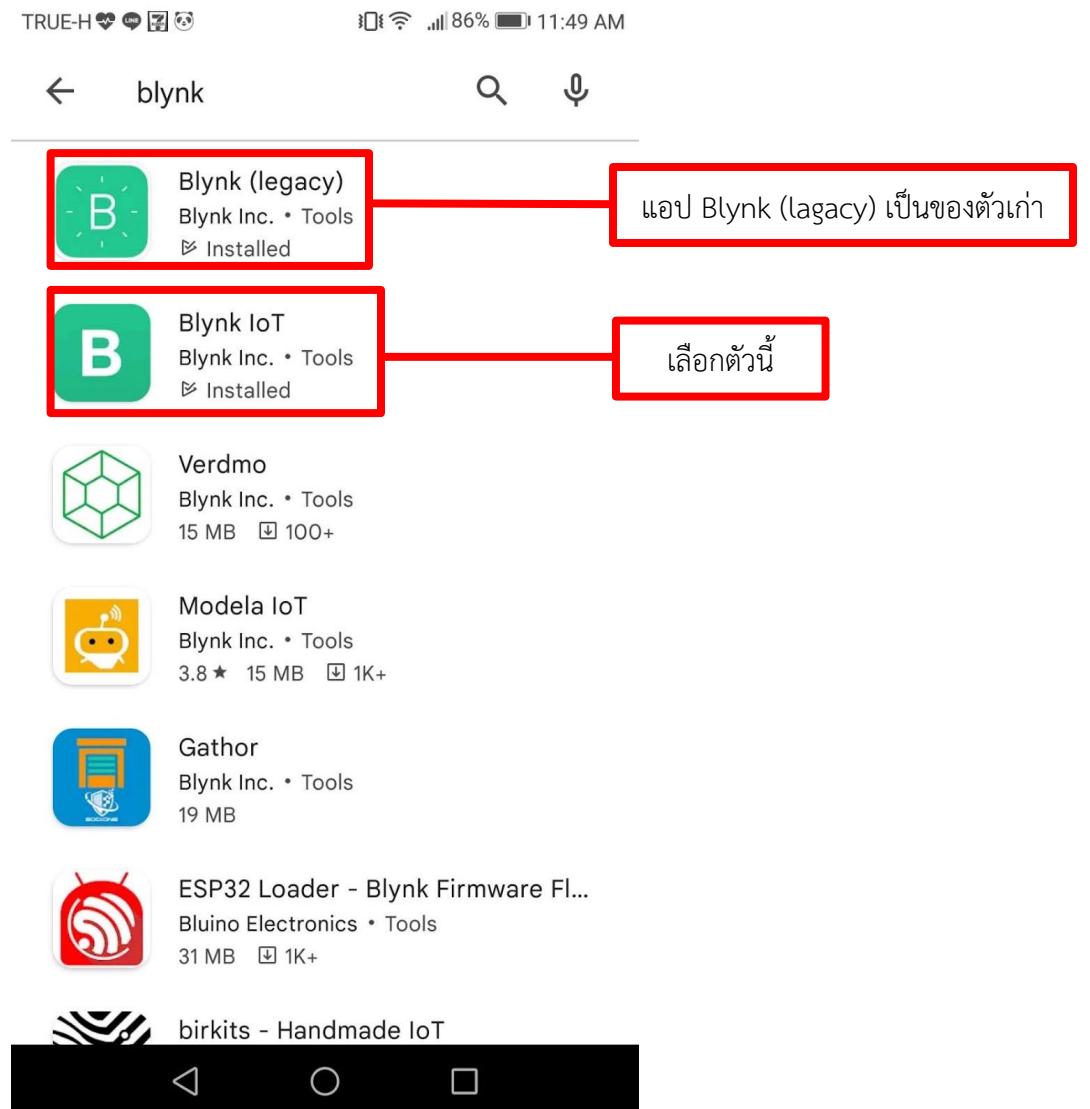
- Blueprints
- Blynk.Air (OTA)
- Webhooks
- Integrations

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 with the text 'Start by creating your first template'. Below it, a descriptive text explains what a template is: 'Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices.' At the bottom right of the 'My Templates' section, there's a green button with a plus sign and the text '+ New Template', which is highlighted with a red rectangular border.

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 red 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 (highlighted with a red box)

Test Blynk

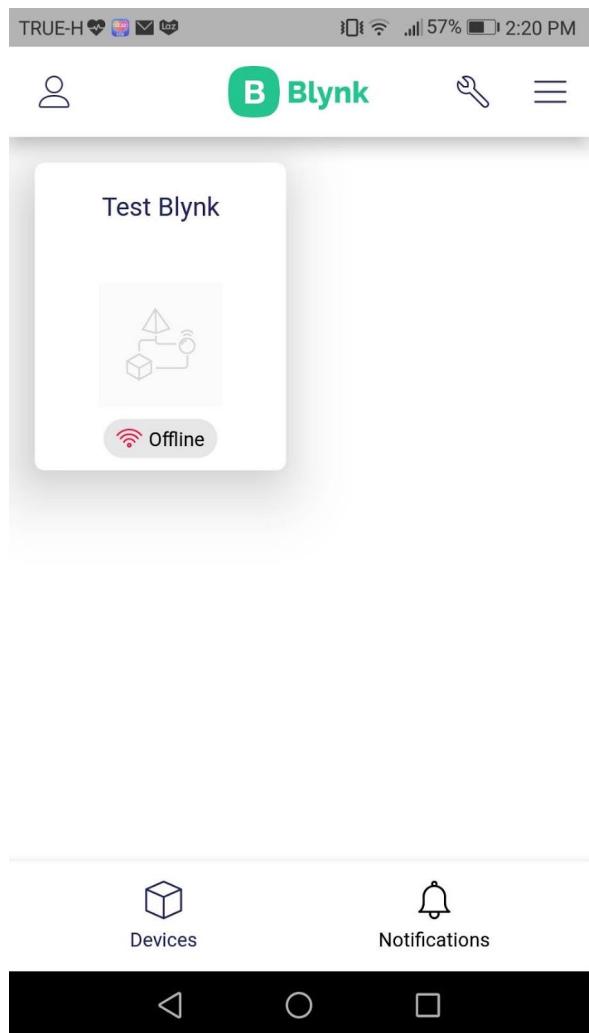
Test Blynk

Cancel **Create** (highlighted with a red box)

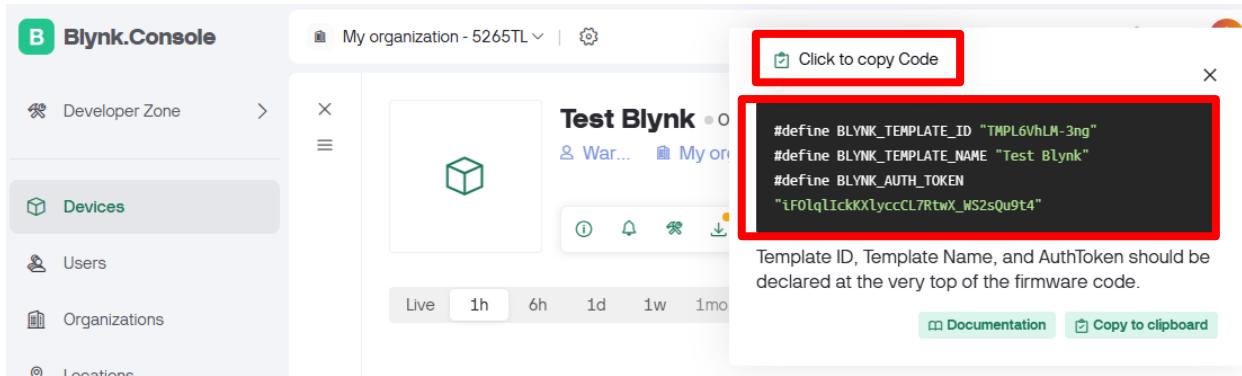
Point on the cards to see instruct

กด Create เพื่อสร้าง

6. ในส่วนของ Mobile Application หากเราได้ Device แล้วหน้าจะจะปรากฏตามดังรูป



7. กลับมาที่ฝั่งคอมพิวเตอร์ ให้เลือก Device ที่เราสร้างขึ้นมาแล้วกดไปที่ Device Info



ให้ copy ส่วนนีมำเขียนลงใน 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>

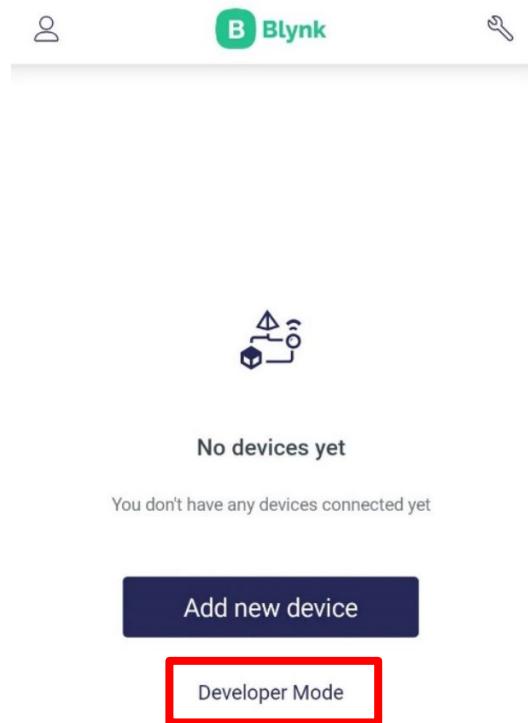
#define LED1 D5
#define LED2 D6
#define LED3 D7

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

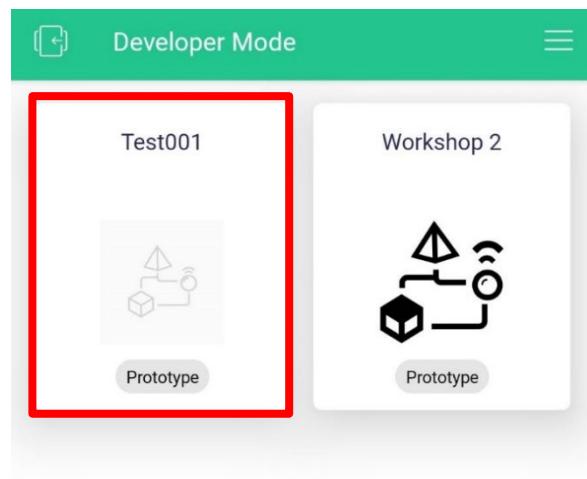
BlynkTimer timer;
void timerEvent();
```

การจัดหน้า Dashboard บน Mobile Application

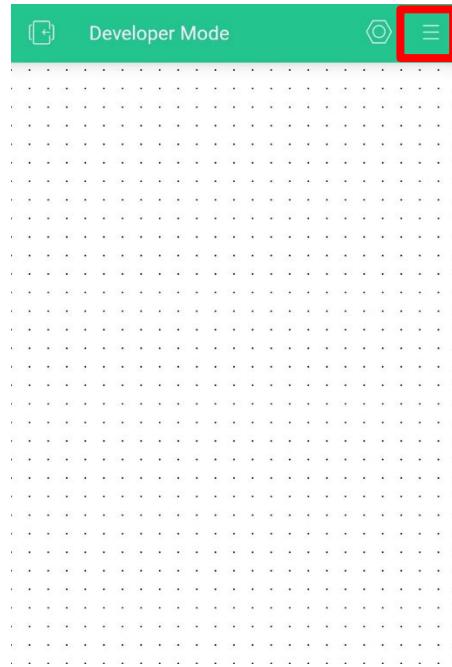
- กดเข้า Developer Mode เพื่อเลือก template



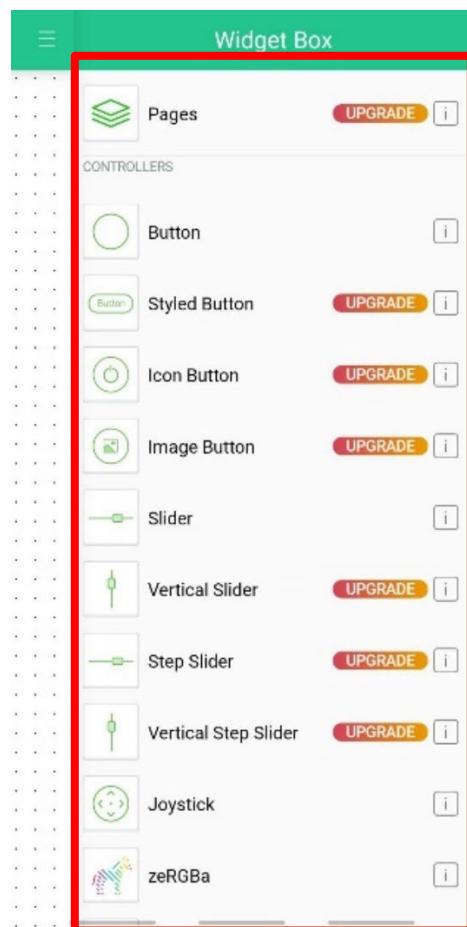
- เลือก Template ที่เราต้องการ



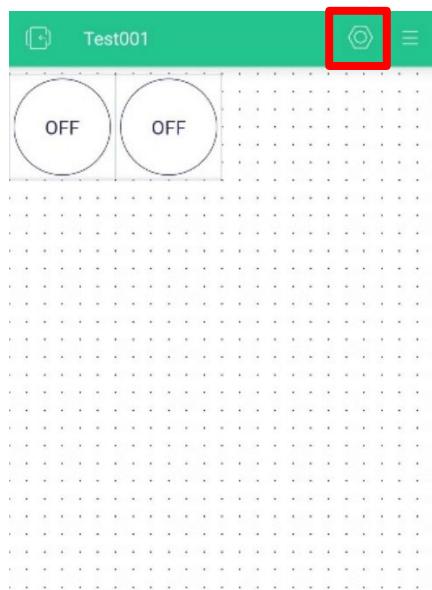
3. กดเพื่อเลือก widget ที่ต้องการในการออกแบบหน้า Dashboard



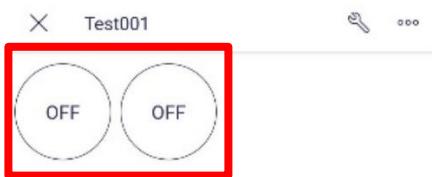
4. เลือก widget ตามที่ต้องการ



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

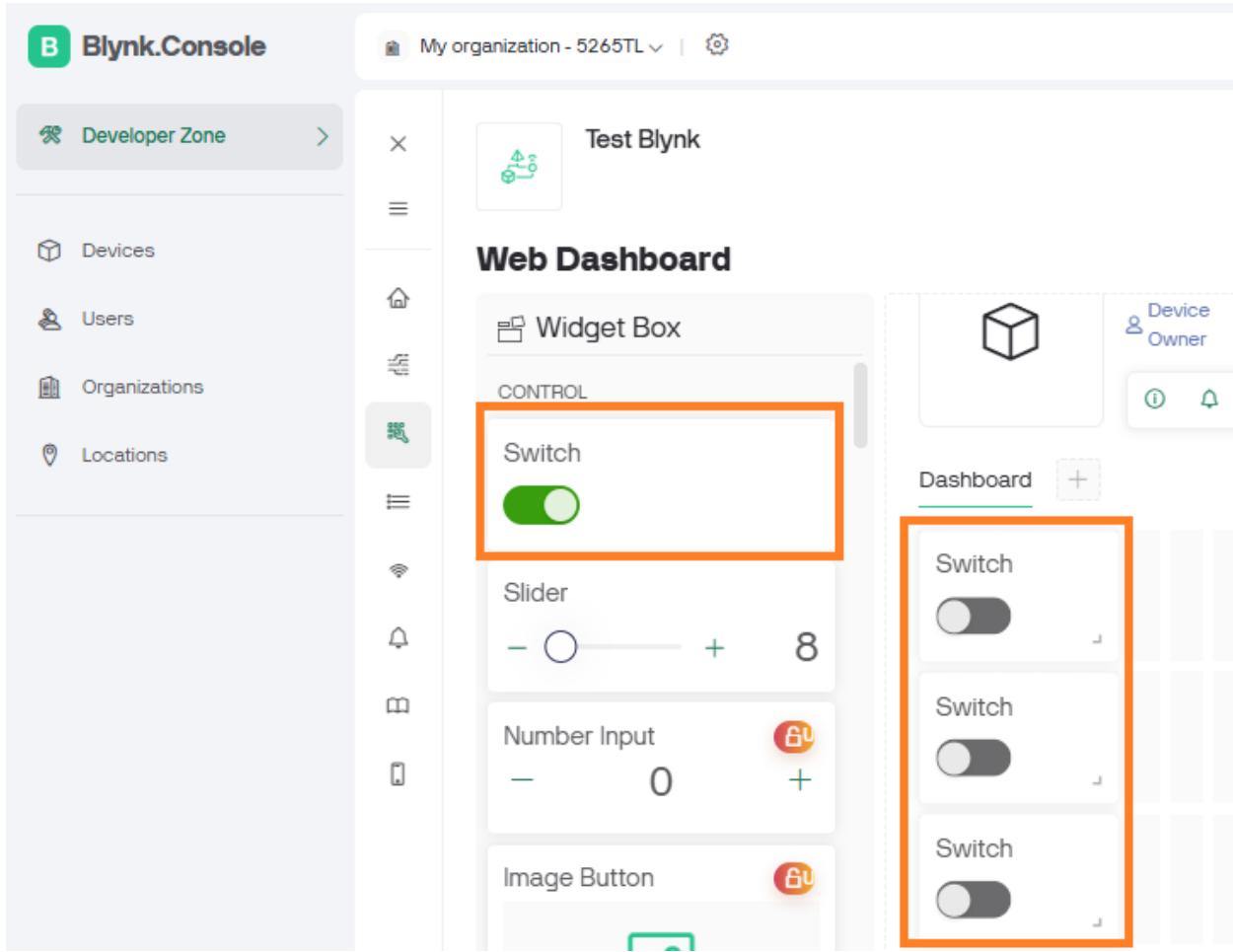
1. เลือก Device จากนั้นกดที่ device Test Blynk ตามภาพ

The screenshot shows the Blynk Console interface. On the left sidebar, the 'Devices' tab is highlighted with an orange box. The main area displays a list of devices under the heading 'Devices'. A search bar at the top says 'Start typing'. Below it is a 'Add Filter' button. A green button labeled '+ New Device' is on the right. The device list includes a row for 'Test Blynk' with columns for Name, Auth Token, Device Owner, and Actions. The 'Name' column shows 'Test Blynk', the 'Auth Token' column shows a long string of characters, and the 'Device Owner' column shows 'warakorn@tni.ac.th [you]'. An 'Actions' button is also present.

- ขั้นตอนมาให้เลือก Edit Dashboard

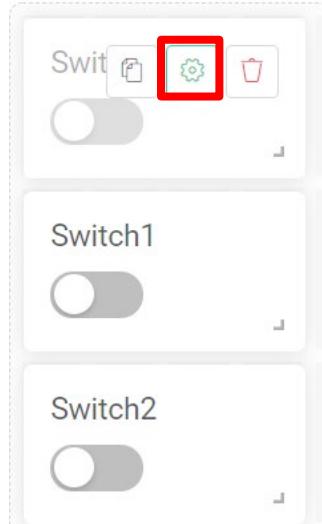
The screenshot shows the Blynk Console interface for the 'Test Blynk' device. The left sidebar shows the 'Devices' tab is selected. The main area displays the device details for 'Test Blynk'. It shows the device icon, name 'Test Blynk', status 'Offline', and a 'My o...' link. Below the device info are several small icons for live data, history, and other settings. At the bottom, there is a section titled 'No Dashboard widgets' with a sub-section 'Edit the dashboard to add widgets' and a green 'Edit Dashboard' button highlighted with an orange box.

- ลาก Widget Switch 3 ตัว ไปยัง Dashboard

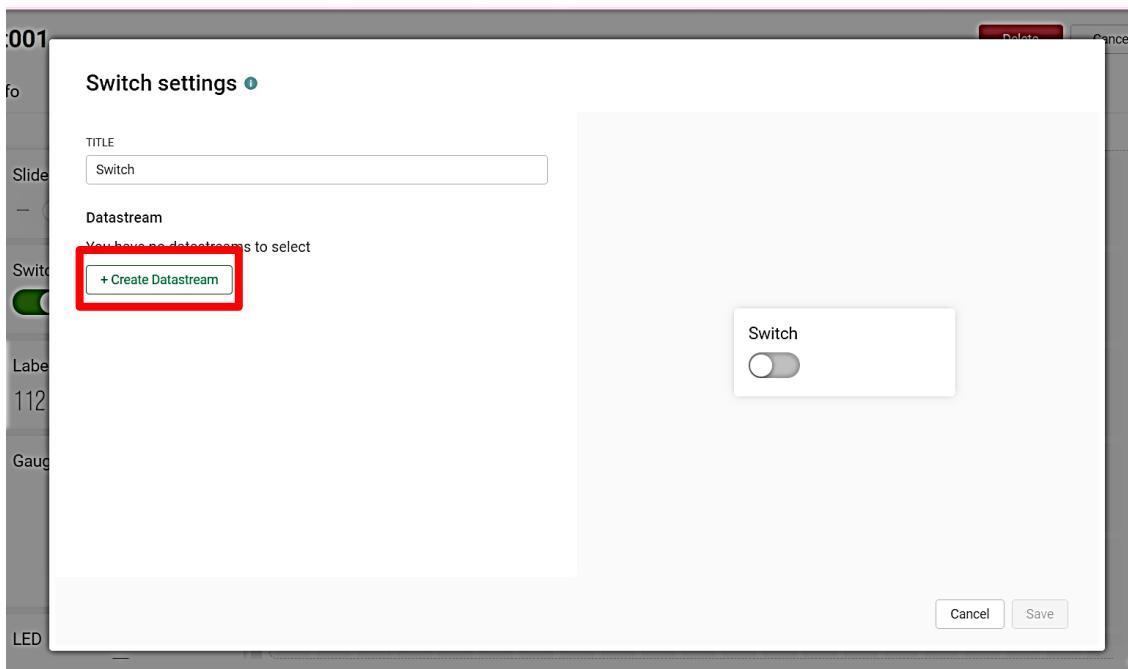


2. ตั้งค่า switch แต่ละตัว

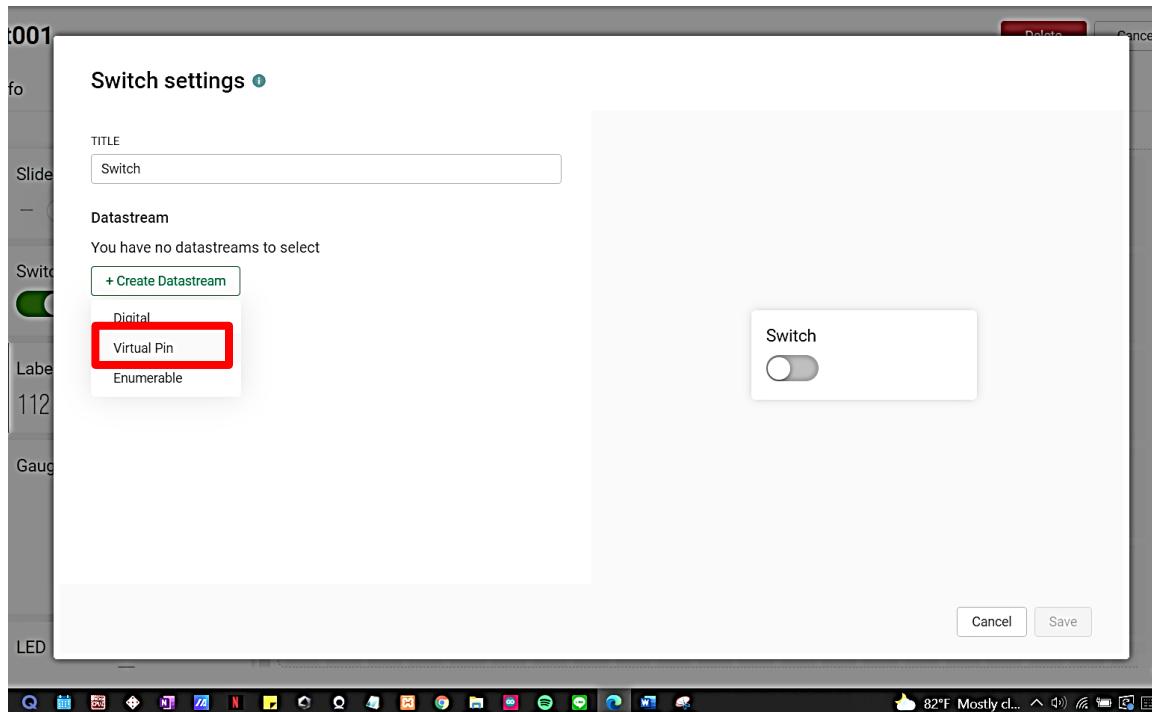
2.1 กดเพื่อตั้งค่า Switch



2.2 กดเพื่อตั้งค่าการรับส่งข้อมูล



2.3 กดเลือก virtual pin



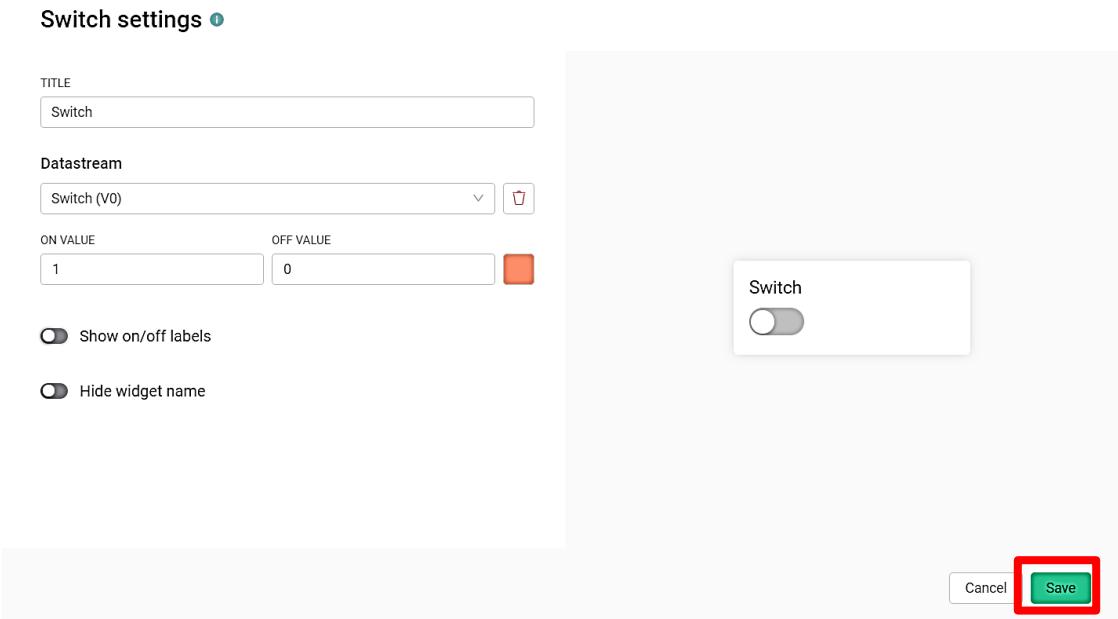
2.4 ให้ตั้งค่า PIN เป็น V0 และกด Create

Datastream

Virtual Pin Datastream

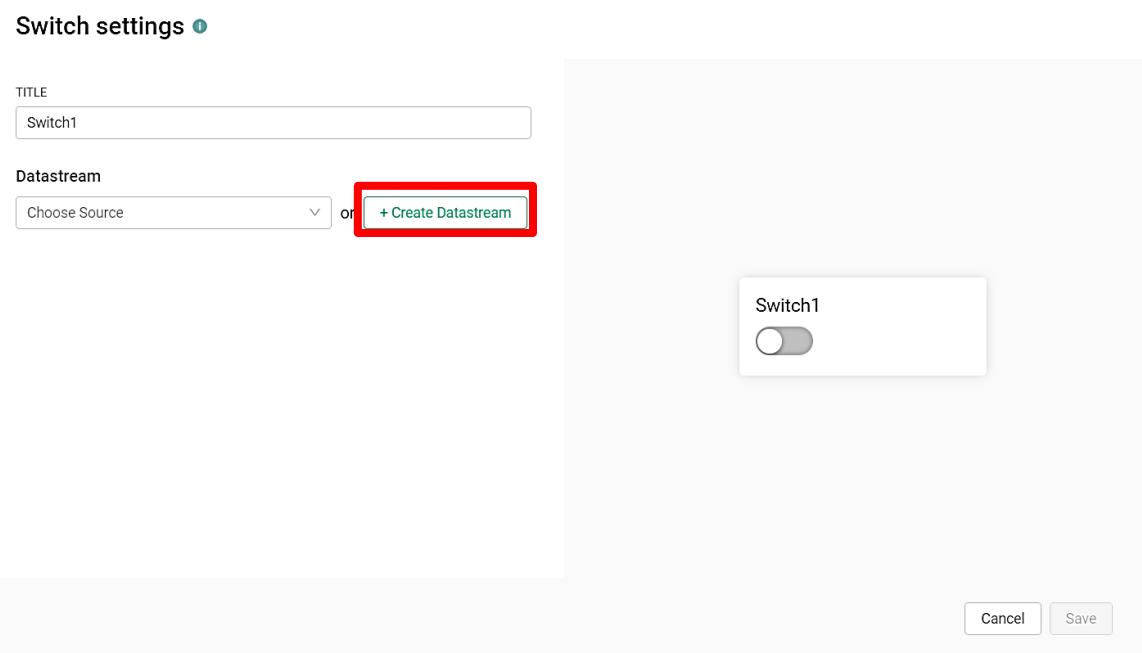
NAME	ALIAS
<input type="text" value="Switch"/>	<input type="text" value="Switch"/>
PIN	DATA TYPE
<input type="text" value="V0"/>	<input type="text" value="Double"/>
UNITS	
<input type="text" value="None"/>	
<input type="button" value="Cancel"/> <input type="button" value="Create"/>	

2.5 กด save เพื่อบันทึกการตั้งค่า switch

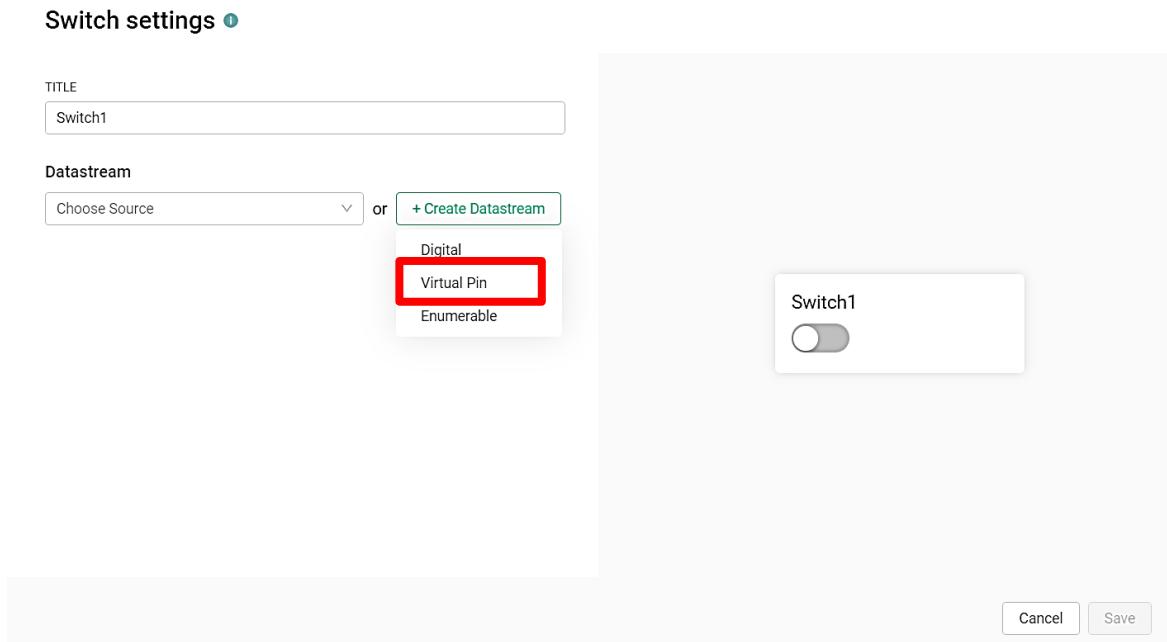


3 ตั้งค่า switch1

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

Cancel Save

4 ตั้งค่า switch2

4.1 กด create datasteam เพื่อตั้งค่า switch2

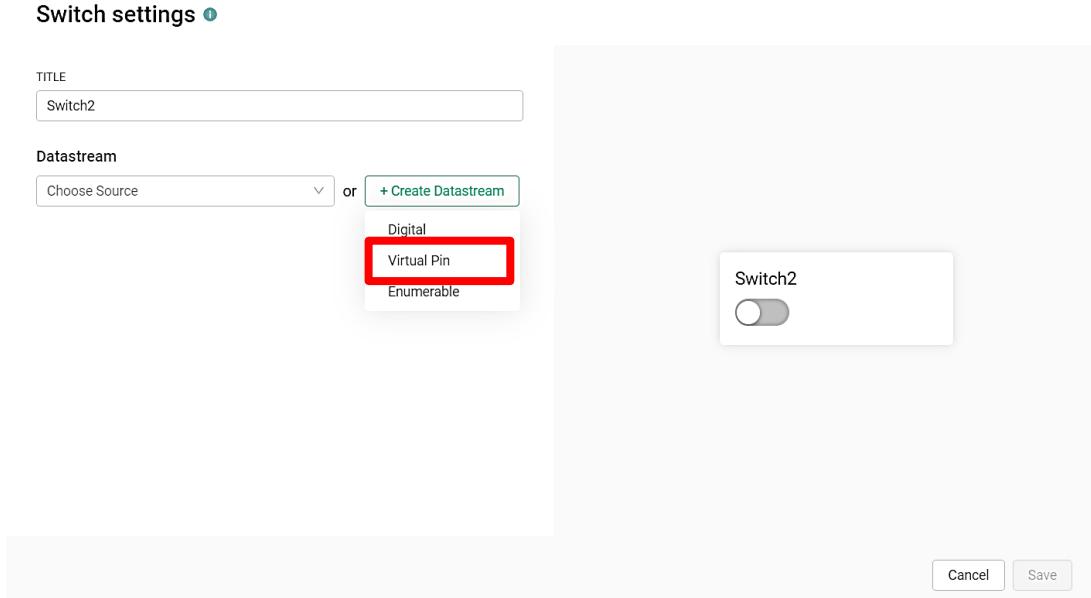
Switch settings ⓘ

TITLE
Switch2

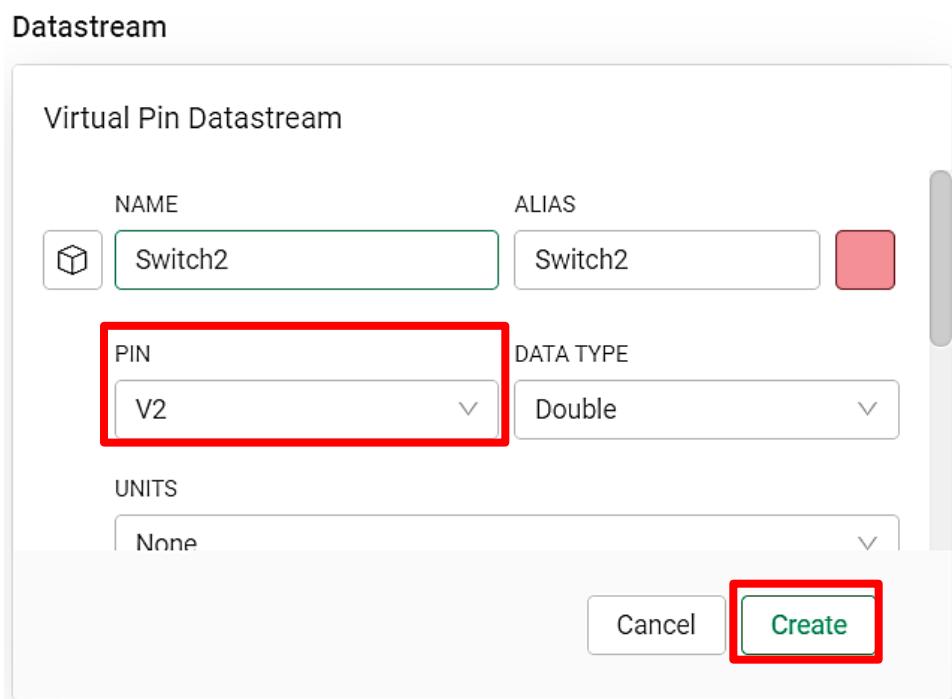
Datastream
Choose Source ▼ or + Create Datastream

Cancel Save

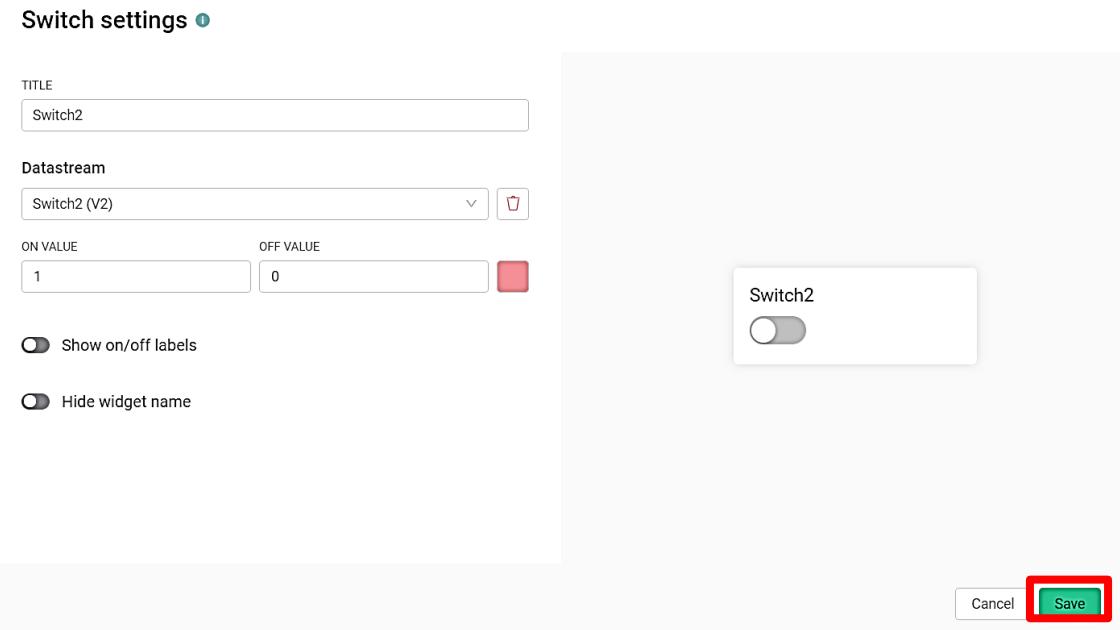
4.2 เลือก Virtual Pin



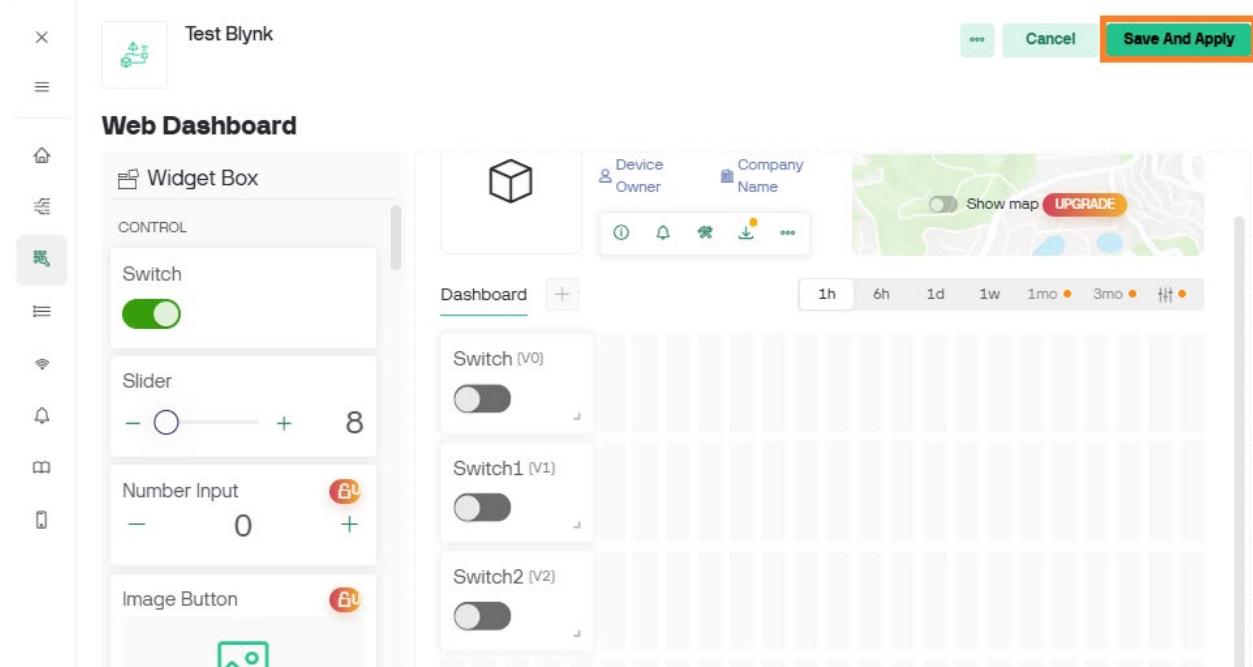
4.3 ให้ตั้งค่า PIN เป็น V2 และกด Create



4.4 กด save เพื่อบันทึกการตั้งค่า switch



4.5 กด save and Apply เพื่อบันทึกการตั้งค่า dashboard

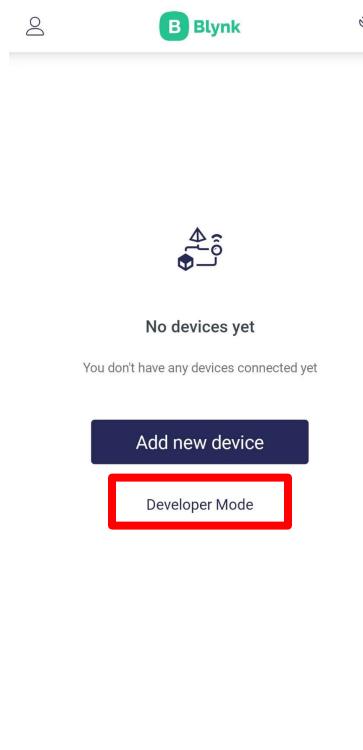


ก่อนใช้งานให้ลง library blynk

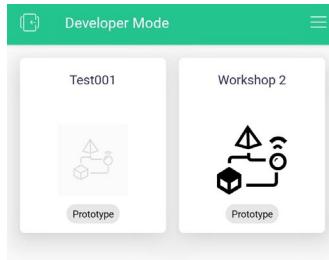


การตั้งค่า Dashboard บนโทรศัพท์

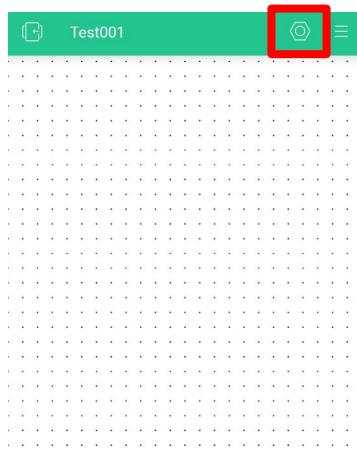
1. เข้าไปที่ Developer Mode



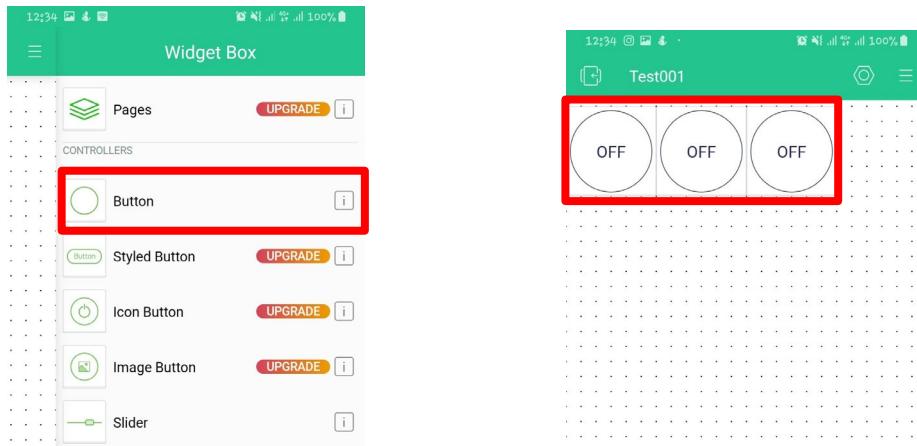
2. เลือก Prototype ของตนเอง



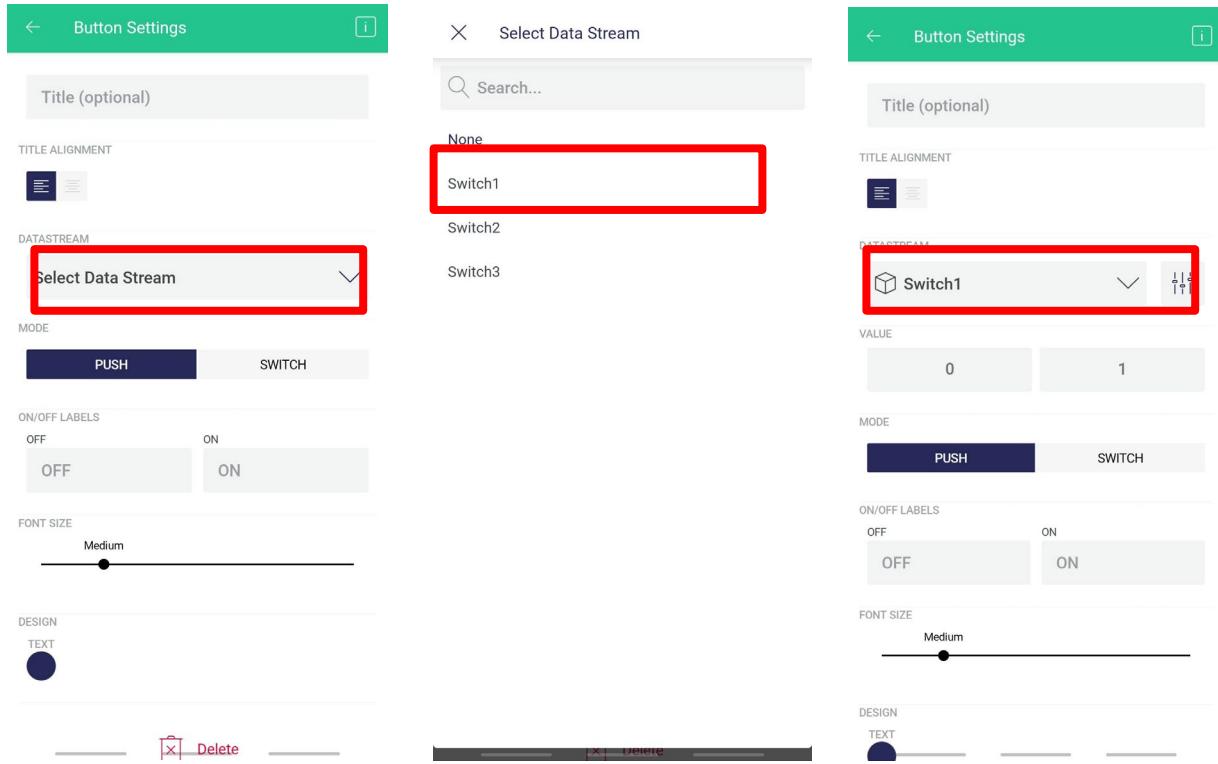
3. กดไปที่ปุ่มดังรูป



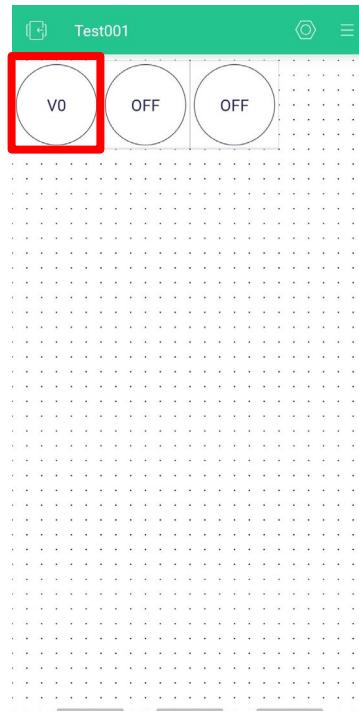
4. เลือกไปที่ Button โดยกดเลือกมาทั้งหมด 3 Button



5. กดไปที่ Button ตัวที่ 1 เพื่อทำการตั้งค่า และกดเลือก data stream ที่ต้องการ

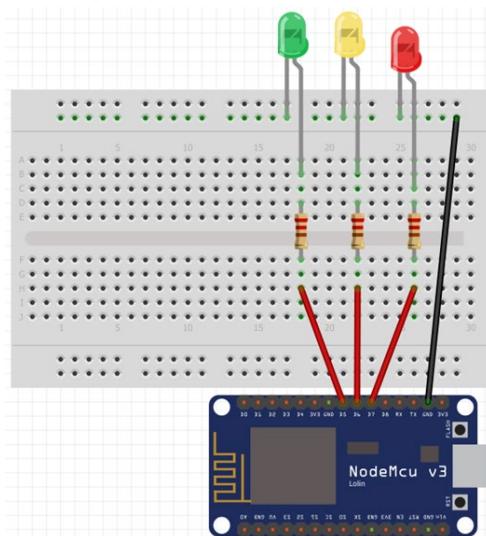


6. จะได้ผลลัพธ์ดังรูป จากนั้นนำมาขึ้นตอนที่ 5 อีก 2 Button



การต่อวงจร

PIN	อุปกรณ์
D5	LED สีเขียว
D6	LED สีเหลือง
D7	LED สีแดง



การเขียน code (กรอบสีแดงคือต้องเปลี่ยนเป็นข้อมูลของตัวเอง)

Workshop_10

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

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

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

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

BlynkTimer timer;
void timerEvent();

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

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

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

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

```

BLYNK_WRITE(V0) { //function การทำงานของ visualpin 0
    if (param.asInt()) { //param.asInt() ? 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