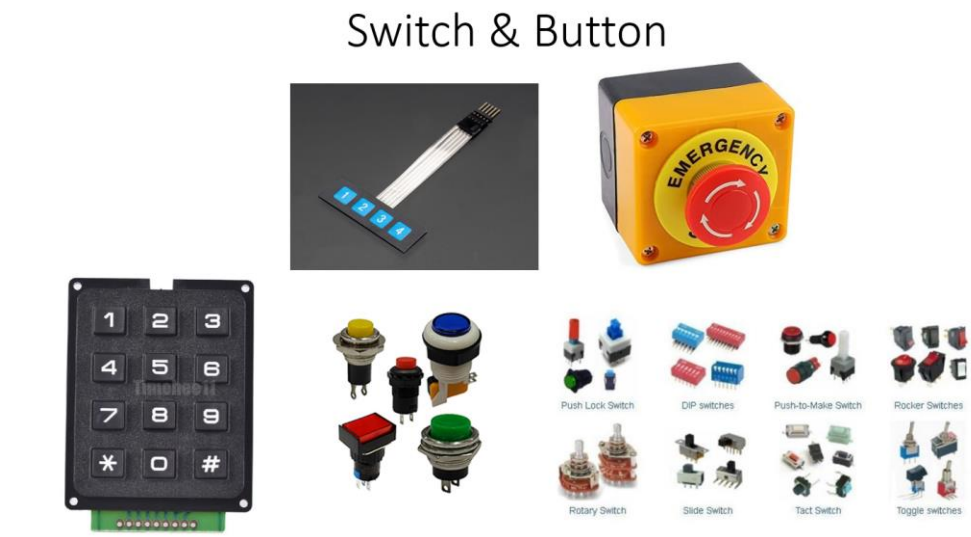
การทดลองที่ 3 : ดิจิทัลอินพุทด้วยสวิตช์และเซ็นเซอร์วัดอุณหภูมิ DHT11

วัตถุประสงค์

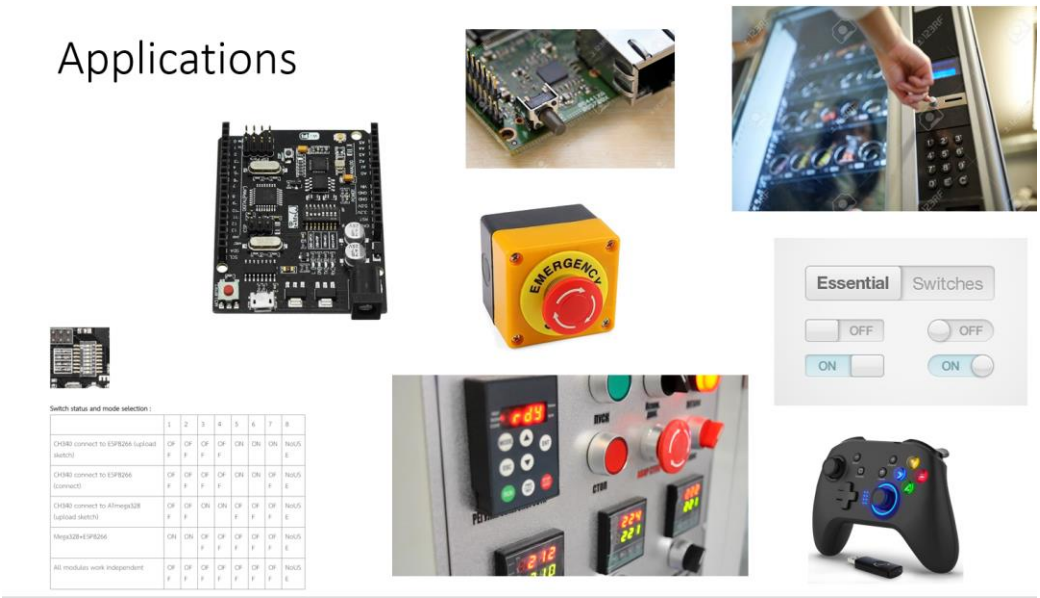
1. เพื่อให้นักศึกษารู้จักการใช้งานบอร์ด NodeMCU ร่วมกับสวิตช์เบื้องต้น รวมถึงการ pull แบบต่าง ๆ
2. เพื่อให้นักศึกษารู้ลำดับขั้นตอนในการติดตั้งโปรแกรม และไลบลารีเพื่อใช้งาน NodeMCU ร่วมกับโมดูลหรือเซ็นเซอร์อื่น ๆ เช่น dht11
3. เพื่อให้นักศึกษาทบทวนการใช้งาน Arduino IDE ที่ได้เรียนมา และใช้งานคำสั่งเพื่อทดลองกับบอร์ด NodeMCU เบื้องต้น

บทนำ

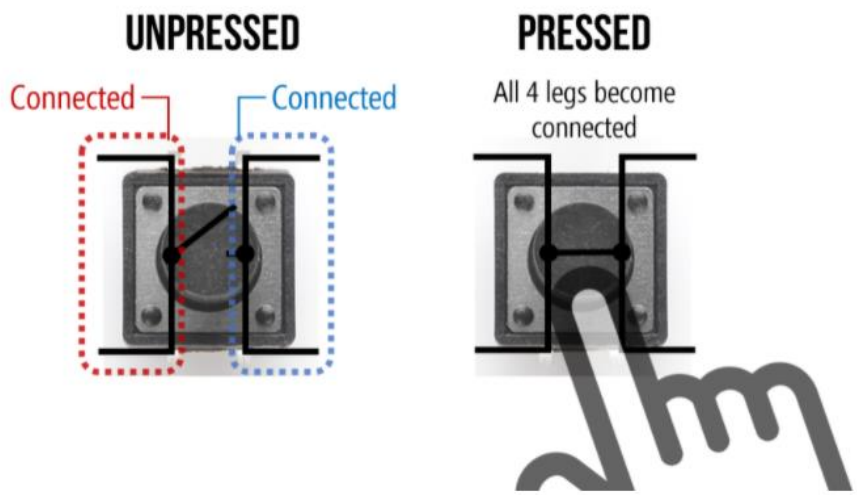
จากในบทที่ผ่านมา General Purpose Input/Output หรือ GPIO ทำหน้าที่เป็นเอาท์พุทในการควบคุมให้ไฟส่องสว่าง โดยในหัวข้อนี้จะใช้งาน GPIO เป็นอินพุท โดยทั่วไปแล้วอุปกรณ์ที่ทำหน้าที่เป็นอินพุทในรูปแบบดิจิทัลอินพุทก็คือสวิตช์นั่นเอง โดยสวิตช์มีหลายรูปแบบ แบ่งการใช้งานตามจำนวนขาเข้าและขาออก หรือลักษณะของการใช้งาน เช่น สวิตช์แบบกดติด-ปล่อยดับ ดังเช่น สวิตช์ที่เป็นกริ่งหรือออดหน้าประตูบ้าน หรือปุ่มที่ใช้งานสำหรับรีเซ็ตอุปกรณ์ เป็นต้น หรืออีกรูปแบบ เช่น สวิตช์แบบกดติด-กดดับ เช่น สวิตช์สำหรับเปิดใช้งานอุปกรณ์ไฟฟ้าทั่วไป นอกจากนี้สวิตช์ยังมีอินพุท เอาท์พุทได้หลายทิศทาง แต่สถานการณ์ทำงานนั้นจะมีเพียงสองสถานะ คือ สถานะเปิด (On) และสถานะปิด (Off)



สวิตช์หรือปุ่มกดแบบต่าง ๆ



ตัวอย่างการนำไปประยุกต์ใช้งาน

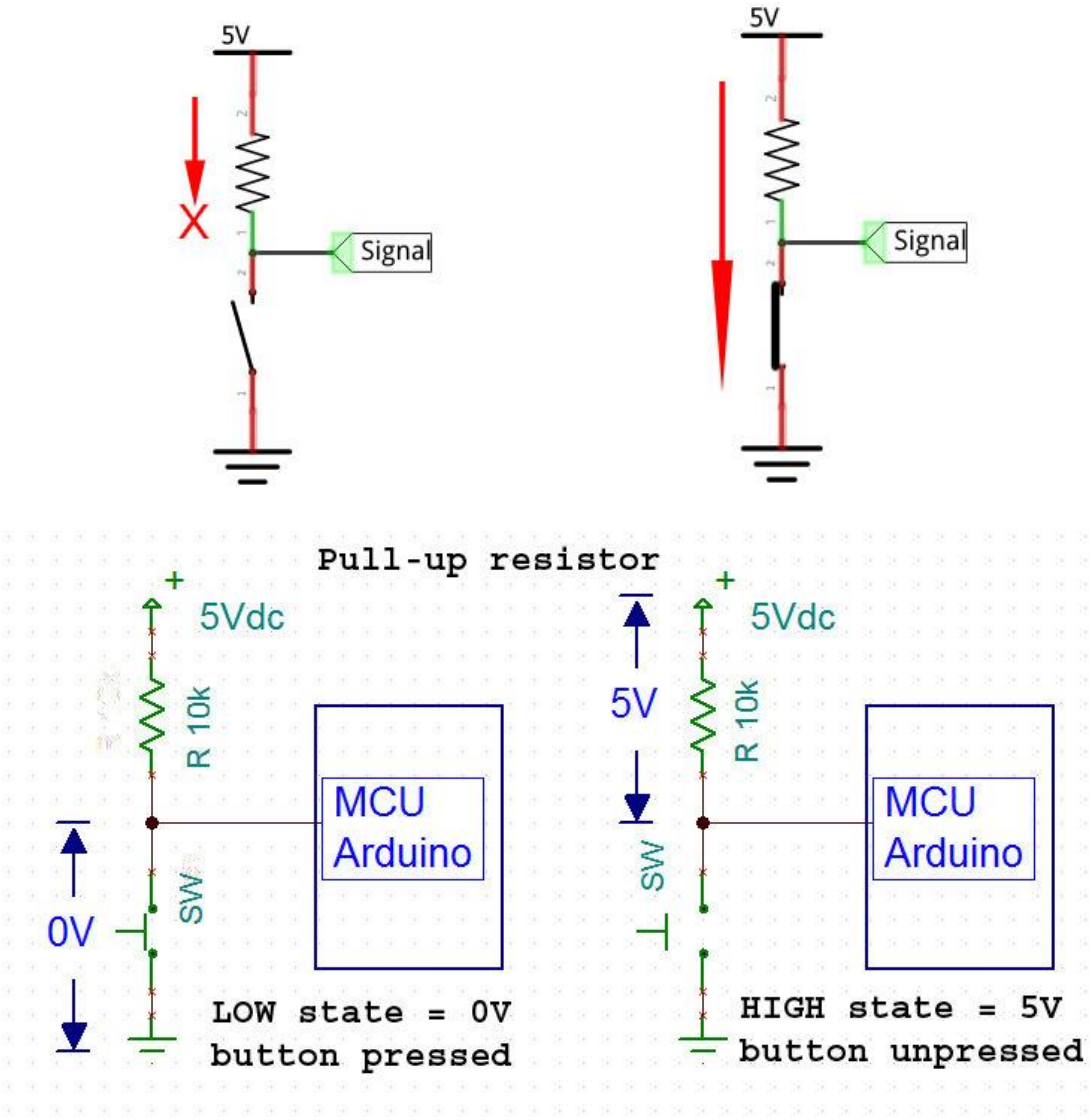


สถานะปุ่มกดของสวิตช์แบบ 4 ขา

การต่อสวิตช์โดยตรง อาจทำให้วงจรเกิดความเสียหายได้ หรือเกิดสถานะ “Floating” (ไม่รู้ 1 หรือ 0 ส่งมามั่วๆ) อย่างไรก็ตาม สามารถต่อโดยตรงได้แต่ต้องใช้คำสั่ง Input_pullup สำหรับ ESP8266 ทุก ๆ ขา GPIO จะมีตัวต้านทานมาให้ทุกพอร์ตสำหรับ Pull-Up ยกเว้น GPIO16 จะเป็น Pull-Down

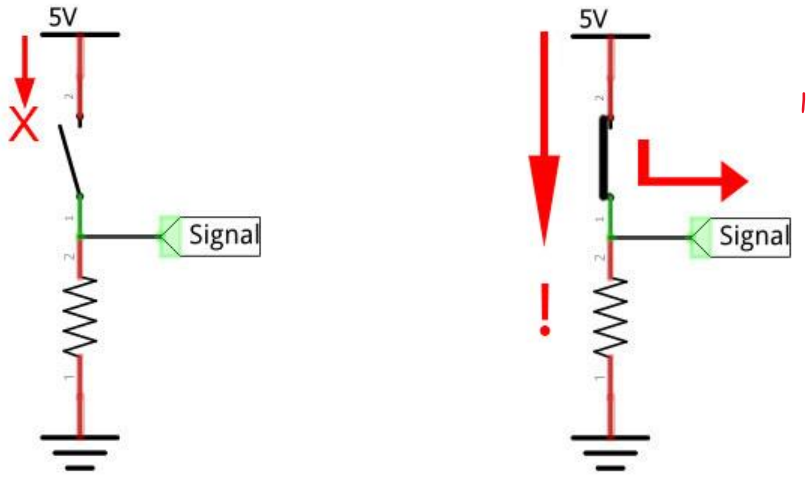
โดยทั่วไป ในการต่อวงจรกับสวิตช์ จะมีรูปแบบการต่อสองแบบด้วยกันได้แก่ แบบ Pull Up และ Pull Down

- แบบ **Pull Up** คือ ในขณะที่ไม่ได้กดสวิตช์ ไมโครคอนโทรลเลอร์จะอ่านแรงดันไฟฟ้าจากแหล่งจ่าย โดยอ่านได้ลอจิก HIGH (‘1’) แต่เมื่อมีการกดสวิตช์ แรงดันไฟฟ้าจะลงมาที่ 0 V ซึ่งเรียกว่าลอจิก LOW (‘0’) ดังนั้นการทำงานภายใต้การต่อแบบ Pull up จึงถูกเรียกว่า **Active Low** นั่นคือลอจิก LOW ในขณะทำงาน



การต่อสวิตช์แบบ Pull Up

- **Pull Down** จะได้ผลลัพธ์กลับกัน คือ หากไม่ได้กดสวิตช์ จะได้ลอจิก LOW ‘0’ แต่เมื่อกดสวิตช์จะได้ลอจิก HIGH ‘1’ ทำให้การต่อแบบ Pull Down ถูกเรียกว่า Active High นั่นคือเกิดการสั่งการในขณะลอจิก HIGH นั่นเอง



การต่อสวิตช์แบบ Pull Down

สำหรับคำสั่งเพื่อใช้งาน GPIO ให้เป็นอินพุทนั้น จะมีคำสั่งที่คล้ายกันกับการสั่งให้เป็น Output โดยจะต้องประกาศ pinMode เช่นเดียวกันเพียงแต่เปลี่ยนเป็น INPUT ดังตัวอย่าง

pinMode(GPIO_Pin, INPUT)

โดยที่ ***GPIO_Pin*** คือ หมายเลขขา GPIO ที่ใช้งาน
อย่างไรก็ตาม ในไมโครคอนโทรลเลอร์มักมีตัวต้านทานมาให้ภายในขาต่าง ๆ แต่ต้องประกาศใช้งานสถานะ Input ให้เป็น Input Pull-Up โดยใน Arduino/NodeMCU นี้สามารถใช้ฟังก์ชัน Input Pull-Up เช่น ใช้คำสั่ง

pinMode(GPIO_Pin,INPUT_PULLUP)

ดิจิทัลอินพุท (digitalRead)

เมื่อมีการกำหนดให้ขาใดขาหนึ่ง ของไมโครคอนโทรลเลอร์ทำหน้าที่เป็น input ด้วยคำสั่ง pinMode ก็สามารถใช้คำสั่ง digitalRead เพื่อสั่งให้ไมโครคอนโทรลเลอร์อ่านค่าสัญญาณที่เชื่อมต่ออยู่กับขานั้นๆได้ โดย syntax ของการใช้คำสั่ง digitalRead เป็นไปตามรูปที่ 2.1

```
digitalRead(pin);
```

รูปที่ 2.1 คำสั่ง digitalRead

ค่าของ pin ที่ใช้ในคำสั่งนี้เป็นหมายเลขของขาดิจิตอลที่ต้องการอ่านค่าว่าเป็นสัญญาณ HIGH หรือ LOW และหากขานั้นไม่ได้ทำการเชื่อมต่อกับอะไรเลย คำสั่ง digitalRead จะทำให้สามารถอ่านค่าออกมาเป็นได้ทั้ง HIGH และ LOW (เกิดขึ้นแบบสุ่ม) ตัวอย่างของการใช้คำสั่งนี้เป็นไปตามรูปที่ 2.2

```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7;   // pushbutton connected to digital pin 7
int val = 0;     // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin 13 as output
  pinMode(inPin, INPUT);      // sets the digital pin 7 as input
}

void loop()
{
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);    // sets the LED to the button's value
}
```

รูปที่ 2.2 ตัวอย่างการใช้คำสั่ง digitalRead

<u>การทดลอง Digital Input ด้วยสวิตช์</u>

สำหรับขั้นตอนการทดลอง จะทำการทดลอง ให้ทำการเชื่อมต่อสวิตช์กับ NodeMCU โดยทดลอง**ต่อวงจรสวิตช์ทั้งแบบ Pull Up และ Pull Down** โดยในที่นี้เชื่อมต่อกับ GPIO4 (Pin D2)

ทดสอบการทำงานของสวิตช์แต่ละตัวโดยใช้คำสั่งหรือโค้ดดังตัวอย่าง

Code Example 3.1 Digital Input

```
int pushButton = D2;
void setup() {
  Serial.begin(9600);
  pinMode(pushButton, INPUT);
}
```

```
void loop() {
  int buttonState = digitalRead(pushButton);
  Serial.println(buttonState);
  delay(100);        // delay in between reads for stability
}
```

ผลลัพธ์ที่ได้เมื่อกดสวิตช์ และเมื่อปล่อยสวิตช์ เป็นอย่างไร ?
ทดลอง โดยการเปิดหน้าจอ Serial Monitor สังเกตผลลัพธ์ที่ได้
เมื่อไม่กดปุ่ม ค่าที่แสดงผลคือ _____ เมื่อกดปุ่ม ค่าที่แสดงผล คือ _____

หากทดลอง**นำตัวต้านทานออก** ให้ Switch เชื่อมต่อโดยตรงกับบอร์ด จะเป็นอย่างไร ?

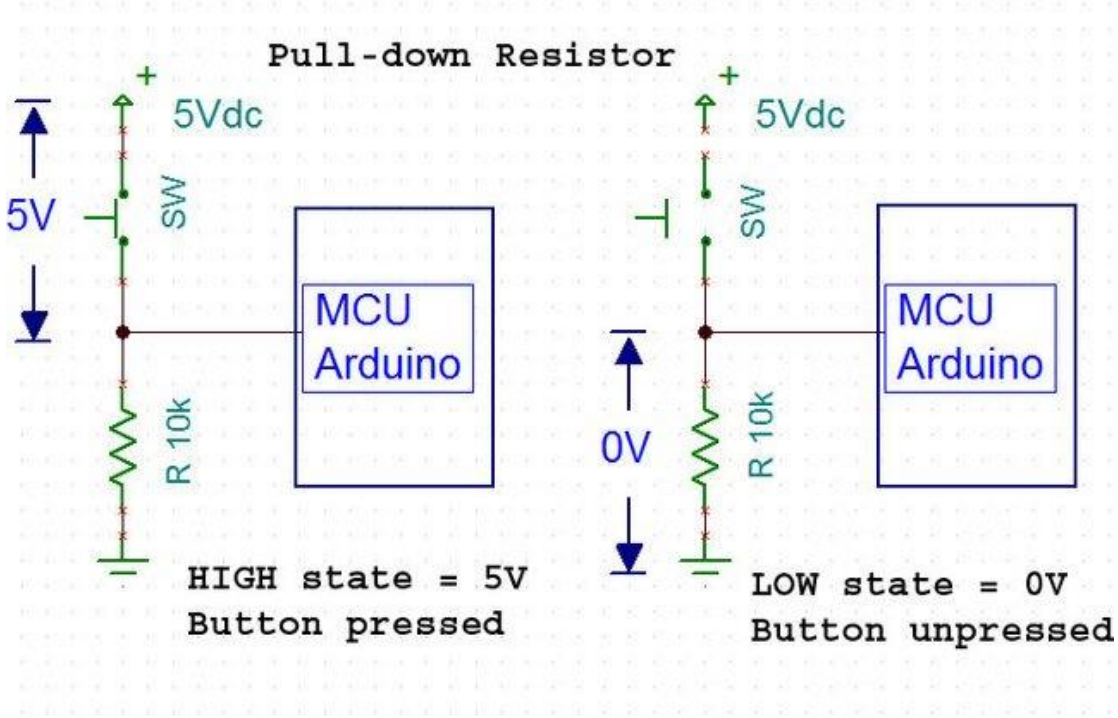
_____

จากนั้น ทดลองใช้คำสั่ง INPUT_PULLUP
Code Example 3.1.1 Digital Input

```
int pushButton = D2;
void setup() {
  Serial.begin(9600);
  pinMode(pushButton, INPUT_PULLUP);
}
void loop() {
  int buttonState = digitalRead(pushButton);
  Serial.println(buttonState);
  delay(100);        // delay in between reads for stability
}
```

ผลลัพธ์ที่ได้เมื่อกดสวิตช์ และเมื่อปล่อยสวิตช์ เป็นอย่างไร ?

จากนั้นทำการทดลองต่อสวิตช์แบบ Pull Down ดังภาพ



ใช้ Code เดิมจาก Example 3.1

ผลลัพธ์ที่ได้เมื่อกดสวิตช์ และเมื่อปล่อยสวิตช์ เป็นอย่างไร ?

ทดลอง โดยการเปิดหน้าจอ Serial Monitor สังเกตผลลัพธ์ที่ได้

เมื่อไม่กดปุ่ม ค่าที่แสดงผลคือ _____ เมื่อกดปุ่ม ค่าที่แสดงผล คือ _____

จากนั้นทำการเชื่อมต่อกับ LED โดยให้ LED อยู่ที่ D2 โดยใช้ความรู้จากการเรียนเรื่อง Digital Output ทดลองเขียนโปรแกรมดังตัวอย่างที่ 3.2 นี้

Code Example 3.2 Digital Input with LED

```
const int buttonPin = D1;
const int ledPin =  D2;
int buttonState = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
```

```
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
delay(100);
}
```

ทำการทดสอบว่าเมื่อกดสวิตช์แล้ว LED จะเปลี่ยนสถานะหรือไม่
เมื่อต่อแบบ Input Pull Up เมื่อกดสวิตช์ และเมื่อปล่อยสวิตช์ LED จะติด หรือดับ ?
และเมื่อต่อแบบ Input Pull Down เมื่อกดสวิตช์ และเมื่อปล่อยสวิตช์ LED จะติด หรือดับ ?

ทดลอง โดยการเปิดกดปุ่ม และปล่อยปุ่ม สังเกตผลลัพธ์ที่ได้
เมื่อไม่กดปุ่ม แอลอีดีที่แสดงผลคือ _____
เมื่อกดปุ่ม แอลอีดีที่แสดงผล คือ _____

แล้วถ้าหากกดสวิตช์ค้างไว้จะเป็นอย่างไร ?

หากต้องการให้สวิตช์ทำงานแบบกดติด-กดดับ (Toggle) โดยต่อสวิตช์แบบ PULL-UP โดยต่อสวิตช์กับ D1 และ LED กับ D2
Code Example 3.3 Digital Input with LED Toggle

```
const int buttonPin =D1;
const int ledPin =  D2;
boolean buttonState;
boolean lastButtonState;
boolean state = false;
```

```
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW && lastButtonState == HIGH) {
    state =!state;
    }
    digitalWrite(ledPin,state);
    lastButtonState = buttonState;
}
```

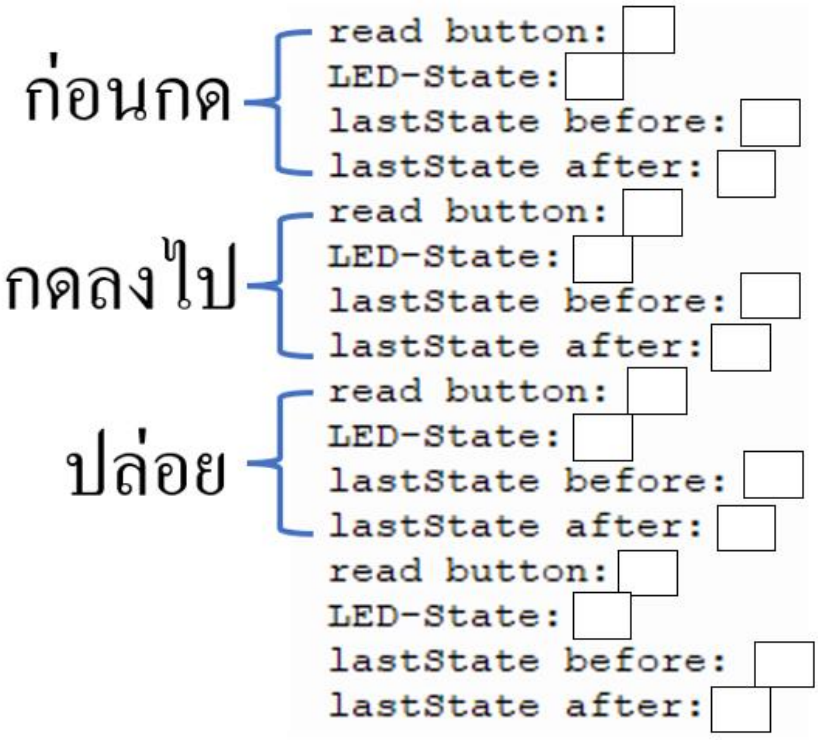
ทดลองกดปุ่มสวิตช์ ผลลัพธ์ที่ได้เป็นอย่างไร
อาจจะเข้าใจยาก ทดลองเพิ่ม Serial ต่าง ๆ เข้าไปดัง Example 3.4

Code Example 3.4 Digital Input with LED Toggle

```
const int buttonPin =D1;
const int ledPin =  D2;
boolean buttonState;
boolean lastButtonState;
boolean state = false;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  Serial.begin(115200);
```

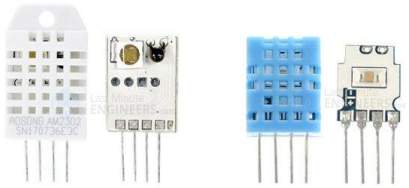
```
}

void loop() {
  buttonState = digitalRead(buttonPin);
  Serial.print("read button :");
  Serial.println(buttonState);
  if (buttonState == LOW && lastButtonState == HIGH) {
    state =!state;
Serial.print("LED-state:");
  Serial.println(state);

    }
    digitalWrite(ledPin,state);
Serial.print("lastState before :");
  Serial.println(lastButtonState);

    lastButtonState = buttonState;

Serial.print("lastState after :");
  Serial.println(lastButtonState);
delay(300);
}
```

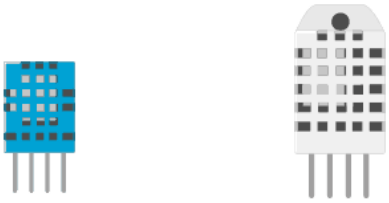
ทดลองกดปุ่มสวิตช์ ผลลัพธ์ที่ได้เป็นอย่างไร ทำการทดสอบว่าเมื่อกดสวิตช์แล้ว LED จะเปลี่ยนสถานะหรือไม่ บันทึกผลจาก Serial Monitor ลงไปในกล่องสี่เหลี่ยม



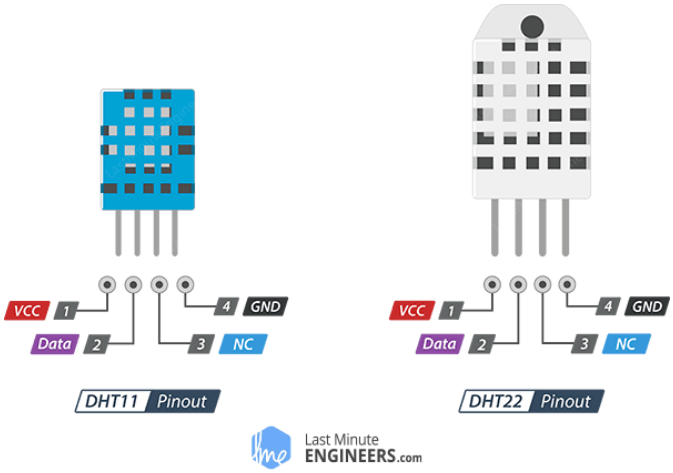
การทดลอง 3.2 วัดอุณหภูมิ และความชื้น DHT11 / DHT22

DHT11 / 22 เป็นเซ็นเซอร์ที่ใช้วัดอุณหภูมิ และวัดความชื้นได้อีกด้วย มีไลบลารี่พร้อมใช่งานกับ Arduino หรือ NodeMCU มีราคาถูก และสามารถใช้วัดค่าได้เที่ยงตรงกว่ามาก เพราะให้ Output ออกมาในรูปแบบของดิจิตอล ใช้วัดอุณหภูมิอากาศโดยรอบ ปัจจุบันเป็นเซ็นเซอร์ซึ่งเป็นที่นิยมใช้เนื่องจากสามารถใช้งานได้ง่าย และวัดค่าออกมาได้อย่างมีประสิทธิภาพ ซึ่งเซ็นเซอร์นี้จะมีสองประเภท DHT11 และ DHT22 ภายในเป็น Resistive Humidity Sensor (สำหรับวัดความชื้นในอากาศ) และ Thermistor (สำหรับวัดอุณหภูมิ)

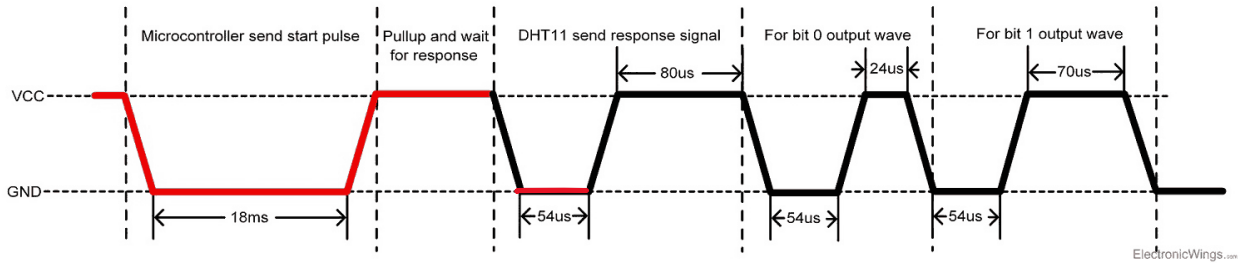




	DHT11	DHT22
Operating Voltage	3 to 5V	3 to 5V
Max Operating Current	2.5mA max	2.5mA max
Humidity Range	20-80% / 5%	0-100% / 2-5%
Temperature Range	0-50°C / ± 2°C	-40 to 80°C / ± 0.5°C
Sampling Rate	1 Hz (reading every second)	0.5 Hz (reading every 2 seconds)
Body size	15.5mm x 12mm x 5.5mm	15.1mm x 25mm x 7.7mm
Advantage	Ultra low cost	More Accurate



ในการเชื่อมต่อกับบอร์ดทดลองนั้นจะใช้เพียง 1 สาย (one wire) ในการสื่อสารเชื่อมต่อ ค่าต่าง ๆ หรือระดับแรงดันจะเปลี่ยนแปลงไปตามค่าลอจิก 0 หรือ 1 ที่ส่งมากับสาย โดยในการสื่อสารนั้นจะมีทั้งหมดสามขั้นตอนคือ จะส่ง request ไปยัง DHT11 จากนั้น เซ็นเซอร์จะส่ง response pulse กลับมา และเริ่มส่งข้อมูล 40 บิต ไปให้กับไมโครคอนโทรลเลอร์



00100101 **00000000** **00011001** **00000000** **00111110**

High Humidity | Low Humidity | High Temperature | Low Temperature | Checksum (Parity)

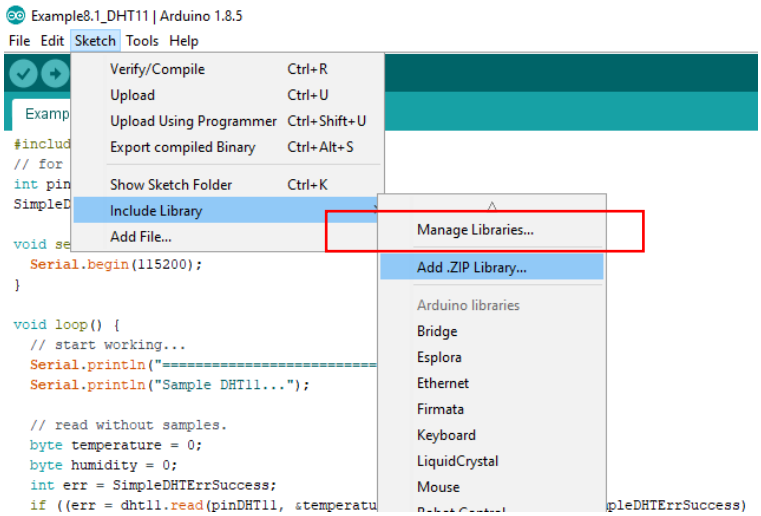
RH = Decimal of 00100101 = 37%

Temperature = Decimal of 00011001 = 25^{0}C

การทดลอง : การติดตั้งไลบลารี วัดอุณหภูมิ และความชื้นด้วย DHT11 Sensor

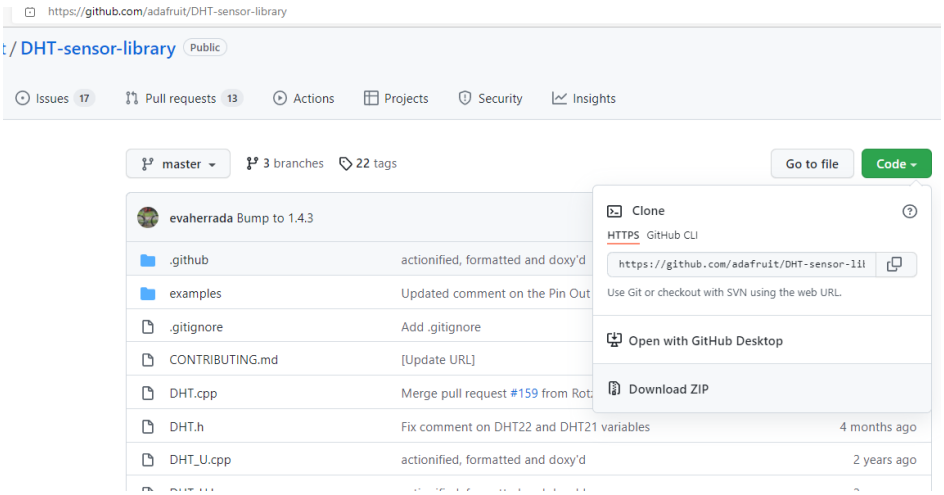
ในการใช้งาน ArduinoIDE มีข้อดีอย่างหนึ่ง คือ ไลบลารีที่สามารถใช้งานได้จากแหล่งต่าง ๆ มีมากมาย ซึ่งทำให้การพัฒนา Prototype ของระบบทำได้โดยง่าย และไม่ต้องเขียนโค้ดที่มีความยุ่งยาก

เพื่อให้สะดวกในการใช้งาน ทำการดาวน์โหลด Search จาก Arduino IDE โดย Sketch -> Include Libraries -> Manage Libraries

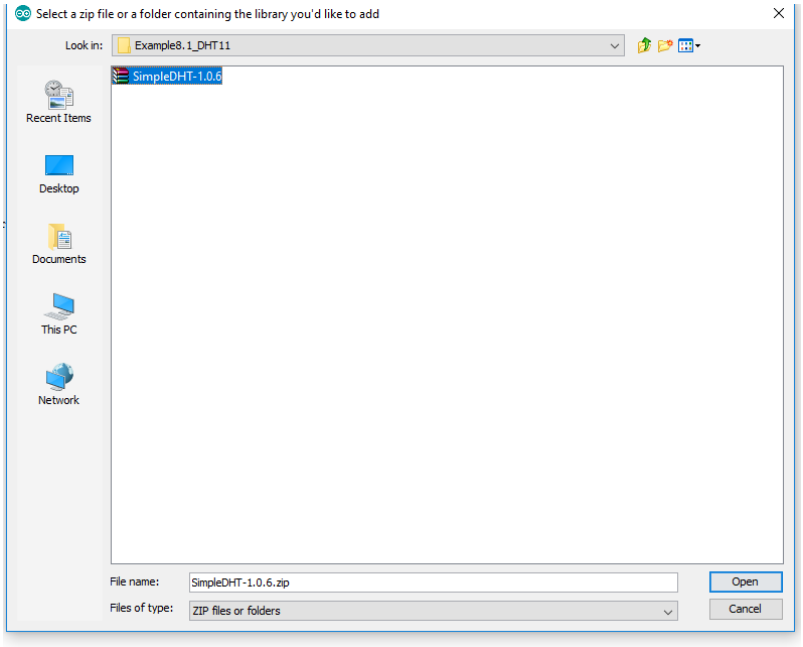


ในหน้าต่างที่ปรากฏให้ค้นหา DHT Sensor จากนั้นกดปุ่ม Install โดยจะมีการถามให้ติดตั้ง Lib เพิ่มเติมให้เลือก “Install All” รอจนติดตั้งเสร็จ (จนขึ้นตัวอักษรสีเขียวว่า Installed)

นอกจากนี้ ยังมีอีกวิธีหนึ่ง คือ การติดตั้งไลบลารีจากภายนอก **(**หมายเหตุ หากติดตั้งวิธีด้านบนไปแล้ว ไม่จำเป็นต้องติดตั้งวิธีนี้อีก)** การติดตั้งไลบลารี หากมีเป็น .ZIP ไฟล์จะสามารถเลือกติดตั้งได้จาก .ZIP ไฟล์ เช่น ดาวน์โหลดจาก https://github.com/adafruit/DHT-sensor-library



การติดตั้ง ทำได้โดยโปรแกรม Arduino IDE โดยในแถบเมนู ให้เลือกเมนู Sketch > Include Library > Add .Zip Library > เลือกไฟล์ Zip ที่ทำการดาวน์โหลดมา

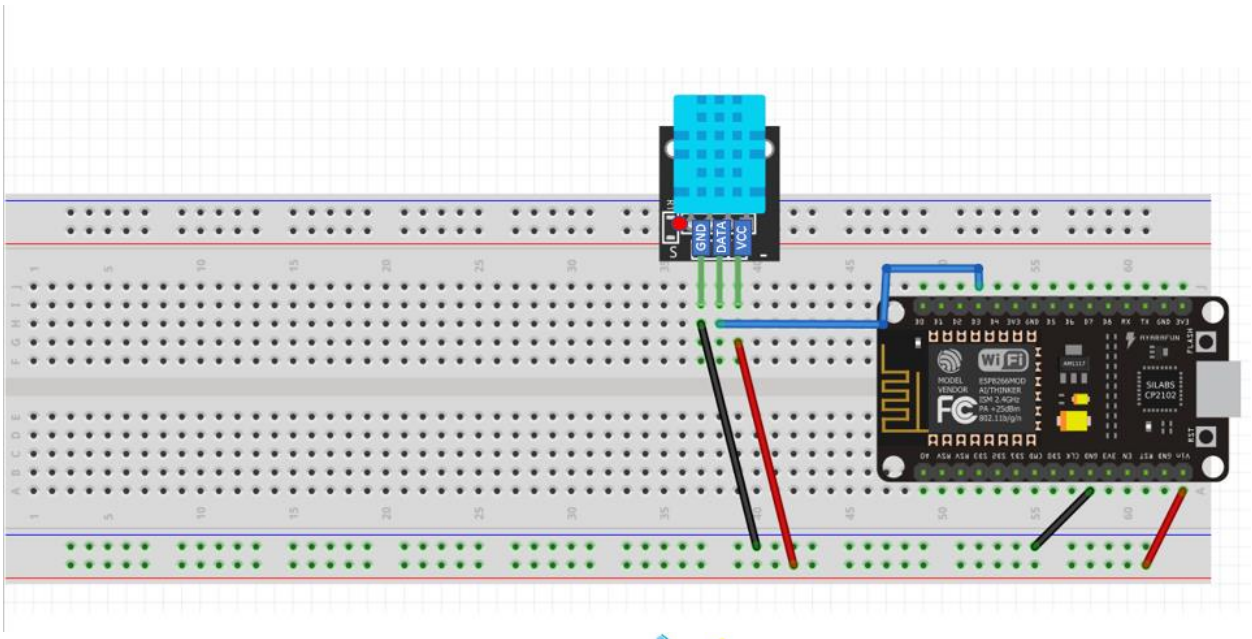


การเชื่อมต่อระหว่าง NodeMCU และ Sensor เชื่อมต่อตามตารางด้านล่างนี้

DHT11/DHT22	NodeMCU
VCC หรือ +	3V3
DATA หรือ OUT	D3 (GPIO0)
GND หรือ -	GND

** ถ้าตัวที่มี 4 ขา ขาที่สาม (นับจากหันหน้าอุปกรณ์เขาเรา) จะไม่ได้ใช้

***ทั้งนี้ บอร์ด DHT11 อาจมีการเชื่อมต่อขาหลายรูปแบบ ให้ดูสัญลักษณ์ที่ขาเป็นหลัก อย่าดูตามวงจรนี้เป็นหลัก



จากนั้นเบิร์นโค้ด ลงบอร์ด NodeMCU โดยเลือกโค้ดที่เบิร์นให้ถูกต้องว่าเลือกตัว DHT11 หรือ DHT22

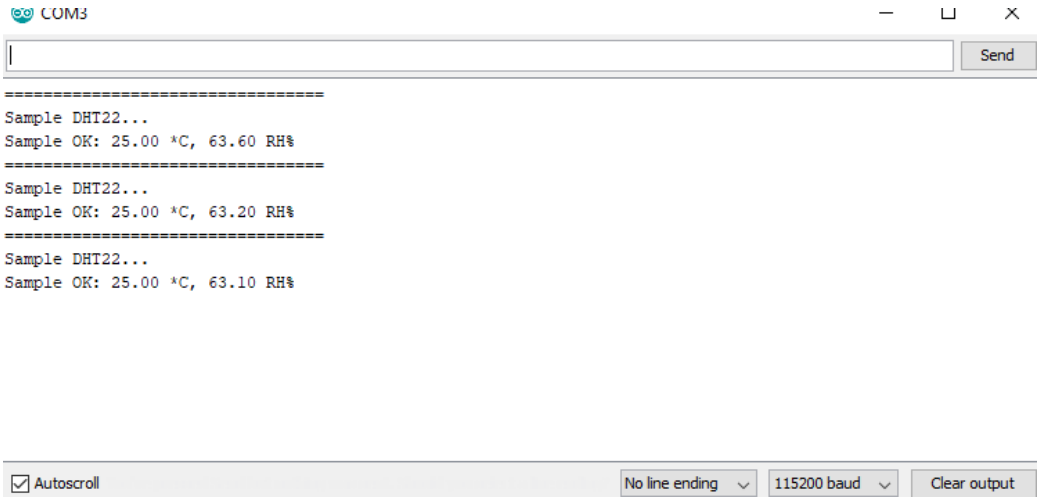
Example 4.1 NodeMCU – DHT11 (ตัวสีฟ้า)

```
#include "DHT.h"
#define DHTPIN D3   //D3  // Digital pin connected to the DHT sensor
// Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
#define DHTTYPE DHT11   // DHT 11


// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);


void setup() {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));
  dht.begin();
```

```
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  float h = dht.readHumidity();  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
```

```
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

จากนั้นทดลองเปิด Serial Monitor ดูผลลัพธ์ที่เกิดขึ้น



โจทย์ปัญหาท้ายการทดลอง

1. ให้ต่อ LED เพิ่มอีกหนึ่งดวง ทำการเขียนโปรแกรมให้การกดสวิตช์แบบกดติดกดดับ โดยให้กด-ปล่อย หนึ่งครั้งให้ LED ทั้งสามดวง สว่างพร้อมกัน 1 วินาที แล้วดับพร้อมกัน 1 วินาที หากกด-ปล่อยอีกครั้งหนึ่งให้ LED สว่าง 1 วินาทีแล้วดับ 1 วินาที ทีละดวง
2. เขียนโปรแกรมรับค่าเซ็นเซอร์ความสว่างของแสง(ด้วย LDR) และค่าอุณหภูมิและความชื้น (ด้วย DHT11) และเชื่อมต่อกับ LED สองดวง โดยหากมีค่าความสว่างมาก ให้ LED 1 ดวง ติด และ หากมีอุณหภูมิสูงกว่าค่าที่กำหนด (ค่าใดค่าหนึ่ง) ให้ LED ดวงที่สองติด หากอุณหภูมิสูงและมีค่าความสว่างด้วย ให้ LED ติดทั้งสองดวง ถ้าไม่มีความสว่าง ให้ LED ดับทั้งสองดวง

การส่งงาน

- **<u>ทำโจทย์ท้ายการทดลอง แล้วเรียกพี่ TA ตรวจ ส่ง</u>**

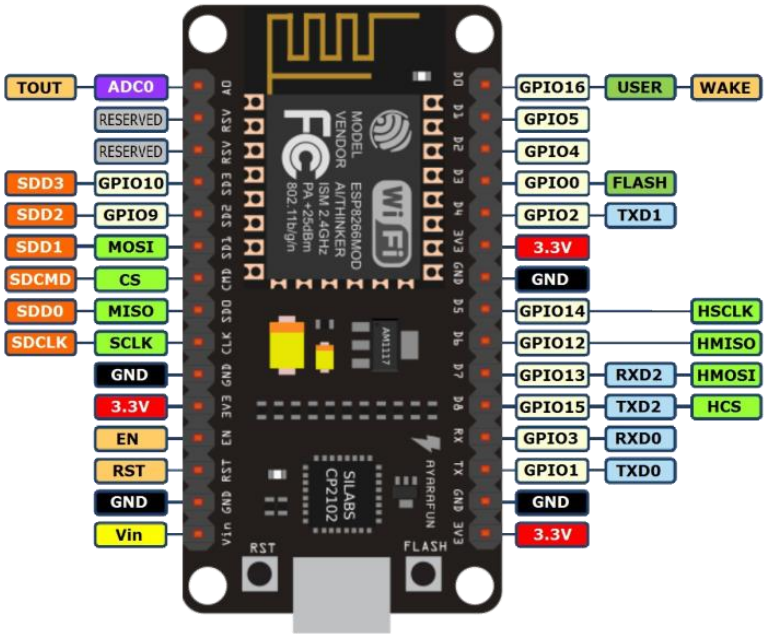
โจทย์พิเศษ (เพิ่ม 3 คะแนน) หากส่งก่อน 17.00 น.

1. ให้ต่อสวิตช์กับ LED สองดวง แล้วให้การกดสวิตช์แต่ละครั้งเป็นการเปลี่ยนโหมดการทำงานโดยมี 3 โหมด เมื่อกดเกินสามครั้งแล้วจะกลับมาโหมดที่หนึ่ง (Mode1 -> Mode2 -> Mode3 ->Mode1) โดยให้แต่ละโหมดแสดงฟังก์ชันการทำงานที่ต่าง ๆ กัน (เช่น Mode 1 ให้ไฟกะพริบพร้อมกันทุกๆ 1 วินาที, โหมดที่สองให้ไฟกะพริบพร้อมกัน ทุก ๆ 3 วินาที หรือ Mode 3 ให้ไฟกะพริบสลับกันติดดับดวงละ 1 วินาที)

Reference

การใช้งานพอร์ตต่าง ๆ ของ NodeMCU

สำหรับบอร์ด NodeMCU มีรายละเอียดของแต่ละขา ดังนี้



รูปที่ 1 ขาต่าง ๆ ของ NodeMCU ESP8266

ตารางที่ 4 NodeMCU Port Details and GPIO Map

Pin	Function	ESP-8266 Pin
TX	TXD	TXD

RX	RXD	RXD
A0	Analog input, max 3.3V input	A0,ADC
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
3V3	3.3V	3.3V
RST	Reset	RST

การส่งงาน

** ถ่ายวิดีโอผลลัพธ์ของคำถามท้ายการทดลอง พร้อมตอบคำถามในวิดีโอที่นักศึกษาถ่าย ทำการ Upload ลง youtube