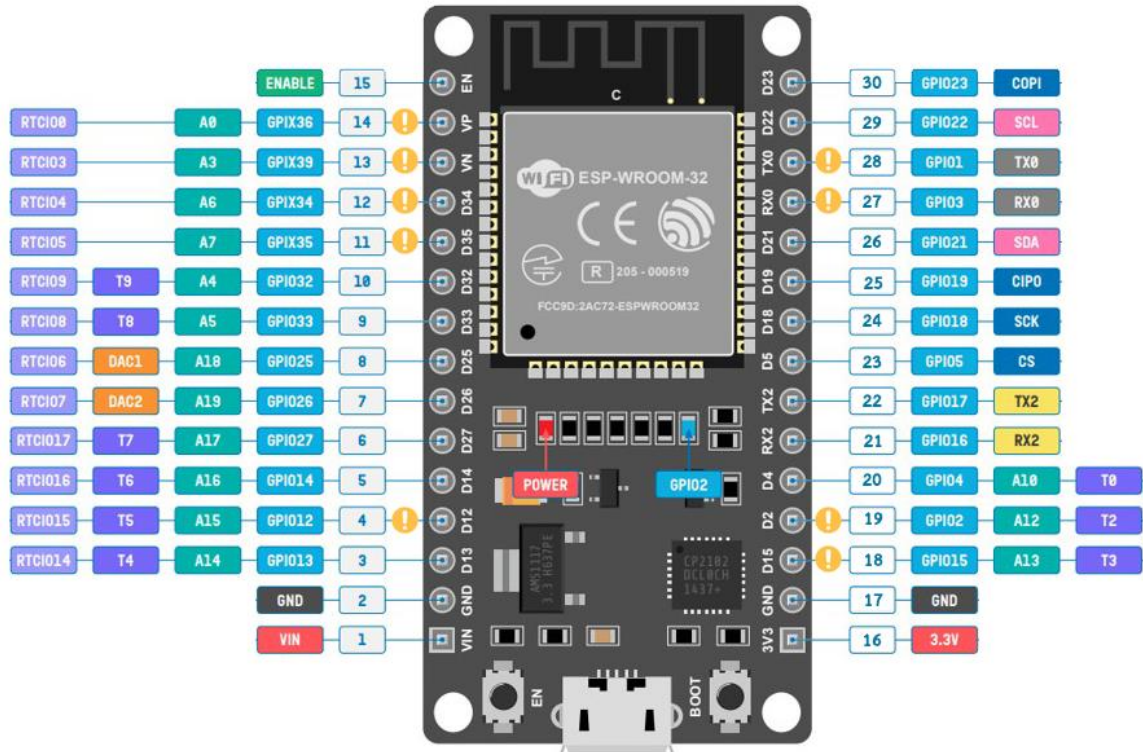


1. บอร์ด ESP32

1.1. ภาพประกอบบอร์ด ESP32 (30 ขา)



1.2. ตารางสรุปขาที่ใช้บนโมดูล ESP32 (30 ขา)

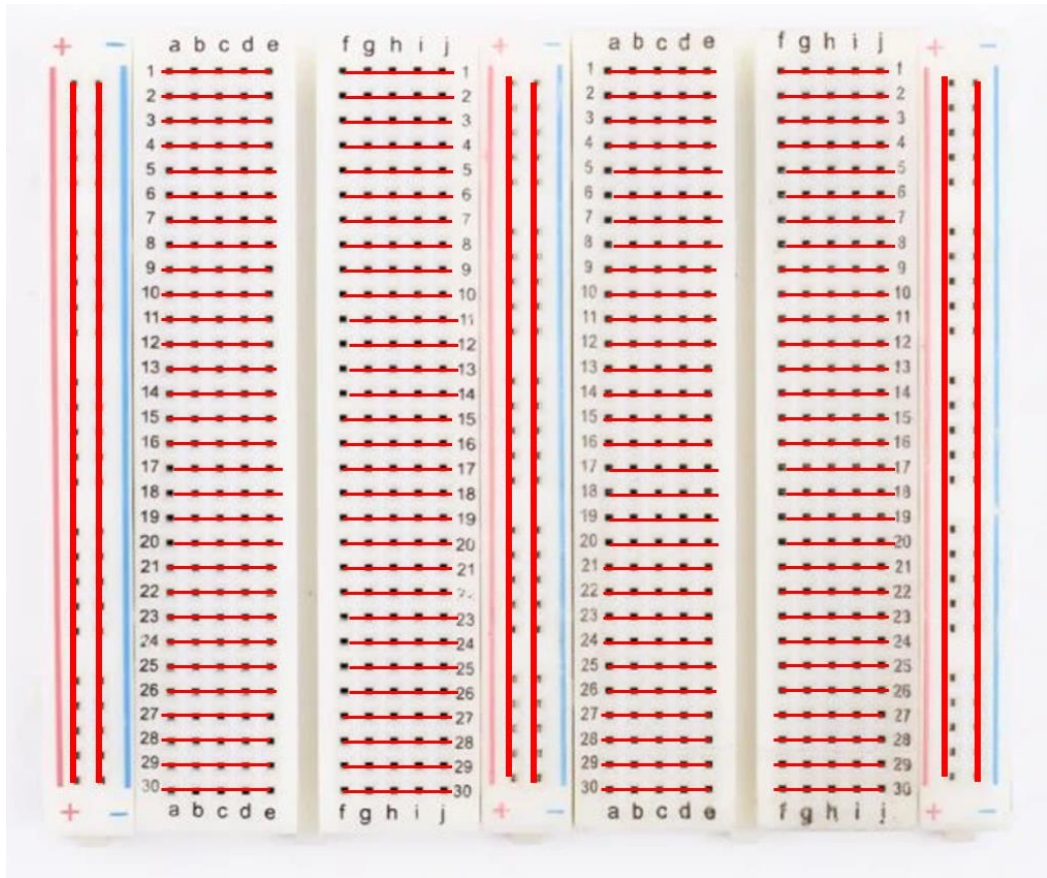
| หมายเลขขา | ขา GPIO | INPUT | OUTPUT | หมายเหตุ |
|-----------|---------|-------|--------|-------------------|
| 1 | - | - | - | VIN ไฟบวก 5 โวลต์ |
| 2 | - | - | - | GND |
| 3 | GPIO13 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 4 | GPIO12 | ได้ | ระวัง | ห้าม HIGH ตอนบูท |
| 5 | GPIO14 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 6 | GPIO27 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 7 | GPIO26 | ได้ | ได้ | DAC2 |
| 8 | GPIO25 | ได้ | ได้ | DAC1 |
| 9 | GPIO33 | ได้ | ได้ | ADC1 |
| 10 | GPIO32 | ได้ | ได้ | ADC1 |
| 11 | GPIO35 | ได้ | ไม่ได้ | Input only |

| | | | | |
|----|-------------|-----|----------------|-------------------------------------|
| 12 | GPIO34 | ได้ | ไม่ได้ | Input only |
| 13 | GPIO39 | ได้ | ไม่ได้ | ADC1 input only |
| 14 | GPIO36 | ได้ | ไม่ได้ | ADC1 input only |
| 15 | EN | - | - | Enable |
| 16 | - | - | - | 3.3V |
| 17 | - | - | - | GND |
| 18 | GPIO15 | ได้ | ระวัง | ต้อง LOW ตอนบูท |
| 19 | GPIO2 | ได้ | ได้ | ต้อง HIGH ตอนบูท แต่ใช้งานทั่วไปได้ |
| 20 | GPIO4 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 21 | GPIO16 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 22 | GPIO17 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 23 | GPIO5 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 24 | GPIO18 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 25 | GPIO19 | ได้ | ได้ | ใช้งานทั่วไปได้ |
| 26 | GPIO21 | ได้ | ได้ | I2C SDA |
| 27 | GPIO3 (RX0) | ได้ | ได้ (ไม่แนะนำ) | มีการรับ Serial ตอนบูท |
| 28 | GPIO1 (TX0) | ได้ | ได้ | ส่ง Serial ตอนบูท มีข้อความรบกวน |
| 29 | GPIO22 | ได้ | ได้ | I2C SCL |
| 30 | GPIO23 | ได้ | ได้ | ใช้งานทั่วไปได้ |

1.3. ข้อควรระวังในการใช้งานขา GPIO

| ขา GPIO | ข้อควรระวัง |
|-------------|-------------------------------------|
| GPIO2 | ต้อง HIGH ตอนบูท แต่ใช้งานทั่วไปได้ |
| GPIO12 | ห้าม HIGH ตอนบูท |
| GPIO15 | ต้อง LOW ตอนบูท |
| GPIO35 | Input only |
| GPIO34 | Input only |
| GPIO39 | ADC1 input only |
| GPIO36 | ADC1 input only |
| GPIO3 (RX0) | ใช้ได้ ไม่แนะนำ |
| GPIO1 (TX0) | ใช้ได้ ไม่แนะนำ |

1.4. ภาพประกอบบอร์ดทดลอง



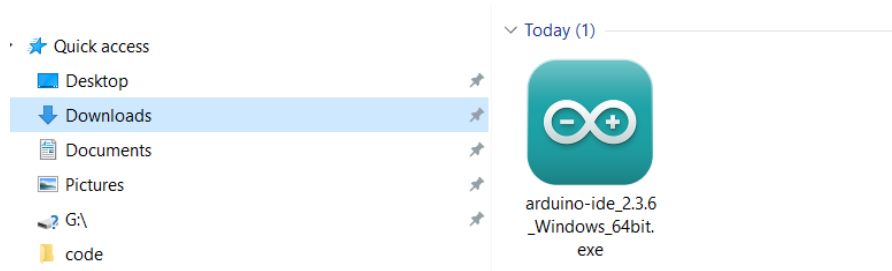
2. เตรียมเครื่องมือและติดตั้งโปรแกรม

2.1. ดาวน์โหลดโปรแกรม Arduino IDE

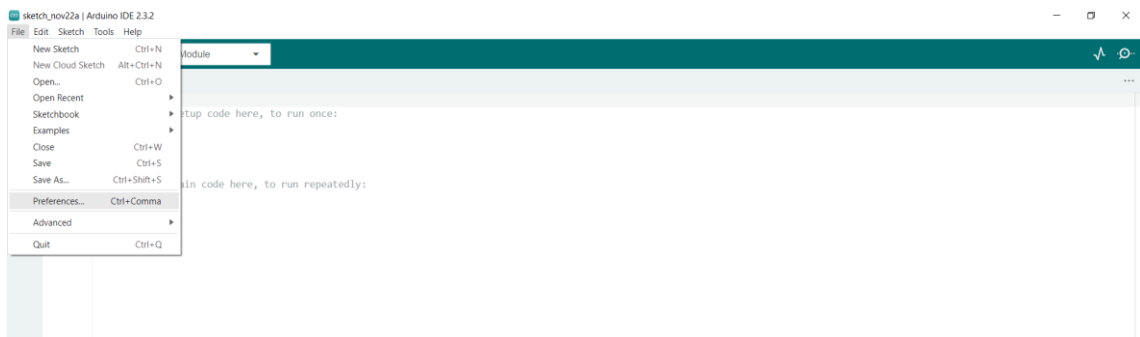
เข้าที่เว็บไซต์ Arduino : <https://www.arduino.cc/en/software/>

The screenshot shows the Arduino IDE 2.3.6 download page. The page has a header with the Arduino logo and navigation links: For Professionals, For Education, For Makers, Products, Community, Documentation, SHOP, and Cloud. The main heading is "Bring Your Projects to Life with Arduino Software". Below this is a large image of the Arduino IDE interface. To the right of the image is the "Arduino IDE 2.3.6 Release notes" section. It states: "The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger. For more details, check the [Arduino IDE 2.0 documentation](#)." Below this is a "DOWNLOAD" button. There is also a "Nightly Builds" section with a "Download a preview of the incoming release with the most updated features and bugfixes." link. At the bottom, it says "The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#)."

2.2. ติดตั้งโปรแกรม

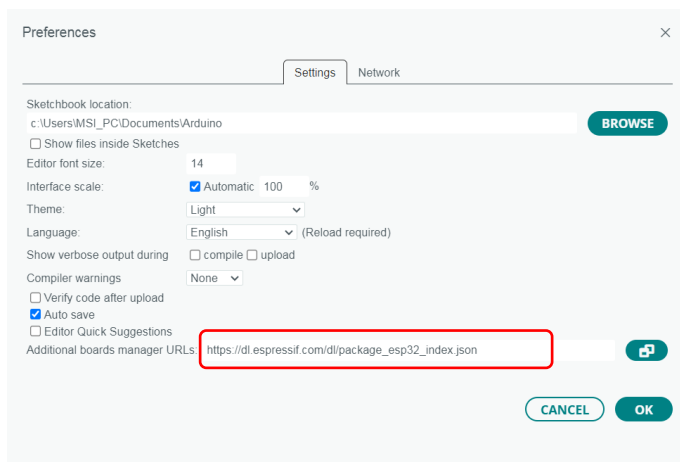


2.3. เปิดโปรแกรมขึ้นมาแล้ว ให้ไปที่เมนู File -> Preferences เพื่อติดตั้งบอร์ด ESP32

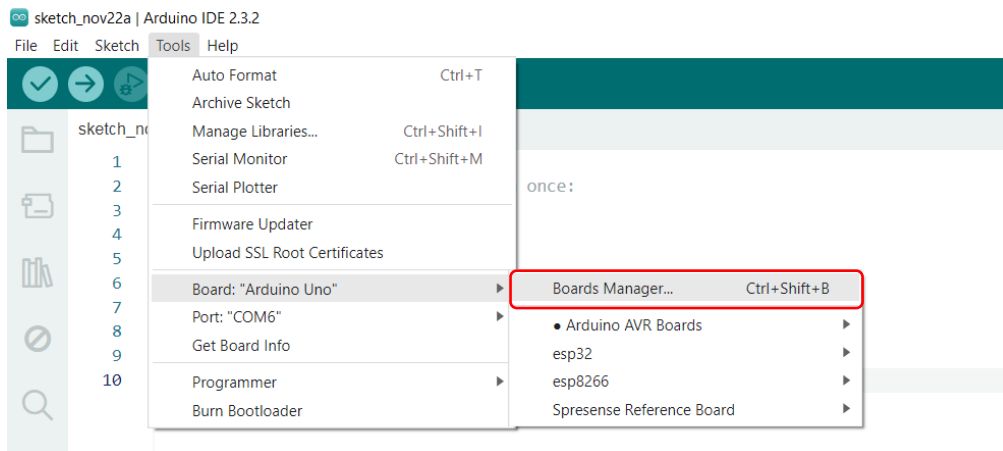


2.4. เพิ่ม https://dl.espressif.com/dl/package_esp32_index.json

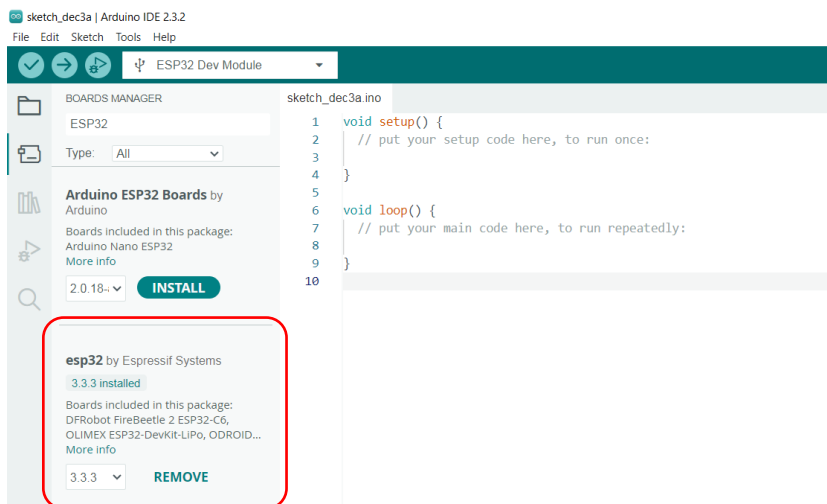
ลงในช่อง Additional Boards Manager



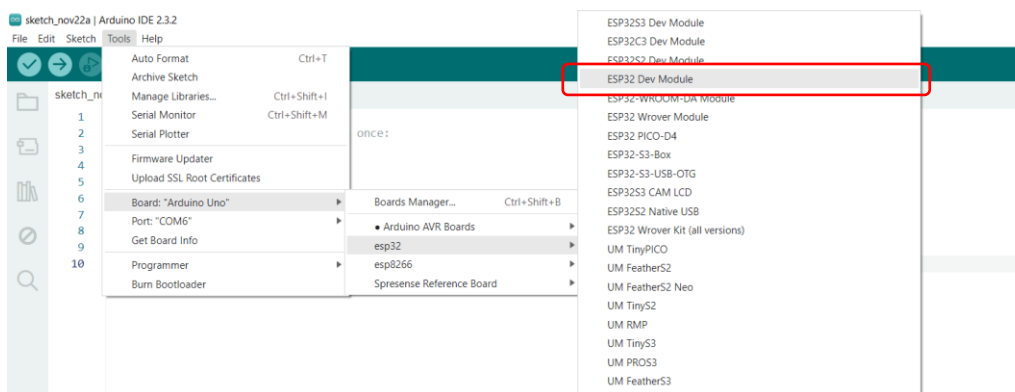
2.5. คลิกไปที่เมนู Tools -> Board -> Board Manager



2.6. พิมพ์คำว่า ESP32 เพื่อค้นหาในช่อง และเริ่มต้นติดตั้งดังภาพ



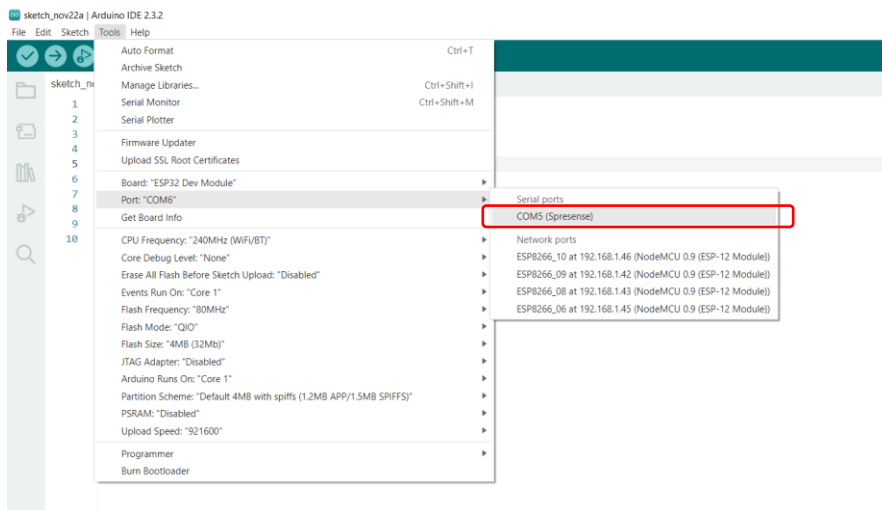
2.7. เมื่อติดตั้งเสร็จสิ้น ในหน้าต่าง Board ก็จะปรากฏประเภทของบอร์ด ESP32 ขึ้นมา และให้เลือกใช้งาน “ESP32 Dev Module”



2.8. เริ่มต้นใช้งานบอร์ด ESP32 โดยเชื่อมต่อสาย USB เข้ากับตัวบอร์ด



2.9. เลือก Port ที่เชื่อมต่อบอร์ด ESP32



เสร็จสิ้นขั้นตอนการติดตั้งโปรแกรม และพร้อมใช้งาน

3. ความรู้การเขียนโปรแกรมมิ่ง และคำสั่งที่ใช้งาน

3.1. ชนิดตัวแปร

| ชนิด | ข้อมูลที่เก็บ | ขนาด | ช่วงค่า |
|---------------|-------------------|--------|----------------|
| boolean | true/false | 1 byte | - |
| char | ตัวอักษร 1 ตัว | 1 byte | -128 ถึง 127 |
| byte | จำนวนเต็มไม่ติดลบ | 1 byte | 0 ถึง 255 |
| int | จำนวนเต็ม | 4 byte | -2.1B ถึง 2.1B |
| unsigned int | จำนวนเต็มบวก | 4 byte | 0 ถึง 4.29B |
| long | จำนวนเต็ม | 4 byte | -2.1B ถึง 2.1B |
| unsigned long | จำนวนเต็มบวก | 4 byte | 0 ถึง 4.29B |

| | | | |
|--------|------------------|---------|----------------|
| float | เลขทศนิยม | 4 byte | ประมาณ 7 หลัก |
| double | ทศนิยมละเอียดสูง | 8 byte | ประมาณ 15 หลัก |
| void | ไม่มีค่า | - | - |
| String | ข้อความ | dynamic | - |

3.2. การประกาศตัวแปร

3.2.1. รูปแบบการประกาศตัวแปร

| | |
|------|--------------------------------------|
| | ชนิดข้อมูล ชื่อตัวแปร; |
| หรือ | ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น; |

3.2.2. กฎในการตั้งชื่อตัวแปร

- ขึ้นต้นด้วย ตัวอักษร หรือ _ (underscore) ได้
- ห้ามขึ้นต้นด้วยตัวเลข
- ใช้ตัวอักษร a-z, A-Z, 0-9 และ _ ได้
- ห้ามใช้ชื่อที่เป็น Keyword ของ C/C++ เช่น if, while, for, return, void, function
- ตัวพิมพ์เล็ก-ใหญ่ ถือว่าเป็นคนละตัว เช่น temp ≠ Temp

3.2.3. การประกาศตัวแปรแบบกำหนดค่าในบรรทัดเดียว

ตัวอย่าง

```
int led = 2;
```

```
boolean status = false;
```

3.2.4. การประกาศตัวแปรแบบประกาศก่อน กำหนดค่าทีหลัง

ตัวอย่าง

```
int led;
boolean status;
led = 2;
status = false;
```

3.2.5. การประกาศตัวแปรหลายตัวในบรรทัดเดียว

ตัวอย่าง

```
int a, b, c;
```

```
int a = 10, b = 20, c = 30;
```

3.2.6. การประกาศตัวแปรแบบ Global มองเห็นได้ทั้งโปรแกรม

ตัวอย่าง

```
int counter = 0;    // Global
void setup() {
  Serial.begin(115200);
}
void loop() {
  counter++;
  Serial.println(counter);
}
```

ตัวแปร counter เข้าถึงได้ในทุกฟังก์ชันเหมาะกับค่าที่ต้องใช้ร่วมกันหลายส่วนของโปรแกรม

3.2.7. ตัวแปรแบบ Local มองเห็นเฉพาะในฟังก์ชัน

ตัวอย่าง

```
void setup() {
  Serial.begin(115200);
}
void loop() {
  int counter = 10;    // Local
  Serial.println(counter);
}
```

ตัวแปร counter ใช้ได้เฉพาะในฟังก์ชัน loop เท่านั้น นอกฟังก์ชันจะใช้ไม่ได้

3.2.8. ตัวแปรแบบ const (ค่าคงที่ เปลี่ยนค่าไม่ได้)

ใช้แทนค่าที่ไม่ควรถูกเปลี่ยน เช่น หมายเลขขา หรือค่าคงที่ทางคณิตศาสตร์
ตัวอย่าง

```
const int LED_PIN = 2;
const float PI = 3.14159;
```

3.2.9. ตัวอย่างการประกาศตัวแปร

ตัวอย่าง

```
boolean status = false;
int count;
int pin = 4;
float temperature = 25.5;
float humidity;
char letter = 'A';
String message;
```

3.3. คำสั่ง if else เช็คเงื่อนไข

ใช้สำหรับตรวจสอบว่าเงื่อนไขเป็น จริง (true) หรือ เท็จ (false) เพื่อตัดสินใจทำหรือไม่ทำ

รูปแบบ

```
if (เงื่อนไข) {
    // คำสั่งเมื่อเงื่อนไขเป็นจริง
} else {
    // คำสั่งเมื่อเงื่อนไขเป็นเท็จ
}
```

ตัวอย่าง

```
int a = 4;
int b = 5;
void setup() {
    Serial.begin(115200);
    if( a > b ) {
        Serial.println("a more than b");
    } else {
        Serial.println("a less than b");
    }
}
void loop() {
}
```

3.4. คำสั่งวนลูป **while**, **for**

3.4.1. คำสั่งวนลูป **while**

ใช้เมื่อ “ยังไม่รู้จำนวนครั้งที่แน่นอน” แต่ต้องการให้ทำงานกว่าเงื่อนไขจะเป็นเท็จ

รูปแบบ

```
while (เงื่อนไข) {
    // คำสั่งทำซ้ำ
}
```

ตัวอย่าง รอกจนกว่าปุ่มจะถูกกด

```
while (digitalRead(5) == LOW) {
    Serial.println("รอกกดปุ่ม...");
    delay(200);
}
Serial.println("กดปุ่มแล้ว!");
```

3.4.2. คำสั่งวนลูป **for**

ใช้เมื่อ “รู้จำนวนครั้งที่แน่นอน” แต่ต้องการให้ทำงานกว่าเงื่อนไขจะเป็นเท็จ

รูปแบบ

```
for (ตัวแปรเริ่มต้น; เงื่อนไข; การเปลี่ยนค่า) {
    // คำสั่งที่ต้องการทำซ้ำ
}
```

ตัวอย่าง พิมพ์ตัวเลข 1 ถึง 5

```
for (int i = 1; i <= 5; i++) {
    Serial.println(i);
}
```

3.5. ตารางสรุปคำสั่งที่ใช้งานบ่อย

| คำสั่ง | คำอธิบาย | ตัวอย่าง |
|---|--|--|
| <code>Serial.begin(boaudrate)</code> | ใช้การสื่อสารระหว่าง ESP32 กับ คอมพิวเตอร์ สามารถกำหนด boaudrate ความเร็วในการส่งข้อมูล โดยทั่วไป 2400 , 4800 , 9600 , 14400 , 38400 , 57600 , 115200 , 230400 , 460800 , 921600 | <code>Serial.begin(115200)</code> |
| <code>Serial.print(data)</code> | สั่งให้พิมพ์ข้อมูล data | <code>Serial.print("Hello Word")</code> |
| <code>Serial.println(data)</code> | สั่งให้พิมพ์ข้อมูล data พร้อมกับขึ้นบรรทัดใหม่ | <code>Serial.println("Hello Word")</code> |
| <code>pinMode(pin, mode)</code> | ใช้กำหนดโหมดของขา pin ให้เป็น INPUT รับค่าจากภายนอก ต้องต่อตัวต้านทาน เพิ่ม (เช่น ปุ่มกด เซนเซอร์) OUTPUT ส่งค่าออกไปภายนอก (เช่น LED, Relay, Buzzer) INPUT_PULLUP รับค่าจากภายนอก ใช้ตัวต้านทาน Pull-Up ภายใน | <code>pinMode(2, OUTPUT)</code> |
| <code>digitalWrite(pin, value)</code> | ใช้เขียนค่าดิจิทัลออกไปที่ขา pin HIGH = 1 (ไฟติด) LOW = 0 (ไฟดับ) | <code>digitalWrite(2, HIGH)</code> |
| <code>digitalRead(pin)</code> | อ่านค่าข้อมูลที่ขา pin เป็น HIGH หรือ LOW | <code>int button=digitalRead(2)</code> |
| <code>analogRead(pin)</code> | ใช้สำหรับอ่านค่าแรงดันไฟที่ขา pin อยู่ในช่วง 0-4095 ADC (Analog-to-Digital Converter) | <code>int button=analogRead(4)</code> |
| <code>analogWrite(pin, value)</code> | ใช้เขียนค่า analog ไปที่ขา pin ช่วงตั้งแต่ 0 ถึง 255 โดยที่ 0 คือไฟดับ และ 255 ไฟสว่างมาก | <code>digitalWrite(2, 120)</code> |
| <code>delay(ms)</code> | ใช้หยุดโปรแกรมตามเวลาที่กำหนด (มิลลิวินาที) 1000 ms = 1 วินาที ระหว่าง delay() โปรแกรมจะไม่ทำอย่างอื่นเลย | <code>delay(500)</code> // หยุดโปรแกรมครึ่งวินาที |
| <code>map(value, fromLow, fromHigh, toLow, toHigh)</code> | แปลงค่าจากช่วงหนึ่งไปสู่อีกช่วงหนึ่ง เช่น เซนเซอร์ให้ค่า 0-1023 ต้องแปลงให้อยู่ในช่วง 0-100 (%) | <code>int value_new = map(value_old, 0, 1023, 0, 100)</code> |

4. การใช้งาน Serial Monitor พิมพ์ Hello World

ใช้ทำอะไร ?

- Serial คือการสื่อสารระหว่าง ESP32 กับ คอมพิวเตอร์
- ใช้ Debug โปรแกรม แสดงค่าต่าง ๆ

ตัวอย่างโค้ดที่ 1 การแสดงข้อความ Hello World ผ่าน Serial Monitor

```
void setup() {
  Serial.begin(115200);
}
void loop() {
  Serial.println("Hello World");
  delay(1000);
}
```

ตัวอย่างโค้ดที่ 2 การรับค่าข้อความ และแสดงผลออกมา ผ่าน Serial Monitor

```
void setup() {
  Serial.begin(115200);
}
void loop() {
  if (Serial.available()) { // ตรวจสอบว่ามีข้อมูลเข้ามาหรือไม่
    String data = Serial.readString(); // อ่านค่าเข้ามา
    data.trim();
    Serial.print("Received: ");
    Serial.println(data); // แสดงผลค่าที่ได้รับได้
  }
}
```

5. การใช้งาน Digital Output

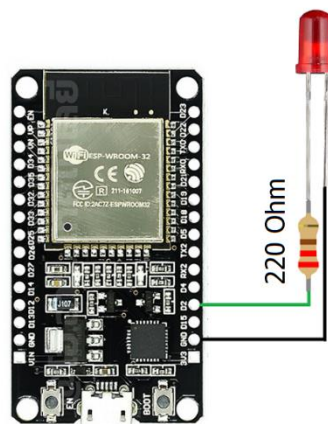
ใช้เขียนค่าดิจิทัลออกไปที่ขา pin โดยที่ HIGH = 1 (ไฟติด) และ LOW = 0 (ไฟดับ)

ใช้ทำอะไร ?

- ควบคุม LED
- เปิด-ปิด Relay หรือไฟบ้าน
- ส่งสัญญาณ ON/OFF

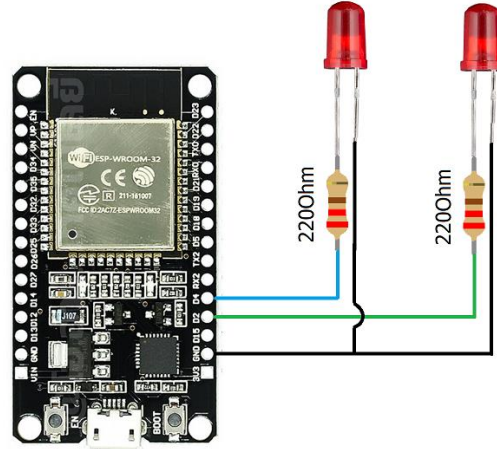
ตัวอย่างโค้ดที่ 3 ไฟกระพริบเปิด-ปิด 1 ดวง

```
int led = 2;
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(500);
  digitalWrite(led, LOW);
  delay(500);
}
```



ตัวอย่างโค้ดที่ 4 ไฟกระพริบสลับเปิด-ปิด 2 ดวง

```
int led1 = 2;
int led2 = 4;
void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}
void loop() {
  digitalWrite(led1, HIGH);
  digitalWrite(led2, LOW);
  delay(500);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  delay(500);
}
```



6. การใช้งาน Digital Input

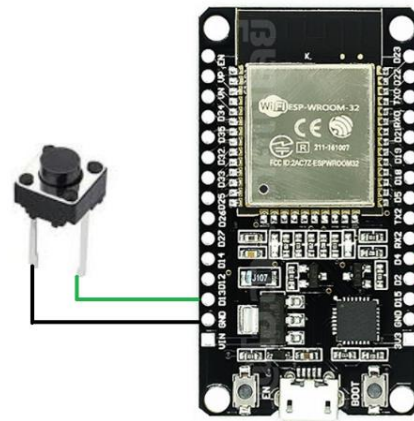
อ่านค่าข้อมูลที่ขา pin เป็น HIGH หรือ LOW

ใช้ทำอะไร ?

- ใช้ตรวจจับ ON/OFF
- ใช้ร่วมกับ Pull-up resistor

ตัวอย่างโค้ดที่ 5 การใช้ปุ่มกด

```
int button = 13;
void setup(){
  pinMode(button, INPUT_PULLUP);
  Serial.begin(115200);
}
void loop(){
  if(digitalRead(button) == LOW){
    Serial.println("Button Pressed");
  }
}
```



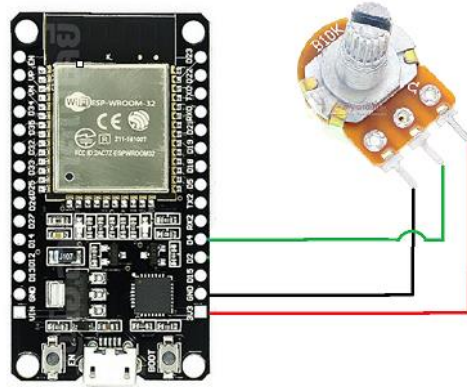
7. การใช้งาน Analog Input

ใช้ทำอะไร ?

- อ่านค่าจากตัวต้านทานปรับค่าได้
- อ่านค่าจากเซนเซอร์ต่าง ๆ

ตัวอย่างโค้ดที่ 6 การอ่านค่าตัวต้านทานปรับค่าได้

```
int r = 4;
void setup() {
  pinMode(r, INPUT);
  Serial.begin(115200);
}
void loop() {
  int value = analogRead(r);
  Serial.println(value);
  delay(100);
}
```



8. การใช้งาน Analog Output

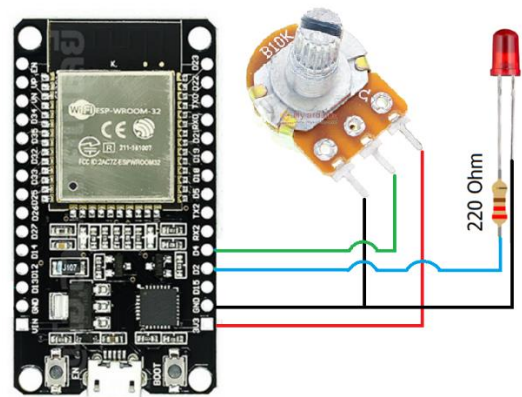
ปกติ ESP32 ไม่มี analogWrite จริง แต่เป็น PWM (LEDC)

ใช้ทำอะไร ?

- ควบคุมความเร็วมอเตอร์ การเพิ่มลดความสว่าง

ตัวอย่างโค้ดที่ 7 ควบคุมความสว่างหลอดไฟ LED

```
int r = 4;
int led = 2;
void setup() {
  pinMode(r, INPUT_PULLUP);
  pinMode(led, OUTPUT);
  Serial.begin(115200);
}
void loop() {
  int value = analogRead(r);
  int value_map = map(value, 0, 4095, 0, 255);
  Serial.print(value);
  Serial.print("\t");
  Serial.println(value_map);
  analogWrite(led, value_map);
  delay(100);
}
```



9. การใช้งาน WiFi

ใช้ทำอะไร ?

- แสดงผลค่าต่าง ๆ บนอินเทอร์เน็ต มือถือ หรือสั่งงานผ่านอินเทอร์เน็ต

9.1. ตัวอย่างโค้ดที่ 8 การเชื่อมต่อ WiFi

```
#include <WiFi.h>
const char* ssid = "iPhone 16";
const char* pass = "1234567890";
void setup(){
  Serial.begin(115200);
  WiFi.begin(ssid, pass);
  while(WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(500);
  }
  Serial.println("WiFi Connected.");
  Serial.print("IP = ");
  Serial.println(WiFi.localIP());
}
void loop(){}
```

9.2. ตัวอย่างโค้ดที่ 9 การเชื่อมต่อ WiFi ไปบันทึกลงบน Google Sheet

- ส่วนของ ESP32

```
#include <WiFi.h>
const char* ssid = "YOUR_SSID";
const char* pass = "YOUR_PASSWORD";
#include <HTTPClient.h>
void setup(){
  Serial.begin(115200);
  WiFi.begin(ssid, pass);
  while(WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(500);
  }
  Serial.println("WiFi Connected.");
  Serial.print("IP = ");
  Serial.println(WiFi.localIP());
  delay(1000);
  HTTPClient http;
  String url = "https://script.google.com/macros/s/AKfycbz-
    PJ4Has5tsriMicKAMR_chvnq4Mt310SSyvbze000feZbV-
    R0fsUxr4fz7_AgwTu9/exec?data=12.6";
  http.begin(url);
  int httpCode = http.GET();
  if(httpCode == 200) {
    String response = http.getString();
    Serial.print("Send Success : ");
    Serial.println(response);
  } else {
    Serial.println("Error code : " + httpCode);
  }
  http.end();
}
void loop(){ }
```

- ส่วนของ Google Sheet (App Script)

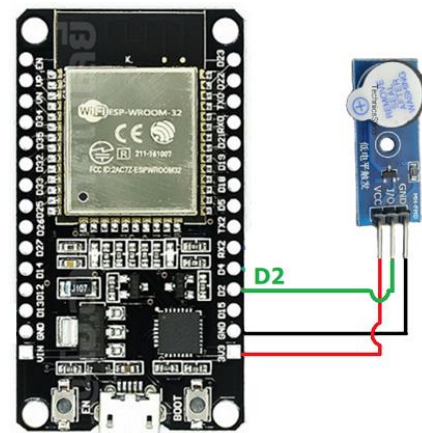
```
function doGet(e) {
  var data = e?.parameter["data"] ?? "";
  data = data.trim();
  if( data=="") {
    return ContentService.createTextOutput("Invalid Data");
  }
  var sheet_id = "1hYCBAlPCUsUyvWEIwbnFVSwy8sgm785BhuyYHOnCb0";
  var sheet = SpreadsheetApp.openById(sheet_id).getSheetByName("data");
  var lib = 'https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.18.1/moment.min.js';
  eval(UrlFetchApp.fetch(lib).getContentText());
  var date = moment().format("YYYY-MM-DD HH:mm:ss");
  sheet.appendRow([date, data]);
  return ContentService.createTextOutput("Save Success.");
}
```

10. การใช้งานเสียง Buzzer

ตัวอย่างโค้ดที่ 10 การเปิด-ปิดเสียงแบบง่าย Active Buzzer (Low Trigger)

```
int pin = 2;
void setup() {
  pinMode(pin, OUTPUT);
}

void loop() {
  digitalWrite(pin, HIGH);
  delay(500);
  digitalWrite(pin, LOW);
  delay(500);
}
```

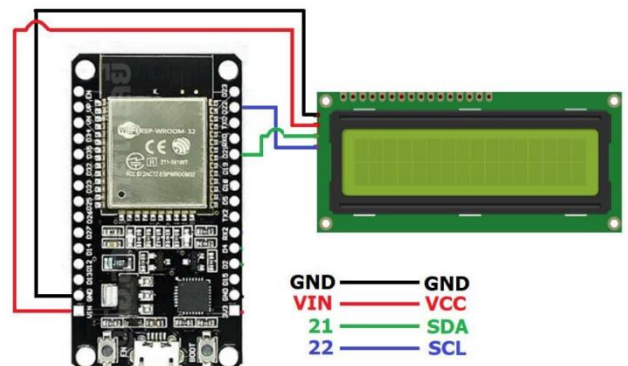


11. การใช้งานจอ LCD

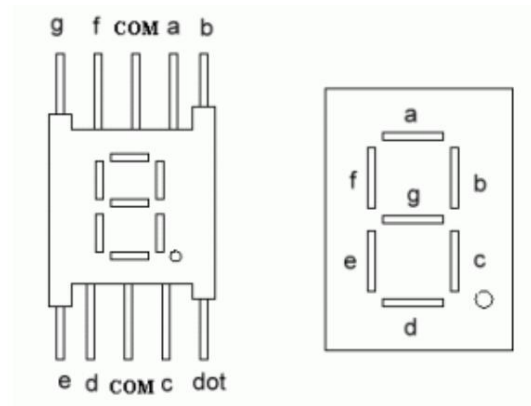
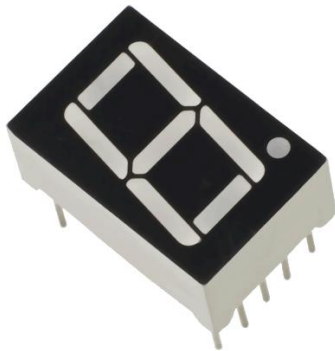
ตัวอย่างโค้ดที่ 11 การแสดงข้อความบนจอ LCD ขนาด 16x2

Library ที่ใช้: LiquidCrystal_I2C (by Martin...) v2.0.0

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);    // ไปที่ตัวอักษรที่ 0 แถว
  ที่ 1
  lcd.print("Welcome To");
  lcd.setCursor(6, 1);    // ไปที่ตัวอักษรที่ 6 แถว
  ที่ 2
  lcd.print("My ESP32");
}
void loop() {
```

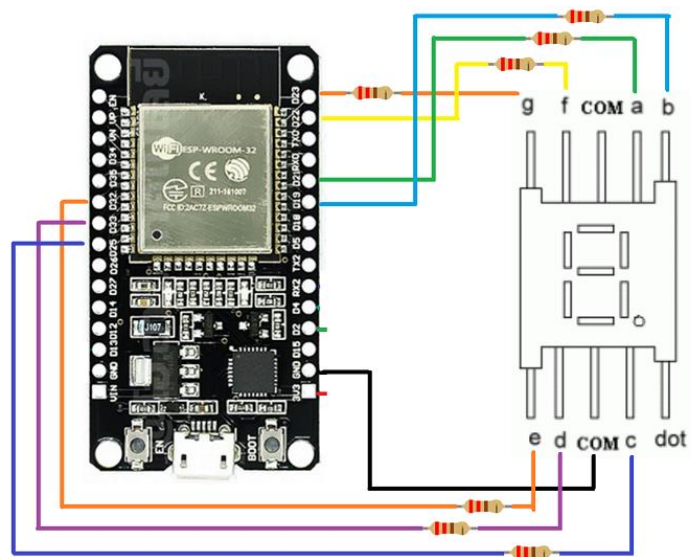


12. การใช้งาน 7-Segment



ตัวอย่างโค้ดที่ 12 แสดงผลตัวเลข 2 (7-Segment แบบลบร่วม Common Cathode)

```
int a = 21;
int b = 19;
int c = 25;
int d = 33;
int e = 32;
int f = 22;
int g = 23;
void setup() {
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
  two();
}
void loop() { }
void two() {
  digitalWrite(a, HIGH);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, HIGH);
  digitalWrite(e, HIGH);
  digitalWrite(f, LOW);
  digitalWrite(g, HIGH);
}
```



ใช้ R=220 โอห์ม ทั้ง 7 ตัว