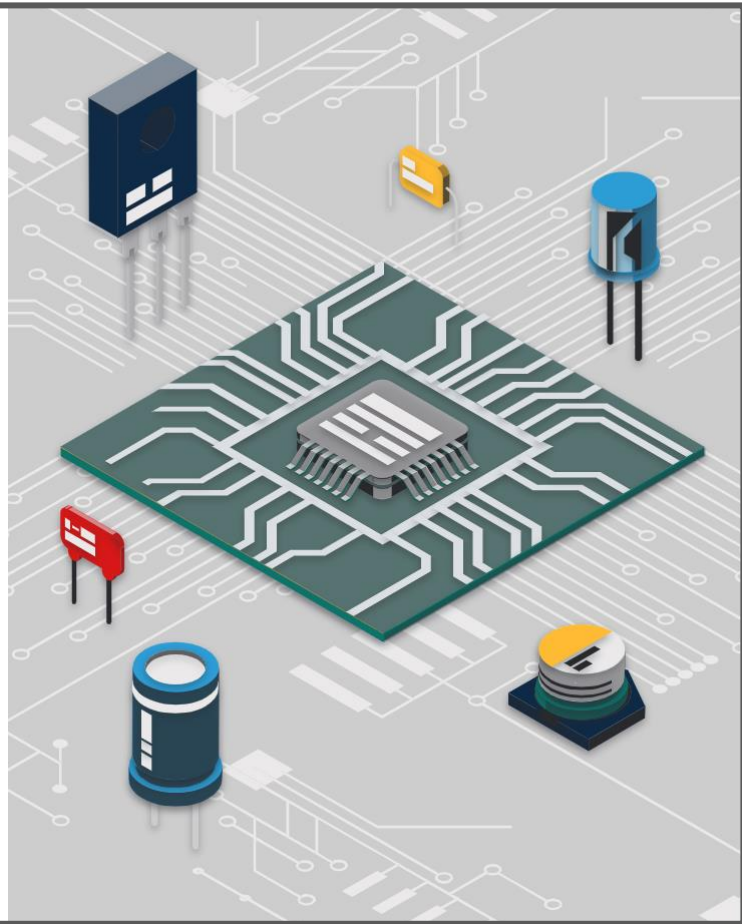


新知所網 NKFW 第四週

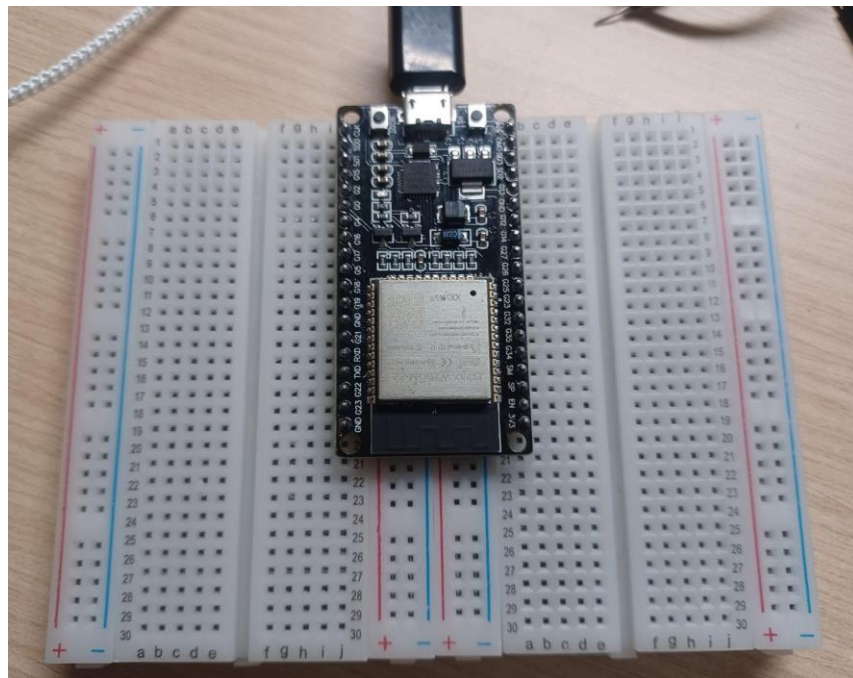
ESP32 應用

講師：鍋子

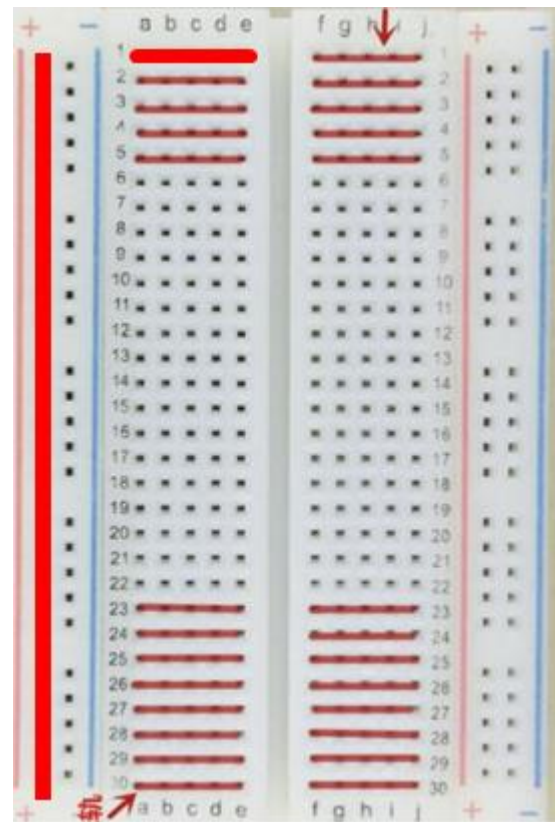
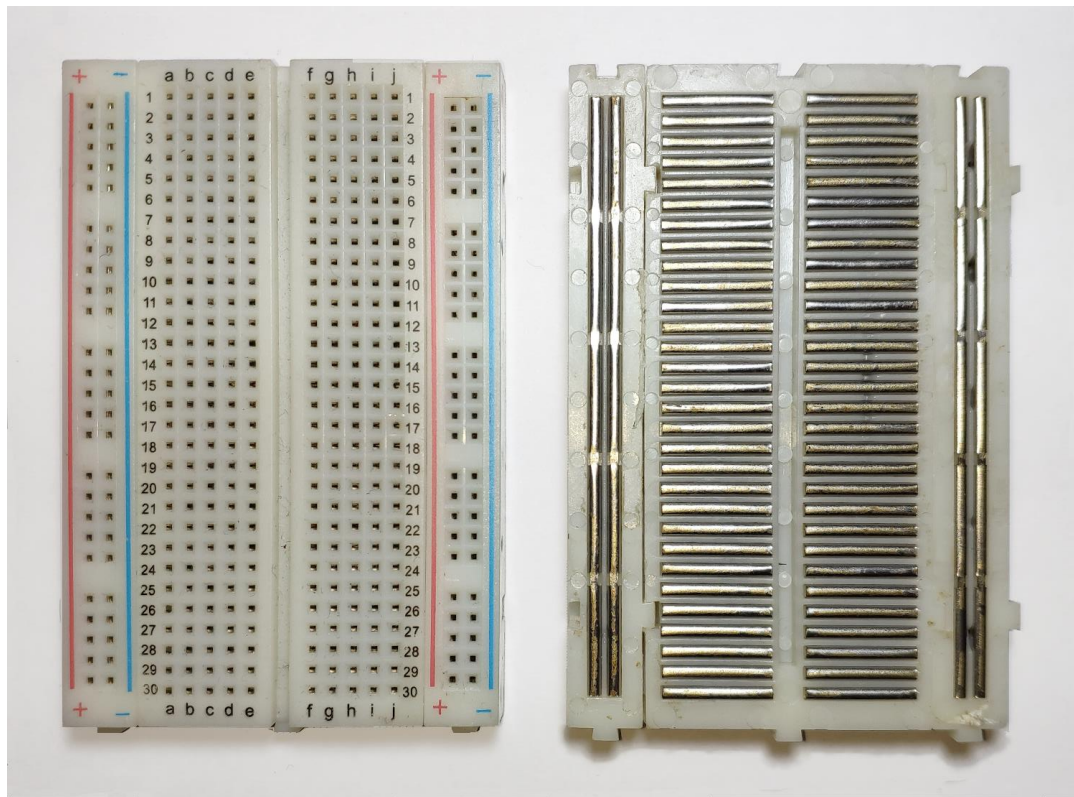


麵包板加工

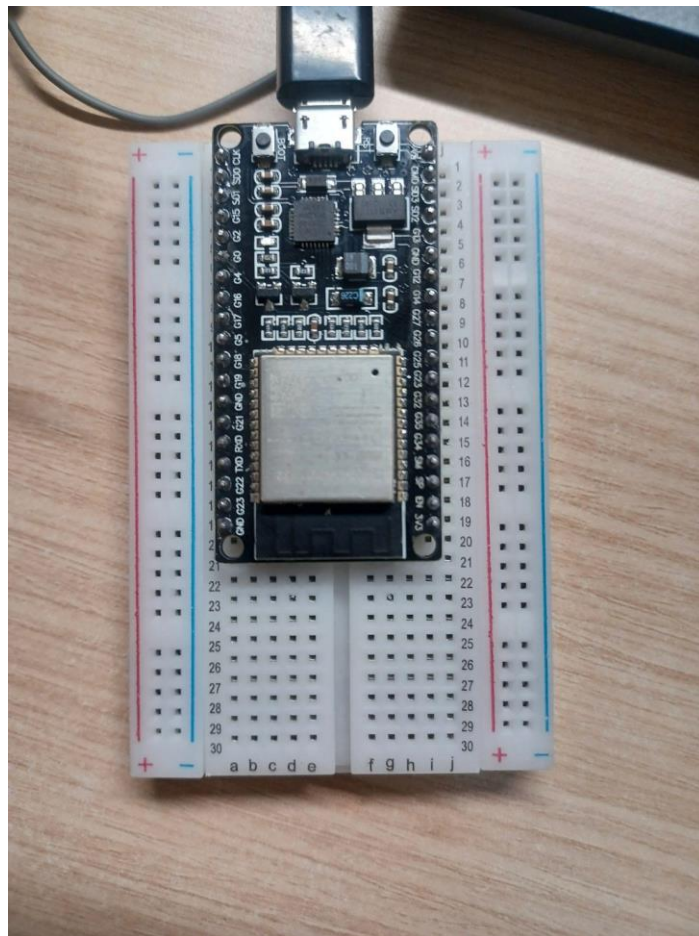
腦力激盪一下為什麼？



複習 麵包板構造



這尷尬的大小.....

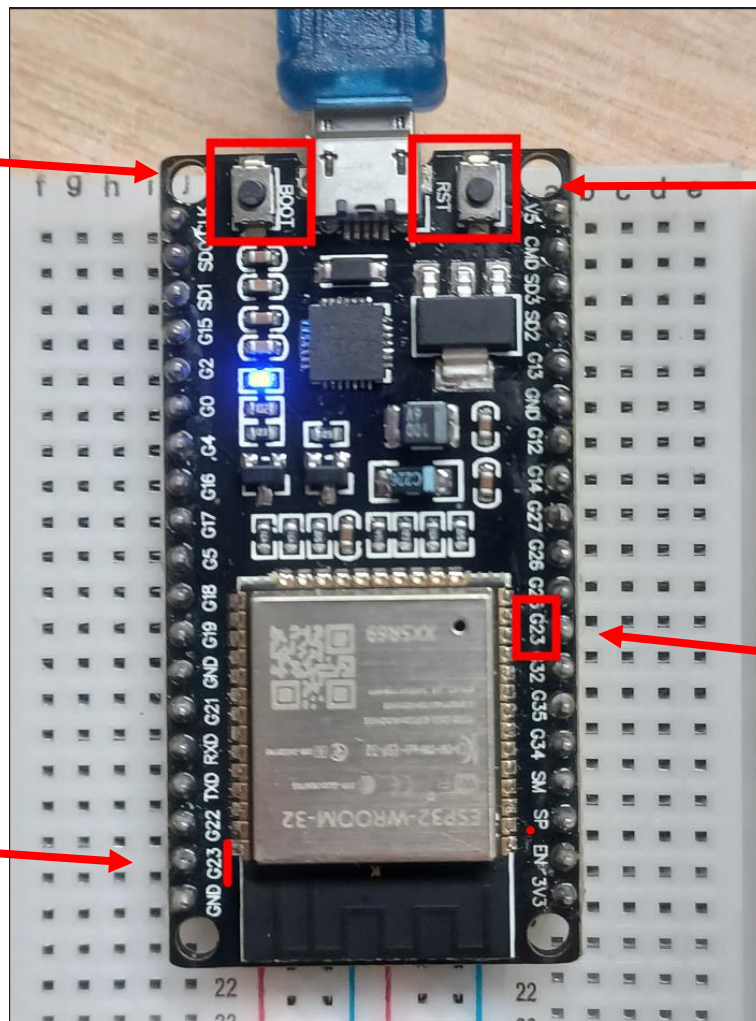


重新進入上傳模式
(按住在上傳)

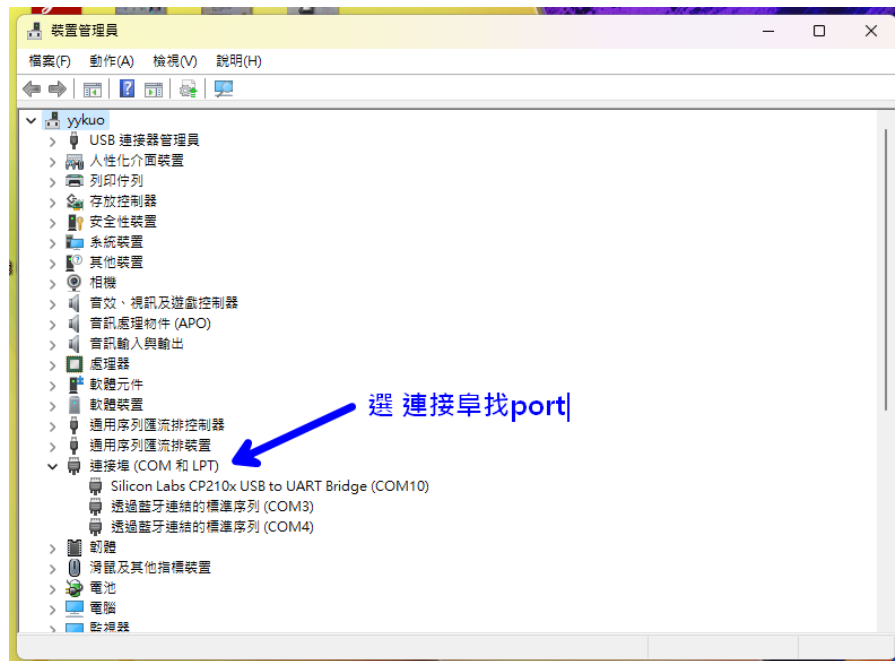
重製板子
使用新上傳的程式碼

板子製造錯誤

正確的



檢查 連接埠 esp32與電腦溝通的端口編號



開始建立專題

File Edit Selection View Go Run Terminal Help

main.cpp analogwriteapi - src main.cpp bme280 - src platformio.ini bme280 PIO Home platformio.ini servo main.cpp dht11 - src

PROJECT TASKS

QUICK ACCESS

- PIO Home
- Open
- PIO Account
- Inspect
- Projects & Configuration
- Libraries
- Boards
- Platforms
- Devices

Debug

- Start Debugging
- Toggle Debug Console

Miscellaneous

- Serial & UDP Plotter
- PlatformIO Core CLI
- Clone Git Project
- New Terminal
- Upgrade PlatformIO Core
- Show Release Notes

Home

Welcome to PlatformIO

Core 6.1.14 · Home 3.4.4

Quick Access

- + New Project
- Import Arduino Project
- Open Project
- Project Examples

Recent News

PlatformIO Labs · 3d

#LearnEmbedded · "Everything You Never Wanted To Know About Linker Script"

A great summary by Miguel Young de la Sota of the linker script language that you can use to build your

PlatformIO Labs · 6d

#BestPractices · "Debugging with PlatformIO: Part 2. Debugging an Embedded Target"

An introduction to debugging of embedded systems using PlatformIO Unified Debugger

PlatformIO Labs · 1w

#LearnEmbedded · "Understanding Computer Systems" by Patricio Bulic

Delve into the intricate world of interrupt handling and management across various processor architectures,

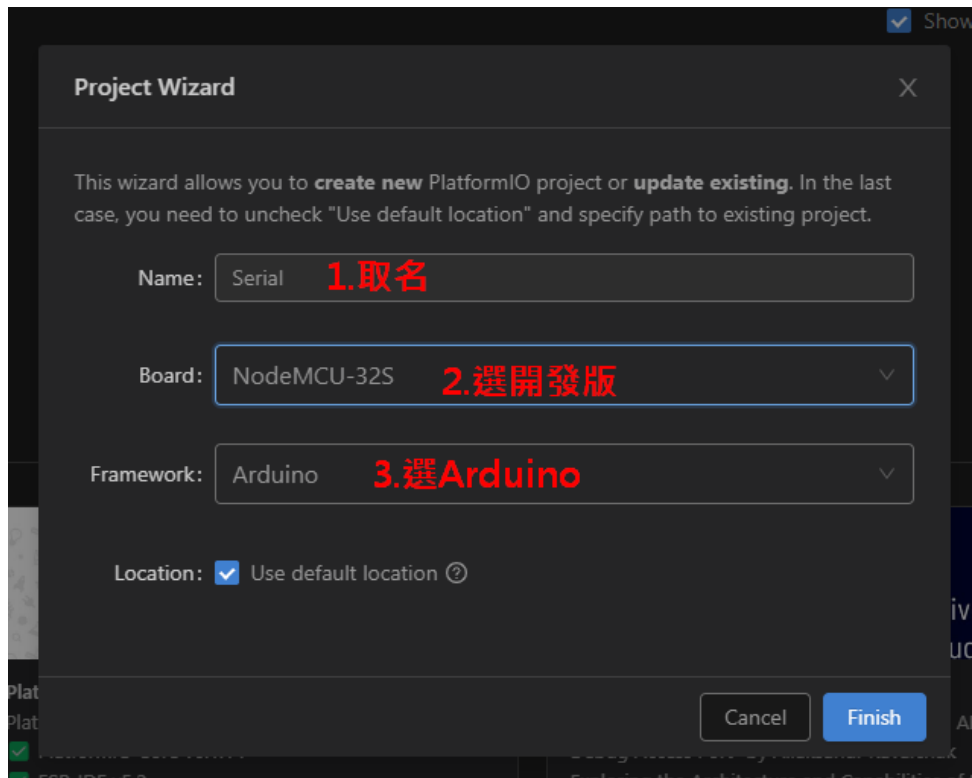
Recent Projects

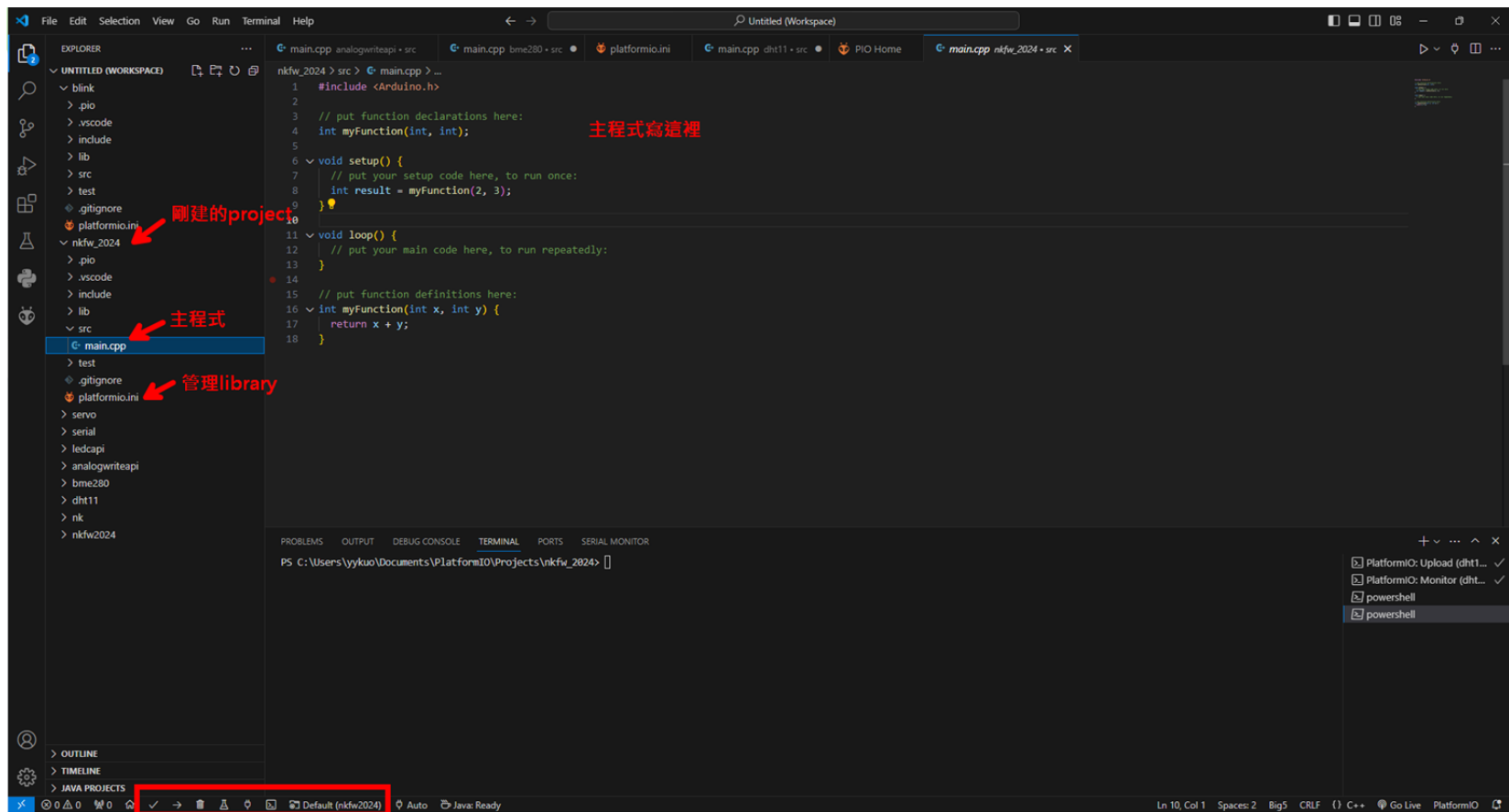
Search project...

Name	Boards	Modified	Action
Projects/dht11	NodeMCU-32S	4 hours ago	Hide Open
Projects/bme280	NodeMCU-32S	7 hours ago	Hide Open
Projects/analogwriteapi	NodeMCU-32S	16 hours ago	Hide Open

env:nodemcu-32s (servo) Auto Java: Ready

Go Live







1. 選單
2. 編譯:檢查程式是否有誤
3. 上傳:程式傳給開發版
4. 清除:結束上一個程式
5. 測試 (目前用不到)
6. 序列監視器
7. 專題環境

實作時間

Serial：訊息接收站

`Serial.begin(9600)` ; 設定鮑率 (電腦傳輸資料的頻率)

`Serial.print("MESSAGE")` ; 回傳的訊息

Serial : 訊息接收站

```
#include <Arduino.h>
```

```
void setup() //初始化
```

```
{
```

```
    Serial.begin(9600);
```

```
}
```

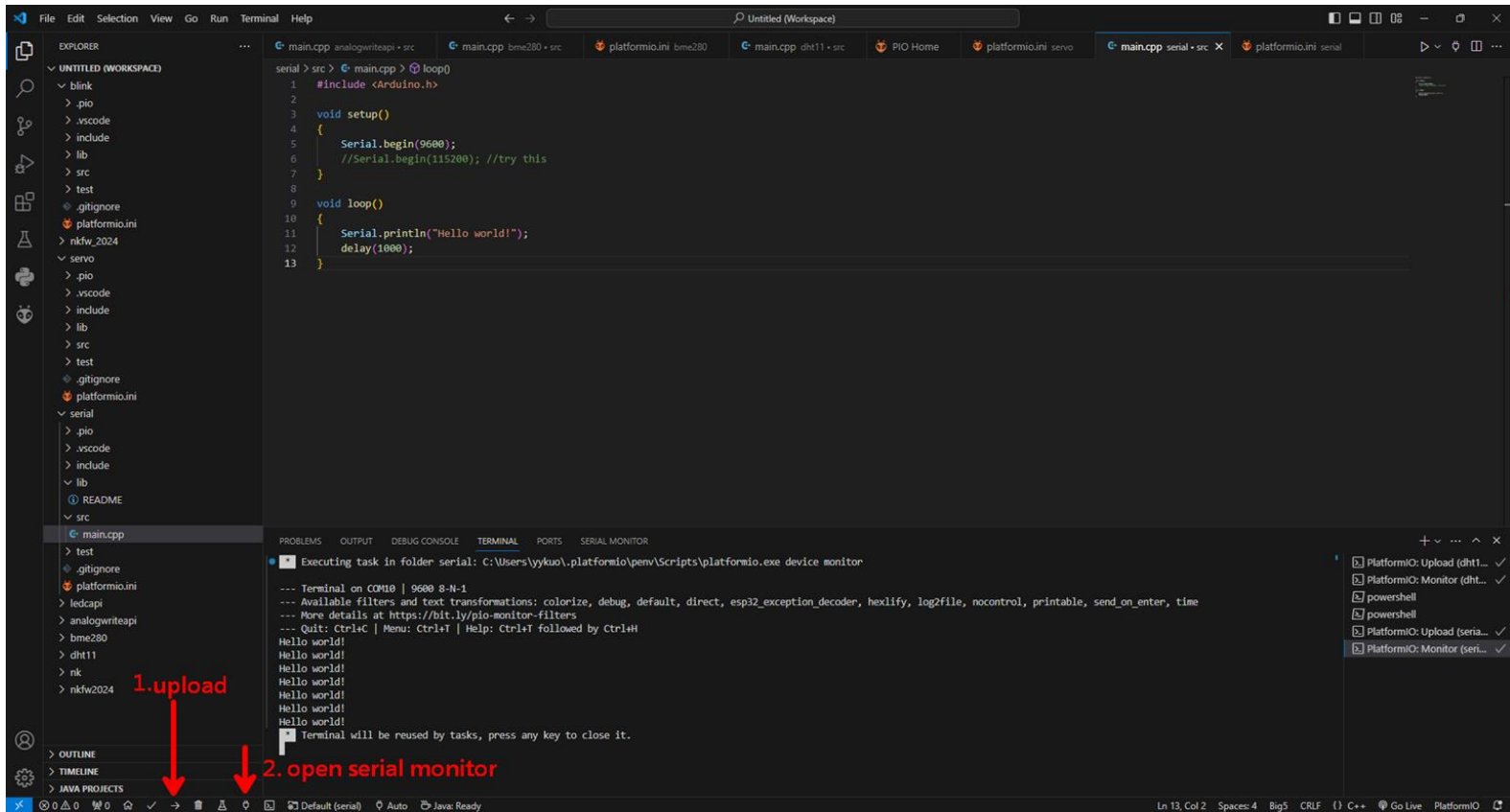
```
void loop() //主程式，不斷循環
```

```
{
```

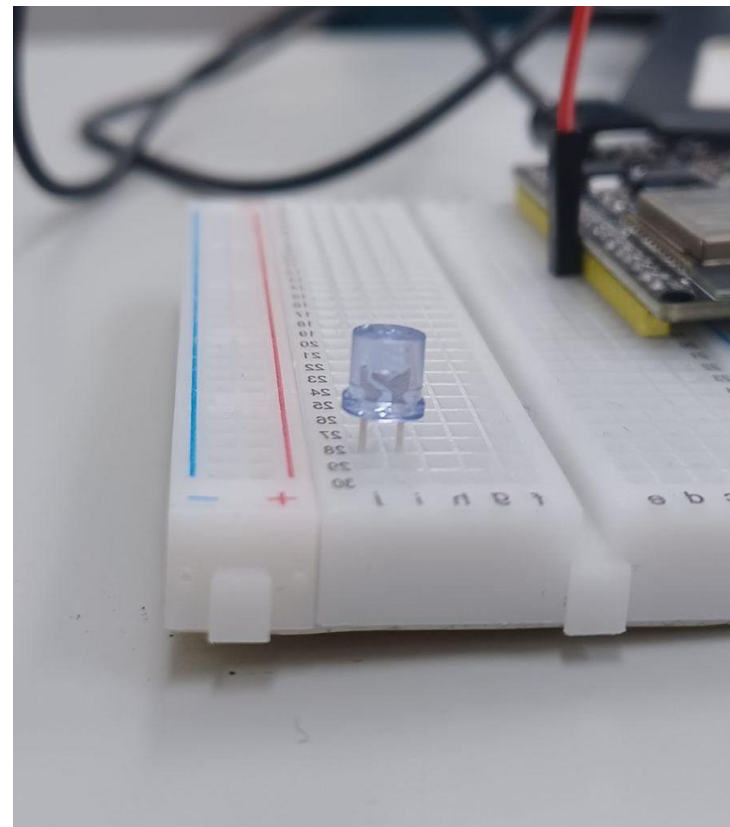
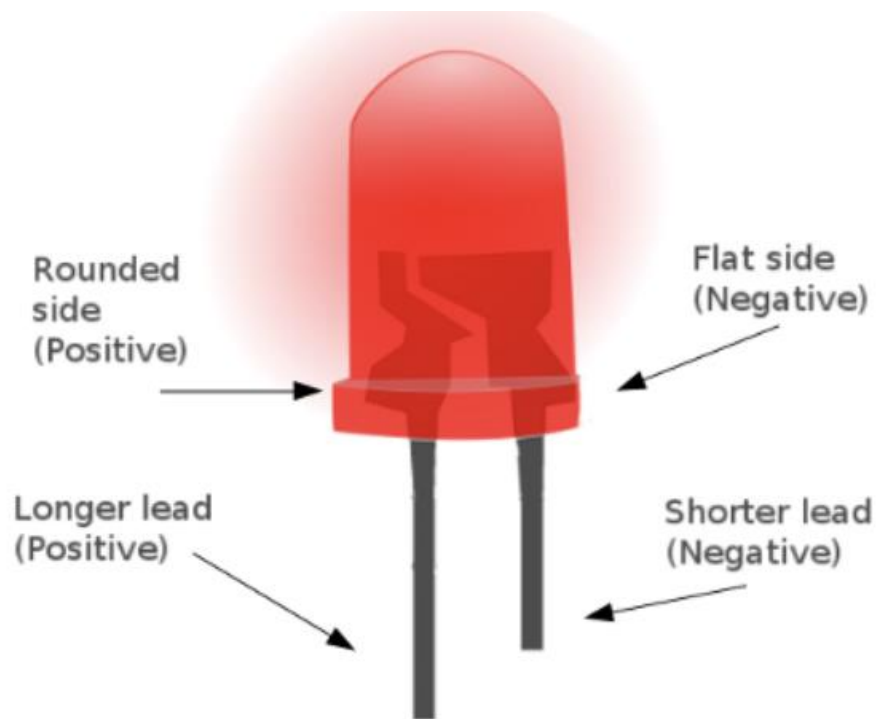
```
    Serial.println("Hello world!");
```

```
    delay(1000);
```

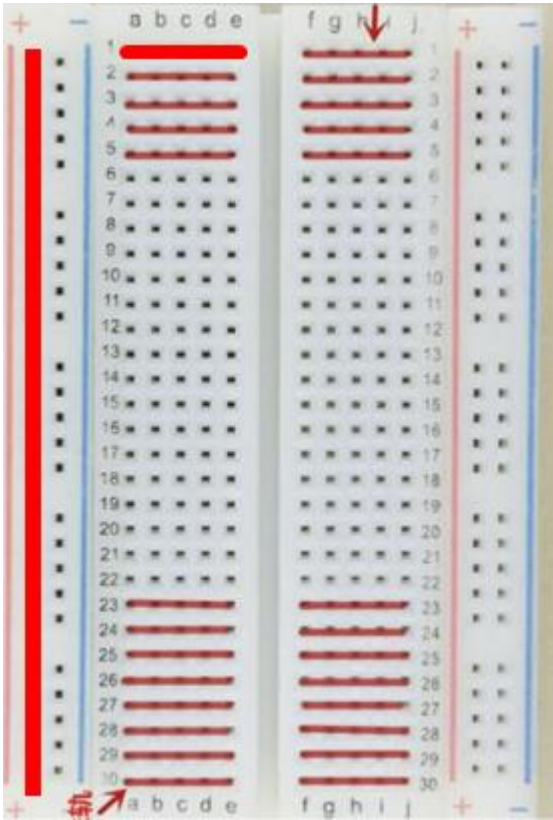
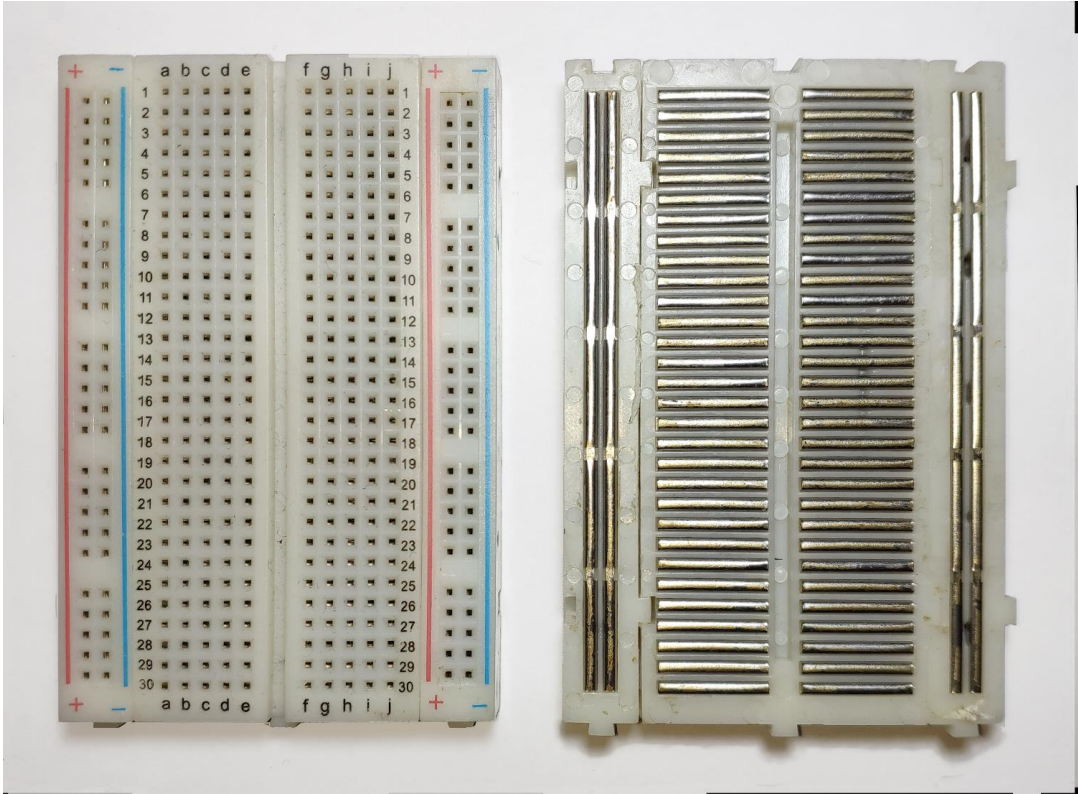
```
}
```



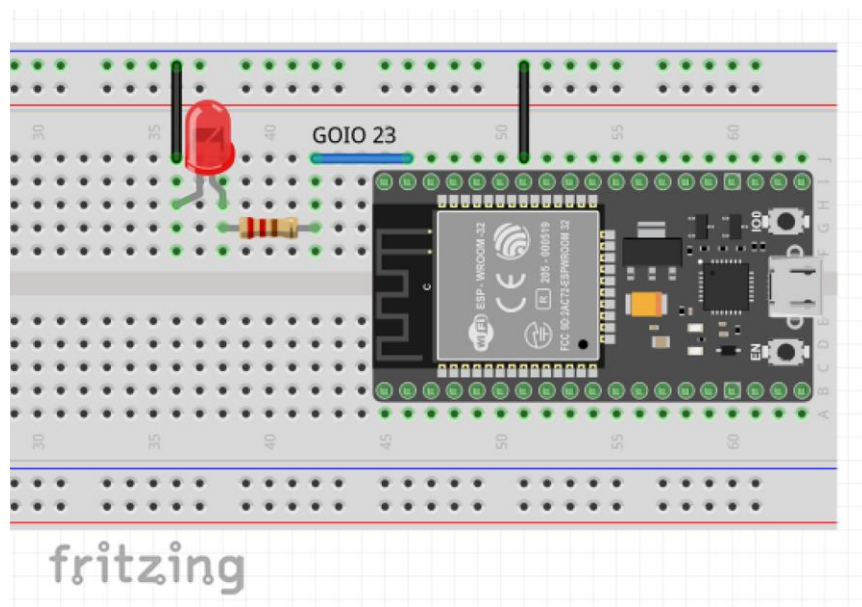
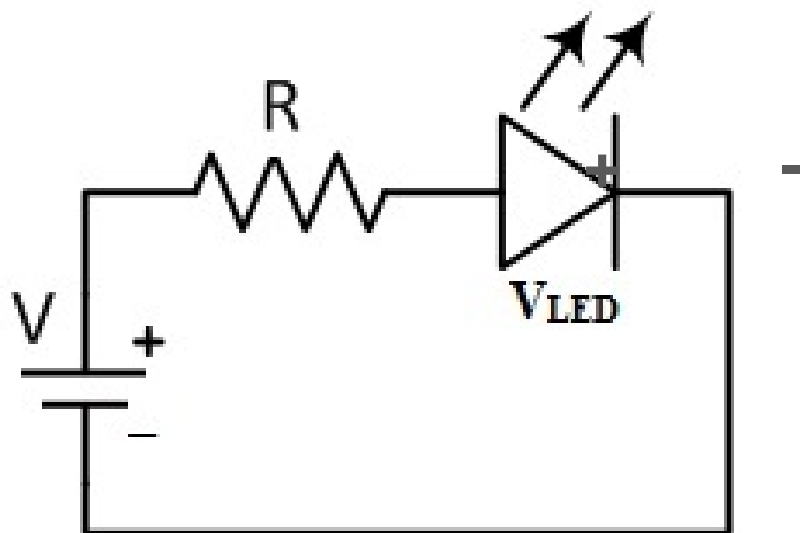
實作時間

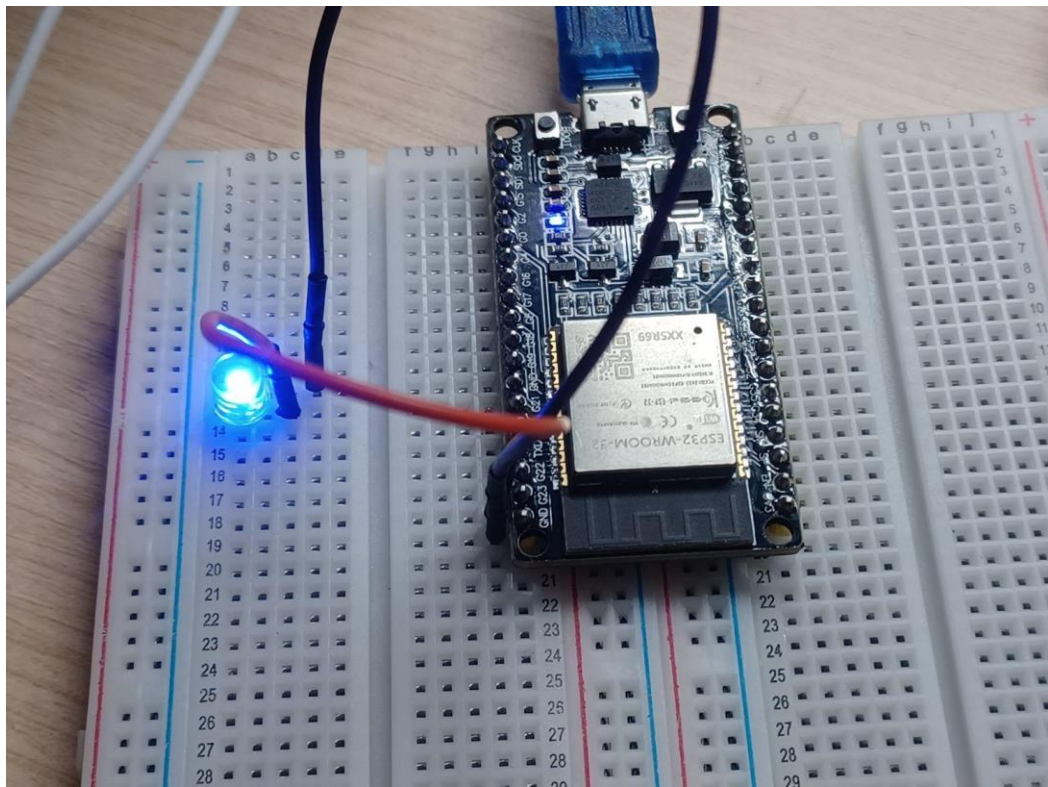


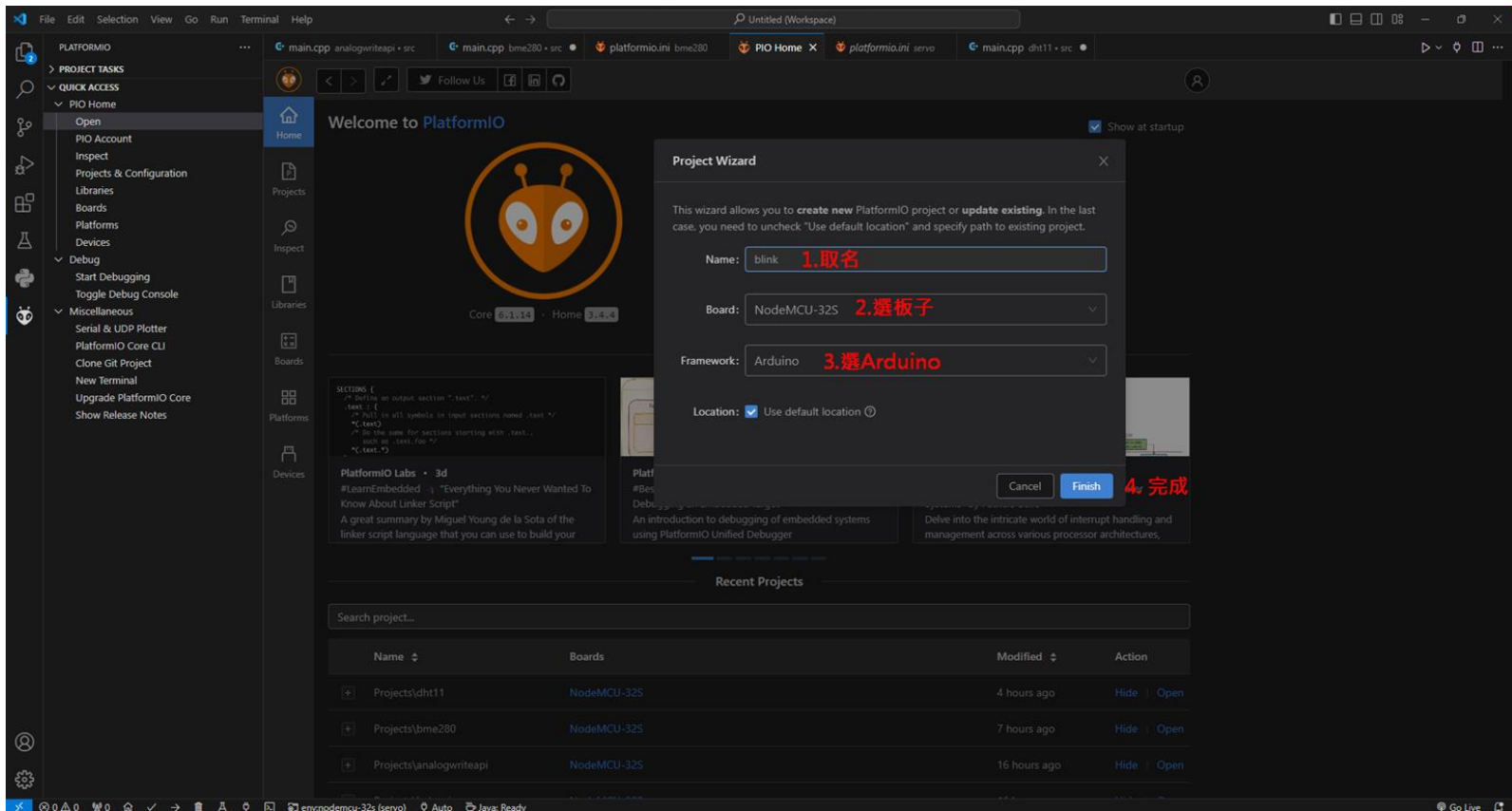
複習麵包板構造



接線圖







板子上LED閃爍

```
#include<Arduino.h>
/*ESP32開發板測試blink*/

#define LED 23

//閃爍之led預設在gpio23上

void setup() {

    Serial.begin(9600);

    pinMode(LED, OUTPUT);

}
```

```
void loop() {

    digitalWrite(LED, HIGH); //對應針腳輸入高電位

    Serial.println("LED is ON"); //序列附輸入訊息

    delay(1000); //等一秒

    digitalWrite(LED, LOW); //對應針腳輸入低電位

    Serial.println("LED is LOW"); //序列附輸入訊息

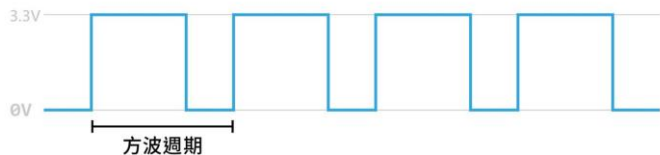
    delay(1000); //等一秒

}
```

PWM脈衝寬度調變：用數位訊號模擬類比訊號

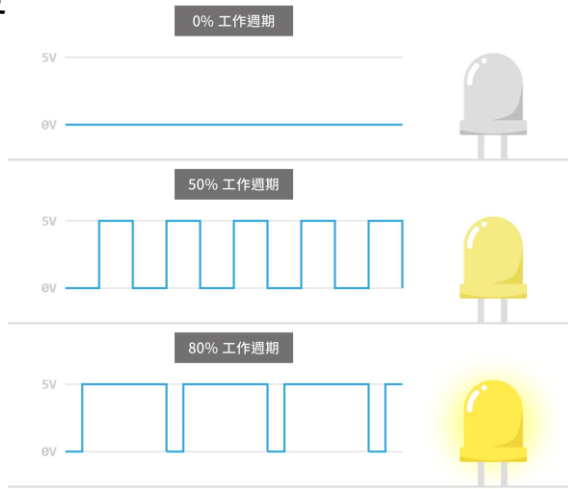
數位訊號：

- 只能輸出高電位 (HIGH) 和低電位 (LOW) 兩種狀態
- 當高、低電位在固定時間周期內有規律的變化時，會產生波形，稱為脈波



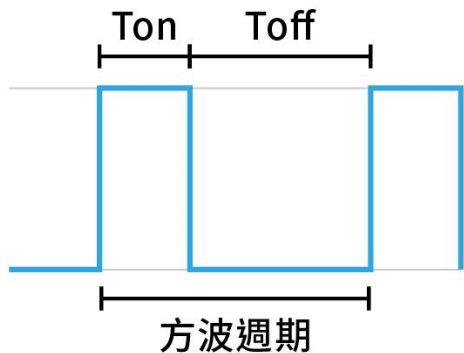
PWM 的原理：

在相同時間周期中，調整高電位在脈波中的時間佔比（提高通電時間的比例），就能調整數位輸出的功率，因此稱做脈波寬度調變。



如何計算 PWM 的工作週期

工作周期是高態時間在整個周期中的所佔的比例



example:

$$T_{on} / \text{方波週期} = 0.3$$

$$\text{工作週期} = 30\%$$

DWMA 給出平均電壓

0% 工作週期



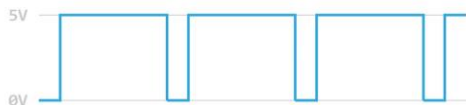
工作周期為 0% 時，輸出的電壓就會是 0V，等同於數位輸出的 LOW。

50% 工作週期



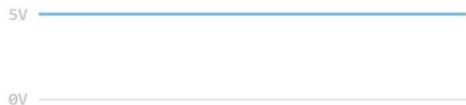
工作周期 50% 的脈波訊號，每個周期的平均電壓就是 1.65V。

80% 工作週期



工作周期是 80%，每個周期平均電壓就會是 2.64V。

100% 工作週期



工作周期為 100% 時，輸出的電壓就會是 3.3V，等同於數位輸出的 HIGH。

每毫秒→

結論

- 相同週期中，改變高態和低態的占比就能控制LED發出不同的亮度。
- 工作週期越高，平均輸出電壓就越大，相對的平均輸出功率也會越大（亮）。
- 可以想成一種人眼錯覺

實作時間

LEDC API

設定三個參數: 通道、頻率以及解析度，接著指定要使用的接腳，才輸出 PWM 訊號。

```
#include<Arduino.h>

// 定義LED參數
int led_pin = 23;      // GPIO23
int brightness = 0;    // 亮度
int fade_amount = 2;   // 變化速度
// 定義PWM參數
int freq = 5000;
int led_channel = 0;
int resolution = 8;
void setup() {
    // 設定PWM通道、頻率、解析度
    ledcSetup(led_channel, freq, resolution);
    // 連接通道與GPIO接腳
    ledcAttachPin(led_pin, led_channel);
}
```

```
void loop() {
    ledcWrite(led_channel, brightness);
    brightness += fade_amount;

    if(brightness <= 0 || brightness >= 255){
        // 若亮度範圍超出0~255，就反轉fade_amount
        fade_amount = -fade_amount;
    }
    delay(30);
}
```

函式

- **ledcSetup**(通道 , 參數頻率 , 解析度)

通道:1~16

pwm 頻率 : 5kHz

解析度 : 越高變化量越精細 (位元數為2的解析度次方) (8:0~255)

- **ledcAttachPin**(接腳 , 通道)

接腳 : 連接的GPIO編號

通道 : 配合**setup**設定的

- **ledcWrite**(通道 , 工作週期)

通道 : 配合**setup**設定的

工作週期 : 配合解析度 , 控制亮度(0~255)

額外練習：在Serial Monitor 中印出 brightness

ans:

```
#include<Arduino.h>

int led_pin = 23;      // GPIO12
int brightness = 0;    // 亮度
int fade_amount = 2;   // 變化速度
int freq = 5000;
int led_channel = 0;
int resolution = 8;

void setup() {
    Serial.begin(9600);
    // 設定PWM通道、頻率、解析度
    ledcSetup(led_channel, freq,
resolution);
    // 連接通道與GPIO接腳
    ledcAttachPin(led_pin, led_channel);
}
```

```
void loop() {
    ledcWrite(led_channel, brightness);
    brightness += fade_amount;
    Serial.print("亮度: ");
    Serial.println(brightness)

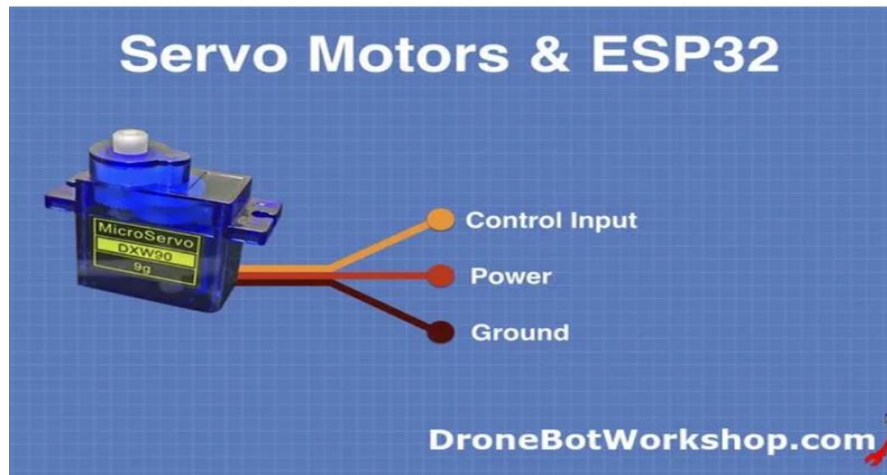
    if(brightness <= 0 || brightness >= 255){
        // 若亮度範圍超出0~255，就反轉fade_amount
        fade_amount = -fade_amount;
    }
    delay(30);
}
```


2. 伺服馬達 servo Motor

一般型式的可以透過PWM精準的控制其旋轉180度，耗電量高，一般筆電供電僅支持一顆

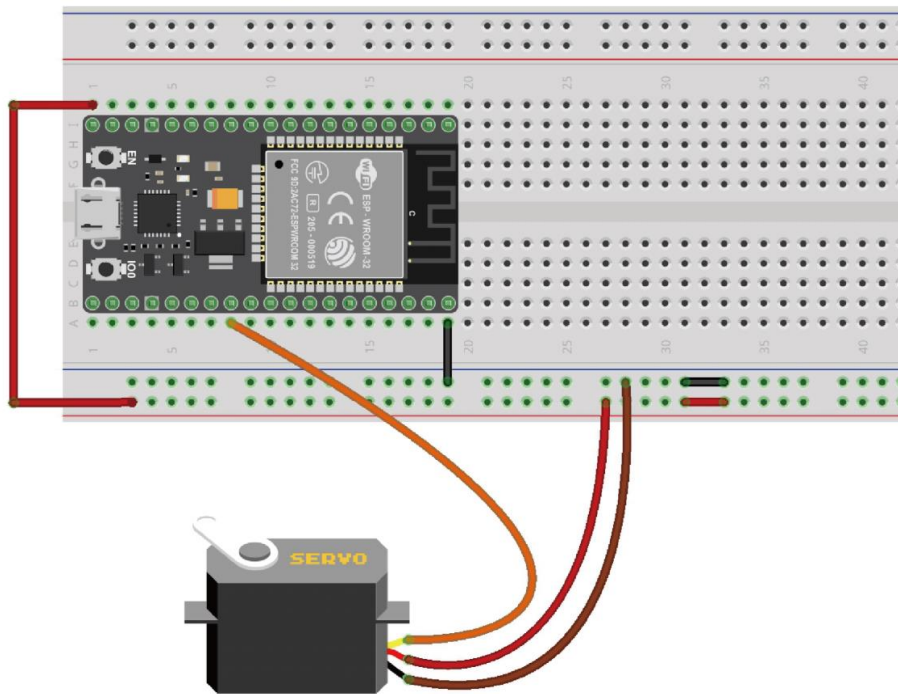
一般伺服器馬達有三條線

1. 電源(紅色) -VCC
2. 負極(棕色) -GND
3. 訊號線(橘色) GPIO



實作時間

接線圖



橘:GPIO13

基本語法 - 函式庫 (Library)

將函式庫比喻為筆盒：

當需要寫字的時候可以去筆盒裡面尋找 “筆” ，得到筆並且直接使用，不需要從頭去製造一支筆。



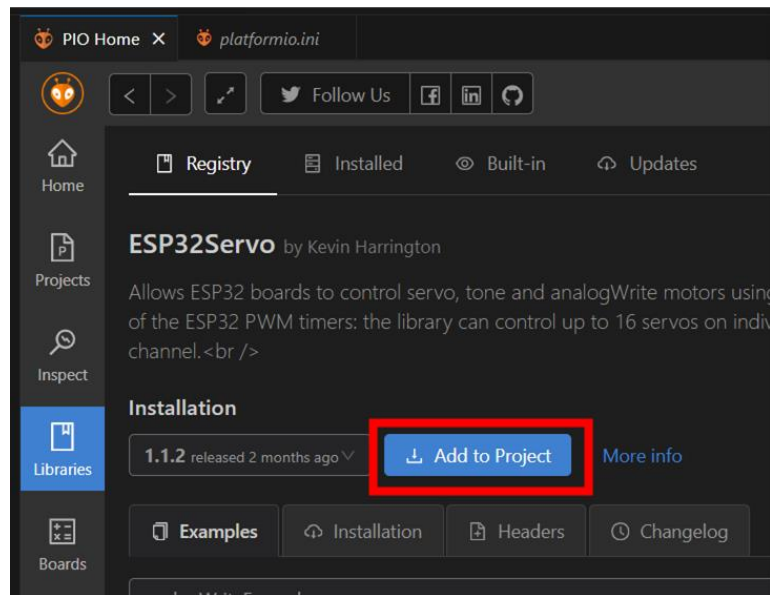
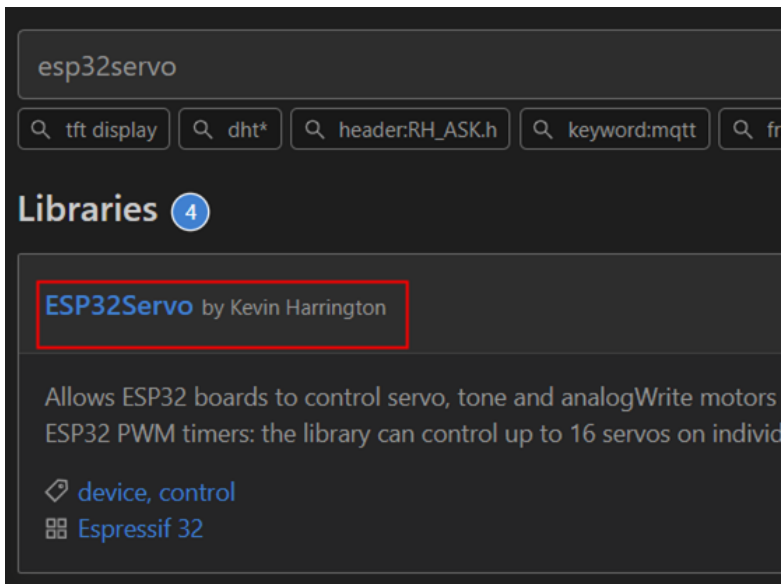
函式庫=>筆盒 (擁有各種工具/文具)

- 函數的集成體

函數=>工具/文具

- 實現各種功能

下載函式庫



範例程式碼-1

```
#include<Arduino.h>

#include <ESP32Servo.h>

int n =0;

Servo myservo;
int servoPin=13;

int pos = 0;    // 定義舵機轉動位置

void setup() {
    myservo.attach(servoPin);

    // 設置舵機控制腳位
    Serial.begin(9600);

}
```

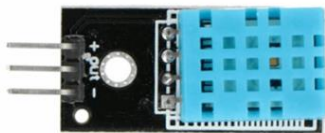
```
void loop() {
    n+=1;
    myservo.write(30); //旋轉至30度
    delay(1000);
    myservo.write(120); //旋轉至120度
    delay(1000);
    Serial.println(n);
}
```

範例程式碼-2

```
#include<Servo.h>
Servo myservo;
int servoPin=16;
int pos = 0;    // 定義舵機轉動位置
void setup() {
    myservo.attach(servoPin);
    // 設置舵機控制腳位
}
```

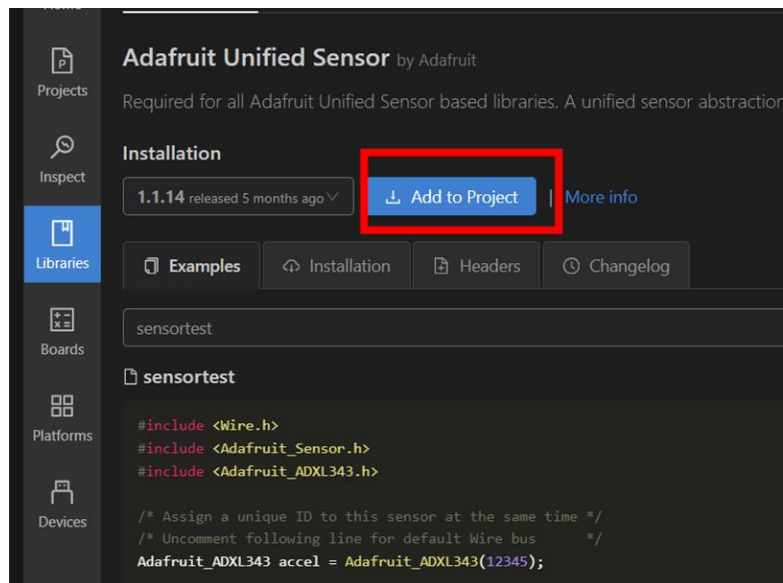
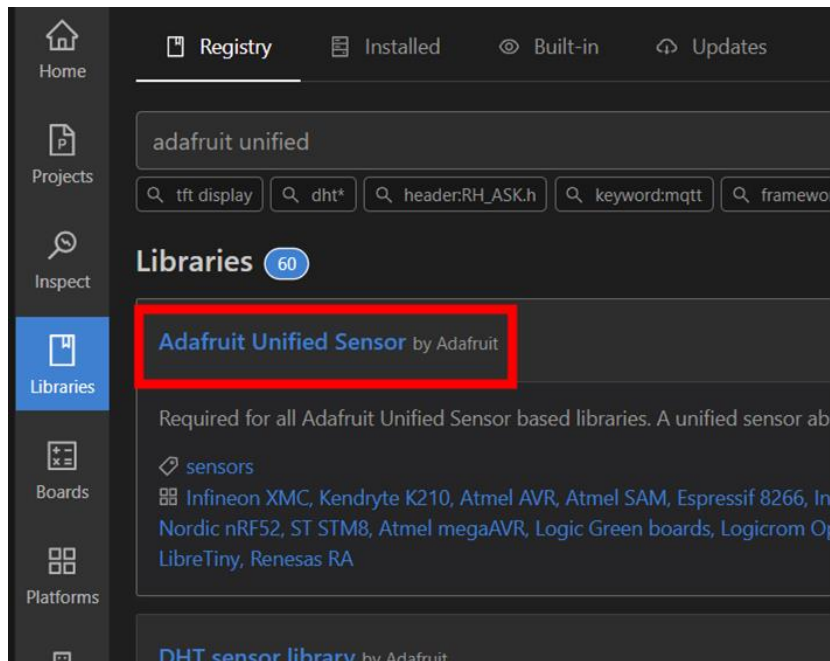
```
void loop() {
    // 0到180旋轉舵機，每次延時15毫秒
    for(pos = 0; pos < 180; pos += 1) {
        myservo.write(pos);
        delay(15);
    }
    // 180到0旋轉舵機，每次延時15毫秒
    for(pos = 180; pos>=1; pos-=1){
        myservo.write(pos);
        delay(15);
    }
}
```

3.DHT11 結合濕度計和測溫元件，量測週遭空氣環境



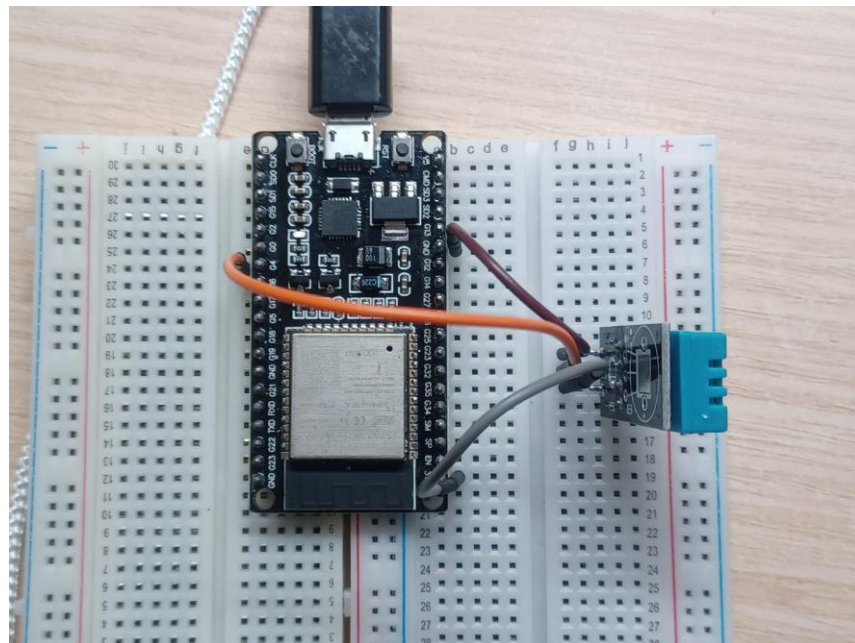
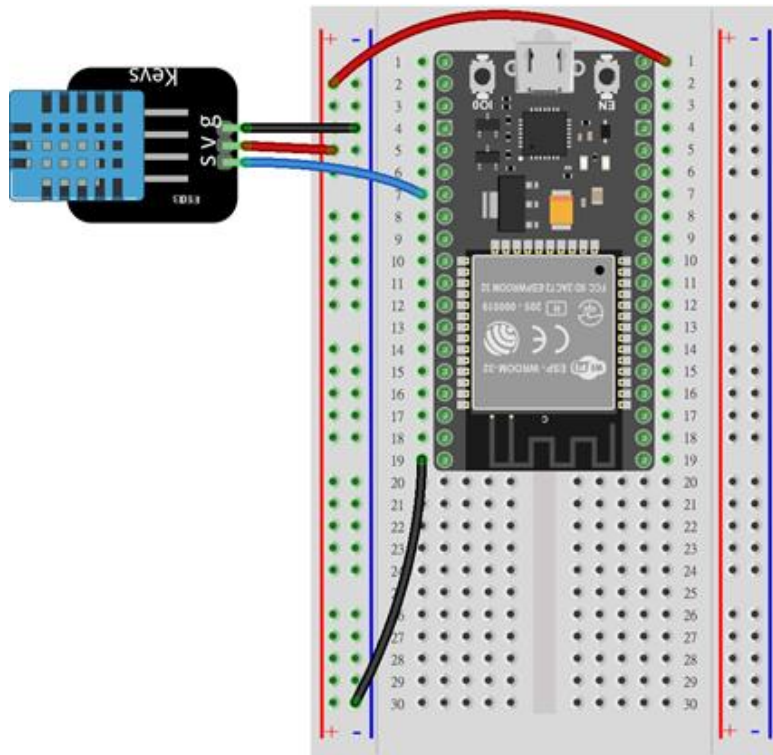
ESP32 Pins	DHT11模組
5v	+
GPIO 4	Out
GND	-

下載函式庫



實作時間

接線圖



範例程式碼

```
#include <Arduino.h>
#include "DHT.h"

#define DHTPIN 4    // DHT sensor pin

#define DHTTYPE DHT11    // DHT 11

DHT dht(DHTPIN, DHTTYPE);

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

```
void loop() {
    delay(2000); //等待兩秒後開始測量
    float h = dht.readHumidity(); // 讀取濕度
    float t = dht.readTemperature(); // 讀取溫度

    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print("%   Temperature: ");
    Serial.print(t);
    Serial.print("*C ");
    Serial.println();
}
```

DHT11 物件

- `dht.readHumidity()` – 讀取絕對濕度
- `dht.readTemperature()` – 讀取攝氏溫度
- `dht.readTemperature(true)` – 讀取華氏溫度

4.BME280 可讀取氣壓、溫度和濕度

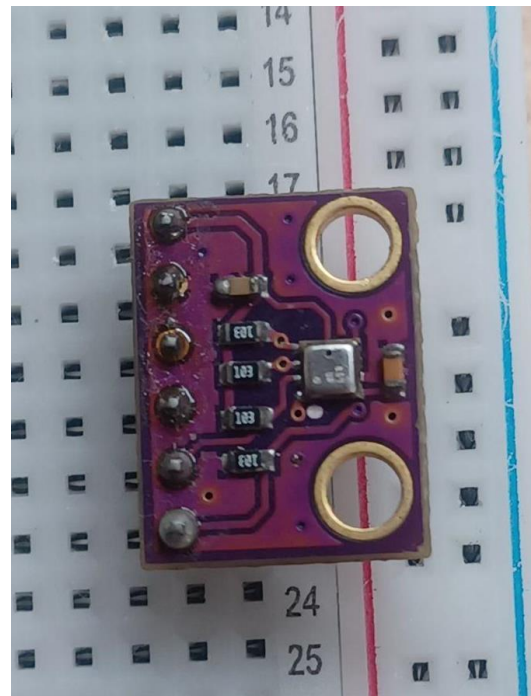


—電源輸入

—接地

—clock

—data





Home



Projects



Inspect



Libraries



Boards



Platforms

Registry

Installed

Built-in

Updates

BMP085



tft display

dht*

header:RH_ASK.h

keyword:mqtt


framework:mbd

platform:espressif8266


more...


Libraries 22


Adafruit BMP085 Library by Adafruit


14,297 1  Arduino

A powerful but easy to use BMP085/BMP180 Library

 sensors

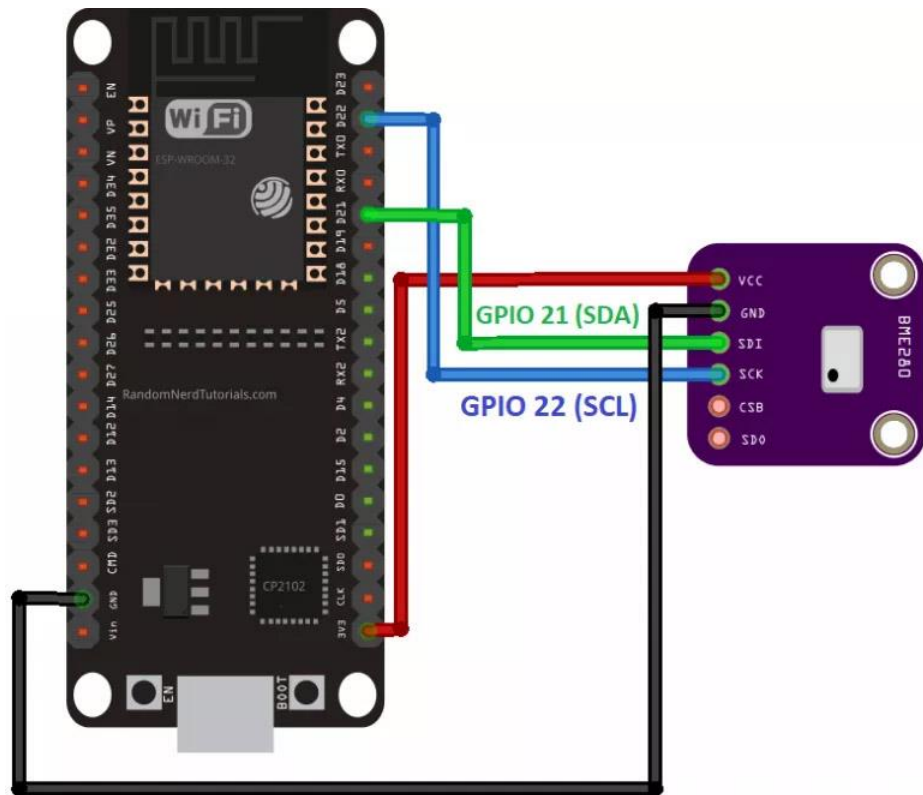
 Infineon XMC, Kendryte K210, Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip PIC32, Nordic nRF51, ST STM32, Teensy, TI MSP430, TI TIVA, Espressif 32, Nordic nRF52, ST STM8, Atmel megaAVR, Logic Green boards, Logicrom OpenCPU Development Platform, Raspberry Pi RP2040, K1921VK, TI MSP432, Heltec CubeCell, LibreTiny, Renesas RA

 sensors

 Infineon XMC, Kendryte K210, Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC32, Microchip PIC32, Nordic nRF51, ST STM32, Teensy, TI MSP430, TI TIVA, Espressif 32, Nordic nRF52, ST STM8, Atmel megaAVR, Logic Green boards, Logicrom OpenCPU Development Platform, Raspberry Pi RP2040, K1921VK, TI MSP432, Heltec CubeCell, LibreTiny, Renesas RA

實作時間

接線圖



範例程式碼

```
#include <Wire.h>
#include <Adafruit_BME280.h>

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // // 此範例使用I2C通信協議，只需要建立一個Adafruit_BME280 對象 bme

unsigned long delayTime;
```

```
void setup() {
    Serial.begin(9600);
    while(!Serial); //get serial running
    Serial.println(F("BME280 test"));

    unsigned status;

    status = bme.begin(0x76);
    while (1) delay(10);

    Serial.println("-- Default Test --");
    delayTime = 1000;
    Serial.println();
}
```

```
void printValues() {  
    Serial.print("Temperature = "); //列印溫度  
    Serial.print(bme.readTemperature());  
    Serial.println(" *C");  
  
    Serial.print("Pressure = "); //列印濕度  
    Serial.print(bme.readPressure() / 100.0F);  
    Serial.println(" hPa");  
  
    Serial.print("Approx. Altitude = "); //列印海拔  
    Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));  
    Serial.println(" m");  
  
    Serial.print("Humidity = "); //列印濕度  
    Serial.print(bme.readHumidity());  
    Serial.println(" %");  
  
    Serial.println();  
}
```

bme 物件

- `bme.readTemperature ()` – 讀取攝氏溫度
- `bme.readHumidity ()` – 讀取絕對濕度
- `bme.readPressure ()` – 讀取壓力，單位為 hPa (百帕斯卡 = 毫巴)
- `bme.readAltitude (SEALEVELPRESSURE_HPA)` – 根據海平面的壓力估算海拔高度