

# Arduino Week02

# Arduino

1. 基本程式介紹

2. 光敏電阻

3. 可變電阻/蜂鳴器

# Arduino 基本程式介紹

## Arduino IDE介紹

- IDE "Integrated Development Environment"(整合開發環境)
- 一個可以開發、編碼、測試和除錯軟體的工具

# 程式結構

```
int ledPin =13;

void setup() {
    pinMode(ledPin, OUTPUT);
}

void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

- `setup()` 函數只在程序開始時運行一次。  
通常用於初始化變數、腳位模式等。
- `loop()` 函數在 `setup()` 之後無限次數地運行。  
大多數Arduino程序的主要部分。

宣告一個ledPin的變數 = 13



將ledPin腳位設定成OUTPUT模式



設定ledPin腳位為高電位 = 5v, 燈亮  
設定延遲時間1000ms = 1秒  
設定ledPin腳位為低電位 = 0v, 燈滅  
設定延遲時間1000ms = 1秒

## 註解//

只需打兩條斜線，後方的文字就會被認定為註解，並且會顯示成灰色

註解可以幫助其他人去理解你的程式碼。

```
1
2 // void setup() {
3   pinMode(LED_BUILTIN, OUTPUT); //設定led腳位
4 }
5
6 // void loop() {
7   digitalWrite(LED_BUILTIN, HIGH); //點亮led
8   delay(1000); //延遲1秒
9   digitalWrite(LED_BUILTIN, LOW); //關閉led
10  delay(1000); //延遲1秒
11 }
```

## ► Arduino 基本程式介紹-1

### 區塊註解/\*\*/

從/\*開始到\*/這之間，包在裡面的都是註解，在裡面也可以換行。

```
1  /*
2   led閃爍範例程式
3   使用Arduino開發板上面的led .
4   無須額外接任何led線路
5  */
6
7  void setup() {
8      pinMode(LED_BUILTIN, OUTPUT); //設定led腳位
9  }
10
11 void loop() {
12     digitalWrite(LED_BUILTIN, HIGH); //點亮led
13     delay(1000); //延遲1秒
14     digitalWrite(LED_BUILTIN, LOW); //關閉led
15     delay(1000); //延遲1秒
```

## 函數 Function 的基本概念

- 什麼是函數？為什麼要使用它們？

函數允許我們將常用的程式碼片段包裝起來，以便多次使用。它可以使程式碼更整潔、易於閱讀和維護。

- 內建函數 `digitalRead()`, `analogWrite()`, `delay()` [補充資料](#)
- Arduino的函式庫，例如 `Servo`, `Wire`等
- 自定義函數

## 接腳模式設定

- 「輸入模式」送入一些訊號讓 Arduino 來讀取
- 「輸出模式」Arduino 要從某隻接腳送出電流，讓外接的電路工作時，則必須被設定為「輸出模式」

工作接腳編號 0-13 與 A0-A5



`pinMode(工作接腳, 模式);`

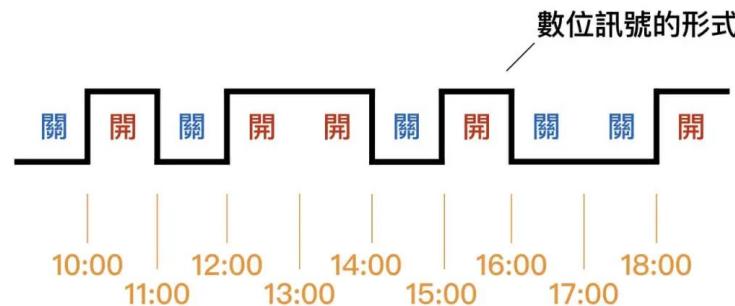


INPUT 或 OUTPUT



## 數位訊號概念

- 數位訊號就是「只有兩種狀態」的訊號，比如說電燈的開關



## 電位狀態設定

- 每個工作接腳都可以是「高電位」或是「低電位」的狀態 其他

工作接腳編號 0-13 與 A0-A5



`digitalWrite(工作接腳, 狀態);`



HIGH 或 LOW



## 執行暫停指令

- 以「毫秒」作為單位

毫秒(ms)  
↓  
`delay(時間長度);`



## - **Serial.begin() /Serial.print()/Serial.println()**

- 「Baud rate 鮑率」

是一種資料傳輸的速率單位。簡單來說，設定 115200，就代表 Arduino 與電腦之間，每一秒鐘能傳輸 11 萬 5 千 2 百個位元的資料

Serial.begin(115200);  
                  ^  
                  大寫  
                  |  
                  鮑率 (Baud rate)  
                  115200 bit/s



## 變數和資料類型

- 變數可以看作是存儲資料的容器。

```
int led = 13;
```

變數類型    變數名稱 ← 數值  
    賦值



```
int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

## 變數和資料類型

類型	中文名稱	占用記憶體大小	數值範圍
boolean	布林	8 bit	true或false(1或0)
byte	位元組	8 bit	0~255
char	字元	8 bit	-128~127
int	整數	16 bit (8位元微控制器) 32 bit (32位元微控制器 )	-32768~32767 -2147483648~2147483647
long	長整數	32 bit	-2147483648~2147483647
float	浮點數	32 bit	+/-3.4E+38
double	雙倍精確度浮點數	32 bit 64 bit (32位元微控制器 )	+/-3.4E+38 +/-1.7E+308

# 算術運算子

- 常見的運算子就是在數值之間進行加減乘除的符號

運算子	行為	範本	說明
+	加法	$x+y$	兩者相加
-	減法	$x-y$	兩者相減
*	乘法	$x*y$	兩者相乘
/	除法	$x/y$	兩者相除
%	整數取餘數	$x \% y$	$x$ 除以 $y$ 的餘數
++	遞增	$x++$	相當於 $x=x+1$
--	遞減	$x--$	相當於 $x=x-1$

## 關係運算子

- 常見的運算子就是在數值之間進行加減乘除的符號

運算子	行為	範本	說明
<code>==</code>	等於	<code>x==y</code>	若 <code>x</code> 與 <code>y</code> 相同，則為 <code>true</code> ，否則是 <code>false</code>
<code>!=</code>	不等於	<code>x!=y</code>	若 <code>x</code> 與 <code>y</code> 不相同，則為 <code>true</code> ，否則是 <code>false</code>
<code>&gt;</code>	大於	<code>x&gt;y</code>	若 <code>x</code> 大於 <code>y</code> ，則為 <code>true</code> ，否則是 <code>false</code>
<code>&lt;</code>	小於	<code>x&lt;y</code>	若 <code>x</code> 小於 <code>y</code> ，則為 <code>true</code> ，否則是 <code>false</code>
<code>&gt;=</code>	大於等於	<code>x&gt;=y</code>	若 <code>x</code> 大於等於 <code>y</code> ，則為 <code>true</code> ，否則是 <code>false</code>
<code>&lt;=</code>	小於小於	<code>x&lt;=y</code>	若 <code>x</code> 小於等於 <code>y</code> ，則為 <code>true</code> ，否則是 <code>false</code>

## 邏輯判斷與控制流程 1 if - else

```
if(條件一){  
    條件一成立時執行的工作  
}  
else if(條件二){  
    條件二成立時執行的工作  
}  
else if(條件三){  
    條件三成立時執行的工作  
}  
:  
else{  
    條件都不成立時執行的工作  
}
```



## 邏輯判斷與控制流程 1 if - else

兩個等號！？ 比較運算子

```
if(digitalRead(7) == HIGH){  
    digitalWrite(13, HIGH);  
}  
  
else{  
    digitalWrite(13, LOW);  
}
```

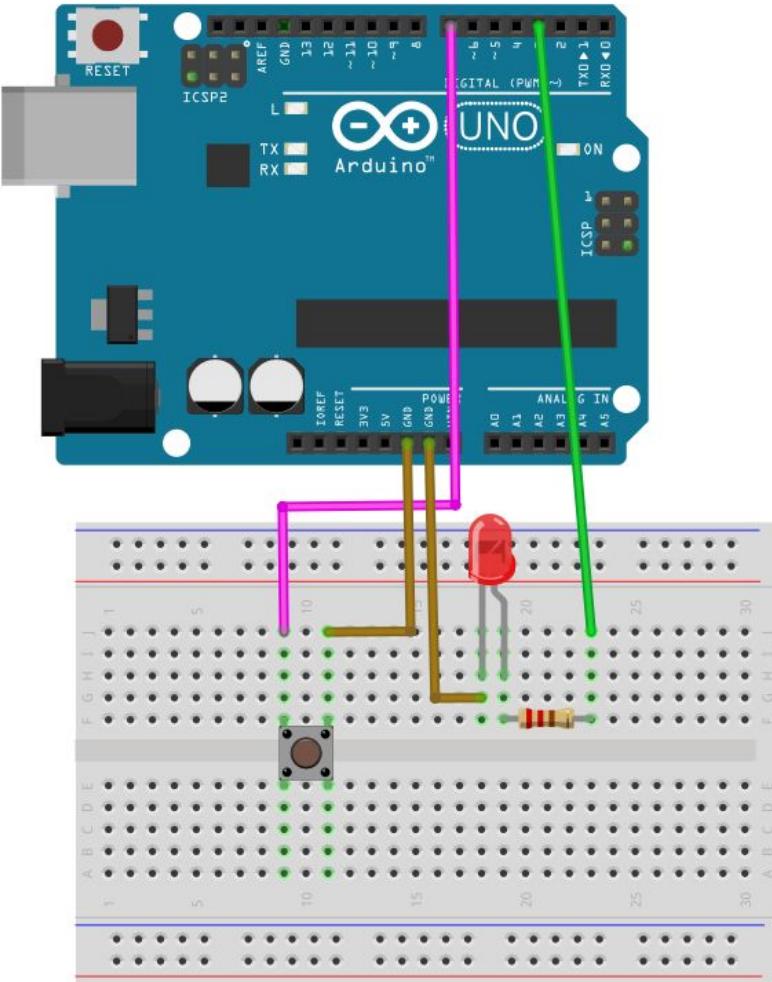


## 邏輯判斷與控制流程 switch / for / while

Arduino練習

**/ BUTTON + LED /**

```
// constants won't change. They're used here to set pin numbers:  
const int buttonPin = 2;      // the number of the pushbutton pin  
const int ledPin = 13;        // the number of the LED pin  
  
// variables will change:  
int buttonState = 0;          // variable for reading the pushbutton status  
  
void setup() {  
    // initialize the LED pin as an output:  
    pinMode(ledPin, OUTPUT);  
    // initialize the pushbutton pin as an input:  
    pinMode(buttonPin, INPUT);  
}  
  
void loop() {  
    // read the state of the pushbutton value:  
    buttonState = digitalRead(buttonPin);  
  
    // check if the pushbutton is pressed. If it is, the buttonState is HIGH:  
    if (buttonState == HIGH) {  
        // turn LED on:  
        digitalWrite(ledPin, HIGH);  
    } else {  
        // turn LED off:  
        digitalWrite(ledPin, LOW);  
    }  
}
```



光敏電阻

► 光敏電阻

## 光敏電阻

把感測到環境的亮度數值以 **類比數值**

回傳給 Arduino, 沒有正負極之分,

當光源比較強時, 電阻值較低。



► 光敏電阻

## 數位訊號

```
digitalRead( 腳位 );
```

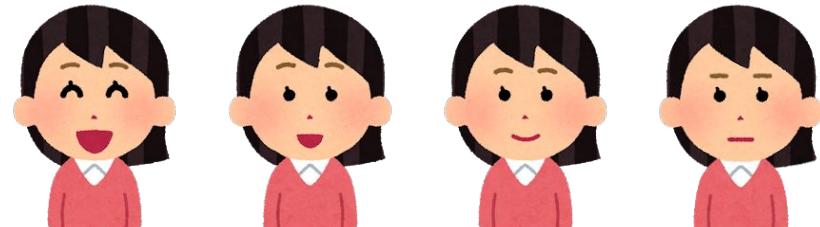
```
digitalWrite( 腳位, 要寫入的數  
值 ); //數值為 0 或 1
```



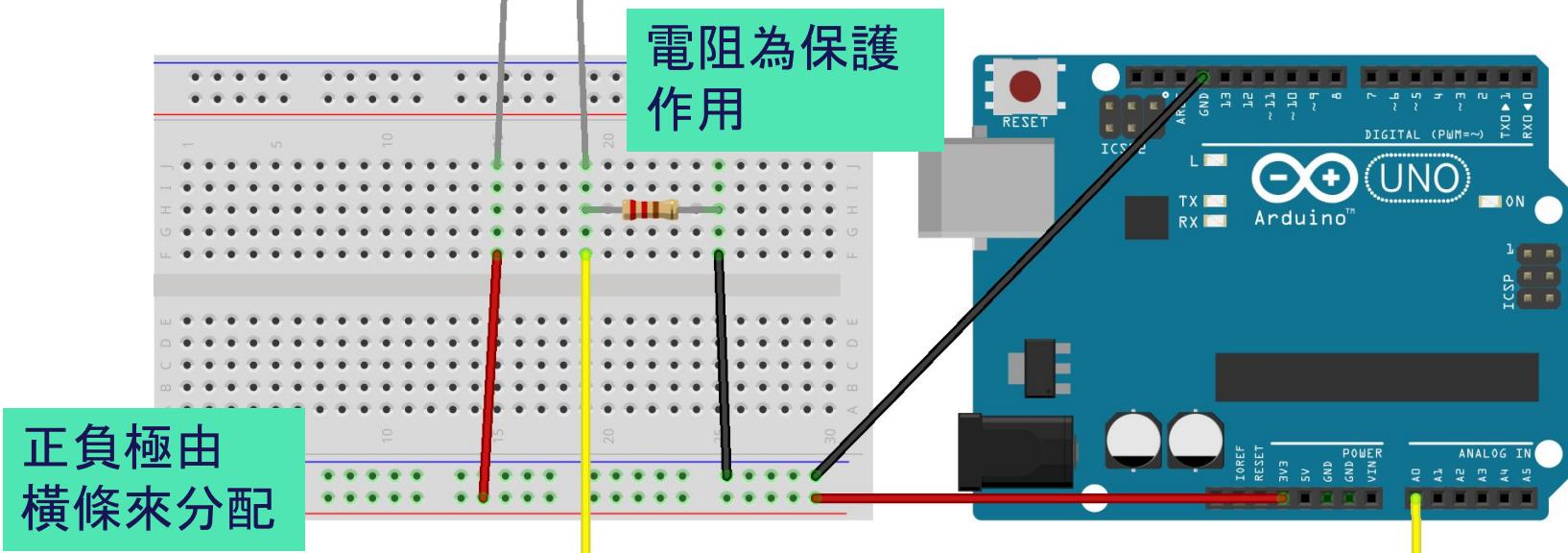
## 類比訊號

```
analogRead( 腳位 );
```

```
analogWrite( 腳位, 要寫入的數  
值 ); //數值為 0 ~ 255
```



## ► 光敏電阻



The screenshot shows the Arduino IDE interface with two main windows. The left window is titled 'sketch\_jun13a | Arduino 1.8.3' and displays the following code:

```
void setup(){
  //設定 serial port
  Serial.begin(9600);
  //設定 A0 為輸入腳位
  pinMode(0, INPUT);
}
void loop(){
  //直接將讀取到的類比數值
  //從 serial 監控視窗印出
  Serial.println( analogRead(0) );
  delay(50);
}
```

The right window is titled '/dev/cu.usbmodem14111 (Arduino Leonardo)' and shows the serial monitor output. A green box highlights the text '開啟序列埠監控視窗' (Open Serial Monitor Window). The output window shows the following data:

Time	Value
500	366
366	366
366	366
366	366
366	366
365	365
365	365
365	365
364	364
364	364
364	364
364	364

The bottom of the right window includes settings for '自動捲動' (Auto scroll), '沒有行結尾' (No line ending), a baud rate selector set to '9600 baud' (highlighted with a red box), and a 'Clear output' button.

## 小挑戰

當天黑就要自動開燈，請用剛剛所學達成這件事情。

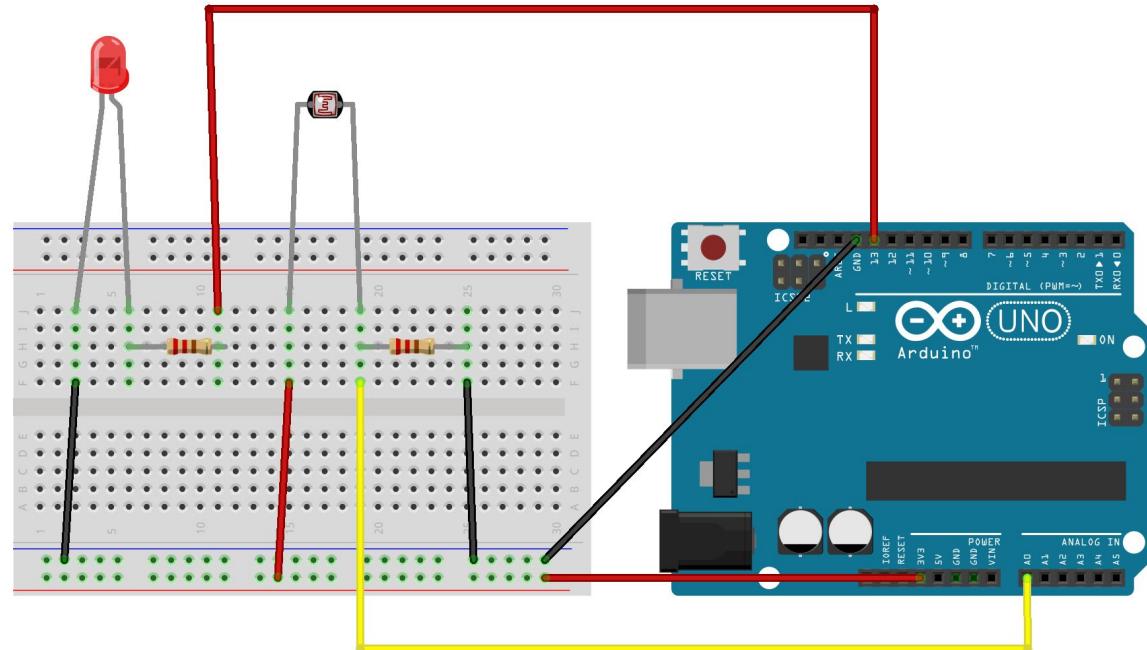


# 解答篇

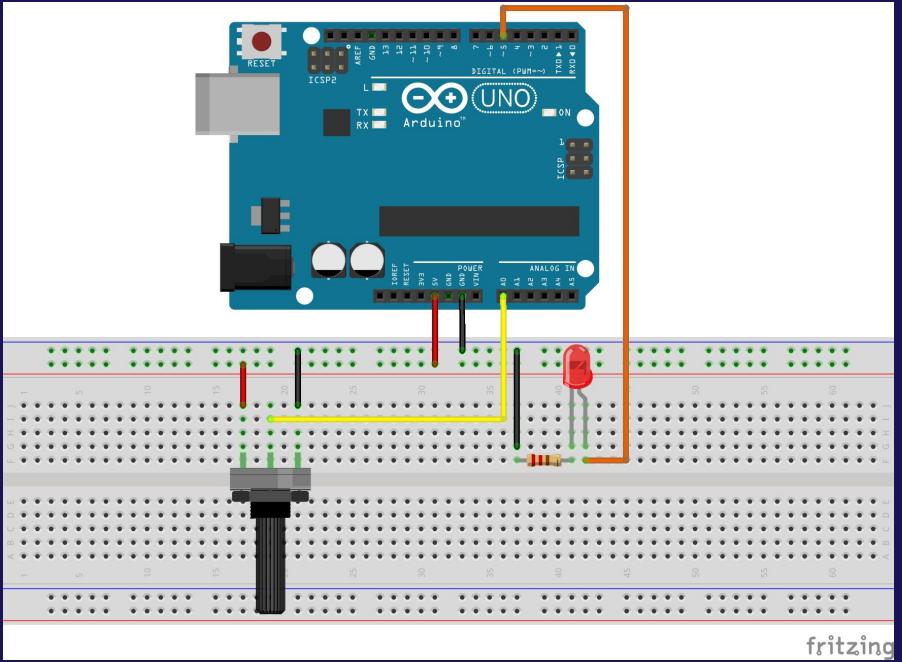
```
void setup(){
  pinMode(13, OUTPUT);
  pinMode(0, INPUT);
}

void loop(){
  //先用一個變數暫存讀取到的數字
  int lightData = analogRead(0);

  //根據讀取到的數值做判斷
  if(lightData < 200){
    //如果數值小於200就開燈
    digitalWrite(13, HIGH);
  }else{
    //否則就關燈
    digitalWrite(13, LOW);
  }
  delay(50);
}
```



# LED + 可變電阻



## 可變電阻控制LED

LED的亮度隨著可變電阻轉動變化

提示：透過map()做數位類比的轉換  
map(value, fromLow, fromHigh, toLow, toHigh)

<https://hackmd.io/@yizhewang/SkGyJJv4N>

► 蜂鳴器 Buzzer

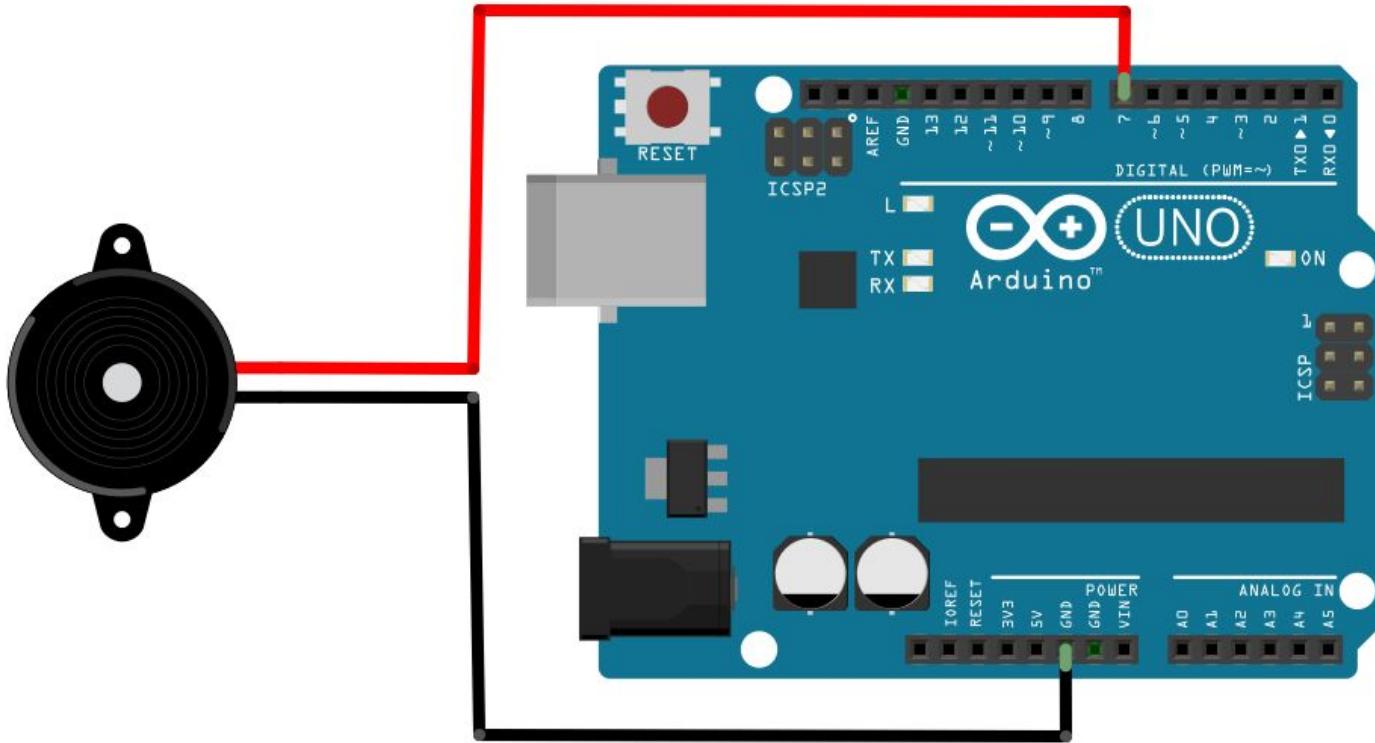
## 蜂鳴器 Buzzer

有源蜂鳴器 固定頻率(通常有白色貼紙)

無源蜂鳴器



```
1 void setup() {  
2     pinMode(7,OUTPUT);  
3 }  
4  
5 void loop() {  
6     digitalWrite(7,HIGH);  
7     delay(1000);  
8     digitalWrite(7,LOW);  
9     delay(2000);  
10 }
```



檔案 編輯 草稿碼 工具 說明

新增 Ctrl+N	
開啟... Ctrl+O	
開啟最近 >	
草稿碼簿 >	
範例 >	<ul style="list-style-type: none"><li>△ 內建範例</li><li>01.Basics &gt;</li><li>02.Digital &gt; <ul style="list-style-type: none"><li>BlinkWithoutDelay</li><li>Button</li><li>Debounce</li><li>DigitalInputPullup</li><li>StateChangeDetection</li><li>toneKeyboard</li><li>toneMelody</li><li>toneMultiple</li><li>tonePitchFollower</li></ul></li><li>03.Analog &gt;</li><li>04.Communication &gt;</li><li>05.Control &gt;</li><li>06.Sensors &gt;</li><li>07.Display &gt;</li><li>08.Strings &gt;</li><li>09.USB &gt;</li><li>10.StarterKit_BasicKit &gt;</li><li>11.ArduinolISP &gt;</li></ul>
關閉 Ctrl+W	
儲存 Ctrl+S	
另存新檔... Ctrl+Shift+S	
頁面設定 Ctrl+Shift+P	
列印 Ctrl+P	
偏好設定 Ctrl+Comma	
離開 Ctrl+Q	

24

25 // note duration

26 int noteDuration

OTE\_B3, NOTE\_C4  
等，etc.:

```
1 #include "pitches.h" //載入音調和頻率的對照檔
2
3 // 旋律
4 int melody[] = {
5     NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, N
6 };
7
8 // 每個音的拍子，4：4分音符，8：8分音符
9 int noteDurations[] = {
10    4, 8, 8, 4, 4, 4, 4, 4
11 };
12
13 void setup() {
14
15     for (int thisNote = 0; thisNote < 8; thisNote++) {
16         // 計算每個音的長度，4分音符：1000 / 4，8分音符：1000/8
17         int noteDuration = 1000 / noteDurations[thisNote];
18         tone(7, melody[thisNote], noteDuration); //tone(PIN腳,
19
20         // 每個音之間要停一小段時間，範例是建議拍子的長度加30%
21         int pauseBetweenNotes = noteDuration * 1.30;
22         delay(pauseBetweenNotes);
23
24         noTone(7); // 停止播放
25     }
26 }
27
28 void loop() {
29     // no need to repeat the melody.
30 }
```

# Q&A

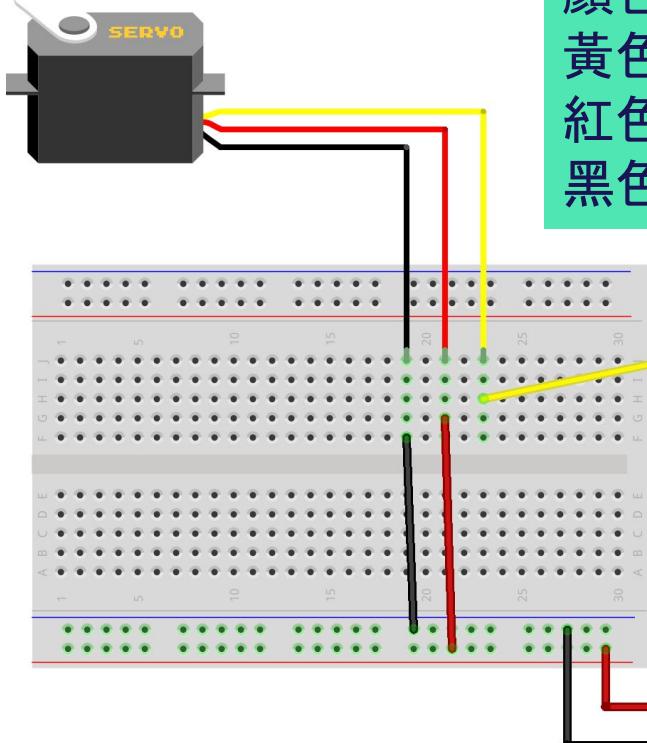
**伺服馬達給我動起來 !!**

# 伺服馬達

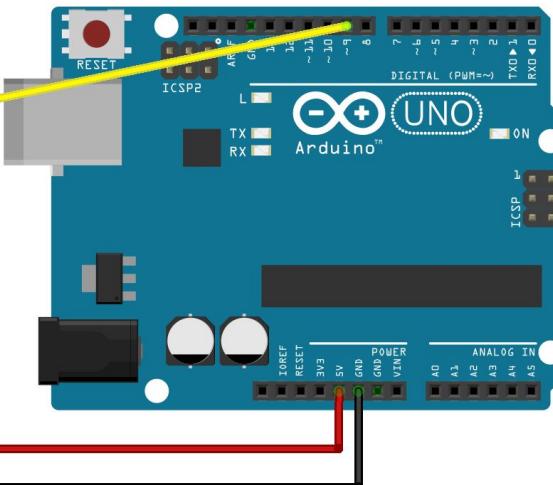
會旋轉的東西

- 可以指定旋轉到幾度
- 範圍只有 0 ~ 180 度
- 紅色接正極
- 黃色接負極
- 咖啡色接訊號





顏色對照  
黃色：咖啡色  
紅色：紅色  
黑色：黃色



fritzing

# 轉到 90 度

```
#include <Servo.h>
Servo myservo;

void setup(){
    myservo.attach(9); // 連接數位腳位9，伺服馬達的訊號線
}

void loop() {
    myservo.write(90); // 使用write，傳入角度
}
```

引入第三方函式



# 來回旋轉



**0, 1, 2, 3, 4, ……, 178, 179, 180**

**180, 179, 178, ……, 3, 2, 1, 0**

⋮

# 來回旋轉

```
#include <Servo.h>
Servo myservo;

void setup(){
    myservo.attach(9); // 連接數位腳位9，伺服馬達的訊號線
}

void loop() {
    for(int i = 0; i <= 180; i+=1){
        myservo.write(i); // 使用write，傳入角度，從0度轉到180度
        delay(20);
    }
    for(int i = 180; i >= 0; i-=1){
        myservo.write(i); // 使用write，傳入角度，從180度轉到0度
        delay(20);
    }
}
```

For迴圈

for( 開始條件; 結束條件; 循環條件 ){  
    條件滿足後做的事情;  
}

# 小挑戰



**請用可變電阻控制馬達角度！**

**提示：**可變電阻是類比輸入，需轉換可變電阻的數值把它對應到馬達輸出的角度。思考一下哪個函式功能，可以將數值範圍做轉換？

(可參考前幾頁的簡報)

# 解答篇

```
#include <Servo.h>
Servo myservo;

int potPin = 0; //定義可變電阻讀取接腳
int val; //定義可變電阻讀取數儲暫存用的變數
int voltage; //定義馬達角度

void setup() {
    myservo.attach(9);
    pinMode(potPin, INPUT); //定義可變電阻讀取接腳為輸入
}

void loop() { //讀取可變電阻電壓值
    val = analogRead(potPin); //val = 0 ~ 1023
    voltage = map(val, 0, 1023, 0, 180); //將val由0~1023線性變換為0~180並存入voltage
    myservo.write(voltage);
    //印出數值檢查執行成果
    Serial.print("val = ");
    Serial.print(val);
    Serial.print(";  voltage = ");
    Serial.print(voltage);
    Serial.println(); //空行
    delay(15);
}
```



# DFplayer 紿我唱起來 !!

[1 Arduino筆記\(88\):DF Player mini製作MP3播放器](#)

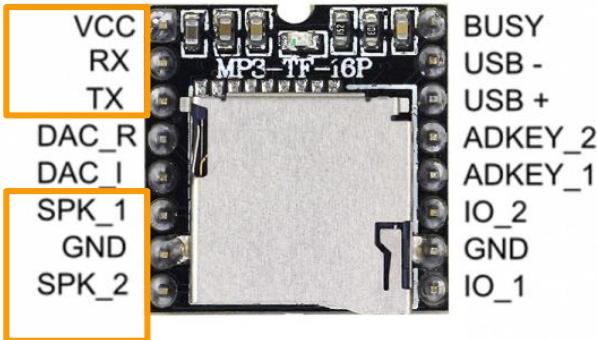
[2 【Arduino進階教學課程】自造 Arduino MP3音樂播放器 DFPlayer Mini](#)

# DFplayer 所需器材



- 備妥以上零件、模組及麵包板、杜邦線
- 在記憶卡中先建立名為 mp3 的資料夾，裡面存放至少 3 首歌  
**(每首長度超過 1 秒，檔案務必命名為 0001.mp3, 0002.mp3, 0003.mp3, ...)**
- 喇叭可先連接杜邦線(將線勾住喇叭後面的孔洞)

# 認識 DFplayer



接腳編號	接腳名稱	功能描述	備註
1	VCC	模塊電源輸入	3.3V – 5V，建議用5V，不可超過5.2V
2	RX	UART 串行數據輸入	
3	TX	UART 串行數據輸出	
4	DAC_R	音頻輸出右聲道	驅動耳機、功放
5	DAC_L	音頻輸出左聲道	驅動耳機、功放
6	SPK2	接小喇叭+	驅動小於3W 喇叭
7	GND	接地	電源地
8	SPK1	接小喇叭-	驅動小於3W 喇叭
9	IO1	觸發口	默認上一曲 (長按音量減)
10	GND	接地	電源地
11	IO2	觸發口	默認下一曲 (長按音量加)
12	ADKEY1	AD 口1	當觸發時是第一首(長按循環第一首)
13	ADKEY2	AD 口2	當觸發時是第五首(長按循環第五首)
14	USB+	USB+ DP	接U 盤或插電腦的USB 口
15	USB-	USB- DM	接U 盤或插電腦的USB 口
16	Busy	播放指示	有音頻輸出低，無音頻輸出高

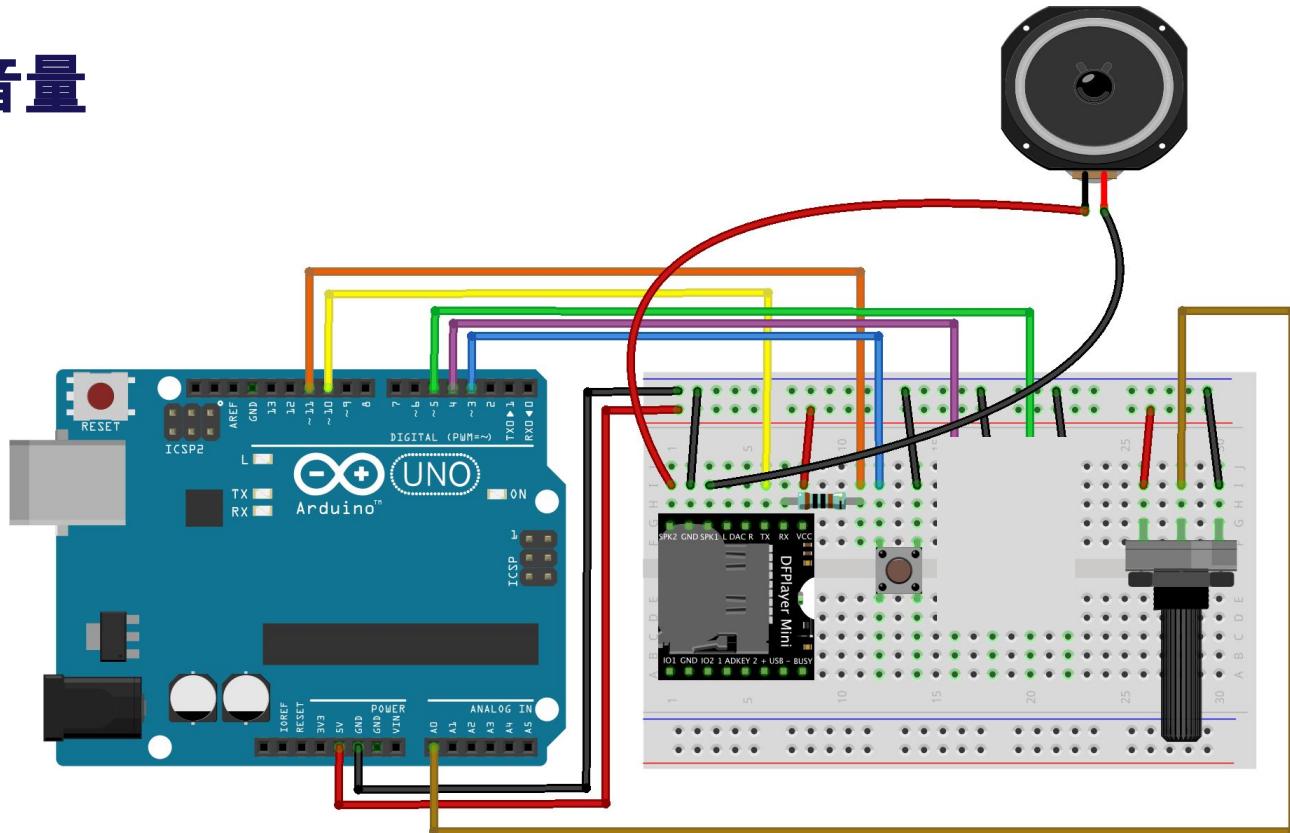
# DFplayer 命令參考



CMD 命令(指令)	對應的功能	參數(16位)
0x01	下一曲	
0x02	上一曲	
0x03	指定曲目(NUM)	1-2999
0x04	音量+	
0x05	音量	
0x06	指定音量	0-30
0x07	指定EQ 0/1/2/3/4/5	Normal/Pop/Rock/Jazz/Classic/Bass
0x08	單曲循環指定曲目播放0-2999	
0x09	指定播放設備 1/2/3/4/5	U 盤/SD/AUX/SLEEP/FLASH
0x0A	進入睡眠-- 低功耗	
0x0B	保留	
0x0C	模塊復位	
0x0D	播放	
0x0E	暫停	
0x0F	指定文件夾播放	1-10(需要自己設定)
0x10	擴音設置 (無)	[DH=1:開擴音] [DL:設置增益0-31]
0x11	全部循環播放	[1:循環播放][0:停止循環播放]
0x12	指定MP3文件夾目	0-9999
0x13	插播廣告	0-9999
0x14	支持15個文件夾	
0x15	停止插播，播放背景	
0x16	停止播放	

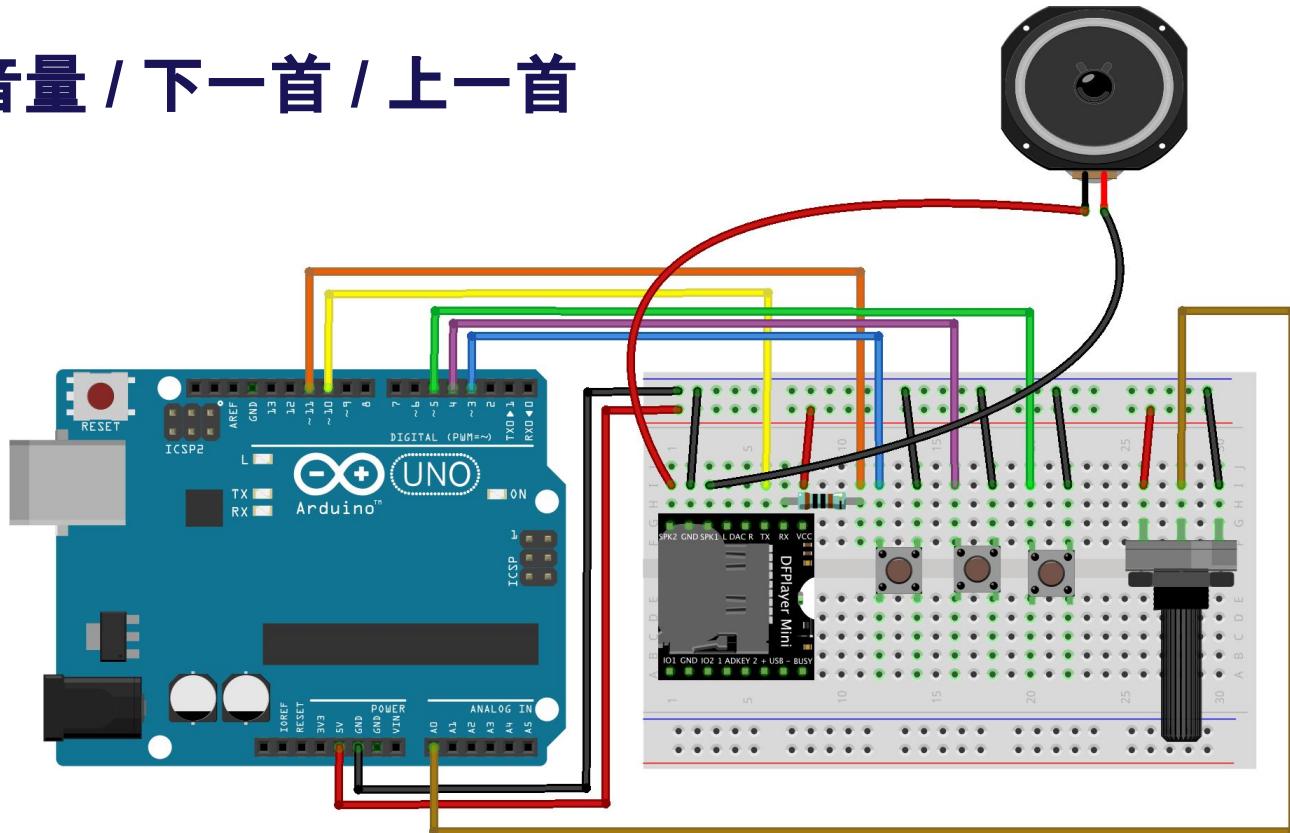
# DFplayer: 暫停 / 音量

1. 完成電路連接(按鈕先接左側的暫停鈕即可)
  2. 下載並在 Arduino 上傳程式碼(DFplayer 韌體已更新, 程式碼的 checksum 項目須移除)
  3. 若按鈕無法運作, 則將「連接按鈕的杜邦線」皆在按鈕下方



# DFplayer:暫停 / 音量 / 下一首 / 上一首

- 刪除程式中的以下註解：  
按鈕、playPrevious()、  
playNext()，使其執行
- 若按鈕無法運作，則將  
「連接按鈕的杜邦線」皆在  
按鈕下方



# 感測器補充介紹

Thanks !

# Q & A

Thanks !

# Arduino 是什麼？

