課本：超圖解ADUINO互動程式設計入門(新版)

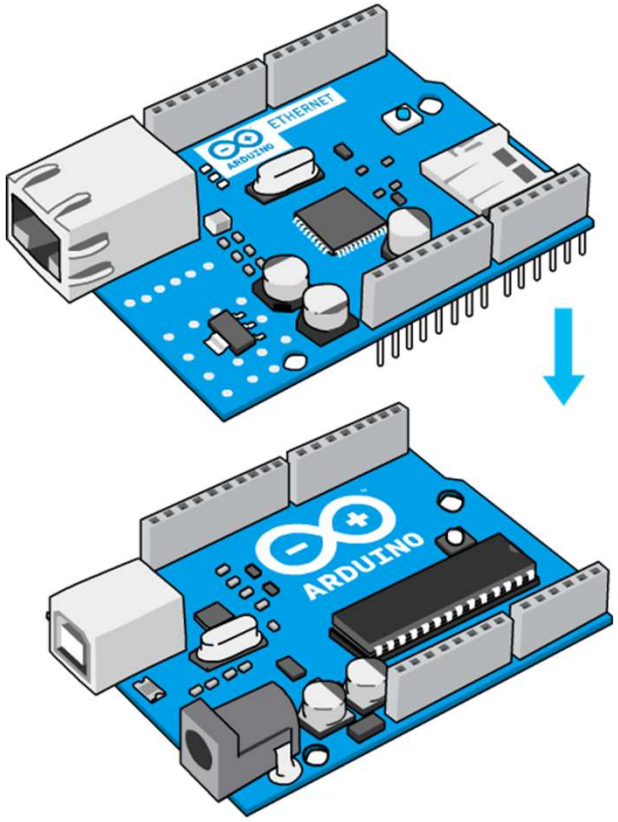
- ARDUINO UNO 3
  http://goods.ruten.com.tw/item/show?21521546948490
- 麵包板線約30條　http://goods.ruten.com.tw/item/show?21521546948490
- 2按鍵開關
- 8LED燈
- 1蜂鳴器　http://goods.ruten.com.tw/item/show?21601560855548
- 220歐姆電阻*12、1K,2K,10K歐姆電阻*1
- 麵包板大　http://goods.ruten.com.tw/item/show?21522602096409
- 共陰全彩LED燈*1　http://goods.ruten.com.tw/item/show?21521546948490
- 光敏電阻*1　http://goods.ruten.com.tw/item/show?21521546948490
- 10K可變電阻*1　http://goods.ruten.com.tw/item/show?21521546948490
- 共陰七段顯示器*1

課本：超圖解ADUINO互動程式設計入門(新版)

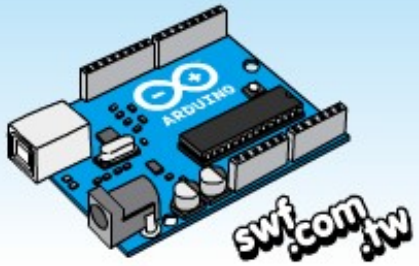
- MAX7219的8*8 LED矩陣模組 *1
  http://goods.ruten.com.tw/item/show?21540784515833
- 16*2 1602 LCD
  http://goods.ruten.com.tw/item/show?21505380775171
- 溫濕度感測器　　http://goods.ruten.com.tw/item/show?216144301017 66
- 超音波感測器　　http://goods.ruten.com.tw/item/show?21407175167270

- 如果有興趣玩自走車的同學可以買(不在必要範圍內)
- Arduino 手機藍芽控制小車學習套件 3輪
  http://goods.ruten.com.tw/item/show?216197159727 63

第一章 認識Arduino

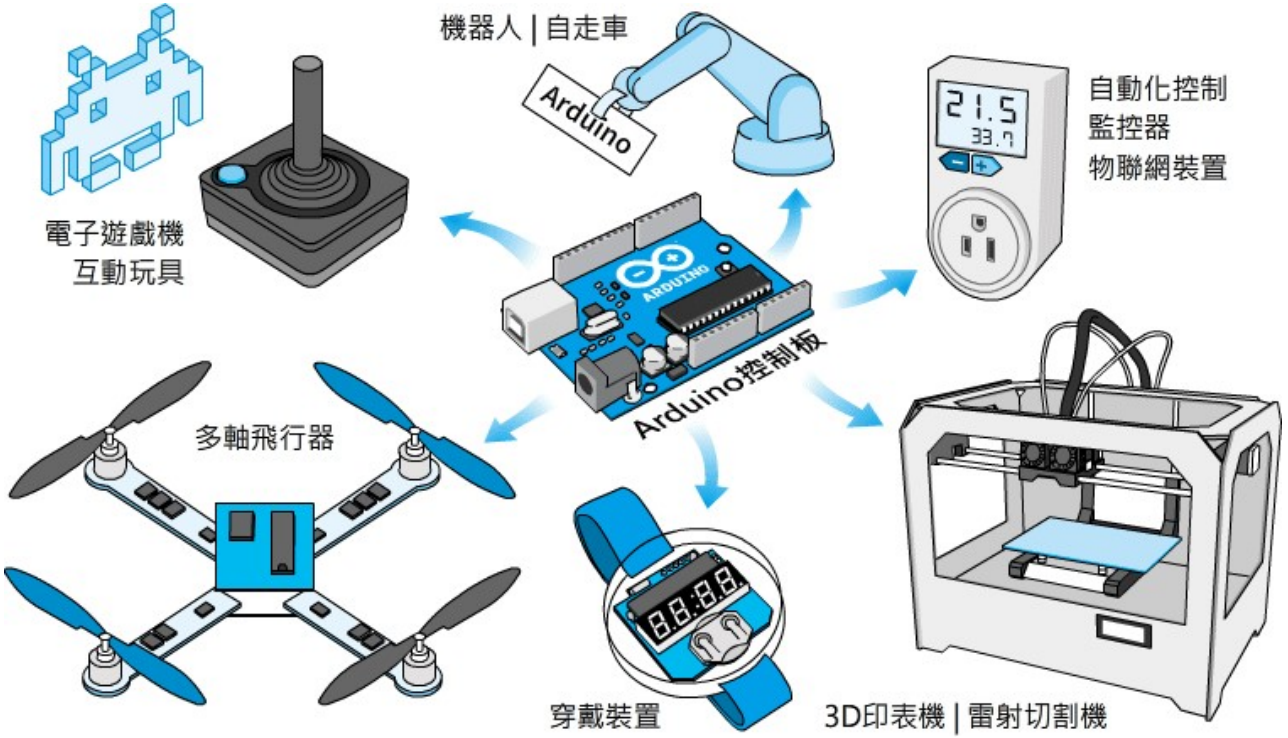


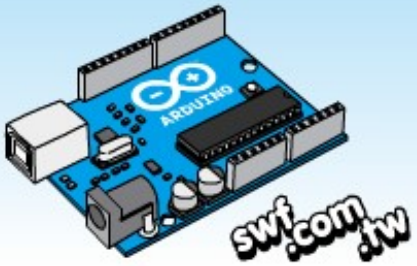
- Arduino微電腦板
- Arduino的擴充板（shield）
- Arduino的接腳
- ATmega328微控器
- Arduino程式開發步驟
- Arduino開發環境安裝與使用



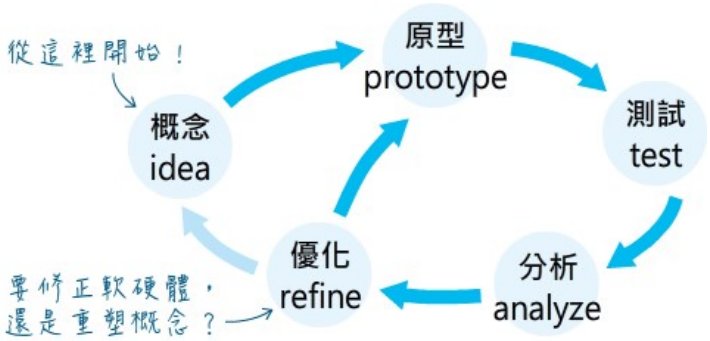
Arduino簡介

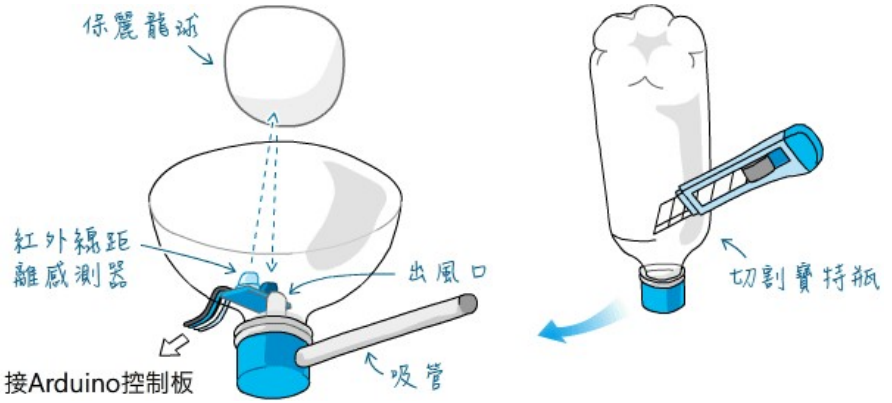
Arduino：開源（open source）微電腦控制板以及程式開發工具。

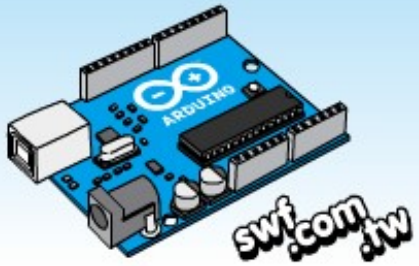



ARDUINO
swf.com.tw

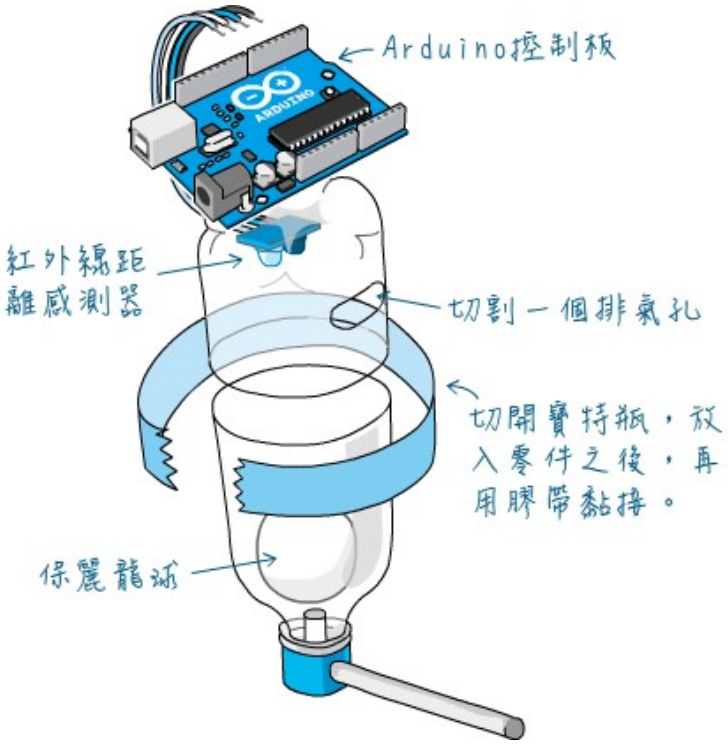
打造原型的過程（一）

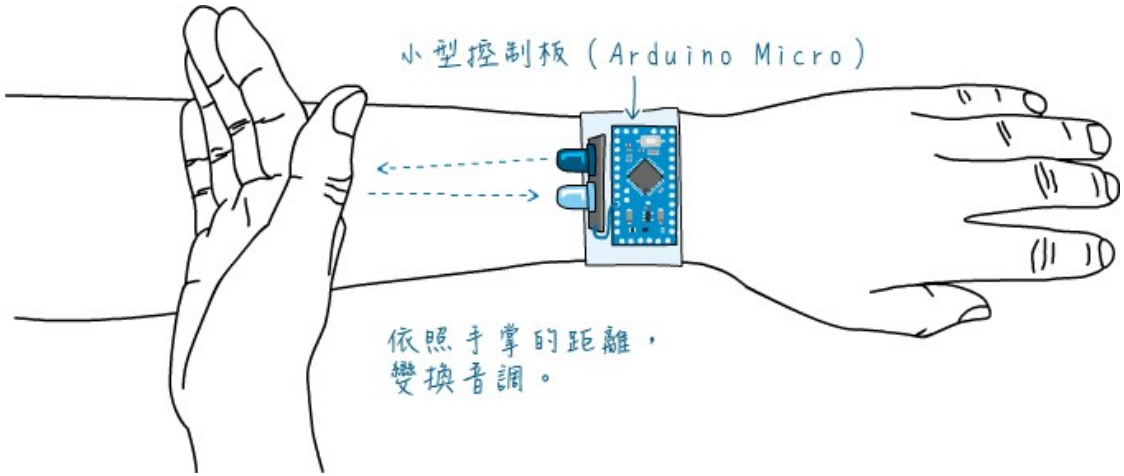

從這裡開始！
概念
idea
原型
prototype
測試
test
分析
analyze
優化
refine
要修正軟硬體，
還是重塑概念？

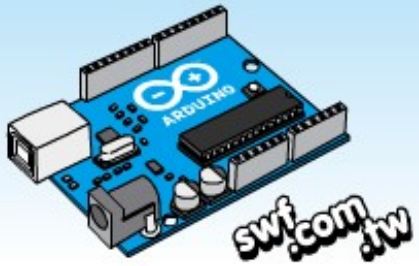

保麗龍球
紅外線距
離感測器
出風口
接Arduino控制板
吸管
切割寶特瓶


ARDUINO
swf.com.tw

打造原型的過程（二）

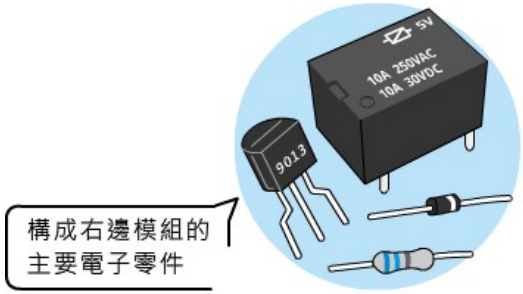

Arduino控制板
紅外線距離感測器
切割一個排氣孔
切開寶特瓶，放入零件之後，再用膠帶黏接。
保麗龍球

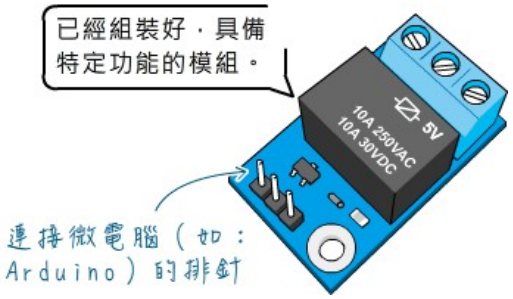

小型控制板（Arduino Micro）
依照手掌的距離，變換音調。

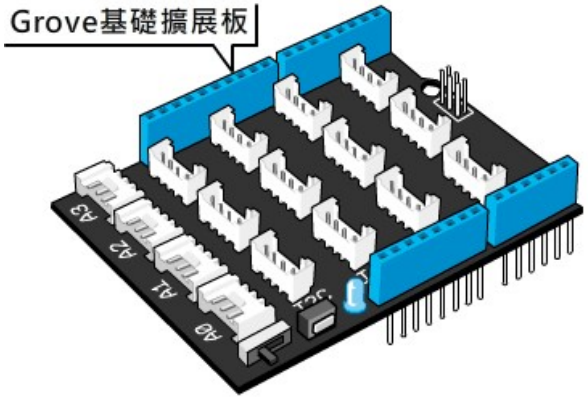


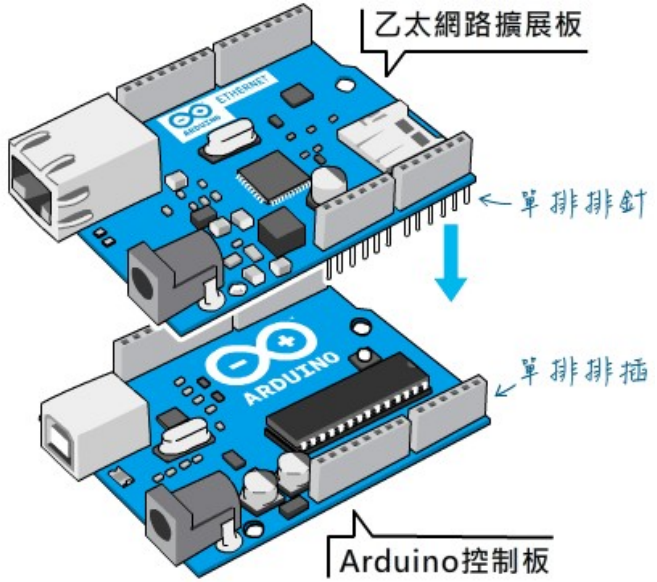
Arduino的擴展板（Shield）

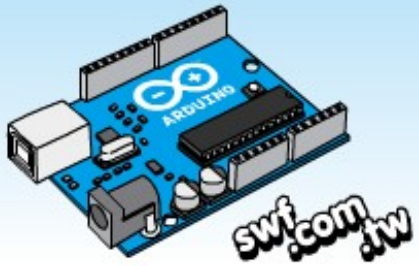
微電腦控制板就像是一個具有大腦和神經線，但是沒有感官和行動能力的物體。



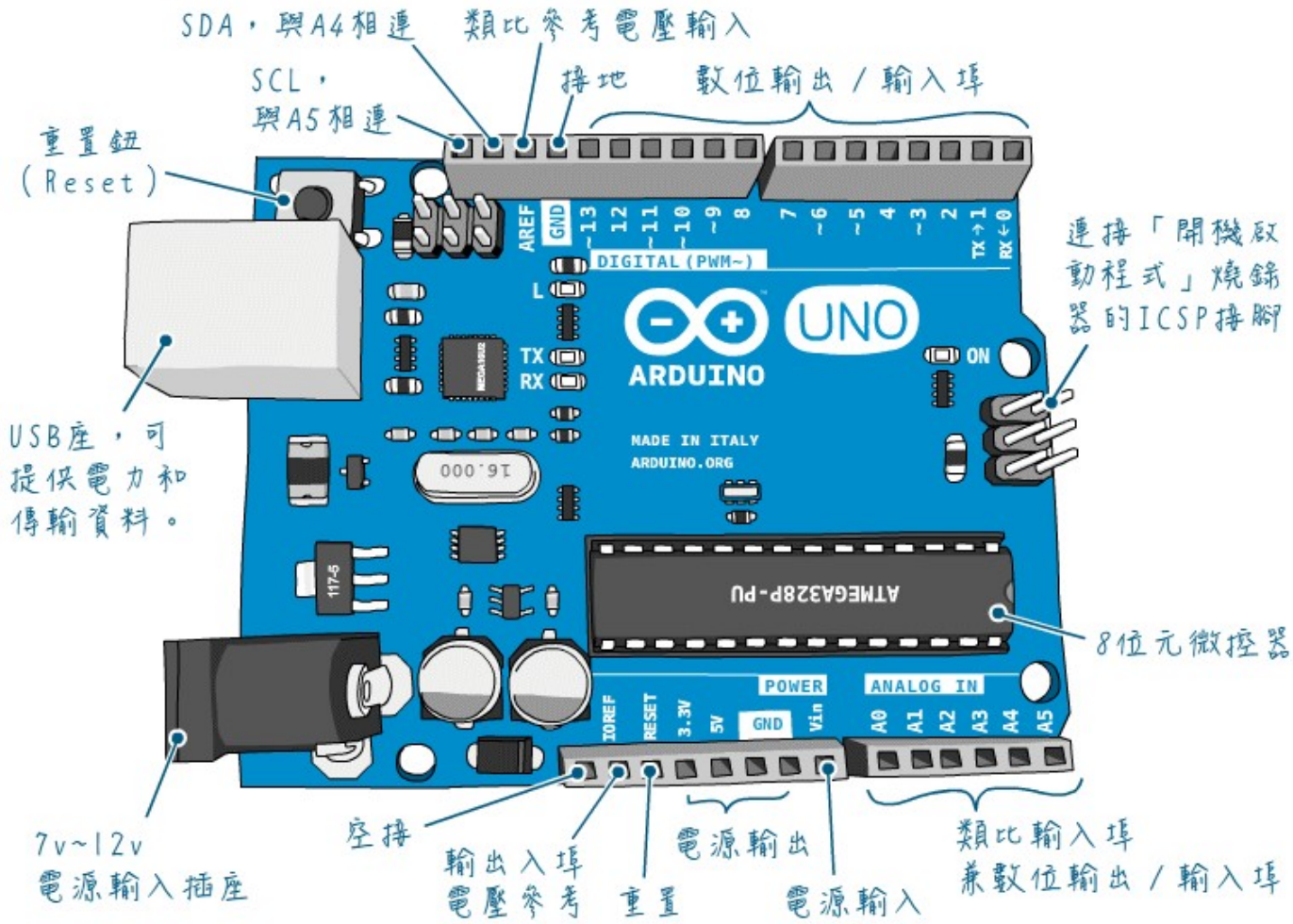


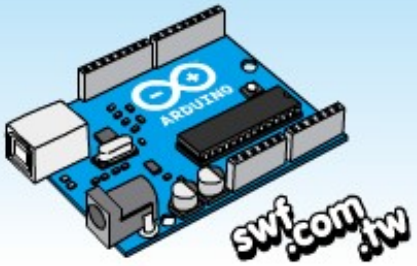





ARDUINO
swf.com.tw

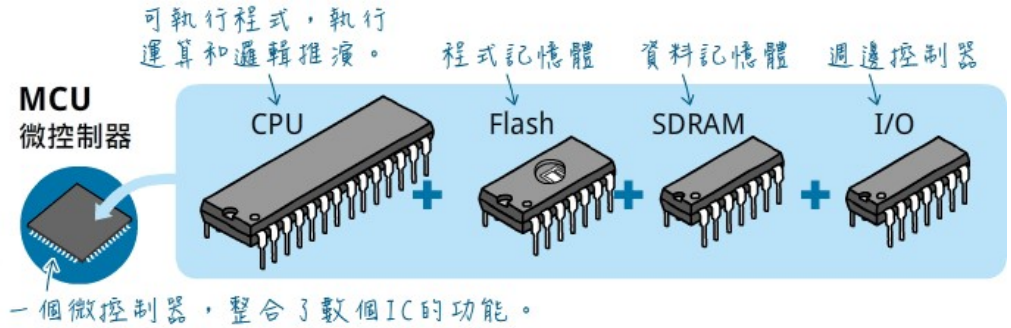
Arduino的接腳

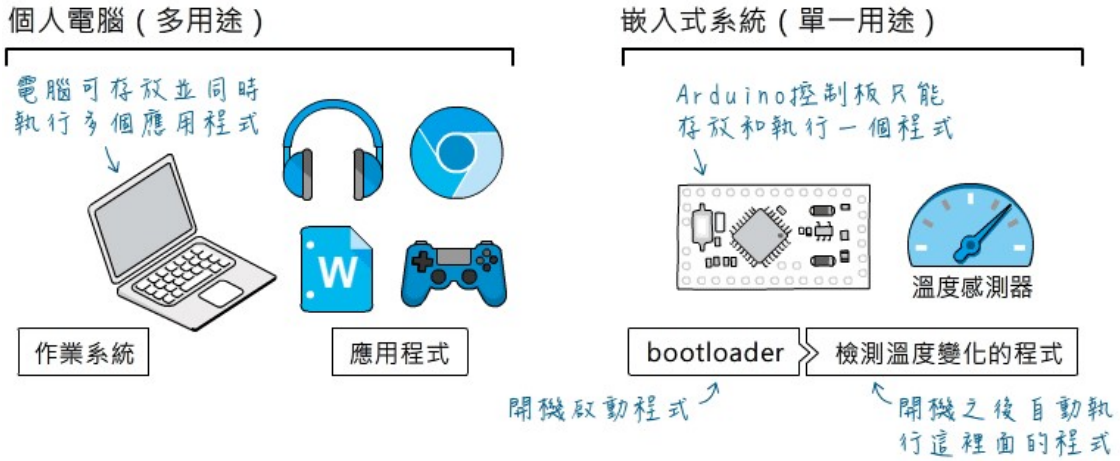

SDA，與A4相連
類比參考電壓輸入
SCL，
與A5相連
接地
數位輸出／輸入埠
重置鈕
(Reset)
連接「開機啟動程式」燒錄器的ICSP接腳
USB座，可提供電力和傳輸資料。
8位元微控器
7v~12v
電源輸入插座
空接
輸出入埠
電壓參考
重置
電源輸出
電源輸入
類比輸入埠
兼數位輸出／輸入埠
AREF
GND
DIGITAL (PWM~)
TX
RX
ON
UNO
ARDUINO
MADE IN ITALY
ARDUINO.ORG
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
Vin
A0
A1
A2
A3
A4
A5

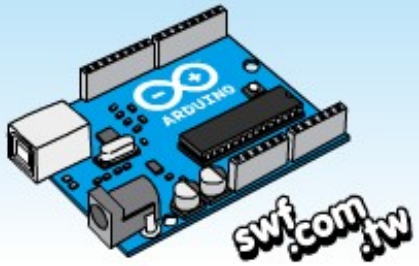

ARDUINO
swf.com.tw

什麼是微控制器？


ARDUINO
ON
哇！智慧型遙控燈具！
普通的電燈
Arduino板和控制模組
手機App

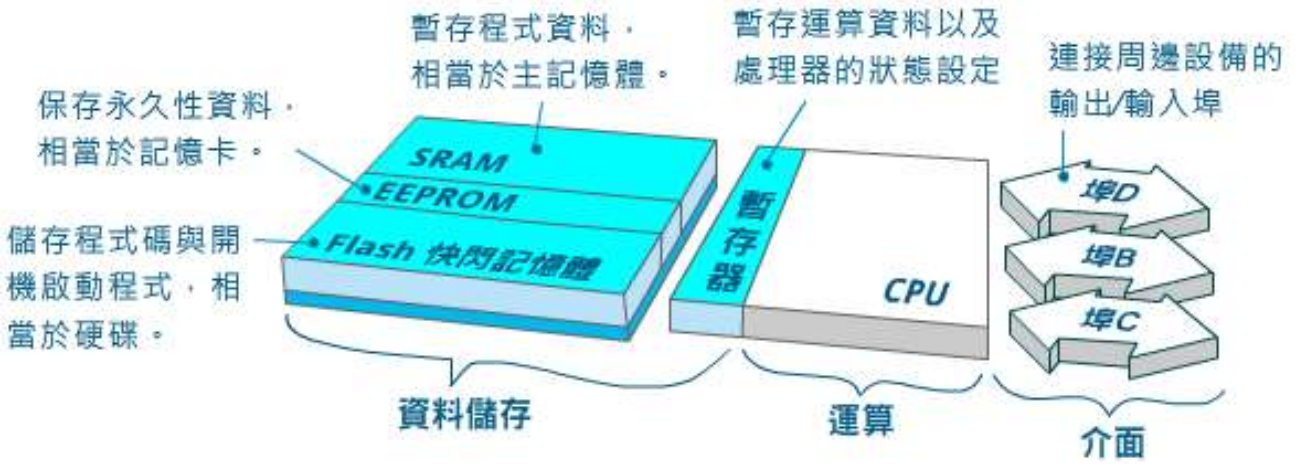

可執行程式，執行運算和邏輯推演。
程式記憶體
資料記憶體
週邊控制器
MCU
微控制器
CPU
Flash
SDRAM
I/O
一個微控制器，整合了數個IC的功能。


個人電腦（多用途）
電腦可存放並同時執行多個應用程式
作業系統
應用程式
嵌入式系統（單一用途）
Arduino控制板只能存放和執行一個程式
溫度感測器
bootloader
檢測溫度變化的程式
開機啟動程式
開機之後自動執行這裡面的程式



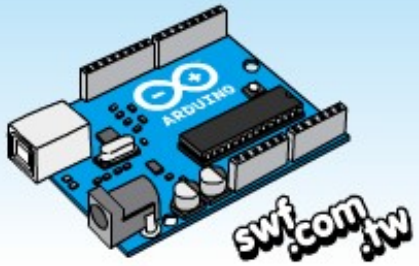
認識ATmega328微控器

Arduino的微控器（Microcontroller）包含CPU、記憶體、類比/數位訊號轉換器以及周邊控制介面。



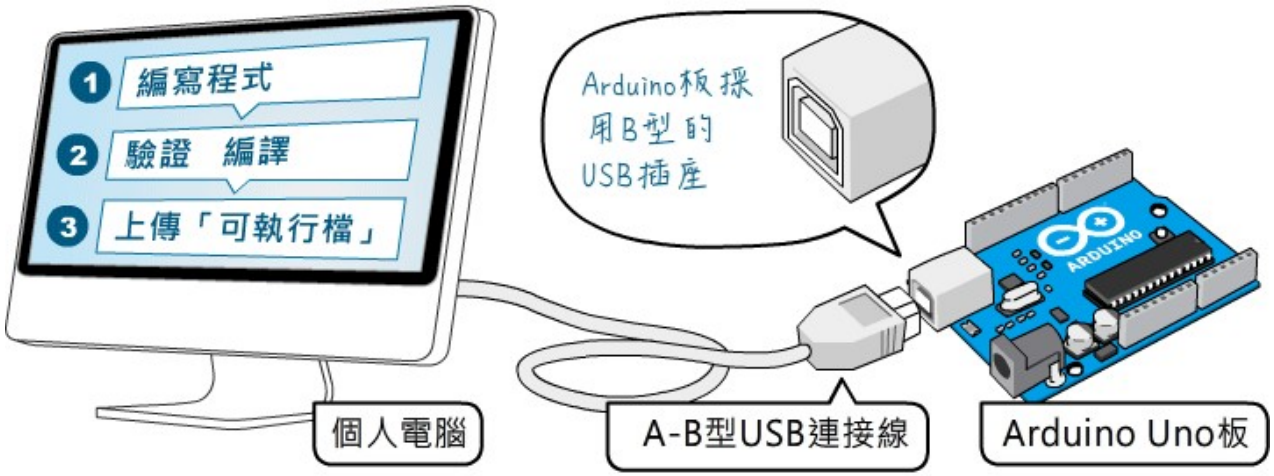
微控器內部具有三種記憶體：

名稱	類型	容量大小	用途
SRAM	揮發性（volatile），代表資料在斷電後消失。	2048bytes (2KB)	資料記憶體；暫存程式運作中所需的資料。
Flash	非揮發性，代表斷電後，資料仍存在。	32768bytes (32KB)	程式記憶體；存放開機啟動程式和我們自訂的程式碼。
EEPROM	非揮發性	1024bytes (1KB)	存放程式的永久性資料。



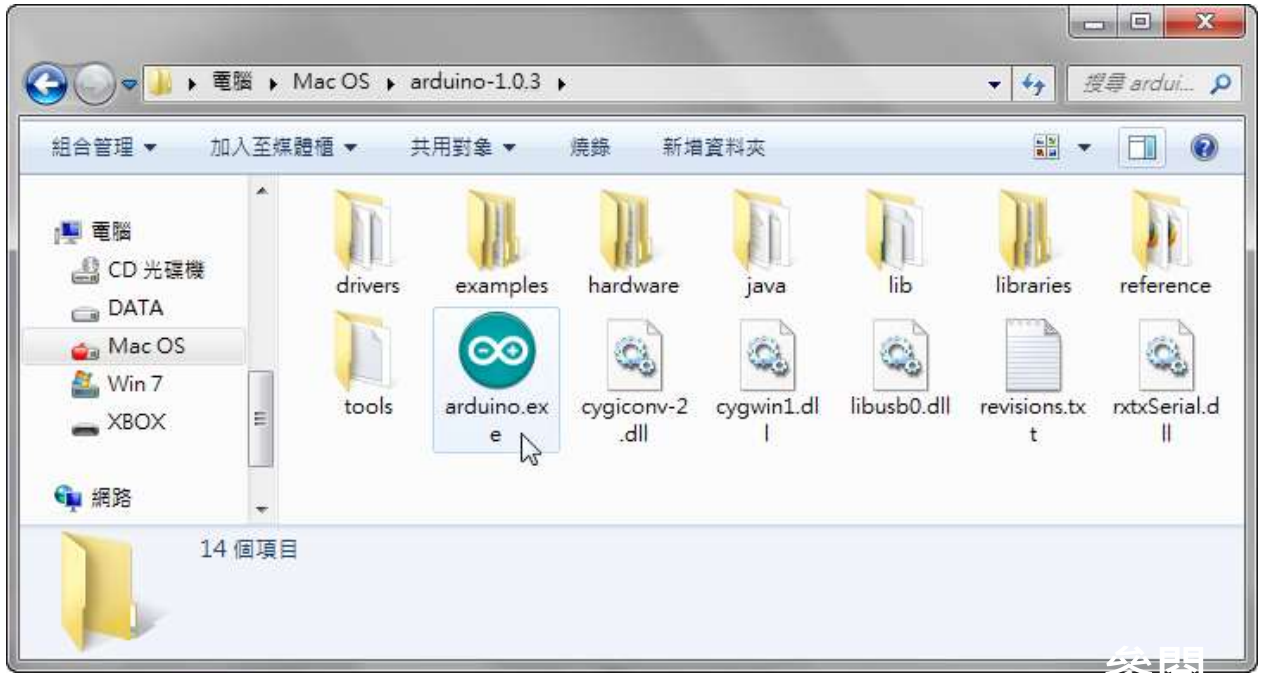
Arduino程式開發步驟

1.規劃裝置的功能和軟　硬體：裝置有什麼用途？需要哪些輸入裝置或感測器元件？有什麼輸出結果？
2.組裝硬體：通常使用麵包板把電子零件組裝起來。
3.編寫程式
4.驗證和編譯：檢查程式內容是否有錯誤，並且編譯（compile）原始碼。
5.上傳：也稱為燒錄，把編譯完畢的程式寫入微處理器內部的記憶體。

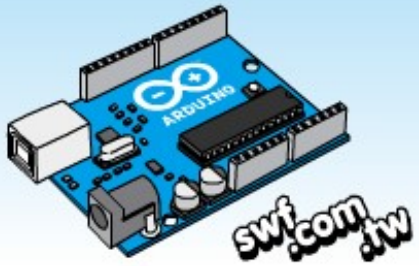


Arduino開發環境安裝

Arduino程式開發工具可以在[arduino.cc下載](arduino.cc)



用USB連接Arduino和電腦，Arduino將被當成序列埠連線裝置。

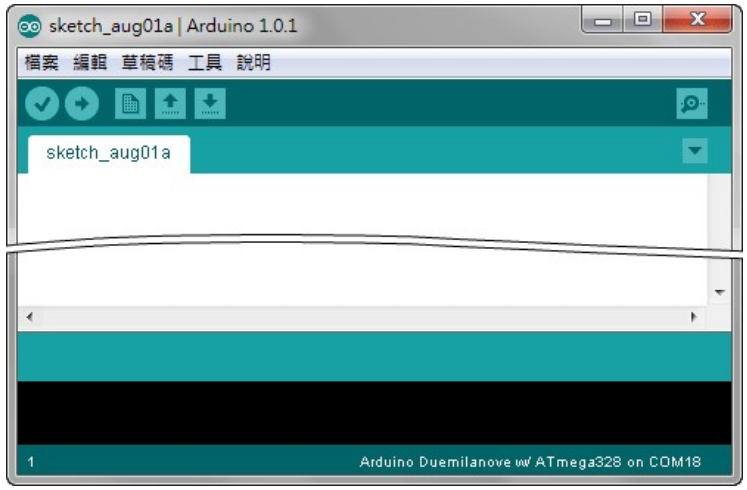


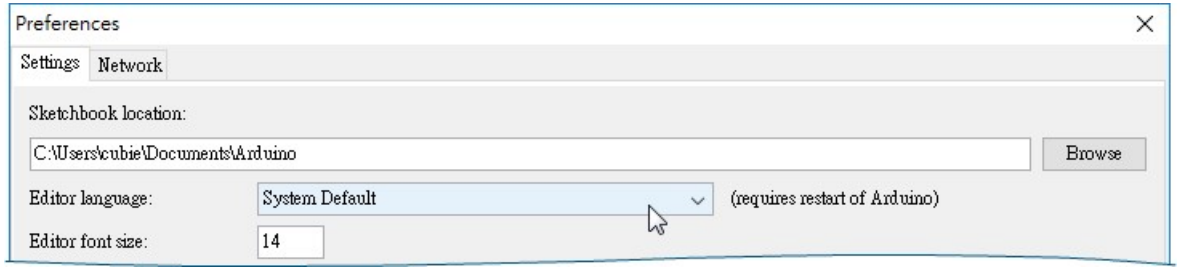
設定中文操作介面

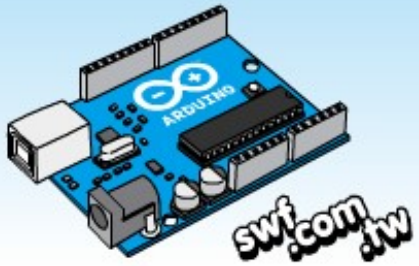
Arduino的程式檔統稱"sketch"，中文版翻譯成草稿碼，本書將混和使用「程式檔」和「草稿碼」這一詞。

選擇主功能表的**File→Preferences**（檔案→偏好設定）指令，從**Preferences**（偏好設定）面板裡的語言選項，選擇“System default”或 “Chinese (Taiwan)” 。

設定完畢後，關閉Arduino再重新開啟。

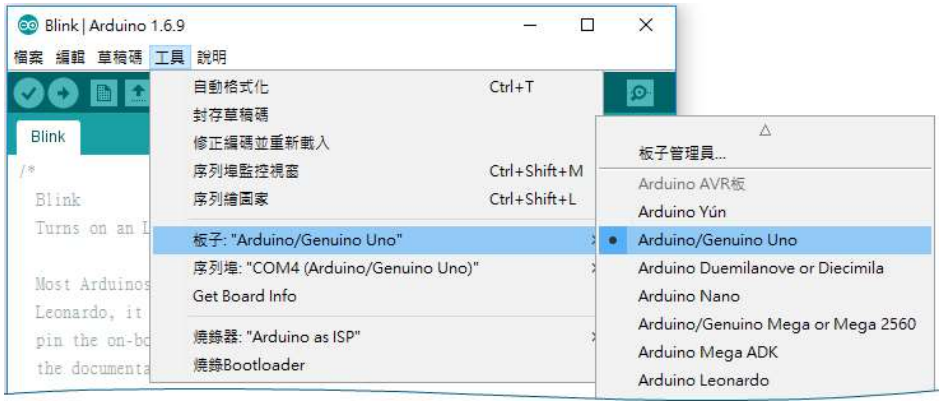


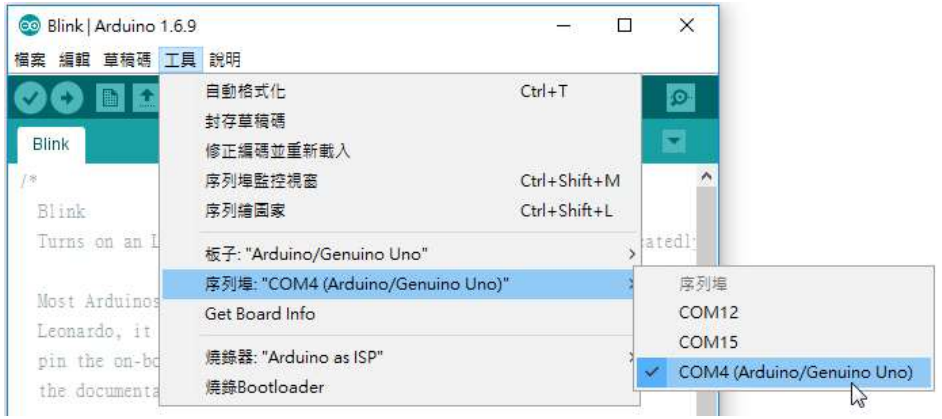


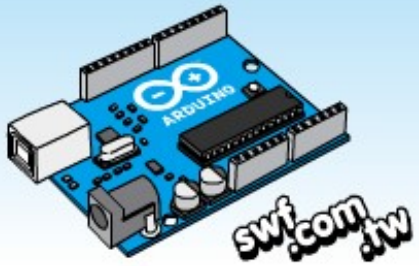


設定Arduino板的選項

第一次使用Arduino程式開發工具時，請先把Arduino板接上電腦的USB，接著在Arduino程式開發工具裡面設定你的Arduino板子類型，以及序列埠編號。

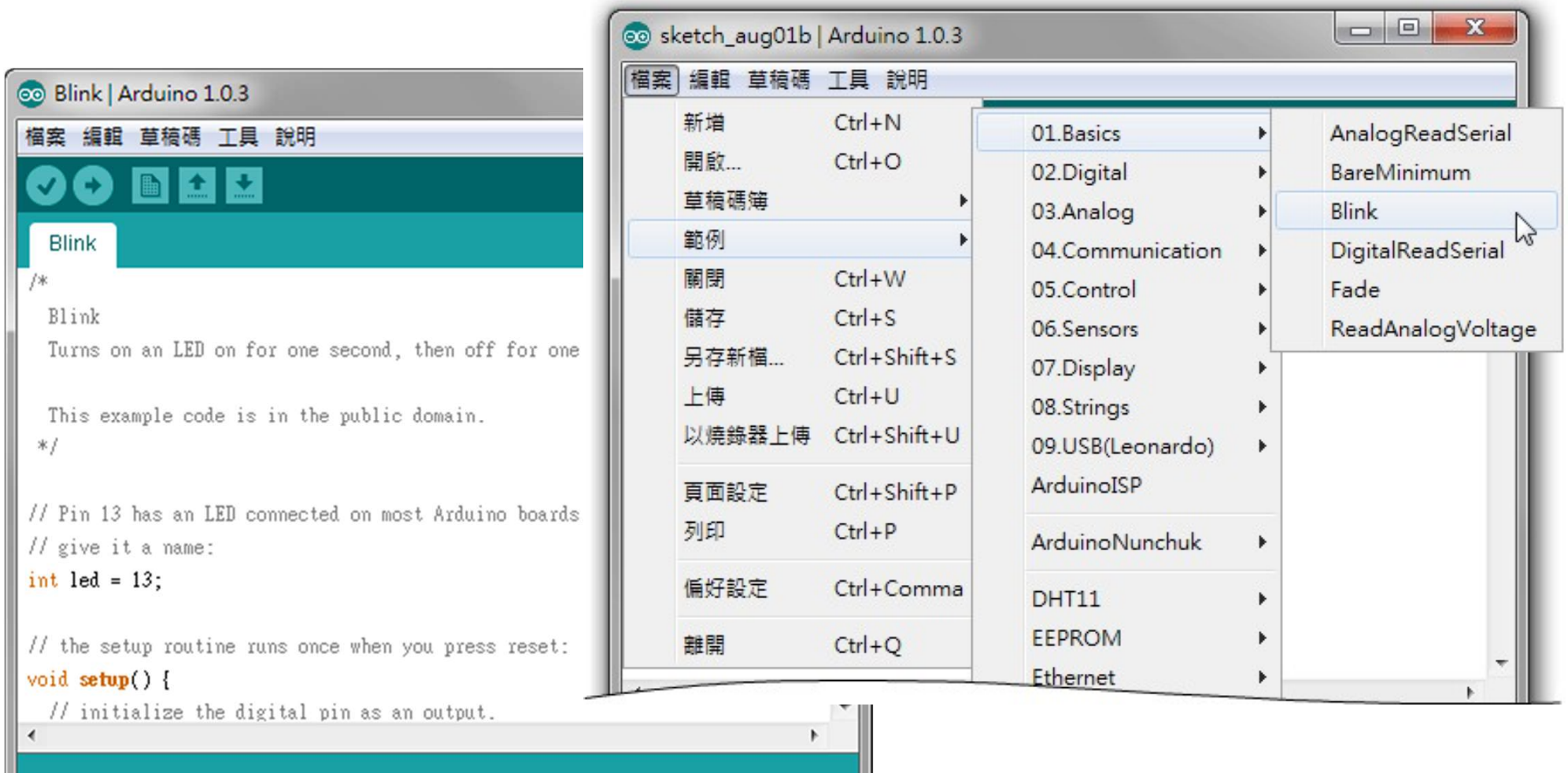


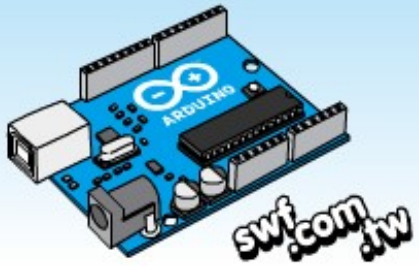




上傳「LED閃爍」程式

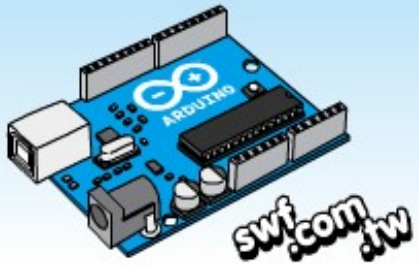
Arduino程式開發工具裡面包含許多現成的範例程式，放在主功能表的「**檔案→範例**」底下。





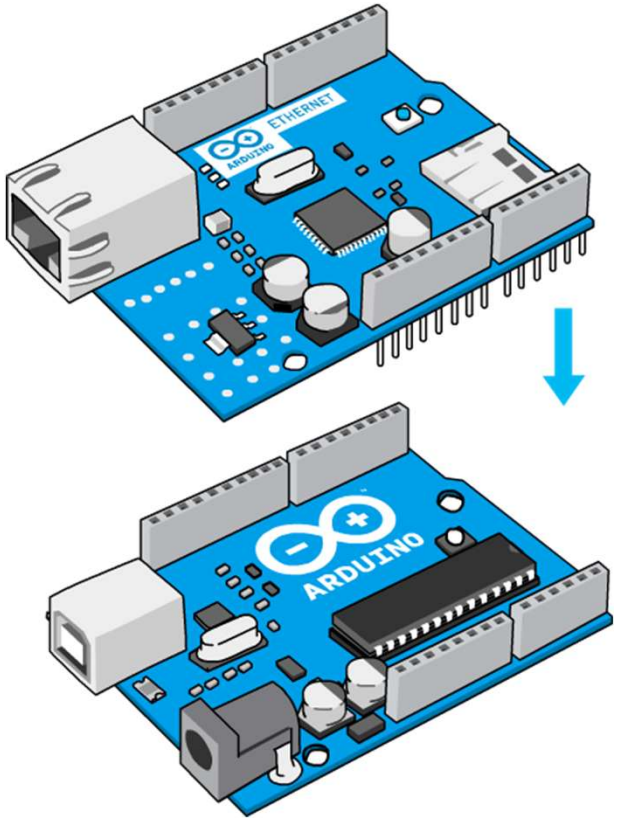
電路學概念

- 接地GND
- 電源VCC
- 串聯
- 並聯
- 麵包版
- 電子元件 +長腳；-短腳。無極性:一樣長
- 先接地，接電阻，接元件，接電源

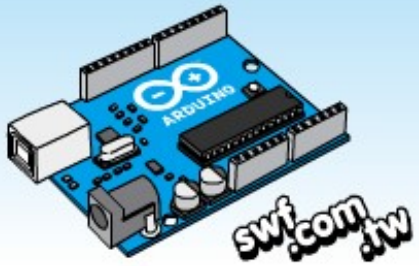


- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

第二章 認識電子零件與工具

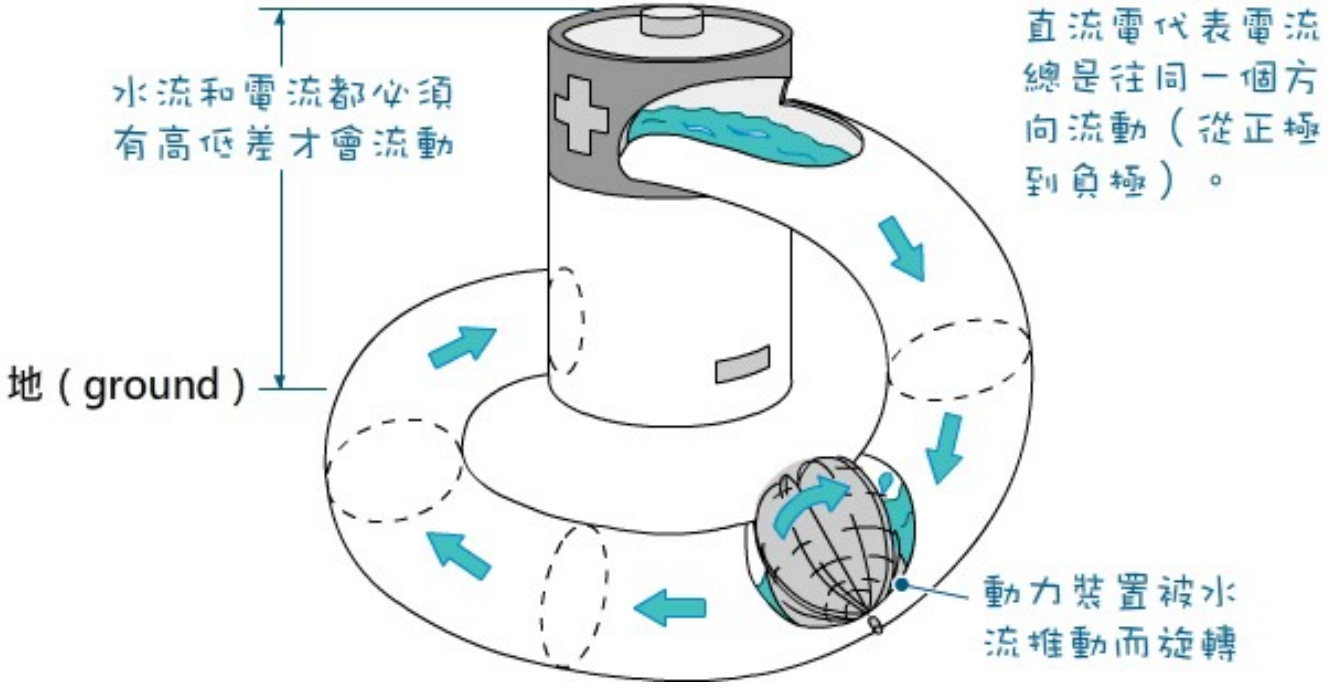


- 電壓、電流與接地
- 電阻
- 電容
- 二極體和LED
- 操作萬用電錶
- 看懂電路圖
- 用麵包板接電路

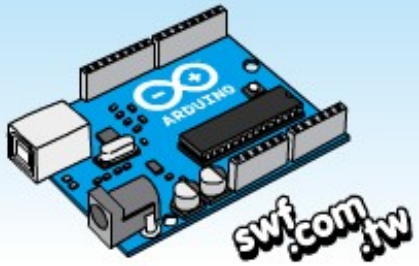


電壓、電流與接地

電位差或電勢差，通常稱為電壓，代表推動電流能力的大小，其單位是伏特（volt，簡寫成V）。處於高位者稱為**正極**，低地勢者為**負極**或**接地**（Ground，簡稱GND）

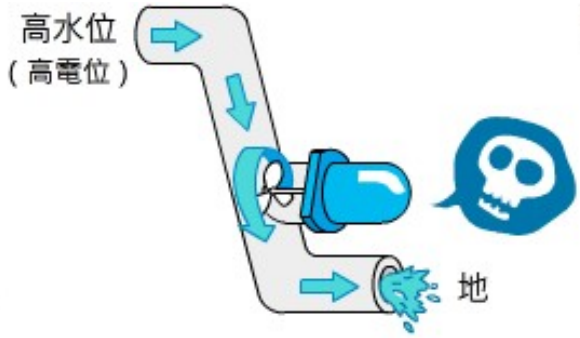


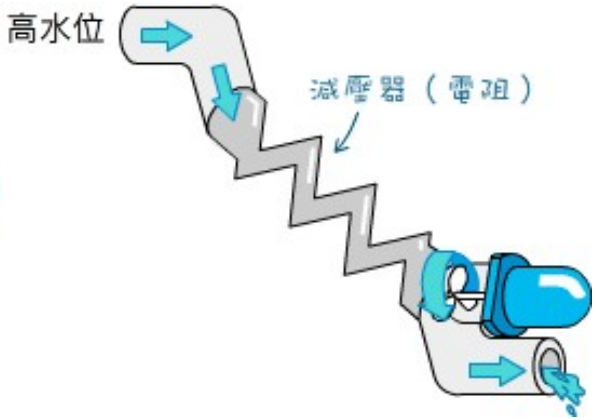
電壓的電路符號

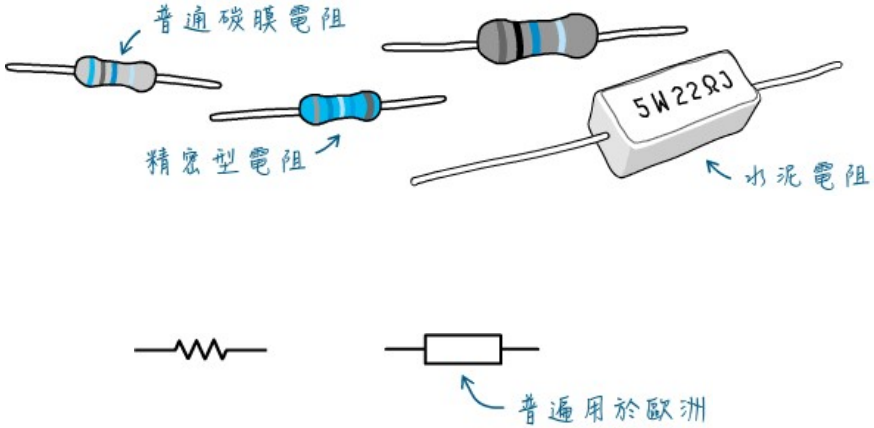


電阻

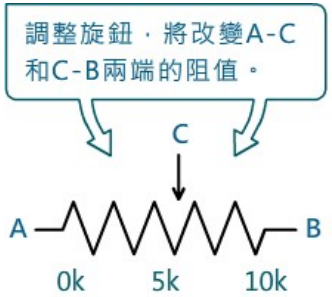
阻礙電流流動的因素叫電阻。電阻能降低和分散電子元件承受的電壓，避免元件損壞。



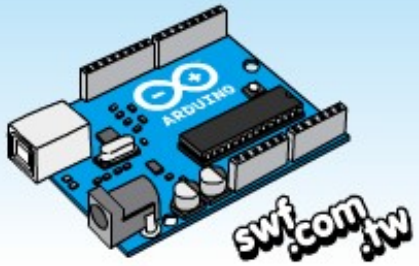




有些電阻具備可調整阻值的旋鈕，稱為可變電阻（簡稱VR）



概念簡圖（電路符號）



電阻的色環

第1色		第2色		乘數		誤差率	
黑	0	黑	0	黑	0		
棕	1	棕	1	棕	1	棕	±1%
紅	2	紅	2	紅	2	紅	±2%
橙	3	橙	3	橙	3		
黃	4	黃	4	黃	4		
綠	5	綠	5	綠	5	綠	±0.5%
藍	6	藍	6	藍	6	藍	±0.25%
紫	7	紫	7	紫	7	紫	±0.1%
灰	8	灰	8	灰	8		
白	9	白	9	白	9		
				金	-1	金	±10%
				銀	-2	銀	±5%

棕 黑 紅

(1 0) x 10^2 = 1000Ω

(1KΩ)

紅 紅 橙

(2 2) x 10^3 = 22000Ω

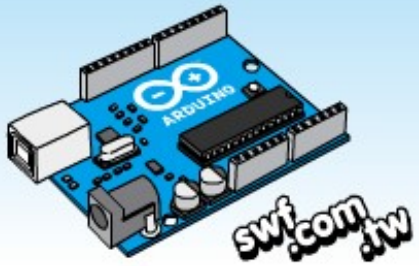
(22KΩ)

黃 紫 黑

(4 7) x 10^0 = 47Ω

橙 紅 棕

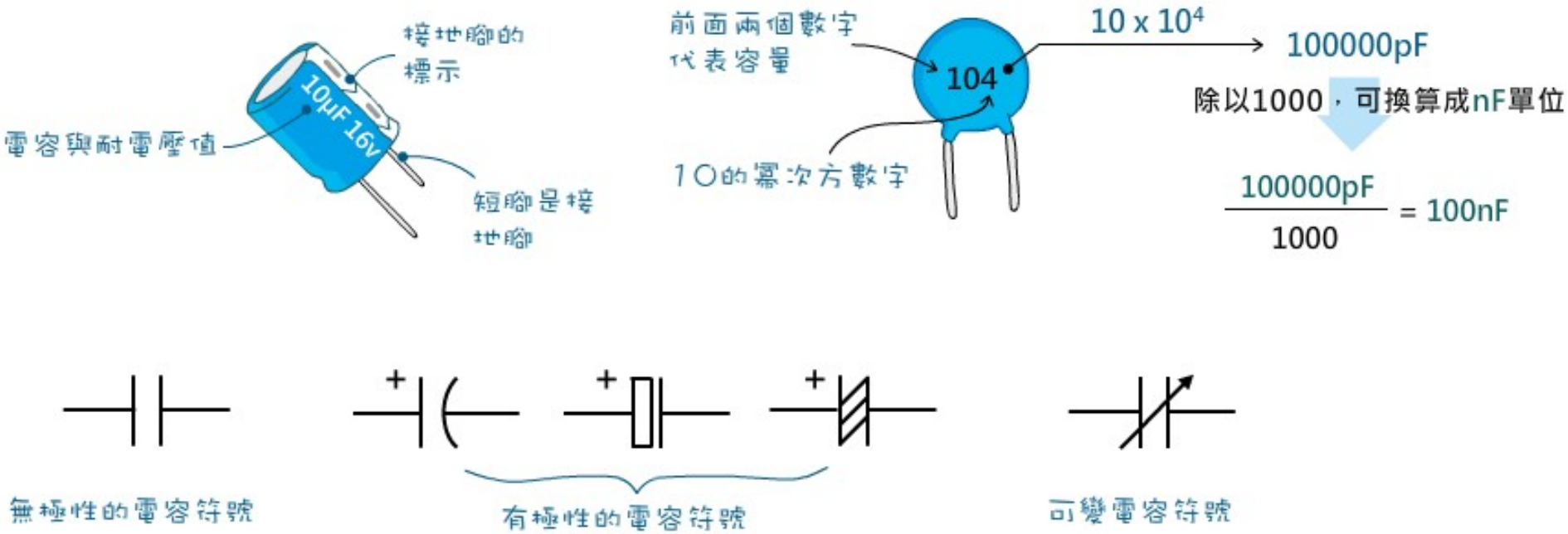
(3 2) x 10^1 = 320Ω

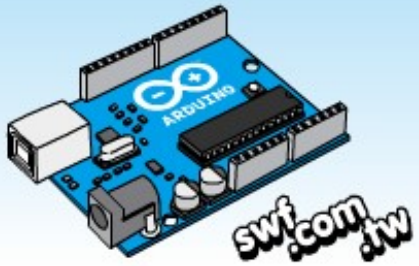


電容

電容器就是電的容器，簡稱電容，單位是法拉（Farad，簡寫成F），代表電容所能儲存的電荷容量。

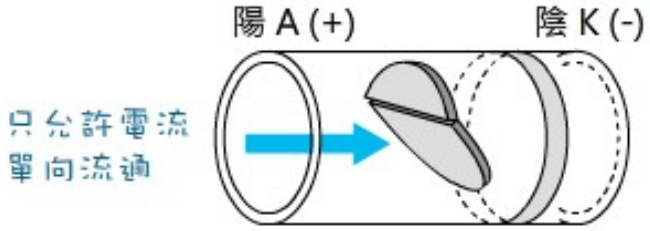
在積體電路和馬達的電源接腳，經常可以發現電容，用於吸收電源瞬間變化的雜訊。

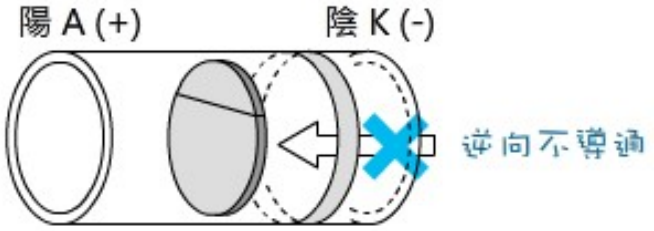


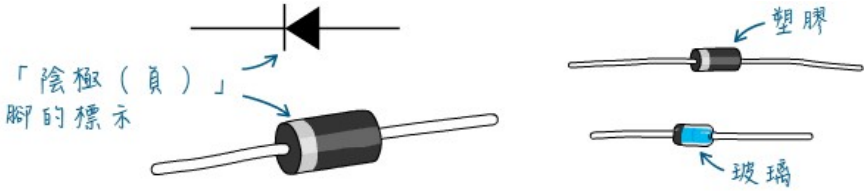


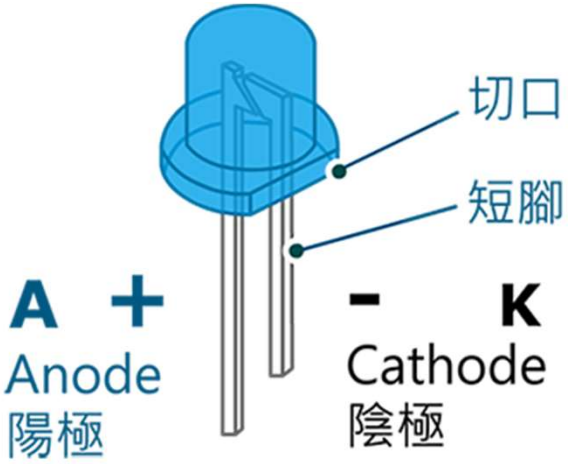
二極體和LED

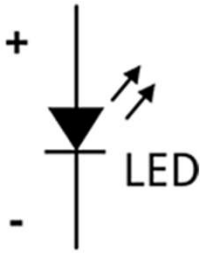
二極體是一種單向導通的半導體元件，其接腳有區分極性。



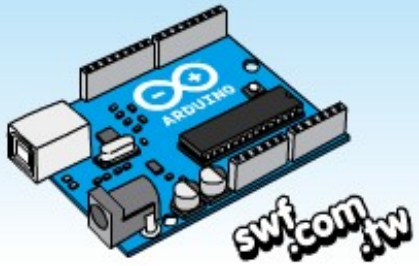






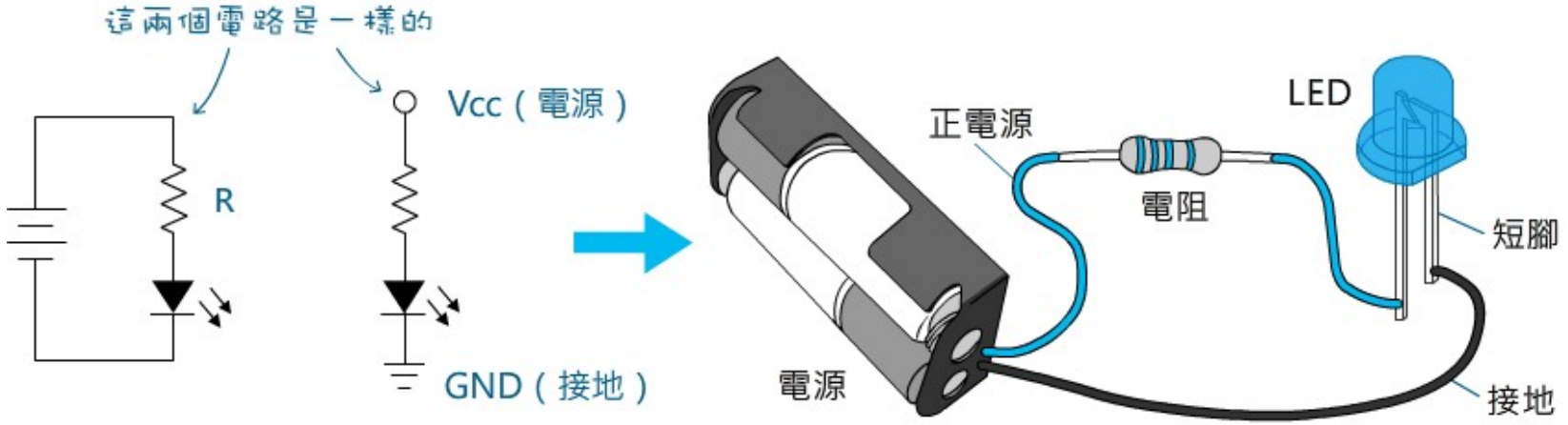


LED同樣是單向導通元件，若接反了它不會亮。長腳接正極（+）、短腳接負極（-，或接地）。



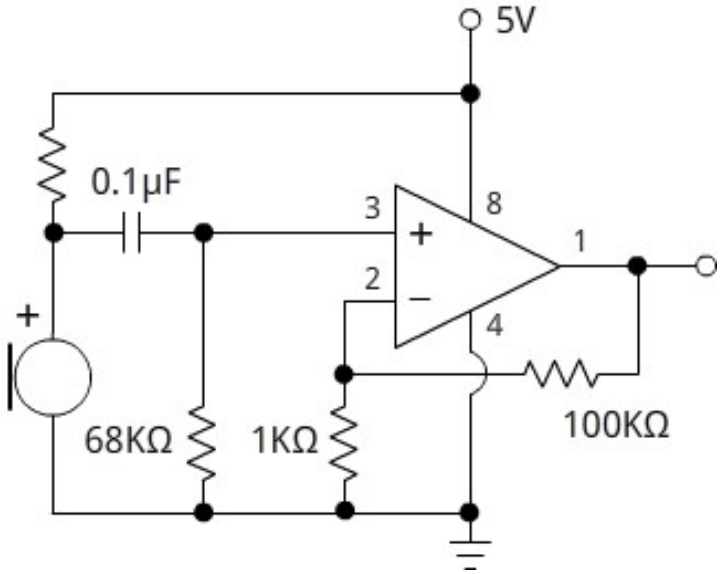
看懂電路圖

電路圖就是展示電子裝置所需的零件型號，以及零件如何相連的藍圖。



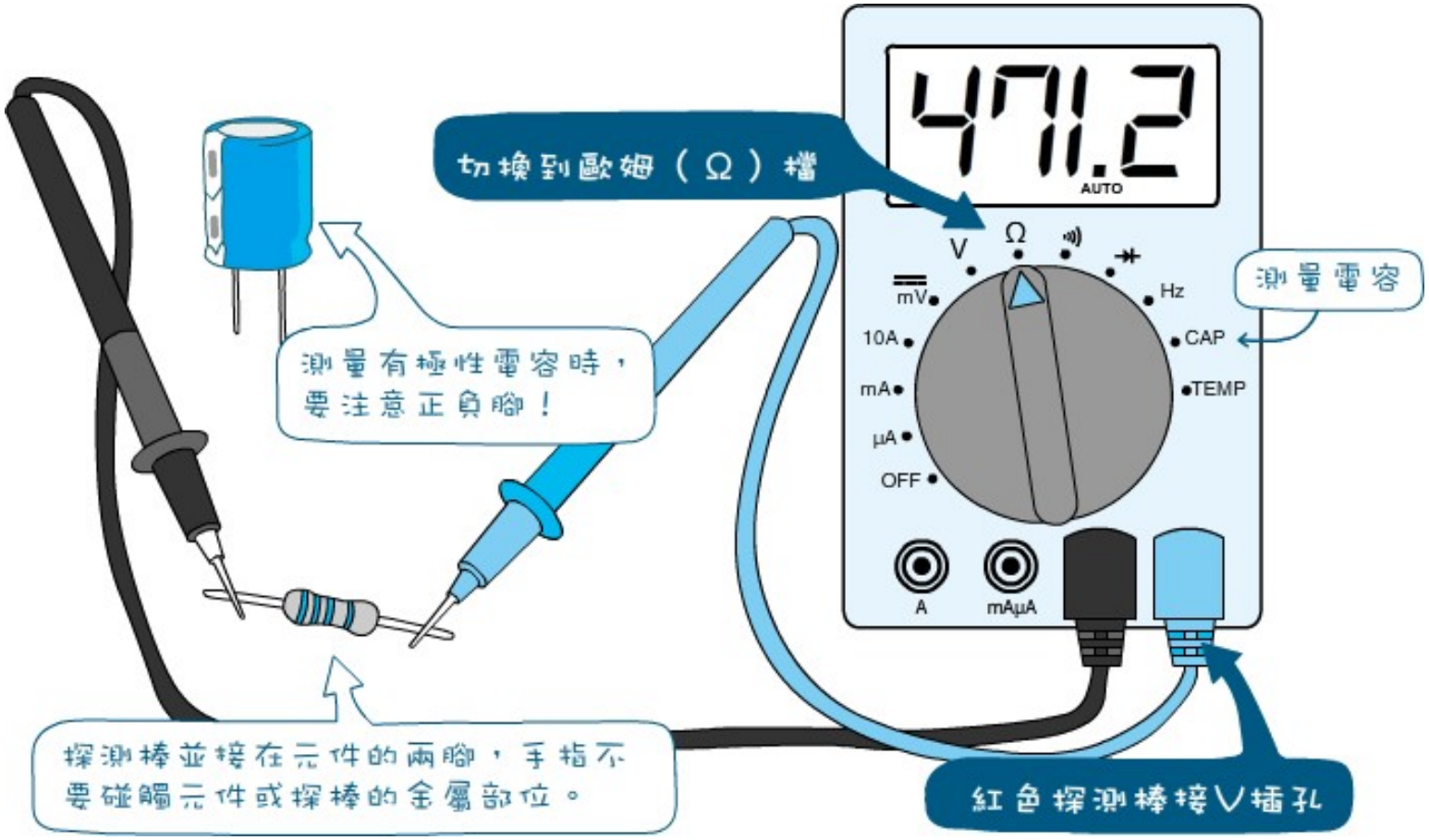
電路圖中，兩條交錯的線條：

- 如果交接處有一個小黑點，代表線路相連。
- 如果交接處沒有圓點，代表線路沒有相連。



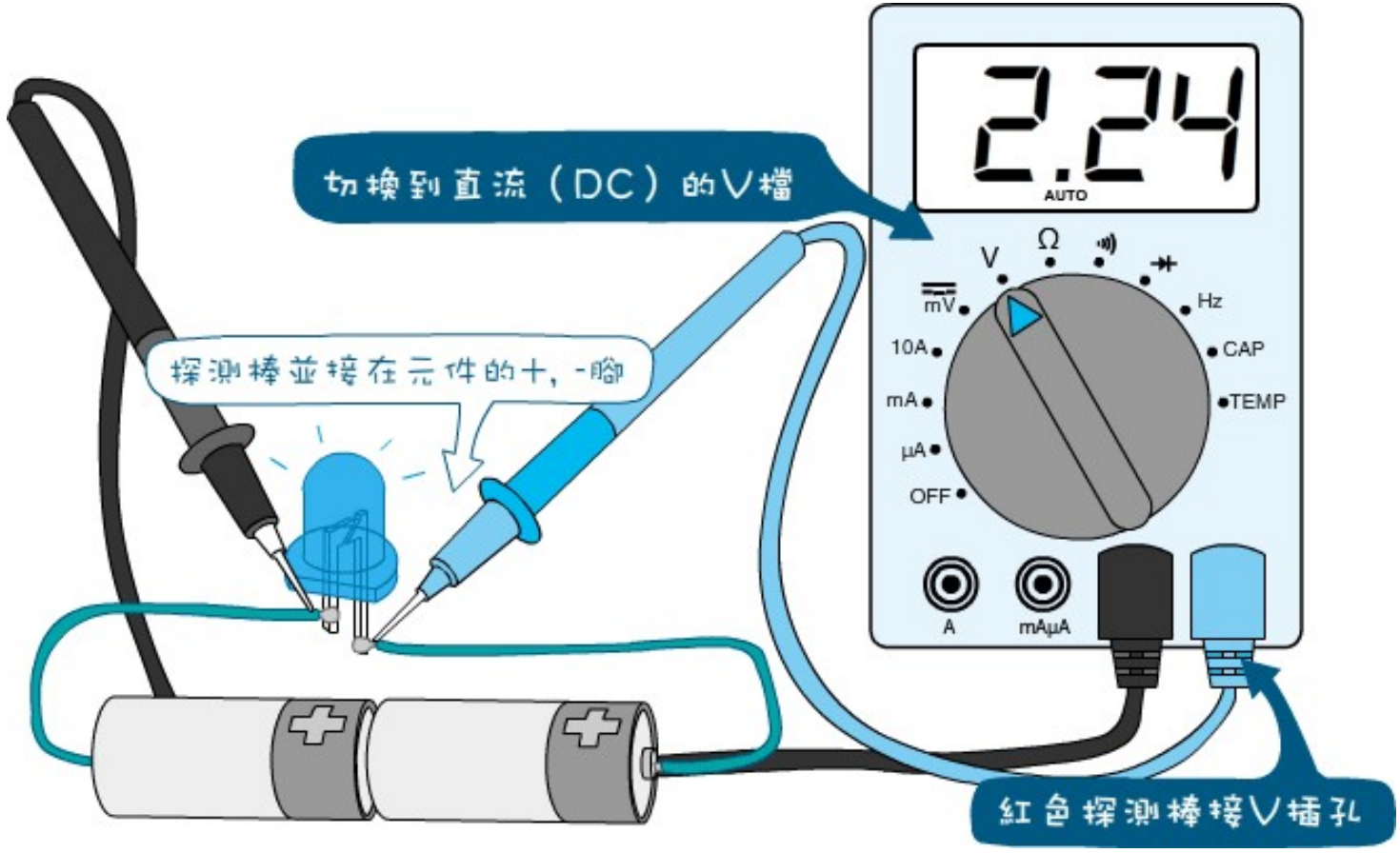
測量電阻或電容

一般稱為「三用電錶」或「萬用電錶」，主要用於測量電壓、電流和電阻值。



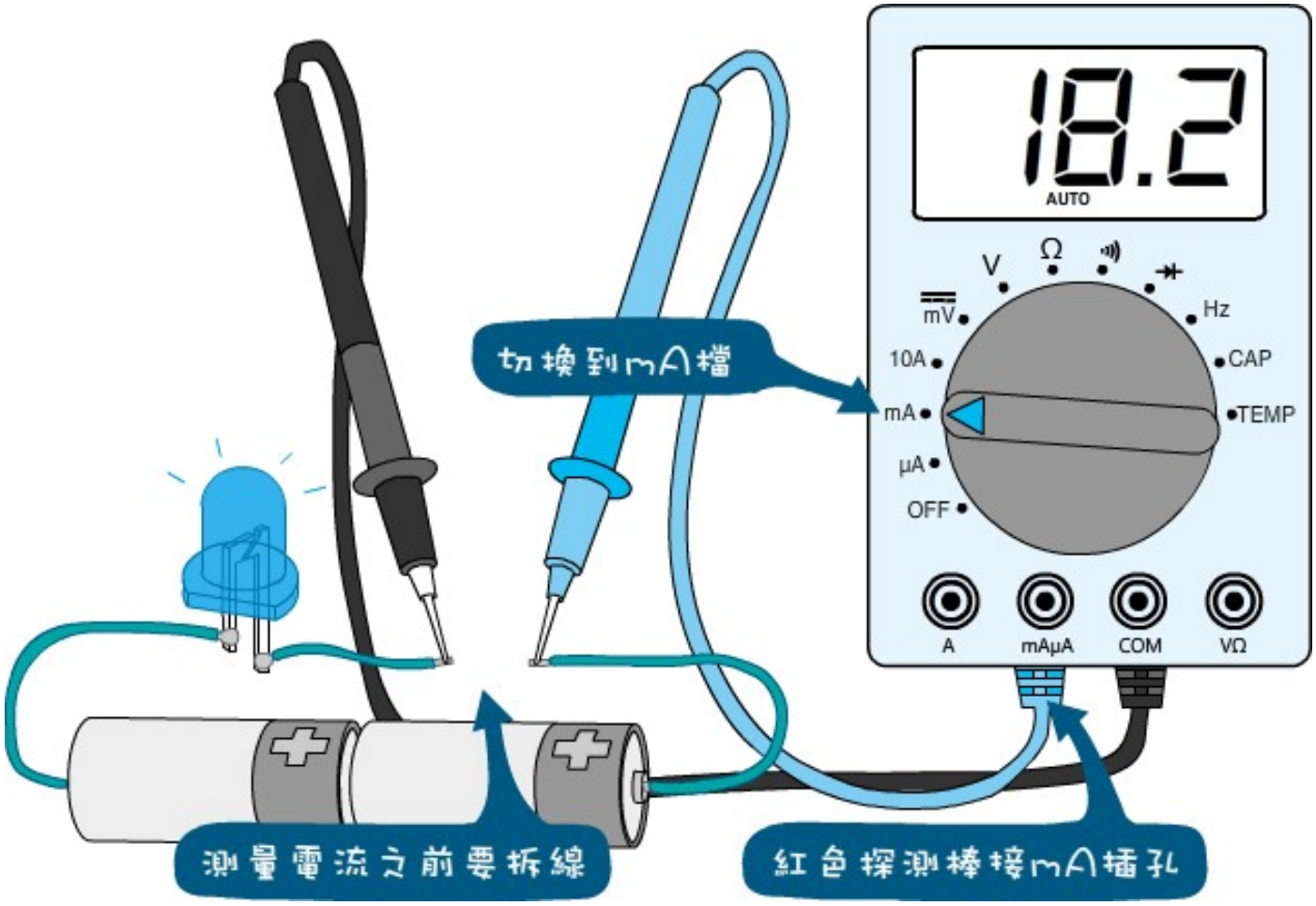
測量電壓

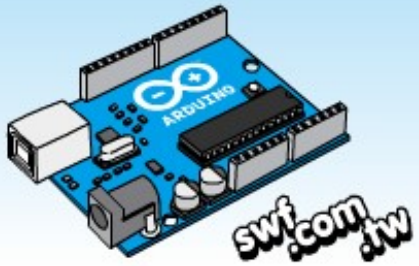
測量電壓或電流的過程中，不要切換檔位；切換檔位之前要先移開測試棒。



測量電流

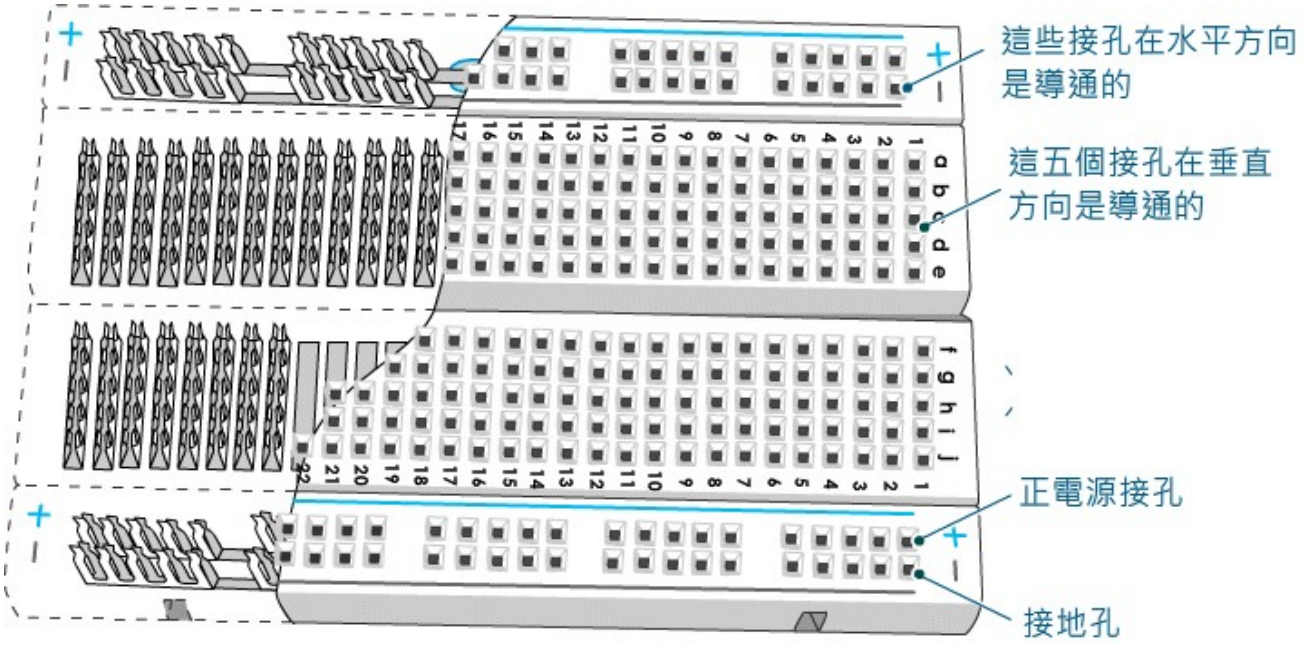
紅色測試棒記得要接在測量電流的插孔。

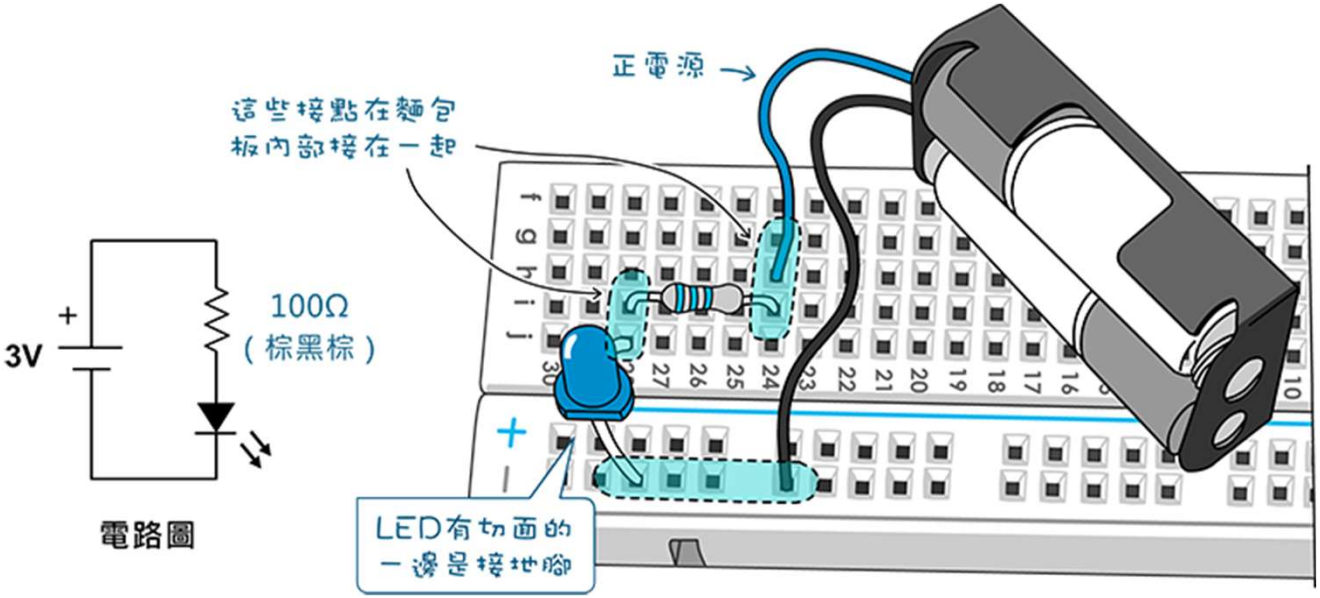




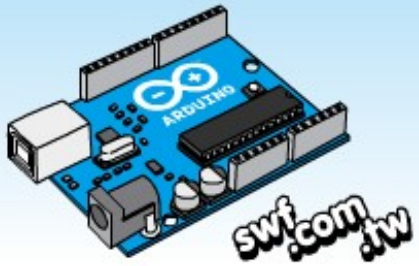
麵包板

麵包板是一種不需焊接，可快速拆裝、組合電子電路的用具，普遍用於電子電路實驗。





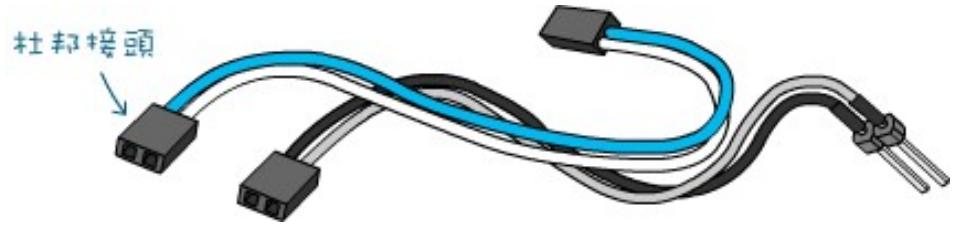
左邊是LED電路圖，以及在麵包板上組裝的樣子。



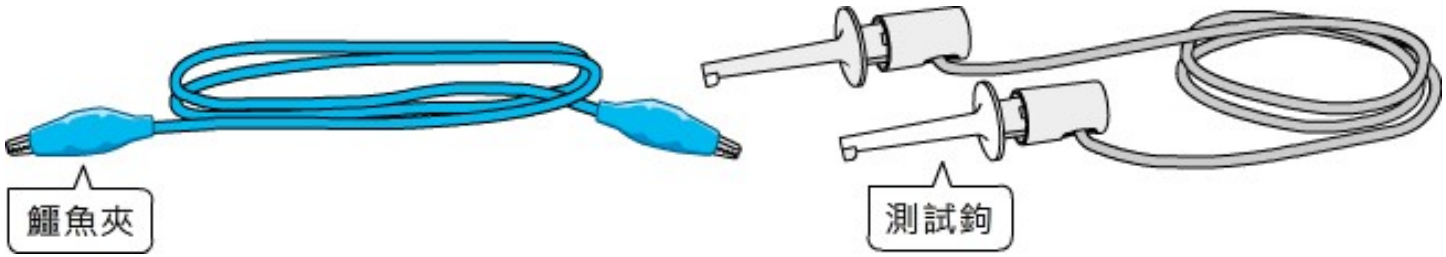
導線與跳線

麵包板的接線

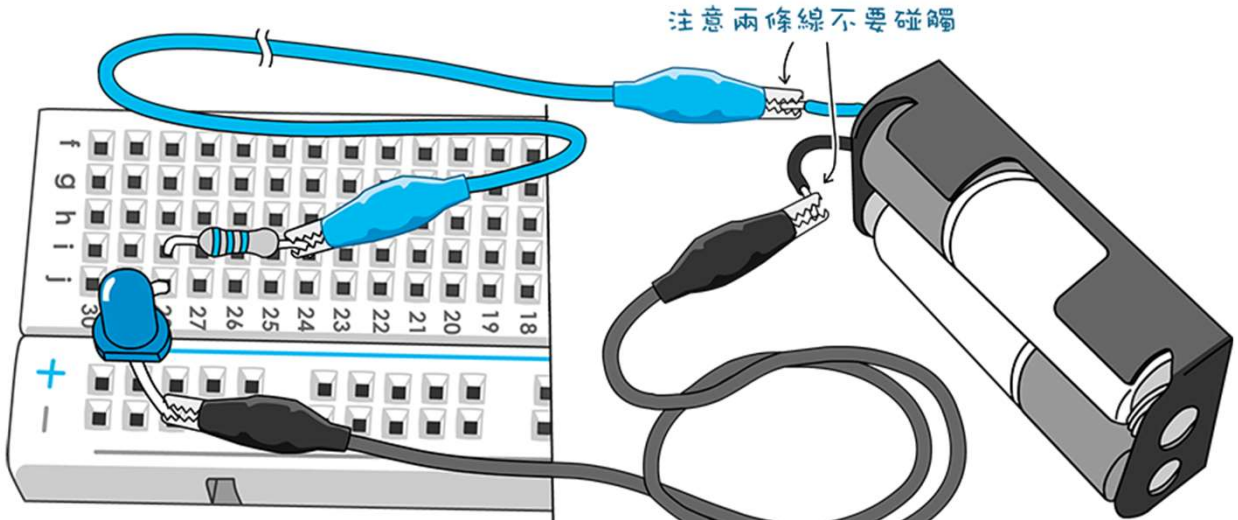
杜邦線

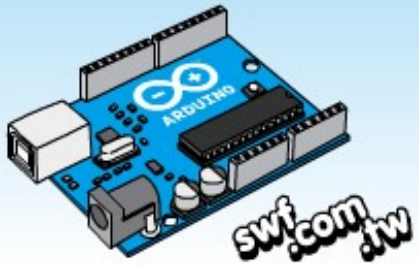


鱷魚夾和測試鉤



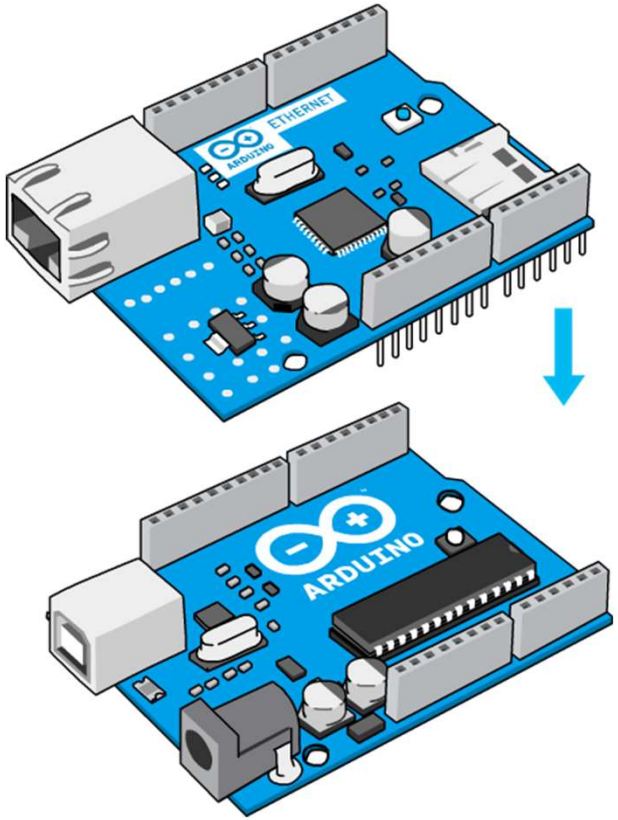
使用的例子



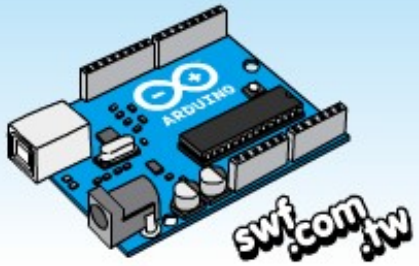


- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

第三章 Arduino互動程式設計入門

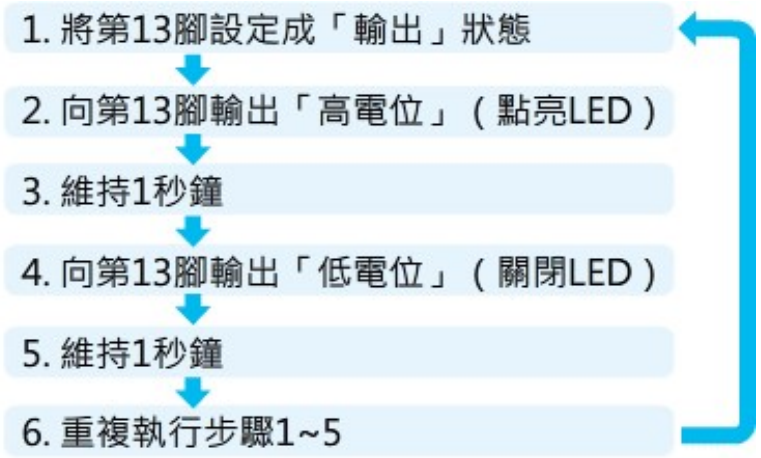


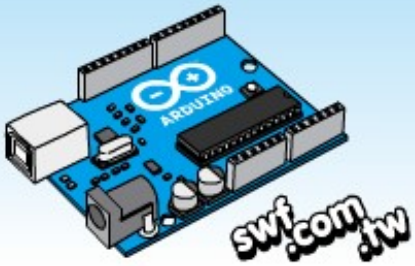
- Arduino程式基礎
- 認識變數
- 認識數字系統
- 認識常數



Arduino程式設計基礎

Arduino的程式要配合硬體的規劃，而且指令敘述要具體、明確，像「開始閃爍LED」這個指示，對電腦來說太抽象了。



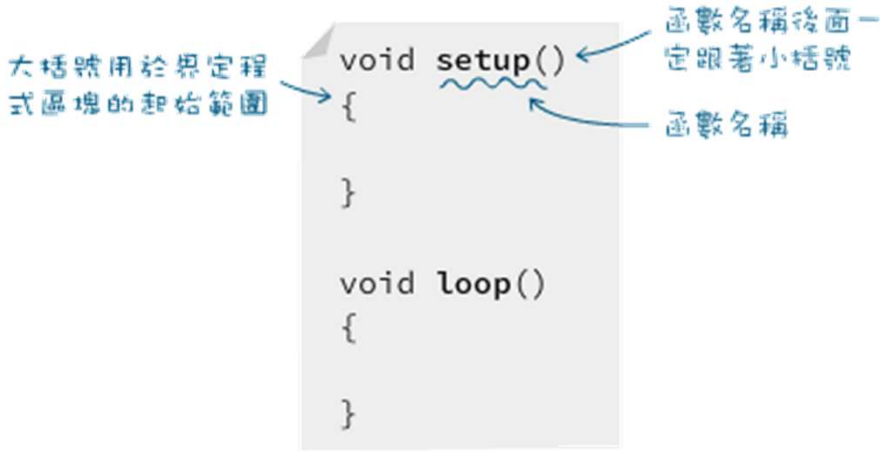


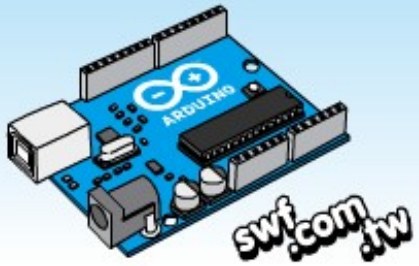
Arduino程式的基本架構

微處理器只認得0和1構成的**機械碼**（machine code）。

高階程式語言必須經過**編譯**（**compile**），才能交給微電腦執行。

所有Arduino程式都是由參數**設置**（**setup**）和**迴圈**（**loop**）兩大區塊所組成，這個「區塊」的正式名稱叫做**函數**（**function**）。





基本指令

設定接腳的工作模式：輸入（INPUT）或輸出（OUTPUT）

```
pinMode(接腳編號, 模式);
```

可能值為1~13或A0~A5

可能值為OUTPUT或INPUT

在指定接腳輸出數位訊號

```
digitalWrite(接腳編號, 輸出訊號);
```

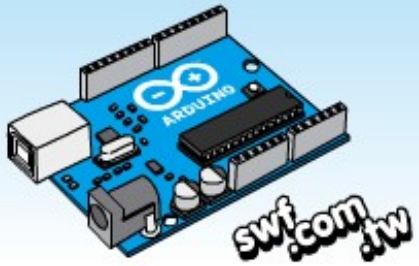
可能值為HIGH或LOW，或寫成數字 1 或 0。

在設定延遲時間

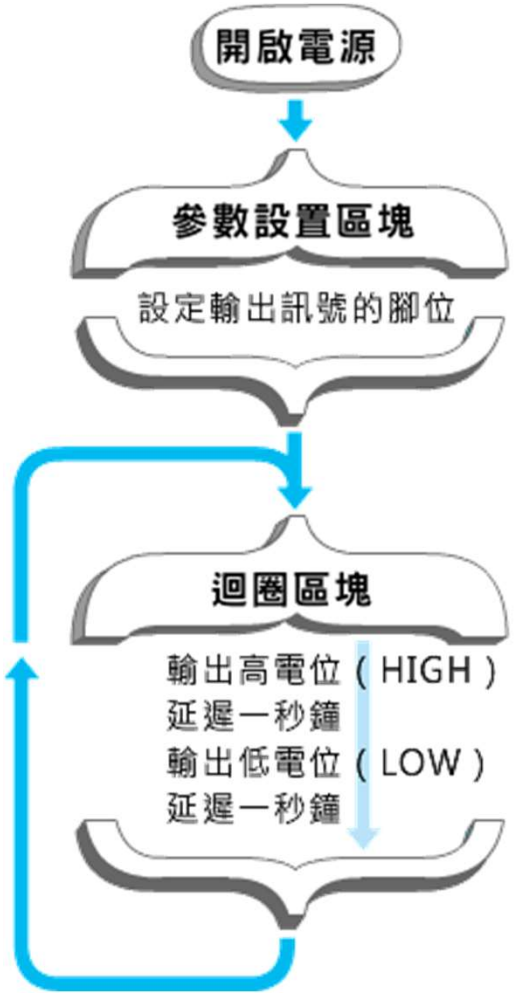
```
delay(延遲毫秒數);
```

ms

$\frac{1}{1000}$ 秒，亦即 10^{-3} 秒。

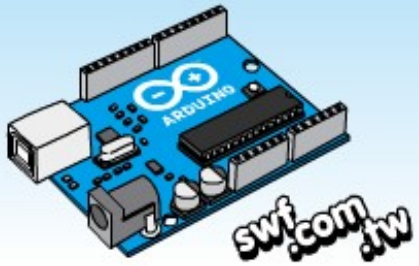


LED閃爍的程式碼




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

```
void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

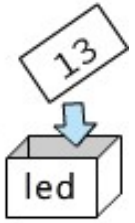


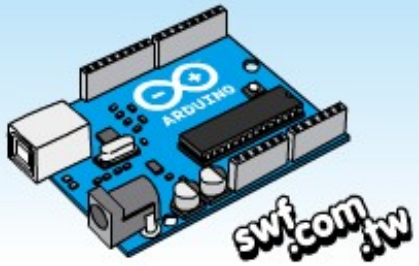
變數

在程式中，暫存資料的容器叫做變數。

變數有助於管理程式碼，像底下的程式碼，若要更改接腳，只需修改變數值：

```
byte led = 13;

void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```





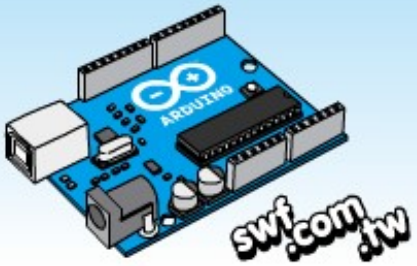
資料類型

資料類型用於設定「資料容器」的格式和容量。在宣告變數的同時，必須設定該變數所能儲存的資料類型。

也能寫成bool

類型	中文名稱	佔用記憶體大小	數值範圍
boolean	布林	8位元（1Byte）	1或0（true或false）
byte	位元組	8位元（1Byte）	0~255
char	字元	8位元（1Byte）	-128~127
int	整數	16位元（2Bytes）	-32768~32767
long	長整數	32位元（4Bytes）	-2147483648~2147483647
float	浮點數	32位元（4Bytes）	±3.4028235E+38
double	雙倍精確度浮點數	32位元（4Bytes）	±3.4028235E+38

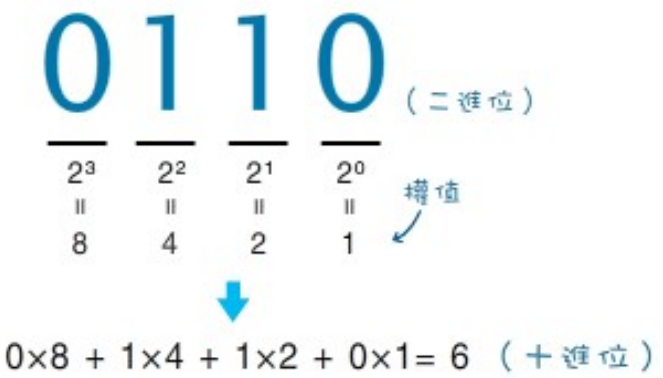
E是科學記號，E+38代表10^{38}



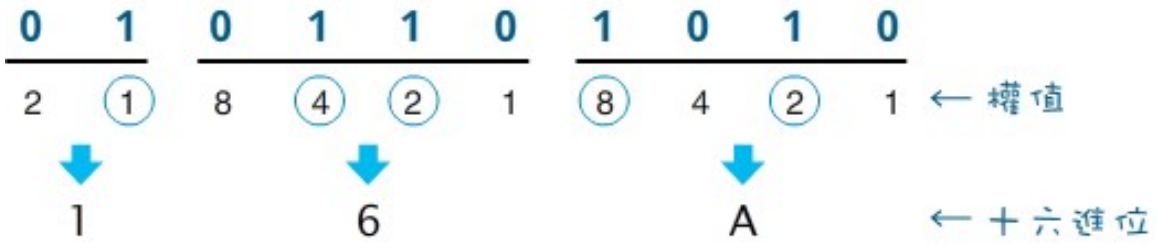
數字系統（一）

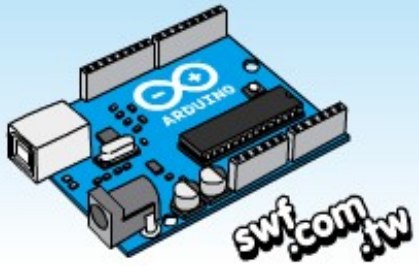
每個數字所在的位置，代表不同的權值（weight）。

二進位數字乘上它所代表的權值的總和，即可換算成10進位數字。



2進位數字換算成16進位數字





數字系統（二）

十進位	十六進位	二進位	十進位	十六進位	二進位
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

不同進位數字的表示方法

```
int a1 = 362;
int a2 = 0b101101010;
int a3 = 0x16A;
```

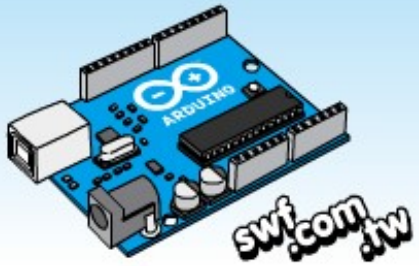
10進位值

「2進位值」也能用一個大寫B代表

B101101010

代表「2進位值」

代表「16進位值」



常數

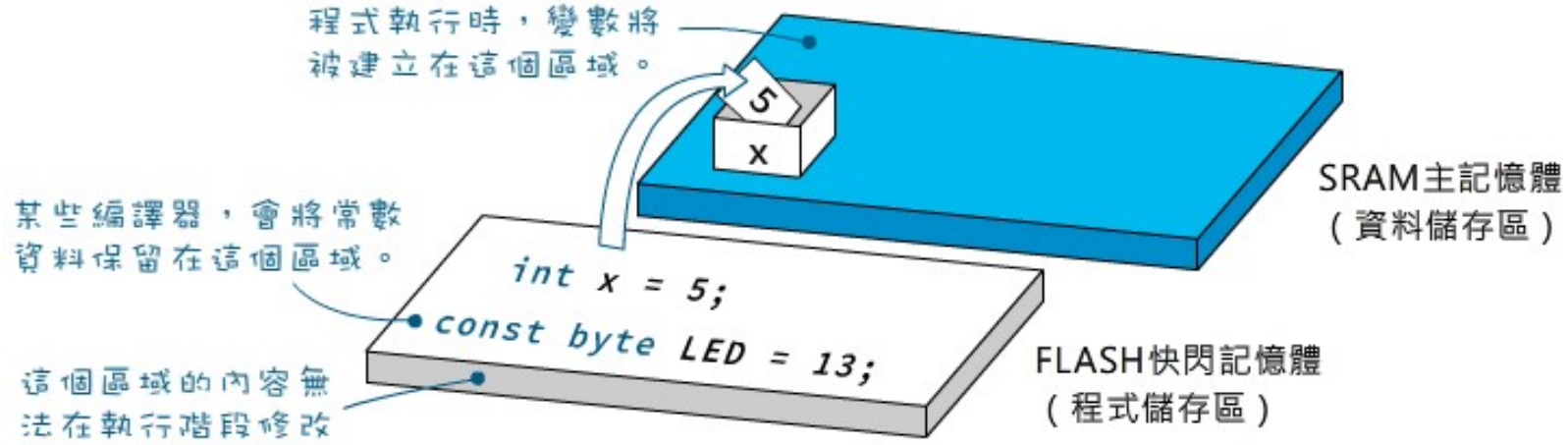
存放固定、不變數值的容器，稱為「常數」。

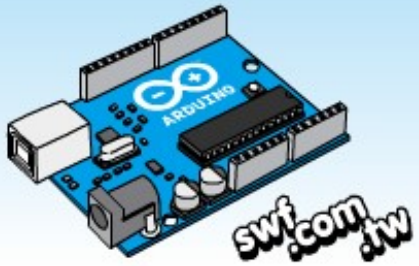
在類型名稱前面加上const

```
const float PI = 3.1415;
```

常數名稱通常全部寫成大寫

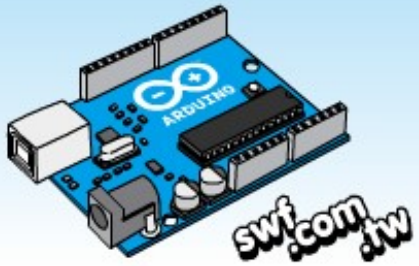
保存變數的容器，將在執行階段被建立在內容可隨意更換的SRAM（主記憶體）中。





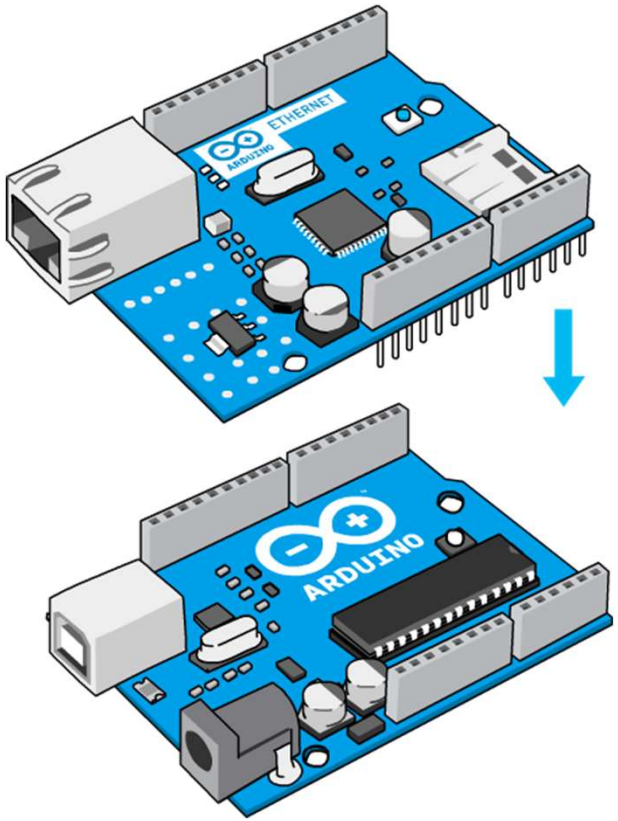
Com Port

- Serial.begin(9600);
- Serial.print(變數)；// 印出
- Serial.print(“文字”);
- Serial.println(); //跳行
- Serial.read(); // 輸入
- 試看看....

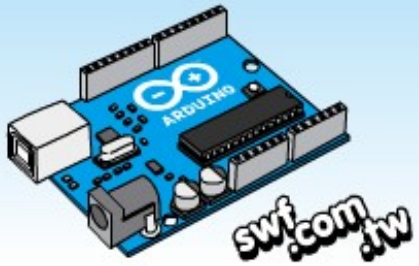


- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

第四章 開關電路與LED跑馬燈

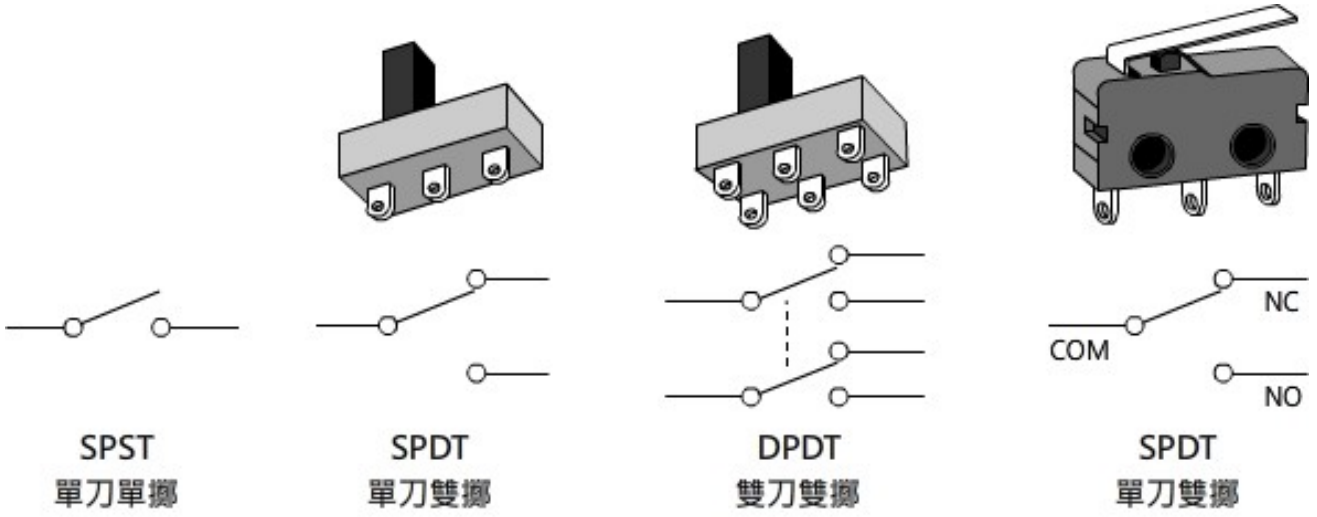


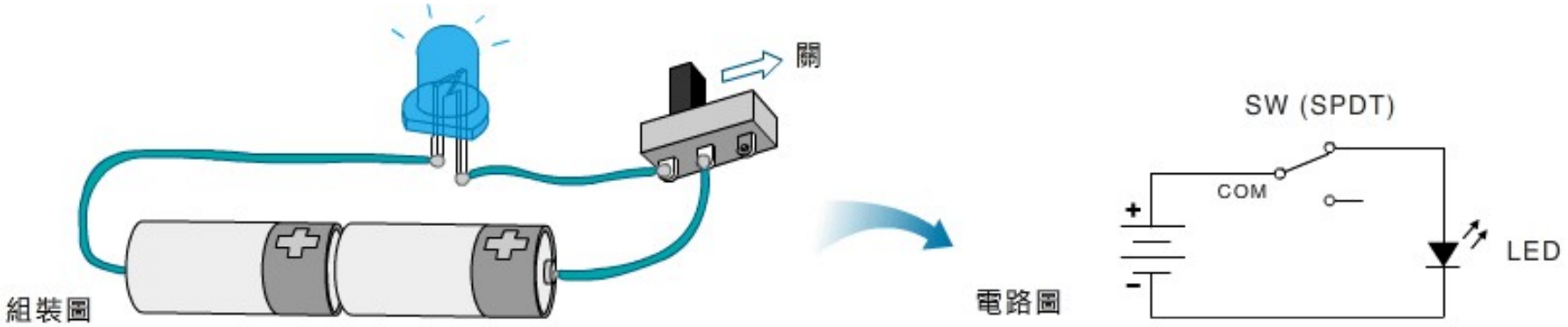
- 認識開關
- 讀取數位輸入值
- if條件式
- 撰寫迴圈程式
- 認識陣列
- 認識埠口與位移操作

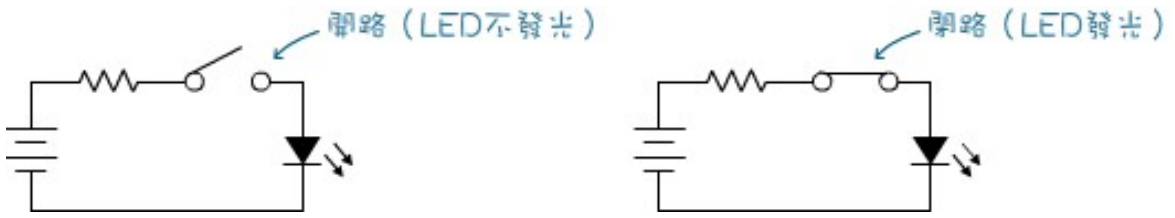


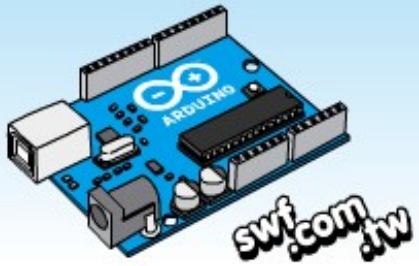
認識開關

開關的外觀與電路符號



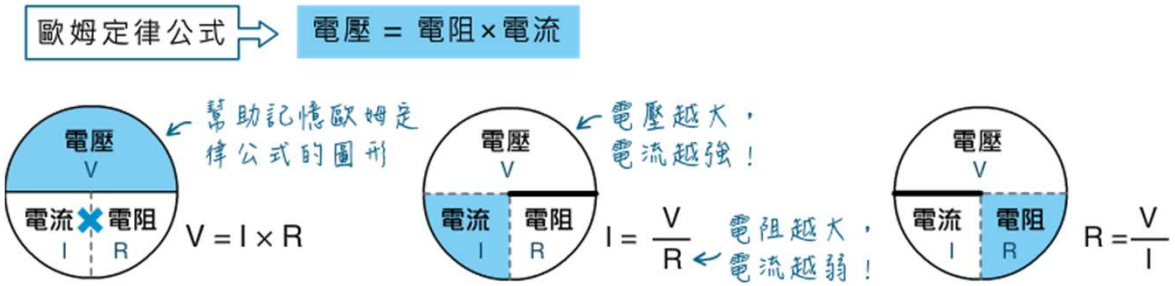




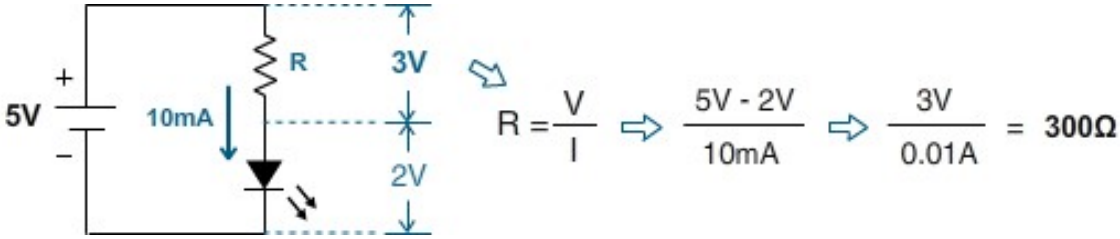


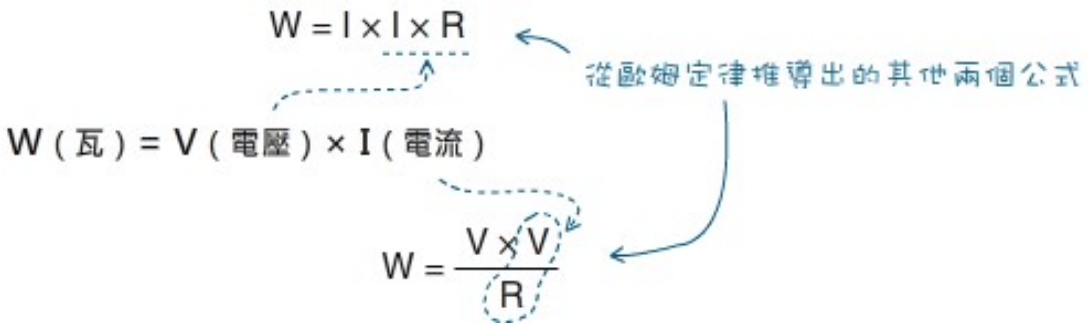
歐姆定律

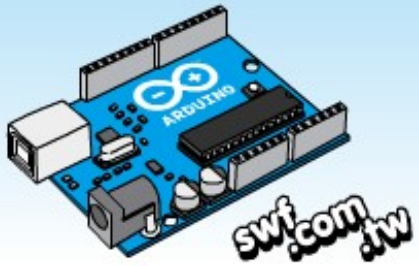
電路中的電壓、電流和電阻之間的關係，可以用歐姆定理表示：**電流和電壓成正比，和電阻成反比。**



透過歐姆定律，可求出電路所需的限流電阻值，以及消耗功率。





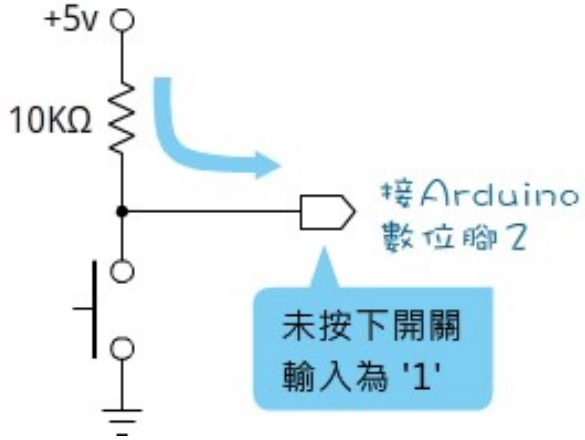


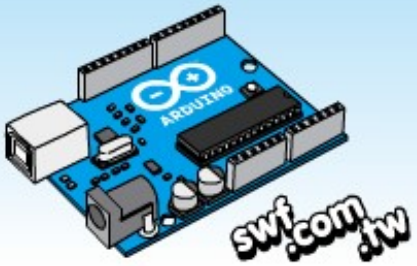
讀取數位輸入值

- 讀取數位輸入值的語法：

```
boolean 變數名稱 = digitalRead(接腳編號);
```

- 電路的接法：



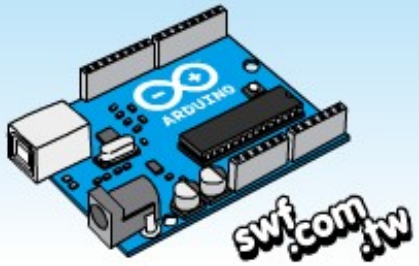


比較運算子

比較運算子	說明
==	如果兩者相等則成立
!=	如果不相等則成立
<	如果左邊小於右邊則成立
>	如果左邊大於右邊則成立
<=	如果左邊小於或等於右邊則成立
>=	如果左邊大於或等於右邊則成立

條件式當中的且、或和反相測試：

名稱	運算符號	運算式	說明
且（AND）	&&	A && B	只有A和B兩個值都成立時，整個條件才算成立。
或（OR）	||	A || B	只要A或B任何一方成立，整個條件就算成立。
反相（NOT）	!	!A	把成立的變為不成立；不成立的變為成立。

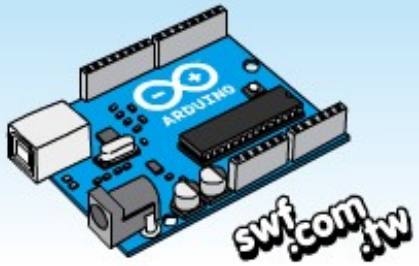


指定運算子

運算子	意義	說明
++	遞增	變數加1
--	遞減	變數減1
+=	指定增加	
-=	指定減少	
*=	指定相乘	
/=	指定相除	

以下三種敘述，意義相同：

```
a = a+1;
a ++;
a += 1;
```

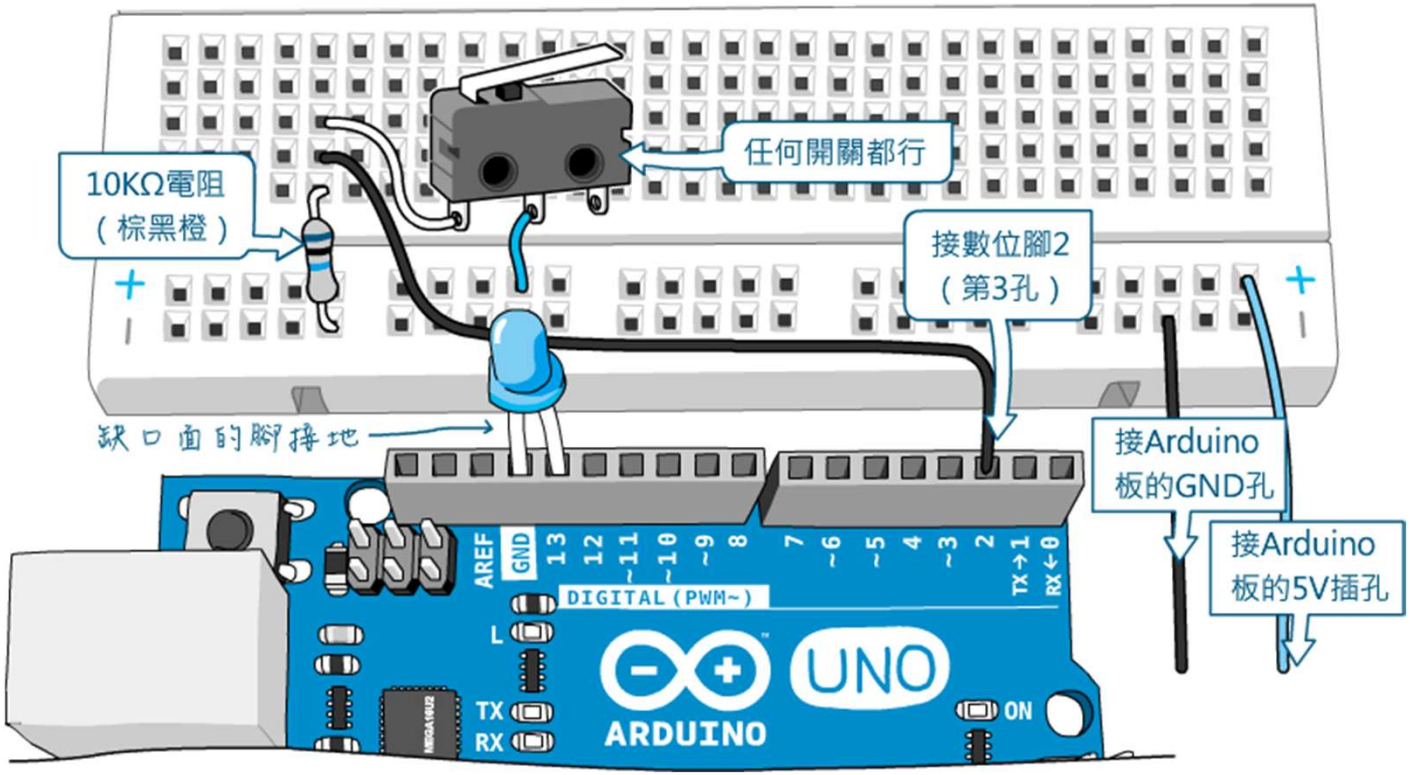


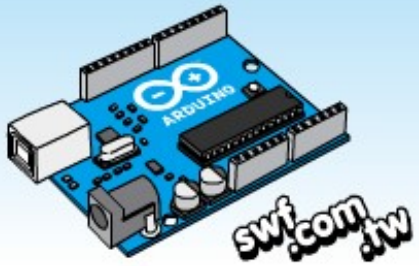
數學運算子

運算子	意義	說明
+	加	
-	減	
*	相乘	
/	相除	得到商值
%	餘數	得到餘數

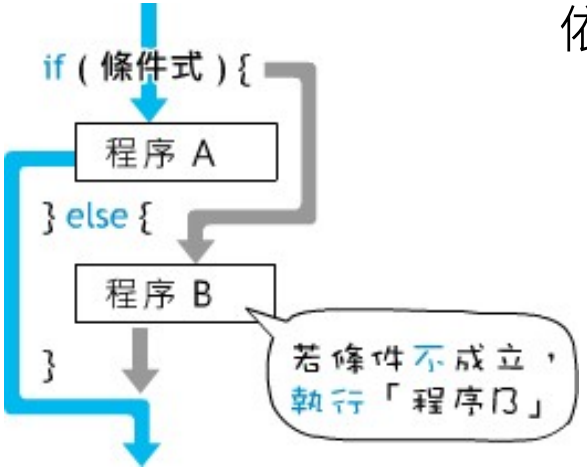
7 / 5　>>>>得到 1，取整數
7 % 5　>>>>得到 2

開關實驗麵包板電路

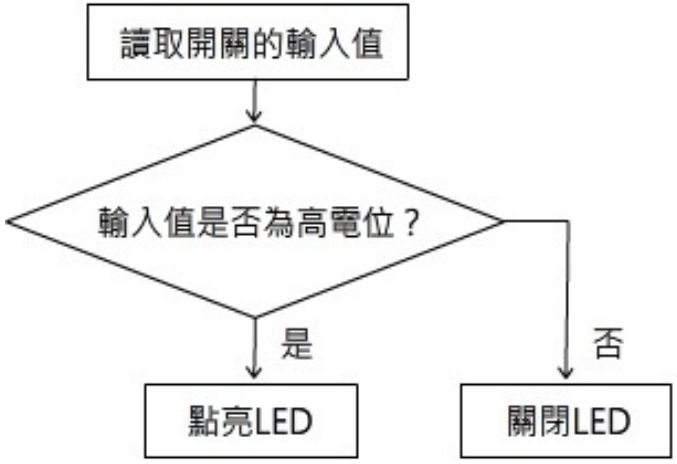

10KΩ電阻
（棕黑橙）
任何開關都行
接數位腳2
（第3孔）
缺口面的腳接地
接Arduino
板的GND孔
接Arduino
板的5V插孔
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON



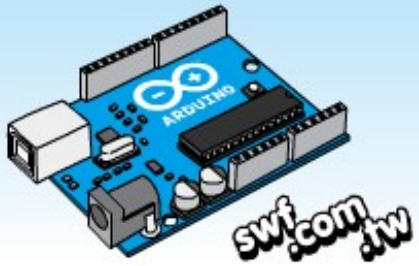
條件判斷式敘述



依據開關的狀態，點亮或關閉LED。

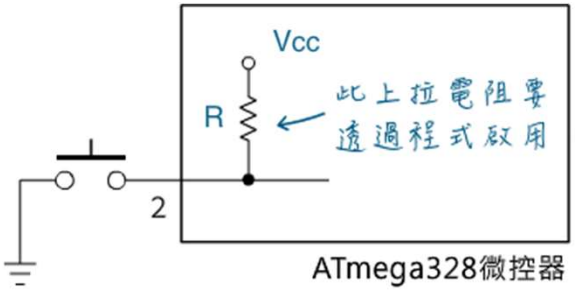



```
const byte LED = 13;
const byte SW = 2;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(SW, INPUT);
}

void loop(){

   boolean val = digitalRead(2);

   if (val){
     digitalWrite(LED, HIGH);
   } else {
     digitalWrite(LED, LOW);
   }
}
```

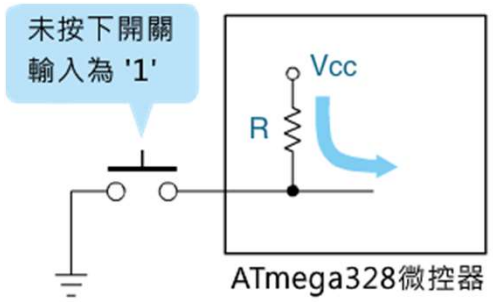


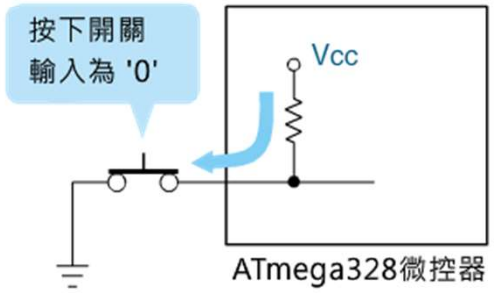
啟用上拉電阻

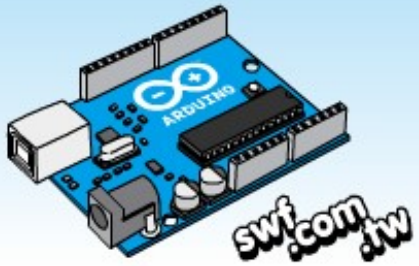
ATmeg328微控器的數位接腳有內建上拉電阻，可透過底下的敘述啟用，啟用內建的上拉電阻後，開關電路就能省略外接電阻。




```
/* 啟用第2腳內部的「上拉電阻」 */
pinMode(2, INPUT_PULLUP);
```

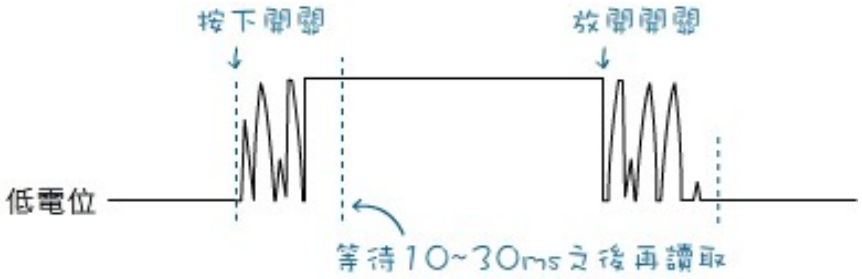




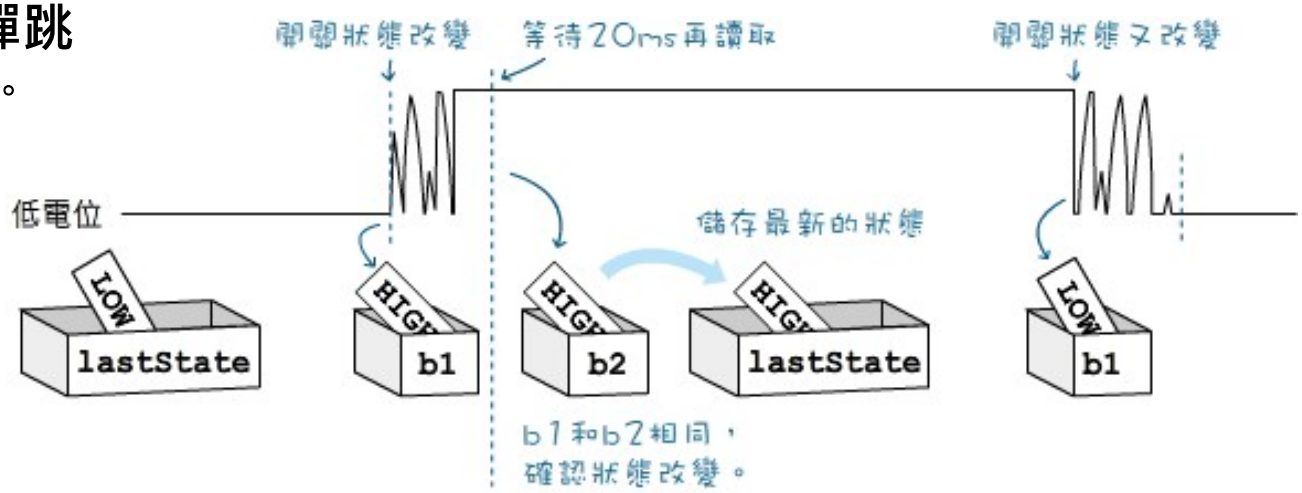


消除開關的彈跳訊號

機械式開關在切換的過程中，電子信號並非立即從0變成1（或從1變成0），而會經過短暫的，忽高忽低變化的彈跳現象。

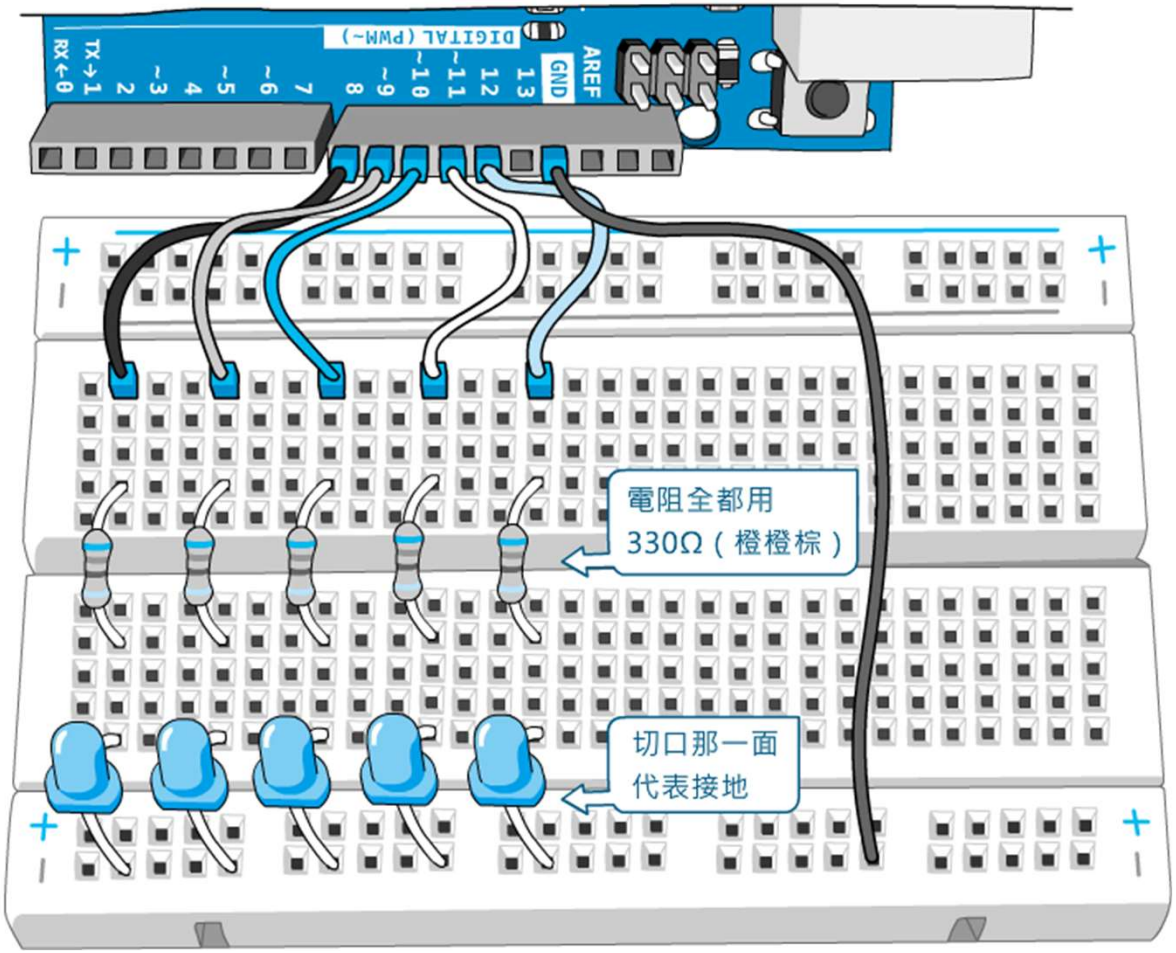


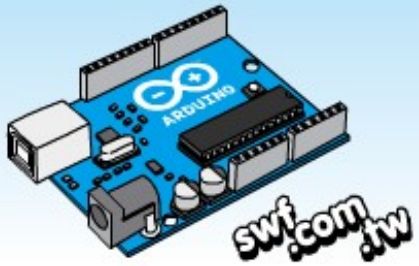
為了避免上述狀況，程式（或者硬體）需要加入所謂的**消除彈跳（de-bouncing）**處理機制。



LED跑馬燈電路

輪流點亮一個LED



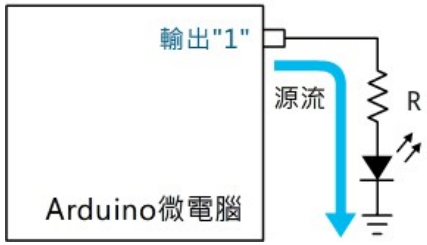


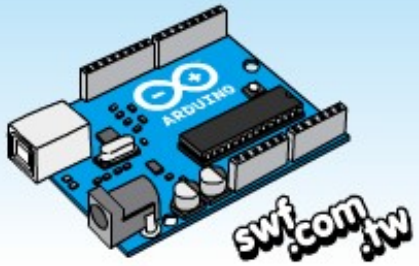
LED跑馬燈程式一

- 輪流點亮一個LED的程式碼

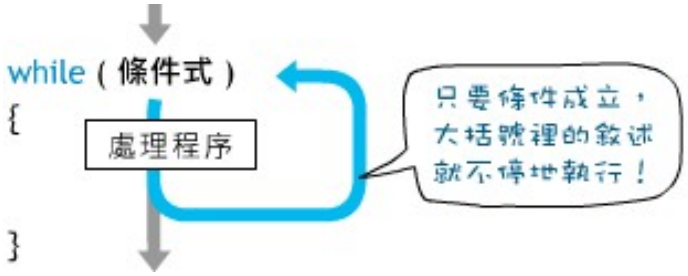
```
void loop() {
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  delay(100);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED3, LOW);
  delay(100);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, HIGH);
  delay(100);
}
```

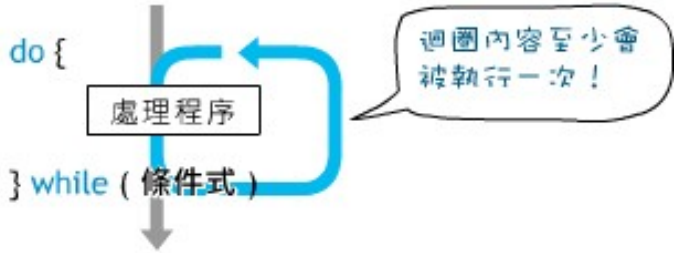
點亮第一個LED



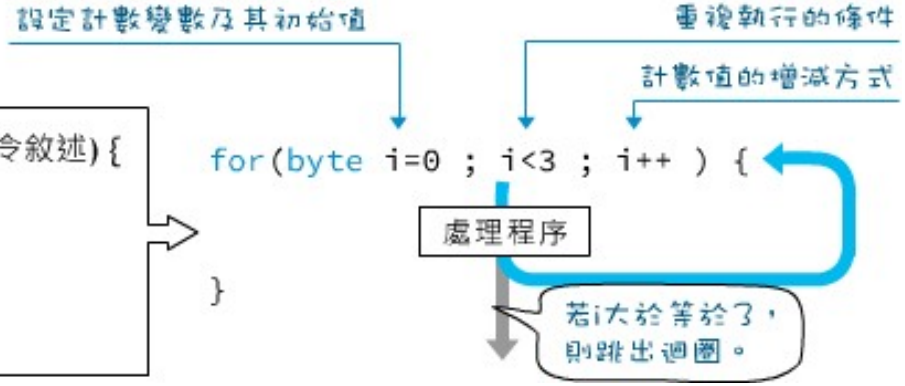


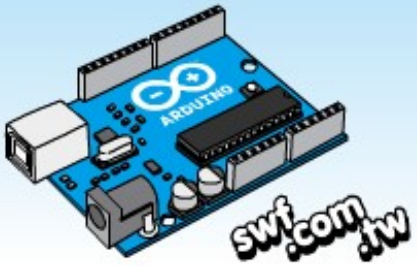
迴圈控制指令






```
for (初始變數；條件式；指令敘述){
  指令敘述1
  指令敘述2
    :
}
```



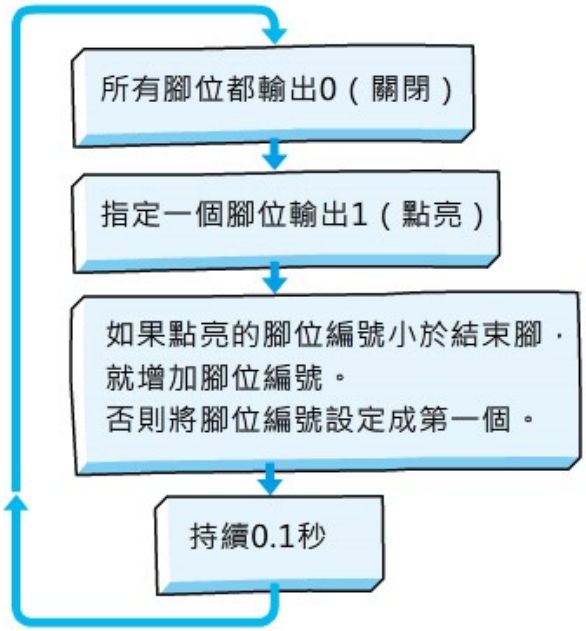


使用for迴圈的跑馬燈程式

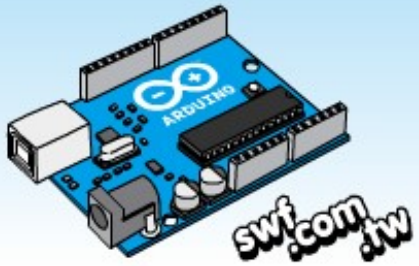
```
void setup() {
  for (byte i = startPin; i <= endPin; i++) {
    pinMode(i, OUTPUT);
  }
}
```

起始腳位 ↓ startPin　　結束腳位 ↓ endPin

主程式迴圈




```
void loop() {

  for (byte i=startPin; i<=endPin; i++) {
    digitalWrite(i, LOW);
  }

  digitalWrite(lightPin, HIGH);

  if (lightPin < endPin) {
    lightPin ++;
  } else {
    lightPin = startPin;
  }

  delay(100);
}
```

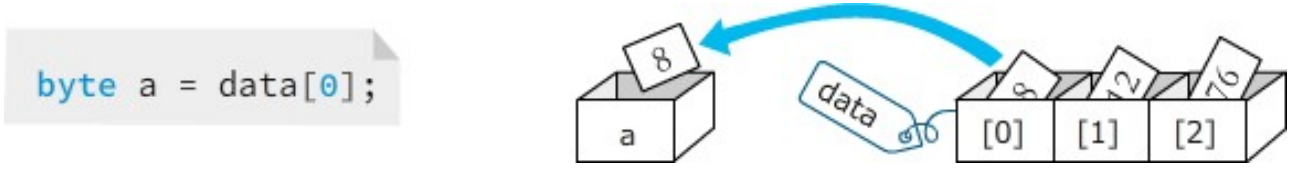


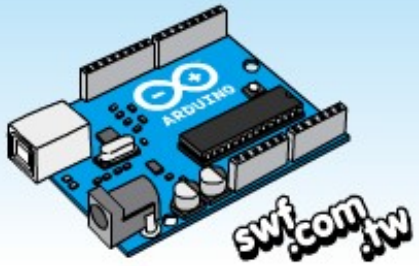
陣列變數

陣列（array）變數可以存放很多不同值，就像具有不同分隔空間的盒子一樣。

宣告陣列的同時可一併設定其值

讀取陣列元素



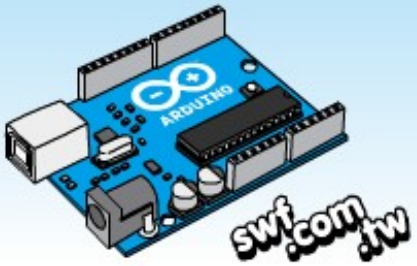


使用陣列的跑馬燈程式

```
const byte LEDs[] = {8,9,10,11,12};
const byte total = sizeof(LEDs);
byte index = 0;

void setup() {
  for (byte i=0; i<total; i++) {
    pinMode(LEDs[i], OUTPUT);
  }
}

void loop() {
  for (byte i=0; i<total; i++) {
    digitalWrite(LEDs[i], LOW);
  }
  digitalWrite(LEDs[index], HIGH);
  if (index < total) {
    index ++;
  } else {
    index = 0;
  }
  delay(100);
}
```

i值將是0~4，依序讀取出8, 9, 10, 11和12元素值。

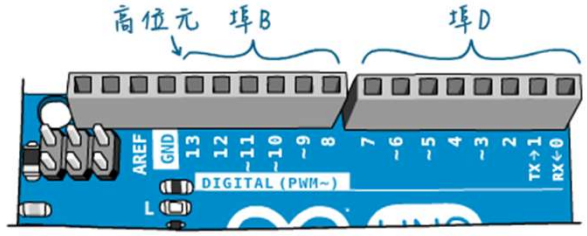
在指定腳位輸出「高電位」



埠口操作與位移指令

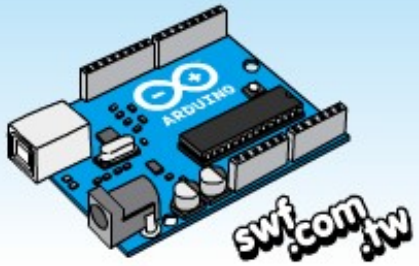
ATMega328處理器，具有D, B兩個數位輸出/輸入埠，以及一個類比埠C。

埠口指令可用一行敘述，將多個腳位設定成輸出或輸入。



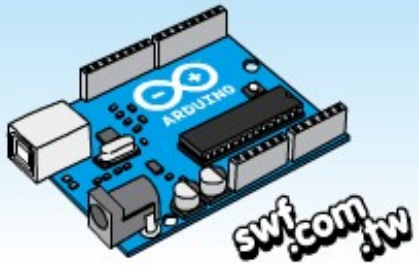
位移（shift）運算子可以將資料裡的所有位元向右或向左移動，空缺的部分補上0。

位移運算	位移結果（二進位）	結果（十進位）	
1	00000001	1	
1 << 1	00000010	2	相當於乘2
1 << 2	00000100	4	相當於乘4
12 >> 0	00001100	12	
12 >> 1	00000110	6	相當於除2
12 >> 2	00000011	3	相當於除4
12 >> 3	00000001	1	



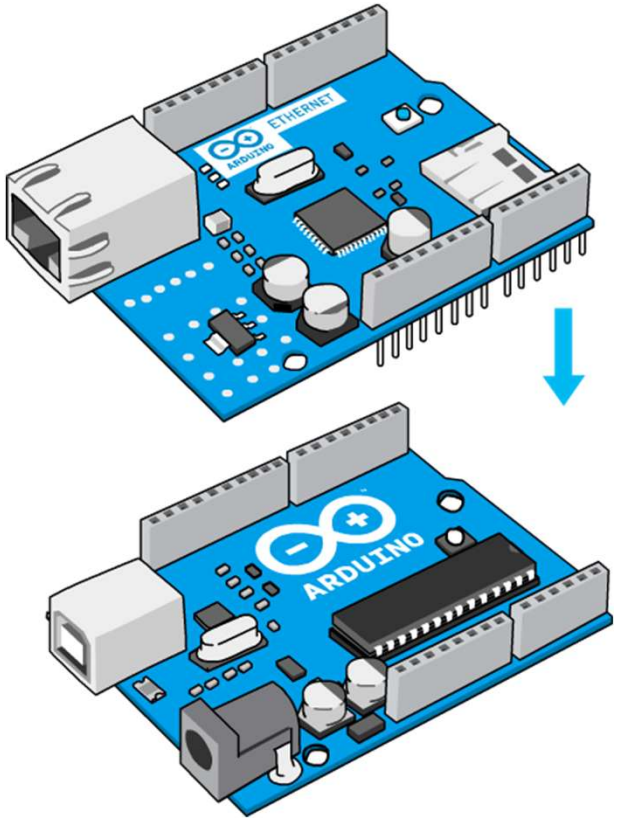
使用埠口與位移製作跑馬燈

```
byte data = B00001;
byte shift = 0;
byte max = 5;

void setup(){
 DDRB = B011111;   // 8~12腳設成輸出。
}

void loop() {
  PORTB = data << shift;   ← 資料往左移之後，輸出到埠口。
  shift ++;
  if (shift == max) {      ← 如果位移等於最大值，就重設為0。
    shift = 0;
  }
  delay(100);
}
```

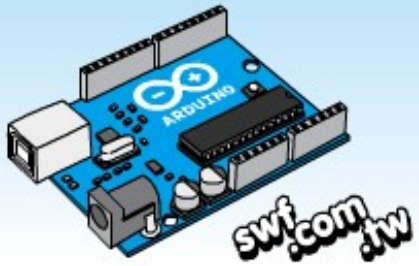


- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

第五章 序列埠通訊



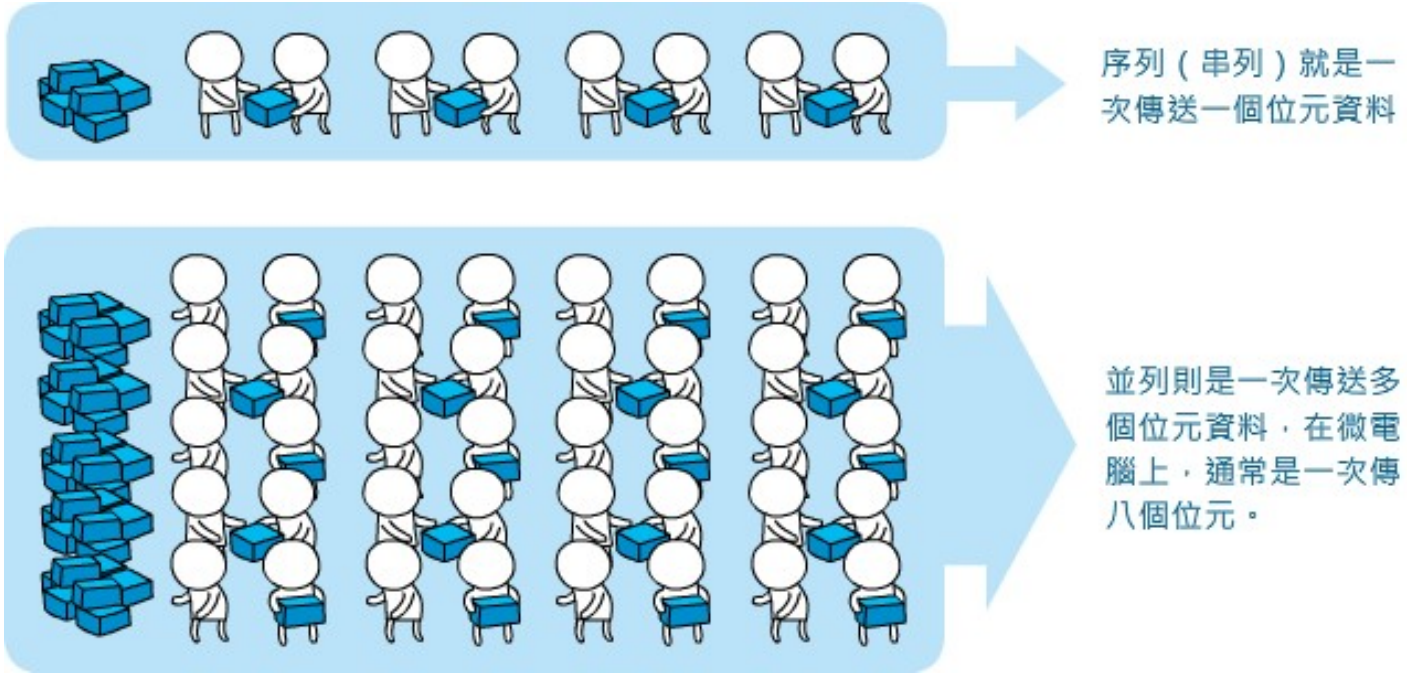
- 並列與序列通訊簡介
- 認識序列埠
- 認識字元資料類型
- 字串資料類型
- 從序列埠監控視窗觀察變數
- 從Arduino接收序列資料
- switch...case控制結構

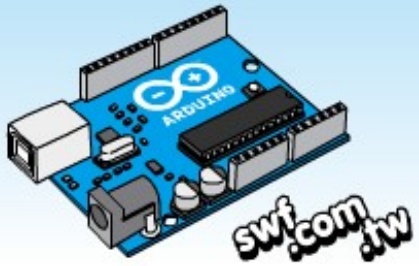


並列與序列通訊簡介

並列代表處理器和周邊之間，有多條資料線連結，
處理器能一口氣輸出或接收多個位元的資料。

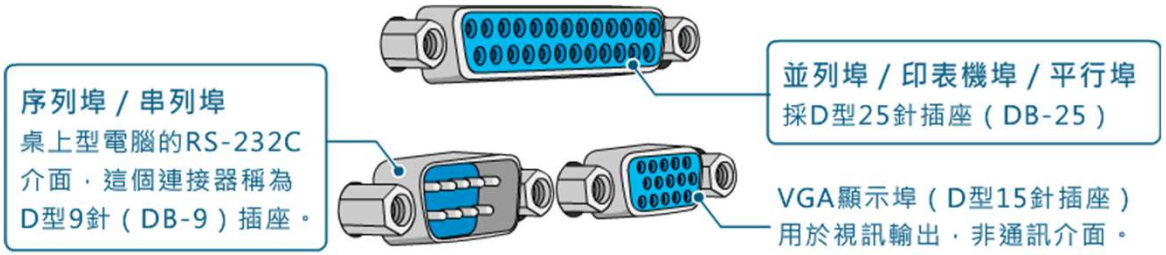
序列則是用少數（通常是兩條或三條）資料線，
將整批資料依序一個個送出或傳入。



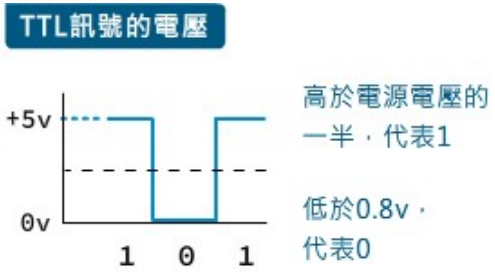


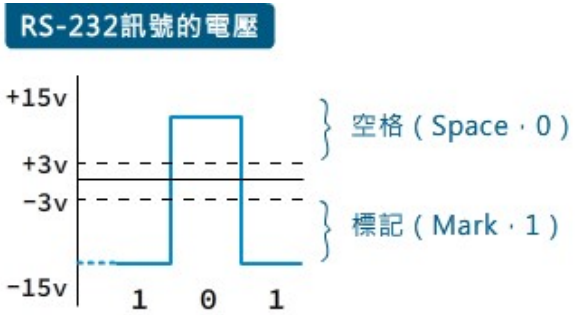
認識序列埠

RS-232是最早廣泛使用的序列埠標準，在系統軟體中稱為COM，每個COM介面同時只能接一個裝置。



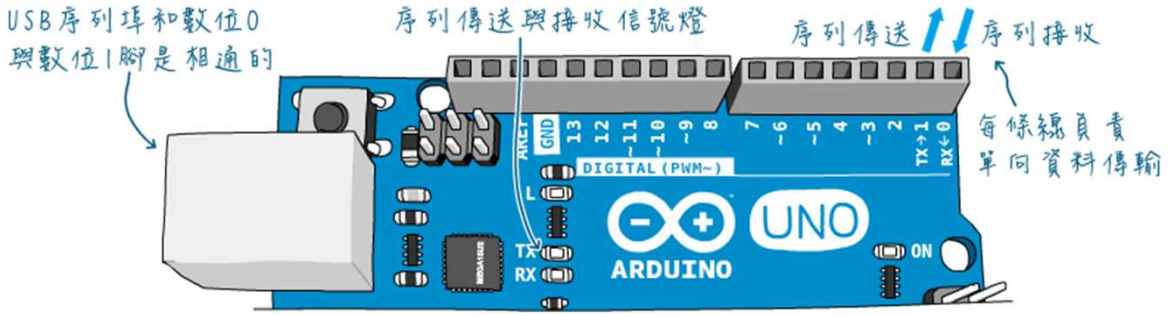
一般數位IC的0與1訊號的電壓準位，分別是0v和5v，稱為TTL或邏輯準位。

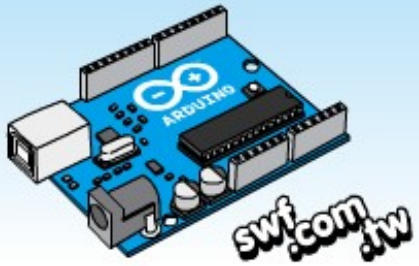




序列埠最重要的三個接腳：

- 數據傳送（TxD）
- 數據接收（RxD）
- 接地（GND）





認識字元資料類型

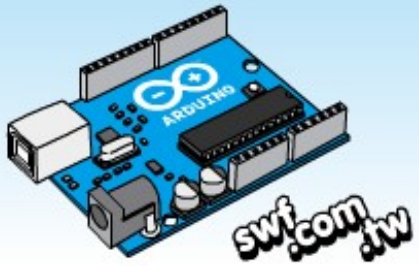
電腦把文字訊息分成字元（character）和字串（string）兩種資料類型。
字元類型的資料值要用單引號（'）刮起來。
每個字元都用一個唯一的數字碼表示。

```
char data = 'A';
```
← 字元要單引號刮起來，不能用雙引號，而且只能存放一個字。

```
char data = 65;
```
← 以數字編碼格式儲存「字元」時，不用單引號！

ASCII定義了128個字元，其中有33個是不能顯示的控制字元。

控制字元	ASCII編碼（10進位）	程式寫法	說明
NULL	0	\0	代表「沒有資料」或字串的結尾。
CR (Carriage Return)	13	\r	換行
LF (Line Feed)，也稱為New Line	10	\n	新行



字串資料類型

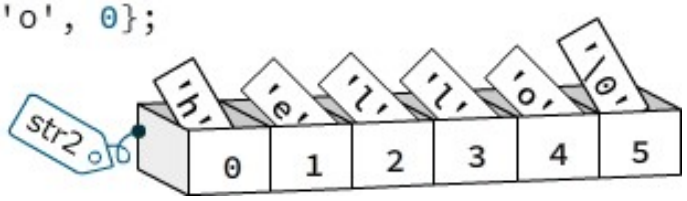
字串是一連串字元（char）的集合，也就是一段文字。
程式採用陣列存放字串，資料值前後一定要用雙引號刮起來。
每個字串都有一個Null字元（ASCII值為0）結尾。

加上Null結尾

```
char str1[] = {'h','e','l','l','o','\0'};
```

明確指定元素數量　　Null值也能用ASCII編碼數字設定

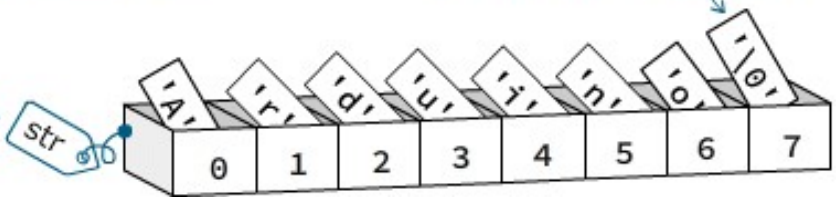
```
char str2[6] = {'h','e','l','l','o', 0};
```

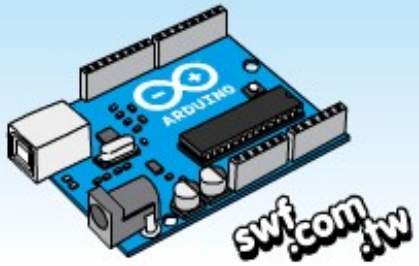


用雙引號刮起來

字串會自動加上一個Null字元結尾（ASCII值為0）

```
char str[] = "Arduino";
```

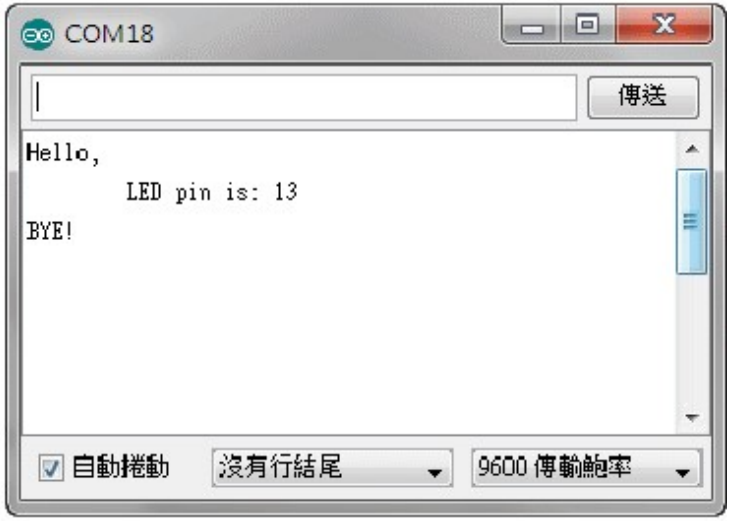


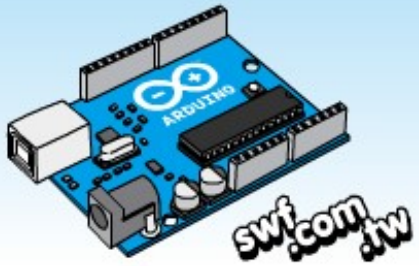


從序列埠監控視窗觀察變數

Arduino內建處理序列埠連線的Serial程式庫
建立序列埠連線的首要任務是設定資料傳輸率，底下的程式設定為9600bps。
序列埠監控視窗的連線速率要和Arduino程式一致。

```
const byte ledPin = 13;

void setup() {
    初始化序列埠，以9600bps速率連線。
   Serial.begin(9600);
   Serial.println("Hello," );
   Serial.print("\tLED pin is: ");
   Serial.print(ledPin);
   Serial.print("\nBYE!");

}

void loop() {
}
```

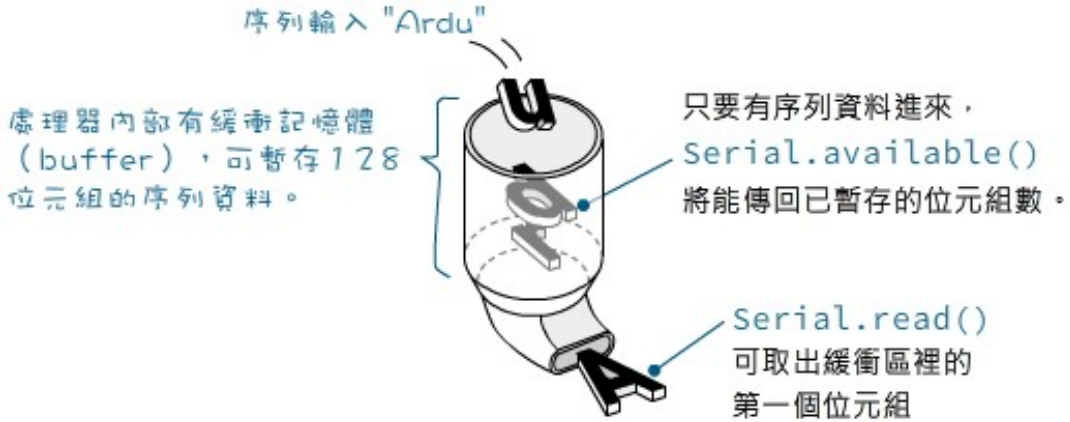




從Arduino接收序列資料

微處理器內部有一個類似儲存槽的緩衝記憶區（buffer），用於暫存來自序列埠的輸入資料。

如果緩衝記憶區裡面沒有資料，Serial.available()將傳回0。

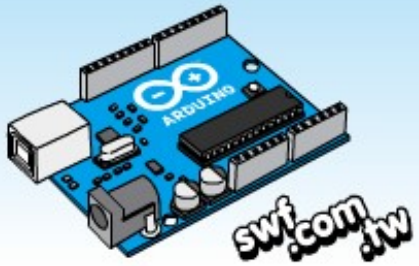



```
void loop() {
  if( Serial.available() ) {
    val = Serial.read();
    if (val == '1') {
      digitalWrite(LED, HIGH);
      Serial.print("LED ON");
    } else if (val == '0') {
      digitalWrite(LED, LOW);
      Serial.print("LED OFF");
    }
  }
}
```

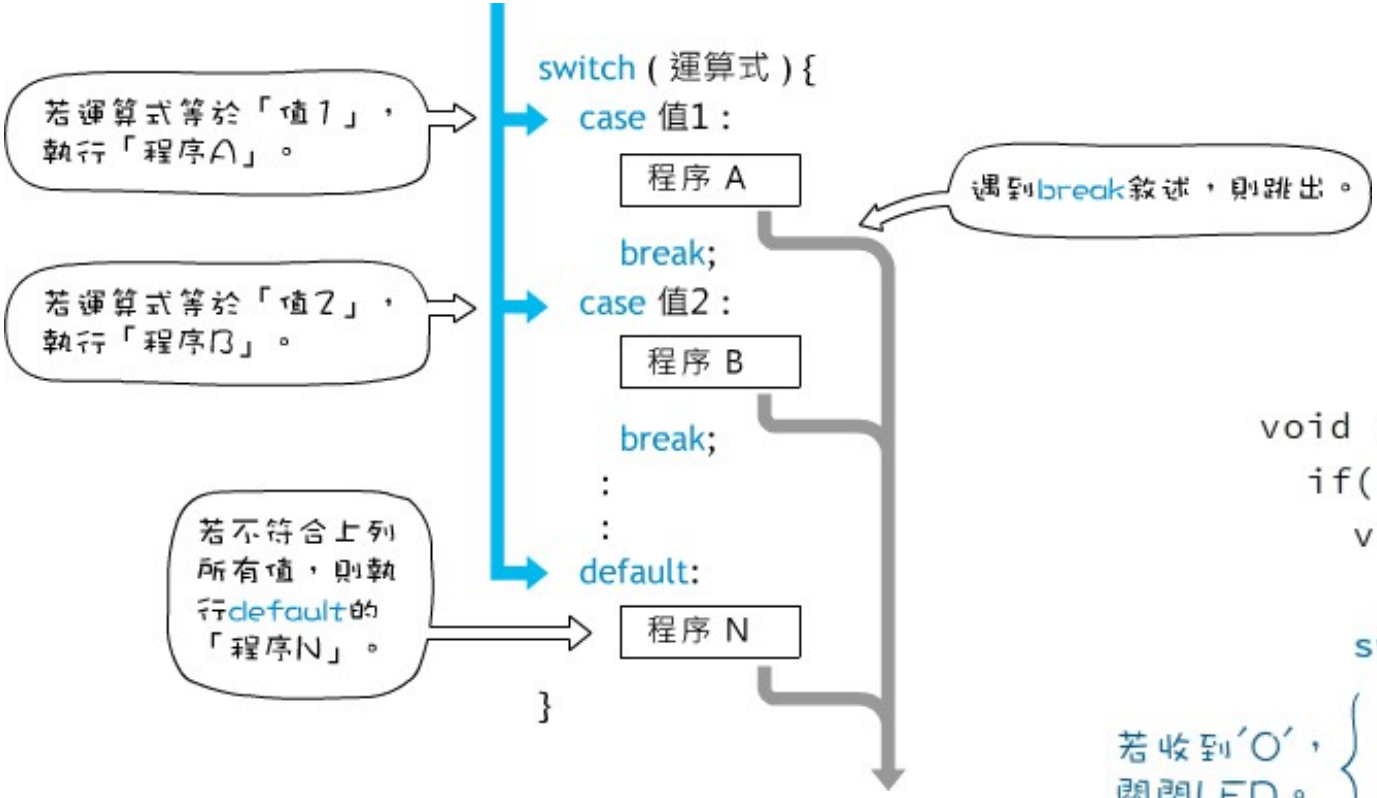
只要有收到字元，條件式的內容將被執行。

若收到'1'，點亮LED。

若收到'0'，關閉LED。



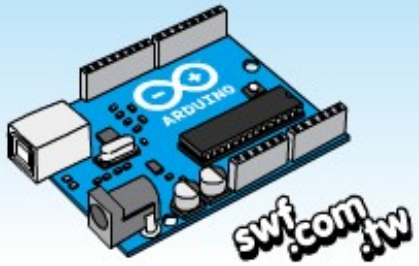
switch...case控制結構




```
void loop() {
  if( Serial.available() ) {
    val = Serial.read();

    switch (val) {
      case '0':
        digitalWrite(LED, LOW);
        break;
      case '1':
        digitalWrite(LED, HIGH);
        break;
    }
  }
}
```

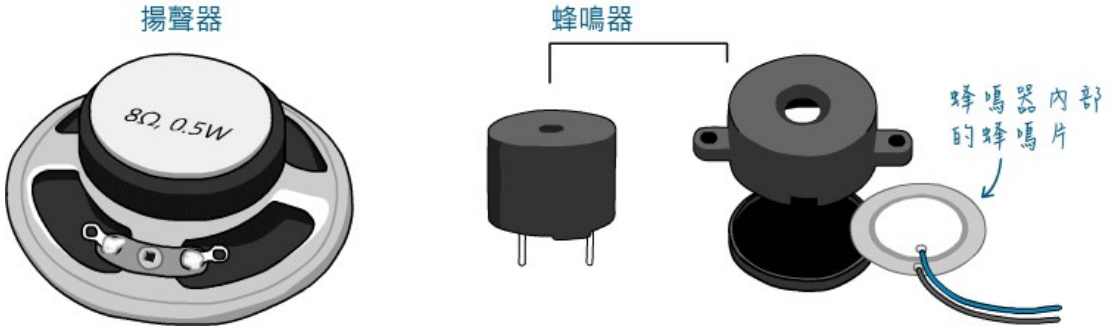
若收到'0'，關閉LED。

switch具有「切換」的涵意：透過比對switch()裡的變數和case後面的值，來決定切換執行哪一段程式。

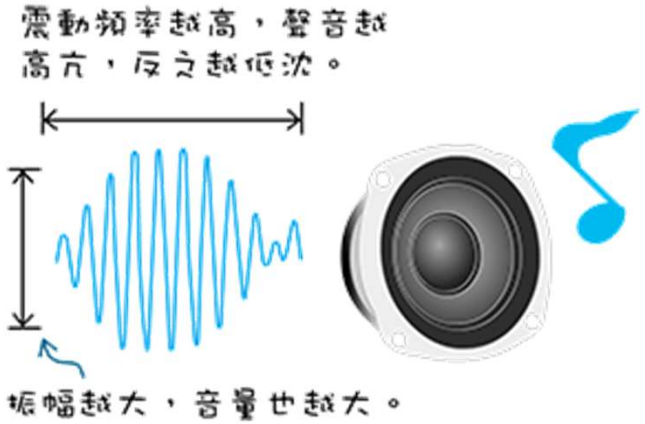


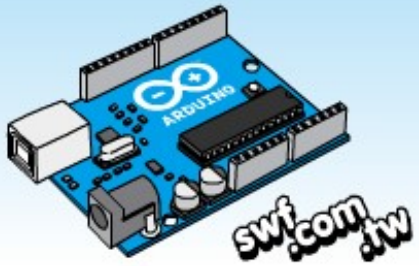
發音體和聲音

電子裝置常見的發音體為揚聲器（喇叭，speaker）和蜂鳴器(buzzle)
或是壓電轉換器（piezo transducer）



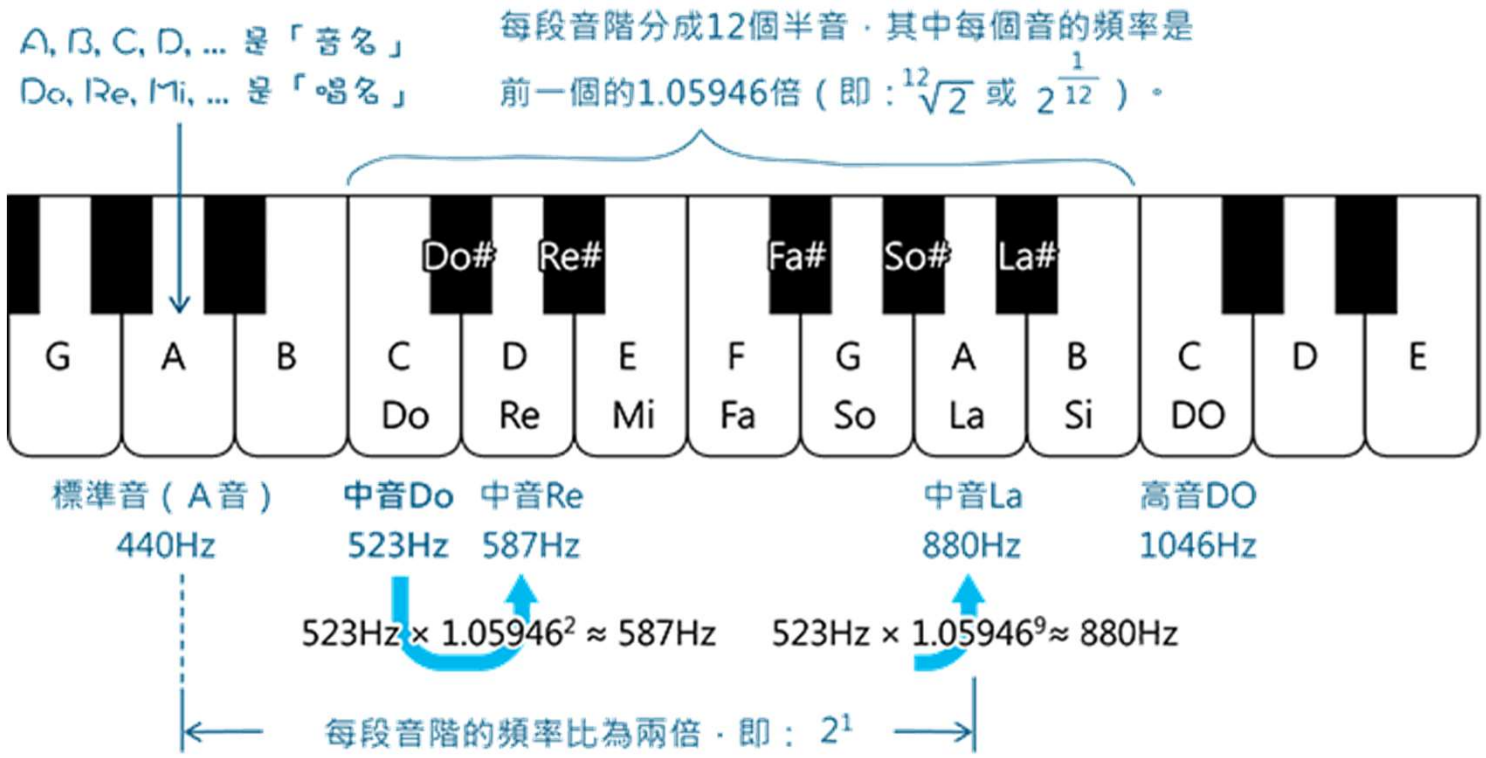
聲音是由震動產生, 其震動的頻率稱為**音頻**，音頻的範圍介於20Hz~200KHz之間，
普通人發出聲音的頻率範圍約為300Hz~3400Hz。

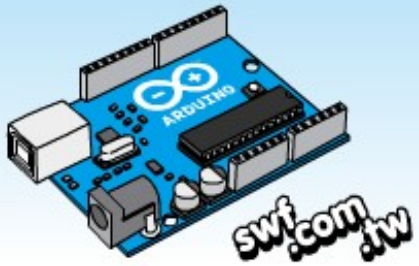




音階與節拍

聲音的頻率（音頻）高低稱為音階（pitch），鋼琴鍵盤就是依照聲音頻率的高低階級（音階）順序來排列。





音階對照表

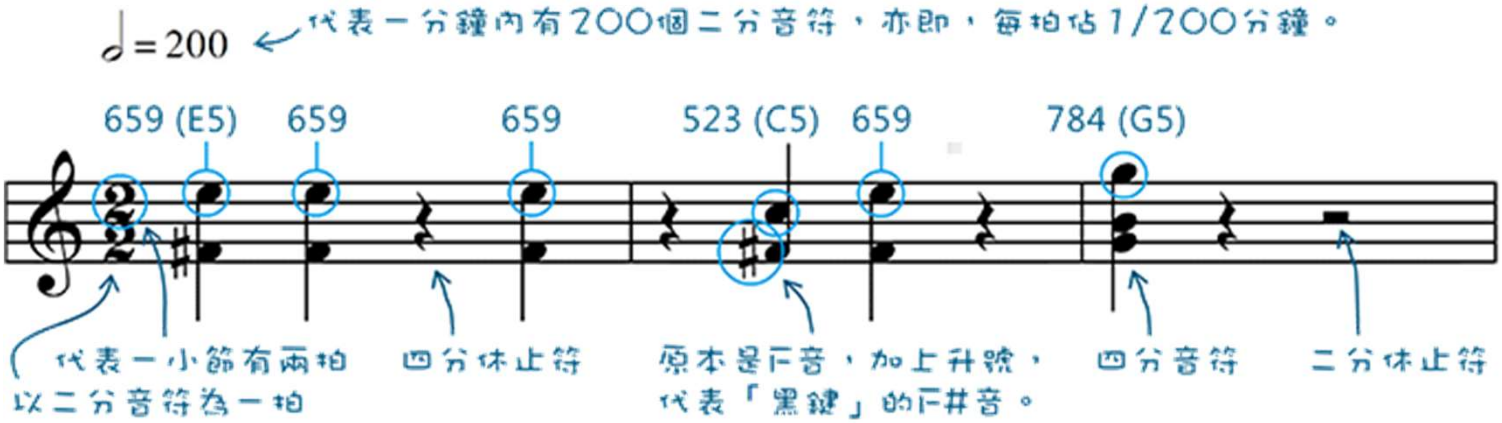
位於鍵盤中間的中央C音（Do）　　88鍵樂器的最高音

	0	1	2	3	4	5	6	7	8
C	16	33	65	131	262	523	1046	2093	4186
C#	17	35	69	139	277	554	1109	2217	4435
D	18	37	73	147	294	587	1175	2349	4699
D#	19	39	78	156	311	622	1245	2489	4978
E	21	41	82	165	330	659	1319	2637	5274
F	22	44	87	175	349	698	1397	2794	5588
F#	23	46	93	185	370	740	1480	2960	5920
G	25	49	98	196	392	784	1568	3136	6272
G#	26	52	104	208	415	831	1661	3322	6645
A	28	55	110	220	440	880	1760	3520	7040
A#	29	58	117	233	466	932	1864	3729	7459
B	31	62	123	247	493	988	1976	3951	7902

88鍵樂器的最低音　　標準音（用於調校樂器，有些採442Hz）

節拍

除了音階，構成旋律的另一個要素是節拍（beat），它決定了音樂的快慢速度。

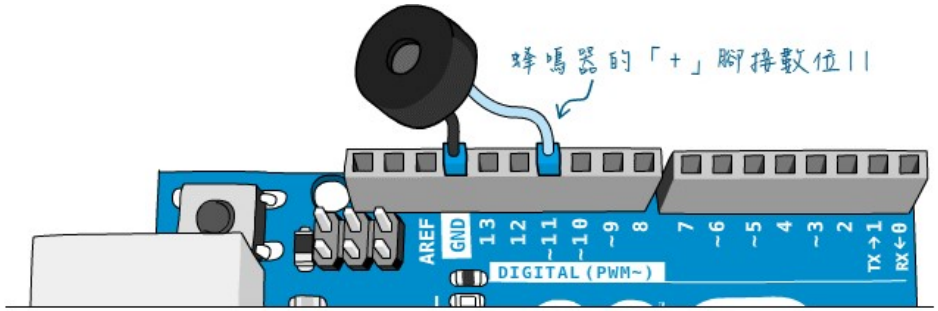


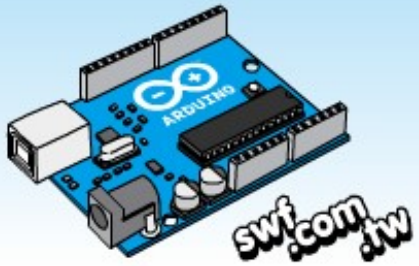
Arduino編輯器內建 “tone"（音調）程式庫，可以輸出指定頻率的聲音和持續時間。

tone(輸出腳位, 頻率, 持續時間);

或： tone(輸出腳位, 頻率);

聲音停止 noTone(輸出腳位);





利用#define指令替換資料

程式碼裡的音階，用音階代碼，比直接用頻率好。

C程式語言有一種用#開頭的特殊指令，稱為巨集（macro），Arduino也是如此。常見用途是在程式編譯之前，載入外部程式檔或者替換字串。

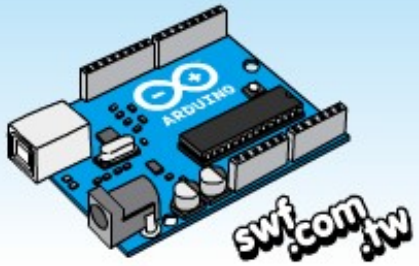
替換文字的巨集指令叫做#define。

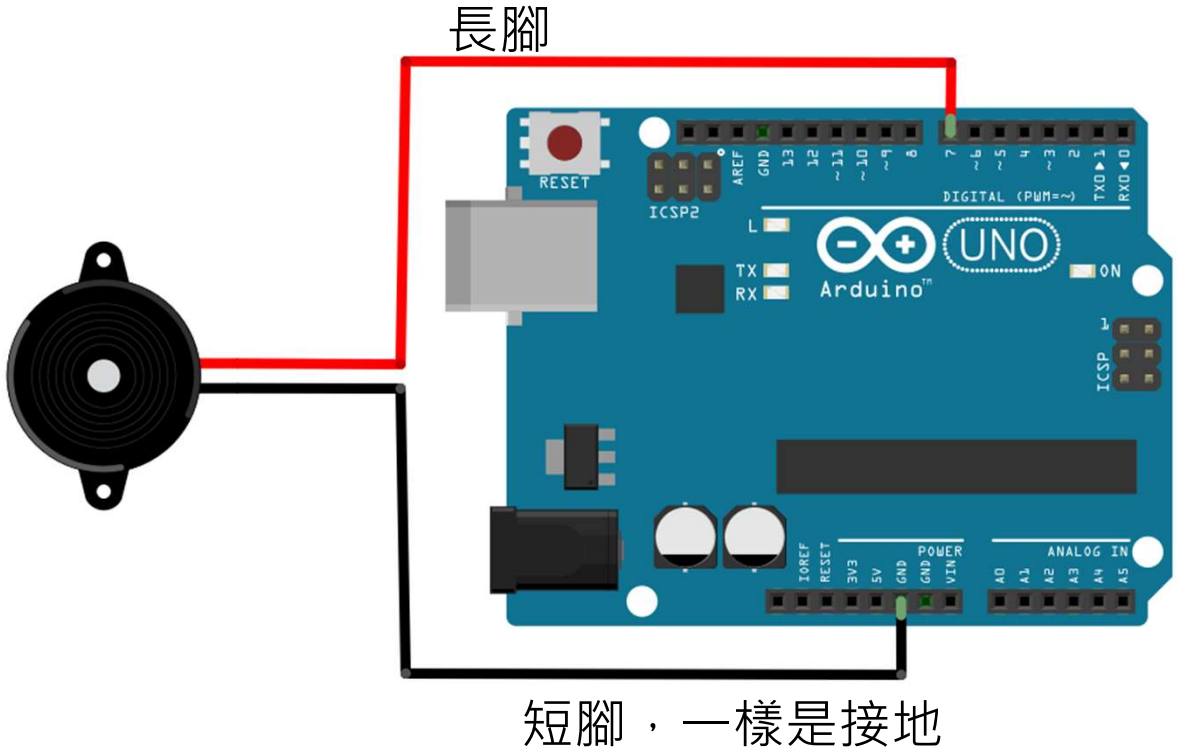
const（常數）代表宣告一個不可改變的資料值，而#define則用於定義置換值。

```
const byte SP_PIN = 11;
#define E5 659
#define C5 523
#define G5 784

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

void loop() {
  tone(SP_PIN, E5, 150);
  delay(150);
  tone(SP_PIN, E5, 150);
  delay(150);
  tone(SP_PIN, E5, 150);
  delay(300);
  tone(SP_PIN, C5, 150);
  delay(150);
  tone(SP_PIN, E5, 150);
  delay(300);
  tone(SP_PIN, G5, 150);
  delay(3000);
}
```

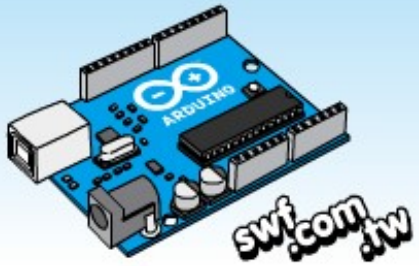
在程式開頭定義替換字（結尾不加分號）

所有E5都會被置換成659

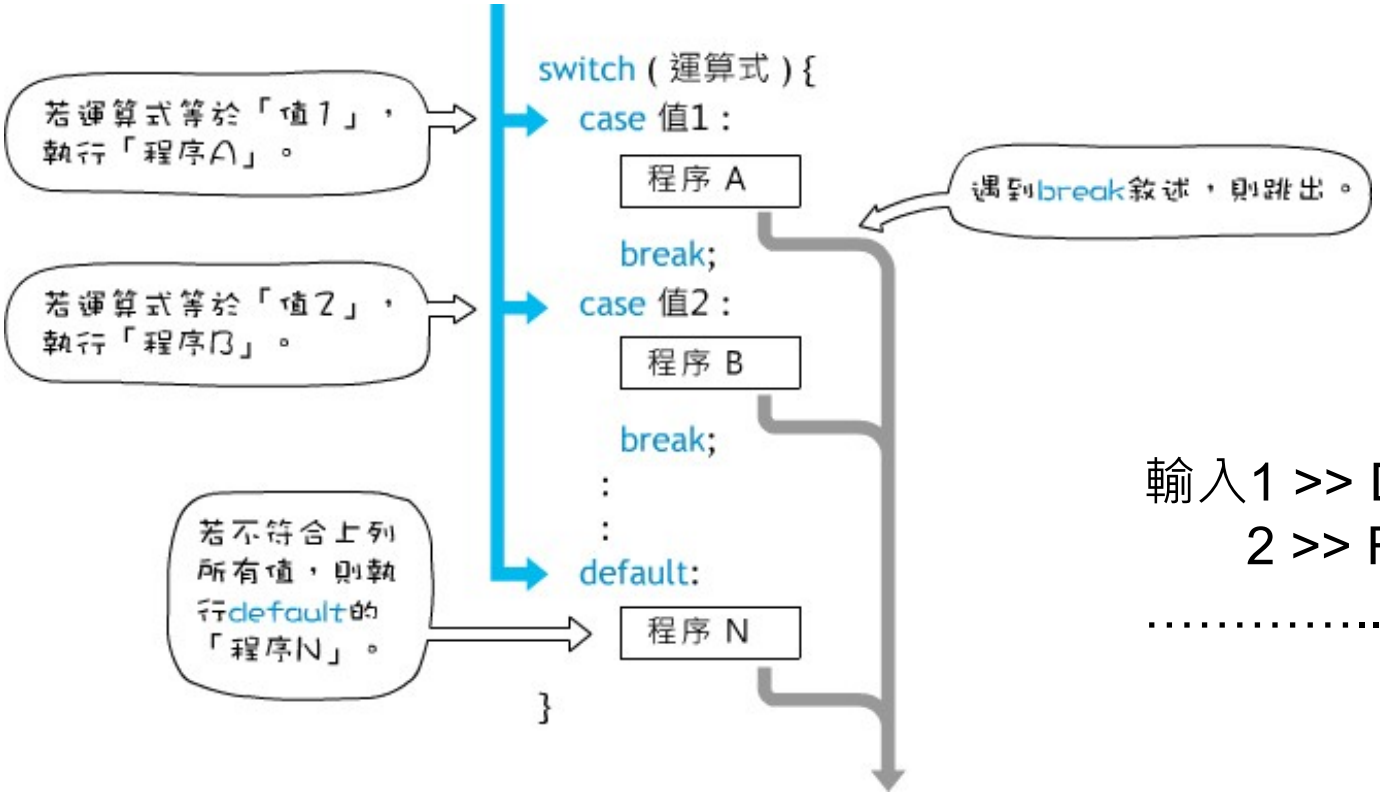




程式非常簡單，tone(7, 頻率) 就能讓蜂鳴器發出聲音了，要持續多久，就用delay()來控制，然後再用noTone(7)就能讓聲音停止了。

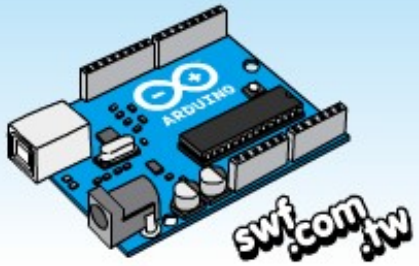


switch...case控制結構



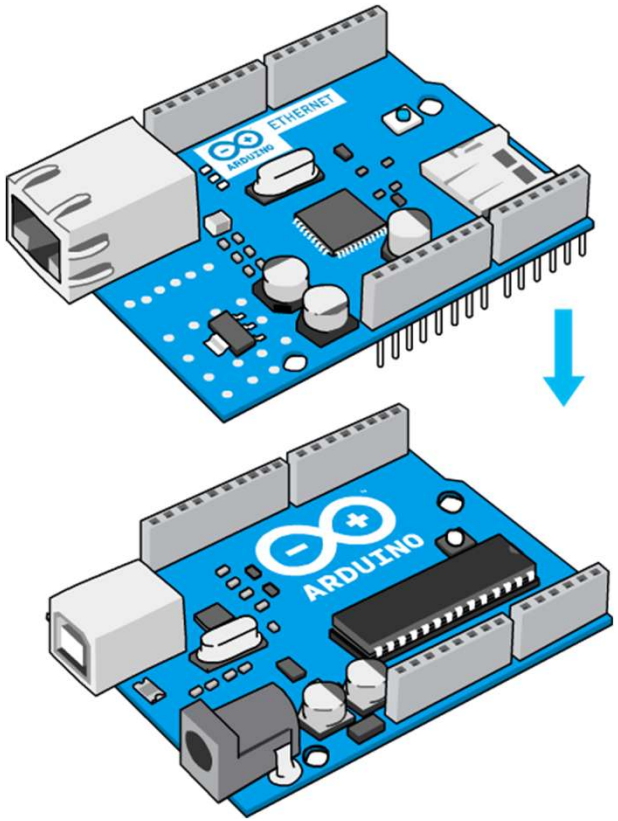
輸入1 >> Do
2 >> Re
.....

switch具有「切換」的涵意：透過比對switch()裡的變數和case後面的值，來決定切換執行哪一段程式。

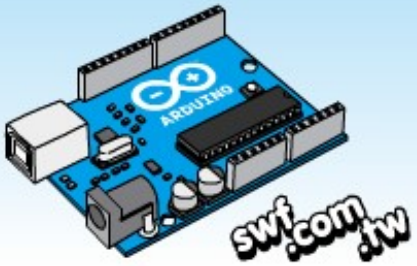


- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

第六章 類比信號處理

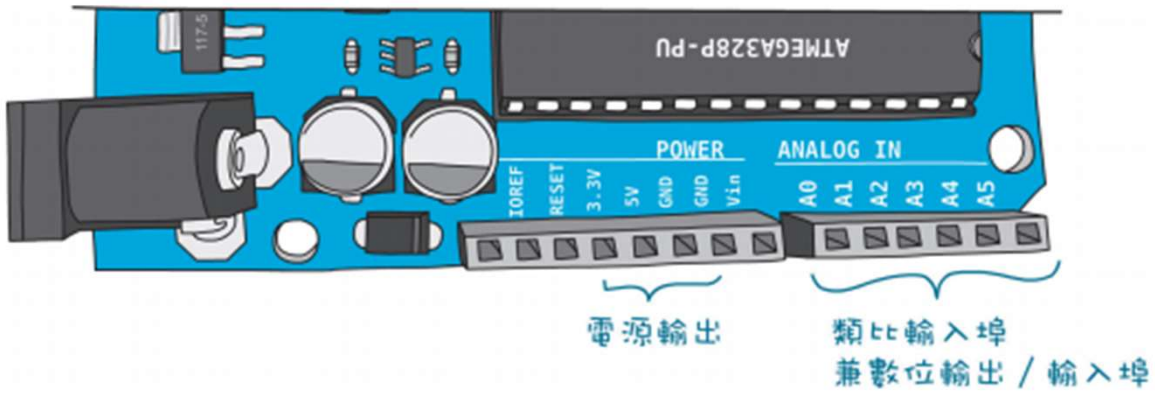


- 讀取類比值
- 從序列埠讀取類比輸入值
- 使用光敏電阻製作小夜燈
- 認識運算放大器
- 拍手控制開關

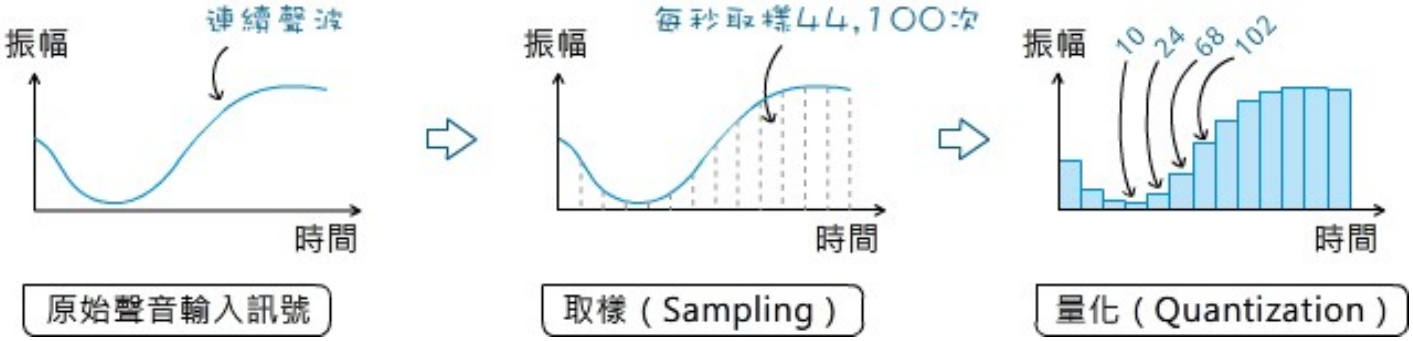


讀取類比值

對Arduino而言，類比資料就是0V到5V之間的電壓變化值，例如：0.8V, 2.7V, 3.6V, ...。



類比轉成數位的過程

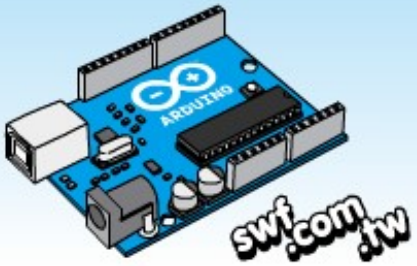


讀取類比值的語法

```
int val = analogRead(類比腳位);
```

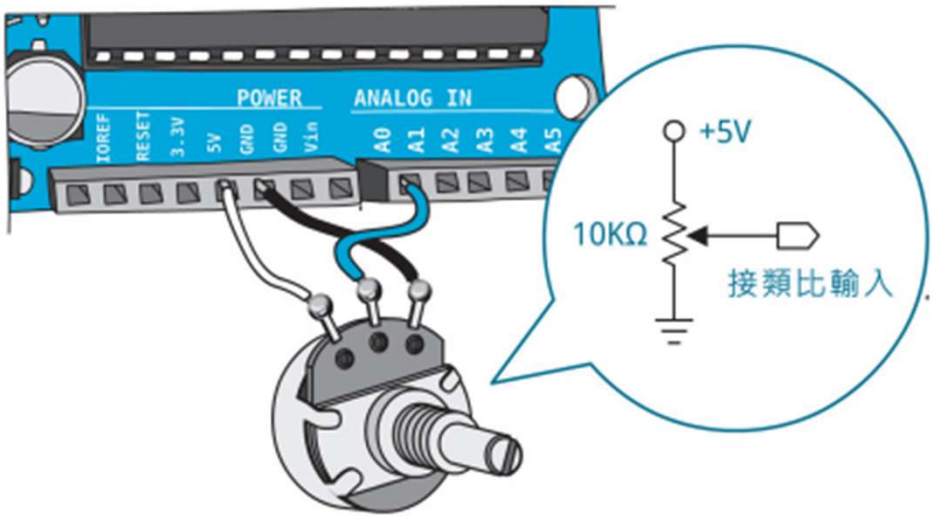
可能值為0~1023

可能值為A0~A5（Leonardo板則是A0~A11）

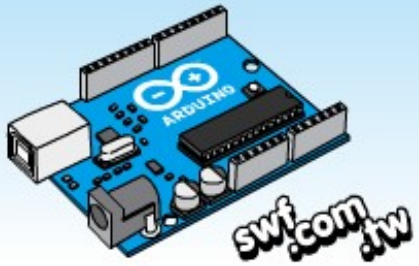


從序列埠讀取類比輸入值

用可變電阻建立一個「電壓調節器」，讓輸出電壓隨著電阻值的變化而改變，藉以模擬類比資料。

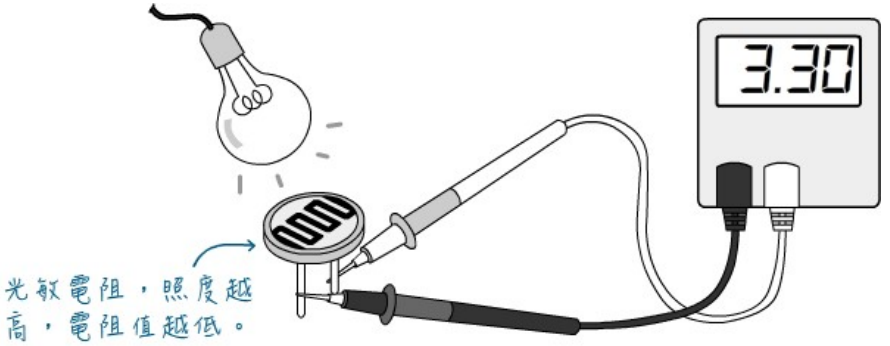



```
const byte potPin = A0;
int val;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = analogRead(potPin);
  Serial.println(val);
  delay(500);
}
```

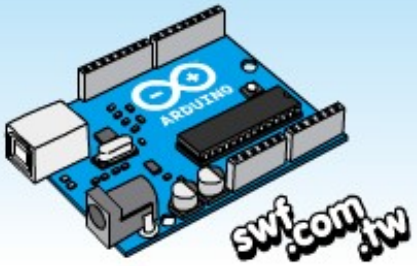


光敏電阻

光敏電阻的阻值會隨著照度（亦即，光的亮度）變化。照度越高，阻值越低。光敏電阻的受光面，有鋸齒狀的感光材料。

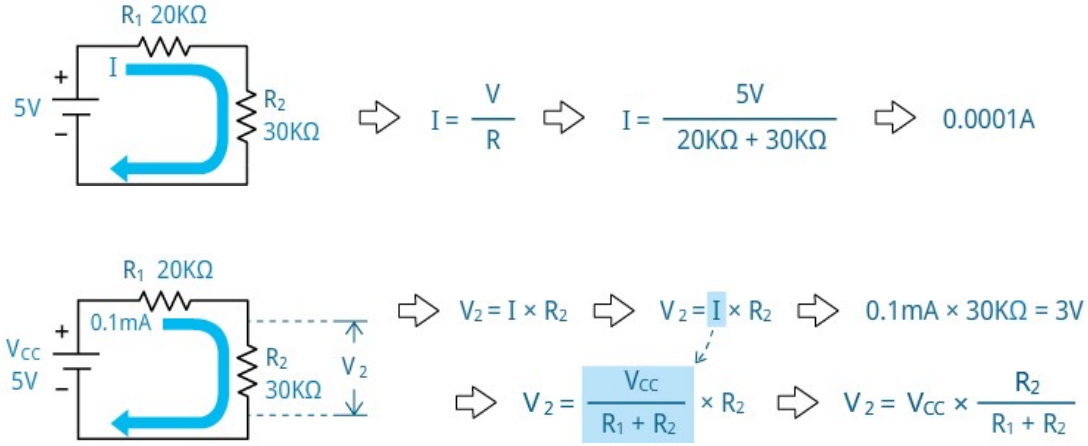


測試條件	CdS電阻值	10KΩ分壓值	4.7KΩ分壓值	1KΩ分壓值
用高亮度LED照射	165Ω	0.08v	0.16v	0.7v
緊急出口指示燈	1KΩ	0.45v	0.87v	2.5v
客廳日光燈	3.3KΩ	1.24v	2.06v	3.83v
室內暗處	18KΩ	3.21v	3.96v	4.73v
用黑色膠布遮蓋	>2MΩ	4.95v	4.98v	4.99v

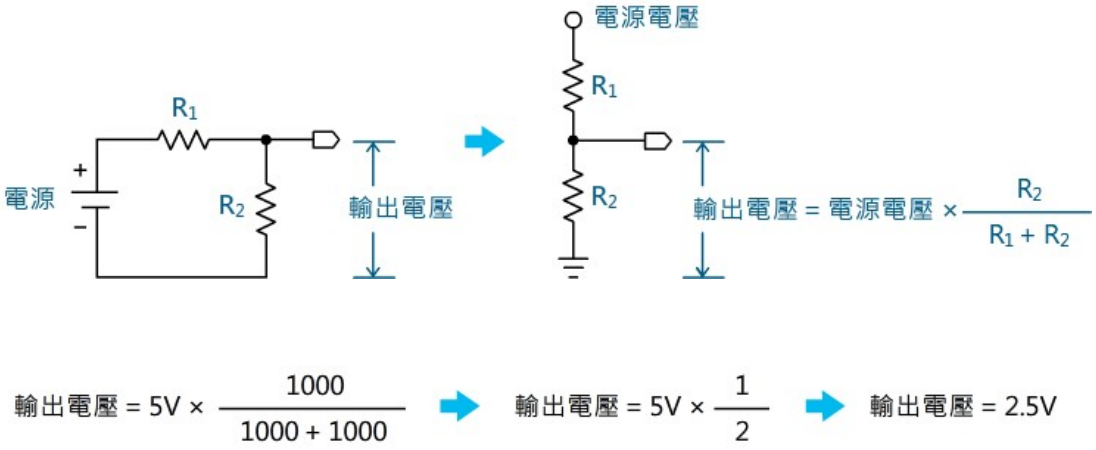


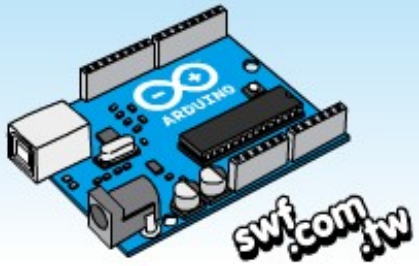
電阻分壓電路

在電子迴路中，流出的電流等於流入的電流。計算電流時，需要把所有電阻值加總。

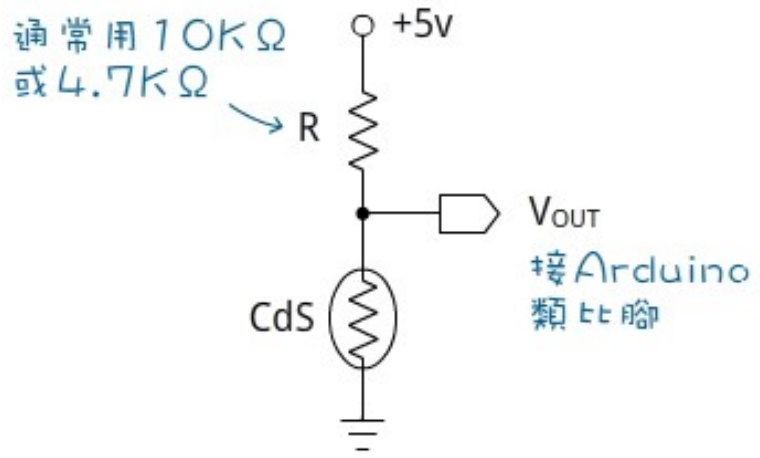


電阻分壓代表分配電壓，使用兩個電阻構成的分壓電路與電壓計算公式：





使用光敏電阻製作小夜燈



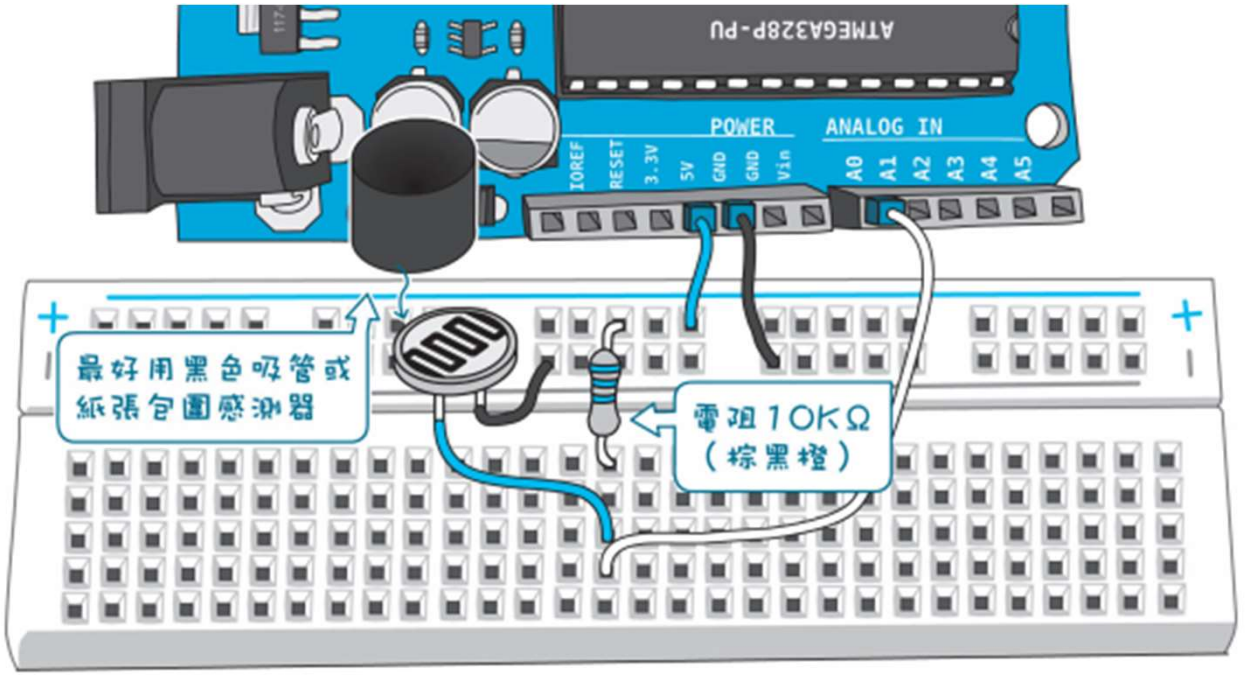
$$V_{OUT} = 5v \times \frac{CdS}{R + CdS}$$

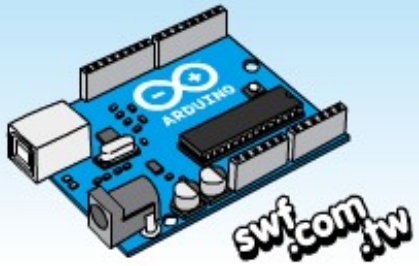
← 電阻分壓的計算式

$$V_{OUT} = 5v \times \frac{3300\Omega}{10000\Omega + 3300\Omega} \approx 1.24v$$

← 在室內光源測得的阻值約3.3KΩ

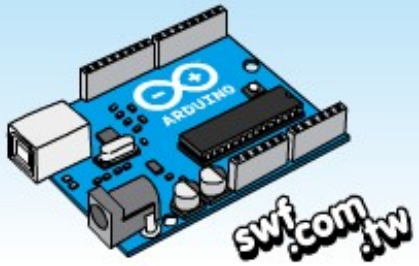
使用光敏電阻和另一個電阻構成分壓電路；光敏電阻會隨著光線變化改變阻值。





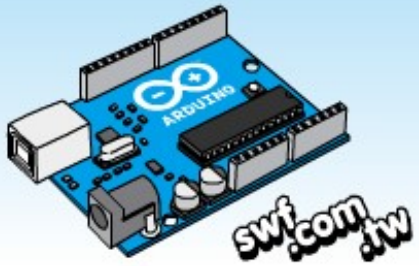
使用光敏電阻開關燈

```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run
repeatedly:
float val = analogRead(A1);
Serial.println(val);
if (val>322) {
  digitalWrite(13, HIGH); }
else {
  digitalWrite(13, LOW);
}
delay(500);
}
```



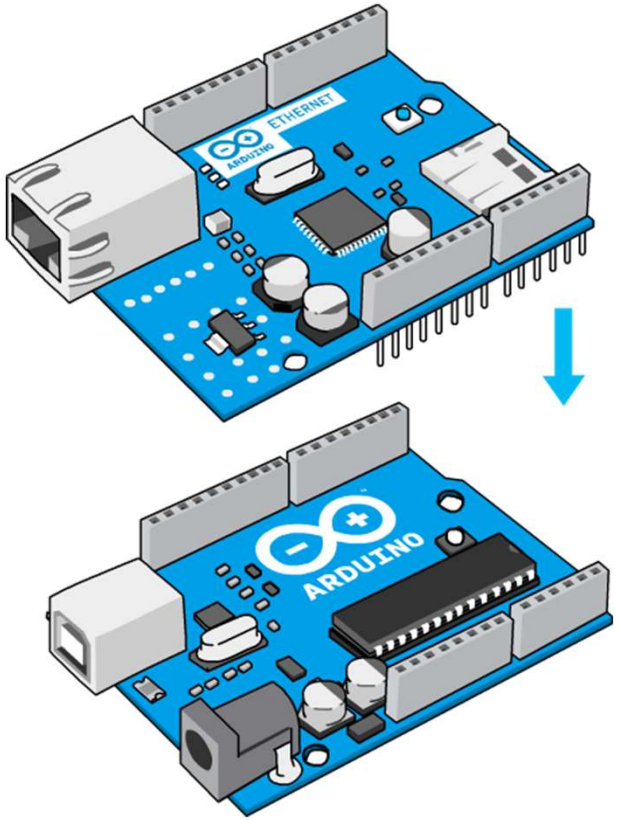
使用光敏電阻計數

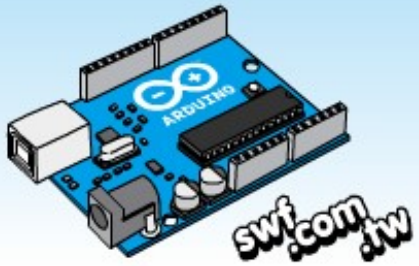
```
int count=0;
int s=0;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);  }
void loop() {
  // put your main code here, to run repeatedly:
float val=analogRead(A0);
Serial.print(val);
Serial.print("    counter : ");
  if (val>350) {
  s=1;
  delay(100);  }
if (val<300) {
  count =count+s;
  s=0;  }
Serial.println(count);
if (Serial.available()) {
  char reset = Serial.read();
  if(reset="0") {
    count = 0;   }  }
delay(1000);  }
```



- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

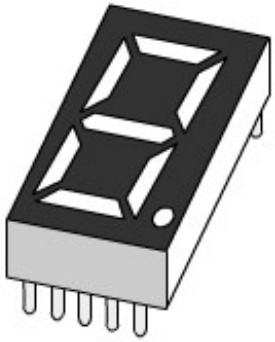
第七章 LED七段顯示器



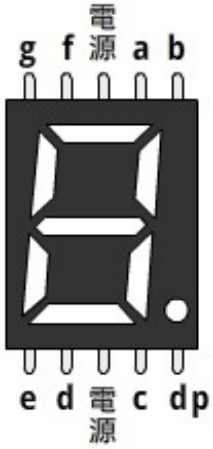


七段顯示器

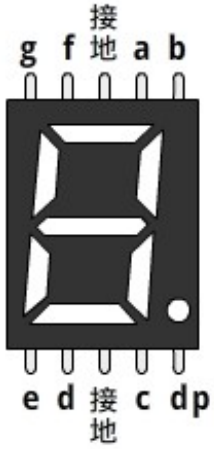
七段顯示器是內建八個LED的顯示元件，為了方便解說，
內部LED分別標上a~g和dp（點）代號。



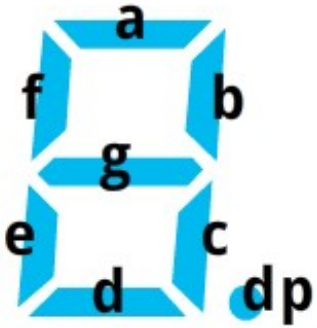
七段顯示器



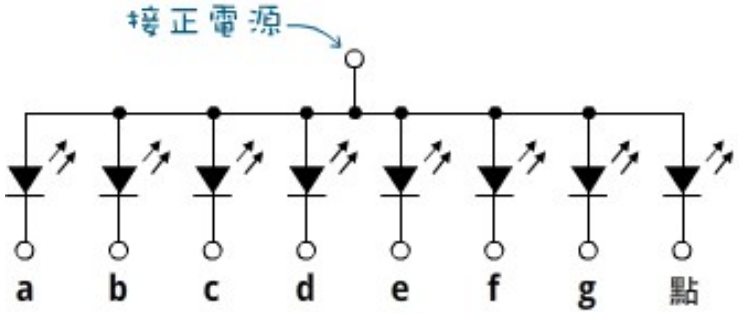
共陽極腳位



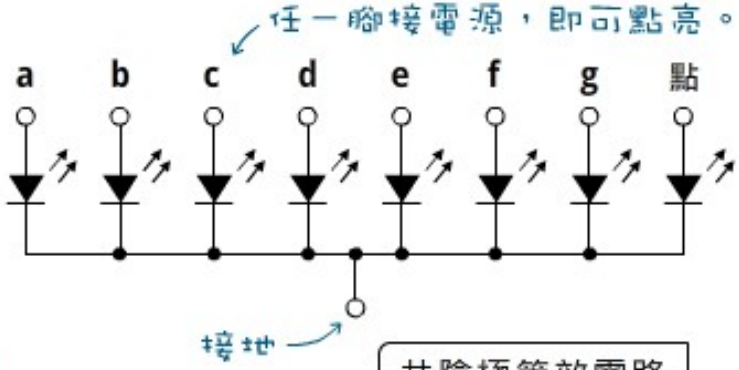
共陰極腳位



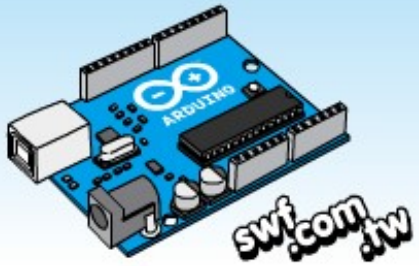
內部LED的編號



共陽極等效電路

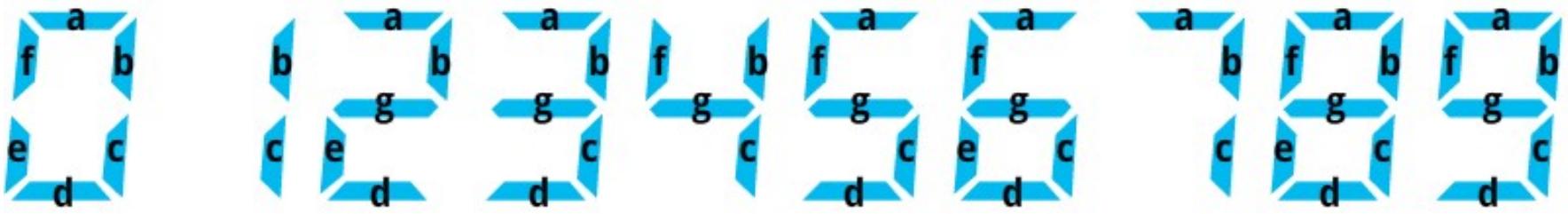


共陰極等效電路



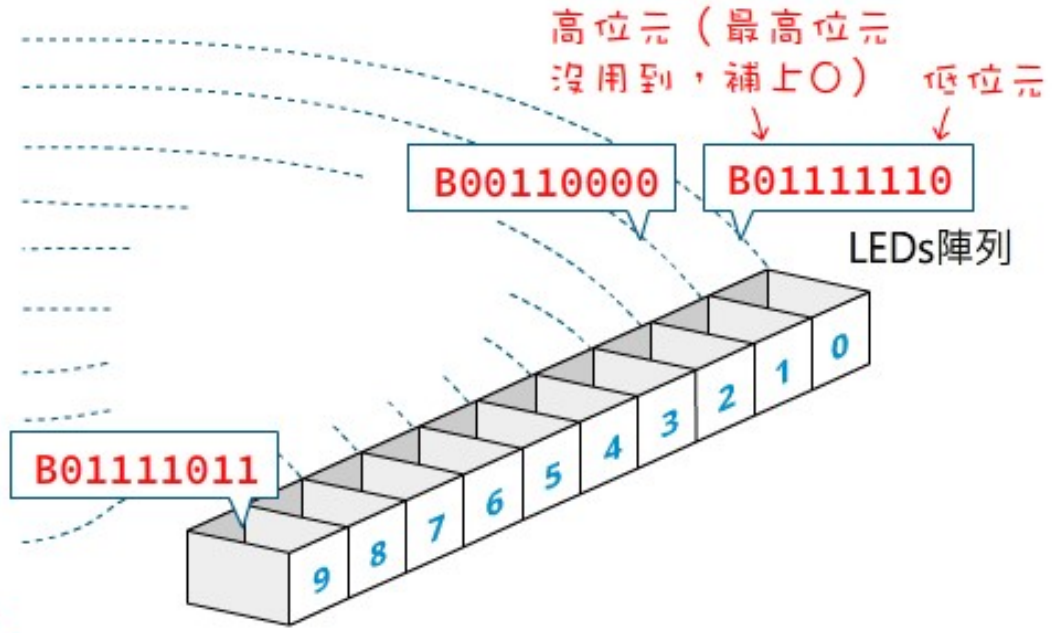
用陣列儲存七段顯示數字

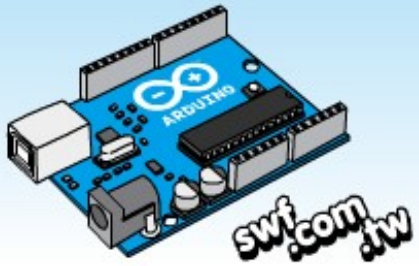
下圖顯示了呈現某個數字所需點亮的LED代號，並用陣列儲存。



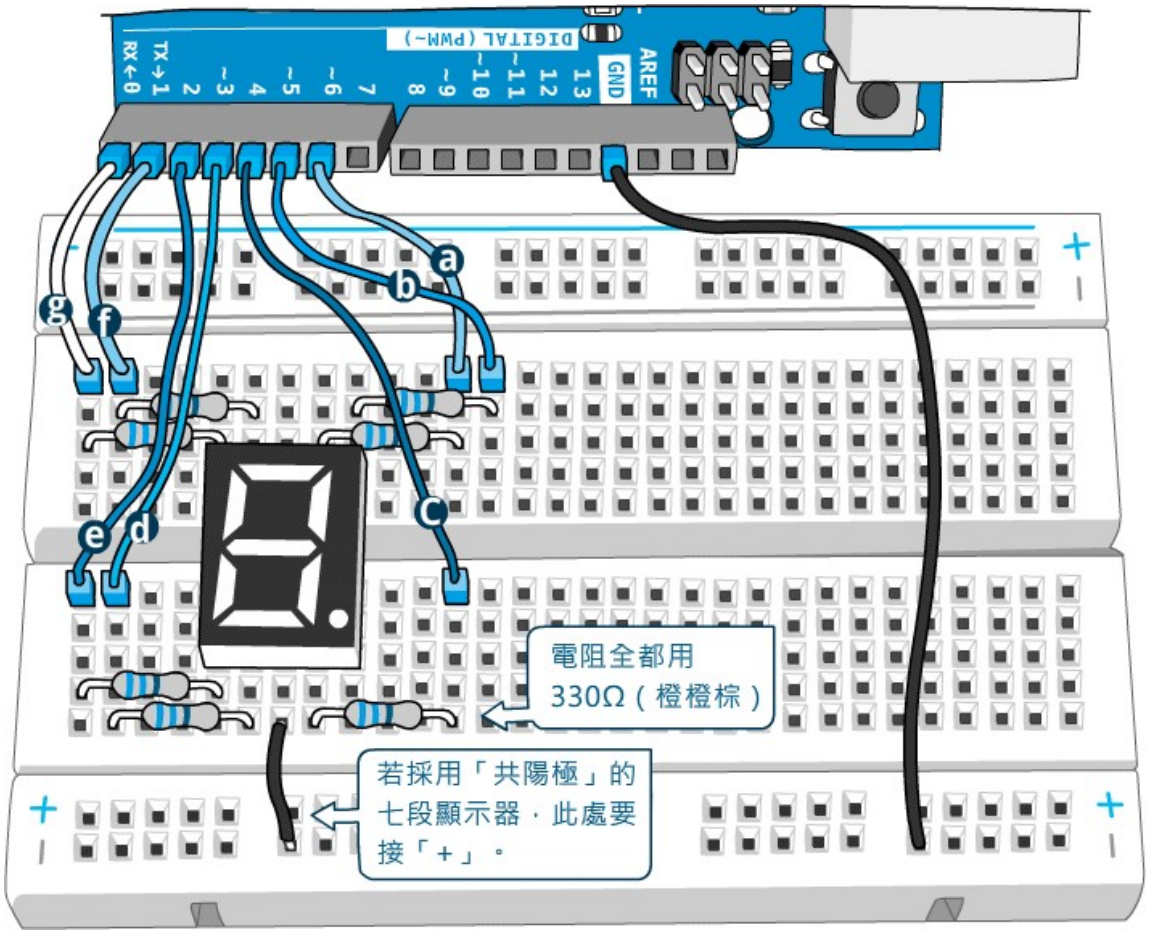
十進位數	a	b	c	d	e	f	g	設定值
0	1	1	1	1	1	1	0	
1	0	1	1	0	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	1	1	0	0	1	
4	0	1	1	0	0	1	1	
5	1	0	1	1	0	1	1	
6	1	0	1	1	1	1	1	
7	1	1	1	0	0	0	0	
8	1	1	1	1	1	1	1	
9	1	1	1	1	0	1	1	

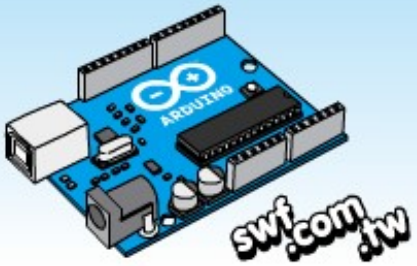
高位元　　低位元



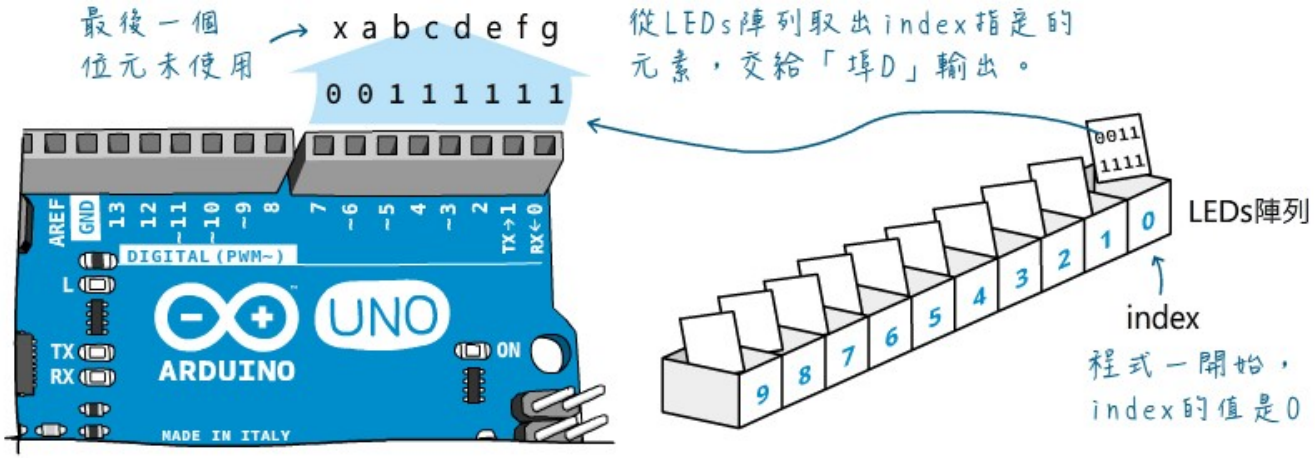


連接LED七段顯示器與Arduino板





顯示數字的程式

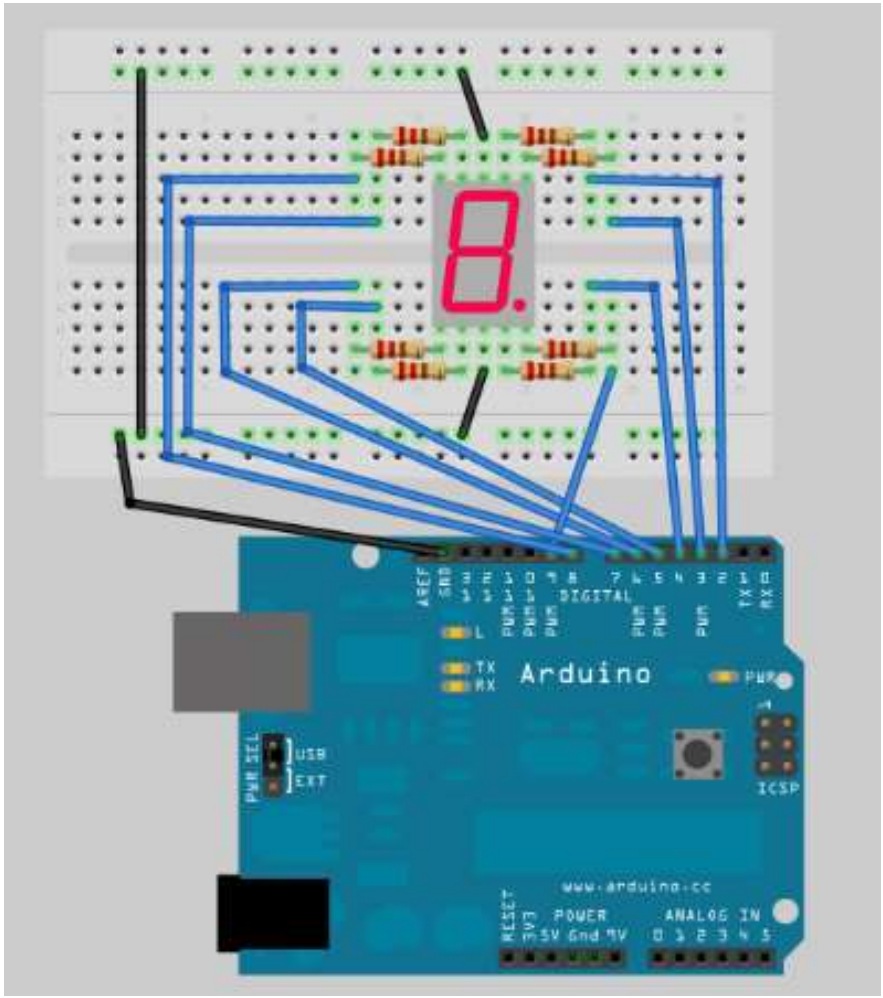


從LEDs陣列，取出數字編碼輸出給「埠D」腳位，傳給七段顯示器。

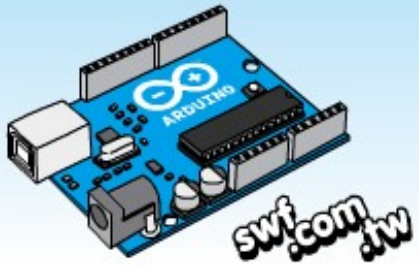
```
void loop() {
  PORTD = LEDs[index];

  index ++;
  if (index == 10) {
    index = 0;
  }
  delay(1000);
}
```

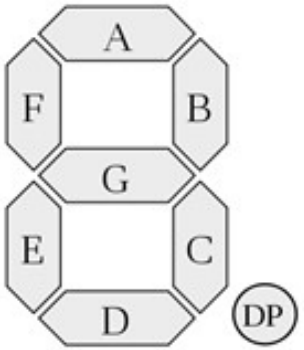
取出陣列裡的數字資料，由「埠口D」輸出。

直接連法



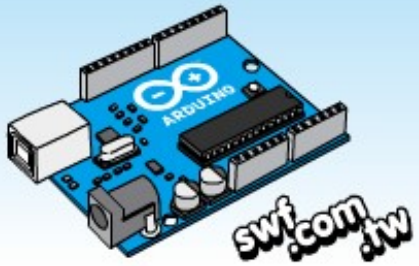
Arduino 腳位	七段示顯器腳位 (筆劃)
2	7 (A)
3	6 (B)
4	4 (C)
5	2 (D)
6	1 (E)
7	9 (F)
8	10 (G)
9	5 (DP)





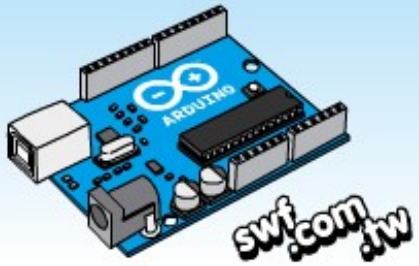
顯示數字	dp	a	b	c	d	e	f	g	十六進制
0	0	1	1	1	1	1	1	0	0x3F
1	0	0	1	1	0	0	0	0	0x06
2	0	1	1	0	1	1	0	1	0x5B
3	0	1	1	1	1	0	0	1	0x4F
4	0	0	1	1	0	0	1	1	0x66
5	0	1	0	1	1	0	1	1	0x6D
6	0	1	0	1	1	1	1	1	0x7D
7	0	1	1	1	0	0	0	0	0x27
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	1	1	0	1	1	0x67

上圖為共陰，共陽則1>>0，0>>1



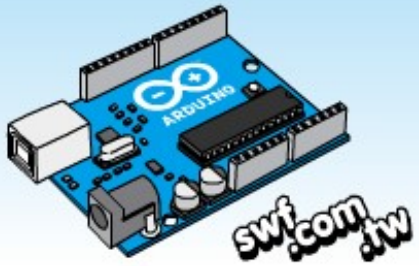
0~9 循環出現

```
#define NUM 8
int pins[NUM] = {2, 3, 4, 5, 6, 7, 8, 9};

#define t true
#define f false
boolean data[10][NUM] = {
  {t, t, t, t, t, t, f, f}, // 0
  {f, t, t, f, f, f, f, f}, // 1
  {t, t, f, t, t, f, t, f}, // 2
  {t, t, t, t, f, f, t, f}, // 3
  {f, t, t, f, f, t, t, f}, // 4
  {t, f, t, t, f, t, t, f}, // 5
  {t, f, t, t, t, t, t, f}, // 6
  {t, t, t, f, f, f, f, f}, // 7
  {t, t, t, t, t, t, t, f}, // 8
  {t, t, t, t, f, t, t, f}, // 9 };

void setup(){
  for(int i = 0; i < NUM; i++){
    pinMode(pins[i], OUTPUT);    }  }

void writeNumber(int n){
  for(int i = 0; i < NUM; i++){
    digitalWrite(pins[i], data[n][i] == t ? HIGH : LOW);
  }
}

void loop(){
  for(int n = 0; n <= 9; n++){
    writeNumber(n);
    delay(1000);   } }
```



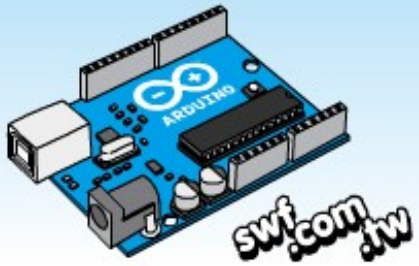
視窗輸入數字，出現在7段顯示器

```
#define NUM 8
int pins[NUM] = {2, 3, 4, 5, 6, 7, 8, 9};
void setup(){
  for(int i = 0; i < NUM; i++){
    pinMode(pins[i], OUTPUT);
    Serial.begin(9600);   }  }
void loop(){
if (Serial.available()) {
char val = Serial.read();
switch (val) {
case ('1'):
digitalWrite(pins[0], 1);
digitalWrite(pins[1], 0);
digitalWrite(pins[2], 0);
digitalWrite(pins[3], 1);
digitalWrite(pins[4], 1);
digitalWrite(pins[5], 1);
digitalWrite(pins[6], 1);
digitalWrite(pins[7], 1);
delay(1000);
break;
case ('2'):
digitalWrite(pins[0], 0);
digitalWrite(pins[1], 0);
digitalWrite(pins[2], 1);
digitalWrite(pins[3], 0);
digitalWrite(pins[4], 0);
digitalWrite(pins[5], 1);
digitalWrite(pins[6], 0);
digitalWrite(pins[7], 1);
delay(1000);
} } }
```



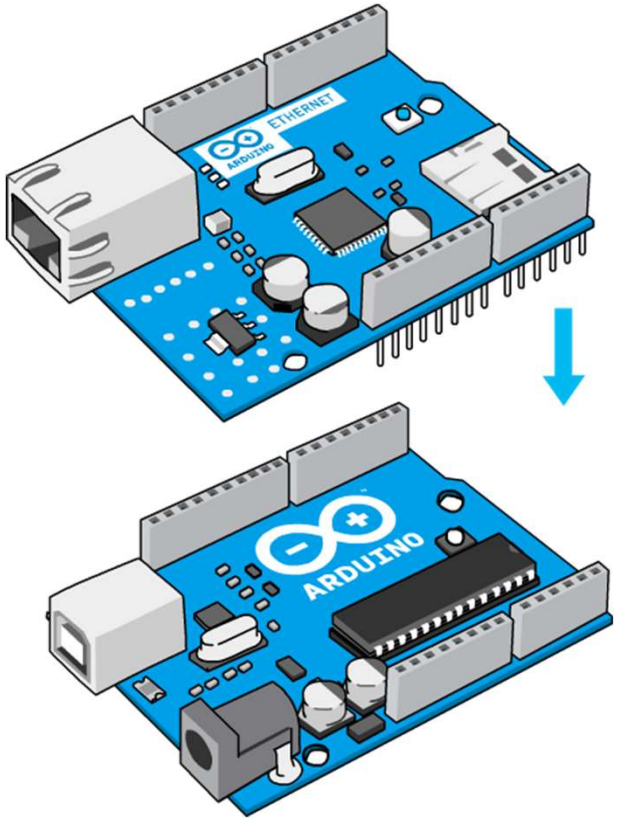
倒數計時器

視窗輸入”S”，
1.(25%)開始倒數計時，黃色LED亮起
2.每個數字間隔1秒，數字由 9 顯示在7段顯示器上；當數字顯示為0 時，即倒數完成
3.紅色LED亮起，黃色滅，閃三下
4.然後長亮。

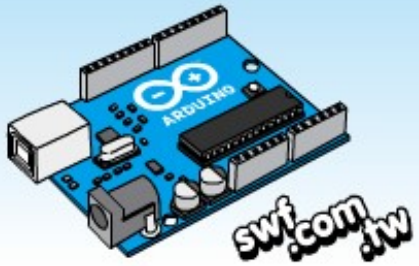


- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

第八章 LED矩陣控制

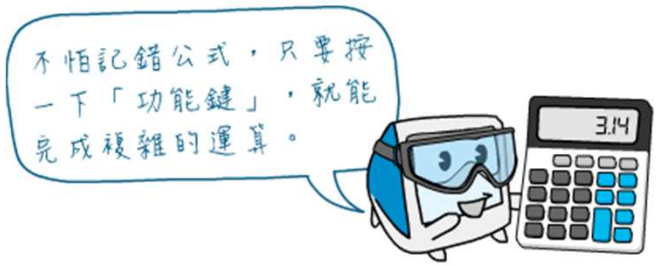


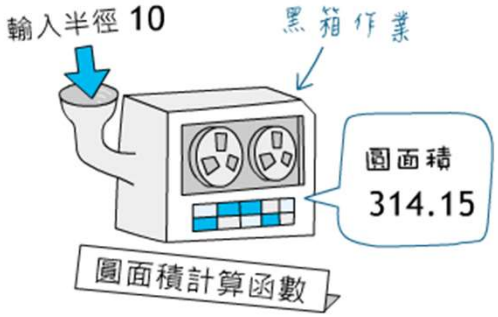
- 建立自訂函數
- 認識變數的有效範圍
- LED矩陣簡介
- 顯示靜態與動態矩陣圖像
- 指標與程式記憶體



建立自訂函數

具有特定功能並且能被重複使用的程式碼，叫做「函數」。

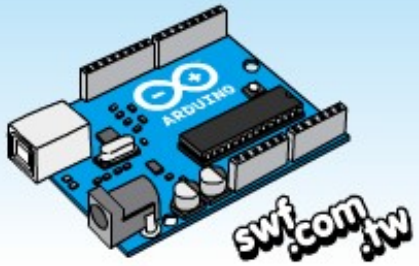




自訂函數的語法範例

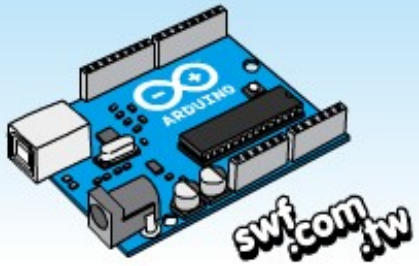
```
傳回值類型  函數名稱 (參數1, 參數2, ...) {
    運算式1;
    運算式2;
        :
    return 運算結果 ;
}
```

```
float cirArea() {
  int r = 5;
  float area = 3.14 * r * r;
  Serial.println(area);
}

void setup() {
  Serial.begin(9600);
}

void loop() {
  cirArea();
  delay(2000);
}
```



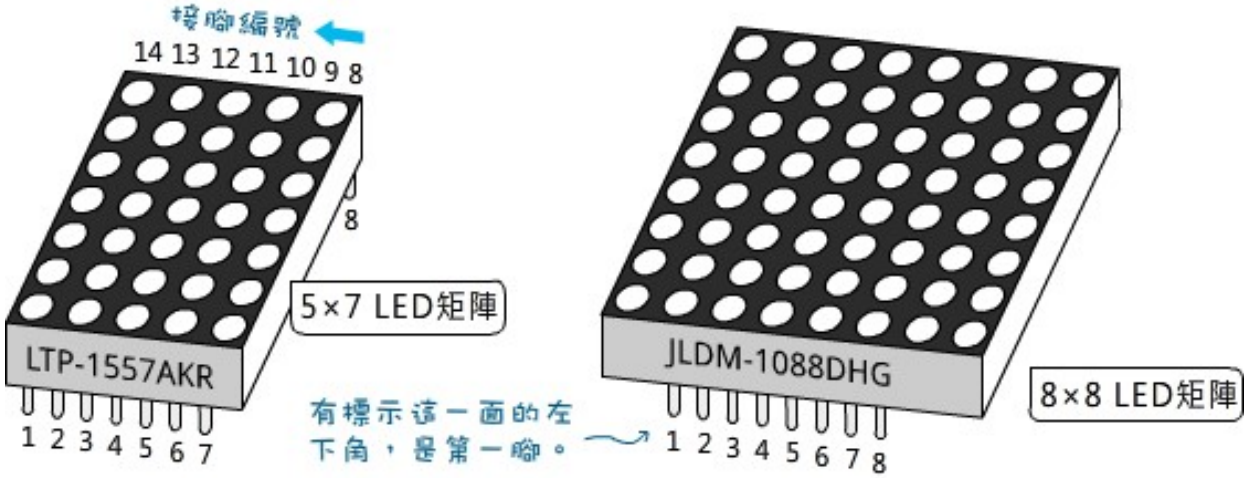
變數的有效範圍

```
int age = 20;                    ← 在室外宣告的變數：全域變數

void check() {
  int age = 10;                  ← 在室內宣告的變數：區域變數
  Serial.print("function: ");
  Serial.println(age);
}

void setup() {
  Serial.begin(9600);
  check();
  Serial.print("setup: ");
  Serial.println(age);
}

void loop() {
}
```

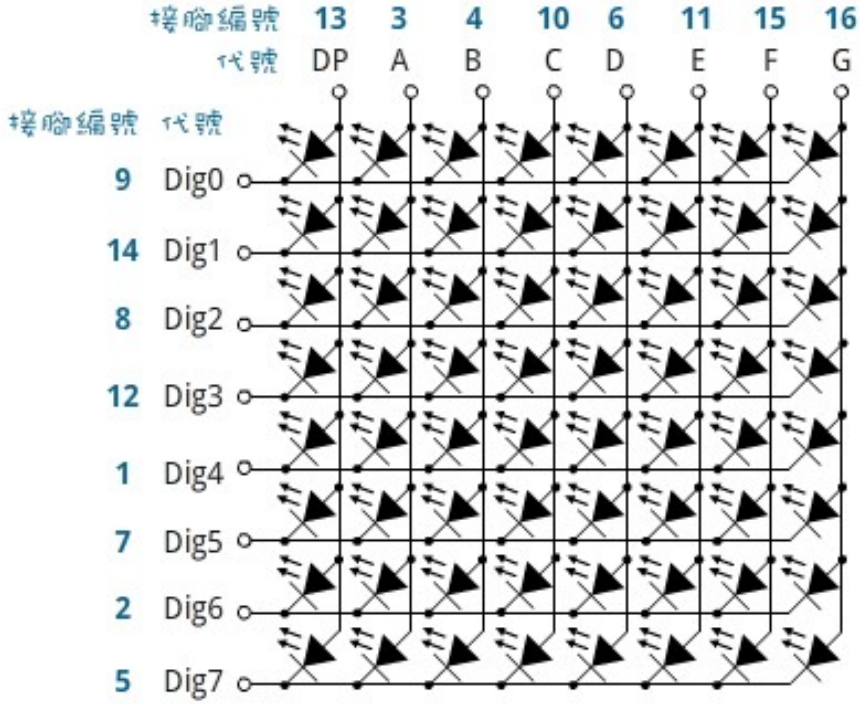


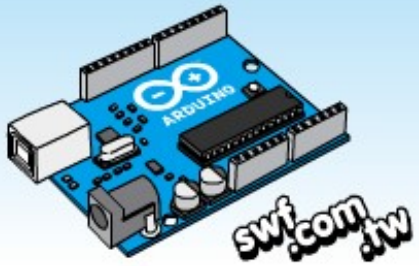
LED矩陣簡介

LED矩陣（LED Matrix）是一種把數十個LED排列封裝在一個方形元件的顯示單元



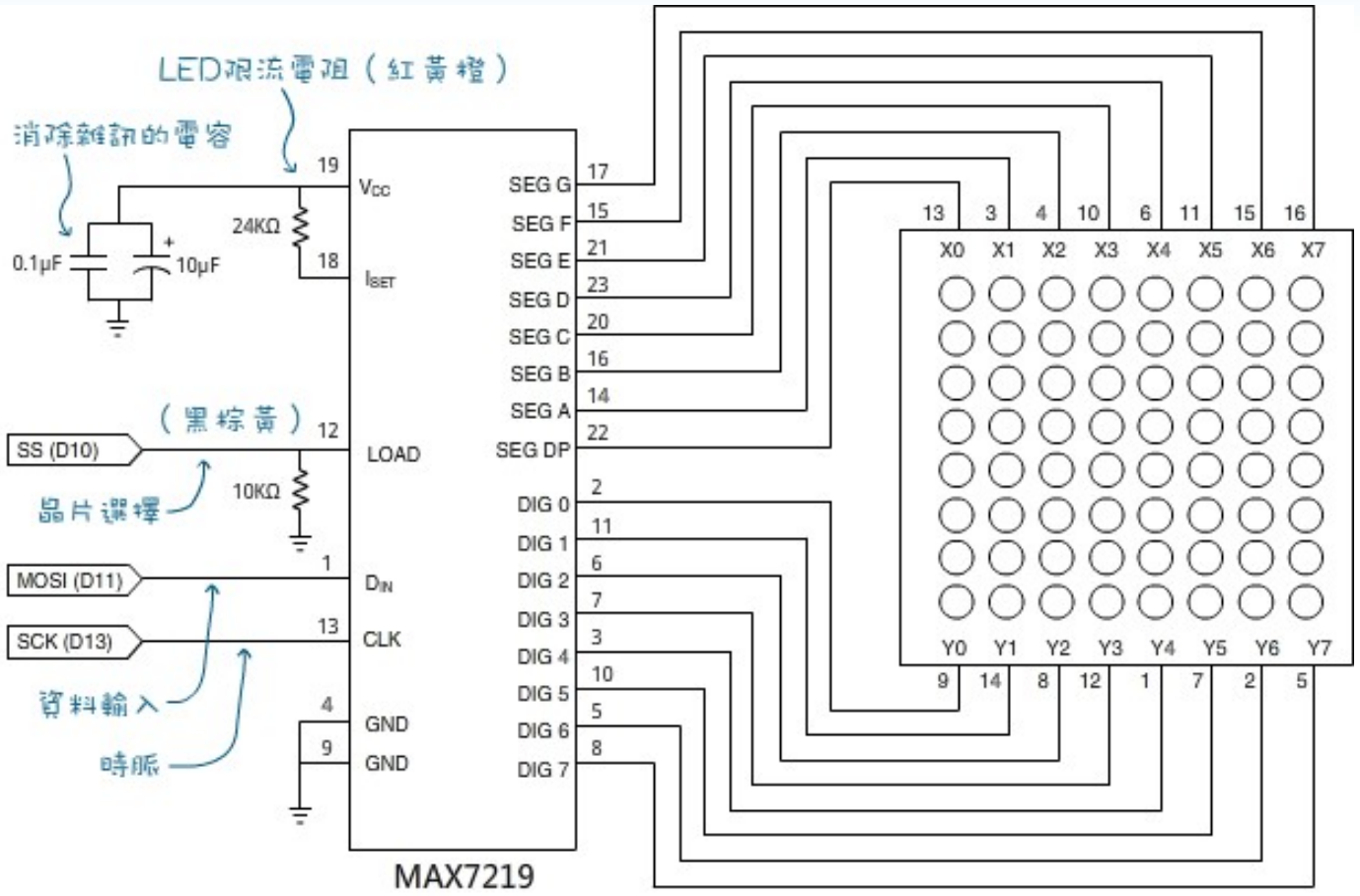
LED矩陣的內部結構



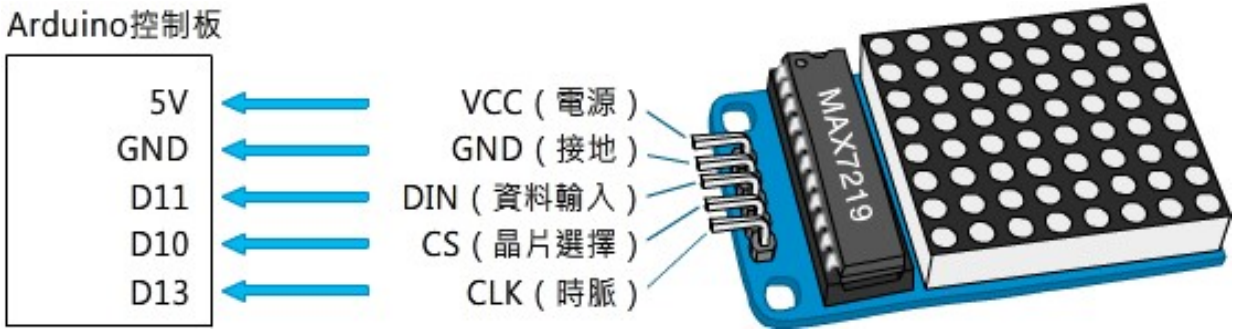


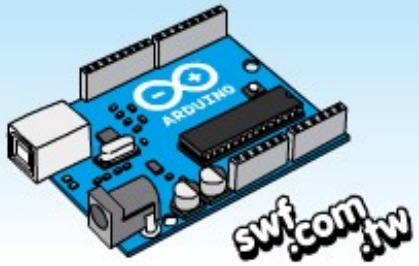
LED矩陣電路

驅動一個8×8單色
LED矩陣的電路圖



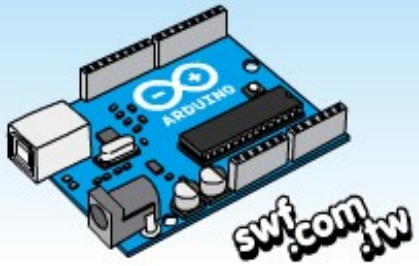
可以採用市售的MAX7219
與8×8rm LED矩陣的套件





LED矩陣 單個字顯示

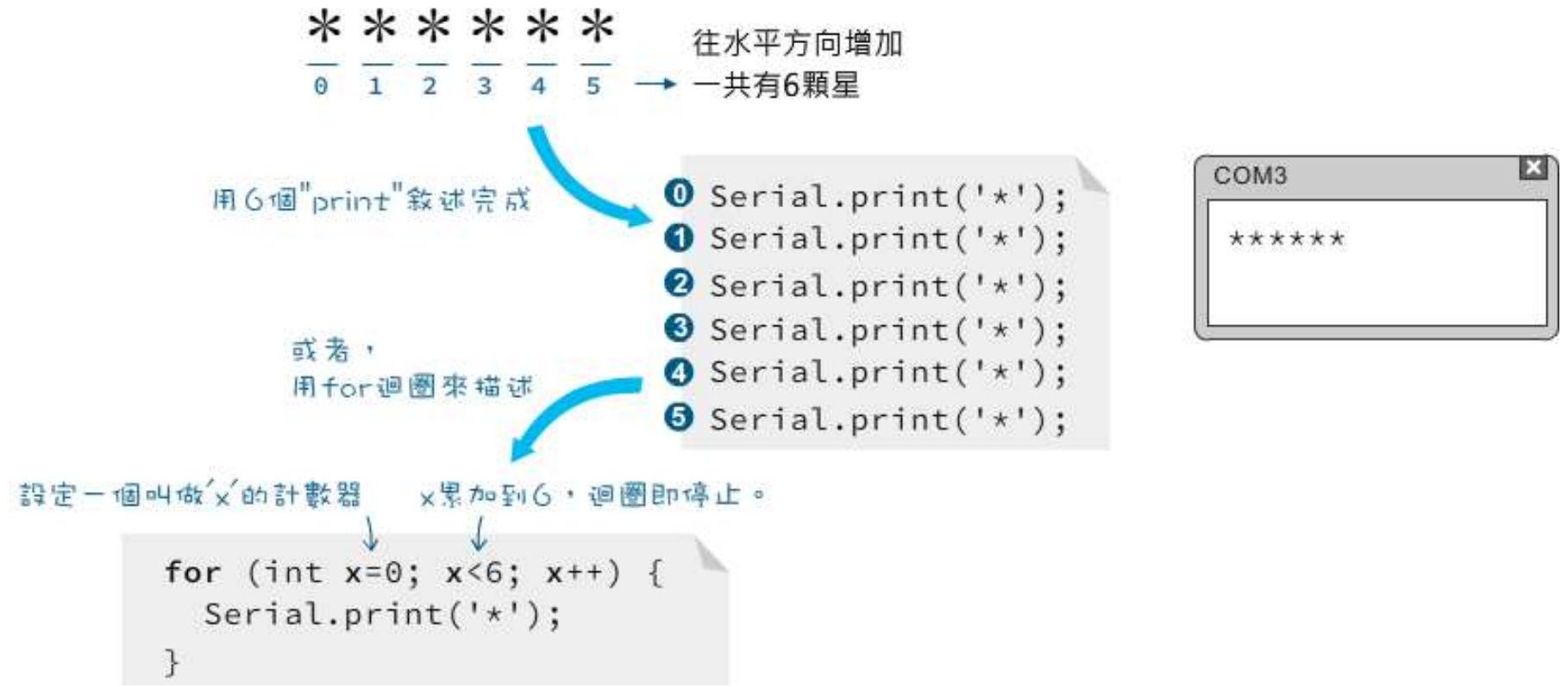
```
#include <SPI.h>
byte sym[8]={0xC0,0x78,0x1E,0x13,0x13,
0x1E,0x78,0xC0};  //英文字母A
const byte NOOP=0x0;  //從此起為固定輸入，啟動LED矩陣
const byte DECODEMODE=0x9;
const byte INTENSITY=0xA;
const byte SCANLIMIT=0xB;
const byte SHUTDOWN=0xC;
const byte DISPLAYTEST=0xF;
void max7219(byte reg, byte data) {
  digitalWrite(SS,0);
  SPI.transfer(reg);
  SPI.transfer(data);
  digitalWrite(SS,1);   }
void setup() {
pinMode(SS,OUTPUT);
digitalWrite(SS,1);
SPI.begin();
max7219(SCANLIMIT,7);
max7219(DECODEMODE,0);
max7219(INTENSITY,8);
max7219(DISPLAYTEST,0);
max7219(SHUTDOWN,1);   //到此結束
for (byte i=0; i<8; i++)  // 清除LED顯示為空白
max7219(i+1, 0); }
void loop() {
for (byte i=0; i<8; i++)
max7219(i+1,sym[i]);   }
```

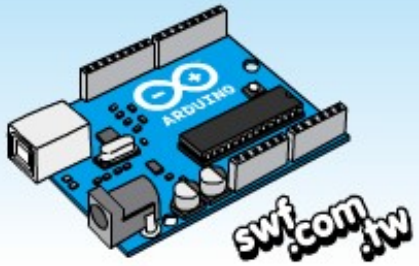


在序列埠監控視窗輸出矩形排列的星號（一）

如何呈現6個水平排列的星號？

- 用6個"print()"函數顯示星號。
- 用一個for迴圈完成。



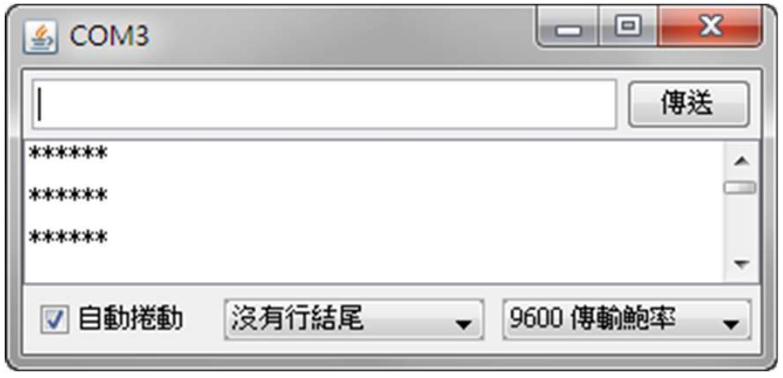


在序列埠監控視窗輸出矩形排列的星號（二）

```
0 * * * * * *
1 * * * * * *
2 * * * * * *
```

往垂直方向增加，
一共有三組，
每一組有6個星號。

分成3段for迴圈來完成



完成6×3排列顯示效果的雙重迴圈程式

```
❶ for (int x=0; x<6; x++) {
    Serial.print('*');
  }
  Serial.print('\n');
❷ for (int x=0; x<6; x++) {
    Serial.print('*');
  }
  Serial.print('\n');
❸ for (int x=0; x<6; x++) {
    Serial.print('*');
  }
  Serial.print('\n');
```

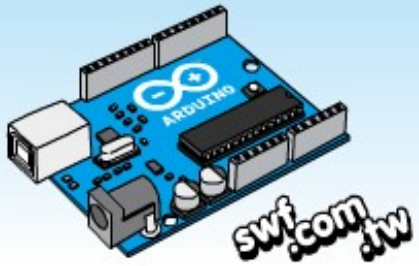
顯示6個星號，
加上1個「新行」結尾。

外層的計數器叫做'y'

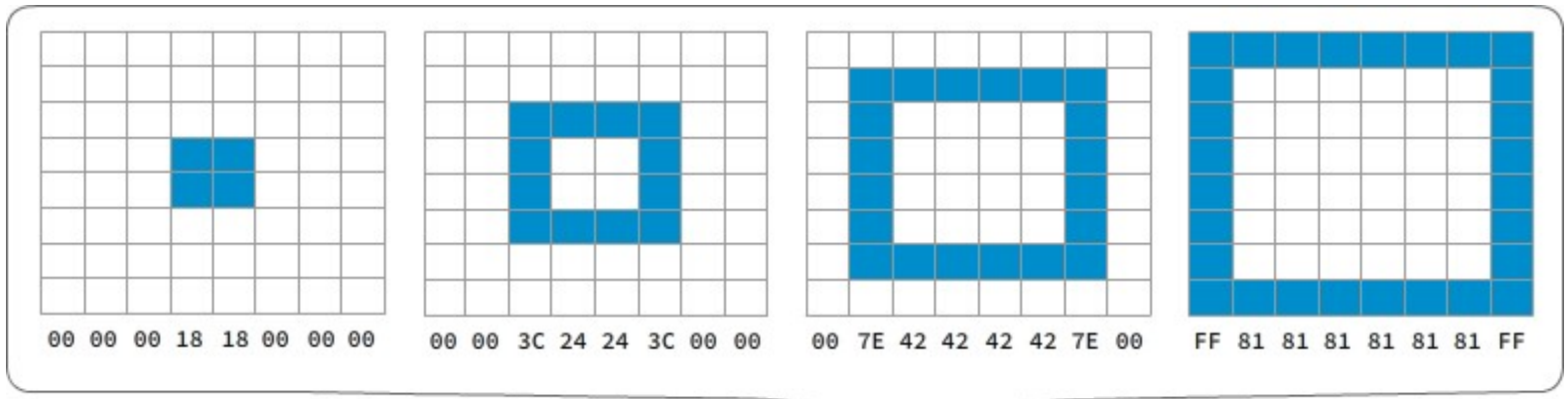
```
for (int y=0; y<3; y++) {
  for (int x=0; x<6; x++) {
    Serial.print('*');
  }
  Serial.print('\n');
}
```

這些敘述將被執行3次，
每次顯示6個星號。

使用雙重for
迴圈描述



顯示動態LED矩陣圖像

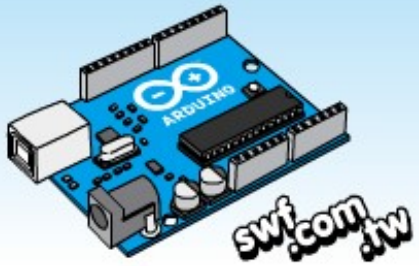


二維陣列定義LED矩陣圖像：

四組樣式　每組有八行

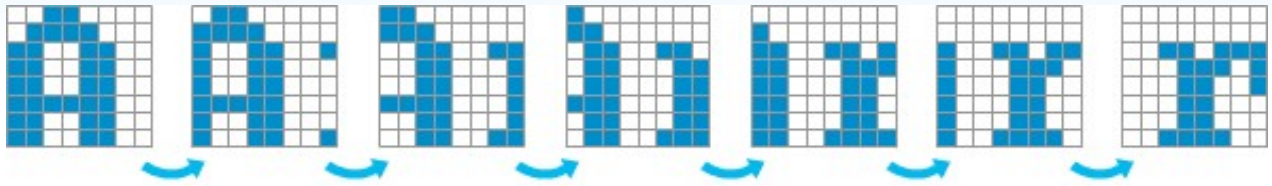
```
const byte sprite[4][8] = {
  { 0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00 },
  { 0x00, 0x00, 0x3C, 0x24, 0x24, 0x3C, 0x00, 0x00 },
  { 0x00, 0x7E, 0x42, 0x42, 0x42, 0x42, 0x7E, 0x00 },
  { 0xFF, 0x81, 0x81, 0x81, 0x81, 0x81, 0x81, 0xFF }
};
```

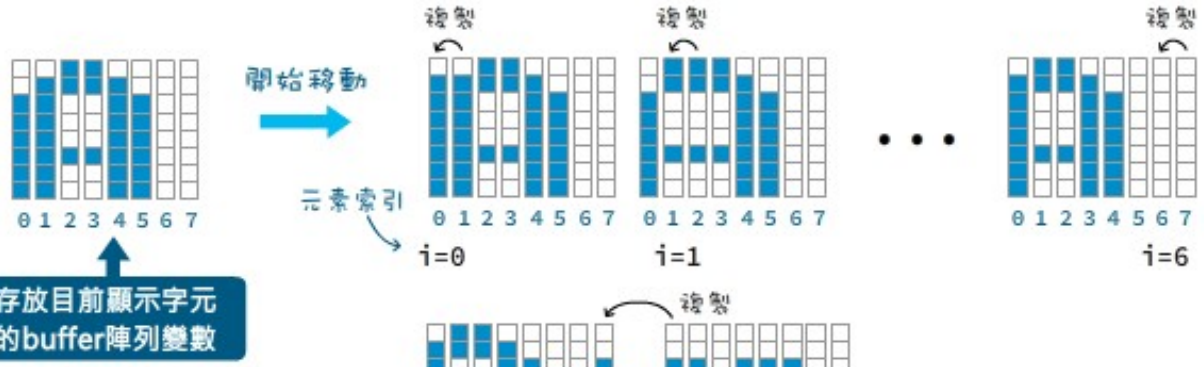
雙重迴圈程式先讀取第一張圖片裡的八行資料，再切換到下一張讀取：



LED矩陣跑馬燈

先把動畫字元暫存在變數，再逐行移動（複製）其中的數據，就能產生捲動效果。

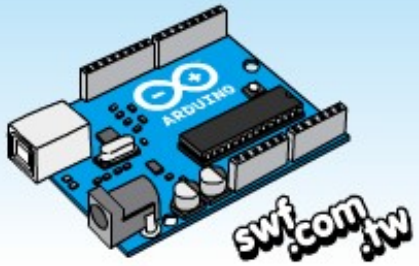





```
傳入下一個字元編號
void scroll(byte chr) {
  for (byte j=0; j<8; j++) {
    for (byte i=0; i<7; i++) {
      buffer[i] = buffer[i+1];
      max7219 (i + 1, buffer[i]);
    }
    buffer[7] = font[chr][j];
    max7219 (8, buffer[7]);
    delay(100);
  }
}
```

捲動目前的字元

逐行複製下個字元到目前的7行

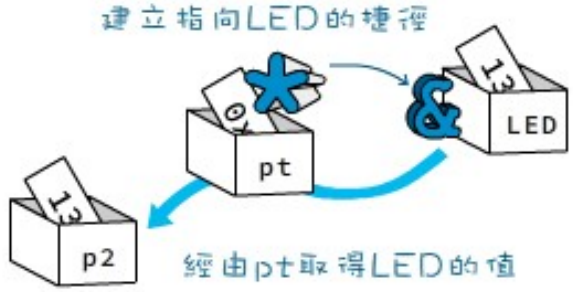


指標與「程式記憶體」

除了透過名稱，還可以透過記憶體位址存取變數資料。

```
byte LED = 13;
byte p1 = LED;
```


```
byte LED = 13;
byte *pt = &LED;
byte p2 = *pt;
```



在常數宣告敘述中加入PROGMEM關鍵字，可避免資料被複製到主記憶體。

```
byte fonts [256] [8] PROGMEM = {
  { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },
  { 0x7E, 0x81, 0x95, 0xB1, 0xB1, 0x95, 0x81, 0x7E },
      :
```

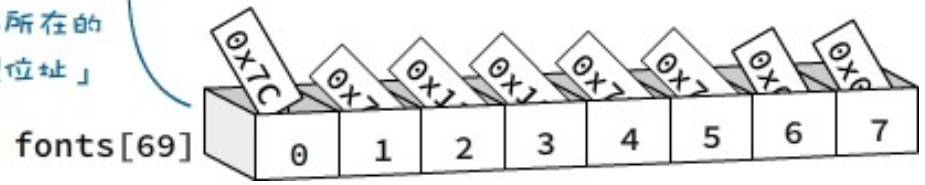
保存在程式記憶體

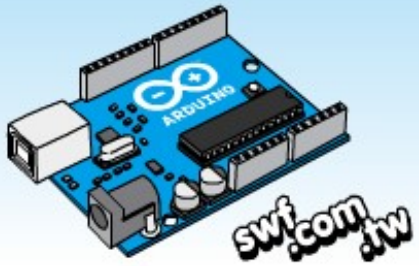
讀取程式記憶體的值，必須透過「指標」。

```
byte chr;
chr = pgm_read_byte (&fonts [69] [0]));
```

整個敘述代表「提取第69個字的第0」

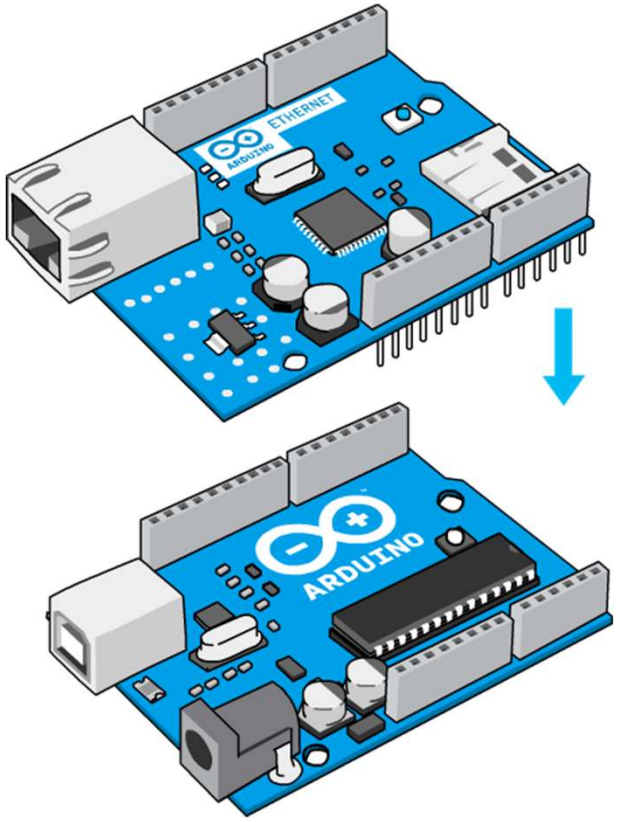
提取此資料所在的「記憶體位址」



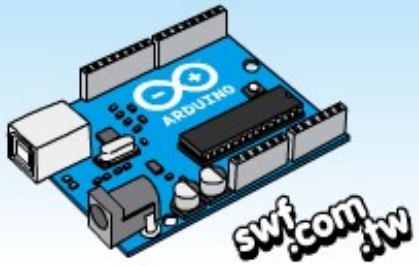


- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1

第九章 LCD顯示器 + 溫濕度感測+ 超音波感測器

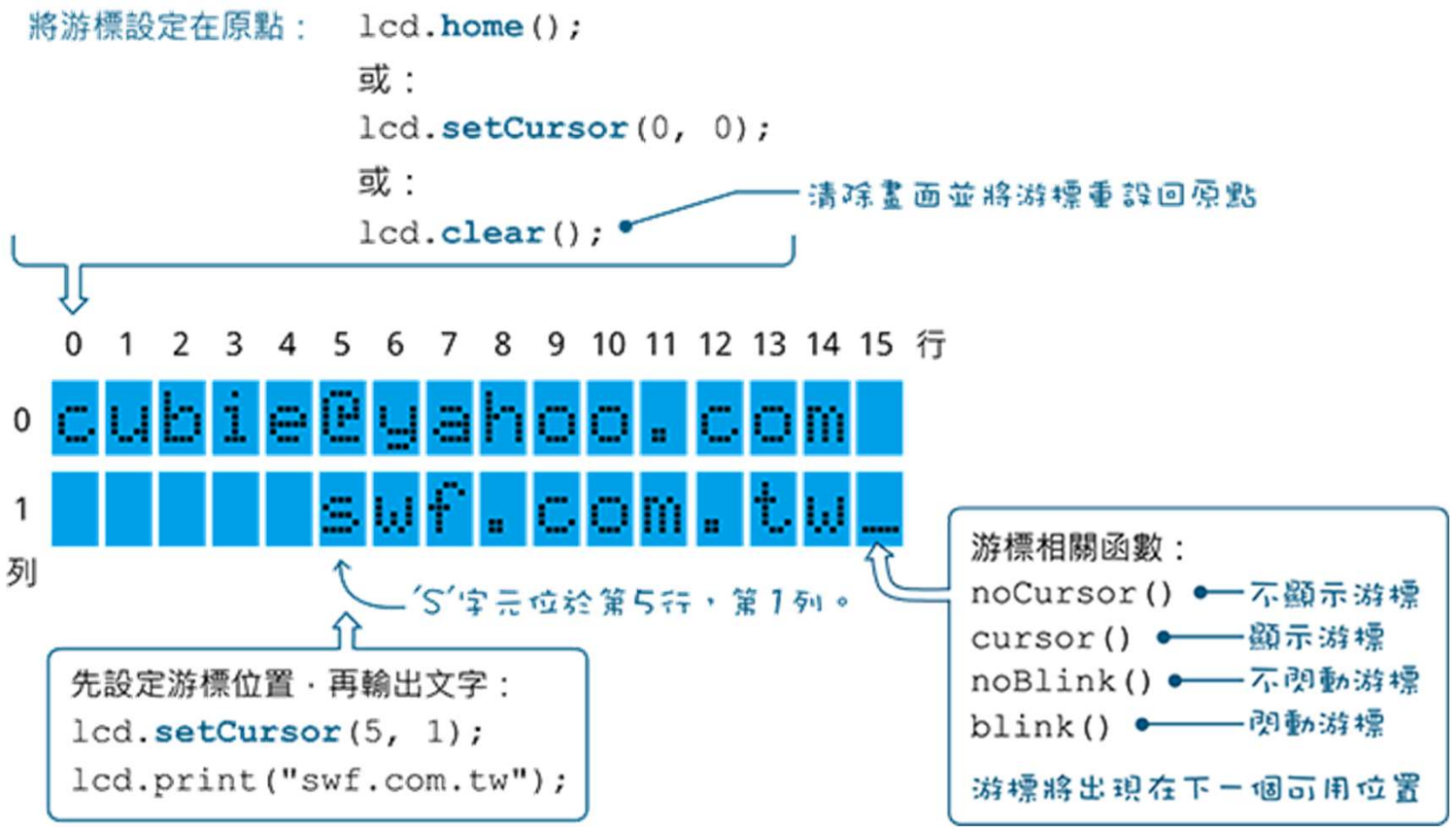


- 串列連接LCD顯示器
- 數位溫濕度感測器
- 超音波感測器

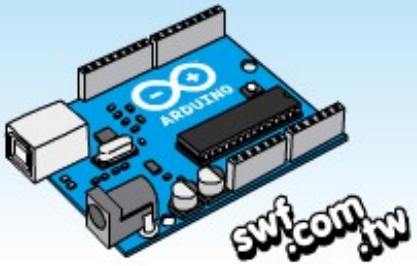


LCD顯示器程式說明

控制LCD模組的程式庫："**LiquidCrystal.h**"
LCD程式庫使用**print()函數**輸出文字。
插入文字的位置，由**游標（cursor）**決定。

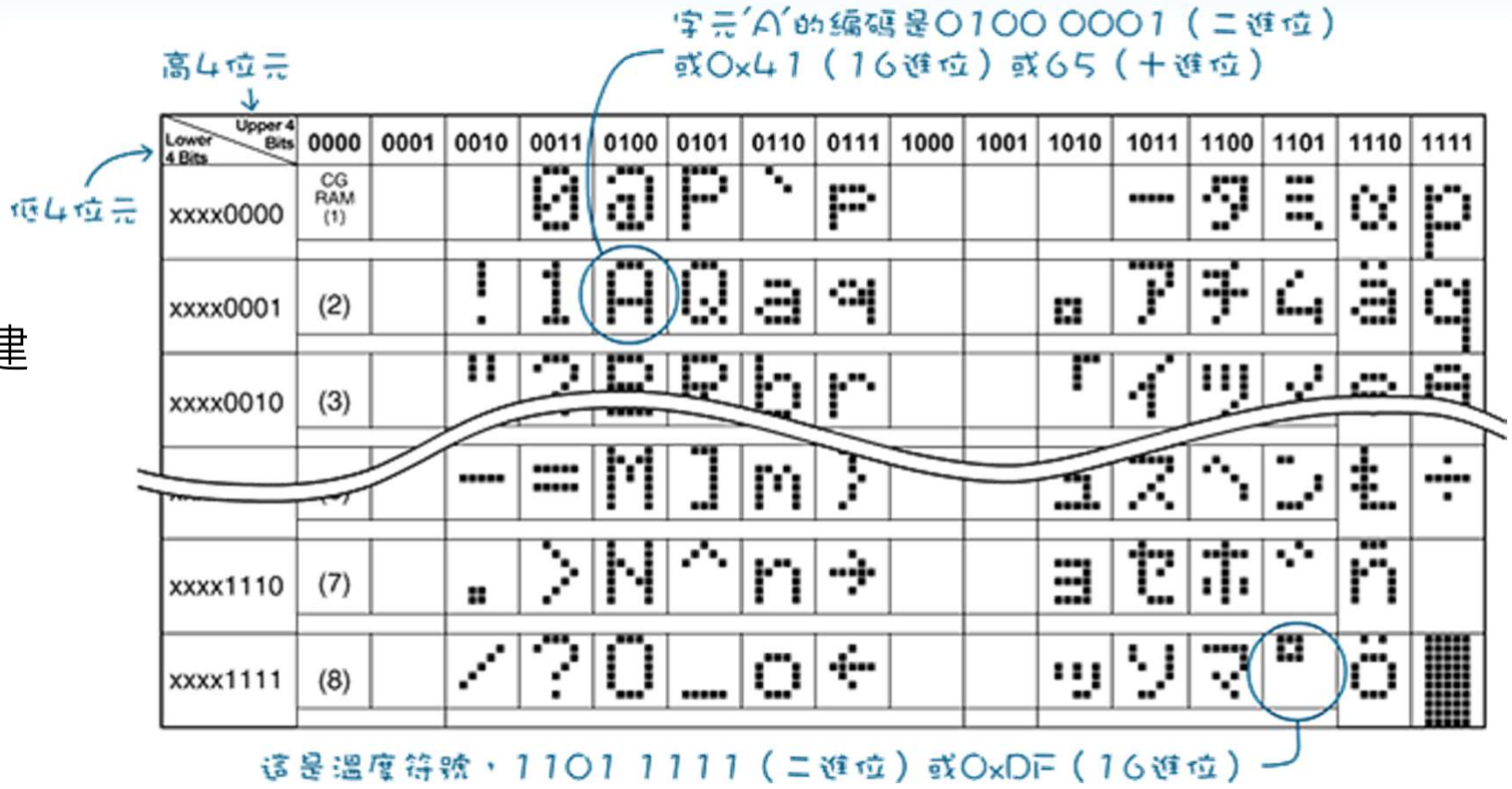


顯示文字暫存在控制晶片的記憶體裡，記憶體保存一列40個字。



顯示特殊符號與日文片假名

HD44780晶片的內建
字體儲存在CGROM



1100 1111
高位元 低位元

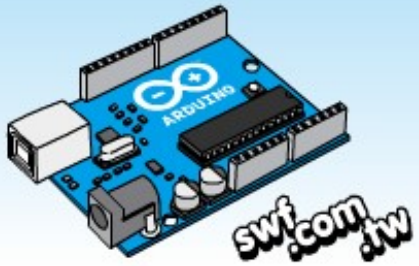
マ イ コ ン
0xCF 0xB2 0xBA 0xDD

```
char str[] = {'8', 'b', 'i', 't', ' ', 0xCF, 0xB2, 0xBA, 0xDD, 0};
                                       マ    イ    コ    ン
lcd.print(str);
```

顯示

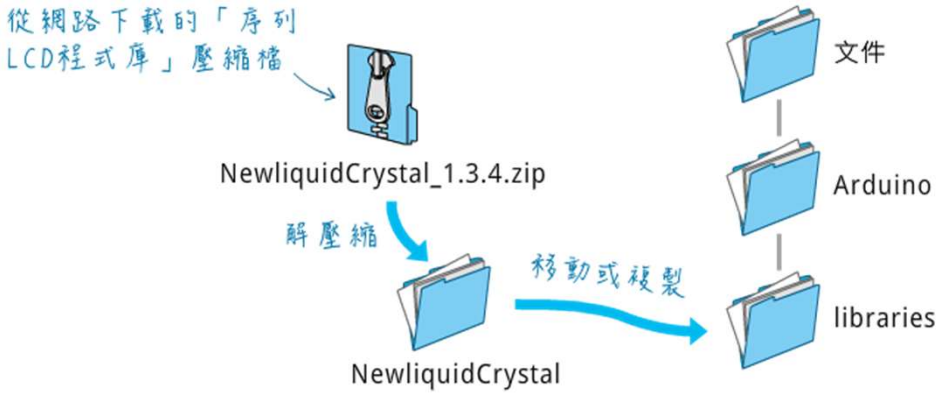
'8bit マイコン'

字串要用NULL結尾

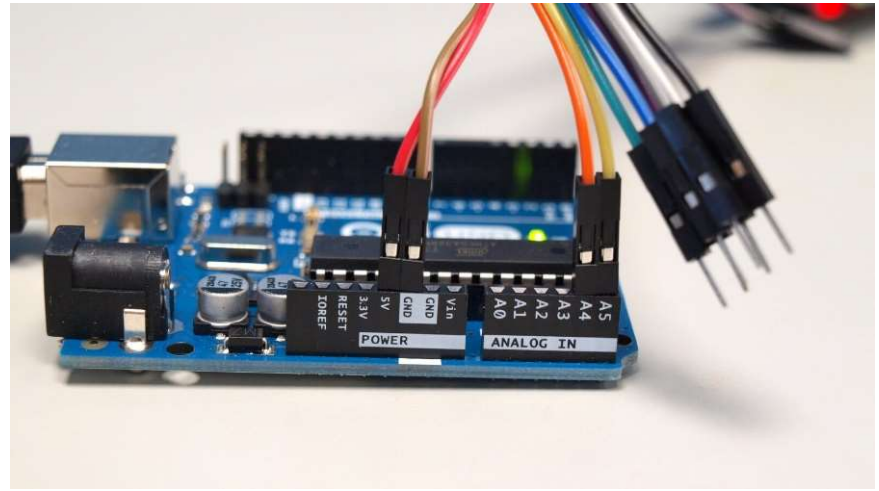


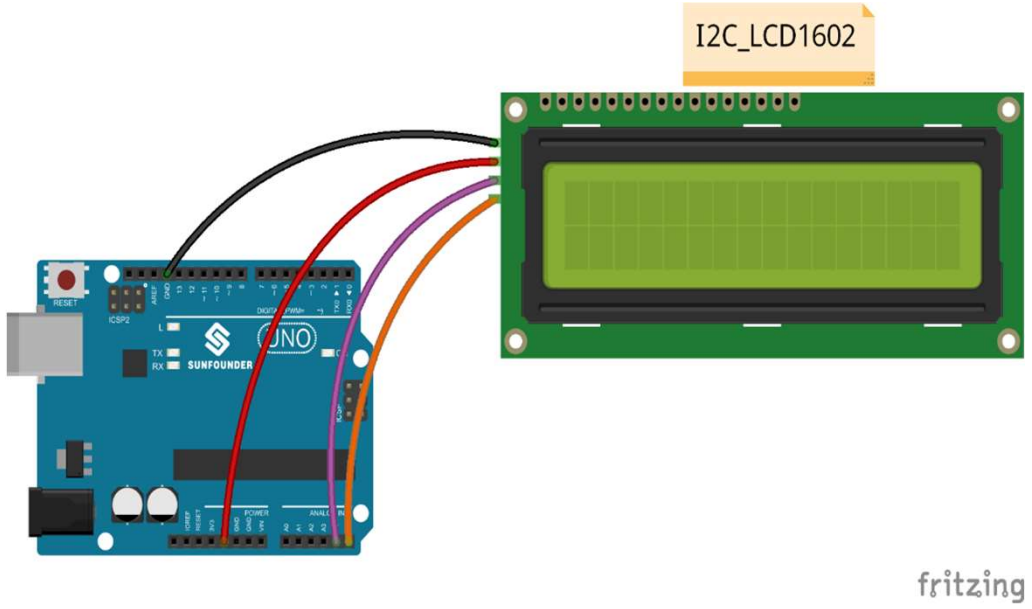
安裝與測試串連LCD模組的程式庫

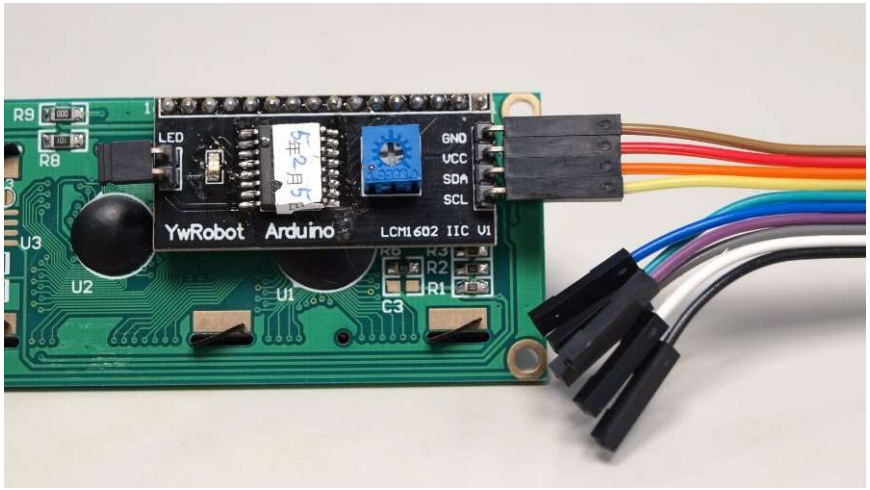
本單元採用兩線式LCD顯示模組的程式庫"New LiquidCrystal"，請先將原本的LiquidCrystal資料夾重新命名，再複製新的LiquidCrystal程式庫。

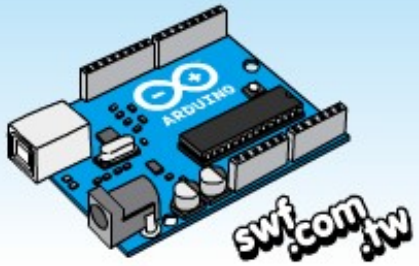


選擇主功能表的**檔案→範例→LiquidCrystal→HelloWorld_SR**，編譯並上傳測試。


IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
POWER
ANALOG IN

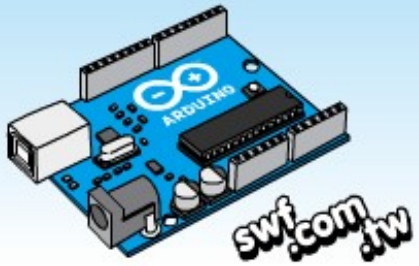

I2C_LCD1602
UNO
SUNFOUNDER
fritzing


R9
R8
LED
GND
VCC
SDA
SCL
YwRobot
Arduino
LCM1602 IIC V1
U3
U2
U1
C3
R3
R2
R1

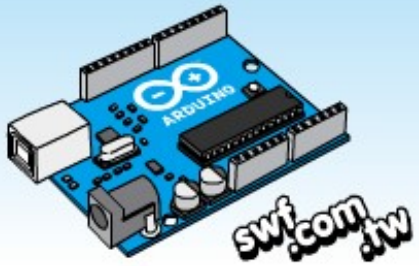


最前方定義完成LCD設定

```
#include <Wire.h>
#include <Wire.h>  // Arduino IDE 內建

// LCD I2C Library，從這裡可以下載：
// https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads-----只要下載NewliquidCrystal_1.3.4.zip 即可

#include <LiquidCrystal_I2C.h>  //網路下載

// Set the pins on the I2C chip used for LCD connections:
//設定 LCD I2C 位址 addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

##//若為16 pin LCD 則第一行改為 0x3F
```

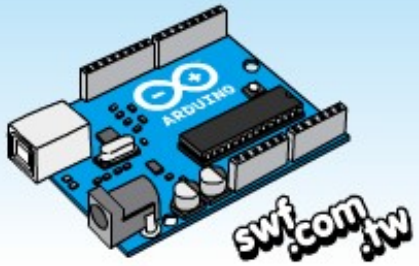



```
void setup() {
 Serial.begin(9600);  // 用於手動輸入文字
 lcd.begin(16, 2);      // 初始化 LCD，一行 16 的字元，共
2 行，預設開啟背光

  // 閃爍三次
  for(int i = 0; i < 3; i++) {
    lcd.backlight(); // 開啟背光
    delay(250);
    lcd.noBacklight(); // 關閉背光
    delay(250);
  }

 lcd.backlight();

  // 輸出初始化文字直接顯示
  lcd.setCursor(0, 0); // 設定游標位置在第一行行首
  lcd.print("Good day!");
  delay(1000);

}
```



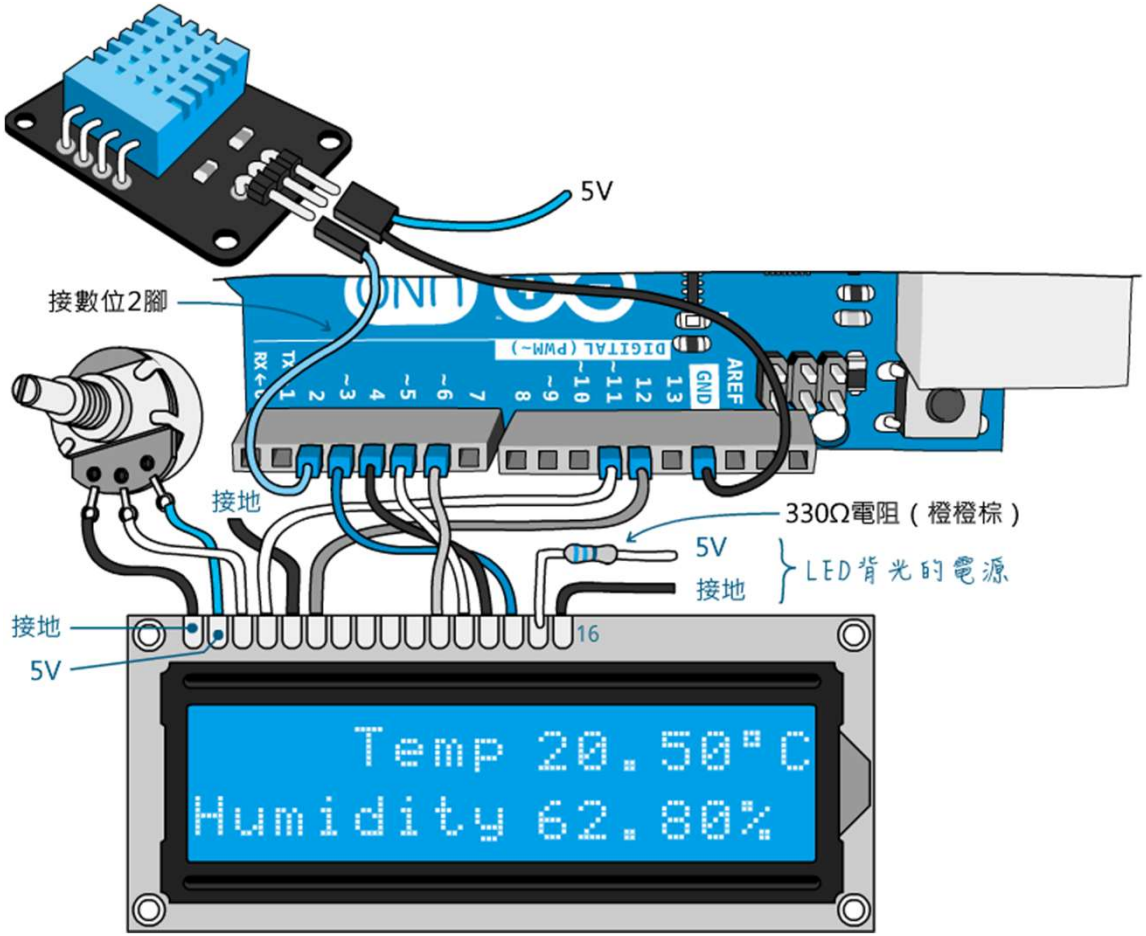
透過視窗輸入在LCD上顯示

```
#include <Wire.h>  // Arduino IDE 內建
// LCD I2C Library，從這裡可以下載：
// https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
#include <LiquidCrystal_I2C.h>

// Set the pins on the I2C chip used for LCD connections:
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  // 設定 LCD I2C 位址
 char x = 0;
void setup() {
  Serial.begin(9600);  // 用於手動輸入文字
  lcd.begin(16, 2);      // 初始化 LCD，一行 16 的字元，共 2 行，預設開啟背光

  // 閃爍三次
  for(int i = 0; i < 3; i++) {
    lcd.backlight(); // 開啟背光
    delay(250);
    lcd.noBacklight(); // 關閉背光
    delay(250);  }
  lcd.backlight();    // 輸出初始化文字
  lcd.setCursor(0, 0);   // 設定游標位置在第一行行首
}
void loop() {
   if(Serial.available()){
  char x = Serial.read();
  lcd.println(x);
  delay(1000); }
  lcd.clear();
}
```

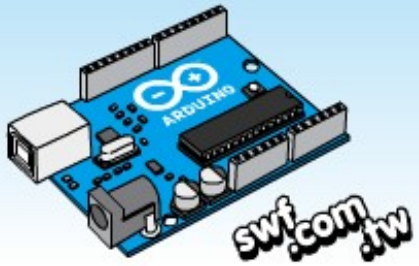


數位溫濕度感測器

DHT11是結合溫濕度感測器及訊號處理IC的感測模組

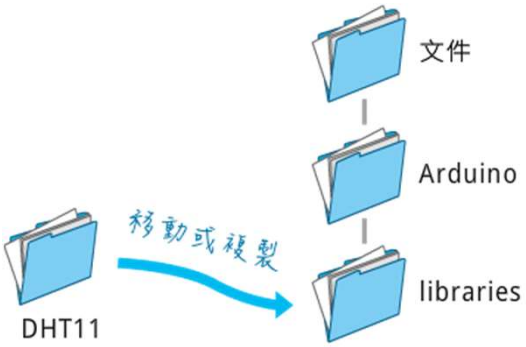


製作數位溫濕度顯示器



接收並顯示DHT11感測器溫濕度值

請將DHT11程式庫資料夾複製到**libraries**裡面。



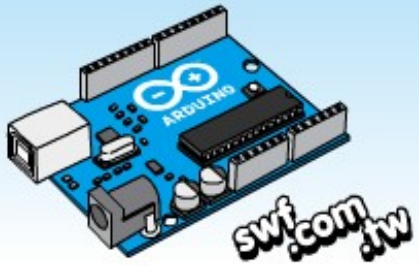
DHT11語法範例：

```
#include <dht11.h>

dht11 DHT11;

int chk = DHT11.read(2);

float temp = DHT11.temperature;

float humi = DHT11.humidity;
```

讀取數位2腳的DHT11資料，如果傳回0，代表讀取成功。

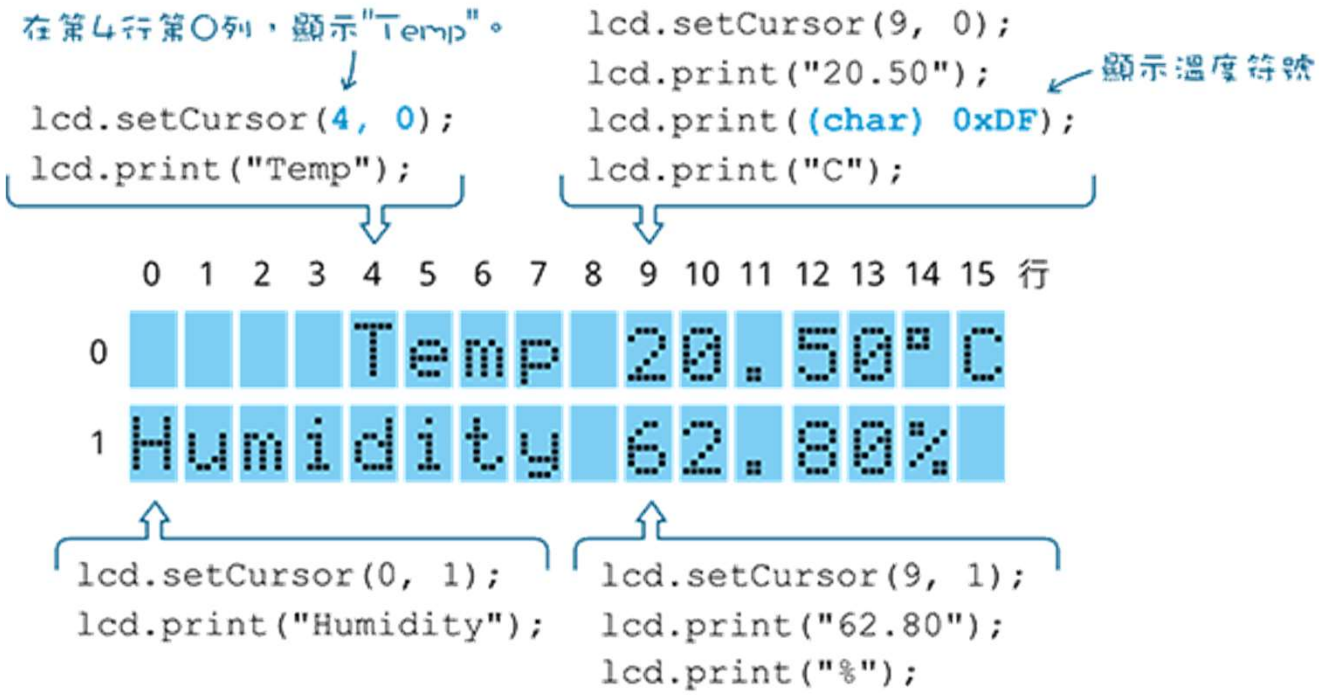
讀取帶小數點的溫度值

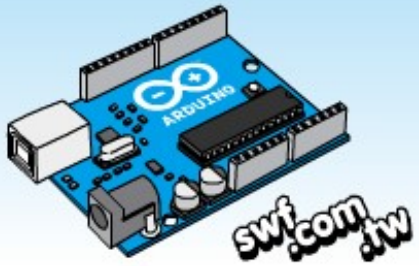
讀取帶小數點的濕度值



用LCD顯示器呈現溫濕度值

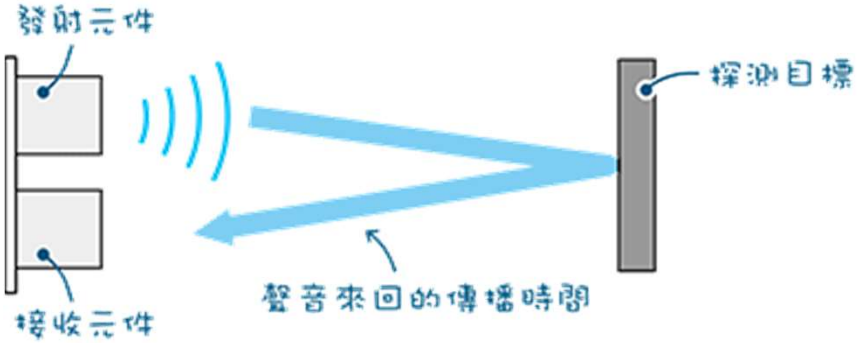
透過LCD程式庫的setCursor()函數，可任意排列文字：





認識超音波

高於人耳可聽見的最高頻率以上的聲波，稱為超音波。



從聲音的傳播速度和傳播時間，可求出距離，而物體的實際距離是傳播時間的一半，從此可求得1公分距離的聲波傳遞時間約為58μs（微秒）：

$$\text{距離} = 344\text{公尺/秒} \times \frac{\text{傳播時間}}{2}$$

聲波在室溫下，空氣中的傳播速度

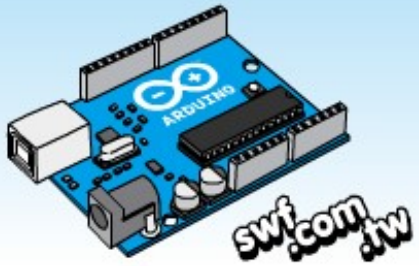
計算聲波前進1公分所需的時間

$$0.01\text{公尺} = \text{時間} \times 172\text{公尺/秒}$$

$$\text{時間} = \frac{0.01\text{公尺}}{172\text{公尺/秒}}$$

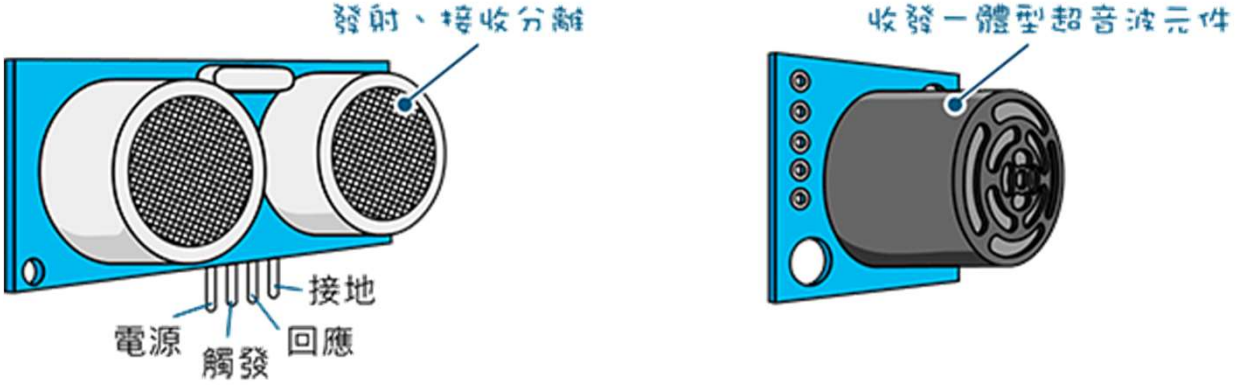
$$\text{時間} \approx 58.1 \times 10^{-6}\ \text{秒}$$

前進1公分所需的時間（單趟）：58.1 μs



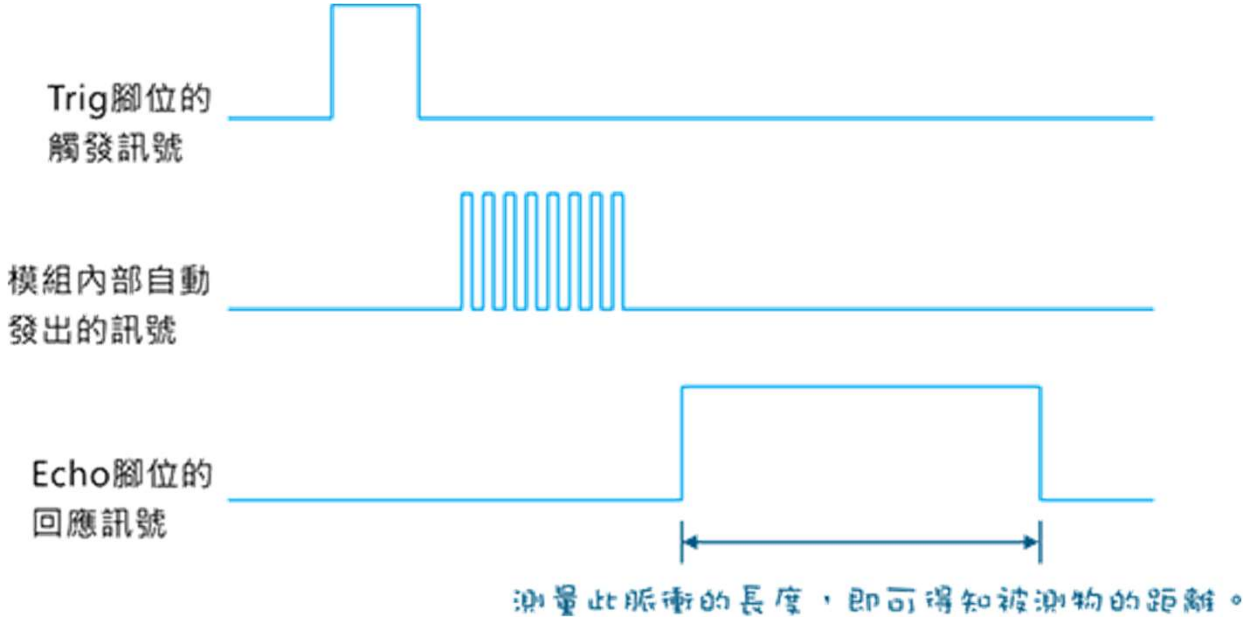
超音波感測器元件簡介

超音波感測器模組上面通常有兩個超音波元件，一個用於發射，一個用於接收；也有收發一體型。

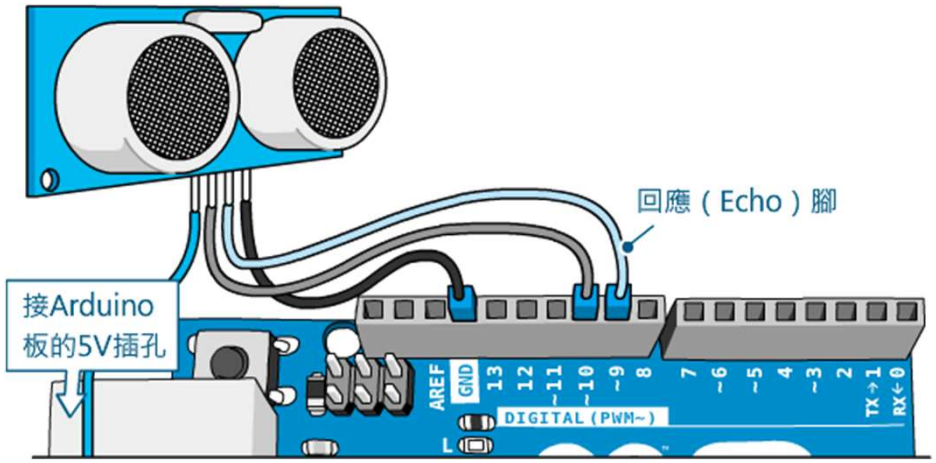


在「觸發」腳位輸入10微秒以上的高電位，即可發射超音波；

發射之後，接收到傳回的超音波之前，「回應」腳位將呈現高電位。

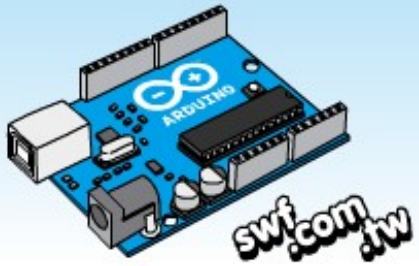


超音波數位量尺

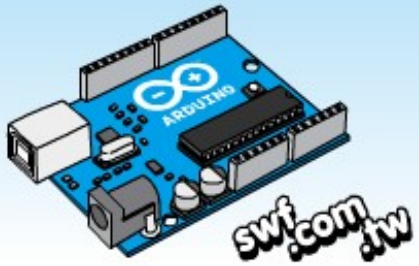


藉由測量脈衝時間長度的pulseIn()函數，可得知超音波的回應時間並藉此計算距離。

```
// 將第9腳的高脈衝時間存入變數d
unsigned long d = pulseIn(9, HIGH);
```




```
const int trig = 3;
const int echo = 4;
const int inter_time = 1000;
int time = 0;

void setup() {
  Serial.begin(9600);
  pinMode (trig, OUTPUT);
  pinMode (echo, INPUT);
}

void loop() {
  float duration, distance;
  digitalWrite(trig, HIGH);
  delayMicroseconds(1000);
  digitalWrite(trig, LOW);
  duration = pulseIn (echo, HIGH);
  distance = (duration/2)/29;  //換成公分
  Serial.print("Data:");
  Serial.print (time/1000); // 第幾次(間隔一秒)
  Serial.print(", d = ");
  Serial.print(distance);
  Serial.println(" cm");
  time = time + inter_time;
  delay(inter_time);
}
```



- 一分鐘回饋:
- https://goo.gl/forms/0C6jWOW5MTX9paos1