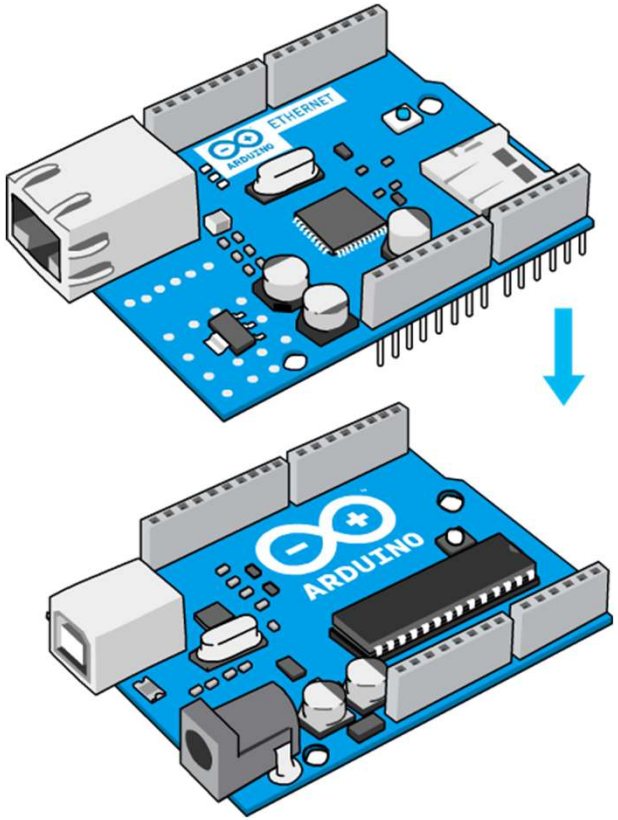
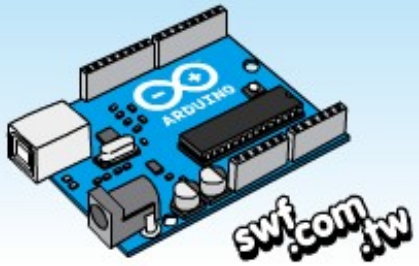
# 第五章 序列埠通訊



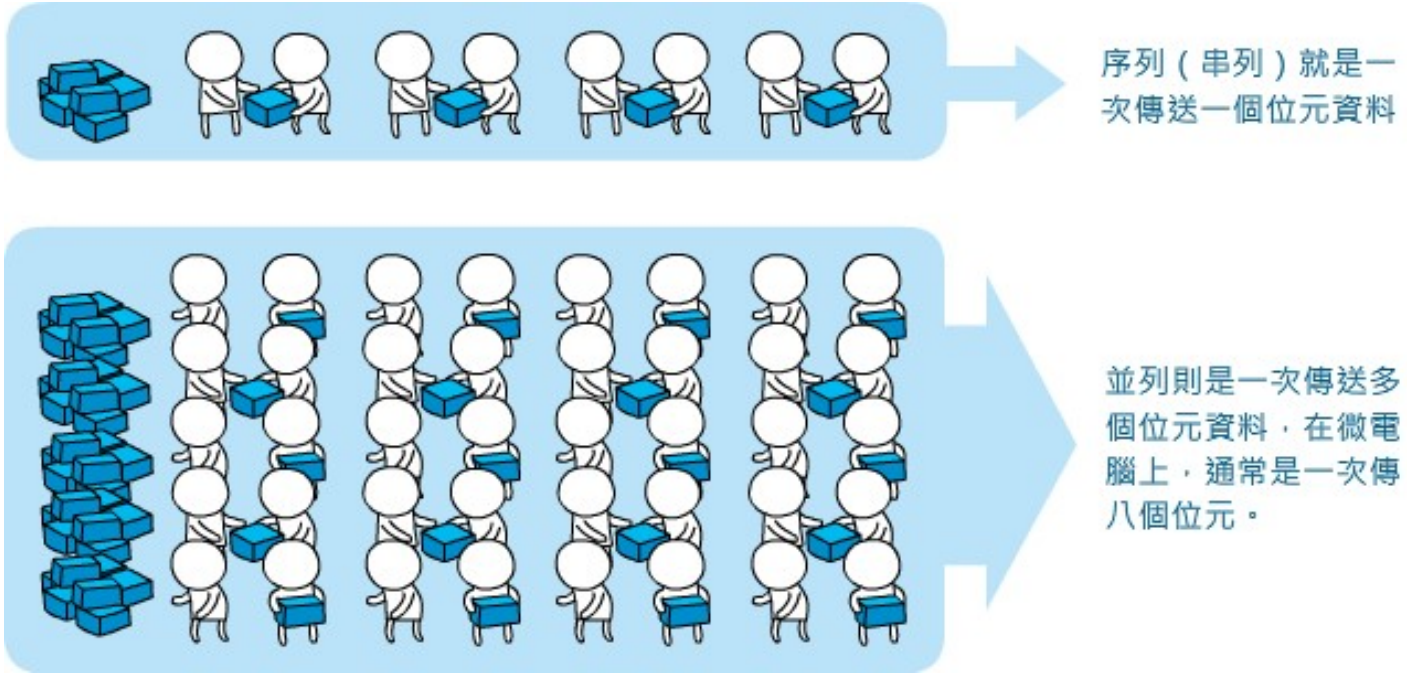

- 並列與序列通訊簡介
- 認識序列埠
- 認識字元資料類型
- 字串資料類型
- 從序列埠監控視窗觀察變數
- 從Arduino接收序列資料
- switch...case控制結構

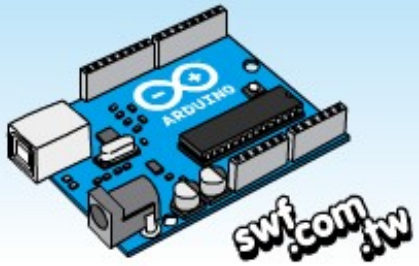



# 並列與序列通訊簡介

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。

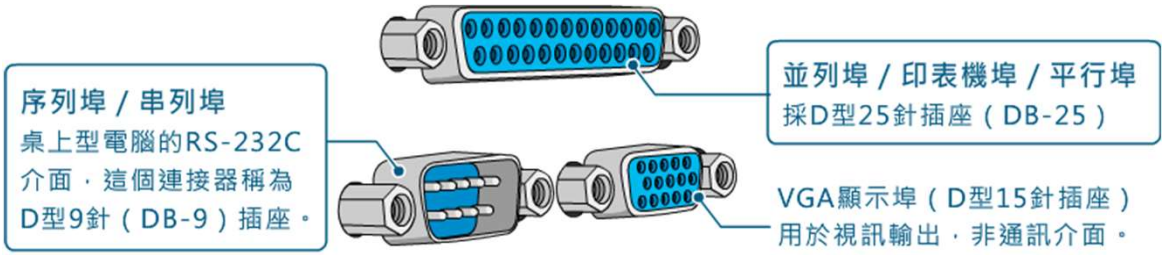
序列則是用少數（通常是兩條或三條）資料線，
將整批資料依序一個個送出或傳入。



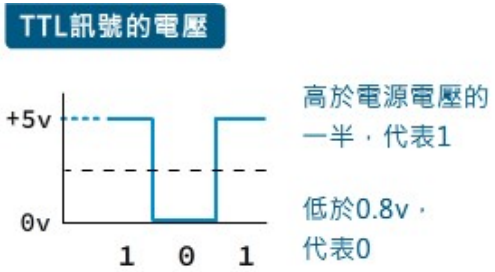


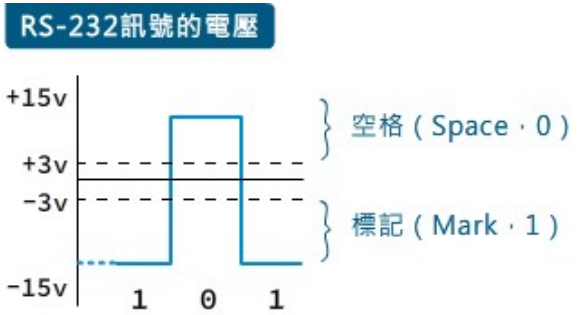

# 認識序列埠

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。



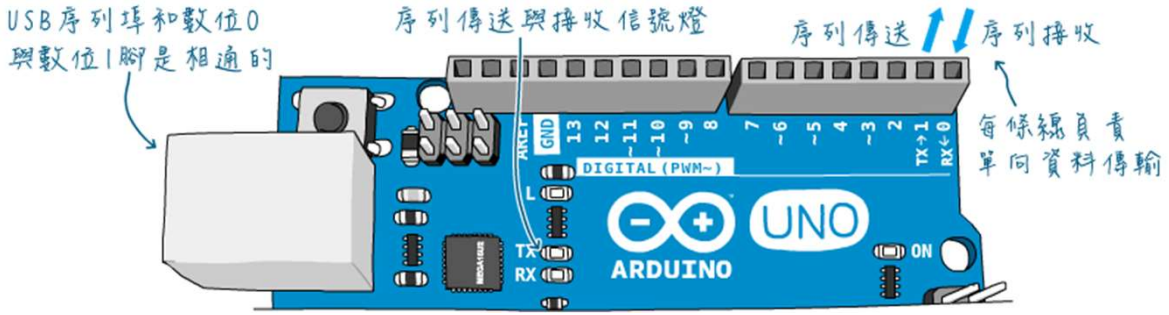

一般數位IC的0與1訊號的電壓準位，分別是0v和5v，稱為TTL或邏輯準位。

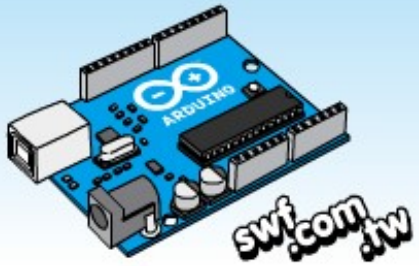



序列埠最重要的三個接腳：

- 數據傳送（TxD）
- 數據接收（RxD）
- 接地（GND）






# 認識字元資料類型

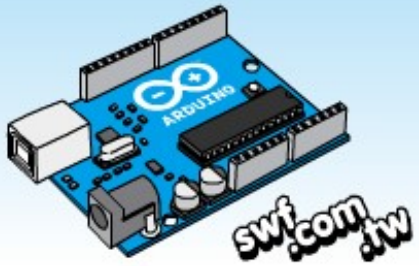
電腦把文字訊息分成字元（character）和字串（string）兩種資料類型。
字元類型的資料值要用單引號（'）刮起來。
每個字元都用一個唯一的數字碼表示。

```
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

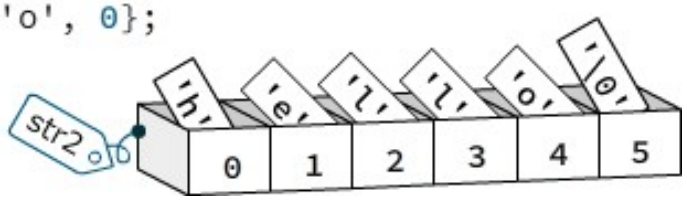
字串是一連串字元（char）的集合，也就是一段文字。
程式採用陣列存放字串，資料值前後一定要用雙引號刮起來。
每個字串都有一個Null字元（ASCII值為0）結尾。

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

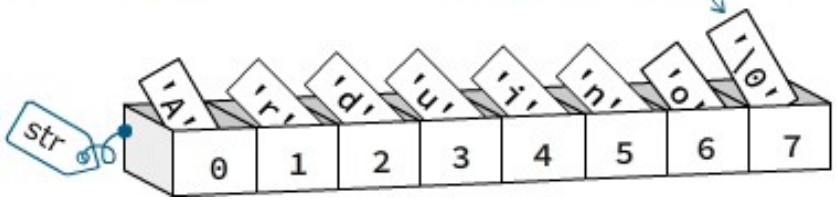
```
char str2[6] = {'h','e','l','l','o', 0};
```

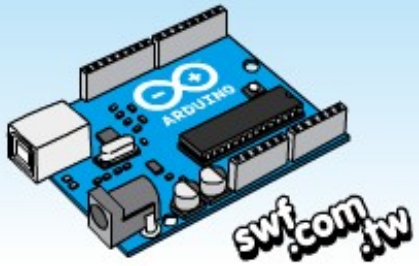



用雙引號刮起來

字串會自動加上一個Null字元結尾（ASCII值為0）

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

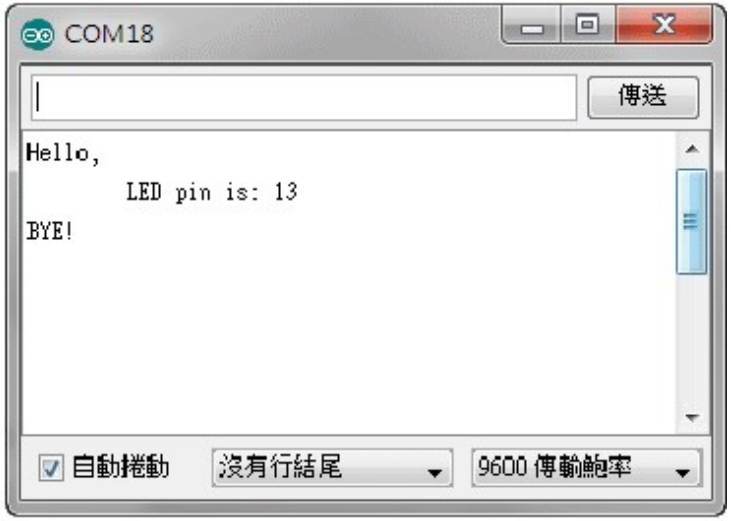


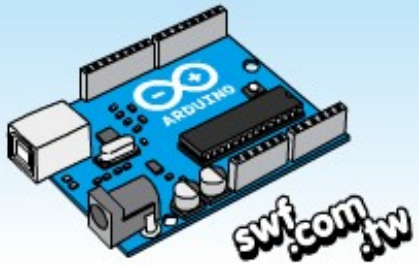



# 從序列埠監控視窗觀察變數

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() {
}
```

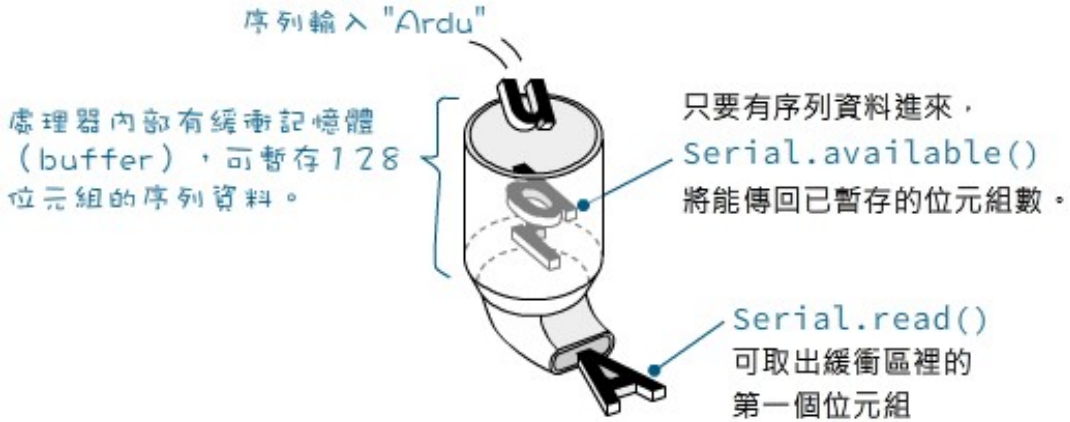





# 從Arduino接收序列資料

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。



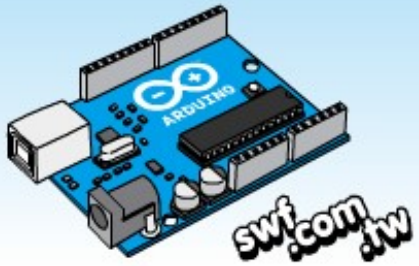


```
void loop() {
  if( Serial.available() ) {
    val = Serial.read();
    if (val == '1') {
      digitalWrite(LED, HIGH);
      Serial.print("LED ON");
    } else if (val == '0') {
      digitalWrite(LED, LOW);
      Serial.print("LED OFF");
    }
  }
}
```

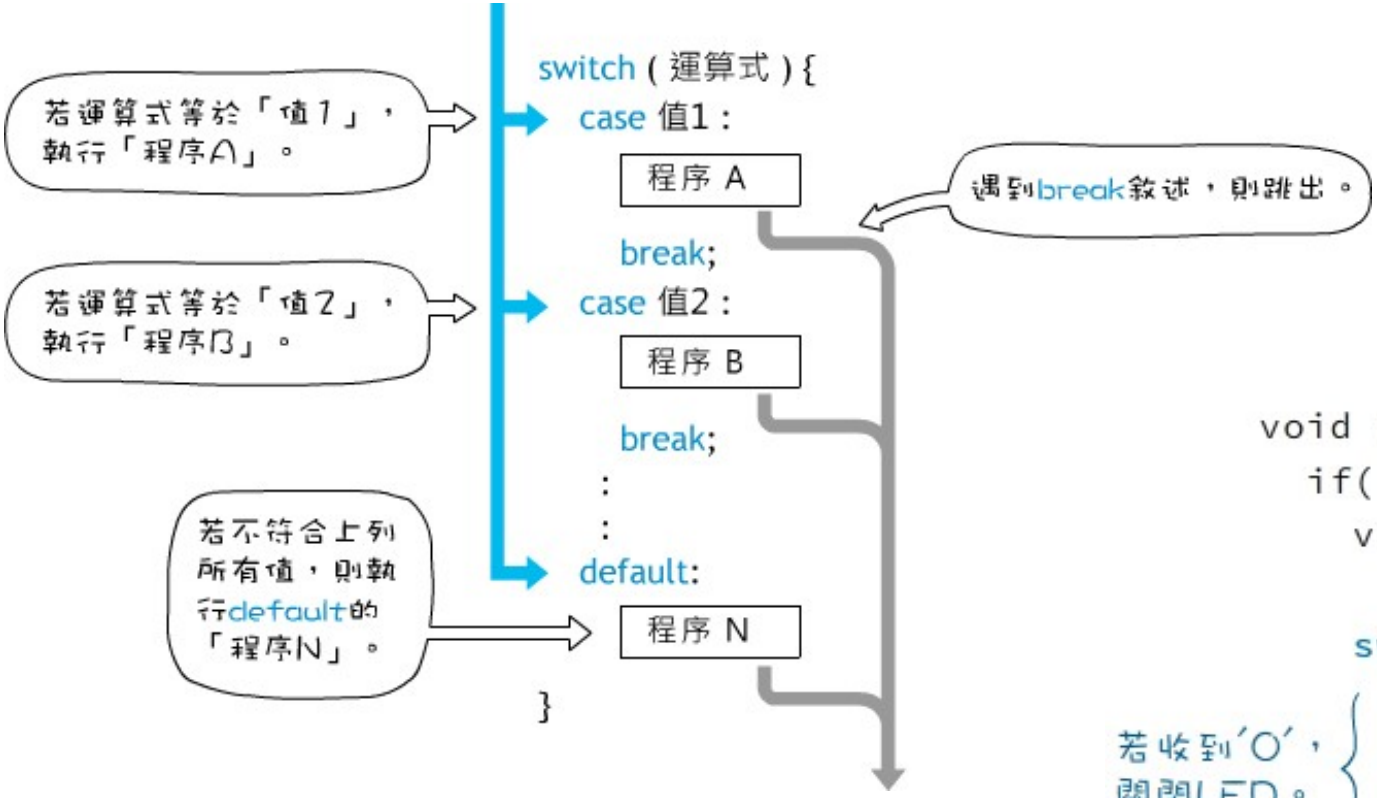
只要有收到字元，條件式的內容將被執行。

若收到'1'，點亮LED。

若收到'0'，關閉LED。



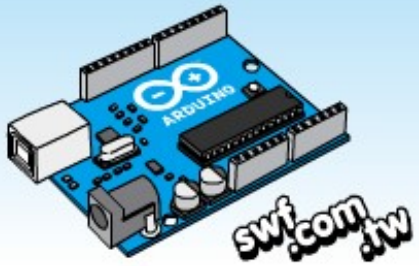

# switch...case控制結構




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

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

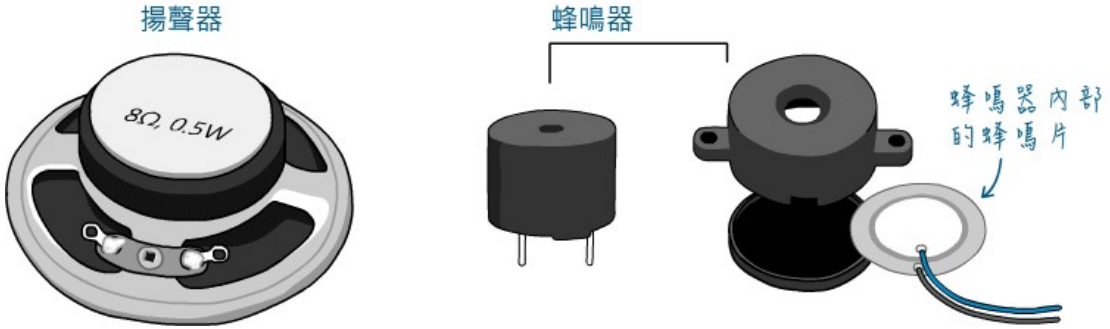
若收到'0'，關閉LED。

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。

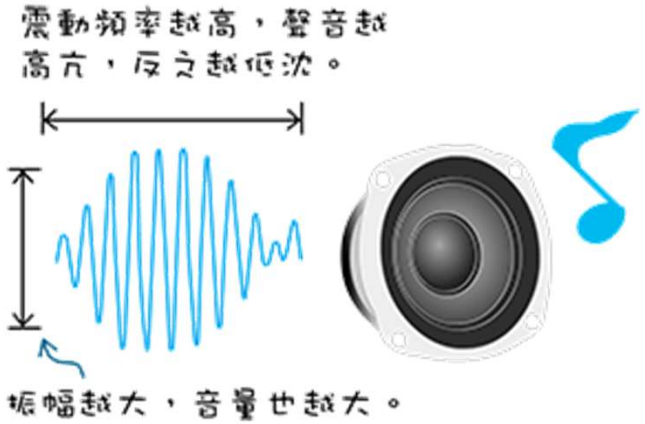


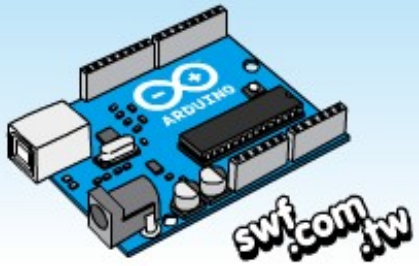

# 發音體和聲音

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）



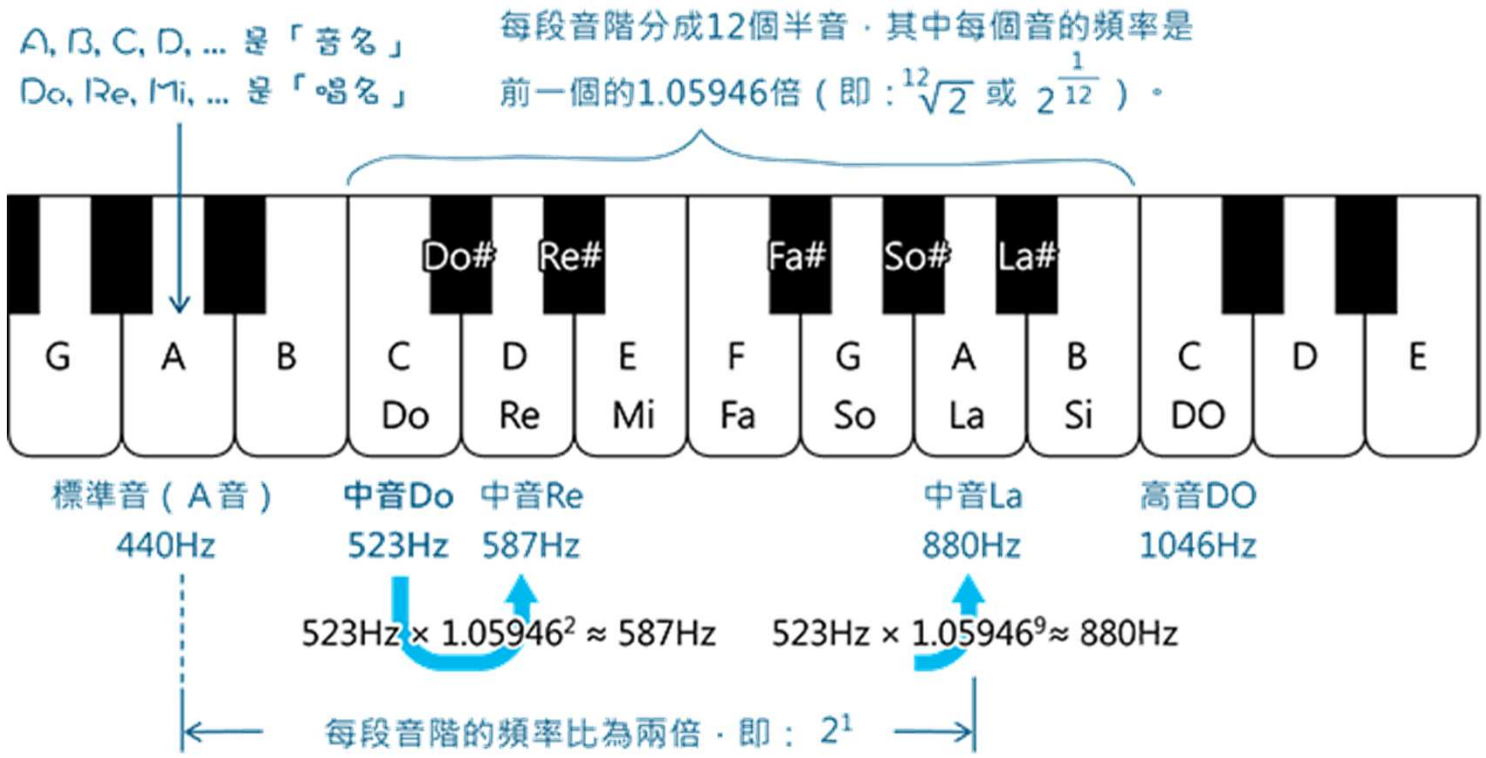

聲音是由震動產生, 其震動的頻率稱為**音頻**，音頻的範圍介於20Hz~200KHz之間，
普通人發出聲音的頻率範圍約為300Hz~3400Hz。

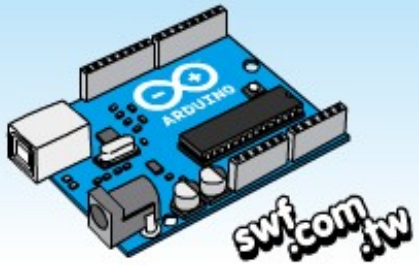



# 音階與節拍

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

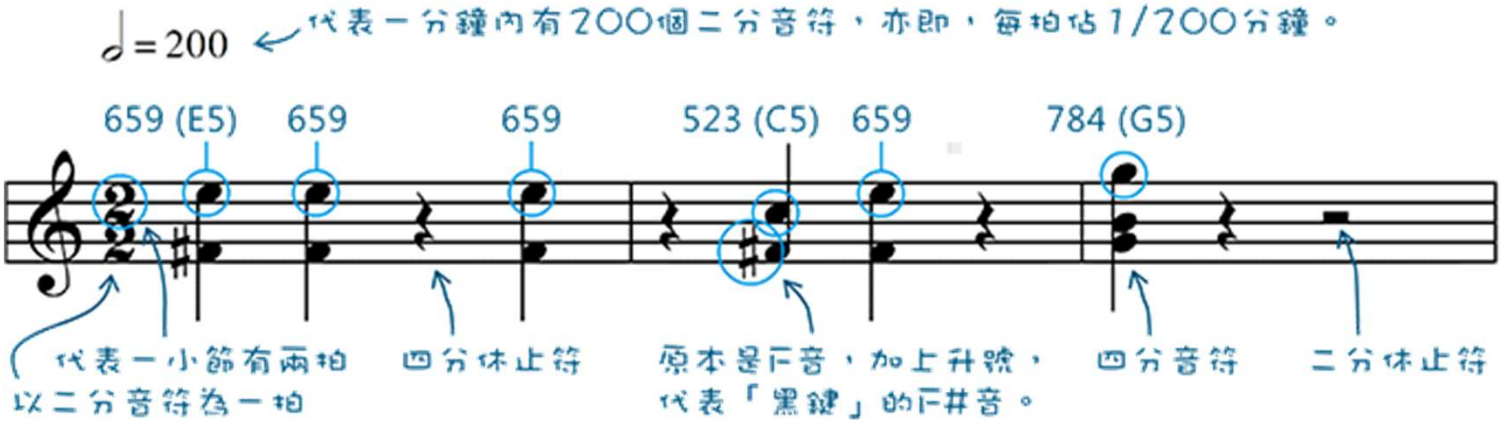
位於鍵盤中間的中央C音（Do）　　88鍵樂器的最高音

| | 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。

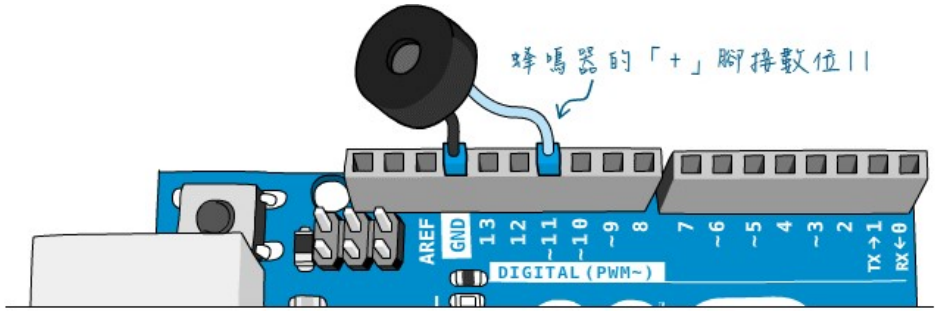


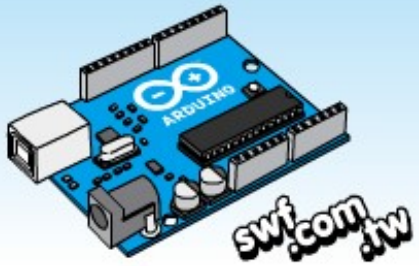

Arduino編輯器內建 “tone"（音調）程式庫，可以輸出指定頻率的聲音和持續時間。

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。

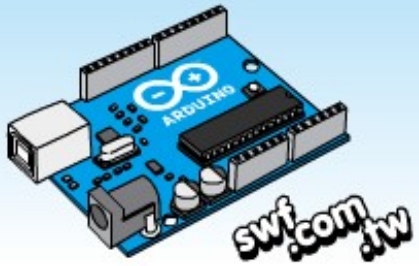
替換文字的巨集指令叫做#define。

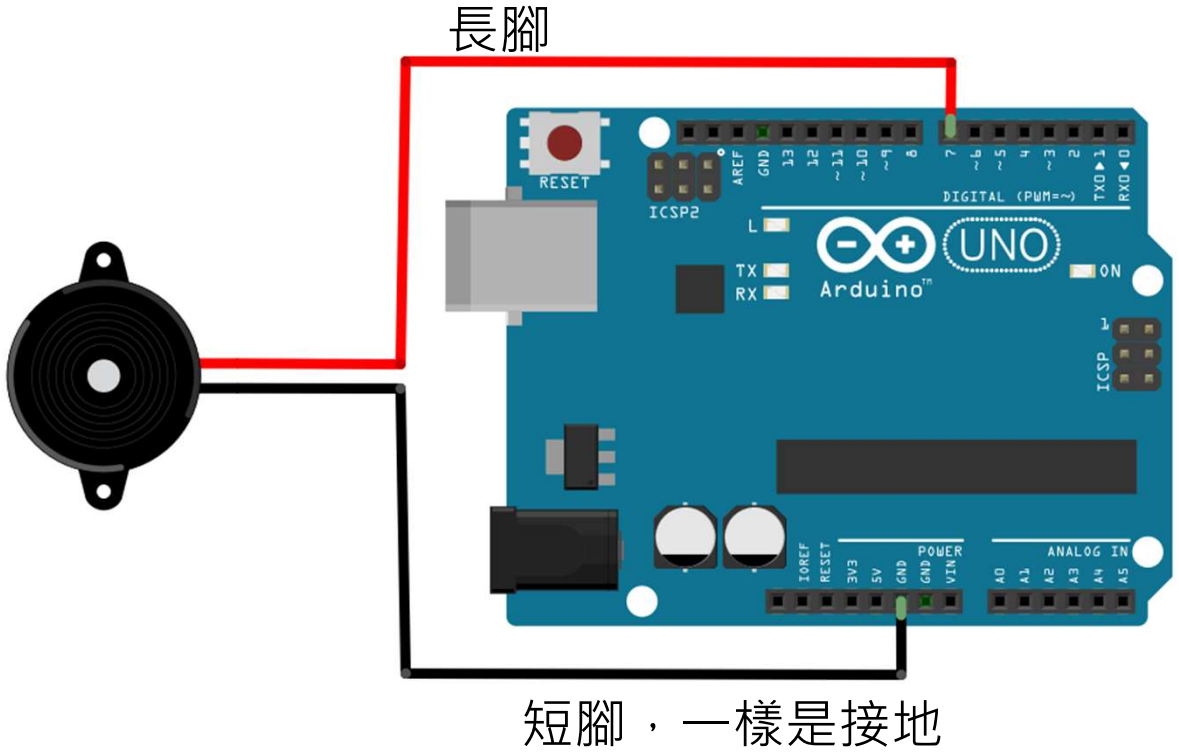
const（常數）代表宣告一個不可改變的資料值，而#define則用於定義置換值。

```
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);
}
```

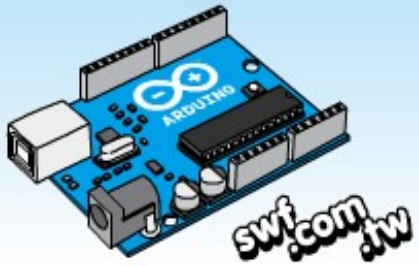
在程式開頭定義替換字（結尾不加分號）

所有E5都會被置換成659

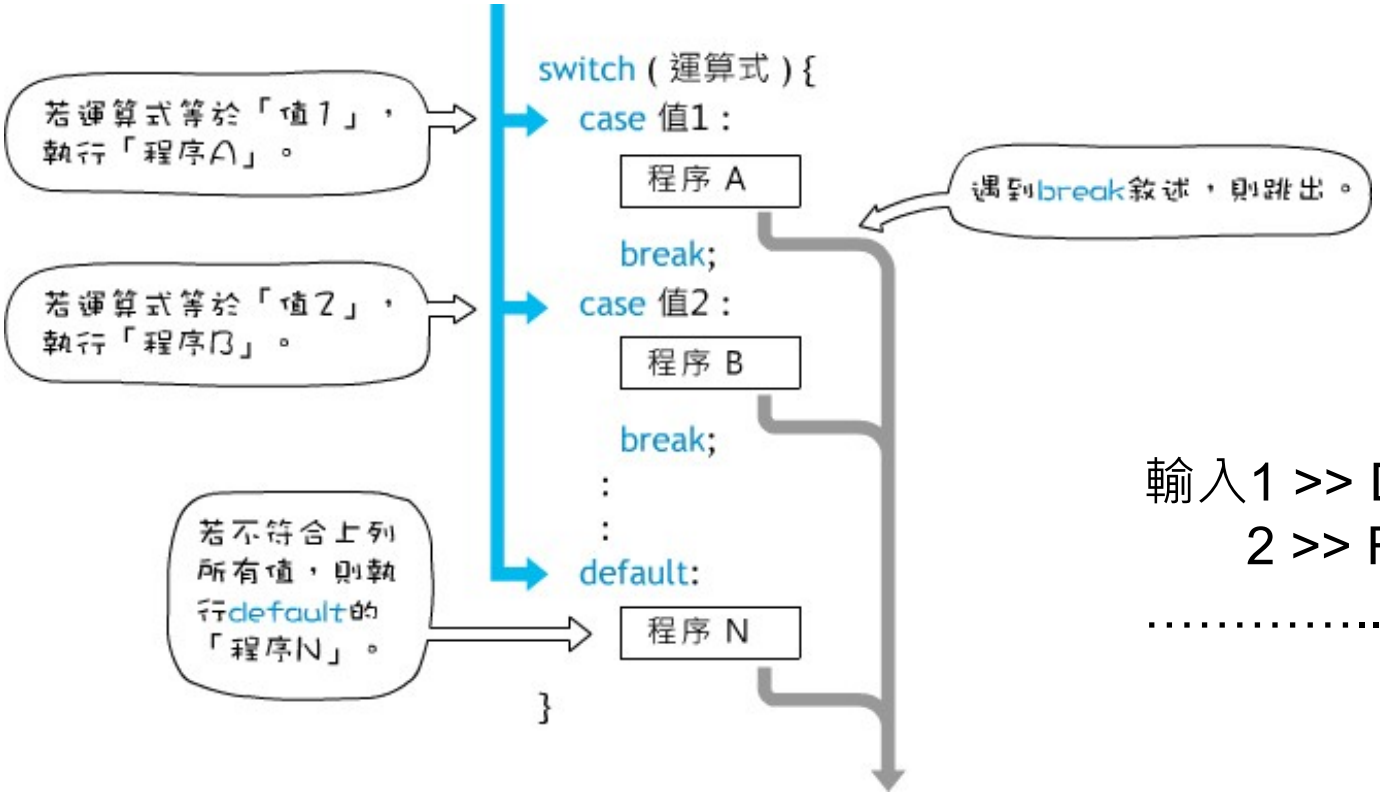



程式非常簡單，tone(7, 頻率) 就能讓蜂鳴器發出聲音了，要持續多久，就用delay()來控制，然後再用noTone(7)就能讓聲音停止了。

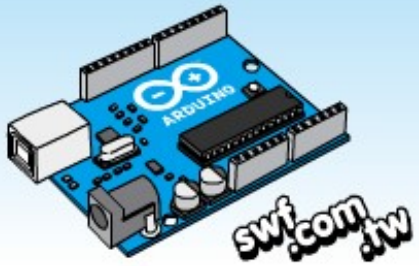



# switch...case控制結構




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