2

LED與開關控制

1 LED霹靂燈位移

2 LED亮度控制

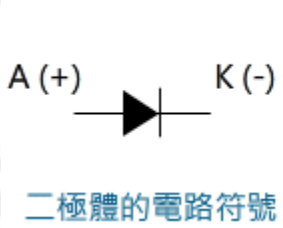
3 LED雨滴燈

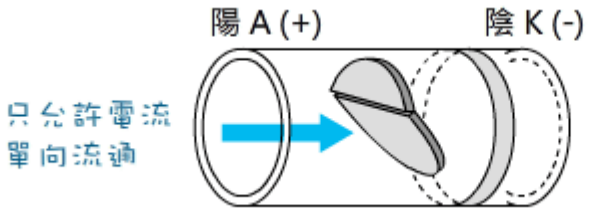
講師 張傑帆 Chang, Jie-Fan

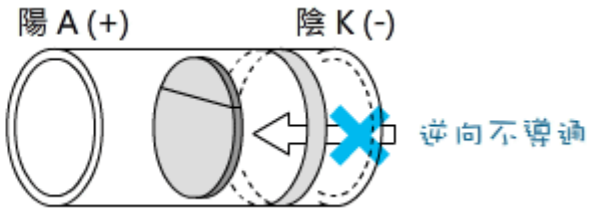
使用 Arduino 控制發光的 LED 元件，Arduino 只能按照一個程式設定好的順序不斷地迴圈執行這些控制操作，如果我們想在程式運行過程中，人為地控制程式執行的流程（電腦領域稱之為互動），那麼就需要添加一些互動元件，按鍵和旋鈕式可變電阻就是兩種最基本的互動元件。

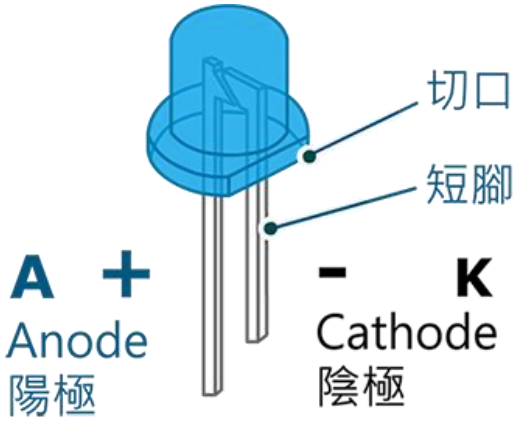
二極體和LED

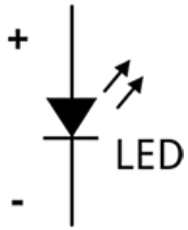
二極體是一種單向導通的半導體元件，其接腳有區分極性。



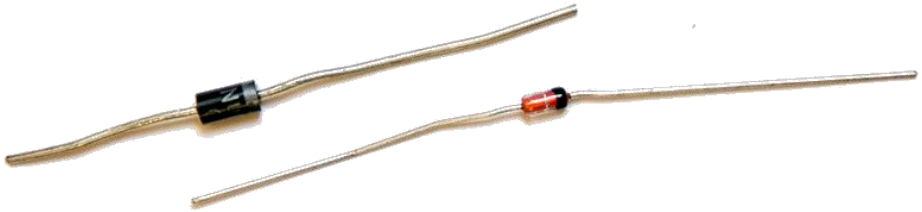








LED同樣是單向導通元件，若接反了它不會亮
- 長腳接正極（+）
- 短腳接負極（-，或接地）



認識發光二極體(LED)

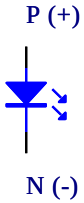
- 發光二極體（light emitter diode，簡記LED）為PN二極體的一種，LED長腳為P型（positive，正），又稱為陽極（anode，簡記A），短腳為N型（negative，負），又稱為陰極（cathode，簡記K）。

- 使用指針式三用電表測量其接腳，首先將三用電表切換至R×10檔，再將其黑棒（內接電池正端）接LED的P型，紅棒（內接電池負端）接LED的N型，LED將會因順向偏壓（P接正，N接負）而導通發亮。

- 當LED外加順向偏壓而導通發亮時，其導通電壓約在1.5V~2.0V之間，而其發光強度與順向電流成正比

- 須串聯一電阻以限制流過LED的電流，一般設計順向電流約在10mA~40mA之間，如果工作電源電壓為5V時，使用的限流電阻為220Ω或330Ω即可。

(a)LED 元件



(b)符號

(b)外觀

認識發光二極體(LED)

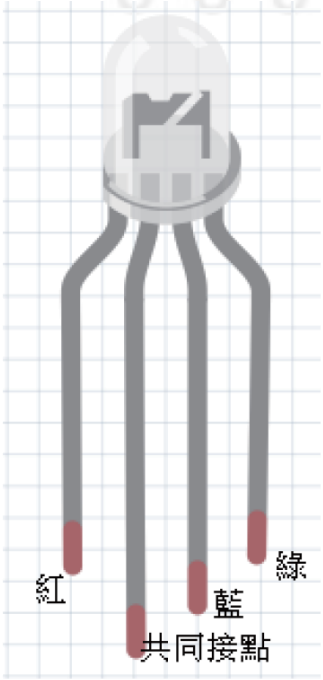
- LED發光的顏色與製造材料有關，而與工作電壓大小無關。
- 製造LED的主要半導體材料為砷化鎵（GaAs）、砷磷化鎵（GaAsP）或磷化鎵（GaP），因其使用材料及能階高低的不同，會發出不同波長的光。常見的LED顏色有紅色、綠色、黃色、白色、藍色等。
- 人眼所能看見的光稱為可見光，可見光波長如表11-1所示，波長介於400nm~780nm之間，其中紅光波長較長，其次為綠光，較短的為藍光，在可見光波長區間之外的為不可見光，例如紅外光及紫外光(UV)等。

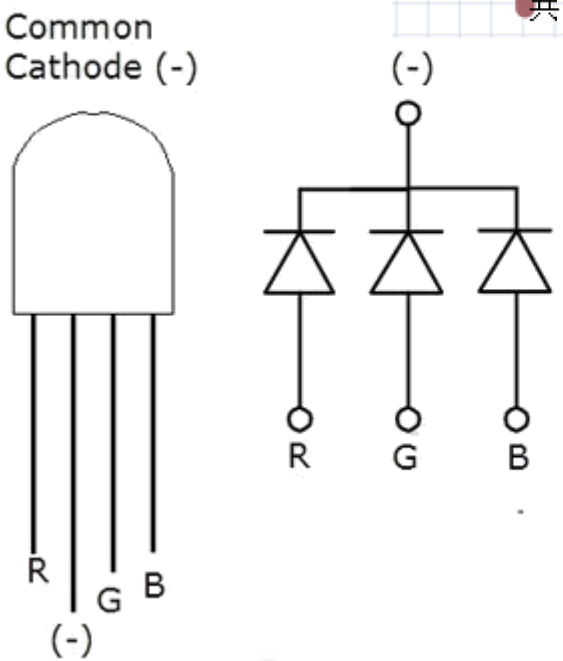
表4-1　　可見光波長

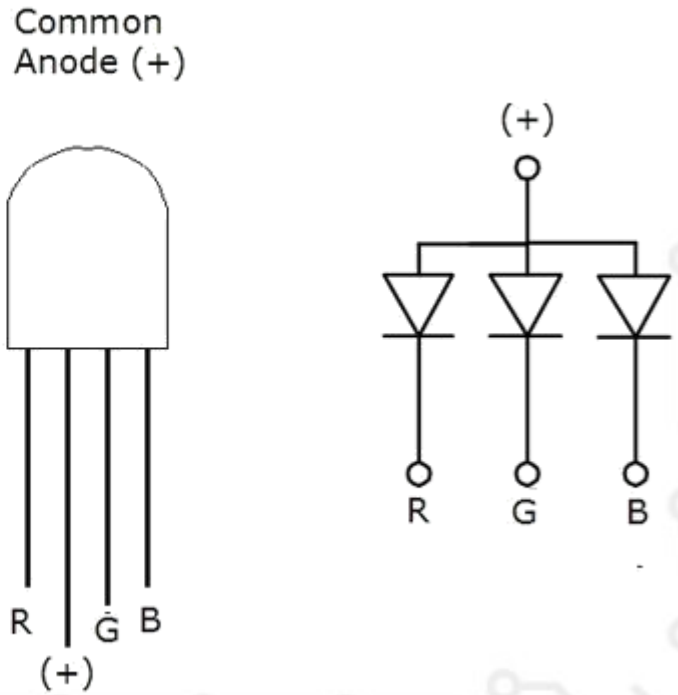
波長（nm）	顏色	區分
400 ～445	紫色	可見光
445 ～500	藍色	可見光
500 ～575	綠色	可見光
575 ～585	黃色	可見光
585 ～620	橙色	可見光
620 ～740	紅色	可見光

多色LED燈

- 分兩種
  - 共陰極
  - 共陽極







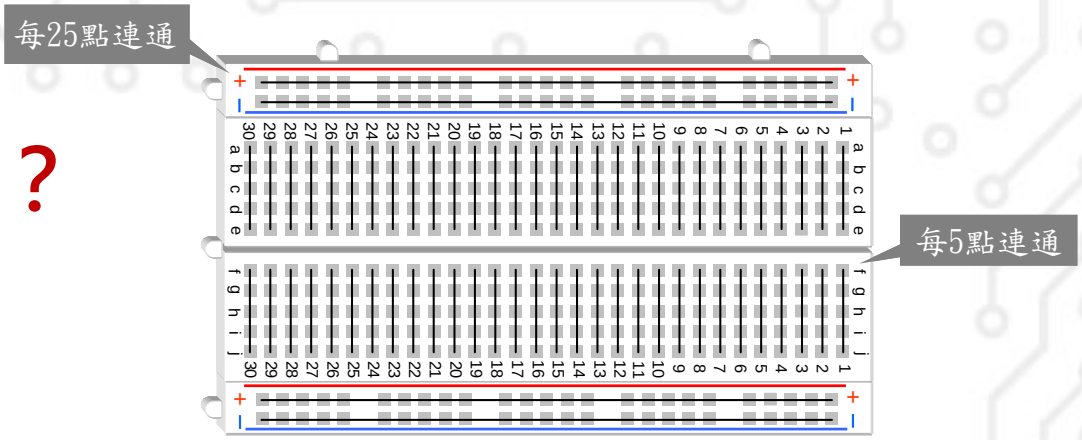
實作練習

一個 **LED** 閃爍實習

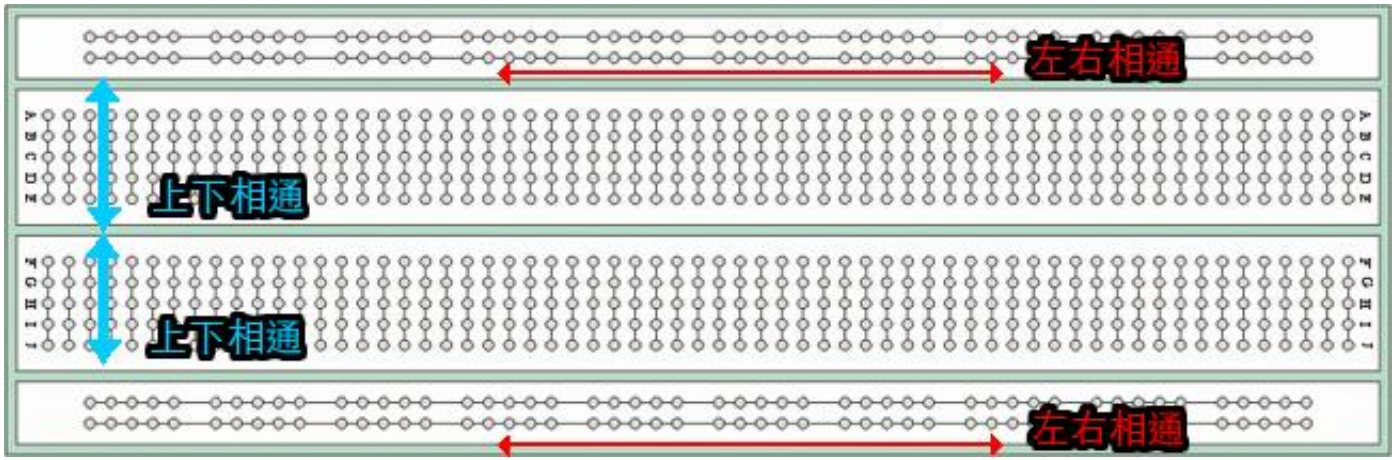
- 功能說明：

利用 Arduino 板控制一個 LED 閃爍亮 1 秒、暗 1 秒。在 Arduino UNO 板中的數位接腳有編號 0~13 等 14 支腳，其中編號 0 與 1（標記為 RX 與 TX）是用於 USB 串列通訊，宜避免當做其它用途使用，接腳不夠用時，也可以使用標記 A0~A5 的類比輸入當成數位接腳編號 14~19 來使用。本例使用 Arduino 板第 13 腳內接橙色 LED 如圖 4-3 標示" L "位置，如您是從官方網站上購買 Arduino 板，則微控制器 ATmega328P 已事先下載範例程式 Blink.ino，功能與本例相同。

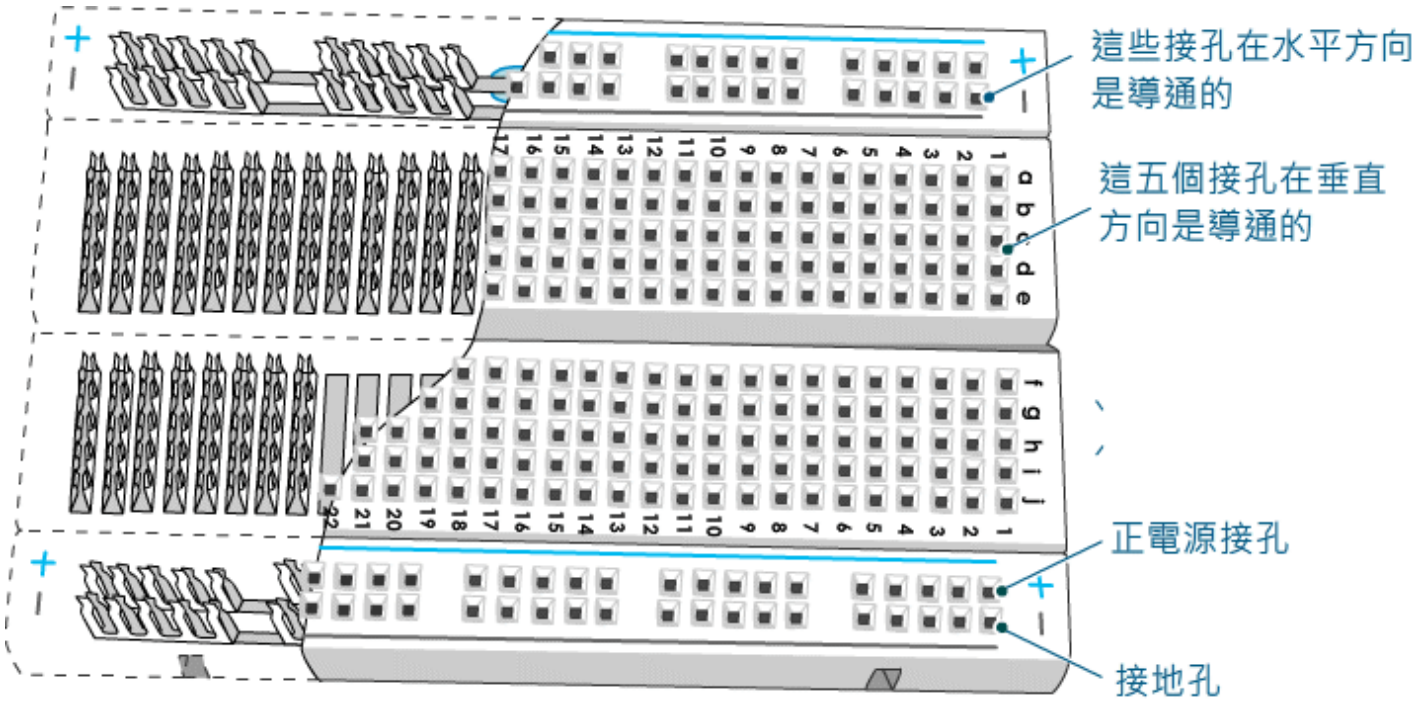
麵包板怎麼使用？



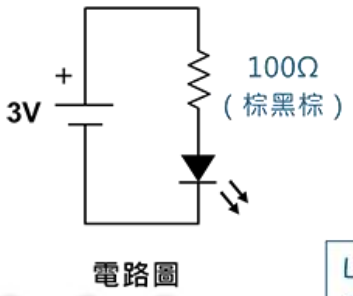
- 免焊萬用電路板（solderless breadboard）
  俗稱麵包板，內部是由一些長條形的磷青銅片組成
- 麵包板是一種不需焊接，可快速拆裝、組合電子電路的用具
- 普遍用於電子電路實驗

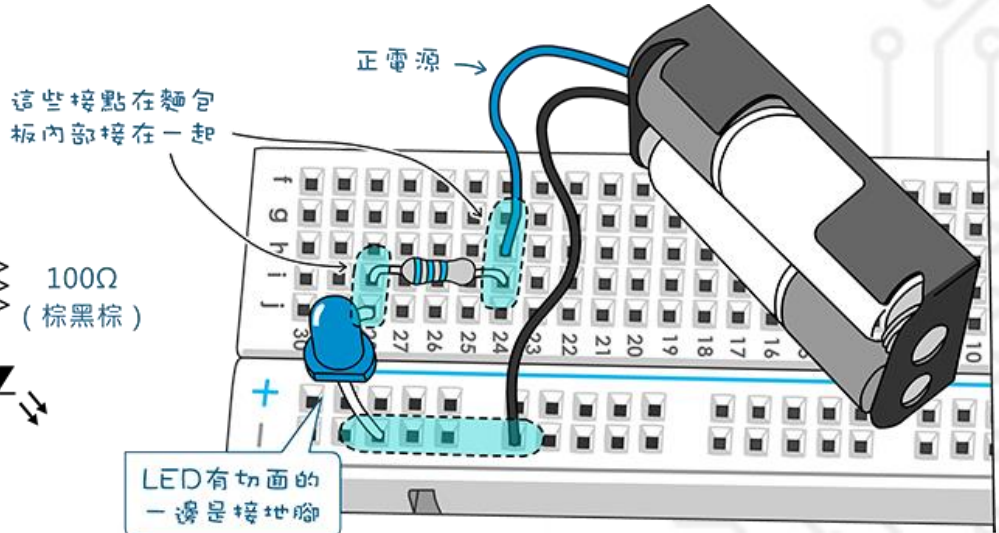


麵包板



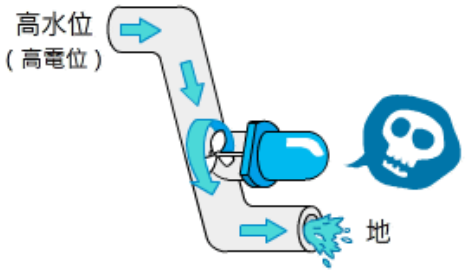
左邊是LED電路圖，以及在麵包板上組裝的樣子。

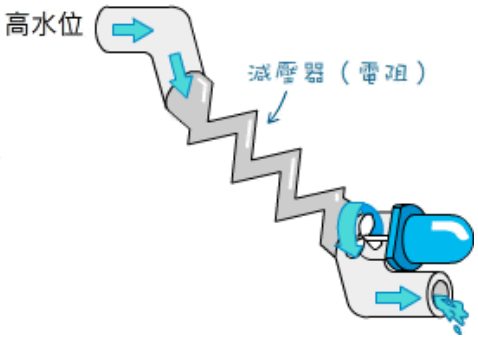




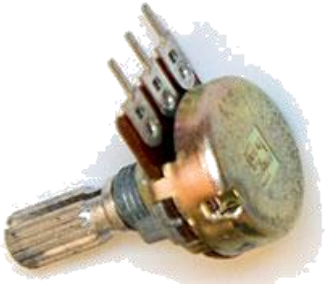
電阻

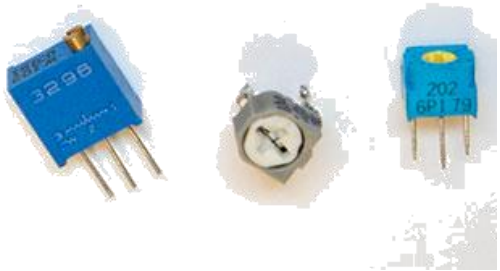
阻礙電流流動的因素叫電阻。電阻能降低和分散電子元件承受的電壓，避免元件損壞。



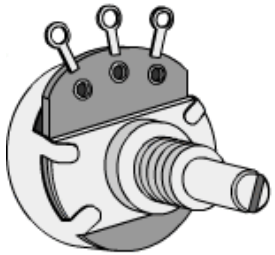


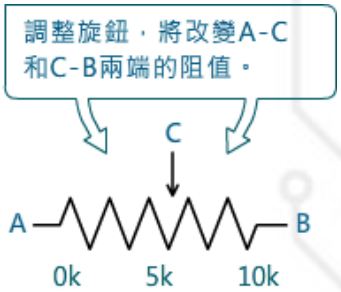
有些電阻具備可調整阻值的旋鈕，稱為可變電阻（簡稱VR）





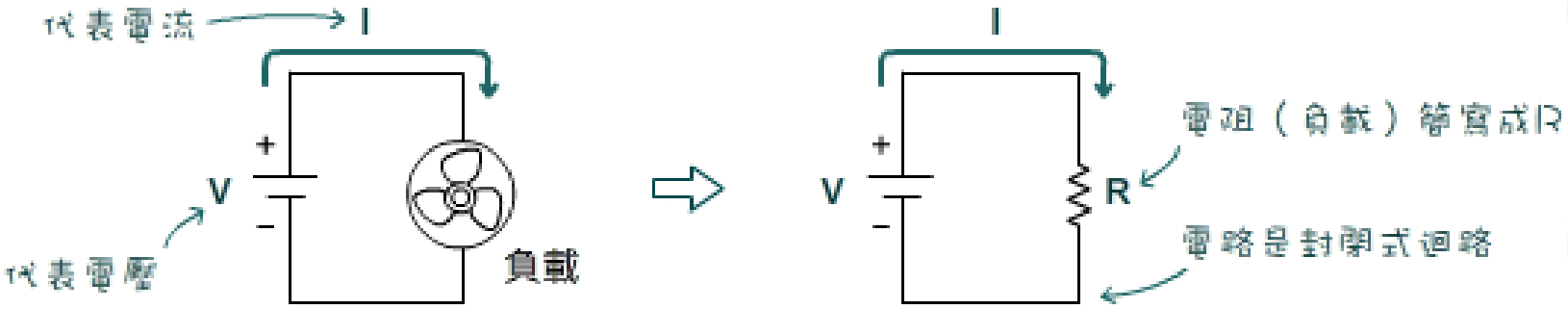
外觀

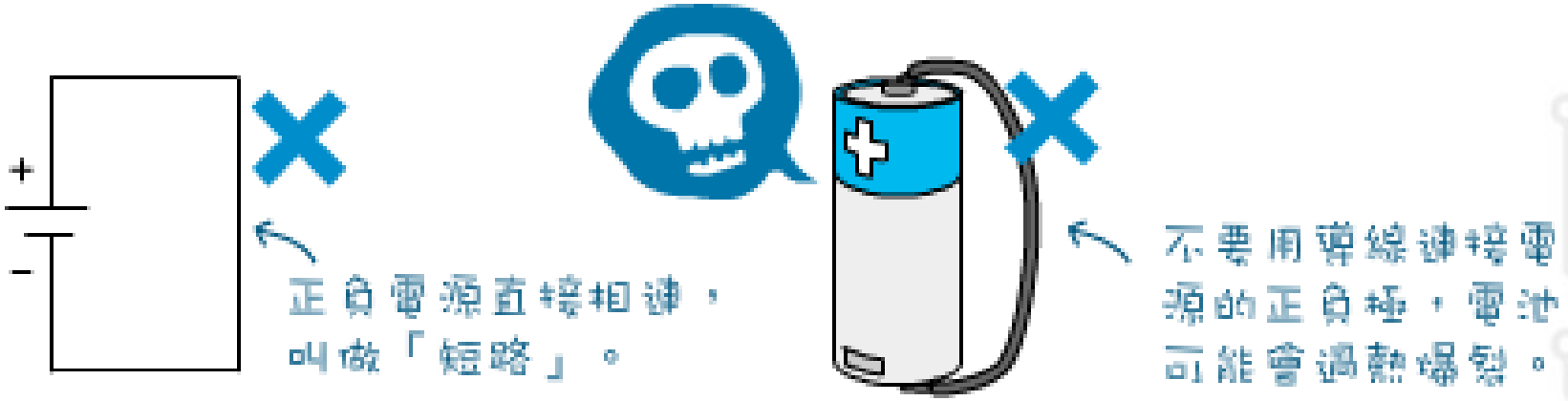




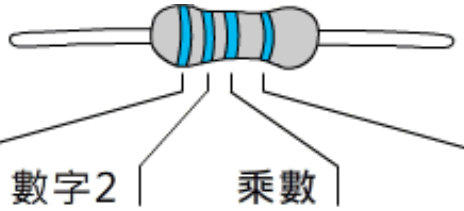
概念簡圖（電路符號）

電路和負載


代表電流
I
+
V
−
代表電壓
負載
I
+
V
−
R
電阻（負載）簡寫成R
電路是封閉式迴路


+
−
正負電源直接相連，
叫做「短路」。
不要用導線連接電
源的正負極，電池
可能會過熱爆炸。

電阻的色環



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

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

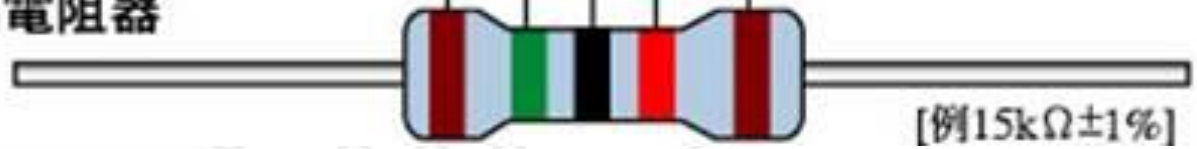
電阻色碼識別表

[例47kΩ±5%]

碳膜電阻器

色彩	第一色	第二色	第三色	倍數	誤差代號
黑	0	0	0	10^0(1Ω)	
棕	1	1	1	10^1(10Ω)	± 1% (F)
紅	2	2	2	10^2(100Ω)	± 2% (G)
橙	3	3	3	10^3(1KΩ)	
黃	4	4	4	10^4(10KΩ)	
綠	5	5	5	10^5(100KΩ)	± 0.5% (D)
藍	6	6	6	10^6(1MΩ)	± 0.25% (C)
紫	7	7	7	10^7(10MΩ)	± 0.10% (B)
灰	8	8	8		± 0.05%
白	9	9	9		
金				10^{-1}(0.1)	± 5% (J)
銀				10^{-2}(0.01)	± 10% (K)

金屬皮膜電阻器



[例15kΩ±1%]

實作練習

- 電路圖及麵包板接線圖：

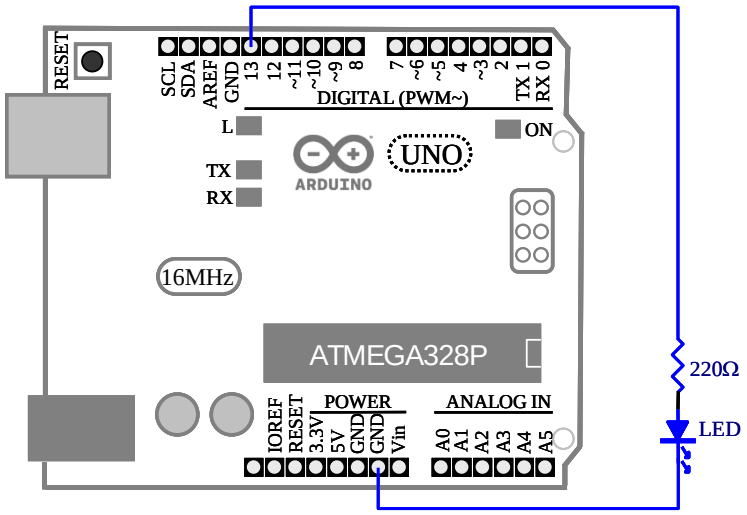


圖 4-3　一個 LED 閃爍實習電路圖

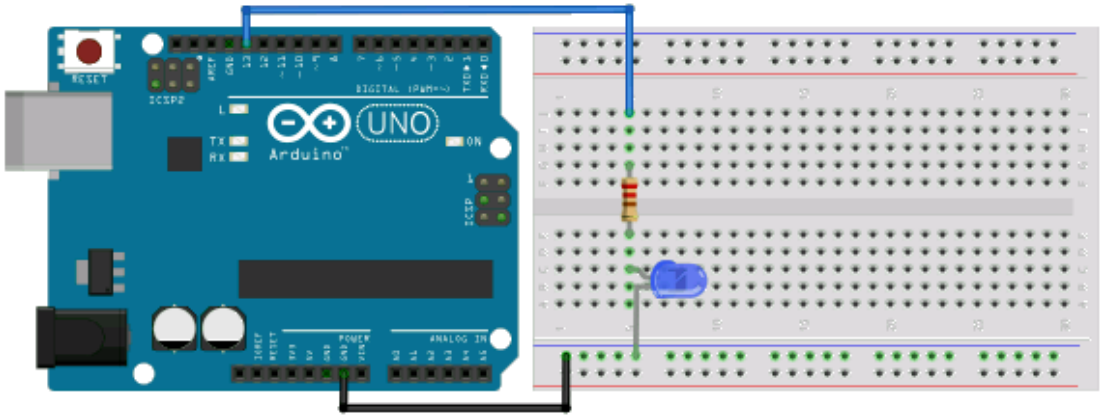


圖 4-4　一個 LED 閃爍實習麵包板接線圖

實作練習

- 程式：

```
const int led = 13;                    //LED 連接至數位輸出接腳 13。
void setup()
{
    pinMode(led,OUTPUT);               //設定數位接腳 13 為輸出模式。
}
void loop()
{   digitalWrite(led,HIGH);            //點亮 LED。
    delay(1000);                       //延遲 1 秒。
    digitalWrite(led,LOW);             //關閉 LED。
    delay(1000);                       //延遲 1 秒。
}
```

練習

1．設計 Arduino 程式，控制一個 LED 閃爍，0.5 秒亮、0.5 秒暗。

2．設計 Arduino 程式，控制兩個 LED 交替閃爍，0.5 秒亮、0.5 秒暗。

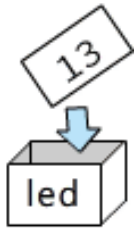
變數

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);
}
```



Arduino的變數與常數

- 在Arduino程式中常使用變數（variables）與常數（constants）來取代記憶體的實際位址，好處是程式設計者不需要知道那些位址是可以使用的，而且程式將會更容易閱讀與維護。

- 一個變數或常數的宣告是為了保留記憶體空間給某個資料來儲存，至於是安排那一個位址，則是由編譯器統一來分配。

變數名稱

- Arduino語言的變數命名規則
  - 與C語言相似，必須是由英文字母、數字或底線符號“_”
  - 第一個字元不可以是數字。因此我們在命名變數名稱時，應該以容易閱讀為原則，例如col、row代表行與列，就比i、j更容易了解。

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。

類型	中文名稱	佔用記憶體大小	數值範圍
boolean	布林	1位元	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

Arduino的變數與常數

表3-1　資料型態

資料型態	位元數	範圍
boolean	8	true（定義為非 0），false（定義為 0）
char	8	–128~+127
unsigned char	8	0~255
byte	8	0~255
int 註 1	16	–32,768~+32,767
unsigned int 註 2	16	0~65,535
word	16	0~65,535
long	32	–2,147,483,648~+2,147,483,647
unsigned long	32	0~4,294,967,295
short	16	–32,768~+32,767
float	32	–3.4028235E+38~+3.4028235E+38
double 註 3	32	–3.4028235E+38~+3.4028235E+38

註 1：在 Arduino Due 板為 32 位元，其餘為 16 位元。

註 2：在 Arduino Due 板為 32 位元，其餘為 16 位元。

註 3：在 Arduino Due 板為 64 位元，其餘為 32 位元。

Arduino的變數與常數

變數宣告

- int ledPin=10;　　//宣告整數變數ledPin，初始值為10。
- char myChar='A';　　//宣告字元變數myChar，初始值為' A'。
- float sensorVal=12.34
  //宣告浮點數變數sensorVal，初始值為12.34。
- int year=2013,moon=7,day=11;
  //宣告整數變數year、moon、day及其初值。

常數

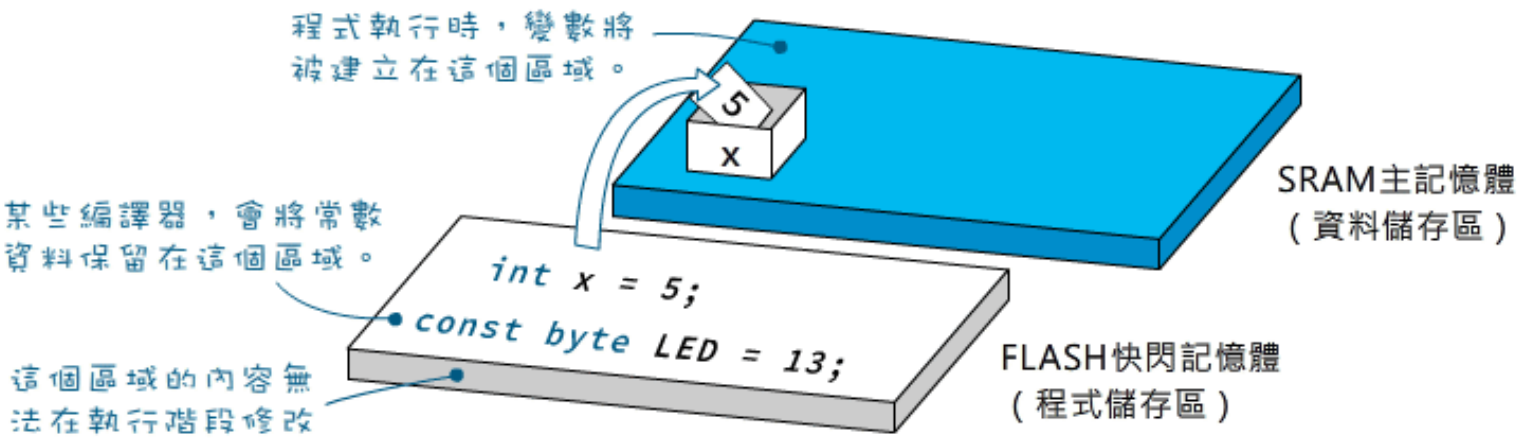
存放數值固定、不變的容器，稱為「常數」

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



變數的生命週期

- **全域變數**
- 宣告在任何函式之外
- 當執行Arduino程式時，全域變數即被產生並且配置記憶體空間給這些全域變數，直到程式結束執行時，才會釋放這些佔用的記憶體空間。
- 全域變數並不會禁止與其無關的函式作存取的動作，因此在使用上要特別小心，避免變數數值可能被不經意地更改。因此除非有特別需求，否則還是儘量使用區域變數

- **區域變數**
- 區域變數又稱為自動變數，被宣告在函式的大括號"{ }"內
- 當函式被呼叫使用時，這些區域變數就會自動的產生，系統會配置記憶體空間給這些區域變數，當函式結束時，這些區域變數又自動的消失並且釋放所佔用的記憶體空間

```
int    total;                          //全域變數 total 在所有函數內皆有效
void setup() {
       //...
}
void loop() {
       int i;                          //區域變數 I 只有在loop()函數內才有效
       for(int j=0;j<100;j++) {        //區域變數 j 只有在for迴圈內才有效
          //...
       }
}
```

變數的有效範圍

在室外宣告的變數：全域變數

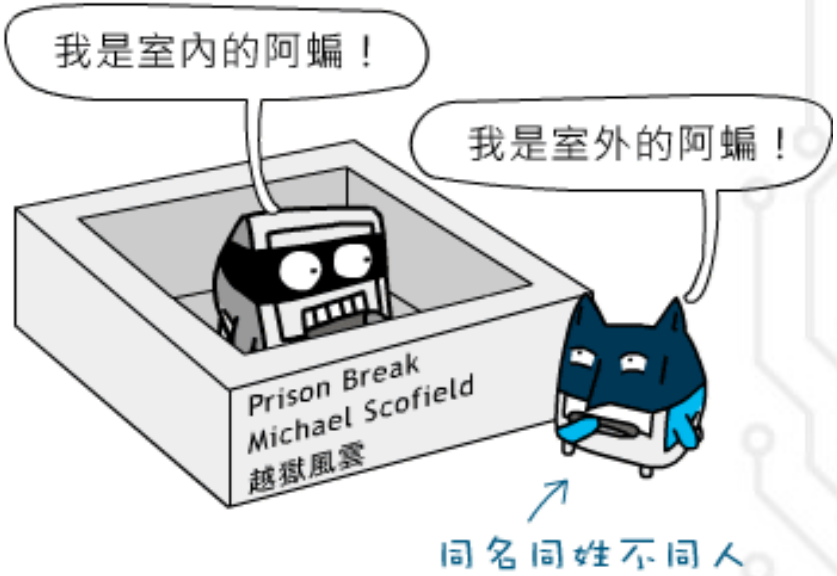
```
int age = 20;

void check() {
  int age = 10;
  Serial.print("function: ");
  Serial.println(age);
}

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

void loop() {
}
```

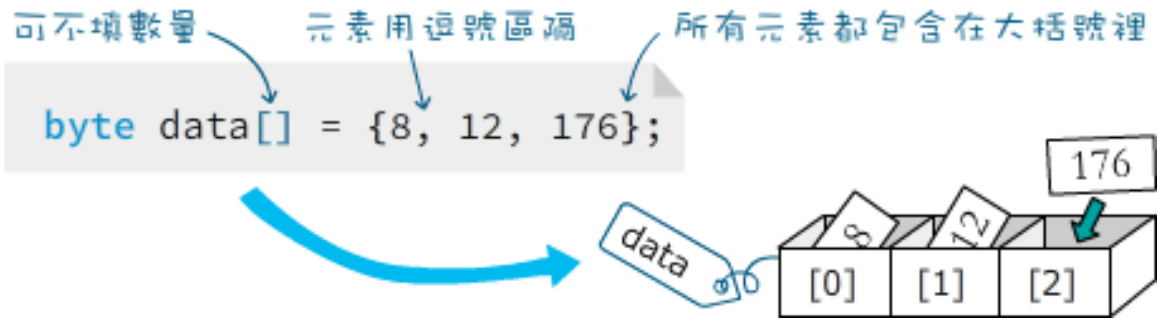
在室內宣告的變數：區域變數



陣列變數

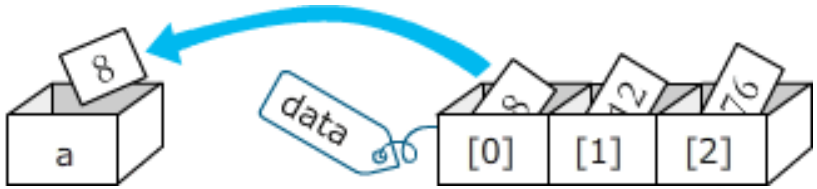
陣列（array）變數可以存放很多不同值，就像具有不同分隔空間的盒子一樣。

宣告陣列的同時可一併設定其值



讀取陣列元素

```
byte a = data[0];
```

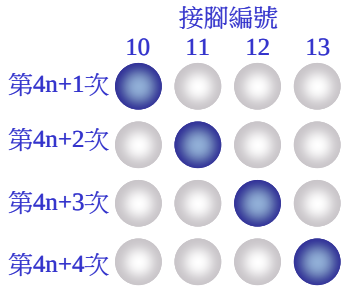


實作練習

四個 LED 單燈右移實習

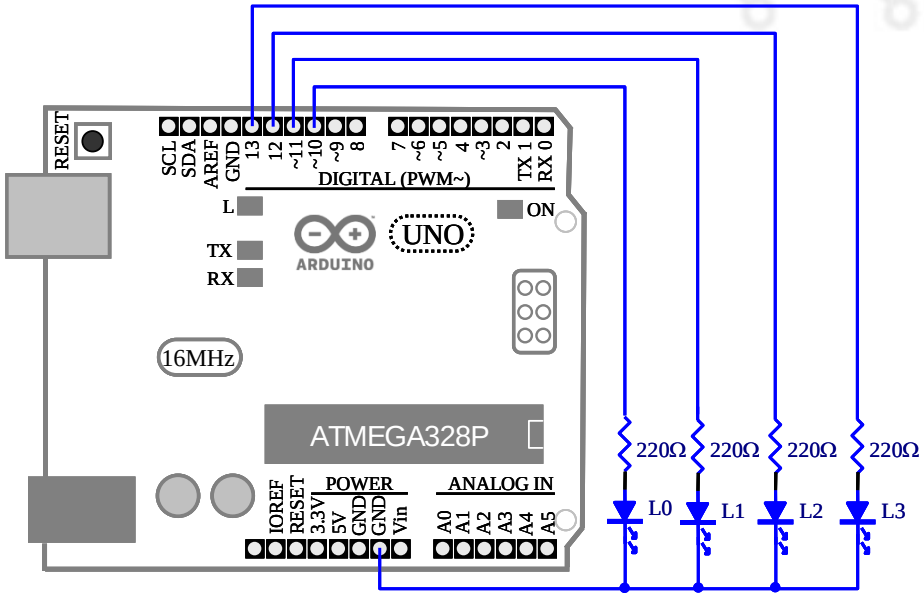
- 功能說明：

利用 Arduino 板數位接腳 10~13 控制四個 LED 執行單燈右移變化。如圖 4-5 所示 LED 共有 4 種變化，n 值為整數 0、1、2、…，4n+1，表示第 1、5、9…等次數位接腳 10~13 的 LED 狀態相同，其餘依此類推。

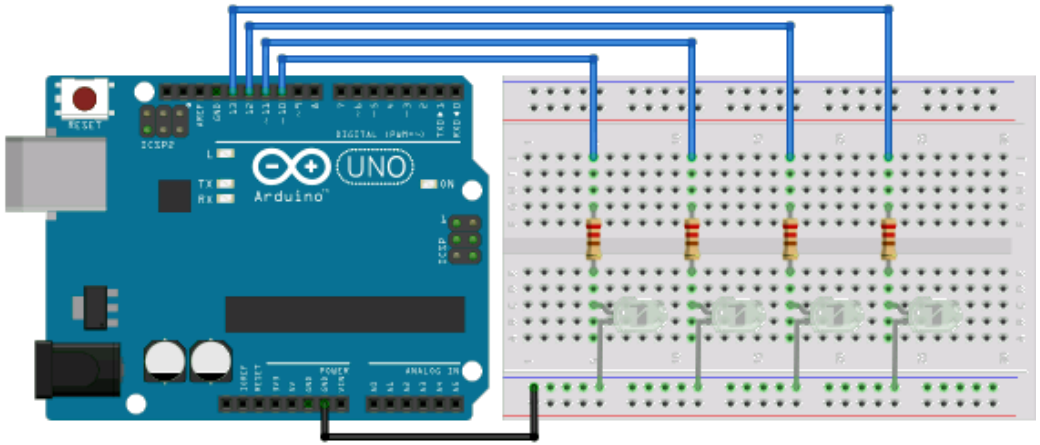


單燈右移變化

- 電路圖麵包板接線圖：



四個 LED 單燈右移電路圖



四個 LED 單燈右移麵包板接線圖`

- 程式：  B101.ino

```
const int led[] = {10,11,12,13};          //LED 連接至數位接腳 10~13。
int i;                                    //LED 索引值。
int j=0;                                  //LED 索引值。
void setup()
{
    for(i=0;i<4;i++)                      //四個 LED。
    {
        pinMode(led[i], OUTPUT);          //設定數位接腳 10~13 為輸出模式。
    }
}
void loop()
{
    for(i=0;i<4;i++)                      //四個 LED。
    {
        digitalWrite(led[i],LOW);         //關閉所有 LED。
    }
    digitalWrite(led[j], HIGH);           //點亮第 j 個 LED。
    delay(1000);                          //延遲 1 秒。
    j++;                                  //右移。
    if(j==4)                              //已右移至最右方?
        j=0;                              //重新點亮最左方 LED。
}
```

Arduino程式流程控制

迴圈控制指令— for迴圈

- for 迴圈

  for 迴圈是由初值運算式、條件運算式與增量或減量運算式三部份組成，彼此之間以分號隔開。

  (1) 初值：設定初值，可由任何數值開始。

  (2) 條件：若條件為真，則執行括號"{ }"中的敘述，否則離開迴圈。

  (3) 增量(或減量)：每執行一次迴圈內的動作後，依增量(減量)遞增(遞減)。

- 指令格式

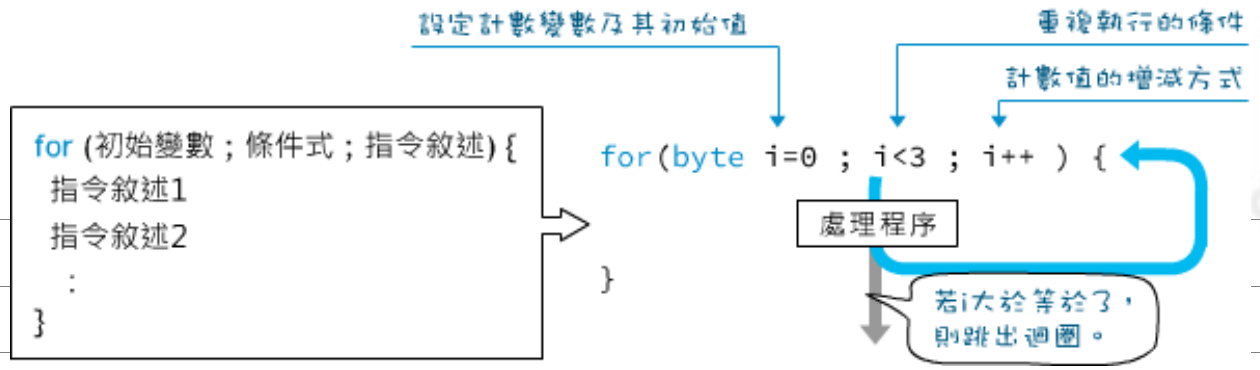
```
for(初值; 條件; 增量或減量)
{
    //敘述;
}
```

Arduino程式流程控制

迴圈控制指令— for迴圈

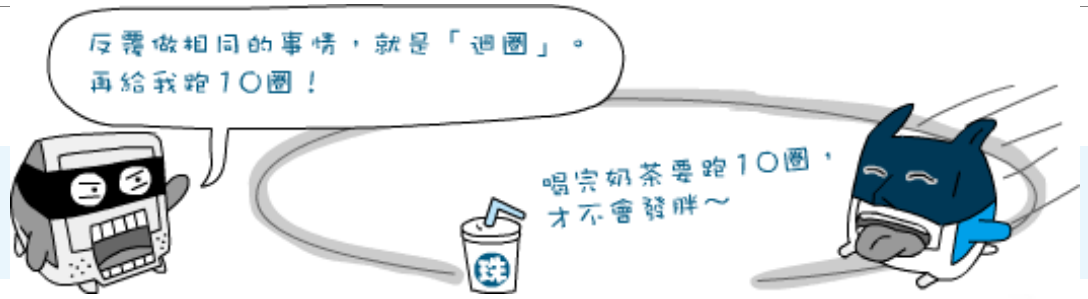
- 程式範例

```
void setup()
{}
void loop()
{
    int i,s=0;                    //宣告整數變數。
    for(i=0;i<=10;i++)            //當 i<=10 時，執行迴圈。
        s=s+i;                    //s=1+2+…+10。
}
```



- 執行結果

```
s=55
```



Arduino程式流程控制

條件控制指令— if敘述

- if 敘述

  if 敘述會先判斷條件式，若條件式為真時，則執行一次大括號{}中的敘述，若條件式為假時，則不執行。if 敘述內如果只有一行敘述時，可以不用加大括號"{ }"，但如果有一行以上敘述時，一定要加上大括號"{ }"，否則在 if 敘述內只會執行第一行敘述，其餘敘述則視為在 if 敘述之外。

- 指令格式

```
if （ 條件式 ）
{
    //敘述;
}
```

Arduino程式流程控制

條件控制指令— if敘述

- 程式範例

```
void setup()
{}
void loop()
{
    int a=2,b=3,c=0;                    //宣告整數變數。
    if(a>b)                             //a 大於 b?
    {
       c=a;                             //若 a 大於 b，則 c=a。
    }
}
```

- 執行結果

```
c=0
```

關係運算子

表3-3　關係運算子

比較運算子	動作	範例	說明
= =	等於	a= = b	a 等於 b? 若為真，結果為 true，否則為 false。
!=	不等於	a!= b	a 不等於 b? 若為真，結果為 true，否則為 false。
<	小於	a< b	a 小於 b? 若為真，結果為 true，否則為 false。
>	大於	a> b	a 大於 b? 若為真，結果為 true，否則為 false。
< =	小於等於	a< = b	若 a 小於或等於 b，結果為 true，否則為 false。
> =	大於等於	a> = b	若 a 大於或等於 b，結果為 true，否則為 false。

- 程式範例

```
void setup()
{
}
void loop()
{
    int x=200;                          //宣告整數變數 x，初始值為 200。
    if(x>100)                           //x 大於 100?
       digitalWrite(LEDpin,HIGH);       //若 x 大於 100 則點亮 LED。
    else                                //x 小於或等於 100。
       digitalWrite(LEDpin,LOW);        //x 小於或等於 200，則關閉 LED。
}
```

Arduino程式流程控制

條件控制指令— if-else敘述

- if-else 敘述

  if-else 敘述會先判斷條件式，若條件式為真時，則執行敘述 1，若條件式為假時，則執行敘述 2。在 if 敘述或 else 敘述內，如果只有一行敘述時，可以不用加大括號"{ }"，但如果有一行以上敘述時，一定要加上大括號"{ }"。

- 指令格式

```
if （條件式）                          //條件式為真，執行敘述1。
{
    //敘述1;
}
else                                   //條件式為假，執行敘述2。
{
    //敘述2;
}
```

Arduino程式流程控制

條件控制指令— if-else敘述

- 程式範例

```
void setup()
{}
void loop()
{
    int a=3,b=2,c=0;                    //宣告整數變數。
    if(a>b)                             //a 大於 b?
    {
        c=a;                            //若 a 大於 b，則 c=a。
    }
    else                                //a 小於或等於 b
    {
        c=b;                            //c=b。
    }
}
```

- 執行結果

```
c=3
```

Arduino程式流程控制

條件控制指令—巢狀if-else敘述

- 巢狀 if-else 敘述

使用巢狀 if-else 敘述時，必須注意 if 與 else 的配合，其原則是 else 要與最接近且未配對的 if 配成一對，通常我們都是以 Tab 定位鍵或空白字元來對齊配對的 if-else，才不會有錯誤動作出現。在 if 敘述或 else 敘述內，如果只有一行敘述時，可以不用加大括號{}，但如果有一行以上敘述時，一定要加上大括號"{ }"。

- 指令格式

```
if （條件1 ）
    if （條件2 ）
        if （條件3 ）                    //條件1、2、3 皆成立時，執行敘述1。
            {//敘述1;}
        else                             //條件1、2 成立且條件3 不成立，執行敘述2。
            {//敘述2;}
    else
        {//敘述3;}                       //條件1 成立且條件2 不成立時，執行敘述3。
else
    {//敘述4; }                          //條件1 不成立時，執行敘述4。
```

Arduino程式流程控制

條件控制指令—巢狀if-else敘述

- 程式範例

```
void setup()
{}
void loop()
{
    int score=75;
    char grade;
    if(score>=60)                    //成績大於或等於 60 分?
       if(score>=70)                 //成績大於或等於 70 分?
          if(score>=80)              //成績大於或等於 80 分?
             if(score>=90)           //成績大於或等於 90 分?
                grade='A';           //成績大於或等於 90 分，等級為 A。
             else                    //成績在 80~90 分之間。
                grade='B';           //成績在 80~90 分之間，等級為 B。
          else                       //成績在 70~80 分之間。
             grade='C';              //成績在 70~80 分之間，等級為 C。
       else                          //成績在 60~70 分之間。
          grade='D';                 //成績在 60~70 分之間，等級為 D。
    else                             //成績小於 60 分。
       grade='E';                    //成績小於 60 分，等級為 E。
}
```

- 執行結果

```
grade='C'
```

小練習

1．設計 Arduino 程式，控制四個 LED 單燈左移。

2．設計 Arduino 程式，控制四個 LED 單燈閃爍左移。

實作練習

四個 LED 霹靂燈變化實習

- 功能說明：

利用 Arduino 板接腳 10~13 控制四個 LED 執行霹靂燈移位變化。如圖 4-8 所示霹靂燈共有 8 種變化，n 值為整數 0、1、2、...8n+1，表示第 1、9、17... 等次數位接腳 10~13 的 LED 狀態相同，餘依此類推。

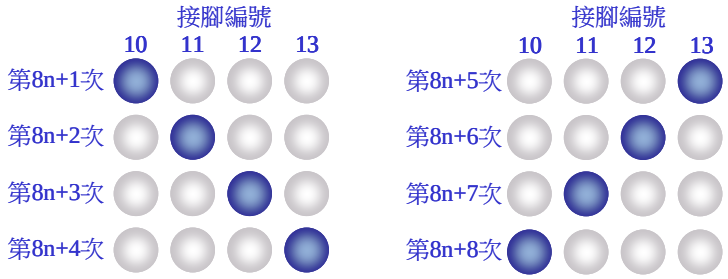
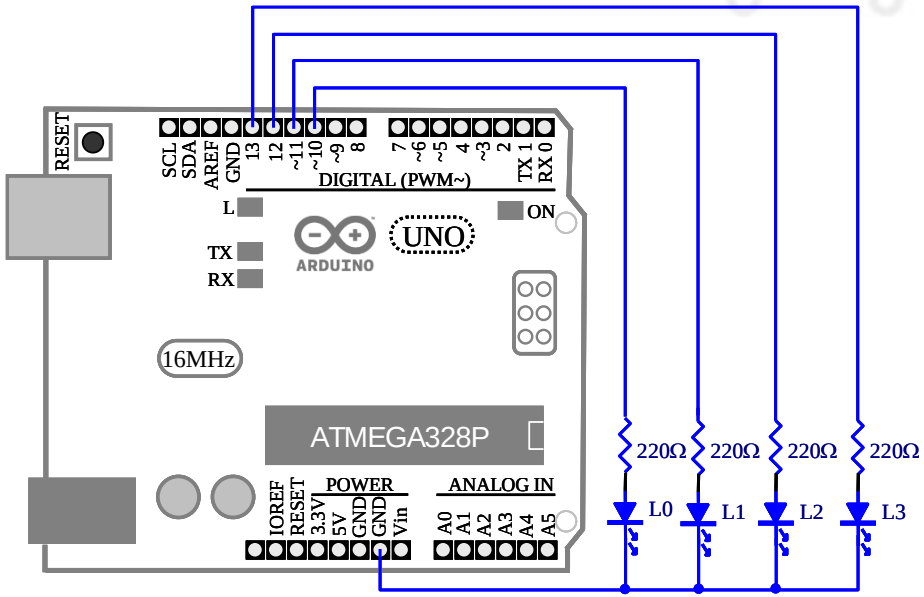
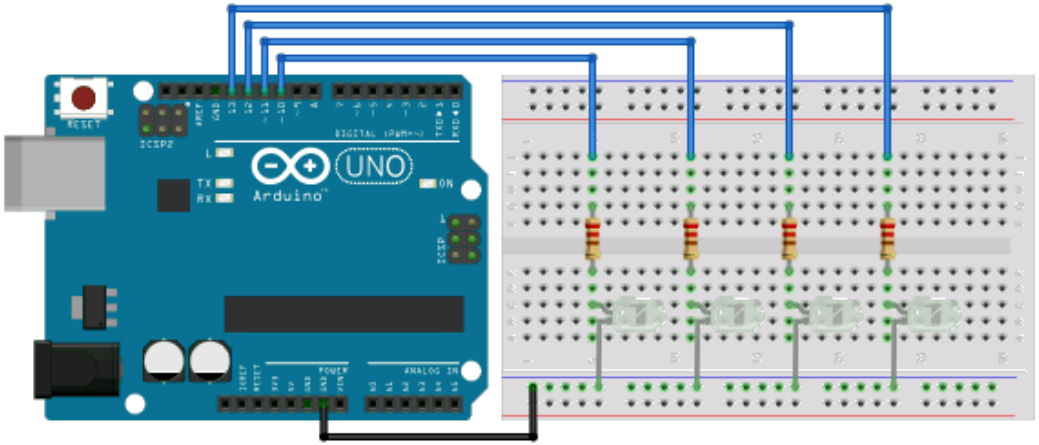


圖 4-8　四個 LED 霹靂燈變化

- 電路圖麵包板接線圖：



四個 LED 單燈右移電路圖



四個 LED 單燈右移麵包板接線圖`

實作練習

- 程式：  B102.ino

```
const int led[]={10,11,12,13};          //LED 連接至數位接腳 10~13。
int i;                                  //LED 索引值。
int j=0;                                //LED 索引值。
int direct=0;                           //LED 移位方向控制。
void setup()
{
    for(i=0;i<4;i++)
        pinMode(led[i],OUTPUT);         //設定數位接腳 10~13 為輸出模式。
}
```

實作練習

```
void loop()
{
    for(i=0;i<4;i++)
    {
        digitalWrite(led[i],LOW);          //設定第 10~13 號數位腳輸出狀態為 LOW。
    }
    digitalWrite(led[j],HIGH);             //設定第 j 號數位腳輸出狀態為 HIGH。
    delay(500);
    if(direct==0)                          //LED 在右移狀態?
    {
        if(j==3)                           //已右移至最右方?
            direct=1;                      //改為左移。
        else                               //尚未移至最右方。
            j=j+1;                         //繼續右移。
    }
    else                                   //LED 在左移狀態。
    {
        if(j==0)                           //已左移至最左方?
            direct=0;                      //改為右移。
        else                               //尚未移至最左方。
            j=j-1;                         //繼續左移。
    }
}
```

(回家)小練習

練習

1．設計 Arduino 程式，控制四個 LED 執行霹靂燈閃爍移位變化。

2．設計 Arduino 程式，控制四個 LED 變化如圖 4-9 所示。

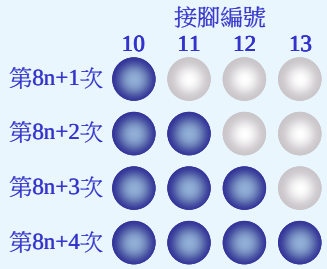
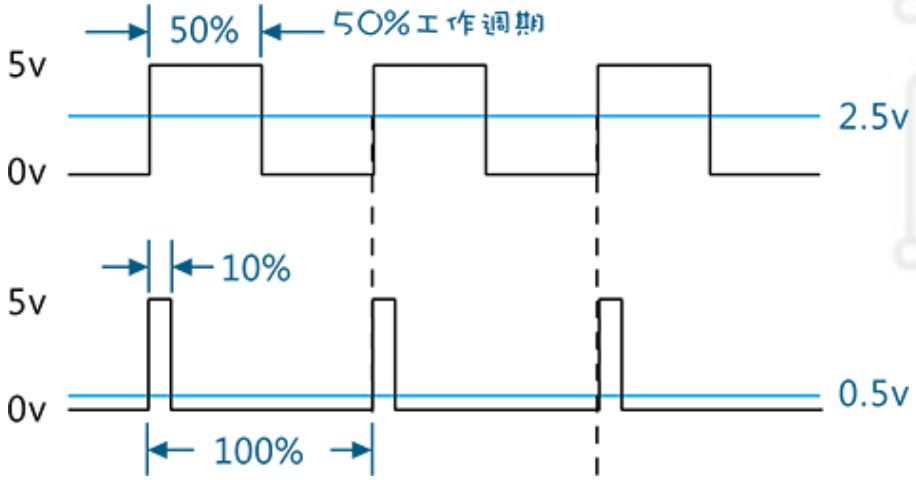


圖 4-9　練習 2

省電節能又環保的PWM變頻技術

高速切換高、低電位（頻率高於30Hz），數位訊號將能模擬類比電壓高低變化的效果。

在數位系統上「模擬」類比輸出的方式，稱為**脈寬調變**（Pulse Width Modulation，簡稱**PWM**）。

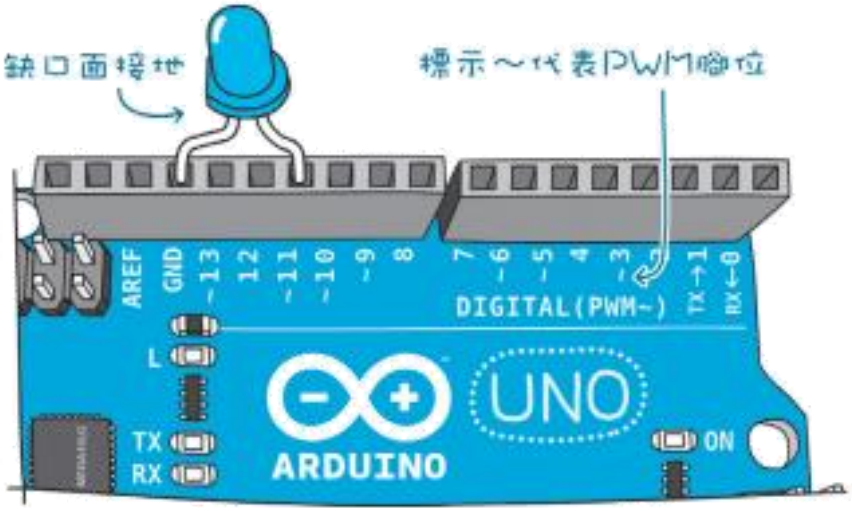


類比輸出電壓 = 脈衝寬度 × 高電位值 → $\frac{\text{輸出電壓}}{\text{高電位值}}$ = 開啟時間百分比

$\frac{\text{輸出電壓}}{\text{高電位值}} \rightarrow \frac{3.3v}{5v} = 0.66 \rightarrow 0.66 \times 100\% = 66\%$

類比輸出（PWM）指令

Arduino微電腦板預設採用1KHz和500Hz兩組不同的PWM輸出頻率：
•接腳5,6：976.5625Hz（約1kHz）
•接腳3, 11以及9, 10：490.196Hz（約500Hz）



analogWrite(5, 168);

$$\frac{3.3v}{5v} \times 255 = 168.3$$

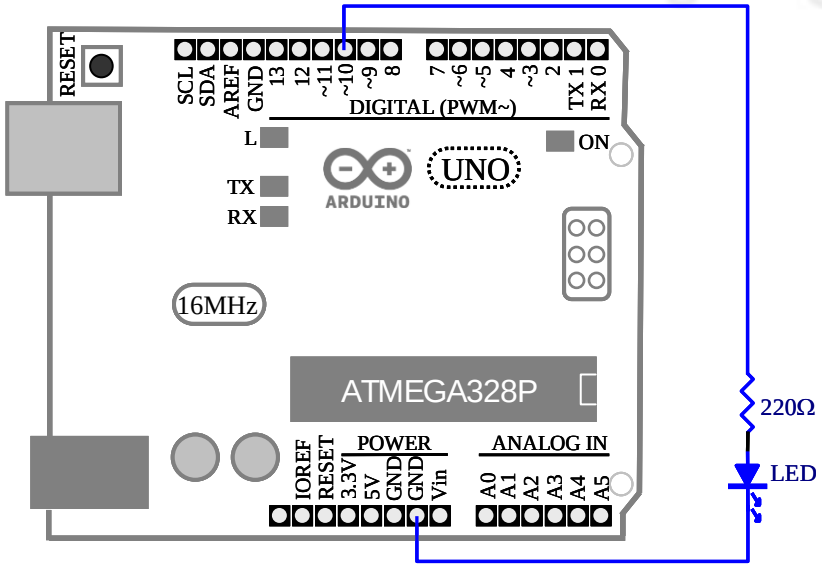
實作練習

一個 LED 亮度變化實習

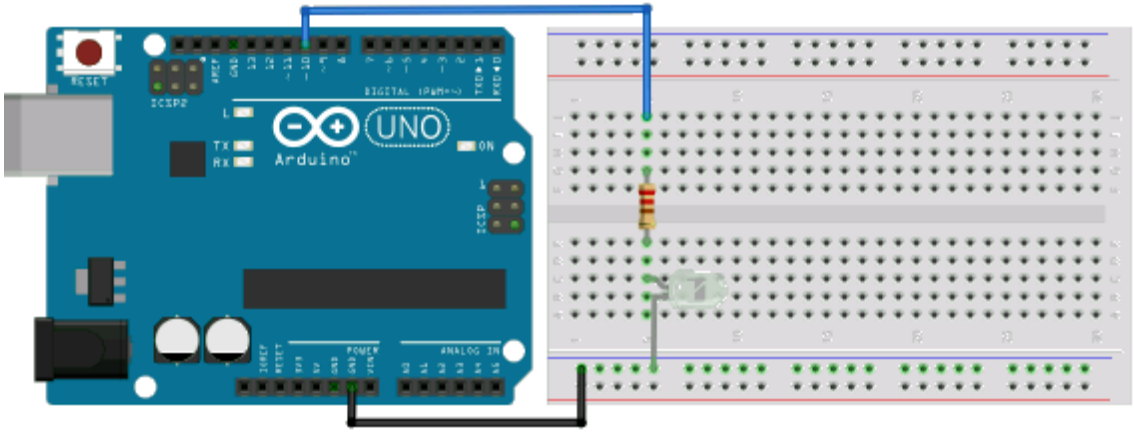
- 功能說明：

利用 Arduino 板 PWM 信號輸出接腳 10 控制一個 LED 亮度由最暗變化至最亮。在 Arduino UNO 板中只有 3、5、6、9、10、11 等接腳可以輸出 PWM 信號，本例使用接腳 10 輸出 PWM 信號，PWM 工作週期愈大，則 LED 愈亮。

- 電路圖及麵包板接線圖：



一個 LED 亮度變化實習電路圖



一個 LED 亮度變化實習麵包板接線圖

函式說明

analogWrite( )函式

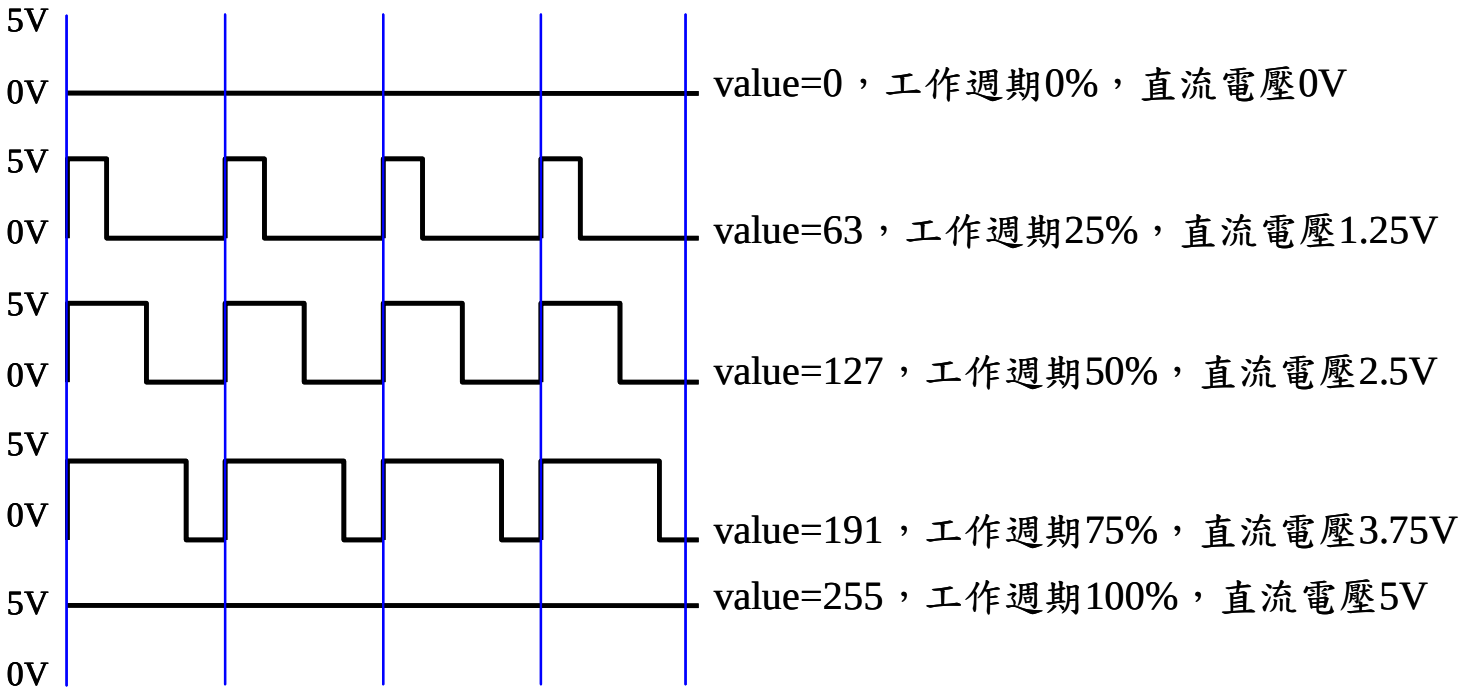
analogWrite( )函式功用是輸出脈波調變信號（Pulse Width Modulation，簡記PWM）至指定接腳，頻率大約是 500Hz，PWM 信號可以用來在控制 LED 的亮度或是直流馬達的轉速，在使用 analogWrite( )函式輸出 PWM 信號時，不需要先使用 pinMode( )函式去設定指定接腳為輸出模式。analogWrite( )函式有 pin 及 value 兩個參數必須設定，pin 參數設定 PWM 信號輸出腳，大多數的 Arduino 板使用 3、5、6、9、10 和 11 等接腳輸出 PWM 信號，value 參數設定 PWM 信號的脈波寬度 t_{on}，其值為 0~255，定義工作週期等於 $\frac{t_{on}}{T}\times 100\%$，而 T 值為 255，因此直流電壓等於 $\frac{t_{on}}{T}\times 5V$。

格式： `analogWrite(pin,value)`

範例：
```
analogWrite(5,127);                    //輸出工作週期為 50%的 PWM 信號至接腳 5
```

函式說明

如圖 4-2 所示，當 value=0 時，工作週期為 0%，直流電壓為 0；當 value=63 時，工作週期為 25%，直流電壓為 1.25V；當 value=127 時，工作週期為 50%，直流電壓為 2.5V；當 value=191 時，工作週期為 75%，直流電壓為 3.75V；當 value=255 時，工作週期為 100%，直流電壓為 5V。



PWM 信號

實作練習

- 程式：  B111.ino

```
const int led = 10;                          //LED 連接至 PWM 接腳 10。
int brightness = 0;                          //LED 亮度。
int lighten = 5;                             //LED 亮度變化量。
void setup()
{
}
void loop()
{
   analogWrite(led,brightness);              //改變 LED 亮度。
   if(brightness<=250)                       //LED 最亮?
       brightness=brightness+lighten;        //LED 漸亮變化。
   else                                      //LED 最亮。
       brightness=0;                         //設定 LED 亮度為最暗。
    delay(50);                               //LED 亮度變化間隔 50ms。
}
```

(回家)小練習

1．設計 Arduino 程式，控制一個 LED 由最暗變化至最亮，再由最亮變化至最暗。

2．設計 Arduino 程式，控制兩個 LED 輪流由最暗變化至最亮，再由最亮變化至最暗。

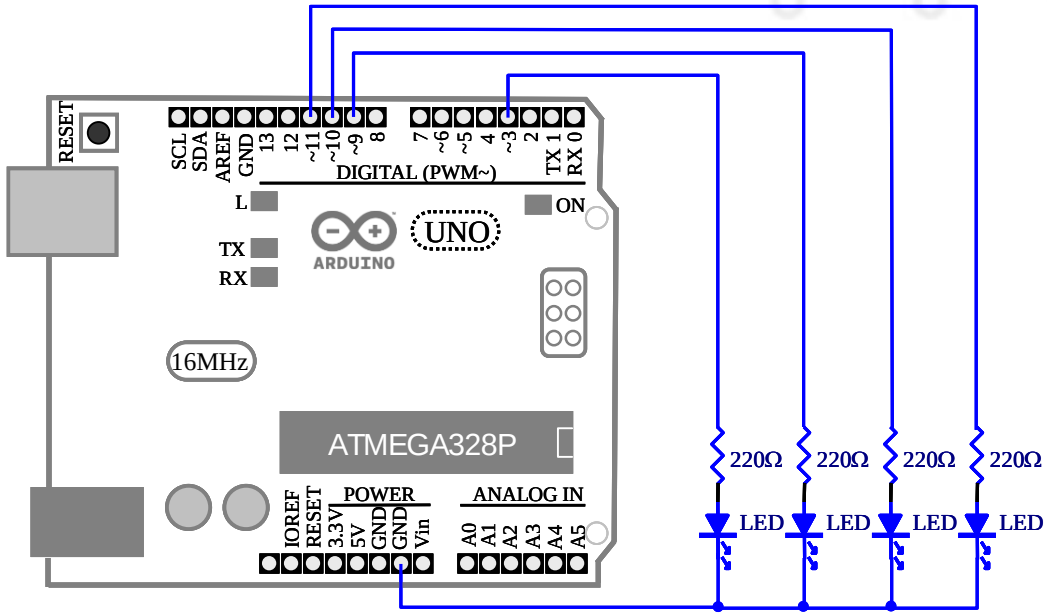
實作練習

四個 LED 亮度變化單燈右移實習

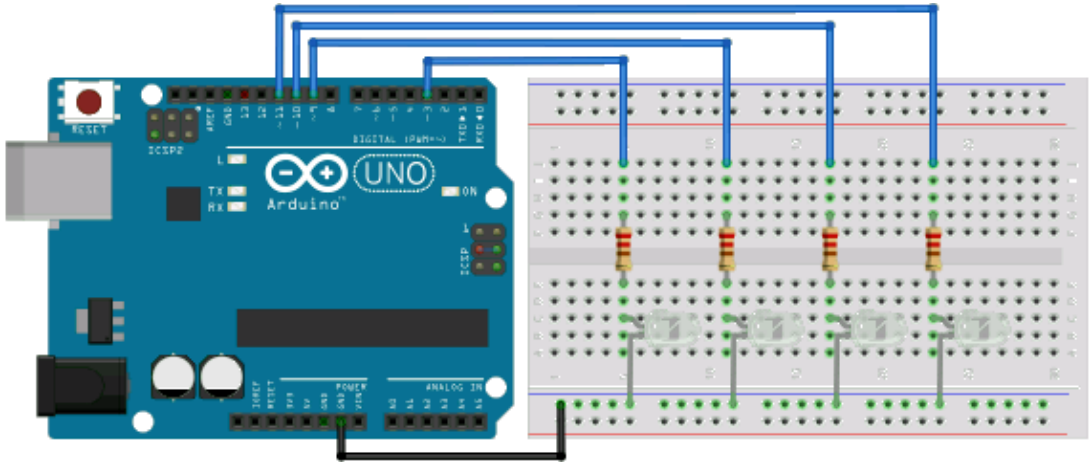
- 功能說明：

  利用 Arduino 板 PWM 信號輸出接腳 3、9、10、11 控制四個 LED 亮度變化並依序單燈右移，每一個 LED 亮度由暗逐漸變亮，再由亮逐漸變暗後，再右移至下一個 LED。LED 亮度由 PWM 信號控制，當 PWM 信號工作週期增加時，LED 亮度增加，反之當 PWM 信號工作週期減小時，LED 亮度減小。

- 電路圖及麵包板接線圖：



四個 LED 亮度變化右移實習電路圖



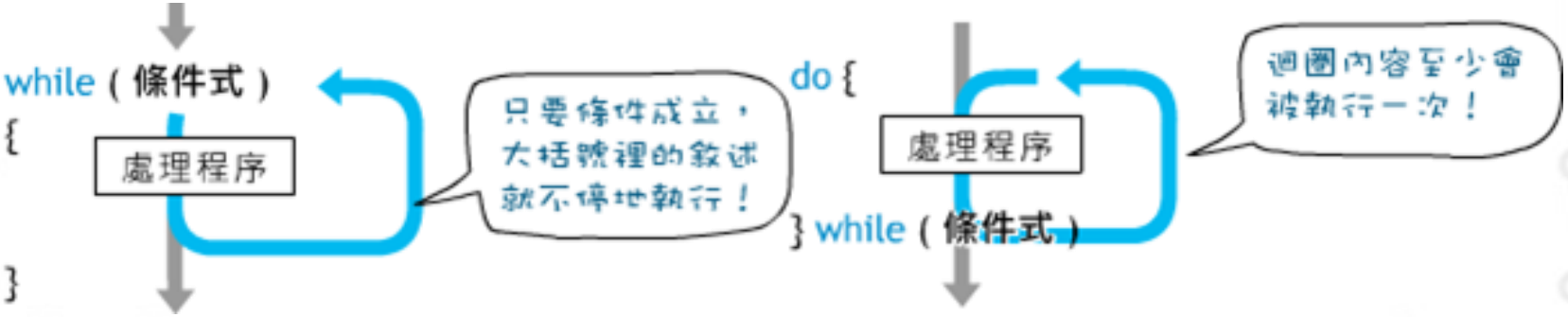
四個 LED 亮度變化右移實習麵包板接線圖

實作練習

- 程式：  B112.ino

```
const int led[] ={3,9,10,11};          //LED 連接至 PWM 信號輸出接腳。
int brightness=0;                      //LED 亮度。
int lighten=5;                         //LED 亮度變化量。
int ledNums=0;                         //作用中的 LED。
int direct=0;                          //LED 漸亮或漸暗變化方向。
void setup()
{
}
```

while 迴圈敘述



```
void loop()
{
    while(1)                                        //重覆變化。
    {
        if(direct==0)                               //LED 由暗逐漸變亮?
        {
            brightness=brightness+lighten;          //LED 漸亮。
            if(brightness>=250)                     //已至最亮?
                direct=1;                           //改變為由亮逐漸變暗。
        }
        else                                        //LED 由亮逐漸變暗。
        {
            brightness=brightness-lighten;          //LED 漸暗。
            if(brightness<=5)                       //已至最暗?
            {
                analogWrite(led[ledNums],0);        //關閉 LED。
                direct=0;                           //改變為由暗逐漸變亮。
                ledNums++;                          //下一個 LED。
                if(ledNums>3)                       //led 已右移至最右方?
                    ledNums=0;                      //重新設定第一個 LED。
            }
        }
        analogWrite(led[ledNums],brightness);//改變 LED 亮度。
        delay(10);                                  //延遲 10ms。
    }
}
```

回家(小)練習

1．設計 Arduino 程式，控制四個 LED 亮度變化並依序左移。每一個 LED 亮度由暗逐漸變亮，再由亮逐漸變暗。

2．設計 Arduino 程式，控制四個 LED 亮度變化並依序來回左右移。每一個 LED 亮度由暗逐漸變亮，再由亮逐漸變暗。

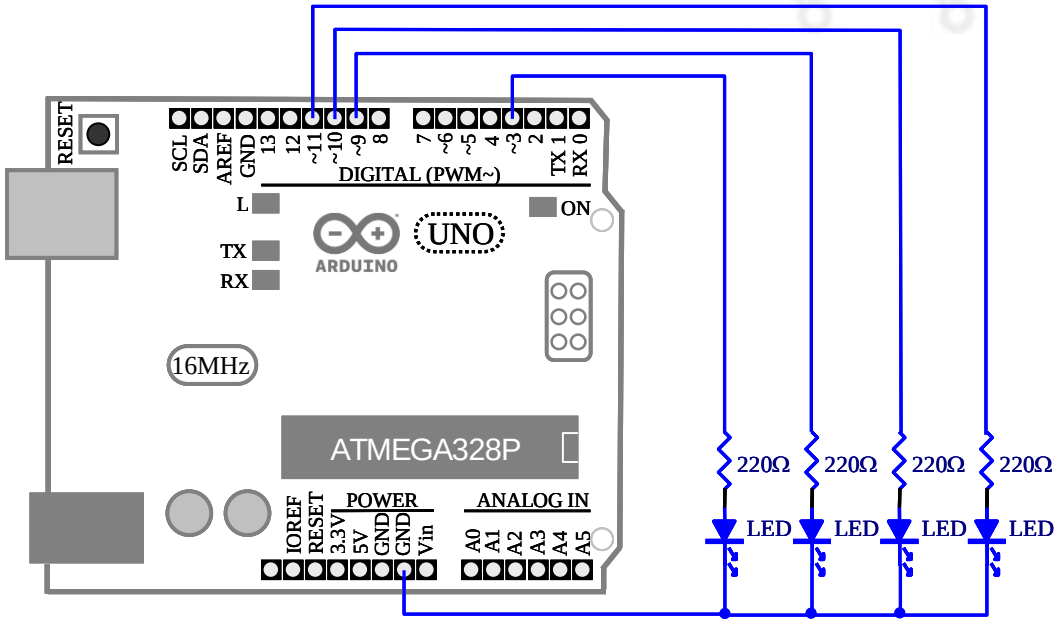
實作練習

雨滴燈實習

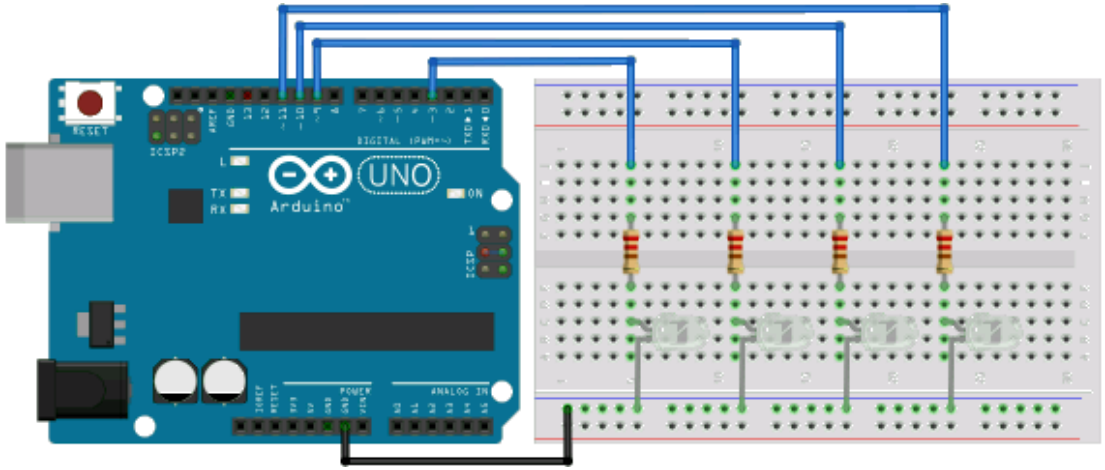
- 功能說明：

  利用 Arduino 板 PWM 信號輸出接腳 3、9、10、11 控制四個 LED 產生右移雨滴變化。

- 電路圖及麵包板接線圖：



四個 LED 亮度變化右移實習電路圖



四個 LED 亮度變化右移實習麵包板接線圖

實作練習

- 程式：  B113.ino

```
const int led[] ={3,9,10,11};          //LED 連接至 PWM 信號輸出接腳。
int varNums;                           //LED 變化種類。
int ledNums;                           //LED 亮度種類。
const int brightness[7][4]=            //LED 變化資料。
  { {250,0,0,0},                       //第 1 次 LED 狀態。
    {100,250,0,0},                     //第 2 次 LED 狀態。
    {50,100,250,0},                    //第 3 次 LED 狀態。
    {5,50,100,250},                    //第 4 次 LED 狀態。
    {0,5,50,100},                      //第 5 次 LED 狀態。
    {0,0,5,50},                        //第 6 次 LED 狀態。
    {0,0,0,5}  };                      //第 7 次 LED 狀態。
```

實作練習

```
void setup()
{
}
void loop()
{
    for(varNums=0;varNums<7;varNums++)        //雨滴變化。
    {
        for(ledNums=0;ledNums<4;ledNums++)
            analogWrite(led[ledNums],brightness[varNums][ledNums]);
        delay(100);
    }
}
```

(回家)小練習

1．設計 Arduino 程式，利用 Arduino 板 PWM 信號輸出接腳 3、9、10、11 控制四個 LED 產生左移雨滴變化。

2．設計 Arduino 程式，控制四個 LED 模擬拖尾霹靂燈變化。

回家作業

即上述之第二題

使雨滴燈可以左右位移模擬霹靂燈

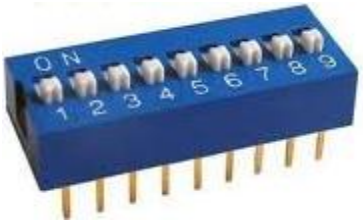
認識開關

- 開關的種類很多，主要用途是接通或斷開電路
  - 接通（ON）時，允許電流通過
  - 斷開（OFF）時，電路電流為零
- 常用的機械開關如搖動開關、滑動開關及按鍵開關等，都是利用金屬片接觸面與接點接觸而產生接通狀態。

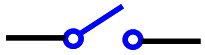
(a) 搖動開關

(b) 滑動開關

(c) 按鍵開關



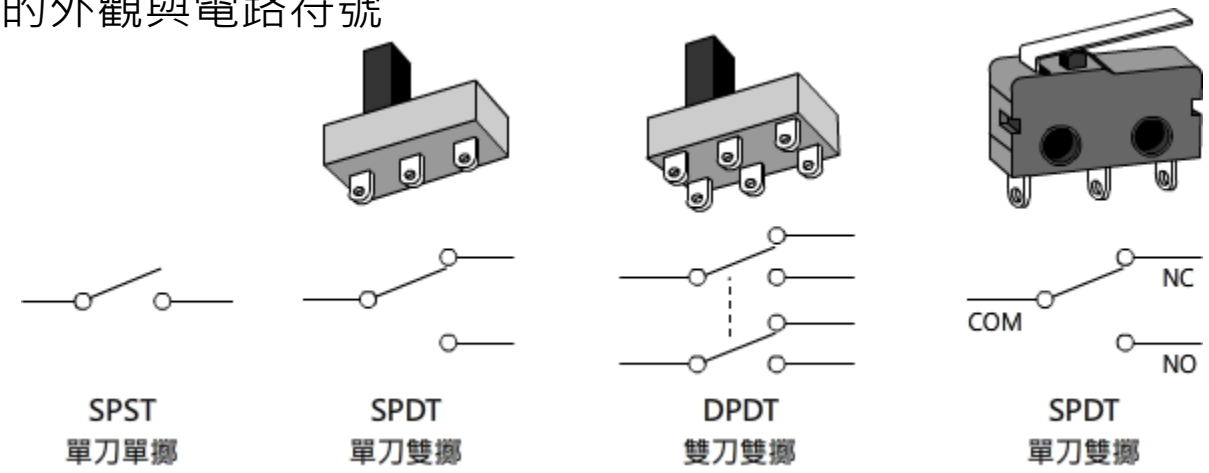
指撥開關

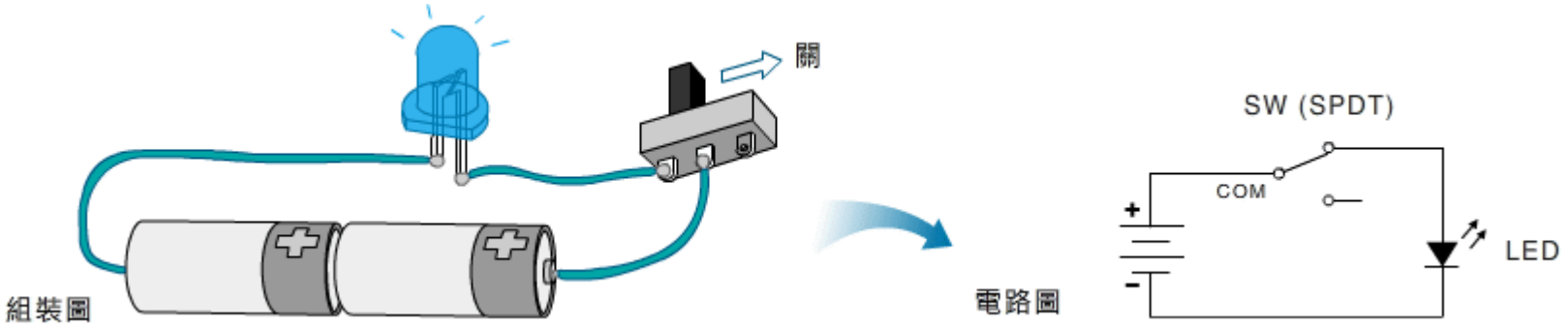


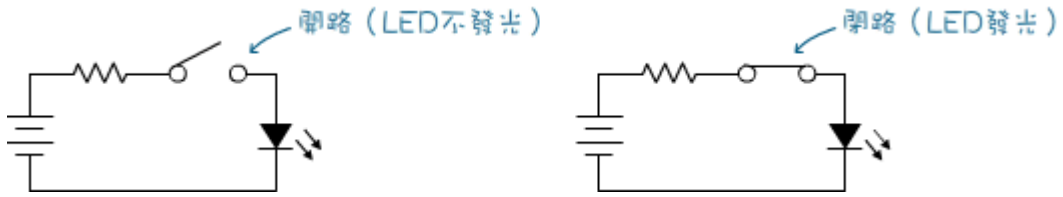
符號

認識開關

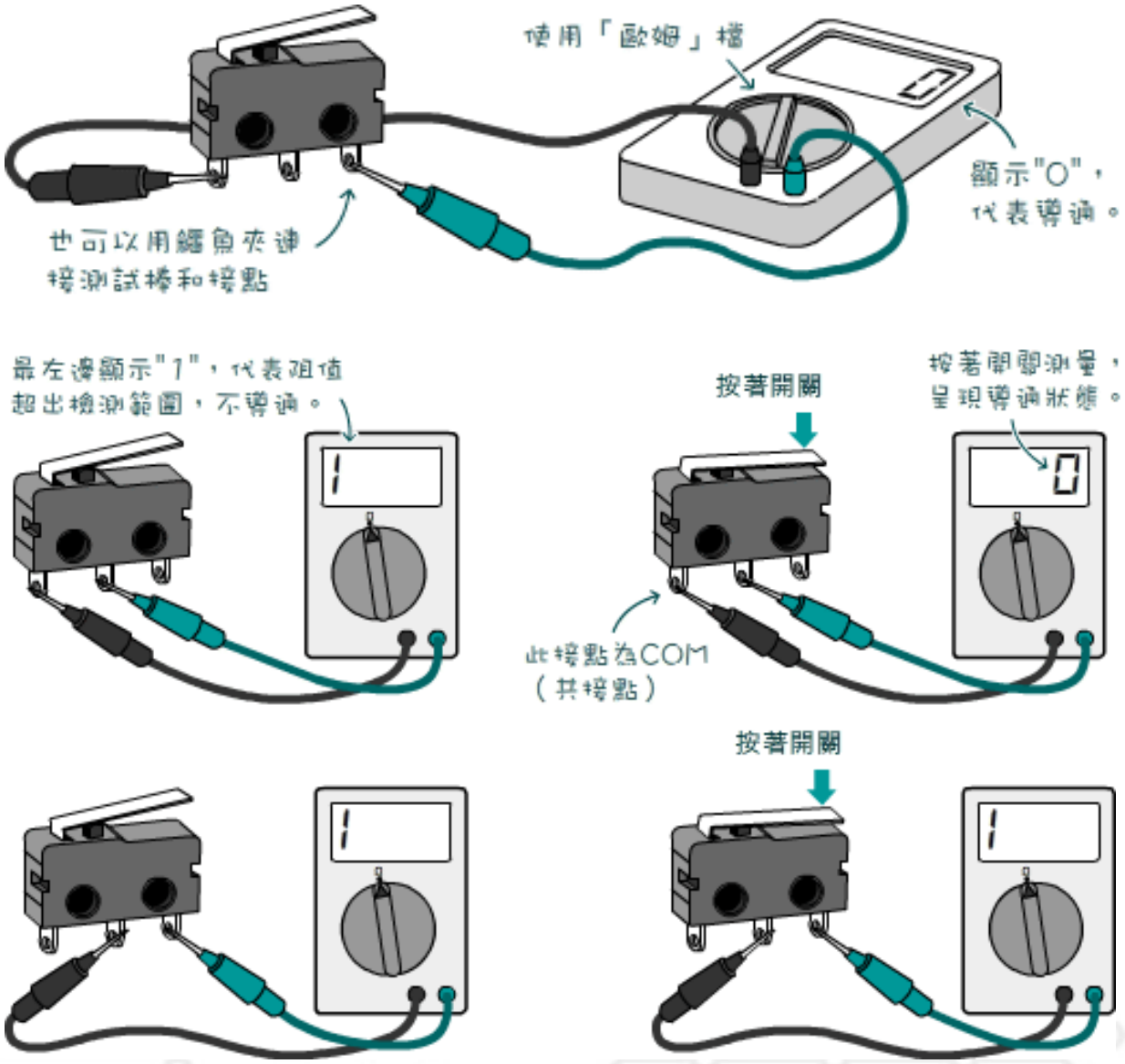
開關的外觀與電路符號



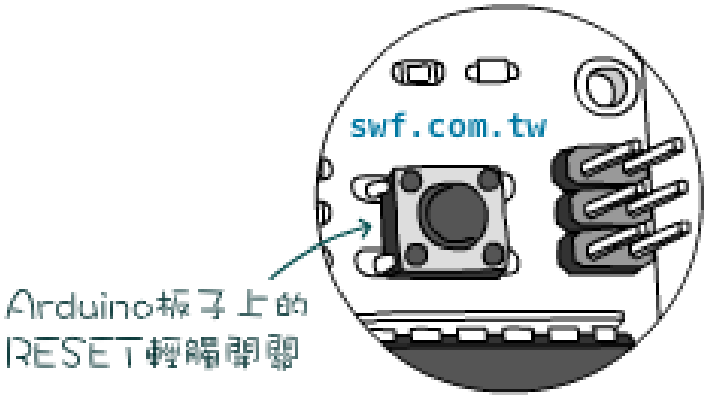


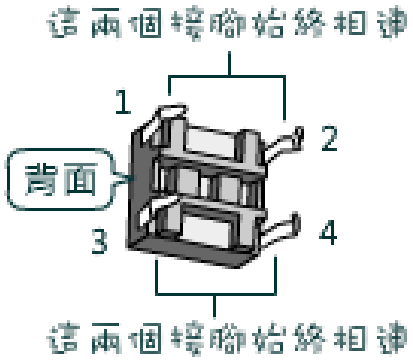


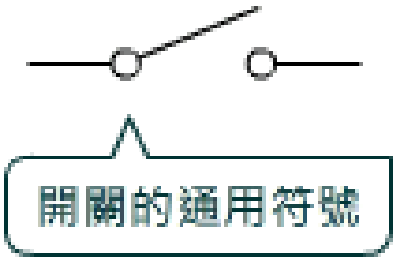
認識開關

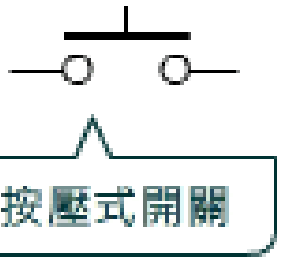

使用「歐姆」檔
顯示"0"，
代表導通。
也可以用鱷魚夾連
接測試棒和接點
最左邊顯示"1"，代表阻值
超出檢測範圍，不導通。
按著開關
按著開關測量，
呈現導通狀態。
此接點為COM
（共接點）
按著開關

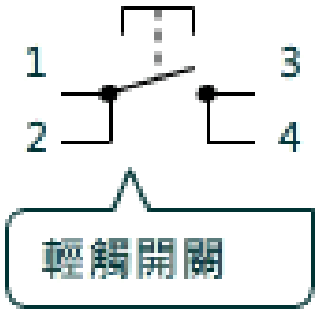
開關也是感測器

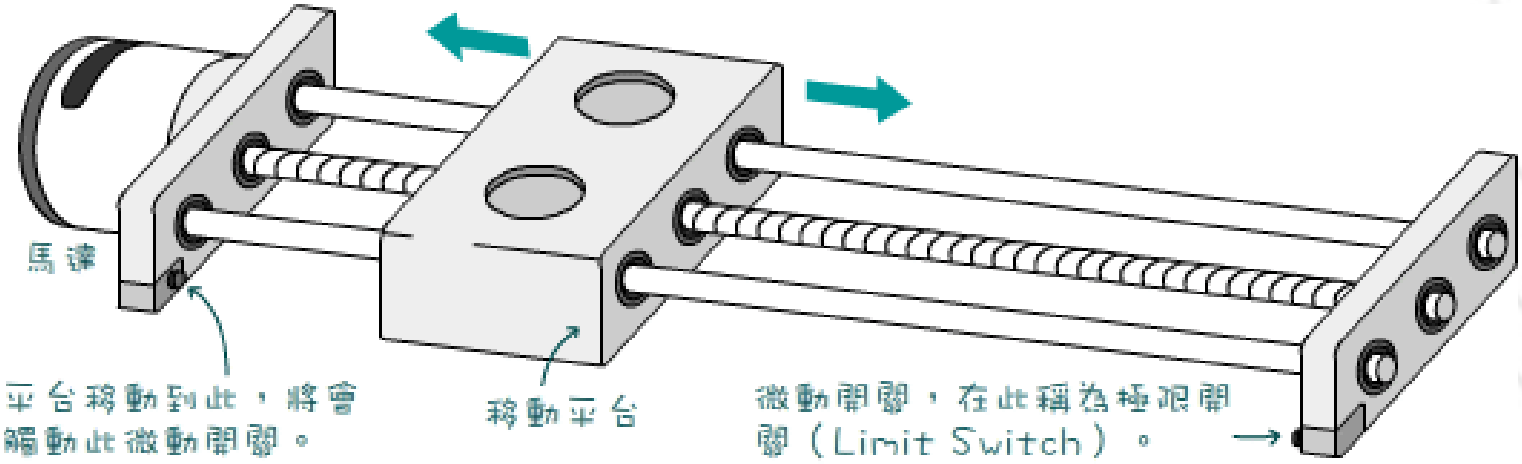

swf.com.tw
Arduino板子上的
RESET輕觸開關


這兩個接腳始終相連
1
2
背面
3
4
這兩個接腳始終相連


開關的通用符號


按壓式開關


1
3
2
4
輕觸開關


馬達
平台移動到此，將會
觸動此微動開關。
移動平台
微動開關，在此稱為極限開
關（Limit Switch）。

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);
```

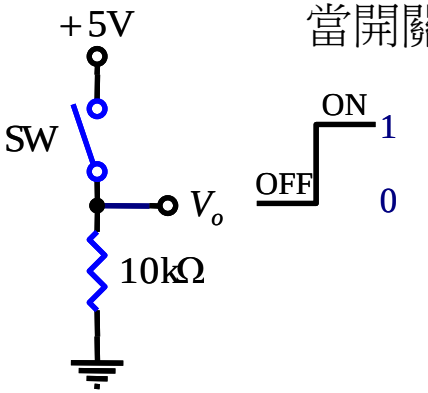
- 電路的接法：

功能說明

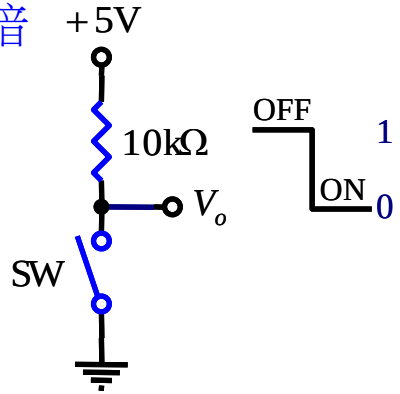
利用一個開關控制一個LED亮與暗

當開關接通(ON)時，LED亮

當開關接通(OFF)時，LED暗



(a) 高電位產生電路



(b) 低電位產生電路

函式說明

digitalRead( )函式

Arduino的digitalRead( )函式功能是在讀取所指定數位輸入腳的狀態，函式只有一個參數pin是在定義數位接腳編號，有兩種輸入狀態：一為高態（HIGH），另一為低態（LOW）。

格式： `digitalRead(pin)`

範例：

```
pinMode(13,INPUT);                  //設定第 13 腳為輸入模式。
int val=digitalRead(13);            //讀取第 13 腳的輸入狀態並存入變數 val 中。
```

實作練習

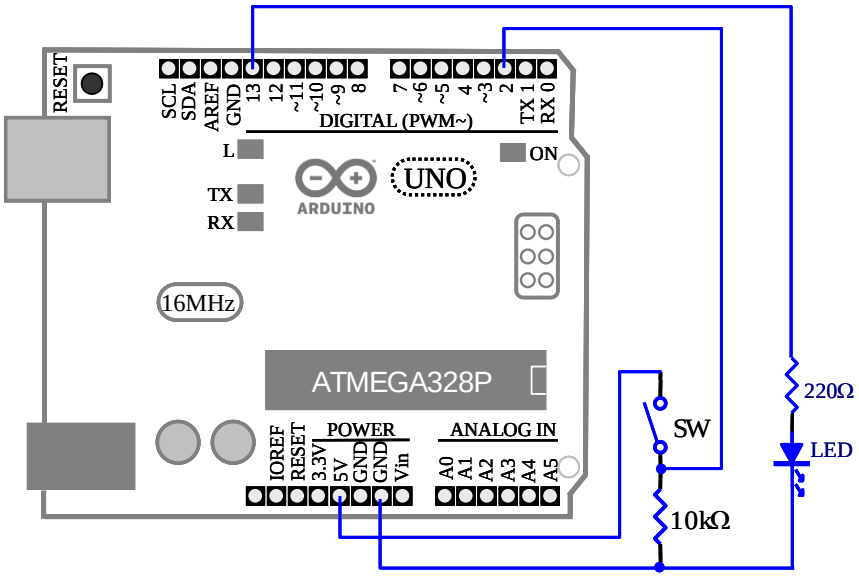
功能說明

利用一個開關控制一個LED亮與暗
當開關接通(ON)時，LED亮
當開關接通(OFF)時，LED暗

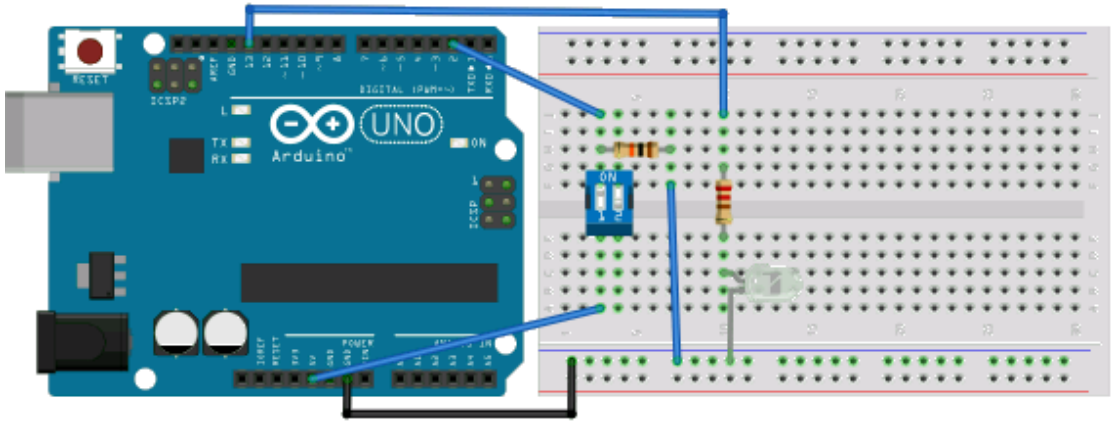
(a) 高電位產生電路　　(b) 低電位產生電路

指撥開關電路

- 電路圖及麵包板接線圖：



一個指撥開關控制 LED 亮與暗實習電路圖



一個指撥開關控制 LED 亮與暗實習麵包板接線圖

實作練習

- 程式：  B121.ino

```
const int sw=2;                    //開關連接至數位接腳 2。
const int led=13;                  //LED 連接至數位接腳 13。
int val;                           //開關狀態。
void setup()
{
    pinMode(sw,INPUT);             //設定數位接腳 2 為輸入模式。
    pinMode(led,OUTPUT);           //設定數位接腳 13 為輸出模式。
}
void loop()
{
    val=digitalRead(sw);           //讀取開關狀態。
    if(val==HIGH)                  //開關接通(ON)？
        digitalWrite(led,HIGH);    //開關接通，點亮 LED。
    else                           //開關斷開(OFF)
        digitalWrite(led,LOW);     //開關斷開，關閉 LED。
}
```

動動腦

- 既然開關可以開關燈那可不可以做其它的控制？
- 例如使LED燈閃爍與暗？
- 例如使LED燈較亮與較暗？

(回家)小練習

1. 設計 Arduino 程式，使用一個指撥開關控制一個 LED，當開關接通(ON)時，LED 閃爍，當開關斷開(OFF)時，LED 關閉。
2. 設計 Arduino 程式，使用一個指撥開關控制一個 LED，當開關接通(ON)時，LED 快速閃爍，反之當開關斷開(OFF)時，LED 慢速閃爍。

實作練習

一個開關控制四個LED移位方向實習

功能說明：

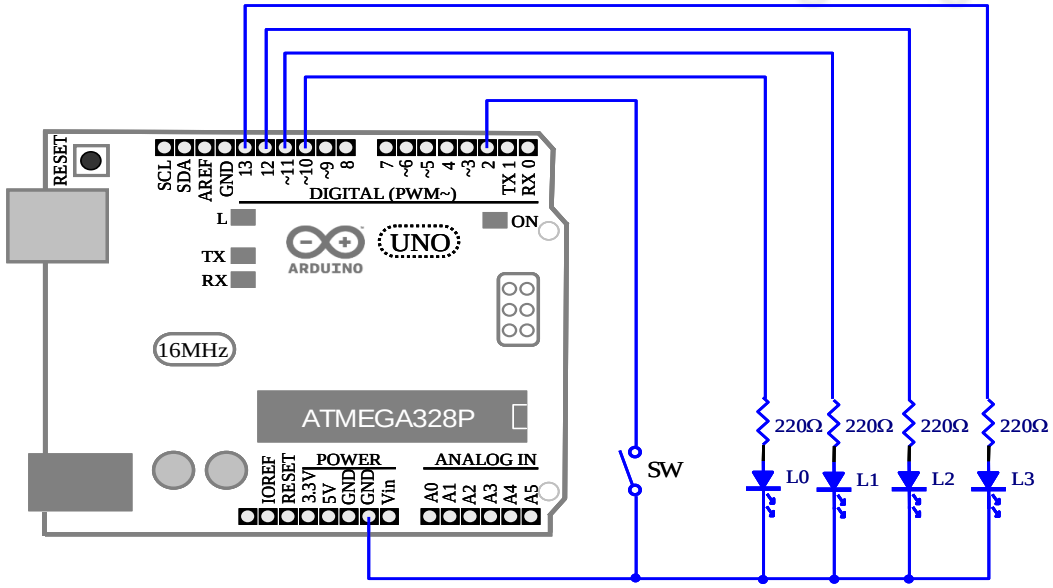
利用一個指撥開關控制四個LED移位方向。

當開關斷開（OFF）時，LED單燈右移

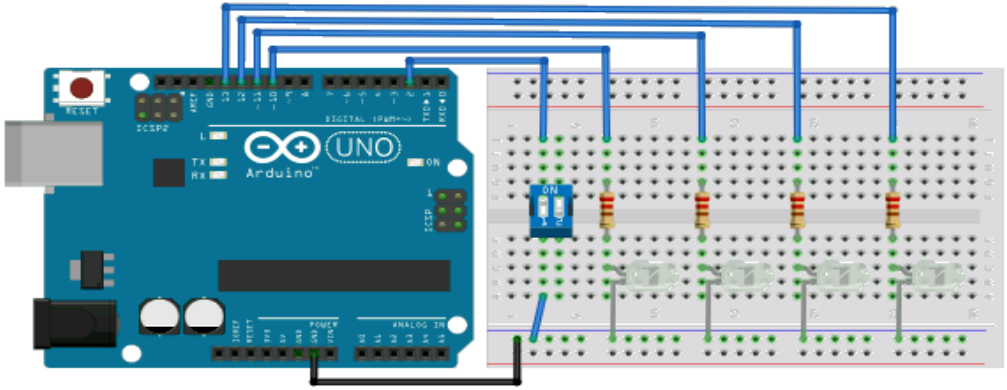
當開關接通（ON）時，LED單燈左移

LED移動速度可以由delay()函式控制，本例使用delay(500)延遲0.5秒。

- 電路圖及麵包板接線圖：



一個指撥開關控制四個 LED 移動方向實習電路圖



一個指撥開關控制四個 LED 移動方向實習麵包板接線圖

實作練習

- 程式：  B122.ino

```
const int sw=2;                          //SW 開關連接至數位接腳 2。
const int led[]={10,11,12,13};           //LED 連接至數位接腳 10~13。
int i;                                   //索引值，第 i 個 LED。
int j=0;                                 //索引值，第 j 個 LED。
int val;                                 //開關狀態。
void setup()
{
    pinMode(sw,INPUT);                   //設定數位接腳 2 為輸入模式。
    for(i=0;i<4;i++)
    {
        pinMode(led[i],OUTPUT);          //設定數位接腳 10~13 為輸出模式。
    }
}
```

```
void loop()
{
    val=digitalRead(sw);              //讀取開關狀態。
    for(i=0;i<4;i++)                  //關閉所有 LED。
    {
       digitalWrite(led[i],LOW);
    }
    if(val==HIGH)                     //開關接通(ON)？
    {
       digitalWrite(led[j],HIGH);     //開關接通，執行 LED 單燈右移。
       delay(500);                    //延遲 0.5 秒。
       if(j==3)                       //LED 已右移至最右方?
          j=0;                        //重新設定 LED 位置在最左方。
       else                           //LED 尚未移至最右方。
          j=j+1;                      //下一個 LED。
    }
    else                              //開關斷開(OFF)，執行 LED 單燈左移 。
    {
       digitalWrite(led[j],HIGH);     //點亮第 j 個 LED。
       delay(500);                    //延遲 0.5 秒。
       if(j==0)                       //LED 已移至最左方?
          j=3;                        //重新設定 LED 位置在最右方。
       else                           //LED 尚未移至最左方。
          j=j-1;                      //下一個 LED。
    }
}
```

(回家)小練習

1. 設計 Arduino 程式，使用一個指撥開關控制四個 LED 移位方向。當開關斷開（OFF）時，LED 單燈閃爍右移；當開關接通（ON）時，LED 單燈閃爍左移。
2. 設計 Arduino 程式，使用一個指撥開關控制四個 LED 移位方向。當開關斷開（OFF）時，LED 單燈右移；當開關接通（ON）時，目前的 LED 單燈閃爍。

一個按鍵開關控制一個LED亮與暗實習

圖 5-12 負脈波產生電路的機械彈跳

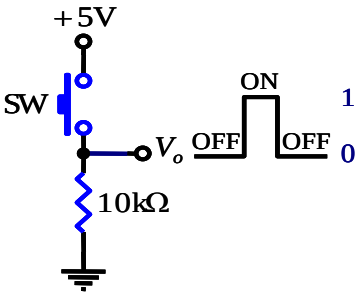
功能說明：

利用一個按鍵開關控制一個LED亮與暗。每按一下按鍵開關，LED的狀態將會改變。若LED原來狀態為暗，按一下按鍵開關時，LED亮，若LED原來狀態為亮，按一下按鍵開關時，LED暗。

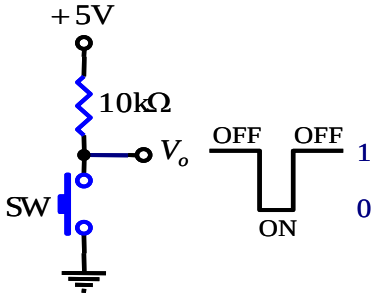
如圖5-11所示為按鍵開關的兩種接線方式

圖5-11(a)為正脈波產生電路，按下開關輸出電壓 為+5V，放開開關輸出電壓 為0V

圖5-11(b)為負脈波產生電路，按下開關輸出電壓 為0V，放開開關輸出電壓 為+5V。



(b) 正脈波產生電路



(c) 負脈波產生電路

按鍵開關電路

函式說明

digitalWrite( )函式

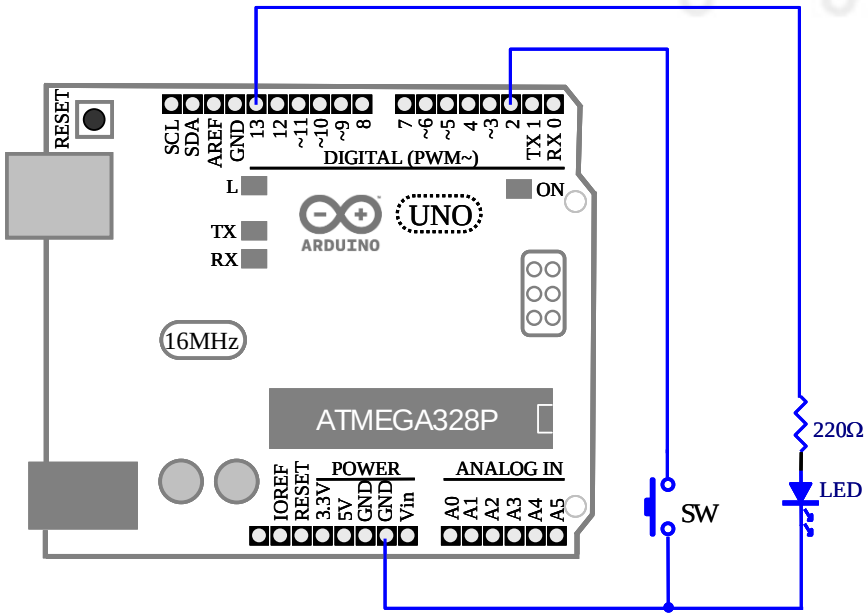
Arduino的digitalWrite( )函式功能是在設定數位接腳的狀態，函式的第一個參數pin是在定義數位接腳編號，第二個參數value是在設定接腳的狀態，有兩種狀態：一為高態（HIGH），另一為低態（LOW）。如果所要設定的數位接腳已經由pinMode( )函式設定為輸入模式，且寫入digitalWrite( )函式的狀態為HIGH時，將會開啟該數位接腳的內部20kΩ上拉電阻。

格式： `digitalWrite(pin,value)`

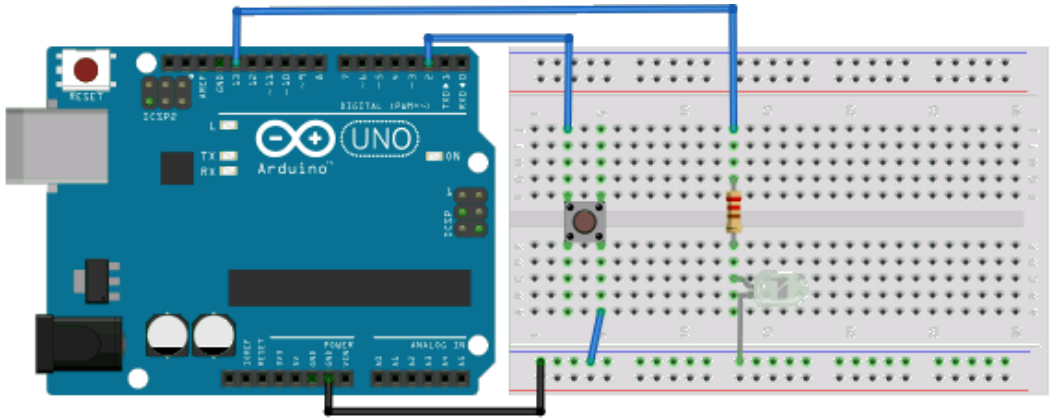
範例：

```
pinMode(2,INPUT);          //設定第 2 腳為輸入模式。
digitalWrite(2,HIGH);      //開啟第 2 腳上拉電阻。
```

- 電路圖及麵包板接線圖：



一個按鍵開關控制一個 LED 亮與暗實習電路圖



一個按鍵開關控制一個 LED 亮與暗實習接線圖

實作練習

- 程式：  B123.ino

```
const int sw=2;                          //按鍵開關連接至數位接腳第 2 腳。
const int led=13;                        //LED 連接至數位接腳第 13 腳。
const int debounceDelay=20;              //按鍵開關穩定所需的時間。
int ledStatus=LOW;                       //LED 初始狀態為 LOW。
int val;                                 //按鍵開關狀態。
void setup()
{
    pinMode(sw,INPUT);                   //設定數位第 2 腳為輸入模式。
    digitalWrite(sw,HIGH);               //開啟數位第 2 腳的內部上拉電阻。
    pinMode(led,OUTPUT);                 //設定數位第 13 腳為輸出模式。
}
```

實作練習

```
void loop()
{
    val=digitalRead(sw);                    //讀取按鍵開關狀態。
    if(val==LOW)                            //按鍵開關被按下?
    {
        delay(debounceDelay);               //消除按鍵開關的不穩定狀態(機械彈跳)。
        while(digitalRead(sw)==LOW)         //按鍵開關已放開?
            ;                               //等待放開按鍵開關。
        ledStatus=!ledStatus;               //改變 LED 狀態。
        digitalWrite(led,ledStatus);        //設定 LED 狀態。
    }
}
```

(回家)小練習

1. 設計 Arduino 程式，使用一個按鍵開關控制一個 LED 閃爍與暗。每按一下按鍵開關，LED 的狀態將會改變。若 LED 原來狀態為暗，按一下按鍵開關時，LED 閃爍，若 LED 原來狀態為閃爍，按一下按鍵開關時，LED 暗。
2. 設計 Arduino 程式，使用一個按鍵開關控制兩個 LED。按一下按鍵開關時，LED 重覆快閃 3 次再將 LED 關閉 1 秒，再按一下按鍵開關時，LED 暗。

回家作業

使雨滴燈可以左右位移模擬霹靂燈

回家作業

一個按鍵開關控制四個LED移位方向

功能說明：

利用一個按鍵開關控制四個LED移位方向。每按一下按鍵開關，LED的移動方向改變

若LED原來移動方向為右移，按一下按鍵開關時，改變為左移；
若LED原來移動方向為左移，按一下按鍵開關時，改變為右移。

同場加映-中斷

以上按鍵程式的邏輯很簡單，每按一下按鍵開關，讓LED的移動方向改變，但是有時要按久一點，才會生效，因為要讓LED跑完一輪才會執行到讀取按鍵開關的程式碼，那有沒有辦法讓按鍵開關一按下去就生效呢？

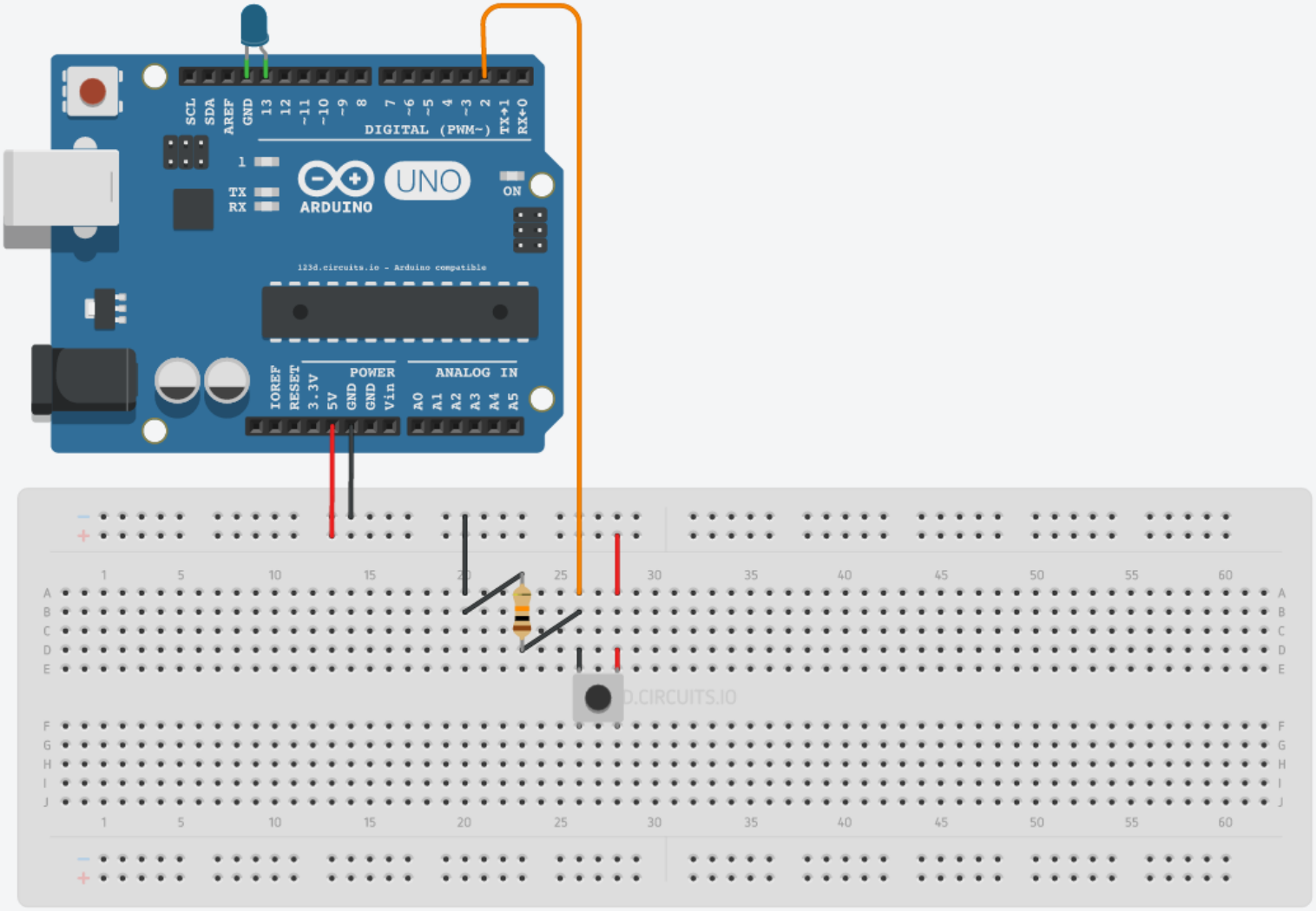
使用中斷！

使Arduino中斷正在執行中的程式，去執行你所指定的程式，執行完後再回來執行原本正在執行中的程式，就像你正在家裡看電視，突然家裡的電話響了，你起身去接電話，接完電話後繼續回來看電視一樣。

支援中斷的腳位

不同的Arduino支援中斷的腳位並不相同

UNO有兩個可以使用的外部中斷，腳位2（ 中斷0 ）和腳位3（中斷1）


SCL
SDA
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
123d.circuits.io - Arduino compatible
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN

中斷命令

attachInterrupt(中斷編號, 執行函式, 觸發模式);

attachInterrupt() 函式的用途是用來指定外部中斷的處理函式(Interrupt Service Routine, ISR)，就像範例程式所示範的指定 buttonStateChanged() 當作 Interrupt 0 外部中斷的處理函式。

attachInterrupt() 函式有三個參數:

1. interrupt: 外部中斷的編號。大部份 Arduino 板子都有兩個外部中斷，編號 0 (Interrupt 0)是在 pin 2 上，而編號 1 (Interrupt 1)是在 pin 3 上。
2. function: 中斷處理函式(Interrupt Service Routine, ISR)。中斷處理函式必須是沒有參數而且不回傳任何東西。
3. mode: 定義什麼狀況下該觸發中斷，有四個可以設定的常數值:
   - **LOW**: 當 pin 為 LOW 時觸發中斷
   - **CHANGE**: 當 pin 狀態改變時觸發中斷，不管是從 HIGH 到 LOW 或從 LOW 到 HIGH
   - **RISING**: 當 pin 狀態從 LOW 到 HIGH 時觸發中斷，RISING 又稱正緣觸發
   - **FALLING**: 當 pin 狀態從 HIGH 到 LOW 時觸發中斷，FALLING 又稱負緣觸發

注意

- 在中斷函式中，delay()函式將不再有作用
- 要在中斷函式內更改的值需要宣告為volatile類型
- 中斷函式通常是短小、執行效率比較高的函式。

B124.ino

```
const int ledPin = 13;                          // LED
volatile int letStatus = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);                      // 把 ledPin 設置成 OUTPUT
  attachInterrupt(0, buttonStateChanged, CHANGE);
}
void loop() {
    digitalWrite(ledPin, letStatus);
    delay(10000);                               // delay一個很長的時間模擬程式執行
}
void buttonStateChanged() {
  letStatus = ~letStatus;                       // 把 led 的狀態反過來
  digitalWrite(ledPin, letStatus);
}
```