

# 4

## 七段顯示器 與數字鍵盤

1 單一七段顯示器

2 四位七段顯示器

3 數字鍵盤

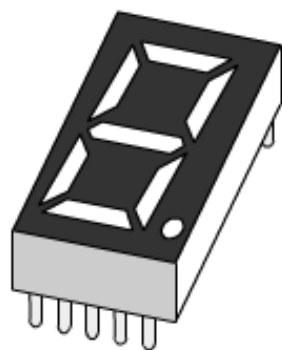
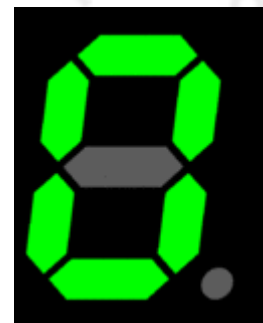
講師 張傑帆 Chang, Jie-Fan

**七段顯示器**（英語：Seven-segment display）為常用顯示數字的電子元件。因為藉由七個發光二極體以不同組合來顯示數字

**數字鍵盤**又稱九宮鍵盤，薄膜式（Membrane）鍵盤中有一整張雙層膠膜，通過膠膜提供按鍵的回彈力，利用薄膜被按下時按鍵處碳心於線路的接觸來控制按鍵觸發。這種鍵盤的成本十分低，市面上絕大部份鍵盤都是膜式鍵盤。

# 七段顯示器

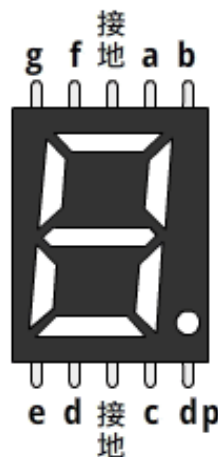
七段顯示器是內建八個LED的顯示元件，為了方便解說，內部LED分別標上a~g和dp（點）代號。



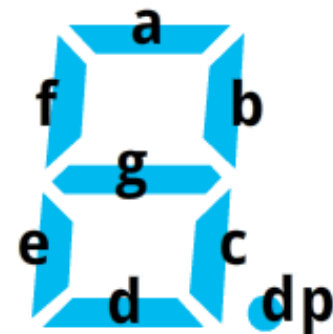
七段顯示器



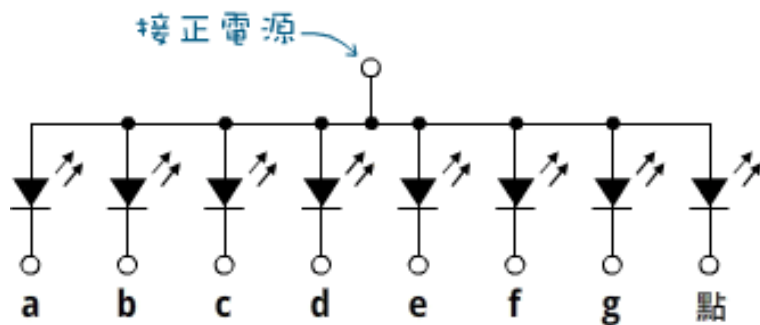
共陽極腳位



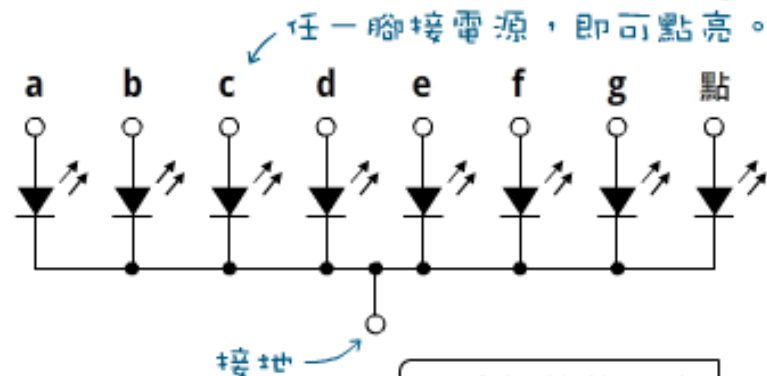
共陰極腳位



內部LED的編號



共陽極等效電路



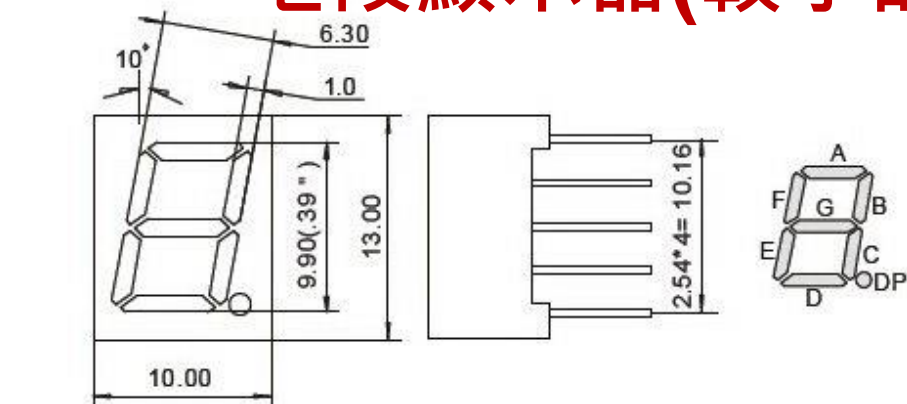
共陰極等效電路



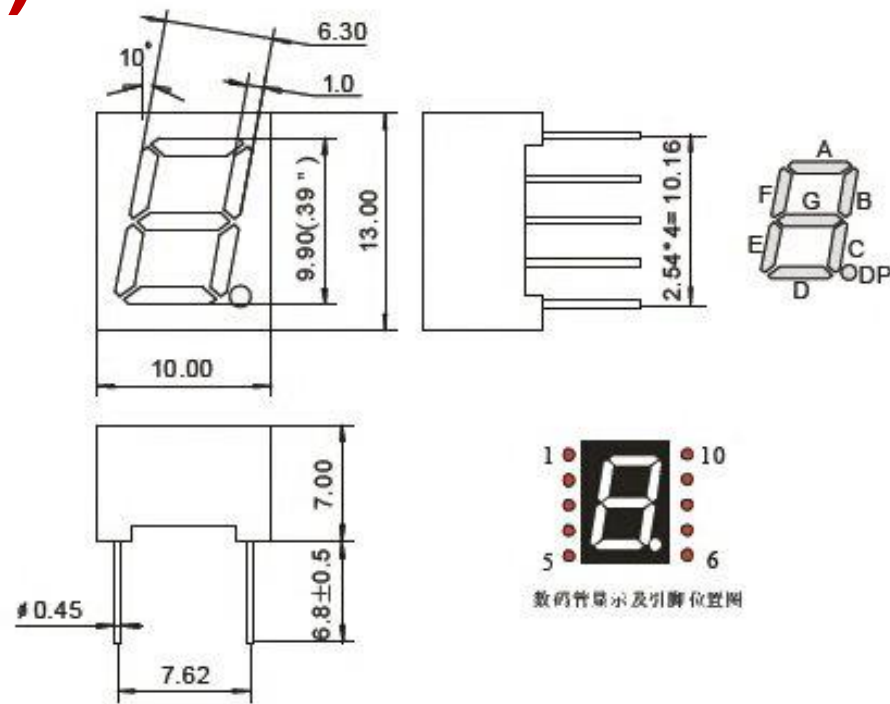
3191B

# 七段顯示器(較小的)

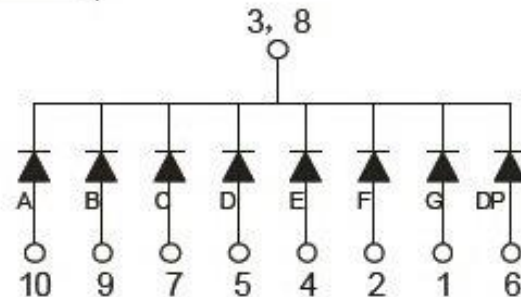
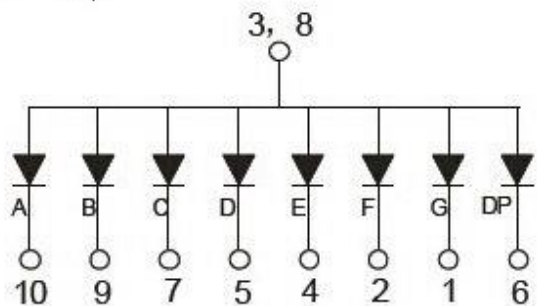
3191A



數碼管顯示及引腳位置圖

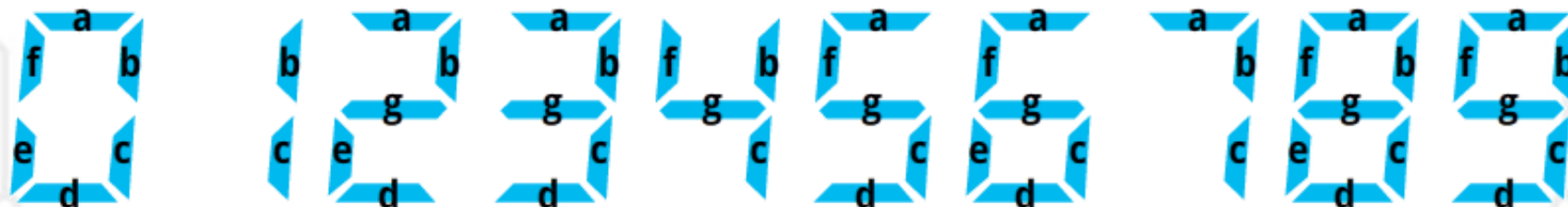


數碼管顯示及引腳位置圖



# 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

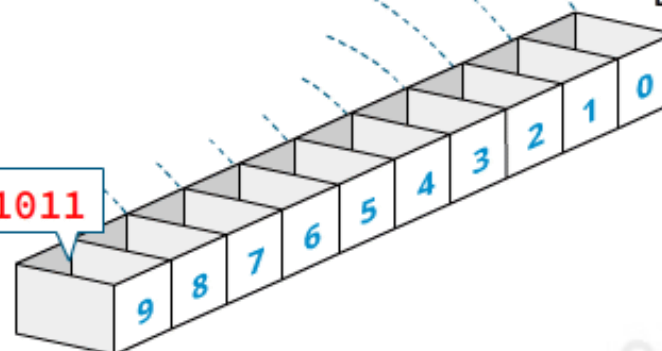
高位元 (最高位元  
沒用到，補上0) 低位元

B00110000

B01111110

LEDs陣列

B01111011



# 認識七段顯示器

表7-1 共陽極七段顯示器字型碼

字型	p	g	f	e	d	c	b	a	字型	p	g	f	e	d	c	b	a
	1	1	0	0	0	0	0	0		1	0	0	1	0	0	1	0
	1	1	1	1	1	0	0	1		1	0	0	0	0	0	1	0
	1	0	1	0	0	1	0	0		1	1	1	1	1	0	0	0
	1	0	1	1	0	0	0	0		1	0	0	0	0	0	0	0
	1	0	0	1	1	0	0	1		1	0	0	1	0	0	0	0

# 認識七段顯示器

表7-2 共陰極七段顯示器字型碼

字型	p	g	f	e	d	c	b	a	字型	p	g	f	e	d	c	b	a
	0	0	1	1	1	1	1	1		0	1	1	0	1	1	0	1
	0	0	0	0	0	1	1	0		0	1	1	1	1	1	0	1
	0	1	0	1	1	0	1	1		0	0	0	0	0	1	1	1
	0	1	0	0	1	1	1	1		0	1	1	1	1	1	1	1
	0	1	1	0	0	1	1	0		0	1	1	0	1	1	1	1

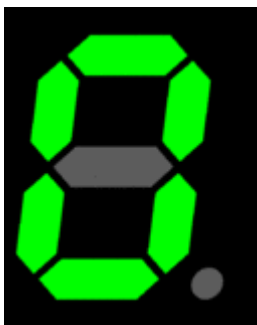


# 實作練習

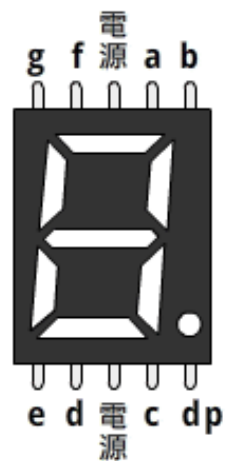
## 一位七段顯示 **0~9** 計數實習

### □ 功能說明：

使用 Arduino 板控制一位七段顯示器顯示 0~9 上數計數。因為是使用共陽極七段顯示器，所以“com”腳必須連接至+5V 電源，再依表 7-1 所示，將 0~9 字型碼由數位接腳 2~9 輸出至顯示器。



□ 電路圖及麵包板接線圖：



共陽極腳位

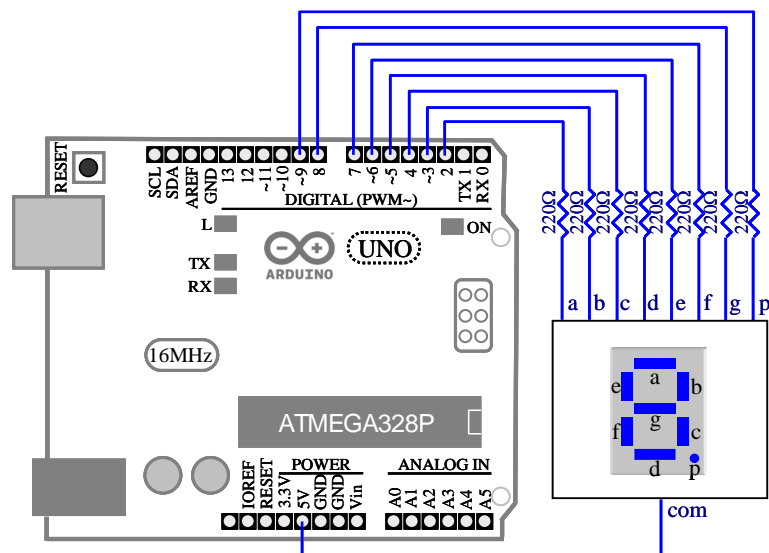


圖 7-3 一位七段顯示 0~9 計數實習電路圖



數碼管顯示及引腳位置圖

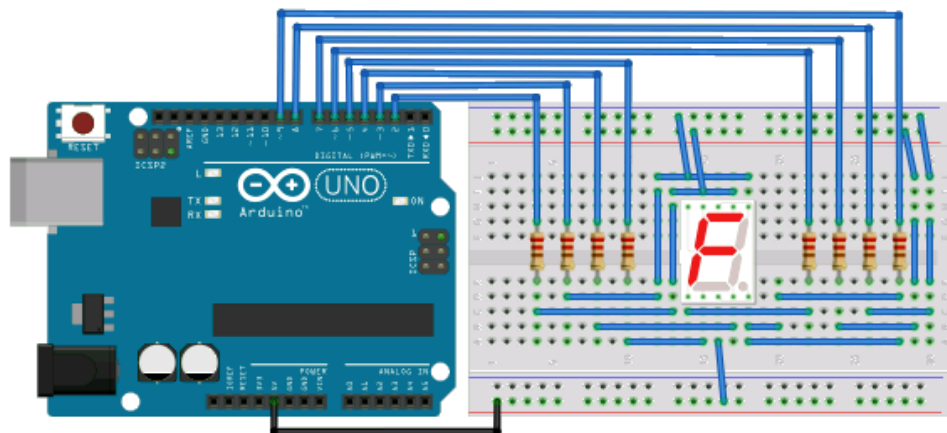


圖 7-4 一位七段顯示 0~9 計數實習麵包板接線圖



# 函式說明

## **bitRead( )** 函式

BitRead( ) 函式的功用是在讀取變數的某一個位元的值，不是 0 就是 1。有兩個參數必須設定，第一個參數 *x* 為變數，第二個參數 *n* 指定所要讀取變數的某一個位元，*n*=0 代表最小有效位元 ( the least-significant )。

格式： `bitRead(x,n)`

---

範例： `int x=B01010101;` //設定 *x* 整數變數初值。

---

`bitRead(x,0);` //讀取 *x* 變數位元 0 的值。

---

# 實作練習

□ 程式： B401.ino

int i;	//數字碼 0~9 的索引值。
int j;	//位元 0~7 的索引值。
const byte num[10]=	//0~9 顯示碼 pgfedcba。
{ B11000000, B11111001,	//0,1
B10100100, B10110000,	//2,3
B10011001, B10010010,	//4,5
B10000010, B11111000,	//6,7
B10000000, B10010000 };	//8,9
const int seg[]={2,3,4,5,6,7,8,9};	//顯示器各段 abcdefgp 數位接腳。
void setup()	
{	
for(i=0;i<8;i++)	
pinMode(seg[i],OUTPUT);	//設定數位腳 4~11 為輸出模式。
}	

```
void loop()  
{  
    for(i=0;i<10;i++)                //數字 0~9。  
    {  
        for(j=0;j<8;j++)              //各段位元 0~7。  
        {  
            if(bitRead(num[i],j))  
                digitalWrite(seg[j],HIGH); //若位元值為 1，設定顯示器該小段為 HIGH  
            else  
                digitalWrite(seg[j],LOW); //若位元值為 0，設定顯示器該小段為 LOW  
        }  
        delay(1000);                  //延遲 1 秒。  
    }  
}
```

# 實作練習



1. 設計 Arduino 程式，控制一位七段顯示器下數並顯示 9~0。
2. 設計 Arduino 程式，控制一位七段顯示器閃爍上數並顯示 0~9。

# 實作練習

## 按鍵開關控制一位七段顯示器上下計數實習

### □ 功能說明：

使用 Arduino 板讀取按鍵開關控制一位七段顯示器上、下數變化。每按一下開關，顯示器會改變計數狀態，若原先為上數，改變為下數；若原先為下數，則改變為上數。

□ 電路圖及麵包板接線圖：

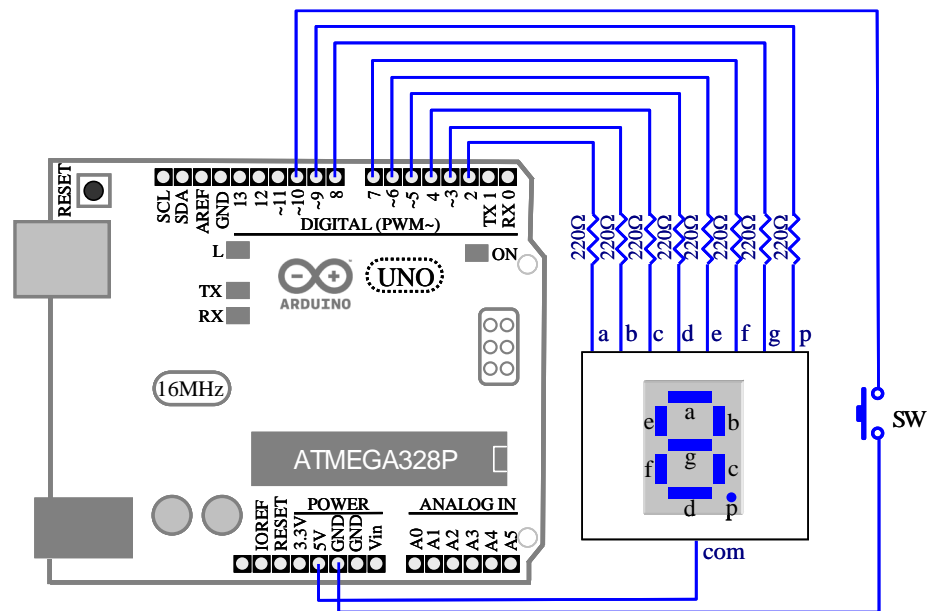


圖 7-5 一個按鍵開關控制一位七段顯示器上下數實習電路圖

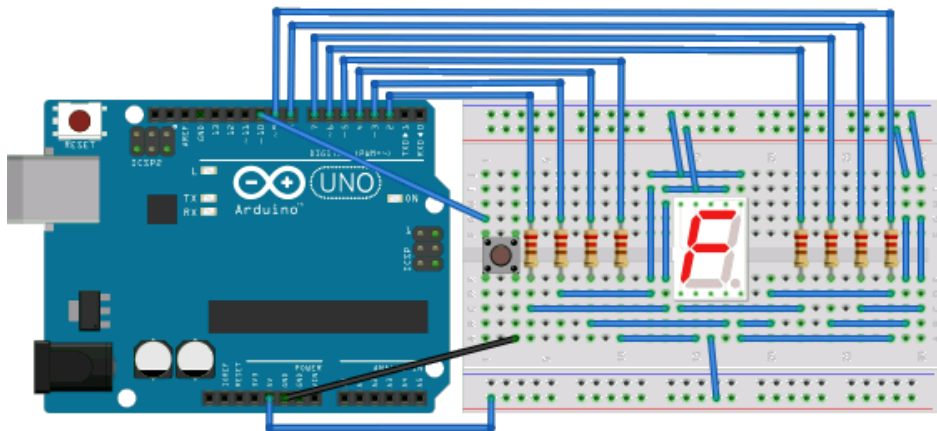


圖 7-6 一個按鍵開關控制一位七段顯示器上下數實習麵包板接線圖



# 實作練習

□ 程式： B411.ino

int i;	// 索引值。
int KeyData;	// 按鍵值。
int numKeys=0;	// 按鍵次數。
int val=0;	// 顯示值。
const int debounceDelay=20;	// 開關穩定所需要的時間 20ms。
const byte num[10]=	// 0~9 顯示碼。
{ B11000000, B11111001,	// 0, 1
B10100100, B10110000,	// 2, 3
B10011001, B10010010,	// 4, 5
B10000010, B11111000,	// 6, 7
B10000000, B10010000 };	// 8, 9
const int seg[]={2,3,4,5,6,7,8,9};	// 七段顯示 abcdefgp 段連接腳位。
const int sw=10;	// 按鍵開關連接至數位接腳 10。
void setup()	
{	
pinMode(sw, INPUT_PULLUP);	// 設定數位接腳 10 為含提升電阻輸入模式。
for(i=0; i<8; i++)	
pinMode(seg[i], OUTPUT);	// 設定數位接腳 2~9 為輸出模式。
}	

```
void loop()  
{  
    KeyData=digitalRead(sw);          //讀取按鍵。  
    if (KeyData==LOW)                 //按鍵被按下?  
    {  
        delay(debounceDelay);        //延遲 20ms 消除機械彈跳。  
        while (digitalRead(sw)==LOW)  //按鍵未放開?  
            ;                          //等待放開按鍵。  
        numKeys++;                    //按鍵次數加 1。  
    }  
}
```

if(numKeys%2==0)	//按鍵次數為偶數?
{	
val++;	//顯示值上數加1。
if(val>9)	//顯示值大於9?
val=0;	//重新設定顯示值為0。
}	
else	//按鍵次數為奇數。
{	
val--;	//顯示值下數減1。
if(val<0)	//顯示值低於0?
val=9;	//重新設定顯示值為9。
}	
for(i=0;i<8;i++)	//設定顯示器各段狀態。
{	
if(bitRead(num[val],i))	//段位元資料為1?
digitalWrite(seg[i],HIGH);	//設定段狀態為HIGH。
else	//段位元資料為0?
digitalWrite(seg[i],LOW);	//設定段狀態為LOW。
}	
delay(1000);	//延遲1秒。
}	

# (回家)小練習



## 練習

1. 設計 Arduino 程式，使用按鍵開關控制一位七段顯示器閃爍上、下數變化。按鍵會改變原來的計數狀態，即上、下數切換。
2. 設計 Arduino 程式，控制一位七段顯示器閃爍上數、下數及停止等變化。

# 實作練習

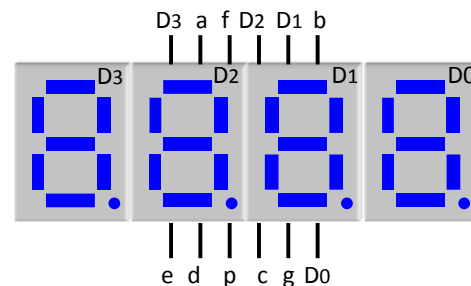
## 四位七段顯示器0000~9999計數實習

功能說明：

使用Arduino板控制四位七段顯示器上數計數並顯示0000~9999。如圖7-7所示四位七段顯示器元件及正面接腳圖，各相同段連接在一起，並且以D3~D0來驅動，其中D3驅動最左邊顯示器，而D0驅動最右邊顯示器，因為是使用PNP電晶體，因此Arduino板輸出低電位可驅動電晶體導通。



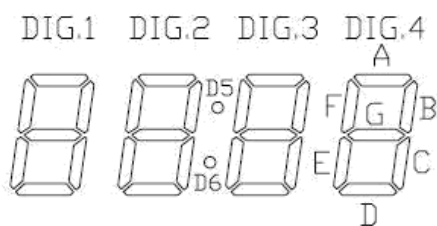
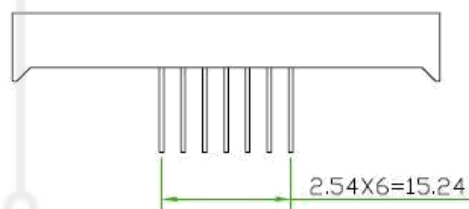
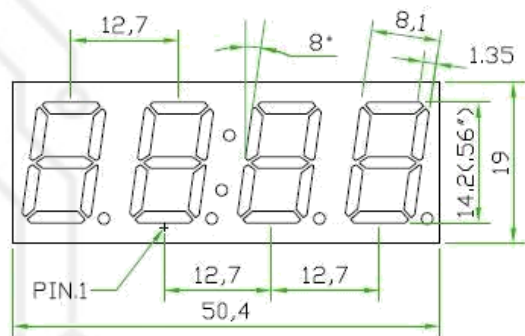
(a) 元件



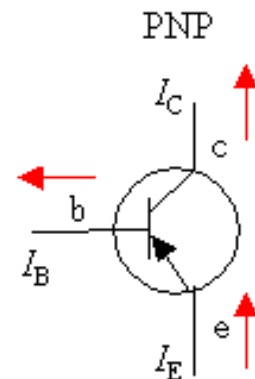
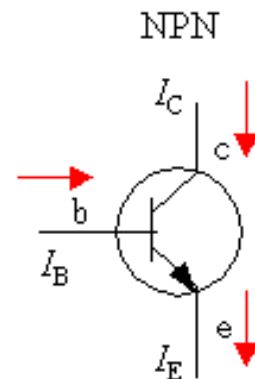
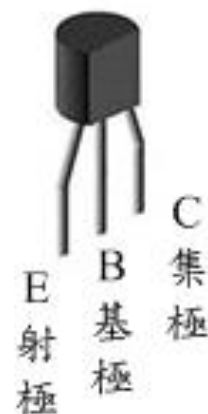
(b) 正面接腳



# 共陽極七段顯示器接腳



電晶體 (**transistor**) 是一種固態半導體元件，可以用於放大、開關、穩壓、訊號調變和許多其他功能。

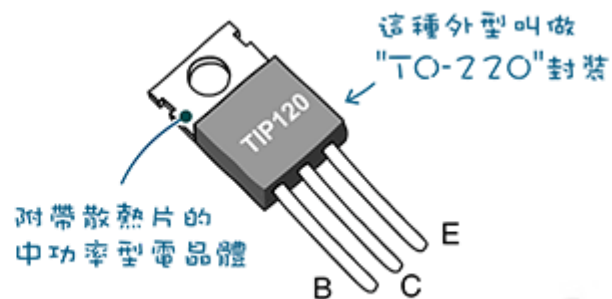
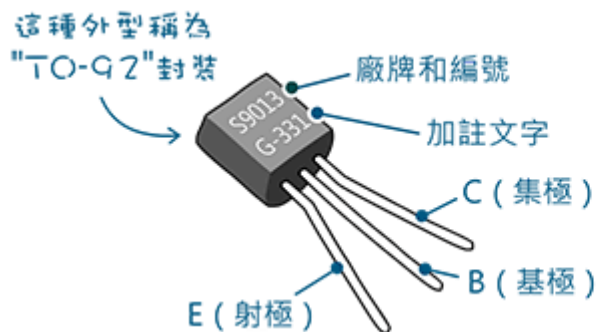
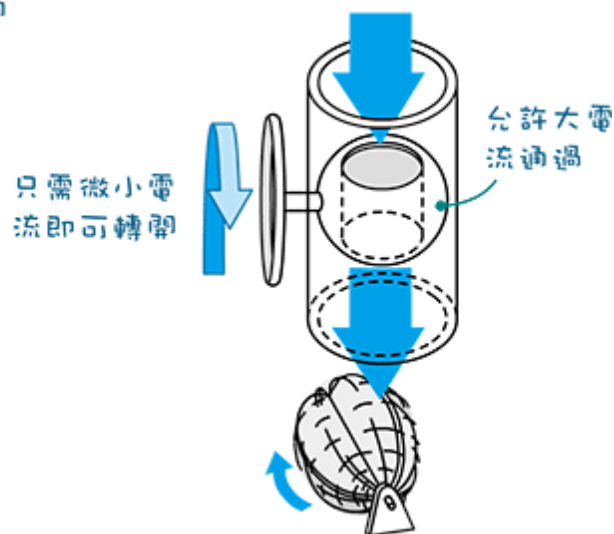
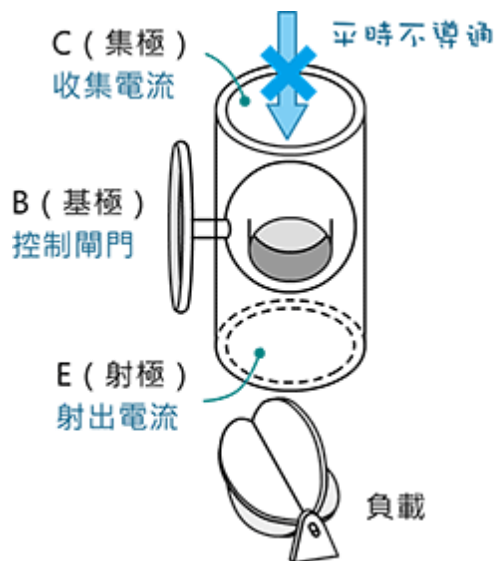




# 認識電晶體元件

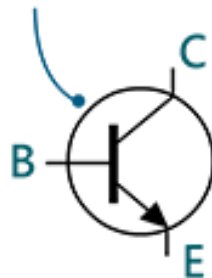
電晶體有三隻接腳，分別叫做B（基極），C（集極）和E（射極）。

圖為NPN

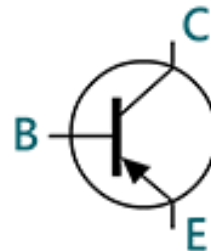


# NPN與PNP類型的電晶體

圖圈可省略不畫



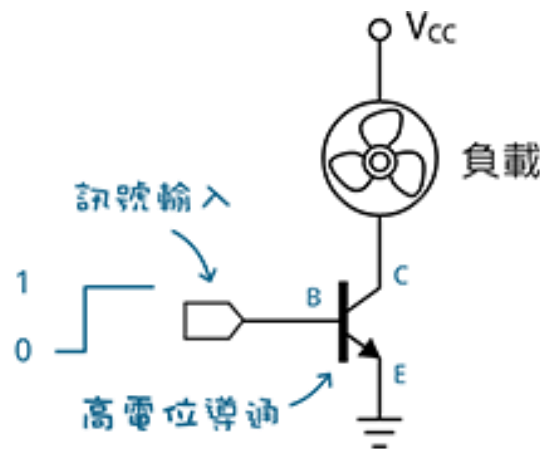
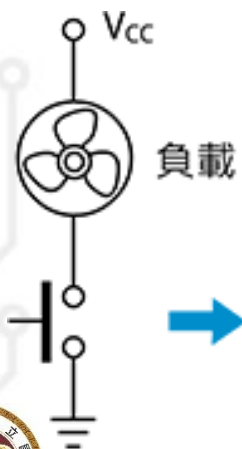
NPN型



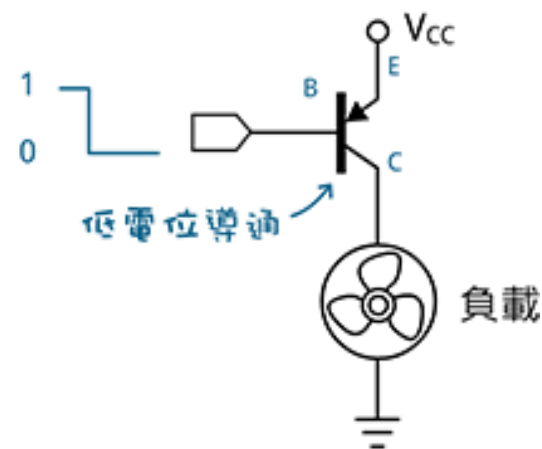
PNP型

## 電晶體原理

當NPN型電晶體的B腳（基極）接上高電位時（例如：正電源），電晶體將會導通，驅動負載；相反地，當PNP型電晶體的基極）接上低電位時（例如：接地），電晶體才會導通。



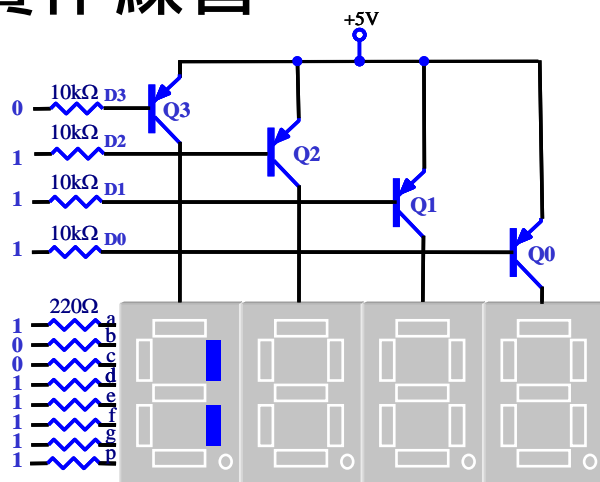
NPN型的接法



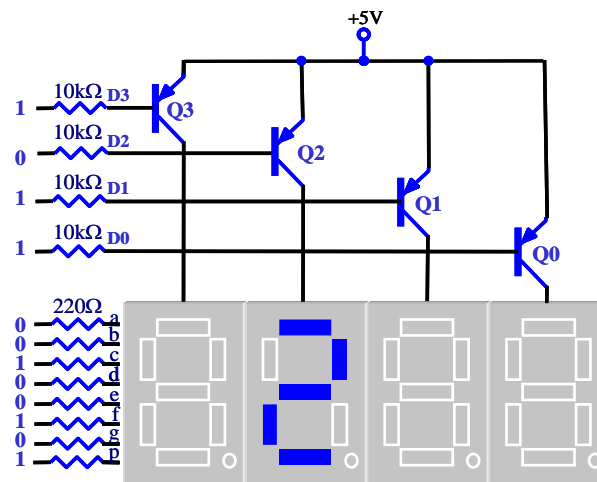
PNP型的接法



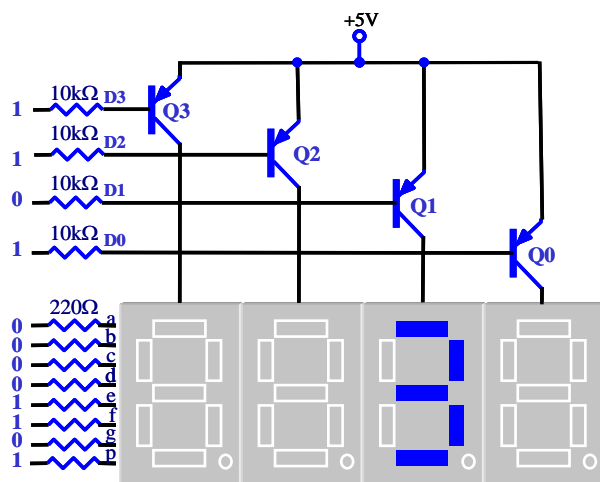
# 實作練習



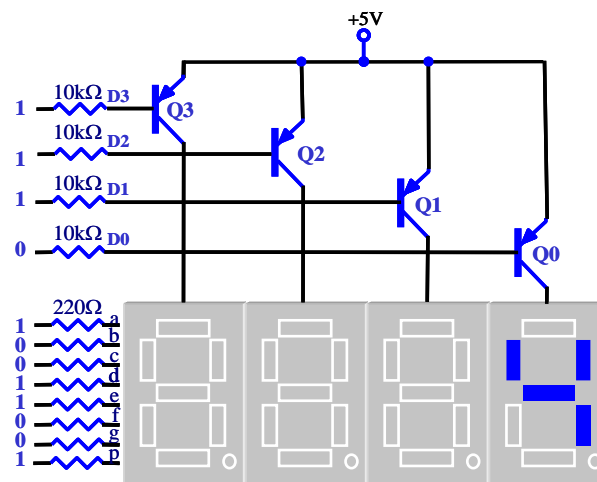
(a) 第一次掃描



(b) 第二次掃描



(c) 第三次掃描



(d) 第四次掃描

# 實作練習

- 電路圖及麵包板接線圖：

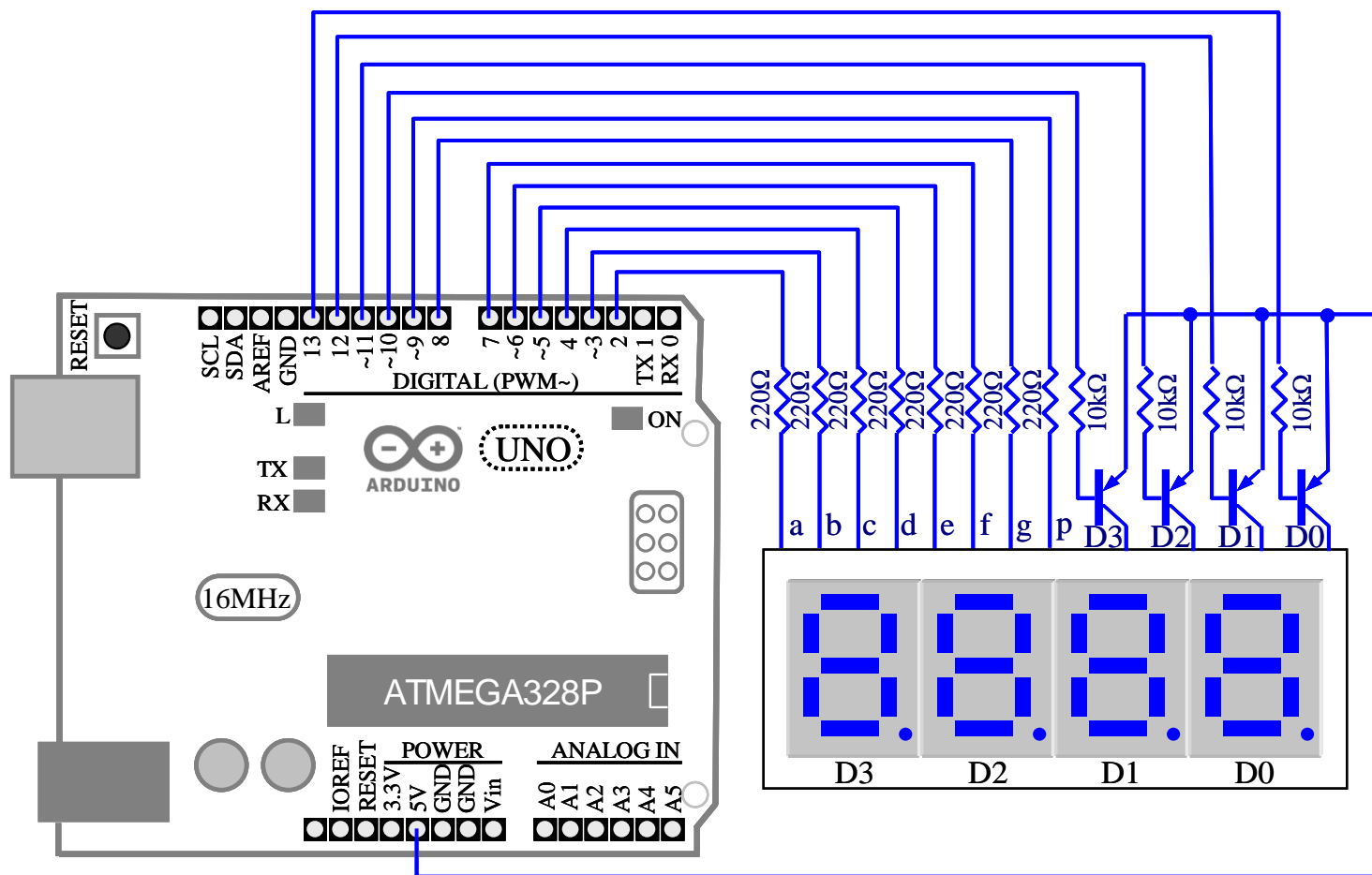
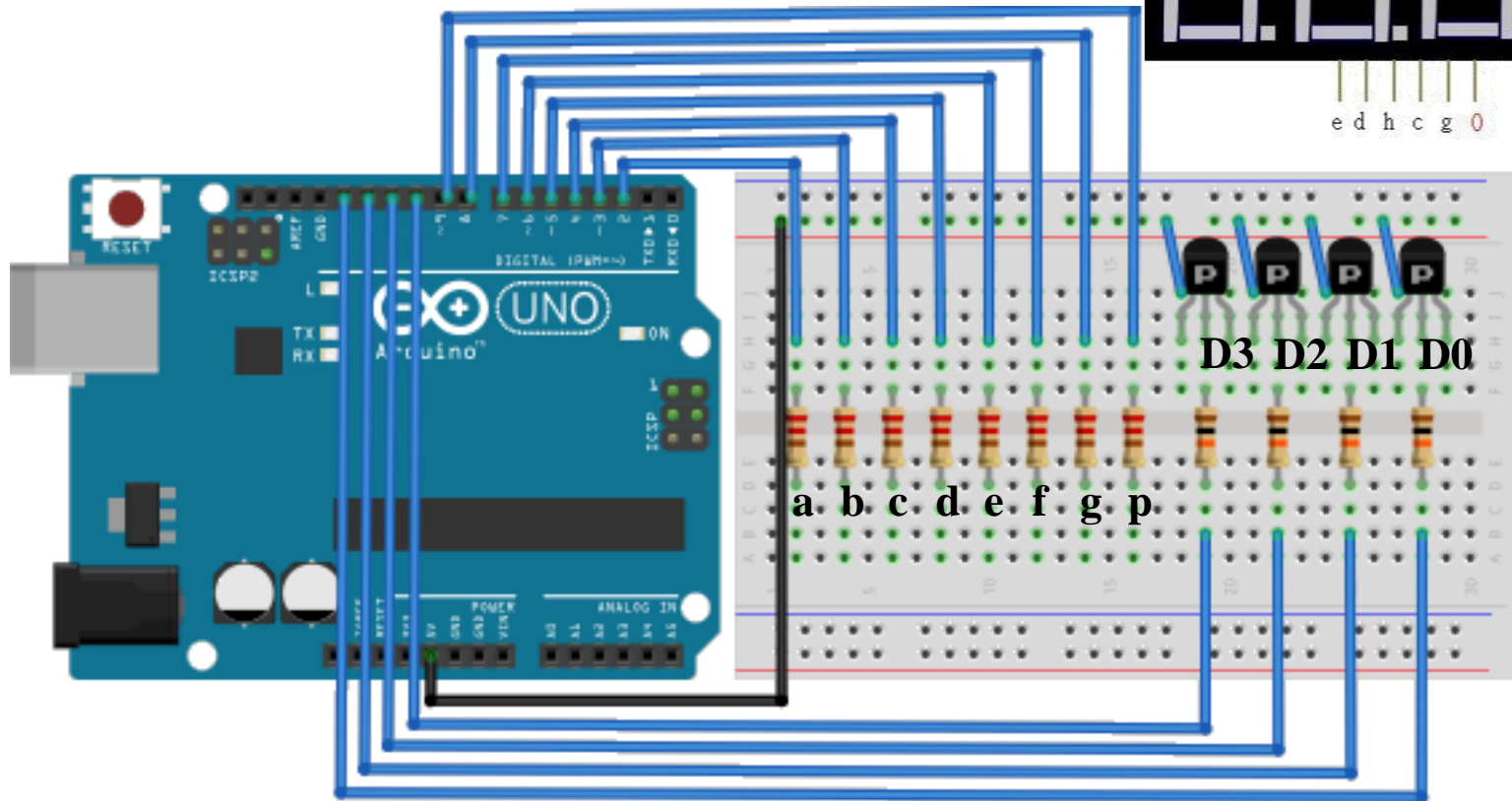


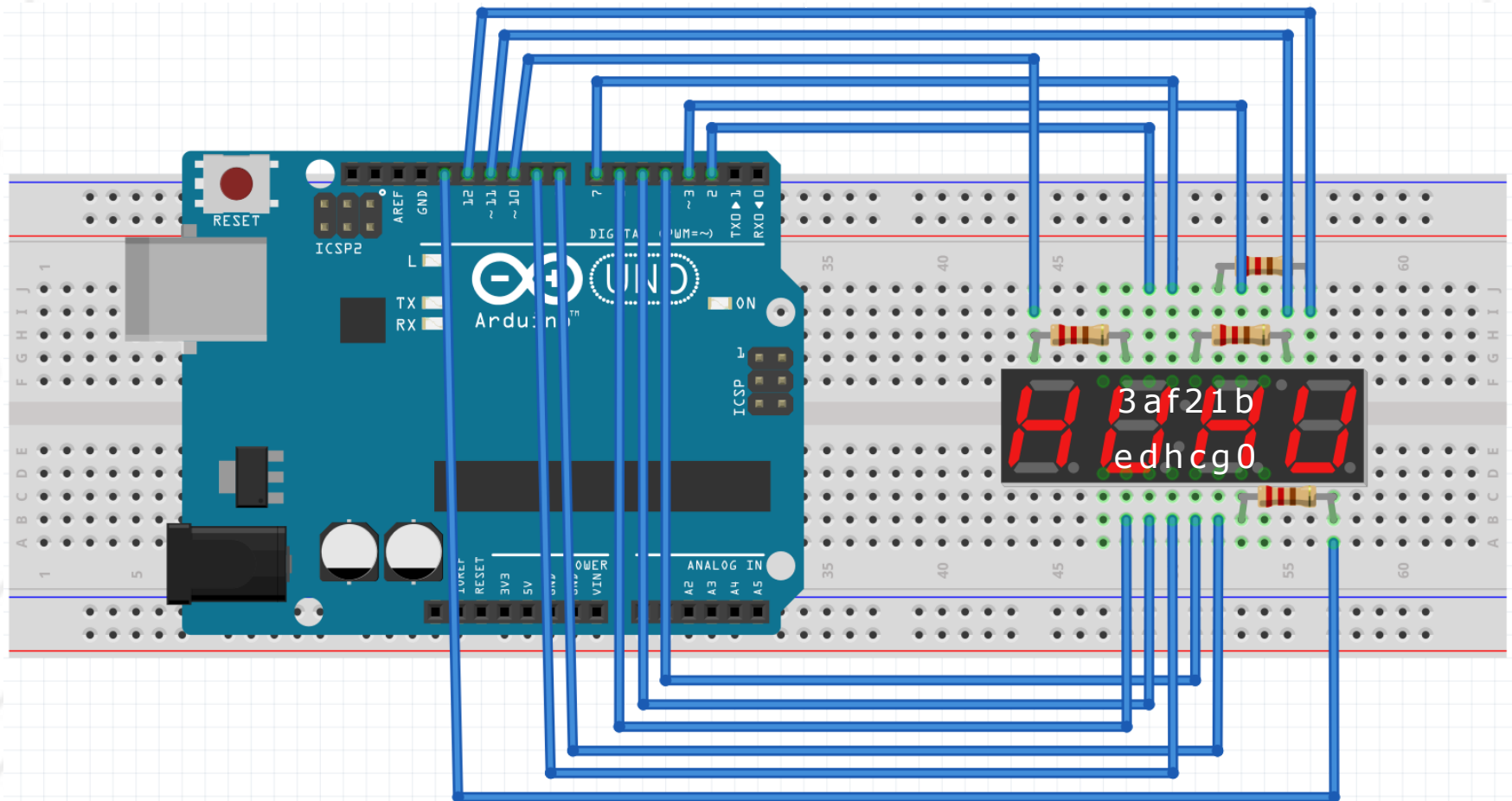
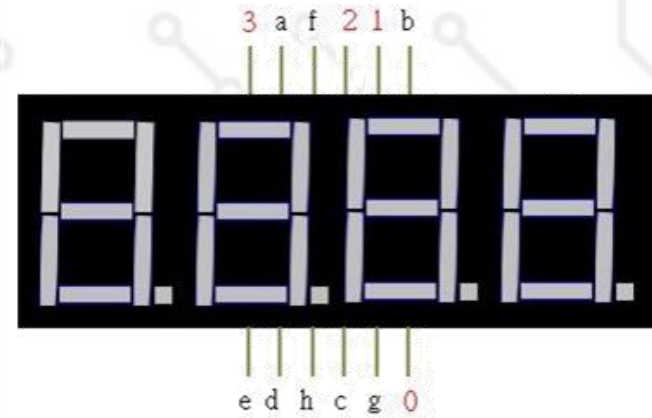
圖 7-9 四位七段顯示器 0000~9999 計數實習電路圖

# 實作練習



四位七段顯示器 0000~9999 計數實習接線圖

# 實作練習-簡化電路





# 函式說明

## **millis( )**函式

Arduino 的 `millis( )` 函式功能是在測量 Arduino 板開始執行至目前為止所經過的時間，這個函式沒有參數，但有一個傳回值，其資料型態為 `unsigned long`，可以測量的範圍為  $0 \sim 2^{32} - 1$ ，最大約 50 天（石英晶體頻率為 16MHz）。

格式： `millis( )`

---

範例： `unsigned long time;` //定義資料型態為 `unsigned long` 的變數。

---

`time=millis( );` //傳回 Arduino 板開始執行至目前為止的時間

---

## **micros( )**函式

Arduino 的 `micros( )` 函式功能是在測量 Arduino 板開始執行至目前為止所經過的時間，單位 `μs`，這個函式沒有參數，但有一個傳回值，其資料型態為 `unsigned long`，可以測量的範圍為  $0 \sim (2^{32} - 1)$ ，最大約 70 毫秒（石英晶體頻率為 16MHz）。

格式： `micros( )`

---

範例： `unsigned long time;` //定義資料型態為 `unsigned long` 的變數。

---

`time=micros( );` //傳回 Arduino 板開始執行至目前為止的時間。

---

# 實作練習

□ 程式： B421.ino

int i,j;	//索引值
int count=0;	//0000~9999 顯示值
int number;	//0000~9999 顯示值
unsigned long time=0;	//計時
const byte num[10]=	//0~9 顯示碼
{ B11000000, B11111001, B10100100, B10110000, B10011001,	
B10010010, B10000010, B11111000, B10000000, B10010000};	
const int seg[]={2,3,4,5,6,7,8,9};	//abcdefghp
const int digit[]={10,11,12,13};	//D1~D4。
void setup()	
{	
for(i=0;i<8;i++)	
pinMode(seg[i],OUTPUT);	//設定數位接腳 2~9 為輸出模式。
for(i=0;i<4;i++)	
{	
pinMode(digit[i],OUTPUT);	//設定數位接腳 10~13 為輸出模式。
digitalWrite(digit[i],HIGH);	//掃描信號初值。
}	
}	

```
void loop()
{
    number=count;
    for (i=3;i>=0;i--) //四位數 D1~D4 。
    {
        for (j=0;j<8;j++) //8 位元顯示碼 abcdefgp 。
        {
            if (bitRead(num[number%10],j)) //讀取目前七段顯示器掃描位數的位元 j 。
                digitalWrite(seg[j],HIGH); //若位元為 1 則 LED 狀態為 HIGH 。
            else
                digitalWrite(seg[j],LOW); //若位元為 0 則 LED 狀態為 LOW 。
        } //用PNP 所以LOW是亮
        digitalWrite(digit[i],LOW); //掃描第 i 行顯示器。
        delay(5); //用簡化電路的話兩者須交換 //掃描時間 5ms 。
        digitalWrite(digit[i],HIGH); //關閉掃描第 i 行顯示器，消除顯示鬼影。
        number=number/10; //掃描下一位數。
        if (millis()-time>=1000) //已經過 1 秒？
        {
            time=millis(); //記錄時間。
            count=count+1; //顯示值上數加 1 。
            if (count>9999) //已計數至 9999？
                count=0; //清除顯示值為 0000 。
        }
    }
}
```

# 實作練習



## 練習

1. 設計 Arduino 程式，控制四位七段顯示器下數計數並顯示 9999~0000 變化。
2. 設計 Arduino 程式，控制四位七段顯示器閃爍上數計數並顯示 0000~9999 變化。

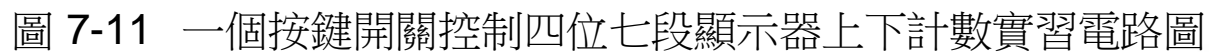
# 實作練習

## 按鍵開關控制四位七段顯示器上下計數實習

### ■ 功能說明：

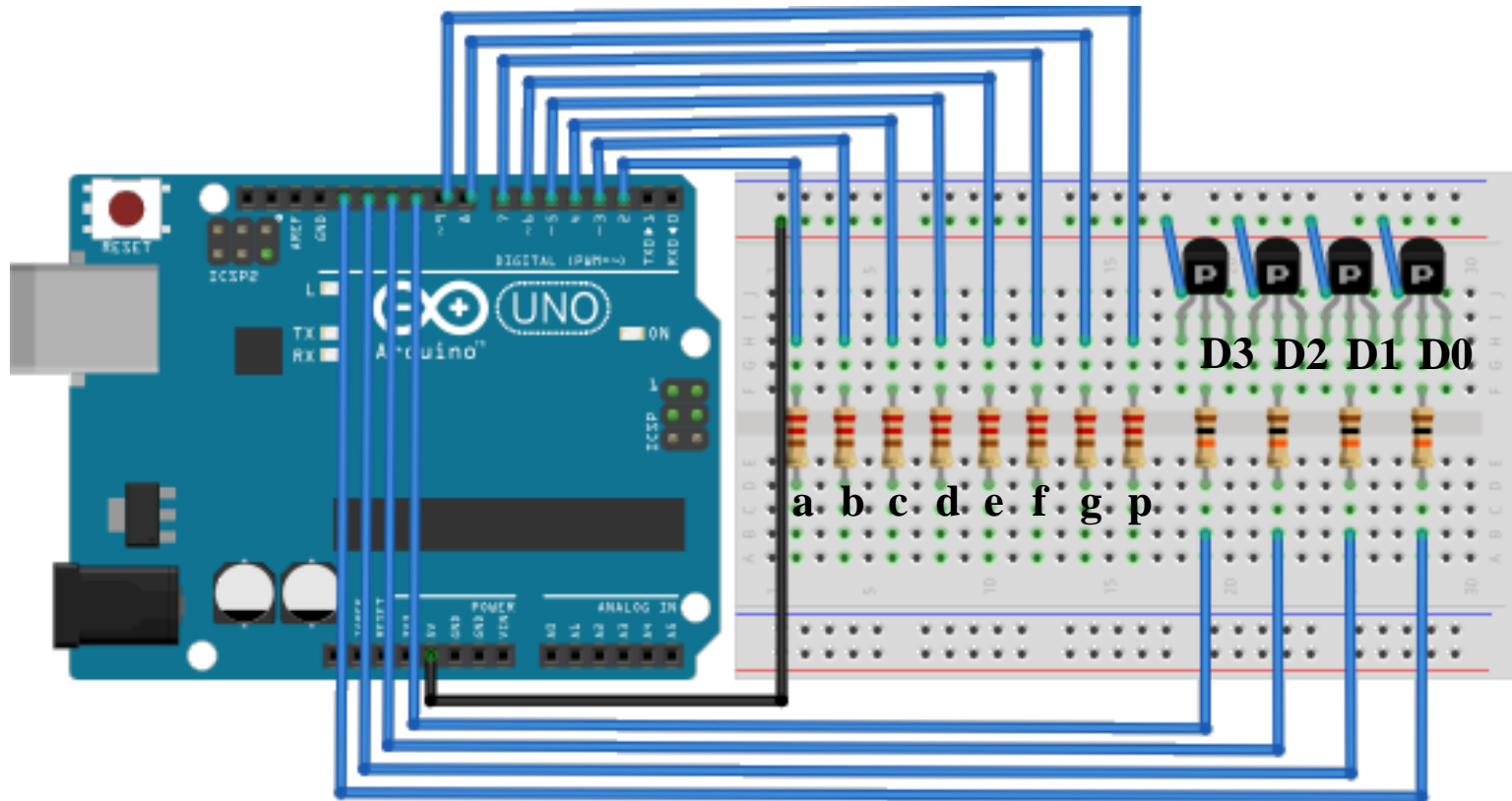
使用一個按鍵開關控制四位七段顯示器上、下數。每按一下按鍵開關，顯示器的狀態會改變，即原先為上數則改變為下數，原先為下數則改變為上數。

□ 電路圖及麵包板接線圖：



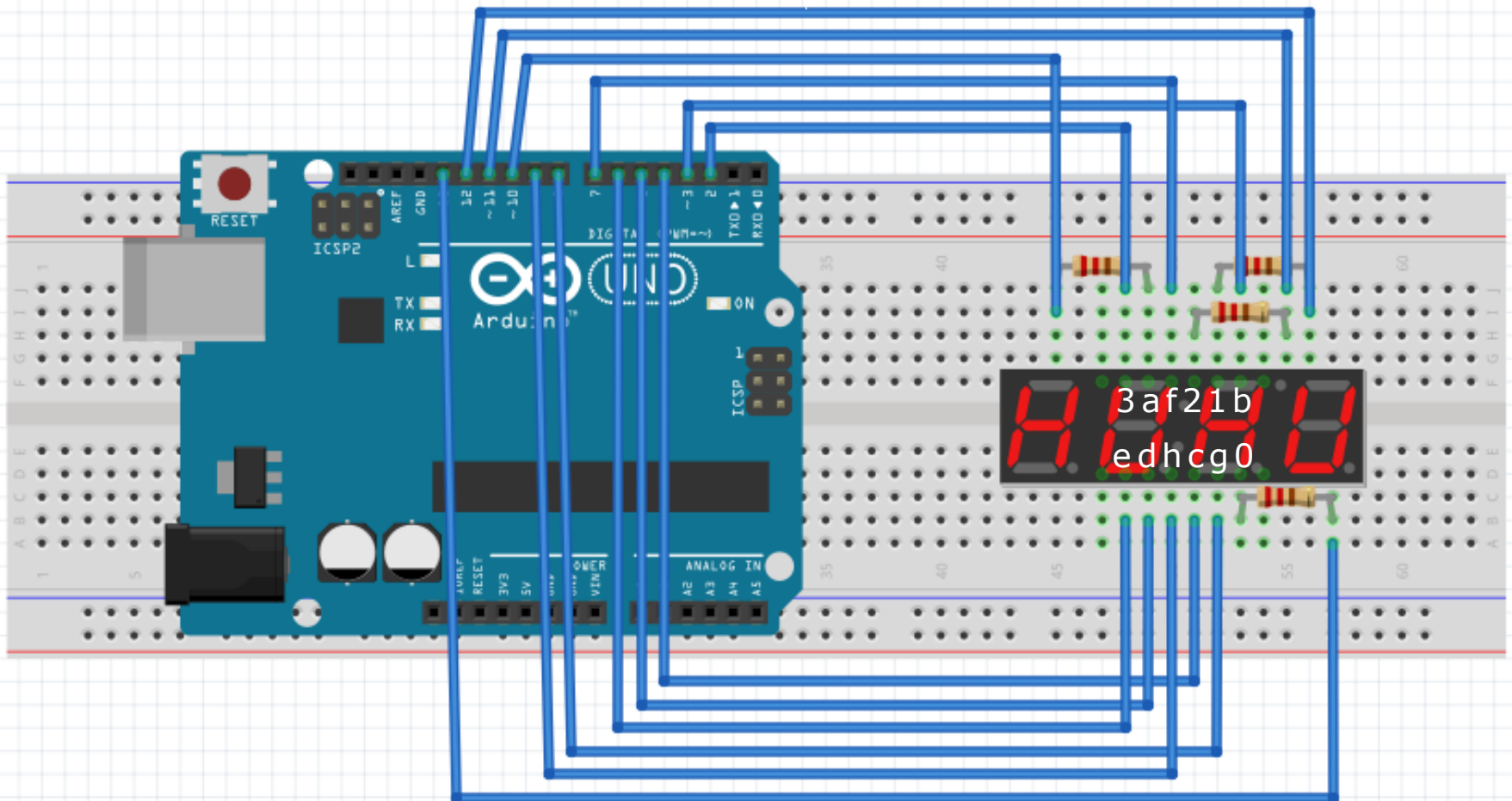
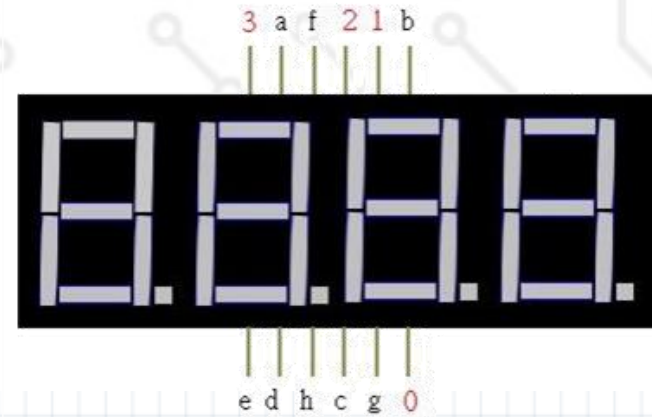


## 實作練習



## 四位七段顯示器 0000~9999 計數實習接線圖

# 實作練習-簡化電路



# 實作練習

□ 程式： B422.ino

int i;	//顯示器位數。
int j;	//顯示器段數。
int count=0;	//計數值。
int number;	//計數值。
int KeyData;	//按鍵值。
int numKeys=0;	//按鍵次數。
const int debounceDelay=20;	//開關穩定所需要的時間 20ms。
unsigned long time=0;	//計時。
const byte num[10]=	//0~9 顯示碼。
{ B11000000, B11111001, B10100100, B10110000, B10011001,	
B10010010, B10000010, B11111000, B10000000, B10010000};	
const int seg[]={2,3,4,5,6,7,8,9};	//顯示器各段 abcdefgp 數位接腳。
const int digit[]={10,11,12,13};	//顯示器共點 D0-D3 數位接腳。
const int sw=14;	//按鍵數位接腳。

# 實作練習

```
void setup()  
{  
    pinMode(sw, INPUT);           //設定數位接腳 14 為輸入模式。  
    digitalWrite(sw, HIGH);       //開啟內部上拉電阻。  
    for(i=0; i<8; i++)            //設定數位接腳 2~9 為輸出模式。  
        pinMode(seg[i], OUTPUT);  
    for(i=0; i<4; i++)            //設定數位接腳 10~13 為輸出模式。  
    {  
        pinMode(digit[i], OUTPUT);  
        digitalWrite(digit[i], HIGH); //關閉掃描信號。  
    }  
}
```

# 實作練習

```
void loop()  
{  
    KeyData=digitalRead(sw);           //讀取按鍵。  
    if (KeyData==LOW)                   //按下按鍵?  
    {  
        delay(debounceDelay);          //消除機械彈跳。  
        while (digitalRead(sw)==LOW)    //按鍵尚未放開?  
        ;                               //等待按鍵放開。  
        numKeys++;                      //記錄按鍵次數。  
    }  
    number=count;                       //顯示計數值。  
    for (i=3;i>=0;i--)                  //四位顯示器。  
    {  
        for (j=0;j<8;j++)              //每位顯示器有 abcdefgp 八段。  
        {  
            if (bitRead(num[number%10],j)) //段位元資料為 1?  
                digitalWrite(seg[j],HIGH); //設定段狀態為 HIGH。  
            else                               //段位元資料為 0。  
                digitalWrite(seg[j],LOW);  //設定段狀態為 LOW。  
        }  
    }  
}
```

# 實作練習

```
digitalWrite(digit[i], LOW); //致能第 Di 位數顯示器。
delay(5); //延遲 5ms。
digitalWrite(digit[i], HIGH); //除能第 Di 位數顯示器。
number=number/10; //下一位數。
if (millis()-time>=1000) //已經過 1 秒?
{
    time=millis(); //儲存時間。
    if (numKeys%2==0) //按鍵次數為偶數?
    {
        count++; //計數值加 1。
        if (count>9999) //計數值大於 9999?
            count=0; //重新設定計數值為 0000。
    }
    else //按鍵次數為奇數。
    {
        count--; //計數值減 1。
        if (count<0) //計數值小於 0000?
            count=9999; //重新設定計數值為 9999。
    }
}
}
```

# 回家作業

請使用一四位七段顯示器與一按鍵開關

設計功能如下：

按一下使其正數

按第二下使其倒數

按第三下停止並閃爍(亮0.5秒、暗0.5秒)





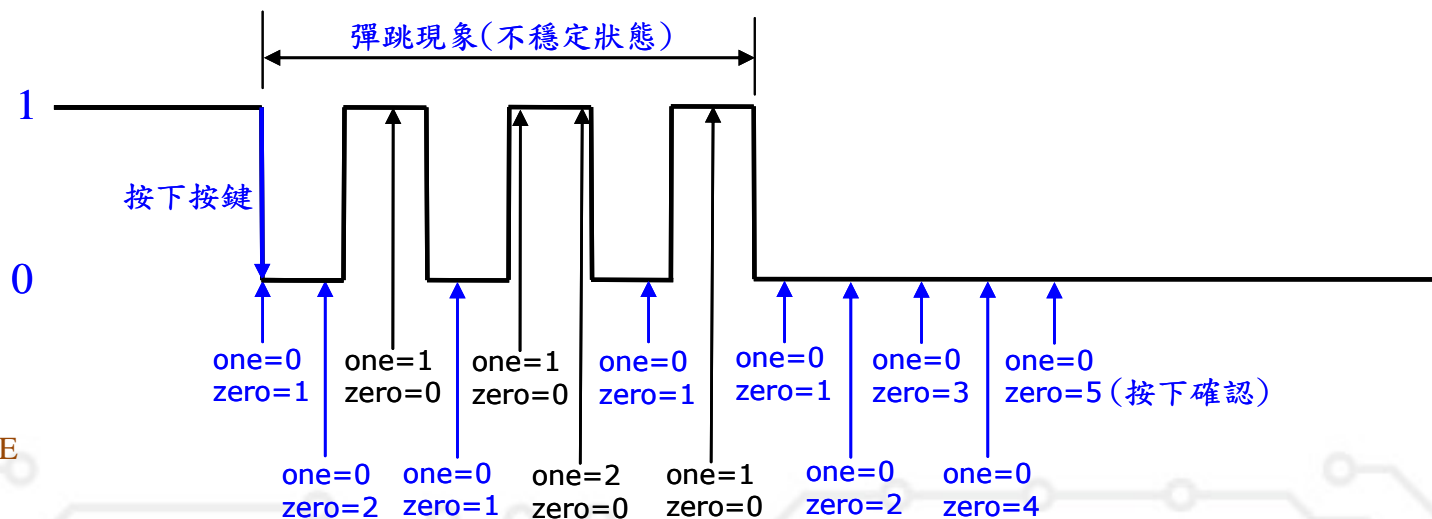
# 補充資料

## 讀取鍵盤並顯示於四位七段顯示器



功能說明：

讀取4×4矩陣鍵盤按鍵值並顯示於四位七段顯示器。在前節中使用延遲方法來消除機械彈跳，簡單但是效果不好。如圖7-13所示，使用連續檢測開關狀態的方式，當檢測開關被按下且與上次鍵值相同時，zero值加1，唯有在開關狀態穩定，才可能檢測5次以上的低電位，如此即可確定開關狀態已穩定。



# 實作練習

- 電路圖及麵包板接線圖：

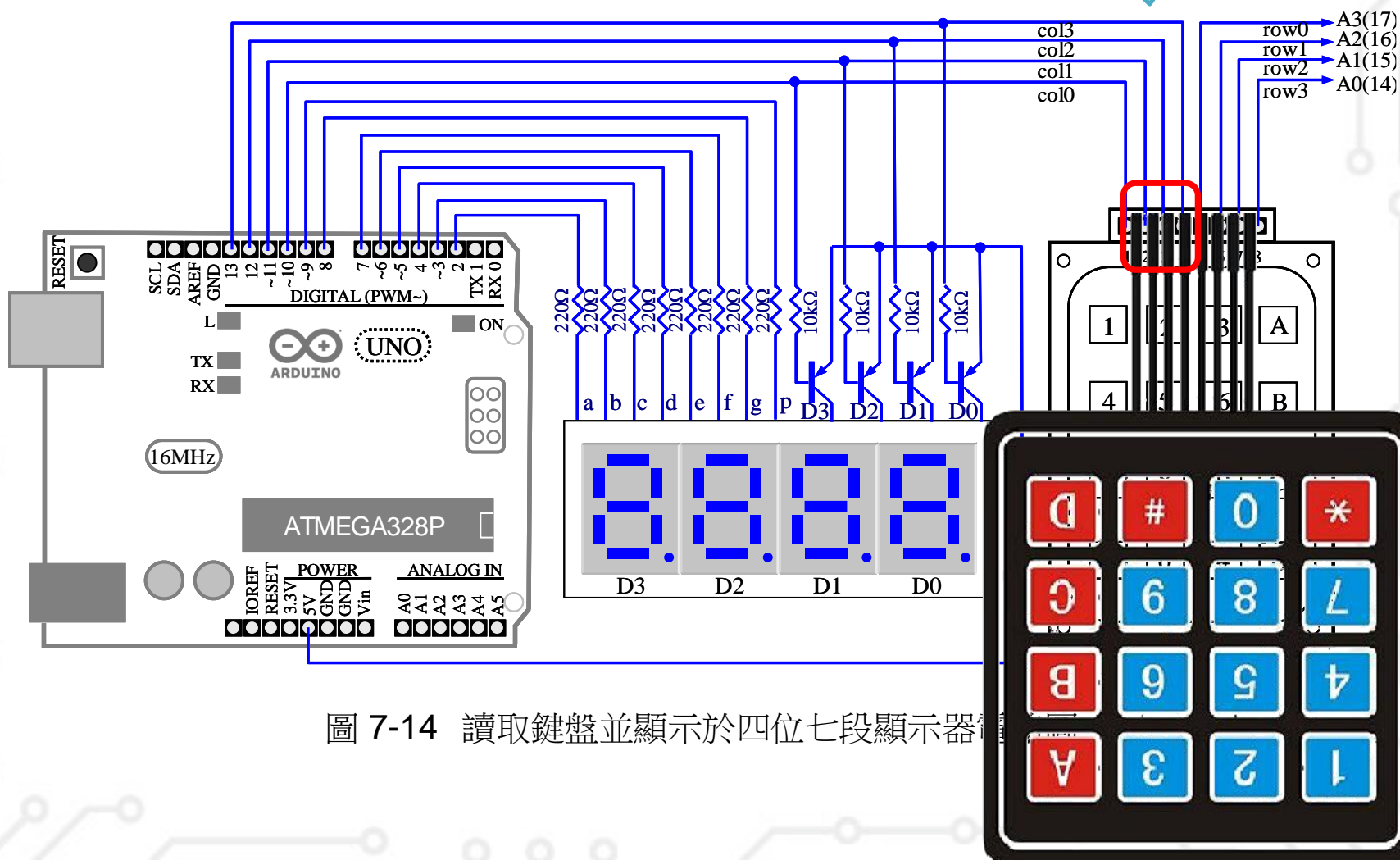


圖 7-14 讀取鍵盤並顯示於四位七段顯示器

# 實作練習

■ 程式： B431.ino

int i,j;	//索引值。
int key=-1;	//鍵值。
int KeyTemp=-1;	//未除彈跳鍵值。
int KeyData=-1;	//已除彈跳鍵值。
int zero=0;	//彈跳鍵值為0的次數。
int one=0;	//彈跳鍵值為1的次數。
const int numCols=4;	//總行數。
const int numRows=4;	//總列數。
const int numSegs=8;	//顯示器段數。
const int col[]={10,11,12,13};	//行的數位接腳。
const int row[]={14,15,16,17};	//電話鍵盤列的數位接腳。
const int seg[]={2,3,4,5,6,7,8,9};	//顯示器段 abcdefgp 的數位接腳。
int count[numCols]={0,0,0,0};	//顯示器初值。
const byte num[10]=	//0~9 顯示碼。
{ B11000000, B11111001, B10100100, B10110000, B10011001,	
B10010010, B10000010, B11111000, B10000000, B10010000};	
const int keyMap[numRows][numCols]=	//4 行*4 列電話鍵盤按鍵定義。
{ {1, 2, 3,10},	//按鍵1、2、3、A。
{4, 5, 6,11},	//按鍵4、5、6、B。
{7, 8, 9,12},	//按鍵7、8、9、C。
{14,0,15,13} };	//按鍵*、0、#、D。

# 實作練習

```
void setup()
{
    for(i=0;i<numSegs;i++)
    {
        pinMode(seg[i],OUTPUT);           //設定段數位接腳為輸出模式。
        digitalWrite(seg[i],HIGH);        //段初始狀態為 HIGH。
    }
    for(i=0;i<numCols;i++)
    {
        pinMode(col[i],OUTPUT);           //設定鍵盤行接腳為輸出模式。
        digitalWrite(col[i],HIGH);        //所有行接腳狀態為 HIGH。
    }
    for(i=0;i<numRows;i++)
    {
        pinMode(row[i],INPUT);            //設定鍵盤列接腳為輸入模式。
        digitalWrite(row[i],HIGH);        //開啟所有列接腳的內部上拉電阻。
    }
}
```

# 實作練習

```
void loop()
{
    for(i=0;i<numCols;i++)
    {
        for(j=0;j<numSegs;j++)          //輸出字型碼。
        {
            if(bitRead(num[count[i]],j)) //若段位元資料為1，則設定HIGH。
                digitalWrite(seg[j],HIGH);
            else                                //若段位元資料為0，則設定LOW。
                digitalWrite(seg[j],LOW);
        }
        digitalWrite(col[i],LOW);
        for(j=0;j<numRows;j++)            //檢視該行所有按鍵狀態。
        {
            if(digitalRead(row[j])==LOW) //有按鍵被按下?
            {
                key=keyMap[j][i];          //轉換按鍵值。
                if(KeyTemp!=key)            //與上次鍵值不同?
                {
                    KeyTemp=key;            //儲存鍵值。
                    one=0;
                    zero=1;                  //開始除彈跳。
                }
            }
        }
    }
}
```

# 實作練習

```
else //與上次鍵值相同。
{
    if (zero<5) //鍵值尚未消除彈跳?
    {
        zero=zero+1; //開始除彈跳。
        if (zero==5) //鍵值已消除彈跳完成?
            KeyData=KeyTemp; //儲存除彈跳鍵值。
    }
}

if (KeyData>=0 && KeyData<=9) //鍵值為 0~9?
{
    count[0]=count[1]; //更新顯示值。
    count[1]=count[2];
    count[2]=count[3];
    count[3]=KeyData;
    KeyData=-1; //清除鍵值。
}

} //用簡化電路的話此區需更改
delay(5); //掃描更新。
digitalWrite(col[i],HIGH); //消除顯示鬼影。
}
}
```

# 實作練習



## 練習

1. 設計 Arduino 程式，讀取 4×4 矩陣鍵盤按鍵值 0~9 並顯示鍵值於四位顯示器。當按下\*鍵時，將四位顯示值由串列埠傳送至 PC 端。
2. 設計 Arduino 程式，讀取 4×4 矩陣鍵盤按鍵值 0~9 並顯示於四位七段顯示器。按下\*鍵時，可將四位顯示值由串列埠傳送至 PC 端。PC 鍵盤輸入值亦可顯示於四位七段顯示器上。

## 回家作業

請試用此鍵盤，實作一簡單計算機

(並試著將程式改成可連續輸入相同數字)

## 回家練習：

(一)用此鍵盤改變單顆LED燈的亮度

(二)做成可改變亮度的雨滴燈！



# 計算機作業提示

```
if(KeyData>=0 && KeyData<=9)
{
    count[0]=count[1];
    count[1]=count[2];
    count[2]=count[3];
    count[3]=KeyData;
    if(!(KeyData>=10 && KeyData<=14))
        KeyData=-1;

    //KeyTemp=100;//使相同按鍵可以被讀入
}
if(KeyData>=10 && KeyData<=14){
    switch(KeyData){
        case 10: //A +
            a = count[0]*1000+count[1]*100+count[2]*10+count[3]*1;
            Serial.print(a);
            Serial.println();
            op = '+';
            count[0]=0;
            count[1]=0;
            count[2]=0;
            count[3]=0;
            break;
        case 11: //B -
```

```
            case 11: //B -
                a = count[0]*1000+count[1]*100+count[2]*10+count[3]*1;
                Serial.print(a);
                Serial.println();
                op = '-';
                count[0]=0;
                count[1]=0;
                count[2]=0;
                count[3]=0;
                break;
            case 12: //C *
                a = count[0]*1000+count[1]*100+count[2]*10+count[3]*1;
                Serial.print(a);
                Serial.println();
                op = '*';
                count[0]=0;
                count[1]=0;
                count[2]=0;
                count[3]=0;
                break;
            case 13: //D /
```



# 計算機作業提示

```
case 15: //# =
```

```
    b = count[0]*1000+count[1]*100+count[2]*10+count[3]*1;
```

```
    if(op=='+')
```

```
        ans = a+b;
```

```
    if(op=='-')
```

```
        ans = a-b;
```

```
    if(op=='*')
```

```
        ans = a*b;
```

```
    if(op=='/')
```

```
        ans = a/b;
```

```
    Serial.print(a);
```

```
    Serial.println();
```

```
    Serial.print(b);
```

```
    Serial.println();
```

```
    Serial.print(ans);
```

```
    Serial.println();
```

```
    tmp = ans;
```

```
    count[0] = tmp/1000;
```

```
    tmp = tmp-1000*count[0];
```

```
    count[1] = tmp/100;
```

```
    tmp = tmp-100*count[1];
```

```
    count[2] = tmp/10;
```

```
    tmp = tmp-10*count[2];
```

```
    count[3] = tmp;
```

```
break:
```

```
case 13: //D /
```

```
    a = count[0]*1000+count[1]*100+count[2]*10+count[3]*1;
```

```
    Serial.print(a);
```

```
    Serial.println();
```

```
    op = '/';
```

```
    count[0]=0;
```

```
    count[1]=0;
```

```
    count[2]=0;
```

```
    count[3]=0;
```

```
    break;
```

```
case 14: //*
```

```
    //break;
```

```
    }
```

```
    KeyData=-1;
```

```
    }
```

```
    }
```

```
    delay(5);
```

```
    digitalWrite(col[i],HIGH);
```

```
    }
```

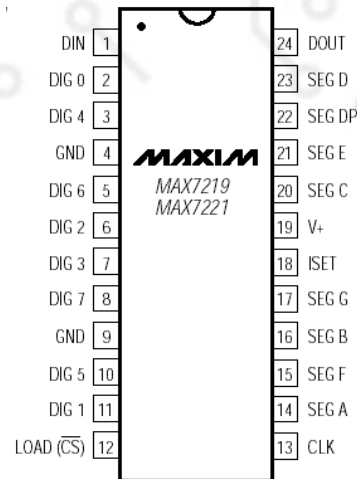
# 課外補充

## 使用 **MAX7219** 驅動四位七段顯示器實習

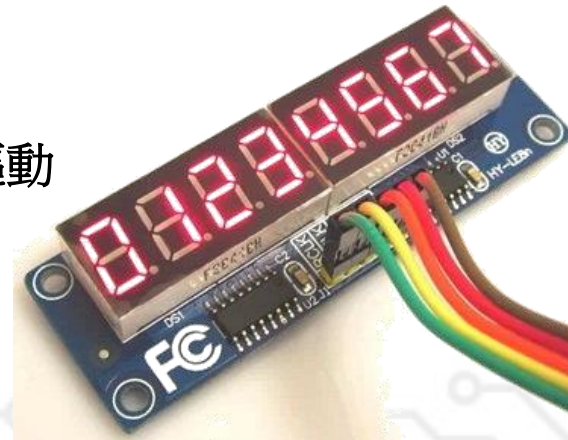
### □ 功能說明：

本節使用 MAX7219 晶片控制四位數共陰極七段顯示器計數並顯示 0000~9999。因為 MMA7219 晶片已經內建多工及解碼電路，因此比前幾節較容易控制多位數七段顯示器。

有關 MAX7219 IC 的相關說明詳見第 9 章，MAX7219 為一 10MHz 的 SPI 串列介面驅動 IC，可以驅動一個共陰極 8×8 矩陣型 LED 顯示器，或是八個共陰極七段顯示器，具有獨立 LED 段驅動、150μA 低功率關閉模式、顯示亮度控制、數字 BCD 解碼選擇等功能。



### 74HC595 驅動



# 課外補充

□ 電路圖及麵包板接線圖：

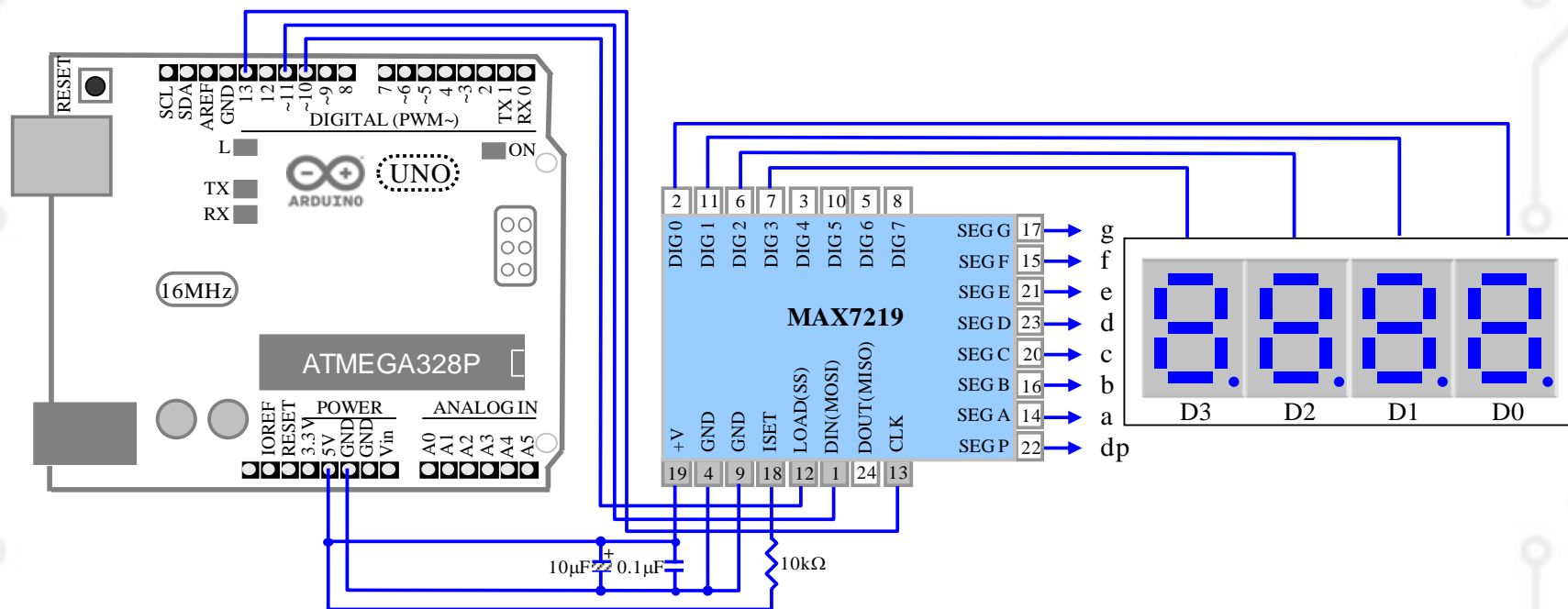


圖 7-15 使用 MAX7219 驅動四位七段顯示器電路圖

# 課外補充

## □ 程式：

<code>#include &lt;SPI.h&gt;</code>	<code>//使用 SPI 函式庫。</code>
<code>const int slaveSelect=10;</code>	<code>//MAX7219 致能腳。</code>
<code>const int decodeMode=9;</code>	<code>//MAX7219 解碼模式暫存器。</code>
<code>const int intensity=10;</code>	<code>//MAX7219 亮度控制暫存器。</code>
<code>const int scanLimit=11;</code>	<code>//MAX7219 掃描控制暫存器。</code>
<code>const int shutDown=12;</code>	<code>//MAX7219 關閉模式暫存器。</code>
<code>const int dispTest=15;</code>	<code>//MAX7219 顯示測試暫存器。</code>
<code>int number=0;</code>	<code>//計數值。</code>
<code>void setup()</code>	
<code>{</code>	
<code>    SPI.begin();</code>	<code>//初始化 SPI 介面。</code>
<code>    pinMode(slaveSelect,OUTPUT);</code>	<code>//設定數位接腳 10 為輸出模式。</code>
<code>    digitalWrite(slaveSelect,LOW);</code>	<code>//除能 MAX7219。</code>
<code>    sendCommand(shutDown,1);</code>	<code>//MAX7219 正常工作。</code>
<code>    sendCommand(dispTest,0);</code>	<code>//關閉顯示器測試。</code>
<code>    sendCommand(intensity,1);</code>	<code>//中等顯示亮度。</code>
<code>    sendCommand(scanLimit,7);</code>	<code>//掃描 8 位數。</code>
<code>    sendCommand(decodeMode,255);</code>	<code>//使用 MAX7219 內部解碼器。</code>
<code>}</code>	

# 課外補充

```
void loop()
```

```
{
```

```
    for (number=0;number<10000;number++)          //計數 0000~9999。
```

```
    {
```

```
        displayNumber(number);                    //顯示計數值。
```

```
        delay(1000);                               //延遲 1 秒。
```

```
    }
```

```
}
```

```
void displayNumber(int number)                    //顯示函式。
```

```
{
```

```
    sendCommand(1,number/100/10);                //顯示千位數。
```

```
    sendCommand(2,number/100%10);                //顯示百位數。
```

```
    sendCommand(3,number%100/10);                //顯示十位數。
```

```
    sendCommand(4,number%100%10);                //顯示個位數。
```

```
}
```

```
void sendCommand(byte command,byte value) // MAX7219 設定命令函式。
```

```
{
```

```
    digitalWrite(slaveSelect,LOW);                //致能 MAX7219。
```

```
    SPI.transfer(command);                         //寫入命令字元。
```

```
    SPI.transfer(value);                          //寫入資料字元。
```

```
    digitalWrite(slaveSelect,HIGH);               //除能 MAX7219。
```

```
}
```

# 課外補充資料

## 基本焊接與佈線

[http://blog.sina.com.cn/s/blog\\_6692b6140101ib47.html](http://blog.sina.com.cn/s/blog_6692b6140101ib47.html)

## PBC入門

<http://arduino.tw/index.php/course/getting-started-eagle/%E5%AE%A2%E8%A3%BD%E5%8C%96arduino%E9%9B%BB%E8%B7%AF%E7%9A%84%E7%AC%AC%E4%B8%80%E6%AD%A5.html>

## 洗電路版

<http://gcycrobot.blogspot.tw/2011/05/diy-arduino-1.html>

