Chapter 5

決策流程指令

人類的生活必須不斷面對決策問題，連我家一個不到三歲的小孩，也常要思考他手裡的十元是要坐電動車還是買棒棒糖。程式語言是協助人類解決問題的工具，當然也有決策流程敘述，Arduino 語言依決策流程點的多寡，分爲以下兩種決策流程敘述，第一是雙向分歧決策流程 if~else~，例如天色暗了就點燈，否則不理會；第二是多向分歧決策流程的 switch...case，例如你身上有 5000 元，走進一家五星級的大飯店用餐，你的分歧點就很多，有自助餐、中式套餐、日本料理、泰國餐點等等分歧點。本章的重點即是探討 Arduino 語言的決策流程敘述。其次，現代正夯的機器人、自駕車更要時時刻刻不斷判斷，這都要使用決策敘述。

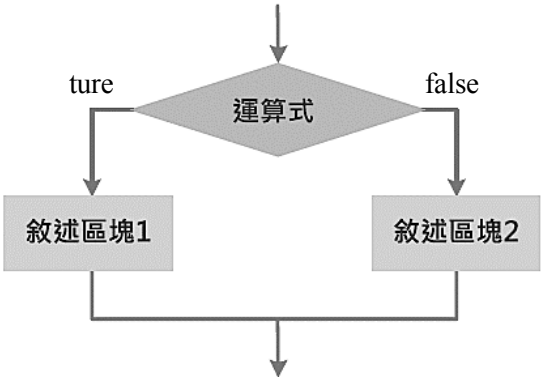
5-1 if...else

在日常生活領域中，常出現“假如～則～，否則～”時，此種決策流程模式有兩種解決問題的方案，故稱爲雙向分歧決策流程，此時可使用 if...else 敘述。if...else 敘述的語法如下：

```
if（運算式）
{
   敘述區塊 1；
}
[else
{
   敘述區塊 2；
}]
```

以上語法說明如下：

1. 運算式的值若爲 1 (true)，則執行敘述區塊 1；運算式的值若爲 0 (false)，則執行敘述區塊 2，其流程圖如下：



2. 敘述區塊的範圍習慣使用大括號({})包圍。例如，以下敘述可依 a 的大小評量其及格與否。

```
if (a>=60){
   b=1;
}
else {
   b=0;
}
```

3. 敘述區塊內的敘述若只有一個，則敘述區塊上下兩個大括號可予省略。例如，以上程式可簡化如下：

```
if (a>=60)
   b=1;
```

```
else
   b=0;
```

4. 有時爲了簡化程式的撰寫，可將否則的部分寫在 if 前面，並省略 else。例如，以下程式同義於上面程式（語法的中括號，代表此部分可省略）。

```
   b=0;
   if(a>=60)
      b=1;
```

5. 若省略敘述區塊上下的大括號，則條件成立時，僅執行敘述區塊的第一個敘述。例如，以下敘述執行之後，b=0，c=1。

```
   a=55;b=0;c=0;
   if(a>=60)
      b=1;
      c=1;   //此敘述並不會因為縮排而屬於 if 敘述的一部份
   Serial.println(b);    /*  b=0, c=1 */
   Serial.println(c);
```

6. if()後面不要緊接分號，例如，

```
   a=55;b=0;c=0;
   if(a>=0);
     b=1;
     c=1;
   Serial.println(b);   /*  b=1, c=1 */
   Serial.println(c);
```

這是初學者常犯的錯誤，因爲 if 遇到分號（；），表示此指令已經結束。

7. 敘述區塊內可以放置任何合法敘述，當然也可以再放置 if；if 中有 if，稱爲巢狀 if，請看以下範例。

▶範例 5-1a

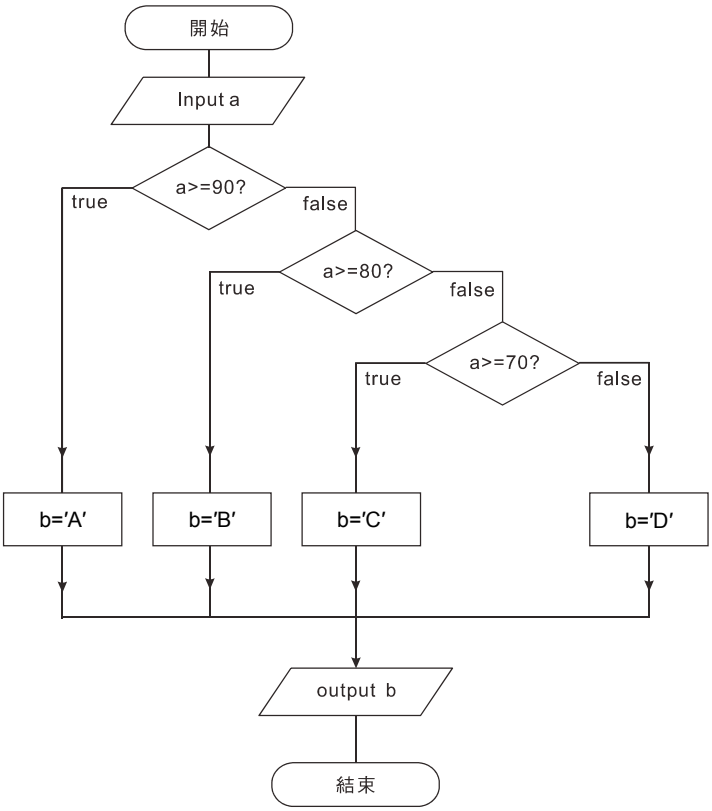
試寫一程式，完成以下要求：

1. 輸入一個 0～100 的分數。
2. 當分數大於等於 90 分時，輸出 A。
3. 當分數介於 80～89 時，輸出 B。

4. 當分數介於 70～79 時，輸出 C。
5. 當分數介於 0～69 時，輸出 D。

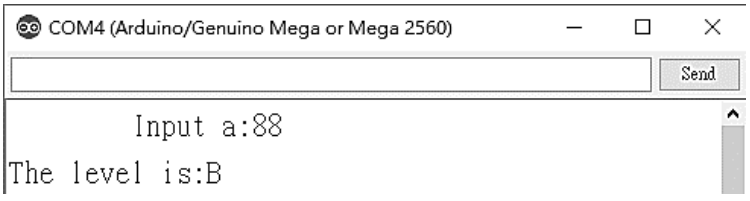
➲ 流程分析

1. 使用流程圖分析如下：



2. 以上每一個決策流程點，都有兩個分歧點，所以適用 if~else。
3. 每一個 else 後面均需進一步決策流程，所以可在每一 else 後面再放置 if。

➲ 執行結果



➲ 程式列印

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   int a;
   char b;
   Serial.print("Input a:");
   while(Serial.available() ==0) {}
   a=Serial.parseInt();
   Serial.println(a);
   if(a>=90)           /* 高於 90 分為 A */
      b='A';
   else
      if(a>=80)        /* 介於 80 與 90 分為 B */
         b='B';
      else
         if(a>=70)     /* 介於 70 與 80 分為 C */
            b='C';
         else
            b='D';     /* 不符合上述情況則為 D */
   Serial.print("The level is:");
   Serial.println(b);
}
```

➲ 補充說明

1. 此題有人會寫成 if (90<a<100)，但 Arduino 並沒有這種運算式，因爲 90<a<100 是數學語言，不是程式語言。
2. 有人會寫成

```
if (a<100 & a>=90) b='A';
if (a<90  & a>=80)  b='B';
if (a<80  & b>=70)  b='C';
if (a<70  & b>=0)   b='D';
```

這樣雖然也可以，但是其執行效率非常差，因爲不管分數爲何，都要執行

四次判斷，所以這樣效率很低。

3. 這題有人會寫成如下：

```
if(a>=90)           /* 高於 90 分為 A */
   b='A';
else if(a>=80)      /* 介於 80 與 90 分為 B */
   b='B';
else if(a>=70)      /* 介於 70 與 80 分為 C */
   b='C';
else
   b='D';      /* 不符合上述情況則為 D */
```

這樣當然可以，而且這樣不用一再縮排，因爲程式一再縮排，程式會右移，那程式很容易被分割爲兩列，因爲分割兩列後的程式，其可讀性會降低。

➲ 自我練習

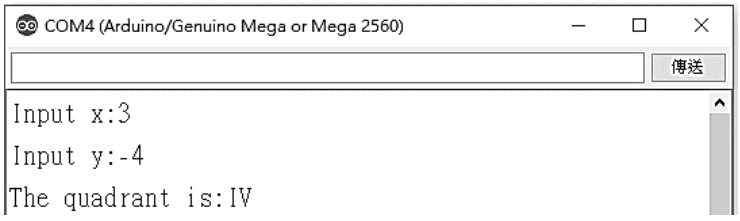
1. 請寫一個程式，完成以下要求：
   (1) 產生一個 0～25 的亂數。
   (2) 當亂數大於 20 時，輸出五個燈。
   (3) 當亂數介於 16～20 時，輸出四個燈。
   (4) 當亂數介於 11～15 時，輸出三個燈。
   (5) 當亂數介於 0～10 時，輸出兩個燈。
2. 請寫一程式，可以根據可變電阻的輸出值調整 LED 亮的個數。左轉到底則 8 個燈全熄滅。逐漸向右轉，逐漸亮 1 個，2 個，3 個 LED，右轉到底則 8 個燈全亮。提示：可變電阻的輸出範圍是 0～1023。
3. 同上題，但改爲左轉到底亮最左邊一個，逐漸右轉，LED 逐漸向右移動，每次僅亮一個，右轉到底，則亮最右邊一個。
4. 請設計一個電路，使用光敏電阻，控制 8 個 LED 的點亮個數，光線越弱，亮的個數越多。
5. 請將光敏電阻放在門口，當有人經過時，可讓 LED 亮著。
6. 參觀人數統計表。請將光敏電阻放在門口，當有人經過時，可讓計數器加 1，本例暫由序列埠視窗輸出或 8 個 LED 顯示計數器的值。

7. 同上一題，但有人走出去也會讓計數器誤判，那要如何克服呢？請用兩個光敏電阻試看看，只有單方向人數要加 1。

▶範例 5-1b

請寫一個程式，可以判斷所輸入座標的所在象限。

➲ **執行結果**（輸入模式請點選『沒有行結尾』）



➲ **程式列印**

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    int x,y;
    String b;
    Serial.print("Input x:");
    while(Serial.available() ==0) {}
    x=Serial.parseInt();
    Serial.println(x);
    Serial.print("Input y:");
    while(Serial.available() ==0) {}
    y=Serial.parseInt();
    Serial.println(y);
    if(x>0)
         if(y>0)
           b="I";       /* 第一象限 */
         else
           b="IV";      /* 第四象限 */
    else
         if(y>0)
           b="II";      /* 第二象限 */
         else
```

```
        b="III";     /* 第三象限 */
    Serial.print("The quadrant is:");
    Serial.println(b);
}
```

➲ 補充說明

1. 此題目有人會寫成

```
    if (x>0 & y>0) b=”I”;
    if (x<0 & y>0) b=”II”;
    if (x<0 & y<0) b=”III”;
    if (x>0 & y<0 ) b=”IV”;
```

這樣雖然沒有錯，但是執行效率非常差，因爲電腦要不斷的比較。

2. 請鍵入以下程式，並輸入(-3,-4)，觀察執行結果。

```
    b="";
    if(x>0)
        if(y>0)
          b="I";
    else
        b="II or III";
    Serial.print("The quadrant is:");
    Serial.println(b);
```

您將會發現答案不對，那是因爲 else 是依附最近的 if，不是依照您的縮排層次，所以本題您若要強調 b="II or III or IV";是依附第一個 if，那就要用大括號改變 else 的依附對象，程式如下：

```
    b="";
    if(x>0){
        if(y>0)
          b="I";
    }
    else
        b="II or III";
    Serial.print("The quadrant is:");
    Serial.println(b);
```

➲ 自我練習

1. 同上範例，但增加判斷是否在 x、y 軸上或原點。（提示：範圍小的要先檢查）
2. 假設您有一個熱敏電阻、有一個光敏電阻，當又熱又亮時請顯示一個燈，不熱而亮顯示兩個燈，不熱又不亮顯示三個燈，熱而不亮顯示四個燈。
3. 假設自來水費率如下：

   100 度以下，每度 3 元。

   100～300 度，超過 100 度的部分，每度 5 元。

   300 度以上，超過 300 度的部分，每度 6 元。

   根據以上條件，寫一程式，可輸入用水度數，得到水費。

▸範例 5-1c

請寫一程式，滿足以下條件。

1. 輸入兩個數。
2. 求輸入兩數極大值。
3. 輸出極大值。

➲ 演算法則

1. 輸入第一數，本例以變數 a 儲存。
2. 輸入第二數，本例以變數 b 儲存。
3. 設定較大值(max)爲第一數。max=a
4. 當第二數(b)大於極大值時，極大值即以 b 取代。

```
if (b>max)
   max=b;
```

5. 輸出極大值(max)即爲所求。

➲ 程式列印（本例變數輸入先用指派代替，程式如下）

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

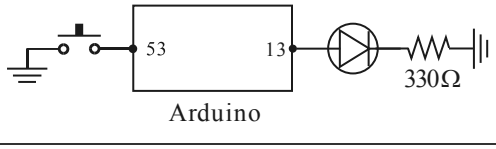
```
}
void loop() {
    int a=3,b=4,max;
    max=a;
    if (b> max);
       max=b;
    Serial.print("The max is:");
    Serial.println(max);
}
```

➲ 自我練習

1. 請寫一程式，滿足以下條件。
   (1) 輸入三個數。
   (2) 求此三個數的極小值。
   (3) 輸出極小值。
2. 若有三個熱敏電阻偵測三台機器的溫度，編號分別是 1,2,3，請找出最熱機器的溫度與機器編號。
3. 若有 3 個光敏電阻，偵測 3 個地方的光線強弱，請使用 3 個 LED 監控，請寫一程式，將光線最暗的地方，點亮 LED。
4. 請產生 5 個 1 到 99 的亂數，請將最大與最小的數字去掉，將中間 3 個亂數取平均，並在序列埠視窗輸出。

▶範例 5-1d

請使用 1 個按壓開關控制 1 個 LED，當按壓開關按下去，則 LED 亮起來，放開按壓開關時，LED 熄滅。



➲ 程式列印

```
const byte pb=53;
const byte led=13;
byte b;
void setup() {
    pinMode(pb,INPUT_PULLUP);
```

```
    pinMode(led,OUTPUT);
}
void loop() {
    b=digitalRead(pb);
    if (b==LOW)
        digitalWrite(led,HIGH);
    else
        digitalWrite(led,LOW);
}
```

➲ 自我練習

1. 請用 4 位元指撥開關控制 4 個 LED 的明滅，指撥開關 on 時，LED 亮；指撥開關 off 時，LED 滅。

▶範例 5-1e

計數器。請寫一個程式，每當按壓開關被按一下，計數值加 1。

➲ 思考步驟

1. 程式設計初步如下：

```
const byte pb=53;
byte a=0;
byte b;
void setup() {
    pinMode(pb,INPUT_PULLUP);
    Serial.begin(9600);
}
void loop() {
    b=digitalRead(pb);
    if (b==LOW)
        a++;
    Serial.println(a);
}
```

2. 每按一下按鍵，計數值卻遞增很多，那是因爲單晶速度太快了，所以我們要用一個迴圈改善，程式如下：

```
const byte pb=2;
byte a=0;
byte b;
void setup() {
    pinMode(pb,INPUT_PULLUP);
    Serial.begin(9600);
}
void loop() {
    b=digitalRead(pb);
    if (b==LOW){
        while(digitalRead(pb)==LOW)
            delay(10);
        a++;
    }
    Serial.println(a);
}
```

3. 也就是當按壓開關未被放開時，使用 while 迴圈繼續等待，直到按鍵被釋放，程式再繼續執行，這樣計數值才不會一直被灌水。關於 while 迴圈，請研讀第六章就會明瞭。

➲ 自我練習

1. 請安排四個按壓開關，其功能分別是遞增一、遞減一、遞增十、遞減十。（本題使用序列埠視窗輸出）
2. 同上題，但輸出請用範例 2-3b 的八個 LED 顯示 a 的值。
3. 同上題，但僅有兩個按鈕就好，且僅能輸入 0 到 8，且用 LED 亮的個數表示計數值。
4. 同上題，但計數值 1 就亮第 1 個燈，計數值 2 亮第 2 個燈，依此類推。

▶範例 5-1f

請使用一個按壓開關控制 LED 的明亮，按一下時亮，且自保持，再按一下時滅，且自保持。

➲ 思考步驟

1. 本例爲了能記憶目前的狀態且自保持，那我們需要一個變數 state，記錄且保留目前的狀態。

```
    bool state=false;
```

2. 當按壓開關被按時，狀態改變。程式如下：

```
    state=!state;
```

➲ 程式列印

```
const byte pb=53;
const byte led=13;
bool state=false;
byte b;
void setup() {
    pinMode(pb,INPUT_PULLUP);
    pinMode(led,OUTPUT);
    digitalWrite(led,state);
}
void loop() {
    b=digitalRead(pb);
    if (b==LOW)      {
        while(digitalRead(pb)==LOW)
            delay(10);
        state=!state;
    }
    digitalWrite(led,state);
}
```

➲ 補充說明

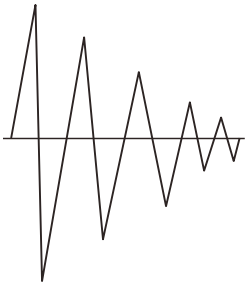
1. 請鍵入以下程式，並觀察執行結果。

```
void loop() {
    b=digitalRead(pb);
    if (b==LOW)      {
```

```
        state=!state;
    }
    digitalWrite(led,state);
}
```

➲ 補充說明

1. 時間的延遲。因爲單晶速度很快，當您按下按鈕的短短時間內，開關接觸瞬間，開關接點會有彈跳現象（on 與 off 之間連續快速反覆變化），如右圖所示：



在此彈跳期間單晶也會重複偵測到此開關上千遍的 on 與 off，所以您的 LED 也會閃爍，有些負載會吃不消，很容易壞掉。所以本例使用 while 迴圈延遲一段時間，跳過這些彈跳電壓，關於 while 迴圈請看第六章。

2. 本例的

```
while(digitalRead(pb)==LOW)
      delay(10);
```

和以下程式效果相同，只是 while 迴圈僅執行一個敘述，所以大括號省略。

```
while(digitalRead(pb)==LOW){
    delay(10);
}
```

都可跳過電路上下彈跳的問題。

3. 本例使用軟體克服電路彈跳問題。但是，非單晶的傳統數位電路，則要用電容與電阻作一個微分電路，以便排除此一彈跳電壓。

➲ 自我練習

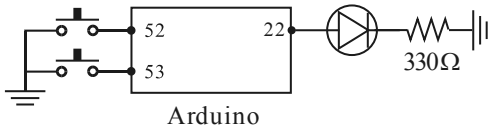
1. 請用兩個按壓開關控制一個 LED 的明滅，其中一個按下去亮，且自保持；另一個按下去滅掉，且自保持。

2. 請用 3 個按壓開關控制 3 個 LED，通通是按一下亮，且自保持，再按一下滅，且自保持。
3. 請用兩個按壓開關控制 8 個 LED 的左旋或右旋。程式執行時，最右邊的一個 LED 亮，按一下第一個按鈕，LED 左移一位元，按第二個按鈕，LED 右移一位元。當 LED 移至最左邊時，若再按第一個按鈕，LED 將會移到最右邊，此稱左旋；同理，當移至最右邊時，若再按第二個按鈕，亦會移到最左邊，此稱右旋。

▶範例 5-1g

計算通話秒數。請寫一程式，使用兩個按壓開關控制一個 LED，當第一個按壓開關被按時，LED 亮起，且開始計時，當第二個開關被按時，LED 熄滅，並使用序列埠監控視窗輸出通話秒數。

➲ 電路設計



➲ 思考步驟

1. Arduino 提供一個時間 millis()函式，它可以傳回程式執行後所經過的秒數，單位是千分之一秒(ms)。
2. 第一個按鍵被按，紀錄此時間 t1。
3. 第二個按鍵被按，紀錄此時間 t2。
4. 兩個時間相減，即爲通話時間。

➲ 程式列印

```
const byte pb1=52;
const byte pb2=53;
const byte led1=22;
long t1,t2,t3;
byte b;
```

```
void setup() {
    Serial.begin(9600);
    pinMode(pb1,INPUT_PULLUP);
    pinMode(pb2,INPUT_PULLUP);
    pinMode(led1,OUTPUT);
    digitalWrite(led1,LOW);
}
void loop() {
    b=digitalRead(pb1);
    if (b==LOW)      {
        while(digitalRead(pb1)==LOW)
            delay(10);
        t1 = millis();
        digitalWrite(led1,HIGH);
    }
    b=digitalRead(pb2);
    if (b==LOW)      {
        while(digitalRead(pb2)==LOW)
            delay(10);
        t2 = millis();  //單位是ms
        t3=(t2-t1)/1000;//s
        digitalWrite(led1,LOW);
        Serial.println(t3);
    }
}
```

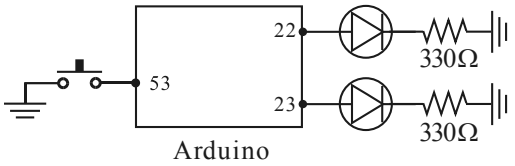
➲ 自我練習

1. 同上範例，假設每 5 秒 1 元，不足 5 秒以 5 秒計，請統計其通話費。
2. 手機通話秒數。同上範例，那如果僅用一個按鈕，那程式如何寫呢。
3. 平交道控制。假設有兩個按壓開關，火車來時壓到一個開關，LED 亮代表柵欄放下，離開時，也壓到另一開關，LED 熄滅代表柵欄升起。
4. 假設您有 3 個停車位，3 個指撥開關，3 個 LED 顯示其停車與否，請寫程式計算其停車時間。

▶範例 5-1h

請寫一程式，使用一個按壓開關，控制兩個 LED，讓使用者在一個短暫的時間內，按一下時亮一個燈；按兩下時亮兩個燈；按三下時全熄滅。

➲ 電路設計



➲ 思考步驟

1. 範例 5-1f，我們使用一個 state 變數記錄且保留當時的狀態，這一題則需要一個計數器，以便統計使用者共按幾下。然後再根據按鍵次數，決定 LED 的明滅。
2. 其次，上一範例的狀態是一直保留，但本範例的按鈕的按鍵次數則希望僅能保留一點時間，就要歸零，這就要根據使用者習慣去衡量，本例就用 0.5 秒，也就是使用者有按鈕，那就繼續等待 0.5s，看他有沒有繼續按，程式如下：

```
t1=t2=millis(); //傳回程式執行後所經過的時間
do{
    b=digitalRead(pb);
    if (b==LOW)       {
        t1 = millis();
        while(digitalRead(pb)==LOW)
            delay(10);
        num=num+1;
    }
    t2=millis();
    t=t2-t1;
}while( (t<500) );//根據習慣調整
```

3. 若沒有繼續按，就表示使用者按完了，這就可以根據使用者按鍵次數，去執行相對應功能。

```
    if (num==1){
        digitalWrite(led1,HIGH);
        digitalWrite(led2,LOW);
    }
    else if (num==2){
        digitalWrite(led1,HIGH);
        digitalWrite(led2,HIGH);
    }
    else if (num==3){
        digitalWrite(led1,LOW);
        digitalWrite(led2,LOW);
    }
```

➲ 程式列印

```
const byte pb=53;
const byte led1=22;
const byte led2=23;
byte num=0;
byte b;
long t1,t2,t;
void setup() {
    Serial.begin(9600);
    pinMode(pb,INPUT_PULLUP);
    pinMode(led1,OUTPUT);
    pinMode(led2,OUTPUT);
    DDRA=B11111111;
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    //t1 = millis();
}
void loop() {
    t1=t2=millis();
    do{
        b=digitalRead(pb);
        if (b==LOW)      {
            t1 = millis();
            while(digitalRead(pb)==LOW)
                delay(10);
            num=num+1;
```

```
        }
        t2=millis();
        t=t2-t1;
    }while( (t<500) );//根據習慣調整
    //當使用者有按鈕時，等待 0.5 秒，並統計共按幾下
    Serial.println(num);//這是一個很好的除錯工具，您可以觀察前面程式正
確與否
    if (num==1){
        digitalWrite(led1,HIGH);
        digitalWrite(led2,LOW);
    }
    else if (num==2){
        digitalWrite(led1,HIGH);
        digitalWrite(led2,HIGH);
    }
    else if (num==3){
        digitalWrite(led1,LOW);
        digitalWrite(led2,LOW);
    }
    num=0; //計數器歸零
}
```

➲ 補充說明

1. Serial.println(num);是一個很好的除錯工具，您可以開啓序列埠監控視窗，一面按鈕，一面觀察 num 的值，這樣就可觀察前面程式正確與否。

➲ 自我練習

1. 請用 1 個按壓開關控制 3 個 LED，按一下時亮第 1 個燈；按兩下時亮第 2 個燈；按三下時亮第 3 個燈；按四下時才全熄滅。

▶範例 5-1i

請寫一程式，使用一個按壓開關，控制兩個 LED，長按 1 秒亮一個燈，長按 2 秒亮兩個燈，長按 3 秒全熄滅。

➲ 電路設計

同上範例。

➲ 思考步驟

1. Arduino 提供一個時間 millis()函式，它可以傳回程式執行後所經過的秒數，單位是千分之一秒(ms)，所以只要紀錄按鍵前瞬間與放開後瞬間的時間，並將兩者相減，即爲使用者按下按鍵所經過的時間，程式如下。

```
    t1=t2=millis();
    b=digitalRead(pb);
    if (b==LOW)      {
        t1 = millis(); //此為按下去的瞬間
        while(digitalRead(pb)==LOW)
            delay(10);
        t2=millis();   //此為放開的瞬間
    }
    t3=t2-t1;//t2-t1 就是按鍵的時間，單位是 ms
```

2. 然後再根據時間的長度，決定執行那一功能，程式如下：

➲ 程式列印

```
const byte pb=53;
const byte led1=22;
const byte led2=23;
byte num=0;
byte b;
long t1,t2,t3;
float t;
void setup() {
    Serial.begin(9600);
    pinMode(pb,INPUT_PULLUP);
    pinMode(led1,OUTPUT);
    pinMode(led2,OUTPUT);
    DDRA=B11111111;
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
}
void loop() {
    t1=t2=millis();
    b=digitalRead(pb);
    if (b==LOW)      {
```

```
        t1 = millis();
        while(digitalRead(pb)==LOW)
            delay(10);
        t2=millis();
    }
    t3=t2-t1;//t2-t1 就是按鍵的時間，單位是 ms
    t=float(t3/1000.0);//要除以 1000.0，那才會得到實數
    //若除以 1000，那會得到整數
    Serial.println(t);//這是一個很好的除錯工具，您可以觀察前面程式正確否
    if (t<0.1) {}
    else if  (t<=1.4){
        digitalWrite(led1,HIGH);
        digitalWrite(led2,LOW);
    }
    else if (t<2.4){
        digitalWrite(led1,HIGH);
        digitalWrite(led2,HIGH);
    }
    else {
        digitalWrite(led1,LOW);
        digitalWrite(led2,LOW);
    }
}
```

➲ 自我練習

1. 請用一個按壓開關控制 8 個 LED，按一秒亮一個 LED，按兩秒亮兩個 LED，依此類推，按九秒全熄滅。

➲ 補充說明

1. 軟體功能的確千變萬化，眞的會讓電子工程師大吃一驚，明明只有一個按鈕，竟然就有如此多的變化。使用單晶所完成的硬體功能，可以簡化電路，若是純硬體電路，那一個功能就要一個按鈕，而且還要用 RC 微分電路克服電路彈跳問題。

▶範例 5-1j

請寫一程式，滿足以下條件。

1. 輸入三個數。
2. 將此三個數由小而大輸出。

➲ 演算法則

1. 分別以 a、b 及 c 表示欲排序的資料。
2. 假如 a 大於 b，則 a 與 b 交換，如下圖的(1)。
3. 假如 b 大於 c，則 b 與 c 交換，如下圖的(2)。
4. 假如 a 大於 b，則 a 與 b 交換，如下圖的(3)，排序已完成，共需進行 3 次的比較與交換，如下圖所示。

a	b	c
(1)	(2)	
(3)		

➲ 程式列印

```
void setup() {
   Serial.begin(9600);
   int a=3,b=4,c=2,t;
   if(a>b)
      { t=a; a=b; b=t; }
   if(b>c)
     { t=b; b=c; c=t; }
   if(a>b)
     { t=a; a=b; b=t; }
   //output
   Serial.print("a,b,c 由小而大排列是 ");
   Serial.print(a);Serial.print(",");
   Serial.print(b);Serial.print(",");
   Serial.print(c);
}void loop() {}
```

➲ 補充說明

1. 若有 4 筆資料需要排序，則共需進行 6 次比較與交換，如下圖所示。

a	b	c	d
(1)	(2)	(3)	
(4)	(5)		
(6)			

2. 若有 5 筆資料需排序，則共需進行 10 次比較與交換，如下圖所示，此即爲氣泡排序法。

a	b	c	d	e
(1)	(2)	(3)	(4)	
(5)	(6)	(7)		
(8)	(9)			
(10)				

3. 以上氣泡排序法，處理 3、4 或 5 筆資料的比較與排序，其比較與交換的次數尚可克服。但是，若欲排序的資料超過 5 個，例如 20 筆資料欲排序，則應待迴圈與陣列敘述介紹以後，程式才有精簡的寫法。

➲ 自我練習

1. 將 4 個數由大而小排序輸出。
2. 將 5 個數由小而大排序輸出。
3. 若有 5 個熱敏電阻，編號分別是 1,2,3,4,5，監控 5 個地方的溫度，請寫一個程式，將這些溫度由小而大輸出，輸出時要含編號，例如，

4	2	1	5	3
200	300	400	500	600

5-2 switch...case

一個決策流程點若同時擁有三個或三個以上的解決方案，則稱此爲多向分歧決策流程。多向分歧決策流程雖也可使用 5-1 的巢狀 if else 解決，但卻

增加程式的複雜度及降低程式可讀性，若此一決策流程點能找到適當的運算式，能使問題同時找到分歧點，則可使用 switch case 敘述。switch case 語法如下：

```
switch（運算式）
  {
    case 常數 1：
        敘述區塊 1；
        break;
    case 常數 2：
        敘述區塊 2；
        break;
    case 常數 3：
        敘述區塊 3；
        break;
    〔default:
        敘述區塊 n；〕
  }
```

以上語法說明如下：

1. switch 的運算式值僅能爲整數或字元。
2. case 的常數僅能整數或字元，且其資料型態應與上面的 switch 運算式相同。
3. 電腦將會依 switch 的運算式值，逐一至常數 1、常數 2 尋找合乎條件的 case，並執行相對應的敘述區塊，直到遇到 break 敘述，才能離開 switch。
4. default 可放置特殊情況，也就是沒有適當的 case，則執行 default。若省略 default，且若沒有任何 case 滿足 switch 運算式，則程式會默默離開 switch 敘述。（備註：語法中，兩旁加中括號表示此敘述可省略。）
5. 敘述區塊可放置任何合法的敘述，當然也可放置 switch 或 if。
6. 以下敘述，可將 1、2、3、4 轉爲對應的季節。

```
void setup() {
    Serial.begin(9600);
    byte a=1;
```

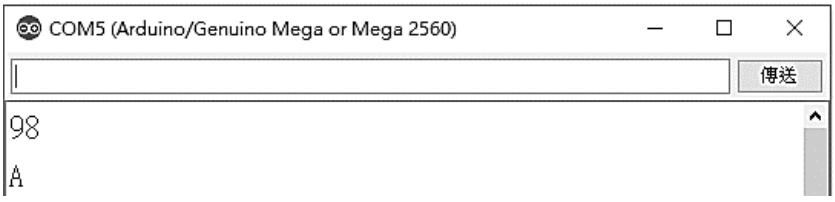
```
    String b="";
    switch(a)
    {
       case 1:
         b="Spring";     /*春*/
         break;
       case 2:
         b="Summer";   /*夏*/
         break;
       case 3:
         b="Fall";        /*秋*/
         break;
       case 4:
         b="Winter";     /*冬*/
         break;
     default :
         b="input error";
    }
    Serial.println(a);
    Serial.println(b);
}void loop() {}
```

7. 有些語言可用逗號將兩種 case 放在一起，但在 C/C++、Arduino 語言中每一 case 僅能放置一個常數，所以若兩個或兩個以上 case，有相同的處理方法，則應將兩個 case 分成兩個敘述，請看以下範例。

▶範例 5-2a

試以 switch case 重作範例 5-1a。

➲ 執行結果



➲ 程式列印

```
void setup() {
   Serial.begin(9600);
   byte a=98;
   String b="";
   switch(a/10) {
      case 10:
      case 9:
         b="A";
         break;//do not leave out
     case 8:
         b="B";
         break;
     case 7:
         b="C";
         break;
     case 6:
     case 5:
     case 4:
     case 3:
     case 2:
     case 1:
     case 0:
         b="D";
         break;
  }
   Serial.println(a);
   Serial.println(b);
}void loop() {}
```

➲ 自我練習

1. 請嘗試將 b=″A″;下一列的 break;去掉，並輸入 92 分，觀察執行結果。
2. 請將範例 5_1a 的自我練習 1～4，以 select case 重作。
3. 開關控制。2 個指撥開關似乎僅能有 2 種控制變化。請寫一程式，讓 2 個指撥開關有 4 種控制變化。例如，

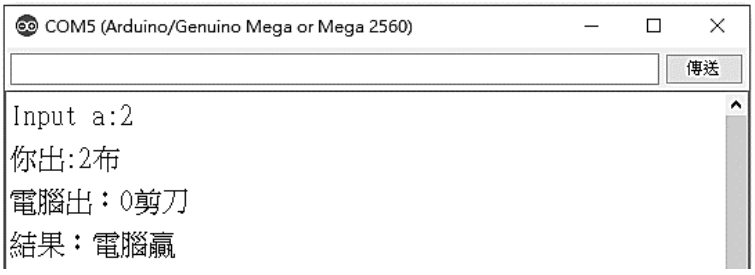
編號	開關 1	開關 2	結果
1	0	0	亮 1 個 LED
2	0	1	亮 2 個 LED
3	1	0	亮 3 個 LED
4	1	1	亮 4 個 LED

4. 請產生 0 到 8 的亂數，並依照亂數點亮對應 LED。例如，產生 3，那就亮第 3 個 LED。
5. 請設計一個電路，可用 8 個 LED 顯示溫度的高低，本例 LED 僅亮一個，溫度低則亮最左邊，溫度提升則 LED 右移。

▶範例 5-2b

猜拳遊戲。請寫一個程式，可以由人和電腦猜拳。

➲ 執行結果



➲ 程式列印

```
void setup(){
  Serial.begin(9600);
  randomSeed(analogRead(0));
}
void loop() {
    int a,b;
    String a1,b1,r="";
    Serial.print("Input a:");
    while(Serial.available() ==0) {}  //在此等待使用者輸入
    a=Serial.parseInt();
    Serial.println(a);
    b = random(0, 2);// get a random number from 0 to 2
```

```
switch (a){
    case 0:
       a1="剪刀";
       switch(b){
          case 0:
             b1="剪刀";
             r ="平手" ;
           break;
          case 1:
             b1="石頭";
             r ="電腦贏" ;
             break;
          case 2:
             b1="布";
             r ="你贏";
             break;
          }
       break;
     case 1:
         a1="石頭";
         switch(b){
            case 0:
               b1="剪刀";
               r ="你贏" ;
               break;
            case 1:
               b1="石頭";
               r ="平手" ;
               break;
            case 2:
               b1="布";
               r ="電腦贏";
               break;
         }
         break;
      case 2:
         a1="布";
         switch(b){
            case 0:
               b1="剪刀";
```

```
                    r ="電腦贏";
                    break;
                case 1:
                    b1="石頭";
                    r ="你贏";
                    break;
                case 2:
                    b1="布";
                    r ="平手";
                    break;
            }
            break;
        }
        Serial.print("你出:");
        Serial.print(a);
        Serial.println(a1);
        Serial.print("電腦出：");
        Serial.print(b);
        Serial.println(b1);
        Serial.print("結果：");
        Serial.println(r);
        delay(2000);
}
```

➲ 自我練習

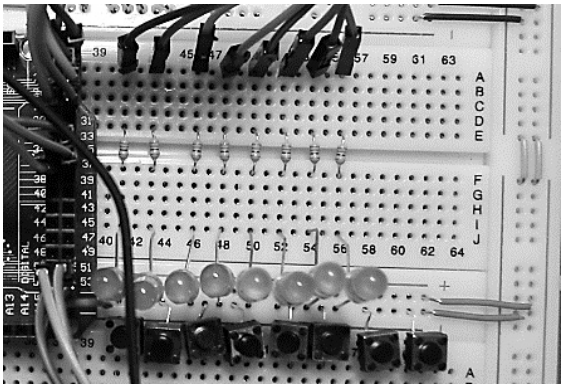
1. 請寫一程式，可以執行三個人的猜拳遊戲，並評定勝負（可讓電腦產生兩個亂數，使用者輸入一個，且由序列埠監控視窗輸出結果。）
2. 同上範例，但改爲使用 3 個按壓開關代表輸入剪刀、石頭、布，3 個 LED 分別表示電腦贏、人贏或平手。
3. 同上題，但改爲只有一個按壓開關，按一下表示『剪刀』，按兩下表示『石頭』，按三下表示『布』。
4. 同上題，但改爲只有一個按壓開關，按一秒表示『剪刀』，按兩秒表示『石頭』，按三秒表示『布』。

▶範例 5-2c

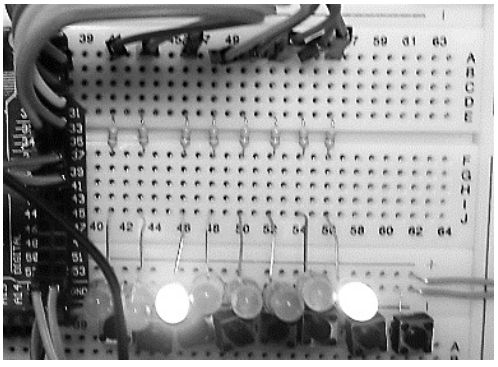
兩人猜拳遊戲。請設計一個電路，雙方各有 3 個按鈕，用來輸入拳法；雙方各有 3 個 LED，用來表示自己和對方的拳法；用 3 個 LED 評判勝負或平手。其次，要等到兩個人都同時按了按鈕，猜拳才有效，才能顯示雙方拳法；接著，延遲 2 秒，顯示勝負；再延遲 2 秒後，燈號全熄，可繼續猜下一次。

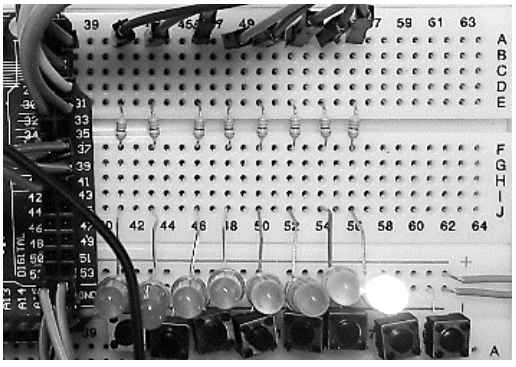
➲ 執行結果

1. 下圖是零件配置實體圖。



2. 下圖左是雙方一起按鍵的結果。
3. 下圖右是評判結果。





➲ 程式列印

```
//The Left is b,The Right is a
//The 0 is scissors, 1 is stone ,2 is cloth
```

```
//The pushbutton 11,10,50,51,52,53 ,The name is
b1,b2,b3,a1,a2,a3
//The Led is PORTA 27,26,25,24,23,22,The name is
b1l,b2l,b3l,a1l,a2l,a3l
const byte b1l=27;
const byte b2l=26;
const byte b3l=25;
const byte a1l=24;
const byte a2l=23;
const byte a3l=22;
const byte c[]={53,52,51,50,10,11};
void setup(){
  Serial.begin(9600);
  for (int i=0;i<=5;i++)
     pinMode(c[i],INPUT_PULLUP);
  DDRA=B111111;
  PORTA=0;
}
byte b1d,b2d,b3d,a1d,a2d,a3d;
void loop() {
    byte d;
    byte a,b;
    do{
         d=PINB;    //讀取 6 個按左開關
         b1d=bitRead(d,5);
         b2d=bitRead(d,4);
         b3d=bitRead(d,3);
         a1d=bitRead(d,2);
         a2d=bitRead(d,1);
         a3d=bitRead(d,0);
     }while((a1d==1 && a2d==1 && a3d==1) || (b1d==1 && b2d==1
&& b3d==1 ));  //只要任一方沒按，就要重覆迴圈
      if (b1d==LOW){
         digitalWrite(b1l, HIGH);
         b=0;
      }
      if (b2d==LOW){
         digitalWrite(b2l, HIGH);
         b=1;
      }
```

```
      if (b3d==LOW){
         digitalWrite(b3l, HIGH);
         b=2;
      }
      if (a1d==LOW){
         digitalWrite(a1l, HIGH);
         a=0;
      }
      if (a2d==LOW){
         digitalWrite(a2l, HIGH);
         a=1;
      }
      if (a3d==LOW){
         digitalWrite(a3l, HIGH);
         a=2;
      }
      delay(2000);
      byte j=judge(a,b);
      PORTA=j;
      delay(2000);
      PORTA=0;
}
byte judge(int a,int b){
  byte r;
    switch (a){
        case 0:
           //a1="剪刀";
           switch(b){
              case 0:
                 //b1="剪刀";
                 //r ="平手" ;
                 r=2;
                break;
              case 1:
                 //b1="石頭";
                 //r ="電腦贏" ;
                 r=1;
                 break;
              case 2:
                 //b1="布";
```

```
                //r ="你贏";
                r=4;
                break;
            }
        break;
    case 1:
        //a1="石頭";
        switch(b){
            case 0:
                //b1="剪刀";
                //r ="你贏" ;
                r=4;
                break;
            case 1:
                //b1="石頭";
                //r ="平手" ;
                r=2;
                break;
            case 2:
                //b1="布";
                //r ="電腦贏";
                r=1;
                break;
        }
        break;
    case 2:
        //a1="布";
        switch(b){
            case 0:
                //b1="剪刀";
                //r ="電腦贏";
                r=1;
                break;
            case 1:
                //b1="石頭";
                //r ="你贏";
                r=4;
                break;
            case 2:
                //b1="布";
```

```
                    //r ="平手";
                    r=2;
                    break;
            }
            break;
      }
      return(r);
}
```

➲ 自我練習

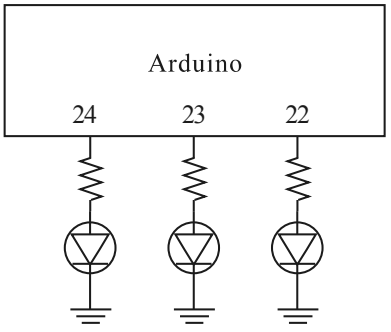
1. 以上範例是兩個同時按下去，猜拳才有效，請您改爲，有按就好，但要等到兩人都按了，才顯示出拳的燈號，並評定結果。

▶範例 5-2d

紅綠燈。同範例 2-3c，但改用 swich case 完成。

➲ 操作步驟

1 請於腳位 24,23,22 配置 3 個紅、黃、綠的 LED，如右圖。



➲ 程式列印

```
void setup() {
   DDRA=B111;//set PORTA output
   //pinMode(24,OUTPUT);
   //pinMode(23,OUTPUT);
   //pinMode(22,OUTPUT);
}
byte i=0;
void loop() {
   switch (i){
      case 0: case 1 :case 2:case 3:case 4:
```

```
            PORTA=B100;
            break;
        case 5:case 6:
            PORTA=B010;
            break;
        case 7: case 8 :case 9:case 10:case 11:
            PORTA=B001;
            break;
    }
    i=(i+1)%12;
    delay(1000);
}
```

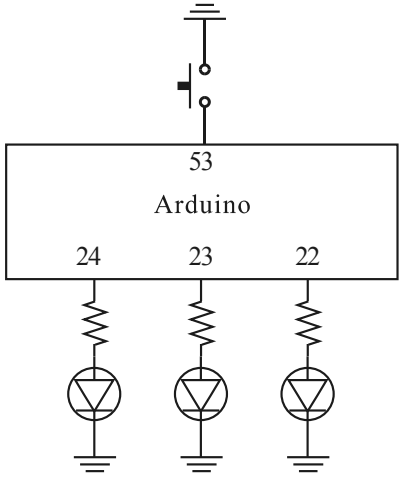
➲ 自我練習

1. 同上範例，但使用六個 LED 控制雙向紅綠燈。這題先不要用陣列，那才能體會陣列的優點。

▶範例 5-2e

同上範例，但改為手動控制紅綠燈，每按一下按壓開關，改變為下一狀態。

➲ 電路設計



➲ 程式列印

```
const byte pb=53;//按壓開關
byte b;
byte i=0;
void setup() {
    DDRA=B111;//set PORTA output
    pinMode(pb,INPUT_PULLUP);
    Serial.begin(9600);
    //pinMode(24,OUTPUT);
    //pinMode(23,OUTPUT);
    //pinMode(22,OUTPUT);
}
void loop() {
    b=digitalRead(pb);
    if (b==LOW){
         while(digitalRead(pb)==LOW)
            delay(10);
         i=(i+1)%3;//改變狀態
    }
    Serial.println(i);
    switch (i){
       case 0:
          PORTA=B100;
          break;
       case 1:
          PORTA=B010;
          break;
       case 2:
          PORTA=B001;
          break;
    }
    delay(100);
}
```

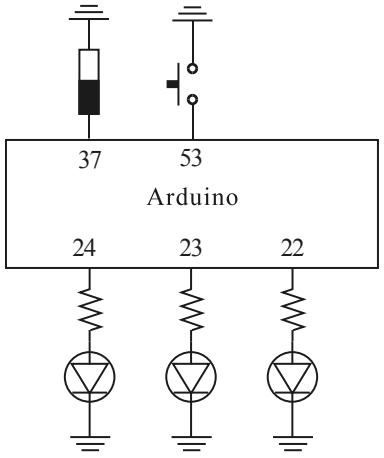
➲ 自我練習

1. 請以手動的方式控制雙向紅綠燈的 6 個 LED。

▶範例 5-2e

使用一個指撥開關控制紅綠燈為自動或手動。自動模式同範例 5-2d，手動模式同範例 5-2e。

➲ 電路設計



➲ 程式設計

```
const byte sw=37;//指撥開關
const byte pb=53;//按壓開關
byte s,b;
byte i=0,j=0;
void setup() {
    DDRA=B111;//set PORTA output
    //pinMode(24,OUTPUT);
    //pinMode(23,OUTPUT);
    //pinMode(22,OUTPUT);
    pinMode(pb,INPUT_PULLUP);
    pinMode(sw,INPUT_PULLUP);
    Serial.begin(9600);
}
void loop() {
    s=digitalRead(sw);
    Serial.println(s);
    switch (s){
       case LOW:   //case 0:也可以 //手動
          b=digitalRead(pb);
```

```
          if (b==LOW){
             while(digitalRead(pb)==LOW)
               delay(10);
             j=(j+1)%3;//改變狀態
          }
          switch (j){
             case 0:
               PORTA=B100;
               break;
             case 1:
               PORTA=B010;
               break;
             case 2:
               PORTA=B001;
               break;
          }
          delay(100);
          break;
    case HIGH://case 1:也可以，本模組為自動
       switch (i){
         case 0: case 1 :case 2:case 3:case 4:
           PORTA=B100;
           break;
         case 5:case 6:
           PORTA=B010;
           break;
         case 7: case 8 :case 9:case 10:case 11:
           PORTA=B001;
           break;
       }
       i=(i+1)%12;
       delay(1000);
    }
}
```

➲ 自我練習

1. 使用一個指撥開關控制紅綠燈爲自動或閃黃燈，自動模式同範例 5-2d。
2. 使用兩個指撥開關控制紅綠燈爲自動、手動或閃黃燈，自動模式同範例 5-2d，手動模式同範例 5-2e。

3. 深夜紅燈秒數過長惹人怨。請設計一個程式，可事先規劃一天裡的所有時段的紅綠燈的秒數。提示：請利用範例 4-3g 電子時鐘。本例先簡化問題，將紅綠燈分爲五種時段三種模式，如下表所示：(本例僅用單向 3 個 LED 即可)

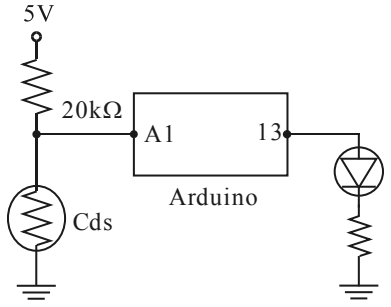
時段	時間	模式
1	22～06	夜間模式
2	06～08	上下班模式
3	08～17	白天模式
4	17～19	上下班模式
5	19～22	白天模式

5-3 實例探討

▶範例 5-3a

夜間自動照明。請寫一程式，可依天氣變暗，LED 自動亮起來，天氣變亮，LED 自動熄掉。

輸入設備：光敏電阻，使用 A1 pin。
輸出設備：LED，使用第 13 pin。



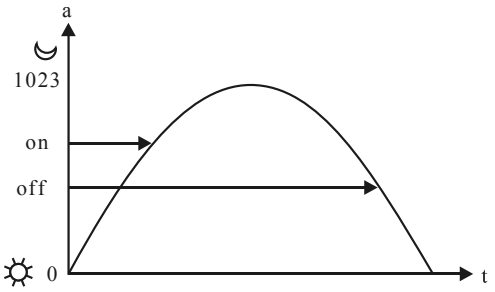
➲ 思考步驟

1. 如何調整與設定 on 的值。手先不要遮光，先觀察序列埠的輸出值，因爲光線變暗，其值會增加，所以將此值約略加一點，當作 on 的值。
2. 這題初學者通常會撰寫程式如下，但請自己觀察執行結果。

```
void loop() {
  a=analogRead(A1);//use A1 pin, a 從 0 到 1023
  Serial.println(a);//輸出在 0 到 1023
  if(a>on)
     digitalWrite(led,HIGH);
```

```
  else
     digitalWrite(led, LOW);
  delay(2000);
}
```

3. 理論上，以上程式沒有問題，但實務上感測器很靈敏，也就是在臨界值的時候，燈光會閃爍，這樣您的路燈會壞掉。所以既然亮起來，就繼續讓他亮，直到天氣較亮時（較亮其值變小），再熄掉就好，如下圖（X 軸是時間，Y 軸是 a 值）：



➲ 程式列印

```
int a;
const byte led=13;
const int on=300;//請自己調
const int off=280;//請自己調
void setup() {
  Serial.begin(9600);
  pinMode(led,OUTPUT);
  digitalWrite(led,LOW);
}
void loop() {
  a=analogRead(A1);//use A1 pin, a 從 0 到 1023
  Serial.println(a);//輸出在 0 到 1023
  if(a>on) {
     digitalWrite(led,HIGH);
  }
  if ( a<off) {
     digitalWrite(led, LOW);
  }
  delay(2000);
}
```

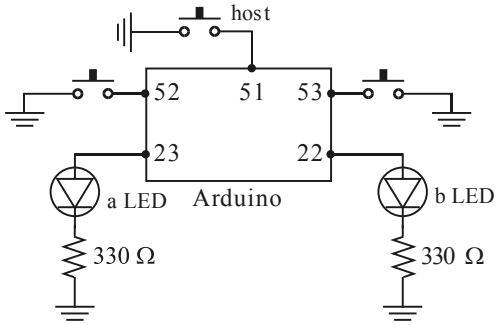
➲ 自我練習

1. 小夜燈、路燈都是這個原理，請自行練習。
2. 洗手台控制。洗手台控制有些是第一次遮光時水流通，第二次遮光時水停止；有些是第一次遮光時，水流通 2 秒停止。小便池則是人站上去時，先沖一點水，人離開時再沖大量的水。本例沒有水流控制閥，請用 LED 代替。
3. 請使用熱敏電阻控制一個 LED，能夠熱了就啓動 LED，冷了就熄滅 LED。
4. 請使用熱敏電阻控制一個 LED，能夠熱了就啓動全亮，溫度下降 LED 亮度也降低。（請用脈寬調變，Pulse Width Modulation，簡稱 PWM 變頻）

▶範例 5-3b

電子搶答器。

輸入設備：雙方各有一個按鈕可搶答，先按的人，燈才會亮。主持人有一按鈕，可取消所有燈號。
輸出設備：雙方各有一個 LED，用來表示取得答題權。



➲ 程式列印

```
const byte a=52;
const byte aled=23;
const byte b=53;
const byte bled=22;
const byte host=51;
void setup() {
  pinMode(a,INPUT_PULLUP);
  pinMode(aled,OUTPUT);

  pinMode(b,INPUT_PULLUP);
  pinMode(bled,OUTPUT);

  pinMode(host,INPUT_PULLUP);//host
```

```
  digitalWrite(aled,LOW);
  digitalWrite(bled,LOW);
  Serial.begin(9600);
}
int a1;
bool ae=true;
int b1;
bool be=true;

void loop() {
    a1=digitalRead(a);
    b1=digitalRead(b);
    Serial.print(a1);
    Serial.print(b1);
    //當我可以按，我按了
    if ( ae & (a1==LOW)) {
        digitalWrite(aled,HIGH);//我燈亮
        be=false;//設定對方無法按
        //只要主持人未按鈕，就等待
        while(digitalRead(host)==HIGH){}
    }
    //當我可以按，我按了
    if ( be & (b1==LOW)) {
        digitalWrite(bled,HIGH);
        ae=false;//設定對方無法按
        while(digitalRead(host)==HIGH){}
    }
    //進行歸位重置動作
    ae=true;
    digitalWrite(aled,LOW);
    be=true;
    digitalWrite(bled,LOW);
}
```

➲ 自我練習

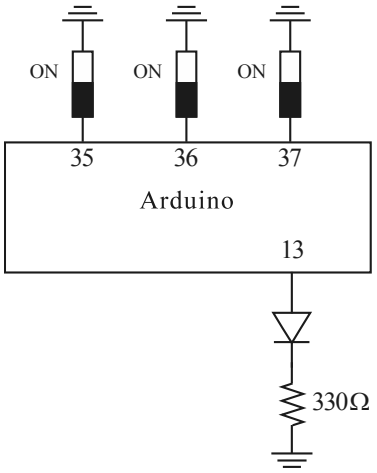
1. 那 3 或 4 個人同時搶答呢？

▶範例 5-3c

表決器。假設有一項評審工作，有 3 位評審，當其中兩人或以上同意，則表示過關且燈亮，請設計此電路。

➲ 電路設計

1. 此題先用指撥開關當作評審按鈕，當有 2 個或以上評審 on 時，表示通過，LED 亮，電路設計如下：



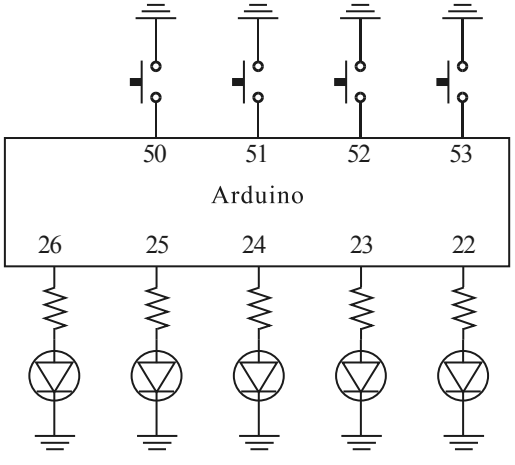
2. 以上想法的程式列印如下：

```
const byte sw1=35;
const byte sw2=36;
const byte sw3=37;
const byte led=13;  //預植的 LED
byte a;
byte b;
void setup() {
   pinMode(sw1,INPUT_PULLUP);
   pinMode(sw2,INPUT_PULLUP);
   pinMode(sw3,INPUT_PULLUP);
   pinMode(led,OUTPUT);
   digitalWrite(led,LOW);
   DDRA=B111;//24,23,22
   Serial.begin(9600);
}
void loop() {
```

```
    a=PINC;
    a=~a;//逐位元反相
    a=a & B00000111;//將不要的位元遮罩
    PORTA=a;
    b=0;
    b=b+bitRead(a,0);
    b=b+bitRead(a,1);
    b=b+bitRead(a,2);
    Serial.println(b);
    if (b>=2)
       digitalWrite(led,HIGH);
    else
       digitalWrite(led,LOW);
}
```

3. 以上作法，程式較簡單，但是評審完，評審還要自己將指撥開關撥回來，有點不方便。

4. 以下是使用 3 個按壓開關讓評審按，這時就要增加一個主持人按鈕，主持人有按鈕時，才表示開始評審，每一評審，按一下按鈕，自己的 LED 亮，超過兩個人通過，則亮通過綠燈，否則亮失敗紅燈。電路如下：



5. 程式設計如下：

```
const byte pb0=50;//host
const byte pb1=51;const byte led1=24;
const byte pb2=52;const byte led2=23;
```

```
const byte pb3=53;const byte led3=22;
const byte led4=25;//pass
const byte led5=26;//failure
byte b1,b2,b3;
byte a1,a2,a3;
byte a=0;
long t1,t2,t;
void setup() {
   pinMode(pb0,INPUT_PULLUP);
   pinMode(pb1,INPUT_PULLUP);
   pinMode(pb2,INPUT_PULLUP);
   pinMode(pb3,INPUT_PULLUP);
   DDRA=B111111;//26,25,24,23,22
   PORTA=0;
   Serial.begin(9600);
}
void loop() {
   t1=t2=millis();
   a=0;
   a1=a2=a3=0;
   PORTA=0;
   //沒按鈕時，在此等待
   while(digitalRead(pb0)==HIGH)
       delay(10);
   //沒按鈕時，pb0 電位是 HIGH，那就重複執行迴圈，這就完成等待的動作
   //Staring
   t1 = millis();
   do{
      //評審 1
      b1=digitalRead(pb1);
      if (b1==LOW)      {
         while(digitalRead(pb1)==LOW)
            delay(10);
         a1=1;
      }
      if (a1==1)
         digitalWrite(led1,HIGH);
      else
         digitalWrite(led1,LOW);
      //評審 2
```

```
        b2=digitalRead(pb2);
        if (b2==LOW)      {
            t1 = millis();
            while(digitalRead(pb2)==LOW)
                delay(10);
            a2=1;
        }
        if (a2==1)
            digitalWrite(led2,HIGH);
        else
            digitalWrite(led2,LOW);
        //評審 3
        b3=digitalRead(pb3);
        if (b3==LOW)      {
            t1 = millis();
            while(digitalRead(pb3)==LOW)
                delay(10);
            a3=1;
        }
        if (a3==1)
            digitalWrite(led3,HIGH);
        else
            digitalWrite(led3,LOW);
        a=a1+a2+a3;
        Serial.println(a);
        if (a>=2)//若通過了，就給燈號
            digitalWrite(led4,HIGH);
        t2=millis();
        t=t2-t1;
    }while( (t<3000)  );//3 秒，根據需求調整
    //時間到了，給燈號
    if (a>=2)
        digitalWrite(led4,HIGH);
    else
        digitalWrite(led5,HIGH);
    delay(2000);
}
```

➲ 自我練習

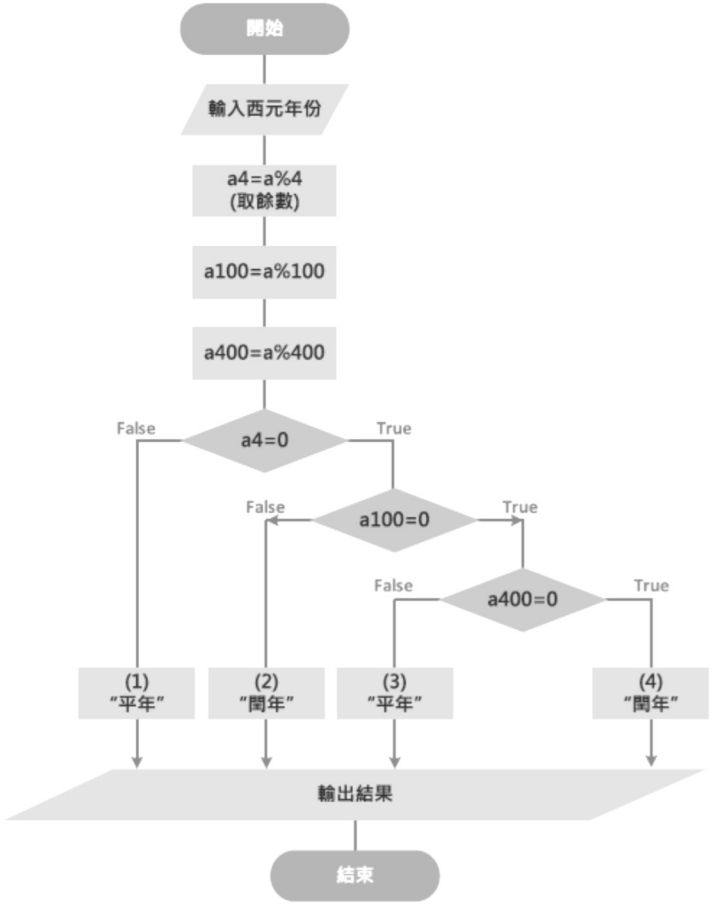
1. 假如有 5 位評審，三位以上通過就通過，請設計電路與程式。

閏年的判斷

西元的閏年爲每 400 年必須有 97 次閏年，其規劃方式如下：

1. 4 的倍數。依此條件共有 100 次。
2. 於 1. 的條件，扣掉 100 的倍數。依此條件，共有 96 次。
3. 於 2. 的條件，再加回 400 的倍數。所以共有 97 次。

以上演算分析，流程圖分析如下：



以下提供一些測試資料，以便對照流程路線。

西元年份	性質	流程路線
3	平年	(1)
4	閏年	(2)
100	平年	(3)
200	平年	(3)
300	平年	(3)
400	閏年	(4)
600	平年	(3)
1200	閏年	(4)
2000	閏年	(4)

▶範例 5-3d

試寫一程式，可以將使用者所輸入的西元年份，判斷其為平年或閏年。

➲ 執行結果

```
Input a:2000
It is a leap year(4)
Input a:1996
It is a leap year(2)
```

➲ 程式列印

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    int a, a4, a100, a400;
    String year="";
    Serial.print("Input a:");
    while(Serial.available() ==0) {}
    a=Serial.parseInt();
    Serial.println(a);
    a4=a%4;
    a100=a%100;
    a400=a%400;
    if(a4==0)
        if(a100==0)
            if(a400==0)
```

```
            year="leap year(4)";
         else
            year="common year(3)";
      else
         year="leap year(2)";
   else
      year="common year(1)";
   Serial.print("It is a ");
   Serial.println( year);
}
```

➲ 補充說明

1. 以上程式，亦可簡化如下，下一章製作萬年曆就可以派上用場。

```
if (a % 4 == 0 & ! (a % 100 == 0) | (a % 400 == 0 ))
      year=" 閏年";
   else
      year=" 平年";
```

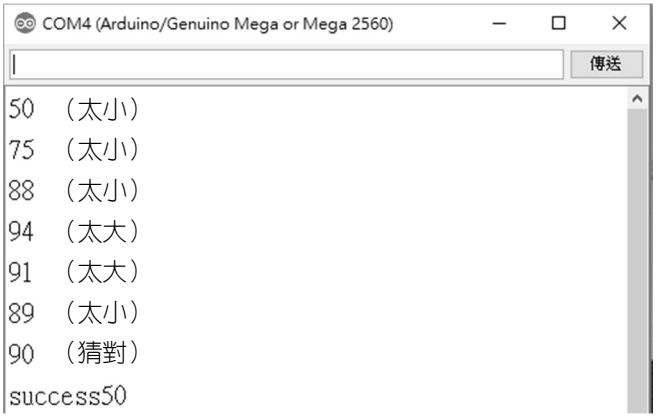
▶範例 5-3e

電腦猜數字。

輸入設備：三個按壓開關。
輸出設備：序列埠監控視窗。
功能說明：使用者默想一個 1 到 100 的數字，由單晶猜，單晶所猜的數字輸出於序列埠監控視窗，使用者使用三個按鈕回答太大、太小、答對。

➲ 執行結果

下圖是我默想 90，請電腦猜，電腦使用二分猜值法猜數字，我逐一回答太小或太大等過程。



➲ 程式列印

```
const byte pb1=51;
const byte pb2=52;
const byte pb3=53;
byte a1=1,a2=100,a;
byte b1,b2,b3;
bool success=false;
void setup() {
   pinMode(pb1,INPUT_PULLUP);
   pinMode(pb2,INPUT_PULLUP);
   pinMode(pb3,INPUT_PULLUP);
   Serial.begin(9600);
}
void loop(){
   do{
     a=(a1+a2)/2;
     Serial.println(a);
     do{
       b1=digitalRead(pb1);
       b2=digitalRead(pb2);
       b3=digitalRead(pb3);
     }while (b1==HIGH & b2==HIGH & b3==HIGH) ;//都沒按要等待
     if (b1==LOW)      {     //太小
        a1=a+1;
     }
     else if (b3==LOW){  //太大
        a2=a-1;
     }
     else{
        success=true; //答對了
```

```
        Serial.print("success");
      }
      delay(1000);//單晶太快了，讓他等一會再讀按鍵
    }while(not success);//沒成功要繼續
    a1=1;a2=100;
}
```

➲ 補充說明

1. 單晶程式要出神入化，就是要好好學習迴圈，請繼續研讀下一章。
2. 此即爲二分猜值法，在此熱身，下一章還有更詳細說明與範例。
3. 本例若使用循序猜值法，從 1,2,3,4 逐一循序猜測，那平均要 50 次才可猜出；但若使用二分猜值法，則最多不會超過 7 次。（因爲 $2^6 < 100 < 2^7$）

➲ 自我練習

1. 使用者猜數字。單晶用亂數產生一個 1 到 100 的數字，由使用者用電腦的鍵盤來猜。單晶用 3 個 LED 表示太大、太小或猜對。

習　題

1. 烘碗機。請設計一個程式，可以使用 1 個按鈕，控制 3 個燈，按一下時，亮第一個燈，且持續一分鐘熄掉；按兩下時，亮第 2 個燈，且持續 1 分鐘熄掉，轉亮第 1 個燈，再持續 1 分鐘也熄掉；按三下時，亮第 3 個燈，且持續 1 分鐘熄掉，轉亮第 2 個燈，持續 1 分鐘熄掉，轉亮第 1 個燈，持續 1 分鐘熄掉；按四下時，全熄滅。
2. 微波爐控制。假如微波爐有 8 個按鈕，分別是『0,1,2,3,4,5,6,加熱』，按一下『加熱』，LED 亮，就開始倒數計時 30 秒，時間到 LED 熄滅；按 2 下『加熱』表示倒數計時 60 秒，LED 亮，時間到熄滅；按『2』、『0』、『加熱』表示加熱 20 秒，LED 亮，時間到熄滅，請寫程式完成此控制。