PWM 的使用

一、 什麼是 PWM?

談過基本訊號控制以後，我們就可以來些不同的變化，這裡要介紹一種常用的訊號規格"PWM"，PWM 全名叫做 **Pulse Width Modulation**，脈衝寬度調變，這是一種利用前面談過的數位(Digital)訊號來"模擬"類比(Analog)訊號的方式，最常見的一些應用是電子類零件的電壓調整，例如:馬達快慢控制、燈光的明暗控制、螢幕亮度控制、喇叭的大小聲/聲音頻率高低控制等等。實際的作法是利用控制數位訊號的腳位，用程式高速切換腳位的 HIGH / LOW 的狀態，通常會搭配系統 TIMER，來輸出一連串的方波，每個方波的長相，如圖 1

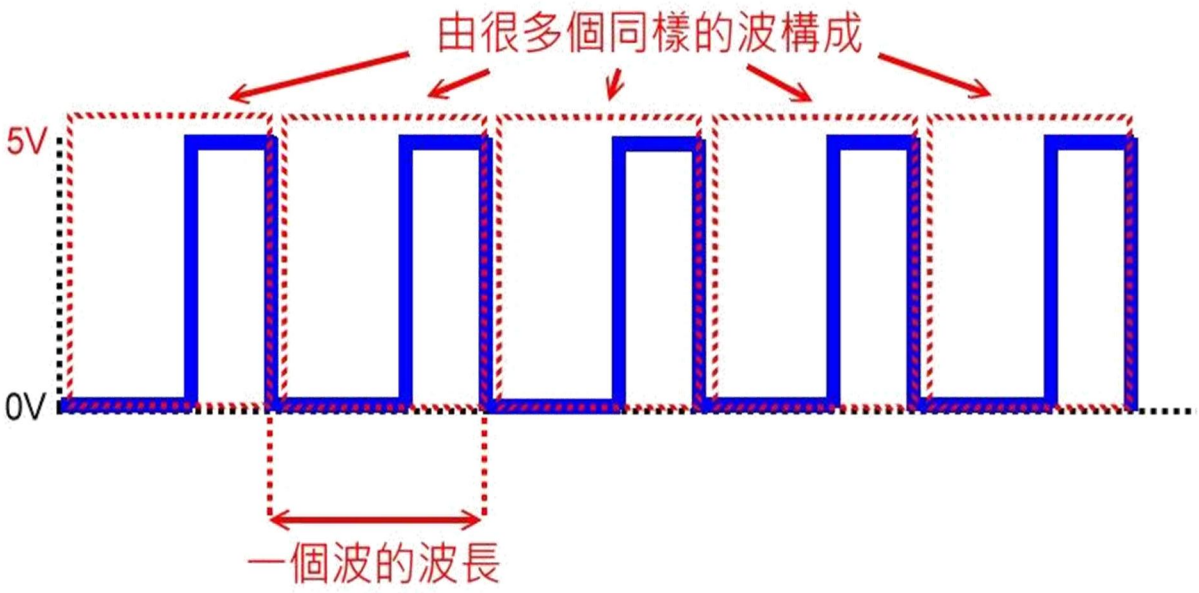

圖 1. PWM 的長相

每一個波型所佔的時間長度，又稱為周期，頻率(Hz)就是指單位時間內出現多少個這樣的波型，EX: 一秒內出現 50 個這種樣子的波型，我們就說頻率就是 60Hz/sec。大量重複的產生這種波，並且透過調整他的周期、頻率和波長，就可以模擬出各種很像類比訊號的輸出，來實作各種應用，如圖 2

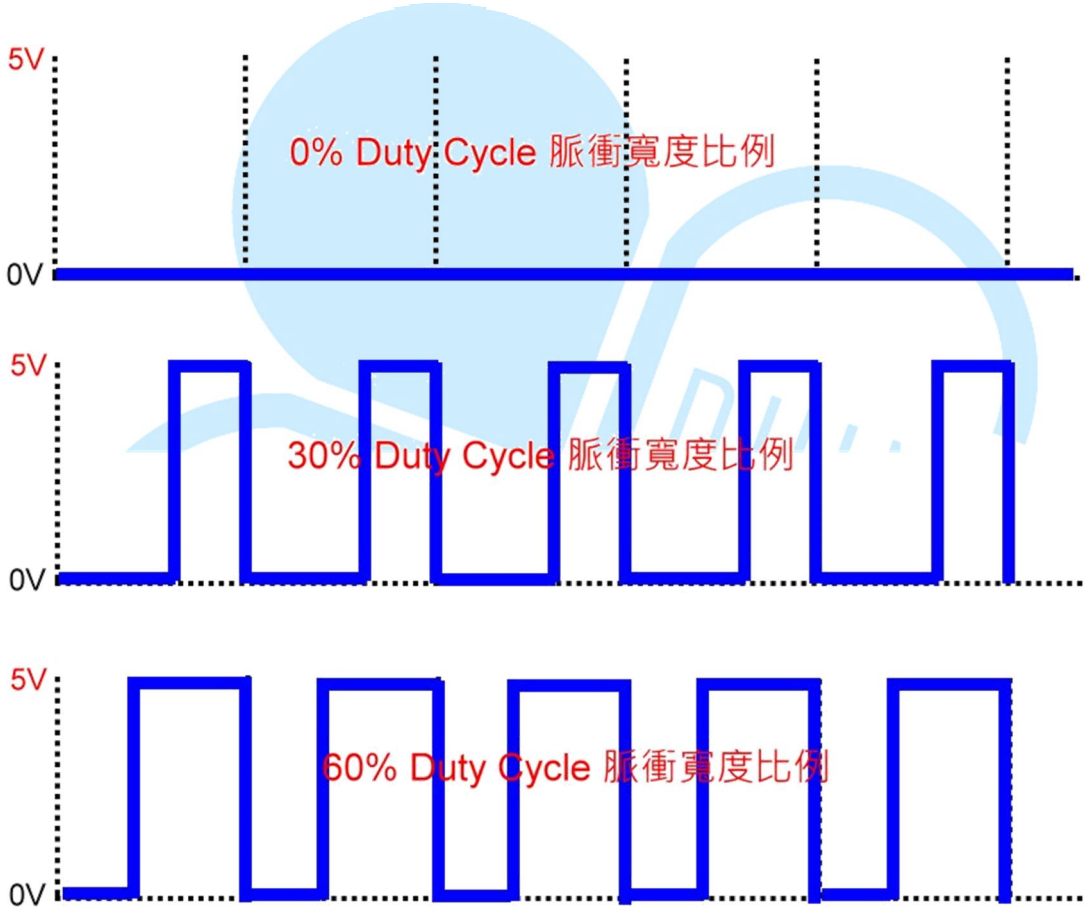

圖 2. 各種不同佔空比的 PWM

要實作普通的 PWM 波型，並不需要寫很複雜的程式(後面會談到複雜或是高頻率的波型實作)，例如想作出圖 2 的第一種波型，只要使用這個指令 analogWrite(腳位,0); 就可以輕易辦到；第二種波型，只要使用這個指令 analogWrite(腳位,78); 就可以輕易辦到；第三種波型，則使用這個指令

analogWrite(腳位,153); 即可辦到。其中，指令裡面那個數字只是個概算，PWM 控制指令的參數範圍介於 0~255，所以若是要 30%為 HIGH，那就是 255 * 30% =大約是 78 左右。

關於 analogWrite 這個指令，是給特定的 digital 數位腳位使用的，會以數位的方式輸出 PWM 訊號，指令格式:

```
analogWrite(腳位, 數值)
```

其中的數值必須介於 0~255 ，對應到 0~5V，所以解析度為 5V / 255 = 0.0196V

像上面使用了 analogWrite(腳位,160); 其中的 160 * 0.0196V = 代表會輸出 3.13V 的模擬類比電壓，頻率則是 1KHz。而腳位的設定則必須是 CAKE 板子上有標示"~" 符號的腳位才能使用，如圖 3

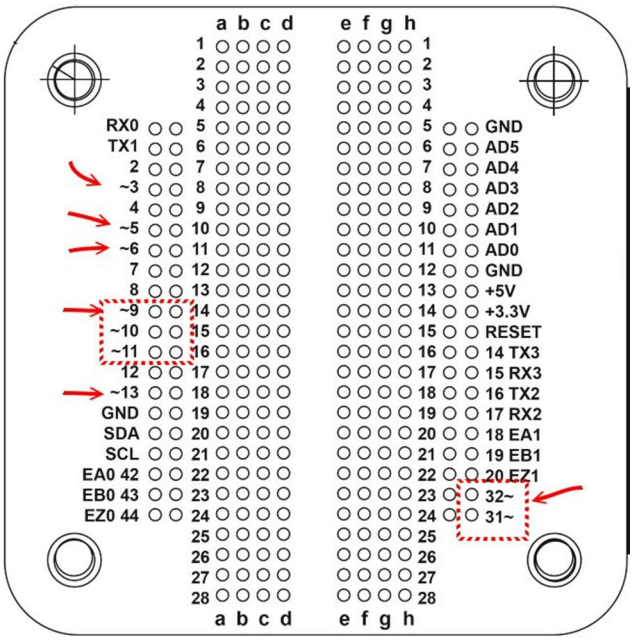

圖 3. 86Duino EduCake 上的 PWM 腳位分佈

更重要是，因為這其實還是數位訊號，還是由一堆不同長短的 HIGH/LOW 狀態構成，對於儲存、傳輸、抗干擾的能力，都是能有很好的表現也能很輕易辦到，也因此用途就日漸的廣泛。最常見又簡單的例子就是控制 LED 燈的亮滅，如以下的片段程式碼:

```
digitalWrite(3,HIGH); // 設定 HIGH 點亮 3 號腳位的 LED 燈
delay(1000);
digitalWrite(3,LOW); // 改成 LOW 熄滅 3 號腳位的 LED 燈
```

以上這段程式碼只能控制 3 號腳位的 LED 燈亮一秒後熄滅，看起來就很單純，是很直覺的數位控制方式，但他只能這樣單純的亮暗切換，看久會很單調。當我們以類比的想法來做這件事情的時候，可以讓他變得很有趣，PWM 就是一種很簡單可以實現這種想法的功能，如以下的程式碼:

```
for(int a=0;a<=255;a++) // 這個迴圈讓 3 號腳位的 LED 燈慢慢亮起來
    {analogWrite(3,a); delay(10);}
delay(1000);
for(int a=255;a>=0;a--)// 這個迴圈讓 3 號腳位的 LED 燈慢慢熄滅
    {analogWrite(3,a); delay(10);}
```

執行效果會變成，LED 燈會慢慢的由暗變亮，到最亮，然後再慢慢的變暗，到熄滅。重覆這個動作並且若做些小微調，還可以讓燈看起來好像有在呼吸的效果，筆者在睡房裡放了一個這樣的燈，並且打開開關以後就可以慢慢的變亮變暗，並且整個動作的週期越來越長，約半小時後完全熄滅不再有動作，根據實驗，搭配輕柔且慢慢變小聲的音樂騙小孩子睡著是很有用的哩，當然程式碼就比這裡列出的要長一些了。而同樣的程式碼不要單純的在迴圈裡面執行 a++，而是改成執行使用小範圍亂數去加減他，就可以模擬出類似燭光搖曳的效果，呼吸燈瞬間立刻變成人工燭光，很有趣喔；在額外加上聲音感測器，立刻就變成可以用嘴去 "吹熄滅"的人工燭光了，也可以用拍手的聲音來達成同樣目的，這可是擺在餐廳桌上非常實用的應用，讀者可以再想想還可以有哪些變化。

當然，以上這段簡易的程式碼其實有些問題在其中，LED 燈要能亮起來需要一定的電壓才行，EX: 2.5V，PWM 要能模擬出這樣的電壓可能需要裡面那個參數設定到 125 以上才行，這樣程式中的迴圈，在 0~124 左右的時候，燈都會是全滅的狀態，看不太出來變化；同樣的，LED 燈可能在 4V 的時候就非常亮了，4~5V 這個區間都會是非常亮的樣子，根本看不出有沒有比較亮的差別，所以實際要做這功能，須要先測試出所選用的 LED 的最暗到最亮的範圍，再去修改程式裡面的迴圈值會有比較好的效果。

以上談的是基本的 PWM 功能，86Duino EduCake 裡面提供的基本 PWM 訊號，頻率預設是 1000Hz/ sec，也就是每個波的長度為: 1sec / 1000 = 0.001 秒，這種速度已經足以應付很多種日常需求了，當然也有很多別的方式可以有更精確的 PWM 輸出來使用。請注意因為腳位的輸出電流非常小，

若要推動大型的電力需求，EX: 馬達、電風扇、甚至冷氣(搞不好讀者想作變頻 DC 冷氣)，都是沒有問題，就是要想辦法把電流放大來用就好，後面我們就來看看 PWM 的一些應用。

二、 PWM 的應用(一)呼吸燈

這個應用電路圖很簡單，只是接一顆 LED，如圖 4，這部分筆者是直接把電阻和和 LED 焊接在一起，使用上比較簡單，只要插兩條線就好，讀者也可以分開插到麵包板上面

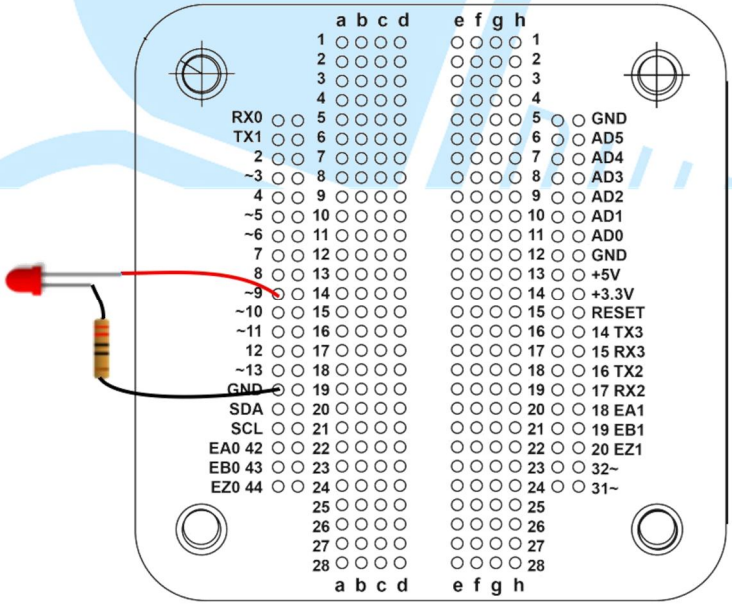

圖 4. 接線圖

直接來看看程式

```
int brightness = 0; // 亮度變數

int fade= 5; // 亮度變化的變數

int delaytime= 50; // 變化速度控制
```

```
void setup()  {
  pinMode(9, OUTPUT);
}

void loop()  {
  analogWrite(9, brightness);
  brightness = brightness + fade ;
  if (brightness <= 50 || brightness >= 205) {
    fade = -fade;
  }
  delay(delaytime);
}
```

其中的 if (brightness <= 50 || brightness >= 205) 裡面那兩個數字就是之前講過的，LED 燈從完全熄滅到稍微開始發亮的位置(我的 LED 是 50)，另一個數字就是最亮的位置，我的是 205，讀者使用的要自己測試，以便確實的去改變這兩個數字，不然會變成變暗以後，要等一陣子才又會變亮；變最亮以後，要等好一陣子才會開始變暗，這中間的效果就看個人如何去調整了。

那要如何才能作到前面談過的燭光效果呢?! 首先，想想看燭光的樣子

1. 燭光一定是會發亮。 << 廢話!!

2. 燭光會搖<< 這是風吹空氣擾動的效果，LED 想要有這種效果只好用馬達搭配機構來搖動他模擬了，這裡因為我們的裝備只有 LED 就先跳過，讀者可結合後面談的馬達控制來做到，不過更簡單是利用透明玻璃紙條+風吹，可以作出類似效果，遠遠看還不錯，有些 7-11 的關東煮上方會有類似這種裝置可以去參考看看。

3. 燭光會因為被風吹而導致亮度的變暗一點或是變亮一點，但很不一定。

其實，上面在列出這些東西的時候，列了第一個條件筆者自己就笑出來，這看起來就很白癡，但，這是必要的，要解決任何問題的時候，必須先把問題拆解成很簡單的一個一個小問題，再拆解成很接近 IDE 程式提供的指令/語法片段的小問題，這樣就可以一個一個轉換成程式，像拼積木一樣把程式組合起來答案就出來了，這對未來解決大型專案是很重要的，以下來看看這段程式(電路不需要任何修改)

```
int brightness = 160;
int fade;
int delaytime= 20;


void setup()  {
  pinMode(9, OUTPUT);
 analogWrite(9, brightness); //  第一個條件，燭光一定是亮的
```

```
 randomSeed(analogRead(0)); //這是為了第三個條件初始化亂數種
子
}

void loop()  {
 fade= random(0, 61); //  先產生一個 0~60  的亂數
 fade-= 30 //  把他減 30，變成-30~+30
   brightness = brightness + fade ; //  這是關鍵的第三條件
 // fade 加入到 brightness  裡面會造成光度可能變大或是變小的不定
變化
  if (brightness <= 60 )    //  需防止燭光熄滅，所以很暗的時候就不能
在暗下去
    brightness = 60;
 if (brightness >= 200)    //  也不要讓燭光太強，所以一個程度就要停
止變更亮
    brightness = 200;
 analogWrite(9, brightness);
  delay(delaytime);
}
```

裡面的那些數字和參數好好的依照需求來微調，出來的效果是很令人滿意的，尤其還可以搭配 RGB3 色 LED 燈，更可作出很多令人意想不到的變化。讀者可以再仔細的想想，生活中有哪些地方是會使用到燈光的效果呢?

三、 PWM 的應用(二)SERVO 馬達控制

各種遙控用的 SERVO 或是任何需要使用 PWM 控制的馬達都可以使用 EduCake 的 PWM 訊號來作控制，這裡就用一顆最容易取得的伺服馬達來作實驗，如圖 5 的 SG90 這種常用的入門級遙控伺服馬達。

圖 5. 常見 SERVO 的接頭長相

這種遙控用的小 SERVO 馬達一顆只要幾十元台幣，拍賣上到處都是，專門用來控制遙控飛機或是小汽車用的。請注意到他的接腳是橘(訊號線，有些不同廠牌的馬達可能是白色)、紅(接正電源)、棕(GND，有些不同廠牌的馬達可能是黑色)三種顏色，其中的橘色就是直接接到 CAKE 的 PWM 腳位就能控制他。這顆馬達是品質很差，很不 OK 的入門級玩具，但拿來做實驗非常好，至少玩壞不會心痛。規格如下

1. 重量 9g

2. 尺寸 23*12.2*29mm

3. 扭力 1.8kg/cm(4.8V，最高到 6V)

4. 速度 0.1sec/60degree(4.8v) ，這是說 SERVO 在 4.8V 電壓時，轉 60 度最快要 0.1 秒

5. Dead band width 10us，個人喜歡翻譯成"無反應間隔"，就是上次對 SERVO 送出的 PWM 訊號和這次的訊號相差若小於 10us，SERVO 有很大機率是完全不會有反應的，或是講成 SERVO 的控制精確度也行，這主要是和 SERVO 裡面的 VR 解析度有關

6. 運動角度約 150 度 << 這很重要，一定要查清楚使用的馬達角度，不然送過大的 PWM 波，可能會讓馬達卡住導致燒毀

類似的 SERVO 像是（http://www.roboard.com/Files/RS-0263/RoBoard_RS-0263.pdf）這種馬達，精確度就明顯好很多，無反應間隔也明顯小，力量又大，一般入門後要玩比較精細的玩家就會採用，整個作出來的作品表現會明顯提升，畢竟是一分錢一分貨。 電路接法如圖 6，這只是純粹簡單的測試，且這顆馬達很不耗電才這樣接。請注意其實伺服馬達是不能這樣接的，馬達的電流相對於控制板來說比較大，在有負載的情況下容易燒毀控制板(EduCake 內部有過流保護不用擔心，其他種類的單晶就得注意了)，一般最好是把馬達的正極接到獨立的電源，然後該獨立電源的 GND 和控制板的 GND 接在一起，甚至可以再加一個二極體防止逆電流，這樣就可以防止馬達電流過強導致控制板有問題，或是控制板電流不夠導致馬達不會動。

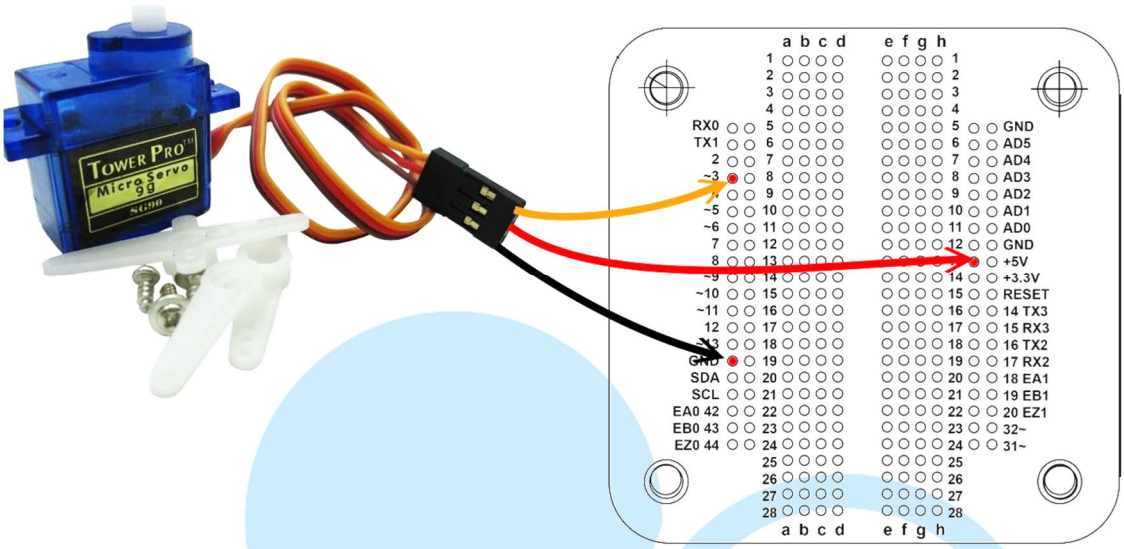

圖 6. SERVO 的接線圖

搭配底下這個程式就能讓馬達左右掃來掃去，這也是筆者最常用來測試 SERVO 耐用度的方式，使用時我還會把 SERVO 擺臂上綁重物，測試他的實際出力、耐用度、消耗電流、溫度的變化等等參數，把馬達徹底操勞到壞掉為止，以便未來使用的時候能掌握設計出來的機器人的功能。且買 SERVO 除了這些注意事項，還要看 SERVO 的 Dead band width 的區域大小(越小越好)，還要看速度、尺寸、長相(牽涉機構好不好設計和固定等等)、齒隙等等的一堆問題，並沒有那麼簡單，還是需要一些使用經驗來挑選適合的 SERVO。

```
#include <Servo.h>
Servo myservo; //  建立 Servo 物件，控制伺服馬達
void setup()
```

```
{
  myservo.attach(3);   // 連接 digital 腳位 3
}
void loop()
{
  for(int a = 30; a <= 150; a++){
    myservo.write(a); //  控制馬達角度 30~150 度
    delay(20);
  }
  for(int a = 150; a >= 30; a--){
    myservo.write(a); // 150~30 度
   delay(20);
  }
}
```

這個程式裡面，除了註解以外還有幾個需要注意的地方，首先第一個迴圈裡面，從 30 跑到 150 度，有讀者會問為何不是 0~180，或是 0~360 度呢?!這是因為 SERVO 裡面的定位是透過一顆 VR 讀取數值來定位，若是讀者買過電子材料行的可變電阻就會知道，一般的可變電阻能轉的角度就是約 120 度。各家 SERVO 製造商對於這個角度設定是差很多的，有些甚至只有 80 度(中點左右各 40 度)而已，有些則可以高達 360 度。若是超過許可的運動角度，有可能害 SERVO 燒毀或是卡住，甚至可能因此害控制板燒毀

(太多 SERVO 同時卡住造成瞬間通過控制板的電流過高)，這是必須很注意的一件事情。一般這直接跟賣家詢問就好，若賣家也答不出來，可以自己測試，用上面的程式，一次往前往後增加一點來測試，EX: 原來的 30~150，改成 25~155，讓他跑一次，看能不能順暢跑完，若可以，那就在加一些，若不行，那就是大概的邊界了。但要越加越少的原則來進行，像是測試到 20~160 時，可能一次加 2 就好，下次就測 18~162，因為一般的 SERVO 很少會超過 120~150 度(有些甚至只有 80 度)，後面的增加幅度要慢一點，以免直接卡住，或是發出怪異的高頻嗡嗡聲音，都得反應快一點立刻拔掉電源來處理。

第二個要注意的問題是 **delay(20);** 這行，這有什麼好注意? 這跟 SERVO 的特性有關，一般的便宜 SERVO，標準的 PWM 接受格式是 50Hz/sec，也就是每個波長= 1 秒/ 50 = 20ms 為每一個脈波的週期長度，所以一個改變位置的指令送出後，需要等待至少 20ms 再送會比較好，況且，SERVO 是有運動速度的，也得給他時間確實的移動到指令要求的時間，而不是拼命的送出移動指令，送太快並沒有意義，可能還會干擾 SERVO 運作。

但有些高級的 SERVO 會有 65Hz、120Hz、200Hz、333Hz 甚至更高，以搭配各家廠商的特殊陀螺儀之類的，來做高速反應的動作，像這種頻率特殊的 SERVO，就得修改程式裡面的這些參數來對應他們，不然控制上就不能發揮這些 SERVO 對指令高速反應的能力。

還有些 SERVO 像是 KONDO 出的會有回饋位置功能，可能像是送出 300us 的 PWM，然後會回傳一個段落的 HIGH 訊號，使用 pulseIn 去量取

來計算目前 SERVO 的位置等等，不過這不是這個章節要談的，後面會在應用中去說明並實作這種功能。

四、 PWM 的應用(三)DC 馬達控制

DC 直流馬達因為控制非常簡單，構造也很簡單，價格便宜又容易取得，在生活中簡直無所不在，舉凡電動門、各種玩具、小型電風扇、電動腳踏車等等都可看到它。Edu Cake 要直接控制直流馬達是不可能的，因為隨便一顆很小的直流馬達都至少要數十 mA 的電流，Edu Cake 和所有的單晶片一樣，輸出的訊號電流都很小，不足以驅動 DC 直流馬達，最省錢的解法就是用電晶體實作電流放大電路，常見的有 9012/9013、2N2222/2N2907、或是 TIP120 之類的達靈頓電晶體等等，然後弄清楚他的規格以後，先計算他需要使用的配對電阻和採購防止逆電流的二極體等等的零件，LAY 好電路，可能在麵包板上先實作，再來焊接，如此這般，一陣努力之後馬達才終於會動，中間萬一有焊接或是零件有問題還得作些測試除錯的步驟。以上的動作對於有電子基礎的人來說很容易辦到，但多數讀者根本沒有電子相關背景，實作這些是有難度的，接線也會很複雜。

這裡採用最簡單的方式，使用現成 IC 也可以輕易作到這件事情，直接利用 Edu Cake 的麵包板加上 L293 這顆 IC(電子材料行一顆約賣數十元台幣)，和一顆常見的 DC 馬達(一樣電子材料行有賣，或是隨便壞掉的汽車玩具內都一定會有)，就可以簡單的作出控制兩顆普通玩具馬達的電路，如圖 7，這張圖先只列出馬達的接法，搭配底下的腳位說明，就可以輕易的接好他，一次控制兩顆馬達的正轉、反轉、停止和速度控制。詳細的 L293 電路規格

在這裡

http://pdf.datasheetcatalog.com/datasheets/90/69628_DS.pdf

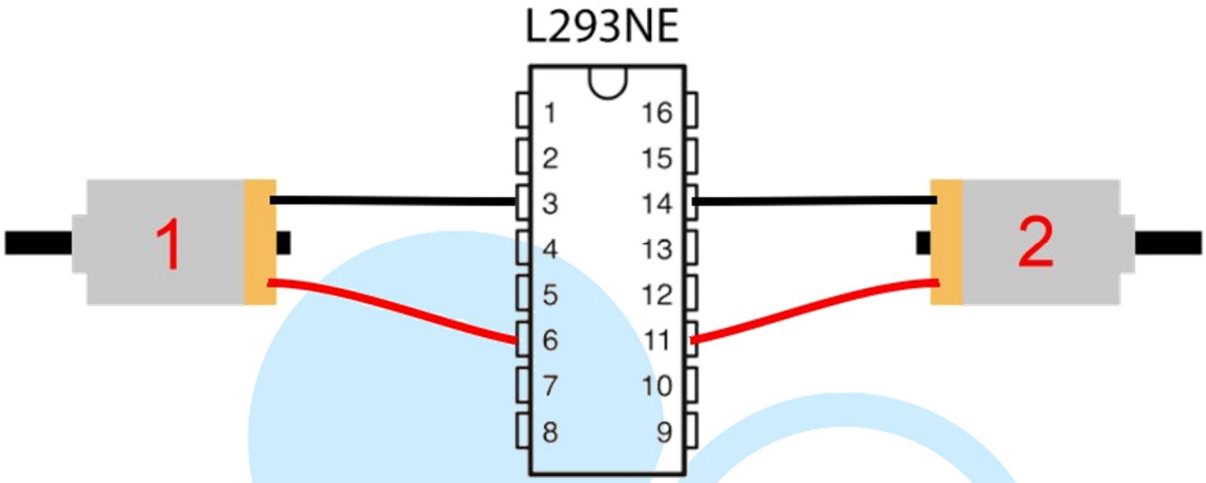

圖 7. L293 的接線圖

在圖 7 中，各腳位說明如下

腳位	功能和接法
4、5、12、13	接到 GND
1、9	分別是馬達 1、2 的速度控制，輸入 PWM 可控制速度，輸入 HIGH 就以最高速運轉
8	馬達的電源，若馬達是 3V 的規格，這裡就直接 5V 電源，他主要支援 4.5 V~36 V.的電源，低於 4.5V 就得利用 PWM 來控制，不然馬達會燒掉。但需注意 L293 的每一顆馬達電流只能到 0.6A，不能接太大顆的馬達，這對普通玩具已經足夠了。
16	VCC，L293 控制電源，給 5V
2、7	控制第一顆馬達的正反轉和停止
10、15	控制第二顆馬達的正反轉和停止

有了腳位定義以後，就可以來實作一個簡單的控制程式，這裏我們想先控制一顆馬達就好，比較容易說明，當然一次兩顆也沒有問題，就是把另外一邊也接上而已。首先，把線路接好如附圖 8，可看到沒幾條線就能接好這個電路，這裡要注意一件事情，馬達的電源，把 L293 腳位 8 這條馬達電源直接接到 EduCake 上面的 5V 是沒關係的，因為普通的玩具小馬達耗電量在沒有負載的時候只有數十 mA，對於 EduCake 從 USB 出來的 5V 電流能提供到 1A 來說是綽綽有餘，但若是要實作大一點的馬達控制，這條線一定要另外接別的獨立電源或是電池，以免害電腦損壞(USB 電流過大是電腦有危險，EduCake 有保護反而不會有事)。

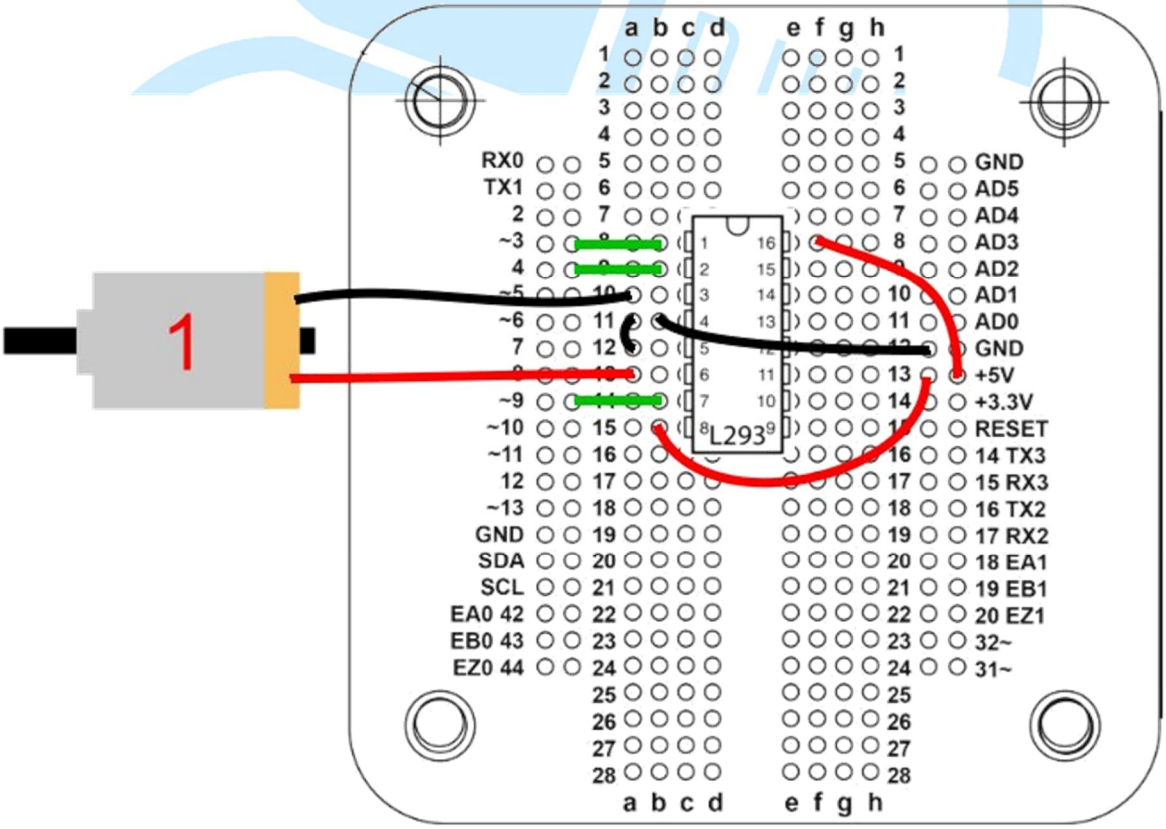

圖 8. 一顆馬達的接線圖

然後撰寫以下的程式，就可以看到馬達會先最高速度左轉一秒，停止，再右轉一秒，停止，然後慢慢的由停止加速到最高速度後停止，不斷重複。

```
int M1=4,M2=9; //  這兩個腳位隨便兩個 digital 就行

int EN=3; //  這個一定要接到有 PWM 輸出的腳位才有用


void setup()
{
   pinMode(M1,OUTPUT);
  pinMode(M2,OUTPUT);
  pinMode(EN,OUTPUT);
}


void loop()
{
   int a;
  digitalWrite(M1,HIGH); //  馬達左轉
  digitalWrite(M2,LOW);
  digitalWrite(EN, HIGH);
  delay(1000);
```

```
digitalWrite(EN, LOW); //  停止
  delay(1000);
  digitalWrite(M1,LOW); //  馬達右轉
  digitalWrite(M2, HIGH);
  digitalWrite(EN, HIGH);

  delay(1000);

  digitalWrite(EN, LOW); //  停止
  delay(1000);

  for(a = 0; a <= 255; a++)
  {
       digitalWrite(M1,HIGH);
     digitalWrite(M2, LOW);
     analogWrite(EN, a); //  使用 PWM 來控制 EN 腳位，就可以控制
速度
     delay(20);
   }
```

```
        digitalWrite(EN, LOW); //  停止
        delay(1000);
    }
```

其中那行 PWM 控制的指令

```
    analogWrite(EN, a); //  使用 PWM 來控制 EN 腳位，就可以控制速度
```

跟之前談 LED 時的有說過的，馬達也和 LED 一樣會有不反應區，電壓從 0 開始升到到某一個數字以後馬達才會開始有動作，每一個馬達都不會一樣，甚至同型號的馬達也會多少有點出入，所以控制迴圈若是從 0 開始跑，會有一段時間馬達都沒反應，讀者要自行去試驗到哪個部份他才會開始動，可能是 0.5V，那對應到 PWM 的數值就是： (0.5V/5V)* 255 = 約是 25 左右，所以迴圈裡面那個 a=0 要改成 a=25，這樣一進迴圈，馬達才會立刻開始慢慢加速；同樣的，因為若是用普通的玩具小馬達，安全運作電壓只能 3V，上圖的電路卻接了 5V，那麼為了安全起見也不能讓迴圈真的跑到 255，而應該改成： (3V/5V) * 255 = 約是 153 左右就得停了。當然，馬達都會有所謂的規定工作範圍，超過一點點沒關係，真的跑到 5V，只要不要持續太久他也不會真的燒掉，何況這也沒負載，但這裡還是該有正確的觀念，所有的電子零件應盡量遵循規格書裡面寫的規範來使用，比較能安全長久的用，超頻超壓使用不是不行，只是那是會縮短零件壽命的行為，盡量要避免。

至於未來可能會想要控制更大型的馬達、廣告中的 DC 變頻冷氣或是載人的汽車(筆者作過載重兩噸的搬運車，用的也是這個程式和電路結構)，電路基本上是一樣，連程式基本原理也會差不多(當然商業型的一定很複雜，要考慮很多事)，只是換掉那個控制 IC 變成功率更大的版本而已，或是改用功率 MOSFET 來作，當然，線也要變粗一點，不然通過的電流太大是會燒斷的。

五、PWM 的應用(四)無刷馬達控制

除了前面講過的東西以外，筆者還做過不少像是遙控飛機、船的控制，這些遙控設備，通常不能用 DC 馬達，因為為了要能瞬間輸出超大力量或是超高轉速，幾萬轉/每分鐘都是基本的，使用普通的 DC 馬達不太合適，且會因為裡面的構造有電刷，連續操勞的情況下磨損的速度會很快，更重要是電刷結構的特性，強大電流通過電刷傳遞到電磁鐵時，會產生很多的電火花，這會嚴重干擾到遙控設備，根本就無法正常操控。所以遙控界用的一定是無刷馬達，不然就是需要作很麻煩的隔離(這根本不會有人採用)。無刷馬達的長相如圖 9，這質感比起前面講的 DC 玩具馬達要好太多了，表面亮亮的那是不銹鋼，有些則是鍍亮鉻，當然也明顯貴，一顆上百元台幣起跳到數萬一顆都有，請注意他有三條線拉出來

圖 9. 無刷馬達的長相

因為要用 EduCake 控制他，所以必須來了解一下相關的規格，通常我們拿到一顆無刷馬達，會先了解他的 KV 值和電流消耗，KV 就是每伏特(v)的轉速(k)。遙控飛機、車、船等等有太多不同的機種型號和載重、速度的需求不同，所以也衍生中出各種不同 KV 值的無刷馬達。而到底要用哪一種無刷馬達，必須看實際的需求而定，以飛機來講，帶的螺旋槳 size 不同，配的 KV 也不一樣，同樣大小的飛機，小螺旋槳搭高 KV 值(扭力相對較小)的無刷馬達；大螺旋槳搭低 KV 值大電流(扭力大)的馬達，才能輸出足夠的推重比，不然飛機飛不動。其實 KV 只是個參考，真正還是要算輸出功率(功率=電壓 * 電流)和機型而定，同樣一架飛機，搭大/小螺旋槳的表現是會明顯不同的。還有一點要注意是，因為通常無刷通過的電流都很大，散熱必須注意，筆者朋友很喜歡改裝超大功率來玩，常常玩遙控玩到一半電線燒斷或是起火都是很習以為常的事情。

各種遙控設備的無刷馬達通常都會透過一種叫作電子變速器，簡稱電變(有些人叫他作電調)的電路來控制無刷馬達，這其實是一種穩壓電路加上馬達控制電路所夠成的一種整合產品，這裏我們不打算實作這種電路(真要作也不難，只是這不是這個章節的主題)，直接買現成的來用就好(網拍入門款一顆只要一兩百就能買到)，如圖 10 上方那個就是

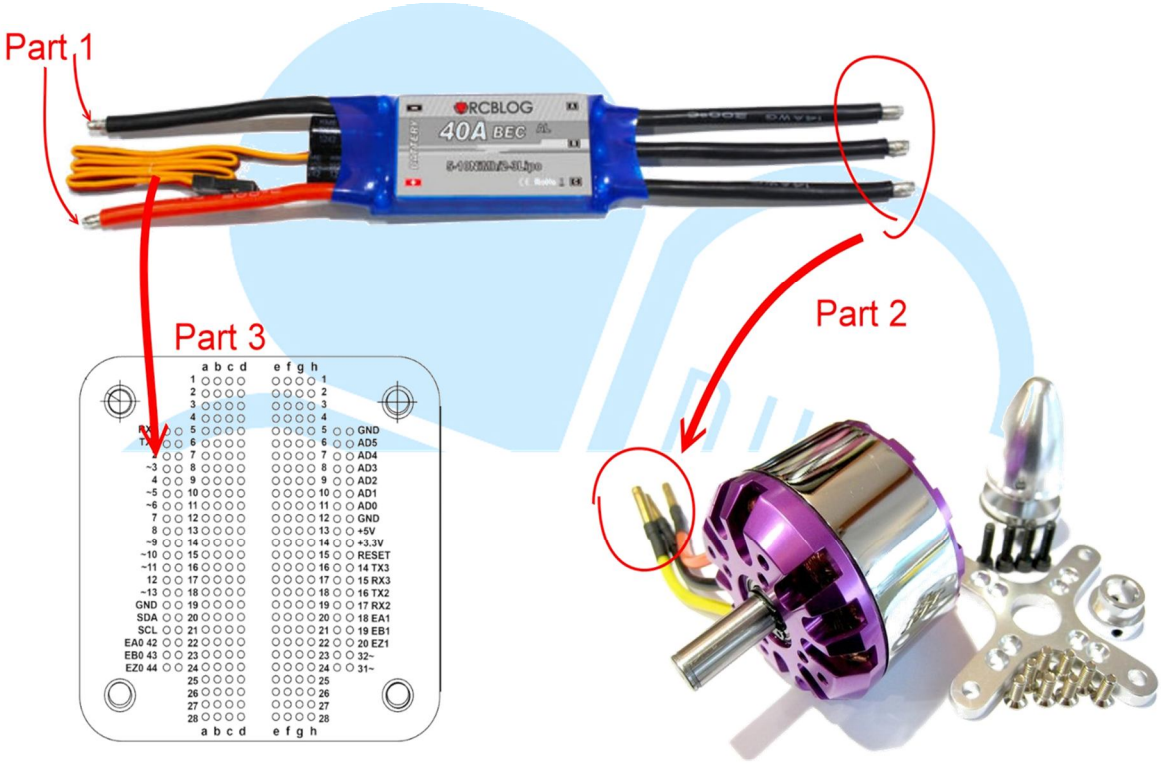

圖 10. 無刷馬達的長相

這個電路分三個部分，上方的電變、右方的無刷馬達和左方的 EduCake 所構成，圖中還標注了 Part1~3。

Part1 就是要給馬達和電變的電源，正負千萬不能接錯(紅色接正，黑色接負應該是常識)，需依照上面標示的規定給，一般都是 7.4V / 11.2V 之類的，但要注意電流要給足，例如這馬達要求 120A 電流，那電池就要大

顆一點的才能輸出足夠的電流，至於到底是要多大顆，請自行到模型店去詢問，這裡先單純談控制；電線也要夠粗，這種電路最常見的狀況就是電線燒斷，電流很大的情況下，粗的電線相對電阻比較低，比較能負荷大電流通過。所以認真說，連電線和電池間的接頭都是必須講究的，接頭也有分電流大小，大電流用的接頭很粗壯一個，可能還講究要鍍金，小電流的接頭基本上就是隨便選了。

Part2 就是電變出來那三條很粗的線要和無刷馬達那三條線連接，怎麼接? 就是按順序接起來就對了，那是由左而右還是由右而左呢?!請注意看電變出來是三條黑線，只有標示 ABC，而不是標示正負，這是因為馬達裡面是線圈，不管如何接，通正負進去會產生磁性，但通負正也會產生磁性，馬達都會轉，唯一的差別是方向不同。筆者在使用時，會先用夾子去夾住他們，去測試旋轉方向，若不是自己想要的，那就把左右兩條線對調接，中間不變，再測一次，應該方向就要對了。這對處理四軸或是多軸的飛機特別要緊，因為像是四軸飛行器的螺旋槳，為了要抵消轉動慣量，螺旋槳都是兩兩成對的，且旋轉方向相反，一定要能確實的把馬達的旋轉方向調整好，才能讓飛機飛上天。

Part3 就是電變左方中間那條線，仔細看會發現那條線和 SERVO 的線一模一樣，事實上，他的訊號規格也和 SERVO 是一模一樣的，所以接法也相同，請參考前面談過的 SERVO 接法，把那條線接到 EduCake 上面，這樣就完成整個電路的連接。且需要詳細的再檢查一次電路，確定所有的接線都是正確的，之前談過那些電路，都是電流很小，隨便亂接也不會怎

樣，但控制無刷馬達的電路因為電流很大，一個不小心燒毀或是失火的情形都會有，一定要不厭其煩，再三檢查確認沒有問題才行。

接好電路後，接下來就要寫程式來控制他了，這部分的控制程式跟前面控制 SERVO 很像，只是因為電變的行為和 SEVRO 還是不太一樣，程式還是要注意。首先，電變需要設定才能用，像飛機來說，螺旋槳主要有兩種狀況：停止、一般速度到全速的變化，這在遙控器上的設定順序通常為：

1. 先打開遙控器並把油門推到最大~再接通接收機電源
2. 這時電變應會有嗶聲，立刻搖控器把油門搖桿推到最低，這個動作是要讓電變抓油門的距離
3. 此時電變會再發出嗶聲，設定ＯＫ
4. 有些電變可以儲存設定，但有些不會有，就變成每次啟動都要重覆這個動作

常見無刷馬達電變設定法約是這樣，但各家的還是會有功能和流程上的不同，需詳閱說明書後才進行比較好，且最好馬達先不要接通，之前有談過馬達電流很強轉速很快，這樣做會比較安全。確認設定好了以後，就可以開始來看看效果，程式碼如下

```
#include <Servo.h>
Servo myservo; //  建立 Servo 物件
void setup()
```

```
{
  myservo.attach(3);   // 連接 digital 腳位 3
}
void loop()
{
 int a = analogRead(0); // 類比腳位 0 的地方使用一顆 VR 來控制馬達的快慢
 a=map(a,0,1024,1000,2100); // 先把讀取到的數值轉換為 PWM 對應角度
   myservo. writeMicroseconds (a);   // 這次使用 writeMicroseconds 作更精確控制
   delay(20);
}
```

其中的 a=map(a,0,1024,1000,2100); 可以讀取到的可變電阻值對應到 PWM 脈波訊號的 1000~2100us 的範圍，然後使用 writeMicroseconds() 寫入，這個指令比較精準，也是比較直覺的 PWM 控制指令，一般筆者都採用這個比較多，write () 指令比較少用，除了不夠精準外，重點是因為 SERVO 種類太多，每一種的角度對應 PWM 都不一樣，這指令裡面傳入的角度參數其實根本沒有意義，還不如直接用 writeMicroseconds()。程式執行後，轉動 VR，便可看到無刷馬達由停止

到變快，一直到最快的速度，發出扎實的加速聲音，真是令人心動呢! 有了這個控制功能，讀者就可以開始把遙控器丟掉了，未來我們會拆開遙控車、飛機，利用電腦和 EduCake 直接來控制這些遙控玩具，作更多的應用。