PWM 的使用

一、 什么是 PWM?

谈过基本讯号控制以后，我们就可以来些不同的变化，这里要介绍一种常用的讯号规格"PWM"，PWM 全名叫做 **Pulse Width Modulation**，脉冲宽度调变，这是一种利用前面谈过的数字(Digital)讯号来"仿真"模拟(Analog)讯号的方式，最常见的一些应用是电子类零件的电压调整，例如:马达快慢控制、灯光的明暗控制、屏幕亮度控制、喇叭的大小声/声音频率高低控制等等。实际的作法是利用控制数字讯号的脚位，用程序高速切换脚位的 HIGH / LOW 的状态，通常会搭配系统 TIMER，来输出一连串的方波，每个方波的长相，如图 1

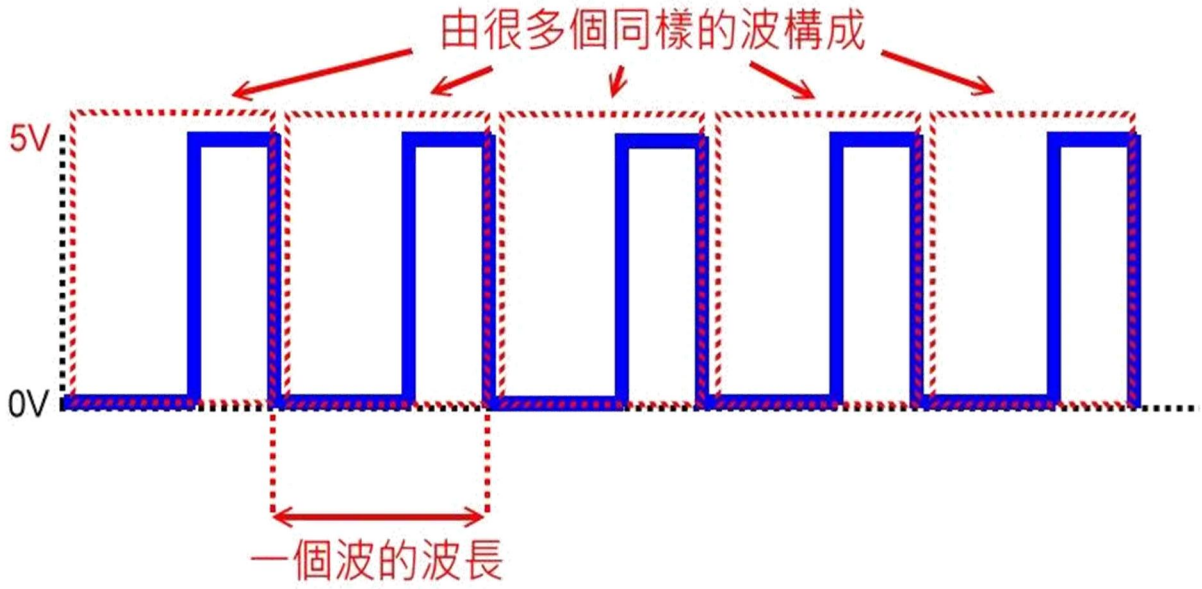

图 1. PWM 的长相

每一个波型所占的时间长度，又称为周期，频率(Hz)就是指单位时间内出现多少个这样的波型，EX: 一秒内出现 50 个这种样子的波型，我们就说频率就是 60Hz/sec。大量重复的产生这种波，并且透过调整他的周期、频率和波长，就可以模拟出各种很像模拟讯号的输出，来实作各种应用，如图 2

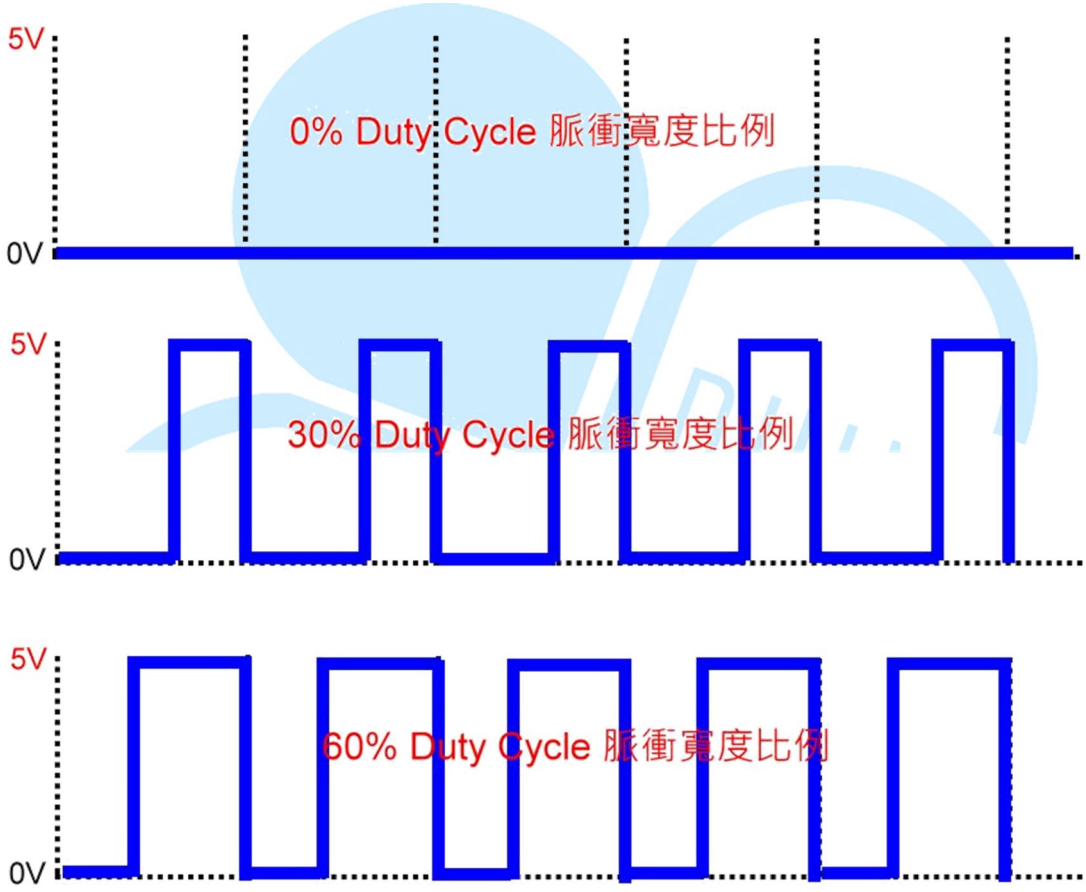

图 2. 各种不同占空比的 PWM

要实作普通的 PWM 波型，并不需要写很复杂的程序(后面会谈到复杂或是高频率的波型实作)，例如想作出图 2 的第一种波型，只要使用这个指令 analogWrite(脚位,0); 就可以轻易办到；第二种波型，只要使用这个指令 analogWrite(脚位,78); 就可以轻易办到；第三种波型，则使用这个指令 analogWrite(脚位,153); 即可办到。其中，指令里面那个数字只是个概算，

PWM 控制指令的参数范围介于 0~255，所以若是要 30%为 HIGH，那就是 255 * 30% =大约是 78 左右。

关于 analogWrite 这个指令，是给特定的 digital 数字脚位使用的，会以数字的方式输出 PWM 讯号，指令格式:

```
analogWrite(腳位, 數值)
```

其中的数值必须介于 0~255 ，对应到 0~5V，所以分辨率为 5V / 255 = 0.0196V

像上面使用了 analogWrite(脚位,160); 其中的 160 * 0.0196V = 代表会输出 3.13V 的仿真模拟电压，频率则是 1KHz。而脚位的设定则必须是 CAKE 板子上有标示"~" 符号的脚位才能使用，如图 3

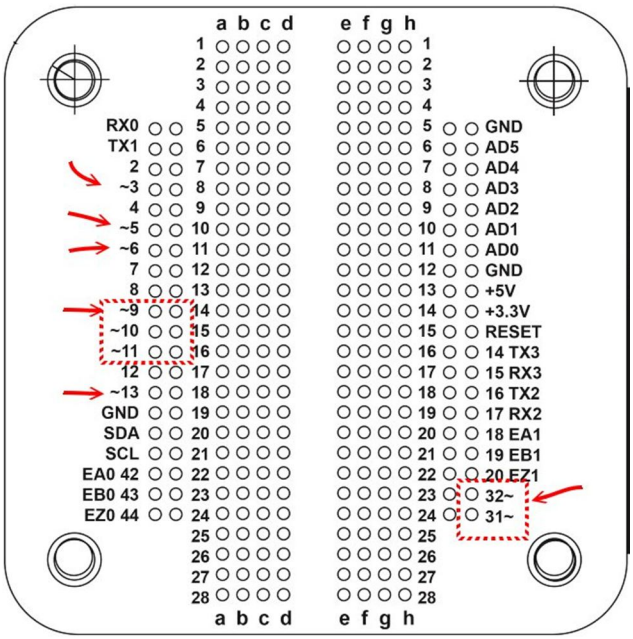

图 3. 86Duino EduCake 上的 PWM 脚位分布

更重要是，因为这其实还是数字讯号，还是由一堆不同长短的 HIGH/LOW 状态构成，对于储存、传输、抗干扰的能力，都是能有很好的表现也能很轻易办到，也因此用途就日渐的广泛。最常见又简单的例子就是控制 LED 灯的亮灭，如以下的片段程序代码:

```
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
```

以上这段程序代码只能控制 3 号脚位的 LED 灯亮一秒后熄灭，看起来就很单纯，是很直觉的数字控制方式，但他只能这样单纯的亮暗切换，看久会很单调。当我们以模拟的想法来做这件事情的时候，可以让他变得很有趣，PWM 就是一种很简单可以实现这种想法的功能，如以下的程序代码:

```
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);}
```

执行效果会变成，LED 灯会慢慢的由暗变亮，到最亮，然后再慢慢的变暗，到熄灭。重复这个动作并且若做些小微调，还可以让灯看起来好像有

在呼吸的效果，笔者在睡房里放了一个这样的灯，并且打开开关以后就可以慢慢的变亮变暗，并且整个动作的周期越来越长，约半小时后完全熄灭不再有动作，根据实验，搭配轻柔且慢慢变小声的音乐骗小孩子睡着是很有用的哩，当然程序代码就比这里列出的要长一些了。而同样的程序代码不要单纯的在循环里面执行 a++，而是改成执行使用小范围随机数去加减他，就可以模拟出类似烛光摇曳的效果，呼吸灯瞬间立刻变成人工烛光，很有趣喔；在额外加上声音传感器，立刻就变成可以用嘴去 "吹熄灭"的人工烛光了，也可以用拍手的声音来达成同样目的，这可是摆在餐厅桌上非常实用的应用，读者可以再想想还可以有哪些变化。

当然，以上这段简易的程序代码其实有些问题在其中，LED 灯要能亮起来需要一定的电压才行，EX: 2.5V，PWM 要能仿真出这样的电压可能需要里面那个参数设定到 125 以上才行，这样程序中的循环，在 0~124 左右的时候，灯都会是全灭的状态，看不太出来变化；同样的，LED 灯可能在 4V 的时候就非常亮了，4~5V 这个区间都会是非常亮的样子，根本看不出有没有比较亮的差别，所以实际要做这功能，须要先测试出所选用的 LED 的最暗到最亮的范围，再去修改程序里面的循环值会有比较好的效果。

以上谈的是基本的 PWM 功能，86Duino EduCake 里面提供的基本 PWM 讯号，频率默认是 1000Hz/ sec，也就是每个波的长度为: 1sec / 1000 = 0.001 秒，这种速度已经足以应付很多种日常需求了，当然也有很多别的方式可以有更精确的 PWM 输出来使用。请注意因为脚位的输出电流非常小，若要推动大型的电力需求，EX: 马达、电风扇、甚至冷气(搞不好读者想作变频 DC 冷气)，都是没有问题，就是要想办法把电流放大来用就好，后面我们就来看看 PWM 的一些应用。

二、 PWM 的应用(一)呼吸灯

这个应用电路图很简单，只是接一颗 LED，如图 4，这部分笔者是直接把电阻和和 LED 焊接在一起，使用上比较简单，只要插两条线就好，读者也可以分开插到面包板上面

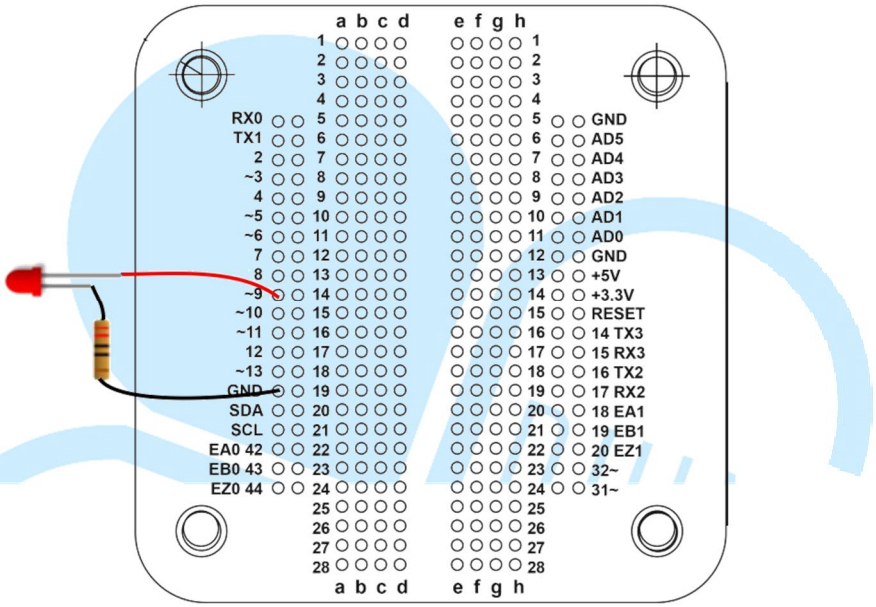

图 4. 接线图

直接来看看程序

```
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) 里面那两个数字就是之前讲过的，LED 灯从完全熄灭到稍微开始发亮的位置(我的 LED 是 50)，另一个数字就是最亮的位置，我的是 205，读者使用的要自己测试，以便确实的去改变这两个数字，不然会变成变暗以后，要等一阵子才又会变亮；变最亮以后，要等好一阵子才会开始变暗，这中间的效果就看个人如何去调整了。

那要如何才能作到前面谈过的烛光效果呢?! 首先，想想看烛光的样子

1. 烛光一定是会发亮。 << 废话!!

2. 烛光会摇<< 这是风吹空气扰动的效果，LED 想要有这种效果只好用马达搭配机构来摇动他模拟了，这里因为我们的装备只有 LED 就先跳过，读者可结合后面谈的马达控制来做到，不过更简单是利用透明玻璃纸条+风吹，可以作出类似效果，远远看还不错，有些 7-11 的关东煮上方会有类似这种装置可以去参考看看。

3. 烛光会因为被风吹而导致亮度的变暗一点或是变亮一点，但很不一定。

其实，上面在列出这些东西的时候，列了第一个条件笔者自己就笑出来，这看起来就很白痴，但，这是必要的，要解决任何问题的时候，必须先把问题拆解成很简单的一个一个小问题，再拆解成很接近 IDE 程序提供的指令/语法片段的小问题，这样就可以一个一个转换成程序，像拼积木一样把程序组合起来答案就出来了，这对未来解决大型项目是很重要的，以下来看看这段程序(电路不需要任何修改)

```
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);
}
```

里面的那些数字和参数好好的依照需求来微调，出来的效果是很令人满意的，尤其还可以搭配 RGB3 色 LED 灯，更可作出很多令人意想不到的变化。读者可以再仔细的想想，生活中有哪些地方是会使用到灯光的效果呢?

三、 PWM 的应用(二)SERVO 马达控制

各种遥控用的 SERVO 或是任何需要使用 PWM 控制的马达都可以使用 EduCake 的 PWM 讯号来作控制，这里就用一颗最容易取得的伺服马达来作实验，如图 5 的 SG90 这种常用的入门级遥控伺服马达。

图 5. 常见 SERVO 的接头长相

这种遥控用的小 SERVO 马达一颗只要几十元台币，拍卖上到处都是，专门用来控制遥控飞机或是小汽车用的。请注意到他的接脚是橘(讯号线，有些不同厂牌的马达可能是白色)、红(接正电源)、棕(GND，有些不同厂牌的马达可能是黑色)三种颜色，其中的橘色就是直接接到 CAKE 的 PWM 脚位就能控制他。这颗马达是质量很差，很不 OK 的入门级玩具，但拿来做实验非常好，至少玩坏不会心痛。规格如下

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) ，这是说 SERVO 在 4.8V 电压时，转 60 度最快要 0.1 秒

5. Dead band width 10us，个人喜欢翻译成"无反应间隔"，就是上次对 SERVO 送出的 PWM 讯号和这次的讯号相差若小于 10us，SERVO 有很大机率是完全不会有反应的，或是讲成 SERVO 的控制精确度也行，这主要是和 SERVO 里面的 VR 分辨率有关

6. 运动角度约 150 度 << 这很重要，一定要查清楚使用的马达角度，不然送过大的 PWM 波，可能会让马达卡住导致烧毁

类似的 SERVO 像是（http://www.roboard.com/Files/RS-0263/RoBoard_RS-0263.pdf）这种马达，精确度就明显好很多，无反应间隔也明显小，力量又大，一般入门后要玩比较精细的玩家就会采用，整个作出来的作品表现会明显提升，毕竟是一分钱一分货。 电路接法如图 6，这只是纯粹简单的测试，且这颗马达很不耗电才这样接。请注意其实伺服马达是不能这样接的，马达的电流相对于控制板来说比较大，在有负载的情况下容易烧毁控制板(EduCake 内部有过流保护不用担心，其他种类的单晶就得注意了)，一般最好是把马达的正极接到独立的电源，然后该独立电源的 GND 和控制板的 GND 接在一起，甚至可以再加一个二极管防止逆电流，这样就可以防止马达电流过强导致控制板有问题，或是控制板电流不够导致马达不会动。

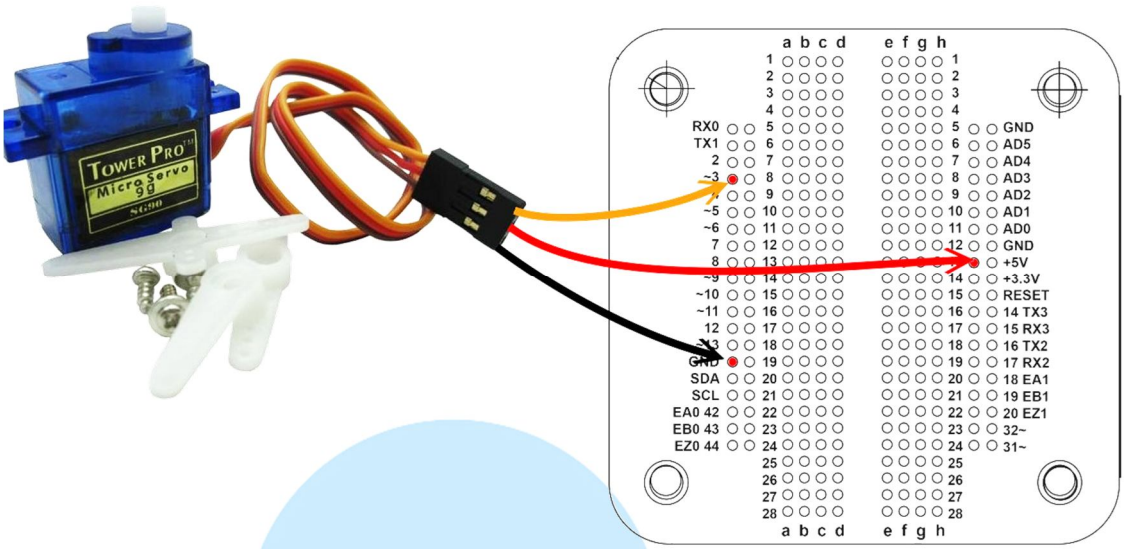

图 6. SERVO 的接线图

搭配底下这个程序就能让马达左右扫来扫去，这也是笔者最常用来测试 SERVO 耐用度的方式，使用时我还会把 SERVO 摆臂上绑重物，测试他的实际出力、耐用度、消耗电流、温度的变化等等参数，把马达彻底操劳到坏掉为止，以便未来使用的时候能掌握设计出来的机器人的功能。且买 SERVO 除了这些注意事项，还要看 SERVO 的 Dead band width 的区域大小(越小越好)，还要看速度、尺寸、长相(牵涉机构好不好设计和固定等等)、齿隙等等的一堆问题，并没有那么简单，还是需要一些使用经验来挑选适合的 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);
  }
}
```

这个程序里面，除了批注以外还有几个需要注意的地方，首先第一个循环里面，从 30 跑到 150 度，有读者会问为何不是 0~180，或是 0~360 度呢?!这是因为 SERVO 里面的定位是透过一颗 VR 读取数值来定位，若是读者买过电子材料行的可变电阻就会知道，一般的可变电阻能转的角度就是约 120 度。各家 SERVO 制造商对于这个角度设定是差很多的，有些甚至只有 80 度(中点左右各 40 度)而已，有些则可以高达 360 度。若是超过许可的运动角度，有可能害 SERVO 烧毁或是卡住，甚至可能因此害控制板烧毁

(太多 SERVO 同时卡住造成瞬间通过控制板的电流过高)，这是必须很注意的一件事情。一般这直接跟卖家询问就好，若卖家也答不出来，可以自己测试，用上面的程序，一次往前往后增加一点来测试，EX: 原来的 30~150，改成 25~155，让他跑一次，看能不能顺畅跑完，若可以，那就在加一些，若不行，那就是大概的边界了。但要越加越少的原则来进行，像是测试到 20~160 时，可能一次加 2 就好，下次就测 18~162，因为一般的 SERVO 很少会超过 120~150 度(有些甚至只有 80 度)，后面的增加幅度要慢一点，以免直接卡住，或是发出怪异的高频嗡嗡声音，都得反应快一点立刻拔掉电源来处理。

第二个要注意的问题是 delay(20); 这行，这有什么好注意？这跟 SERVO 的特性有关，一般的便宜 SERVO，标准的 PWM 接受格式是 50Hz/sec，也就是每个波长= 1 秒/ 50 = 20ms 为每一个脉波的周期长度，所以一个改变位置的指令送出后，需要等待至少 20ms 再送会比较好，况且，SERVO 是有运动速度的，也得给他时间确实的移动到指令要求的时间，而不是拼命的送出移动指令，送太快并没有意义，可能还会干扰 SERVO 运作。

但有些高级的 SERVO 会有 65Hz、120Hz、200Hz、333Hz 甚至更高，以搭配各家厂商的特殊陀螺仪之类的，来做高速反应的动作，像这种频率特殊的 SERVO，就得修改程序里面的这些参数来对应他们，不然控制上就不能发挥这些 SERVO 对指令高速反应的能力。

还有些 SERVO 像是 KONDO 出的会有回馈位置功能，可能像是送出 300us 的 PWM，然后会回传一个段落的 HIGH 讯号，使用 pulseIn 去量取来计算目前 SERVO 的位置等等，不过这不是这个章节要谈的，后面会在应用中去说明并实作这种功能。

四、 PWM 的应用(三)DC 马达控制

DC 直流马达因为控制非常简单，构造也很简单，价格便宜又容易取得，在生活中简直无所不在，举凡电动门、各种玩具、小型电风扇、电动脚踏车等等都可看到它。Edu Cake 要直接控制直流马达是不可能的，因为随便一颗很小的直流马达都至少要数十 mA 的电流，Edu Cake 和所有的单芯片一样，输出的讯号电流都很小，不足以驱动 DC 直流马达，最省钱的解法就是用晶体管实作电流放大电路，常见的有 9012/9013、2N2222/2N2907、或是 TIP120 之类的达灵顿晶体管等等，然后弄清楚他的规格以后，先计算他需要使用的配对电阻和采购防止逆电流的二极管等等的零件，LAY 好电路，可能在面包板上先实作，再来焊接，如此这般，一阵努力之后马达才终于会动，中间万一有焊接或是零件有问题还得作些测试除错的步骤。以上的动作对于有电子基础的人来说很容易办到，但多数读者根本没有电子相关背景，实作这些是有难度的，接线也会很复杂。

这里采用最简单的方式，使用现成 IC 也可以轻易作到这件事情，直接利用 Edu Cake 的面包板加上 L293 这颗 IC(电子材料行一颗约卖数十元台币)，和一颗常见的 DC 马达(一样电子材料行有卖，或是随便坏掉的汽车玩具内都一定会有)，就可以简单的作出控制两颗普通玩具马达的电路，如图 7，这张图先只列出马达的接法，搭配底下的脚位说明，就可以轻易的接好他，一次控制两颗马达的正转、反转、停止和速度控制。详细的 L293 电路规格在这里

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

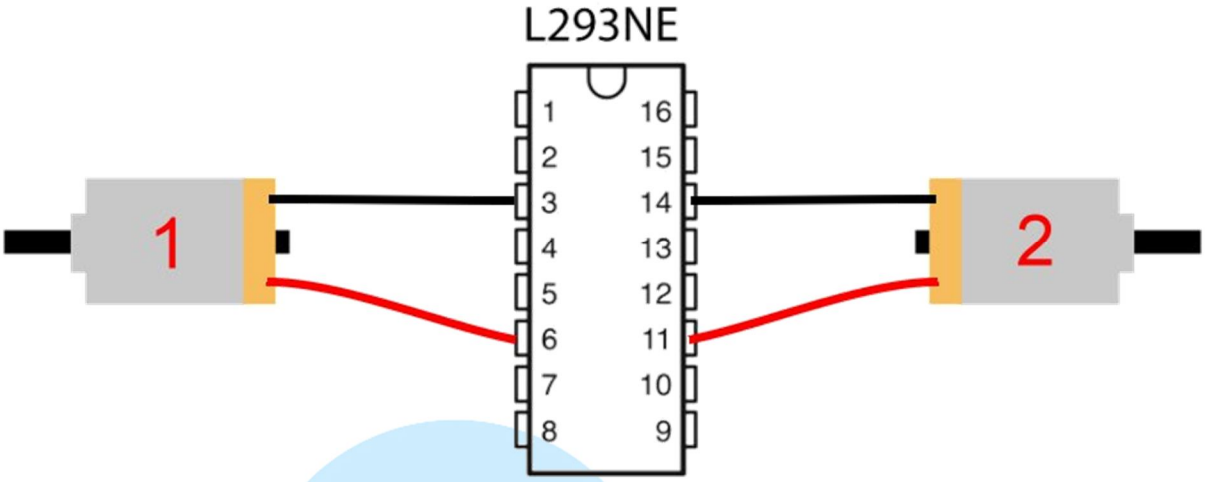

图 7. L293 的接线图

在图 7 中，各脚位说明如下

脚位	功能和接法
4、5、12、13	接到 GND
1、9	分别是马达 1、2 的速度控制，输入 PWM 可控制速度，输入 HIGH 就以最高速运转
8	马达的电源，若马达是 3V 的规格，这里就直接 5V 电源，他主要支持 4.5 V~36 V.的电源，低于 4.5V 就得利用 PWM 来控制，不然马达会烧掉。但需注意 L293 的每一颗马达电流只能到 0.6A，不能接太大颗的马达，这对普通玩具已经足够了。
16	VCC，L293 控制电源，给 5V
2、7	控制第一颗马达的正反转和停止
10、15	控制第二颗马达的正反转和停止

有了脚位定义以后，就可以来实作一个简单的控制程序，这里我们想先控制一颗马达就好，比较容易说明，当然一次两颗也没有问题，就是把另外一边也接上而已。首先，把线路接好如附图 8，可看到没几条线就能接好

这个电路，这里要注意一件事情，马达的电源，把 L293 脚位 8 这条马达电源直接接到 EduCake 上面的 5V 是没关系的，因为普通的玩具小马达耗电量在没有负载的时候只有数十 mA，对于 EduCake 从 USB 出来的 5V 电流能提供到 1A 来说是绰绰有余，但若是要实作大一点的马达控制，这条线一定要另外接别的独立电源或是电池，以免害计算机损坏(USB 电流过大是计算机有危险，EduCake 有保护反而不会有事)。

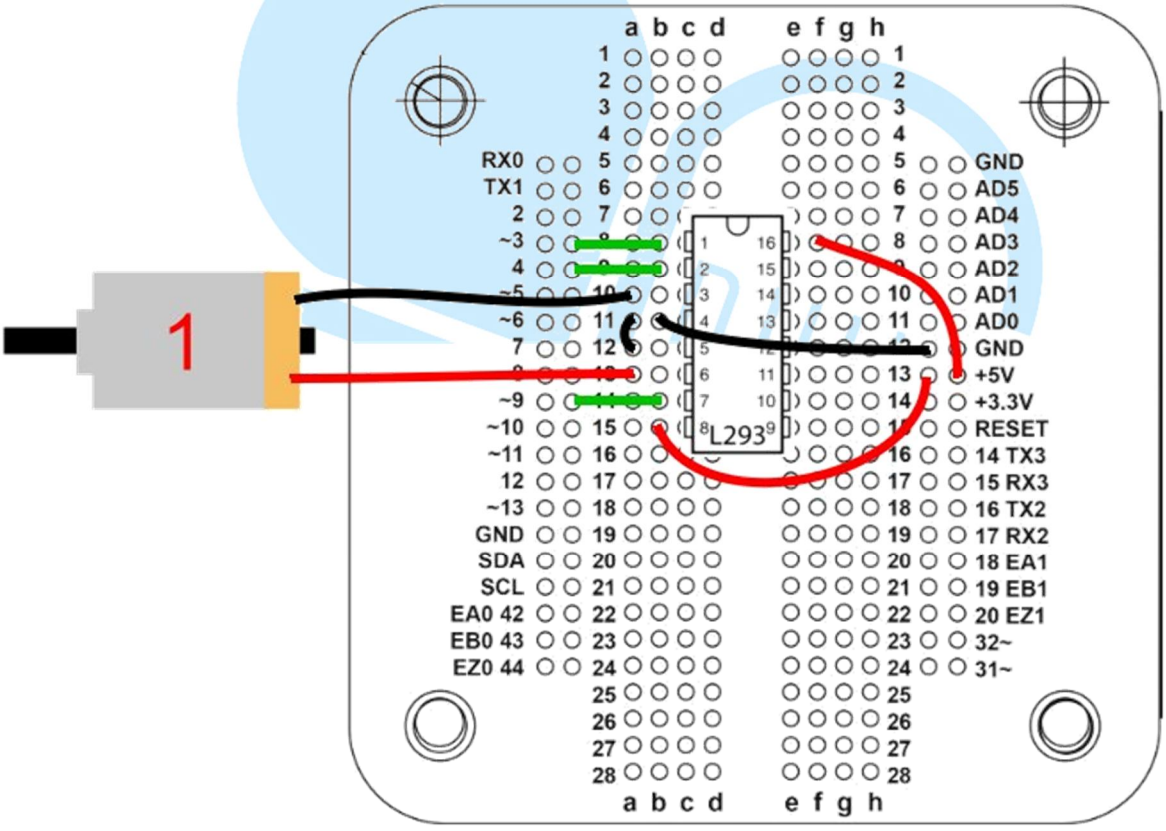

图 8. 一颗马达的接线图

然后撰写以下的程序，就可以看到马达会先最高速度左转一秒，停止，再右转一秒，停止，然后慢慢的由停止加速到最高速度后停止，不断重复。

```
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
```

跟之前谈 LED 时的有说过的，马达也和 LED 一样会有不反应区，电压从 0 开始升到到某一个数字以后马达才会开始有动作，每一个马达都不会一样，甚至同型号的马达也会多少有点出入，所以控制循环若是从 0 开始跑，会有一段时间马达都没反应，读者要自行去试验到哪个部份他才会开始动，可能是 0.5V，那对应到 PWM 的数值就是： (0.5V/5V)* 255 = 约是 25 左右，所以循环里面那个 a=0 要改成 a=25，这样一进循环，马达才会立刻开始慢慢加速；同样的，因为若是用普通的玩具小马达，安全运作电压只能 3V，上图的电路却接了 5V，那么为了安全起见也不能让循环真的跑到 255，而应该改成： (3V/5V) * 255 = 约是 153 左右就得停了。当然，马达都会有所谓的规定工作范围，超过一点点没关系，真的跑到 5V，只要不要持续太久他也不会真的烧掉，何况这也没负载，但这里还是该有正确的观念，所有的电子零件应尽量遵循规格书里面写的规范来使用，比较能安全长久的用，超频超压使用不是不行，只是那是会缩短零件寿命的行为，尽量要避免。

至于未来可能会想要控制更大型的马达、广告中的 DC 变频冷气或是载人的汽车(笔者作过载重两吨的搬运车，用的也是这个程序和电路结构)，电路基本上是一样，连程序基本原理也会差不多(当然商业型的一定很复杂，要考虑很多事)，只是换掉那个控制 IC 变成功率更大的版本而已，或是改用功率 MOSFET 来作，当然，线也要变粗一点，不然通过的电流太大是会烧断的。

五、PWM 的应用(四)无刷马达控制

除了前面讲过的东西以外，笔者还做过不少像是遥控飞机、船的控制，这些遥控设备，通常不能用 DC 马达，因为为了要能瞬间输出超大力量或是超高转速，几万转/每分钟都是基本的，使用普通的 DC 马达不太合适，且会因为里面的构造有电刷，连续操劳的情况下磨损的速度会很快，更重要是电刷结构的特性，强大电流通过电刷传递到电磁铁时，会产生很多的电火花，这会严重干扰到遥控设备，根本就无法正常操控。所以遥控界用的一定是无刷马达，不然就是需要作很麻烦的隔离(这根本不会有人采用)。无刷马达的长相如图 9，这质感比起前面讲的 DC 玩具马达要好太多了，表面亮亮的那是不锈钢，有些则是镀亮铬，当然也明显贵，一颗上百元台币起跳到数万一颗都有，请注意他有三条线拉出来

图 9. 无刷马达的长相

因为要用 EduCake 控制他，所以必须来了解一下相关的规格，通常我们拿到一颗无刷马达，会先了解他的 KV 值和电流消耗，KV 就是每伏特(v)的转速(k)。遥控飞机、车、船等等有太多不同的机种型号和载重、速度的需求不同，所以也衍生中出各种不同 KV 值的无刷马达。而到底要用哪一种无刷马达，必须看实际的需求而定，以飞机来讲，带的螺旋桨 size 不同，配的 KV 也不一样，同样大小的飞机，小螺旋桨搭高 KV 值(扭力相对较小)的无刷马达；大螺旋桨搭低 KV 值大电流(扭力大)的马达，才能输出足够的推重比，不然飞机飞不动。其实 KV 只是个参考，真正还是要算输出功率(功率=电压 * 电流)和机型而定，同样一架飞机，搭大/小螺旋桨的表现是会明显不同的。还有一点要注意是，因为通常无刷通过的电流都很大，散热必须注意，笔者朋友很喜欢改装超大功率来玩，常常玩遥控玩到一半电线烧断或是起火都是很习以为常的事情。

各种遥控设备的无刷马达通常都会透过一种叫作电子变速器，简称电变(有些人叫他作电调)的电路来控制无刷马达，这其实是一种稳压电路加上

马达控制电路所够成的一种整合产品，这里我们不打算实作这种电路(真要作也不难，只是这不是这个章节的主题)，直接买现成的来用就好(网拍入门款一颗只要一两百就能买到)，如图 10 上方那个就是

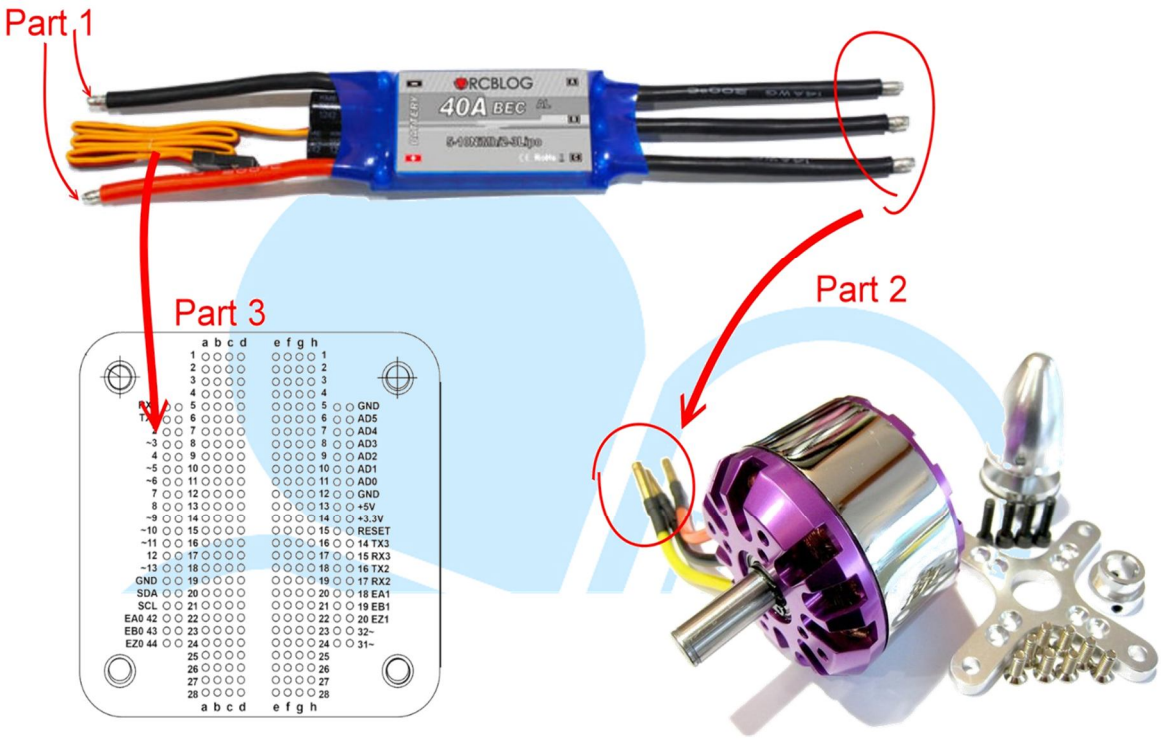

图 10. 无刷马达的长相

这个电路分三个部分，上方的电变、右方的无刷马达和左方的 EduCake 所构成，图中还标注了 Part1~3。

Part1 就是要给马达和电变的电源，正负千万不能接错(红色接正，黑色接负应该是常识)，需依照上面标示的规定给，一般都是 7.4V / 11.2V 之类的，但要注意电流要给足，例如这马达要求 120A 电流，那电池就要大颗一点的才能输出足够的电流，至于到底是要多大颗，请自行到模型店去询问，这里先单纯谈控制；电线也要够粗，这种电路最常见的状况就是电线烧断，电流很大的情况下，粗的电线相对电阻比较低，比较能负荷大电流通过。所以认真说，连电线和电池间的接头都是必须讲究的，接头也有

分电流大小，大电流用的接头很粗壮一个，可能还讲究要镀金，小电流的接头基本上就是随便选了。

Part2 就是电变出来那三条很粗的线要和无刷马达那三条线连接，怎么接？就是按顺序接起来就对了，那是由左而右还是由右而左呢?!请注意看电变出来是三条黑线，只有标示 ABC，而不是标示正负，这是因为马达里面是线圈，不管如何接，通正负进去会产生磁性，但通负正也会产生磁性，马达都会转，唯一的差别是方向不同。笔者在使用时，会先用夹子去夹住他们，去测试旋转方向，若不是自己想要的，那就把左右两条线对调接，中间不变，再测一次，应该方向就要对了。这对处理四轴或是多轴的飞机特别要紧，因为像是四轴飞行器的螺旋桨，为了要抵消转动惯量，螺旋桨都是两两成对的，且旋转方向相反，一定要能确实的把马达的旋转方向调整好，才能让飞机飞上天。

Part3 就是电变左方中间那条线，仔细看会发现那条线和 SERVO 的线一模一样，事实上，他的讯号规格也和 SERVO 是一模一样的，所以接法也相同，请参考前面谈过的 SERVO 接法，把那条线接到 EduCake 上面，这样就完成整个电路的连接。且需要详细的再检查一次电路，确定所有的接线都是正确的，之前谈过那些电路，都是电流很小，随便乱接也不会怎样，但控制无刷马达的电路因为电流很大，一个不小心烧毁或是失火的情形都会有，一定要不厌其烦，再三检查确认没有问题才行。

接好电路后，接下来就要写程序来控制他了，这部分的控制程序跟前面控制 SERVO 很像，只是因为电变的行为和 SEVRO 还是不太一样，程序还是要注意。首先，电变需要设定才能用，像飞机来说，螺旋桨主要有两种状况：停止、一般速度到全速的变化，这在遥控器上的设定顺序通常为：

1. 先打开遥控器并把油门推到最大~再接通接收机电源

2. 这时电变应会有哔声，立刻摇控器把油门游戏杆推到最低，这个动作是要让电变抓油门的距离

3. 此时电变会再发出哔声，设定ＯＫ

4. 有些电变可以储存设定，但有些不会有，就变成每次启动都要重复这个动作

常见无刷马达电变设定法约是这样，但各家的还是会有功能和流程上的不同，需详阅说明书后才进行比较好，且最好马达先不要接通，之前有谈过马达电流很强转速很快，这样做会比较安全。确认设定好了以后，就可以开始来看看效果，程序代码如下

```
#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); 可以读取到的可变电阻值对应到 PWM 脉波讯号的 1000~2100us 的范围，然后使用 writeMicroseconds() 写入，这个指令比较精准，也是比较直觉的 PWM 控制指令，一般笔者都采用这个比较多，write () 指令比较少用，除了不够精准外，重点是因为 SERVO 种类太多，每一种的角度对应 PWM 都不一样，这指令里面传入的角度参数其实根本没有意义，还不如直接用 writeMicroseconds()。程序执行后，转动 VR，便可看到无刷马达由停止

到变快，一直到最快的速度，发出扎实的加速声音，真是令人心动呢！有了这个控制功能，读者就可以开始把遥控器丢掉了，未来我们会拆开遥控车、飞机，利用计算机和 EduCake 直接来控制这些遥控玩具，作更多的应用。