實習一 ：arduino監視視窗之操作使用

```
//chapter 1
Int   val;//定義變數val
Int   ledpin=13;//定義數位介面13
void setup()
{
Serial.begin(9600);
//設置串列傳輸速率為9600，這裡要跟軟體設置相一致。當接入特定設備（如：藍牙）時，我們也要跟其他設備的串列傳輸速率達到一致。
pinMode(ledpin,OUTPUT);//設置數位13 口為輸出介面，Arduino 上我們用到的I/O 口都要進行類似這樣的定義。
}
void loop()
{
val=Serial.read();//讀取PC 機發送給Arduino的指令或字元，並將該指令或字元賦給val
if(val=='R')   //判斷接收到的指令或字元是否是“R”。
{digitalWrite(ledpin,HIGH);
//如果接收到的是“R”字元
//點亮數字13 口LED。
delay(500);
digitalWrite(ledpin,LOW);//熄滅數字13 口LED
delay(500);
Serial.println("Hello World!");
//顯示“Hello World！”字串
}
}
```

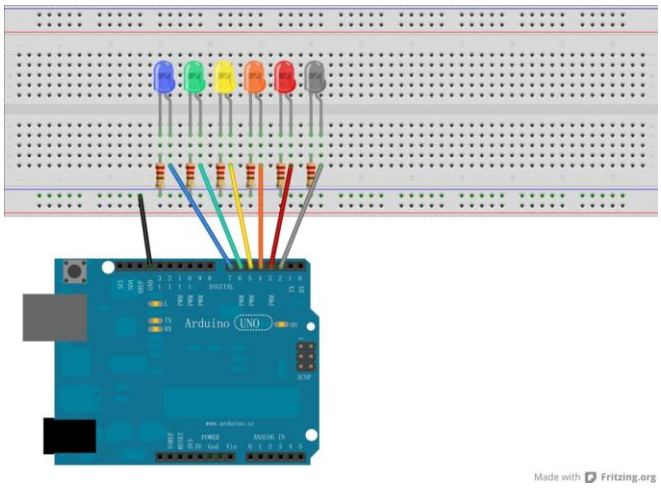
實習二 閃爍燈

```
//chapter 2   led blink
int ledPin = 13; //定義數位10 介面
void setup()
{
pinMode(ledPin, OUTPUT);//定義小燈介面為輸出介面
}
void loop()
{
digitalWrite(ledPin, HIGH); //點亮小燈
delay(1000); //延時1 秒
digitalWrite(ledPin, LOW); //熄滅小燈
delay(1000); // 延時1 秒
}
```

實習三 廣告燈效果實驗

1)實驗器件　Led燈:6個　220Ω的電阻：6個　多彩麵包板實驗跳線：若干

2) 實驗連線 按照二級管的接線方法，將六個LED燈依次接到數字2~7引腳上。如圖：



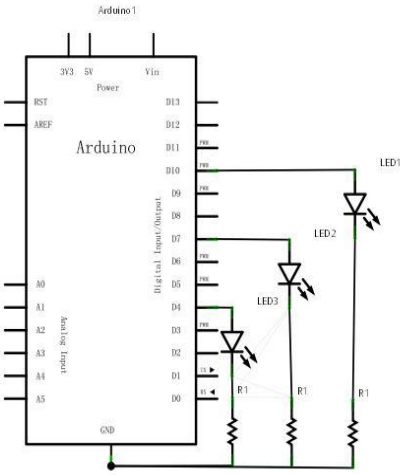
3) 實驗原理 在生活中我們經常會看到一些由各種顏色的led燈組成的看板，看板上各個位置上乩led燈不斷的變話,形成各種效果。本節實驗就是利用led燈程式設計模擬廣告燈效果。程式參考:

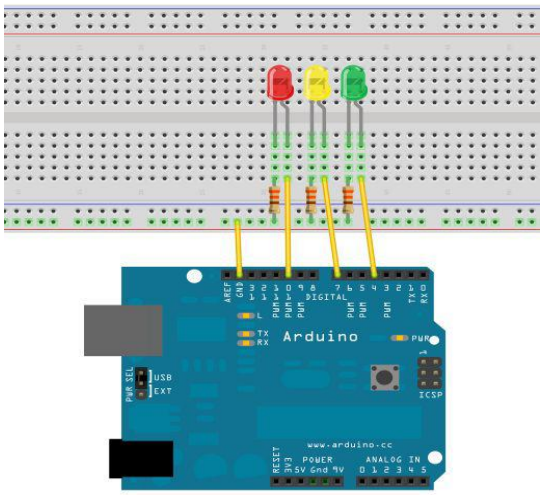
```
int BASE = 2 ;   //第一顆 LED 接的 I/O 腳
int NUM = 6;    //LED 的總數

void setup()
{
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      pinMode(i, OUTPUT);    //設定數字I/O腳為輸出
```

```
    }
}
void loop()
{
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      digitalWrite(i, LOW);      //設定數字I/O腳輸出為"低"，即逐漸關燈
      delay(200);                //延遲
    }
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      digitalWrite(i, HIGH);     //設定數字I/O腳輸出為"低"，即逐漸開燈
      delay(200);                //延遲
    }
}
```

實習四 :交通燈設計實驗

3個顏色的led，就可以實現模擬交通燈的實驗了。完成這個實驗所需的元件除了Arduino 控制器和下載線還需要的硬體如下：紅色M5 直插LED*1 黃色M5 直插LED*1 綠色M5 直插LED*1220Ω電阻*3 麵包板*1準備好上述元件，按照上面小燈閃爍的實驗，下面是我們提供參考的原理圖，使用的分別是數字10、7、4、介面.





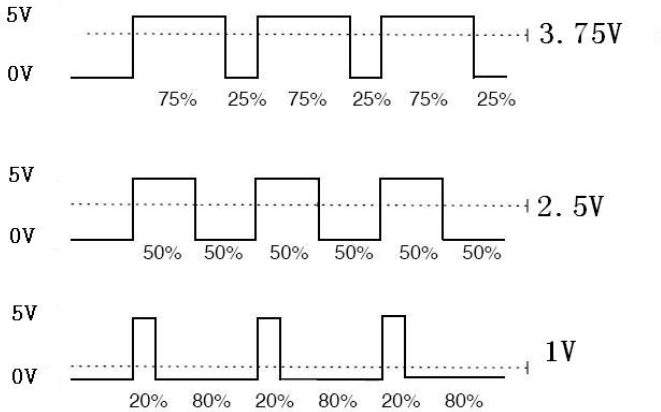
既然是交通燈模擬實驗，紅黃綠三色小燈閃爍時間就要模擬真實的交通燈，我們使用Arduino的delay（ ）函數來控制延時時間下面是一段參考程式：

```
int redled =10; //定義數位10 介面
int yellowled =7; //定義數位7 介面
int greenled =4; //定義數位4 介面
void setup()
{
pinMode(redled, OUTPUT);//定義紅色小燈介面為輸出介面
pinMode(yellowled, OUTPUT); //定義黃色小燈介面為輸出介面
pinMode(greenled, OUTPUT); //定義綠色小燈介面為輸出介面
}
void loop()
{
digitalWrite(redled, HIGH);//點亮紅色小燈
delay(1000);//延時1 秒
digitalWrite(redled, LOW); //熄滅紅色小燈
digitalWrite(yellowled, HIGH);//點亮黃色小燈
delay(200);//延時0.2 秒
digitalWrite(yellowled, LOW);//熄滅黃色小燈
digitalWrite(greenled, HIGH);//點亮綠色小
```

燈
delay(1000);//延時1 秒
digitalWrite(greenled, LOW);//熄滅綠色小燈
}

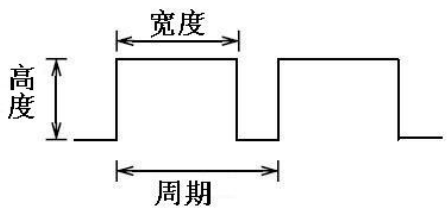
實習五 PWM 調控燈光亮度實驗

Pulse Width Modulation 就是通常所說的PWM，譯為脈衝寬度調製，簡稱脈寬調製。脈衝寬度調製（PWM）是一種對類比信號電平進行數位編碼的方法，由於電腦不能輸出類比電壓，只能輸出0 或5V 的的數位電壓值，我們就通過使用高解析度計數器，利用方波的占空比被調製的方法來對一個具體類比信號的電平進行編碼。PWM 信號仍然是數位的，因為在給定的任何時刻，滿幅值的直流供電要麼是5V(ON)，要麼是0V(OFF)。電壓或電流源是以一種通(ON)或斷(OFF)的重複脈衝序列被加到類比負載上去的。通的時候即是直流供電被加到負載上的時候，斷的時候即是供電被斷開的時候。只要頻寬足夠，任何模擬值都可以使用PWM 進行編碼。輸出的電壓值是通過通和斷的時間進行計算的。輸出電壓=（接通時間/脈衝時間）*最大電壓值



PWM 被用在許多地方，調光燈具、電機調速、聲音的製作等等。

下面介紹一下PWM 的三個基本參數：

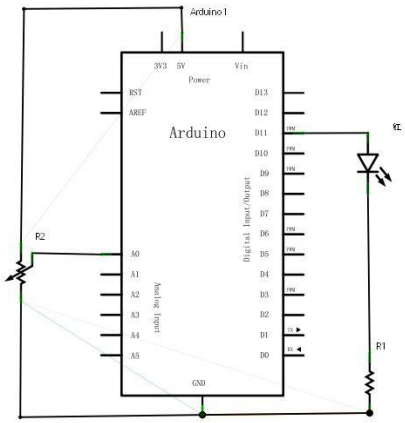


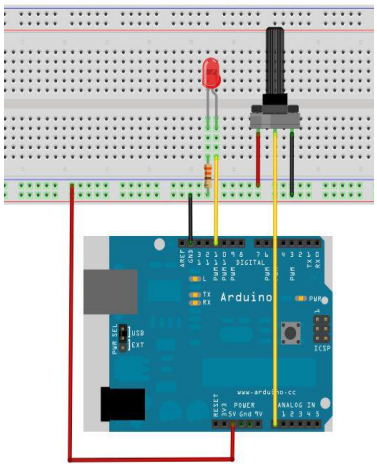
1、脈衝寬度變化幅度（最小值/最大值）
2、脈衝週期（1 秒內脈衝頻率個數的倒數）
3、電壓高度（例如：0V-5V）

Arduino 控制器有6 個PWM 介面分別是數位介面3、5、6、9、10、11，前面我們已經做了按鍵控制小燈的實驗，那是數位信號控制數字介面的實驗，我們也做過電位計的實驗，這次我們就來完成一個用電位計控制小燈的實驗。需要的元器件有：

電位計模組*1 紅色M5 直插LED*1 220Ω直插電阻 麵包板*1 麵包板跳線*1 紮

電位計即為模擬值輸入我們接到模擬口，小燈我們接到PWM 介面上，這樣通過產生不同的PWM 信號就可以讓小燈有亮度不同的變化。先按照下面的原理圖連接實物圖。





在編寫程式的過程中，我們會用到類比寫入analogWrite(PWM 介面，類比值)函數，對於類比寫入analogWrite()函數，此函數用法也很簡單，我們在本實驗中讀取電位計的類比值信號並將其賦給PWM 介面使小燈產生相應的亮度

變化，再在螢幕上顯示出讀取的類比值，大家可以理解為此程式是在類比值讀取的實驗程式中多加了將類比值賦給PWM 介面這一部分，下面給大家提供一段參考來源程式。

參考來源程式：

```
int potpin=0;//定義類比介面0
int ledpin=11;//定義數位介面11（PWM 輸出）
int val=0;// 暫存來自感測器的變數數值
void setup()
{
pinMode(ledpin,OUTPUT);//定義數位介面11為輸出
Serial.begin(9600);//設置串列傳輸速率為9600
//注意：類比介面自動設置為輸入
}
void loop()
{
val=analogRead(potpin);// 讀取感測器的模擬值並賦值給val
Serial.println(val);//顯示val 變數
analogWrite(ledpin,val/4);// 打開LED 並設置亮度（PWM 輸出最大值255）
delay(10);//延時0.01 秒
}
```

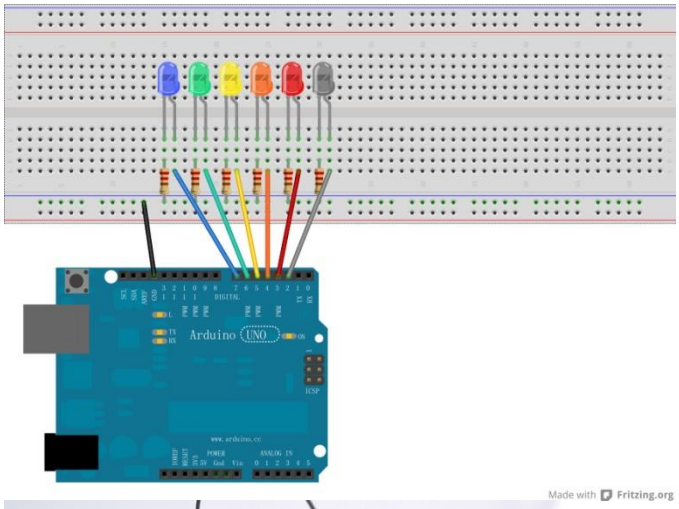
下載完程式，我們旋轉電位計的旋鈕不但可以看到螢幕上數值的變化還也可以清楚的看到我們麵包板上的LED 小燈的亮度也在隨之變化。

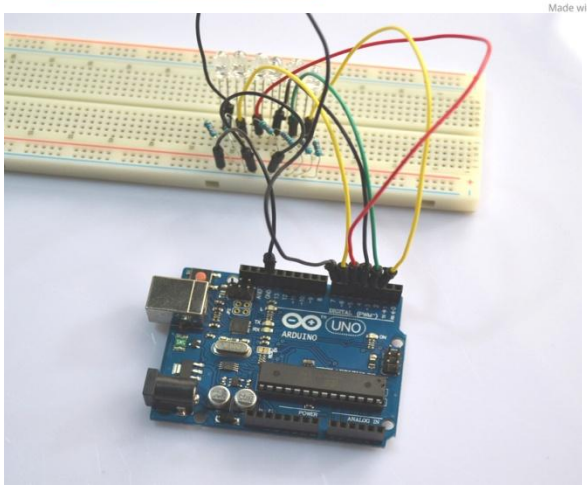
實習六 廣告燈效果實驗

實驗連線

按照二級管的接線方法，將六個LED燈依次接到數字2~7引腳上。如圖：

廣告燈實驗的接線





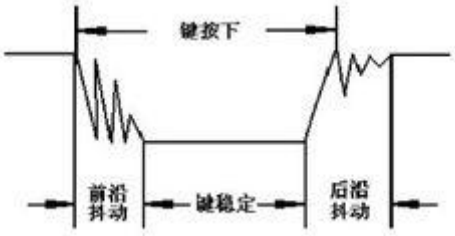
3）實驗原理

在生活中我們經常會看到一些由各種顏色的led燈組成的看板，看板上各個位置上乩led燈不斷的變話,形成各種效果。本節實驗就是利用led燈程式設計模擬廣告燈效果。

程式參考:

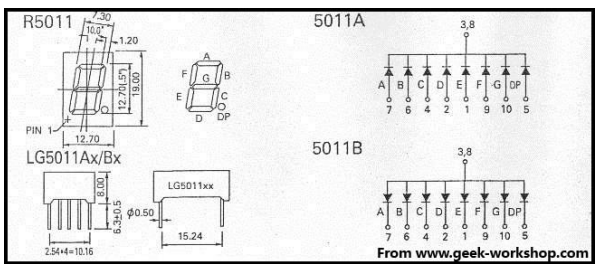
```
int BASE = 2 ;  //第一顆 LED 接的 I/O 腳
int NUM = 6;    //LED 的總數
void setup()
{
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      pinMode(i, OUTPUT);    //設定數字I/O腳為輸出
    }
}
void loop()
{
    for (int i = BASE; i < BASE + NUM; i ++)
```

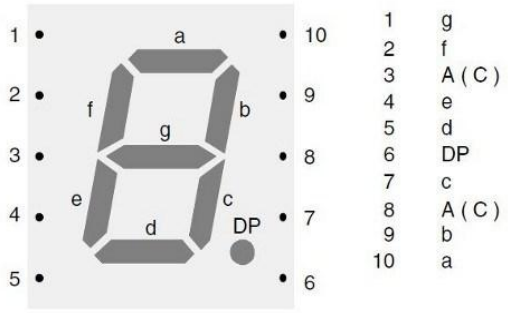
```
    {
      digitalWrite(i, LOW);      //設定數字I/O腳輸出為"低"，即逐漸關燈
      delay(200);            //延遲
    }
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      digitalWrite(i, HIGH);      //設定數字I/O腳輸出為"低"，即逐漸開燈
      delay(200);            //延遲
    }
}

/*實習七: 開關連接實驗(軟體防彈跳)*/
// 常數宣告，宣告 I/O 腳號
const int buttonPin = 2;    // 宣告按鍵為D2 的常數
const int ledPin =   10;      // 宣告 LED 為D10 的常數

// 變數宣告，執行期間數值會改變
boolean buttonState;          // 宣告讀取按鍵狀態的變數
boolean ledState = HIGH;    // 宣告LED的狀態，一開始預設為不亮

void setup() {
  pinMode(ledPin, OUTPUT);              // 初始化 LED pin 為輸出
  digitalWrite(ledPin, ledState);   // 輸出LED狀態至LED，預設為不亮
  pinMode(buttonPin, INPUT);          // 初始化 按鍵開關 pin 為輸入
}

void loop(){
  buttonState = digitalRead(buttonPin);
// 讀取按鍵開關的狀態
  // 檢查按鍵開關是否被按下，若是，其狀態值為 LOW
  if (buttonState == LOW) {
    ledState=!ledState;  // 將 ledState 取補數，即High變Low, Low變High
    digitalWrite(ledPin, ledState);  // 將 ledState 的值直接輸出至LED
    delay(20);    // 延遲20ms略過開關彈跳時間
    while (1) {   // 永久迴圈（等待按鍵放開）
      buttonState = digitalRead(buttonPin);
      if (buttonState == HIGH) break;
// 若按鍵放開則離開 while 迴圈
    }
    delay(20);    // 延遲20ms略過開關彈跳時間
  }
}
```




```
/*   實習八:七段顯示器實驗 由0數至9   */
// 七段控制腳陣列，分別對應a～g段
int   seg7[] = {11, 10, 9, 8, 7, 6, 5};
// 七節顯示器編碼表
char
TAB[]={ 0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x27,0x7F,0x67 };
int ii,jj;      // 迴圈用的變數
// 此 setup 程序只有在微控制器按 reset 時執行一次
```

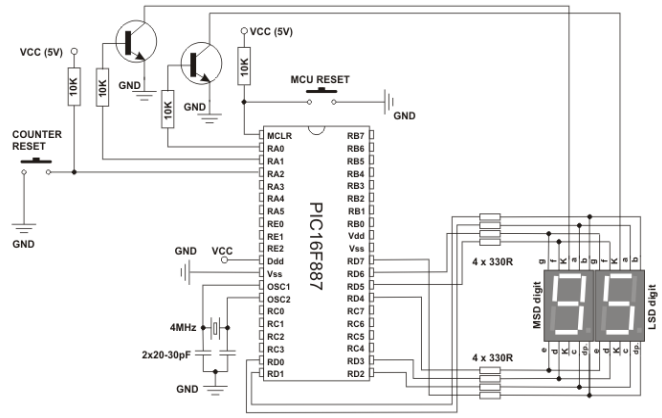
```
void setup() {
  // 初始化指定的數位腳位為輸出模式
  for (ii=0; ii<7; ii++) {
      pinMode(seg7[ii], OUTPUT);
  }
}
// 此 loop 程序會一直重覆執行
void loop() {
  for (ii=0; ii<10; ii++) {   // 依序顯示 0~9
    OutPort(TAB[ii]);          // 取出陣列中對應的數字編碼表
    delay(500);                // 延遲 0.5 秒
  }
}
// 將指定值顯示在七段顯示器上，最低位元為a，依序為 abcdefg
void OutPort(byte dat) {
  for (jj=0; jj<7; jj++) {
    if (dat % 2==1)  // 取出 dat 的最低位元
      digitalWrite(seg7[jj], HIGH);  // 若為 1 代表該段要亮, 輸出高準位
    else
      digitalWrite(seg7[jj], LOW);   // 若為 0 代表該段要滅, 輸出低準位
    dat=dat/2;  // 除2，進行下一位元的處理
  }
}
```

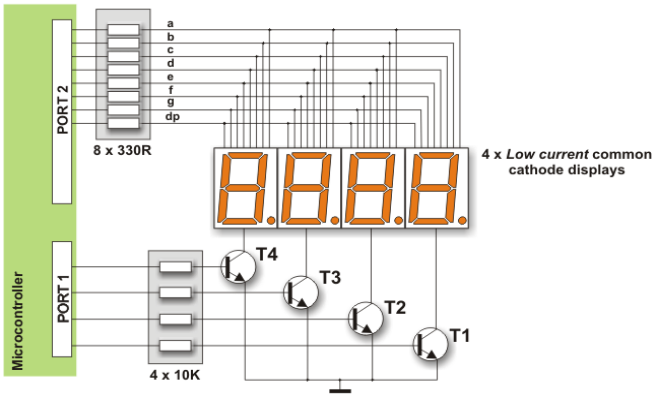


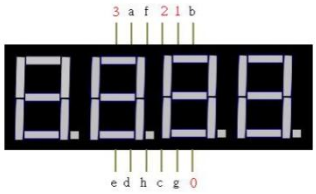



```
/實習九:四位數七段顯示器 /
程式
int i;
int j;
int count=0;
int number;
unsigned long time=0;
const byte num[10]={
      B11000000,   //0
      B11111001,   //1
      B10100100,   //2
      B10110000,   //3
      B10011001,   //4
      B10010010,   //5
      B10000010,   //6
      B11111000,   //7
      B10000000,   //8
      B10010000}; //9
const int seg[]={2,3,4,5,6,7,8,9};
//abcdefgp
const int digit[]={10,11,12,13};   //D0-D3
void setup()
{
  for(i=0;i<8;i++)
     pinMode(seg[i],OUTPUT);
  for(i=0;i<4;i++)
```

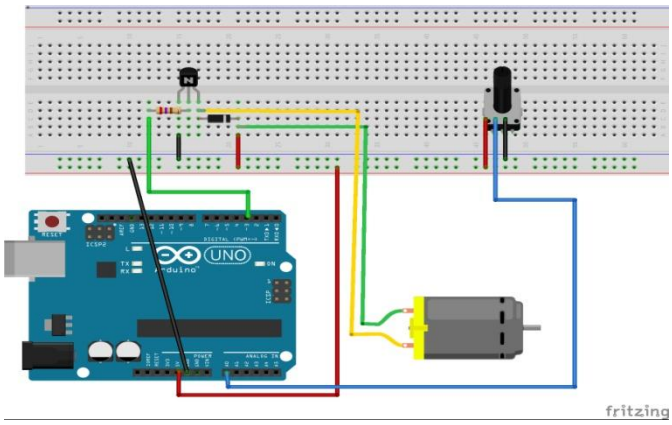
```
  {
    pinMode(digit[i],OUTPUT);
    digitalWrite(digit[i],LOW);
  }
}
void loop()
{
    number=count;
    for(i=3;i>=0;i--)
    {
      for(j=0;j<8;j++)
      {
        if(bitRead(num[number%10],j))
          digitalWrite(seg[j],LOW);
        else
          digitalWrite(seg[j],HIGH);
      }
      digitalWrite(digit[i],HIGH);
      delay(5);
      digitalWrite(digit[i],LOW);
      number=number/10;
      if(millis()-time>=1000)
      {
          time=millis();
          count=count+1;
          if(count>9999)
            count=0;
      }
    }
}
```







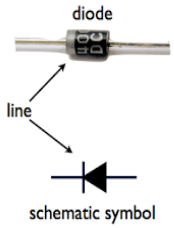
/實習十:DC馬達轉速控制/

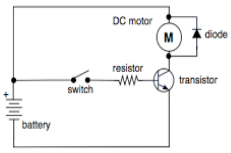


直流馬達的PWM 調速控制為在直流馬達控制系統中，為了減少流經馬達繞線電流及降低功率消耗等目的，常常使用脈波寬度調變信號(PWM)來控制交換式功率元件的開與關動作時間。 其最常使用的就是藉著改變輸出脈波寬度或頻率來改變馬達的轉。

```
int motorPin = 3;
int potentioValue = 0;   //連接可變電阻接腳
```

```
void setup()
{
  pinMode(motorPin, OUTPUT);   //設定輸出
}
void loop()
{
  potentioValue = analogRead(0); //讀取可變電阻數值。
  potentioValue = map(potentioValue, 0,1023,0,255); //數值轉換
  analogWrite(motorPin, potentioValue); // PWM 的輸出
}
```



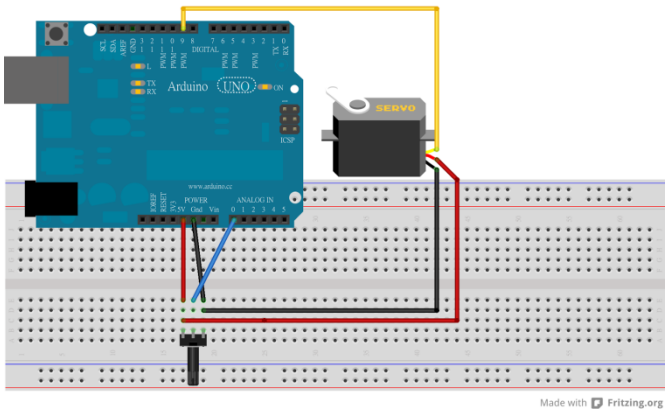


/實習十一:伺服馬達控制/

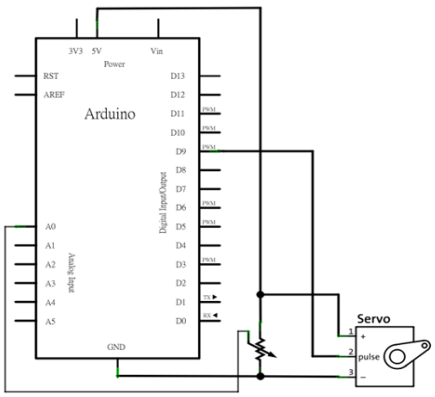
Arduino：使用可變電阻控制伺服馬達

實驗目的透過可變電阻，控制伺服馬達的轉向。

Arduino	伺服馬達
Pin 9	橘線
GND	棕線
5V	紅線




```
// 引用 Servo Library
#include
Servo myservo; //建立一個伺服馬達控制物件
int potpin = 0; //該變數用於存儲用電位器讀出的模擬值
int val;        // 該變數用於儲存伺服馬達角度位置
void setup()
{
myservo.attach(9);    // 由 Pin 9 控制伺服馬達
}
void loop()
{
val = analogRead(potpin);    //讀取電位器控制的模擬值 (範圍在 0-1023)
val = map(val, 0, 1023, 0, 179);   // 把 0-1023 的數值按比例縮放為 0-180 的數值
myservo.write(val);          // 指定伺服馬達轉向的角度
delay(15);
 }     // 等待15ms讓伺服馬達到達指定位置
```




```
// 引用 Servo Library
#include <Servo.h>
 // 建立一個 Servo 物件
Servo myservo;
// 旋轉角度
int value = 0;
```

```
void setup()
{
  myservo.attach(9);  // Servo 接在 pin 9
}
 void loop()
{
  if (value == 0)   value = 180;
  else
     value = 0;
 // 叫 Servo 旋轉角度:
 // myservo.write(0) 是叫 Servo 旋轉到 0 度的位置
 //myservo.write(180) 是叫 Servo 旋轉到 180 度的位置
  myservo.write(value);
  delay(1500);
```

/實習十二:伺服馬達控制/

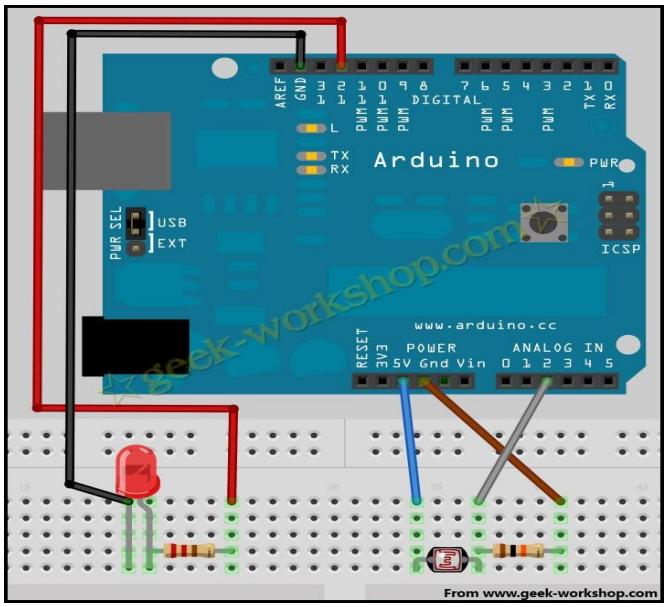
利用 Arduino 控制二顆伺服馬達

```
#include <Servo.h>
Servo servoLeft;            // 宣告左邊伺服馬達
Servo servoRight;           // 宣告右邊伺服馬達
void setup() {
  servoLeft.attach(10);//將 Pin 10 指定為左邊伺服馬達
  servoRight.attach(9);// 將 Pin 9 指定為右邊伺服馬達
}
void loop() {
  initial();            // 馬達位置歸零
  delay(2000);          // 執行後停止兩秒
     same_degree();     // 左右伺服馬達同樣各轉 30 度
  delay(2000);          // 執行後停止兩秒
     initial();         // 馬達位置歸零
  delay(2000);          // 執行後停止兩秒
     dif_degree();  // 左伺服馬達轉 30 度，右伺服馬達轉 150 度
  delay(2000);          // 執行後停止兩秒
}
/* 以下為副程式宣告 */
void initial(){
  servoLeft.write(0);
  servoRight.write(0);
}
void same_degree() {
  servoLeft.write(30);
  servoRight.write(30);
}
void dif_degree() {
  servoLeft.write(30);
  servoRight.write(150);    }
```

實習十三:光敏電阻光控制 led

光敏電阻又稱光導管，常用的製作材料為硫化鎘。實驗前先測量一下當前環境下光敏電阻的亮阻值與暗阻值 下圖是測出來的 LED 亮阻值，為 9.1KΩ

下圖是測出來的 LED 暗阻值，為 32.4KΩ



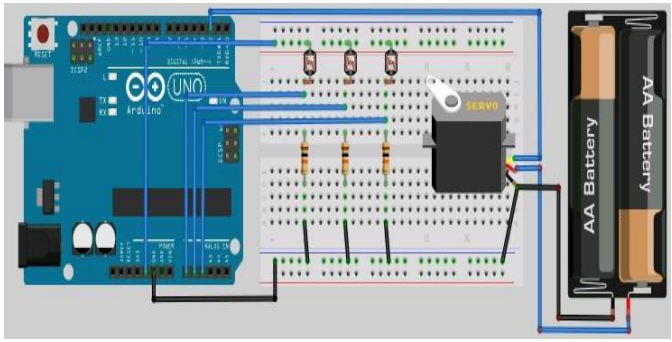
根據測出來的亮阻 9.1KΩ，暗阻 32.4KΩ。選定分壓電阻為 10KΩ。因為當有遮擋物的後，阻值會變大 假設亮阻為 10KΩ(對於光敏電阻來說，與測量出來的 9.1KΩ 差別不大，計算起來更加方便了)，分壓阻值為 10K 歐姆。模擬 2 號口所測量的觸發電壓為 10KΩ 分壓電阻的，在 5V 電源供電下，亮與暗轉換的觸發電壓為 5×10÷(10+10)=2.5V。當光線越暗，光敏電阻的阻值也就越大，分壓兩端電壓也就越小。所以觸發條件就為≤2.5V。

程 式 13

```
int photocellPin = 2 ; // 定義變量 photocellsh=2，為電壓讀取端口。
int ledPin = 12 ; //定義變量 ledPin=12，為 led 電平輸出端口
int val = 0 ; //定義 val 變量的起始值
 void  setup ( )  {
  pinMode ( ledPin, OUTPUT ) ; // 使 ledPin 為輸出模式
 }
 void  loop ( )  {
  val = analogRead ( photocellPin ) ; //從傳感器讀取值
  if ( val<= 512 ) {      //512=2.5V，想讓傳感器敏感一些的時候，把數值調高，想讓傳感器遲鈍的時候把數值調低。
     digitalWrite ( ledPin, HIGH ) ; //當 val 小於 512(2.5V)的時候，led 亮。
  }
  else {
     digitalWrite ( ledPin, LOW ) ;
  }
}
```

實習十四:光線位置控制伺服馬達轉動角度

以三個光敏電阻



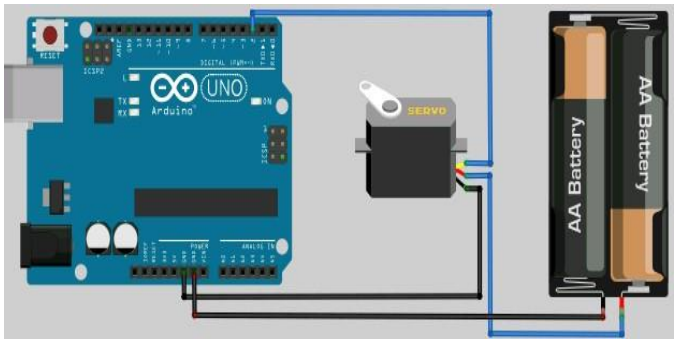
程 式 14

```
#include <Servo.h>
Servo servo;
void setup()
{
  servo.attach(2);
  servo.write(0);
```

```
}
void loop()
{
    const int cds[3]={0,1,2};
    int val[3];
    for(int i=0;i<3;i++)

       val[i]=analogRead(cds[i]);
    if(val[0]>800 && val[1]<800 &&
  val[2]<800)
       servo.write(0);
    else if(val[0]>800 && val[1]>800 &&
  val[2]<800)
       servo.write(45);
    else if(val[0]<800 && val[1]>800 &&
  val[2]<800)
       servo.write(90);
    else if(val[0]<800 && val[1]>800 &&
  val[2]>800)
       servo.write(135);
    else if(val[0]<800 && val[1]<800 &&
  val[2]>800)
       servo.write(180);
}
```

實習十五:鍵盤輸入控制伺服馬達轉動角度

利用串列阜傳送電腦鍵盤輸入按鍵值控制伺服馬達轉動角度:

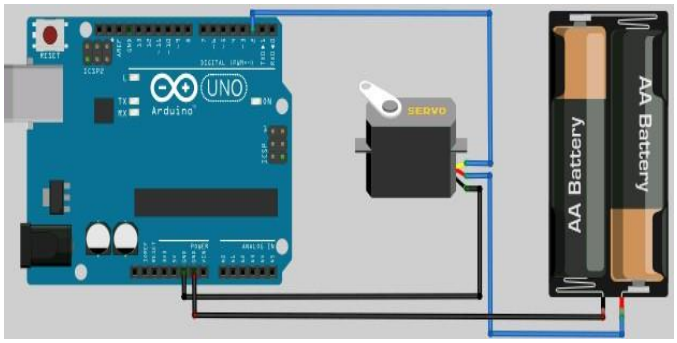


程 式 15

```
#include <Servo.h>
Servo servo;
void setup()
{
   Serial.begin(9600);
   servo.attach(2);
}
void loop()
{
   if(Serial.available())
   {
       char ch=Serial.read();
       Serial.println(ch);
       if(ch=='a')
           servo.write(0);
       else if(ch=='b')
           servo.write(90);
       else if(ch=='c')
           servo.write(180);
   }
}
```

實習十六:鍵盤輸入控制伺服馬達轉動角度

利用串列阜傳送電腦鍵盤輸入按鍵值控制伺服馬達轉動角度:



程 式 16

```
#include <Servo.h>
Servo servo;
```

```
void setup()
{
  Serial.begin(9600);
  servo.attach(2);
}
void loop()
{
  if(Serial.available())
  {
      char ch=Serial.read();
      Serial.println(ch);
      if(ch=='a')
          servo.write(0);
      else if(ch=='b')
          servo.write(90);
      else if(ch=='c')
          servo.write(180);
  }
}
```

文獻參考資料

1.arduino 官方網站 http://arduino.cc/

2.最簡單的互動設計 Arduino 一試就上手 碁峰 孫駿榮、吳明展、盧聰勇

3.超圖解 Arduino 互動設計入門(第二版) 旗標 趙英傑

4. Arduino 互動設計專題與實戰 碁峰 柯博文

5. Arduino 最佳入門與應用 碁峰 楊明豐

6. Arduino 自造指南 碁峰 原文作者 :John Boxall 譯者：曾繁勛,陳麒元,趙涵捷