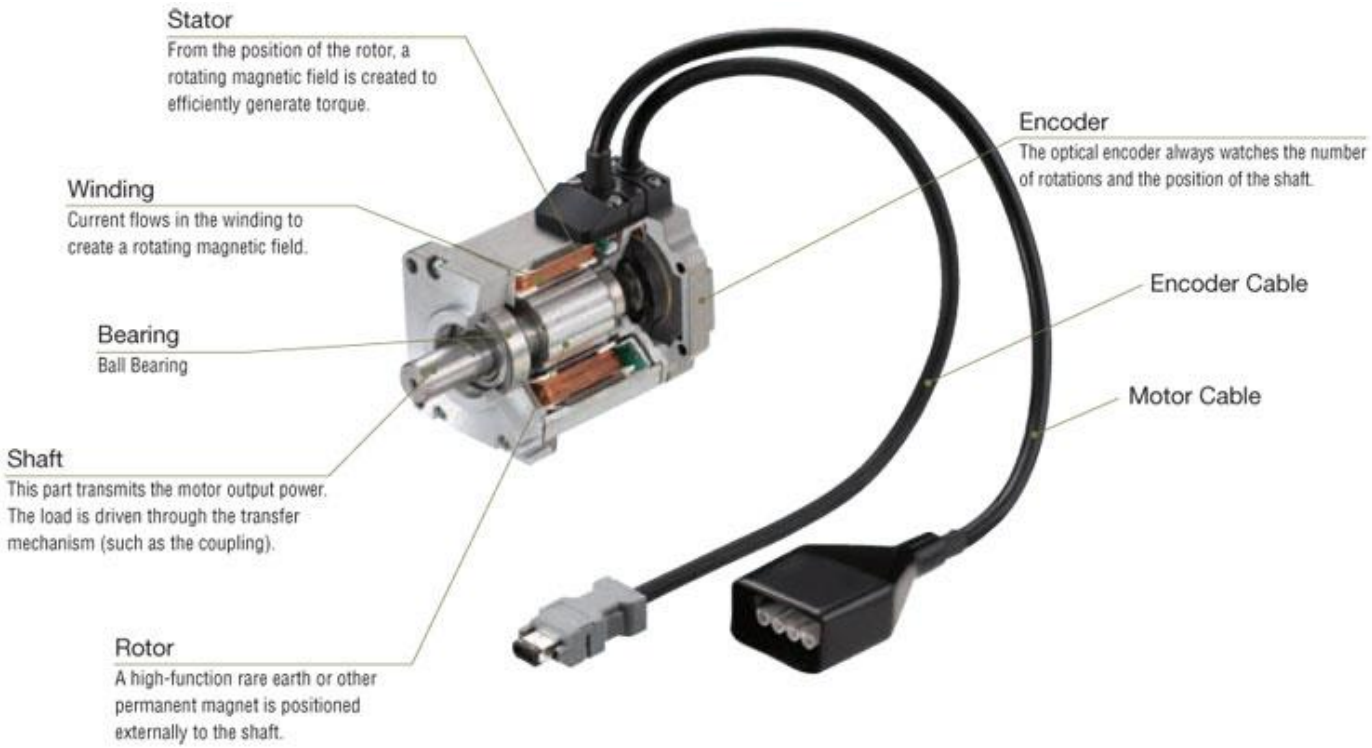
Lecture 11/28 – Servo Motor

FC Tien, YP Liu

Dept. IE&M, Taipei Tech

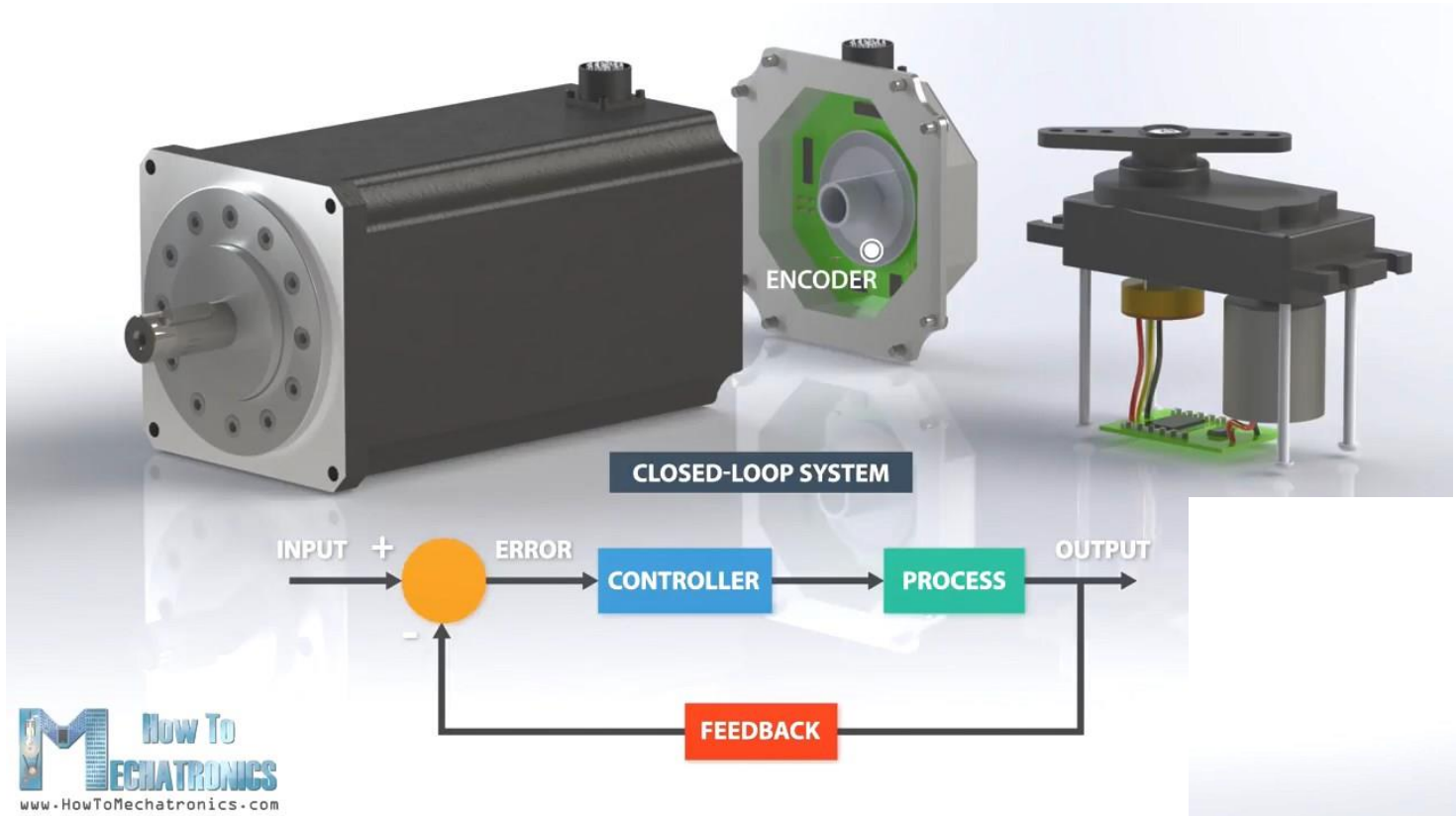
What is Servo Motor?

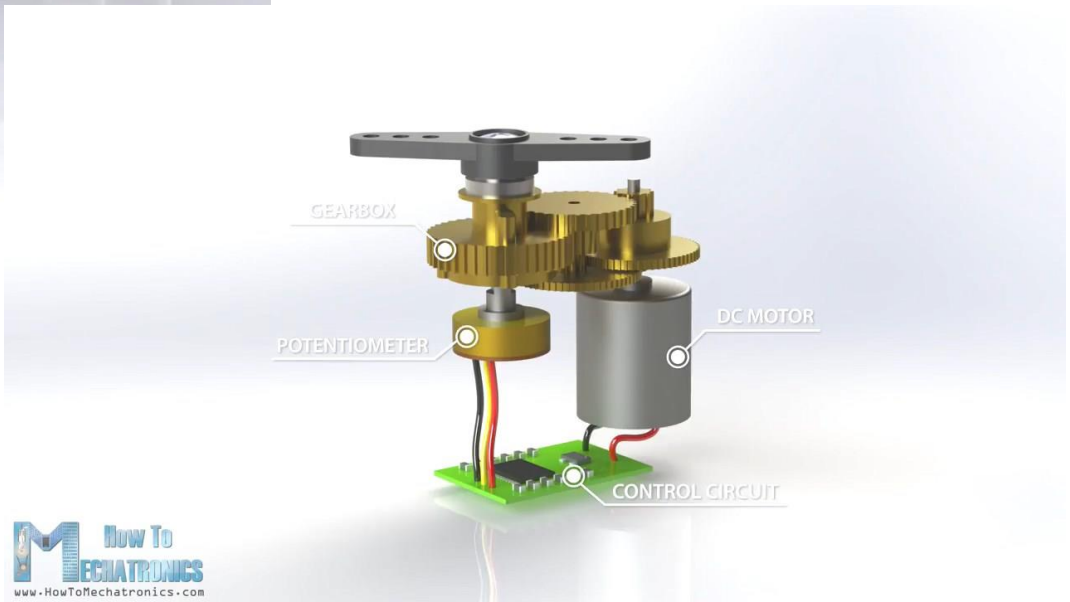
Structure of Servo Motors



伺服馬達（Servomotor）是對用於使用伺服機構的馬達（電動機）總稱。伺服（Servo）一詞來自拉丁文"Servus"，本為奴隸（Slave）之意，此指依照命令動作的意義。所謂伺服系統，就是依照指示命令動作所構成的控制裝置，應用於馬達的伺服控制，將感測器裝在馬達與控制對象機器上，偵測結果會返回伺服放大器與指令值做比較。由此可知，因為伺服馬達是以回饋訊號控制，與藉由輸入脈波訊號控制的步進馬達有所區別

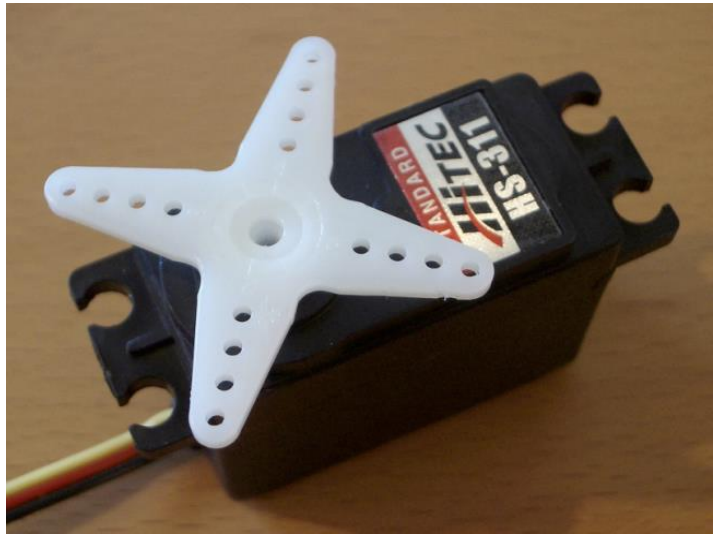
What is Servo Motor?





What is Servo Motor?

- 本實驗介紹一顆簡單的伺服馬達，它只需要很少或根本不需要外加的電路，就能夠輕易的被微處理器所控制，而且已經有現成的輸出端可以連接需要驅動的物件



- 具有位置控制或速度控制兩種
- 內部回傳線路幫助控制位置或速度
- 簡易三線(式)PWM 5V介面

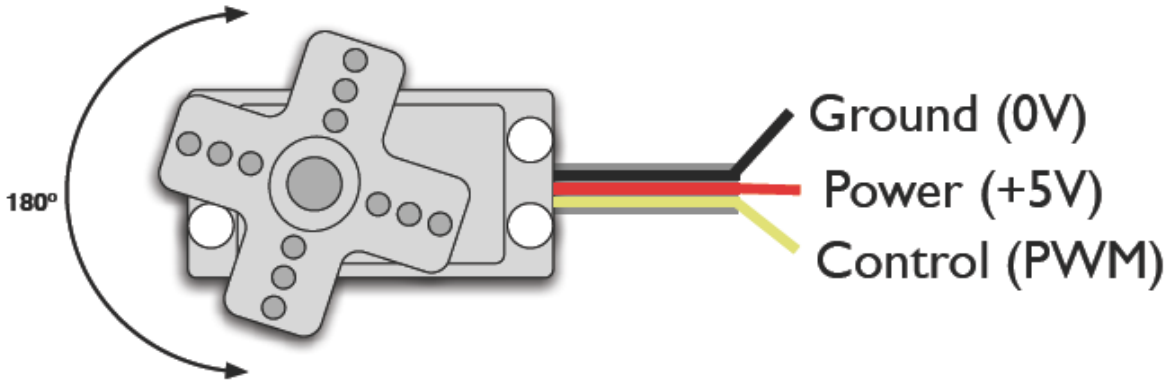
Types of Servo Motor

- 除了本實驗使用的伺服馬達外，市面上也有許多同樣是三線式的伺服馬達，從手指大小的伺服馬達到汽車用的大型伺服馬達都充斥著三線式伺服馬達的蹤影。
- 右圖為另外兩種三線式的伺服馬

9g

157g

三線式伺服馬達

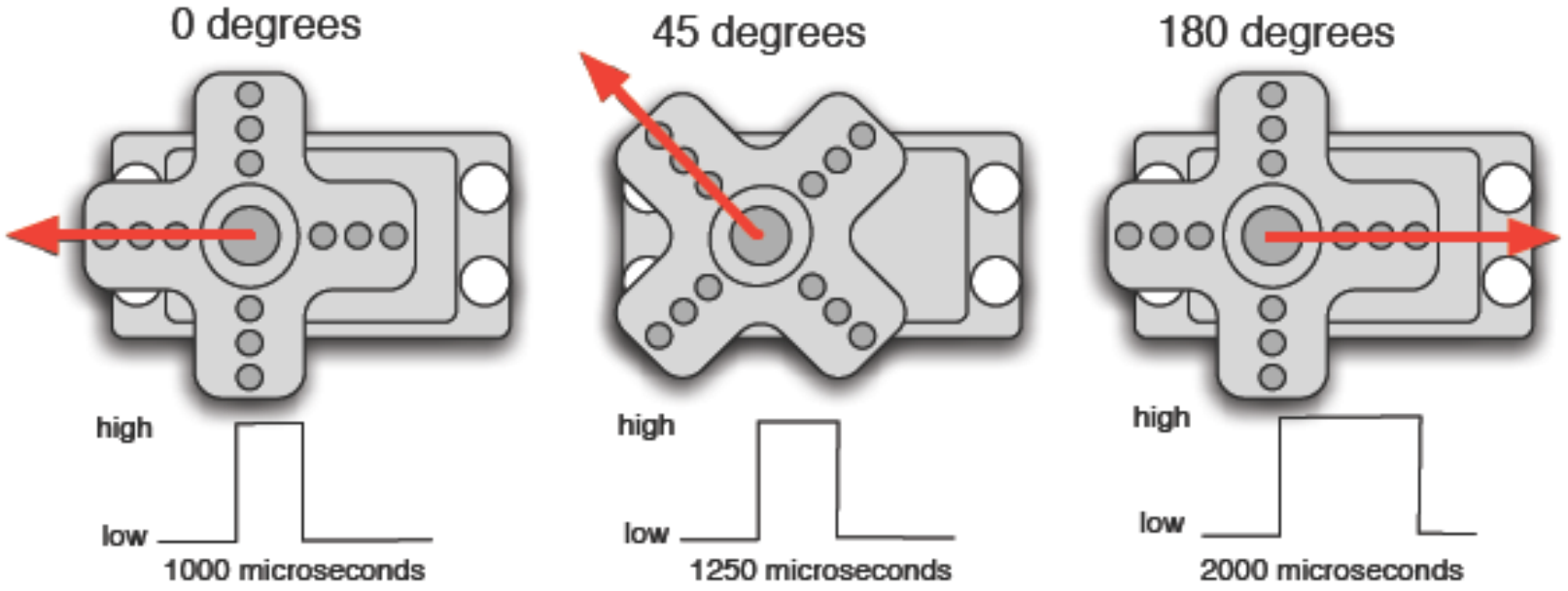


- PWM 頻率為 50 Hz (i.e. every 20 millisecs)
- 脈波寬度範圍 介於 1 到 2 millisecs
- 1 millisec(1000µs) = 完全反轉的位置
- 2 millisec(2000 µs)= 完全正轉位置

- 三線式伺服馬達的三條線各自用途如下圖所示
- 紅色和黑色線連結至電源供應處，而黃色線則為訊號線
- 三線式的伺服馬達控制方式為PWM訊號控制，經由控制PWM訊號HIGH的時間來控制馬達的轉速或位置

三線式伺服馬達

- 下圖為伺服馬達轉動角度與其PWM中HIGH的時間長短(Duty大小)關係圖



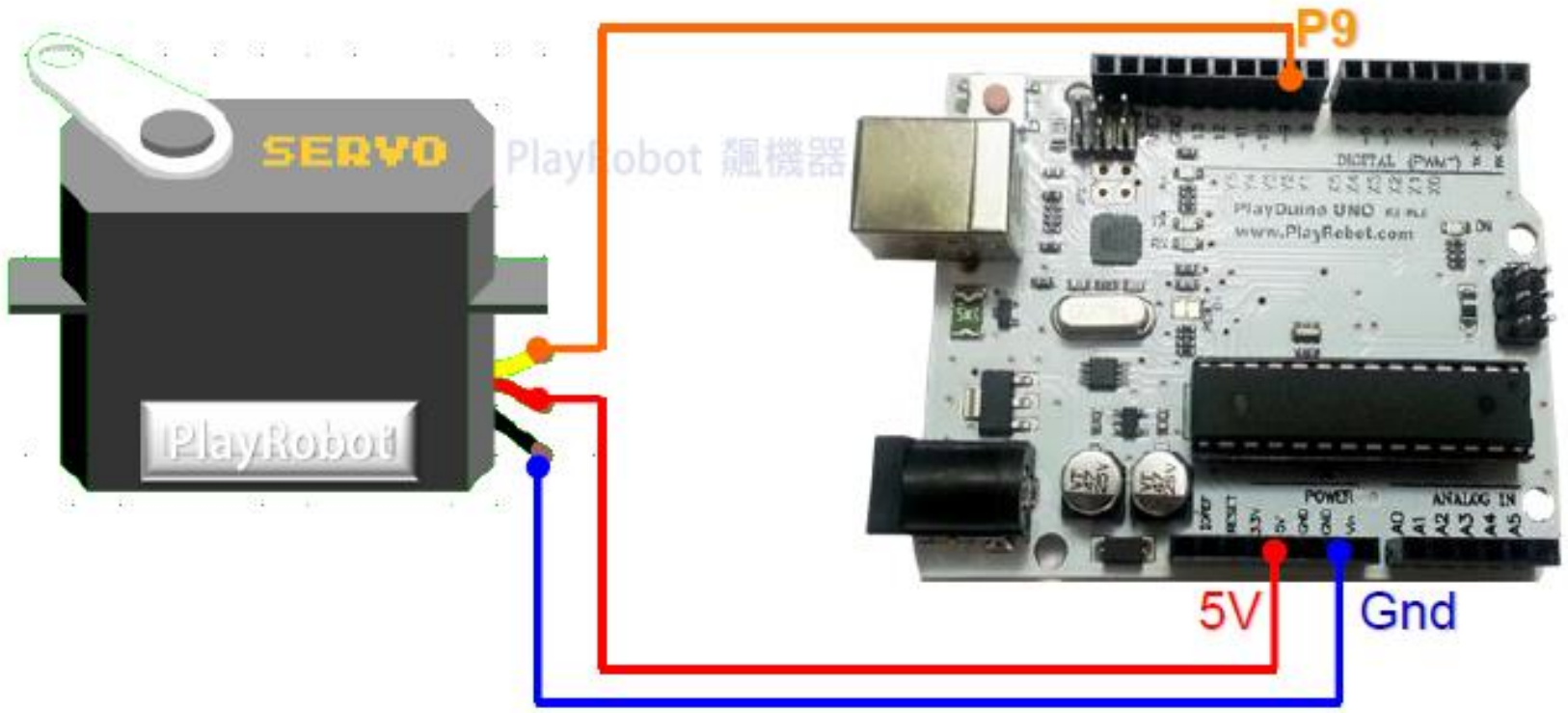
要使用PWM訊號控制伺服馬達需注意以下數點:

轉動的角度會依據不同的脈波

了保持同樣位置,脈波必須不斷傳送

因為伺服馬達轉動需要時間，所以脈波傳送過快的話會沒反應

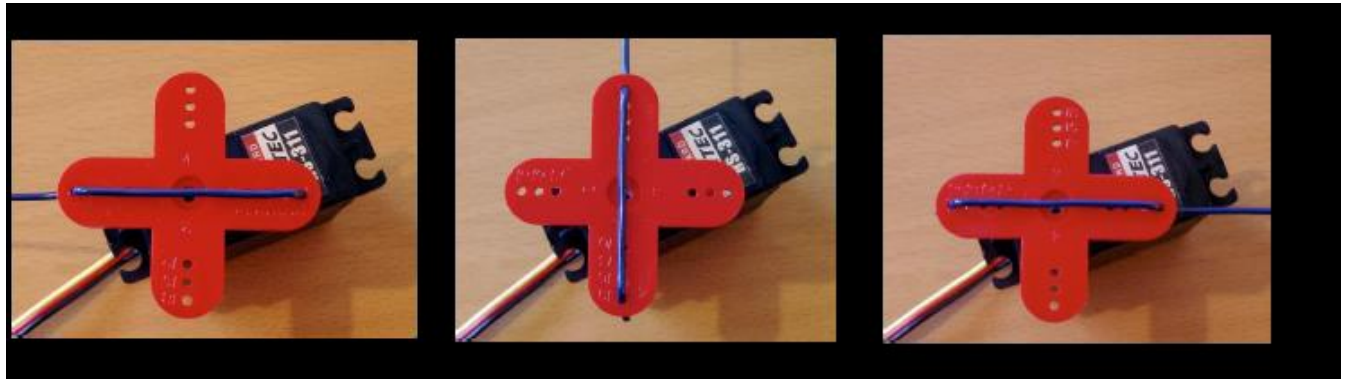
Servo Motor + Arduino 接線



- 以P9 來控制馬達之位置
- 以輸入之時間來控制馬達旋轉之角度
- 或以 Lib 直接控制角度

Practice

- 請練習調整脈波的持續時間(PWM 訊號中HIGH的時間)來得知伺服馬達角度將如何變化



脈波訊號持續時間（μs）	1000	1100	1200	1300	1400	1500
馬達角度						
脈波訊號持續時間（μs）	1600	1700	1800	1900	2000	
馬達角度						

Arduino 程式

```
//伺服馬達歸零(移至中立點)
int servoPin=9;                  //設定伺服馬達由9腳位控制
void setup(){
  pinMode(servoPin,OUTPUT);          //設定9腳位為輸出
 }
void loop(){          //輸出PWM訊號使伺服馬達移動到中立點
  digitalWrite(servoPin,HIGH);
  delayMicroseconds(1500);         //脈波寬度為1.5ms
  digitalWrite(servoPin,LOW);
  delay(20);
}
```

- 先確認執行此程式是否可以將馬達回到中立點
- 建立伺服馬達的函式
- 修改程式: 將伺服馬達定位在最小角度
- 修改程式: 伺服馬達定位在最大角度

Arduino 程式 – 反覆震盪

```
//伺服馬達反覆擺盪
int servoPin=9;                    //設定servoMotor由9腳位控制
void setup(){
  pinMode(servoPin,OUTPUT);        //設定9腳位為輸出
}
void loop(){
  for (int i=0;i<=40;i++) {        //伺服馬達定位在最小角度
    digitalWrite(servoPin,HIGH);
    delayMicroseconds(1000);
    digitalWrite(servoPin,LOW);
    delay(20);
    }
   for (int i=0;i<=40;i++){        //伺服馬達定位在最大角度
    digitalWrite(servoPin,HIGH);
    delayMicroseconds(2000);
    digitalWrite(servoPin,LOW);
    delay(20);
  }
}
```

Arduino 程式 using Lib <Servo.h>

```
#include <Servo.h>          // 引用 Servo Library
Servo myservo;              // 建立一個 Servo 物件
int value = 0;              // 旋轉角度
void setup(){
      myservo.attach(9);    // Servo 接在 pin 9
}
void loop(){
    if (value == 0){
          value = 180;      //myservo.write(180) 是叫 Servo 旋轉到 最大角度的位置
    }else{
          value = 0;        //myservo.write(0) 是叫 Servo 旋轉到 最小角度的位置
    }
    myservo.write(value);
   delay(1500);
}
```

- **經由引入 Lib 控制 servo motor**
- **指令**
  - **Servo myservo; 宣告一個instance**
  - **Servo.attach(pin); 連接控制之腳位**
  - **Servo.write(value); 旋轉角度**
  - **Servo.detach(); 結束控制**

程式 #1 --

RC 伺服馬達(Radio Controlled Servo Motor) 大部份是透過 PWM (Pulse Width Modulation, 脈波寬度調變)來控制，Arduino 裏內建了 Servo Library 讓事情變得很簡單，可以很輕鬆地控制伺服馬達

Command:

#include <Servo.h> // servo motor lib

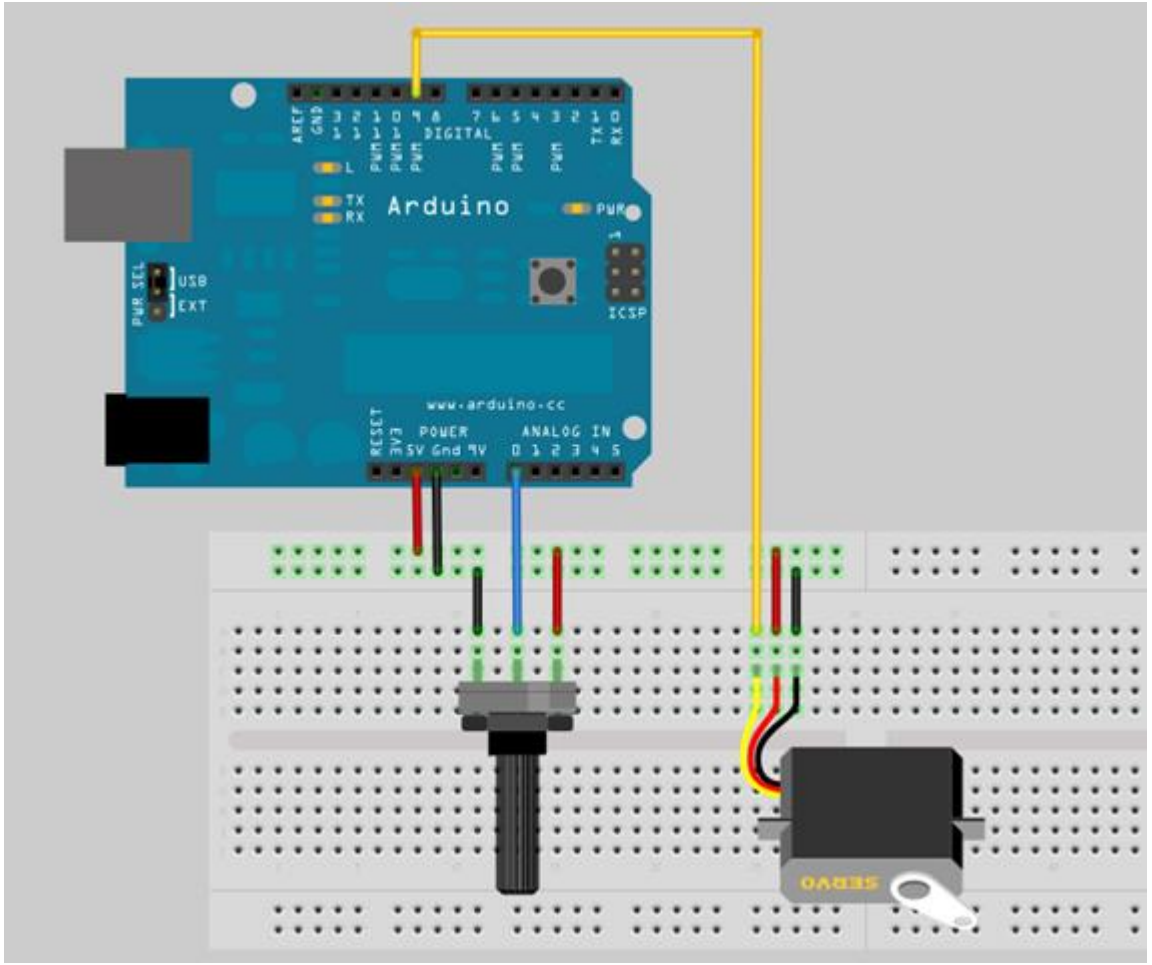
MyServo.attach(9); //Define Servo motor Signal Pin

MyServo.write(Turn_Angle); // Turn servo motor

```
#include <Servo.h>
Servo MyServo;  // build a servo object
int Turn_Angle =0;
int potpin = 0;

void setup() {
MyServo.attach(9); // Servo 接在 p9
Serial.begin(9600);
}

void loop() {
if (Turn_Angle == 0)
    Turn_Angle = 180;
  else
    Turn_Angle = 0;
  Serial.println(Turn_Angle);
  MyServo.write(Turn_Angle);
  delay(1500);
}
```

Practice 2: 另一接法

- 把伺服馬達紅線接到 +5v
- 黑線接到 GND，訊號線接到 pin 9 ，
- 可變電阻中間腳位接到類比輸入(Analog Input) pin 0，
- 剩下的兩支腳位，一支接到 5V，另外一支接到 GND



程式 #2 --可變電阻來控制RC Servo motor

運用可變電阻來控制RC Servo motor之角度

Command:

Turn_Angle = analogRead(potpin); // 讀取可變電阻(數值介於 0 到 1023)

Turn_Angle = map(Turn_Angle, 0, 1024, 0, 179); // scale the value

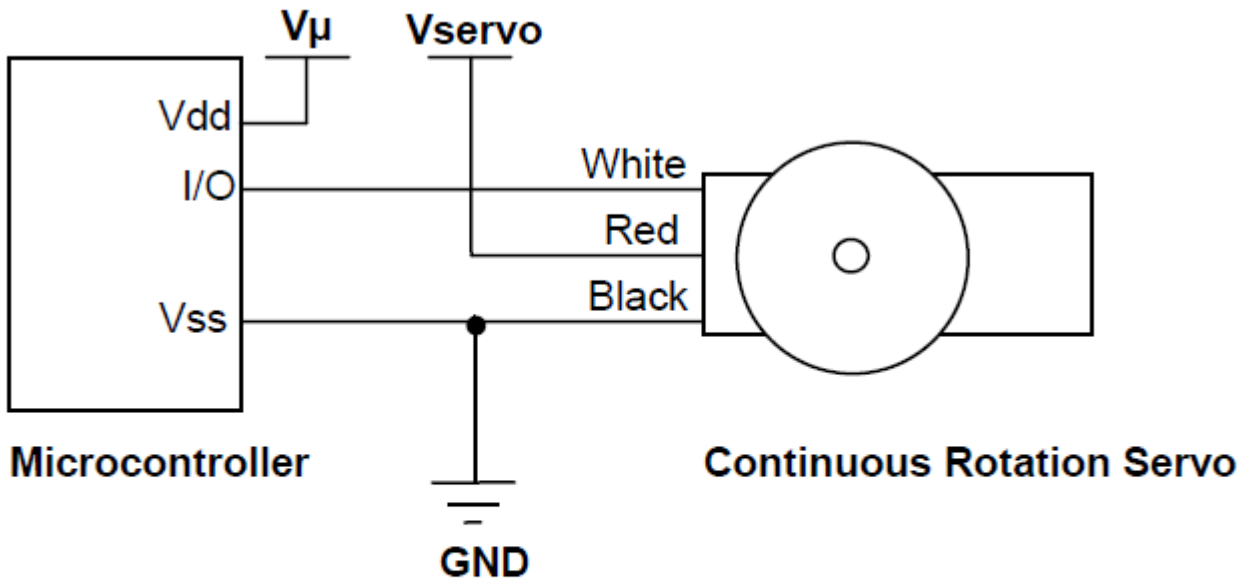
```
#include <Servo.h>
// build a servo object
Servo MyServo;
int Turn_Angle =0;
int potpin = 0;
void setup() {
  MyServo.attach(9);  // Servo 接在 p9
  Serial.begin(9600);
  }
void loop() {
  Turn_Angle = analogRead(potpin); // 讀取可變電阻(數值介於 0 到 1023)
  // put your main code here, to run repeatedly:
  Turn_Angle = map(Turn_Angle, 0, 1024, 0, 179); // scale the value
  Serial.println(Turn_Angle);
  MyServo.write(Turn_Angle);
  delay(15);
}
```

Parallax Continuous Rotation Servo

- **Bidirectional continuous rotation**
- **0 to 50 RPM**

Parallax Continuous Rotation Servo 接線方式

- 白 訊號
- 紅 +5V
- 黑 GND



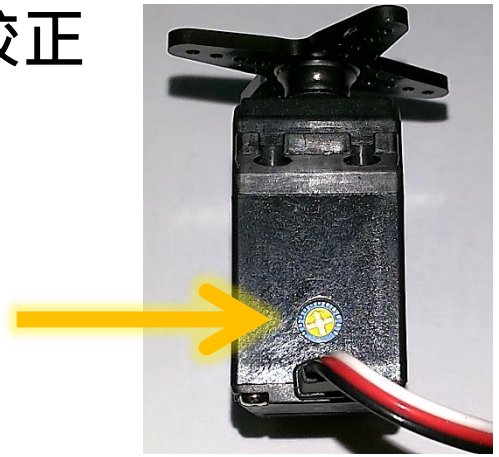
Vμ = microcontroller voltage supply

Vservo = 4 to 6 VDC, regulated or battery

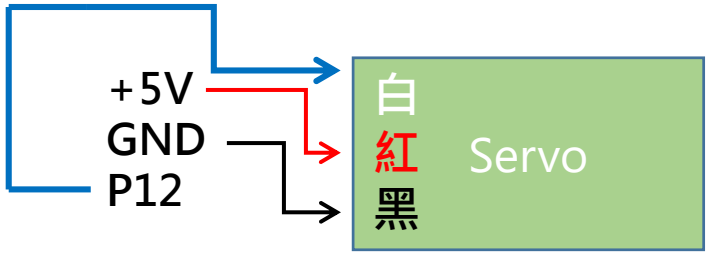
I/O = PWM TTL or CMOS output signal, 3.3 to 5 V; < Vservo + 0.2 V

Parallax Continuous Rotation Servo 驅動方式 1/2

- PWM 1300 us 順時針轉動
- PWM 1500 us 停止
- PWM 1700 us 逆時針轉動
- 停止校正

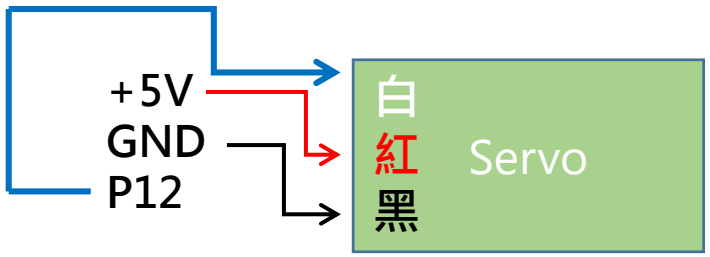


```
#include <Servo.h>

Servo servoLeft;
int servoPin = 12;

void setup() {
  servoLeft.attach(servoPin);
}

void loop() {
servoLeft.writeMicroseconds(1500);
delay(30000);
servoLeft.detach();
}
```

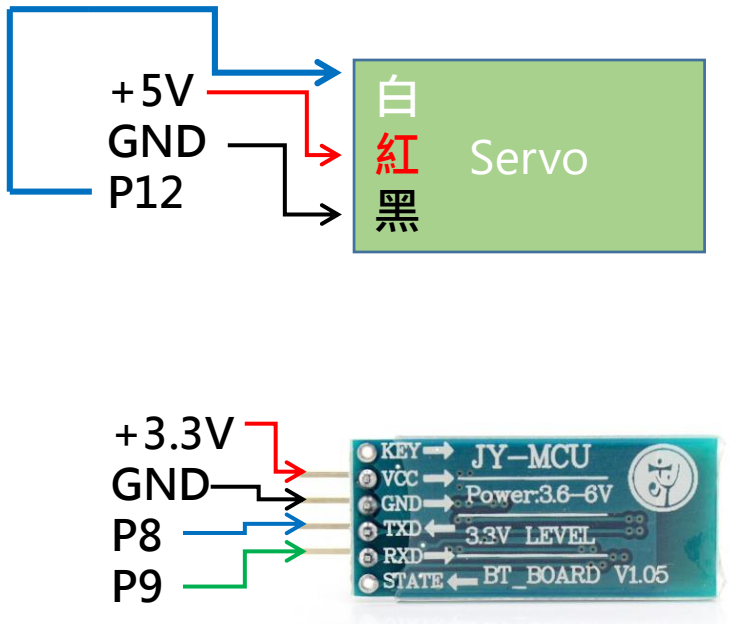


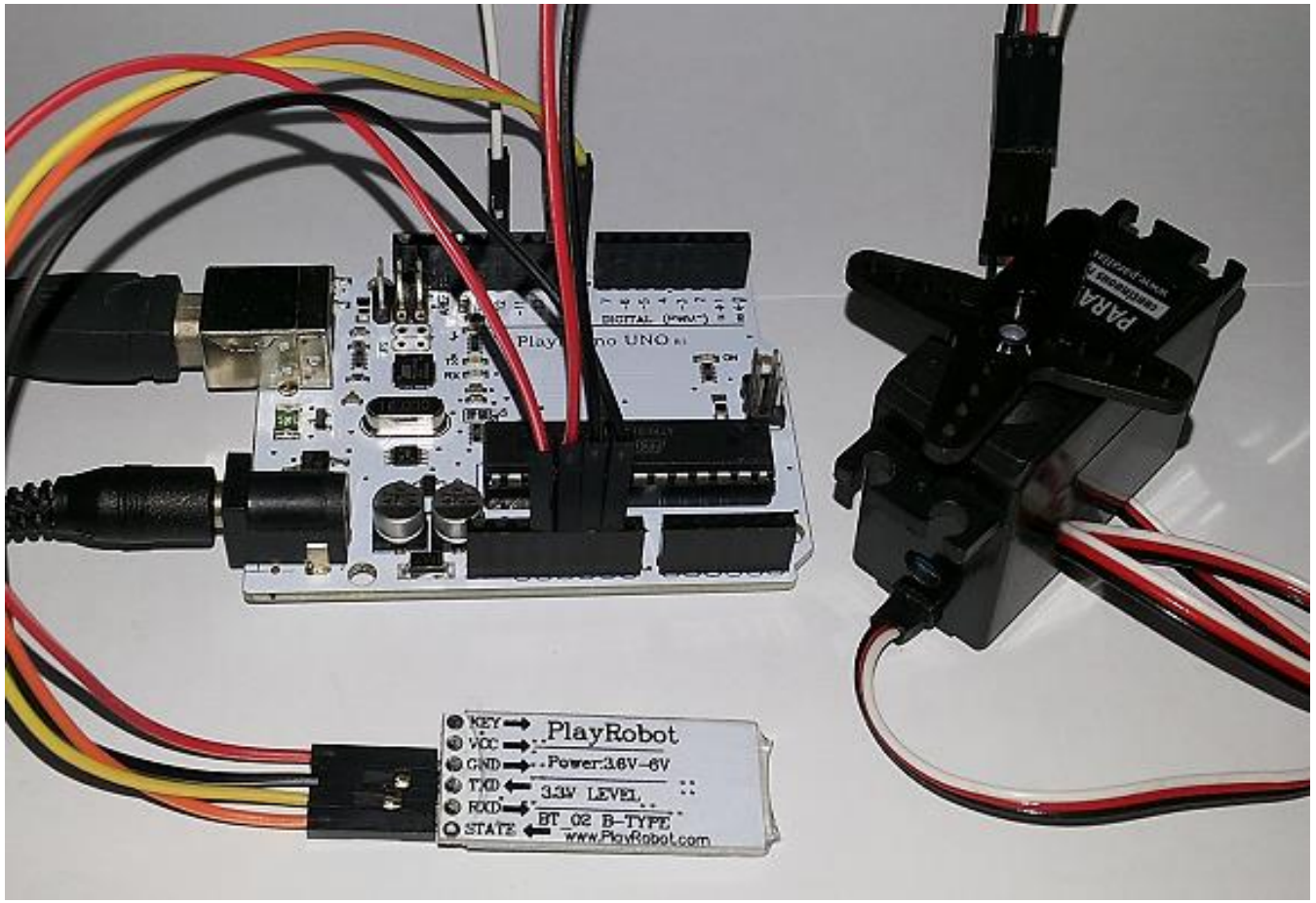
Parallax Continuous Rotation Servo 驅動方式 2/2

```
#include <Servo.h>

Servo servoLeft;
int servoPin = 12;

void setup() {
  servoLeft.attach(servoPin);
}

void loop() {
servoLeft.writeMicroseconds(1300);
delay(3000);
servoLeft.writeMicroseconds(1500);
delay(3000);
servoLeft.writeMicroseconds(1700);
delay(3000);
servoLeft.detach();
}
```

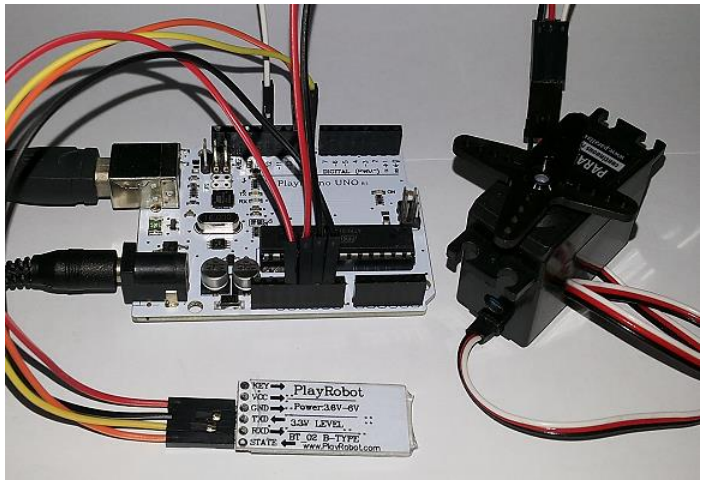


實驗 – 透過藍牙傳輸控制伺服馬達





實驗 – 透過藍牙傳輸控制伺服馬達




```
// 引用程式庫
#include <SoftwareSerial.h>
#include <Servo.h>
Servo servoLeft;
int servoPin = 12;
// 接收腳, 傳送腳
SoftwareSerial BT(8, 9);
char val;  // 儲存接收資料的變數
```

```
void setup() {
  // 設定藍牙模組的連線速率
  BT.begin(9600);
  servoLeft.attach(servoPin);
  servoLeft.writeMicroseconds(1500);
}
```

```
void loop() {
  if (BT.available()) {
    val = BT.read();
    switch(val)
    {
      case '1':
        BT.print(val);
        servoLeft.writeMicroseconds(1300);
      break;
      case '2':
        BT.print(val);
        servoLeft.writeMicroseconds(1500);
      break;
      case '3':
        BT.print(val);
        servoLeft.writeMicroseconds(1700);
      break;
      case '4':
        BT.print(val);
        servoLeft.detach();
      break;
    }
  }
}
```