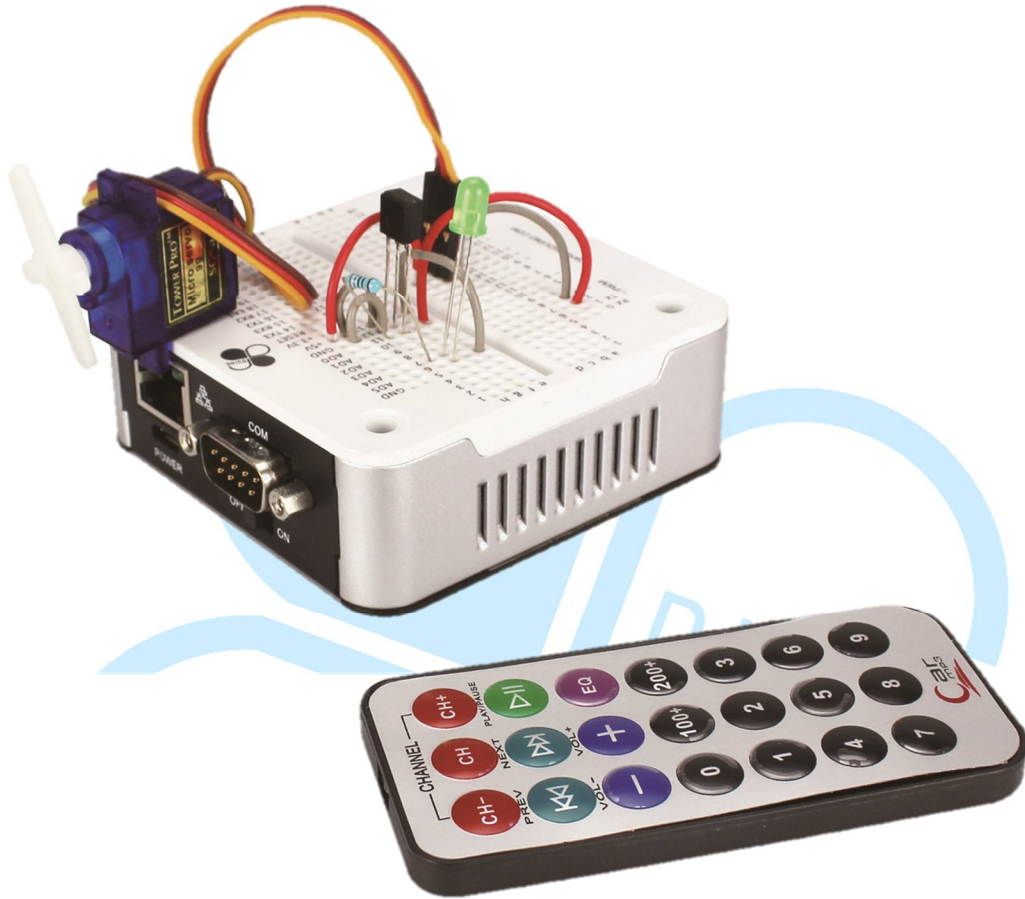


EduCake 紅外線收發功能實作



一、紅外線發射/接收原理與應用介紹

前面章節講解過幾種 86Duino EduCake 可用的通訊方式，像是 UART Serial Port、I²C 等等，不過這些都是以「實體電線連接」的方式通訊的，本篇則另外來介紹一種生活中常見的「無線」通訊方式，也就是「紅外線通訊」，並且使用 86Duino EduCake 來實作紅外線發射、接收等功能，實際動手玩玩看。

紅外線屬於電磁波的其中一個類別，因此得先簡單介紹電磁波的頻譜。一般人眼可見光的波長範圍約在 400nm（紫色）~700nm（紅色）之間。

波長小於紫色，範圍約為 10nm~400nm（頻率大於紫色）的就是常聽到的「紫外線」；而波長大於紅色，範圍約為 700nm~1mm（頻率小於紅色）的就是「紅外線」（或紅外光），也因為英文稱為「Infrared」而常以縮寫「IR」來表示；紫外線（Ultraviolet）則常使用「UV」來表示。紅外線位於電磁波頻譜範圍中如下圖 1 所示：

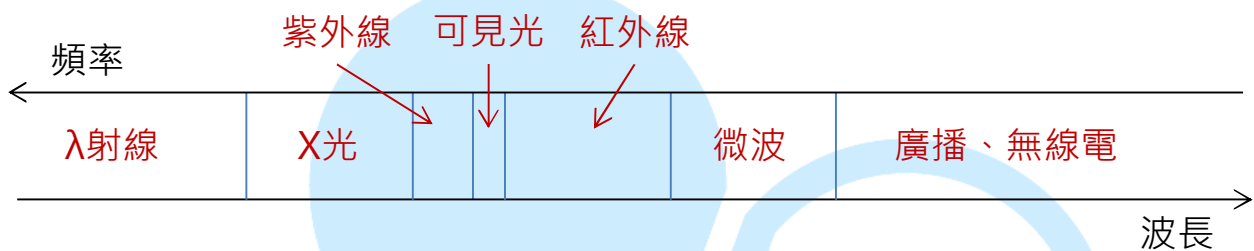


圖 1. 電磁波頻譜簡易示意圖

紅外線頻譜內，還依照可偵測範圍的不同分為近紅外線、短波紅外線、中波紅外線、長波紅外線、遠紅外線等等，如果讀者想知道更多關於電磁波的訊息，可以參考：

http://en.wikipedia.org/wiki/Electromagnetic_radiation

http://en.wikipedia.org/wiki/Visible_spectrum

<http://en.wikipedia.org/wiki/Infrared>

<http://en.wikipedia.org/wiki/Ultraviolet>

雖然紅外線看不到，但生活中其實到處都有著紅外線的存在，例如軍事電影中常看到的「紅外線熱像儀」便是以紅外線感測器接收各種不同溫度物體發出的紅外線，並以不同顏色標示，以便人類從螢幕上觀察溫度的分佈狀況。除了觀察溫度外，紅外線的其他應用例如：衛星或大型望遠鏡觀察外太空的星球、家中冷氣/電視/相機/光碟播放機等電器常見的紅外線遙控器、部分智慧型手機配備的紅外線資料傳輸功能、掌上型遊戲機近距離連線對打功能、人體動作感應燈等等，用途相當廣泛。

本篇文章要介紹的，便是如何實作紅外線無線通訊這項功能。紅外線通訊跟之前介紹過的通訊方式一樣有發射端與接收端裝置。實作時，發射端是由主控電路搭配紅外線發射源，以及接收端主控電路搭配一個紅外線接收器所組成的。紅外線發射器外型就像是一個 LED，只是他會發出特定波長的紅外線電磁波，外型如下圖 2（有些發射器是深藍色的）。

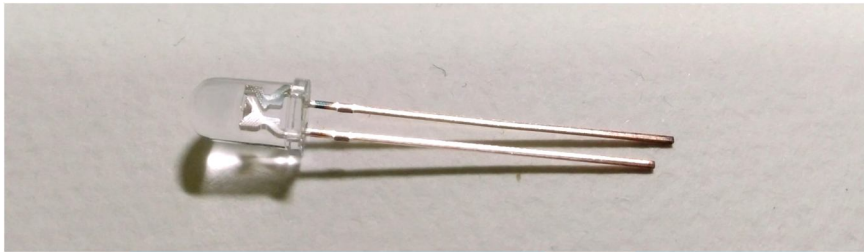


圖 2. 紅外線發射器外觀圖

而紅外線接收器內部結構較為複雜，包含 IR 接收器以及訊號處理電路。接收器整體是一個三腳位的封裝，其中兩腳位是電源、接地，另一腳位是訊號輸出用途，外觀如下圖 3。

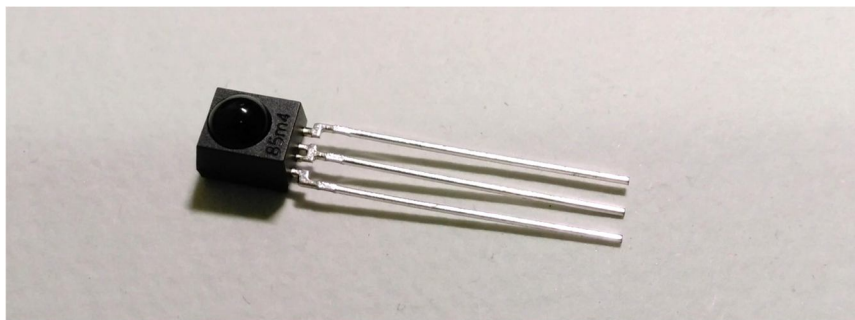


圖 3. 紅外線接收器外觀圖

由於環境中充斥著各種不同來源的紅外線電磁波，為了確保接收端能夠正確收到資料，一般會在傳送端將資料「調變（modulation）」成特定頻率的紅外線訊號。而接收端收到紅外線訊號後，訊號處理電路會進行濾波（僅留下特定頻率範圍的訊號），並針對濾波後的訊號進行「解調（demodulation）」，輸出成為 HIGH 或 LOW 訊號，HIGH/LOW 定義依接收器而定，因此接收端搭配的裝置或處理器便能夠知道傳送端送出的資料是什麼了。紅外線傳輸流程簡易示意圖如下圖 4。

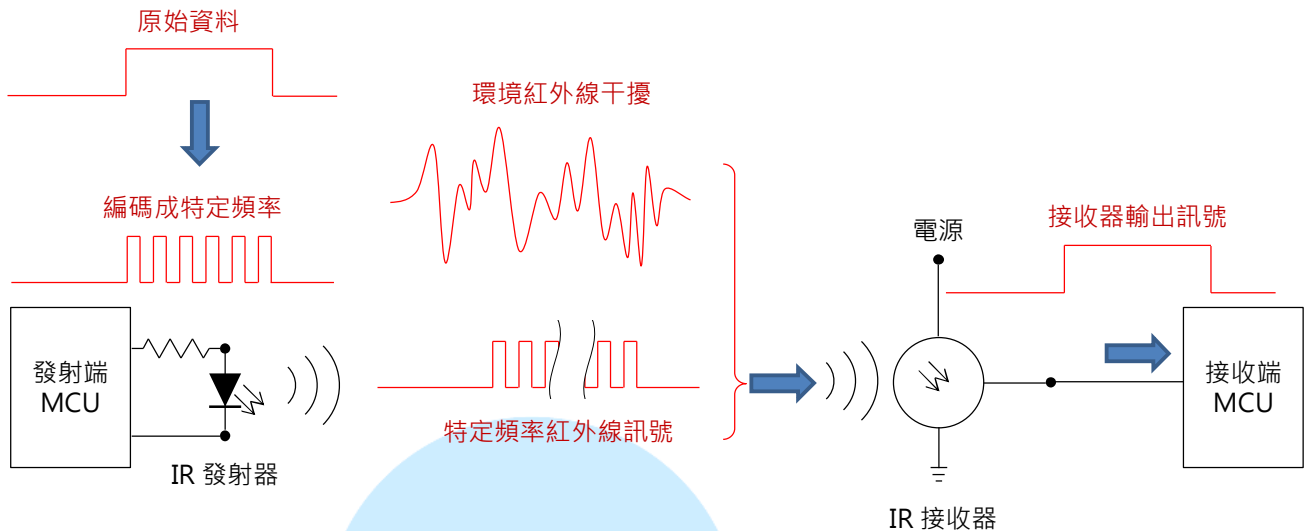


圖 4. 紅外線通訊流程示意圖

上述這種傳送/接收特定紅外線的方式，需要兩端在相同的頻段內，一般常見有 36 kHz、38 kHz、40 kHz、56 kHz 等。須注意這裡提到的「特定頻率」並不是指電磁波光譜的頻率，而是紅外線發射器以特定間隔時間，間歇地發射紅外線的頻率，也稱為「載波頻率」。調變（modulation）與解調（demodulation）便是將資料與載波頻率結合，以及從載波頻率中取出的過程，為無線傳輸常用的方法。另外，由於紅外線發射器比一般 LED 需要略大的電流，因此使用載波頻率間歇式發射紅外線訊號也有避免發射器持續通電過熱的額外優點。

除了以載波頻率將資料編碼以外，不同電器用品，甚至不同廠牌之間，也會有自己的紅外線「通訊協定（protocol）」。由於上述的紅外線訊號在傳送前、接收後都是數字訊號，因此通訊協定為一連串的 HIGH、LOW 電壓組成，依據各廠商的訊號定義不同而有不同的位長度、欄位定義等等。因此，A 廠商的遙控器與 B 廠商的電器之間，即使選用相同波長的紅外線，相同的載波頻率，也不一定能夠相容進行遙控喔。

以一個 SONY 遙控器為例，使用的載波頻率是 40 kHz，遙控器上按鍵被按下後，紅外線發射器會送出不同的位序列，表示對應的 IR 遙控指令，

接收的電器便依據不同的指令執行對應的動作，例如電視機開關電源、調整音量、選取頻道等等。下圖 5 所示為一 SONY 遙控器發射的紅外線指令，讀者可以看到指令共包含了起始訊號與資料訊號，其中 0、1 分別以不同長度的 HIGH/LOW 寬度編碼。另外，當按鍵被持續按著時，依據通訊協定的規定，有的會間隔一段時間重複送出特定數量的相同指令，也有的是一直重複送出指令。因此如果想要遙控某廠牌的電器裝置，就得先瞭解所用的紅外線傳輸規格為何了。

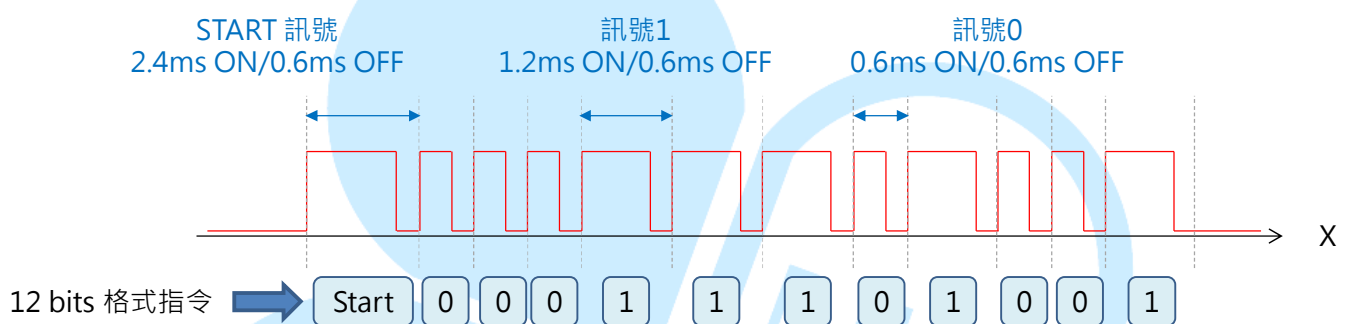


圖 5. 紅外線傳送通訊協定範例

關於 SONY 紅外線通訊協定，可參考：

<http://www.righto.com/2010/03/understanding-sony-ir-remote-codes-lirc.html>

<http://users.telenet.be/davshomepage/sony.htm>

<http://www.cypress.com/?docID=46755>

<http://www.sbprojects.com/knowledge/ir/sirc.php>

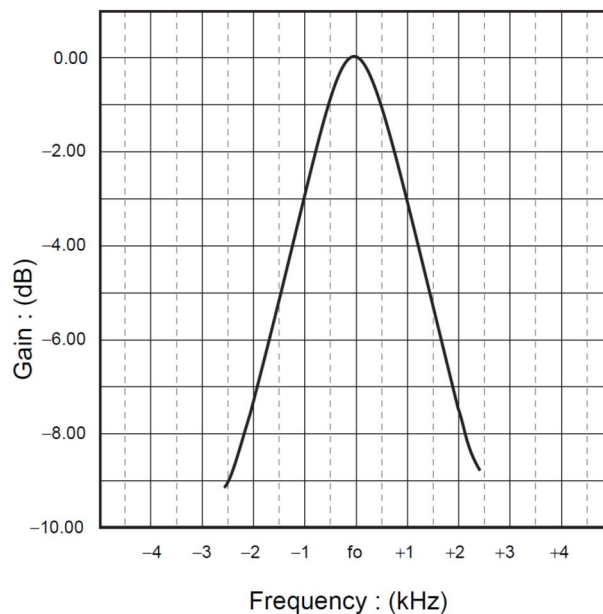
接下來的 86Duino EduCake 實作上，我們選用一般容易買到的 920nm 紅外線發射器，以及筆者在電料行找到的 RPM6938 紅外線接收器來進行功能實作，此接收器的腳位定義如下圖 6：



針腳定義，依序為：
[訊號 GND 電源]

圖 6. 紅外線接收器腳位定義

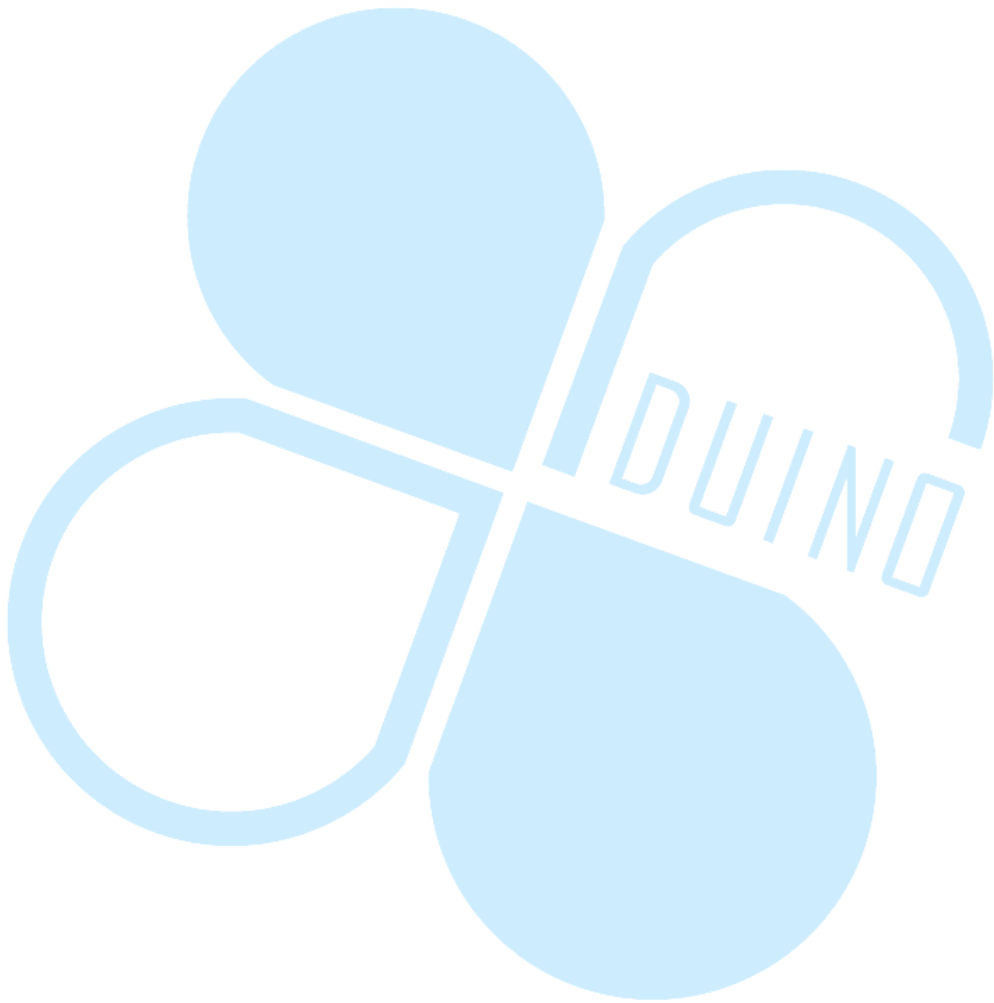
依據 RPM6938 的規格表，電源針腳使用 5V，接收訊號的載波頻率是 37.8kHz，水準有效接收角度約 70 度，垂直約 60 度左右；無接收到訊號時輸出腳為 HIGH，反之為 LOW。須注意雖然規格表上面標示的載波頻率是 37.8kHz，但實際上是在這個頻率「有最佳的接收效果」，如規格表上的圖示，頻率略高/略低時接收的效果會減少，因此頻率稍有誤差也是沒問題的。



接收器的型號眾多，若讀者實作時買的是其他款式紅外線接收器，除了載波頻率外，還得看清楚接收器型號，因為有些接收器電源針腳跟接地針腳是相反的，得查清楚規格才能接線喔。另外也有兩支針腳，長得像深藍色

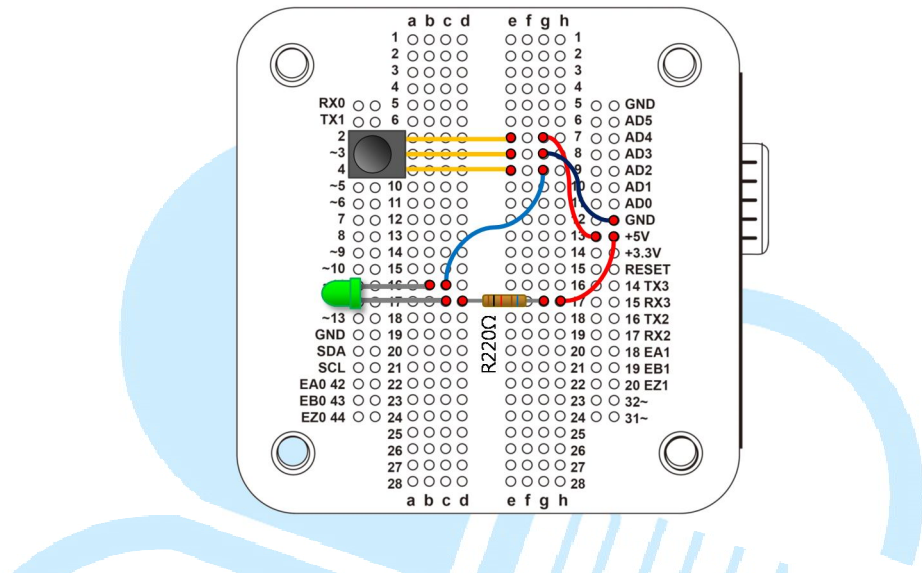
LED 的紅外線接收器，但此類接收器沒有載波頻率的處理功能，須注意別買錯。

紅外線發射器接線方式則跟一般 LED 相同，長腳為正端，短腳為負端。
接下來，就讓我們來練習實作紅外線傳輸的功能吧。



二、 第一個練習 – 測試紅外線接收器

第一個練習，我們暫時不用 EduCake 的程式，先以簡單的方式測試一下紅外線接收器與發射器是否正常，讀者請先依下圖接線：

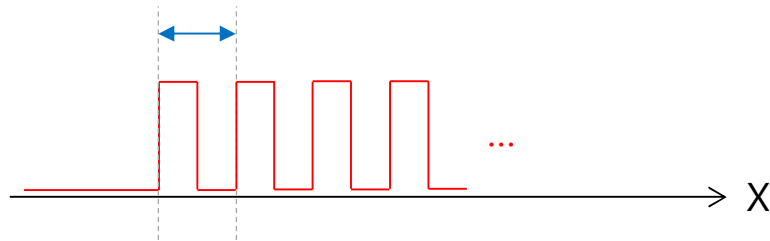


紅外線接收器電源接 5V，接地腳接 GND，輸出訊號腳接 LED 的負端，LED 正端串聯一個 220 歐姆電阻到 5V。由於此紅外線接收器沒接收到訊號時，輸出腳為 HIGH，因此 LED 平常不會亮。接著讀者可以使用手邊的紅外線遙控器，朝著接收器按下任意按鈕，如果遙控器使用的載波頻率大約是 38kHz 左右，就會看到 LED 燈斷斷續續地閃爍囉，也表示手上的紅外線接收器是可以正常運作的。

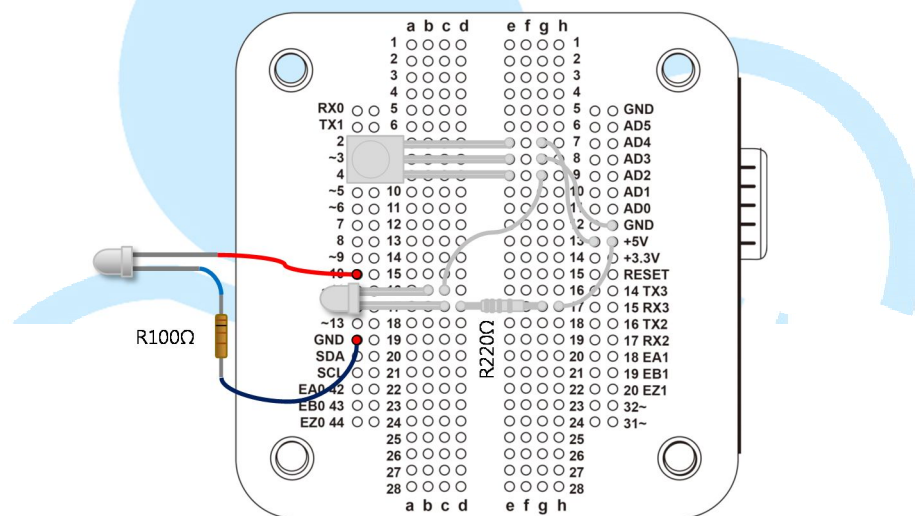
如果手邊沒有紅外線遙控器，只有前面提到的紅外線發射器（長得像 LED 的）零件怎麼測試呢？這就需要使用程式來控制紅外線發射器的閃爍頻率了。

這裡選用的接收器對約 38kHz 的載波頻率有反應，因此需要使用控制器發射如下圖的波形：

週期 = $1/38k$ 秒



要產生這樣的波形，第一種方法可以使用 `digitalWrite()` 來產生，讀者請先依下圖接線：



將紅外線發射器正端接 pin 10，由於發射器所需電流比 LED 大些，因此負端串接一個 100Ω 電阻後接到 GND(而不是 LED 常搭配的 220Ω 電阻)，紅外線接收器的電路不變動。

接著請打開 86Duino Coding IDE，輸入以下程式碼：


```
int IR_pin = 10;

void SendIR( )
{
    for( int i = 0; i < 800; i++ )
    {
        // ON + OFF 26us ~= 38.4kHz
        digitalWrite( IR_pin, HIGH );
        delayMicroseconds( 13 );

        digitalWrite( IR_pin, LOW );
        delayMicroseconds( 13 );
    }
    Serial.println( "IR send." );
}

void setup() {
    Serial.begin( 115200 );

    pinMode( IR_pin, OUTPUT );
}

void loop() {
    SendIR();
    delay( 1000 );
}
```

此程式在「`void SendIR()`」函式裡，使用「`digitalWrite()`」搭配「`delayMicroseconds()`」來產生 PWM 波形，38kHz 波形的週期時間約是 26us 左右，因此使用 LOW/HIGH 各 13us。編譯並上傳程式至 86Duino

EduCake 後，紅外線發射器每秒都會送出一個頻率約 38.4kHz 的波形，就可以觀察到紅外線接收器輸出腳位所接的 LED 閃爍囉。

第一種這是簡易的測試方式，若使用硬體來實作所需的 PWM 波形，頻率控制也會更精確。86Duino EduCake 有提供一個好用的函式庫稱為「TimerOne」，此函式庫可用來設定 86Duino EduCake 內 CPU 的 32 位元數目器功能，並控制特定腳位輸出 PWM 訊號。由於是硬體在定時觸發電壓 HIGH/LOW，因此時序控制上會比程式控制腳位元電壓切換要更精確。功能與前面測試程式相同的程式碼如下：

```
#include "TimerOne.h"
int IR_pin = 10;
void SendIR()
{
    Timer1.pwm( IR_pin, 512, 26 );// pin, duty (512=50%),
period(us)
    delay( 20 );
    Timer1.disablePwm( IR_pin );
    delay( 20 );
    Serial.println( "IR send." );
}
void setup() {
    Serial.begin( 115200 );
    pinMode( IR_pin, OUTPUT );
    Timer1.initialize( 26 );// TimerOne initialize, period(us)
}
void loop() {
    SendIR();
    delay( 1000 );
}
```

想要使用 TimerOne 函式庫，程式一開始要先使用語法「`#include "TimerOne.h"`」，接著在 `setup()` 階段使用「`Timer1.initialize(period);`」做 TimerOne 對象初始化，其中參數欄位 `period` 為計時器的週期，單位為 `microsecond`。

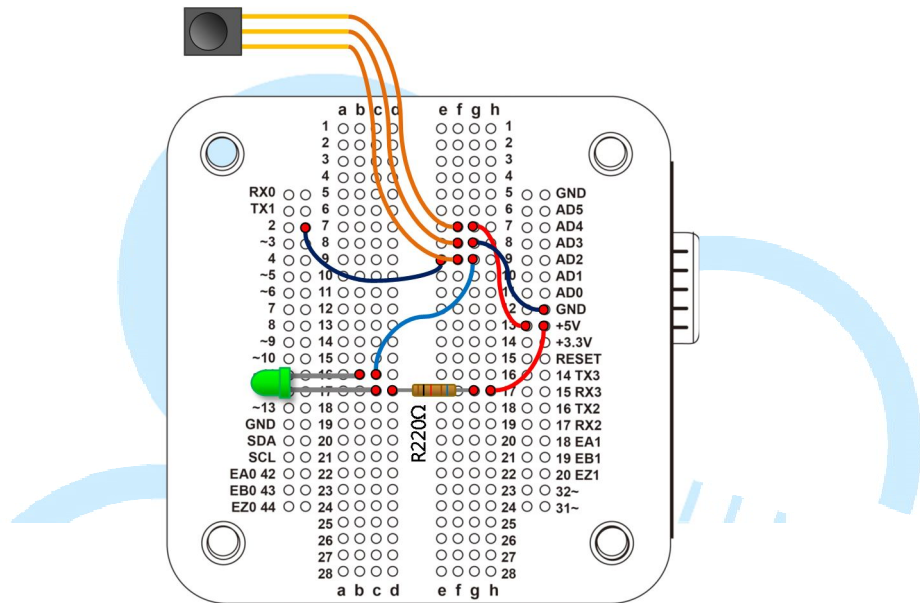
「`void SendIR()`」函式裡，則使用「`Timer1.pwm(IR_pin, 512, 26);`」語法來設定 IR 腳位啟動 PWM 功能，512 表示 50% 的 `duty`，`period` 為 26 μs ，持續時間為 20ms。「`Timer1.disablePwm(IR_pin);`」語法則用來關掉紅外線發射器腳位的 PWM 輸出。如此也可以達到測試紅外線發射器、接收器的目的喔。

三、 第二個練習 - 認識紅外線通訊格式

瞭解紅外線接收/發射的基本原理後，接著就要進展到較實用的階段了。

這個練習題我們來認識一下實際電器用品傳輸的「通訊協定」長甚麼樣子。

讀者請依下圖接線：



此電路保留之前的接線，但額外將紅外線接收器輸出腳位元接到 86Duino EduCake 的數位腳位元 2，以便程式讀取電壓狀態。完成接線後，請打開 86Duino Coding IDE，輸入以下程式碼：

```
int IR_rec_pin = 2;// IR 接收器輸出腳位
int IRstate = LOW;// IR 接收器輸出腳位元狀態
int IRstate_last = LOW;// IR 接收器輸出腳位元狀態
long int time_last = 0;// 紀錄上一次 IRstate 變化的時間

boolean isIdle = true;// 是否在等待 IR 訊號模式
const long int durationMax = 10000;// 一段時間沒變化就進入等待 IR 訊號狀態，單位 us
```

```
const long int durationMin = 400;// 電壓狀態不變的最小持續時間，單位 us
void IR_rec_Check()
{
    IRstate = digitalRead( IR_rec_pin );// 讀取腳位元狀態

    if( IRstate != IRstate_last ){// 這次跟上次腳位元狀態不同

        long int timeNow = micros( );// 取得目前時間
        long int dT = timeNow - time_last;// 上一次腳位元狀態變化經過的時間

        if( dT >= durationMax && !isIdle ){
            isIdle = true;
            Serial.println( "Idling...\n" );
        }
        else if( dT < durationMax && dT > 400 ){
            isIdle = false;
            Serial.print( IRstate == HIGH ? dT : dT ); Serial.print( "
");
        }

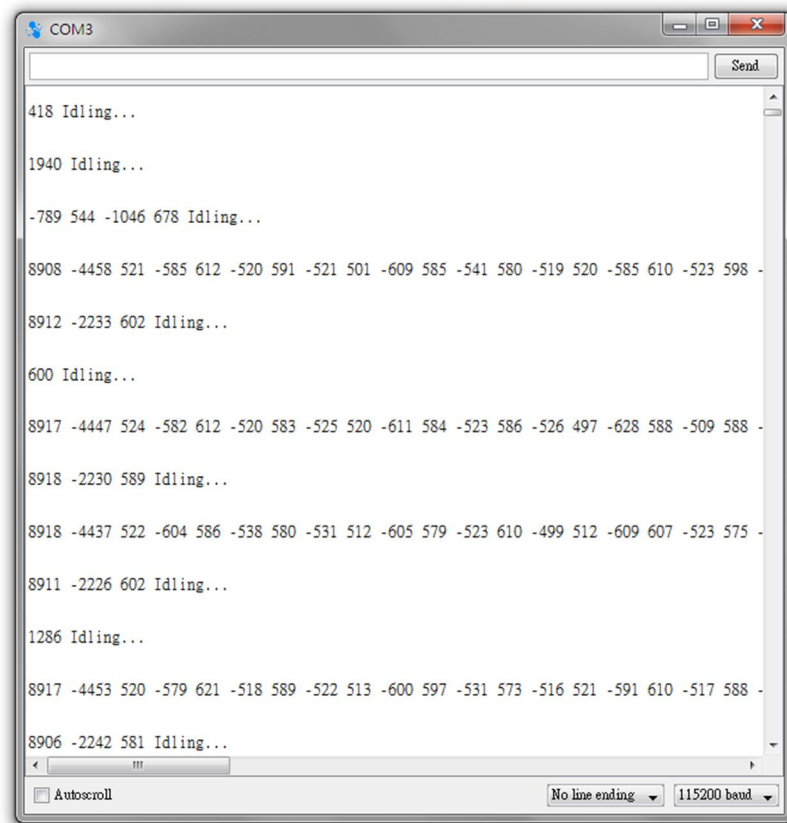
        // 記錄此次時間
        time_last = timeNow;
    }
    // 記錄此次狀態
    IRstate_last = IRstate;
}

void setup() {
    Serial.begin( 115200 );
    pinMode( IR_rec_pin, INPUT );// 設定針腳 I/O 模式
    IRstate = digitalRead( IR_rec_pin );// 取得腳位元狀態初始值
    IRstate_last = IRstate;
}
```



```
void loop() {  
  
    IR_rec_Check();  
    delayMicroseconds( 20);  
}
```

編譯並上傳程式後，請打開 Serial Monitor，使用手邊可以相容所這個紅外線接收器的遙控器（可以先使用前面練習提供的方法測試載波頻率是不是符合），朝著接收器按幾下按鈕看看，Serial Monitor 上便會顯示出一系列的數位，正值代表輸出腳位元電壓 HIGH 的持續時間，負值則是電壓 LOW 的持續時間，如下圖：



這裡須注意，當紅外線發射器「有發射」時，接收器收到紅外線是輸出 LOW，反之為 HIGH，因此這裡讀到的正值實際上是「沒有」收到紅外線的時候喔。

這個程式碼會每隔 $20\ \mu\text{s}$ ，呼叫「`IR_rec_Check()`」函式，使用「`digitalRead()`」讀取接收器輸出腳位元的電壓狀態，當電壓有變化時，使用「`micros()`」語法記錄下當時的毫秒時間，以便跟下一次狀態變化時間點相減取得持續時間長度。程式碼裡面設定了 `durationMax`、`durationMin` 做為過濾條件，大於 `durationMax` 則當作一段時間沒接收到，進入 `idle` 狀態；反之小於 `durationMin` 則當作雜訊不予理會。使用 μs 作為取樣間隔，是因為紅外線接收器會將持續數百 μs （依規格）的紅外線訊號判斷為 `HIGH` 或 `LOW`，低於的當作雜訊濾除，因此設定數十 μs 當作取樣頻率，取樣間隔若太長，偵測出的持續時間會較不精準。

讀者會注意到，即使沒有按下遙控器時，接收器也可能會受環境紅外線影響，輸出持續時間長度不等的 `HIGH/LOW` 訊號，因此為了實際傳輸的正確，就需要「通訊協定」的說明瞭。「通訊協定」是傳輸/接收兩端預先定義好的一連串信號順序定義，包含訊號開頭、結尾、資料長度等等，跟之前提過的 `Serial` 傳輸概念很類似。

筆者在這裡測試使用的是車用 MP3 遙控器，外觀如下：



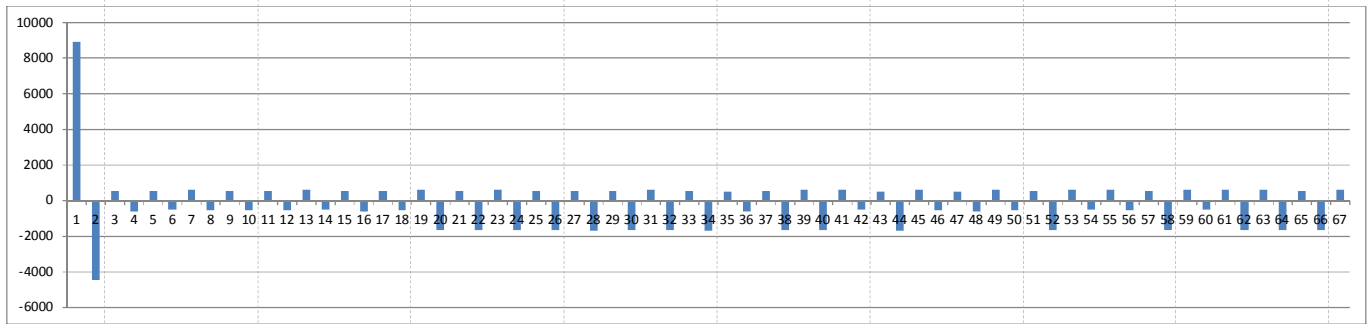
以這個遙控器當例子，每次按下按鈕 0，都會收到持續時間約為「8883
-4487 524 -599 591 -525 594 -516 522 -610 580 -527 596 -513 514
-600 595 -524 593 -1632 596 -1630 516 -1697 606 -1631 602 -1625
519 -1707 609 -1630 585 -1630 525 -595 630 -1594 523 -1704 610

-509 509 -1717 606 -510 517 -609 615 -503 582 -1630 594 -534 591

-506 520 -1707 607 -509 514 -1715 602 -814 1405 -1632 518」的資料

序列，時間長度單位為 μs ，這不太方便觀察，我們使用 Excel 來畫成圖表

如下：



去頭去尾剛好剩下 64 個資料，如果按下其他按鈕則會得到其他資料序列。從圖上可以看到，這個遙控器使用 HIGH 時間約 $600 \mu s$ 、LOW 時間約 $600 \mu s$ 或 $1600 \mu s$ 的組合當作資料位元。由於外界幹擾不容易整串剛好符合這個格式，其他廠商的紅外線編碼方式也會不同，因此當傳送/接收兩端以特定的資料序列溝通時，就能將外界環境的幹擾影響儘量降低。讀者也可以使用手邊各種遙控器試試看會收到怎樣的資料組合喔。

四、 第三個練習-1 – IRremote 函式庫使用介紹(接收)

經過前一章節的練習，紅外線的通訊協定這麼多種，該如何實際應用呢？在 86Duino EduCake 裡面有提供移植自 Ken Shirriff 的「IRremote」函式庫。這個函式庫可以用通訊協定來傳送與接收紅外線資料，並且支援了 NEC、Sony SIRC、Philips RC5、Philips RC6、Sharp、Panasonic、JVC、Sanyo、Mitsubishi 等協議格式，並可輸出原始資料供觀察。這個練習就讓我們使用 IRremote 函式庫來接收紅外線訊號，並且介紹一些實用的重要功能吧。

86Duino EduCake 電路維持與練習 2 一樣的接線方式，接著請讀者打開 86Duino Coding IDE(注意需使用 104 版本以上才有 IRremote 函式庫)，輸入以下程式碼：

```
#include <IRremote.h>
int IR_rec_pin = 2;// IR 接收器輸出腳位

IRrecv IRrecv( IR_rec_pin );// IRremote 函式庫接收用對象

decode_results results;// 解碼結果存放資料用

void Print_IRdecodeResult( decode_results &decodeResults )//
印出解碼成功的訊息以便觀察
{
    int dataLength = decodeResults.rawlen;

    switch( decodeResults.decode_type )
    {
        case NEC:
```

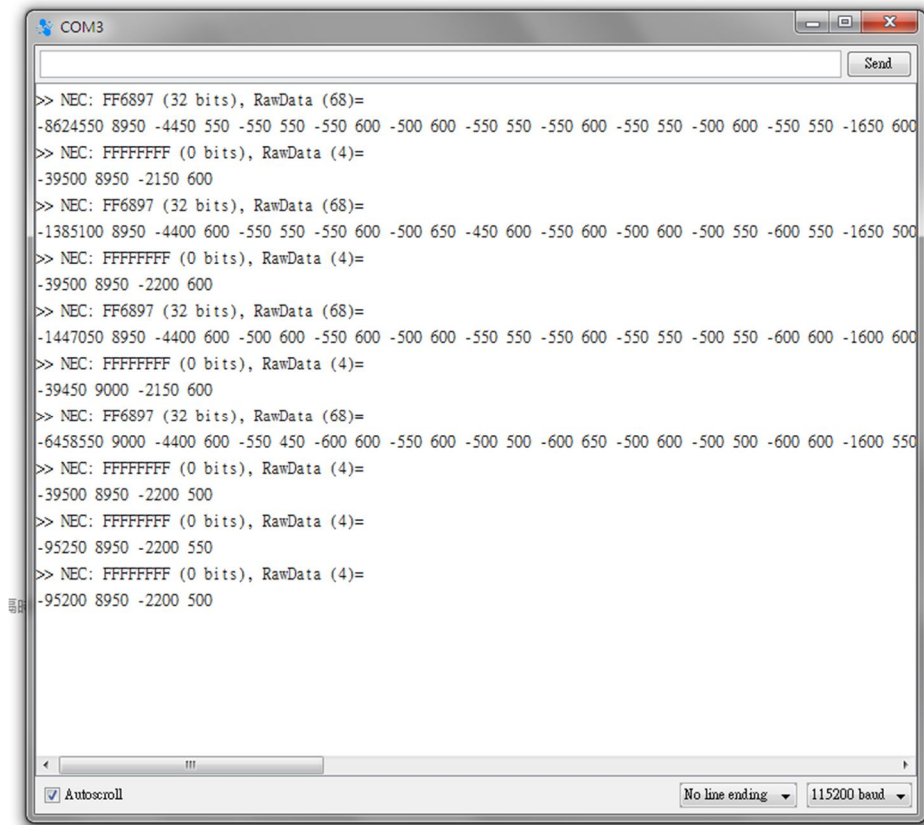
```
Serial.print( ">> NEC:\t" );  
    break;  
  
case SONY:  
    Serial.print( ">> SONY:\t");  
    break;  
  
case RC5:  
    Serial.print( ">> RC5:\t");  
    break;  
  
case RC6:  
    Serial.print( ">> RC6:\t");  
    break;  
  
case DISH:  
    Serial.print( ">> DISH:\t");  
    break;  
  
case SHARP:  
    Serial.print( ">> SHARP:\t");  
    break;  
  
case SANYO:  
    Serial.print( ">> SANYO:\t" );  
    break;  
  
case MITSUBISHI:  
    Serial.print( ">> MITSUBISHI:\t" );  
    break;  
  
case PANASONIC:  
    Serial.print( ">> PANASONIC(addr=\t" );
```



```
Serial.print( results.panasonicAddress );  
    Serial.print( "):\t" );  
    break;  
  
    case JVC:  
        Serial.print( ">> JVC:\t" );  
        break;  
  
    case UNKNOWN:  
        Serial.print( ">> Unknown:\t" );  
        break;  
  
    default:  
        break;  
}  
  
Serial.print( decodeResults.value, HEX );// 通訊協定資料欄  
位，以 16 進位元顯示  
Serial.print( " (" );  
Serial.print( decodeResults.bits, DEC );// 總共收到幾位元資料  
Serial.print( " bits), " );  
  
Serial.print( "RawData (" );  
Serial.print( dataLength, DEC );  
Serial.println( ")= " );  
  
// 印出原始資料序列  
for ( int i = 0; i < dataLength; i++ ) {  
    int data = decodeResults.rawbuf[i] * USECPERTICK;  
    // buf 存放取樣個數，以及每次取樣間隔時間 USECPERTICK  
    if ( ( i % 2 ) == 1 ) { // 偶數存放的是 HIGH  
        Serial.print( data, DEC );// HIGH  
    }  
}
```

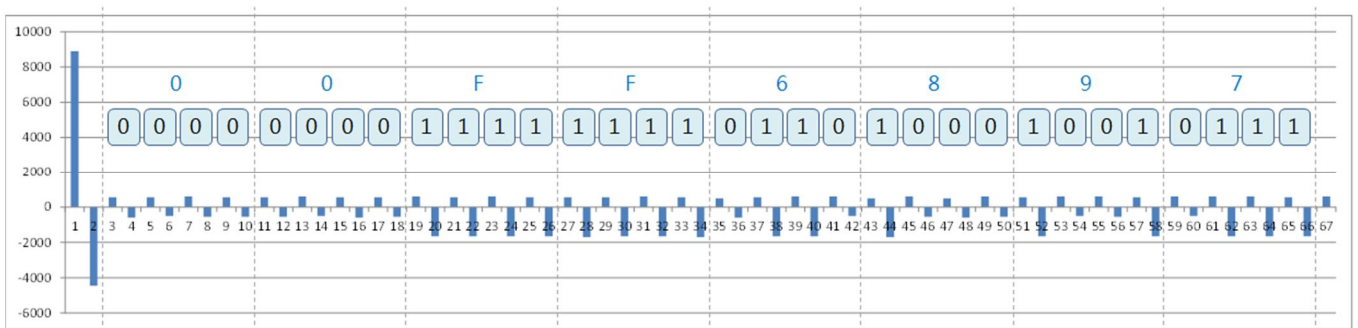
```
else {  
    Serial.print( -data, DEC );// LOW  
}  
Serial.print( " " );  
}  
Serial.println( );  
}  
  
void setup( )  
{  
    Serial.begin( 115200 );  
    IRrecver.enableIRIn( );// 初始化接收對象  
}  
  
void loop() {  
    if ( IRrecver.decode( &results ) )  
    {  
        Print_IRdecodeResult( results );  
        IRrecver.resume( );// 解碼完需要呼叫這一行，才能再繼續接收後續資料  
    }  
}
```

編譯並上傳程式後，請打開 **Serial Monitor**，接著拿遙控器朝紅外線接收器按幾下，以筆者前面練習所用的車用 **MP3** 遙控器而言，會出現下圖畫面：



```
>> NEC: FF6897 (32 bits), RawData (68)=  
-8624550 8950 -4450 550 -550 550 -550 600 -500 600 -550 550 -550 600 -550 550 -500 600 -550 550 -1650 600  
>> NEC: FFFFFFFF (0 bits), RawData (4)=  
-39500 8950 -2150 600  
>> NEC: FF6897 (32 bits), RawData (68)=  
-1385100 8950 -4400 600 -550 550 -550 600 -500 650 -450 600 -550 600 -500 600 -500 550 -600 550 -1650 500  
>> NEC: FFFFFFFF (0 bits), RawData (4)=  
-39500 8950 -2200 600  
>> NEC: FF6897 (32 bits), RawData (68)=  
-1447050 8950 -4400 600 -500 600 -550 600 -500 600 -550 550 -550 600 -550 550 -500 550 -600 600 -1600 600  
>> NEC: FFFFFFFF (0 bits), RawData (4)=  
-39450 9000 -2150 600  
>> NEC: FF6897 (32 bits), RawData (68)=  
-6458550 9000 -4400 600 -550 450 -600 600 -550 600 -500 500 -600 650 -500 600 -500 500 -600 600 -1600 550  
>> NEC: FFFFFFFF (0 bits), RawData (4)=  
-39500 8950 -2200 500  
>> NEC: FFFFFFFF (0 bits), RawData (4)=  
-95250 8950 -2200 550  
>> NEC: FFFFFFFF (0 bits), RawData (4)=  
-95200 8950 -2200 500
```

當這個遙控器按下按鈕 0 時，IRremote 函式庫會進行接收並解碼，此函式庫方便之處在於可以辨識收到的資料序列屬於何種編碼方式，以及共收到了多長的資料。以筆者使用的遙控器為例，IRremote 函式庫解碼出來的通訊協定是 NEC 格式，基本上原始資料的時間序列長度跟練習 2 是一樣的，使用 Excel 畫波形圖如下：



以 NEC 通訊協定的接收器輸出腳位元電壓來說：

- λ 開始訊號：為約「9ms 的 HIGH 與 4.5ms 的 LOW」組合。
- λ 資料邏輯 0：為約「560 μ s 的 HIGH 與 560 μ s 的 LOW」組合。
- λ 資料邏輯 1：為約「560 μ s 的 HIGH 與 1690 μ s 的 LOW」組合。
- λ 重複指令：為約「9ms 的 HIGH 與 2.25ms 的 LOW 和 560 μ s 的 HIGH」組合。

此遙控器按鍵 0 發出的資料，是對應到 0X00FF6897 的編碼，64 個 HIGH/LOW 剛好是 32 個邏輯位元，共 4 個 byte，其中前 16 個位為裝置位址編碼，後 16 個位元為指令編碼。

另外，若持續按著某個按鈕，在收到第一次的資料編碼後，後面收到的則是「重複指令」FFFFFFFF，約每 110ms 會重複一次，例如持續按著音量+按鈕，裝置就可以依此重複指令持續增加音量了。因此即使房間裡一樣都是 NEC 的裝置，也可以根據辨識不同的位元址、指令而做對應動作，不會有衝突發生，這就是通訊協定的妙用。

要使用 IRremote 函式庫，此程式一開始須引用「[IRremote.h](#)」，然後使用「[IRrecv IRrecv\(IR_rec_pin\);](#)」語法宣告一個紅外線接收器物件，傳入的參數為接收器輸出腳位號碼。「[decode_results results](#)」為一個 class 資料結構，裡面存放了：

- λ decode_type：標示通訊協定編碼類型。
- λ panasonicAddress：Panasonic 協議專用的位址欄位。

- λ value：通訊協定資料欄位數值。
- λ bits：數據序列總位元個數。
- λ unsigned int *rawbuf：原始 HIGH/LOW 取樣資料。
- λ rawlen：取樣資料個數。

setup()階段則需使用「[IRrecver.enableIRIn\(\)](#)」執行紅外線接收器物件的初始化，接著在 loop()裡面定期呼叫「[IRrecver.decode\(&results\)](#)」，此函式會掃描目前接收到的紅外線資料，並檢查編碼，如果解碼成功，就將解碼後的結果放入 results 變數中，並回傳 true；反之解碼不成功或還未收完資料則回傳 false。當回傳 true 時，便可以使用「[Print_IRdecodeResult\(\)](#)」函式，印出解碼成功的各種訊息來觀察囉。

與練習 2 相較之下，此範常式可以讓讀者進一步看看手邊各種遙控器是甚麼通訊協定，以及按鈕按下後會傳出甚麼指令資料囉。

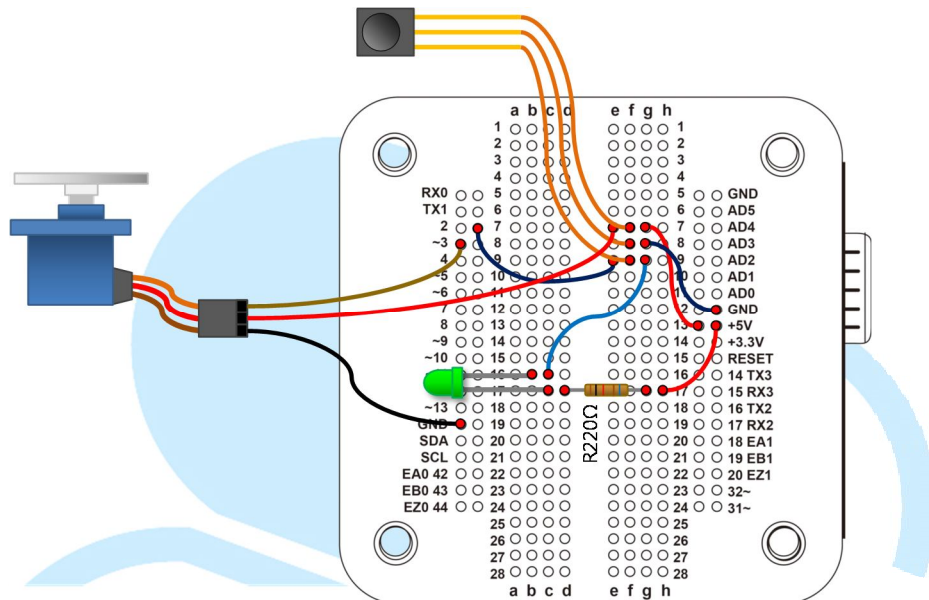
關於 NEC 紅外線通訊協定，也可以參考：

<http://mcudiy.blogspot.tw/2010/11/22-irinfrared-nec-protocol.html>

<http://www.sbprojects.com/knowledge/ir/nec.php>

五、 第三個練習-2 – IRremote 函式庫使用介紹(接收)

這個練習將前一個稍微變複雜些，利用手邊的遙控器讓 EduCake 可以「動起來」。讀者請依下圖接線：



接好線後，在 86Duino Coding IDE 輸入以下程式碼：

```
#include <IRremote.h>
#include <Servo.h>

int IR_rec_pin = 2;// IR 接收器輸出腳位
int servo_pin = 3;// Servo 輸出腳位

IRrecv IRrecver(IR_rec_pin);// IRremote 函式庫接收用對象
decode_results results;// 解碼結果存放資料用

Servo servo_0;// Servo 物件
typedef enum
{
    DIR_NONE = 0,
    DIR_LEFT,
```

```
DIR_RIGHT
} ServoDir;// 轉動方向定義
int servoDir = DIR_NONE;// Servo 轉動方向
unsigned int ServoPosition = 1500;// Servo 角度

void Print_IRdecodeResult( decode_results &decodeResults )//
印出解碼成功的訊息以便觀察
{
    if( decodeResults.decode_type == NEC )
    {
        switch( decodeResults.value )
        {
            case 0x00FFA25D:// CH- button
                servoDir = DIR_LEFT;
                ServoPosition += 50;
                break;

            case 0x00FF629D:// CH button
                servoDir = DIR_NONE;
                ServoPosition = 1500;
                servo_0.writeMicroseconds(ServoPosition);// 置中
                break;

            case 0x00FFE21D:// CH+ button
                servoDir = DIR_RIGHT;
                ServoPosition -= 50;
                break;

            case 0xFFFFFFFF:// Repeat
                if( servoDir == DIR_RIGHT )
                { ServoPosition -= 50; }
                else if( servoDir == DIR_LEFT )
                { ServoPosition += 50; }
```

```
ServoPosition = constrain( ServoPosition, 1100, 1900 );// 限制
在安全範圍

servo_0.writeMicroseconds( ServoPosition );// 控制角
度

Serial.print( "ServoPosition = " );
Serial.println( ServoPosition );
break;

default:
break;
}
}
}

void setup()
{
  Serial.begin( 115200 );
  IRrecver.enableIRIn( );// 初始化接收對象
  servo_0.attach( servo_pin );// 設定 Servo 連接腳
}

void loop() {
  if ( IRrecver.decode( &results ) )
  {
    Print_IRdecodeResult( results );
    IRrecver.resume( );// 解碼完需要呼叫這一行，才能再繼續接
收後續資料
  }
}
```

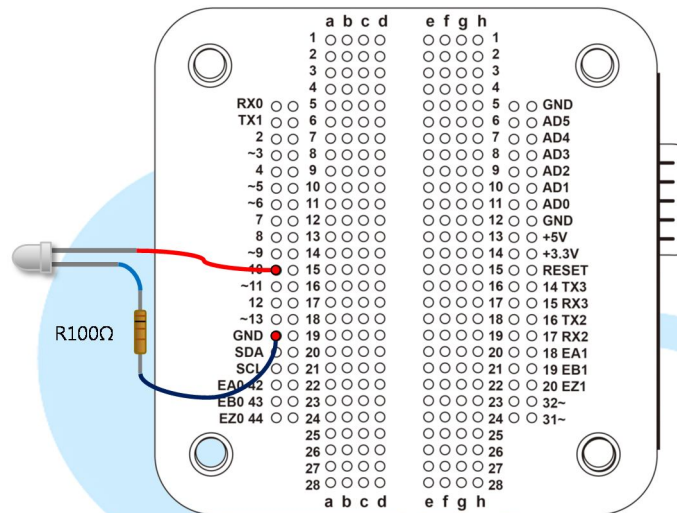
編譯並上傳成功後，便可以用遙控器按鈕控制 EduCake 連接的 RC Servo 左右轉囉。整體程式流程跟前一個練習是一樣的，主要在

「`Print_IRdecodeResult()`」函式內做了改變。由於筆者使用的遙控器，「CH-」按鈕對應的紅外線協定數值為 `0x00FFA25D`，「CH」按鈕為 `0x00FF629D`，「CH+」按鈕為 `0x00FFE21D`，持續壓著的重複碼為 `0xFFFFFFFF`，因此 `switch-case` 語法這邊使用了這幾個數值，且必須為 NEC 編碼才做對應的 `Servo` 控制動作，以減低接收錯誤的影響。如果讀者使用了其他遙控器，由於通訊協定不同、按鈕對應數值不同，就需要修改這些地方才能正常運作喔。



六、 第三個練習-3 – IRremote 函式庫使用介紹(發射)

實作過 IRremote 函式庫的接收功能後，接下來這個練習來實作發射紅外線指令的功能。請依下圖電路接線



接著請打開 86Duino Coding IDE，輸入以下程式碼：

```
#include <IRremote.h>

int ID_send_pin = 10; // IR 接收器輸出腳位

IRsend IR_send; // IRremote 函式庫傳送用物件

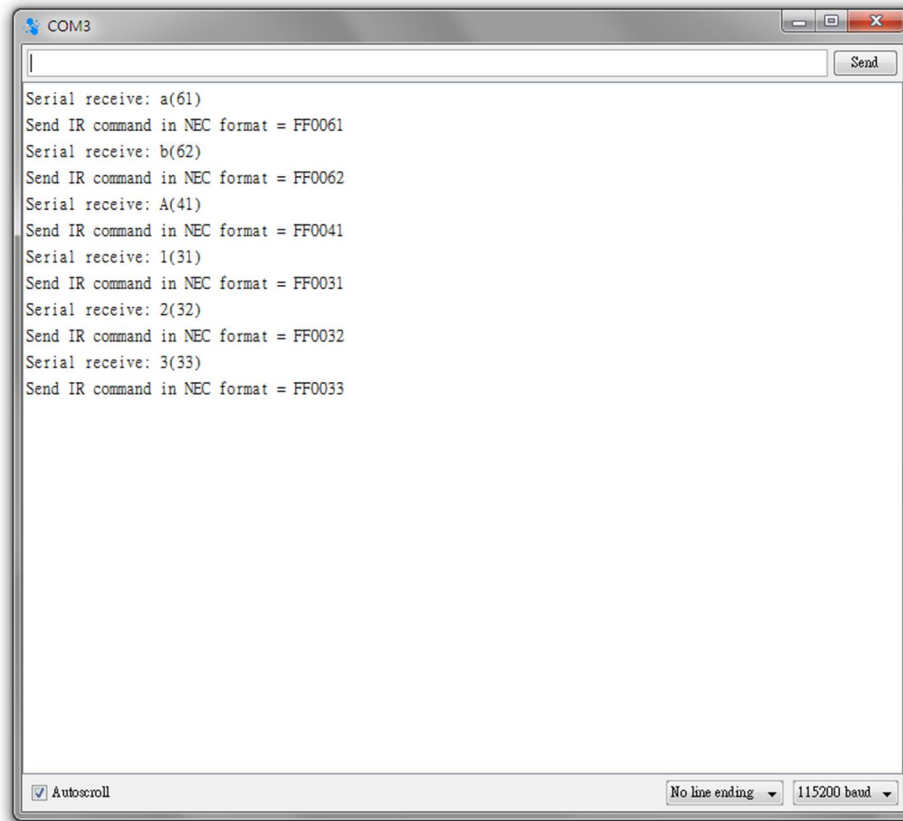
void setup()
{
    Serial.begin( 115200 );

    IR_send.outPin( ID_send_pin ); // 注意 86Duino EduCake 預
    設的 IR 發射器 PWM 腳位為 pin 10
}

void loop() {
```

```
if ( Serial.available() ) {  
  
    char data = Serial.read();  
    //unsigned long cmd = 0x00FF1234;// Addr = 00FF, Data  
    = 1234  
    unsigned long cmd = 0x0;// Addr = 00FF, Data = ?  
    unsigned long DeviceAddr = 0x00FF;  
    // 使用 bitwise or , 將 Serial port 收到的字元編碼合進 IR 指  
    令中  
    cmd = ( DeviceAddr<<16 ) | (unsigned long)data;  
    IR_send.sendNEC( cmd, 32 );// send NEC code format  
    (command, data bits)  
  
    Serial.print( "Serial receive: " );  
    Serial.print( data );// char  
    Serial.print( "(" );  
    Serial.print( data, HEX );  
    Serial.println( ")" );  
  
    Serial.print( "Send IR command in NEC format = " );  
    Serial.println( cmd, HEX );  
}  
    delay( 100 );  
}
```

編譯並上傳程式後，請打開 **Serial Monitor**，然後輸入一些字元，此程式便會將字元的 **ASCII** 編碼加入紅外線通訊協定的指令欄位元，並且在 **Serial Monitor** 顯示資訊如下圖：



此程式在一開始也需引用「[IRremote.h](#)」，然後以「[IRsend IR_send](#)」宣告一個傳送用的物件，接著在 `setup()` 階段使用「[IR_send.outPin\(ID_send_pin\)](#)」指定紅外線發射器街的腳位。注意這個腳位必須為 EduCake 上面有標示「~」的位置，表示此腳位可使用 PWM 輸出。如果沒有使用「[outPin\(\)](#)」指定輸出腳的話，86Duino 的 IRremote 函式庫內是默認為數位腳 10 號。

（在 86Duino Coding IDE 安裝路徑內的 `\hardware\86duino\x86\libraries\IRremote\IRremote.h` 檔案中，有定義 [#define TIMER_PWM_PIN 10](#)）。

在 `loop()` 迴圈內，使用了「[Serial.available\(\)](#)」偵測使用者由 Serial Monitor 傳送的字元符號，並存在「[char data](#)」變數中。傳送的通訊協定

長度，這裡使用與前面練習使用的車用 MP3 遙控器一致，為 32 位，「`cmd = (DeviceAddr<<16) | (unsigned long)data;`」語法用在將位址資料放在左側 16 個位位置，並將資料放於右側 16 個位位置，讀者練習時也可以將位址、資料換成自己想要的數值試試看，不過須注意資料的位元長度喔。

得到欲送出的指令後，使用 IR_send 物件的函式「`IR_send.sendNEC(unsigned long command, int bits)`」來送出 NEC 編碼的指令，由於這個練習的指令資料為 32 位元元，因此 sendNEC 的位長度欄位填入 32。若讀者自行定義了其他長度的指令，這裡也須作相對應的變化。

由於各廠商定義的邏輯位元組合不同，指令格式或載波頻率也不同，因此除了 sendNEC 以外，IRremote 函式庫也提供了其他像是：`sendSony`、`sendRC5`、`sendRC6`、`sendDISH`、`sendSharp`、`sendPanasonic`、`sendJVC` 等函式，使用方法則跟 sendNEC 相同。

另外，若是讀者想定義自己的紅外線通訊協定邏輯組合、載波頻率，而不想使用任何現成廠商的定義，IRremote 也提供了「`sendRaw(unsigned int buf[], int len, int khz)`」這個函式。

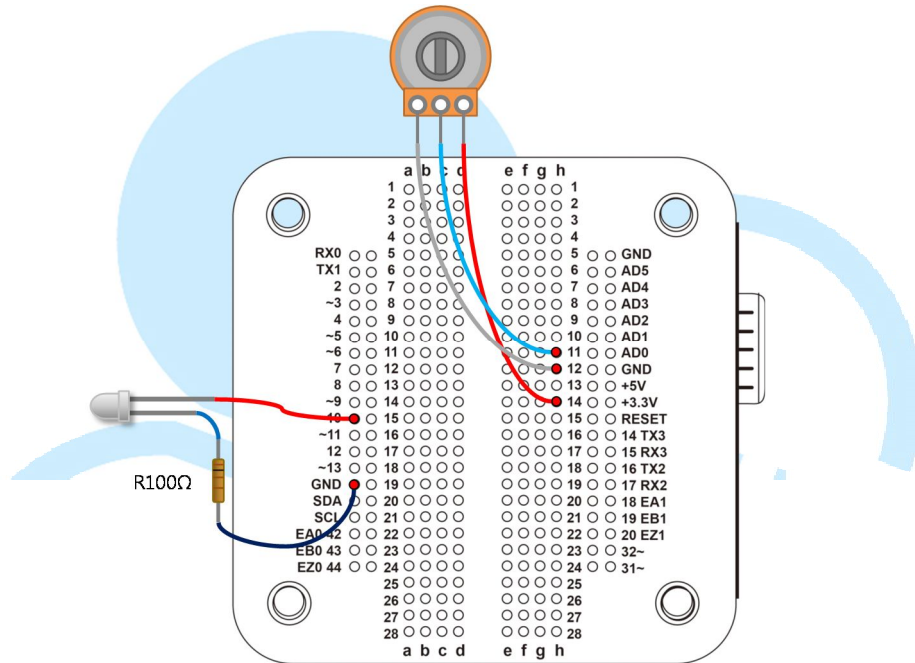
使用時可搭配以下語法：


```
unsigned int cmdBuf[ ] = {  
    8900, 4450, // H L  
    550, 600, 550, 500, // H L H L  
    600, 550, 550, 550,  
    550, 550, 600, 500,  
    550, 600, 550, 550,  
    600, 1650, 550, 1650,  
    600, 1650, 550, 1650,  
    550, 1700, 550, 1650,  
    600, 1650, 550, 1700,  
    500, 600, 550, 1650,  
    600, 1650, 600, 500,  
    500, 1700, 600, 550,  
    500, 600, 600, 550,  
    550, 1650, 600, 500,  
    600, 550, 550, 1650,  
    600, 500, 600, 1650,  
    600, 1650, 550, 1650,  
    600  
}; // 67  
int cmdLength = 67;  
IR_send.sendRaw( cmdBuf, cmdLength, 38 );
```

此段程式碼中「cmdBuf[]」儲存了一系列的 HIGH/LOW 序列時間長度，由 HIGH 開始。「sendRaw(unsigned int buf[], int len, int khz)」的參數欄位需要：資料矩陣（單位為 μs ）、送出的序列長度個數、載波頻率（單位為 kHz）等等，這樣就可以送出特定長度的序列組合囉。讀者也可以嘗試這幾個 IRsend 內建函式，若有第二台 86Duino EduCake 的話也可以與練習三-1 的程式搭配，看看各種指令格式的差異。

七、 第四個練習 – 兩台 86Duino Cake 以紅外線通訊

瞭解 IRremote 的基本用法之後，開始來做點變化，這個練習我們需要兩台 86Duino EduCake，其中一台為發射端，一台為接收端。接收端依練習三-2 的圖接線；發射端請依下圖接線：



接著請打開 86Duino Coding IDE，輸入以下傳送端的程式碼：

```
#include <IRremote.h>

int ID_send_pin = 10; // IR 接收器輸出腳位
int VR_pin = A0;

IRsend IR_send; // IRremote 函式庫傳送用物件

void setup()
{
  Serial.begin( 115200 );
```

```
    IR_send.outPin( ID_send_pin );// 注意 86Duino EduCake 預
    設的 IR 發射器 PWM 腳位為 pin 10
}

void loop() {

    unsigned int VRvalue = analogRead( VR_pin );// 讀取 AD 數
    值 0~1023
    unsigned long DeviceAddr = 0x00AA;// 接收端、傳送端 須配
    合
    // 使用 bitwise or，將資料合進 IR 指令中
    unsigned long cmd = ( DeviceAddr<<16 ) | ( unsigned
    long )VRvalue;

    IR_send.sendNEC( cmd, 32 );// send NEC code format
    (command, data bits)

    Serial.print( "VR value: " );
    Serial.println( VRvalue, DEC );
    Serial.print( "Send IR command in NEC format = " );
    Serial.println( cmd, HEX );

    delay( 200 );
}
```

然後接收端使用以下程式碼：

```
#include <IRremote.h>
#include <Servo.h>

int IR_rec_pin = 2;// IR 接收器輸出腳位
int servo_pin = 3;// Servo 輸出腳位

IRrecv IRrecver( IR_rec_pin );// IRremote 函式庫接收用對象
```

```
decode_results results;// 解碼結果存放資料用

Servo servo_0;// Servo 物件

void Print_IRdecodeResult( decode_results &decodeResults )//
印出解碼成功的訊息以便觀察
{
    // bitwise AND 取得裝置位址
    unsigned int DeviceAddr = ( unsigned
int )( ( decodeResults.value & 0xFFFF0000 ) >> 16 );
    // bitwise AND 取得資料欄位 0~1023
    unsigned int VRvalue = ( unsigned int )( decodeResults.value
& 0x0000FFFF );

    // 要 NEC 編碼，且裝置位址符合才做對應動作
    if( decodeResults.decode_type == NEC && DeviceAddr ==
0x00AA )
    {
        // 從 0~1023 映射到 1000~2000 的數值範圍
        int ServoPosition = map( VRvalue, 0, 1023, 1000, 2000 );
        ServoPosition = constrain( ServoPosition, 1100, 1900 );//
限制在安全範圍

        servo_0.writeMicroseconds( ServoPosition );// 控制角度

        Serial.print( "IR receive OK, raw data = " );
        Serial.print( VRvalue, DEC );
        Serial.print( "ServoPosition = " );
        Serial.println( ServoPosition, DEC );
    }
}

void setup( )
{
```

```
Serial.begin( 115200 );  
  IRrecver.enableIRIn( );// 初始化接收對象  
  servo_0.attach( servo_pin );// 設定 Servo 連接腳  
}  
  
void loop() {  
  if ( IRrecver.decode( &results ) )  
  {  
    Print_IRdecodeResult( results );  
    IRrecver.resume( );// 解碼完需要呼叫這一行，才能再繼續接  
收後續資料  
  }  
}
```

將傳送端、接收端分別上傳程式後，就可以從傳送端轉動可變電阻，來控制接收端的 **Servo** 轉動了。傳送端程式會定時將類比數位轉換器讀到的資料，與裝置位址 **0x00AA** 組合，並傳送出去。

接收端流程與練習三-2 類似，但做解碼成功後，利用：

```
unsigned    int    DeviceAddr    =    (unsigned  
int)((decodeResults.value & 0xFFFF0000 )>>16 )  
    unsigned int VRvalue = (unsigned int)(decodeResults.value &  
0x0000FFFF)
```

這兩個語法，來解出裝置位址欄位以及資料欄位（數位類比轉換器的資料）。接下來增加了額外限制條件：「通訊協定需為 **NEC** 且解出的裝置位址等於接收端設定的位址，才能繼續做 **Servo** 的控制」。這樣即使是同類型的通訊協定，也可避免干擾，確定是自己要接收的指令了，其餘的資料傳送也可以如法炮製。

認識了這個做法，讀者也可以搭配前面章節提過的機械手臂，實作紅外線無線遙控的版本喔。也可以再動動腦，如果要用 86Duino EduCake 實作一個萬用遙控器的話，該怎麼做呢？

