這種星狀通訊應用可以使用在環境監控上，例如加上水位感測器，我們可以監測區域環境內的水位高低，例如 Funduino 的水位感測器可以輕易地將水位高低轉換成類比訊號，再由 nRF24L01 的通訊網路回傳數值，我們就可以了解水面高低的變化，針對異常狀況來做反應。

圖 4-28　Funduino 的水位感測器

4-4 藍牙通訊

2.4GHz 的通訊，當然還有很多常用到的，藍牙就是其中一個。它原本就是被定義在短距離 (個人區域網路，PAN) 中裝置間低功耗、低成本的相互通訊介面，尤其在所謂的行動終端。最初的設計是希望解決 RS-232 的實體線路與資料同步問題，由易利信 (Ericsson) 所提出的一個解決方案。在電腦與手機的周邊裝置連線被大量採用。

不過在物聯網的浪潮下，有一度藍牙幾乎快被 ZigBee 等新一代的無線通訊所取代，原因是藍牙通訊協定下所被限制的裝置數量、支援的網路拓樸以及整體耗電並不適用於物聯網的應用。好在 2010 年 4.0 版的推出重新讓藍牙回到物聯網的戰場上。

藍牙 4.0 最重要的變革就是在電源效率的提升和網路內支援的節點數量，經由協定的演進可以支援到大量的藍牙裝置網路連線。這個部分稱為 Bluetooth Lowe Energy，簡稱 BLE，號稱可以省下近九成的電力消耗。另外還有開發出單模和雙模兩種模式 (Single Mode & Dual Mode)。單模模式提供設備能經由簡易的裝置搜尋和可靠的一對多資料傳輸來達到低成本、低功耗的傳輸目的。雙模模式則是可以與不同藍牙版本進行通訊，表示適用於與各類型的藍牙裝置進行資料交換。其協會也針對這兩種推出了不同的標誌讓大家可以清楚分辨。

圖 4-29　藍牙單模與雙模的標誌
[Bluetooth SMART READY(上)：雙模、Bluetooth SMART(下)：單模]

在與 Arduino 的搭配上，比較常見的模組是 HC-05 和 HC-06 兩款，其藍牙符合 V2.1+EDR 的規範，並且支援 SPP (Serial Port Profile) 的藍牙模組，讓使用者從電腦或是手機上連線時可以視為序列埠的裝置。

HC-05 屬於主從 (Host/Slave) 一體的模組，簡單來說就是功能比較多，可以設定為主端或是從端，一般來說出廠設定為從端。若是選購其他廠商的藍牙模組也是需要注意主從端的配置，以及是否可以調整。以 HC-05 來說，它多了一個 Key (有些模組可能是 EN) 的腳位，當輸入高電位時用來啟動 AT 命令模式，讓我們可以透過一些 ASCII 的字串來改變模組的參數設定。不過要注意有些模組為了省空間可能將此功能移除，選購上要特別注意。

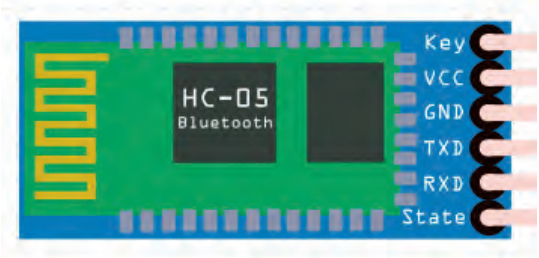


圖 4-30　HC-05 模組示意圖

至於 HC-06 就單純是被設定為是從站模式，接腳上就剩下電源與傳輸共四支腳位。

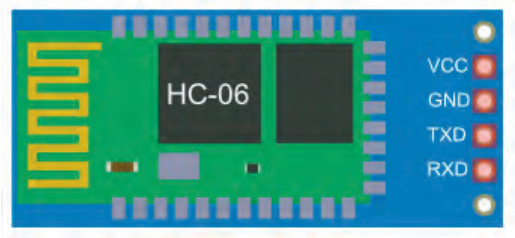


圖 4-31　HC-06 模組示意圖

我們可以先從電腦端來測試看看藍牙模組的 AT 命令。這時候與 Arduino 的連接只是將 Arduino 當作 USB 轉 TTL RS-232 的轉換介面，因此要將藍牙模組的 RX 與 Arduino 的 RX(D0) 對接，藍牙模組的 TX 與 Arduino 的 TX(D1) 對接。另外，因為這時候不需要使用到 Arduino 上的微控制器，我們將 Arduino 板子上的 reset 腳位和 GND 短路，讓微控制器持續處於重置狀態。

AT 模式		連線模式	
Arduino	HC-05/06	Arduino	HC-05/06
5V	5V	5V	VCC
GND	GND	GND	GND
RX (D0)	RXD	RX (D0)	TXD
TX (D1)	TXD	TX (D1)	RXD
5V	Key(HC-05 only)		

註：HC-05 預設速率為 38400bps，HC-06 預設為 9600bps。

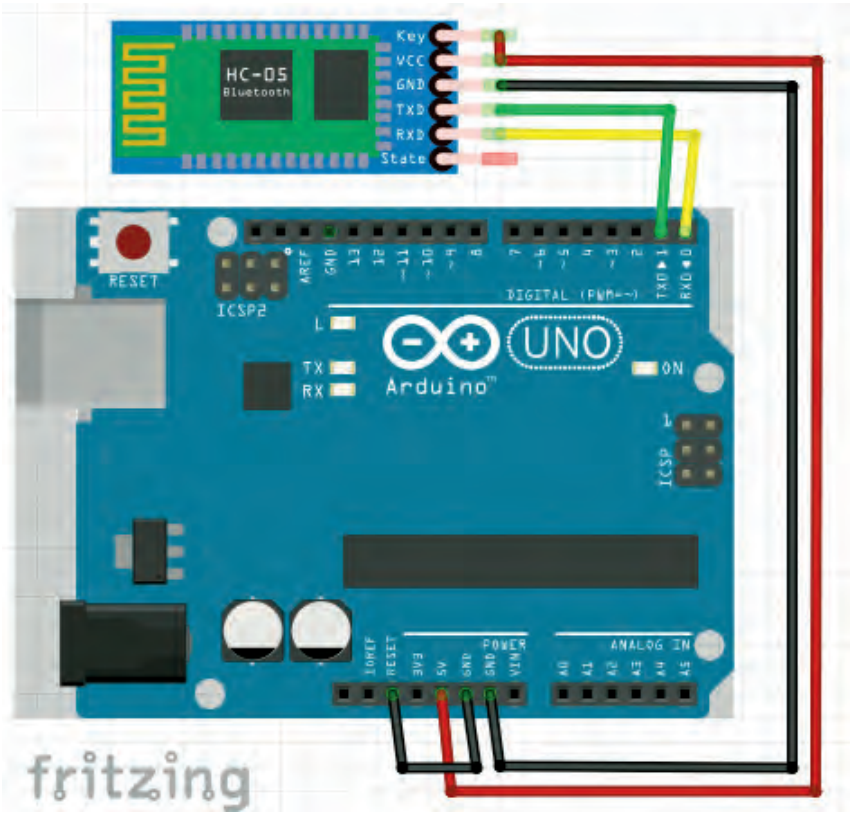


圖 4-32　電腦端透過 Arduino 連接到藍牙模組

記得要將 HC-05 的 Key 腳位也一併接到 5V 的訊號，讓藍牙模組進入設定模式，這時候模組上的 LED 燈號應該是每兩秒一次亮暗變化。開啟序列埠監控視窗，先調整速率為 38400bps 以及結尾帶 NL & CR：

圖 4-33　電腦端的通訊設定

利用輸入“AT”來確認是否正確和模組通訊，如果輸入的字串錯誤，模組會回傳 ERROR，若正確，則會顯示“OK”：

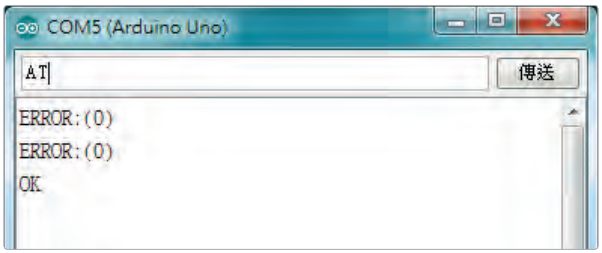


圖 4-34　藍牙模組回覆的訊息

我們也可以查詢模組的相關資料，例如韌體版本、名稱，分別使用 AT+VERSION 和 AT+NAME 來取得：

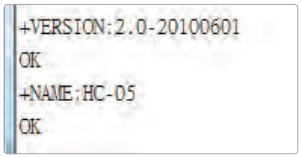


圖 4-35　模組的訊息

想要修改成自己專屬的模組名稱，只需要利用“AT+NAME=”即可修改。AT 命令輸入成功後，會顯示 OK。這時候再確認一次名稱應該就可以發現已經調整為自己剛剛輸入的。這個名稱同時也是 HOST 端在搜尋時會顯示出來的唷！

一般的電腦或是手機又該怎麼連線到模組呢？首先開啟電腦的藍牙介面，它會自動搜尋周遭的藍牙設備，這時候如果有超過一個的藍牙模組，建議連線前先修改名稱或是先關閉尚未需要連線的模組。

圖 4-36　電腦搜尋到藍牙模組

點擊後會要求輸入配對碼，預設密碼為 1234 或是 0000：

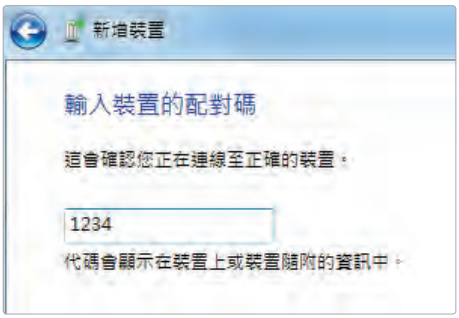
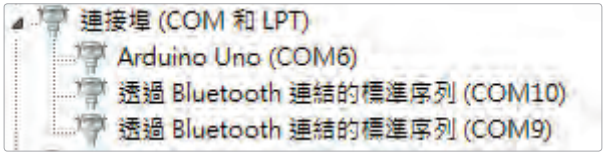


圖 4-37　輸入預設配對碼

待整個驅動流程完成後，應該就可以在裝置管理員中看到兩個由模組產生出來的 COMPORT，在移動載具的通訊上，若能採用藍牙模組就可以避免 USB 線所造成的牽絆，甚至可以直接進行資料的傳輸及程式的更新。



圖：4-38：藍牙模組的序列埠

手機端的部分，大家除了可以自己製作符合自己需求的 APP 外，也有許多玩家完成的 APP 讓我們可以直接下載使用，例如 BT Terminal 就可以讀取到藍牙發送過來的訊息。自己也可以寫個 APP 來讀取藍牙傳過來的訊息。

這邊推薦大家一個很好的 APP，不過目前只有 Android 的手機可以使用，名稱為 RoboRemo。免費版的功能就相當足夠了，有興趣的人也可以安裝付費版，多了圖片的功能。

圖 4-39　RoboRemo APP

RoboRemo 支援完全客製化的圖形元件設計，大家可以依照自己的應用完成需要的使用者介面，通訊的部分支持藍牙、Wi-Fi，因此 Arduino 可以輕鬆的經由此 APP 來與手機進行控制與資料讀取。

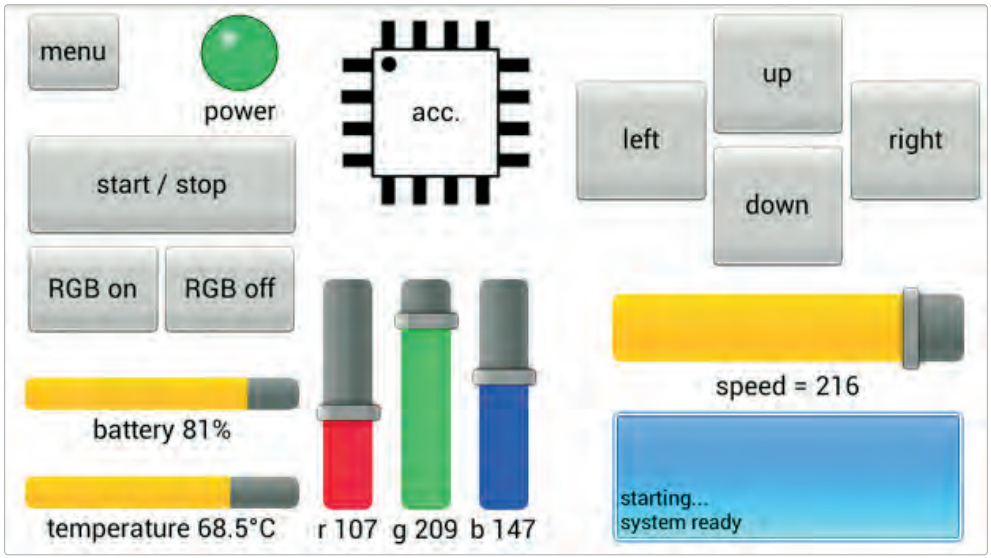


圖 4-40　RoboRemo 的圖形元件

先別急著玩 APP 的編輯，先把 Arduino 這邊的系統設計完成，我們要建立一個情境燈光控制系統，這兩年有廠商把燈泡智能化，可以依照心情需求來做調整。讓你可以隨時調整環境燈光顏色。

要能夠讓顏色被控制，單色 LED 絕對做不到，這邊要用 RGB 三色 LED。不同於單色 LED，RGB LED 共有 4 隻腳：最長的是共陽或是共陰腳，這再購買時可以選擇，會影響的是接線與控制的訊號。剩下三個依序是 R、G、B 三原色的控制。RGB LED 的光會隨 R、G、B 三個腳位電壓訊號的大小而改變。就像我們調色盤的原理一樣。

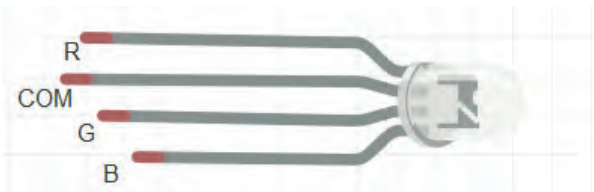


圖 4-41　RGB LED

在 Arduino 的腳位選擇上，我們要選擇能夠支援脈衝寬度調變 (Pulse-width modulation, PWM)，大多簡稱 PWM 的腳位。簡單來說它是利用數位輸出的腳位來模擬類比訊號的輸出。它還是算一種數位訊號，訊號是一個一個的方波，不過這個方波高電位與低電位的時間是可以被調整的。

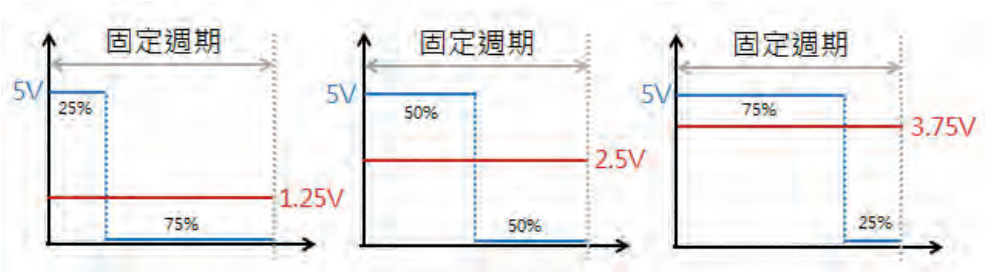


圖 4-42　PWM 訊號原理

Arduino 家族中支援 PWM 的腳位數量並不一樣，以 Arduino UNO 來說，全部共有六個，也就是：3, 5, 6, 9, 10, 11。在 Arduino 上看到腳位數字旁邊有個小蚯蚓圖案的就是了。Arduino 的 PWM 一共可以分成 256 等份，訊號輸入以 0~255 作為區分。

圖 4-43　Arduino UNO PWM 腳位

接線就以標準的 LED 接法：每個訊號腳位需要一個限流電阻。藍牙的部分我們以模擬序列埠來進行連接。

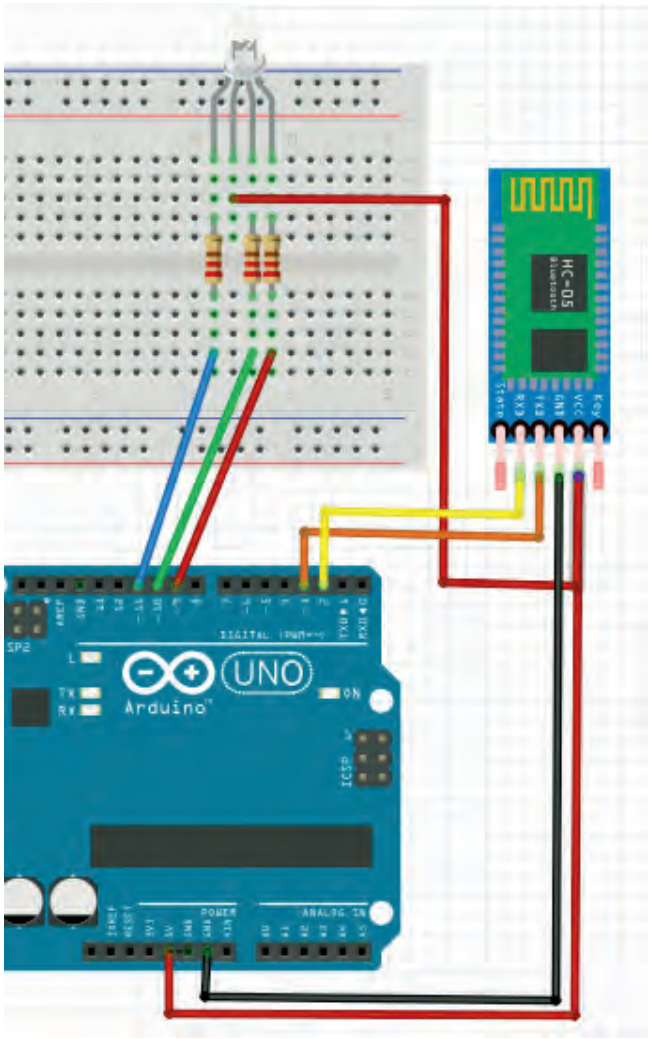


圖 4-44　系統接線

程式的部分，只需要宣告模擬序列埠並指定藍牙模組使用的 D2 和 D3 為 RX, TX。

```
1  //ArduIoT EX4-3
2  #include <SoftwareSerial.h>
3  SoftwareSerial bluetooth(2, 3); // RX, TX
```

由於筆者使用的 RGB LED 為共陽型，因此除了設定三個 PWM 腳位為輸出外，同時也設定為高電位為初始值，表示 LED 熄滅。

```
14   delay(500); //預留藍芽啟動時間
15   bluetooth.begin(9600); // Bluetooth default baud is 9600
16
17   pinMode(red, OUTPUT);  // 設定腳位為輸出
18   pinMode(green, OUTPUT);// 設定腳位為輸出
19   pinMode(blue, OUTPUT); // 設定腳位為輸出
20
21   digitalWrite(red, HIGH);  // 預設關燈
22   digitalWrite(green, HIGH);// 預設關燈
23   digitalWrite(blue, HIGH); // 預設關燈
```

主迴圈的部分，持續等待藍牙端的資料傳輸，RoboRemo 的預設結尾字元為 '\n'。所以以此為命令接收是否完整的判斷。資料收進來後，我們另建了 exeCmd() 的函式來解析並且控制輸出。

```
31   if(bluetooth.available())
32   {
33     char c = (char)bluetooth.read();
34     if(c=='\n') //Roboremo的預設結尾
35     {
36       cmd[cmdIndex] = 0;
37       exeCmd();      //解析命令格式
38       cmdIndex = 0; //陣列歸零
39     }
40     else //接收命令字串
41     {
42       cmd[cmdIndex] = c;
43       if(cmdIndex<99) cmdIndex++;
44     }
45   }
```

在 execmd() 中，要先判斷命令字串的第一個字元是否為 r、g、b 三者之一。第二個為空白字元。程式必須先判斷確認命令正確後，再讀取後面的數值，就可以控制 RGB LED 的燈光變化了。

```
48  void exeCmd()
49  {
50     //命令格式為: r 140
51    if( (cmd[0]=='r' || cmd[0]=='g' || cmd[0]=='b') && cmd[1]==' ' )
52    {
53          int val = 0;
54          for(int i=2; cmd[i]!=0; i++)
55            val = val*10 + (cmd[i]-'0');
56          //因為共陽，數值是255-Val
57          if(cmd[0]=='r') analogWrite(red, 255-val);
58          if(cmd[0]=='g') analogWrite(green, 255-val);
59          if(cmd[0]=='b') analogWrite(blue, 255-val);
60    }
61  }
```

接下來就是重頭戲，RoboRemo 的開發囉！安裝完程式後開啟應該是完全空白的畫面、只有左上角有 Menu 的按鈕。按下後會跑出清單，我們先選擇第二的 edit ui (編輯使用者介面)。

圖 4-45　RoboRemo - edit ui 畫面

點擊後，會發現整個畫面變成灰底，這時候再輕觸一下灰底的部分會看到所有可以使用的元件。這邊要用的是 slider (滑桿)。

圖 4-46　RoboRemo - item 畫面

新增的元件可以直接放大、縮小來調整，點擊可以進行參數的設定。這邊我們需要設定兩個部分：set id 和 send when moved。

- ☑ set id：就是剛剛程式中的第一個字元，依序設定：r、g、b。
- ☑ send when moved：表示當滑桿改變位置時就發出命令字串。

這邊的 set min、set max 預設值是 0 和 255，剛好就是我們需要的範圍，因此不用調整。

圖 4-47　RoboRemo – 設定元件

全部編輯完後，再點擊一次 menu，選擇 don't edit ui 離開編輯畫面，這時候大家可以試試看轉動手機，你的畫面也會隨之變化。

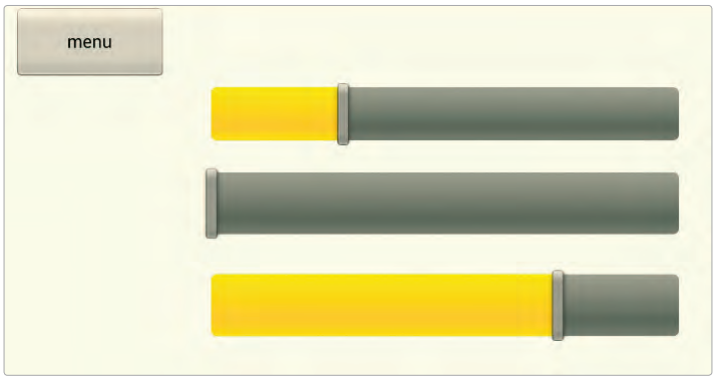


圖 4-48　RoboRemo 編輯完成畫面

要進行連線是從 menu 中的第一個選項：connect。點擊後可以選擇藍牙或是網路：

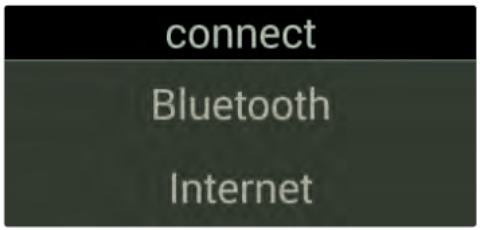


圖 4-49　連線方式選擇

這邊先選擇藍牙，如果你的藍牙預設是關閉時，RoboRemo 會提示你是否要開啟：

圖 4-50　詢問是否開啟藍牙

選擇藍牙裝置：

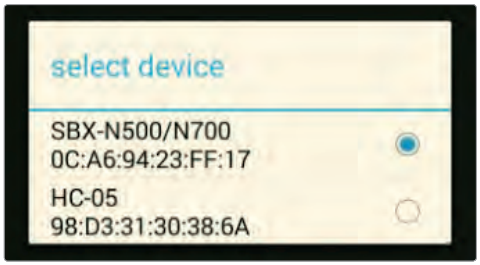


圖 4-51　選定藍牙設備