The Dino 928 互動設計入門與實作

陳胤智/Jevons Chen

Appsduino

大綱

- Arduino硬體簡介
- Arduino IDE安裝
- 基本語法介紹
- 實做練習(Labs)
- 相關模組介紹與實做
- Arduino + ZigBee(IoT)

Appsduino

Arduino 簡介



- 什麼是Arduino？
  - Arduino是基於開放原碼精神的一個開放硬體平臺，其語言和開發環境都很簡單。讓您可以使用它快速做出有趣的東西。
  - 它是一個能夠用來感應和控制現實物理世界的一套工具，也提供一套設計程式的IDE開發環境，並可以免費下載
  - Arduino可以用來開發互動產品，比如它可以讀取大量的開關和感測器信號，並且可以控制各式各樣的電燈、電機和其他物理設備。也可以在運行時和你電腦中運行的程式(例如：Flash，Processing，MaxMSP）進行通訊。

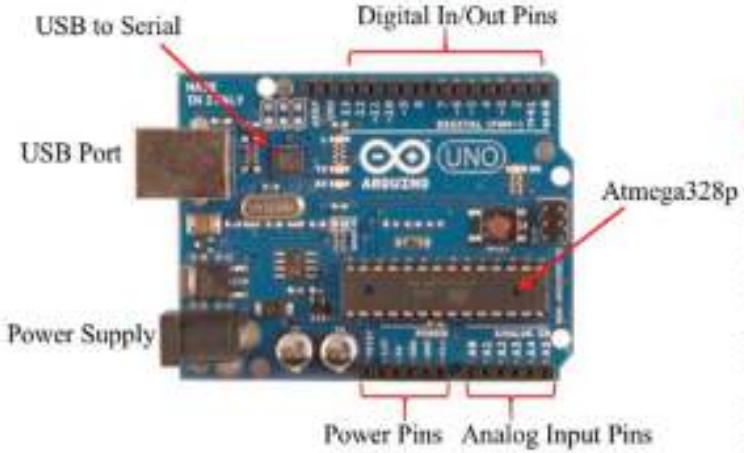
Appsduino

Arduino 簡介

- 特色描述
  - 開放原始碼的電路圖設計，開發介面免費下載，也可依需求自己修改!!
  - 下載程式簡單、方便。
  - 可簡單地與感測器、各式各樣的電子元件連接（如：LED燈、蜂鳴器、按鍵、光敏電阻等等），做出各種各樣有趣的東西。
  - 使用高速(16Mhz)的微處理控制器(ATMEGA328)。
  - 開發語言和開發環境都非常的簡單、易理解，非常適合初學者學習。
- 性能描述
  - Digital I/O 數位輸入/輸出埠D0—D13。
  - Analog I/O 類比輸入/輸出埠A0-A5。
  - 支援ISP下載功能。
  - 輸入電壓：接上USB時無須外部供電或外部5V-9V直流電壓輸入。
  - 輸出電壓：5V直流電壓輸出和3.3V直流電壓輸出和外部電源輸入。
  - 採用Atmel Atmega328微處理控制器
  - Arduino大小尺寸：寬70mm X 高54mm。

Appsduino

Arduino 簡介



Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Arduino UNO組成與規格

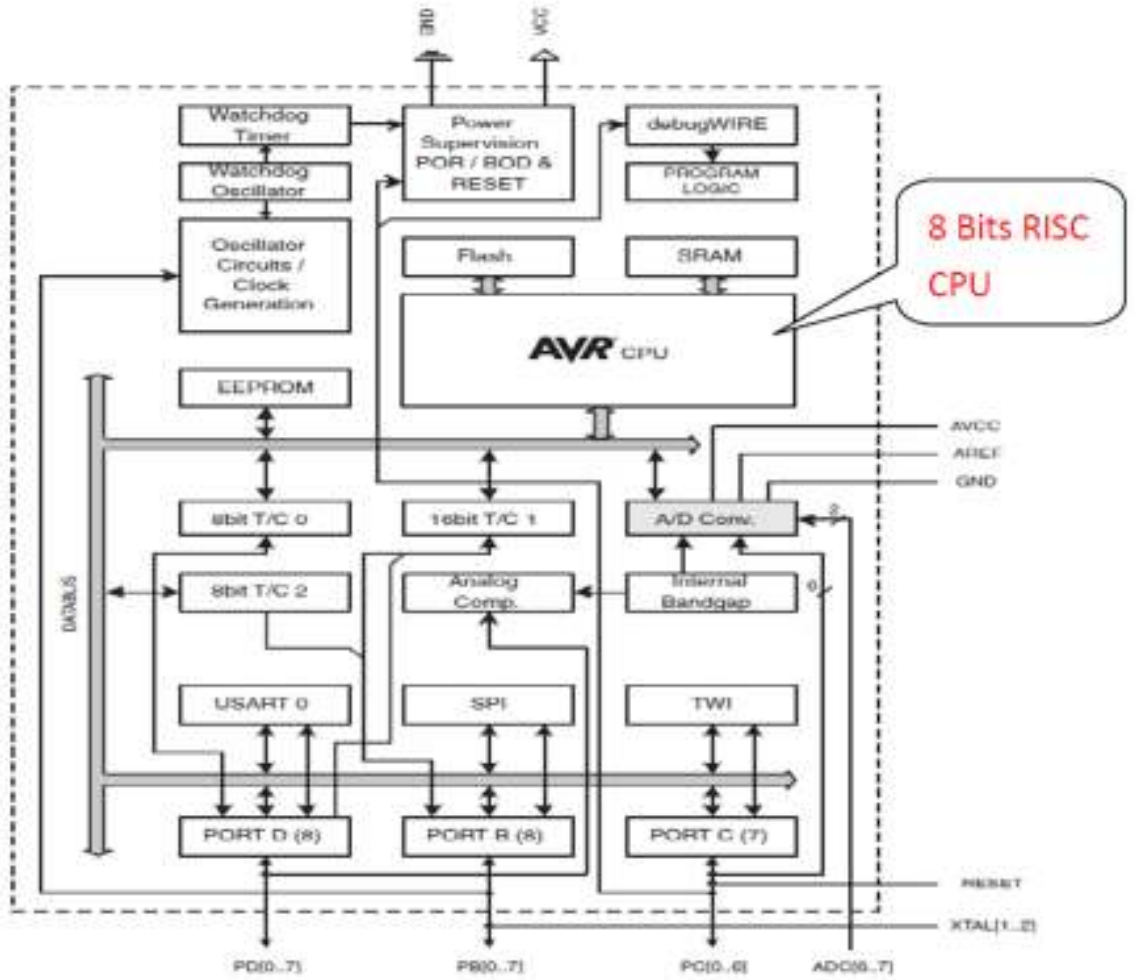
- 幾個比較特殊的埠說明：

VIN埠：VIN是input voltage的缩寫，表示有外部電源時的輸入埠。如果不使用USB供電時，外接電源可以通過此引腳提供電壓。（如電池供電，電池正極接VIN埠，負極接GND埠）。

AREF: Reference voltage for the analog inputs（類比輸入的基準電壓）。使用analogReference() 函數來設定。

Appsdüino

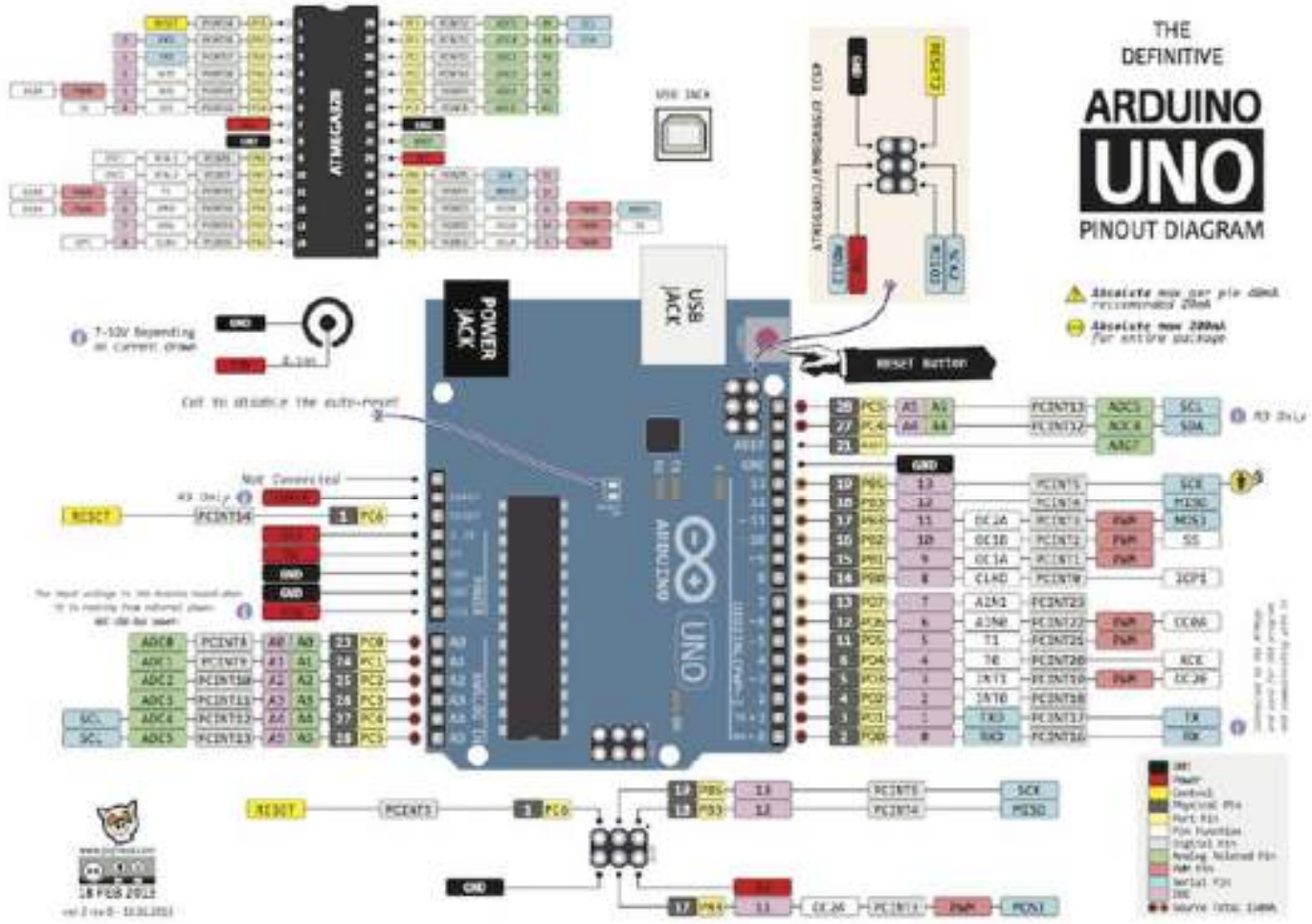
Arduino Engine : Atmega328P



Arduino UNO 核心晶片Atmega328P 架構圖(Source : Datasheet)

Appsdüino

Arduino 詳細腳位圖



Arduino 詳細腳位圖(Source : arduino forum)

Appsdüino

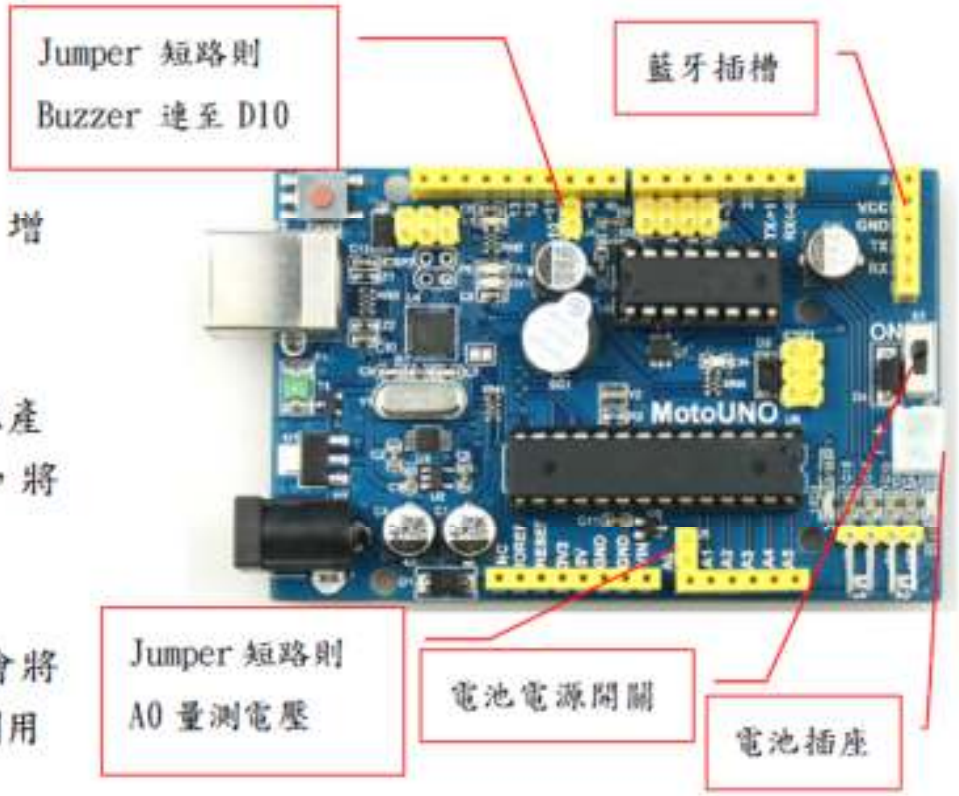
MotoUNO

主要模組功能介紹

1. Appsduino MotoUNO 控制板

Appsduino MotoUNO針對Arduino UNO-R3 增加：

- Buzzer：連接至D10(Jumper)，可以產生melody及警示告知，若欲使用D10，將Jumper開路或拔除即可
- 電池電量檢測：當Jumper短路時，會將Vin的1/2分壓連接至A0，因此即可利用Analog IO A0監測電池的電壓

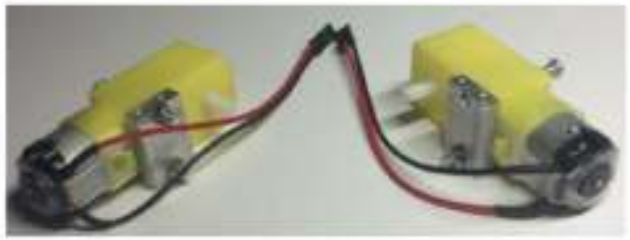


Appsduino

MotoUNO

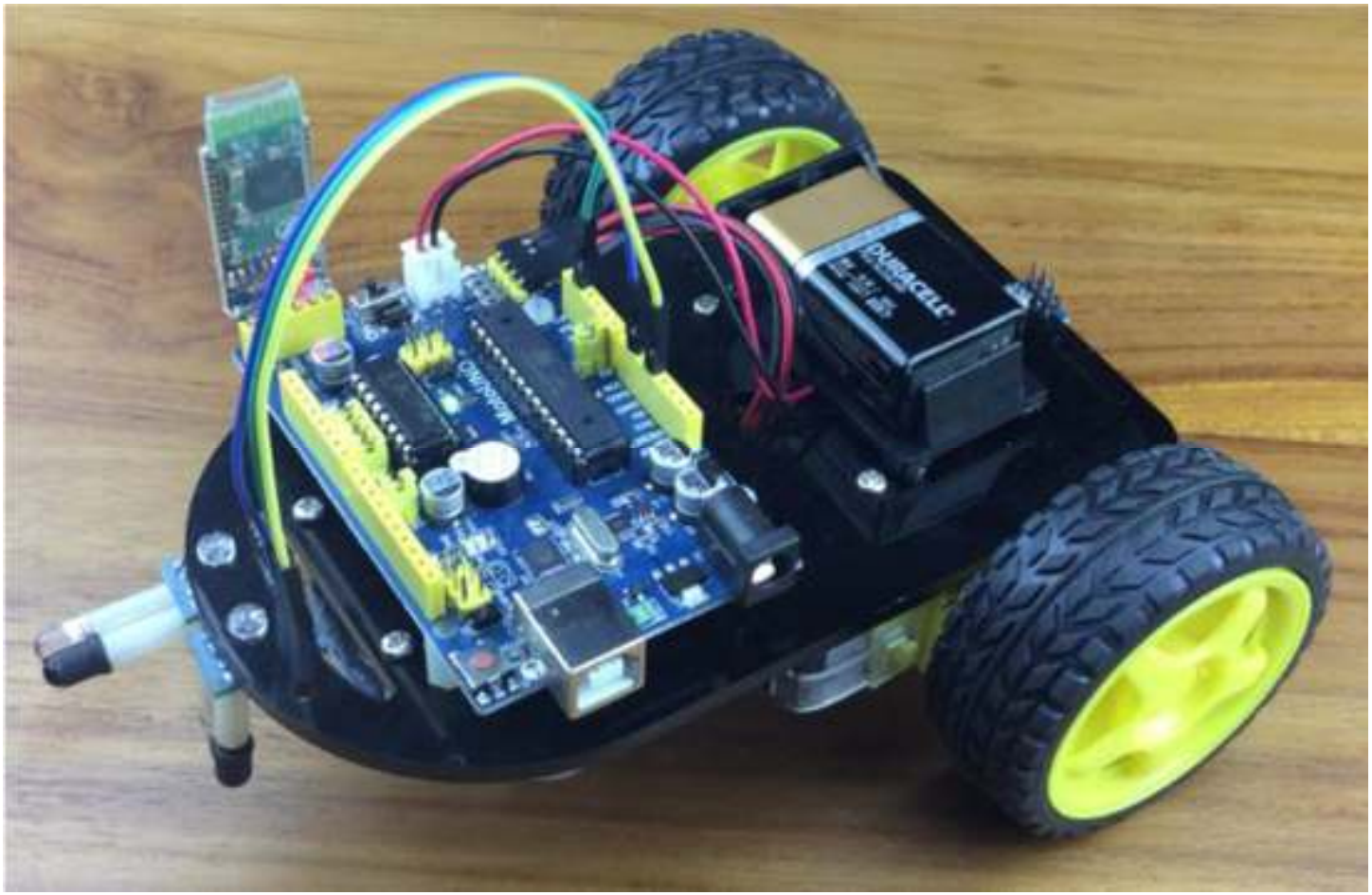
- 藍牙接口 ：將藍牙模組6 Pin排針插入接口，即可與手機或平板通訊，進而連上Internet，開創網路相關應用
- 電源滑動開關 ：黑色開關(向上on/向下off)，可打開或關閉電源
- 防呆電源接口 ：將紅黑電源線接頭插入此電源母座
- DC馬達控制 ：內建馬達控制IC，可直接控制馬達，並提供Jumper，增加使用彈性。當Jumper短路時，D4/D7可控制兩顆DC馬達，HIGH/LOW表示前進/後退，D5/D6則可利用PWM方式控制兩顆DC馬達轉速
- DC馬達接口 ：直接將馬達插入即可，節省時間





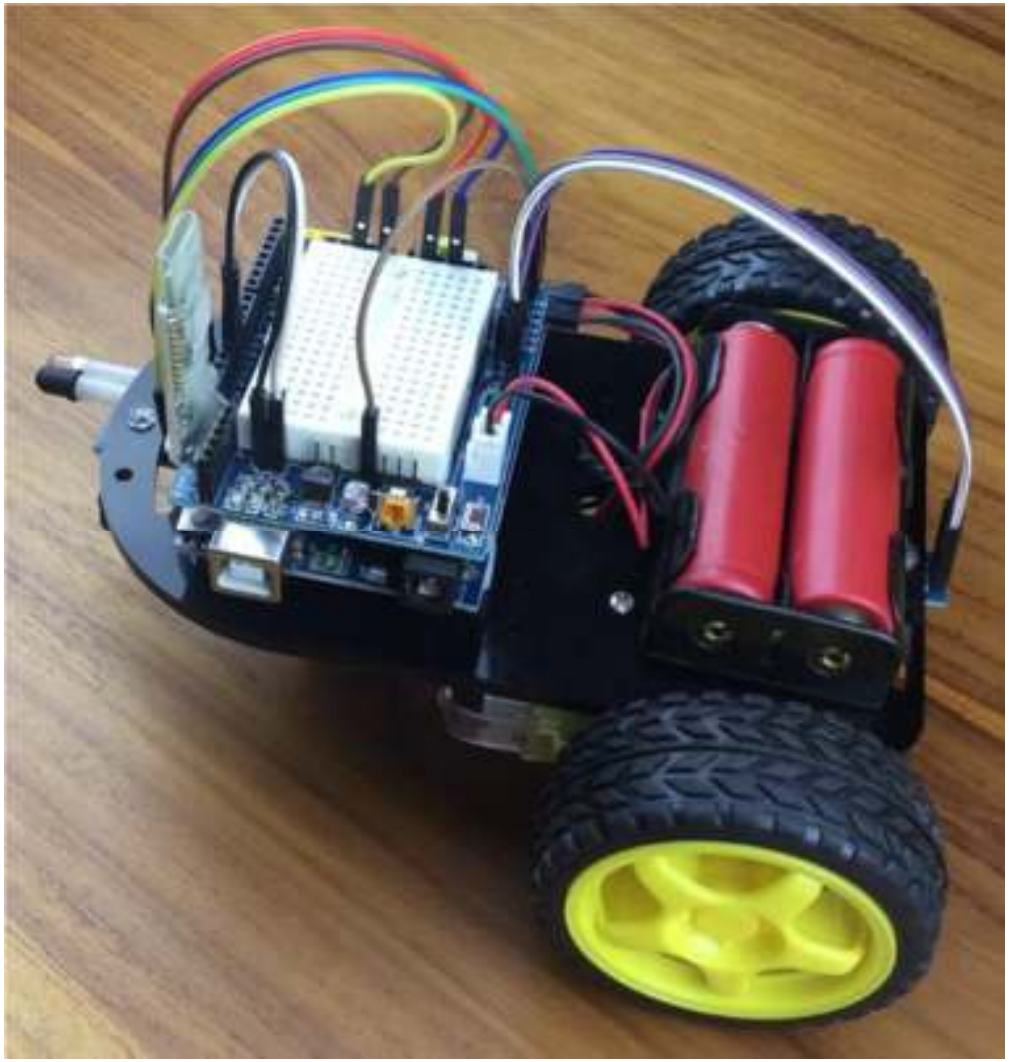
Appsduino

MotoUNO Car



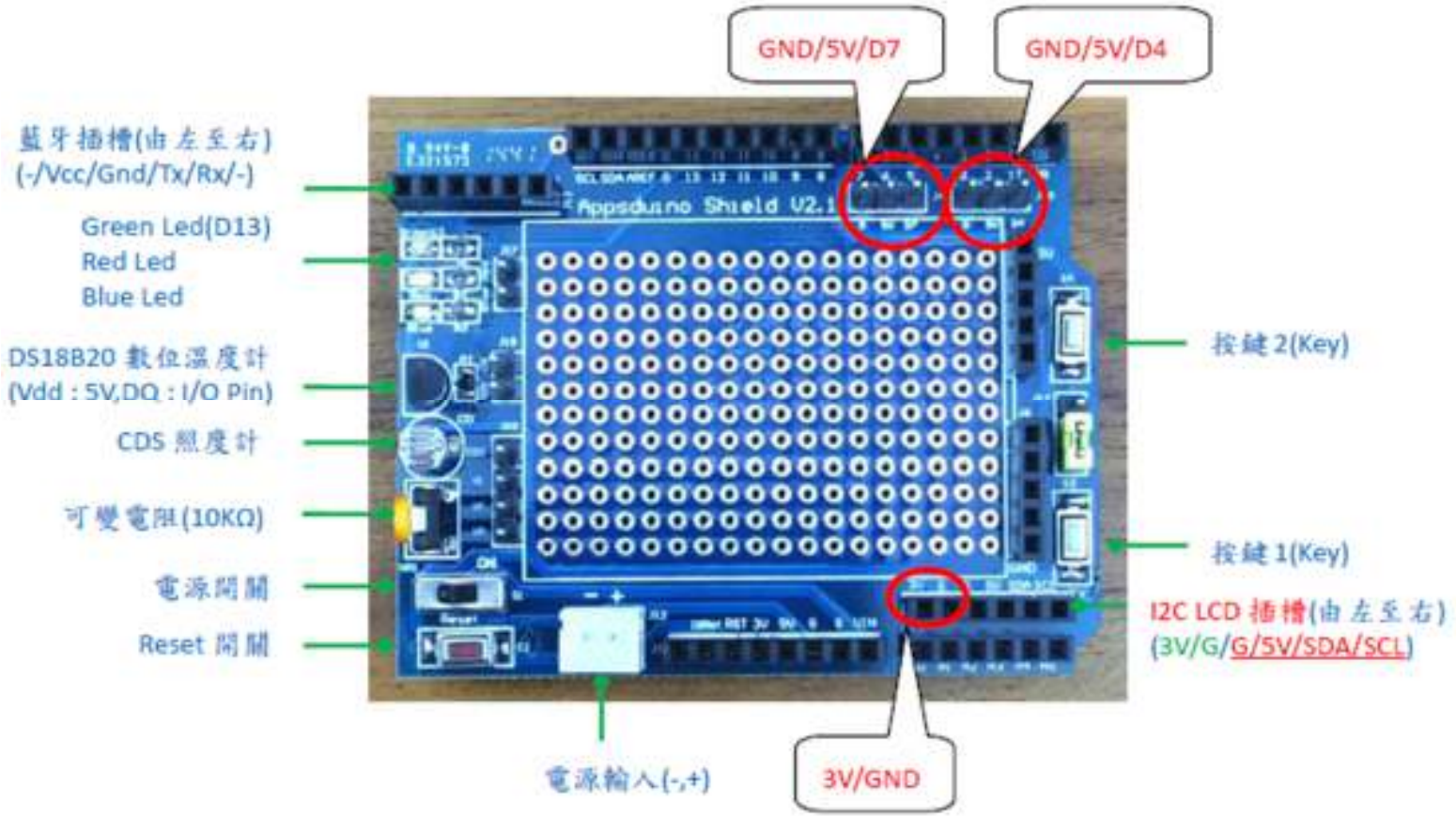
Appsduino

MotoUNO + 感測器擴充板



Appsduino

Appsduino 擴充板



Appsduino

Appsduino 擴充板

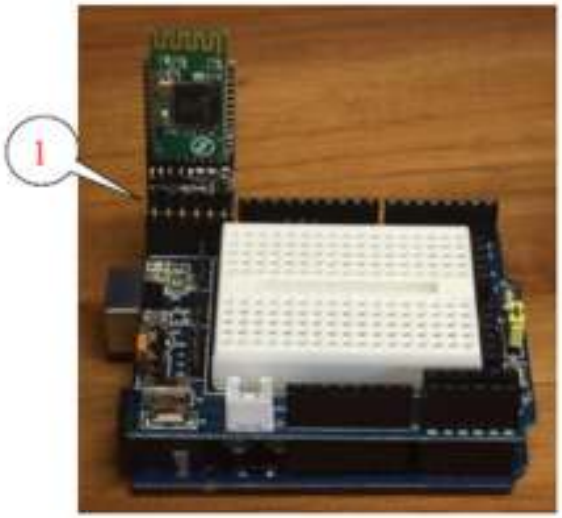
Appsduino Shield V2.1擴充板增加一些如上圖示之常用元件，利用杜邦線連至適當的IO Pins，便可輕鬆學習許多的實驗，詳述如下：

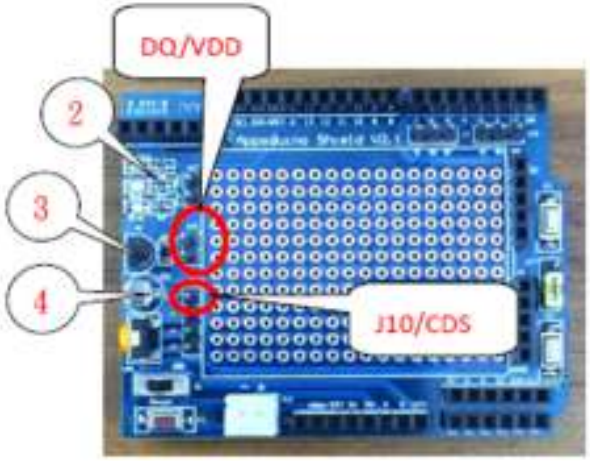
1. 藍牙接口：將藍牙模組6 Pin排針插入接口(元件面向內，如右圖)，即可與手機或平板通訊，進而連上Internet，開創網路相關應用

2. 綠、紅、藍 Led：綠色(Green) Led 已連接至D13，可直接使用，紅、藍 Led可透過J17的兩個排針，用杜邦線連至適當的IO腳位即可

3. 數位溫度計(DS18B20)：將J19的Vdd接腳連至5V，DQ連至適當的Digital IO腳位，即可量測環境的溫度(-55度C ~ +125度C)

4. 光敏電阻(CDS)：當光敏電阻受光時，電阻值變小，若用手指遮擋光敏電阻(暗)，電阻值變大，可利用此特性來監測環境受光的變化，將J10的CDS接腳連至適當的Analog IO腳位(A0~A5)，即可量測環境光線的變化





Appsduino

Appsduino 擴充板

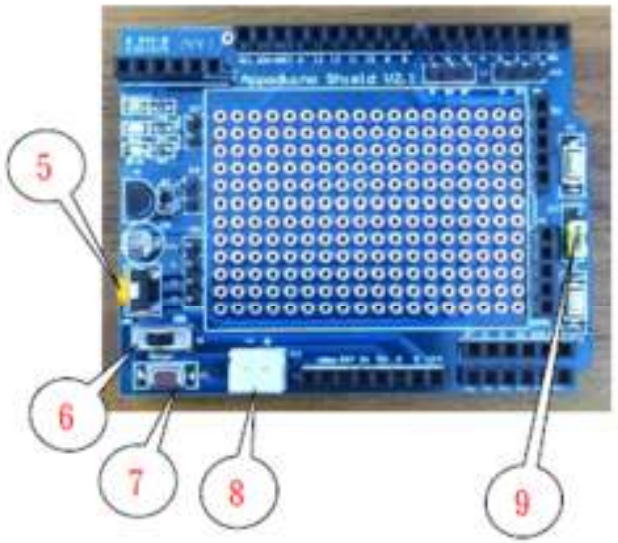
5. 可變電阻/VR(10KΩ)：內建10KΩ的可變電阻，其三支腳分別對應VR/VC/VL腳位，可利用這些腳位並旋轉旋鈕以獲得所需的阻值

6. 電源滑動開關：黑色開關(向右on/向左 off)，可打開或關閉從電源輸入接口送至UNO控制板的電源(VIN)

7. Reset按鍵：紅色按鍵為Reset Key

8. 電源輸入接口：將紅黑電源線接頭插入此電源母座(紅色為正極/+，黑色為負極/-)

9. 測試按鍵(Key)：若將J14 Jumper B/C 短路，則Key 1(S3)按鍵自動連至A1 Pin，無需接線，可將A1設定為Digital I/O 或 利用Analog I/O (A/D)來偵測Key 1按鍵的狀態。Key 2(S4)按鍵則需使用杜邦線將J14 Jumper A 連至適當的數位或類比腳位



Appsduino

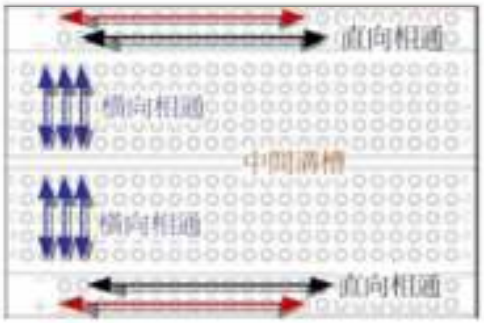
Appsduino 擴充板

10. I2C LCD插槽 ： 若有I2C LCD模組，可將所附的4根杜邦線一端接I2C控制板，另一端直接插入此插槽，如右圖示

11. 3.3V電源 ： 因應3.3V感測器需求而增加一組電源，以便彈性使用

12. 5V/GND 接口 ： 擴充板底部增加5組5V與5組GND接口，以便5V感測器使用

13. 迷你麵包板 ： 擴充板放上麵包板，就可直接插上零件，實做許多有趣的創意 !但需注意麵包板橫向是相通的，如下圖所示







Appsduino

大綱

- Arduino硬體簡介
- Arduino IDE安裝
- 基本語法介紹
- 實做練習(Labs)

Appsduino

Arduino IDE 安裝

1. 下載 Arduino 開發環境(網址：http://arduino.cc/en/Main/Software)



2. 下載後解壓縮到指定資料夾(以 C:/arduino 為例)，雙擊 arduino.exe 啟動主程式即可使用，最好在桌面建立 Arduino 捷徑，以方便使用

3. 將 USB 線接上 Appsduino UNO 控制板與電腦，會出現驅動程式安裝畫面：

Appsduino

安裝USB驅動程式(COM port)





4. 手動指定內附的驅動程式位置(ex：C:/arduino/arduino1.0.5/drivers)，系統即會自行安裝



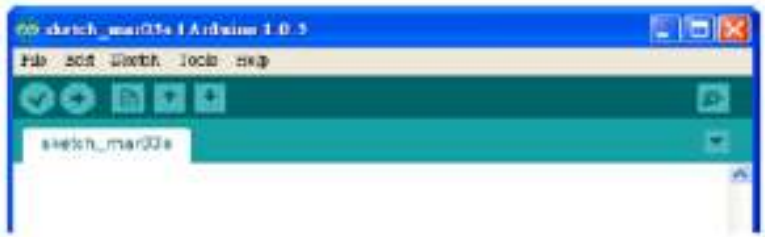


5. 請使用"控制台內系統之裝置管理員"查詢並記下安裝的序列埠，(ex：COM26)

Appsduino

Arduino IDE 執行

6. 雙擊 Arduino.exe 或 Arduino 捷徑，啟動 Arduino 主程式並選擇對應的控制板



Arduino 發展系統介面



選擇對應的控制板(Arduino UNO)

7. 點選連接埠代號 tools-> Serial Port -> COM26 並測試第一個程式(Blink)



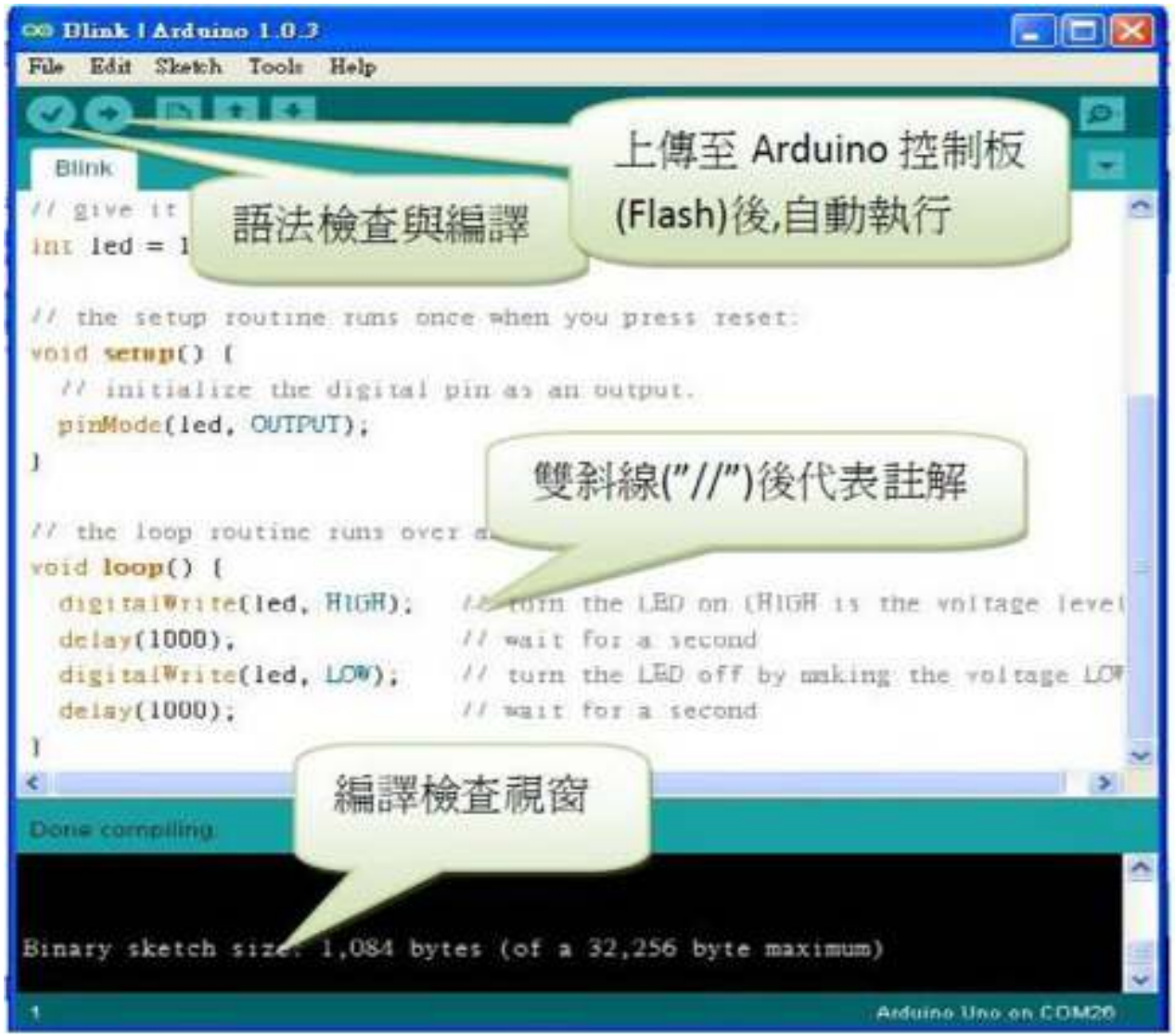
選擇連接埠



測試第一個程式(Blink)

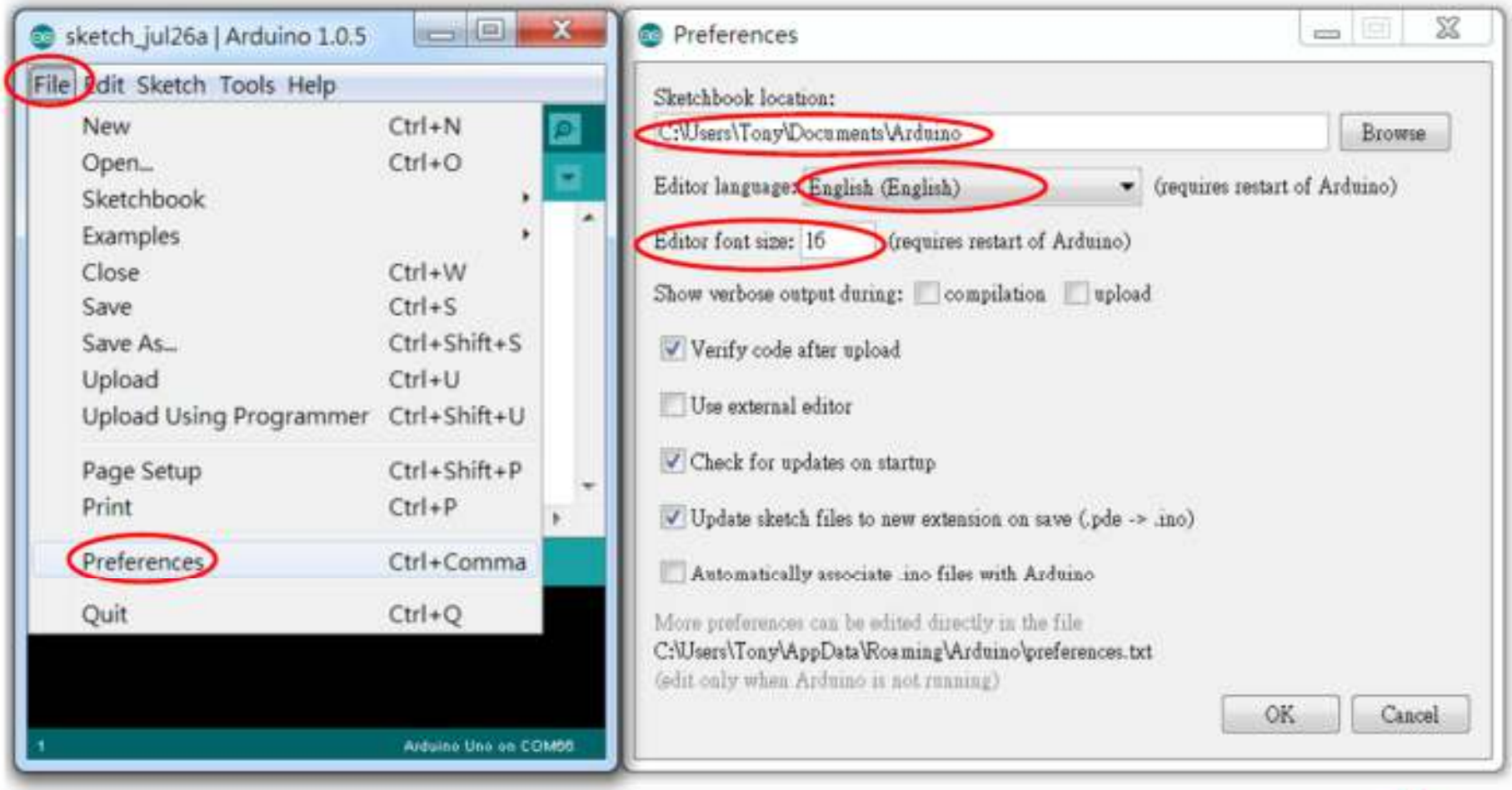
Appsdiino

Arduino IDE 開發環境



Appsduino

修改IDE開發環境個人喜好設定 (存檔路徑、語言、字型)



Appsduino

大綱

- Arduino硬體簡介
- Arduino IDE安裝
- 基本語法介紹
- 實做練習(Labs)

Appsduino

Arduino 程式語法簡介

Arduino 語言是建立在 C/C++基礎上的，只不過把 ATmega 單晶片相關的一些參數設置都函數化，無需去瞭解繁複的底層設定，因此變得簡單容易，一般有興趣的朋友都能輕鬆上手。

Arduino 語言的三個組成要素

- Structure（結構）
  - setup()
  - loop()
  - 控制(Control)
- Variables（變數）
  - Constants(常數)
  - Data Types(資料型態)
- Functions（函數，方法）
  - Digital I/O
  - Analog I/O
  - Time ：時間函數
  - Communication ：通訊函數
  - ....

Appsduino

Arduino基本語法

Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comment)
- /* */ (multi-line comment)
- #define
- #include

Arithmetic Operators

- = (assignment operator)
- + (addition)
- - (subtraction)
- * (multiplication)

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT| INPUT_PULLUP
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- short
- float
- double
- string - char array
- String - object
- array

Conversion

- char()

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - PWM

Due only

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Time

- millis()
- micros()
- delay()
- delayMicroseconds()

Math

Source : **http://arduino.cc/en/Reference/HomePage**

Appsdüino

Arduino 程式主要架構

```
變數型態宣告區 ; // 這裡定義變數或 IO 腳位名稱

void setup()
{

}

void loop()
{

}
```

僅在 Power On 或 Reset 後執行一次，setup()函數內放置初始化 Arduino 控制板的程式，即主程式開始執行前需事先設定好的變數 or 腳位定義等
例如 : pinMode(ledPin,OUTPUT);

在 setup()函數之後，即初始化之後，系統則在 loop()程式迴圈內重複執行。直到 Arduino 控制板被關閉。

; 每行程式敘述(statement)後需以分號(";")結束

{ }(大括號) 函數前後需用大括號括起來，也可用此將程式碼分成較易讀的區塊

Appsdüino

程式的註解與說明

程式的註解就是對代碼的解釋和說明，編寫註解有助於程式設計師(或其他人)瞭解代碼的功能。Arduino 編譯程式在對程式碼進行編譯時會忽略註解的部份。

Arduino 語言中的編寫註解有兩種方式

```
//      單行註解：這整行的文字會被編譯程式忽略
```

```
/*      多行註解：
        在這個範圍內你也可以寫一整篇文章
*/
```

Appsduino

Variables(變數）

Variable(變數)：定義資料暫存之處(以資料暫存型態定義)

資料型態

boolean	true or false
char	'a' , 'b' , 'c'
byte	B10010011
int	-32,768 ~ 32,767
unsigned int	0 ~ 65,535
long	-2,147,483,648 ~ 2,147,483,647(4 Bytes)
unsigned long	0 ~ 4,294,967,295
float	單精度浮點數， 10^-38 ~ 10^38
double	雙精度浮點數， 10^-308 ~ 10^308
string	"I' m Arduino"
array	char A[8] = { 'a' ,' r' ,' d' , 'u' , 'i' , 'n' , 'o' }
void	void setup()

Appsduino

Arduino語言：Constant(常數)

在 Arduino 語言中事先定義了一些具特殊用途的保留字

HIGH | LOW

表示數字 IO 腳位的電位，HIGH 表示高電位(1)，LOW 表示低電位(0)
HIGH 和 LOW 也用來表示你開啟或是關閉了一個 Arduino 的腳位(pin)

INPUT | OUTPUT | INPUT_PULLUP

表示數字 IO 腳位的方向，INPUT 表示輸入，OUTPUT 表示輸出(AVR 能提供 5V 電壓 40mA 電流)
INPUT_PULLUP 表示輸入模式但啟動內部的上拉電阻(Pull up resistor)

true |false

true 表示真(1)，false 表示假(0)

Appsdüino

基本程式設計：if ..

```
if(條件式)
{
 指令敘述 ; //如果只有1行則無須大括號 "{"
與 "}"
}
```

至於條件式是什麼 ?

a > b (a大於b)
a < b (a小於b)
a>=b (a大於等於b)
a<=b (a小於等於b)
a != b (a不等於b)
a == b (a等於b)

例子 :

```
if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120){
digitalWrite(LEDpin, HIGH); }

if (x > 120){
 digitalWrite(LEDpin1, HIGH);
 digitalWrite(LEDpin2, HIGH);
}
```

Appsduino

基本程式設計：if .. else ..

```
if(條件式1)
{
敘述區塊1
}
else if(條件式2)
{
敘述區塊2
}
.
.
else if(條件式n)
{
敘述區塊n
}
```

例子 ：

```
if (pin5 < 500)
{
 // do Thing A
}
else if (pin5 >= 1000)
{
 // do Thing B
}
else
{
 // do Thing C
}
```

Appsdüino

基本程式設計：for

- 當程式需要來回重複執行某一段程式碼時， 就可以使用迴圈

```
for（起始值 ; 條件式 ; 更新值）
 指令一 ;
```

或

```
for（起始值; 條件式; 更新值）{
 指令一;
 指令二;
 指令三;
}
```

例子 ：

```
int PWMpin = 10; //

for (int i=0; i <= 255; i++){
     analogWrite(PWMpin, i);
     delay(10);
}
```

Appsduino

基本程式設計：while

- 當 while 之後的條件成立時，執行括弧內的程式碼

```
while（條件式）
指令一;

或

while（條件式）{
指令一 ;
指令二 ;
指令三 ;
}
```

例子 :

```
int var = 0;

while(var < 200) {
 // 重複做某些事200次
 var++;
}
```

Appsduino

基本程式設計：do ... while

- do ... while 與while 指令運作方式一樣，只是將當條件式移至 while()之後，所以 do...while() 至少會執行一次括弧內的程式碼

```
do {
指令一 ;
指令二 ;
指令三 ;
.....
} while(條件式) ;
```

例子 :

```
int x = 0;

do {
   delay(50) ;
   x = readSensor() ;
} while(x < 100) ;//至少執行一次才檢查
```

Appsduino

基本程式設計：switch ... case

if 敘述是程式裡的分叉路口，switch case 是更多選項的路口。swith..case 根據變數值讓程式有更多的選擇，比起一串冗長的 if 敘述，使用 switch ..case 可使程式碼看起來比較簡潔。

```
switch(value) {
 case 1 :
 ...
   break ;
 case 2 :
 ...
   break ;
 default :  //以上條件都不符合時，預設執行的
              動作
 ...
}
```

例子 ：

```
switch (sensorValue) {
case 23:
digitalWrite(13,HIGH);
break;
case 46:
digitalWrite(12,HIGH);
break;
default: //執行預設的動作
digitalWrite(12,LOW);
digitalWrite(13,LOW);
}
```

Appsduino

Functions : Digital I/O

Arduino 語言中事先定義了許多有用的函數，例如 ： 數位 I/O、類比 I/O、時間(Time)、通訊、音律(Tone)函數、…… 等等。

Digital I/O 函數 ：

- **pinMode(pin, mode)**
  初始化時需指定 pin 的狀態，mode 參數必須是 OUTPUT 或 INPUT，並放在 setup()當中

- **digitalWrite(pin, value)**
  將所指定的 pin 設定高或低、開或關，pin 表示為 0～13，value 表示為 HIGH 或 LOW。但腳位必須先透過 pinMode 設定為輸入或輸出模式，digitalWrite 才能生效。比如定義 HIGH 可以驅動 LED。例 ： digitalWrite(8,HIGH); //將 I/O 腳位 8 設定輸出高電位

- **int digitalRead(pin)**
  讀取所指定 pin 的狀態，pin 表示為 0～13，value 表示為 HIGH 或 LOW。比如可以讀數字感測器。當感測到腳位處於高電位時時回傳 HIGH，否則回傳 LOW。
  例 :val = digitalRead(7); // 讀出腳位 7 的值並指定給 val

Appsdüino

Time(時間函數)

Time(時間)函數 :

- unsigned long millis()

  返回時間函數(單位 ms)，回傳板子開始執行到目前的毫秒數，該函數是指當程式運行就開始計時並返回記錄的參數，該參數溢出大概需要 50 天時間

  例 : duration = millis()-lastTime; // 表示自"lastTime"至當下的時間

- delay(ms)

  延時函數(單位 ms) ，延時所指定的時間，即暫停執行多少毫秒

  例 : delay(500); //暫停半秒 (500 毫秒)

- delayMicroseconds(us)

  延時函數(單位 us),暫停晶片執行多少微秒。

  例 : delayMicroseconds(1000); //暫停 1000 微秒，即暫停 1 毫秒

Appsdüino

大綱

- Arduino硬體簡介
- Arduino IDE安裝
- 基本語法介紹
- 實做練習(Labs)

Appsduino

第一個 Arduino 程式

```
int ledPin = 13;   // 這裡設定所要閃爍的LED燈腳位

void setup() {
  pinMode(ledPin, OUTPUT);        // 這裡設定所要輸出的PIN腳
模式
}

 void loop() {
   digitalWrite(ledPin, HIGH);        // 設定PIN13腳位為高電位 =
5V
   delay(1000);// 設定延遲時間,1000ms = 1秒
   digitalWrite(ledPin, LOW);        // 設定PIN13腳位為低電位 =
0V
   delay(1000);    // 設定延遲時間
}
```

Appsduino

第一個 Arduino 程式

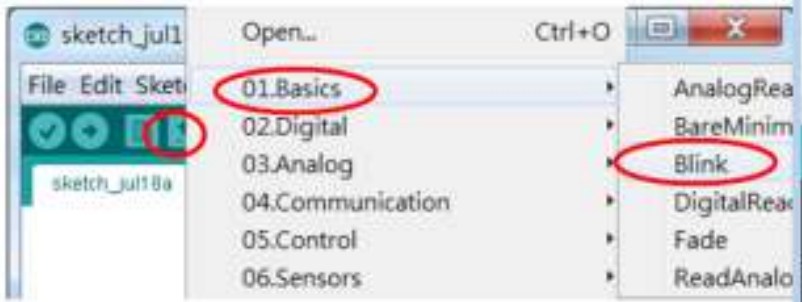
數位輸出實驗

Lab 1: 內建 LED(Pin 13)閃爍實驗

點選 檔案->新增 建立一個新程式



或點選 ，開啟 Arduino 內建的範例程式 Blink，如下所示：





第一個 Arduino 程式

若打錯字的話，最下方會出現 Verify 失敗的訊息，同時在錯誤的地方出現高亮度提示列



Appsduino

第一個 Arduino 程式

程式內容說明

當板子啟動時須設定的參數，只執行一次

```
// 閃紅燈
void setup()                    // 只執行一次
{
  pinMode(13, OUTPUT);          // 設定13號腳為輸出 output
}

void loop()                     // 不斷的重複執行
{
  digitalWrite(13, HIGH);       // 點亮LED
  delay(500);                   // 等 0.5 秒鐘
  digitalWrite(13, LOW);        // 熄滅LED
  delay(500);                   // 等 0.5 秒鐘
}
```

當板子啟動後，會一直執行的工作，並不斷重複執行

試著改變程式 delay( )函數所延遲的時間

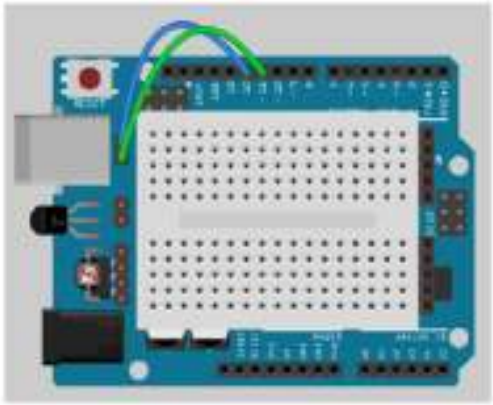
```
void loop()                     // 不斷的重複執
{
  digitalWrite(13, HIGH);       // 點亮LED
  delay(100);                   // 等？秒鐘
  digitalWrite(13, LOW);        // 熄滅LED
  delay(100);                   // 等？秒鐘
}
```

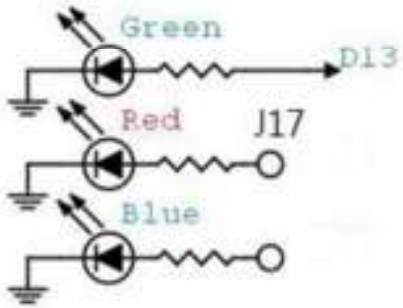
並且重新驗證、上傳程式，觀察燈光閃爍的變化

Lab 2 : 紅、藍 LED 交互閃爍

將擴充板安裝在 UNO 上面(如下圖)







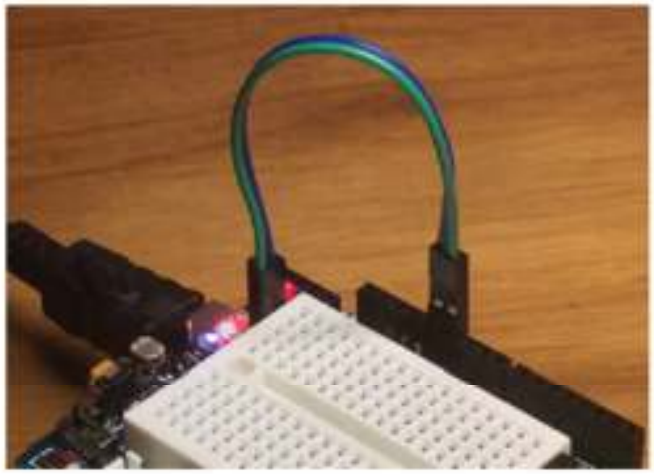
擴充板上面內建三個 LED(Green、Red、Blue)，其中 Green Led 已與 UNO 的接腳 D13 相連，另外兩個 Red、Blue 可經過旁邊 J17 的兩個接腳，連接到擴充板上的任一個數位 I/O 接腳，再由程式控制 LED 亮滅。

Appsdüino

Lab 2：紅、藍 LED 交互閃爍

將 J17 的兩個接腳，分別接到 Appsduino 的數位接腳 D12、D11，分別控制紅燈、藍燈，硬體接線圖如右，程式則如下 ：

```
// 雙閃燈
void setup()  {              // 只執行一次
  pinMode(11, OUTPUT);  // 設定 D11 為輸出 Pin
  pinMode(12, OUTPUT);  // 設定 D12 為輸出 Pin
}
void loop()  {             // 不斷的重複執行
  digitalWrite(11, LOW);    // 熄滅 LED
  digitalWrite(12, HIGH);   // 點亮 LED
  delay(500);               // 等 0.5 秒鐘
  digitalWrite(12, LOW);    // 熄滅 LED
  digitalWrite(11, HIGH); // 點亮 LED
  delay(500);               // 等 0.5 秒鐘
}
```



程式執行後會看到擴充板上的 Red、Blue 兩燈交互閃爍

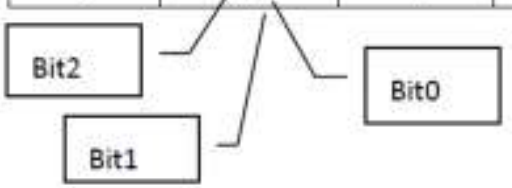
Appsduino

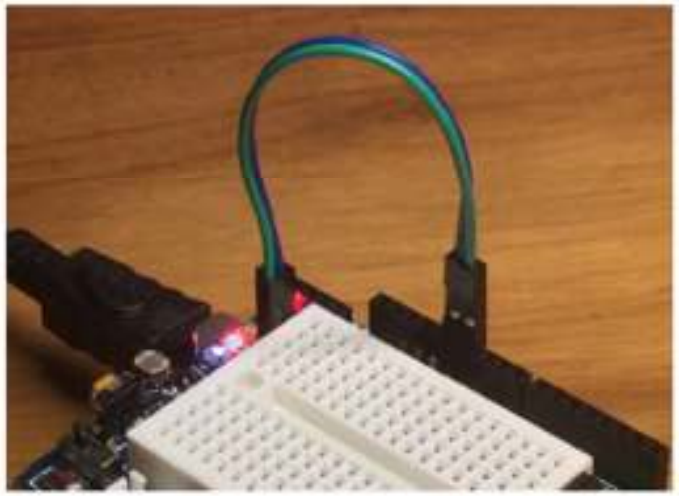
Lab 3：RGB LED 二進制表示

控制 G R B 三個 LED 以二進制方式閃爍，以表示 0-7 的數值

十進位數字可以轉換成二進位的顯示方式，例如下表就是 0-7 的二進位表示，若將二進制的 01 符號對應到 LED，就可以從燈亮看出數字變化。

十進制	二進制	十進制	二進制
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111





硬體線路如同前一個實驗，J19 的兩個接腳，

分別接到 D12(Red)、D11(Blue)如右圖所示。

宣告一個變數 i，從 0-7 逐漸加 1 遞增，並且利用 bitRead( )函數取出指定位元值，

例如 bitRead(3,2)就是從數字 3（相當於二進位 011)取出 bit 2 的值，也就是 0；

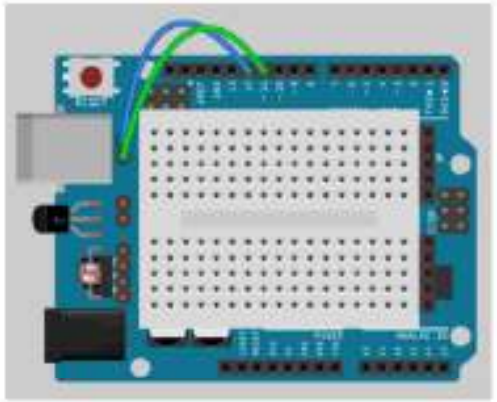
Appsduino

Lab 3：RGB LED 二進制表示

bitRead(3,1)就是從數字 3（相當於二進位 011)取出 bit 1 的值，也就是 1；

bitRead(3,0)就是從數字 3（相當於二進位 011)取出 bit 0 的值，也就是 1；

程式說明



```
// 二進制方式閃爍
void setup()   {              // 只執行一次
  pinMode(11, OUTPUT);       // 設定 11 號腳為輸出 output
  pinMode(12, OUTPUT);       // 設定 12 號腳為輸出 output
  pinMode(13, OUTPUT);       // 設定 13 號腳為輸出 output
}
void loop()                  // 不斷的重複執行
{
  for (int i=0; i<8; i++)  // 變數 i = 0-7
  {
    digitalWrite(11, bitRead(i, 0));   // 取出變數 i 的第 0 個位元值
                                       // 控制 D11 連接的燈亮或滅
    digitalWrite(12, bitRead(i, 1));   // 取出變數 i 的第 0 個位元值
                                       // 控制 D12 連接的燈亮或滅
    digitalWrite(13, bitRead(i, 2));   // 取出變數 i 的第 0 個位元值
                                       // 控制 D13 連接的燈亮或滅
    delay(500);                // 等 0.5 秒鐘
  }
}
```

執行程式，就會看到 LED 以二進制的方式顯示 0~7 數值。

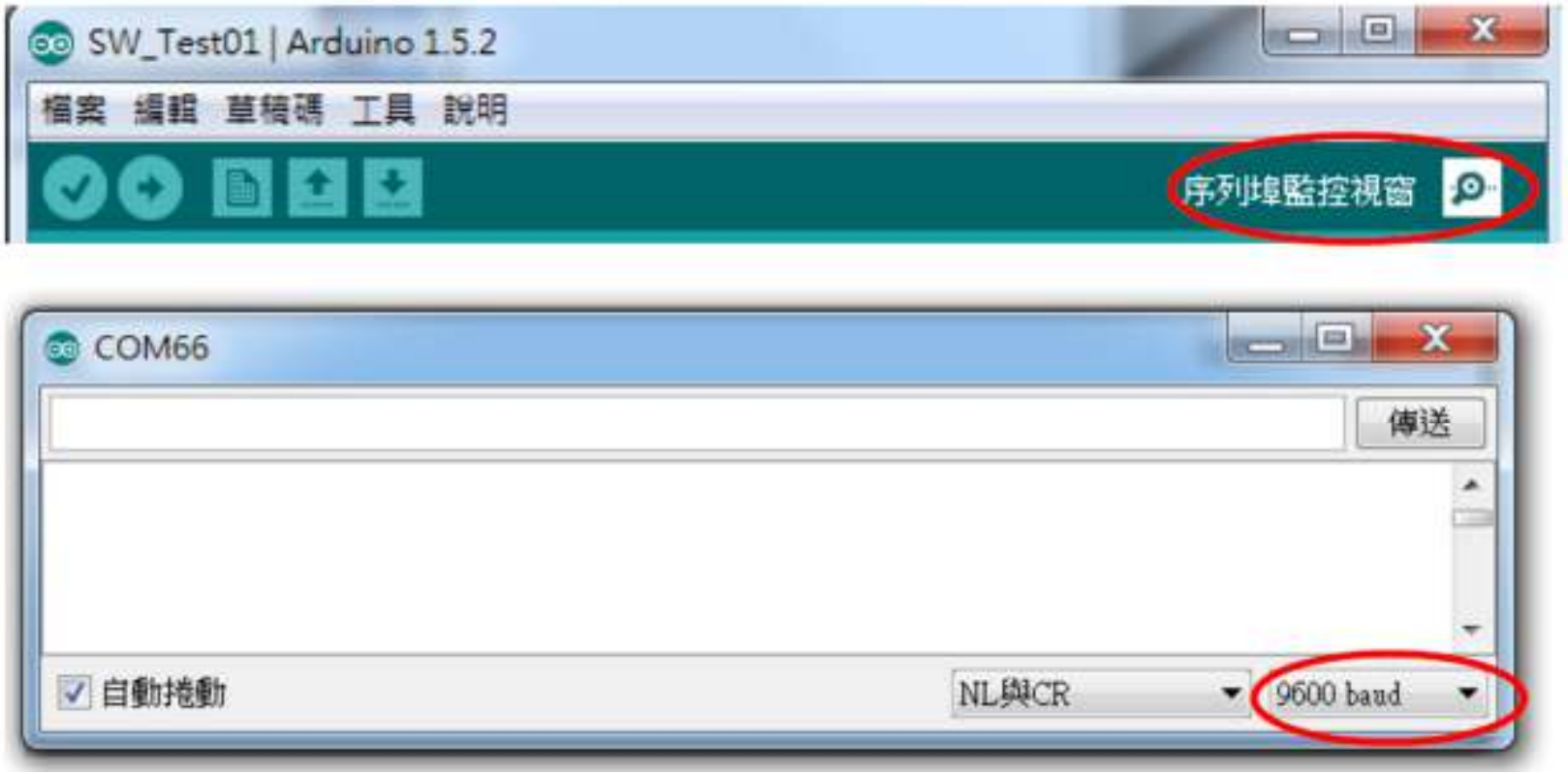
Appsdüino

練習 1：使用RGB LED模擬霹靂燈

Appsduino

串列監控視窗(Serial Monitor)

Arduino 控制板可透過 Serial 物件(UART/序列埠)與 PC 或其他設備互傳資料與訊息，也可據此將變數值送至 PC 的監控視窗以做 debug(除錯)使用，可利用 IDE 整合開發環境右上角的 圖示來開啟序列埠監控視窗，如下所示：



串列(Serial)通訊函數

而序列物件 UART 常用之通訊函數如下 ：

- Serial.begin(speed)

  通訊時雙方速度需一樣方能互傳資料，因此首先需設定傳輸之速度(Baud Rate)，可設定之速度 ： 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 or 115200bps。而預設之資料格式為 8 data bits, no parity, one stop bit。

  例 ： Serial.begin(9600) ; //設定傳輸速度為 9600 bps

- int Serial.available()

  讀取序列埠的資料狀態，回傳讀取到的字元數，若無資料則回傳 0

```
例 ： if (Serial.available() > 0)
        incomingByte = Serial.read();// read the incoming byte
```

Appsduino

串列(Serial) : read(讀取)函數

- int Serial.read()
  讀取序列埠的第一個 Byte 資料，若無資料則回傳-1
  例 :

```
int incomingByte = 0; // for incoming serial data

void setup() {
   Serial.begin(9600);// opens serial port, sets data rate to 9600 bps
}

void loop() {// send data only when you received data
   if (Serial.available() > 0) {//先檢查是否序列埠有資料 ?
      incomingByte = Serial.read();// read the incoming byte
      Serial.print("I received: ");// say what you got
      Serial.println(incomingByte, DEC);
   }
}
```

Appsduino

串列(Serial)：print(輸出)函數

- Serial.print(val)，Serial.print(val,format)

  將變數 val 的數值用文字(ASCII)的型式傳至序列埠，format 為數字進制或小數點取幾位的設定，可為 BIN、OCT、DEC、HEX 或 0、1、2、3…(預設為 DEC/10 進制小數點 2 位)

  例 : Serial.print(78,BIN) ; // 用二進制來表示，即"1001110"

  Serial.println(1.23456, 3) ; //傳送 "1.234"即小數點取 3 位

- Serial.println(val)，Serial.println(val,format)

  與 Serial.print 一樣，都是將變數 val 的數值用文字(ASCII)的型式傳至序列埠，但 println 會加上歸位字元(Carriage Return,ASCII 13 或' \r')與下一行字元(new line,ASCII 10 或' \n')

例 :

```
int analogValue = 0;//variable to hold the analog value
void setup() {
    Serial.begin(9600); //設定序列埠速度為 9600bps
}
```

Appsduino

串列(Serial)：print(輸出)函數

```
void loop() {
    analogValue = analogRead(0); // read the analog input on pin 0
    Serial.println(analogValue);          //以 10 進制文字型式傳送/顯示
    Serial.println(analogValue, DEC);  //以 10 進制文字型式傳送/顯示
    Serial.println(analogValue, HEX);  //以 16 進制文字型式傳送/顯示
    Serial.println(analogValue, OCT);  //以 8 進制文字型式傳送/顯示
    Serial.println(analogValue, BIN);  //以 2 進制文字型式傳送/顯示
    delay(10);              // delay 10 milliseconds before the next reading
}
```

- Serial.write(val), Serial.write(str), Serial.write(buf,len)

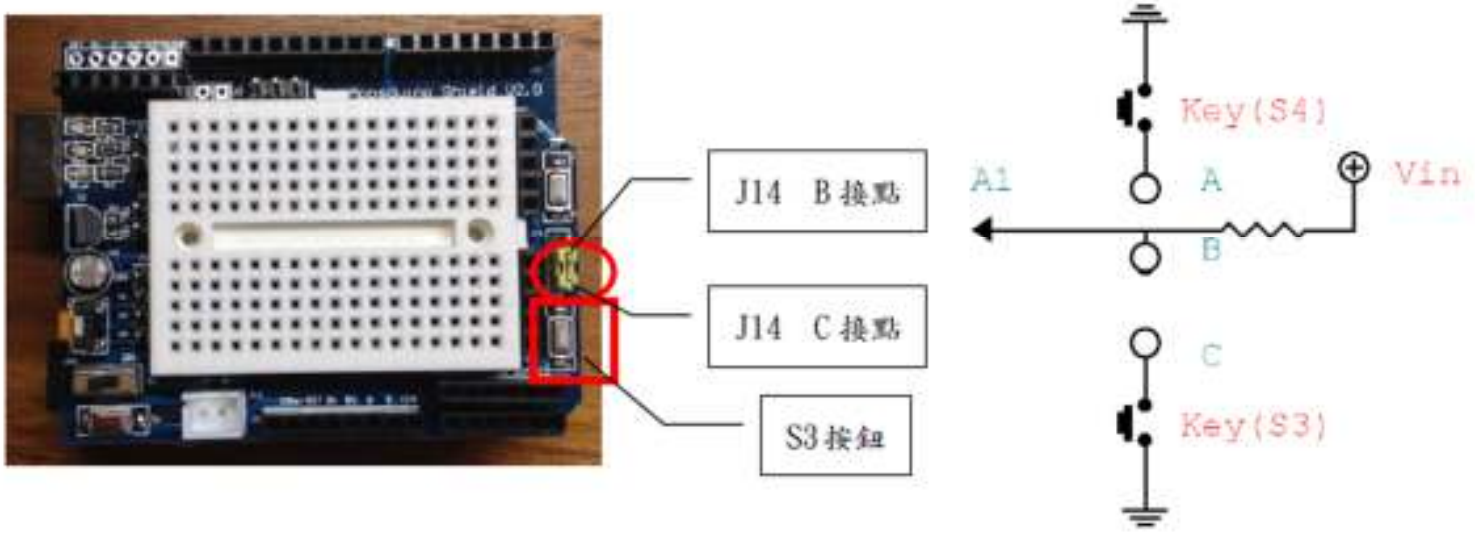
Serial.print 是將變數 val 的數值用文字(ASCII)的型式傳至序列埠，但 Serial.write 則是將資料以數字(binary)的型式傳至序列埠，可能是一個或連續多個位元組

例 : Serial.write(45); // send a byte with the value 45
int bytesSent = Serial.write("hello"); //傳送字串 hello" 並回傳字串長度

Appsduino

Lab 4：按鍵/數位IO(A1)

擴充板上內建兩個按鈕(S3/S4)，如下圖。S3 按鈕輸出的訊號會送到按鈕上方 J14 的C 接點，J14 的B 接點則連通 analog A1 接腳。A0-A5 可當類比輸入，也可當數位 IO 使用，此實驗將其當數位 IO 來使用。按鈕 S4 則需透過杜邦線從 A 接點連至選擇的數位或類比 I/O 腳位



當按下按鈕時，B 點產生 LOW 訊號，放開按鈕 B 點產生 HIGH 訊號。

硬體接線：將 J14 短路，讓按鍵訊號直接連到 A1 腳位，如右上示意圖。

Appsduino

Lab 4 : Key Program

程式： 按下按鈕，連接 D13 的綠色 LED 會點亮。放開按鈕則熄滅。

```
const int SW_pin = A1;  // 指定 A1 接腳連接按鈕
void setup() {
  Serial.begin(9600);           // 初始化序列埠監控視窗的傳輸速度
  pinMode(SW_pin,INPUT_PULLUP); // 規劃 SW_PIN 為數位輸入接腳並啟動其內建上拉電阻
  pinMode(13, OUTPUT);          // 規劃 13 接腳 為數位輸出接腳
}
void loop() {
  int val = digitalRead(SW_pin);  // 取得輸入按鈕的狀態
  if (val==LOW) {  // 依照電路設計  按下按鈕=0  放開按鈕=1
    digitalWrite(13, HIGH);  //  按下按鈕=>燈亮
  } else {
    digitalWrite(13, LOW);   //  放開按鈕=>燈滅
  }
  Serial.println(val);        // 將變數 val 的值顯示到序列埠監控視窗
  delay(1000);          // 程式暫停 1000 毫秒（相當於 1 秒鐘）
}
```

程式執行結果：按下按鈕 Green 燈點亮，放開按鈕 Green 燈熄滅。若同時開啟序列埠監控視窗，

Appsdüino

串列監控視窗輸出(Serial Monitor)

程式執行結果：按下按鈕 Green 燈點亮，放開按鈕 Green 燈熄滅。若同時開啟序列埠監控視窗，



並且將傳輸鮑率設定與程式指定的速度相同(9600)，就可以看到 Appsduino 傳回來的資料，按下按鈕傳回 0，放開按鈕傳回 1。



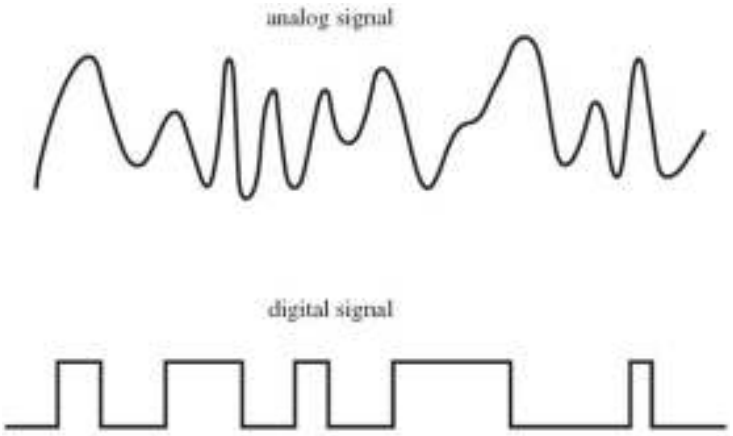
Appsduino

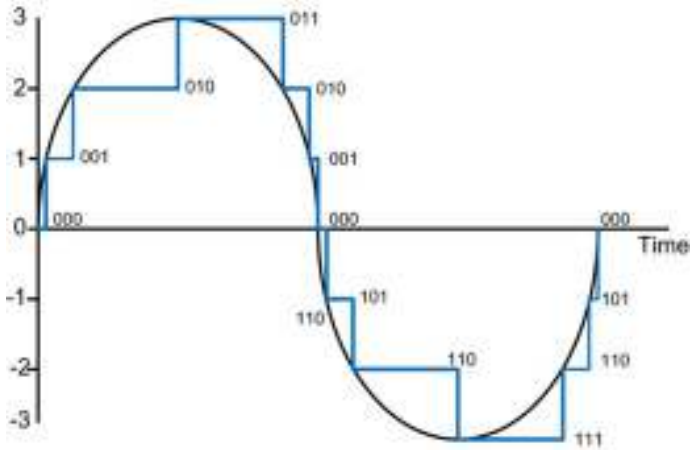
類比訊號 VS 數位訊號

- analog 指類比訊號，真實世界中很多現象..包括聲音大小、光線明暗、空氣質量、溫濕度都是以analog 的方式呈現

- digital指數位訊號，只有兩種狀態 : HIGH/LOW(1/0)，因電腦只能透過電子訊號電壓的高與低來辨識與處理資料，digital訊號普遍用於目前的影、音、網路資料、聲音或電腦檔案儲存上

- 若要使用電腦處理真實世界中的類比訊號，則需透過 Analog to Digital 轉換器轉換，方能達成(如下圖)





A/D Conversion

Appsduino

Functions：類比輸入/輸出(A0~A5)

UNO 板內建六個 10bit 的類比輸入(A0-A5)，可將類比電壓(0-5V)轉換成數值 0-1023

- int analogRead(pin)

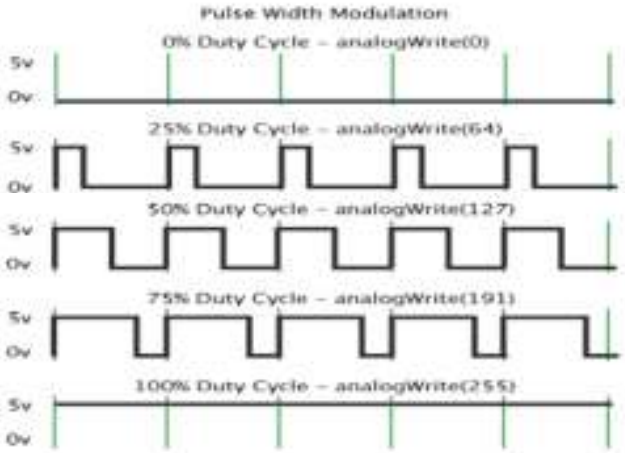
  從指定的pin 讀取數值，並進行 10-bit 的數字轉換，這個方法將輸入的 0-5V 電壓值轉換為 0-1023 的整數值。5V/1024 = 0.0049v 約每 4.9mV(毫伏) 為一個數值。pin 表示為 0～5

  例 ： val = analogRead(0); //讀出類比腳位 0 的值並指定給 val 變數

- analogWrite(pin,Value)

  PWM(Pulse Width Modulation)類比輸出函數，將所指定的pin 寫入數值，據此改變腳位的輸出電壓值。pin 表示 3/5/6/9/10/11，value 表示為 0～255。比如可用於電機 PWM 調速或音樂播放。例如：輸出電壓 2.5 伏特（V），該值大約是 128。Arduino PWM 頻率預設約 490Hz。

  例 :analogWrite(9,128); // 輸出電壓約 2.5 伏特（V）



map/constrain 轉換函數

map/constrain 函數 :

- map(value,fromLow,fromHigh,toLow,toHigh)

  map 函數將一個數值 value 重新從[frowLow ~ frowHigh]區間對應至[toLow ~ toHigh]區間，這對於在不同數值系統之間的轉換相當方便

  例 : val = map(val, 0, 1023, 0, 255); //將 val 從[0, 1023]區間轉換為[0, 255]區間，並將新數值儲存至變數 val

- constrain(x,a,b)

  限制 x 值在 a 與 b 之間，x, a, b 可為任何型態資料(ex: int, long, char, …)

  傳回值 : x 或 a 或 b

  x : a <= x <= b

  a : x < a(下限)

  b : x > b(上限)

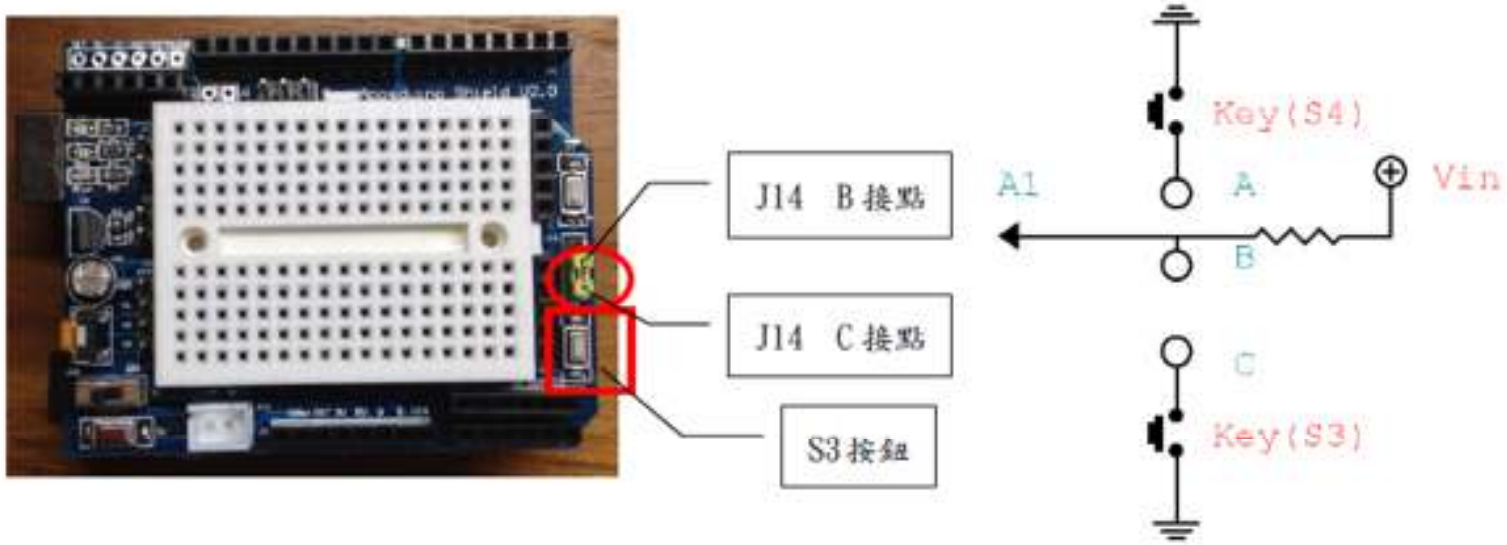
  例 : sensVal = constrain(sensVal, 10, 150);

  // limits range of sensor values to between 10 and 150

Appsduino

Lab 4：按鍵/ 類比 IO(A1)

之前的按鍵實驗使用數位 I/O 的方式，把類比 A1 腳位當數位 I/O 來使用，因為 A0-A5 可當類比輸入，也可當數位 IO 使用。當按下 S3 按鈕，輸出訊號會送到按鈕上方 J14 的 C 接點，J14 的 B 接點則連通 analog A1 接腳(如下圖)，此實驗將其當類比 IO 來使用。



當按下按鈕時，B 點產生從高電壓到低電壓的電壓變化訊號，放開按鈕 B 點則產生由低電壓到高電壓的訊號。我們可透過 ananlgRead(A1)函數讀取類比 A1 腳位的電壓值來辨別按鍵是否被按下。

硬體接線：將 J14 短路，讓按鍵訊號直接連到 A1 腳位，如右上示意圖。

Appsduino

Lab 4-1：按鍵/類比IO Program

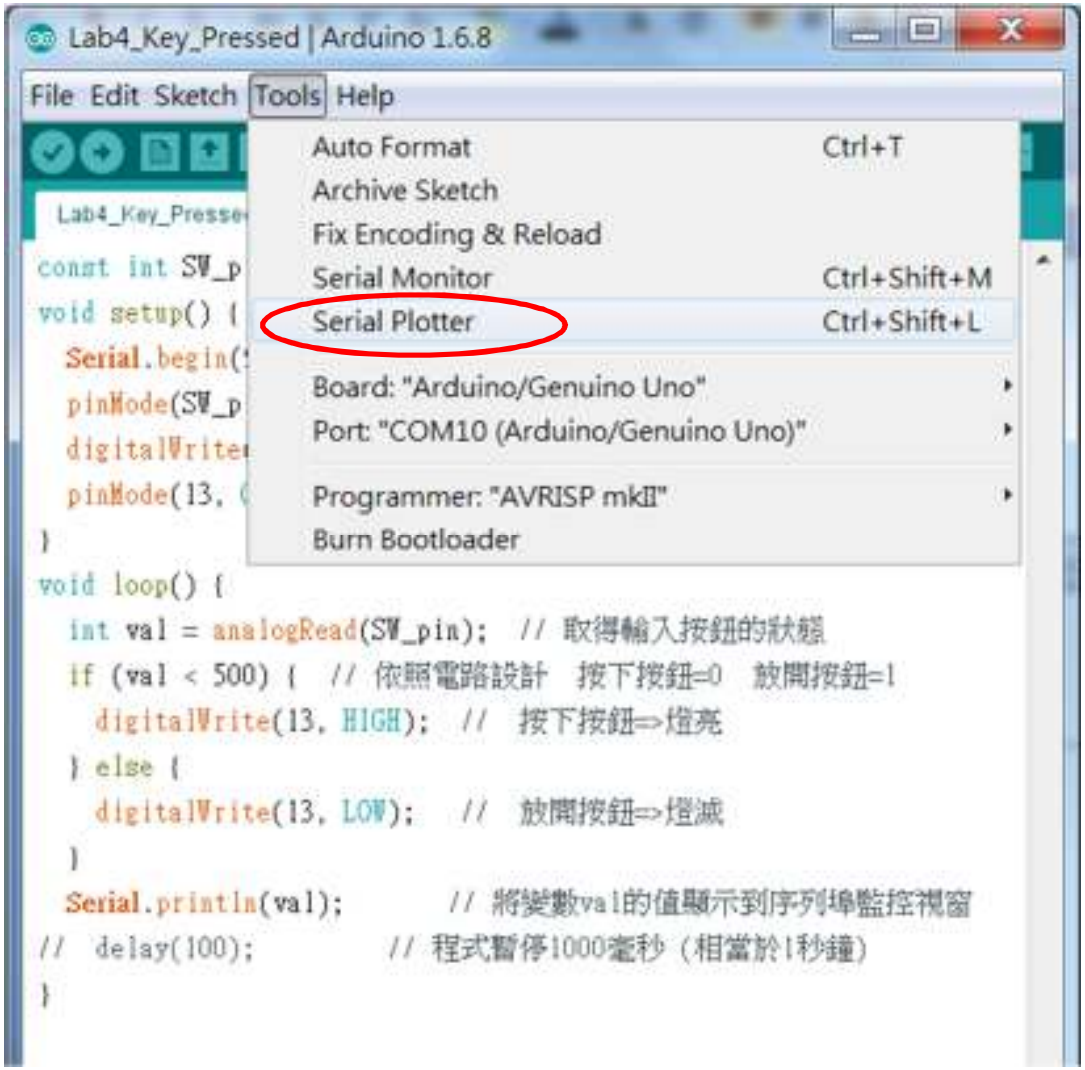
程式：按下按鈕，連接 pin13 的 LED 會點亮。放開按鈕則熄滅。

```
const int SW_pin = A1;  // 指定 A1 接腳連接按鈕
void setup() {
  Serial.begin(9600);              // 初始化序列埠監控視窗的傳輸速度
  pinMode(SW_pin, INPUT);          // 規劃 SW_PIN 為數位輸入接腳
  digitalWrite(SW_pin, HIGH);      // 開啟 SW_PIN 內建上拉電阻
  pinMode(13, OUTPUT);             // 規劃 13 接腳 為數位輸出接腳
}
void loop() {
  int val = analogRead(SW_pin);    // 取得輸入按鈕的狀態
  if (val < 500) {  // 依照電路設計  按下按鈕時為低電位,放開按鈕高電位
    digitalWrite(13, HIGH);  //  按下按鈕=>燈亮
  } else {
    digitalWrite(13, LOW);   //  放開按鈕=>燈滅
  }
  Serial.println(val);          // 將變數 val 的值顯示到序列埠監控視窗
  delay(1000);            // 程式暫停 1000 毫秒（相當於 1 秒鐘）
}
```

使用類比的方式來處理

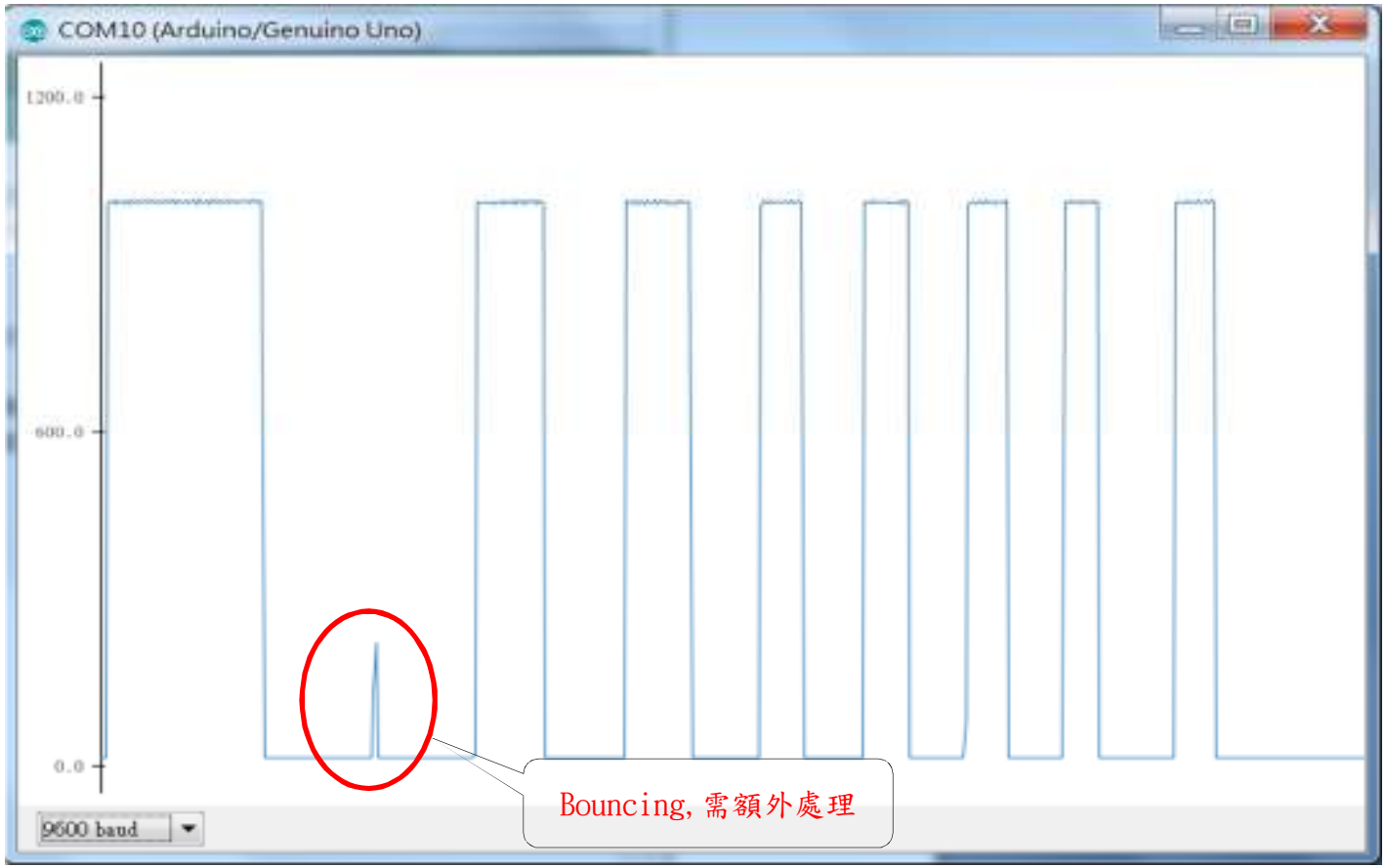
Appsduino

Lab 4-1：按鍵/類比(使用示波器)



Appsduino

Lab 4-1：按鍵/類比(使用示波器)



Appsduino

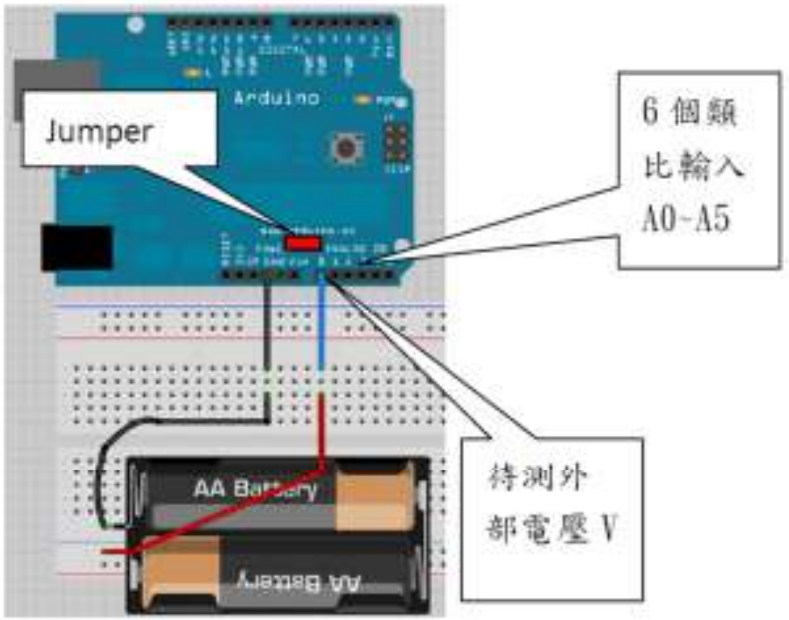
Lab 6：電壓檢測(A0)

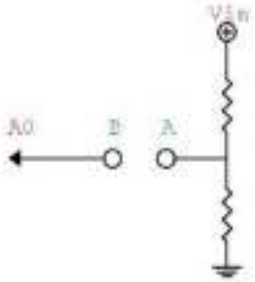
Appsduino UNO 內建六個類比輸入(A0~A5)，解析度 10bit，可以用來將類比電壓(0~5V)轉換成數位值(0~1023)，我們可以設計簡單的程式來偵測外部電壓。

如右圖簡單將外部電源負極接到 GND，正極接到 Analog in 的 A0 接腳。

附記：Appsduino 控制板 A0 上方有一個 jumper(如右圖紅方框位置)，當 Appsduino UNO 使用外接鋰電池時，若將這個 jumper 短路，會將電池 Vin 電壓的 1/2 分壓連接至 A0，因此即可利用 A0 監測正使用的鋰電池電壓，所量之電壓值為 1/2 Vin，因此，真正的電壓需乘上 2。

若要使用 A0 做其他用途，請將 A0 上方的 jumper(A、B)拔開使其開路，即可使用。





Appsduino

Lab 6：電壓檢測(A0)

範例 ：使用 A0 檢測從 Vin 輸入的電池電壓(所量之電壓值 x 2 即為 Vin 之電壓)

```
const int Vpin=0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  int raw_data = analogRead(Vpin);  // 依據電壓傳回介於 0-1023 的值
  Serial.print("raw data : "); Serial.print(raw_data);
  float map_data = raw_data / 1024.0 * 5.0;
  Serial.print("    voltage data : "); Serial.println(map_data);
  delay(1000);
}
```

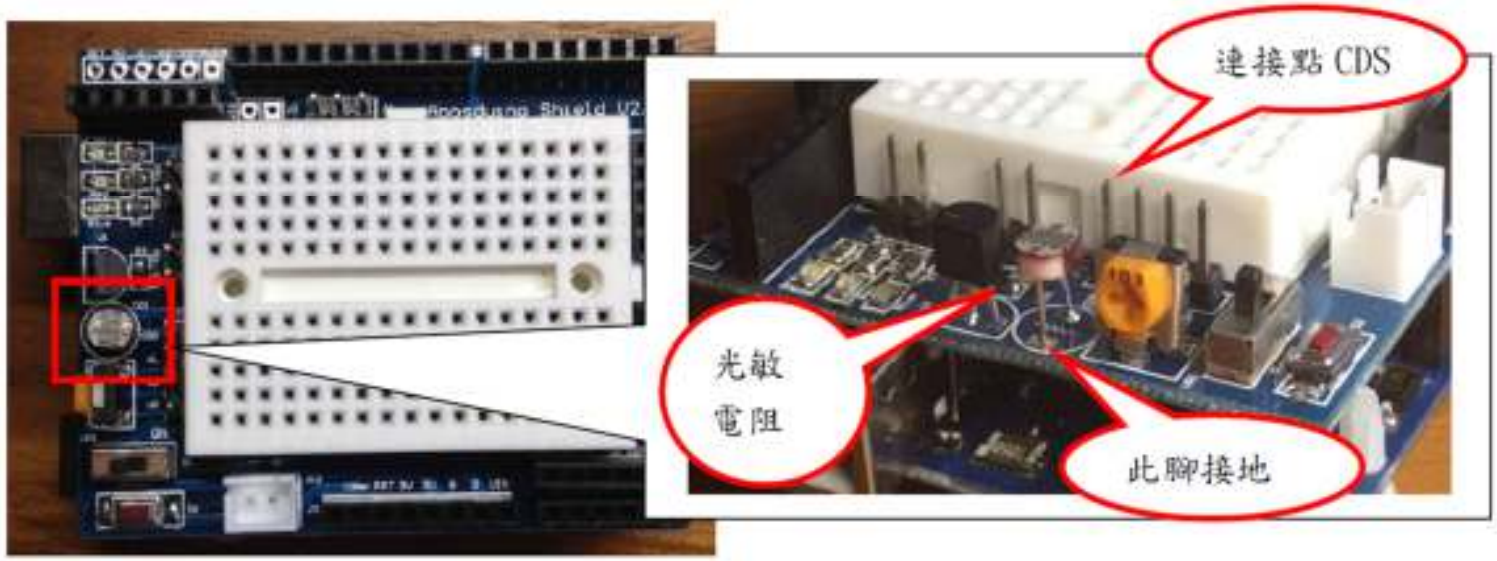
輸出:每隔一秒顯示偵測的原始資料(0~1023)
以及電壓值(0~5V)



Appsduino

Lab 8：Analog I/O(CDS實驗)

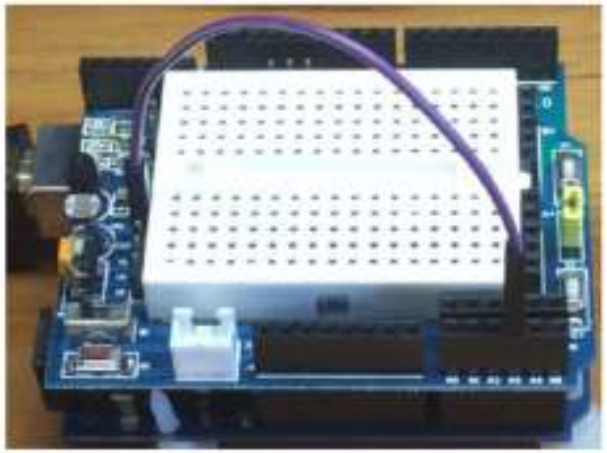
擴充板上內建一個光敏電阻(CDS)，他會感應光線的變化，並且讓電阻值產生變化，光線亮時電阻值變小，光線暗時電阻值變大，電阻值約為在幾百歐姆(亮) ~ 100K(暗)之間變化。



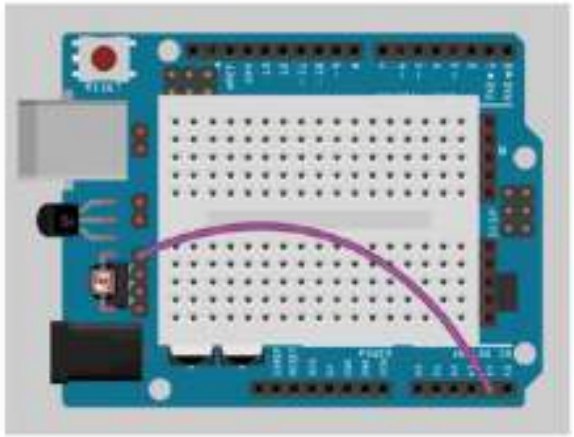
Appsduino

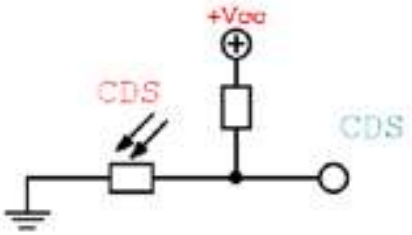
Analog I/O : CDS實驗

光敏電阻一腳已接地，另一腳連接至 CDS。我們將 CDS 直接連接 analog A4 接腳，如右圖：



將 Appsduino 接上 USB 通電，實際測量 A4 點電壓，當光敏電阻受光時，電阻值小，A4 點電壓偏低(約 0.2V)，若用手指遮擋光敏電阻(暗)，電阻值變大，A4 點電壓偏高(約 4.0V)。我們可以實際寫一個程式來偵測 A4 接腳的電壓變化，並且顯示在序列埠監控視窗。





電路示意圖

Appsduino

CDS Program

程式如下 :

```
void setup() {
  pinMode(A4, INPUT_PULLUP);  // 開啟 A4 內建上拉電阻
  Serial.begin(9600);         // 初始化序列埠監控視窗的傳輸速度 9600bps
}
void loop() {
  // 讀取類比輸入接腳 A4 的電壓，回傳介於 0-1023 數值
  int val = analogRead(A4);
  Serial.println(val);  // 將變數 val 的值顯示到序列埠監控視窗
  delay(500);           // 程式暫停 500 毫秒（相當於 0.5 秒鐘）
}
```

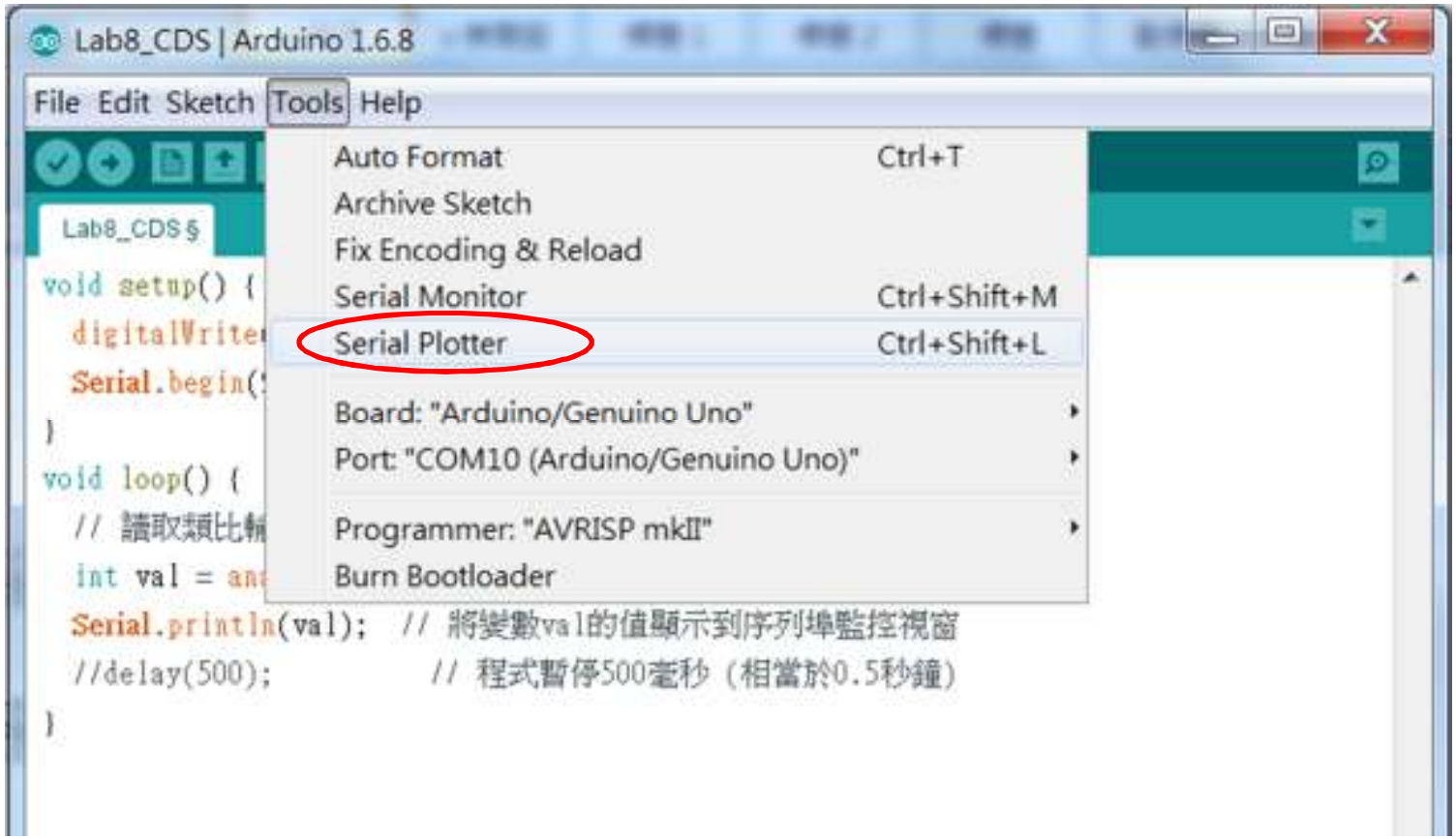
執行結果:

用手指阻斷光敏電阻受光面，就會看到序列埠監控視窗的輸出值不斷改變。



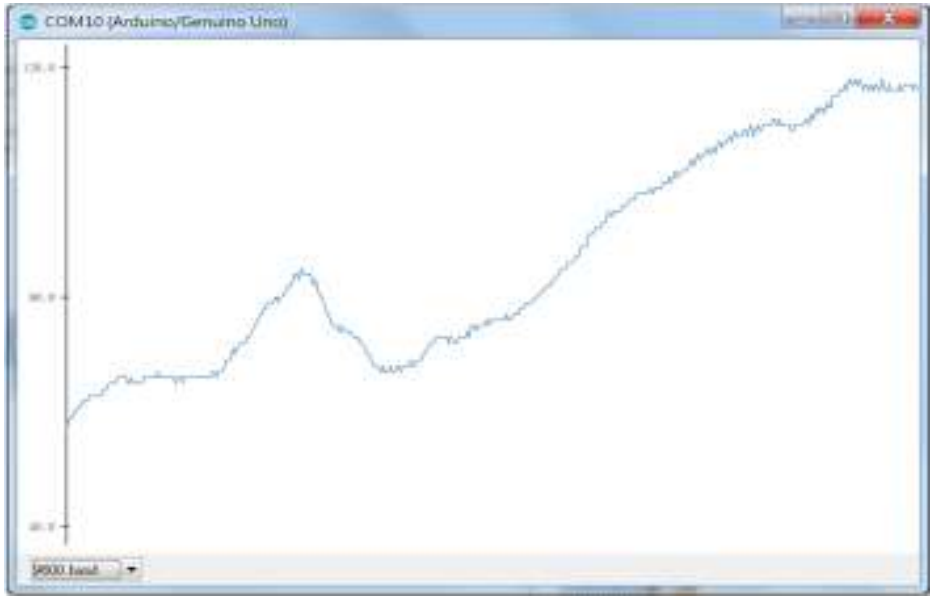
Appsduino

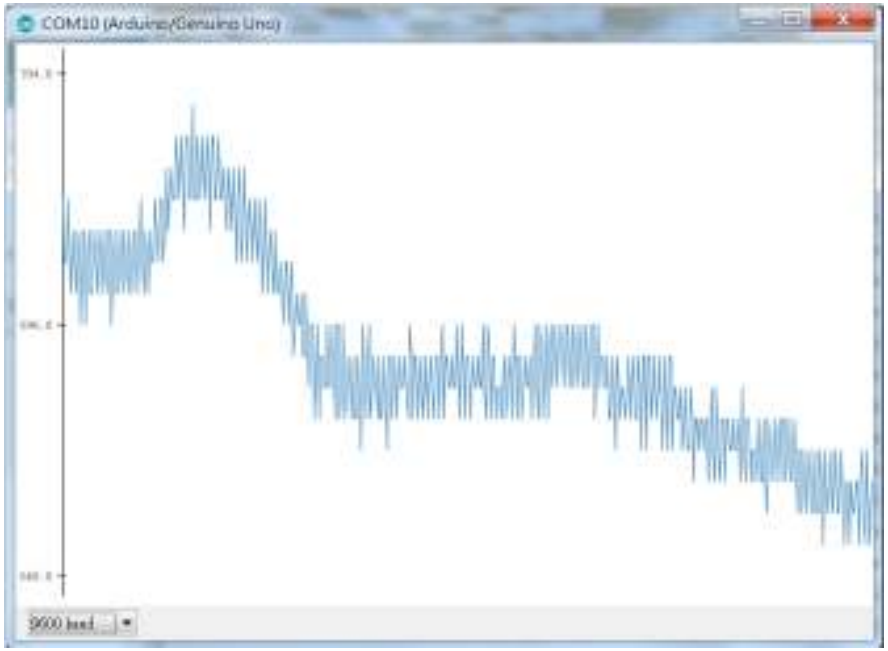
CDS Program (使用示波器)



Appsduino

CDS Program (使用示波器)





Appsduino

Lab 8-1：利用CDS 控制 Led On/Off

```
#define BlueLed   11

void setup() {
  digitalWrite(A4, HIGH);  // 開啟 A4 內建上拉電阻
  pinMode(BlueLed,OUTPUT) ;
  Serial.begin(9600);  // 初始化序列埠監控視窗的傳輸速度 9600bps
}

void loop() {    // 讀取類比輸入接腳 A4 的電壓，回傳介於 0~1023 數值

int val = analogRead(A4);
Serial.println(val);  // 將變數val的值顯示到序列埠監控視窗

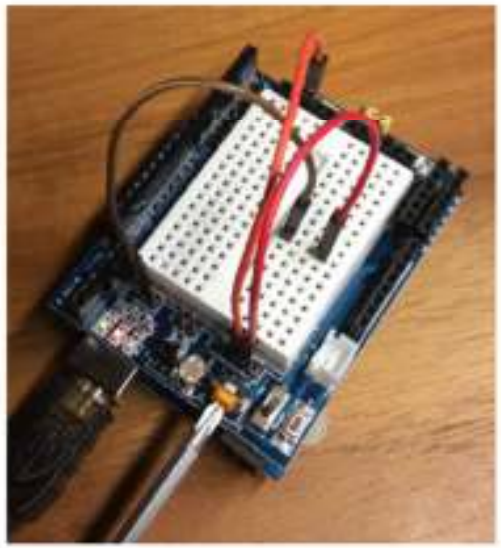
if(val > 500)
   digitalWrite(BlueLed,HIGH) ; //Turn on Led
else
   digitalWrite(BlueLed,LOW) ; // Turn off Led

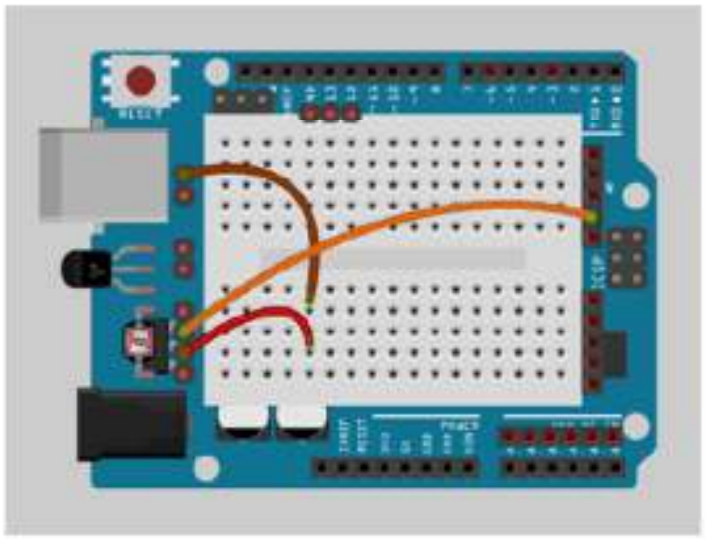
delay(500);          // 程式暫停500毫秒（相當於0.5秒鐘）
}
```

Appsduino

Lab9：LED亮度控制(使用可變電阻)

Appsduino 擴充板內建10KΩ的可變電阻，其三支腳分別對應VR/VC/VL腳位，可利用這些腳位並旋轉旋鈕以獲得所需的阻值。首先將VL針腳連至5V，紅色Led針腳連至麵包板，再連至VC針腳(如圖)，用起子旋轉旋鈕以調整電阻值，即可看到紅色Led亮度的變化，此乃以硬體的方式，即調整外在電阻值以改變電流大小來調控Led的亮度。下個實驗即以軟體的方式(PWM的類比輸出) 來調整Led的亮度。

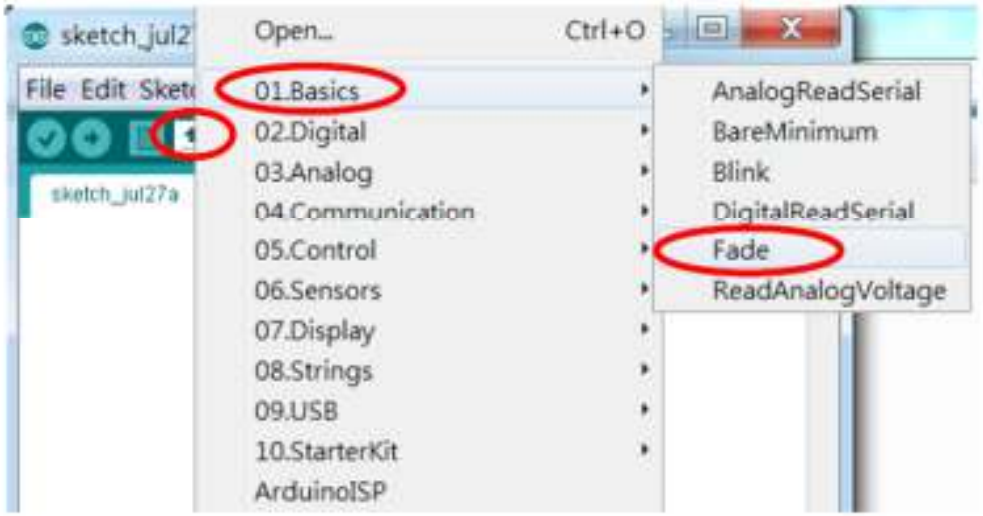


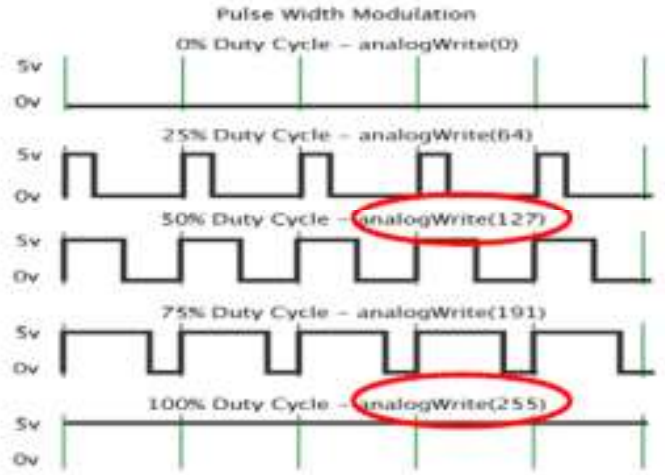


Appsduino

Lab 10：LED亮度控制(類比PWM輸出)

Arduino UNO 的 I/O 腳位 3、5、6、9、10、11 共 6 Pins，具有 PWM(Pulse Width Modulation) 類比輸出功能，可利用 analogWrite(pin, value) 函數將所指定的 pin 寫入數值(value) 0~255，據此改變腳位的輸出電壓值。因此可利用此類比輸出功能來控制馬達的轉速或 LED 燈的明暗度，底下範例為利用此功能來讓擴充板上的紅、藍 Led 漸亮/漸暗，首先將紅、藍 Led 分別用杜邦線連至 D10、D9，如下頁圖，程式請開啟 Arduino IDE 內建範例 ： \Basics\Fade 如下 ：





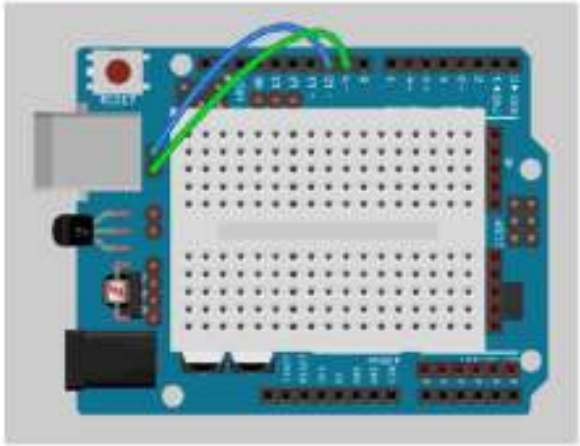
Appsduino

Lab 10：LED亮度控制(類比PWM輸出)

```
/* Program Name : Fade
  This example shows how to fade an LED on pin 9
  using the analogWrite() function. */

int led = 9;                // the pin that the LED is attached to
int brightness = 0;         // how bright the LED is
int fadeAmount = 5;         // how many points to fade the LED by

void setup()  {
  pinMode(led, OUTPUT); // declare pin 9 to be an output:
}
void loop()  {
  analogWrite(led, brightness);     // set the brightness of pin 9:
  brightness = brightness + fadeAmount; // change the brightness for next time through the loop:
  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  delay(30); // wait for 30 milliseconds to see the dimming effect
}
```



可加上 pinMode(10, OUTPUT); // red Led

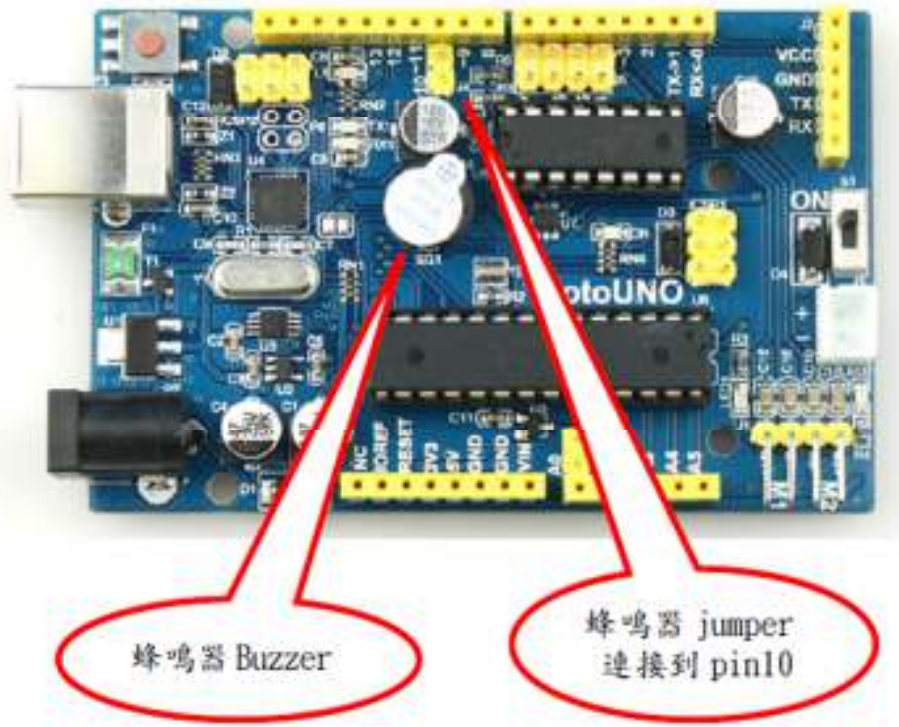
加上 analogWrite(10,255- brightness);// red Led

可更改變化的時間

蜂鳴器(Buzzer)

Appsduino MotoUNO 控制板上安裝一個蜂鳴器(Buzzer)，若要使用 D10 控制 buzzer，就必須將上方的 jumper 短路，若 D10 送出 HIGH，蜂鳴器就會發出嗶聲。程式如下 ：

```
const int Buzzer=10;  // 定義常數
void setup() {
pinMode(Buzzer,OUTPUT);//設定D10為數位輸出
}
void loop() { //對 Buzzer 不斷輸出高、低電位，讓 Buzzer 快速發出振動聲
  digitalWrite(Buzzer, HIGH);
  delay(100); //delay()函數可以控制發聲的頻率
  digitalWrite(Buzzer, LOW);
  delay(100);
}
```



Appsduino

Melody(音律/聲音)函數

Arduino 控制板可透過 tone 的函數去驅動 Buzzer(蜂鳴器)或 Speaker(小喇叭)產生音律或警示聲響，以呈現某個事件的發生與處理

- tone(pin,frequency), tone(pin,frequency,duration)
  tone(pin, frequency)乃在指定的 pin(腳位)上產生所設定頻率的方波(50% duty cycle)，而 duration 則指此方波持續的時間，單位為毫秒(ms)，若要停止，則使用 noTone(pin) 即可

例 1 : tone(8, 262, 500) ; //在 Pin 8 發出 Do(262Hz)的聲音約 0.5 秒(500ms)

```
例 2 : int Frequency[] = { //設定 Do Re Mi Fa So La Si Do(低音頻率)
           262, 294, 330, 349, 392, 440, 494, 523 };
       for(int i=0 ; i < 8 ; i++) {
          tone(8, Frequency[i], 500) ;//發出 Frequency 陣列內之聲音各 0.5 sec
          delay(500) ; } // delay 500ms
       noTone(8) ;    // 停止 Pin 8 發出聲音
```

Appsduino

各種音調的頻率表

下表為各種音調的頻率表（正常的人耳能聽到 20 赫茲到 20,000 赫茲頻率的聲音）

		Do	Re	Mi	Fa	So	La	Si
低音	頻率(Hz)	262	294	330	349	392	440	494
	週期(s)	0.003817	0.003401	0.00303	0.002865	0.002551	0.002273	0.002024
	半週期(us)	1908	1701	1515	1433	1276	1136	1012
中音	頻率(Hz)	523	587	659	698	784	880	988
	週期(s)	0.001912	0.001704	0.001517	0.001433	0.001276	0.001136	0.001012
	半週期(us)	956	852	759	716	638	568	506
高音	頻率(Hz)	1046	1175	1318	1397	1568	1760	1976
	週期(s)	0.000956	0.000851	0.000759	0.000716	0.000638	0.000568	0.000506
	半週期(us)	478	426	379	358	319	284	253

Appsdüino

Lab 11 Buzzer 程式(delay)

```
#define   Buzzer   10    // 定義Buzzer腳位
void setup() {
  pinMode(Buzzer, OUTPUT); // 設定D8為數位輸出

  for(int i=0;i < 8 ; i++) {
    digitalWrite(Buzzer, HIGH); //對Buzzer不斷輸出高、低電位，讓 Buzzer 快速發出振動聲
    delay(100); //delay()函數可以控制發聲的頻率
    digitalWrite(Buzzer, LOW);
    delay(100);
  }
}
void loop()
{
}
```

Appsduino

Lab 11_1 : buzzerMeledy 程式(tone)

```
//--------- AmeiAmei Melody 2013/4/21 ---------
#include "pitches.h"
#define NO_SOUND 0 // make the rests in music
#define LedIndicator 10 // Green Led as Indicator

//array of notes
int melody[] = {
  NO_SOUND,NOTE_E4,NOTE_E4,NOTE_E4,NOTE_D4,NOTE_C4,NOTE_B3,NOTE_A3,NOTE_D4,NOTE_D4,NOTE_D4,
  NO_SOUND,NOTE_A3,NOTE_A3,NOTE_A3,NOTE_A3,NOTE_C4,NOTE_D4,NOTE_E4,
  NO_SOUND,NOTE_E4,NOTE_E4,NOTE_E4,NOTE_D4,NOTE_C4,NOTE_B3,NOTE_A3,NOTE_D4,NOTE_D4,NOTE_D4,//今天今天你要老實講
  NO_SOUND,NOTE_B3,NOTE_B3,NOTE_B3,NOTE_B3,NOTE_C4,NOTE_D4,NOTE_C4,//    我是否有希望
};

// note duration: 1 = whole note, 2 = half note, 4 = quarter note, 8 = eighth note, etc.
int noteDurations[] = {
  2,1,2,2,2,2,2,1,2,2,-2, //-2 代表延長兩個全音(兩拍)
  2,2,2,2,2,2,1,-4,  //-4 代表延長4個全音(4拍)
  2,1,2,2,2,2,2,1,2,2,-2,
  2,2,2,2,2,2,1,-4, //
};

#define Buzzer                8   // enable Pin 8 buzzer function
//-------------------------------------------------
void setup()
{
  pinMode(Buzzer, OUTPUT);
  pinMode(LedIndicator,OUTPUT) ;
  MelodyPlay() ; // wait for melody play finished
 }
```

Appsduino

Lab 12 : buzzerMeledy 程式(tone)

```
//--------------- Main Program -----------------------
void loop()
{
 }

//--------- AmeiAmei Melody Player --------------------------
// Play the melody based on the Notes in melody and NotesDurations array
int pace = 350; // change pace of music("speedy")

void MelodyPlay() {
  int melodysize = sizeof(melody)/2 ;// sizeof(var) return the byte count for the var
  int duration ;

  for (int Note = 0; Note < melodysize; Note++) {//counter of Notes (54 limit the array)
    duration = pace/noteDurations[Note];//Adjust duration with the pace of music
    if(duration < 0)
       duration = pace * abs(noteDurations[Note]) ; // it means long note
//   Serial.println(duration) ;
    tone(Buzzer,melody[Note],duration); //Play note
    analogWrite(LedIndicator,melody[Note]/2) ;

// to distinguish the notes, set a minimum time between them.
    delay(duration*1.2);
  }
  noTone(Buzzer) ;
  analogWrite(LedIndicator,0) ; //turn off Led
}
```

Appsduino

練習 2：Buzzer模擬門鈴或一首歌曲

Appsduino

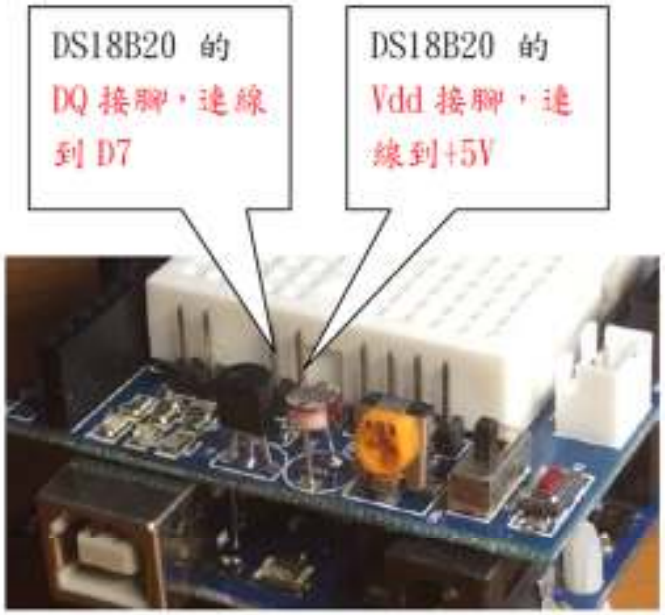
DS18B20 數位溫度計

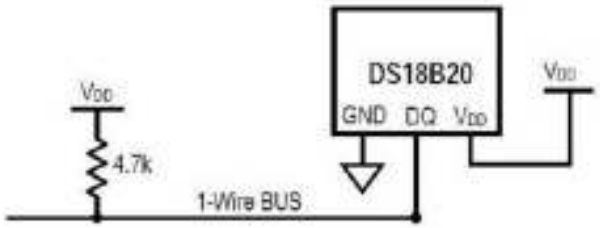
Appsduino 擴充板上已安裝一個 DS18B20 溫度感測 IC，可偵測溫度範圍：攝氏-55 度~125 度，解析度為 9 ~ 12 bit，數位轉換時間約為 40 ~ 750ms，可應用於溫度檢測、恆溫控制、或各種溫度控制系統。

DS18B20 使用單線通訊協定(1-wire)傳輸資料，詳細原理及資料格式可在網路上搜尋到。在 Arduino 社群已經有人將傳輸方式寫成程式庫，我們只要將標頭檔(header file)引入，直接呼叫函式庫就可以取的溫度值。但需將 library 拷貝至 arduino IDE 目錄之 libraries 內，並重新啟動 Arduino 環境。

擴充板上 J19 的兩根排針分別連至 DQ、Vdd(如右圖)，請將 DQ 接到 D7，Vdd 接到+5V，接下來使用程式由 DQ 接腳取得數位溫度值。



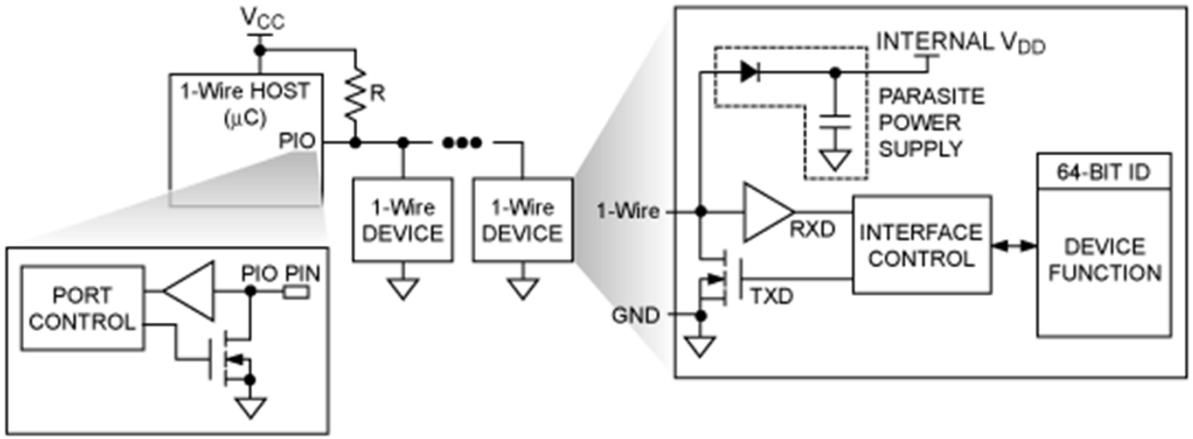




Appsduino

1-Wire 通訊介面

- 1個 Host/Master 與 1或多個Slaves利用1-Wire傳送訊息
- half-duplex 雙向通訊
- 64 bits ID : store on ROM
- Pull up resistor (R) : 4.7K



Source : http://www.maximintegrated.com/app-notes/index.mvp/id/3989

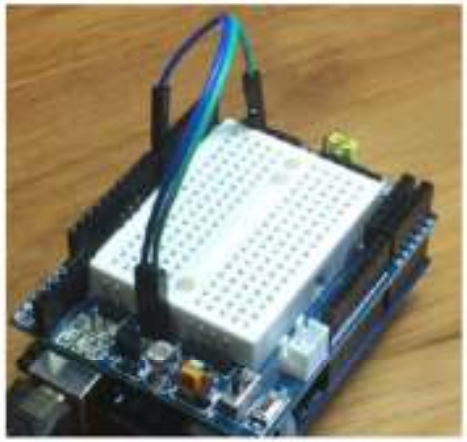
Appsduino

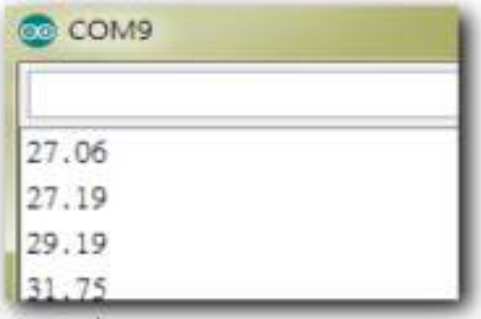
Lab 13：DS18B20取得環境溫度

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define DS18B20_pin 7 //定義常數
OneWire ds(DS18B20_pin);  //   初始化 ds 物件
DallasTemperature DS18B20(&ds);  // 初始化 DS18B20 物件

void setup(void) {
  Serial.begin(9600);
  DS18B20.begin(); // 啟動 DS18B20
}
void loop() {
  DS18B20.requestTemperatures(); //由 DS18B20 物件，偵測環境溫度
  float val = DS18B20.getTempCByIndex(0); // 取得溫度值
  Serial.println(val,2);
  delay(1000);
}
```

請將所附之 libraries 拷貝至安裝 arduino IDE 目錄之 libraries 資料夾內(若有相同，取代之)





輸出結果:每隔一秒更新攝氏溫度值

Lab 14：溫度控制

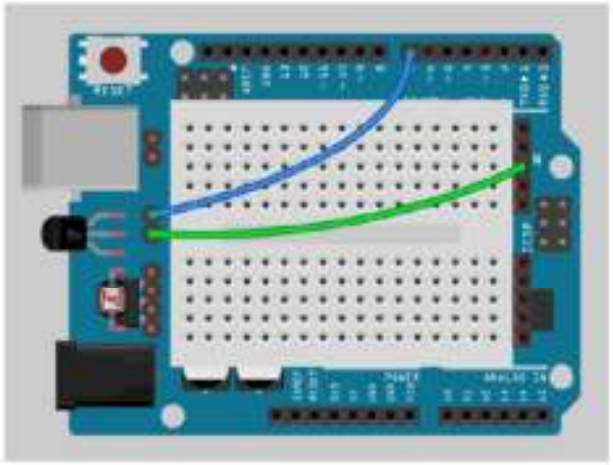
設計一個溫度監控程式，當溫度到達上限值時，就啟動警示燈閃爍，若到達危險溫度就發出警報聲，提醒人們趕快處理。硬體線路如上節，程式如下：

```
#include <Streaming.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define DS18B20_pin 7  // 定義常數
const unsigned long IntervalTime = 1000;
const unsigned int AlarmTemp = 40;  //警戒溫度
const unsigned int UpperTemp = 30;  //上限溫度

unsigned long CurrTime, PrevTime;

OneWire ds(DS18B20_pin);  //   初始化 ds 物件
DallasTemperature DS18B20(&ds); // 初始化 DS18B20 物件
```



溫控實驗接線圖

Appsdüino

Lab 14：溫度控制

```
void setup(void)  {


  pinMode(13, OUTPUT);
  pinMode(8, OUTPUT);
  Serial.begin(9600);
  DS18B20.begin(); // 啟動 DS18B20
  PrevTime = millis();
}

void loop()
{
  CurrTime = millis();
  if (CurrTime -PrevTime > IntervalTime)
  {  //  每隔一秒偵測一次溫度
    DS18B20.requestTemperatures(); //由 DS18B20 物件，偵測環境溫度
    int val = DS18B20.getTempCByIndex(0); // 取得溫度值
    Serial << val << endl;  //顯示溫度
```

Appsduino

Lab 14：溫度控制

```
    if (val >= UpperTemp)  //若到達上限溫度 點亮 D13 綠燈
      digitalWrite(13, HIGH);
    else
      digitalWrite(13, LOW);
    if (val >= AlarmTemp)  //若到達警戒溫度 發出警報聲
      digitalWrite(8, HIGH);
    else
      digitalWrite(8, LOW);
    PrevTime = CurrTime;
  }
}
```

程式執行後，在序列埠監控視窗可看到溫度值，若以熱風吹 DS18B20 使得溫度不斷上升，若到達上限溫度就會點亮 pin13 綠燈，若到達警戒溫度就會發出警報聲。

Appsduino

好用的 Timer Library

- Arduino計時器（Timer）程式庫(由Dr. Monk所開發)
- 下載後，把目錄改名成Timer，搬移到Arduino軟體開發環境的libraries子目錄
- 重新開啟Arduino IDE

★ delay / Timer 區別？簡潔/多工/效率

★ Timer Library API：

- int every(long period, callback)：每隔"period" ms執行callback,Return Event ID
- int every(long period, callback, int repeatCount)
- int after(long duration, callback)
- int oscillate(int pin, long period, int startingValue)：每隔"period" ms,toggle pin 的state(HIGH/LOW)
- int oscillate(int pin, long period, int startingValue, int repeatCount)
- int pulse(int pin, long period, int startingValue)
- int stop(int id)：停止指定的Timer event ID
- int update()：必須置於"loop()"以更新管理timer event ID

Appsdüino

Lab 14_1 TimerTemperature

```
// 首先匯入程式庫標頭檔Timer.h
#include <Timer.h>
Timer t;      // declare one timer event

//-------- DS18B20 Temperature sensor ------------
#define   DS18B20_Pin   7  //Define DS18S20 onewire signal pin on D7
#include <OneWire.h>
#include <DS18B20.h>
DS18B20 dd(DS18B20_Pin);  // Declare one DS18B20 object : "dd" on digital pin 7
//-----------------------------------------------

void setup()
{
  Serial.begin(9600);
  t.every(1000,OneSecTimerHandler); // setup callback function
}

void loop()
{
  t.update();// 在loop裡，呼叫每個計時器的update，它才能運作更新狀態
}

//-------- update Temperature per sec and Tx to Serial port
void OneSecTimerHandler()
{
  Serial.println(dd.getTemperature(),2);// Read DS18B20 temperature data and print it out
}
```

Appsduino

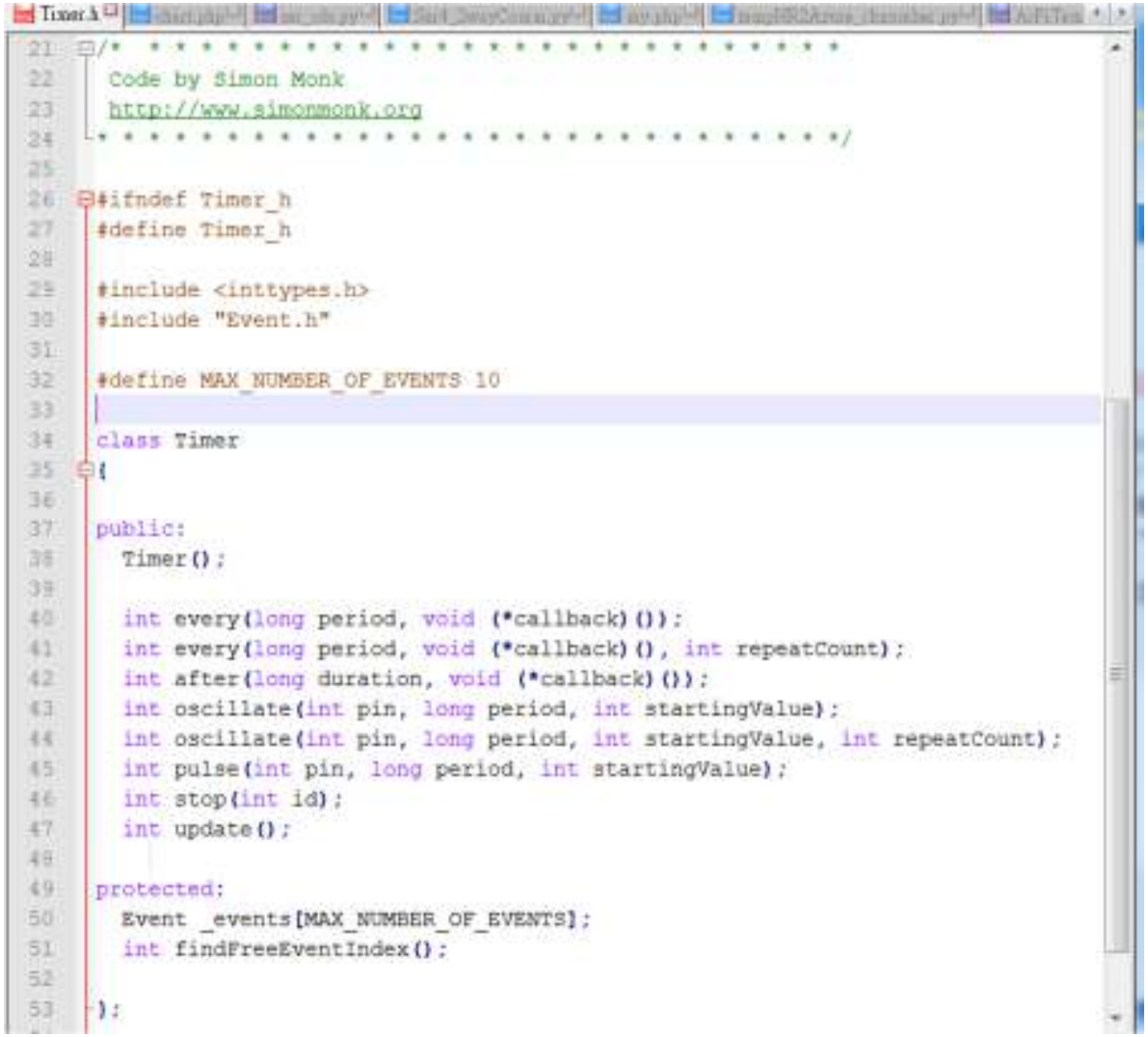
Lab2_1 TimerLed

```
#include <Timer.h>
Timer t;     // declare one timer event
#define   greenLed    13
#define   CDS         A2

void setup()
{
  pinMode(greenLed, OUTPUT);
  digitalWrite(CDS,HIGH) ; // enable pull up resistor
  Serial.begin(9600);
  t.oscillate(greenLed, 1000, LOW); // toggle LED one Sec interval
  t.every(1000, cdsReading); // setup callback function
}
void loop()
{
  t.update();
}
void cdsReading()
{
  Serial.println(analogRead(CDS)); // 讀取Analog A2 Value(ex: CDS)
}
```

Appsduino

Library study @Notepad++



```
/*  * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 Code by Simon Monk
 http://www.simonmonk.org
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#ifndef Timer_h
#define Timer_h

#include <inttypes.h>
#include "Event.h"

#define MAX_NUMBER_OF_EVENTS 10

class Timer
{

public:
  Timer();

  int every(long period, void (*callback)());
  int every(long period, void (*callback)(), int repeatCount);
  int after(long duration, void (*callback)());
  int oscillate(int pin, long period, int startingValue);
  int oscillate(int pin, long period, int startingValue, int repeatCount);
  int pulse(int pin, long period, int startingValue);
  int stop(int id);
  int update();

protected:
  Event _events[MAX_NUMBER_OF_EVENTS];
  int findFreeEventIndex();

};
```

Appsdüino

Library study @Notepad++

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 Code by Simon Monk
 http://www.simonmonk.org
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

// For Arduino 1.0 and earlier
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

#include "Timer.h"


Timer::Timer()
{
}

int Timer::every(long period, void (*callback)(), int repeatCount)
{
	int i = findFreeEventIndex();
	if (i == -1) return -1;

	_events[i].eventType = EVENT_EVERY;
	_events[i].period = period;
	_events[i].repeatCount = repeatCount;
	_events[i].callback = callback;
	_events[i].lastEventTime = millis();
	_events[i].count = 0;
	return i;
}
```

Arduino EEPROM(Read)

```
#include <EEPROM.h>

// start reading from the first byte (address 0) of the EEPROM
int address = 0;
byte value;

void setup()
{
  Serial.begin(9600); // initialize serial and wait for port to open:
}
void loop()
{
  value = EEPROM.read(address); // read a byte from the current address of the EEPROM
  Serial.print(address);
  Serial.print("\t");
  Serial.println(value);
  address = address + 1; // advance to the next address of the EEPROM till end of EEPROM

  if (address == 1024) // only 1024 bytes(0 ~ 1023),so wrap around to address 0 if over 1023
    address = 0;

  delay(500);
}
```

Appsduino

Arduino EEPROM(Write)

```
#include <EEPROM.h>

int addr = 0;

void setup()
{
}

void loop()
{
int val = analogRead(0) / 4; // 0 to 1023 and each byte of the EEPROM can only hold 0 ~ 255

EEPROM.write(addr, val); // these values will remain there when the board is turned off.

addr = addr + 1; // advance to the next address and wrap around to address 0 if over 1023
if (addr == 1024)
    addr = 0;

  delay(100);
}
```

Appsduino

使用 EEPROM Example

```
 // Read Motor seeting from the EEPROM,If addr(0) = "AA" then read the presetting value,otherwise just skip
  if(EEPROM.read(0) == 0xAA) { // Speed setting = Upper_Limit(0.4 + 0.6 * PowerLevel)/100 ; Motor operation is around 40% * Voltage ~ 3.0V
    MotorPowerLevel = EEPROM.read(MotorPowerLevel_eeprom_Addr) ;// restore motor power level
    LeftMotorSpeed = LeftMotorLimit * (35 + 65 * MotorPowerLevel/100)/100 ; // Restore Motor Speed setting
    RightMotorSpeed = RightMotorLimit * (35 + 65 * MotorPowerLevel/100)/100 ; // Restore Motor Speed setting
  }
```

Appsduino

Arduino main() function

Location : arduino-1.6.x\hardware\arduino\avr\cores\arduino\main.cpp
hardware/xxx/arduino/varients/xxx/variant.cpp 中, 定義了 init()

```
#include <Arduino.h>

// Declared weak in Arduino.h to allow user redefinitions.
int atexit(void (* /*func*/ )()) { return 0; }

// Weak empty variant initialization function.
// May be redefined by variant files.
void initVariant() __attribute__((weak));
void initVariant() { }

int main(void)
{
    init();
    initVariant();

#if defined(USBCON)
    USBDevice.attach();
#endif

    setup();

    for (;;) {
        loop();
        if (serialEventRun) serialEventRun();
     }
     return 0;
}
```

Appsduino

The Dino 318 機器人安裝與實作

Appsduino

控制馬達實驗

馬達驅動實驗

MotoUNO 整合一顆 L293D 馬達驅動 IC(如右圖)，透過 jumper 連接 UNO 板的 Pin 7/4/6/5，可以控兩個馬達轉動方向及速度。HIGH



控制板上的輸入接腳 D4 控制 M1 馬達，D7 控制 M2 馬達，當 D4 電壓是 5V/HIGH ，M1 馬達正轉，0V/LOW 則反轉； D7 電壓是 5V/HIGH，M2 馬達正轉，0V/LOW 則反轉。

另外 D5、D6 兩個 Pins 分別控制 M1、M2 馬達的轉速，可利用 PWM 的類比輸出來控制馬達的轉速。

Lab 15: 控制馬達簡單旋轉

確認控制板的 D7/6/5/4 Pins 有接上 Jumper，然後分別把 M1 的兩個接點連線到左馬達的兩條電線(接線若相反馬達可能會反向旋轉，這時若將電線對調，就能正確旋轉)。M2 的兩個接點連線到右馬達的兩條電線(詳如次頁圖片)。

Appsdüino

Lab 15：控制馬達旋轉實驗

程式如下，執行後可觀察到馬達旋轉狀況。

```
#define   leftMotor       4
#define   rightMotor      7
#define   leftMotorSpeed  5
#define   rightMotorSpeed 6

void setup() {
  pinMode(leftMotor, OUTPUT);
  pinMode(rightMotor, OUTPUT);
  digitalWrite(rightMotor, HIGH); //控制右輪前進
  digitalWrite(leftMotor, HIGH ); //控制左輪前進
  analogWrite(leftMotorSpeed,255); //左輪全速前進
  analogWrite(rightMotorSpeed,255);//右輪全速前進
}
void loop()
{

}
```



Appsduino

Lab 16 控制馬達實驗

Lab 16: 控制馬達正、反轉

以下程式可由序列埠監控視窗輸入指令控制馬達旋轉，輸入字元'+'代表前進，字元'-'代表後退，任意鍵代表停車。

```
#define   leftMotor       4
#define   rightMotor      7
#define   leftMotorSpeed  5
#define   rightMotorSpeed 6

void setup()
{
  Serial.begin(9600);
  pinMode(leftMotor, OUTPUT);
  pinMode(rightMotor, OUTPUT);
  Serial.println("Input  '+' : Forward, '-' : Backward, Other key : Stop");
}
```

Appsdüino

Lab 16 控制馬達實驗

```
void loop()
{

  if(Serial.available()) {
    char ch = Serial.read();
    if(ch != -1) {
     if (ch=='+') {
      //  前進
      Serial.println("Forward ");
      digitalWrite(rightMotor, HIGH); //控制右輪前進
      digitalWrite(leftMotor, HIGH ); //控制左輪前進
      analogWrite(leftMotorSpeed,255); //左輪全速前進
      analogWrite(rightMotorSpeed,255);//右輪全速前進
    } else if (ch=='-') {
      //  倒車
      Serial.println("Backward ");
      digitalWrite(rightMotor, LOW); //控制右輪倒轉
      digitalWrite(leftMotor, LOW ); //控制左輪倒轉
      analogWrite(leftMotorSpeed,255); //左輪全速前進
      analogWrite(rightMotorSpeed,255);//右輪全速前進
```

Appsduino

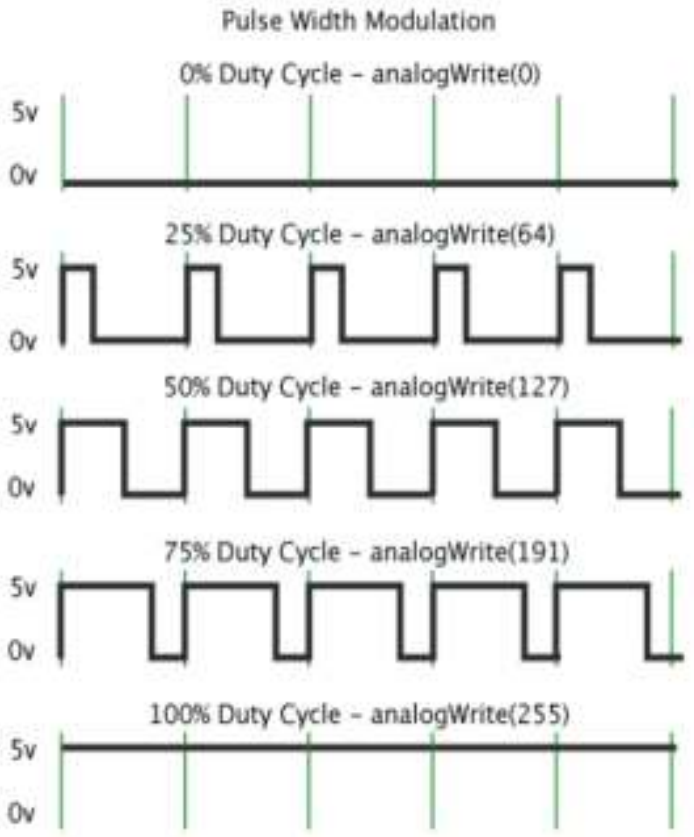
Lab 16 控制馬達實驗

```
    } else if (ch != '\r' && ch != '\n'){
      //  停止
      Serial.println("Stop ");
      analogWrite(leftMotorSpeed,0); //左輪停止
      analogWrite(rightMotorSpeed,0);//右輪停止
    }
  }
  }
}
```

Appsduino

控制馬達轉速實驗

Appsduino UNO 可以產生 8bit 的 PWM 訊號，頻率約在 490Hz，如右圖。使用指令 analogWrite(pinNo, duty_cycle) 就能自己控制 PWM 的特性，pinNo 可為數位輸出入接腳 3、5、6、9、10、11，duty_cycle 值是介於 0~255 的數值用來控制 duty cycle 的寬度，duty cycle 值越大則平均電壓越高。



由於 Pin 5、Pin 6 可以控制馬達是否有動力，藉由快速開關 D5、D6 就能控制馬達轉速，就能從程式來控制轉速了。

以下範例程式從序列埠監控視窗輸入下字元 '+' 代表加速、字元 '-'：代表減速、字元 'F' 控制馬達正轉、字元 'B' 控制馬達反轉、其他字元代表馬達停止，詳細程式如下：

Appsduino

Lab 17：控制馬達轉速實驗

```
#define     leftMotor          4
#define     rightMotor         7
#define     leftMotorSpeed     5
#define     rightMotorSpeed 6

int speed =0;

void setup()
{
   Serial.begin(9600);
   analogWrite(leftMotorSpeed,0); //左輪停止
   analogWrite(rightMotorSpeed,0);//右輪停止
   pinMode(leftMotor, OUTPUT);
   pinMode(rightMotor, OUTPUT);
   Serial.println("'+':speed up     '-':speed down     'F':forward,     'B'backward,     otherwise
break");
}
```

Appsduino

Lab 17：控制馬達轉速實驗

```
void loop()
{
int _speed;

    if(Serial.available()) {
      char ch = Serial.read();
      if(ch != -1) {
        if (ch=='+') {
          //speed = speed+1>10 ? 10 : speed+1;
        speed++;
        if (speed>10) speed=10;
        _speed = map(speed, 1, 10, 100, 255);

        Serial.print("speed , _speed => "); Serial.print(speed); Serial.print(" , ");
Serial.println(_speed);
        analogWrite(leftMotorSpeed, _speed);   //左輪
        analogWrite(rightMotorSpeed, _speed);   //右輪
      } else if (ch=='-') {
        speed--;
        if (speed<=0) {
```

Appsduino

Lab 17：控制馬達轉速實驗

```
      speed=0;
      _speed=0;

    } else {
      _speed = map(speed, 1, 10, 100, 255);
    }
    Serial.print("speed , _speed => ");
    Serial.print(speed);
    Serial.print(" , ");
    Serial.println(_speed);
  } else if (ch=='F') {
    //  前進
    Serial.println("forward");
    digitalWrite(rightMotor, HIGH); //控制右輪前進
    digitalWrite(leftMotor, HIGH ); //控制左輪前進
    analogWrite(leftMotorSpeed, _speed);   //左輪
    analogWrite(rightMotorSpeed, _speed);   //右輪
  } else if (ch=='B') {

    //  倒車
```

Appsdüino

Lab 17：控制馬達轉速實驗

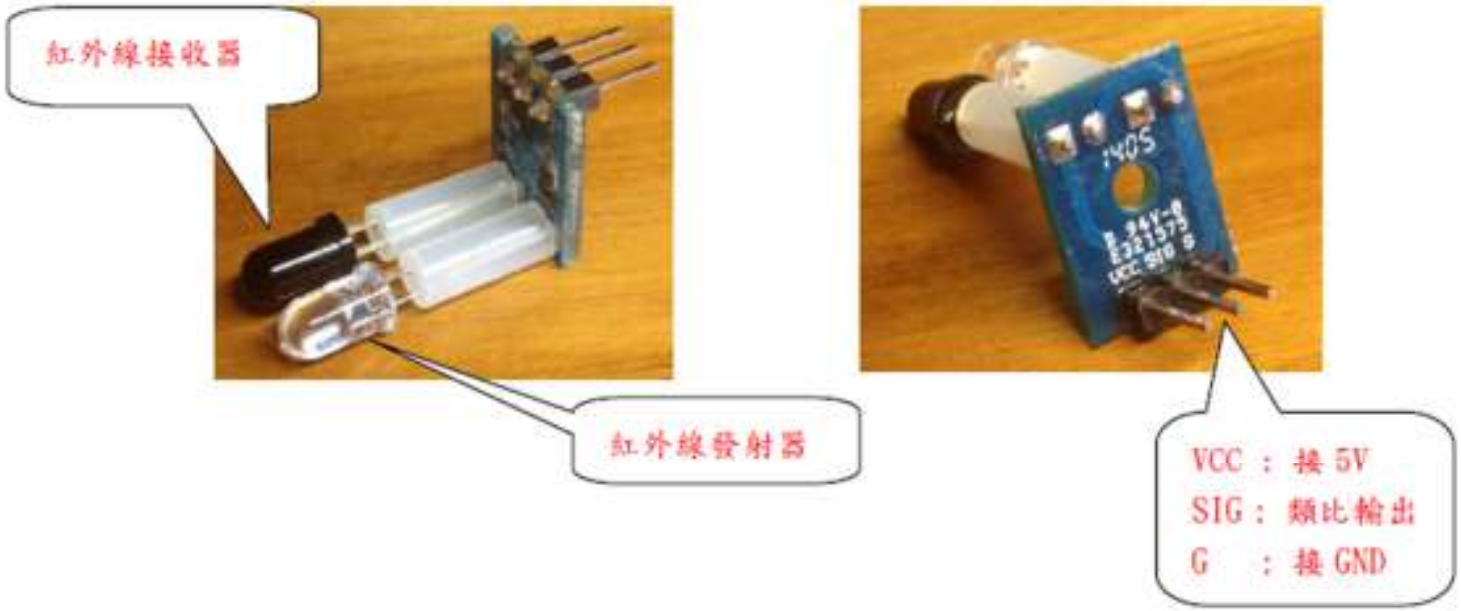
```
            Serial.println("Backward ");
            digitalWrite(rightMotor, LOW); //控制右輪倒轉
            digitalWrite(leftMotor, LOW ); //控制左輪倒轉
            analogWrite(leftMotorSpeed, _speed);    //左輪
            analogWrite(rightMotorSpeed, _speed);    //右輪
          } else if (ch != '\r' && ch != '\n'){
            //    停止
            Serial.println("stop");
            speed = 0;
            analogWrite(rightMotorSpeed, speed);    //右輪停止
            analogWrite(leftMotorSpeed, speed);    //左輪停止
        }
      }
    }
}
```

Appsduino

紅外線尋跡、避障與超音波實驗

Appsduino

紅外線模組

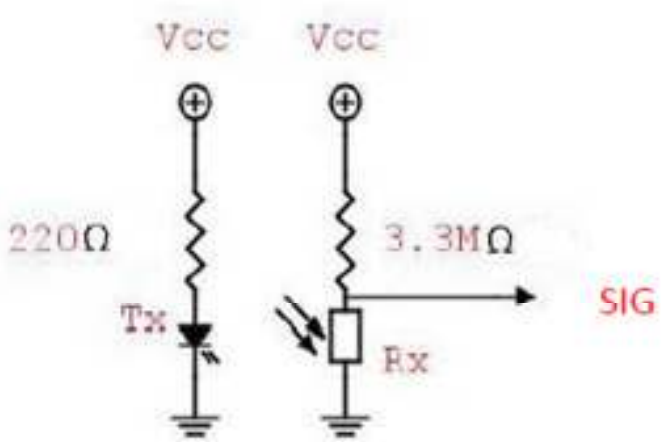


原理說明 ：

紅外線照射到物體，反射回來讓接收器收到，若反射的紅外線越強，將使得接收器兩腳越能導通，於是 SIG 接腳電壓降低，A/D 轉換得到的數位值(0~1023)也就下降，實際測試，若是反射物是張白紙，數位值約為 <50；若反射物是黑紙，數位值約 >300，使用前最好套上黑色熱縮套管，則辨識效果更好，其數位值可能為 0 < SIG < 700，數值大小也與距離、亮度及元件誤差有關。

Appsduino

紅外線模組電路示意圖

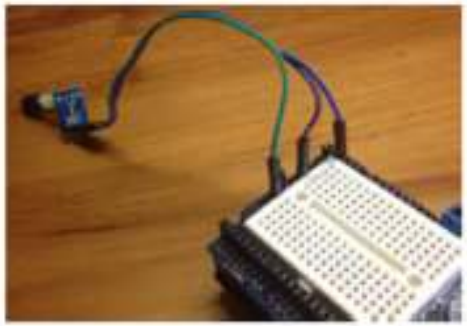


紅外線尋跡避障電路示意圖



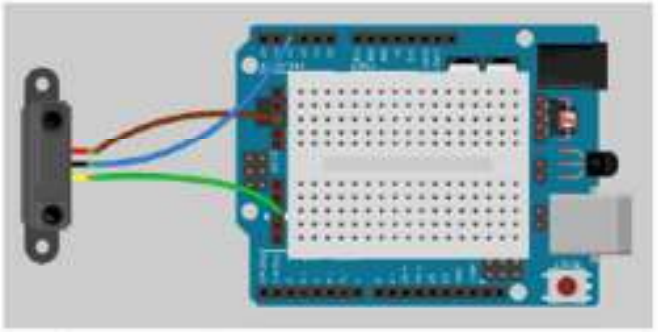
Appsduino

Lab 18：紅外線感測實驗



按照上圖的腳位，將 VCC 接到+5V，G 接到 GND，
SIG 接到 Arduino 的類比腳位 A3，執行下列程式，
並拿張白紙與黑紙做測試，即可顯示數值的變化。

以下程式將紅外線接收器取值並顯示出來。



程式如下

```
#include <Streaming.h>
void setup() {
  Serial.begin(9600);
}

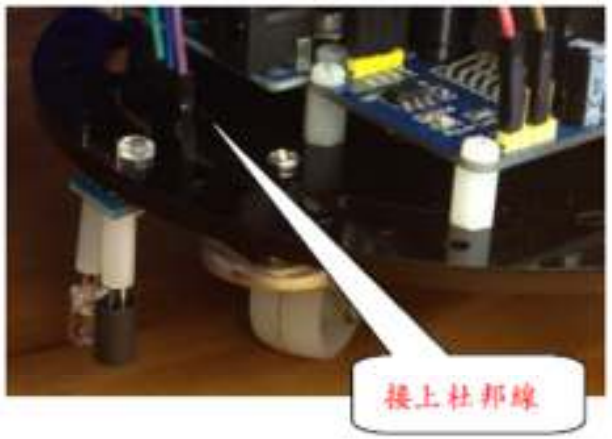
void loop() {
  int raw_data = analogRead(3);//偵測 A3 傳回的電壓值
  //若紅外線反射越強，接受器隨之導通(電阻也會下降)
  //A3 的電壓也會下降，A/D 轉換得到的數位值也會偏低

  Serial << "raw data : " << raw_data << endl;  //顯示結果
  delay(1000);
}
```

Appsduino

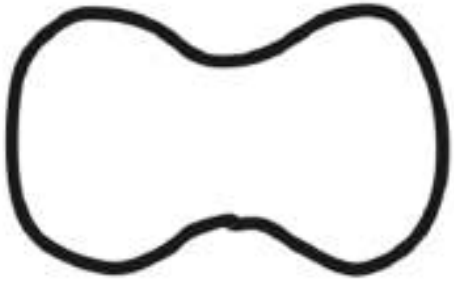
Lab 19：循黑線的自走車

這個實驗是製作一台可以自己沿著黑線走的車子，延續前一個紅外線實驗的電路，將紅外線尋跡模組用尼龍螺絲固定於前緣孔內，接上杜邦線，如圖下所示 ：



另外準備一張白紙上面有黑色的導航線條(可以使用黑色膠帶貼出適合的軌跡)，例如下圖







Appsdüino

Lab 19：循黑線的自走車

```
#include <Streaming.h>
#include "L298Lib.h"
//  沿著黑線走
void setup()
{
  L298_setup();  //初始化 L298 馬達控制板
  DriveSingleMotor(MOTOR_M1, PWR_STOP, DIR_FORWARD);  //右馬達停止轉動
  DriveSingleMotor(MOTOR_M2, PWR_STOP, DIR_FORWARD);  //左馬達停止轉動
}

void loop()
{
  int raw_data = analogRead(3);  // 取得 A3 的輸入數位值(介於 0~1023)
  // 若 raw_data 數位值偏低，代表紅外線照射到白色部分，實際試驗值可能是 250 左右
  // 若 raw_data 數位值偏高，代表紅外線照射到黑色部分，實際試驗值可能是 750 左右
  // 因此大概取中間值做為分界
  if(raw_data<500) {    // 若紅外線照射到白色
    DriveSingleMotor(MOTOR_M1, 5, DIR_FORWARD);  //右馬達出力
    DriveSingleMotor(MOTOR_M2, PWR_STOP, DIR_FORWARD);  //左馬達停止
    digitalWrite(13, HIGH);  // 指示燈點亮
  } else { // 若紅外線照射到黑色
```

Appsdüino

Lab 19：循黑線的自走車

```
    DriveSingleMotor(MOTOR_M1, PWR_STOP, DIR_FORWARD);  //右馬達停止
    DriveSingleMotor(MOTOR_M2, 5, DIR_FORWARD);  //左馬達出力
    digitalWrite(13, LOW);  // 指示燈熄滅
  }
}
```

以上程式使用到 L298Lib 函式庫，請將 L298Lib.cpp 以及 L298Lib.h 兩個檔案複製到 Arduino 目錄的 libraries 目錄之下。函式庫提供多組函數可控制馬達，其中函數 L298_setup( ) 是用來初始化控制板，函數 DriveSingleMotor( )則是控制單馬達 ：

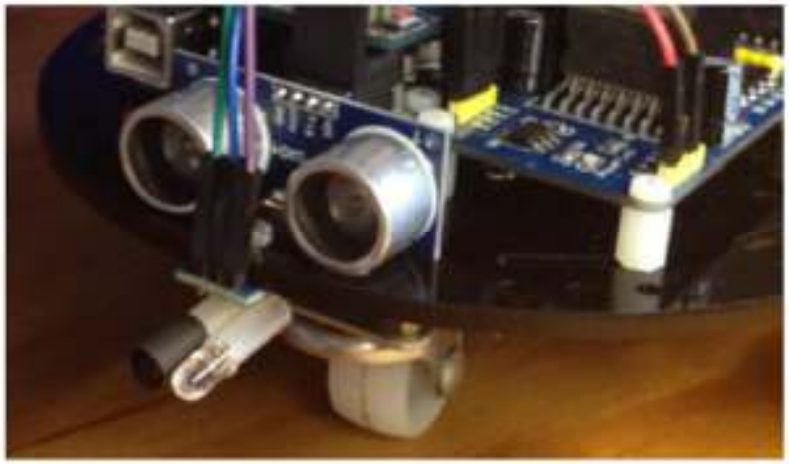
```
void  DriveSingleMotor (int m, int speed, int dir)
```

第一個參數 m 若為 0 代表要控制右馬達，1 代表要控制左馬達。
第二個參數 speed 可為 0(最慢)到 10(最快)。第三個參數 dir 若是 0 代表正轉，1 是反轉。
例如 DriveSingleMotor(0, 5, 0); 命令右馬達以中等速度向前正轉，由於在 L298Lib.h 內已經定義了一些常數，因此也可寫成 DriveSingleMotor(MOTOR_M1, 5, DIR_FORWARD); 意義是一樣的，詳細常數請參考 L298lib.h，詳細的程式設計邏輯請參考 L298lib.cpp。
以上程式執行後就會看見自走車沿著黑線不斷的扭動前進。

Appsdüino

Lab 21:自動閃避障礙物的車子

設計一台自走車，當它在地面上行走，可以自動感應車子前方障礙物，並且自動閃躲避免撞上牆壁。電路設計與上一個實驗完全相同，只是紅外線發射、接受器的感應面改成向前(如下圖)：





可以利用現有的紙箱、隔板布置一個四面均有牆壁的練習場地，讓小車在其中行走，若感應到前方有牆壁障礙物，就能自動閃躲。

Appsduino

Lab 21:自動閃避障礙物的車子

```
#include <Streaming.h>
#include "L298Lib.h"
// 自動閃避障礙物的車子
void setup()
{
  L298_setup();  //初始化 L298 馬達控制板
  DriveSingleMotor(MOTOR_M1, PWR_STOP, DIR_FORWARD);  //右馬達停止轉動
  DriveSingleMotor(MOTOR_M2, PWR_STOP, DIR_FORWARD);  //左馬達停止轉動
}
void loop()
{
  int raw_data = analogRead(3);  // 取得 A3 的輸入數位值(介於 0~1023)
  // 若 raw_data 數位值偏低，代表沒有太多紅外線反射=>前方無障礙物
  // 若 raw_data 數位值偏高，代表有很多紅外線反射=>障礙物接近
  // 因此大概取中間值做為分界
  if(raw_data>500) {  // 遇到前有障礙物了
    DriveSingleMotor(MOTOR_M1, PWR_STOP, DIR_FORWARD);  //左右馬達都停止
    DriveSingleMotor(MOTOR_M2, PWR_STOP, DIR_FORWARD);
    digitalWrite(13, HIGH);
```

Appsduino

Lab 21:自動閃避障礙物的車子

```
    DriveSingleMotor(MOTOR_M1, 3, DIR_BACKWARD);  //先讓左右馬達倒轉=>倒車 0.5 秒
    DriveSingleMotor(MOTOR_M2, 3, DIR_BACKWARD);
    delay(500);
    DriveSingleMotor(MOTOR_M1, 5, DIR_BACKWARD);  // 車子原地右偏轉 0.7 秒鐘
    DriveSingleMotor(MOTOR_M2, PWR_STOP, DIR_FORWARD);
    delay(700);
    DriveSingleMotor(MOTOR_M1, PWR_STOP, DIR_FORWARD);  //左右馬達都停止
    DriveSingleMotor(MOTOR_M2, PWR_STOP, DIR_FORWARD);
  } else {
    DriveSingleMotor(MOTOR_M1, 4, DIR_FORWARD);  // 左右馬達恢復正轉，車子前進
    DriveSingleMotor(MOTOR_M2, 4, DIR_FORWARD);
    digitalWrite(13, LOW);
  }
}
```

此程式原理與上一個實驗大致相同，同學可以試著改寫程式，當遇到障礙物時，也發出 buzzer 的警報聲。

Appsdüino

藍牙與MeetAndroid Library介紹

Appsduino

Arduino 藍牙通訊模組

此藍牙模組使用 UART(Tx/Rx)介面來收發訊息，
並能與藍牙手機或平板無線通訊，其特性如下 ：

- 藍牙協定: Bluetooth V2.1 + EDR
- 工作頻率: 2.4~2.48GHz, ISM Band
- 介面: UART
- 輸入電壓: 3.3V/5V
- 工作溫度: -20℃ ~ +75℃

藍牙模組操作 baud rate : 1200 ~ 115,200 bps
目前預設值 ：

Baud rate : 57,600bps
ID : Appsduino
PIN Code(配對密碼) : 0000

其腳位定義如右 ：



Pin	ID	說明
1	PWR SEL	X(沒接)
2	VCC	5V
3	GND	
4	TXD	UART TXD(接至 UNO RX Pin/D0)
5	RXD	UART RXD(接至 UNO TX Pin/D1)
6	MCLR	重置原先之設定值，正常沒接

Appsduino

Arduino 藍牙通訊模組

此藍牙模組使用 UART(Tx/Rx)介面來收發訊息，並能與藍牙手機或平板透過無線通訊

預設值 ：

Baud rate ： 57,600bps

ID ： Appsduino

PIN Code(配對密碼) ： 0000

其腳位定義如下 ：

Pin	ID	說明
1	PWR SEL	X(沒接)
2	VCC	5V
3	GND	
4	TXD	UART TXD(接至 UNO RX Pin/D0)
5	RXD	UART RXD(接至 UNO TX Pin/D1)
6	MCLR	X(重置原先之設定值,正常沒接)





Appsduino

BT Baud rate速率對應表

<p>	<r>	備註
1	1200	設定爲 1200 bps
2	2400	設定爲 2400 bps
3	4800	設定爲 4800 bps
4	9600	設定爲 9600 bps
5	19200	設定爲 19200 bps
6	38400	設定爲 38400 bps
7	57600	設定爲 57600 bps
8	115200	設定爲 115200 bps
9	230400	設定爲 230400 bps
A	460800	設定爲 460800 bps
B	921600	設定爲 921600 bps
C	1382400	設定爲 1382400 bps

Appsduino

修改藍牙模組名稱與Baud rate

修改藍牙模組名稱

傳送：AT+NAMEname
回傳：OKname
name：所要設置的名稱，需 20 個字元以內。
例：傳送AT+NAMEDino911
回傳 ： OKname
這時藍牙名稱改為Dino911

藍牙模組 Baud rate 修改及速率對應表

指令：AT+BAUD<p>
回傳：OK<r>
例 ： 要設定 Bluetooth 成 9600 bps，則需送 "AT+BAUD4" command to Bluetooth module.
若成功則回送"OK9600" 訊息 !

Note 1 ： 設定時需注意雙方的 Baud Rate 需一致
Note 2 ： 此 BT 模組雖支援至 1,382,400bps(黃色部分)， 但 Arduino UNO 控制板之 UART 僅支援至 115,200bps
Note 3 ： 預設之資料格式為 8 data bits, no parity, one stop bit

Appsduino

修改藍牙模組名稱程式

```
//使用" AT+NAME" 命令更改藍牙的ID Name，綠燈亮表示更改完成
int LedGreen = 13 ;

void setup()
{
   pinMode(LedGreen,OUTPUT) ;
   digitalWrite(LedGreen,LOW) ; // turn off LED
   Serial.begin(57600);    // Baud rate of HC-0x bluetooth module(設定已知之速率)
   delay(1000);
   Serial.write("AT+NAMEAppsduino"); // Change the "Appsduino" name to your ID name,
                                     //將"Appsduino"改成你欲設定之名稱
   while(!Serial.available()) ; // waiting for 'ok' message
   Serial.read(); // flush feedback message
   digitalWrite(LedGreen,HIGH) ; // turn on LED
}

void loop()
{
}
```

紅色 ID 改成你欲設定之名稱

Appsduino

修改藍牙 Baud rate 程式

```
//此範例將藍牙的Baud Rate 從9600bps更改為57600 bps，綠燈亮表示更改完成
int LedGreen = 13 ;

void setup()
{
   pinMode(LedGreen,OUTPUT) ;
   digitalWrite(LedGreen,LOW) ; // turn off LED
   Serial.begin(9600);          // Baud rate of HC-0x bluetooth module(設定已知之速率)
   delay(1000);
   Serial.write("AT+BAUD7"); // Change baud rate to 57600 bps
   while(!Serial.available()) ; // waiting for 'ok' message
   Serial.read(); // flush feedback message
   digitalWrite(LedGreen,HIGH) ; // turn on LED
}

void loop()
{
}
```

Appsduino

MeetAndroid BT 程式庫命令格式

- MeetAndroid Libary
- Android to Arduino Data Format :

終止符號：19

Event ID	data1;data2; ...	0x13

Event ID : A ~ Z or a ~ z

起始符號

ex : A1;-0.9876;1.7654 0x13
c1 0x13

Appsduino

MeetAndroid Library 說明

MeetAndroid Library常用函數使用說明：

- AppsduinoBT：使用前需宣告一個 MeetAndroid 的物件，例如：
  MeetAndroid AppsduinoBT; //宣告一個 MeetAndroid 的物件名稱叫 AppsduinoBT
- void registerFunction(void(*)(uint8_t, uint8_t),uint8_t)：宣告接收到起始符號時的處理程序(processHandler)，例如：
  AppsduinoBT.registerFunction(BTCommHandler, 'c'); //當接收到起始符號為'c'時跳至 BTCommHandler() 函數處理(Call Back Function)

- bool receive(void)：檢查是否 received buffer 有接收到資料，若有資料(回傳 true)，並會自動跳至預設的處理程序(processHandler)執行，此函數需置於 loop()內
- void getString(char[])：若收到的資料是字串形式，則可以此函數來取出
- int getInt()：若收到的資料是整數，則可以此函數來取出
- long getLong()：若收到的資料是長整數，則可以此函數來取出
- float getFloat()：若收到的資料是浮點數，則可以此函數來取出
- void send(char)：可用 send()函數來傳送字元、字串、整數、長整數,.. 等不同資料
- void send(const char[])：
- void send(uint8_t)：
- void send(int)：
- void send(unsigned int)：
- void send(long)：

Appsduino

藍牙程式範例

```
#include <MeetAndroid.h> // include Android Bluetooth Library header

MeetAndroid AppsduinoBT; //-- declare one MeetAndroid Bluetooth object  --

void setup() {
Serial.begin(57600); // setup baud rate
AppsduinoBT.registerFunction(BTCommHandler, 'c'); //register one call back function
}

void loop() {
AppsduinoBT.receive() ; // you need to keep this in your loop() to receive events
}

void BTCommHandler (byte flag, byte numOfValues)  {//'c' flag to receive BT cmd
  int cmd = AppsduinoBT.getInt();
  If (cmd == ??) do_action ;
  ......
}
```

需自行定義可能的命令與格式，
並修改此處與命令處理程序

Appsduino

藍牙程式範例

```
#include <MeetAndroid.h> // include Android Bluetooth Library header

MeetAndroid AppsduinoBT; //-- declare one MeetAndroid Bluetooth object  --

void setup() {
Serial.begin(57600); // setup baud rate
AppsduinoBT.registerFunction(BTCommHandler, 'c'); //register one call back function
}

void loop() {
AppsduinoBT.receive() ; // you need to keep this in your loop() to receive events
}

void BTCommHandler (byte flag, byte numOfValues)  {//'c' flag to receive BT cmd
  int cmd = AppsduinoBT.getInt();
  If (cmd == ??) do_action ;
  ......
}
```

需自行定義可能的命令與格式，
並修改此處與命令處理程序

Appsduino

藍牙接收處理程式

```
//--- 'c' flag to receive the command to drive the motor's direction next step ----

void Dino911BTCommandHandler(byte flag, byte numOfValues)
{

 int command = AppsduinoBot.getInt();

     DCMotor(LeftMotorSpeed,RightMotorSpeed,command);
}
```

Appsduino

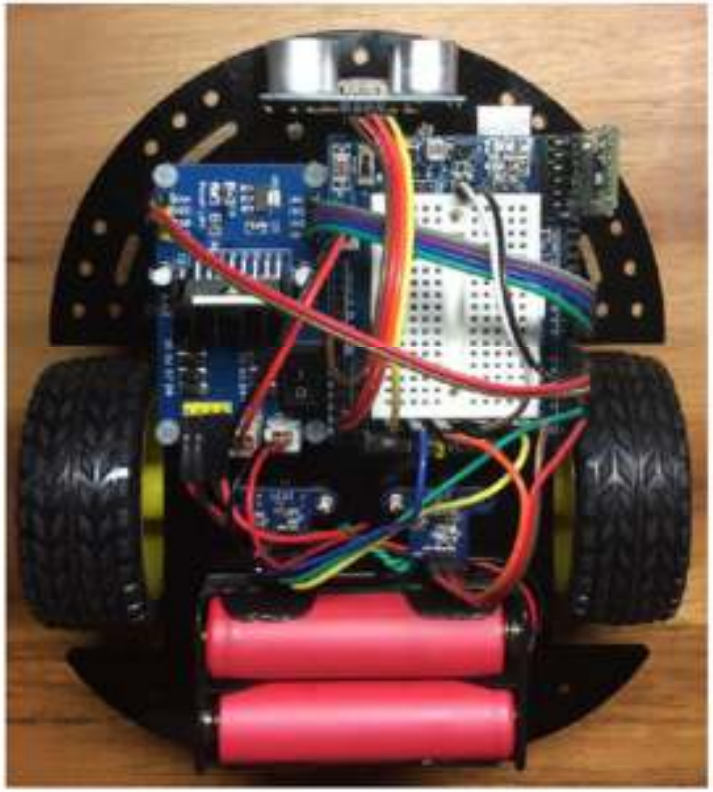
Android and 藍牙互動控制實驗

Appsduino

Dino 928 接線安裝

◆ I/O 對應接腳如下，接線示意圖如下頁 ：(D ： Digital I/O，A ： Analog I/O)

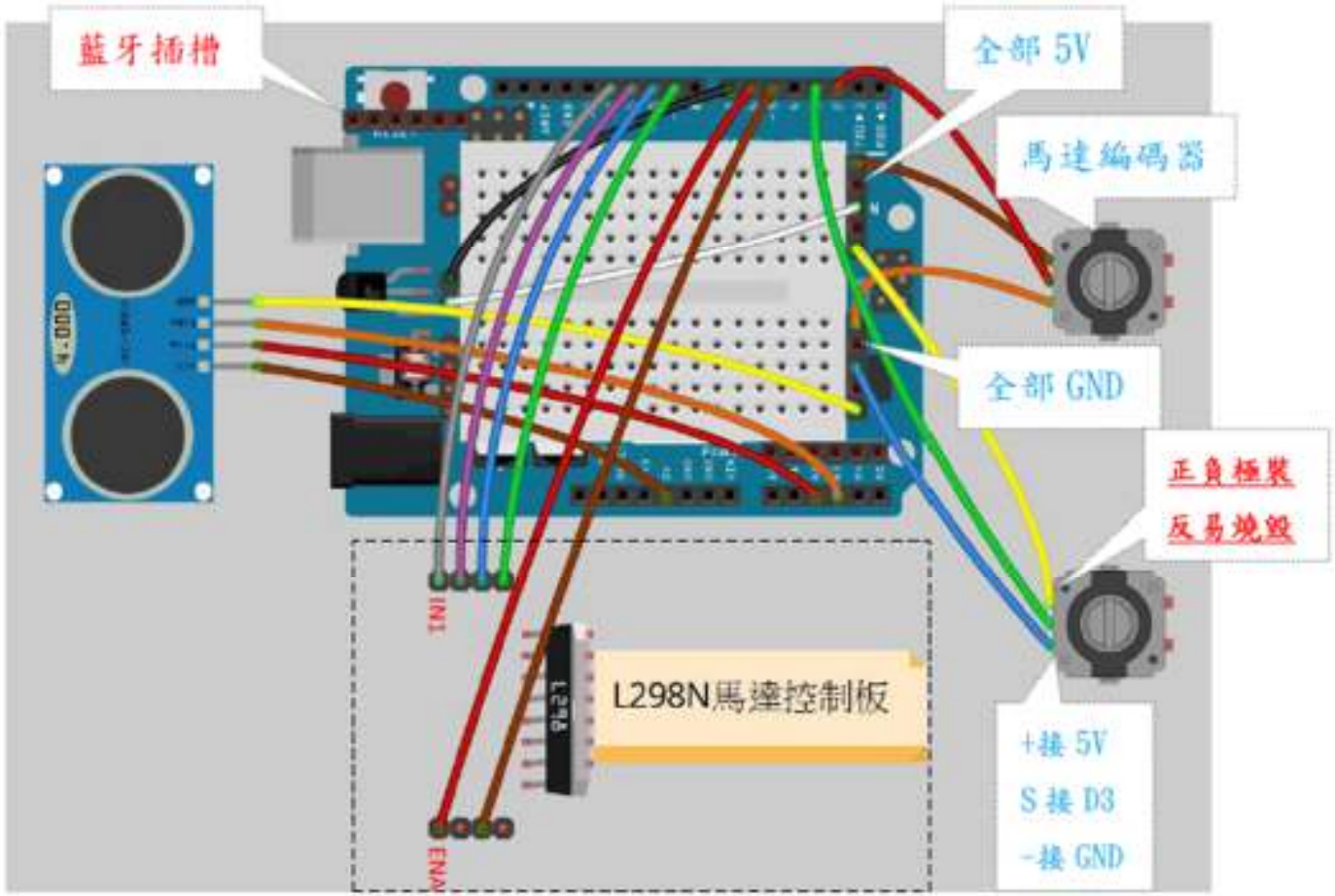
腳位名稱/ID	UNO 腳位	說明
S(右編碼器)	D2	INT0
S(左編碼器)	D3	INT1
ENB(L298N)	D5	PWM/速度(左馬達)
ENA(L298N)	D6	PWM/速度(右馬達)
DQ(DS18B20)	D7	數位溫度計
IN1	D12	右馬達
IN2	D11	右馬達
IN3	D10	左馬達
IN4	D9	左馬達
Buzzer	D8	連接 UNO D8 之 Jumper
TRIG(超音波)	A2	類比腳位 A2
ECHO(超音波)	A3	類比腳位 A3
電池電壓偵測	A0	連接 UNO A0 之 Jumper



Appsduino

The Dino 928 接線示意圖

◆ I/O 對應接腳接線圖(此乃模擬 UNO 接上 Appsduino 擴充板之接線圖)



Appsduino

上傳 Dino 928 程式範例

◆ 安裝藍牙模組

在 J13 的 6 孔排母座插上藍牙模組，元件面朝麵包板(如右圖)



◆ 上傳所附之 TheDino928_V10 範例程式至 UNO 板

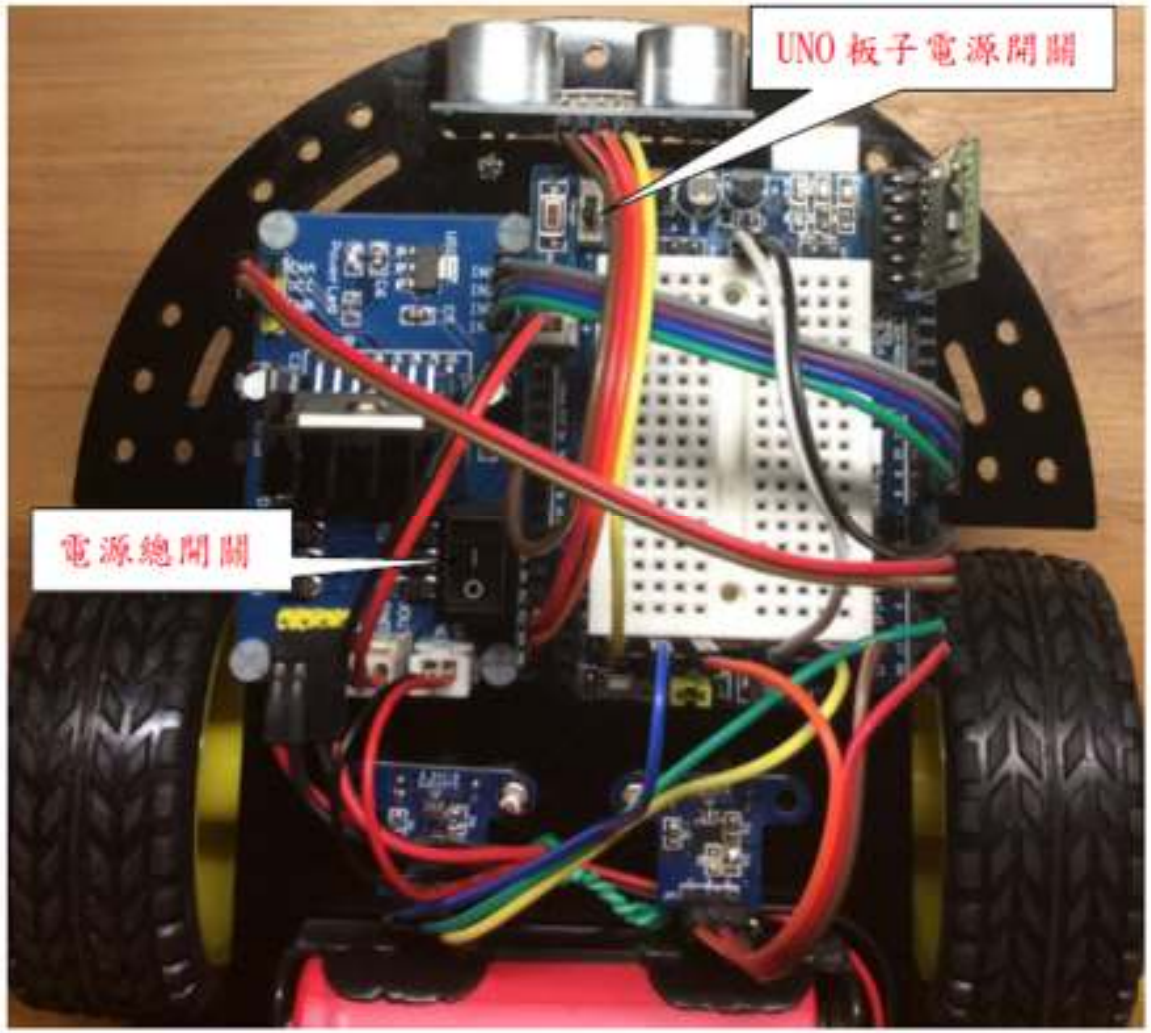


◆ 打開電源開關

當接線安裝完畢，請檢察編碼器、超音波與數位溫度計的電源(VCC/GND)正負極性是否正確？然後滑動擴充板上的開關至 ON 與按下馬達控制板上的總開關，則 UNO 與馬達控制板上之 LED 應會亮起，即可進行學習之旅，若 LED 燈未亮，請檢察電池是否接觸良好？(可從電池盒負極(彈簧)往正極推之或旋轉一下電池，使之與正極有良好接觸)

Appsdüino

確認UNO與總開關的電源開關是否打開



Appsduino

Google Play 下載 Dino 928 App

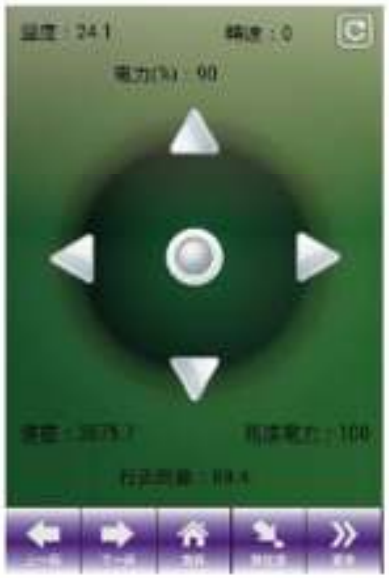
The Dino 928 結合智慧型手機/平板，透過其內建的藍牙功能(Bluetooth)，使其成為控制與訊息顯示中心。請從 Google Play 下載"The Dino 928 互動軟體"後安裝，即可使用智慧型手機/平板與 Dino 928 機器人互動。除利用傳統開發 App 的方式(Java, C, ...)，更可利用 App Inventor 或長茂科技所提供之雲端發展平台 CmoreIO，透過拼圖式與簡單的設定，即可快速開發簡單的 App 應用，無須撰寫程式，讓您專注於創意的實現！



The Dino 928 App



Android App



CmoreIO Android App

Appsdüino

Dino 928 App 簡介

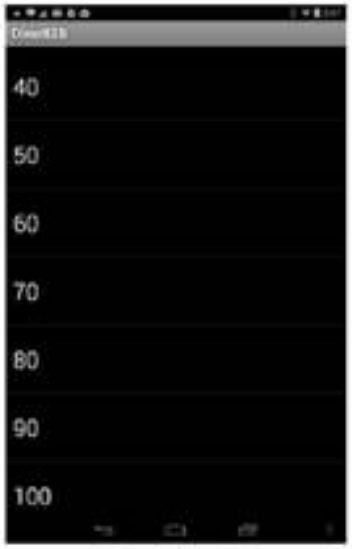
◆ 直線性修正(PID 控制)

Dino928 動力使用兩個獨立的直流馬達，各自的扭力或轉速以及輪子的摩擦力都有些許不同，因此前進時要保持一直線會有些難度，因此在兩個馬達上裝有兩組編碼器，行進時會把轉動訊息傳回，可據此資訊透過 PID 機制修正兩邊馬達的轉速，可改善 Dino928 前進時的直線性，此功能可透過 App 的"直線性修正"按鍵來啟動，若啟動後仍無改善，則需調整 PID 程式中的參數，以達到較佳效果。



◆ 馬達電力/轉速調整

除了可透過 App 控制馬達的方向外，也可使用 App 的"馬達設定"按鍵調整馬達的電力/轉速，範圍為 10 ~ 100%，如右圖示



Appsduino

Dino 928 App 聲控簡介

◆ Android APP 聲控命令(請先將智慧型裝置的藍牙配對(PIN code:0000 並連上網路(WiFi/3G)):

- "向前進" : 車子前進
- "後退" : 後退
- "停止" : 停止
- "向右旋轉" : 向右旋轉
- "向左旋轉" : 向左旋轉
- "左轉" : 左轉
- "右轉" : 右轉

✓ 也可聲控設定移動距離(單位:公分)

- "前進 50 公分"
- "後退 100 公分"
- "右轉 50 公分"
- "左轉 90 公分"



Appsdüino

Dino 928 App Gsensor 簡介

- Android G Sensor 體感控制(請先將智慧型裝置的藍牙配對(PIN code:0000)並連接)

  一般 Android 智慧型手機/平板都內建加速度計(G Sensor)，可偵測 X、Y、Z 軸的加速度變化，因此可用來操控機器人的行進動作



Appsdüino

Arduino+ZigBee物聯網之旅

Appsduino

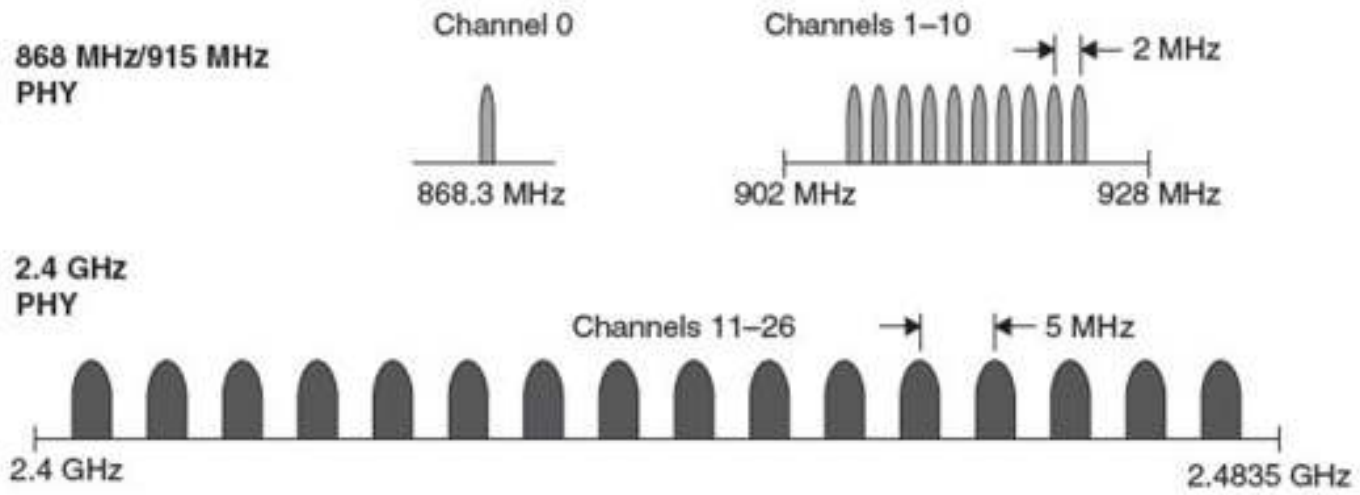
ZigBee(802.15.4)的應用 --> IoT

ZigBee技術是藉由在各種物品上置入一個低耗電以及低成本的ZigBee無線模組來達到存取資訊以及進行控制的技術。



Appsdüino

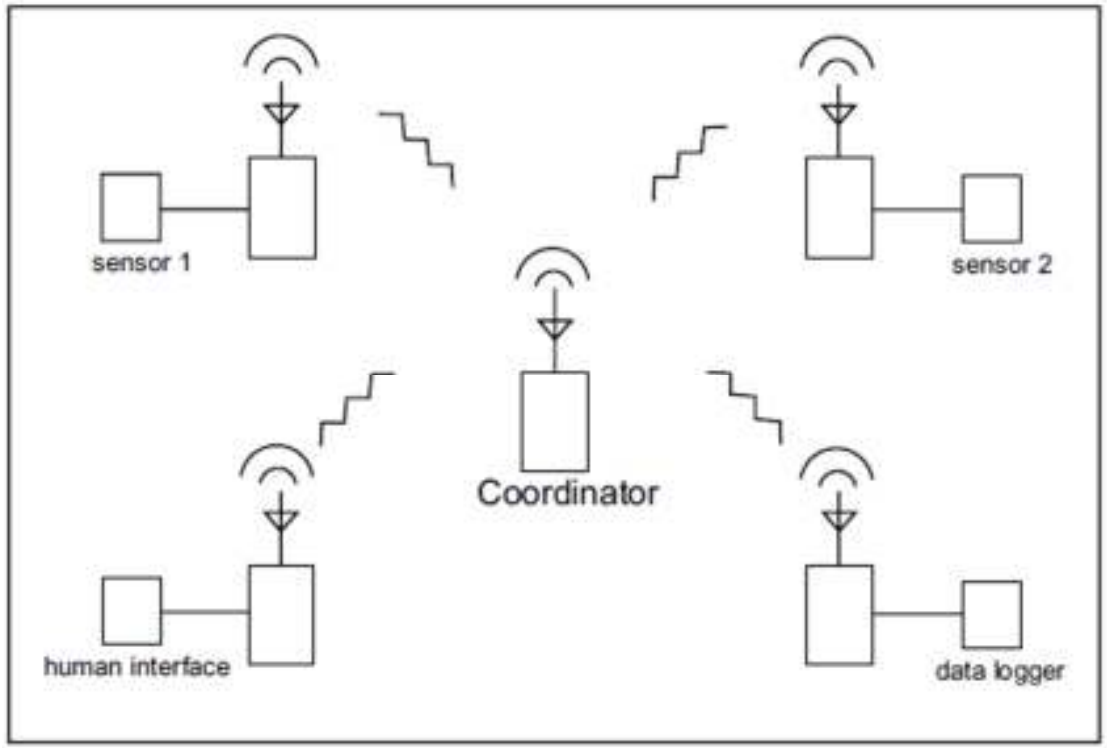
ZigBee Physical Overview



PHY	Frequency Band	Channel Numbering	Spreading Parameters		Data Parameters		
			Chip Rate	Modulation	Bit Rate	Symbol Rate	Modulation
868/915 MHz	868-870 MHz	0	300 kchip/s	BPSK	20 kb/s	20 kbaud	BPSK
	902-928 MHz	1 to 10	600 kchip/s	BPSK	40 kb/s	40 kbaud	BPSK
2.4 GHz	2.4-2.4835 GHz	11 to 26	2.0 Mchip/s	O-QPSK	250 kb/s	62.5 kbaud	16-ary Orthogonal

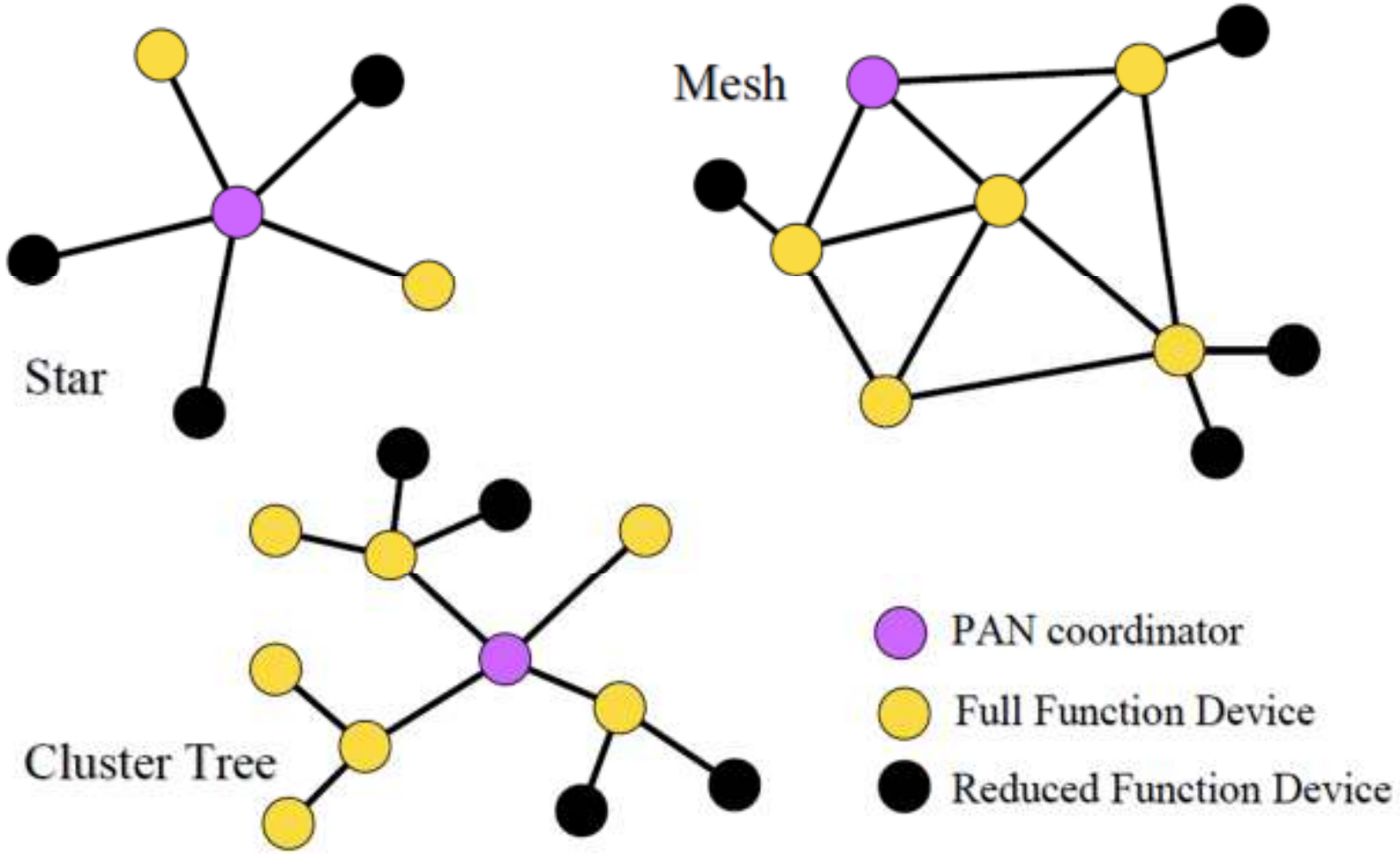
Appsdüino

ZigBee Network



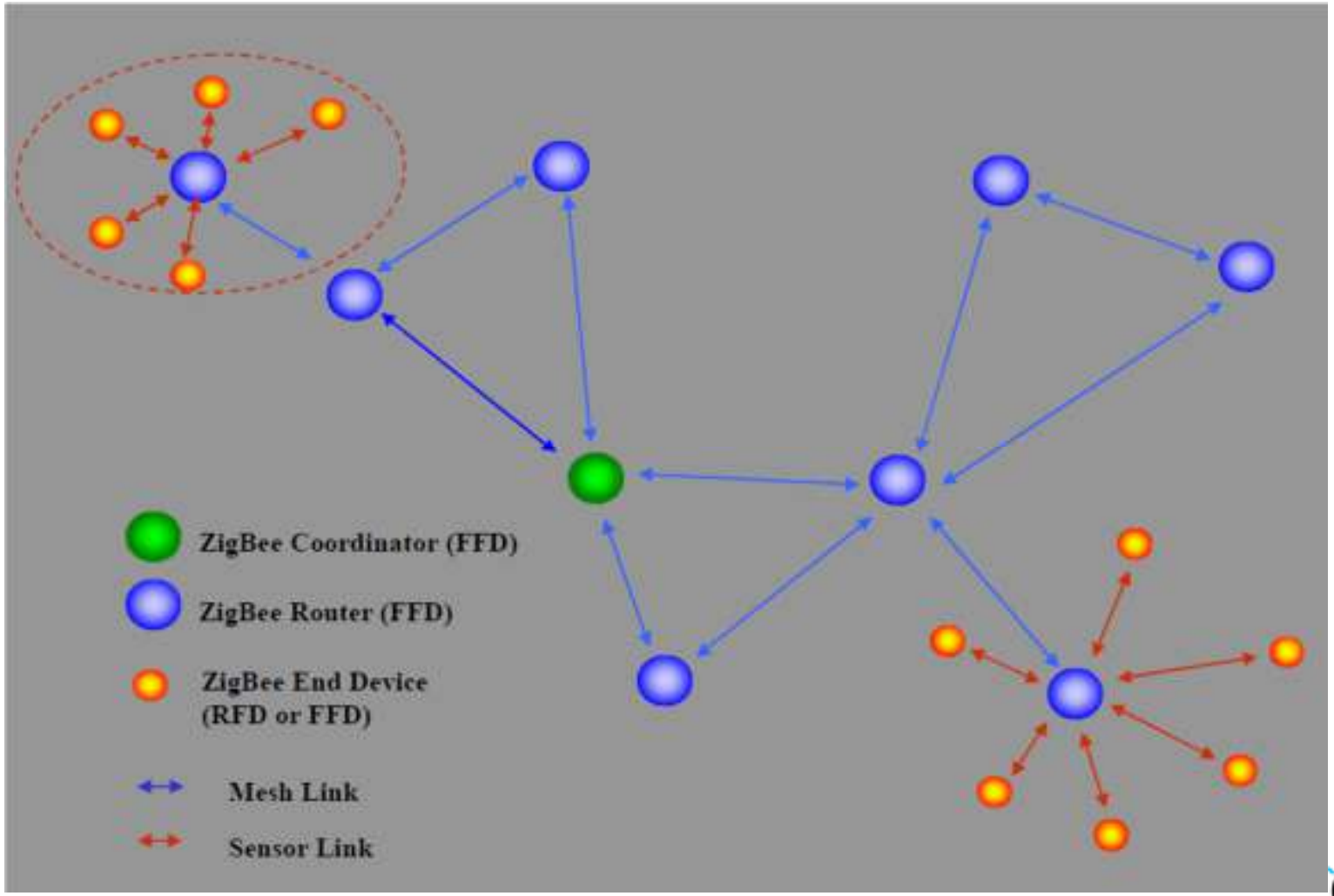
Appsdüino

ZigBee Network Topologies



Appsdüino

ZigBee Network Signal Model



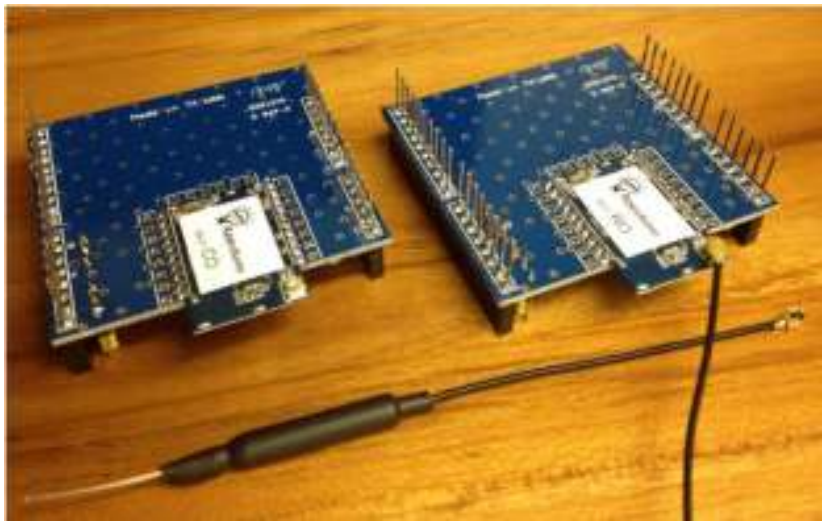
Appsdüino

ZigBee Arduino 擴充板

- Plug and Play(無需轉接板)
- TI CC2530/CC2591(100mW)
- UART I/F



ZigBee(2mW) 擴充板(CO/RO)

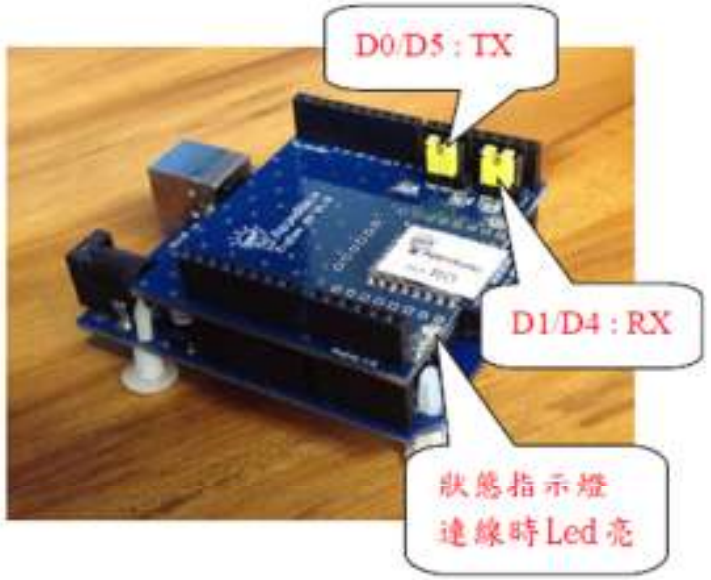


長距離ZigBee(100mW) 擴充板(CO/RO)

Appsduino

Arduino + ZigBee P&P

由於此 ZigBee 擴充板使用串列埠(TX/RX)與 Arduino 單片機通訊，有兩個 Jumper 可用以選擇是使用 Arduino 的硬體 UART(D0/D1 : 1,200 ~ 115,200bps)或軟體的 D5/D4 Port(1,200 ~ 38,400bps)，而 ZigBee 擴充板使用 Arduino I/O 控制腳位如下 ：



- D9 : RESET Pin
  - 當 D9 = LOW : Reset(重置) ZigBee 模組
- D6 : 模式選擇/Mode Select
  - 0 : BroadCast(廣播) mode
  - 1 : Unicast/Setting(點對點/設定) mode

ZA ZigBee 模組針對不同應用提供三種不同的網路模式。主機（Coordinator）和路由設備（Router）所運行模式通過模式選擇 D6 來決定。

Appsdüino

ZigBee RO位址(1,2,.. ~ 2047)

- D0/D5 : TX(ZigBee 的資料傳送) ➔ 接 UNO 的資料接收 D0(HW)或 D5(Software Serial)
- D1/D4 : RX(ZigBee 的資料接收) ➔ 接 UNO 的資料傳送 D1(HW)或 D4(Software Serial)

- ✓ ZigBee 模組 Baud Rate : 目前預設 9,600 bps
- ✓ 使用前請先設定 RO 的位址,其有效位址 : 1,2, ~~ 2047 (11 bits)
- ✓ 擴充板上有連線指示 LED 燈,當正在搜尋連線時,則 LED 閃爍,已連線時則 LED 恆亮

Appsduino

AppsBee Class(類別)說明



公開屬性資料(Public Data) :

```
int UniMyID : 儲存目前 RO ID 值
int RxSourceID : 儲存接收封包來源之 ID 值
int value_int : 當收到/傳送整數封包時數值存放之位置
float value_float : 當收到/傳送浮點數封包時數值存放之位置
String value_String : 當收到/傳送字串封包時字串存放之位置
TCommand cmd :  當收到/傳送通用數位控制命令存放之類別參數位置
    int TargetID ; //欲傳送之目標 ID
    int SourceID ;  //發送命令之原始 ID
    byte CommandID ;//可當作控制命令型態
    byte Parameter ; //可當作控制腳位

byte RxCmd : 紀錄收到資料之型態(_isString、_isInt、_isFloat、
_isGCommand)
```

公開函數(Public Methods) :

```
AppsBee  AppsBeeName(TX,RX,role) ; //宣告 AppsBee 物件(軟體 UART)
AppsBee  AppsBeeName(Serial,role) ; //宣告 AppsBee 物件(硬體 UART)
void begin(long speed); //設定傳輸速率(Baud Rate) ,預設為 9600 bps
boolean receive() ;   // 檢查是否有收到封包資料
void SendString(int TargetID, String value); //傳輸字串
void SendFloat(int TargetID, float value);    //傳輸浮點數
void SendInt(int TargetID, int value);        //傳輸整數
void SendGCommand(int TargetID, TCommand cmd);//傳輸通用控制命令
char *SettingMode(settingType mode,long SettingData) ;//參數設定
```

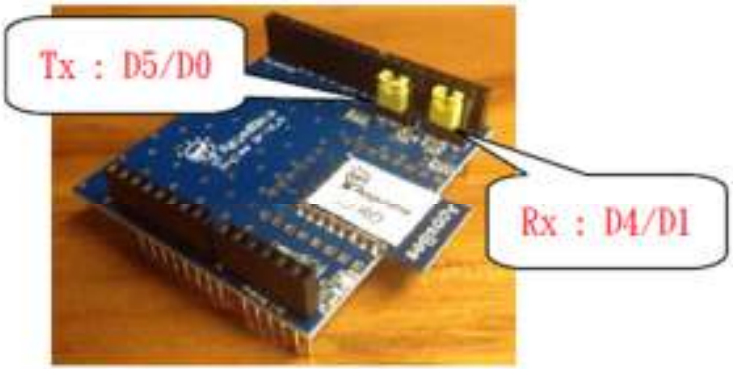
Appsduino

宣告AppsBee物件(需區分CO 與 RO)

◆ 使用 Software Serial 方式(使用腳位 D5/D4，Jumper 需調至 D5/D4)：

```
AppsBee  AppsBeeName(TX,RX,role) ; //宣告 AppsBee 物件，名稱為 AppsBeeName
```

TX ： AppsBee 傳送之腳位(D5)
RX ： AppsBee 接收之腳位(D4)
role ：
- CO ： 表示此模組是主機(CO)
- RO ： 表示此模組是路由(RO)



範例 ：

```
AppsBee AZbee(5,4,RO);//  宣告一個 AppsBee 的 RO 物件，名稱為 AZbee
AppsBee AZbee(5,4,CO);//  宣告一個 AppsBee 的 CO 物件，名稱為 AZbee
```

Appsduino

宣告AppsBee物件(需區分CO 與 RO)

◆ 使用硬體 Serial 方式(使用腳位 D0/D1， Jumper 需調至 D0/D1)：

```
AppsBee  AppsBeeName(Serial,role); //宣告 AppsBee 物件，名稱為 AppsBeeName
```

TX : AppsBee 傳送之腳位(D0)
RX : AppsBee 接收之腳位(D1)
role :
- CO : 表示此模組是主機(CO)
- RO : 表示此模組是路由(RO)

範例 :

```
AppsBee AZbee(Serial,RO);//宣告一個使用 H/W UART 的 AppsBee RO 物件，名稱為 AZbee
AppsBee AZbee(Serial,CO);//宣告一個使用 H/W UART 的 AppsBee CO 物件，名稱為 AZbee
```

Appsduino

AppsBee 初始函數：begin()

```
// Baud rate 設定(使用 AppsBee 模組前需設定傳輸速率(Baud Rate))

void begin(long speed); //預設為 9600 bps
```

範例：

```
AppsBee AZbee(5,4,RO);//  宣告一個 AppsBee 的 RO 物件，名稱為 AZbee

 Void setup()
{
    AZbee.begin(9600) ; // 設定 AZbee baud rate = 9600bps
}
```

Appsduino

AppsBee 傳送接收資料型態

目前 AppsBee 程式庫可傳送與接收四種資料型態 ： 整數、浮點數、字串(String) 與通用數位控制命令(TCommand)

使用四個不同的Form Type 來表示接收端接收到的資料型態，並將其儲存於公開變數 RxCmd 變數中，其四種資料型態如下所述 ：

_isString ： 表示收到字串
_isInt ： 表示收到整數
_isFloat ： 表示收到浮點數
_isGCommand ： 表示收到通用數位控制命令

Appsduino

AppsBee 傳送函數

```
void SendString(int TargetID, String value); //傳輸字串
void SendFloat(int TargetID, float value);    //傳輸浮點數
void SendInt(int TargetID, int value);        //傳輸整數
void SendGCommand(int TargetID, TCommand cmd);//傳輸通用控制命令
```

TargetID：為資料將傳送之目的位址(destination address)

- BROADCAST or 0xFFFF ：表示廣播(即將資料傳送給所有 nodes)
- SENDtoCO or 0 ：表示將資料傳給主機(CO)/0，CO 的 ID = 0
- 其餘 ID 編號(整數) ： 表示將資料傳給指定 ID

Appsduino

AppsBee 傳送函數範例

```
AppsBee AZbee(5,4,RO);//  宣告一個 AppsBee 的 RO 物件，名稱為 AZbee

String  Str =  "Hello World !!" ; // 定義字串
float   float_Value = 12345.67 ;  // 定義浮點數
int     int_value   = 12345 ;     // 定義整數

Void setup()
{
    AZbee.begin(9600) ; // 設定 AZbee baud rate = 9600bps

    AZbee.SendString(BROADCAST, Str); //將字串 Str 廣播給所有 nodes
    AZbee.SendFloat(SENDtoCO,float_value);//將浮點數傳送給 CO
    AZbee.SendInt(2,int_value);// 將整數傳送給 ID=2 的node
    //-------- 設定通用控制命令的參數 --------------------
    AZbee.cmd.TargetID = 2 ; //傳送目標 ID 設定
    AZbee.cmd.CommandID = 0 ;//命令型態(Cmd Type)，ex:設定腳位為 LOW
    AZbee.cmd.Parameter = 12;//ex : 欲控制的腳位(Pin 12)
    AZbee.SendGCommand(AZbee.cmd.TargetID,AZbee.cmd);// 將通用控制命令
                                                     // 傳送給 ID=2 的node
}

Void loop()
{

} // end loop()
```

Appsduino

AppsBee 接收/檢查封包函數

```
boolean receive(); // 檢查是否有收到封包 ? true : 有收到；false : 無封包
```

當 receive()回傳 true 時，可根據 RxCmd 之資料型態是_isString、_isInt、_isFloat 或_isGCommand 來讀取其相對應的變數 : value_String、value_int、value_float 或其內的 TCommand Cmd 的參數即可

範例 :

```
AppsBee AZbee(5,4,R0);//  宣告一個 AppsBee 的 R0 物件，名稱為 AZbee
Void setup()
{
    AZbee.begin(9600) ; // 設定 AZbee baud rate = 9600bps
}
```

Appsduino

AppsBee 接收函數範例

```
Void loop()
{
    if (AZbee.receive()) { //檢查是否 AppsBee 模組有收到資料 ?
                           // 如果 AppsBee 接收到資料，則檢查收到之資料型態
        lcd.setCursor(0,1);   // 設定資料顯示在 LCD 的第二行的起頭
        switch(AZbee.RxCmd) { //判斷是收到哪一個類型的資料
            case _isString :   // 接收到字串資料 String
                lcd << AZbee.value_String  ; //將字串顯示在 LCD 上
                lcd << AZbee.RxSourceID ; // 顯示字串之來源 ID
                break;
            case _isFloat :    // 接收到浮點數資料
                lcd << _FLOAT(AZbee.value_float,1);//將收到浮點數顯示在 LCD 上
                lcd << AZbee.RxSourceID ; // 顯示浮點數之來源 ID
                break;
            case _isInt :      // 接收到整數資料
                lcd << AZbee.value_int ; //將收到整數顯示在 LCD 上
                lcd << AZbee.RxSourceID ; // 顯示整數之來源 ID
                break;
            case _isGCommand : //  收到通用命令封包
                lcd << AZbee.cmd.TargetID <<","<< AZbee.cmd.CommandID << "," <<
                AZbee.cmd.Parameter ; //將收到通用命令的參數顯示在 LCD 上
                TCmdHandler(AZbee.cmd);//處理通用控制命令
                break;
        }
} // loop()
```

Appsduino

AppsBee 接收函數範例

```
//------------------- TCommand Handler ----------------------
// input : TCommand
// cmd : cmd.CommandID 命令型態可自行依需求定義或增加(此範例僅是一種用法)
//       - 0 : set pin cmd.Parameter to LOW(設定腳位為 LOW)
//       - 1 : set pin cmd.Parameter to HIGH(設定腳位為 HIGH)
//       - 2 : set pin cmd.Parameter to toggle(設定腳位為狀態變換)
//       - others : user define
//-----------------------------------------------------------

void TCmdHandler(TCommand cmd)
{
switch(cmd.CommandID) {
    case 0 :  // ->L
       digitalWrite(cmd.Parameter, LOW);
       break;
    case 1 :  // ->H
       digitalWrite(cmd.Parameter, HIGH);
       break;
    case 2 :  // Toogle : H->L/L->H
       digitalWrite(cmd.Parameter, !digitalRead(cmd.Parameter));
       break;
    default : //
       break ;
  }
}
```

Appsduino

AppsBee 參數查詢與設定函數

可利用 SettingMode()函數查詢與設定 AppsBee 模組的多種設定參數，包括網路 ID、傳輸速率、通訊頻道以及 RO/路由模組的 ID，此函數會回傳目前資料或設定是否成功之相關訊息。

當 SettingData 參數為 0 時，則為查詢指令，SettingMode()函數會回傳目前 AppsBee 模組設定之資料。

```
//-------- AppsBee Setting Mode Method ---------------
//  mode : setting mode from enum settingType in AppsBee.h
//  Setting data :
//          0/long : 0 - inquiry(查詢指令), long - setting data
//          data   : the setting data to AppsBee module
//  return  :  response message from AppsBee
//-----------------------------------------------------------
```

Appsduino

AppsBee 參數查詢與設定函數

```
char *SettingMode(settingType mode,long SettingData) ;// 參數設定

mode : //設定模式
    PANID :      // 設定網路 ID(有效值為 1 - 65535)
                 // 一個 CO 可建置一個專屬的 ZigBee 網路，不同 CO 建置的 ZigBee
                 // 網路需使用不同的網路 ID 或通訊頻道予以區隔，以避免混淆
                 // SettingData : 1 - 65535,預設/default : 4372

    BAUD_RATE : // 設定傳輸速率(有效值為 1200,...,115200)
                 // SettingData : 1200/2400/.../115200, 預設 : 9600bps

    CHANNEL :    // 設定通訊頻道(可設定單頻道或多頻道)
                 // AppsBee 在 2.4Ghz 頻段提供 ZigBee 規格的頻道 11,12,...,26,
                 // 頻道乃以 32 Bit(4 個位元組)來對應頻道 0 ~頻道 31(有效為
                 // 頻道 11~26)，可設定相對應的 Bit=1 來讓該模組運行於單頻道或
                 // 多頻道多頻道
                 // 預設為頻道 11(即 00000000 00000000 00100000 00000000)=8192
                 // SettingData : 11 - 26(單頻道設定); 預設 : 13
                                : 00000xxx xxxxxxxx xxxxx000 00000000
                                : 設定上面相對應的 Bit=1(多頻道設定)

    UNI_SEC_ADDR : //設定 RO 之 ID(有效值為 1,..2047 ; -1 : 表未設定 ID)
                 // SettingData : 1 - 2047(有效值); 預設 : -1
```

Appsduino

AppsBee 參數查詢與設定範例

```
AppsBee AZbee(5,4,RO);//   宣告一個 AppsBee 的 RO 物件，名稱為 AZbee

 Void setup()
{
    AZbee.begin(9600) ; // 設定 AZbee baud rate = 9600bps
    Serial.begin(9600) ; // 設定 Serial baud rate = 9600

    Serial << AZbee.SettingMode(BAUD_RATE,0); //查詢目前的傳輸速率
    Serial << endl ; // move to new line(End of Line)

    Serial << AZbee.SettingMode(UNI_SEC_ADDR,0) ; // 查詢目前的 ID
    Serial << endl ; // move to new line

    Serial << AZbee.SettingMode(PANID,1234) ; // 將網路 ID 設定為 1234
    Serial << endl ; // move to new line

    Serial << AZbee.SettingMode(CHANNEL,20) ; // 將通訊頻道設定為頻道 20
    Serial << endl ; // move to new line

    AZbee.SettingMode(CHANNEL,B00000000 00000000 0111000 00000000) ;
                                    // 將通訊頻道設定為頻道 11/12/13
    Serial << AZbee.SettingMode(UNI_SEC_ADDR,1234) ;
                                    // 將 RO ID 設定為 1234
    Serial << endl ; // move to new line

    Serial << AZbee.SettingMode(BAUD_RATE,19200) ;
                                    // 將傳輸速率設定為 19200 bps
    Serial << endl ; // move to new line
}
```

Appsduino

ZigBee 溫度接收程式



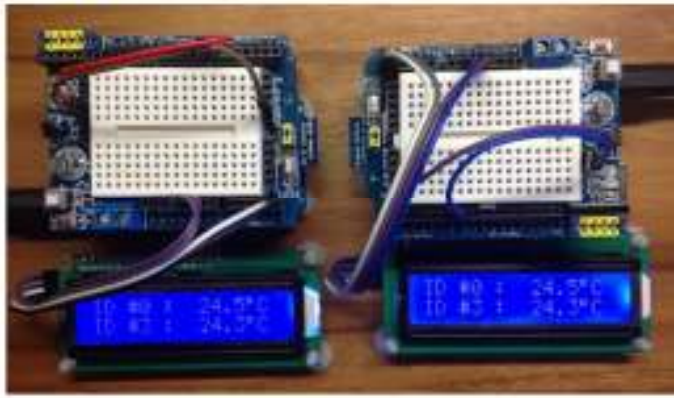
```
#include <Streaming.h>
#include <SoftwareSerial.h>
#include <AppsBee.h>

//--宣告 AppsBee 物件
//RX, TX    Software Serial support 9600bps ~ 57600bps,can't support 115200bps
over
AppsBee AZbee(5, 4, RO);      //AppsBee(TX, RX, role)
//AppsBee AZbee(5, 4, CO);       //AppsBee(TX, RX, role)
//AppsBee AZbee(Serial, RO);    //RO 使用硬體 serial,AppsBee(Serial,Role)
//AppsBee AZbee(Serial, CO);    //CO 使用硬體 serial,AppsBee(Serial,Role)
```

Appsduino

ZigBee 溫度接收程式

```
void loop()
{
  tick.update(); // check timer event
  CheckAppsBeeReceiveData();   //檢查 Zigbee 是否接收到資料
}
//-----------------------------------------------------
// 檢查 Zigbee 是否接收到資料
void CheckAppsBeeReceiveData()
{
  if (AZbee.receive()) {
    // 如果 Zigbee 接收到資料
      lcd.setCursor(4, AZbee.RxSourceID % 2) ; // display message based on source ID
      lcd << AZbee.RxSourceID << " : " ; // display source ID
      switch(AZbee.RxCmd) { //判斷是收到哪一個類型的資料
        case _isString :   // 接收到字串資料 _isString
          lcd << AZbee.value_String  ;
          break;
        case _isFloat :    // 接收到浮點數資料 _isFloat
          lcd <<  _FLOAT(AZbee.value_float, 1) << (char)(0xDF) << "C " ;// LCD degree "o" Char
          break;
        case _isInt :      // 接收到整數資料 _isInt
          lcd << AZbee.value_int ;
          break;
        default:
          break;
      }
  }
}
```

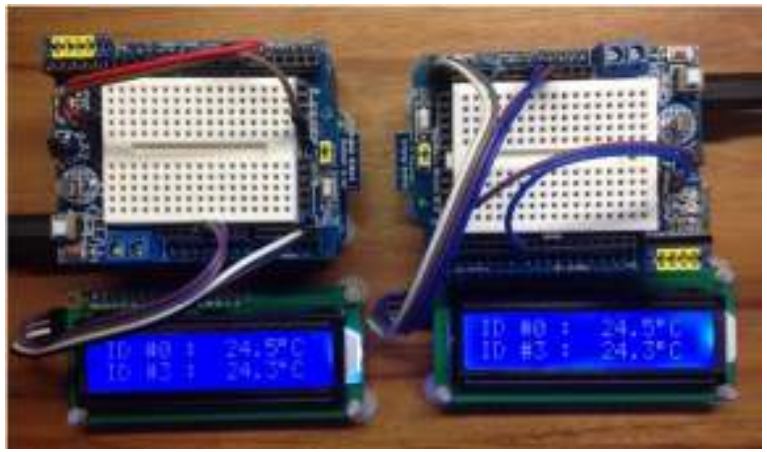


Appsduino

ZigBee CO 群發資料 to ROs

```
/*
 Data Format : Temperature
 Timer Interval : 1 sec
*/
void OneSecTimerHandler() //-- update per sec --
{
    float Temp ;

    Temp = dd.getTemperature();     // read the temperature from DS18B20
    lcd.setCursor(0,AZbee.UniMyID % 2);      // Display from 2nd Row
    lcd << "ID #" << AZbee.UniMyID << " : " << _FLOAT(Temp,1) << (char) (0xDF) << "C " ;
    AZbee.SendFloat(BROADCAST,Temp) ; // Tx Temperature to CO
}
```



Appsduino

TempCDS_CO 接收資料 from ROs

TempCDS_CO

```
#include <Streaming.h>
#include "SoftwareSerial.h"
#include <AppsBee.h>

// 設定與PC的連線方式
//#define PC_Serial  Serial  // for UNO or MEGA CO
//#define PC_Serial  Serial1    // only for MEGA CO
#define PC_Serial  Serial    // UNO

// 設定與PC的連線速度
#define PC_speed 115200

//-----------------------------
//宣告Zigbee物件
//以下是 3 種不同CO的設定方式
//-----------------------------
AppsBee AZbee(5, 4, CO);    //UNO  CO使用軟體serial   AppsBee(TX, RX, role)
//AppsBee AZbee(11, 4, CO);   //MEGA CO使用軟體 Serial   AppsBee(TX, RX, role)
//AppsBee AZbee(Serial2, CO);   //MEGA CO使用硬體 Serial2

long speed=9600;  // Zigbee 的傳輸速度
```

Appsduino

TempCDS_CO 接收資料 from ROs

```
//-----------------------------------------------------
#include <i2cmaster.h>
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>  // F Malpartida's NewLiquidCrystal library
#define I2C_ADDR    0x27  // Define I2C Address for the PCF8574T
//---(Following are the PCF8574 pin assignments to LCD connections )----
// This are different than earlier/different I2C LCD displays

#define BACKLIGHT_PIN  3
#define LED_OFF 1
#define LED_ON  0

/*-----( Declare objects )-----*/
LiquidCrystal_I2C  lcd(I2C_ADDR,2,1,0,4,5,6,7); // declare I2C LCD object

//-----------------------------------------------------
long startTime;
const long duration = 1000;  //控制每秒檢查一次事件
#define   GreenLed    13
#define   CDS         A2
#define   Buzzer      8
#define   RedLed      11
#define   BlueLed     10
```

Appsduino

TempCDS_CO 接收資料 from ROs

```
void setup()
{
  pinMode(GreenLed,OUTPUT) ;
  pinMode(CDS,INPUT_PULLUP) ;
  pinMode(RedLed, OUTPUT);
  pinMode(BlueLed, OUTPUT);
  pinMode(Buzzer, OUTPUT);
//-----------------------
  PC_Serial.begin(115200);

  //------- Timer ------------
  lcd.begin (16,2);  // initialize the lcd
  // Switch on the backlight
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(LED_ON);
  lcd.backlight();  //Backlight ON if under program control
  lcd.clear();
  lcd.setCursor(0,0); //Start at character 0 on line 0
  lcd.print(F("Connecting...")) ;
  //------------------------------------
  AZbee.begin(speed);
  lcd.setCursor(0,0); //Start at character 0 on line 0
  lcd << F("ID#") << AZbee.UniMyID << "          ";
  //------------------------------------
  startTime  = millis();
}
```

Appsduino

TempCDS_CO 接收資料 from ROs

```
void loop()
{
  // 讓LED每兩秒閃爍一次
  if (millis()-startTime >= duration) {
    digitalWrite(GreenLed, !digitalRead(GreenLed));
    startTime  = millis();
  }
  CheckZigBeeReceiveData();   //檢查Zigbee是否接收到資料
}
//-----------------------------------------------------
// 檢查Zigbee是否接收到資料
void CheckZigBeeReceiveData()
{
  // 如果 Zigbee 接收到資料
  if (AZbee.receive()) {
    switch(AZbee.RxCmd) { //判斷是收到哪一個類型的資料
      case _isString :      // 接收到字串資料
        PC_Serial << "[" << AZbee.RxSourceID << "," << AZbee.value_String << "]" << '\n' ;  //將字串資料後送給PC
        lcd.setCursor(0,1); //Start at character 0 on line 0
        lcd << AZbee.RxSourceID << "," << AZbee.value_String ;
        break;
      case _isGCommand : //  收到通用命令封包
        lcd << AZbee.cmd.TargetID  << "," << AZbee.cmd.CommandID << "," << AZbee.cmd.Parameter ;
        TCmdHandler(AZbee.cmd); // execute the related function
        break;
    }
  }
}
```

Appsduino

TempCDS_CO 接收資料 from ROs

```
//---------------- TCommand Handler --------------------
// input : TCommand
// cmd :
//      - 0 : set pin cmd.Parameter to LOW
//      - 1 : set pin cmd.Parameter to HIGH
//      - 2 : set pin cmd.Parameter to toggle
//      - others : user define
//-------------------------------------------------------

void TCmdHandler(TCommand cmd)
{
  switch(cmd.CommandID) {
    case 0 :  // ->L
       digitalWrite(cmd.Parameter, LOW);
       break;
    case 1 :  // ->H
       if(cmd.Parameter == Buzzer)
          tone(Buzzer,262,1000) ;
       else
          digitalWrite(cmd.Parameter, HIGH);
       break;
    case 2 :  // Toogle : H->L/L->H
       digitalWrite(cmd.Parameter, !digitalRead(cmd.Parameter));
       break;
    default : //
       break ;
  }
}
```

Appsdüino

TempCDS_RO 傳送資料 to CO

TempCDS_RO

```
//-------------------------------------------------
//   每秒讀一次溫度(DS18B20)並廣播給所有的Nodes(CO/ROs)
//   將J19的DQ Pin接至D7,Vdd 接至 5V
//-------------------------------------------------
#include <Streaming.h>
#include <SoftwareSerial.h>
#include <AppsBee.h>

//--宣告 AppsBee 物件
//RX, TX  Software Serial support 9600bps ~ 57600bps,can't support 115200bps over
AppsBee AZbee(5, 4, RO);   //AppsBee(TX, RX, role)
//AppsBee AZbee(5, 4, CO);    //AppsBee(TX, RX, role)
//AppsBee AZbee(Serial, RO); //RO使用硬體serial,AppsBee(Serial,Role)
//AppsBee AZbee(Serial, CO); //CO使用硬體serial,AppsBee(Serial,Role)

/*-----( Import needed libraries )-----*/
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>  // F Malpartida's NewLiquidCrystal library
//-----( Declare Constants )-----
#define I2C_ADDR    0x27  // Define I2C Address for the PCF8574T
//---(Following are the PCF8574 pin assignments to LCD connections )----
#define BACKLIGHT_PIN  3
#define  LED_OFF  1
#define  LED_ON  0
/*-----( Declare objects )-----*/
LiquidCrystal_I2C  lcd(I2C_ADDR,2,1,0,4,5,6,7); // declare I2C LCD object
```

Appsduino

TempCDS_RO 傳送資料 to CO

```
//------------------------------
#include <Timer.h>
Timer tick ;// setup one sec timer event
//-------- DS18B20 Temperature sensor ------------
#define   DS18B20_Pin   7  //Define DS18S20 onewire signal pin on D7
#include <OneWire.h>
#include <DS18B20.h>
DS18B20 dd(DS18B20_Pin);  // on digital pin 7

#define   CDS         A2
#define   Buzzer      8
#define   RedLed      11
#define   BlueLed     10
#define   Target_ID   BROADCAST  //Specify the Target ID
//-------------------------------------------------------

void setup()
{
   pinMode(CDS,INPUT_PULLUP) ;
   pinMode(RedLed, OUTPUT);
   pinMode(BlueLed, OUTPUT);
   pinMode(Buzzer, OUTPUT);
//-------------------------------------------------------------
```

Appsduino

TempCDS_RO 傳送資料 to CO

```
  AZbee.begin(9600); // initialize AppsBee with baud rate = 9600 bps
  Serial.begin(9600) ; // initialize H/W UART baud rate
  lcd.begin(16, 2);    //Initialize LCD as 16 x 2
  // Switch on the backlight
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(LED_ON);
  lcd.backlight();  //Backlight ON if under program control
  lcd.setCursor(0,1);     // Display from 2nd Row
  lcd << "#" << AZbee.UniMyID << ":" ; // Display current ID
//---------- Set up Timer -------
  tick.every(2000, SecTimerHandler);// read temperature/humidity once 2 second
//---------------- 通用數位控制指令 TCommand cmd ------------------------------------
  AZbee.cmd.TargetID = Target_ID ;
  AZbee.cmd.CommandID = 1 ; // 0 : set to LOW, 1 : set to HIGH, 2 : Toogle Light Cmd
  AZbee.cmd.Parameter = RedLed ; // default to turn on Buzzer
  if (AZbee.cmd.TargetID != AZbee.UniMyID)
     AZbee.SendGCommand(AZbee.cmd.TargetID,AZbee.cmd);  //PackFrame06(cmd.TargetID, cmd);

  delay(1000) ;
  AZbee.SendString(BROADCAST,"Hello ! I'm Tony") ; // Broadcast message to all nodes
}
```

Appsduino

TempCDS_RO 主程式

```
void loop()
{
  tick.update(); // check timer event
  CheckAppsBeeReceiveData();   //檢查Zigbee是否接收到資料
}
```

Appsduino

TempCDS_RO 傳送資料 to CO

```
// 檢查Zigbee是否接收到資料
void CheckAppsBeeReceiveData()
{
  if (AZbee.receive()) {
    // 如果 Zigbee 接收到資料
      lcd.setCursor(4, AZbee.RxSourceID % 2) ; // display message based on source ID
      lcd << AZbee.RxSourceID << " : " ; // display source ID
      switch(AZbee.RxCmd) { //判斷是收到哪一個類型的資料
        case _isString :   // 接收到字串資料 _isString
          lcd << AZbee.value_String  ;
          Serial << "Received ID #" << AZbee.RxSourceID << " : " << AZbee.value_String << endl ; ;
          break;
        case _isFloat :    // 接收到浮點數資料 _isFloat
          lcd <<  _FLOAT(AZbee.value_float, 1) << (char)(0xDF) << "C " ;// LCD degree "o" Char
          Serial << "Received ID #" << AZbee.RxSourceID << " : " << _FLOAT(AZbee.value_float, 1) << endl ;
          break;
        case _isInt :      // 接收到整數資料 _isInt
          lcd << AZbee.value_int ;
          Serial << "Received ID #" << AZbee.RxSourceID << " : " << AZbee.value_int << endl ; ;
          break;
        case _isGCommand : //  收到通用命令封包
          lcd << AZbee.cmd.TargetID  << "," << AZbee.cmd.CommandID << "," << AZbee.cmd.Parameter ;
          TCmdHandler(AZbee.cmd); // execute the related function
          break;
      }
  }
}
```

Appsduino

TempCDS_RO 傳送資料 to CO

```
/*
 Data Format : Temperature
 Timer Interval : 1 sec
*/
void SecTimerHandler() //-- update per sec --
{
    float Temp ;
    String sensorMsg ="" ;

    sensorMsg += (int) (dd.getTemperature() * 10) ;    // read the temperature from DS18B20
    char lastDigit = sensorMsg.charAt(sensorMsg.length()-1) ;
    sensorMsg.setCharAt(sensorMsg.length()-1,'.') ; // divide 10.0 to get real temperature
    sensorMsg += lastDigit; // restore the digital after point
    sensorMsg += ","; // Add ',' separater
    sensorMsg += analogRead(CDS) ;
    lcd.setCursor(0,AZbee.UniMyID % 2);     // Display from 2nd Row
    lcd << "#" << AZbee.UniMyID << ":" << sensorMsg  ;
    AZbee.SendString(SENDtoCO,sensorMsg) ; // Tx Temperature to CO
 }
```

Appsduino

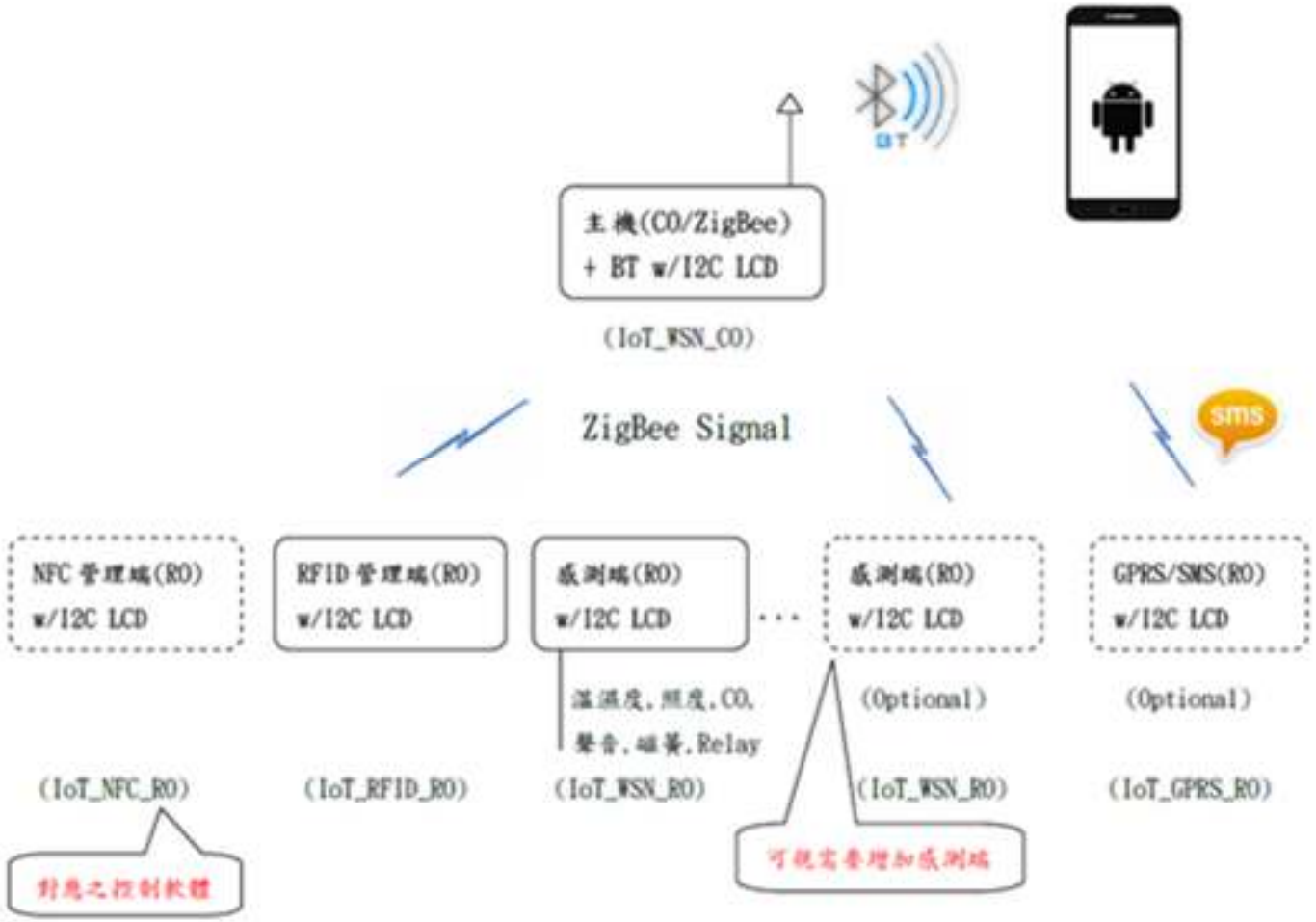
TempCDS_RO 傳送資料 to CO

```
//---------------- TCommand Handler --------------------
// input : TCommand
// cmd :
//      - 0 : set pin cmd.Parameter to LOW
//      - 1 : set pin cmd.Parameter to HIGH
//      - 2 : set pin cmd.Parameter to toggle
//      - others : user define
//-------------------------------------------------------

void TCmdHandler(TCommand cmd)
{
  switch(cmd.CommandID) {
    case 0 :  // ->L
       digitalWrite(cmd.Parameter, LOW);
       break;
    case 1 :  // ->H
       if(cmd.Parameter == Buzzer)
          tone(Buzzer,262,1000) ;
       else
          digitalWrite(cmd.Parameter, HIGH);
       break;
    case 2 :  // Toogle : H->L/L->H
       digitalWrite(cmd.Parameter, !digitalRead(cmd.Parameter));
       break;
    default : //
       break ;
  }
}
```

Appsduino

物聯網無線感測應用發展系統

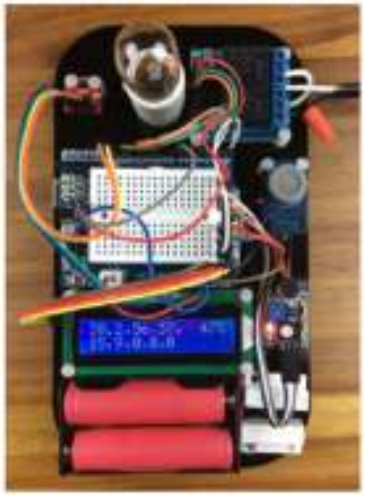


Appsdüino

物聯網無線感測應用發展系統



主控端(CO)



感測端(RO)



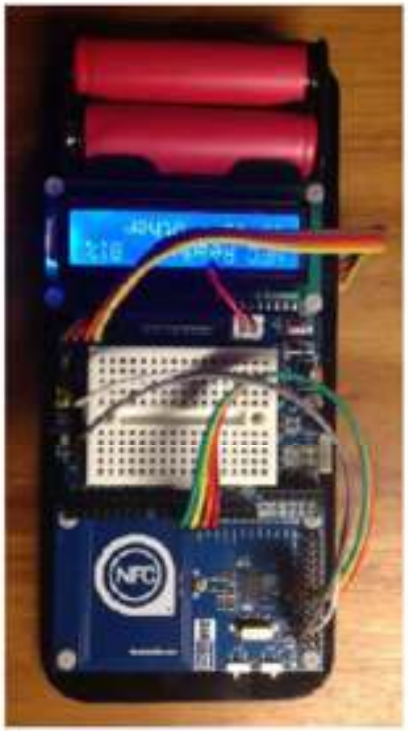
RFID 管理端(RO)



IoTWSN App

Appsdüino

物聯網NFC無線感測應用模組



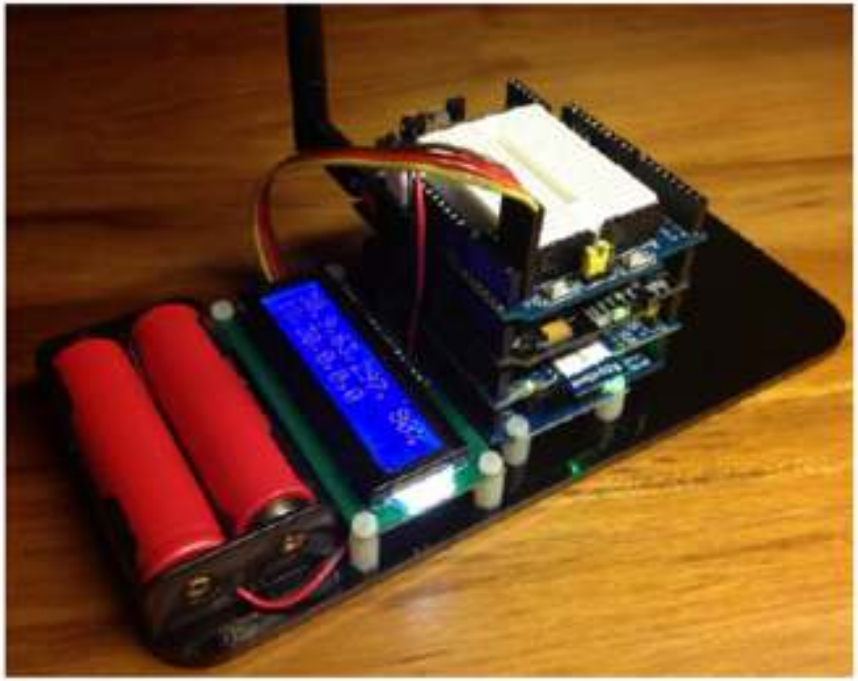
NFC 管理端(RO)



Android App

Appsdüino

物聯網GPRS/SMS監控與訊息模組



GPRS/SMS 訊息管理端(RO)



GPRS 訊息控制與通知

Appsduino


多謝/Thanks for Listening !
email : tp6up454@gmail.com
Mobile : 0932-621380