

3

CHAPTER

Arduino 程式語法

官方網站函式網頁

讀者若對本章節程式結構不了解之處，請參閱如圖 197 所示之 Arduino 官方網站的 Language Reference (<http://arduino.cc/en/Reference/HomePage>)，或參閱相關書籍(Anderson & Cervo, 2013; Boxall, 2013; Faludi, 2010; Margolis, 2011, 2012; McRoberts, 2010; Minns, 2013; Monk, 2010, 2012; Oser & Blemings, 2009; Warren, Adams, & Molle, 2011; Wilcher, 2012)，相信會對 Arduino 程式碼更加了解與熟悉。

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)
- + `/` (division)

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

圖 197 Arduino 官方網站的 Language Reference

資料來源：Language Reference (<http://arduino.cc/en/Reference/HomePage>)

Arduino 程式主要架構

程式結構

- setup()
- loop()

一個 Arduino 程式碼(Sketch)由兩部分組成

setup()

程式初始化

void setup()

在這個函式範圍內放置初始化 Arduino 開發板的程式 - 在重複執行的程式 (loop())之前執行，主要功能是將所有 Arduino 開發板的 Pin 腳設定，元件設定，需要初始化的部分設定等等。

變數型態宣告區 ; // 這裡定義變數或 IO 腳位名稱

void setup()

```
{
  

}
```

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);

loop()

迴圈重複執行

void loop()

在此放置你的 Arduino 程式碼。這部份的程式會一直重複的被執行，直到 Arduino 開發板被關閉。

void loop()

```
{
}

```

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

區塊式結構化程式語言

C 語言是區塊式結構的程式語言，所謂的區塊是一對大括號：『{ }』所界定的範圍，每一對大括號及其涵括的所有敘述構成 C 語法中所謂的複合敘述 (Compound Statement)，這樣子的複合敘述不但對於編譯器而言，構成一個有意義的文法單位，對於程式設計者而言，一個區塊也應該要代表一個完整的程式邏輯單元，內含的敘述應該具有相當的資料耦合性（一個敘述處理過的資料會被後面

的敘述拿來使用)，及控制耦合性 (CPU 處理完一個敘述後會接續處理另一個敘述指定的動作)，當看到程式中一個區塊時，應該要可以假設其內所包含的敘述都是屬於某些相關功能的，當然其內部所使用的資料應該都是完成該種功能所必需的，這些資料應該是專屬於這個區塊內的敘述，是這個區塊之外的敘述不需要的。

命名空間 (naming space)

C 語言中區塊定義了一塊所謂的命名空間 (naming space)，在每一個命名空間內，程式設計者可以對其內定義的變數任意取名字，稱為區域變數 (local variable)，這些變數只有在該命名空間 (區塊) 內部可以進行存取，到了該區塊之外程式就不能在藉由該名稱來存取了，如下例中 `int` 型態的變數 `z`。由於區塊是階層式的，大區塊可以內含小區塊，大區塊內的變數也可以在內含區塊內使用，例如：

```
{
    int x, r;
    x=10;
    r=20;
    {
        int y, z;
        float r;
        y = x;
        x = 1;
        r = 10.5;
    }
    z = x; // 錯誤，不可使用變數 z
}
```

上面這個例子裡有兩個區塊，也就有兩個命名空間，有任一個命名空間中不可有兩個變數使用相同的名字，不同的命名空間則可以取相同的名字，例如變數

r，因此針對某一個變數來說，可以使用到這個變數的程式範圍就稱為這個變數的作用範圍 (scope)。

變數的生命期 (Lifetime)

變數的生命始於定義之敘述而一直延續到定義該變數之區塊結束為止，變數的作用範圍：意指程式在何處可以存取該變數，有時變數是存在的，但是程式卻無法藉由其名稱來存取它，例如，上例中內層區塊內無法存取外層區塊所定義的變數 r，因為在內層區塊中 r 這個名稱賦予另一個 float 型態的變數了。

縮小變數的作用範圍

利用 C 語言的區塊命名空間的設計，程式設計者可以儘量把變數的作用範圍縮小，如下例：

```
{
int tmp;
    for (tmp=0; tmp<1000; tmp++)
        doSomething();
}
{
    float tmp;
    tmp = y;
    y = x;
    x = y;
}
```

上面這個範例中前後兩個區塊中的 tmp 很明顯地沒有任何關係，看這個程式的人不必擔心程式中有藉 tmp 變數傳遞資訊的任何意圖。

特殊符號

;(semicolon)

{ } (curly braces)

// (single line comment)

/* */ (multi-line comment)

Arduino 語言用了一些符號描繪程式碼，例如註解和程式區塊。

;(分號)

Arduino 語言每一行程序都是以分號為結尾。這樣的語法讓你可以自由地安排代碼，你可以將兩個指令放置在同一行，只要中間用分號隔開（但這樣做可能降低程式的可讀性）。

範例：

```
delay(100);
```

{}(大括號)

大括號用來將程式代碼分成一個又一個的區塊，如以下範例所示，在 `loop()` 函式的前、後，必須用大括號括起來。

範例：

```
void loop(){  
    Serial.println("Hello !! Welcome to Arduino world");  
}
```

註解

程式的註解就是對代碼的解釋和說明，撰寫註解有助於程式設計師(或其他人)了解代碼的功能。

Arduino 處理器在對程式碼進行編譯時會忽略註解的部份。

Arduino 語言中的撰寫註解有兩種方式

```
//單行註解：這整行的文字會被處理器忽略
/*多行註解：
    在這個範圍內你可以
    寫 一篇 小說
*/
```

變數

程式中的變數與數學使用的變數相似，都是用某些符號或單字代替某些數值，從而得以方便計算過程。程式語言中的變數屬於識別字 (identifier)，C 語言對於識別字有一定的命名規則，例如只能用英文大小寫字母、數字以及底線符號

其中，數字不能用作識別字的開頭，單一識別字裡不允許有空格，而如 `int`、`char` 為 C 語言的關鍵字 (keyword) 之一，屬於程式語言的語法保留字，因此也不能用為自行定義的名稱。通常編譯器至少能讀取名稱的前 31 個字元，但外部名稱可能只能保證前六個字元有效。

變數使用前要先進行宣告 (declaration)，宣告的主要目的是告訴編譯器這個變數屬於哪一種資料型態，好讓編譯器預先替該變數保留足夠的記憶體空間。宣告的方式很簡單，就是型態名稱後面接空格，然後是變數的識別名稱

常數

- HIGH | LOW
- INPUT | OUTPUT
- true | false
- Integer Constants

資料型態

- boolean
- char
- byte

- int
- unsigned int
- long
- unsigned long
- float
- double
- string
- array
- void

常數

在 Arduino 語言中事先定義了一些具特殊用途的保留字。HIGH 和 LOW 用來表示你開啟或是關閉了一個 Arduino 的腳位(Pin)。INPUT 和 OUTPUT 用來指示這個 Arduino 的腳位(Pin)是屬於輸入或是輸出用途。true 和 false 用來指示一個條件或表示式為真或是假。

變數

變數用來指定 Arduino 記憶體中的一個位置，變數可以用來儲存資料，程式人員可以透過程式碼去不限次數的操作變數的值。

因為 Arduino 是一個非常簡易的微處理器，但你要宣告一個變數時必須先定義他的資料型態，好讓微處理器知道準備多大的空間以儲存這個變數值。

Arduino 語言支援的資料型態:

布林 **boolean**

布林變數的值只能為真(true)或是假(false)

字元 **char**

單一字元例如 A，和一般的電腦做法一樣 Arduino 將字元儲存成一個數字，即使你看到的明明就是一個文字。

用數字表示一個字元時，它的值有效範圍為 -128 到 127。

PS：目前有兩種主流的電腦編碼系統 ASCII 和 UNICODE。

- ASCII 表示了 127 個字元，用來在序列終端機和分時計算機之間傳輸文字。
- UNICODE 可表示的字量比較多，在現代電腦作業系統內它可以用來表示多國語言。

在位元數需求較少的資訊傳輸時，例如義大利文或英文這類由拉丁文，阿拉伯數字和一般常見符號構成的語言，ASCII 仍是目前主要用來交換資訊的編碼法。

位元組 byte

儲存的數值範圍為 0 到 255。如同字元一樣位元組型態的變數只需要用一個位元組(8 位元)的記憶體空間儲存。

整數 int

整數資料型態用到 2 位元組的記憶體空間，可表示的整數範圍為 -32,768 到 32,767；整數變數是 Arduino 內最常用到的資料型態。

整數 unsigned int

無號整數同樣利用 2 位元組的記憶體空間，無號意謂著它不能儲存負的數值，因此無號整數可表示的整數範圍為 0 到 65,535。

長整數 long

長整數利用到的記憶體大小是整數的兩倍，因此它可表示的整數範圍從 -2,147,483,648 到 2,147,483,647。

長整數 unsigned long

無號長整數可表示的整數範圍為 0 到 4,294,967,295。

浮點數 float

浮點數就是用來表達有小數點的數值，每個浮點數會用掉四位元組的 RAM，注意晶片記憶體空間的限制，謹慎的使用浮點數。

雙精準度 浮點數 double

雙精度浮點數可表達最大值為 $1.7976931348623157 \times 10^{308}$ 。

字串 string

字串用來表達文字信息，它是由多個 ASCII 字元組成(你可以透過序串埠發送一個文字資訊或者將之顯示在液晶顯示器上)。字串中的每一個字元都用一個組元組空間儲存，並且在字串的最尾端加上一個空字元以提示 Arduino 處理器字串的結束。下面兩種宣告方式是相同的。

```
char word1 = "Arduino world"; // 7 字元 + 1 空字元
char word2 = "Arduino is a good developed kit"; // 與上行相同
```

陣列 array

一串變數可以透過索引去直接取得。假如你想要儲存不同程度的 LED 亮度時，你可以宣告六個變數 light01，light02，light03，light04，light05，light06，但其實你有更好的選擇，例如宣告一個整數陣列變數如下：

```
int light = {0, 20, 40, 65, 80, 100};
```

"array" 這個字為沒有直接用在變數宣告，而是[]和{}宣告陣列。

控制指令

string(字串)

範例

```
char Str1[15];  
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};  
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};  
char Str4[ ] = "arduino";  
char Str5[8] = "arduino";  
char Str6[15] = "arduino";
```

解釋如下：

- 在 Str1 中 聲明一個沒有初始化的字元陣列
- 在 Str2 中 聲明一個字元陣列(包括一個附加字元)，編譯器會自動添加所需的空字元
- 在 Str3 中 明確加入空字元
- 在 Str4 中 用引號分隔初始化的字串常數，編譯器將調整陣列的大小，以適應字串常量和終止空字元
- 在 Str5 中 初始化一個包括明確的尺寸和字串常量的陣列
- 在 Str6 中 初始化陣列，預留額外的空間用於一個較大的字串

空終止字元

一般來說，字串的結尾有一個空終止字元（ASCII 代碼 0），以此讓功能函數（例如 `Serial.printf()`）知道一個字串的結束，否則，他們將從記憶體繼續讀取後續位元組，而這些並不屬於所需字串的一部分。

這表示你的字串比你想要的文字包含更多的個字元空間，這就是為什麼 Str2 和 Str5 需要八個字元，即使“Arduino”只有七個字元 - 最後一個位置會自動填充

空字元， `str4` 將自動調整為八個字元，包括一個額外的 `null`， 在 `Str3` 的，我們自己已經明確地包含了空字元(寫入 `'\0'`)。

使用符號：單引號?還是雙引號?

- 定義字串時使用雙引號(例如“ABC”)，
- 定義一個單獨的字元時使用單引號(例如'A')

範例

字串測試範例(stringtest01)

```
char* myStrings[]={
    "This is string 1", "This is string 2", "This is string 3",
    "This is string 4", "This is string 5", "This is string 6"};

void setup(){
    Serial.begin(9600);
}

void loop(){
    for (int i = 0; i < 6; i++){
        Serial.println(myStrings[i]);
        delay(500);
    }
}
```

char* 在字元資料類型 `char` 後跟了一個星號 `'*` 表示這是一個“指標”陣列， 所有的陣列名稱實際上是指標，所以這需要一個陣列的陣列。

指標對於 C 語言初學者而言是非常深奧的部分之一， 但是目前我們沒有必要瞭解詳細指標，就可以有效地應用它。

型態轉換

- char()
- byte()
- int()
- long()
- float()

char()

指令用法

將資料轉程字元形態：

語法：char(x)

參數

x: 想要轉換資料的變數或內容

回傳

字元形態資料

unsigned char()

一個無符號資料類型佔用 1 個位元組的記憶體:與 byte 的資料類型相同，無符號的 char 資料類型能編碼 0 到 255 的數位，為了保持 Arduino 的程式設計風格的一致性，byte 資料類型是首選。

指令用法

將資料轉程字元形態：

語法：unsigned char(x)

參數

x: 想要轉換資料的變數或內容

回傳

字元形態資料

```
unsigned char myChar = 240;
```

byte()

指令用法

將資料轉換位元資料形態：

語法：byte(x)

參數

x: 想要轉換資料的變數或內容

回傳

位元資料形態的資料

int(x)

指令用法

將資料轉換整數資料形態：

語法：int(x)

參數

x: 想要轉換資料的變數或內容

回傳

整數資料形態的資料

unsigned int(x)

unsigned int(無符號整數)與整型資料同樣大小，佔據 2 位元組: 它只能用於存儲正數而不能存儲負數，範圍 0~65,535 ($2^{16} - 1$)。

指令用法

將資料轉換整數資料形態：

語法：unsigned int(x)

參數

x: 想要轉換資料的變數或內容

回傳

整數資料形態的資料

```
unsigned int ledPin = 13;
```

long()

指令用法

將資料轉換長整數資料形態：

語法：int(x)

參數

x: 想要轉換資料的變數或內容

回傳

長整數資料形態的資料

unsigned long()

無符號長整型變數擴充了變數容量以存儲更大的資料， 它能存儲 32 位元(4 位元組)資料:與標準長整型不同無符號長整型無法存儲負數， 其範圍從 0 到 4,294,967,295 ($2^{32}-1$) 。

指令用法

將資料轉換長整數資料形態：

語法：unsigned int(x)

參數

x: 想要轉換資料的變數或內容

回傳

長整數資料形態的資料

```
unsigned long time;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("Time: ");
    time = millis();
    //程式開始後一直列印時間
    Serial.println(time);
    //等待一秒鐘，以免發送大量的資料
    delay(1000);
}
```

float()

指令用法

將資料轉換浮點數資料形態：

語法：float(x)

參數

x: 想要轉換資料的變數或內容

回傳

浮點數資料形態的資料

邏輯控制

控制流程

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

Arduinio 利用一些關鍵字控制程式碼的邏輯。

if ... else

If 必須緊接著一個問題表示式(expression)，若這個表示式為真，緊連著表示式後的代碼就會被執行。若這個表示式為假，則執行緊接著 else 之後的代碼。只使用 if 不搭配 else 是被允許的。

範例：

```
#define LED 12
void setup()
{
  int val =1;
  if (val == 1) {
    digitalWrite(LED,HIGH);
  }
}
void loop()
{
}
```

for

用來明定一段區域代碼重覆指行的次數。

範例：

```
void setup()
{
  for (int i = 1; i < 9; i++) {
    Serial.print("2 * ");
    Serial.print(i);
    Serial.print(" = ");
    Serial.print(2*i);

  }
}
void loop()
{
}
```

switch case

if 敘述是程式裡的分叉選擇，switch case 是更多選項的分叉選擇。switch 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 可使程式代碼看起來比較簡潔。

範例：

```
void setup()
{
  int sensorValue;
  sensorValue = analogRead(1);
  switch (sensorValue) {

    case 10:
      digitalWrite(13,HIGH);
      break;

    case 20:
      digitalWrite(12,HIGH);
```

```
break;

default: // 以上條件都不符合時，預設執行的動作
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
}
}
void loop()
{
    }
```

while

當 while 之後的條件成立時，執行括號內的程式碼。

範例：

```
void setup()
{
    int sensorValue;
    // 當 sensor 值小於 256，閃爍 LED 1 燈
    sensorValue = analogRead(1);
    while (sensorValue < 256) {
        digitalWrite(13,HIGH);
        delay(100);
        digitalWrite(13,HIGH);
        delay(100);
        sensorValue = analogRead(1);
    }
}
void loop()
{
    }
```

do ... while

和 **while** 相似，不同的是 **while** 前的那段程式碼會先被執行一次，不管特定的條件式為真或為假。因此若有一段程式代碼至少需要被執行一次，就可以使用 **do...while** 架構。

範例：

```
void setup()
{
  int sensorValue;
  do
  {
    digitalWrite(13,HIGH);
    delay(100);
    digitalWrite(13,HIGH);
    delay(100);
    sensorValue = analogRead(1);
  }
  while (sensorValue < 256);
}
void loop()
{
}
```

break

Break 讓程式碼跳離迴圈，並繼續執行這個迴圈之後的程式碼。此外，在 **break** 也用於分隔 **switch case** 不同的敘述。

範例：

```
void setup()
{
}
void loop()
{
  int sensorValue;
  do {
    // 按下按鈕離開迴圈
```

```

    if (digitalRead(7) == HIGH)
        break;
    digitalWrite(13,HIGH);
    delay(100);
    digitalWrite(13,HIGH);
    delay(100);
    sensorValue = analogRead(1);
}
while (sensorValue < 512);
}

```

continue

continue 用於迴圈之內，它可以強制跳離接下來的程式，並直接執行下一個迴圈。

範例：

```

#define PWMPin 12
#define SensorPin 8
void setup()
{
}
void loop()
{
    int light;
    int x ;
    for (light = 0; light < 255; light++)
    {
        // 忽略數值介於 140 到 200 之間
        x = analogRead(SensorPin) ;

        if ((x > 140) && (x < 200))
            continue;

        analogWrite(PWMPin, light);
        delay(10);
    }
}

```

```
}
```

return

函式的結尾可以透過 **return** 回傳一個數值。

例如，有一個計算現在溫度的函式叫 `computeTemperature()`，你想要回傳現在的溫度給 `temperature` 變數，你可以這樣寫：

```
#define PWMPin 12
#define SensorPin 8

void setup()
{
}

void loop()
{
    int light;
    int x ;
    for (light = 0; light < 255; light++)
    {
        // 忽略數值介於 140 到 200 之間
        x = computeTemperature() ;
        if ((x > 140) && (x < 200))
            continue;

        analogWrite(PWMPin, light);
        delay(10);
    }
}

int computeTemperature() {

    int temperature = 0;
    temperature = (analogRead(SensorPin) + 45) / 100;
    return temperature;
}
```

算術運算

算術符號

- = (給值)
- + (加法)
- (減法)
- * (乘法)
- / (除法)
- % (求餘數)

你可以透過特殊的語法用 **Arduino** 去做一些複雜的計算。+ 和 - 就是一般數學上的加減法，乘法用 * 示，而除法用 / 表示。

另外餘數除法(%), 用於計算整數除法的餘數值: 一個整數除以另一個數，其餘數稱為模數，它有助於保持一個變數在一個特定的範圍(例如陣列的大小)。

語法：

```
result = dividend % divisor
```

參數：

- dividend：一個被除的數字
- divisor：一個數字用於除以其他數

{ } 括號

你可以透過多層次的括弧去指定算術之間的循序。和數學函式不一樣，中括號和大括號在此被保留在不同的用途(分別為陣列索引，和宣告區域程式碼)。

範例：

```
#define PWMPin 12
#define SensorPin 8
```



```
void setup()
{
    int sensorValue;
    int light;
    int remainder;

    sensorValue = analogRead(SensorPin) ;
    light = ((12 * sensorValue) - 5 ) / 2;
    remainder = 3 % 2;

}
void loop()
{
}
```

比較運算

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

當你在指定 if,while, for 敘述句時，可以運用下面這個運算符號：

符號	意義	範例
==	等於	a==1
!=	不等於	a!=1
<	小於	a<1
>	大於	a>1
<=	小於等於	a<=1
>=	大於等於	a>=1

布林運算

- && (and)
- || (or)
- ! (not)

當你想要結合多個條件式時，可以使用布林運算符號。

例如你想要檢查從感測器傳回的數值是否於 5 到 10，你可以這樣寫：

```
#define PWMPin 12
#define SensorPin 8
void setup()
{
}
void loop()
{
  int light;
  int sensor ;
  for (light = 0; light < 255; light++)
  {
    // 忽略數值介於 140 到 200 之間
    sensor = analogRead(SensorPin) ;

    if ((sensor >= 5) && (sensor <=10))
      continue;

    analogWrite(PWMPin, light);
    delay(10);
  }
}
```

這裡有三個運算符號：交集(and)用 **&&** 表示；聯集(or)用 **||** 表示；反相 (finally not)用 **!**表示。

複合運算符號：有一般特殊的運算符號可以使程式碼比較簡潔，例如累加運算符號。

例如將一個值加 1，你可以這樣寫：

```
Int value = 10 ;
value = value + 1 ;
```

你也可以用一個複合運算符號累加(++):

```
Int value = 10 ;
value ++;
```

複合運算符號

- ++ (increment)
- -- (decrement)
- += (compound addition)
- -= (compound subtraction)
- *= (compound multiplication)
- /= (compound division)

累加和遞減 (++ 和 --)

當你在累加 1 或遞減 1 到一個數值時。請小心 `i++` 和 `++i` 之間的不同。如果你用的是 `i++`，`i` 會被累加並且 `i` 的值等於 `i+1`；但當你使用 `++i` 時，`i` 的值等於 `i`，直到這行指令被執行完時 `i` 再加 1。同理應用於—。

`+=`, `-=`, `*=` and `/=`

這些運算符號可讓表示式更精簡，下面二個表示式是等價的：

```
Int value = 10 ;
value = value + 5 ;    // (此兩者都是等價)
value += 5 ;          // (此兩者都是等價)
```

輸入輸出腳位設定

數位訊號輸出/輸入

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

類比訊號輸出/輸入

- analogRead()
- analogWrite() - PWM

Arduino 內含了一些處理輸出與輸入的切換功能，相信已經從書中程式範例略知一二。

PinMode(Pin, mode)

將數位腳位(digital Pin)指定為輸入或輸出。

範例

```
#define sensorPin 7
#define PWNPin 8
void setup()
{
  pinMode(sensorPin,INPUT); // 將腳位 sensorPin (7) 定為輸入模式
}
void loop()
{
}
```

digitalWrite(Pin, value)

將數位腳位指定為開或關。腳位必須先透過 pinMode 明示為輸入或輸出模式 digitalWrite 才能生效。

範例：

```
#define PWNPIN 8
#define sensorPin 7
void setup()
{
  digitalWrite (PWNPIN,OUTPUT); // 將腳位 PWNPIN (8) 定為輸入模式
}
void loop()
{
}
```

int digitalRead(Pin)

將輸入腳位的值讀出，當感測到腳位處於高電位時回傳 HIGH，否則回傳 LOW。

範例：

```
#define PWNPIN 8
#define sensorPin 7
void setup()
{
  pinMode(sensorPin,INPUT); // 將腳位 sensorPin (7) 定為輸入模式
  val = digitalRead(7); // 讀出腳位 7 的值並指定給 val
}
void loop()
{
}
```

int analogRead(Pin)

讀出類比腳位的電壓並回傳一個 0 到 1023 的數值表示相對應的 0 到 5 的電壓值。

範例：

```
#define PWNPin 8
#define sensorPin 7
void setup()
{
  PinMode(sensorPin,INPUT); // 將腳位 sensorPin (7) 定為輸入模式
  val = analogRead (7); // 讀出腳位 7 的值並指定給 val
}
void loop()
{
}
}
```

analogWrite(Pin, value)

改變 PWM 腳位的輸出電壓值，腳位通常會在 3、5、6、9、10 與 11。value 變數範圍 0-255，例如：輸出電壓 2.5 伏特（V），該值大約是 128。

範例：

```
#define PWNPin 8
#define sensorPin 7
void setup()
{
  analogWrite (PWNPin,OUTPUT); // 將腳位 PWNPin (8) 定為輸入模式
}
void loop()
{ }
}
```

進階 I/O

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

tone(Pin)

使用 Arduino 開發板，使用一個 Digital Pin(數位接腳)連接喇叭，如本例子是接在數位接腳 13(Digital Pin 13)，讀者也可將喇叭接在您想要的腳位，只要將下列程式作對應修改，可以產生想要的音調。

範例：

```
#include <Tone.h>

Tone tone1;

void setup()
{
  tone1.begin(13);
  tone1.play(NOTE_A4);
}

void loop()
{
}
```

表 1 Tone 頻率表

常態變數	頻率(Frequency (Hz))
NOTE_B2	123
NOTE_C3	131
NOTE_CS3	139
NOTE_D3	147
NOTE_DS3	156
NOTE_E3	165
NOTE_F3	175
NOTE_FS3	185
NOTE_G3	196
NOTE_GS3	208
NOTE_A3	220
NOTE_AS3	233
NOTE_B3	247

常態變數	頻率(Frequency (Hz))
NOTE_C4	262
NOTE_CS4	277
NOTE_D4	294
NOTE_DS4	311
NOTE_E4	330
NOTE_F4	349
NOTE_FS4	370
NOTE_G4	392
NOTE_GS4	415
NOTE_A4	440
NOTE_AS4	466
NOTE_B4	494
NOTE_C5	523
NOTE_CS5	554
NOTE_D5	587
NOTE_DS5	622
NOTE_E5	659
NOTE_F5	698
NOTE_FS5	740
NOTE_G5	784
NOTE_GS5	831
NOTE_A5	880
NOTE_AS5	932
NOTE_B5	988
NOTE_C6	1047
NOTE_CS6	1109
NOTE_D6	1175
NOTE_DS6	1245
NOTE_E6	1319
NOTE_F6	1397
NOTE_FS6	1480
NOTE_G6	1568
NOTE_GS6	1661
NOTE_A6	1760
NOTE_AS6	1865

常態變數	頻率(Frequency (Hz))
NOTE_B6	1976
NOTE_C7	2093
NOTE_CS7	2217
NOTE_D7	2349
NOTE_DS7	2489
NOTE_E7	2637
NOTE_F7	2794
NOTE_FS7	2960
NOTE_G7	3136
NOTE_GS7	3322
NOTE_A7	3520
NOTE_AS7	3729
NOTE_B7	3951
NOTE_C8	4186
NOTE_CS8	4435
NOTE_D8	4699
NOTE_DS8	4978

資料來源：

https://code.google.com/p/rogue-code/wiki/ToneLibraryDocumentation#Ugly_Details

表 2 Tone 音階頻率對照表

音階	常態變數	頻率(Frequency (Hz))
低音 Do	NOTE_C4	262
低音 Re	NOTE_D4	294
低音 Mi	NOTE_E4	330
低音 Fa	NOTE_F4	349
低音 So	NOTE_G4	392
低音 La	NOTE_A4	440
低音 Si	NOTE_B4	494
中音 Do	NOTE_C5	523
中音 Re	NOTE_D5	587
中音 Mi	NOTE_E5	659
中音 Fa	NOTE_F5	698

音階	常態變數	頻率(Frequency (Hz))
中音 So	NOTE_G5	784
中音 La	NOTE_A5	880
中音 Si	NOTE_B5	988
高音 Do	NOTE_C6	1047
高音 Re	NOTE_D6	1175
高音 Mi	NOTE_E6	1319
高音 Fa	NOTE_F6	1397
高音 So	NOTE_G6	1568
高音 La	NOTE_A6	1760
高音 Si	NOTE_B6	1976
高高音 Do	NOTE_C7	2093

資料來源：

https://code.google.com/p/rogue-code/wiki/ToneLibraryDocumentation#Ugly_Details

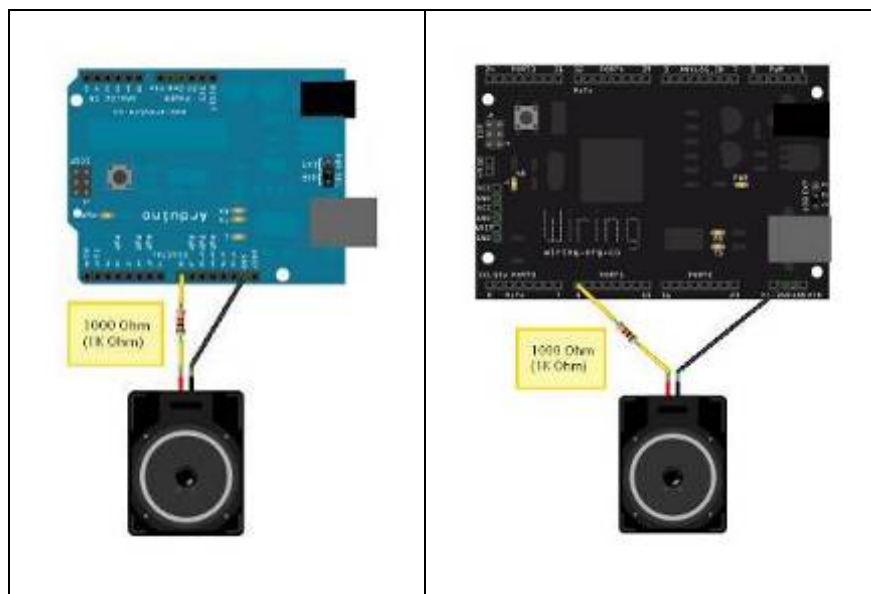


圖 198 Tone 接腳圖

資料來源：

https://code.google.com/p/rogue-code/wiki/ToneLibraryDocumentation#Ugly_Details

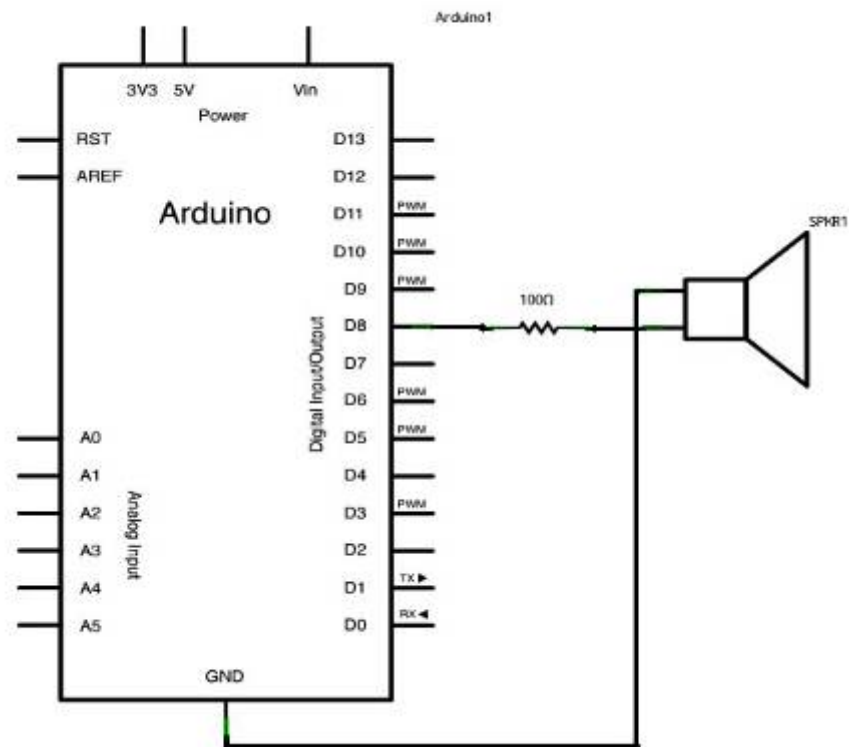


圖 199 Arduino 喇叭接線圖

Mario 音樂範例：

```
/*
  Arduino Mario Bros Tunes
  With Piezo Buzzer and PWM
  by: Dipto Pratyaksa
  last updated: 31/3/13
*/
#include <pitch.h>

#define melodyPin 3
//Mario main theme melody
int melody[] = {
  NOTE_E7, NOTE_E7, 0, NOTE_E7,
  0, NOTE_C7, NOTE_E7, 0,
  NOTE_G7, 0, 0, 0,
  NOTE_G6, 0, 0, 0,
```

```

NOTE_C7, 0, 0, NOTE_G6,
0, 0, NOTE_E6, 0,
0, NOTE_A6, 0, NOTE_B6,
0, NOTE_AS6, NOTE_A6, 0,

NOTE_G6, NOTE_E7, NOTE_G7,
NOTE_A7, 0, NOTE_F7, NOTE_G7,
0, NOTE_E7, 0, NOTE_C7,
NOTE_D7, NOTE_B6, 0, 0,

NOTE_C7, 0, 0, NOTE_G6,
0, 0, NOTE_E6, 0,
0, NOTE_A6, 0, NOTE_B6,
0, NOTE_AS6, NOTE_A6, 0,

NOTE_G6, NOTE_E7, NOTE_G7,
NOTE_A7, 0, NOTE_F7, NOTE_G7,
0, NOTE_E7, 0, NOTE_C7,
NOTE_D7, NOTE_B6, 0, 0
};
//Mario main them tempo
int tempo[] = {
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    9, 9, 9,
    12, 12, 12, 12,
    12, 12, 12, 12,
    12, 12, 12, 12,

    12, 12, 12, 12,

```

```

12, 12, 12, 12,
12, 12, 12, 12,
12, 12, 12, 12,

9, 9, 9,
12, 12, 12, 12,
12, 12, 12, 12,
12, 12, 12, 12,
};

//

//Underworld melody
int underworld_melody[] = {
    NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
    NOTE_AS3, NOTE_AS4, 0,
    0,
    NOTE_C4, NOTE_C5, NOTE_A3, NOTE_A4,
    NOTE_AS3, NOTE_AS4, 0,
    0,
    NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
    NOTE_DS3, NOTE_DS4, 0,
    0,
    NOTE_F3, NOTE_F4, NOTE_D3, NOTE_D4,
    NOTE_DS3, NOTE_DS4, 0,
    0, NOTE_DS4, NOTE_CS4, NOTE_D4,
    NOTE_CS4, NOTE_DS4,
    NOTE_DS4, NOTE_GS3,
    NOTE_G3, NOTE_CS4,
    NOTE_C4, NOTE_FS4, NOTE_F4, NOTE_E3, NOTE_AS4, NOTE_A4,
    NOTE_GS4, NOTE_DS4, NOTE_B3,
    NOTE_AS3, NOTE_A3, NOTE_GS3,
    0, 0, 0
};

//Underwolrd tempo
int underworld_tempo[] = {
    12, 12, 12, 12,
    12, 12, 6,

```

```

3,
12, 12, 12, 12,
12, 12, 6,
3,
12, 12, 12, 12,
12, 12, 6,
3,
12, 12, 12, 12,
12, 12, 6,
6, 18, 18, 18,
6, 6,
6, 6,
6, 6,
18, 18, 18, 18, 18, 18,
10, 10, 10,
10, 10, 10,
3, 3, 3
};

void setup(void)
{
    pinMode(3, OUTPUT); //buzzer
    pinMode(13, OUTPUT); //led indicator when singing a note
}

void loop()
{
    //sing the tunes
    sing(1);
    sing(1);
    sing(2);
}

int song = 0;

void sing(int s){
    // iterate over the notes of the melody:
    song = s;
    if(song==2){

```

```

Serial.println(" 'Underworld Theme'");
int size = sizeof(underworld_melody) / sizeof(int);
for (int thisNote = 0; thisNote < size; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/underworld_tempo[thisNote];

    buzz(melodyPin, underworld_melody[thisNote],noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);

    // stop the tone playing:
    buzz(melodyPin, 0,noteDuration);

}

}else{

Serial.println(" 'Mario Theme'");
int size = sizeof(melody) / sizeof(int);
for (int thisNote = 0; thisNote < size; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/tempo[thisNote];

    buzz(melodyPin, melody[thisNote],noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);

```

```

        // stop the tone playing:
        buzz(melodyPin, 0,noteDuration);

    }
}

void buzz(int targetPin, long frequency, long length) {
    digitalWrite(13,HIGH);
    long delayValue = 1000000/frequency/2; // calculate the delay value between
transitions
    /// 1 second's worth of microseconds, divided by the frequency, then split in half
since
    /// there are two phases to each cycle
    long numCycles = frequency * length/ 1000; // calculate the number of cycles for
proper timing
    /// multiply frequency, which is really cycles per second, by the number of sec-
onds to
    /// get the total number of cycles to produce
    for (long i=0; i < numCycles; i++){ // for the calculated length of time...
        digitalWrite(targetPin,HIGH); // write the buzzer Pin high to push out the dia-
phram
        delayMicroseconds(delayValue); // wait for the calculated delay value
        digitalWrite(targetPin,LOW); // write the buzzer Pin low to pull back the dia-
phram
        delayMicroseconds(delayValue); // wait again or the calculated delay value
    }
    digitalWrite(13,LOW);
}

/*****
 * Public Constants
 *****/

#define NOTE_B0  31
#define NOTE_C1  33

```



```
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
```

```
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
```

```
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
```

shiftOut(dataPin, clockPin, bitOrder, value)

把資料傳給用來延伸數位輸出的暫存器，函式使用一個腳位表示資料、一個腳位表示時脈。bitOrder 用來表示位元間移動的方式（LSBFIRST 最低有效位元或是 MSBFIRST 最高有效位元），最後 value 會以 byte 形式輸出。此函式通常使用在延伸數位的輸出。

範例：

```
#define dataPin 8
#define clockPin 7
void setup()
{
  shiftOut(dataPin, clockPin, LSBFIRST, 255);
}
void loop()
{ }
```

unsigned long pulseIn(Pin, value)

設定讀取腳位狀態的持續時間，例如使用紅外線、加速度感測器測得某一項數值時，在時間單位內不會改變狀態。

範例：

```
#define dataPin 8
#define pulsein 7
void setup()
{
  Int time ;
  time = pulsein(pulsein,HIGH); // 設定腳位 7 的狀態在時間單位內保持為 HIGH
}
void loop()
{  }
```

時間函式

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

控制與計算晶片執行期間的時間

unsigned long millis()

回傳晶片開始執行到目前的毫秒

範例:

```
int  lastTime ,duration;
void setup()
{
  lastTime = millis() ;
}
void loop()
{
  duration = -lastTime; // 表示自"lastTime"至當下的時間
}
```

delay(ms)

暫停晶片執行多少毫秒

範例:

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.print(millis());
  delay(500); //暫停半秒 (500 毫秒)
}
```

「毫」是 10 的負 3 次方的意思，所以「毫秒」就是 10 的負 3 次方秒，也就是 0.001 秒。

表 3 常用單位轉換表

符號	中文	英文	符號意義
p	微微	pico	10 的負 12 次方
n	奈	nano	10 的負 9 次方
u	微	micro	10 的負 6 次方
m	毫	milli	10 的負 3 次方
K	仟	kilo	10 的 3 次方
M	百萬	mega	10 的 6 次方
G	十億	giga	10 的 9 次方
T	兆	tera	10 的 12 次方

delay Microseconds(us)

暫停晶片執行多少微秒

範例:

```
void setup()
{
  Serial.begin(9600);
}
```

```

}
void loop()
{
  Serial.print(millis());
  delayMicroseconds (1000); //暫停半秒（500 毫秒）
}

```

數學函式

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()

三角函式以及基本的數學運算

min(x, y)

回傳兩數之間較小者

範例：

```

#define sensorPin1 7
#define sensorPin2 8
void setup()
{
  int val;
  pinMode(sensorPin1,INPUT); // 將腳位 sensorPin1 (7) 定為輸入模式
  pinMode(sensorPin2,INPUT); // 將腳位 sensorPin2 (8) 定為輸入模式
  val = min(analogRead (sensorPin1), analogRead (sensorPin2));
}
void loop()
{
}

```

max(x, y)

回傳兩數之間較大者

範例：

```
#define sensorPin1 7
#define sensorPin2 8
void setup()
{
  int val;
  pinMode(sensorPin1,INPUT); // 將腳位 sensorPin1 (7) 定為輸入模式
  pinMode(sensorPin2,INPUT); // 將腳位 sensorPin2 (8) 定為輸入模式
  val = max (analogRead (sensorPin1), analogRead (sensorPin2)) ;
}
void loop()
{ }
```

abs(x)

回傳該數的絕對值，可以將負數轉正數。

範例：

```
#define sensorPin1 7
void setup()
{
  int val;
  pinMode(sensorPin1,INPUT); // 將腳位 sensorPin (7) 定為輸入模式
  val = abs(analogRead (sensorPin1)-500);
  // 回傳讀值-500 的絕對值
}
void loop()
{ }
```

constrain(x, a, b)

判斷 x 變數位於 a 與 b 之間的狀態。x 若小於 a 回傳 a；介於 a 與 b 之間回傳 x 本身；大於 b 回傳 b

範例：

```
#define sensorPin1 7
#define sensorPin2 8
#define sensorPin 12
void setup()
{
  int val;
  pinMode(sensorPin1,INPUT); // 將腳位 sensorPin1 (7) 定為輸入模式
  pinMode(sensorPin2,INPUT); // 將腳位 sensorPin2 (8) 定為輸入模式
  pinMode(sensorPin,INPUT); // 將腳位 sensorPin (12) 定為輸入模式
  val = constrain(analogRead(sensorPin), analogRead (sensorPin1), analogRead
(sensorPin2));
  // 忽略大於 255 的數
}
void loop()
{
}
```

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

將 value 變數依照 fromLow 與 fromHigh 範圍，對等轉換至 toLow 與 toHigh 範圍。時常使用於讀取類比訊號，轉換至程式所需要的範圍值。

例如：

```
#define sensorPin1 7
#define sensorPin2 8
#define sensorPin 12
void setup()
{
  int val;
  pinMode(sensorPin1,INPUT); // 將腳位 sensorPin1 (7) 定為輸入模式
```



```
PinMode(sensorPin2,INPUT); // 將腳位 sensorPin2 (8) 定為輸入模式
PinMode(sensorPin,INPUT); // 將腳位 sensorPin (12) 定為輸入模式
val = map(analogRead(sensorPin), analogRead (sensorPin1), analogRead
(sensorPin2),0,100) ;
// 將 analog0 所讀取到的訊號對等轉換至 100 – 200 之間的數值
}
void loop()
{ }
```

double pow(base, exponent)

回傳一個數(base)的指數(exponent)值。

範例：

```
int y=2;
double x = pow(y, 32); // 設定 x 為 y 的 32 次方
```

double sqrt(x)

回傳 double 型態的取平方根值。

範例：

```
int y=2123;
double x = sqrt (y); // 回傳 2123 平方根的近似值
```

三角函式

- sin()
- cos()
- tan()

double sin(rad)

回傳角度 (radians) 的三角函式 sine 值。

範例：

```
int y=45;
double sine = sin (y); // 近似值 0.70710678118654
```

double cos(rad)

回傳角度（radians）的三角函式 cosine 值。

範例：

```
int y=45;
double cosine = cos (y); // 近似值 0.70710678118654
```

double tan(rad)

回傳角度（radians）的三角函式 tangent 值。

範例：

```
int y=45;
double tangent = tan (y); // 近似值 1
```

亂數函式

- randomSeed()
- random()

本函數是用來產生亂數用途：

randomSeed(seed)

事實上在 Arduino 裡的亂數是可以被預知的。所以如果需要一個真正的亂數，可以呼叫此函式重新設定產生亂數種子。你可以使用亂數當作亂數的種子，以確保數字以隨機的方式出現，通常會使用類比輸入當作亂數種子，藉此可以產生與環境有關的亂數。

範例：

```
#define sensorPin 7
void setup()
{
  randomSeed(analogRead(sensorPin)); // 使用類比輸入當作亂數種子
}
void loop()
{
}
```

long random(min, max)

回傳指定區間的亂數，型態為 **long**。如果沒有指定最小值，預設為 0。

範例：

```
#define sensorPin 7
long randNumber;
void setup(){
  Serial.begin(9600);
  // if analog input Pin sensorPin(7) is unconnected, random analog
  // noise will cause the call to randomSeed() to generate
  // different seed numbers each time the sketch runs.
  // randomSeed() will then shuffle the random function.
  randomSeed(analogRead(sensorPin));
}
void loop() {
  // print a random number from 0 to 299
  randNumber = random(300);
  Serial.println(randNumber);
}
```

```
// print a random number from 0 to 100
randNumber = random(0, 100); // 回傳 0 – 99 之間的數字
Serial.println(randNumber);
delay(50);
}
```

通訊函式

你可以在許多例子中，看見一些使用序列埠與電腦交換資訊的範例，以下是函式解釋。

Serial.begin(speed)

你可以指定 Arduino 從電腦交換資訊的速率，通常我們使用 9600 bps。當然也可以使用其他的速度，但是通常不會超過 115,200 bps（每秒位元組）。

範例：

```
void setup() {
  Serial.begin(9600);      // open the serial port at 9600 bps:
}
void loop() {
}
```

Serial.print(data)

Serial.print(data, 格式字串(encoding))

經序列埠傳送資料，提供編碼方式的選項。如果沒有指定，預設以一般文字傳送。

範例：

```
int x = 0;    // variable
```

```
void setup() {
  Serial.begin(9600);      // open the serial port at 9600 bps:
}

void loop() {
  // print labels
  Serial.print("NO FORMAT");    // prints a label
  Serial.print("\t");           // prints a tab
  Serial.print("DEC");
  Serial.print("\t");
  Serial.print("HEX");
  Serial.print("\t");
  Serial.print("OCT");
  Serial.print("\t");
  Serial.print("BIN");
  Serial.print("\t");
}
```

Serial.println(data)

Serial.println(data, ,格式字串(encoding))

與 Serial.print()相同，但會在資料尾端加上換行字元（`\n`）。意思如同你在鍵盤上打了一些資料後按下 Enter。

範例：

```
int x = 0;    // variable
void setup() {
  Serial.begin(9600);      // open the serial port at 9600 bps:
}

void loop() {
  // print labels
  Serial.print("NO FORMAT");    // prints a label
  Serial.print("\t");           // prints a tab
  Serial.print("DEC");
  Serial.print("\t");
```

```

Serial.print("HEX");
Serial.print("\t");
Serial.print("OCT");
Serial.print("\t");
Serial.print("BIN");
Serial.print("\t");

for(x=0; x< 64; x++){    // only part of the ASCII chart, change to suit
    // print it out in many formats:
    Serial.print(x);      // print as an ASCII-encoded decimal - same as "DEC"
    Serial.print("\t");   // prints a tab
    Serial.print(x, DEC); // print as an ASCII-encoded decimal
    Serial.print("\t");   // prints a tab
    Serial.print(x, HEX); // print as an ASCII-encoded hexadecimal
    Serial.print("\t");   // prints a tab
    Serial.print(x, OCT); // print as an ASCII-encoded octal
    Serial.print("\t");   // prints a tab
    Serial.println(x, BIN); // print as an ASCII-encoded binary
    // then adds the carriage return with "println"
    delay(200);           // delay 200 milliseconds
}
Serial.println("");      // prints another carriage return
}

```

格式字串(encoding)

Arduino 的 `print()` 和 `println()`，在列印內容時，可以指定列印內容使用哪一種格式列印，若不指定，則以原有內容列印。

列印格式如下：

1. BIN(二進位，或以 2 為基數)，
2. OCT(八進制，或以 8 為基數)，
3. DEC(十進位，或以 10 為基數)，
4. HEX(十六進位，或以 16 為基數)。

使用範例如下：

- Serial.print(78,BIN)輸出為“1001110”
- Serial.print(78,OCT)輸出為“116”
- Serial.print(78,DEC)輸出為“78”
- Serial.print(78,HEX)輸出為“4E”

對於浮點型數位，可以指定輸出的小數數位。例如

- Serial.println(1.23456,0)輸出為“1”
- Serial.println(1.23456,2)輸出為“1.23”
- Serial.println(1.23456,4)輸出為“1.2346”

Print & Println 列印格式(printformat01)

```
/*
使用 for 迴圈列印一個數字的各種格式。
*/
int x = 0;    // 定義一個變數並賦值

void setup() {
  Serial.begin(9600);    // 打開串口傳輸，並設置串列傳輸速率為 9600
}

void loop() {
  ///列印標籤
  Serial.print("NO FORMAT");    // 列印一個標籤
  Serial.print("\t");    // 列印一個轉義字元

  Serial.print("DEC");
  Serial.print("\t");
```

```

Serial.print("HEX");
Serial.print("\t");

Serial.print("OCT");
Serial.print("\t");

Serial.print("BIN");
Serial.print("\t");

for(x=0; x< 64; x++){    // 列印 ASCII 碼表的一部分, 修改它的格式得到需
                           要的內容

    // 列印多種格式：
    Serial.print(x);      // 以十進位格式將 x 列印輸出 - 與 "DEC"相同
    Serial.print("\t");   // 橫向跳格

    Serial.print(x, DEC); // 以十進位格式將 x 列印輸出
    Serial.print("\t");   // 橫向跳格

    Serial.print(x, HEX); // 以十六進位格式列印輸出
    Serial.print("\t");   // 橫向跳格

    Serial.print(x, OCT); // 以八進制格式列印輸出
    Serial.print("\t");   // 橫向跳格

    Serial.println(x, BIN); // 以二進位格式列印輸出
    //                      然後用 "println"列印一個回車
    delay(200);             // 延時 200ms
}
Serial.println("");        // 列印一個空字元，並自動換行
}

```

int Serial.available()

回傳有多少位元組（bytes）的資料尚未被 `read()` 函式讀取，如果回傳值是 0 代表所有序列埠上資料都已經被 `read()` 函式讀取。

範例：

```
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 receive data:
    if (Serial.available() > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();
        // say what you got:
        Serial.print("I received: ");
        Serial.println(incomingByte, DEC);
    }
}
```

int Serial.read()

以 byte 方式讀取 1byte 的序列資料

範例：

```
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 receive data:
    if (Serial.available() > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();
        // say what you got:
        Serial.print("I received: ");
        Serial.println(incomingByte, DEC);
    }
}
```

int Serial.write()

以 byte 方式寫入資料到序列

範例：

```
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.write(45); // send a byte with the value 45
  int bytesSent = Serial.write("hello Arduino , I am a beginner in the Arduino world");
}
```

Serial.flush()

有時候因為資料速度太快，超過程式處理資料的速度，你可以使用此函式清除緩衝區內的資料。經過此函式可以確保緩衝區(buffer)內的資料都是最新的。

範例：

```
void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.write(45); // send a byte with the value 45
  int bytesSent = Serial.write("hello Arduino , I am a beginner in the Arduino world");
  Serial.flush();
}
```

系統函式

Arduino 開發版也提供許多硬體相關的函式：

系統 idle 函式

使用硬體 idle 功能，可以讓 Arduino 進入睡眠狀態，連單晶片都可以進入睡眠狀態，但使用本功能需要使用外掛函式 Enerlib 函式庫，讀者可以到 Arduino 官網：<http://playground.arduino.cc/Code/Enerlib>，下載其函式庫安裝，或到本書範例檔：https://github.com/brucetsao/arduino_RFProgramming，下載相關函式與範例。

ATMega328 微控器具有六種睡眠模式，底下是依照「省電情況」排列的睡眠模式名稱，以及 Enerlib（註：Energy 和 Library，即：「能源」和「程式庫」的縮寫）程式庫的五道函數指令對照表，排越後面越省電。「消耗電流」欄位指的是 ATmega328 處理器本身，而非整個控制板(趙英傑, 2013, 2014)。

表 4 ATMega328 微控器六種睡眠模式

睡眠模式	Energy 指令	中文直譯	消耗電流
Idle	Idle()	閒置	15mA
ADC Noise Reduction	SleepADC()	類比數位轉換器降低雜訊	6.5mA
Power-save	PowerSave()	省電	1.62mA
Standby	Standby()	待機	1.62mA
Extended Standby		延長待機	0.84mA
Power-down	PowerDown()	斷電	0.36mA

微控器內部除了中央處理器（CPU），還有記憶體、類比數位轉換器、序列通訊…等模組。越省電的模式，仍在運作中的模組就越少。

例如，在” Power-Down”（電源關閉）睡眠模式之下，微控器僅剩下外部中斷和看門狗計時器（**Watchdog Timer**）仍持續運作。而在 Idle 睡眠模式底下，SPI, UART（也就是序列埠）、計時器、類比數位轉換器等，仍持續運作，只有中央處理器和快閃記憶體（Flash）時脈訊號被停止。

時脈訊號就像心跳一樣，一旦停止時脈訊號，相關的元件也隨之暫停。各種睡眠模式的詳細說明，請參閱下表。

表 5 Active Clock Domains and Wake-up Sources in the Different Sleep Modes

Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

Sleep Mode	Active Clock Domains					Oscillators		Wake-up Sources							Software BOD Disable
	clk _{CPU}	clk _{FLASH}	clk _{IO}	clk _{ADC}	clk _{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	
Idle			X	X	X	X	X ⁽²⁾	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X ⁽²⁾	X ⁽³⁾	X	X ⁽²⁾	X	X	X		
Power-down								X ⁽³⁾	X				X		X
Power-save					X		X ⁽²⁾	X ⁽³⁾	X	X			X		X
Standby ⁽¹⁾						X		X ⁽³⁾	X				X		X
Extended Standby					X ⁽²⁾	X	X ⁽²⁾	X ⁽³⁾	X	X			X		X

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
 2. If Timer/Counter2 is running in asynchronous mode.
 3. For INT1 and INT0, only level interrupt.

資料來源：ATmega328 微控器的資料手冊，第 39 頁，「Power Management and Sleep Modes (電源管理與睡眠模式)」單元

(<http://www.atmel.com/images/doc8161.pdf>)

範例：

Arduin 進入睡眠狀態範例(

/*

Enerlib: easy-to-use wrapper for AVR's Sleep library.

By E.P.G. - 11/2010 - Ver. 1.0.0

Example showing how to enter in Idle mode and exit from it with INT0.

```

*/

#include <Enerlib.h>

Energy energy;

void INT0_ISR(void)
{

    /*
    The WasSleeping function will return true if Arduino
    was sleeping before the IRQ. Subsequent calls to
    WasSleeping will return false until Arduino reenters
    in a low power state. The WasSleeping function should
    only be called in the ISR.
    */
    if (energy.WasSleeping())
    {
        /*
        Arduino was waked up by IRQ.

        If you shut down external peripherals before sleeping, you
        can reinitialize them here. Look on ATmega's datasheet for
        hardware limitations in the ISR when microcontroller just
        leave any low power state.
        */
    }
    else
    {
        /*
        The IRQ happened in awake state.

        This code is for the "normal" ISR.
        */
    }
}

```

```

}

void setup()
{
    Serial.begin(9600);
    Serial.println("Program Start") ;
    attachInterrupt(0, INT0_ISR, LOW);
    /*
    Pin 2 will be the "wake button". Due to uC limitations,
    it needs to be a level interrupt.
    For experienced programmers:
    ATmega's datasheet contains information about the rest of
    wake up sources. The Extended Standby is not implemented.
    */

    Serial.println("Now I am Sleeping") ;
    delay(500);
    energy.Idle();
}

void loop()
{

    Serial.println("I am waken ") ;
    delay(1000);
}

```

attachInterrupt(插斷)

當開發者撰寫程式時，在 `loop()` 程式段之中，撰寫許多大量的程式碼，並且重複的執行，當我們需要在某些時後去檢查某一樣硬體，如按鈕、讀卡機、RFID、鍵盤、滑鼠等周邊裝置，若這些檢查、讀取該周邊的函式，寫在 `loop()` 程式段之中，則必需每一個迴圈都必需耗時去檢查，不但造成程式不順暢，還會錯失讀取這些檢查、讀取該周邊的函式的時機。

這時後我們就需要用到這些 Arduino 開發板外部插斷接腳，由於 Arduino 開發板使用外部插斷接腳，不同開發板其接腳都不太相同，我們可以參考表 6 之 Arduino 開發板外部插斷接腳對照表。

表 6 Arduino 開發板外部插斷接腳對照表

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	

attachInterrupt(第幾號外部插斷, 執行之函式名稱, LOW/HIGH);

參數解說：

- 第一個參數：使用那一個外部插斷，其接腳請參考參考表 6 之 Arduino 開發板外部插斷接腳對照表。
- 第二個參數：為執行之函式名稱；在 Arduino 程式區自行定義一個若使用插斷後，執行的函式名稱
- 第三個參數：驅動外部硬體插斷所使用的電位， HIGH 表高電位，LOW 表低電位

範例：

```

外部插斷測試程式(IRQTest)
void setup() {
  // put your setup code here, to run once:

  Serial.begin(9600);
  Serial.println("Program Start");
  attachInterrupt(0, TheButtonPressed, LOW);
}

```

```
void loop() {  
    // put your main code here, to run repeatedly:  
    Serial.print("now program run in loop()");  
}  
  
void TheButtonPressed()  
{  
    Serial.println("The Button is pressed by user");  
}
```

章節小結

本章節概略的介紹 Arduino 程式撰寫的語法、函式等介紹，接下來就是介紹本書主要的內容，讓我們視目以待。

4

CHAPTER

自製使用者函式庫

如果讀者有讀過筆者在 T 客邦的一篇文章：『Maker 物聯網實作：用 DHx 溫濕度感測模組回傳天氣溫溼度』，網址為：

<http://www.techbang.com/posts/26208-the-internet-of-things-daily-life-how-to-know-the-temperature-and-humidity>(曹永忠, 許智誠, & 蔡英德, 2015e)，提到 DHT11 的感測元件，一般來說，Maker 找到這個元件，只要去 Google 或 Github 搜尋一下，很快就找到 DHT11 的感測模組對應函式庫，但是隨著越來越多開發板的種類，筆者拙作：

86Duino 程式教學(網路通訊篇):86duino Programming (Networking Communication)(曹永忠, 許智誠, & 蔡英德, 2015a, 2015b)中，使用 DHT11 的感測元件，開始發現，除非使用者永遠使用 Arduino 開發板，或等待他人將感測元件對應的函式庫寫出來並分享給大家使用，否則一離開 Arduino 開發領域，馬上就會遇到您熟悉的感測元件似乎都離我們遠去，不在是那麼易學易用了(曹永忠, 許智誠, & 蔡英德, 2015f)。

自己看規格書攥寫函式庫

如果您是 C 語言高手，也是軟體程式設計的高手，那這篇文章您應該不用繼續讀下去了，但是要自己看感測器的規格書(DataSheet)來攥寫函式庫，那可能這篇文章會很難以閱讀，所以筆者將 86Duino 程式教學(網路通訊篇):86duino Programming (Networking Communication)(曹永忠 et al., 2015a, 2015b, 2015f)的改寫可用的函式庫的內容，也分享在本書當中，相信許多讀者未來遇到這樣的問題，可以得到一些協助。

首先，我們採用下圖所示之 DHT11 溫濕度感測器，並且使用 86Duino 系列的

EDUCAKE 開發板開發相關程式。



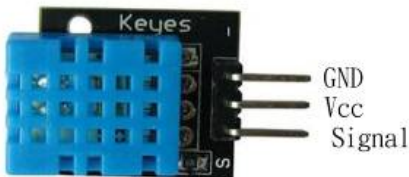
圖 200 DHT11 溫濕度感測器

使用 Arduino 開發板範例

如上圖所示，先參考溫濕度感測模組(DHT11)腳位接法，在遵照下表之溫濕度感測模組(DHT11)接腳表進行電路組裝，完成如下圖所示之插好 DHT11 的 EDUCAKE。

表 7 溫濕度感測模組(DHT11)接腳表

接腳	接腳說明	Arduino 開發板接腳
S	Vcc	電源 (+5V) Arduino +5V
2	GND	Arduino GND
3	Signal	Arduino digital pin 7



資料來源：Arduino 程式教學(常用模組篇):Arduino Programming (37 Modules)(曹永忠, 許智誠, & 蔡英德, 2015c, 2015d)

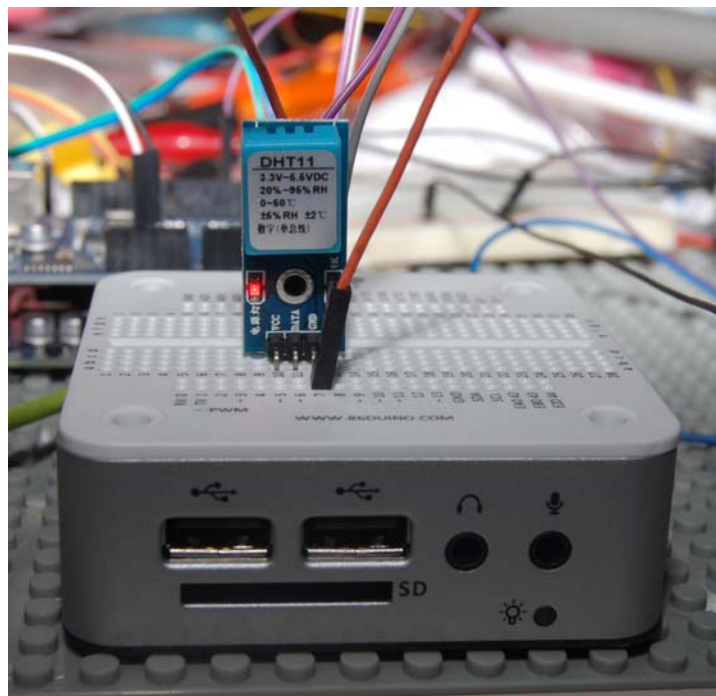


圖 201 插好 DHT11 的 EDUCAKE

我們打開 Arduino 開發板的開發工具：Sketch IDE 整合開發軟體，
編寫一段程式，如下表所示之 DHT11 溫濕度感測模組測試程式，我們
就可以透過溫濕度感測模組(DHT11)來偵測任何溫度與濕度。

表 8 DHT11 溫濕度感測模組測試程式

DHT11 溫濕度感測模組測試程式(DHT_Ok_for_Arduino)
<pre> int DHpin=7; byte dat[5]; byte read_data() { byte data; for(int i=0; i<8;i++) { </pre>

```

        if(digitalRead(DHpin)==LOW)
        {

            while(digitalRead(DHpin)==LOW);                //
            等待 50us

            delayMicroseconds(30);
            //判斷高電位的持續時間，以判定數據是 '0' 還是 '1'

            if(digitalRead(DHpin)==HIGH)
                data |= (1<<(7-i));

            //高位在前，低位在後

            while(digitalRead(DHpin) == HIGH);                //數
            據 '1' ，等待下一位的接收
        }
    }
    return data;
}

void start_test()
{
    digitalWrite(DHpin,LOW);                //拉低總線，發
    開始信號
    delay(30);
    //延遲時間要大於 18ms，以便檢測器能檢測到開始訊號；
    digitalWrite(DHpin,HIGH);
    delayMicroseconds(40);                //等待感測器
    響應；
    pinMode(DHpin,INPUT);
    while(digitalRead(DHpin) == HIGH);
    delayMicroseconds(80);                //發出響應，
    拉低总线 80us；
    if(digitalRead(DHpin) == LOW);
    delayMicroseconds(80);                //線路
    80us 後開始發送數據；

    for(int i=0;i<4;i++)                //接收溫
    溼度數據，校驗位不考慮；

```

```

        dat[i] = read_data();

        pinMode(DHpin,OUTPUT);
        digitalWrite(DHpin,HIGH);           //發送
完數據後釋放線路，等待下一次的開始訊號；
    }

    void setup()
    {
        Serial.begin(9600);
        pinMode(DHpin,OUTPUT);
    }

    void loop()
    {
        start_test();
        Serial.print("Current humidity = ");
        Serial.print(dat[0], DEC);           //顯示濕
度的整數位；
        Serial.print('.');
        Serial.print(dat[1],DEC);           //顯示
濕度的小數位；
        Serial.println('%');
        Serial.print("Current temperature = ");
        Serial.print(dat[2], DEC);           //顯示溫
度的整數位；
        Serial.print('.');
        Serial.print(dat[3],DEC);           //顯示溫
度的小數位；
        Serial.println('C');
        delay(700);
    }

```

資料來源：Arduino 程式教學(常用模組篇):Arduino Programming (37 Modules)(曹永忠 et al., 2015c, 2015d, 2015f)

原始碼下載網址：<https://github.com/brucetsao/techbang/tree/master/201510>

如下圖所示，我們可以看到溫濕度感測模組測試程式結果畫面，在 Arduino 開發板上，可以非常正常使用 DHT11 溫濕度感測模組與對應的函式庫與範例。

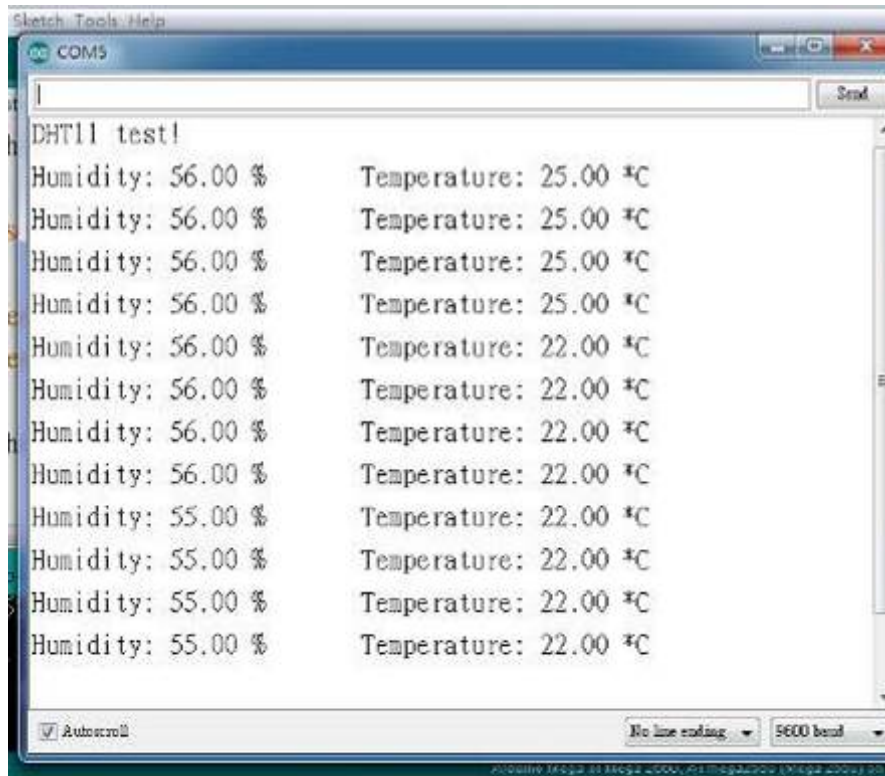


圖 202 DHT11 溫濕度感測模組測試程式結果畫面

但是，如果把程式原封不動的移到 86Duino 的開發工具，並把該 dht 的函式庫也安裝到 86Duino 的開發工具，則會發現函式庫不相容，比較圖如下圖所示，我們發現，無痛轉移開發板並不是這樣的簡單，因為這樣簡單的範例程式，卻連編譯都失敗，更不用說能夠成功運行了。

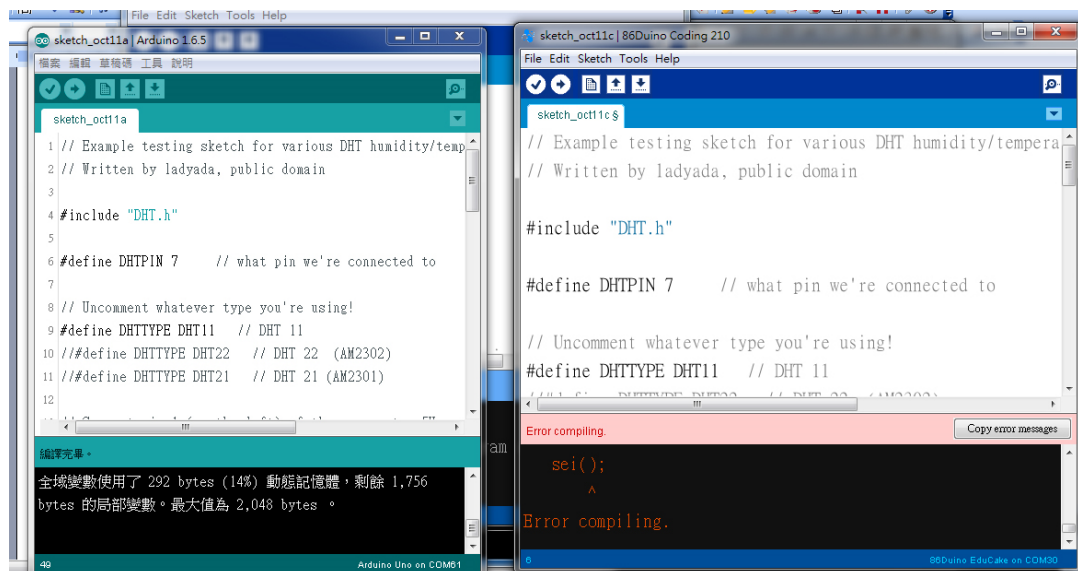


圖 203 無法正常編譯的 DHT11 範例

尋求原廠協助

為了能夠把 DHT11 溫濕度感測模組安裝在 86Duino 的 EDUCAKE 開發板上，並能夠正常運作，筆者找上了 86Duino 的開發者：瞻營全電子股份有限公司。

86Duino 是國內『瞻營全電子股份有限公司』開發出來相容於 Arduino 開發板的傑作，其相容度更是少人能比，相信 DHT11 溫濕度感測模組這樣基礎的模組，一定是可以完全支援的，果然在瞻營全電子股份有限公司的商品官網：<http://shop.dmp.com.tw/>，馬上找到 DHT11 溫濕度感測模組的產品支援，網址如 <http://shop.dmp.com.tw/INT/products/67>，在其官網找到『dht11_test.ino』，筆者跟據本實驗的接腳，改寫如下表所示之 DHT11 溫濕度感測模組測試程式，我們就可以透過 DHT11 溫濕度感測模組來偵測任何溫度與濕度。

表 9 DHT11 溫濕度感測模組測試程式

DHT11 溫 濕 度 感 測 模 組 測 試 程 式 (DHT_OK_For_86duino_Product)	
//	
//	FILE: dht11_test.ino


```
// AUTHOR: Rob Tillaart
// VERSION: 0.1.00
// PURPOSE: DHT library test sketch for DHT11 && Arduino
// URL:
//
// Released to the public domain
//

#define DHT_LIB_VERSION "0.1.10"

#define DHTLIB_OK                0
#define DHTLIB_ERROR_CHECKSUM   -1
#define DHTLIB_ERROR_TIMEOUT    -2
#define DHTLIB_INVALID_VALUE    -999

#define DHTLIB_DHT11_WAKEUP     18
#define DHTLIB_DHT22_WAKEUP     1
#define TIMEOUT (F_CPU/40000)

class dht
{
public:
    int read11(uint8_t pin);
    int read21(uint8_t pin);
    int read22(uint8_t pin);

    double humidity;
    double temperature;

private:
    uint8_t bits[5]; // buffer to receive data
    int read(uint8_t pin, uint8_t wakeupDelay);
};

dht DHT;

#define DHT11_PIN 7
```

```

void setup()
{
  Serial.begin(9600);
  Serial.println("DHT TEST PROGRAM ");
  Serial.print("LIBRARY VERSION: ");
  Serial.println(DHT_LIB_VERSION);
  Serial.println();
  Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
}

void loop()
{
  // READ DATA
  Serial.print("DHT11, \t");
  int chk = DHT.read11(DHT11_PIN);
  switch (chk)
  {
    case DHTLIB_OK:
      Serial.print("OK,\t");
      break;
    case DHTLIB_ERROR_CHECKSUM:
      Serial.print("Checksum error,\t");
      break;
    case DHTLIB_ERROR_TIMEOUT:
      Serial.print("Time out error,\t");
      break;
    default:
      Serial.print("Unknown error,\t");
      break;
  }
  // DISPLAY DATA
  Serial.print(DHT.humidity, 1);
  Serial.print(",\t");
  Serial.println(DHT.temperature, 1);

  delay(2000);
}

int dht::read11(uint8_t pin)

```

```

{
    // READ VALUES
    int rv = read(pin, DHTLIB_DHT11_WAKEUP);
    if (rv != DHTLIB_OK)
    {
        humidity      = DHTLIB_INVALID_VALUE; // invalid value, or is
NaN preferred?
        temperature = DHTLIB_INVALID_VALUE; // invalid value
        return rv;
    }

    // CONVERT AND STORE
    humidity      = bits[0]; // bits[1] == 0;
    temperature = bits[2]; // bits[3] == 0;

    // TEST CHECKSUM
    // bits[1] && bits[3] both 0
    uint8_t sum = bits[0] + bits[2];
    if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;

    return DHTLIB_OK;
}

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht::read21(uint8_t pin)
{
    // dataformat & wakeup identical to DHT22
    return read22(pin);
}

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht::read22(uint8_t pin)

```

```

{
    // READ VALUES
    int rv = read(pin, DHTLIB_DHT22_WAKEUP);
    if (rv != DHTLIB_OK)
    {
        humidity      = DHTLIB_INVALID_VALUE; // invalid value, or is
NaN preferred?
        temperature = DHTLIB_INVALID_VALUE; // invalid value
        return rv; // propagate error value
    }

    // CONVERT AND STORE
    humidity = word(bits[0], bits[1]) * 0.1;

    if (bits[2] & 0x80) // negative temperature
    {
        temperature = -0.1 * word(bits[2] & 0x7F, bits[3]);
    }
    else
    {
        temperature = 0.1 * word(bits[2], bits[3]);
    }

    // TEST CHECKSUM
    uint8_t sum = bits[0] + bits[1] + bits[2] + bits[3];
    if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;

    return DHTLIB_OK;
}

////////////////////////////////////
//
// PRIVATE
//

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_TIMEOUT

```

```
int dht::read(uint8_t pin, uint8_t wakeupDelay)
{
    // INIT BUFFERVAR TO RECEIVE DATA
    uint8_t mask = 128;
    uint8_t idx = 0;

    // EMPTY BUFFER
    for (uint8_t i = 0; i < 5; i++) bits[i] = 0;

    // REQUEST SAMPLE
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
    delay(wakeupDelay);
    digitalWrite(pin, HIGH);
    delayMicroseconds(40);
    pinMode(pin, INPUT);

    // GET ACKNOWLEDGE or TIMEOUT
    unsigned int loopCnt = TIMEOUT;
    while(digitalRead(pin) == LOW)
    {
        if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
    }

    loopCnt = TIMEOUT;
    while(digitalRead(pin) == HIGH)
    {
        if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
    }

    // READ THE OUTPUT - 40 BITS => 5 BYTES
    for (uint8_t i = 0; i < 40; i++)
    {
        loopCnt = TIMEOUT;
        while(digitalRead(pin) == LOW)
        {
            if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
        }
    }
}
```

```

        unsigned long t = micros();

        loopCnt = TIMEOUT;
        while(digitalRead(pin) == HIGH)
        {
            if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
        }

        if ((micros() - t) > 40) bits[idx] |= mask;
        mask >>= 1;
        if (mask == 0)    // next byte?
        {
            mask = 128;
            idx++;
        }
    }
    return DHTLIB_OK;
}
//
// END OF FILE
//

```

資料來源：瞻營全電子股份有限公司(<http://shop.dmp.com.tw/INT/products/67>)

原始碼下載網址：<https://github.com/brucetsao/techbang/tree/master/201510>

果然，原廠的服務無卸可擊，如下圖所示，我們可以看到溫濕度感測模組測試程式結果畫面。

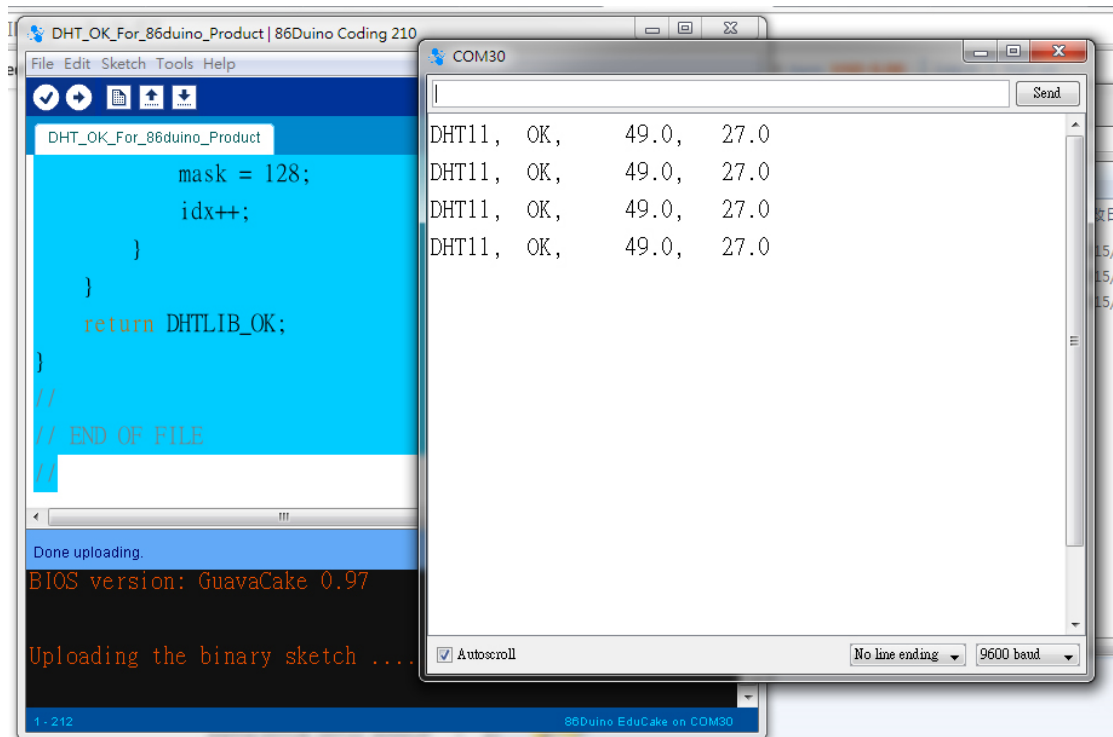


圖 204 官網可正常執行之 DHT11 溫濕度感測模組測試程式結果畫面

改寫成函式庫的語法

但是，上述的程式，在使用 DHT11 溫濕度感測模組時，並不像以往我們熟悉的方式，把 DHT11 溫濕度感測模組的使用，封裝成一個專有的函式庫提供給使用者使用，而是將 DHT11 溫濕度感測模組使用的類別、方法等都與主程式混在同一隻程式，這樣不但使用上很容易誤解，未來改寫功能時，更有極大個可能將 DHT11 溫濕度感測模組使用的類別、方法給與破壞等問題。

但是，筆者看看上面瞻營全電子股份有限公司的 DHT11 溫濕度感測模組的範例程式，對於 DHT11 溫濕度感測模組所建構的類別、方法非常清楚，並且易懂，所以希望可以將這隻程式『DHT_Bad_for_86duino』，成功轉成封裝的函式庫。

首先，寫過 Arduino 函式庫的人，都知道，一般函式庫都有兩隻程式，一隻是.h 檔，就是一般的 include 檔，另一個就是 C++ 程式檔，一般為 CPP 檔。

一般說來，.h 檔，就是一般的 include 檔都是定義大部份的巨集，或 define 一些變數，主要.h 檔用來將主體的類別設計出來，但是只有將方法(Method)與屬性(Property)用 Interface 的方式先行定義出來，所以我們很快的就把原有的程式內，所有的『#define』與『class』等分離出來，如下表所式，寫在 dht.h 的 include 檔上。

表 10 改寫為 Include 檔

改寫為 Include 檔(dht.h)
<pre>#define DHT_LIB_VERSION "0.1.10" #define DHTLIB_OK 0 #define DHTLIB_ERROR_CHECKSUM -1 #define DHTLIB_ERROR_TIMEOUT -2 #define DHTLIB_INVALID_VALUE -999 #define DHTLIB_DHT11_WAKEUP 18 #define DHTLIB_DHT22_WAKEUP 1 #define TIMEOUT (F_CPU/40000) class dht { public: int read11(uint8_t pin); int read21(uint8_t pin); int read22(uint8_t pin); double humidity; double temperature; private: uint8_t bits[5]; // buffer to receive data int read(uint8_t pin, uint8_t wakeupDelay); };</pre>

資料來源：86Duino 程式教學(網路通訊篇):86duino Programming (Networking Communication)(曹

原始碼下載網址：<https://github.com/brucetsao/techbang/tree/master/201510>

如下圖所示，我們發現編譯有問題。

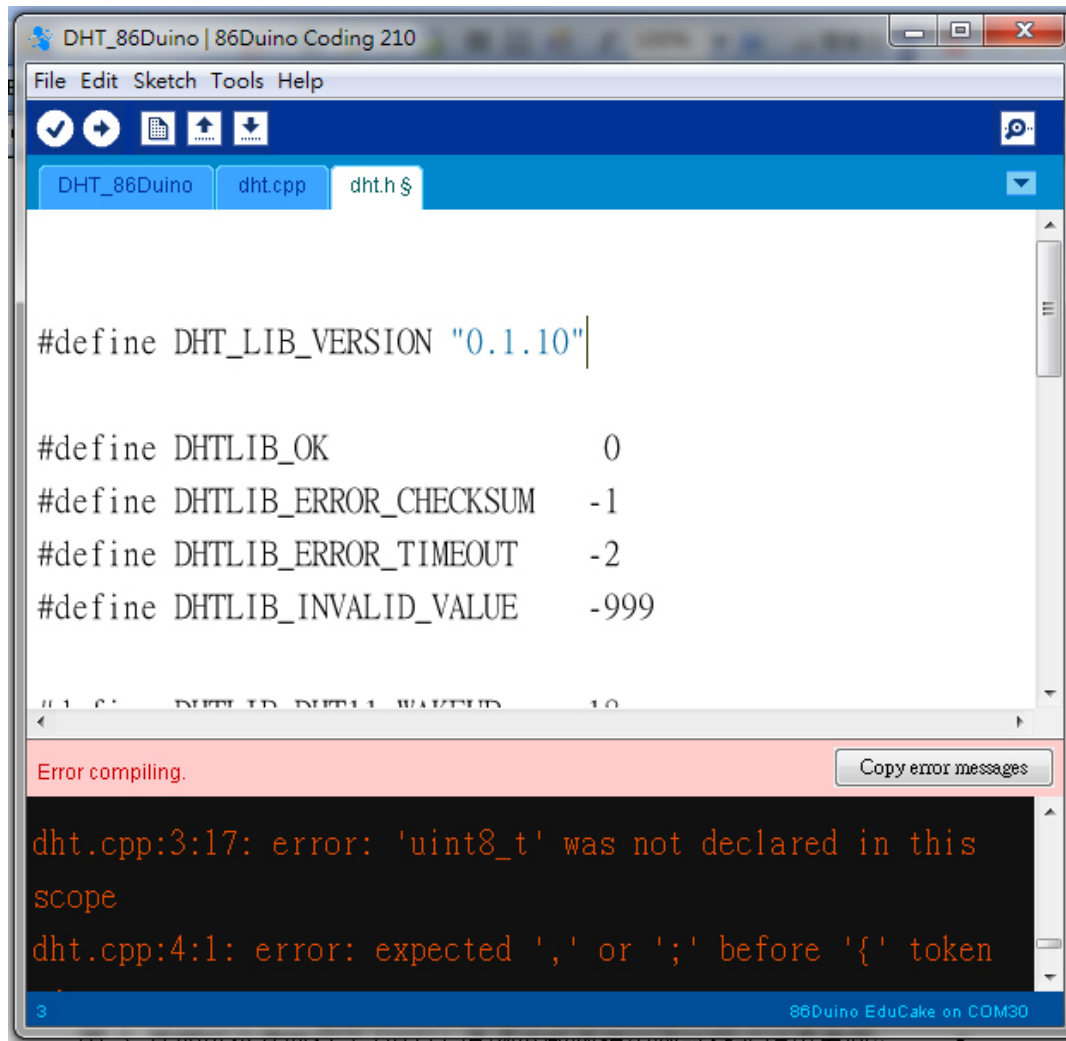


圖 205 無法編譯的 include 檔

但是仔細看一看，如上圖所示，我們發現編譯的許多問題都是型態有問題，我們發現，原來 `class` 寫在 `arduino` 的 `ino` 檔內，許多型態的宣告是自動，但是一但將這些程式獨立到外部的 C++ 程式，這些 `include` 檔的宣告，就不會自動幫我們加上，所以筆者將這些常用的型態宣告找出來後，發現大部份的宣告都在『`Arduino.h`』之中，而 `Arduino.h` 又會用到 `stdint.h`，所以如下表所示，我將這兩行 `include` 程式

加到 dht.h 檔之中。

表 11 常用的型態宣告 include 檔

<pre>#include <Arduino.h> #include <stdint.h></pre>

如下表所示，我將這上表這兩行 include 程式加到 dht.h 檔之中，並且開始編譯。

表 12 改寫為 Include 檔

擴增 Arduino 型態 include 檔之 include (dht.h)
<pre>#include <Arduino.h> #include <stdint.h> #define DHT_LIB_VERSION "0.1.10" #define DHTLIB_OK 0 #define DHTLIB_ERROR_CHECKSUM -1 #define DHTLIB_ERROR_TIMEOUT -2 #define DHTLIB_INVALID_VALUE -999 #define DHTLIB_DHT11_WAKEUP 18 #define DHTLIB_DHT22_WAKEUP 1 #define TIMEOUT (F_CPU/40000) class dht { public: int read11(uint8_t pin); int read21(uint8_t pin); int read22(uint8_t pin); double humidity; double temperature; private:</pre>

```
uint8_t bits[5]; // buffer to receive data
int read(uint8_t pin, uint8_t wakeupDelay);

};
```

資料來源：86Duino 程式教學(網路通訊篇):86duino Programming (Networking Communication)(曹

永忠 et al., 2015a, 2015b)

原始碼下載網址：<https://github.com/brucetsao/techbang/tree/master/201510>

如下圖所示，我們可以看到 dht.h 可以正常編譯成功。

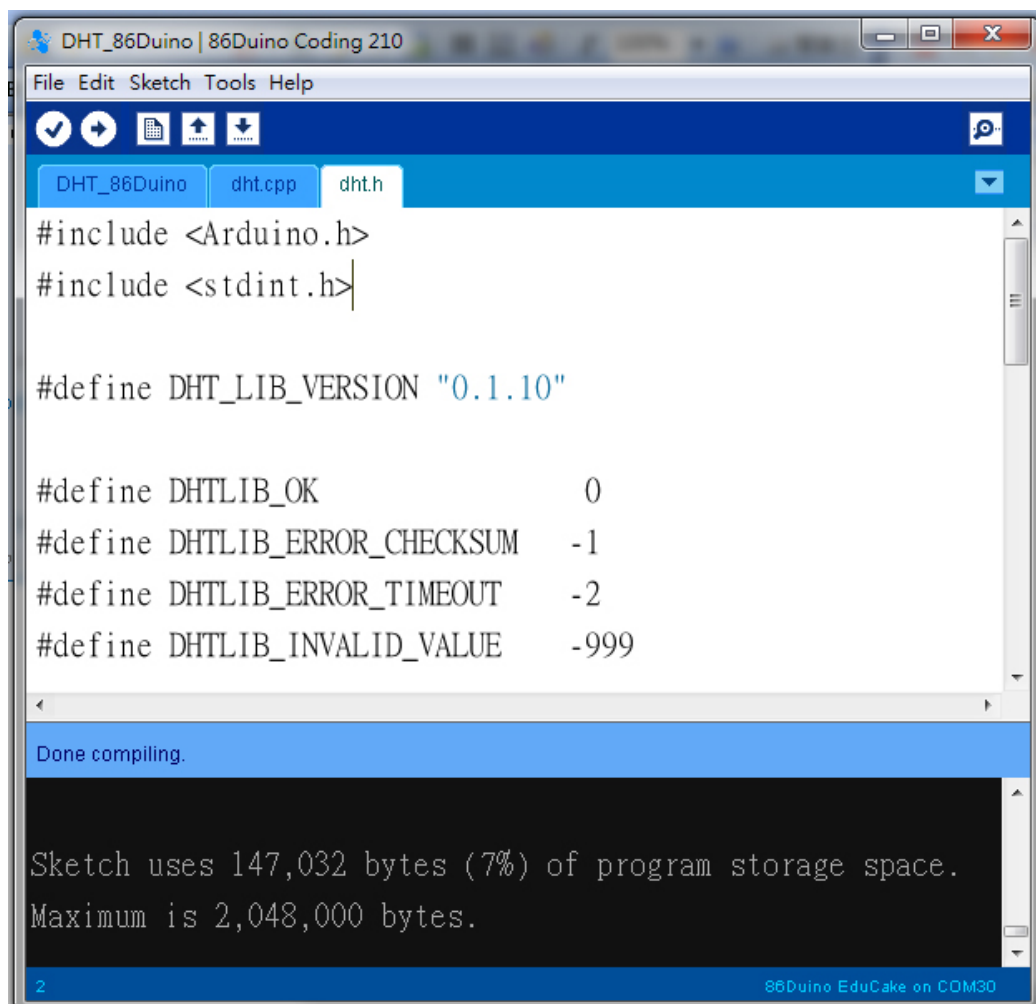


圖 206 可以編譯的 include 檔

將函式庫元件實體寫出的語法

寫過 Arduino 函式庫的人，都知道 C++ 程式檔，一般為 CPP 檔，最主要的用處就是將主體的類別給與實作出來，如下表所示，所以我們很快的就把原有的程式內，所有的實作的方法(Method)，將那些程式碼移到 dht.cpp 檔上，其內容主體大部份是有『dht::』等相關的內容。

表 13 改寫為 CPP 檔

改寫為 CPP 檔(dht.cpp)
<pre> int dht::read11(uint8_t pin) { // READ VALUES int rv = read(pin, DHTLIB_DHT11_WAKEUP); if (rv != DHTLIB_OK) { humidity = DHTLIB_INVALID_VALUE; // invalid value, or is NaN prefered? temperature = DHTLIB_INVALID_VALUE; // invalid value return rv; } // CONVERT AND STORE humidity = bits[0]; // bits[1] == 0; temperature = bits[2]; // bits[3] == 0; // -*olw[23TEST CHECKSUM // bits[1] && bits[3] both 0 uint8_t sum = bits[0] + bits[2]; if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM; return DHTLIB_OK; } // return values: </pre>

```

// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht::read21(uint8_t pin)
{
    // dataformat & wakeup identical to DHT22
    return read22(pin);
}

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht::read22(uint8_t pin)
{
    // READ VALUES
    int rv = read(pin, DHTLIB_DHT22_WAKEUP);
    if (rv != DHTLIB_OK)
    {
        humidity      = DHTLIB_INVALID_VALUE; // invalid value, or is
NaN preferred?
        temperature = DHTLIB_INVALID_VALUE; // invalid value
        return rv; // propagate error value
    }

    // CONVERT AND STORE
    humidity = word(bits[0], bits[1]) * 0.1;

    if (bits[2] & 0x80) // negative temperature
    {
        temperature = -0.1 * word(bits[2] & 0x7F, bits[3]);
    }
    else
    {
        temperature = 0.1 * word(bits[2], bits[3]);
    }

    // TEST CHECKSUM

```

```

uint8_t sum = bits[0] + bits[1] + bits[2] + bits[3];
if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;

return DHTLIB_OK;
}

////////////////////////////////////
//
// PRIVATE
//

// return values:
// DHTLIB_OK
// DHTLIB_ERROR_TIMEOUT
int dht::read(uint8_t pin, uint8_t wakeupDelay)
{
    // INIT BUFFERVAR TO RECEIVE DATA
    uint8_t mask = 128;
    uint8_t idx = 0;

    // EMPTY BUFFER
    for (uint8_t i = 0; i < 5; i++) bits[i] = 0;

    // REQUEST SAMPLE
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
    delay(wakeupDelay);
    digitalWrite(pin, HIGH);
    delayMicroseconds(40);
    pinMode(pin, INPUT);

    // GET ACKNOWLEDGE or TIMEOUT
    unsigned int loopCnt = TIMEOUT;
    while(digitalRead(pin) == LOW)
    {
        if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
    }
}

```

```

loopCnt = TIMEOUT;
while(digitalRead(pin) == HIGH)
{
    if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
}

// READ THE OUTPUT - 40 BITS => 5 BYTES
for (uint8_t i = 0; i < 40; i++)
{
    loopCnt = TIMEOUT;
    while(digitalRead(pin) == LOW)
    {
        if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
    }

    unsigned long t = micros();

    loopCnt = TIMEOUT;
    while(digitalRead(pin) == HIGH)
    {
        if (--loopCnt == 0) return DHTLIB_ERROR_TIMEOUT;
    }

    if ((micros() - t) > 40) bits[idx] |= mask;
    mask >>= 1;
    if (mask == 0)    // next byte?
    {
        mask = 128;
        idx++;
    }
}
return DHTLIB_OK;
}
//
// END OF FILE

```

資料來源：86Duino 程式教學(網路通訊篇):86duino Programming (Networking Communication)(曹

永忠 et al., 2015a, 2015b, 2015f)

原始碼下載網址：<https://github.com/brucetsao/techbang/tree/master/201510>

如下圖所示，我們可以看到溫濕度感測模組測試程式結果畫面，我們發現，主要的錯誤訊息：***dht.cpp:2:5: error: 'dht' has not been declared***。

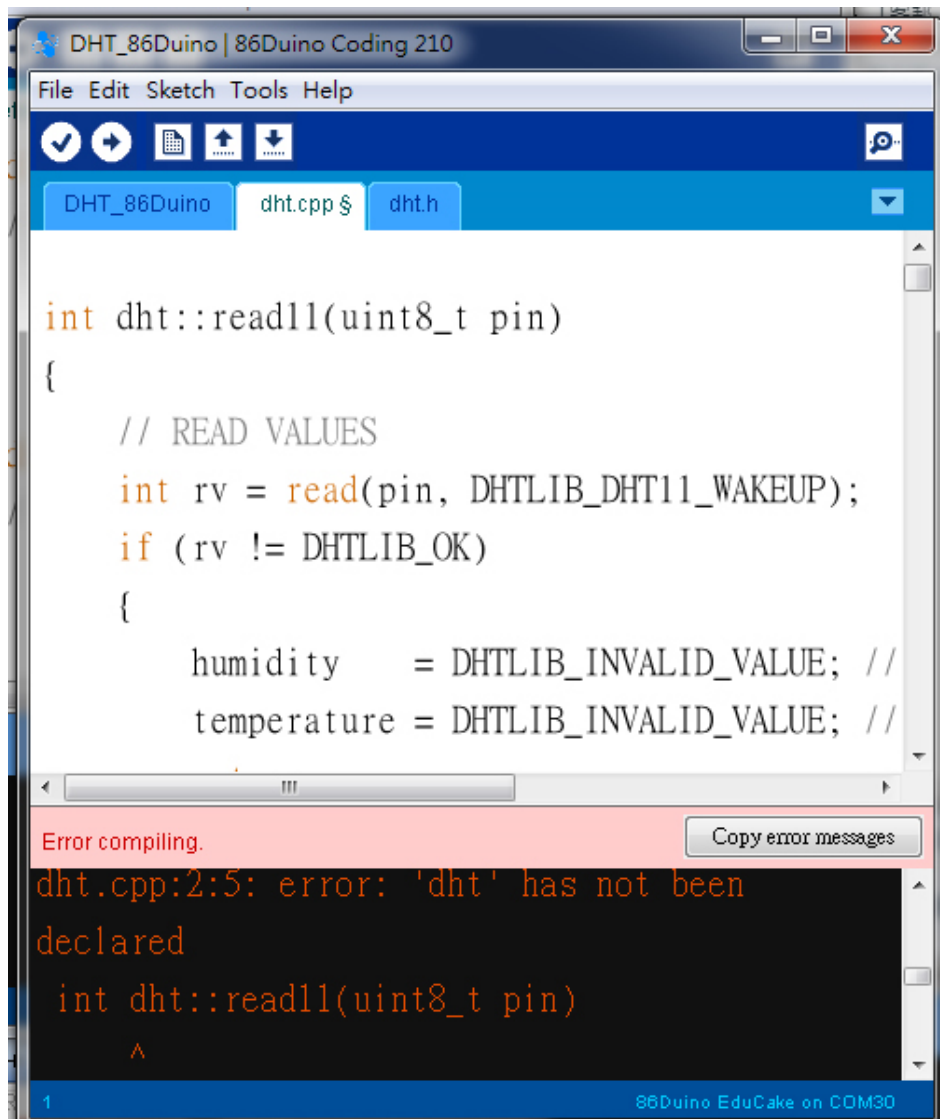


圖 207 無法編譯的 CPP 檔

我們發現，dht.cpp 雖然將 dht 類別實作出來，但是卻缺乏 dht 類別的宣告與產生，但是我們又不能在 dht.cpp 之中又把 dht 類別重新宣告與產生，雖然這樣可以編譯成功，但是，Arduino 函式庫的寫法並非這樣，這樣又回到原來

『DHT_OK_For_86duino_Product』的寫法。

所以筆者將 dht 類別的宣告與產生的 include 檔，用 include 的方法加入程式，將宣告與產生類別的 include 檔含入的語法，將 dht 類別的宣告與產生送入 dht.cpp 之中。

表 14 將宣告與產生類別的 include 檔含入的語法

```
#include "dht.h"
```

如下表所示，我將這上表這一行 include 程式加到 dht.cpp 檔之中，並且開始編譯。

表 15 改寫為 Include 檔

將 include 包入函式實體之 CPP 檔 (dht.cpp)

```
#include "dht.h"
int dht::read11(uint8_t pin)
{
    // READ VALUES
    int rv = read(pin, DHTLIB_DHT11_WAKEUP);
    if (rv != DHTLIB_OK)
    {
        humidity    = DHTLIB_INVALID_VALUE; // in-
        valid value, or is NaN preferred?
        temperature = DHTLIB_INVALID_VALUE; // in-
        valid value
        return rv;
    }

    // CONVERT AND STORE
    humidity    = bits[0]; // bits[1] == 0;
    temperature = bits[2]; // bits[3] == 0;

    // TEST CHECKSUM
```

```

        // bits[1] && bits[3] both 0
        uint8_t sum = bits[0] + bits[2];
        if      (bits[4]      !=      sum)      return
DHTLIB_ERROR_CHECKSUM;

        return DHTLIB_OK;
    }

    // return values:
    // DHTLIB_OK
    // DHTLIB_ERROR_CHECKSUM
    // DHTLIB_ERROR_TIMEOUT
    int dht::read21(uint8_t pin)
    {
        // dataformat & wakeup identical to DHT22
        return read22(pin);
    }

    // return values:
    // DHTLIB_OK
    // DHTLIB_ERROR_CHECKSUM
    // DHTLIB_ERROR_TIMEOUT
    int dht::read22(uint8_t pin)
    {
        // READ VALUES
        int rv = read(pin, DHTLIB_DHT22_WAKEUP);
        if (rv != DHTLIB_OK)
        {
            humidity      = DHTLIB_INVALID_VALUE; //
invalid value, or is NaN preferred?
            temperature = DHTLIB_INVALID_VALUE; //
invalid value
            return rv; // propagate error value
        }

        // CONVERT AND STORE
        humidity = word(bits[0], bits[1]) * 0.1;

```

```

        if (bits[2] & 0x80) // negative temperature
        {
            temperature = -0.1 * word(bits[2] & 0x7F, bits[3]);
        }
        else
        {
            temperature = 0.1 * word(bits[2], bits[3]);
        }

        // TEST CHECKSUM
        uint8_t sum = bits[0] + bits[1] + bits[2] + bits[3];
        if (bits[4] != sum) return
DHTLIB_ERROR_CHECKSUM;

        return DHTLIB_OK;
    }

    //////////////////////////////////////
    //
    // PRIVATE
    //

    // return values:
    // DHTLIB_OK
    // DHTLIB_ERROR_TIMEOUT
    int dht::read(uint8_t pin, uint8_t wakeupDelay)
    {
        // INIT BUFFERVAR TO RECEIVE DATA
        uint8_t mask = 128;
        uint8_t idx = 0;

        // EMPTY BUFFER
        for (uint8_t i = 0; i < 5; i++) bits[i] = 0;

        // REQUEST SAMPLE
        pinMode(pin, OUTPUT);
        digitalWrite(pin, LOW);
        delay(wakeupDelay);
    }

```

```

digitalWrite(pin, HIGH);
delayMicroseconds(40);
pinMode(pin, INPUT);

// GET ACKNOWLEDGE or TIMEOUT
unsigned int loopCnt = TIMEOUT;
while(digitalRead(pin) == LOW)
{
    if (--loopCnt == 0) return
DHTLIB_ERROR_TIMEOUT;
}

loopCnt = TIMEOUT;
while(digitalRead(pin) == HIGH)
{
    if (--loopCnt == 0) return
DHTLIB_ERROR_TIMEOUT;
}

// READ THE OUTPUT - 40 BITS => 5 BYTES
for (uint8_t i = 0; i < 40; i++)
{
    loopCnt = TIMEOUT;
    while(digitalRead(pin) == LOW)
    {
        if (--loopCnt == 0) return
DHTLIB_ERROR_TIMEOUT;
    }

    unsigned long t = micros();

    loopCnt = TIMEOUT;
    while(digitalRead(pin) == HIGH)
    {
        if (--loopCnt == 0) return
DHTLIB_ERROR_TIMEOUT;
    }
}

```

```

        if ((micros() - t) > 40) bits[idx] |= mask;
        mask >>= 1;
        if (mask == 0)    // next byte?
        {
            mask = 128;
            idx++;
        }
    }
    return DHTLIB_OK;
}
//
// END OF FILE

```

資料來源：86Duino 程式教學(網路通訊篇):86duino Programming (Networking Communication)(曹

永忠 et al., 2015a, 2015b)

原始碼下載網址：<https://github.com/brucetsao/techbang/tree/master/201510>

如下圖所示，我們可以看到 dht.cpp 可以正常編譯成功。

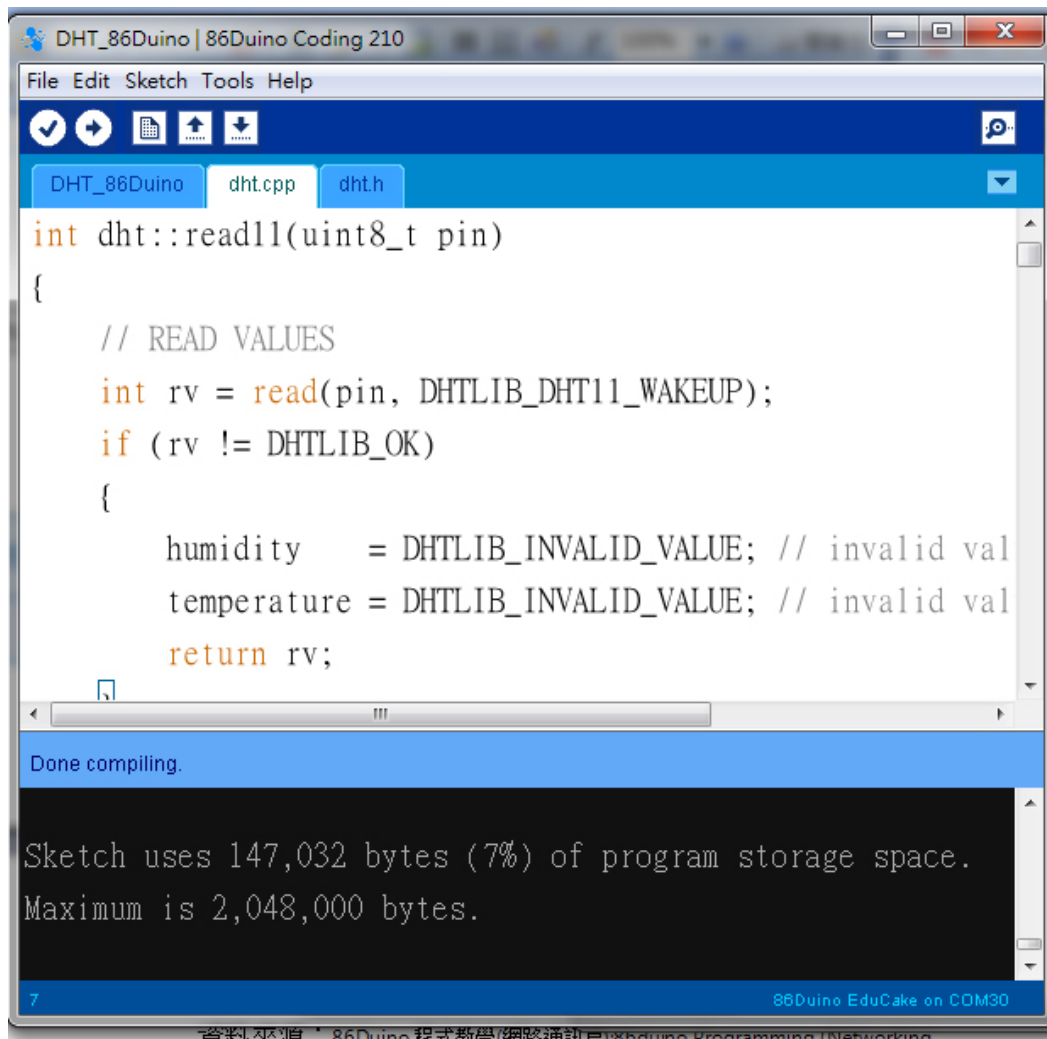


圖 208 可以編譯的 dht.cpp 檔

整合函式庫的範例

到這裡我們已經可以成功將 DHT11 溫濕度感測模組的函式庫改寫到 dht.cpp & dht.h 之中，但是我們發現，原來剩下的程式：`int chk = DHT.read11(DHT11_PIN);`，卻因 dht 類別的宣告與產生的相關程式碼都已移到 dht.cpp & dht.h 之中而無法編譯成功。

原因是原有的程式(DHT_OK_For_86duino_Product.ino)是沒有使用外部的函式庫，所以當將 dht 類別的宣告與產生的相關程式碼移出後，DHT 的物件自然消失

不見了。

所以我們既然已將 DHT11 溫濕度感測模組給與函式化，我們就可以使用標準的 Arduino 元件函式庫的語法，將該函式庫的 include 檔給與含入，就可以開始 DHT11 溫濕度感測模組。

如下表所示，我們使用『#include "dht.h"』的語法將 DHT11 溫濕度感測模組的函式庫包入，就可以使用『dht DHT』的方式，來產生 DHT 物件，其它的地方就跟原有的原有的程式(DHT_OK_For_86duino_Product.ino)一樣，可以不需修改就可以輕易使用 DHT11 溫濕度感測模組

表 16 整合函式庫的範例測試程式

整合函式庫的範例測試程式(DHT_86Duino)
<pre> #include "dht.h" #define DHT11_PIN 7 dht DHT; void setup() { // put your setup code here, to run once: Serial.begin(9600); //宣告監控畫面的通訊速率 Serial.println("DHT PROGRAM "); Serial.print("LIBRARY VERSION: "); // Serial.println(DHT_LIB_VERSION); Serial.println(); Serial.println("Type,\tstatus,\tHumidity (%),\tTemperature (C)"); } void loop() { // put your main code here, to run repeatedly: Serial.print("DHT11, \t"); int chk = DHT.read11(DHT11_PIN); if (chkDHT(chk) == 0) // 簡 查 </pre>

```

    {
        Serial.println("ERROR on init DHT Sensor") ;
        while (true) ;
    }

// DISPLAY DATA
Serial.print(DHT.humidity, 1);
Serial.print(",\t");
Serial.println(DHT.temperature, 1);

delay(2000);
}

unsigned int chkDHT( int chk )
{
    switch (chk)
    {
        case DHTLIB_OK:
            Serial.print("OK,\t");
            return 1;
        case DHTLIB_ERROR_CHECKSUM:
            Serial.print("Checksum error,\t");
            return 0;
        case DHTLIB_ERROR_TIMEOUT:
            Serial.print("Time out error,\t");
            return 0;
        default:
            Serial.print("Unknown error,\t");
            return 0;
    }
}
}

```

資料來源：86Duino 程式教學(網路通訊篇):86duino Programming (Networking Communication)(曹

永忠 et al., 2015a, 2015b, 2015f)

原始碼下載網址：<https://github.com/brucetsao/techbang/tree/master/201510>

如下圖所示，我們可以看到使用函式庫方式之 DHT11 溫濕度感測模組測試程式結果畫面。

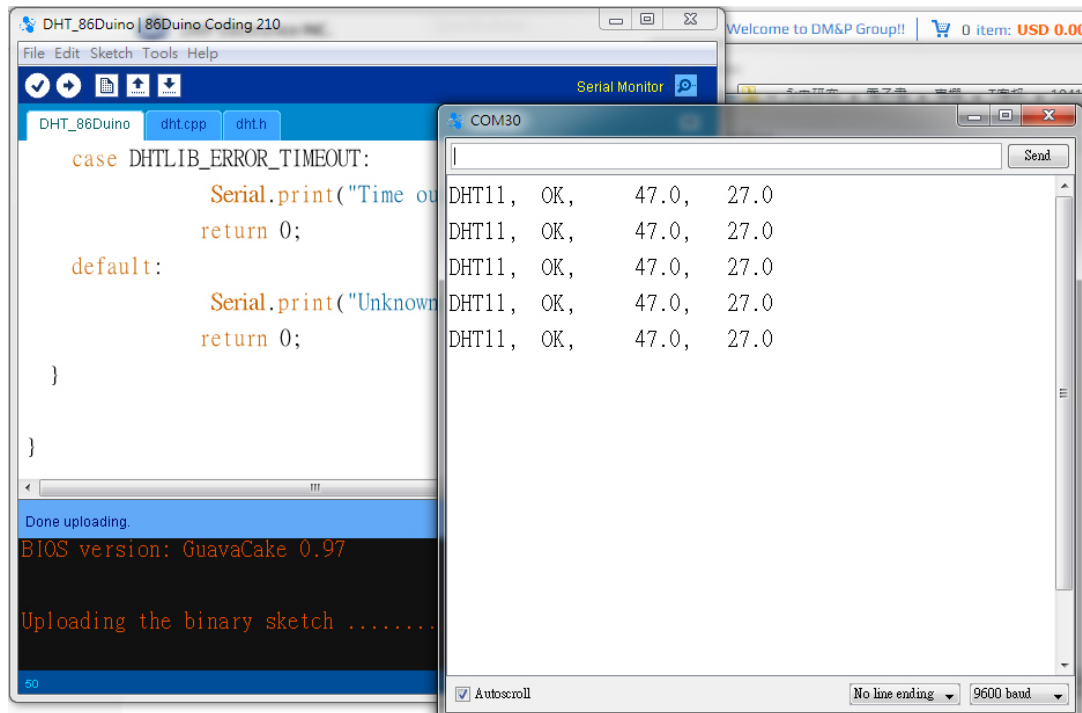


圖 209 使用函式庫方式之 DHT11 溫濕度感測模組測試程式結果畫面

章節小結

本章內容稍為艱深難讀，且讀者需要熟悉 C 語言與 C++ 語言的基礎，並對 Arduino 開發板對於外加的函式庫用法也有相當程度的了解，不過，網路上對於這樣進階的文章很少，國內更少，希望透過本文分享，可以有更多 Makers 可以在 Maker 路上走個更遠，更深入，這也是筆者分享此文的初衷。