

# Microcomputer Control

## Lecture 3

### Arduino程式撰寫基礎

Chih-Chiang Chen

[ccchenevan@mail.ncku.edu.tw](mailto:ccchenevan@mail.ncku.edu.tw)

2021


## 基本認知

- “程式”是指揮Arduino控制板做事情的一連串指令。
- 程式語言分爲“高階語言”與“低階語言”。
  - 高階  $\iff$  自排車  $\iff$  靈活性低、容易開、不需瞭解汽車結構較多
  - 低階  $\iff$  手排車  $\iff$  靈活性高、不容易開、需瞭解汽車結構較多
- Arduino控制板的程式語言爲簡化的“C語言”和“C++語言”。
  - C語言  $\iff$  “高階語言”
  - C語言  $\iff$  “真”工程師需習得之技能，初心者(大一)就能學
- 單晶片微電腦(控制板)只認得0和1構成的指令(稱“機械碼”)，我們必須把高階語言“翻譯”成0和1的機械碼，才能交給單晶片微電腦(控制板)執行。此“翻譯”過程叫做編譯(compile)。
- 單晶片微電腦的運算能力與記憶容量都很小，故常用較接近機械碼的組合語言(assembly)(低階語言)來開發其程式。

## 資料類型

- 資料類型

設定資料類型相當於設定“資料容器(變數)”的“可容納物類型(變數格式)”與“容器之容量(所對之位元數)”。

(藍色表都屬於整數類、表與C語言不同)

類型	名稱	佔用記憶體大小	數值範圍(10進制)
boolean	布林	8位元	0或1
byte 	位元組	8位元	0-255
char	字元	8位元	-128-127
int	整數	16位元	-32768-32767
long	長整數	32位元	-2147483648-2147483647
float	浮點數	32位元	$\pm 3.4028235E+38$
double 	雙倍精確浮點數	32位元	$\pm 3.4028235E+38$
unsigned char	正字元	8位元	0-255
unsigned int	正整數	16位元	0-65535
unsigned long	正長整數	32位元	0-4294967295

- C語言沒有byte資料型態
- C語言的double資料型態為64位元
- 藍色: 都歸屬於整數類

## 資料類型

- 設定某個變數之資料型態(即所謂的宣告變數) – 不設定初始值

資料型態 變數名稱;

- 設定某個變數之資料型態(即所謂的宣告變數) – 設定初始值

資料型態 變數名稱 = 初始值;

若資料型態為char，則初始值有兩種寫法：

- 初始值 = 'A' 表示把A這個字元放到該變數
- 初始值 = 65 表示把ASCII碼為65所對應到之字元放到該變數
- 當某個變數yyyy為char資料型態，則：
  - yyyy = 'a' 表示把a這個字元放到該變數yyyy
  - yyyy = 97 表示把ASCII碼為97所對應到之字元放到該變數yyyy

注意！我們不能把一個字元存放到一個資料型態為char的變數zzzz之後，又透過執行「www = zzzz」企圖想把zzzz所存放的字元所對應到的ASCII碼放到變數www中。

## 資料類型

- 如果在宣告變數時，在上述之資料類型前面加上`const`，則該變數將轉成常數，意即宣告完數值後後續程式不能再更改其數值。
- 在Arduino所使用的簡化版C語言中，`HIGH`為布林資料型態並已被設定為1，`LOW`為布林資料型態並已被設定為0；另外，`true`為布林資料型態並已被設定為1，`false`為布林資料型態並已被設定為0。

## 2進制 vs. 10進制

**10進制**(每一位數可有1, 2, 3, ..., 8, 9表示)轉**2進制**(每一位數可有0與1表示)：

$$362 \div 2 = 181 \longrightarrow \text{餘 } 0$$

$$181 \div 2 = 90 \longrightarrow \text{餘 } 1$$

$$90 \div 2 = 45 \longrightarrow \text{餘 } 0$$

$$45 \div 2 = 22 \longrightarrow \text{餘 } 1$$

$$22 \div 2 = 11 \longrightarrow \text{餘 } 0$$

$$11 \div 2 = 5 \longrightarrow \text{餘 } 1$$

$$5 \div 2 = 2 \longrightarrow \text{餘 } 1$$

$$2 \div 2 = 1 \longrightarrow \text{餘 } 0$$

$\implies$  由下至上，由左至右

**101101010**

**2進制**(每一位數可有0與1表示)轉**10進制**(每一位數可有1, 2, 3, ..., 8, 9表示)：

2進制	0	1	0	1	1	0	1	0	1	0
轉	×	×	×	×	×	×	×	×	×	×
換	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
過										
程	0	256	0	64	32	0	8	0	2	0
10進制	$0 + 256 + 0 + 64 + 32 + 0 + 8 + 0 + 2 + 0 = 362$									

我們說“2進制中的10110101”相當於“10進制中的362”。

## 2進制 vs. 10進制

**10**進制(每一位數可有1, 2, 3, ..., 8, 9表示)中的數值1, 2, 3, ..., 8, 9對應與**2**進制(每一位數可有0與1表示)中的數值快速轉換(參考前頁轉換方法可得)：

10進制數值	2進制數值
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

## 10進制 vs. 16進制

**10進制**(每一位數可有1, 2, 3, ..., 8, 9表示)轉**16進制**(每一位數可有1, 2, 3, ..., 8, 9, A(=10), B(=11), C(=12), D(=13), E(=14), F(=15)表示)：

$$\begin{array}{lll} 326 \div 16 = 22 \longrightarrow \text{餘 } 10 = A & \implies & \text{由下至上，由左至右} \\ 22 \div 16 = 1 \longrightarrow \text{餘 } 6 & & 16A \end{array}$$

注意，10進制中的10, 11, 12, 13, 14, 15對應A, B, C, D, E, F。

**16進制**(每一位數可有1, 2, 3, ..., 8, 9, A(=10), B(=11), C(=12), D(=13), E(=14), F(=15)表示)轉**10進制**(每一位數可有1, 2, 3, ..., 8, 9表示)：

16進制	1	6	A
轉	×	×	×
換	$16^2$	$16^1$	$16^0$
過			
程	256	96	10
10進制	256 + 96 + 10 = 362		

我們說“16進制中的16A”相當於“10進制中的362”。



## 10進制 vs. 16進制

**10**進制(每一位數可有1, 2, 3, ..., 8, 9表示)中的數值1, 2, 3, ..., 8, 9, 10, 11, 12, 13, 14, 15對應與**16**進制(每一位數可有1, 2, 3, ..., 8, 9, A(=10), B(=11), C(=12), D(=13), E(=14), F(=15)表示)中的數值快速轉換(參考前頁轉換方法可得)：

10進制數值	16進制數值
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F



## 2進制 vs. 16進制

$$\begin{array}{ll} 6 \div 2 = 3 \longrightarrow \text{餘 } 0 & \implies \text{由下至上，由左至右} \\ 3 \div 2 = 1 \longrightarrow \text{餘 } 1 & 0110 \text{ (不足四位元要補滿四位)} \end{array}$$

---

$$\begin{array}{ll} 1 \div 2 = 0 \longrightarrow \text{餘 } 1 & \implies \text{由下至上，由左至右} \\ & 0001 \text{ (不足四位元要補滿四位)} \end{array}$$

---

我們說“2進制中的10110101”相當於“16進制中的16A”。

---

在撰寫程式時，我們寫成：

$$\begin{array}{ll} a = 362 & (10\text{進制}) \\ a = 0b101101010 & (2\text{進制}) \\ a = 0x16A & (16\text{進制}) \end{array}$$

## 運算子

### ● 算數運算子

符號	功能	範例	說明
=	設定(assignment)	$x = y$	將 x 變數的內容設定給 y 變數
+	加(addition)	$a = x + y$	將 x 與 y 變數的值相加，其和放入 a 變數內
-	減(subtraction)	$b = x - y$	將 x 變數減 y 變數的值，其差放入 b 變數內
*	乘(multiplication)	$c = x * y$	將 x 與 y 變數的值相乘，其積放入 c 變數內
/	除(division)	$d = x / y$	將 x 變數除以 y 變數的值，其商放入 d 變數內
%	取餘數(modulo)	$e = x \% y$	將 x 變數除以 y 變數的值，其餘數放入 e 變數內

## 運算子

### ● 比較運算子

符號	功能	範例	說明
==	等於(equal to)	$x==y$	比較 x 與 y 變數是否相等，相等其結果為 1，不相等則為 0
!=	不等於(not equal to)	$x!=y$	比較 x 與 y 變數是否不相等，不相等其結果為 1，相等則為 0
<	小於(less than)	$x<y$	若 x 變數小於 y 變數，其結果為 1，否則為 0
>	大於(greater than)	$x>y$	若 x 變數大於 y 變數，其結果為 1，否則為 0
<=	小於或等於 (less than or equal to)	$x<=y$	若 x 變數小於或等於 y 變數，其結果為 1，否則為 0
>=	大於或等於 (greater than or equal to)	$x>=y$	若 x 變數大於或等於 y 變數，其結果為 1，否則為 0

注意！運算後的結果之資料型態為布林。

## 運算子

- 布林運算子

符號	功能	範例	說明
&&	及(and)	$(x > y) \&\& (x > z)$	若 x 大於 y 和 z 變數，其結果為 1，否則為 0
	或(or)	$(x > y)    (x > z)$	若 x 大於 y 或大於 z 變數，其結果為 1，否則為 0
!	反相(not)	$!(x > y)$	若 x 大於 y，其結果為 0，否則為 1

注意！運算後的結果之資料型態為布林。

## 運算子

- 位元算子 ( $x = 0x26$ ;  $y = 0xe2$ )

符號	功能	範例	說明
&	位元及(bitwise and)	$a = x \& y$	將 $x$ 、 $y$ 變數作位元及運算，其結果為 $0x22$ 放入 $a$ 變數中， $x$ 、 $y$ 內容不變
	位元或(bitwise or)	$a = x   y$	將 $x$ 、 $y$ 變數作位元或運算，其結果為 $0xe6$ 放入 $a$ 變數中， $x$ 、 $y$ 內容不變
^	位元互斥或(bitwise xor)	$a = x \wedge y$	將 $x$ 、 $y$ 變數作位元互斥或運算，其結果為 $0xc4$ 放入 $a$ 變數中， $x$ 、 $y$ 內容不變
~	取 1's 補數(bitwise not)	$a = \sim x$	將 $x$ 變數取 1's 補數運算，其結果為 $0xd9$ 放入 $a$ 變數中， $x$ 內容不變
<<	位元左移(bitshift left)	$a = x << 2$	將 $x$ 變數的值左移兩個位元，其結果為 $0x98$ 放入 $a$ 變數中， $x$ 內容不變
>>	位元右移(bitshift right)	$a = x >> 1$	將 $x$ 變數的值左移一個位元，其結果為 $0x13$ 放入 $a$ 變數中， $x$ 內容不變

## 運算子

- 位元算子 ( $x = 0x26$ ;  $y = 0xe2$ )

符號	功能	範例	說明
&	位元及(bitwise and)	$a = x \& y$	將 $x$ 、 $y$ 變數作位元及運算，其結果為 $0x22$ 放入 $a$ 變數中， $x$ 、 $y$ 內容不變

$$0x26 = 0b00100110$$

$$0xe2 = 0b11100010$$

$$a = 0b00100010 = 0x22$$

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



## 運算子

### ● 簡潔算數算子

符號	功能	範例	說明
++	加 1 (increment)	a++	等於 $a=a+1$
--	減 1 (decrement)	a--	等於 $a=a-1$
+=	加入(compound addition)	a+=b	等於 $a=a+b$
-=	減去(compound subtraction)	a-=b	等於 $a=a-b$
*=	乘入(compound multiplication)	a*=b	等於 $a=a*b$
/=	除(compound division)	a/=b	等於 $a=a/b$
&=	位元及(compound bitwise and)	a&=b	等於 $a=a\&b$
=	位元或(compound bitwise or)	a =b	等於 $a=a b$
%=	取餘數(compound bitwise modulo)	a%=b	等於 $a=a\%b$
<<=	位元左移(compound bitshift left)	a<<=b	等於 $a=a<<b$
>>=	位元右移(compound bitshift right)	a>>=b	等於 $a=a>>b$

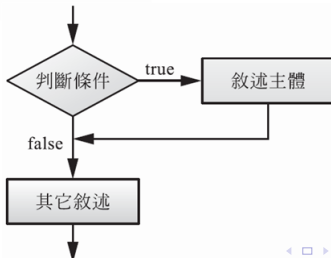
## 控制流程

- if敘述

```
if (判斷條件)
{
    敘述 1;
    敘述 2;
    .....
    敘述 n;
}
```

} If 敘述的主體

- 說明：**「判斷條件」之資料型態必須為**整數**或**布林**。當判斷條件為**整數****資料型態**時，數值**非0**時則條件成立，當值**等於0**時則條件不成立；當判斷條件為**布林**時，數值**等於1**時則條件成立，當值**等於0**時則條件不成立；

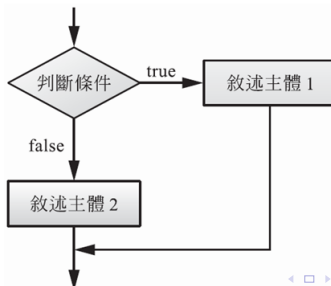


## 控制流程

- if-else敘述

```
if (判斷條件)
{
    敘述主體 1; //若判斷條件成立，則執行此部份
}
else
{
    敘述主體 2; //若判斷條件不成立，則執行此部份
}
```

- 說明:



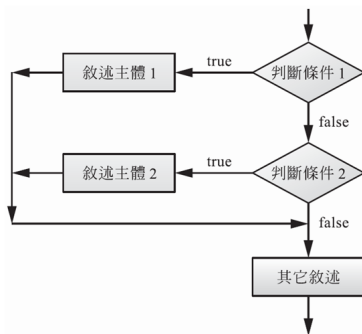
## 控制流程

- if-else-if敘述

```
if (判斷條件 1)
{
    敘述主體 1;
}
else if (判斷條件 2)
{
    敘述主體 2;
}
```

若判斷條件 1 成立，則執行這個部份  
若判斷條件 2 成立，則執行這個部份

- 說明：



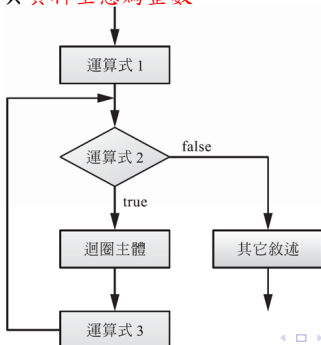
## 控制流程

- for迴圈

```
for (運算式 1; 運算式 2; 運算式 3)
{
    迴圈主體 ;
}
```

其中運算式1用以設定迴圈變數初值，其資料型態為整數，運算式2用以設定判斷條件 ( $i \leq 255$ )，其資料型態為整數或布林，運算式3用以設定迴圈變數增量 ( $i+=5$ )，其資料型態為整數。

- 說明:



## 陣列

- 一維陣列( $1 \times n$ 矩陣) – 不設定初始值

資料型態 陣列名稱[陣列大小( $n$ )];

- 範例:

```
int led[10];      // 宣告整數陣列 led，可存放 10 個元素
float temp[7];    // 宣告浮點數陣列 temp，可存放 7 個元素
char name[12];    // 宣告字元陣列 name，可存放 12 個元素
```

- 一維陣列( $1 \times n$ 矩陣) – 設定初始值

資料型態 陣列名稱[陣列大小( $n$ )]={初始值0, 初始值1, ..., 初始值 $n-1$ };

- 範例:

```
int led[4]={0x01,0x02,0x04,0x08}; //宣告整數陣列 led，並設定陣列初值
```

定義完此陣列後，我們在程式中輸入“ $a = led[0]$ ”會得到“ $a = 0x01$ ”，“ $a = led[1]$ ”會得到“ $a = 0x02$ ”，“ $a = led[2]$ ”會得到“ $a = 0x04$ ”，“ $a = led[3]$ ”會得到“ $a = 0x08$ ”。

## 陣列

- 二維陣列( $m \times n$ 矩陣) – 不設定初始值

資料型態 陣列名稱[陣列大小( $m$ )][陣列大小( $n$ )];

- 二維陣列( $m \times n$ 矩陣) – 設定初始值

資料型態 陣列名稱[陣列大小( $m$ )][陣列大小( $n$ )]  
= {{初值00, 初值01, ..., 初值0n}, ..., {初值m0, 初值m1, ..., 初值mn}};

- 範例:

```
int num[3][2]={ {10,11}, {20,22}, {30,33} } ;
```

定義完此陣列後，我們在程式中輸入“ $a = \text{num}[0][1]$ ”會的到“ $a = 11$ ”，“ $a = \text{num}[2][0]$ ”會得到“ $a = 30$ ”。

## 自訂函數

- 函數原型宣告(告訴編輯器此程式用到什麼函數)

函數回傳值資料型態 函數名稱(引數型態1, 引變數型態2,..., 引變數型態 $n$ )

若函數不需要引數，則引數型態的地方改成void(C語言要求)或直接空著不填即可(Arduino專用簡化版C語言才可以)；若函數沒有回傳值，則函數回傳值資料型態的地方改成void。

- 範例:

```
int add(int, int);    /* add 函數原型的宣告 */
```

函數add的回傳值資料型態是整數，並且有兩個引數，兩個都是整數型態。

- 範例:

```
void star(void);    /* star 函數原型的宣告 */
```

函數star的沒有回傳值，並且也沒有引數。

注意！程式中有用到函數時，都要在程式的一開始將此程式會用到的函數之函數原型做好宣告。當然，我們也可以直接在需要做函數原型宣告的地方，直接用函數定義來取代。



## 自訂函數

- 函數定義(告訴編輯器此函數的內容)

```
函數回傳值資料型態 函數名稱(引數型態1 引數1,..., 引數型態n 引數n)
{
    變數宣告;
    函數內容;
    return 運算子;
}
```

若函數不需要引數，則引數型態的地方改成void(C語言要求)或直接空著不填即可(Arduino專用簡化版C語言才可以)；若函數沒有回傳值，則函數回傳值資料型態的地方改成void，且“return 運算子;”不需寫。

- 範例:

```
int add(int num1, int num2)
{
    int sum;
    sum=num1+num2;
    return sum;
}
```

函數add有兩個引數，都為整數型態。當函數add被**呼叫**時需指定其兩個引數之數值，而函數add執行完時，函數add將以整數型態回傳此兩指定數值之引數和。

**注意！**程式中有用到函數時，都要在程式中(一般放程式的最後面)將此程式會用到的函數定義好。

## 自訂函數

- 在程式中呼叫函數(使用函數) – 當所定義之函數是具有回傳值時

變數 = 函數名稱(引數1,..., 引數 $n$ )

- 範例:

out = add (5,3)

呼叫函數add並給其兩引述分別為5與3，且將其回傳值指定給變數out。

- 在程式中呼叫函數(使用函數) – 當所定義之函數是沒有回傳值時

函數名稱(引數1,..., 引數 $n$ )

- 範例:

evan (6,1)

呼叫函數evan並給其兩引述分別為6與1，而函數evan沒有回傳值。

- 在程式中呼叫函數(利用函數) – 當所定義之函數是沒有回傳值且沒有引數時

函數名稱()

- 範例:

star ()

呼叫函數star，而函數add沒有回傳值也沒有引數。

## 局部/全域變/常數

在撰寫程式中，我們經常會設定變數/常數。隨著變數宣告的地方(寫在哪裡)之差別，可分類成：

- **局部變/常數：**

若變/常數是在某一個函數的定義內做宣告，則此變/常數則為局部變/常數，只能在該函數內被使用。

- **全域變/常數：**

若變/常數是非在任意一個函數的定義內做宣告，則此變/常數則為全域變/常數，整個程式都可使用。