機械手臂軟體教程

- 本教程包含 30 餘個進度，適合國中小用拖拉方式來寫程式（可選 S4A 或 Ardublock），同時亦附 C 語言版本供高中職以上玩家學習，就算您沒有程式基礎，也可經由這 30 餘個教程學得控制手臂的必備程式知識。

- 要學習如何控制 meArm.Joystick（之後簡稱 meArm），首先要安裝 1 個軟體和 1 個驅動程式（步驟 A 到 C）。安裝完後便有原廠的 C 語言開發環境。

- 如果您要用圖形開發環境（S4A 或 Ardublock），參考步驟 D，E 。

- 開始測程式後若要再回復原廠設定，請到 Step 38 或下載 meArm_bt.ino（藍牙），meArm_bt2.ino（藍牙＋自動展示，按搖桿回復手動），或 meArm.ino 並在 Arduino IDE 內燒入。

- 教程源碼可一次下載。（教程源碼.zip）

- 要用安卓手機透過藍牙控制手臂，請下載安卓 APP：meArm_Joystick.apk

- 安卓 APP 源碼（App Inventor 2），請下載：meArm_Joystick.aia

註：要用藍牙，需升級藍牙擴充板喔。藍牙板為 HC- 06。

A) Arduino IDE

B) 安裝驅動程式：

Windows：

雙擊 Arduino_driver.exe 即可。

MAC：

1) 雙擊 ch34xInstall.pkg

2) 如果是 Yosemite (OSX 10.10), 請打開 terminal, 輸入以下後重新開機（其它版本不用此步驟）

sudo nvram boot-args="kext-dev-mode=1"

Linux：

按照 CH341SER_LINUX.zip 內所附之 readme.txt 操作即可

C) 確認 Arduino IDE 開發環境

Windows：

1）系統- 裝置管理員- 連接埠（插入機器手臂，確認"CH- 340"在那一個 COM 埠）

2）Board 選 “Arduino UNO”

3）Port 選 “CH- 340” 所在埠，在此為 COM3

4）Programmer 選 “Arduino as ISP”

MAC：

1）應用程式- 工具程式- 系統資訊（插入機器手臂，確認"USB2.0-Serial"有顯示）
2）Board 選 “Arduino UNO”

3）Port 選 “dev/tty.wchusbserial410”

4）Programmer 選 “Arduino as ISP”

D）S4A

1）在 S4A 安裝處，備份 S4A.Image 並以所附的 S4A.Image 取代之 （若是藍牙板，請下載 S4A_bt.Image，改名為 S4A.Image 後取代之）

Windows：C:\Program Files\S4A\S4A.Image

MAC：Macintosh HD/應用程式/S4A/S4A.Image

2）打開 Arduino IDE，上傳 S4AFirmware15_meArm.ino 到機器手臂

E) Ardublock

先安裝 Arduino IDE，然後再將 “ardublock-beta-20140828.jar” 拷貝到 :

Windows 7 : C:\Users\lienhungcheng\Documents\Arduino\tools\ArduBlockTool\tool\ardublock-beta-20140828.jar

Windows XP : C:\Program Files\Arduino\tools\ArduBlockTool\tool\ardublock-beta-20140828.jar

MAC： /Users/lienhungcheng/Documents/Arduino/tools/ArduBlockTool/tool/ardublock-beta-20140828.jar

註： 其中路徑上的 “lienhungcheng” 應改為您的 username，更詳細步驟在此。

 Arduino_driver.EXE227 KB

 CH341SER_LINUX.ZIP8 KB

 CH341SER_MAC.ZIP250 KB

 ardublock-beta-20140828.jar5 MB

 S4AFirmware15_meArm.ino7 KB

 S4A.image7 MB

 教程源碼.zip1 MB

 meArm_Joystick.aia667 KB

 meArm_Joystick.apk2 MB

 meArm.ino2 KB

 S4A_bt.image7 MB

 mearm_bt.ino3 KB

 meArm_bt2.ino4 KB

Step 1: 給我一盞燈

我們可以透過 Arduino 開發環境，用 C 語言來寫程式控制 MeArm，也可以在圖形介面用拖拉方式產生程式，現在我們開始第一堂課：“給我一盞燈”

使用 S4A

步驟 1 ： 打開 S4A 程式

步驟 2 ：拖拉成此程式
步驟 3 ：點擊綠色旗子，即可看到 MeArm 的 LED 燈亮了。

使用 ArduBlock

步驟 1 ：選擇 Tools/ArduBlock

步驟 2 ：拖拉成此程式

步驟 3 ：按 “Upload to Arduino” 即可看到 MeArm 的 LED 燈亮了。

使用 Arduino IDE

步驟 1 ：打開 _01.ino

步驟 2 ：點擊 “Upload” ，是一個右向箭頭符號，即可看到 MeArm 的 LED 燈亮了。

說明：LED 的腳位為 3（S4A 中腳位為 12），所以當設定該腳位為 HIGH 時，燈就亮了。

註：您可在 S4A 內直接打開所附的 “.sb” 檔（圖形介面），也可在 ArduBlock 內直接打開所附的 “.abp” 檔（圖形介面），也可在 Arduino 中直接打開所附的 “.ino” 檔（C 語言），直接上傳到手臂運行，亦可產生同樣效果。


01.abp1 KB


_01.ino89 bytes


01.sb114 KB

Step 2: 3 秒後熄滅它

如果我們要在 3 秒後熄滅 LED 燈，那就在 delay 3000 毫秒後，設腳位為 LOW 即可。

02.abp3 KB

02.sb114 KB

_02.ino135 bytes

Step 3: 再閃一下，寶貝

燈既然能亮能滅，那何不閃爍一下呢？就讓我們間隔設為 1 秒讓它閃兩次。

_03.ino246 bytes

03.sb114 KB

03.abp7 KB

Step 4: 變數初體驗

假設我們要將閃爍時間從間隔 0．2 秒改為間隔 0．1 秒或 0．05 秒，要手動改變每個設定就會很累，這時可用變數來增加效率，我們只要設定一次變數，未來要改變時間間隔時就只要更動一次變數內容即可。

04.abp8 KB

_04.ino332 bytes

04.sb114 KB

Step 5: 變數也可以自嗨

既然有了變數，那麼就讓它多變幾次，您可試一下變數或減半，或加倍，或加 0.1 秒，或減 0.25 秒喔。

```
mytime = mytime / 2;

mytime = mytime * 2;

mytime = mytime + 100;

mytime = mytime - 250;
```

_05.ino303 bytes

05.abp9 KB

05.sb114 KB

Step 6: 使用照妖鏡讓變數現出原形

既然變數這麼方便，所以更要掌握變數才行，我們可以用以下指令讓變數現出原形。

"Serial.println(變數名);"

其中，Serial 是序列埠，作用是將 Arduino 的資料傳送到電腦的序列埠，這時只要按下 "Serial Monitor" 就可看到資料內容了。

_06.ino377 bytes

06.sb114 KB

06.abp11 KB

Step 7: 如果，我是真的...

“如果” 是一種判斷式，就像 “如果下雨，就帶雨傘。如果刮風，就穿風衣” 。情況成立，也就是“真”時，才做後面的動作。例如：

“如果（10＞1），則亮 LED 燈。”

這時因為 10 一定大於 1，所以 LED 燈就會亮了。

07.sb114 KB

07.abp3 KB

_07.ino118 bytes

Step 8: 如果，變數也來湊熱鬧

通常我們會將變數跟 “如果” 搭配使用，例如本例中的 light 變數，它可能代表的是光線亮度，當我們在判斷是否開燈時，就可根據 light 值，例如此例若 light 值小於 10 就開燈。

```
int light = 5;
if (light < 10)

{

digitalWrite(3, HIGH);

delay(1000);

}
```

08.sb115 KB

08.abp4 KB

_08.ino152 bytes

Step 9: 如果的好朋友：布林

在前例中的 “if (light < 10)” 判斷式，若成立，則稱之為 “真”，反之則為 “假”。這 “真假” 兩種值我們又稱為 “布林”，也就是 “true/false”。亦即結果為真，則進行後續動作，若為假，則不做任何動作。

09.sb115 KB

09.abp8 KB

_09.ino263 bytes

Step 10: 向左走，向右走（if-else）

既然布林有兩種值，那當然可以 “若真則向左邊，否則向右邊”

if (true) {

left turn

} else {

right turn

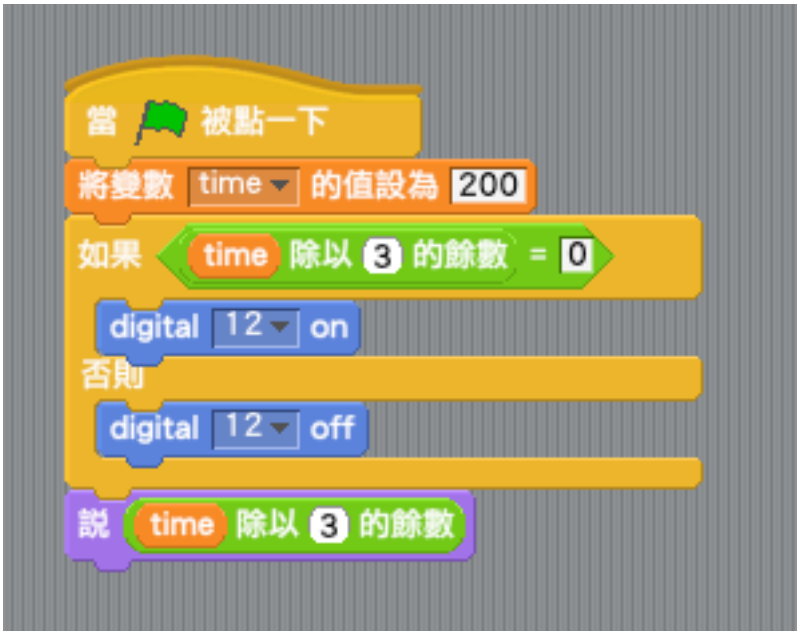
}

10.sb115 KB

10.abp4 KB

_10.ino193 bytes

Step 11: 向左走，向右走（if-else）續 1



了解（if-else）功能後，接著就可對判斷式加變化，例如：“若是 3 的倍數，則點燈，否則熄燈。”（其中｜是取餘數之意）

```
if (num ｜ 3 == 0)

{

digitalWrite(2, HIGH);

} else {

digitalWrite(2, LOW);

}
```

11.sb115 KB

11.abp8 KB

Step 12: 向左走，向右走（if-else）續 2

因為判斷式是程式設計的重要基礎，所以值得咱們再熟練熟練，接續上一個教程，你能寫出來嗎？

“如果是單數，則 LED 閃一下，否則閃兩下”

12.abp14 KB

12.sb116 KB

Step 13: 迴圈：叫我大神，重覆的事情交給我做就對了

除了判斷式之外，電腦最世界無敵猛的把戲就是迴圈了，重覆的事情找它做就對了。

例如：連續閃爍 LED 5 次

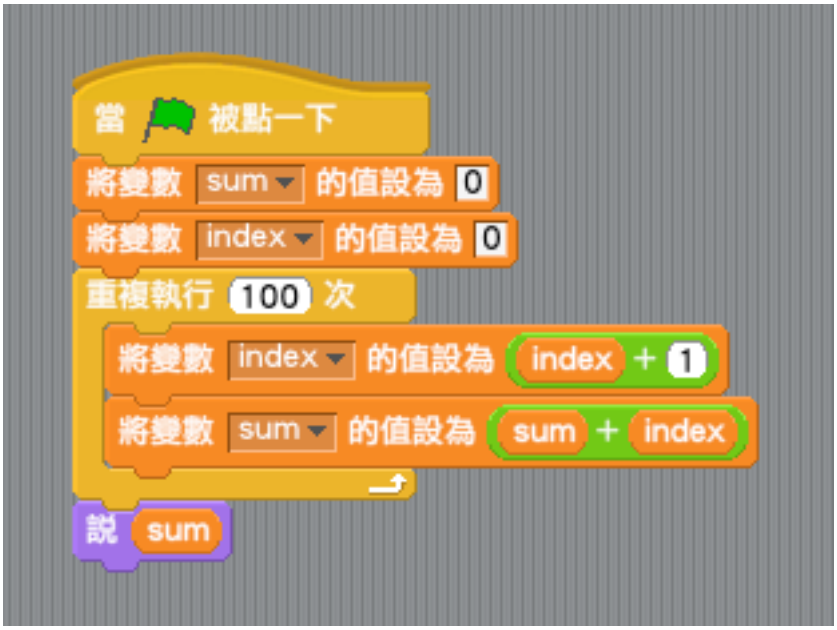
基本指令格式為 “for (int i = 0; i < 5; i++)"

13.abp7 KB

13.sb116 KB

_13.ino281 bytes

Step 14: 用迴圈幫你從 1 加到 100



還記得數學神童高斯怎麼累加 1 到 100 嗎，若能妥善運用迴圈，就算不是神童也能算出最後答案哩。

14.abp7 KB

14.sb118 KB

_14.ino228 bytes

Step 15: 迴圈也可以只加偶數喔

若迴圈只能做一成不變的事那就小看迴圈了。在此例中，咱們要在每個輪迴內加入判斷式，若成立才做動作，也就是在 1 到 100 之中，我們跳過奇數，只累加偶數。

15.abp9 KB

15.sb118 KB

_15.ino267 bytes

Step 16: 迴圈的間隔（step）

接續上一個只加偶數的問題，本節要介紹“迴圈的間隔”，是一種隔幾次才做一次的概念。所以若要加總 1 到 100 之間的偶數，意思就是：

“從 0 開始，每隔 2 次（中間空 1 次）才加總”。

16.sb118 KB

16.abp7 KB

_16.ino221 bytes

Step 17: 那能不能用 step 計算一年有幾個星期天呢

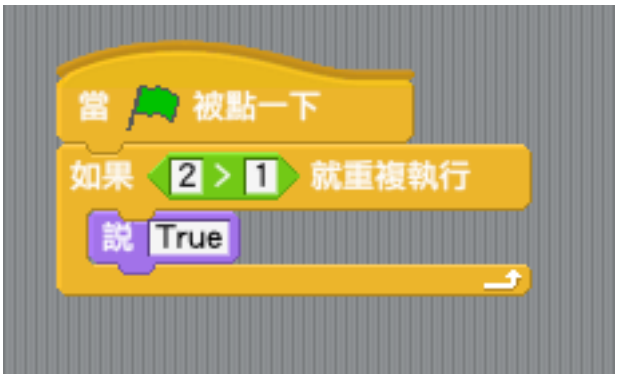
接續上一個“間隔”概念，你能算出一年有多少個星期天嗎？（要間隔多少呢？）

17.sb120 KB

17.abp9 KB

_17.ino276 bytes

Step 18: 迴圈 2：當條件符合就一直做，一直做，一直做 …



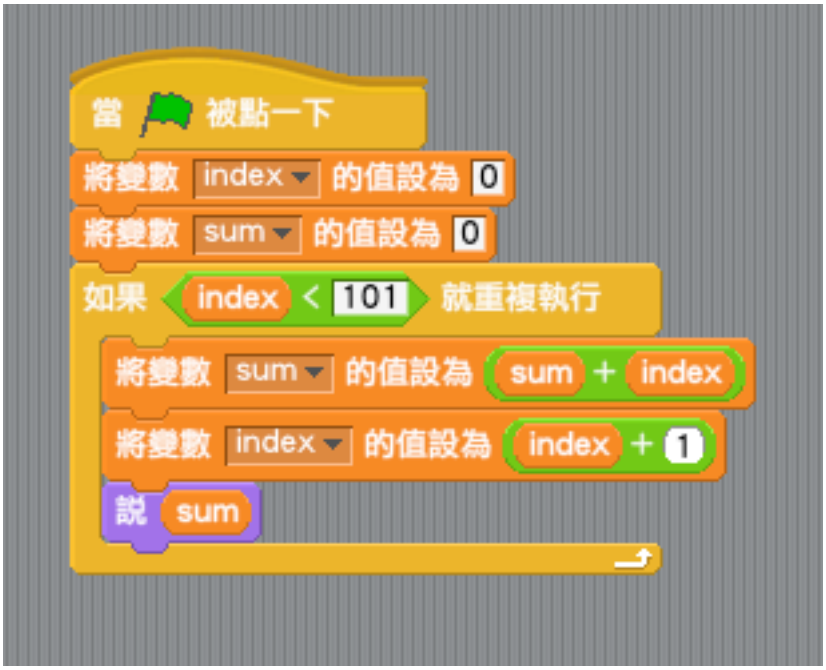
不管任何程式語言，迴圈都是如此的重要，所以在此要再介紹迴圈的另一種格式 while。這指令的要點就是如果條件為真，就一直做到天荒地老。像本例的條件（2 > 1）永遠為真，所以就會永遠一直印出字串 "true"。

18.sb119 KB

18.abp3 KB

_18.ino112 bytes

Step 19: 迴圈 2 的標準型



為了不讓電腦跑迴圈跑到掛，通常會加一條件限制，讓布林值不要永遠都是 "true"，並獲得所需結果。本節內容即是如何用 while 加總 1 到 100。

19.sb119 KB

19.abp9 KB

_19.ino223 bytes

Step 20: 讓 LED 閃到歪腰

若要重覆做一個永遠不變的事情，用 while 是最方便的，本節即是要挑戰如何讓 LED 燈持續閃爍到天荒地老。

20.abp5 KB

20.sb120 KB

_20.ino194 bytes

Step 21: 函數：每次閃爍一次都要寫 4 行，真麻煩啊

為了讓程式更簡潔，設計師習慣將“相同的程式碼包成一個黑箱，只要執行黑箱即可”。這個黑箱就叫函數或副程式。妥善利用副程式不僅程式看起來簡潔，而且運行效率提高，一舉兩得。

本例要展示如何將“閃爍”程式碼包進黑箱。

21.abp6 KB

21.sb121 KB

_21.ino232 bytes

Step 22: 函數＋迴圈：讓它閃 10 次

包成黑箱後，我們便可視其為一條指令，將它放到迴圈中，要連續執行幾次都行。本例展示如何在迴圈中呼叫副程式，讓 LED 閃爍 10 次。

22.sb121 KB

22.abp6 KB

_22.ino246 bytes

Step 23: 迴圈也可做在函數內

不僅可以在迴圈中運行副程式，反過來當然也可以在副程式中運行迴圈。此例要展示的即為直接在副程式中設定要執行的迴圈次數。

23.abp6 KB

23.sb121 KB

_23.ino252 bytes

Step 24: 函數的參數（僅限 C 語言，Ardublock 尚無此功能）

_24

```
void flash(int repeat){
  for (int i = 0; i < repeat; i++){
        digitalWrite(3, HIGH);
        delay(100);
        digitalWrite(3, LOW);
        delay(100);
  }
}

void setup(){
  pinMode(3, OUTPUT);
  flash(5);
  delay(1000);
  flash(7);
}
void loop()
{
}
```

副程式的作用強大，還可以更靈活使用，例如我們在呼叫副程式時，順帶提供一個參數，指定要它閃爍 LED 幾次。

_24.ino245 bytes

Step 25: 可以給多個參數嗎？（僅限 C 語言，Ardublock 尚無此功能）

```
_25 §
void flash(int repeat, int t){
  for (int i = 0; i < repeat; i++){
        digitalWrite(3, HIGH);
        delay(t);
        digitalWrite(3, LOW);
        delay(t);
  }
}

void setup(){
  pinMode(3, OUTPUT);
  flash(5, 50);
  delay(1000);
  flash(7, 200);
}

void loop()
{
}
```

副程式不僅可以接受參數，而且還可以同時接受多個。所以如果我們要設定 LED 閃爍幾次，每次延遲幾毫秒，只要透過參數指定就行了。本例展示兩個參數的副程式。

 _25.ino257 bytes

Step 26: 挑戰再挑戰（僅限 C 語言，Ardublock 尚無此功能）

_26

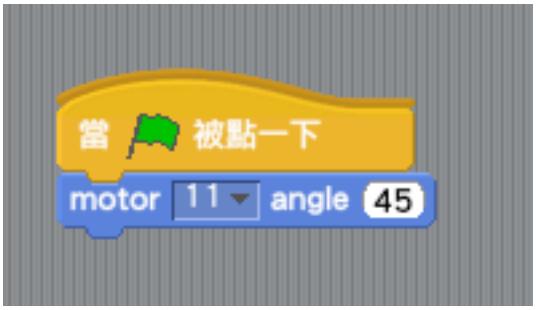
```
void sum(int num){
  int total = 0;
  for (int i = 1; i <= num; i++){
        total = total + i;
  }
  Serial.println(total);
}

void setup(){
  Serial.begin(9600);
  sum(10);
}

void loop()
{
}
```

帶有參數的副程式既強大又常用，接下來的練習是鍛鍊自己是否能做出一個加總的副程式，例如若要加總 1 到 100，則只要寫一條指令即可。

sum(100);

_26.ino190 bytes

Step 27: 伺服馬達初體驗



花了那麼多篇幅講了程式最核心的概念，現在要進入機器手臂正題了，只要你前面有跑過，相信我，接下來花不了多少時間你就能學會駕馭機器手臂了。

機器手臂是由伺服馬達所帶動，我們只要設定伺服馬達所接的腳位，接下來再設定所要轉到的角度即可。要注意的是伺服馬達的角度範圍是 0 到 179 度，如果角度超出兩邊範圍，或部件卡住，或因為螺絲鎖太緊導致馬達轉不到指定角度，則會因為電流過載而降低馬達和主板壽命。

此範例是將一顆伺服馬達設定接到腳位 11，然後轉到 0 度。

第一顆馬達為底座馬達。

27.abp1 KB

27.sb113 KB

_27.ino131 bytes

Step 28: 伺馬達角度範圍 0～179 度

既然伺服馬達的角度範圍是 0～179 度，那咱們就依序將這伺服馬達從 0 度慢慢轉至 179 度吧。

28.abp5 KB

28.sb114 KB

_28.ino236 bytes

Step 29: 試一下第二顆馬達，但只能 45～145 度

搞定了第一顆，那第二顆就依樣畫葫蘆唄。但這次只能從 45 度到 145 度。第二顆馬達可調整前後角度。

29.abp6 KB

29.sb114 KB

_29.ino314 bytes

Step 30: 試一下第三顆馬達，但只能 90～179 度

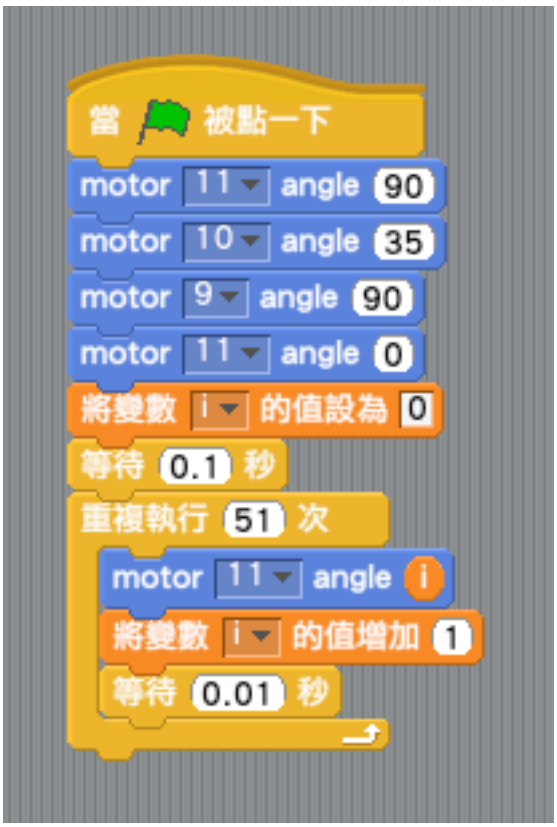
搞定前兩顆，第三顆馬達能調整上下，咱也依樣畫葫蘆，但限制只能從 90 度到 179 度。

30.sb114 KB

30.abp7 KB

_30.ino385 bytes

Step 31: 試一下第四顆馬達，但只能 0～50 度



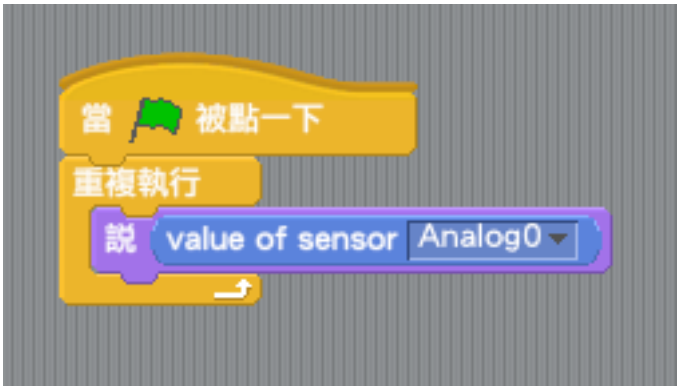
最後一顆馬達是夾具馬達，度數在 0 到 50 度之間。若未按照標準組裝步驟組成夾具，則會出現夾具緊閉或開大絕，這時則需打掉重練。

31.sb114 KB

31.abp8 KB

_31.ino456 bytes

Step 32: 讀搖桿資料，0～1023



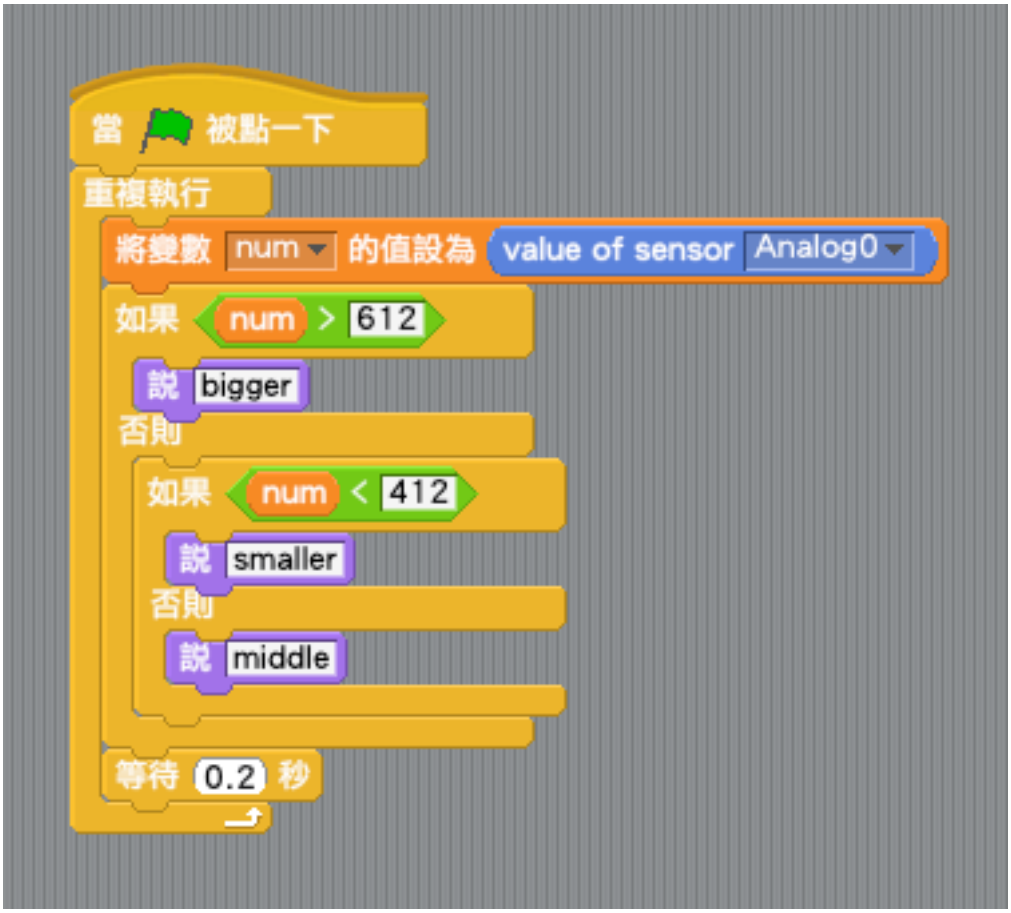
搞 完馬達，接下來要玩搖桿。本機器手臂附有二顆搖桿，每一顆皆有橫縱兩向，總共四向，所以能控制 4 顆馬達，搖桿依次接到 A0, A1, A2, A3。所以我們只要讀取這幾個腳位的值便可知道搖桿的角度。每個腳位值的範圍從 0 到 1024。此例中我們讀 A0 腳位的值然後由序列埠印出。

32.abp2 KB

32.sb114 KB

_32.ino107 bytes

Step 33: 將搖桿指令分類為大，中，小



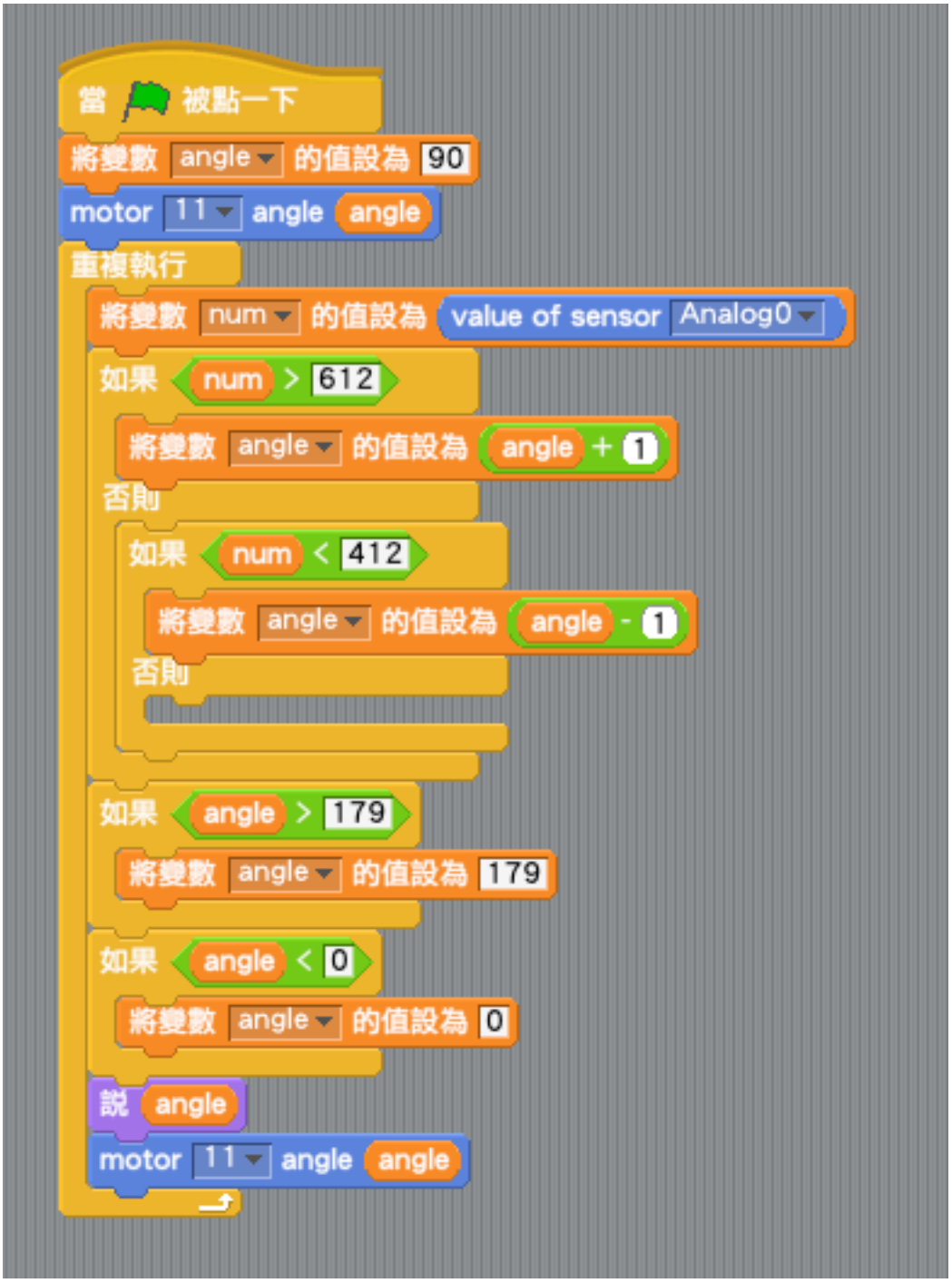
為了使用者操作方便，我們只將搖桿區分為 2 個動作，也就是前進和後退。根據此概念，我們可以設定若搖桿值超過 612，則是前進，若小於 412，則是後退，若介在中間則不進不退。此例結果可由序列埠得知結果是否正確。

33.sb116 KB

33.abp8 KB

_33.ino303 bytes

Step 34: 結合搖桿與馬達



接下來要將搖桿與馬達結合在一起，也就是如果搖桿前進，則馬達增加 1 度，反之則減一度。

34.sb117 KB

Step 35: 陣列終於出現了

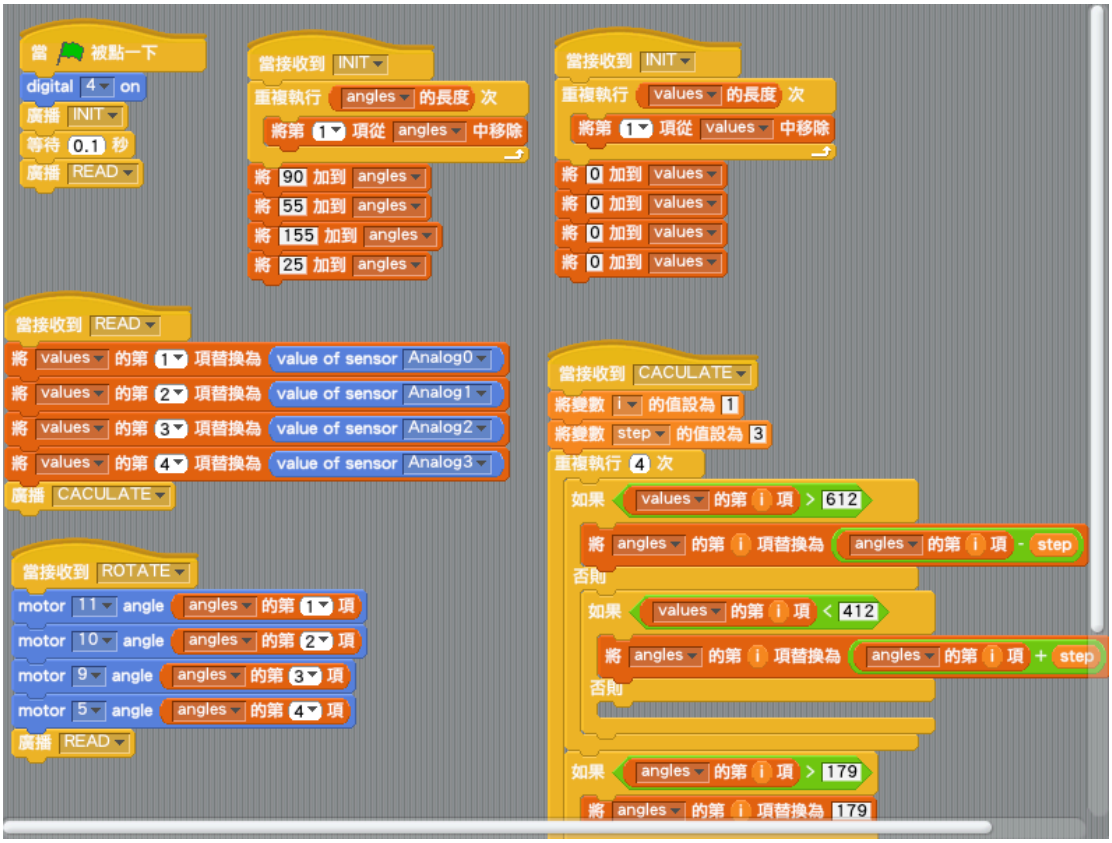
因為一次要讀取四個輸入，然後藉以控制四個輸出，這時若使用陣列，會讓程式顯得更簡潔。在此例中，我們要設定一個含有 4 個元素的陣列，給初值然後再印出來。

35.sb119 KB

35.abp9 KB

_35.ino268 bytes

Step 36: 將全部整合起來，手臂就全聽你的了



接下來要把全部整合起來，這樣就可以經由搖桿來控制所有馬達，進而控制整隻手臂。加油，你能看得懂程式的。

（圖片來源：Wei-Yu Chen）

36.sb116 KB

36.abp30 KB

_36.ino1 KB

Step 37: 機器手臂的自動模式

既然已經知道馬達的控制方法，咱們便可以事先設定每顆馬達的角度，然後一直重覆，這就是我們常在電視上看到的自動機器手臂的酷樣。此範例只是個參考，你也可以試著調到所需要的角度和次數，建立自己的生產線喔。


37.sb115 KB


37.abp31 KB


_37.ino1 KB

Step 38: 如何回復出廠手臂模式

請在 Arduino 環境中（非 Ardublock），燒入附件程式即可。所以大家多多練習，不用怕回不去滴。祝好運！

若有任何問題，歡迎 email 我，或加我好友，直接 PM 我或在 FB 上留言。


meArm_bt.ino3 KB