# 單元 1
# 導論與程式的編譯與執行

## 電腦程式語言

人與人之間溝通的工具稱爲語言，世界上所有民族，因其發源地不同，所以就有許多語言。例如，華語、英語及德語等。其次，人與電腦溝通的工具，則稱爲電腦程式語言。那爲什麼沒有電視語言、冰箱語言或冷氣語言呢？那是因爲這些機器的功能較爲簡單，只要幾個按鈕就能發揮其功能，但是電腦的功能非常多，多到連用整個鍵盤的所有按鍵都無法表現其功能，所以必須使用一些類似單字所組成的片語與敘述來發揮其功能，這些單字與片語的集合就稱爲電腦程式語言，簡稱程式語言。就如同人類也無法用 26 個字母表達所有感受與思維，必須藉助這些字母的排列組合，先組成單字，再由單字組合成片語與句子，才能充分表達其思維。目前流行的程式語言有 Java、C、C++、Visual Basic、C#、Python 等。

## 程式設計

使用程式語言中的指令連結資料稱爲一個敘述，再串連這些敘述使其完成一件工作，就稱爲程式設計。

## 程式設計功能

程式設計可將一連串重複的工作，使用程式語言，寫一段程式、儲存，然後就可無限次數重複使用，這樣可節省很多時間。例如，在計算器時代，要計算長方形面積，要進行 3＊2 的結果，就要按『3』、『＊』、『2』、『＝』等可得到答案，但是以上步驟當要重複計算時，都要重複

按以上運算元與運算子，這樣非常耗時與不便，所以有電腦程式語言的發展。所謂電腦程式設計，是利用代號或稱變數儲存以上運算元，再將運算元與運算子結合，形成一個敘述，那此一敘述就可重複利用，例如，以上計算通常我們會利用

```
a=b*c;
```

其次，只要輸入 b 與 c，就可無限次計算其值，此即爲程式設計。

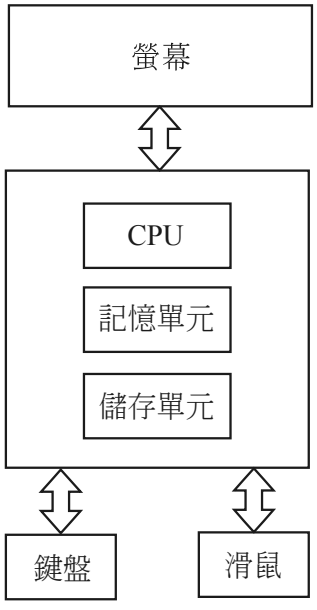
## 應用

以前的程式設計的應用比較偏重商用資料處理、科學數據的計算，例如，C/C++、Visual Basic、Java、C#、JavaScript、PHP 等等，這些程式設計的輸出入僅用到鍵盤、滑鼠、螢幕與硬碟或光碟等科技設備，只有特定人士、特定功能才會使用電腦。但是以目前的生活科技，無人自駕車已經可以在街上跑、鴻海郭台銘董事長也說五年後他的工廠不用開燈、不用人、不用供餐、不用供宿，使用機器人就可 24 小時生產，所以現在的程式設計可說是融入我們的生活了。還有更可怕的事，現在人口越來越少，會凸顯人口老化問題，居家生活照顧就要仰賴機器人了，每個人眼睛一張開，就要學習與電腦機器人一起生活。所以，本書採用最新的程式語言 Arduino，因爲 Arduino 不但可解決目前所有商用運算，也增加了許多 I/O 接點，讓您不只可以處理商用問題，也可以直接管控所有工業與生活週邊設備，達到物聯網的境界（不只人與人能聯繫，人與物，物與物也要能聯繫）。

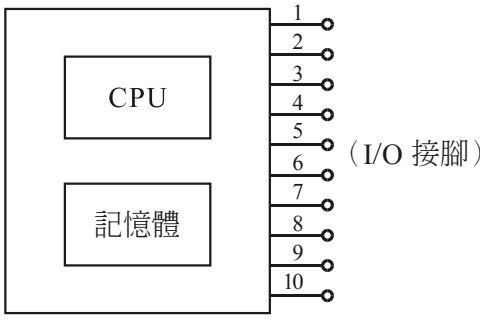
## 微電腦 VS 單晶片

現在幾乎家家戶戶都有桌上型微電腦或筆電，所謂微電腦或筆電就是整合 CPU、記憶單元、儲存裝置（硬碟與光碟等）、輸出入單元（鍵盤、滑鼠、螢幕）等零件，並協調其一起工作的設備，如下圖左。但是單晶片就神奇了，它竟然可以將 CPU、記憶單元整合在一個 IC 裡面，然後預留若干 I/O 接腳如下圖右，這些 I/O 接腳可以讓使用者，以軟體

程式的方式，設定低電位或高電位，進而控制所有負載的 ON 與 OFF。傳統的自動控制，都要使用硬體接線，所以體積龐大。但是，使用單晶片控制就神奇了，不但體積小、價格低，控制也非常彈性，所以現在幾乎所有家電都使用單晶片控制了。以傳統電鍋爲例，它是利用雙金屬片當作開關，所以傳統電鍋僅有單一功能，那就是僅能煮飯；電子鍋的功能就多了，它因爲使用單晶片當作控制，所以可以依照穀物的不同、或要吃白飯或稀飯而調整炊煮時間；冷氣機、洗衣機的功能也很多，冷氣機可設定恆溫、洗衣機可依衣物多寡自動調整水量與時間；汽車裡的單晶數量就更可觀了，所有安全配備與自動駕駛技術都有單晶聯合控制。機器人也是要有很多光、熱、距離、色彩等感測器，這樣才能做出決策，進行下一步動作，這也要藉助細小單晶片的協助，才能發揮機器人功效。所以，目前單晶片的普及率已經非常高了。




微電腦或筆電模組




單晶片模組

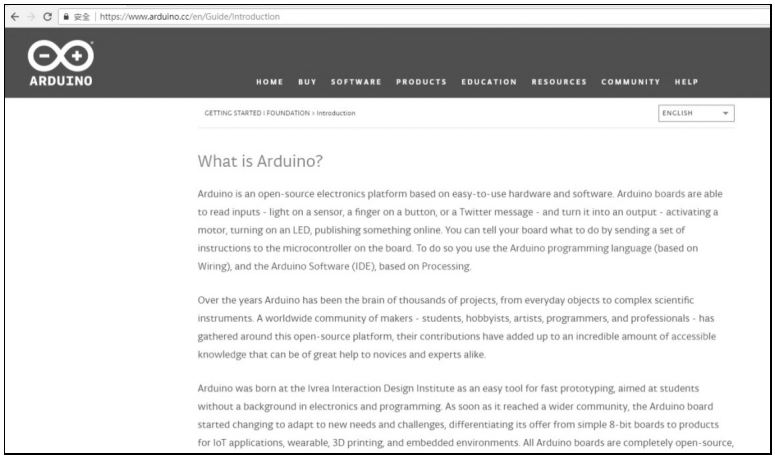
## 什麼是 Arduino

Arduino 其實就是一顆新開發的微小單晶片，那爲什麼如此地快速

竄紅呢？請看以下說明。請開啓 Arduino 官網(https://www.arduino.cc/)，畫面如下：


請點選上圖『WHAT IS ARDUINO』，畫面如下：


這個畫面有三個大的選項，分別是『WHAT IS ARDUINO』、『ARDUION BOARD』與『ARDUION SOFTWARE』，先展開第一個選項『WHAT IS ARDUION』，按一下『Learn more about Arduino』，畫面如下圖。



以上畫面說明 Arduino 產品的緣由，大致上是說它在 2003 年由義大利的 Ivrea 互動設計學院所研發，主要是希望不懂電子、不懂程式設計的人，也能非常簡單、非常快速且以非常少量的金錢開發出眞實世界的科技產品。它提供一塊造價非常便宜（不超過 50 美金）且體積很小的微控板，讓您能接收一些感測器，且做出一些判斷，就能設計出很多且有互動效果的產品。例如，機器人、3D 印表機、多軸飛行器、物聯網裝置等等。

將以上畫面往下捲，畫面如下圖，此畫面說明了 Arduino 的特色如下，共有五點，以下畫面寫得非常淺顯、清楚易懂，有不懂的單字，請複製，並貼到翻譯軟體就行。

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

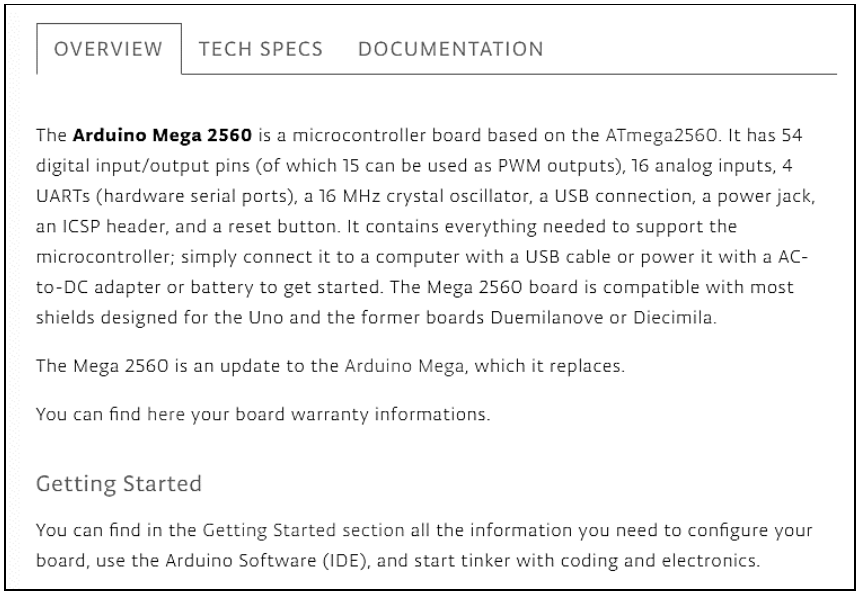
筆者將以上特色，已經以序言的方式呈現，請讀者瀏覽。

## 晶片微控板

一個單晶片 IC，再配合電源、穩壓、過載防護、傳輸、燒錄及拉出 I/O 接腳，稱爲單晶微控板，簡稱微控板，也有的書翻譯成單板電

腦，意思是說，它是單塊電路板的電腦。繼續按一下第二選項的『Discover the official Arduino boards』，畫面如下圖。


由上圖可知其微控板型號可說非常多，有入門款、加強款、物聯網款、教育總包套件款、穿戴式輕薄款、停產退役款等，這麼多款各有其用途，初學者當然眼花繚亂，請先比較其價格、I/O 腳位數與體積就好。其次，目前市面上書籍大都以 UNO 爲對象，但本書選用 MEGA 2560，我的主要理由是，MEGA 2560 的 I/O 腳位數目是 UNO 的三倍多（MEGA 是 54 隻/UNO 是 14 隻），但價格卻不到 UNO 的兩倍。I/O 腳位數目多，那掃描電路就會簡潔，例如，本書的四位數七段顯示器、8×8 點陣 LED，就比別書的軟硬體都還簡潔。而且每一個實驗都有專用腳位，作完不用拆，最後要做專題時，可以讓這些腳位同時連接多種 I/O 感測器與多種輸出設備同時動作。下圖是 MEGA 2560 微控板的總論，請隨意瀏覽，知道它有 54 隻 I/O 接腳就好。



OVERVIEW | TECH SPECS | DOCUMENTATION

The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

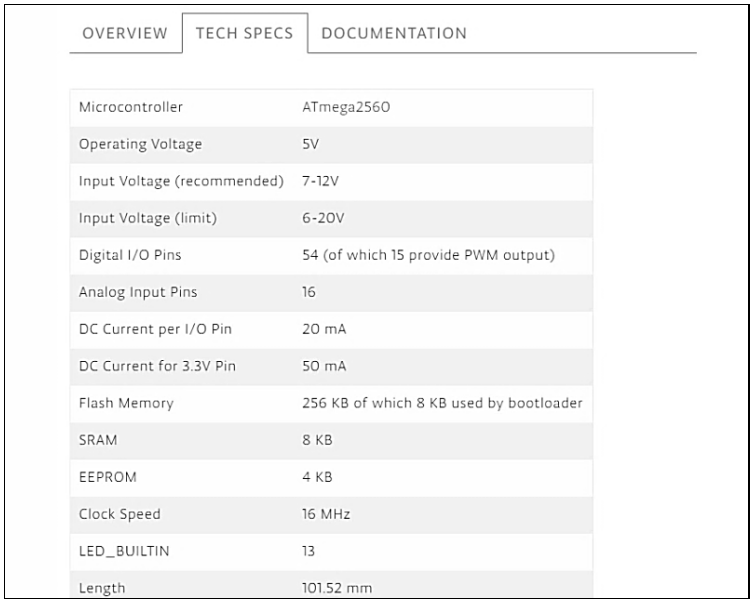
The Mega 2560 is an update to the Arduino Mega, which it replaces.

You can find here your board warranty informations.

Getting Started

You can find in the Getting Started section all the information you need to configure your board, use the Arduino Software (IDE), and start tinker with coding and electronics.

下圖是其技術規格，看看瀏覽就好，大意是說它其工作電壓是 5V，有 54 隻 I/O 接腳，反正記憶體和腳位是多到初學者用不完。

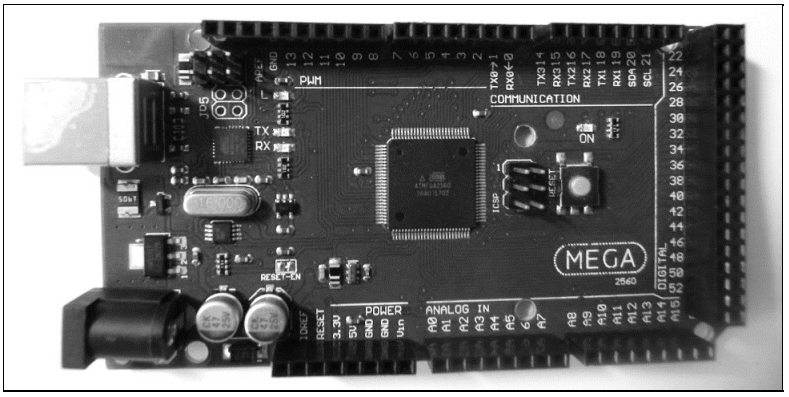


OVERVIEW | TECH SPECS | DOCUMENTATION

| | |
|---|---|
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Length | 101.52 mm |

下圖是 MEGA 2560 微控板的實體照片，ATMAGA 2560 IC 放在中間，這顆 IC 稱爲單晶片，旁邊的方形洞洞都是連接這顆 IC 的 I/O 腳位，這些腳位就可以讓使用者外接輸入或輸出元件，然後使用軟體程式，讀取、或指派這些腳位的電壓。

腳位編號請您配合下圖微控板，上面左邊編號是 0～13，上面右邊是 14～21，右邊雙排是 22～53，但 23,25,27,29 剛好缺角沒印出。以上都是數位接腳，電壓只有兩種情況，0 或 5V；下面則是 A0~A15，這些稱爲類比腳位，它可以模擬輸出 0～5V 電壓，或輸入 0～5V 電壓。

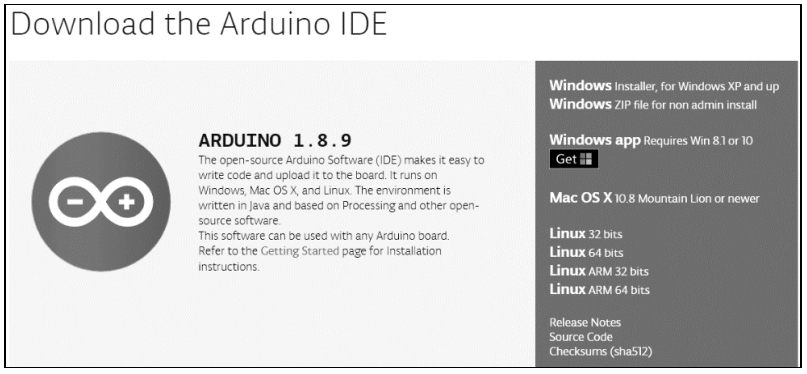



下表是微控板腳位與其暫存器名稱與對照表。每個腳位，除了有編號外，還有名稱，也就是可用編號或名稱使用這些腳位。例如，腳位 29～22 這 8 隻腳，其名稱爲 PORTA。往後我們可以用腳位名稱或腳位編號使用這些腳位。就如同每個學生有座號，也有名字，可以用座號叫人，也可以用名字叫人。

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PORTA | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 |
| PORTB | 13 | 12 | 11 | 10 | 50 | 51 | 52 | 53 |
| PORTC | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| PORTD | 38 | | | | 18 | 19 | 20 | 21 |
| PORTE | | | 3 | 2 | 5 | | 1 | 0 |
| PORTF | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| PORTG | | | 4 | | | 39 | 40 | 41 |
| PORTH | | 9 | 8 | 7 | 6 | | 16 | 17 |
| PORTI | | | | | | | | |
| PORTJ | 未接 | 未接 | 未接 | 未接 | 未接 | | 14 | 15 |
| PORTK | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 |
| PORTL | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 |

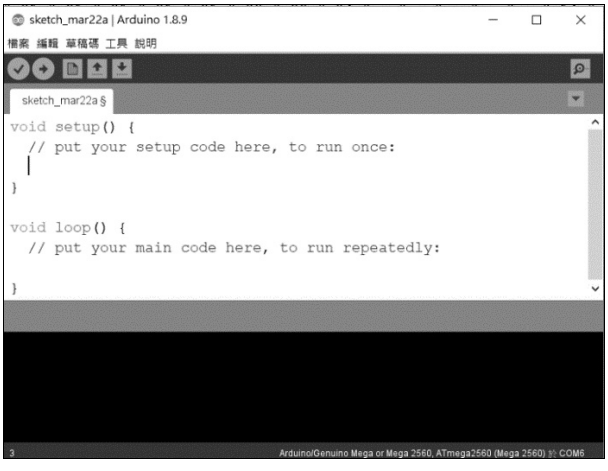
## Arduino 軟體下載與安裝

所有傳統的單晶片，都要自備類似記事本的編輯器，編寫程式、存檔、離開，然後使用其所提供的編譯程式，若有錯誤則要繼續開啓記事本修改，再編譯，直到完全正確。然後還要購買萬元燒錄器，將程式燒到單晶片。但是 Arduino 就神了，竟然有免費整合編輯視窗，整合以上編輯、編譯、燒錄（上傳）於單一視窗，此稱爲整合編輯程式（IDE, Integrated Development Environment 的縮寫）。請於 Arduino 官網點選『SOFTWARE/DOWNLOADS』，畫面如下，請點選『Windows installer for windows xp and up』，即可下載最新執行檔（*.exe）。接著，請開啓檔案總管，至下載區按兩下該執行檔，即可安裝。



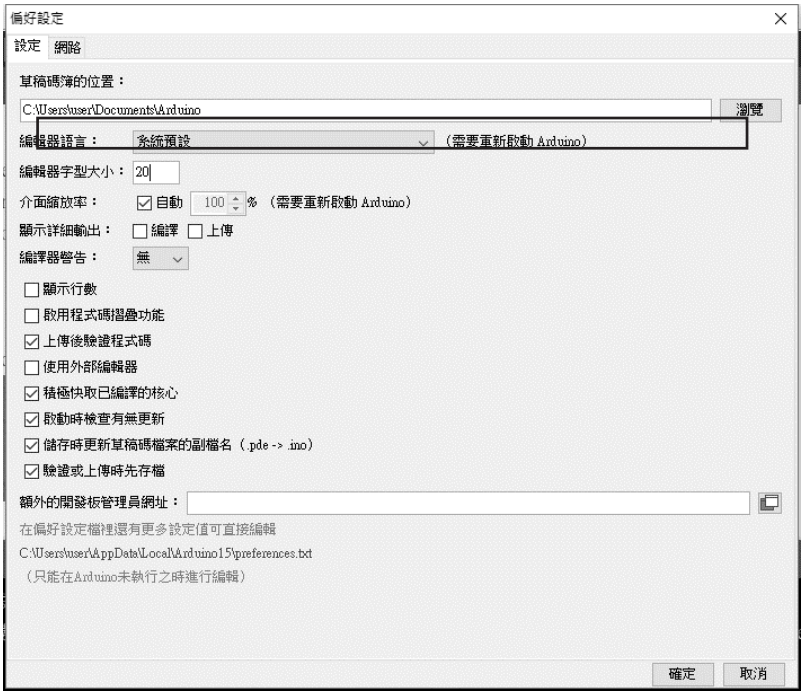

### ➲ IDE 畫面

下圖是開啓 Arduino IDE 畫面。

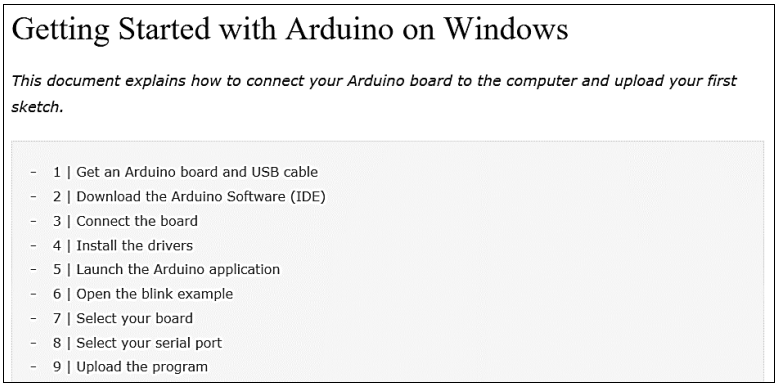


## ➲ 功能表的語言設定

理論上安裝後，功能表的語言會依據作業系統的語言自動完成設定，但若不是您所要的語言，請於功能表點選『檔案/偏好設定』，如下圖，即可點選您要的語言。




下圖是點選功能表的『說明/入門手冊』，畫面如下圖：




我們將上圖 Arduino 的單晶微控板使用步驟說明如下：

## ➲ 插入微控板

請依照指示，將微控板 USB 插頭插入電腦 USB 插座。請留意微控板右上角電源指示燈是否亮起。

## ➲ 點選開發板型號

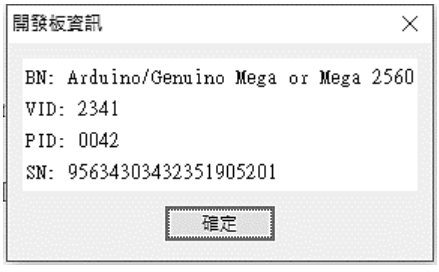
請點選功能表的『工具/開發板資訊』即可點選您所使用的微控板型號。(請依照您的微控板型號點選，筆者是點選『Arduino/Genuino Mega or Mega 2560』)

## ➲ 點選通訊埠編號

請點選功能表的『工具/序列埠』即可點選您所使用的序列埠編號（備註：系統會出現可用編號讓您點選)。

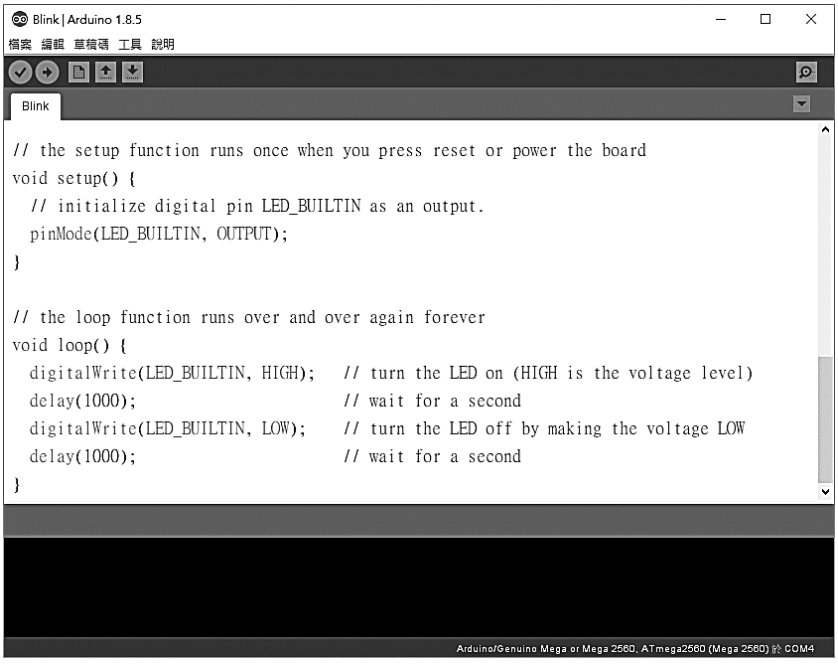
## ➲ 取得開發版資訊

請點選功能表的『工具/開發版資訊』即可取得您的微控板資訊。若出現下圖，才表示以上設定就緒，才能上傳程式。(實驗的中途，若改插入同學的微控板，那也要重覆以上步驟，直到看到下圖，才表示有抓到微控板，才能上傳程式。)




## ➲ 開啓測試程式

下圖是開啓測試程式 Blink（請點選功能表的『檔案/開啓』，預設路徑是 C:/ProgramFiles(x86)/Arduino/examples/01.Basics/Blink)，其功能是直接使用微控板預植的 LED（不管什麼板 UNO、MEGA…)，都是腳位 13，以常數『LED_BUILTIN』表示)，並令其閃爍明滅各一秒。




## ➲ 驗證程式

按一下工具列的『驗證』按鈕  ，即可編譯程式。

## ➲ 上傳程式

按一下工具列的『上傳』按鈕  ，即可上傳程式到微控板（上傳又稱爲燒錄）。上傳結束將會自動執行程式，請觀察微控板上所預植 LED 是否明滅（此顆 LED 就在腳位 13 旁）。

## ➲ 補充說明

1. setup()函式僅在程式一開始時執行一次。所以通常放置執行程式的初始設定、或程式執行後只執行一次的指令。
2. loop()函式則會重複不斷的被執行，所以就放置一些需要反覆執行的指令。
3. delay()函式是程式延遲程式，單位是 ms，所以延遲 1000ms，等於延遲 1 秒。因爲指派 LED 亮，總要停留一點時間，使用者才有機會看到。

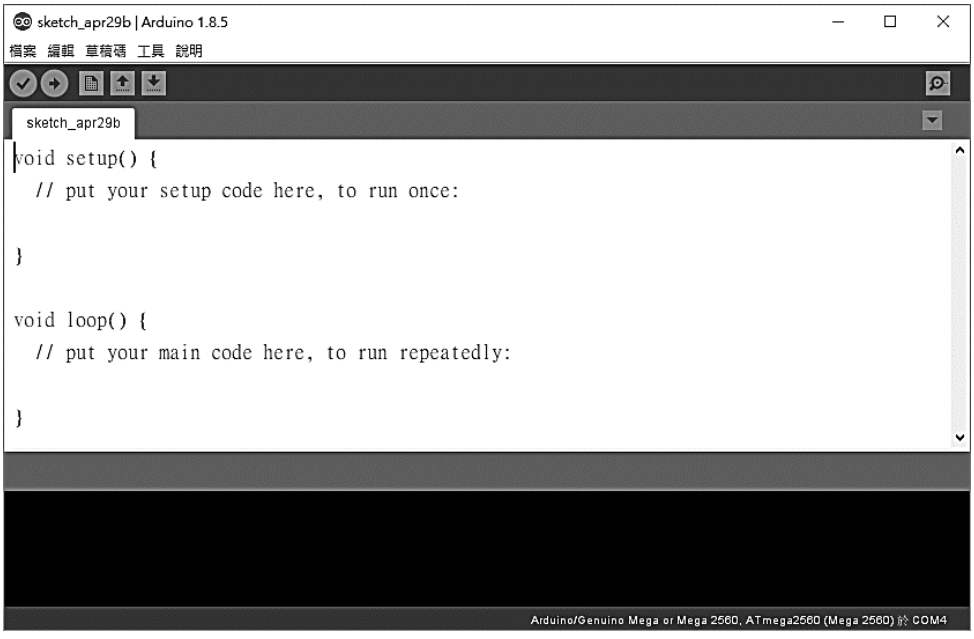
4. 雙斜線『//』稱爲註解，僅給人看，電腦不予編譯，沒有鍵入也沒關係。

**▶範例 1a**

請設計一程式，讓內建 LED 明亮 1.5 秒，滅掉 0.5 秒。

## ➲ 操作步驟

**1** 開新檔案。開新檔案，出現程式樣版如下：(點選功能表的『檔案／新增』或工具列的『新增』圖項。



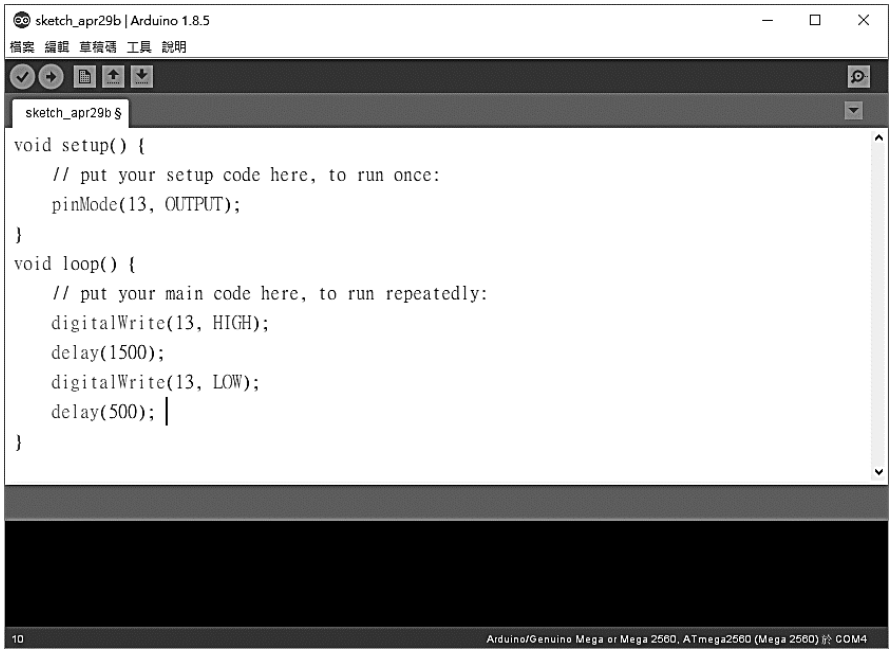

**2** 請將以下程式填入 setup(){}大括號裡面。請留意大小寫要一致，打完 pinMode，pinMode 會自動變紅色字體；打完 OUTPUT，OUTPUT 會自動變綠色字體，若沒自動變色，那表示您拚字錯誤。還有，分號（;）是半形的，也不能漏打。

```
    pinMode(13, OUTPUT);
```

**3** 請將以下程式填入 loop(){}大括號裡面。

```
    digitalWrite(13, HIGH);
    delay(1500);//單位是 ms,表示延遲 1.5s
    digitalWrite(13, LOW);
    delay(500);
```

**4** 完成後，如下圖。




**5** 存檔。

(1) 點選功能表『檔案/另存新檔』，並自行設定路徑。本書預設路徑是 d:\ad，且再以各章名稱分類，例如，ch01, ch02 等。

(2) 檔名僅輸入主檔名就好，副檔名預設爲『ino』，請不要輸入。

(3) 輸入檔名後，系統會自動以檔名建立資料夾。例如，檔名若爲 blink，那會先自動建立 blink 資料夾，再以 blink.ino 爲檔名，儲存在 blink 資料夾內。

**6** 驗證。(按一下工具列的『驗證』按鈕  ，即可編譯程式。若有錯誤，請自行修改)

**7** 上傳。按一下工具列的『上傳』按鈕  ，即可上傳程式到微控板(上傳又稱爲燒錄)。上傳結束將會自動執行程式，請觀察微控板上所預植 LED 是否明滅(此顆 LED 就在腳位 13 旁)。

## ➲ 自我練習

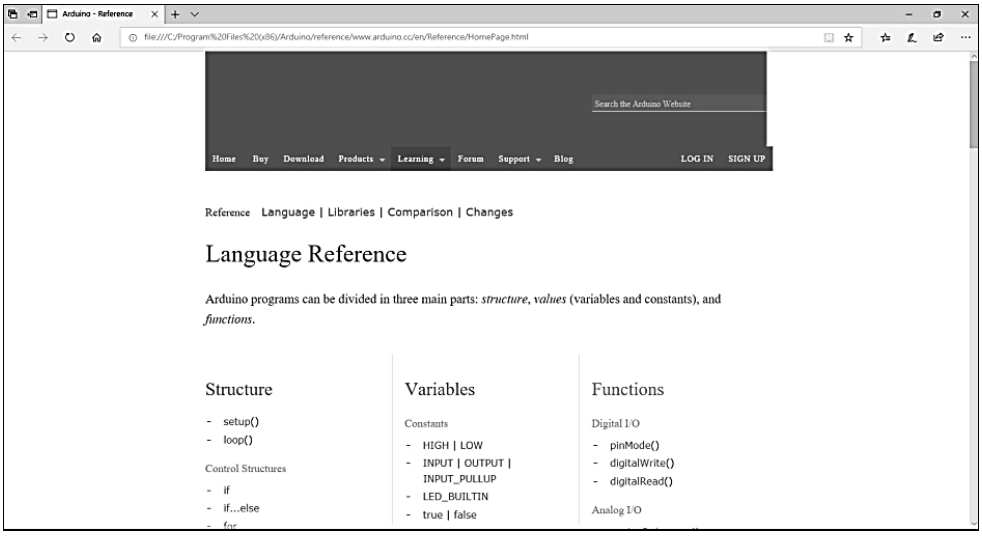
1. 請鍵入以下程式，並觀察執行結果。(請比較程式放在 setup()與 loop()的不同)

```
void setup() {
   pinMode(13, OUTPUT);
   digitalWrite(13, HIGH);
   delay(1500);
   digitalWrite(13, LOW);
   delay(500);
}
void loop() {}//絕不能省略
```

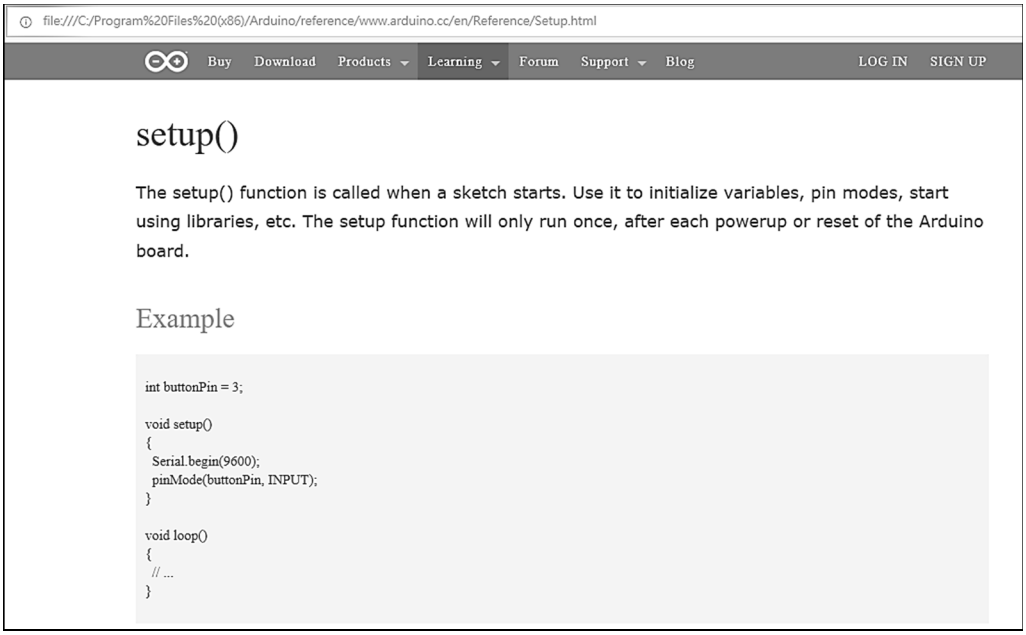
2. 假設有一個燈號亮 5 秒，接著以 0.5 秒閃爍 3 次，熄滅 3 秒，重複以上動作，請寫程式完成(請用預植編號 13 的 LED 即可)。

## ➲ 線上使用手冊

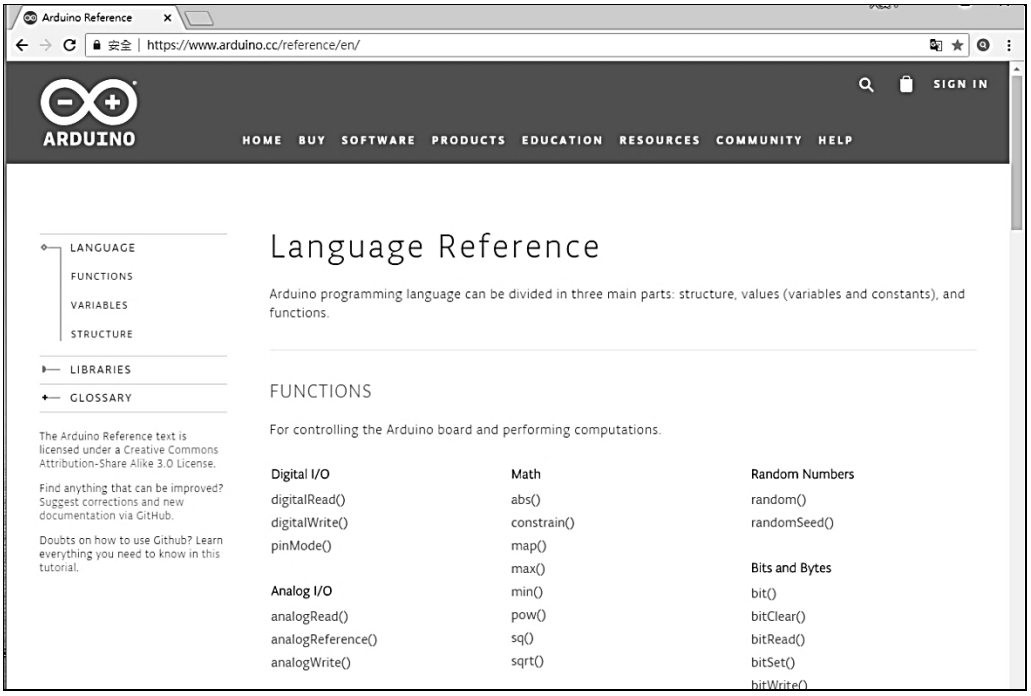
Arduino 的線上使用手冊以兩種方式存在，分別是安裝時就放到使用者硬碟的『說明/參考文件』與官網的『RESOURCE/REFERENCE』，兩者內容大致相同，但官網隨時更新，且我認爲較詳細。以下是開啓硬碟使用手冊的步驟：點選 Arduino IDE 功能表的『說明/參考文件』畫面如下圖，這就是 Arduino 所提供的軟體指令手冊，不論指令分類、指令解說、範例等都很詳細，請讀者自行探索。




於上圖點選 setup()，畫面如下圖，出現 setup()的解說。




下圖是點選 Arduino 官網的『RESOURCE/REFERENCE』的參考文件，此參考文件隨時更新，請讀者自行探索。

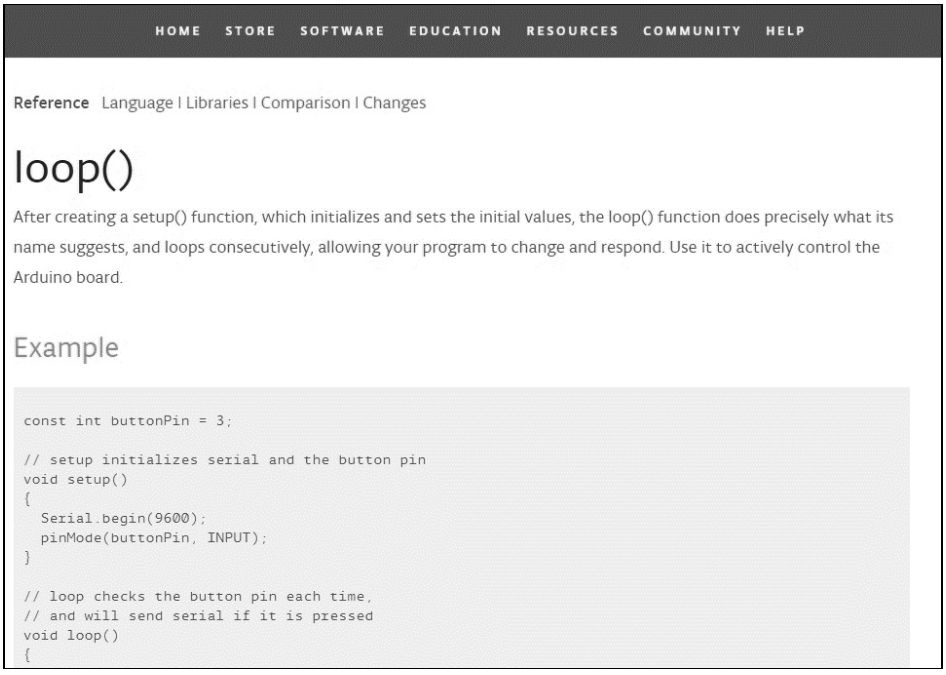



## ➲ 資料搜尋

於上圖按一下『放大鏡』圖項，畫面出現如下圖，可於下圖輸入您要查詢的關鍵字。例如，下圖我已經輸入『loop』：

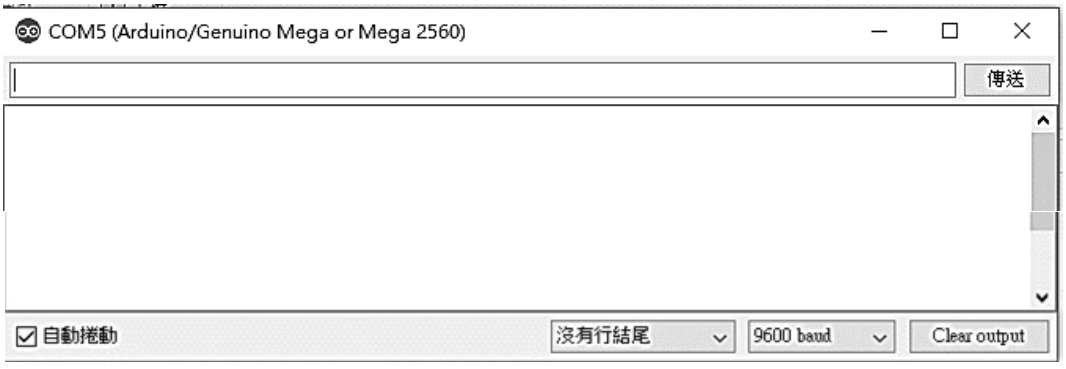

並按一下『Enter』，那就可以找出『loop』的說明，如下圖所：

繼續按一下上圖的『Arduino loop』，即可得到『loop』的說明，如下圖：



# 單元 2
# 程式的輸出入與序列埠監控視窗

人類號稱萬獸之王，眼睛、鼻子等五官與手腳是人類的輸出入設備，所有程式語言都是幫助人類處理事情的工具，當然也有輸出入設備。輸出入也是所有程式設計的第一步，本單元先介紹 Arduino 的序列埠監控視窗。序列埠監控視窗如下圖所示：




它可以讓我們用電腦的鍵盤與螢幕和微控板雙向溝通。此一功能是以往其他微控板所沒有的功能，因爲透過此序列埠監控視窗，不僅可以學習 Arduino 語言，學習 Arduino 所能完成的所有程式設計工作。例如，以下程式可以學習 Arduino 的運算子與數學函式。

```
Serial.println(4+2);
Serial.println(4>=2);
Serial.println(pow(2,3));
```

其次，於 Arduino 程式中，從感測器所讀到的值、或運算結果要從輸出元件輸出前，都可將結果先行輸出，這樣萬一程式不如預期，可以先縮小範圍，測試您的程式邏輯錯還是硬體接線錯。例如，以下程式，可以

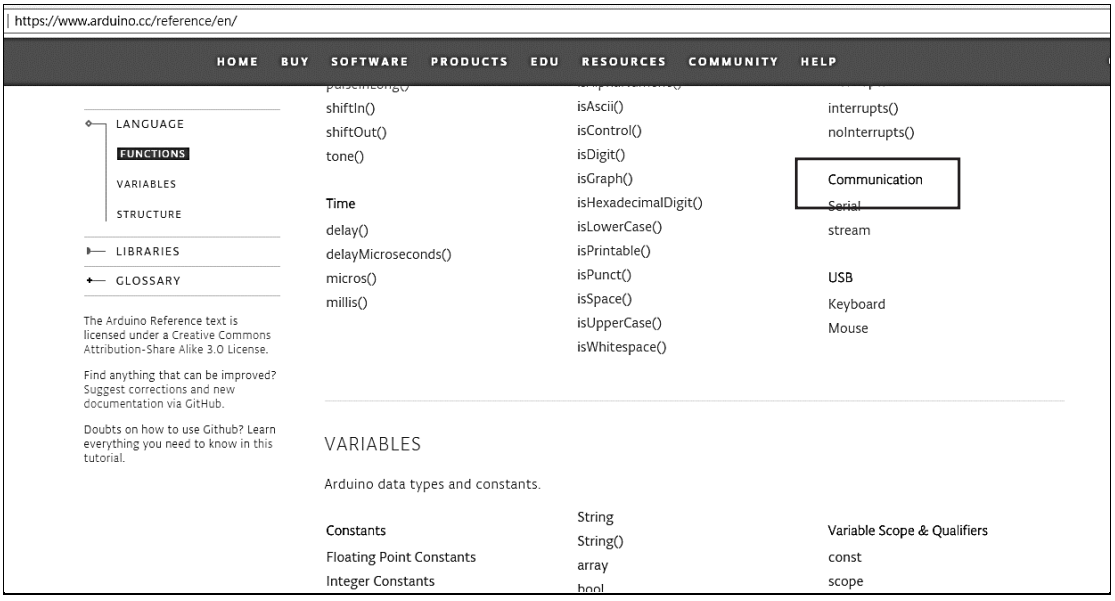
將指撥開關的輸入值，先行輸出於序列埠視窗，這樣可以確認此部分程式或硬體是否正確。

```
a=digitalRead(37);//讀取腳位 37 的電壓值
Serial.println(a);//於序列埠視窗輸出電壓值
```

以上功能分別介紹如下：

## Serial 物件

要讓微控板與電腦溝通，那就要透過 Serial 物件，請開啓 Arduino 參考文件，如下圖所示。




請於上圖點選『Serial』，就有 Serial 物件的所有方法，如下圖。這些方法可以讓我們啓用序列埠，然後進行輸出文數字、輸入字元、輸入字串與數值，分別說明如下。(補充說明：以下這些函式，在舊式函式導向程式設計稱爲『函式』，但在目前物件導向的程式設計領域，函式已經改稱爲『方法』)

Functions

If (Serial)
available()
availableForWrite()
begin()
end()
find()
findUntil()
flush()
parseFloat()
parseInt()
peek()
print()
println()
read()
readBytes()
readBytesUntil()
setTimeout()
write()
serialEvent()

## ➲ 啓用序列埠

要使用通訊埠，首先要使用 begin()方法啓用序列埠，先規定雙方傳輸速度，本例規定 9600，程式如下：

```
Serial.begin(9600);
```

## ➲ 輸出

要在序列埠輸出文數字，要用 print() 或 println() 方法，兩者的差別是 println()輸出後換行歸位，print()則沒有。例如，以下程式可輸出『GwoshengGwosheng』。

```
Serial.print("Gwosheng");
Serial.print("Gwosheng");
```

以下程式可輸出如右圖兩列 Gwosheng。

| | |
|---|---|
| `Serial.println("Gwosheng");`<br>`Serial.println("Gwosheng");` | `Gwosheng`<br>`Gwosheng` |

若是 print 或 println 方法內接變數，則會轉印出此變數的內容。例如，以下程式，可將變數所代表的內容輸出至電腦螢幕。(a,b 這些代號稱爲

變數，請看第十單元）

```
int a=3;//規定資料的儲存空間，請看第十單元
String b="Gwosheng";//規定資料的儲存空間，請看第十單元
Serial.println(a);//3
Serial.println(b);//Gwosheng
```

以下程式可輸出『a=3』。有加雙引號『""』的是字串，沒加雙引號的是變數，字串就直接輸出，變數就往前找此變數所儲存的值，並輸出。

```
Serial.print("a=");//字串就直接輸出 a=
Serial.print(a);//變數，輸出變數的值 3
```

**▶範例 2a**

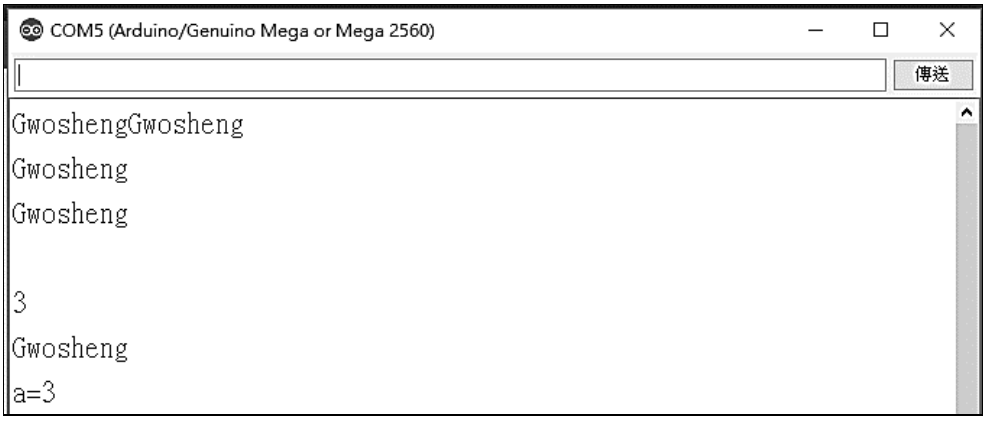
示範以上程式。

## ➲ 操作步驟

本例只要插入 Arduino 微控板就好，不用準備任何電路。

## ➲ 執行結果

程式『驗證』、『上傳』後，請開啓『序列埠監控視窗』（功能表的『工具/序列埠監控視窗』）。



## ➲ 程式列印

雙斜線『//』稱爲註解，僅給人看，電腦不予編譯，初學者可以不用打。

```
void setup() {
   // put your setup code here, to run once:
   Serial.begin(9600);
   Serial.print("Gwosheng");
   Serial.print("Gwosheng");
   Serial.println();//only linefeed
   Serial.println("Gwosheng");
   Serial.println("Gwosheng");
   Serial.println();//only linefeed
   int a=3;//規定資料的儲存空間，請看第十單元
   String b="Gwosheng";//規定資料的儲存空間，請看第十單元
   Serial.println(a);
   Serial.println(b);
   Serial.print("a=");
   Serial.print(a);
}void loop() {}//此函式雖然沒有放程式，但也不能刪除
```

## ➲ 自我練習

1. 請於序列埠監控視窗輸出自己的座號與名字，本例輸出座號後換行歸位，再輸出姓名。
2. 同上題，請在同一列輸出自己座號與名字。

## ➲ 輸入

Arduino 可以分別使用 read()、readString()及 parseInt()等方法讀取使用者所輸入字元、字串與數字，分別說明如下：

## ➲ 字元輸入

要讓序列埠輸入字元（僅一個英文字母或數字視爲字元），也是要同上範例，先使用 begin()啓動序列埠，再使用 read()方法讀入字元，程

式如下：

```
char c=Serial.read();//宣告 c 為 char 型態變數，請看第十單元
```

## ➲ available() 方法

大部分程式語言的鍵盤輸入，都會癡癡的等待使用者輸入，但是 Arduino 有特殊原因，它必須兼顧很多輸入設備，所以不會也不能等待，若特別需要它一定要原地等待使用者輸入，那就要使用 available() 方法，因爲 available()方法可判斷使用者是否已經輸入，available() 的語法如下：

Reference > Language > Functions > Communication > Serial > Available

### Serial.available()

Description

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes). `available()` inherits from the Stream utility class.

Syntax

```
Serial.available()
```

*Arduino Mega only:*

```
Serial1.available()
Serial2.available()
Serial3.available()
```

Parameters

None

Returns

The number of bytes available to read .

也就是 available()傳回大於等於『1』，才表示使用者有輸入，例如，以下程式，會請單晶原地等待您輸入，若鍵盤緩衝區沒有資料，就重複迴圈，也就是原地癡癡的等。while 稱爲迴圈指令，請看第 15 單元

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    while(Serial.available() ==0) {}//若未輸入，則一直在此等待。
    char c=Serial.read();
    Serial.println(c);
}
```

2. 請輸入以下程式（沒有使用 available()函式），並觀察執行結果

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    char c;
    c=Serial.read();
    Serial.println(c);
}
```

3. 以下左右程式也不同。請自行鍵入、執行，並觀察結果。下圖左是正確的，while 迴圈有加兩個{}，下圖右是錯誤的，c=Serial.read()會屬於 while 迴圈，因爲 while 沒有大括號，那此迴圈會包含與執行一個敘述，這都在第 15 單元詳細介紹。

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   char c;
   while(Serial.available() ==0) {}
   c =Serial.read();
   Serial.println(c);
   Serial.println("aa");
}
```

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   char c;
   while(Serial.available() ==0)
     c =Serial.read();
   Serial.println(c);
   Serial.println("aa");
}
```

## ➲ 字串輸入

readString()可讀取字串（一連串的字元稱爲字串），例如，以下程式可讀取與輸出字串。

```
void setup() {
   Serial.begin(9600);
}
void loop() {
   while(Serial.available() ==0) {}
   String c=Serial.readString();
   Serial.println(c);
}
```

## ➲ 自我練習

請輸入以上程式，並觀察執行結果。

## ➲ 數字輸入

Mega 2560 還可直接輸入數字，例如，以下程式可輸入一個整數。

```
   int a=Serial.parseInt();
```

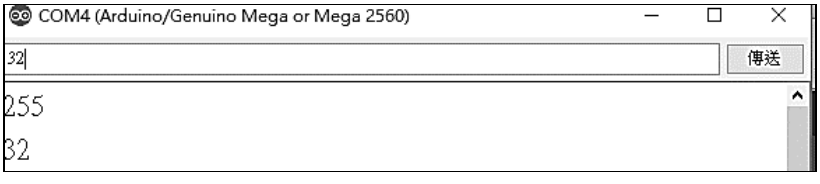
int 是變數的資料型態，表示 a 可儲存-32768～32767 的整數，將於第 10 單元進一步介紹。

**▶範例 2b**

示範以上程式。

## ➲ 輸出結果

請開啓序列埠監控視窗（功能表的『工具/序列埠監控視窗』）。



## ➲ 程式列印

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    int a;
    while(Serial.available() ==0) {}
    a=Serial.parseInt();
    Serial.println(a);
}
```

## ➲ 自我練習

1. 有了 Serial 鍵盤與螢幕輸出入方法，那也可把 Arduino 拿來學習 C 語言程式設計的工具，寫出 C 語言能做的程式。例如，請寫一程式，可以輸入兩個整數、計算其和，且輸出結果（提示：每輸入一個整數，就要一個 while 迴圈等待。其次，往後各章還有很多這方面的題目，可用來學習 C 語言）。

# 單元 3
# I/O 腳位探索

我們已經於第 1 單元介紹 Arduino Maga 2560 有 54 個 I/O 接腳，如下表。

| 暫存器\位元 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PORTA | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 |
| PORTB | 13 | 12 | 11 | 10 | 50 | 51 | 52 | 53 |
| PORTC | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| PORTD | 38 | | | | 18 | 19 | 20 | 21 |
| PORTE | | | 3 | 2 | 5 | | 1 | 0 |
| PORTF | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| PORTG | | | 4 | | | 39 | 40 | 41 |
| PORTH | | 9 | 8 | 7 | 6 | | 16 | 17 |
| PORTI | | | | | | | | |
| PORTJ | 未接 | 未接 | 未接 | 未接 | 未接 | | 14 | 15 |
| PORTK | A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 |
| PORTL | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 |

這些接腳都可以使用軟體的設定，分別指派當作數位輸出、數位輸入、或有上拉電阻 INPUT_PULL UP 的輸入等 3 種功能。使用這些腳位當作數位輸出入的步驟有二，分別是指派腳位功能與指派或讀取腳位電位。

## 指派腳位功能

Arduino 指派腳位功能有兩種方式，分別是單隻腳位的 pinMode 指派與 8 隻腳一起指派的 DDRA 指令（Data Direction of Port A 的縮寫）。例如，以下程式可使用 pinMode 指派腳位 13 作爲數位輸出。

```
pinMode(13, OUTPUT);
```

以下程式可指派腳位 29,28,27,26,25,24,23,22 作爲數位輸出。

```
pinMode(29, OUTPUT);
pinMode(28, OUTPUT);
pinMode(27, OUTPUT);
pinMode(26, OUTPUT);
pinMode(25, OUTPUT);
pinMode(24, OUTPUT);
pinMode(23, OUTPUT);
pinMode(22, OUTPUT);
```

因爲以上腳位 29～22 剛好是 PORTA，所以以上程式亦可簡化爲

```
DDRA=B11111111;// 1 是輸出，指派 PORTA 為輸出
```

則指派 PORTA 爲輸出。以上一次指派 8 個位元的功能，B 表示後續數字代表二進位，1 表示輸出，0 表示輸入。同理，若是

```
DDRA=B00000000;// 0 是輸入，指派 PORTA 為輸入
```

則指派 PORTA 爲輸入。也就是 pinMode 是配合腳位名稱，一次指派一個腳位的功能；DDRA 是配合暫存器名稱，一次指派 8 個腳位的功能，同理 PORTB、PORTC 就用 DDRB、DDRC 等指派其功能。

## ➲ 數位輸出

Arduino 腳位若當作數位輸出，使用手冊的說明如下：

> Properties of Pins Configured as OUTPUT
>
> Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. This means that they can provide a substantial amount of current to other circuits. Atmega pins can source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), or run many sensors, for example, but not enough current to run most relays, solenoids, or motors.
>
> Short circuits on Arduino pins, or attempting to run high current devices from them, can damage or destroy the output transistors in the pin, or damage the entire Atmega chip. Often this will result in a "dead" pin in the microcontroller but the remaining chip will still function adequately. For this reason it is a good idea to connect OUTPUT pins to other devices with 470Ω or 1k resistors, unless maximum current draw from the pins is required for a particular application.

大意是說，當輸出時，若設定爲 HIGH，則電壓有 5V（請留意有些微控板是 3.3V），且每隻腳位最大輸出電流是 20mA。其次，因爲 LED 只要 10mA 就很亮，所以驅動 LED 可說綽綽有餘，但是驅動 LED 電流若太大，LED 也會燒毀，所以要加上限流電阻如下：

$$\frac{5-1.7}{0.01} = 330\,\Omega$$

以上 1.7(V)稱爲 LED 的順向切入電壓，10mA=0.01A 這樣計算時，單位才一致。

## 指派電位

單晶片的優點是您可下指令指派任何接腳爲 HIGH 或 LOW。指派的方式有兩種，分別是單隻腳位的 digitalWrite 指派與八位元的暫存器名稱整體指派。例如，以下程式，您可指派接腳 22 爲 HIGH。

```
digitalWrite(22, HIGH);
```

以下程式，您可指派其爲 LOW。

```
digitalWrite(22, LOW);
```

所有腳位都有一個暫存器名稱，還可使用暫存器名稱一起指派 8 隻腳的電位，例如，以下程式可快速指派 PORTA 的 8 個位元全爲 HIGH，

PORTA 是暫存器名稱，Arduino 暫存器名稱請看本單元開頭的表格。

```
    PORTA=B11111111;
```

以下程式可快速指派 PORTA 的 8 個位元全爲 LOW。

```
    PORTA=0;//0 就 0，當然不用再指派任何進位。
```

**▶範例 3a**

PORTA 腳位探索實習。(PORTA 共有 8 隻腳位，本書往後簡稱 PA)

**1** 請鍵入以下程式，驗證、上傳。

```
void setup() {
  // put your setup code here, to run once:
    DDRA=B11111111;// 指派 PA 為輸出
    PORTA=B11111111;
}void loop() {}
```

**2** 請用三用電表，檢驗微控板的腳位 22, 23, 24, 25, 26, 27, 28, 29 是否爲高電位 5V。

**3** 若沒三用電表，請依照附錄 A，焊接一個 LED 電位筆，然後負端先插入 Gnd，正端就可當作電位探測筆，LED 亮就代表有電壓 5V。

**4** 鍵入以下程式，重新驗證、上傳，再檢驗微控板的腳位 22, 23, 24, 25, 26, 27, 28, 29 是否爲低電位 0V。

```
void setup() {
    DDRA=B11111111;// 指派 PA 為輸出
    PORTA=0;
}void loop() {}
```

## ➲ 自我練習

1. 請鍵入以下程式，並驗證 PA、PB、PC 對應腳位是否爲高電位。

```
void setup() {
  // put your setup code here, to run once:
    DDRA=B11111111;//指派 PA 為輸出
    PORTA=B11111111;//設定 PA0～7 均為高電位
    DDRB=B11111111;//指派 PB 為輸出
    PORTB=B11111111;//設定 PB0～7 均為高電位
    DDRC=B11111111;
    PORTC=B11111111;
}
```

2. 請探索 PORTF、PORTK、PORTL 腳位在哪裡？並寫程式檢驗這些腳位電壓是否能隨著您的程式而變化。

# 單元 4
# 資料的數位化

電腦的最小記憶單元爲位元(bit)，每個位元僅能儲存 0 或 1，8 個位元稱爲一個位元組(Byte)，那資料如何儲存於電腦呢？答案就是要將資料數位化。

## 資料數位化方法

日常生活可見的資料爲數值、字元、字串、聲音、圖片、影片，數值再分爲整數、與實數，且還有正負整數與正負實數，以上資料數位化的方法各有不同，但最後都是將以上資料轉爲二進位的方式，才能儲存於電腦，以上資料的數位化方法，分別說明如下：

## 正整數

所有整數都要先指定資料的長度，本例先假設是 8 位元，其餘 16 位元或 32 位元也是相同原理。其次，還要假設僅儲存正數，還是同時儲存正負整數。首先，以儲存正數爲例，那其二進位編碼如下，共可表示 0 到 255 的正整數。

| 正數 | 二進位編碼 |
|---|---|
| 0 | 00000000 |
| 1 | 00000001 |
| 2 | 00000010 |
| 3 | 00000011 |
| 4 | 00000100 |
| 5 | 00000101 |
| 6 | 00000110 |
| 7 | 00000111 |

| 正數 | 二進位編碼 |
|---|---|
| 254 | 11111110 |
| 255 | 11111111 |

▶範例 4a

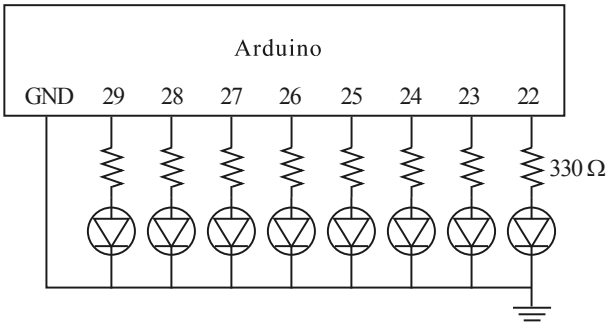
示範正整數的數位化。請鍵入以下程式，並觀察 8 個 LED 的亮滅，請問與上表的正數是否相符。

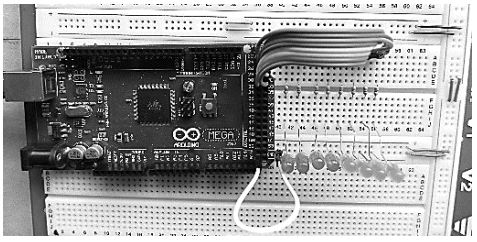
## ➲ 輸出結果

請留意 8 個 LED 代表 8 位元，燈號『亮』就是 1，燈號『滅』就是 0。

## ➲ 操作步驟

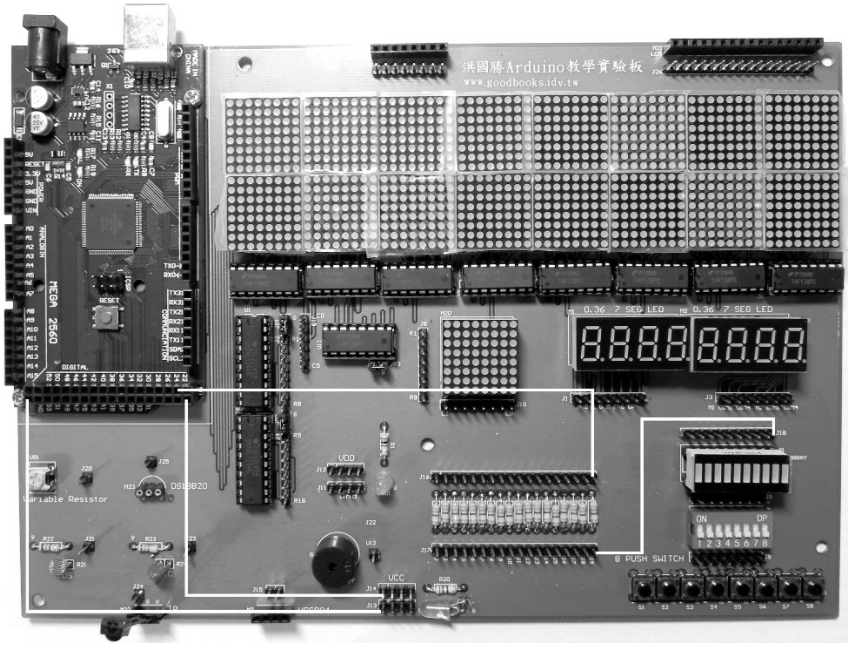
**1** 請準備 LED 電路如下圖左，若使用麵包板，則實體接線如下圖右：（電阻 330Ω顏色是橙橙棕，LED 長腳爲正，若腳被剪斷過，則往 LED 裡面瞧，接點小的爲正端，接點大的爲負端，下圖 LED 長腳請接電阻，短腳接地）



**2** 若使用本書實驗板，那只要將 22, 23, 24, 25, 26, 27, 28, 29 的腳位使用杜邦線連接到限流電阻(J16)，再從限流電阻另一端(J17)連接到 LED (J18)即可，Gnd 內部已自動連接，如下圖所示，下圖僅示範連接 3 條線，第 1 條線是微控板的 5V 接到實驗板的 Vcc，第 2 條線是微控板的 Gnd 接到實驗板的 Gnd，第 3 條線是微控板的腳位 22 先

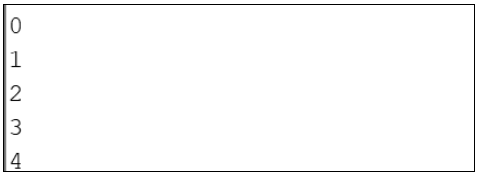
連到電阻，再從電阻另一端連接到 LED，其餘 23, 24, 25, 26, 27, 28, 29 接腳，請比照腳位 22 連接。在電源指示燈正常情況下，即可進行實驗。其次，因爲微控板有過載保護，若微控板電源指示燈熄了，實驗板指示燈也會熄滅，表示您的電路有問題，微控板自動切掉電源，此時請迅速拔掉電源連接線，檢查電路。(本書實驗板的 LED 是將 10 個 LED 連續排列，稱爲 LED Bar，印有型號的爲正端)




## ➲ 程式列印

```
void setup() {
  DDRA=B11111111;
  Serial.begin(9600);
}
unsigned char i=0;//宣告 i 為 8 位元正整數
void loop() {
  PORTA=i;
  Serial.println(i);
  delay(1000);//延遲與停留 1000ms,m=0.001，所以是 1 秒
  i=(i+1)%256;
}
```

執行結果 下圖是『序列埠監控視窗』的執行結果。




## ➲ 補充說明

1. i=(i+1)%256; 首先執行 i+1，表示 i 值先加 1，例如，i=4; 執行 i+1，那結果是 5；其次，『%』是取餘運算子，例如，i+1 結果是 256，那%256，結果是 0，也就是此運算可以保障 i 一定在 0～255 之間循環；『=』是指派運算子，最後將結果指派給 i。請留意『=』是指派運算子，表示『=』右邊先運算，再將結果指派回左邊的符號，此與數學的『相等』不一樣。所以請不要在此糾結它們怎會相等。

※負整數

若要考慮正負數，那最高位元則用來表示正或負數，最高位元爲 0 時表示正數，所以可表示的範圍是 0 到 127 如下頁表；最高位元爲 1 時表示負數，那到底可負多少？則要取 2 補數。所謂 2 補數是先取 1 補數再加 1，例如，

10000000

最高位元爲 1，表示負數，那到底負多少？先取 1 補數（0 變 1，1 變 0 稱爲 1 補數）

01111111

再加 1，所以是

10000000

那就是代表 $1*2^7$，結果是

-128

又例如，

11111111

最高位元是 1，代表負數，先取 1 補數

00000000

再加 1

00000001

所以是代表 -1，如下表：

| 二進位編碼 | 數字 |
|---|---|
| 00000000 | 0 |
| 00000001 | 1 |
| 00000010 | 2 |
| 00000011 | 3 |
| 00000100 | 4 |
| 00000101 | 5 |
| 00000110 | 6 |
| 00000111 | 7 |
| | |
| 01111110 | 126 |
| 01111111 | 127 最大正數 |
| | |
| 10000000 | -128 最小負數 |
| 10000001 | -127 |
| | |
| 11111110 | -2 |
| 11111111 | -1 |

由上表可知，若最高位元代表正負數，則 8 位元可表示的數值範圍爲-128～127。(備註：以上關於負數如何編碼，這是前人智慧的累積，是經過不斷討論與試驗的結果。因爲，若直接編碼，那結果如下：

| 二進位編碼 | 數字 |
|---|---|
| 10000000 | -0 |
| 10000001 | -1 |
| 10000010 | -1 |
| 11111110 | -126 |
| 11111111 | -127 |

這樣會有 0 與-0 的問題，重複了，所以計算機前輩就繼續腦力激盪，想出這種二補數的編碼，且沿用到今天。科學的發展，有時候也會革命，例如，早期也是認爲地球是平的，月亮與太陽繞我們運轉等等，請您也仔細琢磨與研究，說不定您也可以改變歷史）

**▶範例 4b**

示範包含正負整數的數位化。請鍵入以下程式，並觀察 8 個 LED 的亮滅，請問與上表的正負數是否相符？（電路同範例 4a）

## ➲ 輸出結果

下圖是『序列埠監控視窗』的執行結果。

```
-128
-127
-126
-125
```

1. 請留意數字 -128 的表示，燈號的明滅就是其二進位，本例會是 10000000。
2. 請留意數字 -1 的表示，燈號的明滅就是其二進位，本例會是 11111111。

## ➲ 程式列印

```
void setup() {
  DDRA=B11111111;
  Serial.begin(9600);
}
char i=-128; //宣告 i 為 8 位元正負整數,這樣 i 可表示-128~127
void loop() {
  PORTA=i;
  Serial.println((int)i);//轉為整數
  delay(1000);
  i=i+1;
}
```

## ➲ 補充說明

1. 本例 Serial.println((int)i);(int)爲將字元型態轉爲整數，請修改爲 Serial.println(i),並觀察執行結果。

## ➲ 自我練習

1. 請線上查詢 Arduino 資料型態 unsigned int 共使用多少 bit 代表 1 個整數，其所能代表的數字範圍爲何？其次，int 共使用多少 bit 代表 1 個整數，其所能代表的數字範圍爲何？

# 字元

鍵盤能用的字元有大小寫的 a,b,c、數字 0～9、還有一些控制字元，例如，跳列、歸位（回到該列最左邊）、跳頁、return 等，這些字元總共沒有超過 127 個，因此使用 1 個 byte 儲存就綽綽有餘，所以這些字元的編碼，在 1960 年就已經編碼完成，稱爲 ASCII 碼(American Standard Code for Information Interchange)，如下圖所示：也就是我們將所有字元編號，此編號與我們的座號相同，當叫到 65 號，那大家都能體認此爲字元『A』，當叫到 33 號，那就表示此爲字元『！』。

| DEC Value | Character | DEC Value | Character | DEC Value | Character | DEC Value | Character |
|---|---|---|---|---|---|---|---|
| 0 | null | 32 | space | 64 | @ | 96 | ` |
| 1 | | 33 | ! | 65 | A | 97 | a |
| 2 | | 34 | " | 66 | B | 98 | b |
| 3 | | 35 | # | 67 | C | 99 | c |
| 4 | | 36 | $ | 68 | D | 100 | d |
| 5 | | 37 | % | 69 | E | 101 | e |
| 6 | | 38 | & | 70 | F | 102 | f |
| 7 | | 39 | ' | 71 | G | 103 | g |
| 8 | | 40 | ( | 72 | H | 104 | h |
| 9 | tab | 41 | ) | 73 | I | 105 | i |
| 10 | line feed | 42 | * | 74 | J | 106 | j |
| 11 | | 43 | + | 75 | K | 107 | k |
| 12 | | 44 | , | 76 | L | 108 | l |
| 13 | carriage return | 45 | - | 77 | M | 109 | m |
| 14 | | 46 | . | 78 | N | 110 | n |
| 15 | | 47 | / | 79 | O | 111 | o |
| 16 | | 48 | 0 | 80 | P | 112 | p |
| 17 | | 49 | 1 | 81 | Q | 113 | q |
| 18 | | 50 | 2 | 82 | R | 114 | r |
| 19 | | 51 | 3 | 83 | S | 115 | s |
| 20 | | 52 | 4 | 84 | T | 116 | t |
| 21 | | 53 | 5 | 85 | U | 117 | u |
| 22 | | 54 | 6 | 86 | V | 118 | v |

由上圖可知共使用 7 位元，0 到 127，前面 32（0～31）爲控制字元，48～57 是 0～9 的數字，65～90 是大寫英文字元，97～122 是小寫英文字元。

**▶範例 4c**

示範輸出 ASCII 碼結果。請自行鍵入以下程式，並觀察燈號與輸出結果。（電路同範例 4a，請注意觀察，0～31 爲控制字元，沒有輸出，32 是空白也沒有輸出）

```
void setup() {
  DDRA=B11111111;
  Serial.begin(9600);
}
char i=0;//宣 i 為字元變數
void loop() {
  PORTA=i;
  Serial.print((int)i);Serial.print(":");Serial.println(i);
  delay(200);
  i=(i+1)%128;
}//請留意 print((int)i)與 println(i)輸出結果，前者是輸出整數，後者是輸出字元
```

**▶範例 4d**

示範英文小寫字元的表示。請自行鍵入以下程式，並觀察燈號與輸出結果。（電路同範例 4a）

## ➲ 輸出結果

1. 請留意字元'a'的表示，且內部是以$(97)_{10}$，燈號的明滅就是其二進位，本例會是 01100001。

```
void setup() {
  DDRA=B11111111;
  Serial.begin(9600);
}
```

```
char i;
void loop() {
  i='a';//字元用單引號
  Serial.print((int)i);Serial.print(":");Serial.println(i);
  PORTA=i;//97,0x61
  delay(1000);

  i='b';//字元用單引號
  Serial.print((int)i);Serial.print(":");Serial.println(i);
  PORTA=i;//98,0x62
  delay(1000);

  i='A';//字元用單引號
  Serial.print((int)i);Serial.print(":");Serial.println(i);
  PORTA=i;//65,0x41
  delay(2000);
}
```

## ➲ 自我練習

1. 請自行於 Arduino 線上手冊搜尋『ASCII』。
2. 請鍵入以下程式，將 8 個 LED 的明滅填入下表，請問此與 d 的 ASCII 是 0x64 或 B01100100 是否相符。0x 開頭表示此爲 16 進位，B 開頭表示此爲 2 進位，請看下一單元

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

```
void setup() {
  DDRA=B11111111;
  Serial.begin(9600);
  char i='d';
  Serial.print((int)i);Serial.print(":");Serial.println(i);
  PORTA=i;//
}void loop() {}
```

# 字串

連續字元的集合稱爲字串，例如，

```
"ABC"
```

請留意前面字元用單引號，本例字串則用雙引號。C/C++/Arduino 都是使用字元陣列表示字串。例如，以下陣列 a 即可儲存以上字串。

```
   char a[]="ABC";//字串用雙引號
```

**▶範例 4e**

示範字串的表示。請自行鍵入以下程式，並觀察燈號與輸出結果。（電路同範例 4a）

## ➲ 輸出結果

1. 請留意 a[0]是字元 'A'，編號是 65，內部記憶體是 01000001，燈號會是 01000001。
2. a[1]是字元 'B'，編號是 66，a[2]是字元 'C'，編號是 67。

```
65:A
66:B
67:C
```

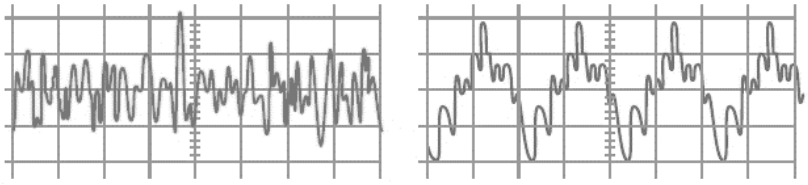
## ➲ 程式列印

```
void setup() {
  DDRA=B11111111;
  Serial.begin(9600);
}
void loop() {
  char a[]="ABC";//字串用雙引號
  Serial.print((int)a[0]);Serial.print(":");
  Serial.println(a[0]);
  PORTA=a[0];//65,0x41
  delay(1000);
  Serial.print((int)a[1]);Serial.print(":");
  Serial.println(a[1]);
  PORTA=a[1];//66,0x42
```

```
  delay(1000);
  Serial.print((int)a[2]);Serial.print(":");
  Serial.println(a[2]);
  PORTA=a[2];//67,0x43
  delay(12000);
}
```

## ※聲音

蚊子、蜜蜂飛舞、樹枝搖動等大自然的聲音、人類或樂器發出的聲音都是震動空氣所形成的聲波，如下圖所示，這些波形是一種類比信號，其頻率一直在變化。



聲音的數位化就是要將以上不斷變化的頻率取樣、儲存下來。要重新播放時，再使用電子元件震盪出這些頻率，推動蜂鳴器或喇叭的薄膜，那就可以將原音重現。要將以上不斷變化的頻率記錄下來，就要靠取樣與量化。分別說明如下：

### ➲ 取樣

取樣的原理就如同寓言裡的瞎子摸象，取樣的多寡，會決定其眞實度，例如，您只摸 1 次，摸到鼻子的就說大象像水管，摸到耳朵的人，就說大象像扇子…也就是取樣少，就容易失眞。所以若要接近原音，取樣頻率就要高，例如，CD 與 MP3 的取樣頻率是 44100Hz（1 秒取樣 44100 次，電腦這位瞎子很盡責，每秒摸了 44100 次或稱摸了 44100 個部位，且將這些大小或內容記錄下來），DVD 音質的取樣頻率是 96000Hz。

## ➲ 量化

每一個取樣要使用多少 bit 記錄，稱爲量化。例如，人類的身高大約 30 到 200cm，若只採用 1bit，那就只有將全部的身高分爲 2 個等級，若只採用 2bits，那就只有將全部的身高分爲 4 個等級，那身高的描述當然很粗糙，又例如，目前國中會考成績最後只分 5 等第，高中學測分爲 15 級分，這些都是量化的等級，CD 的量化採用 16bits，將聲音的頻率分爲 $2^{16}$ 個等級記錄下來，DVD 則用 24bits，也就是將聲音分爲 $2^{24}$ 個等級，那當然較接近原音，下表是常見不同用途的聲音的取樣頻率、量化等級與音軌數。

| 聲音用途 | 取樣頻率（Hz） | 量化等級（bit） | 音軌數 |
|---|---|---|---|
| 電話通訊 | 11,025 | 8 | 1 |
| 收音機廣播 | 22,050 | 8 | 1 |
| CD 音樂 | 44,100 | 16 | 2 |
| DVD 音樂 | 96,000 | 24 | 2 |

## ➲ 聲音檔案大小

聲音檔案大小的公式是：

取樣頻率(Hz)*量化等級(bit)*軌道數*秒數

例如，一個 2 秒 2 軌的 CD 檔案，其檔案大小是

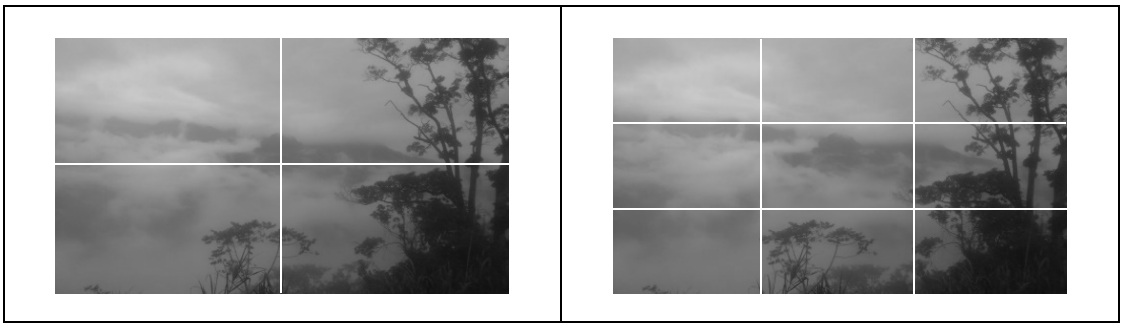
44100×16×2×2 = 2,822,400 bit
2,822,400 bit / 8 bit = 352,800 byte
352,800 byte / 1024byte = 344k byte

※

## ➲ 圖片或照片

圖片或照片要數位化是先規劃取樣的大小，取樣的大小就是像素(pixel)。例如下圖左是將照片規劃爲 2＊2 像素(一張照片只摸 4 個點)，下圖右是規劃爲 3＊3 像素，像素越多，當然所需摸的次數就多，所需

記憶體就多，重繪時較能原圖重現，但是耗費記憶體也多，傳輸也費時，所以就要依照圖片的用途，取一個平衡點。其次，還要規劃每一個像素的顏色要多少 bit 儲存，若使用 1bit，那再漂亮細膩的色彩也只剩 2 個等級了。若使用 2bit，那全部的色彩只剩 4 個等級；我們目前所稱的『全彩』就使用 24bits，那就有 $2^{24}$ = 16,777,216 個色彩變化。



又例如，下圖我將 1 張圖片規劃爲長與寬爲 8＊8 的像素，每一個像素只有 1bit，那就只能代表 0 與 1，且我將高位元規劃在下面，那 0x91, 0x4A, 0x80, 0x52, 0x3F, 0x32, 0x5F, 0x92 就可以代表一個『洪』，檔案大小就是 64bits。下圖 0x 開頭表示此爲 16 進位，請看下一單元。其次，後續單元，我們還會將此圖片使用字幕機顯示，這就能解釋資料如何儲存，資料如何重現。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0x91 | 0x4A | 0x80 | 0x52 | 0x3F | 0x32 | 0x5F | 0x92 |

範例 4f 示範 BMP 相片的檔案大小。

下圖我同時以小畫家與檔案總管開啓 gwosheng.bmp 的結果。表示這張照片寬度是 177 像素，高度是 221 像素，位元深度 24，就是全彩，且 BMP 檔案類型沒有資料壓縮，所以檔案大小是

177v221×24 = 938808 bits

938808/8 = 117351 bytes

117351/1024 = 114.6k bytes

※ 


### ➲ 自我練習

1. 請自行於檔案總管觀察任意*.bmp 檔案，並嘗試計算其檔案大小。

## 影像

因爲人類眼睛有視覺暫留現象，所以電視或電影都只是快速連續播放圖片或照片，您就覺得動作是連續的。影像的數位化原理也同聲音，要先定義取樣頻率，每一次取樣，就是取到 1 張照片，而且每張照片的數位化就同上面的『圖片或照片』數位化；播放時，還要先將這些資料先組成圖片，再連續播放這些圖片或照片，這就是電視或電影的播放原理了。若這些資料是放在網際網路，那網路也是串列的快速逐一傳輸這些 0 與 1，然後先組成圖片，再連續播放圖片，就形成您所看到的影片。