Chapter 2

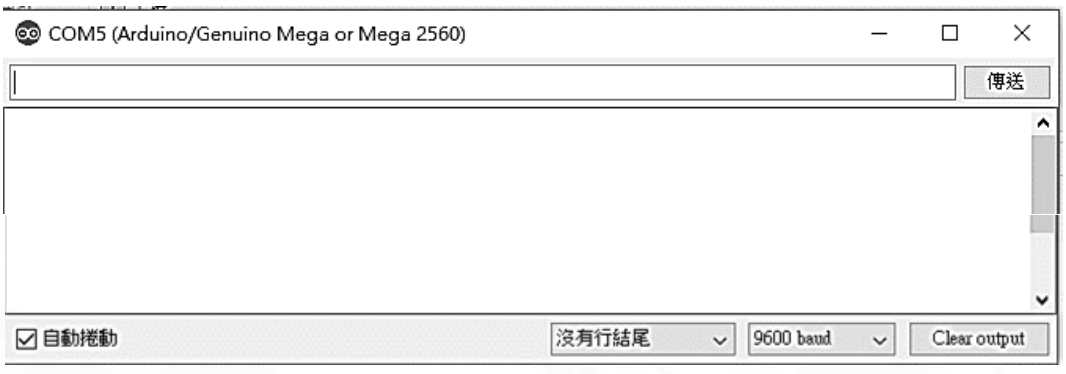
基本輸出入

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，本章先介紹 Arduino 的基本輸出入。

2-1 序列埠監控視窗

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));
```

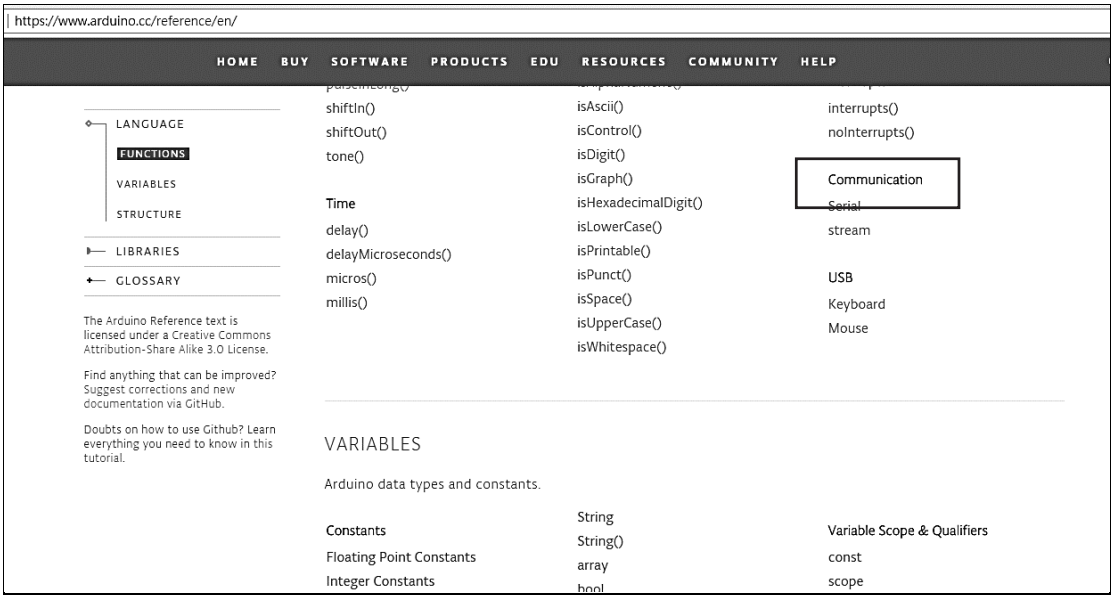
其次，於 Arduino 程式中，從感測器所讀到的值、或運算結果要從輸出元件輸出前，都可將結果先行輸出，這樣萬一程式不如預期，可以先縮小範圍，測試您的程式邏輯是否正確。例如，以下程式，可以將指撥開關的輸入值，先行輸出於序列埠數窗，這樣可以確認此部分程式或硬體是否正確。

```
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
```

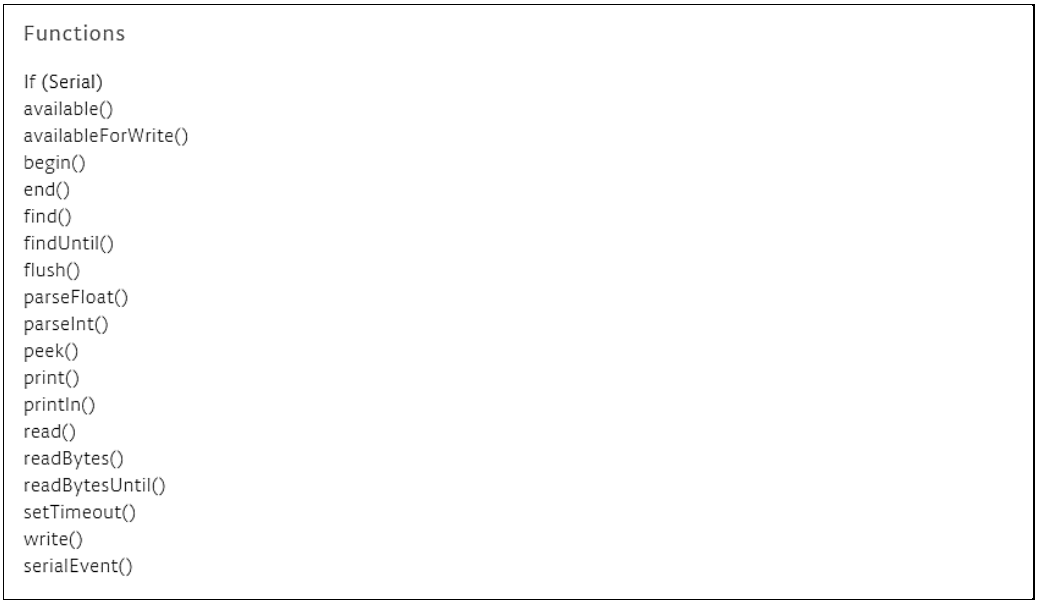
以上功能分別介紹如下：

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』）



➲ 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");` `Serial.println("Gwosheng");`	`Gwosheng` `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。

```
int a=3;//規定資料的儲存空間，請看第三章
String b="Gwosheng";//規定資料的儲存空間，請看第三章
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。

```
Serial.print("a=");//a=
Serial.print(a);//3
```

▶範例 2-1a

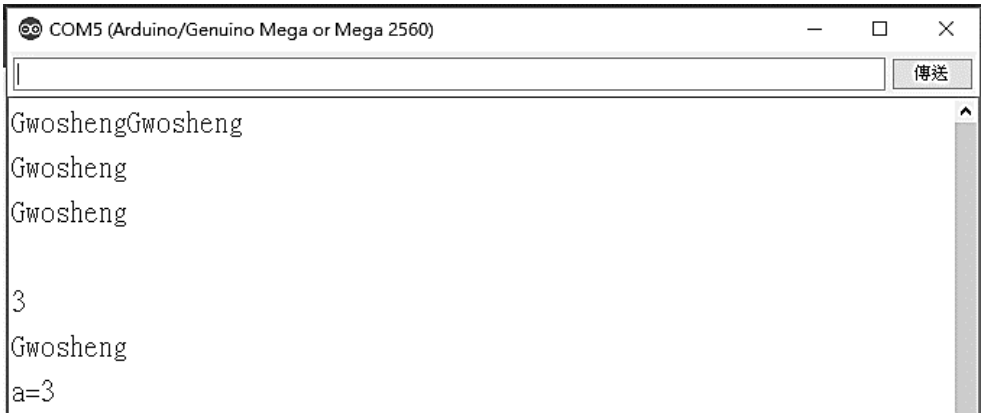
示範以上程式。

➲ 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，請看 4-2 節。

```
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，請看第三章
    String b="Gwosheng";//規定資料的儲存空間，請看第三章
    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），也是要同上單元，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();
```

➲ 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，此時才使用 read() 方法讀取，程式如下：（關於 if 指令用法，請看第五章）

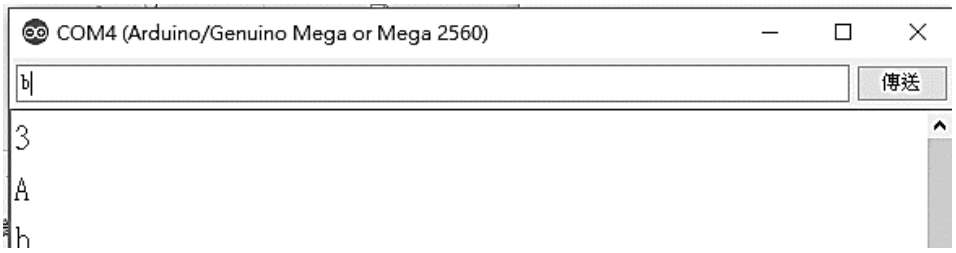
```
if(Serial.available() >=1){
     char c=Serial.read();
     Serial.println(c);
}
```

▶範例 2-1b

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() {
    // put your setup code here, to run once:
    Serial.begin(9600);
}
void loop() {
    // put your main code here, to run repeatedly:
    if(Serial.available() >=1){
      char c=Serial.read();
      Serial.println(c);
    }
}
```

以下程式，還會請單晶原地等待您輸入。while 請看第六章的迴圈。

```
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);
}
```

➲ 自我練習

1. 請將以上兩個程式最後一個大括號前加上一個指令 Serial.println("aa"); 如以下程式，並比較其執行結果的差異。

```
void setup() {
    Serial.begin(9600);
}
void loop() {
    char c;
    if(Serial.available() >=1){
        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");
}
```

由以上程式，您會發現 if 只是一個指令，瞬間就執行完畢，繼續下一指令，但 while(){}卻是一個迴圈，他會痴痴的在原地一直循環，一直等待，直到您輸入資料，請看第六章就會明瞭。

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，請看第六章。

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

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

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

```
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，例如，以下程式可讀取與輸出字串。(一連串的字元稱爲字串)

```
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（UNO 不行），例如，以下程式可輸入一個長整數。

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

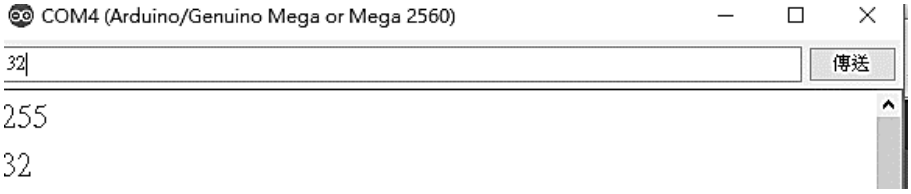
long 是變數的資料型態，將於第三章介紹。

▶範例 2-1c

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

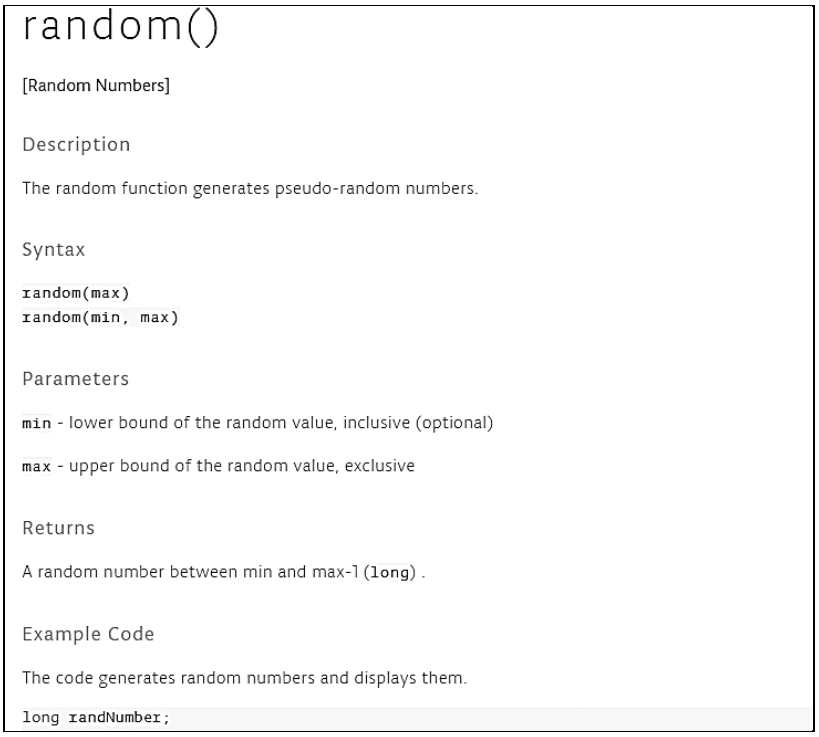
```
void setup() {
    Serial.begin(9600);
}
void loop() {
    long a;
    while(Serial.available() ==0) {}
    a=Serial.parseInt();
    Serial.println(a);
}
```

➲ 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）。

2-2 亂數

電腦常常需要模擬一些執行結果，例如，樂透開獎或擲骰子此時就需要亂數來協助，Arduino 產生亂數的方法是使用 random()函式，其語法如下圖：



random()

[Random Numbers]

Description

The random function generates pseudo-random numbers.

Syntax

```
random(max)
random(min, max)
```

Parameters

`min` - lower bound of the random value, inclusive (optional)

`max` - upper bound of the random value, exclusive

Returns

A random number between min and max-1 (`long`) .

Example Code

The code generates random numbers and displays them.

```
long randNumber;
```

以下程式可產生 10 到 19 的整數亂數。

```
int r = random(10, 20);//10..19
```

其中 min 若省略，則視爲 0，例如，以下程式可產生 0 到 299 的整數亂數。

```
int r = random(300);//0..299
```

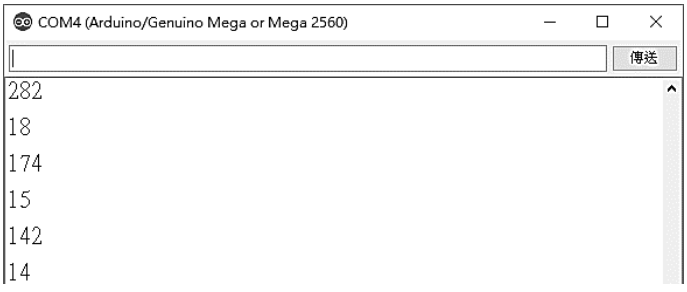
其次，因爲亂數的產生是一種查表的動作，爲了讓每次都能有不同的起始點，那就是要使用 randomSeed()函式。例如，以下程式將 A0 腳位空接，那就可隨機讀取一個雜訊，就將此雜訊作爲一個不同亂數起點的依據。

```
randomSeed(analogRead(0));//用 0 或 A0 都表示 A0 腳位
```

▶範例 2-2a

範例 2-2a 示範以上程式。

➲ 執行結果



➲ 程式列印

```
void setup(){
  Serial.begin(9600);
  // if analog input pin 0 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(0));
}
void loop() {
  // print a random number from 10 to 19
  r = random(10, 20);
  Serial.println(r);
  delay(1000);
  // print a random number from 0 to 299
  int r;
  r = random(300);
  Serial.println(r);
}
```

➲ 自我練習

1. 請將以上範例的 randomSeed(analogRead(0))去掉，並觀察執行結果，請留意每次執行程式時(『將序列埠監控視窗』關閉，再重新開啓序列埠視窗，即可重新執行程式)，其亂數順序是否都相同。

2. 有了 random()函式，就可完成擲骰子、樸克牌、樂透開獎等程式。例如，請寫一程式，可以模擬擲三顆骰子，並於序列埠監控視窗輸出結果。

2-3 數位輸出入

我們已經於 1-2 節介紹 Arduino Maga 2560 有 54 個 I/O 接腳，如下圖。

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							
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			
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 可用 pinMode 或 DDRA 指令指派腳位功能。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;// 指派 PORTA 為輸出
```

一次指派 8 個位元的功能，B 表示其爲二進位，DDRA 是 Data Direction of Port A 的縮寫。

➲ 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），且每支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：

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

1.7(V)稱爲 LED 的順向切入電壓。

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，分別是單支腳位的 digitalWrite 指派與八位元整體指派。例如，以下程式，您可指派接腳 22 爲 HIGH，LED 亮。。

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

以下程式，您可指派其爲 LOW，LED 不亮。

```
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 暫存器名稱請看本節開頭的表格。

```
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 2-3a

PORTA 腳位探索實習。(PORTA 共有 8 隻腳位，本書簡稱 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 是否為高電位 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 是否為低電位 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;
}
```

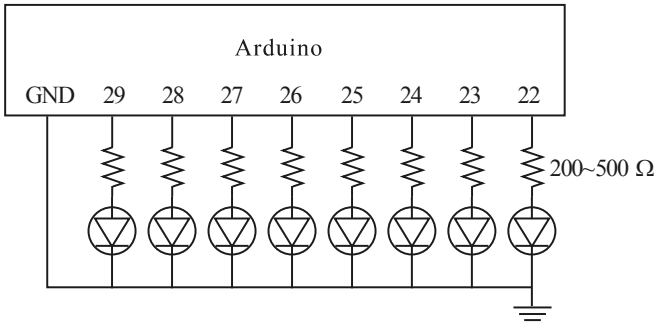
2. 請探索 PF、PK、PL 腳位在哪裡？並寫程式檢驗這些腳位電壓是否能隨著您的程式而變化。

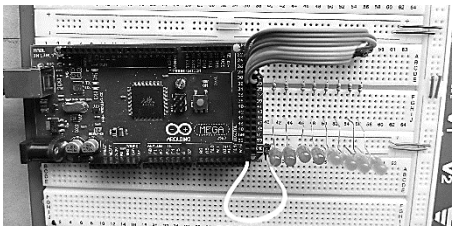
▶範例 2-3b

跑馬燈。

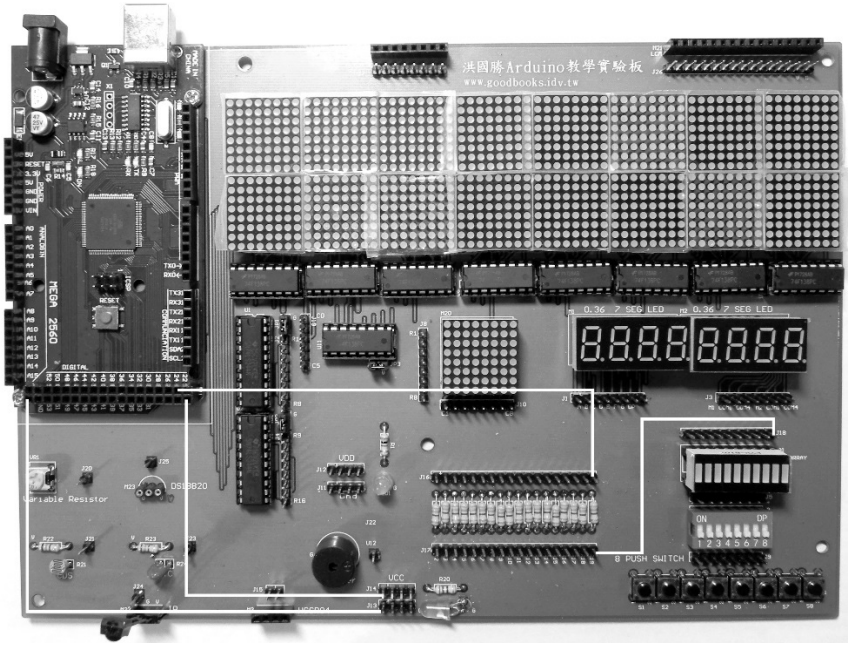
➲ 操作步驟

1 請準備 LED 電路如下圖左，麵包板實體接線如下圖右：





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，微控板的 5V 接到實驗板的 Vdd，微控板的 Gnd 接到實驗板的 Gnd，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。因為微控板有過載保護，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。



3 鍵入以下程式、驗證、上傳，並觀察執行結果，請留意 2 進位表示法。

```
void setup() {
  DDRA=B11111111;
}
```

```
void loop() {
  PORTA=B1; //二進位
  delay(1000);
  PORTA=0;  //0 就 0，任何進位都相同
  delay(1000);
  PORTA=B11;
  delay(1000);
  PORTA=B00100011;
  delay(1000);
  PORTA=3;  //十進位
  delay(1000);
  PORTA=4;  //十進位
  delay(1000);
}
```

4 鍵入以下程式、驗證、上傳，並觀察執行結果。（本程式請比較不同進位的數值）

```
void setup() {
  DDRA=B11111111;
}
void loop() {
  PORTA=0xFF;//0x 開頭代表 16 進位，15*16+15=255
  delay(1000);
  PORTA=0;
  delay(1000);
  PORTA=0x23;
  delay(1000);
  PORTA=0;
  delay(1000);
  PORTA=0101;//0 開頭代表八進位，1*8^2+1=65
  delay(1000);
  PORTA=65;
  delay(1000);
  PORTA=B1000001;
  delay(1000);
  PORTA=0;
  delay(1000);
}
```

5 以上是整體 8 位元的電位指派，以下則是單支腳位的電位指派。請鍵入

以下程式、驗證、上傳，並觀察執行結果。

```
void setup() {
   //單隻腳位指派腳位功能
   pinMode(29, OUTPUT);
   pinMode(28, OUTPUT);
   pinMode(27, OUTPUT);
   pinMode(26, OUTPUT);
   pinMode(25, OUTPUT);
   pinMode(24, OUTPUT);
   pinMode(23, OUTPUT);
   pinMode(22, OUTPUT);
}
void loop() {
   //單隻腳位指派電位
   digitalWrite(22, HIGH);
   delay(1000);
   digitalWrite(22, LOW);
   digitalWrite(23, HIGH);
   delay(1000);
   digitalWrite(23, LOW);
   digitalWrite(24, HIGH);
   delay(1000);
   digitalWrite(24, LOW);
   digitalWrite(25, HIGH);
   delay(1000);
}
```

6 以下程式請觀察 0～255 的二進位表示法：

```
void setup() {
  DDRA=B11111111;
}
byte i=0;
void loop() {
  i=(i+1)%256;
  PORTA=i;
  delay(500);
}
```

➲ 單晶與家電開關控制

前面的輸出都僅控制 LED 的輸出，那如何控制家電的電燈、冷氣、電視的開與關呢？因爲單晶的電壓信號非常小，無法直接讓電燈、冷氣、電視工作，那就要靠繼電器或**固態繼電器**（Solid State Relay，縮寫：SSR）等協助，才能啓動以上電器，請讀者自行深入研究探索以上元件。

➲ 自我練習

1. 同上電路，但是產生 0 到 255 亂數，依照亂數來點亮 LED。
2. 同上題，延遲時間也是亂數。

▶範例 2-3c

紅綠燈，假設有一紅綠燈時序如下，請寫一程式完成此功能。

時序	紅	黃	綠
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
5	0	1	0
6	0	1	0
7	0	0	1
8	0	0	1
9	0	0	1
10	0	0	1
11	0	0	1

➲ 操作步驟

1 請於腳位 24,23,22 配置紅、黃、綠等 LED。

2 時序 0,1,2,3,4，3 個 LED 輸出分別是 100，那就指派 a=B100，並輸出。

3 時序 5,6 輸出 010，那就指派 b=B010，並輸出，每個時序 1 秒，那 2 個時序 2 秒。

4 時序 7,8,9,10 輸出 001，那就指派 c=B001，並輸出，每個時序 1 秒，5 個時序就 5 秒。

➲ 程式列印

```
void setup() {
  DDRA=B11111111;
}
const byte a=B100;
const byte b=B010;
const byte c=B001;
void loop() {
    PORTA=a;
    delay(1000);
    PORTA=a;
    delay(1000);
    PORTA=a;
    delay(1000);
    PORTA=a;
    delay(1000);
    PORTA=a;
    delay(1000);
    PORTA=b;
    delay(2000);
    PORTA=c;
    delay(5000);
}
```

➲ 自我練習

1. 同範例，那如何讓綠燈熄滅前，先閃爍兩次。

2. 請寫一程式，控制 8 個 LED 的明滅順序如下：(每個時序 1 秒)

時序	L7	L6	L5	L4	L3	L2	L1	L0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	1
3	0	0	0	0	0	1	1	1
4	0	0	0	0	1	1	1	1
5	0	0	0	1	1	1	1	1
6	0	0	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1

3. 請寫一程式，控制 8 個 LED 的明滅順序如下：（每個時序 1 秒）

時序	L7	L6	L5	L4	L3	L2	L1	L0
1	1	1	1	1	1	1	1	1
2	0	1	1	1	1	1	1	0
3	0	0	1	1	1	1	0	0
4	0	0	0	1	1	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	1	0	1	0	0
7	0	0	1	0	0	0	1	0
8	0	1	0	0	0	0	0	1

4. 請觀察路口紅綠燈的時序，使用 6 個 LED，規劃一個雙向紅綠燈系統。（請留意要有雙向都紅燈的清空路口時序，再轉爲綠燈）
5. 請使用 8 個 LED，規劃一個雙向紅綠燈且有左轉燈的系統。

➲ 數位輸入

Arduino 當輸入時，有兩種可能，一種是接收其他 IC 的輸出，這時只要設定其爲 INPUT 即可，但若要接收按壓開關或指撥開關等開關裝置，則可設定其具有上拉電阻(INPUT_PULLUP)，這樣可以簡化外部電路，Arduino 說明文件如下圖。

Properties of Pins Configured as INPUT_PULLUP

There are 20K pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed by setting the pinMode() as INPUT_PULLUP. This effectively inverts the behavior of the INPUT mode, where HIGH means the sensor is off, and LOW means the sensor is on.

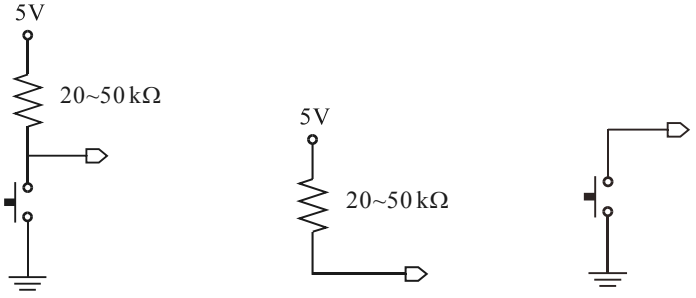
The value of this pullup depends on the microcontroller used. On most AVR-based boards, the value is guaranteed to be between 20kΩ and 50kΩ. On the Arduino Due, it is between 50kΩ and 150kΩ. For the exact value, consult the datasheet of the microcontroller on your board.

When connecting a sensor to a pin configured with INPUT_PULLUP, the other end should be connected to ground. In the case of a simple switch, this causes the pin to read HIGH when the switch is open, and LOW when the switch is pressed.

The pullup resistors provide enough current to dimly light an LED connected to a pin that has been configured as an input. If LEDs in a project seem to be working, but very dimly, this is likely what is going on.

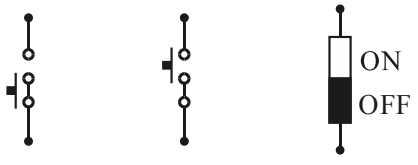
The pullup resistors are controlled by the same registers (internal chip memory locations) that control whether a pin is HIGH or LOW. Consequently, a pin that is configured to have pullup resistors turned on when the pin is an INPUT, will have the pin configured as HIGH if the pin is then switched to an OUTPUT with pinMode(). This works in the other direction as well, and an output pin that is left in a HIGH state will have the pullup resistors set if switched to an input with pinMode().

開關的標準電路如下圖左，按鍵沒按壓是高電位，押下去是低電位，這樣就可以判斷開關有沒有被按。其次，Arduino 爲了讓使用者簡化電路，此上拉電阻（下圖中）可用軟體指派，所以使用者只要接一個開關就好，如下圖右。（補充說明，沒上拉電阻可以嗎？當然不可以，因爲開關沒按壓時，Arduino 輸入腳位浮接（懸空未接稱爲浮接），此時就無法判定其電壓的高與低了。



➲ 指撥開關

指撥開關的內部結構如下圖，開關可上下滑動，下圖左是下滑，電路處於斷路，下圖中往上滑動，則電路接通，往後爲了簡化電路，均以下圖右表示指撥開關。

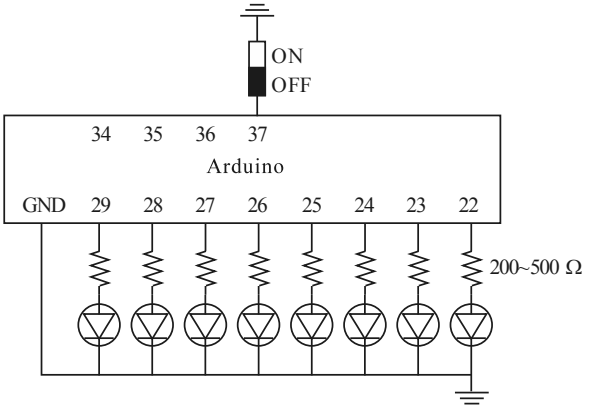


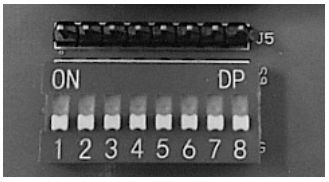
▶範例 2-3d

示範使用指撥開關。

➲ 操作步驟

1 完成以下電路，其中 LED 同上範例 2-3b，指撥開關的實驗板實體圖如下圖右，杜邦線從 J5 連接到 Arduino 微控板對應腳位即可，Gnd 內部已經連接。





2 PINC 可一次讀取 PORTC 暫存器 8 隻腳全部電壓，請鍵入以下程式，並觀察執行結果，請留意開關 ON，其值是 LOW，LED 滅掉。

```
void setup() {
    Serial.begin(9600);
    pinMode(37,INPUT_PULLUP);//PC0
    DDRA=B11111111;//29~22
    PORTA=0;
}
void loop() {
    byte a=PINC;//一次讀取 PC 的 8 個位元
    Serial.println(a);//於序列埠監控視窗輸出結果
    PORTA=a;//使用 PA 的 LED 輸出結果
}
```

3 digitalRead()一次僅讀取指定腳位電壓，請鍵入以下程式，並觀察執行結果，請留意開關 ON，其值是 LOW，LED 滅掉。

```
void setup() {
    Serial.begin(9600);
    pinMode(37,INPUT_PULLUP);//PC0
    DDRA=B11111111;
    PORTA=0;
}
void loop() {
    byte b=digitalRead(37);//讀一個位元
    Serial.println(b);
    digitalWrite(22,b);   //寫入一個位元
}
```

➲ 自我練習

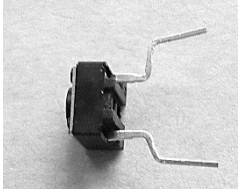
1. 請將以上程式的 INPUT_PULLUP 改爲 INPUT，程式如下，並觀察執行結果。

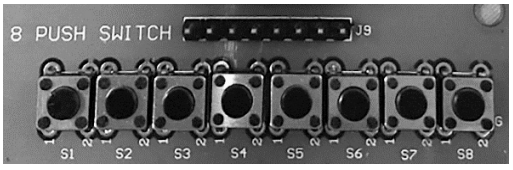
```
pinMode(37,INPUT);//PC0
```

2. 請用 8 位元指撥開關，控制 8 個 LED 的明滅，且 ON 時才亮。

➲ 按壓開關

前面的指撥開關是用手指滑動開關，控制 ON 與 OFF，但是按壓開關則是按下去時 ON，當您放開開關時，則有彈簧協助，又自動回到 OFF 狀態，按壓開關示意圖如下圖左，實體圖如下圖右。還有，有些按壓開關有四隻腳，有些則有兩隻腳，若是四隻腳，則內部兩兩相通，請用三用電表檢驗，以免使用到成雙連接那兩支，那就永遠是 ON，沒有按壓效果了。四隻腳的沒辦法插麵包板，若買到了，只好剪掉多餘的兩隻腳。還有也要買腳長一點的，這樣可將腳撐開，如下圖中，那才剛好可插到麵包板的地線，這樣佈線較簡單。下圖右則是實驗板實體圖，請連接 J9 到微控板即可，Gnd 內部已經連接。



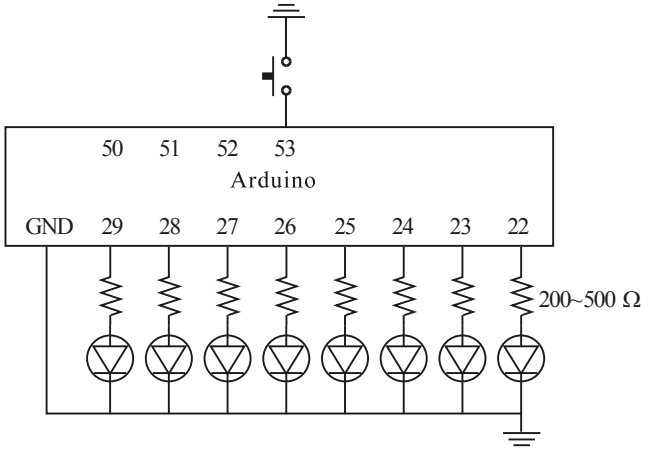


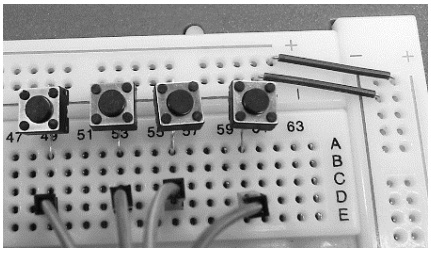
▶範例 2-3e

示範使用按壓開關。

➲ 操作步驟

1 完成以下電路，其中 LED 同上範例 2-3b、按壓開關麵包板接法如下圖右。





2 鍵入以下程式，並觀察執行結果，請留意開關 ON，其值是 LOW，LED 滅掉。

```
void setup() {
  pinMode(53,INPUT_PULLUP);//PB0
  DDRA=B11111111;//29~22
  PORTA=0;
  Serial.begin(9600);
}
void loop() {
    byte a=PINB;
    Serial.println(a);
    PORTA=a;
}
```

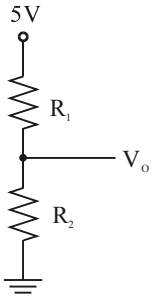
➲ 自我練習

1. 請用 digitalRead()與 digitalWrite()重作以上範例。
2. 請用 8 個按壓開關，分別控制 8 個 LED 的明滅，且按下去才亮。

2-4 類比輸出

前面的數位輸出，其輸出值只有 HIGH 與 LOW，HIGH 的時候是 5V，LOW 的時候是 0V。但是類比輸出卻可以「模擬」輸出 0 到 5 的任意電壓值。

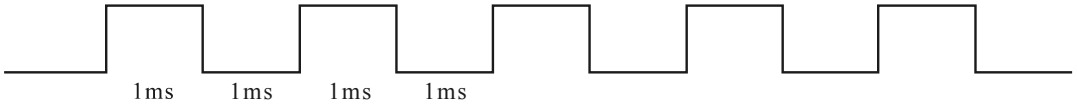
Arduino 的輸入電壓是 5V，那要如何得到其他電壓呢？調整輸出電壓的方式有兩種，一種是調整電阻，如下圖。只要改變 R1, R2 的比例，就可以改變輸出 Vo 的值。



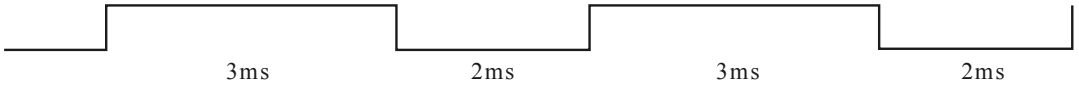
$$V_o = \frac{5}{R_1 + R_2} \times R_2$$

另一種是改變頻率，例如，我們若能以很快的方式，輪流輸出 5V 與 0V，如下圖一，那您用三用電表的 DC 檔量起來就會是 2.5V；同理，若輸出 5V 的時間調整爲 3ms，0V 時間調爲 2ms，如下圖二，那其直流輸出將會是 5*3/(3+2)=3，這種改變脈波寬度比例的方式稱爲脈寬調變（Pulse Width Modulation，簡稱 PWM 變頻）。脈寬調變還有一個優點，那就是省電，因爲傳統的電壓調整方式，R1 所消耗的功率完全沒有作用，它只是轉爲熱能浪費掉了。脈寬調變的理論很早就有，只是當時單晶價格太貴，現在單晶成本降低了，所以現在有很多變頻冰箱、冷氣、風扇等。而且變頻的方式控制冷氣溫度當然較舒適，因爲非變頻是溫度上升時，馬達啓動且全速運轉，溫度下降了，馬達就停止，溫度忽高忽低，當然不舒適，但是直流變頻就可以輸出一個任意直流電壓，讓馬達可以以任何速度持續運轉，所以溫度就能接近恆溫。

圖一



圖二

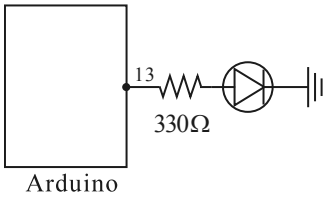


▶範例 2-4a

示範以脈寬調變的方式改變輸出電壓值。

➲ 操作步驟

1 完成以下電路。（本例使用 Arduino 預植 LED）



2 完成以下程式，並觀察 LED 的明亮程度。同理，若輸出接直流馬達，那馬達轉速就有變化。

```
byte a;
void setup() {
  pinMode (13,OUTPUT);
}
void loop() {
    a=255;
    analogWrite(13,a);//a 僅能從 255~0
    delay(1000);
    a=127;
    analogWrite(13,a);
    delay(1000);
    a=63;
    analogWrite(13,a);
    delay(1000);
    a=31;
    analogWrite(13,a);
    delay(1000);
}
```

請用三用電表量測腳位 13 的電壓。

➲ 補充說明

1. a 值僅能從 0 到 255，0 最小，255 最大。以 127 爲例，其輸出平均電壓

爲 2.5，其頻率則依使用腳位而不同，大部分是 490Hz，但若使用腳位 5,6 則其頻率爲 980Hz，Arduino 手冊說明如下所示。

> **analogWrite()**
>
> [Analog I/O]
>
> Description
>
> Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to `analogWrite()`, the pin will generate a steady square wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin. The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.
>
> On most Arduino boards (those with the ATmega168 or ATmega328P), this function works on pins 3, 5, 6, 9, 10, and 11. On the Arduino Mega, it works on pins 2 - 13 and 44 - 46. Older Arduino boards with an ATmega8 only support `analogWrite()` on pins 9, 10, and 11.
> The Arduino DUE supports `analogWrite()` on pins 2 through 13, plus pins DAC0 and DAC1. Unlike the PWM pins, DAC0 and DAC1 are Digital to Analog converters, and act as true analog outputs.
> You do not need to call `pinMode()` to set the pin as an output before calling `analogWrite()`.
> The `analogWrite` function has nothing to do with the analog pins or the `analogRead` function.

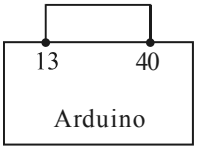
2. 類比輸出僅能使用腳位 0~13，但是腳位 0,1 還兼具序列埠傳輸。因爲電腦上傳程式給微控板時採用序列埠傳輸，所以請盡量不要用這兩隻腳，若您一定要用，那上傳程式時，要先將這兩支接腳的負載拔掉，等到傳輸完畢，再將電路接上。

▶範例 2-4b

同上範例，但使用序列繪圖家，觀察波形。

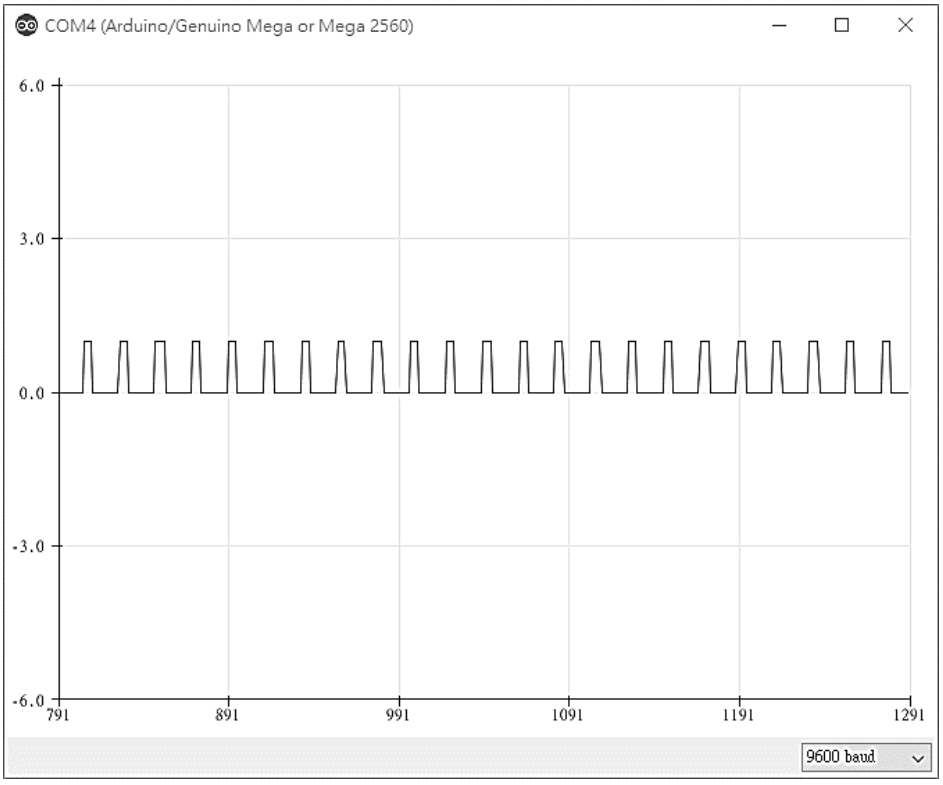
➲ 電路圖

麻煩從腳位 13 拉一導線，再插入腳位 40，如下圖。



➲ 執行結果

本例讀取腳位 40 的電壓，再交由序列繪圖家輸出，如下圖。



➲ 程式列印

```
byte a;
void setup() {
  Serial.begin(9600);
  pinMode (13,OUTPUT);
  pinMode (40,INPUT);
}
void loop() {
    a=63;
    analogWrite(13,a);
    byte b=digitalRead(40);
    Serial.println(b);
}
```

➲ 補充說明

本例 a=63，如上圖，高電位時間僅佔大約 1/4，請自行改變 a 的值，並觀察高電位與低電位的時間比值。

➲ 自我練習

1. 請設計一個電路，可以使用 8 位元指撥開關控制 1 個 LED 的明亮程度。
2. 請設計一個電路，可以使用 8 位元指撥開關控制 8 個 LED 的明滅個數。

2-5 類比輸入

前面的數位輸入僅能輸入高電位與低電位兩種狀態。但是類比輸入接腳，A0~A15 卻可輸入任意電壓值。請看以下範例說明。

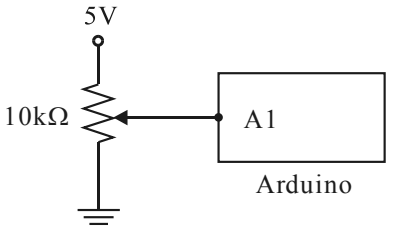
▶範例 2-5a

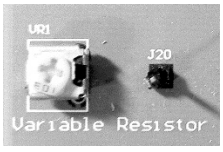
示範類比輸入。

➲ 操作步驟

1 完成以下電路。

任意拿一個 1kΩ 以上的可變電阻，第一隻腳接 5V，第三隻腳接地，第二隻腳輸出接到微控板 A0~A15 任意腳位，本例使用類比輸入腳位 A1，依照分壓定則，A1 腳位電壓將會在 0～5V 變化。下圖右則是實驗板可變電阻的實體圖，5V 與 Gnd 已經內部連接，使用者只要使用杜邦線將輸出(J20)連接到微控板腳位 A1 即可。





2 完成以下程式。

```
int a;
void setup() {
  Serial.begin(9600);
```

```
}

void loop() {
  a=analogRead(A1);//read A1 pin, a 可得 0 到 1023
  Serial.println(a);
  Serial.println(a/205);//輸出電壓在 0 到 5V
  delay(50);
}
```

3 因為 A1 接腳是一個 10 位元比較器，所以輸出值是 0～1023，請調整可變電阻值，並從序列埠視窗觀察 a 之值是否在 0～1023，將此值除以 205 就可得到其實際電壓 0～5V，可用三用電表測量 A1 腳位的電壓，是否相符。

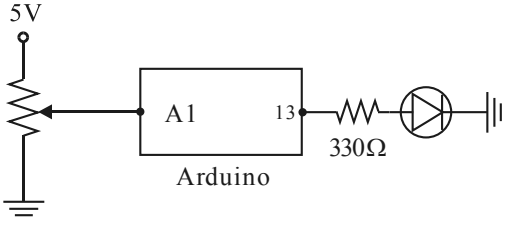
➲ 自我練習

1. 請設計一個電路，可以測量乾電池的電壓。

▶範例 2-5b

請設計一個電路，可以用可變電阻調整 LED 的明暗程度。

電路設計如右圖



➲ 程式列印

```
void setup() {
    Serial.begin(9600);
    pinMode(13,OUTPUT);
}
void loop() {
    int a;
    a=analogRead(A1);//從 A1 腳位讀取電壓，a 從 0 到 1023
    Serial.println(a);
```

```
    analogWrite(13,a/4);//將 a 值輸出，但 a 僅能 0~255,所以要除以 4，縮
小範圍
}
```

➲ 自我練習

1. 請設計一個電路，可以將可變電阻的輸出轉爲 0 到 255，並用 8 個 LED 輸出其值。
2. 於範例 2-3b 的跑馬燈程式 6 中，請用可變電阻調整其延遲時間，越是右轉則速度變快。

➲ 光敏電阻

光敏電阻（簡稱 CDS）是一種特殊電阻，當光線改變時，其電阻值會改變，依照分壓定則，如下圖，光敏電阻改變時，輸出電壓就會改變，此時我們只要測量其輸出電壓的改變，就可以推論其光線的明暗變化。

▶範例 2-5c

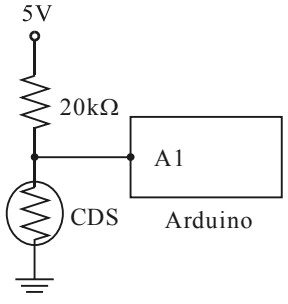
示範光敏電阻。

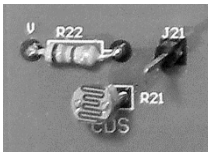
➲ 操作步驟

1 測量光敏電阻值，本例測得其電阻接近 20kΩ。

2 請用東西阻擋其光線，使用三用電錶歐姆檔，觀察其電阻值變化。本例發現光線變暗時，電阻值變大。

3 請使用相近的電阻與其串連，本例使用 20kΩ 電阻（紅黑澄）和它串連，如下圖左。下圖右是實驗板實體圖，使用者只要使用杜邦線連接排針(J21)到微控板腳位 A1 即可，5V 與 Gnd 內部已經連接。





4 請完成以下程式。

```
int a;
void setup() {
  Serial.begin(9600);
}
void loop() {
  a=analogRead(A1);//a 得到 0 到 1023
  Serial.println(a);Serial.println(a/204);
  delay(50);
}
```

5 請用東西遮擋光敏電阻，使其變暗，並以序列埠視窗觀察其值是否變大。

6 請將 a 除以 204 輸出，並用三用電表測量微控板 A1 接點電壓，請觀察此值與 a/204 是否相符？

➲ 自我練習

1. 請設計一個電路，可以用光敏電阻調整 LED 的明暗，天氣變暗，則 LED 變亮。
2. 請設計一個電路，可以將光敏電阻的輸出轉爲 0 到 255，並用 8 個 LED 輸出其值。

➲ 熱敏電阻

熱敏電阻是一種特殊電阻，當其周圍溫度變化時，其電阻值將會改變。我們也是依照分壓定則，如下圖，光敏電阻改變時，輸出電壓就會改變，此時我們只要測量其輸出電壓的改變，就可以推論其溫度的高低變化。

▶範例 2-5b

示範熱敏電阻。

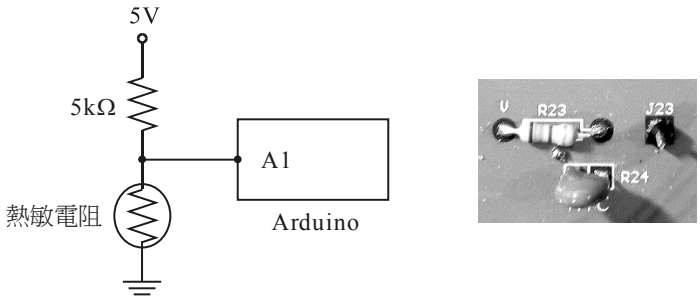
➲ 操作步驟

1 測量熱敏電阻值，本例測得接近 5kΩ。

2 請用吹風機改變熱敏電阻周圍溫度，使用三用電表歐姆檔，觀察其電阻值

變化。本例發現溫度變熱時，電阻值變大。

3 下圖右是實驗板實體圖，使用者只要使用杜邦線連接排針(J23)到微控板腳位 A1 即可，5V 與 Gnd 內部已經連接。



4 請完成以下程式。

```
int a;
void setup() {
  Serial.begin(9600);
}
void loop() {
  a=analogRead(A1);//a 從 0 到 1023
  Serial.println(a);
  delay(50);
}
```

5 請用吹風機吹熱敏電阻，改變熱敏電阻周圍溫度，並以序列埠視窗觀察其值是否變大。

6 請將 a 除以 204 輸出，並用三用電表測量微控板 A1 接點電壓，請觀察此值與 a/204 是否相符？

➲ 自我練習

1. 請設計一個電路，可以用熱敏電阻調整 LED 的明暗程度，天氣變熱，則 LED 變亮。(補充說明，若您有變頻直流馬達，那就是天氣熱了，馬達轉速變快)
2. 請設計一個電路，可以將熱敏電阻的輸出轉爲 0 到 255，並用 8 個 LED 輸出其值。

2-6 實例探討

廣告 LED 燈

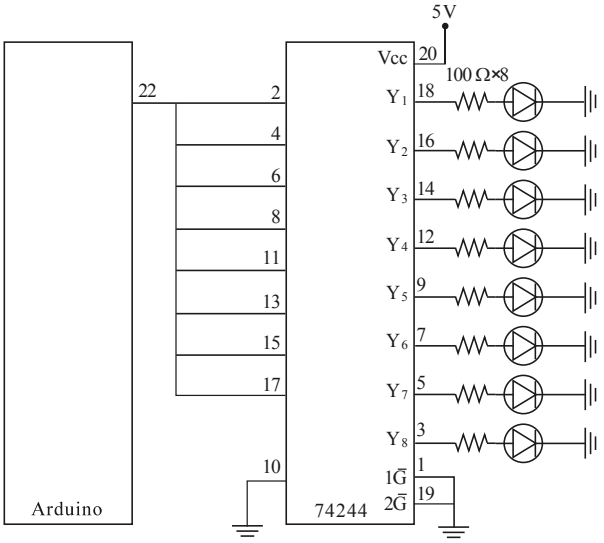
市面上有許多廣告燈，它是用 LED 排列出所要顯示的中英文字，但很可惜都不會自動明滅，本節就要教您如何使用 Arduino 控制他們的閃爍明滅。

▶範例 2-6a

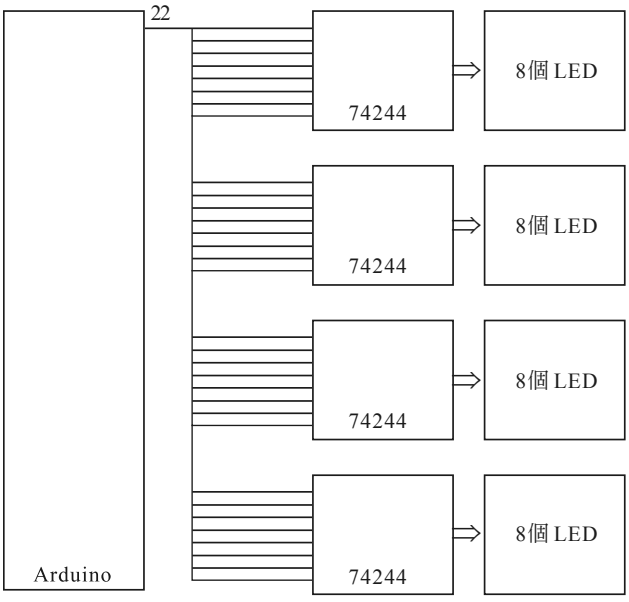
示範製作 LED 廣告燈。本例假設用 LED 排列出『泉勝出版』四個字。

➲ 電路圖

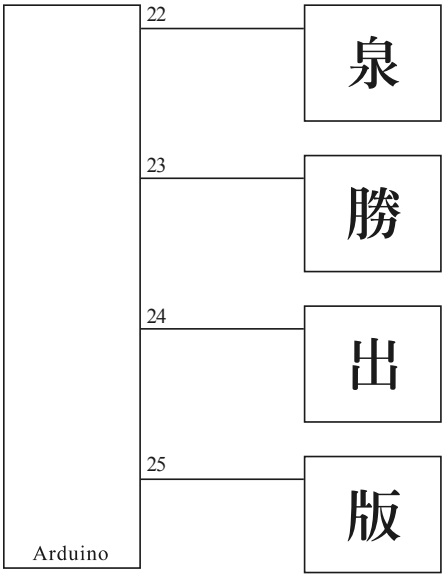
1. 電路如下圖，每個 LED 都至少要 10mA 才會亮，所以每個 LED 都要使用一個 74244 緩衝接腳驅動輸出，如下圖。



2. 請依筆畫與廣告燈字型大小排列 LED，所以每個字幾乎都超過 32 個 LED，『泉』也不例外，所以『泉』就要 4 個 74244 驅動，如下圖。（若不行，請研究 Arduino 扇出數，中間再用 244 緩衝）



3. 每個字要一根 Arduino 腳位控制，本例顯示四個字，使用 4 根腳位控制，腳位分別是 22,23,24,25 所以電路示意圖如下：（每個字的電路都是同上圖）



4. 本例四個字使用 4 個腳位，Arduino Maga 2560 共有 54 個腳位，所以可控制 54 個字。台中花博的機械鳥控制也是相近的道理。

➲ 程式控制

1. 本例使用腳位 22,23,24,25，分別控制四個字，所以您就可以使用

```
degitalWrite(22,HIGH);
degitalWrite(22,LOW);
```

控制第一個字，其它三個字的原理都相同。然後四個字要如何輪流明滅，就自己發揮了。

2. 若要在這些廣告燈顯示不同的文字，就要繼續研讀本書後續章節，使用點陣 LED，秀出千遍萬化的文字。

➲ 補充說明

1. 本例若接 120 個 LED，每個 LED 亮時耗費 10mA，那同時亮時就要 1200mA，所以您就得額外自己使用變壓器供電，且留意此變壓器的額定功率。