

3

串列埠

1 串列通訊介紹

2 串列通訊的約定

3 串列通訊的應用

4 用串列通訊控制 LED

講師 張傑帆 Chang, Jie-Fan

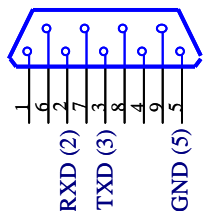
數位訊號的缺點是每次的訊息量小，只有 0 和 1 兩種狀態，所以就出現了串列通訊的概念，通過多次 0 或 1 的數位訊號組合來表達更豐富的資訊。本章要介紹 Arduino 中最基本的串列通訊。

RS-232介面

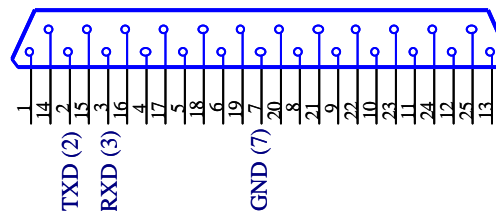
在USB介面尚未普及之前，很多周邊裝置如滑鼠、數據機、搖桿、條碼掃描機等——都是使用RS-232C介面。RS-232C介面是由美國電子工業協會 (Electronic Industries Association，簡記EIA) 制定的標準，是屬於串列介面的一種，EIA另外還制定了RS-422A及RS-485等標準，電壓特性比較如表6-1所示。因為RS-232C與TTL / CMOS的邏輯準位不同，所以需要使用如MAX232等介面IC來轉換。

表 6-1 傳輸介面電壓特性比較

特性	TTL	CMOS	RS-232C	RS-422A	RS-485
低態電壓(LOW)	$\leq 0.4V$	$\leq 0.05V_{DD}$	+5V~+15V	+2V~+6V	+2V~+6V
高態電壓(HIGH)	$\geq 2.4V$	$\geq 0.95V_{DD}$	-5V~-15V	-2V~-6V	-2V~-6V



(a) 9 pin D 型接頭



(b) 25 pin D 型接頭

圖 6-1 RS-232 介面



USB介面

如下圖所示為通用串列匯流排（**Universal Serial Bus**，簡記**USB**），是由Intel及Microsoft等幾家大廠所發起的串列介面標準。**USB**介面是目前在電腦上應用最廣泛的通訊介面，主要目的是在整合複雜的周邊連接埠成為單一標準，現今幾乎所有周邊裝置皆有支援**USB**介面。**USB**介面有**四條線**，中間**兩條線**負責**資料傳送與接收**，兩邊兩條線為**電源線與接地線**，**USB**介面屬於串列介面的一種，透過串聯方式最多可串接**127**個裝置，具有速度快、連線簡單及不需要外接電源等優點。早期**USB1.1**介面傳輸速度為**12Mbps**，**USB 2.0**可達**480Mbps**，而**USB3.0**更可達**5Gbps**。



(a) A 型



(b) B 型



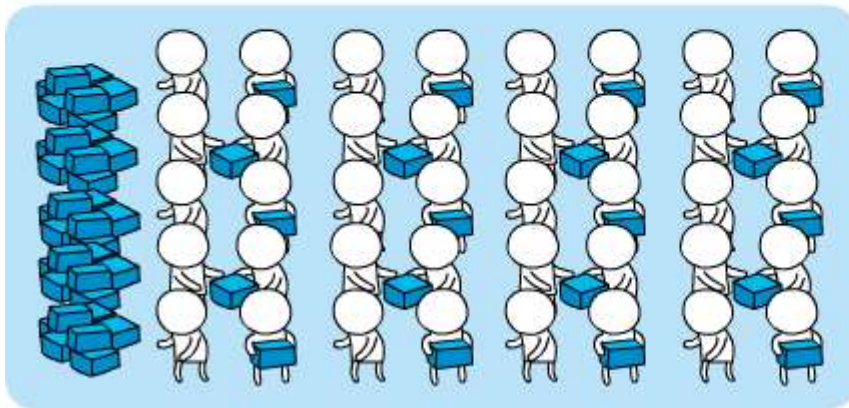
並列與序列通訊簡介

並列代表處理器和周邊之間，有多條資料線連結，處理器能一口氣輸出或接收多個位元的資料。

序列則是用少數（通常是兩條或三條）資料線，將整批資料依序一個個送出或傳入。



序列（串列）就是一次傳送一個位元資料



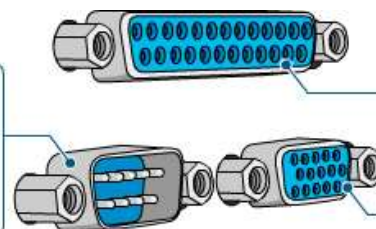
並列則是一次傳送多個位元資料，在微電腦上，通常是一次傳八個位元。



認識序列埠

RS-232是最早廣泛使用的序列埠標準，在系統軟體中稱為COM，每個COM介面同時只能接一個裝置。

序列埠 / 串列埠
桌上型電腦的RS-232C
介面，這個連接器稱為
D型9針 (DB-9) 插座。

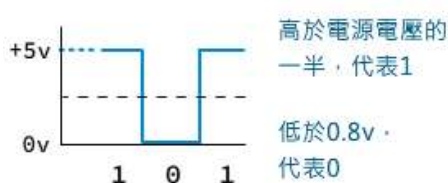


並列埠 / 印表機埠 / 平行埠
採D型25針插座 (DB-25)

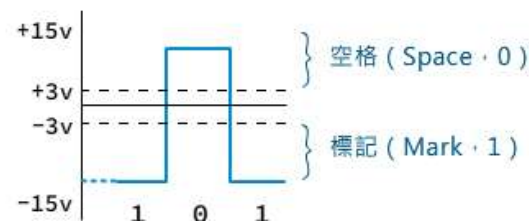
VGA顯示埠

一般數位IC的0與1訊號的電壓準位，分別是0v和5v，稱為TTL或邏輯準位。

TTL訊號的電壓



RS-232訊號的電壓



序列埠最重要的三個接腳：

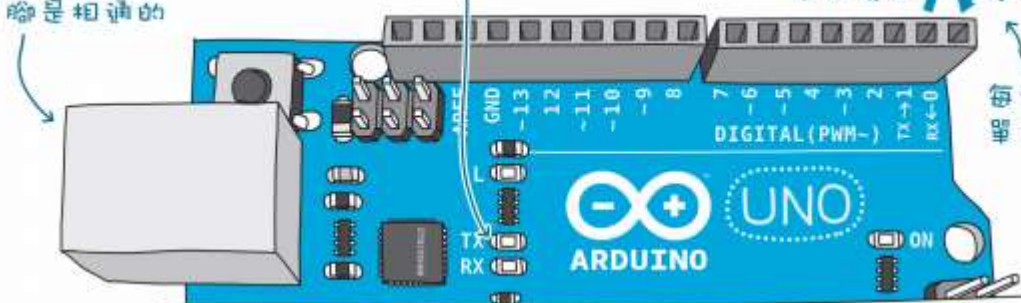
- 數據傳送 (TxD)
- 數據接收 (RxD)
- 接地 (GND)

USB序列埠和數位0
與數位1腳是相通的

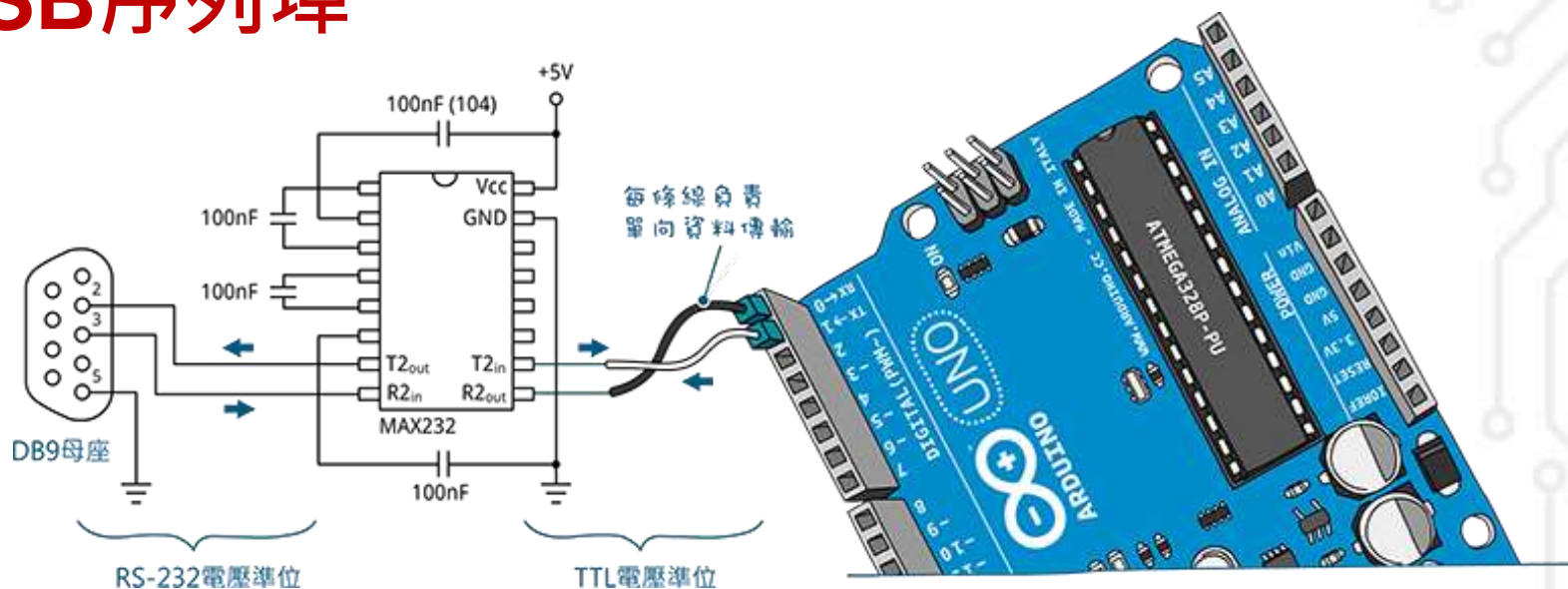
序列傳送與接收信號燈

序列傳送 ↑ 序列接收 ↓

每條線負責
單向資料傳輸



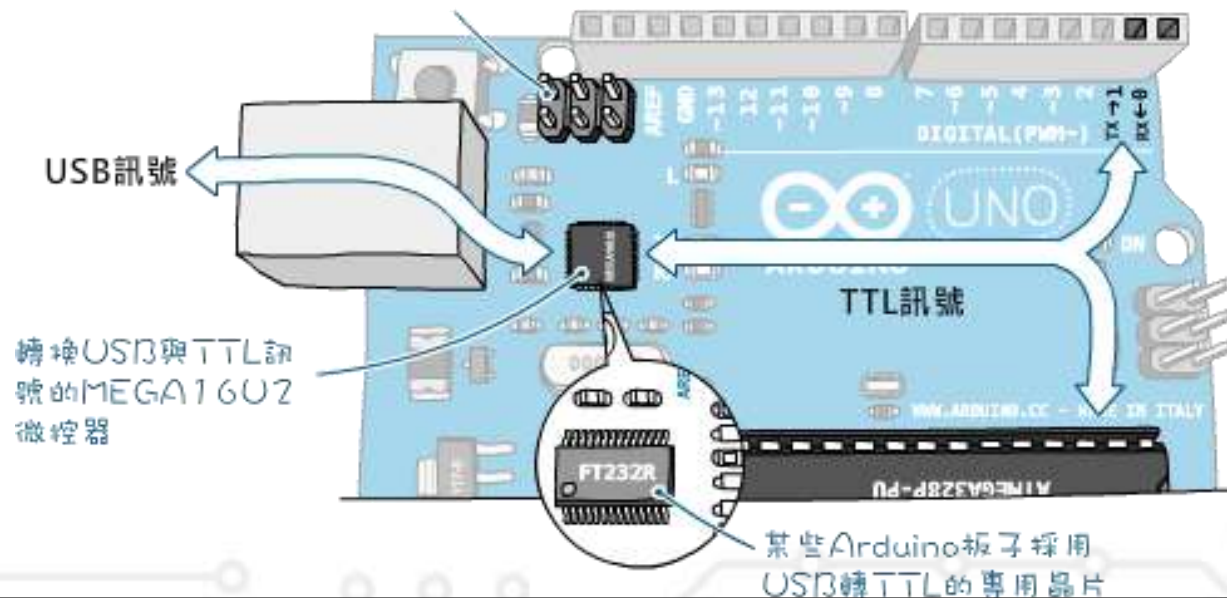
USB序列埠



RS-232電壓準位

TTL電壓準位

燒錄MEGA16U2微
控器程式的ICSP插座



轉換USB與TTL訊
號的MEGA16U2
微控器

某些Arduino板子採用
USB轉TTL的專用晶片



串列通訊介紹

串列通訊是指使用一條資料線，將資料一位元一位元地依次傳輸，每一位元資料佔據一個固定的時間長度。使用串列埠通訊時，發送和接收到的每一個字元實際上都是一次一位元傳送的，每一位為 1 或者為 0。

在串列通訊中，“**雙方約定好**”這一點很重要，因為從實質上來說，通訊的信號就是一堆 0 和 1 的數字，如果沒有約定好這些 0、1 數位組合所代表的意義，那麼雙方不可能知道對方所發送資訊的含義，就好像如果兩個人交談時使用不同的語言，那麼他們的交談是沒有任何意義的。

串列通訊的約定

串列通訊中的這種約定包含兩方面，一方面是通訊的**速率要一致**，另一方面是**字元的編碼要一致**。



串列傳輸速率

通訊速率是指單位時間內傳輸的訊息量，可用位元速率和串列傳輸速率來表示。位元速率是指每秒傳輸的二進位位元數，用 **bps** (bit/s) 表示。串列傳輸速率是指每秒傳輸的符號數，若每個符號所含的訊息量為 1 位元，則串列傳輸速率等於位元速率。在電子學中，一個符號的含義為高電位或低電位，它們分別代表 “1” 和 “0”，所以每個符號所含的訊息量剛好為 1 位元，因此常將位元速率稱為串列傳輸速率，即：

$$1 \text{ 位元 (bit)} = 1 \text{ 位 / 秒 (1bps)}$$

常用的串列傳輸速率有：110、300、600、1200、2400、4800、9600、19200、38400、115200 等，最常用的是 9600。



ASCII 碼

ASCII 碼是由美國國家標準學會 (American National Standard Institute , ANSI) 制定的，其英文全稱是 American Standard Code for Information Interchange，它是現今最通用的單字元編碼系統，主要是為了解決大家在串列通訊中的資訊一致性問題。在 Arduino 中也採用這種字元編碼方式。

在電腦中，所有的資料在儲存和運算時都用 0 或 1 來表示，像 a、b、c、d 這樣的字母（包括大寫共 52 個），以及 0、1 等數字還有一些常用的符號（*、#、@ 等），在電腦中都要使用 0 或 1 來表示，而實際用哪些 0、1 組合表示哪個符號，每個人都可以約定自己的一套定義（這個定義就叫編碼），只要雙方的編碼一致就可以通訊了。而要想讓更多人互相通訊而不造成混亂，那麼大家就必須使用相同的編碼規則，於是美國有關的標準化組織就定義了所謂的 ASCII 編碼，統一規定了上述的常用符號如何用 0、1 的組合來表示。ASCII 是基於拉丁字母的一套電腦編碼系統。它主要用於顯示現代英語和其他西歐語言。



標準 ASCII 碼

ASCII 碼使用指定的 7 bit 或 8 bit 資料組合來表示 128 或 256 種可能的字符。標準 ASCII 碼使用 7 bit 資料來表示所有的大寫和小寫字母、數字 0 到 9、標點符號，以及在美式英語中使用的特殊控制字元。

其中 0 ~ 31 及 127 (共 33 個) 是控制字元或通訊專用字元，如：LF (換行)、CR (歸位)、FF (換頁)、DEL (刪除)、BS (退格)、BEL (響鈴) 等；通訊專用字元包括：SOH (文頭)、EOT (文尾)、ACK (確認) 等

ASCII 值為 8、9、10 和 13 分別轉換為退格、製表、換行和歸位字元。這些字元並沒有特定的圖形顯示，但會依不同的應用程式，而對文字顯示產生不同的影響，其餘為可顯示字元。

- 32 ~ 126 (共 95 個) 是字元 (32 是空格)
 - 48 ~ 57 為 0 到 9 十個阿拉伯數字
 - 65 ~ 90 為 26 個大寫英文字母
 - 97 ~ 122 號為 26 個小寫英文字母

其餘為一些標點符號、運算子符號等，標準 ASCII 碼如下頁表所示。



▼表 標準 ACSII 碼

二進位數字	十進位數字	含義 / 字元	解釋
0000 0000	0	NUL(null)	空字元
0000 0001	1	SOH(start of headline)	標題開始
0000 0010	2	STX(start of text)	正文開始
0000 0011	3	ETX(end of text)	正文結束
0000 0100	4	EOT(end of transmission)	傳輸結束
0000 0101	5	ENQ(enquiry)	請求
0000 0110	6	ACK(acknowledge)	收到通知
0000 0111	7	BEL(bell)	響鈴
0000 1000	8	BS(backspace)	退格
0000 1001	9	HT(horizontal tab)	水平定位字元
0000 1010	10	LF(NL line feed, new line)	換行鍵
0000 1011	11	VT(vertical tab)	垂直定位字元
0000 1100	12	FF(NP form feed, new page)	換頁鍵
0000 1101	13	CR(carriage return)	歸位鍵
0000 1110	14	SO(shift out)	不用切換
0000 1111	15	SI(shift in)	啟用切換
0001 0000	16	DLE(data link escape)	資料連結轉義
0001 0001	17	DC1(device control 1)	設備控制 1
0001 0010	18	DC2(device control 2)	設備控制 2
0001 0011	19	DC3(device control 3)	設備控制 3
0001 0100	20	DC4(device control 4)	設備控制 4
0001 0101	21	NAK(negative acknowledge)	拒絕接收

續下頁表



二進位數字	十進位數字	含義 / 字元	解釋
0001 0110	22	SYN(synchronous idle)	同步空閒
0001 0111	23	ETB(end of trans. block)	傳輸塊結束
0001 1000	24	CAN(cancel)	取消
0001 1001	25	EM(end of medium)	介質中斷
0001 1010	26	SUB(substitute)	替補
0001 1011	27	ESC(escape)	換碼 (溢出)
0001 1100	28	FS(file separator)	文件分隔符號
0001 1101	29	GS(group separator)	分組符
0001 1110	30	RS(record separator)	記錄分離符
0001 1111	31	US(unit separator)	單元分隔符號
0010 0000	32	(space)	空格
0010 0001	33	!	
0010 0010	34	"	
0010 0011	35	#	
0010 0100	36	\$	
0010 0101	37	%	
0010 0110	38	&	
0010 0111	39	'	
0010 1000	40	(
0010 1001	41)	
0010 1010	42	*	
0010 1011	43	+	
0010 1100	44	,	
0010 1101	45	-	
0010 1110	46	.	
00101111	47	/	
00110000	48	0	
00110001	49	1	
00110010	50	2	
00110011	51	3	
00110100	52	4	

續下頁表



二進位數字	十進位數字	含義 / 字元	解釋
00110101	53	5	
00110110	54	6	
00110111	55	7	
00111000	56	8	
00111001	57	9	
00111010	58	:	
00111011	59	;	
00111100	60	<	
00111101	61	=	
00111110	62	>	
00111111	63	?	
01000000	64	@	
01000001	65	A	
01000010	66	B	
01000011	67	C	
01000100	68	D	
01000101	69	E	
01000110	70	F	
01000111	71	G	
01001000	72	H	
01001001	73	I	
01001010	74	J	
01001011	75	K	
01001100	76	L	
01001101	77	M	
01001110	78	N	
01001111	79	O	
01010000	80	P	
01010001	81	Q	
01010010	82	R	
01010011	83	S	

續下頁表



二進位數字	十進位數字	含義 / 字元	解釋
01010100	84	T	
01010101	85	U	
01010110	86	V	
01010111	87	W	
01011000	88	X	
01011001	89	Y	
01011010	90	Z	
01011011	91	[
01011100	92	\	
01011101	93]	
01011110	94	^	
01011111	95	_	
01100000	96	`	
01100001	97	a	
01100010	98	b	
01100011	99	c	
01100100	100	d	
01100101	101	e	
01100110	102	f	
01100111	103	g	
01101000	104	h	
01101001	105	i	
01101010	106	j	
01101011	107	k	
01101100	108	l	
01101101	109	m	
01101110	110	n	
01101111	111	o	
01110000	112	p	

續下頁表



二進位數字	十進位數字	含義 / 字元	解釋
01110001	113	q	
01110010	114	r	
01110011	115	s	
01110100	116	t	
01110101	117	u	
01110110	118	v	
01110111	119	w	
01111000	120	x	
01111001	121	y	
01111010	122	z	
01111011	123	{	
01111100	124		
01111101	125	}	
01111110	126	~	
01111111	127	DEL(delete)	刪除



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



串列通訊的應用

Arduino串列介面

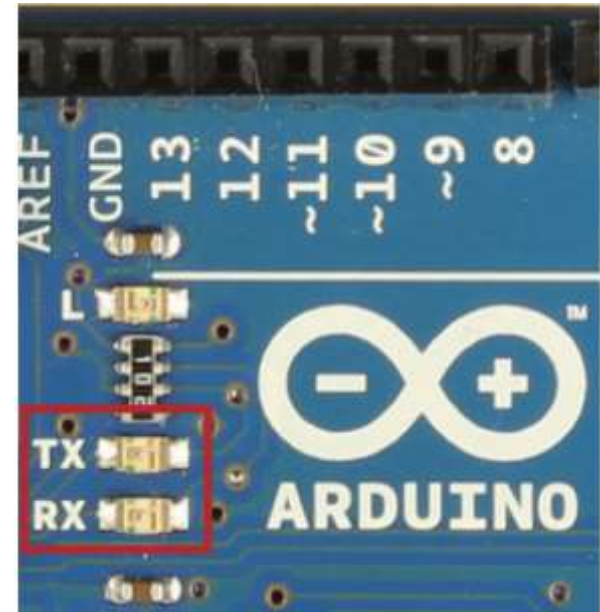
早期的Arduino板採用RS-232介面，現今皆採用USB介面，在Arduino板子上的USB接頭旁邊有一個晶片負責USB介面與TTL介面的信號轉換。一個標準的Arduino板至少有一個硬體串列埠，使用RX腳（數位0號接腳）與TX腳（數位1號接腳）與電腦連線互動。在Arduino IDE中內建Serial Monitor程式來顯示由Arduino板所傳輸的文、數字資料內容。Arduino提供Serial串列函式庫來簡化串列通訊的複雜性，使用者可以輕鬆使用Serial函式庫來設定連線，傳送及接收資料。

- 在 Arduino 端進行串列通訊的接腳稱為串列埠，一般分為發送和接收
 - 發送用 TX 表示
 - 接收用 RX 表示。





◆ Arduino上的串列通訊接腳



◆ Arduino上通訊指示燈



傳遞序列訊息給電腦

參數設置函數

設定序列埠連線速率

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

↖ S大寫!

```
Serial.print("Hello ");  
Serial.print("World.");
```

輸出："Hello World."

```
Serial.println("Hello ");  
Serial.print("World.");
```

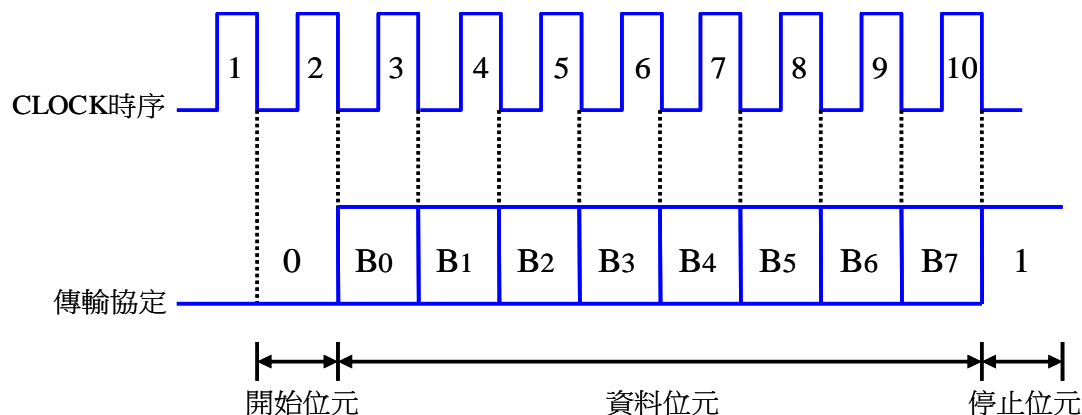
輸出："Hello
World."



函式說明

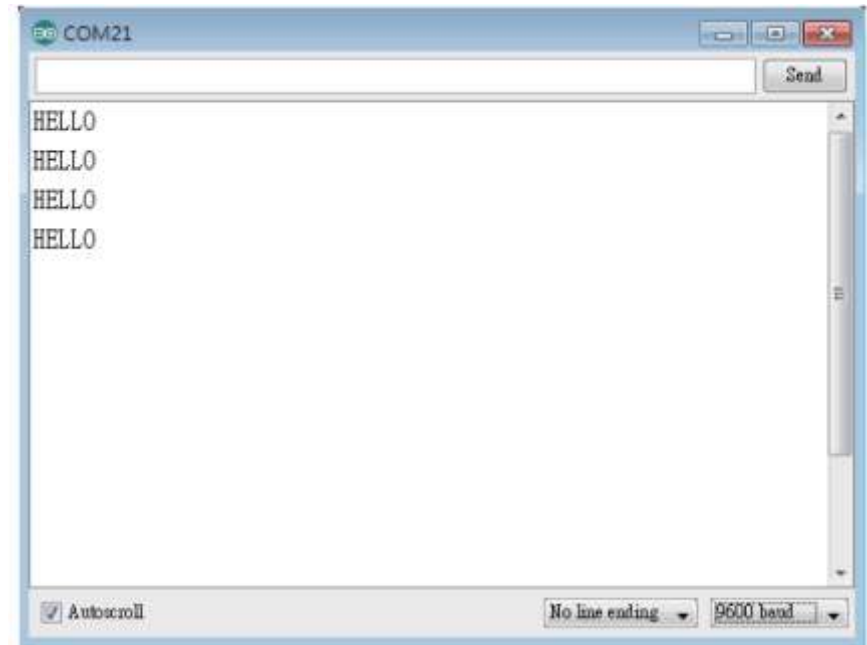
Serial.begin()函式

要建立一個串列通訊連線，首先是傳送端與接收端必須要設定相同的「通訊協定」，所謂「通訊協定」是指資料的傳輸格式，在串列埠中常用的傳輸格式如下圖所示，由1個開始位元、8個資料位元及1個停止位元所組成。其次是設定傳輸速率，或稱為鮑率（Baud rate），可以設定的速率為300 bps到115200 bps之間，常見的選擇為9600 bps。



請按一下ArduBlock功能列的最右邊：
Serial Monitor。藉此叫出另一個視
窗：

◆ Serial Monitor視窗



這個視窗是Arduino軟體中內建的串列埠通訊視窗，由於Win7並沒有內建相關的軟體，因此這是一個非常好用的串列埠工具。

當初通訊方塊的new line設定為TRUE，顯示的時候才會每一行分隔出來，各位可以試試看將new line設定為FALSE，顯示的方式應該就會不太一樣。



從序列埠監控視窗觀察變數

Arduino內建處理序列埠連線的Serial程式庫

建立序列埠連線的首要任務是設定資料傳輸率，底下的程式設定為9600bps
序列埠監控視窗的連線速率要和Arduino程式一致。

```
const byte ledPin = 13;
```

```
void setup() {
```

初始化序列埠，以9600bps速率連線。

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

```
Serial.println("Hello,");
```

```
Serial.print("\tLED pin is: ");
```

```
Serial.print(ledPin);
```

```
Serial.print("\nBYE!");
```

```
void loop() {
```

參數設置函數

設定序列埠連線速率

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

S大寫！



函式說明

Serial.print() 函式

格式： Serial.print(val)

Serial.print(val, format)

範例： Serial.print('A');	//輸出字元 A。
Serial.print("Hello");	//輸出字串 Hello。
Serial.print(65 , BIN);	//輸出數值 65 的二進位值 1000001。
Serial.print(65 , OCT);	//輸出數值 65 的八進位值 101。
Serial.print(65 , DEC);	//輸出數值 65 的十進位值 65。
Serial.print(65 , HEX);	//輸出數值 65 的十六進位值 41。
Serial.print(12.3456);	//輸出數值 12.35。
Serial.print(12.3456,1);	//輸出數值 12.3。
Serial.print(12.3456,2);	//輸出數值 12.35。
Serial.print(12.3456,3);	//輸出數值 12.346。
Serial.print(12.3456,4);	//輸出數值 12.3456。



函式說明

Serial.println() 函式

Serial.println() 函式與 Serial.print() 函式有相同的格式，唯一不同的是：輸出 val 文字或數值資料結束後，再輸出一個歸位字元(ASCII 13 或\r)與一個換行字元(ASCII 10 或\n)，簡單來說就是在輸出文字或數值資料結束後，游標移至下一列的開頭。

格式： `Serial.println(val)`

`Serial.println(val, format)`



函式說明

Serial.write()函式

- ◆ Serial.write()函式可以將ASCII文字或數值輸出到串列埠，
- ◆ 如果是設定val數值參數，輸出為數值資料的ASCII文字；
- ◆ 如果是設定str字串參數，輸出為字串；
- ◆ 如果是設定無號數字元 (unsigned char) 的buf陣列文字或數值資料，輸出為ASCII文字，長度由len參數決定。

格式： `Serial.write(val)`

`Serial.write(str)`

`Serial.write(buf,len)`

範例： `Serial.write(65);` //輸出字元 A。

`Serial.write("ABC");` //輸出字元 ABC。



實作練習

從 **Arduino** 板傳送訊息至電腦實習

□ 功能說明：

使用 Arduino 板子透過串列埠傳送 26 個大寫英文字母及其 10 進制 ASCII 碼訊息至電腦中。將 Arduino 與電腦連接，然後將程式上傳至 Arduino 板後，再開啟 Serial Monitor 視窗來顯示所傳送的内容如圖 6-5 所示。



圖 6-5 從 Arduino 板傳送訊息至電腦



實作練習

□ 程式： B301.ino

```
byte val=65; //定義位元組數值資料 val=65。
void setup()
{
    Serial.begin(9600); //初始化串列埠，設定鮑率為 9600bps。
}
void loop()
{
    for(int i=0;i<26;i++) //26 個大寫英文字母。
    {
        Serial.write(val+i); //輸出字母字元至串列埠。
        Serial.print('='); //輸出字元'=' 至串列埠。
        Serial.println(val+i); //輸出 ASCII 碼至串列埠。
        delay(1000); //延遲 1 秒後，再輸出下一個字母。
    }
}
```



(回家)小練習



1. 設計 Arduino 程式，使用 Arduino 板透過串列埠傳送 26 個小寫英文字母及其 10 進制 ASCII 碼訊息至電腦中。
2. 設計 Arduino 程式，以 Arduino 板透過串列埠傳送 26 個大寫英文字母及其 16 進制 ASCII 碼訊息至電腦中。

3. 請試著在每一行字前加一字串"Arduino"，並加上行數

Ex:

Arduino 1: a=97

Arduino 2: b=98

Arduino 3: c=99

Arduino 4: d=100



字串資料類型

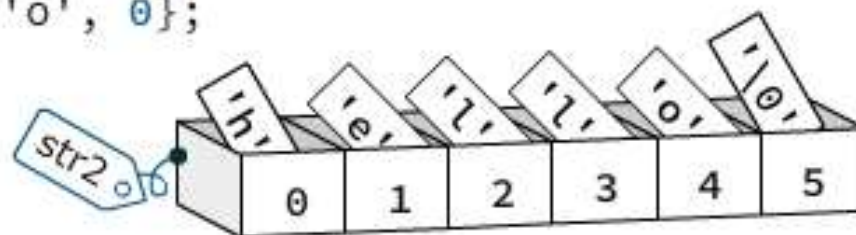
字串是一連串字元 (char) 的集合，也就是一段文字。
程式採用陣列存放字串，資料值前後一定要用雙引號括起來。
每個字串都有一個Null字元 (ASCII值為0) 結尾。

`char str1[] = {'h','e','l','l','o','\0'};`

↖ 加上Null結尾

↖ 明確指定元素數量
`char str2[6] = {'h','e','l','l','o', 0};`

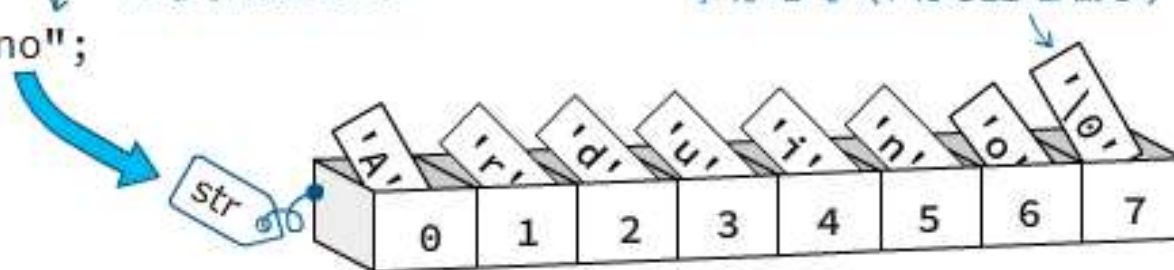
↖ Null值也能用ASCII編碼數字設定



`char str[] = "Arduino";`

↖ 用雙引號括起來

字串會自動加上一個Null字元結尾 (ASCII值為0)



實作練習

從 **Arduino** 板傳送 **LED** 狀態至電腦實習

□ 功能說明：

使用 Arduino 板控制 LED 單燈右移，同時將 LED 目前狀態傳送至電腦中。當 LED 亮時，狀態為邏輯 1；當 LED 暗時，狀態為邏輯 0。將 Arduino 與電腦連接，然後將程式上傳至 Arduino 板後，再開啟 Serial Monitor 視窗來顯示所傳送的内容如圖 6-6 所示。



圖 6-6 Serial Monitor 程式顯示情形



實作練習

□ 路圖及麵包板接線圖：

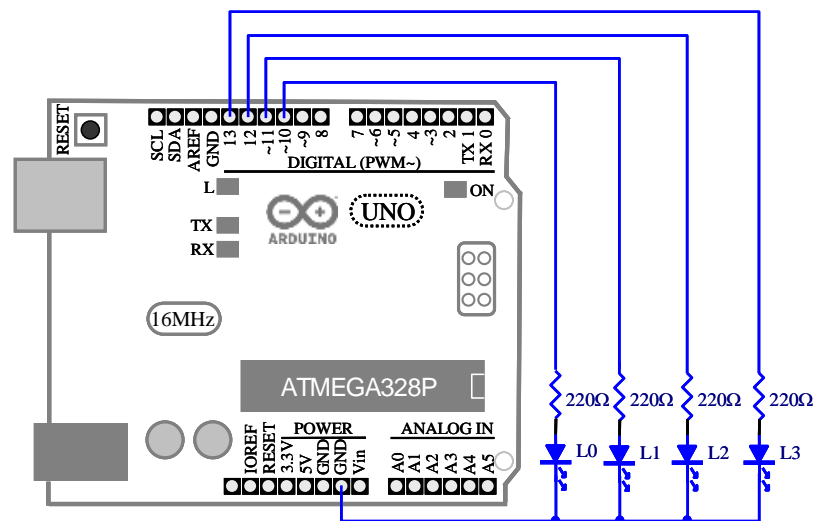


圖 6-7 從 Arduino 板傳送 LED 狀態至電腦實習電路圖

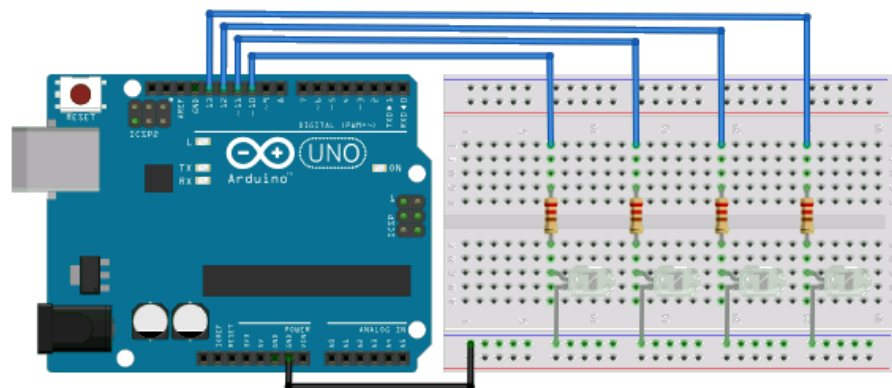


圖 6-8 從 Arduino 板傳送 LED 狀態至電腦實習麵包板接線圖



實作練習

□ 程式： 302.ino

```
int led[] = {10, 11, 12, 13};           // 數位接腳 10~13 連接至四個 LED。
int i=0, j;                             // LED 編號。
void setup()
{
    Serial.begin(9600);                  // 初始化串列埠，設定鮑率為 9600bps。
    for(int i=0; i<4; i++)
    {
        pinMode(led[i], OUTPUT);        // 設定數位接腳 10~13 為輸出模式。
    }
}
```



實作練習

```
void loop()  
{  
  
    Serial.print("LED=");          //顯示。  
    for(j=0;j<4;j++)  
    {  
        if(j==i)  
            Serial.print("1");      //LED 亮則顯示 1。  
        else  
            Serial.print("0");      //LED 暗則顯示 0。  
    }  
    Serial.println();              //換下一行。  
    digitalWrite(led[i],HIGH);     //第 i 個 LED 亮。  
    delay(1000);                   //延遲 1 秒。  
    digitalWrite(led[i],LOW);      //關閉第 i 個 LED。  
    i++;                           //右移。  
    if(i>3)                         //已移位至最右?  
        i=0;                       //重設初值。  
}
```



實作練習



1. 設計 Arduino 程式，使用 Arduino 板控制 LED 單燈左移，同時將 LED 目前狀態傳送至電腦中。當 LED 亮時，狀態為 HIGH；當 LED 暗時，狀態為 LOW。
2. 設計 Arduino 程式，使用 Arduino 板控制 LED 單燈閃爍左移，同時將 LED 目前狀態傳送至電腦中。當 LED 亮時，狀態為 HIGH；當 LED 暗時，狀態為 LOW。



函式說明

Serial.available() 函式

Serial.available() 函式可以得到從串列埠所讀取到的位元組數目，沒有參數，傳回值為位元組數目。

格式： Serial.available()

範例： char num=Serial.available(); //讀取電腦傳入的位元組數目，存入 num 中。

Serial.read() 函式

Serial.read() 函式可以讀取電腦傳入的 8 位元數值資料，沒有參數，傳回值為 8 位元數值資料。

格式： Serial.read()

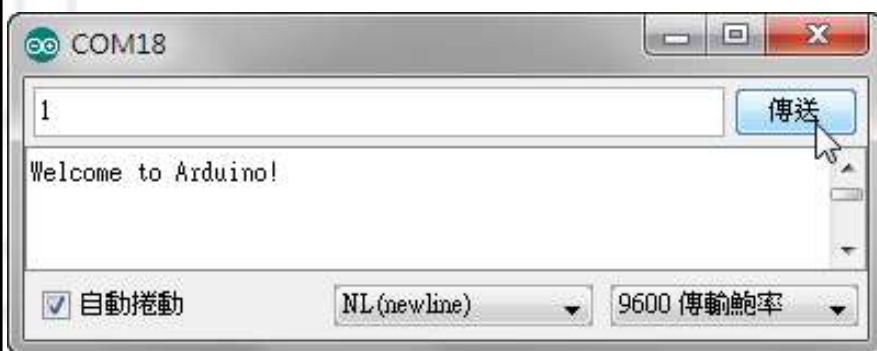
範例： char ch=Serial.read(); //讀取電腦傳入的 8 位元數值資料，存入 ch 中。



從Arduino接收序列資料

微處理器內部有一個類似儲存槽的緩衝記憶區 (buffer)，用於暫存來自序列埠的輸入資料。

如果緩衝記憶區裡面沒有資料，`Serial.available()`將傳回0。



```
void loop() {  
  if( Serial.available() ) { ← 只要有收到字元，條件式的内容將被執行。  
    val = Serial.read();  
    if (val == '1') {  
      digitalWrite(LED, HIGH);  
      Serial.print("LED ON");  
    } else if (val == '0') {  
      digitalWrite(LED, LOW);  
      Serial.print("LED OFF");  
    }  
  }  
}
```

若收到'1'，點亮LED。

若收到'0'，關閉LED。



實作練習

從 **Arduino** 板接收電腦訊息實習

□ 功能說明：

從 Arduino 板接收電腦傳送的訊息，並於 Serial Monitor 視窗顯示其 10 進制 ASCII 碼。將 Arduino 與電腦連接，然後將程式上傳至 Arduino 板後，接著開啟 Serial Monitor 視窗。如圖 6-9 所示，在傳送欄位輸入“ABCD”後，按下 **傳送** 鈕，在 Serial Monitor 視窗中會顯示 A、B、C、D 四個字元的 ASCII 碼。



實作練習

□ 程式： B311.ino

```
int num = 0; //自串列埠中讀取可用位元組數目。
void setup()
{
    Serial.begin(9600); //初始化串列埠，設定傳輸速率為 9600。
}
void loop()
{
    if(Serial.available()>0) //是否至少有接收到一個可用的字元?
    {
        num=Serial.read(); //讀取字元資料。
        Serial.print("I received:"); //輸出"I received:"字串。
        Serial.println(num, DEC); //輸出所讀取到的字元數值。
    }
}
```



實作練習



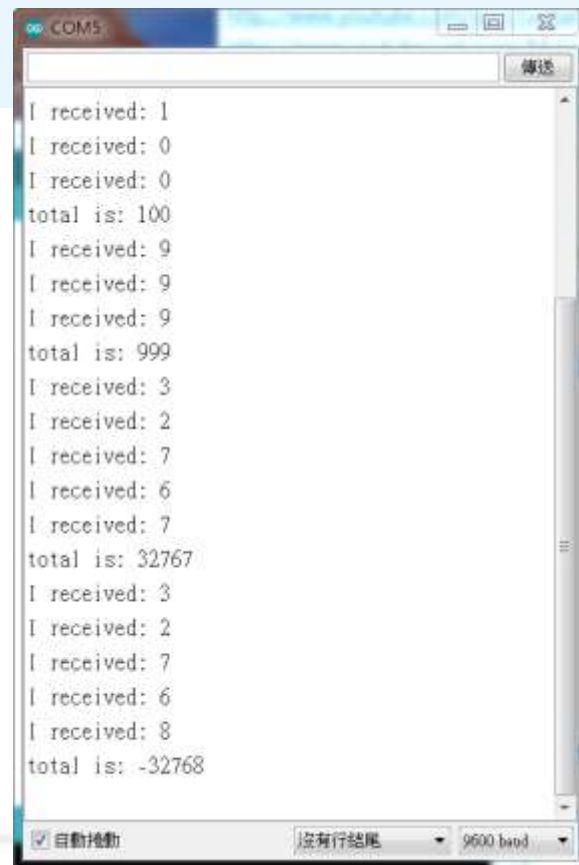
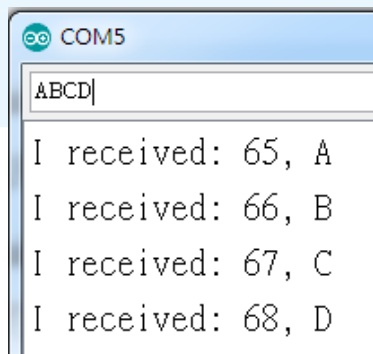
1. 設計 Arduino 程式，從 Arduino 板接收電腦傳送的訊息，並於 Serial Monitor 視窗顯示其 16 進制 ASCII 碼。
2. 設計 Arduino 程式，從 Arduino 板接收電腦傳送的訊息，並於 Serial Monitor 視窗顯示所接收的字元。

回家作業

◆ 設計Arduino程式

從Arduino板接收傳腦傳送的訊息

並於Serial Monitor視窗顯示所接收的數字
須可接收與顯示 -32768以上到32767以下的
的數字。



接收連續序列埠組成數字

按鍵的輸入值是字串，
而類比輸出需要的是數字格式。


透過右邊的加工手續可
將'1', '6', '8'字元轉變成數字168。

❶ Arduino收到字串 '168\n'

❷ 處理第一個字：

此變數預設為0
$$pw = pw * 10 + (_in - '0')$$


收到字元'1'
$$0 * 10 + ('1' - '0')$$

轉換成數字
在pw變數存入1


❸ 處理第二個字：

此變數值為1
$$pw = pw * 10 + (_in - '0')$$

收到字元'6'
$$1 * 10 + ('6' - '0')$$



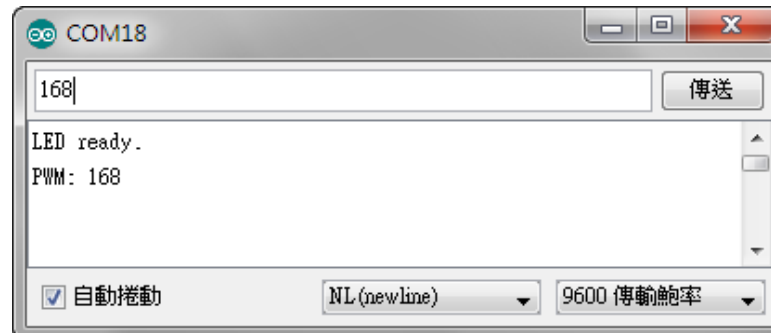
❹ 處理第三個字：

此變數值為16
$$pw = pw * 10 + (_in - '0')$$

$$16 * 10 + ('8' - '0')$$

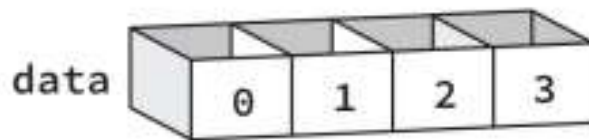


❺ 讀取到字元'\n'，轉換完畢！

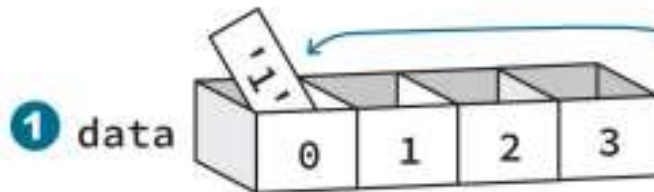


透過序列埠調整燈光亮度 (二)

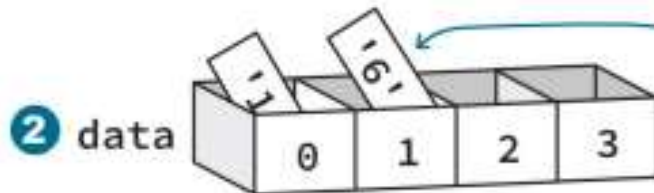
第二種轉換方式，先把字元存入陣列，最後再透過atoi()轉換。



一開始，先預設包含四個空元素的data陣列。
變數i值為0。



收到第一個字元，存入data陣列的第i個元素，接著將i值加1 (i變成1)。



收到第二個字元，存入data陣列的第i個元素，再將i值加1 (i變成2)。



收到第三個字元，存入data陣列的第i個元素，再將i值加1 (i變成3)。



最後補上代表字串結尾的NULL字元

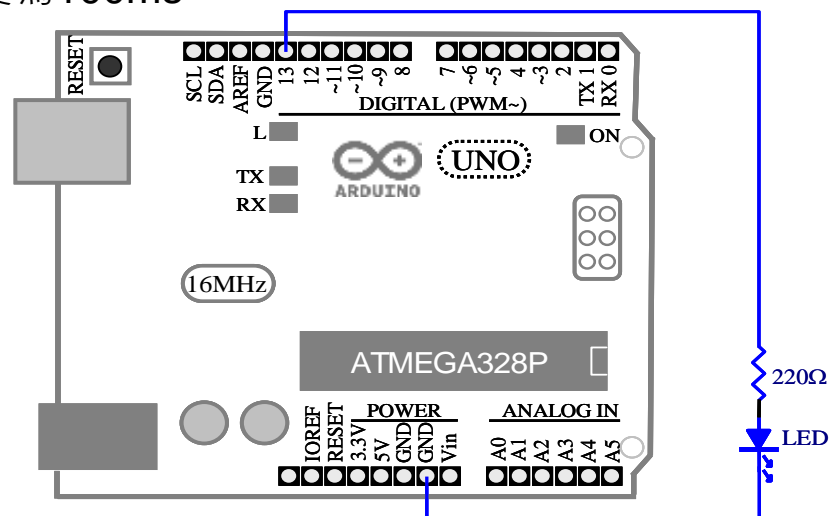


實作練習

電腦鍵盤控制LED閃爍速度實習

從Arduino板接收電腦鍵盤輸入鍵值0~9來控制LED閃爍速度

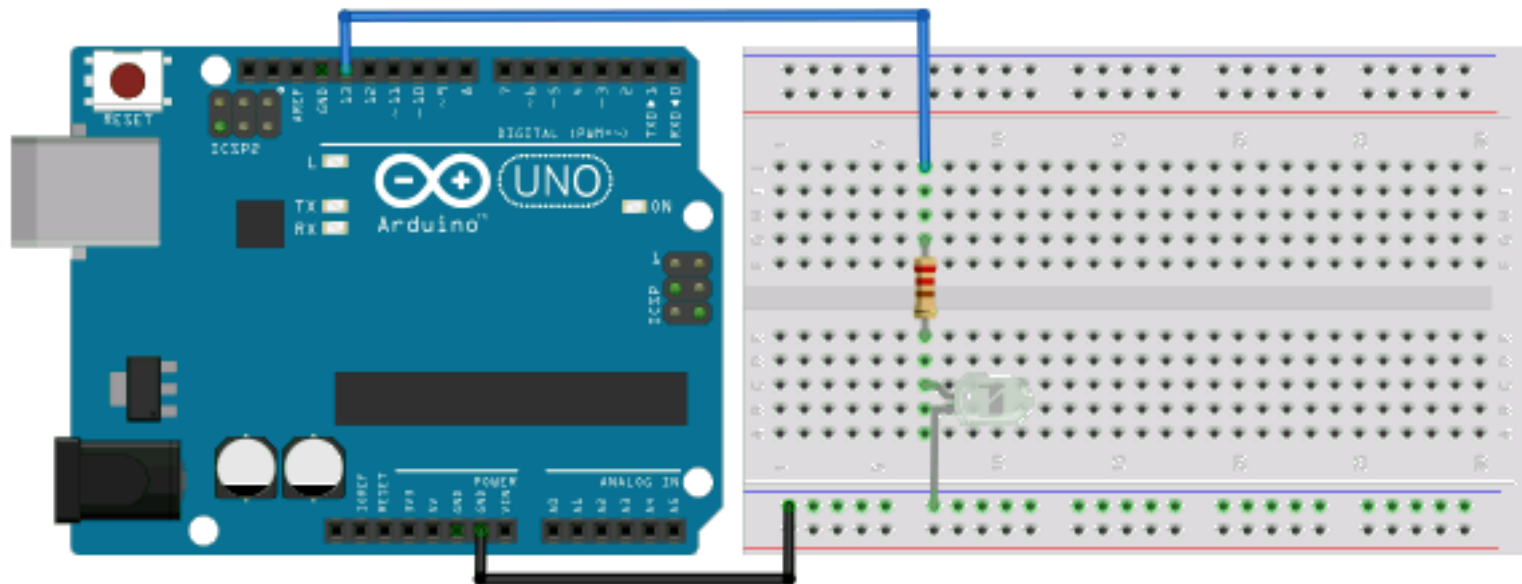
- 輸入鍵值1的LED閃爍速度為 100ms
- 輸入鍵值2的LED閃爍速度 200ms ...
- 輸入鍵值9的LED閃爍速度為 900ms
- 鍵值0的LED閃爍速度為 1000ms
- 預設閃爍速度為100ms



電腦鍵盤控制 LED 閃爍速度實習電路圖



實作練習



電腦鍵盤控制LED閃爍速度實習麵包板接線圖



實作練習

□ 程式： B321.ino

int num=0;	// 電腦鍵盤按鍵值。
int flash=100;	// 預設 LED 閃爍速度為 100ms。
int led=13;	// LED 連接至數位接腳 13。
void setup()	
{	
Serial.begin(9600);	// 初始化串列埠，設定傳輸速率為 9600 baud。
pinMode(led, OUTPUT);	// 設定數位接腳 13 為輸出模式。
}	

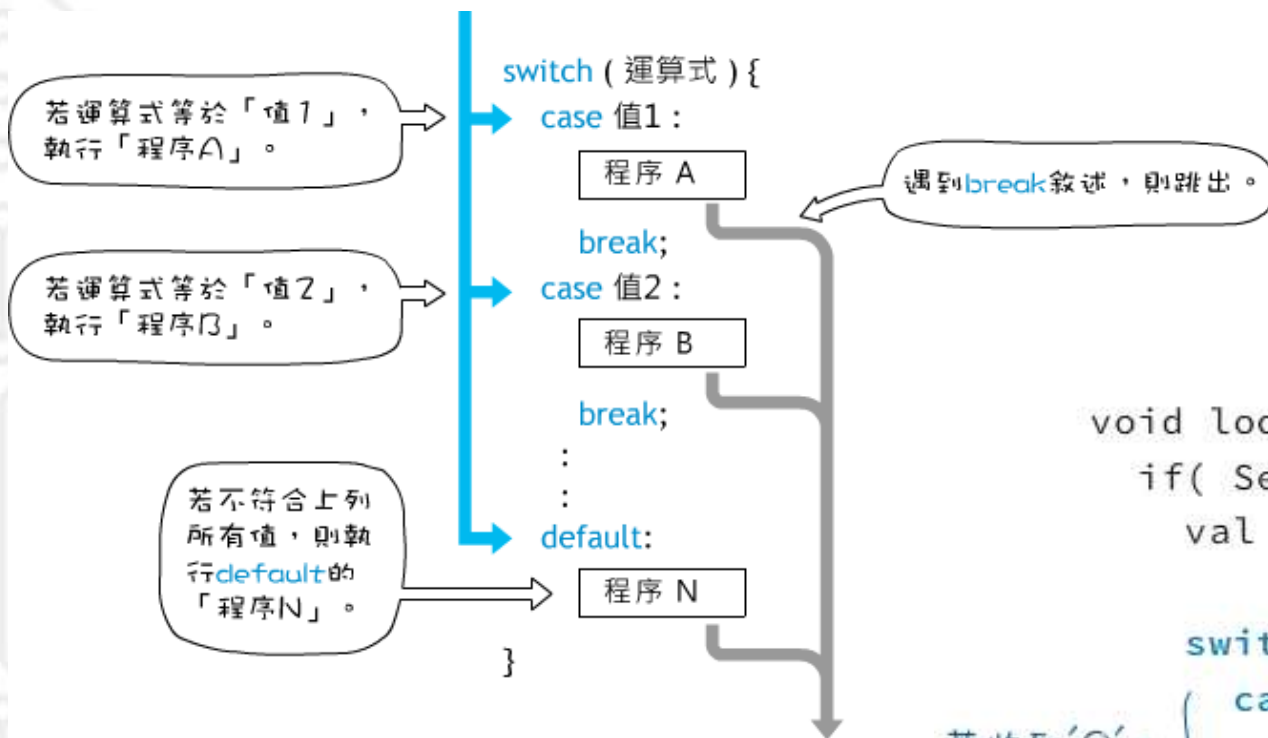


實作練習

```
void loop()  
{  
  if (Serial.available()>0)           //是否接收到可用的字元?  
  {  
    num = Serial.read();               //讀取按鍵值。  
    num=num-'0';                       //將鍵值 0~9 轉成數值 0~9。  
    if(num>=0 && num<=9)  
    {  
      if(num==0)                       //鍵值為 0?  
        flash=1000;                   //閃爍速度為 1000ms。  
      else                             //鍵值為 1~9。  
        flash=num*100;                //閃爍速度為 100ms~900ms。  
    }  
  }  
  digitalWrite(led,HIGH);              //設定 LED 為 HIGH(亮)。  
  delay(flash);                        //延遲時間。  
  digitalWrite(led,LOW);               //設定 LED 為 LOW(暗)。  
  delay(flash);                        //延遲時間。  
}
```



switch...case控制結構



switch具有「切換」的涵意：透過比對switch()裡的變數和case後面的值，來決定切換執行哪一段程式

```
void loop() {  
  if( Serial.available() ) {  
    val = Serial.read();  
  
    switch (val) {  
      case '0':  
        digitalWrite(LED, LOW);  
        break;  
      case '1':  
        digitalWrite(LED, HIGH);  
        break;  
    }  
  }  
}
```

若收到'0'，關閉LED。



實作練習



1. 設計 Arduino 程式，接收電腦鍵盤輸入鍵值來控制 LED。輸入 0 鍵則 LED 暗，輸入 1 鍵則 LED 亮，輸入 2 鍵則 LED 快閃，輸入 3 鍵則 LED 慢閃。
2. 設計 Arduino 程式，接收電腦鍵盤輸入鍵值來控制 LED 並顯示提示訊息。輸入 0 鍵則 LED 暗，輸入 1 鍵則 LED 亮，輸入 2 鍵則 LED 快閃，輸入 3 鍵則 LED 慢閃。

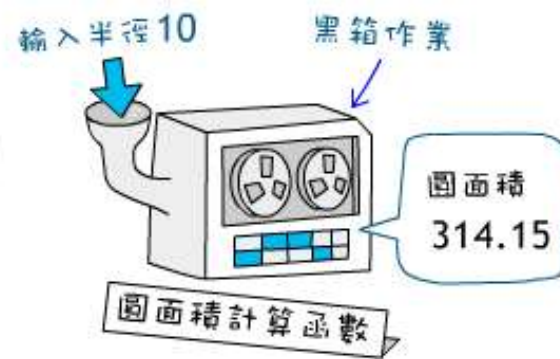
(可試用switch case 練習)



建立自訂函數

具有特定功能
並且能被重複使用的
程式碼，叫做「函數」

不怕記錯公式，只要按一下「功能鍵」，就能完成複雜的運算。



```
float cirArea() {  
    int r = 5;  
    float area = 3.14 * r * r;  
    return area;  
}
```

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

```
void loop() {  
    cirArea();  
    delay(2000);  
}
```

自訂函數cirArea()

宣告半徑 (5)

計算圓面積

輸出圓面積

自訂函數的語法範例



實作練習

從 **Arduino** 板接收電腦訊息控制 **LED** 移位方向實習

□ 功能說明：

從 Arduino 板接收電腦訊息控制 LED 移位方向。按 R 鍵則 LED 單燈右移，按 L 鍵則 LED 單燈左移。



實作練習

□ 電路圖及麵包板接線圖：

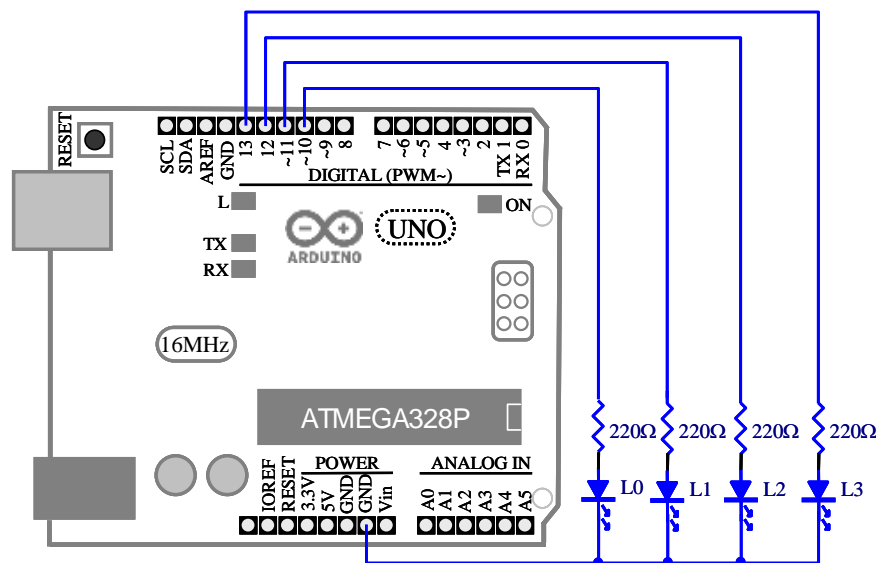


圖 6-13 從 Arduino 板接收電腦訊息控制 LED 亮與暗實習電路圖

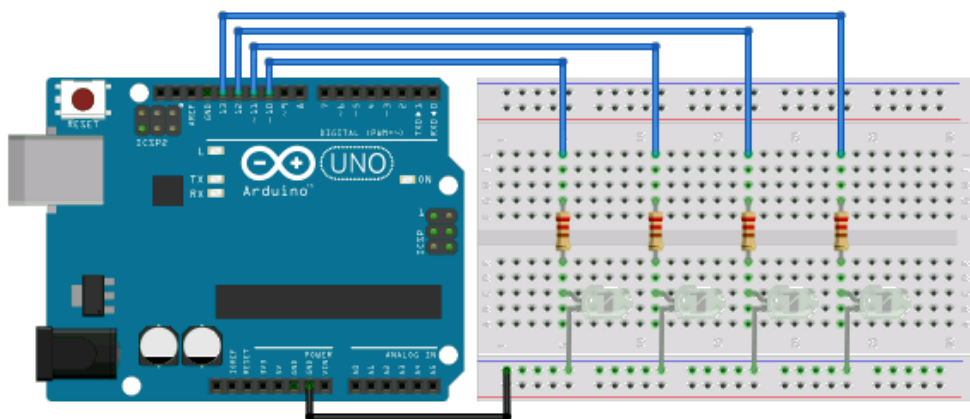


圖 6-14 從 Arduino 板接收電腦訊息控制 LED 亮與暗實習麵包板接線圖



實作練習

□ 程式： B323.ino

```
int i;                                //陣列資料指標。
int key;                              //按鍵值。
int temp;                             //資料暫存區。
int led[]={10,11,12,13};             //數位接腳 10~13 連接至四個 LED。
int status[]={1,0,0,0};              //LED 初始狀態。
void setup()
{
    Serial.begin(9600);               //初始化串列埠，鮑率為 9600bps。
    Serial.println("press R : LED shift right");
    Serial.println("press L : LED shift left");
    for(i=0;i<4;i++)                  //設定數位接腳 10~13 為輸出模式。
        pinMode(led[i],OUTPUT);
}
```



實作練習

```
void loop()  
{  
    if(Serial.available()>0)                //按任意鍵?  
        key = Serial.read();                //讀取按鍵值。  
    if(key=='R' || key=='r')                //按 R 鍵或 r 鍵?  
    {  
        temp=status[3];                    //LED 單燈右移。  
        for(i=2;i>=0;i--)  
            status[i+1]=status[i];  
        status[0]=temp;  
        display();  
    }  
    else if(key=='L' || key=='l')            //按 L 鍵或 l 鍵?  
    {  
        temp=status[0];                    //LED 單燈左移。  
        for(i=0;i<3;i++)  
            status[i]=status[i+1];  
        status[3]=temp;  
        display();  
    }  
}
```



實作練習

```
void display()                                //LED 狀態顯示。
{
    for(int i=0;i<4;i++)
    {
        if(status[i]==1)
            digitalWrite(led[i],HIGH);      //LED 亮。
        else
            digitalWrite(led[i],LOW);        //LED 暗。
    }
    delay(200);
}
```



1. 設計 Arduino 程式，接收電腦訊息控制 LED 移位方向。按 R 鍵則 LED 單燈右移；按 L 鍵則 LED 單燈左移；按 F 鍵則 LED 同時閃爍；按 S 鍵則關閉所有 LED。
2. 設計 Arduino 程式，接收電腦訊息控制 LED 移位方向。按 R 鍵則 LED 單燈閃爍右移；按 L 鍵則 LED 單燈閃爍左移；按 F 鍵則 LED 同時閃爍；按 S 鍵則關閉所有 LED。



回家作業三

同上練習

增加輸入數字0~9可控制位移速度

依據 輸入值 設定 移動速度

設定時，記得要把 **key** 從字元轉成 數字

速度可參考 B321



可設定為：9r / 9l

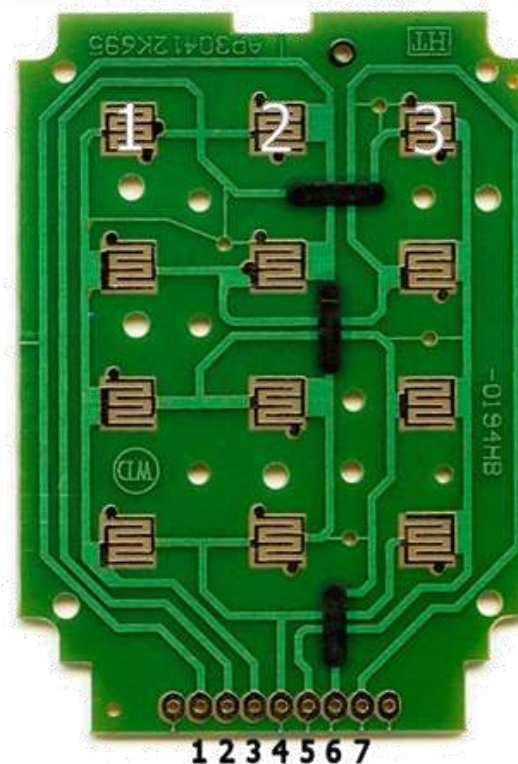
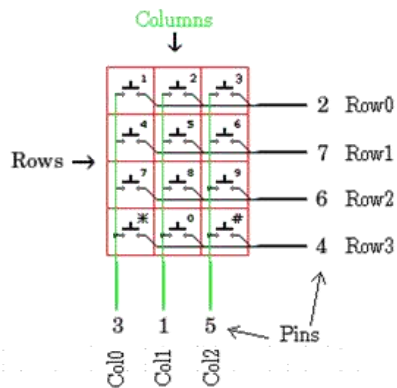
數字鍵盤



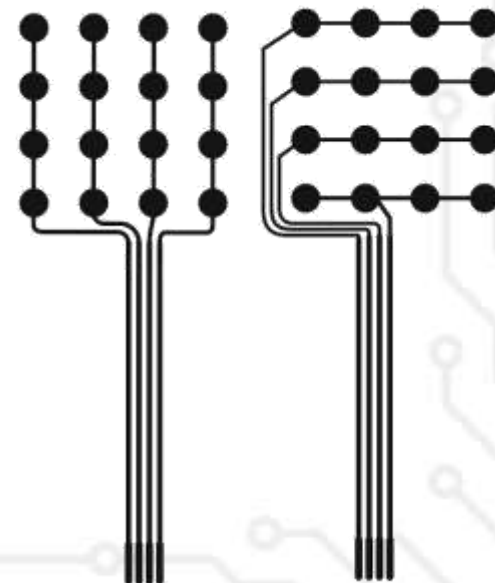
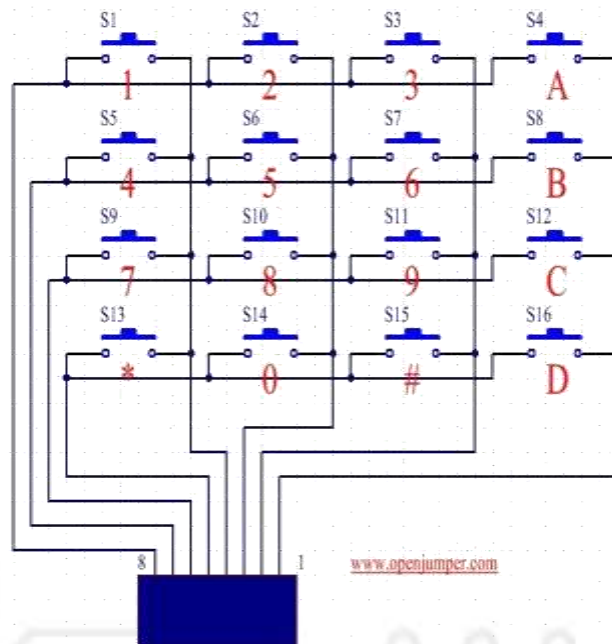
1: 2+3
2: 1+2
3: 2+5
4: 3+7
5: 1+7
6: 5+7
7: 3+6
8: 1+6
9: 5+6
*: 3+4
0: 1+4
#: 5+4

Schematic symbol for a button

⏏ Open / Released
⏏ Closed / Pressed

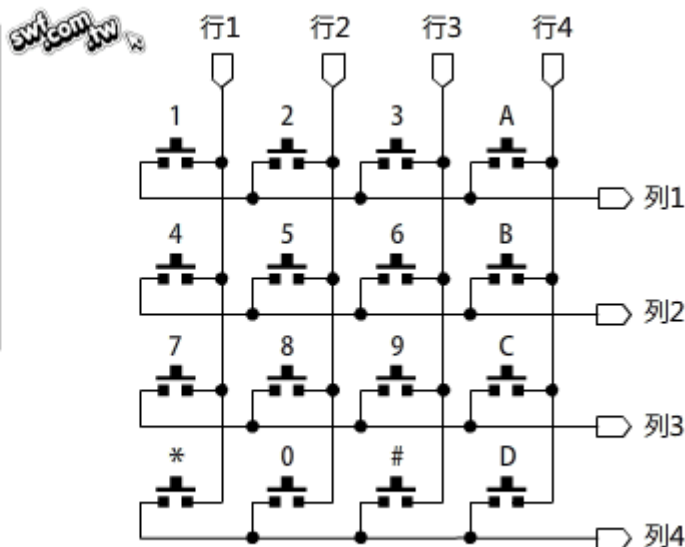
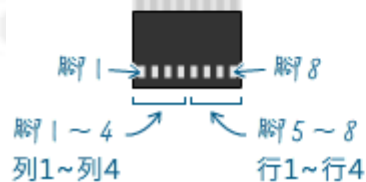


1: 2+3
2: 1+2
3: 2+5
4: 3+7
5: 1+7
6: 5+7
7: 3+6
8: 1+6
9: 5+6
*: 3+4
0: 1+4
#: 4+5

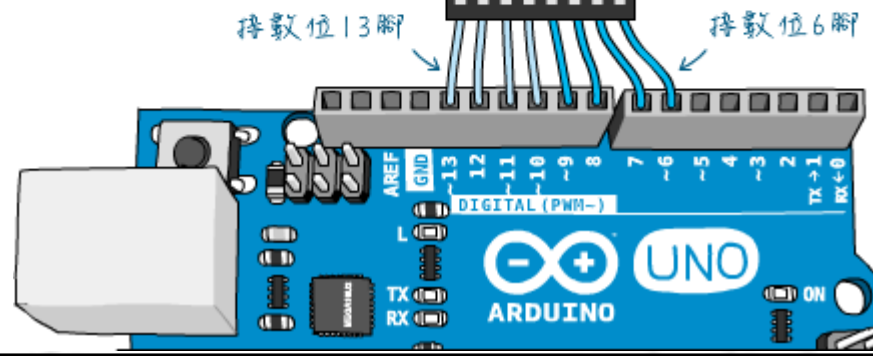


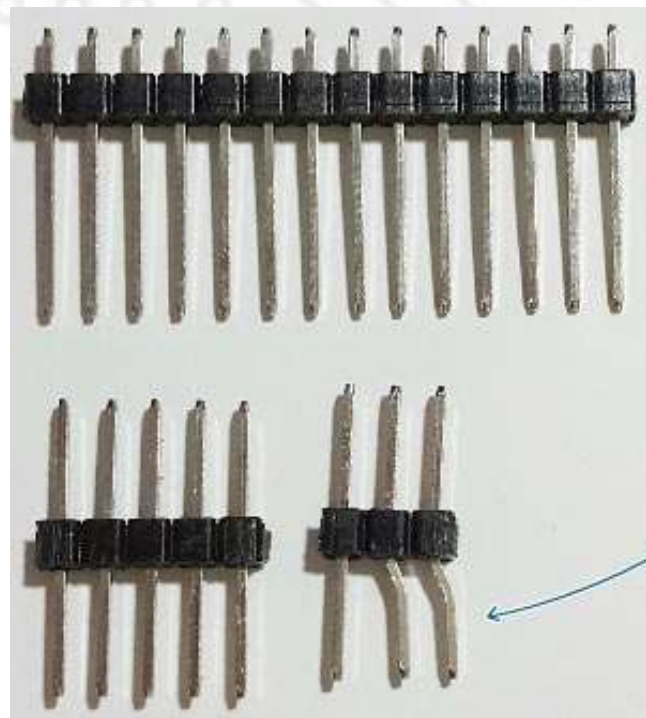
薄膜按鍵模組 (hex keypad)

- 由16個按鍵（開關）交織而成
- 有些按鍵模組直接使用按鍵（微觸）開關組裝，連接電路與程式都相同



接數位13腳 接數位6腳

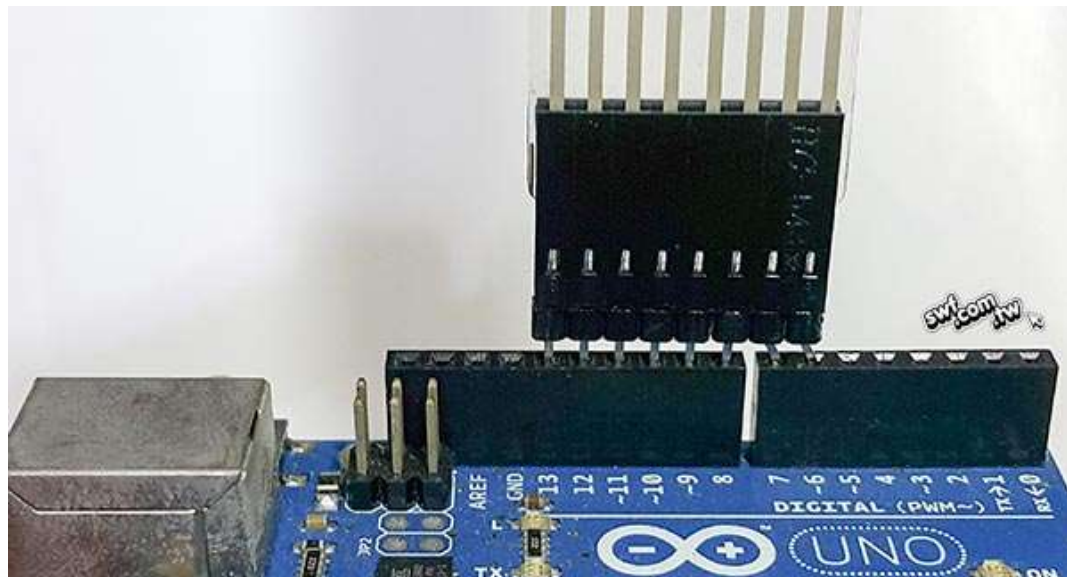




長腳型排針

swi.com.tw

使用尖嘴鉗「整形」
之後的樣子



swi.com.tw

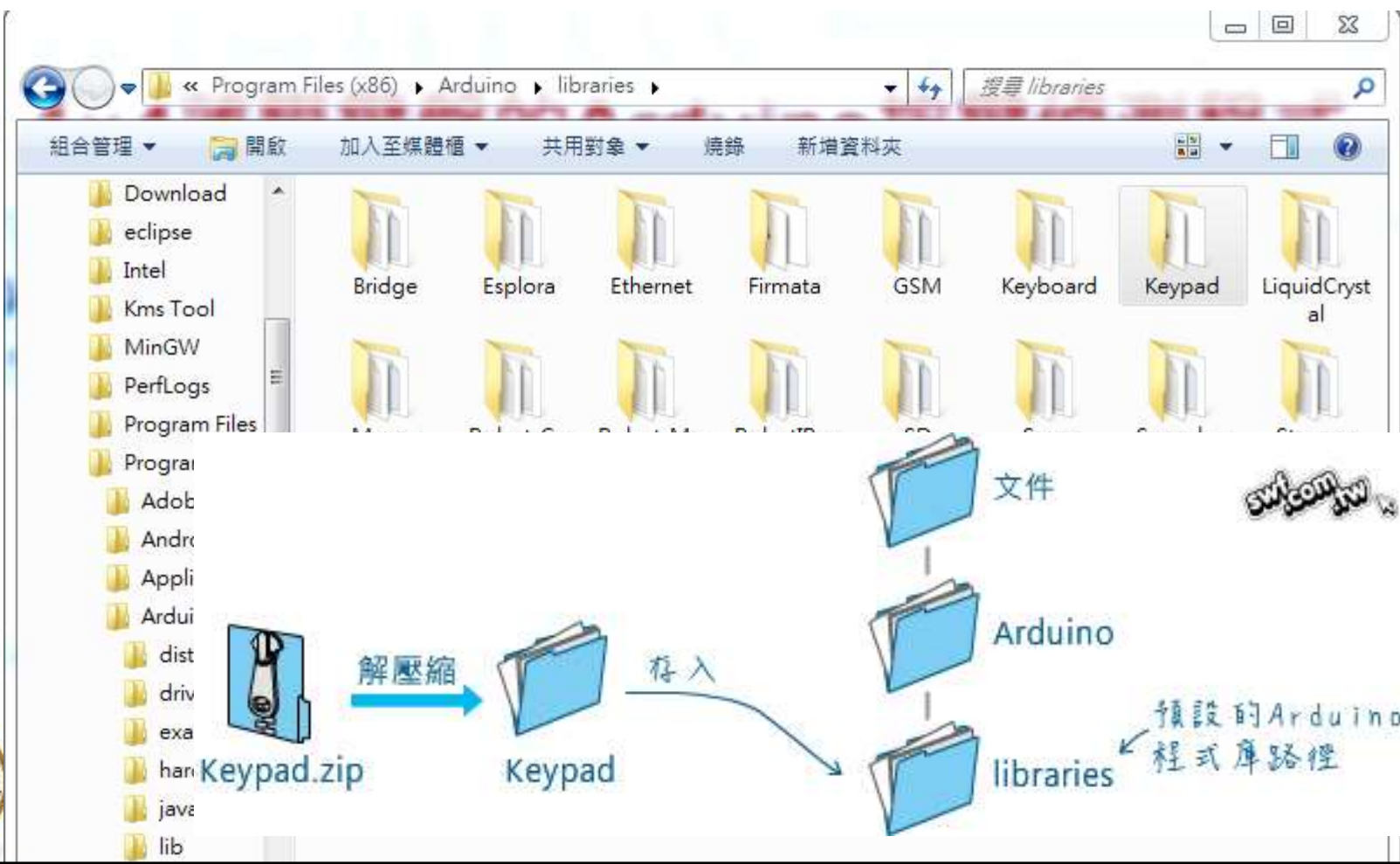


NTU CSIE

4×4薄膜鍵盤的Arduino按鍵偵測程式

下載函式庫後安裝 (官方[範例](#))

<http://arduino.cc/playground/uploads/Code/Keypad.zip>





```
1 #include <Keypad.h>    // 引用Keypad程式庫
2
3 #define KEY_ROWS 4 // 按鍵模組的列數
4 #define KEY_COLS 4 // 按鍵模組的行數
5
6 // 依照行、列排列的按鍵字元（二維陣列）
7 char keymap[KEY_ROWS][KEY_COLS] = {
8     {'1', '2', '3', 'A'},
9     {'4', '5', '6', 'B'},
10    {'7', '8', '9', 'C'},
11    {'*', '0', '#', 'D'}
12 };
13
14 byte colPins[KEY_COLS] = {9, 8, 7, 6}; // 按鍵模組，行1~4接腳。
15 byte rowPins[KEY_ROWS] = {13, 12, 11, 10}; // 按鍵模組，列1~4接腳。
16
17 // 初始化Keypad物件
18 // 語法：Keypad(makeKeymap(按鍵字元的二維陣列), 模組列接腳, 模組行接腳, 模組列數, 模組行數)
19 Keypad myKeypad = Keypad(makeKeymap(keymap), rowPins, colPins, KEY_ROWS, KEY_COLS);
20
21 void setup(){
22     Serial.begin(9600);
23 }
24
25 void loop(){
26     // 透過Keypad物件的getKey()方法讀取按鍵的字元
27     char key = myKeypad.getKey();
28
29     if (key){ // 若有按鍵被按下...
30         Serial.println(key); // 顯示按鍵的字元
31     }
32 }
```

```

1 const byte colPins[4] = {9, 8, 7, 6}; // 設定「行」腳位
2 const byte rowPins[4] = {13, 12, 11, 10}; // 設定「列」腳位
3 const char keymap[4][4] = { // 設定按鍵的「行、列」代表值
4     {'1', '2', '3', 'A'},
5     {'4', '5', '6', 'B'},
6     {'7', '8', '9', 'C'},
7     {'*', '0', '#', 'D'}
8 };

```



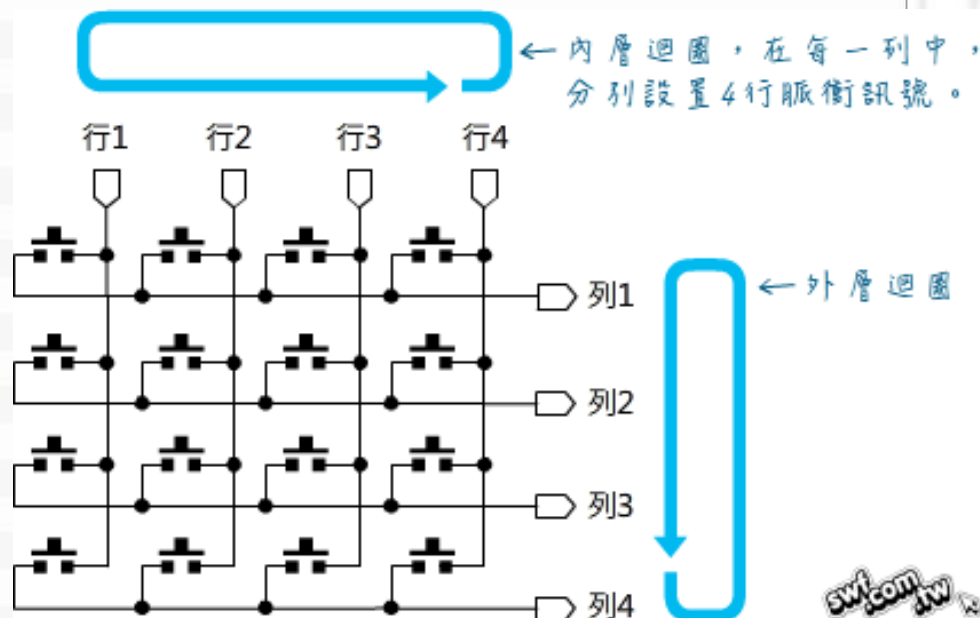
Keypad2.ino

自行撰寫掃描按鍵的程式

```

9
10 byte i, j; // 暫存迴圈的索引數字
11 byte scanVal; // 暫存掃描到的按鍵值
12
13 void setup(){
14     Serial.begin(9600);
15
16     for (i = 0; i <= 3; i++) {
17         pinMode(rowPins[i], INPUT);
18         pinMode(colPins[i], OUTPUT);
19         digitalWrite(colPins[i], HIGH);
20         digitalWrite(rowPins[i], HIGH);
21     }
22 }
23
24 void loop() {
25     for (i = 0; i <= 3; i++) {
26         for (j = 0; j <= 3; j++) {
27             digitalWrite(colPins[j], LOW);
28             scanVal = digitalRead(rowPins[i]);
29
30             if (scanVal == LOW) { // 如果輸入值是「低電位」...
31                 Serial.println(keymap[i][j]); // 輸出按鍵代表的字元
32                 delay(200); // 掃描按鍵的間隔時間
33                 digitalWrite(colPins[j], HIGH);
34                 break; // 跳出迴圈
35             }
36             digitalWrite(colPins[j], HIGH);
37         }
38     }
39 }

```



swf.com.tw




```
int a=0,b=0,num=0,ans;  
char op;
```

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

```
void loop() {  
    // 透過Keypad物件的getKey()方法讀取按鍵的字元  
    char key = myKeypad.getKey();
```

```
    if(key>='0' && key<='9') {  
        num = num*10 + (key-'0');  
        Serial.println(num);  
    }
```

```
    switch(key) {  
        case '*': //清除  
            a = b = ans = num = 0;  
            Serial.println("clear\n");  
            break;
```

```
        case 'A': //A +  
            a = num;  
            num = 0;  
            op = '+';  
            Serial.print(a);  
            Serial.println('+');  
            break;
```

```
        case '#': //# =
```

```
            b = num;
```

```
            switch(op)
```

```
            {  
                case '+':
```

```
                    ans = a + b;
```

```
                    break;
```

```
                case '-':
```

```
                    ans = a - b;
```

```
                    break;
```

```
                case '*':
```

```
                    ans = a * b;
```

```
                    break;
```

```
                case '/':
```

```
                    ans = a / b;
```

```
                    break;
```

```
            }
```

```
            num = ans;
```

```
            Serial.print(a);
```

```
            Serial.print(op);
```

```
            Serial.print(b);
```

```
            Serial.print('=');
```

```
            Serial.println(ans);
```

```
            break;
```

```
    }
```



Keypad_serial_calc.ino

