EduCake 的 Serial 通訊使用

一、 Serial 通訊介紹

前面介紹了各種訊號輸出入控制功能，已經可以利用多樣的感測器輸入、馬達、LED 等零件搭配出不同的花樣；這些都是在控制板上就可以達成的功能，那如果控制板要跟其他裝置，例如另一塊控制板、智慧型手機、電腦等等進行互動呢？讀者應該已經發現，之前的程式練習範例中，已經利用 IDE 上的「Serial Monitor」做過控制板往電腦端回傳數值或訊息進行除錯了，這就是控制板跟外界的主要通訊功能，也是控制板使用上的一大重點；本文將詳細介紹 86Duino EduCake 在「串列通訊埠」方面的功能，並練習相關程式的寫法。

談到「串列通訊埠」(Serial Port，也稱為串列通訊埠)，就得跟「並列通訊埠」(Parallel Port)一起做個比較，如圖 1 的示意圖，串列通訊的概念一般是經由單一條通訊線路，以「分時」的方式一位元一位元依序傳送資料，而並列通訊則以多條線路「同時」傳送多個位元資料；優點當然是並列通訊頻寬較大，而串列通訊較省線路空間，但同時脈下需較長的時間才能跟並列通訊傳送一樣多的資料，另外並列通訊在長距離傳輸時也會有訊號同步的問題；目前電腦硬體相對剛起步的年代已經可以做到高時脈的情況下，串列通訊已經可以達到相當高的傳輸速度，因此電腦周邊通訊接頭已經淘汰並列埠很長一段時間了，例如常見的 USB、SATA、RS232、RS422、RS485、I^2C、IEEE 1394、PCI-E、Thunderbolt 等等都是採用串列通信的方式，接頭與插槽也較不占空間。或許有讀者曾看過早期印表機的 printer port(或

parallel port)的 DB25 接頭，以現在消費性產品講究輕薄短小的標準來看，那可是相當占空間呢。

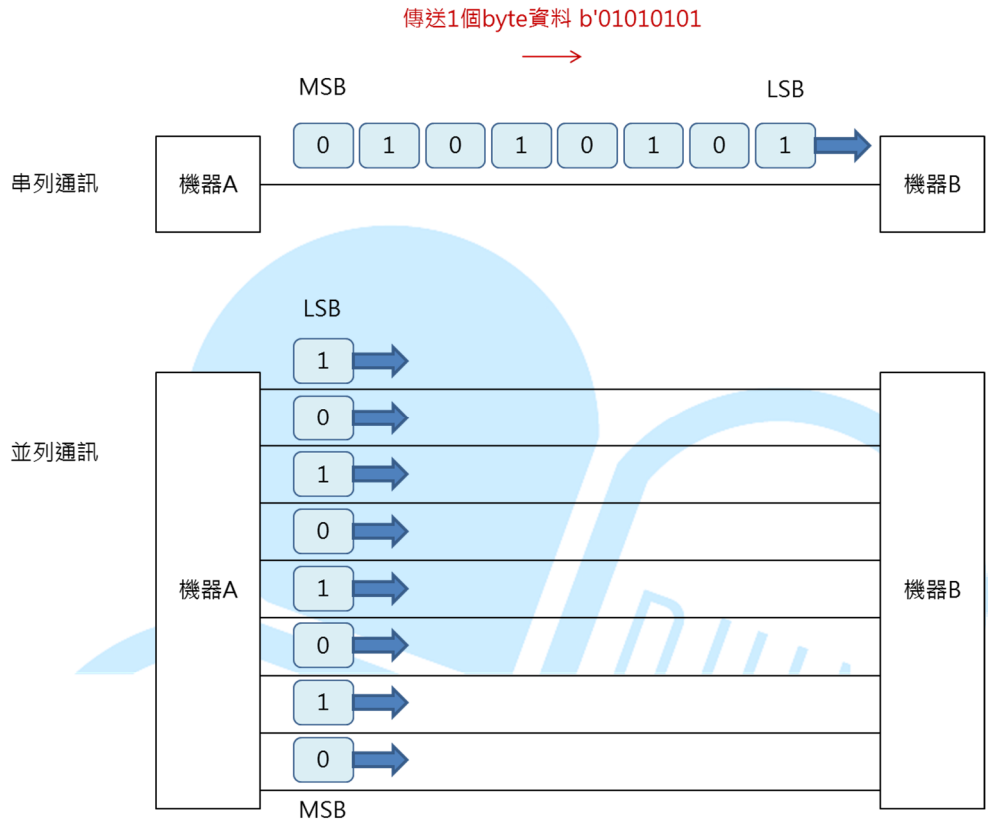


圖 1. 串列埠與並列埠傳輸概念比較圖

串列式通信也不只有單一條線，只是線路的確相對並列通信要少得多，圖 1 部分僅是概念示意，實際上串列通信仍需配合一些功能線才能正常運作（如同步訊號之類）。舉個例來說，之後章節會提到的 I^2C 通信方式，就有一條時脈線（SCL）與一條資料線（SDA），需靠 SCL 的輔助才能取得正確的 SDA 資料時序；而 RS485 則是資料線（D+）與資料線反向訊號（D-）各一條，用以減少雜訊干擾，各種通信規格各有不同的特色。像 RS232、RS422、RS485 這些通信方式各有不同的電器及機械定義標準，但都屬於 UART（通用非同步收發傳輸器，Universal Asynchronous

Receiver/Transmitter），UART 僅是通稱而已。以 86Duino EduCake 的通訊介面也是屬於 UART 的通訊方式，有 TX、RX 線路各一條（也有些地方會標示成 TxD/RxD，是一樣的東西），TX 負責從控制板送出訊號，而 RX 則負責接收外界傳來的訊號。這種 TX/RX 各自有專用線路且可以同時收送的，又稱為「全雙工」式通訊，表示此線路可允許連接兩端的機器同時收送資料；「半雙工」則是線路上同時只能有傳送或接收，不能同時進行，像 I^2C、RS485 通訊就是半雙工的機制；而 86Duino 的 UART、電腦上常見的 RS232 則是全雙工的機制。「全雙工」與「半雙工」可參考圖 2 的說明。

TX/RX 這兩個腳位在這邊使用的是一種稱為 TTL 5V 的電壓位準，表示在通信線上的電壓 LOW/HIGH 在 0~5V 之間；但像 RS232 的邏輯 HIGH(1)則是定義成-3~-15V，邏輯 LOW(0)則是 3~15V。所以當兩個這類型裝置以 UART 互相連接時，需注意通訊腳位的電壓範圍是否相容，且機器 1 的 TX 需連接機器 2 的 RX，機器 1 的 RX 需連接機器 2 的 TX。以 TTL 的 UART 要跟 RS232 通訊為例，通常得用上 MAX232 之類的晶片進行電壓準位轉換才行。

通常具備 UART 的機器，在 TX/RX 端會有各自的暫存緩衝區，且為 FIFO（First in first out，先進入的資料先取出）的機制，緩衝區用在暫存即將送出以及剛接收到的資料，以免 CPU 無法及時處理，如圖 2 所示。此機制之後也會在範例程式中使用，讀者可先了解有此機制的存在；不同機器上緩衝區的大小可能也不同，甚至可能沒有緩衝區，得看各種機器的規格而定。

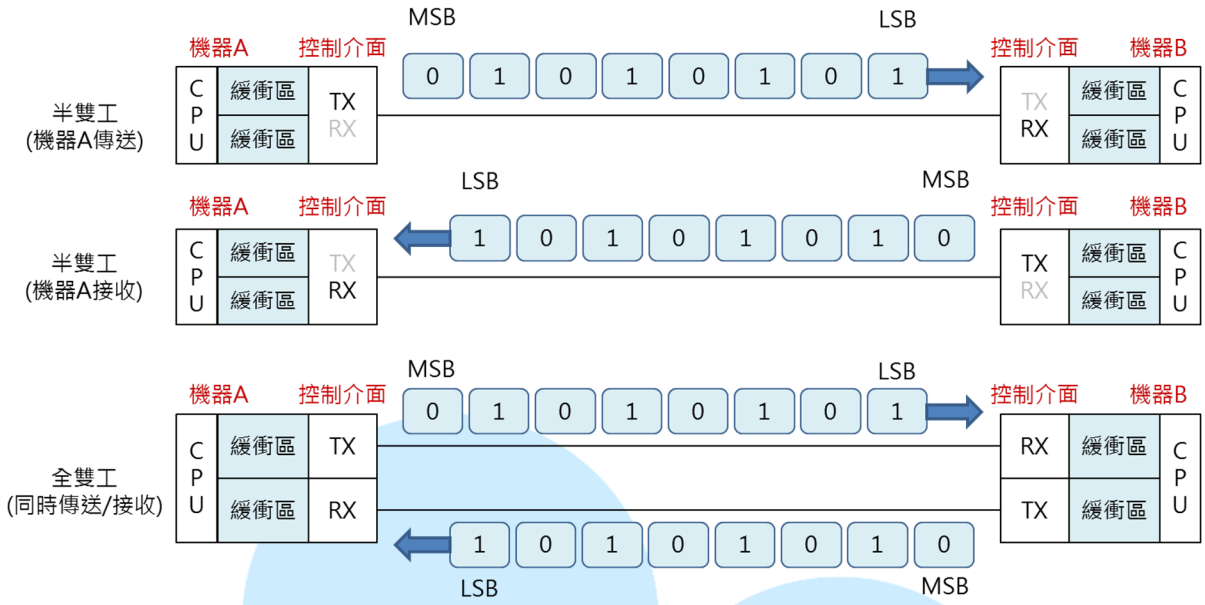


圖 2. UART 全雙工/半雙工機器互連及緩衝區示意圖

UART 還有另一個特點是通訊協定的處理，傳送/接收雙方機器都必須遵守共通的通訊協定以及參數設定，才能正確無誤地與對方溝通。UART 通訊協定如圖 3 所示，一開始會先送出起始位元，接著是資料位元，之後是同位檢查與結束位元，每傳輸一個資料單位（Data Frame）都須遵守這個規則。起始位元用在讓接收方機器判斷何時為 Frame 的起始點，UART 的「非同步 Asynchronous」就是這個意思；其他名詞則在下方作介紹。

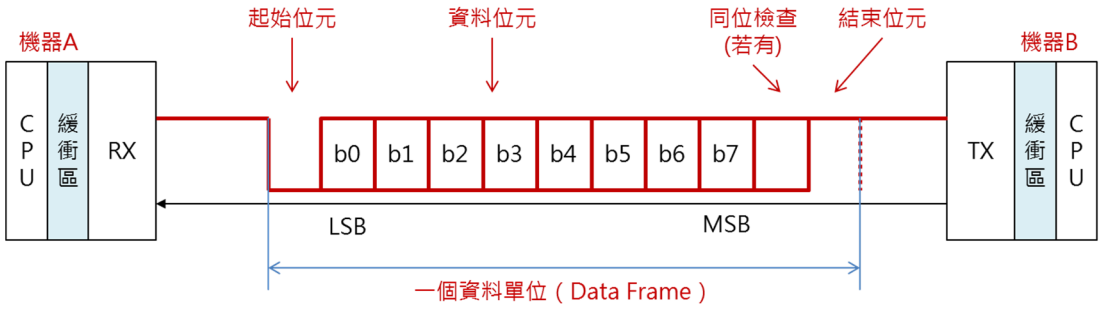


圖 3. UART 通訊協定結構

以一般電腦為例，若讀者已經將控制板接上電腦的 USB 插槽，便能從我的電腦→裝置管理員看到如圖 4 的畫面，由於控制板上已有晶片將 TTL UART 轉換成 USB 訊號，因此電腦便能將控制板辨識為一個虛擬 COM Port（電腦上通用串列埠的一般稱呼）並對他傳輸/接收資料。滑鼠在裝置上按右鍵→內容，便可看到串列埠的參數設定。

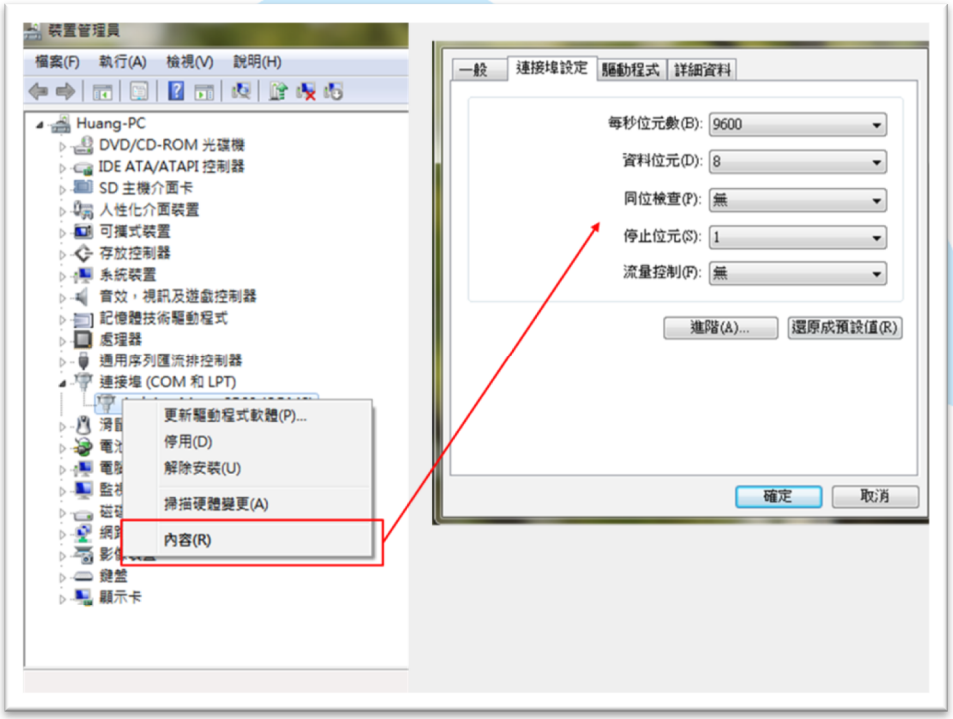


圖 4. 裝置管理員 COM Port 設定畫面

各項參數簡介如下：

- **每秒位元數(也稱為 Baudrate、鮑率)**

  設定串列傳輸的「傳輸速度」，單位為 bit/second（bps）。由於 UART 本身沒有資料同步訊號，僅觀察起始位元作為傳輸開始的信號，因此通訊兩邊機器必須設定為相同的數值，才能用相同的

速度進行傳輸及收訊採樣，否則接收方可能收到錯誤訊息或根本收不到東西；例如 Serial Monitor 的設定跟控制板不一致時，傳回訊息可能看到亂碼這種情況。數字越大表示每秒傳輸的資料量越多，但使用時須注意機器本身的可設定上限，運作速度（時脈）較低的機器，可能只支援 4800 或 9600 的速度。部分較新的機器可能支援自動偵測 Baudrate（Automatic baud rate detection，簡稱 ABR、autobaud）的功能，以機器所載規格為準。

- **資料位元**

設定一個傳輸單位裡面含有幾個位元的資料，通常以傳輸 1 個 byte（=8bits）當作基本單位，所以這裡設定為 8 即可；傳輸時資料位元以先 LSB（最低位元）後 MSB（最高位元）的方式進行傳輸。

- **同位檢查(也稱為奇偶校驗、Parity check)**

此欄位用在檢查資料本身的正確性，有時候當環境本身干擾多、通信線路因老化或接觸不良導致傳輸品質不佳時，可能 A 機器傳了資料位元 b'01010101，結果 B 機器收到了 b'01000101，資料便出現了錯誤。同位檢查可設定為：

1. 無（No Parity）：不進行檢查，訊號中便沒有這個位元。

2. 奇同位（Odd Parity）：當資料位元中有偶數個 1，則同位檢查位元為 1，反之為 0；資料位元與同位檢查位元全部帶有奇數個 1。

3. 偶同位（Even Parity）：當資料位元中有奇數個 1，則同位檢查位元為 1，反之為 0；資料位元與同位檢查位元全部帶有偶數個 1。

4. 標記（Mark）：少用的設定方式，不管資料位元狀況，都把同位檢查位元設定為 1。

5. 空格（Space）：少用的設定方式，不管資料位元狀況，都把同位檢查位元設定為 0。

- **結束位元**

  標示一段資料傳輸的結尾，若傳輸過程中雙方機器出現些微的不同步狀況，也可讓在這段時間內進行時鐘的同步校正。

- **流量控制**

  當資料傳輸時，接收方由於處理速度不足、緩衝區域太小等等原因，導致來不及接收資料，便可以啟動這個功能。當接收端收完資料便會告知傳送端自己已經收完資料，讓傳送端可以繼續下個資料的傳送。

一般預設參數通常是：資料位元：8、同位檢查：無、停止位元：1、流量控制：無，至於每秒位元數就得要使用者自己設定了。使用 UART 通

訊時，需特別注意接線是否正確，以及雙邊參數是否都設定正確，通訊有問題時可先從這邊著手解決。上述通信機制看起來是不是有點複雜呢？沒關係，幸好 86Duino EduCake 的 Code100 IDE 裡面已經有包裝好的函式庫可以方便直接使用，也就是接下來範例程式練習會使用到的「Serial Library」，以下程式範例練習，參數都設定為：資料位元 8、無同位檢查、停止位元 1、無流量控制、鮑率 115200 bps。接下來就讓我們以範例進行練習吧~

二、 第一個程式 – 練習 Serial.write( )

前面提到的資料位元，雖然在電路上都是 HIGH(1)/LOW(0)的訊號，但程式在傳送跟接收資料時其實可以用不同的解讀（或解碼）方法，分為「數字」跟「字元」兩種。數字的解讀方式便是直接以二進位、十進位、十六進位等方法直接解讀；「字元」的編碼/解碼方法則是依照「ASCII 表」的定義，十進位 0~127 各代表不同的「符號」，ASCII 表的資料在網路上很容易找到，讀者可以直接搜尋相關資料閱讀。例如資料位元傳送 b'01000001，以「數字」方式解讀就是十進位的數字 65，以「字元」方式解讀則是符號「A」。

第一個範例程式，我們先做 86Dino EduCake 與電腦雙方進行單一 byte，且以「數字」作編碼/解碼的練習。以 86Duino EduCake 的程式寫作為例，讀者請打開 86Duino Coding 100 IDE，輸入以下程式碼：

```
void setup()
{
  Serial.begin(115200);// 設定 Serial Port 的 Baudrate
}

void loop()
{
  if(Serial.available())
  {
    int value = Serial.read();// 以數字方式解讀收到的資料
    delay(100);
    Serial.write(value + 1);// 以數字方式回傳收到的資料
  }// end if(Serial.available()

}
```

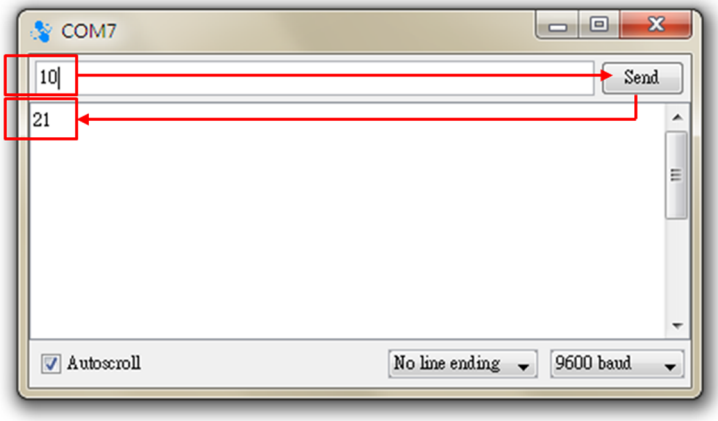
此範例程式功用為：當使用者傳輸一個 byte 的資料給 86Duino EduCake 時，EduCake 會將收到的數值以數值方式解釋，加上 1 之後回傳。

一開始在 setup( )裡，先以 Serial.begin(115200)進行 Serial Port 的 Baudrate 設定；接著 loop( )裡以 Serial.available( )檢查是否 Serial port 有接收到數值，若有接收到資料時，Serial.available( )會回傳目前緩衝區內暫存的資料 byte 數量。

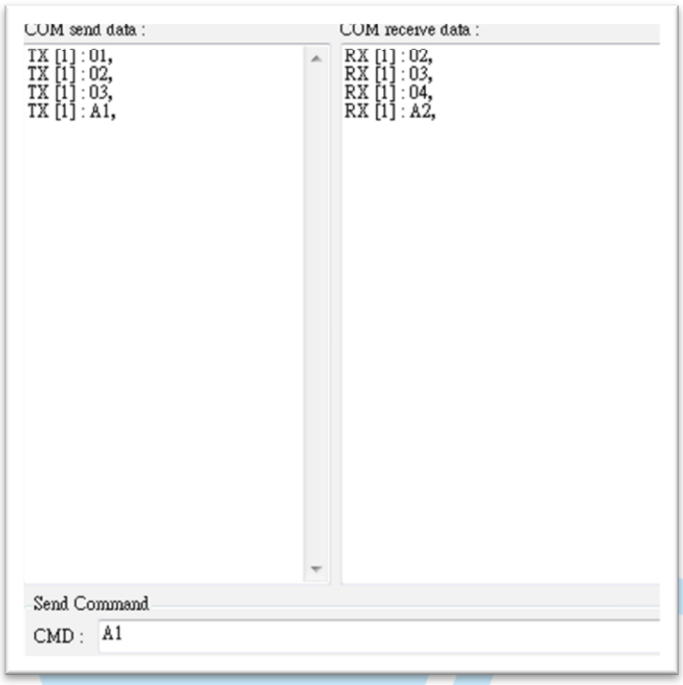
由於以「數值」方式做解碼，因此宣告 byte value 變數進行接收（也可以使用 unsigned int、int 等進行宣告，視使用者數值傳換狀況而定），以 Serial.read( )做讀取。讀取完資料後，暫停一小段時間把收到的數值加上 1 再送出(此處 delay(100)只是用來看出延遲傳送的效果，也可以不加延遲)，

傳輸資料則是使用 Serial.write( )指令。注意 Serial.read( )、Serial.write(byte)指令都是以 1 個 byte 當作單位，傳送與接收資料時須留意資料內容，以免超出 1 個 byte 可表示的數字範圍（0~255）。

此範例暫時不使用 Serial Monotor，因為 Serial Monitor 預設會使用「ASCII 字元」方式進行傳送與接收解碼，當使用者從上方輸入框輸入任何數字或文字，Serial Monitor 會將輸入的資料分開為各別字元傳送出去，例如當輸入「數字 10」，實際上 Serial Monitor 會送出代表'1'、'0'字元的兩個資料，在 ASCII 表中分別為十進位 49 跟 48，因此會看到 EduCake 分別回傳了 49+1=50、48+1=49 兩個資料回來，被 Serial Monitor 解讀為'2'、'1'，跟原本預期的數字 10+1=11 大不相同，如下圖所示：



這邊以一個電腦端 UI 來作範例如下：此介面純粹以數值方式做傳送與接收（以 16 進位方式呈現），當傳送 1 時收到 2，傳送 2 收到 3...傳送 A1 收到 A2，依此類推。



三、 第二個程式 – 練習 Serial.print( )

接著這個範例程式我們使用字元的方式做練習，請在 Coding 100 IDE

輸入以下程式：

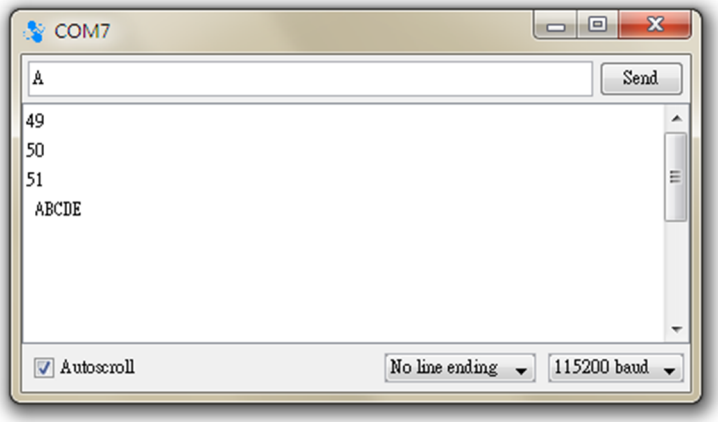
```
void setup()
{
  Serial.begin(115200);// 設定 Serial Port 的 Baudrate
}

void loop()
{
if(Serial.available())
  {
    byte value = Serial.read();// 以數字方式解讀收到的資料
    delay(100);

    if(value == 'A')
    {
```

```
            Serial.write(32);// space
            Serial.write(65);// A
            Serial.write(66);// B
            Serial.write(67);// C
            Serial.write(68);// D
            Serial.write(69);// E
            Serial.write(10);// \n
        }
        else
    {
            Serial.println(value);//  以數字方式回傳收到的資料
        }
      }// end if(Serial.available()
    }
```

接著上傳程式至 EduCake 後，開啟 Serial Monitor，此範例便可使用 Serial Monitor 做通訊。此程式功能為，當使用者由輸入框輸入字元'A'，則 EduCake 回傳十進位 32、65~69、10，而 Serial Monitorg 收到資料後會以字元方式解釋這幾個資料，這幾個資料在 ASCII 表中代表了符號'空白'、'A'、'B'、'C'、'D'、'E'、'換行'顯示在視窗中，注意 ASCII 表格中十進位 0~31、127 是「控制字元」，表示這些符號是無法顯示的，而是執行某些動作，例如範例中 Serial.write(10)便可讓 Serial Monitorg 視窗游標換到新的一行（讀者也許會在其他程式語言書中看到'\n'，就是此處的換行控制字元），表中 32~126 則是「可以顯示的字元」，像是'0'~'1'、'a'~'z'、'A'~'Z'等符號。如果使用者不是傳送字元'A'，則傳送其他資料都會讓 EduCake 執行 Serial.println(value)指令，此指令會將「value 的十進位值」以字元方式回傳給 Serial Monitor，且換到新的一行(Serial.print( )功能相同只是不換行)；

Serial.print( )、Serial.println( )也是之前章節用來除錯或印出變數內容的語法。當使用者由 Serial Monitor 送出 1，實際上 EduCake 執行 byte value = Serial.read( )後收到的是 ASCII 表中代表字元 1 的十進位數值「49」，而 Serial.println(49)則會傳出代表十進位值 49「字元」的兩個 byte 十進位「52」、「57」給 Serial Monitor，因此視窗上會顯示 49 兩個字元；這個範例程式也可以用來觀察各個「可顯示字元」實際在 ASCII 表中對應的十進位值是多少。

實際測試如下圖，傳送'1'、'2'、'3'後分別收到字元對應的十進位值 49、50、51，傳送'A'則收到空白 ABCDE 並換到下一行。



由上面兩個範例應該不難看出 Serial.write(value)跟 Serial.print(value)兩個指令的差別，Serial.write( )是傳送出 value 這個 byte，接收端收到的一樣是 value 這個資料，只是接收端選擇用數字或符號去解釋這個資料而已；而 Serial.print( )則是把 value 所代表的數值符號以 ASCII 對應碼傳送出去，接收端也要以 ASCII 字元方式解釋才能看出收到資料的意義。

Serial.print( )也有不同的參數可用，例如 Serial.print(value, DEC)是以 10 進位顯示 value 的數值字元（預設就是十進位，因此第二個參數欄可以不輸入 DEC）、Serial.print(value, BIN)是二進位顯示、Serial.print(value, OCT)是八進位顯示、Serial.print(value, HEX)是以十六進位顯示等等，讀者可以嘗試看看不同參數的效果。Serial.println( )不加任何參數則是單純換行，對於以字元解釋資料時與 Serial.write(10)功能相同

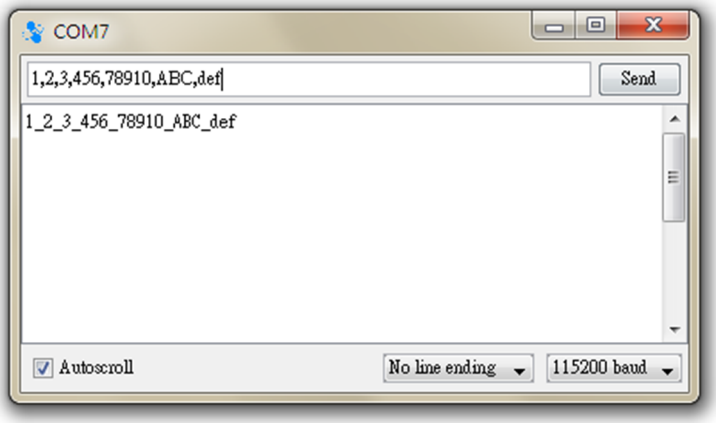
四、 第三個程式 – 傳送/接收多個 byte(數字)

練習完傳送單一個 byte 資料後，接著來練習傳送/接收多個 byte 的狀況，也是多數專題應用時會碰到的狀況。請在 Coding 100 IDE 輸入下列程式碼：

```
void setup()
{
  Serial.begin(115200);// 設定 Serial Port 的 Baudrate
}

void loop()
{
  int byteCount = Serial.available();// 取得緩衝區內的資料 byte 數量
  if(byteCount > 0)
  {
    byte RX_buff[byteCount];// 宣告陣列，用來複製緩衝區內的資料
    for(int i = 0; i < byteCount; i++)
    {
      RX_buff[i] = Serial.read();// 將收到的所有資料存到陣列裡
    }

    for(int i = 0; i < byteCount; i++)
```

```
        {
            if(RX_buff[i] != 44)//  當暫存區的數值不是 44(字元',')
            {
               Serial.write(RX_buff[i]);
            }
            else
            {
               Serial.write(95);// '_'
            }
          }

          Serial.println();//  換行

        }// end if(byteCount > 0)

    }
```

接著上傳程式至 EduCake 後，開啟 Serial Monitor。此程式功能為，當使用者在輸入框輸入一連串字元，其中可包含逗號','，則 EduCake 會將符號','替換為符號'_'並依原本順序傳回給 Serial Monitor，如下圖。



loop( )中先使用 int byteCount = Serial.available( )檢查目前 Serial 緩衝區內有多少資料，如果 byteCount 大於 0 表示緩衝區內有資料，則宣告一個用來接收 Serial Port 暫存區資料的 byte 陣列名為 RX_buff，陣列長度為 byteCount，然後使用 for 迴圈配合 Serial.read( )將緩衝區數值複製進陣列裡，其實此處的 for 迴圈也可以用 Serial.readBytes(RX_buff, byteCount)指令取代，可以讓程式碼變得更簡潔。前面提到 Serial Port 有 FIFO（先入先出）的暫存緩衝區，運作機制是每執行一次 Serial.read( )這個指令，緩衝區內就會少一個 byte 的資料。接著使用另一個 for 迴圈並檢查 RX_buff 內容，當內容不是十進位值 44（表示原本收到字元',' ），則回傳原本內容，反之以 Serial.write(95);// '_'替換為底線字元。

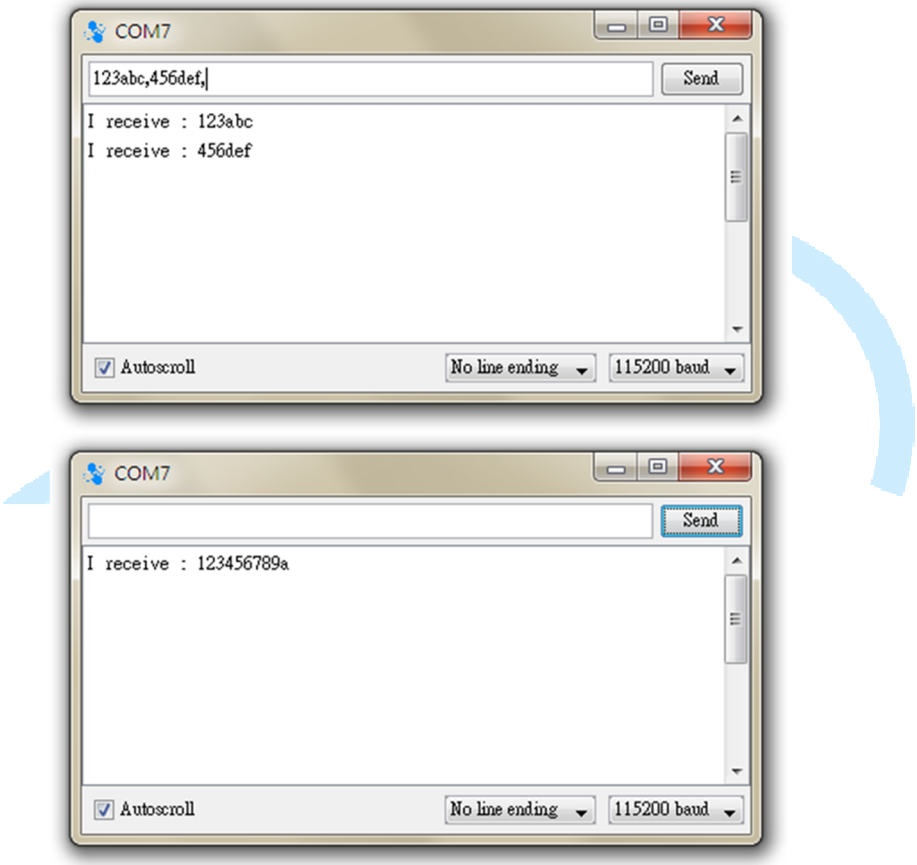
此程式需特別注意一個地方，由於 loop( )內程式碼不多執行時間相當快，當 loop( )沒有加上適當的延遲時間時，若使用者以較緩慢的速度傳送了 10 個 byte，實際 byteCount = Serial.available( )不一定會得到 10，可能好幾個 loop( )才會收完總共 10 個 byte，但不影響目前這個範例的執行效果。

五、 第四個程式 – 傳送/接收多個 byte(字串)

86Duino EduCake 的 Serial Port 還有一些其他函式可以用，適合用於雙方以字串互相傳輸的狀況，請讀者輸入以下程式：

```
void setup()
{
  Serial.begin(115200);// 設定 Serial Port 的 Baudrate
  Serial.setTimeout(2000);// 設定 Serial.readBytesUntil();最長等待時間，單位 ms
}

void loop()
{
  char RX_buff[10];// 宣告陣列，用來複製緩衝區內的資料
  int byteCount = Serial.readBytesUntil(',', RX_buff, 10);// 讀到','或讀了 10 個才結束等待
  //Serial.print("byteCount : ");
  //Serial.println(byteCount);

  if(byteCount > 0)
  {
    Serial.print("I receive : ");
    for(int i = 0; i < byteCount; i++)
    {
      Serial.write(RX_buff[i]);
    }
    Serial.println();// 換行
  }// end if(byteCount > 0)

  delay(100);// 等回傳完
  Serial.flush();// 清空暫存區域
}
```

上傳程式至 EduCake 後，開啟 Serial Monitor。此程式功能為，當使用者輸入一連串字元，其中包含字元','時，EduCake 便會讀取到此字元前為止，並回傳「I receive：讀到的字串」；或者如果沒有讀到','但先接收到了 10 個字元，也會中止讀取並回傳，執行效果如下圖。



由於程式碼裡使用了 Serial.readBytesUntil(',', RX_buff, 10)指令，此行程式會讀取 Serial Port 至讀取到字元','，或讀取了 10 個 byte 為止，如果沒有設定適當的截止時間，當外部裝置沒有送出字元','，或送出至少 10 個 byte 的資料，那麼程式可能就卡在這一行再也不會往下執行了。因此一開始在 setup( ) 裡先使用 Serial.setTimeout(2000) 用來設定 Serial.readBytesUntil( )最長等待時間，單位是 ms。Loop( )裡先宣告 char 型態的矩陣 RX_buff，長度 10（長度大於 readBytesUntil 的設定即可）；

int byteCount = Serial.readBytesUntil(',', RX_buff, 10)可以取得實際上讀到了幾個 byte 的資料，如果 byteCount 大於 0 才做回傳。

Serial.print("I receive：")指令可以把一連串字元以 ASCII 編碼傳送出去，注意程式中字串需用雙引號頭尾包夾供 IDE 辨識為字串資料，實際傳送的資料不會有雙引號在內。或者讀者也可以把這串資料拆開成一個個字元傳送，字元則是用單引號包夾：

```
Serial.print('I');
Serial.print(' ');
Serial.print('r');
Serial.print('e');
Serial.print('c');
Serial.print('e');
Serial.print('i');
Serial.print('v');
Serial.print('e');
Serial.print(' ');
Serial.print(':');
Serial.print(' ');
```

也是一樣的效果，但就麻煩很多了對吧?

程式碼最後面使用了 Serial.flush( )指令，由於使用者可能一次傳送超過 10 個 byte，為了下一次不會讀到剩下的資料，因此使用這個指令來清空緩衝區，須注意傳送與接收緩衝區都會被清空，所以在執行這一行之前會先加上一點延遲時間，讓前面的 Serial.write( )、Serial.print( )可以傳完資料。若沒有加上些微延遲時間，會發現應該回傳的資料沒有傳完就中斷了，讀者不仿試試看。

六、 第五個程式

最後一個範例程式使用上面的幾個函式配合幾個新函式做點有趣的功能，讓 86Duino EduCake 變成一個以串列通訊指令操作的計算機，讀者請輸入以下程式碼：

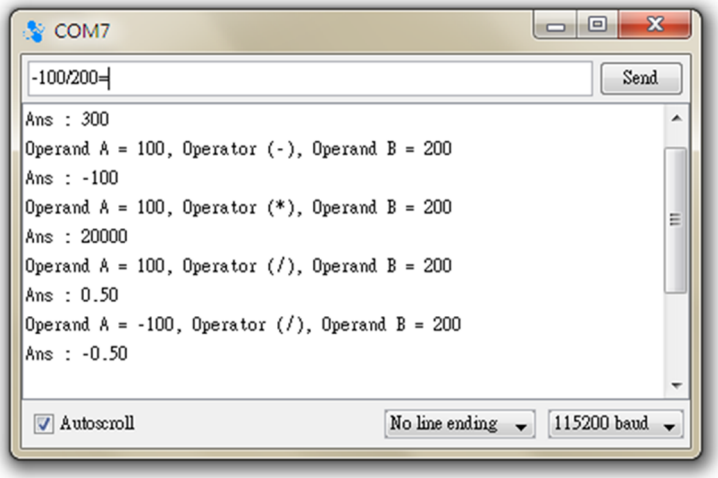
```
int Op_A = 0;// 運算元 A
int Op_B = 0;// 運算元 B

void setup()
{
  Serial.begin(115200);// 設定 Serial Port 的 Baudrate
  Serial.println("Please enter format : Operand A +(or -) Operand B =");// 傳送指示字串
  Serial.setTimeout(1000);// 設定 Serial.parseInt ();最長等待時間，單位 ms
}

void loop()
{
  // 讀取運算元 A
  Op_A = Serial.parseInt();
  Serial.print("Operand A = "); Serial.print(Op_A);

  // 讀取運算子
  char opr = Serial.read();
  Serial.print(", Operator ("); Serial.write(opr); Serial.print(")");

  // 讀取運算元 B
  Op_B = Serial.parseInt();
  Serial.print(", Operand B = "); Serial.println(Op_B);
```

```
        if(Serial.read() == '=')
        {
          Serial.print("Ans : ");
          switch(opr)
          {
            case '+':
              Serial.println(Op_A + Op_B);
              break;

            case '-':
              Serial.println(Op_A - Op_B);
              break;

            case '*':
              Serial.println(Op_A * Op_B);
              break;

            case '/':
              Serial.println((float)(Op_A) / (float)(Op_B));
              break;

            default:
              Serial.println("illiegal operator !!!");
              break;
          }// end switch
        }// end if(Serial.read() == '=')
    }
```

上傳程式至 EduCake 後，一樣開啟 Serial Monitor。此程式功能為，

當使用者從 Serial Monitor 輸入此格式:[整數 A][+-*/運算符號][整數 B][=]

（其中不含空格符號）然後按下傳送，則 EduCake 會將收到指令及運算結果回傳，整數可以是正或負數；如下圖所示：



程式碼一開始先宣告兩個 int 型態變數 Op_A 跟 Op_B，用來儲存運算元的數值。

setup( )部分與範例四相同，loop( )流程一開始讀取運算元 A，這裡使用了 Serial.parseInt( )這個指令，此指令會以「字元解釋」方式依序檢查 Serial 緩衝區的資料，直到找到不是數字的字元為止（所以跟 Serial.readBytesUntil 一樣要設定 Serial.setTimeout），然後將找到的一連串「數字字元」轉換成 int 型態；例如收到了字元'1'、'2'、'3'、'+'、'1'，則會轉換出十進位整數「123」，緩衝區剩下'+'、'1'，注意如果是收到'-'、'1'、'2'、'3'、'+'，則 Serial.parseInt( )指令會把第一個'-'字元算在一起，變成十進位整數「-123」。

中間使用 char opr = Serial.read( )讀取運算子的符號，接著一樣用 Op_B = Serial.parseInt( )讀取運算元 B 的整數值。最後剩下一個符號，使用 if(Serial.read() == '=')檢查是否是'='符號，如果是等於符號就可以開始

做數學運算。這個範例程式支援「加減乘除」四則運算，為了程式閱讀方便這裡使用 switch-case 語法做運算符號的判斷，如果前面收到的運算符號是'+'，則把運算元 A、B 相加後回傳，其他算法依此類推；運算子'/'的部分計算處理稍有不同，是因為整數相除可能出現小數點，所以計算前先把運算元 A、B 轉成 float 型態再做除法，不然整數直接相除後小數點可是會消失喔。Switch-case 注意要寫 default，以免使用者輸入了不支援的運算符號。

讀者也可以試著把運算元 A、B 用 float 型態宣告，然後把 Serial.parseInt( )改成 Serial.parseFloat( )，這樣程式就可以支援小數點運算囉。

http://arduino.cc/en/Reference/Serial

http://arduino.cc/en/Serial/ParseFloat

http://arduino.cc/en/Serial/ParseInt

http://arduino.cc/en/Serial/ReadBytes

七、 第六個程式 GPS 的實作

本章節最後，我們來實作一個 GPS 的簡單應用，看看現在已經普及到生活中的 GPS 到底是如何結合 EduCake 來發揮他的功能。

全球定位系統（Global Positioning System，簡稱 GPS），又稱全球衛星定位系統，這是美國國防部研製和維護的中距離圓型軌道衛星導航系統。涵蓋地球表面絕大部分地區（98%），可以提供準確的定位、測速和高精度的時間，搭載非常精準的原子鐘對衛星時間作校準，至少遠超過我們平時使用的手錶和時鐘，也可以順便用來幫 EduCake 作計時器和取得時鐘時間的功能。詳細的 GPS 原理就跳過了，有興趣的讀者可以 GOOGLE 或是到維基百科查詢 GPS 或是全球定位系統，就可以找到很多相關的原理和知識，這裡直接看來實作。

首先，想辦法取得一個 GPS 模組，一般拍賣都有很多，現在市面上的 GPS 模組的輸出訊號多數都 RS232/TTL 規格，也有 I2C 規格或是 SPI，但不管哪個規格，EduCake 都能處理，筆者選用最簡單的 TTL 規格版本 (NEO-6M)，如圖 1 所示

圖 1. TTL 訊號輸出的 GPS 模組

這種 GPS 模組一個約數百台幣就可買到，接線很單純，由右至左分別是: NC(懸空不接)、VCC(給 3.6~5.2V，這可直接接 EduCake 的 5V)、RX(UART 接收命令用，像是設定傳輸速度之類的指令)、TX(UART 資料輸出)、GND、1PPS(定時脈波，週期一秒)，輸出資料的速度是 1~5Hz。

GPS 模組有幾個狀態需要了解一下，第一個是冷起動，這是指從完全斷電的情況下通電，GPS 模組必須初始化、抓到衛星、重新計算目前的相關資訊等等，這些動作是需要一點時間的，通常至少是二三十秒的時間，實際的行車導航會遠超過這個時間，因為還要重新產生附近的圖資、路徑規畫等資訊，會有數十秒~一兩分鐘以上，且這是在有抓到 3 顆以上衛星的情況下才行(這跟 GPS 運作原理有關)；第二個是溫啟動，待機情形下繼續工作，約需要 1~數秒；第三種情況是重新捕捉衛星，這最容易發生在城市大樓密集處、橋下、進出地下停車場、進出隧道時，因為衛星訊號在這些地方會微弱或是根本收不到，但再次收到衛星資訊時，GPS 會和 buffer 裡面存下的之前的資訊來比對，迅速修正目前位置，通常這個動作約一秒內。

接線方式如圖 2，主要只接 5V、GND、TX 就好，其他的不需要接，若需要取得精確的一秒鐘，那就多接 1PPS 那條線就好，可以獲得週期一秒，HIGH 時間 100ms 的固定脈波，板子上的標示很清楚，應不會有接錯的疑慮才是

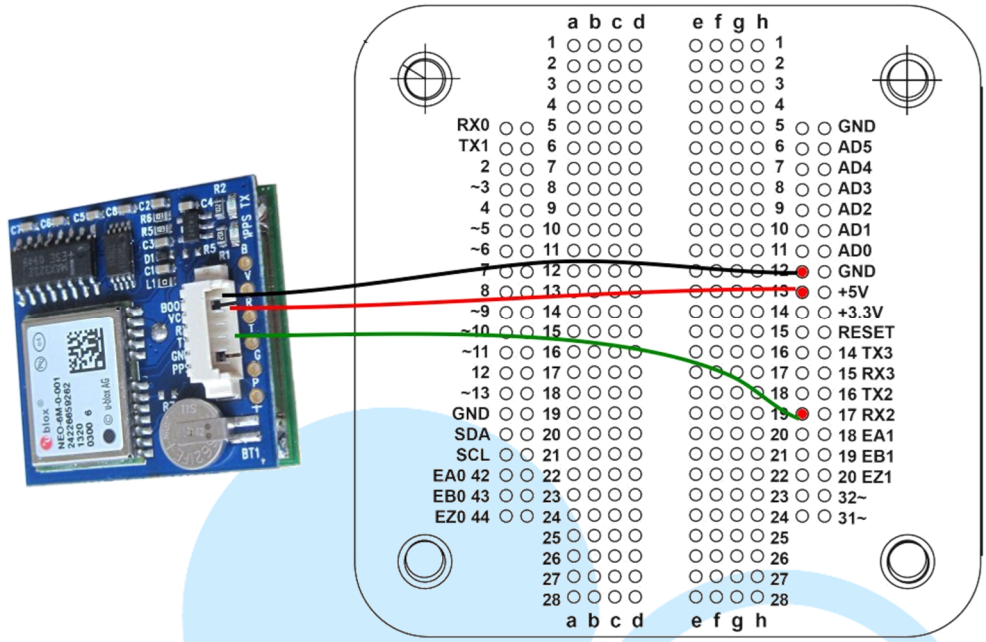


圖 2. GPS 模組接線方式

接好線以後，Educake 裡面放入以下程式碼

```
void setup()
{
  Serial.begin(9600);
  Serial2.begin(9600); //GPS 接在這裡
  Serial.println("Begin");
  delay(2000); //  這裡需要頓一下，GPS 開機初始化需要時間
}

void loop(){
    char ch;
    if ( Serial2.available()) //  有收到東西時
    {
     Serial.println("Get:");
     while (Serial2.available()) //就開始一直收，直到沒東西
     {
      ch=Serial2.read();
      Serial.print(ch); //並且一直印到畫面上
     }
    }
}
```

這個程式只是用來把 GPS 接收到的東西原封不動印出來 Serial Monitor 看，有助於我們先了解到底會出現什麼，結果會看到一堆類似底下字串的東西，這種東西就是 GPS 模組抓到的衛星定位資訊，有很多種，我們需要用程式一行一行來解析這些字串以便取得想要的資訊，底下是 GPGGA 類型的封包

$GPGGA,073849.00,2301.37777,N,12012.94773,E,1,09,0.098,41.2,M,-2.5,M, ,*42

這行字裡面的每一個資訊都用小寫的逗號"，"隔開，分別代表以下意思

$GPGGA	GGA 規範的訊息開頭
073849.00	代表時時分分秒秒.秒秒，所以這是說國際標準時間早上 7 點 38 分 49.00 秒的 UTC 時間，台灣來說，必須加上八小時，所以這代表台灣時間 15 點 38 分 49 秒才對
N	代表北半球，南半球就會是 S
2301.37777	代表緯度 度度分分.分分分分分
E	代表東半球，西半球就是 W
12012.94773	代表經度 度度度分分.分分分分分
1	定位代號，0 代表無效、1 代表已定位、2 代表偏差修正、3 是軍用格式(一般這是收不到的，收到也解不出來)
09	抓到 9 顆衛星，一般低於 3~5 顆就根本沒準確度可言了
0.098	水平精確度，實測覺得這應該不可能，哪會這麼準，這數值筆者測試約會在 0.08~15 左右跳來跳去，當作參考就好，主要是衛星數量，數量越少就越不準；附近是否空曠造成的影響最多，若有很多高樓大廈，會因為衛星訊號的反射折射等因素，就算抓到很多顆也會有比較大誤差
41.2	離海平面的高度
M	單位是公尺
-2.5	地表平均高
M	單位公尺
*42	checksum 檢查碼，確認這則 GPS 訊息是否有誤

基本上有收到這個已經所有想要的資訊都有了，但 GPS 還有很多別的訊息是可以使用的

以下的這兩個就是

$GPGLL, 2301.37777,N, 12012.94773,E, 073849.00,A,A*62

這是 GLL 封包，裡面主要只有經緯度和 UTC 時間的資訊，因為解釋方式都和上面一樣就不重複，請自行查詢上面的表格。

$GPRMC,073141.00,A,2301.37777,N,12012.94773,E,0.016,0,190214,,,A*77

這是 RMC 封包，跟上面的也是很像，但其中比較不一樣是第三個參數 A，代表這訊息是可用的，若為 V 就代表這個訊息不正確，可能是衛星數太少或是訊號微弱不足以算出位置；經度資訊後面那個 0.016 代表速度，單位是節(1 節=1 海浬/hr=1.852 公里/hr)；"0" 代表對地方向；190214 是日期，代表 2014 年 2 月 19 號；再來的參數是磁極的偏移量和方向，不過這裡都沒資訊；A*77 是 checksum 檢查碼。

GPS 其實很有很多種別的封包，不過因為有些資訊牽涉很複雜的數學驗證，不適合這裡討論，有興趣的讀者可以參考 http://goo.gl/QSfSsS，另外，經緯度的定義參考 http://goo.gl/rDvgzl ，

.E7.B6.93.E7.B7.AF.E5.BA.A6.E8.A1.A8.E7.A4.BA.E5.8F.8A.E8.BD.89.E6.8F.9B， 1 度是 60 分，1 分是 60 秒，截取到的經緯度必須經過這個換算才能變成真正 GOOGLE MAP 能接受的座標。組合以上封包能獲得的資訊，其實就能截取到目前所在位置的: 經度、緯度、速度、高度、日期、原子鐘等級的精確時間、方向等等的資訊，底下就來寫一個真正的範例，把這些資訊從 GPS 傳回的封包裡面抓出來吧。

另外為了能比較直覺的獲得資訊，也使用 16*2 LCD 來協助顯示到液晶模組上面，這樣即使不接電腦，也可直接使用。

```
#include <LiquidCrystal.h>

// rs 接 digital 3
// rw 接 digital 4
// enable 接 digital 5
// d4, d5, d6, d7 接 digital 6、7、8、9
LiquidCrystal lcd(3,4,5, 6,7,8,9);
int LightBri = 32;   //對比控制接 digital 32

String data="";
int mark = 0;
boolean Start_Flag=false;
boolean valid=false;
String GGAUTCtime,GGAlatitude,GGAlongitude;
String GPStatus,SatelliteNum,HDOPfactor,Height;
String PositionValid,RMCUTCtime,RMClatitude;
String RMClongitude,Speed,Direction,Date,Declination,Mode;

void setup()
{
  pinMode(LightBri, OUTPUT);
  analogWrite(LightBri, 40);   // 設定對比，越低螢幕顯示的字會越黑
  lcd.begin(16,2);
  lcd.clear();
  lcd.setCursor(0,0);   //設定游標在第一列開始印
  lcd.print("DMP Demo");

  Serial.begin(9600);
  Serial2.begin(9600); // GPS   連接在這裡
```

```
        Serial.println("Begin");
      // 這裡最好先停頓 1~10 秒,主要是 GPS 開機後需要一點時間作初始化和抓取衛星的動作
        // 這個動作需要多少時間得看附近的地形是否容易抓到夠多的衛星而定
        // 若數量不夠，GPS 會一直沒辦法輸出正確訊息
        delay(2000);
      }

    void loop()
    {
      while (Serial2.available()> 0)
      {
        if(Start_Flag){   //開始旗標是 true 就可以開始抓取相關欄位的資料
            data=readGPS(); // 第一，先把封包種類資訊抓出來，可能是 GPGGA、GPGSV 等等
            Serial.println(data);
            if(data.equals("GPGGA")){
              // 抓到 GGA 的封包，這裡就是參照前面談過的表格內容
              // 每隔一個逗號都是一項資訊,呼叫 readGPS 函數按順序把欄位內容讀取出來
              GGAUTCtime=readGPS();
              GGAlatitude=readGPS();
              GGAlatitude+=readGPS(); // N 或 S，直接加在緯度後面
              GGAlongitude=readGPS();
              GGAlongitude+=readGPS(); // W 或 E，直接家在經度後面
              GPStatus=readGPS(); // 定位狀態，0 無效，其他數字都是有效定位
              SatelliteNum=readGPS(); // 抓到的衛星數量
              HDOPfactor=readGPS(); // 精確度
              Height=readGPS(); // 距離海平面高度
```

```
                Start_Flag =false;

                if (GPStatus>0) //  必須是有效定位才可以設定
valid=true，以利後面判斷
                    valid=true;
                else
                    valid=false;
                data="";
            }
            else if(data.equals("GPGSA") ){   // GSA 封包跳過不處理
                Start_Flag =false;
                data="";
            }
            else if(data.equals("GPGSV")){   // GSV 封包跳過不處理
                Start_Flag =false;
                data="";
            }
            else if(data.equals("GPRMC")){
                //  抓到 RMC 的封包，這裡就是參照前面談過的表格內容
                //  每隔一個逗號都是一項資訊，呼叫 readGPS 函數按順序
把欄位內容讀取出來
                RMCUTCtime=readGPS();
                PositionValid=readGPS(); //  讀取到 A 才是有效封包
                RMClatitude=readGPS();
                RMClatitude+=readGPS();
                RMClongitude=readGPS();
                RMClongitude+=readGPS();
                Speed=readGPS(); //速度
                Direction=readGPS(); //方向
                Date=readGPS(); //  日期
                Declination=readGPS();
                valid=true;
```

```
        Start_Flag =false;
                data="";
            }
            else if(data.equals("GPVTG")){
                Start_Flag =false;
                data="";
            }
            else{
                Start_Flag =false;
                data="";
            }
          }

          if(valid){
            if(PositionValid=="A")
            {
                Serial.println("Get a valid position");
                output(); // 印出前面抓到的正確訊息到 LCD 和 serial
port
            }
            else
                Serial.println("Not a valid position");
            valid=false;
            PositionValid="";// 重新把檢查旗標清空，才能確實判斷下
次是否有抓到正確資訊
          }
          if(Serial2.find("$")){ // 若從 GPS 來的資料裡面含有$，代表開
始抓到資訊了
            Serial.println("Get GPS string...");
            Start_Flag =true; // 就可以把開始旗標設定為 true 準備開始
處理
          }
        }
    }
```

```
    //    從整串的 GPS 傳回字串裡面讀取被逗號隔開的資料
    //    範例 $GPGGA,073849.00, 2301.37777,N,
12012.94773,E,1,09,0.098,41.2,M,-2.5,M, ,*42
    String readGPS(){
      String value="";
      int temp;
    startRead:
      if (Serial2.available() > 0)
      {
        temp=Serial2.read(); //   一個字一個字讀取出來
        if((temp==',')||(temp=='*')) //需要一直讀取，直到讀到逗號或
是*號才是一段完整資訊
        {
          if(value.length()>0)   // value 裡面的長度比 0 大代表有確實
讀到資訊，直接傳回
            return value;
          else
            return "";   // 否則就傳回空字串
        }
        else if(temp=='$')   // 讀到$代表這是一段訊息的最開頭，準
備開始截取資訊
          Start_Flag =false;
        else
          value+=char(temp); // 把每一個讀到的字都加入 value 字
串
      }

      // 這個 delay()的數值需要稍大或稍小調整，1~3 都可以
      // 等待一下，避免讀取速度過快，GPS 根本沒資訊傳回
      delay(1);
      goto startRead; // 跳回 if 開頭，不斷的讀取，直到確認能讀到
完整資訊
    }
```

```
void output() //  輸出相關訊息的程式都包裝在這裡
{
  //  底下這一大段 serial.print 純粹是用來把訊息印出來看
  Serial.print("Date:");
  Serial.println(Date);
  Serial.print("UTCtime:");
  Serial.println(GGAUTCtime);
  Serial.print("Latitude:");
  Serial.println(GGAlatitude);
  Serial.print("Longitude:");
  Serial.println(GGAlongitude);
  Serial.print("GPStatus:");
  Serial.println(GPStatus);
  Serial.print("SatelliteNum:");
  Serial.println(SatelliteNum);
  Serial.print("HDOPfactor:");
  Serial.println(HDOPfactor);
  Serial.print("Height:");
  Serial.println(Height);
  Serial.print("Speed:");
  Serial.println(Speed);
  Serial.print("Direction:");
  Serial.println(Direction);
  Serial.print("Mode:");
  Serial.println(Mode);

  //  顯示相同的訊息到 LCD 上面
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("time:");
  lcd.print(GGAUTCtime);
  lcd.setCursor(0,1);
  lcd.print("Satellite:");
```

```
        lcd.print(SatelliteNum);
        delay(600); //  因為 16*2 LCD 螢幕只有兩行，需要頓一個時間以
後清除畫面，重新顯示新資訊
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("lat:");
        lcd.print(GGAlatitude);
        lcd.setCursor(0,1);
        lcd.print("lon:");
        lcd.print(GGAlongitude);

        delay(600); //  再度停頓一段時間後清畫面
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Height:");
        lcd.print(Height);
        lcd.print("m");
        lcd.setCursor(0,1);
        lcd.print("Speed:");
        lcd.print(Speed);
        lcd.print("km");
        delay(550); //  最後還是要頓一段時間，避免立刻回主迴圈，被清
掉畫面，最後兩行資訊來不及看到
    }
```