EduCake シリアル通信の使い方

1. シリアル通信の紹介

前章では各種信号制御による、モータや LED 等の制御を紹介しました。これらの機能は本体のコントロールパネルにて制御を行っていましたが、他のボードやスマートフォンと接続はどのように行っているのでしょうか？

本文では 86Duino EduCake のシリアル通信の使い方とプログラミングの書き方を紹介する。

シリアル通信とパラレル通信の仕様比較は図 1 を参照。

シリアルは 1 つの信号線にて“Time sharing”の形で 1 ビット毎にデータを順番に転送する。パラレルは複数の信号線にて“同時”に複数データを転送する。シリアルよりパラレルの通信スピードが速い、またシリアルは省スペースだが、パラレルは長距離でのデータを転送する際に信号の同期問題が発生することがある。

最近のコンピュータは昔のコンピュータに比べて高速のクロックで設計されており、高速のシリアル通信が可能となっている。また、最近の PC にはパラレルポートは設計されていない。何故なら、USB, SATA, RS232, RS422, RS485, I2C, IEEE 1394, PCI-E, Thunderbolt 等のシリアル通信を基本機能に持つインターフェイスのコネクタは、パラレルポートで使用する DB25 のコネクタよりも非常に小さいコネクタを使用している事も一つの理由です。

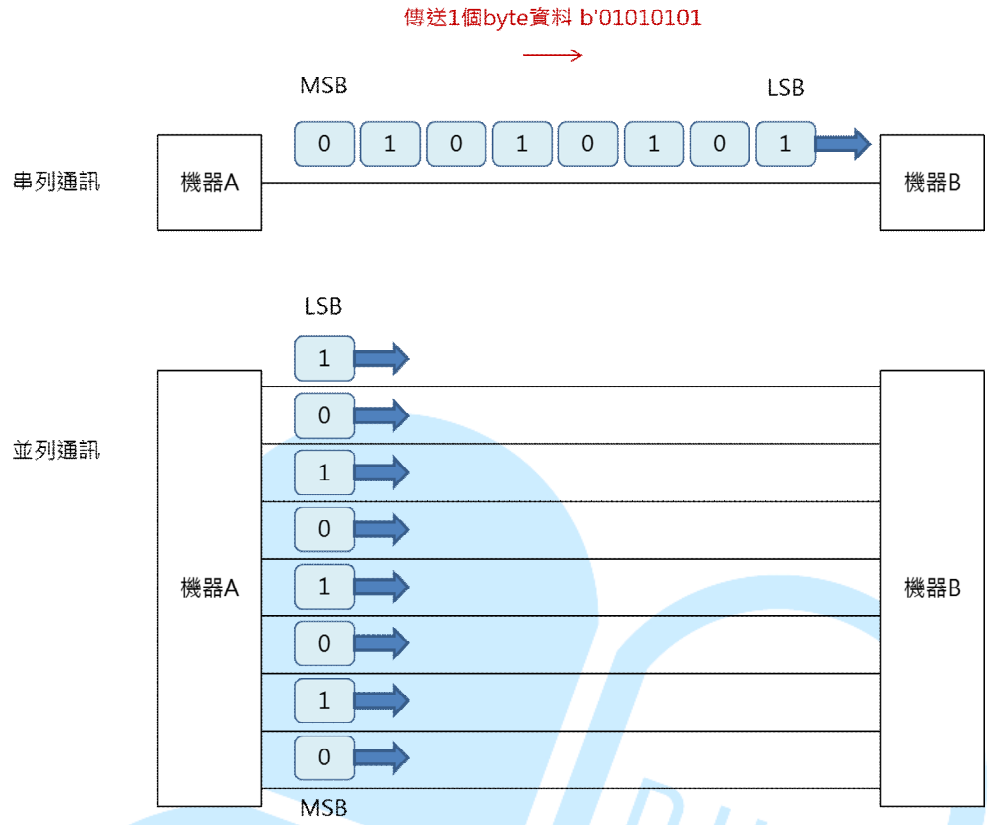


図 1. シリアル通信とパラレル通信の仕様比較

実際にはシリアルポートは一つの信号線だけでなく、パラレルポートと比べると少ないが制御信号線等を使って正しく通信を行っている。

例えば、I²C は一つのクロック(SCL)と一つのデータ線(SDA)があり、SCL により正しくデータが転送される通信方式。RS485 は一つのデータ線(D+)と一つの逆方向データ線(D-)があって正しくデータが転送される通信方式。

RS232、RS422、RS485 等のシリアル通信は異なる電気仕様の定義を持つが通称 UART（汎用非同期送受信機、ユニバーサル非同期レシーバ/トランスミッタ）インターフェイスの一つである。 86Duino EduCake の UART 通信インターフェイスは TX、RX で送受信を行う。（他の機器にて TxD / RxD と表現されている場合があるが同じ意味）なお、同時にデータが送受信できる通信方式は「全二重」と言われ(例：86Duino の UART とパソコ ンの RS232)、交互に送信または受信を行う場合は「半二重」と言われている（例：I²C と RS485）

「全二重」と「半二重」については図 2 を参照

TX/RX が使用される TTL 5V の電圧は LOW/HIGH： 0/5V だが、RS232 論理 HIGH（1）は -3～-15V、論理 LOW（0）3～15V と定義されいるので、二つの機器は UART で接続されたときに通信端子の電圧範囲に注意しており、機器 1 の TX と機器 2 の RX あるいは機器 1 の RX と機器 2 の TX との接続ができるように電圧を変換するため MAX232 等の IC を利用する。一般的な UART 機器には TX/RX にてバッファがあり、バッファを持つ機器は通信する際に CPU の負担がかなり減少しており便利な機能と言える。勿論、バッファ搭載していない機器もある。

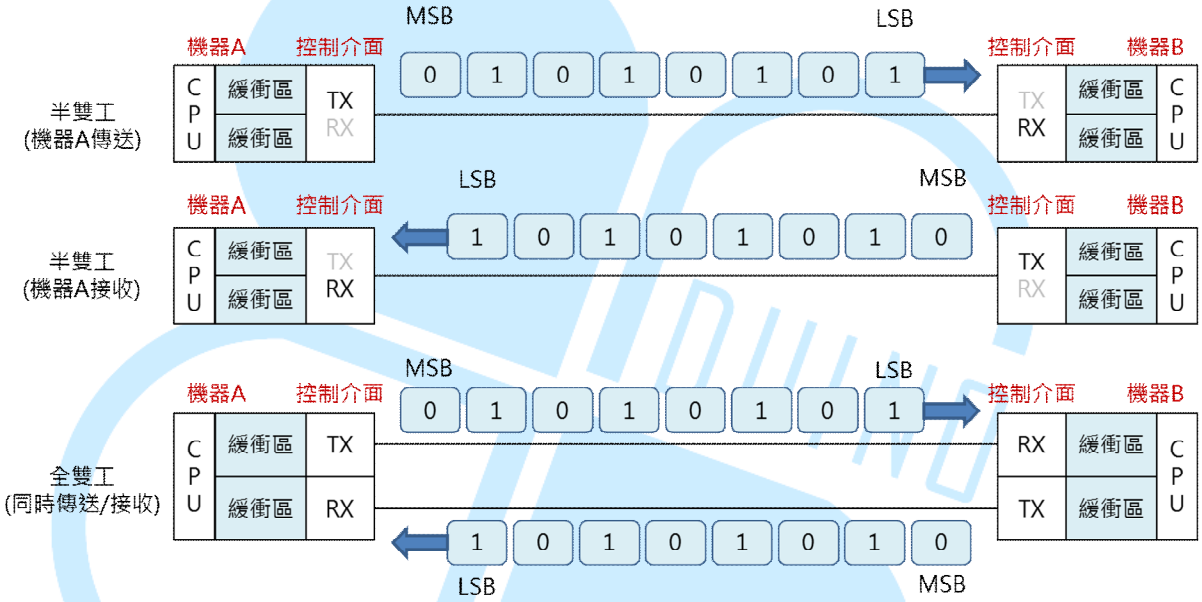


図 2. UART「全二重」と「半二重」及びバッファの説明

UART 通信には共通の特徴として、送受信の機器夫々でデータを正しく転送する為に同じ通信設定にする事がある。図 3 の通り、データ転送順として最初にスタートビットを送り、次にデータビットを送り、最後はストップビットを送る。UART の非同期 Asynchronous 通信では受信機器が受け取るスタートビットによりデータフレームの開始位置が決められる。

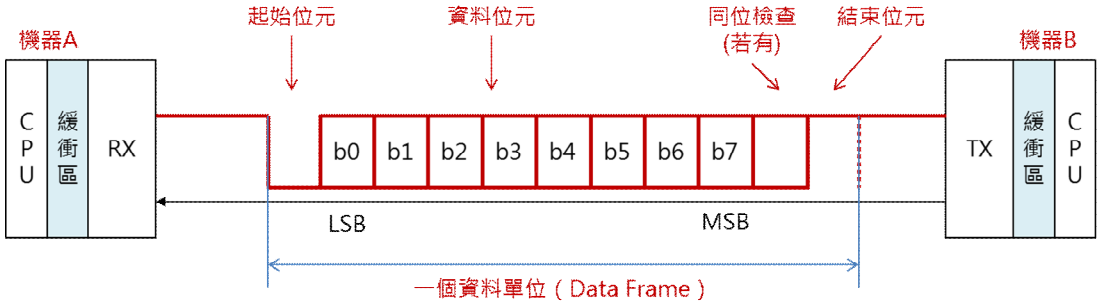


図 3. UART 通信プロトコル

一般的パソコンを例として、USB/シリアル変換アダプタを差し込んだ場合、図 4 のようなコントロールパネル画面が見えるはずだ。コントロールパネルにて TTL UART 信号/USB 信号変換チップにより、バーチャルシリアルと認識される。マウスの右ボタンをクリックして内容を押すと、そのシリアル設定画面が見える。

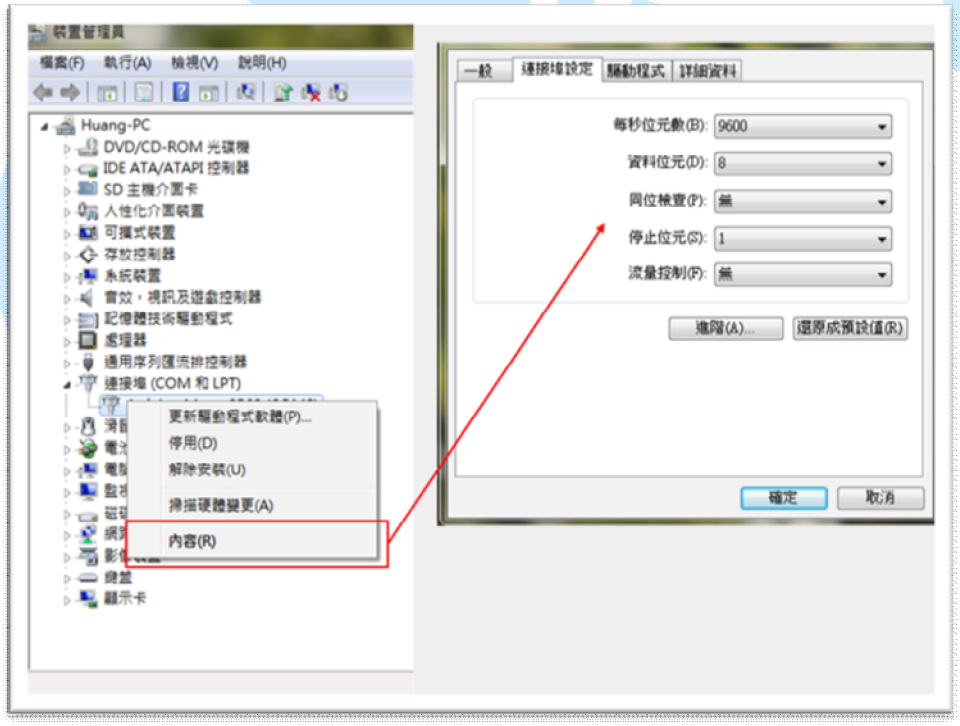


図 4. COM (シリアル)Port の設定画面

各種設定は以下の通り：

- **ボーレート Baudrate**

ボーレートというのはシリアルの通信速度、単位は bit/second（bps）。UART には同期信号がないので、両側機器にて同じ通信速度を設定して、正しくデータを転送することができる。さもなければ、通信エラーや通信できないことが発生する可能性があるので、上記のことに注意しなければならない。ある旧機器の通信速度は 4800 或いは 9600 しか対応していないもの、またある新機種では自動的にボーレートを認識できる機能を搭載している機器もある。

- **データビット(data bits)**

通常一つの転送パケットデータとして 1byte を使い、単位は 1byte(=8bits)だから、“8“を設定する。データの転送としては LSB から始まり最後が MSB となる。

- **パリティ(Parity check)**

データの伝送時に発生する、データの誤りを検出する。データを構成するビット列を一定の単位毎に区切り、それぞれに含まれる、値が「1」であるビットの個数の偶奇(奇数か偶数か)を添付する。この値をパリティビット(parity bit)という。受信側ではデータ本体から 1 の個数を調べ、パリティビットと比較することで、伝送中に誤りが生じたか否かを知ることができる。パリティビットが一致しない場合、奇数個の誤りが生じていることがわかるが、パリティビットが一致した場合、誤りが存在しないのか偶数個の誤りが存在するのかを知ることはできない。

パリティビットの値を、1 の個数が奇数個なら「1」、偶数個

なら「0」とする方式を「偶数パリティ」(even parity)、逆に、奇数個なら「0」、偶数個なら「1」とする方式を「奇数パリティ」(odd parity)という。

1. No Parity：データチェックを行わない。
2. 奇数パリティ（Odd Parity）：奇数個なら「0」、偶数個なら「1」とする方式を「奇数パリティ」(odd parity)という。
3. 偶数パリティ（Even Parity）：パリティビットの値を、1の個数が奇数個なら「1」、偶数個なら「0」とする方式を「偶数パリティ」
4. マーク（Mark）：全ての数値は 1 を設定する。
5. スペース（Space）：全ての数値は 0 を設定する

- **ストップビット(stop bits)**

  シリアルの非同期通信(UART)で、一つのデータ列の最後に付けられるビット。ここでも同期することができる。

- **フロー制御(flow control)**

  データ通信において、受信側の処理が追いつかずにデータを取りこぼしたりするのを防ぐため、通信状況に応じて送信停止や速度制限などの調整を行う機能のこと。コンピュータ間あるいはコンピュータと周辺機器の間でデータのやり取りを行う場合、相手から受信したデータはバッファメモリにいったん記録され、その後プロセッサから読み出されて処理される。このとき、送信が速過ぎてバッファメモリがあふれそうになったり、受信側が何らかの処理に忙しくてデータの処理を進められない場合などに、送信側の機器にこれを通知して、送信を一時中断したり、速度を低下させたりする。こうしたデータの流れの調整をフロー制御という。

86Duino EduCake を Code100 IDE と接続して使う初期設定“シリアルライブラリ”として(データビット：8、パリティチェック：なし、ストップビット：1、フローコントロール：なし、ボーレート：115200bps)以下の演習を説明していく。

2. 演習 1　Serial.write()

データは 2 つのシリアルポート間で、データの流れとしては電気的な High/Low"1"と "0"により転送される。ソフトアプリケーションでは、2 つのシリアルポート間のデータ転送は、バイナリデータ、バイナリにコード化された文字、10 進、16 進、その他のフォーマットで行われる。一方、文字データは、ASCII 文字列として符号化される。例えばデータパケットとして b'01000001 が転送された場合、これは 10 進数の 65 が符号化されたものとなる。一方、ASCII 文字の'A'を符号化した場合も同じ値となる。

この演習では、EduCake と開発用 PC の間で 1 バイトのデータをバイナリで転送する。 86DuinoCoding100 IDE を起動し、以下のコードを入力。

```
void setup()
{
  Serial.begin(115200);// シリアルポートボーレート設定
}

void loop()
{
  if(Serial.available())
  {
    int value = Serial.read();// バイナリデータ受信
    delay(100);
    Serial.write(value + 1);// バイナリデータ送信
  }// end if(Serial.available()
}
```

サンプルプログラム機能説明：

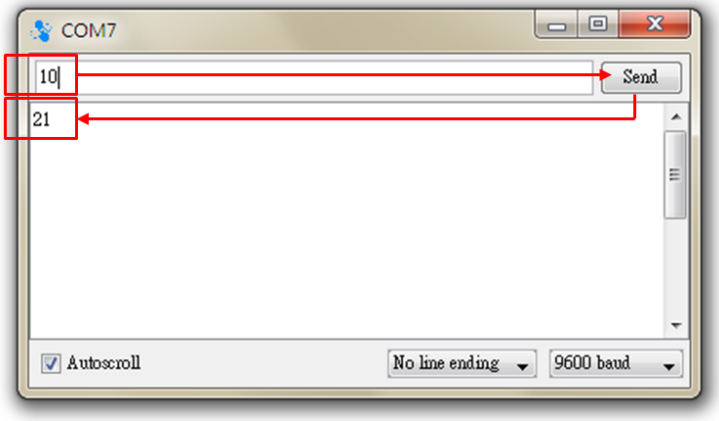
- 開発用 PC からシリアルポート経由で EduCake に 1 バイトのデータを送信する。EduCake はデータ受信後、受信したデータに 1 を加えて開発用 PC に返送する。

上記アプリケーション実行により、setup() 関数内では、Serial.begin(115200)関数にてシリアルポートのボーレート設定がコールされます。loop()関数の中では、Serial.available()関数がコールされ、受信バッファに受信データがあるかを確認しバッファ内のデータ数を返します。

Serial.Read()関数は受信データを読み込むために呼び出されます。データは値としてデコードされているので、int または unsigned int として扱うことができます。受信データを読み取り、その値に 1 を加えて Serial.write()関数を使って、100 ミリ秒の遅延後に PC に送信します。100 ミリ秒の遅延はデータの流れを示すために加えています。（あなたが違いを確認するために遅延を削除することができます）

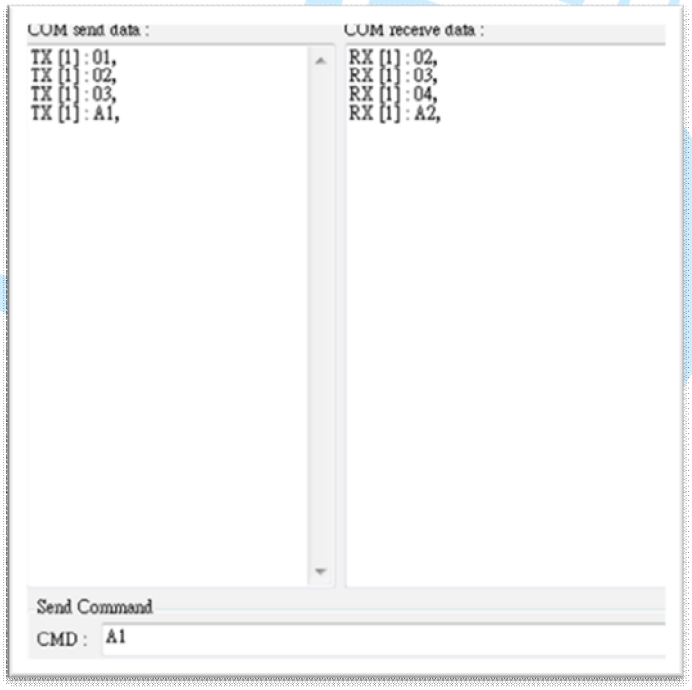
Serial.write()と Serial.read()関数で、一度に 1 バイトのデータを受信します。送信値は 1 バイトのデータ(0～255)で表せます。

シリアルモニタはデフォルトで ASCII 文字でデータを送受信するのでこの演習では使用しません。シリアルモニタを使用すると、数値の 10 は 2 つの ASCII 文字、 '1 'と'0'として送信されます。 ’1’と'0'の ASCII 文字としては 49 と 48 になりますので、結果として上記のサンプルアプリでは、数値データ(10+1)すなわち 11 の代わりに数値表現の 49 +1 = 50 と 48+1 = 49 、シリアルモニタとしては次のように’2’と’1’が表示される。



以下は、サンプルプログラムを実行した際の画面表示結果となる。

- データは 16 進数に変換された値で転送される。1 の値が送られれば返送として 2 を受け取る。2 を送れば 3 を返送で受け取り、A1 を送れば A2 を受け取る。



3. 演習 2 Serial.print( )

この項では文字の転送を練習します。以下のコードを IDE に入力して下さい。

```
void setup()
{
  Serial.begin(115200);// シリアルのボーレート設定
}

void loop()
{
if(Serial.available())
  {
    byte value = Serial.read();// 受信データの読み取り
    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);// 受信文字の 10 進数を返送
    }
  }// end if(Serial.available()
}
```

上記のコードを EduCake にダウンロードするとシリアルモニタが EduCake と通信を始め、次の様に機能する。

- シリアルモニタから文字'A'が送られると、EduCake は 10 進数で以下を返送する('32','65','66','67','68','69','10') シリアルモニタは EduCake から ASCII 文字列として'space','A','B','C','D','E' そして'LineFeed'を受け取り表示する。

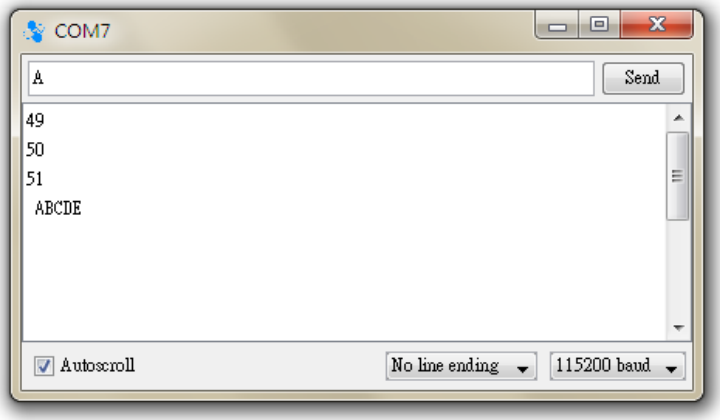
0〜31 までの ASCII コードは関連する表示または印刷可能な文字を持たない制御文字である。上記コードの様に Serial.write(10)を実行するとシリアルモニタには制御コードとして改行を意味する'LineFeed'が送られる。(他のプログラミング言語では'¥n'の組み合わせで表現される事もある)

32~126 までの ASCII コード 32 は、画面に表示可能な'0'〜'9','a'〜'z','A'〜'Z'等の文字である。

シリアルモニタが'A'の文字以外のデータを送信すると、EduCake は Serial.println()関数を使い受け取った ASCII コードの 10 進数を返します。Serial.println()とよく似た関数として Serial.print()があるが、こちらはデータの最後に'line feed'を付加せずに 10 進数でデータを送る。Serial.pringln()関数は、デバッグメッセージを送信するために、前の章で使用されている。

文字'1'がシリアルモニタから送られると、EduCake は"byte value=Serial.read()"関数にて文字'1'の ASCII コードを意味する 49 を受信する。EduCake が Serial.println(49)関数を使って 49 の値を転送すると、シリアルモニタは 4 と 9 の ASCII コードを意味する 2 つのデータ'52'と'57'を受信し'49'のデータ受信と解釈する。このアプリケーションは、表示可能な ASCII コード文字を観察し、それに対応する十進値を表示するために使用することができる。

‘1’、’2’'、’3’の文字が個別に EduCake に送られた場合、以下のシリアルモニタ画面に表示されるように EduCake は文字‘1’、’2’'、’3’の ASCII コードとして’49’、’50’、’51’を返送する。そして’A’の文字が送られると 'ABCDE’に'line feed’を付加して返送する。



上記サンプルプログラムでは Serial.write()と Serial.print()関数の違いを確認する事が出来ます。Serial.write()は数値として受信したデータをバイト情報としてデータ転送します。Serial.print()は ASCII 文字として受信したデータを ASCII 文字としてデータ転送します。

Serial.print()の拡張パラメータとして Serial.print(value,DEC)は 10 進数として、Serial.print(value,BIN)はバイナリとして、Serial.print(value,OCT)は 8 進数として、Serial.print(value,HEX)は 16 進数としてそれぞれの値を送ります。Serial.println()は Serial.print()と似ていますが、夫々のデータに’Line feed’を加えて送ります。開発用 PC を経由 EduCake に 1 バイトのデータを送信します。

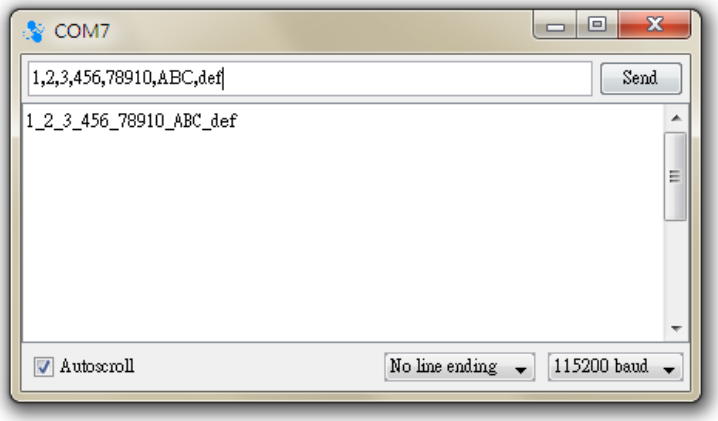
4. 演習 3　複数バイトの送受信(数値)

前項では一度に 1 バイトのデータをシリアルポートから転送する方法を説明しました。ここでは複数バイトの送受信を試します。IDE から以下のコードを入力して下さい。

```
void setup()
{
  Serial.begin(115200);// ボーレート設定
}

void loop()
{
  int byteCount = Serial.available();// バッファ内のバイト数取得
  if(byteCount > 0)
  {
    byte RX_buff[byteCount];// バッファデータ用配列宣言
    for(int i = 0; i < byteCount; i++)
    {
      RX_buff[i] = Serial.read();// バッファから読み取り
    }

    for(int i = 0; i < byteCount; i++)
    {
      if(RX_buff[i] != 44)//
      {
        Serial.write(RX_buff[i]);
      }
      else
      {
        Serial.write(95);// '_'
      }
    }
    Serial.println();// Linefeed

  }// end if(byteCount > 0)

}
```

上記コードを EduCake にダウンロードするとシリアルモニタと接続され、以下の様な動作が行われる。

- 連続した文字(‘,’を含む文字)がシリアルモニタから送られると、EuCake は以下に示すように','を’_’に代えて返送する。



loop()内の”int byteCount=Serial.available()”により現在の受信バッファのデータ長(バイト数)を読み取ります。次に RX_buff という受信データ用のバッファ配列を byteCount のバイト数分だけ定義します。Loop 内で受信バッファデータが Serial.read()関数にて RX_buffer 配列にコピーされます。loop 内の Serial.readBytes(RX_buff,byeCoun)関数により簡単に実行されます。

loop 内のコードはシリアルポートの FIFO(first in first out:先入先出)バッファデモとなります。いつも Serial.read()はの実行によりバッファから 1 バイトを読み出し、結果として読みだされたバイトを削除してバッファをシフトします。

2 番目の loop では RX_buff 配列のデータが 44 でなければ単純に Seiral.write(RX_buff[i])を使って返送する動作を繰り返します。もし RX_buff 配列のデータが 44、即ち ASCII コードで','の場合、Serial.write(95)関数により','の代わりに’_’を返送します。

注：loop()関数内のコードはとても速く処理されます。10 バイトのデータがシリアルモニタからゆっくりと適当な間隔無で転送される場合、

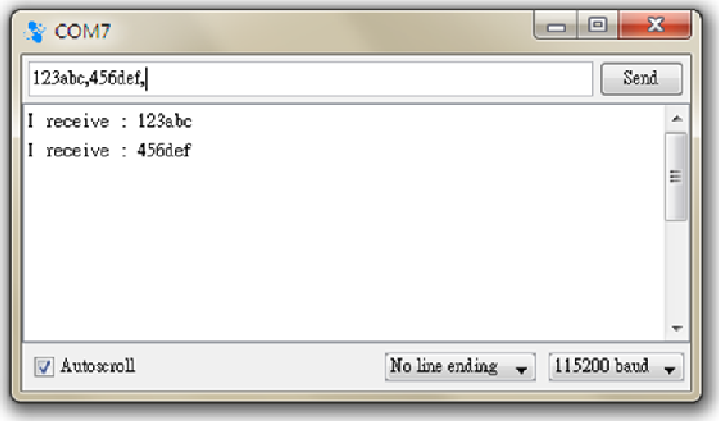
byteCount=Serial.avalable()は 10 バイト全てのデータを直ちに取り込めず 10 バイトのデータを受信するマルチループが必要となるかもしれません。しかしながら、今回の演習では心配はありません。

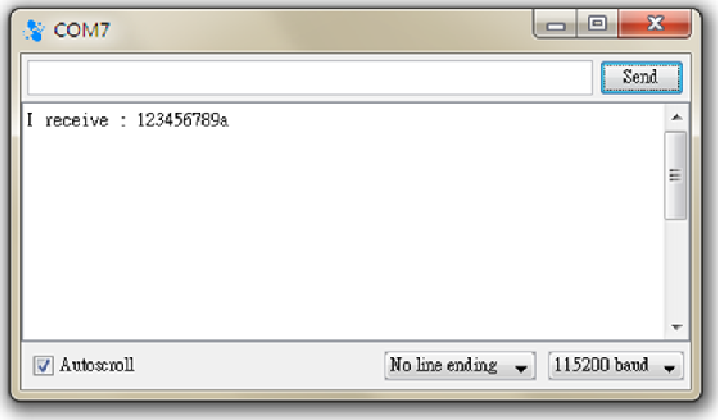
5. 演習 4　複数バイトの送受信(文字データ)

EduCake のシリアル通信には文字列を使って通信する機能も提供しています。86Duino IDE に以下のコードを入力します。

```
void setup()
{
  Serial.begin(115200);// ボーレート設定
  Serial.setTimeout(2000);// Serial.readBytesUntil()関数の
                          //継続時間設定：単位 m 秒
}

void loop()
{
  char RX_buff[10];// 文字列配列定義
  int byteCount = Serial.readBytesUntil(',', RX_buff, 10);// 受信
文字に','が検出されるか 10 文字まで続ける
  //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();// Line feed
  }// end if(byteCount > 0)

  delay(100);//
  Serial.flush();// バッファクリア
}
```

そして EduCake は','を受け取った際に受信データと共に”I receive: “表示を加えてシリアルモニタに以下の様に返します。”I receive: xxxx” また、10 文字の文字列の中に’,’が含まれていなかった場合は、その時点で、”I receive: “表示を加えてシリアルモニタに以下の様に返します。





上記のコードでは“Serial.readByteUntil (‘,’, RX_buff, 10)” 関数が読み取りと RX_buff 配列へのコピーを行います。データの RX_buff へのコピーは','文字が検出されるまでか 10 文字がコピーされるまで続きます。適当なタイムアウト時間が設定されていないと','文字が含まれない場合や、10 文字に満たない場合はアプリケーションはこの命令で止まったようになります。“Serial.setTimeout (2000)” 関数の()内の値は最大の待ち時間をミリ秒で表し、この場合は 2 秒となり Serial.readBytesUntil ()関数の追加データ待ち時間となります。

RX_buff 文字配列は loop ()関数の最初で宣言され、配列の長さは readBytesUntil 関数の “int byteCount = Serial.readBytesUntil(‘,’, RX_buff, 10)”値と等しいか大きく無ければいけない。

readBytesUntil 関数にて受信バッファから複数の文字を読み取ります。byteCount の値が“if (byteCount >0)”構文によって 0 より大きければシリアルモニタに転送されます。

Serial.print(“I receive : “)関数にて ASCII キャラクタの転送を行います。ASCII キャラクタ文字列はダブルクオテーション(“)にて囲まれた文字列をまとめてデータとして送ります。Serial.print()関数を使って"i receive :" の文字列を送った別の方法として以下の様に個別に文字を送る事も出来ます。

```
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(' ');
```

上記のコードも同じ結果となりますが、有効なコードの記述とは言えません。

本操作に於いてデータの取りこぼしを回避する為、最後の行にある Serial.flush()という関数で送受信両方のバッファをクリアします。本関数を呼び出す時、delay(100)という関数を使って遅延を発生させ Serial.wirte()と Serial.print()関数の早めの処理が競合する事を確認出来ます。

遅延無での Serial.flush()関数の実行では不完全なデータ転送となるでしょう。

6. 演習 5

この章では、以前に紹介した章の中から機能を組み合わせてもっと面白いシリアル通信を使った EduCake の応用を試してみましょう。86Duino IDE に以下のコードを入力します。

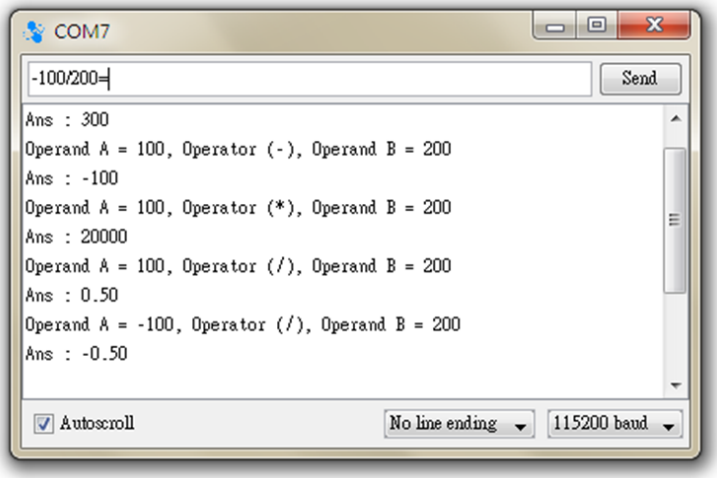
```
int Op_A = 0;// 変数定義
int Op_B = 0;//

void setup()
{
  Serial.begin(115200);// ボーレート設定
  Serial.println("Please enter format : Operand A +(or -) Operand B =");// 入力指示メッセージ
  Serial.setTimeout(1000);//Serial.parseInt()用タイムアウト設定
}

void loop()
{
  // バッファデータから整数値読み取り A
  Op_A = Serial.parseInt();
  Serial.print("Operand A = "); Serial.print(Op_A);

  // バッファデータから検索
  char opr = Serial.read();
  Serial.print(", Operator ("); Serial.write(opr); Serial.print(")");

  // バッファデータから整数値読み取り 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() == '=')
    }
```

上記のコードを EduCake にダウンロードすると、シリアルモニタとの通信が始まります。Serial monitor からは、以下の様なフォーマットで転送データを送れます。(データ間のブランク用 space 無で)

[最初の整数値][数値演算][次の整数値]

EduCake は必要な演算を行い結果を以下の様に返します。



上記アプリケーションでは数値演算で使用される、OP_A と OP_B　2 つの整数宣言から始まっています。setup()関数は演習 4 の関数に似ています。

loop()内ではまず Serial.parseInt()関数にてバッファからデータを読み取り最初の非演算子の整数値 Op_A とします。この Serial.parseInt()はバッファからデータを読み取り最初の数字即ち数字ではない文字までの値を整数値として取り込みます。例えば‘1’, ‘2’, ‘3’, ‘+’, ‘1’ (“123+1”) この様なデータを読み取った場合、’+’までの最初の 3 つの数字が一つの数値 123 として解析されます。もしデータが‘-‘文字から始まるような‘-‘, ‘1’, ‘2’, ‘3’, ‘+’, ‘1’ (“-123+1”)の場合、負の値である -123 と解析されます。そして Serial.read()にて次の文字を読み取り数式の演算子とします。Serial.read()の後 Serial.parseInt()にて残りのデータを 2 番目の非演算子 Op_B として解析します。最後に “ if(Serial.read() == ‘=’ “ 構文にてバッファから読み取った次の文字が‘=’ (等号)かをチェックします。 もし次の文字が‘=’、即ち演算指令なら switch-case 構文に従って 4 種の数値演算(加算、減算、乗算、除算)の一つが実行されます。. それぞれ 4 種の演算が要求された演算子で Op_A と Op_B の値を使って実行されます。結果からすると減算と乗算は指示された Each of the

4 math function will be executed with Op_A and Op_B as the required operands.

Since result from the addition, subtraction and multiplication math function based on

two integer value is also an integer, special handling is not needed. Since the division

math function between two integer operands may create a floating point number (such

as dividing 3 by 2 will yield 1.5), both operands (Op_A and Op_B) are converted to

floating point number for the division math function. Otherwise, the fractional value for

the result from a division math function will be truncated and lost.

Instead of using integer based operands, you can declare Op_A and Op_B as floating

point numbers and use the Serial.parseFloat () function in place of the Serial.parseInt ()

function. For more information about the functions used in this exercise, visit the following

URL:'、 '文字が含まれる文字の文字列は、から送信されると

シリアルモニタ、 EduCake はデータの文字列を受け取り、 '、'文字の後に停止検出される。

その後、 EduCake は受信データの文字列を返す「私は受け取ります」に追加

このような" ： XXXX は受信」などのバックシリアルモニタに文字列データを、受け取った。

データの文字列を'、 '文字なしで送信された場合、 EduCake リターン

で追加された： 「XX は受信」で戻ってシリアルモニタへの文字データの文字列、以下に示すように始まる。

上記のコードでは、 「 Serial.readByteUntil （ '、' 、 RX_buff 、 10 ） "関数を読み取るために使用される

と RX_buff 配列にデータをコピーします。データが受信バッファにコピーされるから'、 '文字までの RX_buff アレイが検出されるか、 10 文字がコピーされます。受信したメッセージが含まれていないとき」を、 ' 、適切な時間を設定する

文字やメッセージの長さは、アプリケーションのコード行で停止し、 10 文字未満です。 「 Serial.setTimeout （2000 ） 「セットアップ内部機能、 （）関数は、 theSerial.readBytesUntil （）関数が追加データを待機するために、 2 秒である、 （ミリ秒）の時間の最大長を設定するために使用される。

RX_buff の文字配列をこの配列の長さは readBytesUntil 機能、 「INT に、byteCount = Serial.readBytesUntil （ ' 、 ' 、 RX_buff の中の文字の長さと等しいまたはパラメータより大きくしなければならないループ（ ）関数の先頭で宣言されています、 10） " 。

readBytesUntil 機能は、受信バッファからの複数バイトのデータを読み込みます。中に、byteCount の値は、 "（に、byteCount > 0 ）場合には、「ゼロ· データよりも大きいときはシリアルモニタに再送信する。

Serial.print は（ 「XX が受け取る： " ）関数は、 ASCII 文字列を送信することができます。送信される文字列が二重引用符（ "）で囲む必要があります。

二重引用符（ " ）文字は、受信ノードに送信されるデータ· ストリームの一部ではありません。代わりに、「私が受け取る： "送信するために Serial.print （ ）関数を使用する文字列を、以下に示すように、あなたは、別の関数呼び出しで、個々の文字を送信することができます。

上記のコードは同じ結果を提供しますが、それはコーディングする効率的な方法ではありません。

現在の操作からデータ余っを回避するために、コードの最後の行は、呼び出し送信および受信バッファの両方に、バッファをクリアする Serial.flush （ ）関数は、この関数によってクリアされます。 Serial.flush （ ）関数を呼び出すときに、以前のを確実にするために（ 100 ）関数の遅延を呼び出すことで機能を遅延させることがよいでしょう

Serial.write （）と Serial.print （ ）関数は、自分のタスクを競った。あなたが呼び出してみてください

不完全なデータ転送が発生します影響を参照するには、遅滞なく Serial.flush （ ）関数。

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

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

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

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

一、　第六個程式 GPS 的實作

この最後の演習では、簡単な GPS アプリケーションを介して動作します。 GPS（全地球測位システム）を最初、米国国防総省によって作成されました。

GPS は、広くデバイスの位置を追跡し、走行速度、ナビゲーションに使用されている地球の表面の 98%を覆う asatellite ナビゲーションシステムである。また、 GPS システムはまた、精度の時刻サービスを提供する。私たちは、タイマー機能を提供し、 EduCake の正しい時刻を設定するために GPS を使用することができます。それは深く、GPS に掘るためにこのチュートリアルの意図の中にはありません。あなたは、GPS の詳細については、興味のある方は、インターネットを検索したり、「GPS」と「全地球測位システム」キーワードを使用してウィキペディアを照会。

行使により操作するには、 GPS モジュールが必要になります。かなりの数があります。

RS- 232 （ UART）で構築され、市場で利用可能な GPS モジュール、様々な、私は²C または SPI

インターフェイス。 TTL の UART インタフェースを備えた NEO- 6M の GPS モジュールは、のために使用される

図 1 に示すように、このセクションの運動。

図 1. TTL 信号輸出の GPS モジュール

NEO-6M のスペック：

1.NC (No connection)

2.VCC (3.6 to 5.2V power source, which can be attached to EduCake's 5V output)

3.RX (Receive)

4.TX (Transmit)

5.Ground

6.PPS

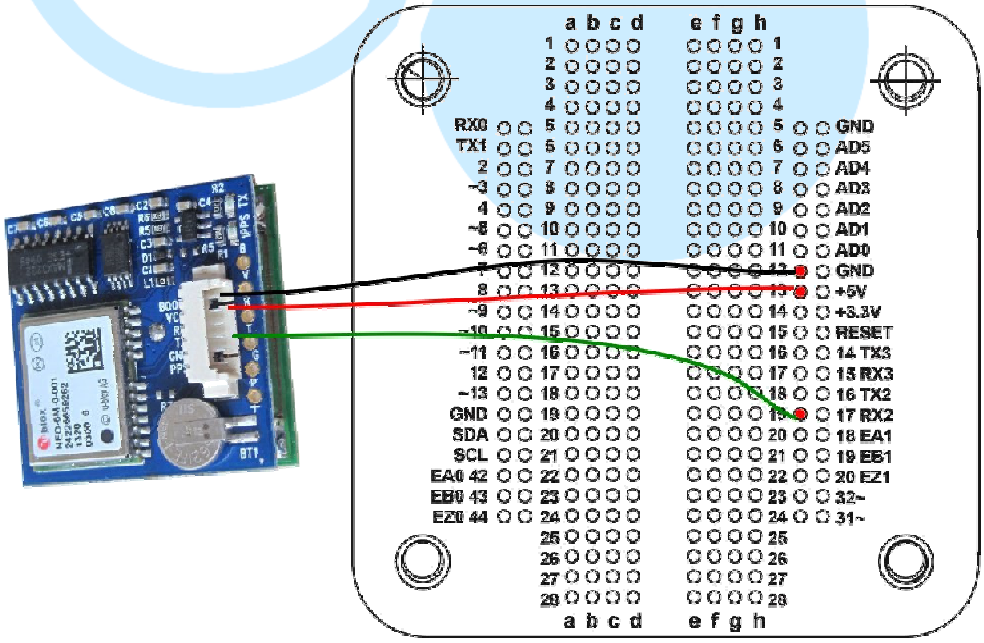


図 2. GPS モジュールの接続方

GPS モジュールについて知る必要があることがいくつかあります。
（コールドスタート）に 1 電源： 、 GPS モジュールがあるとの電源をオンに必要がある場合
初期化され、利用可能な衛星を検索し、計算するために衛星信号を使用して 20〜30 秒と長くかかる場合があります現在の情報、 。ほとんどの GPS ナビゲーションデバイスは、電源オン時に再生成された地図データとルーティングする必要が初期化を完了するためにはるかに長い 30 秒以上かかりますいる電源オン時の処理。ほとんどの GPS ナビゲーションデバイスをキャプチャする必要があります。
機能とするために、3 つ以上の衛星からの信号は、数分かかることがあります初期化プロセスを完了します。
2 ウォームリセット：ウォームリセット（スタンバイからウェイクアップ） 、GPS ナビゲーション
デバイスが準備ができて、わずか数秒かかる場合があります。
3 再捕獲は、衛星信号を失った：広域市のご旅行の際はどこに
高層ビルは、駆動、地下駐車場の構造では、密接に位置しています
橋の下やトンネルを介して、 GPS デバイスは、信号を受信することができない。衛星からの再捕捉衛星信号に有している。再捕獲衛星信号は、 GPS 装置に格納する、データと新しいデータを比較する
デバイスのバッファと必要な調整や修正を行う。
一般的には、それが再捕獲された衛星信号を失うために、ほんの数秒がかかる。

EduCake 上ネオ 6M GPS モジュールを使用するには、モジュールの 5V、GND と TX を接続する

図-2 に示すように EduCake への信号、。 EduCake 上の接続は明らかにされ

マークされた。このような移動を計算するようなタイミングを伴い、正確なデータを取得する スピードは、1 秒タイミング信号を提供ネオ 6M モジュールから PPS ピンを使用することができます

EduCake に GPS モジュールを接続した後に以下のコードを入力します。

86Duino IDE：

上記のコードは、第二シリアルポートから GPS モジュールからのデータをキャプチャし、 シリアルモニタに、受信したデータは、第一シリアルポートを介して、再送信した

次のような GPS モジュールからの生データを参照することを可能に：

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

我々から取得したデータストリームから関連データを解析するためのコードを記述する必要があります

GPS モジュール。 GPS モジュールからのデータストリームは、典型的には、特定の NMEA に付着する

文と呼ぶ形式、。別の会社からの GPS モジュールを採用することができる

異なるデータフォーマット。上記の NMEA センテンス内のデータの個々の作品です 。

次の表に示すように、カンマ（"、"）で区切られた：

$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

ほとんどの GPS が必要とする情報を含む前記データ· ストリームに加えて、

アプリケーションは、そのような GLL 及び RMC メッセージなどの他の GSP データ形式がある
形式：
- $ GPGLL 、 2301.37777 、 N 、 12012.94773 、 E、 073849.00 、 A、A * 62　GLL メッセージフォーマットは、 GGA より短く、緯度、経度、 UTC 時間情報を含む。
- $ GPRMC 、 073141.00 、 A 、 2301.37777 、 N 、 12012.94773 、 E、 0.016,0,190214 、 、 、 * 77 A
RMC メッセージフォーマットは、上記メッセージの " A "である第 3 変数は、データは""アクティブ表し、 "V "はボイド表す場合、有用であり、データを示しているか否かを示すために使用される GGA 、幾分類似している衛星信号の欠如によって引き起こされることができる、使用可能ではない。経度データ後の値、 「 0.016 」ノット（ 1 ノット= 1.15078 マイル/時または 1.852 キロ/時）での速度測定を旅表す。次の変数は、 "0"度の角度を追跡する。次の変数は、「 190214 」は、 2014 年 2 月 19 日の日付を表す。上記の中で空白になって次の変数、
データメッセージは、磁気変動と方向を表す。最後の変数は、 「A * 77 」であり、
サム。
このスコープ内ではない GPS メッセージフォーマットの多くの異なるバリエーションがあり、
チュートリアル。あなたがより多くを学ぶことに興味を持っている場合は、「 NMEA 」を使用してインターネットを検索し、
「 NMEA メッセージタイプ」のキーワード。 NMEA は国立海洋電子機器の協会。
データを視覚化するために、我々はデータを表示する 16× LCD モジュールを使用することができます。

```
#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
    }
```