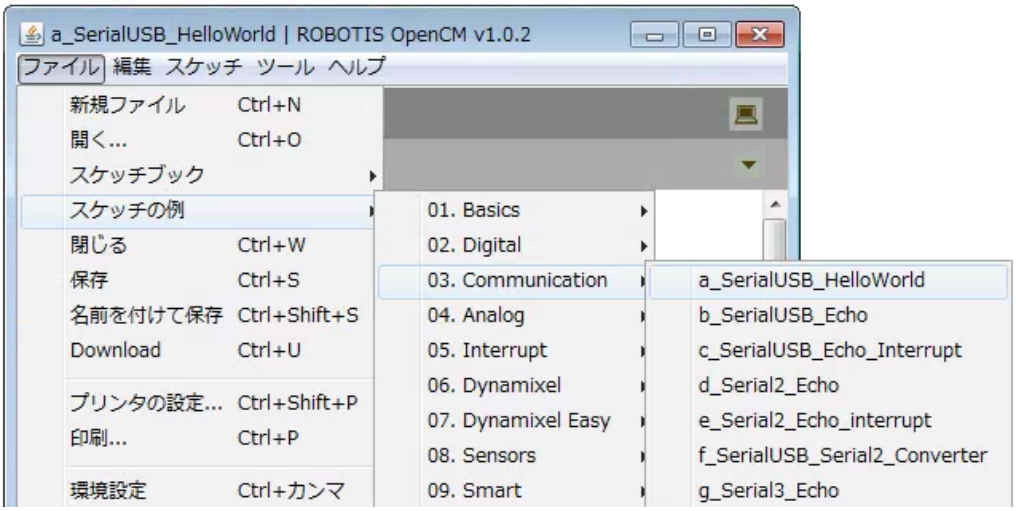
5　OpenCM で「Hello World!!」してみよう

Hello World とは，ディスプレイ上に文字列を出力する非常に簡単なプログラムのことである．プログラミングの入門書では，初めてのプログラミングの例として，この「Hello World」という文字列を出力するプログラムが紹介される．OpenCM でも，この Hello World のプログラムサンプルが用意されている．以下の図のように，[ファイル]-[スケッチの例]-[03.Communication]-[a_SerialUSB_HelloWorld]を選択して開いてみよう．



以下は，a_SerialUSB_HelloWorld のソースコードである．まずソースコードを理解しよう．

```
/* SerialUSB_HelloWorld

 USB Serial print "Hello World!!" to PC
 You can see it any terminal program, putty, TeraTerm, Hyper Terminal, etc...

                    Compatibility
 CM900                   O
 OpenCM9.04              O


 created 16 Nov 2012
 by ROBOTIS CO,.LTD.
*/
volatile int nCount=0;

void setup() {
}

void loop() {
  //print "Hello World!!" to PC though USB Virtual COM port
  SerialUSB.println("Hello World!!");
  SerialUSB.print("nCount : "); // display nCount variable and increase nCount.
  SerialUSB.println(nCount++);//SerialUSB.print("\r\n");

  delay(1000);
}
```

1～13 行目までの/* */で囲まれている部分はプログラムのコメント，すなわちプログラムではなく参考用のメモなどを記した部分である．同じく，//で始まる行も，その行の最後までがコメントとして認識される．コメントを除いたプログラムの最初の行は 14 行目の

volatile int nCount=0;

である．これはグローバル変数（大域変数）として変数 nCount を整数型として宣言していて，その初期値を 0 としている．ここまでは C や JAVA などのプログラミング経験者であれば，誰もが簡単に理解できるはずである．しかし，先頭に付いている volatile（ボラティル）を知っている人は多くないだろう．volatile は型修飾子の一つであり，volatile を付けて宣言された変数の取り扱い方法を変更する．ここでは詳しくは説明しないが，volatile を付けて宣言された変数は常に最新の値が参照される．しかし，ここでは深く考えずに，volatile 以降を見ていこう．

前述したように，OpenCM のプログラムは setup()で必要なピンの設定等を行い，loop()で実際の動作プログラムを書くという構成となっている．このプログラムでは，setup()には内容がないため，何も設定をしていないことを意味する．loop()にも簡単なコードが数行書かれているだけである．

SerialUSB.println(″Hello World!!″);

は USB ケーブルを通して PC に”Hello World!!”と改行コードを送信する命令であり，

SerialUSB.print(″nCount : ″);

は USB ケーブルを通して PC に”nCount :”を送る命令（改行コードは送らない），

SerialUSB.println(nCount++);

は変数 nCount の値を PC に送り，その後に nCount の値を一つ増加させる命令である．

delay(1000);

は 1 秒（1000 ミリ秒）間プログラムを停止する命令であり，全体としては 1 秒間隔で PC に”Hello World!!”と変数 nCount の値を送信するプログラムとなっている．

プログラムを理解できたら，コンパイル，ダウンロードして実行してみよう．これまでの LED 点滅プログラム等とは異なり，OpenCM ボード上では目に見える変化が現れない．しかし，正しくプログラムがダウンロードされていれば，OpenCM は USB ケーブルを通じて PC に上記のメッセージを送信している．これを確認するには，開発環境画面の右上にある PC の形をしたアイコンをクリックする．

このアイコンは，シリアルモニタを起動するボタンであり，クリックすると以下のようなシリアルモニタ画面が表示され，その中に OpenCM から送信されたメッセージが表示される．

6　アナログ入力をしてみよう

マイコンボード内では全ての演算はデジタルで行われる．デジタルとは，0 と 1 の二つの数字，すなわち 2 進数で全てを表現する．一方で，実際の世界はアナログである．アナログとはデジタルの反対の概念であり，連続した量を意味する．例えば，下図左のような可変抵抗について考える．可変抵抗は，レバーを左右に回転させることで抵抗値が変化する部品である．内部は下図右のようになっており，レバーを回ると②が①と③の間を接触しながら移動する．そのため，レバーを回すことによって，①･②間の抵抗値，②･③間の抵抗値は変化し，①と③の間の抵抗値は変化しない．

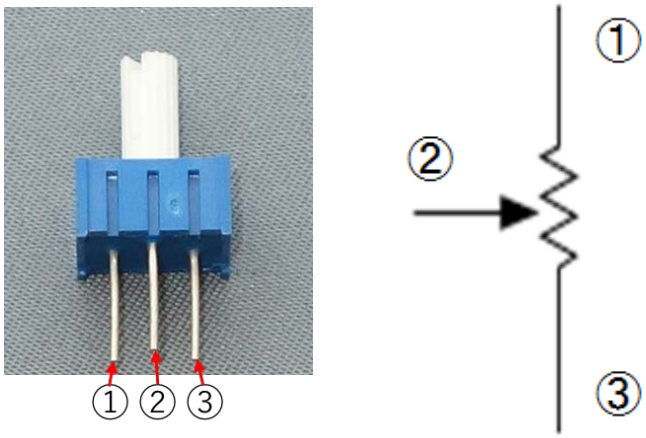


図 6.1　可変抵抗

この可変抵抗に図 6.2 のように配線を行ったとすると，②の電圧は③に繋がっている GND に対して 0V～3.3V の値になり，レバーの廻した量に応じて②の電圧が変化する．

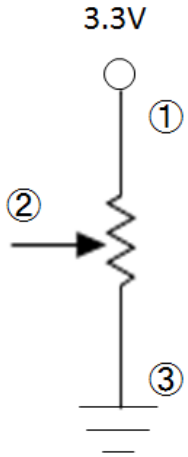


図 6.2　可変抵抗への接続

このようなアナログの値をマイコンで扱うにはデジタルデータに変換する必要がある．アナログ量（電圧）をデジタル量（2 進数）に変換する装置を AD 変換器（Analog Digital Converter，ADC）と呼ぶ．OpenCM 内には AD 変換器が内蔵されており，A0 から A9 まで合計 10 本のアナログ入力をデジタル値に変換できる入力ピンが用意されている．

ここで，例えば上図の②に着目すると，0V～3.3V までの電圧が出力されるが，その値としては 1.95V，2.599V，3.00001V など，様々な値を取りうる．しかし，AD 変換器でデジタルに変換する際には AD 変換器の性能によって分解能が決まる．例えば，8bit の AD 変換器であれば，変換後のデジタル値が 8bit になることを意味し，アナログ値全体が 256 段階で表現される．

上の説明でデジタルとは 0 と 1 で全てを表現すると言ったため，二つの値しか表現できないと勘違いする人もいるが，0 と 1 の組み合わせで表現するというのが正しい表現である．例えば，8bit の場合は 00000000b から 11111111b まで，計 256 種類を 0 と 1 の組み合わせで表現できる．ここで，数字の最後に付く b はその数字が 2 進数であることを意味している．

OpenCM の AD 変換器は 12bit の ADC であるため，0V～3.3V までの電圧を 0～4095 までの 4096 段階（2 の 12 乗）の値で表現できる．OpenCM では変換可能な電圧の上限が 3.3V となっており，3.3V より高い電圧は 4095 となる．OpenCM で AD 変換のテストを行うため，ブレッドボードに図 6.3 のように配線しよう．①は 3.3V に，③は GND に，②は A1 ピンに繋がるように配線する．

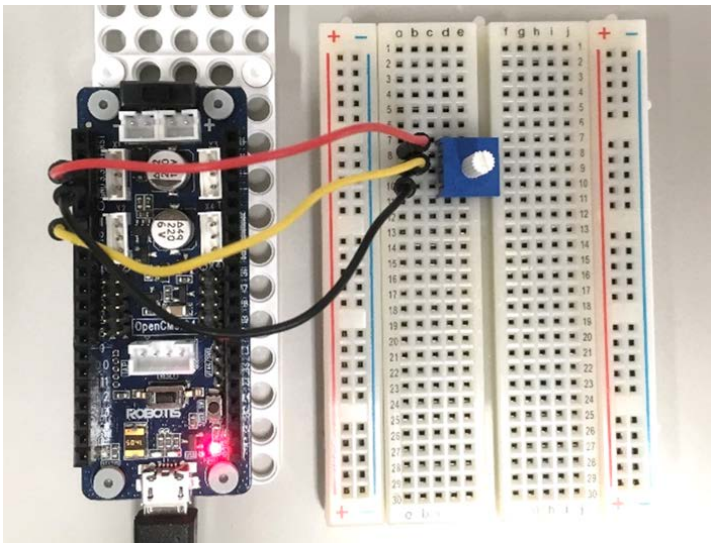
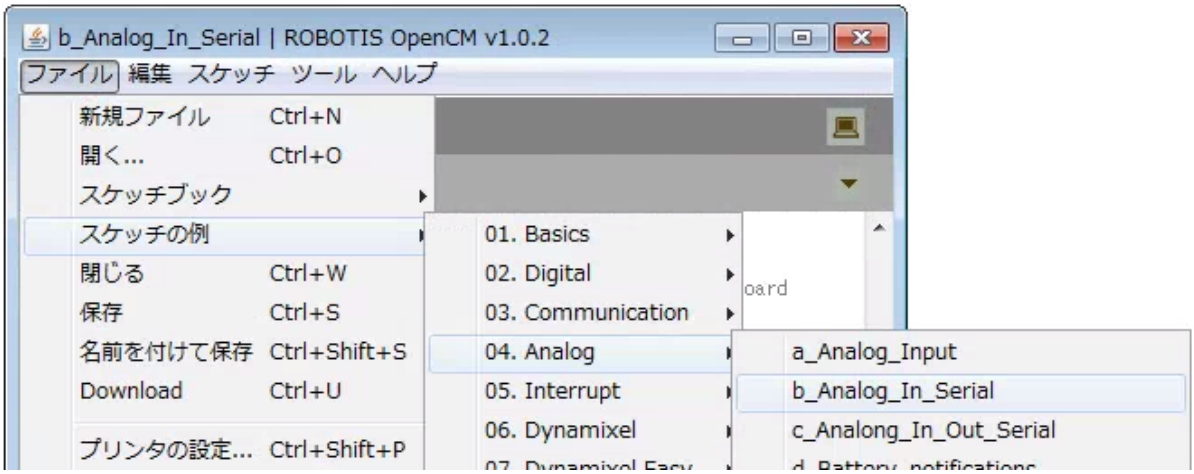


図 6.3　アナログ入力の回路

配線が終わったら，[ファイル]-[スケッチの例]-[04.Analog]から b_Analog_In_Serial を選択し，サンプルプログラムを開いてみよう．



以下は b_Analog_In_Serial のソースコードである。

```
18  // Analog input pin.  You may need to change this number if your board
19  const int analogInputPin = 1;
20
21  void setup() {
22    // Declare analogInputPin as INPUT_ANALOG:
23    pinMode(analogInputPin, INPUT_ANALOG);
24  }
25
26  void loop() {
27    // Read the analog input into a variable:
28    int analogValue = analogRead(analogInputPin);
29
30    // print the result:
31    SerialUSB.println(analogValue);
32    //need some delay because coming out too fast from USB COM port
33    delay(100);
34  }
```

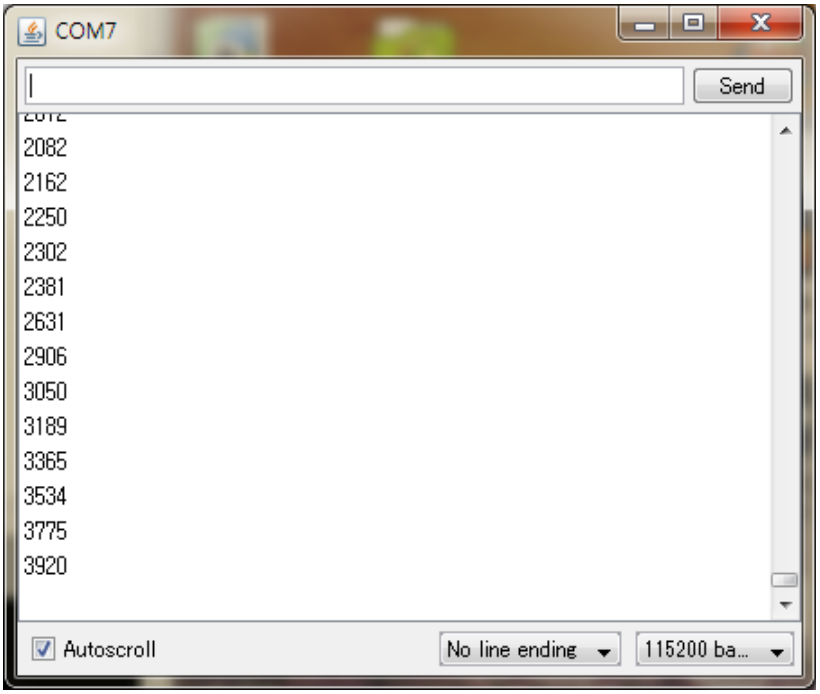
const int analogInputPin = 1;は，変数 analogInputPin を整数の 1 として定義する命令である．const は定数を意味し，const を付けて宣言された変数は定数となり，書き換えのできない読み取り専用の変数となる．

pinMode(analogInputPin, INPUT_ANALOG);は，analogInputPin すなわち 1 番ピン（A1）をアナログ入力として使うと設定している．

int analogValue = analogRead(analogInputPin);は，A1 ピンに入力されたアナログ値を AD 変換した結果の値を整数型の変数 analogValue に代入するという命令である．

SerialUSB.println(analogValue);を用いて変数 analogValue の値をPCに送っている．この動作を dekay(100);を間に入れることで約 0.1 秒間隔で行っている．

コンパイル・ダウンロード後，シリアルモニタを実行して動作を確認してみよう．正しく動作していれば，下図のようにシリアルモニタ画面に 0 から 4095 までの値が表示され，可変抵抗のレバーを廻すとそれに従って表示される値が変わるはずである．



7　7 セグメントディスプレイに数字を表示してみよう

7 セグメントディスプレイ（seven-segment display）は，LED 等を 7 個使用してアラビア数字 8 の字を表現している電子部品である．7 セグメントの 7 は，7 つに分かれた表示のことから由来しており，各々の LED を適切に点灯／消灯させることで，0 から 9 までのアラビア数字を表示できる．また，A, b, c, d, E, F などの文字も表示することが可能であり，16 進数の簡易的な表示にも使われる．ただし，実際の 7 セグメントLEDには，小数点を表示するための LED が付いている場合も多く，マイコントレーニングキットに入っているセグメントLED（図 7.1）も，8 個の LED で構成されている．

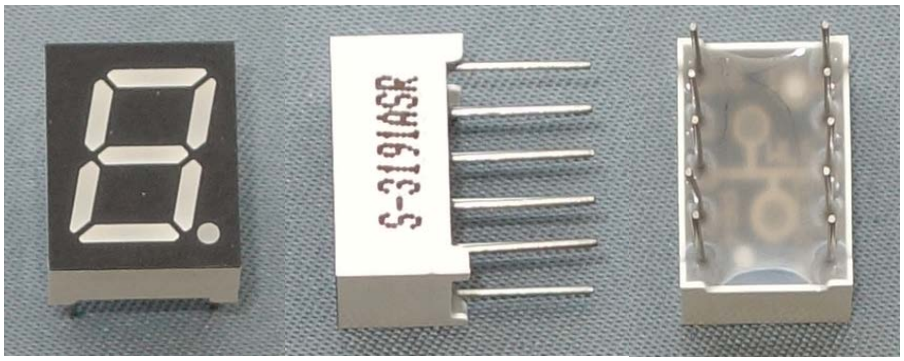


図 7.1　7 セグメント LED

図 7.2 は，マイコントレーニングキットに入っている8セグメント LED の構成である．この 7 セグメント LED はアノード・コモンとなっている．LED の2つのピンをアノードとカソードと呼ぶが，＋側のピンをアノード，－側のピンをカソードと呼ぶ．つまり，アノード・コモンとは，7 セグメント LED を構成する LED のアノードが共通になっている，という意味である．単純に 8 個の LED を接続するには16本のピン(8LED×2)が必要となるが，アノードを共通とすることで，9本のピンを接続するだけで8つの LED を点灯・制御できる．市販の7セグメント LED には，カソード（－側のピン）が共通となっているカソード・コモンも存在するため注意が必要である．

図 7.2 左は各 LED を表す記号であり，右は各 LED がどのピンに接続されているかを示す図である．1 から 10 までの数字は 7 セグメント LED の脚番号であり，7 セグメント LED 表面を見ている状態で一番左上が 1 番，その下が 2 番となり，反時計まわりで番号が増えて行く．右下は6番，右上は 10 番ピンとなる．

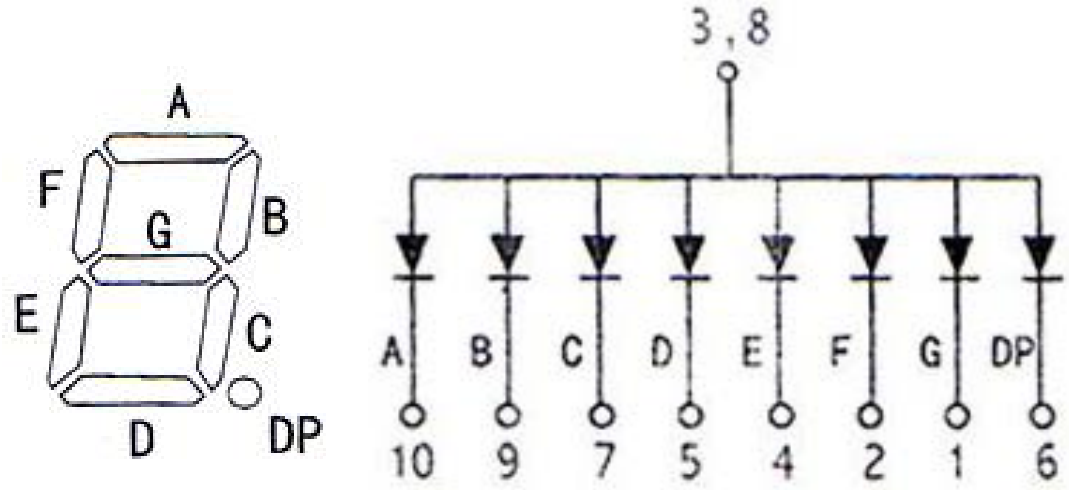


図 7.2　7 セグメント LED の内部構成

OpenCM の 8 本の GPIO ピン（A8, A9, D10, D11, D12, D13, D14, D15）と 7 セグメント LED の G, F, E, D, A, B, C, DP ピンを間に抵抗（100Ω，茶黒茶）を入れてそれぞれ繋いでみよう．7 セグメントの 3 番，8 番ピンがアノードなので，この脚は 3.3V に接続する（どちらかに接続すればよい）．図 7.3 に接続の様子を示す．

接続できたら，7セグメント LED に 0～9を表示するようにプログラムを書いてみよう．

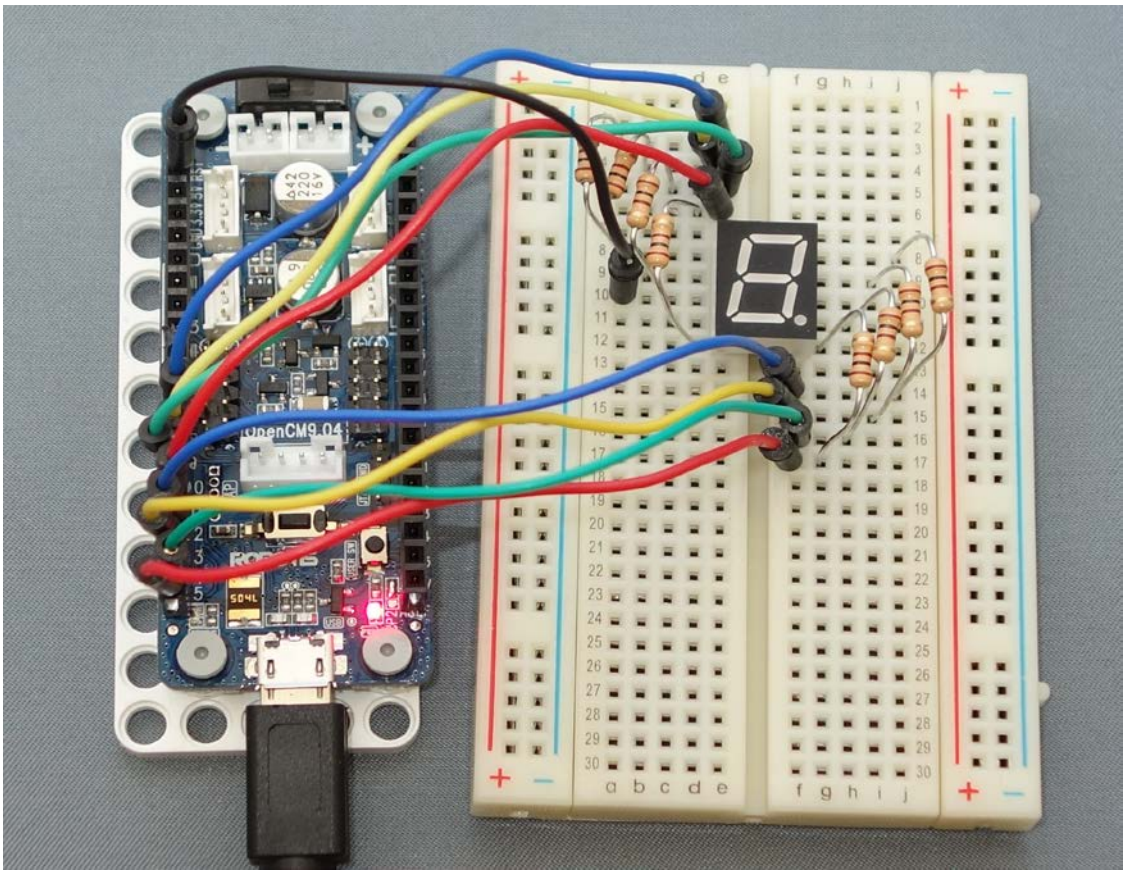


図 7.3　7 セグメント LED との接続