EduCake 概略(入門編)

1. 概要

日常生活の中では様々な玩具、電気製品、リモコン、工場生産ライン、コントローラ、各種機器制御等にマイコンや CPU 等が組み込まれています。電子回路等を学習した事が無い一般の方にとって、マイコンをどのように使うかは非常に難しい事です。様々な電子部品の機能、回路の設計、簡易言語の記述等を習得するには非常に多くの時間が掛かり、普通の方はこれらの理解の大変さから直ぐに学習を諦めてしまいます。

しかし、これらの流れが数年前にイタリアのエンジニアにより開発された Arduino と呼ばれるマイコンをベースとしたオープンソースのハードに依って解消されました。この Arduino が提供する開発環境は初心者にとって非常に使いやすいものでした。ソフトの構築手法が親切で、各国の言語での解説等がネット上から簡単に入手出来ます。　関連部品として提供するモジュール等が安価で入手しやすい事も強みとなっています。　しかし最も重要な事は回路が完全に公開されている事です。　これらのメリットから様々な分野で広く教材として用いられている Arduino です。

台湾メーカが独自の設計技術により製造した 86Duino 制御基板シリーズはこのような背景からリリースされました。　台湾メーカが独自研究で開発製造した 86Duino は Arduino と比較して以下の様な素晴らしい規格・仕様となっています：

- 300MHz 32 ビット　x86 SoC
- 128MB DDR3　メモリ内蔵
- PC レベルのネットワーク I/F
- USB 2.0 x 2 ポート
- Micro-SD
- オープンソースハードウェア
- DOS, Windows, Linux 等の OS に対応

- Arduino 完全互換開発環境を IDE にて提供　従来 Arduino を経験したユーザであれば新たに時間を掛けて習得する必要が無い。　元々Arduino に付属しているサンプルプログラムは全く修正せずに 86Duino で使用する事が出来る。大部分の Arduino が持っているファンクションライブラリも 86Duino でサポートされている。

86Duino シリーズの EduCake は初心者向けにデザインも洗練されたブレッドボードとして使える教育向け仕様です。　学生向けの教材等にも使える小型 PC をベースとした制御 Box です。

図 1-EduCake 外観

図 2-EduCake 背面

図 3-EduCake 前面

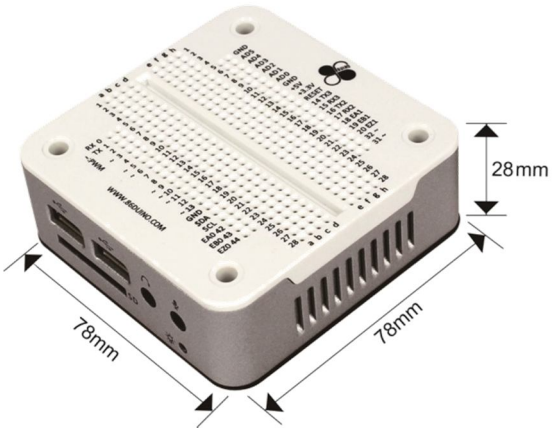

図 4-EduCake サイズ

EduCake を真上から見るとシルク上で Digital 0~13, Analog 0~5, GND, 5V, 3.3V, RX/TX 等がはっきり見えます。Arduino Leonardo と全く一緒のピン配列となっています。 (プログラミング記述に関しても 1 文字の変更も無く使用できる)

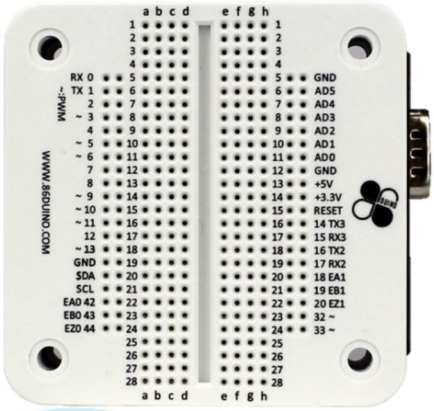

図 5- EduCake ピン配列

主なピン、コネクタ紹介:

A) Digital0~20, 31,32, 42~44　合計 26 個の一般のデジタル IO に使用可能。電流最大容量 16mA、電流制限により誤った使い方から保護する事で、初心者に大胆かつ安心して回路作成を試す事が可能。

B) Analog0~5　アナログ入力

C) TTL3 セット　RX/TX, RX2/TX2, RX3/TX3　各種の通信に利用可能

D) "~"標記の PWM サポートピンは Arduino 互換　但し 86Duino で構成した EduCake は Digital 13, 31, 32 の 3 ピンも PWM として使用可能

E) SCL/SDA は I²C 用ピン　Arduino 328 では I²C を使用する場合 Analog 4, 5 が犠牲となるが 86Duino では同時使用が可能。

F) EA0~1, EB0~1, EZ0~1 はモーションコントローラのエンコーダとして使用　この機能は Arduino ではオプションとなっている

G) 5V は 5V 入力のバイパス出力となる　3.3V は Max800mA 出力

おそらく Arduino の経験者でも親切な設計と感じる事でしょう。　操作も一緒、追加された機能は未経験者でも直ぐに使いこなせ、外見サイズも丁度良い手のひらに乗る適度なサイズ、金属のキャビネットで内部回路を保護しかつ美しいデザインに仕上がっています。

2. 開発画面

86Duino 開発画面を紹介すると DMP 開発チームの心遣いを感じる事になるでしょう。この画面は色以外 Arduino 標準 IDE と全く一緒で、当然操作方法も同じ、本当に使用者に取ってありがたい事です。以下は正式バージョンの IDE 画面です。お馴染みの画面ですよね？

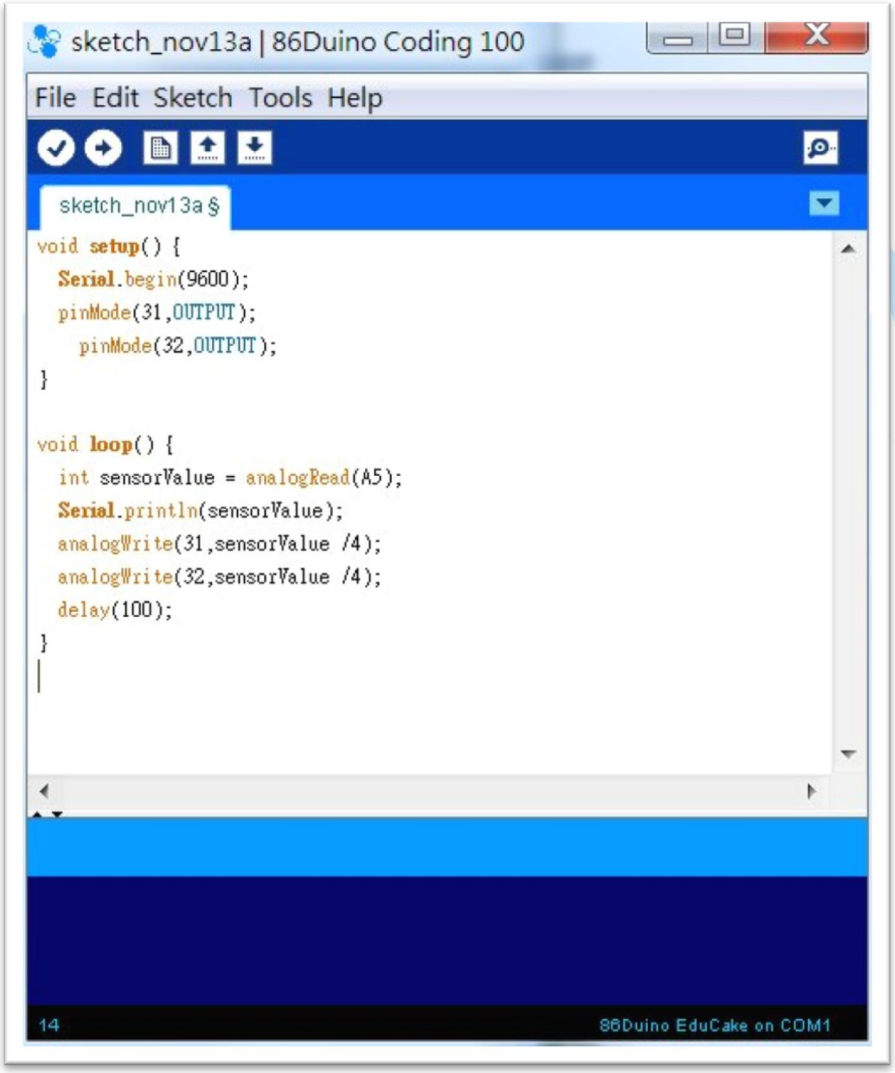

図 6-開発画面　Arduino と相似した画面

次に File-> Example と操作した画面が以下となります。

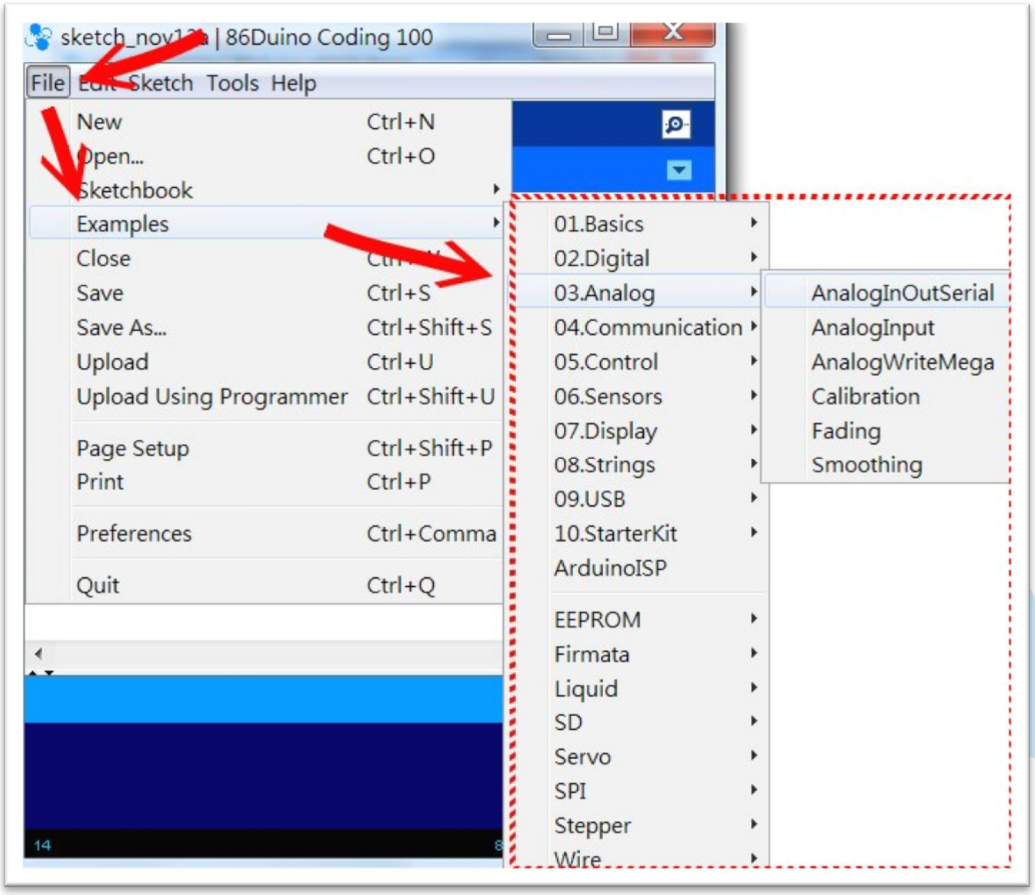

図 7-開発画面の操作形式も相似

操作関連の詳細を必要とする場合、以下のサイトをご覧ください。

http://www.86Duino.com　基本的に htpp://Arduino.cc の公式サイトの reference と同様ですので、既にご存知の方は確認不要です。通常多くの場合プログラムの記述では参照が必要でしょうが、多くの基本応用サンプルと配線図がありますので簡単に作業が出来る筈です。

86Duino 開発 IDE のファイルカタログ構造は Anduiro の構造と少し違いがありますが 86Duino の開発には支障がありません。

プログラムの記述完了後のインストールプロセスは全て一緒です。注意すべき点は Tools -> Board の選択時 86Duino EduCake を選ぶことです。間違えないで下さい(以下の通り)

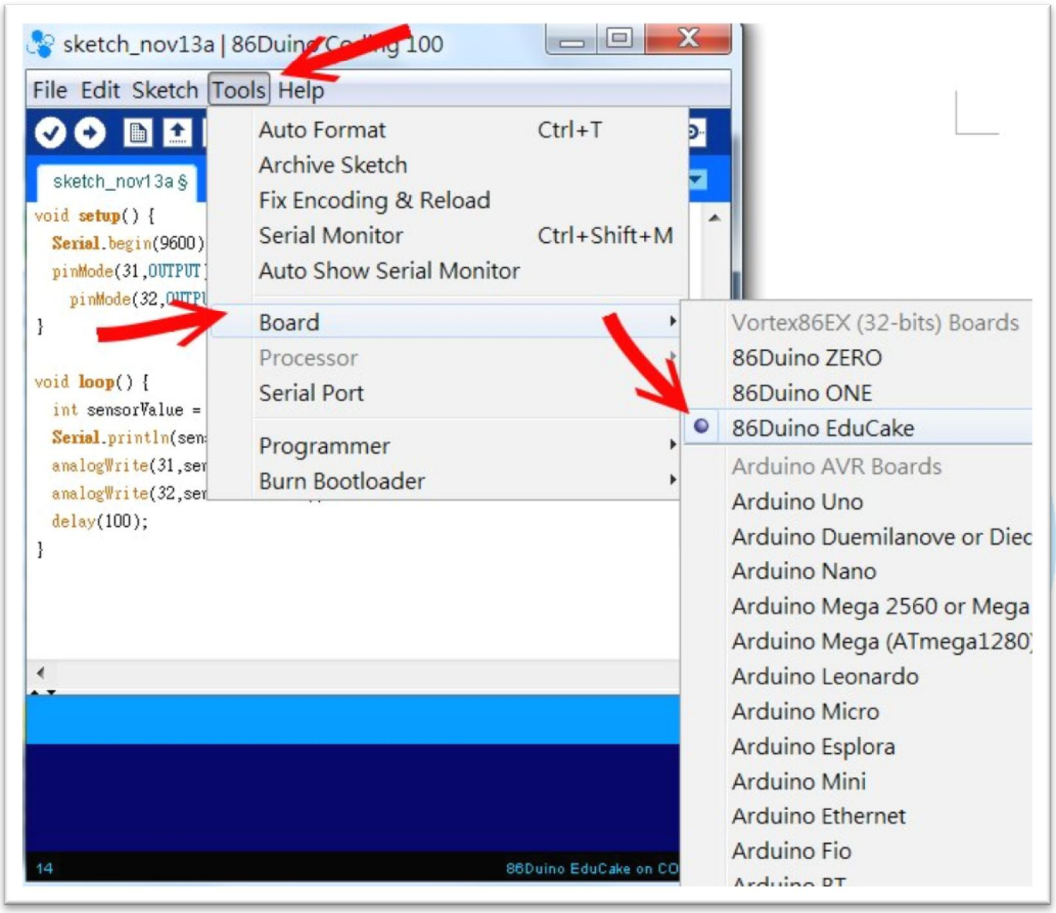

図 8-開発画面　86Duino EduCake を選択

Board の選択後 SerialPort も正しく選択する事が重要です。設定は以下の画面で設定が可能ですので、対応する COM 番号に都度設定が必要です。

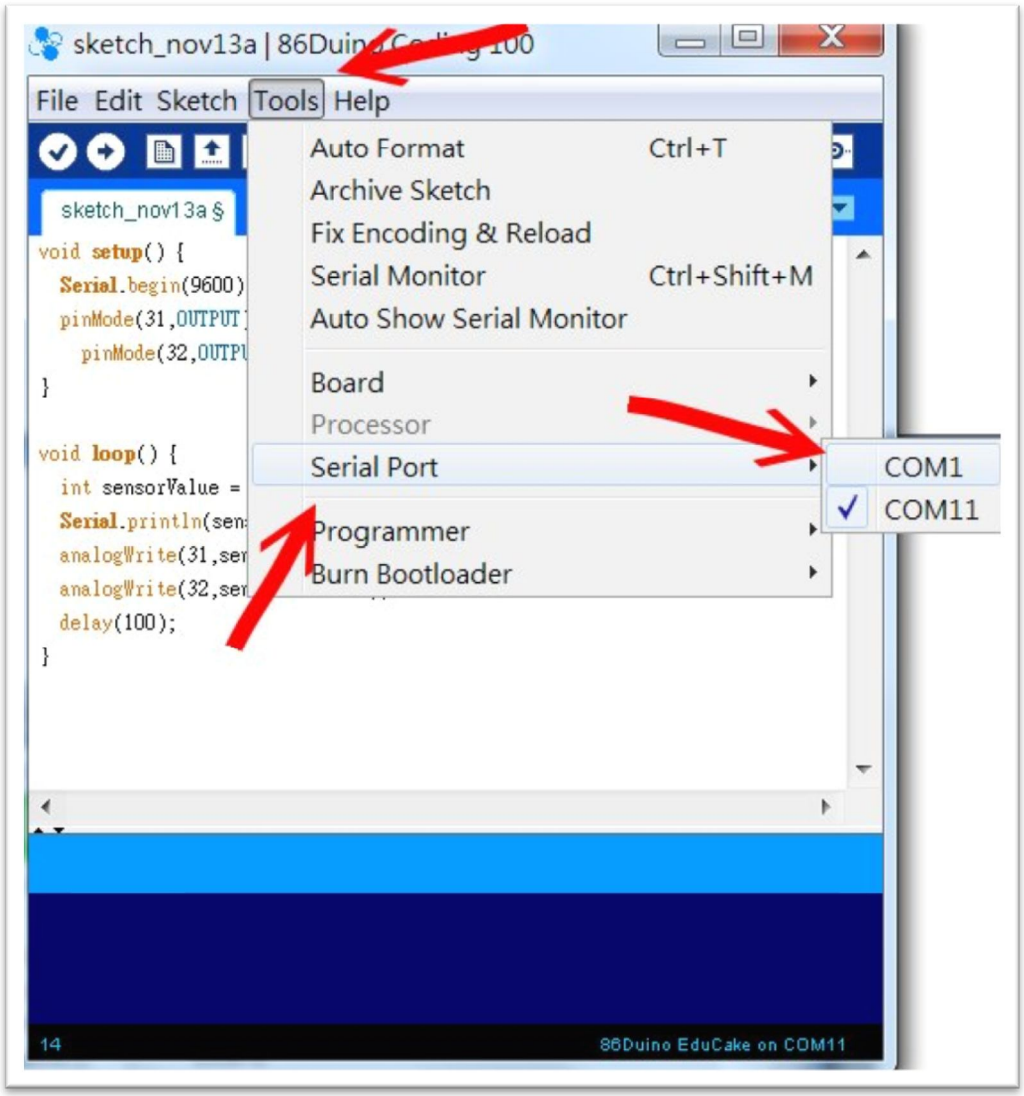

図 9-開発画面 COM11 設定

使用する COM を選ぶとプログラムを Educake にアップロードし実行する事が出来ます。以下の画面の左上にある✔記号はアップロードデータと IDE 上のデータ比較の為殆ど使いません。何故ならアップロード時に同様なチェックを行っているから、敢えて✔は必要ないでしょう。

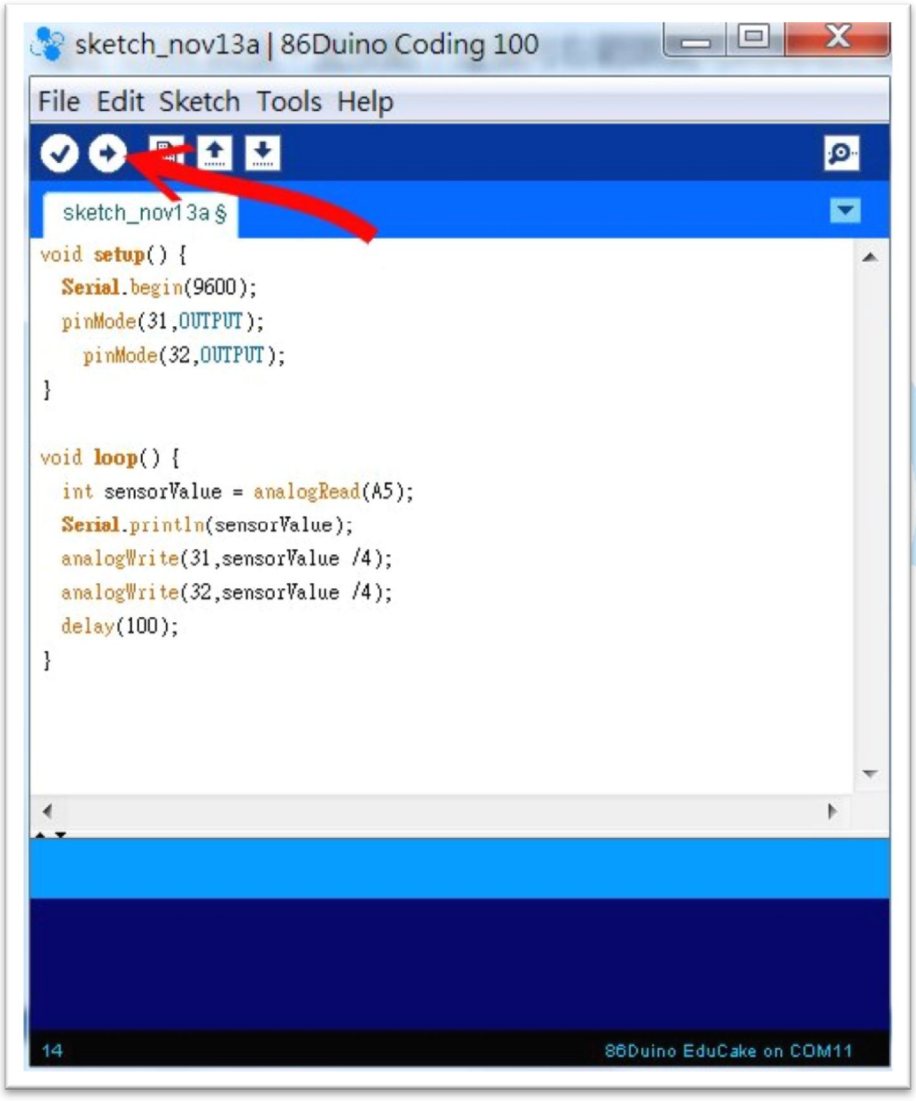

図 10-開発画面　　プログラミング開始！

3. プログラミング第一回

先ず、最も簡単なプログラミングを記述して実践しましょう。LED 1 個、抵抗 1 個を用意して以下の様に接続しましょう。抵抗を使わなくても構いません。何故なら出力電流は非常に小さく、過電流で壊れたりしません。抵抗を用意することは LED を守る事になりますので、出来れば準備して下さい。

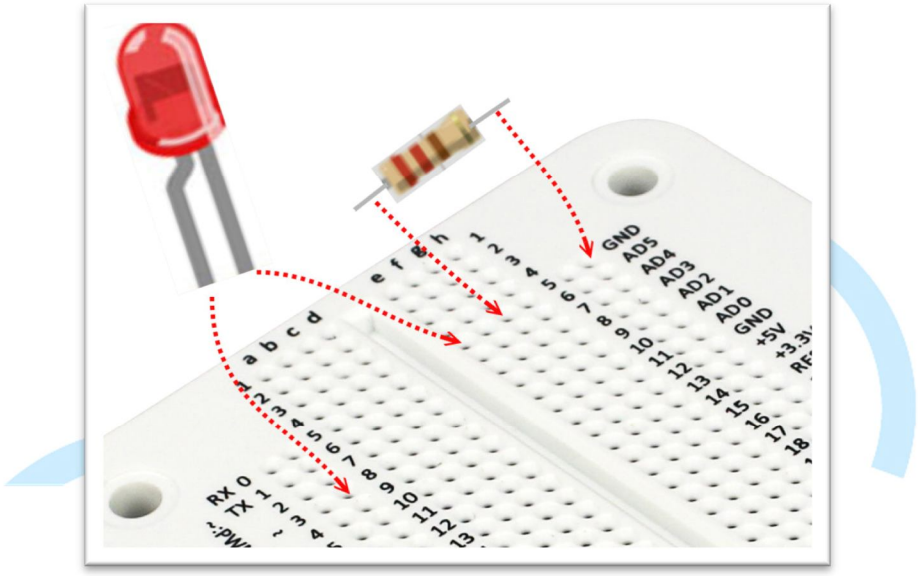

図 11-準備した LED と抵抗を Educake に接続

注意：LED の+側を digital ピン 3 に接続、そして IDE を起動後以下のプログラムを入力:

```
void setup()
{
  pinMode(3, OUTPUT); // digital ピン 3 を Output に設定
}
void loop()
{
  digitalWrite(3, HIGH); // digital ピン 3 を HIGH に設定->LED 点灯
  delay(1000);
  digitalWrite(3,LOW); // digital ピン 3 を Low に設定->LED 消灯
  delay(1000);
}
```

プログラムをアップロードした後、LED が 1 秒間隔で点滅する事が確認出来ます。このサンプルは Arduino で作成されたものと同じです。勿論このプログラムを修正する事も出来ます。Digital ピンの PWM 機能を使って analogWrite コマンドを併用すると LED の点灯を明るくしたり暗くしたり、または乱数を使えばランダムに明かりを変化させる事も出来ます。LED 一つで無限の変化を表現する事が出来ます。

これまでの使い方は Adruino と全く一緒でプロセスも非常に簡単です。経験者であろうが初心者であろうが問題無く使いこなせます。

4. プログラミング第 2 回

1 回目のプログラミングが成功すれば、次の段階に移ります。1 回目のプログラムの延長で 8 個の Digital ピンに LED を差し込んで応用します。配線は以下の通りです。

先ず、8 個の LED と抵抗を差し込みます。

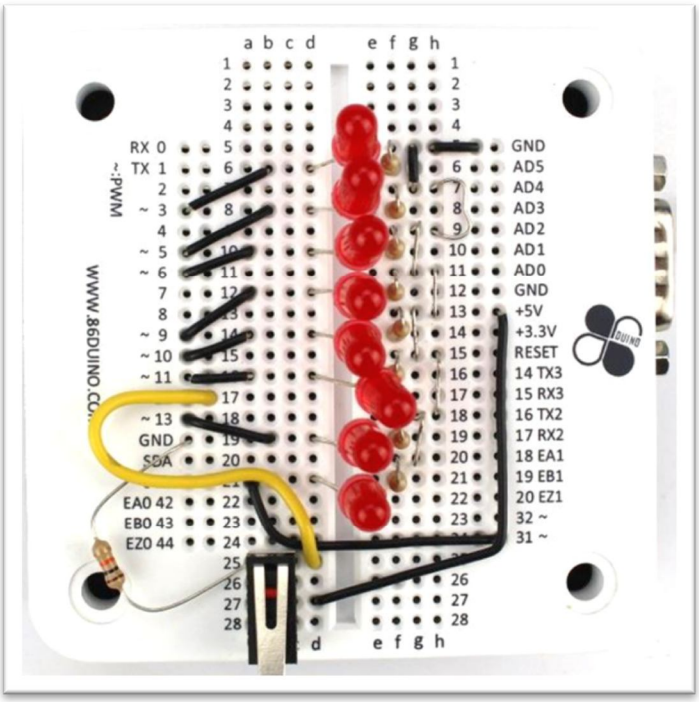

図 12-8 個の LED と抵抗を差し込んだ状態

以下が別アングルです

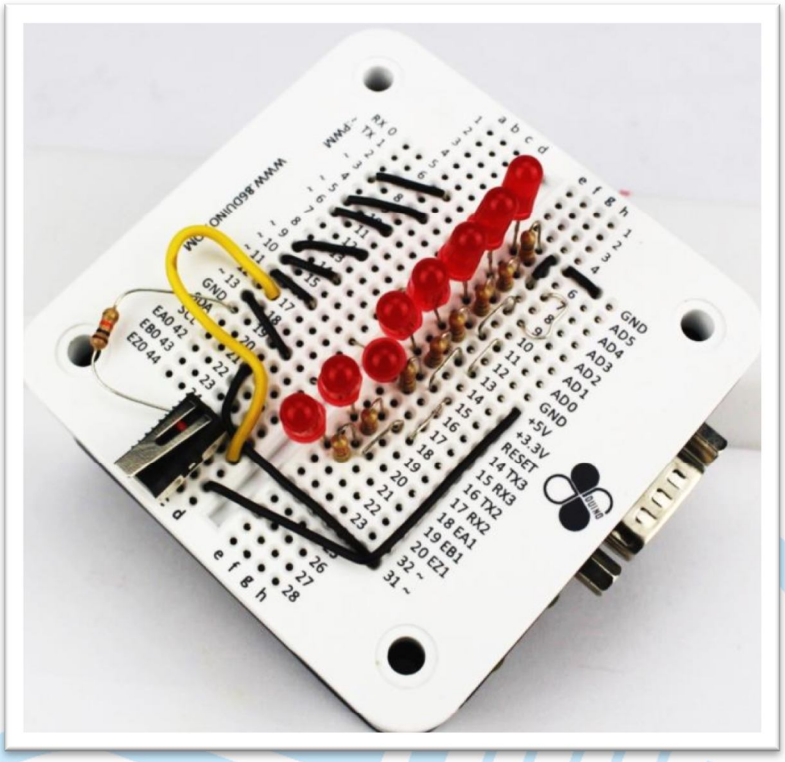

図 13-角度を変えた 8 個の LED と抵抗を差し込んだ写真

後にスイッチを使った処

理を行いますので、スイッチ回路も先に取りつけます。

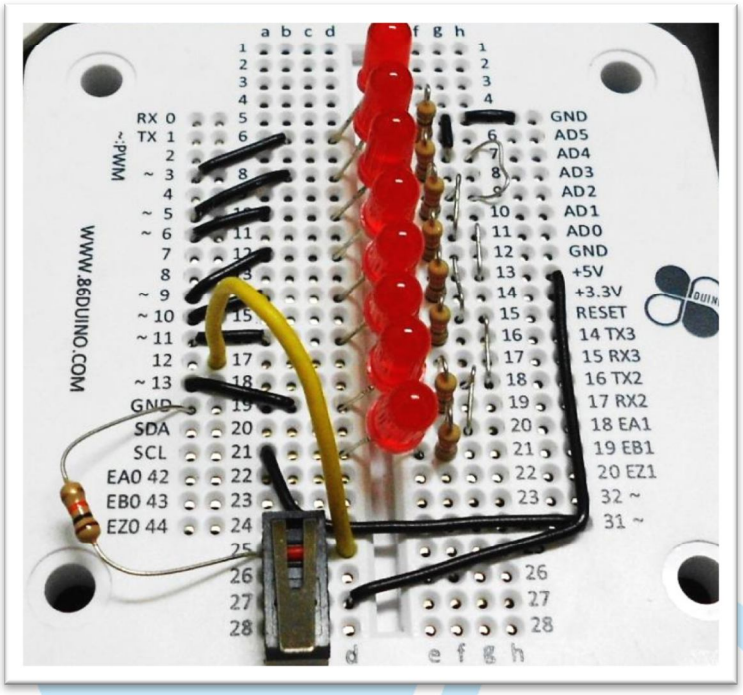

図 14-スイッチ、配線を取りつけた状態

配線取り付け後、この配線を用いて様々な事が行えます。まず、明かりを連続して点灯させる実験を行います。プログラムソースコードは以下の通りです。

```
// 8 個の Digital ピンを使用
// 敢えて PWM～記号を選択(応用の幅が広がる)
// 配線の変更が少なくなり学習に便利
int led[]={
   3,5,6,9,10,11,13,31}; // 8 個のピン番号を配列として記憶、アクセスに便利
int pos =0;
void setup() {
   for(int a=0;a<8;a++) // 先ず、8 個のピンを出力に設定
      pinMode(led[a],OUTPUT);
}

void loop() {
   digitalWrite(led[pos],LOW); // 前段の LED を消灯する
   pos=(pos+1)%8; //次に点灯する LED の位置を設定
   digitalWrite(led[pos],HIGH); // 次の LED を点灯
```

```
    delay(200); // 遅延、点灯間隔を調整　大：ゆっくり、小：速い
}
```

以上で 8 個の LED が順番に点灯／消灯を繰り返します

以下の記述は

```
pos=(pos+1)%8;
```

以下と同じプログラミングとなり、どちらでも使えます

```
pos ++;
if (pos>=8) pos =0;
```

最後の行 delay(200)は()内の変数で点灯のスピードを変化させる効果があります。

5. プログラミング第 3 回

前のプログラムが完了したら、この章では COM の通信機能を使ってパソコン等との通信により更なる応用が出来ます。パソコン側でも対応する応用プログラムが必要ですが通信ソフトを用いて Educake の制御が出来ます。パソコン側、Educake 側共にプログラミングが必要です。

EduCake 側でのプログラミング:

```
int led[]={
   3,5,6,9,10,11,13,31};
int pos =0;
void setup() {
   Serial.begin(9600); // COM の通信速度設定
// 通信速度 9600/19200/38400/115200 等が一般的
// 両者の通信速度は一緒にする必要がある
   for(int a=0;a<8;a++)
      pinMode(led[a],OUTPUT);
}

void loop() {
```

```
    char ch;
    if (Serial.available()) // パソコン側からのデータ有無を確認
    {
      ch=Serial.read(); // データがあれば 1 回に 1 Byte を読み取る
      if (ch>='1' && ch<='8') // 入力として 1~8 の数字のみ有効
        digitalWrite(led[ch-49],HIGH);// 対応する LED を点灯
    }
    delay(200);
}
```

以下のコマンドは

```
digitalWrite(led[ch-49],HIGH);
```

ch-49 の意味はキャラクタとして数字の 1～8 は ASCII コードの 49～57 に対応しており、入力記号を 49 マイナスし 0～7 に変換する事で上記プログラミングの Digital ピンに対応する事が出来る。この際、IDE 画面の右上にある検索記号をクリックする事で通信モニタが確認出来る。1～8 を入力する事で対応する LED が点灯する。画面は以下の通り

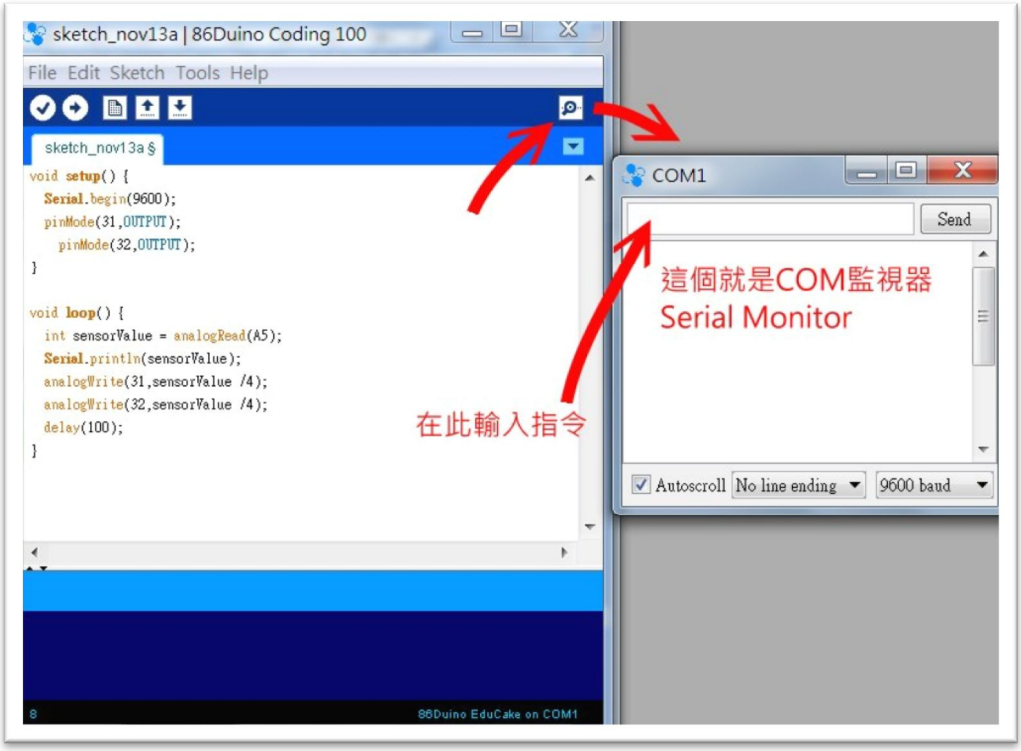

図 15-コマンド入力

矢印の箇所で 1～8 の数字を直接入力して対応する LED を点灯させる。この画面はプログラミング開発の過程で、正しくプログラミングが行われているかの検証や通信機能のテストを行うのに良く使われる。

但し注意して欲しい事は COM が使われると、後々VB/C#を使って UI 画面を作成し EDUCAKE を操作する場合 SerialMonitor 画面を開く事は出来ない。プログラムが衝突するからです。従って UI 画面を実行する場合は SerialMonitor 画面は必ず閉じて下さい。そうしないとパソコンのリセットが必要になり、その後 UI での認識となります。

COM を記述出来る全ての通信ソフトで Educake との通信が可能です。ここでは VS2008(2005～2013 版全て使用可能)の VB.net/C#を使ってみましょう。画面は以下の通り。(中国語 Ver.となっています)

図 16-LED 制御画面例

先ず、COM のボーレートを 9,600 に設定。画面の“連線”ボタンをクリックして Educake と通信する。

この部分に関しては、Windows のプログラミング手法に依るものなので説明は省きますので VB に含まれるサンプルプログラムを参考にして下さい。より多くの変化を求めるのであれば Mouse スクロールの動きによって点灯／消灯と応用させる事も可能です。更に、可変抵抗と結合し点灯の明るさを制御する事も可能です。

一部の人には C#が必要になりますが、VB2C#のキーワードでネット上で検索が可能です。

敢えて記述するとこのプログラムには少し欠点があります。一度点灯すると消灯の処理が含まれてない事です。目的の LED をボタン選択して点灯させたかどうか明確にする為、以下の処理を追加して消灯処理を行う。

```
digitalWrite(led[ch-49],HIGH);
```

次の様に修正

```
{
 digitalWrite(led[ch-49],HIGH);
 delay(1000);
 digitalWrite(led[ch-49],LOW);
}
```

修正内容は 1 秒の点灯後に消灯が行われる。

6. プログラミング第 4 回

前の基本機能を理解したうえで次のステップに移ります。この章ではデジタルに関する紹介です。LED 点灯を使ったゲームを作成する場合、以下のプログラムが参考として使えます。

```
int led[]={
   3,5,6,9,10,11,13,31};

// ゲーム：LED 移動
// 3 つの変数を使って LED の点灯位置と方向を制御
```

```
int nowPos =2; //  最初に点灯する LED
int midPos=1; //  中間に点灯する LED
int lastPos=0; //最後に点灯する LED
int dir=1; //  点灯する LED の方向    1:小->大, -1:大->小

int spd = 20; //  移動速度

void setup() {
  Serial.begin(9600);
  for(int a=0;a<8;a++)
    pinMode(led[a],OUTPUT);
  randomSeed(analogRead(0)); //  乱数関数初期化
}

void loop() {
  if (Serial.available())
  {
    char ch=Serial.read();
    if (ch=='1') // UI からの入力操作を実行
    {
      spd =20;
      // nowPos =2;
      midPos=nowPos-1;
      lastPos=nowPos-2;
      dir =1;
    }
  }
  if (spd<220) //  速度が段々ゆっくりとなり、220 以上になった時移動が止まり、乱数を発生
  {
    digitalWrite(led[nowPos],HIGH); //  先頭の LED を最も明るく点灯
    if (midPos<8 && midPos>=0)
```

```
    analogWrite(led[midPos],40); //  中間の LED を次に明るく点灯
  if (lastPos<8 && lastPos>=0)
    analogWrite(led[lastPos],15); //最後の LED を暗く点灯

  delay(spd);

  spd +=5;
  //速度が段々ゆっくりとなり、220 以上になった時移動が止まり、乱数を
発生
  if (spd>=220 )
  {
    if (midPos<8 && midPos>=0) //  以下はこれらの配列を消灯する
      digitalWrite(led[midPos],LOW);
    if (lastPos<8 && lastPos>=0)
      digitalWrite(led[lastPos],LOW);
    digitalWrite(led[nowPos],LOW);

    digitalWrite(led[random(0, 8)],HIGH);//乱数が発生した位置の LED が点
灯

    spd =1000;
  }

  if (lastPos<8 && lastPos>=0)
    digitalWrite(led[lastPos],LOW); //最後尾の LED を消灯して、前進の準備
  lastPos=midPos;
  midPos=nowPos;
  nowPos+=dir;
  if (nowPos>7)
  {
    nowPos=7;
    midPos=8;
    lastPos=9;
```

```
            dir=-1;
        }
        else    if (nowPos <0)
        {
            nowPos=0;
            midPos=-1;
            lastPos=-2;
            dir=1;
        }
    }

    delay(spd/3);
}
```

最後の行は

```
delay(spd/3);
```

spd 変数によって delay を変化させる。この部分の速度の変更はプログラミングによりいろいろ設定できるが、UI を使ってスピードを変更する事も可能。

次に UI 若しくは SerialMonitor を使って制御したくない場合、直接スイッチ(Digital12 に取りつけた)を使ってコントロール可能。修正後のプログラムコードは以下の通り。 (重複する箇所は。。。で省略):

```
。。。
void setup() {
  。。。
  pinMode(12,INPUT);   // この行を追加して 12 ピンを入力とする
}

void loop() {
  int bb;
```

```
    bb=digitalRead(12); //  ピン 12 の状態を読み取る
    Serial.println(bb); //  スイッチの ON/OFF をデバッグ用に出力
    if (bb==1) // 1：スイッチ ON
       run_again(); //再度実行

    。。。
}

void run_again() //  上記の LOOP 以外に、以下を記述して再初期化
{
    spd =20;
    midPos=nowPos-1;
    lastPos=nowPos-2;
    dir =1;
}
```