電気刺激キットプログラム解説

2014/7/8

梶本裕之

```
//Simple Communication Sample
// Send 61 ch stimulation data at 60 fps.
// Receive 61 ch impedance data at 12 fps.

import processing.serial.*;

// The serial port:
Serial myPort;

//user definitions
String COM_PORT="COM4"; //change this!
```

↑本キットではマイクロコントローラとして mbed (www.mbed.org)を使っています．mbed は PC に USB 接続すると USB メモリとして表示されると同時にシリアルポートとしても認識されます． mac および Linux ではこれはデフォルト動作ですが，Windows の場合はドライバをインストールしておく必要があります．下記を参照ください．

http://mbed.org/handbook/Windows-serial-configuration

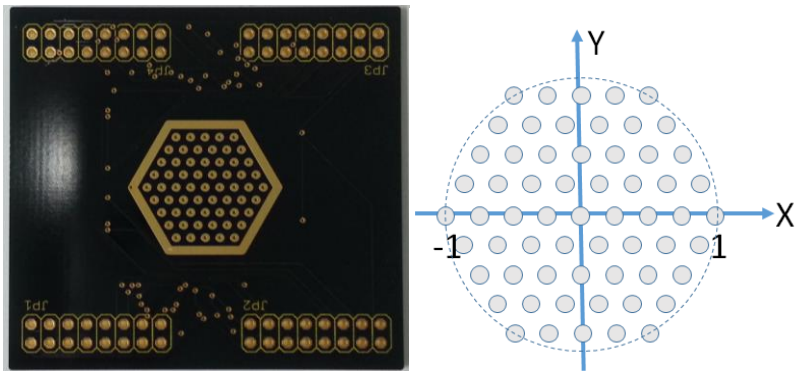
```
//hardware definitions
//     -1.0             0                  1.0
//0.87          4    3    2    1    0
//            10    9    8    7    6    5
//          17 16 15 14 13 12 11
//        25 24 23 22 21 20 19 18
//0.0 34 33 32 31 30 29 28 27 26
//        42 41 40 39 38 37 36 35
//          49 48 47 46 45 44 43
//            55 54 53 52 51 50
//-0.87           60 59 58 57 56

int ELECTRODE_NUM=61;
float[] Electrode_Pos_X ={
```

0.5,　0.25,　0.0,　-0.25,　-0.5,
0.625,　0.375,　0.125,　-0.125,　-0.375, -0.625,
0.75,　0.5,　0.25,　0.0,　-0.25,　-0.5,　-0.75,
0.875,　0.625,　0.375,　0.125,　-0.125,　-0.375,　-0.625,　-0.875,
1.0,　0.75,　0.5,　0.25,　0.0,　-0.25,　-0.5,　-0.75,　-1.0,
0.875,　0.625,　0.375,　0.125,　-0.125,　-0.375, -0.625,　-0.875,
0.75,　0.5,　0.25,　0.0,　-0.25,　-0.5,　-0.75,
0.625,　0.375,　0.125,　-0.125,　-0.375,　-0.625,
0.5,　0.25,　0.0,　-0.25,　-0.5};

float[] Electrode_Pos_Y ={
0.8660254 , 0.8660254 , 0.8660254 , 0.8660254 , 0.8660254 ,
0.649519 , 0.649519 , 0.649519 , 0.649519 , 0.649519 , 0.649519 ,
0.4330127 , 0.4330127 , 0.4330127 , 0.4330127 , 0.4330127 , 0.4330127 , 0.4330127 ,
0.21650635 , 0.21650635 , 0.21650635 , 0.21650635 , 0.21650635 , 0.21650635 , 0.21650635 , 0.21650635 ,
0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 , 0.0 ,
-0.21650635 , -0.21650635 , -0.21650635 , -0.21650635 , -0.21650635 , -0.21650635 , -0.21650635 , -0.21650635 , -0.21650635 ,
-0.4330127 , -0.4330127 , -0.4330127 , -0.4330127 , -0.4330127 , -0.4330127 , -0.4330127 ,
-0.649519 , -0.649519 , -0.649519 , -0.649519 , -0.649519 , -0.649519 ,
-0.8660254 , -0.8660254 , -0.8660254 , -0.8660254 , -0.8660254};

↑本キットでは電気刺激用の電極として 2mm 間隔，最密充填配置の 61 個の電極を使っています．これは指先全体を刺激できるサイズを意図しています．



正方形の格子ではないため，電極の位置と座標の対応関係は簡単ではありません．つまり正方形の格子であれば，座標(x,y)に刺激を与えたいときに使う電極はすぐわかりますが，今回

の特殊な配置ではちょっと工夫が必要になります.
プログラムのこの部分は電極の座標のデータベースになっています. X 座標, Y 座標の値が格納されています. 電極全体の六角形の一辺の長さを 1 とした時の数字で規格化されています. ですから大体, 半径 1 の円の中に電極が配置されていると考えればよいです.

```
//software definitions
int PC_MBED_STIM_PATTERN=0xFF;
int PC_MBED_MEASURE_REQUEST=0xFE;
int MBED_PC_MEASURE_RESULT=0xFF;
```

↑MBED にたいするシリアル通信のヘッダです

```
float y=-1.5, x=-1.5;
color statuscolor;
int timer=0;
byte[] stimulation = new byte[ELECTRODE_NUM];
int[] impedance = new int[ELECTRODE_NUM];

void setup(){
  int pin;

   size(400,400);

  // Open the port. baud rate=921600
  myPort = new Serial(this, COM_PORT, 921600);
```

↑
mbed に対するシリアル通信は 921.6kbps というシリアル通信としては比較的早い速度を使っています.

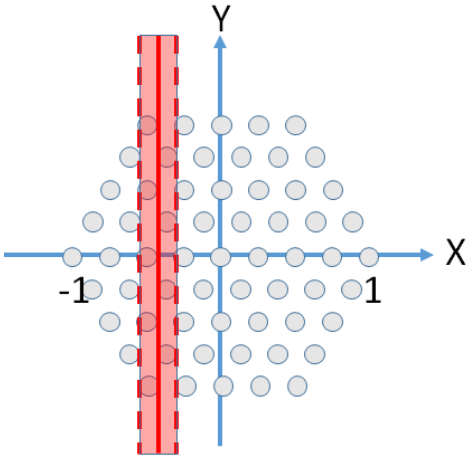
```
}

//assume 60fps reflesh rate
void draw(){
  int i,pin,rcv;
```

```
//present moving vertical line
x=x+0.05;
if(x>1.5)x=-1.5;
for(pin=0;pin<ELECTRODE_NUM;pin++){
  if(Electrode_Pos_X[pin]>x-0.125 && Electrode_Pos_X[pin]<x+0.125){
    stimulation[pin]=40; //Pulse Width in micro-seconds. Should be less than 50. (Larger the stronger sensation
  }else{
    stimulation[pin]=0;
  }
}
```

↑刺激する電極を指定する部分です．ここでは横に動く縦棒を刺激パターンとして表現しています．

x-y 平面（x:横軸，y:縦軸として）で縦棒は，x がある定数であることを意味します．ここでは電極全てに対して，各電極と縦棒（x=定数の直線）との間の距離が一定値 0.125（この数値は電極間隔の半分に当たります）以下であるかどうかを判定し，そうであれば刺激するために stimulation という配列の該当部分に刺激量を代入しています．

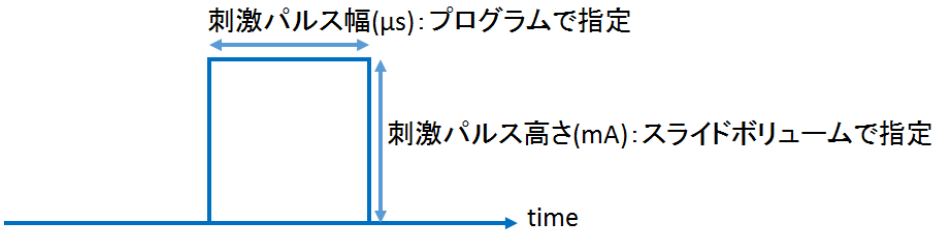


例えば斜めの棒を表示する場合は棒の直線を表す式と点との間の距離を計算する公式を使えばよく，

ある座標点に刺激を与えたい場合は，その座標（x,y)と各電極の座標の間の距離を計算し，その距離が一定値以下（例えば 0.125 以下）である場合に刺激する，などとすれば良いことになります．同様に円なども表示できます．

ここでは刺激量として 40 を指定しています．これは電気刺激パルスの幅を指定しており，

40 マイクロ秒を意味します．少なければ刺激は弱く，大きければ刺激は強くなります．しかし電気刺激の場合，パルス幅を倍にしても刺激の強さが倍になるわけではありません．当面は 40 に固定で良いと思われます．100 位上の指定はしないでください．
本キットでは刺激の強さを手元のスライドボリュームでも変えられるようになっていますが，このスライドボリュームでは電気刺激パルスの高さ（電流値）を調整しています．



```
/*
  //present moving horizontal line
  y=y+0.05;
  if(y>1.5)y=-1.5;
  for(pin=0;pin<ELECTRODE_NUM;pin++){
    if(Electrode_Pos_Y[pin]>y-0.125 && Electrode_Pos_Y[pin]<y+0.125){
      stimulation[pin]=40;
   }else{
      stimulation[pin]=40; //Pulse Width in micro-seconds. Should be less than 50. (Larger the stronger sensation
    }
  }
*/

  //send stimulation signal to mbed
  myPort.write((byte)PC_MBED_STIM_PATTERN);
  for(pin=0;pin<ELECTRODE_NUM;pin++){
    myPort.write(stimulation[pin]);
  }
```

↑ここでは刺激パターンを mbed にシリアル通信で送信しています．
最初に刺激パターンであることを示すヘッダを送り，次に 61 電極分のパルス幅指令値（刺激しない電極は 0）を送信しています．合計 62byte が送信されます．
多バイトの送信を一回で行う関数がやや挙動不審だったため，1 バイトずつ送っています．

```
  //receive impedance data (every 5 times)
  timer=(timer+1)%5;
  if(timer==0){
    myPort.write((byte)PC_MBED_MEASURE_REQUEST);
    rcv = myPort.read();
    for(pin=0;pin<ELECTRODE_NUM;pin++){
      rcv=myPort.read();
      impedance[pin]=rcv;
      impedance[pin]=(150-impedance[pin])*4;
      if(impedance[pin]>255)impedance[pin]=255;
      else if(impedance[pin]<0)impedance[pin]=0;
    }
  }
```

↑ここでは各電極の抵抗値の情報を mbed から得ています．この抵抗値は，電極に接触している皮膚の抵抗値なので，皮膚が接触しているかどうかがわかります．これにより接触位置も接触面積もわかりますから，タッチパネルとして使えることになります．

細かなことですが，本ハードウエアでは刺激電流値を制御する回路を採用しており，計測によって得られるのは一定の電流を与えた時の電圧値になります．つまり皮膚抵抗が高いほど電圧は大きくなり，結果として mbed から帰ってくる数値は大きくなります．このプログラムでは接触した時（＝抵抗値が下がった時＝電圧が下がった時）を画面で明るく表示したいため，多少数値をいじる工夫をしています．

この部分は今後のサンプルプログラムで改良される可能性の高いところです．例えばプログラムを起動して最初の数回のデータを平均して保存しておき，その値との差分だけを返すことで較正されたデータを得ることが出来ます．また得られたデータから，接触重心を計算すればマウスカーソルとして使えるはずです．接触面積を計算すれば指の押下圧の代替となることも期待できます．さらに研究上の発展トピックとしては，皮膚の抵抗値に応じた刺激量の調整をすることで，より不快感の無い電気刺激ができるようになることがわかっています．

なおここでは Processing の draw 関数(秒間 60 回呼び出し)内で 5 回に 1 回だけこの計測を行っています．これは毎回計測するとシリアル通信がややビジーになり，プログラムが滑らかに動かなくなることが有るためなので本質的ではありません．

```
  //draw electrodes. Red: Stimulation, Green: Impedance (touch)
  for(pin=0;pin<ELECTRODE_NUM;pin++){
    statuscolor=color(stimulation[pin]<<2,impedance[pin],100);
    fill(statuscolor);
    ellipse(Electrode_Pos_X[pin]*180+200,200-Electrode_Pos_Y[pin]*180,20,20);
  }
```

↑電極を画面に配置し，各電極の色で刺激パターンと接触状態を表現しています．刺激パターンは赤で，接触状態は緑で表現されます．

```
}
```