Gainer-mini で広がる計測・制御の世界

鳥取環境大学環境学部　足利裕人

目次

Ver.1　2012 年 3 月 23 日

1　はじめに

高度情報化社会が訪れ，コンピュータを用いた計測・制御および通信機能は，現代社会の医療や産業，交通等あらゆる分野の基盤となっている。一方，社会の変化に比べ，学校現場での理科実験へのコンピュータの活用は，遅々とした変化しか感じられない。しかし，誰でも手軽にコンピュータに様々な機器を接続し，自分用に発展させることができるフィジカルコンピューティングが登場し，理科での計測・制御の救世主になろうとしている。

計測・制御の理科実験への活用が進まない原因には，高価な教材パッケージや，プログラミング等ソフトウェアのしきいの高さがある。

フィジカルコンピューティングとは，既存のディスプレイやキーボードやマウスだけではなく，必要に応じて新しいハードウェアを使って，コンピュータと人間とのコミュニケーションを実現するという考え方である。

コンピュータのユーザインタフェースをハードウェアを使って拡張したり，コンピュータで電子機器を制御したりするのに便利な手法であり，2006 年に小林茂氏らによって Gainer(ゲイナー) ツールキットが開発された。 Gainer を利用することにより，センサーからの情報をパソコンに取り込んだり，パソコンからアクチュエータの制御を行ったりすることができる。 Gainer は，工学系・美術系(メディアアート系)の学生や研究者, ハードウェアに興味があるコンピュータファンを意識して開発された。また，Gainer の小型版 Gainer mini が株式会社アールティにより発売されている。

ここでは動作させるための言語に，フリーソフトの processing を用いた。Processsing は BenFry 氏と CaseyReas 氏によってつくられ，デザイナー／アーティスト向けの視覚デザインを行うための，プログラミング言語と開発環境である。

Java をベースにした手軽なプログラミング環境であり，エディタが内蔵され，すぐに作り始めることができる。また，作ったソースコードはクリックひとつですぐに動作し，作品を Web 用に諸津力し，手軽に公開できる。また，マウスだけでなくマイク，カメラ，など様々な入力機器を扱える。

2　Gainer-mini を動作させよう

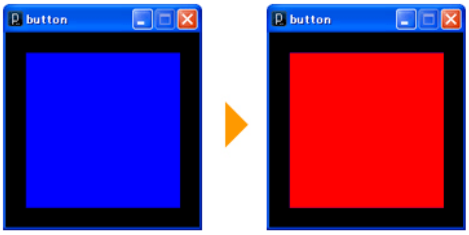
① USB ドライバーのインストール

http://www.gainer-mini.jp/?page_id=51　の指示にしたがってインストールします。

② 動作用ソフトウェア「Processing」の download と setup

http://www.gainer-mini.jp/?page_id=58　の指示にしたがってインストールします。

Processsing は BenFry 氏と CaseyReas 氏によってつくられ視覚デザインを行うためのプログラミング言語と開発環境です。フリーソフトであり，Gainer-mini の動作環境が整っています。



③ 動作確認をしよう。

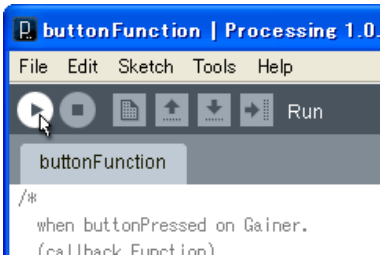
Gainer-mini を USB ケーブルでパソコンと接続します。Gainer-mini に搭載された青の LED が点滅していれば, インストールは成功です。

次に，②のリンクのページの「Button Function」のソフトを実行します。

Gainer の基板上の黒いボタンを押したり離したりして，画面上の四角形が青から赤に，また青に変化すれば完了です。

このプログラムは以下のリンクにあります。

processing-1.5.1¥modes¥java¥examples¥Topics¥Gainermini¥button



“button”のアイコンをダブルクリックして，メニューバーの左端“FILE”の真下の play マークのボタンが実行“run”ボタンです。

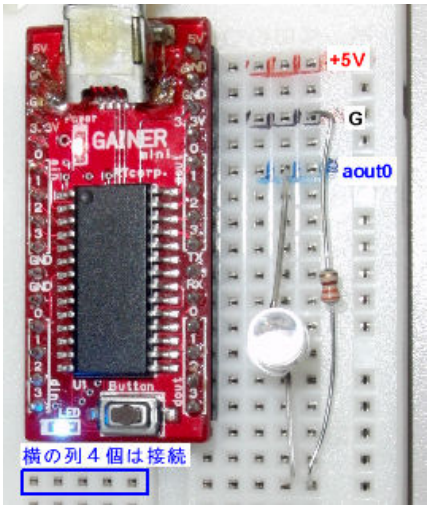
Processing のプログラムは以下のようになります。

```
/*                                  説明など，実行しない部分です。/*と*/で囲みます
  when buttonPressed on Gainer.     タイトル
*/
import processing.gainer.*;         Gainer のライブラリーを呼び出します
Gainer gainer;
void setup(){                       setup という関数を定義します。「初期化」に必要な関数です
  size(200,200);                    画面のサイズを指定します
  background(0);                    背景を黒にします
  gainer = new Gainer(this);        Gainer が使えるように初期化します
}
void draw(){                        draw という関数を定義します。「メインループ」で必要です。
  if(gainer.buttonPressed){         もし，Gainer に搭載されたボタンが押されたら
    fill(255,0,0);                  画面を青で塗ります
  }else{                            そうでなければ
    fill(0,0,255);                  画面を赤で塗ります
  }
  rect(20,20,160,160);              長方形(20,20)-(160,160)を描きます。
}
```

『値を変えて，色や長方形の大きさを変えてみよう。プログラムが理解しやすくなります』

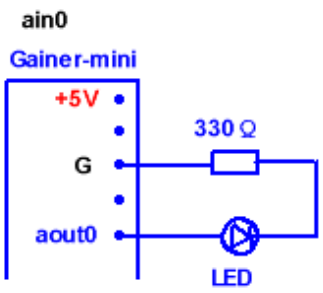
3　アナログ出力で LED を点灯しよう

関数 draw　部分を少し変更して，ブレッドボード上に配線した LED を点灯します。



まず，回路は右図のように組みます。ブレッドボードに Gainer-mini と LED, 330Ω の保護抵抗（LED に流れる電流は aout0 の 5V から赤色 LED の電圧降下 1.8V を引き，その 3.2V を 330Ω で割り算して約 10mA になります）を配線します。アナログの出力ポートの 0 番に+5V を出力することで，LED が点灯します。

では，プログラムを書き換えてみましょう。

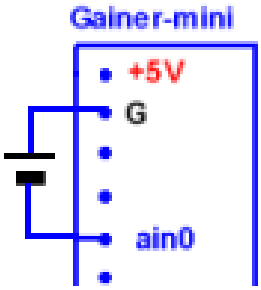


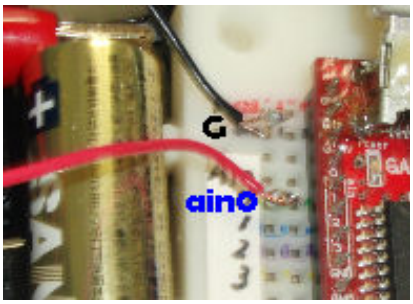

```
void draw(){
  if(gainer.buttonPressed){
    fill(255,0,0);
    gainer.analogOutput(0,255);
  }else{
    fill(0,0,255);
```

```
      gainer.analogOutput(0,0);
    }
    rect(20,20,160,160);
  }
```

赤で示した部分が aout0 に 255（+5V に対応）と 0V を出力する部分です。

4　アナログデータを測ろう

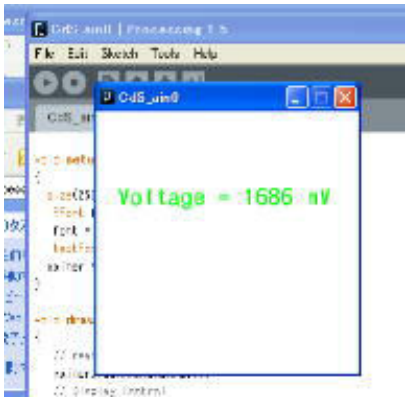




(1)　ain0 ポートで乾電池の電圧を測ろう

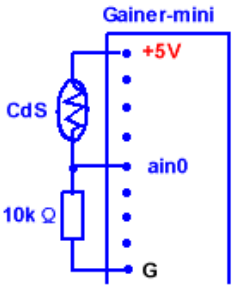
0（0V）~255（5V）のアナログの値をアナログ入力ポートの 0(ain0)で読み込みます。右図のように，みのむしクリップのー端子を Gainer-mini の G 端子に，+端子を ain0 端子に接続します。

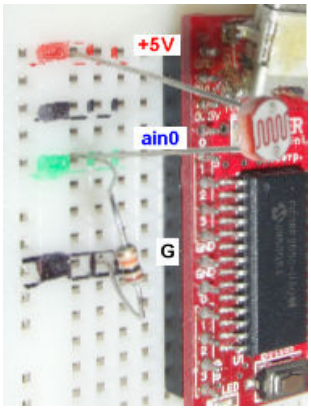
電池の電圧を測ってみましょう。5V が 255 なので，ain0 の測定値に 5/255=1/51 をかけると V の単位の値で表されます。ここでは単位を mV　にするために，さらに 1000 倍しています。ファイル名は”voltage_ain0”です。




```
/*
 * AnalogInput(0)
*/
 import processing.gainer.*;
Gainer gainer;
void setup() {
  size(250, 250, P3D);
    PFont font;                                  PFont オブジェクト変数の定義
    font = loadFont("JSSGothic-Md-48.vlw");   フォントファイルを PFont オブジェクトに読み込む
    textFont(font, 24);                          使用するフォントを選ぶ
  gainer = new Gainer(this);
}
 void draw() {
    // read analog port
    gainer.peekAnalogInput();                    一度だけアナログポートの状態を analogInput[]に取得する
    // Display Control
    background(255,255,255);                  前の数値の表示を白で消す
    fill(255, 0, 0);                             赤の文字で次の text の内容を描く
    text("Voltage = " +  (gainer.analogInput[0]  * 1000  /  51  + " mV") ,20, 90);        (20,90)の位置に ain0 の値を電圧値に換算して表示する
    }
```

(2)　CdS（照度センサー）を用いて明るさを測ろう

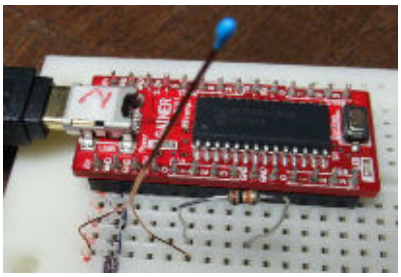




右図のように回路を組むと，Cds と抵抗で，5V の電圧を分けあいます。明るいと Cds の抵抗値が減り，Cds の両端の電圧が減り，10kΩの固定抵抗の両端の電圧が高くなって，ain0 端子の電圧は高くなります。つまり ain0 の値は大きくなります。逆に暗くなると Cds の抵抗値が増え，固定抵抗の両端の電圧が低くなります。(1)のプログラムの ain0 の計測部分を 1 行，以下に変えると ain0 そのままの値で明るさが表示されます。

```
  text("brightness = " +  (gainer.analogInput[0]) ,20, 90);
```

(3)　サーミスタで温度を測ろう

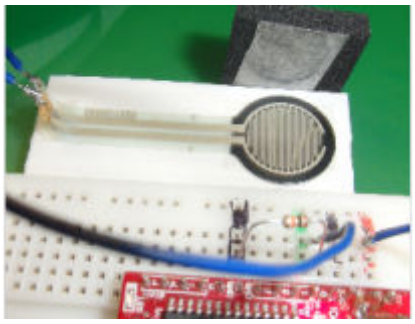


　サーミスタは温度が上がると抵抗値が減少します。サーミスタを(2)の回路の CdS と置き換え，ain0 の計測部分を 1 行，以下に変えるだけです。

問：0℃の氷水で氷水や温度の分かっているお湯などで，値を較正してみましょう。温度計で測定して 0℃のときの ain0 の値が 90，76℃のときの ain0 の値が 241 のとき，以下の○に入る数字や計算式は何でしょう。また，実際にプログラムを変更して測定してみましょう。

```
text("temperature = " +　(gainer.analogInput[0])　* ○ + " ℃",20, 90);
```

(4)　感圧センサで重さを測ろう

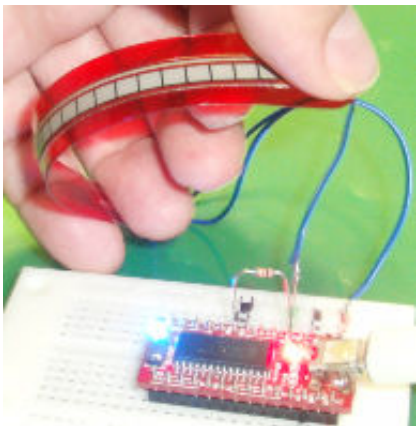


　感圧センサ“FSR402”を(2)の回路の CdS と置き換え，ain0 の計測部分を 1 行，以下に変えるだけです。

　感圧面内に均一に力を加える必要があるため，スポンジを上に乗せて力を分散させます。値を実際の値に近くなるように 2.8 倍し，読みやすいように整数にし，単位 gw を付けました。

```
text("weight = " + int(2.8*(gainer.analogInput[0])) + " gw" ,20, 90);
```

(5)　曲げセンサで曲がり具合を測ろう



　曲げセンサ(4.5 インチ)は，長く平べったいセンサ部分が真っすぐのとき抵抗値は 10kΩですが，曲がるほど抵抗値が増加し，90°のとき約 30~40kΩになります。このセンサを手袋に貼りつけると，指の動きに反応する入力装置ができます。

　曲げセンサを(2)の回路の CdS と置き換え，ain0 の計測部分を 1 行，以下に変えるだけです。

```
text("bending = " +　(gainer.analogInput[0]) ,20, 90);
```

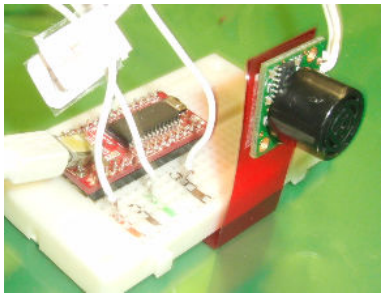
(5)　超音波距離センサで壁との距離を測ろう

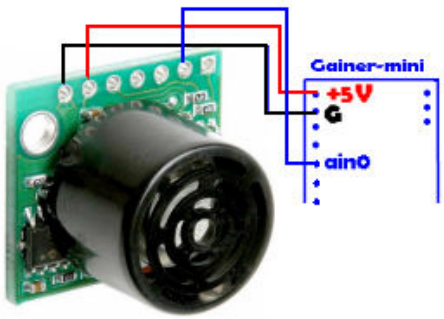
　超音波距離センサ“Maxbotix EZ0”を下図のように配線します。計測値と距離の値は比例します。

　測定する物体の面は，A4 サイズ以上ある方がいいです。

　(1)のプログラムの ain0 の計測部分を 1 行，以下に変えるだけです。

```
text("brightness = " +　(gainer.analogInput[0]) ,20, 90);
```



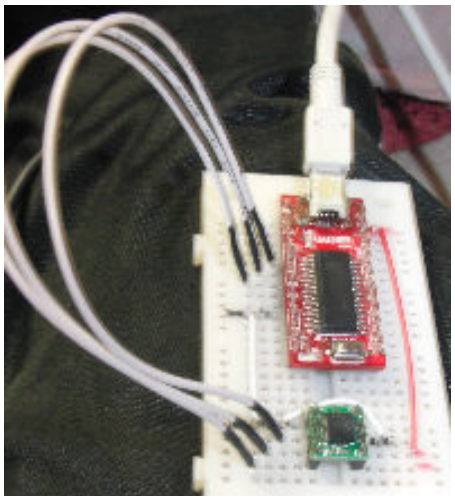


(6)　加速度センサで加速度や傾きを測ろう

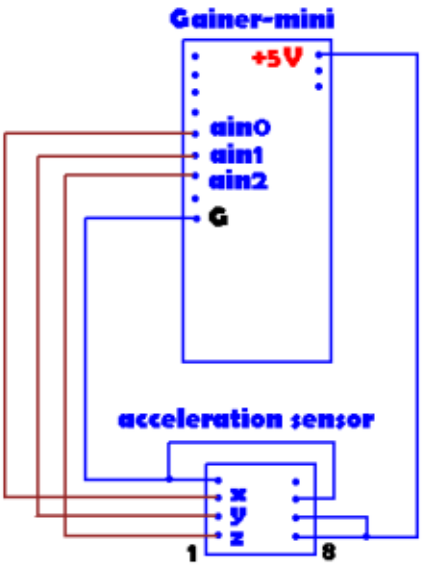
①　3 軸加速度センサ“KXM52-105”を用います。(1)のプログラムを拡張します。入力用ポートを ain0 に加え，ain1，ain2 と 3 つに増やし，それぞれ x,y,z 方向の加速度値に対応させます。画面には x,y,z 方向の加速度値が与えられます。

　地上では重力加速度 g=9.8m/s² なので，写真のように机に置いたときは，通常，y 方向の値が 9.8 になるようにするか，自由落下させたときに 9.8 になるようにするかします。ここではそのままの値を表して

います。ファイル名は“accel_sensor_value_disp”です。



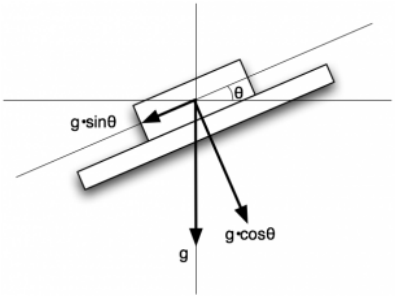
```
/*
 * Accel sensor.
*/
import processing.gainer.*;
Gainer gainer;
void setup() {
 size(250, 250, P3D);
   PFont font;
   font = loadFont("JSSGothic-Md-48.vlw");
   textFont(font, 24);
 gainer = new Gainer(this);
}
void draw(){
 int v;
   // read analog port
   gainer.peekAnalogInput();
   // Display Control
   background(255,255,255);
   fill(255, 0, 0);
   text("Analog[0] = " + gainer.analogInput[0] ,20, 90);
   text("Analog[1] = " + gainer.analogInput[1] ,20, 120);
   text("Analog[2] = " + gainer.analogInput[2] ,20, 150);
}
```



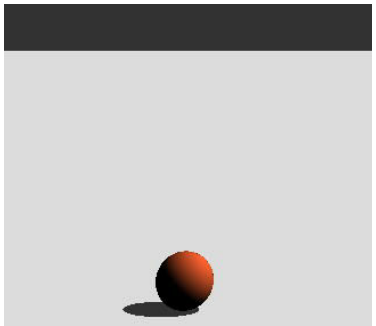
② 傾きへの応用

最近のゲーム機のコントローラやスマートフォンには，加速度センサが内蔵されるようになりました。加速度センサを用いることで，角度や動きを検出することができ，コントローラのダイナミックな動きに応じて画面上の物体を動かしたり，画面の支持方向によって縦横の画面表示を変えたりすることが可能になります。

物体が θ だけ傾いた場合，センサ取り付け方向には $g\sin\theta$ に相当する値が電圧として出力されます。θ が 0 から $\pi/2$ まで変化すると，$\sin\theta$ の値は 0.0 から 1.0 まで変化します。センサを水平に配置した時の値を基準として，垂直に配置した時までの変化幅を求め，逆サイン $\sin^{-1}$ を用いてこの値を角度に変換します。ファイル名は“accel_sensor_Ball_move”です。

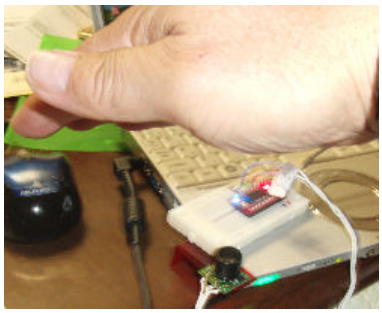



```
/*
 * Ball move with accel sensor.
*/
import processing.gainer.*;
Gainer gainer;
  float xPos,yPos;              ボールの座標用変数
   int xOffset = 0;             オフセット用変数
  int yOffset = 0;
  void setup() {
    size(800,800,P3D);       3D 画面設定
    gainer = new Gainer(this,Gainer.MODE2);
     noStroke();                  ワイヤーフレームなし
  }
  void draw() {
     background(220);              背景色
     fill(50);                     画面上半分の塗色
     rect(0,0,width,height/2);     画面上半分の矩形
     gainer.peekAnalogInput();     モーションセンサーからの読み込み
     if (xOffset == 0)             読み込み値をオフセット値に設定する
       xOffset = gainer.analogInput[1];
```



```
    if (yOffset == 0)
      yOffset = gainer.analogInput[0];
    println("offsetx = " + xOffset + "Analog[0] = " + gainer.analogInput[0]);
    float xSpeed=gainer.analogInput[1] - xOffset;          ボールの速度の計算
    float ySpeed=gainer.analogInput[0] - yOffset;
    xPos+=-xSpeed;                                          ボール移動量の計算
    yPos+=-ySpeed;
    translate(width/2+xPos,height*4/5,-200+yPos);       ボール位置の設定
     fill(50);                                          影の塗色
     ellipse(-25,30,80,80/5);                           影（楕円）の座標と大きさ
     directionalLight(255, 255, 255, -1, 1, 0);     直線光の設定
     fill(255,100,50);                                  ボールの塗色
     sphere(30);                                        ボール描画
 }
```

5　アナログ入力値に応じた音を出そう



Processing で標準で用意されている **Minim** というライブラリを利用して音を出します。ここでは曲げセンサを接続してテルミンのような音を出して演奏します。曲げセンサは曲げ具合を確認しながら演奏できるので，感覚的に演奏しやすいです。

また，右図のように超音波距離センサを用い，センサの上で手のひらを近づけたり，遠ざけたりして音の高低を変えます。まさにテルミンの感じですね。

光センサなど，いろんなセンサを工夫して，オリジナル楽器を作ることができます。

では，「曲げセンサ」を使用したときのプログラムを見てみましょう。ファイル名は“sound_ain0”です。

```
import ddf.minim.*;                                     ライブラリのインポート
import ddf.minim.signals.*;
import processing.gainer.*;
Minim minim;                                            Minim 本体のオブジェクト変数
AudioOutput out;
SineWave sine;
Gainer gainer;
void setup() {
  minim = new Minim(this);                              Minim の本体を生成
  out = minim.getLineOut(Minim.STEREO);
  sine = new SineWave(440, 0.5, out.sampleRate());
  sine.portamento(200);
  out.addSignal(sine);
  gainer = new Gainer(this);
  gainer.beginAnalogInput();
}
void draw() {
  int analogInput = (gainer.analogInput[0] - 80) * 2;       曲げセンサ用の計算式です
  println(analogInput);                                 ain0 の値を表示
  background(204);
  ellipse(width/2, height/2, analogInput, analogInput);     aino の値による半径の楕円を表示
  float freq = map(analogInput, 0, 180, 0, 3600);
  sine.setFreq(freq);
}
void stop() {
  out.close();
  minim.stop();
  super.stop();
}
```

「超音波センサ」を使うときは，計算式を　**int analogInput = (5 + gainer.analogInput[0]) * 4**; に変更します。

６　あるタイミングで動作させよう

(1)　キー入力で Gainer-mini 上の LED を制御しよう

プログラムを実行しているときに，あるタイミングで測定したり制御したりしたいとき，あるキーボードのキーをタイプするのが確実です。マウスボタンだと，画面のどこにあるかで，別の動作が実行されるときがあるからです。では，Gainer-mini 上の LED を"p"，"s"のキーで点，滅させてみましょう。ファイル名は“key_led”です。

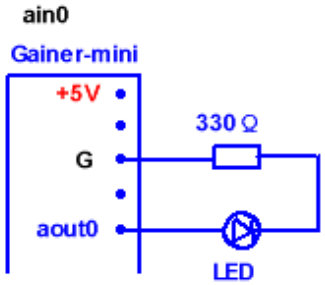
```
/*
  when keyPressed on Gainer.
*/
import processing.gainer.*;
Gainer gainer;
void setup(){
  gainer = new Gainer(this);
}
void draw(){
  background(0);
}
void keyPressed() {
    if(key == 'p' ) {
  gainer.turnOnLED();
    }
    if(key == 's' ) {
  gainer.turnOffLED();
}
```

p のキーが押されたら
Gainer 上の青色 LED を点灯

s のキーが押されたら
Gainer 上の青色 LED を消灯

(2)　１秒おきに動作させよう

秒を変数として読み込む関数 second()を用いて，偶数秒のときに点灯（255(5V)を aout0 に出力），奇数秒のときに消灯する 1 秒おきの点滅を実現させます。回路は 3 の回路です。draw 関数の部分を以下のように変更します。ファイル名は“onesec_on_off_led”です。




```
void draw(){
   int s = second();
   if (s % 2 !=0){
     fill(255,0,0);
     gainer.analogOutput(0,255);
   }else{
     fill(0,0,255);
      gainer.analogOutput(0,0);
        }
   rect(20,20,160,160);
}
```

s が偶数のとき，2 で割った余りが 0 になります。**s % 2** は剰余（割り算のあまり）の計算です。,

(3)　タイマーで数秒おきに LED を点滅させよう

繰り返しループを用いて時間をかせぎます。ここでは，約 1 秒おきに点滅させます。回路は 3 の回路です。ファイル名は“Timer_LED”です。

```
/*
 Timer test    LED
*/
import processing.gainer.*;
```

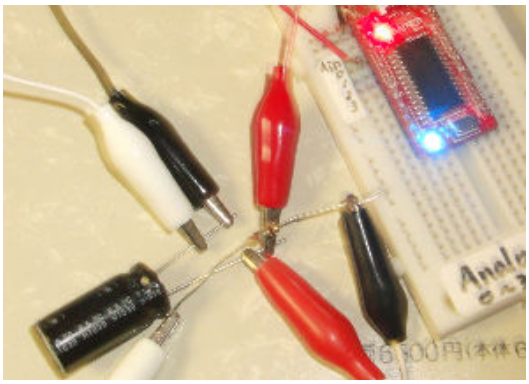
```
Gainer gainer;
void setup(){
 gainer = new Gainer(this);
 }
void draw(){
   gainer.analogOutput(0,255);
   timer();                                  新しい関数「timer」を呼び出します
   gainer.analogOutput(0,0);
    timer();
 }
void timer(){                                関数「Timer」です。約１秒間時間をかせぎます
 for(int i=0 ; i < 50000 ; i++){             次の｛｝内のループ処理を 50001 回繰り返します
   for(int j=0 ; j < 10000 ; j++){}}         空（から）の処理を 10001 回繰り返します
}
```

７　計測データをエクセルファイルにしよう

(1)　１秒おきに電圧を測定してグラフにしよう

コンデンサの放電曲線のように，時間とともに変化していく様子を調べることができます。回路は４の(1)を使います。どれかキーを打つと終了し，１秒ごとの電圧の変化がテキストデータとして日付とともに保存されます。エクセルでそのファイルを開くときに，テキストファイルウィザードで，区切るデータの形式を「スペースによって右まはた左に揃えられた固定長フィールドデータ」を選択します。ファイル名は“voltage1_second_file”です。

```
/**
 * Voltage.   data logger
**/
import processing.gainer.*;
Gainer gainer;
 PrintWriter logFile;
 void setup()
{
 size(250, 250);
  PFont font;
   font = loadFont("JSSGothic-Md-48.vlw");
   textFont(font, 24);
   logFile = createWriter("log-data.txt");
 gainer = new Gainer(this,Gainer.MODE2);
 frameRate(60);
}
 int ss = 0;
   void draw(){
     int v;
      gainer.peekAnalogInput();
         background(255,255,255);                          //Display Control
     int s = second();
     fill(255, 0, 0);
     if (s != ss){                                         時刻が１秒たった時の処理
     text("Voltage_anao = " + gainer.analogInput[1] ,20, 120);   時刻と ain0 の値を記録する。
               logFile.println(year() + "/" + month() + "/" + day() + " "
                   + hour() + ":" + minute() + ":" + second() + " "
                   + gainer.analogInput[1]) ;
                 ss = s;                                   秒の値を変数 ss に退避
             }
           }
      void keyPressed()  {
```



コンデンサは 10μF,放電用の抵抗は１MΩ程度を選ぶとよい。5V 以上に充電しないこと。

```
  logFile.flush();                                  // Writes the remaining data to the file
  logFile.close();                                   // Finishes the file
  exit();                                            // Stops the program
}
```

上の表は log-data ファイル。右図はこのファイルをエクセルに読み込んで描かせたグラフである。線の太さは誤差を表している。

(2)　落下時の加速度の様子をグラフにしよう

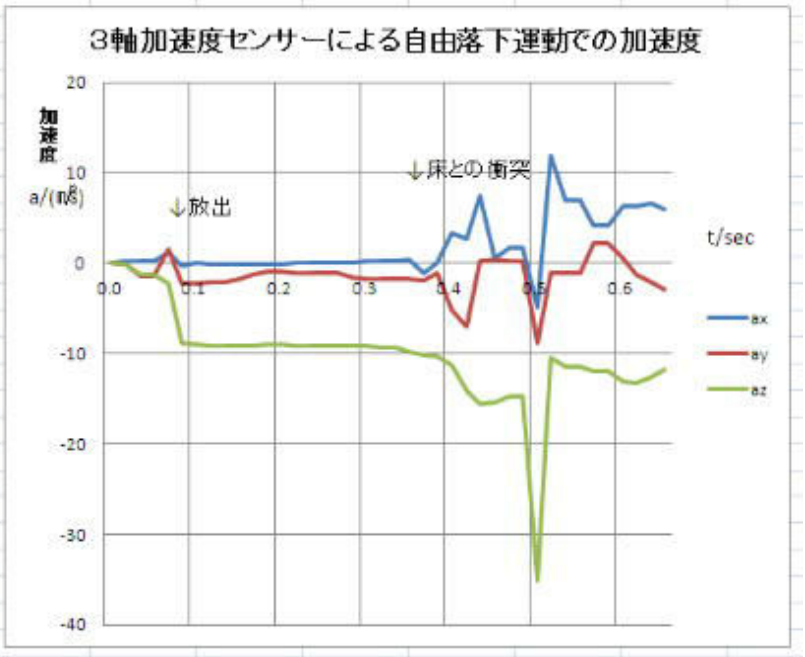
　４の(6)の回路を発泡ポリスチレンやクッション材に包んで落下させると，ｘ，ｙ，ｚ方向の加速度の変化の様子がわかる。運動方程式より，力と加速度は向きが等しく比例関係にあるので，加速度の大きい方向に力も大きくなっています。ファイル名は“accel_sensor_file”です。

```
/*
 * Accel sensor.   data logger
 *
*/
import processing.gainer.*;
Gainer gainer;
 PrintWriter logFile;

void setup()
{
 size(250, 250);
 noStroke();
 colorMode(RGB, 1);
   PFont font;
   font = loadFont("JSSGothic-Md-48.vlw");
   textFont(font, 24);
   logFile = createWriter("log-data.txt");
 gainer = new Gainer(this,Gainer.MODE2);
```

```
 frameRate(60);
}

  void draw(){
    int v;
     gainer.peekAnalogInput();                                  // read analog port
     background(255,255,255);                                   // Display Control
    fill(255, 0, 0);
    text("Analog[0] = " + gainer.analogInput[0] ,20, 90);
    text("Analog[1] = " + gainer.analogInput[1] ,20, 120);
    text("Analog[2] = " + gainer.analogInput[2] ,20, 150);
        logFile.println(year() + "/" + month() + "/" + day() + " "
                + hour() + ":" + minute() + ":" + second() + " "
                + gainer.analogInput[0] + " "
                + gainer.analogInput[1] + " "
                + gainer.analogInput[2]);
    }
   void keyPressed()   {
      logFile.flush();                                          // Writes the remaining data to the file
      logFile.close();                                          // Finishes the file
      exit();                                                   // Stops the program
   }
```

加速度センサを y 軸方向に垂直な面に設置し，水平に保って自由落下させた結果です。y 軸方向が重力加速度の値にほぼなるように，換算はエクセルの表の中で行います。

	A	B	C	D	E	F	G
4	ax	ay	az	ax	ay	az	t
5	149	149	216	0	0	0	0.00
6	150	148	215	0.16	-0.2	-0.2	0.02
7	150	140	208	0.16	-1.5	-1.3	0.03
8	150	140	208	0.16	-1.5	-1.3	0.05
9	156	158	202	1.15	1.48	-2.3	0.07
10	147	135	162	-0.8	-2.3	-8.9	0.08
11	149	135	161	0	-2.3	-9	0.10
12	148	136	160	-0.2	-2.1	-9.2	0.12
13	148	136	160	-0.2	-2.1	-9.2	0.13
14	148	138	160	-0.2	-1.8	-9.2	0.15
15	148	141	160	-0.2	-1.3	-9.2	0.17
16	148	143	161	-0.2	-1	-9	0.18
17	148	143	161	-0.2	-1	-9	0.20
18	149	142	160	0	-1.1	-9.2	0.22
19	149	142	160	0	-1.1	-9.2	0.23
20	149	142	160	0	-1.1	-9.2	0.25
21	149	142	160	0	-1.1	-9.2	0.27
22	149	139	160	0	-1.6	-9.2	0.28
23	150	138	160	0.16	-1.8	-9.2	0.30
24	150	138	159	0.16	-1.8	-9.3	0.32
25	150	138	159	0.16	-1.8	-9.3	0.33
26	151	138	156	0.33	-1.8	-9.8	0.35
27	142	137	154	-1.1	-2	-10	0.37
28	149	142	153	0	-1.1	-10	0.38
29	169	117	147	3.28	-5.2	-11	0.40
30	165	106	130	2.62	-7.1	-14	0.42
31	194	150	121	7.38	0.16	-16	0.43

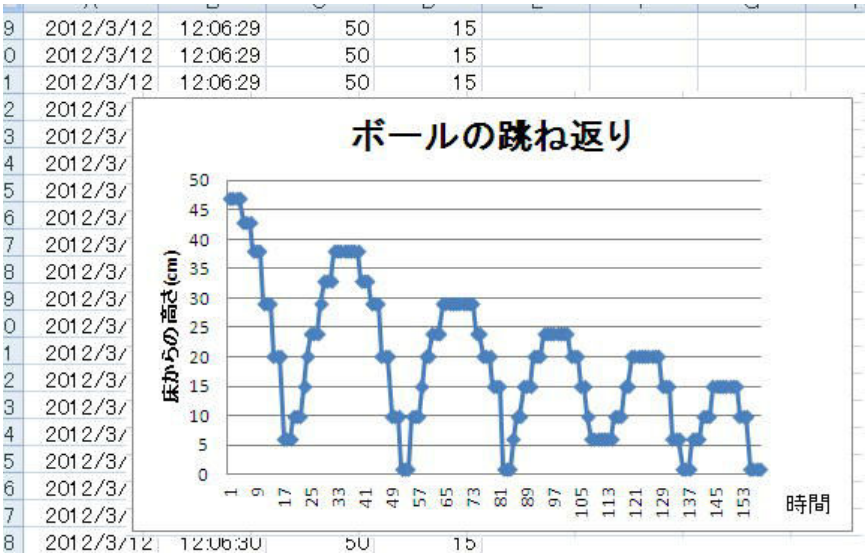


(3)　距離センサを用いて，ボールの跳ね返り係数を求めよう

大きなボールを床に反射させ，落下距離から，ボールの床からの高さを測ることで高さの比の平方根から反発係数（跳ね返り係数）を算出することができます。プログラムは４の(6)の距離の計測に，ファ

イルへの書き出し機能をつけたものです。ファイル名は“distance_sencer_file”です。1m 程度の一定の高さに床に向けて距離センサをセットし，ボールを近づけた状態から床に落としてバウンドさせます。エクセルでグラフ化して反発係数を求めます。下の図の例では始めの高さと最初の跳ね返り後の高さはそれぞれ 47cm，39cm なので，跳ね返り係数 e=$\sqrt{39/47}$ =0.9 となります。

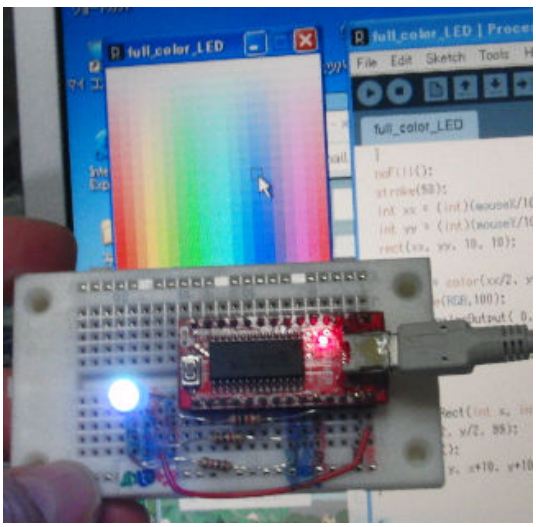
```
/*
 * AnalogInput0 Censer:LV-MaxSoner-EZ0 + file
*/
import processing.gainer.*;
Gainer gainer;
 PrintWriter logFile;
void setup() {
 size(250, 250, P3D);
   PFont font;
   font =
loadFont("JSSGothic-Md-48.vlw"
);
   textFont(font, 24);
   logFile =
createWriter("log-data.txt");
 gainer = new Gainer(this);
 frameRate(60);
}
void draw() {
   // read analog port
   gainer.peekAnalogInput();
   // Display Control
   background(255,255,255);
   fill(255, 0, 0);
   text("Distance[cm] = " +   (gainer.analogInput[0])*400/87 ,20,
90);
   logFile.println(year() + "/" + month() + "/" + day() + " "
                   + hour() + ":" + minute() + ":" + second() + " "
                   + (gainer.analogInput[0]*400/87));
  }
 void keyPressed()  {
     logFile.flush(); // Writes the remaining data to the file
     logFile.close(); // Finishes the file
     exit(); // Stops the program
  }
```



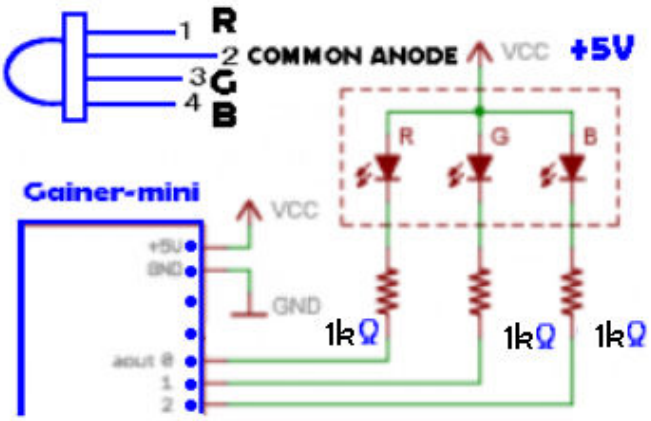
8　アナログ出力でいろんなオブジェクトを制御しよう

(1)　フルカラーLED でいろんな色を作ろう

　1 つのケースの中に赤，青，緑の RGB 3 色の LED が内蔵されたフルカラーLED“ RT5-5818RGBW-B ”は，３本の各色を指定する脚と，共通の＋（ANODE）または－（KATHODE）用の脚があります。ここで使うのは＋極が共通の LED です。図のように回路を構成します。ファイル名は“full_color_LED”です。“run”すると，写真のように，カラーパレットが表示されます。マウスカーソルを持っていったところの色に対応して，LED の色が変化します。



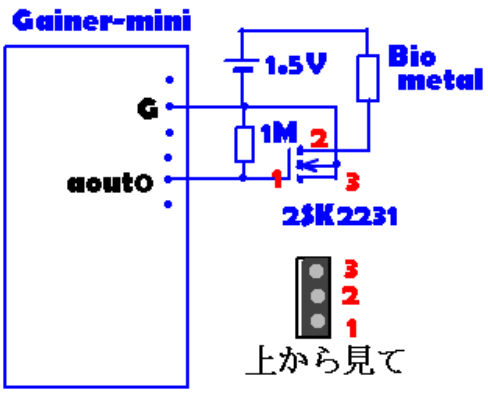
```
/*
* full_color_led
*/
import processing.gainer.*;
Gainer gainer;
void setup() {
 size(200, 200);
 colorMode( HSB, 100);
 gainer = new Gainer(this);
 }
void draw() {
 colorMode(HSB,100);
 for(int x = 0; x < 200; x += 10){
    for(int y = 0; y < 200; y += 10){
       colorRect( x, y);
    }
 }
 noFill();
 stroke(50);
 int xx = (int)(mouseX/10)*10;
 int yy = (int)(mouseY/10)*10;
 rect(xx, yy, 10, 10);
 color c = color(xx/2, yy/2, 99);
 colorMode(RGB,100);
 gainer.analogOutput( 0, (2 * (99 -   (int)red(c))) );
 gainer.analogOutput( 1, (2 * (99 - (int)blue(c))) );
 gainer.analogOutput( 2, (2 * (99 - (int)green(c))) );
}
void colorRect(int x, int y) {
 fill( x/2, y/2, 99);
 noStroke();
 rect( x, y, x+10, y+10);
}
```

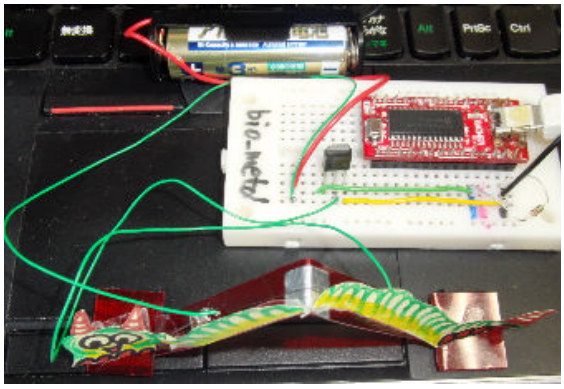



RGB は赤（Red），緑（Green），青（Blue）の三つの原色を混ぜて幅広い色を再現する加法混色の一種です。均等に混ぜれば，赤と黄色で緑，緑と青でシアン，赤と青で紫が作られます。中間色はそれらの配合具合でできます。
一方 HSB では，色の三属性である色相(Hue)，彩度(Saturation)，明度(Brightness もしくは Value)を用いて色を指定します。

(2)　バイオメタル尺取虫を動かそう

7 の(2)のプログラムがそのまま使えます。１秒おきにバイオメタルの電源が ON，OFF され，ON のときに電流が流れたバイオメタルはジュール熱で発熱して縮み，尺取虫は脚を広げます。OFF で脚を閉じます。両足には同じ方向に毛並みの揃った洋服ブラシの布が貼りつけてあり，進行方向前後の摩擦の差で進みます。ファイル名は 6 の(2)の onesec_on_off_LED です。LED の代わりにバイオメタルに通電します。

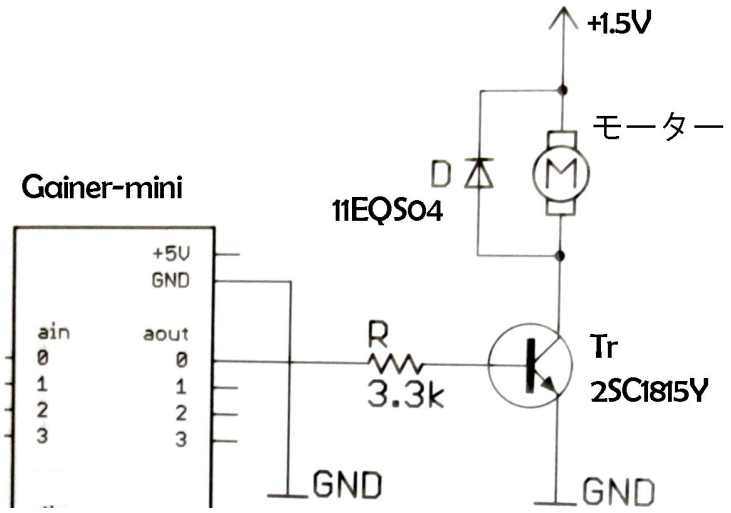


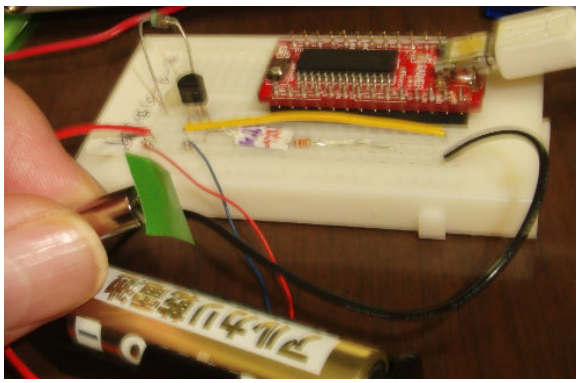


(4)　直流モーターを制御しよう

モーターの制御は，車やロボットの制御につながります。モーターを動かすときは，aout0 に 5V を出力し，トランジスタ 2SC1815 にベース電流を流し，コレクタ電流を増やして外部電源の電池 1.5V につないだ小型モーターを動かします。モーターと並列に接続したダイオードは，トランジスタが OFF のときに発電機となって逆向きの電流がトランジスタに流れ，トランジスタを破壊するのを防ぎます。下の回路図のように接続します。モーターの種類によって，モーター用電源の電圧を決めます。ここでは 1.5V 用のモーターを使っています。

プログラムは 6 の(3)の“Timer_LED”を用います。1 秒ごとにモーターが ON, OFF を繰り返します。



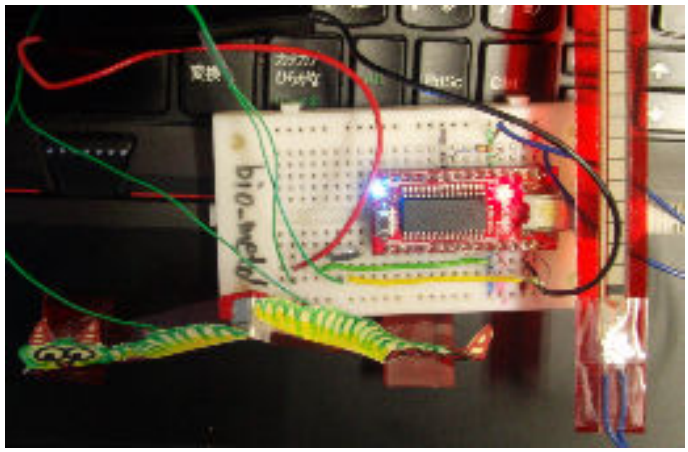


9　アナログ入力した値で，アナログ出力してみよう

(1)　曲げセンサで尺取虫の速さを制御しよう

7の(2)の①のプログラムの draw 関数，timer 関数部分を以下のように変更します。

```
void draw(){
     gainer.peekAnalogInput();
      int s = gainer.analogInput[0];
  曲げセンサーの値を ain0 に読み込みます
    int n = (s / 40) * 10000;
  入力値に比例して，繰り返し数を計算します
    gainer.analogOutput(0,255);
  バイオメタルに電流を流して縮めます
    timer(n);
    gainer.analogOutput(0,0);
  バイオメタルを伸ばします
    timer(n);
  }

void timer(int   n){
 for(int i=0 ; i < n ; i++){
   for(int j=0 ; j < 20000 ; j++){}}
   }
```



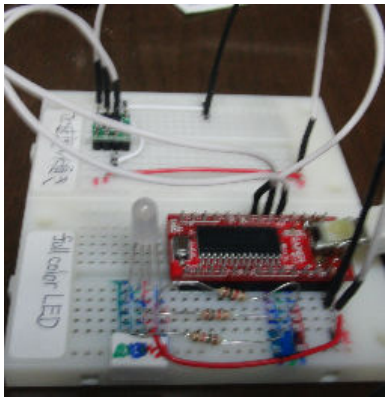
計算した n の値を受けて，繰り返します

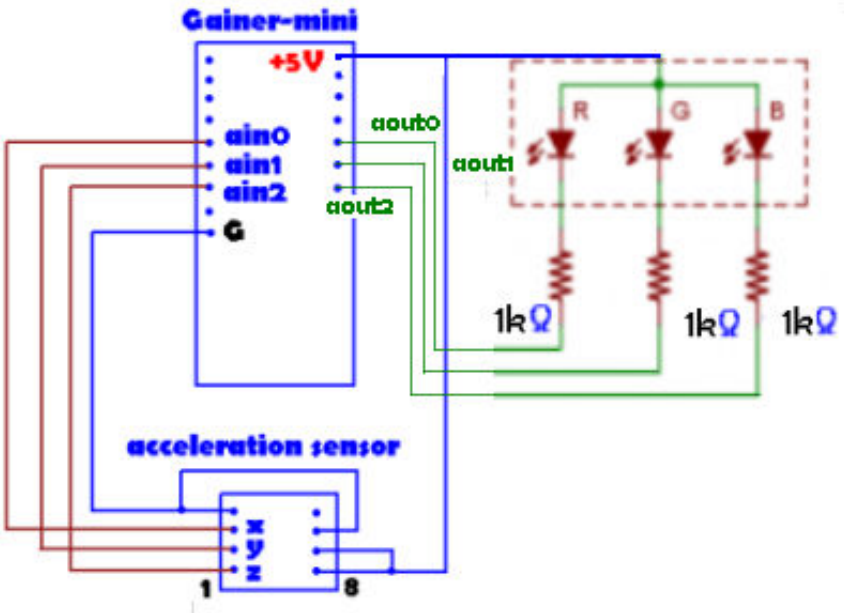
曲げセンサと 10kΩの抵抗の位置は，4 の(1)と同じです。センサを曲げるほど入力値が小さくなり，バイオメタルの伸び縮みの周期が短くなります。

(2)　加速度センサの傾きで，フルカラーLED の色を制御しよう

　加速度度センサの x，y，z の値に応じて R，G，B の値を変化させ，フルカラーLED にいろんな色を表示させます。

　回路は入力側に 4 の(6)の①，出力側に 8 の(2)を使います。ファイル名は
“full_color_accelerator”です。






```
/*
* full_color_led_acceleration
*/
import processing.gainer.*;
Gainer gainer;
void setup() {
  gainer = new Gainer(this);
 }
void draw() {
  gainer.peekAnalogInput();
  colorMode(RGB,255);
  gainer.analogOutput( 0, (gainer.analogInput[0]) );
  gainer.analogOutput( 1, (gainer.analogInput[1]) );
  gainer.analogOutput( 2, (gainer.analogInput[2]) );
 }
```

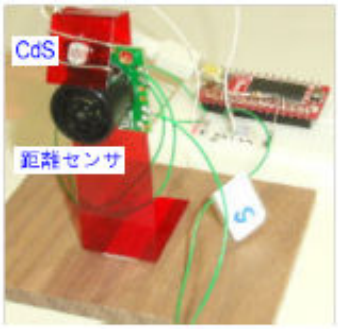
アナログ入力を可能にする
色は 255 段階
加速度の x， y， z の値をそのまま R，G，B の値にする

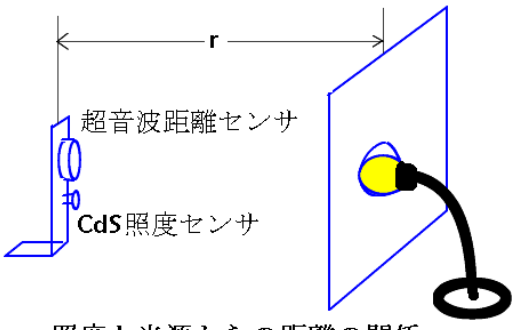
１０　２つの物理量の間のグラフを描こう

・　CdS と LED を用いて，暗くなるほど LED が明るく点灯するようにしよう

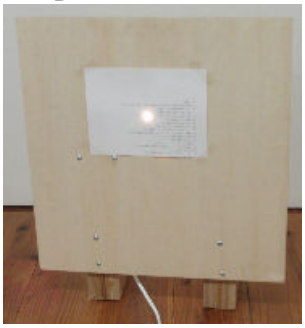
光源の電球と反射板

　明るさを測る CdS と距離センサを組み合わせ，エクセルの散布図のグラフを用いて明るさが光源からの距離の二乗に反比例することを

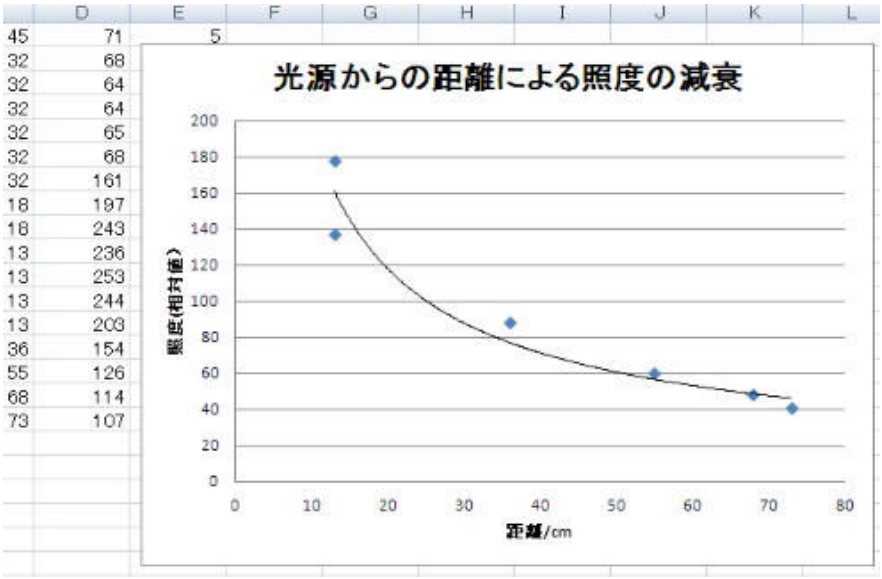


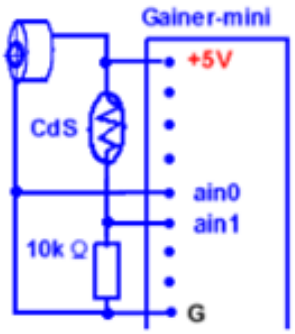


照度と光源からの距離の関係



```
/*
 *
distance_brightness_second_file
*/
 import processing.gainer.*;
Gainer gainer;
 PrintWriter logFile;

void setup()
{
   size(250, 250);
      PFont font;
     font =
loadFont("JSSGothic-Md-48.vlw")
;
     textFont(font, 24);
     logFile =
createWriter("log-data.txt");
   gainer = new
Gainer(this,Gainer.MODE2);
   frameRate(60);
}
   int ss = 0;
    void draw(){
       int v;
       // read analog port
       gainer.peekAnalogInput();
       // Display Control
       background(255,255,255);
       int s = second();
       fill(255, 0, 0);
       if (s != ss){
       text("Distance = " + gainer.analogInput[0]*400/87 ,20, 120);
       text("Brightness = " + gainer.analogInput[1] ,20, 120);
                  logFile.println(year() + "/" + month() + "/" + day() + " "
                      + hour() + ":" + minute() + ":" + second() + " "
                      + (gainer.analogInput[0]*400/87) + " " + (gainer.analogInput[1])) ;
            ss = s;
                }
             }
        void keyPressed()   {
       logFile.flush(); // Writes the remaining data to the file
       logFile.close(); // Finishes the file
       exit(); // Stops the program
    }
```





１１　発展

どういう計測・制御を応用した実験ができるか考えてみよう

参考文献

・+GAINER—PHYSICAL COMPUTING WITH GAINER，くるくる研究室，九天社

・Built with Processing[Ver. 1.x 対応版] -デザイン/アートのためのプログラミング入門 ， 田中孝太郎，前川峻志，ビー・エヌ・エヌ新社

参考 URL

・㈱アールティー　「Gainer-mini」：http://www.gainer-mini.jp/