第 2 講 プログラムから LED を制御する

Processing を使って簡単な絵を書く。Arduino を組み合わせる。

1 Processing とは

Processing[1] は、Casey Reas と Benjamin Fry によるオープンソースプロジェクトであり、かつては MIT メディアラボで開発されていた。電子アートとビジュアルデザインのためのプログラミング言語であり、統合開発環境 (IDE) である。 視覚的なフィードバックが即座に得られるため、初心者がプログラミングを学習するのに適しており、電子スケッチブックの基盤としても利用できる。Java を単純化し、グラフィック機能に特化した言語といえる。

-wikipedia-

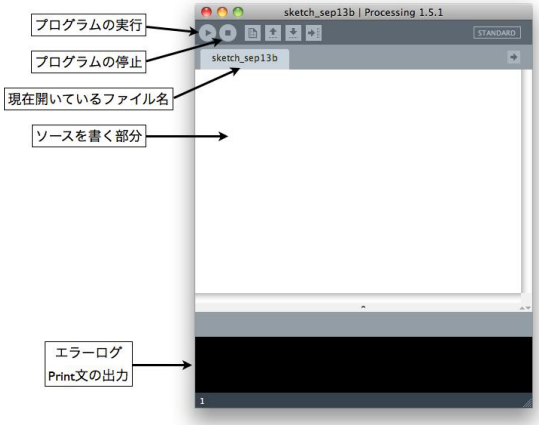

図 1: Processing の IDE (統合開発環境)

2 Processing の基礎

以下で、Processing による基本的なプログラミングの説明を行います。なお、メニューバーの「Help → Reference」でインターネットブラウザが起動し、より詳しい説明が確認できますので、そちらも参考にしてください。ただし、英語表記です。Processing にはあらかじめサンプルコード (お手本になるプログラム) が、たくさん用意されているので、それを動かしてみてどんなことができるかを見ておくのも良いかもしれません。サンプルコードは、メニューバーの「File → Example」から見ることができます。

初期化とループ処理

プログラムの一番最初に行われる処理のことを「初期化」という。Processing では setup という関数で初期化が行われる。

Processing のプログラムで、メインの処理を行う。自分で終わることを宣言する、stop ボタンを押して中断する、エラーになるかしない限り、プログラムの実行は終わらずに、Processing は draw 関数を実行し続ける。

[1] http://processing.org/

```
void setup(){
  // 初期化の処理をここに書く
  // プログラム開始時に1度だけ実行される
}

void draw(){
  // 繰り返したい処理をここに書く
  // setup()が実行された後にプログラムが終了するまで繰り返される
}
```

- **ウインドウサイズの指定**

  size(w, h); w = ウィンドウの幅 (px) h = ウィンドウの高さ (px)

  setup() の中でしか使ってはいけません。

- **カラーモードの設定**

  colorMode(RGB, 256); RGB = 赤・緑・青の組み合わせ (Red, Green, Blue) 256 = 色の範囲を 256 階調 (段階) で指定

- **図形の原点指定**

  rectMode(CORNER); CORNER = 四角形の左上が原点 (デフォルト) CENTER = 四角形の中心が原点※似たような設定に ellipseMode(円の原点指定), imageMode(画像の原点指定) の設定がある。

- **線を書く**

  line(x1, y1, x2, y2); x1 = 始点の x 座標 y1 = 始点の y 座標 x2 = 終点の x 座標 y2 = 終点の y 座標

- **三角形を書く**

  triangle(x1, y1, x2, y2, x3, y3); x1 = 角 1 の x 座標 y1 = 角 1 の y 座標 x2 = 角 2 の x 座標 y2 = 角 2 の y 座標 x3 = 角 3 の x 座標 y3 = 角 3 の y 座標

- **四角形を書く**

  rect(x, y, sx, sy); x = 原点の x 座標 y = 原点の y 座標 sx = 四角形の幅 sy = 四角形の高さ

- **円を書く**

  ellipse(x, y, sx, sy); x = 原点の x 座標 y = 原点の y 座標 sx = 円の幅 sy = 円の高さ

- **図形に色をつける**

  fill(r, g, b); r = Red の値 g = Green の値 b = Blue の値 fill(value); value = r, g, b に同じ値を代入する

- **色を消す**

  noFill();

- **縁取り線に色をつける**

  stroke(r, g, b); stroke(value); 色の指定方法は fill と同じ

- **縁取り線の幅を調節する**

  strokeWeight(value); value = 線の太さ (px) の指定

- **縁取り線を消す**

  noStroke();

- **図形の縁をなめらかにする**

  smooth();

- **ウィンドウの背景色を指定する**

  background(r, g, b); background(value); 色の指定方法は fill と同じ

TRY1

Processing で 本日行う実習のベースとなるプログラムを作成する。

1. ウィンドウサイズが 幅が 300 ピクセル、高さが 300 ピクセル
2. カラーモードは RGB の 256 階調
3. ウィンドウの背景色は黒
4. タテが 200 ピクセル、ヨコが 200 ピクセル の 白い円が、ウインドウの中心に表示される

(時間のある人は、色々と改造してみましょう。)

```
void setup(){
  size(??? , ???);
  colorMode(RGB, ???);
  ellipseMode(CENTER);
  background(???);
}

void draw(){
  background(???, ???, ???, ???);
}
```

ヒント: ??? には数字が入ります

Processing で マウスのクリックを検出する

Processing では、「mousePressed」という命令文を用いることで、マウスのクリック動作を簡単に検出することができる。

```
if (mousePressed == true){
  // マウスボタンを押した時の処理
} else {
  // マウスボタンを離した時の処理
}
```

TRY2

TRY1 で作ったプログラムを改造し、マウスが押されたときに PC に図形が表示されるようにする。Gainer mini を Processing で 使えるようにするライブラリのインポート Processing で Gainer mini を利用するには、ライブラリと呼ばれるプログラムのパーツを用いて Processing の機能を拡張する必要がある。Processing のメニューバーの中の Sketch → Import Library → gainer を選択すると、今書いているプログラムの一番初めに、ライブラリを読み込むための命令文 import processing.gainer.* が書き込まれる。Gainer オブジェクトの作成上で挿入されたインポート文のすぐ下に Gainer gainer; void setup() 内に、gainer = new Gainer(this); Gainer mini の ボタンを利用する if (gainer.buttonPressed) ¡¡ Gainer mini のボタンが押された時の処理 ¿¿

TRY3

TRY2 で作ったプログラムを改造し、Gainer mini のボタンが押された時に、PC に図形が表示されるようにする。

Gainer mini の digital output を利用する digital output (ON, OFF の切り替え) digital output を利用することで、Processing 上から、Gainer mini に流れる電流の ON 、OFF を制御することができる。ON にしたい場合 gainer.setHigh(0); (※数字部分は dout のピン番号) OFF にしたい場合 gainer.setLow(0); (※数字部分は dout のピン番号)

TRY4

TRY3 で作ったプログラムに命令文を追加して、Gainer mini のボタンが押された時に、dout 0 に接続した LED が光るようにする。