第 1 講 基本的なプログラミングと電子回路の作成

Processing を使って絵を書く。基本的な子電回路を組む。

1　下準備

ノート PC の準備

本実習では、各個人に 1 台のノート PC を割り当てます。次回の授業からは開始までに、あらかじめロッカー内から自分が使用するノート PC を取り出して準備を行なってください。授業終了時には、ノート PC を収められていた箱に収納し、ロッカー内の指定位置まで返却してください。

このノート PC は、他の実習授業でも使用しますので、まず最初に各個人のアカウントを作成します。授業内の作業は、各個人アカウントで行うようにしてください。指示にしたがって、共有アカウントでログインし、個人用アカウントの作成、パスワードの設定を行なってください。

実習で用いる部品

図 1: Arduino UNO

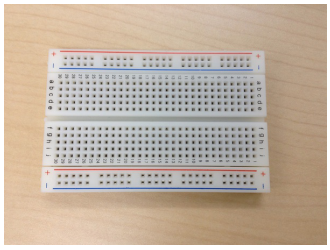

図 2: ブレッドボード

図 3: ジャンプワイヤ

図 4: 【図を差し替える】抵抗、スイッチ、センサ etc.

注意!

Arduino などの電子部品は壊れやすいので取り扱いに注意してください。

- 静電気 (Arduino が壊れてしまいます)
- 電子部品のピンは曲がりやすいので無理に差し込まないように注意! (壊れます)
- PC に刺したまま配線をいじらない (最悪、PC ごと壊れます …)

2 Processing の基礎

本実習では Processing[1] というプログラミング言語を用います。Processing は、Java を単純化し、グラフィック機能に特化した言語です。

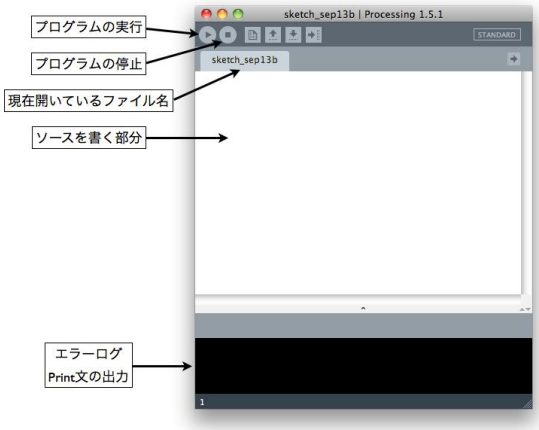

図 5: Processing の IDE (統合開発環境)

基本的な図形を描く

- 四角形を書く: rect(x, y, w, h);

  x = x 座標、y = y 座標、w = 四角形の幅、h = 四角形の高さ

- 円を書く: ellipse(x, y, w, h);

- 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

図形の色を変える

Processing で色を指定する最も簡単な方法は、Red、Green、Blue をそれぞれ 256 段階で指定する方法です。

```
// 四角形を赤で表示する
fill(255, 0, 0);          // 赤は r = 255, g = 0, b = 0
rect(10, 10, 100, 50);
```

グレースケールで指定する方法もあります。

```
fill(0);                  // 0（黒）から 255（白）で指定
```

白から黒の色で指定したいときはこちらの方が楽です。

- 図形の色を変える: fill(r, g, b);

- 図形の色を消す: noFill();

[1] http://processing.org

- 線に色をつける: stroke(r, g, b);
- 線の色を消す: noStroke();
- ウィンドウの背景色を変える: background(r, g, b);

初期化とループ処理

Processing では、プログラムを実行時に setup 関数が 1 度実行され (初期化) 、その後 draw 関数 が繰り返される (ループ処理)。setup と draw を用いるとアニメーションなどの動的な表現ができる。

```
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
  // size(w, h); などはここへ
}

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
}
```

TRY1: Processing でベースとなるプログラムを作成する

1. ウィンドウサイズを幅 300 pixel、高さ 300 pixel に
2. ウィンドウの背景色を黒に
3. 幅 100 pixel、幅 100 pixel の円を青色で表示

```
void setup() {
  size(???, ???);
}

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

  fill(???, ???, ???);
  ellipse(???, ???, ??? ???);
}
```

TRY2: マウスボタンがクリックされたら図形を表示する

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ウィンドウに図形が表示されるようにする。

Processing では、あらかじめ用意されている「mousePressed」という変数を用いることで、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) {
  // マウスボタンを押したときの処理
} else {
  // マウスボタンを押していないときの処理
}
```

TRY3: マウスポインタで図形を操作する

TRY2 で作ったプログラムを改造し、マウスポインタの位置に合わせて図形が表示されるようにする。

Processing では、あらかじめ用意されている「mouseX」「mouseY」という変数を用いることで、マウスポインタの x 座標と y 座標を取得できる。

```
void setup() {
  size(300, 300);
}

void draw() {
  ellipse(mouseX, mouseY, 32, 32);
}
```

3　電子回路の基礎

電気、電流、抵抗のおさらい

よく、電気の流れは水の流れに例えられます。厳密にいえばさまざまな違いはありますが、その性質を大まかにとらえるには、水の流れに例えて理解するのは有効な方法です。

- 電圧

  電圧は 2 点間の高度 (電位) の違いを表す用語です。水は高度の高いところから低いところに向けて流れますが、電気も電位の高いところから低いところに向けて流れます。水の場合には、それぞれの地点の高さを比較するのに、海抜などを基準として用います。電気の場合には、グランド (GND) を基準として比較します。電子回路ではよく「グランドを接続する」ということが行われます。これは、基準であるグランドを共通にしないと、回路の部分ごとの電圧の基準が共通にならず、意図した通りに電気が流れてくれないためです。電圧の単位はボルト (V) です。数字が大きければ大きいほど、電圧が高いことを示します。

- 電流

  電流は、水の流れが水流であるのと同じように電気の流れです。電流は、電圧の高いところから低いところに向けて流れます。水流も多い場合少ない場合がありますが、電流も多い場合や少ない場合があります。電流の単位はアンペア (A) です。数字が大きければ大きいほど、多くの電気が流れることを意味します。

- 抵抗

  水の場合にも、何も障害物がない場合と、くねくね曲がって流れにくくしている場合では、水の流れにくさが異なります。同様に、電気の場合にも電流が流れやすい場合と流れにくい

場合があります。この電流の流れにくさを表すのが抵抗です。抵抗の単位はオーム (Ω) です。数字が大きければ大きいほど、電流が流れにくいことを示します。

補助単位

電圧、電流、抵抗の単位はそれぞれボルト (V)、アンペア、オームですが、実際にはこれに接頭辞がついた補助単位が用いられる場合が多くあります。以下は、登場する補助単位の例です。

- 1,000 倍を表す単位がキロ (例: 10 k Ω)
- 1,000,000 倍を表す単位がメガ (例: 1 M Ω)
- 1/1,000 を表す単位がミリ (例: 10 mA)

オームの法則

電子部品に電圧をかけすぎたり、電流を流しすぎると壊れてしまう。そのために、電源側に抵抗器を配置し、電流量を調節する必要がある。

オームの法則は、

$$V(\text{電圧}) = I(\text{電流}) \times R(\text{抵抗}) \tag{1}$$

R (抵抗) を求めるために指揮を変形すると、

$$R = \frac{(\text{電源電圧} - LED \text{にかかる電圧})}{LED \text{に流したい電圧}} \tag{2}$$

スイッチによって **LED** を点灯させる回路を作る

スイッチを押すと LED が点灯する、という回路を作ってみましょう今回は電源を取るために Arduino を用います。Arduino を用いると、プログラムで LED を制御できますが、それは次回以降に。

Arduino、スイッチ、抵抗、LED を用いて簡単な回路を組む。

使うもの

- 電源 (Arduino, 5V)
- ブレッドボード
- LED
- タクトスイッチ
- 抵抗 (330 Ω)

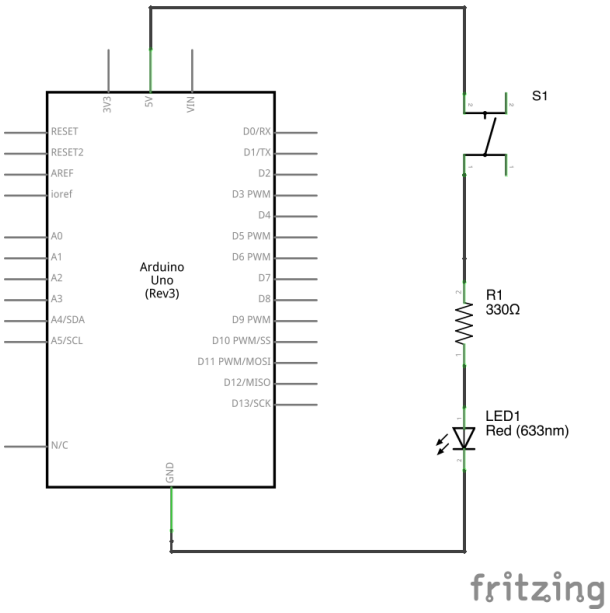

図 6: 回路図

図 7: 配線例: タクトスイッチを離した状態だと LED が消灯し、押した状態だと LED が点灯する