

## 第2講 プログラミングでLEDを制御する

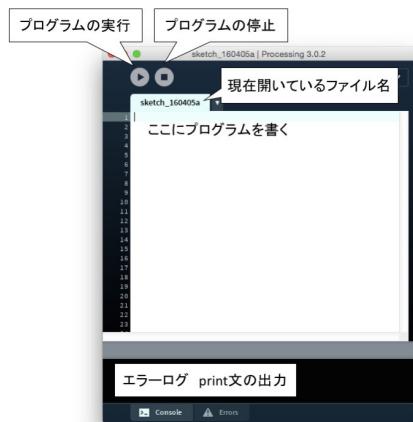
### 1 本実習の目標

- Processing で図形や絵を描いてみる
- Processing に Arduino (Firmata) をインストールする
- Processing から Arduino を制御するためのシリアルポートを把握する
- Arduino の Digital Output を用いて LED を光らせる
- Arduino の Digital Input を用いてスイッチの ON/OFF を取得する

### 2 Processing の基礎

本実習では Processing<sup>1</sup> 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。

Processing による基本的なプログラミングを行います。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。

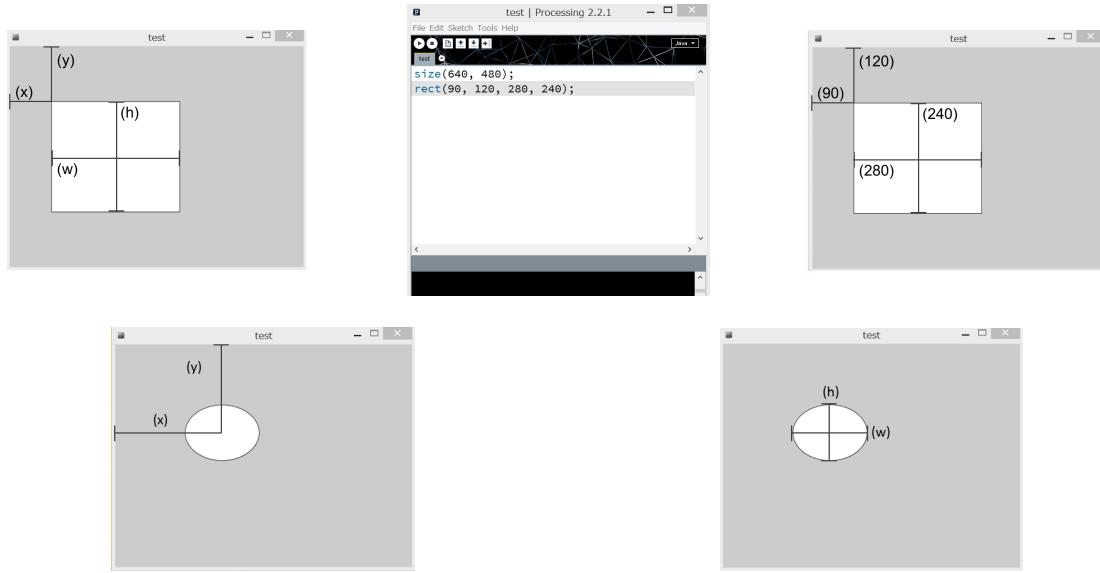


#### 基本的な図形を描く

- 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

---

<sup>1</sup><http://processing.org>



## 図形の色を変える

Processing で図形などの色を指定したいときは、Red、Green、Blue をそれぞれ 256 段階で指定します。

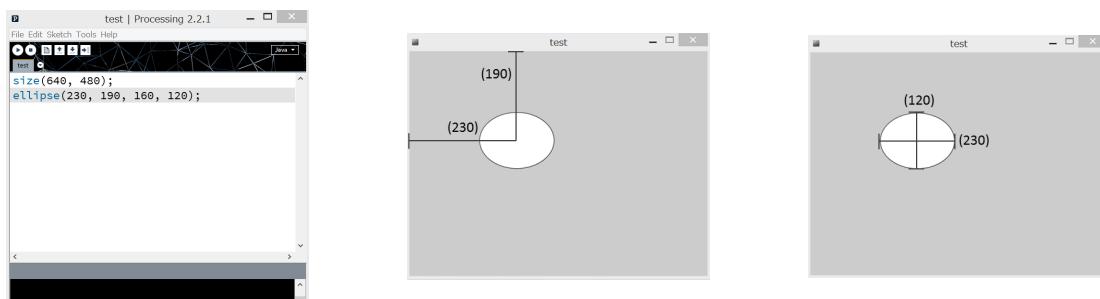
```
// 四角形を赤で表示する
fill(255, 0, 0);           // 赤は r = 255, g = 0, b = 0
rect(10, 10, 100, 50);
```

また、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();`
- 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 関数が繰り返されます（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ます。

```
void setup(){
    // 初期化の処理をここに書く
    // プログラム開始時に一度だけ実行される
    // 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、高さ 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します。

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ます。

```
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なります。

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 座標を取得できます。

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

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

## 3 Arduino の準備

まず、Processing から Arduino を用いるための準備をしましょう。本実習ではスイッチやセンサ、LEDなどを PC から制御するために Arduino<sup>2</sup> というマイコンを用います。

本来、Arduino と Processing を連携させるためには、シリアル通信を用いますが、そのためのプログラムを書くのは少し面倒です。実習では楽をするために、Firmata<sup>3</sup> を用います。

### ★ライブラリのインストール方法

まず、ライブラリをインストールするために WiFi に接続します。

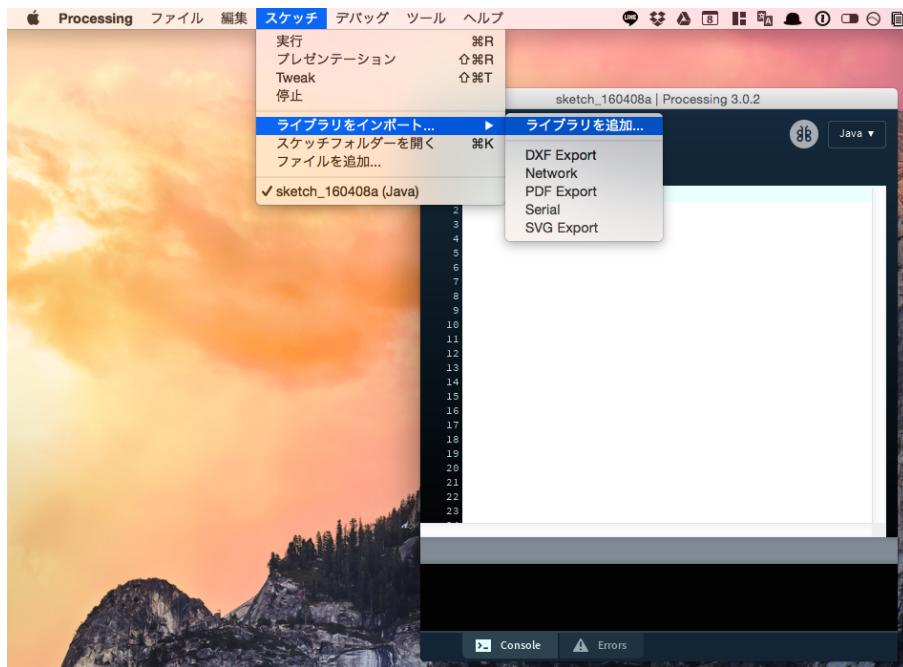


Processing のメニューから以下のライブラリを追加します。

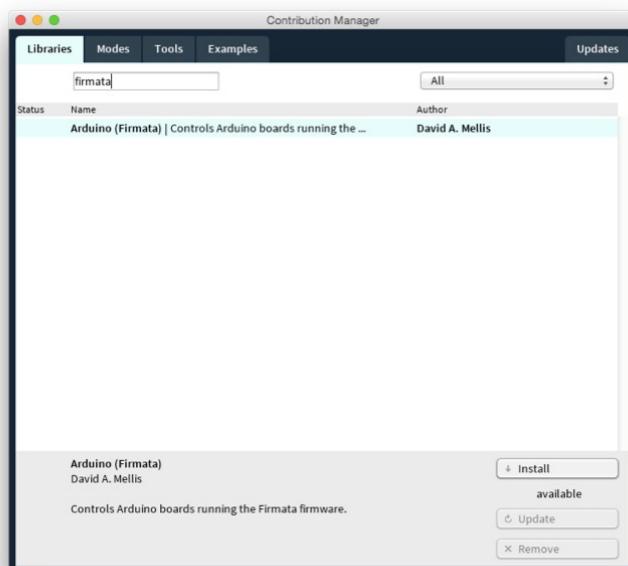
- スケッチ → ライブラリをインポート... → ライブラリを追加...

<sup>2</sup><http://arduino.cc>

<sup>3</sup><http://firmata.org>



- 図2のように検索バーで「firmata」と入力し、「Arduino(Firmata)」を選択、「Install」をクリックします。



これでライブラリのインストールは完了です。

## 4 シリアルポートの確認

Processing から Arduino にインストールした Firmata を操作するには、使用しているシリアルポートの環境を知る必要があります。まず、下記のコードを Processing に入力してください。

```
import processing.serial.*;
import cc.arduino.*;
Arduino arduino;

void setup() {
    println(Arduino.list());
}
```

すると、Processing の下部のコンソールに以下のようなメッセージが表示されます。

```
/dev/cu.Bluetooth-Incoming-Port
/dev/cu.Bluetooth-Modem
/dev/cu.usbmodem1421
/dev/tty.Bluetooth-Incoming-Port
/dev/tty.Bluetooth-Modem
/dev/tty.usbmodem1421
```

次に、「/dev/...」から始まる記述に対して、それぞれ順番に 0 から番号を付与します。

```
[0] /dev/cu.Bluetooth-Incoming-Port
[1] /dev/cu.Bluetooth-Modem
[2] /dev/cu.usbmodem1421
[3] /dev/tty.Bluetooth-Incoming-Port
[4] /dev/tty.Bluetooth-Modem
[5] /dev/tty.usbmodem1421
```

この中から、「/dev/tty.usbmodem…」もしくは「/dev/tty.usbserial…」から始まる記述の番号（上記の例では 5 番）をメモしておいてください。Arduino を使うためにはシリアルポートの指定をしなければなりません。続いて、下記のコードの Arduino.list()[5] の部分の数字を先ほどメモした番号に変更し、Processing に入力してください。

```
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;

void setup() {
    // Arduino の初期化
    // シリアルポートの指定など
    // Arduino.list()[5] は環境によって変える
    arduino = new Arduino(this, Arduino.list()[5], 57600);
}
```

これで Processing から Arduino を用いるための準備は完了です。これらの命令は今後も Arduino を用いる際に必ず使うので、テンプレートとして保存しておきましょう。

- ファイル → 名前を付けて保存... → 「ArduinoTemplate」という名前で保存します。

今後、Arduino を使用するプログラムを書く際は、この「ArduinoTemplate」をもとに編集しましょう。

## 5 Digital Output

では、まず簡単なプログラムで動作を確認してみましょう。LED が点灯するプログラムを作成してみましょう。Arduino 側は、Digital Out の 13 番に LED を接続しておきます。

### 5.1 LED を点灯させる

Digital Output を使って LED を点灯させてみましょう。

回路

LED を接続したピンに電圧をかけると LED が点灯する回路を作りましょう。

- 使う部品

- Arduino
- 抵抗
- LED

- ポイント

- 5V → LED → 抵抗 → GND の順に
- LED には極性があるので向きに注意

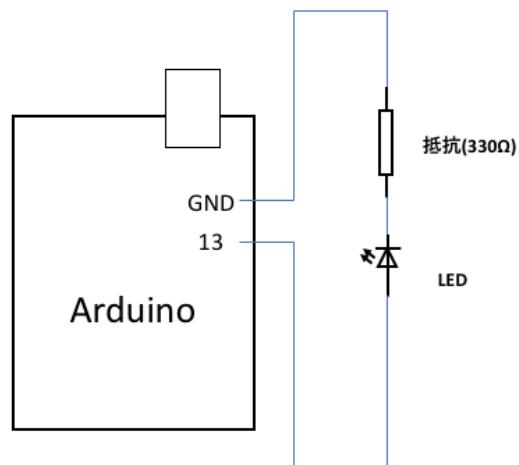


図 1: 回路図

プログラム

```

import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin = 13; // LED を接続したピンの番号

void setup() {
    arduino = new Arduino(this, Arduino.list()[5], 57600);

    // Arduino のピンモードを設定
    // ここでは 13 番ピンを Output 用に設定
    arduino.pinMode(ledPin, Arduino.OUTPUT);
}

void draw() {
    // Arduino の 13 番ピンを HIGH (5V) に
    arduino.digitalWrite(ledPin, Arduino.HIGH);
}

```

## 5.2 LED を点滅させる

Digital Output を使って LED を点滅させてみましょう。

### プログラム

```
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin = 13; // LED を接続したピンの番号

void setup() {
    arduino = new Arduino(this, Arduino.list()[5], 57600);

    // Arduino のピンモードを設定
    // ここでは 13 番ピンを Output 用に設定
    arduino.pinMode(ledPin, Arduino.OUTPUT);
}

void draw() {
    // Arduino の 13 番ピンを HIGH (5V) に
    arduino.digitalWrite(ledPin, Arduino.HIGH);
    delay(500); // 500ミリ秒間待つ

    // Arduino の 13 番ピンを LOW (0V) に
    arduino.digitalWrite(ledPin, Arduino.LOW);
    delay(500);
}
```

## 5.3 Processing の入力に応じて LED を点灯させる

Processing の画面上でマウスを押すと LED が点灯するプログラムを作成してみましょう。Arduino 側は、Digital Out の 13 番に LED を接続しておきます。

### mousePressed

mousePressed 変数はマウスが押されているか押されていないかによって、それぞれ true と false に値が変わります。これを用いると、「マウスがクリックされたときに何かをする」という動作が実現できます。

```
if (mousePressed) {
    // マウスが押されているときの処理
} else {
    // マウスが押されていないときの処理
}
```

## プログラム

```

import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin = 13;
color bgColor = color(0);

void setup() {
    size(400, 200);
    arduino = new Arduino(this, Arduino.list()[5], 57600);
    arduino.pinMode(ledPin, Arduino.OUTPUT);
}

void draw() {
    if (mousePressed) {
        arduino.digitalWrite(ledPin, Arduino.HIGH);
        bgColor = color(255,0,0);
    } else{
        arduino.digitalWrite(ledPin, Arduino.LOW);
        bgColor = color(0);
    }
    background(bgColor);
}

```

画面をクリックすると、LED が点灯します。

## 6 Digital Input

### 6.1 スイッチの ON/OFF を読み取る

Arduino の Digital Input を用いてスイッチの ON/OFF を取得する。

#### 回路

デジタル回路の場合、入力端子がどこにも接続されていないような状態（オープン）が起こると、電圧が High または Low に定まらず誤動作の原因になります。そのため、回路を安定させるためにプルアップ抵抗/プルダウン抵抗と呼ばれる抵抗を用います。

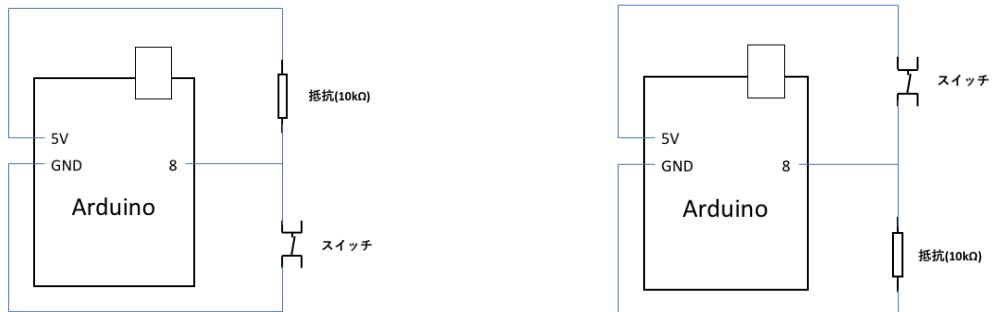


図 2: プルアップ抵抗 (左) と プルダウン抵抗 (右)

## プログラム

```
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int switchPin = 8; // スイッチを接続したピンの番号

void setup() {
    size(400, 300);
    arduino = new Arduino(this, Arduino.list()[5], 57600);
    arduino.pinMode(switchPin, Arduino.INPUT); // ピンモードを Input に
}

void draw() {
    // 8 番ピンの電圧を取得し、それが HIGH ならば
    if (arduino.digitalRead(switchPin) == Arduino.HIGH) {
        background(255, 0, 0); // 背景を赤に
    } else {
        background(0, 0, 0); // そうでなければ (LOW ならば) 背景を黒に
    }
}
```