# 第 4 講 Analog Output を用いて LED を制御する

## 1　本実習の目標

- Arduino の Analog Output を用いて LED を徐々に光らせる。
- Arduino の Analog Output を用いてフルカラー LED を光らせる。

## 2　Analog Output

前回までは、Digital Input と Digital Output を用いて、スイッチの ON/OFF を取得したり、LED を点滅させたりしました。しかし、LED をじわじわと光らせるといった緩やかな変化を取得したい場合 Digital Input/Output を用いても取得することはできません。そのような場合には Digital Input/Output ではなく Analog Input/Output を使います。

今回は Analog Output を用いて、LED を任意の明るさで光らせる、ということをやってみましょう。

### PWM (Pulse Width Modulation)

実は、Arduino においての Analog Output は実際に電圧が変化するのではなく、PWM (Pulse Width Modulation) と呼ばれる方式で擬似的に実現しています。

Processing から Arduino の Analog Output を用いるためには準備として、

```
// pinNum は回路に合わせて変える
arduino.pinMode(pinNum, Arduino.OUTPUT);
```

が必要となります。これは Digital Output のときと同様です。

Digital Output のときは arduino.digitalWrite() を用いましたが、Analog Output の場合は

```
// value は 0〜255 で設定
arduino.analogWrite(pinNum, value);
```

を用います。Digital Output のときは、Arduino.HIGH (5V) か Arduino.LOW (0V) のどちらかにしましたが、Analog Output の場合は 0〜255 の値で自由に設定できます。これにより、緩やかに LED の明るさを変化させるといったことが可能になります。

## 3　LED を徐々に光らせる

では、Analog Output を用いて実際に LED の光り方を細かく制御してみましょう。

### 3.1　LED のフェードインとフェードアウト

#### for 文

for 文はある一定の回数だけ繰り返し処理を実行したい場合に用いる文です。書式は以下の通りです。初期化式には通常の代入文を記述します。継続条件式には式が真の間、ループを続けさせるための条件式を記述します。再初期化式には「i++」や「i−」、「i = i+2」といった式を記述します。

```
for(初期化式; 継続条件式; 再初期化式) {
  文;
}
```

## 回路

第 2 講の Digital Output で作成した LED を光らせる回路と似ています。違いは、出力させるピンが PWM であるかどうかです。注意してください。

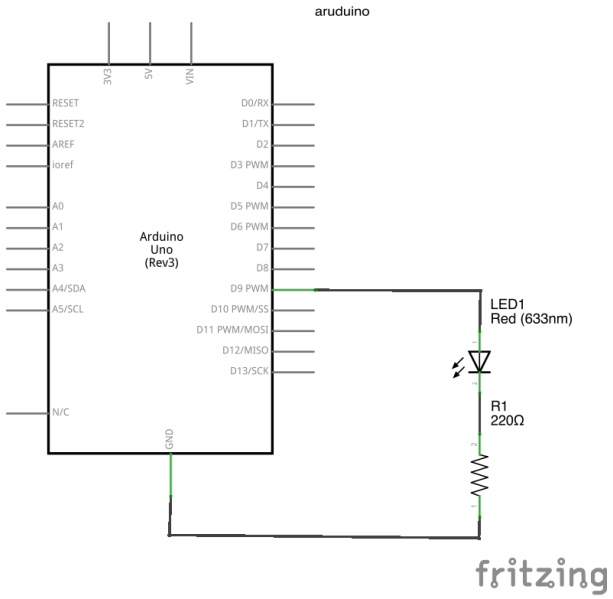


図 1: LED のフェードインとフェードアウトの回路図

## プログラム

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

Arduino arduino;
int ledPin = 9; // LEDが接続されたピン番号

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

void draw(){
  //LEDがだんだん明るくなる
  for(int i=0; i<255; i=i+10){
    arduino.analogWrite(ledPin,i);
    delay(50);
  }
  //LEDがだんだん暗くなる
  for(int i=255; i>0; i=i-10){
```

```
    arduino.analogWrite(ledPin,i);
    delay(50);
  }

}
```

### mouseX と mouseY

ついでに Processing の新しい命令を覚えましょう。前回の実習で、マウスボタンの ON/OFF (true/false) を mousePressed 変数で取得できることを学びました。今回はボタンを押したかどうかではなく、マウスポインタの座標を取得してみましょう。マウスポインタの x 座標と y 座標を取得するためにはそれぞれ mouseX 変数と mouseY 変数を用います。

### サンプルプログラム

```
void setup() {
  size(256, 256);
}

void draw() {
  background(0);
  // マウスポインタの座標に円を描く
  ellsepse(mouseX, mouseY, 32, 32);
}
```

次に、 mouseX、mouseY の値によって LED の明るさを変化させる、ということをやってみましょう。

回路は前回とほとんど同じで OK ですが、接続するピンに注意してください (ledPin を自分の回路で用いてるピンと置き換える)。また、PWM を用いる場合は、番号の前に ”~” が付いているピン (~ 9 とか) を用いてください。こちらにも注意してください！

### プログラム

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

Arduino arduino;
int ledPin = 9;   // LEDが接続されたピン番号

void setup() {
  size(256, 256);
  arduino = new Arduino(this, Arduino.list()[5], 57600);
  arduino.pinMode(ledPin, Arduino.OUTPUT);
}

void draw() {
  // マウスのx座標をLEDの明るさにする
  arduino.analogWrite(ledPin, mouseX);
}
```

# 4　フルカラー LED

フルカラー LED を用いると、光の三原色をそれぞれ制御することによって、様々の色の表現が可能となります。フルカラー LED の構造は単純で、内部に赤、緑、青の 3 つの LED が入っていると考えれば良いです。そのため、単色 LED のときは光り方を制御するために 1 つの Output を用いましたが、フルカラー LED の場合は 3 色をそれぞれ制御する必要があるため 3 つの Analog Output を用います。また、端子が 4 本出ているものが多いですが、これはアノード (+) またはカソード (-) が 3 つ分共通になっているためで、それぞれアノードコモン (図 2 (左) ) 、カソードコモン (図 2 (右) ) と呼びます。

## 回路

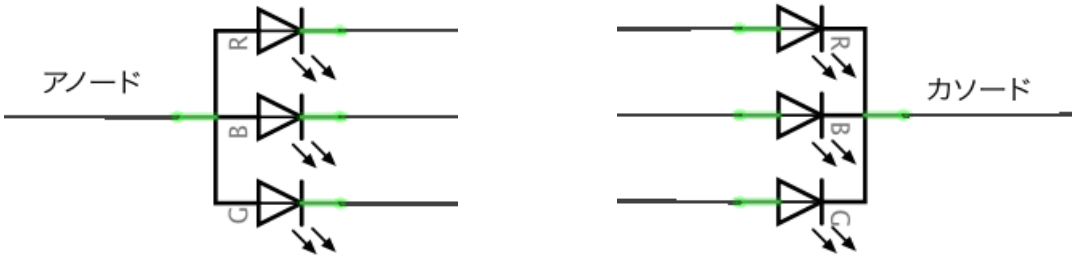


図 2: フルカラー LED アノードコモン (左) とカソードコモン (右)

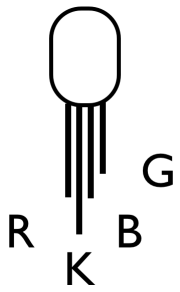


図 3: フルカラー LED

本実習では、カソードコモンのものを用います。それぞれの端子は

R: 赤

G: 緑

B: 青

K: - (カソード)

に対応します。順番に注意してください！

## 回路

## プログラム

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

Arduino arduino;
int LED_R = 9;     // LED 赤
int LED_B = 10;    // LED 青
int LED_G = 11;    // LED 緑
```

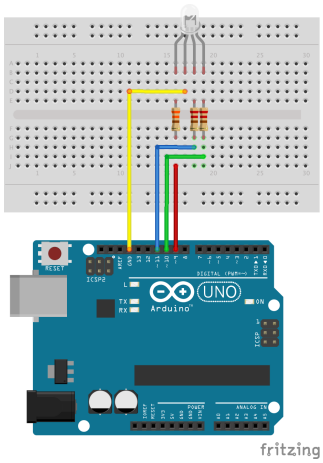


図 4: フルカラー LED の配線図

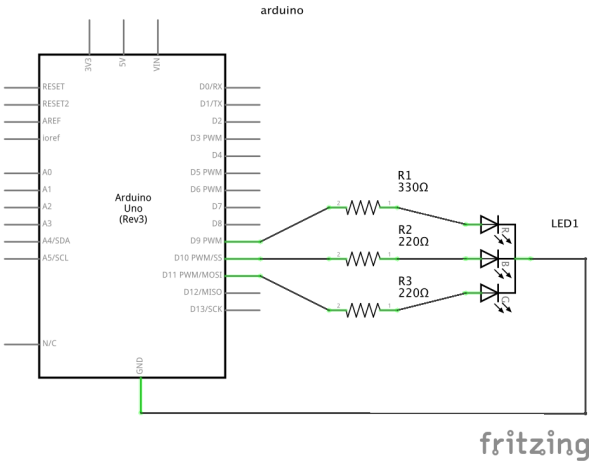


図 5: フルカラー LED の回路図

```
void setup() {
  size(256, 256);
  arduino = new Arduino(this, Arduino.list()[5], 57600);
  //LEDのピンを出力に設定する
  arduino.pinMode(LED_R, Arduino.OUTPUT);
  arduino.pinMode(LED_B, Arduino.OUTPUT);
  arduino.pinMode(LED_G, Arduino.OUTPUT);
}

void draw() {
  // LEDの明るさをセット
  arduino.analogWrite(LED_R, mouseX);
  arduino.analogWrite(LED_B, mouseY);
  arduino.analogWrite(LED_G, 127);
}
```