第 6 講 フルカラー LED を制御する

フルカラー LED を制御し、任意の色で光らせる。Gainer mini の「analog output」「digital output」の違いを試してみる。

フルカラー LED とは

フルカラー LED を用いると、光の三原色を一つの LED の中で扱うことができる。光の三原色を混色することで、様々の色の表現ができる。白色に点灯させるには、すべての出力が均等になるようにする。フルカラー LED には、内部に赤、緑、青の 3 つの LED が入っている。外部に足が 4 本出ているものが多いが、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、カソードコモンと呼ぶ。本実習では、カソードコモンのものを用いる。足の長さで、それぞれが何に対応しているかが区別できるようになっている。

実習で用いる LED は、それぞれの足が・R:赤色・K:(カソード) R K G ・B:青色・G:緑色 B に対応する。

フルカラー LED の接続

カソードコモンタイプの LED は出力から GND へ流れ出す形で制御する。本実習で用いるカソードコモンタイプの場合、赤・緑・青・カソードの 4 本の足がある。Gainer mini の出力から、それぞれの色に適した値の抵抗器を経由して接続し、カソードは GND に接続する。

回路図 その 1 回路図 その 2

パターンによって動作を振り分ける

条件分岐には通常 if 文を使うが、数多くの振り分けを行うとプログラムが頻雑になってしまう。こういった場合の時には、switch 文を使うと便利である。

switch 文では、if 文のように true と false の 2 択ではなく、変数の内容に応じて適切な場所にワープさせることが可能である。

```
switch(<<入力変数>>){
  case <<パターン1>> : ←コロン
    <<入力変数 == パターン1の時の処理>>
    break; ← セミコロン
  case<<パターン2>> :
    <<入力変数 == パターン2の時の処理>>
    break;
  default :
    <<どのcaseにも当てはまらなかった時の処理 >>
    break;
}
```

TRY1

Processing で switch 文を用いて、キーボードの 1 が押されたら赤、2 が押されたら緑、3 が押されたら青、4 が押されたら白の四角形が表示され、プログラムの起動時や、他のキーが押されれば黒の四角形が表示されるようなプログラムを作成する。

[Hint 1] キーボードの押したキーの判別は システム変数の key を用いる ことで簡単に取得できる。[Hint 2] case 以下でのキーの指定は、” ”(ダブルクォーテーション) ではなく、’’ (シングルクォーテーション) で囲む [Hint 3] 四角形の色の切り替えには fill(); を用いる [Hint 4] ...(前略)...

```
void draw(){
  switch(key){
    case ‘1’:
      fill(???, ???, ???);
      break;

     ...(中略)...
  }
  rect(???, ???, ???, ???);
}
```

1　Gainer mini の Digital Output を用いる

Gainer mini の dout ピンを用いて、電流のオン・オフ制御を簡単に切り替えることができる。

setHigh(int ch); ch で指定した dout ピンの状態を High にする (ON にする) setLow(int ch); ch で指定した dou ピンの状態を Low にする (OFF にする)

TRY2

1 ページ目の「回路図 その 1」を参考にして、デジタル出力を使った回路を組み立てる。(抵抗は、赤には 330 Ω、緑・青には 180 Ωを用いる。)

TRY3

TRY1 のプログラムを改造して、画面に表示される四角形の色と、フルカラー LED の色が同じになるようなプログラムを作成する。黒の時は LED を消灯する。

[Hint1] (回路図通り組み立てていれば) 赤色は dout0、緑色は dout1、青色は dout2 にそれぞれ対応する。

[Hint2]

```
import processing.gainer.*;
Gainer gainer;
void setup() {
  size(200, 200);
  colorMode(RGB, 256);
  rectMode(CENTER);
  gainer = new Gainer(this);
}
```

```
void draw() {
  switch(key) {
    case '1':
      fill(255, 0, 0);
      gainer.setHigh(0);
      gainer.setLow(1);
      gainer.setLow(2);
      break;
    case '2':
      <<緑色にする時の処理>>
    case '3':
      <<青色にする時の処理>>
    case '4':
      <<白色にする時の処理>>
    default:
      <<LEDを消す時の処理 >>
  }
  rect(width/2, height/2, 100, 100);
}
```

2　Gainer mini の Analog Output を用いる

Gainer mini の aout ピンを用いることで、アナログ値を出力することができる。アナログ出力を用いることで、フルカラー LED で様々な色を表現することができる。

analogOutput(int ch, int value); ch で指定したアナログポート value で指定した値を出力する。(value は 0255 までの 256 階調)

余談:アナログ出力の正体 Gainer mini でアナログ出力と呼んでいるものは、実際には電圧が変化するものではなく、PWM (Pulse Width Modulation: パルス幅変調) と呼ばれる方式で擬似的に実現している。例えば、値が 1 に設定した場合、出力信号は非常に短い区間だけ 5V で、残りの区間は 0V になる。値を 127 に設定した場合は、半分の区間が 5V で残りの半分は 0V になっている。値を 255 に設定したときは常に 5V が出力される。

TRY4

1 ページ目の「回路図 その 2」を参考にして、TRY2 で組み立てた回路を改造し、アナログ出力を使った回路を組み立てる。

TRY5

Processing でマウスの動きに反応して、フルカラー LED の色を制御できるようなプログラムを作成する。マウスは 2 軸の値しか取れないので、RGB のいずれかの値を固定値にする。

[Hint 1] マウスの X 軸の値は mouseX で、マウスの Y 軸の値は mouseY で取得できる。

[Hint 2]

```
import processing.gainer.*;
Gainer gainer;
void setup() {
```

```
  size(255, 255);
  colorMode(RGB, 256);
  gainer = new Gainer(this);
}

void draw() {
  background(???, ???, ???);
  gainer.analogOutput(0, ???);
  gainer.analogOutput(1, ???);
  gainer.analogOutput(2, ???);
}
```