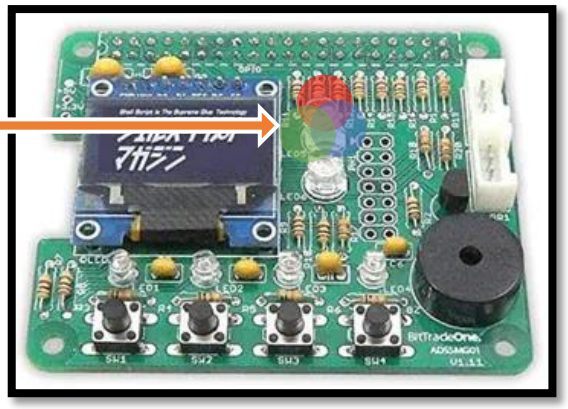
フルカラーLED とは

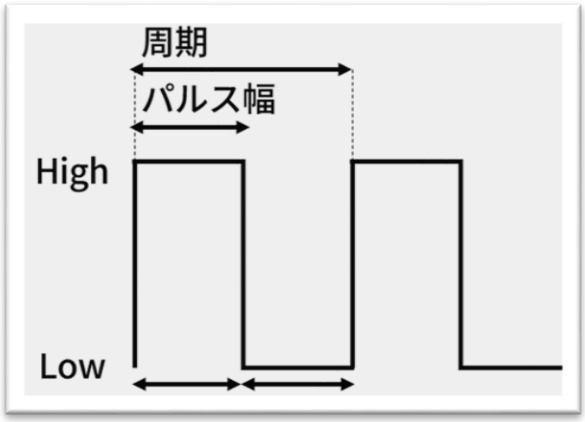


フルカラーLED とは赤・緑・青の 3 色の LED が
1 つに纏められた LED のこと。
光の三原色を組み合わせる事で様々な色を作り出せる。
入門ボードには 2 個搭載されており、それぞれ 3 本の
信号線（アノード）と 1 本の共通グランド（カソード）
によって 3 色の LED 制御が可能となる。

PWM 出力とは

フルカラーLED を制御する方法として、今回は PWM 出力を使用する。
PWM（Pulse Width Modulation：パルス幅変調）出力とは、電圧の High - Low を高速に切り替えて、
一定の周期の中で High-Low の比率を調整することで、中間の値を疑似的に表現する信号方式のこと。

右の図で例えると、周期の中で半分の時間 High に
なっており、もう半分の時間 Low になっている。
この場合、全体の平均出力が 50%となる為、
LED の明るさを以前のデジタル出力と比べて
半分の明るさに調節する事ができる。
この High を継続している時間をパルス幅と言い、
周期の中のパルス幅の比率をデューティ比と言う。
つまり、右の図はデューティ比 ＝50%となる。



フルカラーLED の GPIO 番号

今回使用するフルカラーLED の
GPIO 番号は以下の通りです。

LED_B = 2
フルカラーLED 内の青色

LED_G = 3
フルカラーLED 内の緑色

LED_R = 28
フルカラーLED 内の赤色

表 5.1.1　Raspberry Pi の拡張コネクタ

備考	機能	ピン名	GPIO番号	ピン番号		ピン番号	GPIO番号	ピン名	機能	備考
50mAまで(1番ピンと17番ピンの合計)		3.3V		1		2		5V		
1.8kΩプルアップ抵抗付	I²C1 (SDA)	GPIO2	8	3		4		5V		
1.8kΩプルアップ抵抗付	I²C1 (SCL)	GPIO3	9	5		6		GND		
	GPCLK0	GPIO4	7	7		8	15	GPIO14	UART0 (TXD)	起動時にシリアルコンソールとして使用
		GND		9		10	16	GPIO15	UART0 (RXD)	起動時にシリアルコンソールとして使用
	SPI1 (CS1)	GPIO17	0	11		12	1	GPIO18	SPI1 (CS0)、PWM0	その他の機能：PCM_CLK
		GPIO27	2	13		14		GND		
		GPIO22	3	15		16	4	GPIO23		
50mAまで(1番ピンと17番ピンの合計)		3.3V		17		18	5	GPIO24		
	SPI0 (MOSI)	GPIO10	12	19		20		GND		
	SPI0 (MISO)	GPIO9	13	21		22	6	GPIO25		
	SPI0 (SCLK)	GPIO11	14	23		24	10	GPIO8	SPI0 (CS0)	
		GND		25		26	11	GPIO7	SPI0 (CS1)	
拡張基板(Hat)に搭載のEEPROM用信号		ID_SD		27		28		ID_SC		拡張基板(Hat)に搭載のEEPROM用信号
		GPIO5	21	29		30		GND		
		GPIO6	22	31		32	26	GPIO12	PWM0	
	PWM1	GPIO13	23	33		34		GND		
その他の機能：PCM_FS	SPI1 (MISO)、PWM1	GPIO19	24	35		36	27	GPIO16	SPI1 (CS2)	
							28	GPIO20	SPI1 (MOSI)	その他の機能：PCM_DIN
		GND		39		40	29	GPIO21	SPI1 (SCLK)	その他の機能：PCM_DOUT

※Raspberry Pi タイプ B は 26 ピンまでになります。　　拡張コネクタのピンアサイン

練習（TryFullColor.java）

以下のサンプルプログラムを記述して実行し、SW1 を押している間にフルカラーLED の 3 色（RGB）全てのデューティ値がインクリメント（1 増加）して点灯するか確かめなさい。

```
/*
 * TryFullColor.java
 * Date    : 2022/01/01
 * Author : IE1A 99 K.Murakami
 */

// GPIO ピンを利用するために必要なクラスを読み込む
import com.pi4j.wiringpi.Gpio;
import com.pi4j.wiringpi.GpioUtil;
import com.pi4j.wiringpi.SoftPwm;

public class TryFullColor {
    // Thread.sleep メソッドで発生する割り込み例外を throws する
    public static void main (String[] args) throws InterruptedException {

        System.out.println("プログラム開始");

        // デジタル出力信号を定数化
        final int HIGH = 1;
        final int LOW = 0;
        // デジタル入力信号を定数化
        final int ON = 0;
        final int OFF = 1;
        // PWM 出力の範囲を定数化（8 ビット）
        final int DUTY_MIN = 0;
        final int DUTY_MAX = 255;

        // LED のピン番号を宣言
        final int LED4 = 27;
        // SW のピン番号を宣言
        final int SW1 = 7;
        // フルカラーLED のピン番号を宣言
        final int LED_R = 28;
```

インポート文を忘れずに

このサンプルプログラム内でデジタル出力関連は使用しないが、以降の課題の為に記述しておく

デューティ値は 8 ビット（0~255）の範囲で扱う
デューティ比 0% = 0 を出力 → 消灯
デューティ比 50% = 127 を出力 → 明るさ半分
デューティ比 100% = 255 を出力 → 明るさ最大
となり、0～255 の範囲を超えてはならない

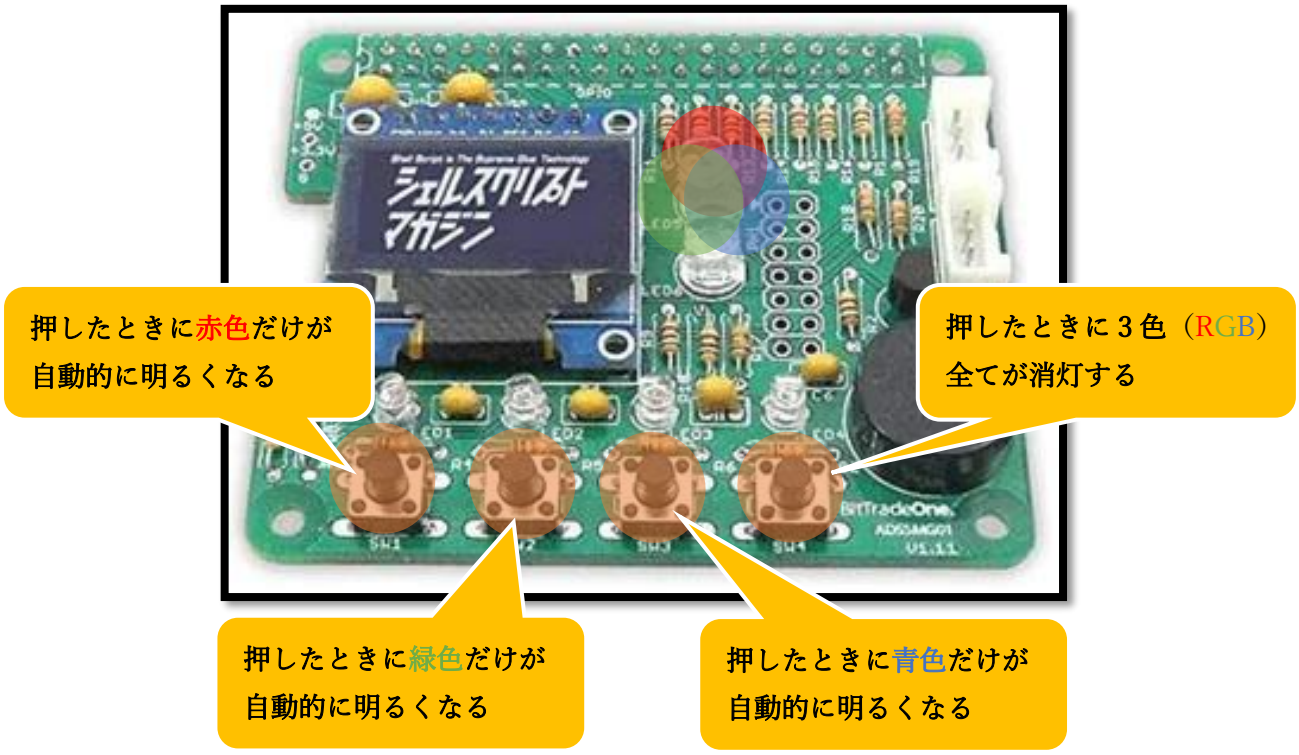
```
final int LED_G = 3;
final int LED_B = 2;

// GPIO を初期化
Gpio.wiringPiSetup();
System.out.println("GPIO 初期化完了");

// 各 LED を出力に設定
Gpio.pinMode(LED4, Gpio.OUTPUT);
// 各 SW を入力に設定
Gpio.pinMode(SW1, Gpio.INPUT);
// 各フルカラーLED を PWM 出力に設定
SoftPwm.softPwmCreate(LED_R, DUTY_MIN, DUTY_MAX);
SoftPwm.softPwmCreate(LED_G, DUTY_MIN, DUTY_MAX);
SoftPwm.softPwmCreate(LED_B, DUTY_MIN, DUTY_MAX);
System.out.println("GPIO 入出力設定完了");

// フルカラーLED の明るさを保持する変数を定義
int dutyR = 0;
int dutyG = 0;
int dutyB = 0;

// プログラムを終了させない為に無限ループする
while(true){
    // SW1 が ON になっている場合
    if(Gpio.digitalRead(SW1) == ON){

        SoftPwm.softPwmWrite(LED_R, dutyR); //赤点灯
        SoftPwm.softPwmWrite(LED_G, dutyG); //緑点灯
        SoftPwm.softPwmWrite(LED_B, dutyB); //青点灯
        System.out.println("R = " + dutyR + ", G = " + dutyG + ", B = " + dutyB);

        // 赤色のデューティ値が最大値未満の場合
        if(dutyR < DUTY_MAX){
            dutyR++;
        }
        // 緑色のデューティ値が最大値未満の場合
        if(dutyG < DUTY_MAX){
```

```
                    dutyG++;
                }
                // 青色のデューティ値が最大値未満の場合
                if(dutyB < DUTY_MAX){
                    dutyB++;
                }
            }
            // 100 ミリ秒待機する
            Thread.sleep(100);
        }
    }
}
```

課題 1 （FullColorAuto.java）

以下の条件でフルカラーLED の明るさを制御するプログラムを作成しなさい。
デューティ値を増やし続ける処理は **for 文を使用する**こと。

・SW1 を押したとき → 赤色のデューティ値を 0.2 秒間隔で 0 から 255 まで 5 ずつ増やし続ける。
・SW2 を押したとき → 緑色のデューティ値を 0.2 秒間隔で 0 から 255 まで 5 ずつ増やし続ける。
・SW3 を押したとき → 青色のデューティ値を 0.2 秒間隔で 0 から 255 まで 5 ずつ増やし続ける。
・SW4 を押したとき → 3 色（RGB）全てを消灯する。



課題 2 （FullColorAdjust.java）

前述の練習を別名保存し、以下の条件でフルカラーLED の明るさを制御するプログラムを作成しなさい。
3 色（RGB）全てのデューティ値の初期値は 0 とする。

【SW4 を押していない（離している）間】

・LED4 を消灯する。
・SW1 を押している間 → 赤色のデューティ値が最大値（255）未満の場合、インクリメントする。
・SW2 を押している間 → 緑色のデューティ値が最大値（255）未満の場合、インクリメントする。
・SW3 を押している間 → 青色のデューティ値が最大値（255）未満の場合、インクリメントする。
※SW1~3 を同時押しした場合、練習（TryFullColor.java）と同様の動作になる。

【SW4 を押している間】

・LED4 を点灯する。
・SW1 を押している間 → 赤色のデューティ値が最小値（0）より大きい場合、デクリメントする。
・SW2 を押している間 → 緑色のデューティ値が最小値（0）より大きい場合、デクリメントする。
・SW3 を押している間 → 青色のデューティ値が最小値（0）より大きい場合、デクリメントする。