「IoT 制作演習Ⅰ」第 03 章　GPIO 基本 アナログ出力

アナログ（PWM）出力…青色 LED を使う

《何ができる？》LED を点灯させるだけの「L チカ」を以前やったが、LED の明暗を制御することもできる。LED あかりの調光器などもそうだが、植物工場などで太陽光の代わりに栽培に使われるあかりもそうである。LED の明暗の制御をすることで人工的に朝・昼・夕方・晩を作り出す（図は LED ライト植物工場）。

アナログ出力…PWM と DAC

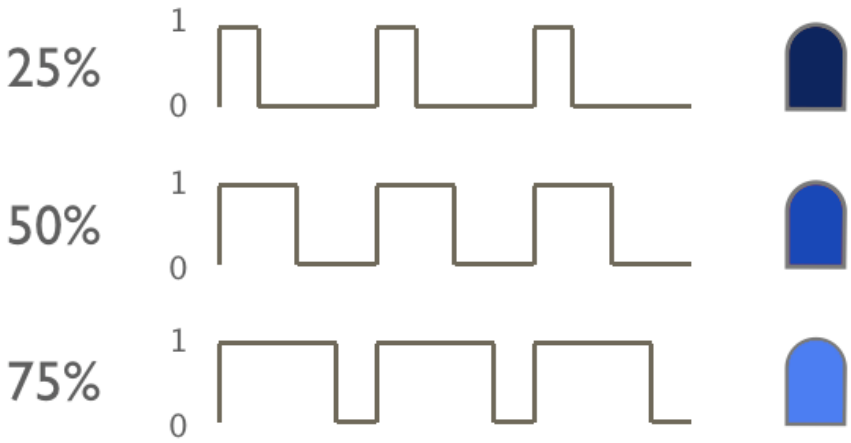
ESP32 には、2 種類のアナログ出力がある。

- PWM(Pulse Width Modulation)…今回やる方
  - パルス幅変調 HIGH/LOW のデジタル出力を細かくオン・オフして疑似的にアナログ電圧にする
- DAC(Digital-Analog Converter)
  - D-A コンバータ デジタルの値を直接アナログ電圧にコンバートする
  - Arduino UNO などの一般的な Arduino ボードでは実装されていない高度な機能

ここでは、Arduino ボードで一般的に使われている PWM のアナログ出力をやってみる。

PWM（パルス幅変調）の原理

PWM（Pulse Width Modulation）は、次の図のような周期的なパルス信号を生成し、1 になっている時間と 0 になっている時間の比率（デューティー比）を変えることで送信する電力を変化させる。



LED の点灯に PWM を使うと、1 になっている時間が短いと人間の目には LED が暗くなったように見え、時間が長いと明るく見える。

アナログ（PWM）出力に必要な関数

ESP32 では LED_PWM という、主に LED の明るさを制御するための PWM が実装されている。

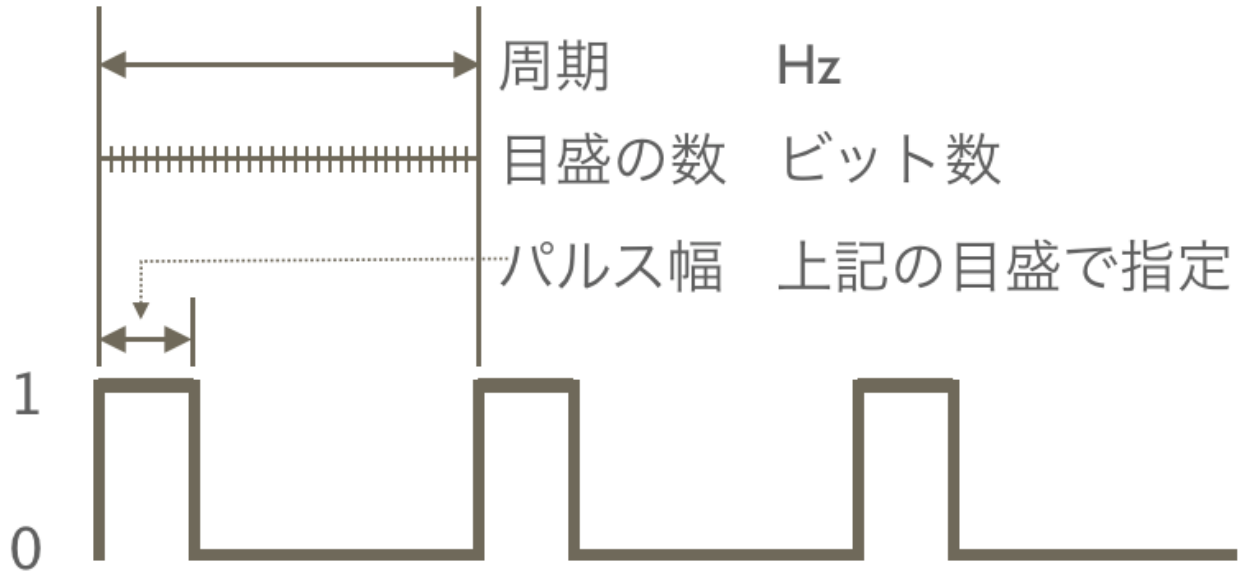
double **ledcSetup**(uint8_t chan, double freq, uint8t bit_num)	
LED PWM で利用するチャネルを設定する。	
【パラメータ】	chan：利用するチャネル。0～15。 freq：PWM の周波数。 bit_num：デューティ比を表すビット数。
【戻り値】	実際に設定された PWM の周波数。

void **ledcAttachPin**(uint8_t pin, uint8_t chan)	
LED PWM で利用するピンとチャネルを結びつける。	
【パラメータ】	pin：利用するピン chan：利用するチャネル
【戻り値】	なし

void ledcWrite(uint8_t channel, uint32_t duty)	
LED PWM を利用して、指定したデューティで PWM 出力する。	
【パラメータ】	channel：利用するチャネル。0～15 duty：デューティ 0～(2^{bit_num1-1})まで bit_num は、ledcSetup()で設定した bit_num
【戻り値】	なし

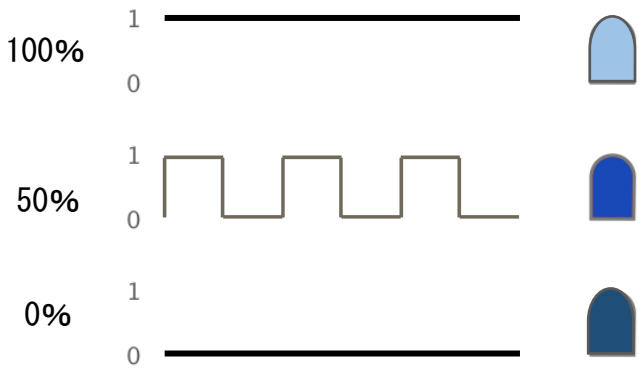
ESPr Developer で利用可能なピン（利用不可のピンは除く）

アナログピン番号	GPIO 番号	アナログピン番号	GPIO 番号	アナログピン番号	GPIO 番号
A4	32	A12	2(Boot Mode)	A16	14
A5	33	A13	15	A17	27
A10	4	A14	13	A18	25
A11	0(Boot Mode)	A15	12	A19	26



《サンプル①》　IO4 番ピンにアナログ(PWM)出力を、パルス周期 12,800Hz・目盛りの数 8 ビット（2^8⇒0～255）・パルス幅 255、127、0 で LED の明暗を制御する。

```
const int pwmPin = A10;//GPIO IO4pin

void setup() {
  ledcSetup(0, 12800, 8);
  ledcAttachPin(pwmPin, 0);
}

void loop() {
  ledcWrite(0, 255);
  delay(1000);
  ledcWrite(0, 127);
  delay(1000);
  ledcWrite(0, 0);
  delay(1000);
}
```



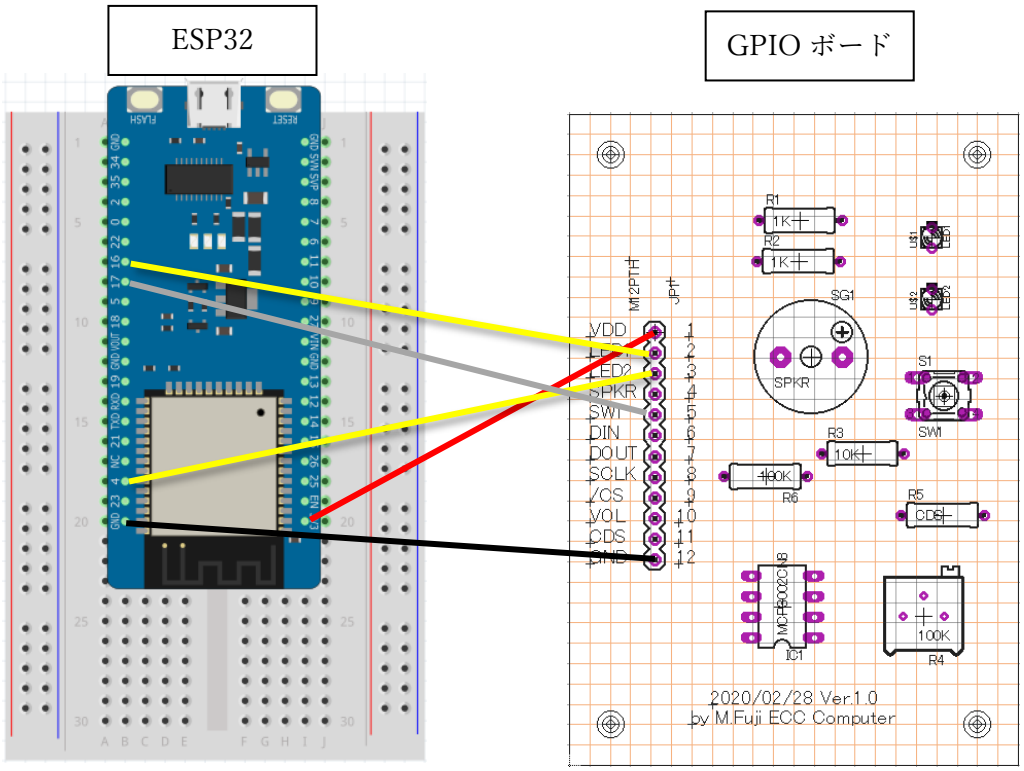
演習では、2 個目の LED の配線を行い、上記サンプルを動かした後、書きかえてみる。

第３回 演習

【第３回の目標】

ESP32 ボードに 2 個目の LED をつなぎ、アナログ(PWM)出力を行い、LED の明暗を制御する。

【1.　ESP32 と LED ほか電子工作部品との接続】



上記実体配線図を元に、5 本のジャンパーコードの配線を行う。接続するピンは以下の一覧の通り。

① ジャンパーコード（黄）ESP32 の IO16 － GPIO ボード LED1(2 番ピン)

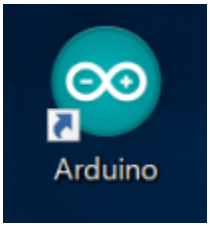
② ジャンパーコード（黄）ESP32 の IO4 － GPIO ボード LED2（3 番ピン）

③ ジャンパーコード（白）ESP32 の IO17 － GPIO ボード SW1(5 番ピン)

④ ジャンパーコード（赤）ESP32 の 3V3 － GPIO ボード VDD(1 番ピン)

⑤ ジャンパーコード（黒）ESP32 の GND － GPIO ボード GND(12 番ピン)

【2. Arduino スケッチのサンプルプログラムをプログラム実行】



デスクトップのアイコンをダブルクリックして
Arduino IDE を起動する。

【課題 08】プロジェクト名「kad08_PWM」

まずは、《サンプル①》を打ち込んで、動作確認をする

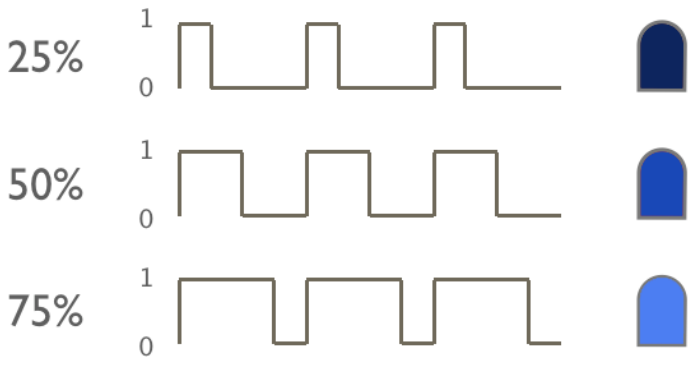
```
const int pwmPin = A10;//GPIO IO4pin

void setup() {
  ledcSetup(0, 12800, 8);
  ledcAttachPin(pwmPin, 0);
}

void loop() {
  ledcWrite(0, 255);
  delay(1000);
  ledcWrite(0, 127);
  delay(1000);
  ledcWrite(0, 0);
  delay(1000);
}
```

・GPIO IO4 番ピンを使う。

・チャネル 0 に 12800Hz のパルス周期で、8 ビット分のパルス目盛り(0～255)でセット
・チャネル 0 を GPIO IO4 番ピンに割り当てる

・デューティ比 100%で 1 秒点灯
(デューティ 255/255=100%)
一番明るい
・デューティ比 50%で 1 秒点灯
(デューティ 127/255≒50%)
半分の明るさ
・デューティ比 0%で 1 秒消灯
(デューティ 0/255=0%)
消える

動作が確認できたら、デューティ比 25%, 50%, 75%, 100%, 0% となるデューティ値を 1 秒ずつ繰り返すスケッチに書きかえて動作するか確認せよ。徐々に明るくなってから消灯をくり返せば課題 OK です。



この後、100%, 0%

【課題 09】 プロジェクト名「kad09_PWMfading」

IO4 番ピンに接続した LED が 10 ミリ秒おきに、デューティ 0～255 まで 1 ずつインクリメントし、255～0 まで 1 ずつデクリメントする三角波のような緩やかな明暗（フェードと呼ばれる）をくり返すスケッチを作成せよ。

ヒント（スケッチの枠組みのソース部分）

```
const int pwmPin = 4; //A10 でも 4 でもどちらでも使えるのを試す

void setup() {
  Serial.begin(115200);
  //PWM のセットアップを記述
  //PWM チャネルとピンの割り当てを記述
}
void loop() {
  static uint8_t duty = 0; //アナログ出力のデューティ値
  static int diff = 1; //増分・減分（使わなくても可）
  Serial.printf("%3d¥n", duty);// デューティの表示
  /*
    ここに PWM でチャネル 0 に LED の明暗を制御するコードを書く
    loop()関数が繰り返されるたびに duty が 0～255～0 と増減する
    コードを書く
  */
  delay(10); // 10 ミリ秒間隔で繰り返し
}
```

シリアルモニタ

```
  1   0 から 255 に
  2   255 から 0 に
  3   繰り返し表示
…
  9
 10
 11
…
254
255
254
253
…
  2
  1
  0
  1
…
```

文法コメント

```
static uint8_t duty = 0;
```

static 修飾子は、静的変数を宣言するもので、宣言された関数（ここでは loop()関数）が終了しても値を保持する変数となる。初期化子によって 0 が代入・初期化されているが、これは最初の 1 回目の実行のみで loop()関数の呼び出しが 2 回目以降になると無視される。その際に、前回実行された際の変数の値が保持されていて利用される。

※loop()関数の外側に変数 duty を宣言して、グローバル変数として利用する方法もあるが、この場合、setup()関数や自分が定義した関数からもアクセス可能になるため、モジュールの独立性が低いコードになる（バグが入り込みやすくなる）。

uint8_t 型は、符号なし 8 ビット整数型のこと。unsigned char 型の別名定義。C99 と C++11 から追加された。組み込み用途での C/C++では、符号の有無・ビット幅が明確なため、よく使われる。

```
Serial.printf("%3d¥n", duty);
```

C 言語の printf()関数と同じ書式指定子が使える。

"%3d¥n"は、右揃え 3 桁の整数＋改行となる。

【課題 10】 プロジェクト名「kad10_SwitchPWM」

IO17 番ピンに接続したタクトスイッチを押したら、IO4 番ピンに接続した青色 LED が 25%, 50%, 75%, 100%, 0%のデューティ比で点灯するスケッチを作成せよ。

ヒント（スケッチの枠組みのソース部分）

```
const int pwmPin = A10; // GPIO IO4pin
const int btnPin = 17; // GPIO IO17pin
bool flag = false; //フラグ初期値false:押されていない状態
//他にもグローバル変数を作ってもよい（配列も可）。

void setup() {
    //シリアル入出力の準備
    //アナログ(PWM)出力の準備
    //デジタル入力の準備
}

void loop() {
  /*
    ここにフラグ変数 flag を活用して、スイッチがおされたときだけ
    デューティの値が 25%, 50%, 75%, 100%, 0%と変化し青色 LED
    の明暗が変化するソースをかく
    同時にシリアルモニタにデューティの値
    が表示されるソースをかく
  */
  delay(10);
}
```

シリアルモニタ

```
duty=63
duty=127
duty=191
duty=255
duty=0
duty=63
duty=127
duty=191
duty=255
duty=0
…
スイッチがおされたときだけ
デューティの値が表示される
```

kad07_Count 参照