# ４．パルス制御回路

## 1. 目的

現在広く利用されているマイコンボードである Arduino を用い，LED やモータを用いて様々なパルス信号による制御手法について理解する．

## 2. マイコン (マイクロコントローラ)

### 2.1. 構成

普段の生活において，日頃から意識することもなく様々なところでマイコンによる電子制御が行われている．マイコンという呼称は，以前（数十年前）は「マイクロコンピュータ」の略称であり，マイクロプロセッサ，メモリ，周辺チップなどで構成された小さなコンピュータシステムを意味していた．しかし，現在ではマイコンという呼称は主に「マイクロコントローラ」の略称として使われており，マイクロコントローラは１個のシリコンにプロセッサおよびメモリ，周辺回路を実装したチップで，チップ単体で基本的な処理が可能な構成になっている．

### 2.2. マイコンによる電子制御

マイコンが処理できる信号は，基本的にはディジタル信号である．しかし，一般に普及しているマイコンの多くはチップ内に AD コンバータを実装しており，アナログ信号を入力し（ディジタル化して）処理できるようになっているものも多い．一方で，ディジタル値をアナログ信号に変換するための DA コンバータを実装しているものはほとんど無く，そのためマイコン自体の出力信号はほぼディジタル信号である．そのため，マイコンから周辺装置を制御するためにディジタル信号を用いた様々な制御方法がある．

### 2.3. Arduino

Arduino(アルドゥイーノ) は、AVR マイコン (ATMEL 社製のマイクロコントローラ)、入出力ポートを備えた基板、C++風の Arduino 言語とそれの統合開発環境から構成されるシステムである (Wikipedia より)。簡単に言うと，マイコンボード (Arduino 本体) と C 言語的にプログラムが組める開発環境 (Arduino IDE) を一つのパッケージにしたものである．

従来のマイコン (例えば PIC など) の場合，少なくともマイコン（およびマイコンを載せるために外部クロックなど最低限必要な部品が搭載されている基板)，マイコンにプログラムを書き込むためのライタ，プログラムを組むための開発環境が必要となっている．しかし，Arduino ではマイコンその他部品を搭載した完成品が安価で購入でき，パソコンから直接プログラムを書き込むことができるので書き込み用ライタも必要なく，C ベースでプログラムを組むことができる開発環境が無料で簡単に入手できる．また，現在広く使用されていることから参考文献なども多く，比較的容易にセンサやモータなどによる電子工作を行うことができる．

## 3. パルス制御

### 3.1. パルス

パルスとは，一般に短時間に急激に変化する信号の総称と定義されている．ディジタル回路（または電子回路）の分野では，一定の幅を持った矩形波（例えば，通常は '0' の状態である時間幅で '1' になる信号，またはその逆）をパルスと呼び，このパルスが繰り返された信号を一般にクロック信号と呼ぶ．

パルス信号自体は，単なる '0'/'1' を一定時間変化させた信号である．しかし，パルス幅やパルスの繰り返し周波数などを任意に変化させるなどによって，アナログ情報をパルスで表現することができる．このように，パルス信号（またはパルス信号の組み合わせ）によって波形信号のようなアナログ信号を変換・生成することをパルス変調を呼ぶ．

### 3.2. 主なパルス変調方式

主に利用されているパルス制御方式としては，以下の方式が挙げられる．

#### 3.2.1. パルス幅変調 (Pulse Width Modulation, PWM) 方式

パルス幅変調 (PWM) は、パルス幅とその間隔，および正負により波形を表すものである．ディジタル回路の場合，アナログ信号の振幅を，ある周期の繰り返しパルス（クロック信号）のデューティ比（1 クロックあたりの'0'/'1' の割合）に変換して表現するものである．例えば，1 クロックあたりの時間幅を $T$ とすると，振幅が最小の場合にはパルス幅が 0 で振幅が大きくなるにつれてパルス幅が広がり，振幅が最大ではパルス幅が $T$ となる．

1 クロックのうち，信号が '1' の時間を $T_{on}$，'0' の時間を $T_{off}$ とする．この時，デューティ比 = 0.5 の PWM 信号は図 1 となる．

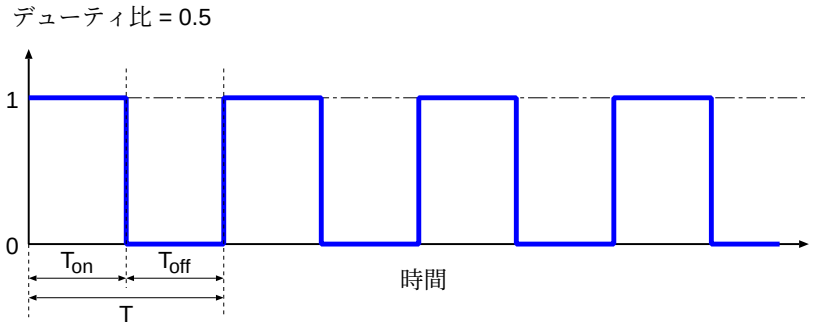



図 1: デューティ比 = 0.5 の PWM 信号

さらに，デューティ比を変化させた時の PWM 信号は図 2 のようになる．

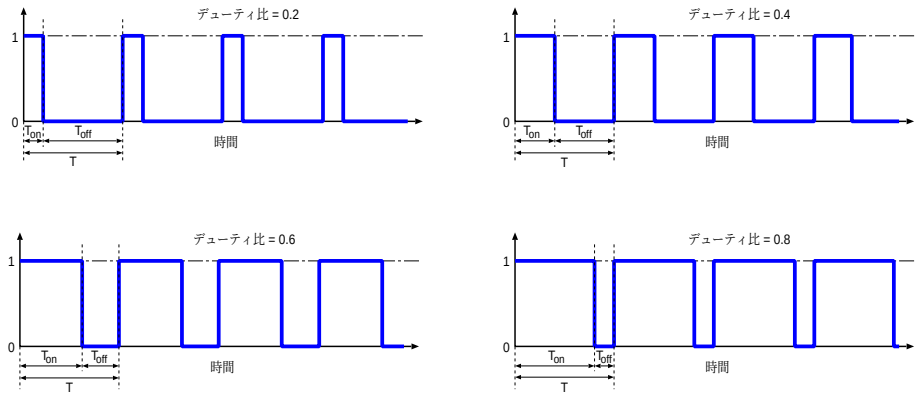



図 2: 様々なデューティ比での PWM 信号

ディジタル回路ではパルス信号を柔軟に生成できるので，パルス幅や周期を様々な値にすることができる．現在普及しているマイコンの多くは，チップに PWM ユニットを実装している．

#### 3.2.2. パルス振幅変調 (Pulse Amplitude Modulation, PAM) 方式

パルス振幅変調 (PAM) は，一定間隔のパルスの電圧の振幅および正負により波形を表すものである．主にアナログ時分割多重伝送に用いられている．

#### 3.2.3. パルス符号変調 (Pulse Code Modulation, PCM) 方式

パルス符号変調 (PCM) は，サンプリングされた信号を量子化し、二進符号化して表現したものである．主にデジタル伝送に用いられているが，オーディオ関係でもよく使われ，PCM 音源などともいう．

#### 3.2.4. その他

- **パルス密度変調 (Pulse Density Modulation, PDM) 方式**：D/A によく使われる．
- **パルス位置変調 (Pulse Position Modulation, PPM) 方式**
- **パルス周波数変調 (Pulse Frequency Modulation, PFM) 方式**：最近ではあまり使われていない．

### 3.3. PWM 変調による制御

3.2.1 節のように，多くのマイコンでは PWM ユニットを実装されており，PWM 変調によって様々な周辺装置を制御することができる．以下にその代表的なものを挙げる．

#### 3.3.1. LED の調光

LED の明るさは，流れる電流に比例する．そのため，LED の調光（明るさの制御）のためには流れる電流量を制御してやればいいが，マイコン単体で電流を制御することはできず，ディジタル回路として電流を変化させる回路を追加したとしても，その回路は複雑になる．また，LED は電球のように電圧を変化させて調光することもできず，順方向電圧以下になると電流が流れなくなり発光しなくなるため．暗い明るさの部分を表現することができない．

そこで，LED の調光には PWM 方式を用いる方法が一般的である．PWM 方式により，光が点滅しているように見えない程度の早さで信号が ON している時間を変化させることで，光っている時間を増減させる．この方法により，人間の目で見た時は残像によって明るさが変化しているように見える．ただし，この変化は目の視感度と LED の発光特性のため，単にパルス幅を直線的に変化させるだけは不自然であり，実際に調光して波形を変化させパルスの変化曲線を決める必要がある．

#### 3.3.2. DC モータの回転速度制御

DC モータの回転速度は，単位時間あたりの回転数で表される ((例) **rpm**：一分間あたりの回転数)．また，DC モータは電圧がかかっている間だけ回転している．そこで PWM 方式を用いることにより，LED の場合と同様に，回転が滑らかに見える程度の早さで信号が ON している時間を変化させることによって，実際に回転している時間を増減させる．このようにして，単位時間あたりの回転数を制御することができる．

## 4. 実験概要，使用装置

### 4.1. 実験概要

本実験では，マイコンボードである Arduino を用いて以下の実験を行う．

(1) PWM 信号による LED 制御 (調光)

- LED 制御では，PWM 信号をいろいろと変化させ，実際にパルス信号と LED の発光を確認しながら，LED の調光に関する実験を行う．

(2) パルス信号によるモータ制御

(2-1) PWM 信号による DC モータの回転速度制御

* LED の調光と同様に，PWM 信号をいろいろと変化させ，パルス信号と DC モータの回転速度の制御に関する実験を行う．

(2-2) パルス信号によるサーボモータの回転角度制御

* PWM 制御とは違うパルス信号により制御されるモータであるサーボモータについて，実際にパルス信号を確認しながらサーボモータの動作実験を行う．

### 4.2. 実験装置

- マイコンボード　Arduino Uno
- ブレッドボード
- ブレッドボート用配線ケーブル
- 発光ダイオード（LED)：弾丸型 10mm
- DC モータ (ソーラーモータ)
- サーボモータ (SG90)
- 抵抗（510 Ω）
- 可変抵抗（50K Ω）

## 5. PWM 信号による LED 制御（調光）

### 5.1. 実験１：LED の点滅

※ PWM 信号を用いた LED の調光を行う前に，LED の動作確認を行う．

Arduino 上にある 13 番のディジタル入出力ピンとつながっている LED を点滅させる回路（「インタラクティブなデバイスを作る（１）」の L チカ）を参考に，図 3 のようにさらに LED を 9 番ピンに接続し，双方の LED が交互に点滅するプログラムを作成する．点滅の間隔は前と同じく 500[ms] とする．

**※以降の実験のため，外部に接続する LED は 9 番ピンを使用する．**

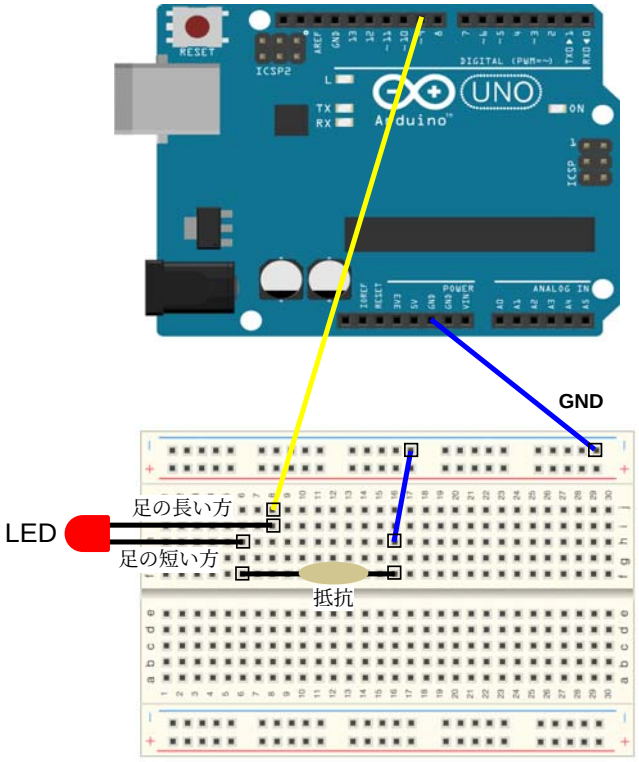



図 3: 接続図

(実験手順)

【1】図 3 のように Arduino，LED，抵抗等を接続

【2】Arduino IED の起動

【3】実験 1 のサンプルプログラムの記述

【4】コンパイル＆回路の転送

【5】動作確認

以下にサンプルプログラムを示す．Arduino 上の LED と外部に接続した LED が交互に 500[ms] で点滅することを確認する．

```
/**** 実験１：LED の点滅 - 0.5 秒毎に点灯／消灯する ****/
#define  LED1_PIN    9        // 外部 LED 接続ピン番号
#define  LED2_PIN    13       // オンボード LED ピン番号

void setup()
{
  pinMode(LED1_PIN, OUTPUT);
  pinMode(LED2_PIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED1_PIN, HIGH);
  digitalWrite(LED2_PIN, LOW);
  delay(500);                 // 500ms(0.5 秒) の遅延
  digitalWrite(LED1_PIN, LOW);
  digitalWrite(LED2_PIN, HIGH);
  delay(500);                 // 500ms(0.5 秒) の遅延
}
```

---

### 5.2. 実験２：LED の調光（その１） - 一定の時間間隔で明るさを変化させる

#### 5.2.1. 一定間隔で消灯→点灯を繰り返す

図 3 と同じように Arduino の 9 番ピンに LED を接続し，PWM 信号により接続された LED の明るさを図 4 のように一定間隔で変化させるプログラムを作成する．

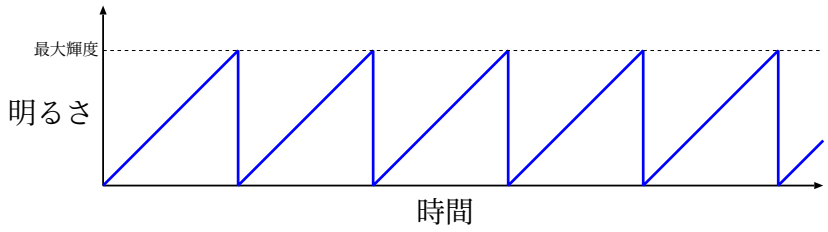



図 4: 経過時間と明るさ

**○ Arduino による PWM 信号の出力**

Arduino IED には，PWM 信号を生成・出力するために **analogWrite()** という関数が実装されている．

- **analogWrite(ピン番号，値)**
  - **ピン番号**：PWM 出力可能なピンの番号
  - **値**：デューティ比を指定する値 (0 〜 255)

この関数は，名前は **analogWrite()** となっているが，出力信号は D/A されたアナログ信号ではなく PWM 信号である．値は，0 でデューティ比が 0.0(全て OFF) であり，255 でデューティ比が 1.0(全て ON) となる．なお，Arduino では PWM 信号を出力できるピン番号が決まっており，Arduino 上のディジタル入出力側で　が記載されているピンが PWM 信号を出力できるピンである．

**※注）**：**analogRead()** 関数では，実際のアナログ信号の振幅を AD コンバータで 10bit のディジタル値に変換している．一方，**analogWrite()** 関数では，波形の振幅ではなく時間的にアナログ的に表されたディジタル値と考えることができる．

(実験手順)

【1】図 5 のように Arduino，LED，抵抗等を接続

【2】Arduino IED の起動

【3】実験 2 のサンプルプログラムを記述

【4】コンパイル＆回路の転送

【5】動作確認

この実験では，一定間隔で PWM 信号のデューティ比を変化することにより，LED の明るさとオシロスコープに表示されるパルス信号はどのように変化しているかを確認する．次ページに，LED の明るさを一定時間毎に変化させるためのサンプルプログラムと接続図をを示す．

```
/**** 実験２：LED の明るさを一定時間毎に変化させる ****/
#define  LED1_PIN    9        // 外部 LED 接続ピン番号
#define  DELAY_MS    4        // 遅延時間

#define  PWM_MIN     0        // デューティ比の最小値
#define  PWM_MAX     255      // デューティ比の最大値

void setup()
{
  pinMode(LED1_PIN, OUTPUT);
}

void loop()
{
  int i;
  for (i = PWM_MIN; i <= PWM_MAX; i++) {
    analogWrite(LED1_PIN, i);
    delay(DELAY_MS);
  }
}
```

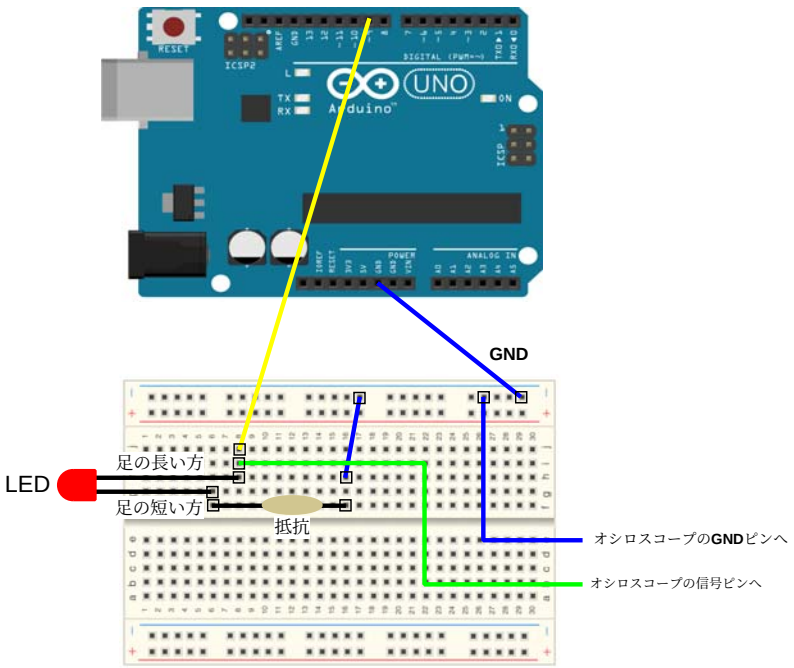



図 5: 接続図

### 5.2.2. 実験３：LED の調光（その２） - 一定間隔でなだらかに消灯→点灯→消灯を繰り返す

**（課題１）5.2.2 節と同様に，Arduino の 9 番ピンに接続された LED を図 6 のように明るさを一定間隔で消灯→点灯→消灯と繰り返すプグラムを作成する。消灯から点灯，点灯から消灯になるまでの時間間隔は各々1[s] とする．**

5.2.1 節のプログラムを参考に (課題１) のプログラムを作成し，5.2.1 節と同様に LED の明るさとオシロスコープに表示されるパルス信号はどのように変化しているかを確認する．プログラムは，消灯から点灯にするための for ループ文と，点灯から消灯にするための for ループ文に分けて記述すると作成しやすい．

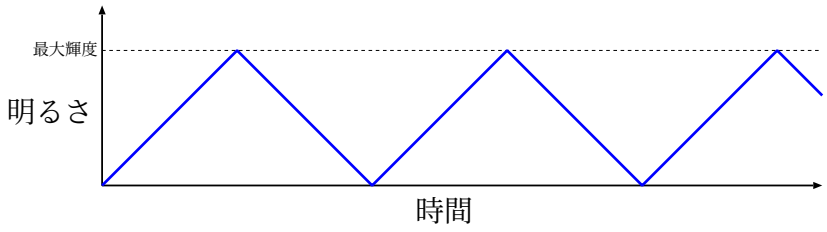



図 6: 経過時間と明るさ

(実験手順)

【1】図 5 のように Arduino，LED，抵抗等を接続

【2】Arduino IED の起動

【3】LED の明るさを一定間隔で変化させるプログラムを記述

【4】コンパイル＆回路の転送

【5】動作確認

作成するプログラムのサンプルを次ページに示す

```
/**** 実験３：（課題１）LED の明るさを一定間隔で消灯→点灯→消を繰り返す（未完成）****/
#define  LED1_PIN    9        // 外部 LED 接続ピン番号
#define  DELAY_MS    4        // 遅延時間

#define  PWM_MIN     0        // デューティ比の最小値
#define  PWM_MAX     255      // デューティ比の最大値

void setup()
{
  pinMode(LED1_PIN, OUTPUT);
}

void loop()
{
  int i;

  /*
   * (1) 徐々にパルス幅を広げ点灯させる 5.2.1 節と同じく i の値を徐々に大きくする.
   */

  /*
   * (2) 徐々にパルス幅を広げ点灯させる 5.2.1 節とは逆に i の値を徐々に小さくする.
   */
}
```

---

### 5.3. 実験４：LED の調光（その３） - 可変抵抗を使って明るさを制御する

**(課題２) 図 7 のように，今までの回路に可変抵抗を追加し，5.2 節のプログラムと「インタラクティブなデバイスを作る」の課題３を基に，LED の明るさを可変抵抗で調整できるようにする．**

5.2 節で作成したプログラムと「インタラクティブなデバイスを作る」の課題３のプログラムを参考に，可変抵抗による A/D 値 (analogRead() 関数で得られる値) を用いて LED の明るさを制御するプログラムを作成する．

(実験手順)

【1】図 7 のように Arduino，LED，抵抗，可変抵抗等を接続

【2】Arduino IED の起動

【3】LED の明るさを可変抵抗で調整できるようにするプログラムを記述

【4】コンパイル&回路の転送

【5】動作確認

5.2 節と同様に，LED の明るさとオシロスコープに表示されるパルス信号はどのように変化しているかを確認する．この時，可変抵抗の抵抗値をテスターで，また LED につながる出力信号の'0' と'1' のパルス幅（＝デューティ比）を測定できるようにし，抵抗値を変化させた時のパルス幅の変化を測定しその関係をグラフ化する．さらに，その時の明るさについても確認する．

**※ analogRead() の AD 値と analogWrite() のデューティ比の値の違い**

Arduino において analogRead() により得られる値は 0 〜 1023 (10bit) であるが，analogWrite() によって設定できるデューティ比の値は 0 〜 255 (8bit) である．そのため，analogRead() により得られた値をデューティ比として用いる場合には，analogRead() の値を 1/4 にする必要がある．

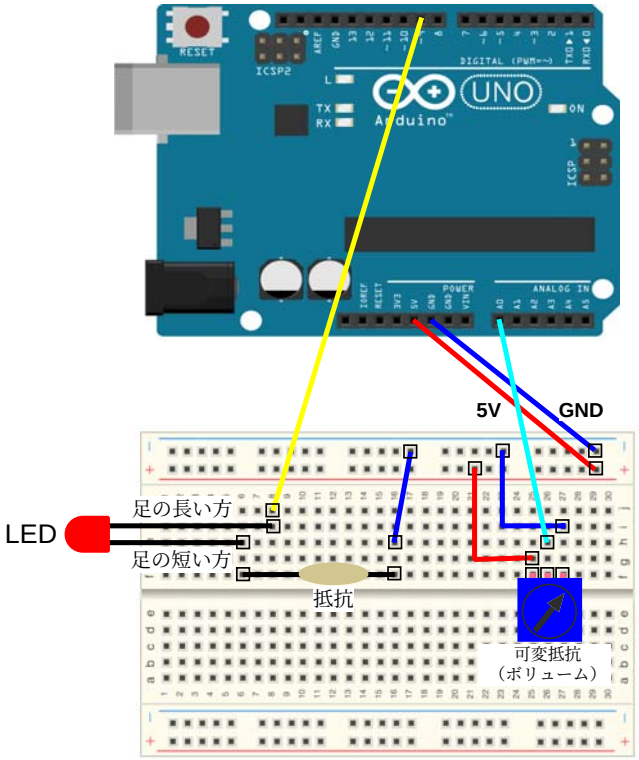



図 7: 接続図

# 6. モータ制御

## 6.1. PWM 信号による DC モータの回転速度の制御

### 6.1.1. 実験５：一定間隔で回転速度を変化させる

図 8 のように，図 3 の LED を抵抗に置き替えて DC モータを接続し，DC モータの回転速度を一定間隔で変化させる．

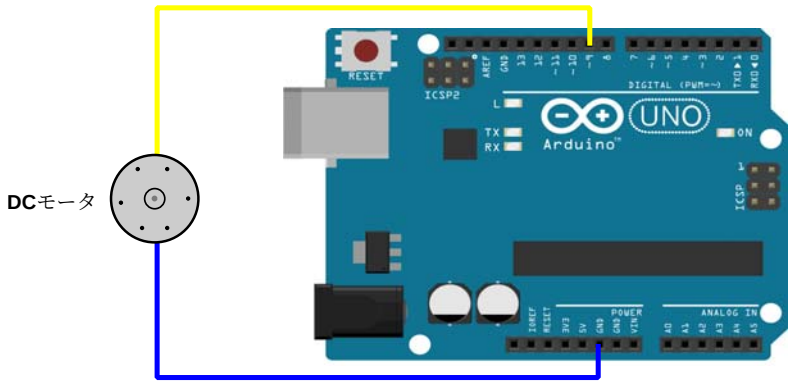



図 8: 接続図

DC モータは，LED の調光を同じ方法により回転速度を変化させることができる．そこで、この実験では，5.2.1 節，5.2.2 節で作成したプログラムを用い，実際に DC モータの回転速度も LED の明るさと同様に図 4，図 6 のような変化をするかどうかを確認する．その際，オシロスコープを用いてパルス幅と回転速度の関係についても確認する．

(実験手順)

【1】図 8 のように Arduino に DC モータを接続

【2】Arduino IED の起動

【3】5.2.1 節または 5.2.2 節で作成したプログラムを開く

【4】コンパイル＆回路の転送

【5】動作確認

動作確認は，5.2.1 節，5.2.2 節で作成したプログラム各々で行う．

### 6.1.2. 実験６：可変抵抗を使って回転速度を制御する

**(課題 3) 図 9 のように，DC モータと可変抵抗を接続し，DC モータの回転速度を可変抵抗で調整できるようにする．ただし，DC モータは可変抵抗の中間値で停止し，抵抗値が中間値より小さくなると右方向に徐々に回転し抵抗値が 0 で回転速度が最大に，また抵抗値が中間値より大きくなると左方向に徐々に回転し抵抗値が最大値で回転速度が最大になるようにする．**

DC モータには極性があり，加える電圧の向きによって回転方向が決まる．そのため，可変抵抗の抵抗値によって DC モータに加える電圧の向きを変えられる必要がある．このような DC モータの回転方向を変えるような回路では，通常モータドライバを用いることが多いが，この課題では Arduino にモータドライバの機能も持たせる．

そこで，図 9 のように DC モータの両端子を Arduino の PWM 信号が出力可能なピンに接続する．回転する際には，いずれか一方の端子を ’0’ にすることで GND に接続されている状態とし，もう一方の端子に PWM 信号を与える．

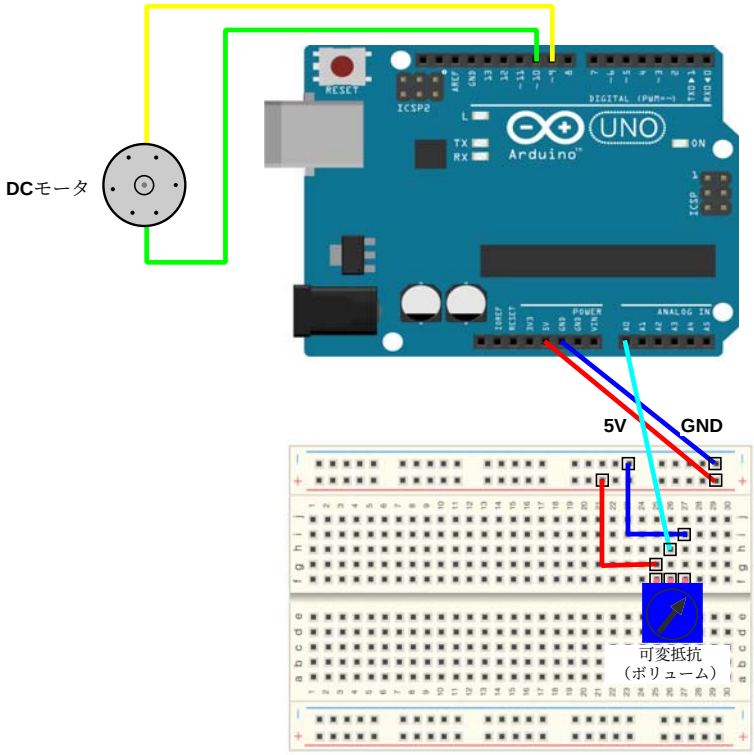



図 9: 接続図

```
/*
 * 実験６：可変抵抗を使って回転速度を制御する（未完成）
 */
#define  VR_PIN        A0      // 可変抵抗ピン
#define  MOTOR_PIN1    9       // DC モータ端子 1
#define  MOTOR_PIN2    10      // DC モータ端子 2

void setup()
{
  pinMode(VR_PIN,     INPUT);
  pinMode(MOTOR_PIN1, OUTPUT);
  pinMode(MOTOR_PIN2, OUTPUT);
}

void loop()
{
  int adc, pwm1, pwm2;

  /**** 可変抵抗値取り込み ****/
  adc = analogRead( 0 );

  /**** 各端子のデューティ比 ****/
  //
  // adc の値は 0〜1023 となるので、adc < 512 の場合は pwm1 = 0(GND)
  // にして pwm2 に 0〜255 のデューティ比の値を与える.
  // adc >= 512 の場合は，逆に pwm2 = 0(GND) にして pwm1 に 0〜255
  // のデューティ比の値を与える.
  //
  // adc が 511, 512 の時はデューティ比の値が 0 で，adc が 0 又は
  // 1023 の時には双方のの条件でデューティ比のが値が 255 となる.
  //
  if (adc < 512) {
    pwm1 = 0;
    pwm2 = <<<< 各自で考える >>>>;
  }
  else {
    pwm1 = <<<< 各自で考える >>>>;
    pwm2 = 0;
  }
  // PWM 信号出力
  analogWrite( 9, pwm1);
  analogWrite(10, pwm2);
}
```

(実験手順)

【1】図 9 のように Arduino に DC モータと可変抵抗を接続

【2】Arduino IED の起動

【3】サンプルを元にプログラムを作成する．

【4】コンパイル＆回路の転送

【5】動作確認

モータの両端子をオシロスコープに接続し，オシロスコープに表示されるパルス信号の変化と，その時のモータの回転状態（回転方向，回転速度）を確認する．

---

### 6.2. サーボモータの制御

#### 6.2.1. サーボモータの原理

サーボモータの概略を図 10 に示す．

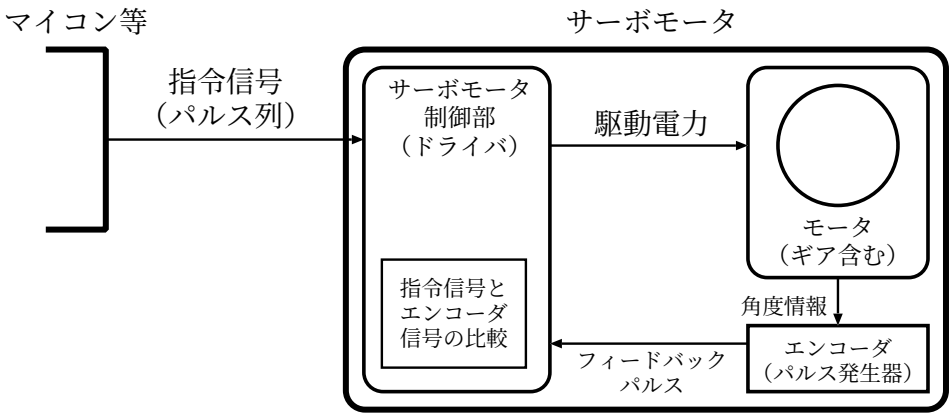



図 10: 概略図

サーボモータには，モータの位置決めをするための制御部が内蔵されている．制御部では，入力された指令信号となるパルス列と現在のモータ位置に応じてエンコーダで生成・フィードバックされたパルス列とを比較する．その結果，指令信号であるパルス信号の示す位置に移動するために現在位置から左右どちらに回転すればよいか決定し、それに応じてモータが回転する．

サーボモータの指令信号となる制御パルスは，図 11 のように一定間隔のパルス信号であり，そのパルス幅によりサーボモータの回転位置を指定する．例えば，図 11 のようにパルスの幅が 1.5msec のときは回転位置が中央位置になり，これよりパルス幅が広ければ右方向へ，また狭ければ左方向へ回転することになる．

#### 6.2.2. 実験７：一定間隔で回転角度を変化させる

Arduino には，サーボモータを制御するための関数が実装されている．そこで，この実験ではその関数を用い，サーボモータを図 12 のように Arduino に接続し，サーボモータを一定の時間間隔で左右に回転させるプログラムを作成する．

次ページに，サーボモータを回転させるためのサンプルプログラムを示す．

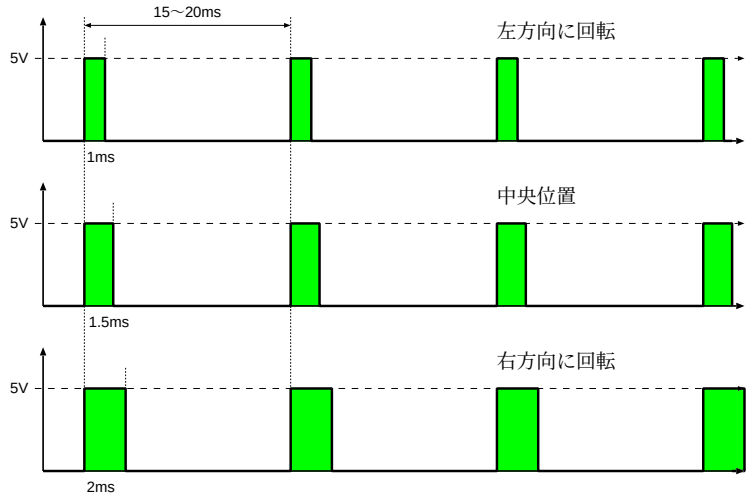



図 11: 制御パルス

```
/**** 実験７：サーボモータを一定時間で動かす ****/
#include <Servo.h>

#define  SERVO_PIN    2       // サーボモータ用信号ピン
#define  DELAY_MS     10      // 遅延時間

Servo myservo;

void setup()
{
  myservo.attach(SERVO_PIN);
  Serial.begin(9600);
}

void loop()
{
  int angle = 0;  // 回転角
  int value;

  /**** 0 度 ～ 180 度まで動かす ****/
  for (angle = 0; angle <= 180; angle++) {
    myservo.write(angle);
    value = myservo.read(); // サーボモータから読み取った現在の回転角
    Serial.println(value);
    delay(DELAY_MS);
  }
  /**** 180 度 ～ 0 度まで動かす ****/
  for (angle = 180; angle >= 0; angle--) {
    myservo.write(angle);
    value = myservo.read();
    Serial.println(value);
    delay(DELAY_MS);
  }
}
```

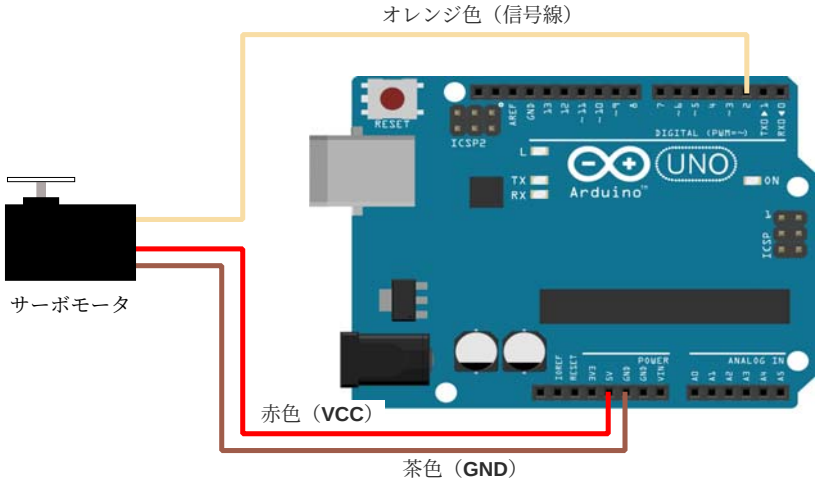



図 12: 接続図

(実験手順)

【1】図 12 のように Arduino にサーボモータを接続

【2】Arduino IED の起動

【3】サンプルプログラムの記述

【4】コンパイル&回路の転送

【5】動作確認

サンプルプログラムでは，サーボモータから読み取った現在の回転角度が Arduino IDE のシリアルモニタに表示できるようになっている．そこで，動作確認では，Arduino のサーボモータ接続ピン側にオシロスコープも接続し，パルス信号の幅とサーボモータの回転角度，またシリアルモニタに表示される値がどのようになっているかを確認する．オシロスコープを接続する際には，ブレッドボードを使用する．

### 6.2.3. 実験８：可変抵抗を使って回転角度を制御する

**(課題 4) 図 13 のように，サーボモータと可変抵抗を接続し，サーボモータの回転速度を可変抵抗で調整できるようにする．**

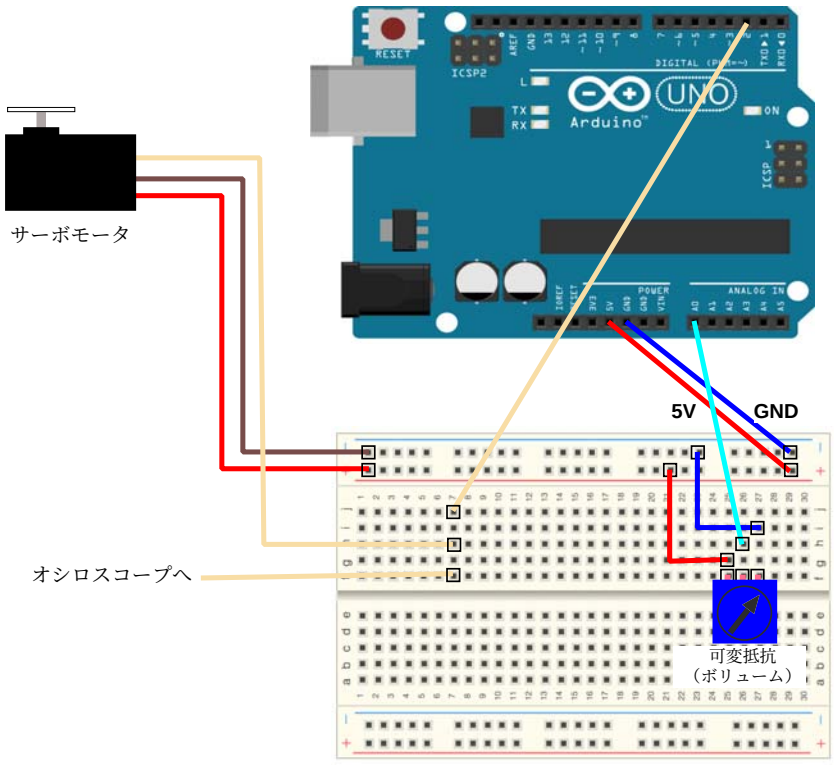



図 13: 接続図

(実験手順)

【1】図 9 のように Arduino にサーボモータと可変抵抗を接続

【2】Arduino IED の起動

【3】サンプルを元にプログラムを作成する．

【4】コンパイル＆回路の転送

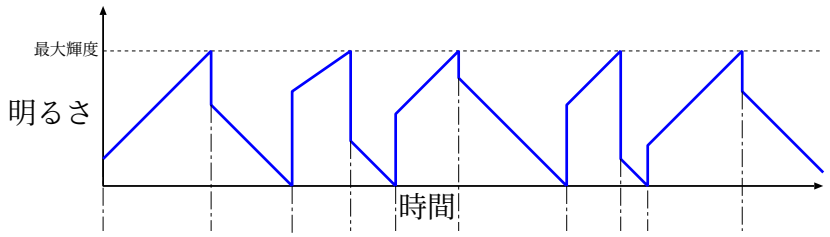
【5】動作確認

Arduino のパルス出力側にオシロスコープを接続し，オシロスコープに表示されるパルス信号の幅振の変化と，その時のモータの回転角度を確認する．

## 7. 検討課題

### ○ろうそくの炎のように LED を点灯させるプログラムを作成せよ．

5.2.2 節のプログラムを参考に，消灯から点灯，点灯から消灯になる際の開始値をランダムに設定することにより，図 14 のように明るさを変化させることができ，キャンドルライトのような揺らめいた明かりを実現できる．
(ヒント)for 文のループの開始値を rand() 関数で設定できるようにする。




上昇時、下降時各々のループの開始値を
ランダムに設定する．

図 14: 経過時間と明るさ