

J4 情報システム実験実習 II 実験報告書

題目 DC モータの PWM 制御と速度フィードバック制御の基礎

実施年月日 2025 年 7 月 01 日 天候 晴れ 温度 23.6 °C 湿度 37 %

2025 年 7 月 11 日 天候 晴れ 温度 24.2 °C 湿度 69 %

提出年月日 2025 年 7 月 17 日

共同実験者 06 班

宮本 嘉本 長尾 見山 アヌ

提出者

通し番号21 学籍番号22059 氏名 来間 空

1. 目的

省略

1. 使用機器

省略

2. 実験方法

省略

3. 実験結果

実験 3.1 3)のプログラムと、動作内容を記載したコメントを

3.1. 図 1 に示す。

```
ledpwm2.py
#初期状態は青色に光り、ボタンを押し続けていると赤色に光るプログラム
import wiringpi as pi
import time

LED_PIN_1 = 23 #赤色LED
LED_PIN_2 = 24 #青色LED
SW_PIN = 17 #ボタン

pi.wiringPiSetupGpio()

#LEDピンを出力モードに設定
pi.pinMode( LED_PIN_1, pi.OUTPUT )
pi.pinMode( LED_PIN_2, pi.OUTPUT )

#ボタン用ピンを入力用に設定
pi.pinMode( SW_PIN, pi.INPUT )

#PWMを0~100%に設定
pi.softPwmCreate( LED_PIN_1, 0, 100 )
pi.softPwmCreate( LED_PIN_2, 0, 100 )

while True:
    time.sleep(1)

    #初期状態は消灯
    pi.digitalWrite( LED_PIN_1, pi.LOW)
    pi.digitalWrite( LED_PIN_2, pi.LOW)

    if(pi.digitalRead( SW_PIN ) == pi.HIGH ):#ボタンが押されたら
        value = 0
        #徐々に明るさを上げながら赤色に光る
        while( value < 100):
            pi.softPwmWrite( LED_PIN_1, value )
            pi.softPwmWrite( LED_PIN_2, 0)
            time.sleep(0.01)#0.01秒間隔
            value = value + 1
        #押されてないときは
    else
        value = 0
        #徐々に明るさを上げながら青色に光る
        while( value < 100):
            pi.softPwmWrite( LED_PIN_1, 0 )
            pi.softPwmWrite( LED_PIN_2, value )
            time.sleep(0.01)#0.01秒間隔
            value = value + 1
```

図 1 実験 3.1 3)のプログラムおよびコメント

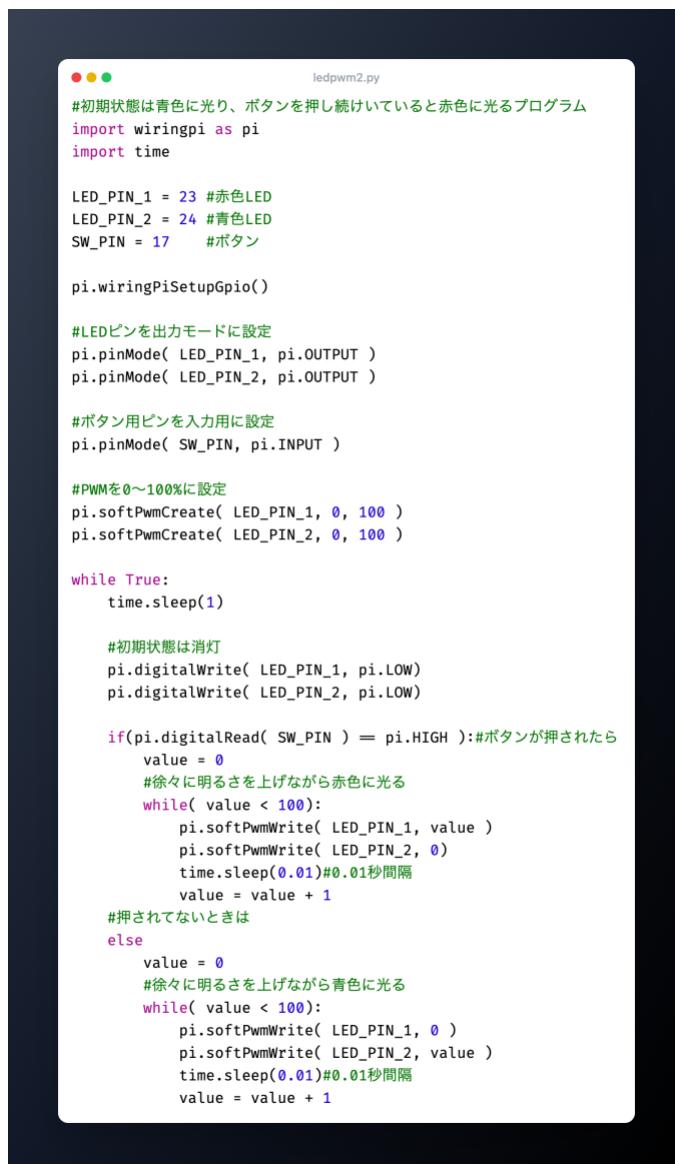
図 1 の動作内容を以下に示す。

- プログラムを実行したとき、LED の初期状態は LED1, LED2 どちらも消灯となる。
- スイッチを押している間
 1. 次の動作を Duty 比 0~99 まで繰り返す。 (Duty 比を表す変数を value として初期化する。)
 - 1.1. LED1, すなわち赤色 LED を value のぶんだけ光らせる。 LED2 は消灯する。
 - 1.2. 0.01 秒間隔で、 value をインクリメントする。つまり、 duty 比が 1 増加したことであり、明るさが 1% 増加したことと同値である。
 2. Duty 比 99 から 100 へインクリメントされた瞬間、内側の while ループを抜け出し、すぐ外側の無限ループ先頭に戻る。
 3. LED1 は duty 比 100 で 1 秒間点灯状態を保持する。
 4. 2 に戻る。

- スイッチを離している間
 1. スイッチを離している間は次の動作を Duty 比 0~99 まで繰り返す。この時 value は初期化される。
 - 1.1. LED2, すなわち青色 LED を value のぶんだけ光らせる。LED1 は消灯する。
 - 1.2. 0.01 秒間隔で、value をインクリメントする。つまり、duty 比が 1 増加したということであり、明るさが 1% 増加したことと同値である。
 2. Duty 比 99 から 100 へインクリメントされた瞬間、内側の while ループを抜け出し、すぐ外側の無限ループ先頭に戻る。
 3. LED2 は duty 比 100 で 1 秒間点灯状態を保持する。
 4. 2 に戻る。

このように、図 1 のプログラムは、ボタンの状態に応じて赤または青の LED を徐々にフェードインさせるプログラムである。

3.2. 実験 3.2.3) で作成したプログラムおよびプログラムの詳細を記載したコメントを図 2 に示す。



```

● ● ●
ledpwm2.py
#初期状態は青色に光り、ボタンを押し続けていると赤色に光るプログラム
import wiringpi as pi
import time

LED_PIN_1 = 23 #赤色LED
LED_PIN_2 = 24 #青色LED
SW_PIN = 17    #ボタン

pi.wiringPiSetupGpio()

#LEDピンを出力モードに設定
pi.pinMode( LED_PIN_1, pi.OUTPUT )
pi.pinMode( LED_PIN_2, pi.OUTPUT )

#ボタン用ピンを入力用に設定
pi.pinMode( SW_PIN, pi.INPUT )

#PWMを0~100%に設定
pi.softPwmCreate( LED_PIN_1, 0, 100 )
pi.softPwmCreate( LED_PIN_2, 0, 100 )

while True:
    time.sleep(1)

    #初期状態は消灯
    pi.digitalWrite( LED_PIN_1, pi.LOW)
    pi.digitalWrite( LED_PIN_2, pi.LOW)

    if(pi.digitalRead( SW_PIN ) == pi.HIGH):#ボタンが押されたら
        value = 0
        #徐々に明るさを上げながら赤色に光る
        while( value < 100):
            pi.softPwmWrite( LED_PIN_1, value )
            pi.softPwmWrite( LED_PIN_2, 0)
            time.sleep(0.01)#0.01秒間隔
            value = value + 1
        #押されてないときは
    else
        value = 0
        #徐々に明るさを上げながら青色に光る
        while( value < 100):
            pi.softPwmWrite( LED_PIN_1, 0 )
            pi.softPwmWrite( LED_PIN_2, value )
            time.sleep(0.01)#0.01秒間隔
            value = value + 1

```

図 2 実験 3.2 3) のプログラムおよびコメント

3.3. 実験 3.3 2) で計測した電圧値をグラフ化したものと図 3 に示す。

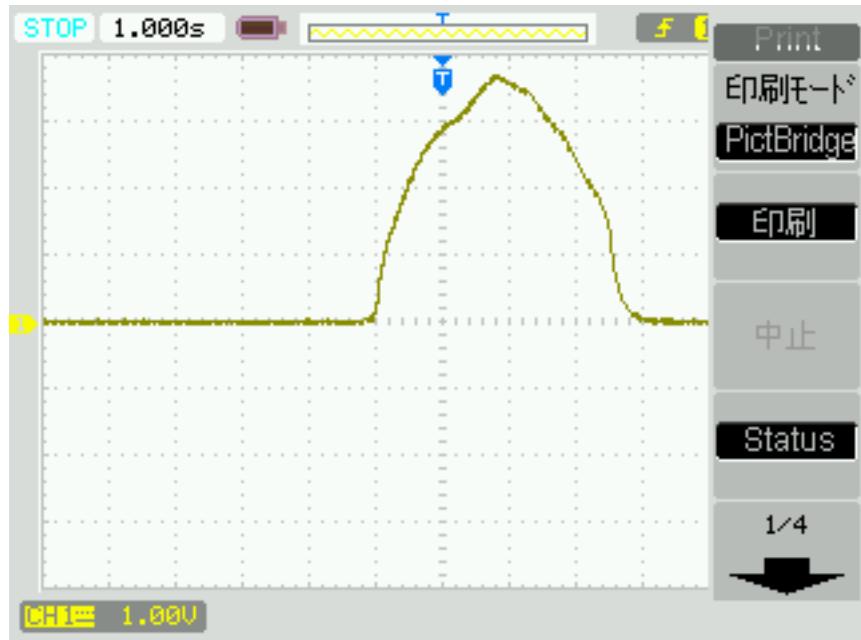


図 3 実験 3.3 2) で計測した電圧値

4. 考察と調査

4.1. 速度フィードバック制御型のアルゴリズム

速度フィードバック制御系のアルゴリズムのフローチャートを図 4 に示す。

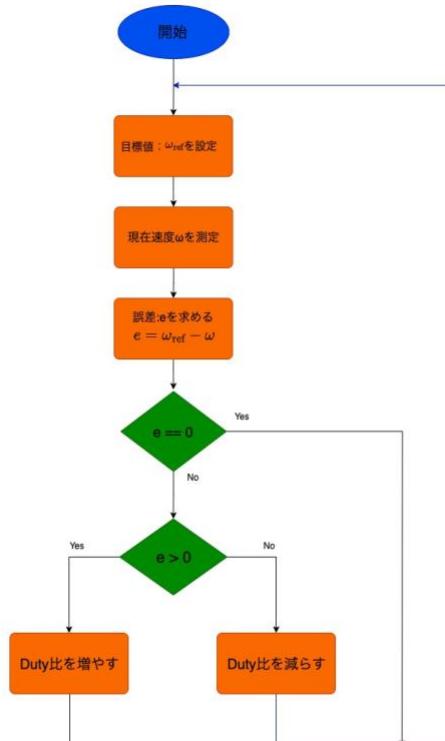


図 4 速度フィードバック制御系のアルゴリズムのフローチャート

図 4 より、誤差： e が 0 に収束し、目標値： ω_{ref} を維持することができると考えられる[1].

4.2. モータドライバ回路を構成する H ブリッジ回路の動作

H ブリッジ回路とは、4 つのスイッチを制御することで、モータを「正転」、「逆転」、「ブレーキ」、「停止」状態に切替えることができる回路である[2]. 正転はモータを回転させることで、逆転はその正転とは逆方向にモータを回転させることである. ブレーキとはモータを素早く停止させ、物理的な負荷が加わっても回転しにくいという特徴がある. 一方で、停止はモータを停止させることで、物理的な負荷が加わると回転しやすい. H ブリッジ回路の動作を図 5 に示す.

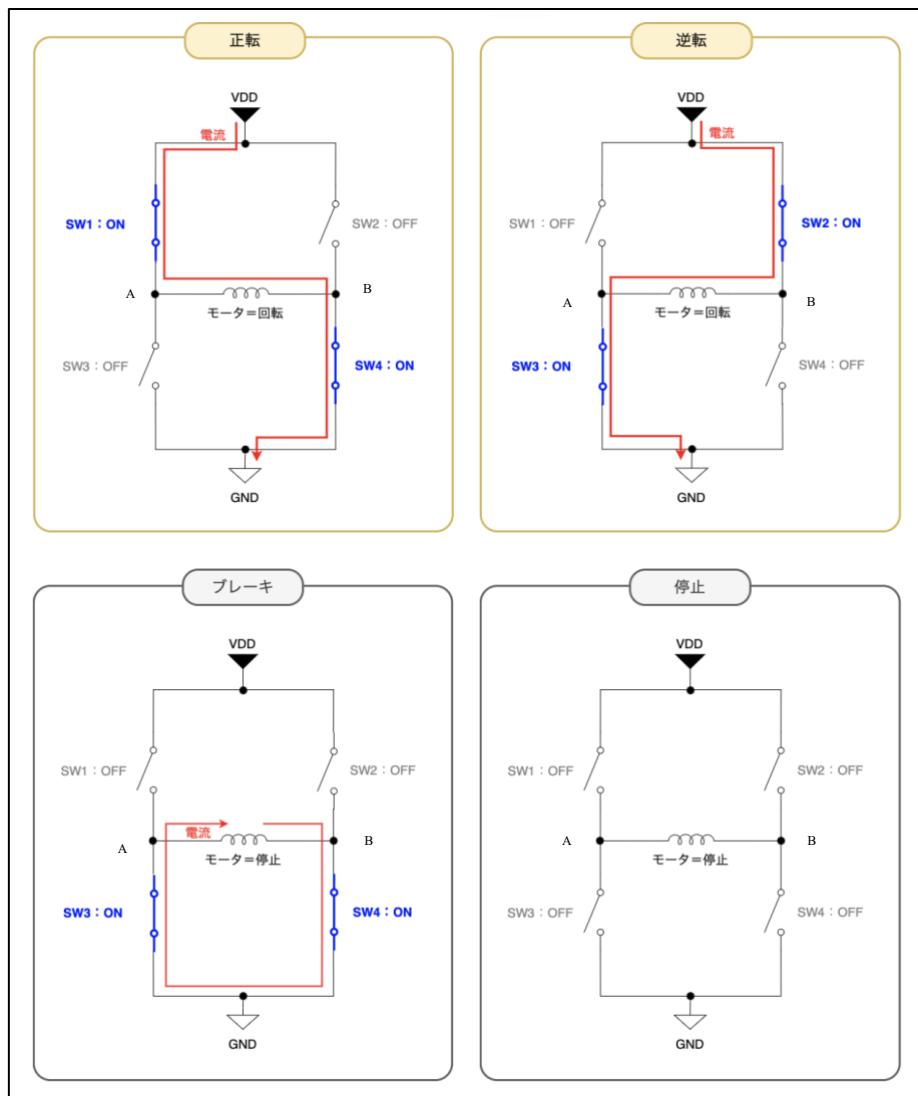


図 5 H ブリッジ回路の動作[3]

図 5 より、モータの両端の端子を A,B とし、スイッチの左上を sw1、右上を sw2、左下を sw3、右下を sw4 とする. よって、DC モータが動作する際のスイッチ状態および電流の方向を表 1 に示す.

表 1 H ブリッジ回路のスイッチの状態と電流方向

動作	sw1	sw2	sw3	sw4	電流方向
正転	ON	OFF	OFF	ON	A→B
逆転	OFF	ON	ON	OFF	B→A
ブレーキ	OFF	OFF	ON	ON	双方向循環電流
停止	OFF	OFF	OFF	OFF	なし

図 5, 表 1 より, 正転, 逆転, 停止, ブレーキ動作するための条件を以下に説明する.

正転は端子 A が+側, B が-側になる時であり, つまり sw1 と sw4 が ON になった時である.

逆転は正転と逆方向の回転を加えれば良いため, 端子 B が+側, A が-側になる時である. よって, sw2 と sw3 を ON にする.

ブレーキは端子 A,B はどちらも GND に接続され端子 A = B (短絡状態) となる. この時, モータに流れていた電流が一気に GND 奪われてしまうため, 回転エネルギーが抵抗分に消費され急減速する. つまり, sw3 と sw4 を ON にする.

停止は, モータ端子はどこにも接続されず開放されるので, 慣性で徐々に減速する. よって, 4 端子全て OFF にする.

これにより, 4 素子の組合せを正しく切り替えることで, DC モータの正転, 逆転, 停止, ブレーキの各モードを安全に実現できる.

5. 感想

1 週目の実験では, Raspberry Pi による LED の点灯制御を行った. Raspberry Pi のピン接続は指示通り行ったため容易であった. 一方で, PWM 制御による LED の明るさを変化させるプログラムでは, 自分たちでプログラムを考える必要があったため, 少し苦戦した. 2 週目では, Raspberry Pi とモータドライバ IC を接続し, PWM 制御による実験を行った. Duty 比の概念をモータの動きと実験 3.3.2)で計測したグラフを見ることによって, より深い理解を得ることができた. 1 週目の内容を応用したので, モーターを動かすプログラムを作るのは容易であった. また, この実験全体を通して, モータの正転や逆転などの動作原理を理解することができた. 速度フィードバック制御に関しても, フローチャート作成を通して, PID 制御などの他のモータ制御方法にも興味を持てたため, 非常に学びのある実験であった.

参考文献

- [1] 熊谷正朗, 「制御の基礎」,
https://www.mech.tohoku-gakuin.ac.jp/rde/contents/sendai/mechatro/archive/RMSeminar_No09.pdf, 東北学院大学工学部ロボット開発工学研究室, 2025 年 7 月 16 日参照.
- [2] 「H ブリッジ」, <https://tohan-denshi.co.jp/technical-information/8244/>, 東阪電子機器株式会社, 2025 年 6 月 2 日更新.
- [3] 「【モータ駆動回路】H ブリッジ回路の動作原理について理解する」,
<https://practicalelectronicsblog.com/hbridge/>, 2024 年 8 月 2 日更新.