

## 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の基礎

実施年月日 2024 年 7 月 19 日 天候 晴 温度 26.7°C 湿度 72%

2024 年 7 月 26 日 天候 晴 温度 27.5°C 湿度 66%

提出年月日 2024 年 8 月 9 日

共同実験者 4 班

<u>赤木</u>	<u>湯原</u>	<u>金塚</u>	<u>小原</u>
<u>鈴木</u>	<u>中川</u>	<u>比留田</u>	<u>森</u>

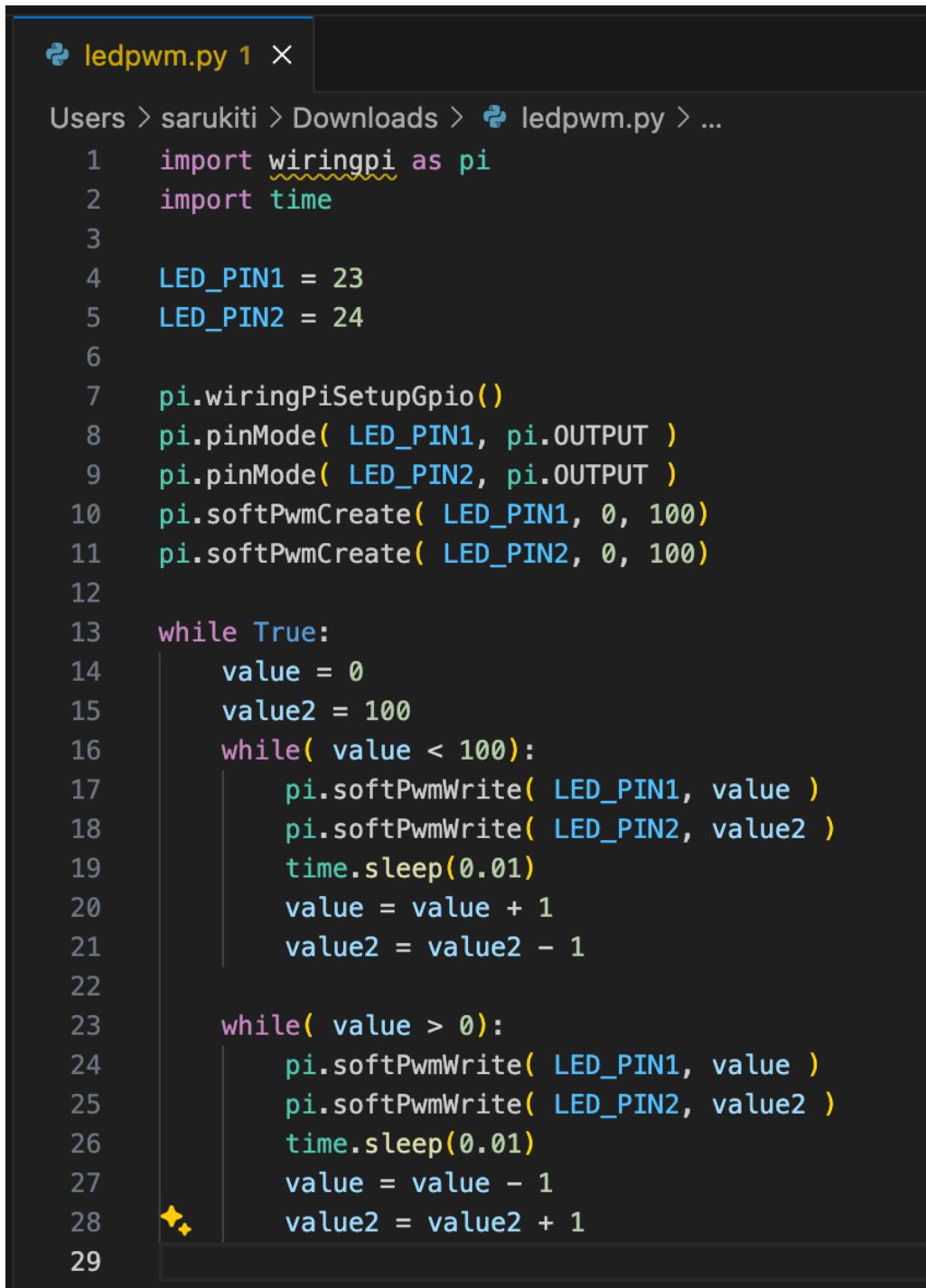
提出者

学籍番号 21035 氏名 江谷 空

## 4. 実験結果

### 4.1. 実験 3.1 3)の実行結果

実行したプログラムを図 1 に示す。このプログラムを実行している間、LED の明るさはゼロから最大、最大からゼロに段階的に変化するのを繰り返す。明るさを増加させる局面ではデューティー比を 0.01 秒毎に 1 ずつ増加させている。また、明るさを減少させる局面ではデューティー比を 0.01 秒毎に 1 ずつ減少させている。



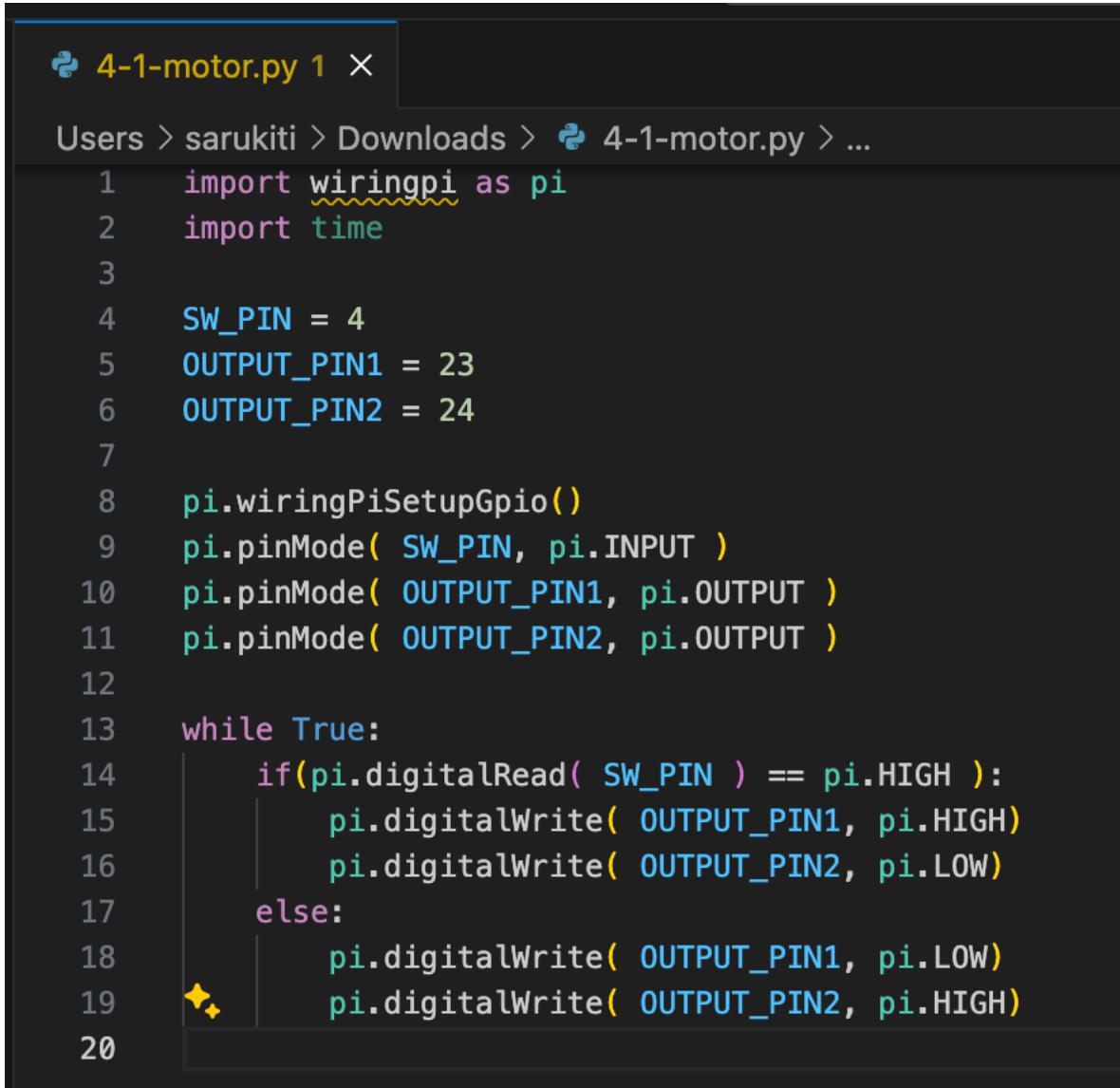
```
ledpwm.py 1 ×  
Users > sarukiti > Downloads > ledpwm.py > ...  
1 import wiringpi as pi  
2 import time  
3  
4 LED_PIN1 = 23  
5 LED_PIN2 = 24  
6  
7 pi.wiringPiSetupGpio()  
8 pi.pinMode( LED_PIN1, pi.OUTPUT )  
9 pi.pinMode( LED_PIN2, pi.OUTPUT )  
10 pi.softPwmCreate( LED_PIN1, 0, 100)  
11 pi.softPwmCreate( LED_PIN2, 0, 100)  
12  
13 while True:  
14     value = 0  
15     value2 = 100  
16     while( value < 100):  
17         pi.softPwmWrite( LED_PIN1, value )  
18         pi.softPwmWrite( LED_PIN2, value2 )  
19         time.sleep(0.01)  
20         value = value + 1  
21         value2 = value2 - 1  
22  
23     while( value > 0):  
24         pi.softPwmWrite( LED_PIN1, value )  
25         pi.softPwmWrite( LED_PIN2, value2 )  
26         time.sleep(0.01)  
27         value = value - 1  
28         value2 = value2 + 1  
29
```

図 1 RaspberryPi 上で Python を使って PWM 信号を出力し、LED の明るさを制御するプログラム

## 4.2. PWM 制御による DC モーターの回転制御

### 4.2.1. 実験 3.2 1) の実験結果

使用したプログラムを図 2 に示す。このプログラムを実行している時、タクトスイッチを押している場合はモータが正転し、そうでない場合はモータが逆転する。

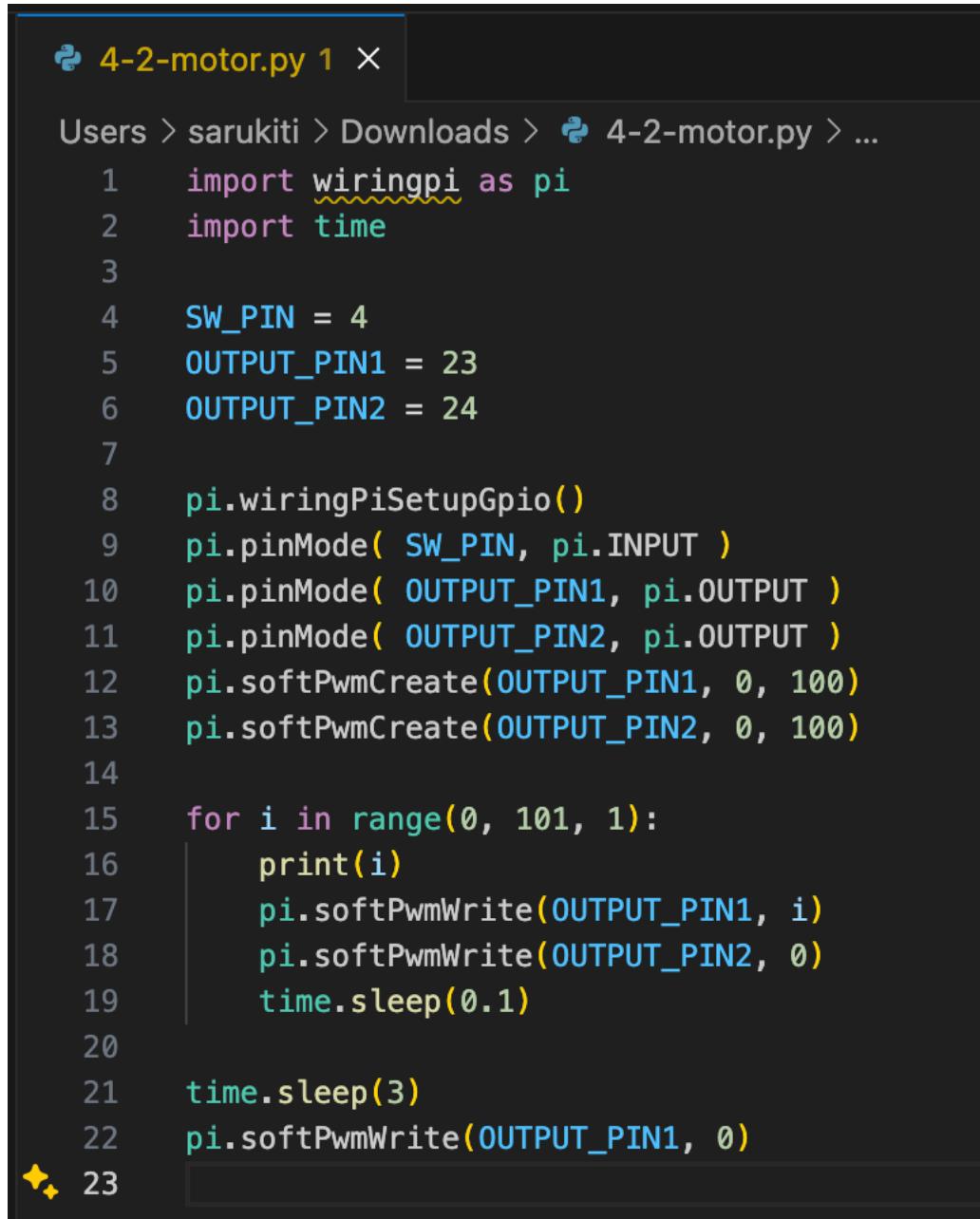


```
4-1-motor.py 1 ×
Users > sarukiti > Downloads > 4-1-motor.py > ...
1 import wiringpi as pi
2 import time
3
4 SW_PIN = 4
5 OUTPUT_PIN1 = 23
6 OUTPUT_PIN2 = 24
7
8 pi.wiringPiSetupGpio()
9 pi.pinMode( SW_PIN, pi.INPUT )
10 pi.pinMode( OUTPUT_PIN1, pi.OUTPUT )
11 pi.pinMode( OUTPUT_PIN2, pi.OUTPUT )
12
13 while True:
14     if(pi.digitalRead( SW_PIN ) == pi.HIGH ):
15         pi.digitalWrite( OUTPUT_PIN1, pi.HIGH)
16         pi.digitalWrite( OUTPUT_PIN2, pi.LOW)
17     else:
18         pi.digitalWrite( OUTPUT_PIN1, pi.LOW)
19         pi.digitalWrite( OUTPUT_PIN2, pi.HIGH)
20
```

図 2 タクトスイッチによってモータの回転方向を変更するプログラム

#### 4.2.2. 実験 3.2 2)の実験結果

使用したプログラムを図 3 に示す。これはデューティー比を 0%から 100%まで変化させ、3 秒後にデューティー比をゼロに戻すプログラムである。これを実行すると、モータの回転数は徐々に増加し、3 秒間フル回転して停止した。



```
4-2-motor.py 1 ×  
Users > sarukiti > Downloads > 4-2-motor.py > ...  
1 import wiringpi as pi  
2 import time  
3  
4 SW_PIN = 4  
5 OUTPUT_PIN1 = 23  
6 OUTPUT_PIN2 = 24  
7  
8 pi.wiringPiSetupGpio()  
9 pi.pinMode( SW_PIN, pi.INPUT )  
10 pi.pinMode( OUTPUT_PIN1, pi.OUTPUT )  
11 pi.pinMode( OUTPUT_PIN2, pi.OUTPUT )  
12 pi.softPwmCreate(OUTPUT_PIN1, 0, 100)  
13 pi.softPwmCreate(OUTPUT_PIN2, 0, 100)  
14  
15 for i in range(0, 101, 1):  
16     print(i)  
17     pi.softPwmWrite(OUTPUT_PIN1, i)  
18     pi.softPwmWrite(OUTPUT_PIN2, 0)  
19     time.sleep(0.1)  
20  
21 time.sleep(3)  
22 pi.softPwmWrite(OUTPUT_PIN1, 0)  
23
```

図 3 デューティー比を 0 から 100 まで増加させた後、3 秒後に 0 に戻すプログラム

#### 4.2.3. 実験 3.2 3)の実験結果

使用したプログラムを図 4 に示す。これはデューティー比を 0%から 100%まで 1 秒に 10%ずつ変化させ、3 秒待った後 100%から 0%まで 1 秒に 10%ずつ戻すプログラムである。これを実行すると、モータの回転数は 1 秒毎に増加し、3 秒間フル回転した後、1 秒毎に減少し、最後には停止した。

```
4-3-motor.py 1 ×  
Users > sarukiti > Downloads > 4-3-motor.py > ...  
1 import wiringpi as pi  
2 import time  
3  
4 SW_PIN = 4  
5 OUTPUT_PIN1 = 23  
6 OUTPUT_PIN2 = 24  
7  
8 pi.wiringPiSetupGpio()  
9 pi.pinMode( SW_PIN, pi.INPUT )  
10 pi.pinMode( OUTPUT_PIN1, pi.OUTPUT )  
11 pi.pinMode( OUTPUT_PIN2, pi.OUTPUT )  
12 pi.softPwmCreate(OUTPUT_PIN1, 0, 100)  
13 pi.softPwmCreate(OUTPUT_PIN2, 0, 100)  
14  
15 for i in range(0, 110, 10):  
16     print(i)  
17     pi.softPwmWrite(OUTPUT_PIN1, i)  
18     pi.softPwmWrite(OUTPUT_PIN2, 0)  
19     time.sleep(1)  
20  
21 time.sleep(2)  
22  
23 for i in range(90, -10, -10):  
24     print(i)  
25     pi.softPwmWrite(OUTPUT_PIN1, i)  
26     pi.softPwmWrite(OUTPUT_PIN2, 0)  
27     time.sleep(1)  
28
```

図 4 モータの回転数を増減させるプログラム

## 5. 考察

### 5.1. 速度フィードバック制御系を構成するためのアルゴリズムをフローチャートで図5に示す。

まず、目標の速度  $r$  と現在の速度  $y$  が入力される。これを元に誤差  $e=r-y$  を計算し、 $e$  がゼロであった場合、現在のデューティー比を維持する。 $e$  がゼロでなかった場合は、 $e>0$  であればデューティー比を現在より大きく、 $e<0$  であればデューティー比を現在より小さくする。その後は、現在の速度  $y$  を受け取り、誤差を計算して、それを小さくする方向にデューティー比を動かす処理をひたすら続けることで、目標の速度  $r$  に近い速度でモータを回す処理を実現できる。

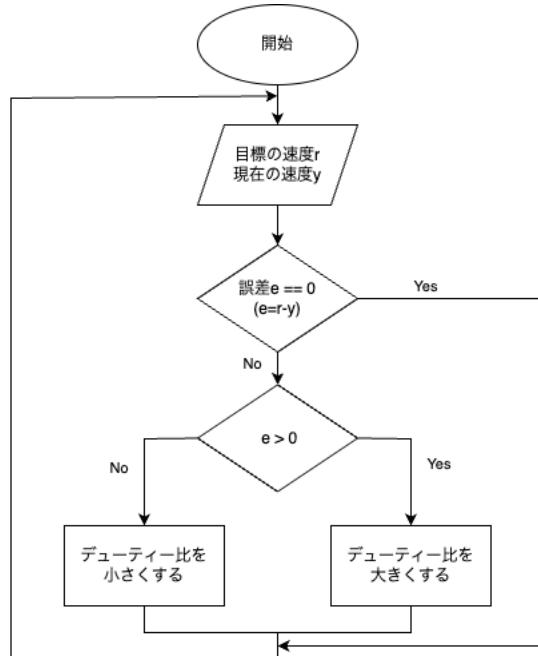


図 5 速度フィードバック系を構成するアルゴリズムのフローチャート

### 5.2. ブラシ付 DC モータの接続を変更するための H ブリッジ回路を以下の図 6 に示す。

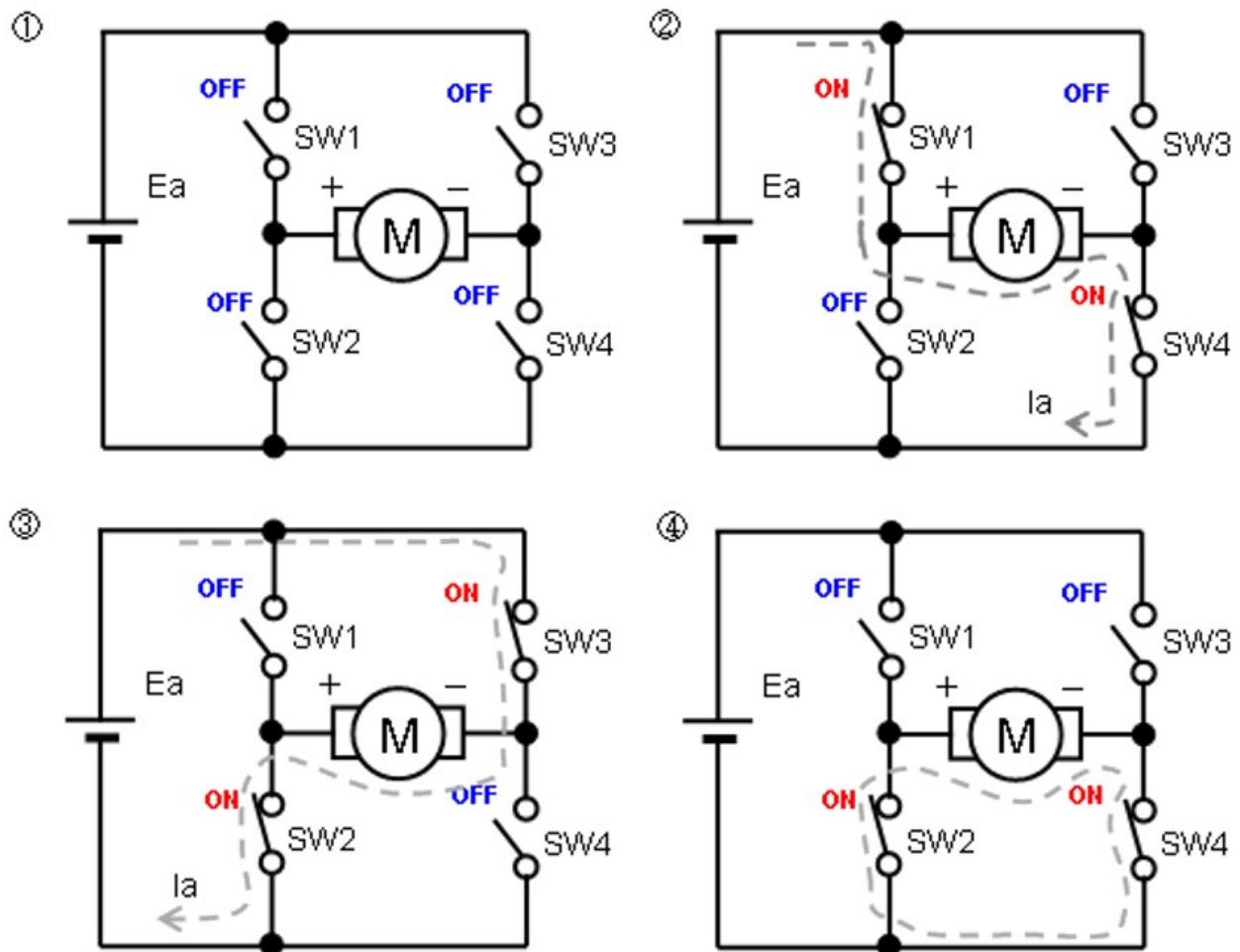
ブラシ付 DC モータには 2 本の電源端子があり、この接続を変更することで 4 種類の動作を行わせることができる。

図 6 の①のように、H ブリッジ回路を構成するスイッチの 4 つ全てを開放した場合、モータの端子は 2 本ともどこにも接続されない状態となる。この場合、モータの回転数は自然に減少していく、やがて停止する。

②のように、スイッチの左上と右下を短絡させた場合、モータの左側端子は電源ラインに、右側端子は GND に接続される。この場合、モータに電流が流れ、モータが正転する。

③のように、スイッチの右上と左下を短絡させると、モータの左側端子が GND に、右側端子が電源ラインに接続される。この場合、モータに流れる電流は②と逆になり、モータが逆転する。

④のように、左下と右下のスイッチを短絡させると、モータの端子は 2 本とも GND に接続される。この場合、モータに流れていた電流が一気に GND に奪われてしまうため、モータは急停止する。これをショートブレーキという。



## ブラシ付DCモータ 接続変更用回路

図 6 ブラシ付 DC モータ 接続変更用回路[1]

### 6. 感想

WiringPi を用いて RaspberryPi の GPIO を制御するのは初めてであったが、他のライブラリと同様に簡単に使用することができた。Python 版 WiringPi は最早サポートされていない状態となっているため、gpiozero 等の他のライブラリを使用して実験を行うのが適切だと思う。

### 参考文献

- [1] 「H ブリッジ回路によるブラシ付 DC モーターの駆動:原理」, TechWeb Powerd by ROHM,  
<https://techweb.rohm.co.jp/product/motor/motor-types/260/>, 2024 年 8 月 9 日参照。