

実験実施日 2024 年 10 月 10 日, 12 日

コンピュータ科学実験b

ハードウェア Raspberry Pi による制御

学生番号: 102210017
氏名: 安藤駿

1 はじめに

これまで様々なモノは個別に役割を果たしていたが、モノにセンサやアクチュエータを搭載し、インターネットに接続可能とすることで、現実世界の様々な情報収集、モノの遠隔制御、モノ同士の相互作用が可能となる。この考え方が IoT (Internet of Things) である。Raspberry Pi を用いて IoT のデバイス管理を体験、学習する。Raspberry Pi を用いた LED、スイッチ、ステッピングモータの制御を行う。

2 課題 2-1 単一 LED (Light Emitting Diode) の点滅

2.1 目的・概要

Raspberry Pi を用いて LED1 個の点滅を行う。Raspberry Pi と LED を接続し、Raspberry Pi 上でコードを実行して LED の点滅制御を行う。

2.2 必要な部品

2.2.1 Raspberry Pi とその他部品

- Raspberry Pi 3 model B+
- Micro SD カード (Raspberry Pi OS 書き込み済み)
- HDMI ケーブル
- Micro USB 電源ケーブル
- モニタ

実験室のモニタと自宅のモニタを使用した。

- USB キーボード

使用する製品は、SANWA 社の SKB-KG3WN である。シリアル番号は、20200500196 である。

- USB マウス

使用する製品は、ELECOM 社の M-K6URBK/RS である。シリアル番号は、1203314C である。

2.2.2 ブレッドボード

ブレッドボードは電子回路の試作に使用される。電子部品の端子を差し込む穴が多数存在し、それぞれの穴に差し込んだ端子が接続される。この接続関係を考慮して部品を配置することで、はんだ付けを行うことなく電子回路を組むことができる。

2.2.3 GPIO ブレッドボード接続ケーブル

Raspberry Pi の GPIO をブレッドボードへ接続することで、部品の配線が容易となる。

2.2.4 抵抗内蔵型 LED

長足側に抵抗が内蔵されている。

2.3 実験方法

2.3.1 Raspberry Pi の起動

Raspberry Pi と液晶ディスプレイを HDMI ケーブルで接続した。Raspberry Pi の USB 端子にキーボードとマウスを接続した。Micro SD カードスロットに Micro SD カードを挿入した。その後、Micro USB 電源ケーブルを Raspberry Pi に接続した。このときの状況を、図 1 に示す。



図 1 ケーブル類を接続した Raspberry Pi

2.3.2 OS のインストールと初期設定

country を Japan, time を Tokyo に設定した。username と password を設定した。その後、実験室内的 wifi に接続した。また、ソフトウェアのアップデートは行わなかった。ブラウザに firefox を選択した。そして、リスタートをした。

2.3.3 日付の確認と設定

ターミナルから「date」を入力し、日時を確認した。その後、「sudo date 101014052024」より、正しい日時を設定した。

2.3.4 Raspberry Pi のリモートログイン (SSH) 設定

「sudo raspi-config」より、設定画面を開き、Interfacing Options, SSH を選び、SSH server を有効にした。その後、hostname を選択し、「jikken-102210017」とした。sudo reboot より、再起動した。自分のコンピュータから「shun@jikken-102210017.local」より、リモート接続を行った。

2.3.5 単一 LED (Light Emitting Diode) の点滅

GPIO ブレッドボード接続ケーブルをブレッドボードと Raspberry Pi の GPIO 端子に接続した。図 2 の回路図ように、GPIO4 と GND に LED を接続した。本実験で用いる LED は長足側に抵抗が内蔵されているので、長足側を GPIO へ、短足側を GND へ接続した。

Raspberry Pi による LED 点滅制御コード 2-1.py を作成した。これを、図 3 に示す。

その後, 2-1.py を Raspberry Pi 上で実行した.

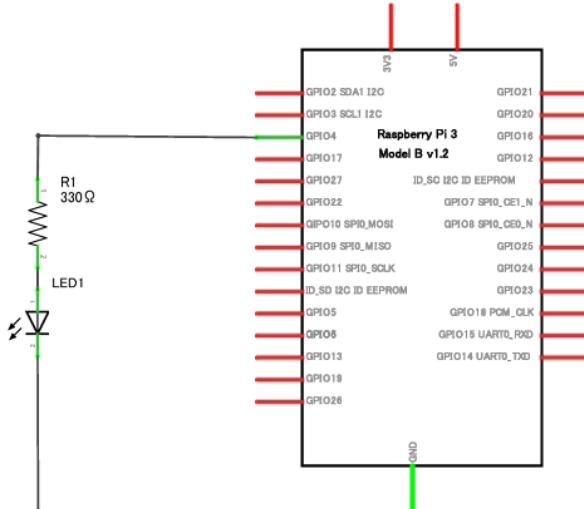


図 2 Raspberry Pi と LED1 個を接続する回路図

```
from gpiozero import OutputDevice
import time
# LED 接続 GPIO 端子番号
LED_PIN = 4
# LED 用の出力端子を生成する
led = OutputDevice(LED_PIN)
while True:
    # LED 接続端子を High にする
    led.on()
    time.sleep(1)
    # LED 接続端子を Low にする
    led.off()
    time.sleep(1)
```

図 3 2-1.py

2.4 実験結果

2-1.py を実行した結果、接続した LED が点滅した。この時の状況を、図 4 に示す。

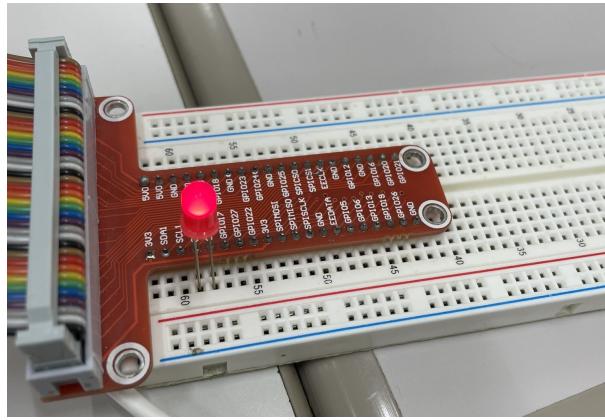


図4 LED 点滅の様子

2.5 考察

LED の点滅が 1 秒ごとだったのは, time.sleep (1) で LED が on と off にされているからだと考えた.

3 課題 2-2 単一スイッチの状態読み取り

3.1 目的・概要

Raspberry Pi を用いて押しボタンスイッチ 1 個の状態読み取りを行う. 押しボタンスイッチを使用する.

3.2 必要な部品

3.2.1 課題 2-1 と同じ部品

Raspberry Pi 3 model B+, Micro SD カード, HDMI ケーブル, Micro USB 電源ケーブル, モニタ, USB キーボード, USB マウスプレッドボード, GPIO プレッドボード接続ケーブルは, 課題 2-1 と同じものを使用する.

3.2.2 押しボタンスイッチ

2 種類の端子を持ち, ボタンが押された間は 2 端子が導通状態, ボタンが押されていない間は解放状態となる.

3.2.3 抵抗 ($1k \Omega$)

金属皮膜抵抗 $1k \Omega$

3.2.4 配線ケーブル

オス-オスを使用した.

3.3 実験方法

図 5 の回路図のように, GPIO17 と GND の間に抵抗と押しボタンスイッチを配置した. このときの状況を, 図 6 に示す.

Raspberry Pi による押しボタンスイッチ読み取りコード 2-2.py を作成した. これを, 図 7 に示す. その後, 2-2.py を Raspberry Pi 上で実行し, スイッチを押したり離したりした.

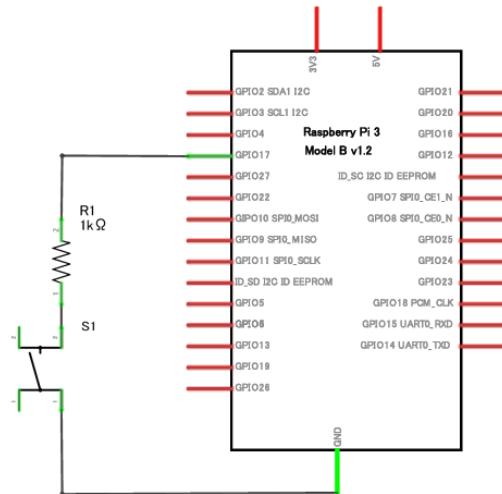


図 5 Raspberry Pi とスイッチ 1 個を接続する回路図

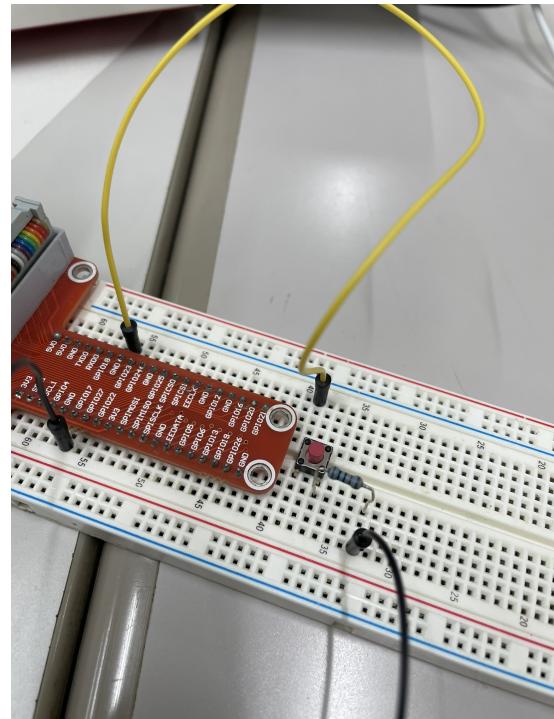


図 6 課題 2-2 配線状況

```

from gpiozero import InputDevice
import time

# スイッチ接続 GPIO 端子番号
SW_PIN = 17

# スイッチ用にプルアップモードで入力端子を生成する
switch = InputDevice(SW_PIN, pull_up=True)

while True:
    # スイッチ接続端子の状態読み取り
    if switch.value == 1: # 押されていれば 1 (True)

```

```
    print("Switch on")
else:
    print("Switch off")
time.sleep(0.5)
```

図 7 2-2.py

3.4 実験結果

2-2.py を実行した結果、switch on と switch off の出力がされた。この時の状況を、図 8 に示す。

```
shun@jikken-102210017:~ $ python3 2-2.py
Switch off
Switch on
Switch off
KeyboardInterrupt
```

図 8 スイッチを押した結果

3.5 考察

スイッチが反応するのが 0.5 秒ごとだったのは、コードの最終行の sleep.time (0.5) のためであると考えられる。これによって、連続でスイッチが押されたように反応することを防いでいると考えた。

4 課題 2-3 複数 LED とスイッチの制御

4.1 目的・概要

Raspberry Pi を用いて複数の LED とスイッチの制御を行う。

4.2 必要な部品

4.2.1 課題 2-1, 2-2 と同じ部品

Raspberry Pi 3 model B+, Micro SD カード, HDMI ケーブル, Micro USB 電源ケーブル, モニタ, USB キーボード, USB マウスブレッドボード, GPIO ブレッドボード接続ケーブル, 抵抗内蔵 LED, 押しボタンスイッチ, 抵抗, 配線ケーブルは課題 2-1, 2-2 と同じものを使用する。

4.3 実験方法

表 1 のように、LED と抵抗、押しボタンスイッチを配置した。このときの状況を、図 9 に示す。

表 1 回路の接続方法

接続する部品	GPIO 端子の接続 1	GPIO 端子の接続 2
LED1	GPIO4	GND
LED2	GPIO17	GND
LED3	GPIO27	GND
LED4	GPIO22	GND
抵抗と SWITCH1	GPIO5	GND
抵抗と SWITCH2	GPIO6	GND

Raspberry Pi による複数 LED とスイッチの制御コード 2-3.py を作成した。これを、図 10 に示す。その後、2-3.py を Raspberry Pi 上で実行し、二つのスイッチを押したり離したりした。

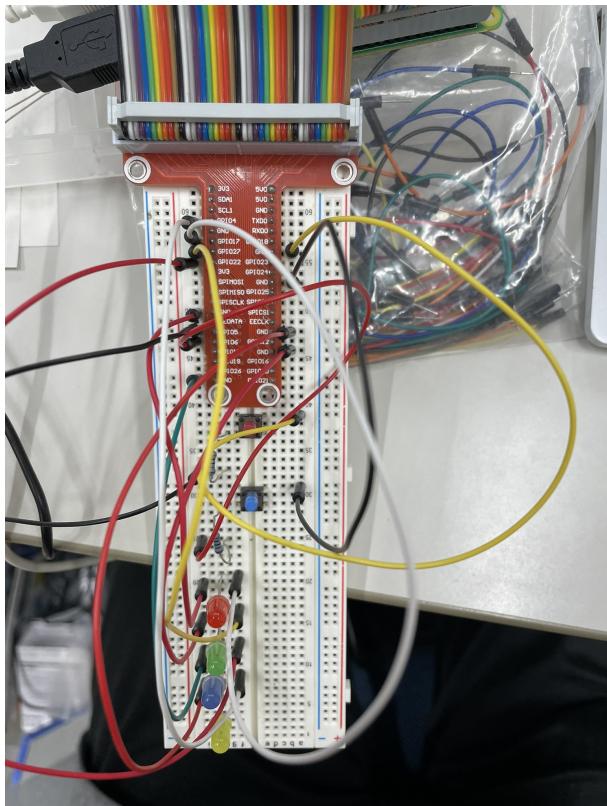


図 9 課題 2-3 の配線状況

```
//省略

isStopped=False
isReversed=False
i = 0

while True:
    # スイッチ 1 接続端子の状態読み取り
    if switch1.value == 1:
        if(isStopped == False):
            isStopped = True #止める判定
        else:
            isStopped = False #止める判定
        print("Switch1 on")
    # スイッチ 2 接続端子の状態読み取り
    if switch2.value == 1:
        if(isReversed == False):
```

```
    isReversed = True #逆判定
else:
    isReversed = False #逆判定
print("Switch2 on")
print(i)
if isStopped == True:
    time.sleep(1)
else:
    if isReversed == True:
        i-=1
    else:
        i+=1
j = i%4
leds[j].on()
time.sleep(0.5)
leds[j].off()
time.sleep(0.5)
```

図 10 2-3.py

4.4 実験結果

2-3.py を実行すると, LED が順番についた. スイッチ 1 を押すと, LED の点滅が止まった. 再びスイッチ 1 を押すと, LED の点滅が再開した. スイッチ 2 を押すと, LED の点滅の順番が逆になった. 再びスイッチ 2 を押すと, LED の点滅の順番がもとに戻った. この時の状況を, 図 11 に示す.

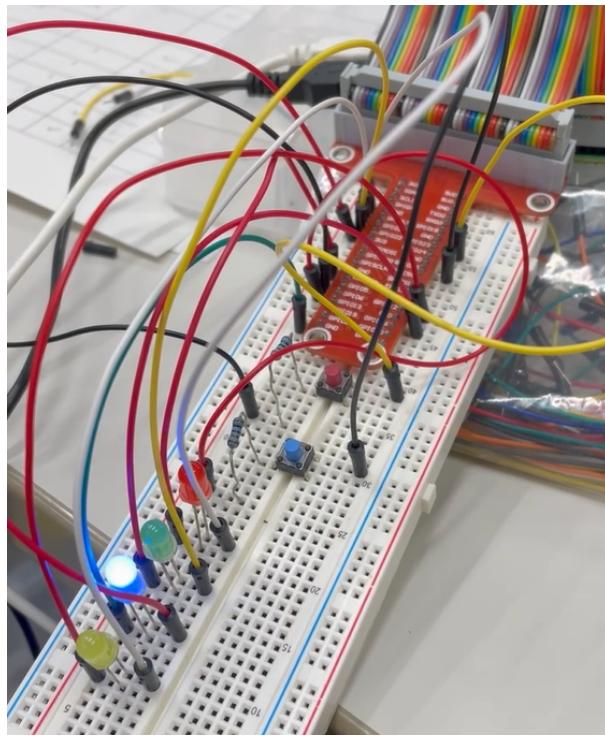


図 11 スイッチを押した結果

4.5 考察

複数の LED を順番に点滅させることができたのは, while 文内で leds という配列に対し 0 3 のインデックスを与えたからだと考えられる. また, 課題 2 と同じように, スイッチが反応するのが 0.5 秒間隔だったのは, sleep.time (0.5) のためであると考えられる.

5 課題 2-4 単一ステッピングモータの制御

5.1 目的・概要

Raspberry Pi を用いてステッピングモータの制御を行う. Raspberry Pi とステッピングモータ 1 個を接続し, ステッピングモータをユニポーラ 1 相励磁, 2 相励磁, 1-2 相励磁により正転, 逆転, 停止を行うコードを作成して動作する.

5.2 必要な部品

5.2.1 課題 2-1, 2-2 と同じ部品

Raspberry Pi 3 model B+, Micro SD カード, HDMI ケーブル, Micro USB 電源ケーブル, モニタ, USB キーボード, USB マウスプレッドボード, GPIO プレッドボード接続ケーブルは課題 2-1, 2-2 と同じものを使用する.

5.2.2 ステッピングモータ

モータの各相（コイル）に、ある順序で励磁パルスを与えると一定角度回転する。

5.2.3 ステッピングモータ制御回路

モータをあらかじめ設定した励磁方式に従って電流を流すように制御する。

5.2.4 2.1mm DC ジャック

USB DC5V to DC12V 昇圧ケーブルを接続し、モバイルバッテリーを使う。

5.2.5 USB DC5V to DC12V 昇圧ケーブル

モバイルバッテリーと DC ジャックを接続する。

5.2.6 モバイルバッテリー

ELECOM 社の EC-M01BK を使用した。

5.2.7 配線ケーブル

オス-オスとメス-オスの両方を使用した。

5.3 実験方法

表 2 のように、ステッピングモータとステッピングモータ制御回路を配置した。モバイルバッテリーを USB DC5V to DC12V 昇圧ケーブルを用いて、DC ジャックに接続した。このときの状況を、図 12 に示す。

表 2 回路の接続方法

接続元	接続先
モータ制御回路 IN 1	GPIO6
モータ制御回路 IN 2	GPIO13
モータ制御回路 IN 3	GPIO19
モータ制御回路 IN 4	GPIO26
モータ制御回路 +	DC ジャック +
モータ制御回路 -	DC ジャック -
DC ジャック -	GND

Raspberry Pi による 1 相励磁による正転、逆転、停止制御コード 2-4-1.py, 2 相励磁による正転、逆転、停止制御コード 2-4-2.py, 1-2 相励磁による正転、逆転、停止制御コード 2-4-3.py, を作成した。これらを、図 13、図 14、図 15 に示す。その後、2-4-1.py, 2-4-2.py, 2-4-3.py, を Raspberry Pi 上で実行した。

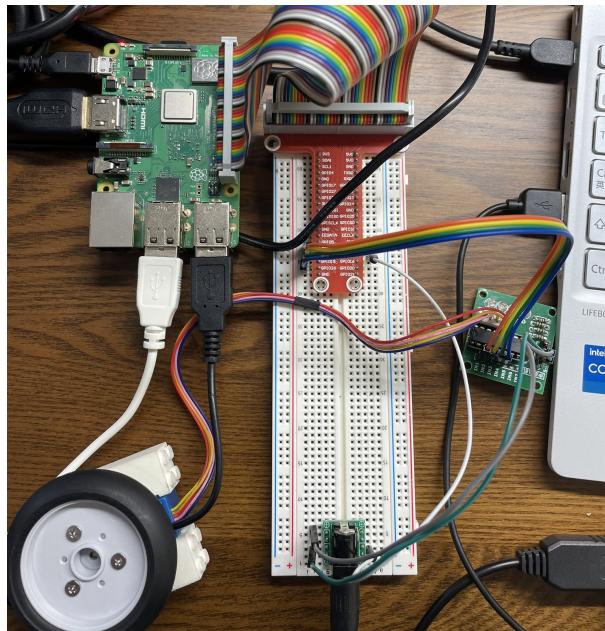


図 12 課題 2-4 の配線状況

```
//省略
```

```
# 1相励磁によるステッピングモータ運転
while True:
    i = 0
    for j in range(0, 1000):
        out_motor_pin(i)
        i += 1
        if i >= 4:
            i = 0 # 一周したら戻る
        # 亂調を避けるために少し待つ
        time.sleep(TIME_SLEEP)

    i = 3
    for j in range(0, 1000):
        out_motor_pin(i)
        i -= 1
        if i <= -1:
            i = 3 # 一周したら戻る
        # 亂調を避けるために少し待つ
        time.sleep(TIME_SLEEP)
```

```

for j in range(0, 1000):
    out_motor_pin(i)

    # 亂調を避けるために少し待つ
    time.sleep(TIME_SLEEP)

```

図 13 2-4-1.py

```

//省略

# 逆回転で指定した 2 相を励磁する関数 2
def two_out_motor_pin2(pin_num):
    #1 つ後ろのコイル
    pin_next = pin_num + 1
    if pin_next >= 4:
        pin_next -= 4

    for i in range(0, 4):
        if i == pin_num or i == pin_next:
            motor_pins[i].on()
        else:
            motor_pins[i].off()

    # 2 相励磁によるステッピングモータ運転
    while True:
        i = 0
        for j in range(0, 1000):
            two_out_motor_pin1(i)
            i += 1
            if i >= 4:
                i = 0 # 一周したら戻る
            # 乱調を避けるために少し待つ
            time.sleep(TIME_SLEEP)

        i = 3
        for j in range(0, 1000):

```

```

two_out_motor_pin2(i)
i -= 1
if i <= -1:
    i = 3 # 一周したら戻る
# 亂調を避けるために少し待つ
time.sleep(TIME_SLEEP)

for j in range(0, 1000):
    two_out_motor_pin1(i)

# 亂調を避けるために少し待つ
time.sleep(TIME_SLEEP)

```

図 14 2-4-2.py

```

//省略

# 1, 2 相励磁によるステッピングモータ運転
while True:
    i = 0
    for j in range(0, 1000):
        if j%2==0:
            out_motor_pin(i)
        else:
            two_out_motor_pin1(i)
            i += 1
        if i >= 4:
            i = 0 # 一周したら戻る
# 亂調を避けるために少し待つ
time.sleep(TIME_SLEEP)

i = 3
for j in range(0, 1000):
    if j%2==0:
        out_motor_pin(i)
    else:
        two_out_motor_pin2(i)

```

```

    i -= 1
if i <= -1:
    i = 3 # 一周したら戻る
# 亂調を避けるために少し待つ
time.sleep(TIME_SLEEP)

for j in range(0, 1000):
    two_out_motor_pin1(i)

# 亂調を避けるために少し待つ
time.sleep(TIME_SLEEP)

```

図 15 2-4-3.py

5.4 実験結果

2-4-1.py を実行すると、ステッピングモーターがある程度正転し、逆転し、停止し、再び正転した。

2-4-2.py を実行したときも同様に、正転、逆転、停止を繰り返した。また、2-4-1.py を実行したときよりも、振動が小さかった。

2-4-3.py を実行したときも同様に、正転、逆転、停止したが、回転速度は 2-4-1.py, 2-4-2.py を実行したときよりも遅かった。

また、TIME SLEEP を 0.2 に設定し実行したところ、2-4-1.py ではランプが 1 つずつつき、2-4-2.py ではランプが 2 つずつつき、2-4-3.py ではランプが 1 つつくと 2 つつくを繰り返した。

5.5 考察

2 相が 1 相に対して振動が小さかったのは、二つの相を同時に吸収するからだと考えられる。

1-2 相励磁方式の回転速度が遅かったことに疑問を抱いたので、これについて調べた。1-2 相励磁方式は、1 相励磁方式と 2 相励磁方式とを交互に行うことで回転磁界を作り、回転子を回転する方式である。SW1 から SW4 の各コイルは、表 3 に示すように変化する。1 相励磁時から 2 相励磁に切り替わると $1/4$ ピッチずれた状態になり、再び 1 相励磁になるとそれが $1/2$ ピッチになる。このように、1-2 相励磁方式は 1 相励磁方式や 2 相励磁方式に対してステップ角が半分になるので、回転速度が遅くなる。(ana-dig, 2024)

表 3 1-2 相励磁方式の各相の入力状態

SW / step	1	2	3	4	5	6	7	8
SW1	1	1	1	0	0	0	0	0
SW2	0	0	1	1	1	0	0	0
SW3	0	0	0	0	1	1	1	0
SW4	1	0	0	0	0	0	1	1

本来今回の実験では、USB DC5V to DC12V 昇圧ケーブルとモバイルバッテリーは使わないはずであったが、指導書を読み間違えてしまったことにより、課題 2-5 のようにこれらを使用して回路を作成してしまった。幸い大きな問題はなかったが、今後は間違えないように気を付けたい。

6 課題 2-5 複数ステッピングモータの制御

6.1 目的・概要

Raspberry Pi を用いて、複数の押しボタンスイッチからの入力に基づき、複数のステッピングモータを制御する。

6.2 必要な部品

6.2.1 課題 2-1, 2-2, 2-4 と同じ部品

Raspberry Pi 3 model B+, Micro SD カード, HDMI ケーブル, Micro USB 電源ケーブル, モニタ, USB キーボード, USB マウスブレッドボード, GPIO ブレッドボード接続ケーブル, 抵抗内蔵 LED, 押しボタンスイッチ, 抵抗, 配線ケーブル, ステッピングモータ, ステッピングモータ制御回路, 2.1mm DC ジャック, USB DC5V to DC12V 昇圧ケーブル, モバイルバッテリーは課題 2-1, 2-2, 2-4 と同じものを使用する。

6.3 実験方法

表 4 のように、ステッピングモータ、テッピングモータ制御回路、抵抗、押しボタンスイッチを配置した。このときの状況を、図 16 に示す。

表 4 回路の接続方法

接続する部品	接続 1	接続 2
抵抗と SWITCH1	GPIO5	GND
抵抗と SWITCH2	GPIO23	GND
モータ制御回路 1 IN 1	GPIO6	
モータ制御回路 1 IN 2	GPIO13	
モータ制御回路 1 IN 3	GPIO19	
モータ制御回路 1 IN 4	GPIO26	
モータ制御回路 1 +	DC ジャック +	
モータ制御回路 1 -	DC ジャック -	
モータ制御回路 2 IN 1	GPIO4	
モータ制御回路 2 IN 2	GPIO17	
モータ制御回路 2 IN 3	GPIO27	
モータ制御回路 2 IN 4	GPIO22	
モータ制御回路 2 +	DC ジャック +	
モータ制御回路 2 -	DC ジャック -	
DC ジャック -	GND	

複数の押しボタンスイッチからの入力に基づき、複数のステッピングモータを制御するコード 2-5.py を作成した。

これを、図 17 に示す。その後、2-5.py を Raspberry Pi 上で実行し、二つのスイッチを押したり離したりした。

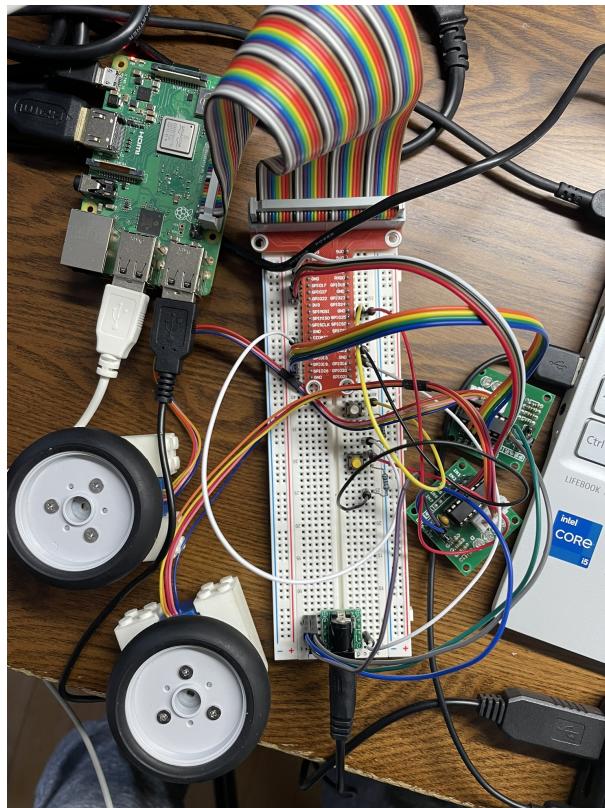


図 16 課題 2-5 の配線状況

```
//省略

isStopped=False
isReversed=False

# 1相励磁によるステッピングモータ運転
i = 0 #モーター1
j = 0 #モーター2
while True:
    # スイッチ1接続端子の状態読み取り
    if switch1.value == 1:
        if(isStopped == False):
            isStopped = True #止める判定
        else:
            isStopped = False #止める判定
    print("Switch1 on")
    time.sleep(1)
```

```

# スイッチ 2 接続端子の状態読み取り
if switch2.value == 1:
    if(isReversed == False):
        isReversed = True #逆判定
    else:
        isReversed = False #逆判定
    print("Switch2 on")
    time.sleep(1)

if isStopped == True:
    time.sleep(1)
else:
    i += 1
    if i >= 4:
        i = 0

if isReversed == True:
    j-=1
    if j <= -1:
        j = 3
else:
    j+=1
    if j >= 4:
        j = 0
out_motor1_pin(i)
out_motor2_pin(j)
# 亂調を避けるために少し待つ
time.sleep(TIME_SLEEP)

```

図 17 2-5.py

6.4 実験結果

2-5.py を実行すると、2つのステッピングモータが同じ方向に回転した。スイッチ 1 を押すと、両方のモータが停止した。再びスイッチ 1 を押すと、両方のモータが回転した。スイッチ 2 を押すと、片方のモータが逆回転した。再びスイッチ 2 を押すと、モータの回転方向が戻った。

6.5 考察

停止と逆回転の実装は、課題 2-3 と同じ原理である。isStopped, isReversed の boolean の値を使い、ステッピングモータの回転を制御している。この方法が直感的に理解しやすいと考えた。

7まとめ

7.1 実験を通して分かったこと

Raspberry Pi を用いた LED, スイッチ, ステッピングモータの制御の方法を理解した。

7.2 工夫したこと

実験で行ったことについて逐一メモや写真に記録を残し、後から確認しやすいようにした。

7.3 反省点

課題 2-4 で仕様書を読み間違えてしまった。仕様書では、Raspberry Pi の 5V 出力を使うはずだったが、課題 2-5 と同様にモバイルバッテリーを使ってしまった。今回の実験では大きな問題にならなかったが、今後はこのようなミスをなくしたい。

参考文献

[1] PassMark: ステッピングモータの動作原理.

<https://ana-dig.com/stepper-motor1/> 2024.