実験実施日 2024 年 10 月 17 日, 24 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 3 マニュアル操作ステッピングモータカーの作成

## 2.1　目的・概要

Raspberry Pi を用いて 5 つのボタンにより, 前進, 後退, 左回転, 右回転, 停止が制御可能な敷設コースを走破可能なステッピングモータカーを作成する.

## 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　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.

#### 2.2.5 抵抗（1k Ω）

金属皮膜抵抗 1k Ω

#### 2.2.6 配線ケーブル

オス-オス, メス-オスを使用した.

#### 2.2.7 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.

#### 2.2.8 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.

#### 2.2.9 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.

#### 2.2.10 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.

#### 2.2.11 モバイルバッテリー

ELECOM 社の EC-M01BK を使用した.

#### 2.2.12 構築部材

ステッピングモータマウンタや従輪の構築部材にレゴを用いた.

### 2.3 実験方法

表 1 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, DC ジャックを配置した.

表 1　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 |
| 抵抗と SWITCH3 | GPIO16 | GND |
| 抵抗と SWITCH4 | GPIO20 | GND |
| 抵抗と SWITCH5 | GPIO21 | 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 | |

レゴを用いて, マウンタ, Raspberry Pi, ブレッドボード, モバイルバッテリーを搭載したステッピングモータカーを作成した. これを図 1 に示す.

このマニュアル操作モータカーを動かすためのコード 3.py を作成した. これを図 3 に示す. スイッチ 1 が前進, スイッチ 2 が左旋回, スイッチ 3 が右旋回, スイッチ 4 が後退, スイッチ 5 が停止に対応している.

コースの入り口にステッピングモータカーを置き, 作成した 3.py を実行した. そして, コースの状況に合わせてスイッチを押し, 前進, 右折, 左折, 後退, 停止をさせ, 出口までモーターカーを走らせた. 使用するコースを図 2 に示す.

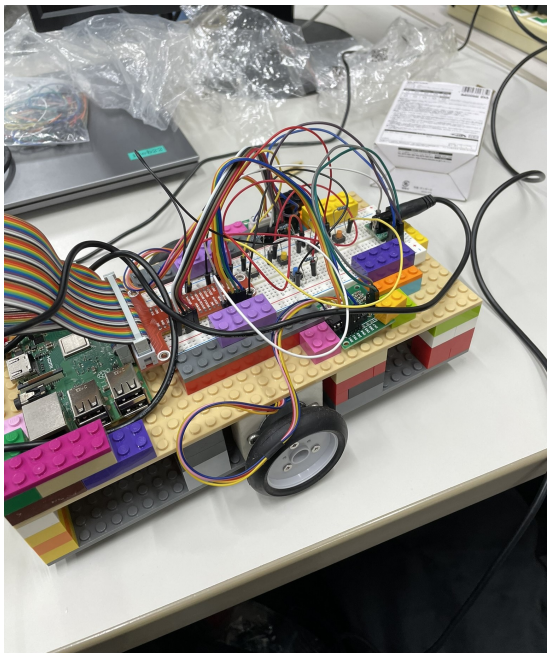

図 1　作成したステッピングモータカー

図 2　使用するコース

```
//省略

isForward=False
isBack=False
isStopped=False
isTurnRight=False
isTurnLeft=False

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:
        isForward = True
        isBack=False
        isTurnRight=False
        isTurnLeft=False
        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:
    isForward = False
    isBack=True
    isTurnRight=False
    isTurnLeft=False
    print("Switch2 on")
    time.sleep(1)

# スイッチ 3 接続端子の状態読み取り
if switch3.value == 1:
    isForward = False
    isBack=False
    isTurnLeft=False
    isTurnRight=True
    print("Switch3 on")
    time.sleep(1)

# スイッチ 4 接続端子の状態読み取り
if switch4.value == 1:
    isForward = False
    isBack=False
    isTurnLeft=True
    isTurnRight=False
    print("Switch4 on")
    time.sleep(1)

# スイッチ 5 接続端子の状態読み取り
if switch5.value == 1:
    if(isStopped == False):
        isStopped = True
    else:
        isStopped = False
    print("Switch5 on")
    time.sleep(1)

if isStopped == True:
    time.sleep(1)
else:
    if(isForward == True):
```

```
        i += 1
        if i >= 4:
            i = 0
        j -=1
        if j <= -1:
            j = 3

    if(isBack == True):
        i -= 1
        if i <= -1:
            i = 3
        j +=1
        if j >= 4:
            j = 0

    if(isTurnLeft):
        i += 1
        if i >= 4:
            i = 0
        j +=1
        if j >= 4:
            j = 0

    if(isTurnRight):
        i -= 1
        if i <= -1:
            i = 3
        j -=1
        if j <= -1:
            j = 3

    out_motor1_pin(i)
    out_motor2_pin(j)
    # 乱調を避けるために少し待つ
    time.sleep(TIME_SLEEP)
```

図 3　3.py の一部

## 2.4 実験結果

3.py を実行しスイッチ 1 を押した結果, ステッピングモータカーが前進した. 一つ目の右折でスイッチ 5 を押した結果, モータカーが停止した. そして, スイッチ 3 を押すと, モータカーが右旋回をした. また, 次のT字路にて, スイッチ 5 で停止させた後スイッチ 2 を押すと, 左旋回をした. このように, コースに合わせてスイッチを押した結果, 走破することができた. 提出した課題 3.MP4 にこのときの状況が記録されている.

## 2.5 考察

各スイッチに対し一つずつ boolean の値を用いたことで, 5 つの状態を正しく管理することができた. これは, 課題 2-5 で作成したコードと原理は同じと考えられる.

初めにモータのタイヤをステッピングモータカーの後部に設置し, 前部には別のタイヤを設置したところ, 絨毯の上でうまく旋回しなかった. そこで, モータのタイヤを真ん中に設置し別のタイヤを使わないようにした結果, うまく旋回するようになった. これはステッピングモータのタイヤのみに力がかかるようになり, 余計な摩擦が減ったからだと考えられる.

# 3 課題 4 自動走行ステッピングモータカーの作成

## 3.1 目的・概要

Raspberry Pi と距離センサ計を用いて, 1 つ以上の T 字路, 右折, 左折路を含むコースを自動で走破可能なステッピングモータカーを作成する.

## 3.2 必要な部品

### 3.2.1 課題 3 と同じ部品

Raspberry Pi とその他部品, ブレッドボード, GPIO ブレッドボード接続ケーブル, 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, 構築部材は課題 3 と同じ部品を使用した.

### 3.2.2 赤外線式距離センサ計: Sharp GP2YOA21

80cm 程度までの距離を計測でき, 距離に応じた電圧（約 0.5-3.2V）を出力する.

### 3.2.3 A/D コンバータ: MCP3424 with address switch

アナログ信号の振幅を離散的な周期で切り出し, 符号で表されたデジタル信号に変換する. 標本化, 量子化, 符号化の順にデジタル信号に変換していく.

標本化は, 連続なアナログ信号の振幅値を離散的な周期で切り出すことである. 量子化は, 離散的な周期で切り出された振幅値を、離散的な振幅値に近似することである. 符号化は, 離散的な振幅値を”0” と”1” の 2 値で表す符号に変換することである.（rohm, 2024）

#### 3.2.4 抵抗（1M Ω）

金属皮膜抵抗 1M Ω

## 3.3 実験方法

表 2, 図 4 に示すように, ステッピングモータ, テッピングモータ制御回路, 抵抗, DC ジャック, GP2Y0A21, MCP3424 を配置した.

表 2　回路の接続方法

| 接続する部品 | 接続 1 | 接続 2 |
|---|---|---|
| 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 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 1, 2 + | DC ジャック + | |
| モータ制御回路 1, 2 - | DC ジャック - | |
| DC ジャック - | GND | |
| 1M Ω抵抗 1 | GP2YOA21 1 vo | MCP3424 ch1+ |
| 1M Ω抵抗 2 | GP2YOA21 2 vo | MCP3424 ch2+ |
| 1M Ω抵抗 3 | GP2YOA21 3 vo | MCP3424 ch3+ |
| MCP3424 ch1-,ch2-,ch3- | GND | |
| MCP3424 +5v | 5v | |
| MCP3424 GND | GND | |
| MCP3424 SDA | SDA1 | |
| MCP3424 SCL | SCL1 | |
| GP2YOA21 1,2,3 GND | GND | |
| GP2YOA21 1,2,3 Vcc | 5v | |

課題 3 で作成したステッピングモータカーにレゴを用いて GP2YOA21 を設置した. これを図 5, 図 6 に示す.

この自動走行ステッピングモータカーを図 2 のコースに対し, 順走させるためのコード 4-1.py と逆走させるためのコード 4-2.py を作成した. これらを図 7, 8 に示す.

コースの入口にステッピングモータカーを置き, 作成した 4-1.py を実行した. その後, コースの出口にステッピングモータカーを置き, 作成した 4-2.py を実行した.

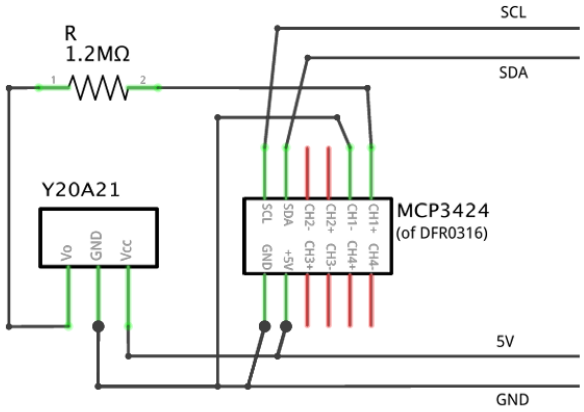


図 4　GP2Y0A21 と MCP3424 を用いた距離計測回路

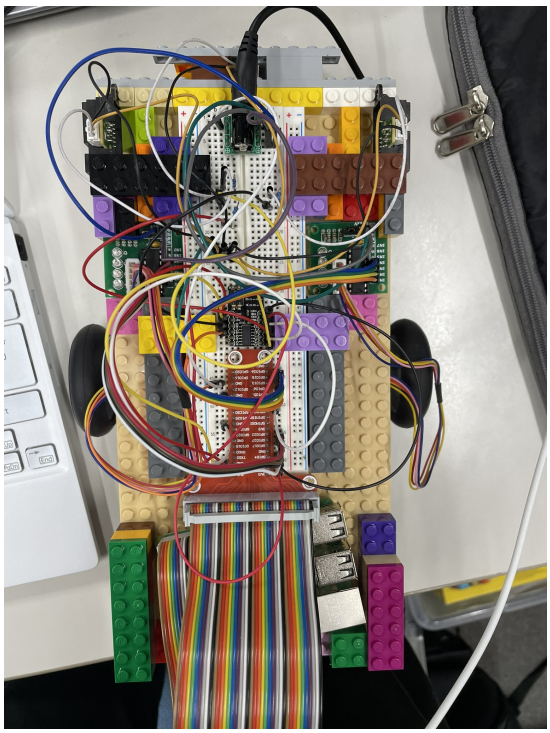

図 5　作成したステッピングモータカー

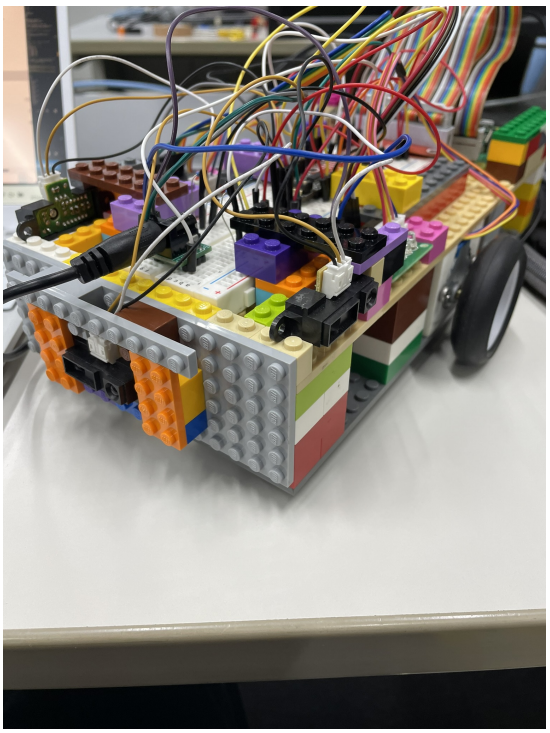

図 6　作成したステッピングモータカー

```
//省略

i = 0 #モーター 1
j = 0 #モーター 2
k = 0 #時間管理用
isForward = False
```

```
isTurnRight=False
isTurnLeft=False
while True:
    k += 1
    if(k%200==0):
        config1 = (mcp3424.cfg_read | mcp3424.cfg_ch1 | mcp3424.cfg_once |
                mcp3424.cfg_12bit | mcp3424.cfg_PGAx1)
        config2 = (mcp3424.cfg_read | mcp3424.cfg_ch2 | mcp3424.cfg_once |
                mcp3424.cfg_12bit | mcp3424.cfg_PGAx1)
        config3 = (mcp3424.cfg_read | mcp3424.cfg_ch3 | mcp3424.cfg_once |
                mcp3424.cfg_12bit | mcp3424.cfg_PGAx1)

        # ch1
        data1 = i2c.write_byte(addr, config1)
        time.sleep(1 / mcp3424.sps_12bit)
        while True:
            data1 = i2c.read_i2c_block_data(addr, 0, 3)  # Read the current value
            if data1[2] >> 7 == 0:
                break
            time.sleep(0.001)
        volt1 = mcp3424.to_volt(data1, 12)

        # ch2
        data2 = i2c.write_byte(addr, config2)
        time.sleep(1 / mcp3424.sps_12bit)
        while True:
            data2 = i2c.read_i2c_block_data(addr, 0, 3)  # Read the current value
            if data2[2] >> 7 == 0:
                break
            time.sleep(0.001)
        volt2 = mcp3424.to_volt(data2, 12)

        # ch3
        data3 = i2c.write_byte(addr, config3)
        time.sleep(1 / mcp3424.sps_12bit)
        while True:
            data3 = i2c.read_i2c_block_data(addr, 0, 3)  # Read the current value
            if data3[2] >> 7 == 0:
                break
```

```
        time.sleep(0.001)
    volt3 = mcp3424.to_volt(data3, 12)
    print("ch1:"+ str(volt1) + "ch2:"+ str(volt2) + "ch3:"+ str(volt3))

    if volt2 < 1.0:
        isForward = True
        isTurnRight=False
        isTurnLeft=False
    elif volt1 > volt3:
        isForward = False
        isTurnRight=True
        isTurnLeft=False
    else:
        isForward = False
        isTurnRight=False
        isTurnLeft=True

    if volt1 > 1.7:
        isForward = False
        isTurnRight=True
        isTurnLeft=False
    if volt3 > 1.7:
        isForward = False
        isTurnRight=False
        isTurnLeft=True

#print(isForward)
if(isForward == True):
    i += 1
    if i >= 4:
        i = 0
    j -=1
    if j <= -1:
        j = 3

if(isTurnRight):
    i -= 1
    if i <= -1:
        i = 3
```

```
        j -=1
        if j <= -1:
            j = 3

    if(isTurnLeft):
        i += 1
        if i >= 4:
            i = 0
        j +=1
        if j >= 4:
            j = 0

    out_motor1_pin(i)
    out_motor2_pin(j)
    # 乱調を避けるために少し待つ
    time.sleep(TIME_SLEEP)
```

図 7　4-1.py の一部

```
//省略

    if volt2 < 1.5:
        if volt3 < 1.5 and volt3 > 0.8:
            isForward = True
            isTurnRight=False
            isTurnLeft=False
        elif volt3 > 1.5:

            isForward = False
            isTurnRight=False
            isTurnLeft=True
        else:
            print("go straight\n")
            for l in range(500):
                i += 1
                if i >= 4:
                    i = 0
```

```
            j -=1
            if j <= -1:
                j = 3
            out_motor1_pin(i)
            out_motor2_pin(j)
            # 乱調を避けるために少し待つ
            time.sleep(TIME_SLEEP)
        isForward = False
        isTurnRight=True
        isTurnLeft=False

if volt2 > 1.0:
    if volt1 > volt3:
        print("turn right\n")
        for l in range(800):
            i -= 1
            if i <= -1:
                i = 3
            j -=1
            if j <= -1:
                j = 3
            out_motor1_pin(i)
            out_motor2_pin(j)
            # 乱調を避けるために少し待つ
            time.sleep(TIME_SLEEP)
    else:
        print("turn left\n")
        for l in range(800):
            i += 1
            if i >= 4:
                i = 0
            j +=1
            if j >= 4:
                j = 0
            out_motor1_pin(i)
            out_motor2_pin(j)
            # 乱調を避けるために少し待つ
            time.sleep(TIME_SLEEP)
```

```
        if volt1 > 1.7:
            isForward = False
            isTurnRight=True
            isTurnLeft=False
        if volt3 > 1.7:
            isForward = False
            isTurnRight=False
            isTurnLeft=True

    #print(isForward)
    if(isForward == True):
        i += 1
        if i >= 4:
            i = 0
        j -=1
        if j <= -1:
            j = 3

    if(isTurnRight):
        i -= 1
        if i <= -1:
            i = 3
        j -=1
        if j <= -1:
            j = 3

    if(isTurnLeft):
        i += 1
        if i >= 4:
            i = 0
        j +=1
        if j >= 4:
            j = 0

    out_motor1_pin(i)
    out_motor2_pin(j)
    # 乱調を避けるために少し待つ
    time.sleep(TIME_SLEEP)
```

図 8　4-2.py の一部

### 3.4　実験結果

4-1.py を実行した結果, 入り口から出口まで走破した. このときの状況は課題 4 順走.MP4 に記録されている. 一つ目の T 字路では左折し最短経路を通ったが, 二つ目の T 字路では左折をして行き止まりまで進んでから U ターンして出口に向かった.

4-2.py を実行した結果, 出口から入り口まで走破した. このときの状況は課題 4 逆走.MP4 に記録されている. モータカーの右側の壁に沿って, コースを走行した.

### 3.5　考察

初めにコードを作成したとき, while 文内で毎ループごとにセンサーの値を受け取る設計にした. しかし, これではモータカーはほとんど動かなかった. そこでセンサーの動作を少なくするため, 毎ループごとに整数 k の値を 1 増やしていき, k を 200 で割った余りが 0 になるときだけ, センサーの値を受け取るように変更した. この結果モータカーはうまく動いたことから, センサーの処理にかかる時間が大きすぎたことが原因であると考えられる.

今回使用した GP2Y0A21 と MCP3424 について, 1M Ωの抵抗一つを接続して使用したところ, 約 20cm の距離で 1.0V の出力があり, 約 10cm の距離で 1.5V の出力があった. モータカーの前面のセンサーが 1.0V 以上で旋回するようにするとうまく機能したのは, このためであると考えられる.

コースを順走させるには, モータカーの前方に壁が来るときに旋回をする単純なアルゴリズム（4-1.py）でよかった. しかし, 逆走させるには T 字路でモータカーの前方に壁がないときに旋回をしなければならない. そこで, モータカーの右側の壁と一定の距離を保ちながら走行するアルゴリズム（4-2.py）を考えた. 右側のセンサーの出力が 0.8V から 1.5V になるように適宜旋回をすると, 壁との距離を 10cm から 15cm に保ちながら走行することができた. しかし, ゴールへ壁伝いで行けないコースでは, このアルゴリズムは使えないと考えられる.

## 4　まとめ

### 4.1　実験を通して分かったこと

ステッピングモータカーの作成や GP2Y0A21 と MCP3424 を用いた距離計測の方法を理解した.

### 4.2　工夫したこと

外観の美しさや環境に配慮して, ガムテープを全く使わずにステッピングモータカーを作成した. センサーの取り付けが特に難しかったが, 工夫してレゴを配置した.

また, 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.

### 4.3 反省点

実験 4 で, 曲がり角を記憶したり地図を作成したりするような複雑なアルゴリズムではなく, その瞬間のセンサーの情報を判断するだけの単純なアルゴリズムしか作れなかった. もっと創意工夫を凝らしたアルゴリズムを考えるべきだった.

## 参考文献

[1] PassMark: *A/D* コンバータとは？.
https://www.rohm.co.jp/electronics-basics/ad-converters/ad_what2 2024.