実施日 2024 年 10 月 19 日, 27 日

コンピュータ科学実験b

ハードウェア 調査課題

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　調査課題 1（2-2）

2.1　チャタリングとは

チャタリングとは, リレー、スイッチがオンする際に機械的な振動によって短い周期のオン・オフを繰り返すことである. スイッチをオンにすると, 接点でスイッチが跳ね返り, 弾性振動により接点とくっついたり離れたりを繰り返す. これにより, スイッチをオンしてからある間, 負荷に Hi-Lo のパルス波形が印加されてしまい, スイッチを何度も押したような挙動をしてしまう.

2.2　ハードウェア的方法による対策

RC フィルタを用いてチャタリング波形の除去を行う方法がある. これにより, Hi-Lo のパルスをなだらかにすることができる. しかし, 出力波形が鈍ることと, 出力ラインに抵抗が入るので, 負荷のインピーダンスが低いと電圧降下が発生するというデメリットがある.

2.3　ソフトウェア的方法による対策

マイコンで信号を処理する場合は, ソフト処理でフィルタをすることができる. マイコンに入力された信号を一定間隔で読み取り, 規定回数連続で Hi 判定してはじめて Hi レベルが入力されたと判定する. これにより, チャタリングを防ぐことができる.

（analogista, 2021）

3　調査課題 2（2-2）

3.1　プルアップ及びプルダウンの使用

プルアップは, 入力ピンがデバイスに接続されていないときに, ピンの状態を安定させるために HIGH に保つ. そして, スイッチが押されたとき LOW になる. それに対し, プルダウンは, 入力ピンがデバイスに接続されていないときに LOW に保ち, スイッチが押されたとき HIGH になる.

つまり, スイッチが押されていないときにピンを HIGH に保ちたい場合にプルアップが使われ, LOW に保ちたい場合に, プルダウンが使われる.

3.2　プルアップ及びプルダウンの電子回路上での実装

プルアップは, Vcc（3.3V や 5V）, 抵抗, 入力ピン（GPIO など）, スイッチ, GND を図 1 のように配置し, スイッチが押されていないときは HIGH に保たれている. それに対し, プルダウンは, Vcc（3.3V や 5V）, 抵抗, 入力ピン（GPIO など）, スイッチ, GND を図 2 のように配置し, スイッチが押されていないときは LOW に保たれている.

（voltechno, 2019）

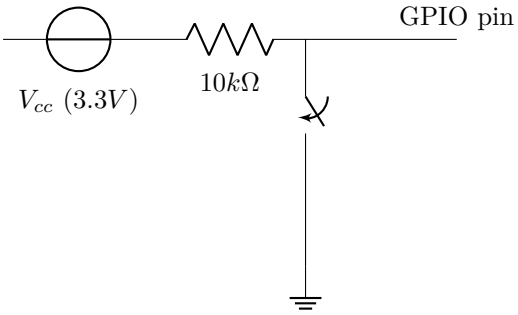

図 1　プルアップの実装

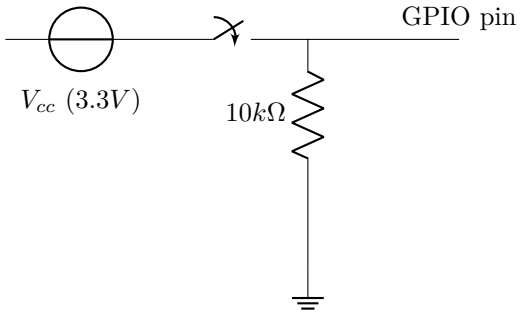

図 2　プルダウンの実装

4　調査課題 3（2-5）

4.1　ステッピングモータとは

ステッピングモータとは制御モータの一種で, 電流を流す相を切り替えることで一定の角度ずつ動いて回転する. パルスモータ，ステップ，ステッパモータなどとも呼ばれることがある.（techcompass, 2021）

4.2　ステッピングモータのユニポーラ駆動方式

ステッピングモータは, 固定子の励磁する相（SW1 から SW4 の各コイル）を, あるシーケンスに従って順番に切り換えて回転する. ここでは, 3 つのユニポーラ駆動方式について調べた.

4.2.1　1 相励磁方式

1 相励磁方式は, 常に固定子の 1 相コイル（巻線）だけを励磁（通電）し, 相を順番に切り換えて回転磁界を作り, 回転子を回転する方式である. SW1 から SW4 の各コイルは, 表 1 に示すように変化する.

表 1　1 相励磁方式の各相の入力状態

SW / step	1	2	3	4
SW1	1	0	0	0
SW2	0	1	0	0
SW3	0	0	1	0
SW4	0	0	0	1

4.2.2　2 相励磁方式

2 相励磁方式は, 常に固定子の 2 相コイル（巻線）を励磁（通電）し, 相を順番に切り換えて回転磁界を作り, 回転子を回転する方式である. SW1 から SW4 の各コイルは, 表 2 に示すように変化する. 常に二つの相が励磁されるので, 1 相励磁方式に比べ, 起動トルクが与えられ乱調が生じにくい. しかし, 1 相励磁に比べて 2 倍の電流容量が必要である.

表 2　2 相励磁方式の各相の入力状態

SW / step	1	2	3	4
SW1	1	1	0	0
SW2	0	1	1	0
SW3	0	0	1	1
SW4	1	0	0	1

4.2.3　1-2 相励磁方式

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, 回転子を回転させる方式である. SW1 から SW4 の各コイルは, 表 3 に示すように変化する. ステップ角はハーフ・ステップになるので, 回転速度は, 1 相励磁方式や 2 相励磁方式の半分となる.

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

5　調査課題 4（2-5）

5.1　台形速度制御とは

台形速度制御とは, モーターやロボットなどの機械を制御する際に, 目標速度に到達するまでの速度の変化を図 3 で示すように, 台形のような形に設定する制御方法である. この手法は, 機械を滑らかに加速, 減速させるために用いられる. 加速区間, 定速区間, 減速区間の 3 つのフェーズに分けらる. 加速区間は, 一定の加速度で速度を上昇させる区間であり, 設定された最大速度に到達すると終了する. 定速区間は, 速度が最大に達した後, この速度を一定に保ちながら機械を動作させる区間である. 減速区間は, 停止するために一定の加速度で速度を落とす区間である.

（Tajima, 2019）

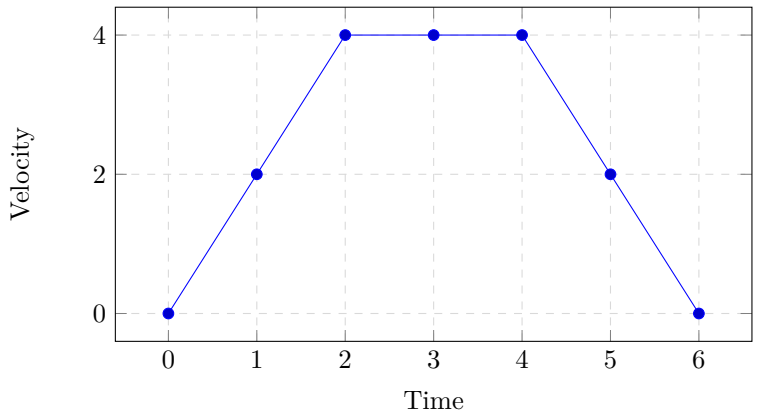

図 3　台形加速

5.2　ステッピングモータの台形速度制御

課題 2-4 で作成したステッピングモータの制御をするコードの time.sleep の値を変化させることで台形加速をするコード daikei.py を作成した. これを図 4 に示す. TIME SLEEP INITIAL は初期の time.sleep の停止時間であり, 減速区間でこの値まで停止時間を長くしていく. TIME SLEEP FASTEST は定速区間の time.sleep の停止時間であり, 加速区間でこの値まで停止時間を短くしていく. ACCELERATION STEPS は, 加速, 減速に用いるステップ数である. TOTAL STEPS は, 全体のステップ数であり, TOTAL STEPS - 2 * ACCELERATION STEPS の値が定速区間のステップ数である. current sleep はステップごとの停止時間を示している.

加速区間で, (TIME SLEEP INITIAL - TIME SLEEP FASTEST) / ACCELERATION STEPS の値だけ, 停止時間を短くする. その後停止時間が TIME SLEEP FASTEST と同じになったら, 定速区間へ移る. 減速区間でも同様に停止時間を長くし, 停止する.

このように sleep time で停止させる時間を制御することで台形速度制御を実装した. しかし, 停止時間を調整するだけでステッピングモータの速度を完全に制御できるかどうか分からない. 特にモータカーとして走らせる際に, 荷重によりトルク不足などの問題が起き完全な台形加速にならないかもしれない. そのため, 実際に

稼働させてみることが必要だと考えた.

```
from gpiozero import OutputDevice
import time

# ステッピングモータ接続 GPIO 端子番号
OUTPUT_PIN1_1 = 6
OUTPUT_PIN1_2 = 13
OUTPUT_PIN1_3 = 19
OUTPUT_PIN1_4 = 26

# 初期のモータ回転間隔
TIME_SLEEP_INITIAL = 0.01

# 定速運転時のモータ回転間隔
TIME_SLEEP_FASTEST = 0.002

# 加速・減速に使うステップ数
ACCELERATION_STEPS = 100

# 全体の運転ステップ数
TOTAL_STEPS = 1000

# モータ用の出力端子を生成する
motor_pins = [
    OutputDevice(OUTPUT_PIN1_1),
    OutputDevice(OUTPUT_PIN1_2),
    OutputDevice(OUTPUT_PIN1_3),
    OutputDevice(OUTPUT_PIN1_4),
]

# 指定した 1 相のみを励磁する関数
def out_motor_pin(pin_num):
    for i in range(0, 4):
        if i == pin_num:
            motor_pins[i].on()
        else:
            motor_pins[i].off()
```

```
#台形加速開始

#最初の time.sleep の時間
current_sleep = TIME_SLEEP_INITIAL

i = 0
# 加速区間
for step in range(ACCELERATION_STEPS):
    out_motor_pin(i)
    i = (i + 1) % 4
    time.sleep(current_sleep)

    # 加速：ステップごとに待機時間を短くしていく
    current_sleep -= (TIME_SLEEP_INITIAL - TIME_SLEEP_FASTEST) / ACCELERATION_STEPS
    if current_sleep < TIME_SLEEP_FASTEST:
        current_sleep = TIME_SLEEP_FASTEST

# 定速区間
for step in range(TOTAL_STEPS - 2 * ACCELERATION_STEPS):
    out_motor_pin(i)
    i = (i + 1) % 4
    time.sleep(current_sleep)

# 減速区間
for step in range(ACCELERATION_STEPS):
    out_motor_pin(i)
    i = (i + 1) % 4
    time.sleep(current_sleep)

    # 減速：ステップごとに待機時間を長くしていく
    current_sleep += (TIME_SLEEP_INITIAL - TIME_SLEEP_FASTEST) / ACCELERATION_STEPS
    if current_sleep > TIME_SLEEP_INITIAL:
        current_sleep = TIME_SLEEP_INITIAL
```

図 4　daikei.py

6　調査課題 5

6.1　I2C とは

I2C（Inter-Integrated Circuit）とは, Philips Semiconductors 社（現在の NXP Semiconductors 社）が開発した通信規格である. IC 間の通信を担い, マイコンとその周辺機器（EEPROM など）の通信でよく使われる.

6.2　SDA と SCL

I2C では, SDA と SCL の 2 本の信号線を用いて通信を行う. SDA は Serial Data の略で, データ用の信号線であり, SCL は Serial Clock の略で, クロック用の信号線である. それぞれの信号線にはプルアップ抵抗が接続される.

6.3　マスタとスレーブ

I2C では, デバイスはその役割によってマスタとスレーブに分けらる. マスタは通信の開始, クロック信号の生成, 通信の終了を行う. マスタはスレーブが持つスレーブアドレスを指定することで特定のスレーブと通信する. 図 5 のように, 1 つのマスタと複数のスレーブを SDA と SCL に接続する.
（ensatellite, 2021）

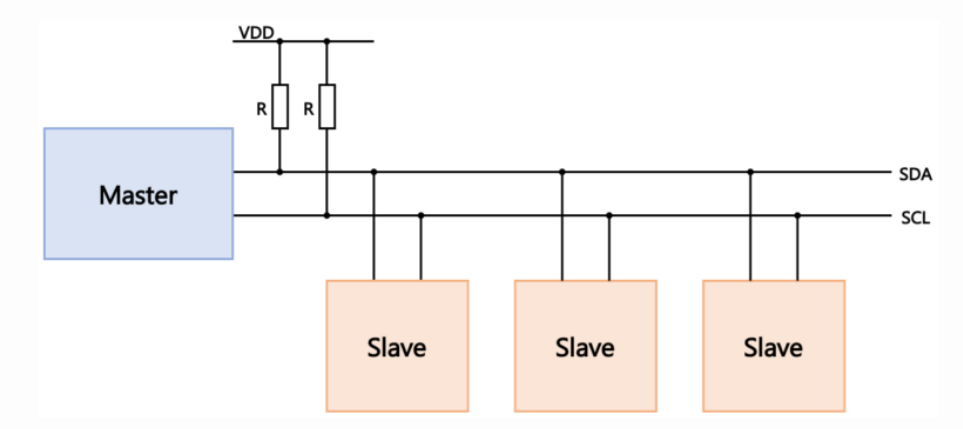

図 5　SDA と SCL の接続

7　調査課題 6

7.1　AD 変換とは

A/D 変換（Analog-to-Digital Conversion）とは, アナログ信号をデジタル信号に変換するプロセスのことである. 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. 標本化とは, 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. 量子化とは, 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. 符号化とは, 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. このような手順で, 連続値であるアナログ信号の各ポイントをサンプリングし, それを離散値であるデジタル値として表現していく.

A/D 変換は, 音声データのデジタル化や温度, 圧力, 光の強さなどのセンサー値のデジタル化, 医療機器や測定機器でのデータ処理などに使われている.

7.2 D/A 変換とは

D/A 変換（Digital-to-Analog Conversion）とは, デジタル信号をアナログ信号に変換するプロセスのことである. 電圧または電流に変換, 平滑化の順にアナログ信号に変換していく. 電圧または電流に変換では, 入力されたデジタル値を基に、コンバータが対応するアナログ電圧または電流を出力する. 平滑化では, デジタル値の変換は階段状になるため、スムーズな連続信号を得るために低パスフィルタを使用する.

D/A 変換は, 音楽再生, ビデオ再生などで使用されている.

（pabasic, 2016）

参考文献

[1] PassMark: **チャタリングとは？ 原因と対策方法について**.
https://analogista.jp/chattering/ 2021.

[2] PassMark: **プルアップ抵抗・プルダウン抵抗とは？ 電子回路に必須の考え方**.
https://voltechno.com/blog/pullup-pulldown/ 2019.

[3] PassMark: **ステッピングモータとは？**.
https://techcompass.sanyodenki.com/jp/training/servo/motor_tips/001/index.html 2021.

[4] PassMark: **台形速度の軌跡生成をしてロボットを制御する方法**.
https://tajimarobotics.com/acceleration-limited-feed-profile-2/ 2019.

[5] PassMark: *I2C* **の概要と仕組み**.
https://ensatellite.com/ja/i2c/ 2021.

[6] PassMark: *AD* **変換と** *DA* **変換の基本***:* **デジタル機器の心臓部**.
https://pabasic.com/100_engineer/105_digital/ad-da/ 2016.