
IOT 組込み機器制作実習

リリース 1.0

太田 佳希

2025 年 09 月 26 日

目次:

第1章 IOT 組込み制作実習 <RGB 識別ゲーム>

1.1 制作実習テーマ概要.....1

1.2 グループ4 メンバ.....1

第二章 担当項目名.....2

2.1 スピーカプログラム作成及び配線.....2

第三章 考察.....3

3.1 反省点.....3

3.2 改善点.....3

第四章 課題.....4

4.1 I2C.....4

4.2 GUI プログラミングについての概要説明.....5~6

4.3 git.....7

4.4 MQTT 概要.....8

第一章 IOT 組込み制作実習 <RGB 識別ゲーム>

1.1 制作実習テーマ概要

RGB を使用した識別ゲーム

1.2 グループ 4 メンバ

学籍番号/氏名/担当

2417104/今村匠/プログラム、配線

2417105/太田佳希/スピーカプログラムの作成

2417108/小谷怜香/配線、発表資料作成

2417115/ドディンフォン/プログラム及び資料作成補助

2417118/長谷川凱也/プログラム及び資料作成補助

第二章 担当項目名

2.1 スピーカプログラムの作成及び配線

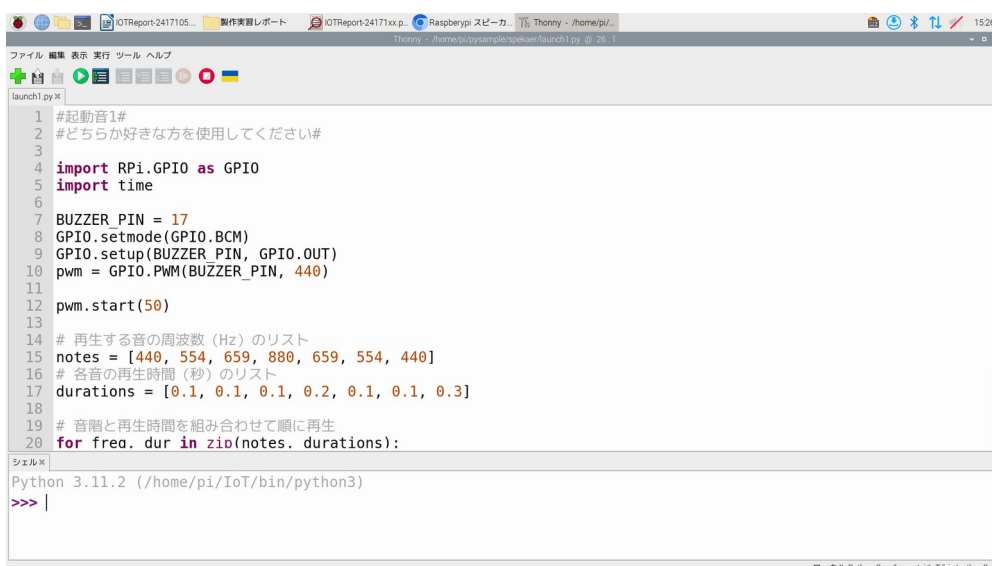
今回私は RGB 識別ゲームに使用する効果音の作成及び、スピーカの配線を行いました。

まずはじめにスピーカの配線についてインターネット等で調べました。使用部材はスピーカ、ブレッドボード、抵抗(330Ω)、ジャンパーピンです。15 番ピンに抵抗(330Ω)を挟んで、スピーカの+端子に接続しました。スピーカの-端子は GND に接続しました。

次にスピーカ発音のプログラムをインターネットを使用して作成しました。こちらも特に不具合なく動作しました。プログラミングは Thony を使用しています。

また、効果音は、ゲームスタート音、正解音、不正解音、ゲーム終了音の四種類作成しました。

以下にプログラムを掲載します。(一部抜粋)



```
1 #起動音1#
2 #どちらか好きな方を使用してください#
3
4 import RPi.GPIO as GPIO
5 import time
6
7 BUZZER_PIN = 17
8 GPIO.setmode(GPIO.BCM)
9 GPIO.setup(BUZZER_PIN, GPIO.OUT)
10 pwm = GPIO.PWM(BUZZER_PIN, 440)
11
12 pwm.start(50)
13
14 # 再生する音の周波数 (Hz) のリスト
15 notes = [440, 554, 659, 880, 659, 554, 440]
16 # 各音の再生時間 (秒) のリスト
17 durations = [0.1, 0.1, 0.1, 0.2, 0.1, 0.1, 0.3]
18
19 # 音階と再生時間を組み合わせて順に再生
20 for freq, dur in zip(notes, durations):
    pwm.ChangeFrequency(freq)
    time.sleep(dur)
    pwm.ChangeFrequency(440)
    time.sleep(0.1)
    pwm.stop()
    time.sleep(0.5)

シエル%
Python 3.11.2 (/home/pi/IoT/bin/python3)
>>> |
```

第三章 考察

3.1 反省点

今回の作業では効率を考え、スピーカプログラムとメインプログラムを分担して作成しました。なので、メインプログラムとスピーカプログラムを結合させる際、整合性が取れず、発表するまでに結合ができませんでした。

もう少しコミュニケーションを取り、連携しながらプログラムを作成するべきだと感じました。

3.2 改善点

上記に記載した通り、もう少しコミュニケーションを取りながらプログラムを作成するべきだと感じました。

また、スピーカプログラムを作成したラズベリーパイとメインプログラムを作成したラズベリーパイのバージョンに相違があったことも、スピーカプログラムをスムーズに実装できない原因だと考察しました。

以後、このようなことがないように、ラズベリーパイのバージョンを揃える、コーディングルールを決めてプログラムを作成する。等の改善が必要だと考えました。

第四章 課題

4.1.1 I2C の概要

I²C（Inter-Integrated Circuit）は、複数のデバイス（マイコン、センサー、EEPROM など）を2本の信号線（SDA：データ線、SCL：クロック線）で接続できるシリアル通信規格。

4.1.2 I2C と SPI の違い

I²C は2本の信号線（データとクロック）で複数のデバイスを接続できるシリアル通信方式で、スレーブごとにアドレスを持ち、主に低速で簡単な接続が求められる用途に使用される。一方、SPI は4本の信号線を使い、個別のデバイス選択ピンで接続先を切り替える方式で、より高速で安定した通信が可能だが、接続数が増えると配線が複雑になる。用途に応じて、I²C は省配線・複数接続向き、SPI は高速・高信頼性が求められる場面に向いている。

4.1.3 I2C のアドレスについて

I²C のアドレスは、マスターが通信相手のスレーブデバイスを識別するための番号で、通常7ビットまたは10ビットで構成されている。通信時にマスターがこのアドレスを送信し、一致したスレーブだけが応答することで、複数のデバイスを同じバス上で使い分けられる。

4.2.1 CUI のメリット、デメリット

CUI（キャラクタユーザインタフェース）は、キーボード操作によってコマンドを入力し、システムを操作する方式である。メリットは、操作が正確で軽量なため低スペックな環境でも動作し、複雑な作業を自動化しやすい点だ。デメリットは、コマンドの知識が必要で直感的でなく、初心者には扱いにくい点である。

4.2.2 GUI のメリット、デメリット

GUI（グラフィカルユーザインタフェース）は、視覚的な画面操作で直感的に使えるため、初心者にもわかりやすく操作しやすいのがメリットだ。一方で、処理が重くなりやすく、高度な操作や自動化には不向きなことがデメリットである。

4.2.3 TkEasyGUI の概要

`tkinter` をベースにした EasyGUI（正式には `easygui`）は、Python で GUI を使った対話的な操作を簡単に実装できるライブラリである。コードを書く量が少なく、ボタン、入力ボックス、メッセージ表示などが手軽に使えるため、GUI 初心者やちょっとしたツール作成に適している。複雑なレイアウトやカスタマイズは苦手だが、すぐに使えるのが特徴である。

4.3.1 git の概要

Git は、ファイルの変更履歴を管理する分散型バージョン管理システムで、ソースコードの追跡や共同開発を効率的に行うために使用される。ローカルでも履歴管理ができ、複数人での作業でも変更を統合・管理しやすいのが特徴である。

4.4.1 MQTT の概要

MQTT は、IoT 機器などの通信に使われる軽量なメッセージ通信プロトコルで、発行（Publish）と購読（Subscribe）の仕組みにより、低帯域や不安定なネットワークでも効率よくデータをやり取りできる。

4.4.2 MQTT のメリット、デメリット

MQTT のメリットは、軽量で省電力・低帯域でも安定して通信でき、多数のデバイスを効率よく管理できることである。デメリットは、セキュリティ機能が基本的にシンプルで、設定や運用で補う必要がある点と、大量データの転送には向かない点である。