

4. Pythonプログラミング

今回のポイント

- PythonによるGPIO制御
- Visual Studio Code
- Lチカ
- ポーリング・割り込み
- カメラモジュール
- 超音波センサー

PythonによるGPIO制御

gpiozeroは、Raspberry PiでGPIOピンを制御するための**Pythonライブラリ**です。

シンプルで直感的なコードで、LED・ボタン・センサーなどのハードウェアを制御することができます。

低レベルな設定やデバウンス処理などをライブラリの内部で自動処理するため、電子工作の**初心者でも手軽に**ハードウェア制御を始められます。

Visual Studio Code

Visual Studio Code (VS Code) とは

Visual Studio Codeは、Microsoftが提供する**IDE**（統合開発環境）です。

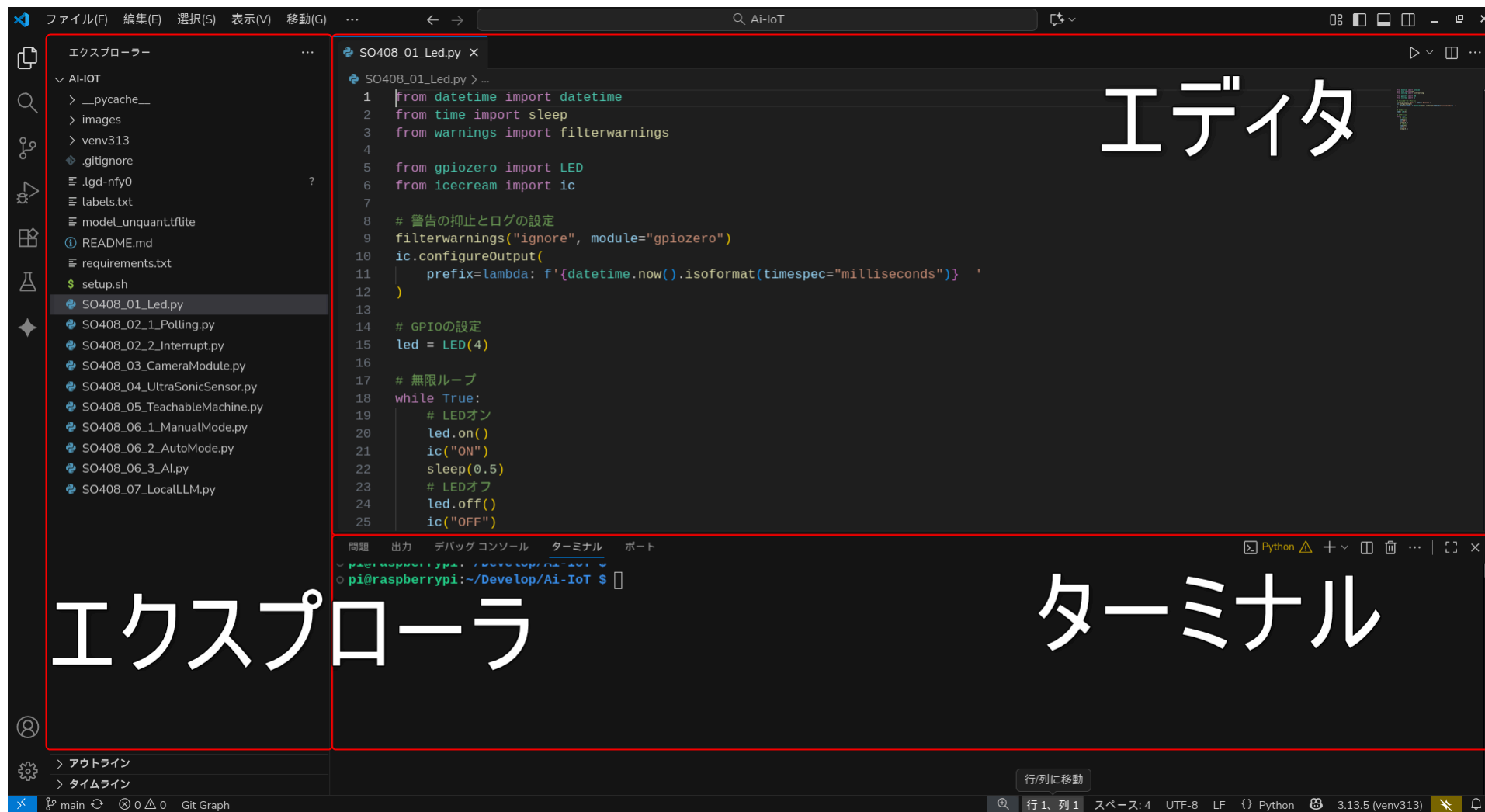
誰でも**無料**で利用できるオープンソースソフトウェアです。

様々なOSに対応しており、WindowsやmacOSだけでなく、Raspberry Piでも動作します。

軽量で高速な動作が特徴で、多くのプログラミング言語に対応しています。

コーディングだけでなく、プログラムを実行する**ターミナル**や**デバッガー**も内蔵されています。

また、豊富な**拡張機能**が提供されており、プログラミング開発に必要な機能を手軽に追加できます。



Lチ力

Lチカのプログラム例

```
from datetime import datetime
from gpiozero import LED
from time import sleep
from warnings import filterwarnings

# GPIOZeroの警告を非表示
filterwarnings('ignore', module='gpiozero')

# GPIOの設定
led = LED(4)

# 無限ループでLEDを点滅
while True:
    led.on()
    print(f'ON : {datetime.now().isoformat(timespec="milliseconds")}')
    sleep(0.5)
    led.off()
    print(f'OFF : {datetime.now().isoformat(timespec="milliseconds")}')
    sleep(0.5)
```

ポーリング・割り込み

ポーリング

ソフトウェアで外部機器の状態変化を**一定周期**でチェックする方法。

(例) 定期的にアプリを開いて、新着メッセージがないかチェックする。

割り込み

ハードウェアで外部の状態変化を検知し、**割り込み信号**を伝達する方法。

(例) 新着メッセージが届くと、スマホから通知が来てアプリを開く。

ポーリングのソースコード

```
from datetime import datetime
from gpiozero import LED, Button
from time import sleep
from warnings import filterwarnings

# GPIOZeroの警告を非表示
filterwarnings('ignore', module='gpiozero')

# GPIOの設定
led = LED(4)
button = Button(21)

# 無限ループでボタンの状態を監視
while True:
    if button.is_pressed:
        led.on()
        print(f'ON : {datetime.now().isoformat(timespec="milliseconds")}')
    else:
        led.off()
        print(f'OFF : {datetime.now().isoformat(timespec="milliseconds")}')
    sleep(0.5)
```

割り込みのソースコード

```
from datetime import datetime
from gpiozero import LED, Button
from warnings import filterwarnings

# GPIOZeroの警告を非表示
filterwarnings('ignore', module='gpiozero')

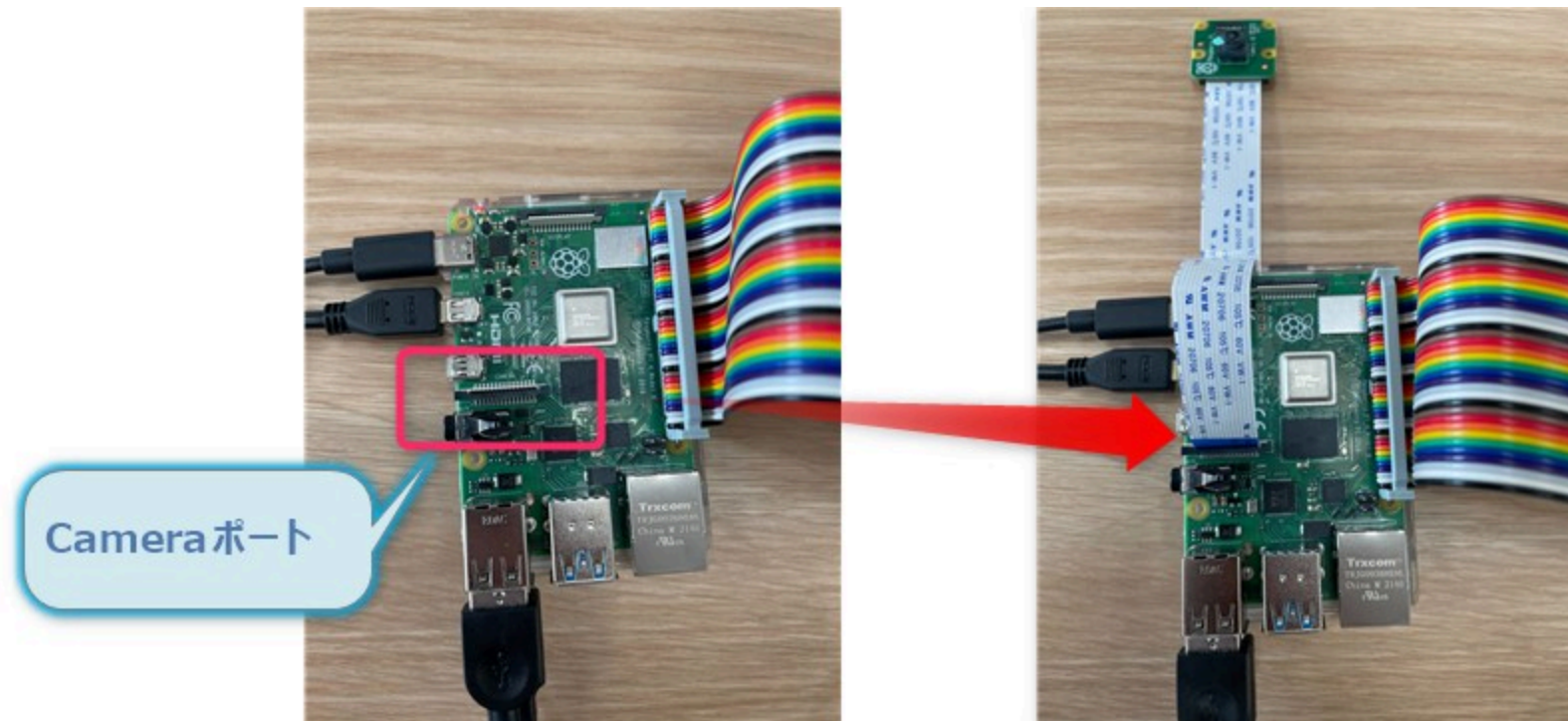
# GPIOの設定
led = LED(4)
button = Button(21)

# 無限ループでボタンの割り込みを検出
while True:
    button.wait_for_press()
    led.on()
    print(f'ON : {datetime.now().isoformat(timespec="milliseconds")}')
    button.wait_for_release()
    led.off()
    print(f'OFF : {datetime.now().isoformat(timespec="milliseconds")}')
```

カメラモジュール

カメラモジュールとは

カメラモジュールは、ラズパイで静止画や動画を撮影できるモジュールです。
カメラモジュール専用ポートに接続します。※向きに気をつけてください。



カメラモジュールのソースコード

```
from datetime import datetime
from picamera2 import Picamera2
from time import sleep

# カメラの開始
camera = Picamera2()
camera.start()

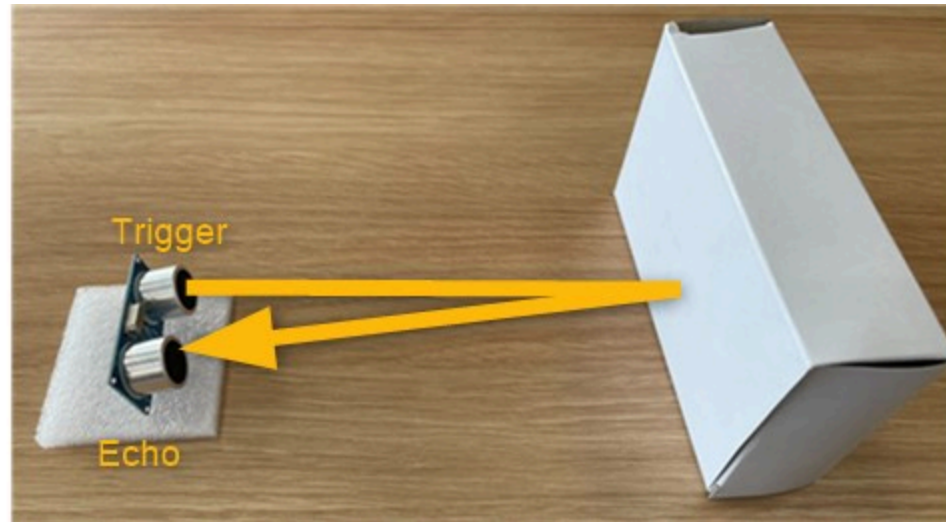
# 10回繰り返し
for _ in range(10):
    now = datetime.now().strftime('%Y%m%d%H%M%S')
    camera.capture_file(f'capture_{now}.jpg')
    sleep(1)

# カメラの終了
camera.close()
```

超音波センサー

超音波センサーとは

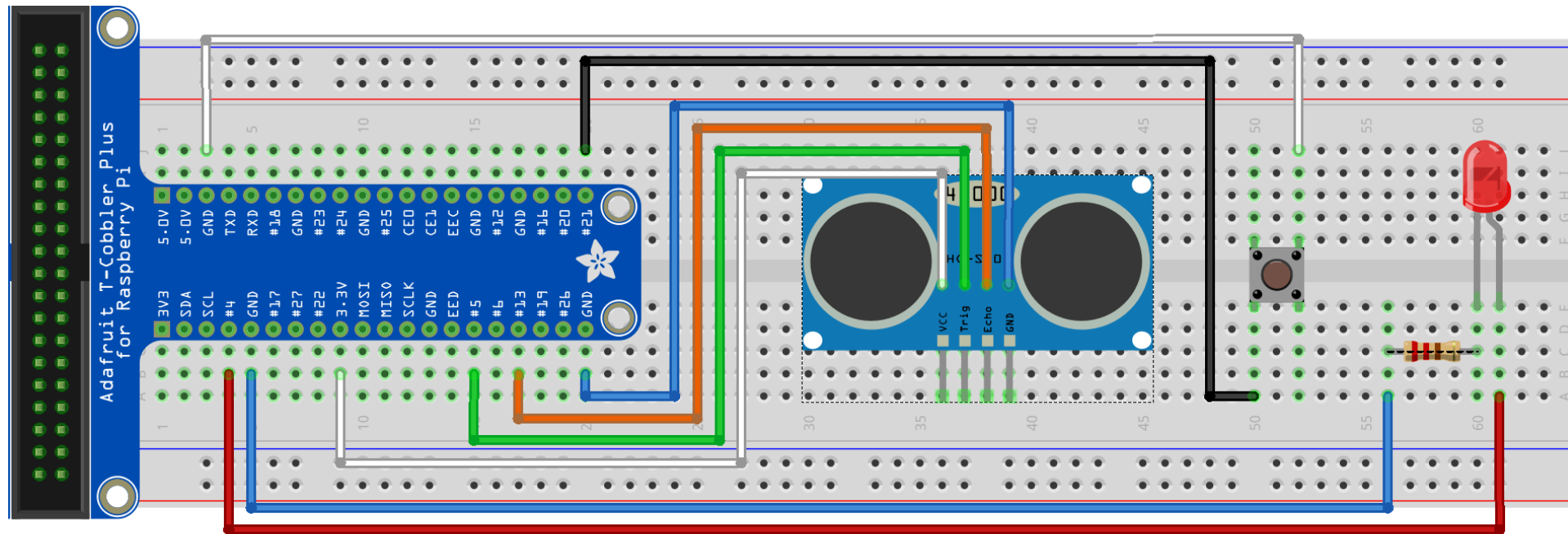
超音波センサーは、左側（トリガー）から超音波を送信し、跳ね返ってきた超音波を右側（エコー）で受信するまでに**かかった時間**を測定するセンサーです。



以下の式を計算することで、対象物までの**距離を計算**することもできます。

$$\text{距離}[cm] = \text{往復するのにかかった時間}[秒] \times \text{音速}[cm/秒] \div 2$$

超音波センサーの回路図



超音波センサー
白: 3.3V
緑: GPIO #5 Trigger
橙: GPIO #13 Echo
青: GND

タクトスイッチ
白: GND
黒: GPIO #21

LED
赤: GPIO #4
青: GND

超音波センサーのソースコード

```
from gpiozero import DistanceSensor
from time import sleep

# GPIOの設定
sensor = DistanceSensor(echo=13, trigger=5)

# 無限ループ
while True:
    # 距離の取得
    d = sensor.distance * 100
    print(d)
    sleep(1)
```

まとめ

- PythonによるGPIO制御
- Visual Studio Code
- Lチカ
- ポーリング・割り込み
- カメラモジュール
- 超音波センサー