

# 実験②…I/O性能の測定

森岡 澄夫

リスト1 GPIO Zeroを使ったGPIOトグル処理プログラム(Python)

```
import time
from gpiozero import LED

rate = 50 # Hz

gpio = LED(17)

while True:
    gpio.on()
    time.sleep(0.5 / rate)
    gpio.off()
    time.sleep(0.5 / rate)
```

## 実験②-1…ライブラリによる GPIO トグルの速度と周波数精度

### ● Python (GPIO Zero) と C 言語 (lgpio) でアクセス

ラズベリー・パイのGPIOにアクセスするための手段には、従来から複数の方法が公式・非公式に提供されています。

現在の標準的な手段は、Python上でGPIO Zero ライブラリを使う方法です<sup>(1)</sup>。詳細は特設の第4章で解説しています。ここでは、GPIO端子に“H”→“L”→“H”→“L”→…とトグルする値を出力したときの速度について調べます。

リスト1に示すのが、LEDクラスを使って信号を出力するPythonコードです。コードの中のrate値を変えることで、トグル周波数を調整できます。GPIO端子の値設定とsleepを繰り返す極めてシンプルなコードです。

また、GPIO ZeroライブラリをC言語から使う方法は2024年2月時点で不明ですが、lgpioライブラリ<sup>(2)</sup>はC言語で利用できます。lgpioライブラリを使ったGPIOトグルのコードをリスト2に示します。

リスト1とリスト2のコードを実行し、実際のGPIO端子出力の周波数などをオシロスコープで測定しました。

リスト2 lgpioを使ったGPIOトグル処理プログラム(C言語)

```
#include <stdio.h>
#include <stdlib.h>
#include "lgpio.h"

#define GPIO_PIN 17

int main(int argc, char *argv[])
{
    int freq = 1000;
    double freq_f, intvl;
    int gp;

    if (argc > 1)
        freq = atoi(argv[1]);
    freq_f = (double)freq;
    intvl = 0.5 / freq_f;
    fprintf(stderr, "freq: %f, intvl: %f\n",
            freq_f, intvl);

    gp = lgGpiochipOpen(4);
    lgGpioClaimOutput(gp, 0, GPIO_PIN, 0);

    while(1) {
        lgGpioWrite(gp, GPIO_PIN, 0); // LOW
        lguSleep(intvl);
        lgGpioWrite(gp, GPIO_PIN, 1); // HIGH
        lguSleep(intvl);
    }

    return EXIT_SUCCESS;
}
```

### ● 測定結果

#### ▶ (1) 設定値とのずれ

観測された最高周波数、平均周波数、最低周波数が、それぞれの設定値とどれだけずれていたかを図1に示します。

C言語でlgpioを使う方がやや高精度ですが、いずれも設定値が500Hzくらいまでであっても周波数誤差は10%以下という程度で、正確なハードウェア制御の用途には使えません。

比較のためにラズベリー・パイ4/3でも測定を行いました。図2に示す通りラズベリー・パイ5の方が若干設定値に近いくらいで、大きな差はありませんでした。

#### ▶ (2) 最短トグル周期 (sleepをコメントアウト)

実際の出力周波数は、GPIOアクセスにかかる時間