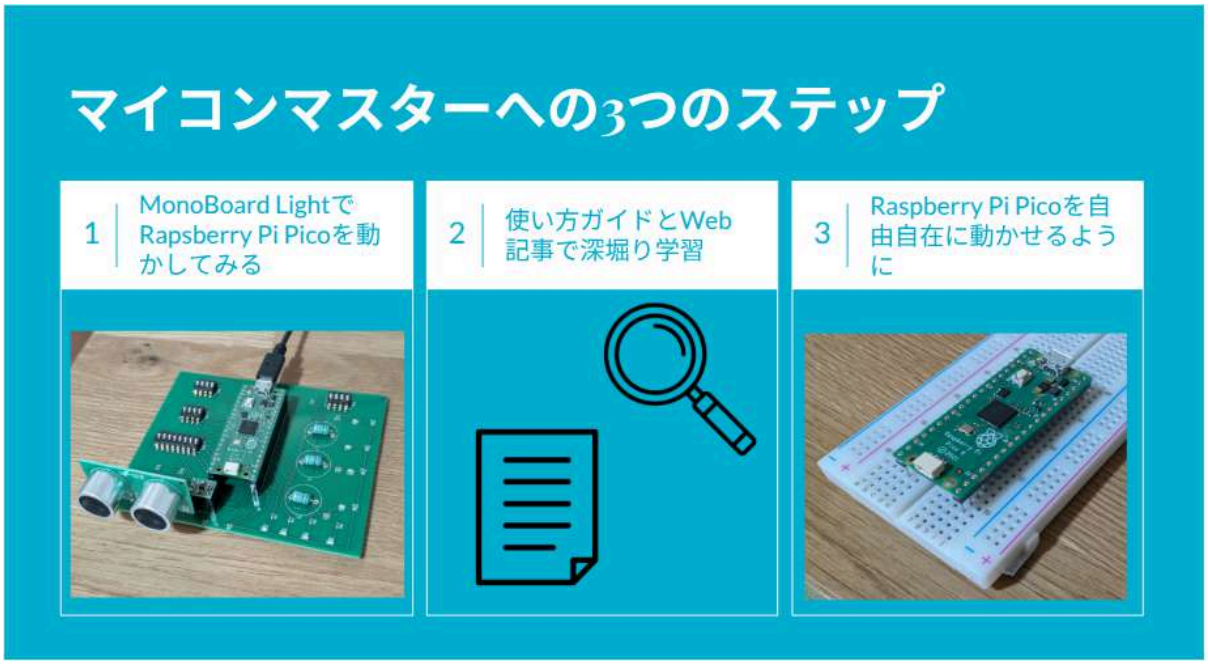
MonoBoard Light
使い方ガイド
rev 0

~はじめてのSTEAM基板マニュアル~

・目次

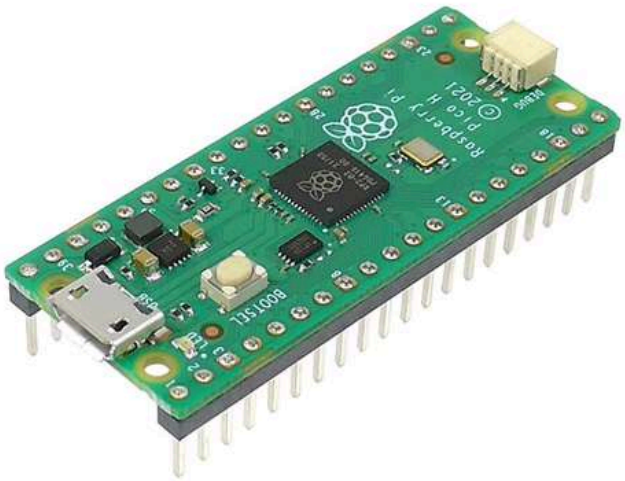
はじめに



MonoBoard Lightは、Raspberry Pi Picoと組み合わせて使うことで、初心者から中級者までが「マイコンの基礎から応用」を楽しみながら習得できるスターターボードです。電子工作やプログラミングは、最初の一歩でつまずきやすい分野ですが、MonoBoard LightではLEDやスイッチ、センサーなどの基本的な回路があらかじめ搭載されており、Raspberry Pi Picoを差し込むだけで学習をスタートできます。

Raspberry Pi Picoは「手のひらサイズの小さなコンピュータ(マイコン)」です。一般的に売られていて、入手性も高いので、電子工作によく使われるアイテムの一つです。これから、「自分の手で何かを作ってみたい」「自作〇〇に挑戦してみたい」という方は、「周囲の電子部品を直接動かすことのできる」マイコンは必須道具になってきます。マイコンを上手に制御できるようになると、出来ることの幅が広がり、柔軟に工作出来るようになります。MonoBoardを通して、マイコンの制御方法や、電子工作の知識/テクニックを学ぶことを通して、皆さん一人一人が「ものづくりに挑戦する」土壌を作っていただければと思います。

(下の画像がRaspberry Pi Picoです。スターターキットではすでにPico+MonoBoardが合体している状態になっていますが、取り外しも可能です。ただし、Pico単体で使いたいとき以外は取り外さないでください。)



学びの流れは大きく3つのステップに分かれています。まずは基板を通してRaspberry Pi Picoの動きを体感し、手を動かしながら「電子部品がどのように働くか」を直感的に理解します。次に、使い方ガイド(ホームページからDownload)やWeb記事を活用し、仕組みや応用方法を深く掘り下げることで知識を整理します。最後に、学んだ内容を活かして自分のアイデアを形にし、Raspberry Pi Picoを自由自在に操れる力を身につけます。

MonoBoard Lightの目的は、単なる「動作確認」や「作例の追体験」にとどまらず、使いながら自然に原理や応用を理解できることです。学習のゴールは「自分で考え、工夫し、作れるようになること」。MonoBoardをきっかけに、電子工作やプログラミングの世界により深く踏み込み、自分だけの「ものづくり」を生み出していく楽しさを体験してください。

0. 専門用語

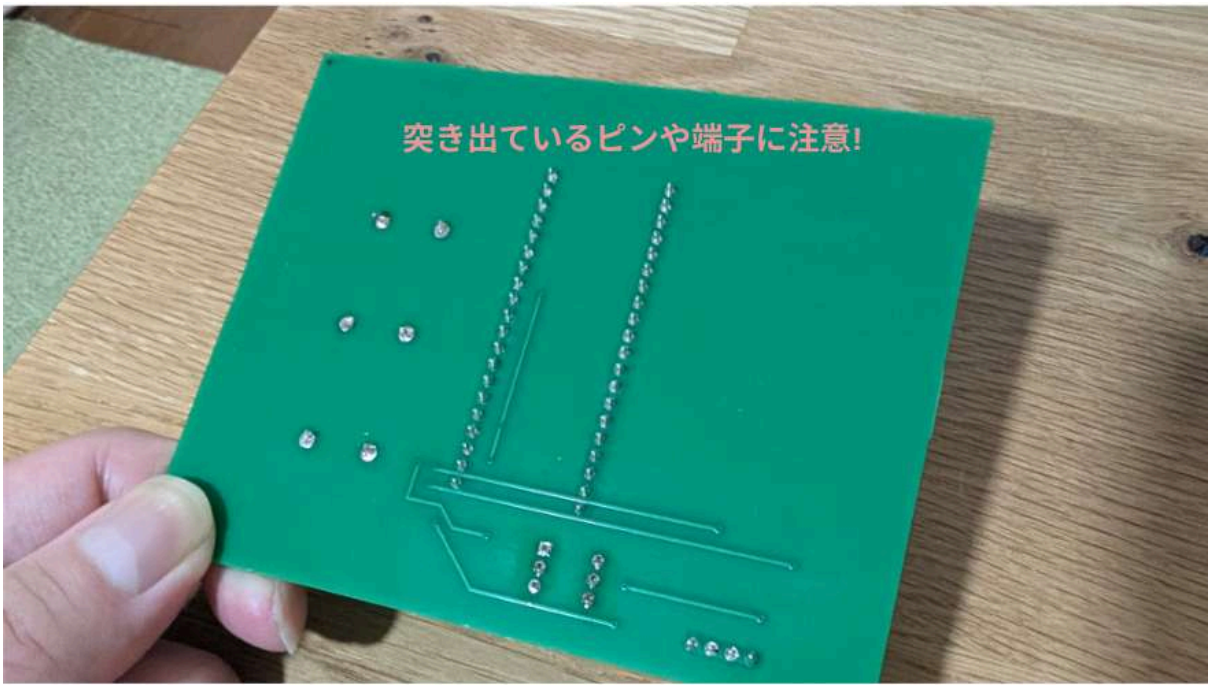
この使い方ガイドでは普段の日常会話では出てこないような電気の専門用語や部品の名前が出てきます。以下に書いている言葉がよく出てきますので、この章に立ち返って、意味を確認しながら読み進めていってください。
※専門用語に記載のある言葉には*マークがついています。

電流(I)は電圧(V)/抵抗(R) : オームの法則。I=V/R(電流=電圧/抵抗)で求められる
ソフトウェア : 機械に対して「命令や手順」を教えるプログラムを纏めたもの
実装 : 基板上に部品が取り付けられていること
モジュール: まとまった部品のひとかたまり
端子 : 電流の出入り口や、他の電気器具と接続する際の金具になるもの
分圧 : 一つの電源電圧を複数の抵抗で分けて、必要な電圧を作り出す方法
フィルタリング回路 : 特定の基準に基づいて周波数を制限する回路
デジタル信号 : 0と1の2つの状態(電圧の高低)で情報を表す信号
矩形波 : ON/OFFを繰り返す四角い波。(3.3V=ON/0V=0FF)
内部抵抗 : GPIO端子で信号を出したり、受け取ったりする際に使う抵抗のこと

1. 使い方と注意点

MonoBoard Lightは、USBケーブルを使ってPCにRaspberry Pi Picoを差し込み、基板上の回路や部品を使って学習するための教育用基板です。使い方はシンプルですが、電子機器を安全に扱うためにいくつかの注意点があります。

基板の裏面にはんだ付けされたピンや端子が突き出ています。先が尖っているため、強く押さえると指を傷つける恐れがあります。机の上に直接置くと傷がつく場合もあるので、作業の際は下敷きとなるシートやマットを敷くと安心です。



基板は電気を通す部品でできているため、水や飲み物に触れさせないようにしてください。濡れた手で触るのも避けましょう。また、通電中（USBケーブルなどで電源が入っている状態）に金属製の物を接触させるとショートの原因になります。電源を入れた状態で、動作中にむやみに触れないことが大切です。

さらに、持ち運ぶときは部品に強い力をかけないようにしてください。特にピンソケットやセンサー部分は折れやすいので注意が必要です。

これらの注意点を守れば、安全に学習を進められます。MonoBoardを通して、正しい扱い方を身につけることも電子工作の大切なステップです。

2. インストール関連

2-1. Thonnyインストールの手順
MonoBoardを動かすにはThonnyという*ソフトウェアをインストールする必要があります。
このThonnyというソフトウェア上で「命令」を出すことで、MonoBoardを自由自在に操ることが出来ます。簡単にインストールできるので、手順に沿ってゆっくりと進めていきましょう。

①公式サイトを開く
まず、Thonnyの公式ダウンロードページにアクセスします。
[Thonny公式ダウンロードページ]

②自分の**OS**に合ったインストーラをダウンロードする
ページを開くと、以下のようにOSごとのボタンがあります。
自分の「PCのOSバージョン」にあったインストーラをダウンロードしてください。
※PC OSバージョンの調べ方
Windows: WIndowsキー＋Rキーを同時に押す。→ファイル名を指定して実行→winverと入力→OKをクリック→「Windowsのバージョン情報」ウィンドウが表示されます

- Windows → Windows: Installer with 64bit-Python 3.10を選択

  ※Windows10以降のPCならほぼすべて64bit対応しているので、64bit-Pythonの方を選択してください。32ビットオペレーティングシステムの場合のみ32bit版を選択してください。

- macOS → macOS installer(最新版を選択)

- Linux → 各ディストリビューションの方法に従う

インストーラを実行する
ダウンロードしたファイルをダブルクリックして、指示に従って進めてください。

- 「Next」や「同意します」を押して進めるだけで大丈夫です。
- 特別な設定は不要で、基本はすべてデフォルトでOKです。

インストール完了後、**Thonny**を起動する

- Windowsの場合：「スタートメニュー」から「Thonny」をクリック

  ※下記のようなアイコンが追加されます。

- macOSの場合：「アプリケーション」フォルダに追加されます

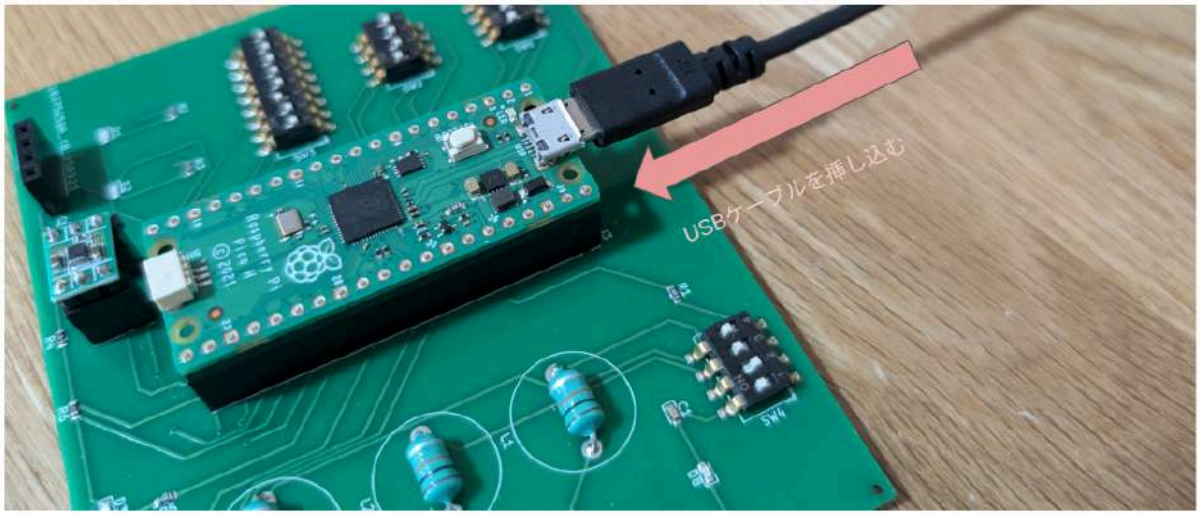
起動すると下記のようなエディター画面が表示されます。

以上でThonnyのインストールは完了です。これでThonny上でRaspberry Pi Picoを動作させる環境が作れました。

2-2. ThonnyとPicoの接続手順

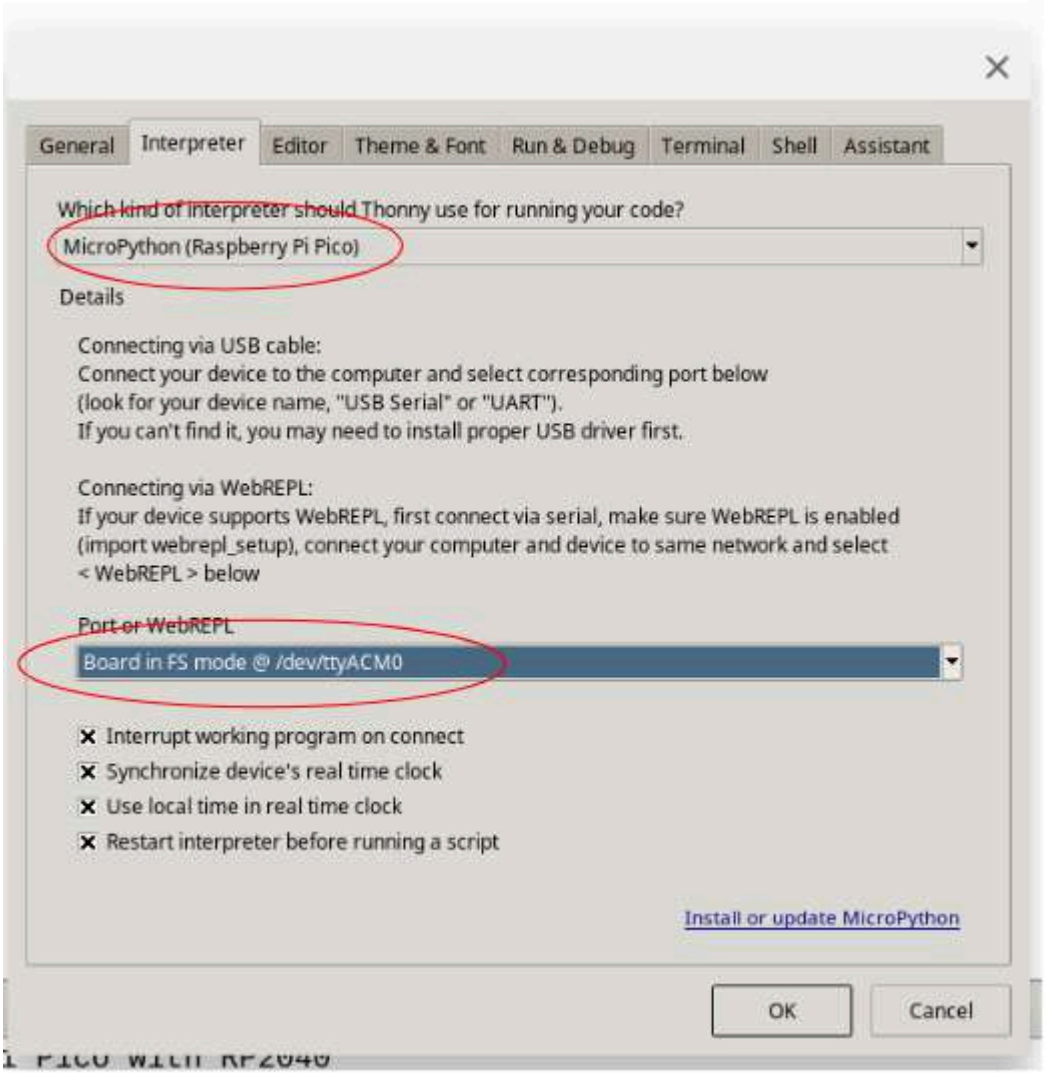
次にThonnyとPicoを接続していきたいと思います。付属のUSBケーブルをPicoのコネクタ

に挿し込んでください。また、もう一方のコネクタをPCに挿し込んでください。



Thonnyのエディタ画面の表示にあるRUN→Configure interpreterを選択してください。

下記の写真のように、インタープリタをMicroPython(Raspberry Pi Pico)、Port or WebREPLにBoard in FS mode @ /dev/ttyACM0を選択してください。



接続が成功すると、Thonny下部にShell内に表示が現れます。

これでPicoを通して、MonoBoardを制御する準備ができました。(ここまでのセットアップ、お疲れ様でした!)。次はスクリプトの実行の仕方を覚えましょう!

2-3. スクリプトのダウンロード

MonoBoardを操作する際に必要になるスクリプトをMonoMonoのホームページからダウンロードしてください。下のリンクからHPにアクセスし、Downloadと書かれたボタンをクリックしてください。

リンク: [MonoMonoホームページ]

Downloadページに飛ぶと、ページの真ん中あたりにScriptと書かれた項目があり、MonoBoard Light rev0.zipと書かれている下にDownloadボタンがあります。(赤丸で囲っているボタンです)。このボタンを押すと、必要なスクリプトがzipファイル形式でDownloadされます。

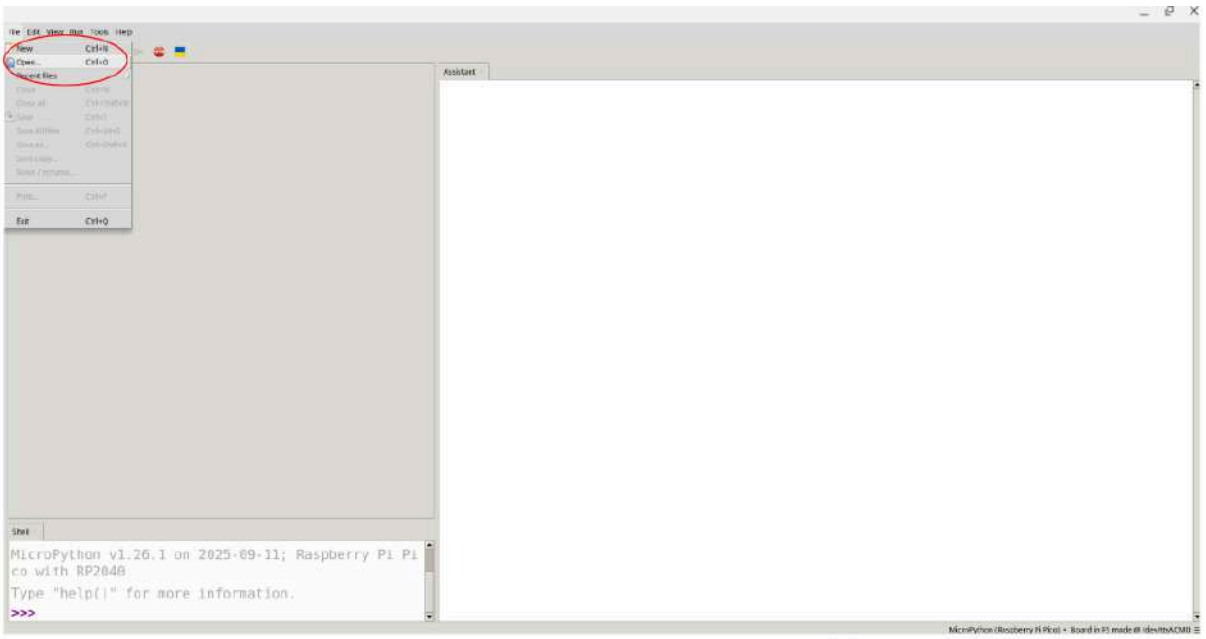
DownloadしたzipファイルをRight Clickして、「すべて展開」もしくは「すべて解凍を選択してください」。zipファイルの中のディレクトリ構成は下の写真のようになっています。

※Thonny上でスクリプトを読み込む際に、ディレクトリ情報は必要になるので、展開した先のディレクトリは覚えておくようにしてください。

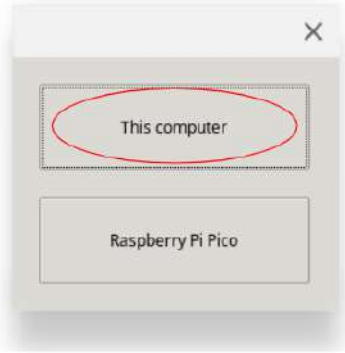
2-4. スクリプトの実行方法

ここでは、Thonny上でどうやってスクリプトを動作させるかを覚えていきましょう!

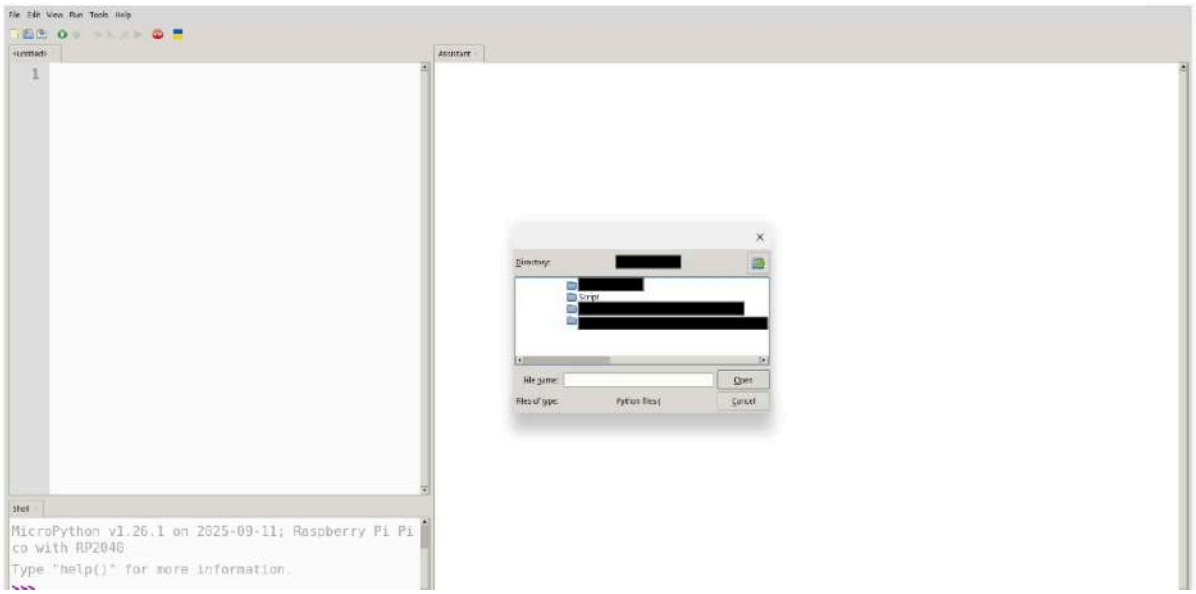
左上からFileというボタンをクリックし、次にOpenをクリックしてください。



Openをクリックすると、下の画面が出てくるので、This computerをクリックしてください。

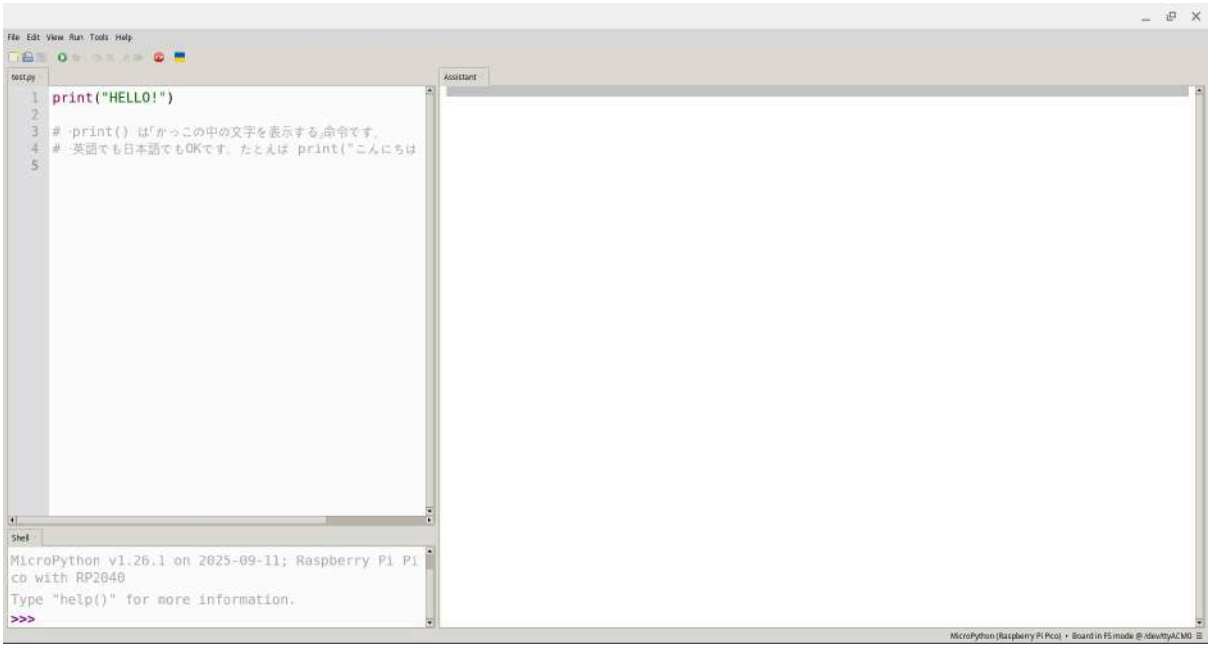


This Computorをクリックすると、PCのディレクトリが開きます。さきほどDownloadしたScriptがあるディレクトリまで移動してください。

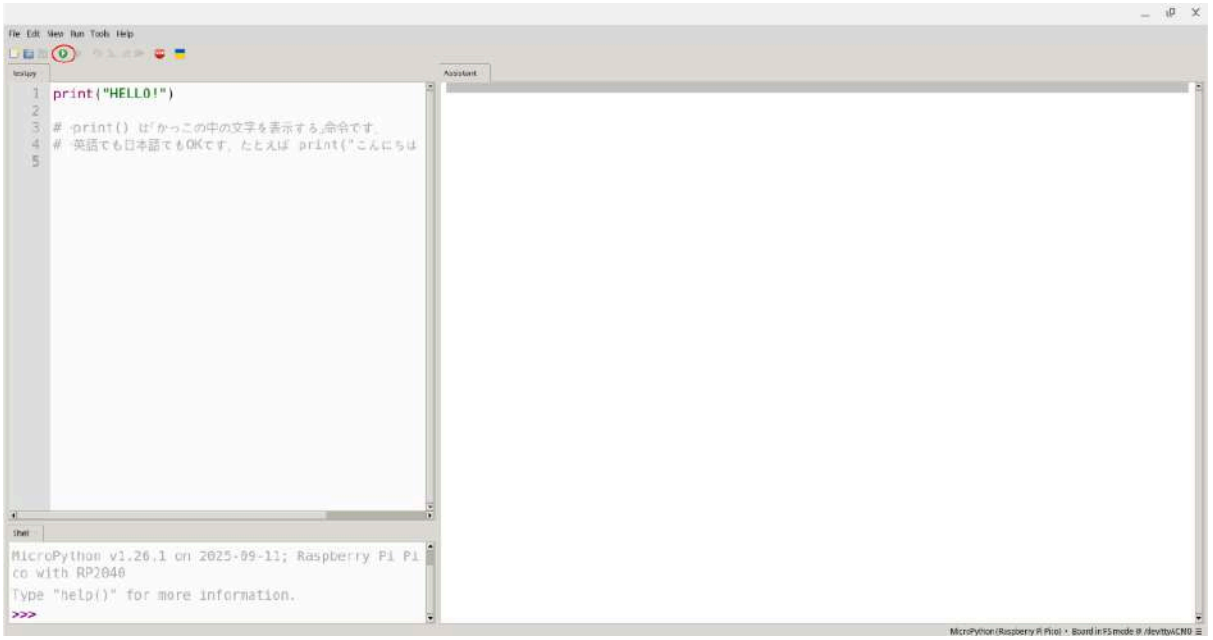


Scriptの中にtest.pyというスクリプトがあるので、クリックして開いてください。

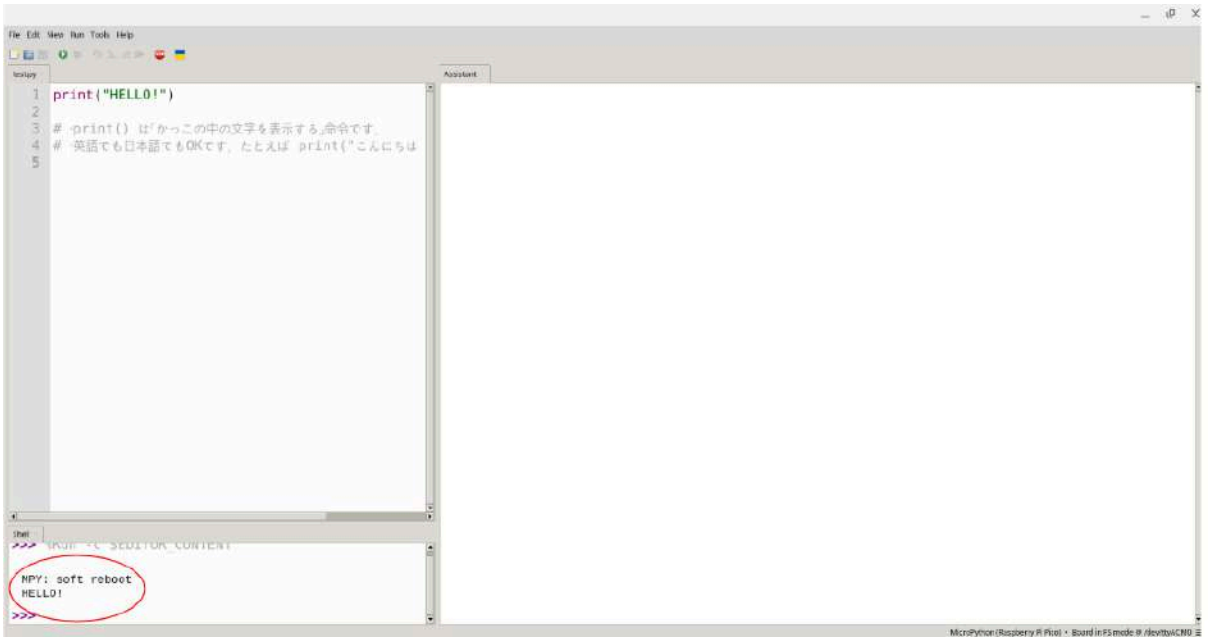
そうすると下記の画面になり、test.pyがThonny上に表示されます。



この画面になったら、「緑色のボタン」を押してください。このボタンがスクリプトの実行ボタンになります。

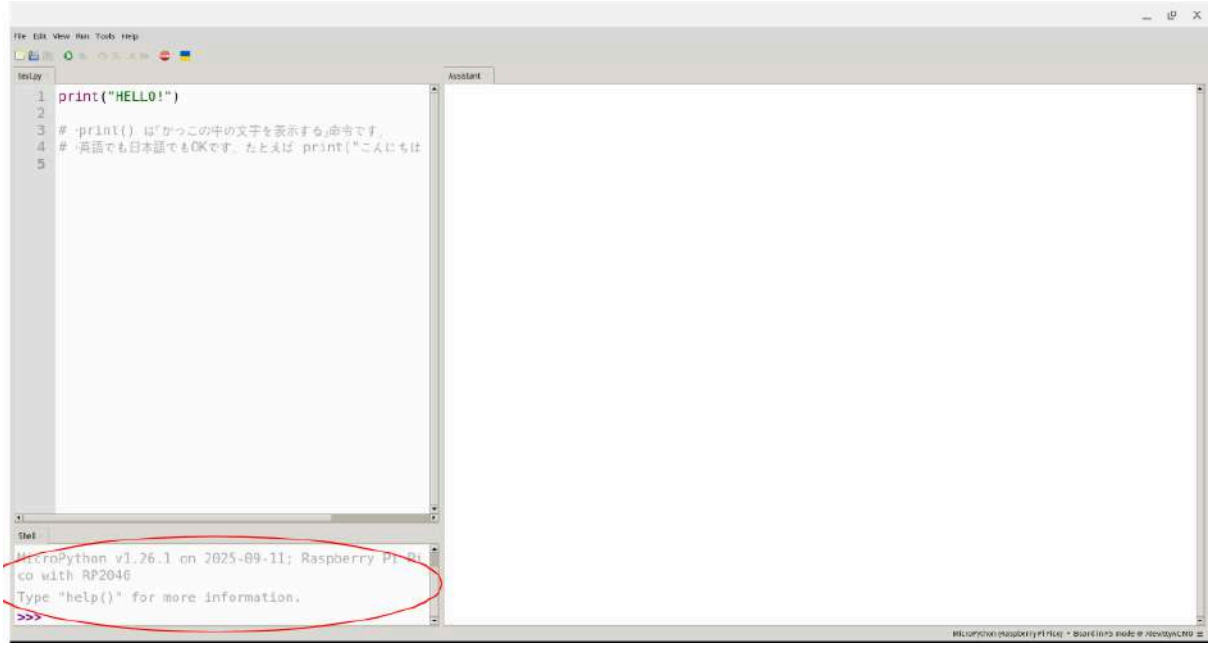


実行すると、下のShellと書かれた枠の中に、HELLO!と文字が映し出されます。(表示が出たら実行成功です!)



スクリプトを停止したい場合は、「赤色のボタン」を押してください。

ボタンを押すと、Shell内の表示(実行結果)が消え、もとの表示に戻ります。(無事、スクリプトが停止できました)



ここまでがThonny上でPythonスクリプトを動作させる一連の流れです。この使い方ガイド内で紹介するスクリプトは全て、この手順で動かすことができます。スクリプトをThonny上から消したいときは、test.pyの横にうっすらと表示されている×ボタンをクリックしてください。

※スクリプトが予期せぬ動き方をした場合や、突然止まってしまっても焦らずに、まずは赤色の停止ボタンでStopさせましょう。それでも治らない場合は、一度、USBケーブルを抜き挿しし、再び「ThonnyとPicoの接続手順」からやり直しましょう。

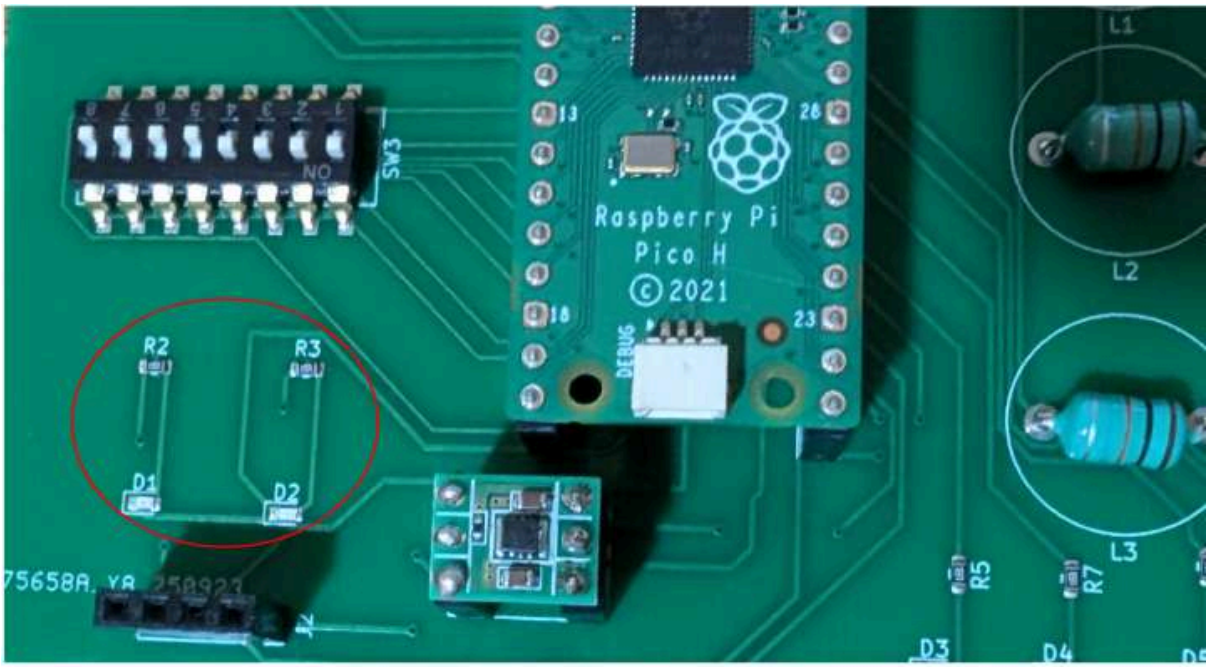
ここまでの「準備運動」、お疲れ様でした! これでMonoBoardを動かす準備が整ったので、ここからは、実際にボードを動かしながら、楽しく「マイコンの基礎～応用」を学んでいきましょう。

3. 初めての「Lチカ」をやってみよう

~LEDを光らせてみよう~

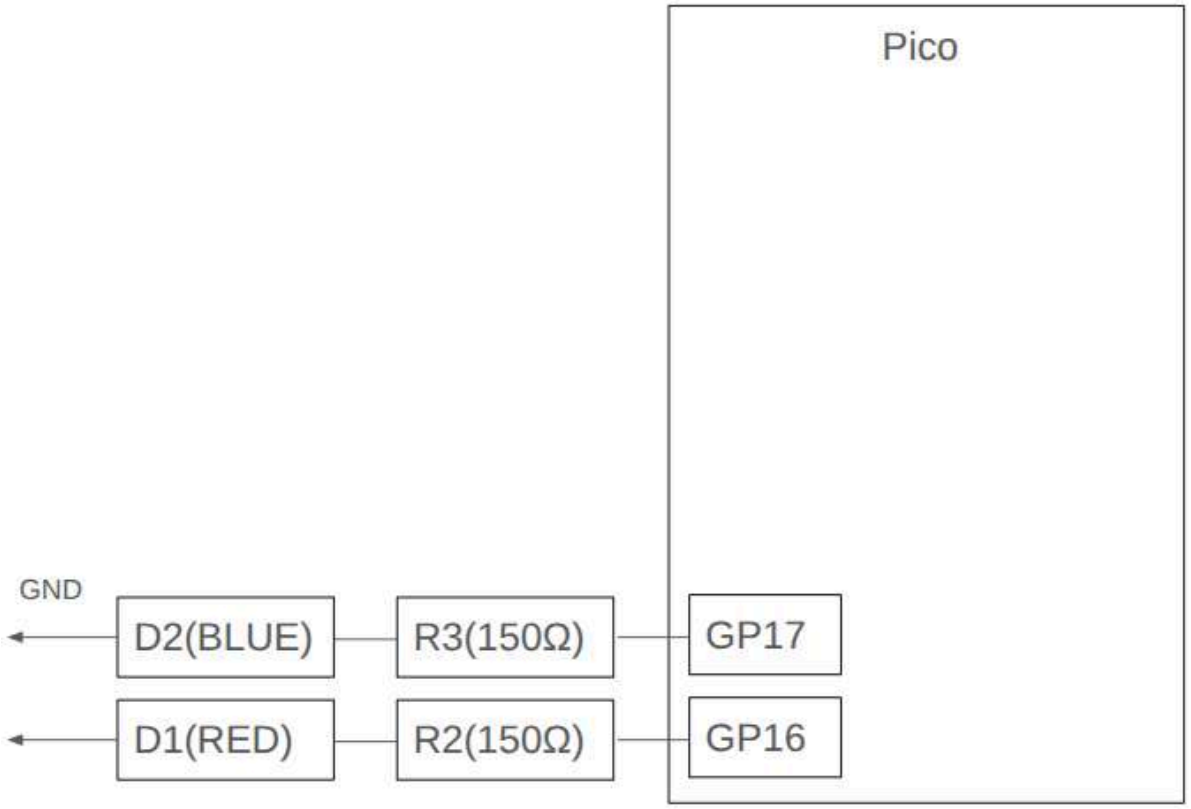
この章では電子工作入門の定番ともいえる「Lチカ(LEDをチカチカ光らせる)」をやってみましょう。LED自体は「電流を光に変換する」部品なので、多くの電流を流せばたくさん光り、少ない電流では弱い光を放ちます。この章ではMonoBoardに実装されているLEDを自由自在に操れるようになるためのテクニックを中心に紹介していきます。

MonoBoard上ではLEDを光らせる電子回路が組んであります。「百聞は一見にしかず」ということで、早速LEDを光らせてみましょう!
ボードの左下部分に注目してください。R2/R3/D1/D2と白い文字で書いてあるのがわかるでしょうか。



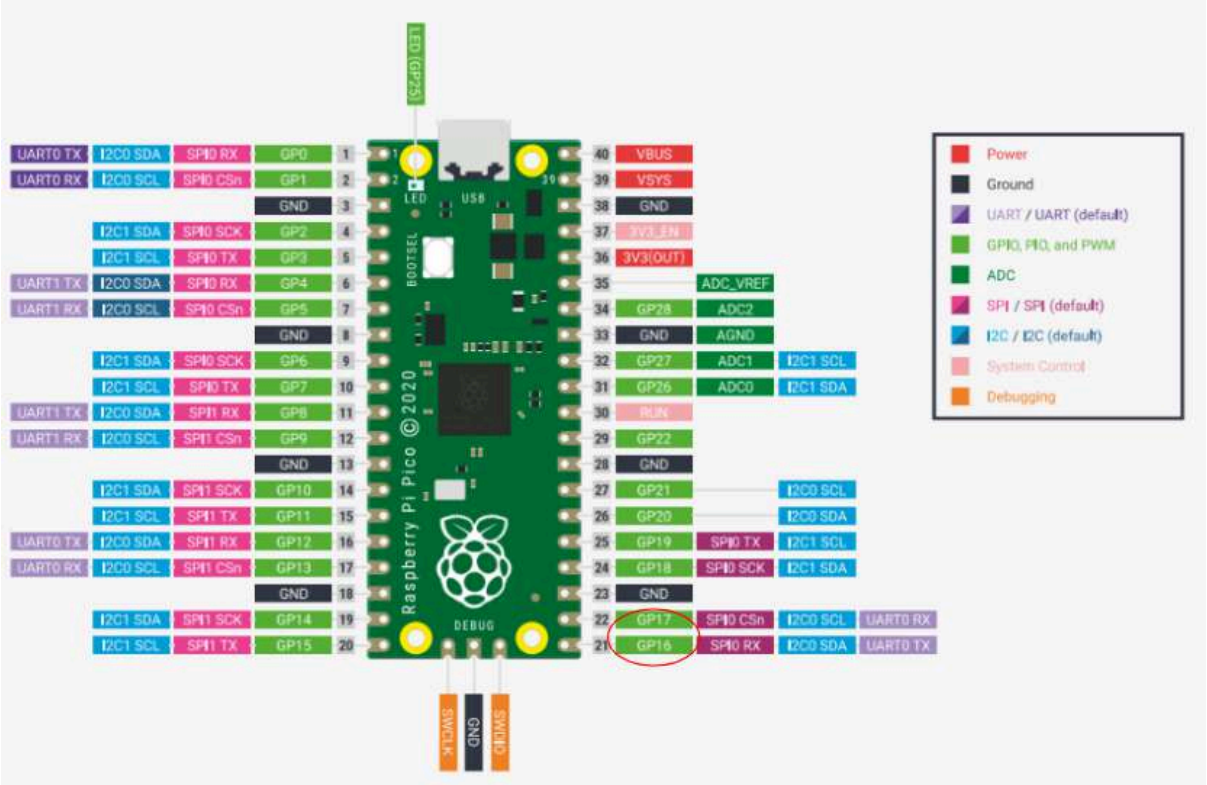
RはResistor(抵抗)の意味で、DはDiode(ダイオード)の意味です。ダイオードの中でもD1/D2はLED(Light Edmitting Diode)を指してます。部品に全く聞き馴染みが無いという方、大丈夫です。まずは部品名と役割から解説していきますね。
まずはResistor(抵抗)。抵抗は「電気の流れやすさを調整する」部品です。電子回路では流れてほしい電気の量に合うように、抵抗を選びます。部品に使われている材料によって「電気の流れやすさ」が変化し、その度合いを「抵抗値(Ω)」と呼んでいます。R2とR3には同じ150Ωが実装されています。
次に、Diode(ダイオード)。ダイオードは電気の流れを一方向だけに通す部品です。回路の逆方向に電流が流れてほしくないときに、このダイオードを使って、逆向きの電流をブロックしていま

す。(水が逆流するのを防ぐように、電子回路でも一方向しか電流が流れてほしくない場合は、このダイオードをよく使います。)
その中でも、「電気が通ると光るダイオード」がLEDというものです。D1には赤色に光るLED、D2には青色に光るLEDが実装されています。抵抗とLEDの役割がわかったところで、この二つがどうやって接続されているのかをみてみましょう。下の図はRとDとPicoの接続関係を図にしたものです。

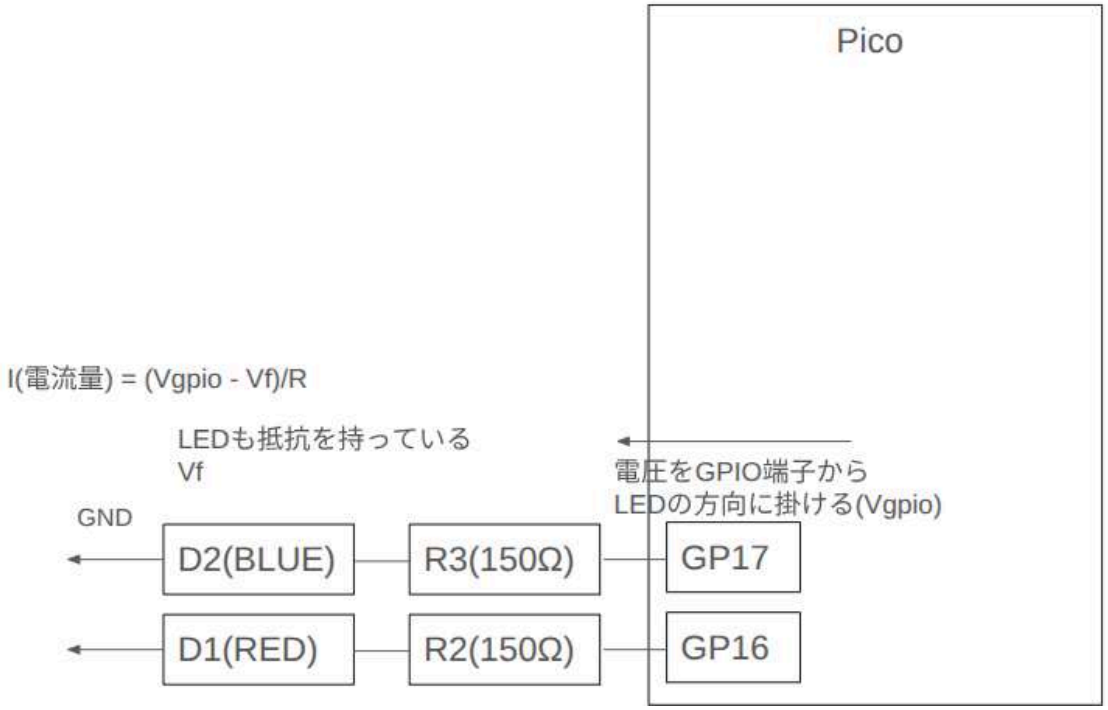


Picoの端子(GP16/17)からそれぞれ150Ωの抵抗(R3/R2)、その先にLEDが接続されている関係になっています。ここでいきなり、「GP17」「GP16」が出てきました。いったいこのGPという端子は何なのでしょうか。抵抗とダイオード/LEDに続き、「GPという端子」に関しても説明させてください。

この「GPという書かれた端子」は「PicoとMonoBoardをつなぐ入出力」としての役割を担っています。入力モードに設定すれば、外部から送られてくるセンサーの信号や電圧値を読み取ることができ、出力モードにすると、LEDを光らせたり、電圧を供給したりすることが出来ます。
こういった役割の端子を一般的にはGPIO端子といい、よくGP〇〇という表記のされ方をします。
今回はPico上のGP16端子とGP17端子を使って、MonoBoardを制御していきます。
下の図はPicoの仕様書から抜粋してきたピン配置図(どこにどのピンが実装されているかを示す図)なのですが、赤枠で囲ったGP16とGP17を使っています。

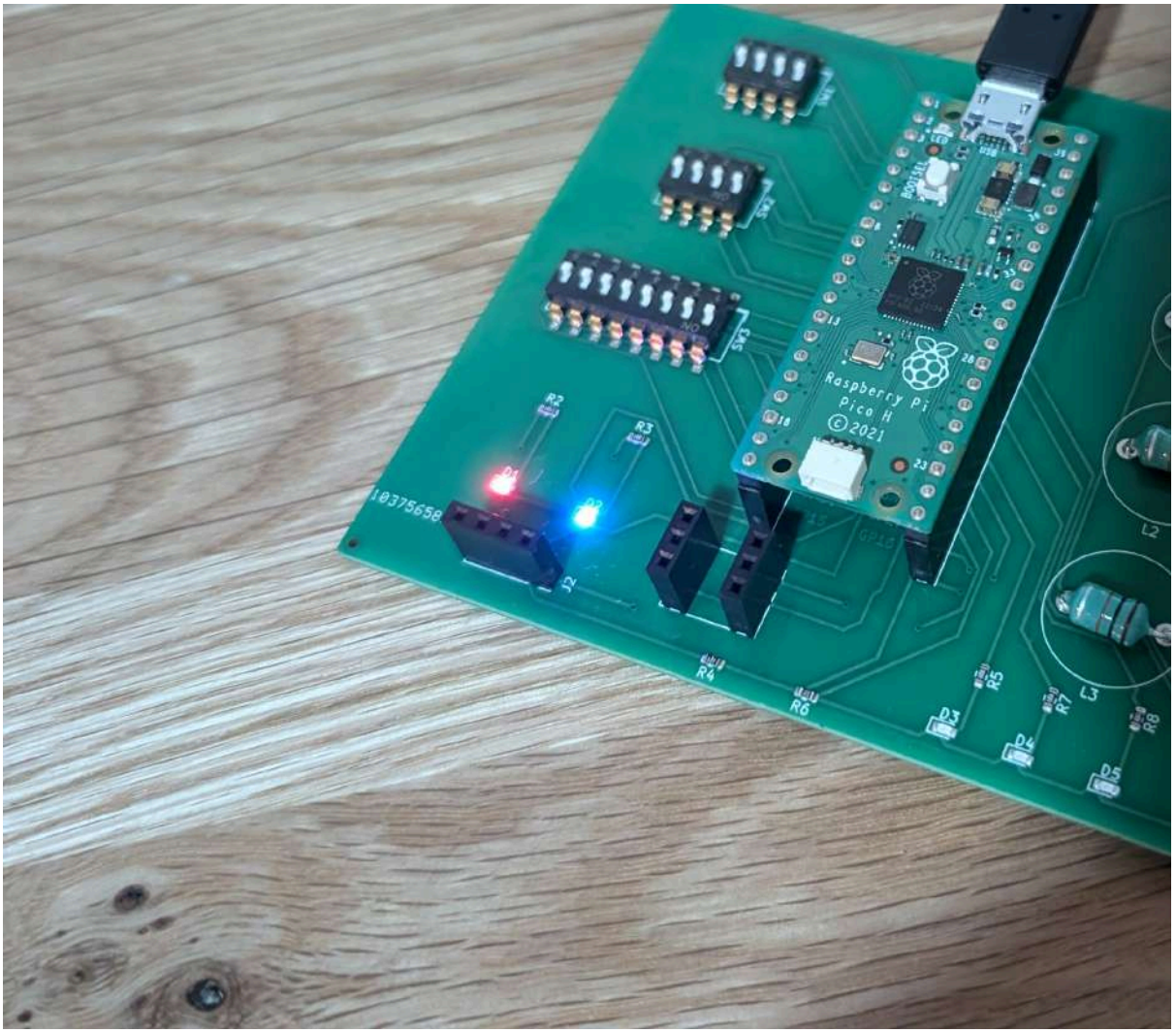


GPIO端子からLEDを制御するとしても、過大な電流が流れるとLEDが壊れてしまう可能性があるので、さきほど説明したように、間に抵抗を接続して、「LEDに流れる電流をちょうどよく」調整しています。
では、回路上にどのような電圧/電流が流れているのかを下の図をもとに説明します。
GPIO端子は「電圧を発生させる機能」を持っており、3.3Vの直流電圧を出力することが出来ます(この電圧をVgpioとしましょう)。また、LED自身も「LEDの光が光り始める電圧(しきい値)」をもっており、この電圧を超えないと、LEDの光は点灯しません(この電圧をVfとしましょう)。回路全体に流れる*電流(I)は電圧(V)/抵抗(R)で求められることができるので、抵抗が大きければ大きいほど、流れる電流値は少なくなります。抵抗値(R)によって、LEDに流れる電流量を調整することが出来ます。



だーーーっと難しい説明が続いたと思いますが、安心してください。(ここまで説明を読んでくださって有難うございます。)
説明が全て理解できなくても全く問題ないです! ここからは実際にLEDを光らせながら、体験的にLEDの動作を学んでいきましょう。

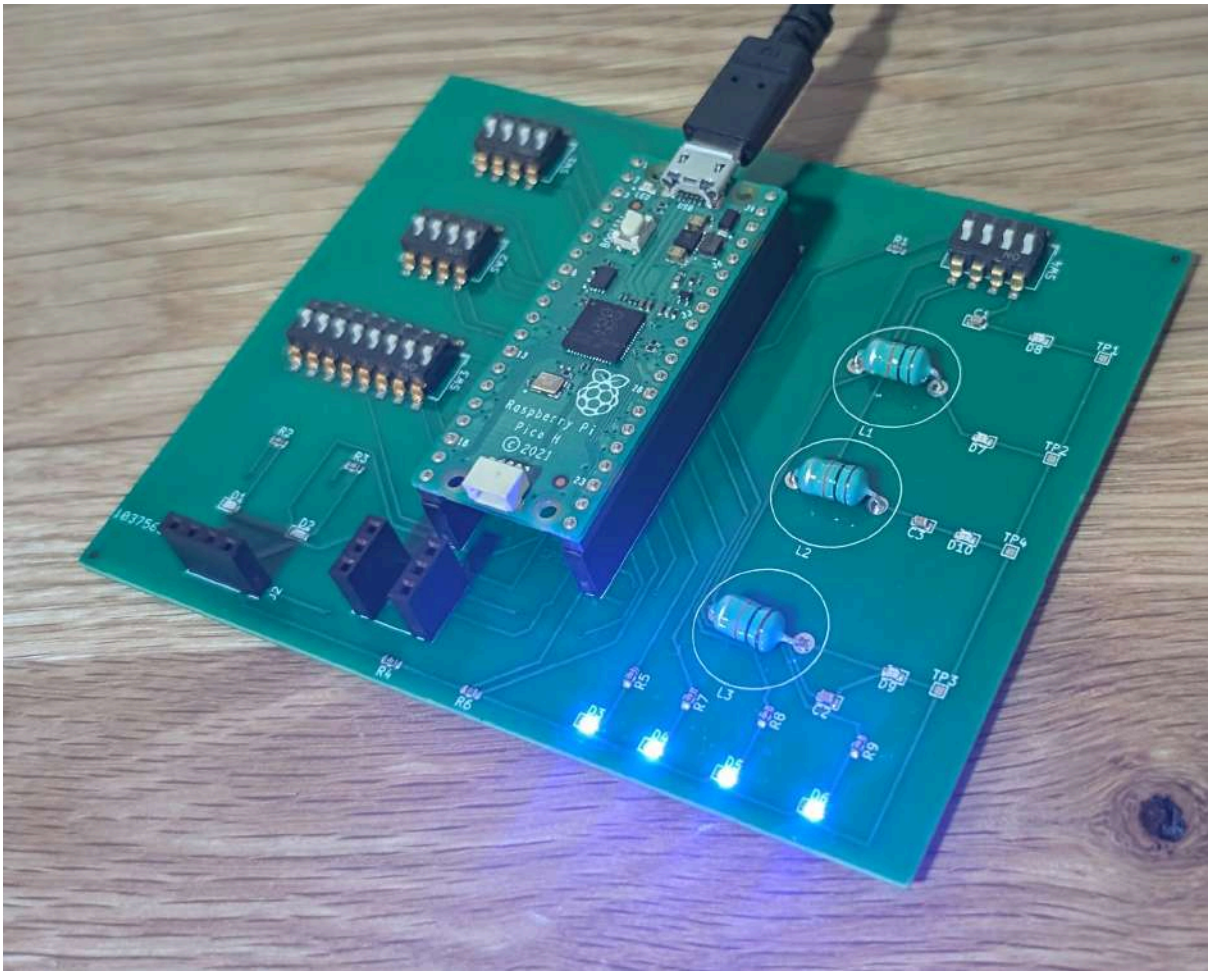
Monoboard LightをPCに接続して、led1.pyを実行してください。スクリプト1の中身は「GPIO(GP16/17)から3.3V(Vgpio)の電圧を発生させて、LED(D1とD2)が10秒間点灯した後に、消灯する」仕組みになっています。



led1.py 実行結果

どうですか、期待通り光ったでしょうか？
Vgpio=3.3Vとして、Vf=2.3V(LEDが光り始めるまでの電圧)、R=150Ωと考えると、
(3.3-2.3)V/150Ω=0.0066…A、6.7mA程度の電流値がそれぞれの回路に流れていることになります。
(初めてLEDを光らせた方、おめでとうございます!! マイコンマスターへの大きな第一歩です!)

動作原理がわかったところで、ついでに、右下に配置されている「D3～D6のLEDの点灯」も確認しておきましょう。D3～D6も同様に、それぞれ150Ωの抵抗(R5/R7/R8/R9)に接続されており、R5-GP21, R7-GP22, R8-GP26, R9-GP27の対応関係でGPIO端子とも接続されています。
led2.pyを実行してください。led1.pyと同様にD3～D6のLEDが10秒間点灯してから、消灯する仕組みになっています。



led2.py 実行結果

皆さん、もうLEDの使い方はマスターできましたね!
光らせることに成功したら、下の練習問題をやってみてください。次章以降の内容が分かりやすくなります。

練習問題1
led2.pyに書いてあるFREQとDUTYという変数はそれぞれ、GPIO端子から出力する波の「周波数と明るさ」を調整するパラメータになっています。以下の1,2の問題をやってみることで、LEDの光り方にどのような影響を与えるのかを理解できるはずです。

1. スクリプト2に記載のあるFREQ=1000の値を10～10000の間の好きな値に設定して、LEDの光り方がどう変化するのか確かめてみてください。

2. スクリプト2に記載のあるDUTY=10000の値を1000～10000の間の好きな値に設定して、LEDの光り方がどう変化するのか確かめてみてください。

値を変えていくうちに、FREQとDUTY2つの変数がLEDの光り方にどう影響与えるのかがわかってきたのではないでしょうか。

練習問題2

led3.pyというファイルを実行してください。このスクリプトは「LEDを１個ずつ順番に光らせる」スクリプトです。pins = [21, 22, 26, 27]に変更して、何が変化するのかを確かめてみてください。

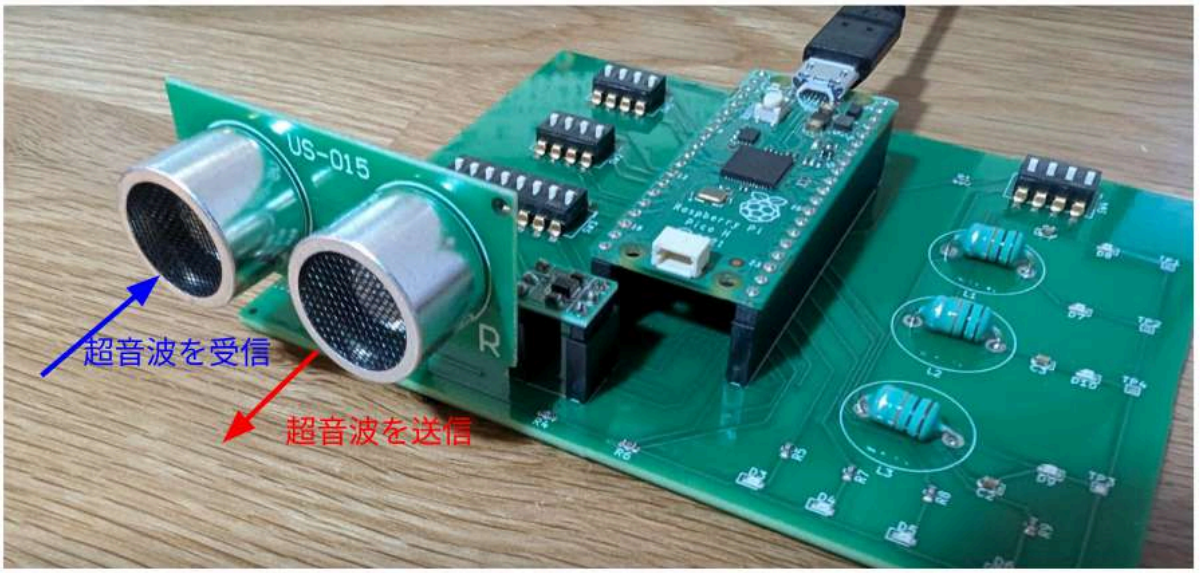
練習問題3
led4.pyは「D1/D2とD3/D4/D5/D6」が交互に点灯するスクリプトです。A_ON_TIME(GP16/17)を出力する時間、B_ON_TIME(GP21/22/26/27)を出力する時間を変化させて実行してみましょう。

これで皆さんはもうLEDマスターです。練習問題で使ったスクリプトを応用して、オリジナルスクリプトにチャレンジしてみてください!

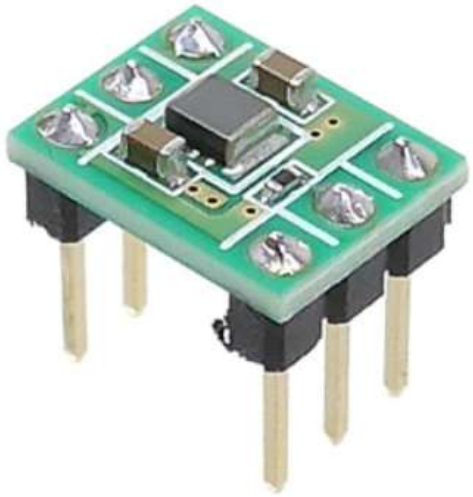
4. 測距センサーを動かしてみよう

～センサー入門編～

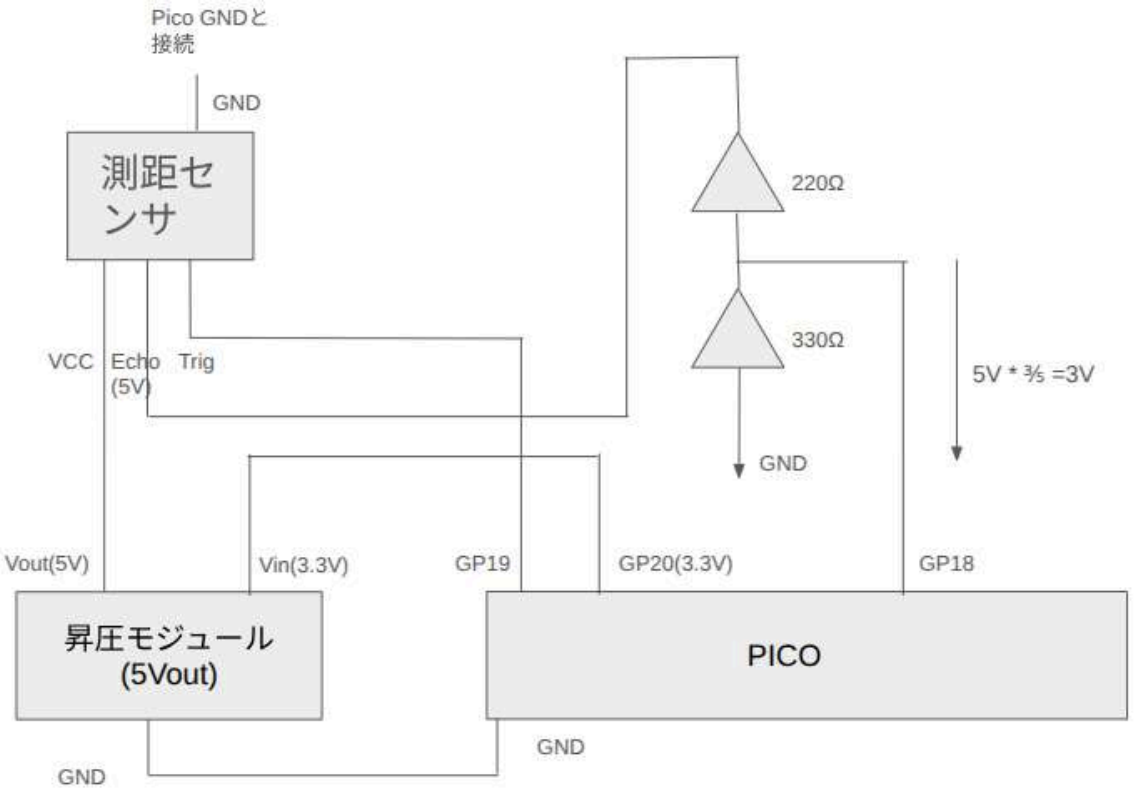
我々の身近には様々なセンサーがあります。今回はその一つ、測距センサーを動かして、その原理や仕組みを理解していきましょう。今回使う測距センサは「超音波を使って距離を測定する」仕組みになっています。モジュールの正面に「超音波を送信する部品」と「超音波を受信する部品」がつけられており、内部のチップが「反射時間から距離を算出」してくれる仕組みになっています。



この測距センサは簡単に距離が測れるので、電子工作でも重宝される*モジュールなのですが、Picoと組み合わせて使用する際にいくつか注意すべき点があります。LEDの時と同様に、GPIO端子の3.3V電圧を使用してしまうと、超音波の強度が下がってしまうので、測定できる距離が短くなってしまいます。また、3.3V電圧ではセンサーに搭載されているICチップの処理に時間がかかってしまうので、場合によっては精度の低い測定結果になってしまいます。以上の理由から、今回はPICOと測距センサの間に「入力された電圧(3.3V)を、より高い電圧(5V)に変換する部品」である、昇圧モジュールを実装する仕組みにしています。
昇圧モジュールは下の写真に示している部品で、測距センサが安定して動作するために必要な電圧(5V)を供給する役割を担っています。



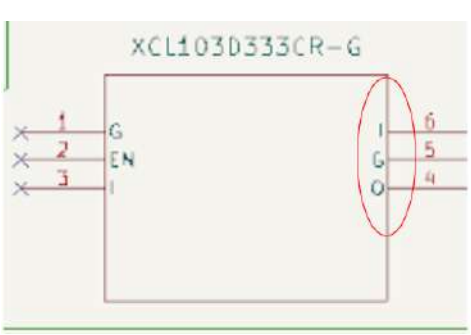
PICO-昇圧モジュール-測距センサの接続を下の図に表しています。この図の実装に従って、必要な部品を実装していきましょう!

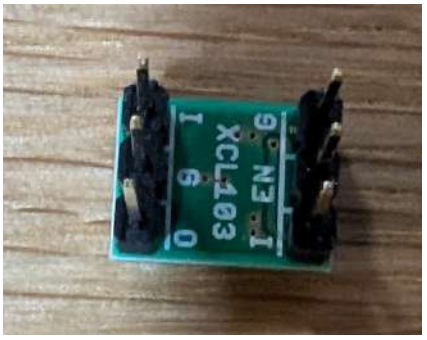


概要を説明するために図を載せましたが、今の段階で理解できていなくても大丈夫です。図の内容としては。PICOのGPIOピン(GP20)から昇圧モジュールに対して3.3Vに電圧を入力し、昇圧モジュール内でVout(5V)を生成し、その出力(5V)がセンサに供給されている仕組みになっています。また、このセンサはTrigger-Echo式という仕組みで距離を求めており、Triggerで超音波を発射して、Echoでその反射を検出しています。それぞれ、TriggerピンをGP19に、Echoピンを抵抗

経由でGP18に接続しています。(なぜEchoピンのみ抵抗が必要なのかは後の章で説明させてください)。原理が理解できたところで、早速、センサーの実装をやってみましょう。

・測距センサーの実装
まずは、昇圧モジュールをU1と書かれたピンヘッダに挿してください。
※挿し込む際は、下の回路図のように右側にIGOのシルクの並びがくるように挿し込んでください。



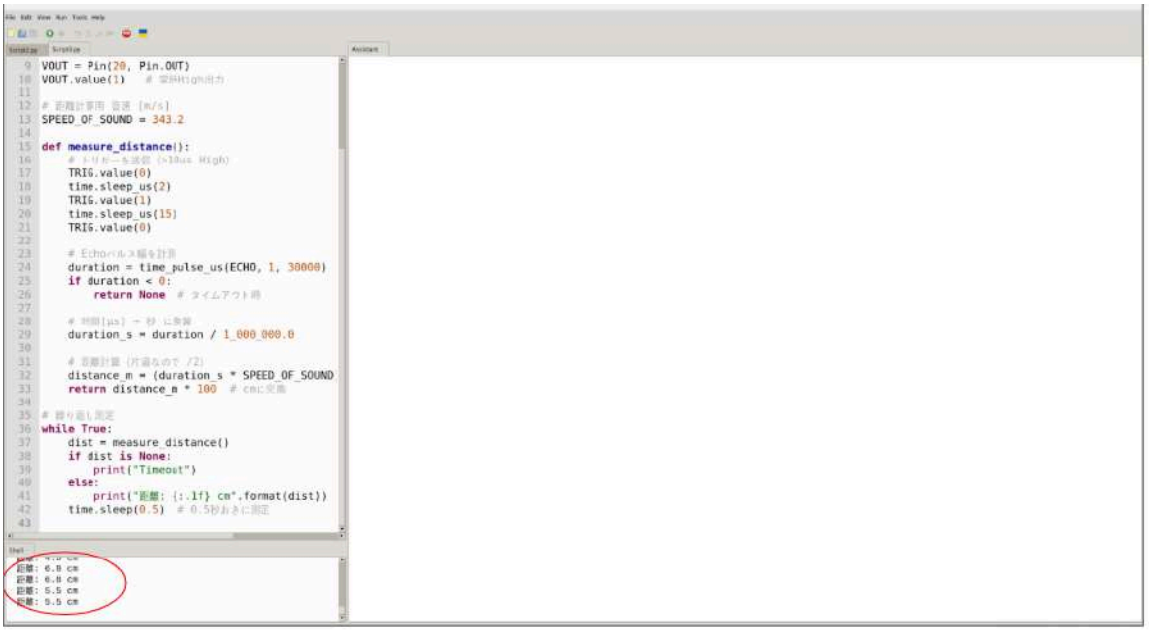


※IGOと書いてあるシルクのピンが右側のピンソケットに挿さるようにしてください。

次に、付属品のセンサーをピンソケット(J2)に写真の向きになるように挿してください。

全てのパーツが実装できたら、スクリプトのsensor1.pyを動かしてみて下さい。
※実行結果にTimeoutが常に表示されている場合は、一度スクリプトを停止し、昇圧モジュールとセンサーがコネクタにきちんと挿さっているかを確認してみてください。(挿し直したり、リトライするのがオススメです。)

正しく実行できると、下記の写真のように「物体との距離(cm)」がShellの中に表示される仕組みになっています。センサーの前に手をかざしてみたり、物体の距離を調整したりしてみてください。距離がリアルタイムで測れていることがわかると思います。



```
VOUT = Pin(20, Pin.OUT)
VOUT.value(1)   # 常時High出力

# 距離計算用 音速 [m/s]
SPEED_OF_SOUND = 343.2

def measure_distance():
    # トリガーを送信 (>10us High)
    TRIG.value(0)
    time.sleep_us(2)
    TRIG.value(1)
    time.sleep_us(15)
    TRIG.value(0)

    # Echoパルス幅を計測
    duration = time_pulse_us(ECHO, 1, 30000)
    if duration < 0:
        return None  # タイムアウト時

    # 時間[μs] → 秒 に変換
    duration_s = duration / 1_000_000.0

    # 距離計算 (往復なので /2)
    distance_m = (duration_s * SPEED_OF_SOUND) / 2
    return distance_m * 100  # cmに変換

# 繰り返し測定
while True:
    dist = measure_distance()
    if dist is None:
        print("Timeout")
    else:
        print("距離: {:.1f} cm".format(dist))
    time.sleep(0.5)  # 0.5秒おきに測定
```

```
距離: 6.8 cm
距離: 6.8 cm
距離: 5.5 cm
距離: 5.5 cm
```

sensor1.py 実行結果

測距センサーを動かせたところで、その理解を深めるためにも以下の3つの練習問題に取り組んでみてください。

練習問題1

センサーとの距離だけでなく、受信面(トランスデューサ(筒のような部品))に対して、「同じ距離で上下」に動かしてみてください。結果はどうなるでしょうか？

練習問題2
time.sleep(0.5)の(0.5)部分を(0.1)や(1.0)に変えると表示の動き方はどう変わるでしょうか？(0.1秒、1.0秒)

練習問題3
time_pulse_us(ECHO, 1, 30000)の30000(3000μs=0.03秒)を60000(0.06秒)に変化させると測定距離がどうなるでしょうか？
※30000μsは「超音波の往復時間をいつまで待つのか(どれくらい先の対象物まで計るのか)」を設定している値です。倍にすると倍の距離まで測れるのでしょうか。

・センサーの仕組みについて(解説編)
せっかくセンサーを動かせたので、その原理についても少し解説させて下さい。
ちらっと紹介したTrigger-Echo方式ですが、実際のところは、GP19がTrigger端子に電流を流すと同時に、センサーの送信側が稼働を始めます。そして、超音波を発射し、その反射を受け取った受信面がEcho端子を介して、GP18に測定結果を出力しています。ここで注意点が一点あります。Echo端子は「入力電圧と同じ電圧を出力する」ことになっているので、Echo端子からの出力は5Vになっています。
(「Echo端子とGPIO端子の間にだけ抵抗が接続されている(Trigger端子とGPIO端子は直接接続されている)」と説明しましたが、まさにこの特徴がその理由になっています)
GP18に限らず、GPIOが受信できる電圧は3.3Vまでなので、5→3.3Vまで電圧レベルを落としてあげる必要があります。そこで、下記の図で示すようにR4(220Ω)とR6(330Ω)を使って、3Vに*分圧した電圧をGP18に入力しています。
※センサー近くにあるR4(220Ω)とR6(330Ω)を探してみてください。

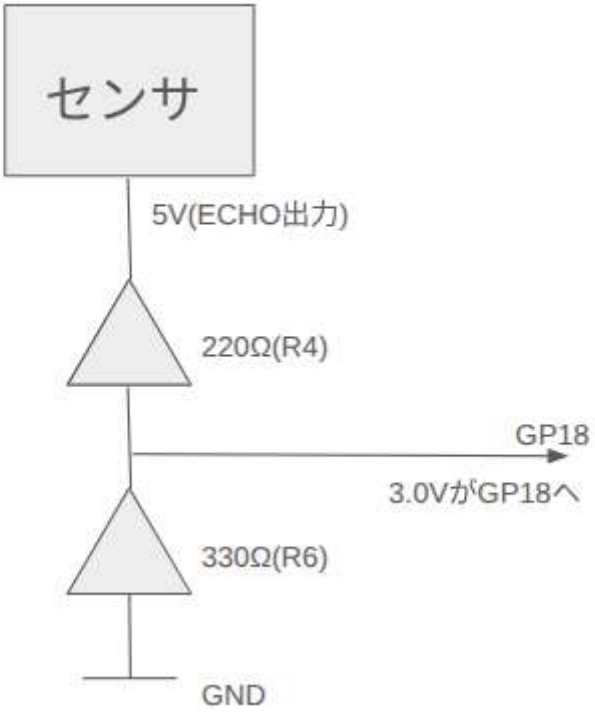


図1: ECHOの出力(5V)を分圧して、GP18へ入力

一般的な分圧の式は下記の図2に示すとおりです。Vin=Echo端子、R1=R4, R2=R6と置き換えると上の式↑が導出できます。

一般的な2つの抵抗を使った分圧の計算式は以下の通りです：

$$V_{out} = V_{in} \times \frac{R2}{R1 + R2}$$

- V_{in}：センサの出力電圧（例：5V）
- $R1$：上側の抵抗（Vinと接続）
- $R2$：下側の抵抗（GNDと接続し、出力はこのR2との接点）

図2: 分圧の計算式

どうでしょうか、センサーの原理が「動かす体験を通して」わかってきたのではないでしょうか。

5. 距離に応じてLEDを光らせてみよう

~制御の組み合わせ編～

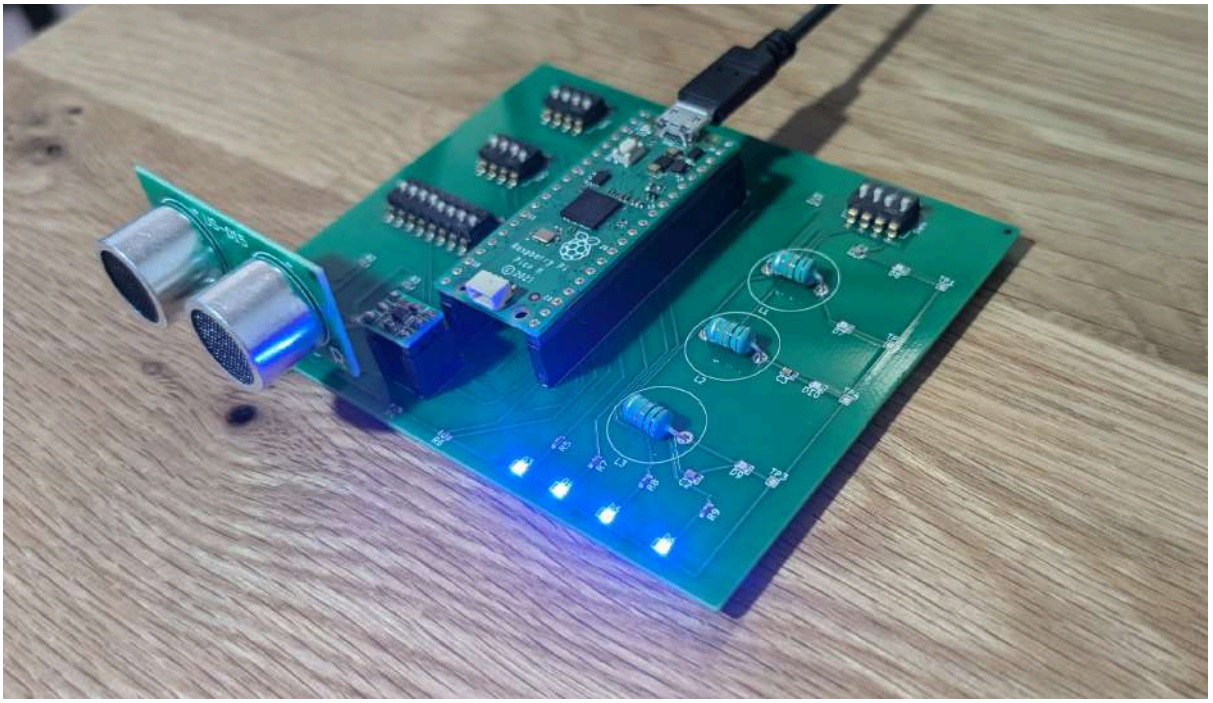
さあ、ここまで来た皆さんは既に「LED」と「センサー」の使い方に関して、だいぶと慣れてきたのではないでしょうか。この章では、第三章と第四章で学んだ知識を組み合わせて、「測定した距離

に応じてLEDが光る」制御を作りたいと思います。この制御を通して、PICO(マイコン)の基本的な制御と、値の入出力関係を学んでいきましょう。

・条件分岐
作り始める前に1個だけ紹介しておきたいのが、条件分岐という考え方です。条件分岐はプログラミングにおいて重要な考え方なので、ここでしっかり習得しておきましょう。条件分岐は、プログラムの中で、「もし〇〇であるなら～する」という判断(分岐)をさせる仕組みです。測距センサーに限らず、センサー全般の制御で、この考え方が非常に重要になります。
例えば、距離センサーであれば、「測定値」に対して「ある条件を満たしたら、次の動作をする」と判断できます。「もし距離が10cmより短いなら、LEDを一つ点灯させる」「もし距離が10cm～20cmの間であるならば、LEDを二つ点灯させる」など、条件→行動のルールによって電子回路に”判断力”を与えることが出来ます。よりイメージを掴んでもらうために下の練習問題をやってみてください。

練習問題
branch.pyを実行してください。Xの値が3より大きい or Xの値が3以下(条件)でLEDを点灯するか/消灯するか(分岐)が選択できるスクリプトです。変数Xの値を変化させて確かめてください。

・センサーxLEDの実装
回路図や実装部品に関しては、第三章と第四章で説明したので省略します。先程、学んだ「条件分岐」がどう活用されるのかを学んでいきましょう。条件分岐を含めた実装例として、測距センサーの値が「10cm以下であればD3」、「10cm～20cmの範囲であればD3とD4」、「20cm～30cmの範囲であればD3とD4とD5」、「30cm以上であればD3とD4とD5とD6」を点灯させるスクリプト(sensor_led_1.py)を作ってみました。早速、実行してみてください。



sensor_led_1.py 実行結果

センサーに手を近づけてみると、先程の条件に合わせてLEDの点灯数が変化するのがわかるかと思います。

練習問題
sensor_led_1.pyの下記の距離(10/20/30)を変更して、自分の好きな距離でLEDが点灯するように工夫してみましょう。

```
    if dist_cm <= 10:
        LEDs[0].value(1)
    elif dist_cm <= 20:
        LEDs[0].value(1)
        LEDs[1].value(1)
    elif dist_cm <= 30:
        LEDs[0].value(1)
        LEDs[1].value(1)
        LEDs[2].value(1)
```

MEASURE_INTERVALL_S=0.2(センサーの反応速度)を変更してみよう。(0.05(毎秒20回)、1.0(毎秒1回)と変化させたときに、LEDの反応速度も速くなるのか確かめてみよう)

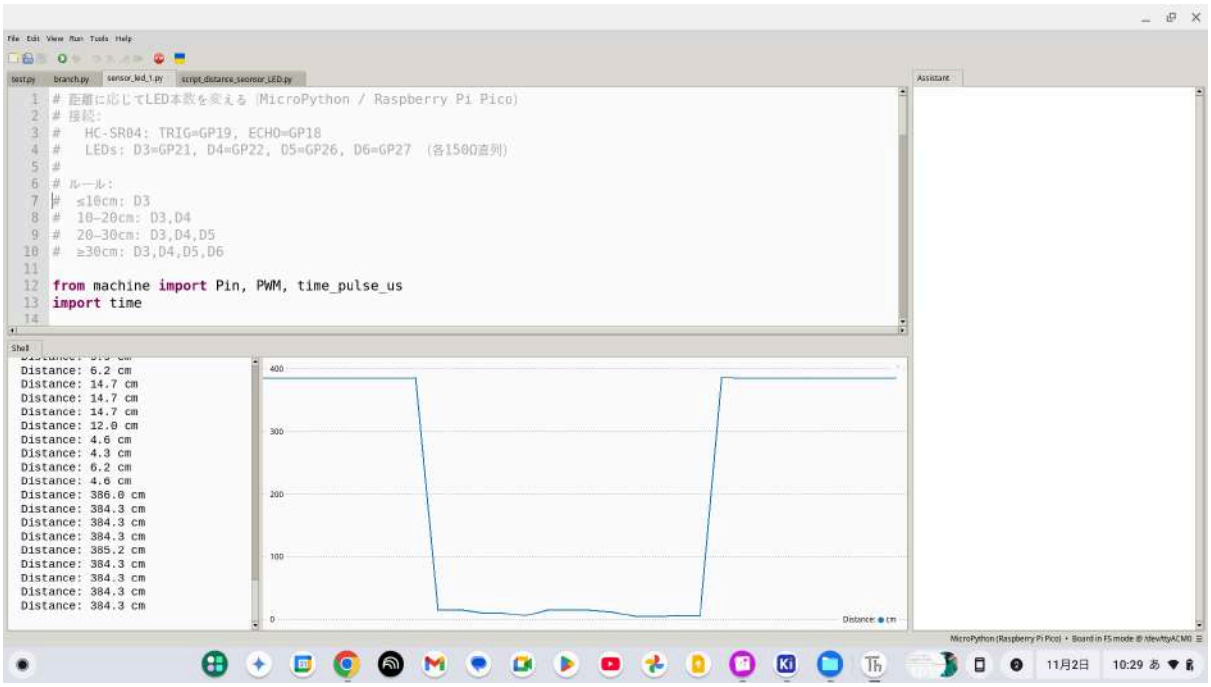
6. 取得したデータをグラフ化してみよう

～データサイエンス基礎入門～

測距センサーからの距離データの取得方法はもうマスターできましたね。次に、センサーで取得した距離データを、PC上でグラフに「可視化」してみましょう。グラフ化はデータサイエンスにおいて非常に重要で、単なる数字ではわかりにくい現象を「見える化」することで、変化の傾向や周期性を理解し、現象そのものをより深く理解することが出来ます。
Thonnyには「出力した数字をリアルタイムにグラフ表示」してくれるPlotterという機能があります。Picoなどのマイコンでセンサー値を取得したときに、その変化を折れ線グラフとして表示してくれます。早速、Plotterを使ってみましょう。
下記の写真のように、ツールバーからView→Plotterをクリックしてください。

PlotterをONにすると、Shellの右隣にPlotterのウインドウが表示されます。表示を大きくしたい場合は、ウインドウの外枠あたりにカーソルを持っていくと、矢印が表示されるので、拡大したい方法にドラッグしてください。

PlotterをONにした状態で、さきほどのsensor_led_1.pyを実行すると、下記のような折れ線グラフが表示されます。



sensor_led_1.py 実行結果(Plotter=ON)

自動でグラフウィンドウが開き、横軸=時間(秒)、縦軸=距離(cm)の折れ線グラフが描画されましたね。センサーが取得した距離に応じて、グラフのラインが上下に変化します。これにより「目には見えない距離の変化」を”可視化”することに成功しました。このように、直感的に現象を捉えることは、ものづくりを進めるうえでも、役立つスキルです。
より深く仕組みを理解するためにも下記の練習問題に挑戦してみてください

練習問題1
(スクリプト内の)下の行の.1fを.2fや.0fに変更してみてください。実行結果の表示のどこかが変わっているはずです。
print("Distance: %.1f cm" % dist)

次の章では、マイコン制御において”耳”の役割を果たしている、*フィルタリング回路を動かしながら学んでいきましょう。

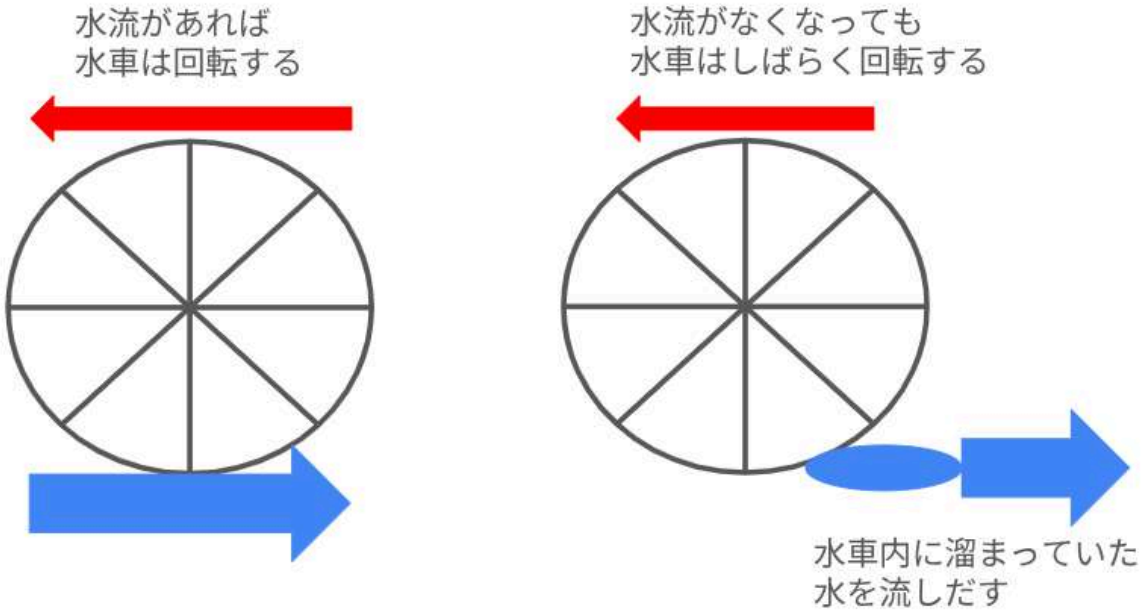
7. フィルタリング回路を動かしてみよう

～フィルタリングの不思議～

電子回路には特定の周波数のみを信号として取り出し、その他の不必要な信号を取り除くために、”フィルタリング”という仕組みを使います。この仕組みによって、電子回路は全体の信号から「見たい信号」だけを取り出して扱うことができる訳です。フィルタリングは電子部品の組み合わせで実現することができるのですが、MonoBoard上には必要な部品が既に実装してあります。これらを組み合わせてフィルタリングがどうやって実現されているのかを理解していきましょう!
まずは、フィルタリングに必要になる各部品(抵抗、インダクタ、コンデンサ)についての説明です。
※抵抗に関しては第一章、Lチカで紹介したので、省略しますね。

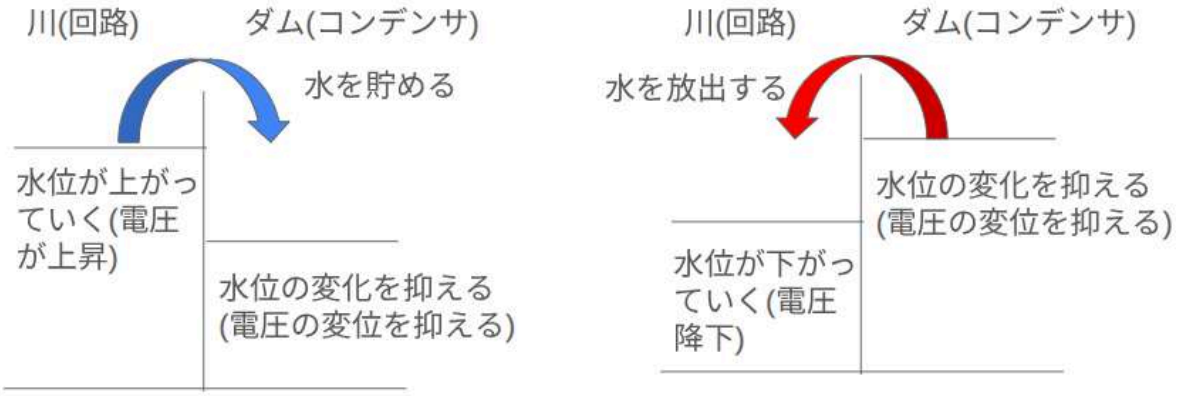
・フィルタリングに必要な部品

・インダクタ(Inductor)
インダクタは”導線の塊”のようなものだと思ってください。ただし、この導線の塊には特徴があり、「電流の変化に対して逆向きの電力を発生させる」という性質をもっています。つまり、インダクタは「変化」を嫌う性質ともいえるので、電流が減ると増やそうとしますし、増えると減らそうとする向きに働きます。
※インダクタは「流れにくい水車」をイメージしてください。電流が流れ始めるとすぐには動きませんが(逆向きの電圧が発生)、一度動いてしまうと、急には止めることができません。(止めようとする方向とは反対方向に電圧が発生する)



インダクタのイメージ図

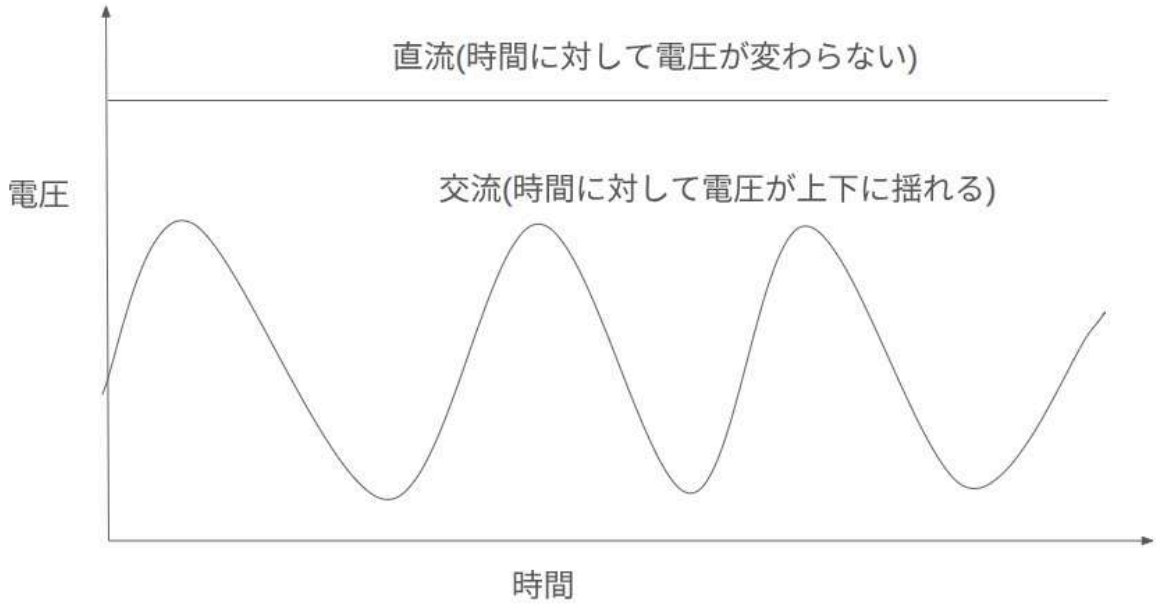
・コンデンサ(Capacitor)
次にコンデンサの紹介です。コンデンサは2枚の金属板と絶縁体(電気を通さない物質)を使って作られています。コンデンサも「変化」を嫌う性質を持っているのですが、インダクタが「電流の変化」を嫌う性質を持っているのに対して、コンデンサの場合は「電圧の変化」を嫌う性質を持っています。
※コンデンサは「水を貯めることができるダム」をイメージしてください。ダムは水位(電圧)が急に上がろうとすると、水を吸い込んで水位の変化を抑えますね。ただ逆に水位が下がると、貯めていた水を放出して安定させようとします。



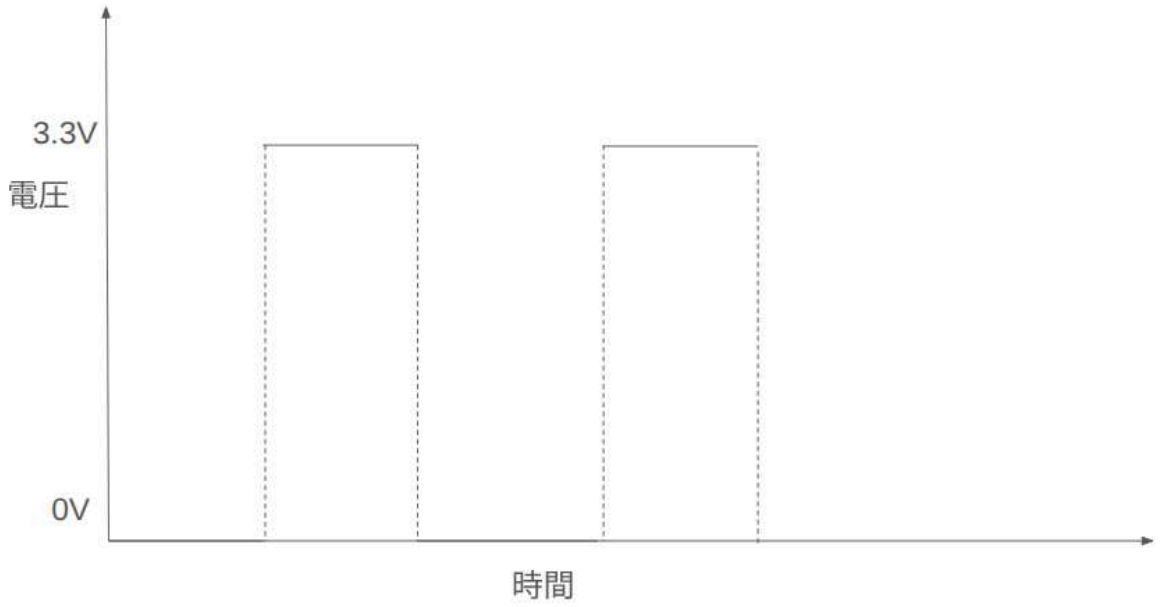
コンデンサのイメージ図

抵抗も含めて、この3つの部品を英語にすると、抵抗=Resistor, インダクタ=Inductor, コンデンサ=Capacitorとなります。この3つを組み合わせて作るのが「RLC回路」(フィルタリング回路)の正体になります。
※RLCはそれぞれの部品の頭文字！

・電気の波
もう一つ、「電気の波」に関してもここで説明させてください。既にGPIO端子から「周波数の異なる波」を送れることを学習しましたね。直流電圧(3.3V一定の電圧)と交流電圧(周波数の異なる電圧)の特徴は”感覚的に”わかっていると思いますが、ここでは、その”原理”について少し説明させてください。電気の波は大きく分けて直流と交流の二種類があります(下図)。直流は「時間に対して電圧が変わらない波」で、交流は「時間に対して電圧が変わる波」です。

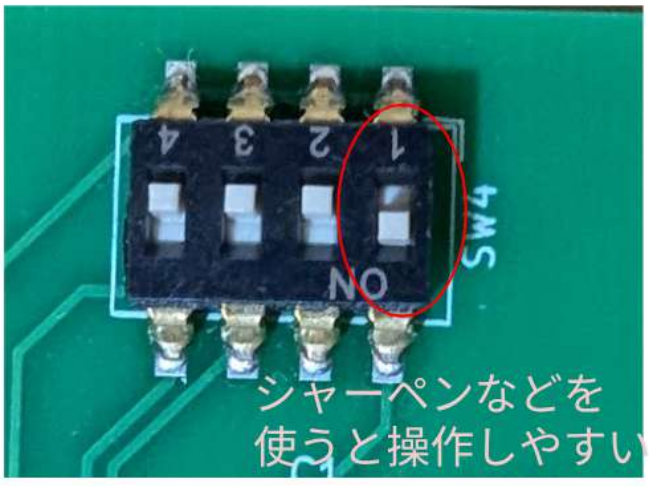


GPIO端子は「*デジタル信号を使った交流」を出力しています。(下に書いた波形のような形です)。デジタル信号は常に0Vか3.3Vのどちらかの値を周期的に取っています。平均電圧は3.3V(ON)と0V(OFF)の時間の比で変化します。ONの時間が長ければ、平均電圧(周期ごと)が高くなり、OFFの時間が長ければ、平均電圧は低くなります。
※この時間の比のことをデューティ比と呼びます。第三章のLEDのスクリプトでも使ったDUTY(LEDの明るさを調整する変数)のことですね。



(DUTY比に加えて、)周波数が遅いと、0Vと3.3Vがくっきりと別れたような波形(ONとOFFが明確に判別できる)になるのですが、周波数を速くすることで、ONとOFFの境界が曖昧になり、”擬似的な”交流を作り出すことができます。
※第三章のLEDのスクリプトでも使ったFREQ(周波数)のことですね。

・RLC回路の実践
では早速、電子回路を動かしながら、それぞれの部品が実際にどう動くかを学んでいきましょう。
まずはR(抵抗)とC(コンデンサ)で組まれた回路を動作させたいので、SW4の1をONしてください。



添付のrlc1.pyを実行すると、下の図に示すように、GP28からR1(150Ω)とC1(0.1μF)に3.3Vの直列電圧(3.3V)が流れます。コンデンサを直列接続している(直列電圧は通さない)ので、実行しても変化がないのですが、スクリプト内では3.3Vと0Vを10秒おきに切り替える制御をしています。
Pico-スイッチ1-電気回路の接続は下の図のようになっています。

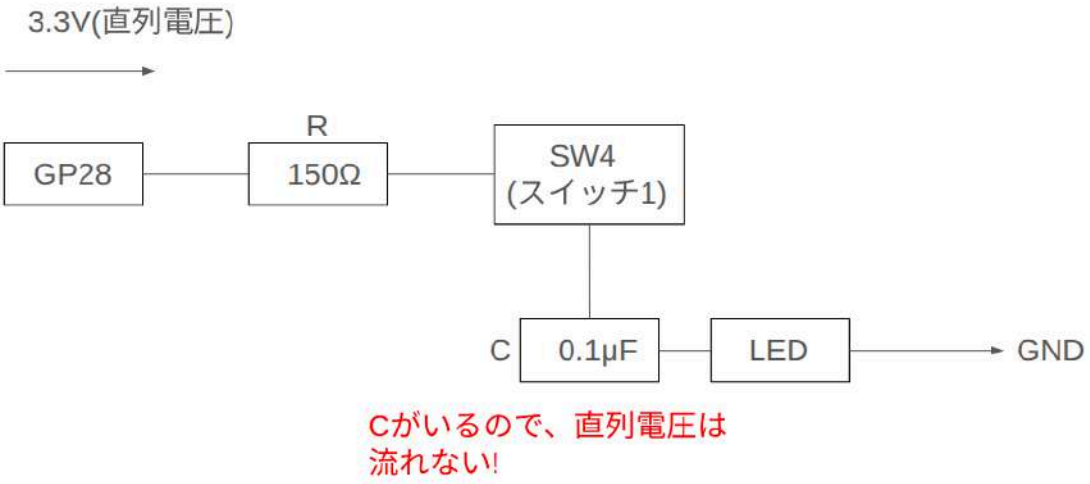
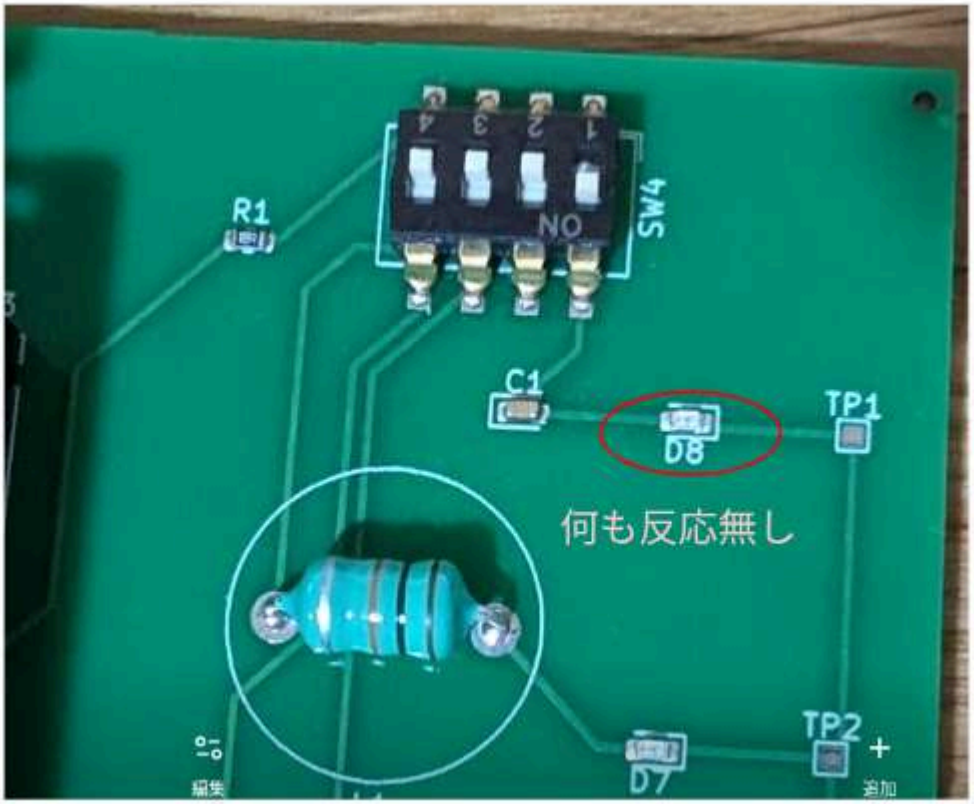


図 Pico-SW-電気回路の関係

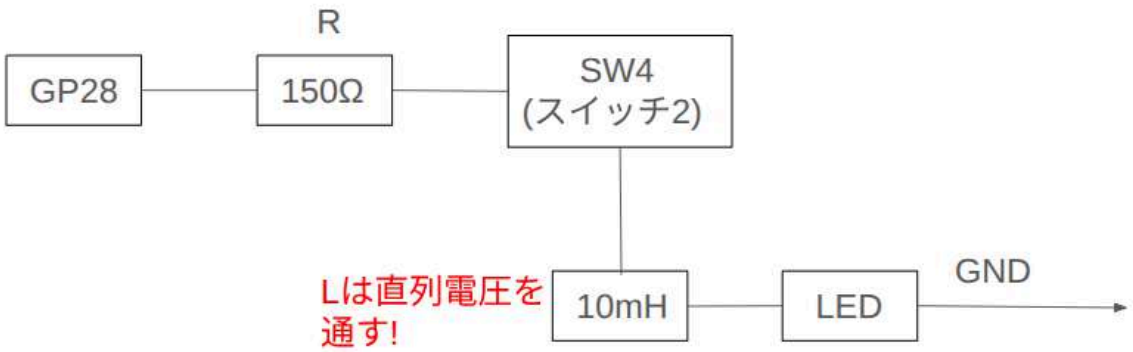


rlc1.py 実行結果

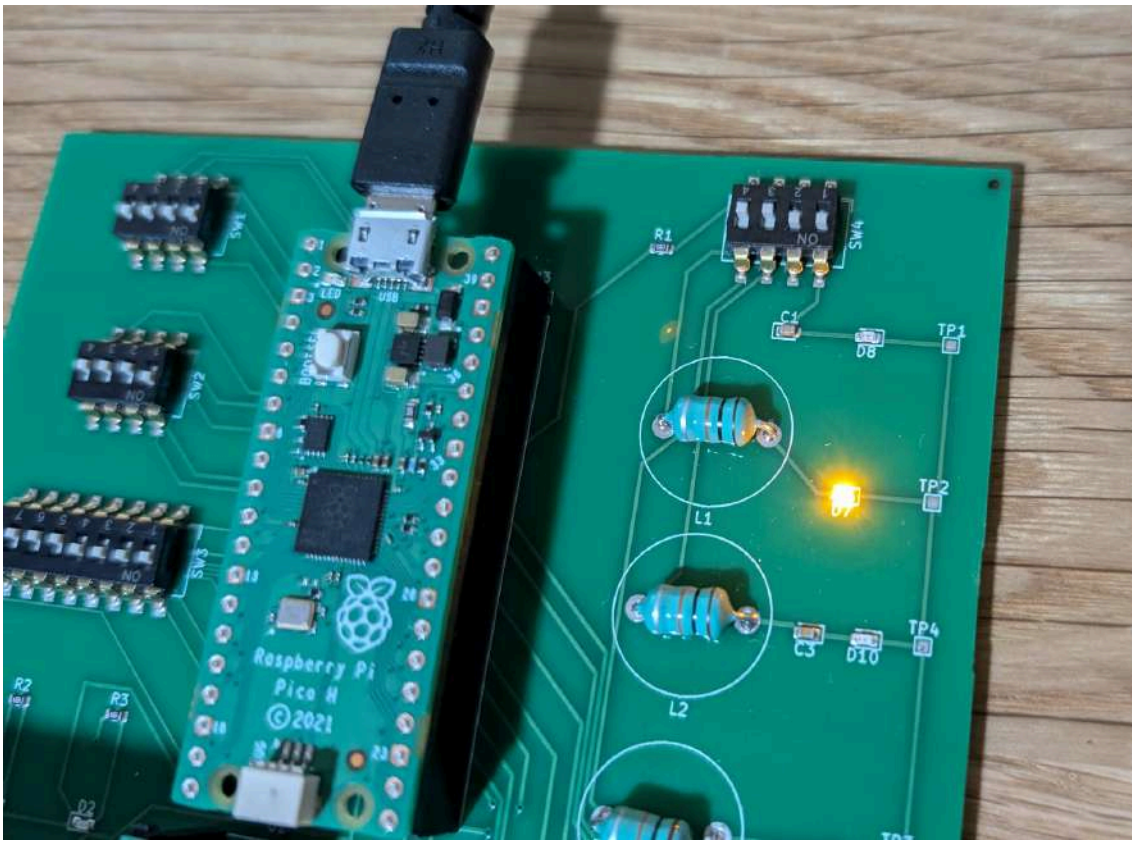
LED(D8)は何も光っていませんね。コンデンサは説明した通り、「電圧変化」に対して敏感なので、電圧をかけ始めたタイミングで僅かに電流が流れるものの、「常に3.3Vが流れ続ける」直流電圧に対しては何も反応はしないんです。(いったんコンデンサに蓄えられている電荷が一杯になってしまうと、コンデンサの後に接続されているLEDに対して直流電圧を通さないのです。)

では、コンデンサの代わりにインダクタが直列に接続されている場合はどうでしょうか？インダクタも同様に変化を嫌う性質があると書きましたが、直列電圧の場合、変化が一瞬なので(0→3.3V)、長い目で見ると、直流電圧を通しそうではないでしょうか。

rlc1.pyを実行した状態で、左隣のスイッチ2をONにしてみてください。スイッチ2の回路は下に書いているとおりです。

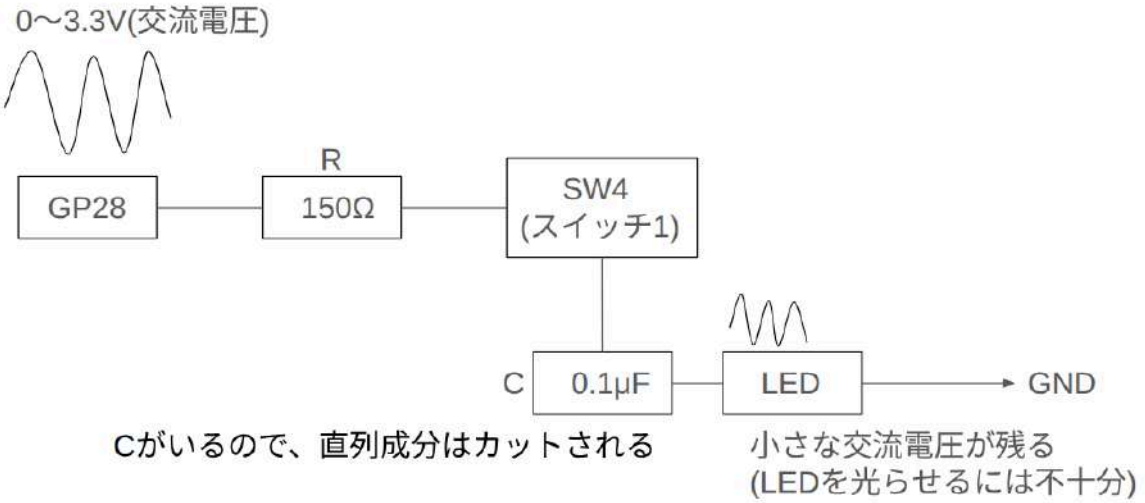


Lは「電流の変化を嫌う」性質を持っているので、瞬間的には電気を流さないものの、3.3Vが安定されて供給されると、(銅線で出来ているということもあり、)そのままの電圧を通します。よって、D7のLEDが光ってみえます。



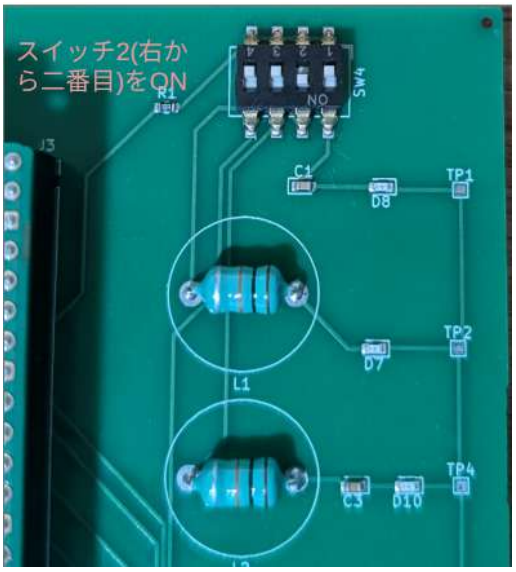
スイッチ2をONにしたとき

では、交流だとどうでしょうか？
「電圧が0～3.3Vの間で変化する」交流電圧を発生させるrlc2.pyを実行してみてください。
スイッチ1(コンデンサが直結されている回路)とスイッチ2(インダクタが直結されている回路)の結果を見比べてみてください。スイッチ1は下の図で交流電圧になったものの、交流成分に含まれている直列成分がコンデンサでカットされるので、微小な電流はLEDに流れるものの、LEDを光らせるには十分な電流量ではありません。



スイッチ2の回路はどうでしょうか。LEDが明るくなったり、暗くなったり、周期的に変化していますね! これはどうしてでしょうか？
交流電圧ということは「電圧が変化している波」なので、同じ抵抗値(150Ω)に対して、電流量が時間によって、変化します。その結果、インダクタの「電流に対して変化を嫌う性質」がフルに働き、電圧が高いほど、インダクタは電流を流さない方向に働くので、暗く見え、電圧が低いほど、インダクタは電流を流したがる方向に働くので、明るく見えます。
そのサイクルが”波”として繰り返されるので、明るさが周期的に変化するわけです。

・周波数を深堀り
直流/交流の違いや、フィルタリングに使われる部品(RLC)の特性がわかったところで、「周波数」に関してさらに深堀りしていきましょう。今まで説明した中で、度々出てきてたDUTY比(LEDの明るさを調整するパラメータ)という考え方がありましたね。このDUTY比は「一周期(一サイクル)の中でONの時間がどれくらいあるか」という内容で、ON時間の長さによって、LEDの明るさが変化するというものでした。それと対をなすパラメータで、「周波数」という定義があり、周波数は「1秒間にON/OFFサイクルが何回繰り返されるか(単位:Hz)」というものでした。GPIO端子から出力される波は3.3Vと0Vの出力を繰り返していますが、この繰り返しの速さを制御しているのがこの周波数になるんですね。なので、周波数が高ければ、周期の短い波になり、周波数が低ければ、周期の長い波になります。より具体的なイメージを持ってもらうために周波数ごとのLEDの光り方の違いを実際に見てみましょう。早速、rlc3.pyを実行してください。rlc3.pyは「50kHz～500kHzまでの周波数を3秒ごとに変化させる」スクリプトです。



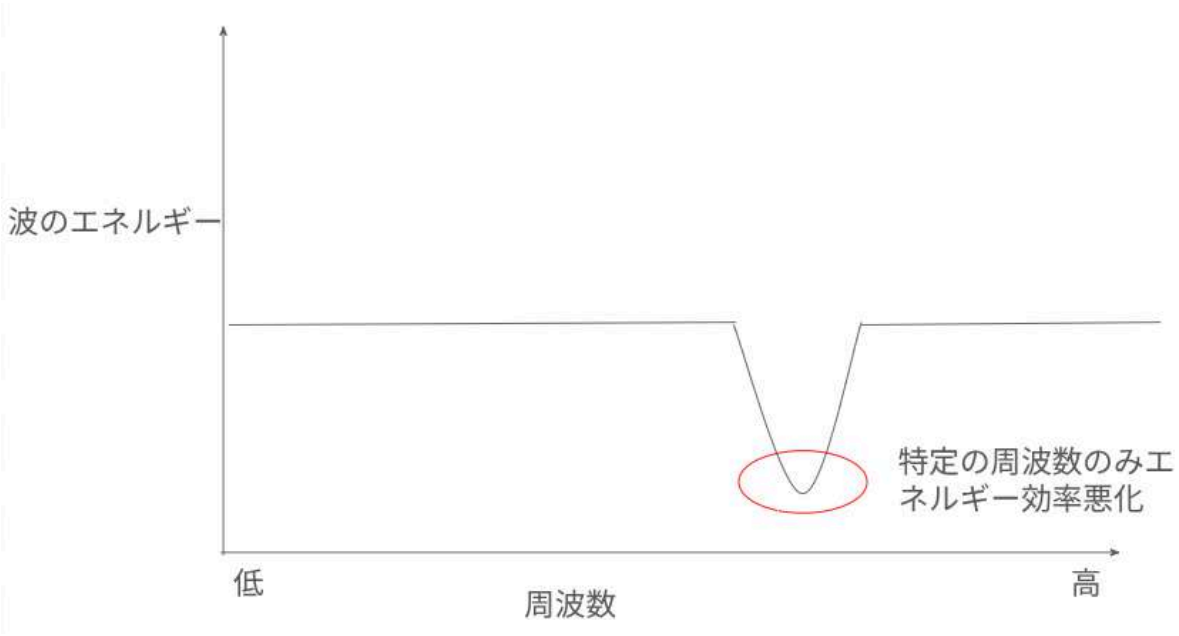
・50kHz(低周波)
交流の周期(ON/OFF)がゆっくりしていて、インダクタは電流変化に十分追従できています。その結果、電流は*矩形波に近い形で表現され、LEDには大きな平均電流が流れ、明るく光ります。

・100kHz～500kHz(高周波)
周波数を上げるごとに、徐々に波の周期が速くなっていきます。インダクタには「電流変化に抵抗する性質」があるので、周波数を上げていく度に、抵抗成分が大きくなる特徴があります。ゆえに、(周波数があがると)平均電流が減少し、LEDの光も弱く見えています。まるで、インダクタがほぼ電流を通さない”フィルタ”のように振る舞っていることに見えませんか?
フィルタリング回路の理解をさらに、深めるために、下の練習問題に取り組んでみましょう。
練習問題1.
周波数の範囲を「50kHz～500kHz」から「100Hz～10kHz」、「1MHz」に設定を変えてみよう。

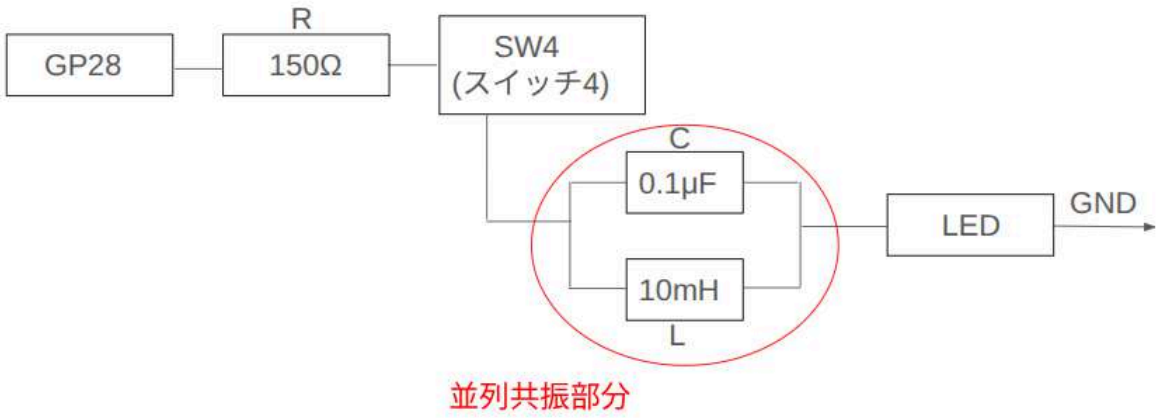
練習問題2.
周波数切替の速さ(time.sleep(3))を0.5/1/5に変更して、周波数切替の速さを変えてみよう。

周波数の振る舞いに関して、かなり詳しくなることができましたね。では、少し難しいトピックではあるのですが、周波数の応用編でもある「共振現象」についてみてみましょう。

・共振現象(応用編)
共振とは、ある特定の周波数の時だけ、エネルギー伝搬効率が変化する現象のことです。RLC回路の部品配置次第で、ある特定の周波数に対して「最も効率の良いエネルギー伝搬」と「最も効率の悪いエネルギー伝搬」の構成を作ることが出来ます。わかり易い例として、ここではLとCを並列に接続した「最も効率の悪いエネルギー伝搬」である並列共振を学んでいきましょう。LとCを並列で接続した場合は、特定の周波数で、LとCの間で電流が打ち消し合い、電気的に見るとまるで”大きな抵抗”が繋がっているように見えます。この特定の周波数のことを”並列共振周波数”と呼びます。下の図のように、”特定の周波数”でのみエネルギー効率がガクンと下がってしまうようなイメージです。



スイッチ4にはこの”共振並列回路”が実装されていて、下の図のような回路になっています。LとCの値を”特定の周波数に対して反応する”ような組み合わせにしています。



rlc4.pyはこの回路の共振周波数である「5kHz」と離れた周波数を交互に出力するスクリプトになっています。(DUTY比は一定で、周波数のみ切り替えています)。
スイッチ4(SW4内の一番左のスイッチ)をONにして、スクリプトを実行してみてください。
※5kHzでのLEDの光り方に注目してみてください。

これで「共振現象」の特徴に関して理解が深まりましたね! 周波数は電子回路を扱う上で、非常に重要な概念なので、しっかりマスターしましょう。

8. 二進数を体験してみよう

この章では「二進数」に関して勉強していきましょう。人間同士が共通の言語(日本人なら「日本語」、アメリカ人なら「英語」)を使って、コミュニケーションするように、機械にもコミュニケーションするための「二進数」という言語があります。(機械同士のコミュニケーションに使う機械語とも言えますね。)
例えば、人間が使う数字は「0」～「9」までの10種類がありますが、機械語は「0」と「1」の二種類で表現することが出来ます。この「0」と「1」の組み合わせこそが、機械が理解できる最も基本的な情報のかたまりになります。その”かたまり”は「2の累乗(2を何回かけたか)」というルールに従っています。(2の一乗=2、2の二乗=2x2=4、2の三乗=2x2x2=8というように、掛けるたびに数が二倍ずつ増えていきます)
例えば、十進数の3は二進数では「11」と表すことが出来ます。下の図のように、一番右の桁が1($1x2^0$)、その左の桁が1($1x2^1$)を意味しており、足し合わせると3(1+2)になる訳です。機械にとっては、電圧があるときは1、ないときは0というように、電圧ON/OFFの組み合わせで情報を扱えるほうが便利という訳です。

二進数の計算の方法

・二進数の四則演算

二進数の四則演算(足し算・引き算・掛け算・割り算)をそれぞれ説明していきます。二進数でも基本的な計算方法は十進数と同じです。ただし使える数字が「0」と「1」に限られているので、少しだけ考え方がシンプルになります。
※二進数の演算はシンプルなのですが、感覚を掴むのにコツが必要です。この先、各演算のルールを記載していますが、慣れないうちは難しいと思うので、手を動かしながら習得していきたい方は「二進数の計算を体験してみよう」まで読み飛ばしてもらっても全く問題ないです。

・足し算の場合
足し算は以下の表のルールだけで出来ます。

計算結果が1+1になる桁は0にして、一桁分繰り上がる点がポイントです。例題では2桁目に1+1があるので、一桁分繰り上がり、繰り上がった桁でも1+1になり、さらに繰り上がった桁でも1+1になるので、最終的に一桁分左に「1」が繰り上がっています。

式	結果	桁上り
0+0	0	なし
0+1	1	なし
1+0	1	なし
1+1	0	1(繰り上がり)

例(11+6=17の計算方法)

		1	0	1	1	11
+		0	1	1	0	6
	1	0	0	0	1	17

・引き算の場合
引き算は足し算とは逆に「上の桁から借りてくる」ルールを使います。例題では1桁目で0-1が発生しているので、一桁上の桁から1を借りてきて、「1-1」の結果は1になります。(ややこしいところです)
借りられた桁も0-1になるので、二桁上の1を借りてきます。このとき、一桁上の行も1になります。(借りが通過する桁も1に変わる)
※いちばん左まで探しても1が見つからない場合は、その桁数では計算できず、結果は負になります。

式	結果	繰り下がり
0-0	0	なし
1-0	1	なし
1-1	0	なし
0-1	1	1(繰り下がり)

例(10-3=7の計算方法)

	1	0	1	0	10
-	0	0	1	1	3
	0	1	1	1	7

・掛け算の場合
掛け算は「1のところだけ足し算する」イメージです。10進数の筆算と似ているので感覚的に掴みやすいと思います。例題では101(5)に対して(3)の一桁目の1を掛け算し、次に101(5)に対して(3)の二桁目の1を掛け算し、左に一桁ずらしています。最後に演算結果を足し合わせると掛け算は完了です。

例(5x3=15の計算方法)

		1	0	1	5
x			1	1	3
		1	0	1	101(5)掛ける3の一番右の桁
	1	0	1	0	101(5)掛ける3の右から二番目の桁(左に一桁シフトする)
	1	1	1	1	15

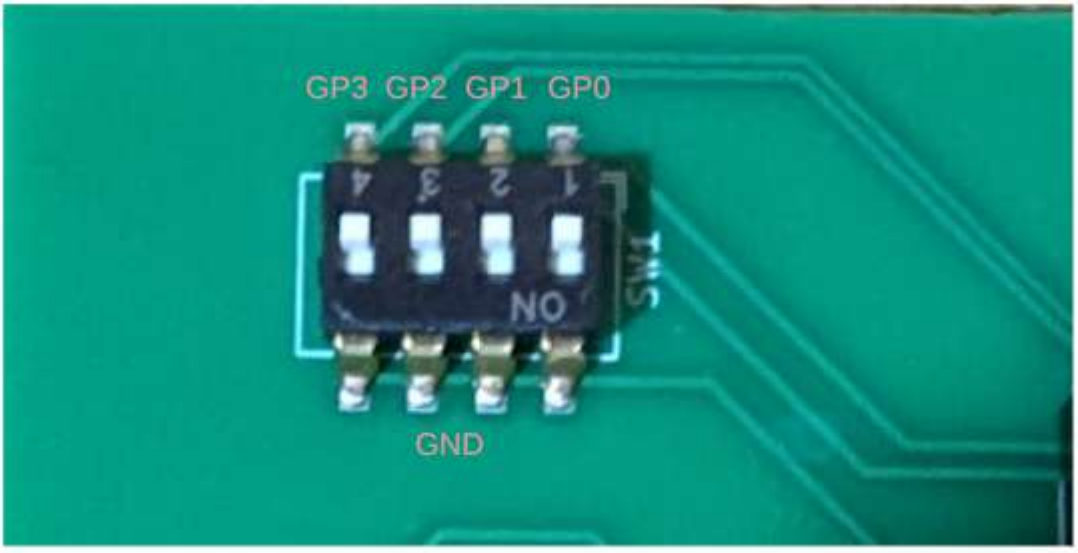
・割り算の場合
割り算も十進数の筆算と同じで、何回引けるのかを考えながら計算します。

	例(5x3=15の計算方法)				
		1	0	1	5
x			1	1	3
		1	0	1	101(5)掛ける3の一番右の桁
	1	0	1	0	101(5)掛ける3の右から二番目の桁(左に一桁シフトする)
	1	1	1	1	15

・二進数計算を体験してみよう

二進数は慣れていないと感覚が掴みづらいので、演算のルールが把握できていなくても大丈夫です。「実際に手を動かしながら」二進数を勉強していくと、徐々に感覚が掴めてくると思います。

SW1(左上)のスイッチとPicoのGPIO端子は「1対1の関係」で接続されており、それぞれのスイッチの端子に対して、GPIO端子が一つ割り当てられています。



GPIOは*内部抵抗を持っており、内部抵抗でスイッチの状態(GNDに接続されていないか/接続されているのか)を読み取ることが出来ます。ONと書かれた下側にはGNDと接続する配線がされており、スイッチをON側に下げるとGPIO端子はH→Lに移行したと読み取ります。(この読み取りの為に、内部抵抗は使用されます）
※スイッチのON/OFF操作はシャープペンシルやピンセットなど、先の尖ったものが便利です。

試しにスクリプトのbinary_1を実行してみてください。binary_1は「SWの論理状態を各GPIO端子で読み取る」スクリプトなので、すべてのスイッチがOFF状態だと、下のようにすべての端子がHを読み取ることになります。これはGPIO端子の内部抵抗を通して、3.3V(H)に接続されているためです。

4(GP3):H 3(GP2):H 2(GP1):H 1(GP0):H

次に、SW1の1と書かれたスイッチをON側に倒してみて下さい。そうすると、下のように変化しますよね。

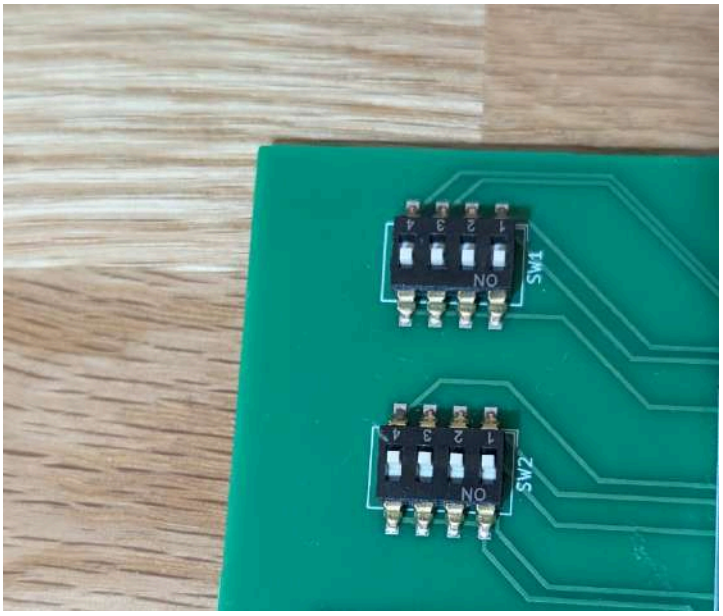
4(GP3):H 3(GP2):H 2(GP1):H 1(GP0):L

これはGP1がスイッチ1の状態を読み取ることで、GNDに接続=Lと判断しています。同様に、スイッチ2～4までをON側に倒しましょう。最終的には下記の結果が得られますね。

4(GP3):L 3(GP2):L 2(GP1):L 1(GP0):L

仕組みは理解できたでしょうか？このH/Lの仕組みを、さきほどの二進数計算(0/1)に当てはめれそうですよね。
ここからはスイッチを使った、二進数計算をしていきましょう。

・SWを使った二進数計算練習
SWとGPIO端子の対応関係がわかったところで、ここからは「複数のSWを使った二進数計算」を進めていきたいと思います。SW1とSW2の4スイッチを4桁同士の演算に見立てて、実際に手を動かしながら計算結果を確認することで、二進数の感覚を掴んでもらいたいと思います。では早速、添付のbinary2.pyを実行してください。



上下に実装されているSW1とSW2

binary2.pyを実行すると、下記の画面が出てくると思います。四則演算がどれでも選択できるのですが、例として、「番号を入力」という項目に1(加算)を入力してみましょう。

1: 加算 (+)
2: 減算 (-)
3: 乗算 (*)

4: 除算 (/)
番号を入力→ 1

そうすると、「SW1とSW2の加算結果」が表示されます。この例ではSW1は全てON側に倒しているので、0000となっており、SW2は全てOFF側なので1111となっています。()の中には10進数に変換した値が表示されています。一つの演算が終わると、10秒後に新しい演算入力を求めるスクリプトになっているので、各スイッチを操作しながら演算の組み合わせを試してみてください。

SW1 : 0000 (0)
SW2 : 1111 (15)
演算 : 0 + 15 = 1111 (10進:15)

練習問題として、下記のSWの状態にして、演算結果がどうなるか、埋めてみてください。

1:加算
SW1 : 0010
SW2 : 1001

2:減算
SW1 : 0100
SW2 : 0011

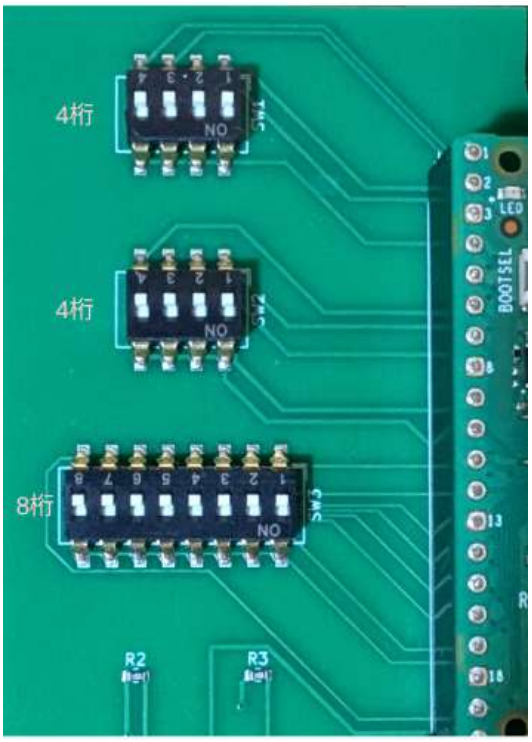
3:乗算
SW1 : 1000
SW2 : 0111

4:除算
SW1 : 0101
SW2 : 0011

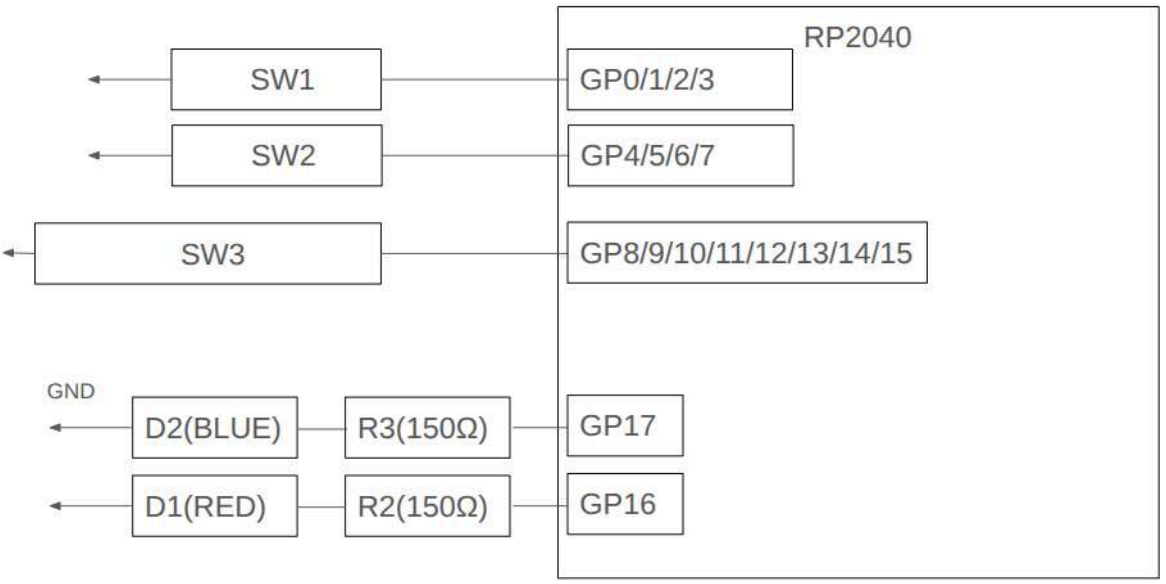
二進数の四則演算の感覚が掴めましたかね？次の章では応用編に取り組んでいきましょう。第三章で学習した「LEDの光らせ方」を参考にして、「計算した結果に応じてLEDの光り方」を変化させるスクリプトに挑戦してみましょう。

9. 計算結果に応じてLEDを光らせてみよう

SWを動かしているうちに気づかれた方もいるかもしれませんが、4桁同士の二進数の計算結果は最大で8桁の二進数で表現することが出来るんです。4桁の二進数は0000(0)～1111(15)までの値を取るので、SW1(1111)+SW2(1111)の乗算結果は11100001(225)となります。すなわち、4桁同士の計算の場合は最大で8桁の計算結果になるということです。そこで、基板上にSW3(8桁)を配置して、2の8乗まで表現できるようにしました。



SW3もまた、それぞれのスイッチが各GPIO端子に接続されている配線になっています。
ここでは、SW3が8桁の二進数を格納できることを利用して、「SW1とSW2の計算結果とSW3の値を比較する」演習をやってみましょう。ただ、比較するだけでは面白くないので、値が一致した場合は青LED(正解)を点灯、値が不一致の場合は赤LED(不正解)という形式にしましょうか。値の比較からLED点灯までの回路図と処理フロー(流れ)は以下の通りです。
・回路図



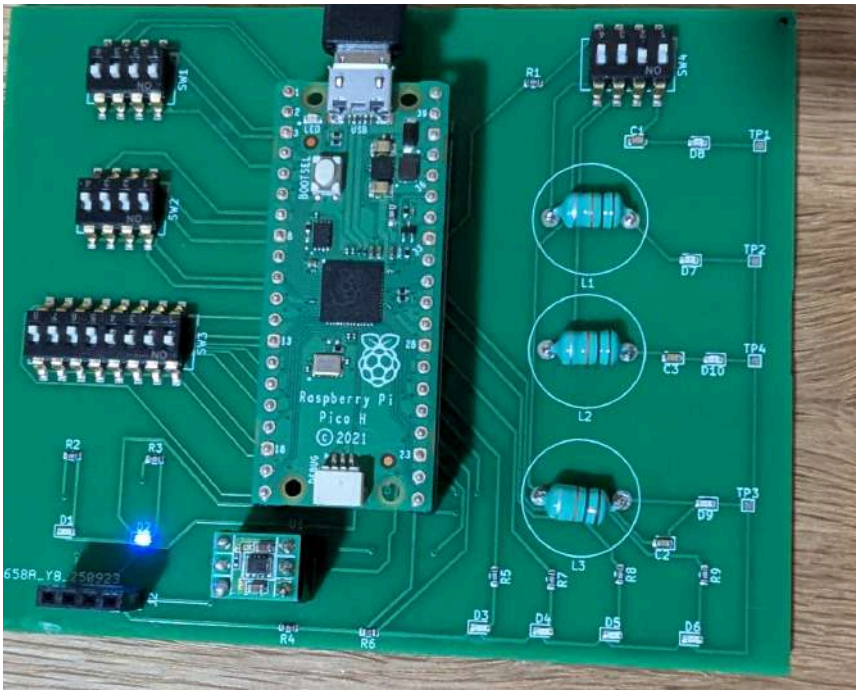
SW1はGPIO0～3の端子に、SW2はGPIO4～7の端子に接続されています。(GPIOがH/Lを読み取ることで、スイッチが0/1どちらの状態にいるのかがわかります。)

同様にしてSW3の各スイッチもGP8～15のそれぞれに紐づけられているので、SW1、SW2に入力された値をSW3の値をPico内で比較できるわけです。比較した結果に応じて、Picoではさらに GP17(青色LEDが接続されている端子)がGP16(赤色LEDが接続されている端子)のどちらかを光らせるのかを選択する。という流れになっています。一見複雑そうに見える処理でも、順番に噛み砕いていくと、単純な処理の連続であることがわかりますね。

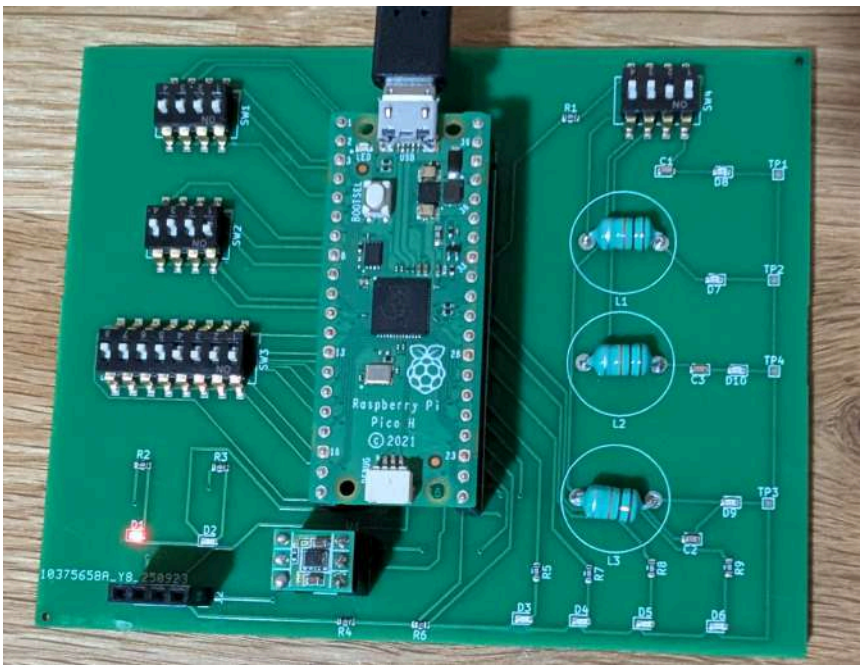
・処理フロー

①SW1とSW2の演算(加算/除算/乗算/除算)をPico内で行う
↓
②①の演算結果とSW3の値をPicoが比較
↓
③正解であれば青LEDを点灯/不正解の場合は赤LEDを点灯

上記の回路図とフローをベースに作成したのがbinary3.pyです。実行すると、正解であれば赤色LED(GP16)が、不正解であれば青色LED(GP17)がリアルタイムで表示されますね。リアルタイムで追従するスクリプトなので、実行中にスイッチを切り替えてみてください。



正解の場合：青色LED(GP17)が光る



不正解の場合：赤色LED(GP16)が光る

二進数の計算の感覚がかなり掴めてきたのではないでしょうか。二進数の感覚がわかってくると、コンピュータに命令を送るときに、どういった命令文を送れば良いのかが感覚的に掴みやすくなります。筆者も10進数で取得したデータを二進数に変換するときに、慣れるまで時間がかかりました。今のうちから二進数の感覚を掴んでおくことは、将来、製造業やIT業界で働く際に必ず役に立つ日がくるはずです。

・おわりに

本マニュアルを最後まで読んでいただき、本当に有難うございます。途中、難しいところもあったと思いますが、実際に手を動かしながら学ぶことの面白さを感じて頂けていれば幸いです。一見すると、電子工作の世界は難しそうに見えるかもしれませんが、その本質は「自分のアイデアーつでどんなものでも作れる」自由さにあります。LEDを光らせることも、センサーで測定することも、自作の回路を組むことも、全てはあなたの発想次第です。本マニュアルでは、そんな体験を一人でも多くの方にかんじてもらえるよう、できる限り実践的な構成を心がけました。

電子工作は「失敗」から学べる分野です。それらは決して「間違い」ではなく、改善のヒントになります。「なぜそうなるのか」を考えながら色んな失敗をして、理解を深めて言っていただければと思います。

また、この教材で紹介した内容は、ほんの入口にすぎません。Raspberry Pi PicoをMonoBoardから取り外して、通信モジュールや他のセンサー、モーターと組み合わせることで無限の応用が可能になります。ぜひ、「自分で作った仕組み」で色んな工作をしてみてください。