Raspberry Pi Pico の使い方（基本編）

（Pico2W 若干対応 2026 年 7 月版）

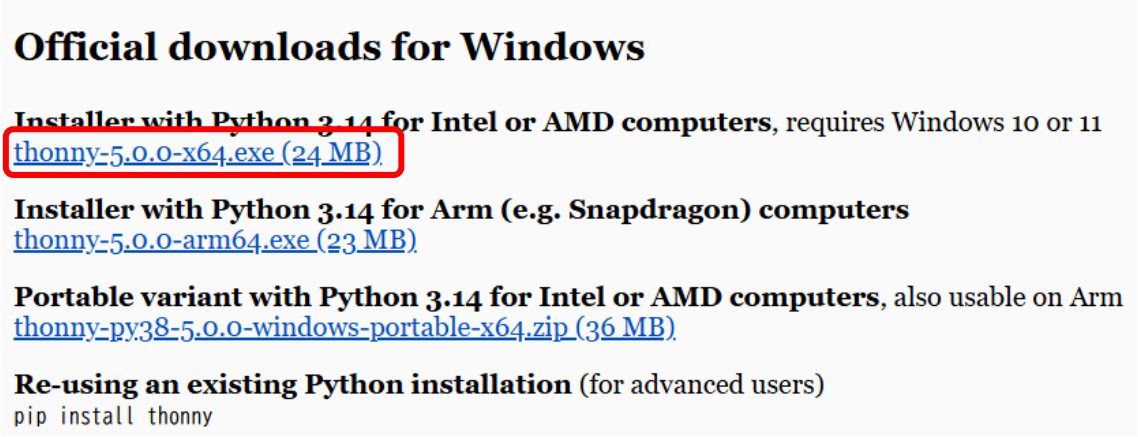
①Raspberry Pi Pico 使ってみよう（準備編）

必要なもの

・Raspberry Pi Pico2W（以下、Pico）※PICOW でも可（動作しない個所があります）

・USB ケーブル　Micro-B（データ通信対応）

・ThonnyIDE　https://thonny.org/　←からダウンロード　　下の赤枠をダウンロード



Official downloads for Windows

Installer with Python 3.14 for Intel or AMD computers, requires Windows 10 or 11
thonny-5.0.0-x64.exe (24 MB)

Installer with Python 3.14 for Arm (e.g. Snapdragon) computers
thonny-5.0.0-arm64.exe (23 MB)

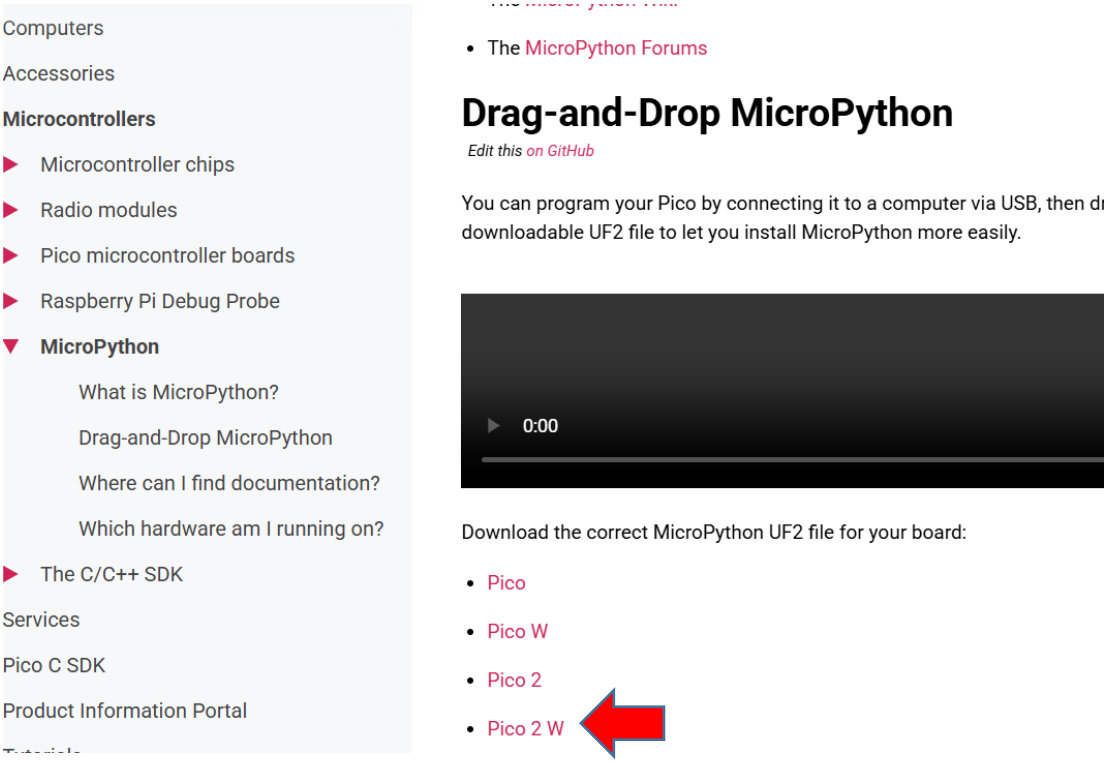
Portable variant with Python 3.14 for Intel or AMD computers, also usable on Arm
thonny-py38-5.0.0-windows-portable-x64.zip (36 MB)

Re-using an existing Python installation (for advanced users)
`pip install thonny`

その都度、プログラム動作必要なものがありますので、それに合わせて購入してください。

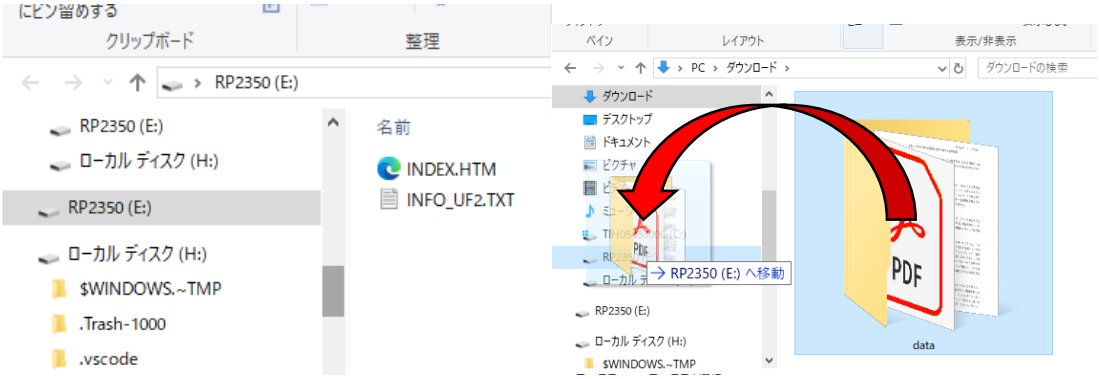
(1) 事前に micropython を使用するための uf2 ファイルをダウンロードしておきます。
※もし PICOW を使う人は「PicoW」を選択。

https://www.raspberrypi.com/documentation/microcontrollers/micropython.html



(2) Pico の『BOOTSEL』を押しながら USB ケーブルとパソコンと Pico をつなぎます。

(3) リムーバブルディスクとして表示されるのでダウンロードした uf2 ファイルをドラッグして中に入れます。

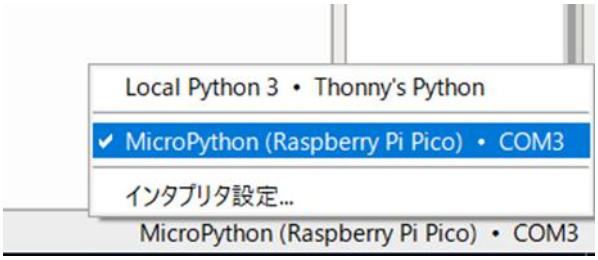


(4) 一度、リムーバブルディスクは消えますが、これで準備完了です。

②Thonny IDE の使い方

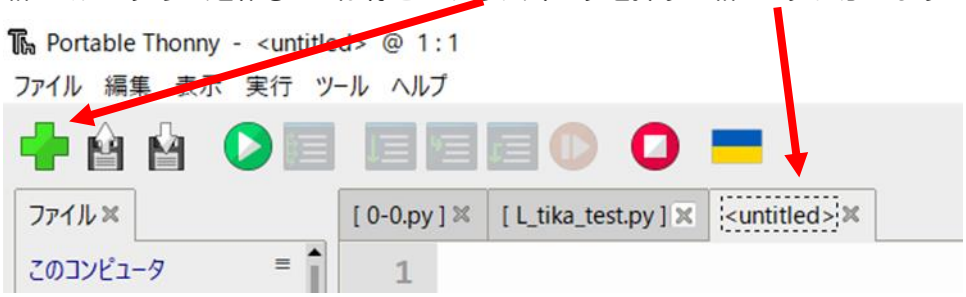
(1) パソコンに USB ケーブルを使って Pico を接続します。

(2) Thonny IDE を立ち上げ右下のデバイスを「MicroPython」に変更します。パソコンによって COM○の番号が違いますが同じようなものを選んでください。



③プログラムを作成しよう

新しくプログラムを作るときは緑色のプラスマークを押すと新しいタブが出ます。



④練習問題1_内蔵 LED を点滅させよう。

最初から埋め込まれている LED を点滅させてみます。

```
from machine import Pin
import utime

t=0.5
L=25

for k in range(10):
    LNO = Pin("LED", Pin.OUT)
    #PicoW の場合、"LED"をL変更
    LNO.value(1)
    utime.sleep(t)
    LNO.value(0)
    utime.sleep(t)
```

プログラム入力後は保存して  を押してみましょう。

点滅したら、次のとおりに仕様を変更しましょう。

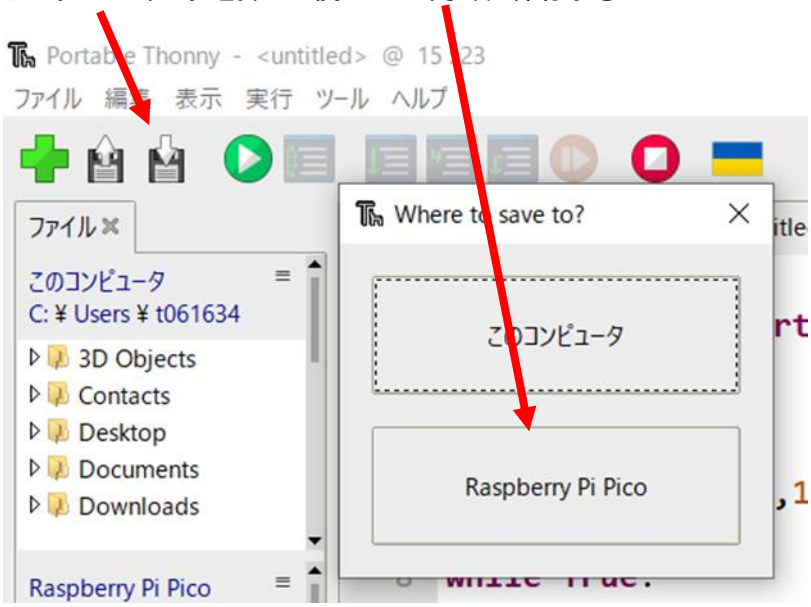
A点滅の速さを遅くしたり早くしたりしましょう。

B点滅する回数を 15 回に変更しましょう。

C点滅後は LED が点灯のままにしましょう。

⑤保存しよう

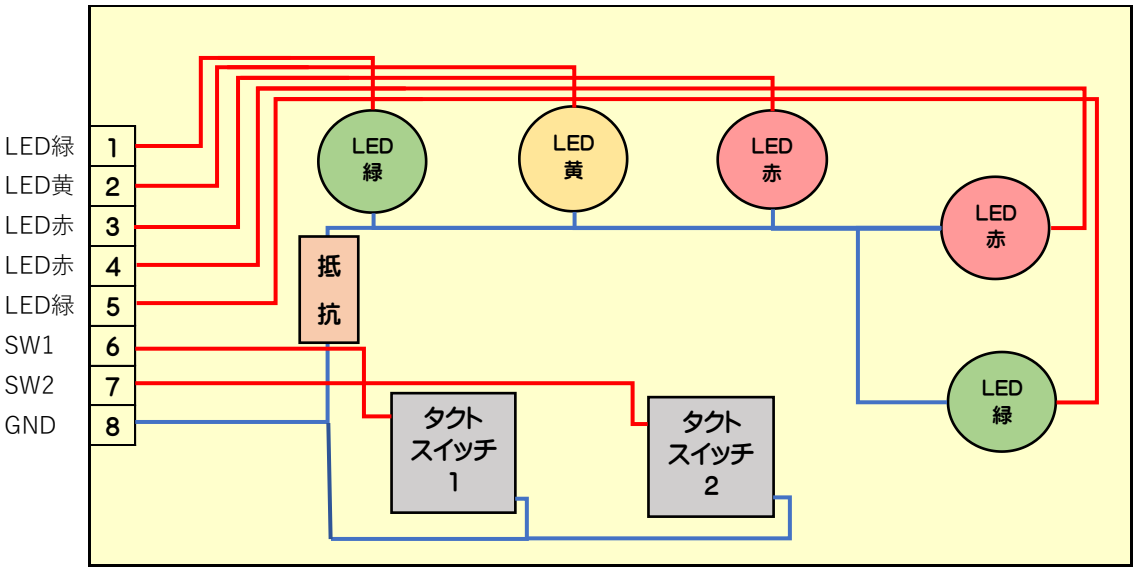
フロッピーマークを押した後、Pico 内部に保存する



保存の際は最後の「.py」をつけるのを忘れずに！

⑥練習問題2_LED を制御させてみよう

機材があれば以下の内容でユニバーサル基板にはんだ付けして LED を制御する基盤を作ってみましょう。



ラズパイ pico2W はプルダウン接続に大きなバグが発生するので、プルアップ接続で使用する。

PICO2W GPIOの配列

UART0_TX	I2C0_SDA	SPI0_RX	GP0	1		40	VBUS			
UART0_RX	I2C0_SCL	SPI0_CSn	GP1	2		39	VSYS			
			GND	3		38	GND			
	I2C1_SDA	SPI0_SCK	GP2	4		37	3V3_EN			
	I2C1_SCL	SPI0_TX	GP3	5		36	3V3(OUT)			
UART1_TX	I2C0_SDA	SPI0_RX	GP4	6		35		ADC_VREF		
UART1_RX	I2C0_SCL	SPI0_SCn	GP5	7		34	GP28	ADC2		
			GND	8		33	GND	AGND		
	I2C1_SDA	SPI0_SCK	GP6	9		32	GP27	ADC1	I2C1_SCL	
	I2C1_SCL	SPI0_TX	GP7	10		31	GP26	ADC0	I2C1_SDA	
UART1_TX	I2C0_SDA	SPI1_RX	GP8	11		30	RUN			
UART1_RX	I2C0_SCL	SPI1_CSn	GP9	12		29	GP22			
			GND	13		28	GND			
	I2C1_SDA	SPI1_SCK	GP10	14		27	GP21			
	I2C1_SCL	SPI1_TX	GP11	15		26	GP20			
UART0_TX	I2C0_SDA	SPI1_RX	GP12	16		25	GP19	SPI0_TX	I2C1_SCL	
UART0_RX	I2C0_SCL	SPI1_CSn	GP13	17		24	GP18	SPI0_SCK	I2C1_SDA	
			GND	18		23	GND			
	I2C1_SDA	SPI1_SCK	GP14	19		22	GP17	SPI0_CSn	I2C0_SCL	UART0_RX
	I2C1_SCL	SPI1_TX	GP15	20		21	GP16	SPI0_RX	I2C0_SDA	UART0_TX

自作したボードを1～9につなぎましょう。LED：GP0～GP4　sw：GP5～GP6
先ほどのプログラムを変更して LED が点滅するかチェックしてみましょう。
どの番号がどの LED につながっていたか理解できましたか？

⑦LED を連続で点滅させてみよう

LED が点灯することが理解出来たら、自分で考えて LED を 1 つずつ点灯させるプログラムを作ってみよう。

⑧リストを覚えよう

リストを覚えるとプログラムが長くならない他に、条件が違うと動作が変化するプログラムにも対応できるので覚えよう。

```
2  from machine import Pin
3  import utime
4
5  t=0.5
6  L=[0,1,2,3,4]
7  ※この形が【リスト】
8
9  for k in range(5):
10     LNO = Pin(L[k], Pin.OUT)
```

```
from machine import Pin
import utime

t=0.5
L=[0,1,2,3,4]


for k in range(5):
    LNO = Pin(L[k], Pin.OUT)

    LNO.value(1)
    utime.sleep(t)
    LNO.value(0)
    utime.sleep(t)
```

⑨スイッチで制御しよう

```
sw1=Pin(sw[0],Pin.IN,Pin.PULL_UP)
sw2=Pin(sw[1],Pin.IN,Pin.PULL_UP)
                          ※スイッチの設定
while True:

    if sw1.value()==0: ※スイッチ1が押されたとき
        LNO = Pin(L[0], Pin.OUT)
        LNO.value(1)
        print("sw1")
```

```
from machine import Pin
import utime

L=[0,1,2,3,4]
sw=[5,6]
sw1=Pin(sw[0],Pin.IN,Pin.PULL_UP)
sw2=Pin(sw[1],Pin.IN,Pin.PULL_UP)

while True:

    if sw1.value()==0:
        LNO = Pin(L[0], Pin.OUT)
        LNO.value(1)
        print("sw1")
    elif sw2.value()==0:
        LNO = Pin(L[1], Pin.OUT)
        LNO.value(1)
        print("sw2")
    else:
        for x in range(5):
            LNO = Pin(L[x], Pin.OUT)
            LNO.value(0)
            print("OFF")

    utime.sleep_ms(20)
```

このプログラムができたら次の仕様に変更しよう

A　sw1 を押したら右から 1 つずつ LED が点灯する

B　sw2を押したら左から 1 つずつ LED が点灯する

※間隔は任意で構いません。

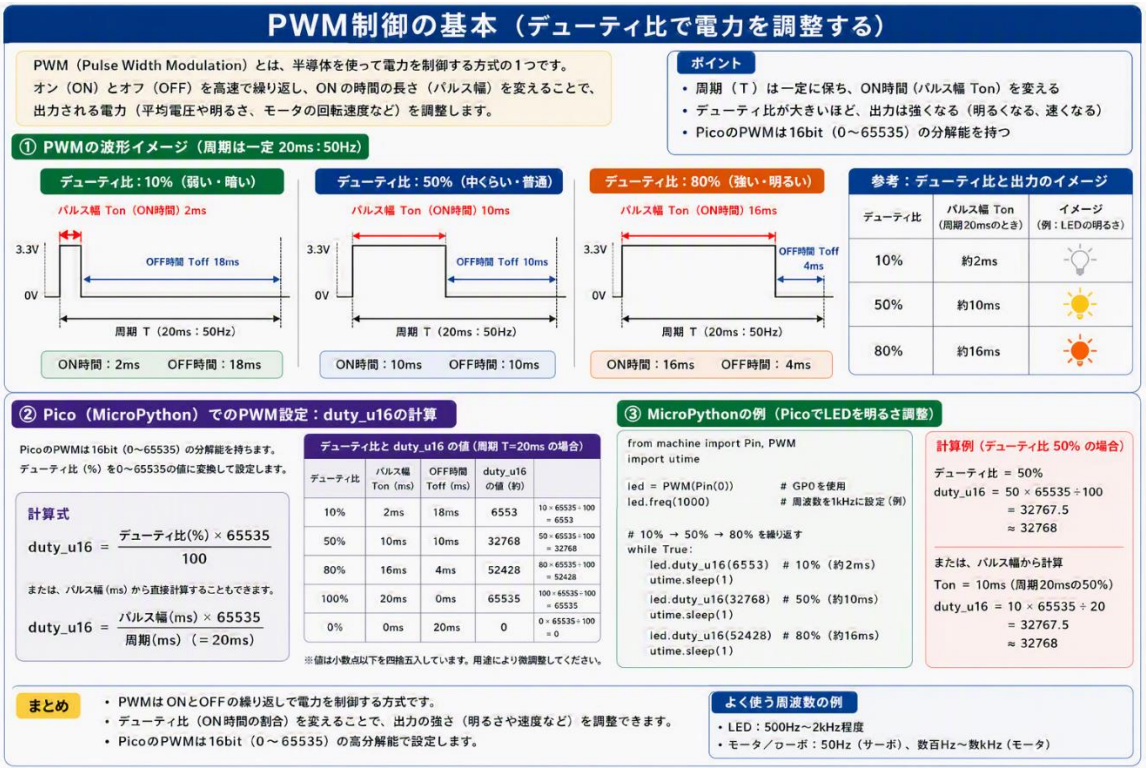
※PICO2W はプルダウンのときに動作にバグが発生するためプルアップで制御する。

課題：if sw1.value()==0:　なぜ「0」のときなのか調べてみよう。

⑩PWM 制御を理解しよう

```
1  from machine import Pin,PWM
2  import utime
3
4  L=[0,1,2,3,4]
5  sw=[5,6]
6  t=0.5
7
8  n=0
9  LEP=PWM(Pin(L[n]))
10 LEP.freq(1000)          ※PWM の設定
11
12 pwd=65535/10
13
14 for x in range(10):
15     for ud in range(10):
```

```
from machine import Pin,PWM
import utime

L=[0,1,2,3,4]
sw=[5,6]
t=0.5

n=0
LEP=PWM(Pin(L[n]))
LEP.freq(1000)

pwd=65535/10

for x in range(10):
    for ud in range(10):
        LEP.duty_u16(int(pwd*ud))
        utime.sleep(0.05)
    for dd in reversed(range(10)):
        LEP.duty_u16(int(pwd*dd))
        utime.sleep(0.05)
```



このプログラムができたら次の仕様変更をしよう

タクトスイッチ1を押すと 0.5 秒間隔で点滅し、タクトスイッチ2を押すと PWM で点滅するプログラム

A　sw1 を押したら緑 LED が点滅する

B　sw2を押したら黄 LED 制御が PWM 制御で点滅する

C　sw1 を押したら PWM 制御で点滅を早くする

D　sw2 を押したら PWM 制御で点滅を遅くする

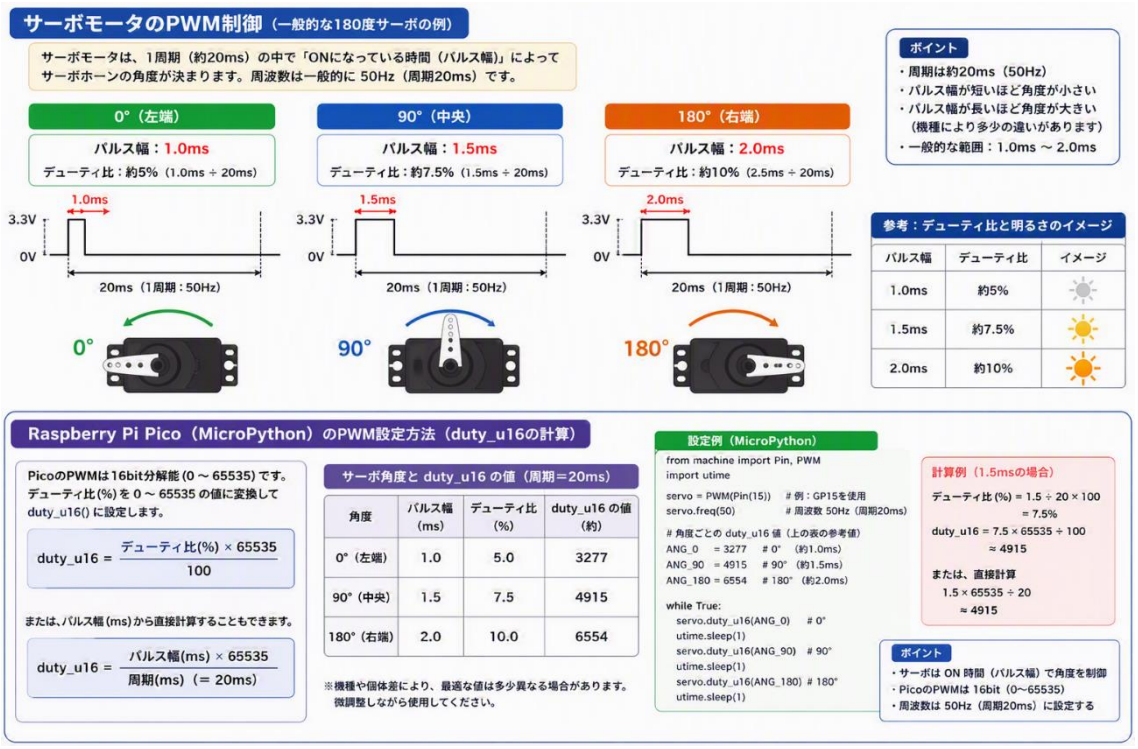
E　デューティー比を 100%から徐々に下げて消灯させる

F　sw1 を押すと右から順に LED を PWM 制御で点灯させる、sw2 を押すと左から順に LED を PWM 制御で点灯させる。

※点滅や 1 つずつ点灯させる時間の間隔は任意。

```
from machine import Pin,PWM
import utime

L=[0,1,2,3,4]
sw=[5,6]
t=0.5

n=0
LEP=PWM(Pin(L[n]))
LEP.freq(1000)

pwd=65535/10

for x in range(10):
    for ud in range(10):
        LEP.duty_u16(int(pwd*ud))
        utime.sleep(0.05)
    for dd in reversed(range(10)):
        LEP.duty_u16(int(pwd*dd))
        utime.sleep(0.05)
```

⑪サーボモータを動かしてみよう



次のとおりプログラムを変更してみよう。

デューティー比を角度で設定する。[0,90,180]

※計算式を考えてみよう。

```
from machine import Pin,PWM
import utime

servo1=PWM(Pin(15))
servo1.freq(50)

du=[1638,4751,7864]
#デューティ比に 65535 をかけた値
for x in range(3):
    servo1.duty_u16(int(du[x]))
    utime.sleep(2)
```

プログラムを変えて次の動作をさせてみよう。

A　サーボをボタンでコントロールしよう。

ボタンを押さなければサーボホーンを真ん中に（90°）

タクトスイッチ1を押すとサーボホーンを左（0°）

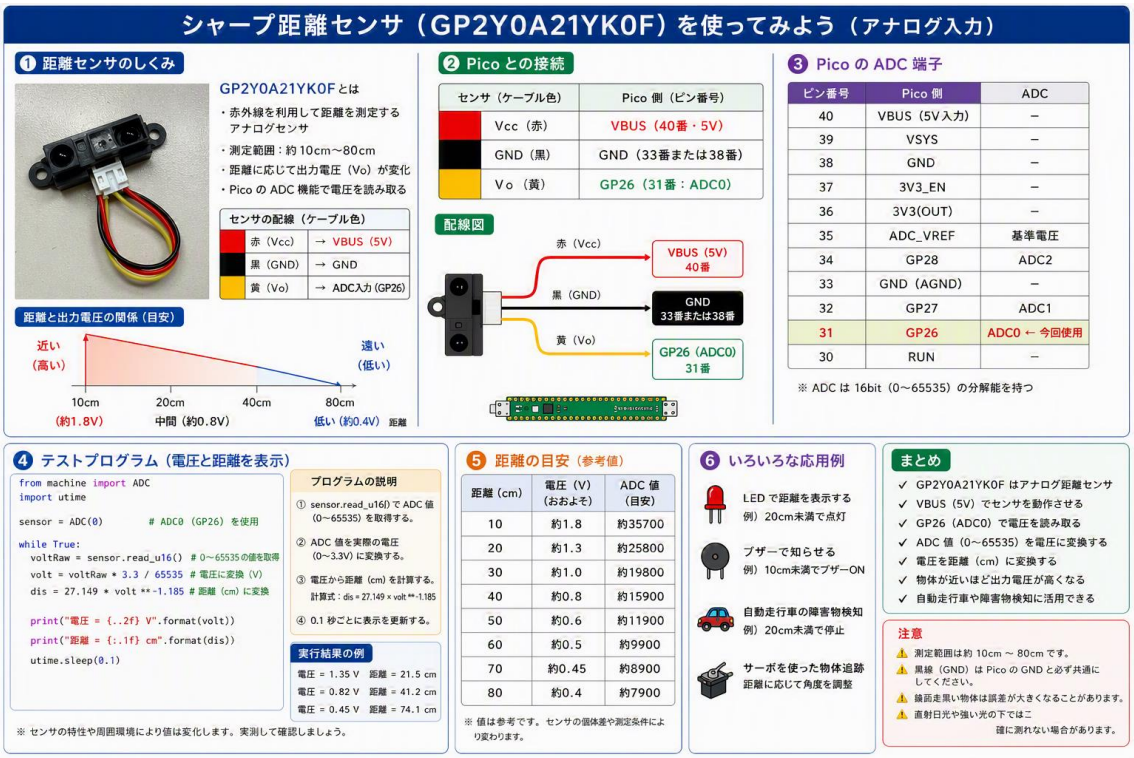
タクトスイッチ2を押すとサーボホーンを右（180°）

B　サーボホーンを左右に動かすとき、PWM 制御を使ってゆっくり動かしてみよう。

※【もしできたら】もし、ボタンを押し続けた場合、サーボホーンを右あるいは左に固定する

C　戻るときもゆっくり戻してみよう。

⑫距離センサを使ってみよう（AD 変換）




```
from machine import ADC
import utime

sensor = ADC(0)

while True:
    adc_raw = sensor.read_u16()
    voltage = adc_raw * 3.3 / 65535
    print("ADC =", adc_raw)
    print("V =", round(voltage, 2))

    utime.sleep(0.5)
```

A　プログラムを変えて次の動作をさせてみよう。

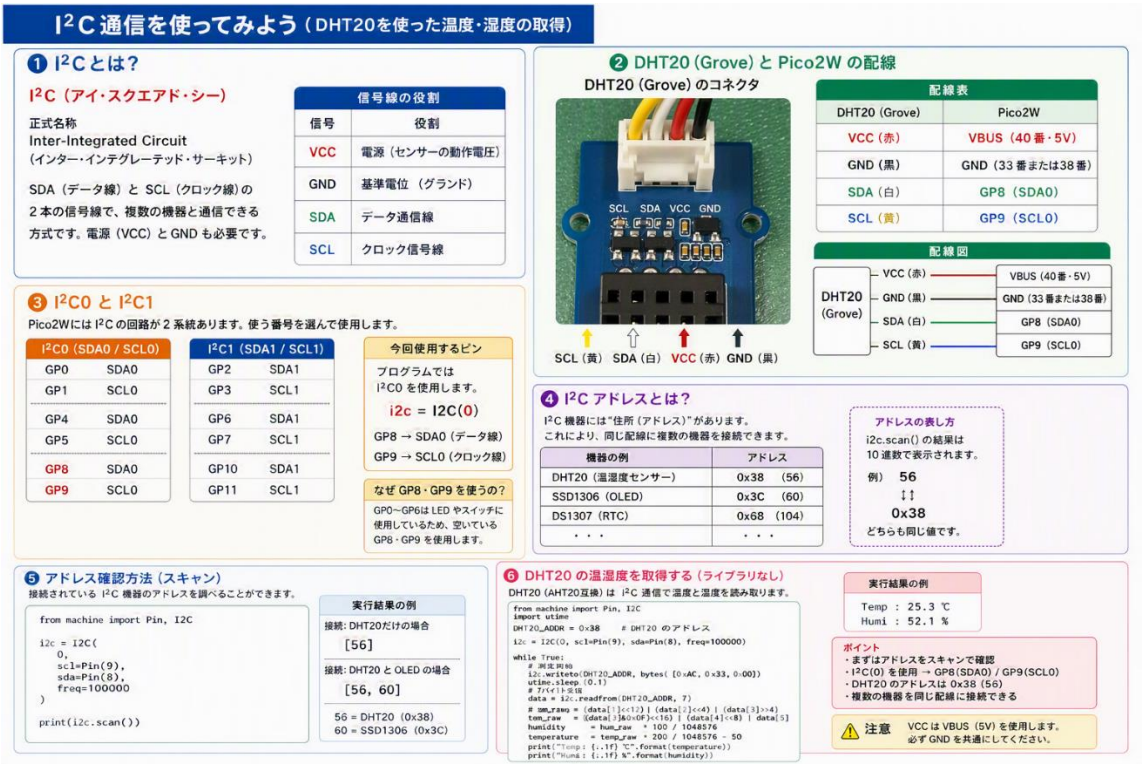
※センサーに誤差があるので実際に距離がどのくらいの adc_raw かを調べてみよう

30cm 以上なら緑の LED が点灯する

20cm 以上なら黄色の LED が点灯する。

10cm 以下なら赤色の LED が点灯する。

⑬i2c 通信でセンサーを使ってみよう

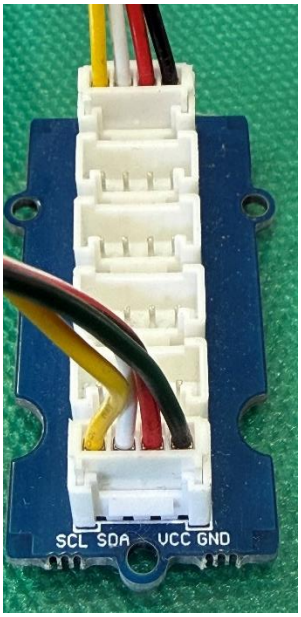
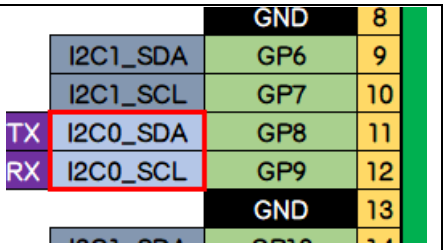


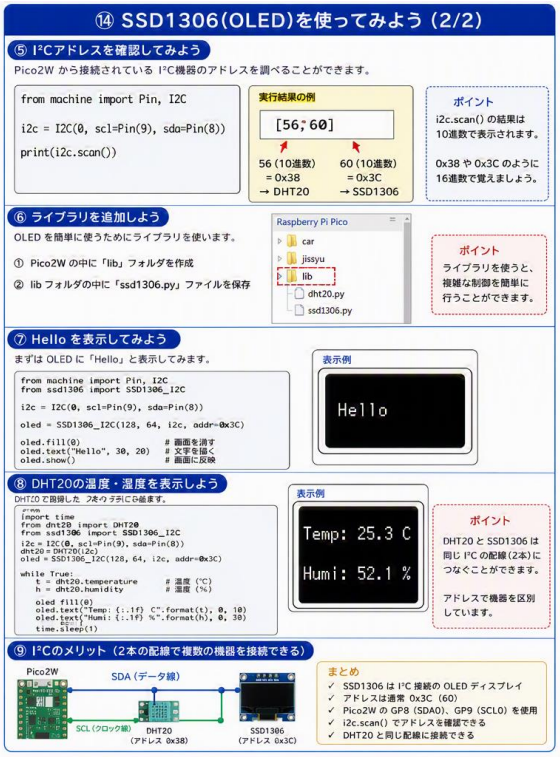
今回は DHT20 というセンサーを I2C 通信で接続し、気温を取得します。

LED やスイッチの配線があるので次に使う GP8 と GP9 に接続します。

GPIO ピンは、I2C 通信の SDA や SCL としても利用できます。

I2C の赤い Vcc は 5V、GND はラズパイに GND に接続してください。

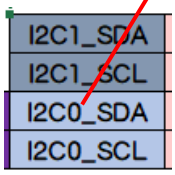
		 VCC（赤）は5V GND（黒）は GND SDA（白）は GP8（SDA） SCL（黄）は GP9（SCL）
Grove DHT20	Grove I2C ハブ	



まずは接続が間違えていないか I2C に必要なアドレスを確認してみよう。

i2c = I2C(0,scl=Pin(9),sda=Pin(8))

```
from machine import Pin, I2C
i2c = I2C(0,scl=Pin(9),sda=Pin(8))
print(i2c.scan())
```

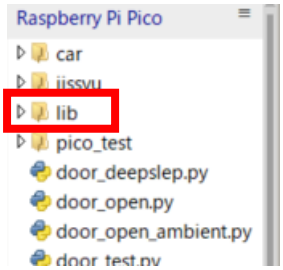


I2C のポートは 2 つあるので I2C"0"の場合はプログラムにも"0"となります。I2C1 の場合は"1"

チェックした答えは[56]と出ますが、この値は 10 進数ですので、16 進数に直します。
プログラムで使うのは 16 進数ですので[0x38]となります。

今後はスムーズにプログラムを作るためにライブラリというある機種を動かすためのプログラムを使って動作させます。

PICO2W の中に「lib」というフォルダを作ります。その中にdht20.py というプログラムを保存します。



アドレスがわかったので次は温度が取得できるか右のプログラムを打ち込んで動作確認をしてみよう。

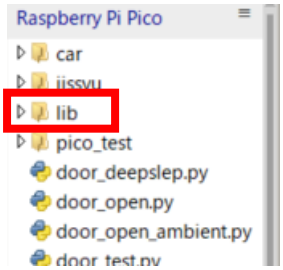
```
from machine import Pin, I2C
import  time   # タイムスリープを使うために追加
from dht20 import DHT20

i2c = I2C(0, scl=Pin(9), sda=Pin(8))
dht20 = DHT20(i2c)

while True:
    t = dht20.temperature
    h = dht20.humidity

    print("tem:", t)
    print("Hum:", h)
    print("----------------")

    time.sleep(1) # 1 秒待って繰り返す
```

⑭i2c 通信でディスプレイを使ってみよう

温度センサと同様にディスプレイも配線します。
配線後はもう一度 i2c チェックのプログラムを動作させると今度は[56, 60]と2つ出ます。
[60]は 10 進数ですので 16 進数になるので[0x3C]がディスプレイのアドレスです。

今後はスムーズにプログラムを作るためにライブラリというある機種を動かすためのプログラムを使って動作させます。

PICO2W の中に「lib」というフォルダを作ります。その中に ssd1306.py というプログラムを保存します。



右のプログラムを入力させ、実行すると”Hello”と出力することを確認します。

```
from machine import I2C, Pin
import time
from ssd1306 import SSD1306_I2C

# 接続設定（ピン番号は環境に合わせてください）
i2c = I2C(0, scl=Pin(9), sda=Pin(8), freq=400000)

WIDTH = 128
HEIGHT = 64

oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)

oled.fill(0)
oled.text("Hello", 0, 0, 1)
oled.show()
```

A　DHT20 で取得した気温や湿度を表示させてみましょう。

```
from machine import Pin, I2C
import time
from dht20 import DHT20
from ssd1306 import SSD1306_I2C

# 1. I2C の共通設定（周波数は 400kHz でどちらも正常に動きます）
i2c = I2C(0, scl=Pin(9), sda=Pin(8), freq=400000)

# 2. ディスプレイとセンサーの初期化
WIDTH = 128
HEIGHT = 64
oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
dht20 = DHT20(i2c)

print("温湿度表示プログラムを開始します。")

while True:
    t = dht20.temperature
    h = dht20.humidity

    oled.fill(0)

    str_t = "Temp: " + str(t) + " C"
    str_h = "Humi: " + str(h) + " %"

    oled.text(str_t, 0, 10, 1) #  上側に気温を表示
    oled.text(str_h, 0, 30, 1) #  下側に湿度を表示

    oled.show()

    print("T:", t, " H:", h)

    time.sleep(1)
```