

Raspberry Pi Pico W MultiAirSensorBoard

サンプルプログラム ユーザーガイド

Version 1.1

2024/1/2



目次

1. はじめに	3
2. サンプルプログラムの概要	3
3. プログラムのインストール方法	4
3.1. PC から Pico W に接続してプログラムの読み書きが出来るようにする	4
3.2. Thonny などの IDE から Pico W にプログラムを書き込む.....	6
4. プログラムの実行方法	9
5. 各プログラムの説明	10
5.1. 単機能のプログラム	12
5.1.1. OLED に文字を表示する	12
5.1.2. DHT20 から温度、湿度データを取得する	12
5.1.3. LPS25HB から温度、気圧データを取得する	13
5.1.4. MH-Z19C から CO2 データを取得する.....	14
5.1.5. SCD30 から温度、湿度、CO2 データを取得する	14
5.1.6. マルチエアースセンサーボード上の LED を光らせる	16
5.1.7. マルチエアースセンサーボード上のタクトスイッチが押された事を検知する	16
5.2. 総合的なプログラム	17
5.2.1. 3つのセンサーから環境データを取得し、それを OLED に表示する（シンプルな構成）	17
5.2.2. 3つのセンサーから環境データを取得し、それを OLED に表示する	18
5.2.3. 3つのセンサーから環境データを取得し、それを OLED に表示する& Ambient にアップロードする	20
5.2.3.1. サンプルプログラムの説明	20
5.2.3.2. Ambient にデータアップロードするための設定.....	22
6. ライセンスについて	25
7. おわりに.....	25

1. はじめに

この資料は、fireflake 製マルチエアースセンサーボードのためのサンプルプログラム、の説明資料です。

サンプルプログラムを使う際には、お手元に、Raspberry Pi Pico W, fireflake 製ベースボード, fireflake 製マルチエアースセンサーボードの 3 点があり、下の画像のように組み立ててあるものとします。



基本的に本資料の情報が正となりますが、サンプルプログラムのバージョンによっては、本資料の説明と動作が異なるかもしれません。その場合にはサンプルプログラムそれ自体を正とします。

2. サンプルプログラムの概要

マルチエアースセンサーボード用サンプルプログラムは MicroPython で書かれています。MicroPython は組み込み向けに最適化された Python です。Pico W では C/C++ も動作しますが、MicroPython の方が手軽に使う事が出来るため、これを採用しました。お好みに応じて C, C++ を使うことも、もちろん可能です。

サンプルプログラムには、大きく分けて 2 種類のプログラムが含まれています。

ひとつは単機能のプログラムです。これはマルチエアースセンサーボード上のセンサーモジュールからデータを取得する、あるいは OLED に文字を表示する、といったシンプルなプログラムです。

もうひとつは総合的なプログラムです。これは各センサーモジュールから取得したデータを OLED に表示する、といったループを繰り返すプログラムです。

どちらも、プログラム本体と関連ライブラリーを Pico W にインストールするだけで動作します。（ただしごく一部のプログラムは、別途設定が必要です。これについては各々のプログラムの説明のところで説明します。）

3. プログラムのインストール方法

ここでは、サンプルプログラムを Pico W にインストールする方法を説明します。

3.1. PC から Pico W に接続してプログラムの読み書きが出来るようにする

この方法については、Raspberry Pi 公式サイトの説明が、（英語ですが）分かりやすくまとまっています。

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

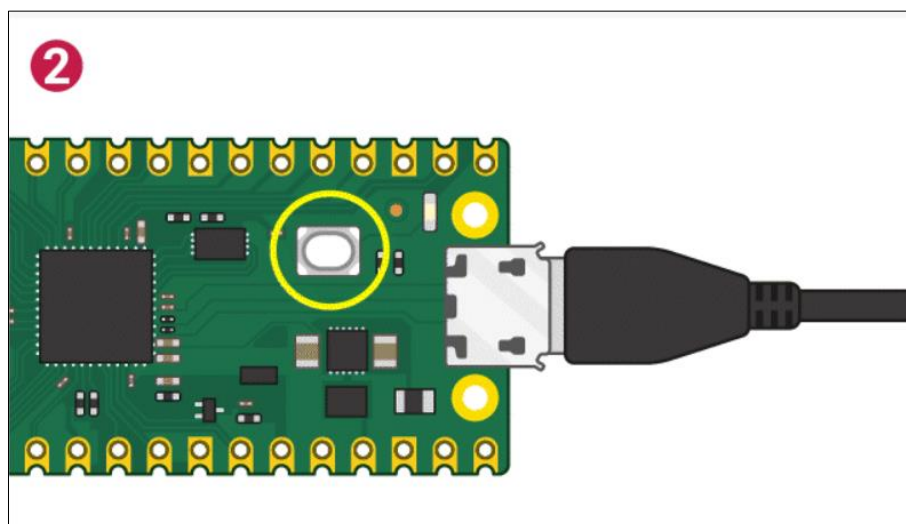
以下は、公式サイトの説明を抜粋したものになります。

まず、公式サイトから UF2 ファイルをダウンロードしてください。「Download the correct MicroPython UF2 file for your board:」と書いてある箇所で「Raspberry Pi Pico W」のダウンロードリンクをクリックしてください。

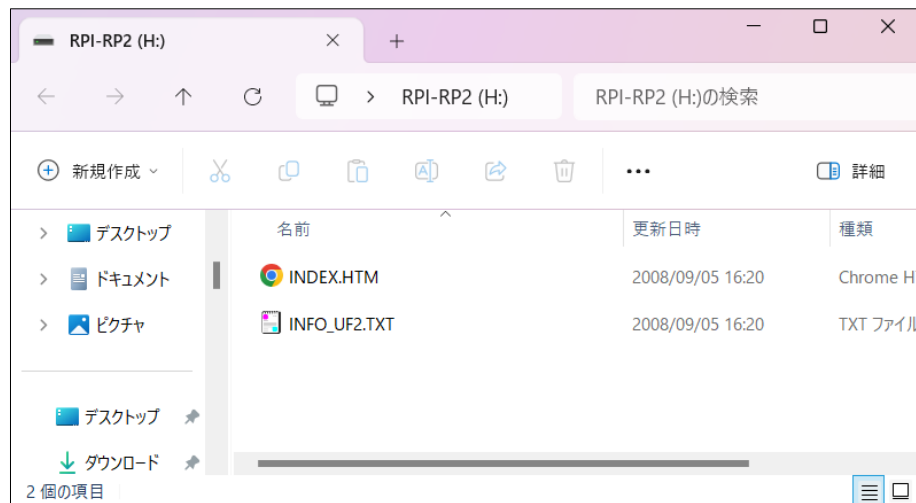
Download the correct MicroPython UF2 file for your board:

- Raspberry Pi Pico
- Raspberry Pi Pico W with Wi-Fi and Bluetooth LE support

Pico W の BOOTSEL ボタンを押したままにして、PC の USB ポートに接続します。接続出来たら、BOOTSEL ボタンをはなします。



Pico W が、RPI-RP2 という大容量ストレージデバイスとして認識されます。Windows の場合、RPI-RP2 のエクスプローラー画面が開きます。



さきほどダウンロードした MicroPython UF2 ファイルを RPI-RP2 のエクスプローラー画面にドラッグ&ドロップします。すると Pico W が再起動して、内部で MicroPython が実行され、USB シリアル経由でプログラムの読み書きと実行ができるようになります。

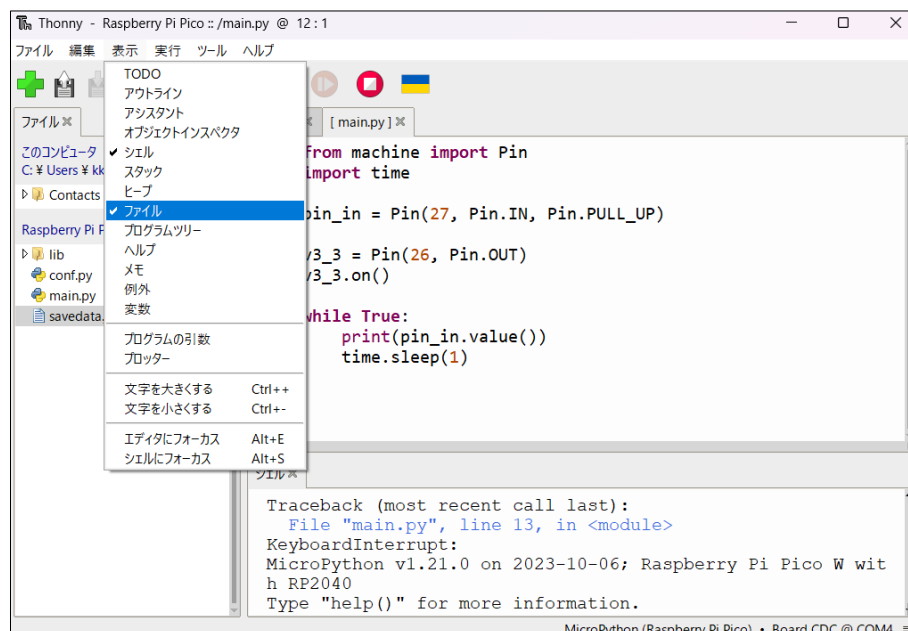
3.2. Thonny などの IDE から Pico W にプログラムを書き込む

サンプルプログラムを Pico W に書き込むには、MicroPython 向けの IDE を使うと便利です。ここでは Thonny を例にして説明します。

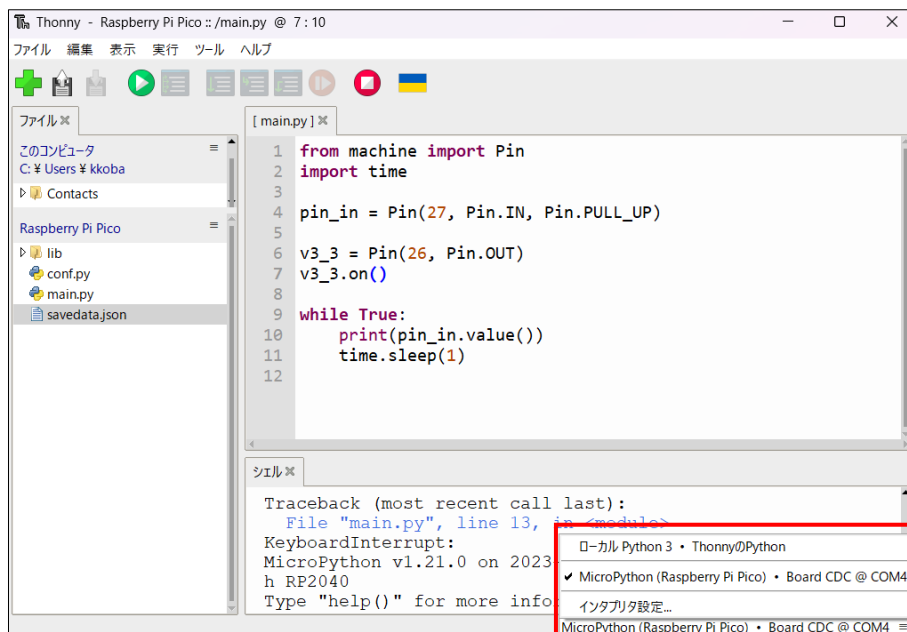
Thonny 公式サイト(<https://thonny.org/>)

Thonny については Web 上に使い方やインストール方法の情報が多くあるので、ここではその説明は行いません。ここでは Thonny のインストールと起動が済んでいるものとして説明します。

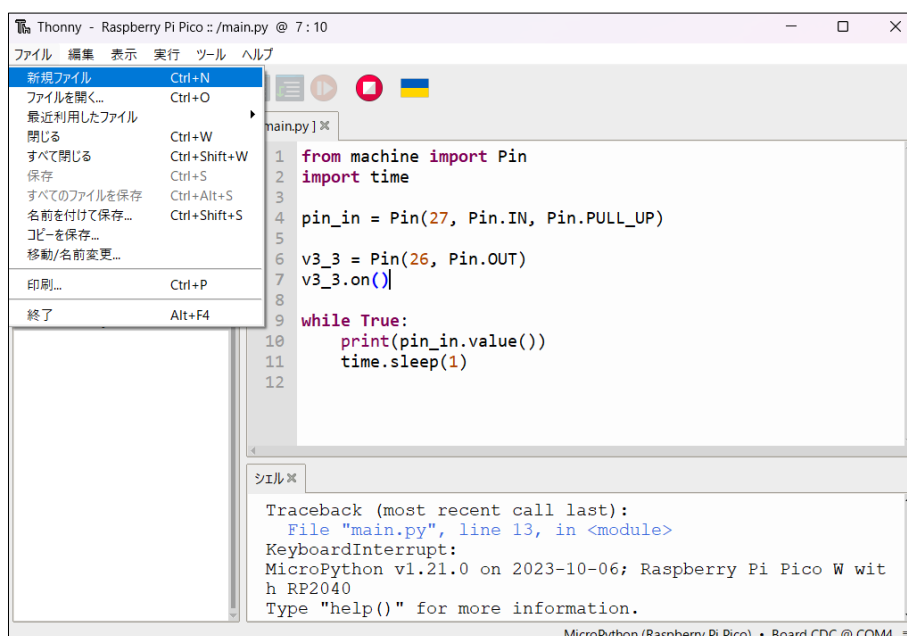
Thonny の表示メニューを開いて、ファイル表示にチェックを入れてください。すると左側にファイルツリーが表示されます。ここでファイルツリーの中に、先ほど PC に接続した Pico W が「Raspberry Pi Pico」として表示されているので、それを確認してください。



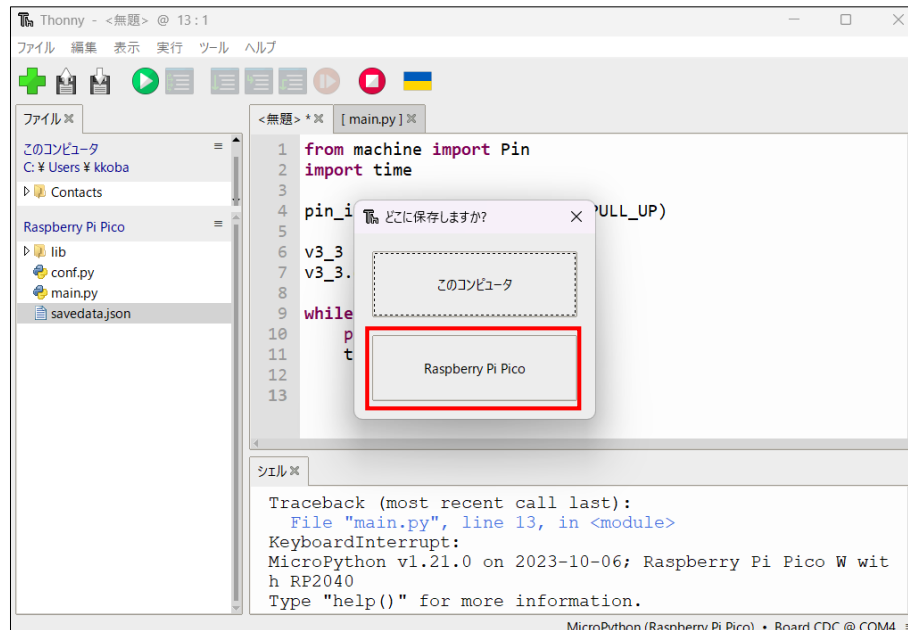
もしもない場合は、Thonny 右下のシリアルポート割り当てメニューを開いて、Pico W が接続されているシリアルポートを選択してください。



Pico W がある事を確認したら、ファイルメニューから「新規ファイル」を選びます。無題のファイルが開くので、インストールしたいサンプルプログラムのコードをそこにコピーします。



無題のファイルを、サンプルプログラムのファイルと同じ名前を付けて Pico W に保存します。これを繰り返して、動かしたいサンプルプログラムのコードをすべて Pico W に保存してください。lib フォルダの作成が必要なプログラムの場合は、ファイルツリーの中で右クリックして「新しいディレクトリ」メニューを選び、lib と名付けて OK ボタンを押すと lib フォルダが出来ます。サンプルプログラムをすべて Pico W に保存したらインストールは完了です。

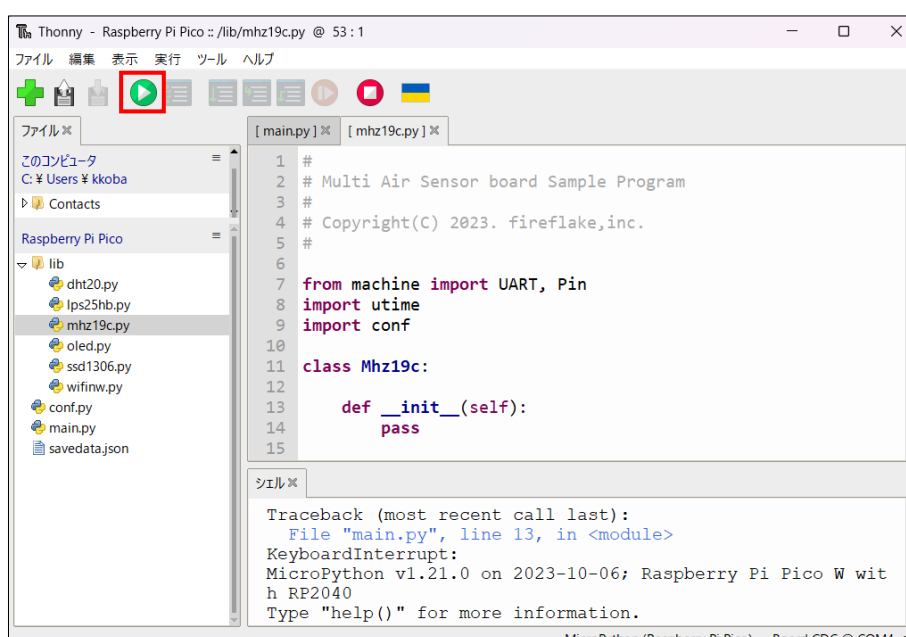


4. プログラムの実行方法

ここでは、サンプルプログラムを Pico W 上で実行する方法を説明します。

プログラムの実行には 2 種類の方法があります。ひとつは Pico W と Thonny を接続して、Pico W に入っているプログラムを Thonny で開いて編集して保存し※、その状態で実行ボタンを押す方法です。これを行うと Pico W 上でそのプログラムが実行されます。

※：Pico W に入っているプログラムを Thonny で開いて編集し、保存せずに実行ボタンを押すと、そのプログラムが Pico W のフラッシュメモリに書き込まれたうえで実行されます。ただし複数のファイルを編集してひとつも保存せずに実行ボタンを押すと、フロントに表示されているプログラムだけがフラッシュメモリに書き込まれて実行されます。



もうひとつは、Pico W にただ電源投入する方法です。分かりやすいイメージとしては、Pico W が載ったベースボードに AC アダプタから電源供給して、独立電源で動作させるというものです。これを行うと、自動的に Pico W にインストールされたプログラムが実行されます。この場合、プログラムは必ず main.py が実行されるという制約がある※ので注意してください。またこの方法で動かすと、エラーやログはどこにも表示されません。エラーやログを確認するには、それらを何らかのデバイス（例えば OLED）に表示するなどの仕組みを組んだうえでプログラムを実行する必要があります。

※：boot.py という名前のファイルが Pico W にインストールされていれば、それは main.py より先に動作しますが、べつの目的があって用意された仕組みなので、通常は main.py から実行してください

5. 各プログラムの説明

ここからは、各サンプルプログラムの説明をします。サンプルプログラムは、
ff-pico-examples/multiairsensors/src/以下に、下記の構成で置いてあります。
(工事中と記載してあるものは今後置かれる予定です。)

src

└conf.py

└conf_ambient.py (工事中)

└main.py

└main_ambient.py (工事中)

└main_simple.py

└lib/

└dht20.py

└lsp25hb.py

└mhz19c.py

└multiair_led.py (工事中)

└multiair_sw.py (工事中)

└oled.py

└scd30.py (工事中)

└ssd1306.py

サンプルプログラムの機能は下記になります。

◆単機能のプログラム

センサー系のサンプルプログラムには、マルチエアースセンサーボードにデフォルトで搭載する事を想定している DHT20, LPS25HB, MH-Z19C 用のものに加え、SCD30 用のサンプルプログラムを入れています。SCD30 は他のセンサーよりも高額ですが、MH-Z19C では代替出来ない精度やキャリブレーションの仕組みがあるため、これを使った空器環境測定のニーズも多いだろうと考えたからです。

- OLED に文字を表示する
- DHT20 から温度、湿度データを取得する
- LPS25HB から温度、気圧データを取得する
- MH-Z19C から CO2 データを取得する
- SCD30 から温度、湿度、CO2 データを取得する
- LED を光らせる
- タクトスイッチが押された事を検知する

※注意

LPS25HB などの上記のモジュール名は、マルチエアースセンサーボードに載る事が想定されるモジュールの正式名称ではなく略称です。モジュールの正式名称や、それが買える EC サイトについては「[マルチエアースセンサーボード仕様書](#)」をご確認ください。正しくないモジュールをマルチエアースセンサーボードに接続した場合、サンプルプログラムは正しく動作しない場合がありますのでご注意ください。

◆総合的なプログラム

全ての総合的なプログラムでは、温度データを取れるセンサーとして DHT20 と LPS25HB を扱っていますが、取得するのは DHT20 の温度データのみです。

- 3つのセンサーから環境データを取得し、それを OLED に表示する（シンプル構成）
- 3つのセンサーから環境データを取得し、それを OLED に表示する
- 3つのセンサーから環境データを取得し、それを OLED に表示する & Ambient にアップロードする

5.1. 単機能のプログラム

5.1.1. OLED に文字を表示する

- このプログラムは、128x32 ドット、0.91 インチの有機 EL ディスプレイ（OLED）に文字を表示するプログラムです。OLED には最大で 4 行の表示が行えます。Pico W と OLED は I2C で通信します。
- Thonny から動作させる場合、下記の構成でサンプルプログラムを Pico W にインストールし、oled.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico（Pico W は Thonny 上ではこのように表記されます）

└oled.py

└ssd1306.py

- 独立電源で動作させる場合は、上記の構成でサンプルプログラムを Pico W にインストールし、oled.py のファイル名を main.py に変えたうえで、Pico W に電源投入してください。

5.1.2. DHT20 から温度、湿度データを取得する

- このプログラムは、温湿度センサーの DHT20 から温度と湿度を取得するプログラムです。Pico W と DHT20 は I2C で通信します。
- このプログラムは、動作の初めに DHT20 と通信出来るかを確認します。通信出来ない場合はエラーを出力します。以降、DHT20 と通信して環境データを取得するときは、取得失敗しても 6 回までリトライしますが、全て失敗するとエラーを出力します。
- Thonny から動作させる場合、下記の構成でサンプルプログラムを Pico W にインストールし、dht20.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico（Pico W は Thonny 上ではこのように表記されます）

└dht20.py

- 独立電源で動作させる場合は、上記の構成でサンプルプログラムを Pico W にインストールし、dht20.py のファイル名を main.py に変えたうえで、Pico W に電源投入してください。ただこれをして、DHT20 から取得したデータを OLED に表示する仕組みなどがなければ、見た目上は何も起きません。

5.1.3. LPS25HB から温度、気圧データを取得する

- このプログラムは、温度気圧センサーの LPS25HB から温度と気圧を取得するプログラムです。Pico W と LPS25HB は I2C で通信します。
- このプログラムは、動作の初めに LPS25HB と通信出来るかを確認します。通信出来ない場合はエラーを出力します。以降、LPS25HB と通信して環境データを取得するときは、取得失敗しても 6 回までリトライしますが、全て失敗するとエラーを出力します。
- Thonny から動作させる場合、下記の構成でサンプルプログラムを Pico W にインストールし、lps25hb.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico (Pico W は Thonny 上ではこのように表記されます)

└─lps25hb.py

- 独立電源で動作させる場合は、上記の構成でサンプルプログラムを Pico W にインストールし、lps25hb.py のファイル名を main.py に変えたうえで、Pico W に電源投入してください。ただこれをして、LPS25HB から取得したデータを OLED に表示する仕組みなどがなければ、見た目上は何も起きません。

5.1.4. MH-Z19C から CO2 データを取得する

- このプログラムは、CO2 センサーの MH-Z19C から CO2 を取得するプログラムです。Pico W と MH-Z19C は UART で通信します。
- このプログラムは、動作の初めに MH-Z19C から CO2 データを取得出来るかを確認します。取得出来ない場合はエラーを出力します。以降、MH-Z19C と通信して環境データを取得するときは、取得失敗しても 6 回までリトライしますが、全て失敗するとエラーを出力します。
- MH-Z19C は起動後のプレヒートに 1 分を要しますが、このプログラムはプレヒート中もデータを取り込んで表示します。プレヒート中に表示されるデータは正しい値ではありません。
- Thonny から動作させる場合、下記の構成でサンプルプログラムを Pico W にインストールし、mhz19c.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico (Pico W は Thonny 上ではこのように表記されます)

└─mhz19c.py

- 独立電源で動作させる場合は、上記の構成でサンプルプログラムを Pico W にインストールし、mhz19c.py のファイル名を main.py に変えたうえで、Pico W に電源投入してください。ただこれをして、MH-Z19C から取得したデータを OLED に表示する仕組みなどがなければ、見た目上は何も起きません。

※MH-Z19C のキャリブレーションについて

MH-Z19C は他の CO2 センサー同様セルフキャリブレーション機能を有しています。これは屋外の大気の CO2 濃度が 400ppm 前後（2023 年の日本の CO2 濃度 420ppm 前後です）であることを利用した校正になります。セルフキャリブレーション機能を ON/OFF する関数は、サンプルプログラムのなかに含まれています。

セルフキャリブレーションモードでは 24 時間の間の CO2 の濃度計測の結果の最低値を元に補正を行います。24 時間以内のサイクルにおいて十分な換気がなされる環境ではセルフキャリブレーションが利用可能ですが、換気が不十分な環境ではセルフキャリブレーションを OFF にして利用してください。

MH-Z19C ならびに、セルフキャリブレーションの詳細についてはデータシートを参照してください。

参考：気象庁 - 大気中二酸化炭素濃度の観測結果

https://ds.data.jma.go.jp/ghg/kanshi/obs/co2_monthave_ryo.html

5.1.5. SCD30 から温度、湿度、CO2 データを取得する

- このプログラムは、温湿度 CO2 センサーの SCD30 から温度、湿度、CO2 を取得するプログラムです。Pico W と SCD30 は I2C で通信します。
- このプログラムは、動作の初めに SCD30 と通信出来るかを確認します。通信出来ない場合はエラーを出力します。以降、SCD30 と通信して環境データを取得するときは、取得失敗しても 6 回までリトライしますが、全て失敗するとエラーを出力します。

- Thonny から動作させる場合、下記の構成でサンプルプログラムを Pico W にインストールし、scd30.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico (Pico W は Thonny 上ではこのように表記されます)

└scd30.py

- 独立電源で動作させる場合は、上記の構成でサンプルプログラムを Pico W にインストールし、scd30.py のファイル名を main.py に変えたうえで、Pico W に電源投入してください。ただこれをして、SCD30 から取得したデータを OLED に表示する仕組みなどがなければ、見た目上は何も起きません。

5.1.6. マルチエアースセンサーボード上の LED を光らせる

- このプログラムは、マルチエアースセンサーボード上の 3 つの LED を光らせるプログラムです。
- Thonny から動作させる場合、下記の構成でサンプルプログラムを Pico W にインストールし、multiair_led.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico (Pico W は Thonny 上ではこのように表記されます)

└multiair_led.py

- 独立電源で動作させる場合は、上記の構成でサンプルプログラムを Pico W にインストールし、multiair_led.py のファイル名を main.py に変えたうえで、Pico W に電源投入してください。

5.1.7. マルチエアースセンサーボード上のタクトスイッチが押された事を検知する

- このプログラムは、マルチエアースセンサーボード上のタクトスイッチが押された事を検知し、押された場合はコンソールにそれを表示するプログラムです。
- Thonny から動作させる場合、下記の構成でサンプルプログラムを Pico W にインストールし、multiair_sw.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico (Pico W は Thonny 上ではこのように表記されます)

└multiair_sw.py

- 独立電源で動作させる場合は、上記の構成でサンプルプログラムを Pico W にインストールし、multiair_sw.py のファイル名を main.py に変えたうえで、Pico W に電源投入してください。ただこれをして、タクトスイッチが押された事を検知してそれを OLED に表示する仕組みなどがなければ、見た目上は何も起きません。

5.2. 総合的なプログラム

5.2.1. 3つのセンサーから環境データを取得し、それを OLED に表示する（シンプルな構成）

- このプログラムは、タイトルにあるように「3つのセンサーから環境データを取得し、それを OLED に表示する」動作を行う総合的なプログラムです。
- このプログラムは、処理の流れが分かりやすいように、他の2つの総合的なプログラムと比較して、シンプルに組んでいます。ですので長期稼働のための実用的な機構は含めていません。**エラーが出力された場合は、再起動したり OLED に表示するといった処理はせずそこで停止します。**
- このプログラムは、周期的に3つのセンサーからデータを取得して OLED に表示する事を繰り返します。初期の動作周期は5秒です。
- このプログラムには、conf.py という設定用ファイルが含まれます。conf.py に含まれる設定は I2C チャンネル情報、UART チャンネル情報、I2C デバイスアドレス、ループ待ち時間などです。conf.py は初期設定のまま使う事が可能です。必要があれば編集して使ってください。
- Thonny から動作させる場合、下記の構成でプログラムを Pico W にインストールし、main.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico（Pico W は Thonny 上ではこのように表記されます）

```
└main.py (main_simple.py を左記に名称変更してインストールしてください)
└conf.py
└ lib/
    ├── dht20.py
    ├── lsp25hb.py
    ├── mhz19c.py
    ├── oled.py
    └── ssd1306.py
```

- 独立電源で動作させる場合は、上記の構成でプログラムを Pico W にインストールし、Pico W に電源投入してください。

5.2.2. 3つのセンサーから環境データを取得し、それを OLED に表示する

- このプログラムは、タイトルにあるように「3つのセンサーから環境データを取得し、それを OLED に表示する」動作を行う総合的なプログラムです。
- このプログラムは、長期稼働のための実用的な機構を含めています。エラーが出力された場合は、再起動したり OLED に表示するといった処理が行われます。そういった処理が入っている分、プログラムの可読性は落ちています。プログラムは下記のように動作します。
 1. プログラムが起動すると、OLED との通信テスト → 3つのセンサーとの通信テスト → MH-Z19C のセルフキャリブレーション OFF を順次行います。OLED との通信に失敗した場合、3つのセンサーとの通信テストに失敗した場合は、それ以上の処理を行わずにエラー出力を行います。処理を停止した状態でタクトスイッチを押すとリセットがかかり再起動します。
 2. 通信テストが終わったら、マルチエアースセンサーボード上のタクトスイッチが押されているかいないかを判定して、**通常モード**で動作するか、**デバッグモード**で動作するかを判定します。**タクトスイッチが押されていない場合は通常モード、押されている場合はデバッグモードで動作します。**
 3. 以降は周期的に3つのセンサーからデータを取得して OLED に表示する事を繰り返します。初期の動作周期は5秒です。この動作中にエラーが起きた場合は、通常モードになっているかデバッグモードになっているかで処理が分かれます。通常モードで動作した場合は、ウォッチドッグタイマが ON になり、エラーが起きると8秒後に自動再起動するようになります。デバッグモードでエラーが起きた場合は、そのエラーを OLED に出力してそれ以上の処理を行いません。また、処理を停止した状態でタクトスイッチを押すとリセットがかかり再起動します。
 4. **プログラム稼働中に、タクトスイッチを5秒以上長押ししてから離すと、OLED に「CO2 auto calibration on」と表示されて CO2 センサーのセルフキャリブレーションが ON になります。**セルフキャリブレーションについては「MH-Z19C から CO2 データを取得する」のチャプターを読んでください。
- このプログラムには、conf.py という設定用ファイルが含まれます。conf.py に含まれる設定は I2C チャンネル情報、UART チャンネル情報、I2C デバイスアドレス、ループ待ち時間などです。conf.py は初期設定のまま使う事が可能です。必要があれば編集して使ってください。
- このプログラムの**通常モード**で動作している Pico W を、PC に接続して Thonny の停止ボタンを押すと、その8秒後に Pico W が自動再起動して、Thonny と Pico W の接続が解除されます。**Thonny からの接続を解除せずに Pico W を停止させたい場合は、以下の手順を行ってください。**
 1. 通常モードで動作している Pico W を Thonny に接続します。
 2. Thonny の停止ボタンを押します。
 3. Pico W が再起動する前に、マルチエアースセンサーボード上のタクトスイッチを押しっぱなしにします。
 4. その状態で Pico W が再起動すると、Pico W がデバッグモードで動作します。
 5. その状態で Thonny の停止ボタンを押すと、ウォッチドッグタイマが OFF なので自動再起動が起きず、Thonny からの接続が解除されません。

- Thonny から動作させる場合、下記の構成でプログラムを Pico W にインストールし、main.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico (Pico W は Thonny 上ではこのように表記されます)

```
└main.py
└conf.py
└ lib/
    └dht20.py
    └lsp25hb.py
    └mhz19c.py
    └oled.py
    └ssd1306.py
```

- 独立電源で動作させる場合は、上記の構成でプログラムを Pico W にインストールし、Pico W に電源投入してください。

5.2.3. 3つのセンサーから環境データを取得し、それを OLED に表示する & Ambient にアップロードする

5.2.3.1. サンプルプログラムの説明

- このプログラムは、無償でも使える IoT データの可視化サービスの Ambient と連携し、そこにセンサーから取得した環境データをアップロードするプログラムです。Ambient については、下記公式サイト「ドキュメント」メニューから多くの情報にアクセス出来ます。また Ambient は、fireflake にとって外部のサービスになるため、それに関する質問などをいただいても応じられないケースがあります。すみませんが、その点はご了承ください。

Ambient 公式サイト (<https://ambidata.io/>)

- このプログラムは、タイトルにあるように「3つのセンサーから環境データを取得し、それを OLED に表示 & Ambient にアップロードする」動作を行う総合的なプログラムです。
- このプログラムは、長期稼働のための実用的な機構を含めています。エラーが出力された場合は、再起動したり OLED に表示するといった処理が行われます。そういった処理が入っている分、プログラムの可読性は落ちています。プログラムは下記のように動作します。
 1. プログラムが起動すると、OLED との通信テスト → 3つのセンサーとの通信テスト → MH-Z19C のセルフキャリブレーション OFF → 周囲の Wi-Fi の SSID スキャンを順次行います。OLED との通信に失敗した場合、3つのセンサーとの通信テストに失敗した場合、周囲に conf.py に書いた SSID の Wi-Fi が無い場合は、それ以上の処理を行わずにエラー出力を行います。処理を停止した状態でタクトスイッチを押すとリセットがかかり再起動します。
 2. 通信テストなどが終わったら、マルチエアースセンサーボード上のタクトスイッチが押されているかいないかを判定して、これ以降に通常モードで動作するか、デバッグモードで動作するかを判定します。タクトスイッチが押されていない場合は通常モード、押されている場合はデバッグモードで動作します。
 3. Wi-Fi 接続します。一定回数リトライして接続に失敗した場合は、通常モードになっているかデバッグモードになっているかで処理が分かれます。通常モードで動作した場合は、ウォッチドッグタイマが ON になり、エラーが起きると 8 秒後に自動再起動するようになります。デバッグモードでエラーが起きた場合は、そのエラーを OLED に出力してそれ以上の処理を行いません。また、処理を停止した状態でタクトスイッチを押すとリセットがかかり再起動します。
 4. 以降は周期的に 3つのセンサーからデータを取得して OLED に表示し、さらにそのデータを Ambient にアップロードする事を繰り返します。初期の動作周期は、Ambient サーバーに負荷をかけないように 60 秒になっています。する事を繰り返します。この動作中にエラーが起きた場合は、通常モードになっているかデバッグモードになっているかで処理が分かれます。処理の内容は 3 と同様です。
 5. プログラム稼働中に、タクトスイッチを 5 秒以上長押ししてから離すと、OLED に「CO2 auto calibration on」と表示されて CO2 センサーのセルフキャリブレーションが ON になります。セルフキャリブレーションについては「MH-Z19C から CO2 データを取得する」のチャプターを読んでください。
- このプログラムには、conf.py という設定用ファイルが含まれます。conf.py に含まれる設定は I2C チャンネル情報、UART チャンネル情報、I2C デバイスアドレス、ループ待ち時間、Wi-Fi の SSID とパスワード、Ambient の接続情報などです。このプログラムでは、conf.py は初期設定のまま使う事が出来ません。プログラムを正しく動

かすためには、Wi-Fi 情報の設定と、Ambient 接続情報の設定が必要になります。詳しくは「Ambient にデータアップロードするための設定」を読んでください。

- このプログラムの**通常モード**で動作している Pico W を、PC に接続して Thonny の停止ボタンを押すと、その 8 秒後に Pico W が自動再起動して、Thonny と Pico W の接続が解除されます。**Thonny からの接続を解除せずに Pico W を停止させたい場合は、以下の手順を行ってください。**

6. 通常モードで動作している Pico W を Thonny に接続します。
7. Thonny の停止ボタンを押します。
8. Pico W が再起動する前に、マルチエアセンサーボード上のタクトスイッチを押しっぱなしにします。
9. その状態で Pico W が再起動すると、Pico W がデバッグモードで動作します。
10. その状態で Thonny の停止ボタンを押すと、ウォッチドッグタイマが OFF なので自動再起動が起きず、Thonny からの接続が解除されません。

- Thonny から動作させる場合、下記の構成でプログラムを Pico W にインストールし、main.py を開いている画面で、Thonny の実行ボタンを押してください。

.Raspberry Pi Pico (Pico W は Thonny 上ではこのように表記されます)

└main.py (**main_ambient.py** を左記に名称変更してインストールしてください)

└conf.py (**conf_ambient.py** を左記に名称変更してインストールしてください)

└ lib/

└dht20.py

└lsp25hb.py

└mhz19c.py

└oled.py

└ssd1306.py

- 独立電源で動作させる場合は、上記の構成でプログラムを Pico W にインストールし、Pico W に電源投入してください。

5.2.3.2. Ambient にデータアップロードするための設定

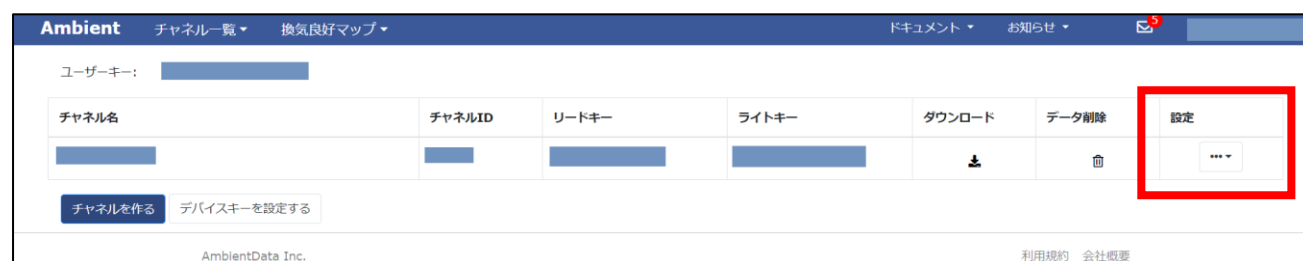
測定した環境データを Ambient にアップロードするためには、Ambient 側と Pico W 側の双方の設定をする必要があります。

◆ Ambient 側の設定

ここでの説明は粗いものになります。詳細は、下記のチュートリアルなど Ambient の公式ドキュメントを参照してください。

<https://ambidata.io/docs/>

1. Ambient にユーザー登録を行い、チャンネルをひとつ登録してください。
2. チャンネルを登録したら、設定のプルダウンリストをクリックし、表示される設定変更をクリックしてください。



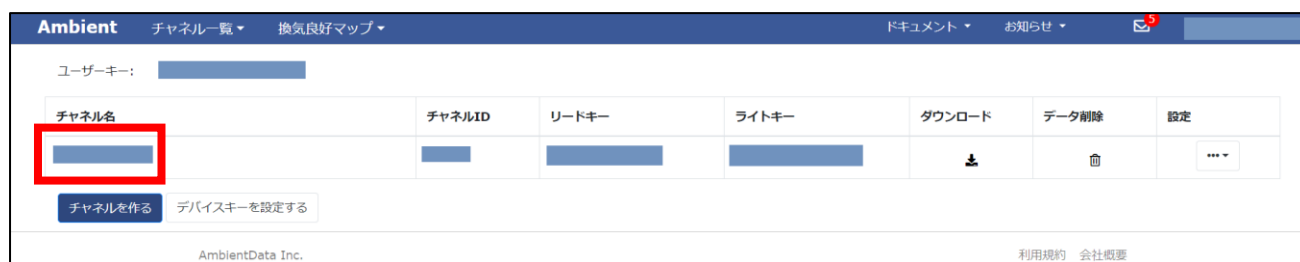
The screenshot shows the Ambient web interface. At the top, there's a navigation bar with 'Ambient', 'チャンネル一覧', '換気良好マップ', 'ドキュメント', and 'お知らせ'. Below the navigation bar, there's a section for 'ユーザーキー:'. A table lists channels with columns: 'チャンネル名', 'チャンネルID', 'リードキー', 'ライトキー', 'ダウンロード', 'データ削除', and '設定'. The '設定' column has a dropdown menu icon. A red box highlights this '設定' button. Below the table, there are buttons for 'チャンネルを作る' and 'デバイスキーを設定する'. At the bottom, it says 'AmbientData Inc.' and '利用規約 会社概要'.

3. 表示される設定画面に下記のように入力し、チャンネル属性を設定するボタンを押してください。



The screenshot shows the Ambient channel settings page. It has a header bar. Below it, there's a form with the following fields: 'チャンネル名' (Channel Name) with the value '空気環境測定チャート', '説明' (Description) with the value '説明', and a list of data points. The data points are arranged in two columns. The first column has 'データ1' (Temperature), 'データ3' (Air Pressure), 'データ5' (Data Name 5), and 'データ7' (Data Name 7). The second column has 'データ2' (Humidity), 'データ4' (CO2), 'データ6' (Data Name 6), and 'データ8' (Data Name 8). Each data point has a color-coded square next to it. The 'データ4' field is highlighted with a blue box. At the bottom, there are fields for '緯度' (Latitude) and '経度' (Longitude), both with the value '0.0'.

4. チャンネル名をクリックして、チャンネル内部情報の画面に移動してください。



チャンネル名	チャンネルID	リードキー	ライトキー	ダウンロード	データ削除	設定
[Channel Name]	[Channel ID]	[Lead Key]	[Light Key]	[Download Icon]	[Delete Icon]	[Settings Icon]

チャンネルを作る デバイスキーを設定する

AmbientData Inc. 利用規約 会社概要

5. チャート作成ボタンを押して、1 つめのチャートを作ります。



Ambient 換気良好マップ + ⚙️

6. 1 つめのチャートの設定は下記のようにします。作成例なので好きにカスタマイズしてください。設定が出来たら、設定するボタンを押して保存します。



チャート設定変更

チャンネルデータ 位置 写真

チャート名

温度、湿度

チャンネル

空気環境測定チャート

透過度

z-index

チャート種類

折れ線グラフ

d1:温度

☐ 表示なし ☒ 左軸 ☐ 右軸

d2:湿度

☐ 表示なし ☐ 左軸 ☒ 右軸

d3:気圧

☐ 表示なし ☐ 左軸 ☐ 右軸

d4:CO2

☐ 表示なし ☐ 左軸 ☐ 右軸

d5

☐ 表示なし ☐ 左軸 ☐ 右軸

d6

☐ 表示なし ☐ 左軸 ☐ 右軸

d7

☐ 表示なし ☐ 左軸 ☐ 右軸

d8

☐ 表示なし ☐ 左軸 ☐ 右軸

左軸

最小値

-20

最大値

60

log?

☐

右軸

最小値

0

最大値

100

log?

☐

軸の最小値、最大値は空白のままにすれば自動的に設定されます。

表示件数

1000

日付指定

☐

23 / 25

7. 2 つめのチャートの設定は下記のようにします。設定が出来たら、設定するボタンを押して保存します。

チャート設定変更

チャンネルデータ

位置

写真

チャート名

気圧、CO2

チャンネル

空気環境測定チャンネル

透過度

z-index

チャート種類

折れ線グラフ

d1:温度

☐ 表示なし

☐ 左軸

☐ 右軸

d2:湿度

☐ 表示なし

☐ 左軸

☐ 右軸

d3:気圧

☐ 表示なし

☒ 左軸

☐ 右軸

d4:CO2

☐ 表示なし

☐ 左軸

☒ 右軸

d5

☐ 表示なし

☐ 左軸

☐ 右軸

d6

☐ 表示なし

☐ 左軸

☐ 右軸

d7

☐ 表示なし

☐ 左軸

☐ 右軸

d8

☐ 表示なし

☐ 左軸

☐ 右軸

左軸

最小値

800

最大値

1200

log?

☐

右軸

最小値

0

最大値

5000

log?

☐

軸の最小値、最大値は空白のままにすれば自動的に設定されます。

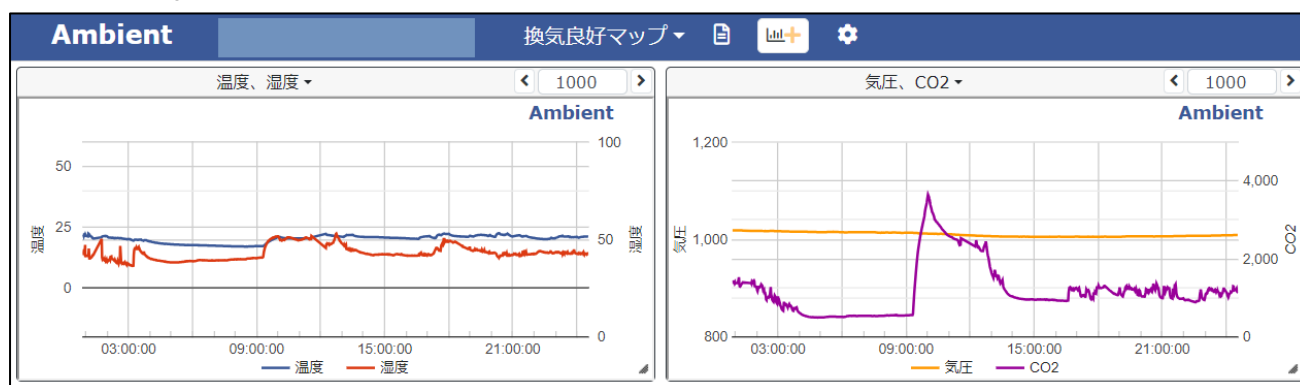
表示件数

1000

日付指定

☐

8. このようにチャートが作成できれば成功です。（この画像は作成後しばらく経っているのデータが多数アップロードされています。）



◆Pico W 側の設定

conf.py (conf_ambient.py を左記に名称変更したもの) に、Pico W が Wi-Fi 接続するための値と、Ambient にデータアップロードするためのを追記します。以下の値を文字列として追記してください。

Ambient 関連

AMBIENT_CHANNEL_ID = const("") ←Ambient のチャンネル ID を入れてください。
 AMBIENT_WRITE_KEY = const("") ←Ambient のライトキーを入れてください。

WIFI_SSID = const("")	←Pico W を接続する Wi-Fi の SSID を入れてください。
WIFI_PASSWORD = const("")	←Pico W を接続する Wi-Fi のパスワードを入れてください。

これらの設定を行ったうえで、main.py（main_ambient.py を左記に名称変更したもの）を動作させると、測定した空気環境データを Ambient にアップロードする事が出来ます。

6. ライセンスについて

サンプルプログラムのうち、fireflake 制作のプログラムは MIT ライセンスで配布されます。それ以外のプログラムは、それぞれのプログラムが配布されているライセンスに従います。fireflake 制作のプログラムは下記になります。

main.py
main_simple.py
main_ambinet.py
conf.py
conf_ambient.py
oled.py
lps25hb.py
mhz19c.py
scd30.py

7. おわりに

本資料が、皆さんが作る事を楽しめる一助になれば幸いです。Happy Hacking!!