

# Zephyr(RTOS)でOpenPLCを 実装してみた

ミソジ 2026/3/27

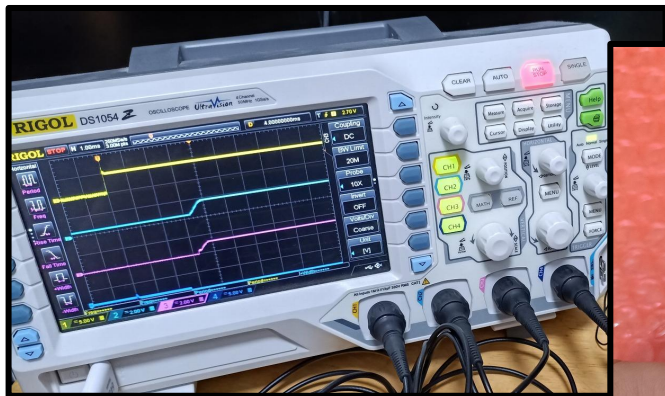
Zephyr Project meetup: Nagoya, Japan  
#ZephyrRTOS

# 自己紹介

ハードウェアのエンジニアで、趣味でブログとか書いてます

名前: ミソジ [@misoji\\_engineer](https://misoji-engineer.com/)

ブログ: エンジニアの電気屋さん(<https://misoji-engineer.com/>)



# アジェンダ

## Zephyr(RTOS)にOpenPLCを実装した話

- ・ OpenPLCとは？
- ・ LinuxをRTOSにしたら、もっと高速になるのでは？
- ・ Zephyr(RTOS)に実装した結果・メリット
- ・ まとめ

OpenPLCとは？

# OpenPLC

OpenPLC・・・オープンソースの「ソフトウェアPLC」の一つ

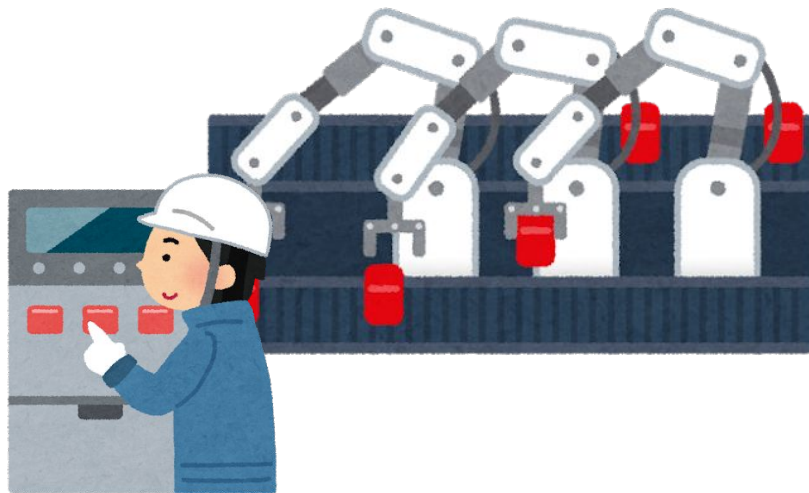
OpenPLC Editor/Runtime



<https://autonomylogic.com/>

PLC・・・ Programmable Logic Controller

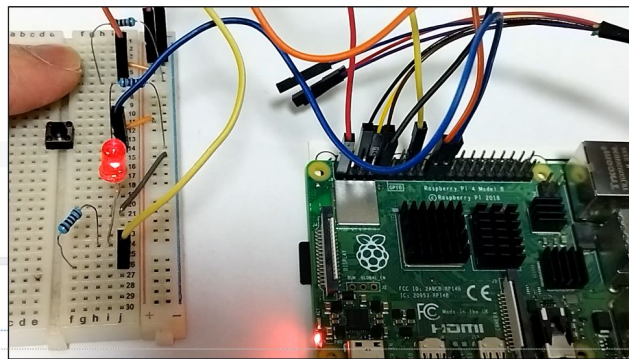
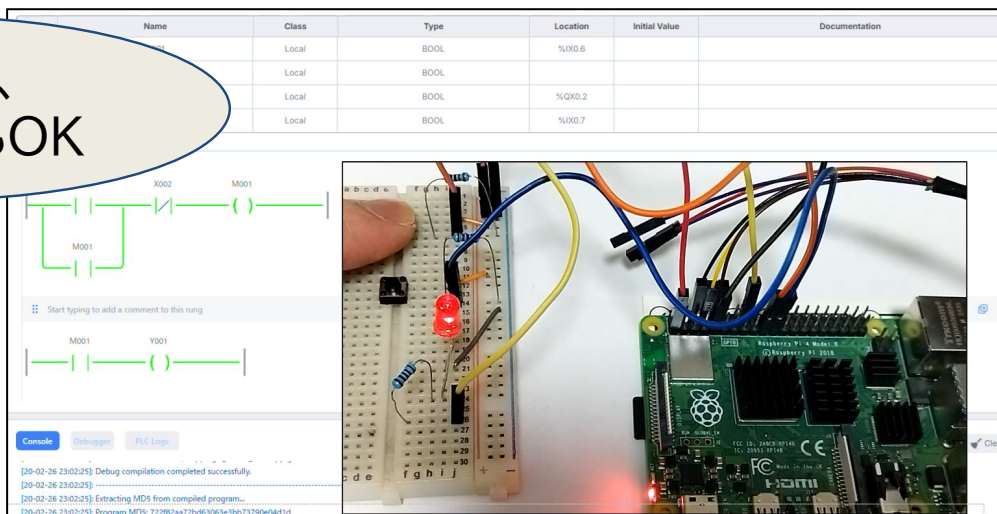
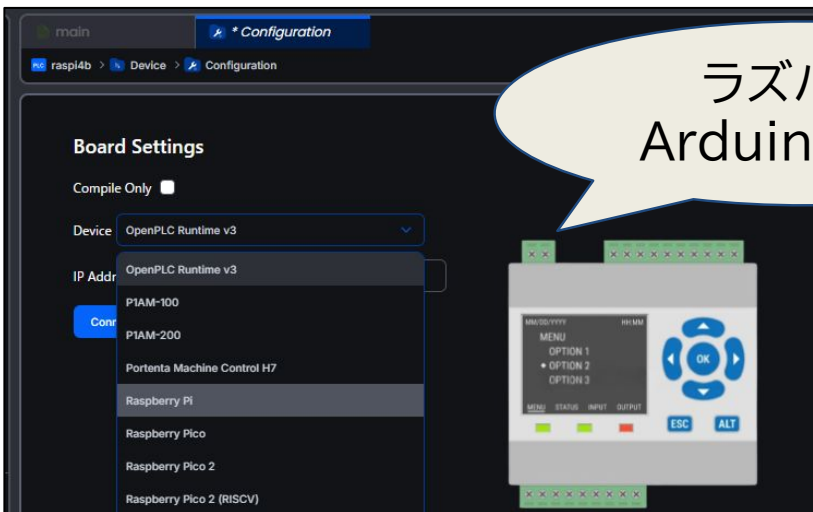
\*工場・FA業界で使われるコントローラ



# ラズパイでもPLCが動く

趣味でも十分にデバッグ+遊べる、ソフトウェアPLC

ラズパイ、  
ArduinoもOK



ラズパイ含めた色々なボードが使用可能。→ 実際にデバッグもできる

# デモ動画(通常のOpenPLC)

[https://youtu.be/51srWoKP\\_Sg](https://youtu.be/51srWoKP_Sg)

The screenshot displays the OpenPLC software interface. On the left is a navigation tree with categories like Functions, Programs, Data Types, Resource, Device, Configuration, Servers, and Remote Devices. The main workspace shows a ladder logic program with two rungs. The first rung contains a normally open contact X001, a normally closed contact X002, and a coil M01. The second rung contains a normally open contact M01 and a coil Y01. Below the ladder logic is a console window with the following log messages:

```
[20-02-26 23:02:25]: Debug compilation completed successfully.
[20-02-26 23:02:25]:
[20-02-26 23:02:25]: Extracting MD5 from compiled program...
[20-02-26 23:02:25]: Program MD5: 722f82aa72bd63063e3bb73790e04d1d
[20-02-26 23:02:25]: R... MD5 from target... 1,10.
[20-02-26 23:02:25]: M...
[20-02-26 23:02:25]: D...
[20-02-26 23:02:25]: Debugger started successfully. Found 0 debug variables.
```

On the right side of the interface, there is a photograph of a Raspberry Pi 4B board connected to a breadboard. The breadboard contains a red LED, a resistor, and a small black component. Various colored cables are plugged into the Pi's ports.

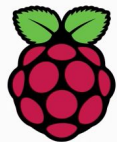
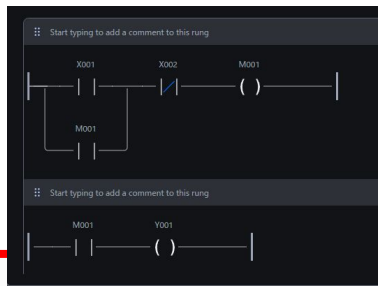
**OpenPLC Runtime v3 + Raspberry Pi 4B**

# Linux上でのOpenPLC

# Linux + OpenPLC

Raspbian(Linux)上で、簡単にOpenPLCはテスト可能

ラズパイでは  
Raspbian  
(Linux)で動く



Raspberry Pi OS



デバッグモード・ネットワーク(MQTT/OPCUA)など色々機能あり

オープンソースで中身は分かる \*ライセンスはGPL

thiagorvalves / OpenPLC\_v3

Code Issues 43 Pull requests 13 Actions Projects Wiki Security Insights

OpenPLC\_v3 Public Watch 87 Fork 567 Star 1.5k

master Go to file Code About

thiagorvalves Fix compilation error with ARRAY inside FB bb35f69 · last month

· .vscode	Update MatIEC version	2 years ago
· documentation	Updated in accord to the new setti...	11 months ago
· utils	Include Python loader to execute P...	5 months ago
· webserver	Fix compilation error with ARRAY i...	last month

OpenPLC Runtime version 3

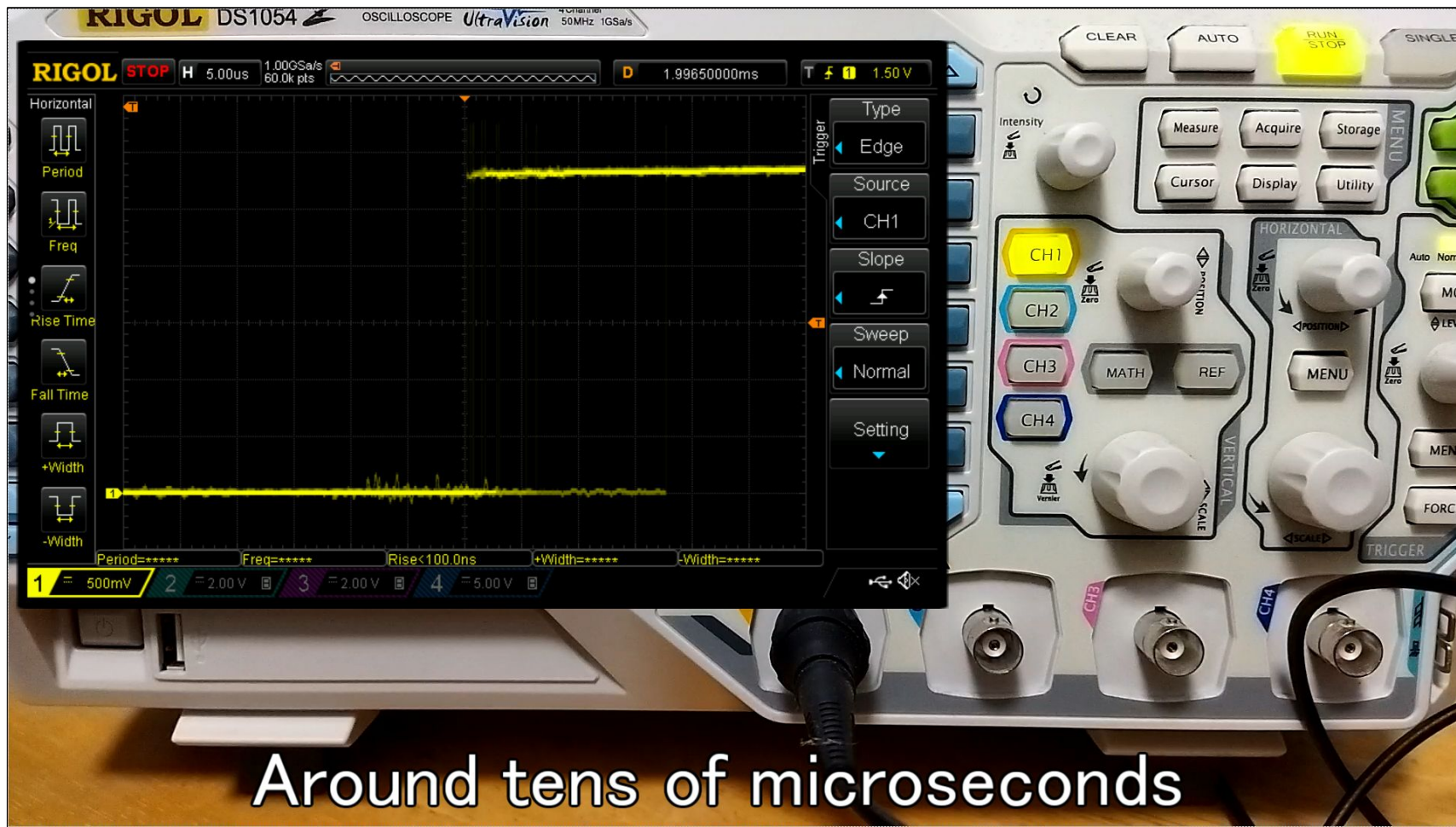
- Readme
- GPL-3.0 license
- Activity
- 1.5k stars
- 87 watching
- 567 forks

Report repository

[https://github.com/thiagorvalves/OpenPLC\\_v3](https://github.com/thiagorvalves/OpenPLC_v3)

# Linux + OpenPLC のジッタのデモ

<https://youtu.be/3xPex-61GqQ>



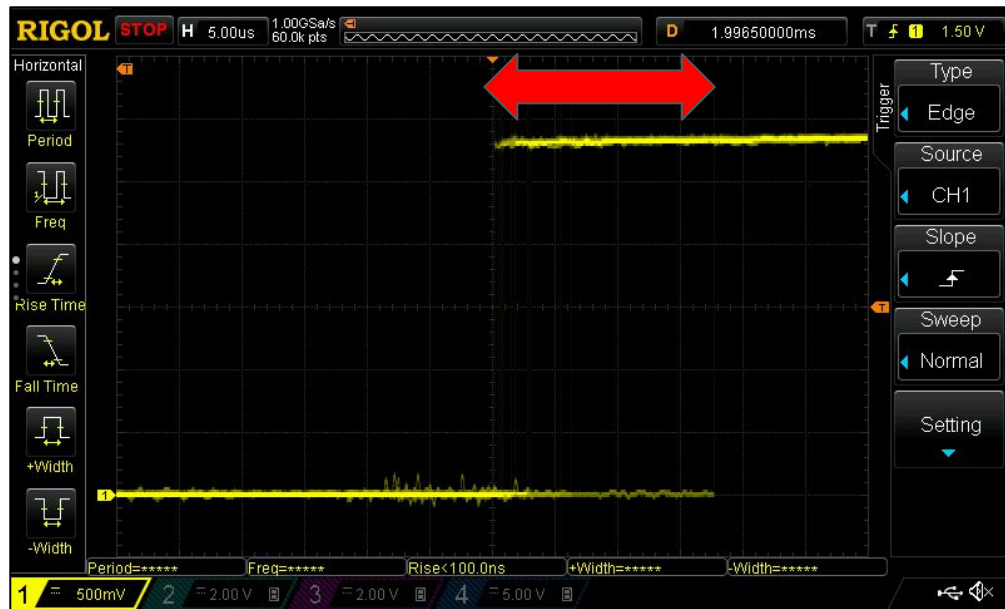
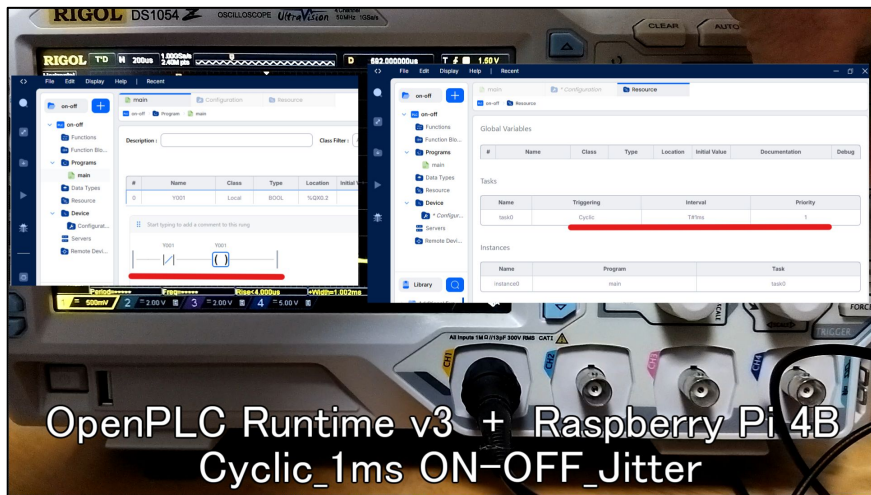
Around tens of microseconds

# 便利だけど…ちょっと遅い。

標準だとms以下の制御周期が設定できない + ジッタも大きい

最短1ms周期

数十usのジッタ

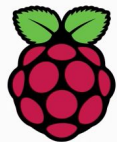
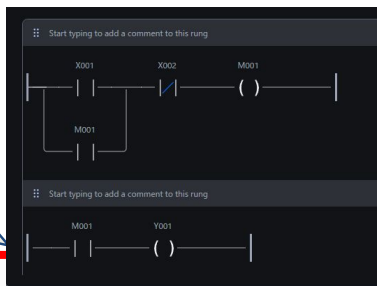


LinuxをRTOSにしたら、  
もっと高速になるのでは？

# 高速周期+低ジッタへ

大体考えることはみんな同じ。Real-Timeのパッチ or OS。

デフォ周期20ms  
±  
数十usジッタ



Raspberry Pi OS



案①:Linuxカーネルに「PREEMPT\_RT」の  
Real-Timeのパッチを入れる  
→ただし、他処理も色々改善いれないとダメそう

\*OpenPLC公式の掲示版でも色々ネタが上がっていた  
<https://openplc.discussion.community/post/openplc-hard-real-time-os-10244581>

案②:ごっそりRTOSに変える + 機能を絞る

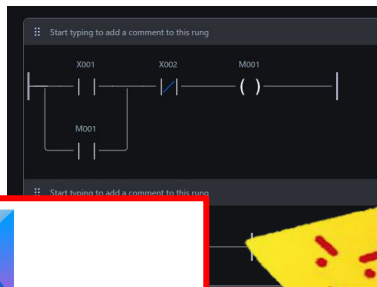


RTOS:  
Real-Time Operating System

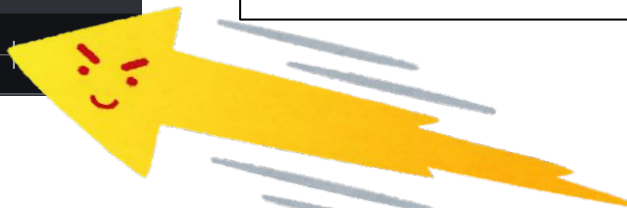
# Zephyr(RTOS)で実装

機能は最低限で良いので、高速なRTOS環境を作って遊ぶ

Zephyr(RTOS)で  
PLCを動かす



ms(ミリ)ゆっくり周期 + usレベルのジッタ  
↓  
us(マイクロ)の高速周期 + nsレベルのジッタ



案②: ごっそりRTOSに変える + 機能を絞る



Zephyr®

RTOS:  
Real-Time Operating  
System



# Zephyr(RTOS)に 実装した結果

# Zephyr(RTOS)+OpenPLC

Geminiと相談しながら、ポーティング+移植 →ビルドも通った

## ①Overlay(デバイスツリー)の追加

- ・ラズパイ4B
- ・ラズパイPico2W
- ・Nordic\_nRF54L15-DK

## ②OpenPLCのライブラリの移植

- ・PLC規格のIEC61131-3の  
最低限だけ修正+移植

## ③main.cファイルの作成

入出力見て→ラダーのロジック実行  
→ループ繰り返し

## ④prj.conf・Cmakelistの作成

→Zephyrのビルドが通るように

```
File Explorer: MY-OPENPLC
├── boards
│   ├── nrf54115dk_nrf54115_cpuapp.overlay
│   ├── rpi_4b.overlay
│   └── rpi_pico2_rp2350a_m33_w.overlay
├── build
├── build_1
├── src
│   ├── core
│   │   └── lib
│   │       ├── accessor.h
│   │       ├── iec_std_FB.h
│   │       ├── iec_std_functions.h
│   │       ├── iec_std_lib.h
│   │       ├── iec_types_all.h
│   │       └── iec_types.h
│   ├── c_blocks.h
│   ├── communication.h
│   ├── Config0.c
│   ├── Config0.h
│   ├── glueVars.cpp
│   ├── ladder.h
│   ├── LOCATED_VARIABLES.h
│   ├── POUS.c
│   ├── POUS.h
│   ├── Res0.c
│   ├── utils.cpp
│   └── VARIABLES.csv
├── main.c
├── CMakeLists.txt
└── prj.conf
README.md

Terminal Output:
src > C main.c > config_run_(unsigned long)
1 /* Zephyr main.c template */
2 #include <zephyr/kernel.h>
3 #include <zephyr/drivers/gpio.h>
4 #include <zephyr/sys/printk.h>
5 #include "iec_std_lib.h" // Include path to lib folder
6
7 // Functions defined in Config0.c
8 extern void config_init_(void);
9 extern void config_run_(unsigned long tick);
10 extern unsigned long long common_ticktime_; // Cycle time defined in Config0.c (ns)
11 extern void glueVars(void);
12 extern void updateTime(void);
13
14 // Variables defined in LOCATED_VARIABLES.h (extern declaration if necessary)
15 // Mapping based on POUS.c definition:
16
17 -- The ASM compiler identification is GNU
18 -- Found assembler: C:/Users/ioten/zephyr-sdk-0.17.4/arm-zephyr-eabi/bin/arm-zephyr-eabi-gcc.exe
19 -- Found gen_kobject_list: C:/Users/ioten/zephyrproject/zephyr/scripts/build/gen_kobject_list.py
20 -- Configuring done (0.9s)
21 -- Generating done (0.6s)
22 -- Build files have been written to: C:/Users/ioten/zephyrproject/test/my-openplc/build
23 -- west build: building application
24 [1/209] Generating include/generated/zephyr/version.h
25 -- Zephyr version: 4.3.99 (C:/Users/ioten/zephyrproject/zephyr), build: v4.3.0-3034-g2062a12ed267
26 [209/209] Linking OX executable zephyr/zephyr.elf
Memory region      Used Size  Region Size  %age Used
FLASH:             35836 B    4 MB         0.85%
RAM:               127264 B   520 KB       23.90%
IDT_LIST:          0 GB          32 KB        0.00%
Generating files from C:/Users/ioten/zephyrproject/test/my-openplc/build/zephyr/zephyr.elf for board: rpi_pico2
Converted to uf2, output size: 71680, start address: 0x10000000
Wrote 71680 bytes to zephyr.uf2

(.env) C:\Users\ioten\zephyrproject\test\my-openplc\west build -p -b rpi_pico2/rp2350a/m33/w
-- west build: making build dir C:\Users\ioten\zephyrproject\test\my-openplc\build pristine
-- west build: generating a build system
```

# GitHubのリンク先

さくっと趣味で作ったPre-test版だから、参考までに

The screenshot shows a GitHub repository page for 'zephyr-openplc-pretest' by user 'iotengineer22'. The repository is public and has 1 branch and 0 tags. The main branch is selected. The repository description states: 'This is a pre-test version of OpenPLC adapted from the OpenPLC v3 program to run on Zephyr RTOS.' The repository contains several files and folders: 'boards', 'src', 'CMakeLists.txt', 'LICENSE', 'README.md', and 'prj.conf'. The README file is selected and shows the title 'Zephyr RTOS + OpenPLC (Pre-test Version)' and the description: 'This is a pre-test version of OpenPLC adapted from the OpenPLC v3 program to run on Zephyr RTOS. Original OpenPLC v3: [https://github.com/thiagorvalves/OpenPLC\\_v3](https://github.com/thiagorvalves/OpenPLC_v3)'.

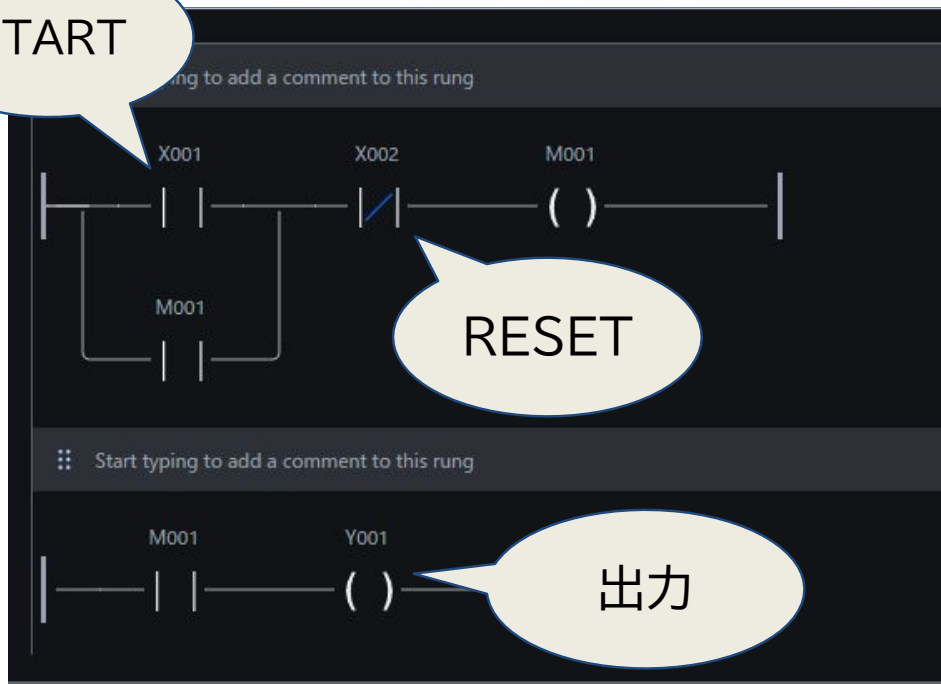
<https://github.com/iotengineer22/zephyr-openplc-pretest>

\*OpenPLCのライブラリを一部移植+修正しているので、ライセンスはGPL。

# ラダーの自己保持回路のテスト(1)

冒頭のデモでも紹介した、基本のラダー回路

START



RESET

出力

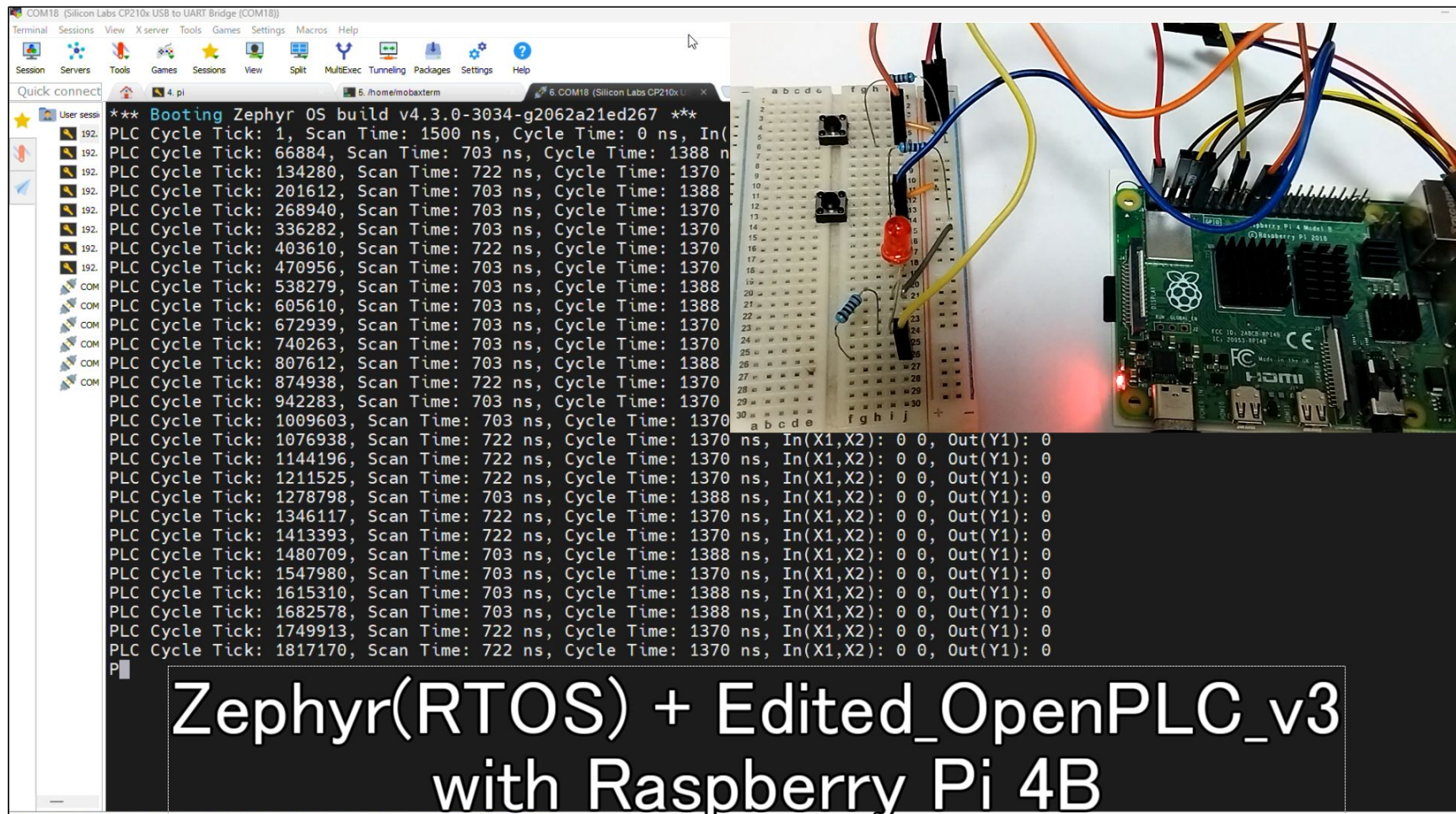
ラダーロジックによるシンプルな自己保持回路のデモ。

- \* \*\*X001\*\* : Startボタン(A接点)
- \* \*\*X002\*\* : RESETボタン(B接点)
- \* \*\*M001\*\* : 内部補助リレー
- \* \*\*Y001\*\* : 出力LED

\*ラダーから.CへのコンパイラはOpenPLCのものを流用

# デモ動画(Zephyr + OpenPLC + 4B)

<https://youtu.be/QVRoRsNYECE>



The image displays a terminal window on the left and a photograph of a Raspberry Pi 4B board on the right. The terminal window shows the following output:

```
COM18 (Silicon Labs CP210x USB to UART Bridge (COM18))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect
User sessi
*** Booting Zephyr OS build v4.3.0-3034-g2062a21ed267 ***
192. PLC Cycle Tick: 1, Scan Time: 1500 ns, Cycle Time: 0 ns, In(
192. PLC Cycle Tick: 66884, Scan Time: 703 ns, Cycle Time: 1388 n
192. PLC Cycle Tick: 134280, Scan Time: 722 ns, Cycle Time: 1370
192. PLC Cycle Tick: 201612, Scan Time: 703 ns, Cycle Time: 1388
192. PLC Cycle Tick: 268940, Scan Time: 703 ns, Cycle Time: 1370
192. PLC Cycle Tick: 336282, Scan Time: 703 ns, Cycle Time: 1370
192. PLC Cycle Tick: 403610, Scan Time: 722 ns, Cycle Time: 1370
192. PLC Cycle Tick: 470956, Scan Time: 703 ns, Cycle Time: 1370
COM PLC Cycle Tick: 538279, Scan Time: 703 ns, Cycle Time: 1388
COM PLC Cycle Tick: 605610, Scan Time: 703 ns, Cycle Time: 1388
COM PLC Cycle Tick: 672939, Scan Time: 703 ns, Cycle Time: 1370
COM PLC Cycle Tick: 740263, Scan Time: 703 ns, Cycle Time: 1370
COM PLC Cycle Tick: 807612, Scan Time: 703 ns, Cycle Time: 1388
COM PLC Cycle Tick: 874938, Scan Time: 722 ns, Cycle Time: 1370
PLC Cycle Tick: 942283, Scan Time: 703 ns, Cycle Time: 1370
PLC Cycle Tick: 1009603, Scan Time: 703 ns, Cycle Time: 1370
PLC Cycle Tick: 1076938, Scan Time: 722 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1144196, Scan Time: 722 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1211525, Scan Time: 722 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1278798, Scan Time: 703 ns, Cycle Time: 1388 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1346117, Scan Time: 722 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1413393, Scan Time: 722 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1480709, Scan Time: 703 ns, Cycle Time: 1388 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1547980, Scan Time: 703 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1615310, Scan Time: 703 ns, Cycle Time: 1388 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1682578, Scan Time: 703 ns, Cycle Time: 1388 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1749913, Scan Time: 722 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
PLC Cycle Tick: 1817170, Scan Time: 722 ns, Cycle Time: 1370 ns, In(X1,X2): 0 0, Out(Y1): 0
P
```

The photograph on the right shows a Raspberry Pi 4B board connected to a breadboard. The breadboard contains several components, including a red LED, a blue potentiometer, and various resistors. The Raspberry Pi board is connected to the breadboard via jumper wires. The board's red power LED is illuminated.

Zephyr(RTOS) + Edited\_OpenPLC\_v3  
with Raspberry Pi 4B



# ラダーの最短ON/OFF回路のテスト(2)

ジッタ測定で紹介した、基本のラダーの最短ON/OFF回路

#	Name	Class	Type
0	Y001	Local	BOOL

Start typing to add a comment to this rung

```
graph LR; Y001_IN[Y001] --- Y001_OUT((Y001)); Y001_OUT --- Y001_IN;
```

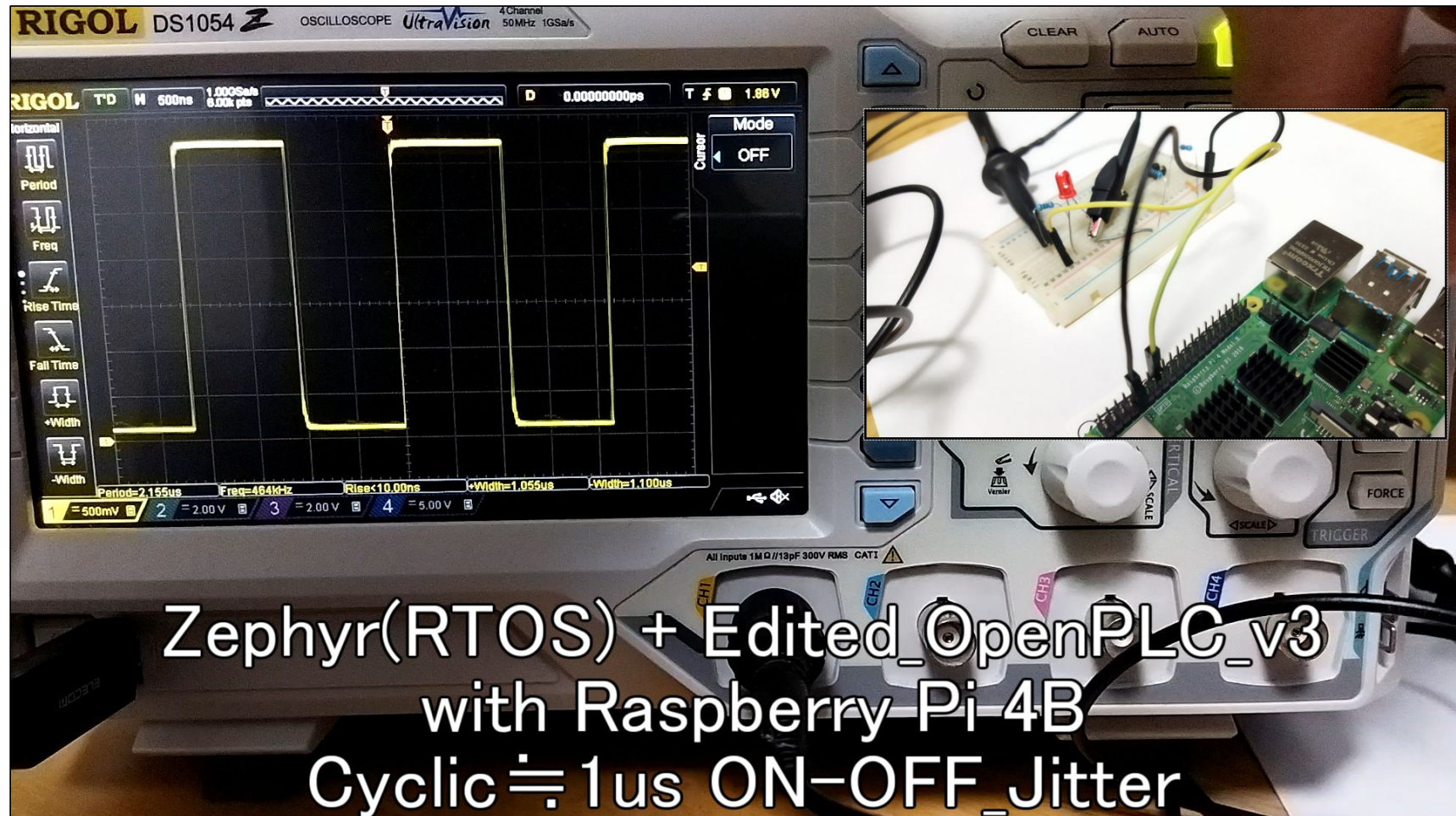
ラダーロジックによるシンプルな最短ON/OFFのデモ。

\* \*\*Y001\*\* : 出力LED

\*ラダー→.Cへのコンパイラは  
OpenPLCのものを流用

前回値を反転出力  
→最短ON/OFFループ

# デモ動画(Zephyr + OpenPLC のジッタ)<https://youtu.be/xayHDxoTsn0>

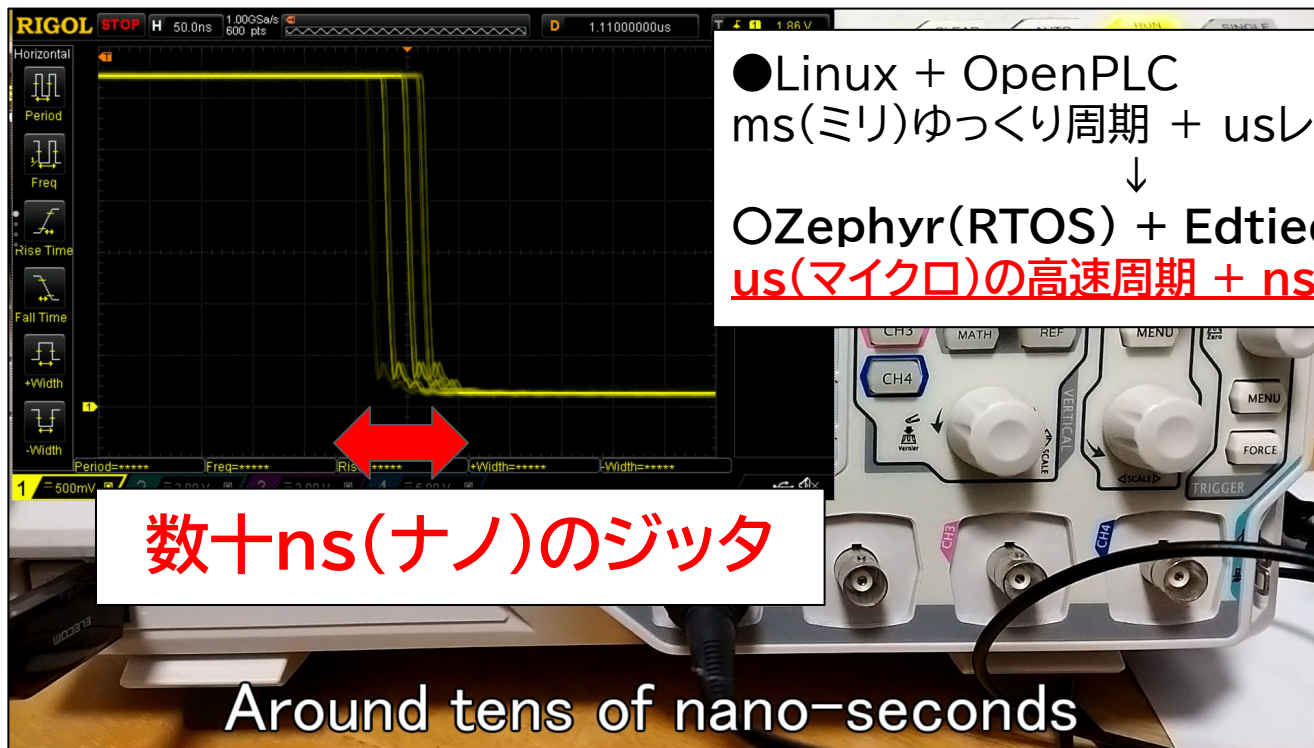


The image shows a RIGOL DS1054 oscilloscope displaying a square wave signal. The waveform exhibits significant jitter, with the rising and falling edges appearing irregular. The oscilloscope's interface shows various settings: a horizontal scale of 500ns, a vertical scale of 500mV, and a frequency of 464kHz. The signal is measured on channel 2 (CH2) with a 2.00V scale. An inset image in the upper right corner shows a Raspberry Pi 4B board connected to a breadboard circuit, which is likely the source of the signal being measured. The breadboard contains several integrated circuits and resistors, with jumper wires connecting them to the Pi's pins.

Zephyr(RTOS) + Edited\_OpenPLC\_v3  
with Raspberry Pi 4B  
Cyclic  $\approx$  1us ON-OFF\_Jitter

# Zephyr(RTOS)で低ジッタ

ラズパイ4Bで狙い通り、ns(ナノ)の低ジッタ →OK



●Linux + OpenPLC  
ms(ミリ)ゆっくり周期 + usレベルのジッタ

↓  
○Zephyr(RTOS) + Edtied\_OpenPLC  
us(マイクロ)の高速周期 + nsレベルのジッタ

数十ns(ナノ)のジッタ

Around tens of nano-seconds

# Zephyr(RTOS)に 実装するメリット

# Zephyrの良いところ

ハード抽象化が強力 → Zephyr対応のCPUなら簡単に横展できる

\*オープニングで紹介した、RTOSの比較表

CPU(マイコン)変えた場合

- レジスタ調査
- コードを書き直し
- 開発環境を再インストール

…という苦行から解放

凄くメリット  
大きい



OS無しもいいけど...

Zephyr(RTOS)使えば  
多くのサポートが有り

	Zephyr RTOS	FreeRTOS	Azure RTOS (Thre
ライセンス	Apache 2.0	MIT	MIT
設計思想	フルスタック型	ミニマルなカーネル	高性能・高信頼性
管理団体	<u>Linux Foundation</u>	Amazon (AWS)	Microsoft (Azure)
ハード抽象化	<u>強力 (DeviceTree)</u>	ベンダー依存	中程度

# 今回のメリット例

メインプログラム(main.c)を変えずに、Pico2W/nRF54L15に横展

Overlay(デバイスツリー)  
少し調整しただけ

ARMコア/メーカー違う  
4つのボードを  
全てが同じmain.c



ラズパイ4B\_A72



Pico2W\_M33



nRF54L15-DK\_M33



XIAO nRF54L15 Sense M33

# HAL (Hardware Abstraction Layer)

OSがハードウェアの違いを、抽象化(吸収)してくれる

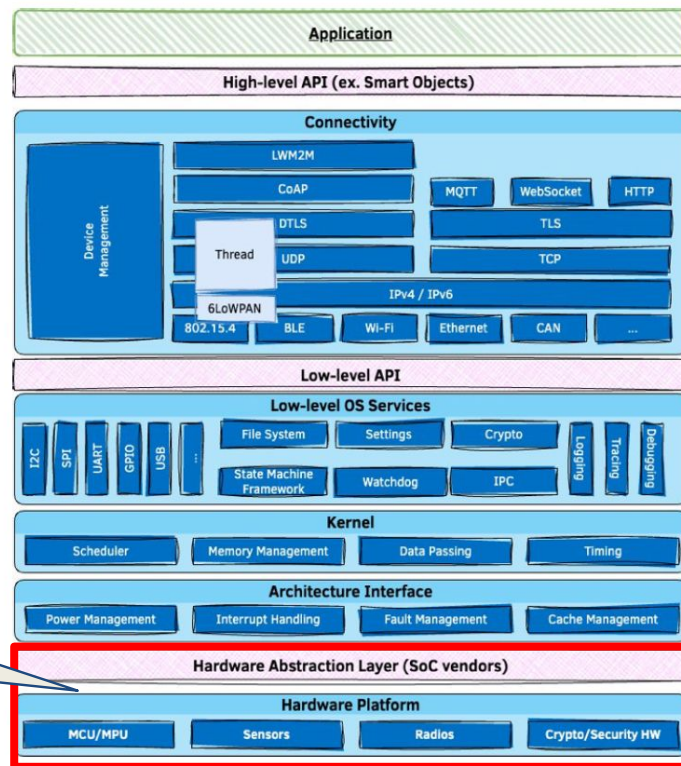
- ・ユーザはどのICでも標準的なドライバ・APIを使える・調整すればOK



各IC/メーカーの違いを  
OSが抽象化(吸収)

- ・Zephyr公式の資料

<https://www.zephyrproject.org/zephyr-overview/>



# ビルドも同じ環境

同じZephyr環境で、ボード名変えてビルドしただけ →OK

ラズパイ\_4B:

west build -p -b **rpi\_4b**

ラズパイ\_Pico2W:

west build -p -b **rpi\_pico2/rp2350a/m33/w**

Nordic\_nRF54L15-DK:

west build -p -b **nrf54l15dk/nrf54l15/cpuapp**

Seed\_XIAO nRF54L15:

west build -p -b **xiao\_nrf54l15/nrf54l15/cpuapp**



<https://www.youtube.com/watch?v=r-i1I2-w9lw>

<https://youtu.be/-4zIHnI9Joc>



まとめ

# まとめ

## Zephyr(RTOS)で OpenPLCを実装+遊べた！



•Linux→RTOSを使うことで、高速周期+低ジッタへ  
(オープンソースで中身も分かり、良い勉強になりました)

•Zephyr(RTOS)の抽象化の良いところ紹介できた  
(どのメーカー、どのICでも実装・横展会が可能です)

•OpenPLCのライセンスがGPLなので、注意が必要。  
(教育や勉強用にはちょうど良いのですが...)