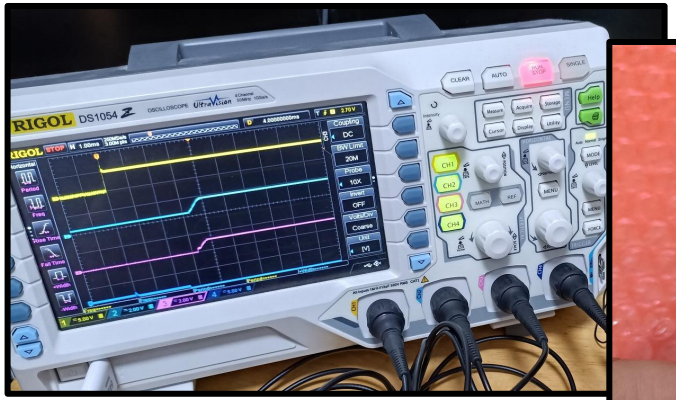
Zephyr(RTOS)でOpenPLCを実装してみた

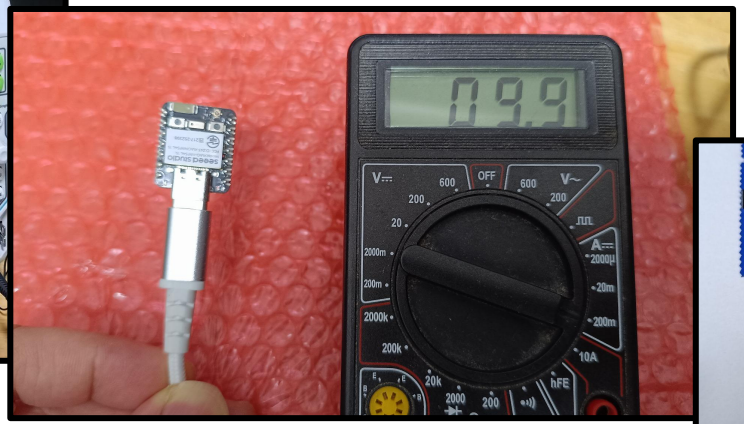
ミソジ
2026/3/27
Zephyr Project meetup: Nagoya, Japan
#ZephyrRTOS

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
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

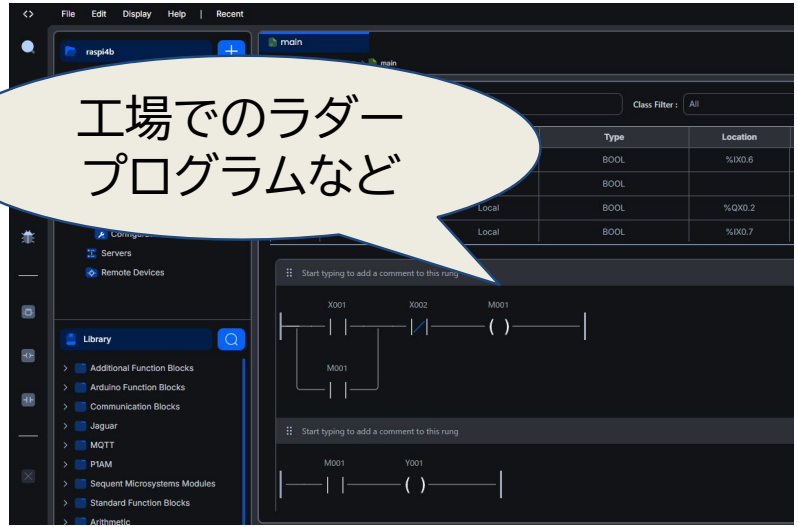
- OpenPLCとは？
- LinuxをRTOSにしたら、もっと高速になるのでは？
- Zephyr(RTOS)に実装した結果・メリット
- まとめ

OpenPLCとは？

OpenPLC

OpenPLC･･･オープンソースの「ソフトウェアPLC」の一つ

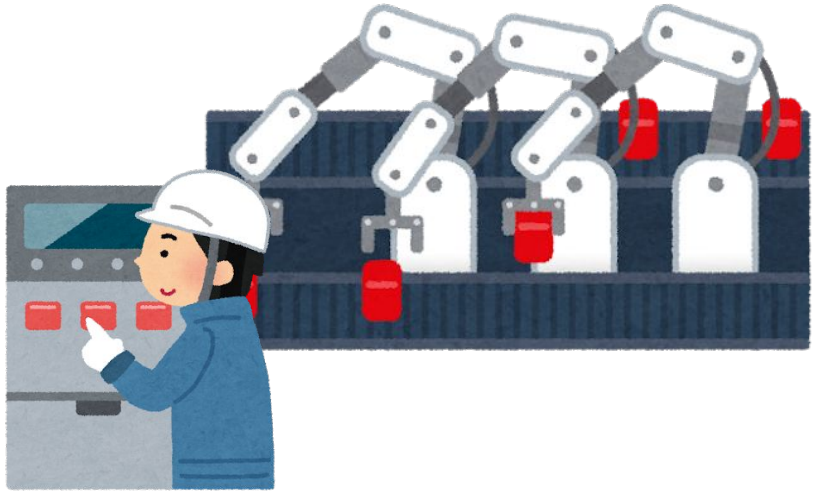
OpenPLC Editor/Runtime



工場でのラダー
プログラムなど

https://autonomylogic.com/

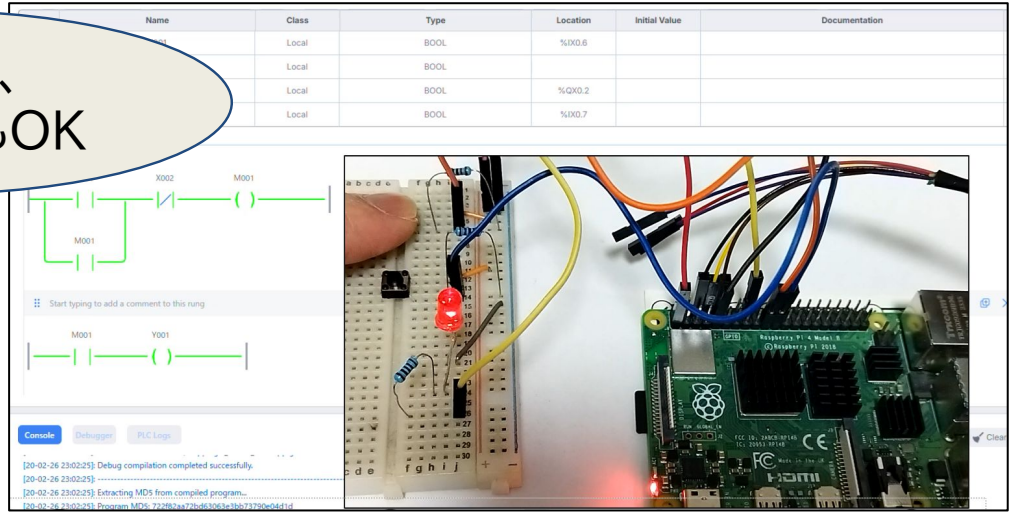
PLC･･･ Programmable Logic Controller
*工場・FA業界で使われるコントローラ



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

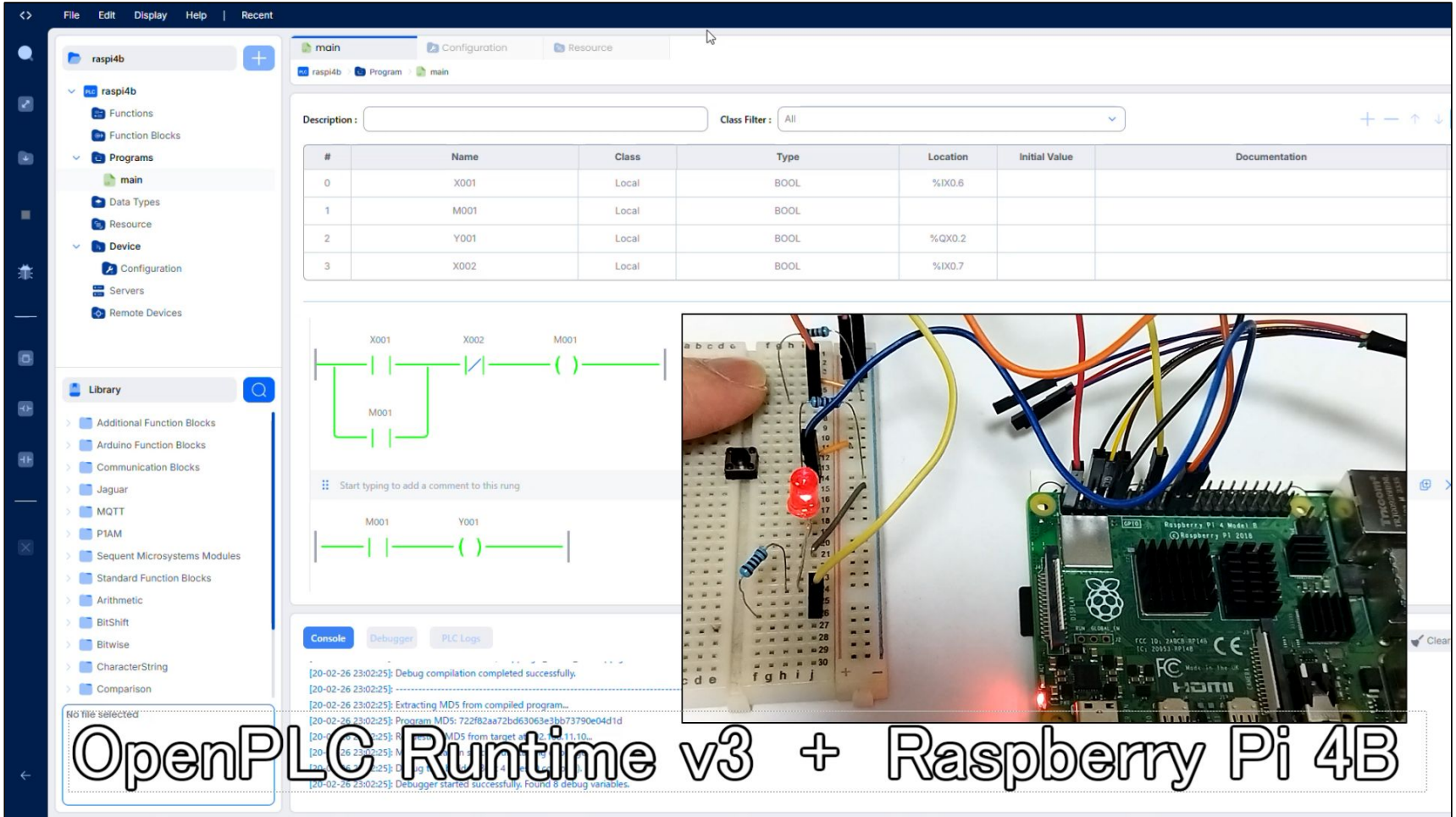
ラズパイ、
ArduinoもOK



ラズパイ含めた色ん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

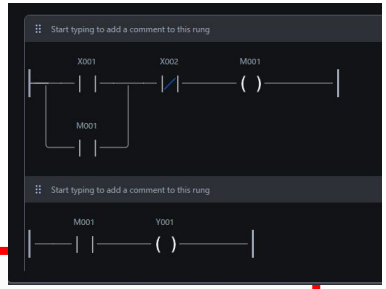


Linux上でのOpenPLC

Linux + OpenPLC

Raspbian(Linux)上で、簡単にOpenPLCはテスト可能

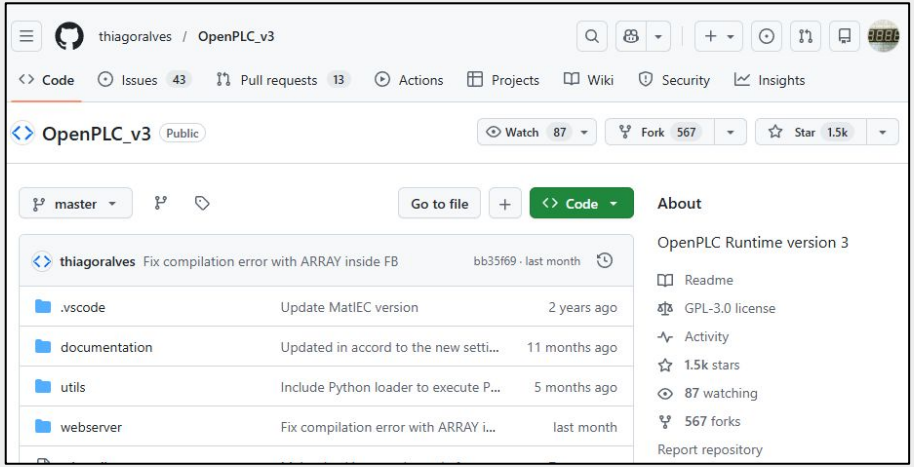
ラズパイでは
Raspbian
(Linux)で動く



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

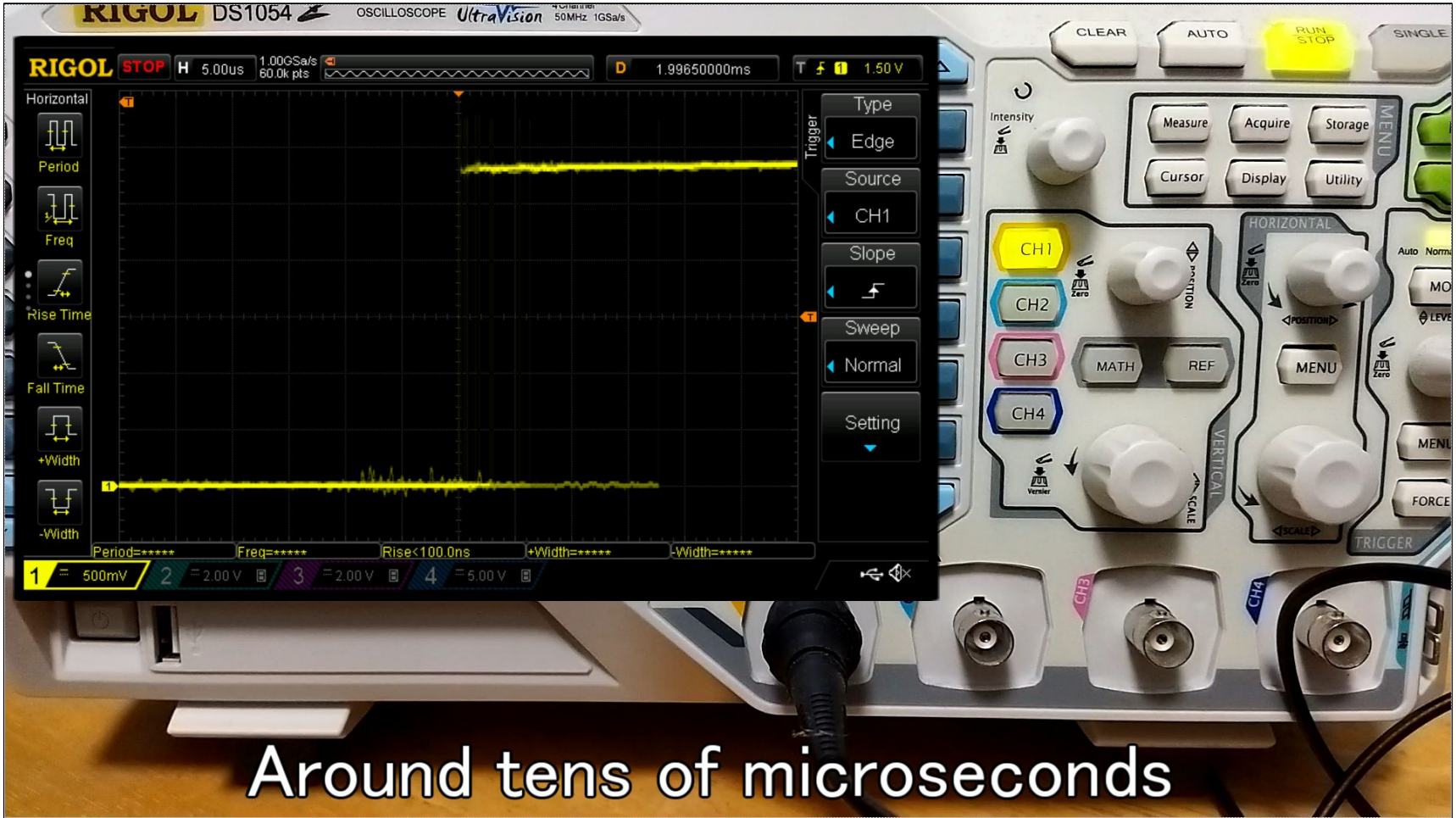
オープンソースで中身は分かる　*ライセンスはGPL



https://github.com/thiagoralves/OpenPLC_v3

Linux + OpenPLC のジッタのデモ

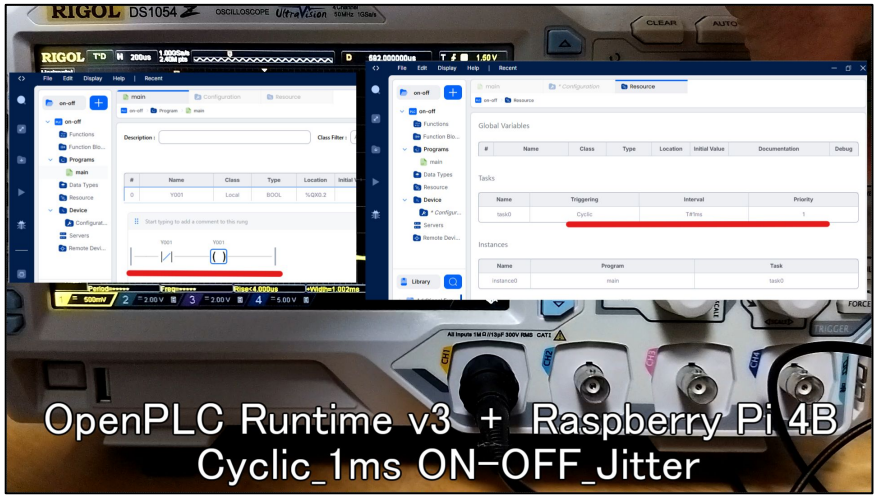
https://youtu.be/3xPex-61GqQ



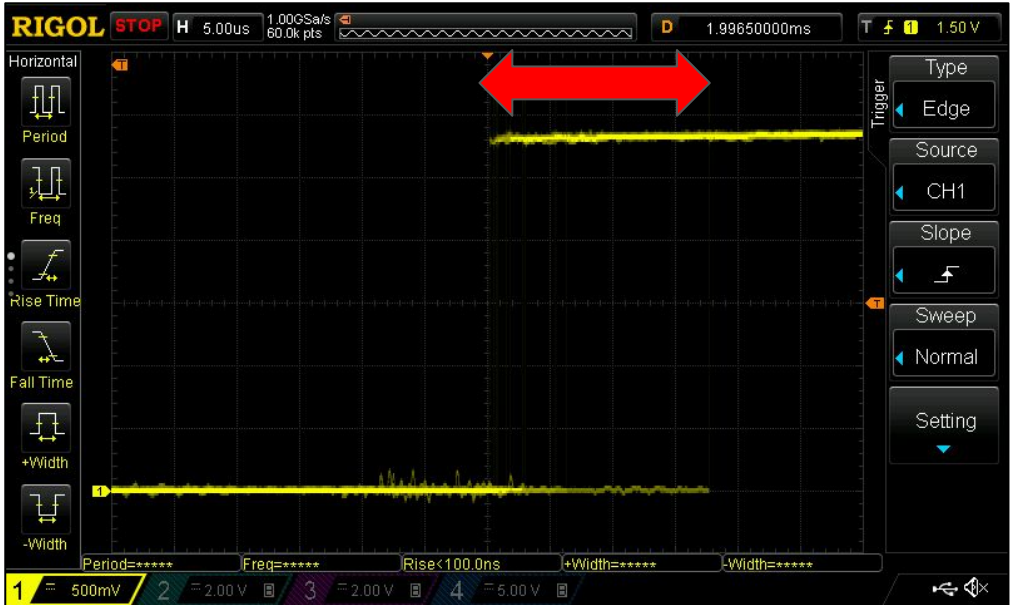
便利だけど…ちょっと遅い。

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

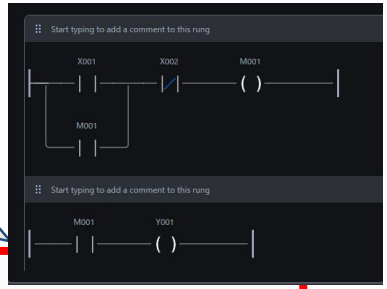


LinuxをRTOSにしたら、もっと高速になるのでは？

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

Zephyr®

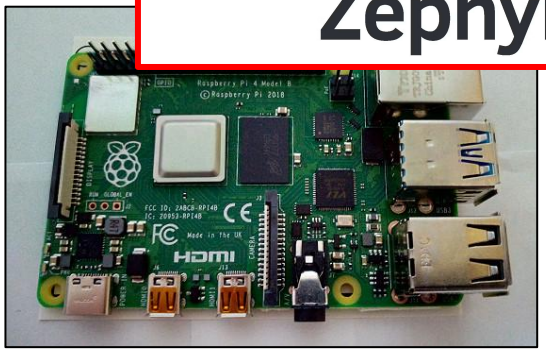
RTOS:
Real-Time Operating System

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

Zephyr®



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

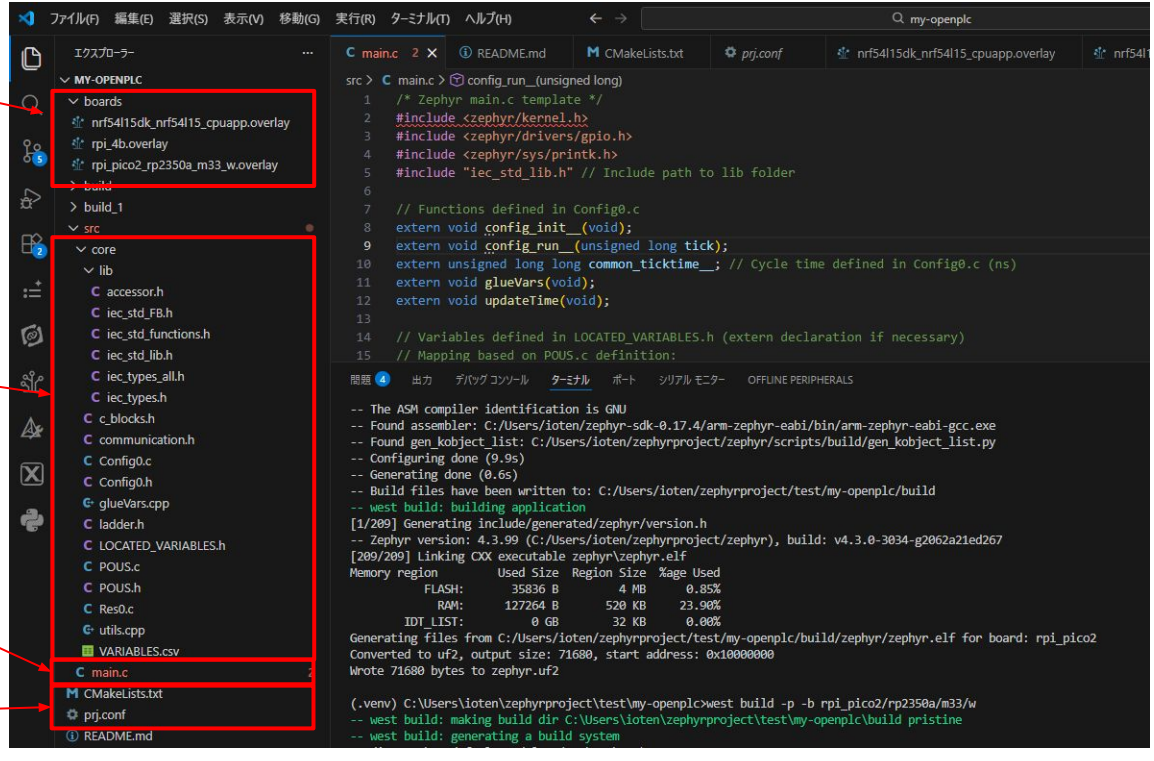
Geminiと相談しながら、ポーティング+移植　→ビルドも通った

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

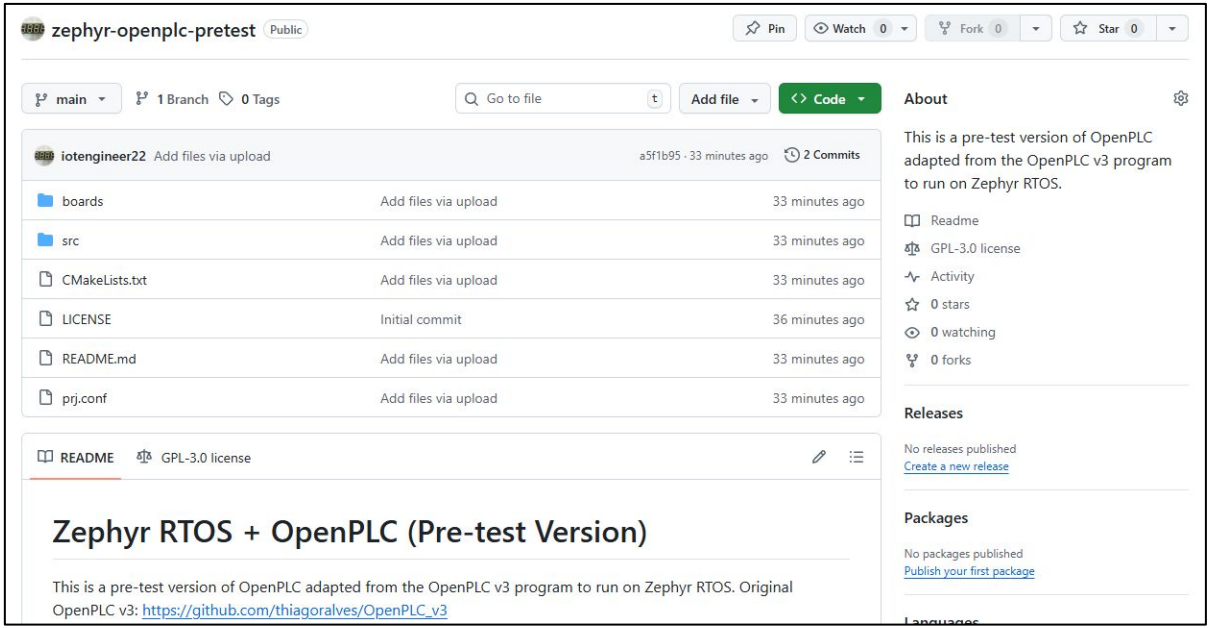
③main.cファイルの作成
入出力見て→ラダーのロジック実行
→ループ繰り返し

④prj.conf・Cmakelistの作成
→Zephyrのビルドが通るように



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

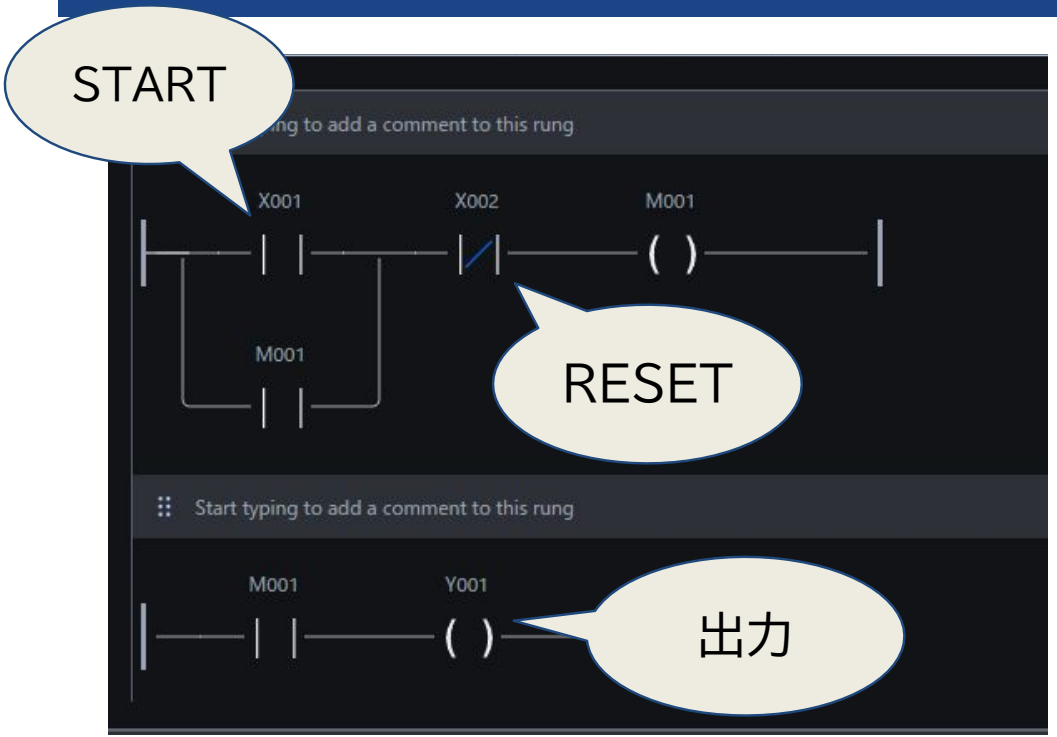


https://github.com/iotengineer22/zephyr-openplc-pretest

*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



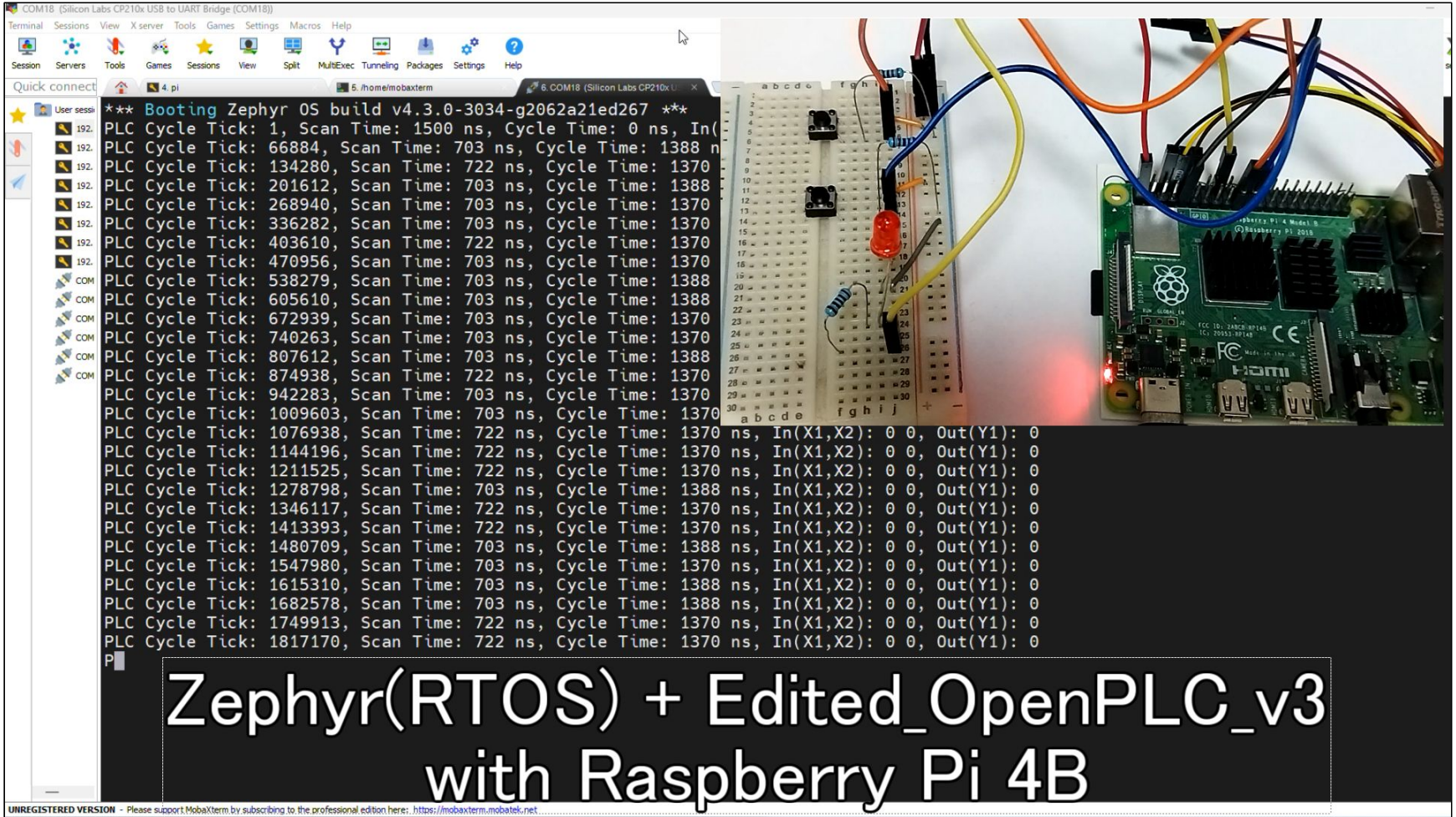
ラダーロジックによるシンプルな
自己保持回路のデモ。

* **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

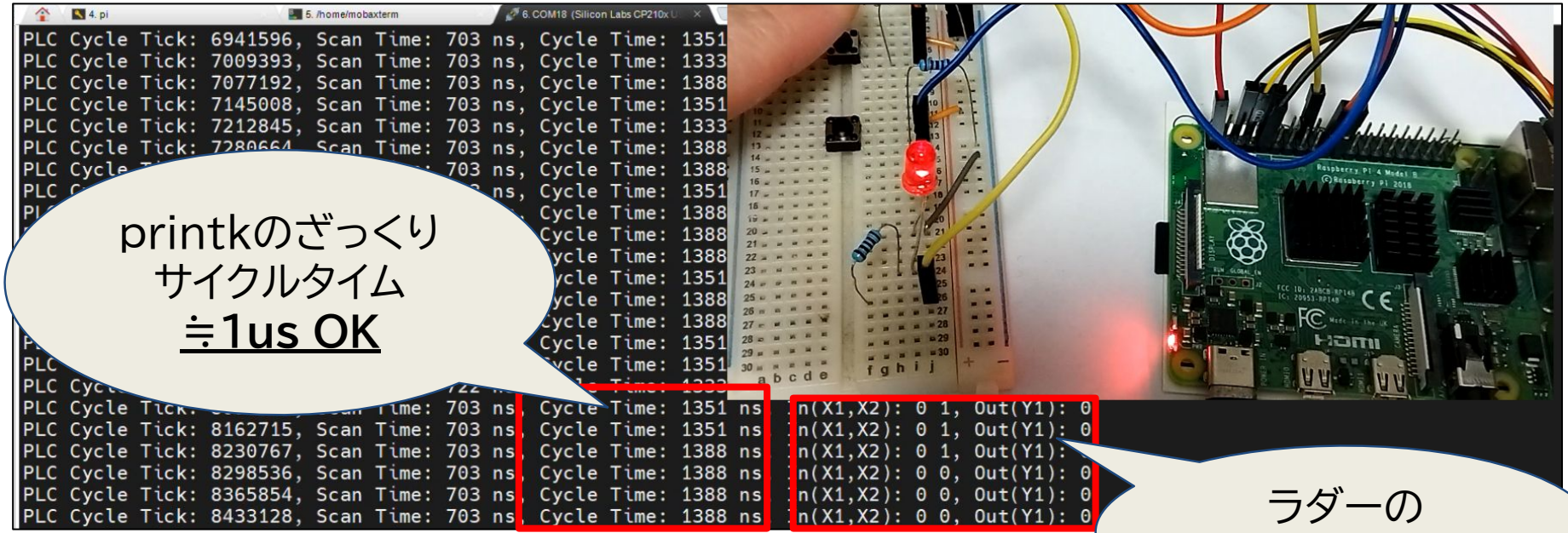
デモ動画(Zephyr + OpenPLC + 4B)

https://youtu.be/QVRoRsNYECE



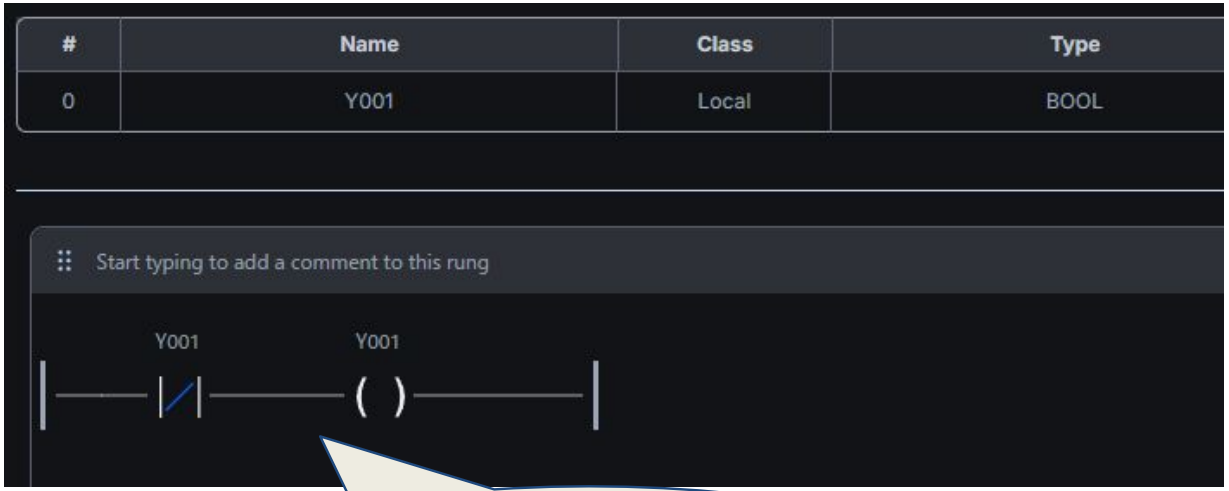
Zephyr(RTOS)でOpenPLC動く

ラズパイ4Bで狙い通り、us(マイクロ)の高速周期　→OK



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



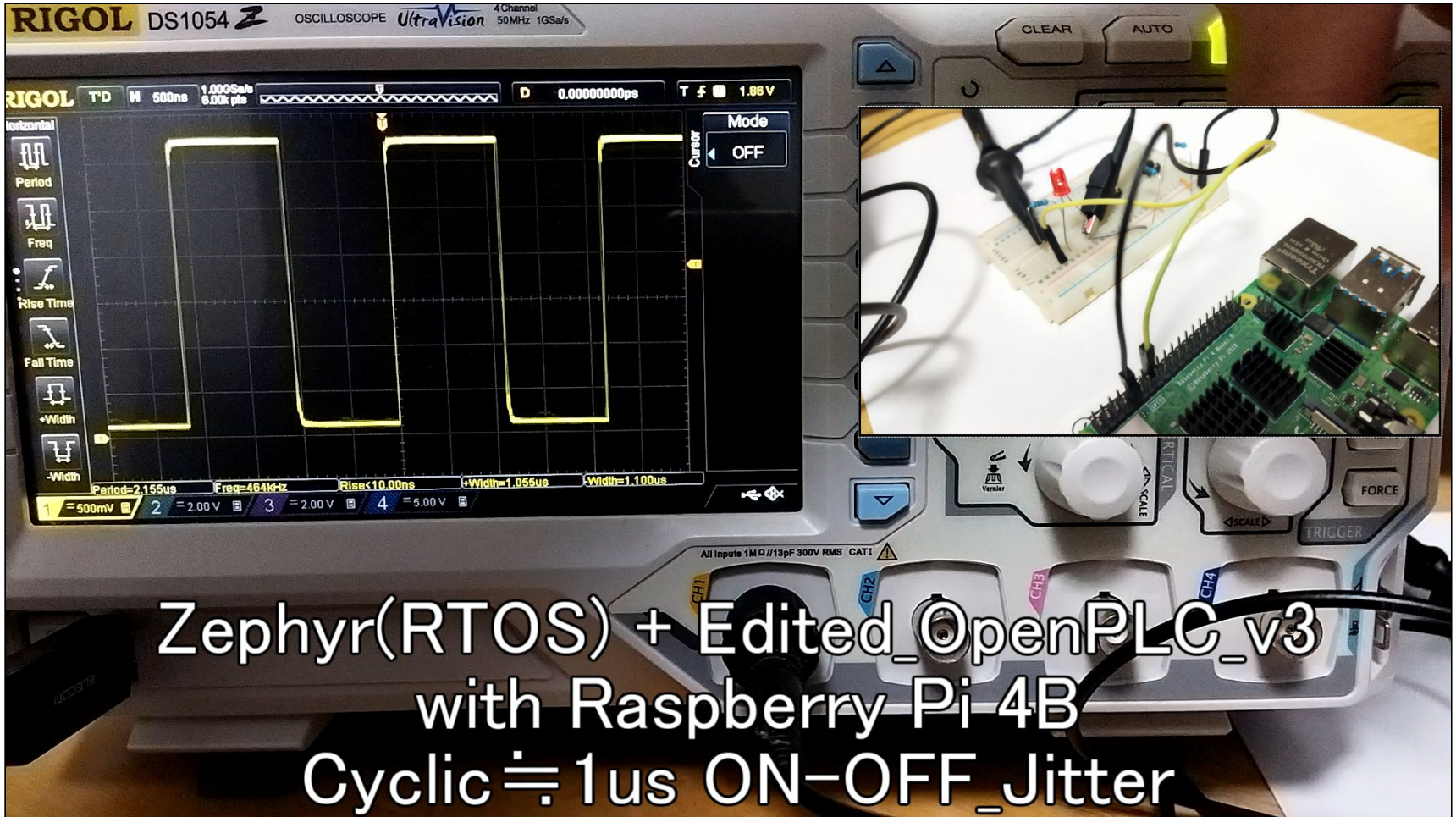
前回値を反転出力
→最短ON/OFFループ

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

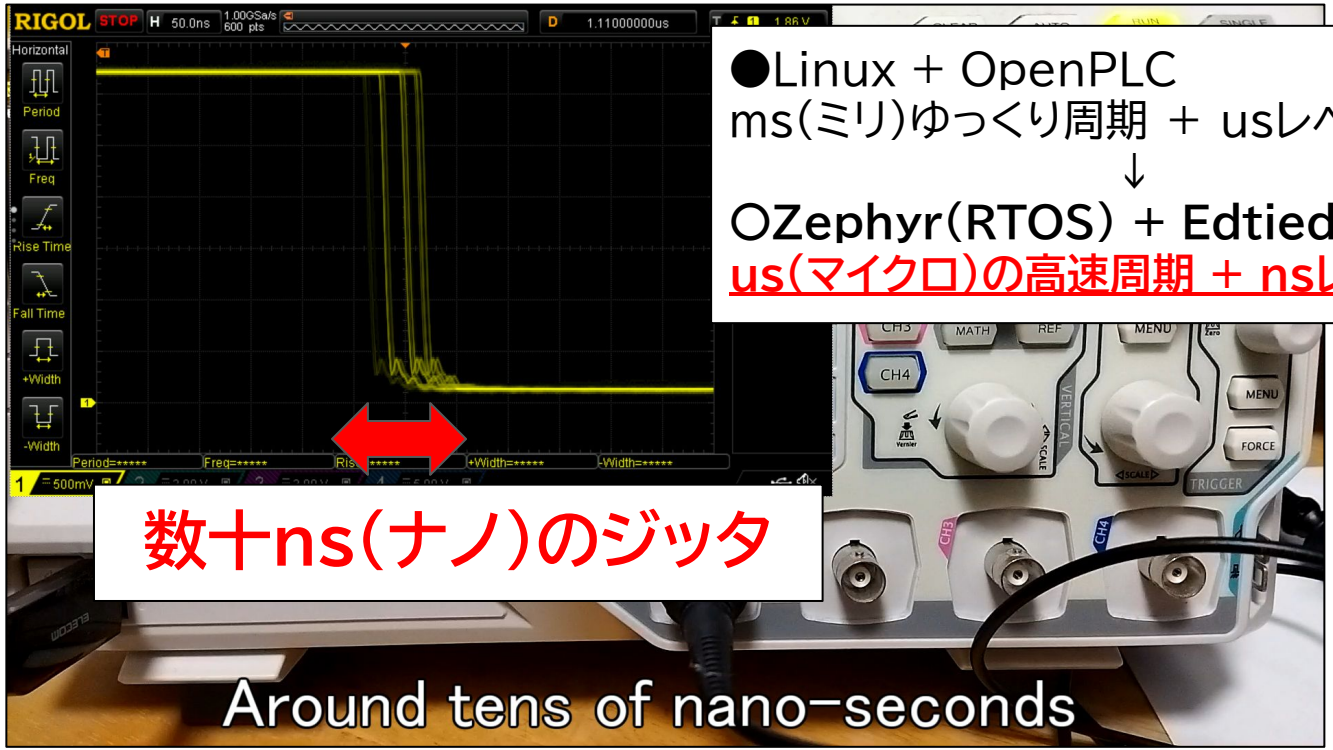
*ラダー→.Cへのコンパイラは
　OpenPLCのものを流用

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



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

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

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

*オープニングで紹介した、RTOSの比較表

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 (Threa
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	Linux Foundation	Amazon (AWS)	Microsoft (Azure)
ハード抽象化	強力 (DeviceTree)	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

エクスプローラー

MY-OPENPLC

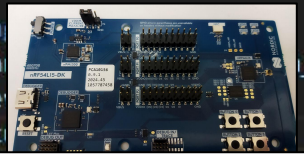
- boards
  - nrf54l15dk_nrf54l15_cpuapp.overlay
  - rpi_4b.overlay
  - rpi_pico2_rp2350a_m33_w.overlay
  - xiao_nrf54l15_nrf54l15_cpuapp.overlay
- build
- build_1
- build_2
- src
  - core
  - main.c
- src_onoff
- .gitignor
- CMak
- prj.conf
- README.md

Overlay(デバイスツリー)
少し調整しただけ

```
22
23
24  static const struct gpio_dt_spec x001 = GPIO_DT_SPEC_GET(DT_ALIAS(plcin1), gpios);
25  static const struct gpio_dt_spec x002 = GPIO_DT_SPEC_GET(DT_ALIAS(plcin2), gpios);
26  static const struct gpio_dt_spec y001 = GPIO_DT_SPEC_GET(DT_ALIAS(plcout1), gpios);
27
28  int main(void) {
29      config_init_
30      glueVars();
31
32      // Set cycle
33      common_tickt
34
      unsigned lon
      32_t sta
      t pre
      t act
      64_t nex
```

ラズパイ4B_A72

Pico2W_M33



nRF54L15-DK_M33

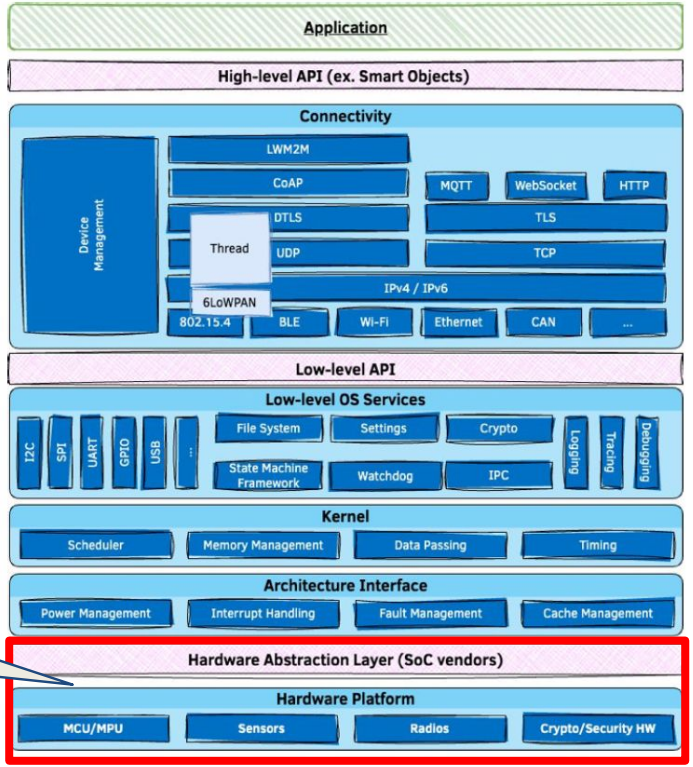
XIAO nRF54L15 Sense M33

ARMコア/メーカ違う
4つのボードを
全てが同じmain.c

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



Zephyr®

各IC/メーカの違いを
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

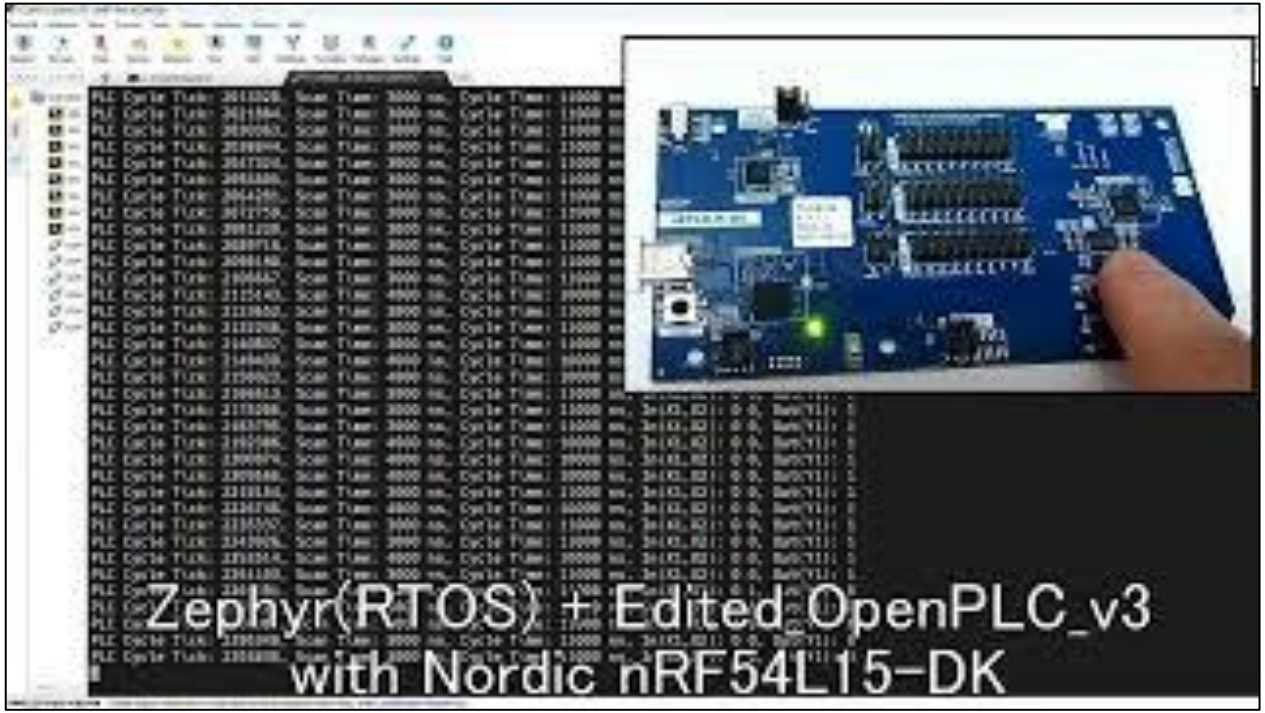
ラズパイ_4B:	west build -p -b rpi_4b
ラズパイ_Pico2W:	west build -p -b rpi_pico2/rp2350a/m33/w
Nordic_nRF54L15-DK:	west build -p -b nrf54l15dk/nrf54l15/cpuapp
Seeed_XIAO nRF54L15:	west build -p -b xiao_nrf54l15/nrf54l15/cpuapp

https://www.youtube.com/watch?v=r-i1l2-w9lw

https://youtu.be/-4zlHnl9Joc

色んなCPU・ボードに横展開

同じZephyr環境で、ボード名変えてビルドしただけ　→OK



https://youtu.be/yJj0W5qVyIY

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、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…）