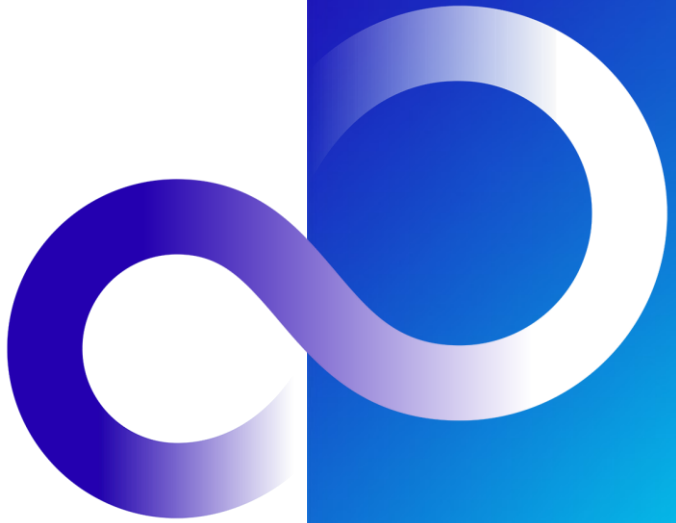
ZephyrRTOSの概要、特色とそのOSSプロジェクト運営

2024年8月30日

富士通株式会社
ミッションクリティカルシステム事業本部
ＵＮＩＸ＆ＦＸシステム事業部　ＦＸ部
常田 裕士



自己紹介

FUJITSU

- 常田 裕士
  - 1978年生まれ
  - 2001年日本大学理工学部物理学科卒業、
    同年、株式会社富士通プログラム技研入社、現在富士通に所属
    携帯電話、車載機開発を経て現在は組み込みLinuxサポートを担当
  - 2018年よりZephyrRTOSへのコミットを始める。
    現在Renesas RA、RaspberryPi Pico, GD32, LED Stripのメンテナ/副メンテナを担当
  - 掲載記事等
    - トランジスタ技術 2024年 9月号 (今月号!)
      オリジナルArduino互換！ KiCadプリント基板作り入門
    - インターフェース 2021年 6月号
      第2特集　C/C++でPython拡張
      第１部 ハードウェア効率化…C/C++で拡張モジュール作り

全体の流れ

FUJITSU

- ZephyrRTOSの概要
  - ZephyrRTOS(以下Zephyr)のあらましを説明し、どのような方向性を持ったプロジェクトなのかを紹介します。
- Zephyrの手触りから技術を知る
  - 環境セットアップからBlinkyサンプルの実行、
    そして、その背景にあるZephyrの特徴的な構造を分析します。
- Zephyrのプロジェクト運営とコミュニティ
  - ZephyrはOSSプロジェクト運営としては先駆的な試みが行われています。
    その新規性と、コミュニティの特徴をお話します。

FUJITSU

Zephyrの概要

Zephyrの概略、歴史

利用の事例

Zephyrの特徴

他のRTOSとの比較

Zephyrの目指すところ

Overview

FUJITSU

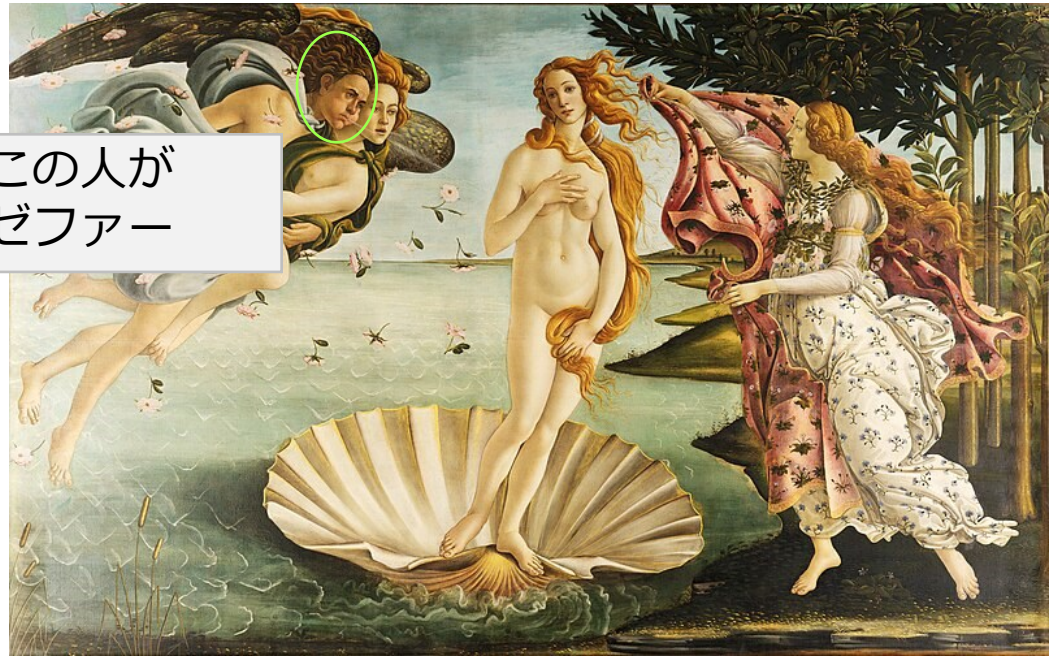
- オープンソースのリアルタイムOS
  - ArduinoとかESP32など主に「Linuxが動かない」領域
- 軽量 (メモリ/フットプリントが小さい)
- BluetoothやMatterなど、IoTのエッジデバイス分野に強い
  - メカの制御にも使える
- 対応しているSoC、開発ボードが非常に多い
- POSIX対応をはじめとして多機能
- OSとしてHALを定義している

Zephyr®

Zephyros (en:Zephyr)

FUJITSU

- 名前はギリシア神話の西風の神Zephyrosの英語読みから



Sandro Botticelli - La nascita di Venere
File:Sandro Botticelli - La nascita di Venere - Google Art Project - edited.jpg - Wikimedia Commons

History

FUJITSU

- 2014年頃、RTOSの老舗のWindRiverとIntelが開発を開始
  - WindRiverのVxWorksはBoeing B787などで使われている
  - いわば「組み込みガチ勢」とも言えるベンダー
- 2015年にWindRiver Rocketとして製品化されたものが
  Linux Foundationに寄贈され、 2016にオープンソースとして公開された
- 2024年7月(先月)にversion 3.7.0がリリース
- 2024年8月14日、10万コミットを達成 (現在1.22commits per hours)

Zephyr Project Developer Summit 2023 でのAnas Nasif氏の講演
9 Years in the Making, the Story of Zephyr, Starting with Commit Nr.
https://static.sched.com/hosted_files/eoss2023/8d/ZDS2023_Anas_nashif_Nine_years_in_the_making_zephyr.pdf

リアルタイムOS列伝（7） インテルのIoT戦略から生まれたRTOS「Zephyr」は徒花で終わらない

https://monoist.itmedia.co.jp/mn/articles/2011/05/news035.html

参加企業

FUJITSU

Platinum Members

自動車

プラットフォーマー

プラットフォーマー

医療機器
(補聴器)

医療機器
(光学機器)

https://zephyrproject.org/project-members/ より注釈は筆者による

半導体メーカーが多いが、車、医療機器など、安全を必要とする企業の参画も増えている。

競合？

FUJITSU

- 自社サービスに関連付けてRTOS製品を展開するベンダーは厳密な競合ではないものの、一定の緊張関係は発生すると考える。
  - Amazon
    - FreeRTOSを2017年に買収。GPLからMITへライセンスを変更している。
  - Microsoft
    - 2019年にExpressLogicを買収、ThreadXをAzure RTOSとしてリブランディング
    - 2023年にAzure RTOSはEclipse Foundationに寄贈されOSS化された。
- 結局どれもOSSになり、緩やかな協力・競合関係にあるともいえる。
- 歴史的にみても、明確な「覇権」は取りづらい分野。

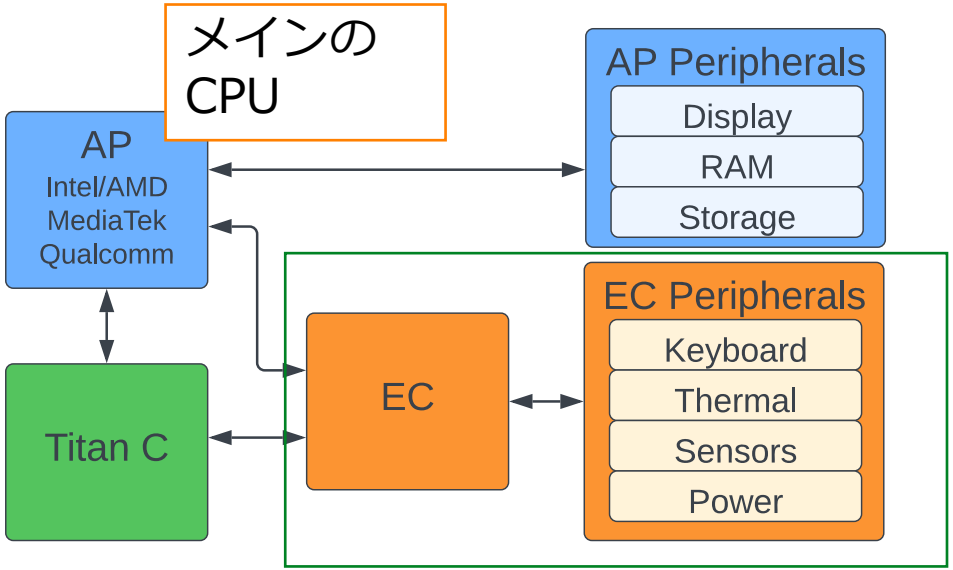
誰がどこでZephyrを使ってる？ #1

FUJITSU

- Google Chromebook の電源・ボタン制御

- 電源回りのハードウェアの
  インターフェースを統一して
  汎用的に使えるようにしている。

- メインのCPUでは動いていない。
- システムのスキマを埋めている



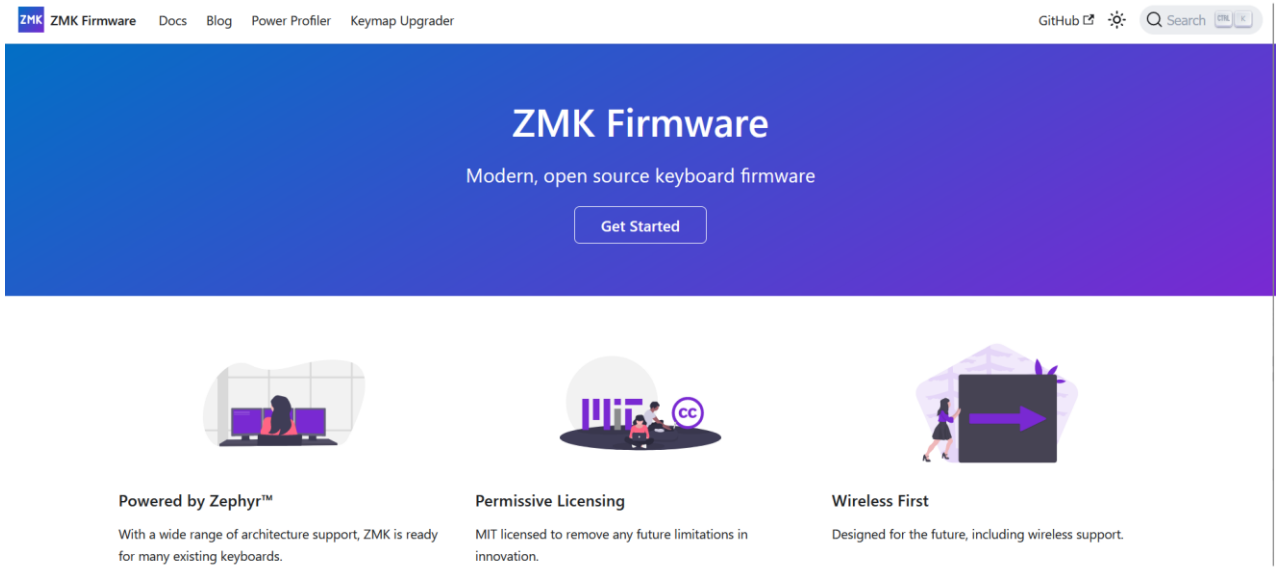
Zephyrでデバイスを制御している

https://www.zephyrproject.org/chromeoss-embedded-controller/ より。注釈は筆者による

誰がどこでZephyrを使ってる？ #2

FUJITSU

- ZMK



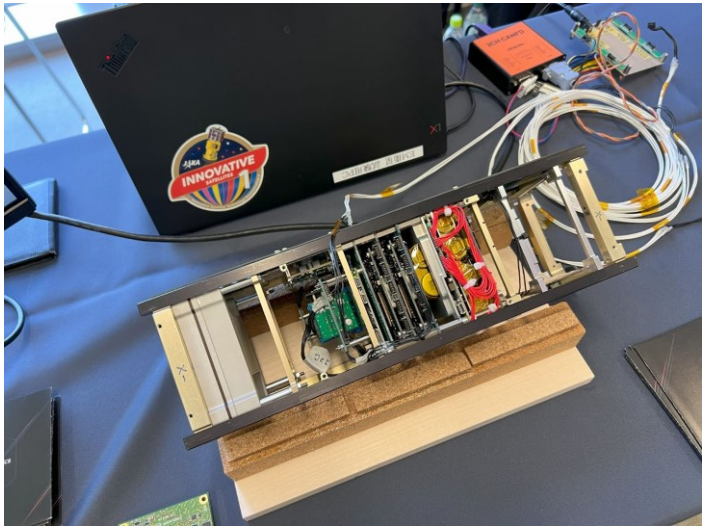
https://zmk.dev

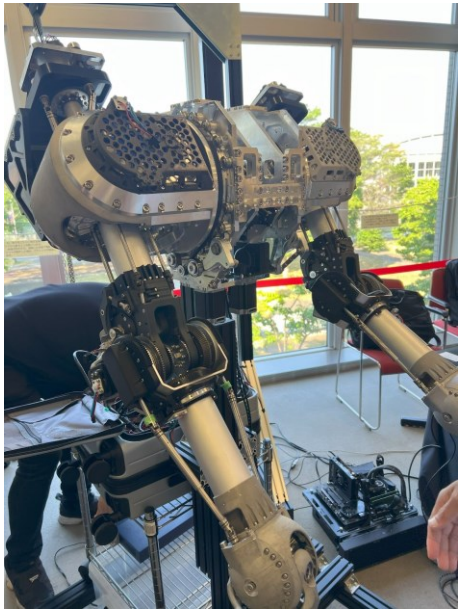
- カスタマイズ性に優れた、自作キーボード向けのファームウェア。
- Bluetoothのサポートに利用している。

誰がどこでZephyrを使ってる？ #3

FUJITSU

- ロボット・宇宙開発
  - 札幌のSpaceCubicsさんが、福島第一原発の廃炉ロボット、CubeSatへの利用を行っている。

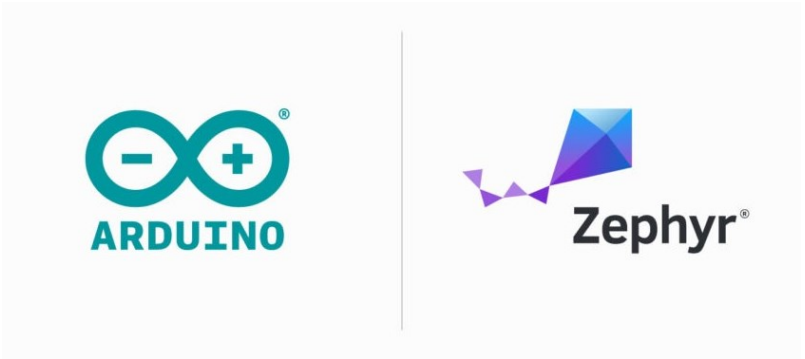




誰がどこでZephyrを使ってる？ #4

FUJITSU

- 電子工作
  - Mbed OSがEOLとなるため、Zephyrへ移行する。
    The end of Mbed marks a new beginning for Arduino | Arduino Blog
  - 実はZephyr向けのArduino API実装は元コンセプトと現実装の半分くらいを私が作成している。(その後、Google Summer of codeで進めてもらっている)



https://blog.arduino.cc/2024/07/24/the-end-of-mbed-marks-a-new-beginning-for-arduino/ より

機能 #1 ネットワーク(Connectivity)

FUJITSU

- コンシューマー機器向けのBlueToothを中心に、プロトコル実装が充実している
  - Bluetooth
    - Bluetoothの「盟主」のNordic Semiconductorが精力的にコミット
  - TCP/IP
    - OSにインテグレートされて「第１級」のサポートとなっている点がLWIPよりも優位
  - Matter/OpenThread
    - スマートホーム向けの規格のMatterも早期から対応が進められている
  - ModBus
    - FA系もスコープに入っている。
  - CANBUS
    - 車載も対応している。

機能 #2 SBOM生成

FUJITSU

- SPDX2形式のSBOM=Software Bill Of Materials（ソフトウェア部品表）の出力にも対応
  - セキュリティ、OSSのライセンス管理で近年注目されている
  - 2021年５月にサイバーセキュリティ強化の米大統領令でソフトウェアサプライチェーンの透明性に関してSBOMへの対応が要請されている。

https://spdx.dev

- 使用しているコードの出処を明確にして、プログラムと合わせて管理できる。
  - 脆弱性が検出されたら、影響有無がすぐにわかる。

広範なサポート対象

FUJITSU

- ARM, RISC-Vをはじめx86, SPARC, MIPSなど多くのアーキテクチャに対応。
- Nios II, ARC, MicroBlazeなどFPGA向けのソフトコアCPUの対応もある。
- ドライバも後述するDeviceTreeによって効率よく再構成、再利用が可能で、現在600種類以上の開発ボードが利用できる。

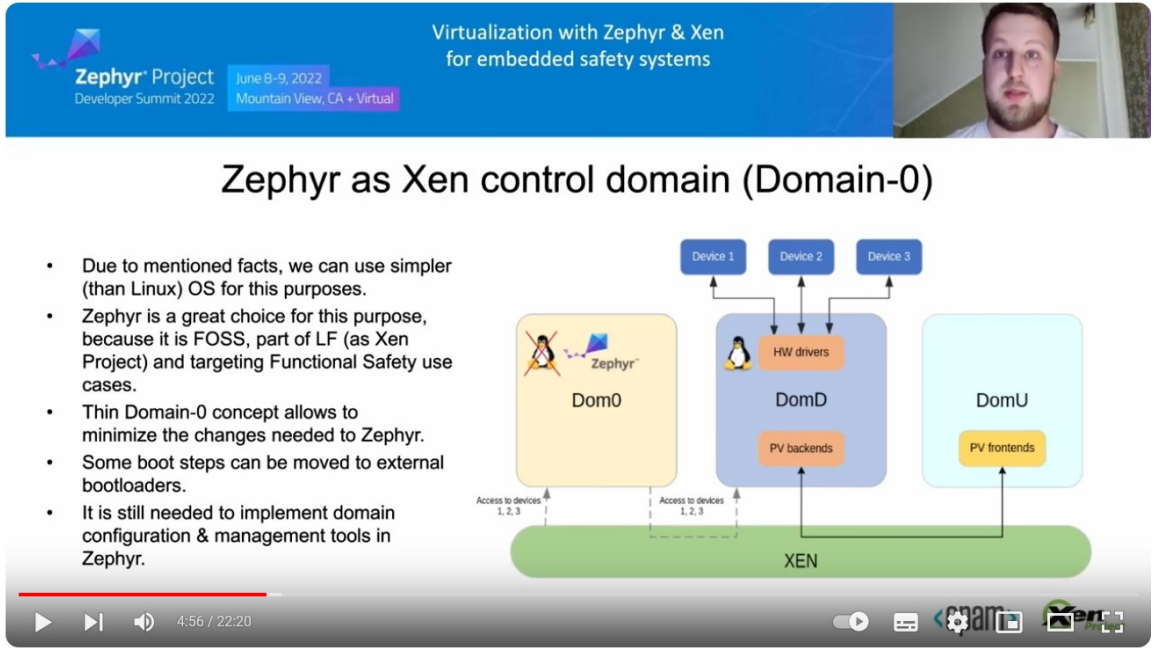
機能安全

FUJITSU

- ISO26262 ASIL-Bを目指している
- カーネル部のみで認証取得の計画
  - ドライバはメンテナンス状況によって品質のバラつきがある。
    これはLinuxと同じ
- MicroKernelアーキテクチャで分離したうえでASIL-D を取得しているQNXと比較すると見劣りする部分でもある。

個人的に注目しているトピック

- XenのDOM-0/DOM-DへのZephyrの適用
- Xenでドライバ機能を提供するOSをLinuxからZephyrに差し替える。
- 機能安全では、レビューによる検証が必要で、Linuxの規模だと対応が難しいがZephyrだと実現が視野に入る。
- RPi5のサポートに合わせてXenの開発者が動きだしている。



Zephyr Project Developer summit 2022 の Demtro Firsov氏の講演
Virtualization with Zephyr &Xen for Embedded Safety system より引用
https://www.youtube.com/watch?v=hGubBUuiaWg

他のRTOSとの比較

FUJITSU

- コードベース
  - FreeRTOS 250k (デモ除く)
  - TOPPERS 130k (asp3_raspberrypi_pico_gcc-20230401.tar.gz)
  - NuttX 2800k (デモ除く)
  - Zephyr 1900k (テスト、デモ、BSP含まず、他36,000kの外部モジュール)

- 「システム」にどこまで含めるかの考え方の影響が大きい。
  - Zephyrは外部モジュール含めて巨大な「ディストリビューション」を形成していると考えるのが良い。
  - グラフィック機能、暗号化、ブートローダーなどが外部モジュールとなっている。

Zephyrの目指すところ

- Connected と絞っていることに注目。
- とはいえ”best-in-class”と言ってるので「天下盗り」に名乗りを上げている。
- 実感としては２つの目標
  に向かっている
  - 最強のIoT-OS
  - OSSのエコシステムにおける、
    Safety実現のキーコンポーネント

Zephyr’s Vision

The Zephyr Project strives to deliver the **best-in-class RTOS** for connected resource-constrained devices, built to be secure and safe.

Open Source Summit Japan 2023 Kate Stewart氏の講演
Zephyr Project: Results from Applying Open Source Best Practices in Embedded より引用
https://static.sched.com/hosted_files/ossjapan2023/94/20231206%20OSSJ%20Zephyr%20Presentation.pdf

FUJITSU

Zephyrの手触りから技術を知る

まずは簡単に使ってみる

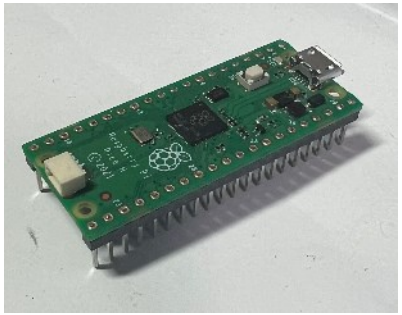
手触り感を確認する

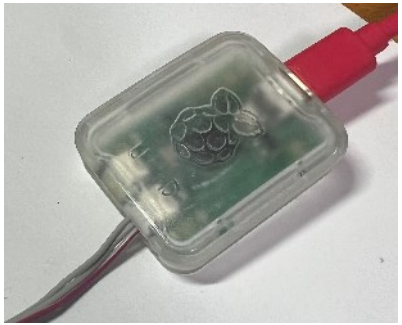
- west
- ドライバHAL
- ANSI/POSIX
- DeviceTree
- 割り込みベクタテーブル

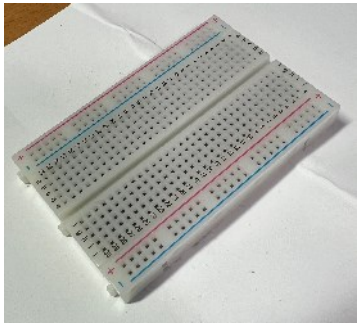
Zephyrを使ってみる

FUJITSU

- Raspberry Pi Picoが安くておすすめ。
  - Pico2は入手したら対応進める予定。
- Pico Hははんだ付け苦手な人にもおすすめ。
- Raspberry Pi Debug Probeも用意する。
- ブレッドボードやジャンプワイヤの配線も忘れずに！







セットアップ

FUJITSU

- Getting Started Guideに従ってセットアップ。
  - コマンド類のインストール
    - apt install …
  - Pythonのvenvを作成する。
    - python –m venv …
  - westコマンドのインストール
    - pip install west
  - ZephyrRTOSのソースを取得
    - west init ; west update
  - コンパイラなどをインストール
    - west sdk install (このコマンドは先週私が追加した！)
- インストーラーなどをダウンロードしていないところに注目
  - 最近のOSSの配布のエコシステムに乗っかっている

コンパイルと書き込み

FUJITSU

- Zephyrでは west というフロントエンドのコマンドを使う。
- コンパイルからテストの実行、SBOMの生成まで、全部このコマンドで実行できる。
- コンパイルと書き込みも簡単に。

```
$ west build –b rpi_pico samples/basic/blinky
$ west flash --runner pyocd
```

デバッグ

FUJITSU

- これもwestコマンドを使う

```
$ west debug --runner pyocd
```

- Debugも一発でgdbが動く。
  設定に3日も時間溶かさなくて良い！
- VSCodeなどフロントエンドとのインテグレーションは課題（だと思う）。

Lチカ(Blinky)のソース #1

FUJITSU

```
#include <stdio.h>
#include <zephyr/kernel.h>
#include <zephyr/drivers/gpio.h>

/* 1000 msec = 1 sec */
#define SLEEP_TIME_MS   1000

/* The devicetree node identifier for the "led0" alias. */
#define LED0_NODE DT_ALIAS(led0)

/*
 * A build error on this line means your board is unsupported.
 * See the sample documentation for information on how to fix this.
 */
static const struct gpio_dt_spec led = GPIO_DT_SPEC_GET(LED0_NODE, gpios);

int main(void)
{
        int ret;
        bool led_state = true;
```

Lチカ(Blinky)のソース #2

FUJITSU

```
        if (!gpio_is_ready_dt(&led)) {
                return 0;
        }

        ret = gpio_pin_configure_dt(&led, GPIO_OUTPUT_ACTIVE);
        if (ret < 0) {
                return 0;
        }

        while (1) {
                ret = gpio_pin_toggle_dt(&led);
                if (ret < 0) {
                        return 0;
                }

                led_state = !led_state;
                printf("LED state: %s¥n", led_state ? "ON" : "OFF");
                k_msleep(SLEEP_TIME_MS);
        }

        return 0;
}
```

westコマンド #1

FUJITSU

- Zephyrの総合的なフロントエンドを提供するコマンド
  - Zephyrの操作はほぼすべてこのコマンド経由で実行する
  - 以前はcmake & ninjaを直接使っていたが、全部west経由に。
  - いまではZephyrの特徴の一つと言えるほどプロセスの中核に居座っている
- GitHubのghコマンドやVueやflutterのコマンドなど、
  Web系の文脈から見ると自然な導入に見える
  - Zephyrは組み込みの文脈だけがルーツとなっていない！

westコマンド #2

FUJITSU

- westコマンドはpipでインストールしたが、主なサブコマンドの実体はzephyrのソースツリーの中に含まれている。
- pipで入るのはコア機能のコマンドだけ。

- 動きの速いZephyr本体のソースにコマンドの機能を含めて、コマンドの追加や更新を迅速に行えるようにしている。
- この辺にもWeb系寄りの設計思想がうかがえる。
- Gitのツリーはソースだけでなく開発環境も含む。

ドライバAPI (HAL)

FUJITSU

- gpio_is_ready_dt
- gpio_pin_configure_dt
- gpio_pin_toggle_dt
  - いずれもGPIOドライバのAPIで、ベンダーやハードの違いを気にせず同じように使える。
- FreeRTOSなどでは、ドライバのI/Fを定義せず、BSP提供のdriverで直接ハードをたたくケースが多い。
  ZephyrのほうがLinuxのような汎用OSに近い構成となっている。
  - 移植性は高い。
  - その代わりにHAL層のオーバーヘッドが発生する。
  - HALで規定していないデバイス固有の機能が使えない。

ANSI/POSIX対応

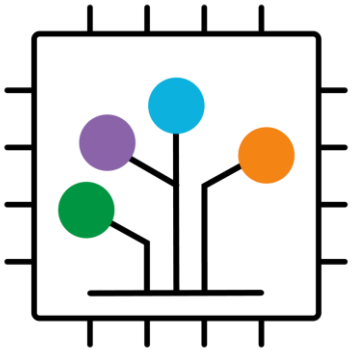
FUJITSU

- ANSI対応
  - printfをはじめとして、一般的なANSI-Cの関数が使える
  - Newlibよりも、「不十分だが軽量な」libc実装も提供されている
    - Newlib: 組み込み向けでGCCと合わせてよく利用されているlibc実装
    - Picolibc: Newlibよりも後方互換性を重視せず、最適化を図ったlibc実装
      Thread Local Storageが使える
    - Minimal-libc: Zephyrで持っている最低限のlibc実装(ANSI未対応)
- POSIX対応
  - 互換レイヤーとして提供されている。
  - 機能が細分化され、それぞれでON/OFFできる。
  - 印象としてはまだ「第一級」にはなれていない。
    - 元々POSIX対応を強く意識しているNuttXなどと比べると見劣りするところではある。

DeviceTree

FUJITSU

- DeviceTreeはZephyrを特徴づける最大の要素
  - 「汎用RTOS」であるため、カーネルの基本機能で強い「色」はついていない。
- Linuxと違って、DeviceTreeを静的にコンパイルして組み込む。
- 組み込み用途で必要なフットプリント最小化と、
  実装と構成の分離を同時に実現している。
  - このトレードとして、
    大変煩雑なC言語のマクロの記述が必要になる
  - このpros/consのトレードの判断こそが
    Zephyrを特徴づけている



LinuxでのDeviceTree

FUJITSU

- DeviceTreeの仕様にある、DTB形式（DeviceTree Blob）を使う。
- テキストで記述されたDTS(DeviceTree Source)をDTBにコンパイルする
- カーネルが起動時にDTBを読んで、構成(Configuration)データとして利用する。

ZephyrでのDeviceTree

FUJITSU

- DeviceTreeの書式はLinuxと同じ（DeviceTreeの仕様通り）
- ZephyrではDTBを使わない
- DeviceTreeから巨大なC言語のコード（マクロ）を生成する

- 動的な構成の変更はできない。
- 実装と構成を分離して、コードの再利用性を高めるのが目的。
  - Linuxでもこの目的で導入された。
    LKML: Linus Torvalds: Re: [GIT PULL] omap changes for v2.6.39 merge window (ARM関連のコードが激増し、Linus氏が激怒したという逸話)
  - Zephyrでも同様に効果が上がっている。

LinuxとのDeviceTreeの扱いの比較

FUJITSU

	Linux	Zephyr
書式	DeviceTreeの仕様どおり	DeviceTreeの仕様どおり
バイナリ表現	DTBを使う	使わない
組み込み方	DTBをカーネル起動処理中に 読み込む	コンパイル時にマクロとして展開。 メモリ上に情報は置かれない。
動的な構成	オーバーレイで構成を変えられる	コンパイル時に確定するので不可
メモリ使用量	多い（システム全体からは微小）	無い（一般的にはROM領域の情報になる）

具体的な現れ方

- Lチカのソースで DeviceTreeの情報にアクセスしているのは以下の箇所。

```
#define LED0_NODE DT_ALIAS(led0)
static const struct gpio_dt_spec led = GPIO_DT_SPEC_GET(LED0_NODE, gpios);
```

- 以下のようなDeviceTree定義と対応する。

```
/ {
	aliases {
		led0 = &myled0;
	};

	leds {
		compatible = "gpio-leds";
		myled0: led_0 {
			gpios = <&gpio0 13 GPIO_ACTIVE_LOW>;
		};
	};
};
```

DeviceTreeの闇

FUJITSU

- DeviceTreeでは、配列の定義があるので、Zephyrのマクロでも配列をループ処理するためのマクロ(!)が生成される。
  - ～_FOREACH_…という名前で、マクロを引数に取って、それを各要素に適用する高階関数(高階マクロ？)が作られる。
- COND_CODE_1
  - #ifdefと似たような動きをするが、マクロの引数で与えた値でコードの有効無効を切り替える機能。
  - 中身は完全に黒魔術（##を使った多重のマクロ展開をうまく使っている。解析が非常に面倒）
  - 上記のFOREACHと組み合わせると、DeviceTreeの定義に合わせて、無駄なくコードを生成できる。

一例

```
#define SERIAL_DEFINED_0              1
#define EXTERN_SERIAL_N(i)            extern arduino::ZephyrSerial Serial##i;
#define DECLARE_EXTERN_SERIAL_N(n, p, i) COND_CODE_1(SERIAL_DEFINED_##i, (), (EXTERN_SERIAL_N(i)))

/* Declare Serial1, Serial2, ... */
DT_FOREACH_PROP_ELEM(DT_PATH(zephyr_user), serials, DECLARE_EXTERN_SERIAL_N)
```

私が作成したZephyr向けArduinoAPIの中から。gsoc-2022-arduino-core/cores/arduino/zephyrSerial.h

DeviceTree上の配列serialsを見て、0番目以降の要素に対してSerial1, Serial2, …と割り振って宣言する。ここで0番目はすでにSerialとして定義済みなので除外したい。

COND_CODE_1は第一引数が定義済みで真なら第二引数、そうでなければ第三引数をコードに置き換えられる。ただし、第一引数は内部で別のマクロと連結して処理されるため、整数リテラルとして定義されている必要がある。

ここでは、序数を連結すると、整数に展開できるSERIAL_DEFINED_0を定義して、この動作に適応させて処理を行っている。

複雑なコードであるが、DeviceTreeの要素を適切に扱えており、また、すべてコンパイル時計算となるので効率的である。DeviceTreeによる実装と構成の分離、HALはLinuxから受け継いだHeritageの部分にもあたる。
dirtyではあるが、この仕組みによって非常に巧く処理されており、Zephyrを特徴づけている部分でもある。

割り込みベクタテーブルの処理 #1

FUJITSU

- Zephyrでは割り込みベクタテーブルを直接記述しない。
- IRQ_CONNECTのマクロで、割り込み番号とそれに対応するハンドラの関数ポインタを指定する。コード例はraspberrypi picoのGPIOより。

```
#define GPIO_RPI_INIT(idx)                                                  ¥
        static void bank_##idx##_config_func(void)                          ¥
        {                                                                   ¥
                IRQ_CONNECT(DT_INST_IRQN(idx), DT_INST_IRQ(idx, priority),  ¥
                            gpio_rpi_isr, DEVICE_DT_INST_GET(idx), 0);      ¥
                irq_enable(DT_INST_IRQN(idx));                              ¥
        }                                                                   ¥
```

割り込みベクタテーブルの処理 #2

FUJITSU

- このマクロは、割り込み番号、ハンドラのアドレスのデータを.intListセクションに出力する

```
.intList        0x00000000ffff8000       0x28
 *(SORT_BY_ALIGNMENT(.irq_info*))
 .irq_info      0x00000000ffff8000        0x8 zephyr/arch/common/libisr_tables.a(isr_tables.c.obj)
                0x00000000ffff8000              _iheader
 *(SORT_BY_ALIGNMENT(.intList*))
 .intList       0x00000000ffff8008       0x10 zephyr/drivers/gpio/libdrivers__gpio.a(gpio_rpi_pico.c.obj)
 .intList       0x00000000ffff8018       0x10 zephyr/drivers/serial/libdrivers__serial.a(uart_pl011.c.obj)
~~~~~~~~~~
 .text.gpio_rpi_isr
                0x0000000010001670       0x64 zephyr/drivers/gpio/libdrivers__gpio.a(gpio_rpi_pico.c.obj)
```

割り込みベクタテーブルの処理 #3

FUJITSU

- 一旦リンクして、pythonで.intListのセクションを解析して割り込みベクタテーブルを作成する。

```
struct _isr_table_entry __sw_isr_table _sw_isr_table[26] = {
        {(const void *)0x0, (ISR)z_irq_spurious}, /* 0 */
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        {(const void *)0x10004e00, (ISR)0x10001671}, /* 13 */
```

- ドライバの実装と割り込みベクタの構成を分離するための工夫。

Pythonの活用

FUJITSU

- westコマンド
- コンパイル中の処理
  - DeviceTreeのマクロ化
  - Sysytem-Callの定義
  - 割り込みハンドラテーブルの生成
- CI関連の様々な処理

- westコマンドが目立つが、実際にはビルドプロセス中の様々な細かい処理がpythonで行われており、これで一つの大きなシステムを成している。
- DeviceTreeや割り込みハンドラなど、Zephyrの「色」が出ているところの処理が多い。

FUJITSU

Zephyrのプロジェクト運営とコミュニティ

組織

ツール

コミュニケーション

OpenSource Project

FUJITSU

- Linux Foundation主導のオープンソースプロジェクト
- Apache2ライセンスで提供
  - WebサーバーのApacheのほかAndroidがApacheライセンスの代表例
- コミットが多く、開発のスピードがとても速いプロジェクト
  - 通算で1.22commits/Hour
- 今回招待いただいた背景にもあるように、近年急速に伸びている。

Linux Foundation

FUJITSU

- Linuxの開発者のLinus Torvaldsを雇用している非営利団体
- Linux Kernel, Zephyrをはじめ、1100を超えるオープンソースプロジェクトを支援する財団
- EU Cyber Resilience Actなど法令化の対してロビーイングを行う業界団体的な側面もある。
- Zephyrの統括はVice PresidentのKate Stewartさん。
  同時に SPDX、ELISAなどコンプライアンス系のプロジェクトも担当している。

Governance Model

FUJITSU

- Technical Steering Committee(TSC)
  - 技術的な意思決定をする会議体
  - スポンサーになるとこの会議に対しての投票権が得られる
    - 投票権があるといっても、独断で方針決定ができるわけではない。
    - ベンダーニュートラルに運営されている。
- 各機能ごとの担当者
  - Maintainer (機能ごとのリーダー)
  - Collaborator (主要開発者)
  - Contributor (貢献者)

    これはスポンサーであるかどうかによらず、貢献の度合いで選出される。

    MaintainerはTSCと共同して方針を決定できる立場なので、貢献度の高い個人もZephyrの全体的な指針に対して一定の影響を持ち得る立場となる。

開発サイクル

FUJITSU

- GitHubのWikiに全部書いてある。
- 概ね4か月に1回リリース
- 2年に1回Long Term Support版が出る
  - 予測可能な計画を立てているのは Linuxとは異なるポリシーになる
- 直近では、先月に3.7(LTS)がリリースされている。
- リリース直前はレビューとマージがとても遅くなる。

Release milestone dates

Milestone	3.7 (LTS)	4.0
Planning	2024/01/12	2024/06/26
Review target milestones	2024/05/10	2024/10/11
Release and Timeline Announcement	2024/05/31	2024/11/01
Feature Freeze (RC1)	2024/06/14	2024/11/08
2nd Release Candidate (RC2)	2024/06/28	2024/11/15
Hard Freeze (RC3)	2024/07/12	2024/11/22
Release	2024/07/26	2024/11/29
End Of Life	2027/01/26	

LFの他のプロジェクトとの関連 #1

FUJITSU

- ELISA Project
  - Linuxをセーフティクリティカル用途に使うためのシステム構築や認証を支援するプロジェクト
    - 車載向けLinuxのAutomotive Grade Linuxも連携して活動している。
    - Boeingがスポンサーになり、航空宇宙がスコープに加わった。
  - LFの諸々のプロジェクトに対しての機能安全に関するメタ・プロジェクト的な雰囲気がある。
    - Zephyrも安全性分析の観点でこのプロジェクトと連携がある。
    - 仮想化のXen、Real-Time Linuxなどとも関連して進めている。

ELISA
Enabling Linux in Safety Applications

https://elisa.tech/

LFの他のプロジェクトとの関連 #2

FUJITSU

- SPDX
  - SBOMの一つのフォーマットであるSPDXの開発を推進する。
  - 国内の活動も活発。「手書きSBOM」を意識したSPDX Liteは日本からの提案が標準化されている。
  - ZephyrはSPDX2形式のSBOMを出力可能。
  - SBOMはサイバーセキュリティに関連して、政府調達要件に組み込まれる方向に進んでいる。

- ELISA, SPDXともにZephyrの担当のKate Stewart氏の担当でもある。複数のプロジェクトで「安全」のためのオープンソースのアセットを作り上げるような動きになっている。
- この点に着目して新規メンバーになっている企業もあると考える。

GitHub

FUJITSU

- Githubを中心にワークフローが作られている。
- 実は非常に大きな特徴。
  - WiKiや議論用のBBSの機能もGitHubのもので賄っている
  - ワンストップのポータルになっている.
- GitHub Actionで作られたCIが極めて充実している。
- GitHub主導のCIシステムとしてはかなり先駆的で
  モデルケースとなりえる。

GitHub Actionを使ったCI

FUJITSU

- Pull Request作成時(パッチ投稿時)に実行される。
  - 結構なリソースを使う
  - 全コンパイルではないが、かなりの量のコンパイル確認が流れる。
  - 終わるまで半日ぐらいかかることも。

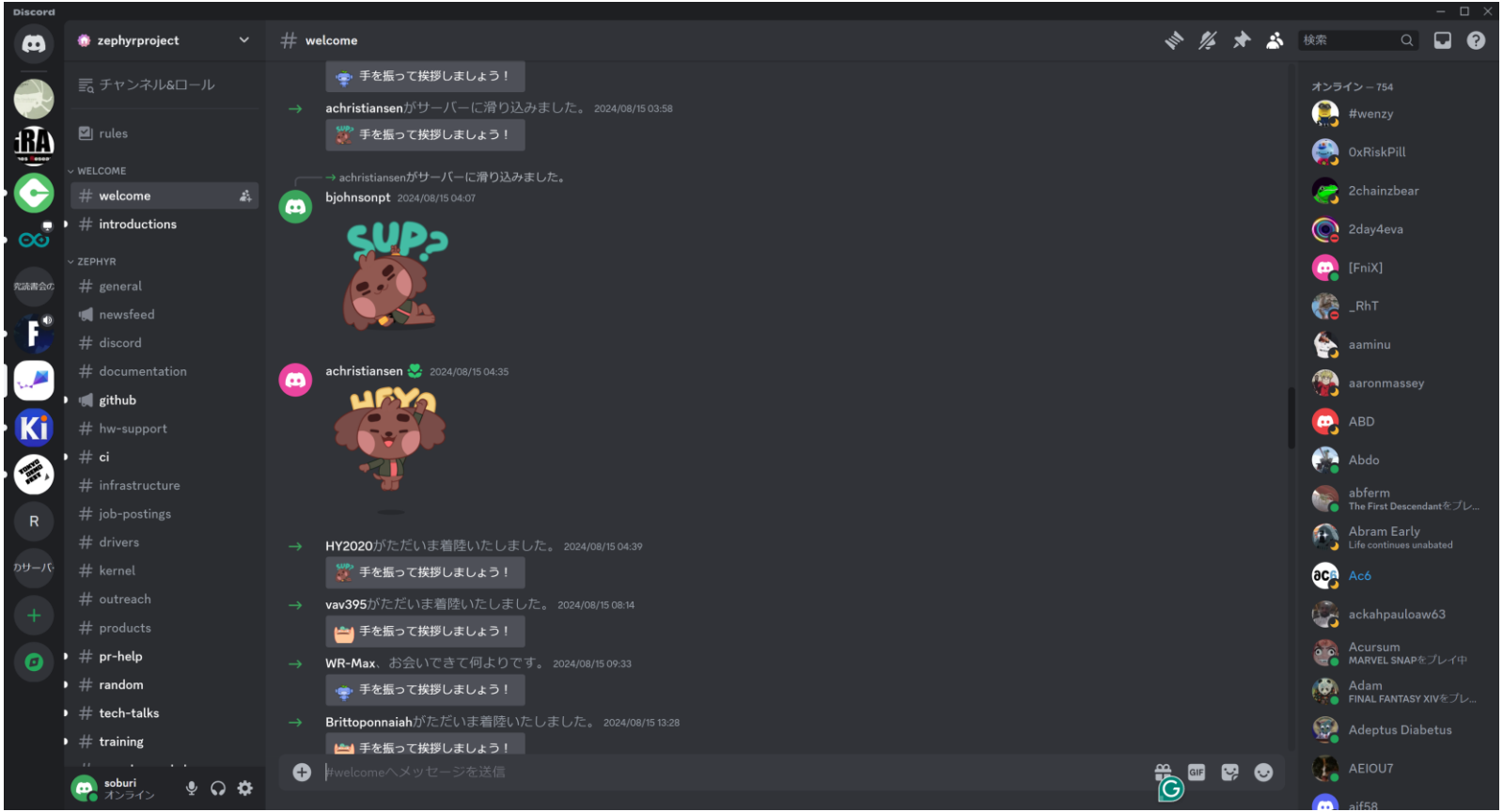
- Integration Platformとして選ばれているターゲットに対しては同時に自動試験も実施している。
- 商用の静的チェックツールのCoverityも利用している。

Discord

FUJITSU

- オンラインコミュニケーションのサービス
  - SlackやTeamsと類似したもの
- 元々ゲームのオンラインチャット向けのサービスとして出発。
  - 現在でもメインストリームはそちら。
  - 故にかなり砕けた文化がある。
- OSSだと、Linuxを筆頭にメールベースのコミュニケーションを主にしているものも多い。
  - Zephyrの競合ともいえるNuttXもメールベース
  - ZephyrにもMLはあるが、ミーティングの告知ぐらい。議論には使っていない。
- とはいえ、メールは「古い文化」
  新しいツールの導入の結果、若い開発者が多いように感じる。

Discord: ScreenShot



開発に参加する

FUJITSU

- 情報源
  - Zephyr Weekly updates
    - Linux FoundationのBenjamin Cabéさんがまとめてる Zephyrの週報。読むと、なんとなく最近の開発の雰囲気が伝わる。
      https://zephyrproject.org/zephyr-weekly-update-new-soc-porting-guide/
- Discord
  - メインのコミュニケーションチャネル。昨日やベンダーでチャンネル分け。
  - 面白いところだと #job-postiongs として求人のチャネルがある。
- Github
  - Pull Request作って議論を始めるのももちろんOK. (Code First!)

Meetup

FUJITSU

- 日本で言うところの「勉強会」
- 各地でそれぞれ運営しているが、Linux Foundationが公式に広報や運営の面でサポートしている。
- 今までの「Users Group」モデルよりも公式がしっかりサポートしている。
- この辺のやりとりもDiscordで機動的に行われている。
- 今年のOSC北海道で行ったZephyrに関するセッションもこのスキームでLFからサポートしてもらっている。
- ノベルティを確保したので、日本でもやろうかと計画中。

Zephyr Project Meetup

FUJITSU

https://zephyrproject.org/event/zephyr-project-meetup-munich-germany/ より引用

Beginner Friendly

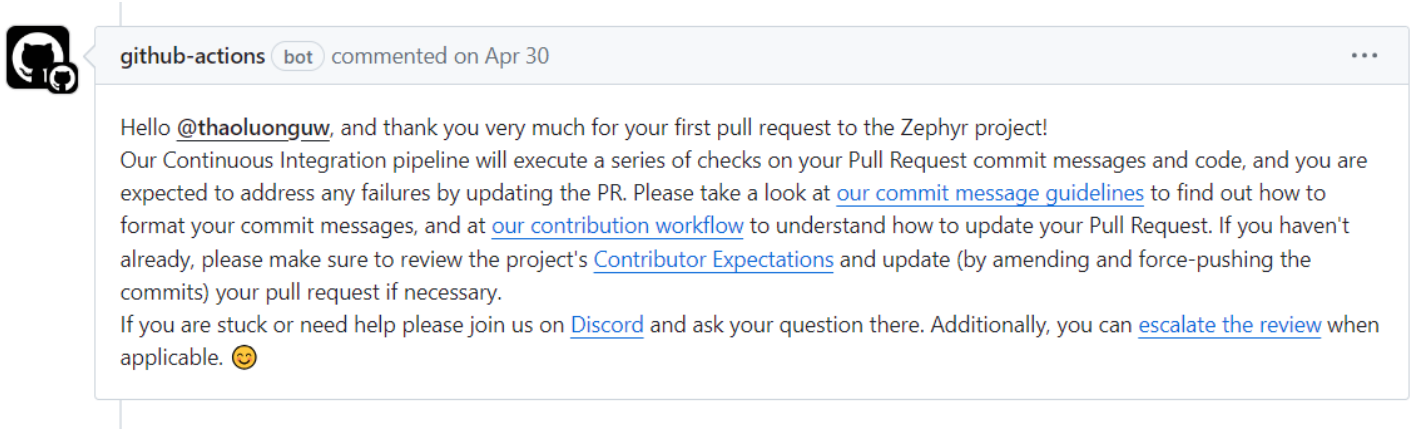
FUJITSU

- プロジェクトマネジメントのKPIとして、
  新規コントリビューター数を非常に重視している

- 私もGitHubの使い方や、かなり基本的な事項を指導しながらレビューを進めた経験が何度かある。

- 自動応答メッセージや、Weekly Updateを通じて、新規コントリビューターへの称賛を送って、定着を目指している。

- 初投稿歓迎の自動応答メッセージ



github-actions bot commented on Apr 30

Hello @thaoluonguw, and thank you very much for your first pull request to the Zephyr project!
Our Continuous Integration pipeline will execute a series of checks on your Pull Request commit messages and code, and you are expected to address any failures by updating the PR. Please take a look at our commit message guidelines to find out how to format your commit messages, and at our contribution workflow to understand how to update your Pull Request. If you haven't already, please make sure to review the project's Contributor Expectations and update (by amending and force-pushing the commits) your pull request if necessary.
If you are stuck or need help please join us on Discord and ask your question there. Additionally, you can escalate the review when applicable. 😊

- 週報で新規コントリビューターを紹介

A big thank you to the **16 individuals** who had their first pull request accepted this week, 💙 🙌: @alexstanoev-nordic, @nngt88, @pyadvichuk, @duynguyenxa, @juliaazziz, @LeoBRIANDSmile, @aa889788, @yiding, @konrad1s, @unsanded, @thales-nascimento, @pblxptr, @LiLongNXP, @Robibobo1, @asingh-GiN, and @00thirdeye00.

まとめ

FUJITSU

- ZephyrRTOSはIoTに強みのあるオープンソースのリアルタイムOS
  - 汎用志向で、逆説的にスケジューラやメモリ管理のような基本機能に強い「色」はない。
- Linuxから設計思想と成功モデルを引き継ぎ、多様なデバイスのサポートを実現
  - 特筆すべきはDeviceTreeで、いくらかのDirtyさはあるが、Linuxで実現している多種にわたるドライバの管理容易性を受け継ぎ、成功モデルとして取り込んでいる。
- GitHubやDiscordなど新しいツールを活用し、オープンソースプロジェクト運営の先駆的なマネジメントを実践し
  若手、新規の開発者の参入に効果を上げている。
  - プロジェクトマネジメントもLinux Foundationが持つOSSコミュニティ運営のノウハウが活用され、新しいインフラを使ってオープンで民主的な運営がなされている。