

Zephyr(RTOS)でARMとRISC-Vの コア間通信を試してみた

ミソジ 2026/3/27

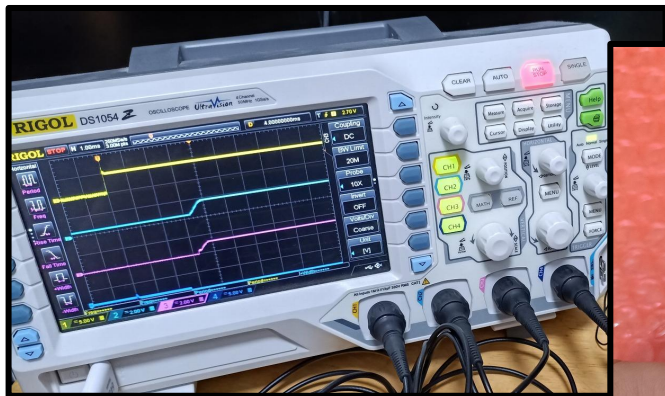
Zephyr Project meetup: Nagoya, Japan
#ZephyrRTOS

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://twitter.com/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)でコア間通信を試してみた話

- ・ このテーマを選んだ理由
- ・ Zephyr(RTOS)のトレンド・強いところ
- ・ ARMとRISC-Vでコア間通信したら、面白いのでは？
- ・ 実装+テストしてみた

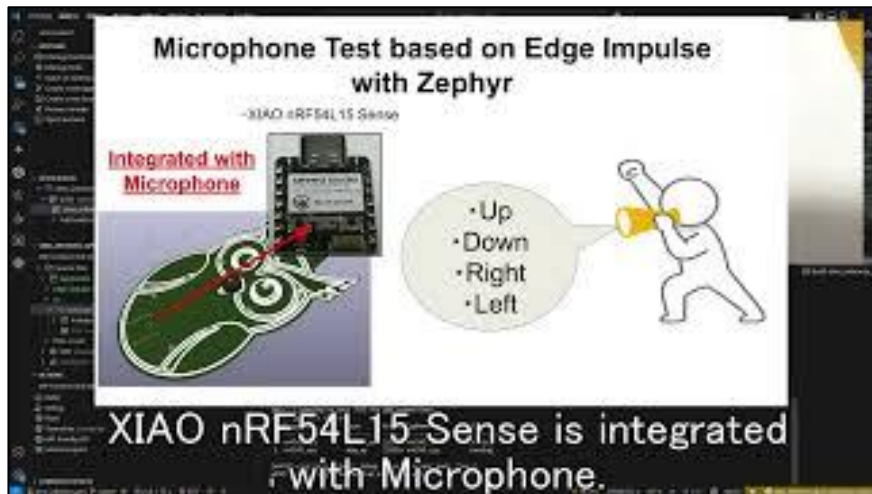
このテーマを選んだ理由

Edge AI

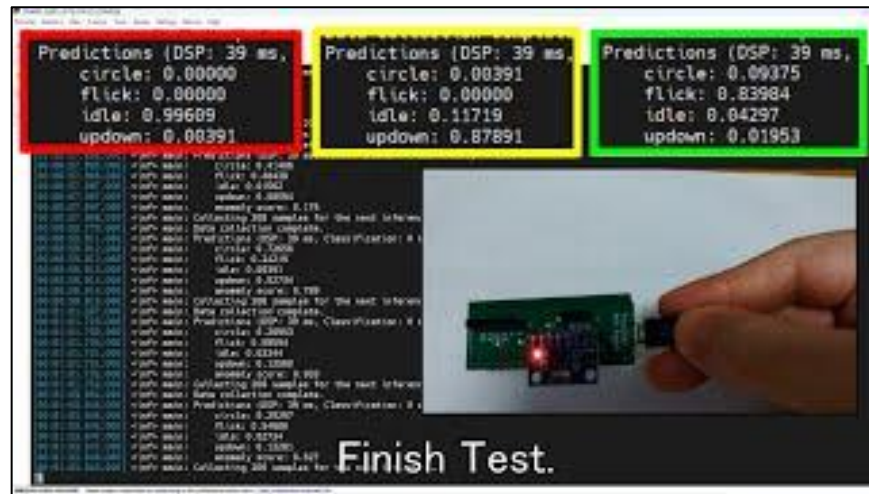
去年はZephyr + Edge AIを多く実装しました

- コンテスト、勉強会、カンファレンスでも良く使っていました
(見栄え良い + トレンド性ある + 分かりやすい)

・XIAO nRF54L15 Senseでの音声認識



・ラズパイPico2Wでのジェスチャー認識



とあるカンファレンスで...

「鉄板」+「良い質問」が来ました

- Edge AIが～
- Zephyrが～
- コンテストで～



とあるカンファレンスで...

「鉄板」+「良い質問」が来ました

- Edge AIが～
- Zephyrが～
- コンテストで～



それは
Best Choice
ですか？



とあるカンファレンスで...

「鉄板」+「良い質問」が来ました

- Edge AIが~
- Zephyrが~
- コンテストで~



それは
Best Choice
ですか？

意識(誤訳):

Zephyrに
実装する意味あるの？



致命傷で済んだ。

~~俺も思っていた。~~マジでクリティカル。



Zephyrである必要は？

正直言うと...無い。OS無しでもEdge AI処理できる

もちろん軽量・省電力というEdge AIに有効なRTOSですが...
→無理して使う必要は無い。



<https://www.zephyrproject.org/>

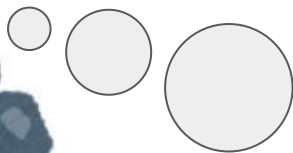
去年参加したEdge AIのコンペでも
自分以外、ほぼArduino(OS無し)で実装



2025年8月~9月
Edge AI Earth Guardians
<https://www.hackster.io/contests/earthguardians>

ただ、このままでは悔しいので...

Zephyr + Edge AIが有効であるケースを、考えてみる。



Zephyr(RTOS)の トレンド・強いところ

Zephyrの強み

SoC内のコア(OS)間の通信・仮想化もトレンドの一つ。



最近ではSoC内の
コアの種類・数が多い
(使い切れない...)

ヘテロジニアス
(異種混合)化

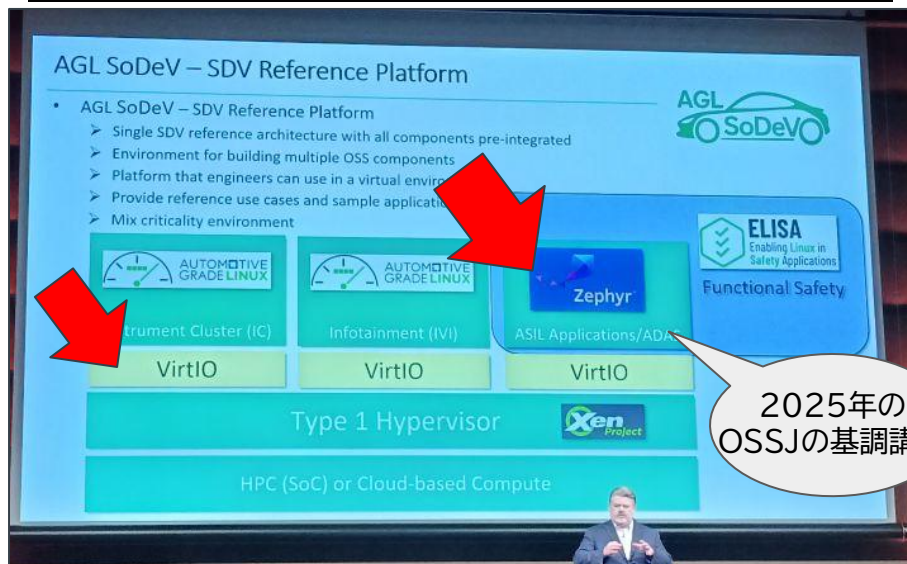
サブ_Mコア
(RTOS)

メイン_Aコア
(Linux)

NPU
(AI処理)

機能安全_Mコア
(RTOS)

車載系のプラットフォーム例
コア(OS)間の違いを越えた、I/O仮想化(VirtIO)



<https://www.youtube.com/watch?v=QjiWMYTrMQU>

Zephyrのコア間通信のサポート例

色々あるが、IPCが一番ハード寄りで自分好み

Geminiに聞いてみた

一番分かりやすい例えは、**「物流(ロジスティクス)」**です。

- **IPC** = 道路や橋(物理的なインフラ)
- **VirtIO** = 交通ルールとトラックの規格(運び方の作法)
- **OpenAMP** = 運送会社・物流システム全体(管理と運用)

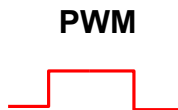
IPCとEdgeAIを
絡めればメリットあるのでは...

項目	IPC (Inter-Processor Communication)	VirtIO (Virtual I/O)	OpenAMP (Open Asymmetric Multi-Processing)
定義	コア間通信の総称	I/O仮想化の標準規格	マルチコア連携のフレームワーク
レイヤー	物理層・データリンク層(ハード寄り)	トランスポート層(抽象化)	アプリケーション・管理層(ソフト全体)
主な役割	データの通り道を確認する	メモリの使い方の作法を共通化する	コアの起動・停止やメッセージを管理する
具体例	共有RAM、メールボックス、割り込み	vring(リングバッファ)	RemoteProc, RPLMsg

IPC (Inter-Processor Communication)

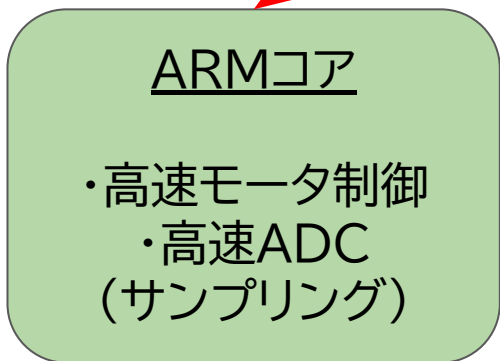
IPCで「軽量+高速」と「複雑+重い」タスクを分ける

高速な信号のため、
他処理の影響を小さくしたいケース



PWM

ADC
(Analog-Digital)



軽量+高速 タスク

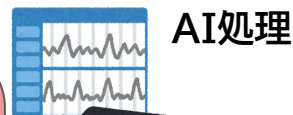


IPC

- ・共有RAM
- ・メールボックス
- ・割り込み



複雑+重い タスク



AI処理



WIFI, LTE,
Bluetooth

ちょうど良いネタ(SoC)がある

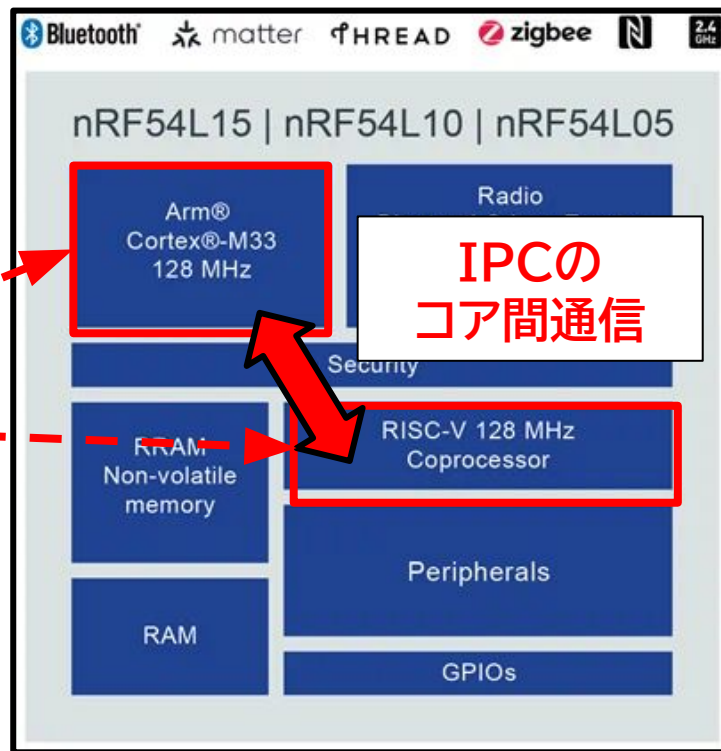
Nordic nRF54L15は「ARM」と「RISC-V」の2つのコア

■Nordic Wireless SOC nRF54L15



ARM_M33

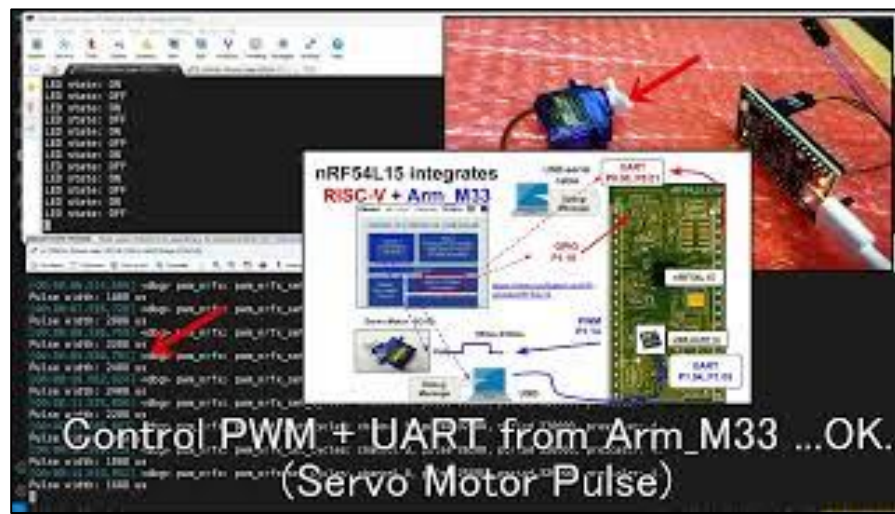
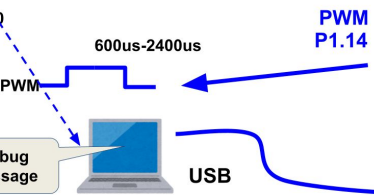
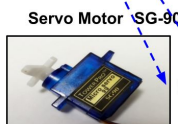
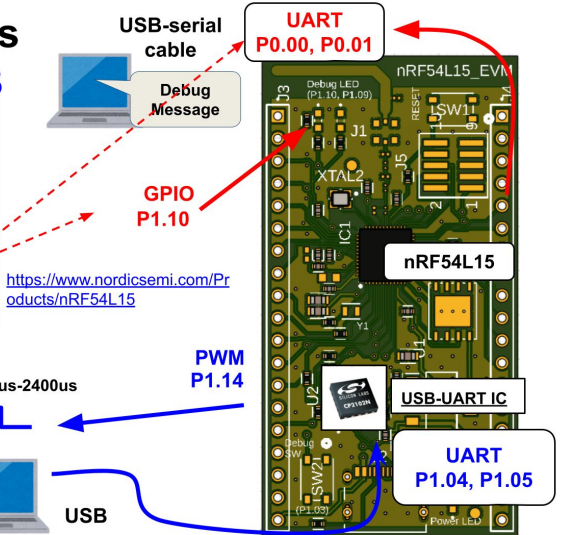
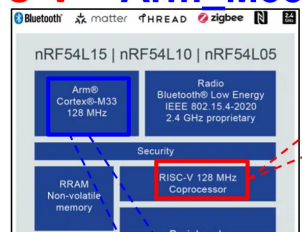
RISC-V



ARMもRISC-Vも単独ではテスト済

まだ、IPCのコア間通信はテストしていなかった。

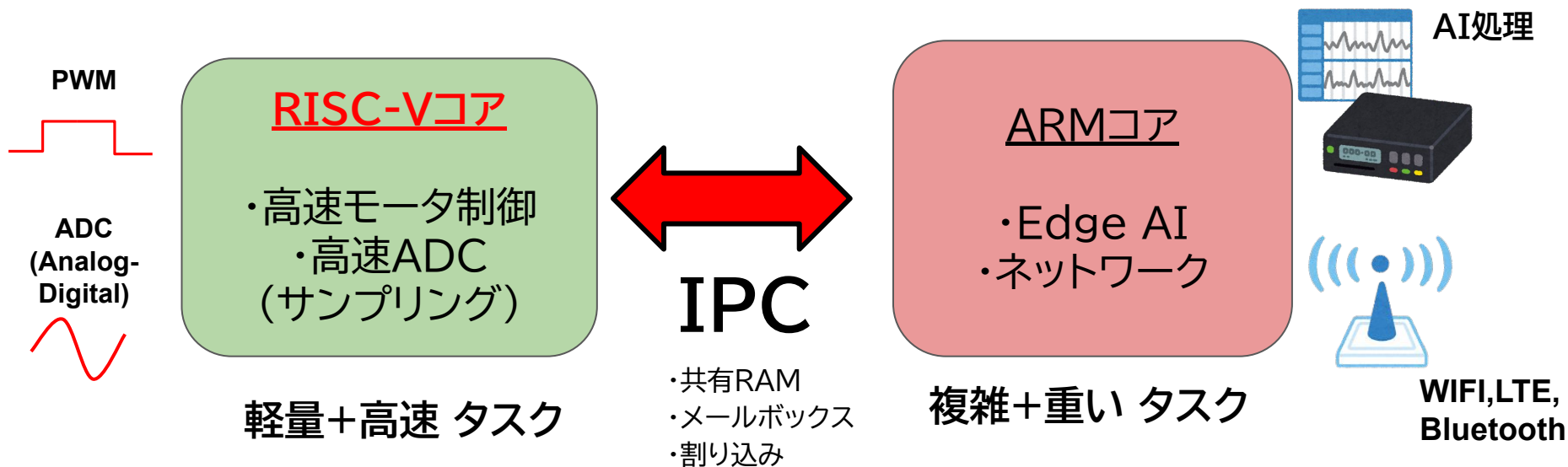
nRF54L15 integrates
RISC-V + Arm M33



■ 去年nRF54L15のカスタムボードで RISC-VとARM_M33の簡単な(GPIO・UART・PWM)動作確認はしていた
プロジェクトURL: <https://www.hackster.io/iotengineer22/maker-s-nrf54l15-debug-board-1a6a88>

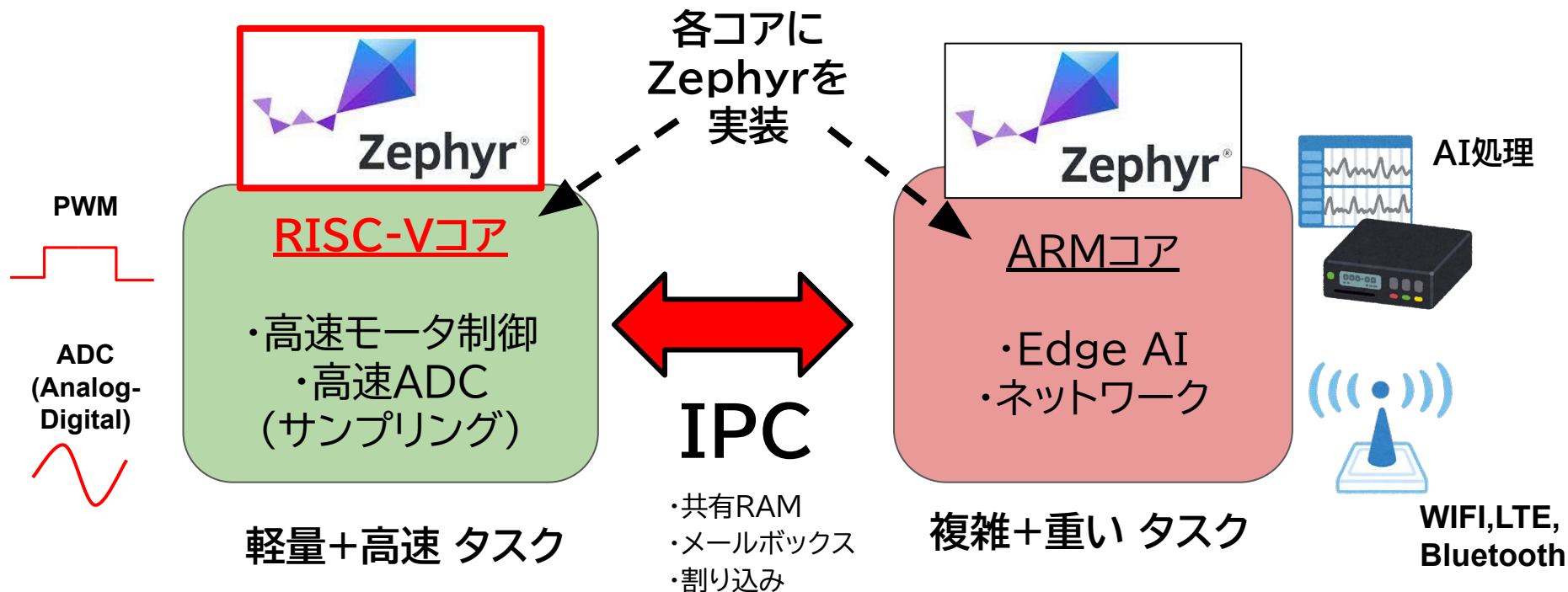
Zephyrの良さが活きる。

RTOSとして、「軽量」かつ「多くのアーキテクチャ」に対応



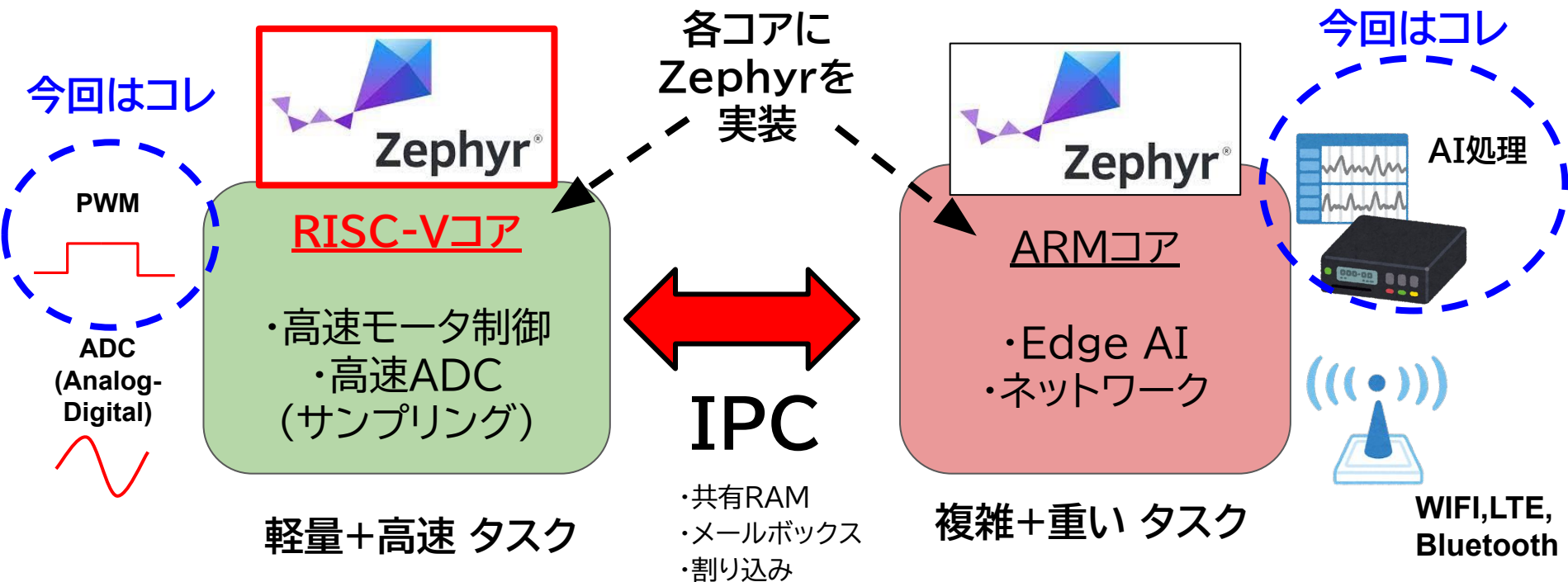
Zephyrの良さが活きる。

RTOSとして、「軽量」かつ「多くのアーキテクチャ」に対応



Zephyrの良さが活きる。

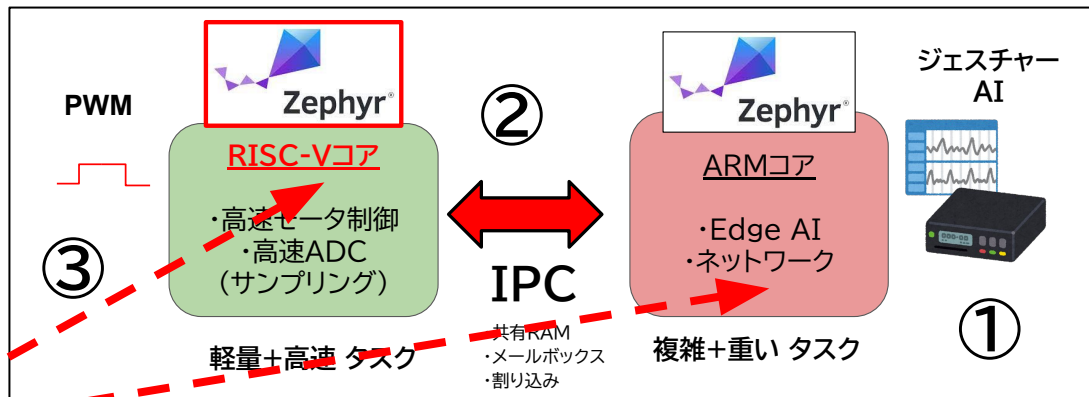
RTOSとして、「軽量」かつ「多くのアーキテクチャ」に対応



テスト実装してみる

シンプルな内容でテスト実装してみた。

■ Nordic Wireless
SOC nRF54L15



- ① ARM側でジェスチャーAI処理
- ② IPCでデータ共有 *項目とスコア値のみ
- ③ RISC-V側のPWMを変更

実装+テストしてみた

nRF54L15 + IPC

Zephyrのサンプルを調整 + 上手いことEdge AI処理を結合

*AIモデルは従来通り
Edge ImpulseのC++出力
(今回は説明省略)

①remoteフォルダにRISC-V側の
main.c、Makelist、prj.conf

②通常通りARM側の
main.c、Makelist、prj.conf

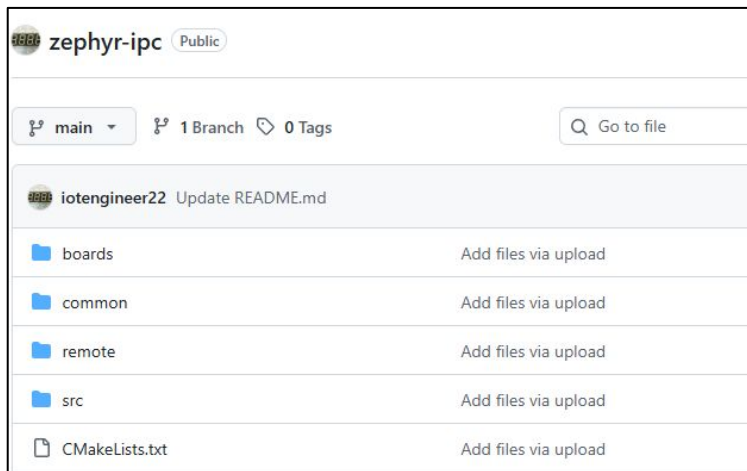
③NordicのRISC-V用の
スニペット(nordic-flpr)を使用

The screenshot displays the Zephyr IDE interface. On the left, the 'エクスプローラー' (Explorer) pane shows a project tree for 'nrf54l15dk_nrf54l15_cpuapp.overlay'. The tree is expanded to show the 'common' directory, which contains sub-directories 'edge-impulse-sdk' and 'model-parameters'. Below these, the 'remote' directory is expanded, showing 'boards', 'src', and 'trite-model' sub-directories. The 'src' directory under 'remote' contains files 'icmsg_common.h', 'main.c', 'CMakeLists.txt', and 'prj.conf'. The 'src' directory under 'boards' also contains 'icmsg_common.h' and 'main.cpp'. The 'trite-model' directory contains '.gitignore', 'CMakeLists.txt', 'Kconfig.sysbuild', and 'prj.conf'. On the right, the 'README.md' file is open, showing the project structure and build instructions. The structure includes 'CMakeLists.txt' (Main build configuration), 'prj.conf' (Host core configuration), 'src/' (Host application entry point), 'remote/' (Remote core application), and 'common/' (Common definitions). The build instructions mention using 'west' to build the application.

GitHubのリンク先

Zephyr公式サンプルの方を見てね。

(雑な)自分のテストプログラム



zephyr-ipc Public

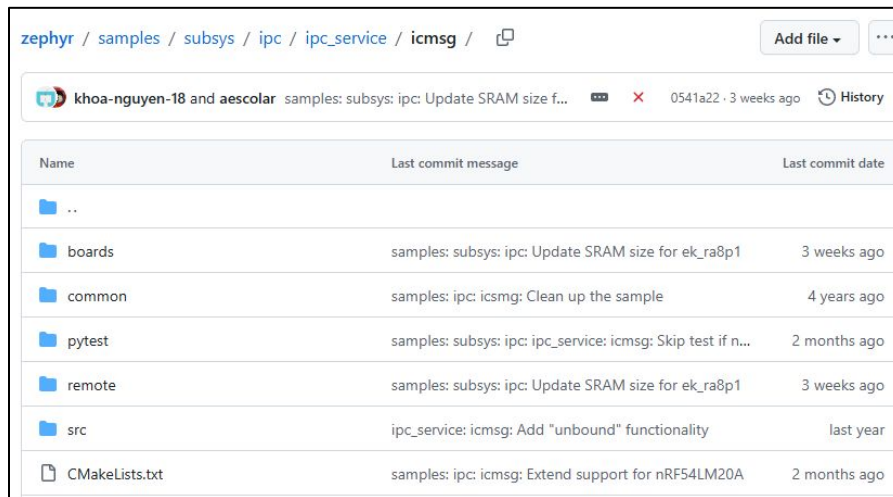
main 1 Branch 0 Tags

iotengineer22 Update README.md

- boards Add files via upload
- common Add files via upload
- remote Add files via upload
- src Add files via upload
- CMakeLists.txt Add files via upload

<https://github.com/iotengineer22/zephyr-ipc>

Zephyr公式のIPCのサンプル



zephyr / samples / subsys / ipc / ipc_service / icmsg

khoa-nguyen-18 and aescolar samples: subsys: ipc: Update SRAM size f... 0541a22 · 3 weeks ago

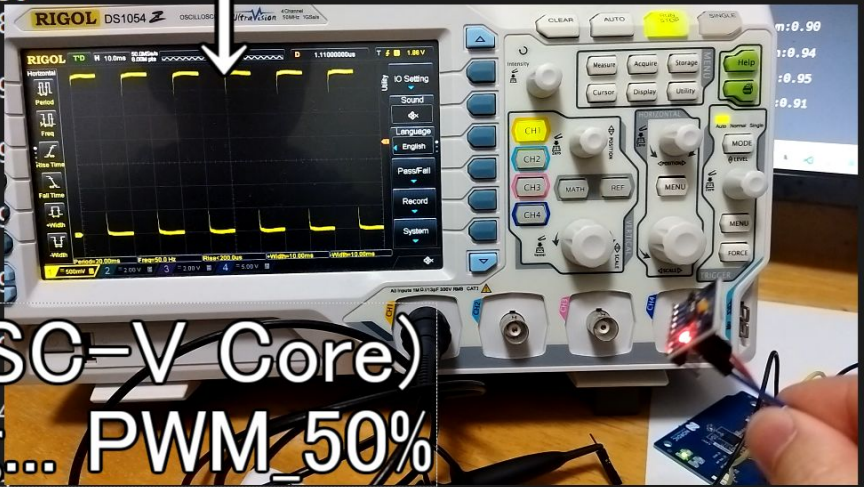
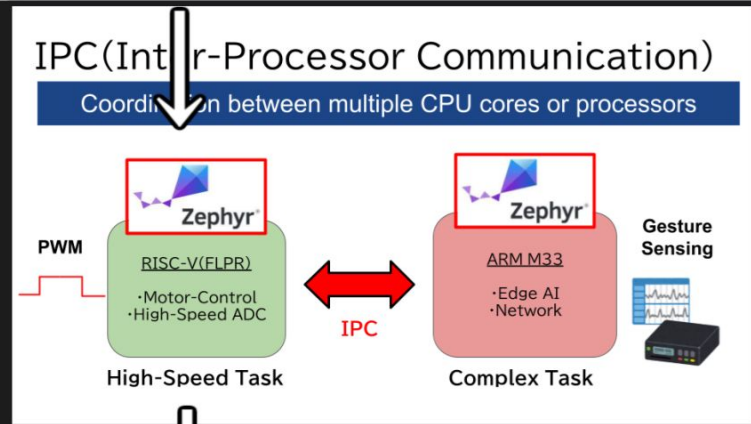
Name	Last commit message	Last commit date
..		
boards	samples: subsys: ipc: Update SRAM size for ek_ra8p1	3 weeks ago
common	samples: ipc: icmsg: Clean up the sample	4 years ago
pytest	samples: subsys: ipc: ipc_service: icmsg: Skip test if n...	2 months ago
remote	samples: subsys: ipc: Update SRAM size for ek_ra8p1	3 weeks ago
src	ipc_service: icmsg: Add "unbound" functionality	last year
CMakeLists.txt	samples: ipc: icmsg: Extend support for nRF54LM20A	2 months ago

https://github.com/zephyrproject-rtos/zephyr/tree/main/samples/subsys/ipc/ipc_service/icmsg

IPCデモ動画(nRF54L15-DK)

https://youtu.be/ACCoVr_i27I

```
00:12:02.225,666] <inf> RISC_V: PWM: 0%
00:12:04.148,195] <inf> RISC_V: Received: idle:0.96
00:12:04.148,208] <inf> RISC_V: PWM: 0%
00:12:06.068,762] <inf> RISC_V: Received: idle:1.00
00:12:06.068,775] <inf> RISC_V: PWM: 0%
00:12:07.995,347] <inf> RISC_V: Received: idle:0.99
00:12:07.995,361] <inf> RISC_V: PWM: 0%
00:12:09.918,923] <inf> RISC_V: Received: idle:1.00
00:12:09.918,936] <inf> RISC_V: PWM: 0%
00:12:11.842,459] <inf> RISC_V: Received: idle:1.00
00:12:11.842,472] <inf> RISC_V: PWM: 0%
00:12:13.766,010] <inf> RISC_V: Received: idle:1.00
00:12:13.766,023] <inf> RISC_V: PWM: 0%
00:12:15.689,546] <inf> RISC_V: Received: circle:0.50
00:12:17.612,068] <inf> RISC_V: Received: updown:0.8
00:12:17.612,085] <inf> RISC_V: PWM: 100%
00:12:19.537,633] <inf> RISC_V: Received: updown:0.9
00:12:19.537,649] <inf> RISC_V: PWM: 100%
00:12:21.463,197] <inf> RISC_V: Received: updown:0.9
00:12:21.463,214] <inf> RISC_V: PWM: 100%
00:12:23.386,778] <inf> RISC_V: Received: updown:0.9
00:12:23.386,795] <inf> RISC_V: PWM: 100%
00:12:25.309,332] <inf> RISC_V: Received: updown:0.9
00:12:25.309,349] <inf> RISC_V: PWM: 100%
00:12:27.230,916] <inf> RISC_V: Received: flick:0.8
00:12:27.230,933] <inf> RISC_V: PWM: 100%
00:12:29.155,426] <inf> RISC_V: Received: flick:0.8
00:12:29.155,442] <inf> RISC_V: PWM: 100%
00:12:31.077,947] <inf> RISC_V: Received: flick:0.8
```

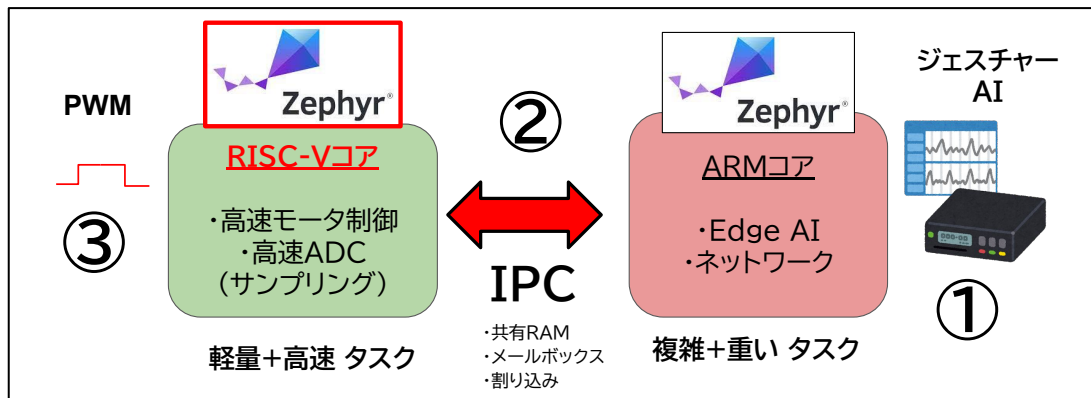


(Logging RISC-V Core)
Flick testing... PWM 50%

IPCでテスト成功

ログ見ても、IPCでコア間のデータ転送→OK

- ①ARM側でジェスチャーAI処理
- ②IPCでデータ共有
- ③RISC-V側のPWMを変更



```
[00:12:42.613,297] <inf> RISC_V: Received: updown:0.91
[00:12:42.613,313] <inf> RISC_V: PWM: 100%
[00:12:44.539,900] <inf> RISC_V: Received: updown:0.80
[00:12:44.539,917] <inf> RISC_V: PWM: 100%
[00:12:46.464,435] <inf> RISC_V: Received: idle:0.91
[00:12:46.464,449] <inf> RISC_V: PWM: 0%
[00:12:48.389,982] <inf> RISC_V: Received: idle:0.99
[00:12:48.389,996] <inf> RISC_V: PWM: 0%
```

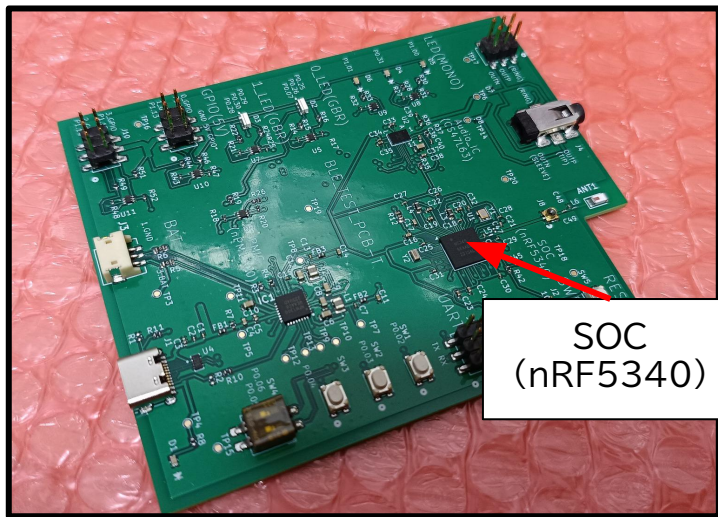
```
[00:12:44.539,739] <inf> host: Predictions (DSP: 40 ms, Classification: 100%)
[00:12:44.539,764] <inf> host: circle: 0.00391
[00:12:44.539,776] <inf> host: flick: 0.00391
[00:12:44.539,787] <inf> host: idle: 0.19531
[00:12:44.539,799] <inf> host: updown: 0.79687
[00:12:44.539,899] <inf> host: Sent inference result: updown:0.80
[00:12:46.464,274] <inf> host: Predictions (DSP: 39 ms, Classification: 100%)
[00:12:46.464,299] <inf> host: circle: 0.00781
[00:12:46.464,311] <inf> host: flick: 0.02344
[00:12:46.464,323] <inf> host: idle: 0.91016
[00:12:46.464,335] <inf> host: updown: 0.05469
[00:12:46.464,435] <inf> host: Sent inference result: idle:0.91
```

他のIPC応用例

実は知らずにIPC使っていたケース

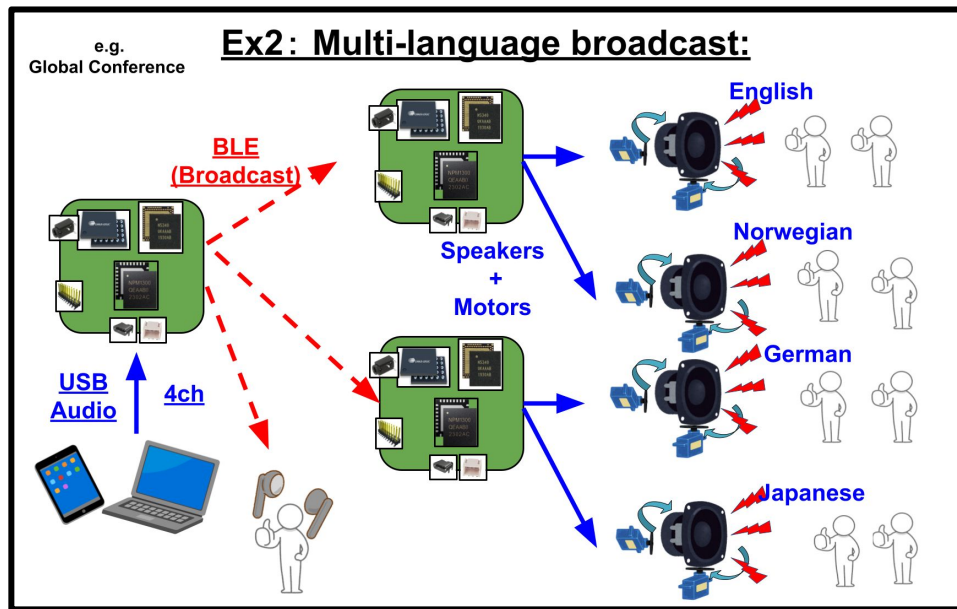
Bluetooth LE Audio対応のトランスミッター・スピーカー

*BLE…Bluetooth Low energy



去年BluetoothのAudio開発コンテスト
プロジェクトURL:

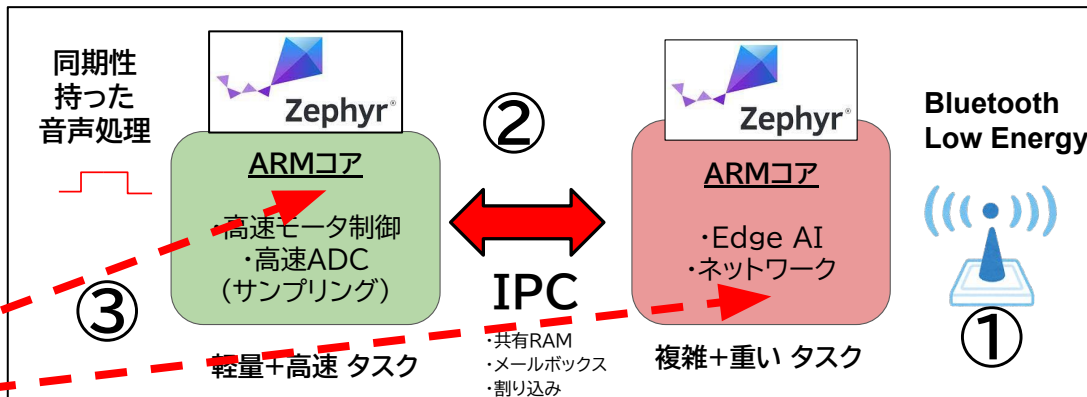
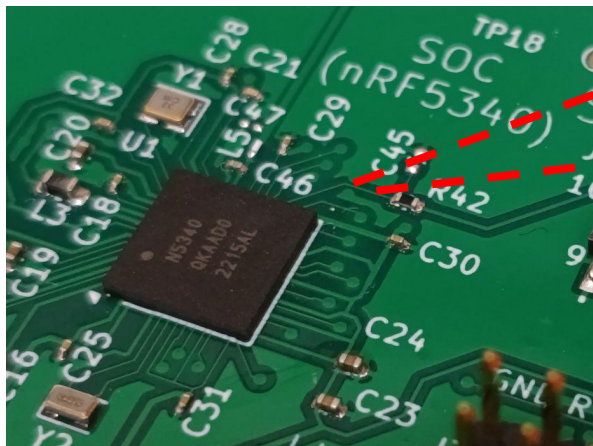
<https://www.hackster.io/iotengineer22/adaptive-directional-ble-audio-speaker-2d892d>



nRF5340 (ARM_M33 のデュアル + IPC)

「ネットワーク処理コア」と「音声処理のコア」を分けていた

- Nordic Wireless
SOC nRF5340
*ARM M33が2個内蔵



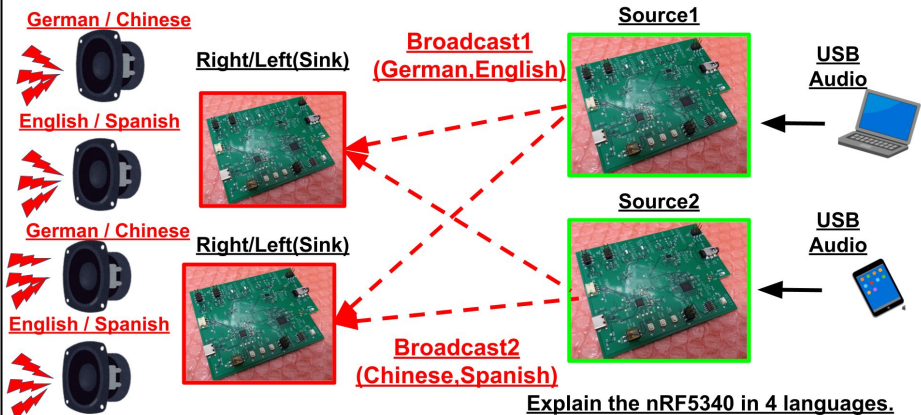
- ① ARM側でBluetooth LE処理
- ② IPCで音声データ共有
- ③ ARM側で同期性を持った音声出力

Bluetooth LE Audio

高い同期精度が必要 →デュアルコアでの分散処理 + IPC

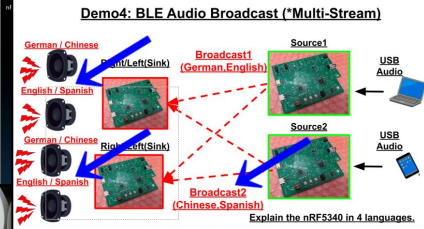
- 今までのBluetoothには無い機能…Unicast/Broadcastができる
→(同期合わせた)特定オーディオの一斉送信

Demo4: BLE Audio Broadcast (*Multi-Stream)



The screenshot shows a terminal window with logs for a BLE Audio Broadcast. The logs include the following information:

```
*** Booting rpf Connect SDK v3.0.0-38f64578e2 ***
** 0x100_220000_05_v4.0.0-38f64578e2 **
[0] 00:00:00.257,900 <--> FW info:
  nRF5340:
    NCS: base version: 1.0.0
    Time: 00:00:00.257,900
    FW: info:
      <--> FW info:
      <--> FW info: Compiled for: nrf5340 device
      <--> bt_mgmt_ttr_cfg controller: Softdevice: Version 6.0 (Dede), Revision B424
      <--> bt_mgmt_local Identify addr: 00:00:00:00:00:00 (random)
      <--> audio_usb Ready for USB host to send/receive
      <--> broadcast_source: Broadcast source 0x20000840 started
      <--> main: Broadcast source: nRF5340 BROADCASTER started
      <--> bt_mgmt_local Identify addr: C4:5E:30:17:10:0A (random)
      <--> main: BT adv ready
      <--> bt_mgmt_adv Advertising successfully started
      <--> le_audio: LC3 codec config for source:
      <--> le_audio: Frequency: 48000 Hz
      <--> le_audio: Duration: 20000 us
      <--> broadcast_source: Broadcast source 0x20000840 started
      <--> le_audio: Channel allocation: 8x2
      <--> le_audio: Frames per frame: 128 (96000 bps)
      <--> le_audio: Octets per frame: 128 (96000 bps)
      <--> le_audio: Frames per SDU: 1
      <--> le_audio: Frames per 500: 1
      <--> le_audio: USB BT first data received.
```



Checking Sound Synchronization.
→Good!

まとめ

まとめ

Zephyr(RTOS)で コア間通信(IPC)を実装+遊べた！



- Zephyrがコア間通信を強くサポートされていたのを学べた
(複数/異種のコアを持つSoCに対応)
- Zephyrが様々なアーキテクチャに対応
(ARM+RISC-Vのようなコア間でも通信できた)
- 分散処理したり、厳密なRTOS制御したい場合に使える
(EdgeAIや厳密なネットワーク同期制御など)

IPC (Inter-Processor Communication)

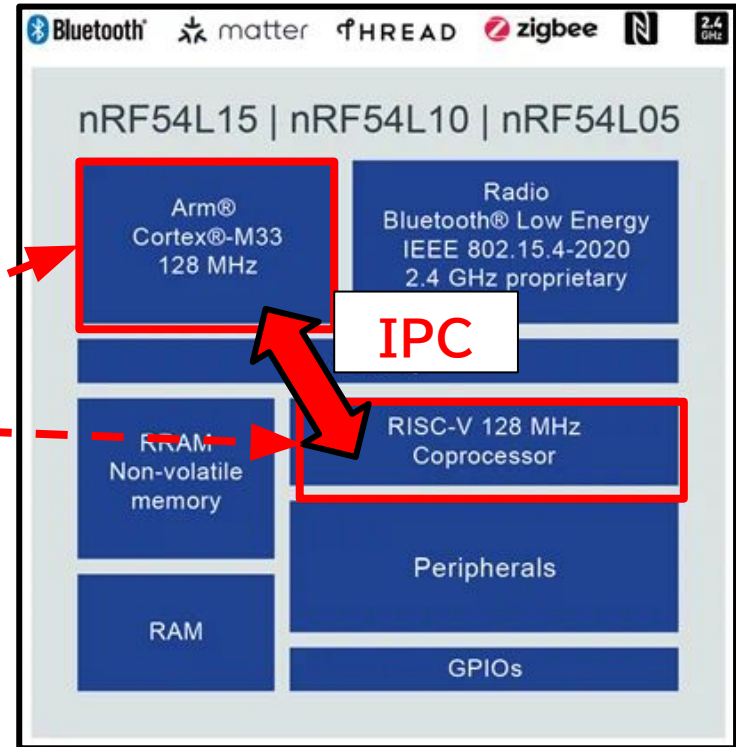
Coordination between multiple CPU cores or processors

■ Nordic Wireless SOC nRF54L15



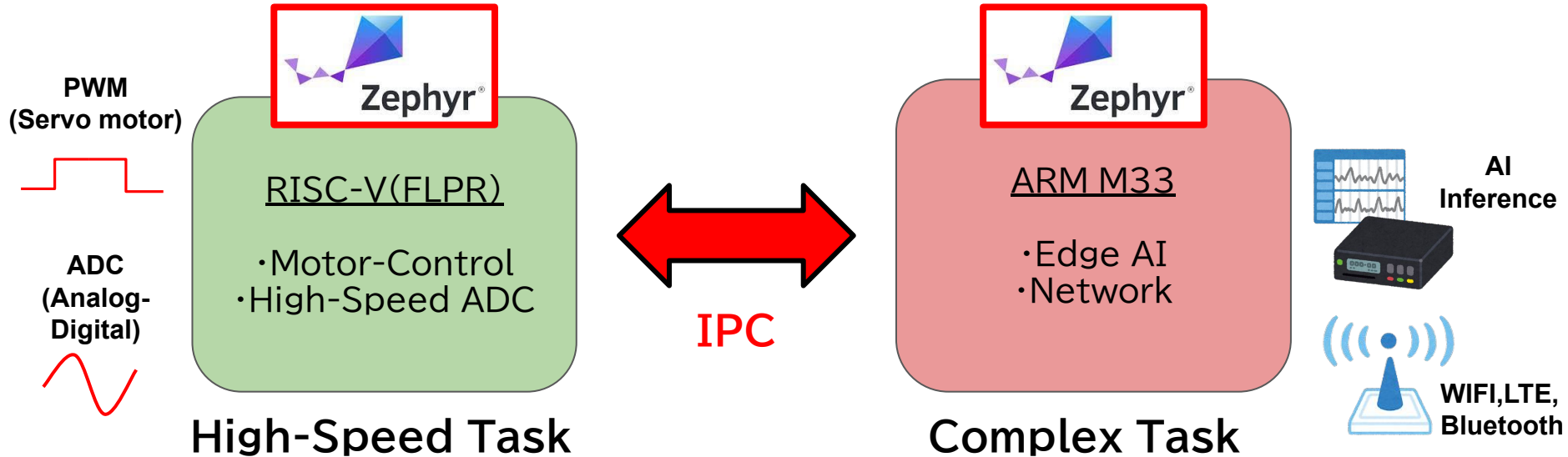
ARM_M33

RISC-V



IPC (Inter-Processor Communication)

Coordination between multiple CPU cores or processors



IPC(Inter-Processor Communication)

Coordination between multiple CPU cores or processors

