# RISC-Vベクタ拡張を用いた宇宙物体軌道計算のベクトル化

北九州市立大学　国際環境工学部　情報システム工学科　4年
山崎進研究室　　谷　賢太郎

## 背景・目的

地球を周回する，人工衛星やスペースデブリを含む宇宙物体の数は年々急激に増加している．

このような膨大な宇宙物体の軌道を正確かつ高速に計算することは重要である．

本研究では，衛星軌道の算出を行うアルゴリズムについての調査とそのベクトル化を行う．

## 使用する書籍

著者：　Montenbruck Oliver
　　Gill Eberhard
タイトル：
**『Satellite Orbits : Models, Methods and Applications』**
出版社：Springer
発売日：2013-10/3

## 提案内容

『Satellite Orbits』に基づく衛星軌道の算出を行うアルゴリズムについて，ベクトル化を行う．
（スペースデブリと人工衛星の軌道は同じように算出できるため，衛星軌道の算出は即ち宇宙物体の軌道の算出となる）

## 新規性

2023年6月現在，「Satellite Orbits」と「RISC-V」をキーワードにした先行研究は見つかっていない．

## 実証方法

『Satellite Orbits』で示されているプログラムと，ベクトル化したプログラムを比較する．

## これからの計画

『Satellite Orbits』の閲読，衛星軌道の算出を行うアルゴリズムの調査

## RISC-Vについて

・命令セットアーキテクチャ（ISA）
・オープンソース
・ISAがシンプル

## 使用する書籍について

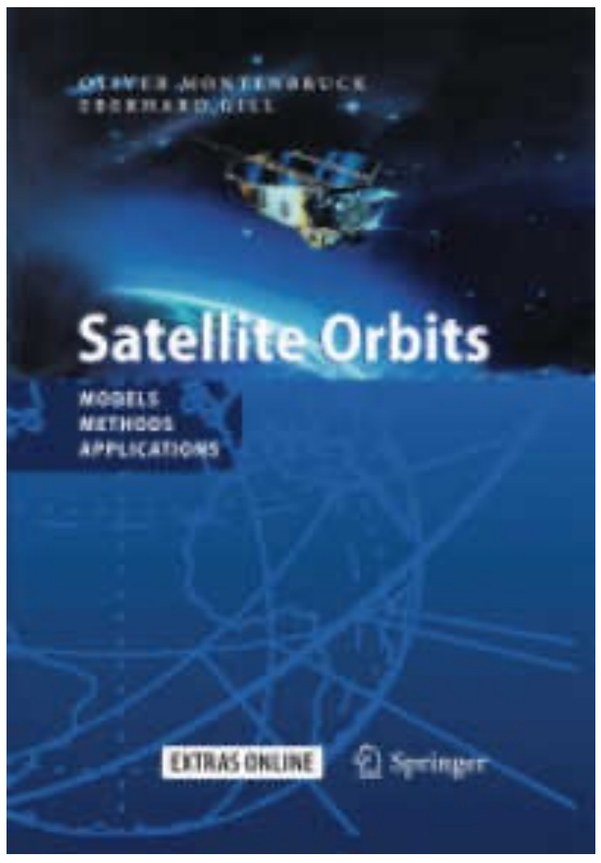
**『Satellite Orbits : Models, Methods and Applications』**

衛星軌道計算に関する公式，C++のプログラム例が記載されている．

この書籍を用いて，衛星軌道の算出を行うアルゴリズムについての調査と，プログラム例を用いたベクトル化と比較を行う．

例）$\ddot{\boldsymbol{r}} = -\frac{GM}{r^3}\boldsymbol{r}$

$\ddot{\boldsymbol{r}}$ ：衛星の加速度ベクトル
G：万有引力定数
M：地球の質量
r ：衛星と地球間の距離
**r** ：衛星と地球間の位置ベクトル




出典URL：
https://www.kinokuniya.co.jp/f/dsg-02-9783642635472

SORACOM

# もうサーバーは作らない！<br>手間も費用もかからない「サーバーレス」の<br>選び方と始め方のコツ

Aug. 31, 2023

SWEST25 / EmbLT

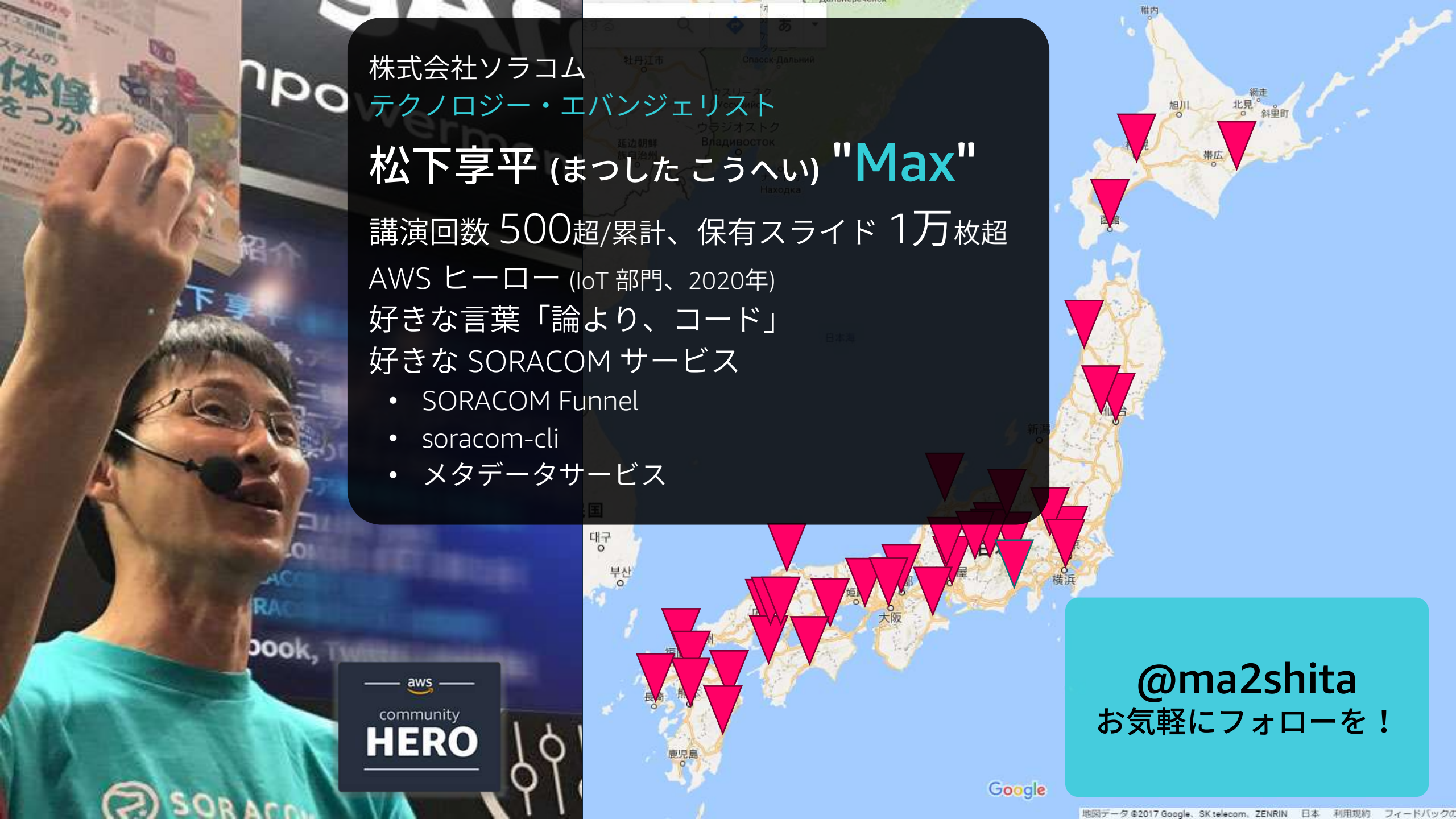
株式会社ソラコム

テクノロジー・エバンジェリスト

松下 享平 (Max / @ma2shita)

SORACOM

# #SWEST25
# #クラウド完全に理解した
# #SORACOM


株式会社ソラコム
テクノロジー・エバンジェリスト
松下享平 (まつした こうへい) "Max"
講演回数 500超/累計、保有スライド 1万枚超
AWS ヒーロー (IoT 部門、2020年)
好きな言葉「論より、コード」
好きな SORACOM サービス
• SORACOM Funnel
• soracom-cli
• メタデータサービス
aws
community
HERO
@ma2shita
お気軽にフォローを！
Google


SORACOM
Max

# IoT バックエンド構築の課題

SORACOM

## 準備

仮想サーバーを立ち上げてから、OS、あとは PHP + Laravel をインストールして API と変換処理を実装した後、ストレージに MySQL を設定して…

**楽しいけれど、お金にならない時間！**

## 非機能要件の実装

MQTT 対応？認証処理？
ワークロード調整？ストレージ見積？

**面倒なうえに、お金にならない時間！**

## 運用や費用

モニタリング？スケールアウト/イン？
データが流れていない時のコスト？

**手間がかかるのに、だれもお金を払ってくれない！**

SORACOM

IoT バックエンドは
不可欠

サーバー作りや
運用は大変

SORACOM

ハードウェアもソフトウェアも
サーバーも利用する

# 作らずに、創る

自前主義からの脱却

# マネージドサービス、サーバーレス、SaaS

SORACOM

## IaaS
仮想サーバー

OS より上を
ユーザーが
運用するもの

## マネージドサービス
《IaaS との異なる点》 OS レイヤは抽象化・隠ぺい化されている

クラスターや AZ 配置を
ユーザーが設定するもの

### サーバーレス
アプリ(コード)実行環境が
提供されるもの

API を通じて “機能” が
提供されるもの

## SaaS
アプリケーション自体が
提供されるもの

**自由度は高いが、広いスキルが必要**
**開始までの時間もかかる**

**自由度は低いが、学ぶことが少ない**
**すぐに始められる**

AZ; アベイラビリティ・ゾーン。データセンターとも言い換えられる。
IaaS; Infrastructure as a Service
SaaS; Software as a Service

# サーバーレス、初めての一歩は FaaS
# — Function as a Service

SORACOM

《AWS》
AWS Lambda

《Microsoft Azure》
Azure Functions

《Google Cloud》
Cloud Functions

# 1分40秒で作る、HTTP REST サーバー

SORACOM

HTTP GET →

```
❯ http https://46s37swh3uwsgghx7bcv4rtlhi0dmwiu.lambda-url.us-east-1.on.aws/
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 42
Content-Type: application/json
Date: Wed, 30 Aug 2023 10:01:47 GMT
X-Amzn-Trace-Id: root=1-64ef138b-01c4732a11edba5138b73e1d;sampled=0;lineage=38004923:0
x-amzn-RequestId: 1a3900cf-0daf-41c8-b3c5-4b9b6d649ded

"Hello from Lambda! Welcome serverless !!"
```

レスポンス →

# コンソールのホーム 情報

デフォルトレイアウトにリセット ＋ ウィジェットを追加

## 最近アクセスしたサービス 情報

- Lambda
- WorkSpaces
- Amazon EventBridge
- CloudWatch
- S3
- EC2
- DynamoDB
- IoT Core
- API Gateway
- IAM
- サポート
- Amazon WorkDocs

すべてのサービスを表示

## AWS へようこそ

**AWS の開始方法**
AWS を最大限に活用するために基礎を学び、有益な情報を見つけましょう。

**トレーニングと認定**
AWS のエキスパートから学び、スキルと知識を深めましょう。

**AWS の最新情報?**
新しい AWS のサービス、機能、およびリージョンについてご覧ください。

## AWS Health 情報

未解決の問題

0 過去 7 日間

スケジュールされている変更

## コストと使用状況 情報

SORACOM

# ね？簡単でしょ！
# 作らずに、創る
## 自前主義からの脱却

# 「ServerlessDays Tokyo 2023」で！

SORACOM

/serverless/ Tokyo 2023 DAYS

23-24 SEPTEMBER

Three Tracks. Two Days. One Community

9/23 (土)
@東京/日比谷


https://tokyo.serverlessdays.io


SORACOM
Max
明日は「ChatGPT × IoT」の話をするよ！
明日も楽しみましょう！！

# IoT の「つなぐ」を簡単に

You Create. We Connect.

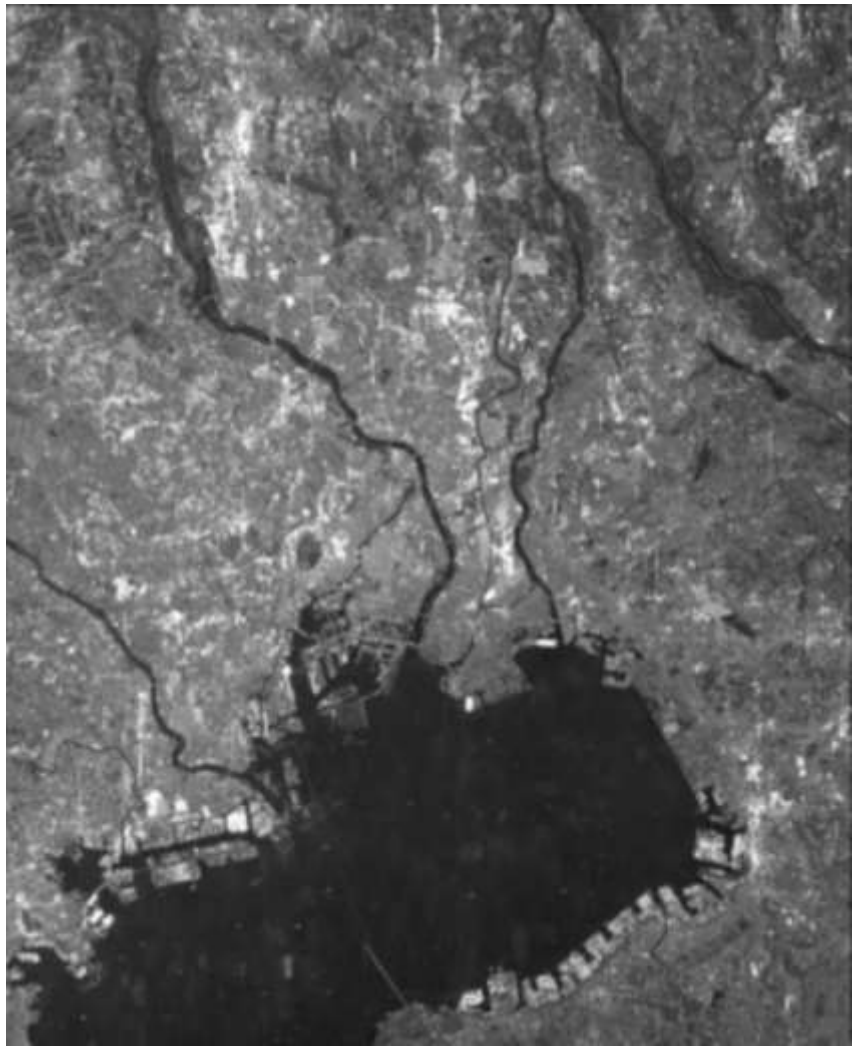
SORACOM

# タイル状に分割したSAR画像のElixirを用いた並列・分散処理

北九州市立大学
関翔太

# 背景

- SAR衛星：人工衛星から電波を送信し，地表から跳ね返ってきた電波を受信することで，地表の画像を生成する技術[1].
- SAR衛星では電波が使われているので，昼でも夜でも明るさに関係なく，雲をすり抜けて地表面を観察することができる.



提供：ALOS-2 PALSAR-2サンプルプロダクト（校正済み）
2014年8月29日東京 高分解能3mモード（HH偏波）（JAXA）

[1] リモート・センシング技術センター. SAR リモートセンシング基礎セット.

- SARのデータは農業や防災分野などで活用されている [2, 3].
- SARでは，受信した電波のデータを画像に変換する処理（再生処理）に時間がかかる.

→ 再生処理を高速化

→ SARのデータ活用がより有用に

特に，防災分野ではリアルタイム性が重要

[2] 山崎進. SAR 衛星によるリアルタイム土砂災害情報提供システムの実現に向けた衛星画像の分散並列処理の実現. 第 66 回宇宙科学技術連合講演会, 11 2022.
[3] R.J.Brown Brisco B and M.Manore. Early season crop discrimination with combined SAR and TM data. Canadian J. Remote Sens, Vol. 5, pp. 98–102, 1989.

# 研究目的

- SAR画像をそのまま再生処理する
- SAR画像を並列・分散処理を用いて再生処理する

以上の２つの場合を比較して，どれだけ実行速度が速くなるかを明らかにする．

# アプローチ

並列・分散処理の機能の優れたプログラミング言語Elixirを用いて[3],

- SAR画像をそのまま再生処理する
- SAR画像を並列・分散処理を用いて再生処理する

以上の２つのプログラムを作成し，実行速度を計測する.

[3] Elixir Team. Elixir: Elixir is a dynamic, functional language for building scalable and maintainable applications. , 2012. https://elixir-lang.org.

# 並列・分散処理のアルゴリズム

①SAR画像をタイル状に分割

②分割タイルごとに並列・分散処理を用いて
　再生処理を行う

③再生処理後のタイル画像を結合し，１枚の
　SAR画像に戻す

SWEST25 EmbLT 2023/8/31

# NervesとSpresenseをHostIFで通信してみた

パーソルクロステクノロジー株式会社
第1技術開発本部　第4設計部　設計2課　阿部耕二

# 目次

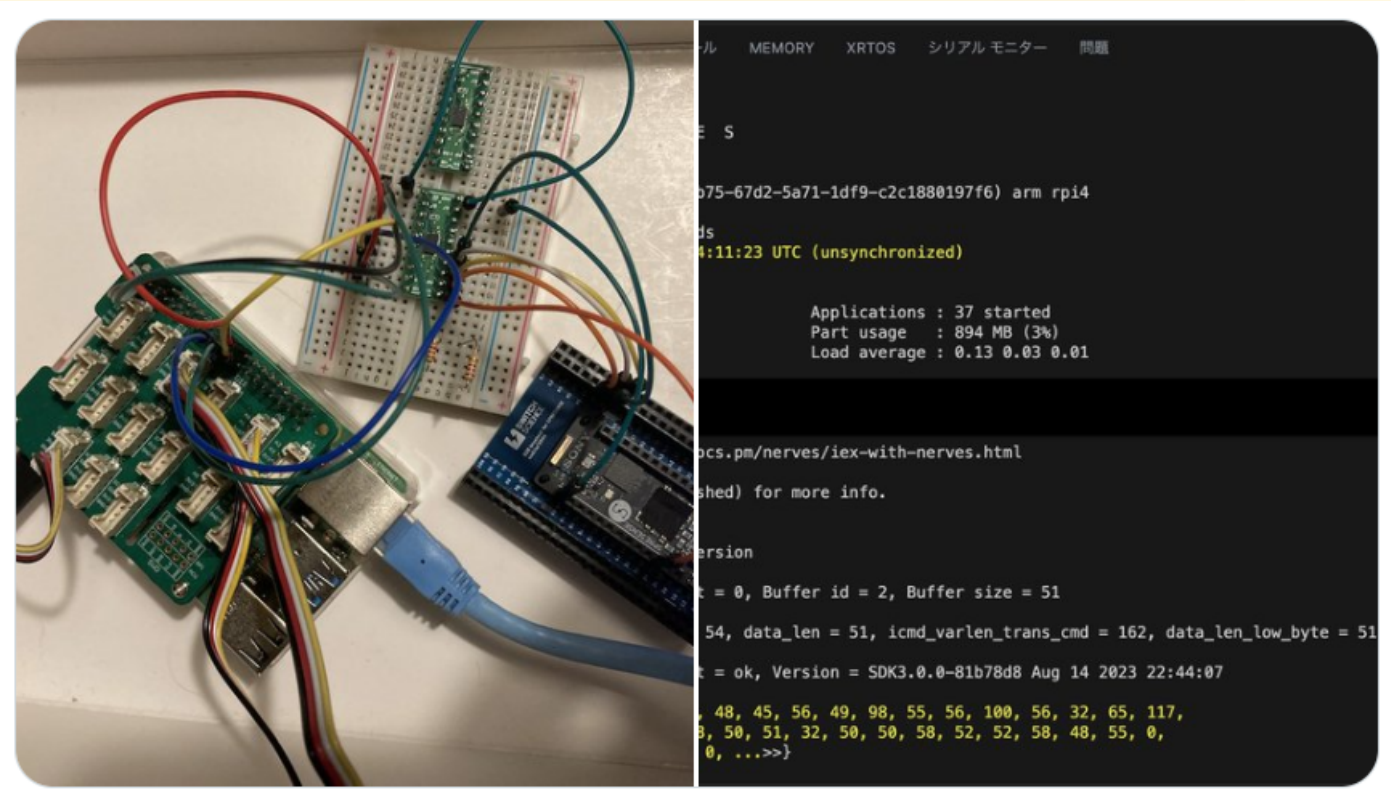
- 自己紹介
- 概要
- 背景
- Spresense, HostIFとは?
- 開発環境
- 動作確認結果
- ソースコード解説
- 感想

# 自己紹介

- 名前: 阿部　耕二（あべ　こうじ）
- 所属: パーソルクロステクノロジー株式会社
  第1技術開発本部 第4設計部 設計2課
- 医療機器の組込みソフトウェア開発。C言語。
- 趣味: 宇宙開発（リーマンサットプロジェクト広報メンバー）
- LAPRASポートフォリオ: https://lapras.com/public/k-abe
- Twitter: @juraruming

# 概要

Nerves（ターゲットはラズパイ4）とSpresenseがHostIFでデータ受信できた。



# 背景

- Spresenseは以前から興味をもっており、技術書を書いたりしていた。
- リーマンサットプロジェクトの衛星RSP-02の記事でSpresenseのHostIF機能（※）を知る。
  ※Spresense外部のホストICに通信インタフェース機能を提供する
- **Spresenseをシステムのメインマイコンではなく、サブマイコンで使う**という選択肢・可能性にワクワクした。
- HostIF機能のホストはモダンな開発環境に憧れがあり、Elixirも学びたいという理由でNervesにしたいと考えた。

# Spresense, HostIFとは?

- Spresenseとは?

- ソニーが開発したボードコンピュータ
- Arduino UNOではパワー不足、ラズパイではハイスペックすぎる、という場合にマッチするかもしれない。
- 以前、こちらにSpresenseの魅力を書きました
  Spresenseの魅力を語る!!!

■ **Spresense SDK チュートリアルより引用**
**https://developer.sony.com/spresense/development-guides/sdk_tutorials_ja.html**

- **GPS**
- **Audio（再生・録音）**
- **Camera**
- **LTE（LTE拡張ボードでクラウド接続のサンプルコードもあり）**
- **SMP（マルチコア）**
- **その他、いろいろ試せるチュートリアルがある!!!**

- スタートガイド (IDE 版)

## 1.1. SDK examples

| カテゴリ | Example名 | 説明 |
|---|---|---|
| Peripheral Driver | adc_monitor | A/Dコンバータを使用してアナログ入力値を表示するサンプルです。 |
| GPS | gnss | GNSS機能を用いて位置情報を取得するサンプルです。 |

## HostIFとは?

- Spresense外部のホストICとの通信インタフェース機能を提供
  通信IFはI2CかSPIを選択可能
- 1KByteの通信バッファをサイズ・転送方向を設定可能
- **バッファに排他ロック機構**がある
  ホストとSpresenseは安全にデータ送受信が可能!!!

- HostIF ホスト受信

図は下記より引用

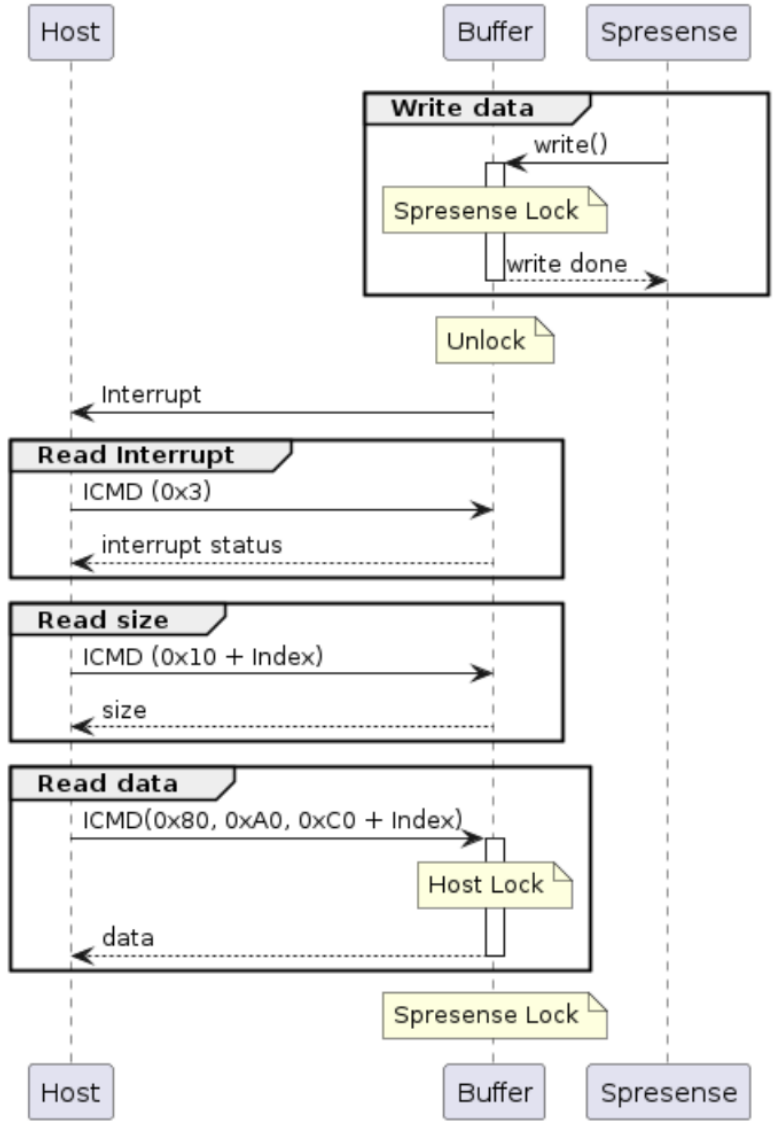
Spresense SDKの開発
-> 開発ガイド
-> 5.15.3.5.1. HOST 受信シーケンス

https://developer.sony.com/spresense/development-guides/sdk_developer_guide_ja.html#_host_受信シーケンス

5.15.3.5. Host 通信シーケンス

5.15.3.5.1. HOST 受信シーケンス

Host が Spresense からのデータを受信するシーケンスを以下に示します。

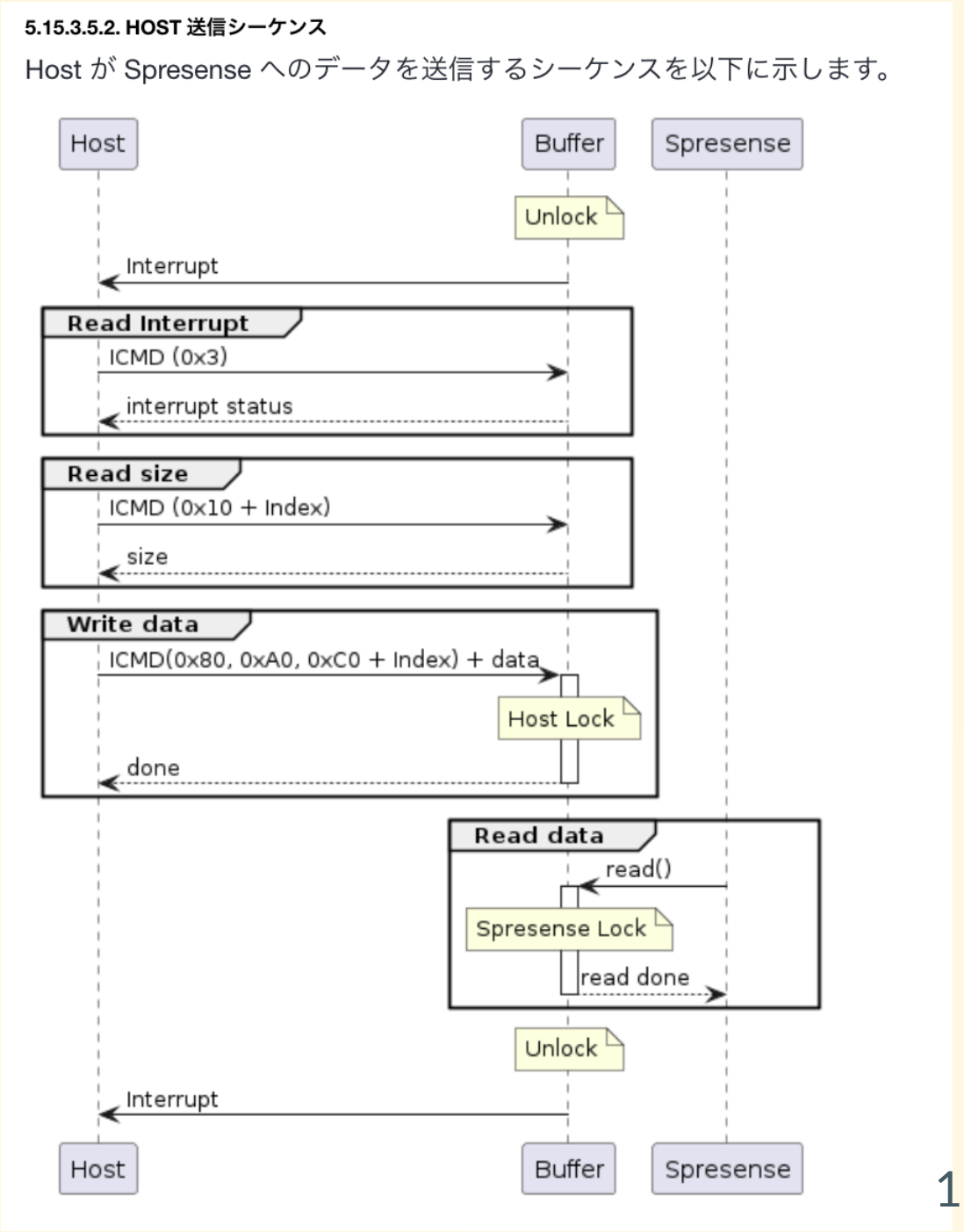


- HostIF ホスト送信

図は下記より引用

Spresense SDKの開発
-> 開発ガイド
-> 5.15.3.5.2. HOST 送信シーケンス

https://developer.sony.com/spresense/development-guides/sdk_developer_guide_ja.html#_host_送信シーケンス

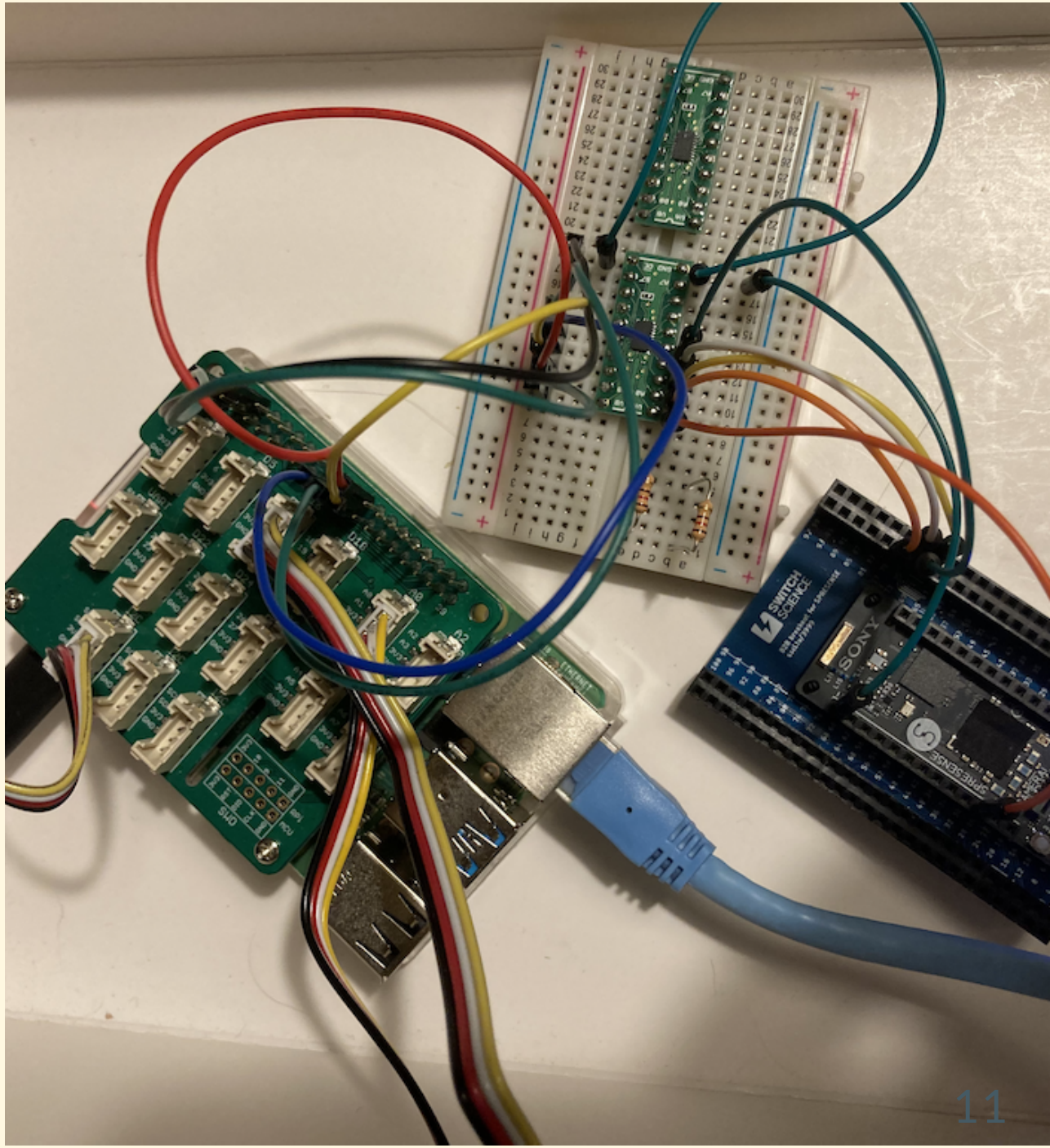


# 開発環境

- ハードウェア

- ラズパイ4B
  HostIFのホストとする。
  Nervesを書き込む。
- Spresense メインボード
  HostIFをホストに提供する。

ホストとSpresenseとの通信IFは**SPI**を選択し動作確認した。



- レベル変換IC
  ラズパイ3.3V, Spresense 1.8Vなのでレベル変換する。
  https://akizukidenshi.com/catalog/g/gK-17062
- SPRESENSE用 100ピンB2Bコネクタ ピッチ変換基板
  SpresenseメインボードにHostIFのIOピンがないので変換基板でHostIFの信号を引き出す
  https://www.switch-science.com/products/3999/#erid10805888

## ソフトウェア

- Nerves

```
$ mix nerves.info
Nerves:           1.10.3
Nerves Bootstrap: 1.11.5
Elixir:           1.15.4

$ elixir -v
Erlang/OTP 26 [erts-14.0.2] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:1] [jit:ns] [dtrace]
Elixir 1.15.4 (compiled with Erlang/OTP 26)
```

- Spresense SDK

```
NuttShell (NSH) NuttX-11.0.0
nsh> uname -a
NuttX  11.0.0 SDK3.0.0-81b78d8 Aug 14 2023 22:44:07 arm spresense
```

ソースコードはこちらです。

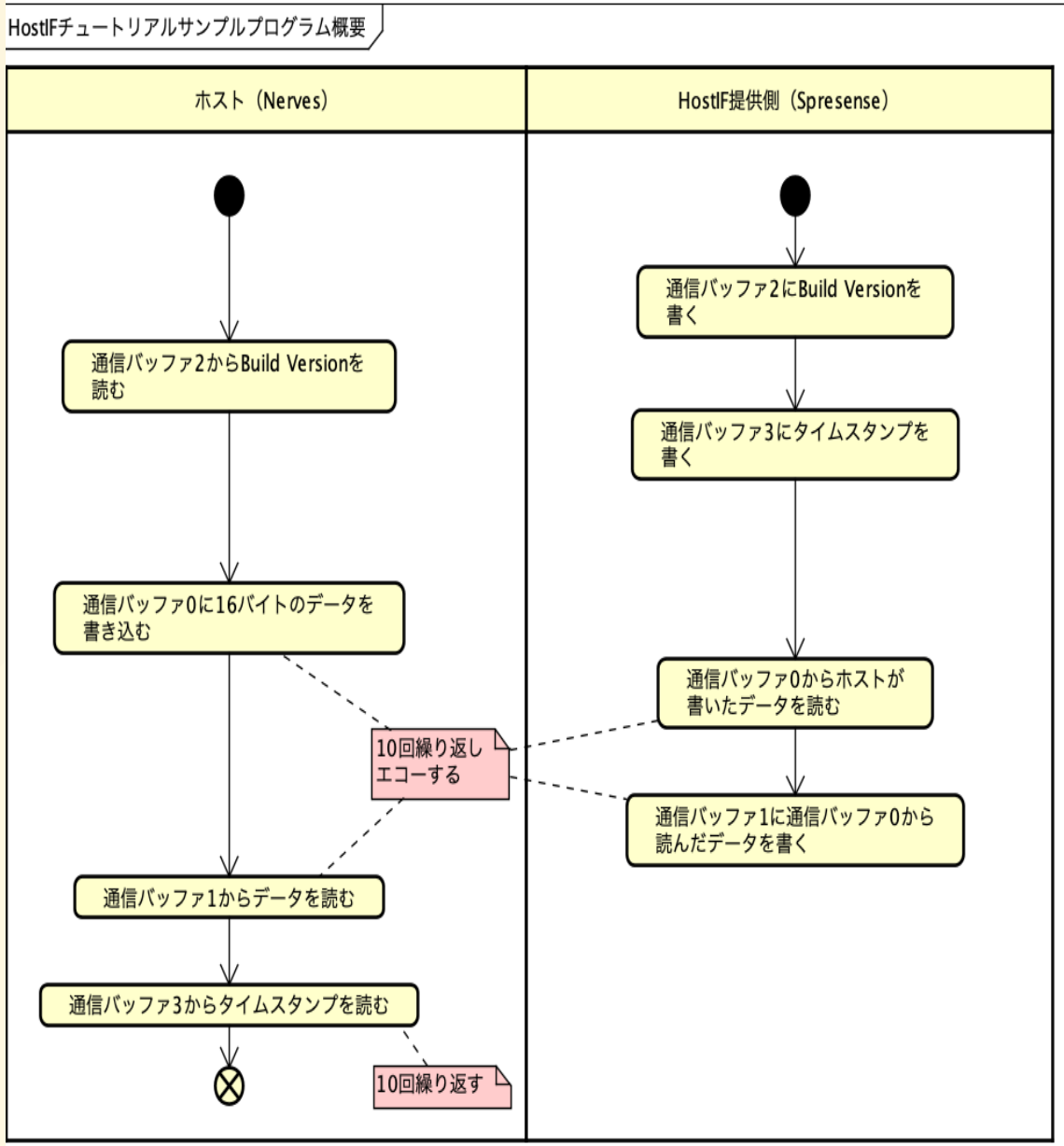
- Nerves（ホスト）:**今回移植したもの**
  https://github.com/grace2riku/nerves_laboratory/blob/main/spresense_hostif/lib/spresense_hostif.ex
- HostIF提供側（Spresense）：チュートリアルのサンプルプログラム
  https://github.com/sonydevworld/spresense/blob/master/examples/hostif/hostif_main.c
- HostIFホスト：移植対象
  https://github.com/sonydevworld/spresense/blob/master/examples/hostif/host_spi_main.c

# 動作確認結果

- HostIFチュートリアルのサンプルプログラムで動作確認した。
  HostIFチュートリアルのサンプルプログラムのホストは
  Spresense、コードはCで実装されている。
- ホストをNerves（ラズパイ4B）で、コードはElixirに移植した。

# HostIFチュートリアルのサンプルプログラムの概要

今回はBuild Versionを読むところまでできた。



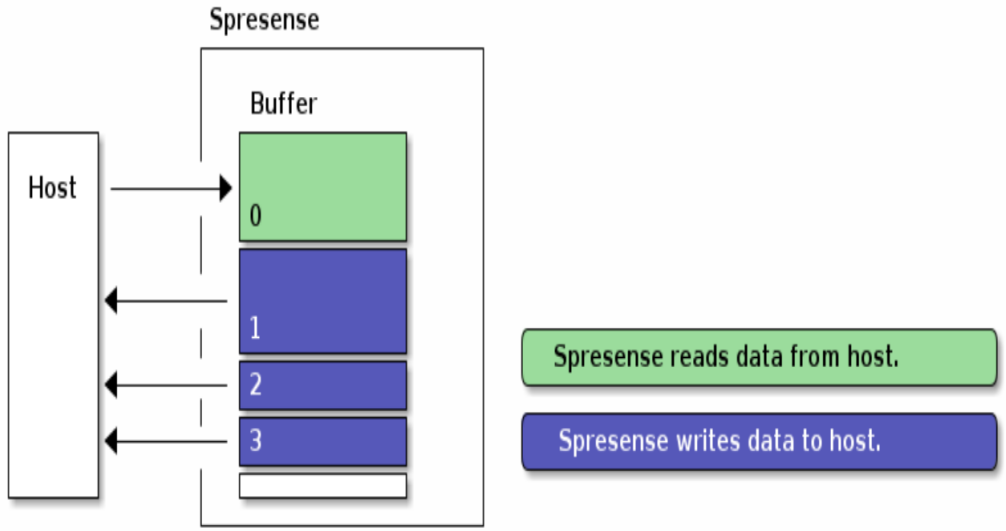
# HostIFチュートリアルのサンプルプログラムの通信バッファの構成

Spresense SDKの開発
-> 開発ガイド
-> 13.1.4.1. 通信バッファ構成

https://developer.sony.com/spresense/development-guides/sdk_tutorials_ja.html#_通信バッファ構成

13.1.4.1. 通信バッファ構成

HostからSpresenseへ送信する通信バッファを1個(Index=0)と、SpresenseからHostへ送信する通信バッファを3個(Index=1,2,3)を作成します。




各通信バッファの用途は以下の通りです。

| Index | Size | Direction | Device filename | Description |
|---|---|---|---|---|
| 0 | 0x100 | Read | /dev/hostifr0 | ループバック受信用(汎用通信用途) |
| 1 | 0x100 | Write | /dev/hostifw1 | ループバック送信用(汎用通信用途) |
| 2 | 0x029 | Write | /dev/hostifw2 | バージョン固定値送信用(固定データ用途) |
| 3 | 0x008 | Write | /dev/hostifw3 | タイムスタンプ送信用(リアルタイム更新データ用途) |

# 実行結果

NervesにSSHログイン後、ログ出力を有効化し
SpresenseHostif.get_version関数を実行する。

```
iex(1)> RingLogger.attach
:ok
iex(2)> SpresenseHostif.get_version

12:37:36.523 [info] cmd_result = 0, Buffer id = 2, Buffer size = 51

12:37:36.523 [info] bufsize = 54, data_len = 51, icmd_varlen_trans_cmd = 162, data_len_low_byte = 51, data_len_high_byte = 64
{:ok,
 <<83, 68, 75, 51, 46, 48, 46, 48, 45, 56, 49, 98, 55, 56, 100, 56, 32, 65, 117,
   103, 32, 49, 52, 32, 50, 48, 50, 51, 32, 50, 50, 58, 52, 52, 58, 48, 55, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, ...>>}

12:37:36.524 [info] cmd_result = ok, Version = SDK3.0.0-81b78d8 Aug 14 2023 22:44:07
```

バージョン情報**SDK3.0.0-81b78d8 Aug 14 2023 22:44:07**が読み出せた。

# ソースコード解説

Nerves（ホスト）：
https://github.com/grace2riku/nerves_laboratory/blob/main/spresense_hostif/lib/spresense_hostif.ex

- get_version関数: バージョン文字列を返す
- get_bufsize関数: HostIF通信バッファのサイズを返す。
  バージョンを格納している通信バッファのサイズを取得している。
- host_receive関数: ホストのデータ受信関数。データが格納されている通信バッファを指定し、データを受信する。

# 感想

- Nerves, Elixirのモダンさを体感できた。
  ライブラリCircuits.SPIを利用することによりSPI通信をサクッと実現できた（SPI通信条件をデフォルトから変更したが問題なく動作した）。
  一番感動したのはSPI受信バイナリデータの**パターンマッチング**の実装部分。
  受信データのステータス、バージョン情報など必要な情報の取り出しをわかりやすく、綺麗に実装できた。
  C言語ばかりやっているためかNerves, Elixir環境のモダンさに余計に感動した。

# spresense_hostif.ex host_receive関数

- SPI.transferでSPI送受信。
  受信データの2byte目がコマンドのステータス（0が正常）
  3byte目以降が受信データ（今回はバージョン情報の文字列）

```
import Bitwise
def host_receive(buffer_id, bufsize, lock) do
  data_len = bufsize - 3
  icmd_varlen_trans_cmd = @icmd_varlen_trans_id + buffer_id
  lock_bit =
    if lock do
      0x40
    else
      0x00
    end

  data_len_low_byte = data_len &&& 0xff
  data_len_high_byte = ((data_len >>> 8) &&& 0x3f) ||| lock_bit

  Logger.info("bufsize = #{bufsize}, data_len = #{data_len}, icmd_varlen_trans_cmd = #{icmd_varlen_trans_cmd}, data_len_low_byte = #{data_len_low_byte}, data_len_high_byte = #{data_len_high_byte}")

  {:ok, ref} = SPI.open("spidev0.0", mode: 1, speed_hz: 800000)

  {:ok, <<_::1-unit(16), cmd_result::1-unit(8), receive_binary_data::bytes>>}
    = SPI.transfer(ref, <<icmd_varlen_trans_cmd::1-unit(8), data_len_low_byte::1-unit(8), data_len_high_byte::1-unit(8), 0xff::size(data_len)-unit(8)>>)

  SPI.close(ref)

  case cmd_result do
    0 -> {:ok, receive_binary_data}
    _ -> {:error, "Error receive data."}
  end
end
```

# 参考資料

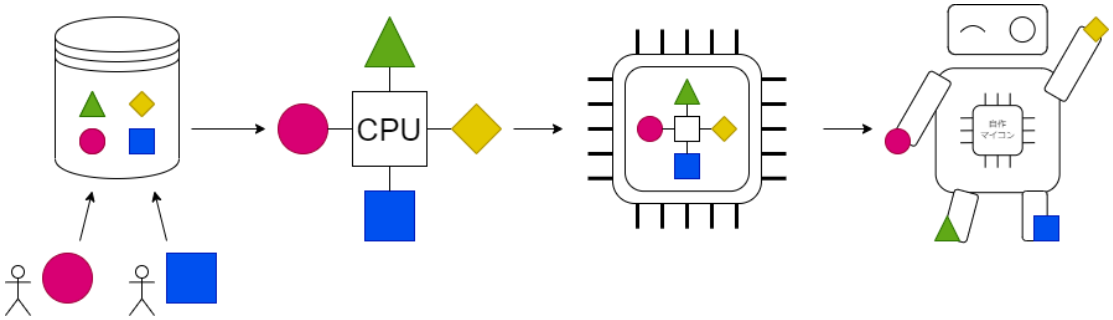
1. Spresense SDK チュートリアル 13. HostIF チュートリアル
   https://developer.sony.com/spresense/development-guides/sdk_tutorials_ja.html#_hostif_チュートリアル
2. Spresense SDK 開発ガイド 5.15. Host Interface (HostIF)
   https://developer.sony.com/spresense/development-guides/sdk_developer_guide_ja.html#_hostif

# ご清聴ありがとうございました🙇

# FPGAを用いたプロトタイピングを容易にするフレームワーク

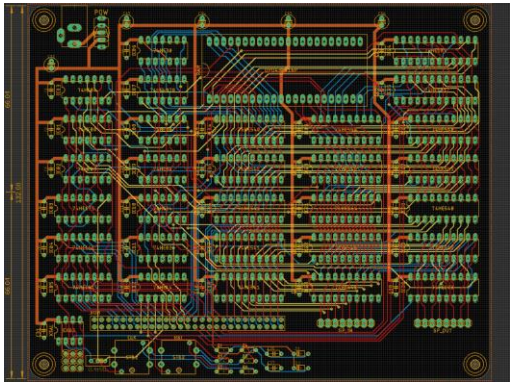
東京大学大学院 新領域創成科学研究科

太田 涼介



# 自己紹介｜太田涼介

リレーコンピュータ

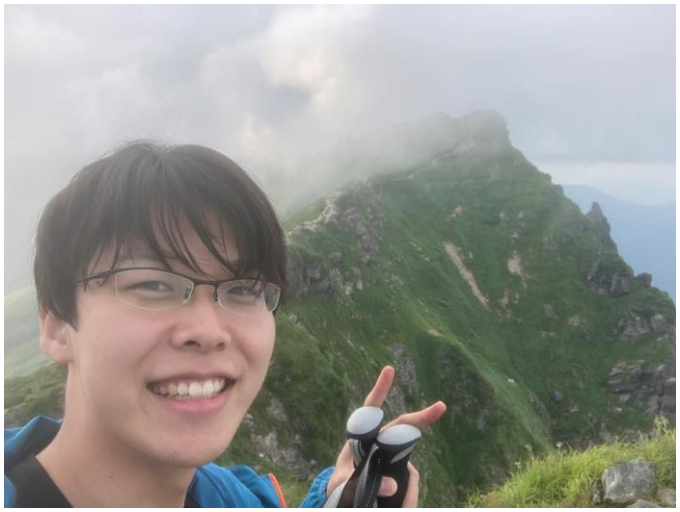


自作CPU


ニコニコ技術部

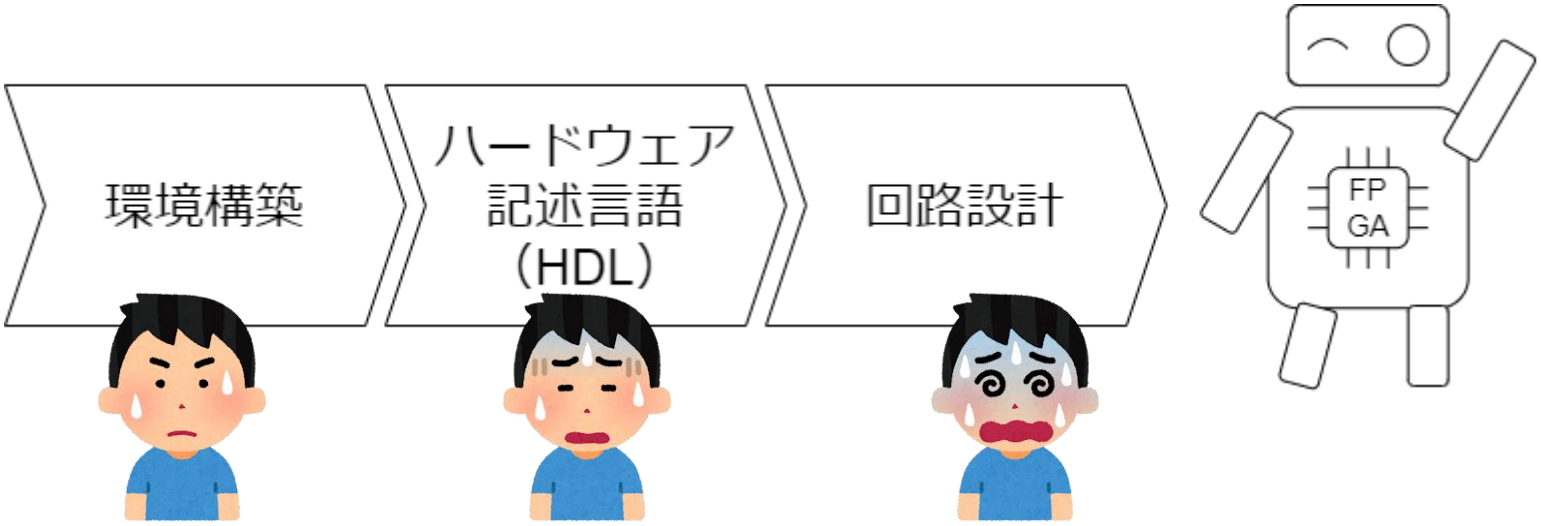
```
; ------------------------------------------------
; OS - タスク管理機能

; struct tcb
@0x0000 .next
@0x0001 .sp
@0x0002 .bp
@0x0003 .ra
@0x0004 .state
#0x10 tcb_size
; enum task_state
#0 task_state_ready
#1 task_state_waiting
#2 task_state_exit

; タスクの初期化
; t0 > initial_state
; t1 > stack_size
; t2 > entry_address
; t0 < task_pointer
os_task_setup:
    loadi t1 task_list
os_task_setup_next:
    mov   t0 t1       ; t0 <= t1
    load  t1 t0 .next ; t0 の次のアイテムのアドレスを、t1に格納する
    brlt  zero t1 os_task_setup_next
    ; t0 が、リストの末尾のアドレスを指している
    ; [task_list] -> A[next=B] -> B[next=null] -> x
    ; 新たに追加するTCBのアドレスを、リストの末尾に連結する
    load  t1 zero heap_top
    store t1 t0 .next
    ; t0 は新たに追加されたTCBのアドレスを指す
```

自作RTOS




Twitter: @kanade_k_1228

# FPGAの難しさ




ベンダ製IDEが
使いにくい

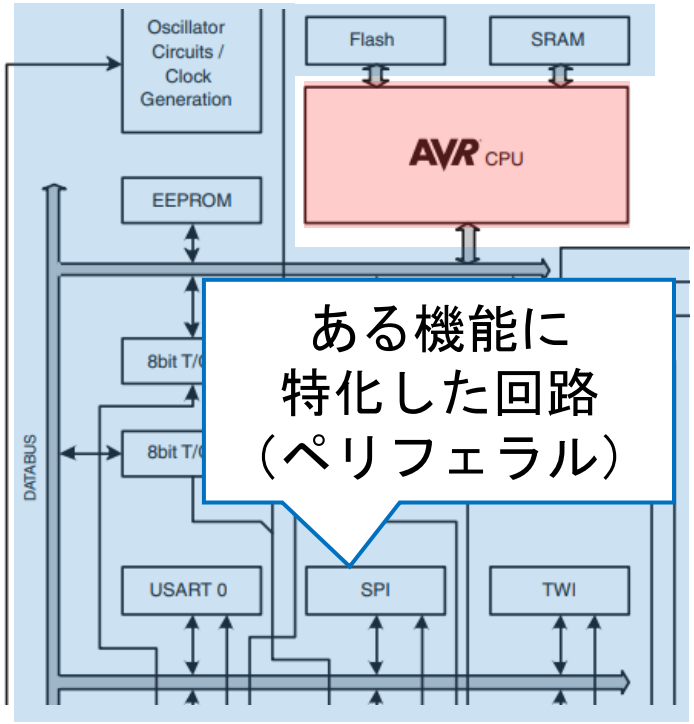
プログラミング
言語と全然
違って難しい

回路わからん
マイコンのが楽！

FPGAを使って「何かを作る」のは難しい

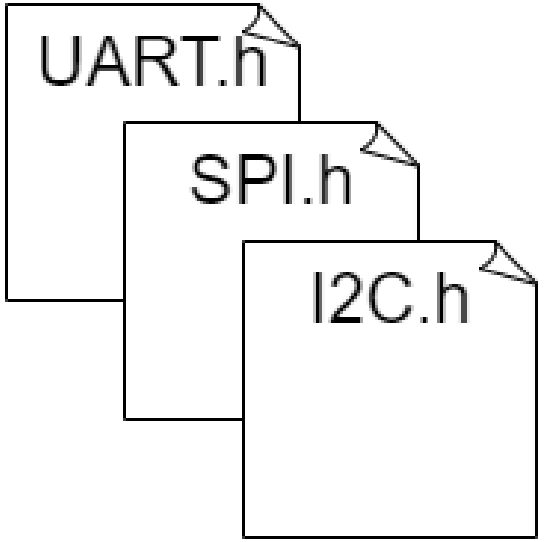
# マイコンの構成

## ハードウェア



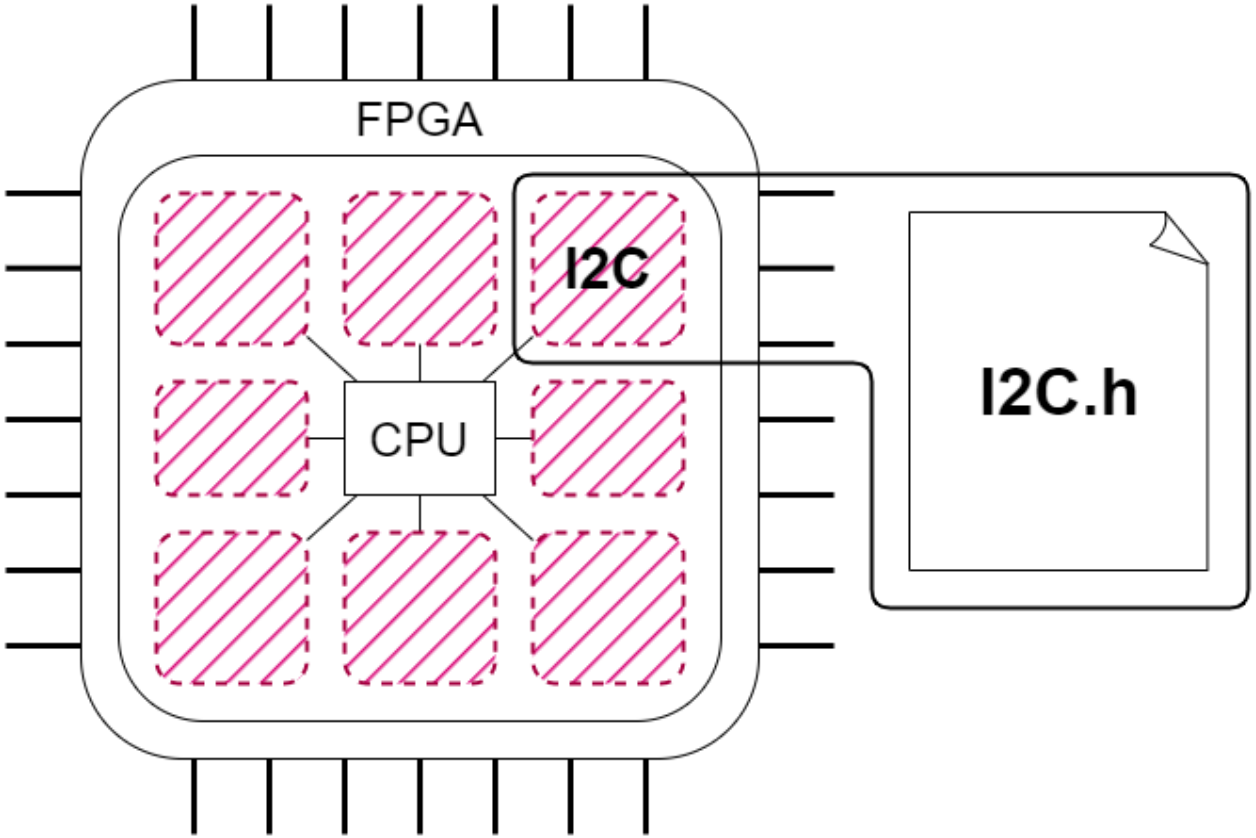

CPUの周りに
周辺回路が接続されている

## ファームウェア




回路を動かすための
ソフトウェア

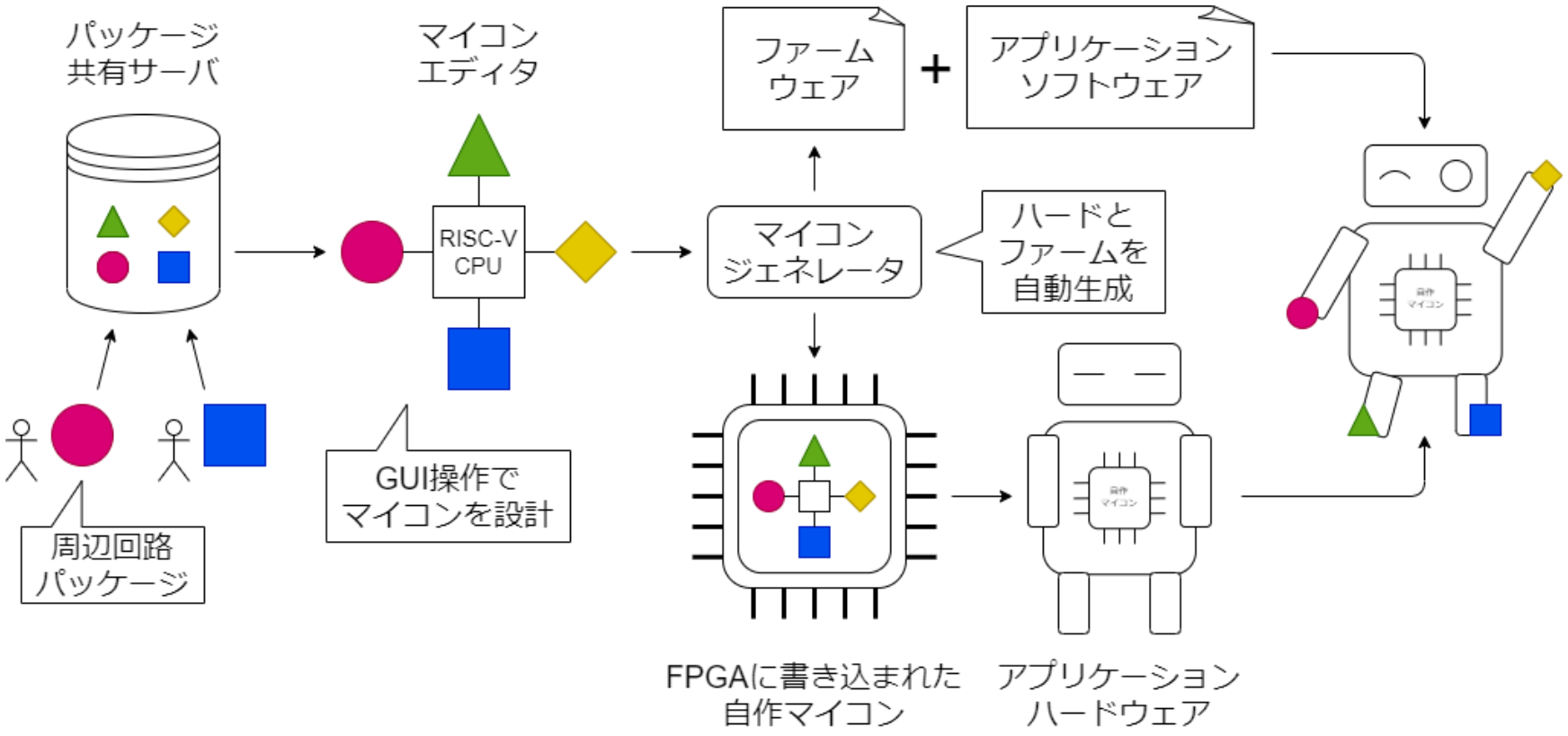
# 提案｜ハードとソフトのパッケージ化




ハードウェアとファームウェアをセットにする

→　共有＆再利用しやすく

# 提案｜実現する開発フロー




ハードウェア設計の知識がないFPGA初心者でも
FPGAを使ったモノを手軽に作れる！

# マイコンジェネレータ

## マイコン構成ファイル

src > micon.mcl

```
uart = uart(uart,UART);
uart.tx => fpga.pin1;
uart.rx <= fpga.pin2;

gpio = gpio(gpio,GPIO);
gpio.io <=> fpga.user_led;

cpu.irq5 <= fpga.pin11;
```

## ファームウェア

src > firmware > firmware.cpp > ...

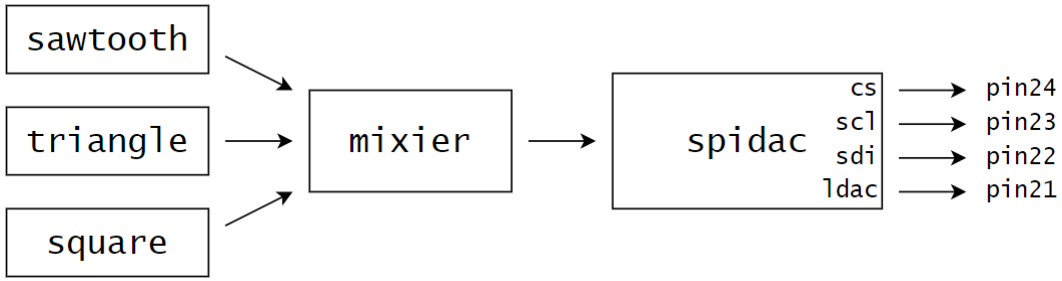
```
#include "firmware.hpp"

ROM_CFG rom_cfg((volatile uint32_t*)0x0200'0000);
/* definitions */
UART uart((volatile uint32_t*)0x0300'0000);
GPIO gpio((volatile uint32_t*)0x0400'0000);
/* end */
```

## ハードウェア

```
uart uart (
    .clk(clk),
    .resetn(resetn),
    .valid(uart_valid),
    .ready(uart_ready),
    .wstrb(uart_valid ? mem_wstrb : 4'b0),
    .addr(mem_addr),
    .wdata(mem_wdata),
    .rdata(uart_rdata),
    .rx(fpga_pin2),
    .tx(uart_tx)
);
wire uart_sel = mem_addr[31:24] == 8'h03;
wire uart_valid = mem_valid && uart_sel;
wire uart_ready;
wire [31:0] uart_rdata;
wire uart_tx;

gpio gpio (
    .clk(clk),
    .resetn(resetn),
    .valid(gpio_valid),
```

# デモ｜シンセサイザ

```
square1  = music(osc_square,  Oscilator);
square2  = music(osc_square,  Oscilator);
square3  = music(osc_square,  Oscilator);
sawtooth = music(osc_sawtooth,Oscilator);
triangle = music(osc_triangle,Oscilator);

mixier.ch0 <=(8)= square1.out;
mixier.ch1 <=(8)= square2.out;
mixier.ch2 <=(8)= square3.out;
mixier.ch3 <=(8)= sawtooth.out;
mixier.ch4 <=(8)= triangle.out;

mixier = music(mixier,Mixier);

dac.analog <=(12)= mixier.out;

sampling = counter(counter,Counter);

dac.sample <= sampling.overflow;

dac = music(spidac,SPIDAC);

dac.cs => fpga.pin24;
dac.scl => fpga.pin23;
dac.sdi => fpga.pin22;
dac.ldac => fpga.pin21;
```




```
Oscilator square1((volatile uint32_t*)0x0700'0000);
Oscilator square2((volatile uint32_t*)0x0800'0000);
Oscilator square3((volatile uint32_t*)0x0900'0000);
Oscilator sawtooth((volatile uint32_t*)0x0A00'0000);
Oscilator triangle((volatile uint32_t*)0x0B00'0000);
Mixier mixier((volatile uint32_t*)0x0C00'0000);
Counter sampling((volatile uint32_t*)0x0D00'0000);
SPIDAC dac((volatile uint32_t*)0x0E00'0000);
```

# ハードウェアオタクの作り方

# TinyGoのススメ

スイッチサイエンス
入江田

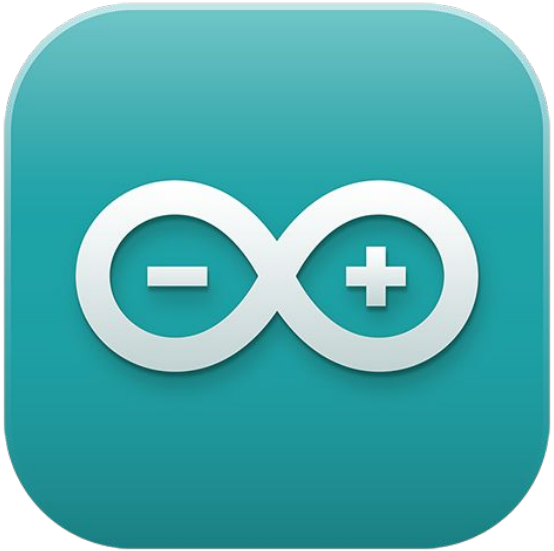
# TinyGoって？

- Go言語のマイコン向けサブセット
- LLVMを利用してネイティブコードを出力
- goroutineという並行記述サポートを言語仕様に持つ
- メモリ管理はGCで楽々
- Goとの互換性は年々向上、多くのGo資産が利用可能に

# 「Arduino IDE 2」について

arduino-cliをベースにUIラッパーをかぶせて作り直されました。

arduino-cliはGo言語で作られました。



# TinyGoとArduino

TinyGoはArduinoをサポートするようになりました。


# TinyGoのサポート状況

対応アーキテクチャ

- ATMegaシリーズ
- ESP8266/32/C3シリーズ
- STM32シリーズ
- nRF52シリーズ
- SAMDシリーズ
- Risc-Vシリーズ

対応バス

- I2C
- SPI
- UART
- USB-MSD/CDC/HID
- Bluetooth/BLE
- ADC/GPIO/PWM
- Wi-Fi/LoRa(まだ対象が少ない)

# arduino-cliとtinygoの比較（1）

コンセプトやサブコマンド体系が似ている

| | ソースコード | アーキテクチャ対応 | 書き込み | ライブラリ管理 |
|---|---|---|---|---|
| arduino-cli | Arduino(C++) | それぞれボードサポートを別途インストール | 書き込みツールもボードサポートに含む | サブコマンドで対応 |
| tinygo | Go | ほとんどのアーキテクチャを対応が内包 | 代表的な書き込みツールを一式内包 | Goの依存管理の仕組みを利用 |

# arduino-cliとtinygoの比較（２）

arduino-cli lib install “ライブラリ名”
arduino-cli compile -b <ボードFQBN> .
arduino-cli upload -b <ボードFQBN> -p <ポート名> .

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
}
```

go get “ライブラリURL”
tinygo build -target ボード名 .
tinygo flash -target ボード名 -port <ポート名> .

```
package main

import (
    "machine"
    "time"
)

func main() {
    led := machine.LED
    led.Configure(machine.PinConfig{Mode: machine.PinOutput})
    for {
        led.Low()
        time.Sleep(time.Millisecond * 500)
        led.High()
        time.Sleep(time.Millisecond * 500)
    }
}
```

# TinyGoの良さ

- GoはC++に比べて理解しやすいし、読みやすい
- PythonやJavaScriptと違いネイティブなので処理効率が良い
- 静的型言語なのでコンパイル時に多くの問題を発見できる
- Goの強力なライブラリ資産やツールをそのまま使える
- CGOという機能によりCの記述を一緒にビルドできる
- LLVMの最適化の恩恵とともにLLVMひとつで多くのアーキテクチャに対応
- また成果物はかなりコンパクト
- 言語仕様に含まれるgoroutineによりマルチタスク機能を内包
- 多くのボードサポートやフラッシュツールを内包しているのでArduinoのように別途インストールする必要がない

# TinyGoの辛さ

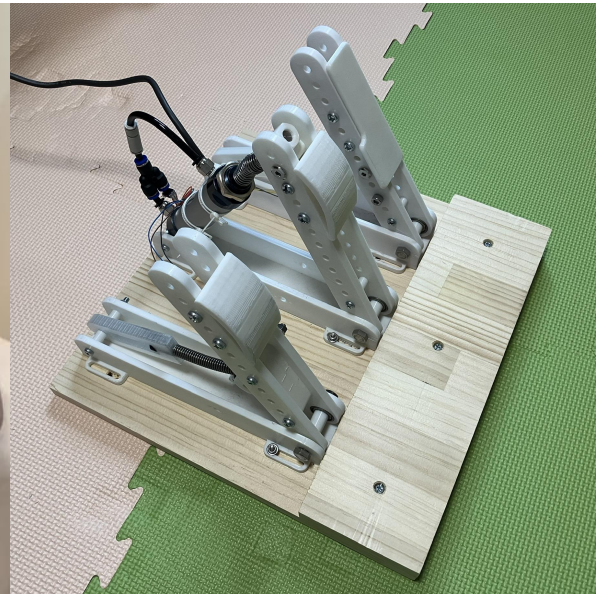
- ボードベンダーからのサポートがない
- 新しいボードサポートが増えるのはArduino対応に比べると遅い
  （ただし、既存アーキテクチャであれば仮のボードサポートの自作もOK）
- Arduino向けのライブラリ資産を取り込むのは難しい
- IDEサポートが今のところない（VSCode拡張はある）
- マルチコアサポート、メジャーなWi-Fi対応がまだ未実装

最近本も出版されました！

https://amzn.asia/d/cf2t7RH

# TinyGoでハンドルコントローラー

高トルクモーターをつかったフォースフィードバックハンドルコントローラー！

# まとめ

- arduino-cliに似た使い心地のtinygoがある
- C++よりは読みやすく書きやすいGo言語でマイコン開発ができる
- arduino-cliと違い一通りのボードサポートが最初から内包している
- Goのライブラリ公開やドキュメントの仕掛けをそのまま利用できる
- Goの強力な依存解決やライブラリ資産を利用可能
- LLVMの最適化の恩恵もあり出力はネイティブかつコンパクト
- C言語のコードを一緒にビルドすることができる
- CPUの多くは対応済み、周辺機能のカバーがまだ不足気味
- 「go 関数呼び出し」にて並行処理が書けちゃう！
- 試してみて良ければ周辺サポートをコントリビュートして！

# arduino-cliとTinyGoのことなら

スイッチサイエンスの
入江田（HN:nobonobo）に聞いてください～！


# SWITCHSCIENCE システムエンジニア募集中！


## 株式会社スイッチサイエンスでは、
## システムエンジニアとして一緒に働いてくれる人を募集しています。

当社では、数十万人のユーザーが利用する自社のウェブショップや社内システムの開発を手がける仲間を求めています。ショップ運営メンバーの業務効率を高め、お客様に素晴らしいショッピング体験を提供し、やりがいを感じる仕事がここにあります。

## 一緒に挑戦しましょう！

# クレジットカードの不正検知

北九州市立大学 国際環境工学研究科
情報工学専攻 融合システムコース

陳酌航　チンシャクコウ

(指導教員: 山崎 進)

# 目次

- クレジットカードの不正検知手法に関する調査研究
- IoTと組み合わせた初歩的な発想

# 調査研究：研究背景

- 世界中のクレジットカード不正利用被害額が**増加**している
  不正利用の対策：予防、検知
  **不正検知システム**が注目されている

- 従来の**不正検知システム**：
  **人手**で不正検出のためのルールを登録する
  システムがルールを参照して判定する
  **人手**で判定結果を分析してルールを改善する

# 調査研究：研究背景

- 従来の**不正検知システム**の問題点：
  人手での対応には限界がある
  ルール構築に手間がかかる
  メンテナンスが難しい

- **機械学習**などの最新手法が活用されている
  従来より高い判定精度も達成できる

# 調査研究：先行研究

- 2018 年に、**不正検知**に運用された**機械学習**の手法に関する調査研究が発表された
- 6 分野での2007 年から2017 年までの文献を調査した
- クレジットカード分野では 19 本の文献から 8 つの手法にまとめられた

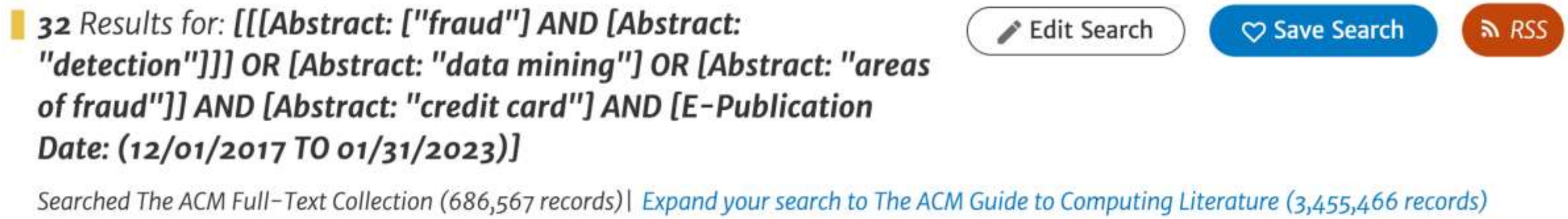
| | 件数 |
|---|---|
| 隠れマルコフモデル | 1 |
| 人工ニューラルネットワーク | 5 |
| 決定木 | 4 |
| サポートベクターマシン | 3 |
| 単純ベイズ | 1 |
| Association rule analysis | 1 |
| ランダムフォレスト | 2 |
| k 近傍法 | 2 |

# 調査研究：研究目的

- 2017 年以後はどのような不正検知の手法があるのかを明らかにする
- 手法の新規性と利点を明らかにする
- どのように進化したのかを明らかにして不正検知の研究動向を把握する

# 調査研究：研究方法

- ACM Digital Library を利用し、**不正検知** AND **クレジットカード** AND **2017.12.1-2023.1.31**の条件で検索した
- 32 件の文献を得た



# 調査研究：研究方法

- その内訳は次の通り：

  **11件の新規性のある研究**

  5件の新規性のある関連研究

  10件の他分野の研究・関連の薄い研究

  5件の新規性のない研究（比較研究など）

  1件の入手できない研究

- 2017年以前の先行研究の調査結果と2017年以後の本研究の調査結果の比較およびまとめを行う

# 調査研究：調査結果

| | 文献タイトル | 提案された手法 | 新規性 |
|---|---|---|---|
| 1 | Multiple perspectives HMM-based feature engineering for credit card fraud detection | 隠れマルコフモデルに基づく取引履歴の特徴量エンジニアリング | 加盟店の取引履歴と取引間の木目細かい時間的依存性の考慮 |
| 2 | Credit card fraud detection based on self-paced ensemble neural network | Self-paced Ensemble に基づく浅層ニューラルネットワーク | Self-paced Ensemble と浅層ニューラルネットワークの組み合わせ |
| 3 | POSTER: A Deep Learning Method for Fraud Detection in Financial Systems | 多層人工ニューラルネットワーク | 多層人工ニューラルネットワークの運用 |
| 4 | Credit Card Fraud Detection with NCA Dimensionality Reduction | 近傍成分分析による次元削減に基づく k 近傍法 | 近傍成分分析による次元削減の運用 |
| 5 | A Behavior-cluster Based Imbalanced Classification Method for Credit Card Fraud Detection | 行動クラスターに基づく不均衡データ分類 | 行動特徴量別のクラスタリング |
| 6 | Real Time Data-Driven Approaches for Credit Card Fraud Detection | RBF カーネルのハイパーパラメータが最適化された 1 クラスサポートベクターマシン・$T^2$管理図 | RBF カーネルのハイパーパラメータが最適化・$T^2$管理図の運用 |
| 7 | An Efficient Real Time Model For Credit Card Fraud Detection Based On Deep Learning | オートエンコーダ | オートエンコーダの運用 |
| 8 | Ensemble Learning for Credit Card Fraud Detection | ランダムフォレストとニューラルネットワークに基づくアンサンブル学習 | ランダムフォレストとニューラルネットワークに基づくアンサンブル学習の運用 |
| 9 | Adversarial Fraud Generation for Improved Detection | 敵対的生成ネットワークによるオーバーサンプリング | クラス境界からの少数派要素の合成 |
| 10 | Credit Card Fraud Detection using Data Pre-processing on Imbalanced Data - both Oversampling and Undersampling | オーバーサンプリングとアンダーサンプリングに基づく k 近傍法 | オーバーサンプリングとアンダーサンプリングの両方同時運用 |
| 11 | Credit Card Fraud Detection based on CSat-Related AdaBoost | 顧客満足度を考慮したアダブースト | 顧客満足度の考慮・総利益からの評価 |

# 調査研究：考察

- 本研究で調査した 11 件すべての文献は**機械学習**に基づく手法を提案した

| 手法 | 先行研究 | 本研究 |
|---|---|---|
| 隠れマルコフモデル | 1 | 1 |
| 人工ニューラルネットワーク | 5 | 3 |
| 決定木 | 4 | |
| サポートベクターマシン | 3 | 1 |
| 単純ベイズ | 1 | |
| Association rule analysis | 1 | |
| ランダムフォレスト | 2 | 1 |
| k 近傍法 | 2 | 2 |
| 敵対的生成ネットワーク | | 1 |
| アダブースト | | 1 |
| アンサンブル学習 | | 1 |
| $T^2$ 管理図 | | 1 |
| オートエンコーダ | | 1 |
| 行動クラスター（k-means 法） | | 1 |

# 調査研究：考察

- 先行研究では**分類器**を主な視点として手法を調査した
- 本研究で調査した7 件の文献は**トレーニングデータの取り扱い**に基づいて改善した
- 複数の手法の利点を1つにまとめる手法も増えた
- アンサンブル学習

| 手法 | 先行研究 | 本研究 |
|---|---|---|
| 隠れマルコフモデル | 1 | 1 |
| 人工ニューラルネットワーク | 5 | 3 |
| 決定木 | 4 | |
| サポートベクターマシン | 3 | 1 |
| 単純ベイズ | 1 | |
| Association rule analysis | 1 | |
| ランダムフォレスト | 2 | 1 |
| k 近傍法 | 2 | 2 |
| 敵対的生成ネットワーク | | 1 |
| アダブースト | | 1 |
| アンサンブル学習 | | 1 |
| $T^2$ 管理図 | | 1 |
| オートエンコーダ | | 1 |
| 行動クラスター（k-means 法） | | 1 |

# IoTと組み合わせた初歩的な発想

- 検出された際の不正利用かどうかの確認：
  カード会社→→（電話）→→ユーザー

- 問題点：
  コールセンターに負担がかかる
  連絡が取れるまでに時間がかかって効率が悪い

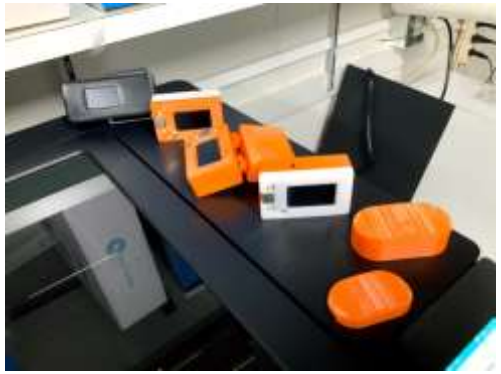
# IoTと組み合わせた初歩的な発想

- IoTと組み合わせて活用できれば：
  取引の検出からユーザーへの通知まで自動化できる

- 検出された際：
  不正検知システム→→（リアルタイム）→→IoTデバイス（テレビなど）
  ユーザーの
- 不正利用の確認の効率化ができる

# EMGデバイス開発

# Bluetooth対応EMGデバイス開発



試作機たち

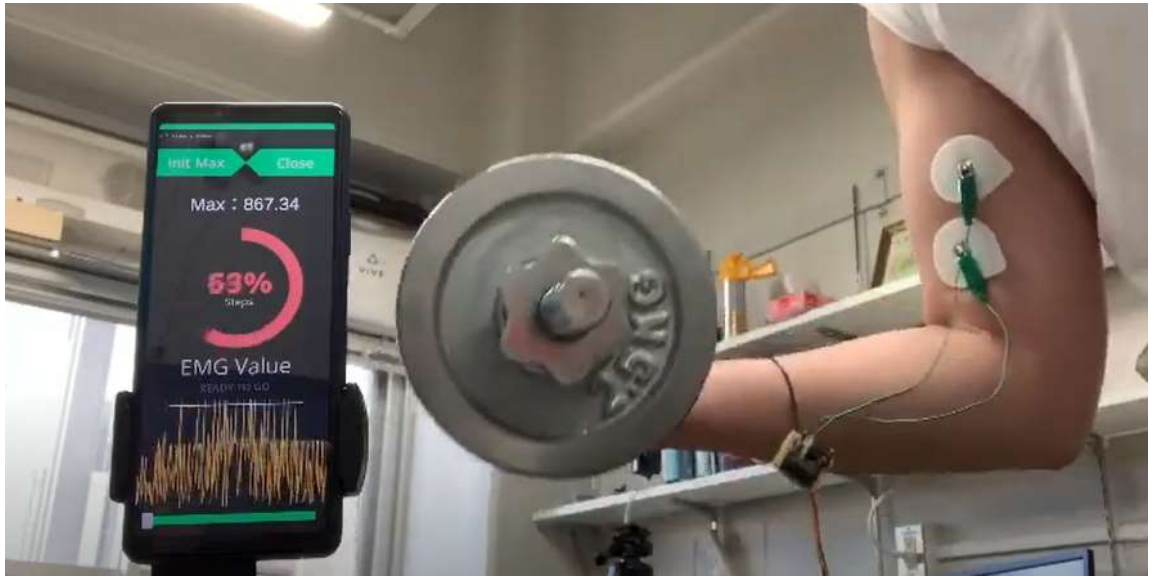
試作機５号

EMGセンサーpicoを見て疑問に思う
・電極が2個しかないの？
・こんな小型になるの？

理論は研究室にあるのだから近いものが作れなるはず・・・
よし作ろう！！



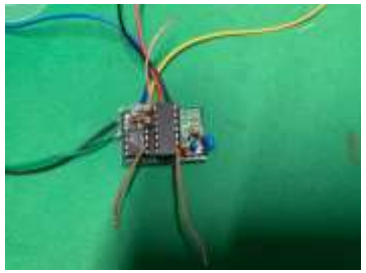
筋電デバイス複数使用時



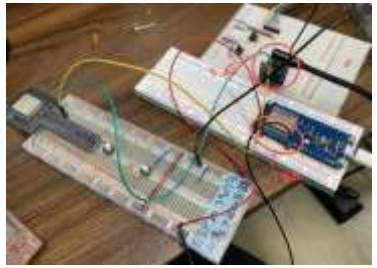

スマホアプリとの連携

# Bluetooth対応EMGデバイス開発

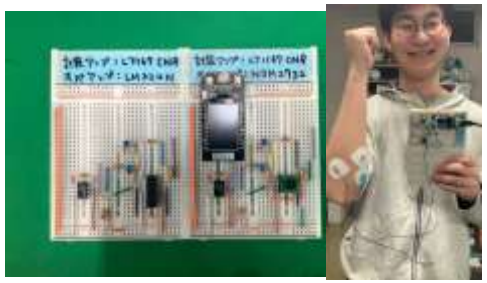
## ５v用の筋電装置と回路



研究室で普及している5v筋電センサ



マイコンを搭載した5v筋電センサー



マイコンとディスプレイを搭載した3.3v筋電センサー（完成計の元回路）

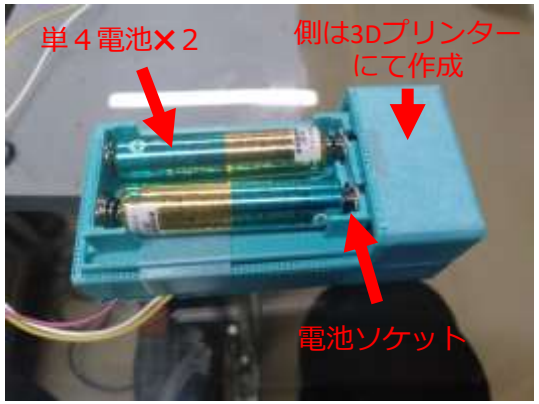
モバイル型のEMGデバイス（筋電）を作成開始。

スマホで繋がるモバイルEMGが欲しい！！（今のBluetoothイヤホンみたいなデバイスがいい！！！）

イメージデザイン

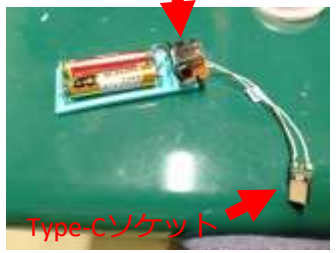
## ５vの単4電池式モバイル筋電装置【試作機１号】


乾電池型マイコン搭載5v筋電センサー試作機一号




乾電池使用

セリアの乾電池式USB充電器

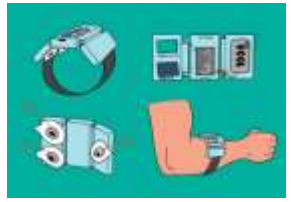



セリアの乾電池式USB充電器中身

5Vのタイプだとリポバッテリー（3.7v～4.0v)を直接使用できないため、まずは、昇圧するために１００均(セリア）の単３電池USB充電器を改造してモバイルタイプを作成。
マイコンはESP32-Tdisplayというスプレイ付きのマイコンを採用(3.3v駆動）。

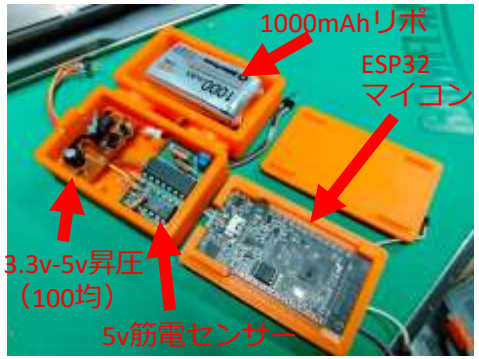
# Bluetooth対応EMGデバイス開発

## ５vのリポバッテリー式モバイル筋電装置【試作機２号】



デバイスイメージ（手書き）

電極取り付け

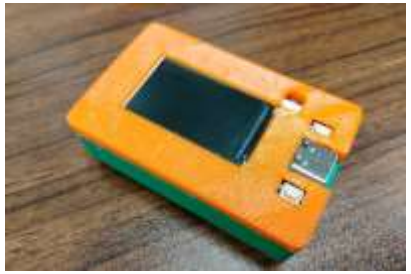



試作機２号中身


試作機２号

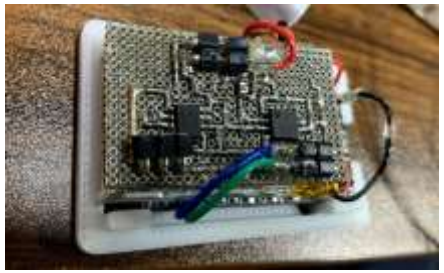
乾電池からリポバッテリーに変更した。
デザインを薄型にするため、本体を２つに分割した。（振動によるノイズを避けるため）
ディスプレイに筋電の絶対値が表示されるようにした。（動作状況を把握するため）

## ５vの単4電池式モバイル筋電装置（乾式電極と湿式電極）【試作機３号】



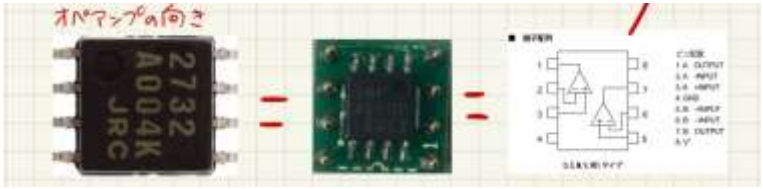
試作機3号(小型化)


乾式電極のトライ



試作３号の内部回路

スマホアプリとの連携



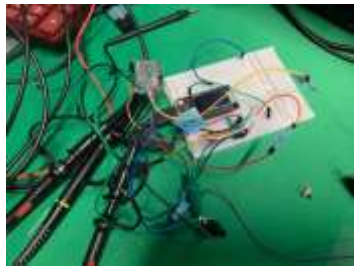

使用したオペアンプ（ＮＪＭ２７３２Ｍ）

ここから3.3vの筋電センサーを作成。
計装アンプ(AD8226)とオペアンプ(NJM2732M)をサーフェースマウント型に変更→小型化の成功

Bluetooth接続用のスマホアプリを作成。
リアルタイムに生波形と最大値を100%にした時の出力を円グラフにて表示

# Bluetooth対応EMGデバイス開発

## NC旋盤を使用した基盤の作成（自作）


波形確認


ESP32マイコン基盤


リポ充電回路


Type-Cリフロー半田

NC旋盤を利用して、自作の基盤作為を行い、業者発注前の回路ミスの追求を行なった。

また、リポバッテリー充電回路をＭＣＰ７３８３１を使用して作成。
回路図は、秋月の製品の回路図を参考にEagleにて作成。

## 基盤の発注から部品の半田＋試作機４号


届いた基盤


半田後（裏）


半田後（表）


バッテリーの装着


簡易ケースの作成


バッテリー装填時


簡易ケースの蓋


試作機３号との比較

Eagleで作成した基盤を発注。
届いた基盤に部品をリフロー式でハンダ付を行なった。

ケースを作成し、回路に関してはほぼほぼ完成になった！！

# Bluetooth対応EMGデバイス開発

## ケースの改良（現状の最良試作）【試作機５号】


試作機５号（本体）

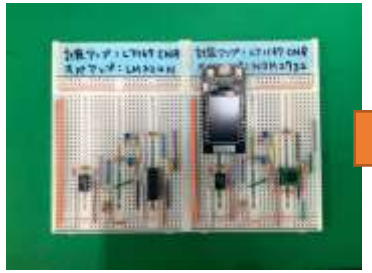

試作機５号の複数使用


没ケース


電極面

ケースのデザインを変更し、GND電極部分を伸ばす形状に変更した。
一度に複数使用することを想定。
バッテリーの持ち時間は1時間30分。
パソコンにて一度に複数のデータを取得可能。
腕相撲の計測に使用する。

## 時系列順、試作機（開発期間：1年）



**回路作成**

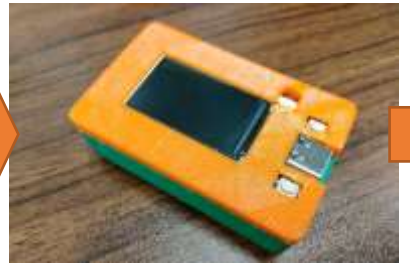
モバイル型のEMGデバイス（筋電）を作成開始。

**試作機１号**

セリアの単３電池USB充電器を改造してモバイルタイプを作成。
マイコンはESP32-Tdisplay。

**試作機２号**

乾電池からリポバッテリーに変更。
デザインを薄型にするため、本体を２つに分割した。筋電の絶対値が表示されるようにした。



**試作機３号**

3.3v筋電センサーを作成。
小型化の成功
Bluetooth接続用のスマホアプリを作成。

**試作機４号**

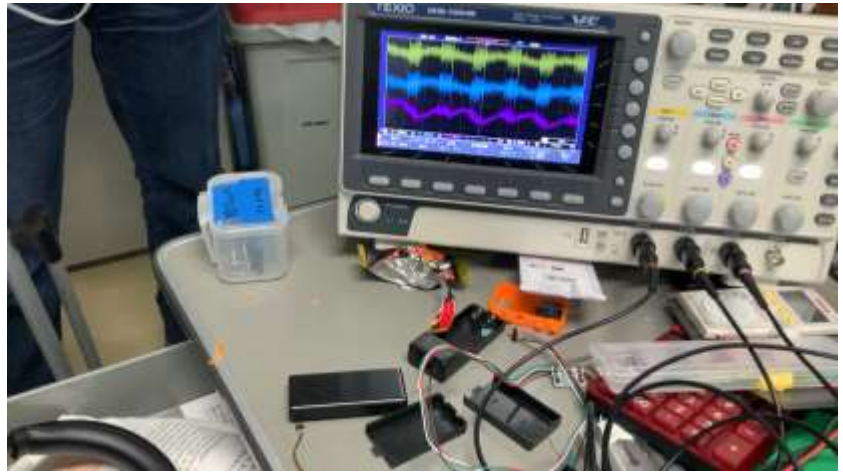
Eagleで作成した基盤を発注。
届いた基盤に部品をリフロー式でハンダ付を行なった。

**試作機５号**

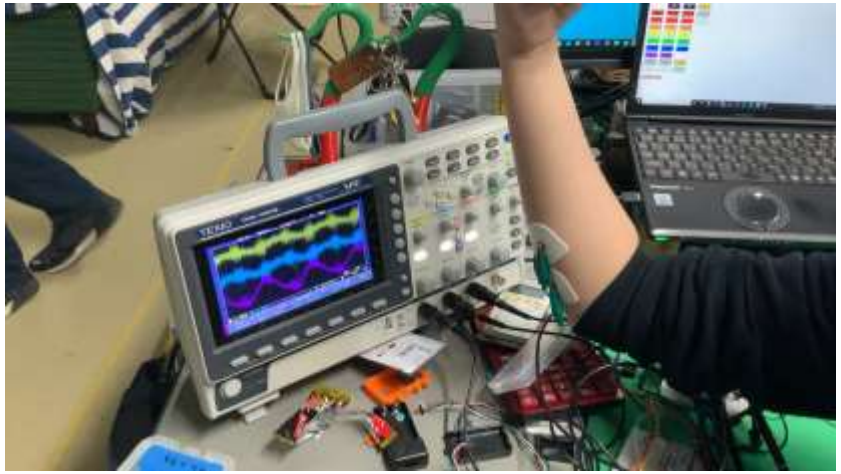
ケースのデザインを変更し、GND電極部分を伸ばす形状に変更した。

# Bluetooth対応EMGデバイス開発

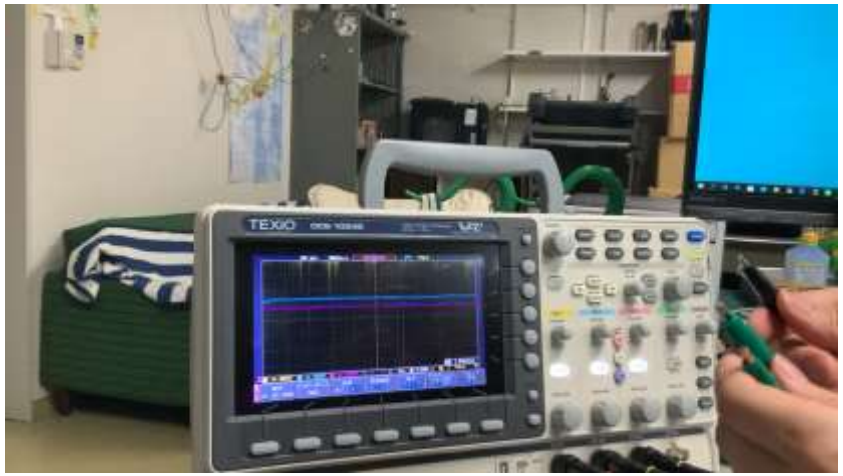
**鉄に触った時の波形変化**



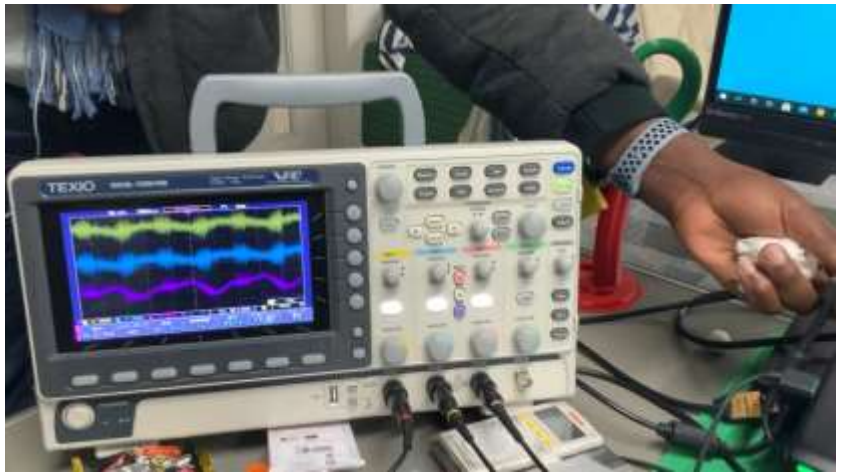
**筋電波形(計装アンプの出力、オペアンプの出力×２)**



**GND接触した時の波形変化**



**AC電源（商用ノイズ）の計測**

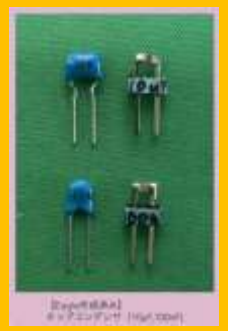


# Bluetooth対応EMGデバイス開発

**使用した部品**

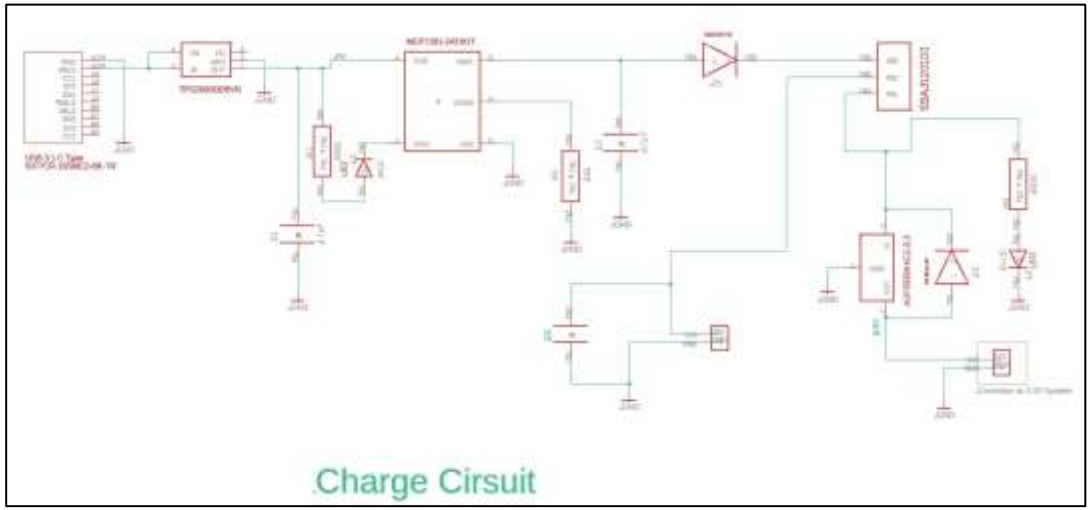
メインチップ

増幅回路（EMGメイン）

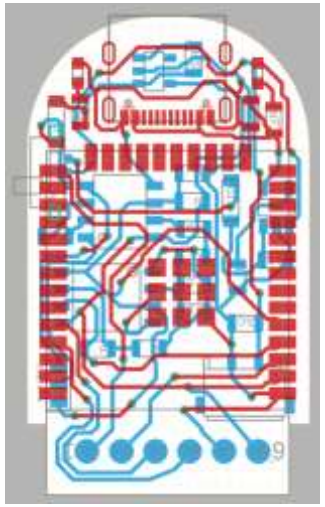



電源回路系統

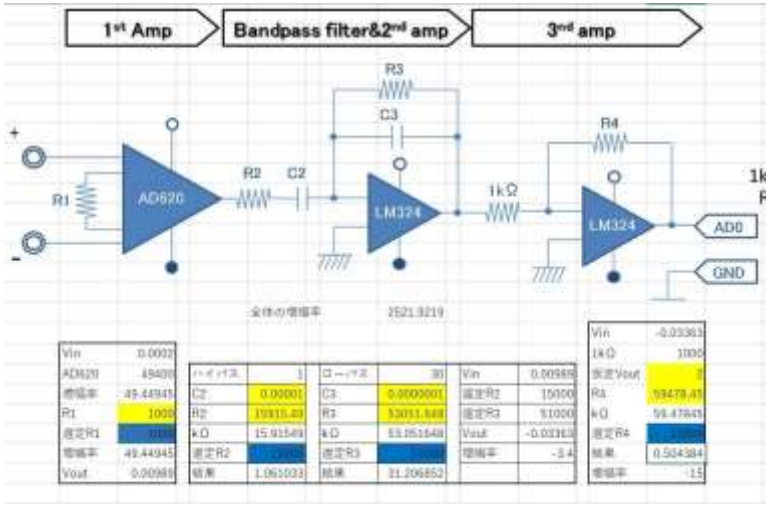
**回路図＋信号増幅の計算**




Schematic



Board




増幅率計算(5v用→本来は3.3vにすべき)

# Bluetooth対応EMGデバイス開発

**課題点**

- バッテリーが短い
- Bluetooth通信が弱い
- ケースが大きい
- GNDの紐が邪魔
- アプリがしょぼい
- 増幅率の見直し
- 別の増幅方法を検討

Facebookのコミュニティに支えられてます。

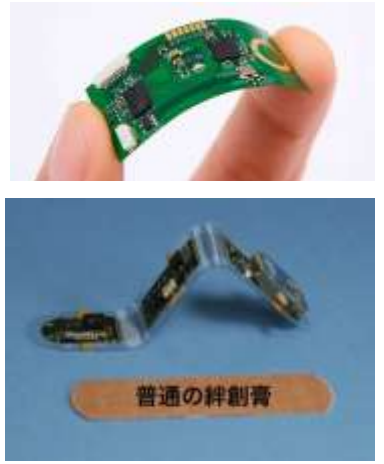

**今後の改良**

充電コネクタの規格
変更

マイコンの変更
省電力・BLE5のもの
nRF52840

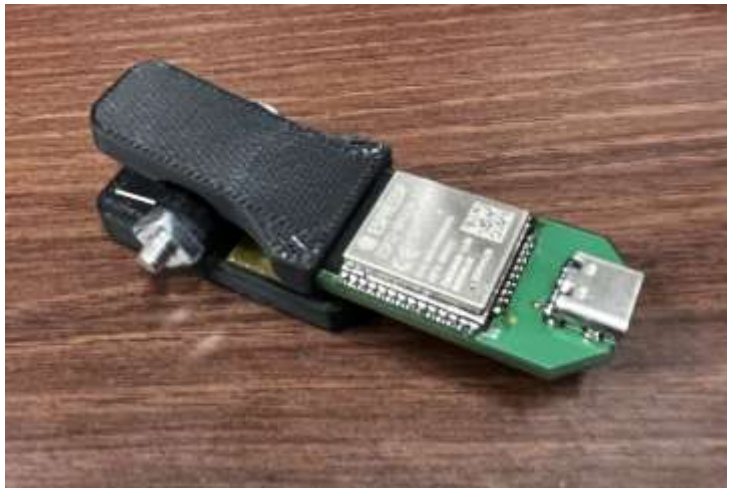
マイコンの変更
省電力・BLE5のもの
ISP1507

乾式電極

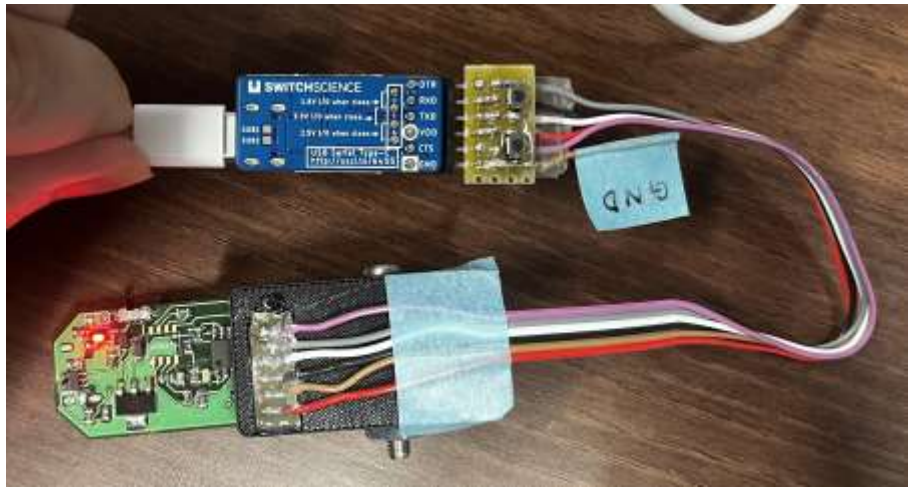



フレキシブル基盤

# プログラム書き込み用のポゴピン作成

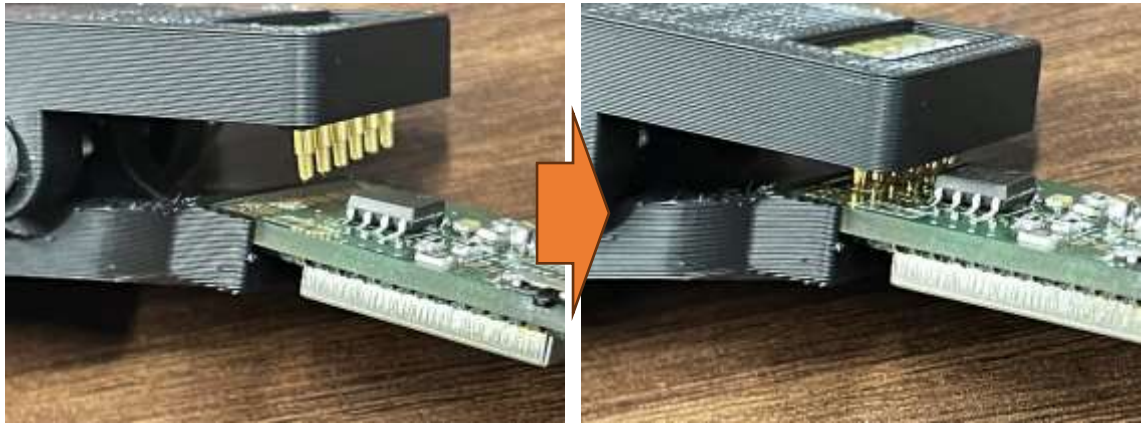


裏




書き込み時

表



接点

# プログラムをBLEに対応