

SWEST27 EmbLT登壇者一覧

No	受付番号	氏名	タイトル	カテゴリ	資料
1	0090	山之上 卓	Vibe Codingで着る電光掲示板を機能拡張してみた	やってみた！作ってみた！ものの紹介	○
2	0001	石垣 良	電子楽器の作品紹介2025	やってみた！作ってみた！ものの紹介	○
3	0100	大栄 豊	小さく作ってかわいいを実現 Stack-chan chibi (スタックちゃん 飛ぶ)	やってみた！作ってみた！ものの紹介	○
4	0049	曾根 卓朗	撮影時刻を高精度に記録できる光学ビーコン	こんな研究してます	○
5	0095	常田 裕士	Modern Xen:VirtIOとZephyrによるイマドキのXen の作り方	やってみた！作ってみた！ものの紹介	○
6	0067	西口 昌利	AtomVMをやってみた	やってみた！作ってみた！ものの紹介	○
7	0033	高村 直也	Xiao RP2040のステートマシンでシリアルCS信号生 成	上手くつながった！/ つながらなかった…話	○
8	0019	片山 立	生成AIと協調して要求を開発する ～イノベーティ ブ思考法による多視点仕様策定プロセス～	こんな研究してます	○
9	0007	井田 健太	Matterと日本の家電	上手くつながった！/ つながらなかった…話	○
10	0082	岩崎 雅弘	学ロボ労災今昔物語	人の失敗は蜜の味？こんなバグやっちゃいま した	—
11	0107	安藤 友樹	AI時代もここから ～悩める組込みシステムエンジ ニア～	組込みへのAIと愛を語る！	—

Vibe Coding で着る電光掲示板 を機能拡張してみた

REG-0090, IS-006 山之上 卓

福山大学名誉教授

公立大学法人福山市立大学大学改革推進アドバイザー



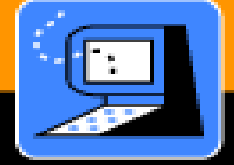
自己紹介



- ・ 1980年、CQ出版社インターフェース6月号「LKIT-16によるFORTRANシステム」
- ・ 1980年に共立出版 bit 12月号のナノピコ教室解答「じゃんけん大会」で私のプログラムが優勝
- ・ 1988年、“パソコン通信を使い電算機言語を開発” 日本経済新聞,11月17日
このとき、(現在名古屋大学の)高田 広章先生にお会いしています。
- ・ 2018年、Maker Faire Tokyo 2018、着る電光掲示板、声で操作するフォークリフトなど出展
- ・ 2023年、Maker Faire Tokyo 2023、シン3次元表示装置(トランジスタ技術賞受賞)、テレポードレッサーなど出展
- ・ 2024年、Ogaki Mini Maker Faire 2024、テレポーターションシステム(送受信)など出展



7/8のハンズオン@広島 Gemini CLIを初めて使ってみた



7月
8

Gemini CLIとバイブコーディングでLINEのチャットボットを作ろうハンズオン（初心者歓迎！）

広島のイノベーションハブひろしまCampsから配信ありのハイブリッドイベント

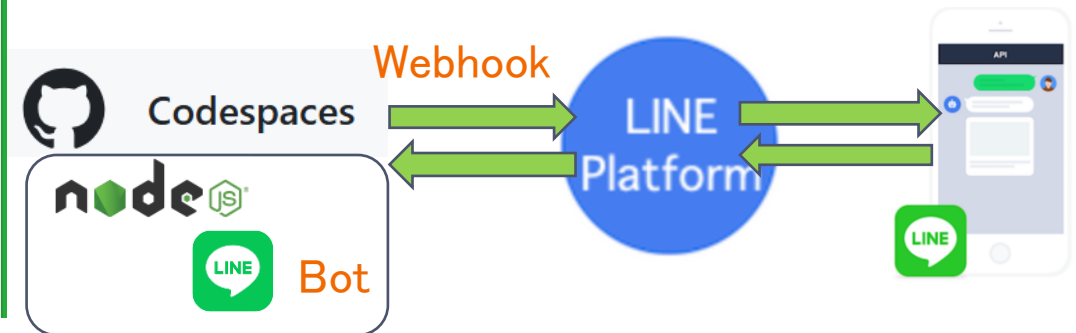
主催：LINE Developer Community

connpass



ー 準備:

- ー LINE ID, LINE Bot アカウント, Messaging API ... Key, Secret,
- ー Github Codespaces
- ー Gemini CLI install
- ー 「Node.jsで簡単なオウム返しLINE Botを作成してください。」
- ー Key, Secretの設定, port 転送の設定
- ー 出力を見ずにひたすらEnterでちょっと失敗。→その日のうちに成功



着る電光掲示板



ProtoPedia

タイトル・概要、開発素材から作品を探す



作品

イベント

開発素材

使い方

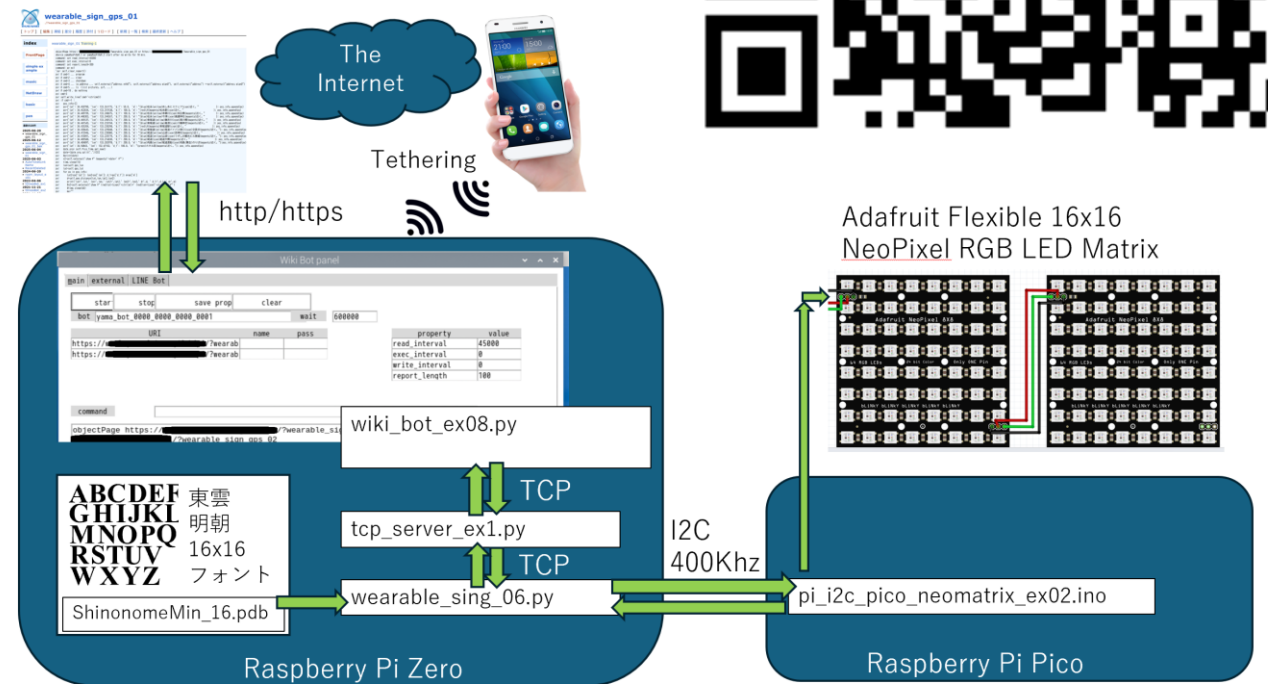
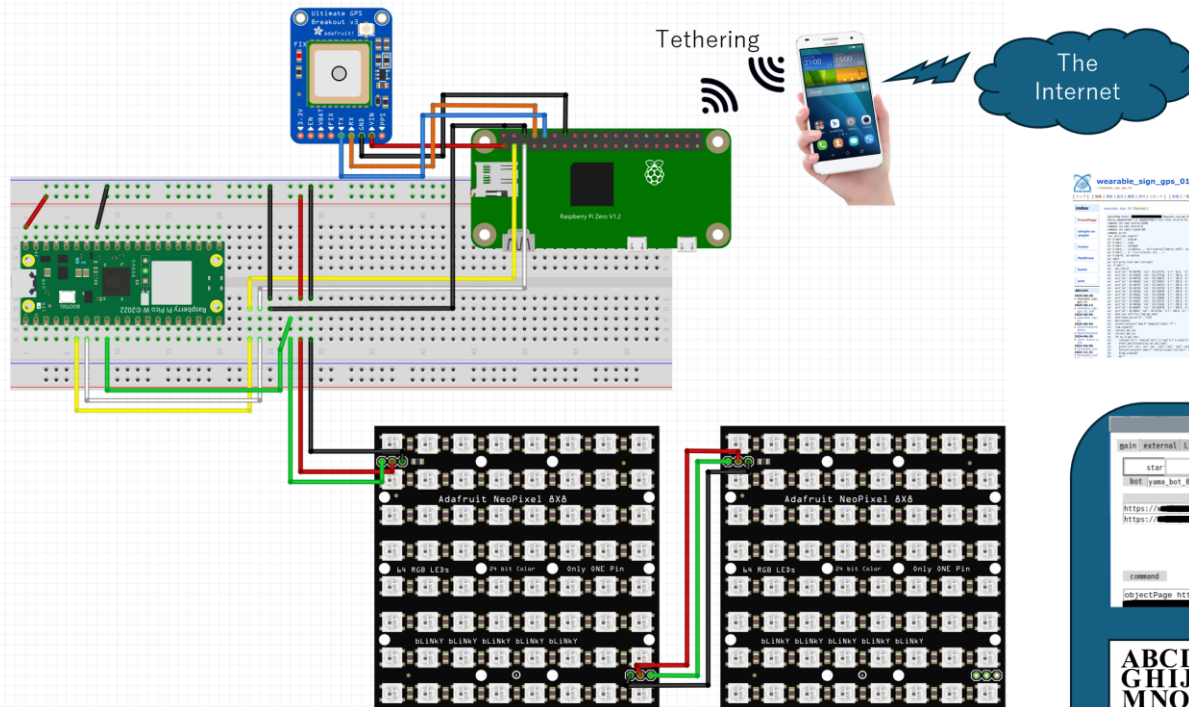
Wikiで制御できる、場所で表示を変えることができる、~~LINEもOK~~、着る電光掲示板（走るのもOK）

開発中 2025.06.14 公開 （2025.08.10 更新） © CC BY 4+ 562

表示内容をインターネット上のWikiのページで制御することができる着る電光掲示板です。場所によってその表示内
こともできます。着て走ることもできます。~~2025年7月22日LINEと連携しました~~



機能拡張前のハードとソフト



Hardware

Software(機能追加前)

7/13 着る電光掲示板 機能拡張スタート



Takashi Yamanoue

7月13日 21:24 · 2人

Raspberry Pi Zero 2 W にgemini-cli をインストールしようとしたけど、認証のところで

1. login with google

を選んだら、Chrome を開くことができずにアウト。

事前に Gemini API Keyを取得して、.env ファイルに、

GEMINI_API_KEY=<gemini api key>

...

2. use GEMINI API Key

を選んだら、

GEMINI_API_KEY environment variable not found. Add that to your

environment and try again (no reload needed if using .env)!

とのことで、.env が見つからない、みたい。

詰んだ。

7/15

Raspberry Pi 4で実験



facebookつながりの友人(それ以前からの友人)からRaspberry Pi 4で開発して、結果をRaspberry Pi Zeroに移すideaを教えてもらう。



Takashi Yamanoue

7月15日 11:13 · 2人

Raspberry Pi 4 にnode.js とgemini-cli をインストールして、先週の今日、のびすけさんに教えてもらったバイブコーディングでおうむ返しbotを作り、ngrok でポートをインターネットに公開して、webhookでlineのアカウントのmessaging api と繋いで、おうむ返し、成功！

次は、着る電光掲示板との接続。

7/15

Raspberry Pi 4 で実験



Takashi Yamanoue

ただ、ちょっと問題が発覚しました。node.jsを再起動するたびに、ngrokのurlが変わるので、それをlineアカウントのmessaging apiで修正しないといけません。ngrokのurlを固定する方法はないかな？

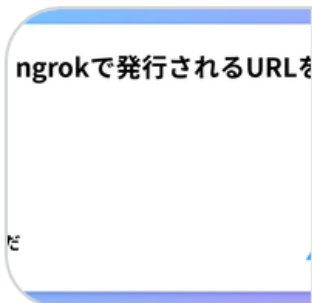
3週間 いいね！ 返信する 編集済み



Takashi Yamanoue

これで良いかな？

https://zenn.dev/y_taiki/articles/ngrok_domain



ZENN.DEV

【簡単】ngrokで発行されるURLを固定する

3週間 いいね！ 返信する プレビューを削除



Takashi Yamanoue
URL固定、成功！

...

7/15

Raspberry Pi Zeroで実験



Takashi Yamanoue

Raspberry Pi 4で動くようになったline botのnodeのプログラムや設定ファイルを Raspberry Pi Zeroにコピーして、ngrokの設定をRaspberry Pi Zeroで、Pi 4の時のngrokの設定Webページを、Pi Zeroでもう一回やって、Pi Zeroで

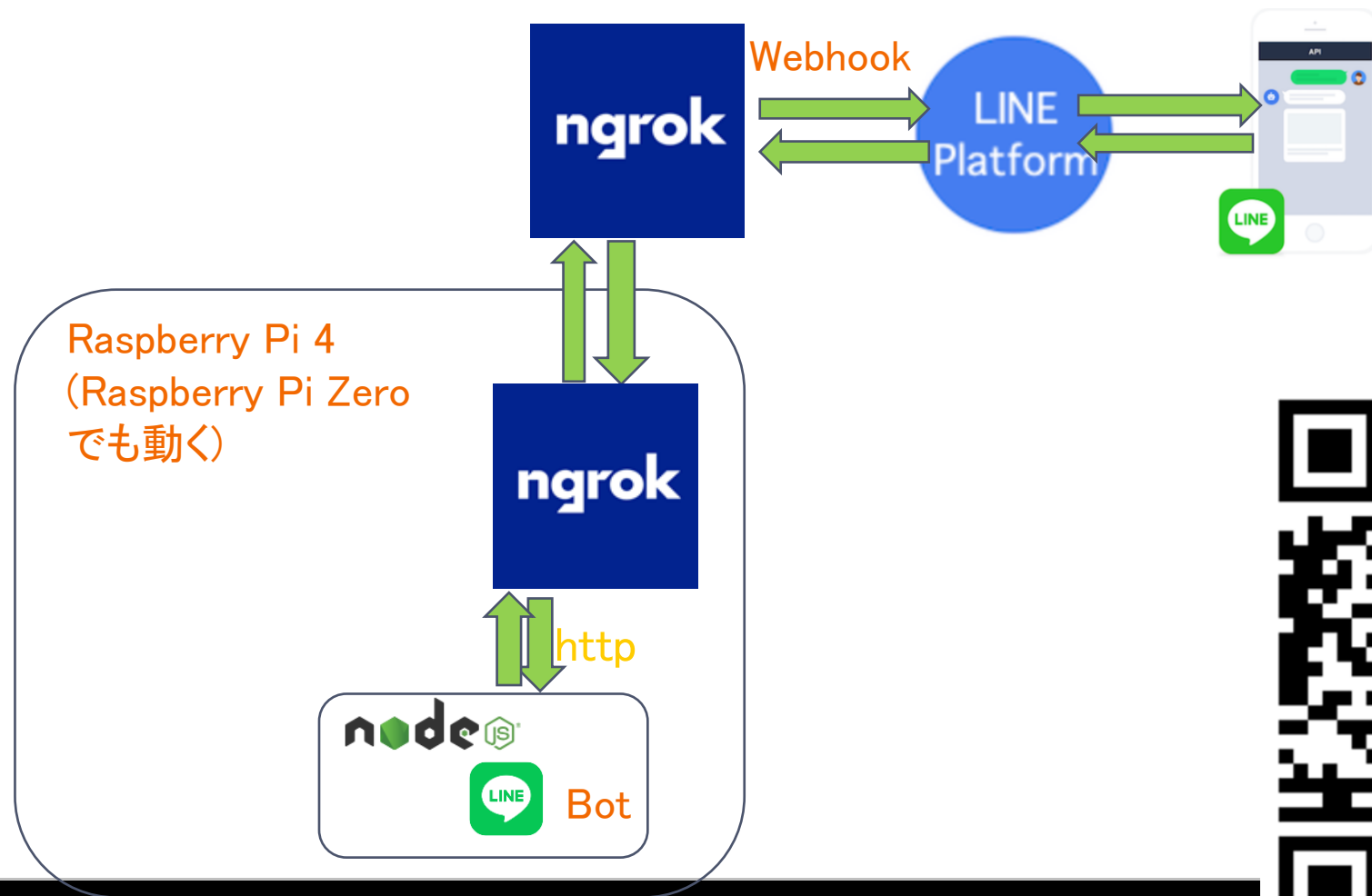
npm start

別のターミナルで ngrok http 3000

ここで表示された urlを、Line アカウントのmessaging apiの Webhookに書いて、(urlの後ろにwebhookを付けるのを忘れないように)、検証して有効にしたら、

Pi Zeroでおうむ返しbotが動きました！

さん、ありがとうございました！



7/16

着る電光掲示板への組み込みで悩む



Takashi Yamanoue

7月16日 1:07 · 2人



line bot はnode.js, 着る電光掲示板のwiki bot はpython。どうやって繋ごうかな？

候補1. tcp ... 私は慣れているので, もし, これが使えると嬉しいです.

候補2. python-shell ... console io を使うみたい。すでにpythonで多用している出力はどうなる？



Takashi Yamanoue

tcp/udp



<https://qiita.com/youtoy/items/385feead64ff905e9f76>

Qiita Advent Calendar

【Node.js 2021 (2つ目)】
Node.js での
TCP通信をシンプルに試す
youtoy
DI株式会社

QIITA.COM

【Node.js 2021 (2つ目)】 Node.js での UDP・TCP通信をシンプルに試す (2021年12月) - Qiita



7/16

着る電光掲示板への組み込みで悩む



Takashi Yamanoue
python-shell

...

https://qiita.com/Ayumu_walker/items/fae0e368d635c6b70fea

Qiita

jsからPythonのコードを
python-shellを使ってみた

Ayumu_walker

QIITA.COM

Node.jsからPythonのコードを呼び出すPython-shellを使ってみ
た - Qiita

i

3週間 いいね！ 返信する プレビューを削除

pythonをrepl起動して裏でリモコンするだけみたいですね。なんらかの入力待ち
はバグりそうです。

3週間 いいね！ 返信する 非表示にする



7/16

着る電光掲示板への組み込みで悩む



Takashi Yamanoue

もしかして、Node.jsじゃなくてpythonでwebサーバーを作った方が早いかな？

...

<https://qiita.com/okhrn/items/4d3c74563154f191ba16>

Qiita

pythonでローカルwebサーバ

QIITA.COM

pythonでローカルwebサーバを立ち上げる - Qiita

okhrn



7/16

着る電光掲示板への組み込みで悩む



Takashi Yamanoue

python, flask というのがあるみたい。これでも良いかな。

...

<https://qiita.com/NaruNeko/items/482ed8dad919d85f658>

Qiita

on]
応答ができるLineBotを
Flask+ngrok)

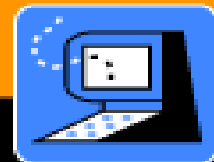
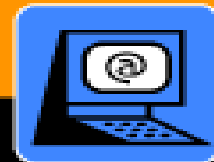
NaruNeko

QIITA.COM

[python] 簡単な応答ができるLineBotを作ってみた (Flask +
ngrok) - Qiita

i

7/16... 結局python + flask



Takashi Yamanoue

gemini に、index.jsを pythonとflaskを使って書き換えて、とお願いしたら、ポート番号がちょっと間違っていました。flaskに書き換えてくれました。ポート番号を書き換えたら、動きました！

3週間 いいね！ 返信する 編集済み



Takashi Yamanoue

なお、これ、geminiに頼む前に自分でやってみたのですが、失敗しました。

3週間 いいね！ 返信する

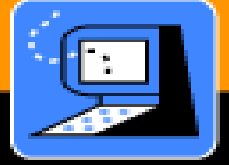


Takashi Yamanoue

例を見て callback method を使っていたのですが、これは webhook method じゃないといけなかったみたい。

3週間 いいね！ 返信する

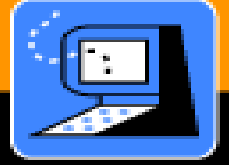
...



＞ index.js をpython flask を使って書き換えてください。

◆ はい、承知いたしました。index.js を Python と Flask を使って書き換えますね。まず、index.js の内容を確認します。
index.js の内容を Python と Flask を使って書き換え、app.py というファイル名で保存します。また、必要なライブラリを記載した requirements.txt も作成します。まず、app.py を作成します。

...



◆ これで、index.js と同等の機能を持つ Python/Flask アプリケーションが app.py として作成され、必要なライブラリが requirements.txt に定義されました。次に、pip を使ってこれらのライブラリをインストールします。

```
shell pip install -r requirements.txt (Install Python packages from ...
```

> ありがとうございました！ 5000を3000に書き換えたら動きました。

7/16



Takashi Yamanoue

7月16日 22:44 · 2人

...

その後、geminiに、先ほどindex.jsから変換して作ってもらった [app.py](#) をクラスにして、他のpythonプログラムから起動したりメッセージを受け取ったりメッセージを送ったりできるようにして欲しいと頼んで、作ってもらいました。うまくいくかどうかは明日やってみます。

 、他7人 コメント1件

 いいね！

 コメントする

 送信

 シェア

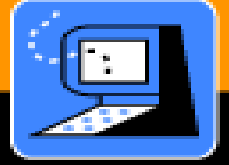
関連度の高い順 ▼



Takashi Yamanoue

色々ゴニョゴニョして、Line->BotはOKになりましたが、逆がまだうまくいきません。

...



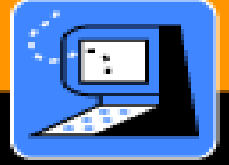
> app.py をclassにして、他のpythoプログラムから起動したり、lineからけ取ったメッセージを取り出したり、lineにメッセージを送ったりすることができるようにしてください。

◆ 承知いたしました。app.py をクラス化し、他の Python プログラムから起動、メッセージの取り出し、メッセージの送信ができるように リファクタリングします。新しい app.py では、LineBotApp というクラスを定義します。このクラスは Flask アプリケーションを内包し、LINE の Webhook 処理とメッセージ送信の機能を提供します。

変更点： * LineBotApp クラスの導入。

...

7/21



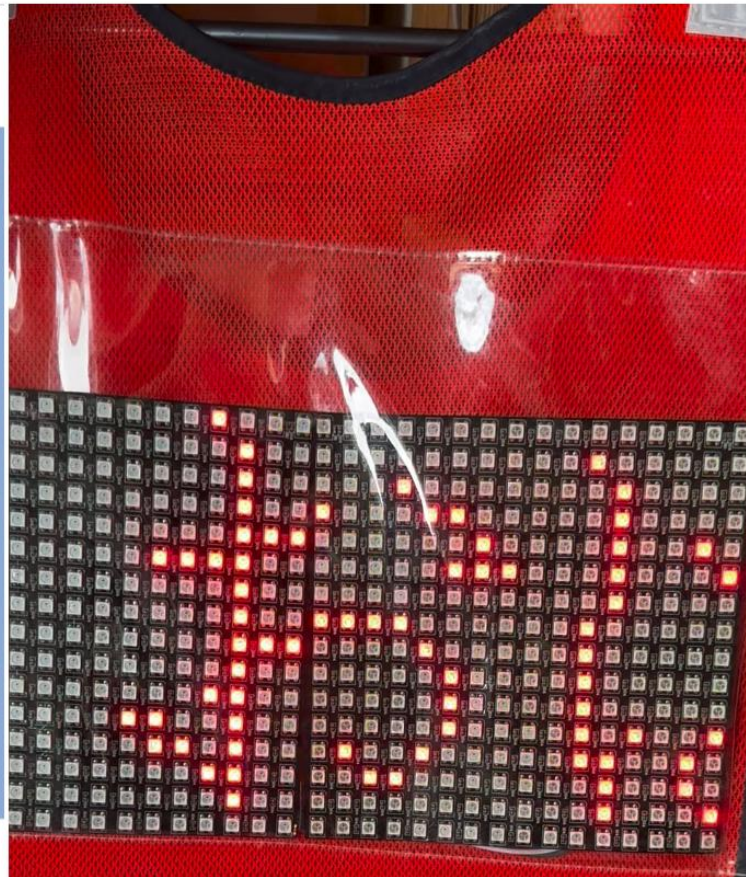
Takashi Yamanoue

7月21日 13:27 · 2人

LINEメッセージ->着る電光掲示板、表示、成功！

< お天気 Bot

🔍 📖 ☰



7/22

着る電光掲示板→LINEで悩む
→ Gemini CLIで解決



着る電光掲示板からLINEへメッセージを送りたいけどどうすればよいか悩む。そして…



Takashi Yamanoue

7月22日 8:52 · 2人

どうして良いかわからなくなったら、geminiに聞けば良いみたい。



Takashi Yamanoue

7月22日 10:46 · 2人

...

LINE bot-着る電光掲示板連携、LINE bot→着る電光掲示板は上手く行くのだけど、着る電光掲示板→LINE bot、push text messageを使って最初は上手くいっていたのだけど、200メッセージ/月の制限を超えたためか、送られなくなってしまいました。

7/22

着る電光掲示板→LINEで悩む → Gemini CLIで解決



＞ app.pyはLINEのメッセージをオウム返しするpythonのclassですが、このclassに、terminalから入力したメッセージをLINEに表示する機能を加えてください。

◆ 承知いたしました。app.pyにターミナルからの入力をLINEに送信する機能を追加しますね。まず、現在のapp.pyのコードを確認します。

現在のapp.pyのコードを以下のように変更します。主な変更点は次のとおりです。

1. ユーザーIDの保存: LINEでメッセージを受信するたびに、そのユーザーのIDを一時的に保存するようにしました。ターミナルからメッセージを送信する際の宛先として利用します。
2. Flaskサーバーのバックグラウンド実行: Webサーバー(Flask)をバックグラウンドのスレッドで実行するように変更しました。これにより、サーバーを動かしながら、ターミナルで文字入力の受付ができます。

7/22



Takashi Yamanoue

7月22日 14:21 · 2人



「Wikiで制御できる、場所に表示を変えることができる、着る電光掲示板（走るのもOK）」のLINE メッセージ表示機能追加、Raspberry Pi 4で動くようになった後、Raspberry Pi Zero にプログラムなどをコピーして、I2C 400Khz化やGPIOのUARTの設定による、Adafruit のUltimate GPS接続などを行って、動作成功！ この後は電源投入後の自動立ち上げの設定。





Takashi Yamanoue

スクリプト書いて、systemd設定して再起動かけたけど、上手く行きません。スクリプト、間違っていないと思うのだけど。

2週間 いいね！ 返信する



Takashi Yamanoue

プログラムの側、自動起動の時のプログラム内部の各部品の手続きのとき、line_botの起動部分が無限ループになっていて、その次に起動するはずのWikiページの読み込みの繰り返しのところに到達していませんでした。line_botの起動部分をthreadにして、自動起動できるようになりました。

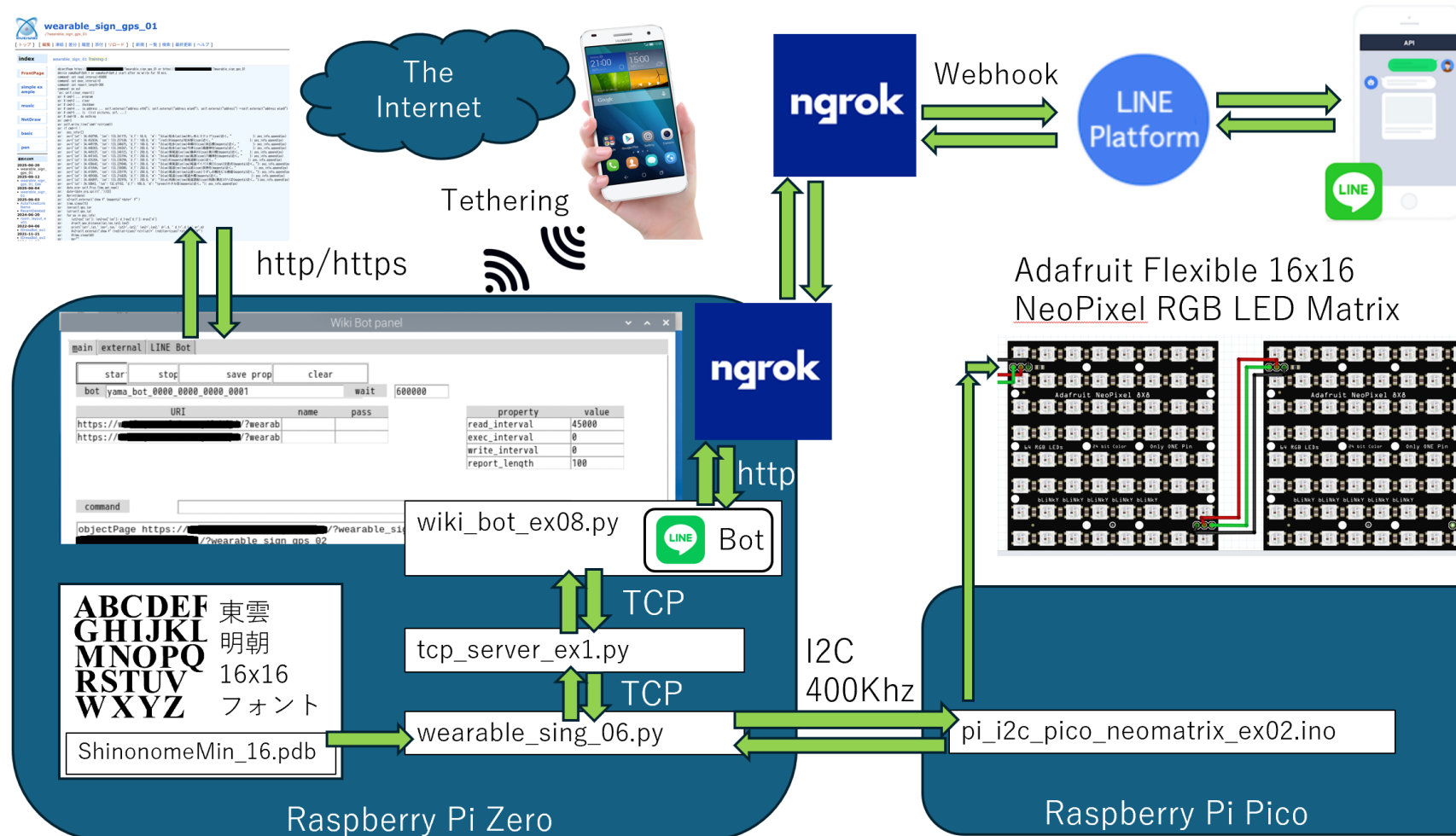
明日は、GPS機能の確認をやりたいと思います。

2週間 いいね！ 返信する 編集済み

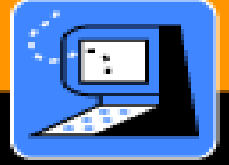


Takashi Yamanoue

GPS機能の確認、最初、未設定のところがあって動かなかったのですが、それを直したら、動きました。



感想



- ・ Vibe Codingは生産性をぐっと上げる可能性がある。
- ・ プログラムの構成の目論見がないとちょっと使いにくいかも。
- ・ 今回に限るかもしれないけど、（使う人が）プログラムが動く環境がわかっていないとダメっぽい。
- ・ 結構、細かい間違いを平気でする（ポート番号を教えたのに違うポート番号のプログラムを作成するなど）ので、プログラミングができない人には勧められないかも。

謝辞



- ・ IoTLT のびすけさん、LTS武村さん、
- ・ Facebook でアドバイスをいただいた方々
- ・ 利用したシステムやプログラムやデータ(東雲明朝フォントなど)を作成された皆様
- ・ 情報処理学会IOT研究会の皆様
- ・ ACM SIGUCCSの皆様



電子楽器の作品紹介2025

石垣 良

<https://risgk.github.io/>

2025年8月28日 SWEST27 EmbLT

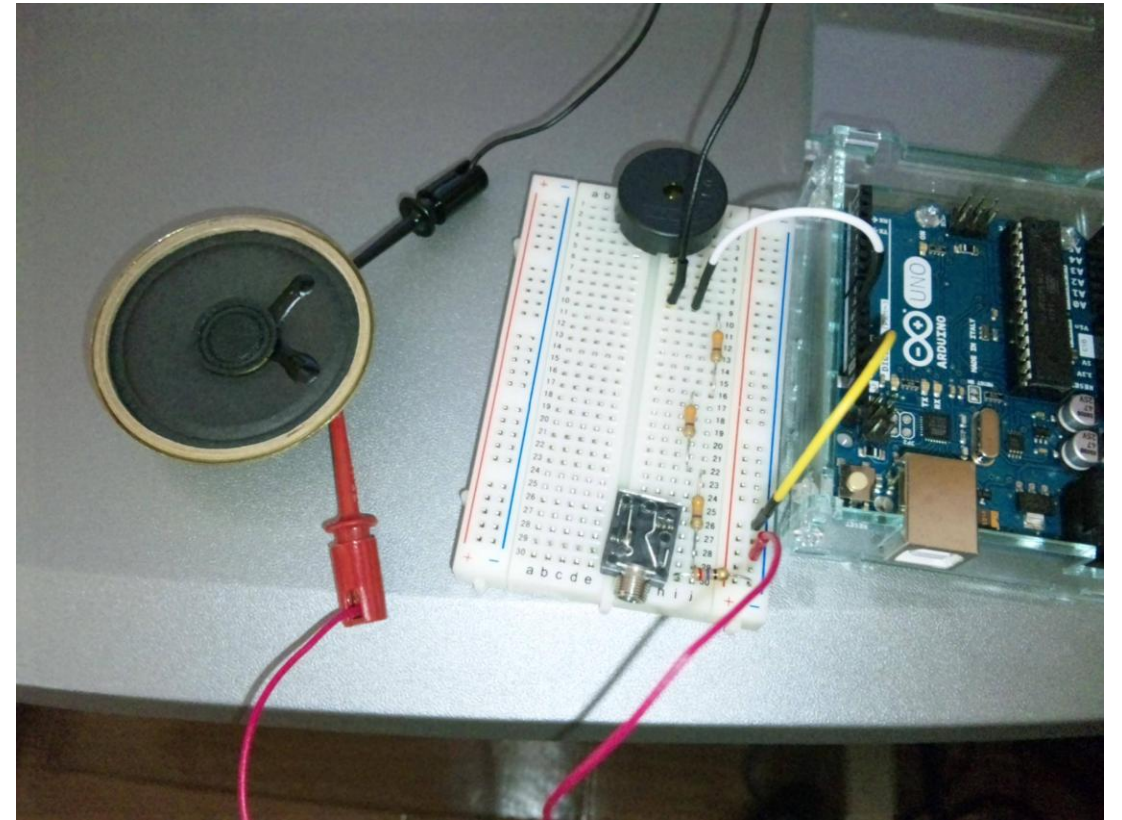
**電子楽器の作品紹介2022
(SWEST24 EmbLT)
の続編です**

以下、しばらく
電子楽器の作品紹介2022
からの再掲です

第1世代

#1 Digital Synth VRA8 (2014年)

- Maker Faireを見学して電子工
作を始めたくなり、派遣先で
「ものづくりクラブ」に参加し
たことが開発のキッカケ
- ピコピコ音を出そうとしていた
が、**IIRフィルター**でシンセサイ
ザーを実現できることに気付く
- Arduinoだけで音を合成する、
Virtual Analog Synthesizer
- **VA-8**という名前を考えていた
が、名前衝突しそうなので**R**を
追加



#2 Digital Synth WRA32 (2014年)

- Web Audio APIを使った、ブラウザで動くシンセサイザー
- **VRA8**制御用のWebアプリ (MIDIコントローラー) を拡張したもの
- フィルター用のスライダーで音色を変えると、音が止まってしまう (VRA8も同様)
- もうメンテしていないが、今ならもっと良いものが作れる気がする

Digital Synth WRA32 5.1.1

Virtual Analog Synthesizer Web App

Synth Settings

MIDI IN
Basic Channel
Volume

We recommend Google Chrome, which implements Web MIDI API

Controllers

VCO 1 Waveform <input type="range" value="0"/>	Mixer VCO 1 Level <input type="range" value="64"/>
VCO 1 Coarse Tune <input type="range" value="64"/>	Mixer VCO 2 Level <input type="range" value="64"/>
VCO 2 Waveform <input type="range" value="0"/>	Mixer VCO 3 Level <input type="range" value="64"/>
VCO 2 Coarse Tune <input type="range" value="64"/>	
VCO 2 Fine Tune <input type="range" value="70"/>	
VCO 3 Waveform <input type="range" value="2"/>	
VCO 3 Coarse Tune <input type="range" value="52"/>	
VCO 3 Fine Tune <input type="range" value="64"/>	
VCF Cutoff Frequency <input type="range" value="91"/>	FEG Attack Time <input type="range" value="21"/>
VCF Resonance <input type="range" value="51"/>	FEG Decay Time <input type="range" value="85"/>
VCF Envelope Amount <input type="range" value="12"/>	FEG Sustain Level <input type="range" value="127"/>
VCF Key Follow <input type="range" value="0"/>	FEG Release Time <input type="range" value="85"/>
AEG Attack Time <input type="range" value="21"/>	
AEG Decay Time <input type="range" value="85"/>	
AEG Sustain Level <input type="range" value="127"/>	
AEG Release Time <input type="range" value="85"/>	

Software Keyboard

C#3	D#3	F#3	G#3	A#3	C#4	D#4	F#4	G#4	A#4	C5				
C3	D3	E3	F3	G3	A3	B3	C4	D4	E4	F4	G4	A4	B4	C5

Transpose

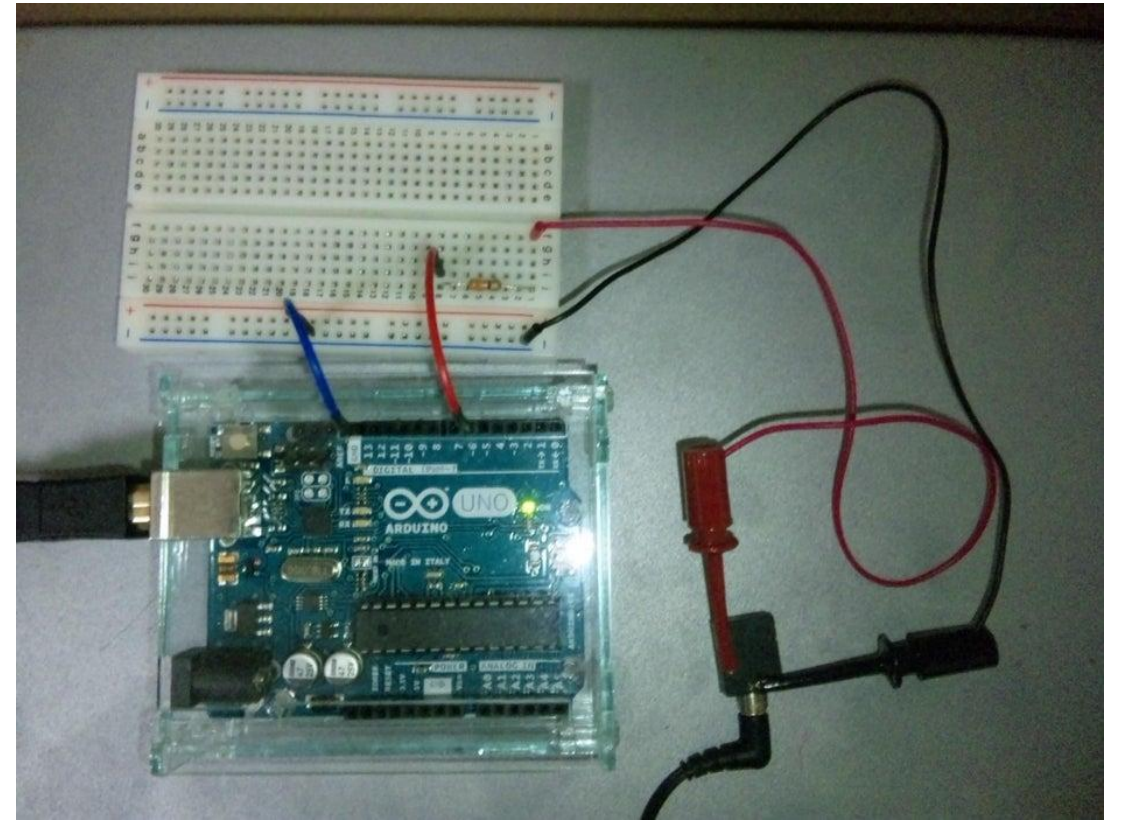
About

Made by ISGK Instruments ([View on GitHub](#))

第2世代

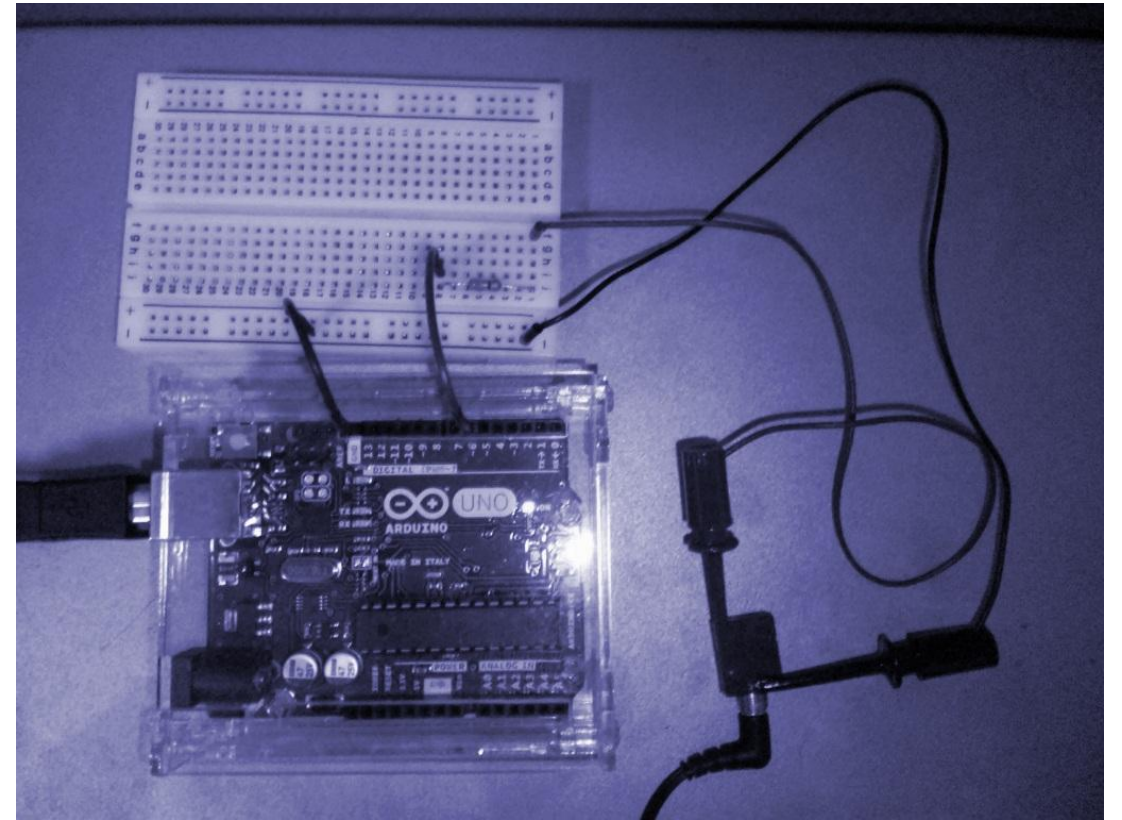
#3 Digital Synth VRA8-M (2015年)

- Monophonic SynthesizerのM
(ノコギリ波と逆ノコギリ波を
重ねたイメージも)
- 「無印」で関数がインライン展
開されていたことに気付
いたことが開発のキッカケ？
(Arduinoでは**always_inline**指
定が必要)
- 「ものづくりクラブ」メンバー
として、Maker Faireに初出展
- なかなか評判が良かったので、
ちょっと調子に乗る



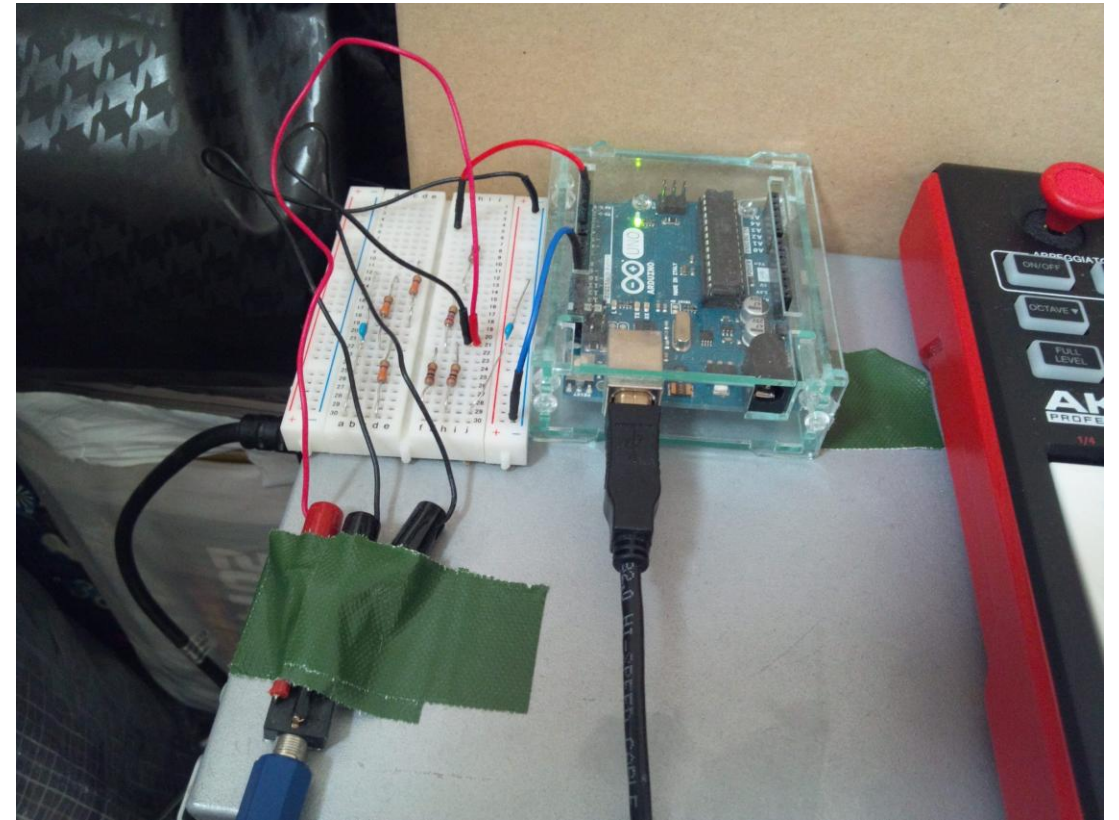
#4 Digital Synth VRA8-X (2015年)

- Experimental SynthesizerのX (シリーズの終わりを考えていたのかも?)
- 浜松市のイベントに間に合わせるように開発
- 色々な波形合成の「実験台」にしようと考えていたが、結局メンテせず
- ちょっと完成度が低いかも...
- シンボルカラーは「青」



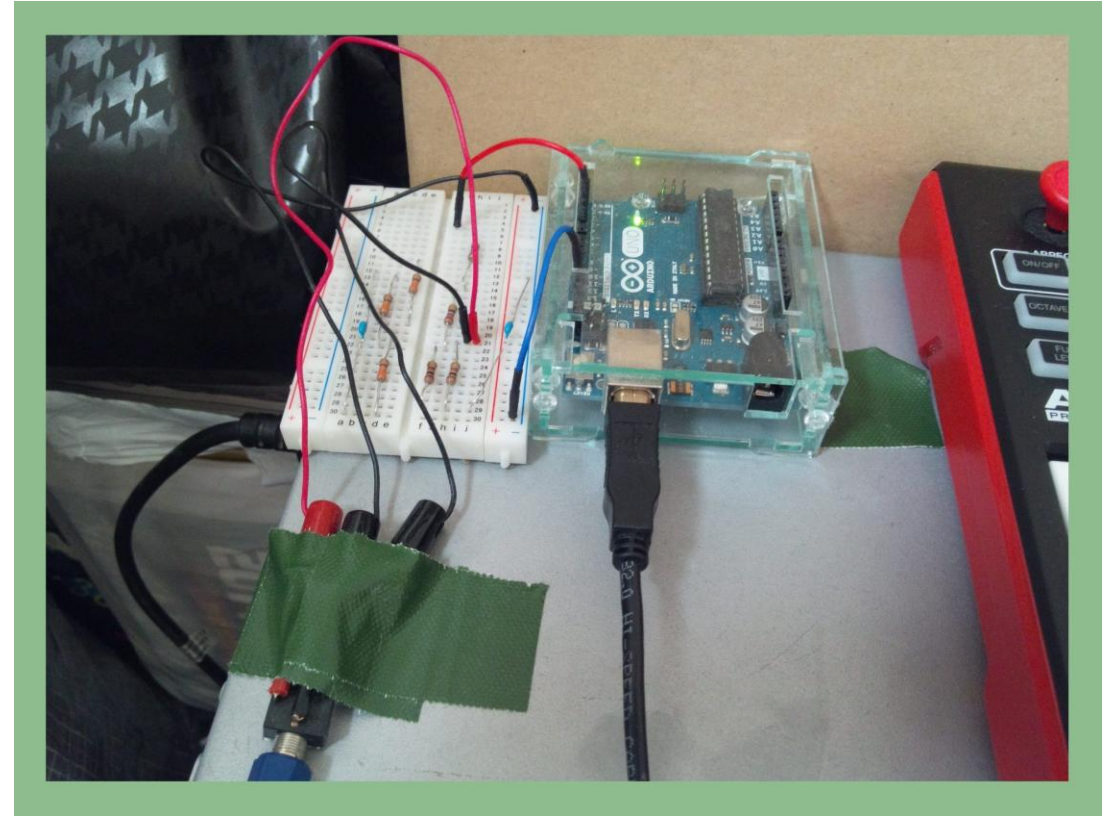
#5 Digital Synth VRA8-P (2016年)

- **P**seudo **P**olyphonic Synthesizerの**P**
- 3和音を出せるが、音質を変えるフィルターが1個なので「疑似」ポリフォニック（パラフォニックとも言う）
- 派遣先が変わったので「ものづくりクラブ」は卒業、この年からずっと個人サークルでMaker Faire Tokyoに出展
- SWEST自由工作発表（初回）



#6 Digital Synth VRA8-Px (2017年)

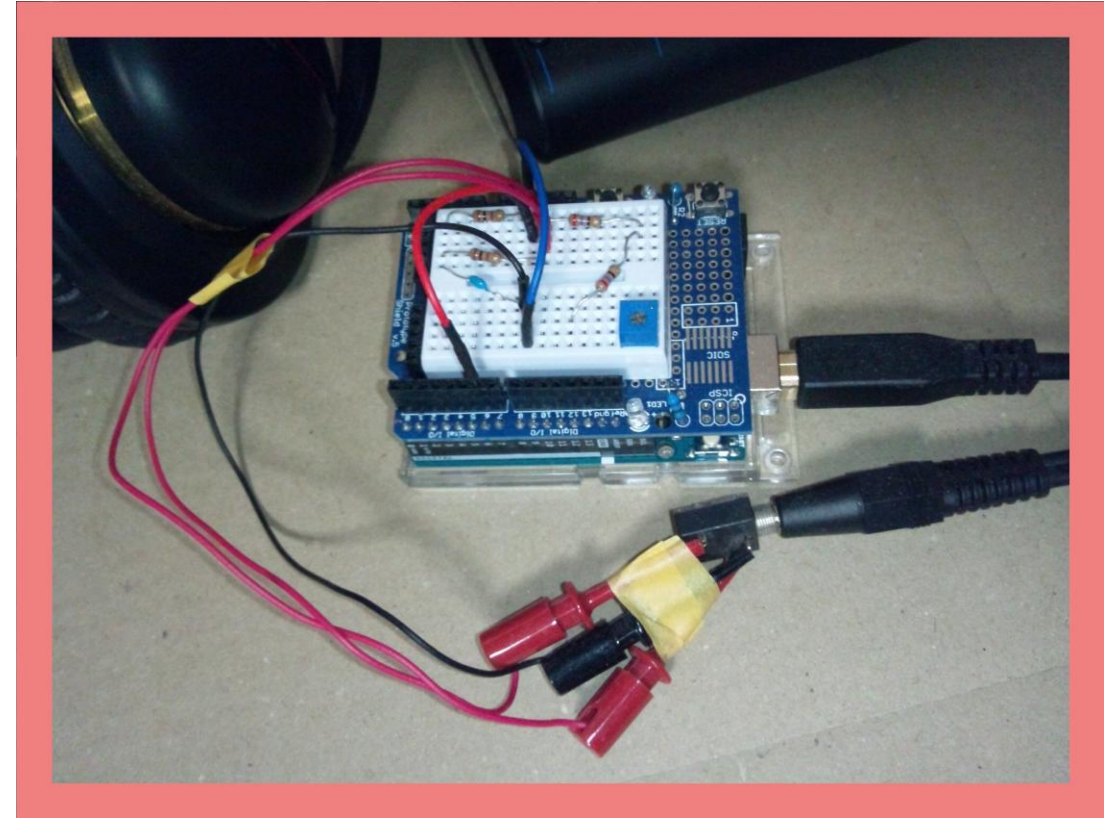
- Paraphonic SynthesizerのPにプラス (x)
- 会社のマイコン研修の素材にArduinoを使ってみたら、それまで**Timer0割り込みを無効にしていなかった**ことに気付く
- **noInterrupts()** を呼んだら、CPU使用率が激減、かつ安定
- それまで**LED点灯でオーバーロード判定**していたが、この作品から**CPU使用率をMIDIデータでデバッグ出力** (計測重要！)



第3世代

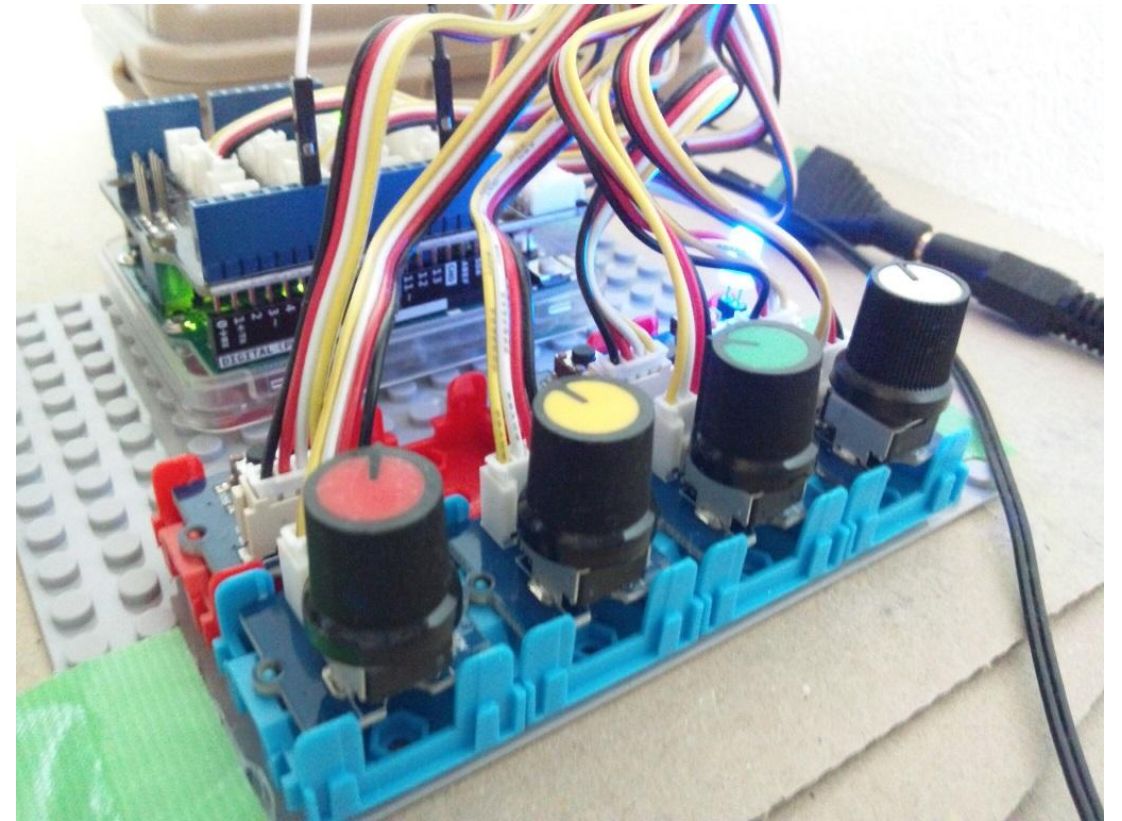
#7 Digital Synth VRA8-N (2018年)

- Monophonic SynthesizerのN
(Mの次とか、ノコギリ波のイメージもある)
- **サンプリング周波数**が15.625 kHzから**31.25 kHz**に倍増 (音質が大幅アップ!)
- もちろん、CPU使用率も倍増したので、頑張って「無駄」を省いて効率化
- SWEST20 インタラクティブセッション **ベストプロダクト賞**
ブロンズ



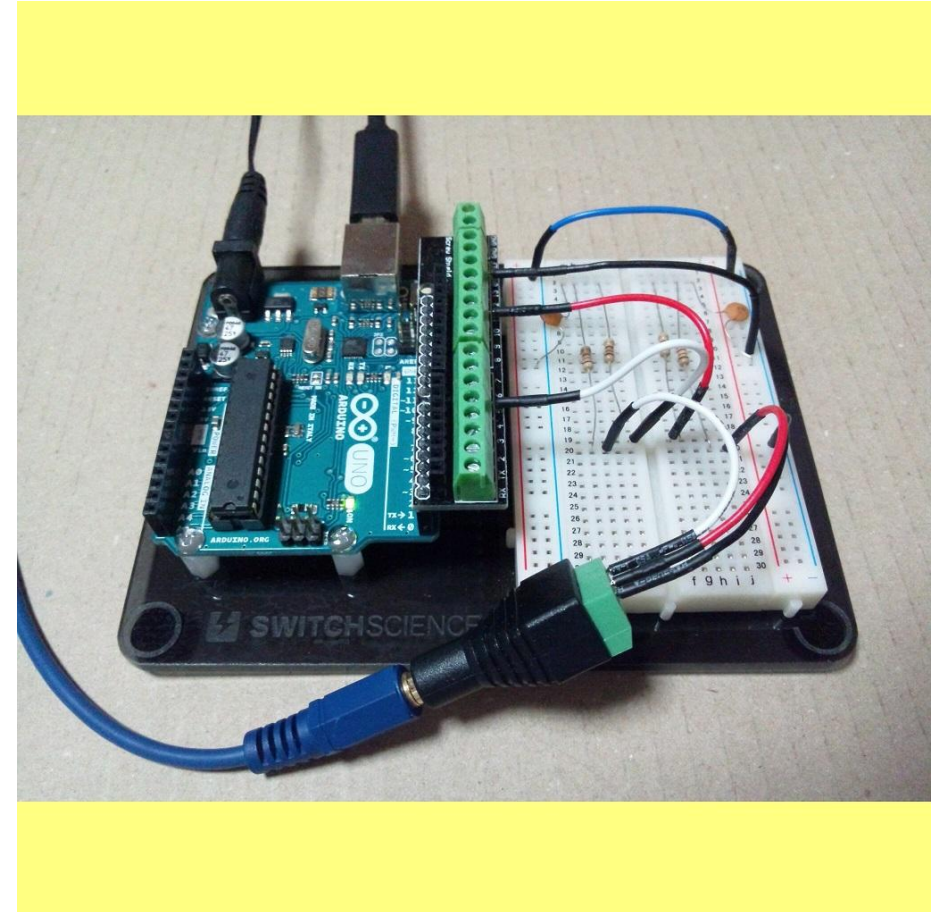
#7（その2）VRA8-N mode-VC（2019年）

- **VRA8-N v2**を電圧制御（Voltage Control）可能にしたモードで、例えば可変抵抗で演奏可能
- モジュラー・シンセサイザーとも接続可能
- **SWEST21 インタラクティブセッション ベストプロダクト賞 シルバー**



#8 Digital Synth VRA8-Q (2020年)

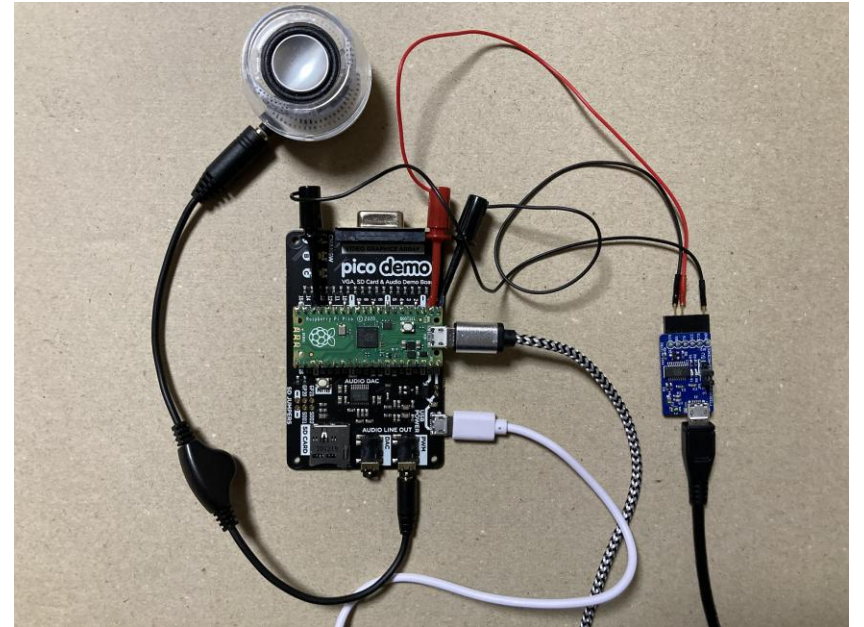
- Quadraphonic SynthesizerのQ
(Pの次のイメージもある)
- サンプリング周波数：31.25 kHz
- **4和音対応** (パラフォニック)
- **ステレオ・コーラス・エフェクター内蔵** (コーラス用のショート・ディレイ・バッファのサイズは512 Byte)
- SWEST22 インタラクティブセッションで発表



#9 pico_synth (2021年)

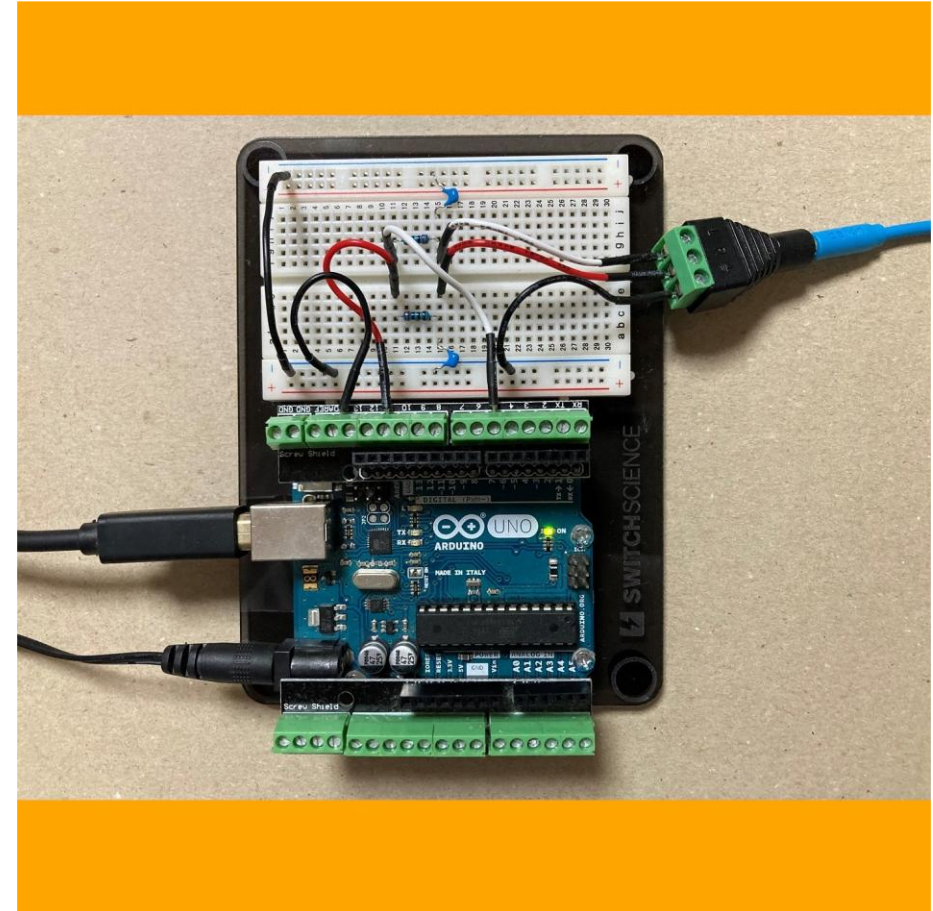
#9.1 pico_synth_ex (2021年)

- 『Interface 2021年8月号』のラズベリー・パイPico特集記事のために試作したシンセサイザと、それを拡張した4和音対応（**ポリフォニック**）シンセサイザ
- 前者は、1オシレータ + 1フィルタ + 1アンプというシンプルな構成
- 後者は、SWEET23 インタラクティブセッション **ベストプロダクト賞 ゴールド**（しかし、v0.1止まり）
- ラズパイPicoは、今後本格的に使いたい



#10 Digital Synth VRA8-U (2022年)

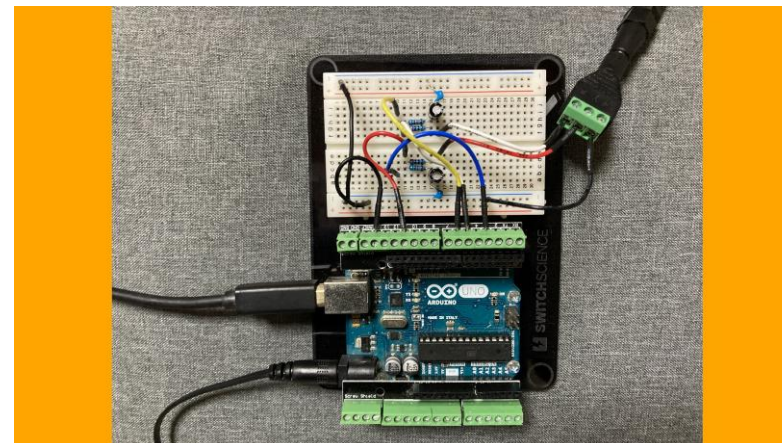
- VRA8シリーズの集大成
- 単音シンセ「VRA8-N」と4和音シンセ「VRA8-Q」を合体させたような仕様
- Monophonic/Paraphonic **U**nited Synthesizer
- パルス波形では、パルス幅変調 (PWM) にも対応
- SWEST24 インタラクティブセッションで発表
 - 【追記】 **ベストプロダクト賞 ゴールド**
 - 【追記】 その後、Maker Faire Tokyo 2022に出展、ライターの藤本健さんに取材・製作していただく



ここから新しい作品です

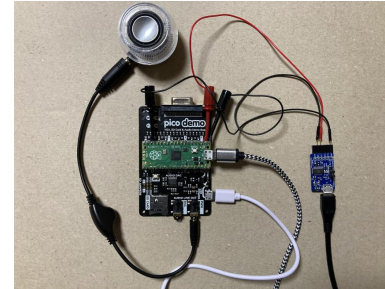
#10.1 Digital Synth VRA8-U type-16 (2023年)

- 「VRA8-U」の16ビット出力版（音質アップ）
 - **デュアルPWM出力**を利用
 - もともと、内部処理16ビット、PWM出力8ビットだった
- 2024年には、「**シンセカイリアルSP2**」というイベントで、ライブハウス「池袋LIVE INN ROSA」で音出し（！）



#9.2 pico_synth 2023 Edition (2023年)

- 『ラズベリー・パイPico/Pico W攻略本』（CQ出版社、2023年）の書き下ろし記事ために制作
 - 「第6部第2章：リアルタイム処理のために軽量化！シンセサイザの製作」
 - 「第6部第3章：音の時間変化に対応したシンセサイザ作り」（書き下ろし）
- 和音には非対応だが、「pico_synth_ex」（2021年）よりも各モジュールの設計・実装を練ることができた
 - 特に、リアルタイム性にこだわり



第4世代

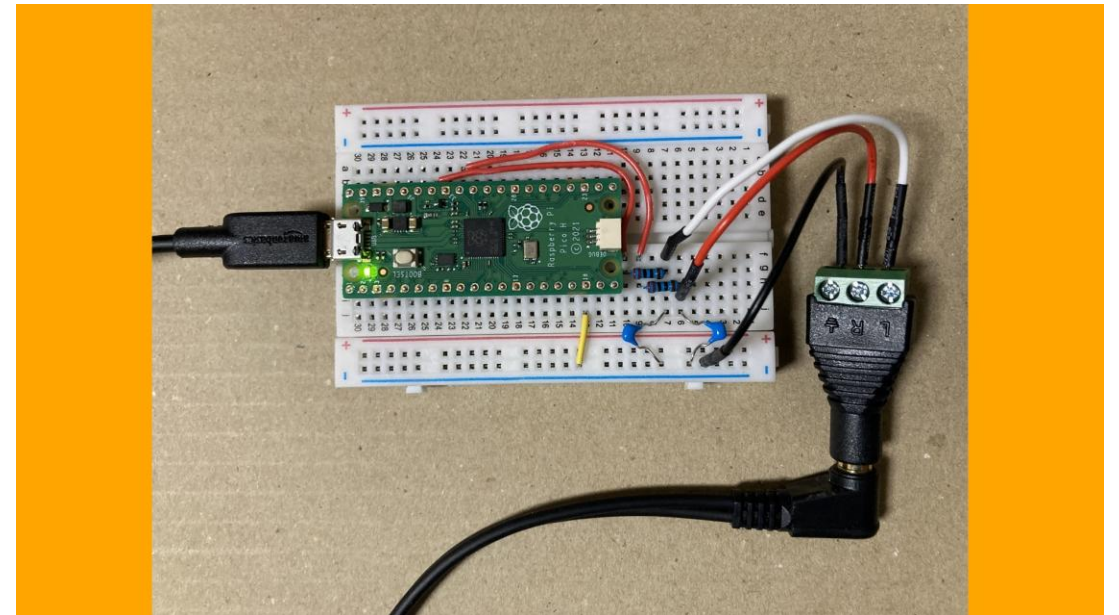
#11 Digital Synth PRA32-U (2023年)

- 「VRA8-U (type-16)」を**ラズパイPico**に移植、改良したシンセサイザー (USB MIDI音源)
 - UARTやMIDI制御にも対応可能
 - シンボルカラーは「オレンジ」のまま (変えれば良かった)
- 開発環境は**Arduino-Pico** (by Earle F. Philhower, III)
 - オーディオ用 (I2S) DAC基板を簡単に扱える
- SWEST25インタラクティブセッションで発表
 - 関連記事: 『Interface 2023年12月号』「**SWEST25参加レポート**」



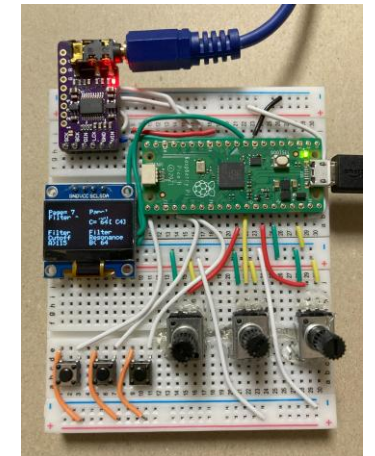
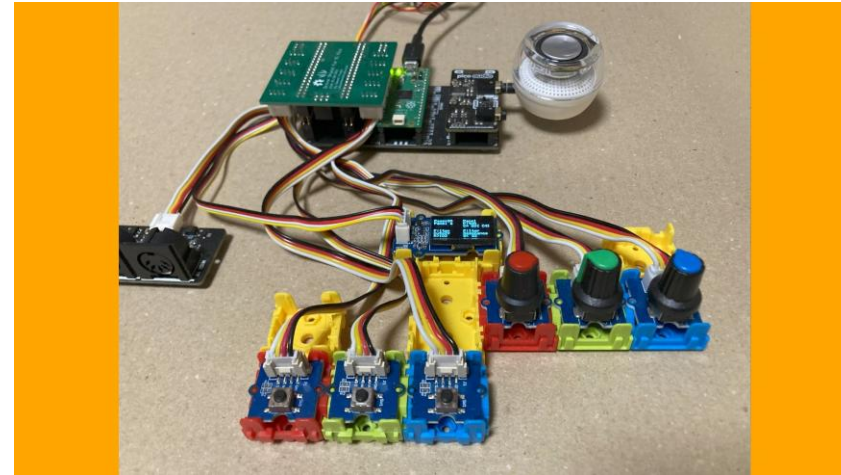
#11.1 Digital Synth PRA32-U PWM Audio Edition (2024年)

- 「PRA32-U」のPWMオーディオ出力版
 - Arduino-PicoのPWM Audio機能を使用
 - DAC基板が不要（低コスト）
 - ノイズを減らすため、RT6150 (PMIC) Power Save PinをHighに
- SWEST26インタラクティブセッションで発表
- 最近、「周期的に信号の不連続が発生、高域やサイン波でクリックノイズが目立つ」問題を発見（詳細は未調査）



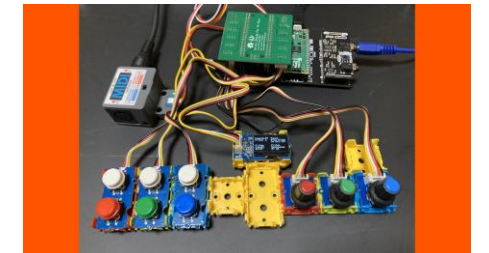
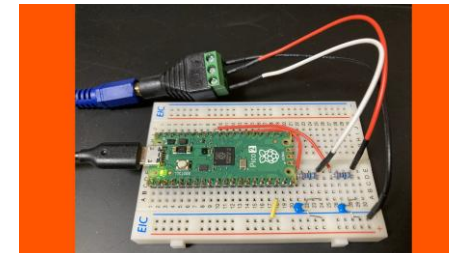
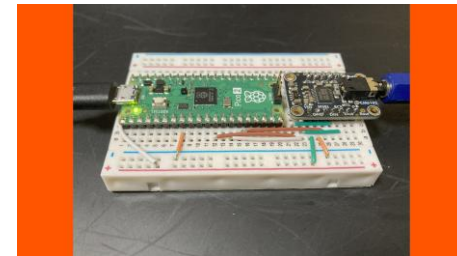
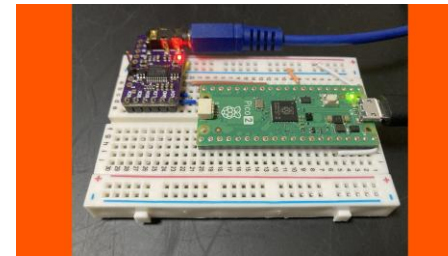
#11.2 Digital Synth PRA32-U with Panel (2024年)

- 「PRA32-U」のコントロールパネル統合版
 - ノブ、キー、OLEDディスプレイ搭載
 - 単体で音色編集や簡易演奏が可能
- ある有名シンセ開発者の方のアドバイス、「PCを介さないでパラメータをエディットできるシンセを開発してほしい」が制作のキッカケ
 - USB MIDI Host対応は未実現
- SWEST26インタラクティブセッションで発表
- その後、シーケンサー機能も追加



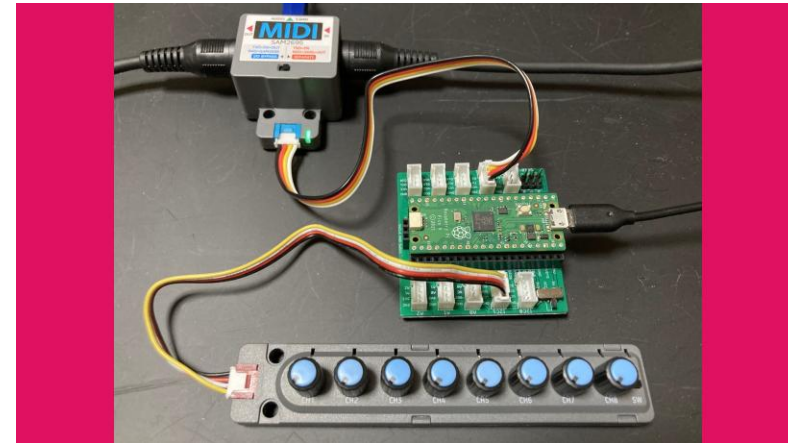
#12 Digital Synth PRA32-U2 (2025年)

- 「PRA32-U」をラズパイPico 2に移植、改良したシンセサイザー
 - シンボルカラーは「濃いオレンジ」
- 歴代最高音質
- 7つのノコギリ波を重ねた波形（スーパー・ソウもどき）や、矩形波の複雑な変形（オシレーター・シンクもどき）に対応
 - まだ、MCUリソースには余裕がある
- SWEST27 インタラクティブセッションで展示（おまけ）
 - NT東京（2日目）、Maker Faire Tokyo 2025に出展予定



#13 MIDI Controller PRMC-1 (2025年)

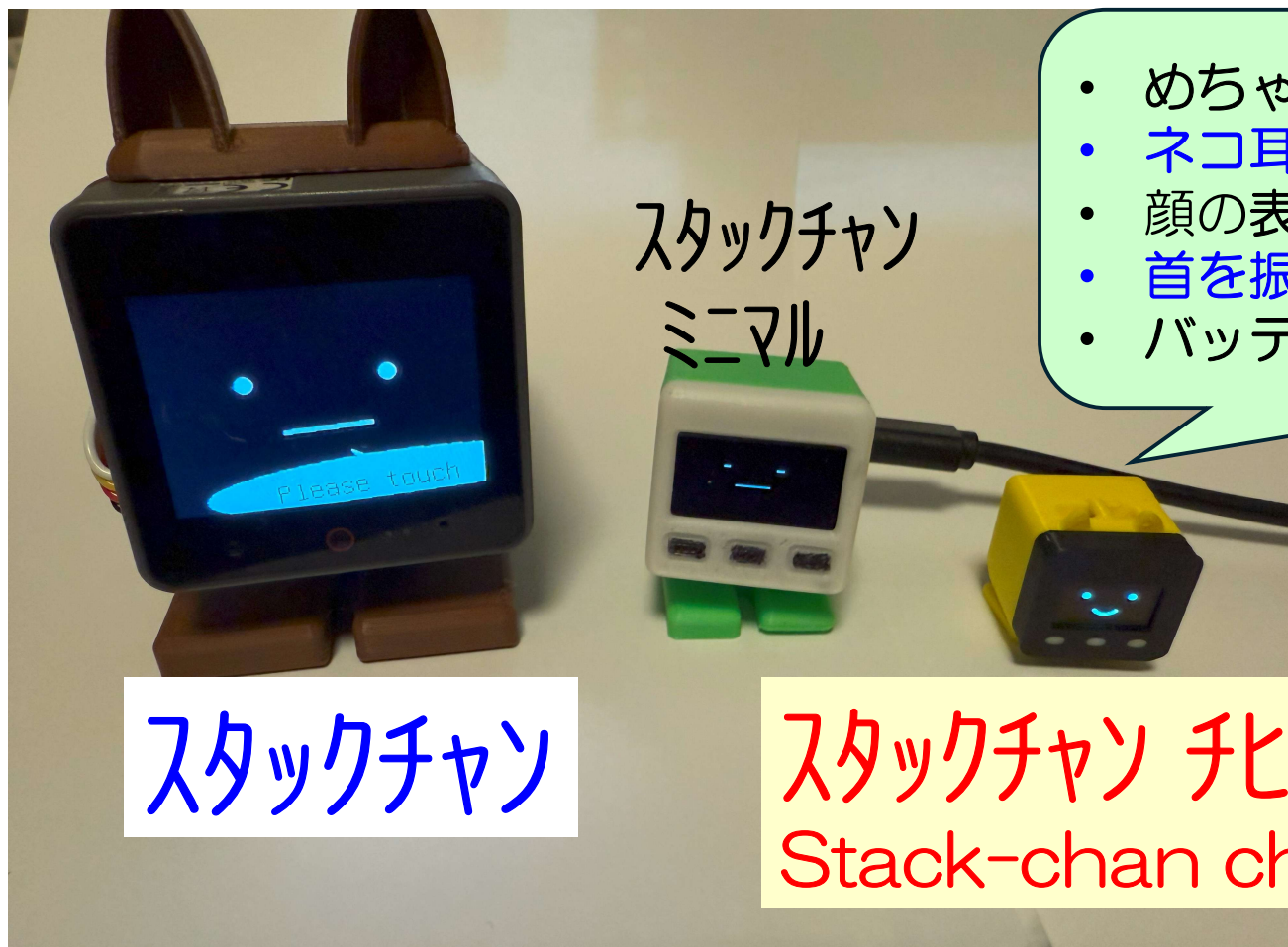
- 4つの和音をアルペジオ（分散和音）として順番に鳴らすシーケンサー機能を搭載
 - 「マシンライブ」と呼ばれる音楽パフォーマンスで、自作シンセを演奏するために制作
 - 「シンセカイリアル」がキッカケ
- コードは**Ruby**のみ、約400行
 - **PicoRuby**（**mruby/c**を用いたラズパイPicoなど向けRuby処理系）を使用
- 最初のバージョンを「**RubyKaigi 2025**」のLTで発表
- SWEST27 インタラクティブセッションで発表



<https://youtu.be/Eq25Ze3kTH4>

To be continued...

小さく作ってかわいいを実現 Stack-chan chibi (スタックちゃん ちび)



スタックちゃん
ミニマル

スタックちゃん

スタックちゃん ちび
Stack-chan chibi

- めちゃくちゃ小さいのはかわいい！
- ネコ耳がかわいい！
- 顔の表情が変わるのはかわいい！
- 首を振るのはかわいい！
- バッテリーで動くのはすごい！

- どんな表情をするのか
気になりますか？
- どんな動きをするのか
気になりますか？
- 中身がどうなっているか
気になりますか？

いろいろな工夫が
てんこ盛りです。

全部解説します！ぜひ「W-3に◎」を投票ください

小さく作ってかわいいを実現 Stack-chan chibi（スタックちゃん ちび）

【自己紹介】

- 大栄 豊（おおえ ゆたか）

- <https://www.facebook.com/yutaka.ohe>

- 元デンソー社員

- 資格：情報処理1種、2種（学生時代に取得）
エンベデッドスペシャリスト

- その他：

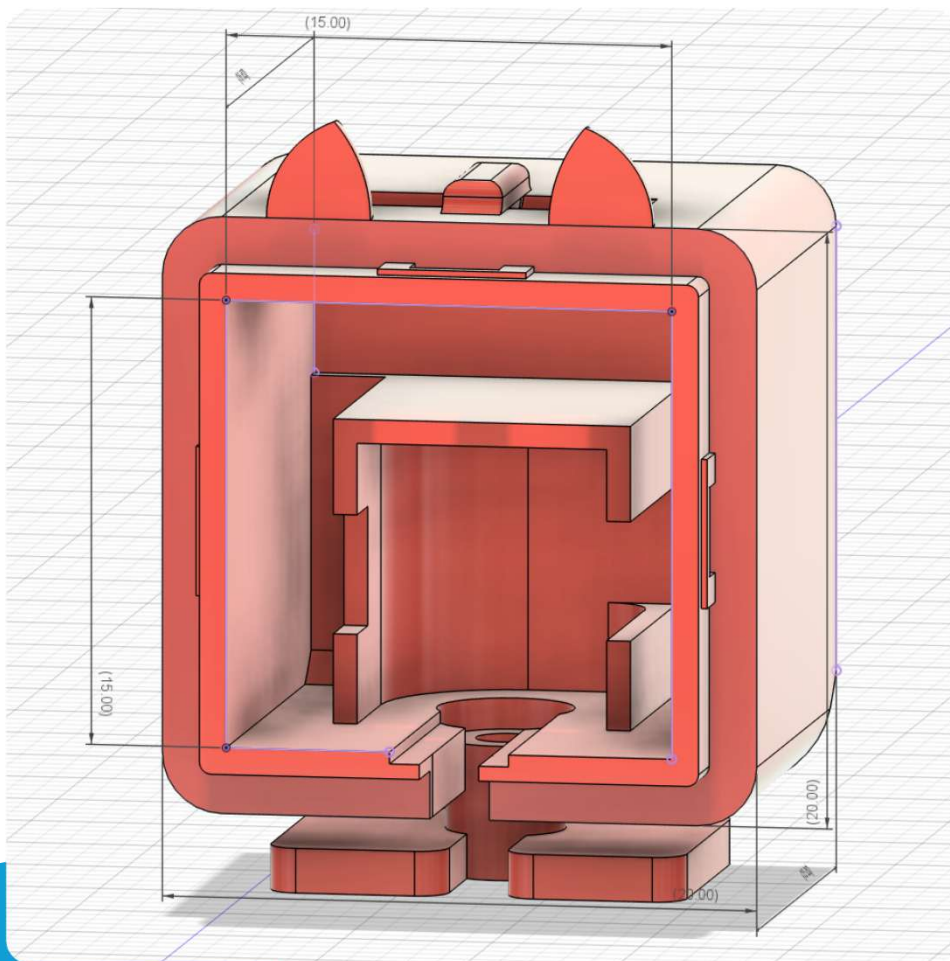
- **SWEST/LED-Camp実行委員**（2015～）

- IoTあるじゃん運営委員（2015～）



まあ、話の長い年寄りです m(_ _)m

Stack-chan chibi 内外形寸法



- 外形 (※)
 - 縦：23mm(足部含む), 20mm(足部除く)
 - 横：20mm
 - 奥行き：20mm
- 内形
 - 縦：15mm
 - 横：15mm
 - 奥行き：16mm

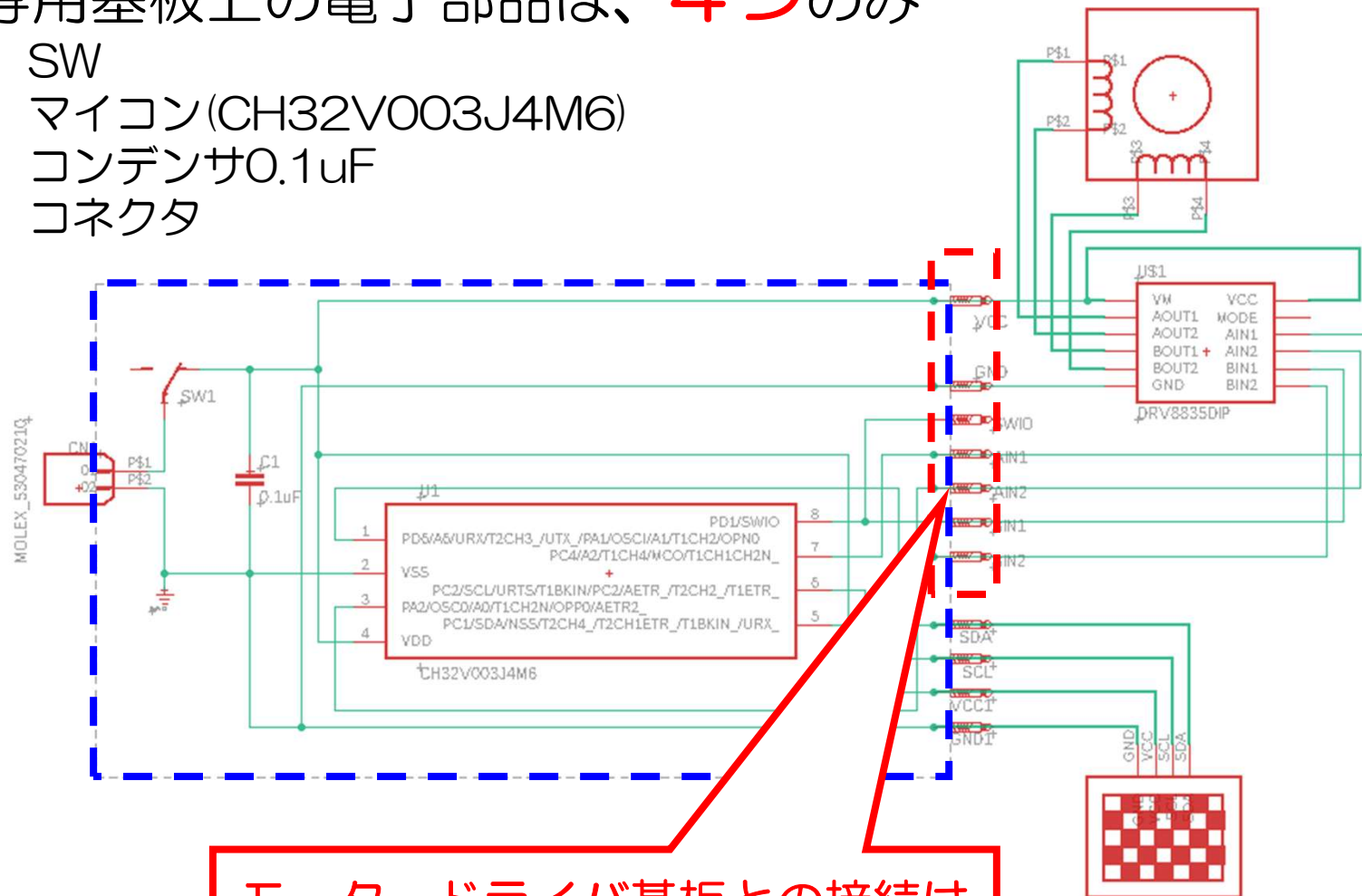
外形体積で1/25

※) Stack-chanは、縦70mm(足部含む)、横55mm、奥行き60mm

Stack-chan chibi 回路図

専用基板上の電子部品は、**4つ**のみ

- SW
- マイコン(CH32V003J4M6)
- コンデンサ0.1 μ F
- コネクタ

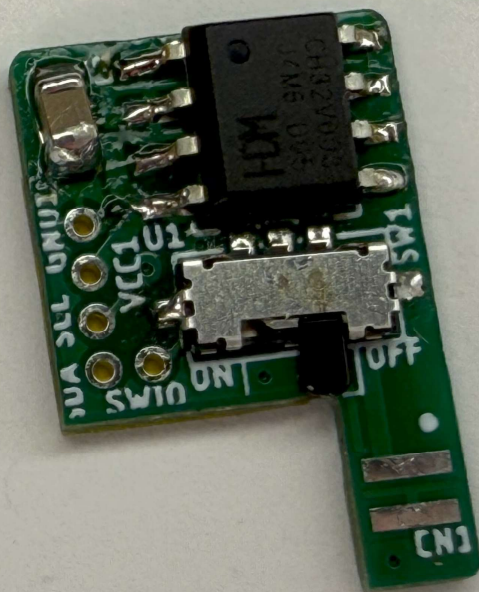
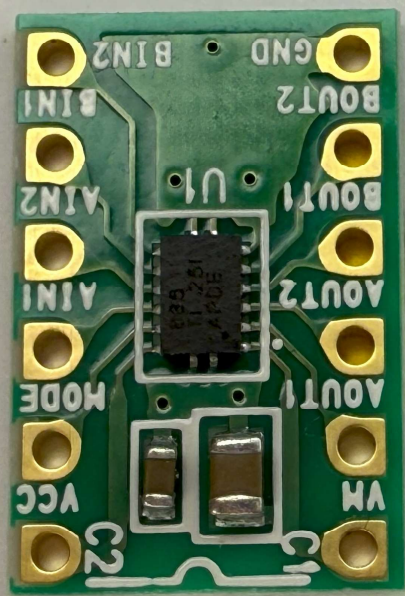


モータードライバ基板との接続は
はんだ付けで貼り合わせ

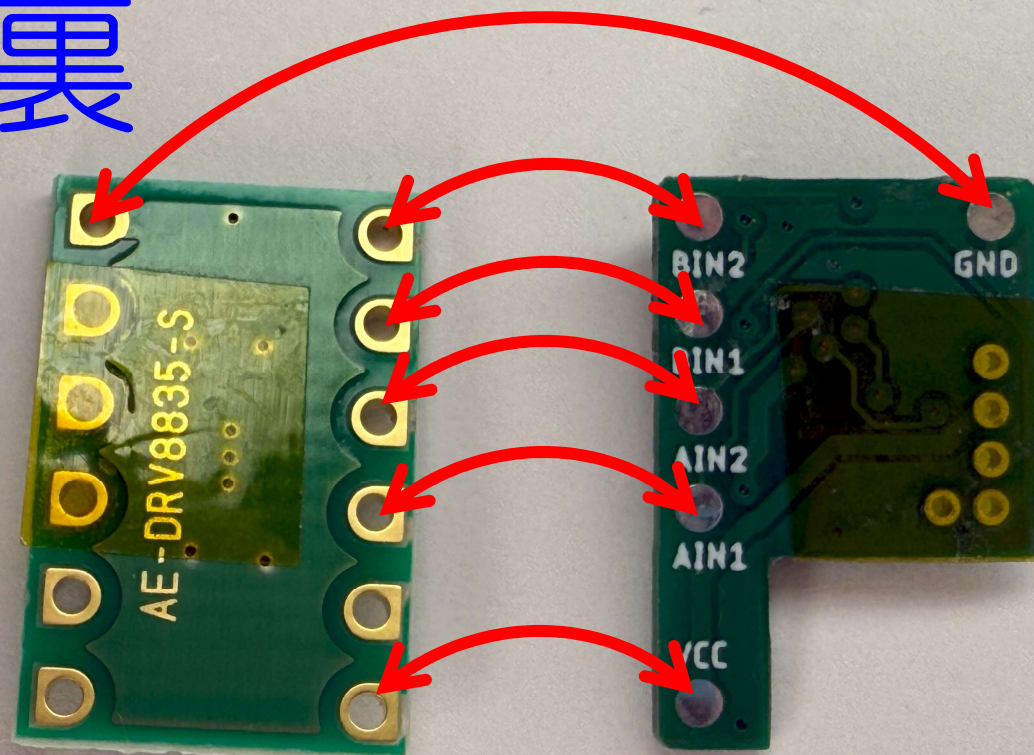
モータドライバ駆動信号、VCC、 GNDの接続

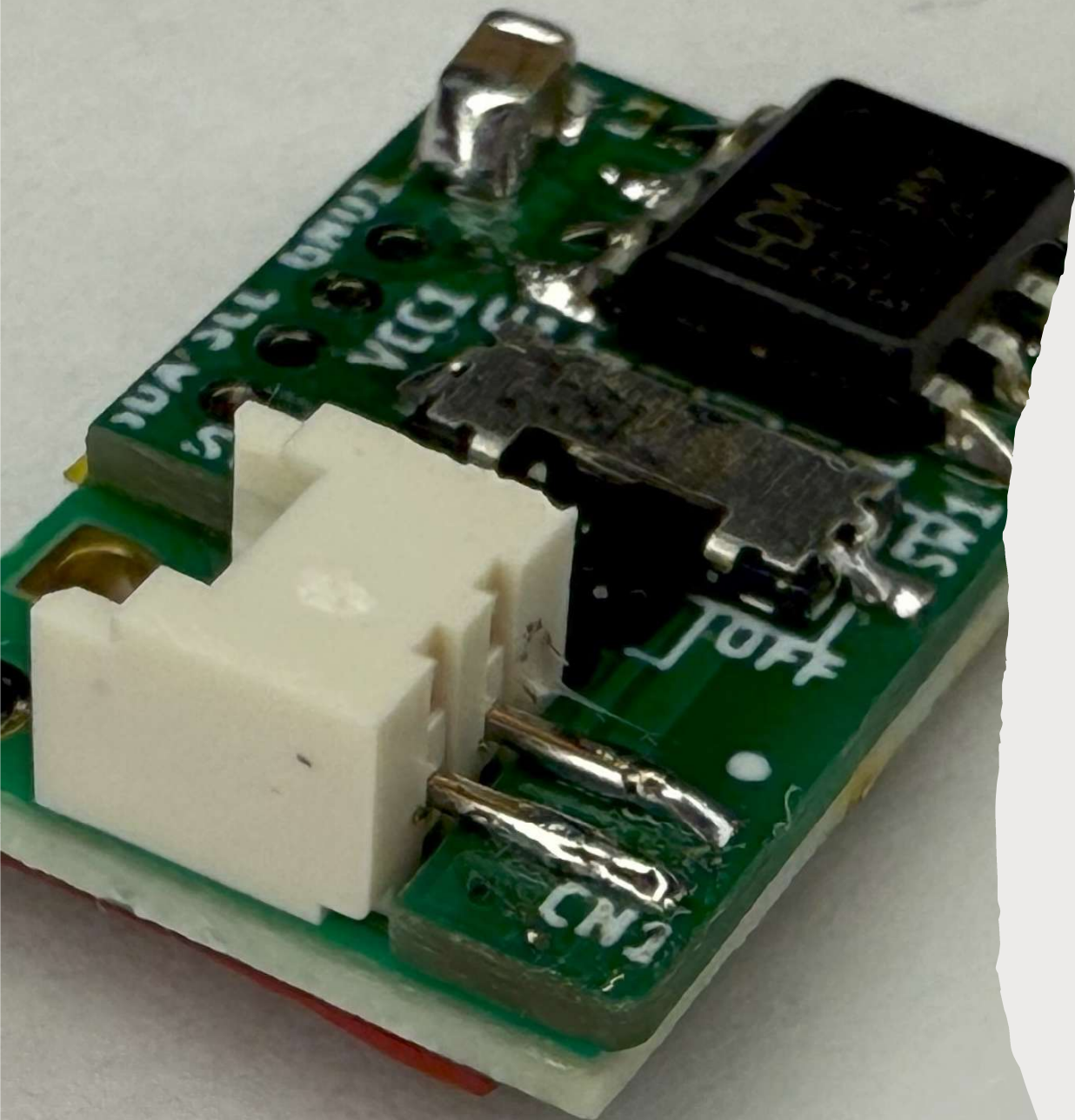
はんだ付けで貼り合わせ

表



裏



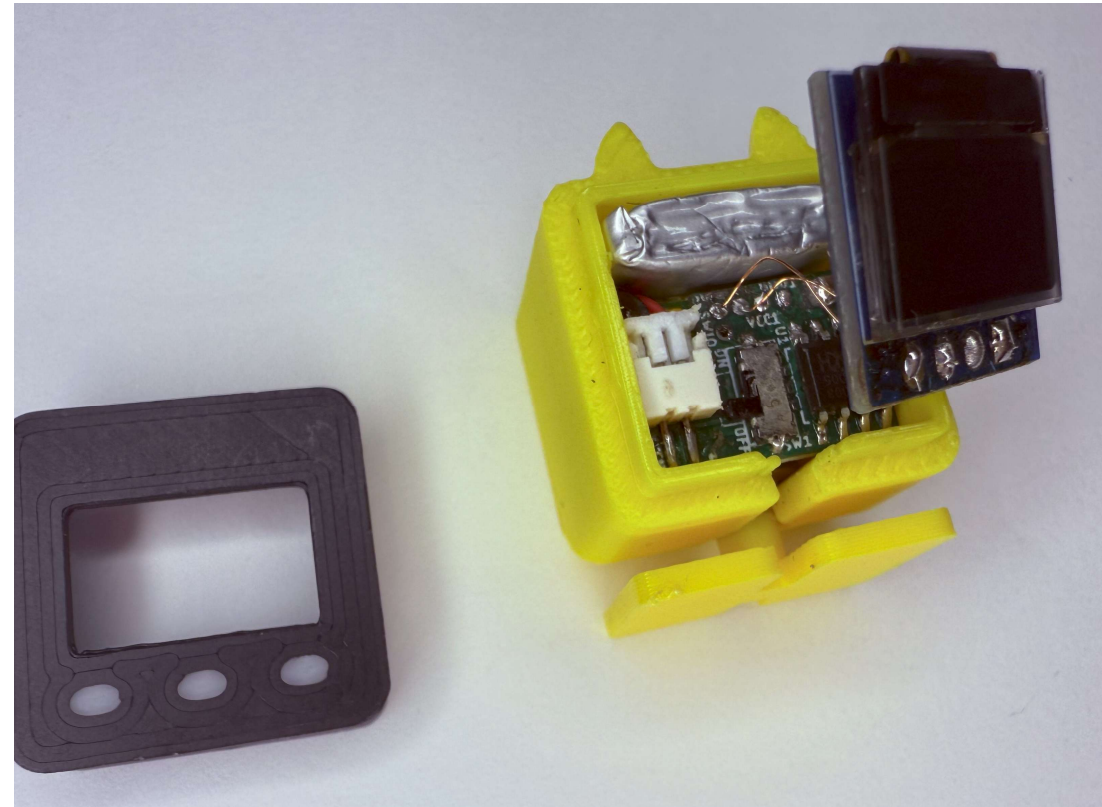
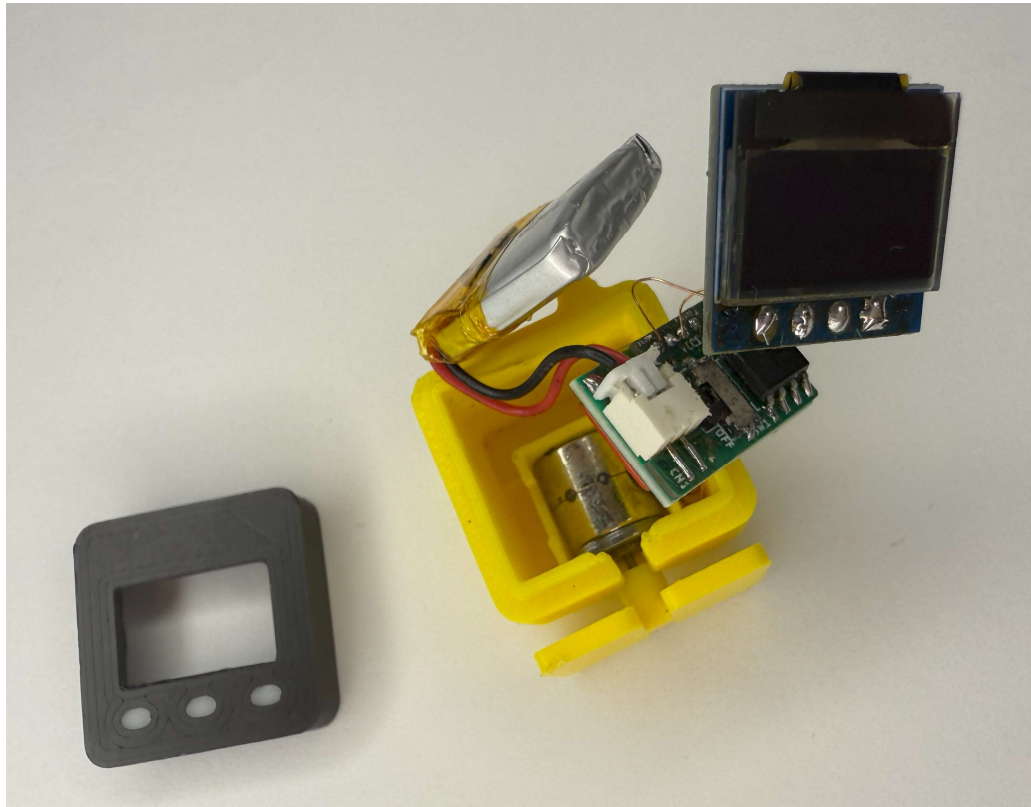


コネクタ実装

- 縦型コネクタを寝かせて、SMD部品のように実装
- P版の厚さをハウジングの高さに合わせて調整

組立て

リチウムポリマー電池は、コネクタから外して再充電可能



マイコンCH32V003J6M4

- 動作電圧
 - 2.7V~5.5Vの広い電圧範囲
→ リチウムポリマー電池（3.7V~4.2V）を網羅
専用の電源回路不要

3.3.1 Operating conditions

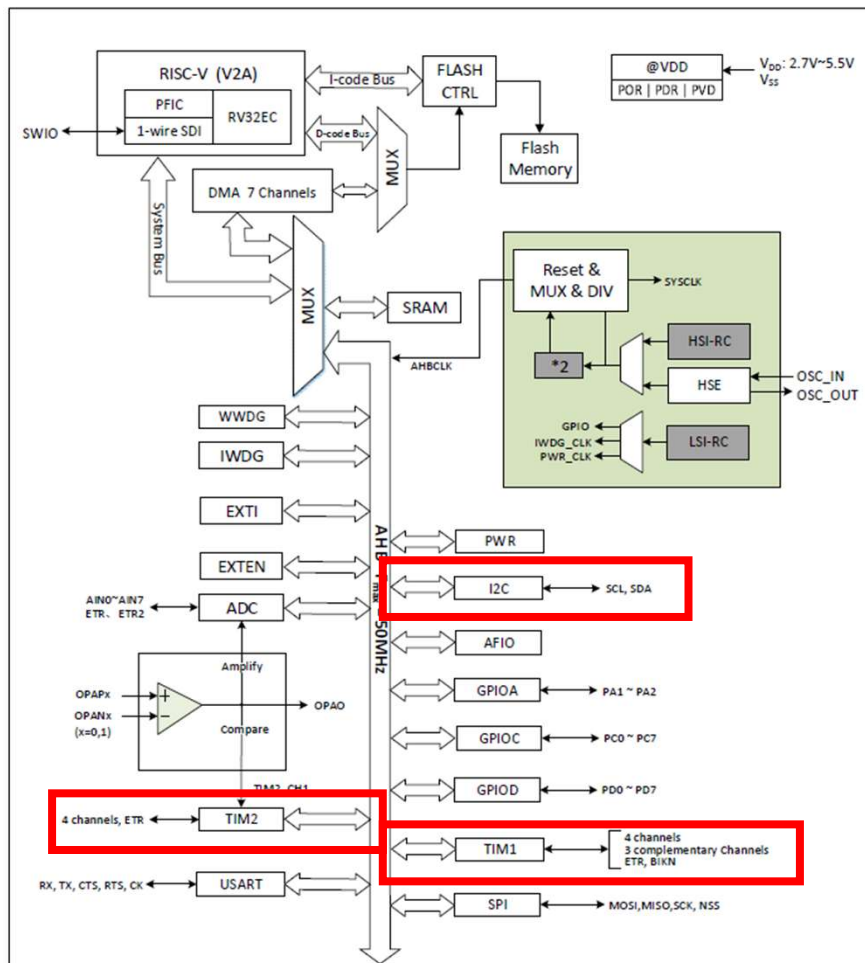
Table 3-2 General operating conditions

Symbol	Parameter	Condition	Min.	Max.	Unit
FHCLK	Internal AHB clock frequency			50	MHz
VDD	Standard operating voltage	ADC not used	2.7	5.5	V
		Use ADC (recommended)	2.8	5.5	
TA	Ambient temperature		-40	85	°C
TJ	Junction temperature range		-40	85	°C

- ROM : 16Kbyte
- RAM : 2Kbyte

マイコンCH32V003J6M4

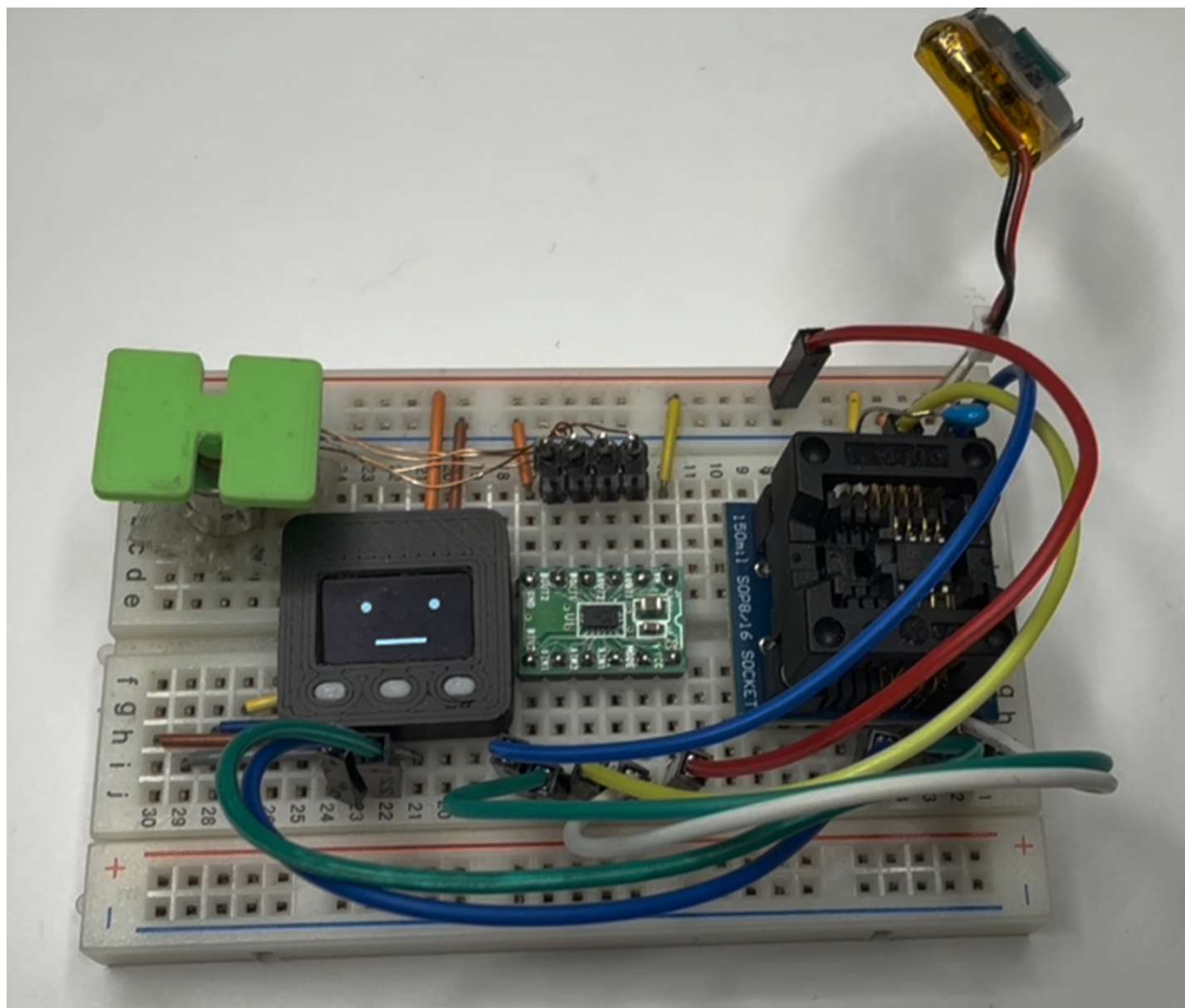
Figure 1-1 System block diagram



#	FUNCTION
1	T2CH3(PD6)
2	VSS
3	T1CH2N(PA2)
4	VDD
5	I2C_SDA
6	I2C_SCL
7	T1CH4(PC4)
8	T2CH4(PD5) / SWIO

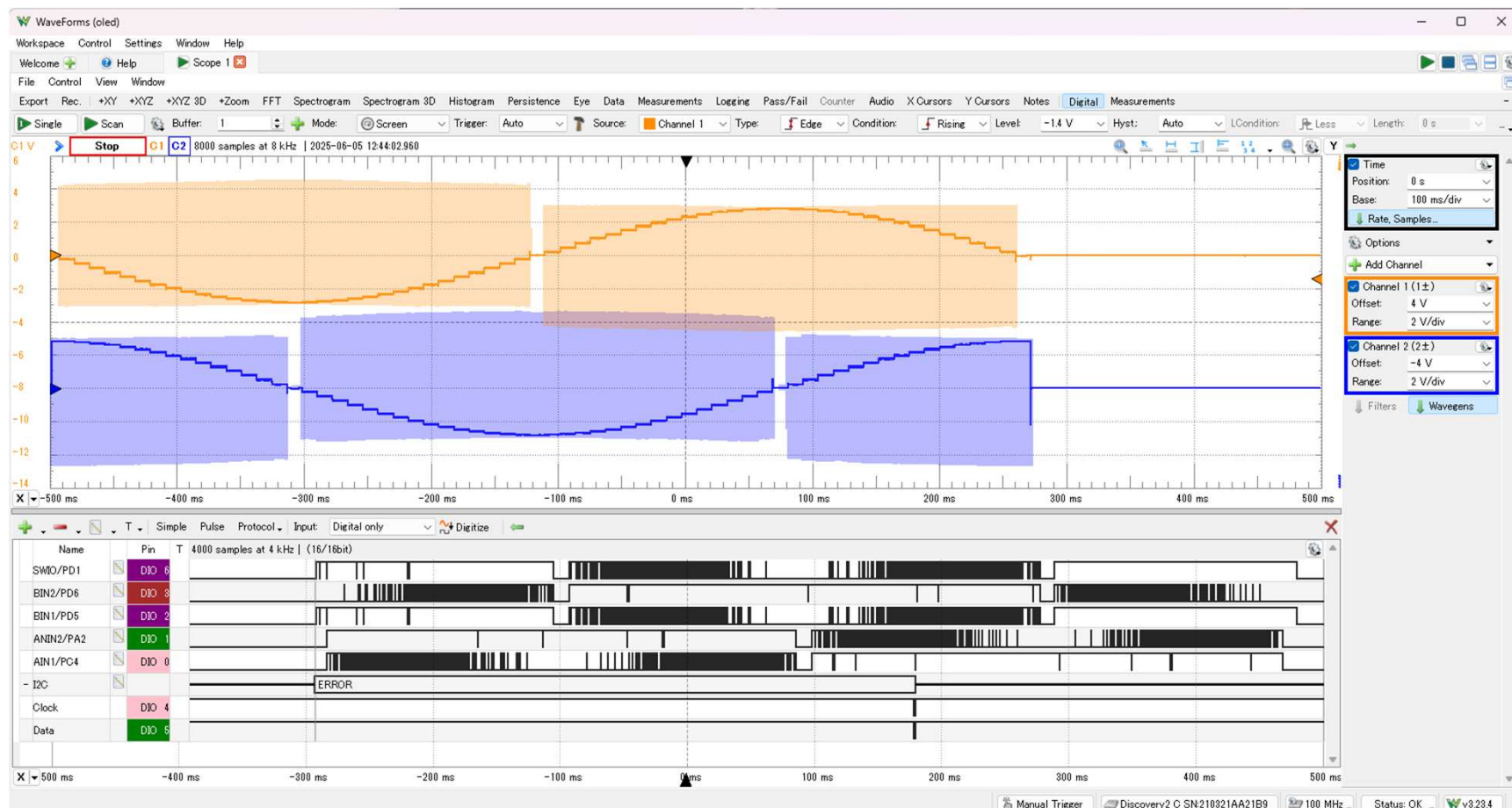
- **TIM1, TIM2**でモータ駆動のPWM出力
 - TIM1 : Pin 3, 7 TIM2 : Pin 1, 8
- **I2C**でOLED駆動
 - SDA : Pin 5 SCL : Pin 6
- デバッグ&コード書き込み
 - SWIO : Pin 8

デバッグ 環境



モーター駆動信号 (PWM)

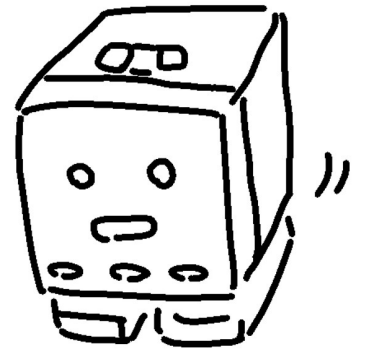
マイクロステップ駆動：1ステップ=360/64deg



Stack-chan chibi (スタックちゃん チビ)



ご清聴ありがとうございました



部品表

#		name	cnt
1	マイコン(RISC-V 8pin)	CH32V003J4M6	1
2	コネクタ(2pin 1.25mm)	molex 53047-0210	1
3	スイッチ	ALPS SSAJ120100	1
4	コンデンサ(0.1uF)	GRM21BR71H104KA01	1
5	モータードライバ	DRV8835	1
6	ステッピングモータ	-	1
7	OLED	0.49'' OLED(64x32 SSD1306)	1
8	リチウムポリマー電池	401515	1



OCB(Optical Clock Beacon)

撮影時刻を高精度に記録できる光学ビーコン



静岡大学木谷研究室 曾根卓朗

BIKEINFORMATICS | KITANI LAB



撮影画像から撮影時刻を1ms 精度で読み取れる時計は作れるか？

2025/08/28



■ アナログストップウォッチ

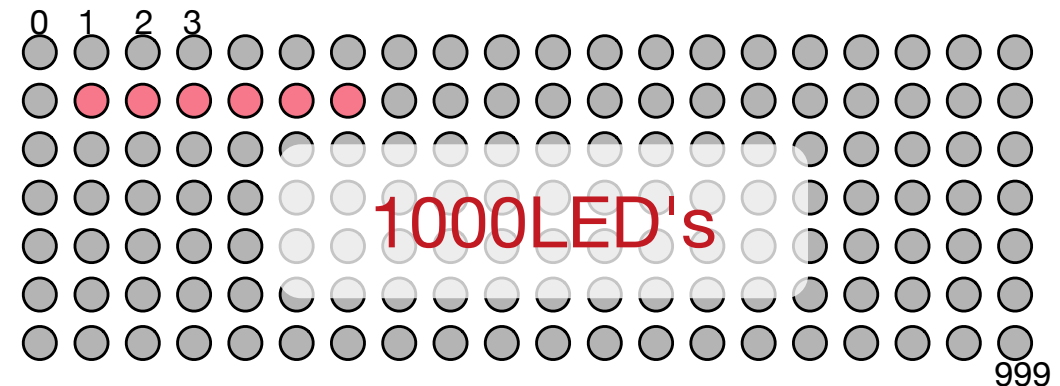
- 高精度同期した機械式時計の製作は高コスト.
- 一般的な使用には適さない.



写真はSEIKO 89ST

■ 大量のLEDでの時刻表示

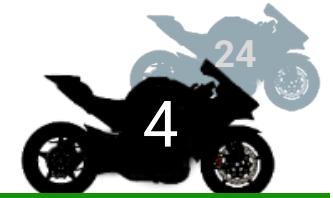
- 撮影時刻検出は容易
- 製作は可能だが大規模化し
- 一般的な使用には適さない



■ デジタル時計

- 製作は容易
- しかし、文字が潰れて読めなくなる問題あり.

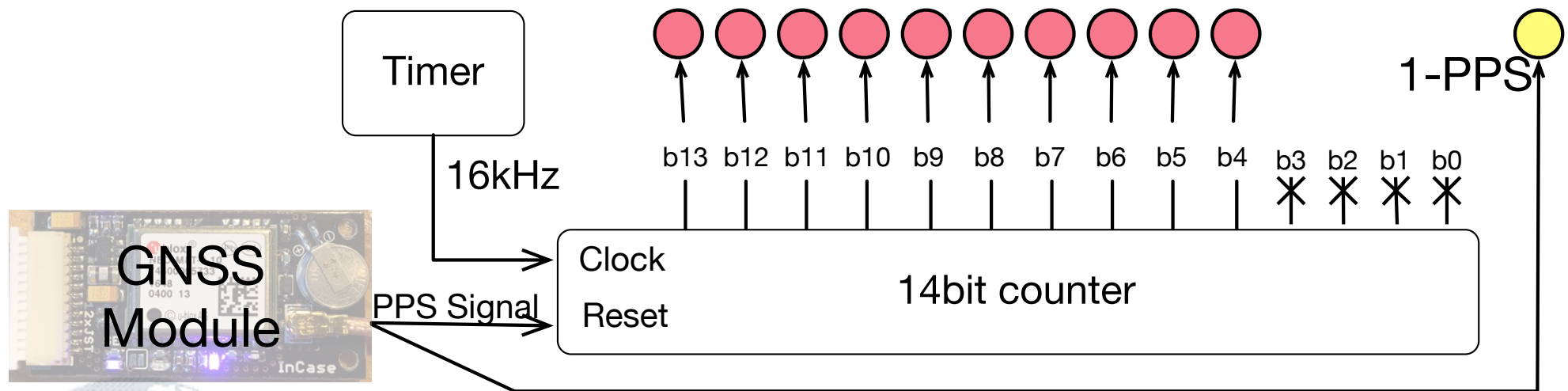




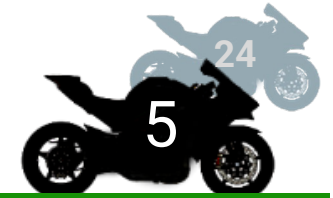
2進数表示デジタル時計での撮影時刻検出

- GNSS時刻同期信号（1-PPS）に同期した2進数表記のデジタル時計は容易に製作できる。

10bit binary representation of current time(0...999)



露光中に点灯状態が変化すると時刻読み取りができなくなる問題がある。



OCB の設計目標

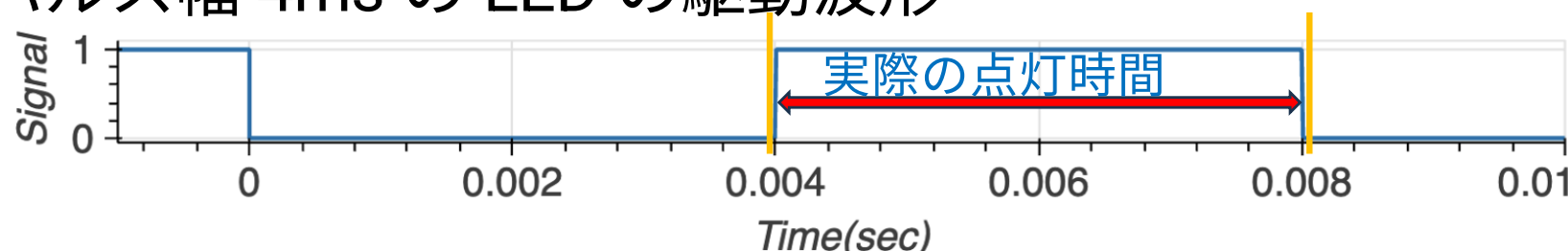
- 撮影画像を解析することで可能な限り高精度な撮影時刻を特定できる光学マーカー (OCB) の設計原理を明らかにする。
一般的な撮影機材を無改造で利用することを前提とする。
- そのために以下の性質を備える必要がある。
 1. **露光条件不変性**：撮影条件の変化の影響を受けにくいこと。
 2. **露光タイミング不変性**：撮影タイミングにより時刻精度が変化しにくいこと。
 3. **露光時間不変性**：露光時間(シャッタースピード) の変化の影響を受けにくいこと。
- 時刻精度性能に対し可能な限り簡易な構成でOCBを実現
 - なるべく少ない光源、なるべく簡素な回路構成



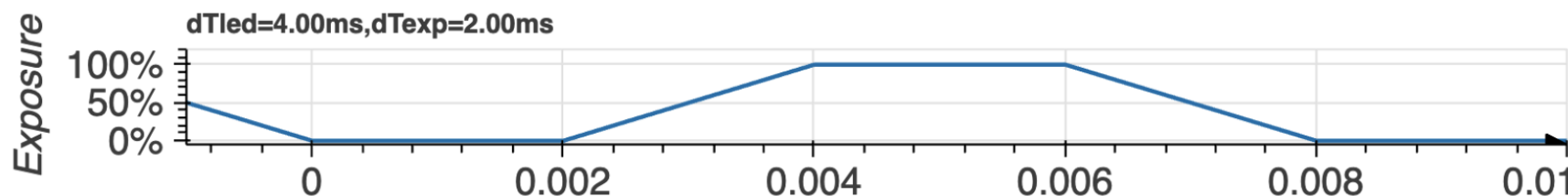
点光源を使った時刻表現

■ 点滅する光源の撮影とピクセル値の関係

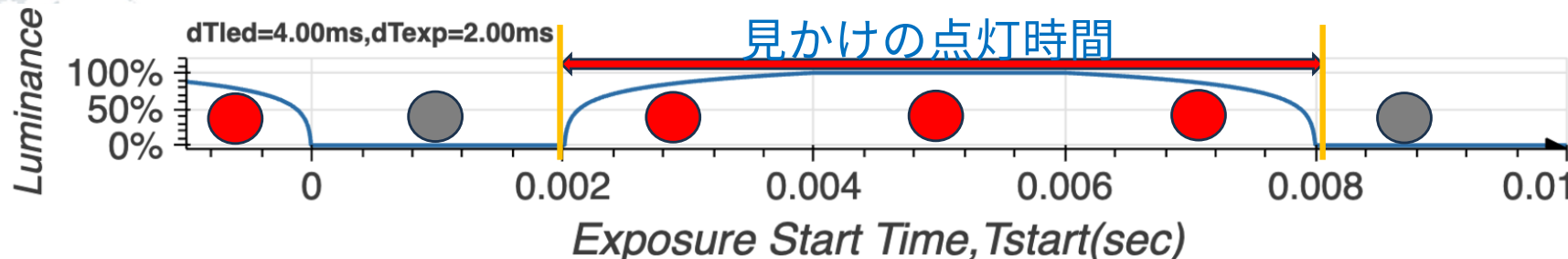
1. パルス幅 4ms の LED の駆動波形



2. 露光時間 2ms で撮影した時の露光量



3. 撮影画像のピクセル輝度（検出されるLED点灯状態）

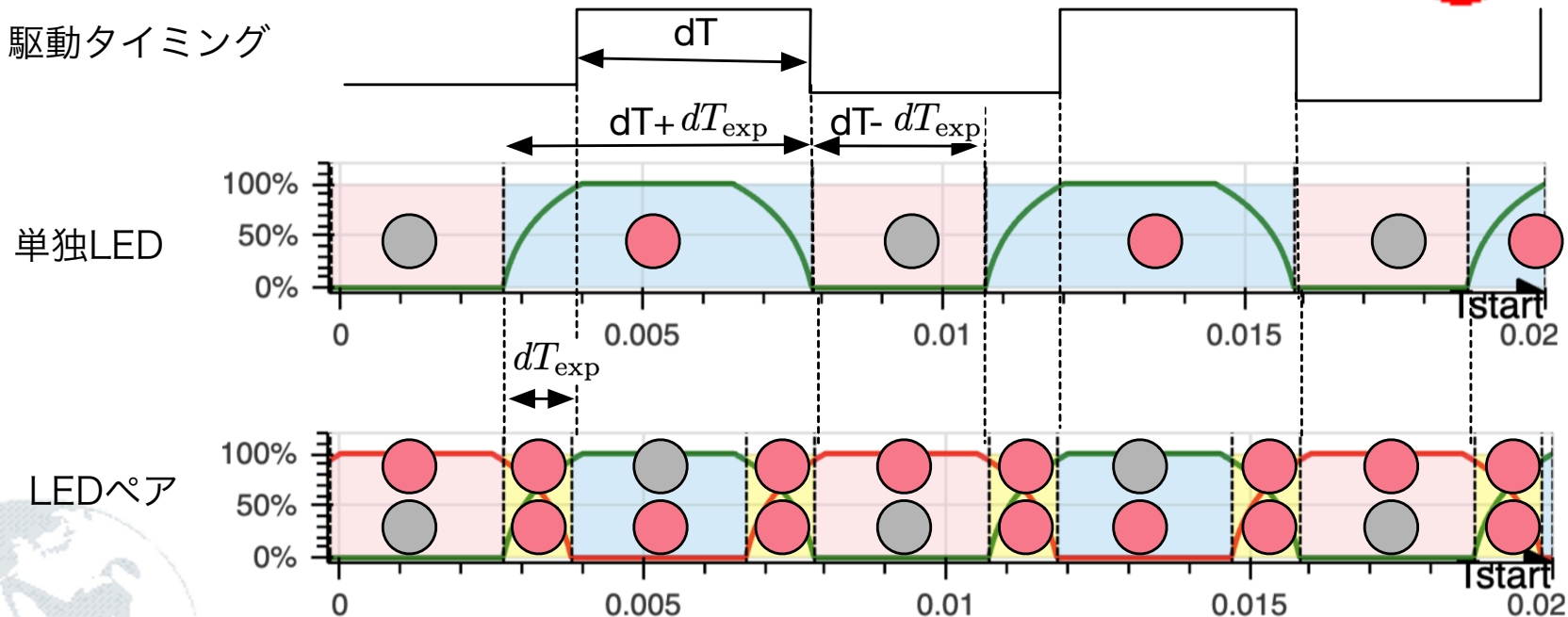


LEDペア



- 点灯状態を反転させた LED のペアで 2 つの状態を表示できる
- 撮影画像は 3 状態になる

駆動タイミング



- 露光中の点灯状態の変化を検出できる
撮影時刻検出を行うために非常に好ましい性質

時刻エンコード方式：2進数表現



- 現在時刻を2進数で表現すれば光源数最小となる
- 2進数の各桁は2状態なので、2個の光源（LEDペア）で表現できる．
- 2進数の桁数をM，必要な光源数 S_n とすると，

$$M = \lceil \log_2(N) \rceil$$

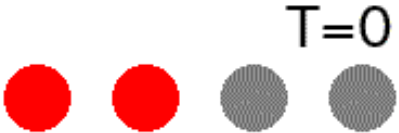
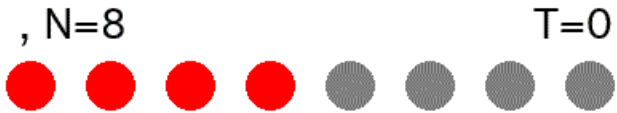
$$S_n = 2 \times \lceil \log_2(N) \rceil$$

N=1000 ならば $M=10$, $S_n = 20$ となる



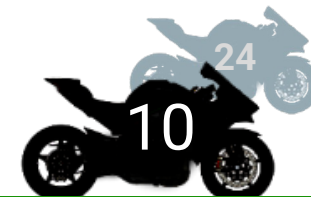
時刻エンコード方式：冗長化

- 現在時刻表記の桁 k を構成する光源の数 $N(k)$ を増やすこと（冗長化）することで、より高精度に現在時刻を表現することができる．冗長度を R で表す．

- $N(k)=2 : R=1$  LEDペア1個
- $N(k)=4 : R=2$  LEDペア2個
- $N(k)=8 : R=4$  LEDペア4個

時刻エンコード方式の 主な設計パラメータ

2025/08/28



1. OCBで表現する定義時間領域は任意に設定できるが、簡単のため1秒で説明する。
2. 定義時間領域の分割数 (**N**) は2以上なら任意であるが、定義域境界での連続性を考慮すると、**2のべき乗**の場合とそうでない場合での分析が必要。

10桁なら 1000 vs 1024

3. **2進数表現**と **Grayコード表現**

数値表現としては等価だがハミング距離が常に1のGrayコードは2進数表現と時刻検出精度に違いが出る可能性がある。

4. 各桁の光源の**冗長性 $R(k)$** ($0 \leq k \leq M-1$)

$R(k)$ が増加すれば時刻検出精度も向上するはず。

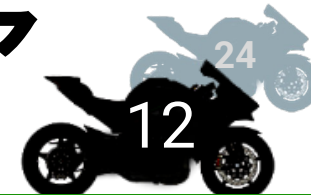


OCB の設計原理解明

- 設計パラメータを網羅的に変化させた探索を行った結果、OCBの構成と撮影画像から読み取れる時刻精度の関係を決定する**設計原理**を明らかにできた。
- 時刻精度は光源の数を増やすことで向上させることができ、**理論上の精度上限は存在しない**。（実装制約のみ）
- これにより利用目的により最適なOCBを設計できるようになった。
- 以下では
 1. 2進数デジタルクロックである **OCB-B1000**
 2. OCBの基本構成の **OCB-G** (GrayCode)
 3. 高精度化が可能となる構成の **OCB-R** (Redundant) , **OCB-V** (Versatile)について解説する。

* T. Sone, M. Mita, and T. Kitani, "A method for designing an optical clock beacon to detect highly accurate image capture time," IEEE Access, 2024.

OCB のプロトタイプハードウェア

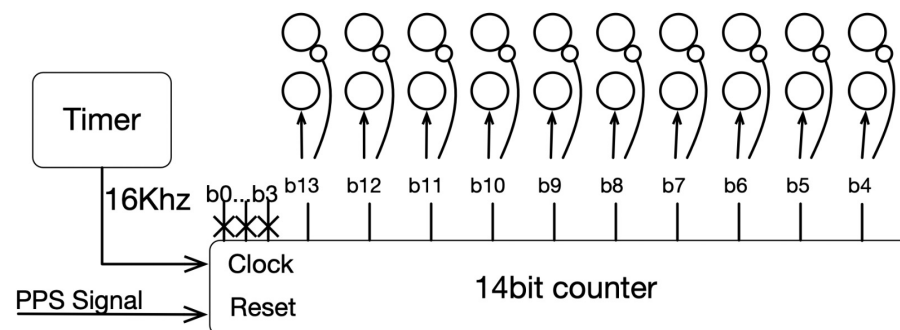


■ CL-Beacon (Complemental-Linear)

- OCB-B1000 を 1-PPS 信号で駆動. 現在時刻をミリ秒単位で 2進数表現.
LED 数 21個
- $dT_{exp} \geq 0.5ms$ で時刻精度は dT_{exp} とほぼ同じとなる.



Green, BlueLEDは画像検出用マーカーLED

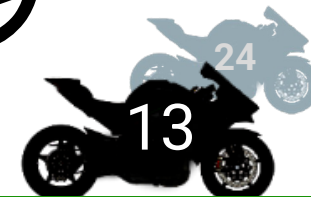


CL-Beacon概観、回路構成

トラ技 2024年2月号p126, 「GPS時刻」をカメラ撮影画像に写し込む装置の制作 で解説しています。

OCB-B1000:: 2進数デジタルクロック(1/2)

2025/08/28

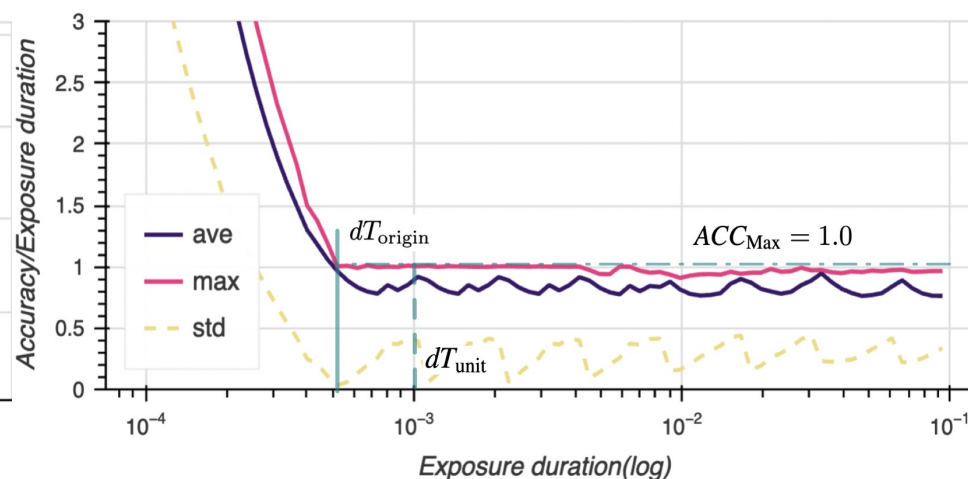
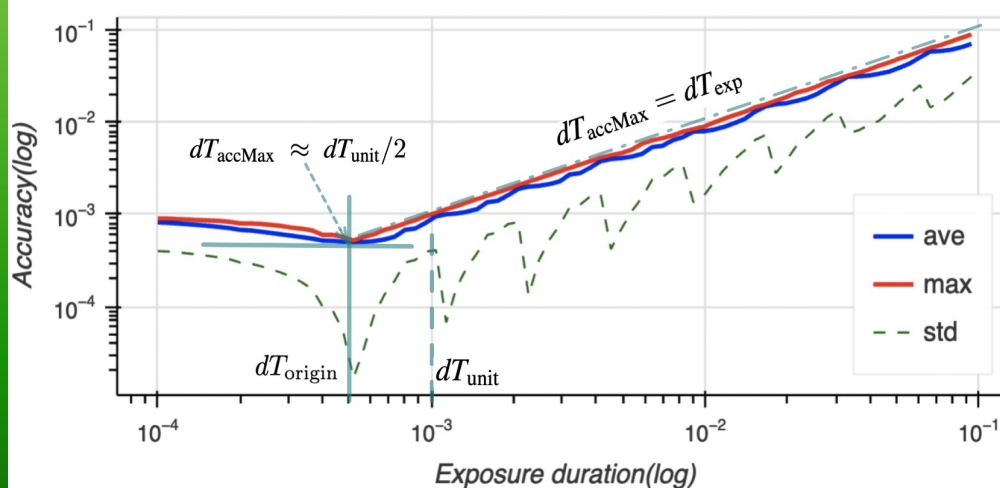


■ $dT_{\text{domain}}=1.0$ 秒 $N=1000$ 限定の2進数表現のOCB

ミリ秒精度の2進数時間表現であることがメリット。

時刻精度はOCB-Gとほぼ同じだが、周期1秒の光源が1個追加で必要。

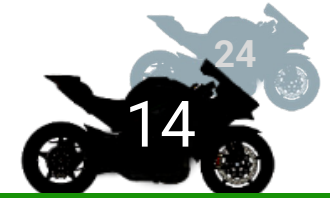
光源数は21個。 Grayコード変換が不要のため回路構成は最小。



Binary
 $N=1000$

● ● ● ● ● ● ● ● ● ● + PPS-LED

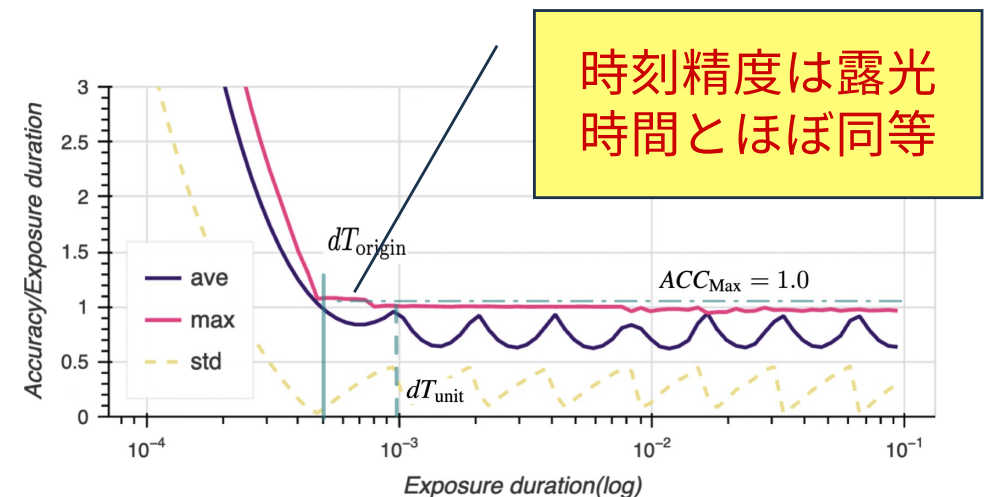
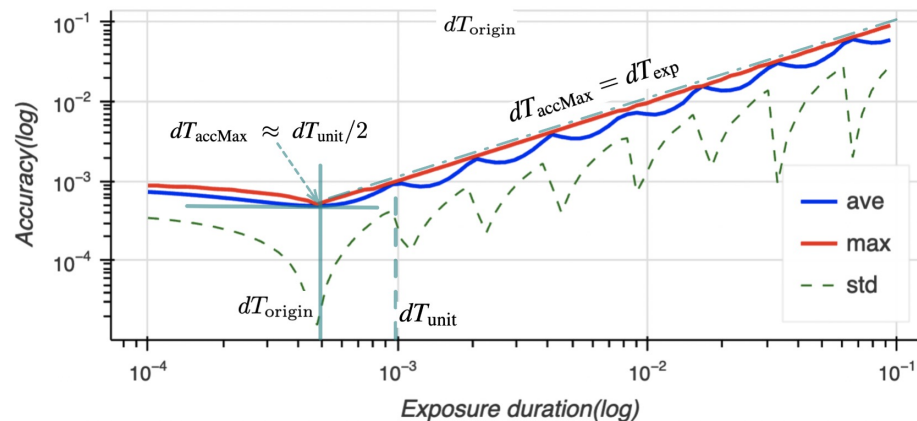
T=0



OCB-G：基本構成OCB

- **Nが2のべき乗**であり現在時刻を **Gray コード** で表現した OCB. 光源数は最小 $S_n = 2 \times \lceil \log_2(N) \rceil$

N=1024 の OCB-G の時刻精度解析結果



- 露光時間が $dT_{unit}/2$ 以上で時刻精度が露光時間と同等

Gray
N=1024

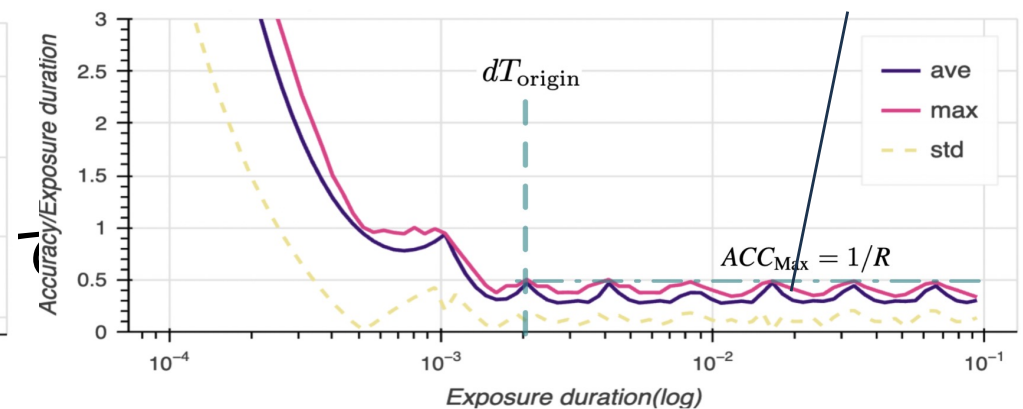
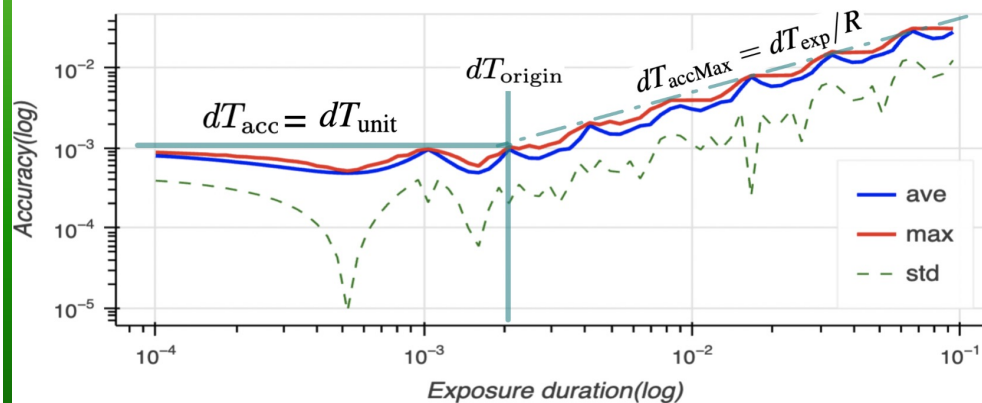
T=0



OCB-R : 冗長構成 OCB

- OCB-Gの各桁を冗長構成とすることで、時刻精度は冗長度に比例して高くなる。

以下は $N=1024$, $R=2$ の OCB-R の精度解析結果。

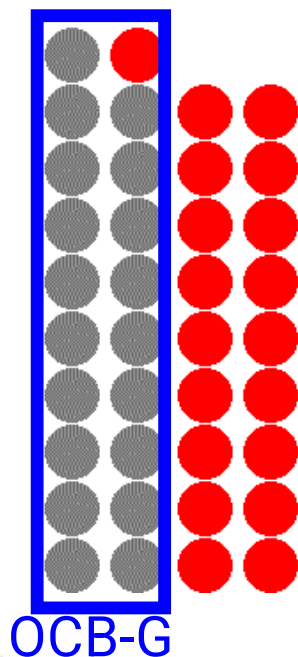


OCB-R のN=1024, 冗長度 (R=2) のサンプル

2025/08/28

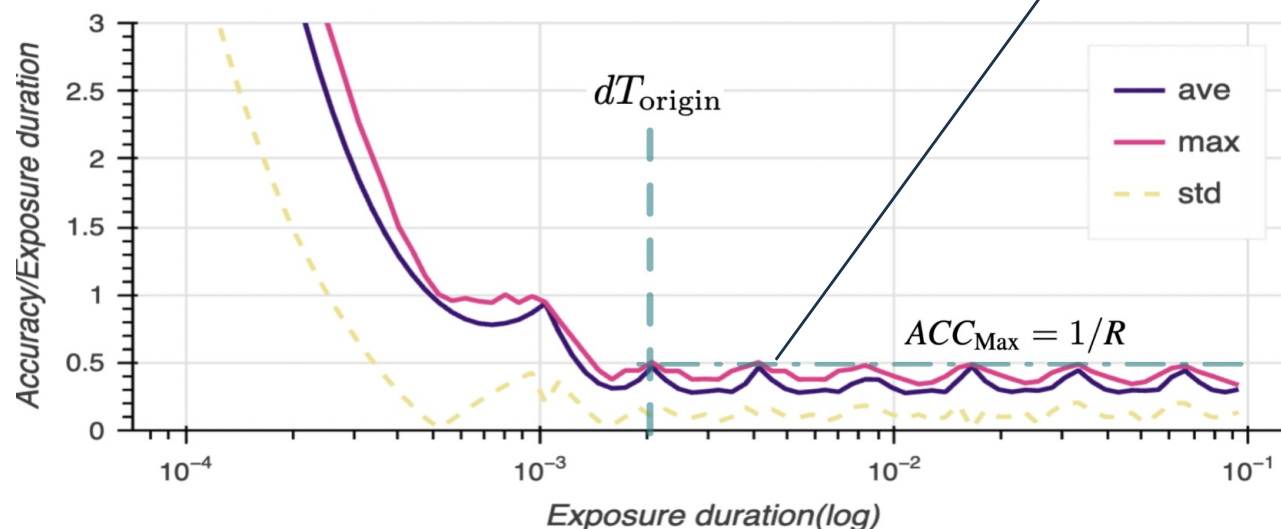


OCB-R, R=2



OCB-G

正規化した時刻精度



注)アニメーションの点灯パターンはイメージ

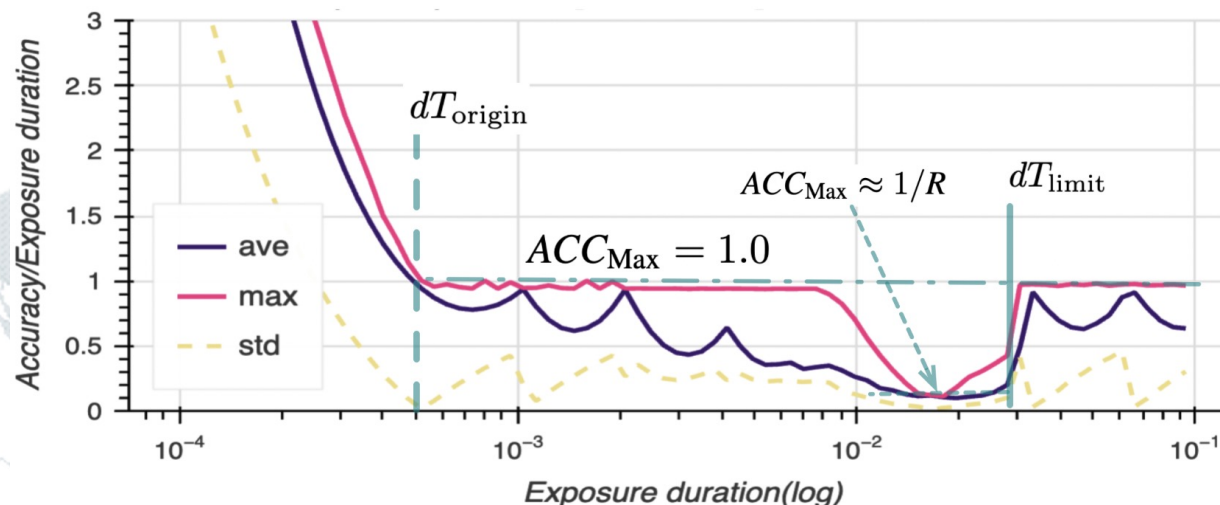


OCB-V：汎用構成 OCB

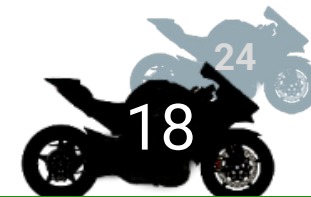
- 一部の桁だけ冗長度を高めることで、時刻精度を高くしたい露光時間を制御できる。

K 桁目の光源を冗長化すると、おおよそ以下の 露光時間の範囲で時刻精度を向上する効果がある。

$$0.7 \times 2^k \times dT_{\text{unit}} \lesssim dT_{\text{exp}} \lesssim 1.7 \times 2^{k+1} \times dT_{\text{unit}}$$



R(4)=8 として
 $dT_{\text{exp}}=16\text{ms}$ 前後のみを
 高精度化した例



OCB-V , N=1024, bit4の冗長度4としたサンプル

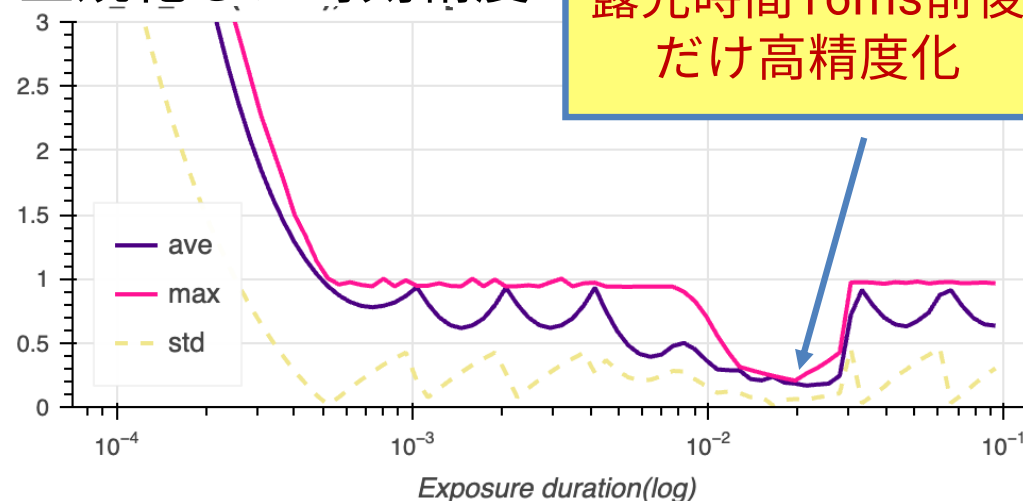
OCB-V, $R(4)=4$



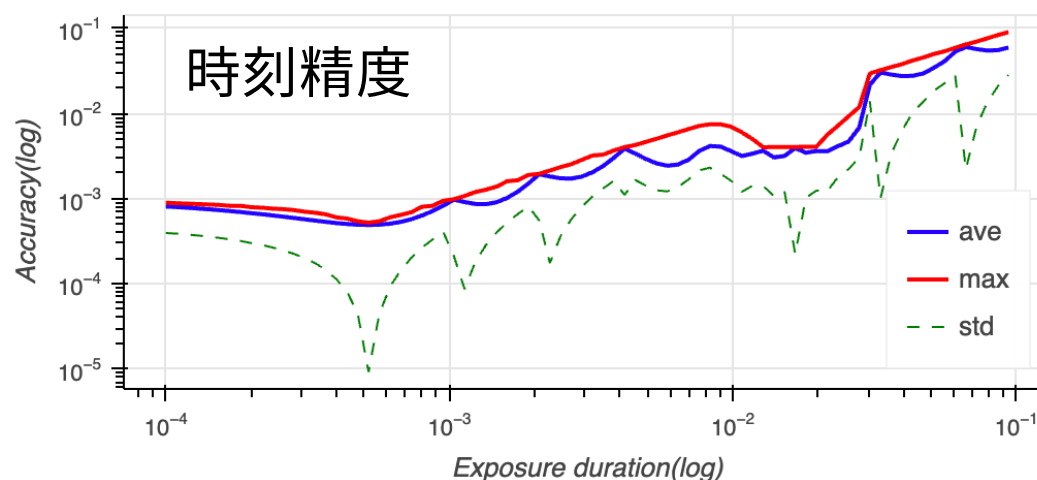
パルス幅約16msの桁を
4倍に冗長度化

注)アニメーションの点灯
パターンはイメージ

正規化した時刻精度



60fpsのビデオ撮影で約4msの
時刻精度が得られる





撮影時刻検出手順

1. OCB 撮影画像から個々の LED の点灯状態(L_{pat})を検出.
2. セグメントテーブルを L_{pat} で検索.

3. セグメ
4. 露光開

- 全点灯パターンを網羅した辞書を作成
- 撮影画像の点灯パターンで辞書検索して撮影時刻と時刻精度を検出

取得.
きる.

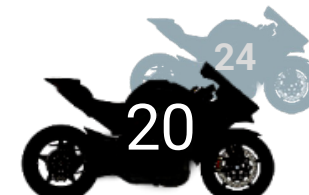


L_{pat}

dT_{exp}

Segment	Tstart	dT	Tstart*	dTacc	L_{pat}
0	Ts(0)	dT(0)	Ts*(0)	dTacc(0)	Lpat(0)
1	Ts(1)	dT(1)	Ts*(1)	dTacc(1)	Lpat(1)
...	...	...	...	...	...
j-2	Ts(j-2)	dT(j-2)	Ts*(j-2)	dTacc(j-2)	Lpat(j-2)
j-1	Ts(j-1)	dT(j-1)	Ts*(j-1)	dTacc(j-1)	Lpat(j-1)

Tstart*
dTacc



画像解析システム (2)

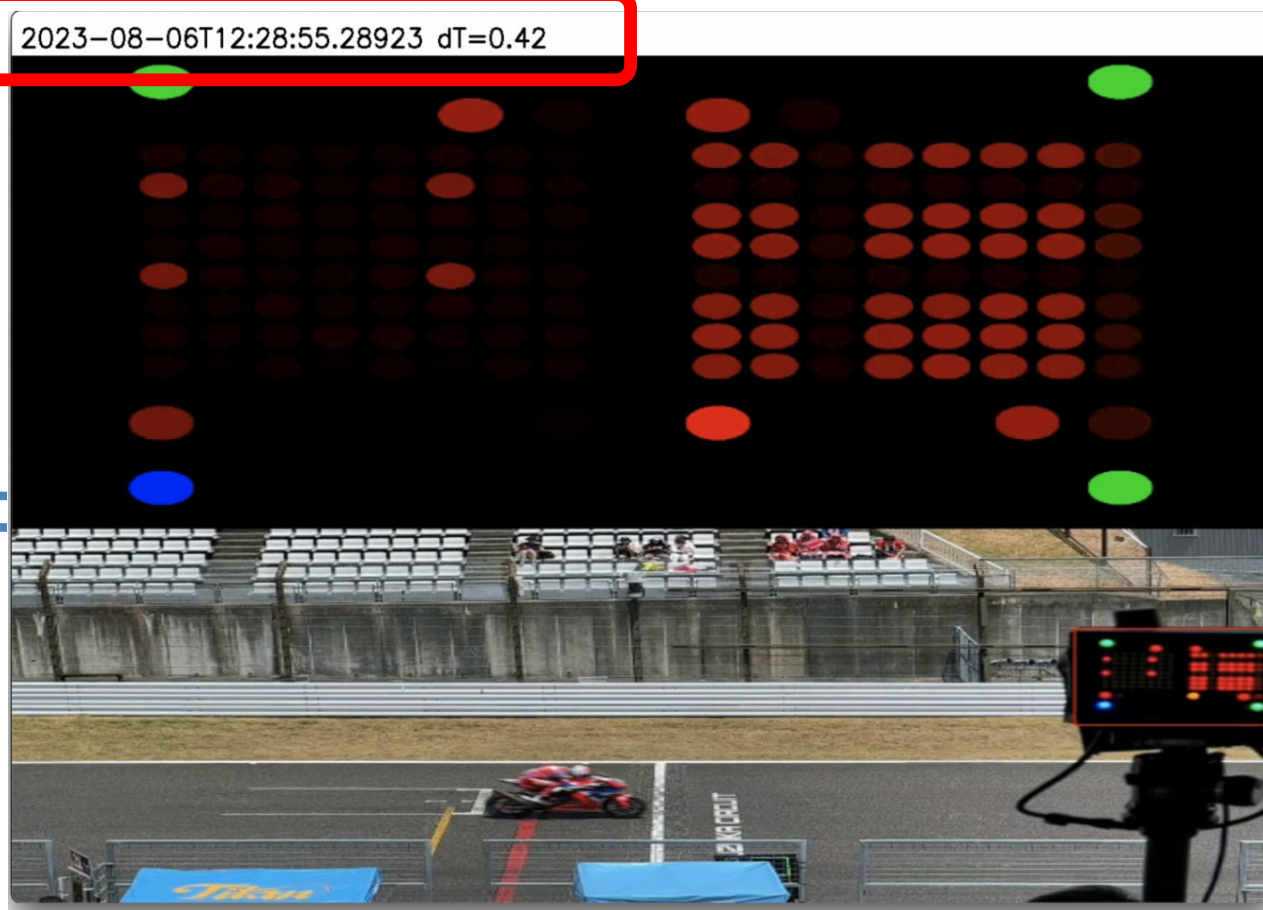
■ 解析結果表示画面サンプル

検出した撮影時刻と
時刻精度

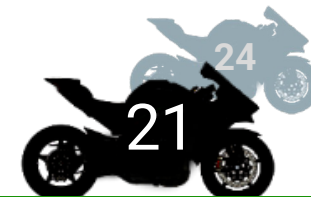
2023-08-06T12:28:55.28923 dT=0.42

マーカ検出
射影変換
点灯状態解析結果

元画像



ローリングシャッター速度計測



- 複数の OCB を同時に撮影することで、画像の位置による撮影時刻の差が検出できる.



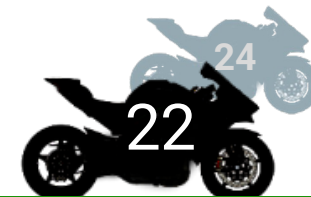
左CL-Beacon, 右CM-Beacon,
水平配置



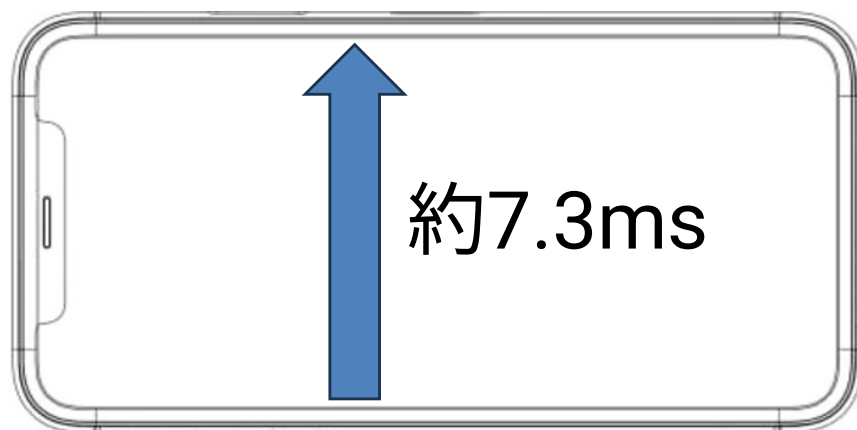
左上CL-Beacon, 右下CM-Beacon,
斜め配置

評価:ローリングシャッター速度計測 (3)

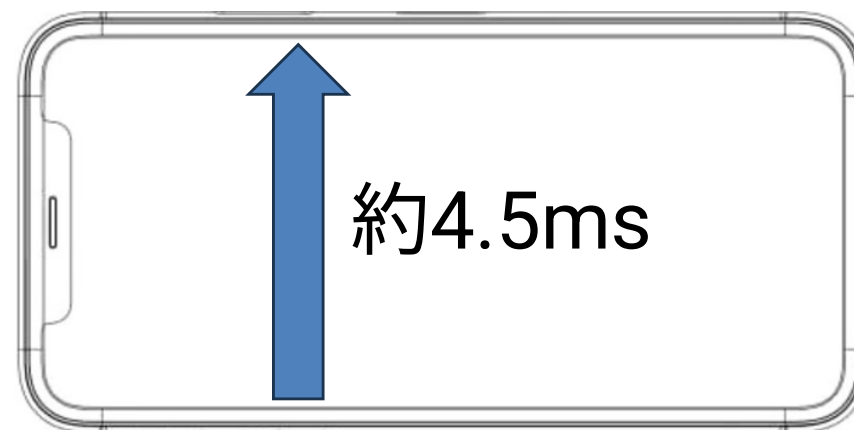
2025/08/28



- iPhone13 ProMaxのローリングシャッター速度推定値
画面上のGCB座標を元に時間差を補正.



30fps撮影時



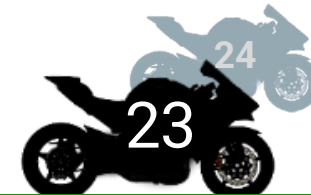
240, 120fps撮影時



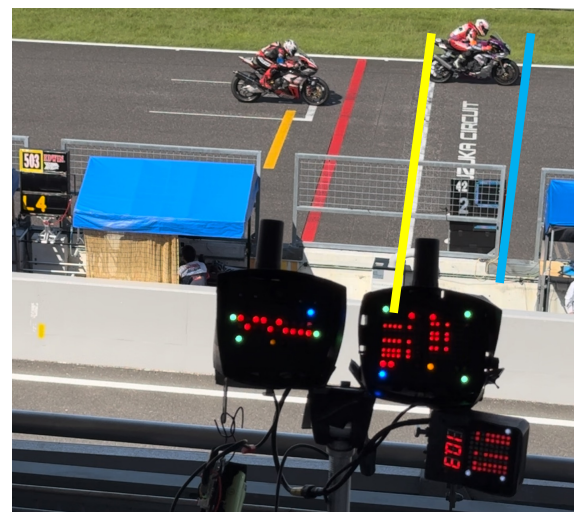
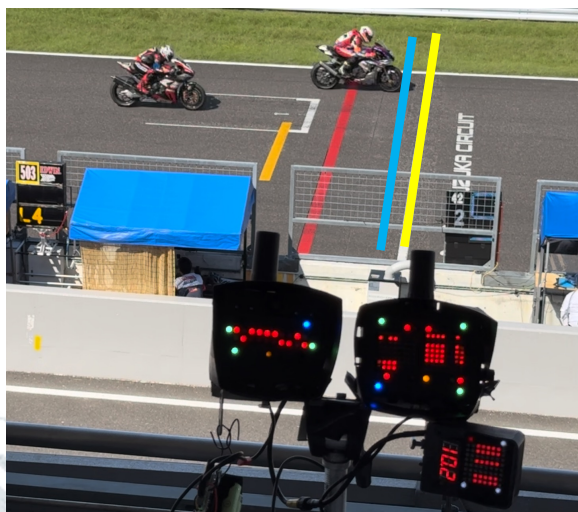
4.5 ms は $1/222$ 秒なので 240 fps撮影には遅すぎる.
頻繁にフレーム欠落が起きる原因の可能性がある.

ローリングシャッターを考慮した時刻誤差補正

2025/08/28



- ローリングシャッタースピードが既知ならば、画像上の OCB の位置と計測対象の位置とで時刻御作法性が可能となる。

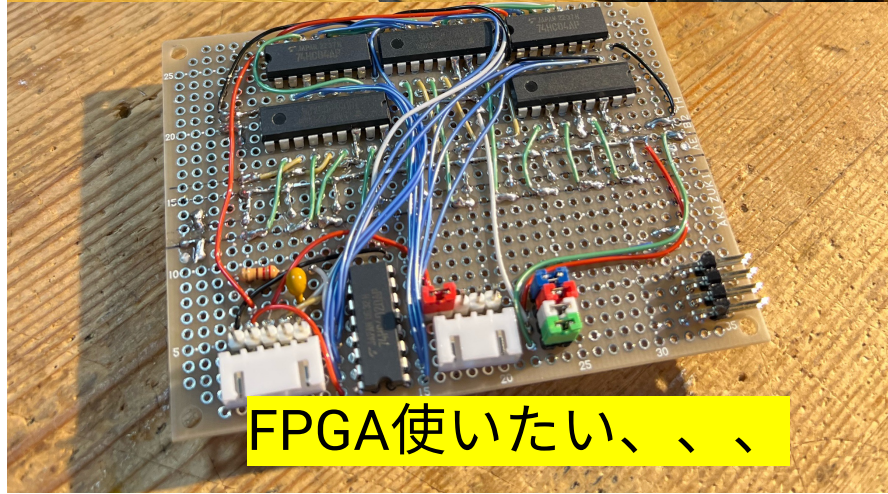
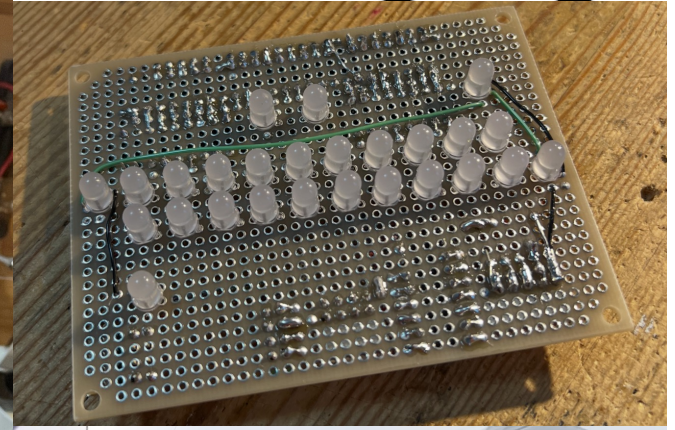


ローリング
シャッター補正

ゴールライン通過時刻計測の例



2025/08/28



FPGA使いたい、、、



おしまい

Modern Xen: VirtIOとZephyrによる イマドキのXenの作り方

常田 裕士

自己紹介

常田 裕士

LinuxとかAndroidの困りごとの相談を受ける仕事をしています。

最近はZephyrRTOSというオープンソースの開発を色々やってます。が、金にはなってません。
(RaspberryPi Pico/Renesas RA maintainer)

去年は講師として呼びいただきました。
ありがとうございます。

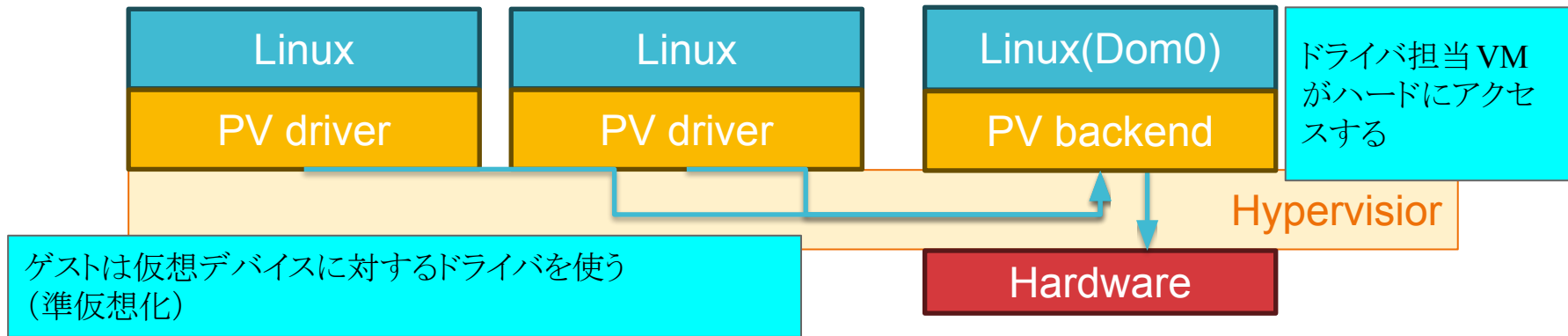


こんな本出しました。是非よろしく。

講演のない
快適な
SWEST

Xen、覚えていますか？

- 2003年に登場したオープンソースのハイパーバイザ
- Para-Virtualization方式で、高速なパフォーマンスを実現、初期のEC2などにも採用され普及した。
- 2010年代でKVMやコンテナの登場でクラウドでは一線から引き下がる。
- 現在では組み込み、機能安全の方向にシフトして開発が続けられている。



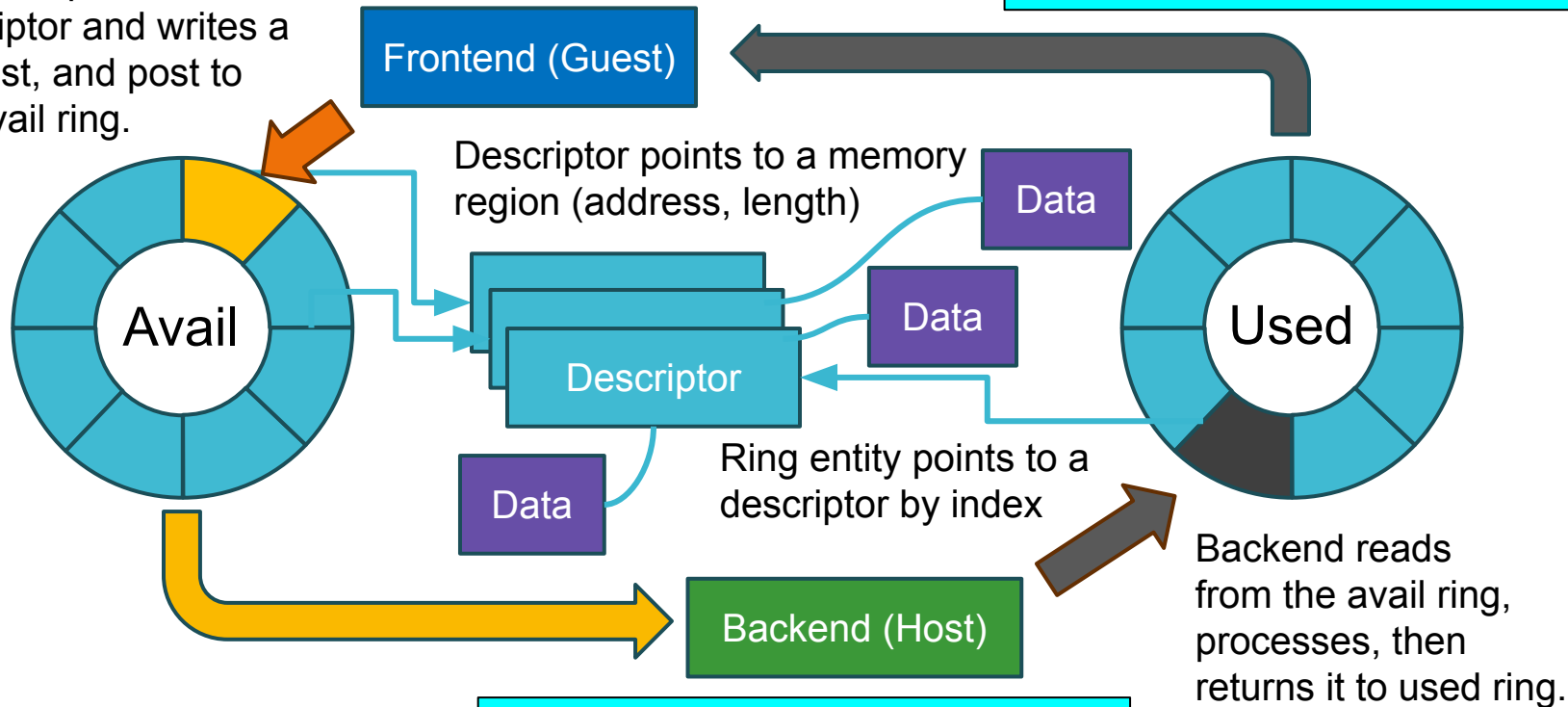
Xenの最近の課題

- XenではLinux仮想マシンを使ってハードを動かしているが、Linuxがリソース食い。
(Hypervisorの仕事に集中するためドライバを持っていない)
- 近年の標準的な準仮想化インターフェースのVirtIOへの対応が進んでいない。

Two rings: arteries and veins

標準化されているVirtIOの仮想デバイスがトレンド。共有メモリを使ったデータ交換方式と仮想デバイスを規定。ドライバは色々実装済み。

Frontend picks free descriptor and writes a request, and post to the avail ring.

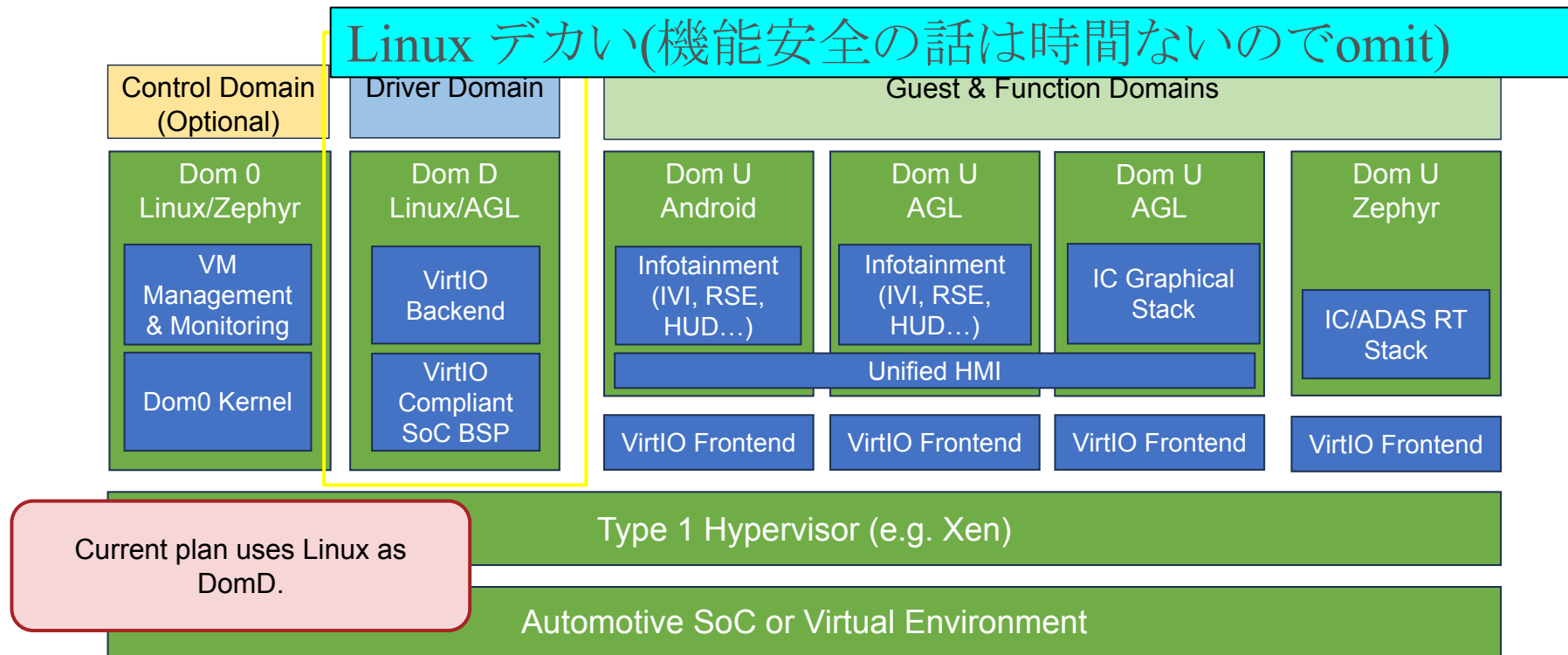


「2本のリング」はLinuxのio_uring二も見られる処理の非同期化の手法

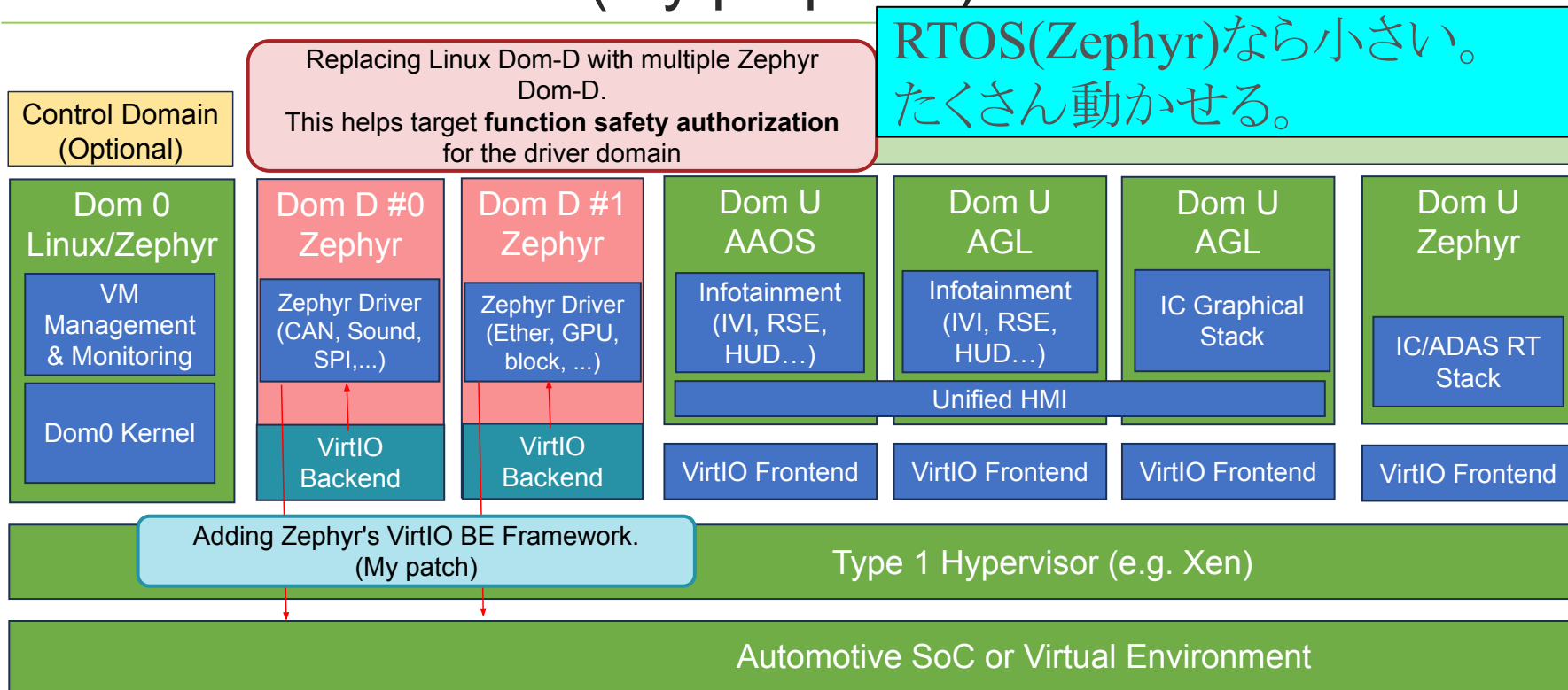
Zephyrを導入して近代化

- 方針:ドライバ担当の仮想マシンをZephyrで動かす
- Zephyrはわりとドライバが多い。
 - Hypervisorはどこもこれで苦労している。
- LinuxをRTOSサイズまでコンパクト化。
 - メモリ消費↓フットプリント↓
- 折角なのでこの際にXen独自の準仮想化I/FのPVを捨ててVirtIOに移行。
 - 仕組みはあったが誰も使ってなかった！
- VMでドライバ実装すると性能が気になるが、HWが何とかしてくれると期待w

SoDeV Architecture



SoDeV Architecture (My proposal)





キーワードで検索



| 新着

| トラブル

| 調査／データ

| IT ▾

| 電機 ▾

| 自動車 ▾

| 建築 ▾

日経クロステック トップ > クルマ技術／動向 > SDVゲームチェンジャー > ホンダが車載Linux「AGL」採用へ、28年めど 脱QNX模索

SDVゲームチェンジャー

+ 特集をフォロー

第1回

ホンダが車載Linux「AGL」採用へ、28年めど 脱QNX模索

清水 直茂 日経クロステック／日経Automotive

2025.05.19

有料会員限定

詳しくは知りませんが、こういう話題と関係があるかもしれないし、ないかもしれない..

RTOSは
マイコンで
なくても使える

宣伝: LF Japan Community Days 大阪



オープンソース チョットデキル系の人たちが
「大阪行って面白い話してこい」ということで色々お話するイベント。
何となく私も呼ばれそうなので来れる方は是非。参加無料です。
前日の10/20に大阪でZephyr meetupも開催予定。

宣伝: OpenSourceSummit Japan と 非公式呑み会



12月に行われるOpen Source Summit Japanに合わせて非公式呑み会を企画中。
去年は海外からも参加者があり盛況でした。OSSJ行かない人もぜひ。詳細決まったらSlackにて！



AtomVMをやってみた (ESP32-S3でBlinky)

mnishiguchi / SWEST27 LT

自己紹介

- 株式会社ハウインターナショナル所属
- ソフトウェアエンジニア(主にブロックチェーン関連Web)
- NervesJP, piyopiyo.ex, autoracex を始めとする技術コミュニティに参加

なぜ AtomVMをやってみたか？

- Elixirを組み込みデバイスで動かしてみたい！
- @kurokouji さんとの出会いがきっかけ
- XIAO ESP32-S3 をお借りした(やるしかない)
- 最近カンファレンスでも話題
- AIと元氣があれば何でもできる！

使ったもの

- 対象デバイス: Seeed Studio XIAO ESP32-S3(LED: GPIO 21)
- 開発環境:
 - LMDE6
 - Elixir 1.17 / OTP 27
 - AtomVM v0.6.6
 - ESP-IDF 5.5
- 生成AI: ChatGPT o4-mini-high



```
Setting pin 2 low  
Setting pin 2 high  
Setting pin 2 low  
Setting pin 2 high  
Setting pin 2 low
```

AtomVM 開発環境の構築

- 第一関門: ビルド済みイメージ → 動かず断念
- 第二関門: ソースからビルド → 成功！
- 成功パターンを掴むと一気に進む

AtomVM のビルド手順

- 1. ESP-IDF のインストール
- 2. AtomVM のビルド
- 3. ESP32-S3 向けファーム作成
- 4. 書き込み(idf.py flash)

Blinkyアプリの作成と実行

- LED(GPIO21)を点滅させる
- GPIO2 → GPIO21に修正
- `mix atomvm.packbeam` で .avm ファイル生成
- `mix atomvm.esp32.flash` で書き込み(USB経由)
- → LEDがチカチカ ✨

monitor用スクリプト紹介

- idf.py monitor は便利...だがクセが強い印象
- 「環境ロード」+「パス移動」が毎回必要
- 覚えてられない！ → シェルスクリプト化
- <https://gist.github.com/mnishiguchi/e0810c4be07529506dd4fb33805cb2f9>

AIをどう活用したか？

- Elixir未実装関数 → ErlangやAtomVM APIで代替
- C/C++ライブラリをAIに食わせて代替案を生成
- AIだけでは不十分 → 組込み知識を持つ仲間の知恵が決め手に
- → AI × コミュニティの力で突破口を作れた

まとめ

- AtomVMでElixirを組み込みに載せられることを実感
- 一度パターンを掴めば怖くない
- AIで効率UPできるが、最終的には人の知見が大事

ご清聴ありがとうございました！

マイコンとC言語しか使えない原始人にも出来た Xiao RP2040のステートマシンで シリアル用CS信号生成

ADの同期シリアル出力をマイコン内蔵SCIで正しく受信する方法

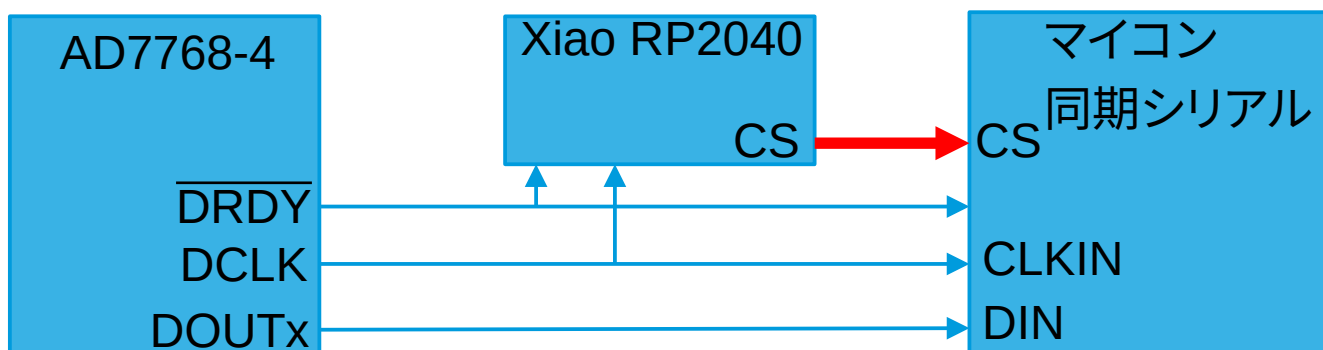
(株)NTシステムデザイン 高村直也

どんなヤツか

- 中学の部活でPC8001 走れスカイライン
- 中学時代にPC8801mkII(model20)買ってもらう
- 秋月でPC8801拡張スロット基板を買ってFM音源ボードを自作 (YM2203?)
- 愛読誌 工学社I/O パソコン制御入門(KeenyBaggs)
- バイブル Z80マシン語ハンドブック 武蔵野マイコンクラブ?
- 高校時代 Z80マシン語でプラネタリウム製作 ステッピングモーター制御
- 地震、気象、海洋、火山、南極などのフィールドの研究者がお客さん
- 海底、火山、南極などで一人黙々と単調なデータ収集をし続ける器械を作るのが喜び

なんの話か

- 24bitデルタシグマADの出力する同期シリアルデータをマイコンで受信する
- マイコンのシリアルインターフェース用にCS信号を生成する必要がある
- 基板を作ったが設計をドジリCS信号の生成回路を追加する必要がある
- その信号をXiao RP2040のステートマシンで生成した



Xiao RP2040とは

- CPU RP2040 Raspberry Pi Picoでも使われているやつ
- FLASH 2MB
- RAM 264kB
- クロック 125MHz
- ステートマシンを4つ内蔵
- 小さい 20mm x 17.5mm
- 秋月で税込み980円

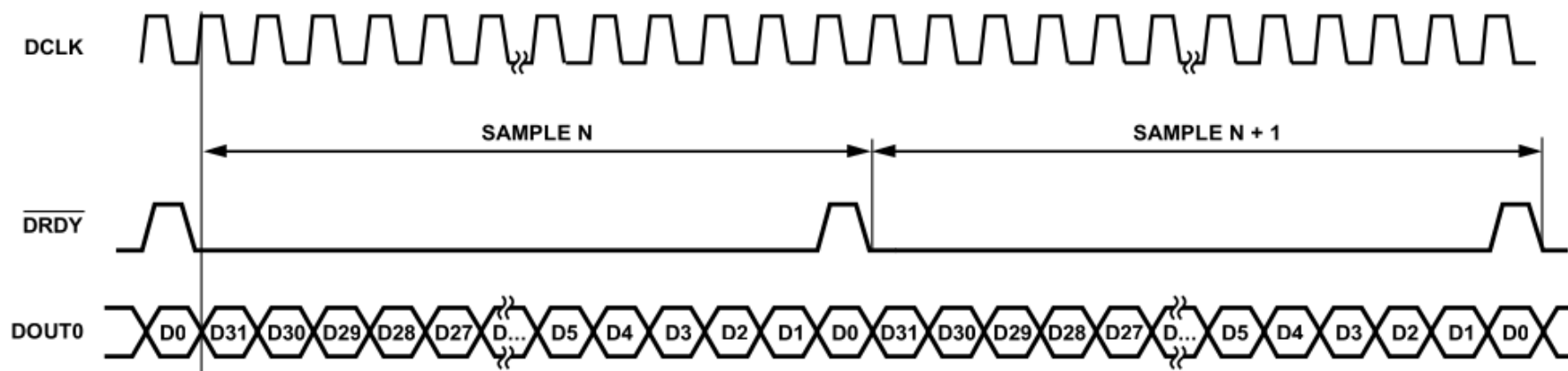
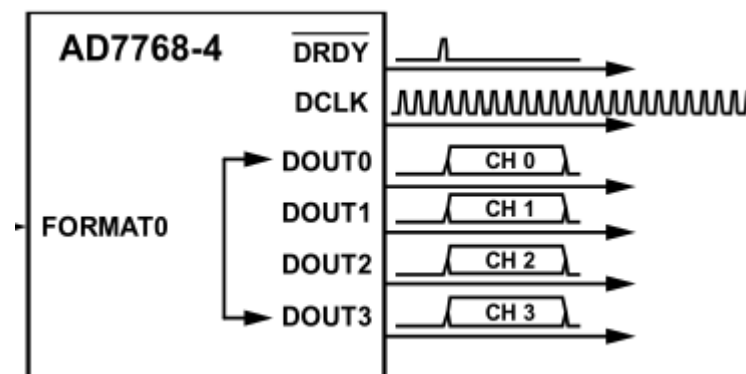


RP2040ステートマシンとは

- CPUとは独立に動作
- GPIOを入出力に出来る
- 機能: 条件分岐・シフトレジスタ・割り込み発生・スクラッチレジスタ
- 昔ならばカウンタやロジックICで組んでいた回路を簡単に実現可能
- 4個のステートマシンはそれぞれ独立して動作
- 125MHz(8ns)の1クロック1動作で高速に動作
- アセンブラまたはMicro pythonでプログラム可能

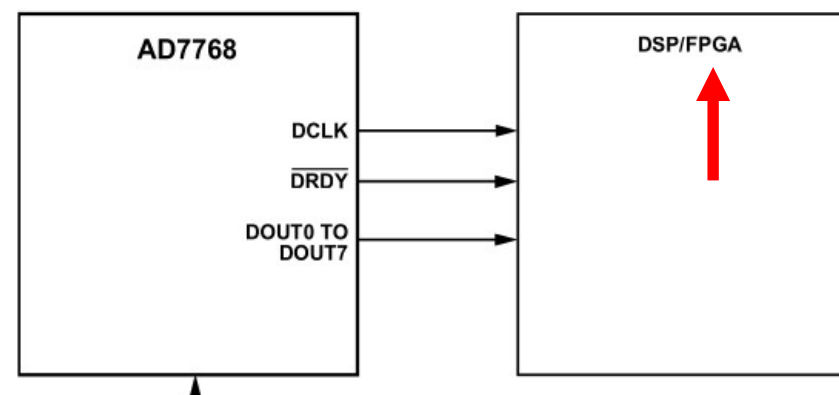
AD7768-4 インターフェース タイミング図

- DRDYの立ち上がりエッジ
- DCLK 3.84MHz
- DOUT 32bit/cycle
- D31-24: 8bit Status D23-0: 24bit AD変換値



CS信号生成が間に合わない

- ADデータを受信するマイコン(RX72N 240MHz)の同期シリアルにCS信号を入れる必要がある
- 当初は、DRDY信号の立ち上がりエッジでマイコンに割り込みをかけて、自分のGPIOでCS信号を生成しようとした ←アホ
- DCLK=3.84MHz 260nsec
- まるで間に合わない!
- ADデータシートの図でも相手はFPGAとか書いてるし
- 本来はマイコン直結するものではないのだろう

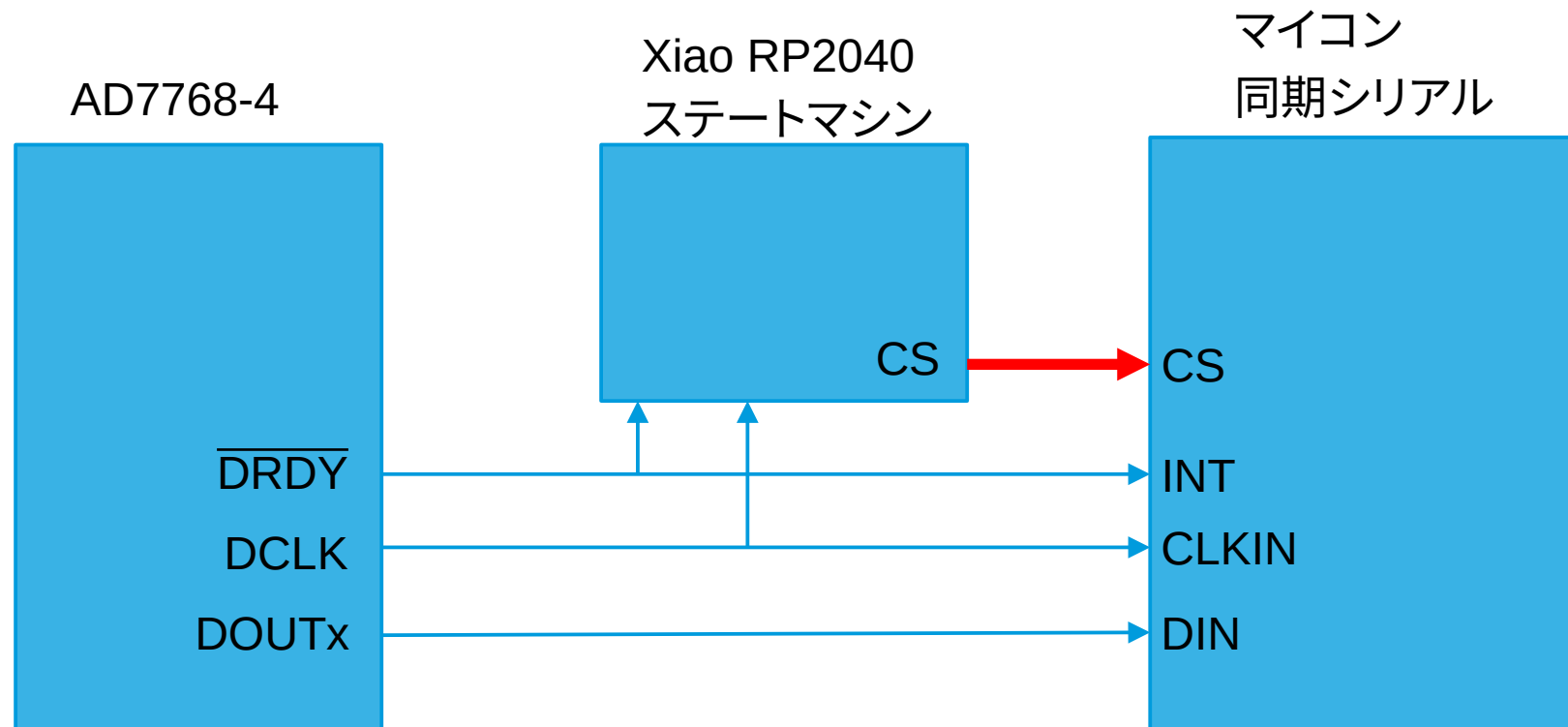


RP2040のステートマシンなら行けるんじゃない？

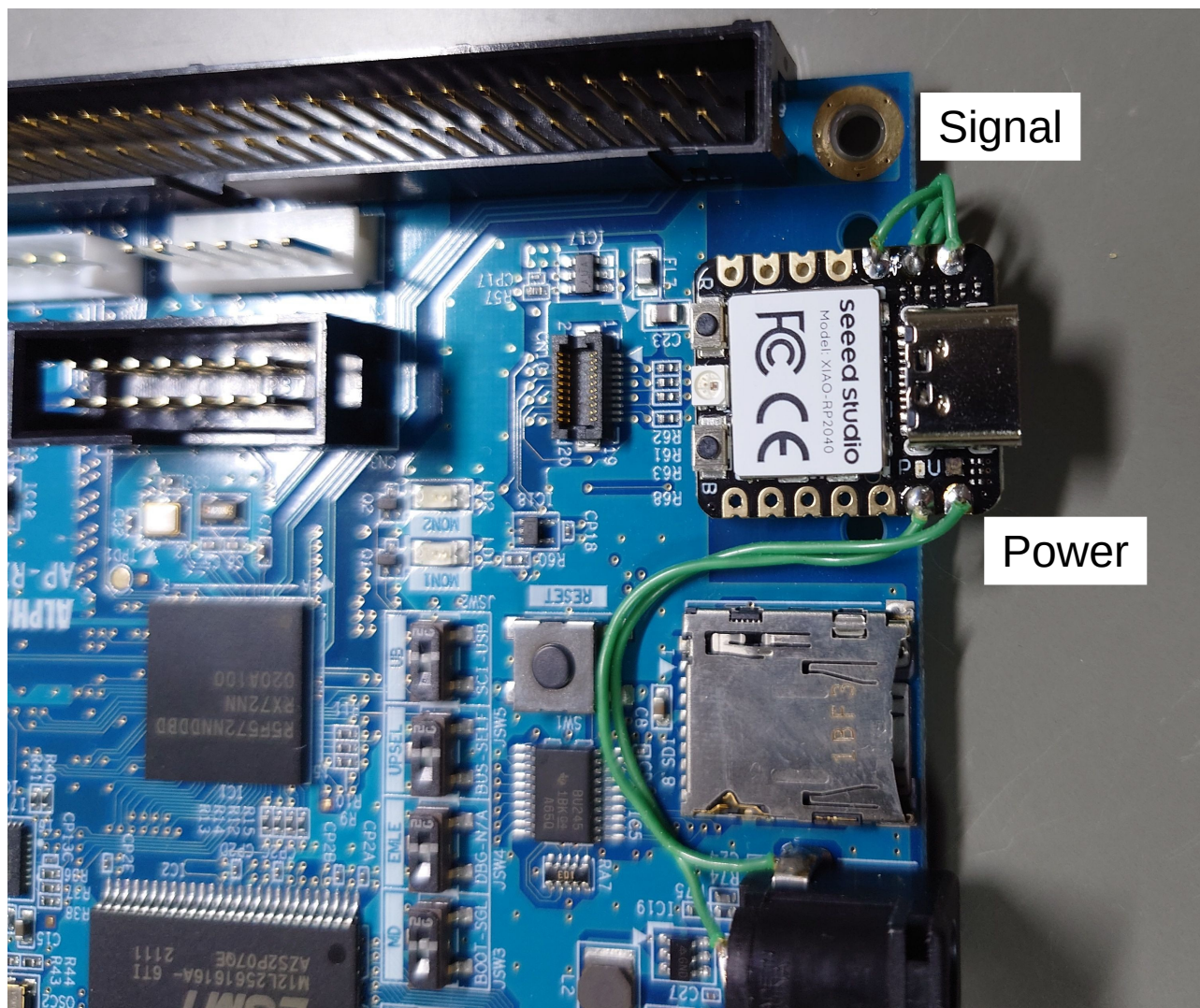
- RP2040動作クロック125MHz 8nsec
- 1クロックで1動作するし
- AD DCLK=3.84MHzに対して125MHzだから余裕あるよね
- 回路組むのと違って、ソフトなんでトライ&エラー出来る
- ロジックで組んだらカウンタとラッチで大変なことになるだろう

配線ブロック図

- DRDYとDCLKからCS生成する



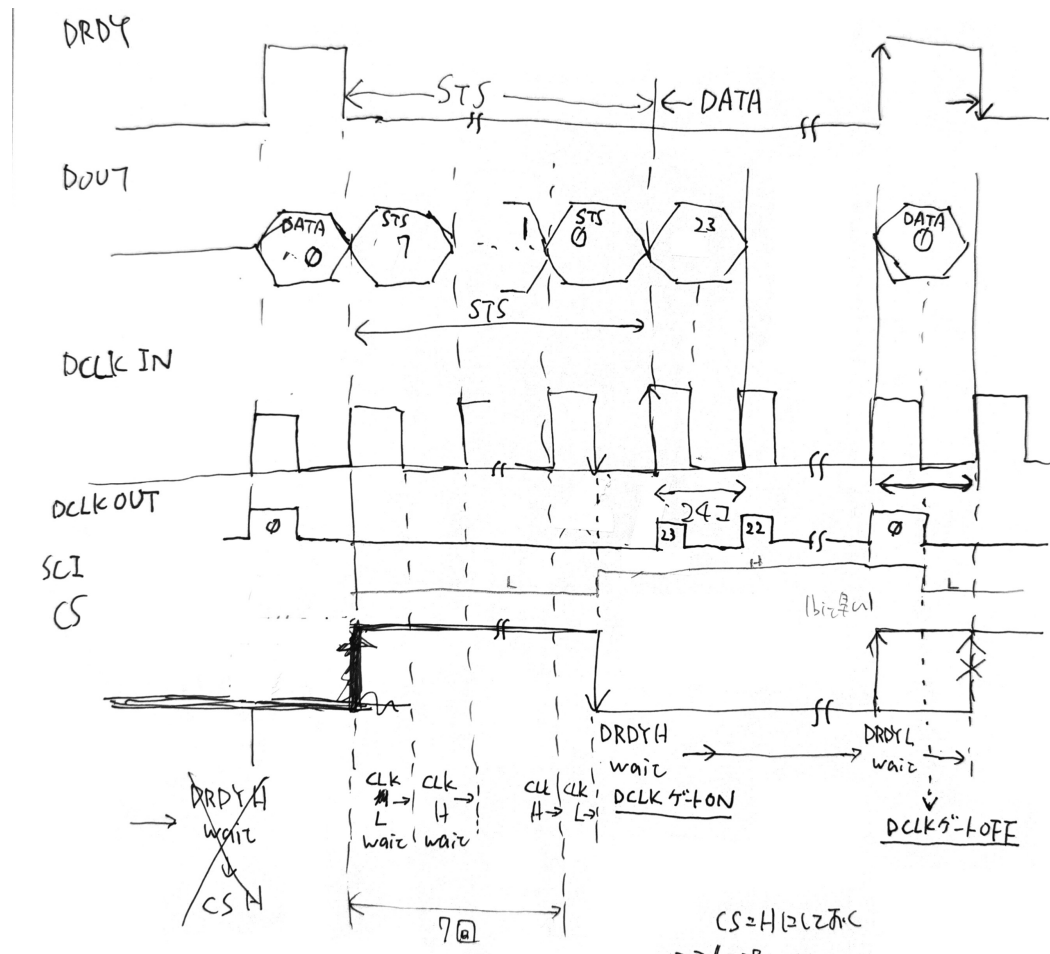
Xiao2040 配線写真



開発環境

- 開発PC: Ubuntu 24.04
- Thonny: Micro pythonのIDE
- Xiao2040: ThonnyでMicro Python Interpreterを書き込み
- PCとXiaoをUSB-Cで接続

手書きタイミング図



原始人なんで
手書きで考えます

ステートマシンソース1

- 主要部分を抜粋

```
from machine import Pin, Timer
import rp2
```

```
# 信号線を定義
```

```
DRDY = 26    # D0 GPIO26 DRDY IN
DCLK = 27    # D1 GPIO27 DCLK IN
CS = 28      # D2 GPIO28 CS OUT
```

```
# StateMachine
```

```
@rp2.asm_pio(set_init=rp2.PIO.OUT_HIGH)
```

```
def drdy_cs():
```

コードは次のページで解説

```
# ステートマシンを定義というかインスタンス作成みたいな？
```

```
sm = rp2.StateMachine(0, drdy_cs, freq=125000000, set_base=Pin(CS))
```

```
# 定義したステートマシンを実行
```

```
sm.active(1)
```

ステートマシンのコード関数
次ページ

ステートマシンソース2

```
def drdy_cs():  
    # CS=L期間(24bit) DRDY立ち上がりを待つ  
    wait(1, gpio, 26)  
    # CS=Highにする  
    set(pins, 1)  
    # DRDY立ち下がりを待つ  
    wait(0, gpio, 26)  
    # STATUS data(8bit)の間はCS=Hを維持する  
    # DCLK立ち上がり7回待つためにx=6セット  
    set(x, 6)  
    label("bit_loop1") # ここにjmpしてくる  
        # DCLK立ち下がり待ち  
        wait(0, gpio, 27)  
        # DCLK立ち上がり待ち  
        wait(1, gpio, 27)  
        # x!=0ならばbit_loop1に飛ぶ xデクリメント  
        jmp(x_dec, "bit_loop1")  
    # STATUS data最後のbit0が終わるのを待つ  
    # DCLK L wait  
    wait(0, gpio, 27)  
    # AD data bit23が始まるのでCS=Lにする  
    set(pins, 0)
```

はい、ここ大事なところ!

set()でGPIO出力 H/L設定

wait()でGPIO入力 H/L待ち

label() + jmp()でループ処理

たった4つの命令しか使っていない。

実際のオシロ波形



めでたしめでたし

- デジタル系の設計をドジっても、追加回路としてRP2040のステートマシンを使えば、**ロジックIC**で面倒な回路を組む必要がなくなる
- ほぼ**ロジックIC**並みの反応速度(125MHz 8ns)で動作する
- 論理はソースコードなので**トライ&エラー**が簡単に出来る
- Xiao RP2040ならば**基板サイズも小さい**ので基板に載せやすい
- そもそもちゃんとインターフェースを設計すれば良い話ですが、**基板作って見たらアレ？**みたいな話は良くあるのでは？←オレだけ？

AI協調による多視点仕様策定プロセス

イノベーティブ思考法との融合による上流 工程革新

夢・アーキテクティング工房 片山 立

SWEST27（組み込みシステム技術に関するサマーワークショップ）

2025年8月28-29日 15分発表



現在のAI活用状況と限界

生成AI活用の現状分析：下流工程偏重の課題

主流な活用領域：「どのようにつくるか」

- コード生成・プログラミング支援
- 設計ドキュメント作成支援
- テストケース自動生成
- 実装レベルのリファクタリング
- 技術文書・API仕様書作成

市場動向：企業でのコード生成AI利用率**85%**、開発効率向上報告**40%**

現状の限界と課題

上流工程への適用不足：最も重要な「**何をつくるか**」の段階（要求分析・仕様策定）への適用が限定的

具体的な問題点：

- 属人的プロセス：経験豊富なエンジニアの直感に依存
- 多角的検討の困難さ：単一視点での意思決定
- 客観的根拠の不足：主観的判断中心

機会の認識：生成AIの能力向上により、従来困難だった上流工程での体系的AI協調が実現可能に

上流工程の課題と重要性

「何をつくるか」段階がシステム開発成否を決定する

上流工程の現状課題

- 属人的な要求・仕様策定プロセス
- 経験と直感に基づく意思決定
- 体系的な分析手法の不在
- 客観的な根拠資料の不足

多角的検討の困難さ

- 市場・技術・ビジネス観点の同時分析困難
- ステークホルダー要求の複雑な調整
- トレードオフ関係の体系的把握困難
- 見落としやすい要求の存在

課題の定量的影響

70%：プロジェクト失敗の要因（要求・仕様起因）

10倍：後工程修正コスト（上流比）

システム開発成否の最重要決定要因：「何をつくるか」の決定品質が、プロジェクト全体の成否を左右します。

解決への道筋：生成AIとの協調により、従来不可能だった「体系的・客観的な上流工程」の実現が可能になります。



提案手法の基本コンセプト

イノベーティブ思考法×AI協調による体系的仕様策定

理論的基盤：イノベーティブ思考法

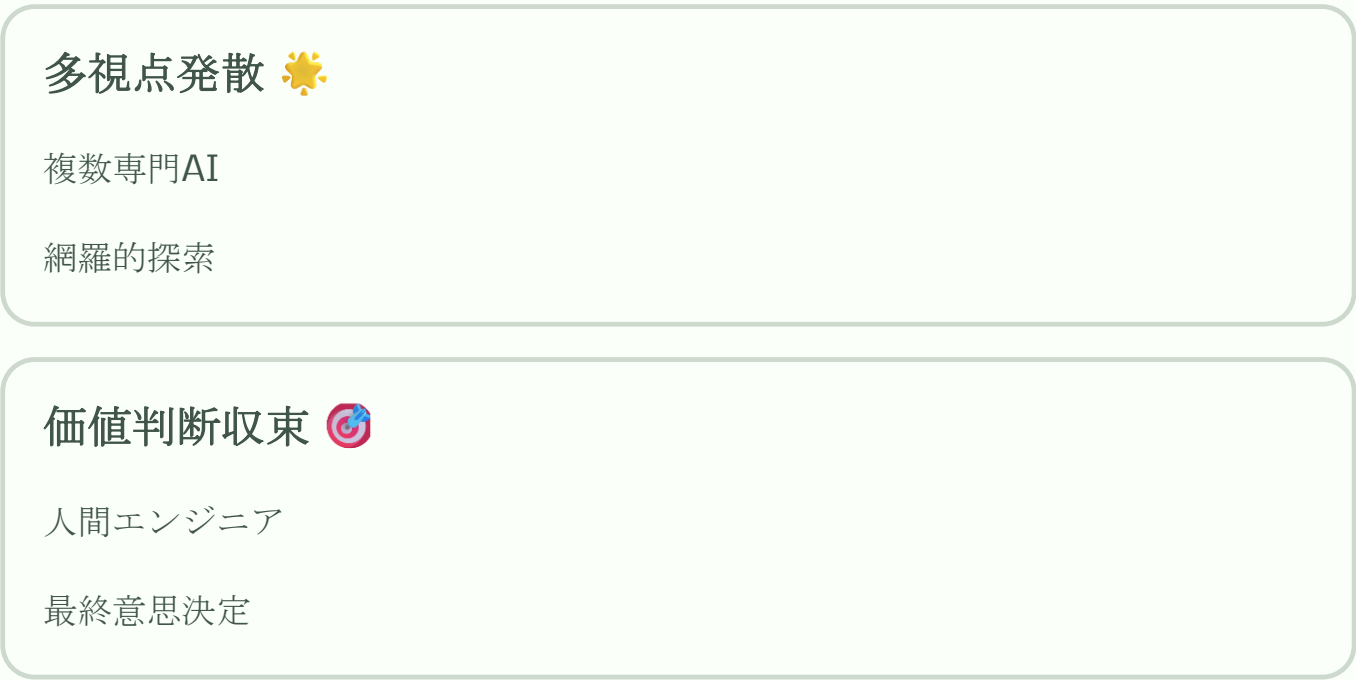
慶応SDM開発の体系的思考フレームワーク

文科省支援により開発された、複雑な問題解決のための構造化思考法



基本プロセス設計

AI協調発散・収束プロセス



新規性

- 体系的AI協調：理論基盤に基づく
- 多視点同時分析：従来困難な領域
- 上流工程特化：最重要段階への焦点

専門*AI*役割分担詳細

多角的分析を担う専門*AI*群の構成と責務



市場分析*AI*

- ユーザーニーズ探索・分析
- 競合他社動向調査
- 市場セグメント分析



ビジネス*AI*

- 収益性・実現性評価
- ビジネスモデル最適化
- 事業戦略整合性確認



技術動向*AI*

- 技術可能性評価
- 実現制約分析
- 将来技術トレンド予測



品質・安全性*AI*

- リスク分析・評価
- 規制・規格対応
- 信頼性設計支援

AI間協調メカニズム：並行分析 → 結果統合 → 整合性確認 → 提案生成

各専門AIが独自の視点から分析を行い、その結果を統合することで、**多角的かつ包括的な仕様策定**を実現します。

AI協調プロセス詳細

発散・収束の体系的実行メカニズム

発散フェーズ：多角的探索

各専門AIによる並行分析

市場分析AI 📊

技術動向AI 🔬

ビジネスAI 💼

品質・安全AI 🛡️

発散フェーズの特徴：

- 網羅的分析：人間単独では困難な多角的同時分析
- バイアス除去：客観的データ・知識ベース活用
- 創発的発見：AI間の相互作用による新視点創出

収束フェーズ：価値判断

人間エンジニアによる意思決定

価値判断・戦略選択

組織方針との整合性確認

トレードオフ関係の最終判断

最終意思決定・責任

収束フェーズの特徴：

- 価値判断：定量化困難な価値観・戦略の反映
- コンテキスト考慮：組織文化・事業環境の統合
- 責任明確化：人間による最終意思決定責任

実証事例設定：話題沸騰ポット多視点仕様策定

SOC定義とステークホルダー分析による体系的アプローチ

System of Concern (SOC) 定義

対象システム：話題沸騰ポット - 家庭用電気式自動湯沸かし器

システム境界：



主要コンポーネント：

- 制御システム (H/W・S/W)
- 加熱・安全監視システム
- ユーザーインターフェース

ステークホルダー分析

Primary Stakeholders：

- エンドユーザー：機能・使いやすさ・安全性
- 製品企画部門：コスト・差別化・売上
- 設計開発部門：技術実現性・品質

要求抽出結果：25+機能要求、15+非機能要求

設計課題

異なる市場セグメント向け3製品の最適仕様策定

- プレミアム市場：拡張性重視
- エコノミー市場：低コスト重視
- 安心志向市場：信頼性・安全性重視



SAI基準アーキテクチャ：拡張性重視設計

モジュール化・プラグイン機構による将来展開対応

Application Layer

- 湯沸かしアプリケーション
- 保温アプリケーション
- 給湯アプリケーション
- 安全監視アプリケーション

Platform Layer

- リアルタイムOS
- デバイスドライバ
- 共通ライブラリ

Service Layer

- 設定管理サービス
- ログ管理サービス
- 診断サービス
- 通信サービス（拡張用）

Hardware Layer

- メインコントローラ（ARM Cortex-M7）
- センサークラスター
- 通信モジュール（Wi-Fi/BT準備）

拡張ポイント設計

1

IConnectedDevice

（IoT機器連携）

2

ICloudService

（クラウド連携）

3

ISmartHome

（スマートホーム）

外部仕様適合

100°C

沸騰温度

±2°Cの精度

≤5W

待機消費電力

3-8分

沸騰完了時間

5年

MTBF目標

圏論的変換プロセス：*SA1*→*SA2/SA3*自動変換

数学的フレームワークによる体系的アーキテクチャ変換

変換関手定義

```
def apply_cost_transformation(sa1):
    transformations = {
        'PlatformLayer': consolidate_platform,
        'Interfaces': simplify_interfaces,
        'ExtensionPoints': remove_extensions,
        'HardwareLayer': integrate_hardware
    }
    return verify_functional_preservation(sa1, sa2)
```

```
def apply_reliability_transformation(sa1):
    enhancements = {
        'SafetyCore': enhance_safety_monitoring,
        'CriticalComponents': add_redundancy,
        'DiagnosticsService': expand_diagnostics,
        'CommunicationLayer': add_security
    }
    return verify_safety_compliance(sa3)
```

圏論的整合性

Architecture Category

Objects: {Component, Interface, Service}

Morphisms:

- depends_on: CompA → CompB
- implements: Comp → Interface
- uses: Comp → Service

変換結果

- 1

SA1
拡張性重視（基準）
- 2

SA2
低コスト（8,000円以下）
- 3

SA3
信頼性・安全性重視

3製品自動生成、数時間の設計時間



AI統合による自動設計エンジン

CategoryGuidedPotDesigner：圏論×生成AI協調システム

AI支援設計エンジン

```
class CategoryGuidedPotDesigner:
    def generate_cost_optimized_architecture(self, sa1):
        # 圏論的制約の抽出
        cost_constraints = self.category_theory.extract_cost_constraints()
        # 変換関手G12の適用
        structural_sa2 = self.apply_functor_G12(sa1)
        # AIによる詳細設計最適化
        optimized_sa2 = self.ai_engine.optimize_architecture(
            base=structural_sa2,
            objective='minimize_cost',
            constraints=cost_constraints,
            domain_knowledge='embedded_system_cost_patterns'
        )
        return self.verify_categorical_consistency(sa1, optimized_sa2)
```

設計結果比較

項目	SA1	SA2	SA3
製造コスト	12,000円	7,800円	15,000円
拡張性	高	基本	中
MTBF	5年	3年	8年
安全機能	標準	簡素	強化

実現効果

80%

設計時間短縮

100%

制約充足確認

3種類

同時最適化

従来手法との優位性

AI協調による仕様策定の革新的効果

定量的効果比較

評価項目	従来手法	提案手法	改善効果
仕様策定時間	数週間	数時間	80%短縮
検討観点数	2-3観点	4+観点	2倍拡大
要求抜け漏れ	15-20%	5%以下	70%削減
仕様変更回数	5-8回	1-2回	75%削減
根拠資料数	5-10件	50+件	5倍増加

革新的効果

探索範囲の飛躍的拡大

- 人間単独では困難な多角的同時分析
- 従来見落とされがちな観点の網羅
- 大量データ・知識ベースの活用

客観性の向上

- データ駆動による根拠のある提案
- 個人バイアスの排除・軽減
- 市場データ・技術動向の客観的反映

プロセスの可視化

- 発散・収束過程の完全な透明性
- 意思決定根拠の記録・トレーサビリティ
- 検討プロセスの再現可能性

組織導入効果

企業・チームへの導入メリットと変革インパクト

仕様変更リスク削減



仕様変更回数削減



後工程手戻り削減

上流工程での十分な検討による効果：

- 初期段階での多角的・網羅的検討
- 見落としやすい要求の早期発見
- ステークホルダー要求の体系的整理

若手育成効果

思考プロセスの可視化

- ベテランの暗黙知の形式化
- 分析手法の段階的学習
- 判断根拠の透明化

実践的スキル習得

- AI協調スキルの習得
- 多視点分析能力の開発
- システム思考の強化

属人性の解消



Before

ベテラン依存

After

組織的能力

- 体系化されたプロセス
- 知識・経験の組織資産化
- 標準化された分析手法
- 再現可能な品質確保

50%

習得期間短縮

3倍

学習効率向上

想定課題と対策

実用化に向けた課題認識と解決アプローチ

想定される課題

AI提案の品質管理

- 誤情報・幻覚への対処
- バイアス・偏見の影響
- 一貫性・整合性の確保

人間の判断支援

- 適切な収束判断の困難さ
- AI依存度の管理
- 責任の所在不明確化

組織変化への対応

- 新しいワークスタイルへの適応
- 既存プロセス・文化との摩擦
- スキル・役割の再定義

対策アプローチ

AI品質保証体制

- 多段階検証システム：複数AI相互チェック、外部データソースとの照合
- 信頼度指標の導入：AI提案の確信度表示、情報源の信頼性評価
- 継続的学習・改善：フィードバックループの構築

人間・AI協調ガイドライン

- 明確な役割分担定義：AI（情報分析・選択肢生成）、人間（価値判断・最終決定）
- 判断支援ツールの開発：トレードオフ可視化、意思決定マトリックス

段階的導入戦略

01

パイロット導入

02

部分的拡張

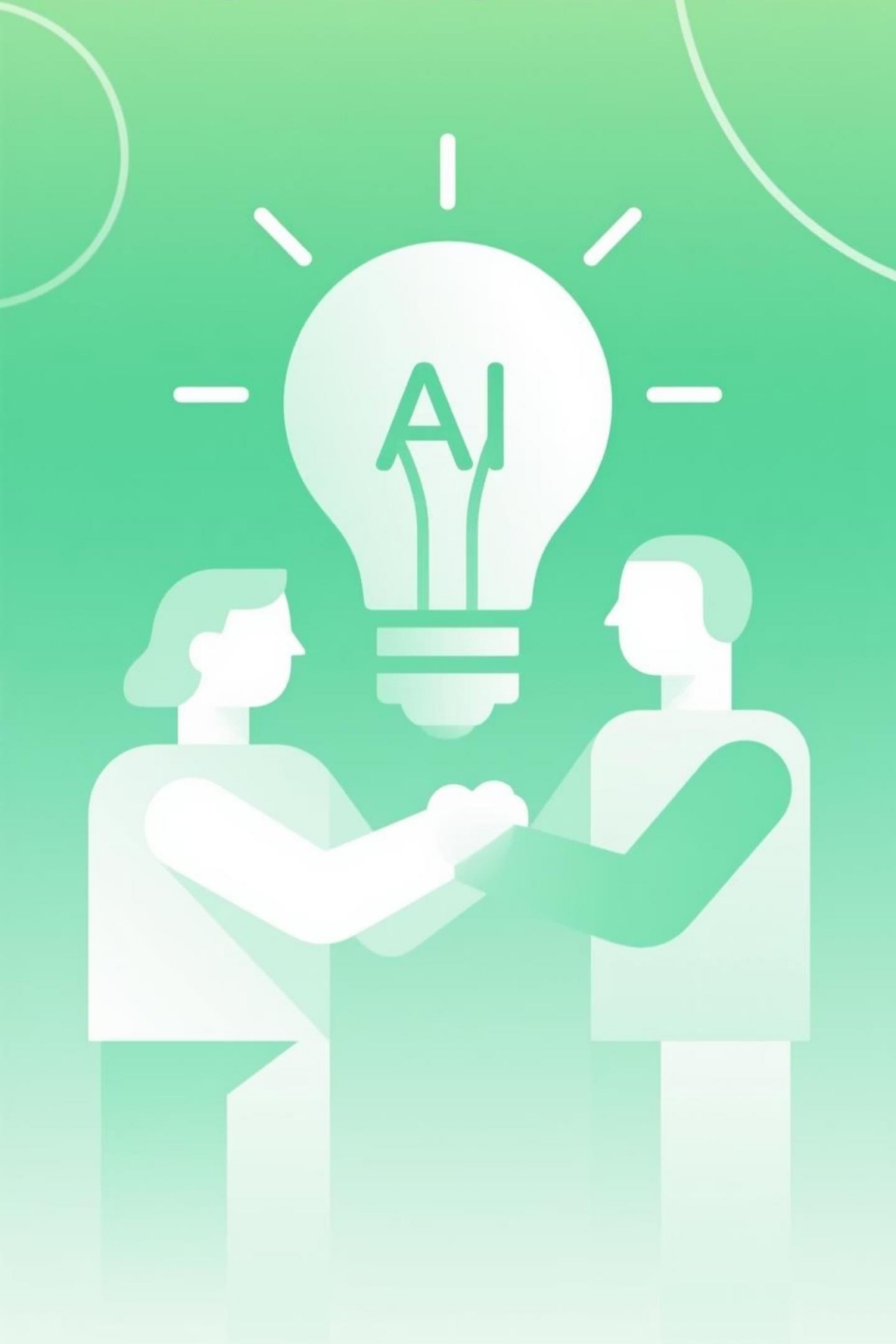
03

全社展開

04

継続改善

成功の鍵：段階的導入と継続的改善により、組織とAIの両方が成長する好循環を構築



まとめ・*SWEST*議論希望項目

システム開発新時代への提言とコミュニティ議論への期待

研究成果のまとめ

主要な結論：「何をつくるか」段階でのAI協調は、従来困難だった上流工程の体系化を実現し、システム開発の根本的革新をもたらす可能性を持ちます

80%

設計時間短縮

70%

要求抜け漏れ削減

2倍

検討観点拡大

5倍

根拠資料増加

*SWEST*議論希望項目

業界別最適化パターン

- 組み込みシステム特有の課題
- ドメイン知識とAI協調

技術的深化議論

- 創発的対話促進手法
- 主観・客観統合方法

組織・人材開発

- 混在チーム運営手法
- スキル・役割の再定義
- 若手育成プログラム設計

システム開発の新しい地平の開拓

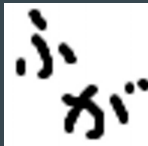
人間とAIの創造的協調によるイノベーション創出基盤の確立

Matterと日本の家電

...

2025-08-28 SWEST27 EmbLT

自己紹介

- 井田 健太
- X(twitter): @ciniml → 
- おしごと: Nature Remoシリーズのファームウェア開発
 - まえはAMD FPGAさわってました
- インターフェースのGOWIN付録小冊子書きました→
- 組込みRust本→



スマートホームとは

- おうちの状態をセンサを使って測定・判定
- おうちの状態を変更・制御
 - e.g. 在室センサが反応したら照明とエアコンをつける
- 現在普及してきているのは、
スマートスピーカを使ったスマートホームが多い
 - Amazon Alexa (Echo)
 - Google Home
 - Apple Home (HomePod)
- スマートスピーカを使った音声制御・自動制御

スマートホーム対応機器の課題

- 異なるベンダのスマートスピーカに対応・接続が必要
- 各社異なるWeb API経由で接続
 - Amazon Alexa Skill
 - Google Home API
 - Apple Home API
- 各APIに対応したうえで各社が要求する認証試験を受ける必要がある
 - とても面倒！
- スマートホームなのにWeb API経由の接続なのでインターネット接続が必須
 - 可用性
 - レイテンシ
 - セキュリティ

Matter

- CSA (Connected Standards Alliance) が管理・策定する
スマートホーム用のプロトコル
 - CSAの前進はZigbee Alliance
- ベンダに依存しない共通プロトコル
- ローカルネットワーク通信
 - Ethernet, Wi-Fi, Thread状のIPv6
- デバイス認証による正規認証デバイスの識別
 - 野良デバイスは警告が出る (コントローラ依存)
- 半年ごとに更新
 - 2022年10月4日にv1.0リリース、現在の最新はv1.4.2

Nature Remo Lapis

- 2024年に発売したMatter対応スマートリモコン
 - スマートリモコン:
赤外線家電をスマートフォンやスマートスピーカーから制御するためのIoTリモコン・デバイス
- **Matter 1.2** に対応
- 新たに**Door Lock Cluster** に対応
 - 連携しているQrio LockをMatter経由で操作可能
- 温湿度センサーを搭載
 - Matterの温湿度センサーを実装
 - エアコンに対応するMatterデバイスの現在温度としても使用



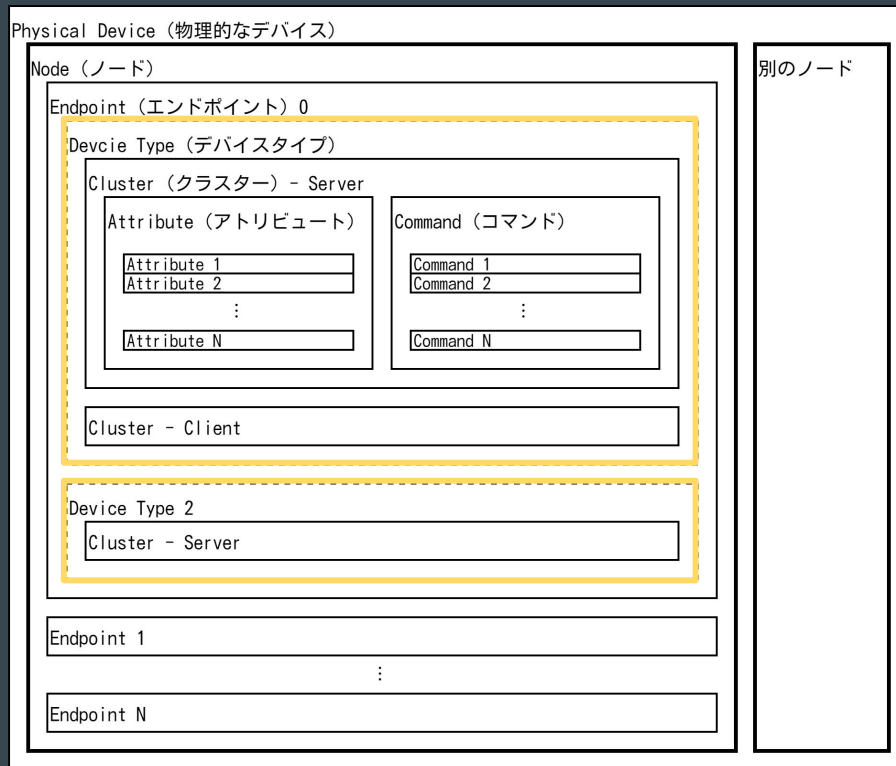
https://shop.nature.global/products/nature-remo-lapis?srsId=AfmBOooCCFwzXaUIHsLjheKr_dfbn8gxNY9sQbbvD0kvRJ7Zej9dFo0

Matterのデータモデル

- スマートホーム機器の種別を表す**デバイス・タイプ**
 - 空調 (Thermostat)、温度センサ、湿度センサ、ドアロック、 etc...
- スマートホーム機器の機能を表す**クラスター**
 - 空調 (Thermostat)、温度センサ、湿度センサ、ドアロック、 etc...
 - ファームウェアアップデート (OTA Requester)、

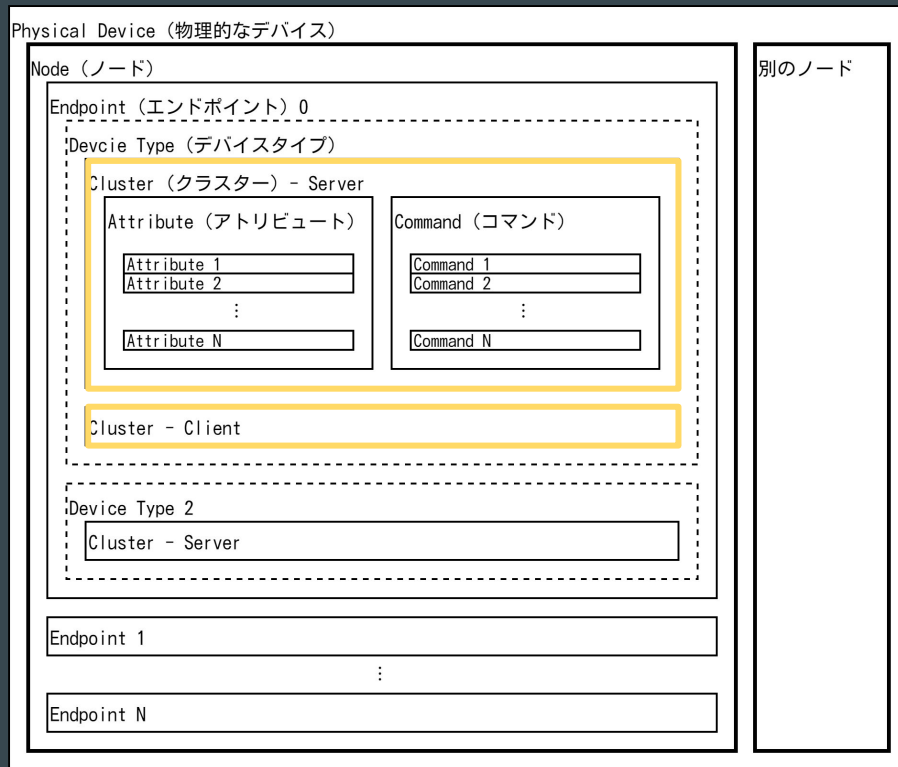
Matterデバイスの基本的な構成要素

- デバイスタイプ
- 1つのまとまった機能の集合
- e.g. ON/OFFライト
サーモスタット
ON/OFFプラグイン
- Matter規格で種類を規定
- 各デバイスタイプの規格に基づき
1つ以上のクラスターを実装



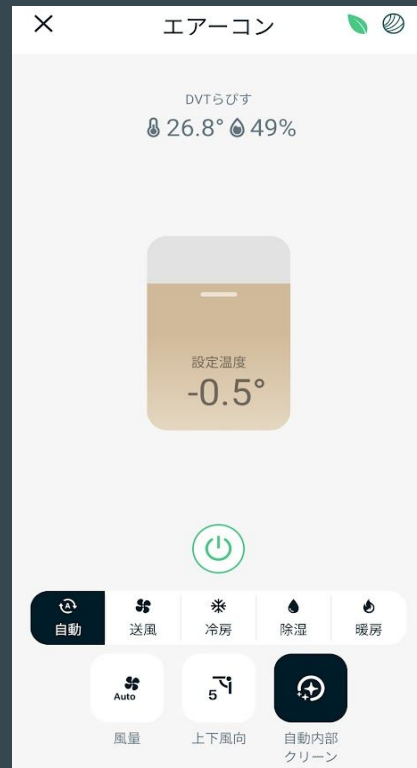
Matterデバイスの基本的な構成要素

- クラスター
- 機能の最小単位
- サーバー
 - 機能を他のノードに提供する
- クライアント
 - 他のノードの機能を利用する
- Matter規格で規定
 - カスタムクラスターも作れる
- e.g. サーモスタット
ファンコントロール
ON/OFFライト
- 読み書きできるアトリビュート
- 呼び出せるコマンド



エアコンをMatter対応にするときの課題

- エアコンの除湿・送風に非対応
- **Nature Remo** 自体は除湿・送風に対応
- MatterのThermostat Clusterでは
除湿・送風モード自体は**オプション**
- 除湿・送風モードがあるかどうかを表す仕組みが無い
 - c.f. AUTOモードはAUTO Featureで表せる
- →主要Matterコントローラでは**暖房・冷房のみ対応**
- **Matter** の仕様変更が必要なので **CSA** にフィードバック済み



まとめ

- Matterはいいぞ
 - とうかさスマートホームデバイスをつくるにあたって避けて通れない
- Matterの規格は日本向けじゃないところがあるので
日本の家電メーカーがんばってくれ

宣伝

- CQ出版のMatterセミナーを10/14にやります。
- NXP RW612搭載ボード FRDM-RW612
- 【特別企画】組み込みマイコン入門講座～定番シリアル通信からAIまで～ | Interface – CQ出版

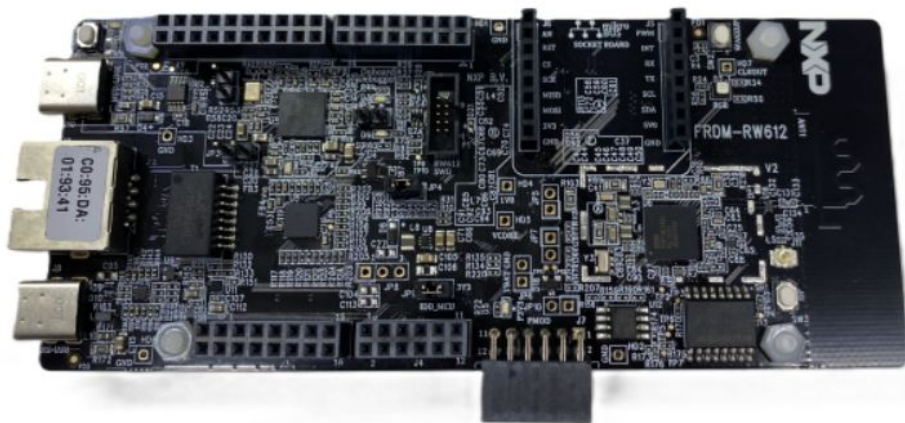
宣伝

- CQ出版の
- NXP RW6
- 【特別企画
- Interface -
-

● セミナ開催スケジュール

最新の申し込み情報は本ページや[Interfaceの公式X](#)（旧Twitter）アカウントなどでお知らせします。

▼テーマ②：スマート・ホーム向け通信規格「Matter」のファームウェア開発入門



実習で使用する評価ボードFRDM-RW612（NXPセミコンダクターズ）

- ・開催日：2025年10月14日 10:00-17:00
- ・受講料：5,000円（ボード・お弁当付き，税込み）
- ・講師：井田 健太氏（Nature株式会社）

[申し込みページはこちら](#)

で～