TEPRA Lite ではじめる BLE リバースエンジニアリング

末田 卓巳 @puhitaku

第54回 情報科学若手の会　#wakate2021

自己紹介

末田 卓巳　Takumi Sueda　@puhitaku

フリー開発者。NICT (情報通信研究機構) で機器解析を手掛けるほか、米住宅ベンチャー HOMMA などで開発に従事。

好きなもの

- 全レイヤーの技術、特に低いもの
- リバースエンジニアリング
- 牛丼
- 音楽

これまでの参加歴

- 若手の会 2014 〜 2018
- 若手の会 2020
  「電子辞書は組み込み Linux の夢を見るか？」

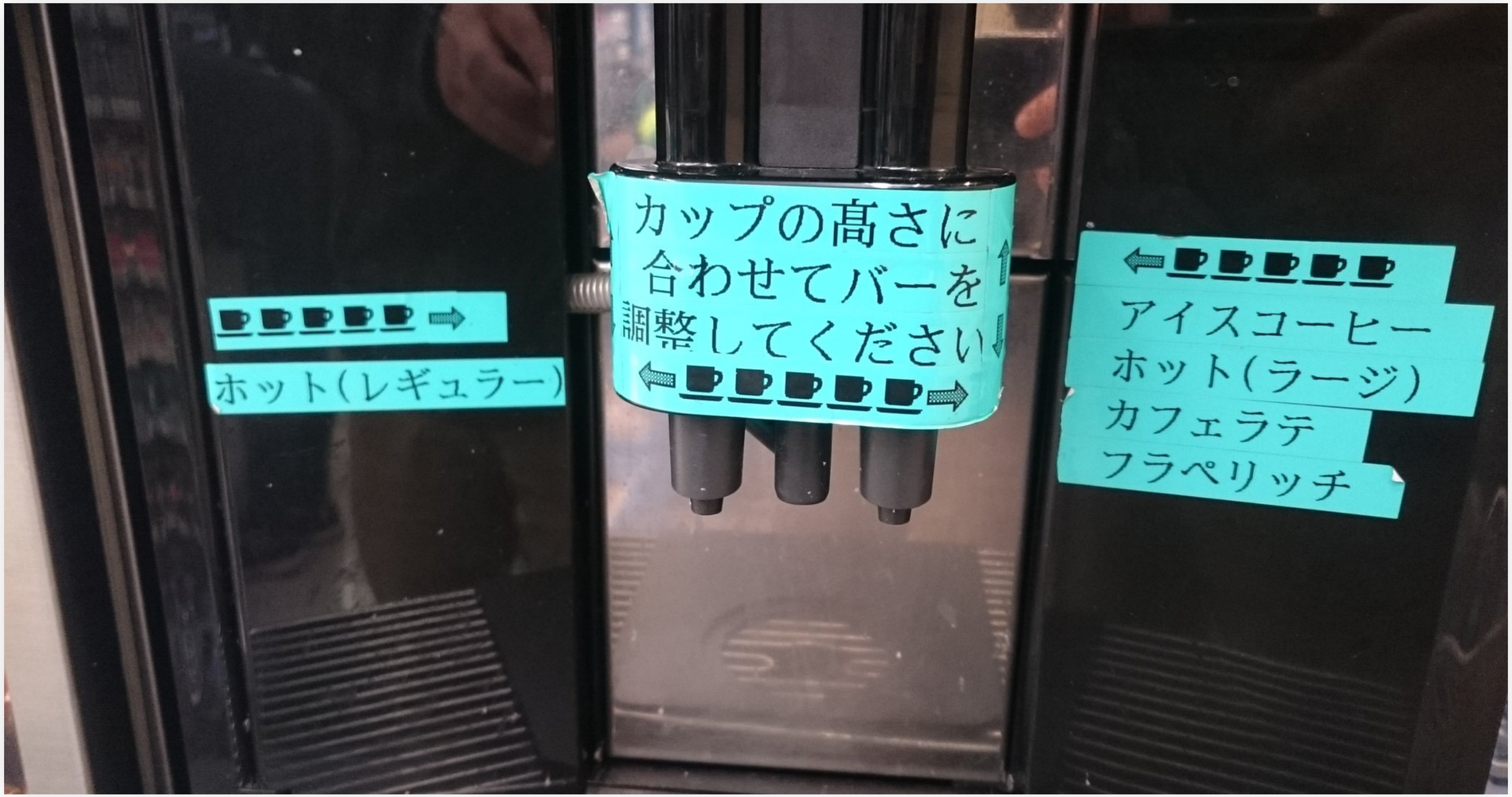
TEPRA（テプラ）とは？

テプラとは

みんな大好きテプラ

https://twitter.com/ejo090/status/425523340430696448

テプラとは



みんな大好きテプラ

https://twitter.com/okayan08/status/797255664464601089

テプラとは

みんな大好きテプラ

https://twitter.com/okayan08/status/565860495744184322

テプラとは

1000万台以上の売上を誇るキングジム社の大ヒット商品

https://www.kingjim.co.jp/products/tepra/

テプラにはちょっとした問題がある

問題: 手で打ち込むのがめんどい

テプラの抱える問題

このふにょふにょのキーボードがきつい

https://www.kingjim.co.jp/products/tepra/sr370.html

テプラの抱える問題

数枚をデコる程度なら楽しいけど ラベルを100枚全部違う文字で出すとしたら？

https://www.kingjim.co.jp/products/tepra/sr370.html

もちろんパソコンにつないで
印刷を自動化できそうなテプラもある

けれど…

テプラの抱える問題

ハイスペック

「テプラ」PRO SR-R980

本体価格/1台

¥42,000＋消費税

対応テープ幅

4〜36mm

ATOK　Win Mac　USB

高い

https://www.kingjim.co.jp/products/brand/tepra-pro.html

テプラの抱える問題

「テプラ」PRO SR3500P

本体価格/1台

¥14,800＋消費税

対応テープ幅

4〜24mm

Win

USB

キーボードと画面と対応OSを削った限界仕様でようやく15,000円

https://www.kingjim.co.jp/products/brand/tepra-pro.html

「1万払うしかないのか…」

「1万払うしかないのか…」

いえ！その必要はありません！

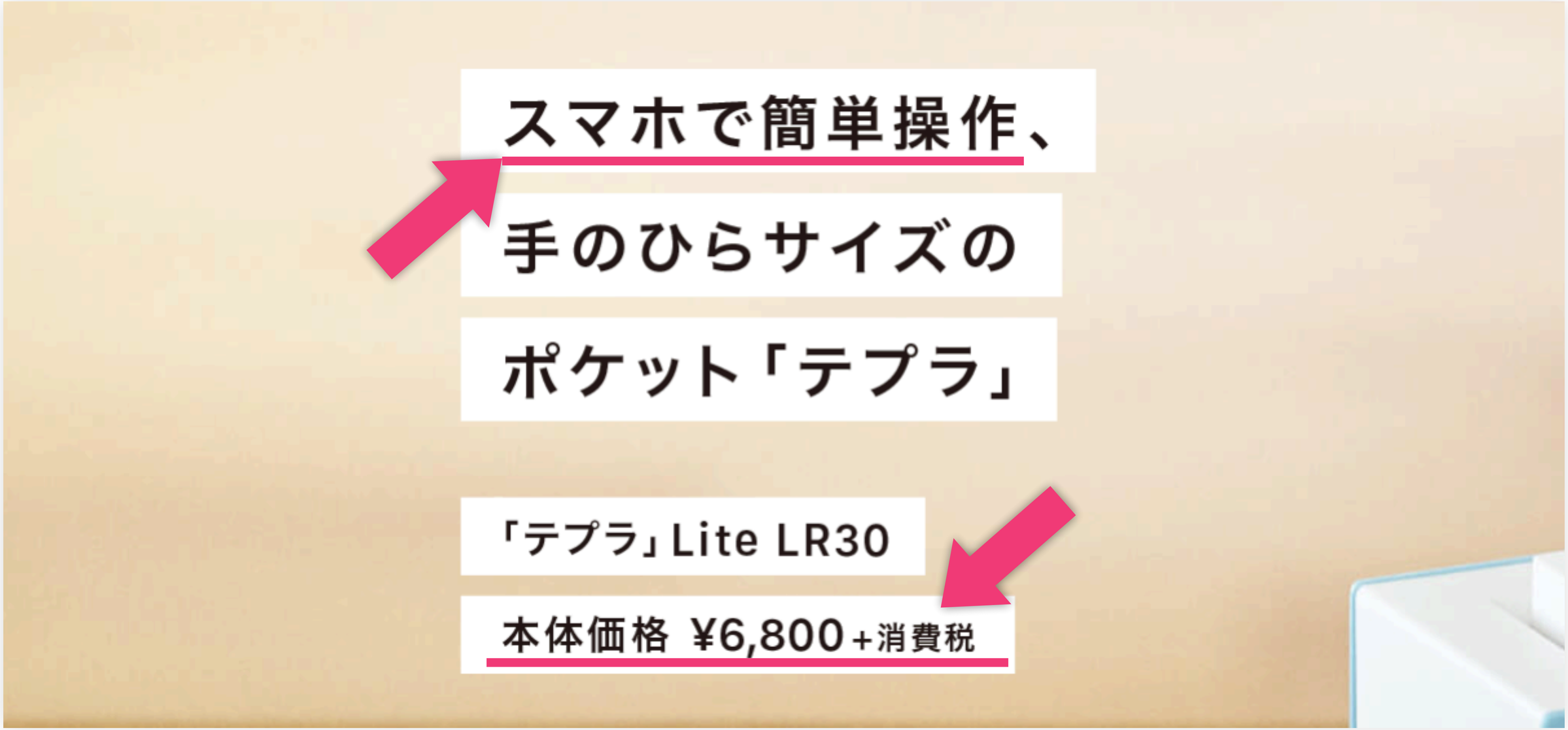
ここで登場するのが
TEPRA Lite LR30

TEPRA Lite LR30

ページがまずシャレオツ

https://www.kingjim.co.jp/sp/lr30/

TEPRA Lite LR30



https://www.kingjim.co.jp/sp/lr30/

TEPRA Lite LR30

実際のスクショ

https://www.kingjim.co.jp/sp/lr30/

TEPRA Lite LR30

プリントボタンを押すといい感じに印刷される

簡単！安い！早い！

TEPRA Lite LR30

しかしまだ「大量にラベルを刷るのが大変」という問題が解決してない

https://www.kingjim.co.jp/products/tepra/sr370.html

スマホアプリには
データインポート機能も
自動印刷機能もないので
自力でなんとかする必要がある

じゃあ仕方ない！

通信内容を
リバースエンジニアリングして
自分のプログラムから
印刷できるようにしよう！

TEPRA Lite ではじめる BLE リバースエンジニアリング

末田 卓巳 @puhitaku

第54回 情報科学若手の会　#wakate2021

本題入ります

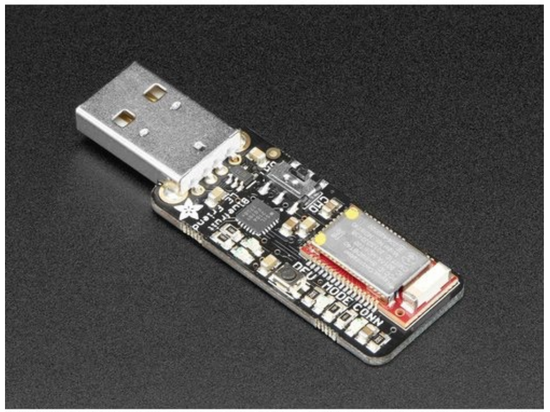
リバースエンジニアリングの手法

Bluetooth Low Energy (BLE) で通信

https://www.kingjim.co.jp/sp/lr30/feature/index.html

BLE の通信内容を見る方法

1. Android の APK をバラして読む
2. BLE sniffer を使って飛んでいる電波を読む
   → 実際の通信を見るほうが早いので 2 を採用



Bluefruit LE Sniffer

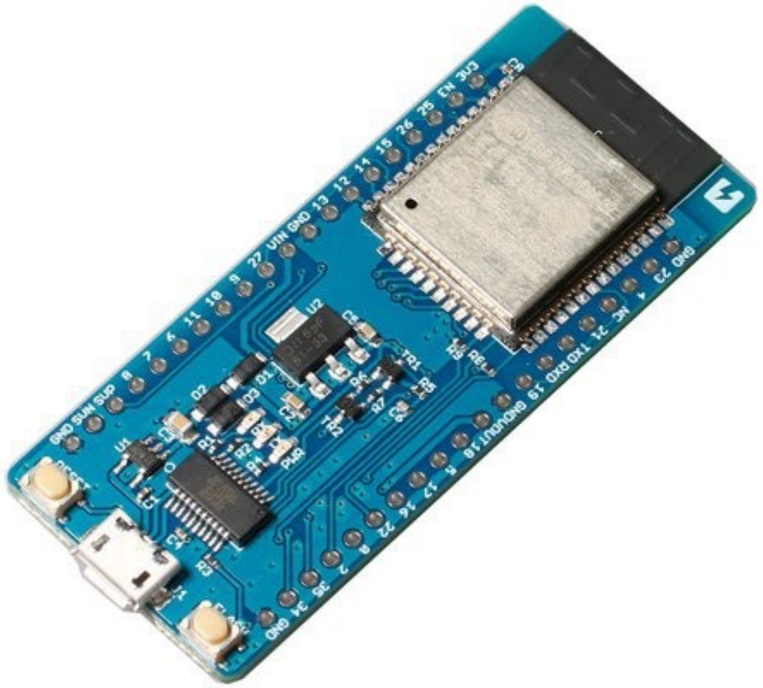
https://www.switch-science.com/catalog/3347/

BLE の通信内容をPCで再現する？

- **PC の Bluetooth 開発は地獄**
- 「副作用が残って2回目に上手く動かない…」
- 「Bluetooth スタックがおかしくなってパソコンごと再起動しないと動かない…」
- 「Bluez（Linux の BT スタック）の Python バインディングが終わってる…」
- 「この OS だと動かない…」
- やりたくない😇

BLE の通信内容をマイコンで再現する！

- **PC の地獄を全部解決できる（個人の感想です）**
- 軽量な Bluetooth スタック
- ペラッペラに薄い API
- リセットボタンでどんな副作用（状態）でも
  一瞬で消去
- MicroPython で短くエレガントに書ける
- 秒で Wi-Fi と BLE を喋らせられる
- 最高すぎる… 🥰



ESPr® Developer 32

https://www.switch-science.com/catalog/3210/

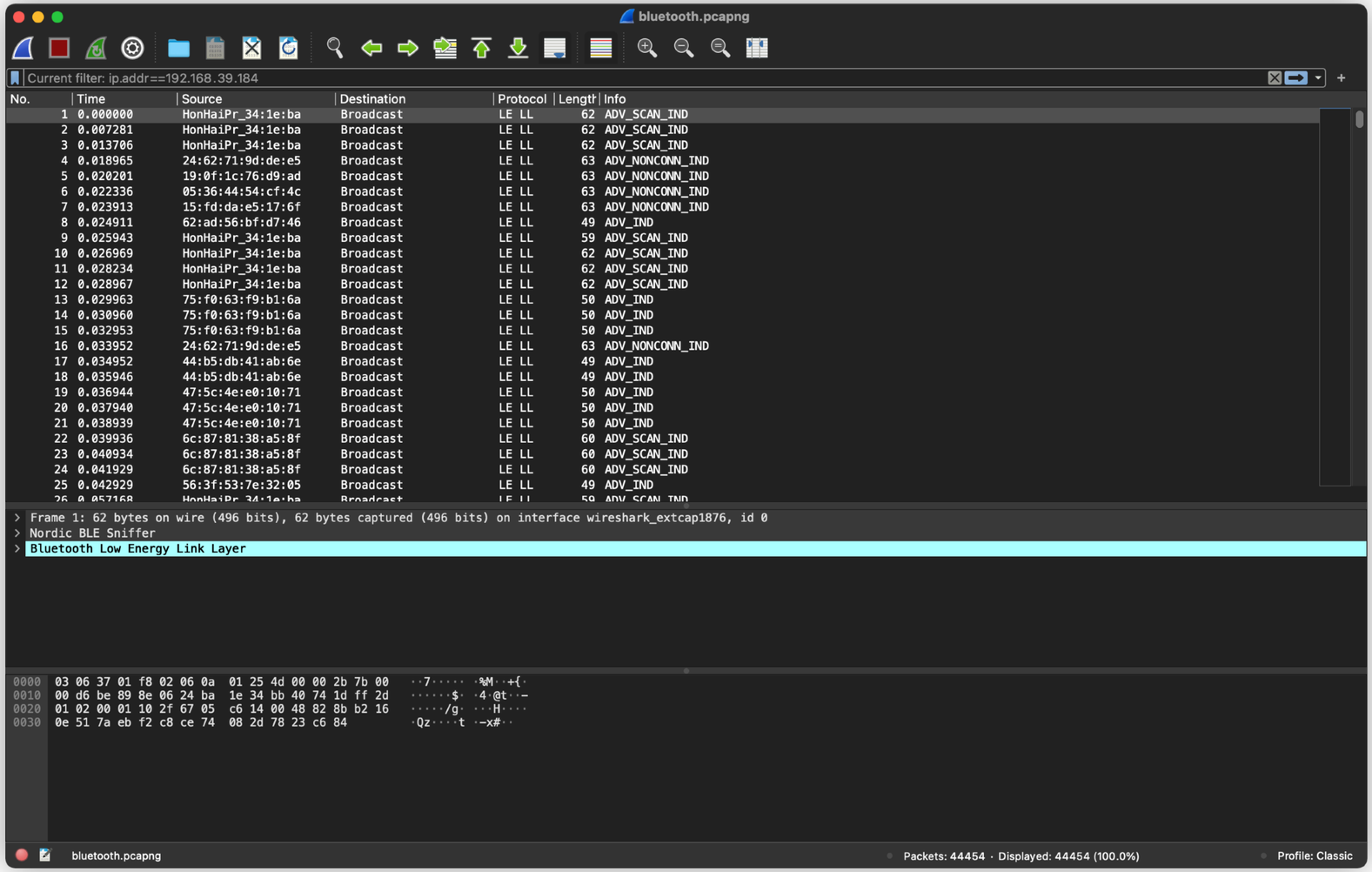
リバースエンジニアリングの流れ

1. 通信をスニッフする
2. BLE の Service と Characteristic を見つける
   （用語解説は後述）
3. 印刷した絵柄と通信を見比べていろいろ察する
4. 察した結果をコードにする
5. 印刷できるまで 2〜4 を繰り返す
6. パソコンから手軽に印刷できる API を生やす
7. 完成！

岡山駅前の桃太郎

1. 通信をスニッフする

1. 通信をスニッフする



吸いました

Bluefruit LE Sniffer を macOS (Apple Silicon) で使う

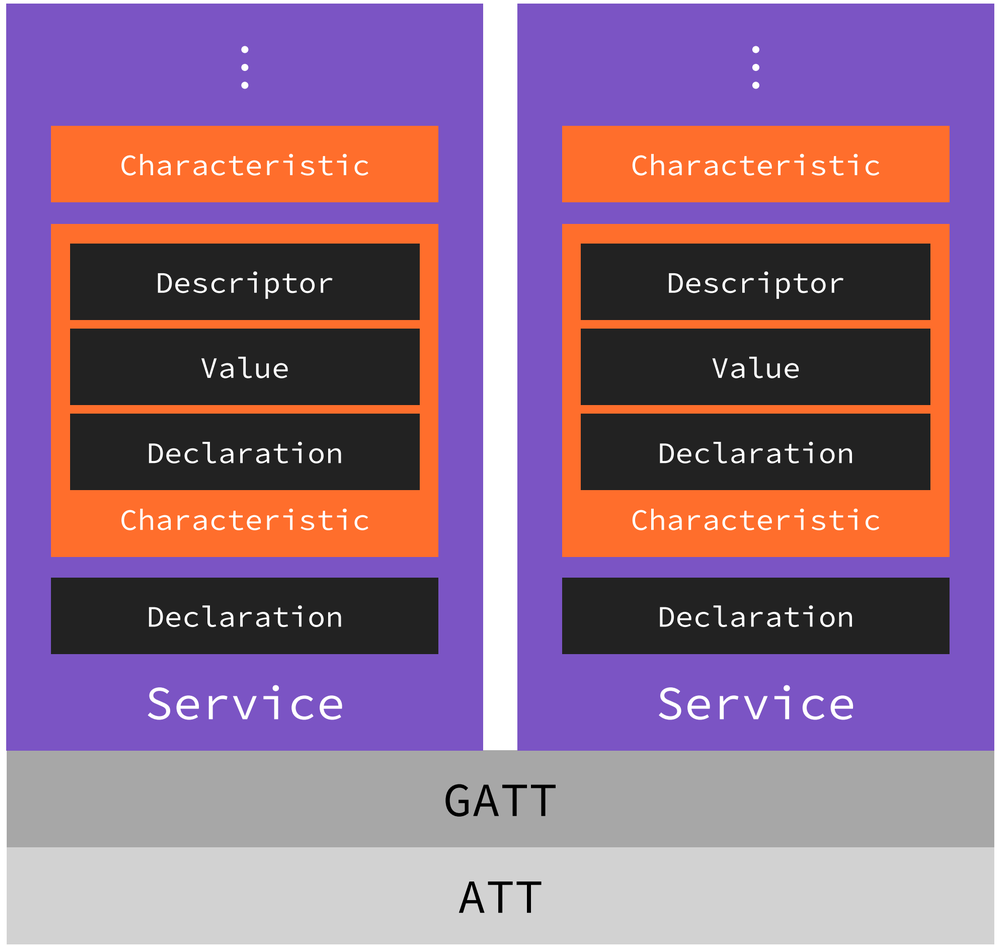
- チップセットのベンダー Nordic が公開している Wireshark 用 sniffer プラグインは Python 2 で書かれている※
- Apple Silicon な Mac だと pyenv で Python 2.7.18 がインストールできない事象に遭遇
- pyenv に PR を出して直しておきました
- macOS + Apple Silicon + pyenv な環境の方はぜひ安心してパケットを吸いまくってください

※ 最新版だと Python 3 になっているが Bluefruit LE Sniffer のファームと対応したものは Python 2

2. BLE の Service と Characteristic を見つける

用語解説

- **ATT (ATTribute Protocol)**
  最大512バイトの値に Attribute Handle という2バイトの通し番号を付けて管理し、読み書きしたり変更を通知させたりできるプロトコル
- **GATT (Generic ATTribute Protocol)**
  複数の Attribute の組み合わせを Service や Characteristic と名付け、ATT の上でアプリケーションを表現できるようにした上位のプロトコル
- **Service**
  その機器が持つ機能を表す概念
- **Characteristic**
  Service に対してコマンドを送ったり値を読んだりするポートあるいはフック付きの変数のようなもの



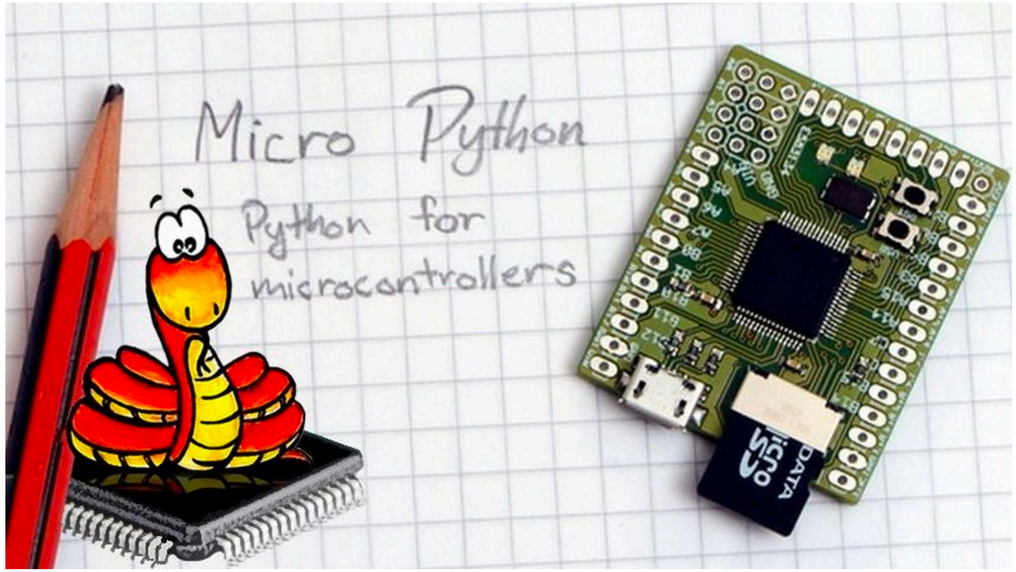
MicroPython で BLE 通信する

MicroPython とは？

- マイコンで動くとても軽い Python 実装
- 対話モード (REPL) もある → 実験しやすい！

Bluetooth API

- bluetooth モジュールを import すると使える
- BLE クラスのメソッドを呼ぶと IRQ で結果が返ってくる大変わかりやすい API を持つ
- たとえば gattc_discover_services を呼ぶと Service の discovery が実行され、prop_write_without_response を呼ぶと Characteristic に値を書き込める



Service と Characteristic を見つけました

- `0x180f` Battery Service … 文字通りバッテリー残量が読める
  - `0x2a19` Battery Level Characteristic
- `0xfff0` TEPRA Lite 独自 Service
  - `0xfff1` TEPRA Lite 独自 Char. … Notify・印刷状況の受信 (RX) 用
  - `0xfff2` TEPRA Lite 独自 Char. … Write Without Response・送信 (TX) 用

Service と Characteristic が全部通し番号で並んでいることがわかる

Service と Characteristic を見つけました

- `0x180f` Battery Service … 文字通りバッテリー残量が読める
  - `0x2a19` Battery Level Characteristic
- `0xfff0` TEPRA Lite 独自 Service
  - `0xfff1` TEPRA Lite 独自 Char. … Notify・印刷状況の受信 (RX) 用
  - `0xfff2` TEPRA Lite 独自 Char. … Write Without Response・送信 (TX) 用

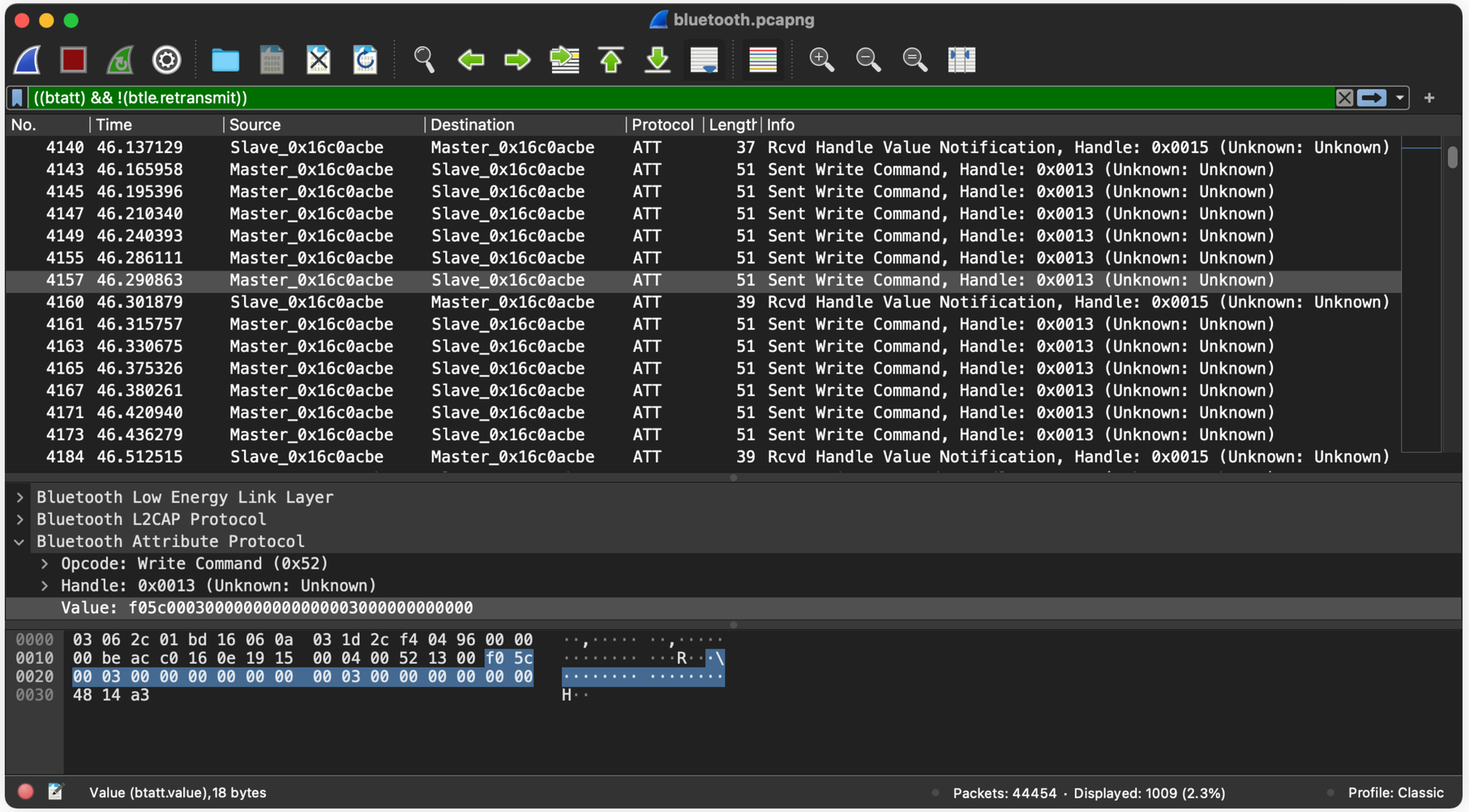
この Characteristic っていう口といい感じに通信すれば
印刷処理が自力で実装可能！！

3. 印刷した絵柄と通信を見比べて
いろいろ察する

3. 絵柄と通信内容を見比べて察する

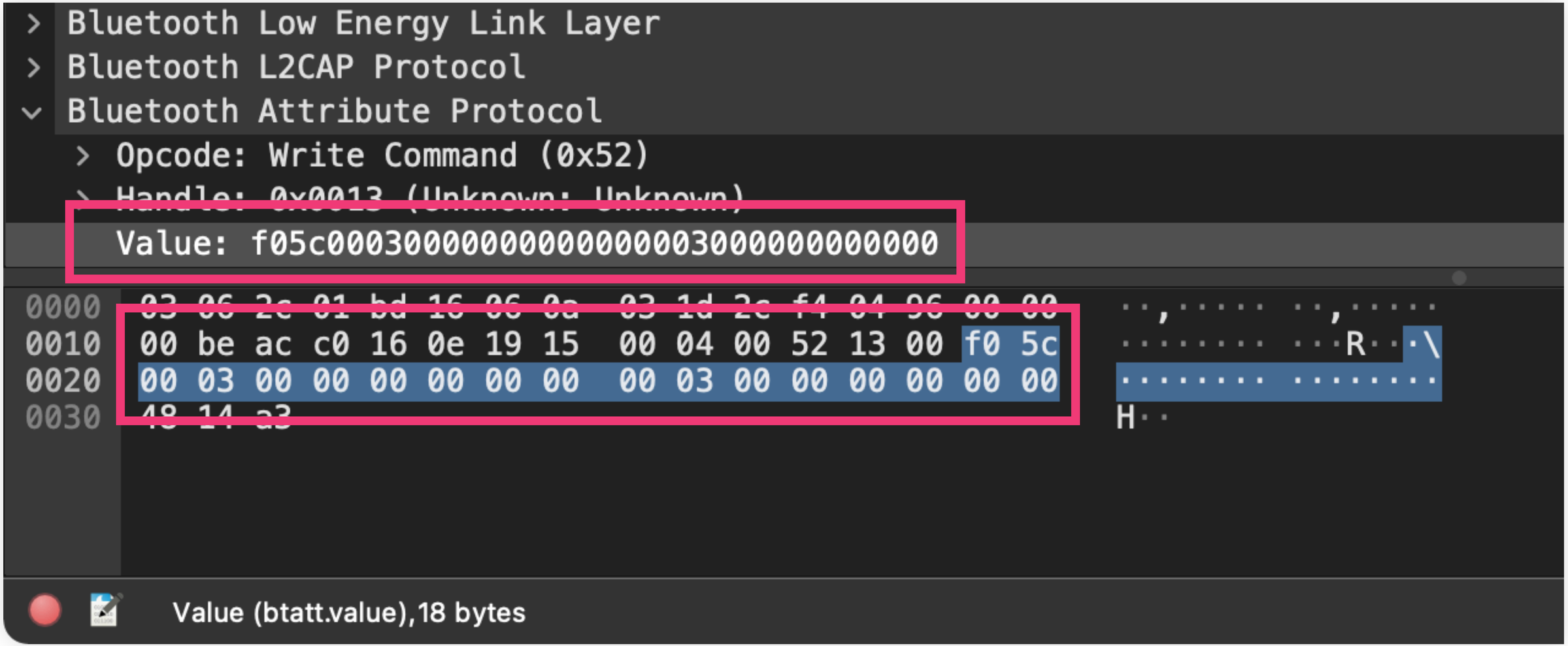
印刷可能範囲の端にハイフンを並べるなどの 異様な印刷データを流し込む

3. 絵柄と通信内容を見比べて察する



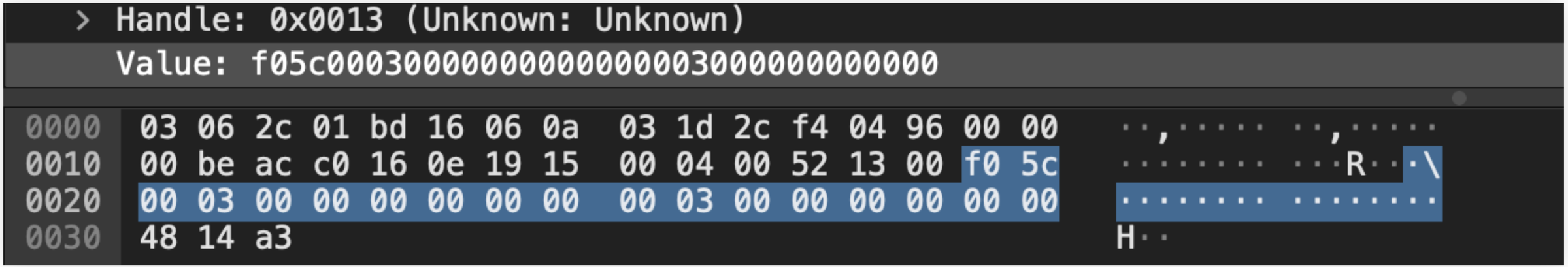
実際のスニッフ結果

3. 絵柄と通信内容を見比べて察する



おや、なにやら連続で大量に送りつけられているデータが…

3. 絵柄と通信内容を見比べて察する



f0 5c = 印刷データを示す2バイト

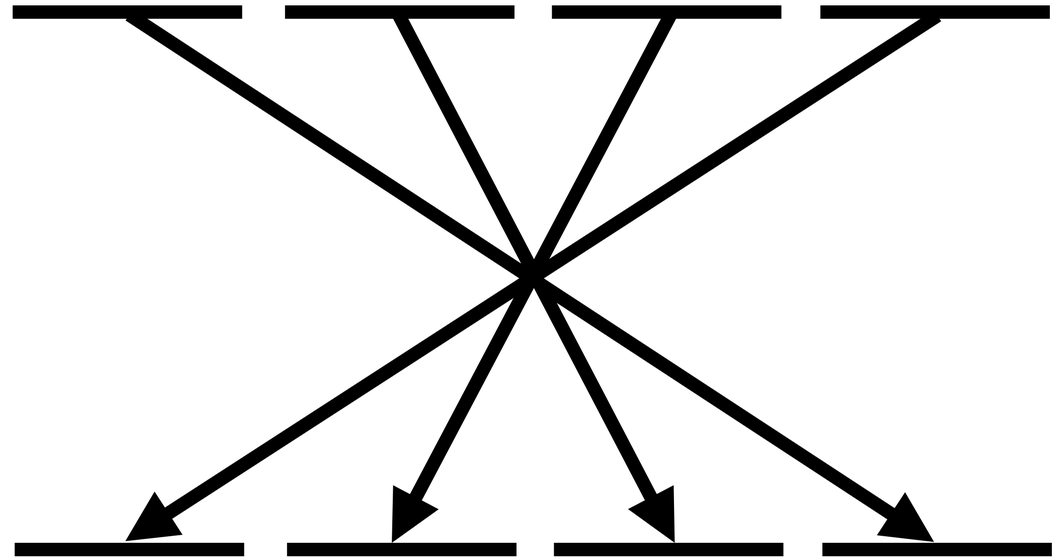
00 03 00 00 00 00 00 00 = 印刷データ1ライン目

00 03 00 00 00 00 00 00 = 印刷データ2ライン目

見えた！

3. 絵柄と通信内容を見比べて察する

Before 00 03 00 00 00 00 00 00



After 00 00 00 00 00 00 00 03

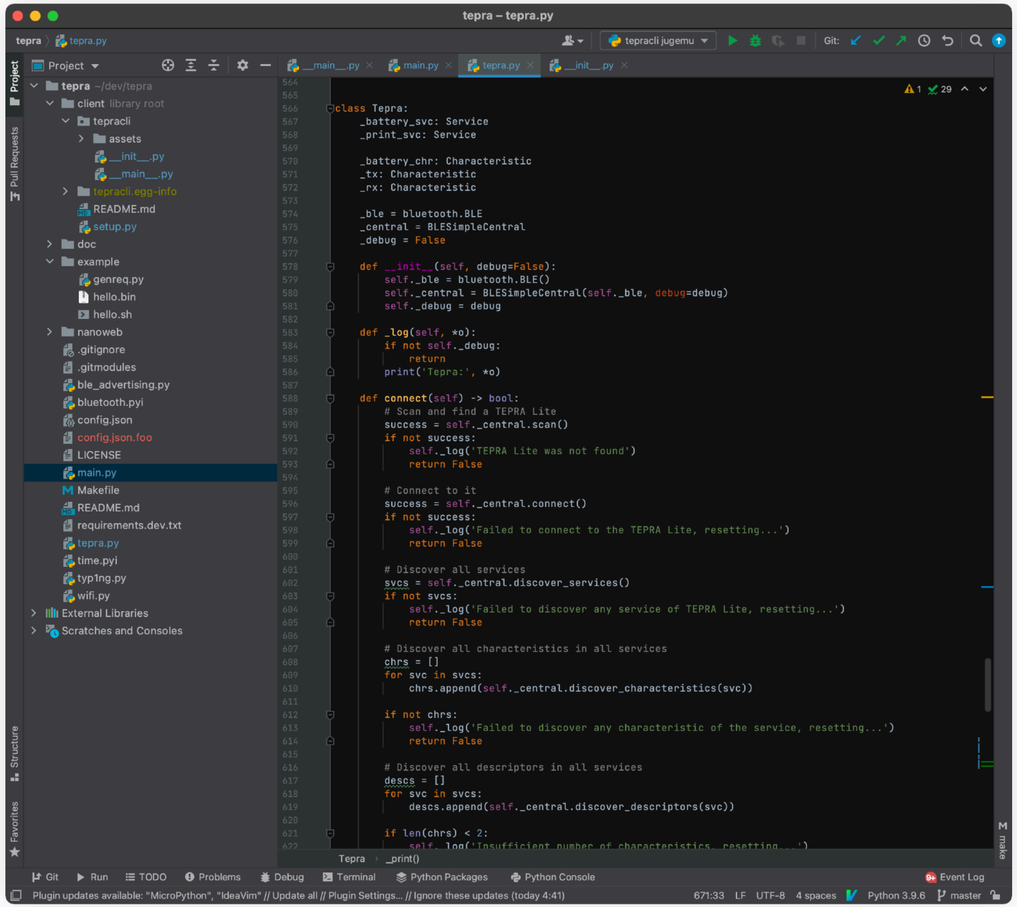
実は 2 Bytes (word) ごとにスワップが必要

スワップした After のバイト列（左）が実際の印刷データ（右）と符合する！

4. 察した結果をコードに落とし込む

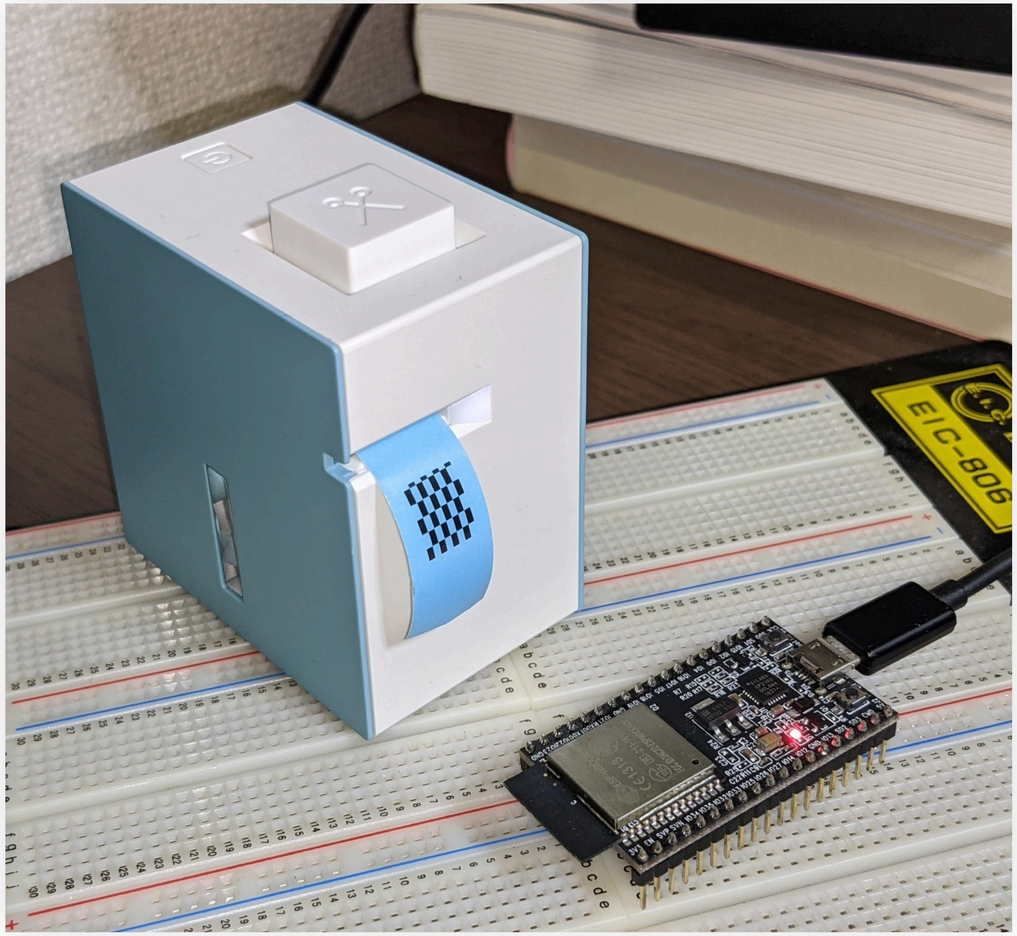
あとはひたすら絵が出るまで試行錯誤

- 印刷データの前後のコミュニケーションも通信を真似て実装
- 一定よりライン数（画像データの量）が少ないと印刷がされないなどの仕様を見つけて都度対応
- 印刷濃度を変える方法もこの過程で発見



印刷できた！

- 1日ほど頑張って印刷することに成功



6. パソコンから手軽に印刷できる API を生やす

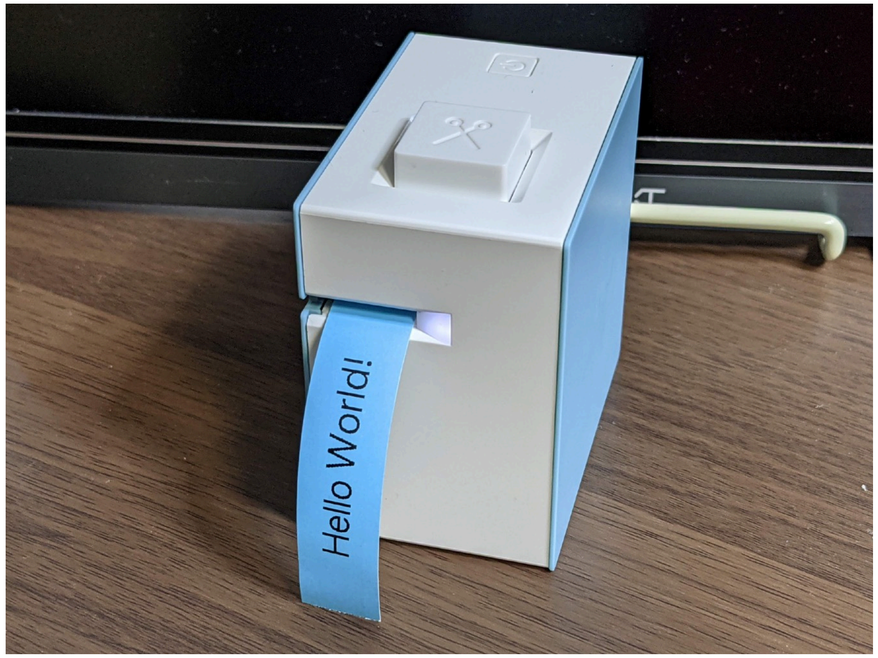
REST API 生やしました

```
 example $ cat hello.sh

#!/bin/sh

set -ue

curl \
  -X POST \
  -H "Content-Type: application/octet-stream" \
  --data-binary @hello.bin \
  ${ESP_ADDRESS}/prints
 example $ ESP_ADDRESS=192.168.39.184 ./hello.sh
```



手元で生成した画像を zlib 圧縮して /prints に POST すると印刷できる

お手軽に印刷できる CLI を作りました

```
 client $ python -m tepracli -a ${ESP_ADDRESS} print --help
Usage: python -m tepracli print [OPTIONS]

Options:
  --preview               Generate preview.png without printing.
  -f, --font TEXT         Path or name of font.
  -S, --fontsize INTEGER  Font size. [px]
  -d, --depth TEXT        Depth of color. Default=0, Min=-3, Max=3
  -m, --message TEXT      Print a text.
  -s, --space TEXT        Leave space between parts. [px]
  -q, --qr TEXT           Draw a QR code.
  --help                  Show this message and exit.
 client $ python -m tepracli -a ${ESP_ADDRESS} print -q "https://example.com" -s 10 -m "#wakate2021"
```

Pillow でいい感じに文字や QR コードを生成して印刷してくれるやつ

7. 完成！！！！！！！

すべて GitHub に上げてます

https://github.com/puhitaku/tepra-lite-esp32

まとめ

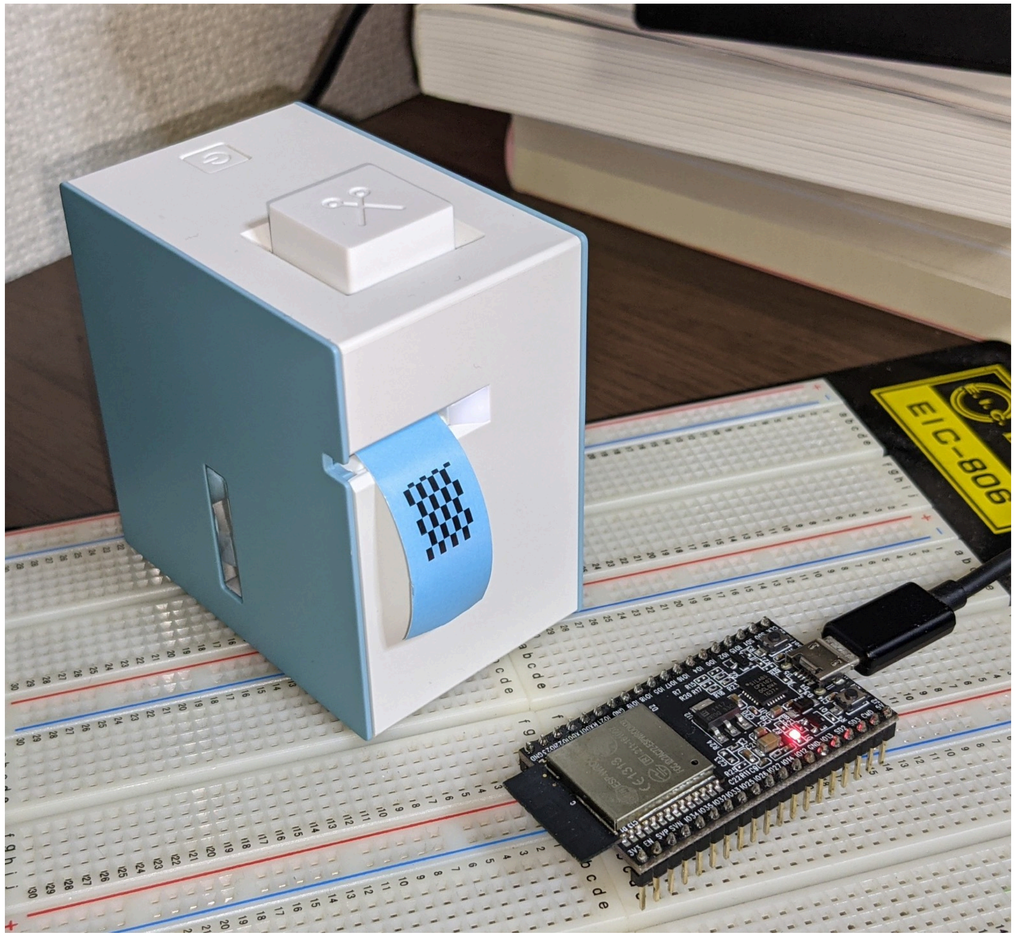
程よい難易度のリバエンを経て
API や CLI まで実装しきれた

複雑な Bluetooth に対する不安は杞憂だった

Sniffer に加えて、平易な API を持つ組み込みの Bluetooth スタックや
MicroPython の対話モードによって泥にハマることなく解析できた

リバースエンジニアリングと「フォワード」エンジニアリングをまとまった形で達成できたのは嬉しい

今後も他にリバエンしたら面白そうなハードがあればぜひ同様に解析したい



おわり