組込みRustでも
でかい？JSONを扱いたい！
…

2023-06-07
人工衛星の開発現場でLT大会

自己紹介

- 井田　健太
- おしごと：ESP32のファームウェアを書く
  - まえはXilinx FPGAさわってました
- seccamp2022, 2023 RISC-V CPU自作ゼミ講師
- twitter: @ciniml

最近の組込みRust環境

- ESP32系がアツい！
- ESP32
  - Espressifの無線機能 (Wi-Fi + Bluetooth) 付きマイコン
  - 通常の開発環境はC/C++
    - ESP-IDFというTCP/IPスタックや
      RTOS搭載環境が提供されている
- 搭載製品いろいろ
  - 液晶付きモジュールが多い。M5StackとかM5Stackとか

https://shop.m5stack.com/products/m5stack-core2-esp32-iot-development-kit

ESP32の組込みRust環境(1)

- ESP32にはいくつかの系統がある
- ESP32-C3
  - CPUコアが**RISC-V**の低コスト製品
- ESP32, ESP32-S3
  - CPUコアが**Xtensa**の高性能製品
  - デュアルコア構成
- RISC-Vはいいけど、Xtensaって何？🤔
  - Cadence (に買収されたTensilica) のカスタム可能CPUコア
  - プロセッサの構成情報からコンパイラ・デバッガ・シミュレータを生成してくれるシステムで使われているベースプロセッサ

ESP32の組込みRust環境(2)

- Xtensaに(公式)LLVMバックエンドが無かったので
  Espressifが**自前でLLVMとrustc**をビルドして提供
- 導入:インストール用のツール **espup** が用意されており簡単
  - https://github.com/esp-rs/espup
  - espup installを実行するだけ。
- (余談:2~3年前はめっちゃめんどくさかった)

作ってみたもの：Remo monitor on M5Paper

- M5Paperという電子ペーパー付きのESP32ユニットに
  Nature Remo / Remo Eの各種センサー情報を表示
- センサデータはRemo Cloud API経由で取れる

温湿度
電力使用量
送信

Nature
Cloud
(on AWS)

温湿度
電力使用量
取得

https://nature.global/nature-remo/nature-remo-3/
https://nature.global/nature-remo-e/

https://shop.m5stack.com/products/m5paper-esp32-development-kit-v1-1-960x540-4-7-eink-display-235-ppi

Nature Remo Cloud APIの仕様

- Webサイト上でREST APIの仕様を公開
  - https://developer.nature.global/
- https://api.nature.global/1/appliances にアクセスすると家電の情報を含むJSONが降ってくる
  - スマートメーターの電力取得に使う
- https://api.nature.global/1/devices にアクセスするとデバイス情報を含むJSONが降ってくる
  - デバイスが持っている温湿度計の情報取得に使う

JSONのサイズ

- 一般ユーザー : 数kBくらい？
  - まあ普通にparseできそうやね
- Nature社員 : 100kB弱
  - いろんなデバッグ用機器が繋がっててデカいのが返ってくる
  - PCとかで扱う分には全く問題ないが...
- ESP32には520kBしかRAMない...
- TLS接続でも結構メモリを使うので余裕はそんなにない
- 普通にメモリに置いてparseすると死ぬ

JSONのストリーム型パーサー？

- Rust実装探したけどぱっと見使いやすそうなの見つからず
  - みんなserde大好きですね
- 仕方ないので練習がてら実装してみた
  - https://github.com/ciniml/fuga-json-seq-parser/
  - json-seq-parserだと汎用的な名前すぎるので、fugaってつけといた！

JSONのストリーム型パーサーの仕様

- no_std
  - ヒープ使わない
- 極力省メモリ
  - 理屈上はJSONのキーを保持できるバッファがあればいけるはず
- コールバック呼び出し
  - 配列開始・終了
  - マップ開始・終了
  - キーまたは値
  - 参照を渡す(コピーしない)

```
FnMut(JsonNode<'node>) ->
Result<ParserCallbackAction, CallbackError>
```

```
pub enum JsonNode<'a> {
    StartMap,
    EndMap,
    StartArray,
    EndArray,
    Key(JsonScalarValue<'a>),
    Value(JsonScalarValue<'a>),
}
```

JSONのストリーム型パーサーの使い方

```
// バッファ256、スタック10でパーサー作成
let mut parser: Parser<256, 10> = Parser::new();
let mut file =
File::open("data/devices.json").unwrap();
let mut reader =
embedded_io::adapters::FromStd::new(&mut file);
let mut indent_level = 0;
loop {
    let result = parser
        .parse(&mut reader, |node| {
            match node {
                JsonNode::EndMap => indent_level -= 1,
                JsonNode::EndArray => indent_level -=
1,
                _ => {}
            }
            for _ in 0..indent_level {
                print!("  ");
            }
```

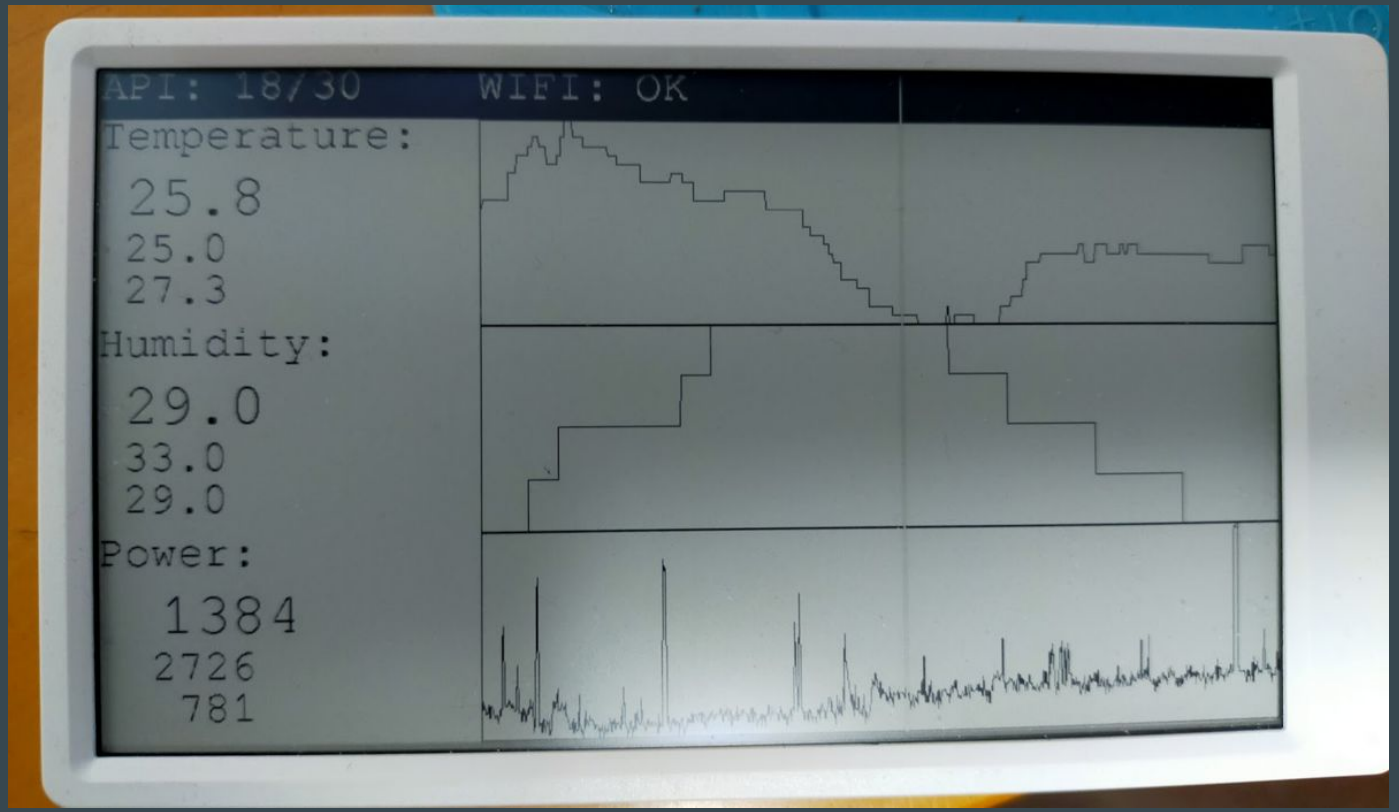
```
            match node {
                JsonNode::StartMap => println!("{{"),
                JsonNode::StartArray => println!("["),
                JsonNode::Key(v) => print!("{}: ", v),
                JsonNode::Value(v) => println!("{},",
v),
                JsonNode::EndMap => println!("}},"),
                JsonNode::EndArray => println!("],"),
            }
            match node {
                JsonNode::StartMap => indent_level +=
1,
                JsonNode::StartArray => indent_level +=
1,
                _ => {}
            }

DefaultParserCallbackResult::Ok(ParserCallbackAction::N
othing)
        });
```

実装したもの

室温
(Remo 3)

湿度
(Remo 3)

瞬時電力
(Remo E Lite)



実装してみた感想

- タグ付き共用体とパターンマッチ便利
  - パーサーのステートマシン書くのがとても楽
  - タプルに対するマッチができるので、複数条件をフラットに書ける
    - 条件を網羅しやすい
- nom便利
  - Rustのパーサーコンビネーターcrate
  - 適当に関数組み合わせたらパースできる
  - 主にJSONの値のパースに使っている

宣伝

- Interface 2023年5月号は組込みRust特集
- M5StampC3を使った記事あり
  - なんかページ数おかしい... (65ページ)
    中林さん書きすぎw
- 組込みRust本もよろしくね！

https://www.c-r.com/book/detail/1403

https://interface.cqpub.co.jp/magazine/202305/

おわり