特集2

第5章 RustからESP-IDFの機能をフルに使えるようにしている仕組み

ステップ3…Rust開発環境の構成を理解する

中林 智之

ESP32-C3+Rustでいろいろ試す前にESP32シリーズのRust開発環境がどのように構築されているか説明します.

具体的なクレート名をまじえつつ，ESP-IDFとstdの関係や，この後作るアプリケーションで使うクレートとESP-IDFの関係を理解していきます．最初はどこに何があるか少し分かりづらいですが，この構成がある程度分かっていると，自分でアプリケーションを作成しようと考えたときに，どのクレートを調べれば何があるのか，分かりやすいかと思います.

今回ESP32-C3でアプリケーションを作成する際のソフトウェアのレイヤ構造は**図1**の通りです．一番下にハードウェア(ESP32)があり，その上にESP-IDFがあります．ここまではRustは関係ありません.

ビルド用のソースコードはGitHubの`play-std`ディレクトリ下にあります.

```
https://github.com/tomoyuki-nakabayashi/interface202305-c3-std-rust
```

本誌に載せきれなかった**リストA**～**リストB**は本誌ウェブ・ページに掲載しています.

```
https://interface.cqpub.co.jp/2305rust2/
```

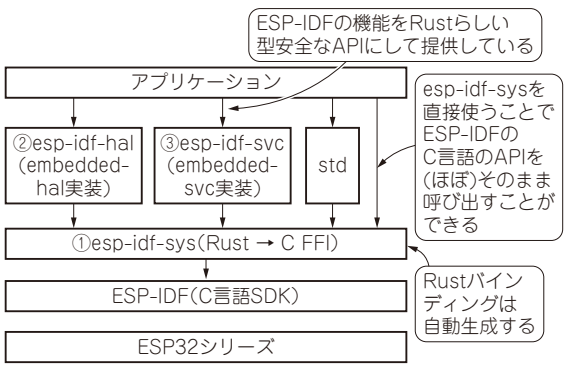

図1　今回のソフトウェア・レイヤ構造

①3つの役割を果たすesp-idf-sys

図1でESP-IDFの上にesp-idf-sysというクレートがあります．これが最も注目すべきクレートで，主に次の3つの役割を果たしています.

● 1，C言語の関数をRustから呼べるようにする

1つはESP-IDFで公開されているC言語の関数をRustから呼べるようにするRust FFI(Foreign Function Interface)バインディングの提供です.

以後，このようにRustからC言語の関数を呼び出せるようにしたものを単にRustバインディング(Rust bindings)と呼びます.

esp-idf-sysより上のレイヤにあるクレートはesp-idf-sysのRustバインディングを経由して，ESP-IDFの機能を使います.

● 2，ESP-IDFをビルドしてRustアプリケーションを構築可能にする

2つ目の役割は，アプリケーションをビルドするときにESP-IDFをビルドして，Rustアプリケーションを構築可能にすることです．主にESP-IDFのビルドとRustバインディングの生成を行います.

先ほど，esp-idf-sysはESP-IDFのRustバインディングを提供している，と書きましたが，esp-idf-sysのGitHubリポジトリ(4)を見るとそのようなコードは一切置かれていません.

ではどうしているのかと言うと，ESP-IDFをビルドした後に，変換ツールのbindgen(5)を使ってC言語のヘッダ・ファイルからRustバインディングを自動生成しているのです.

例えば，C言語の`malloc`関数に対するRustバインディングは**リスト1**のようになります．esp-idf-sysの`build`ディレクトリにはesp-idf-sysクレートのビルド・スクリプトが置かれています．中を少し見てみると分かりますが，それなりに複雑なビルドを行っています．これは，esp-idf-sysクレートの重要な