Rust for embedded devices

Michael Yuan, PhD

Two choices

Raspberry Pi

- Mainstream ARM CPU and architure
- Linux-based system
- Standard Rust toolchain
- “Expensive” $15 for Zero W
- Large “development” PCB board
- Power / resource intensive

ESP32

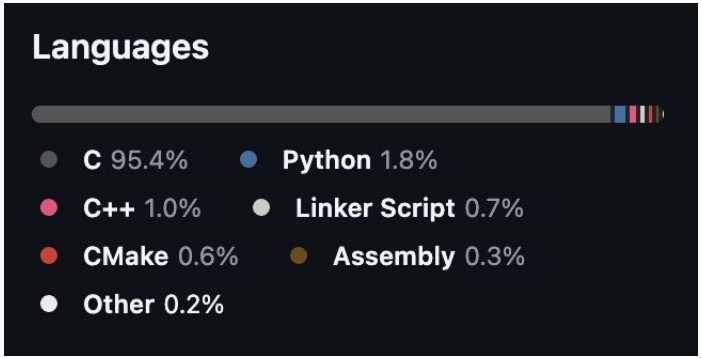
- RISC-based CPU and RF SoC
- RTOS-based “embedded” system
- Customized Rust toolchain
- Very cheap $2 - $5
- Single chip solution
- Very lightweight

ESP32 variants

- S3
  - Dual-core XTensa LX7 MCU (CPU), 240MHz
  - 512KB internal RAM
  - Support 16MB of external RAM
  - WiFi, Bluetooth, USB
- C3
  - RISC-V MCU (CPU), 120MHz
  - 400KB internal RAM
  - Support 16MB of external RAM
  - WiFi, Bluetooth, USB

ESP-IDF

- Developer tool to compile & build ESP32 firmware
  - RTOS + your application
  - Use your Linux, Windows and Mac to compile and build
- Source code: https://github.com/espressif/esp-idf
- Getting started: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html



Rust interface for embedded devices

A Hardware Abstraction Layer (HAL) for embedded systems

- It defines a set of Rust functions (Traits) to access device hardware features
- Device makers provide implementations for those Traits
- Source code: https://github.com/rust-embedded/embedded-hal
- Devices that implements the HAL Traits: https://github.com/rust-embedded/awesome-embedded-rust#driver-crates

Rust for ESP32

This is the Rust SDK you should use: https://github.com/esp-rs/esp-idf-svc

- The raw Rust wrapper for ESP-IDF: https://github.com/esp-rs/esp-idf-sys
- The HAL implementation: https://github.com/esp-rs/esp-idf-hal

Key features of ESP32 Rust SDK

- You can use `std` Rust Standard Library, which makes it a lot easier to develop your apps in Rust
- The `esp-idf-svc` crate provides a “pure Rust” API
- The compiled target is ESP32 firmware that contains both the OS (RTOS) and the app – similar to unikernel apps
- It requires additional compiler forks and Rust targets to support XTensa and RISC-V CPUs

Install Rust and the Cargo toolchain

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

See: https://www.rust-lang.org/tools/install

Install dependencies

See: https://docs.espressif.com/projects/rust/book/installation/std-requirements.html

Linux:

```
sudo apt-get install git wget flex bison gperf python3 python3-pip
python3-venv cmake ninja-build ccache libffi-dev libssl-dev dfu-util
libusb-1.0-0
```

Install Rust toolchain

```
cargo install espup --locked

espup install

. $HOME/export-esp.sh
```

It installs

- Espressif Rust fork with support for Espressif targets
- nightly toolchain with support for RISC-V targets
- LLVM fork with support for Xtensa targets
- GCC toolchain that links the final binary

Get ready and have fun!