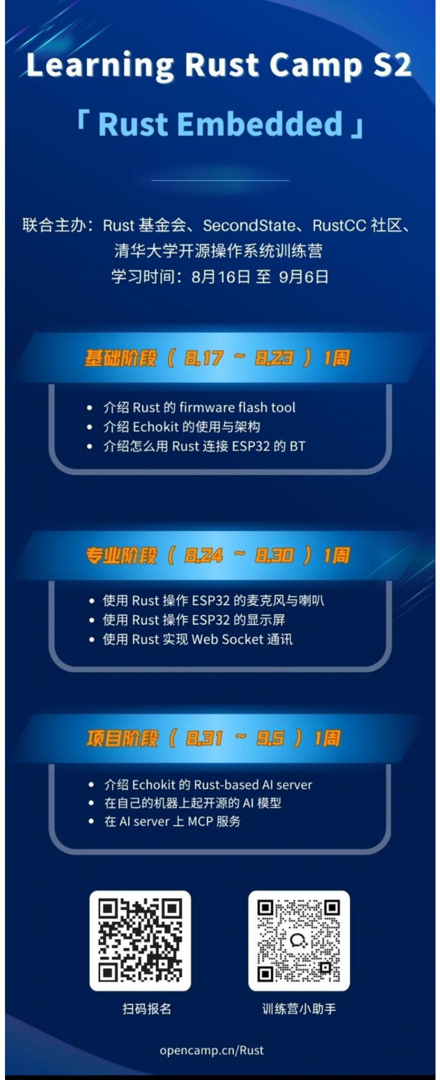
Rust for embedded devices

Michael Yuan, PhD

Introduction:
https://opencamp.cn/Rust/camp/S02

Sign up here:
https://opencamp.cn/Rust/camp/S02/register?code=cHsXpIq2vGdaM



Learning Rust Camp S2

「Rust Embedded」

联合主办：Rust 基金会、SecondState、RustCC 社区、
清华大学开源操作系统训练营
学习时间：8月16日 至 9月6日

基础阶段（8.17 ~ 8.23）1周

- 介绍 Rust 的 firmware flash tool
- 介绍 Echokit 的使用与架构
- 介绍怎么用 Rust 连接 ESP32 的 BT

专业阶段（8.24 ~ 8.30）1周

- 使用 Rust 操作 ESP32 的麦克风与喇叭
- 使用 Rust 操作 ESP32 的显示屏
- 使用 Rust 实现 Web Socket 通讯

项目阶段（8.31 ~ 9.5）1周

- 介绍 Echokit 的 Rust-based AI server
- 在自己的机器上起开源的 AI 模型
- 在 AI server 上 MCP 服务

扫码报名　训练营小助手

opencamp.cn/Rust

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

The EchoKit device

An ESP32-S3 SoC + audio processor + microphone + speaker + buttons + USB

You can use it for FREE with an RMB 168 deposit

https://opencamp.ai/Rust/bbs/2

Echokit
08/04 16:37:59

嵌入式Rust训练营专用设备
EchoKit

🎓【训练营简介】嵌入式 Rust 训练营是一门面向初学者的项目制学习课程，涵盖嵌入式...

¥168

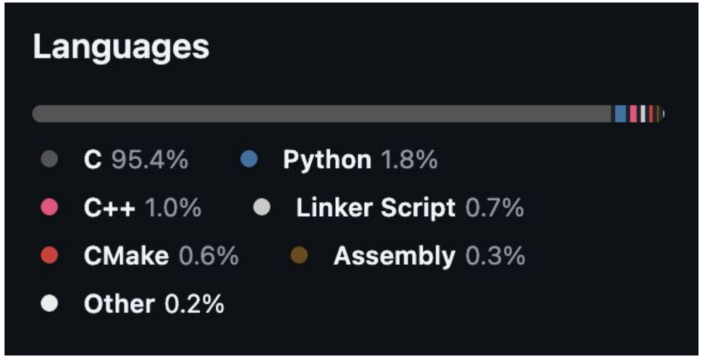
长按识别小程序 跟团购买

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

Contributors 932

+ 918 contributors


Languages

● C 95.4%	● Python 1.8%
● C++ 1.0%	● Linker Script 0.7%
● CMake 0.6%	● Assembly 0.3%
● Other 0.2%	

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!