Prototyping and Product Development with Rust on ESP32

2025-07-16

Rust on Embedded - Elektor Academy Pro Conference

Juraj Michálek
Espressif Systems

Elektor Mag

Special Guest Edition

https://www.elektor.com/products/elektor-special-espressif-guest-edition-2023-pdf-en

Topics for today

Open source - GH org espressif & esp-rs

Bootstrapping Rust no_std

Comparison of Rust std vs. no_std

Slint UI, Embedded Graphics, Bevy ECS

rs-matter, rainmaker-rs

Integration of Rust and OSes including MicroPython

Tips and tricks

Espressif Systems

Who We Are

As a public, multinational, fabless semiconductor company (Espressif Systems, SSE: 688018.SH), we develop cutting-edge, low-power wireless communication chipsets. Our AIoT solutions are green, versatile, and cost-effective. We have a closed-loop development cycle for core technologies, including Wi-Fi & Bluetooth LE & IEEE 802.15.4 protocols, RF, RISC-V MCUs, AI algorithms, operating systems, toolchains, AIoT frameworks, and Cloud services.

1 Billion+
Global IoT SoC Shipments

Global Leader
In Wi-Fi MCU Market

200+ IPs
In AIoT Technology

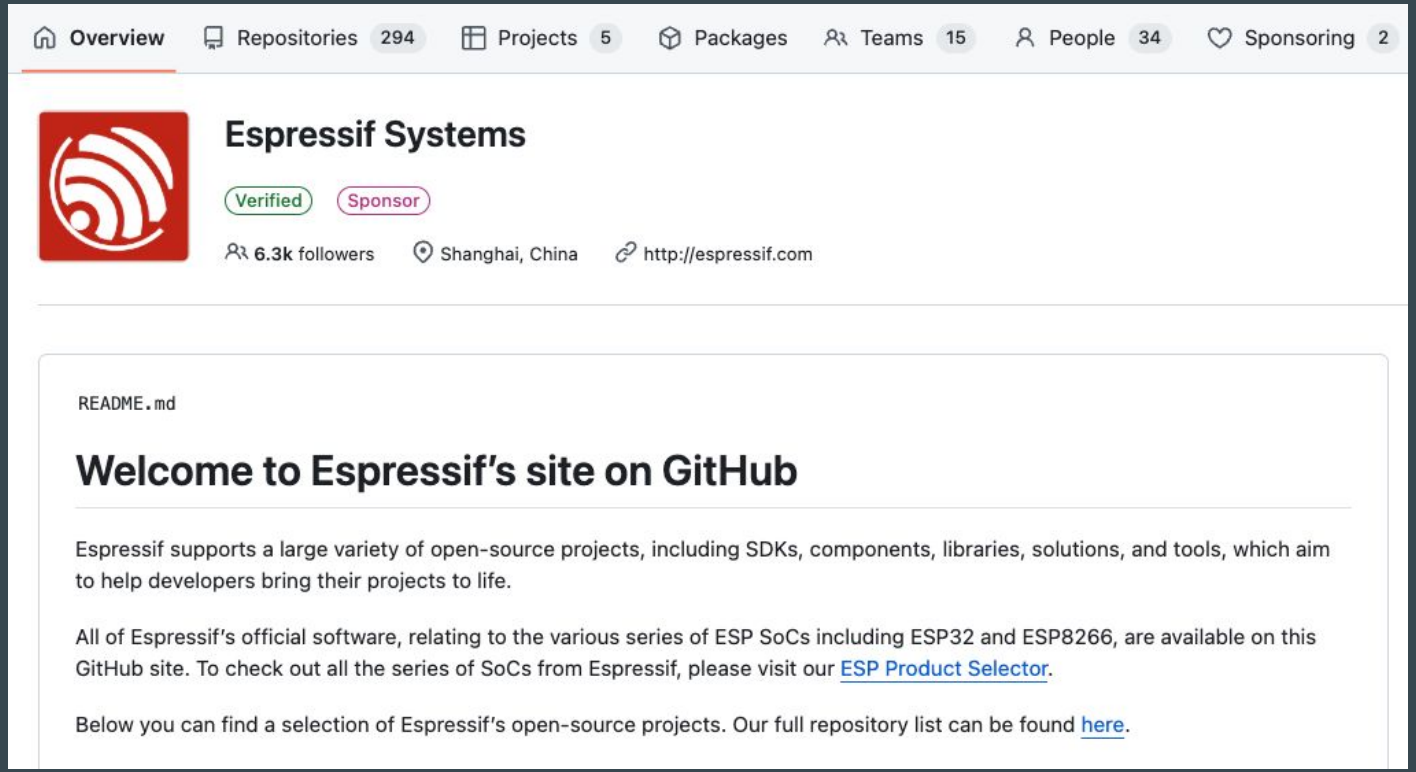
10,000+
Satisfied Customers Worldwide

2 Million+
Active Ecosystem Followers and Developers

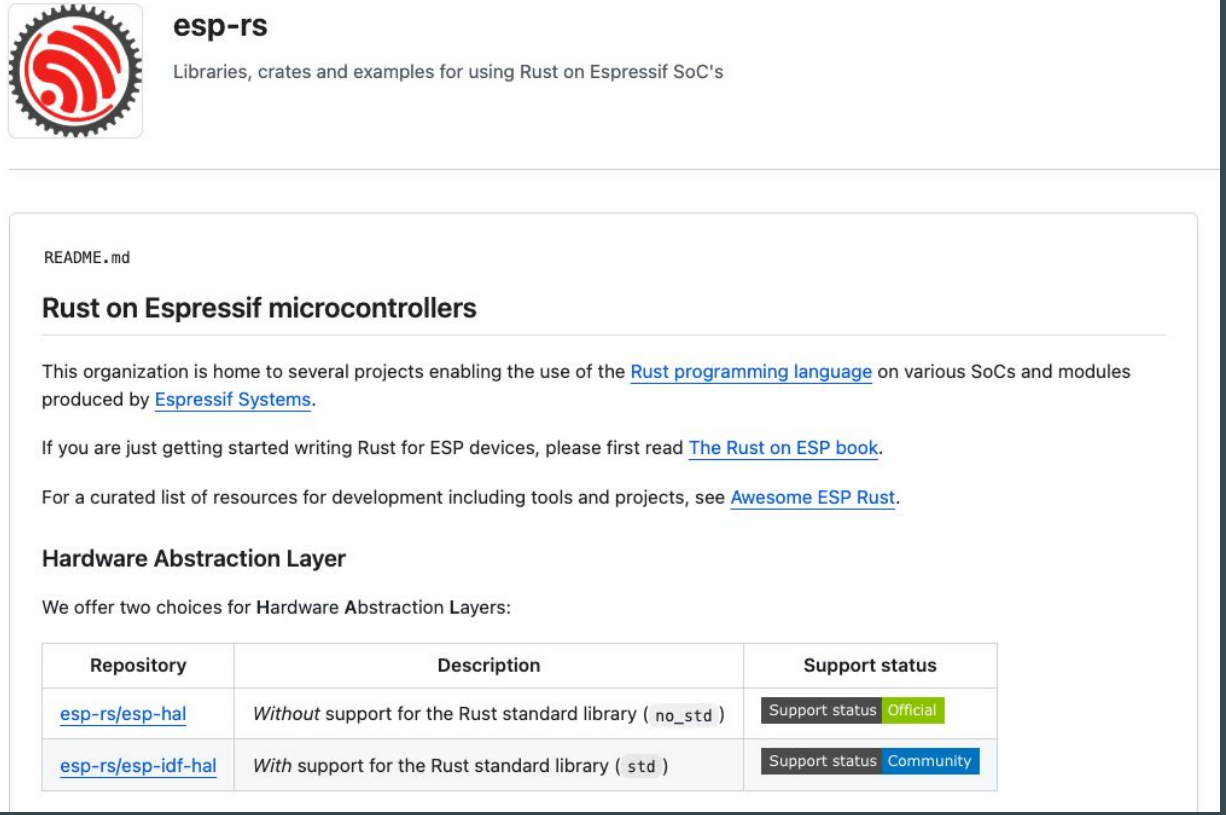
Supports:

- Open Source
- Open Hardware

Open Source


Overview
Repositories 294
Projects 5
Packages
Teams 15
People 34
Sponsoring 2
Espressif Systems
Verified
Sponsor
6.3k followers
Shanghai, China
http://espressif.com
README.md
Welcome to Espressif's site on GitHub
Espressif supports a large variety of open-source projects, including SDKs, components, libraries, solutions, and tools, which aim to help developers bring their projects to life.
All of Espressif's official software, relating to the various series of ESP SoCs including ESP32 and ESP8266, are available on this GitHub site. To check out all the series of SoCs from Espressif, please visit our ESP Product Selector.
Below you can find a selection of Espressif's open-source projects. Our full repository list can be found here.

Open Source - esp-rs



esp-rs

Libraries, crates and examples for using Rust on Espressif SoC's

README.md

Rust on Espressif microcontrollers

This organization is home to several projects enabling the use of the Rust programming language on various SoCs and modules produced by Espressif Systems.

If you are just getting started writing Rust for ESP devices, please first read The Rust on ESP book.

For a curated list of resources for development including tools and projects, see Awesome ESP Rust.

Hardware Abstraction Layer

We offer two choices for **H**ardware **A**bstraction **L**ayers:

Repository	Description	Support status
esp-rs/esp-hal	*Without* support for the Rust standard library (`no_std`)	Support status Official
esp-rs/esp-idf-hal	*With* support for the Rust standard library (`std`)	Support status Community

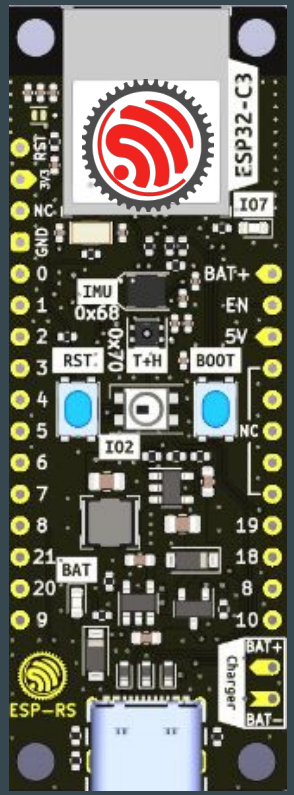
Open Hardware - esp-rust-board

KiCad templates

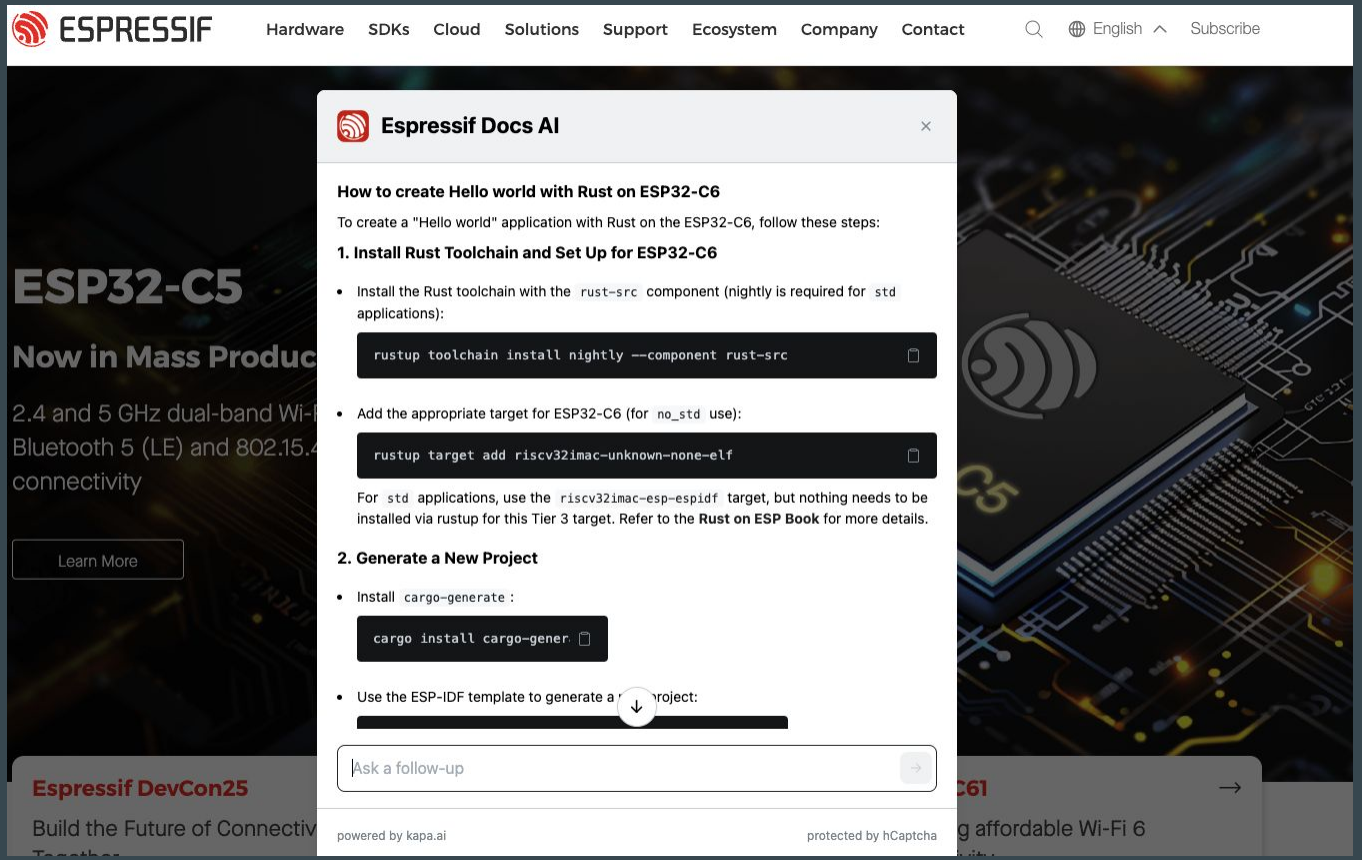
https://github.com/esp-rs/esp-rust-board

ESP32-C3-DevKit-RUST-1

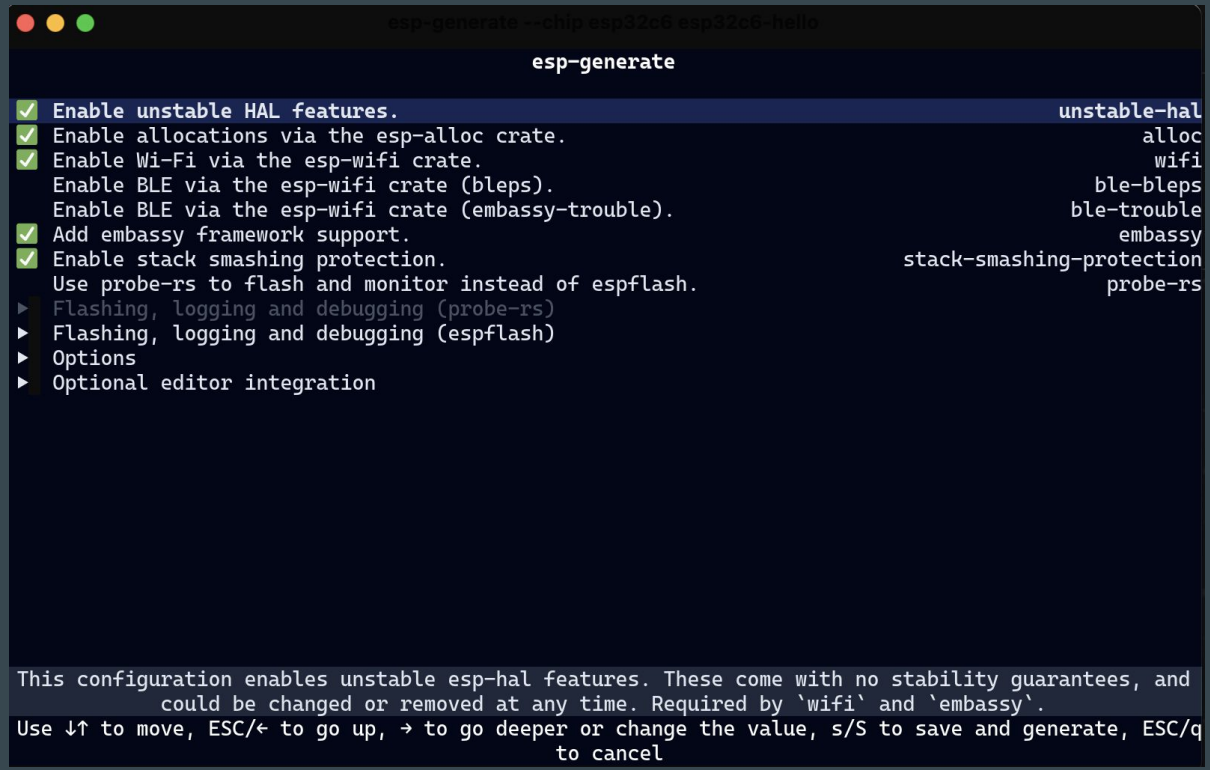
https://www.espressif.com/en/products/devkits



Quick start with AI



esp-generate --chip esp32c6

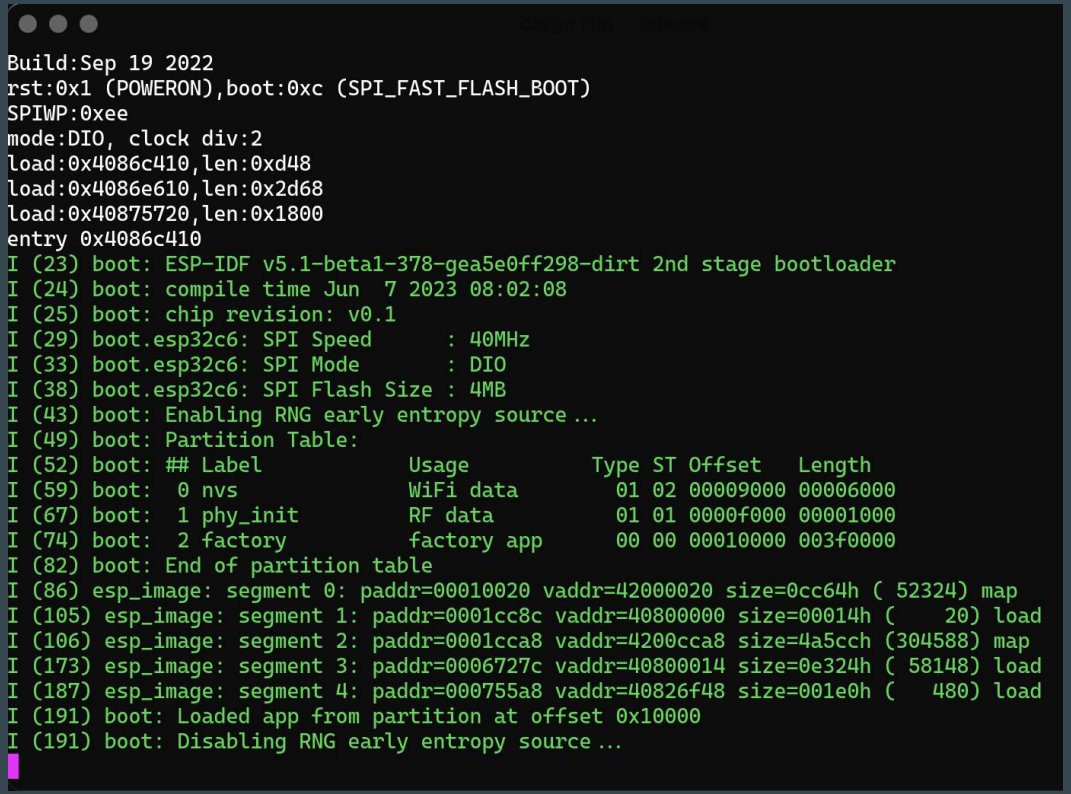


esp-generate - more options

Build, flash, monitor

cargo run --release

shorter: cargo r -r



Simulation: Update wokwi.toml from debug to release

```
[wokwi]

version = 1

gdbServerPort = 3333

#elf = "target/riscv32imac-unknown-none-elf/release/esp32c6-hello"

firmware = "target/riscv32imac-unknown-none-elf/release/esp32c6-hello"
```

wokwi-cli

https://docs.wokwi.com/wokwi-ci/cli-installation

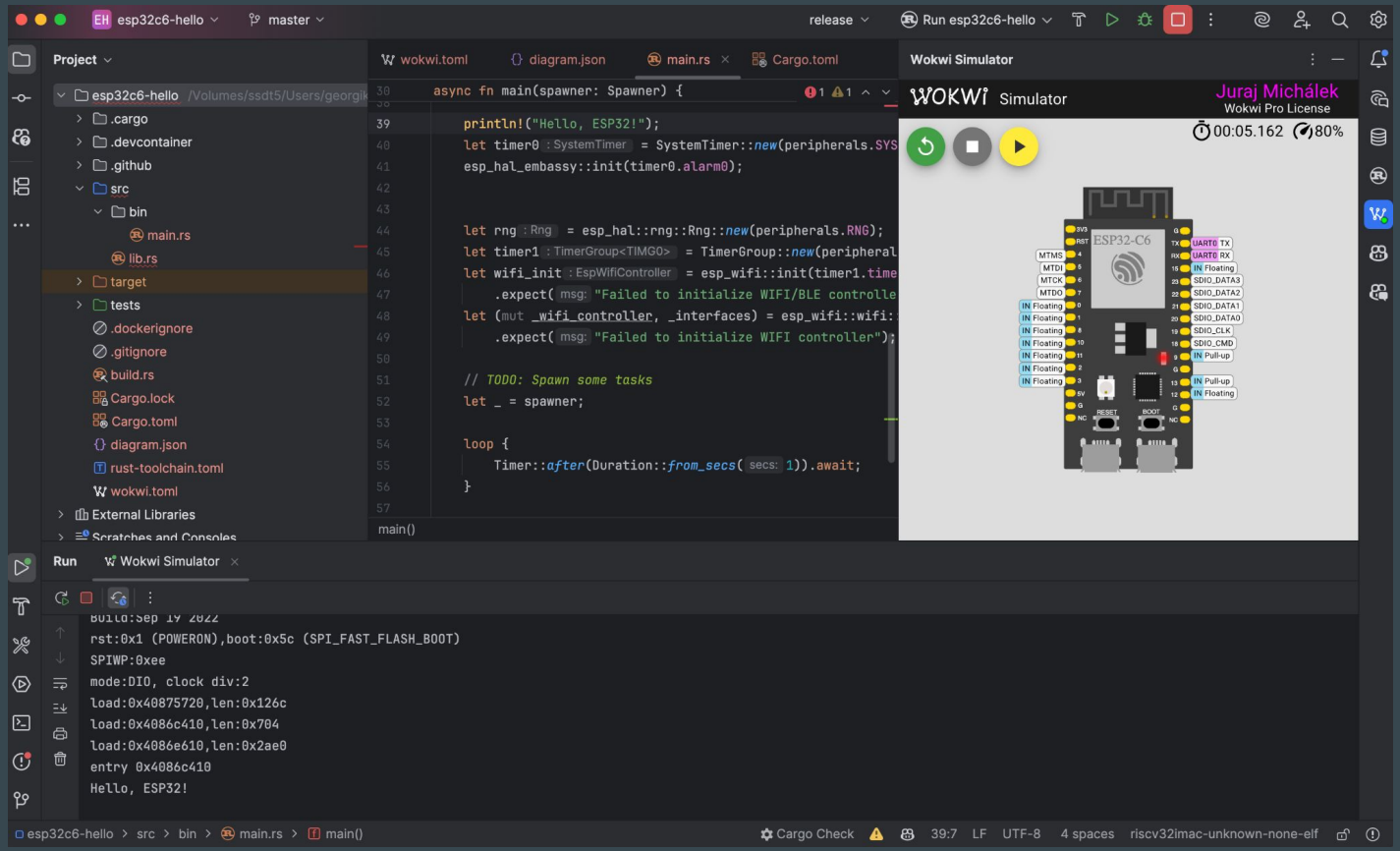
- curl -L https://wokwi.com/ci/install.sh | sh
- iwr https://wokwi.com/ci/install.ps1 -useb | iex

Run simulation: wokwi-cli

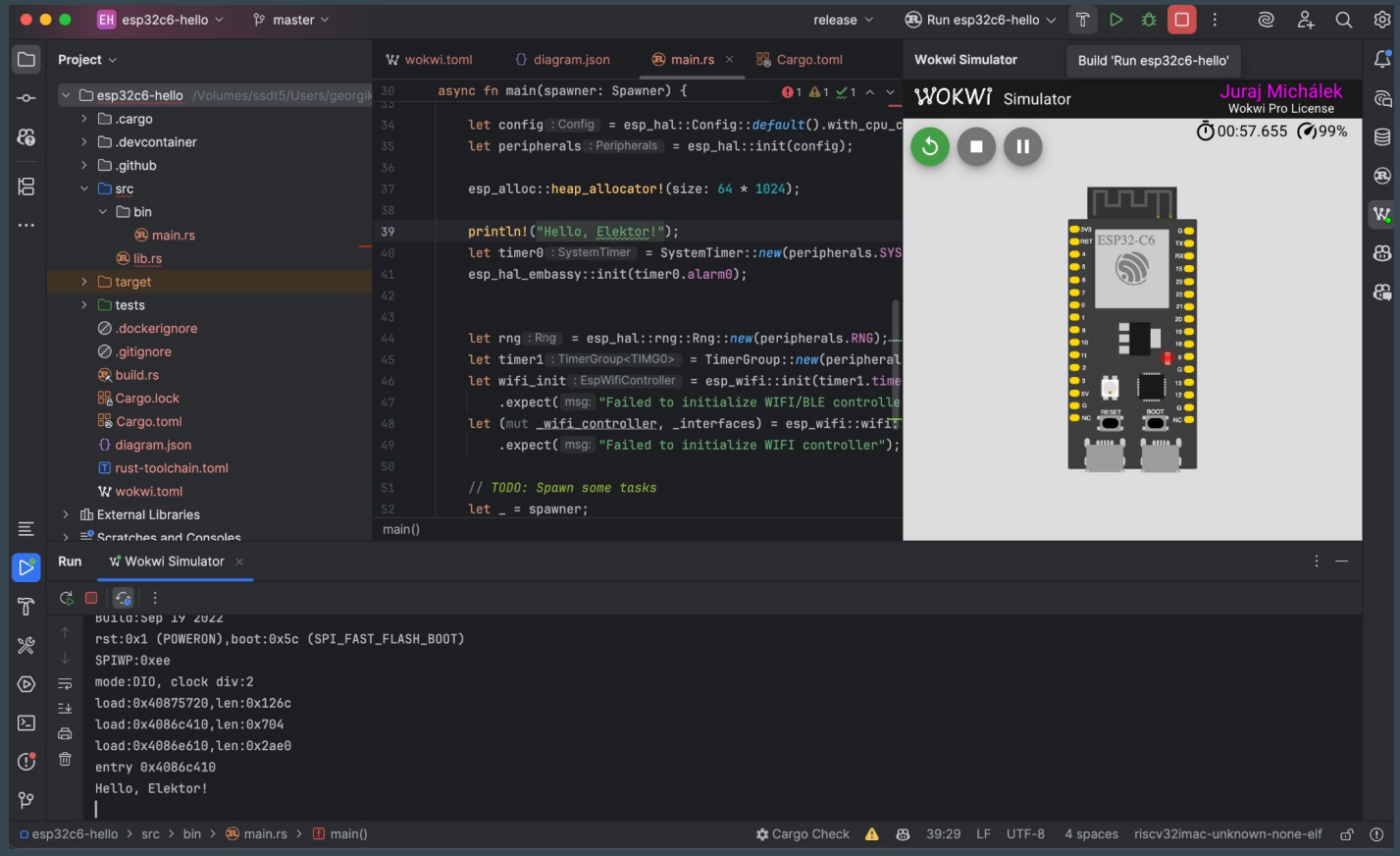
Setup inside existing project: wokwi-cli init

CI suppor: https://docs.wokwi.com/wokwi-ci/getting-started

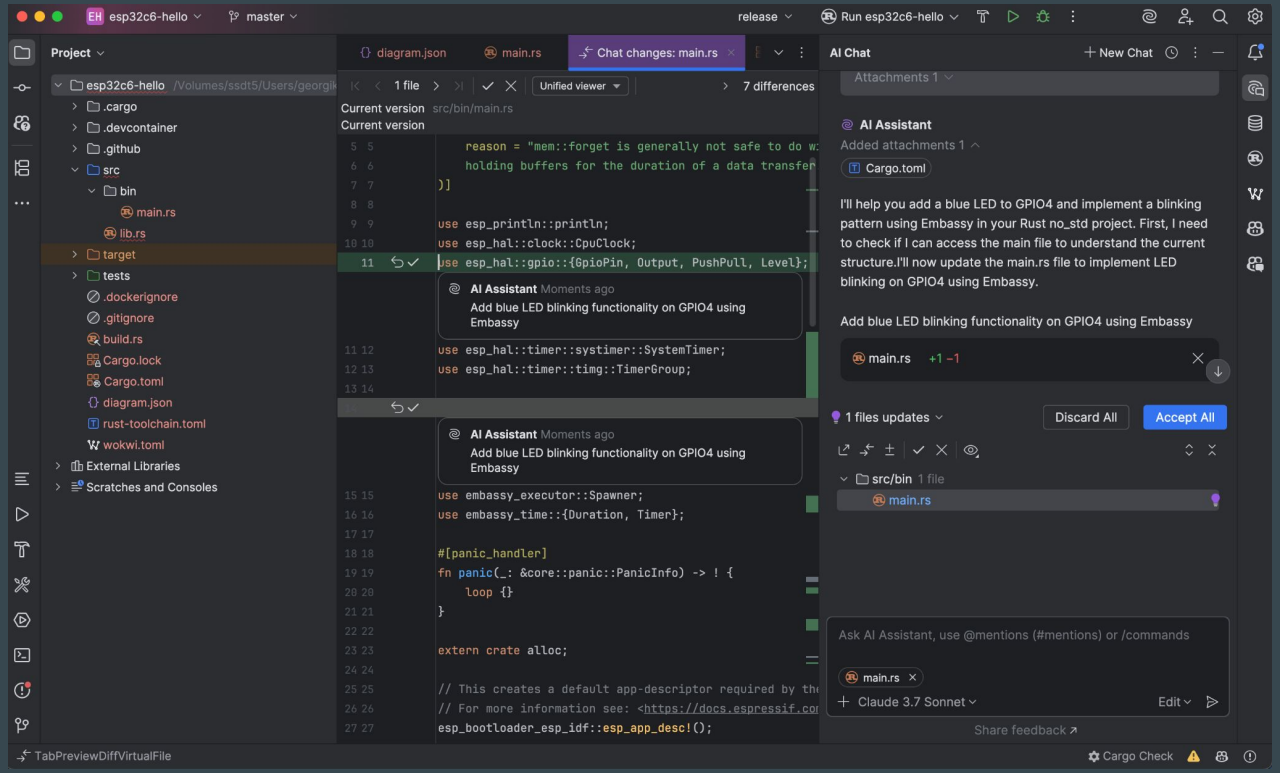
CLion with Rust plugin or RustRover



Auto-reload on build



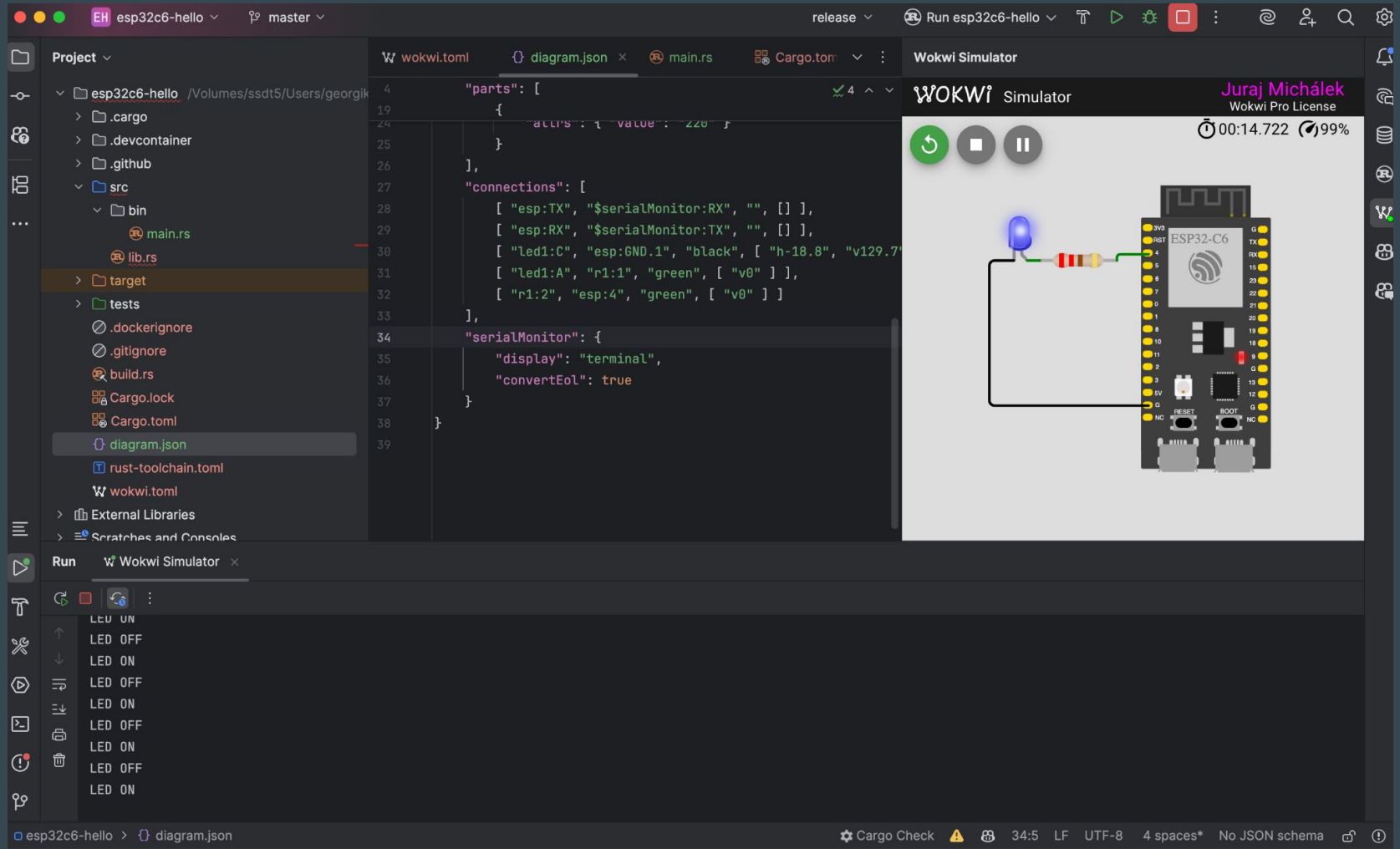
Extending with AI



Caveat

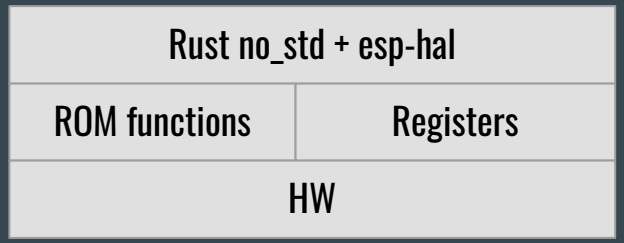
Older models tends to use older esp-hal.

Requires iteration and prompt correction.

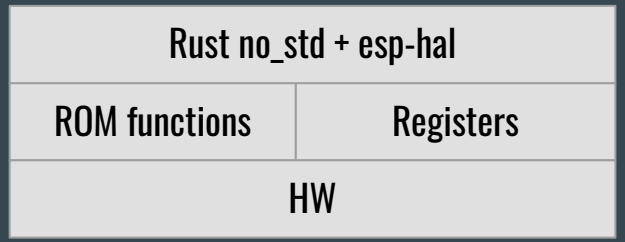

esp32c6-hello
master
release
Run esp32c6-hello
Project
esp32c6-hello /Volumes/ssdt5/Users/georgik
.cargo
.devcontainer
.github
src
bin
main.rs
lib.rs
target
tests
.dockerignore
.gitignore
build.rs
Cargo.lock
Cargo.toml
diagram.json
rust-toolchain.toml
wokwi.toml
External Libraries
Scratches and Consoles
wokwi.toml
diagram.json
main.rs
Cargo.tom
"parts": [
{
}
],
"connections": [
["esp:TX", "$serialMonitor:RX", "", []],
["esp:RX", "$serialMonitor:TX", "", []],
["led1:C", "esp:GND.1", "black", ["h-18.8", "v129.7
["led1:A", "r1:1", "green", ["v0"]],
["r1:2", "esp:4", "green", ["v0"]]
],
"serialMonitor": {
"display": "terminal",
"convertEol": true
}
}
Wokwi Simulator
WOKWi Simulator
Juraj Michálek
Wokwi Pro License
00:14.722
99%
ESP32-C6
RESET
BOOT
Run
Wokwi Simulator
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
LED ON
esp32c6-hello
diagram.json
Cargo Check
34:5
LF
UTF-8
4 spaces*
No JSON schema

Deep dive

There are two ways how to write code for MCUs.

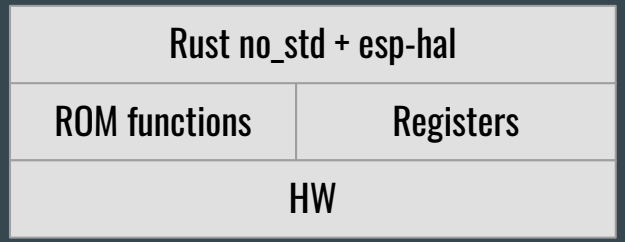

Rust no_std + esp-hal
ROM functions
Registers
HW


C wrapper app
Rust std + esp-idf-hal
ESP-IDF
FreeRTOS
ROM functions
Registers
HW



https://github.com/esp-rs/esp-hal

https://github.com/esp-rs/esp-idf-hal

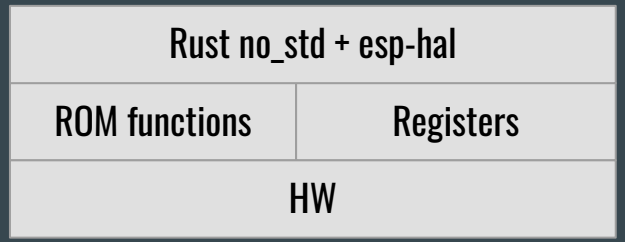


https://github.com/esp-rs/esp-hal

Official Espressif + Community Support

https://github.com/esp-rs/esp-idf-hal

Community support



https://github.com/esp-rs/esp-hal

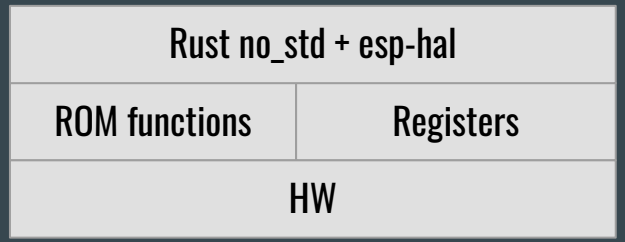
Official Espressif + Community Support

Requires Rust compiler

https://github.com/esp-rs/esp-idf-hal

Community support

Requires Rust compiler + ESP-IDF



https://github.com/esp-rs/esp-hal

Official Espressif + Community Support

Requires Rust compiler

Bleeding edge MCUs and drivers might be missing

https://github.com/esp-rs/esp-idf-hal

Community support

Requires Rust compiler + ESP-IDF

Bleeding edge MCUs and drivers from ESP-DF

Training

Rust no_std

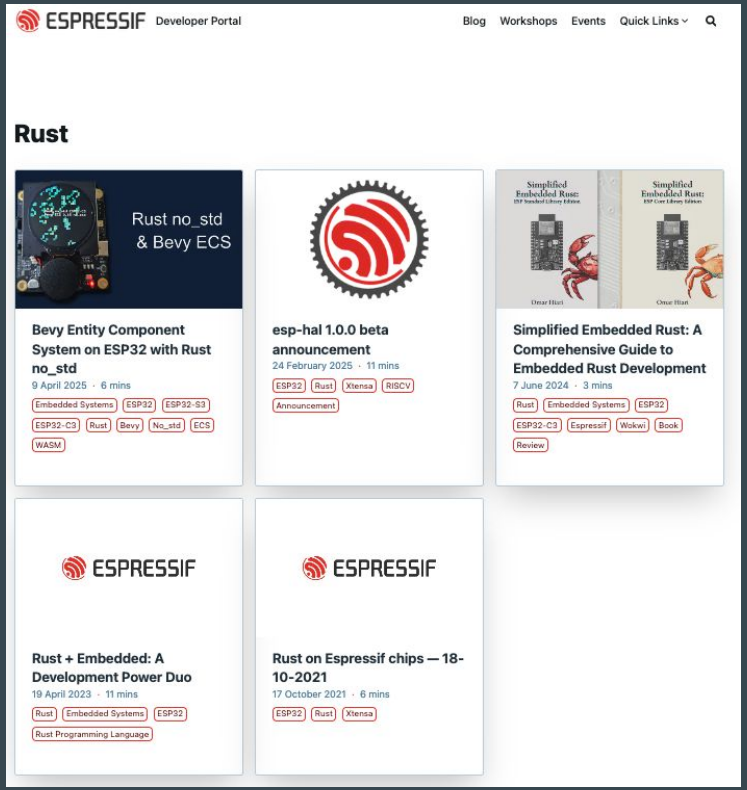
- https://docs.espressif.com/projects/rust/no_std-training/
- https://github.com/esp-rs/no_std-training

Rust std

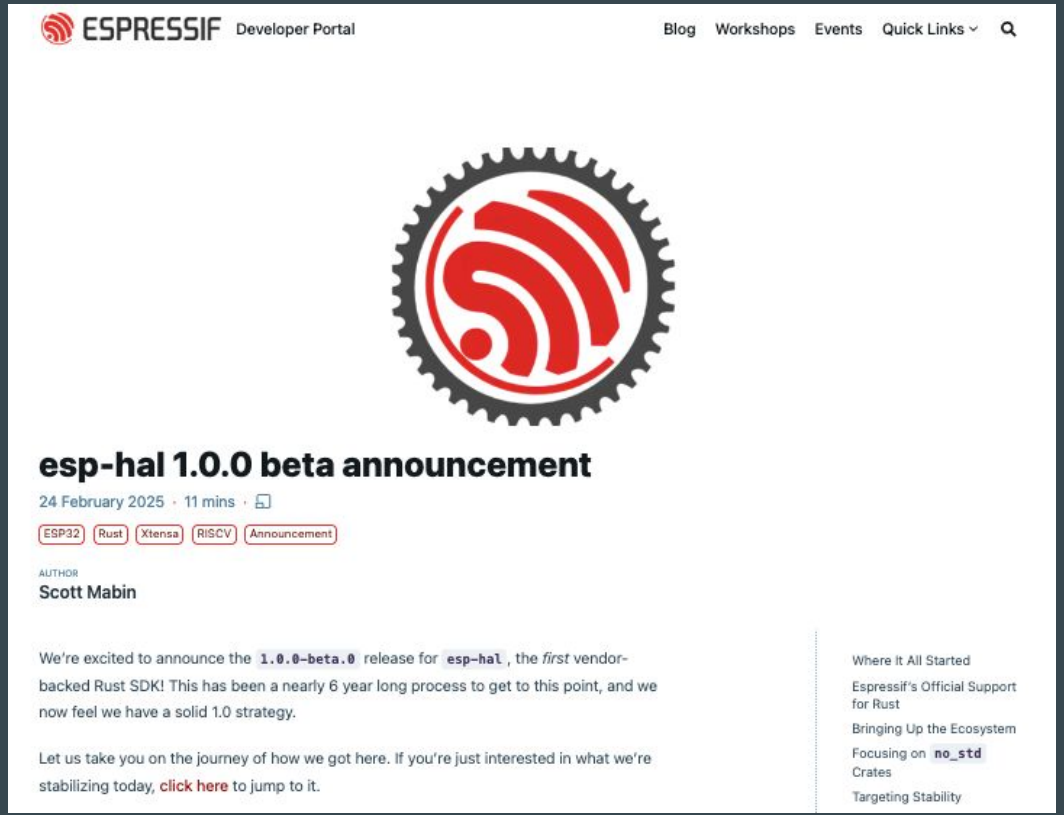
- https://docs.esp-rs.org/std-training/
- https://github.com/esp-rs/std-training
- Created in cooperation with Ferrous Systems

Developer Portal

developer.espressif.com



GitHub: https://github.com/espressif/developer-portal/



https://developer.espressif.com/blog/2025/02/rust-esp-hal-beta/

There are two
architectures
on Espressif MCUs

RISC-V vs Xtensa

RISC-V

- newer
- ESP32-C, ESP32-H, ESP32-P
- upstream Rust compiler is sufficient

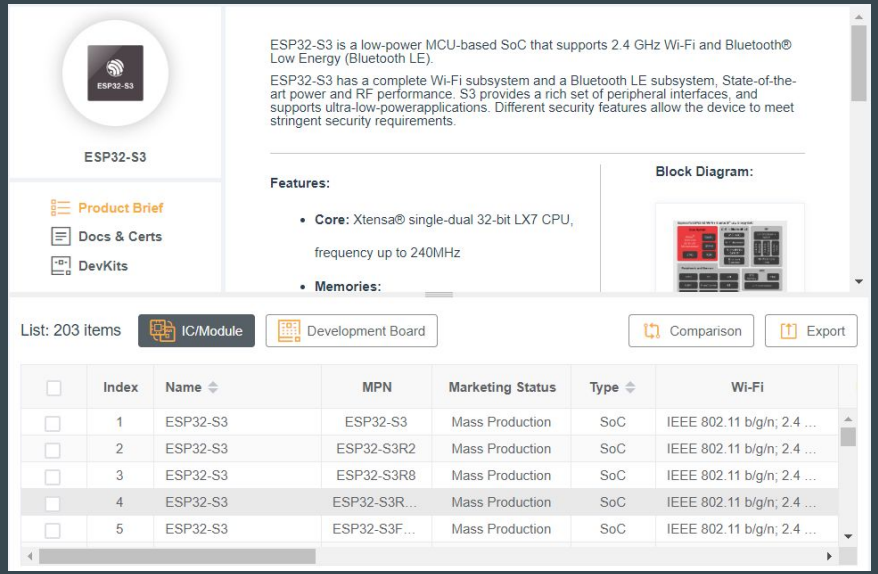
Xtensa

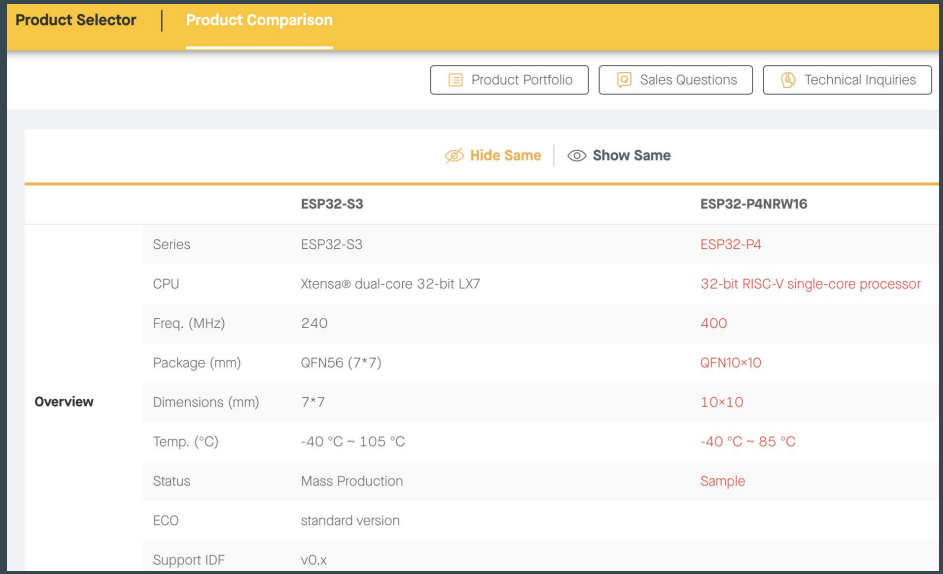
- older
- ESP32, ESP32-S2, ESP32-S3
- requires custom installation
- espup install

https://www.espressif.com/en/products/longevity-commitment

ESP Product Selector + Product Comparison

https://products.espressif.com/





slint
GUI toolkit

Slint

Quick prototyping

Free and commercial version

Rust, C++

Printer MCU demo

Build for no_std:

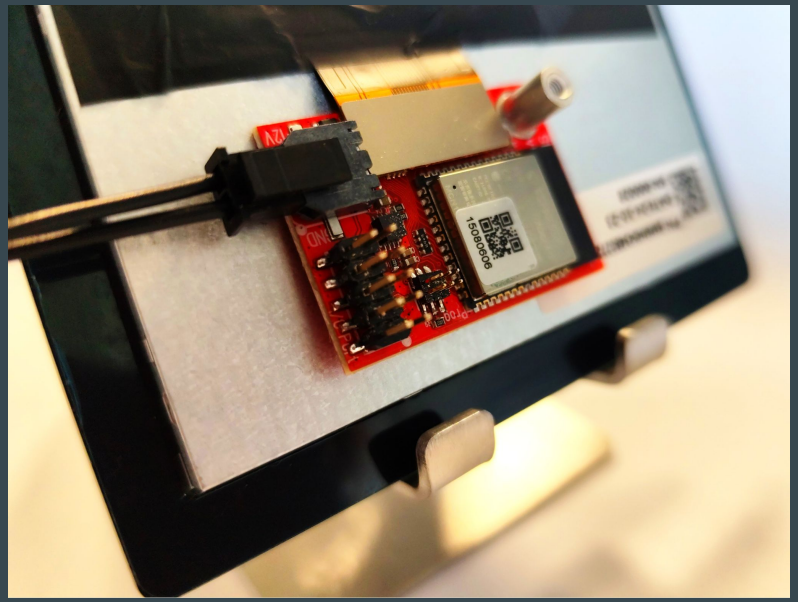
CARGO_PROFILE_RELEASE_OPT_LEVEL=s cargo +esp run -p printerdemo_mcu --target xtensa-esp32s3-none-elf --no-default-features --features=mcu-board-support/esp32-s3-box-3 --release --config examples/mcu-board-support/esp32_s3_box_3/cargo-config.toml

Monitoring: espflash monitor

Slint Workshop

https://github.com/WilstonOreo/slint-esp-workshop/

ESoPe board + RGB Schukat Smartwin display-concept



Will be published at Developer Portal: https://github.com/espressif/developer-portal/pull/495

We welcome feedback.

Example: List of WiFi networks

Rust std

Rust no_std

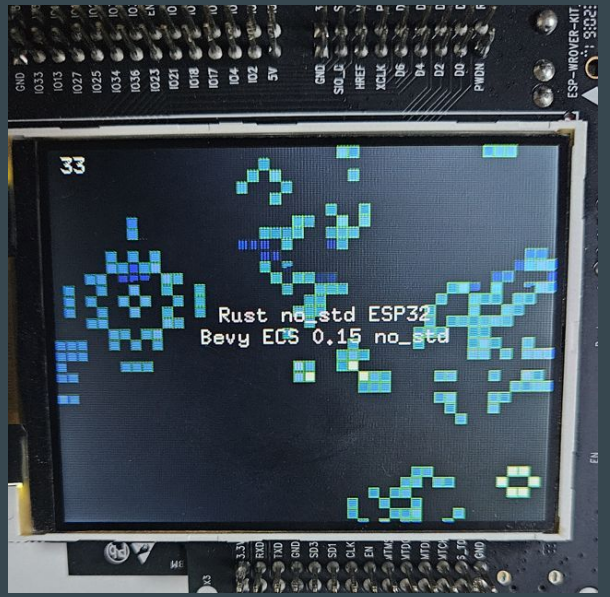
Rust no_std + Embassy

https://github.com/georgik/esp32-list-wifi-networks/

Bevy ECS (Entity Component System)

One of biggest open source Rust projects

- https://bevyengine.org/



Rust no_std + Bevy ECS

- https://developer.espressif.com/blog/2025/04/bevy-ecs-on-esp32-with-rust-no-std/
- https://github.com/georgik/esp32-spooky-maze-game

Rust Bare Metal - Embedded Graphics

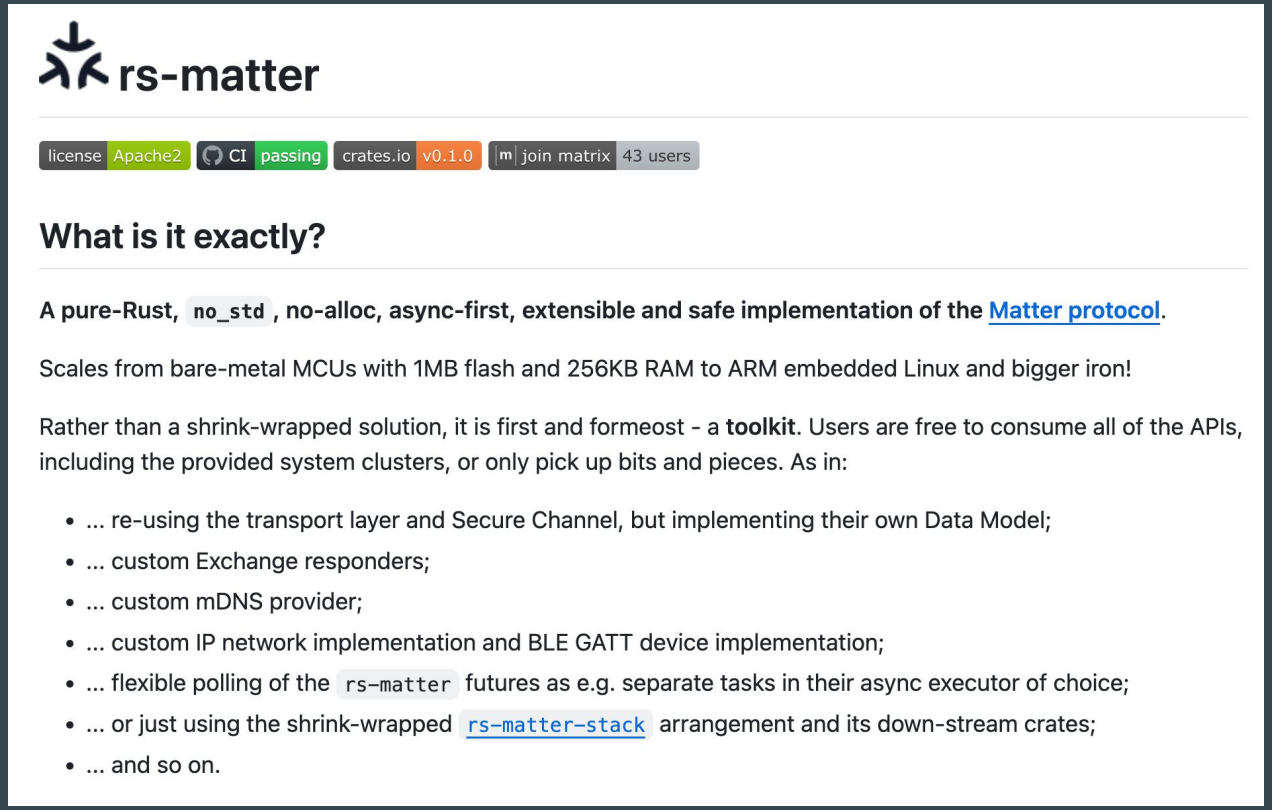
Source: https://github.com/georgik/esp32-spooky-maze-game

Idea: sharing business logic in Rust between multiple targets

Game is using Bevy ECS no_std.

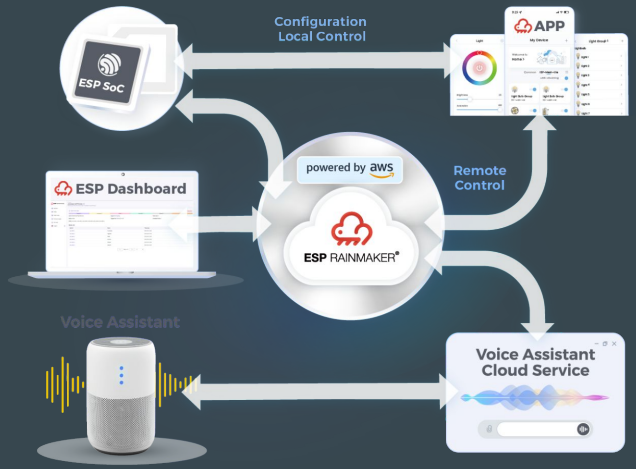
https://github.com/espressif/esp-box

rs-matter



https://github.com/project-chip/rs-matter

rainmaker-rs



Rust Implementation of ESP Rainmaker

A cross-platform implementation of ESP Rainmaker for ESP32 products and Linux using Rust.

- ESP RainMaker is end-to-end IoT development platform which enables development of IoT applications which can be controled remotely.
- However, the C based ESP RainMaker SDK(which can be found here) only supports execution on Espressif's ESP32 SOCs.
- This crate tries to implment similar functionalities for Linux platform along with ESP32(which can further be extended to other microcontrollers).

What is working

- [x] WiFi Provisioning*:
  Providing WiFi for a new device. No need to hardcode WiFi credentials!
- [x] Remote Control:
  Controlling Devices connected to a node over internet using phone application.
- [x] User Node Association:
  Associating a specific node to a user account for control.
- [x] Device sharing:
  Share access to a device with multiple members.
- [x] Contol using Home Assistants:
  RainMaker devices can be added to and controlled using Amazon Alexa / Google Home. More details here

* Currently only supported on ESP32

https://github.com/rainmaker-rs/rainmaker

Exposing Rust to Python REPL

https://github.com/georgik/micropython/commits/experimenta/nmea/

Rust no_std code - as ESP-IDF component (library)

Bridge code - similar like exposing API from C

Sample code:

- https://github.com/georgik/esp32-conways-game-of-life-python/blob/main/rust-no_std-nmea/nmea.py

NMEA altitude: 61.70

Data:

- Latitude: 53°21.6801′ N

- Longitude: -7°29.6628′ W

OSes and integration

Rust no_std - https://github.com/esp-rs/esp-hal - official support

Rust std - https://github.com/esp-rs/esp-idf-hal - community support

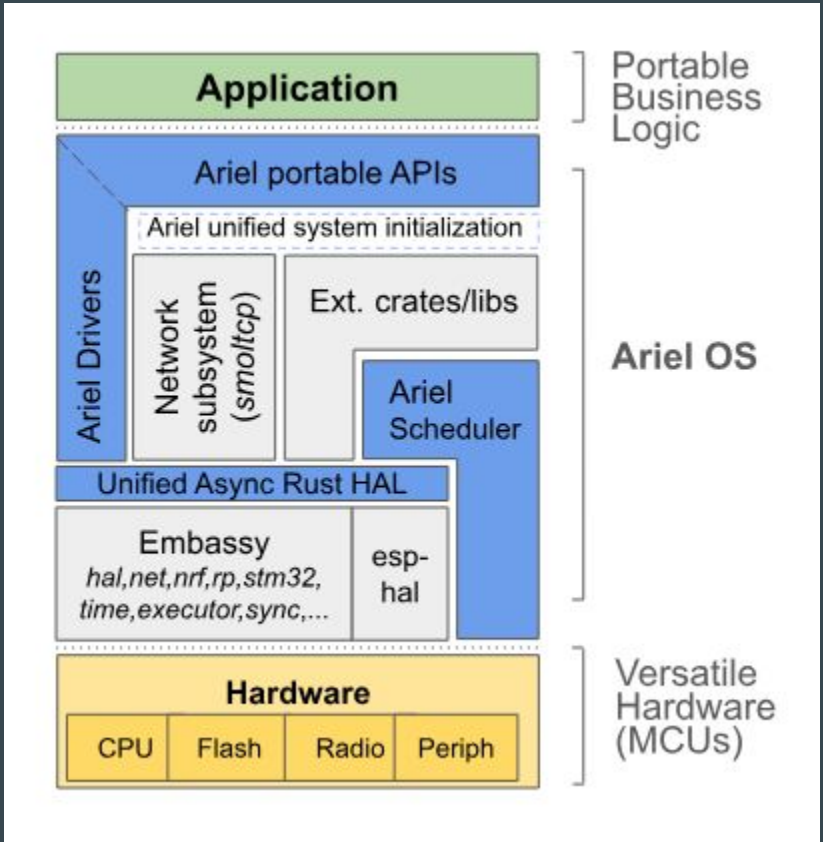
Ariel OS - https://ariel-os.github.io/ariel-os/dev/docs/book/

Zephyr - https://zephyrproject.org/

NuttX - https://nuttx.apache.org/

SVD files: https://github.com/espressif/svd

Ariel OS



https://ariel-os.github.io/ariel-os/dev/docs/book/

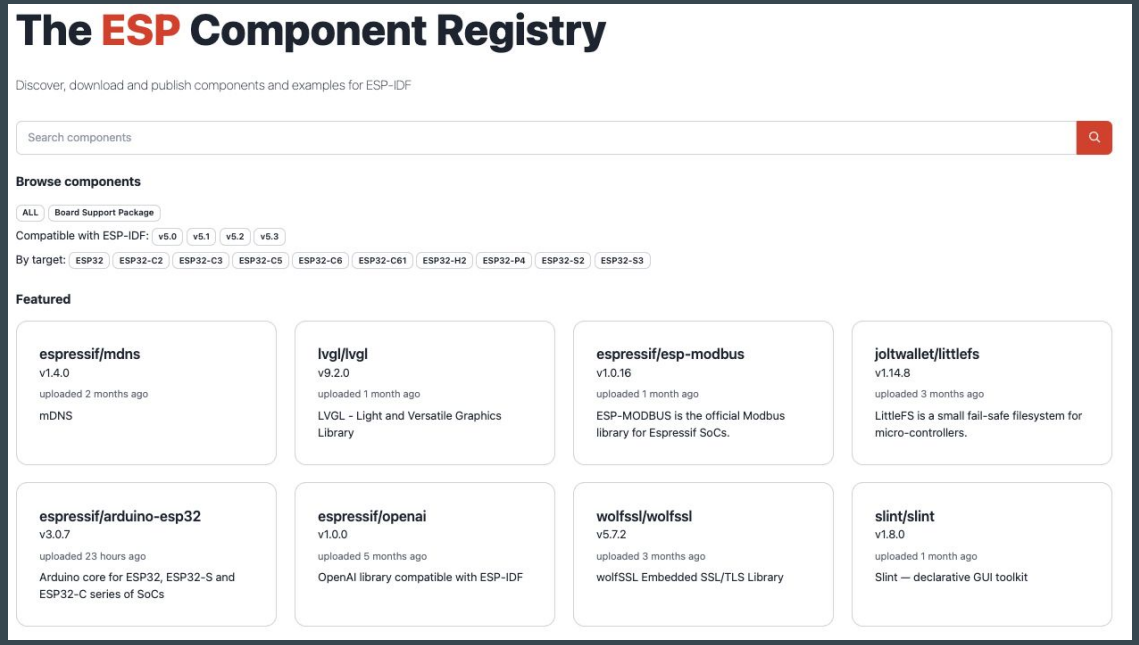
IDE for Rust development

Supported by Espressif:

- VS Code with Espressif Extension - https://developer.espressif.com/tags/vscode/

Supported by JetBrains:

- CLion - https://developer.espressif.com/blog/clion/
- RustRover - https://www.jetbrains.com/rust/



https://components.espressif.com/

Example: https://components.espressif.com/components/espressif/i2c_bus/

Tips

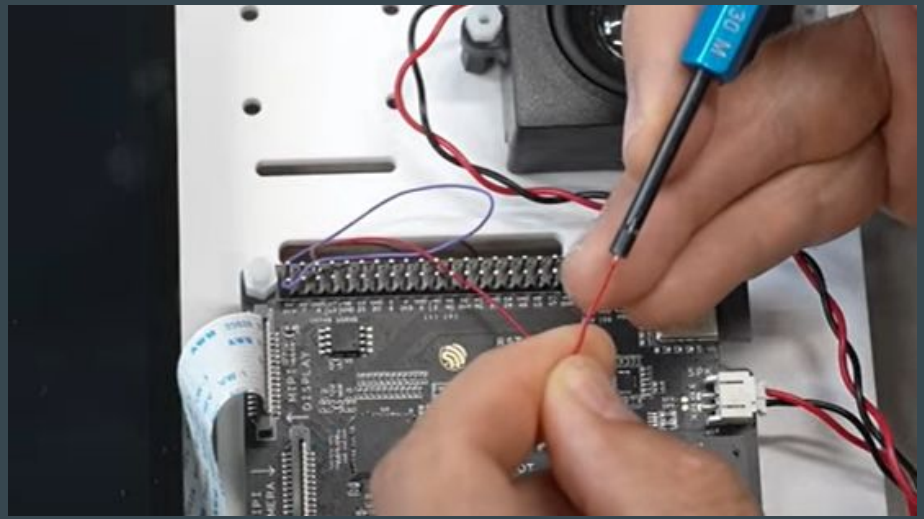
How to “unbrick” ESP32

Many people incorrectly thinks that ESP32 is bricked, in many cases the custom firmware just does not receive UART or USB signals.

Solution: hold BOOT button and press RESET to switch to boot mode

After flashing, just press RESET to return to normal mode.

Wire Wrap Tool



Espressif DevCon 2024 - Tips and Tricks

Architecture

Layers:

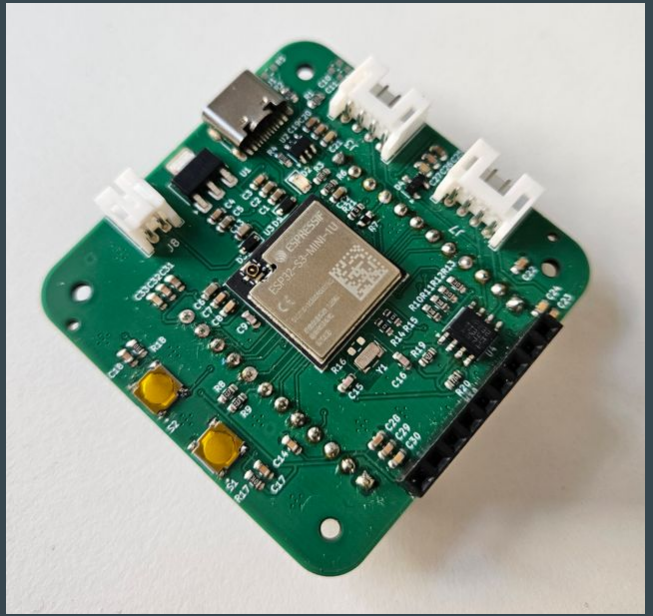
- Application + UI
- Graphical library
- ESP-BSP (Board Support Package)
- ESP-IDF

FRI STACK

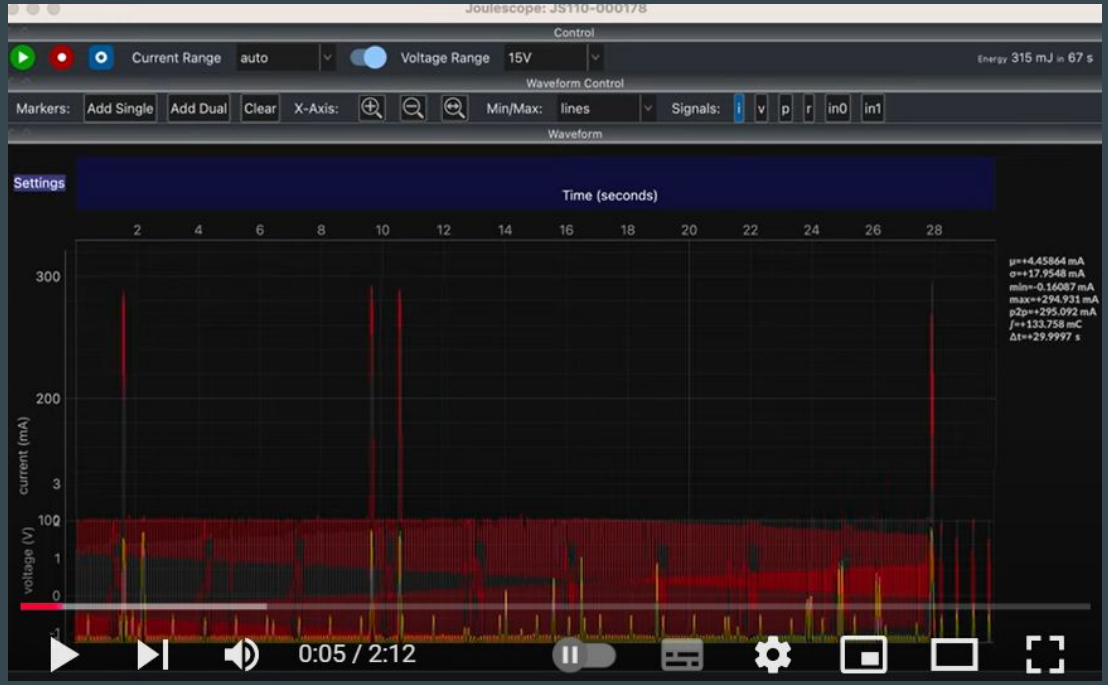
M5Stack compatible Open Hardware stack

Developed by Technical Cybernetics UNIZA

- https://github.com/andsim04/FRI_STACK_HARDWARE



ESP32-C6 TWT



https://www.youtube.com/watch?v=FA1jqZLig4s

Demo: Digital Twins - IoTCraft

Example of combining real and virtual world.

ESP32-C6 observing state of virtual lamp and syncing the state to real Neopixel.

- Access Point + DHCP + NAT on M5Stack AtomS3 using ESP-IDF
  - https://github.com/georgik/esp32-dhcp-server
- IoTCraft
  - https://github.com/georgik/iotcraft

Espressif in Brno

Vlněna Office Park

Espressif Systems (Czech) s.r.o.

Přízova 3, 602 00 Brno

Czechia, Europe

Espressif Developer Conference 2022-2024 - recording

https://www.youtube.com/@EspressifSystems

https://devcon.espressif.com/

Embedded World 2026

Meet us in Nuremberg, Germany

Old games ported to ESP32-P4

OpenTyrian - https://github.com/georgik/OpenTyrian/

Doom / FreeDoom - https://github.com/espressif/esp32-doom

Quake 1 / LibreQuake - https://github.com/espressif/esp32-quake