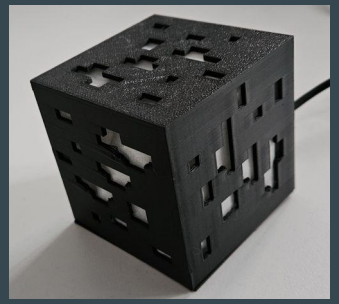
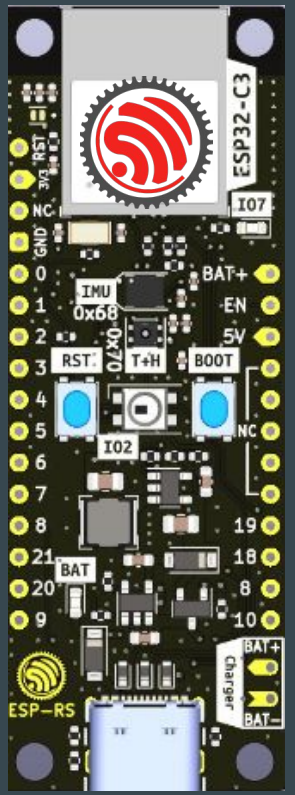
ESP32 Ecosystem

Embedded Development with
Rust, Swift, Python
TypeScript and C

2025-11-20
FEKT VUT

Juraj Michálek
Espressif Systems





Espressif team & friends

Juraj Michálek

- ESP32 Ecosystem Introduction

Petr Gaďorek

- ESP-NOW in Rust

Petr Kubica - https://helceletka.cz/robotarna/

- ESP32 in TypeScript & JavaScript

Jan Beran

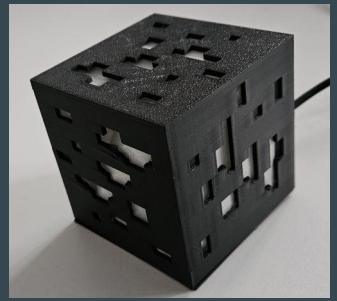
- ESP32-C6 Workshop

Embedded Projects

IoTCraft

Digital Twin in Voxel World

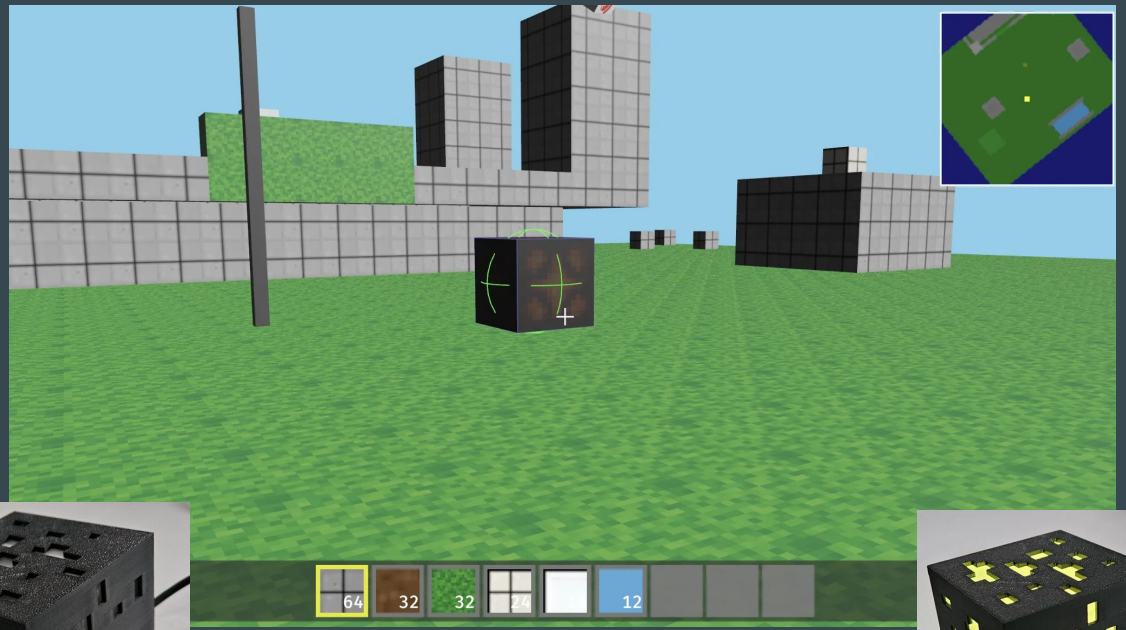
https://github.com/georgik/iotcraft

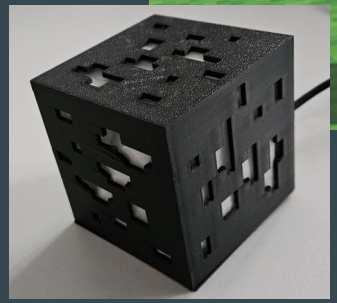


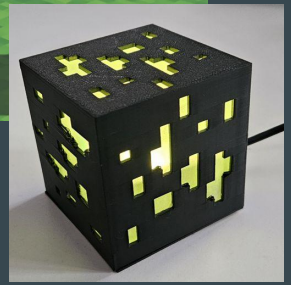
IoTCraft

Digital Twin in Voxel World

https://github.com/georgik/iotcraft







ESP-Rainbow

https://www.crowdsupply.com/atomic14/esp32-rainbow

Retro ESP32

https://github.com/retro-esp32/RetroESP32

OpenTyrian

https://github.com/georgik/OpenTyrian - using SDL3

esp32-quake + LibreQauake

https://github.com/espressif/esp32-quake

https://github.com/lavenderdotpet/LibreQuake

Old games ported to ESP32-P4

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

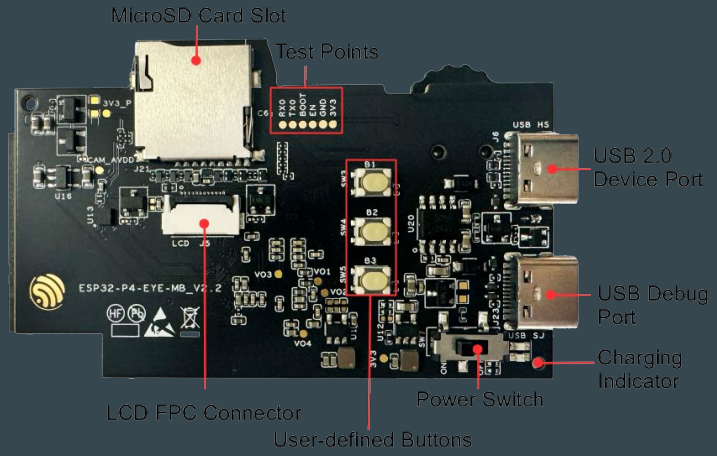
Doom / FreeDoom - https://github.com/espressif/esp32-doom

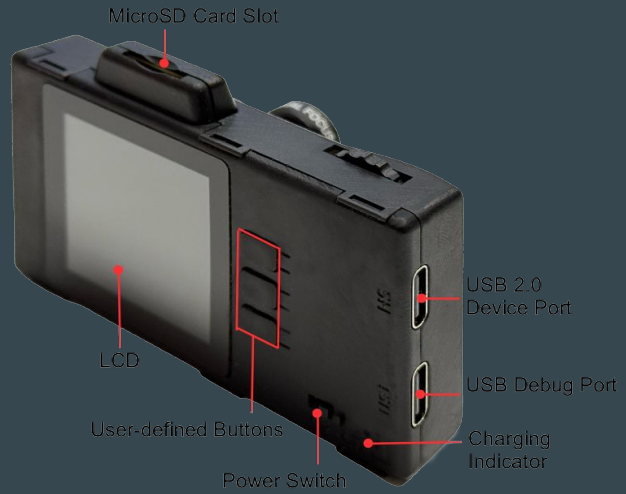
Quake 1 / LibreQuake - https://github.com/espressif/esp32-quake

ESP32-P4-EYE

https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32p4/esp32-p4-eye/user_guide.html

https://github.com/espressif/esp-dev-kits/tree/master/examples/esp32-p4-eye





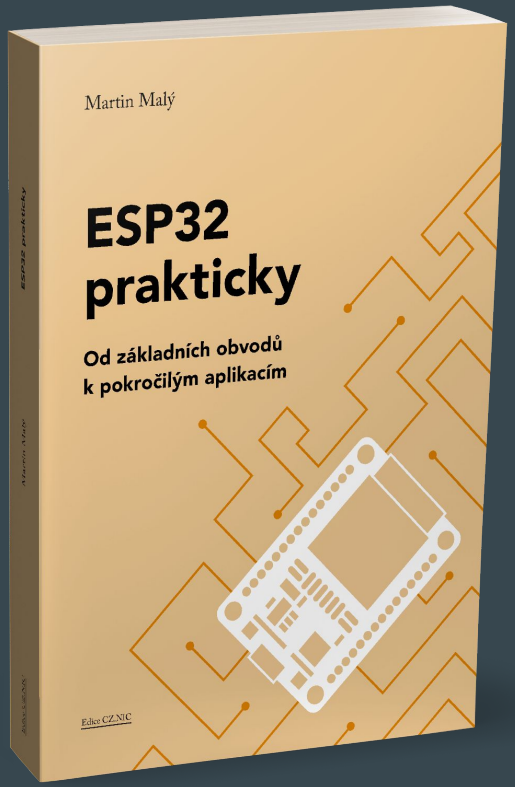
EchoEar

https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/echoear/index.html

Book recommendation: ESP32 prakticky

https://espx.cz

https://github.com/espx-cz



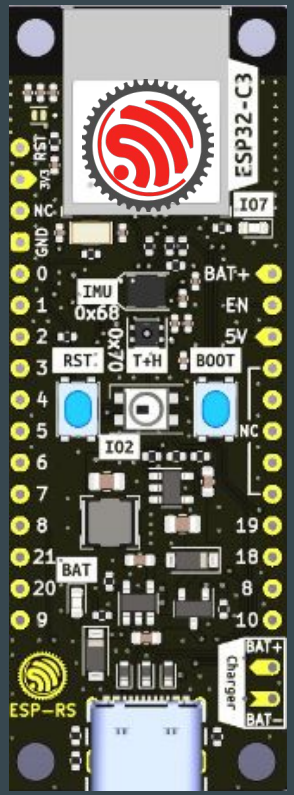
Open Hardware - esp-rust-board

KiCad templates

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

ESP32-C3-DevKit-RUST-1

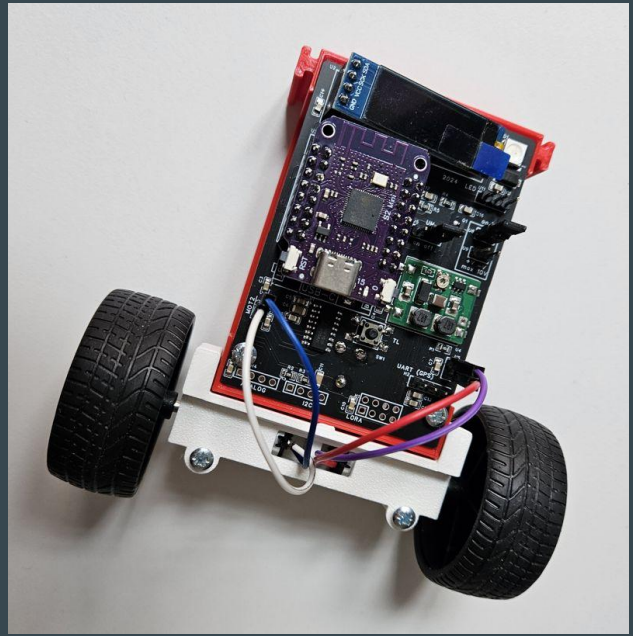
https://www.espressif.com/en/products/devkits



Racing Car in MicroPython

ESP32-S2 - WiFi controller car

Developed by Technical Cybernetics UNIZA

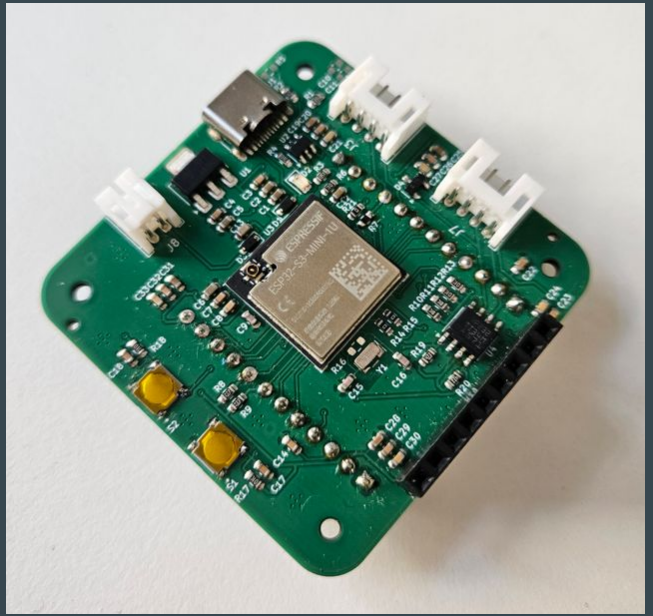


FRI STACK

M5Stack compatible Open Hardware stack

Developed by Technical Cybernetics UNIZA

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



RISC-V vs Xtensa

RISC-V

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

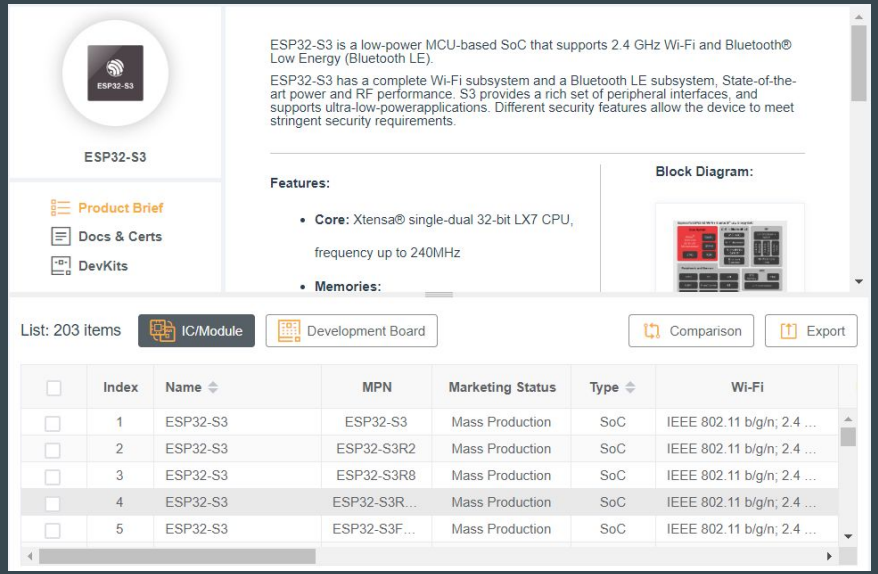
Xtensa

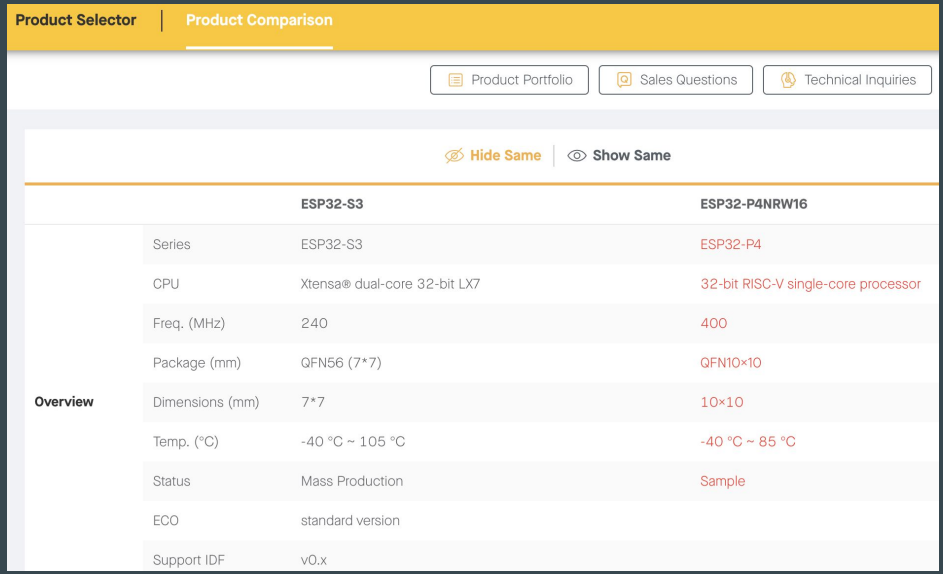
- older
- ESP32, ESP32-S2, ESP32-S3
- requires custom installation
- espup install

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

ESP Product Selector + Product Comparison

https://products.espressif.com/





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

IDE for 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/
- free license for students and open source projects

Rust

slint
GUI toolkit

Slint

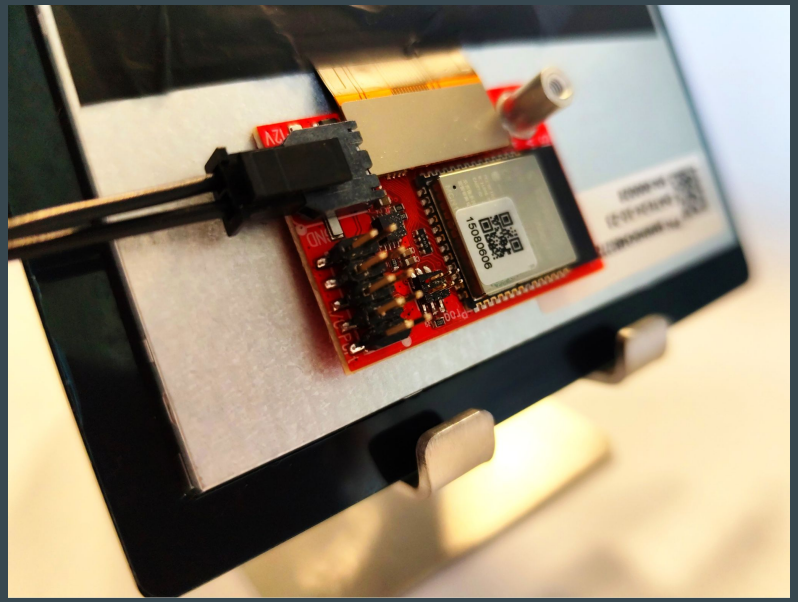
Quick prototyping

Free and commercial version

Slint Workshop

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

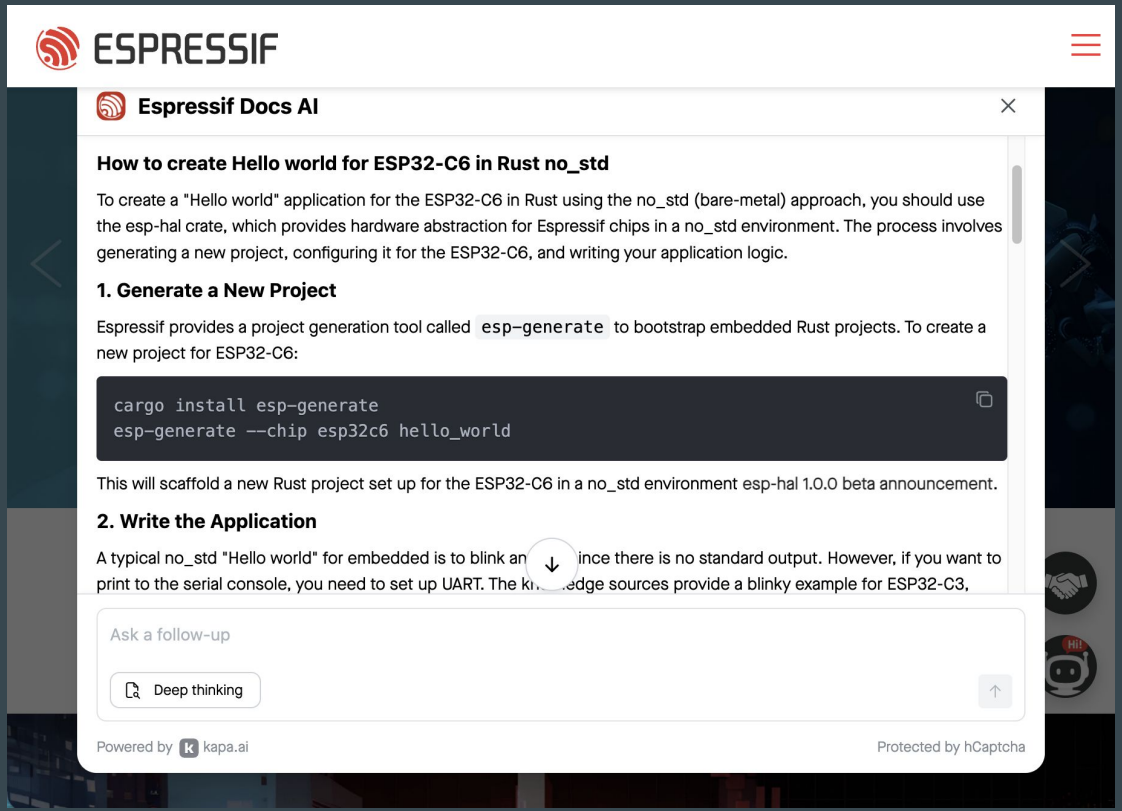
ESoPe board + RGB Schukat Smartwin display-concept



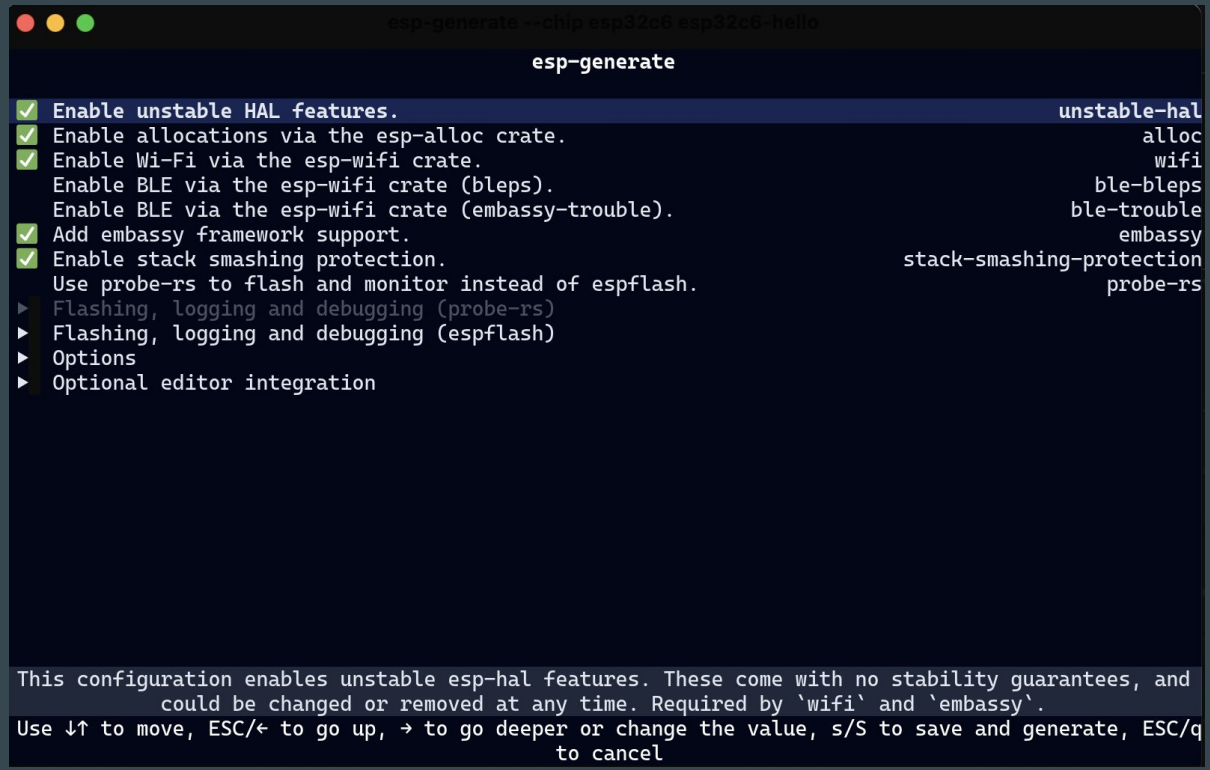
Will be published at Developer Portal: https://github.com/espressif/developer-portal/pull/495

We welcome feedback.

Quick start with AI at espressif.com



esp-generate --chip esp32c6

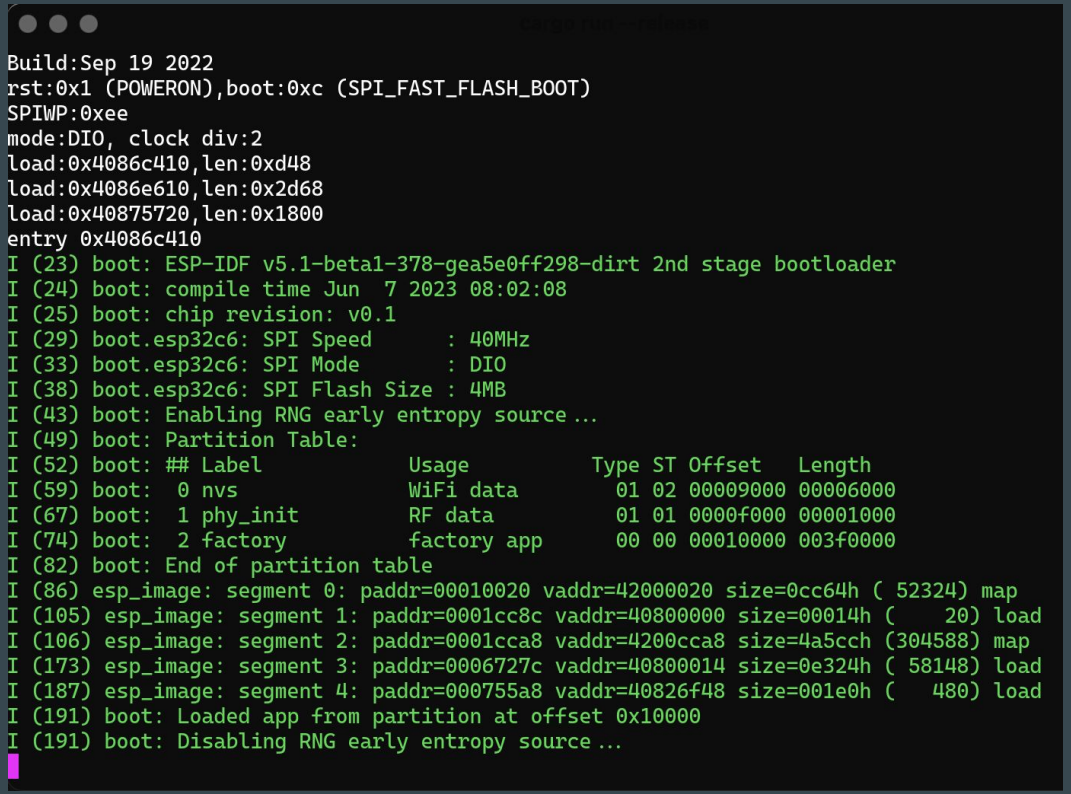


esp-generate - more options

Build, flash, monitor

cargo run --release

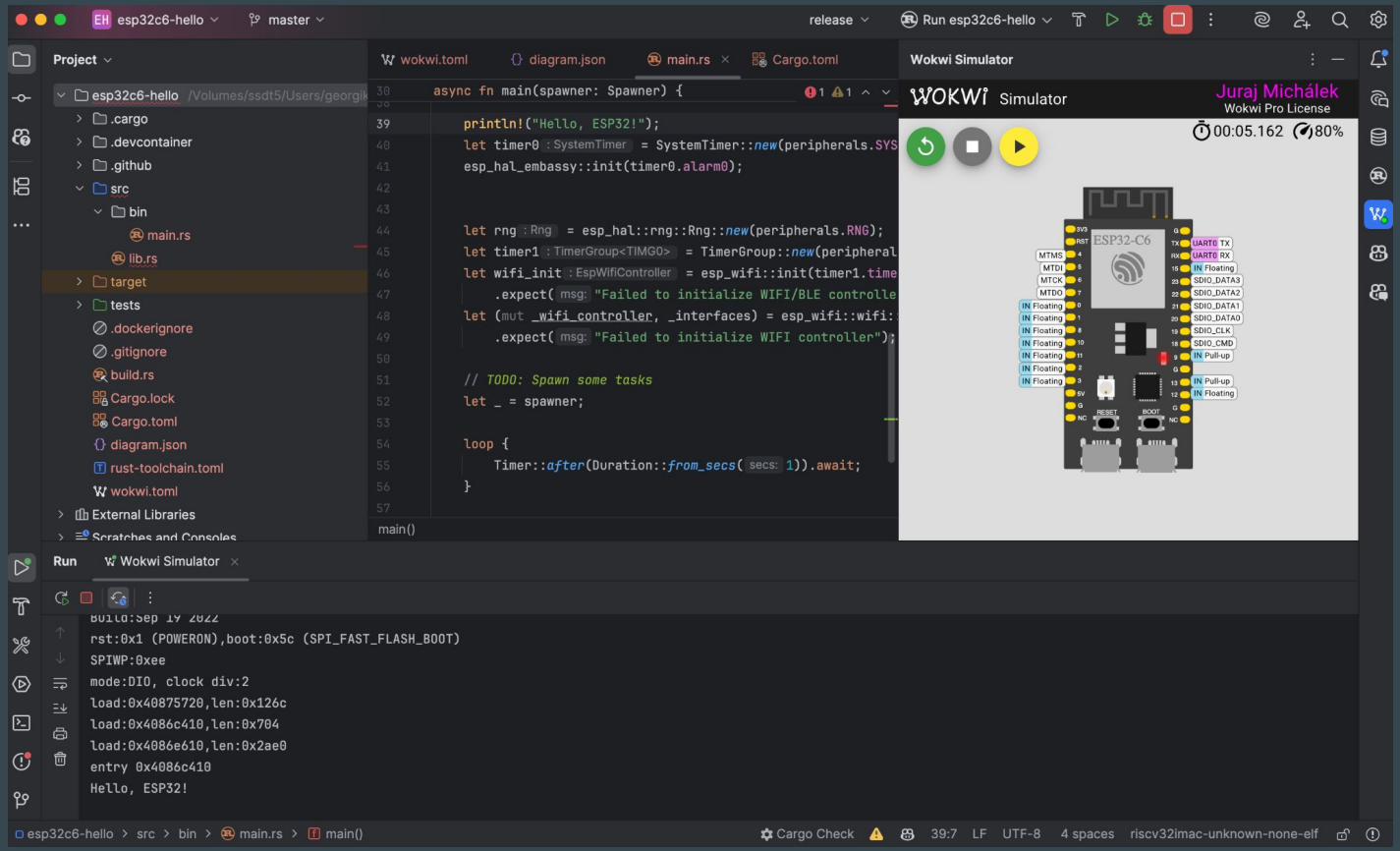
shorter: cargo r -r



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"
```

CLion with Rust plugin or RustRover



https://developer.espressif.com

ESPRESSIF Developer Portal

Blog Workshops Events Products Quick Links

esp-hal 1.0.0 release announcement

30 October 2025 · 3 mins ·

Esp32 Rust Xtensa RISCV Announcement

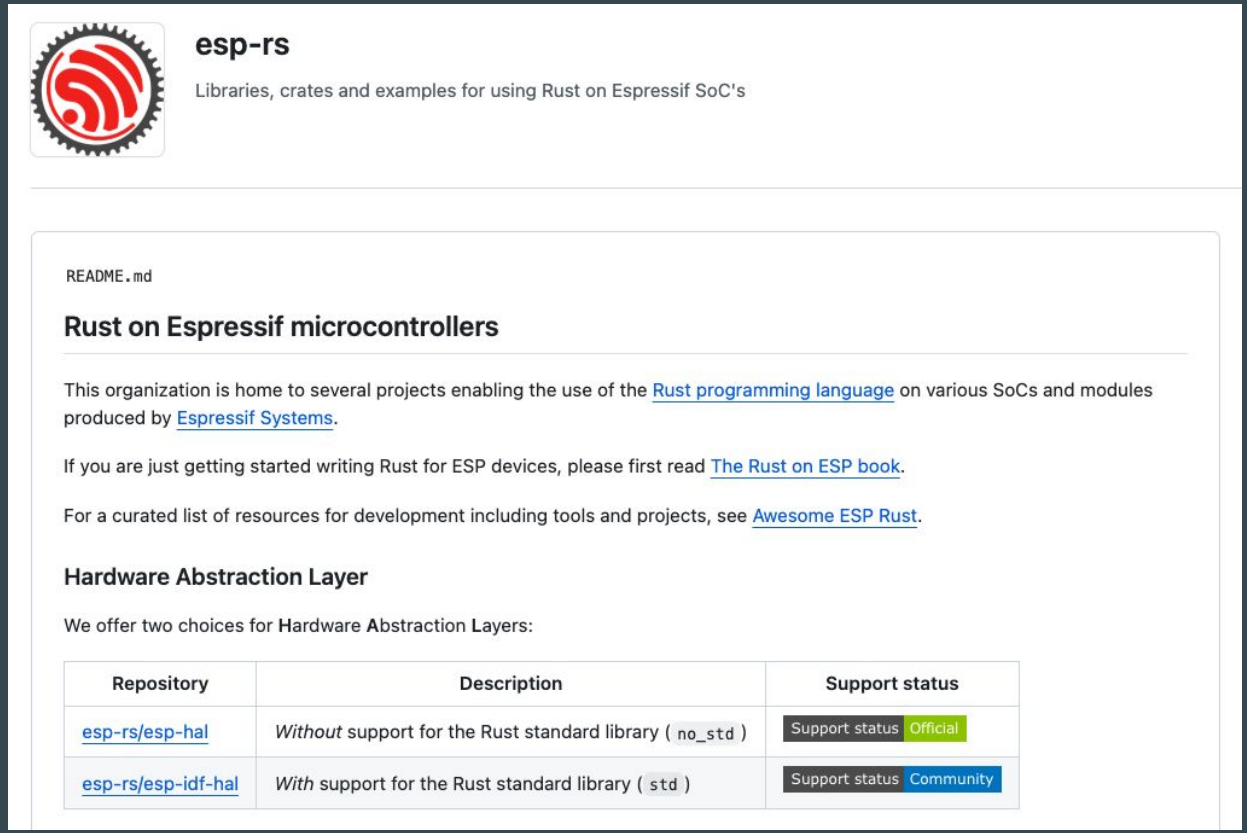
AUTHOR
Scott Mabin

▶ Table of Contents

> ***Announcing esp-hal 1.0, the first Rust SDK for embedded devices.***

In February this year, we announced the first `esp-hal` 1.0 beta release. Since then we've been hard at work, polishing and preparing for the full release. Today, the Rust team at Espressif is excited to announce the official `1.0.0` release for `esp-hal`, the *first* vendor-backed Rust SDK!

Open Source - https://github.com/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.

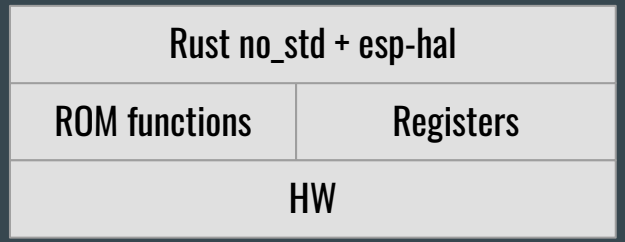
If you are just getting started writing Rust for ESP devices, please first read The Rust on ESP book.

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



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

Official Espressif + Community Support

Requires Rust compiler

Bleeding edge MCUs and drivers might be missing

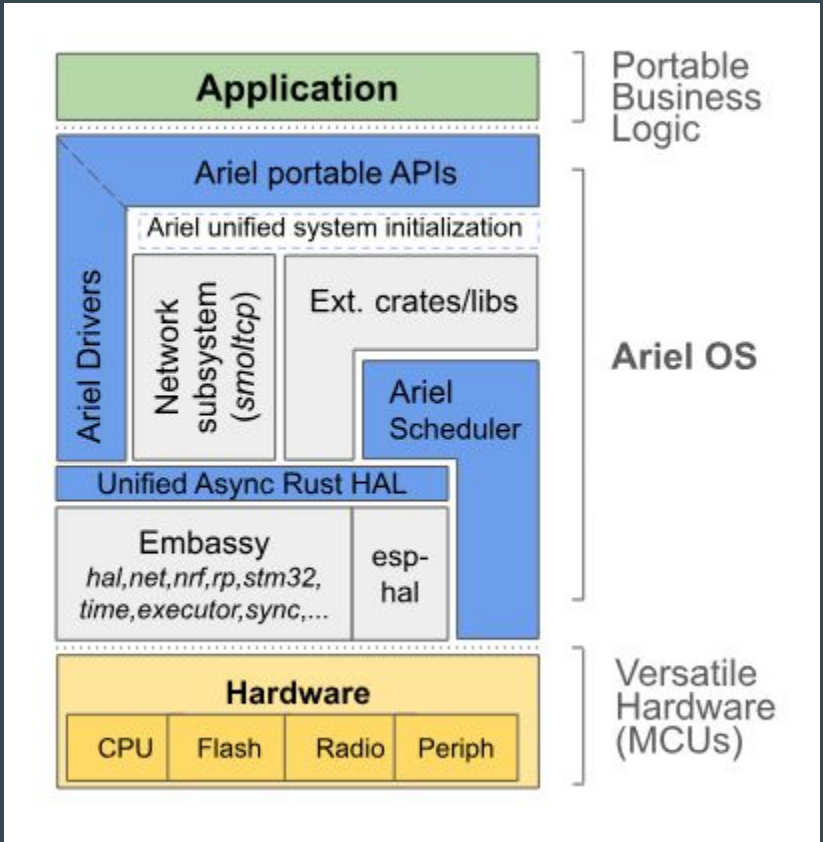
https://github.com/esp-rs/esp-idf-hal

Community support

Requires Rust compiler + ESP-IDF

Bleeding edge MCUs and drivers from ESP-DF

Ariel OS

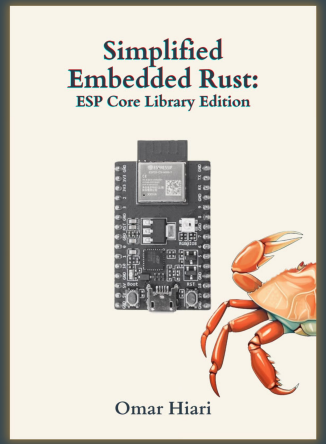


https://ariel-os.github.io/ariel-os/dev/docs/book/

Training

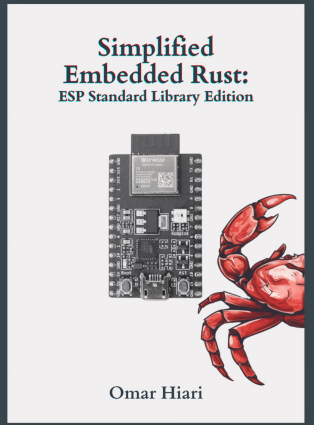
Rust no_std

- https://docs.espressif.com/projects/rust/no_std-training/
- https://github.com/esp-rs/no_std-training
- https://www.theembeddedrustacean.com/c/ser-no-std

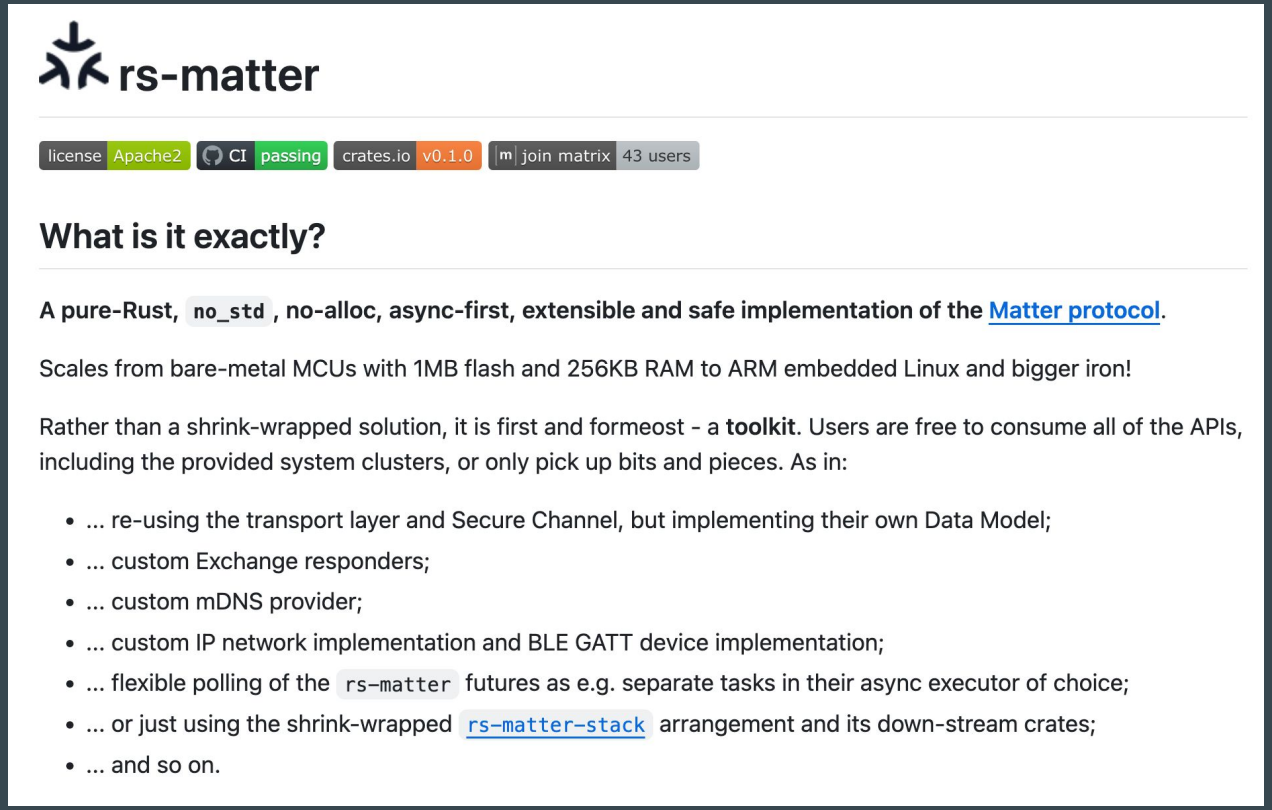


Rust std

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



rs-matter

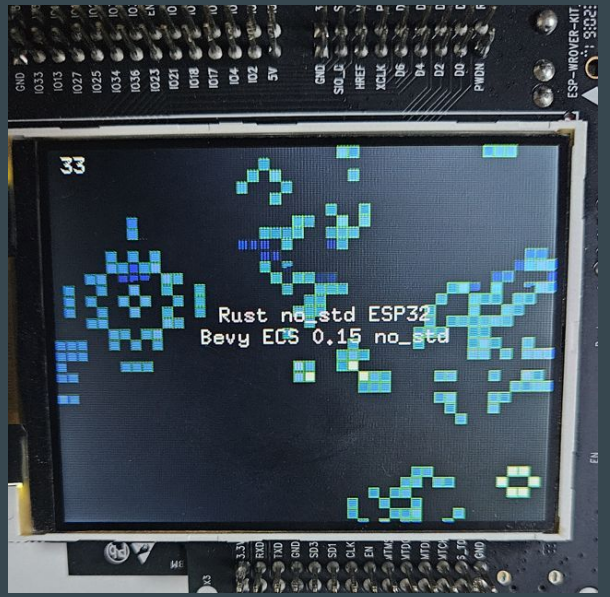


https://github.com/project-chip/rs-matter

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-conways-game-of-life-rs

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

Swift

ESP32 at Do iOS 2024 - Amsterdam

https://do-ios.com/

https://youtu.be/HFnAvDZtW4A?si=NmlpZ53aoVsfAziW

Swift Embedded Examples

Install via Swiftly: https://www.swift.org/swiftly/documentation/swiftly/getting-started

swiftly install && hash -r

idf.py set-target esp32-c6

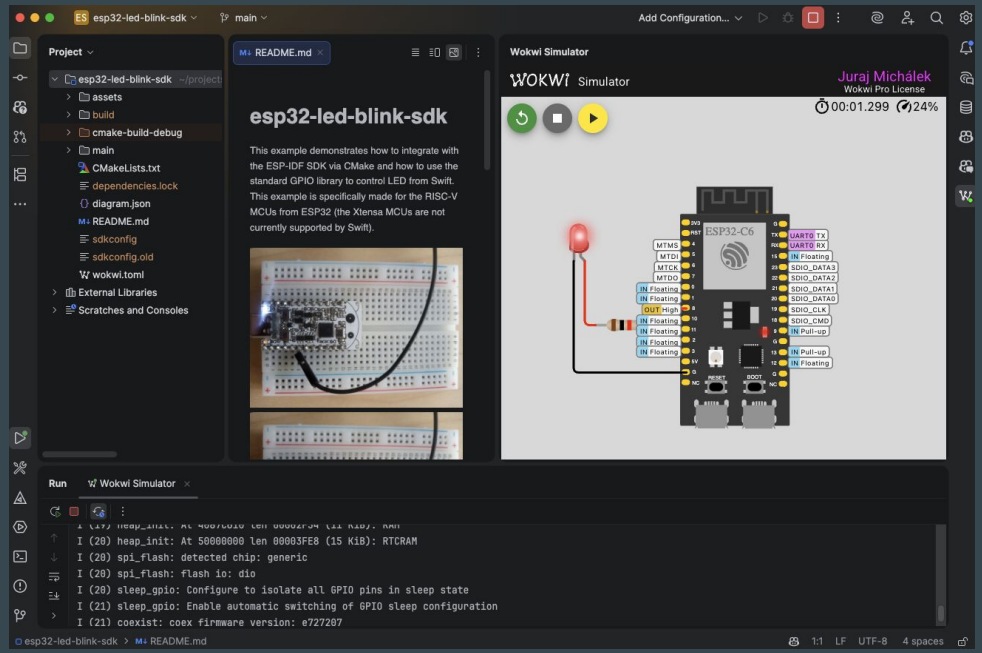
idf.py build flash monitor

Quit monitor: CTRL+]

https://github.com/swiftlang/swift-embedded-examples

Swift Embedded Examples - esp32-led-blink-sdk

CLion + Wokwi plugin



https://developer.espressif.com/blog/clion/

https://plugins.jetbrains.com/plugin/23826-wokwi-simulator

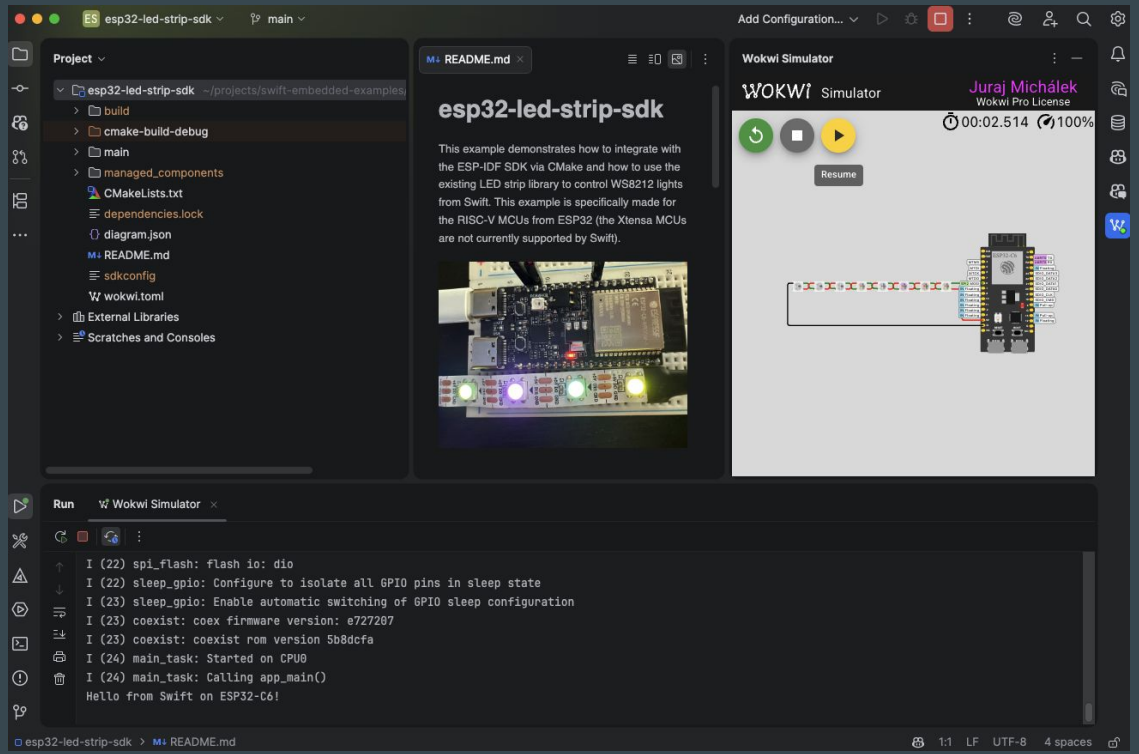
Swift Embedded Examples - esp32-led-strip-sdk

Architecture of ESP-IDF project:

Components

- main
- managed_components
- components (local)

main/idf_component.yml



Supported MCUs

All modern RISC-V: ESP32-C*, ESP32-H*, ESP32-P* family

- easy use of nightly upstream toolchain

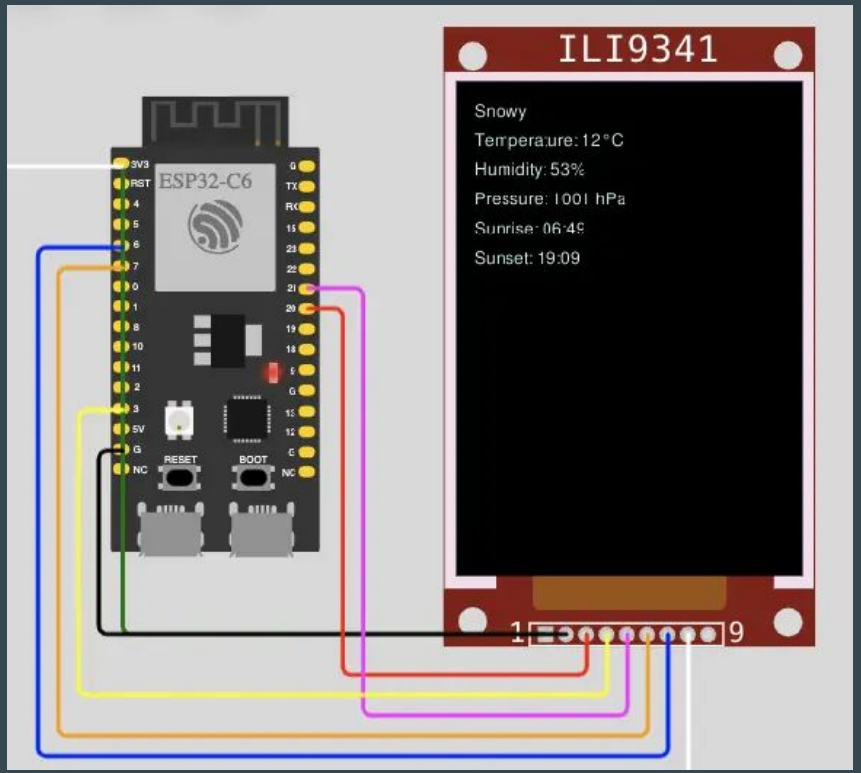
More complex setup: ESP32, ESP32-S* family

- nos supported
- would require custom build of Swift toolchain
- this MCUs are older

ESP32 Weather Display

Display connected via SPI

Board Support Package - using Generic



https://github.com/georgik/esp32-swift-weather-display

Online simulation with Wokwi

Smooth integration with C libraries - SDL3

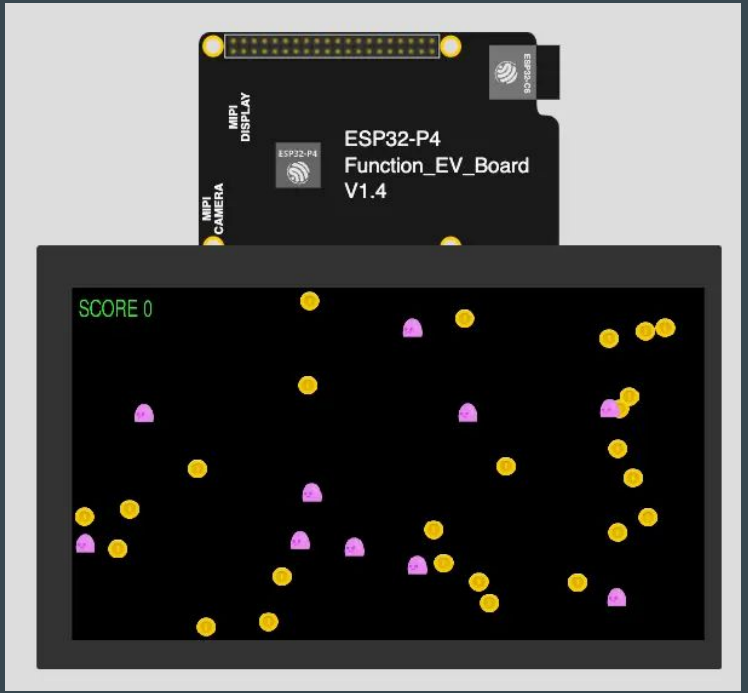
Example with high performance ESP32-P4

RISC-V IMACF

https://github.com/georgik/esp32-sdl3-swift-example

Online simulation with Wokwi

ESP-IDF SDL3 Component



Embedded Swift - Bare Metal

CMSIS System View Description (CMSIS-SVD)

- https://github.com/espressif/svd

Patched version used in Rust no_std

- https://github.com/esp-rs/esp-pacs/
- https://github.com/apple/swift-mmio

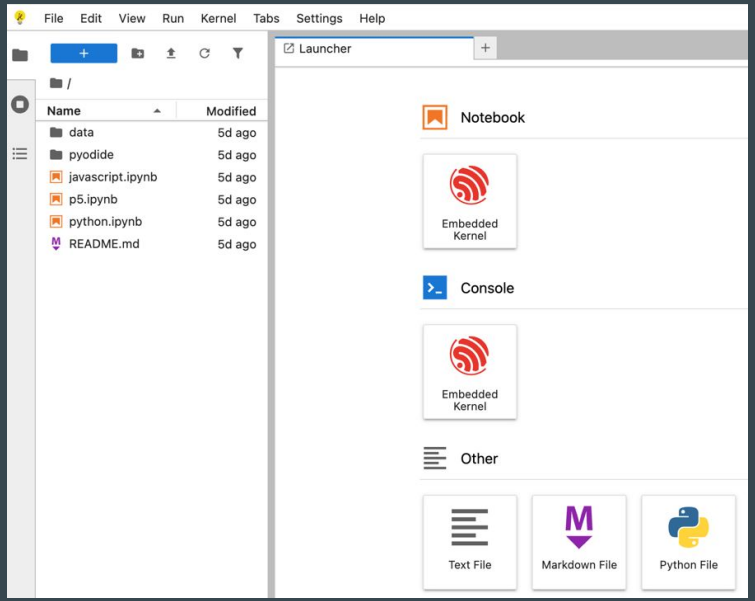
```
SVD2Swift \
  --input ../esp-pacs/esp32c6/svd/esp32c6.svd \
  --output Sources/Registers \
  --access-level public \
  --peripherals GPIO UART0 SPI0
```

MicroPython

Flow3r

https://flow3r.garden/

Jupyter Lite - MicroPython - ESP32

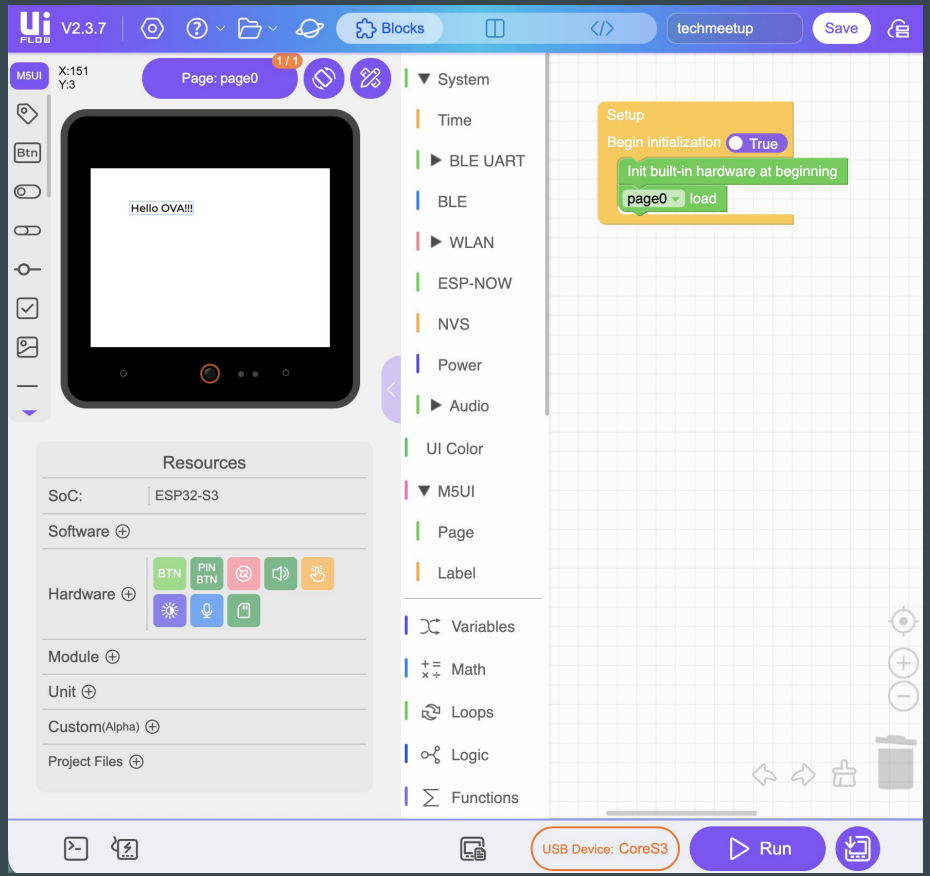


https://espressif.github.io/jupyter-lite-micropython/lab/index.html

M5 Stack UIFlow2

https://uiflow2.m5stack.com/

Blockly + MicroPython

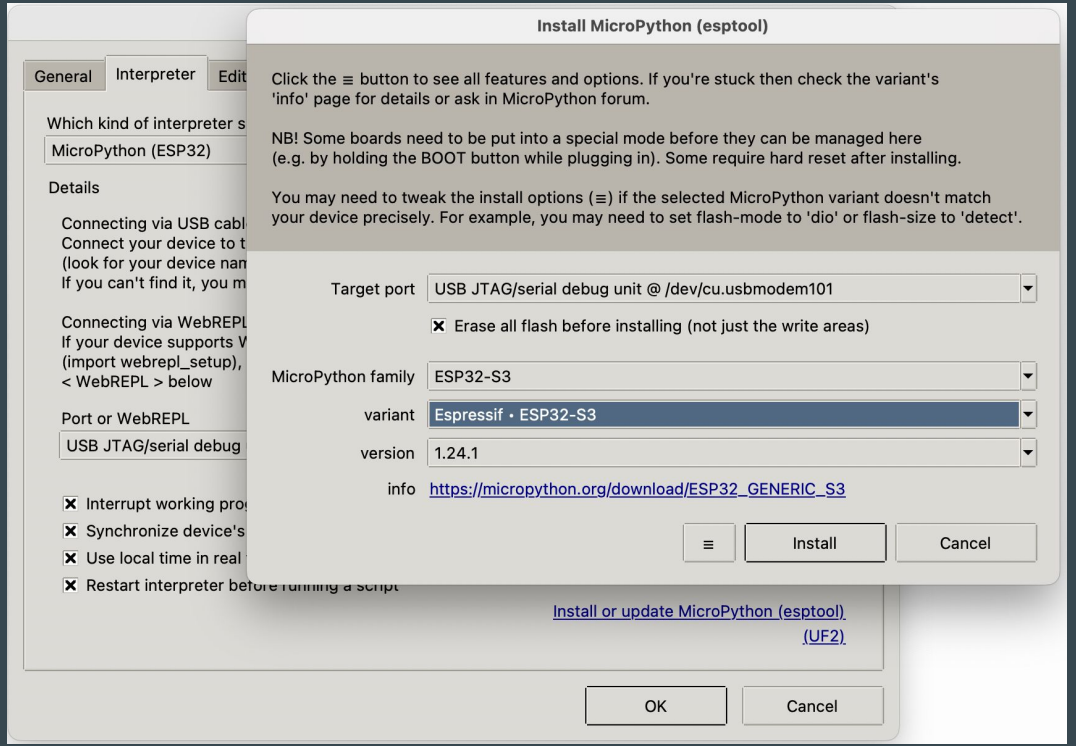


Thonny

Python IDE for beginners

https://thonny.org/

Flash MicroPython directly from IDE



TypeScript - Jaculus

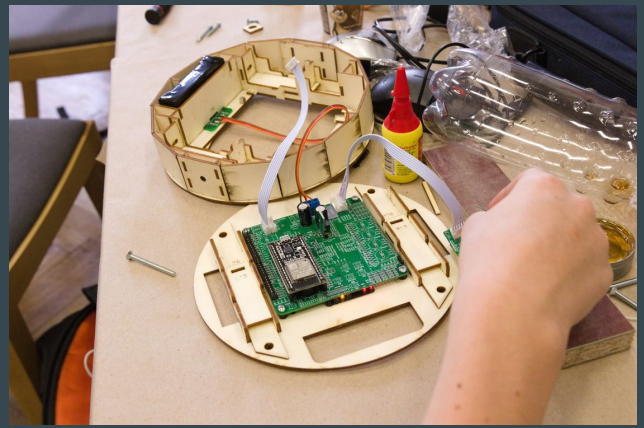
Jaculus

Robotics Brno

https://jaculus.org/

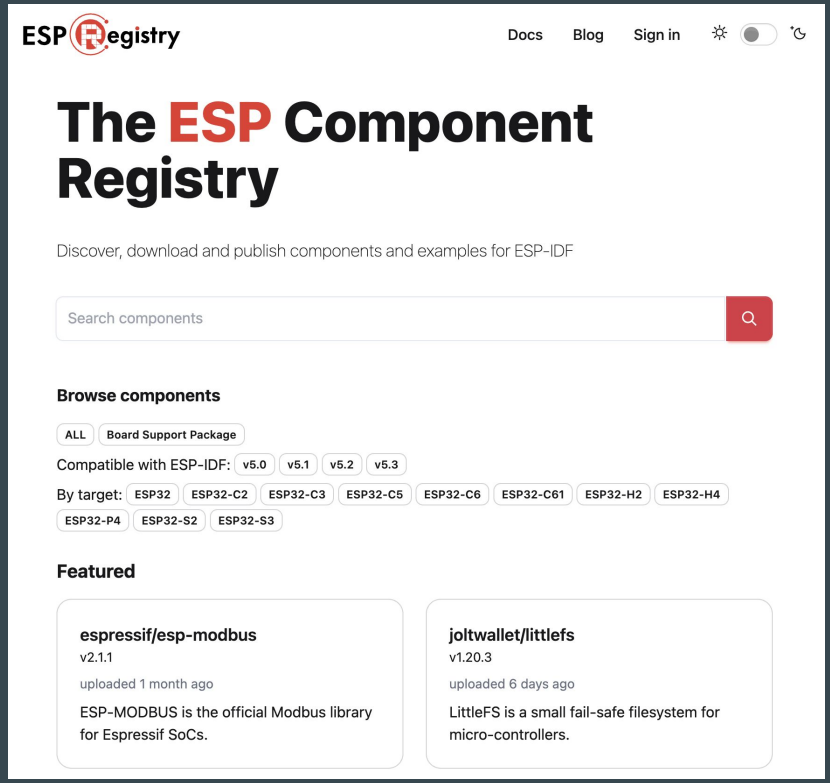
https://installer.jaculus.org/

https://f.kubaandrysek.cz/Jaculus-Wokwi/



Components

- a.k.a. - libraries
- https://components.espressif.com/



Lua as ESP-IDF component

https://github.com/georgik/esp-idf-component-lua

```
luaL_openlibs(L);
// Set the Lua module search path to include the assets directory
if (luaL_dostring(L, "package.path = package.path .. ';./?.lua;/assets/?.lua'")) {
    ESP_LOGE(TAG, "Failed to set package.path: %s", lua_tostring(L, -1));
    lua_pop(L, 1);  // Remove error message from the stack
}
```

Smooth integration with C libraries - SDL3

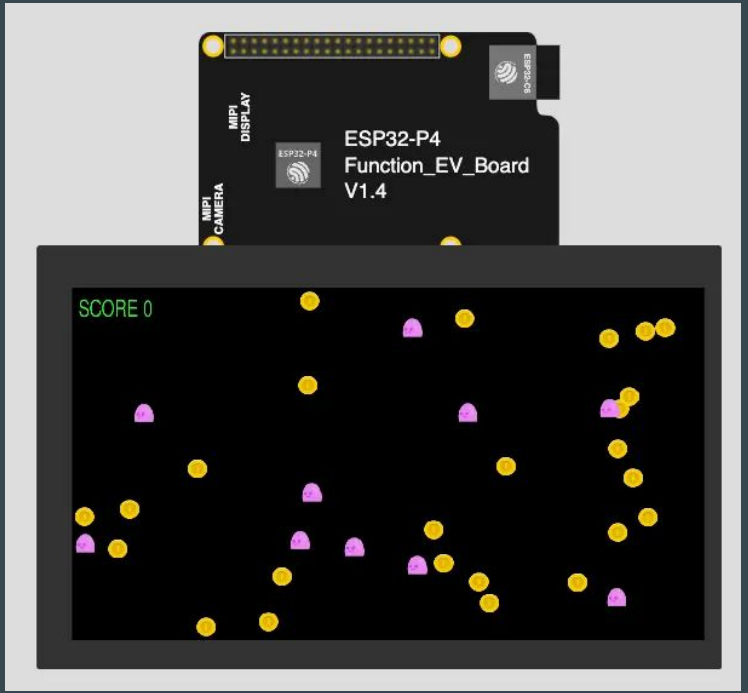
Example with high performance ESP32-P4

RISC-V IMACF

https://github.com/georgik/esp32-sdl3-swift-example

Online simulation with Wokwi

ESP-IDF SDL3 Component



Raylib

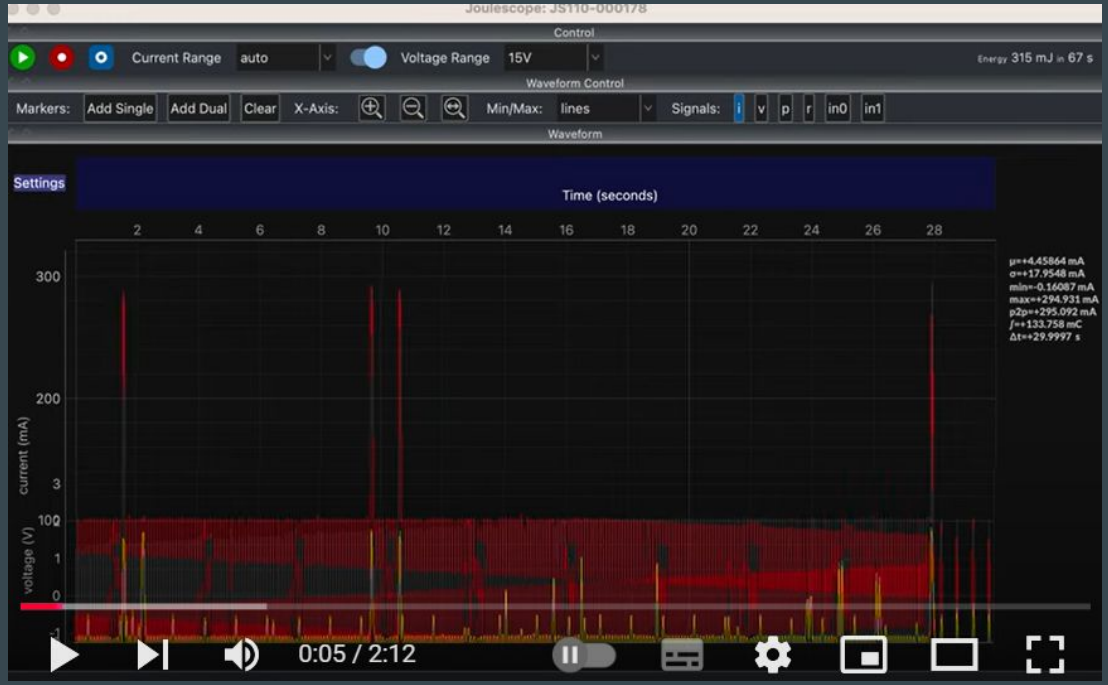
Upstream C library used in ESP-IDF project using git submodule wrapper technique

https://components.espressif.com/components?q=raylib

https://developer.espressif.com/blog/2025/10/debugging-embedded-graphics-with-wokwi-and-ai/

https://github.com/georgik/esp-idf-component-raylib

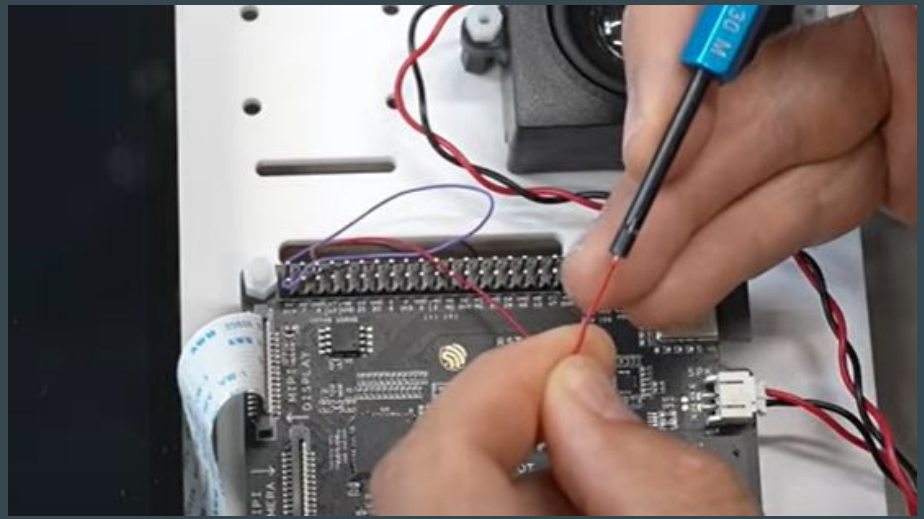
ESP32-C6 TWT



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

Tips for Makers

Wire Wrap Tool



Espressif DevCon 2024 - Tips and Tricks

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 Download Boot mode

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

WiFi Power Saving Modes

ESP32-C6 by default tries to optimize energy usage by default

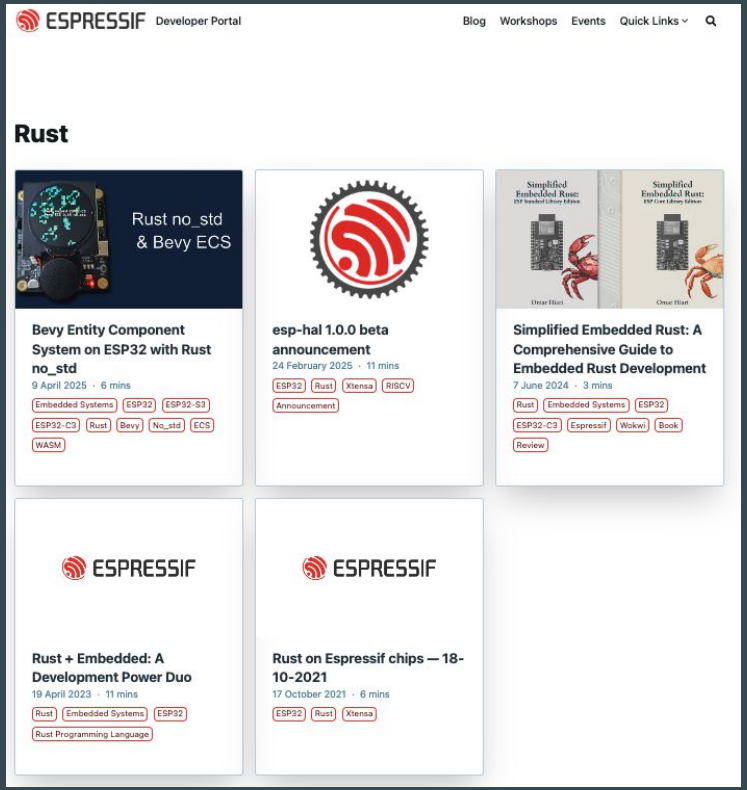
esp_wifi_set_ps(WIFI_PS_MIN_MODEM)

esp_wifi_set_ps(WIFI_PS_MAX_MODEM)

https://docs.espressif.com/projects/esp-idf/en/latest/esp32c6/api-guides/wifi.html#esp32-c6-wi-fi-power-saving-mode

Developer Portal

developer.espressif.com



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

ESP Launchpad

Flash your firmware from web

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

Espressif Developer Conference 2022-2024 - recording

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

https://devcon.espressif.com/

Embedded World 2026

Meet us in Nuremberg, Germany

Elektor Mag

Special Guest Edition

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

Espressif in Brno

Vlněna Office Park

Espressif Systems (Czech) s.r.o.

Přízova 3, 602 00 Brno

Czechia, Europe