

# Rust in the Industry

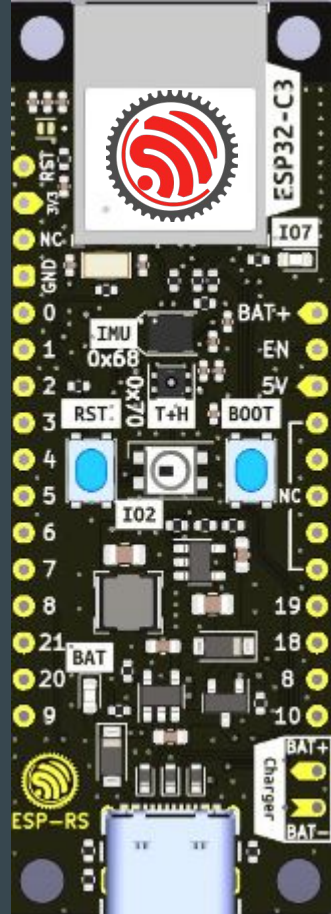
—

## Secure and Sustainable Development



2023-10-24  
betterCode Rust - on-line

Juraj Michálek - Espressif Systems



# Content

Context of Embedded development

Example of **C** app calling **Rust** functions

Example of Rust **Bare Metal** development

Links and tips

# Espressif System

Espressif Systems is a public multinational, fabless semiconductor company established in 2008, with offices in China, the Czech Republic, India, Singapore and Brazil.

Espressif Leads the IoT Chip Market with Over **1 Billion** Global Shipments

ESP8266 - introduces WiFi

ESP32, ESP32-S - Xtensa cores

ESP32-C, H, P series - RISC-V cores

Open source: <https://github.com/espressif>



# C/C++, Rust - Open source

C/C++ - 213 repositories at <https://github.com/espressif>

- 11 400+ starts - ESP-IDF
- Espressif IoT Development Framework (CMake based)

Rust - 51 repositories at <https://github.com/esp-rs/>

- 400+ starts - esp-hal



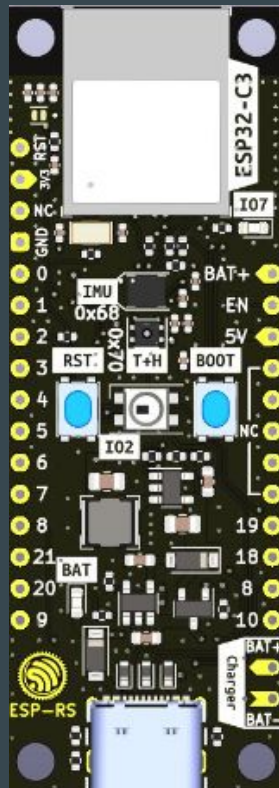
# Designing Open Hardware - esp-rust-board

KiCad templates

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

ESP32-C3-DevKit-RUST-1

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



# Desktop vs. Embedded development

Memory: **GB** (TB) vs. **kB** (MB bytes of PSRAM)

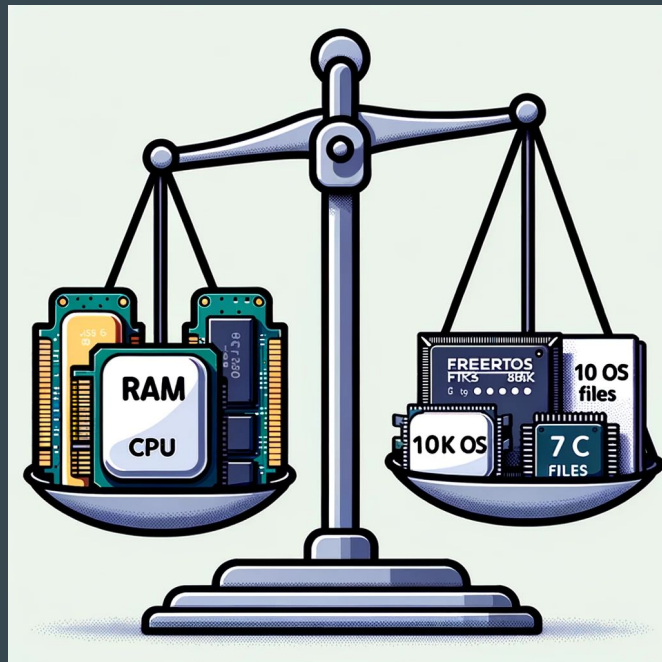
CPU Cores: **8 - 64** vs. **1 - 2**

CPU Frequencies: GHz vs. 20-240 MHz

Power consumption: 100s of Watts vs. Milliwatts

OS kernel: **61k** files vs. 10k files in ESP-IDF

(FreeRTOS kernel: **7** C files)



# OSes and integration and Rust



no\_std a.k.a. bare metal with Rust - <https://github.com/esp-rs/esp-hal> (minimalistic)

ESP-IDF (OS based on FreeRTOS) - <https://github.com/esp-rs/esp-idf-hal>



Zephyr - <https://zephyrproject.org/>

- EDC22 Day 1 Talk 10: Applications of Asymmetric Multiprocessing with ESP32 Devices - including Rust on one core - <https://youtu.be/oble9ObAqxM>



NuttX - <https://nuttx.apache.org/> (as app, Linux-like OS)

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

# Programming languages

## Active support by Espressif teams

- C/C++
  - most common choice - <https://github.com/espressif/esp-idf>
- Rust
  - security and memory guaranties of Rust
  - <https://github.com/esp-rs>
  - multi-target support: Xtensa, RISC-V, plus WASM, desktops or mobile
- Arduino - Maker choice
  - Arduino IDE 2.x
  - note: check the license for production



**esp-rs**

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



# News: Ferrocene compiler - the major milestone in 2023

Ferrocene is an open source qualified Rust compiler toolchain. Ferrous Systems invested its decade of Rust experience to make Rust a first-class language for mission-critical and functional safety systems.

For its first release, Ferrocene 23.06 is a ISO 26262 (ASIL D) and IEC 61508 (SIL 4) qualified version of the existing open-source compiler, rustc, based on Rust 1.68.

<https://ferrous-systems.com/ferrocene/>

Why Ferrocene? <https://www.youtube.com/live/i06djj0KvB8?si=mxXuuWPMpSs-JC2r>

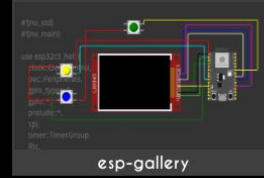
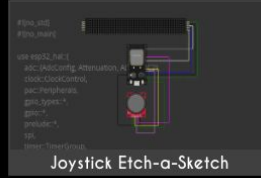
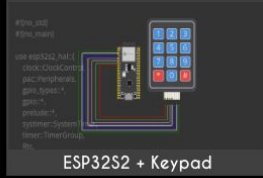
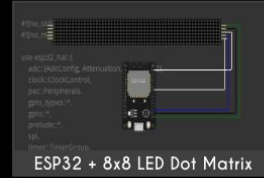
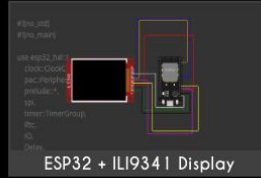
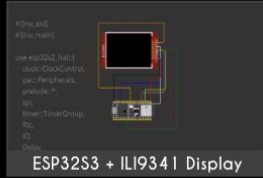
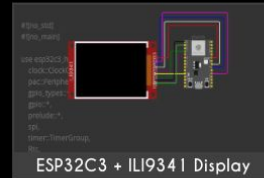
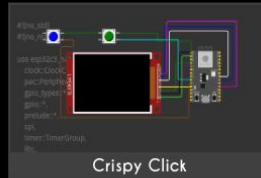
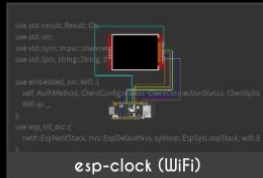
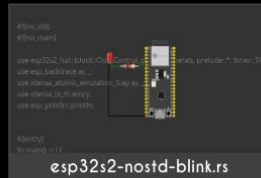
# wokwi.com/rust

Contribute: <https://github.com/wokwi>

EDC22 Day 1 Talk 9: Your browser is ESP32  
- Wokwi - <https://youtu.be/TKe4MgD6O8o>

Rust project examples

+ NEW PROJECT



# IDEs with Rust support

Supported by Espressif:



- VS Code with Espressif Extension

<https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/vscode-setup.html>

Supported by JetBrains:

- RustRover - <https://www.jetbrains.com/rust/>



# Rust component in C/C++ project

ESP-IDF has notion of components (libraries)

Integration of Rust as a component:

<https://github.com/georgik/esp32-idf-no-std-rust-component>

Example with NMEA crate

<https://github.com/georgik/esp32-idf-nmea-example>

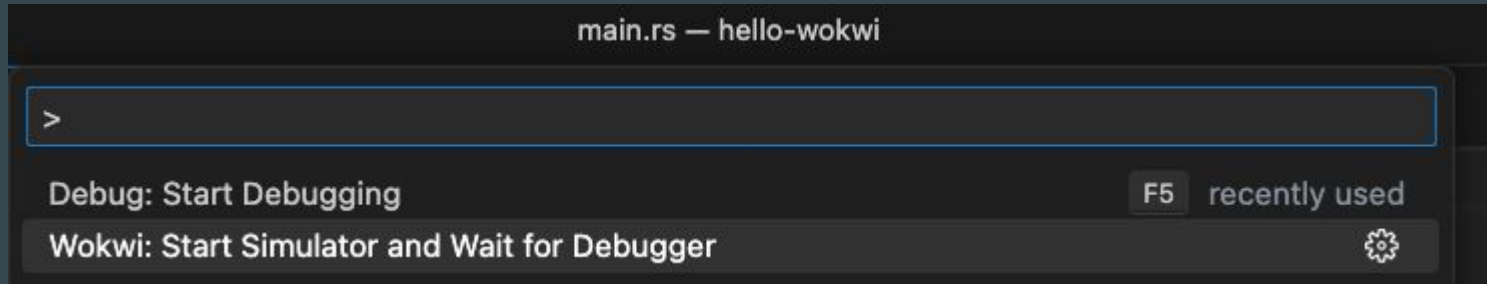
```
idf.py set-target esp32-c3
```

```
idf.py -DCMAKE_BUILD_TYPE=Debug build
```



<https://aerorust.org/>

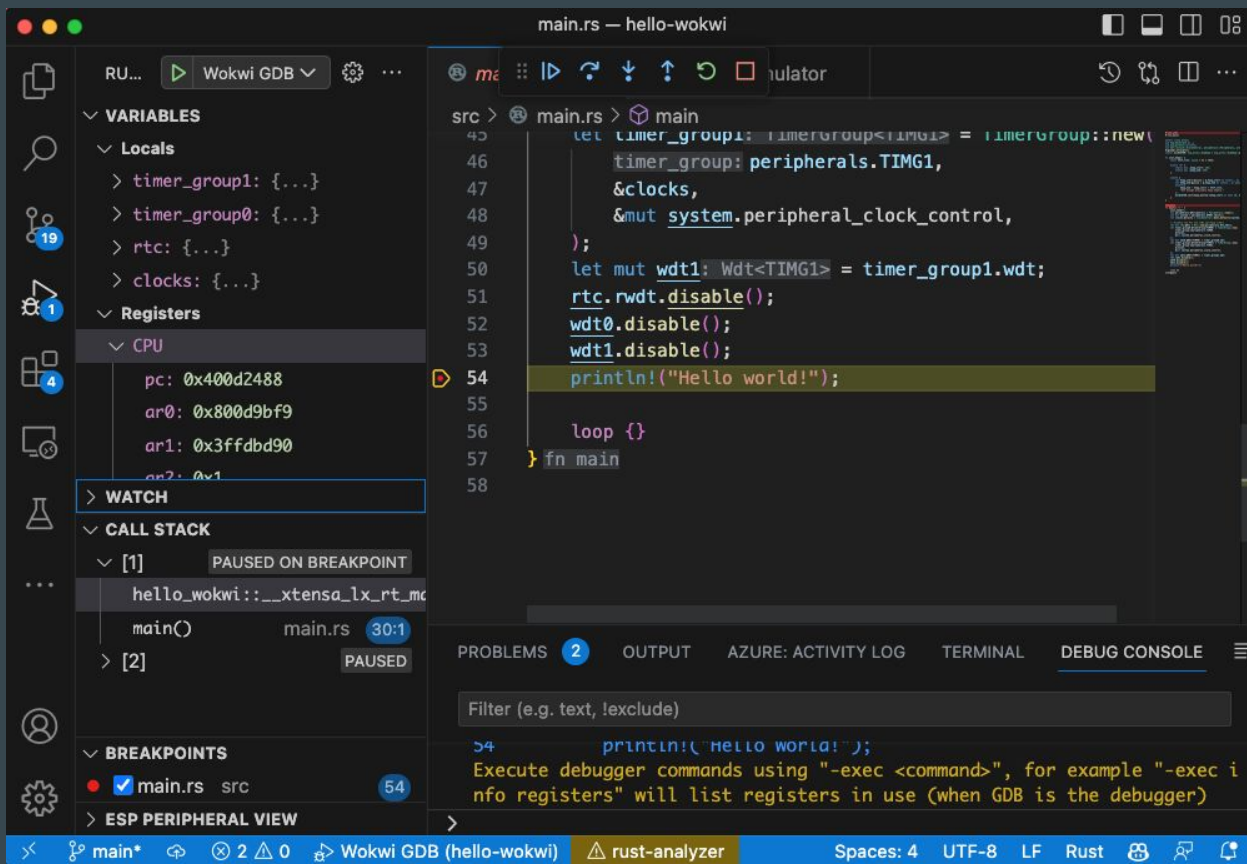
# Debugging



2 steps:

- start simulator with debugger interface
- start debugging session

# Debugging session with Wokwi



# Templates for new projects

## std

- leverage APIs of ESP-IDF
- pick exact version of ESP-IDF (not master branch)
- `cargo generate --vcs none --git https://github.com/esp-rs/esp-idf-template cmake`
- <https://github.com/esp-rs/esp-idf-template>

## no\_std

- no C dependencies
- <https://github.com/esp-rs/esp-template>

# Starting from template

`cargo generate esp-rs/esp-template`

```
👤 Project Name: better-code
🔧 Destination: /Users/georgik/projects/better-code ...
🔧 project-name: better-code ...
🔧 Generating template ...
✅ 👤 Which MCU to target? · esp32
✅ 👤 Configure advanced template options? · true
✅ 👤 Configure project to support Wokwi simulation with Wokwi VS Code extension? · true
✅ 👤 Setup logging using the log crate? · true
✅ 👤 Configure project to use Dev Containers (VS Code and GitHub Codespaces)? · true
✅ 👤 Enable WiFi/Bluetooth/ESP-NOW via the esp-wifi crate? · true
✅ 👤 Add CI files for GitHub Action? · true
✅ 👤 Enable allocations via the esp-alloc crate? · true
```

`cargo build --release`

`cargo espflash flash --release --monitor`



# Rust Embedded Graphics

<https://github.com/embedded-graphics>

embedded-graphics crate + driver (e.g. for SPI display)

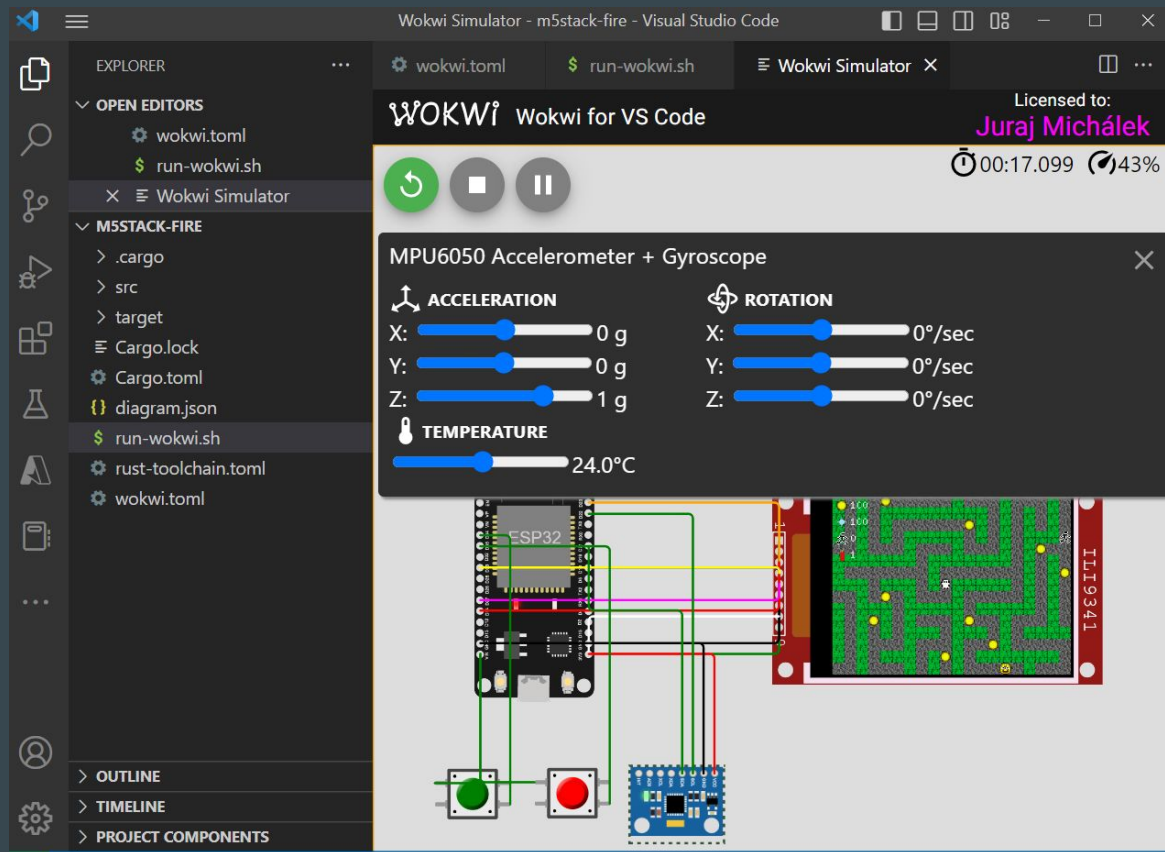


# Wokwi - VS Code Plugin

Add wokwi.toml and diagram.json  
to your project

CTRL+Shift+P - Wokwi: Start Simulator

Example: <https://github.com/georgik/esp32-spooky-maze-game>

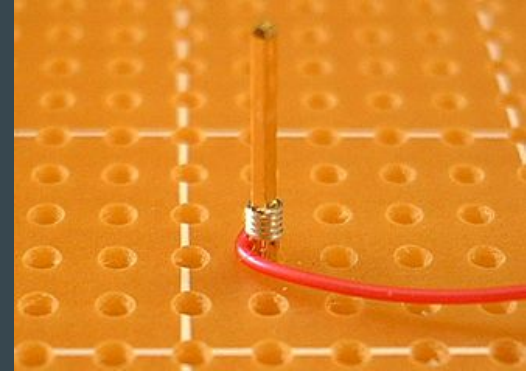


# Wire wrap

Quick, safe way how to connect pins and wires

Wire Wrapping for our Projects by Andreas Spiess:

<https://youtu.be/L-463vchW0o?si=U96aLAwlsLpsPGTW>



picture from Wikipedia: [https://en.wikipedia.org/wiki/Wire\\_wrap](https://en.wikipedia.org/wiki/Wire_wrap)

# LVGL and Rust



[https://github.com/lvgl/lv\\_binding\\_rust](https://github.com/lvgl/lv_binding_rust)

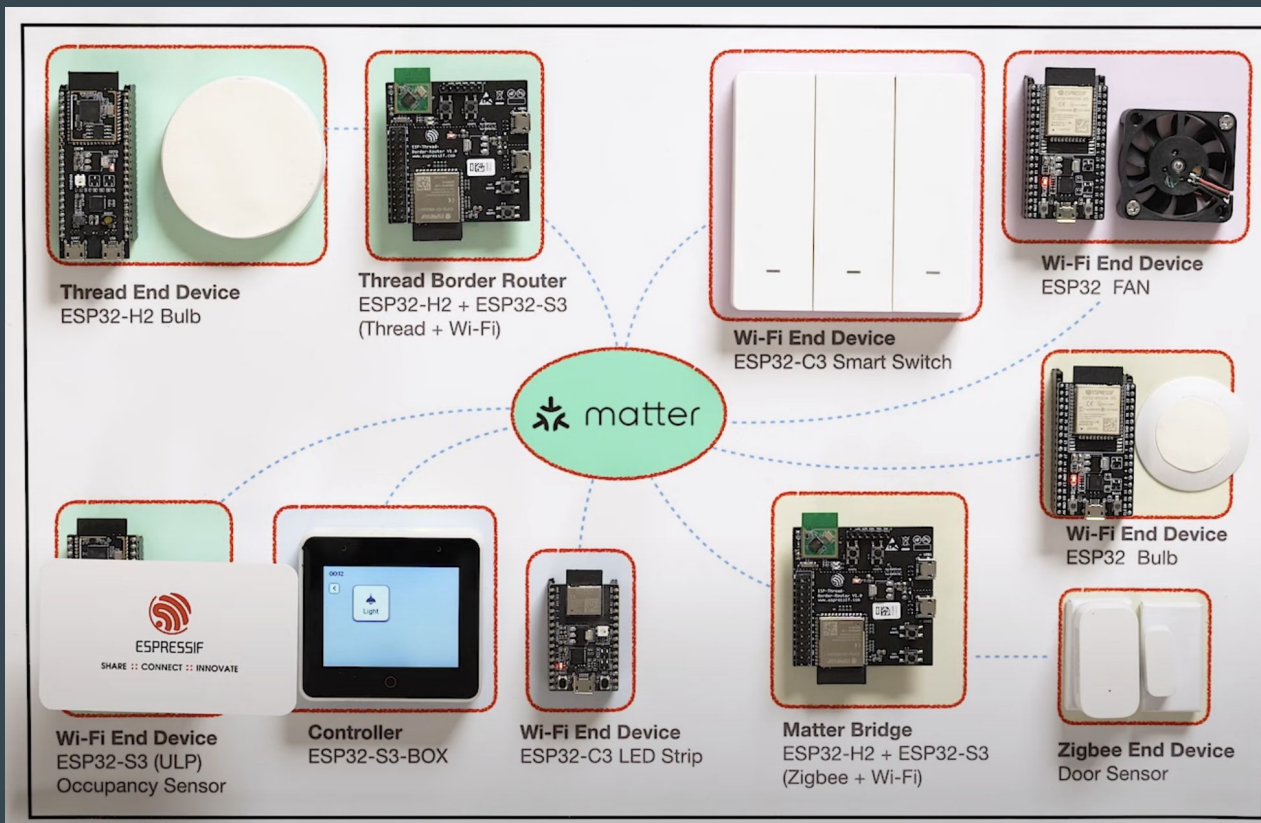
# Slint UI

<https://slint.dev/>

2 options:

- bare metal
- ESP-IDF component: <https://components.espressif.com/components/slnt/slnt>





Espressif's Matter Demo - <https://youtu.be/Jr4LutNgqA>

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

# Espressif and Matter



Support for ESP32 is upstream

<https://github.com/project-chip/connectedhomeip/tree/master/examples/all-clusters-app/esp32>

esp-matter project - <https://github.com/espressif/esp-matter>

rs-matter - Rust - <https://github.com/project-chip/rs-matter>

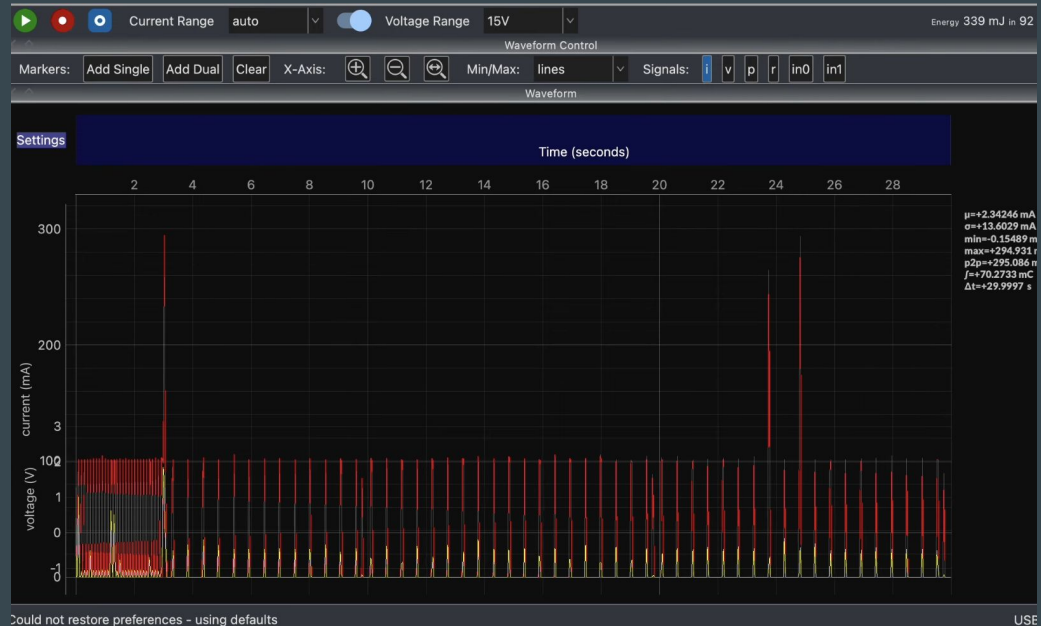
# ESP32-C6

Main feature: WiFi 6 support - reduced power consumption

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

Second important feature:

- Low Power Core - 20 MHz





# Wokwi and Low Power Core simulation ESP32-C6

<https://github.com/wokwi/esp32c6-i2c-lp>

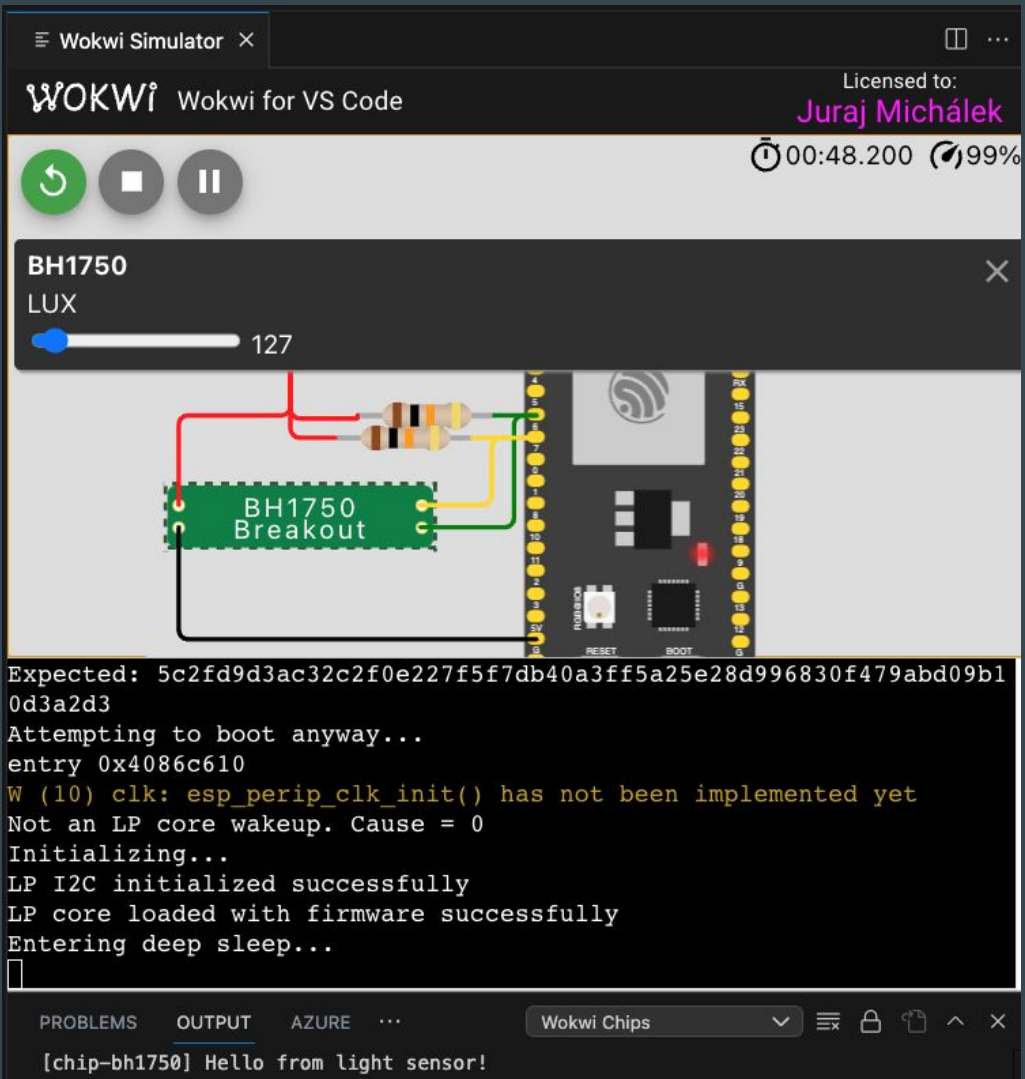
Wokwi Simulator

WOKWI Wokwi for VS Code

Licensed to: **Juraj Michálek**

00:48.200 99%

BH1750  
LUX  
127



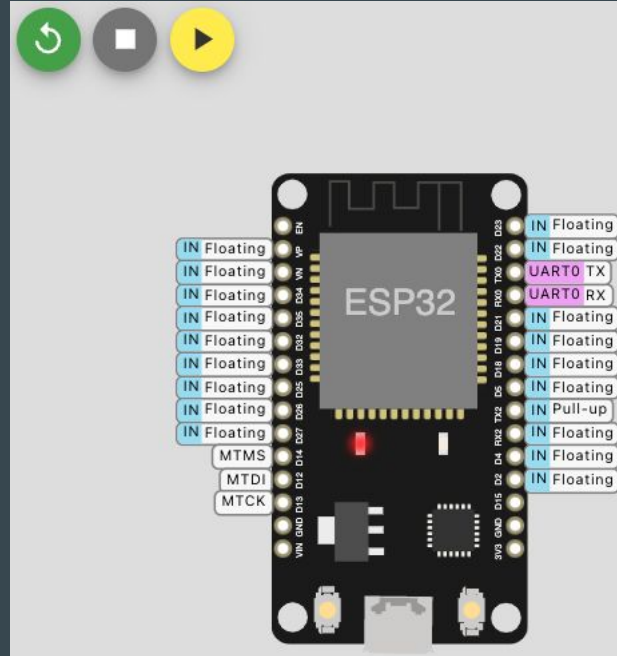
```
Expected: 5c2fd9d3ac32c2f0e227f5f7db40a3ff5a25e28d996830f479abd09b10d3a2d3
Attempting to boot anyway...
entry 0x4086c610
W (10) clk: esp_perip_clk_init() has not been implemented yet
Not an LP core wakeup. Cause = 0
Initializing...
LP I2C initialized successfully
LP core loaded with firmware successfully
Entering deep sleep...

```

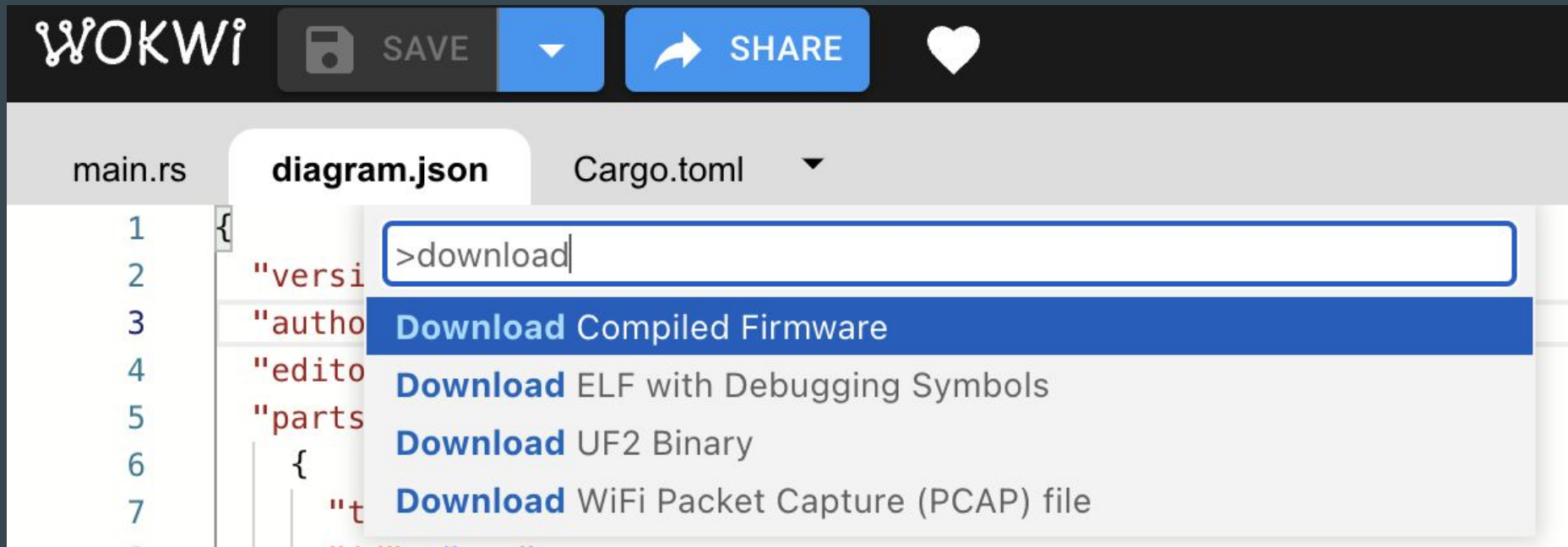
PROBLEMS OUTPUT AZURE ... Wokwi Chips

[chip-bh1750] Hello from light sensor!

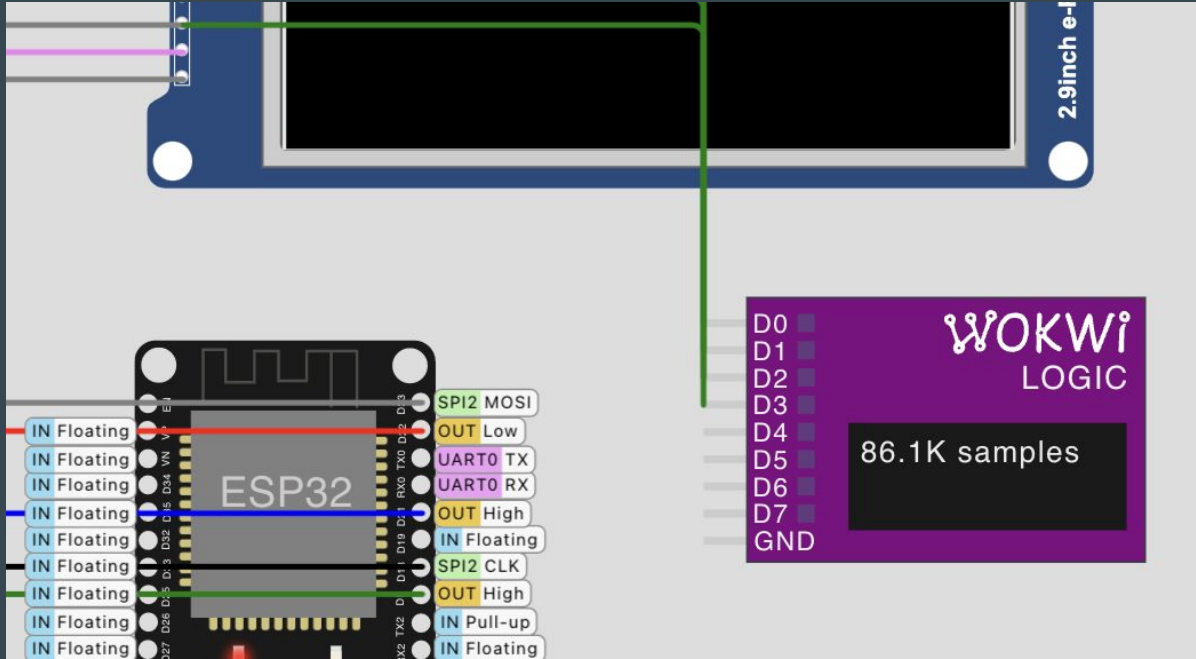
# Pause button - see state of GPIOs



# Wokwi F1 menu (download binary and many more)

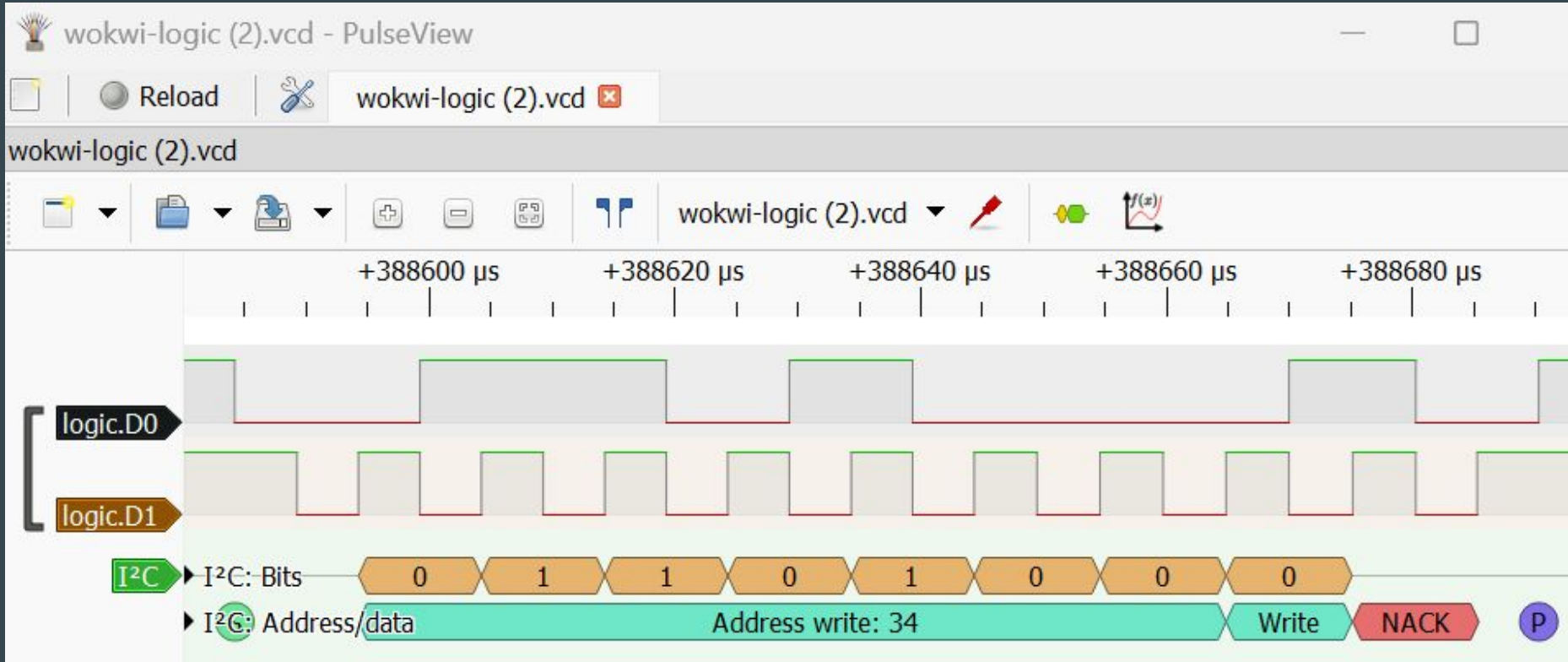


# Wokwi Logic Analyzer



**Hit Stop to download VCD file for PulseView**

# Wokwi + PulseView - examples of I2C



# Custom chips: Rust + e-Paper + ESP32

Simulate your own HW

Wokwi: Getting Started with the Wokwi Custom Chips C API

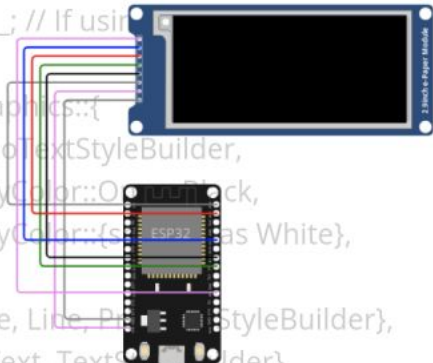
<https://docs.wokwi.com/chips-api/getting-started>

<https://wokwi.com/projects/366167936725307393>

main.rs

```
use esp_idf_sys as _; // If using the `esp-idf` toolchain, alw

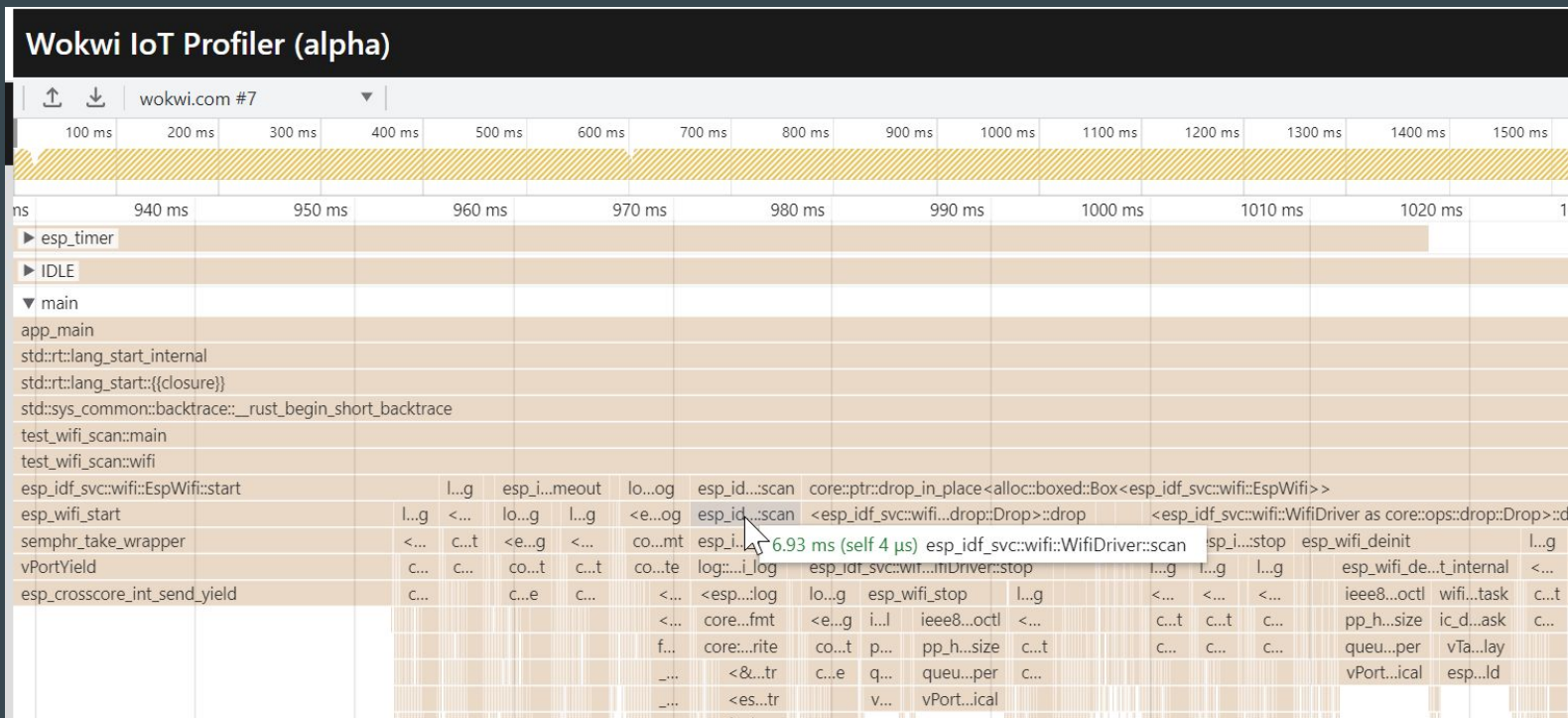
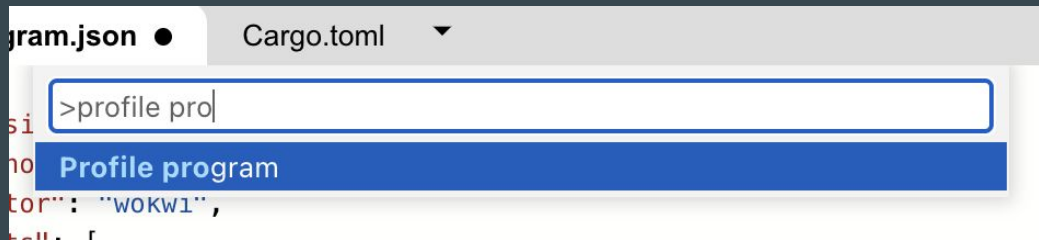
use embedded_graphics::{
    mono_font::MonoTextStyleBuilder,
    pixelcolor::BinaryColor::On as Black,
    pixelcolor::BinaryColor::Off as White,
    prelude::*,
    primitives::{Circle, Line, Point, StyleBuilder},
    text::{Baseline, Text, TextStyleBuilder},
};
```



WOKWI

# Wokwi Profiler

<https://profiler.wokwi.com/>



# Wokwi CI

<https://github.com/wokwi/wokwi-ci-action>



# Books

## The Rust on ESP Book

- <https://esp-rs.github.io/book/>

# Rust STD Training Embedded for ESP32-C3

Material: <https://esp-rs.github.io/std-training/>

GitHub: <https://github.com/esp-rs/espressif-trainings>

Developed by Ferrous Systems, Espressif Systems and Community



# Async with Embassy

Embassy: <https://github.com/embassy-rs/embassy>

Examples of Embassy on ESP32:

- [https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy\\_hello\\_world.rs](https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy_hello_world.rs)
- [https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy\\_spi.rs](https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy_spi.rs)
- [https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy\\_wait.rs](https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy_wait.rs)

# Accelerated learning with AI

Rust has steep learning curve

Refactoring is 10x harder

Very good results: Chat GPT 4, GitHub CoPilot

Less accurate results: Chat GPT 3.5, Bard



# Other languages and frameworks in context of ESP32

New and noteworthy:


- [Ada/Spark](#) - from AdaCore
- [Embedded Wizard](#) - DSL and C
- [Zig](#)

VM based:


- [CircuitPython](#) and [MicroPython](#) - Python-like language
- [DeviceScript](#) - TypeScript language - Microsoft Research
- [Lua](#)
- [Mongoose OS](#)
- [Nanoframework](#) - C# language
- [Toit](#)
- downside: bigger VM
- upside: more robust, comes with OTA and monitoring


# Many chips, many boards - quick help


<https://products.espressif.com/>



ESP32-S3

 Product Brief

 Docs & Certs

 DevKits


ESP32-S3 is a low-power MCU-based SoC that supports 2.4 GHz Wi-Fi and Bluetooth® Low Energy (Bluetooth LE).

ESP32-S3 has a complete Wi-Fi subsystem and a Bluetooth LE subsystem, State-of-the-art power and RF performance. S3 provides a rich set of peripheral interfaces, and supports ultra-low-power applications. Different security features allow the device to meet stringent security requirements.


**Features:**


- **Core:** Xtensa® single-dual 32-bit LX7 CPU, frequency up to 240MHz
- **Memories:**


**Block Diagram:**




List: 203 items

 IC/Module

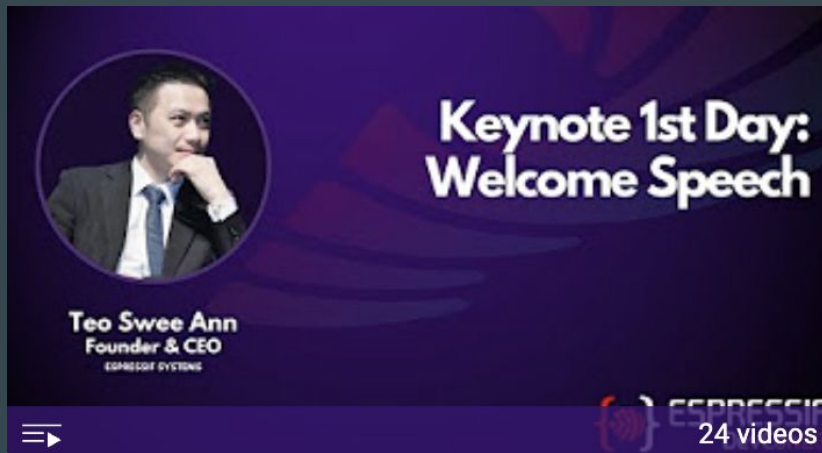
 Development Board

 Comparison

 Export

| <input type="checkbox"/> | Index | Name     | MPN          | Marketing Status | Type | Wi-Fi                      |
|--------------------------|-------|----------|--------------|------------------|------|----------------------------|
| <input type="checkbox"/> | 1     | ESP32-S3 | ESP32-S3     | Mass Production  | SoC  | IEEE 802.11 b/g/n; 2.4 ... |
| <input type="checkbox"/> | 2     | ESP32-S3 | ESP32-S3R2   | Mass Production  | SoC  | IEEE 802.11 b/g/n; 2.4 ... |
| <input type="checkbox"/> | 3     | ESP32-S3 | ESP32-S3R8   | Mass Production  | SoC  | IEEE 802.11 b/g/n; 2.4 ... |
| <input type="checkbox"/> | 4     | ESP32-S3 | ESP32-S3R... | Mass Production  | SoC  | IEEE 802.11 b/g/n; 2.4 ... |
| <input type="checkbox"/> | 5     | ESP32-S3 | ESP32-S3F... | Mass Production  | SoC  | IEEE 802.11 b/g/n; 2.4 ... |

# Espressif Developer Conference 2023 - recording



Developer Conference 2023 - <https://devcon.espressif.com/>

<https://youtu.be/mR3gUNXMEsM?si=7Rkq7wbCICvnM3pw>

# Rust language support

Talk: EDC23 - Rust Bare-metal and Async - Scott Mabin, Juraj Sadel - DevConf September 2023

<https://youtu.be/QPp4WEjx5jU?si=zJwXPT8WxXOFq8oy>

Talk: EDC22 - Rust on Espressif chips - Scott Mabin - DevConf September 2022

<https://youtu.be/qeEmJ-6fPg>

Talk: Embedded Rust on ESP32 - Juraj Michálek - Rust Linz November 2022

<https://youtu.be/OPPPdgoDBQs>





# Embedded World 2024

Meet us in Nuremberg, Germany



**embeddedworld**

Exhibition&Conference

# Visit us in Brno

Espressif Systems (Czech) s.r.o.

Přízova 3, 602 00 Brno

Czechia, Europe

