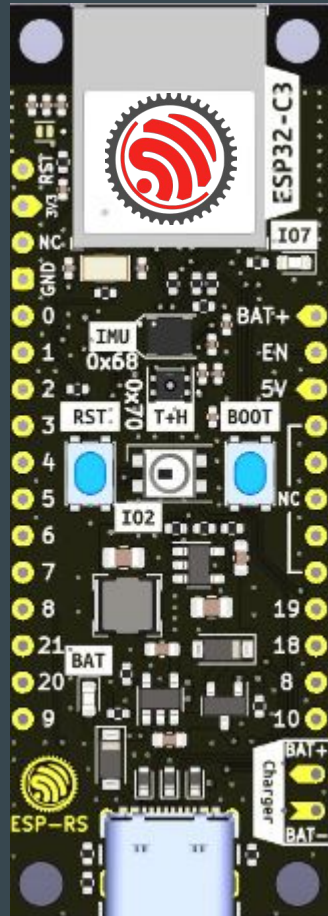


Faster Rust Embedded development with Wokwi for ESP32

2023-06-03
AeroRust - Plovdiv

Juraj Michálek - Espressif Systems



Espressif System

Espressif Systems (688018.SH) is a public multinational, fabless semiconductor company established in 2008, with offices in China, the Czech Republic, India, Singapore and Brazil.

ESP8266 - WiFi

ESP32, ESP32-S - Xtensa cores

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

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



Espressif in Brno

Near Main train station

Espressif Systems (Czech) s.r.o.


Přízova 3, 602 00 Brno

Czechia, Europe





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 ...

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
 - recommended for new design evaluation - <https://github.com/esp-rs>
 - security and memory guaranties of Rust
 - multi-target 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

Other languages and frameworks

New and noteworthy:

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

VM based:

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

IDE

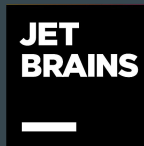
Supported by Espressif:

- VS Code with Espressif Extension - **great Rust support**
<https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/vscode-setup.html>
- Espressif IDE - <https://dl.espressif.com/dl/esp-idf/>



Supported by JetBrains:

- CLion - <https://www.jetbrains.com/clion/> - **great Rust Support**



Supported by SysProgs

- Visual Studio with VisualGDB - <https://visualgdb.com/>



Supported by TARA Systems

- Embedded Wizard - <https://www.embedded-wizard.de/>



GUI Solutions by TARA Systems

Rust language support

Talk: EDC22 Day 1 Talk 7: 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

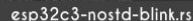
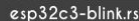
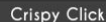
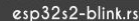
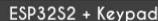
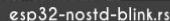
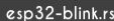
<https://youtu.be/OPPPdgoDBQs>



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

<https://youtu.be/TKe4MgD608o>

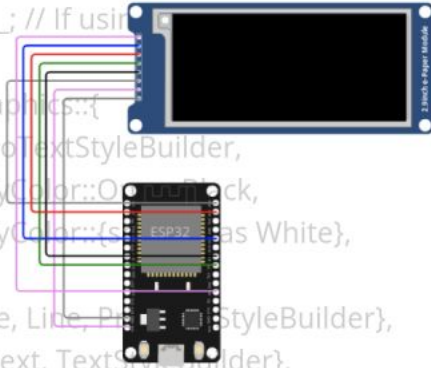
[+ NEW PROJECT](#)



Rust + e-Paper + ESP32

main.rs

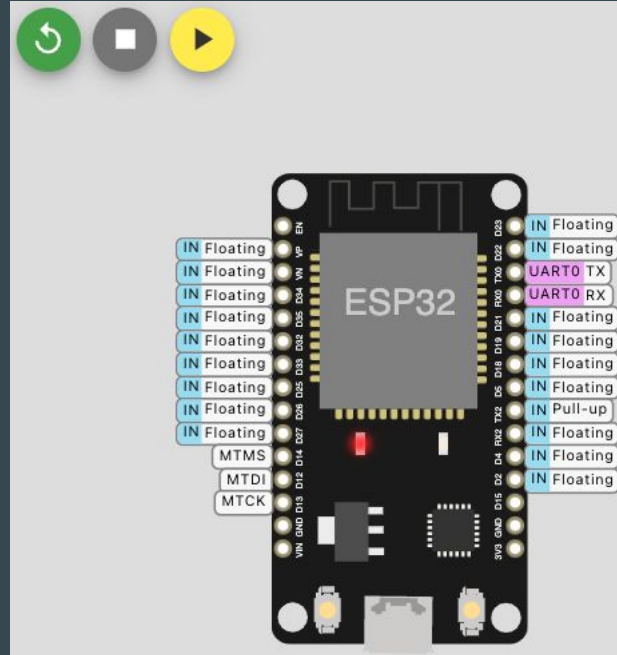
```
use esp_idf_sys as _; // If using the `esp-idf` toolchain, alw  
  
use embedded_graphics::{  
    mono_font::MonoTextStyleBuilder,  
    pixelcolor::BinaryColor::On as Black,  
    pixelcolor::BinaryColor::Off as White,  
    prelude::*,  
    primitives::{Circle, Line, Point, Rectangle, StyleBuilder},  
    text::{Baseline, Text, TextStyleBuilder},  
};
```



WOKWI

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

Pause button - see state of GPIOs



Wokwi F1 menu (download binary and many more)

The screenshot shows the Wokwi web interface. At the top, there is a dark navigation bar with the Wokwi logo, a 'SAVE' button with a floppy disk icon, a dropdown arrow, a 'SHARE' button with a share icon, and a heart icon. Below this, a light gray bar contains tabs for 'main.rs', 'diagram.json' (which is active), and 'Cargo.toml' with a dropdown arrow. The main area shows a code editor with line numbers 1 through 7. A context menu is open over the code, triggered by the F1 key. The menu has a search bar containing '>download|'. Below the search bar, there are four options, each starting with 'Download' in blue text: 'Download Compiled Firmware' (highlighted with a blue background), 'Download ELF with Debugging Symbols', 'Download UF2 Binary', and 'Download WiFi Packet Capture (PCAP) file'.

WOKWI

SAVE

SHARE

main.rs

diagram.json

Cargo.toml

1 {

2 "versi

3 "autho

4 "edito

5 "parts

6 {

7 "t

>download|

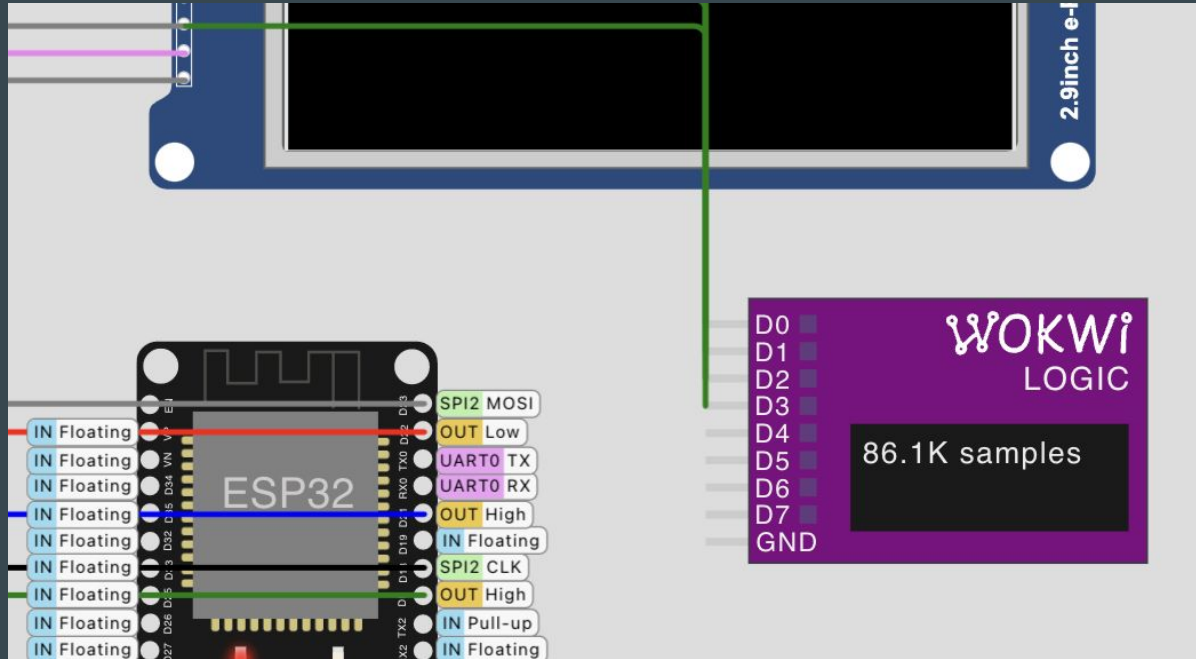
Download Compiled Firmware

Download ELF with Debugging Symbols

Download UF2 Binary

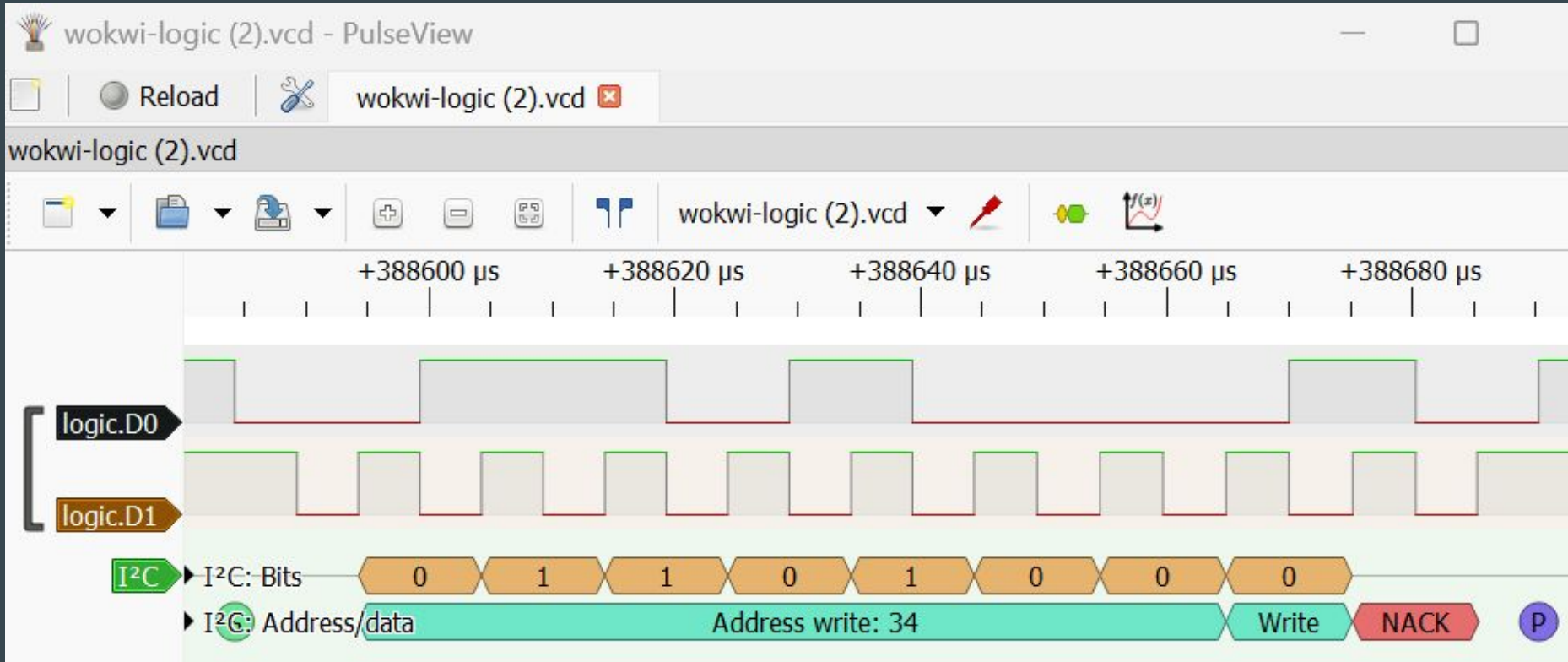
Download WiFi Packet Capture (PCAP) file

Wokwi Logic Analyzer



Hit Stop to download VCD file for PulseView

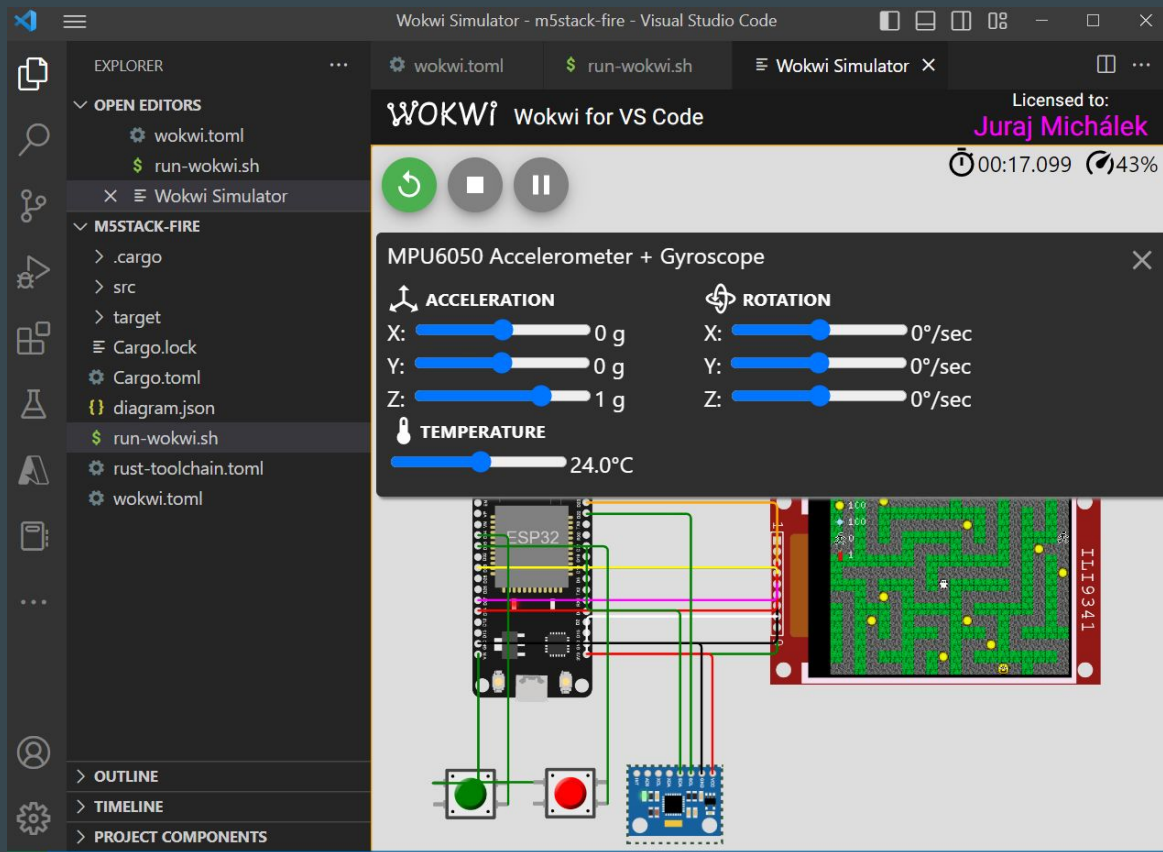
Wokwi + PulseView - examples of I2C



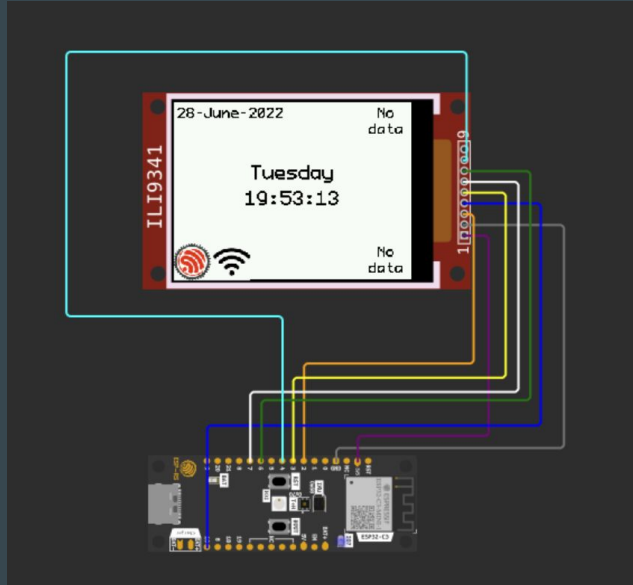
Wokwi - VS Code Plugin

Add wokwi.toml and diagram.json
to your project

CTRL+Shift+P - Wokwi: Start Simulator



Development containers and Wokwi



<https://github.com/playfulFence/esp-clock#dev-containers>

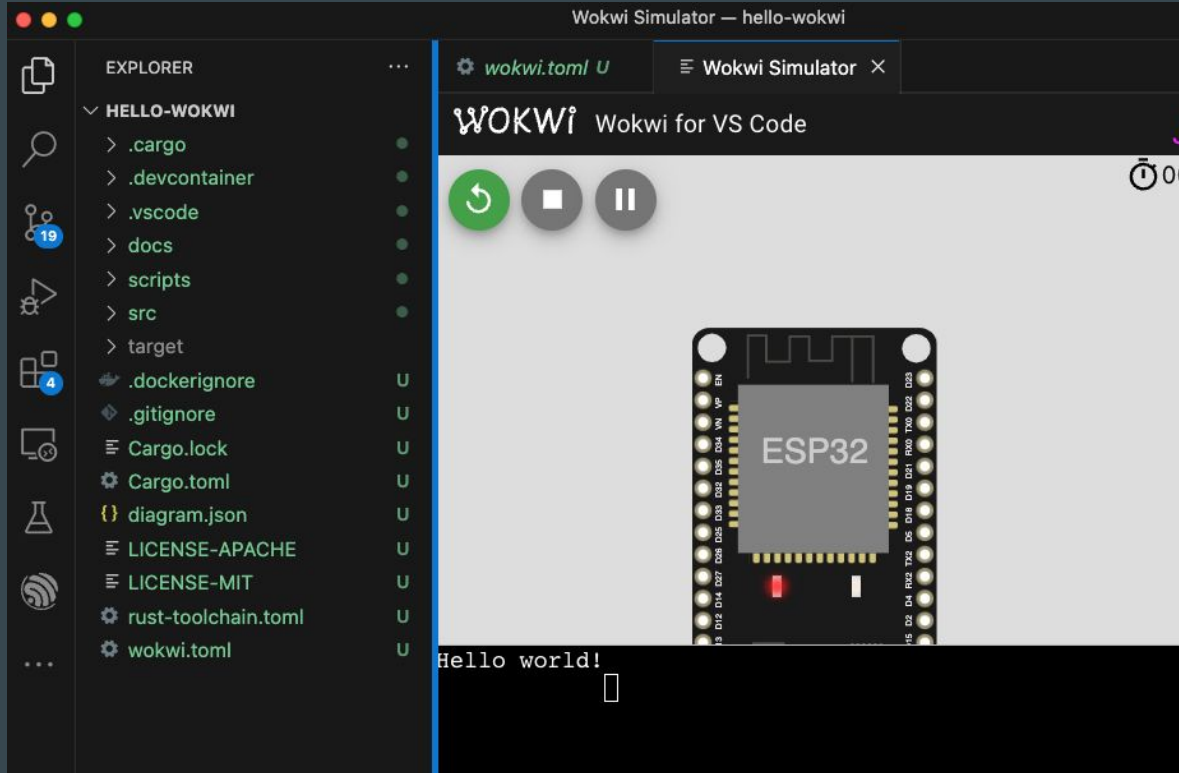
Starting from template

```
cargo generate -a esp-rs/esp-template
```

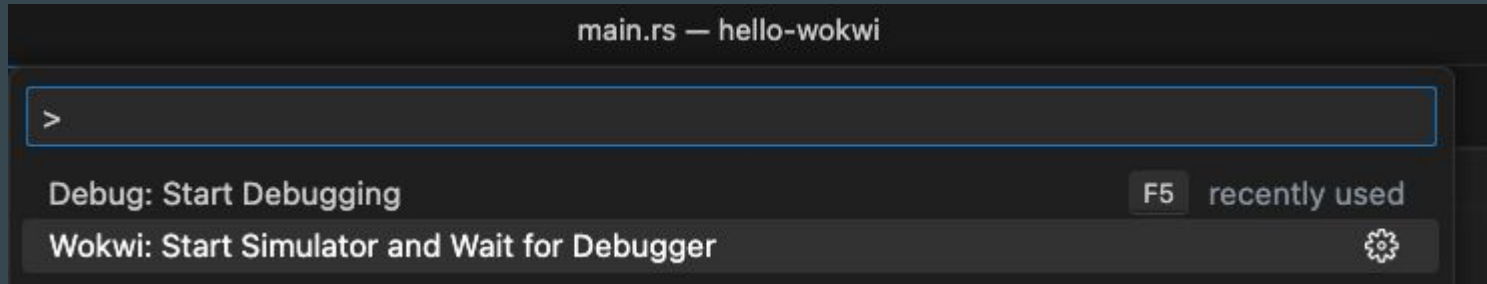
```
👤 Project Name: hello-wokwi
🔧 Destination: /Users/georgik/projects/aero-rust/hello-wokwi ...
🔧 project-name: hello-wokwi ...
🔧 Generating template ...
✅👤 Use template default values? · false
✅👤 Which MCU to target? · esp32
?👤 Configure project to support Wokwi simulation with Wokwi VS Code extensio
✅👤 Configure project to support Wokwi simulation with Wokwi VS Code extension? · true
?👤 Configure project to use Dev Containers (VS Code and GitHub Codespaces)?
✅👤 Configure project to use Dev Containers (VS Code and GitHub Codespaces)? · true
✅👤 Enable allocations via the esp-alloc crate? · true
```

```
cargo build
```

Command/CTRL+Shift+P - Wokwi: Start Simulator

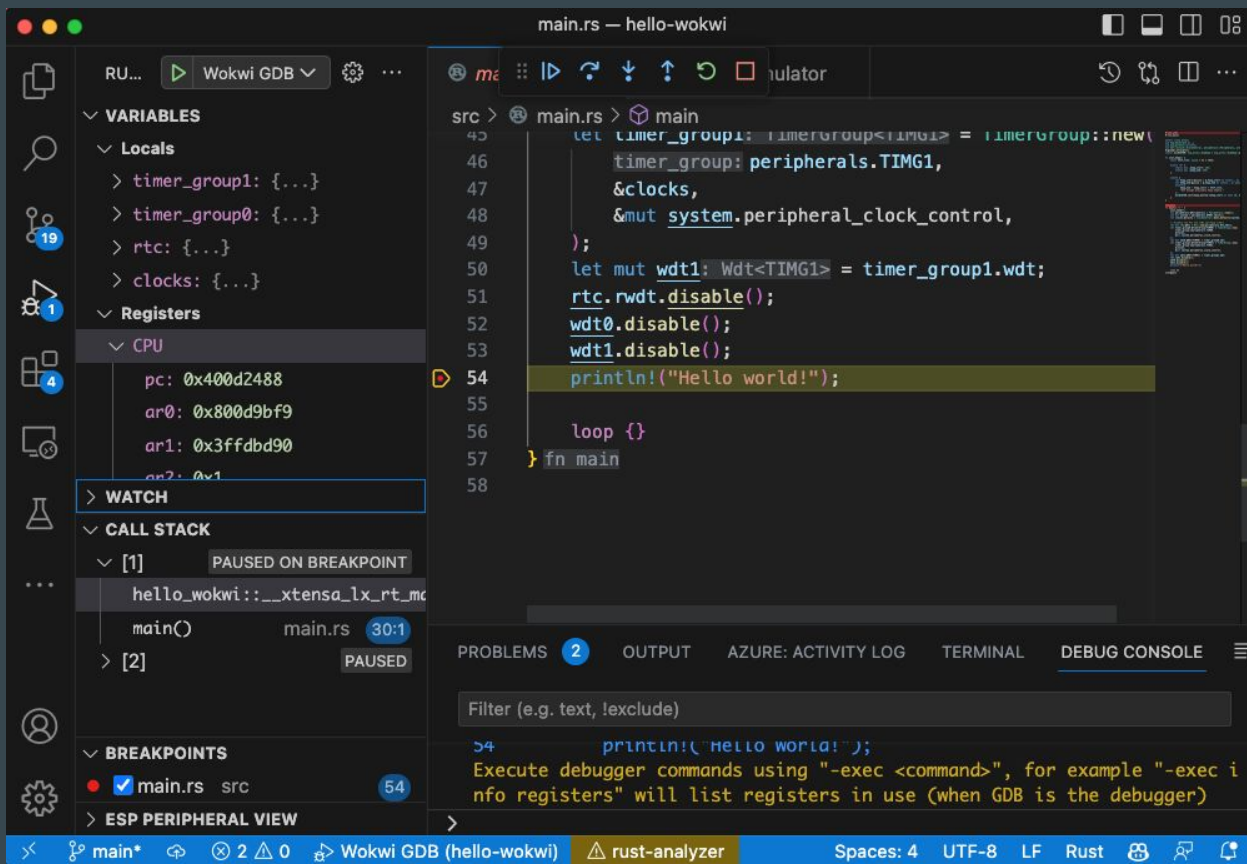


Debugging



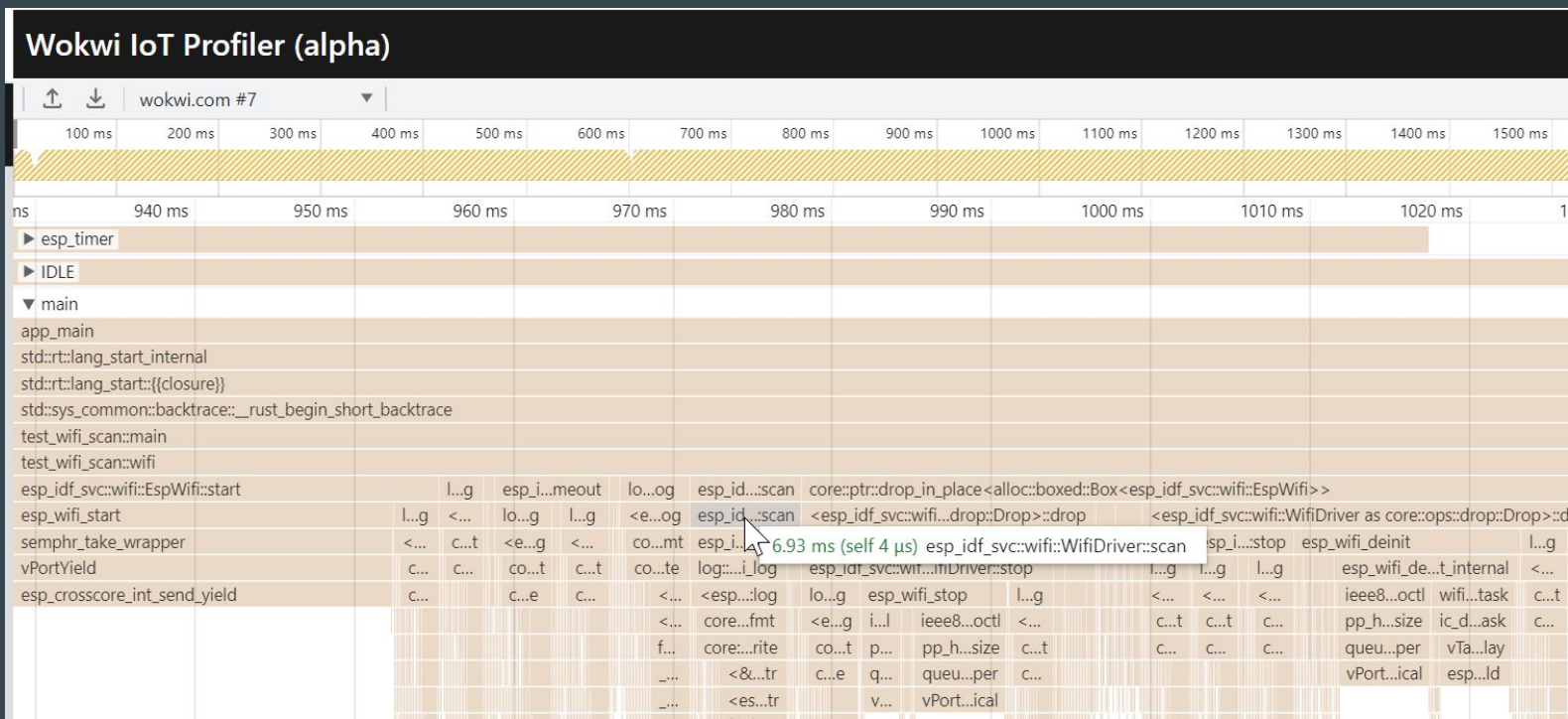
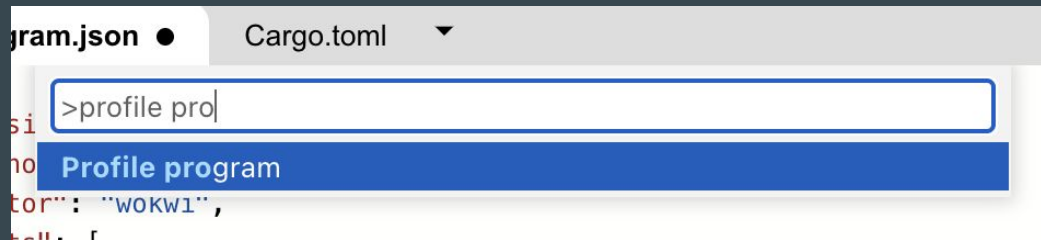
2 steps: simulator, debugger

Debugging session with Wokwi



Wokwi Profiler

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



Wokwi CI

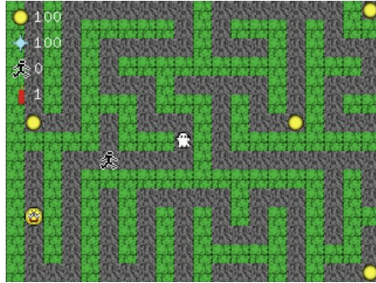
<https://github.com/wokwi/wokwi-ci-action>

Wokwi and CI with screenshot

<https://github.com/georgik/esp32-spooky-maze-game/actions/runs/5111553942>

deploy summary

LCD screen shot (simulated with Wokwi)



Job summary generated at run-time

Wokwi custom chips

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

Implement your own virtual HW.

ESP32-C6

Main feature: WiFi 6 support - reduced power consumption

- <https://www.youtube.com/watch?v=FA1iqZLig4s>

Second important feature:

- Low Power Core - 20 MHz



Wokwi and Low Power Core simulation ESP32-C6

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

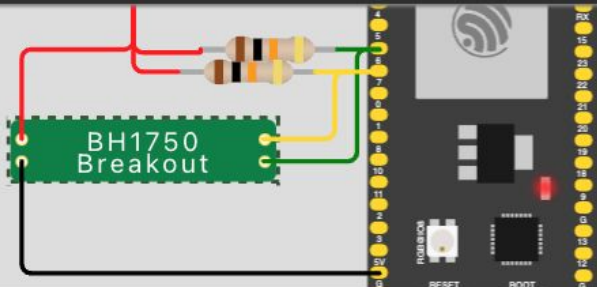
Wokwi Simulator

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...
[chip-bh1750] Hello from light sensor!

PROBLEMS OUTPUT AZURE ... Wokwi Chips

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



ferrous systems

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_spi.rs
- https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy_wait.rs

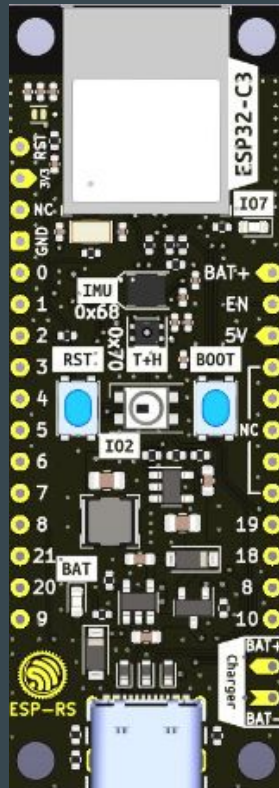
Designing Open Hardware - esp-rust-board

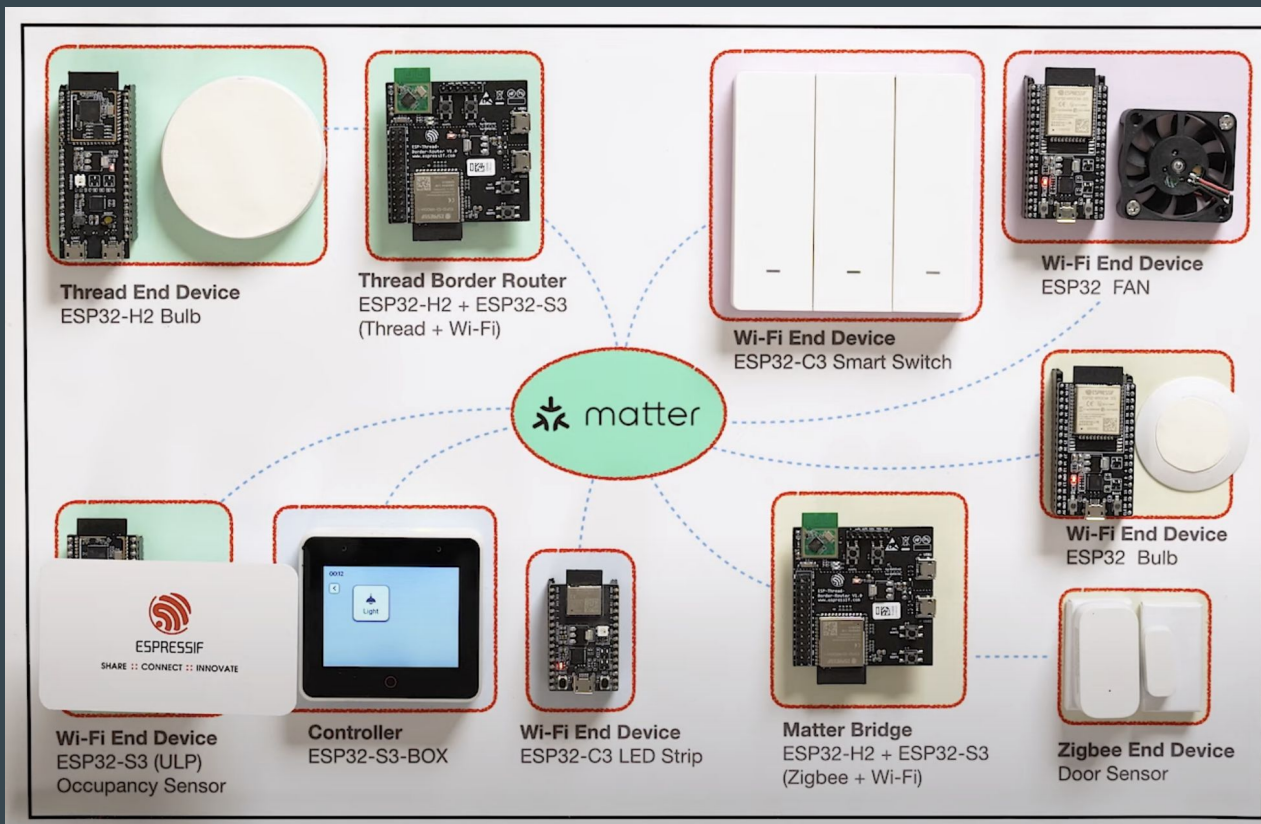
KiCad templates

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

ESP32-C3-DevKit-RUST-1 (available at Mouser, AliExpress)

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





Espressif's Matter Demo - [https://youtu.be/Jr4Lut NgqA](https://youtu.be/Jr4LutNgqA)

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>

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

ESP Launchpad - <https://espressif.github.io/esp-launchpad/>



Quick Start

DIY

Connect

Console

Settings

About

ESP Launchpad helps you to flash the selected firmware image onto your device.

Ensure you have connected your device to the serial USB port. Click on the 'Connect' button in the top menu option, to connect to your attached device.

Note : Once you flash your device, any earlier firmware would be overwritten.

Choose from some of ESP's pre-built, out-of-the-box examples to flash and play

Select Application

Matter-AllClustersApp-TE9

ESP Chipset Type

Flash

Matter-AllClustersApp-TE9

Matter-AllClustersApp-M5Stack-TE9

Matter-LightingApp-TE9

RainMaker-Fan

RainMaker-GPIO

RainMaker-HomekitSwitch

RainMaker-LedLight

RainMaker-MultiDevice

RainMaker-Switch

RainMaker-TemperatureSensor

Multi-target project (PoC) - ESP32 Spooky Maze

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

Idea: sharing business logic in Rust between
multiple targets

Demo: <https://georgik.rocks/rust-bare-metal-application-for-esp32-desktop-android-and-ios/>

Targets: ESP32, ESP32-S2, ESP32-S3, ESP32-C3, M5Stack, Wasm and Desktop

Article: <https://georgik.rocks/rust-bare-metal-application-for-esp32-desktop-android-and-ios/>



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

Espressif Developer Conference 2022 - recording



https://www.youtube.com/playlist?list=PLOzvoM7_Knrc6o-n25jYuXRB2T8UKk1NU

Developer Conference 2023 - scheduled for 12-13 September - <https://devcon.espressif.com/>

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

