

# How to write apps or games in Rust for ESP32, web, desktop or mobile?

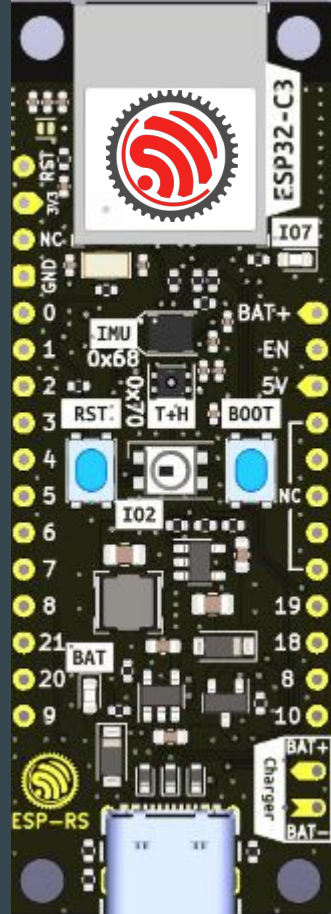


2024-04-17

TechMeetup

Impact Hub - Ostrava

Juraj Michálek - Espressif Systems



# 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

```
#![no_std]
#![no_main]

// This assumes that a LED is connected
// Depending on your target, you may need to change the LED pin
// If your board doesn't have a LED, you may want to add an approx
// pin to the board

use esp32::hal::digital::blocking::OutputPin;
use esp32::hal::timer::Duration;

// esp32-blink.rs
```




```
#![no_std]
#![no_main]

// This assumes that a LED is connected
// Depending on your target, you may need to change the LED pin
// If your board doesn't have a LED, you may want to add an approx
// pin to the board

use esp32s2::hal::digital::blocking::OutputPin;
use esp32s2::hal::timer::Duration;

// esp32s2-blink.rs
```



```
#![no_std]
#![no_main]

// This assumes that a LED is connected to GPIO4
// Depending on your target, you may need to change the LED pin
// If your board doesn't have a LED, you may want to add an approx
// pin to the board

use esp32c3::hal::digital::blocking::OutputPin;
use esp32c3::hal::timer::Duration;

// esp32c3-blink.rs
```



```
#![no_std]
#![no_main]

use esp32::hal::digital::blocking::OutputPin;
use esp32::hal::timer::Duration;
use esp32::peripherals::TimerGroup;


// esp32-nostd-blink.rs
```



```
#![no_std]
#![no_main]

use esp32s2::hal::digital::blocking::OutputPin;
use esp32s2::hal::timer::Duration;
use esp32s2::peripherals::TimerGroup;

// esp32s2-nostd-blink.rs
```



```
#![no_std]
#![no_main]


use esp32c3::hal::digital::blocking::OutputPin;
use esp32c3::hal::timer::Duration;
use esp32c3::peripherals::TimerGroup;

// esp32c3-nostd-blink.rs
```




```
use esp32::clock::Clock;
use esp32::timer::TimerGroup;

// esp-clock (WiFi)
```




```
use esp32s2::clock::Clock;
use esp32s2::timer::TimerGroup;

// Crispy Click
```




```
use esp32c3::clock::Clock;
use esp32c3::timer::TimerGroup;

// ESP32C3 + ILI9341 Display
```




```
use esp32s3::hal::digital::blocking::OutputPin;
use esp32s3::peripherals::TimerGroup;

// ESP32S3 + ILI9341 Display
```




```
use esp32::hal::digital::blocking::OutputPin;
use esp32::peripherals::TimerGroup;

// ESP32 + ILI9341 Display
```



```
use esp32::hal::digital::blocking::OutputPin;
use esp32::peripherals::TimerGroup;

// ESP32 + 8x8 LED Dot Matrix
```




```
use esp32s2::hal::digital::blocking::OutputPin;
use esp32s2::peripherals::TimerGroup;

// ESP32S2 + Keypad
```




```
use esp32s2::hal::digital::blocking::OutputPin;
use esp32s2::peripherals::TimerGroup;

// Joystick Etch-a-Sketch
```



```
use esp32c3::hal::digital::blocking::OutputPin;
use esp32c3::peripherals::TimerGroup;

// esp-gallery
```



SPECIAL EDITION

140 Pages

Special edition guest-edited by



ESPRESSIF

Prototyping With Espressif Chips

ADF, IDF, and Other SDKs



Insights from Espressif Engineers

Automation With Rainmaker and Matter

Trying Out the ESP32-S3-BOX-3



ESP32 and ChatGPT

In this issue

- > An Open-Source Speech Recognition Server
- > Power Duct: Rust + Embedded
- > Facial Recognition With ESP32-S3-EYE
- > Walkie-Talkie with ESP-NOW
- > Another Elektor Christmas Tree Project

and much more!



Unleashing the ESP32-P4 The Next Era of Innovative Microcontrollers

p. 59



Acoustic Fingerprinting Song Recognition With ESP32

p. 80



A Vision for the AIoT Interview with Espressif CEO Teo Swee-Ann

p. 35



# Elektor Mag

## Special Guest Edition

<https://www.elektor.com/products/elektor-special-esspressif-guest-edition-2023-pdf-en>

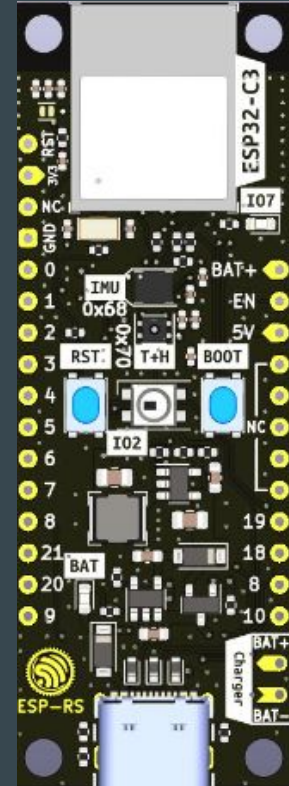
# Designing Open Hardware - esp-rust-board

KiCad templates

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

ESP32-C3-DevKit-RUST-1

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



# Open Hardware - ESP32-S3-BOX-3

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

Joystick controller

[https://github.com/espressif/esp-box/tree/master/examples/esp\\_joystick/joystick\\_controller](https://github.com/espressif/esp-box/tree/master/examples/esp_joystick/joystick_controller)



# 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++ - 229 repositories at <https://github.com/espressif>

- 12 200+ starts - ESP-IDF
- Espressif IoT Development Framework (CMake based)

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

- 500+ starts - esp-hal

Arduino - <https://github.com/espressif/arduino-esp32>

- 12 300+ stars





# Desktop vs. Embedded development

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

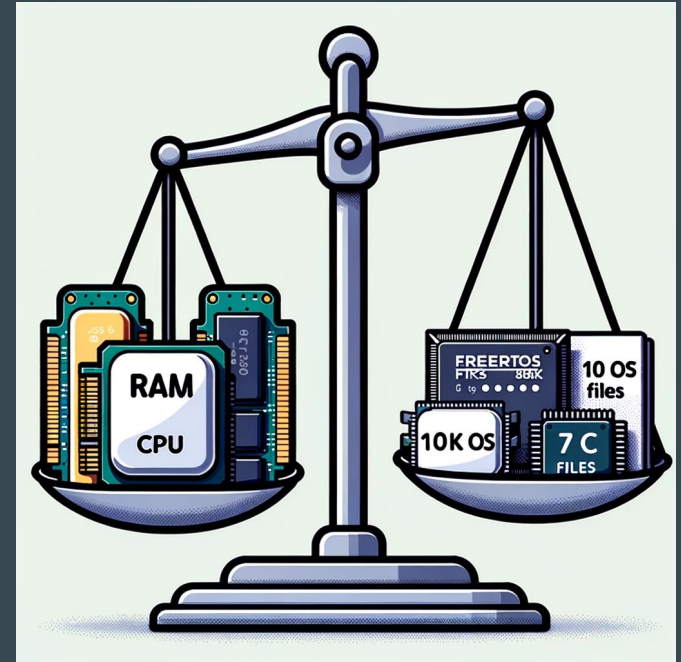
CPU Cores: **8 - 64** vs. **1 - 2**

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)





# ESP32 Lang lab

<https://github.com/georgik/esp32-lang-lab>

E.g. ESP32-C3 - 320 kB

# Rust Embedded Graphics

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

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



# Rust Bare Metal (no\_std) Projects - Case studies

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

- 2D game

RustZX-ESP32 - <https://github.com/georgik/rustzx-esp32>

- ZX Spectrum Emulator

Slint UI - [https://github.com/slnt-ui/slnt/tree/master/examples/printerdemo\\_mcu/esp-idf](https://github.com/slnt-ui/slnt/tree/master/examples/printerdemo_mcu/esp-idf)

- Printer MCU (no\_std / std)

# Multi-target project (PoC) - ESP32 Spooky Maze

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

Idea: sharing business logic in Rust between  
multiple targets

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>

# Wokwi - VS Code Plugin

Add wokwi.toml and diagram.json  
to your project

Wokwi Simulator - m5stack-fire - Visual Studio Code

Wokwi for VS Code

Licensed to: Juraj Michálek

00:17.099 43%

MPU6050 Accelerometer + Gyroscope

ACCELERATION

ROTATION

X: 0 g X: 0°/sec

Y: 0 g Y: 0°/sec

Z: 1 g Z: 0°/sec

TEMPERATURE

24.0°C

ESP32

II19341

OUTLINE

TIMELINE

PROJECT COMPONENTS

CTRL+Shift+P - Wokwi: Start Simulator

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

# Wokwi - custom chip API - Rust

```
// Wokwi Custom Chips with Rust
use wokwi_chips_api::println;
use wokwi_chips_api::pin::{Pin, PinMode, WatchEdge};

// chipInit() will be called once per chip instance.
#[no_mangle]
pub unsafe extern "C" fn chipInit() {
    println!("Hello from Rust Chip!");

    let pin_in = Pin::new("IN", PinMode::Input);
    let pin_out = Pin::new("OUT", PinMode::Output);
    pin_out.write(!pin_in.read());

    pin_in.watch(WatchEdge::Both, move |_pin, value| {
        pin_out.write(!value);
    });
}
```

<https://gist.github.com/urish/92a0b95022e23cdb846d17824913b6c1>

# RustZX and ESP32-RustZX

From PC

<https://github.com/rustzx/rustzx>

To MCU chip

<https://github.com/georgik/rustzx-esp32>



```
rustzx v0.16.0  
  
10 PRINT "Ahoj OVA"  
20 PRINT "TechMeetup"  
30 >PRINT "RustZX demo"  
  
K
```



# Slint UI

<https://slint.dev/>

2 options:

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

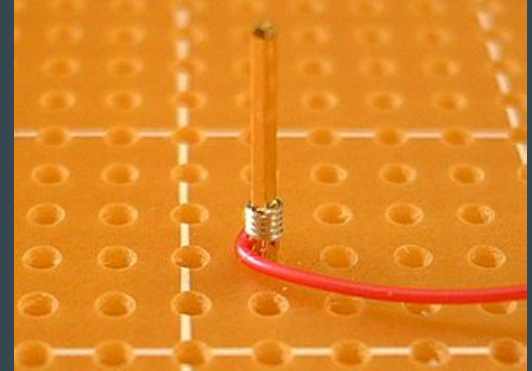


# Wire wrap

Quick, safe way how to connect pins and wires

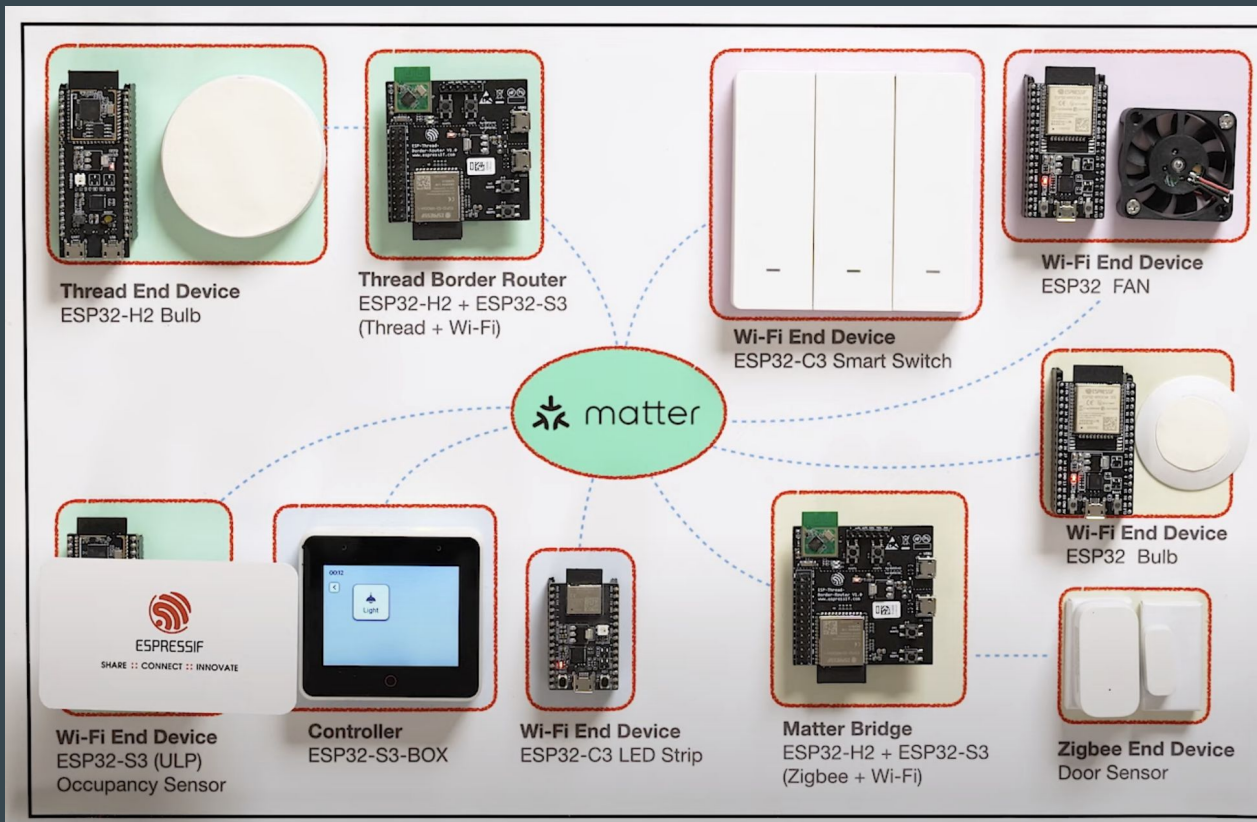
Wire Wrapping for our Projects by Andreas Spiess:

<https://youtu.be/L-463vchW0o?si=U96aLAWlsLpsPGTW>



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





Espressif's Matter Demo - <https://youtu.be/Jr4Lut NgqA>

# 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>

# Many chips, many boards - quick help

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

The screenshot shows the product page for the ESP32-S3. On the left, there is a navigation menu with options: Product Brief, Docs & Certs, and DevKits. The main content area includes a description of the chip as a low-power MCU-based SoC, a list of features (Core: Xtensa® single-dual 32-bit LX7 CPU, frequency up to 240MHz; Memories), and a block diagram. Below this, there are filters for 'IC/Module' and 'Development Board', and buttons for 'Comparison' and 'Export'. A table lists 203 items, with the first five rows visible, showing various ESP32-S3 models and their specifications.

**ESP32-S3**

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 ...

# 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, Rust Rover - <https://www.jetbrains.com/>
- Wokwi Plugin - <https://plugins.jetbrains.com/plugin/23826-wokwi-simulator>

Supported by SysProgs

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

Supported by TARA Systems

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



**JET  
BRAINS**



**SYSPROGS**



**Embedded  
Wizard**

GUI Solutions by TARA Systems



# 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>

# Rainmaker

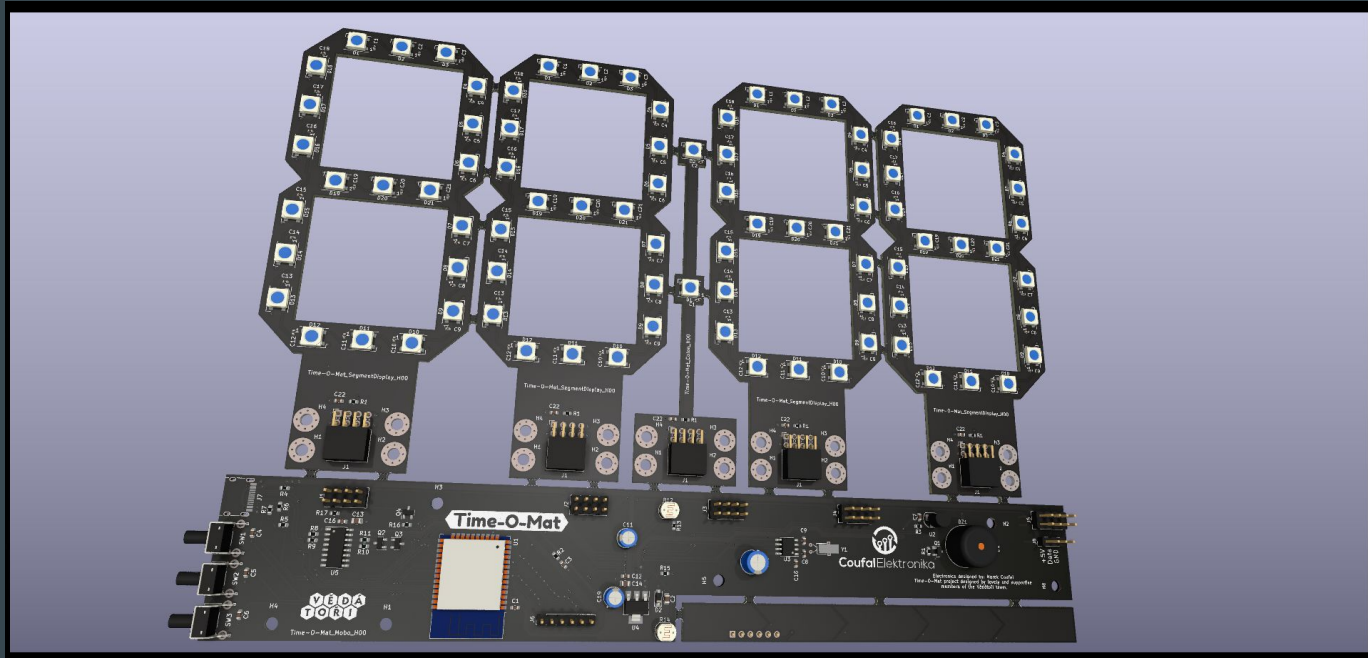
Rust layer - Linux & ESP

<https://github.com/shreyash-b/rainmaker-rs>

Cloud connectivity

# Time-O-Mat - built at summer camp

<https://github.com/vedatori/Time-O-Mat>



# Grafana



<https://grafana.com/blog/2020/06/17/how-to-monitor-a-sourdough-starter-with-grafana/>

<https://github.com/grafana/diy-iot> - Arduino now. We're not Rust yet :)

# Quick start

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

Launchpad for flashing light and light\_switch:

- <https://espressif.github.io/esp-launchpad/?flashConfigURL=https://espressif.github.io/esp-matter/launchpad.toml>

# 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)

Async topic author: Juraj Sadel

# Busy Looping (not using async)

- Very inefficient, blocking

```
fn main() {  
    let mut button = Button::new();  
    let mut led = Led::new();  
    loop {  
        if button.is_pressed() {  
            led.on();  
        } else {  
            led.off();  
        }  
    }  
}
```



# Interrupts (not using async)

- Driven by hardware
- Pretty complex

```
static BUTTON: Mutex<Option<Button>> = Mutex::new(None);
static LED: Mutex<Option<Led>> = Mutex::new(None);
fn main() {
    LED.lock().replace(Led::new());
    BUTTON.lock().replace(Button::new());
    setup_irq(button_event); // Magic
}

fn button_event() {
    if BUTTON.lock().as_mut().unwrap().is_pressed() {
        LED.lock().as_mut().unwrap().on();
    } else {
        LED.lock().as_mut().unwrap().off();
    }
}
```

# Using async

- Don't have to setting and waiting for interrupt resuming the program
- Async executor can do that for us instead
- Power efficient

```
async fn main() {  
    let mut button = Button::new();  
    let mut led = Led::new();  
    loop {  
        button.wait_changed().await;  
        if button.is_pressed() {  
            led.on();  
        } else {  
            led.off();  
        }  
    }  
}
```

# Embassy (EMBedded ASync)

- We need an EXECUTOR to be able to use async
- Controlling which task should run
  
- Embassy consists of multiple crates (Executor, HALs, Networking,...)
- no\_std
- Can be (relatively) easily extendable/configurable with other public crates
- <https://embassy.dev>

# Rust language support

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

<https://youtu.be/OPPPdgoDBQs>



# 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>

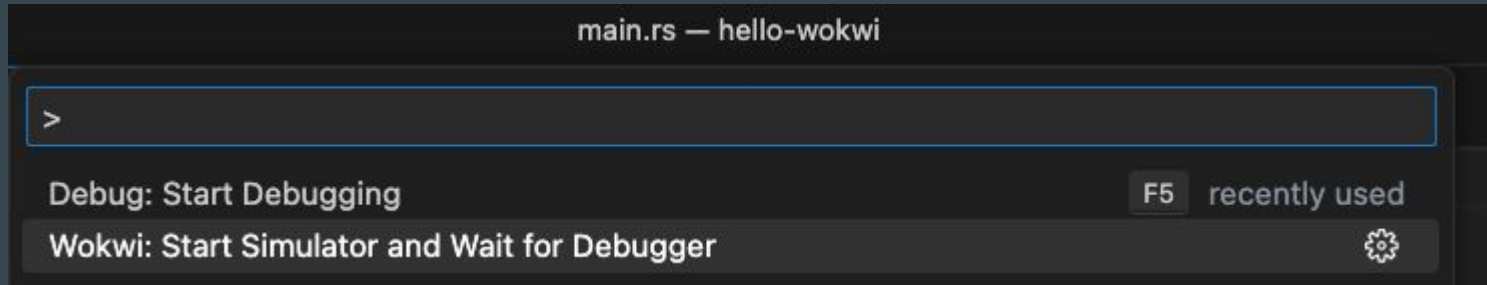
```
idf.py set-target esp32-c3
```

```
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

The screenshot displays the Wokwi IDE interface during a debugging session. The main editor shows the following Rust code:

```
src > main.rs — hello-wokwi
src > main.rs > main
45     let timer_group1: TimerGroup<TIMG1> = TimerGroup::new(
46         timer_group: peripherals.TIMG1,
47         &clocks,
48         &mut system.peripheral_clock_control,
49     );
50     let mut wdt1: Wdt<TIMG1> = timer_group1.wdt;
51     rtc.rwdt.disable();
52     wdt0.disable();
53     wdt1.disable();
54     println!("Hello world!");
55
56     loop {}
57 } fn main
58
```

The left sidebar shows the following panels:

- VARIABLES**
  - Locals
    - timer\_group1: {...}
    - timer\_group0: {...}
    - rtc: {...}
    - clocks: {...}
  - Registers
    - CPU
      - pc: 0x400d2488
      - ar0: 0x800d9bf9
      - ar1: 0x3ffdbd90
      - ar2: 0x1
  - WATCH
  - CALL STACK
    - [1] PAUSED ON BREAKPOINT
      - hello\_wokwi: \_\_xtensa\_lx\_rt\_m...
      - main() main.rs 30:1
    - [2] PAUSED
  - BREAKPOINTS
    - main.rs src 54
  - ESP PERIPHERAL VIEW

The bottom status bar shows: main\* Wokwi GDB (hello-wokwi) rust-analyzer Spaces: 4 UTF-8 LF Rust



# 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
```

# 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

BH1750 Breakout

```
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



# 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 on the left, 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 shows the project name 'diagram.json' and a 'Cargo.toml' dropdown menu. The main area displays a code editor with a file named 'main.rs' open. The code is partially visible, showing lines 1 through 7. A search bar is overlaid on the code, containing the text '>download'. A dropdown menu is open below the search bar, listing several options: 'Download Compiled Firmware' (highlighted in blue), '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

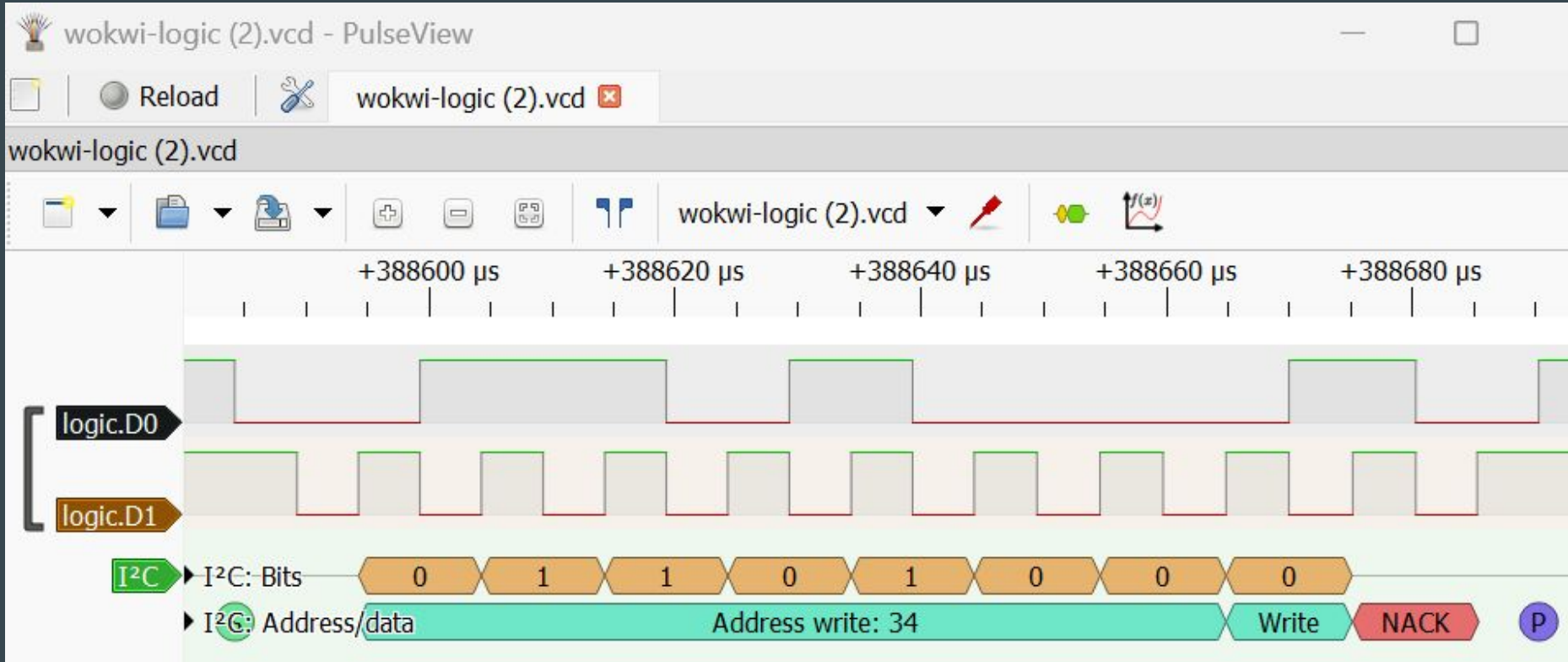
Download UF2 Binary

Download WiFi Packet Capture (PCAP) file





# Wokwi + PulseView - examples of I2C



# Custom chips: Rust + e-Paper + ESP32

Simulate your own HW

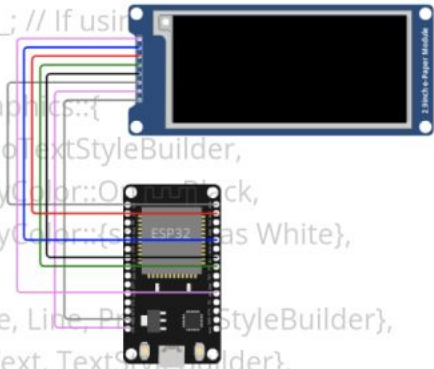
Wokwi: Getting Started with the Wokwi Custom Chips C API

<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, always use `esp-idf-sys`, always  
  
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},  
};
```

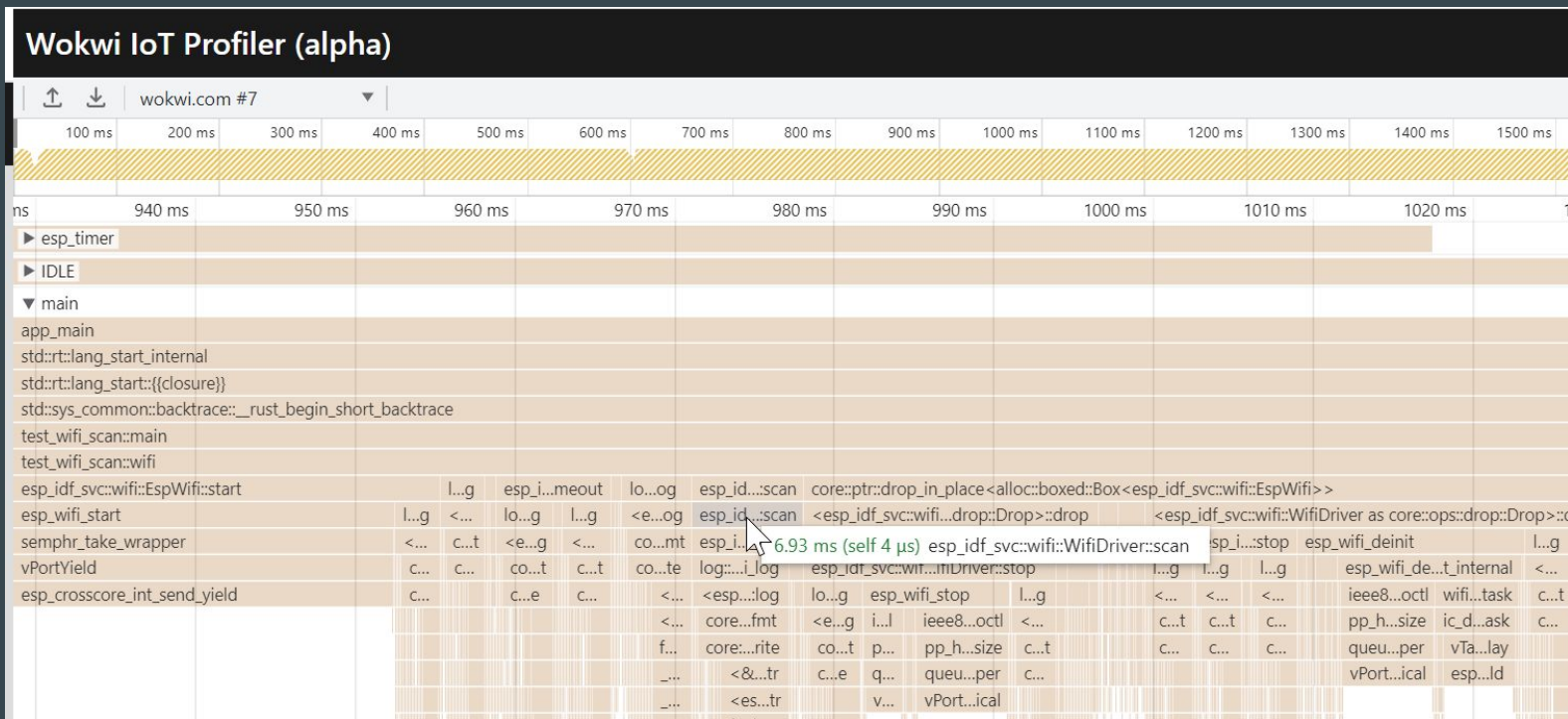
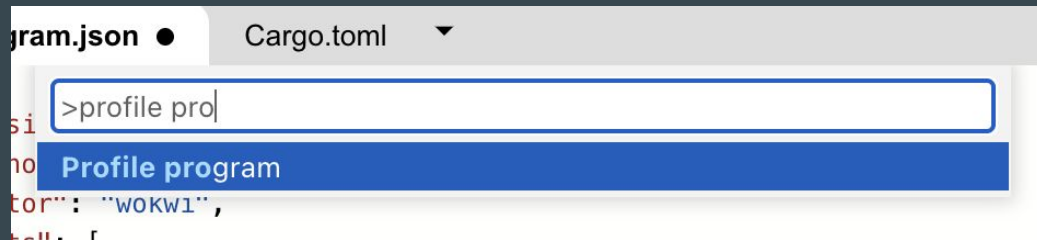


The diagram illustrates the hardware connection between an ESP32 microcontroller and an e-Paper display module. The ESP32 is a small black board with a blue header. The e-Paper display is a larger blue board with a black screen. Colored lines represent the connections between the two boards.

**WOKWI**

# Wokwi Profiler

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



# Wokwi CI

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

# 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

# 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



# OSes - C/CPP, 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



# Other languages and frameworks in context of ESP32

New and noteworthy:

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

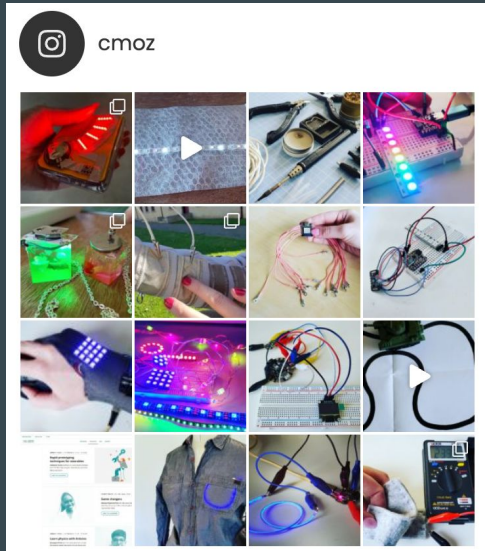
VM based:

- [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

# Wearables

The Ultimate Guide to Informed Wearable Technology

- book: <https://packt.link/01VBv>



# Developing IoT Projects with ESP32

<https://blog.espressif.com/book-review-developing-iot-projects-with-esp32-2nd-edition-facdef7545bb>

# Developing IoT Projects with ESP32

Discover the IoT development ecosystem with ESP32 to create production-grade smart devices

**Second Edition**



**<packt>**

**Vedat Ozan Oner**

# MicroPython Projects

<https://www.packtpub.com/product/micropython-projects/9781789958034>

# MicroPython Projects

A do-it-yourself guide for embedded developers to build a range of applications using Python



Jacob Beningo

**Packt**

[www.packt.com](http://www.packt.com)

# ESP32-C3 Wireless Adventures a Comprehensive Guide to IoT

<https://espressif.github.io/esp32-c3-book-en/>

## ESP32-C3

Wireless Adventure:  
A Comprehensive Guide to IoT

RISC-V Wi-Fi Bluetooth  
ESP-IDF ESP RainMaker



# More books about ESP32

<https://www.espressif.com/en/ecosystem/community-engagement/books>



ESPRESSIF

Hardware

SDKs

Cloud

Solutions

Support

**Ecosystem**

Company

Contact



## Partnership and Resource

AWS Technology Partner

Third-Party Platforms

Third-Party SDKs



## Developer Zone

Espressif DevCon

Tech Blogs

ESP32 Forum



## Community

Courses

Rust

**Books**

Videos

Projects

# 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

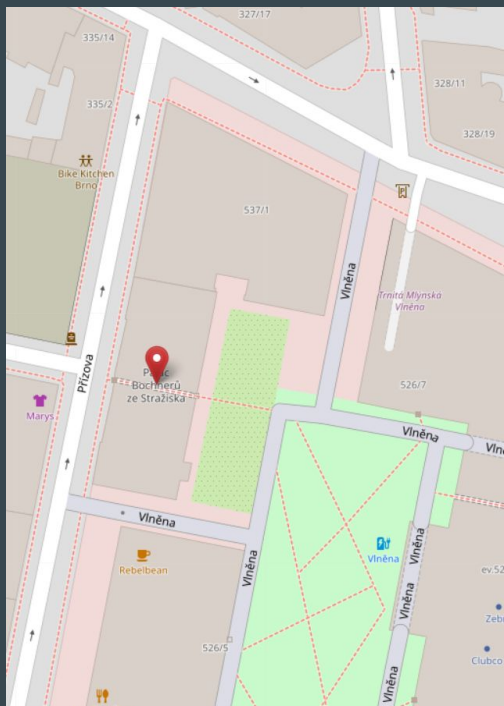
<https://youtu.be/OPPPdgoDBQs>





# Where you can find us?

Prízova 3, 602 00 Brno



<https://makerfaire.cz/praha/>

Výstaviště Designéři Na Výstavišti

# Maker Faire® Prague

May 11 – 12, 2024  
10:00 – 17:00

Výstaviště Praha 📍

A stylized illustration of a red and blue electronic device, possibly a microcontroller board, with various components like a lightbulb, gears, and a satellite dish. The illustration is set against a light blue grid background.



# Embedded World 2025

Meet us in Nuremberg, Germany - 11th - 13th March 2025



**embeddedworld**

Exhibition&Conference

# Espressif Developer Conference 2023 - recording




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

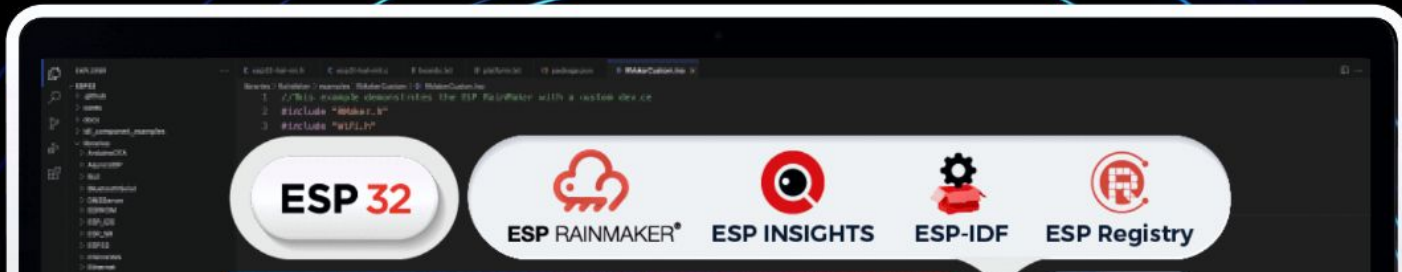
<https://youtu.be/mR3gUNXMEsM?si=7Rkq7wbCICvnM3pw>

<https://devcon.espressif.com/> - Become a speaker

Sep 3-5, 13:00-18:00 CEST

# Espressif DevCon24

 Add to Calendar >



# Visit us in Brno

Espressif Systems (Czech) s.r.o.

Přízova 3, 602 00 Brno

Czechia, Europe

