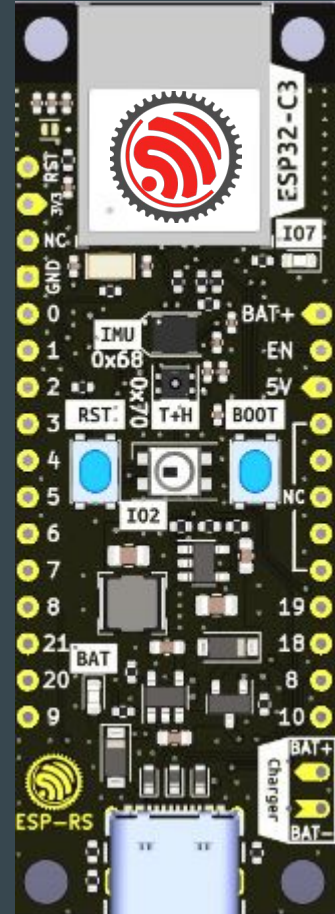


ESP32 Ecosystem



2024-10-24
FEKT VUT - Brno

Juraj Michálek & Team - Espressif Systems



Espressif team

Juraj Michálek - ESP32 Ecosystem Introduction & Matter

Sudeep Mohanty - ESP32-C6 Low Power cores

Peter Dragúň - IDF Monitor Tips & Tricks

Jan Beran - [ESP32-C6 Workshop](#)



Bonus: Stay tuned, some cool demos at the end with many pixels ;-)

Who are you?

Favorite programming language?

Favorite IDE / Editor?

Operating system?

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 - dual-core

ESP32-S - Xtensa arch

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

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



Espressif in Brno

Vlněna Office Park

Espressif Systems (Czech) s.r.o.

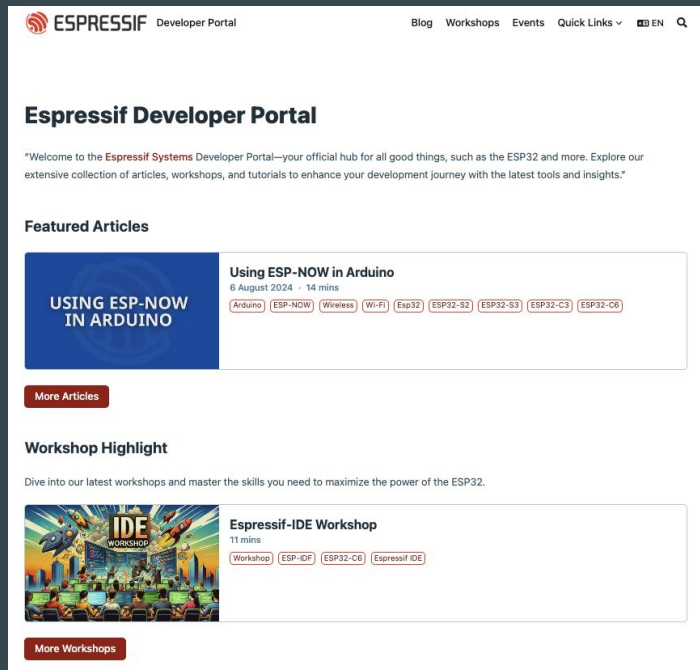
Přízova 3, 602 00 Brno

Czechia, Europe



Developer Portal

developer.espressif.com




The screenshot shows the Espressif Developer Portal homepage. At the top, there is a navigation bar with the Espressif logo, the text "Espressif Developer Portal", and links for "Blog", "Workshops", "Events", "Quick Links", "EN", and a search icon. Below the navigation bar, the main heading is "Espressif Developer Portal". A welcome message follows: "Welcome to the Espressif Systems Developer Portal—your official hub for all good things, such as the ESP32 and more. Explore our extensive collection of articles, workshops, and tutorials to enhance your development journey with the latest tools and insights." The "Featured Articles" section highlights an article titled "Using ESP-NOW in Arduino" with a blue background image. The article details include the date "6 August 2024" and duration "14 mins", along with tags for "Arduino", "ESP-NOW", "Wireless", "Wi-Fi", "Esp32", "ESP32-S2", "ESP32-S3", "ESP32-C3", and "ESP32-C6". A "More Articles" button is located below the article. The "Workshop Highlight" section features a colorful illustration of a workshop. The highlighted workshop is titled "Espressif-IDE Workshop" with a duration of "11 mins" and tags for "Workshop", "ESP-IDF", "ESP32-C6", and "Espressif IDE". A "More Workshops" button is positioned below the workshop highlight.

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

ESP Product Selector + Product Comparison

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



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:



Product Brief
Docs & Certs
DevKits

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 ...

Product Selector | **Product Comparison**

[Product Portfolio](#) [Sales Questions](#) [Technical Inquiries](#)

[Hide Same](#) | [Show Same](#)

	ESP32-S3	ESP32-P4NRW16
Series	ESP32-S3	ESP32-P4
CPU	Xtensa® dual-core 32-bit LX7	32-bit RISC-V single-core processor
Freq. (MHz)	240	400
Package (mm)	QFN56 (7*7)	QFN10*10
Dimensions (mm)	7*7	10*10
Temp. (°C)	-40 °C ~ 105 °C	-40 °C ~ 85 °C
Status	Mass Production	Sample
ECO	standard version	
Support IDF	v0.x	

OSes and integration



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

no_std a.k.a. bare metal with Rust - <https://github.com/esp-rs/esp-hal> (minimalistic)



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

- <https://developer.espressif.com/tags/zephyr/>
- <https://wokwi.com/projects/325751453458104914>



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

- <https://developer.espressif.com/tags/nuttx/>

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

Languages and frameworks

Compiled:

- [Embedded Swift](#)
- [Ada/Spark](#) - from AdaCore
- [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 - <https://developer.espressif.com/tags/vscode/>
- Espressif IDE - <https://developer.espressif.com/tags/espressif-ide/>

Supported by JetBrains:

- CLion - <https://www.jetbrains.com/help/clion/esp-idf.html>

Supported by SysProgs

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

Supported by TARA Systems

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



SYSPROGS

 **Embedded
Wizard**

GUI Solutions by TARA Systems

wokwi.com/esp32

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

EDC24: Flash Less, Do More: The Magic of Virtual Hardware

EDC22: Your browser is ESP32 - Wokwi

Rust project examples

+ NEW PROJECT

```
println!("Blinks an LED");
println!("");
// This assumes that a LED is connected to GPIO5
// Depending on your target, you may be using something else
// If your board doesn't have a LED, it's better to add an approx.
// target to add an approx.
println!("");
use esp::thread;
use esp::time::Duration;

use embedded_hal::digital::blocking::OutputPin;
use esp::hal::gpio::pin_mode::PinMode;
```

esp32-blink.rs

```
println!("Blinks an LED");
println!("");
// This assumes that a LED is connected to GPIO5
// Depending on your target, you may be using something else
// If your board doesn't have a LED, it's better to add an approx.
// target to add an approx.
println!("");
use esp::thread;
use esp::time::Duration;

use embedded_hal::digital::blocking::OutputPin;
use esp::hal::gpio::pin_mode::PinMode;
```

esp32s2-blink.rs

```
println!("Blinks an LED");
println!("");
// This assumes that a LED is connected to GPIO5
// Depending on your target, you may be using something else
// If your board doesn't have a LED, it's better to add an approx.
// target to add an approx.
println!("");
use esp::thread;
use esp::time::Duration;

use embedded_hal::digital::blocking::OutputPin;
use esp::hal::gpio::pin_mode::PinMode;
```

esp32c3-blink.rs

```
println!("");
println!("");
use esp32::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;

#![no_std]
#![no_main]
use embedded_hal::digital::blocking::OutputPin;
use esp::hal::gpio::pin_mode::PinMode;
```

esp32-nostd-blink.rs

```
println!("");
println!("");
use esp32s2::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;

#![no_std]
#![no_main]
use embedded_hal::digital::blocking::OutputPin;
use esp::hal::gpio::pin_mode::PinMode;
```

esp32s2-nostd-blink.rs

```
println!("");
println!("");
use esp32c3::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;

#![no_std]
#![no_main]
use embedded_hal::digital::blocking::OutputPin;
use esp::hal::gpio::pin_mode::PinMode;
```

esp32c3-nostd-blink.rs

```
use esp::thread;
use esp::time::Duration;
use esp::pin_mode::PinMode;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

esp-clock (WiFi)

```
println!("");
println!("");
use esp32::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

Crispy Click

```
println!("");
println!("");
use esp32c3::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

ESP32C3 + ILI9341 Display

```
println!("");
println!("");
use esp32s3::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

ESP32S3 + ILI9341 Display

```
println!("");
println!("");
use esp32::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

ESP32 + ILI9341 Display

```
println!("");
println!("");
use esp32::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

ESP32 + 8x8 LED Dot Matrix

```
println!("");
println!("");
use esp32s2::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

ESP32S2 + Keypad

```
println!("");
println!("");
use esp32::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

Joystick Etch-a-Sketch

```
println!("");
println!("");
use esp32c3::hal::gpio::Pin;
use esp::peripherals::periph::rmt::Rmt;
use esp::time::Duration;
use esp::thread;
use esp::pin_mode::PinMode;
```

esp-gallery

Wokwi - VS Code Plugin

Add wokwi.toml and diagram.json
to your project

wokwi-cli init

CTRL+Shift+P - Wokwi: Start Simulator

Wokwi Simulator - m5stack-fire - Visual Studio Code

EXPLORER

- OPEN EDITORS
 - wokwi.toml
 - run-wokwi.sh
 - Wokwi Simulator
- M5STACK-FIRE
 - .cargo
 - src
 - target
 - Cargo.lock
 - Cargo.toml
 - diagram.json
 - run-wokwi.sh
 - rust-toolchain.toml
 - wokwi.toml
- OUTLINE
- TIMELINE
- PROJECT COMPONENTS

WOKWI Wokwi for VS Code

Licensed to: Juraj Michálek

00:17.099 43%

MPU6050 Accelerometer + Gyroscope

ACCELERATION

X: 0 g

Y: 0 g

Z: 1 g

ROTATION

X: 0°/sec

Y: 0°/sec

Z: 0°/sec

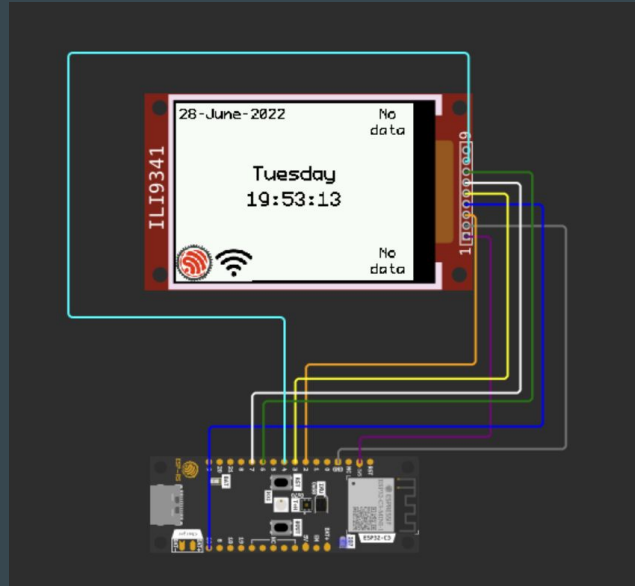
TEMPERATURE

24.0°C

ESP32

I119341

Development containers and Wokwi



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

The **ESP** Component Registry

Discover, download and publish components and examples for ESP-IDF



Browse components

ALL Board Support Package

Compatible with ESP-IDF: v5.0 v5.1 v5.2 v5.3

By target: ESP32 ESP32-C2 ESP32-C3 ESP32-C5 ESP32-C6 ESP32-C61 ESP32-H2 ESP32-P4 ESP32-S2 ESP32-S3

Featured

espressif/mdns

v1.4.0

uploaded 2 months ago

mDNS

lvgl/lvgl

v9.2.0

uploaded 1 month ago

LVGL - Light and Versatile Graphics Library

espressif/esp-modbus

v1.0.16

uploaded 1 month ago

ESP-MODBUS is the official Modbus library for Espressif SoCs.

joltwallet/littlefs

v1.14.8

uploaded 3 months ago

LittleFS is a small fail-safe filesystem for micro-controllers.

espressif/arduino-esp32

v3.0.7

uploaded 23 hours ago

Arduino core for ESP32, ESP32-S and ESP32-C series of SoCs

espressif/openai

v1.0.0

uploaded 5 months ago

OpenAI library compatible with ESP-IDF

wolfssl/wolfssl

v5.7.2

uploaded 3 months ago

wolfSSL Embedded SSL/TLS Library

slint/slint

v1.8.0

uploaded 1 month ago

Slint — declarative GUI toolkit

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

Example: [https://components.espressif.com/components/espressif/i2c bus/](https://components.espressif.com/components/espressif/i2c_bus/)

Example: ESP32 + SDL3 components

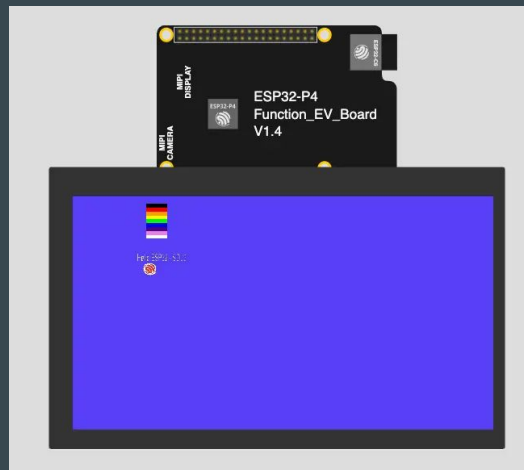
<https://github.com/georgik/esp32-sdl3-example>

[main/idf component.yml](#)

Wrapping components:

<https://github.com/georgik/esp-idf-component-SDL>

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



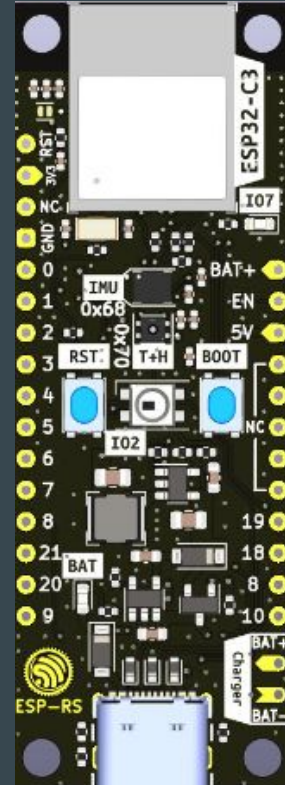
Designing Open Hardware - esp-rust-board

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>





GUI Solutions by TARA Systems

Embedded Wizard

Quick prototyping

Free for small, medium projects

Supports ESP32, WASM or Desktop



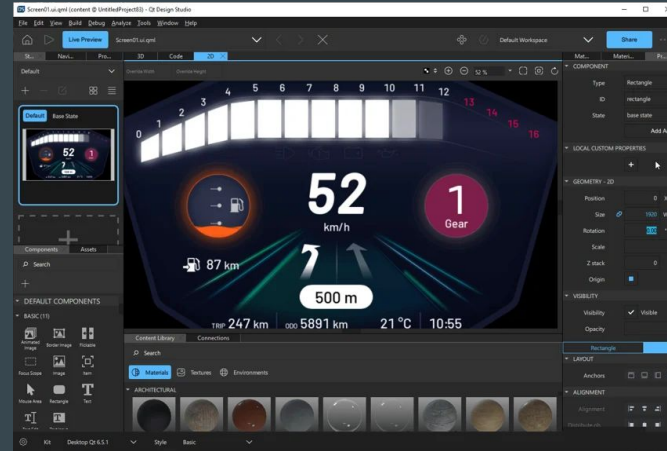


Qt for MCUs

Similar to Qt for desktop

Free for students

Supports ESP32-S3





Slint

Quick prototyping

Free and commercial version

Rust, C++



Rust Bare Metal - Embedded Graphics



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

Idea: sharing business logic in Rust between
multiple targets



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

Some examples of ESP32 based projects

CTAG-TBD

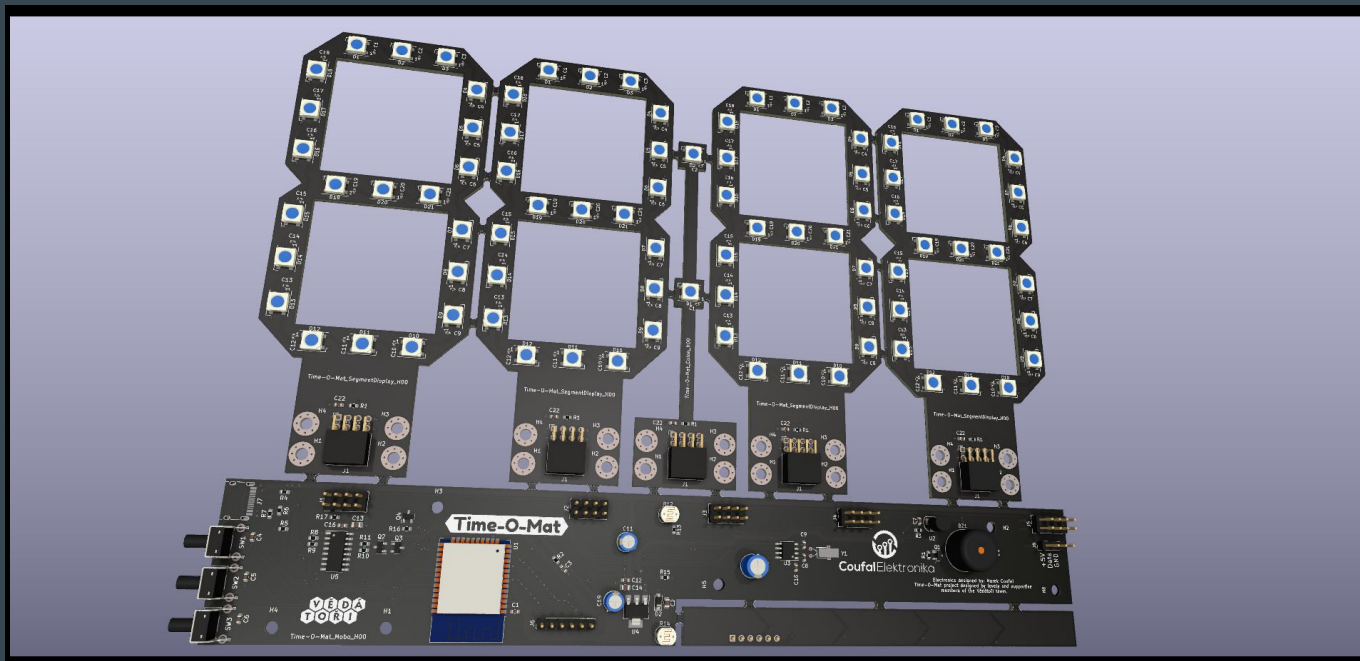
CTAG TBD >>to be determined<< an extendible
open source Eurorack sound module

<https://github.com/ctag-fh-kiel/ctag-tbd>



Time-O-Mat - built at summer camp

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



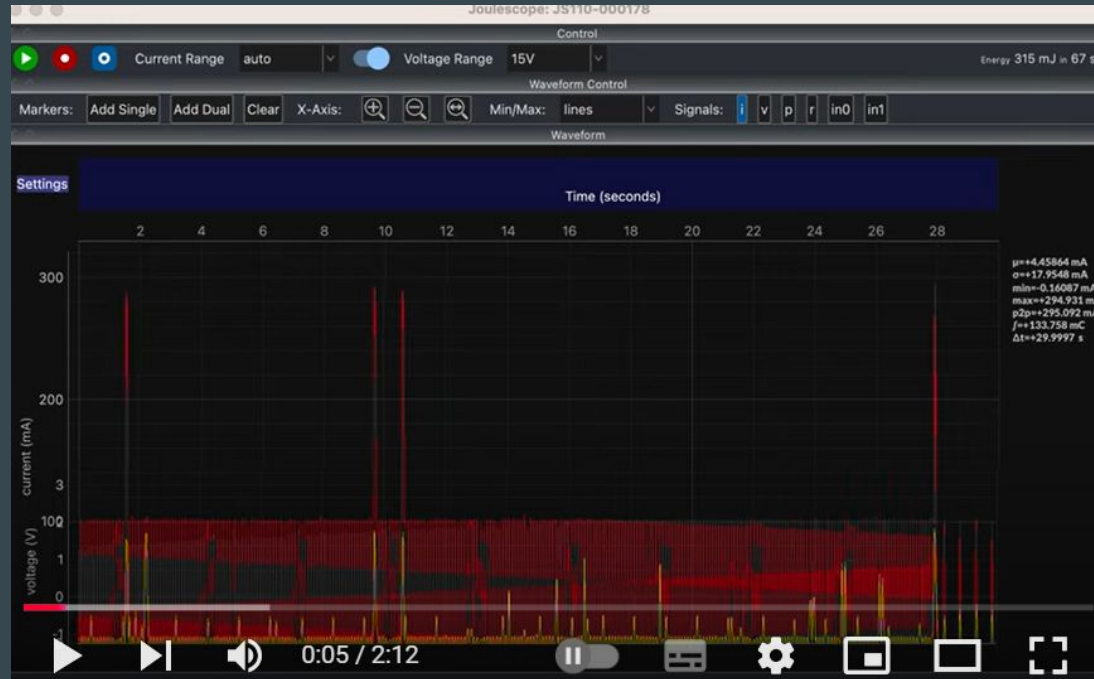
Grafana



<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 :)

ESP32-C6 TWT



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



Many Problems of Home Automation and IoT

Silos

Fragmentation

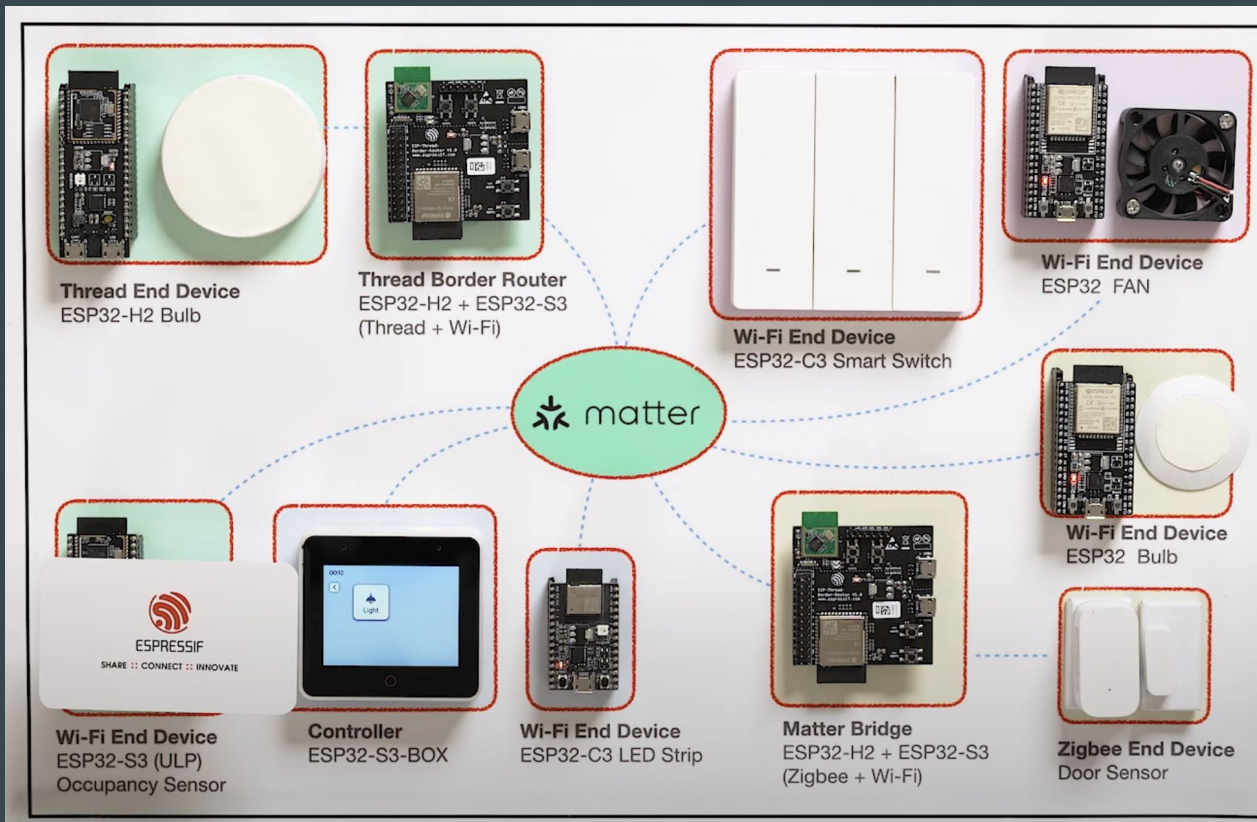
Security

UX

Development

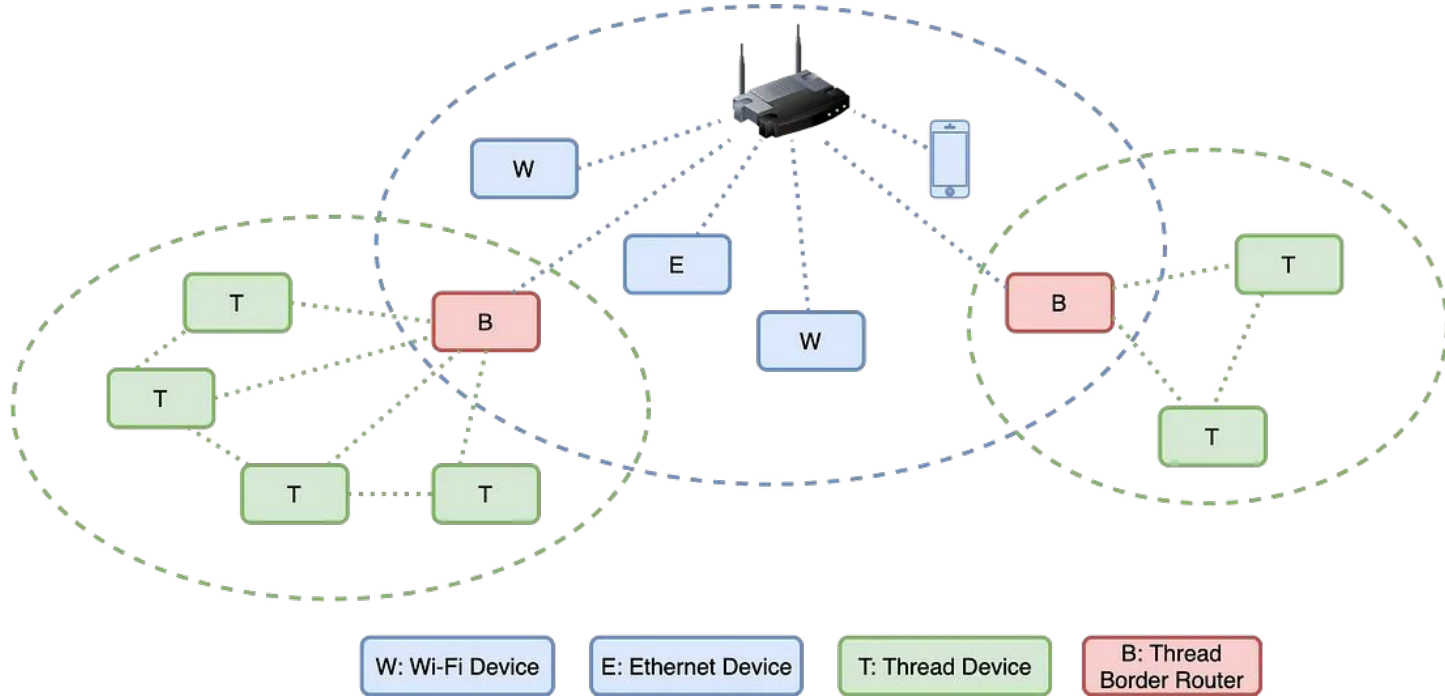
Certification

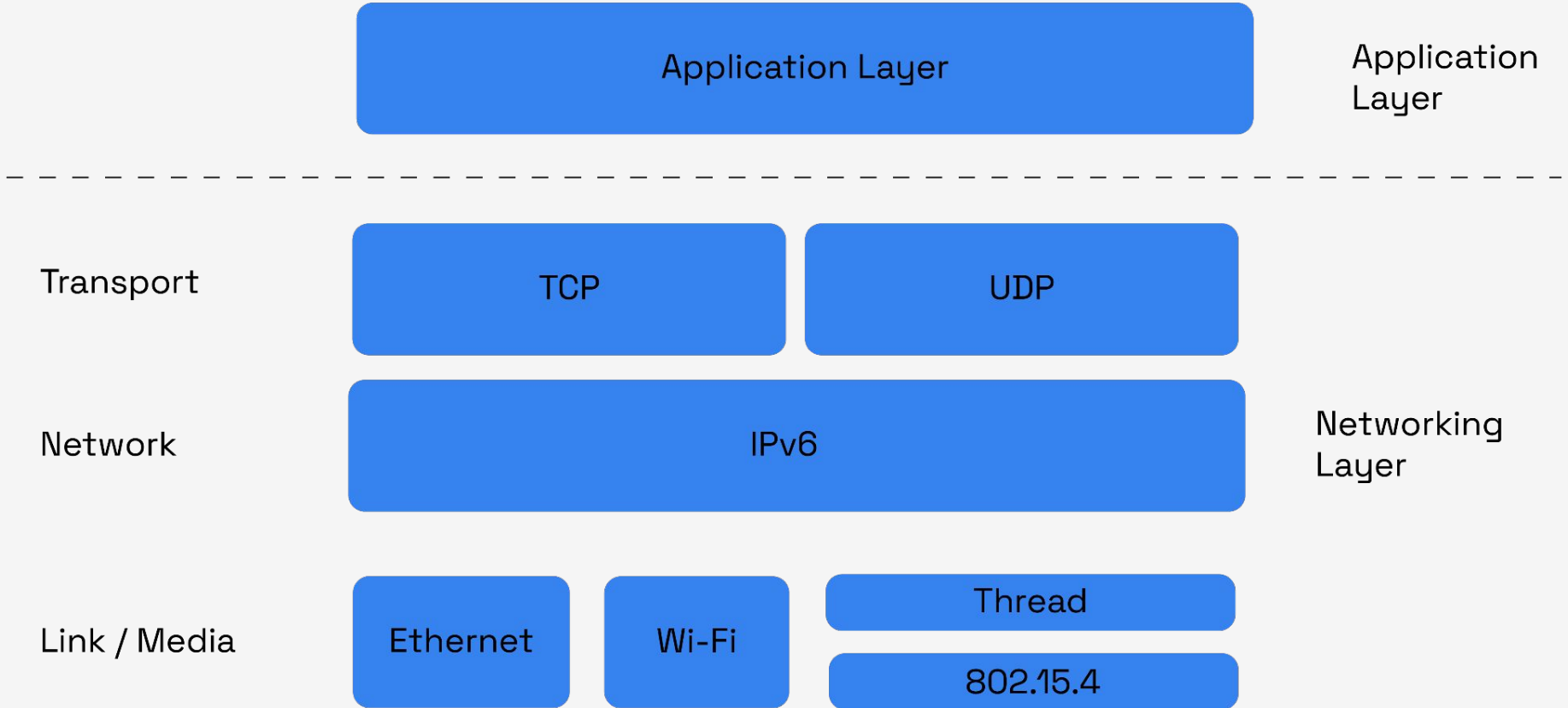
...



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

Matter topology





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>

matter-rs

Early development

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

ESP chips not supported yet

Join Rust ESP32 Community Meeting to see the progress:

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



Async with Embassy

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

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>

Espressif Developer Conference 2022-2024 - recording



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

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

Embedded World 2025

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

