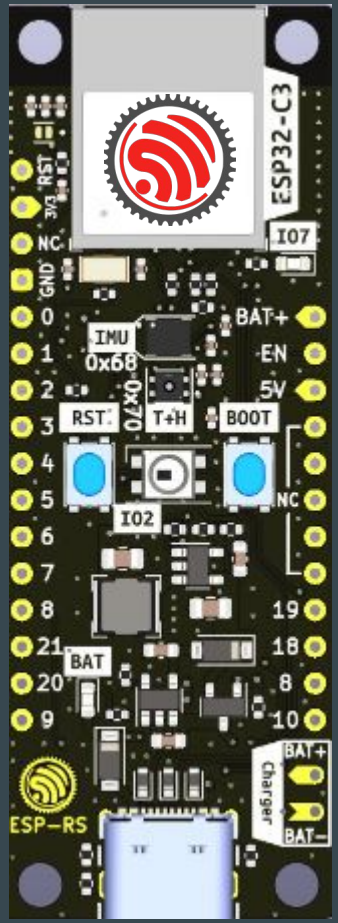
IoT Systems with ESP32



2024-03-15
FIT VUT - Brno

Juraj Michálek - Espressif Systems

What is your favorite?

Programming language

Operating system

Editor

IDE

Cloud provider

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/esp-rs/

- 500+ starts - esp-hal

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

- 12 300+ stars

Elektor Mag

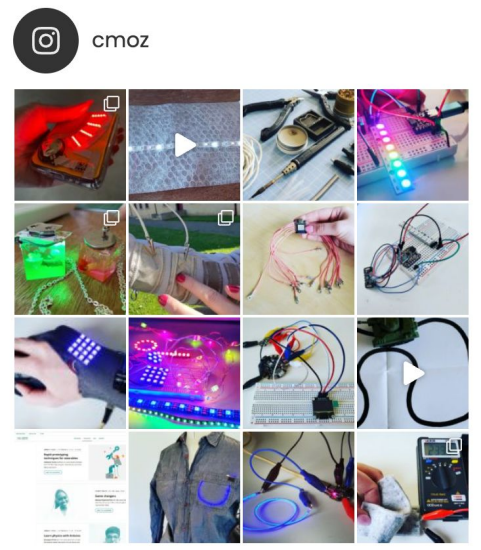
Special Guest Edition

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

Wearables

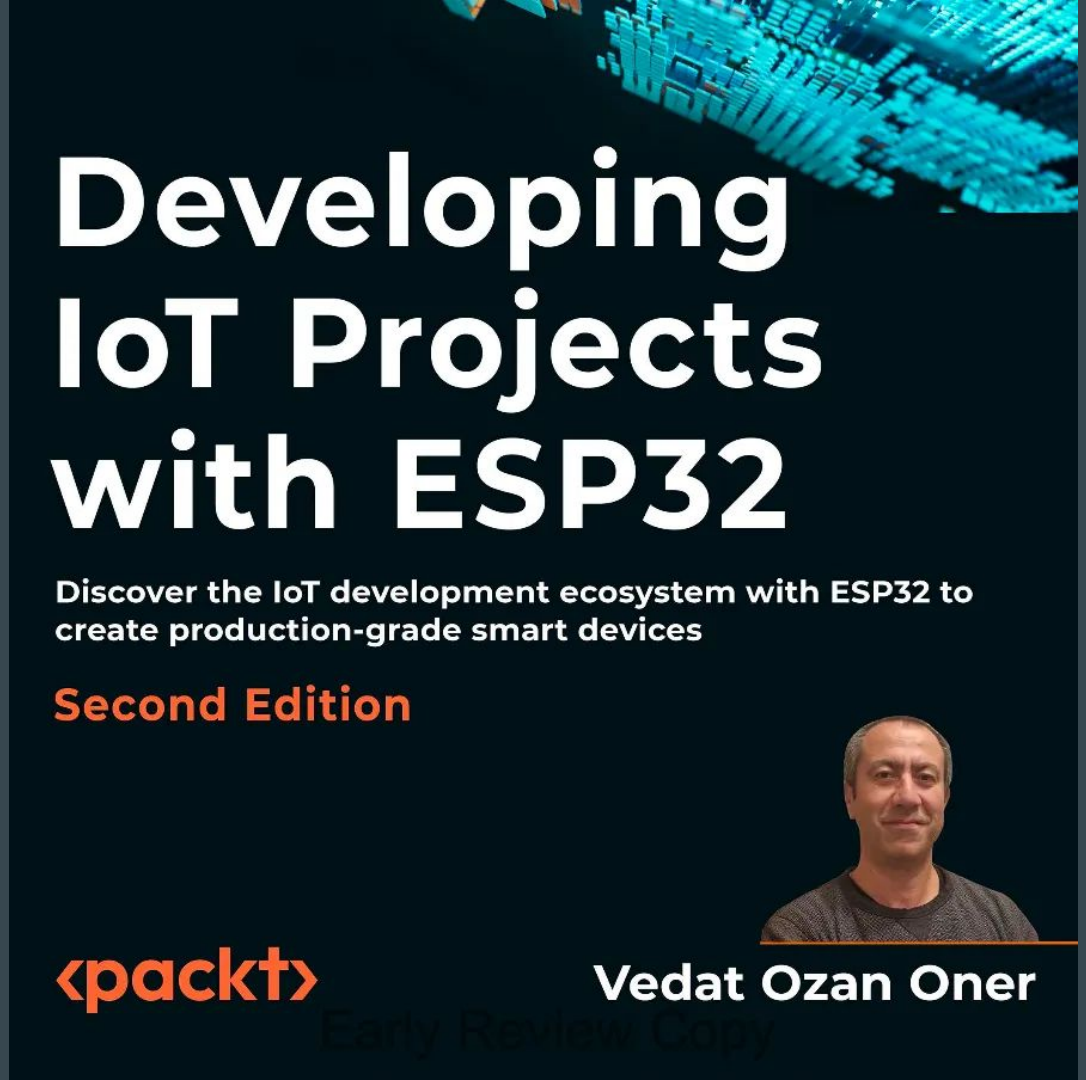
The Ultimate Guide to Informed Wearable Technology

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



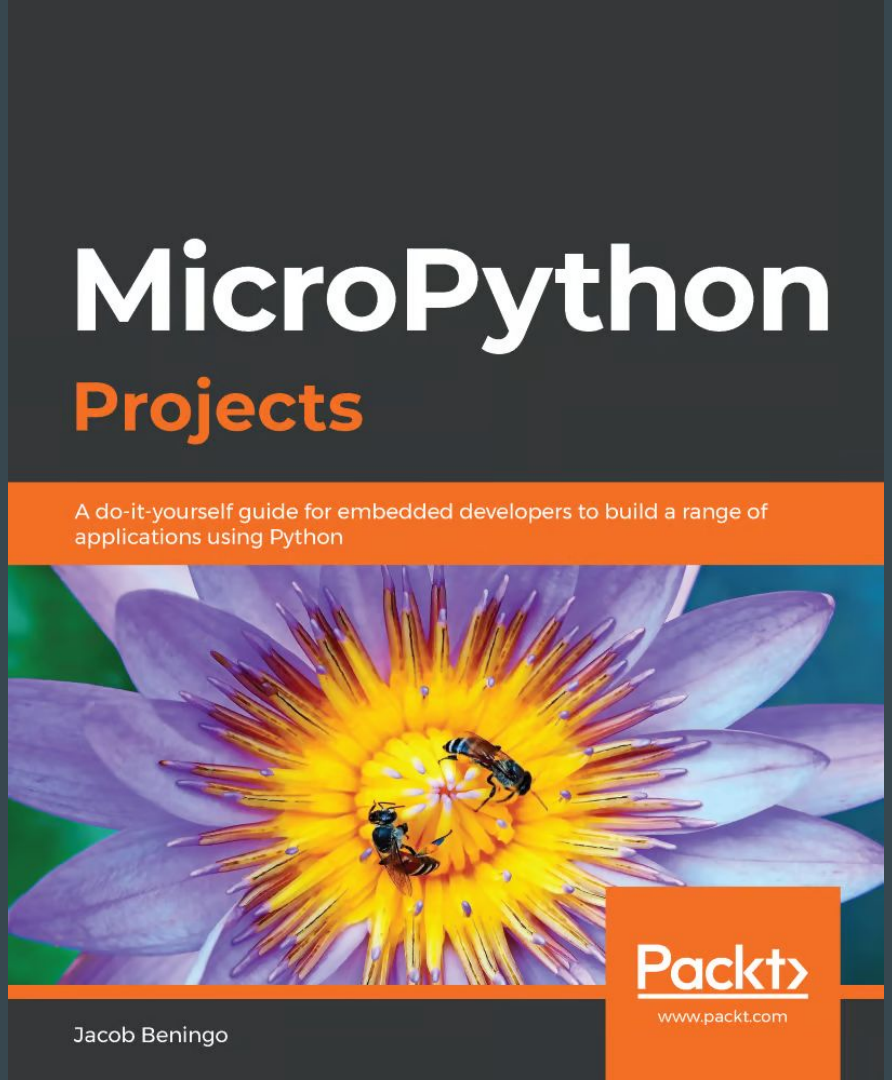
Developing IoT Projects with ESP32

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



MicroPython Projects

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

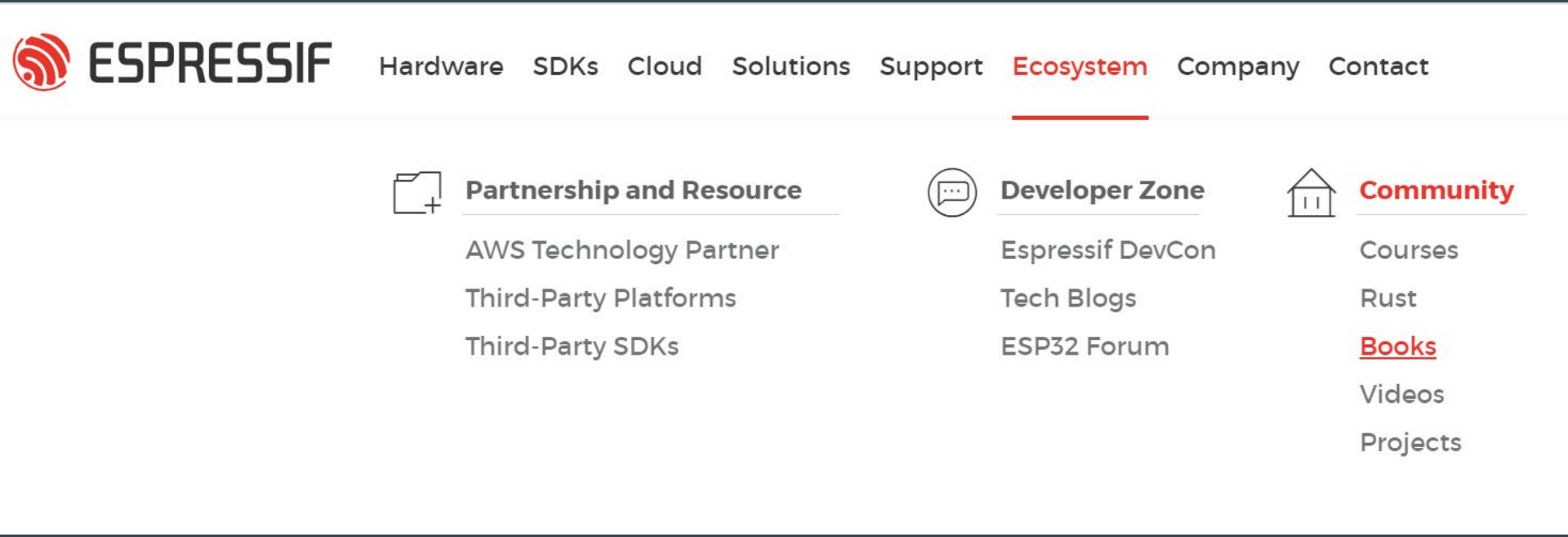


ESP32-C3 Wireless Adventures a Comprehensive Guide to IoT

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

More books about ESP32

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



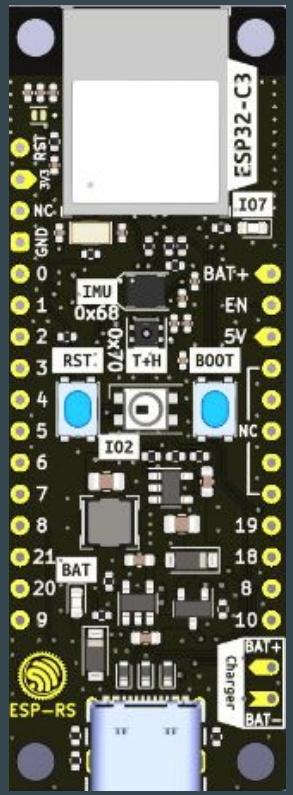
Designing Open Hardware - esp-rust-board

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

Desktop vs. Embedded development

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

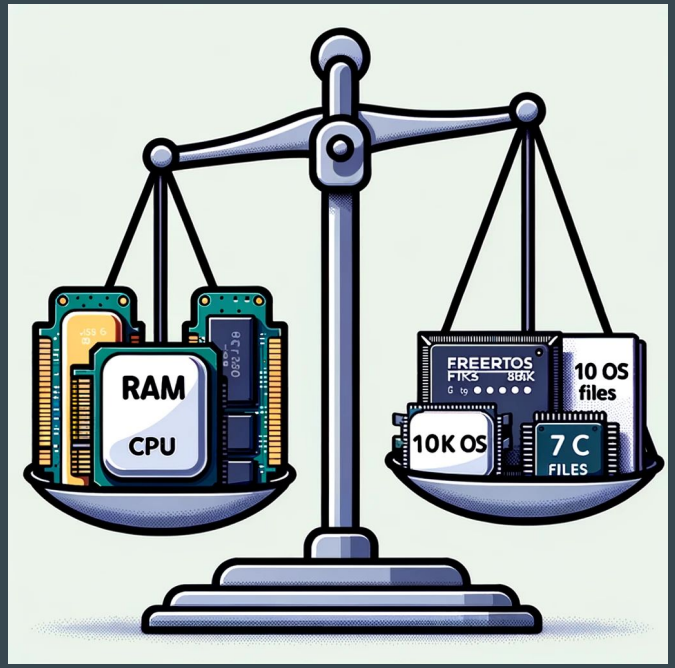
CPU Cores: 8 - 64 vs. 1 - 2

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 - 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

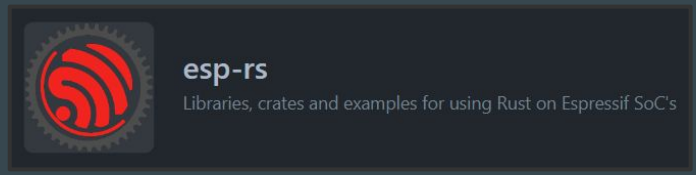
NuttX - https://nuttx.apache.org/ (as app, Linux-like OS)

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

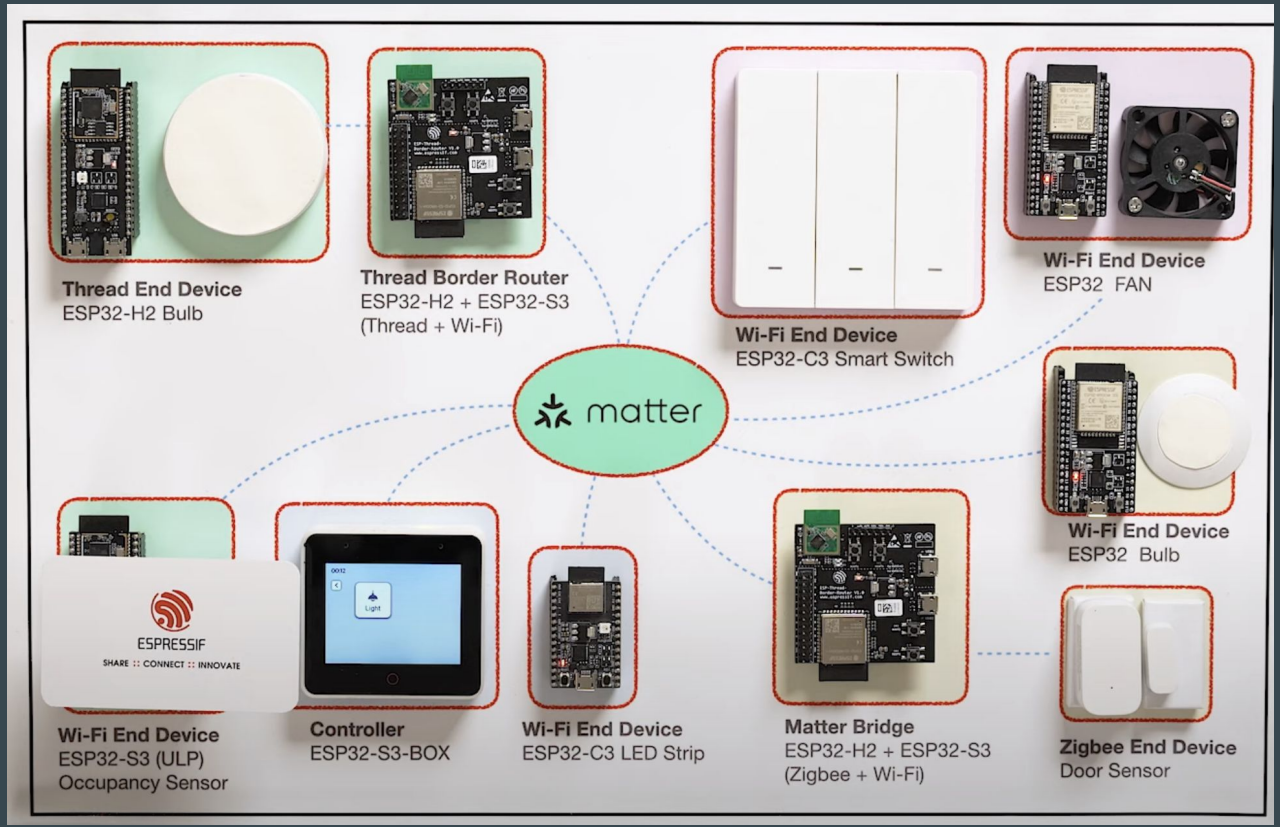


wokwi.com/rust

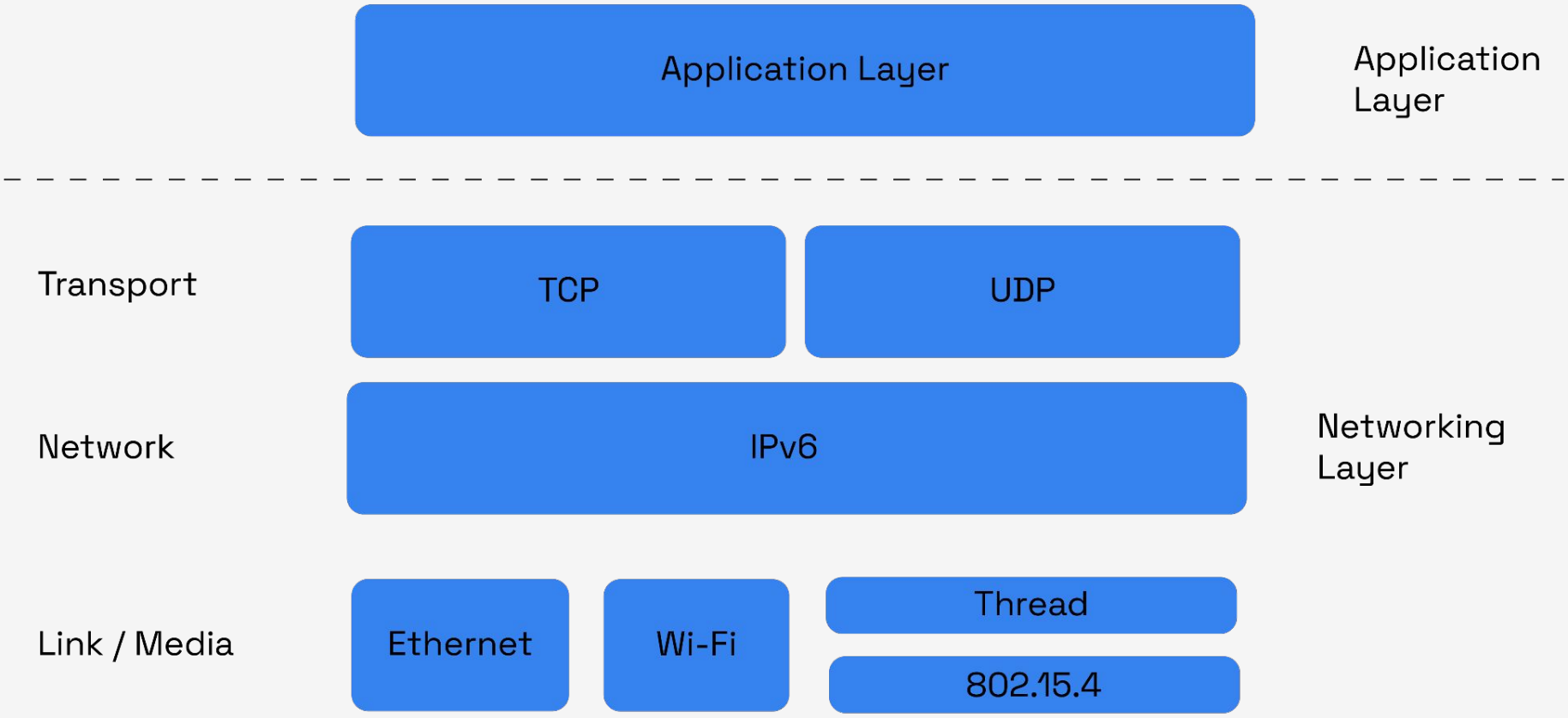
Contribute: https://github.com/wokwi

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


matter



Espressif's Matter Demo - https://youtu.be/Jr4Lut_NgqA


Application Layer
Application Layer
Transport
TCP
UDP
Network
IPv6
Networking Layer
Link / Media
Ethernet
Wi-Fi
Thread
802.15.4

Many Problems of Home Automation and IoT

Silos

Fragmentation

Security

UX

Development

Certification

...

Matter

Repo: https://github.com/project-chip/connectedhomeip

Specs: https://csa-iot.org/developer-resource/specifications-download-request/

Article: https://blog.espressif.com/what-does-matter-mean-to-you-fa3bb53a7793

Matter for end user

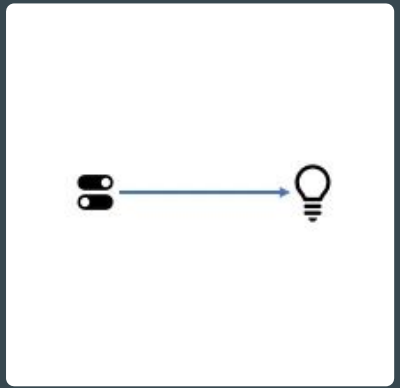
No more silos

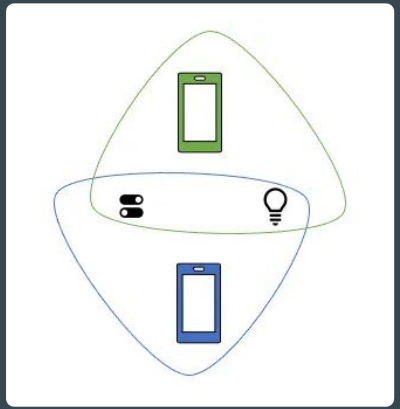
More automation - device-to-device communication

No more transport silos

More ecosystems

Better security





https://blog.espressif.com/what-does-matter-mean-to-you-fa3bb53a7793

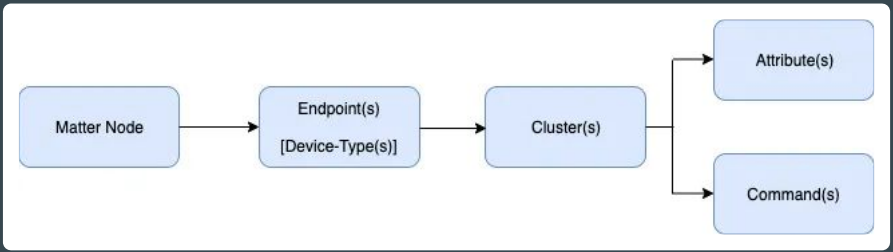
Espressif and Matter

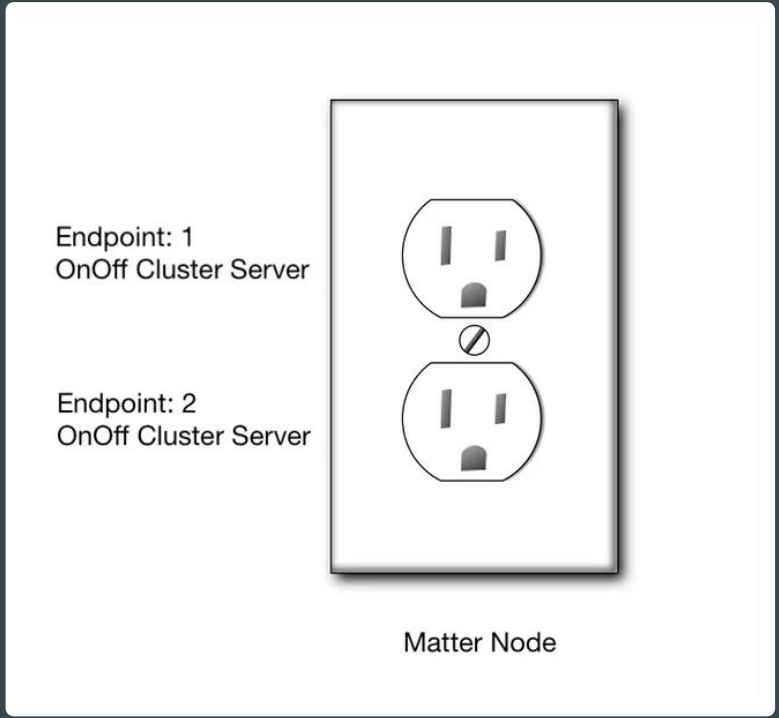
Support for ESP32 is upstream
https://github.com/project-chip/connectedhomeip/tree/master/examples/all-clusters-app/esp32

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

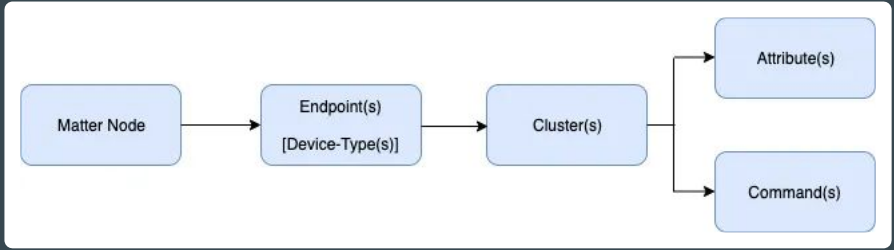
rs-matter - Rust - https://github.com/project-chip/rs-matter

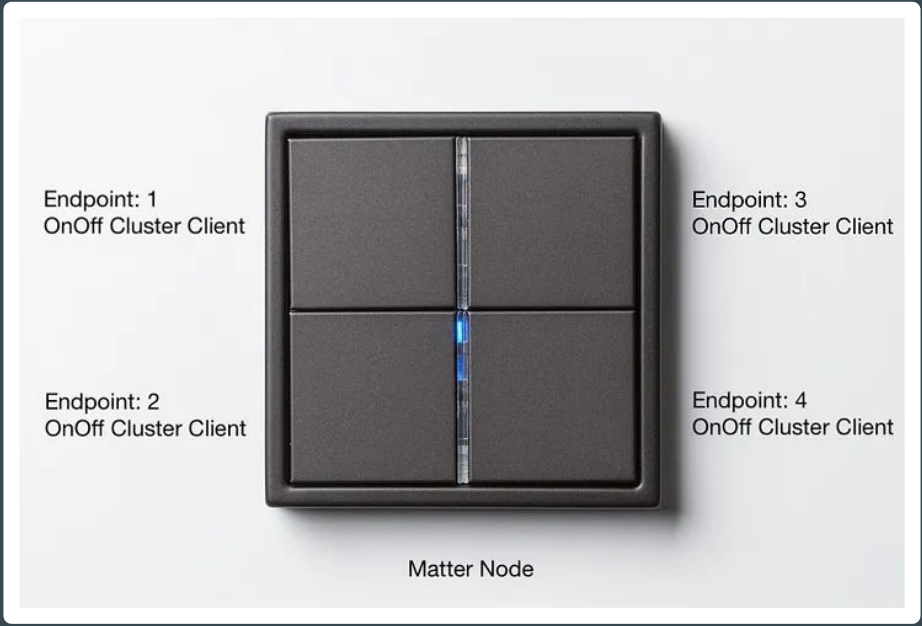
Matter node


Matter Node
Endpoint(s)
[Device-Type(s)]
Cluster(s)
Attribute(s)
Command(s)

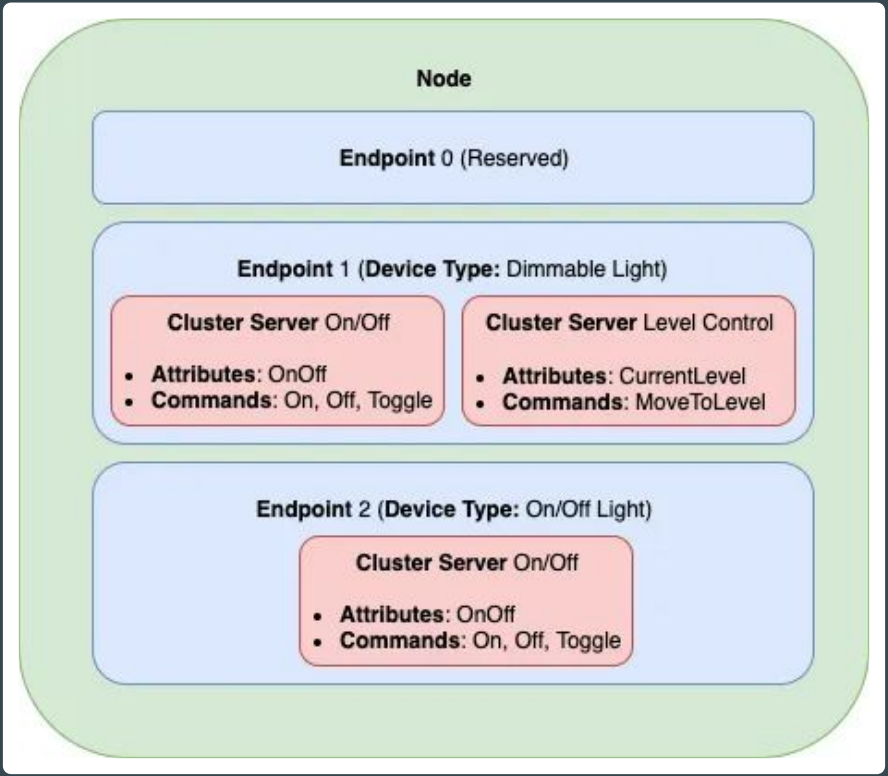

Endpoint: 1
OnOff Cluster Server
Endpoint: 2
OnOff Cluster Server
Matter Node

Matter Node - OnOff Cluster Client - 4 endpoints





Simplistic Representation of a Matter Data Model



https://blog.espressif.com/matter-clusters-attributes-commands-82b8ec1640a0

Cluster list

ID; Cluster Name

0x0003 - Identify

0x0004 - Groups

0x0005 - Scenes

0x0006 - On/Off

0x0008 - Level controlling

0x0045 - Boolean state

0x0050 - Mode Select

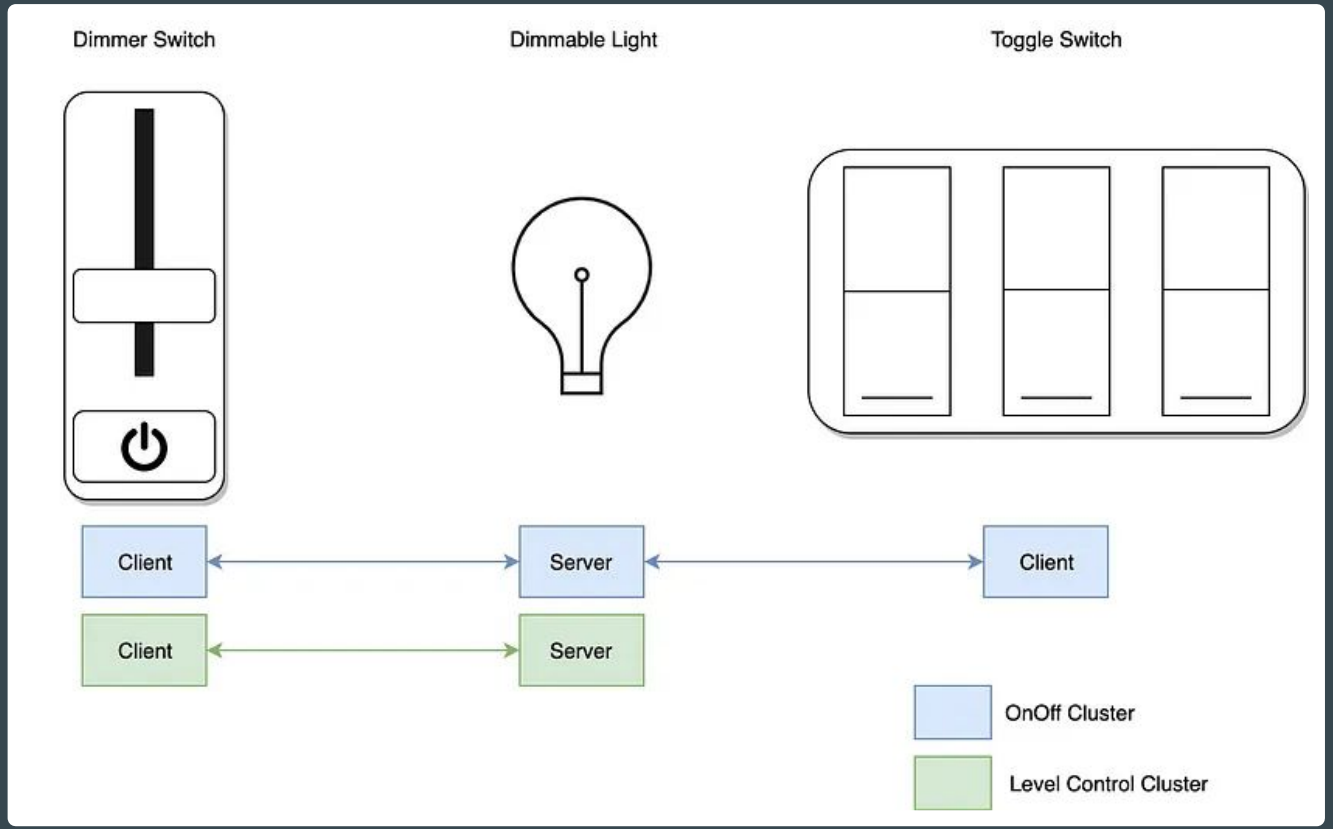
0x0508 - Low Power

0x0503 - Wake On LAN

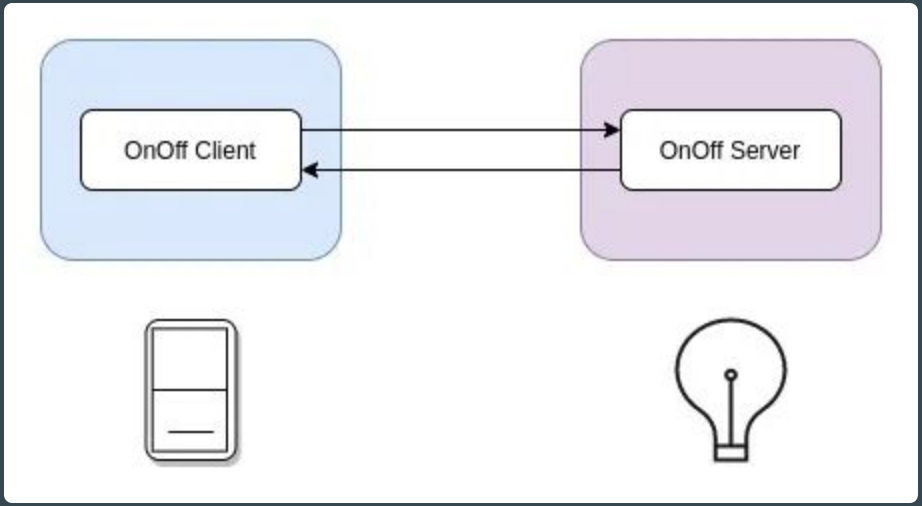
0x003b - Switch

Specification: Matter Application Clusters v1.0

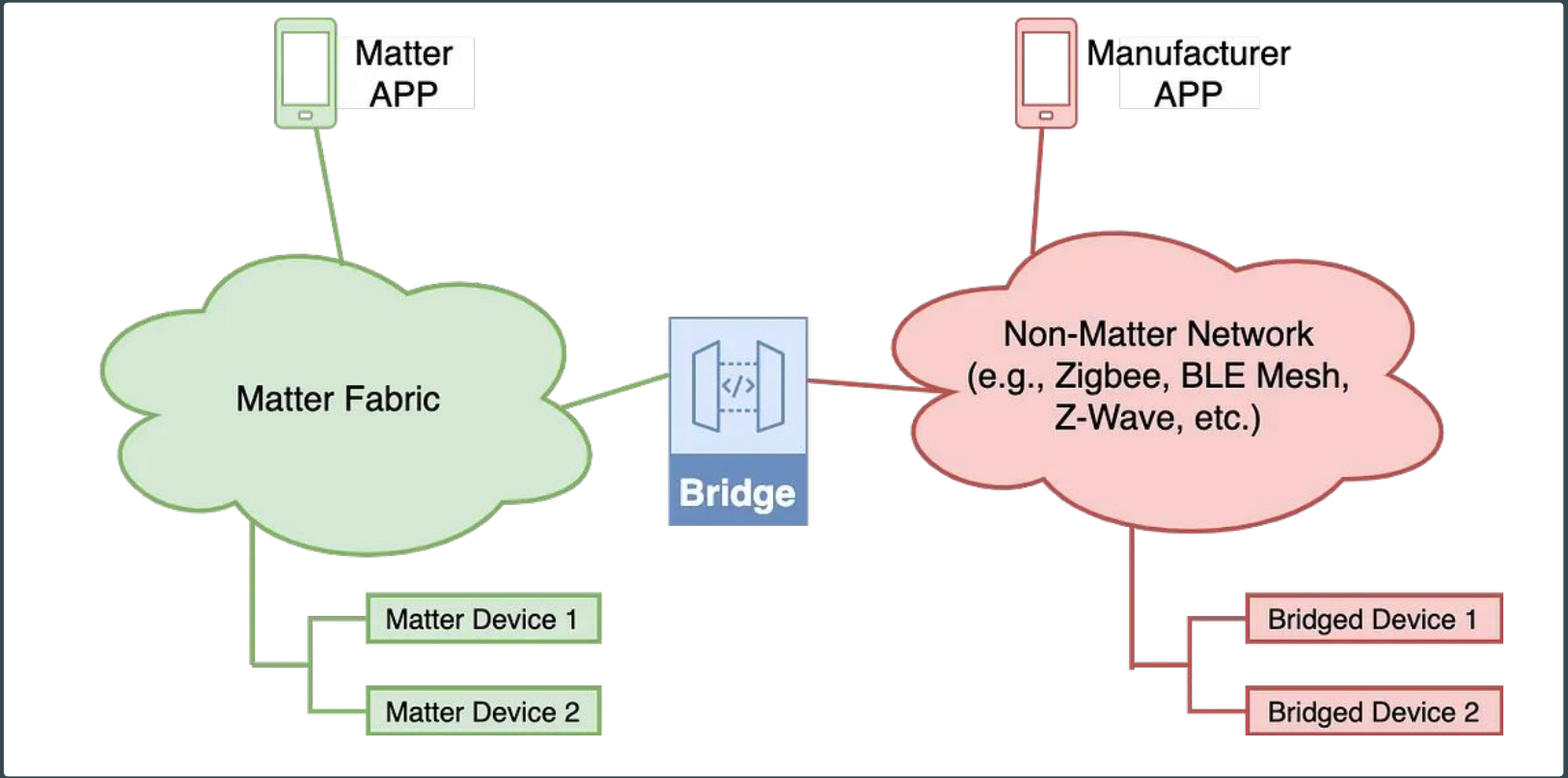
Matter - Client - Server

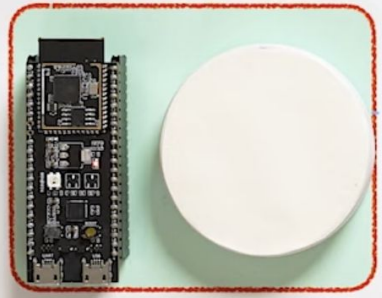


Device-to-device automation



Matter Bridge

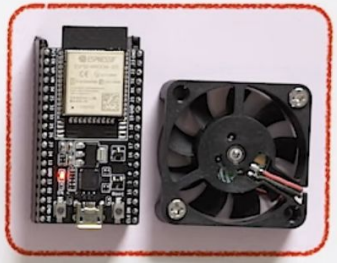




Thread End Device
ESP32-H2 Bulb

Thread Border Router
ESP32-H2 + ESP32-S3
(Thread + Wi-Fi)

Wi-Fi End Device
ESP32-C3 Smart Switch



Wi-Fi End Device
ESP32 FAN

matter

Wi-Fi End Device
ESP32 Bulb

Wi-Fi End Device
ESP32-S3 (ULP)
Occupancy Sensor

Controller
ESP32-S3-BOX

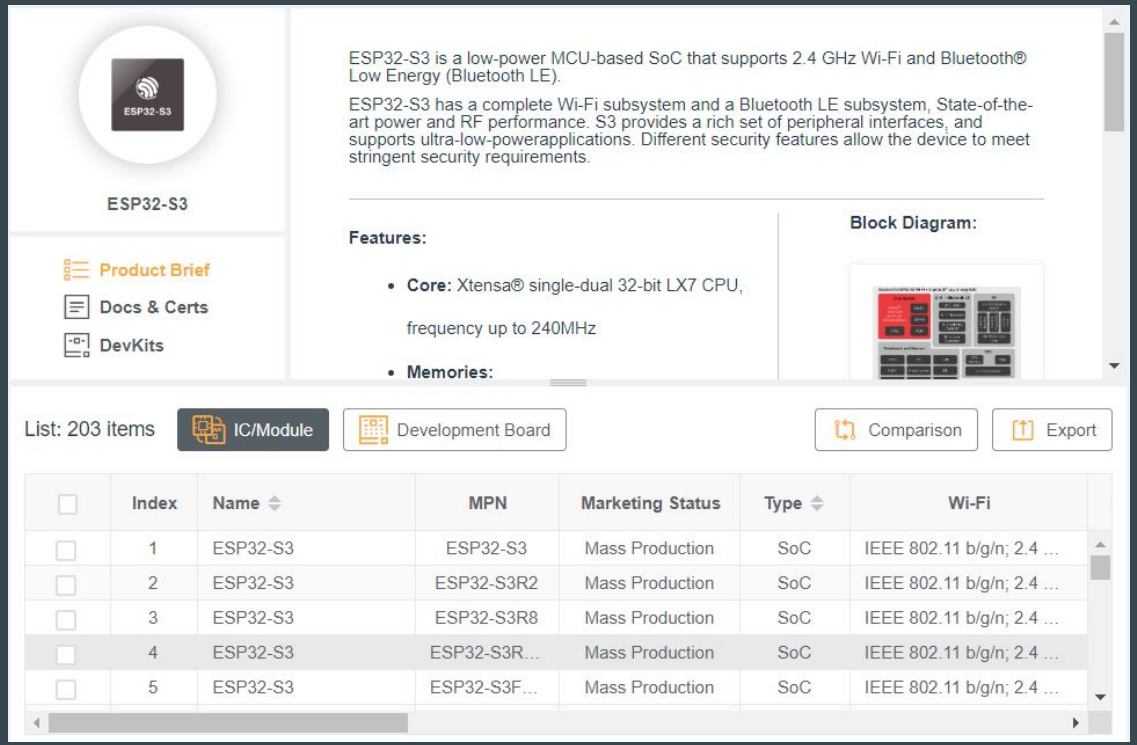
Wi-Fi End Device
ESP32-C3 LED Strip

Matter Bridge
ESP32-H2 + ESP32-S3
(Zigbee + Wi-Fi)

Zigbee End Device
Door Sensor

Many chips, many boards - quick help

https://products.espressif.com/



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/

SysProgs

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

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/i06djjOKvB8?si=mxXuuWPMpSs-JC2r

Rainmaker

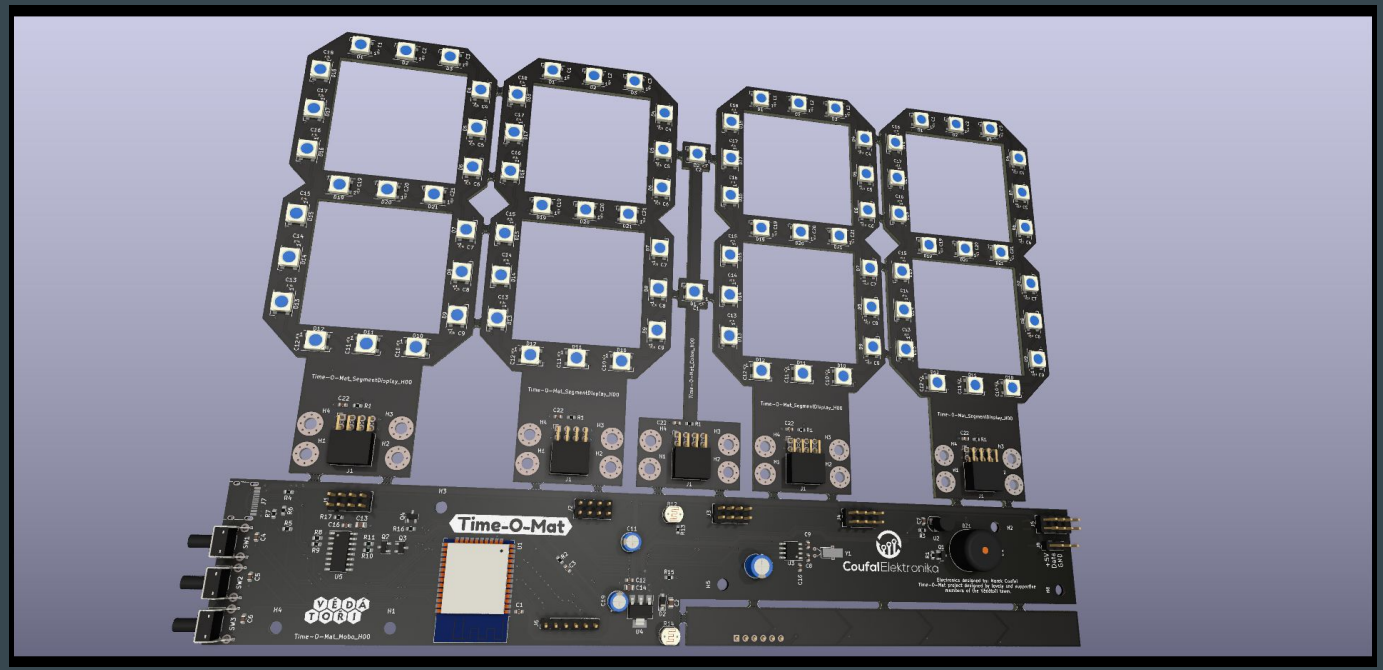
Rust layer - Linux & ESP

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

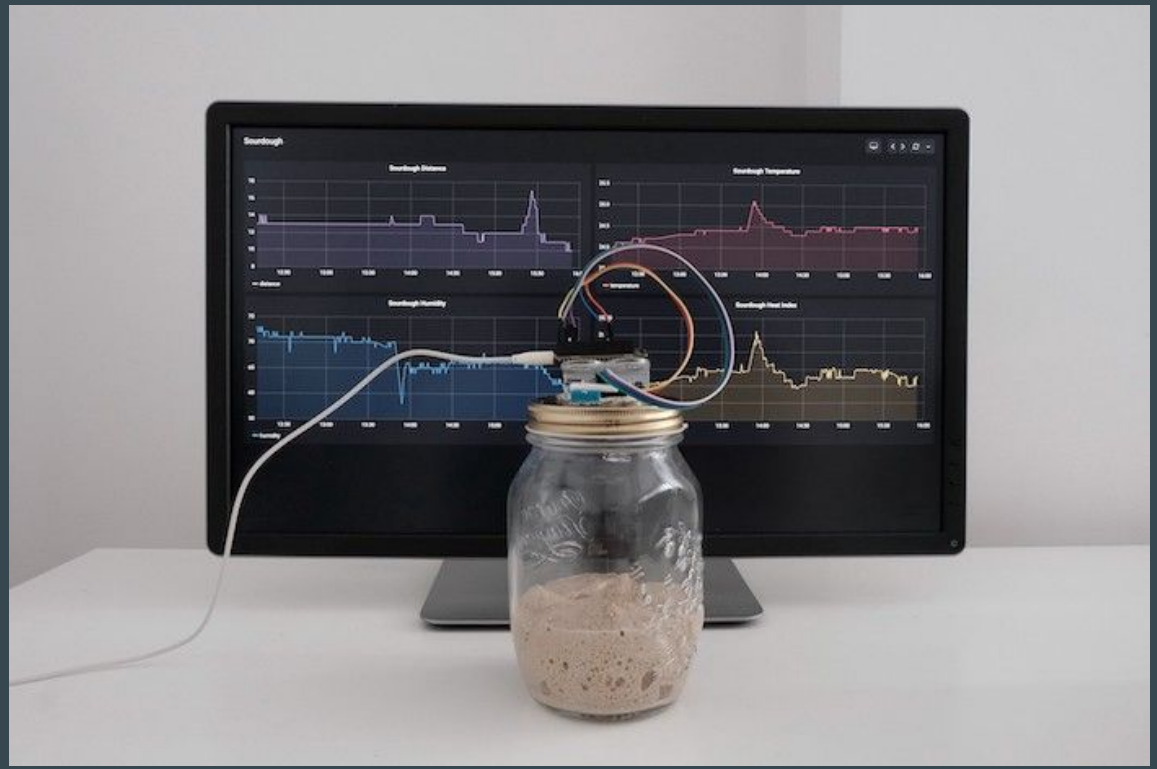
Cloud connectivity

Time-O-Mat - built at summer camp

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



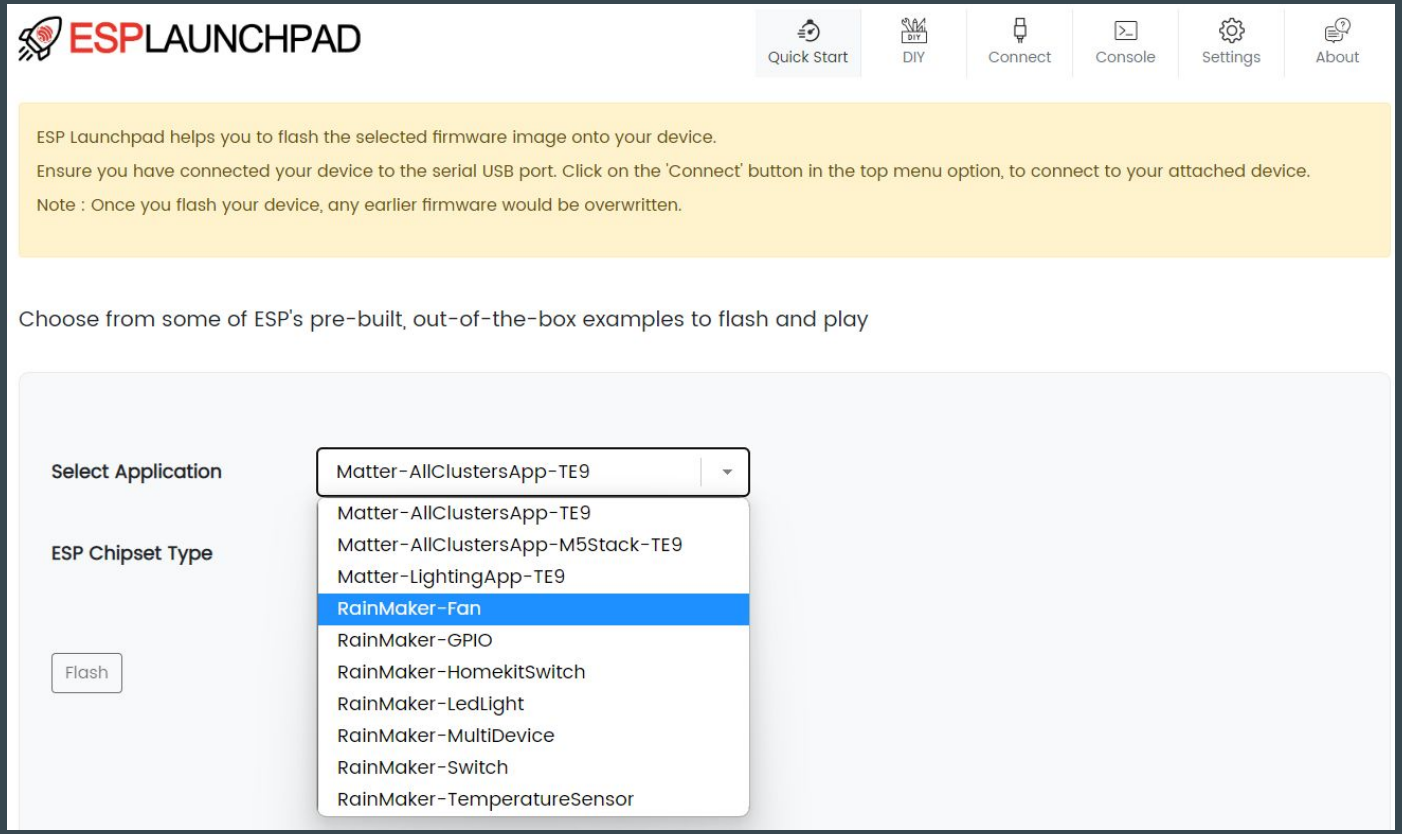
Grafana



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 :)

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



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_spi.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

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

Rust language support

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

https://youtu.be/OPPPdqoDBQs

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

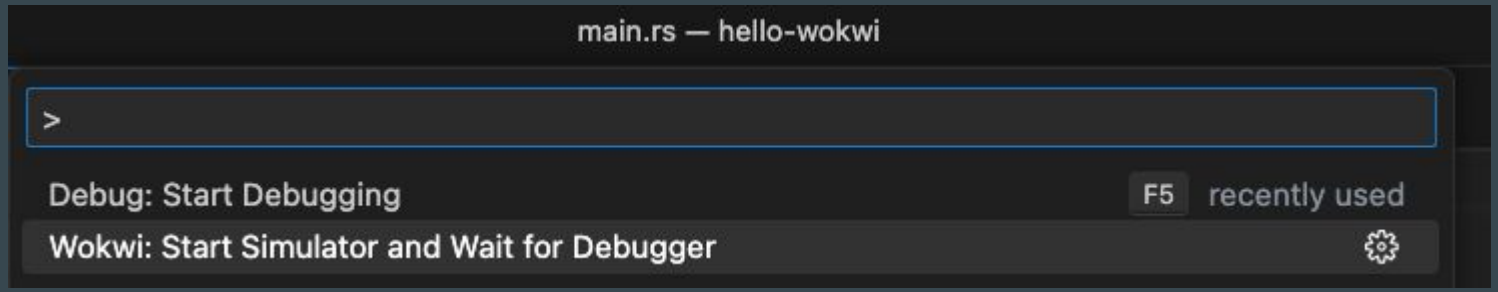
https://github.com/georgik/esp32-idf-nmea-example

idf.py set-target esp32-c3

idf.py -DCMAKE_BUILD_TYPE=Debug build

https://aerorust.org/

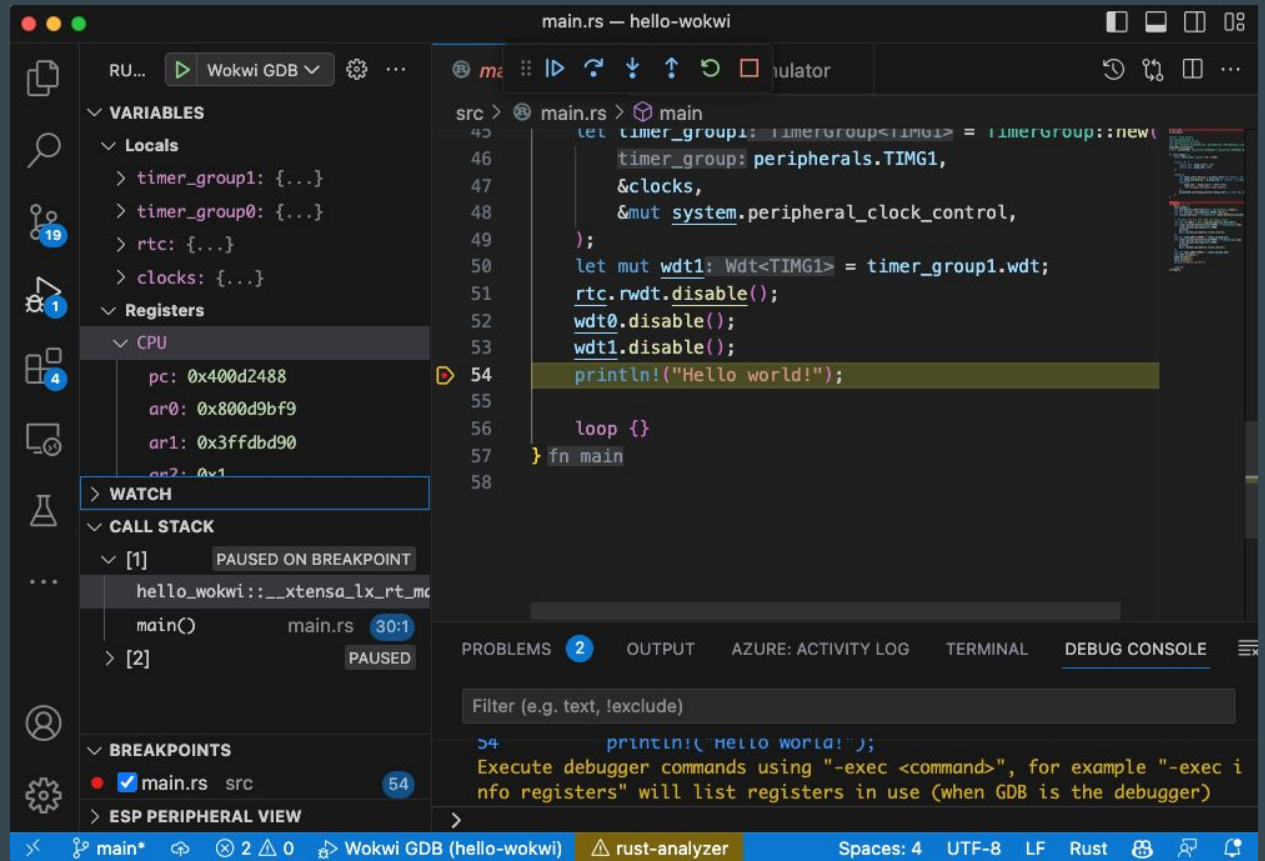
Debugging



2 steps:

- start simulator with debugger interface
- start debugging session

Debugging session with Wokwi



Templates for new projects

std

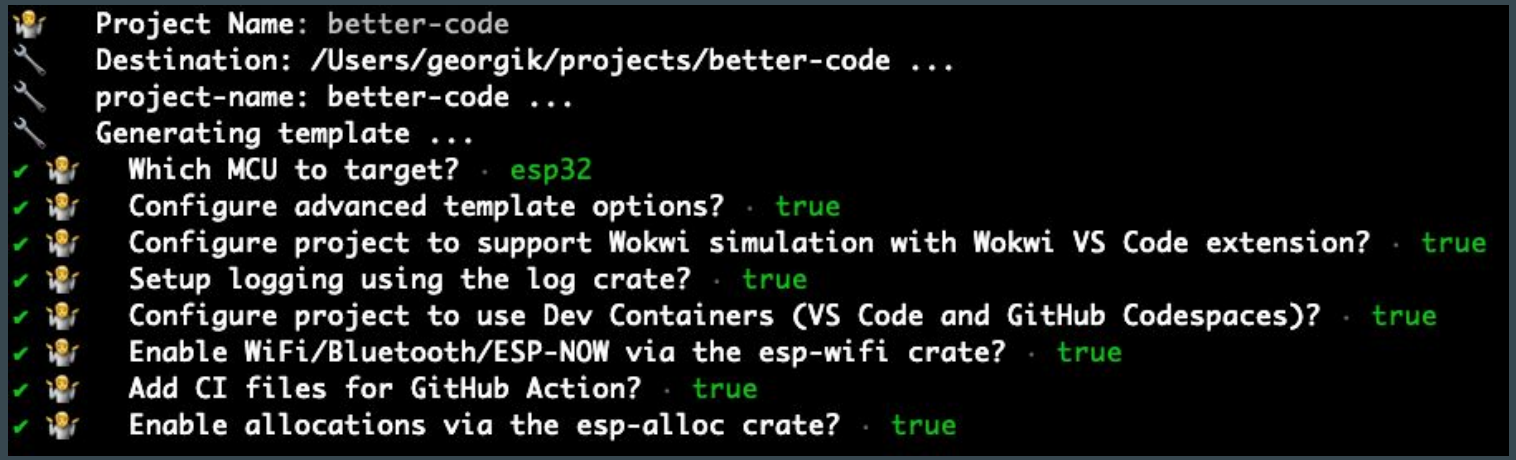
- leverage APIs of ESP-IDF
- pick exact version of ESP-IDF (not master branch)
- cargo generate --vcs none --git https://github.com/esp-rs/esp-idf-template cmake
- https://github.com/esp-rs/esp-idf-template

no_std

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

Starting from template

cargo generate esp-rs/esp-template



cargo build --release

cargo espflash flash --release --monitor

Rust Embedded Graphics

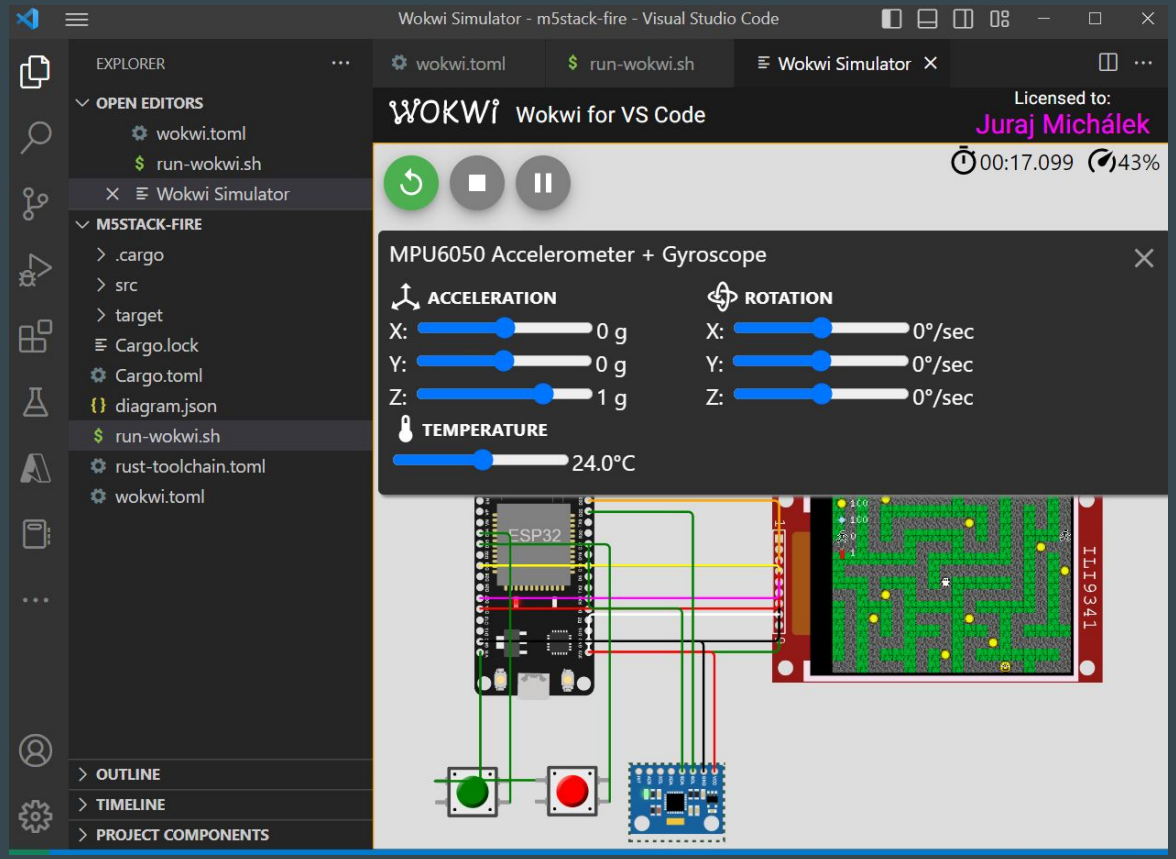
https://github.com/embedded-graphics

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

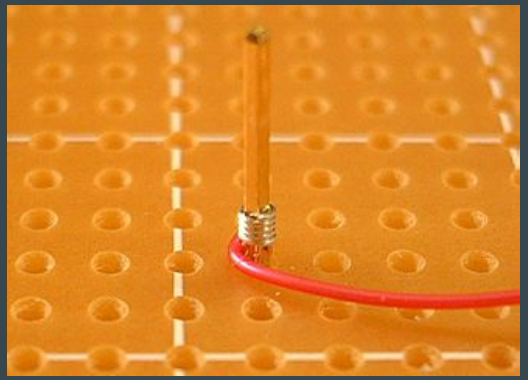
Example: https://github.com/georgik/esp32-spooky-maze-game

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=U96aLAwIsLpsPGTW



picture from Wikipedia: https://en.wikipedia.org/wiki/Wire_wrap

LVGL and 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/slint/slint

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

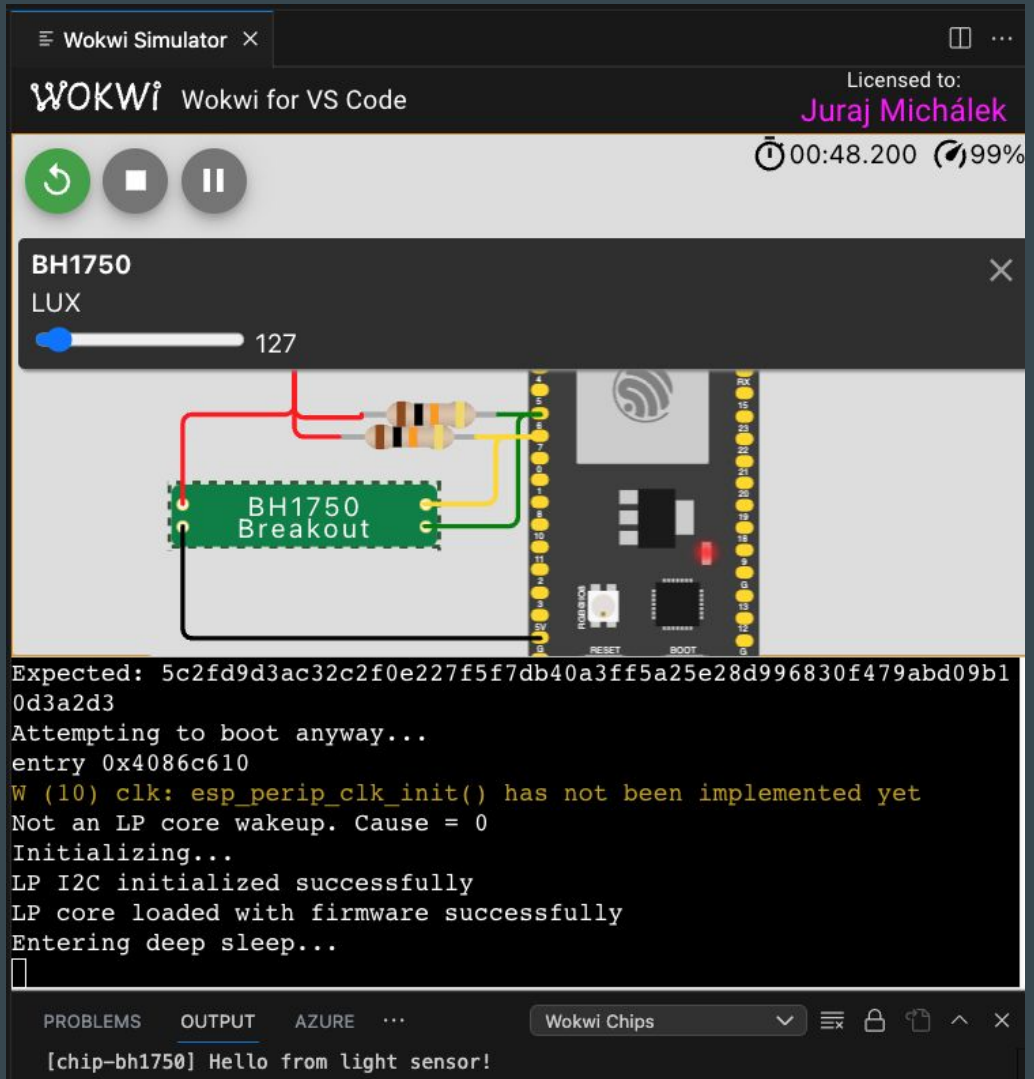
- https://www.youtube.com/watch?v=FA1jqZLig4s

Second important feature:

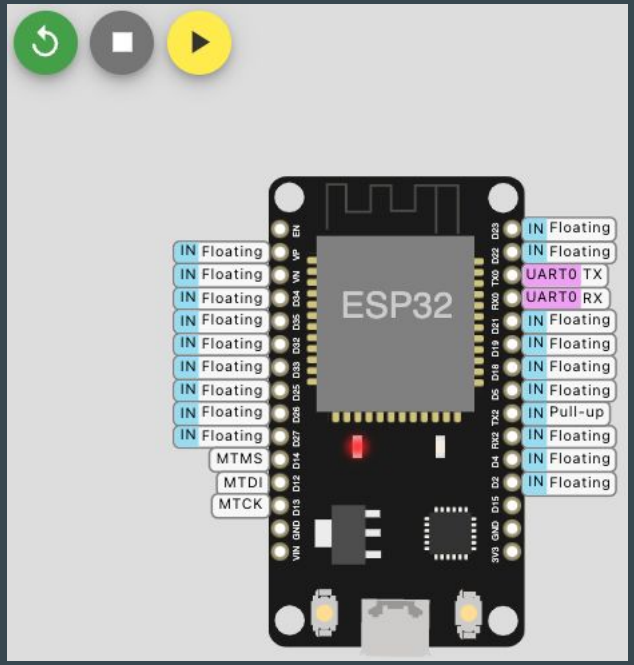
- Low Power Core - 20 MHz

Wokwi and Low Power Core simulation ESP32-C6

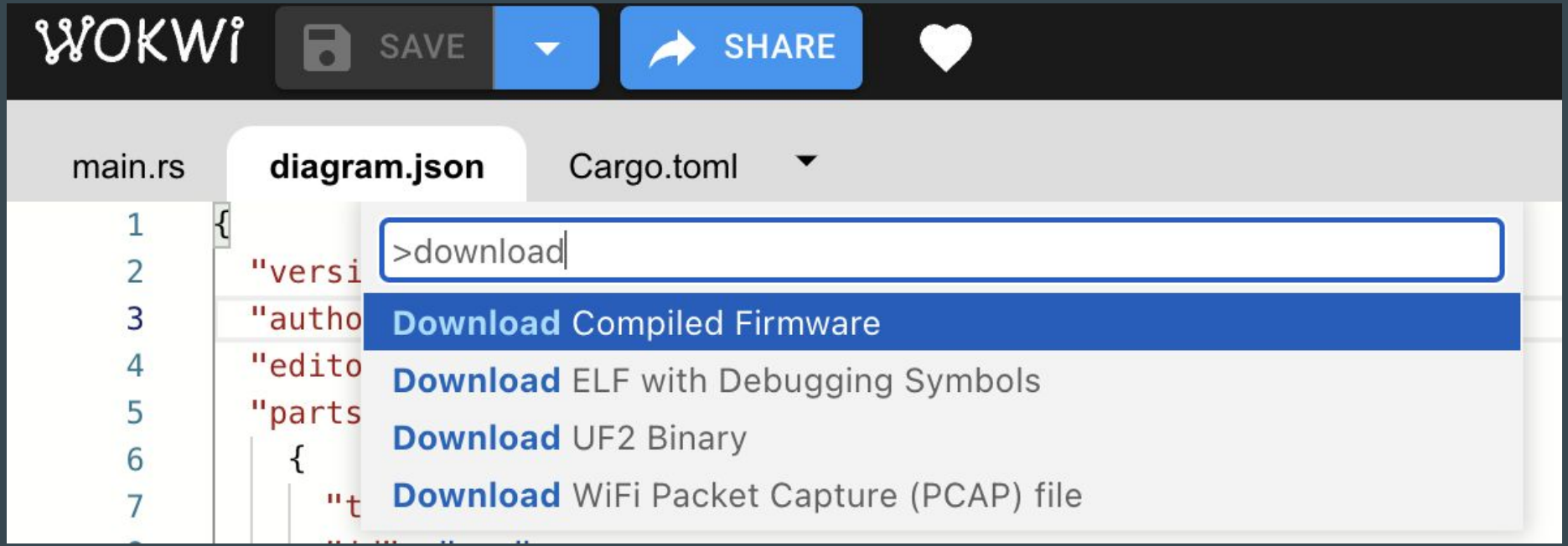
https://github.com/wokwi/esp32c6-i2c-lp



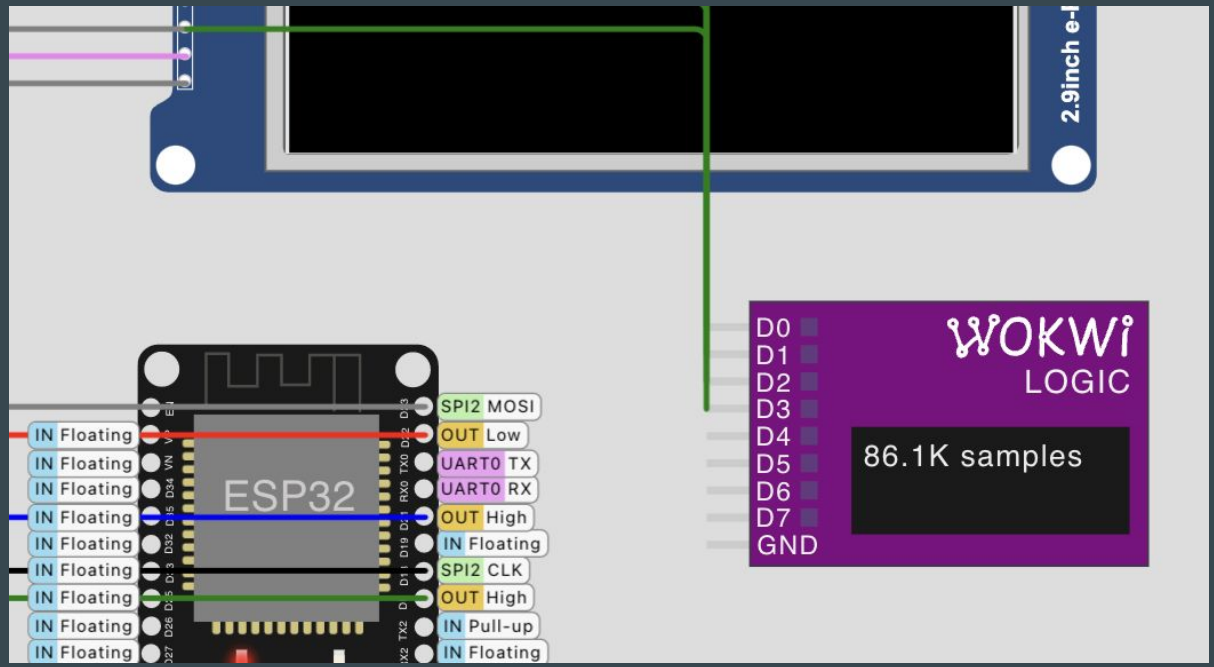
Pause button - see state of GPIOs



Wokwi F1 menu (download binary and many more)

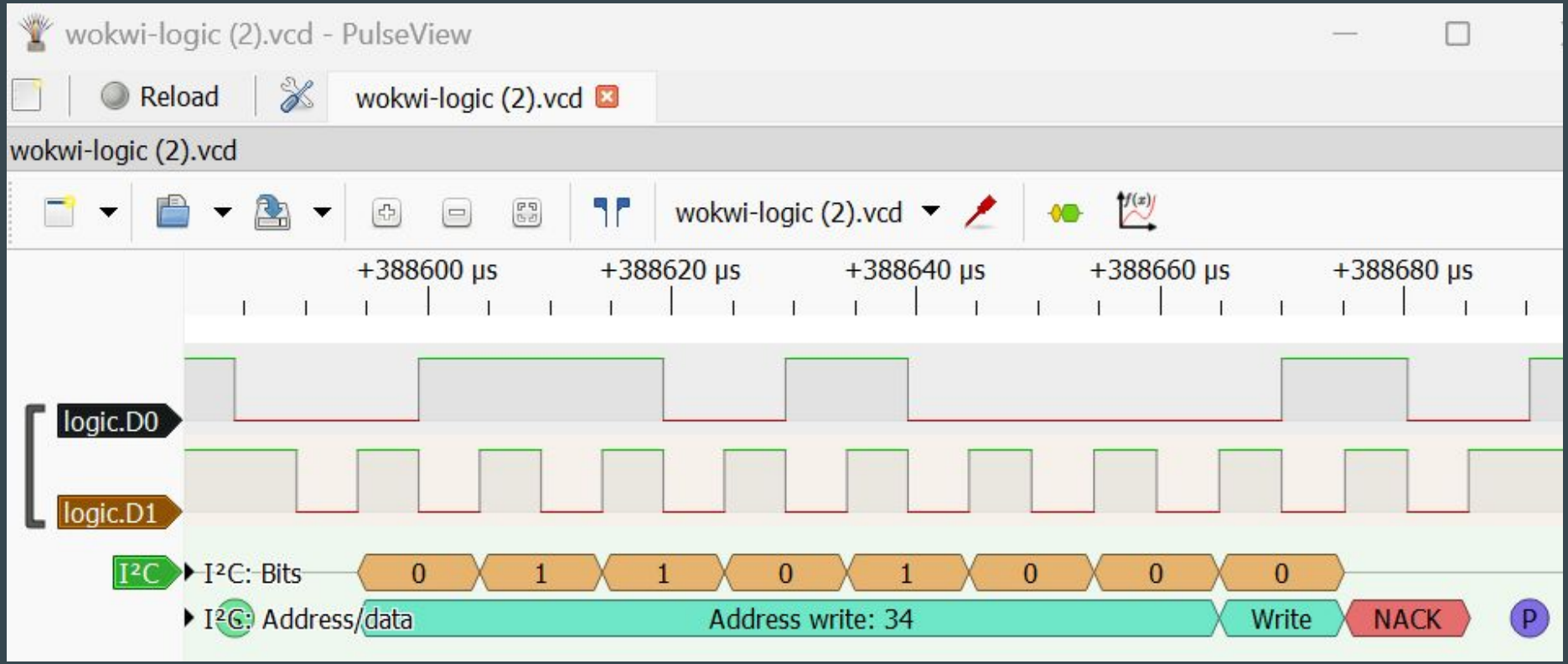


Wokwi Logic Analyzer



Hit Stop to download VCD file for PulseView

Wokwi + PulseView - examples of I2C



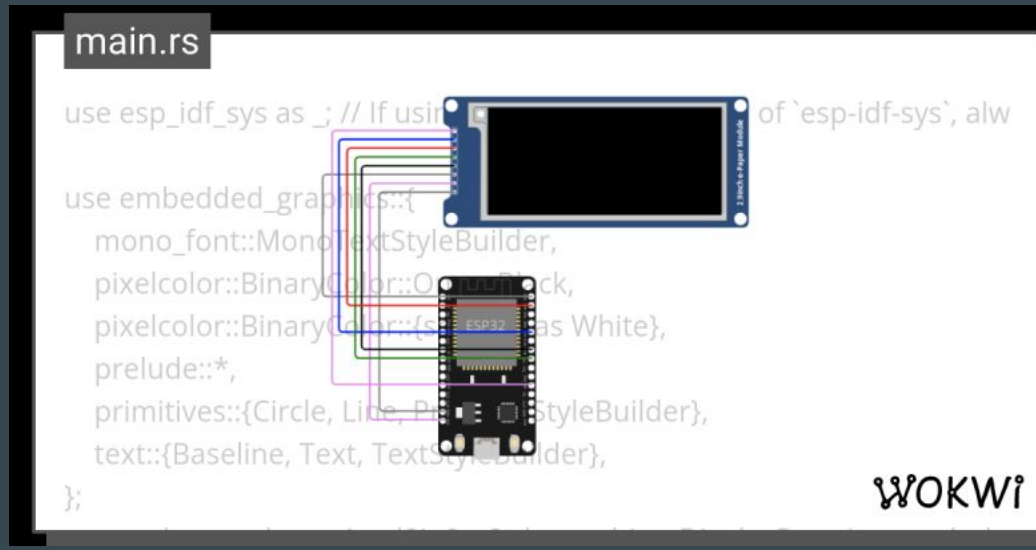
Custom chips: Rust + e-Paper + ESP32

Simulate your own HW

Wokwi: Getting Started with the Wokwi Custom Chips C API

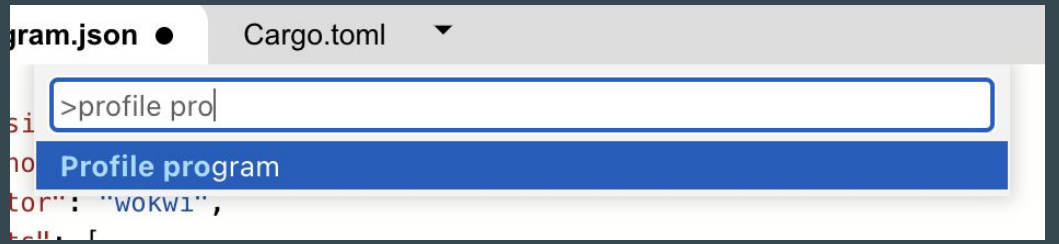
https://docs.wokwi.com/chips-api/getting-started

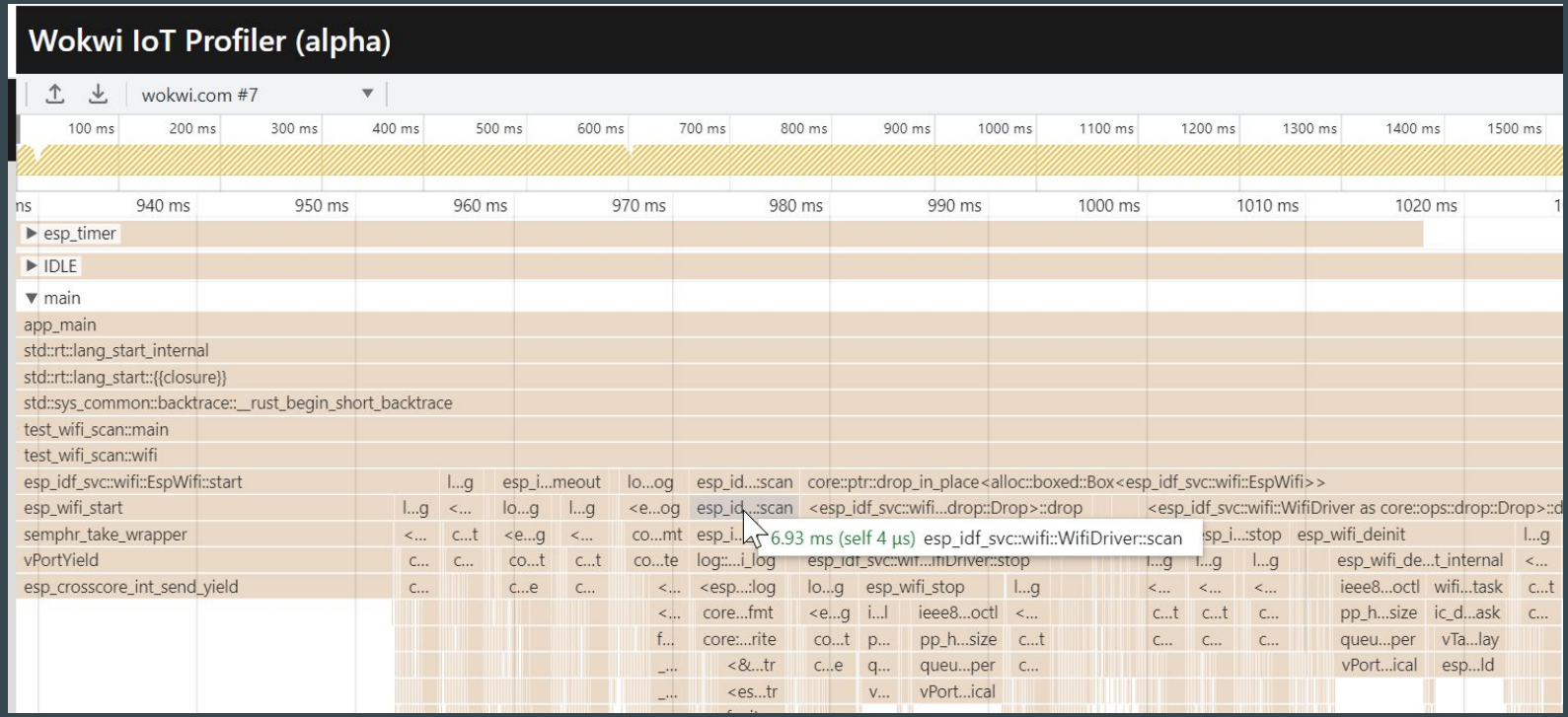
https://wokwi.com/projects/366167936725307393



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

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

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

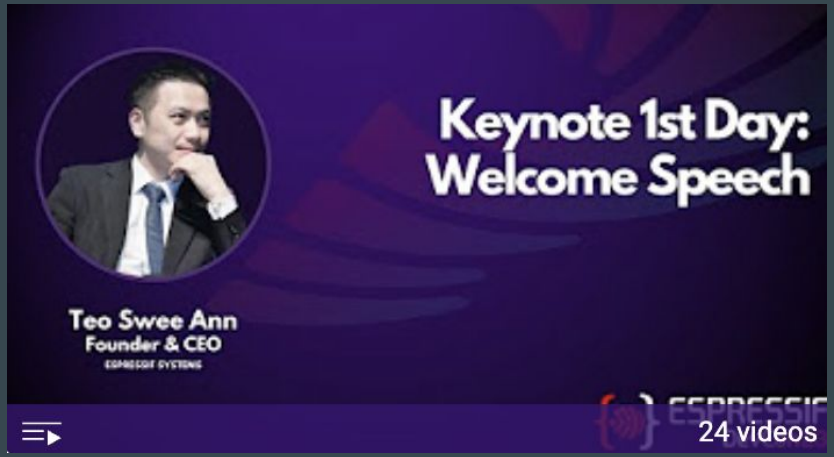
Qt for MCU

https://www.qt.io/product/develop-software-microcontrollers-mcu

Qt for MCU supports ESP32-S3

- students license - registration requires just email

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

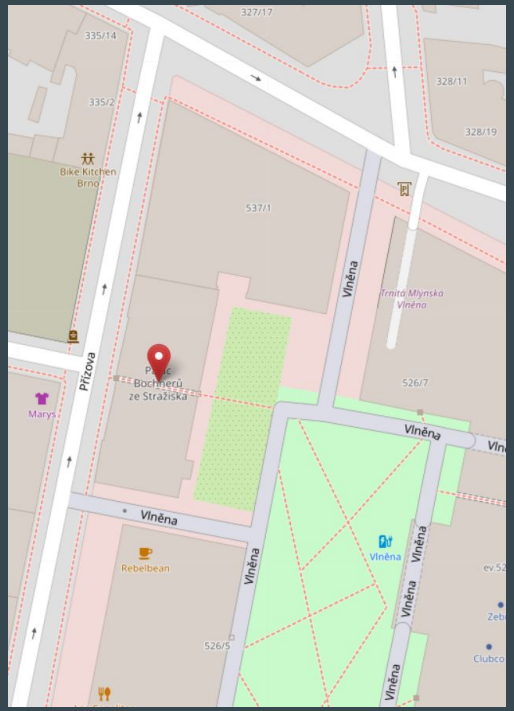
https://youtu.be/qeEmJ_-6fPg

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

https://youtu.be/OPPPdqoDBQs

Where you can find us?

Přízova 3, 602 00 Brno



https://makerfaire.cz/praha/

Embedded World 2024

Meet us in Nuremberg, Germany - 9th - 11th April 2024

Visit us in Brno

Espressif Systems (Czech) s.r.o.

Přízova 3, 602 00 Brno

Czechia, Europe