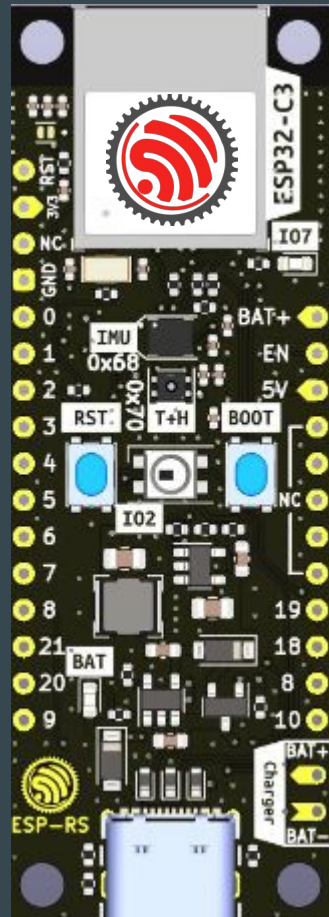


Rust Embedded Development on ESP32 and basics of Async with Embassy



2023-02-08
Rust CZ/SK Meetup - Brno
Juraj Michálek, Juraj Sadel - Espressif Systems



ESP-RS organisation <https://github.com/esp-rs>



esp-rs

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

Rust ESP32 Community meeting

Regular meeting every two weeks

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

Community based effort to make Rust better

The Rust on ESP Book - <https://esp-rs.github.io/book/>

Awesome ESP Rust - <https://github.com/esp-rs/awesome-esp-rust>

- submit your project :-)

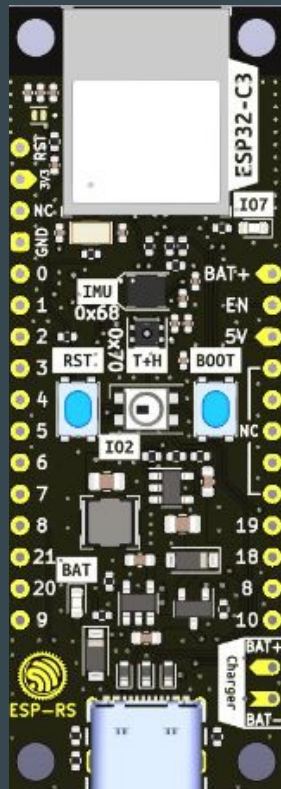
Designing Open Hardware - esp-rust-board

KiCad templates

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

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

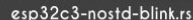
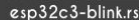
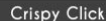
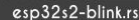
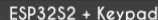
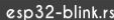
<https://www.espressif.com/en/products/devkits>



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

<https://youtu.be/TKe4MgD6O8o>

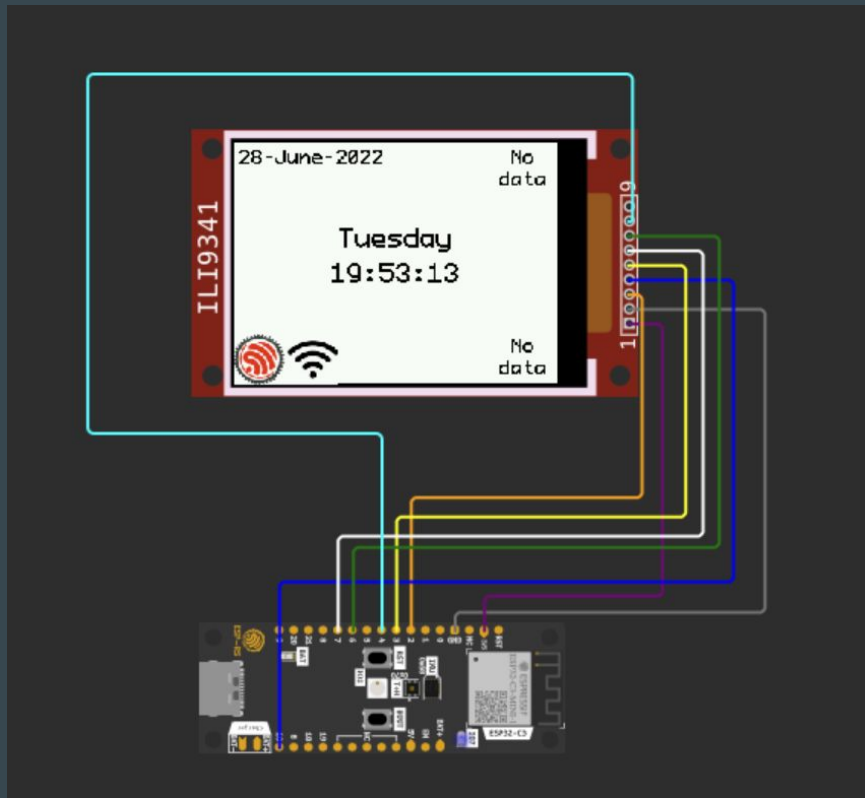
[+ NEW PROJECT](#)



GitPod.io + Wokwi.com



Open in Gitpod



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

Exploring what's possible with Rust

Bare Metal: <https://github.com/esp-rs/esp-hal>

- ESP32 - [esp32-hal/examples](https://github.com/esp-rs/esp-hal/tree/master/examples) - Xtensa
- ESP32-C2 - [esp32c2-hal/examples](https://github.com/esp-rs/esp-hal/tree/master/examples) - RISC-V
- ESP32-C3 - [esp32c3-hal/examples](https://github.com/esp-rs/esp-hal/tree/master/examples) - RISC-V
- ESP32-S2 - [esp32s2-hal/examples](https://github.com/esp-rs/esp-hal/tree/master/examples) - Xtensa
- ESP32-S3 - [esp32s3-hal/examples](https://github.com/esp-rs/esp-hal/tree/master/examples) - Xtensa

STD: <https://github.com/esp-rs/esp-idf-hal/tree/master/examples>

Create new project Rust project

Bare metal:

```
cargo generate https://github.com/esp-rs/esp-template
```

STD with ESP-IDF:

```
cargo generate --vcs none --git https://github.com/esp-rs/esp-idf-template cargo
```

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>

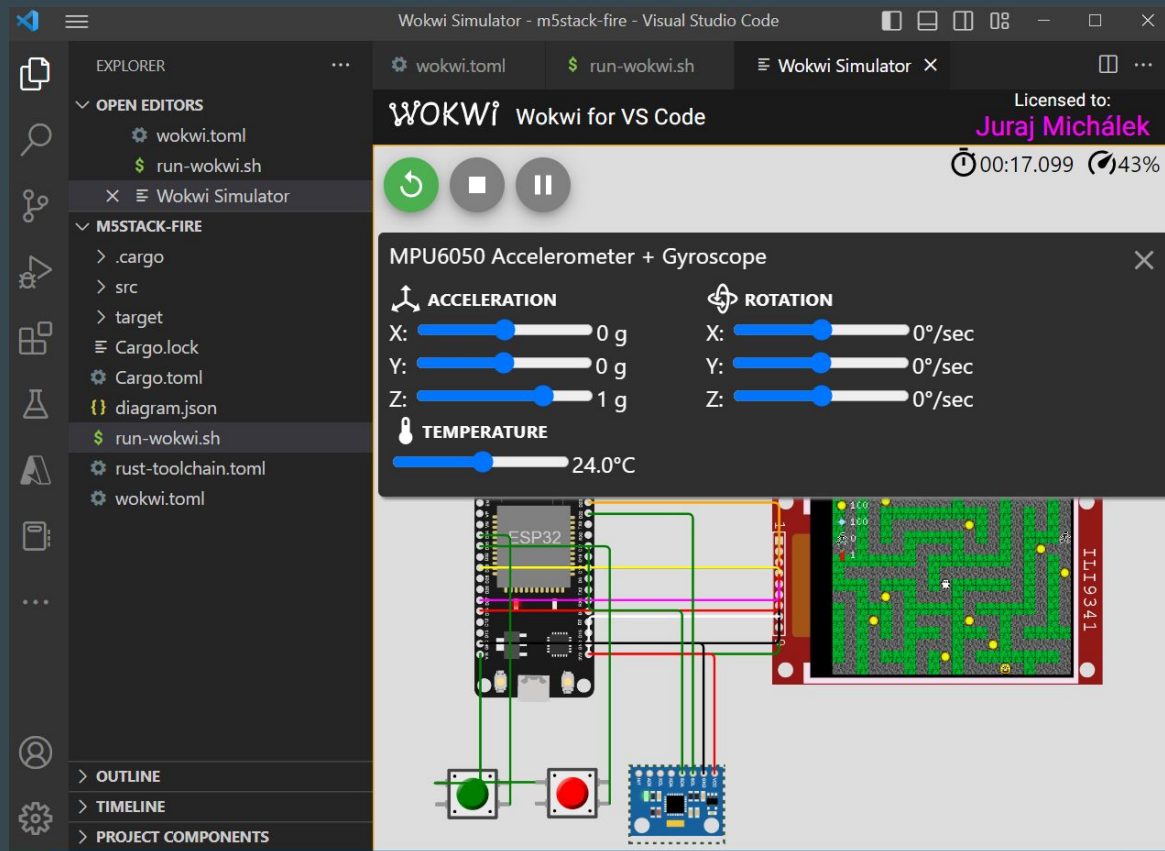
Wokwi - VS Code Plugin

Add wokwi.toml and diagram.json
to your project

cd m5stack-fire

cargo build --release --no-default-features --features "wokwi"

CTRL+Shift+P - Wokwi: Start Simulator



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

Development options

- Local
 - bootstrap whole env with “espup install” - <https://github.com/esp-rs/espup#installation> (written in Rust)
 - Windows Online idf-installer - <https://github.com/espressif/idf-installer/releases/tag/online-2.20> (includes VC Tools)
 - bootstrap with sh/pwsh (older method) - <https://github.com/esp-rs/rust-build#download-installer>
- Local in container
 - Podman, Docker, Lima
 - with VS Code Remote Container extension
- Cloud in GitPod.io or CodeSpaces
 - EDC22 Day 2 Talk 11 Just Few dot files in Repo or Paradigm Shift to Cloud-based Embedded Development - <https://youtu.be/WAwfRyOy7N8>
- Wokwi.com

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();  
        }  
    }  
}
```

async/.await

- Non-blocking operations
- Transforms each async function into future object
- When future blocks I/O it yields -> executor selects different future to execute
- No guessing which future is ready

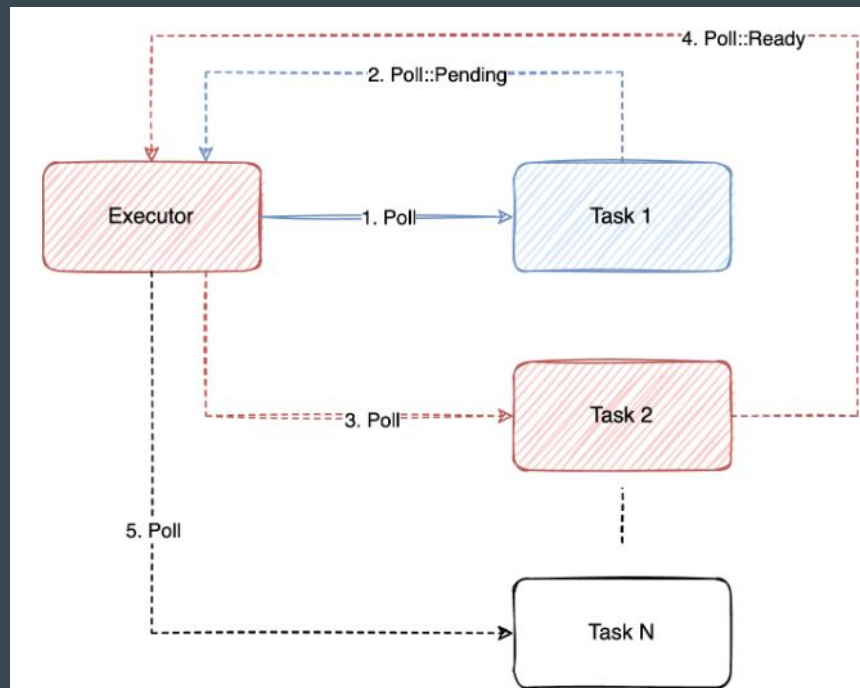
```
trait SimpleFuture {  
    type Output;  
    fn poll(&mut self, wake: fn()) -> Poll<Self::Output>;  
}  
  
enum Poll<T> {  
    Ready(T),  
    Pending,  
}
```

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>

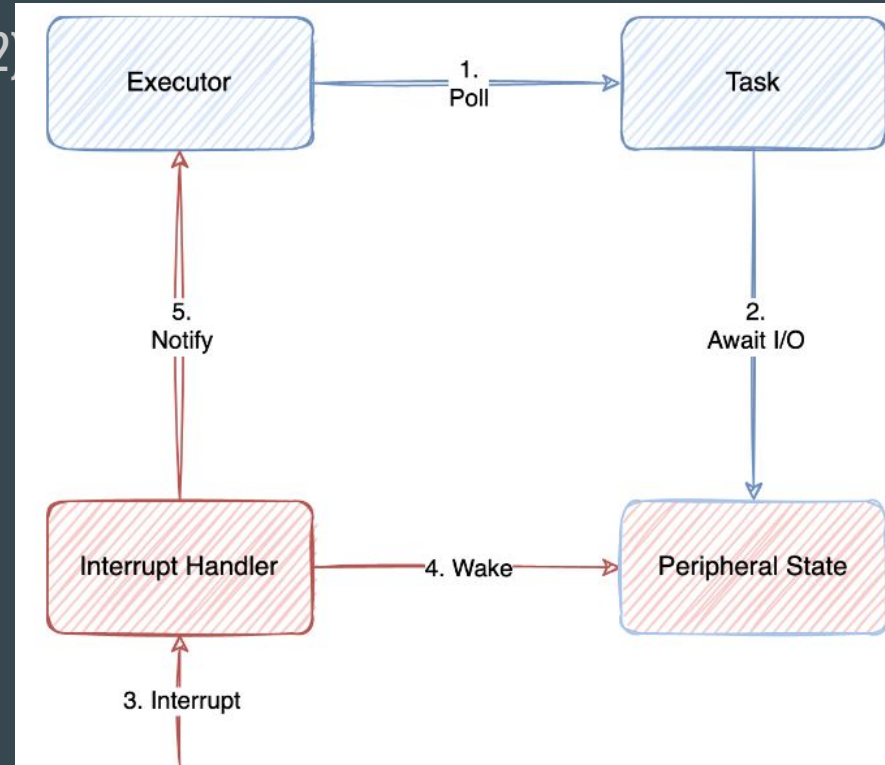
Executor

- Queue of tasks, when created it's polled (1)
- Executing until gets blocked (awaiting async function) (2)
- Executes other task (3)



Interrupts

- Task is polled and executing (1)
- Task uses a peripheral to perform an operation(2)
- Interrupt occurred while awaiting (3)
- HAL ensures that interrupt signal updates the peripheral state (4)
- Executor poll the task (5)



Rust language support

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

<https://youtu.be/OPPPdgoDBQs>



Training Embedded for ESP32-C3 by Ferrous Systems

Training: <https://ferrous-systems.com/training/#package-espressif-beginner-training>

Material: <https://espressif-trainings.ferrous-systems.com/>

GitHub: <https://github.com/ferrous-systems/espressif-trainings/>



ferrous systems

**Some Inspiration
(not necessary in Rust)**

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

Matter-AllClustersApp-TE9

Matter-AllClustersApp-M5Stack-TE9

Matter-LightingApp-TE9

RainMaker-Fan

RainMaker-GPIO

RainMaker-HomekitSwitch

RainMaker-LedLight

RainMaker-MultiDevice

RainMaker-Switch

RainMaker-TemperatureSensor

Flash

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

☐	Index	Name	MPN	Marketing Status	Type	Wi-Fi
☐	1	ESP32-S3	ESP32-S3	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
☐	2	ESP32-S3	ESP32-S3R2	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
☐	3	ESP32-S3	ESP32-S3R8	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
☐	4	ESP32-S3	ESP32-S3R...	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
☐	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 - <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 3rd party and community:

- Visual Studio with VisualGDB - <https://visualgdb.com/>
- CLion - <https://www.jetbrains.com/clion/>

Espressif Developer Conference 2022 - recording



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

OSes and integration



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

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



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>

Further languages and frameworks

Arduino - Maker choice

- warning: limitation on production due to licenses!

VM based:

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

CTAG-TBD

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

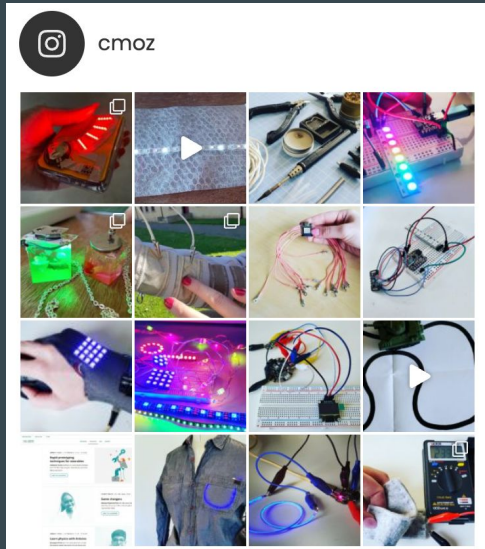
<https://github.com/ctag-fh-kiel/ctag-tbd>



Wearables

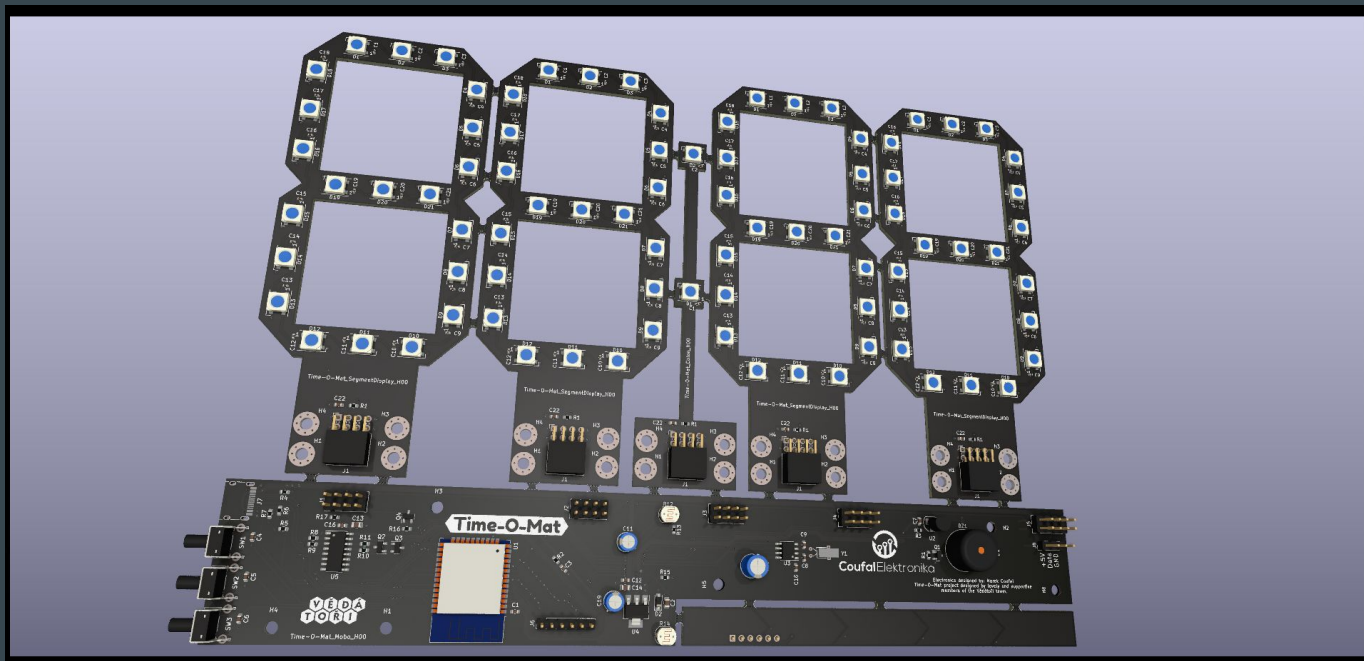
The Ultimate Guide to Informed Wearable Technology

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

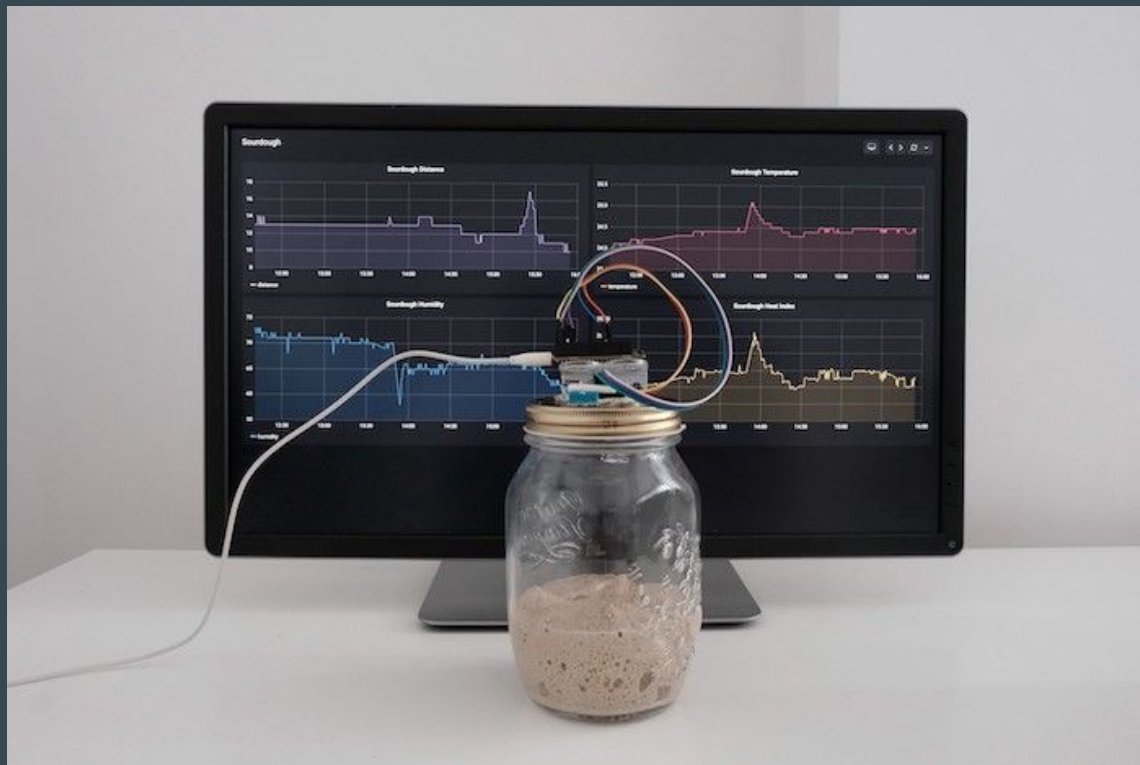


Time-O-Mat - built at summer camp

<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 :)

Embedded World 2023

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



See you in Linz in 2023. We're planning ESP32 Rust Workshop ;-)