Oh crab! 

My watch speaks Rust

Pierre-Yves Aillet

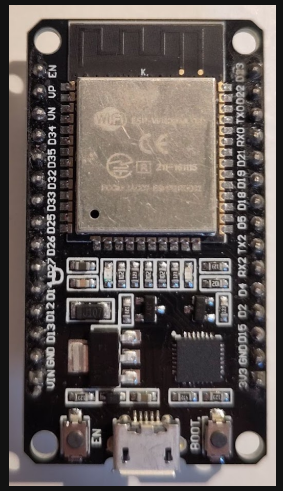
Quelques rappels et définitions sur l’embarqué

Pas d’OS "classique"

- GPIO: General Purpose Input/Output
- I2C: Inter-Integrated Circuit
- SPI: Serial Peripheral Interface
- UART: Universal Asynchronous Receive/Transfer

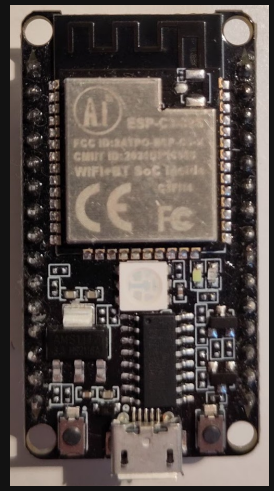
ESP32

ESP32 / ESP32-S*



Architecture: xtensa-lx6
Licensed from Tensilica

ESP32-C*



Architecture: RISC-V
Open standard instruction set architecture

🦀 Rappels sur Rust

> **Rust** est un langage de programmation compilé multi-paradigme conçu et développé par Mozilla Research depuis 2010”, puis par la Rust Foundation

Wikipedia

🦀 Rappels sur Rust

- Memory safety - ownership
- Type safety - explicit error handling, safe refactoring
- Zero Cost Abstractions
- Fearless concurrency
- Great Developer eXperience

🦀 Rappels sur Rust

Structure d'un project

```
.
├── Cargo.toml
└── src
    └── main.rs
```

Manipulations du projet

```
# Construction du projet
$ cargo build

# Lancement du projet
$ cargo run

# Formattage du code
$ cargo fmt
```

🤔 Pourquoi Rust est adapté pour de l'embarqué ?

- no_std
  - core
  - alloc
  - std

hello

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

use esp32_hal::{clock::ClockControl, pac::Peripherals, prelude::*, Delay};
use esp_backtrace as _;
use std::fmt::println; // <= This is not possible with no_std :(

#[xtensa_lx_rt::entry]
fn main() -> ! {
    let peripherals = Peripherals::take().unwrap();
    let system = peripherals.DPORT.split();
    let clocks = ClockControl::boot_defaults(system.clock_control).freeze();

    let mut delay = Delay::new(&clocks);

    loop {
        println!("Hello world!");
        delay.delay_ms(1000u32);
```

🤔 Pourquoi Rust est adapté pour de l'embarqué ?

- no_std
- Typestate pattern

`blinky`

```rust
fn main() -> ! {
    let peripherals = Peripherals::take().unwrap();
    let system = peripherals.DPORT.split();
    let clocks = ClockControl::boot_defaults(system.clock_control).freeze();

    let mut delay = Delay::new(&clocks);

    let io = IO::new(peripherals.GPIO, peripherals.IO_MUX);
    let mut led = io.pins.gpio4.into_push_pull_output();

    loop {
        led.toggle().unwrap();
        delay.delay_ms(1000u32);
        println!("Led is high: {}", led.is_high());
    }
}
```

🤔 Pourquoi Rust est adapté pour de l'embarqué ?

- no_std
- Typestate pattern

```rust
struct Output {}
struct Input {}
struct Gpio4<STATE> {
  _state: PhantomData<STATE>,
}

impl<T> Gpio4<T> {
  fn into_input(self) -> Gpio4<Input> {
    Gpio4 { _state: PhantomData }
  }
  fn into_output(self) -> Gpio4<Output> {
    Gpio4 { _state: PhantomData }
  }
}

impl Gpio4<Input> {
  fn is_high() -> bool { unimplemented!() }
}

impl Gpio4<Output> {
  fn toggle(&mut self) {}
}
```

🤔 Pourquoi Rust est adapté pour de l'embarqué ?

- no_std
- Typestate pattern
- Ownership, &mut unique

```
struct Blinker {
  led: Gpio4<Output>
}

impl Blinker {
  fn run() {
    // [...]
  }
}

fn main() -> ! {
    // [...]

    let mut led = io.pins.gpio4.into_push_pull_output();
    let blinker = Blinker { led };
    blinker.run();

    led.into_input();

    // [...]
}
```

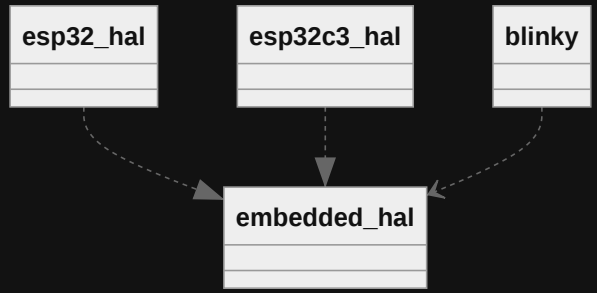
🤔 Pourquoi Rust est adapté pour de l’embarqué ?

- no_std
- Typestate pattern
- Ownership, &mut unique
- Gestion des dépendances et compilation conditionnelle

```
[package]
name = "blinky"
version = "0.1.0"
authors = ["Pierre-Yves Aillet <pyaillet@gmail.com>"]
edition = "2021"
license = "MIT OR Apache-2.0"

[dependencies]
esp32-hal = "0.5.0"

esp-backtrace = { version = "0.2.0", features = ["esp32"] }

xtensa-lx-rt = { version = "0.13.0", features = ["esp32"] }

[features]
default = ["rt"]
rt = ["xtensa-lx-rt"]
```

🤔 Pourquoi Rust est adapté pour de l’embarqué ?

- no_std
- Typestate pattern
- Ownership, &mut unique
- Gestion des dépendances et compilation conditionnelle
- Abstractions



🤔 Pourquoi Rust est adapté pour de l'embarqué ?

- no_std
- Typestate pattern
- Ownership, &mut unique
- Gestion des dépendances et compilation conditionnelle
- Abstractions

```rust
// in embedded-hal
pub trait OutputPin {
    fn set_high(&mut self);
}

// in esp32-hal
impl OutputPin for Gpio4 {
    fn set_high(&mut self) {
        // specific code
    }
}

// in blinky
fn blink(mut led: impl OutputPin) {
    led.set_high();
}
```

😕 Mais...

- Il manque certaines abstractions
  - de base : interrupt, DMA, …
  - de haut niveau : Bluetooth, WiFi[1]

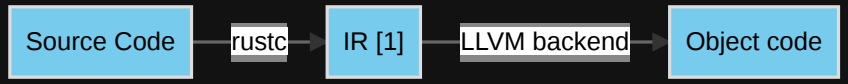
[1] c’est en cours… embedded_svc

⌚ Qu’est-ce que c’est que cette montre ?

La construction du firmware, comment ça fonctionne ?

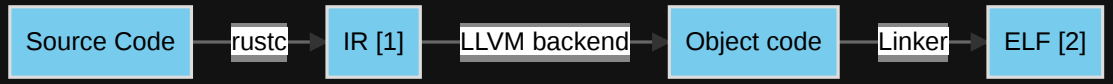
Source Code

La construction du firmware, comment ça fonctionne ?



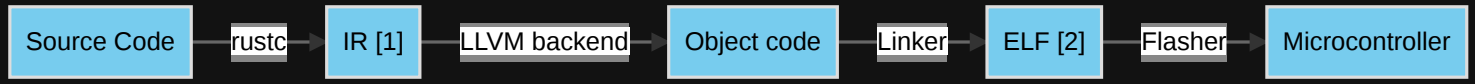
- [1] IR: Intermediate Representation

La construction du firmware, comment ça fonctionne ?



- [1] IR: Intermediate Representation
- [2] ELF: Executable and Linkable Format

La construction du firmware, comment ça fonctionne ?



- [1] IR: Intermediate Representation
- [2] ELF: Executable and Linkable Format

🛠️ Tooling

- toolchains et crosscompilation

```
❯ rustup target list
aarch64-unknown-linux-gnu
aarch64-unknown-linux-musl
aarch64-unknown-none
aarch64-unknown-none-softfloat
riscv32i-unknown-none-elf
thumbv6m-none-eabi
wasm32-unknown-emscripten
wasm32-unknown-unknown
wasm32-wasi
x86_64-apple-darwin (installed)
x86_64-apple-ios
x86_64-fortanix-unknown-sgx
x86_64-fuchsia
x86_64-linux-android
```

🛠️ Tooling

L’architecture xtensa n’est pas supportée par le compilateur rustc…

Parce que le backend xtensa n’existe pas côté llvm…
MAIS !
C’est en cours…
Depuis 3 ans…

Malgré tout, il y a eu des progrès prometteurs dernièrement 🎉

– https://discourse.llvm.org/t/rfc-request-for-upstream-tensilica-xtensa-esp32-backend/65355

– https://lists.llvm.org/pipermail/llvm-dev/2019-March/130796.html

🛠️ Tooling

- Toolchain custom avec un fork maintenu de llvm et rustc
  - https://github.com/esp-rs/rust-build.git
  - https://github.com/espressif/llvm-project
  - https://github.com/esp-rs/rust
- Installation de la toolchain custom avec un outil dédié espup :

```
cargo install espup --git https://github.com/esp-rs/espup
espup install
. export-esp.sh
```

🛠️ Tooling

Création d’un nouveau projet

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

Avec tout ça on est prêts ?

Pas vraiment...

La gestion du WiFi et du Bluetooth, c’est compliqué…

FreeRTOS et esp-idf à la rescousse

> FreeRTOS is a market-leading real-time operating system (RTOS) for microcontrollers and small microprocessors.

https://freertos.org

> ESP-IDF is the official development framework for the ESP32, ESP32-S and ESP32-C Series SoCs.

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

FreeRTOS et esp-idf à la rescousse

Application Example

GPIO output and input interrupt example: peripherals/gpio/generic_gpio.

API Reference - Normal GPIO

Header File

- components/driver/include/driver/gpio.h

Functions

esp_err_t **gpio_config**(*const* gpio_config_t *pGPIOConfig)

GPIO common configuration.

```
Configure GPIO's Mode,pull-up,PullDown,IntrType
```

Parameters: **pGPIOConfig** – Pointer to GPIO configure struct

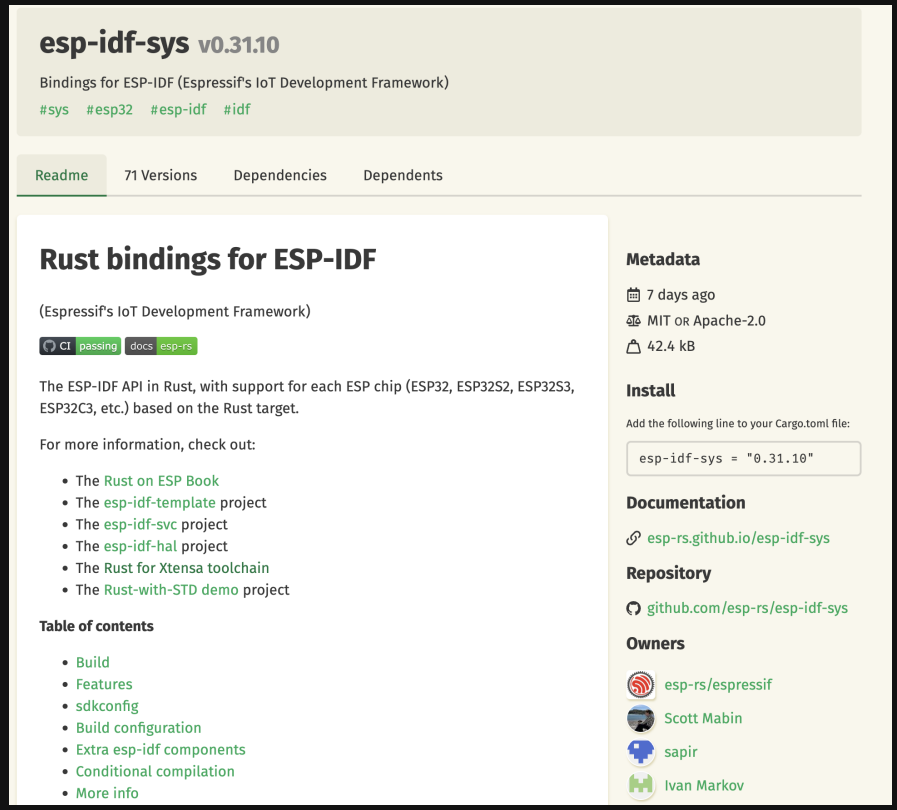
Returns:

- ESP_OK success
- ESP_ERR_INVALID_ARG Parameter error

FreeRTOS et esp-idf à la rescousse

- SPI/I2C/I2S
- Gestion des tasks
- Eventloop
- Bluetooth/BLE
- WiFi
- Timer
- …

🛠️ Tooling



esp-idf-sys v0.31.10

Bindings for ESP-IDF (Espressif's IoT Development Framework)

#sys #esp32 #esp-idf #idf

Readme | 71 Versions | Dependencies | Dependents

Rust bindings for ESP-IDF

(Espressif's IoT Development Framework)

CI passing docs esp-rs

The ESP-IDF API in Rust, with support for each ESP chip (ESP32, ESP32S2, ESP32S3, ESP32C3, etc.) based on the Rust target.

For more information, check out:

- The Rust on ESP Book
- The esp-idf-template project
- The esp-idf-svc project
- The esp-idf-hal project
- The Rust for Xtensa toolchain
- The Rust-with-STD demo project

Table of contents

- Build
- Features
- sdkconfig
- Build configuration
- Extra esp-idf components
- Conditional compilation
- More info

Metadata

7 days ago
MIT OR Apache-2.0
42.4 kB

Install

Add the following line to your Cargo.toml file:

```
esp-idf-sys = "0.31.10"
```

Documentation

esp-rs.github.io/esp-idf-sys

Repository

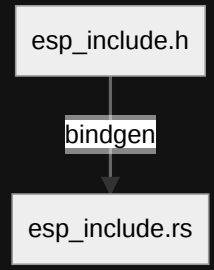
github.com/esp-rs/esp-idf-sys

Owners

esp-rs/espressif
Scott Mabin
sapir
Ivan Markov

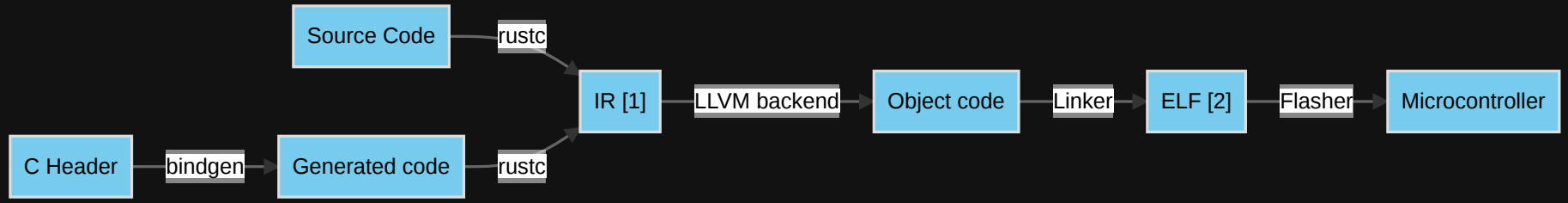
🛠️ Tooling

- toolchains et crosscompilation
- bindgen



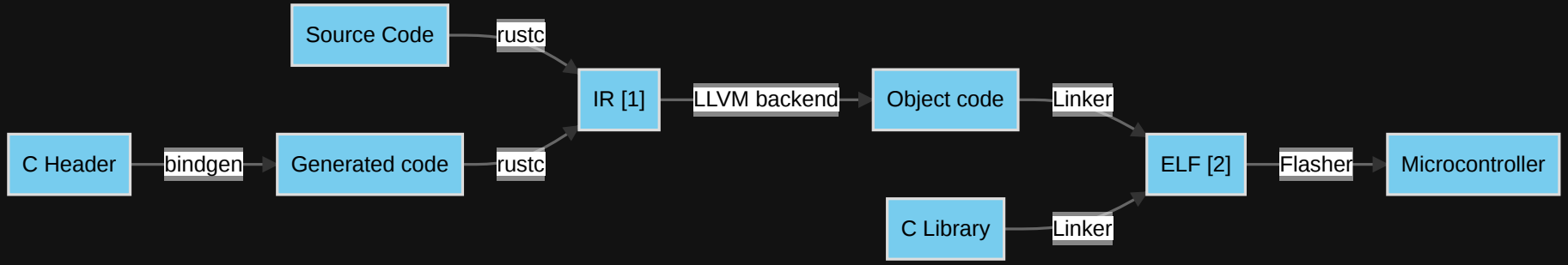
🛠️ Tooling

- toolchains et crosscompilation
- bindgen



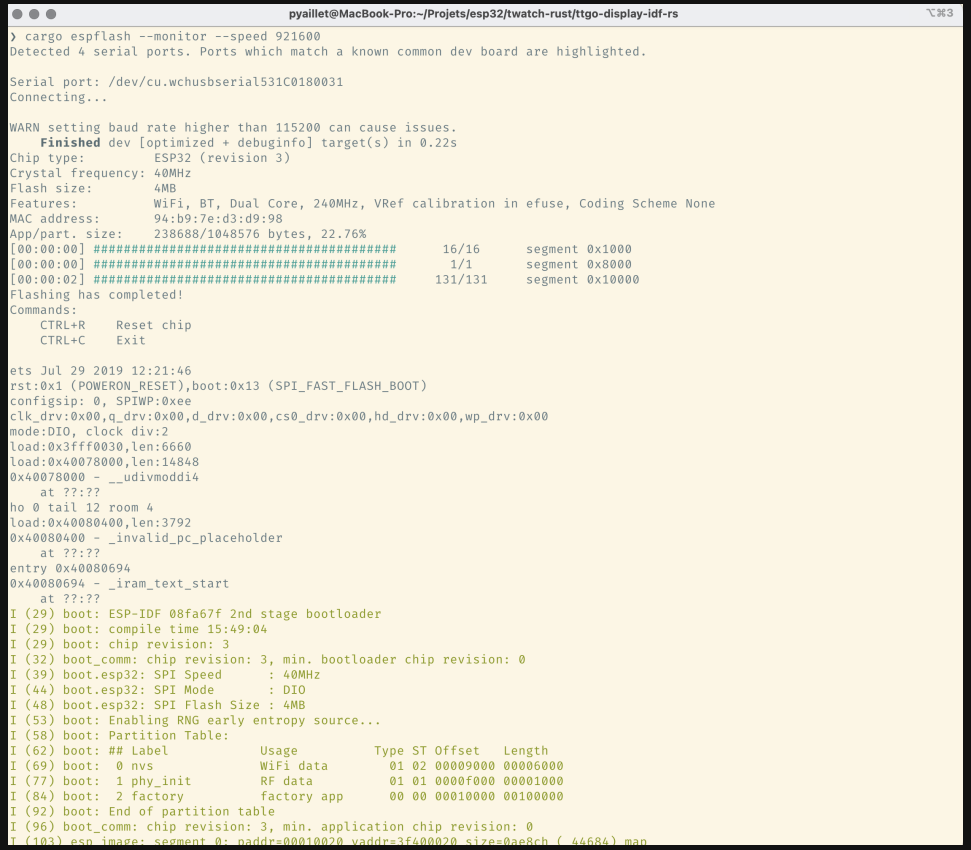
🛠️ Tooling

- toolchains et crosscompilation
- bindgen
- build.rs

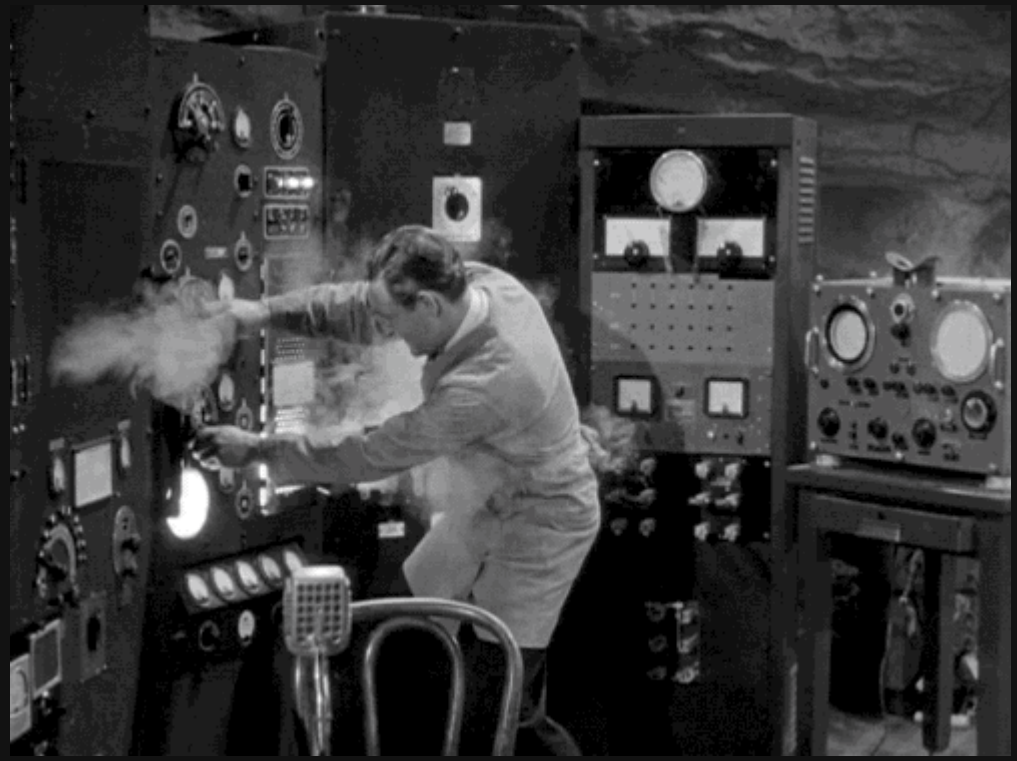


🛠️ Tooling

- toolchains et crosscompilation
- bindgen
- build.rs
- espflash et espmonitor

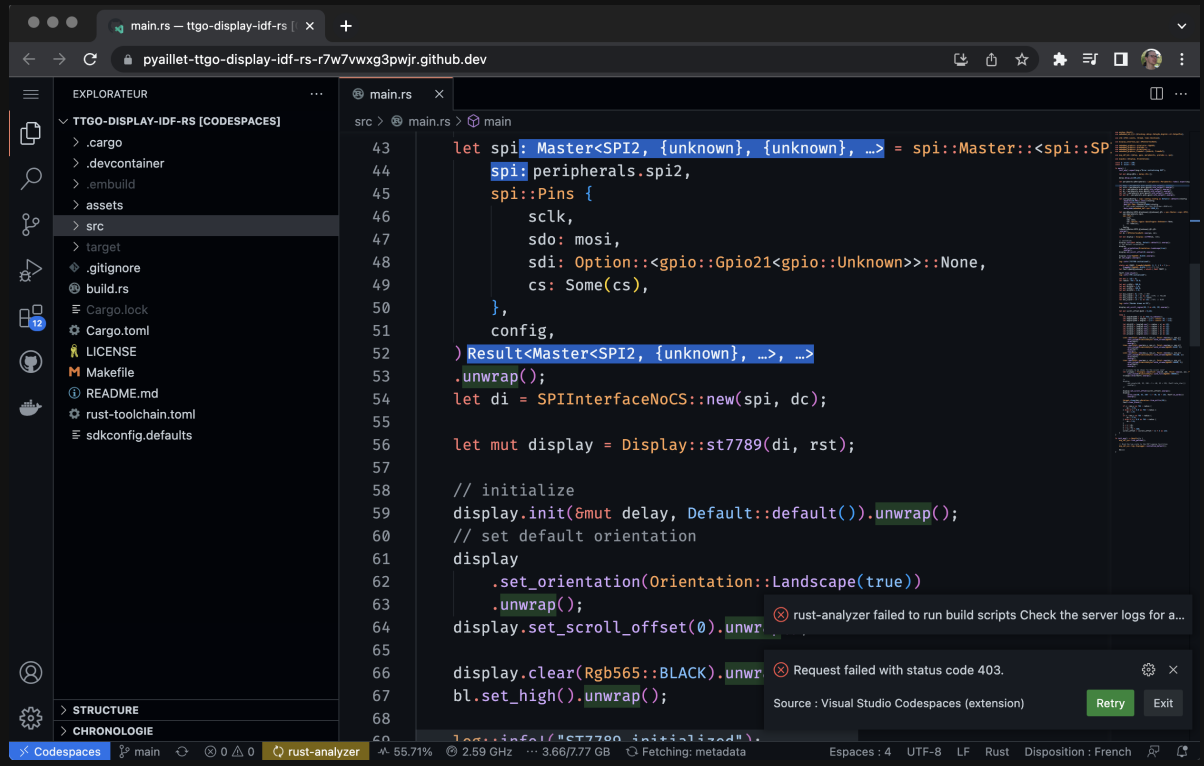


Démo



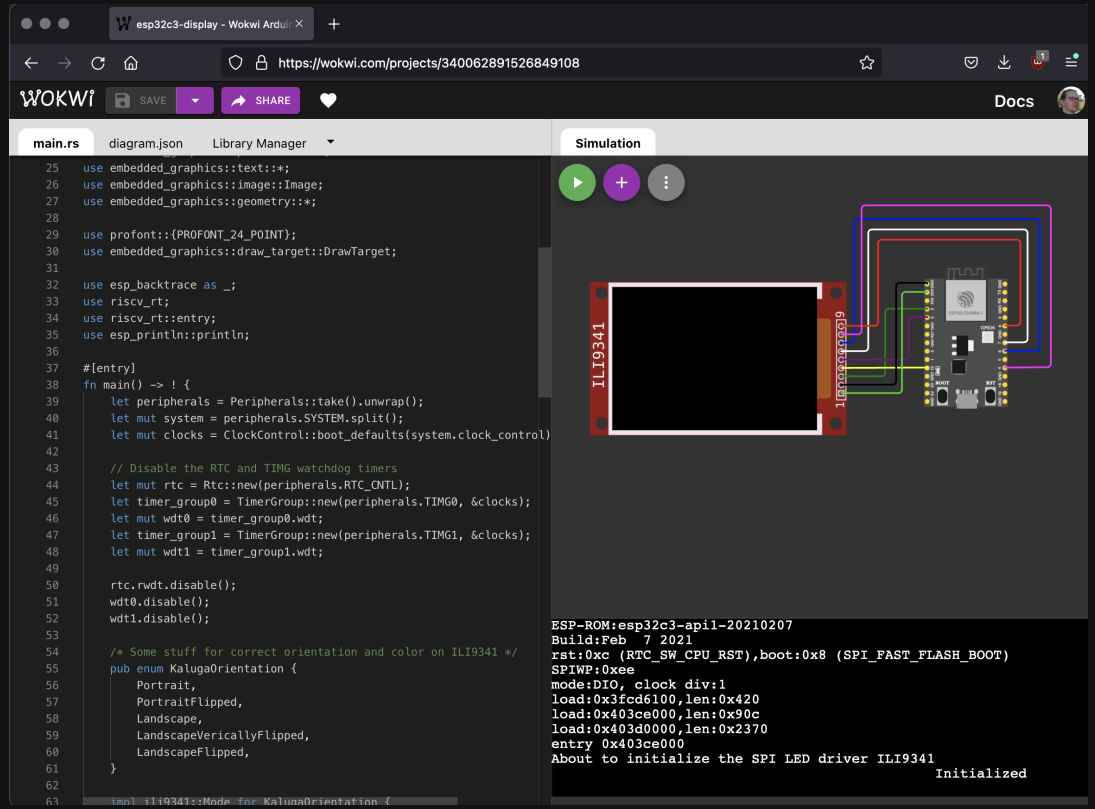
🛠️ Tooling

- toolchains et crosscompilation
- bindgen
- build.rs
- espflash et espmonitor
- devcontainer + esp-web-flash



🛠️ Tooling

- toolchains et crosscompilation
- bindgen
- build.rs
- espflash et espmonitor
- devcontainer + esp-web-flash
- wokwi



🔮 Next?

- Sur la montre…
  - BLE et connexion avec le téléphone
  - Détection d'activité
- Rust embedded
  - async/await avec embassy - https://github.com/embassy-rs/embassy
  - Autres cartes - Apache MyNewt + NimBLE - https://mynewt.apache.org/
  - RTIC - Real Time Interrupt driven Concurrency - https://rtic.rs/1/book/en/

Crédits

- Gentilhomme icônes créées par Freepik - Flaticon
- #esp-rs:matrix.org
- Developing Embedded Rust https://www.youtube.com/watch?v=EughbCeVVxw
- Rust sur de l'IOT ? https://www.youtube.com/watch?v=pl60zczUXt0
- Build scripts https://www.youtube.com/watch?v=pePqWoTnSmQ
- slidev https://sli.dev/
- Theme vuetiful https://github.com/LinusBorg/slidev-theme-vuetiful

Ressources / Références

- Embedded Rust
  - Rust Embedded Workgroup
  - Embedded Rust book
  - Organisation esp-rs (et tous les projets liés sur Github)
  - Awesome embedded Rust
- Espressif Rust
  - Organisation esp-rs (et tous les projets liés sur Github)
  - esp-rs book
  - Ferrous systems Espressif Training
  - Awesome ESP Rust
  - Blog de Scott Mabin
  - Explication du Typestate pattern chez Cliff L. Biffle
  - Explication d'un principe identique au typestate pattern pour un builder chez Akanoa
  - Documentation officielle de esp-idf
  - FreeRTOS
  - Site officiel de la montre

Projets

- Slides : https://github.com/pyaillet/twatch-rust
- Firmware de la montre : https://github.com/pyaillet/twatch-idf-rs
- Github : https://github.com/pyaillet/