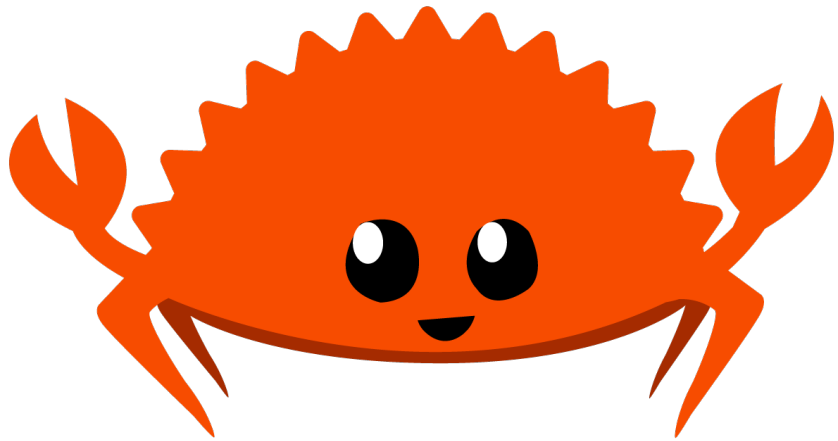
RustIEC - Rust ecosysteem

RustIEC team

November 24, 2022

OVERVIEW

1. Rust in general
2. Rust for web
3. Rust for embedded



Rust ecosystem: general remarks

CARGO

Cargo is your friend!

CARGO

Cargo is your friend!

Cargo.toml:

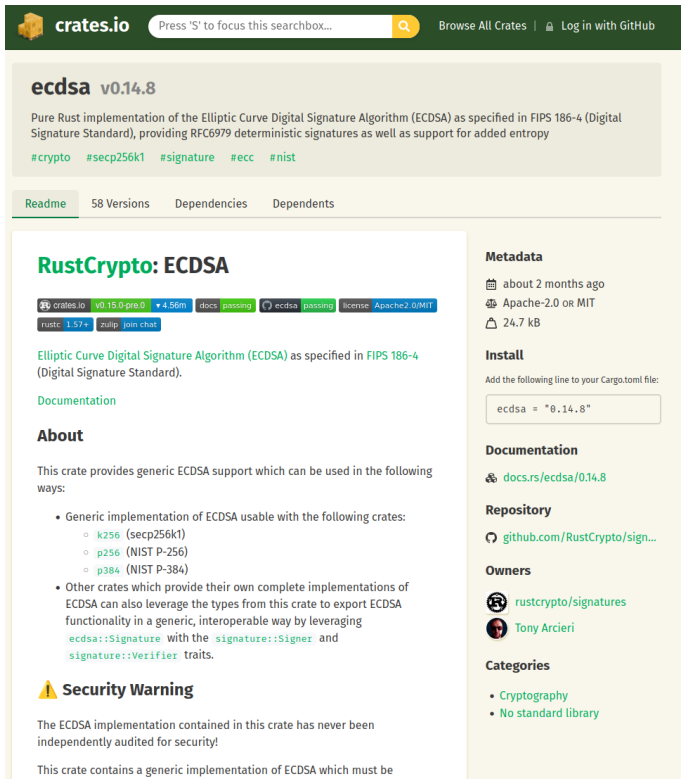
```
1  [package]
2  name = "whisperfish"
3  version = "0.6.0-dev"
4  authors = ["Ruben De Smet
   ↪  <ruben.de.smet@rubdos.be>"]
5  description = "Private messaging
   ↪  ..."
6
7  [dependencies]
8  async-trait = "0.1"
9  bincode = "1.2.1"
10 actix = "0.12"
```

CARGO

USEFUL WEBSITES

Cargo is your new best friend:

- `https://crates.io`
  (e.g., `crates.io/crates/ecdsa`)

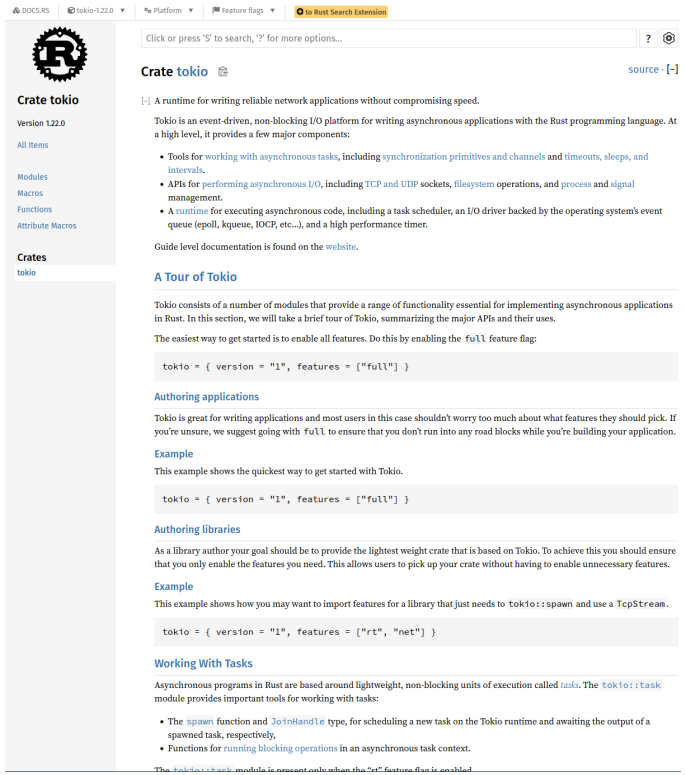


CARGO

USEFUL WEBSITES

Cargo is your new best friend:

- `https://crates.io`
  (e.g., `crates.io/crates/ecdsa`)
- `https://docs.rs`
  (e.g., `docs.rs/tokio`)

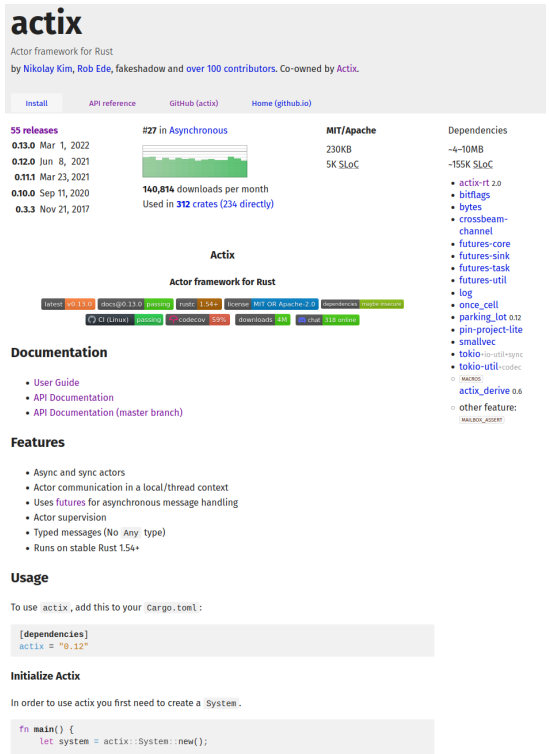


CARGO

USEFUL WEBSITES

Cargo is your new best friend:

- `https://crates.io`
  (e.g., `crates.io/crates/ecdsa`)
- `https://docs.rs`
  (e.g., `docs.rs/tokio`)
- `https://lib.rs`[a]
  (e.g., `lib.rs/crates/actix`)



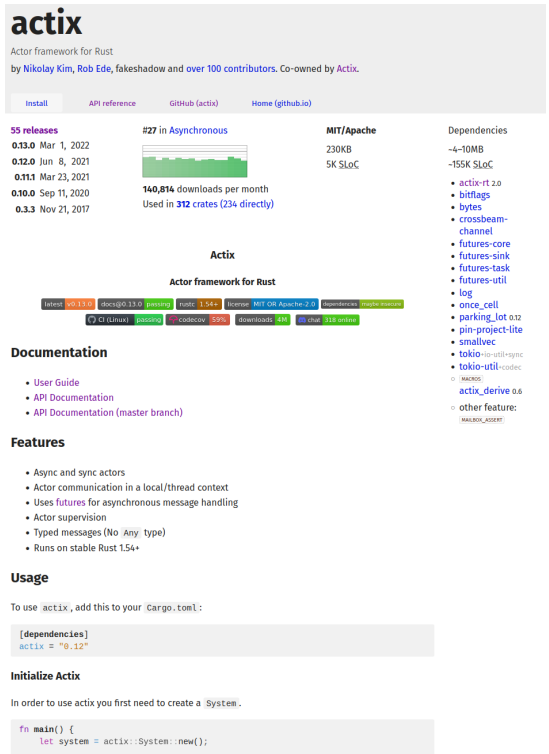
[a] unofficial

CARGO

USEFUL WEBSITES

Cargo is your new best friend:

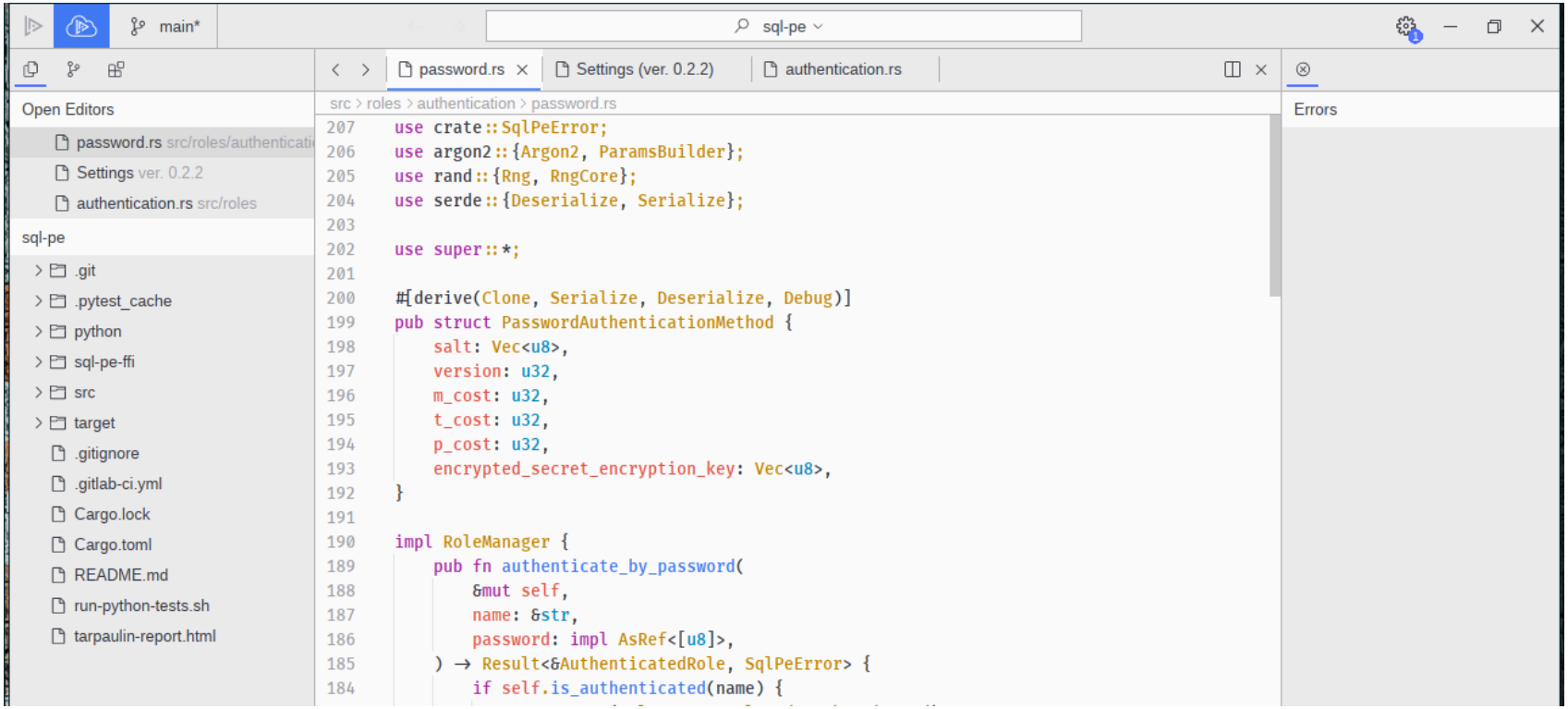
- `https://crates.io`
  (e.g., `crates.io/crates/ecdsa`)
- `https://docs.rs`
  (e.g., `docs.rs/tokio`)
- `https://lib.rs`[a]
  (e.g., `lib.rs/crates/actix`)



[a] unofficial

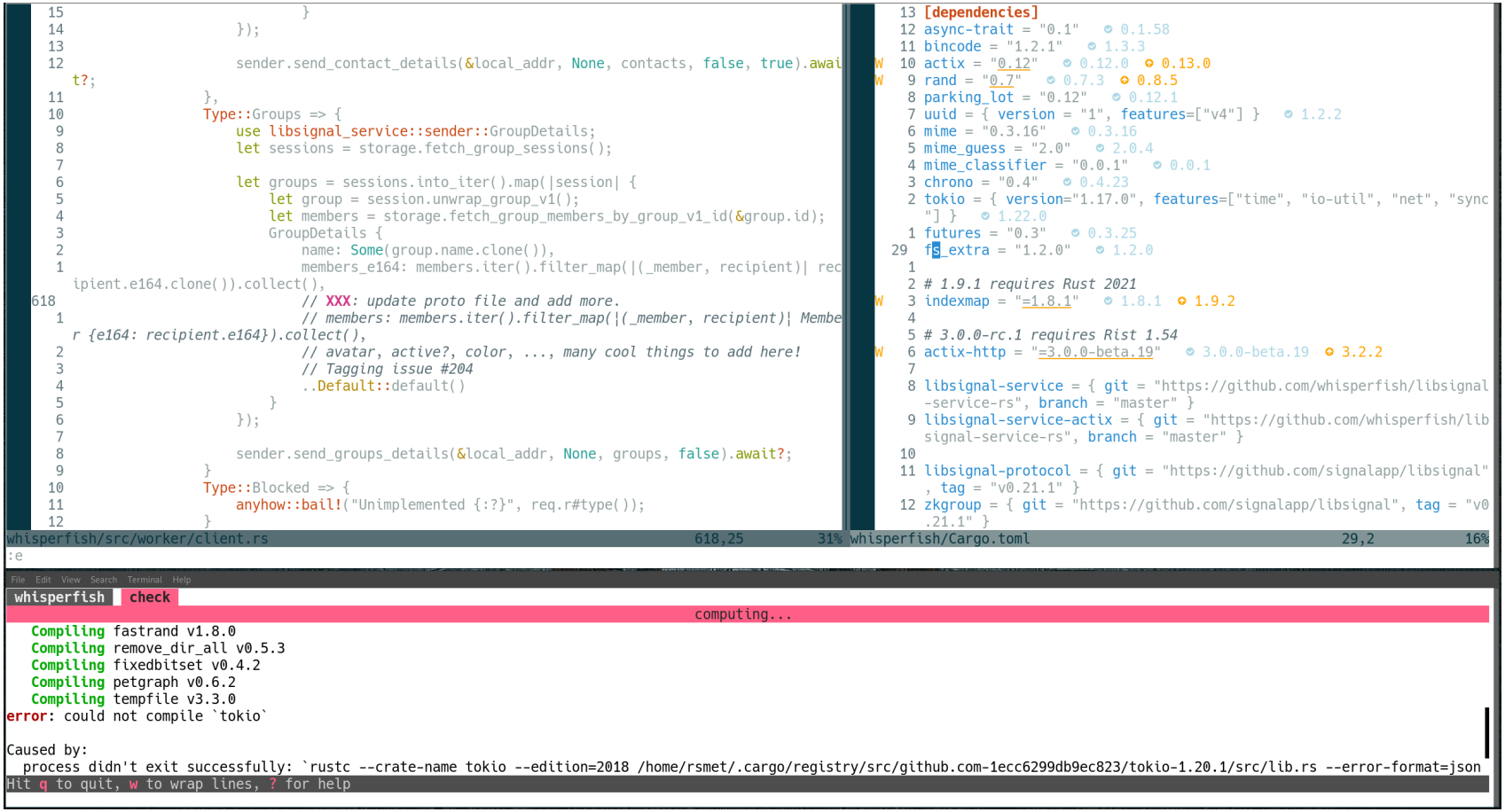
Rust search extension `https://rust.extension.sh/`

INTEGRATED DEVELOPMENT ENVIRONMENTS



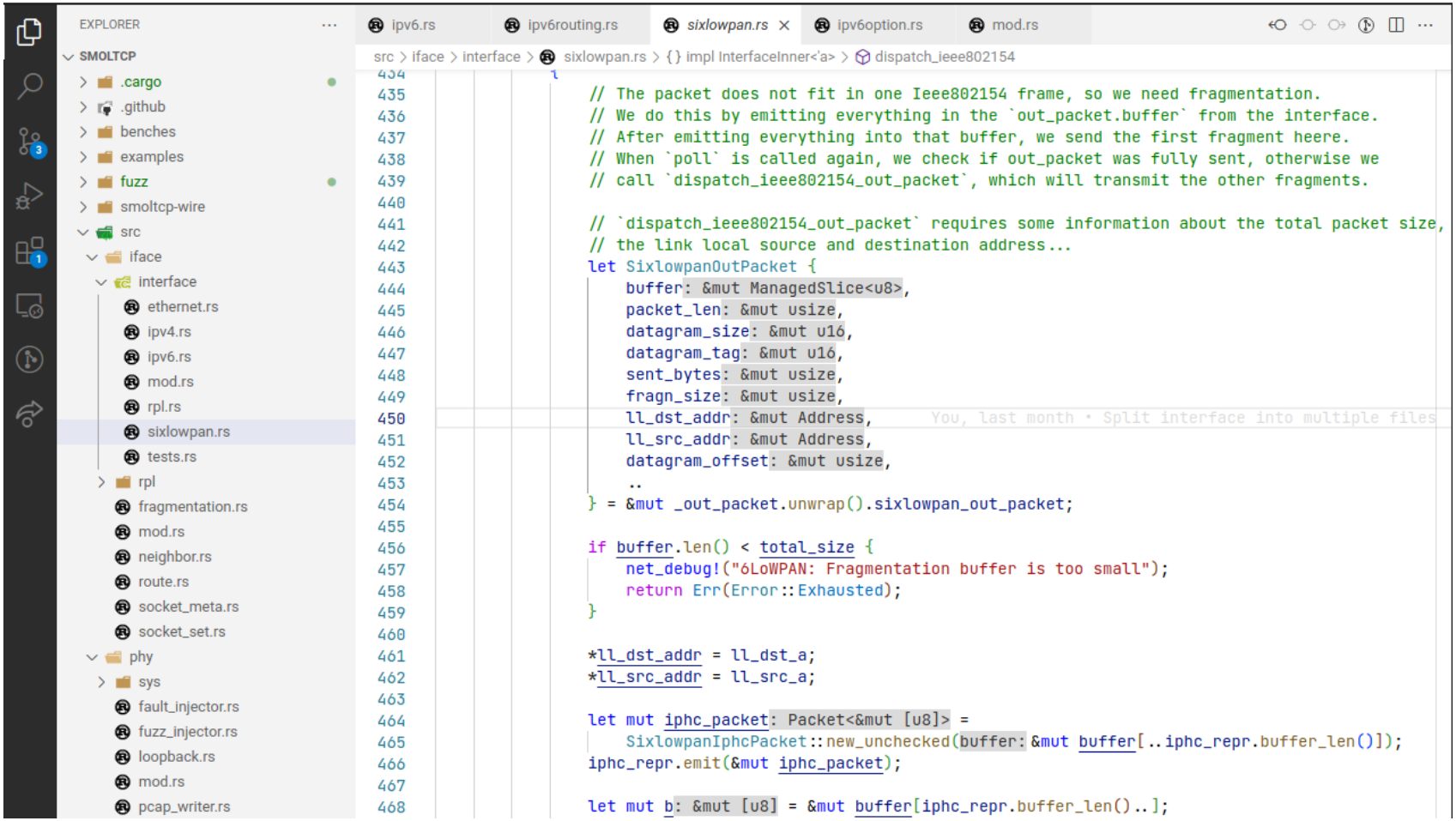
Lapce (`https://lapce.dev/`)

INTEGRATED DEVELOPMENT ENVIRONMENTS



VIM (`bacon` and `coc.nvim`)

INTEGRATED DEVELOPMENT ENVIRONMENTS



VS Code (`https://code.visualstudio.com/`)

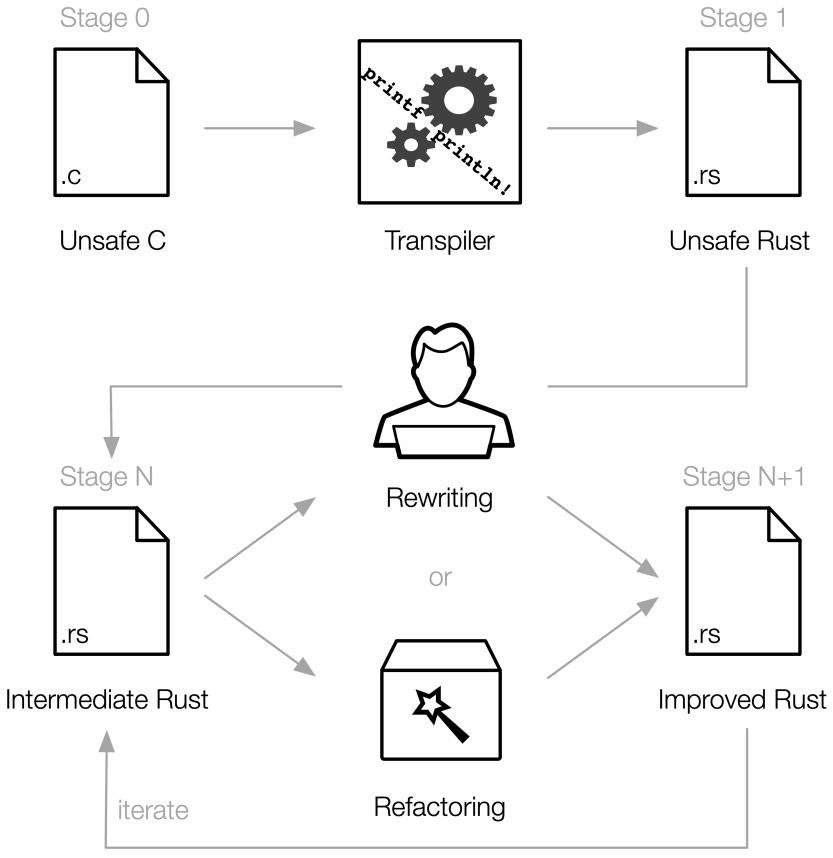
rust.
analyzer

General rule: `rust-analyzer`
(`https://github.com/rust-lang/rust-analyzer`)

RUST AND C

"I have C code, and I want it to become Rust code":

- c2rust (`https://github.com/immunant/c2rust`)



RUST AND C

"I have C code, and I want it to become Rust code":

- c2rust (`https://github.com/immunant/c2rust`)

"I have Rust code, and I need to call it from C or C++":

- cbindgen (`https://github.com/eqrion/cbindgen/`)

```
1  #[no_mangle]
2  pub unsafe extern "C" fn sqlpe_spsm_validate_trust(
3      spsm_ptr: *const SqlPrivilegeStateMachine,
4  ) -> c_int {
5      let spsm = spsm_ptr.as_ref().expect("valid spsm");
6      ...
7  }
```

PARALLELISM AND CONCURRENCY

RAYON: PARALLEL ITERATORS

```
1  use rayon::prelude::*;
2  fn sum_of_squares(input: &[i32]) -> i32 {
3      input.par_iter() // <-- just change that!
4          .map(|&i| i * i)
5          .sum()
6  }
```

PARALLELISM AND CONCURRENCY

CROSSBEAM: LOW-LEVEL SYNCHRONISATION PRIMITIVES

```rust
use crossbeam_queue::ArrayQueue;

let q = ArrayQueue::new(2);

assert_eq!(q.push('a'), Ok(()));
assert_eq!(q.push('b'), Ok(()));
assert_eq!(q.push('c'), Err('c'));
assert_eq!(q.pop(), Some('a'));
```

PARALLELISM AND CONCURRENCY

ASYNC-AWAIT

```
async fn foo() -> MyObject {
    // Long running execution...
    let bytes = fetch_from_the_internet().await;
    decrypt_on_threadpool(bytes).await
}
```

MISCELLANEOUS CRATES

logging `log` and `env_logger`

Example:

```
log::info!("Starting application v{version}");
log::error!("Error connecting to server at {url}!");
```

MISCELLANEOUS CRATES

logging `log` and `env_logger`

errors `thiserror` and `anyhow`

Example:

```
#[derive(thiserror::Error, Debug)]
pub enum AwcWebSocketError {
    #[error("Could not connect")]
    ConnectionError(#[from]
    ↪ awc::error::WsClientError),
    #[error("Websocket connection failed")]
    ProtocolError(#[from] WsProtocolError),
}
```

MISCELLANEOUS CRATES

logging	`log` and `env_logger`
errors	`thiserror` and `anyhow`
serialization	`serde`-family

Example:

```rust
#[derive(Debug, Deserialize, Serialize)]
#[serde(rename_all = "camelCase")]
pub struct PreKeyEntity {
    pub key_id: u32,
    #[serde(with = "serde_base64")]
    pub public_key: Vec<u8>,
}
```

LISTS FOR ECOSYSTEM QUALITY

AREWE*YET.ORG

- https://AreWeAsyncYet.rs/
- https://AreWeGameYet.rs/
- https://AreWeWebYet.org/
- https://www.AreWeLearningYet.com
- ...

https://wiki.mozilla.org/Areweyet

Rust for the Web and Cloud

back-end ecosystem: web and cloud

ETRO
ELECTRONICS &
INFORMATICS

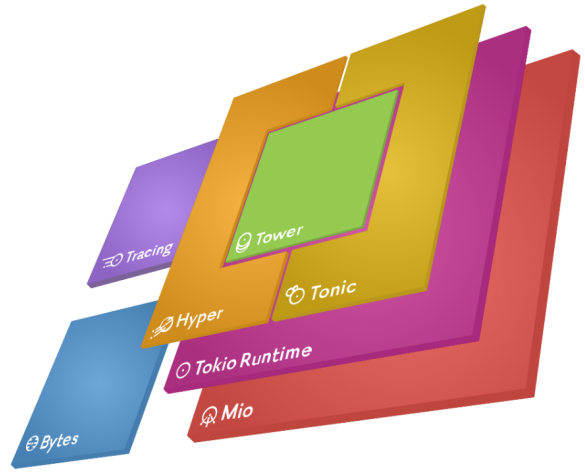
BACK-END ECOSYSTEM

TOKIO

Multi-layer approach:

- Built on Mio: “Metal IO”

  `https://github.com/tokio-rs/mio`



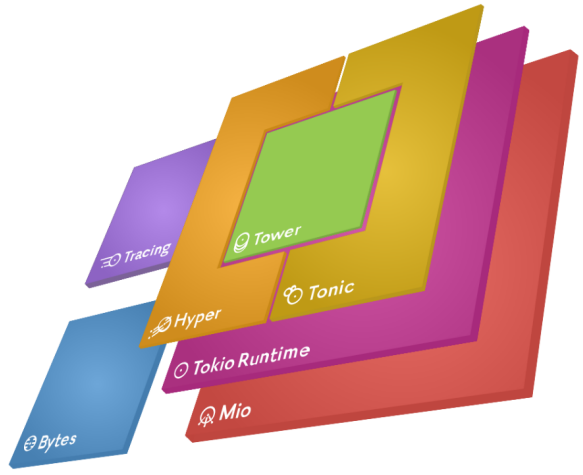
`https://tokio.rs/`

BACK-END ECOSYSTEM

TOKIO

Multi-layer approach:

- Built on Mio: “Metal IO”
  `https://github.com/tokio-rs/mio`
- Tokio asynchronous runtime:
  I/O, timer, filesystem,
  synchronization, and scheduling



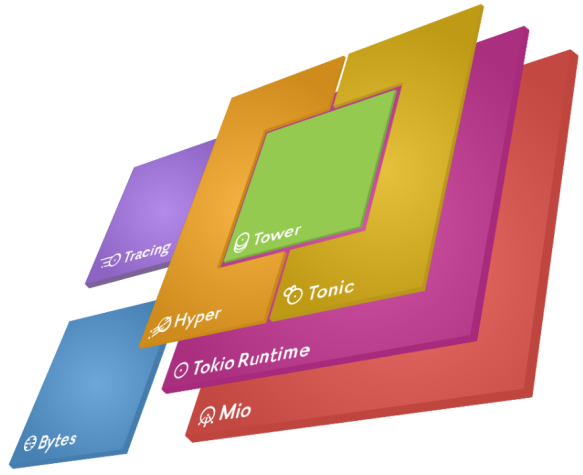
`https://tokio.rs/`

BACK-END ECOSYSTEM

TOKIO

Multi-layer approach:

- Built on Mio: “Metal IO”
  `https://github.com/tokio-rs/mio`
- Tokio asynchronous runtime:
  I/O, timer, filesystem,
  synchronization, and scheduling
- HTTP implementations:
  - Hyper
    `https://github.com/hyperium/hyper`
  - Actix Web
    `https://actix.rs`



`https://tokio.rs/`

Rust for the Web and Cloud
front-end ecosystem

ETRO
ELECTRONICS &
INFORMATICS

FRONT-END ECOSYSTEM

EXAMPLE: FLICK.RS

- wasm-pack

  `https://github.com/rustwasm/wasm-pack/`
- React
- Client-side cryptography in Rust

`https://flickrs.opencloudedge.be`

FRONT-END ECOSYSTEM

EXAMPLE: FLICK.RS

- wasm-pack
  `https://github.com/rustwasm/wasm-pack/`
- React
- Client-side cryptography in Rust

Alternative, 100 % Rust: `https://yew.rs/`

`https://flickrs.opencloudedge.be`

Rust for embedded programming

ETRO
ELECTRONICS &
INFORMATICS

IN C

1. Install IDE and tools for specific microcontroller;
2. Use Hardware Abstraction Layer (HAL) provided by manufacturer;
3. Write firmware . . .

IN RUST

1. Use any preferred IDE and tools[1] for specific microcontroller;
2. Search for HAL on `crates.io` / `lib.rs` / . . .;
   Manufactures don't support Rust yet;
3. Write firmware . . .

[1] Most of the time, cargo can be used for anything

SUPPORTED MICROCONTROLLERS

- Peripheral Access Crates
  - Microchip
  - Nordic
  - NXP
  - Raspberry Pi Silicon
  - SiFive
  - Silicon Labs
  - STMicroelectronics
  - Texas Instruments
  - MSP430
  - Espressif
  - Ambiq Micro
  - GigaDevice
  - XMC
  - Vorago
  - Wiznet

- HAL implementation crates
  - OS
  - Microchip
  - Nordic
  - NXP
  - Raspberry Pi Silicon
  - SiFive
  - STMicroelectronics
  - Texas Instruments
  - MSP430
  - Espressif
  - Silicon Labs
  - XMC
  - GigaDevice
  - Vorago

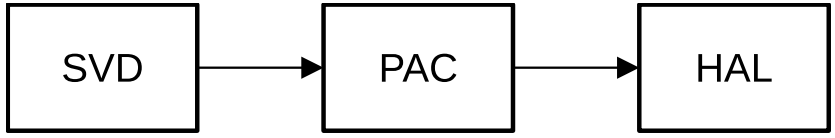
`https://github.com/rust-embedded/awesome-embedded-rust`

PAC OR HAL NOT AVAILABLE?

SVD2RUST

PAC OR HAL NOT AVAILABLE?

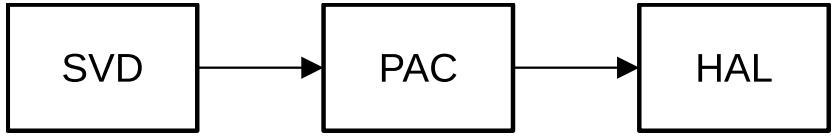
SVD2RUST



SVD: System View Description file

PAC OR HAL NOT AVAILABLE?

SVD2RUST



SVD: System View Description file

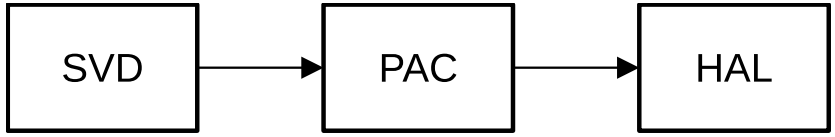
↓ **svd2rust**

PAC: Peripheral Access Crate

```
$ svd2rust -i STM32F30x.svd
$ form -i lib.rs -o src/ && rm lib.rs
$ cargo fmt
```

PAC OR HAL NOT AVAILABLE?

SVD2RUST



SVD: System View Description file

↓ **svd2rust**

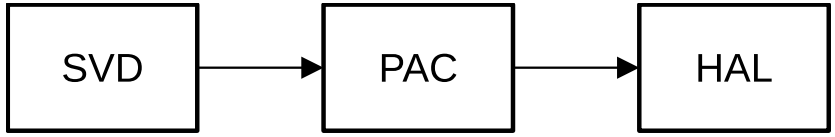
PAC: Peripheral Access Crate

↓ developer

HAL: Hardware Abstraction Layer

PAC OR HAL NOT AVAILABLE?

SVD2RUST



SVD: System View Description file

↓ **svd2rust**

PAC: Peripheral Access Crate

↓ developer

HAL: Hardware Abstraction Layer

EMBEDDED-HAL

Trait[2] *definitions for embedded systems.*

[2] Interfaces in Java or Abstract class in C#

EMBEDDED-HAL

Trait[2] *definitions for embedded systems.*

Example for SPI:

```
1  pub trait Spi<Word> {
2      type Error;
3      fn read(&mut self) -> Result<Word, Self::Error>;
4      fn send(&mut self, word: Word) -> Result<(),
   ↪   Self::Error>;
5  }
```

[2] Interfaces in Java or Abstract class in C#

EMBEDDED-HAL

Trait[2] *definitions for embedded systems.*

Example for SPI:

```
1  pub trait Spi<Word> {
2      type Error;
3      fn read(&mut self) -> Result<Word, Self::Error>;
4      fn send(&mut self, word: Word) -> Result<(),
   ↪   Self::Error>;
5  }
```

Consequences:

- driver authors can support any number of target platforms;
- HAL developers unlock all these drivers for their platform.

[2] Interfaces in Java or Abstract class in C#

PROBE-RS

CARGO REALLY IS YOUR BEST FRIEND!

Flash programming

Downloading a binary to your MCU is easy with probe-rs. We support ELF, ihex and plain binary.

Compatible

ARM, RISC-V, CMSIS-DAP, STLink, JLink probe-rs supports it all. And many more to come!

Cargo integration

With cargo-flash you get everything cargo run gives you, but for embedded targets. No compromises.

Extensible

probe-rs can be used as a library, giving you even more flexibility

GDB integration

probe-rs includes a GDB stub to integrate seamlessly into your usual workflow with common tools..

VSCode integration

We provide a Microsoft DAP server implementation with probe-rs to debug in VSCode.

PROBE-RS

CARGO REALLY IS YOUR BEST FRIEND!

Flash programming

Downloading a binary to your MCU is easy with probe-rs. We support ELF, ihex and plain binary.

Compatible

ARM, RISC-V, CMSIS-DAP, STLink, JLink probe-rs supports it all. And many more to come!

Cargo integration

With cargo-flash you get everything cargo run gives you, but for embedded targets. No compromises.

Extensible

probe-rs can be used as a library, giving you even more flexibility

GDB integration

probe-rs includes a GDB stub to integrate seamlessly into your usual workflow with common tools..

VSCode integration

We provide a Microsoft DAP server implementation with probe-rs to debug in VSCode.

```
cargo flash --release --chip <chip_name>
```

EMBEDDED OPERATING SYSTEMS

Real-Time Interrupt-driven Concurrency (RTIC):

- Currently only Cortex-M

Embassy (Embedded + Async):

- Cortex-M, RISC-V, Xtensa architectures

TOCK:

- Cortex-M, RISC-V, Xtensa architectures

EMBASSY EXAMPLE

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

#[embassy_executor::main]
async fn main(_spawner: Spawner) {
    let p = embassy_nrf::init(Default::default());
    let mut led = Output::new(p.P0_13, Level::Low,
↪   OutputDrive::Standard);

    loop {
        led.set_high();
        Timer::after(Duration::from_millis(300)).await;
        led.set_low();
        Timer::after(Duration::from_millis(300)).await;
    }
}
```

DRIVERS

`https://github.com/rust-embedded/awesome-embedded-rust`

1. AD983x - SPI - AD9833/AD9837 waveform generators / DDS - Intro blog post - crates.io v0.3.0
2. adafruit-alphanum4 - I2C - Driver for Adafruit 14-segment LED Alphanumeric Backpack based on the ht16k33 chip - crates.io v0.1.2
3. ADS1x1x - I2C - 12/16-bit ADCs like ADS1013, ADS1015, ADS1115, etc. - Intro blog post - crates.io v0.2.2
4. ADXL313 - SPI - 3-axis accelerometer - crates.io v0.2.4
5. ADXL343 - I2C - 3-axis accelerometer - crates.io v0.8.0
6. ADXL355 - SPI - 3-axis accelerometer - Intro blog post - crates.io v0.2.3
7. AHT20 - I2C - Humidity and temperature sensor - github - crates.io v0.1.0
8. AHT20-driver - I2C - Humidity and temperature sensor - Intro blog post - github - crates.io v1.2.0
9. AnyLeaf - I2C - pH sensor module - github - crates.io v1.0.1
10. AT86RF212 - SPI - Low power IEEE 802.15.4-2011 ISM RF Transceiver - Intro blog post - crates.io v0.2.0
11. BlueNRG - SPI - driver for BlueNRG-MS Bluetooth module - Intro post crates.io v0.1.0
12. BNO055 - I2C - Bosch Sensortec BNO055 9-axis IMU driver - Intro post crates.io v0.3.3
13. CD74HC4067 - GPIO - 16-channel digital and analog multiplexer - Intro blog post - github - crates.io v0.2.0
14. dht-sensor - 1-Wire - DHT11/DHT22 temperature/humidity sensor driver - github - crates.io v0.2.1
15. DRV8825 - DRV8825 Stepper Motor Driver (based on Stepper) - Intro blog post - crates.io v0.6.0
16. DS1307 - I2C - Real-time clock driver - Intro blog post - crates.io v0.4.0
17. EEPROM24x - I2C - 24x series serial EEPROM driver - Intro blog post - crates.io v0.5.0
18. embedded-ccs811 - I2C - Gas and VOC sensor driver for monitoring indoor air quality - Intro blog post - crates.io v0.2.0

LIBRARIES FOR EMBEDDED

`https://github.com/rust-embedded/awesome-embedded-rust`

1. adskalman: Kalman filter and Rauch-Tung-Striebel smoothing implementation. crates.io v0.14.0
2. atomic: Generic Atomic wrapper type. crates.io v0.5.1
3. bbqueue: A SPSC, statically allocatable queue based on BipBuffers suitable for DMA transfers - crates.io v0.5.1
4. bitmatch: A crate that allows you to match, bind, and pack the individual bits of integers. - crates.io v0.1.1
5. biquad: A library for creating second order IIR filters for signal processing based on Biquads, where both a Direct Form 1 (DF1) and Direct Form 2 Transposed (DF2T) implementation is available. crates.io v0.4.2
6. bit_field: manipulating bitfields and bitarrays - crates.io v0.10.1
7. bluetooth-hci: device-independent Bluetooth Host-Controller Interface implementation. crates.io v0.1.0
8. bounded-registers A high-assurance memory-mapped register code generation and interaction library. `bounded-registers` provides a Tock-like API for MMIO registers with the addition of type-based bounds checking. - crates.io v0.1.1
9. cam-geom: Geometric models of cameras for photogrammetry. crates.io v0.13.0
10. combine: parser combinator library - crates.io v4.6.6
11. console-traits: Describes a basic text console. Used by menu and implemented by vga-framebuffer. crates.io not found
12. `cmim`, or Cortex-M Interrupt Move: A crate for Cortex-M devices to move data to interrupt context, without needing a critical section to access the data within an interrupt, and to remove the need for the "mutex dance" - crates.io v0.2.1
13. `cmsis-dsp-sys` : Rust FFI bindings to the Arm CMSIS_5 math library - crates.io v0.3.1
14. dcmimu: An algorithm for fusing low-cost triaxial MEMS gyroscope and accelerometer measurements crates.io v0.2.2

REFERENCES

- **crates.io**: `https://crates.io`
- **docs.rs**: `https://docs.rs`
- **lib.rs**: `https://lib.rs`
- **Rust Search Extension**: `https://rust.extension.sh`
- **cargo**: `https://doc.rust-lang.org/stable/cargo/`
- **Lapce**: `https://lapce.dev`
- **rust-analyzer**: `https://rust-analyzer.github.io/`
- **c2rust**: `https://github.com/immunant/c2rust`
- **cbindgen**: `https://github.com/eqrion/cbindgen`
- **rayon**: `https://github.com/rayon-rs/rayon`
- **AreWeYet**: `https://wiki.mozilla.org/Areweyet`

REFERENCES

- **crossbeam**: https://github.com/crossbeam-rs/crossbeam
- **envlogger**: https://github.com/rust-cli/env_logger
- **thiserror**: https://github.com/dtolnay/thiserror
- **anyhow**: https://github.com/dtolnay/anyhow
- **serde**: https://serde.rs/
- **Awesome Embedded Rust**: https://github.com/rust-embedded/awesome-embedded-rust
- **svd2rust**: https://github.com/rust-embedded/svd2rust
- **embedded-hal**: https://github.com/rust-embedded/embedded-hal
- **probe-rs**: https://probe.rs/
- **Tock**: https://www.tockos.org/
- **RTIC**: https://rtic.rs/1/book/en/
- **Embassy**: https://embassy.dev/