Rust Operating Systems and Frameworks for Embedded Devices

RustIEC

Thibaut Vandervelden | thvdveld@vub.be

June 5, 2024

WHY OPERATING SYSTEMS OR FRAMEWORKS?

- Timing and Scheduling
- Resource Management
- Modularity and Reusability
- Portability
- Tooling, Security, Community

OVERVIEW OF EMBEDDED OS OPTIONS IN RUST

Frameworks

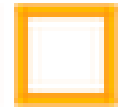
Embassy	Framework with async executor and async HALs `https://embassy.dev/`
RTIC	Real-Time Interrupt-driven Concurrency `https://rtic.rs/`

Operating Systems

Tock OS	Microkernel `https://tockos.org/`
Hubris	Monolithic kernel `https://hubris.oxide.computer/`

EMBASSY

FRAMEWORK

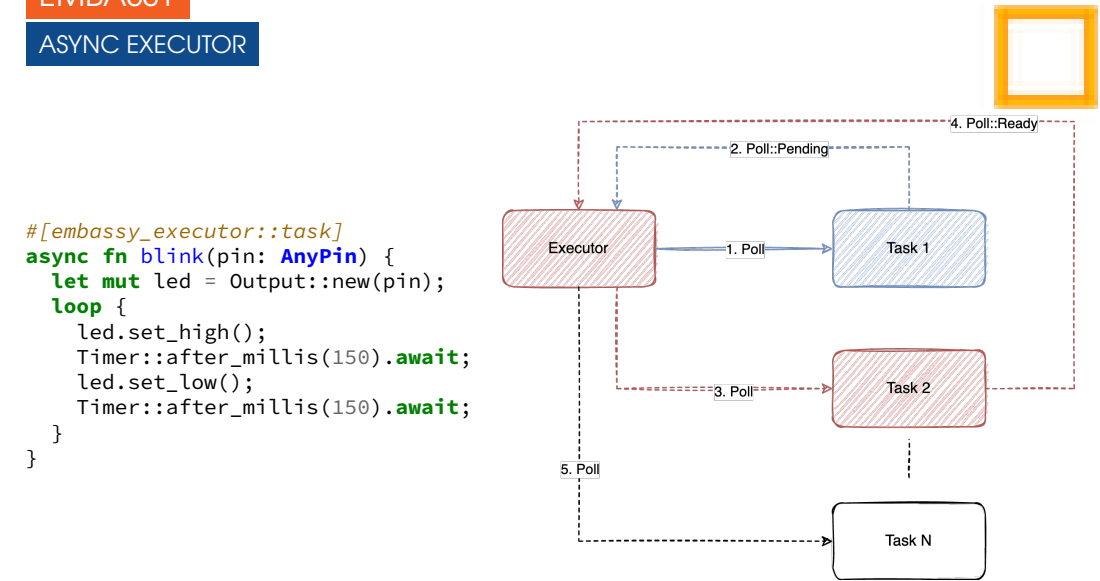


- **Emb**edded meets **as*(s)*y**nc Rust
- Async Executor for embedded devices

Supported platforms	ARM Cortex-M, RISC-V32, AVR, WASM, std
License	MIT or Apache-2.0
Stars on Github	4.6k
Contributors	332

EMBASSY

ASYNC EXECUTOR



```rust
#[embassy_executor::task]
async fn blink(pin: AnyPin) {
  let mut led = Output::new(pin);
  loop {
    led.set_high();
    Timer::after_millis(150).await;
    led.set_low();
    Timer::after_millis(150).await;
  }
}
```

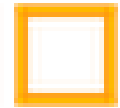
EMBASSY FRAMEWORK

- Async capabable Hardware Abstraction Layers (HALs) (STM32, nRF, ESP32)
- Network stack built using smoltcp
  - IPv4, IPv6
  - TCP, UDP, ICMP, DHCP, ARP
  - IEEE 802.3 (Ethernet), IEEE 802.15.4 + 6LoWPAN

Supported platforms	ARM Cortex-M, RISC-V32, AVR, WASM, std
License	MIT or Apache-2.0
Stars on Github	4.6k
Contributors	332

EMBASSY

SUMMARY



- Async Executor for embedded devices
- HALs with async capabilities (mainly STM32, nRF and ESP32)
- Network stack based on smoltcp
- Runs on many platforms: STM32, nRF, ESP32, etc.
- Lots of examples and documentation

Supported platforms	ARM Cortex-M, RISC-V32, AVR, WASM, std
License	MIT or Apache-2.0
Stars on Github	4.6k
Contributors	332

RTIC FRAMEWORK

- Real-Time Interrupt-driven Concurrency
- Macro-based API for defining tasks and resources
  - Only ARM Cortex-M and RISC-V supported

Supported platforms	ARM Cortex-M, RISC-V32
License	MIT or Apache-2.0
Stars on Github	1.7k
Contributors	82

RTIC FRAMEWORK

```
#[app(device = nrf528xxx_hal::pac)]
mod app {
  #[shared]
  struct Shared {}
  #[local]
  struct Local {
      led: PA5<Output<PushPull>>,
      state: bool,
  }

  #[init]
  fn init(cx: init::Context) -> (Shared, Local) { /*...*/ }
  #[task(local = [led, state])]
  fn blink(cx: blink::Context) { /*...*/ }
  #[task(local = [led, state])]
  fn state_update(cx: state_update::Context) { /*...*/ }
}
```

RTIC

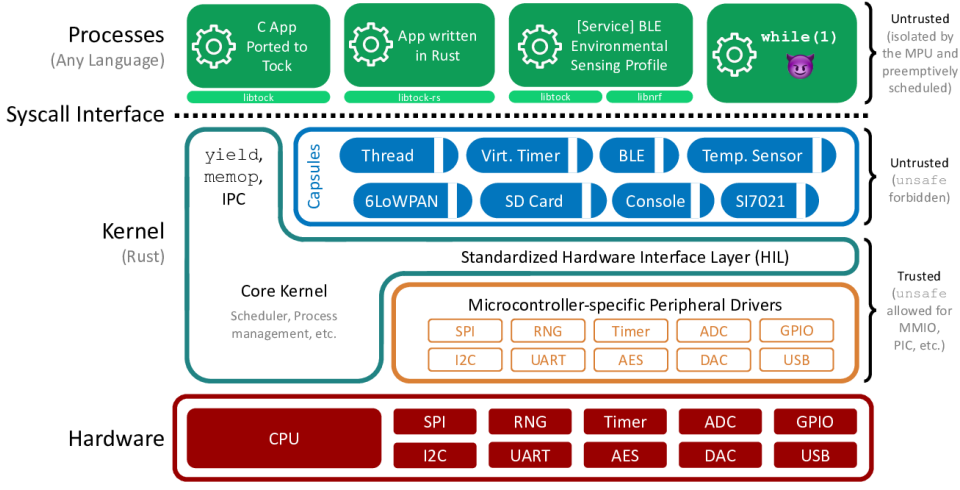
SUMMARY

- Macro-based API for defining tasks and resources
- Prevent deadlocks with Stack Resource Policy
- No HAL or async capabilities (Embassy executor can be used)
- No network stack (smoltcp can be used)
- Only ARM Cortex-M and RISC-V supported

Supported platforms	ARM Cortex-M, RISC-V32
License	MIT or Apache-2.0
Stars on Github	1.7k
Contributors	82

TOCK OS

MICROKERNEL



TOCK OS
SUMMARY

- Memory Protection Unit for application isolation
- Prevent `unsafe` code in applications and capsules (drivers)
- Own network stack (only support IPv6 over IEEE 802.15.4)
- Own executable binary format
- Support for many microcontrollers

Supported platforms	ARM Cortex-M, RISC-V32
License	MIT or Apache-2.0
Stars on Github	5.1k
Contributors	184

HUBRIS

MONOLITHIC KERNEL

- Only support for platforms they use (some STM32 microcontrollers)
- Tasks defined in an **application configuration file**
- Memory Protection Unit for task isolation
- Tasks compiled separately and statically linked
- Synchronous IPC using Rust's borrowing rules
- Network task built using smoltcp

Supported platforms	ARM Cortex-M, RISC-V32
License	Mozilla Public License 2.0
Stars on Github	2.8k
Contributors	49

HUBRIS

SUMMARY

- Focus on robustness
- Not many platforms supported
- Tasks defined in an application configuration file
- Memory Protection Unit for task isolation
- Network stack built using smoltcp

Supported platforms	ARM Cortex-M, RISC-V32
License	Mozilla Public License 2.0
Stars on Github	2.8k
Contributors	49

COMPARISON

Feature	Embassy	RTIC	Tock OS	Hubris
Async capabilities	●	◐	◐	◐
Network Stack	●	○	◐	●
Task Isolation	○	○	●	●
Code size	●	●	◐	○
Platform Support	●	◐	●	○
Documentation & Examples	●	●	●	○
Community Support	●	●	●	○

OTHERS

RIOT-rs A port of RIOT-OS to Rust.
Built using the Embassy framework + threads.
Very experimental.

OTHERS

RIOT-rs A port of RIOT-OS to Rust.
Built using the Embassy framework + threads.
Very experimental.

Others aerugo, bern, drone, lilos, MnemOS, tornado, wasefire, ...

CONCLUSION

RTIC For small projects without network stack

CONCLUSION

RTIC For small projects without network stack

Embassy For more complex projects with a network stack

CONCLUSION

RTIC	For small projects without network stack
Embassy	For more complex projects with a network stack
Tock OS	For advanced projects (multiple applications)

CONCLUSION

RTIC	For small projects without network stack
Embassy	For more complex projects with a network stack
Tock OS	For advanced projects (multiple applications)
Hubris	For very robust projects

Questions?