Status of Rust on Zephyr

Why Rust on Zephyr?

- Safer embedded applications
- Modern tooling (Cargo, crates.io)
- Memory safety without GC
- Fearless concurrency

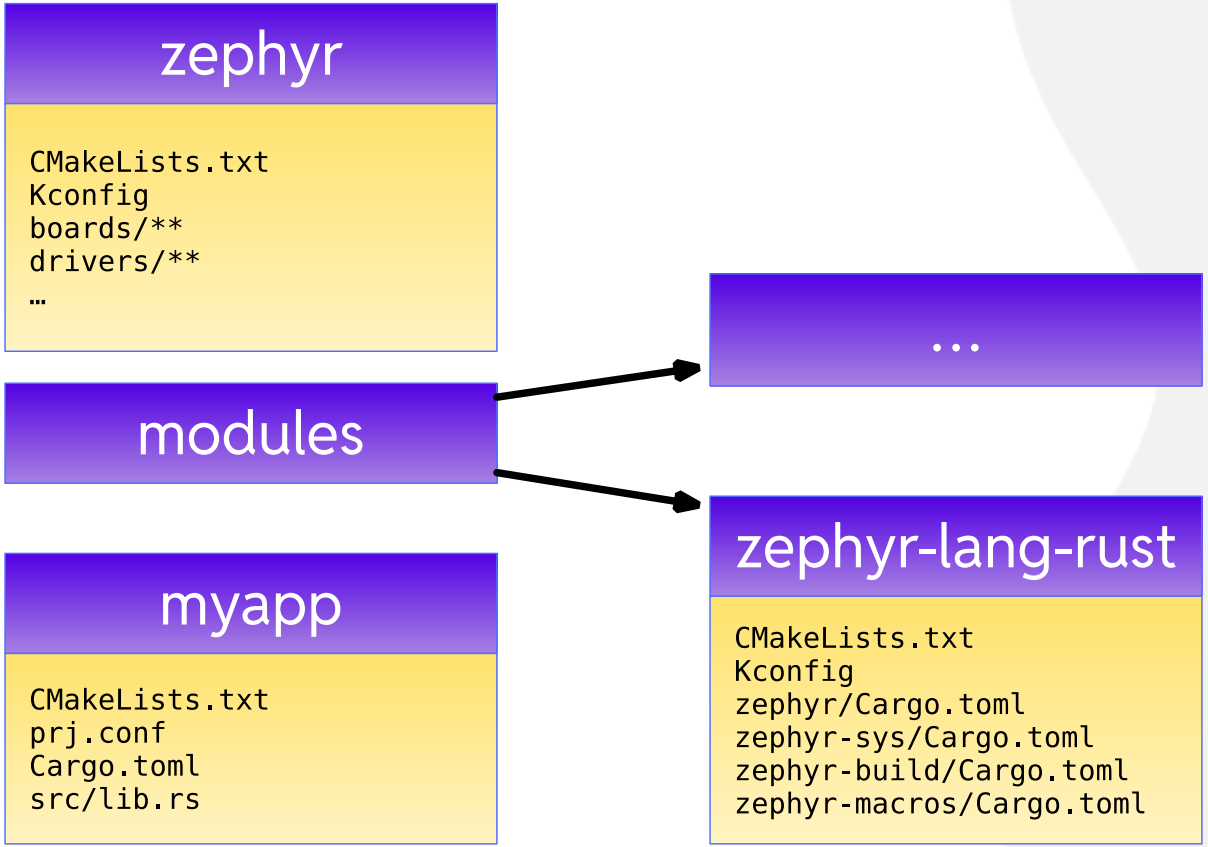
Project Goals

- Write Zephyr apps *entirely* in Rust
- Clean bindings to Zephyr APIs
- Async-friendly device drivers
- Simple developer experience

Not Goals (yet)

- Rewrite Zephyr in Rust
- Replacing Zephyr's build system
- Replacing cargo (Rust's build system)
- Eliminating all C code

Architecture Overview



Current Status

- Rust application builds for Zephyr
- Cargo integration (CMake builds Zephyr, calls cargo to build Rust code)
- Bindings for:
  - Sync things: timers, atomics, channels, mutex/condvar, Arc, semaphores
  - Threads: declarative, decorate regular functions to make threads
  - Work queues
  - Logging
  - Devicetree -> Rust, A few drivers: GPIO, Uart, I2C
  - Kconfig -> values in code, Booleans available for ‘cfg’ macro
- Basic async support:
  - Per-Zephyr-thread async executor (see next presentation)
  - Timer driver for embassy-time

```rust
#![no_std]

use zephyr::raw::GPIO_OUTPUT_ACTIVE;
use zephyr::time::{sleep, Duration};

#[no_mangle]
extern "C" fn rust_main() {
    let mut led0 = unsafe {
        zephyr::devicetree::aliases::led0::get_instance().unwrap()
    };

    led0.configure(&mut gpio_token, GPIO_OUTPUT_ACTIVE);
    let duration = Duration::millis_at_least(500);
    loop {
        led0.toggle_pin(&mut gpio_token);
        sleep(duration);
    }
}
```

What's Working

- Core Zephyr APIs accessible
- Async executors can run in threads
- Small real apps running on real hardware
- Community interest starting

What's Next

- Expand subsystem coverage (networking, sensors, usbd, etc)
- More driver coverage
- Deeper async (ksem_take_async, etc)
- Better DT integration
- Better (for Rust and Zephyr) logging support
- Userspace?

Challenges

- Binding maintenance. Bindgen makes calls directly to C possible, but nice Rust APIs can break with Zephyr API changes
- Mapping C APIs to safe Rust patterns
- Small ecosystem. Reviews are challenging

Get Involved

- Try it!
- Suggest improvements/submit patches
- Help build APIs
- Spread the word
- https://github.com/zephyrproject-rtos/zephyr-lang-rust

Thank You!