RUST MEETS ZEPHYR

BUILDING SAFER EMBEDDED APPLICATIONS

Open Source Summit Europe 2025 | 2025-08-25

Martin Mosler  Zühlke Engineering AG
Principal Consultant and Trainer for Rust, C++, Linux and Security

WHY SHOULD YOU CONSIDER RUST FOR EMBEDDED DEVELOPMENT?

- Memory and concurrency safety concepts
  - Strong type system
  - Bounds checker
  - Ownership, borrow checker and lifetimes
  - Send & Sync marker traits

👉Detect errors at compile time

WHY SHOULD YOU CONSIDER RUST FOR EMBEDDED DEVELOPMENT?

- Modern programming paradigms
  - Algebraic datatypes and pattern matching
  - async / await
    - IO bound code
    - actor based systems with tasks and channels
  - Typestate pattern
  - ...

WHY SHOULD YOU CONSIDER RUST FOR EMBEDDED DEVELOPMENT?

- Tooling
  - Cargo
    - Build
    - Dependency management
    - ...
  - Rust analyzer and Clippy
  - Unit and integration testing
  - ...

HOW DOES RUST INTEGRATE WITH ZEPHYR?

- West -> CMake -> Cargo

```
west build -b <board>
west flash
```

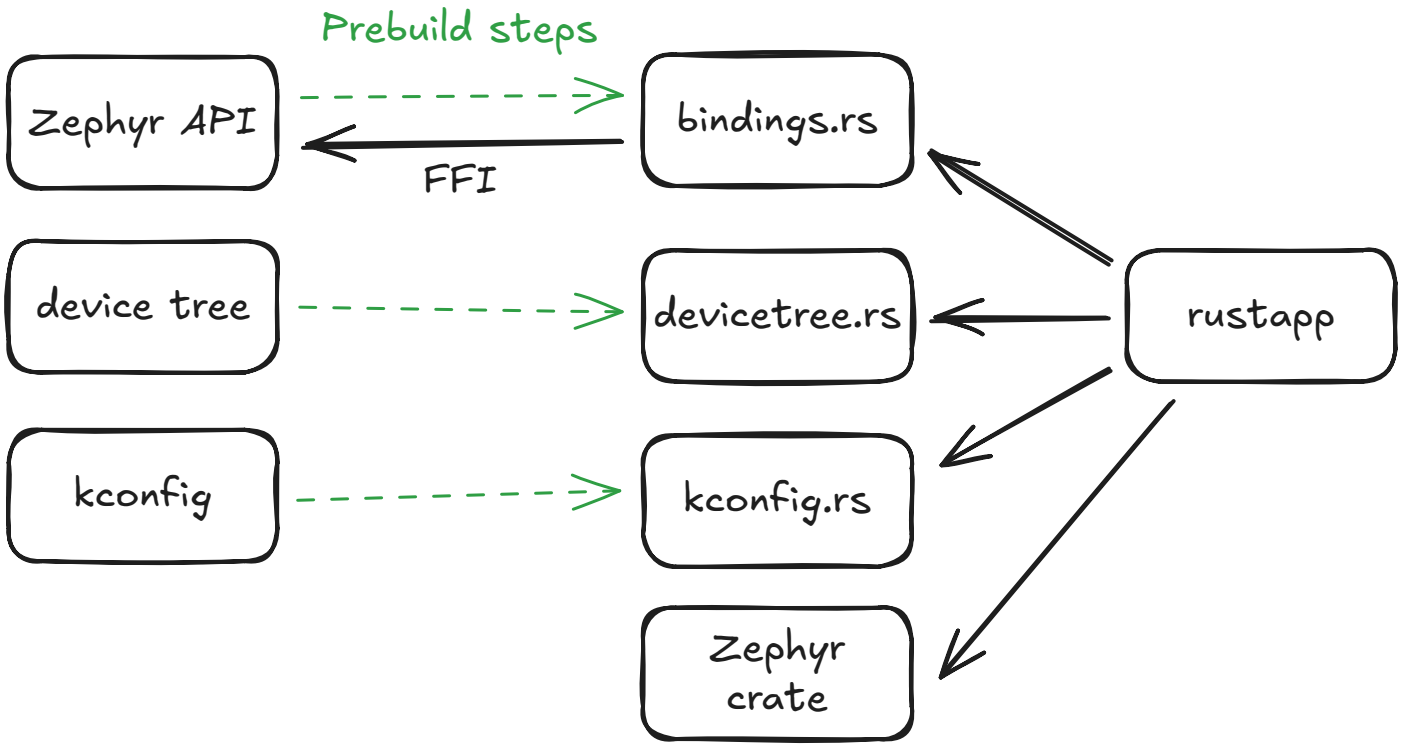
- Rust application links as a static library with Zephyr
- Entry point

```
[no_mangle]
extern "C" fn rust_main() { ... }
```

- no-std

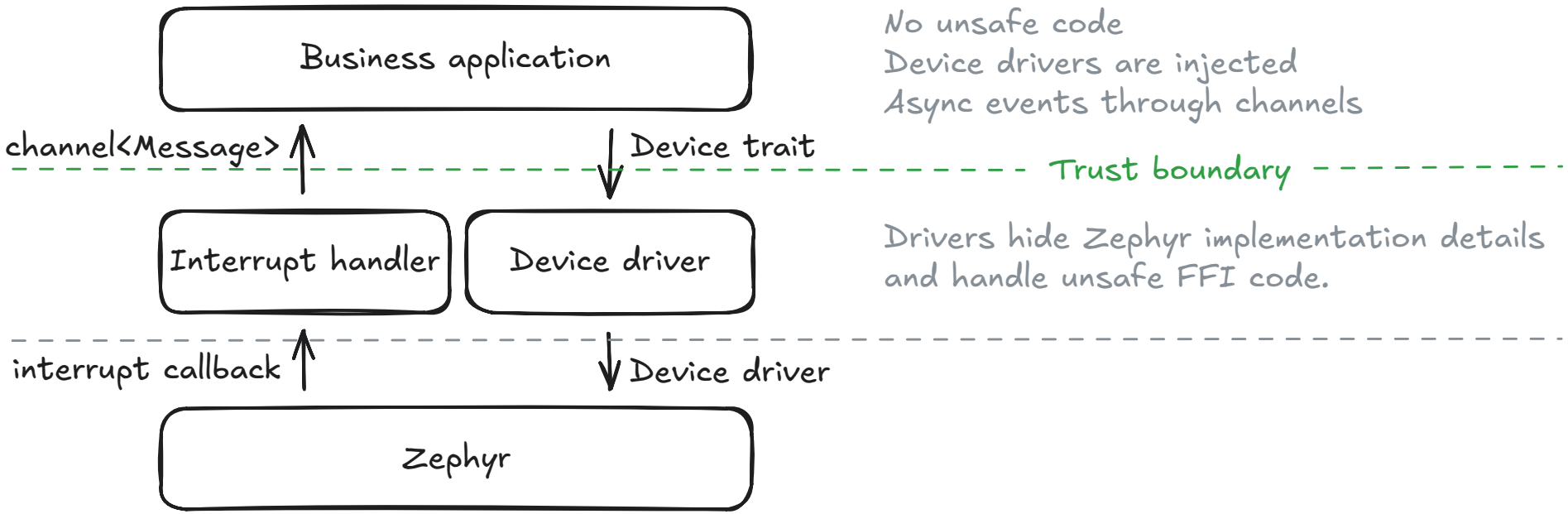
👉Now we can write a (isolated) Rust application

HOW DOES RUST INTEGRATE WITH ZEPHYR'S APIS?



RUST TREATS THE OUTSIDE WORLD AS UNSAFE

- Especially raw pointers without safety guarantees
- Need to separate code using `unsafe` from business logic



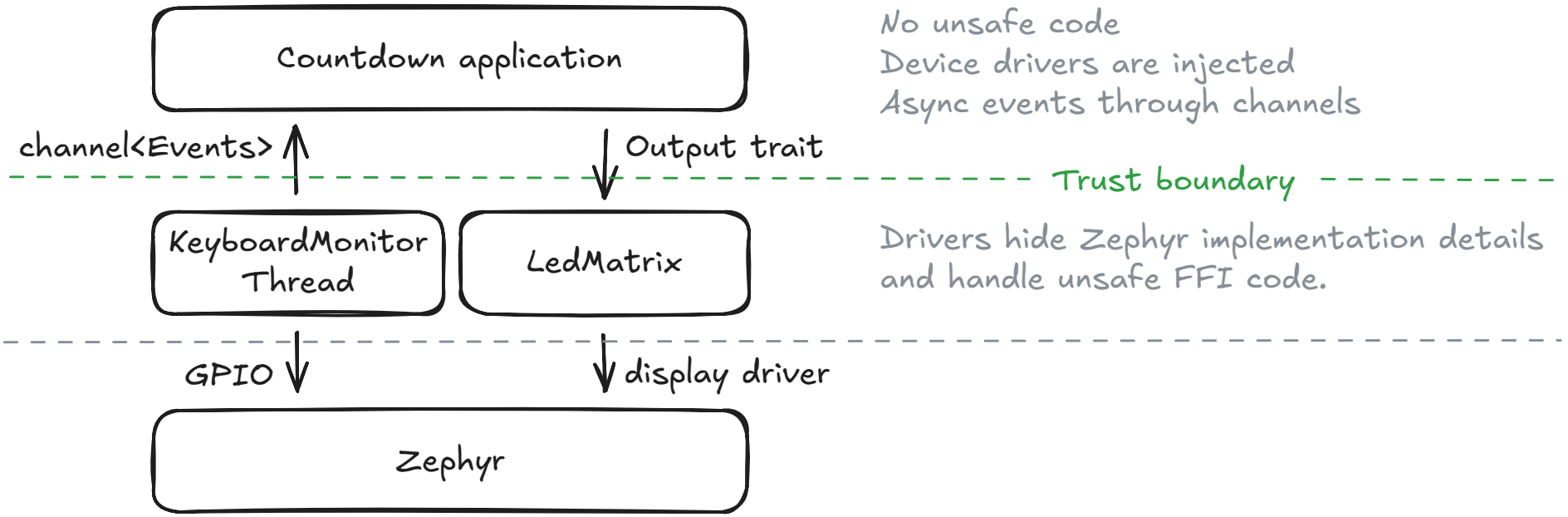
CODE WALKTHROUGH

VISUAL COUNTDOWN ON BBC MICROBIT V2

- Hardware: 2 buttons and 5x5 LED matrix
- Abstraction layer
  - Keyboard thread monitors buttons (GPIOs) and sends events to the application
  - Display driver hides interaction with Zephyr's device driver
- Application: Increase, decrease and run the countdown

CODE WALKTHROUGH

VISUAL COUNTDOWN ON BBC MICROBIT V2



I HAD TO LEARN

- How to configure the code generators
  - to include the display driver after enabling it (bindgen) and
    - `proj.conf`
    - `zephyr-sys/build.rs`
  - generate code for "gpio-keys" in addition to "gpio-leds".
    - `dt-rust.yaml`
- How to get LSP support in your IDE:

```
ln -s build/rust/sample-cargo-config.toml .cargo/config.toml
```

CONCLUSION

- Rust support in Zephyr is ready to be used.
- Today it involves quite some `unsafe` code.
- A clean application architecture is needed to write idiomatic rust.
- Zephyr crate needs more drivers and abstractions.
- Challenges:
  - Bindgen configuration can become difficult.
  - Tradeoff where to add external dependencies.
  - Avoiding duplication of Zephyr functionality.

QUESTIONS?

Example @ https://github.com/zerom0/RustMeetsZephyr

martin.mosler@zuehlke.com