RustChinaConf 2025&
Rust Global China

Rust + BLE:
From technical verification to mass production

Haobo Gu

About me

- I am Haobo
- The author of RMK

In this talk..

- How did RMK evolve from a tiny project to a complex embedded Rust project that ships products to thousands of customers
- The lessons I learned from this journey

What's RMK ?

- RMK is a keyboard firmware library
- It's software running in your keyboard

How the story begins

Making keyboard is fun.

How the story begins

Making keyboard is fun.

And, I was learning Rust, writing Rust is fun, too.

The first try

- stm32h7
- stm32h7xx-hal
- rtic
- usb-device

The first try

It works as expected, but..

The first try

rtic: **Real-Time Interrupt-driven Concurrency**

- Does a keyboard need the "hard" real-time?
- It's hard to write common libraries using rtic
- ARM-only
- Wireless integration?

Takeaway 1

rtic is great for real-time applications, but may not for libraries

Embassy

- Cooperative multi-tasking is fine for keyboards
- Low-power feature is important
- stm32/nRF52/rp are all supported!
- USB and BLE are both supported!

The new tech-stack for RMK

- rtic → embassy-executor
- stm32xxx-hal/rp-hal → embassy-nrf/stm32/rp
- usb-device → embassy-usb
- + nrf-softdevice

Pros

- Unified API
- Simpler multi-tasking
- Support even more MCUs, with BLE
- Low-power ready

Cons

- It's harder when **sharing data** between tasks
- No generic task
- Unstable delay

Takeaway 2

If your project doesn’t strictly require "real-time", go for embassy!

Takeaway 3

Do not communicate by sharing memory, instead, share memory by communicating with `embassy-sync`

Support even more

- Espressif has official support for Rust
- Target: RISC-V and Xtensa

ESP32 support

- esp-idf-hal
- esp32-nimble
- No USB support(yet)

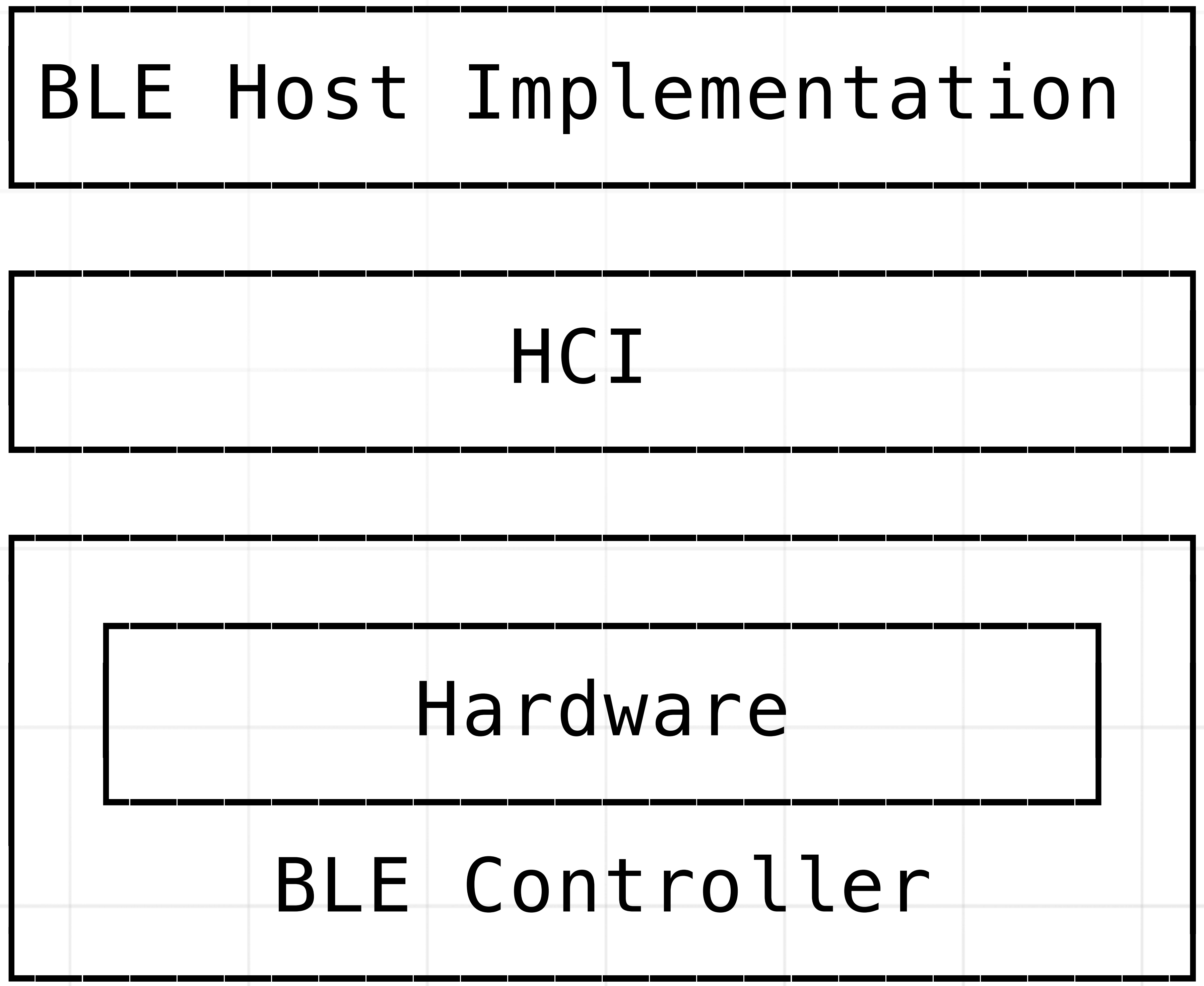
Takeaway 4

Use esp-idf-hal for complex business, and esp-hal for better dev experience

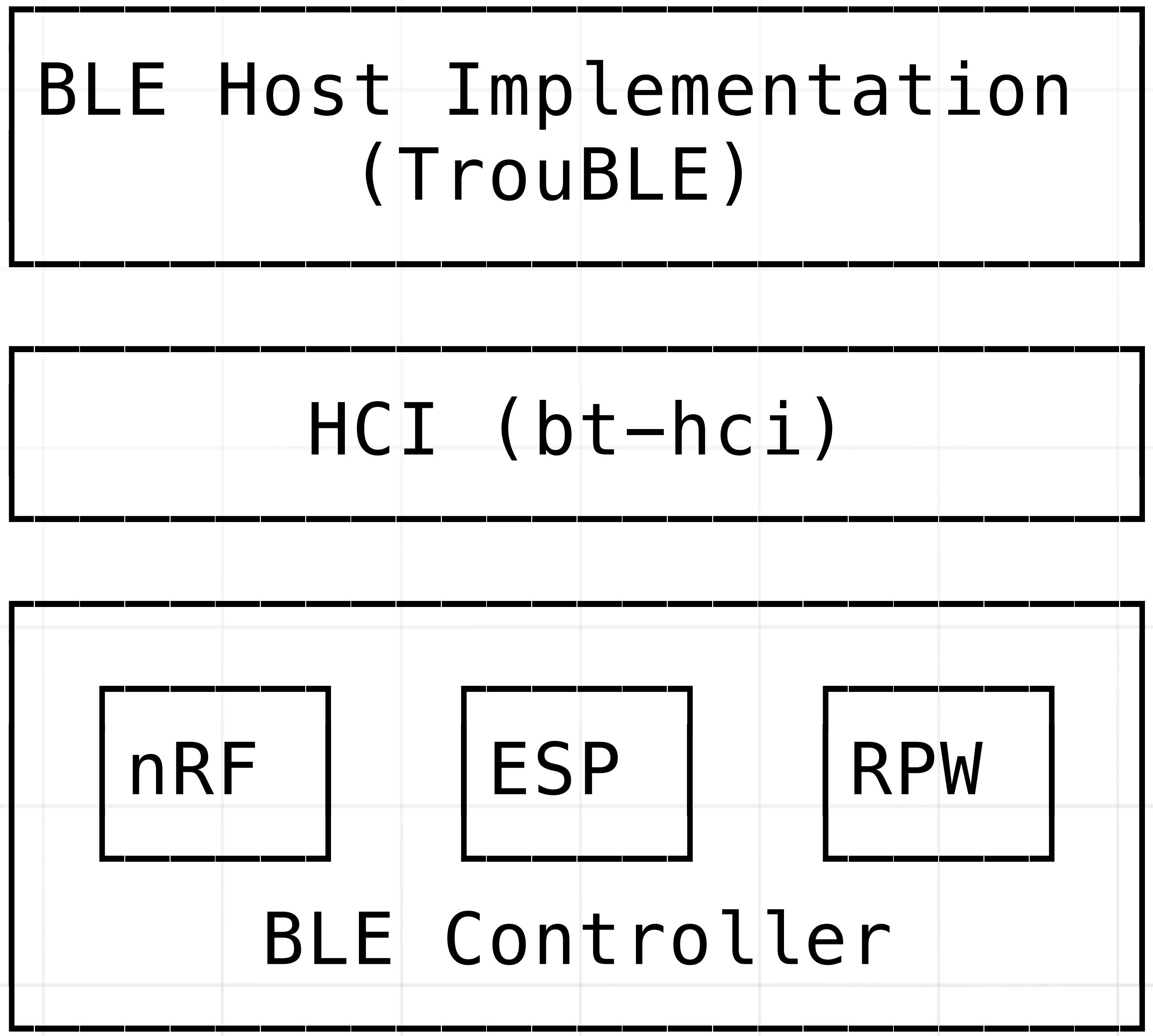
But

Maintaining multiple BLE implementations is **very** hard

BLE HCI



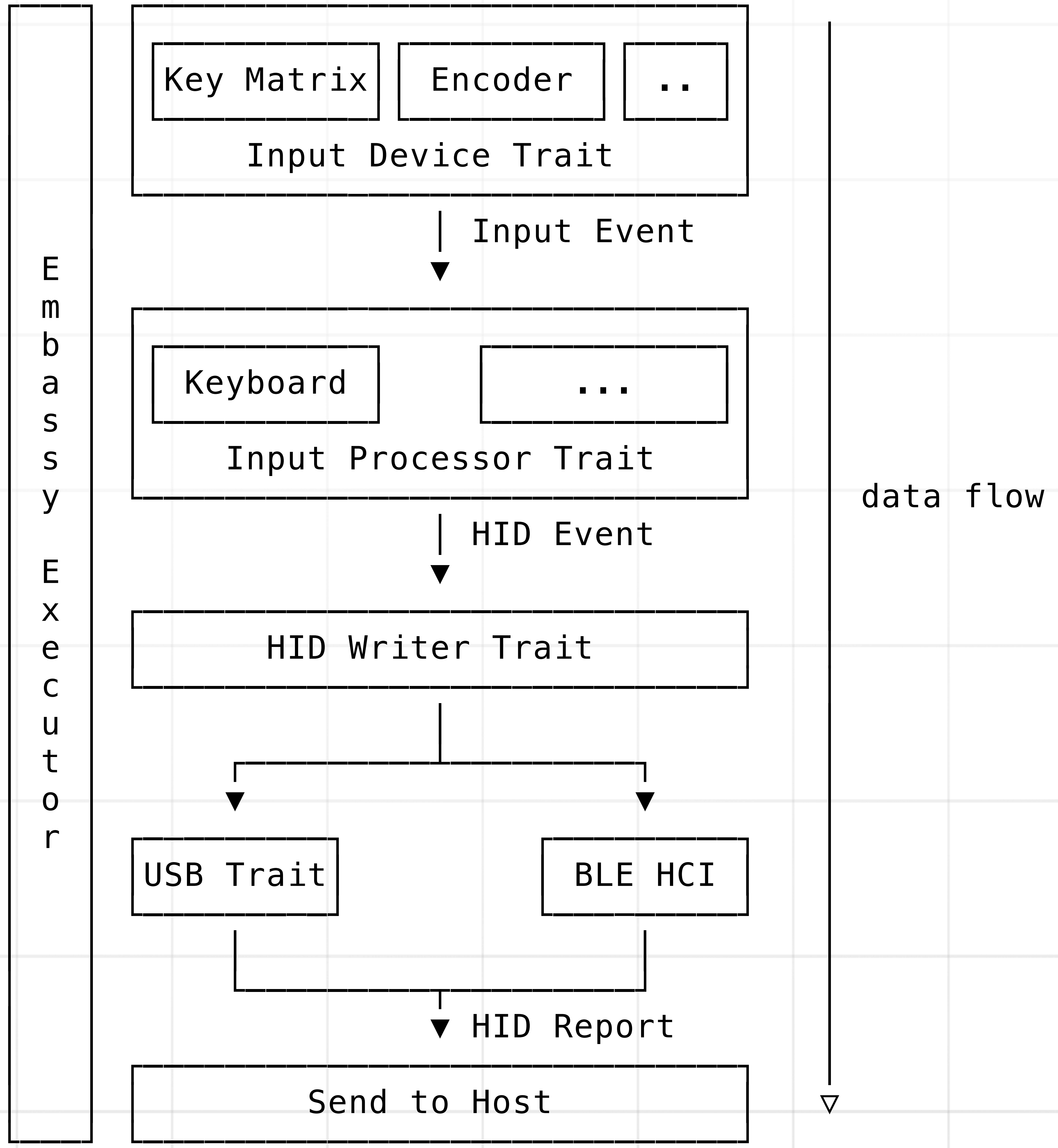
bt-hci and TrouBLE



What I achieved

- A single implementation for nRF52/ESP32/RP-W
- Automatically heterogeneous wireless split keyboard support
- (Bonus) Migrate to esp-hal, which has USB supported for ESP32S3

Current project



To mass production

What’s needed for stepping to mass production?

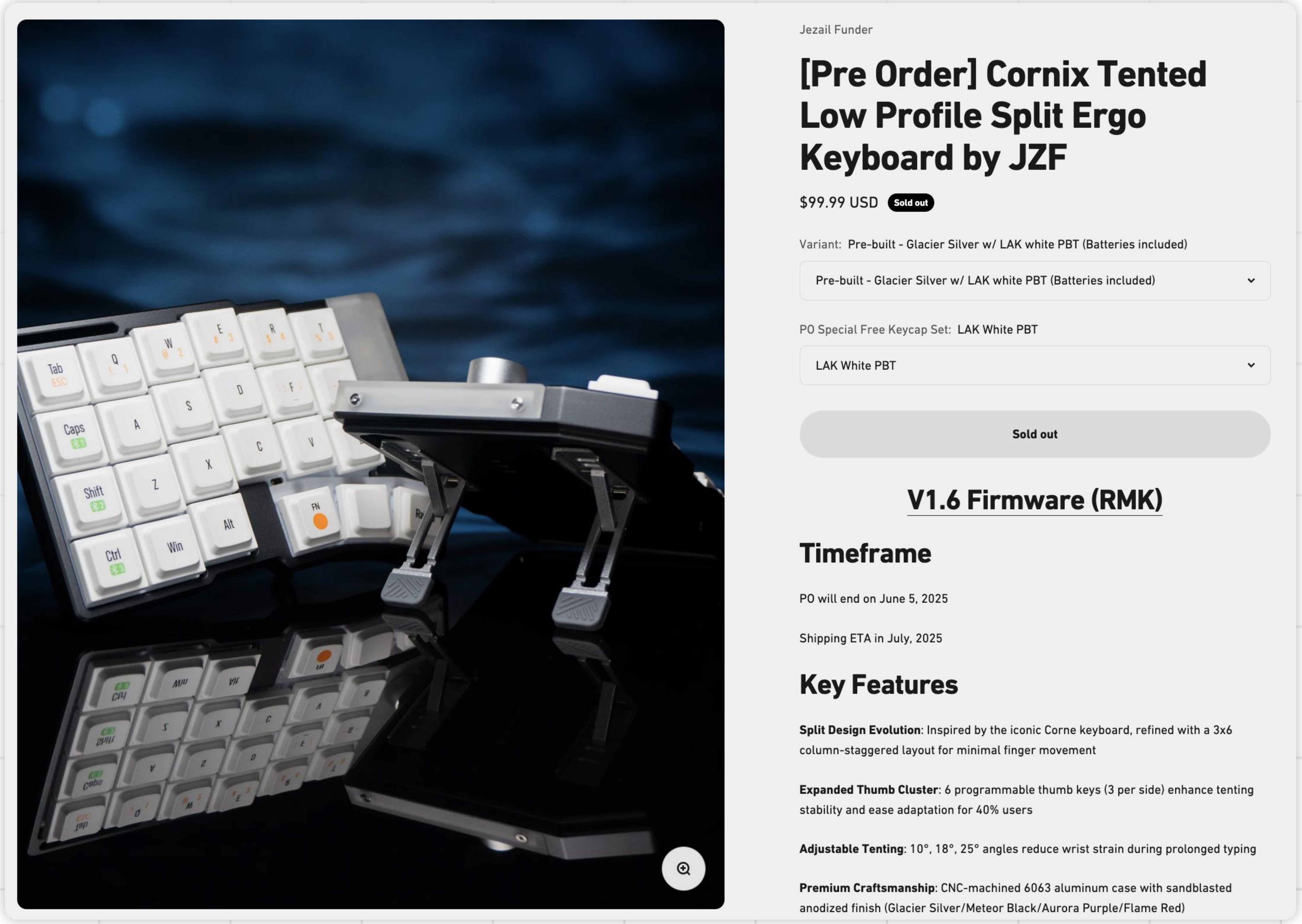
To mass production

Almost NOTHING

To mass production

Almost： Don’t await during borrowing RefCell, and enable clippy 🤣

To mass production



Jezail Funder

[Pre Order] Cornix Tented Low Profile Split Ergo Keyboard by JZF

$99.99 USD Sold out

Variant: Pre-built - Glacier Silver w/ LAK white PBT (Batteries included)

Pre-built - Glacier Silver w/ LAK white PBT (Batteries included)

PO Special Free Keycap Set: LAK White PBT

LAK White PBT

Sold out

V1.6 Firmware (RMK)

Timeframe

PO will end on June 5, 2025

Shipping ETA in July, 2025

Key Features

Split Design Evolution: Inspired by the iconic Corne keyboard, refined with a 3x6 column-staggered layout for minimal finger movement

Expanded Thumb Cluster: 6 programmable thumb keys (3 per side) enhance tenting stability and ease adaptation for 40% users

Adjustable Tenting: 10°, 18°, 25° angles reduce wrist strain during prolonged typing

Premium Craftsmanship: CNC-machined 6063 aluminum case with sandblasted anodized finish (Glacier Silver/Meteor Black/Aurora Purple/Flame Red)

Lessons learned

- It's 2025, don't ask is Rust good enough, just try you'll get answer
- Learn by writing real project
- Choose the hardware that is widely used in the community
- Contribute back to open-source!

Lessons learned

- Use Rust's unit test and integrated test for platform agnostic logic
- Modular design is easy(and great!) in Rust embedded
- A single task for a single peripheral is preferred
- Do refactor early

Challenge

- Binary size and memory usage: rust-lang/rust#62958
- Tooling: Trace? Async debugging?
- Ecosystem: embedded-hal, vendor support
- Resources for C/C++ developers

Links

- RMK : https://github.com/HaoboGu/rmk
- Discord channel: https://discord.com/invite/HHGA7pQxkG
- Slides: https://github.com/HaoboGu/RustConfChina2025

RustChinaConf 2025&
Rust Global China

Thank You!