Rust poisoning my wrist

Ulf Lilleengen (@lulf)

What I’ll talk about…

Meet the hardware!

- PineTime
- Nordic nRF 52832
  - 64 kB RAM
  - 512 kB onboard
  - Bluetooth Low Energy (BLE) radio
- 240x240 touch display
- Heart rate sensor
- 4 MB offboard flash

More info: https://wiki.pine64.org/wiki/PineTime

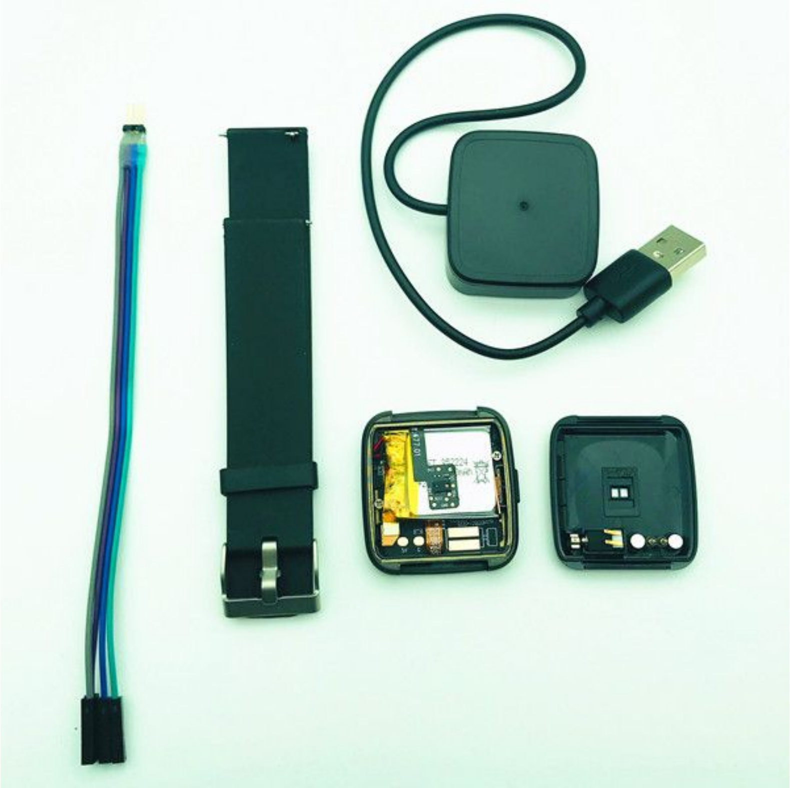
It's hackable!

Unsealed development kit

that you can flash using a

standard SWD probe

(a Raspberry Pi Pico for instance)



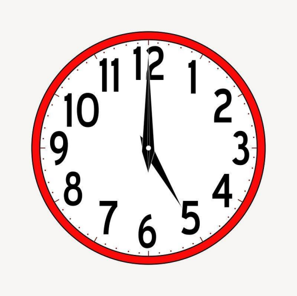
Meet the software!

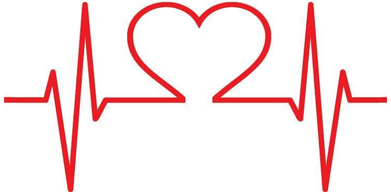
- Out of the box with InfiniTime open source RTOS

INFINI
TIME

Desired features

- Display time!
- Heart rate
- Find my phone
- Workout tracking





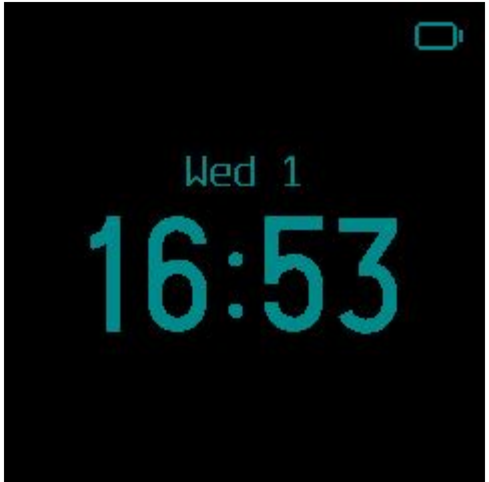
Non functional stuff

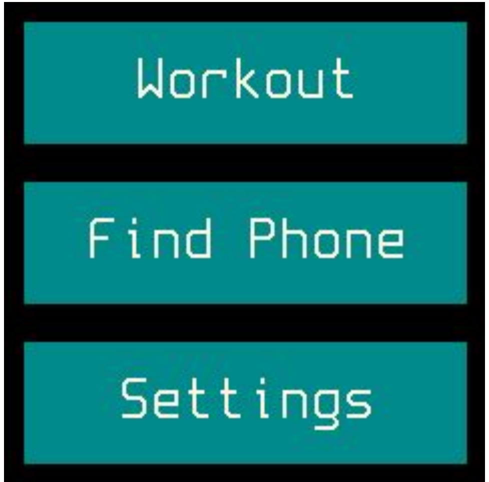
- Decent (1 week) battery drain
- Firmware updates over Bluetooth Low Energy

Step 1: Visuals

Embedded-graphics to the rescue!

- 2d graphics for embedded
- Simulator provides fast prototyping and testing of the UI
- display driver: st7789 well supported
- touch driver: cst816s well supported





Step 2: Connectivity

Solved by embassy + nrf-softdevice

- nRF52 + BLE well supported by Embassy
- nrf-softdevice (interface to proprietary blob) - a fully qualified BLE stack
- BLE Services
  - Device information service
  - Current time service (client)
  - Firmware update service (proprietary Nordic)
- nrf-dfu-target - implementation of the Nordic SDK DFU protocol

Step 3: Storage

Solution: DIY flash driver

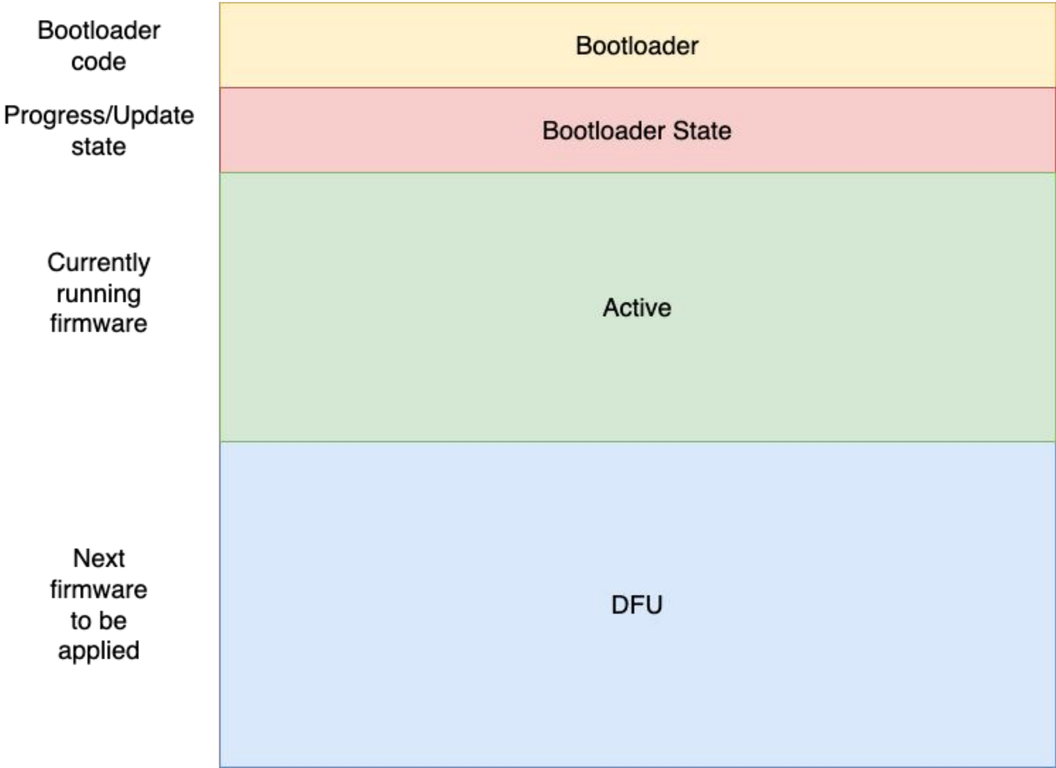
- Let your driver implement and consume HAL traits
  - Use: embedded-hal(-async)
  - Provide API: embedded-storage(-async)

```
impl<SPI: SpiDevice> ReadNorFlash for XtFlash<SPI> {

    fn read(&mut self, offset: u32, buf: &mut [u8])
-> Result<(), Self::Error> {
        todo!()
    }

    fn capacity(&self) -> usize {
        todo!()
    }
}
```

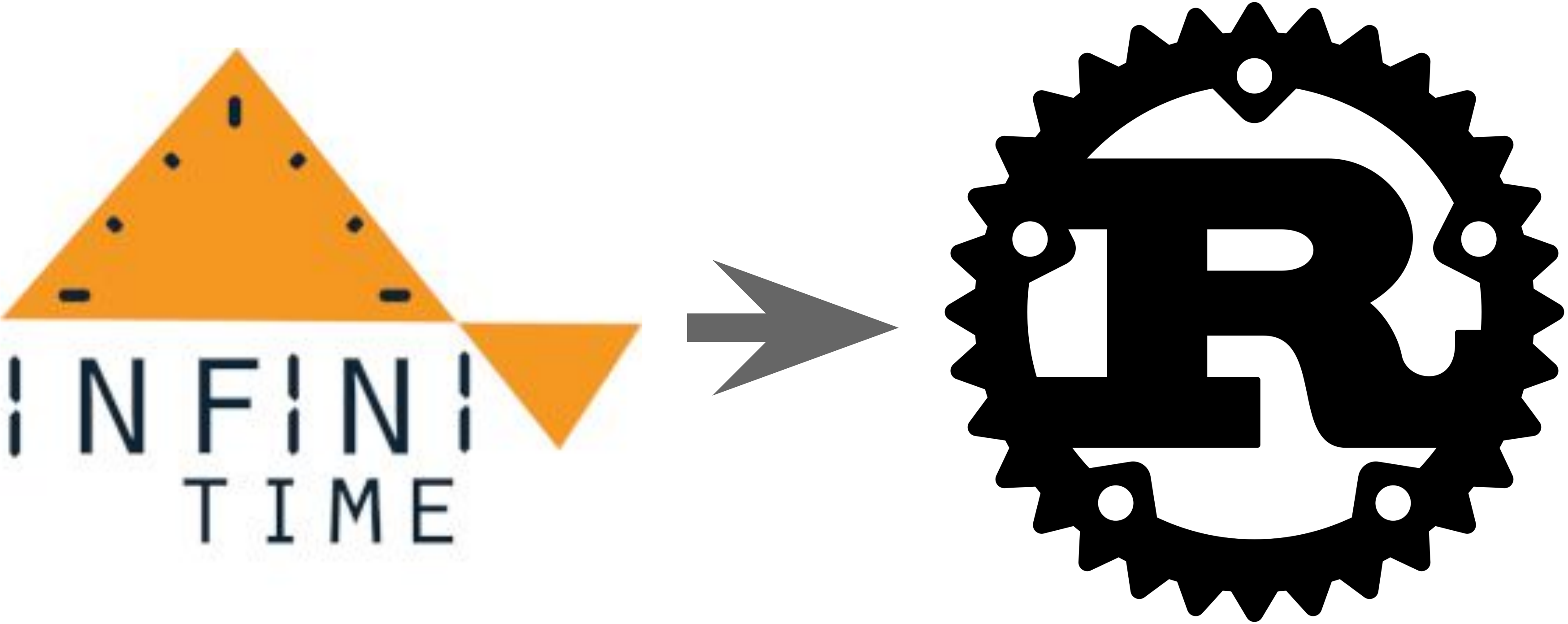
```
impl<SPI: SpiDevice> NorFlash for XtFlash<SPI> {
    fn erase(&mut self, from: u32, to: u32) ->
Result<(), Self::Error> {
        todo!()
    }

    fn write(&mut self, offset: u32, data: &[u8])
-> Result<(), Self::Error> {
        todo!()
    }
}
```

Firmware updates

- embassy-boot
  - simple A/B partition swapping
  - rollback
  - power fail safe

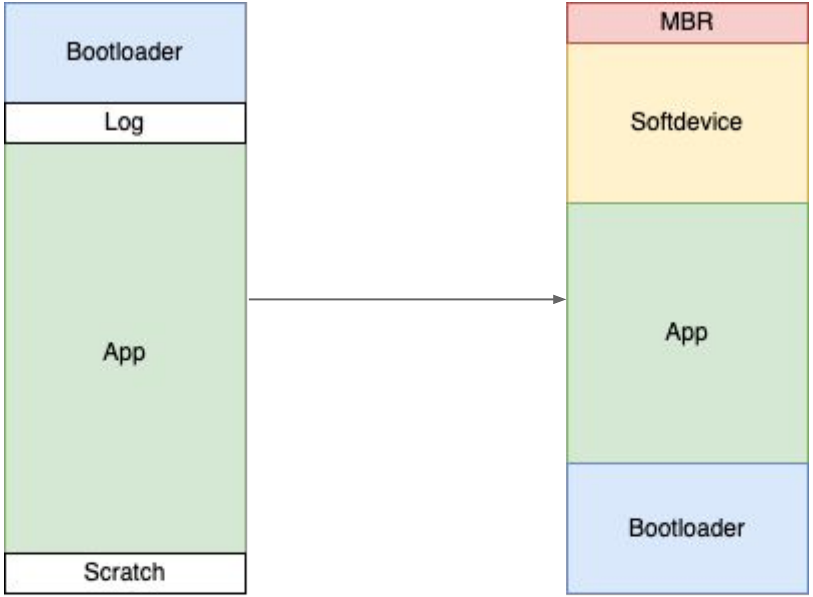


The great switcheroo

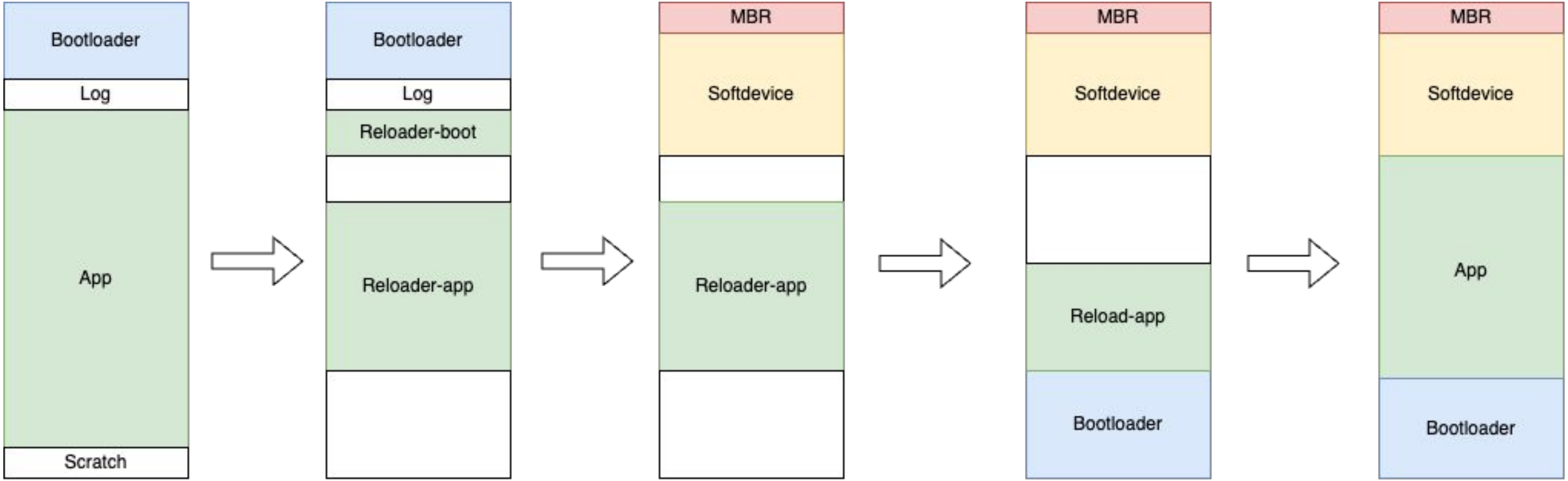


Bootstrapping

- Time for some dirty tricks!
- Infinitime uses NimBLE stack
- Rust requires softdevice stack



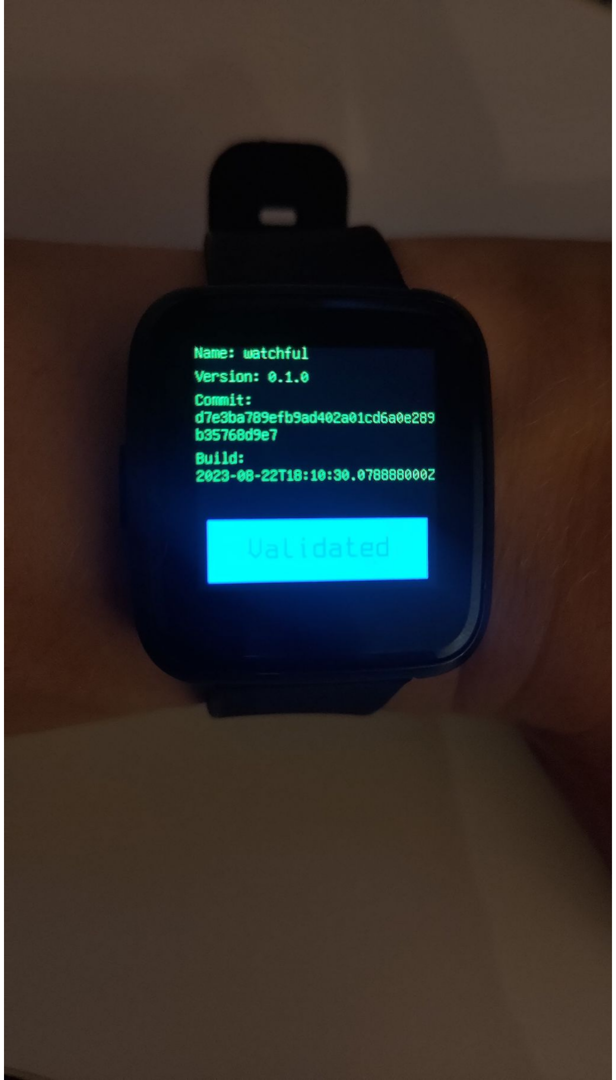
Bootstrapping process



The moment of truth…

The moment of truth...

... beware of the bugs!



```
Name: watchful
Version: 0.1.0
Commit:
d7e3ba789efb9ad402a01cd6a0e289
b35768d9e7
Build:
2023-08-22T18:10:30.078888000Z
```

Summary

- Display: beyond expectations
- Storage: beyond expectations
- Connectivity: matched expectations
- Testing lesson: test upgrade **and** downgrade path

- Increased appreciation of crates.io and Rust ecosystem
- In case I forget to mention: probe-rs is amazing

My Rust Wishlist

- Better support for multiple targets in a cargo workspace

- Make it easier to handle dependencies across repositories

- Keep taking embedded into account for new Rust features

Resources

- Watch RTOS: https://github.com/lulf/watchful/
- Embassy: https://embassy.dev
- Probe-rs: https://probe.rs/
- Embedded Rust: https://github.com/rust-embedded/

- In progress Rust BLE Host: TrouBLE

Thank you!

- GitHub: github.com/lulf
- Blog: lulf.no
- Mastodon: @lulf@hachyderm.io
- Bluesky: @lulf.no