probe-rs

My Name is

Noah Hüsser

aka Yatekii

I'm a

Swiss Software Engineer

Working at

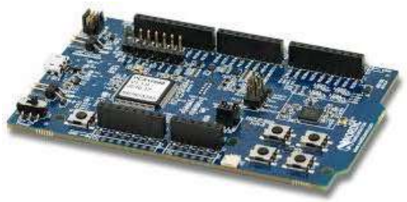
T e c h n o k r a t

https://technokrat.ch

In my Sparetime I like doing

Ju Jitsu

Embedded Debugging



There is Tools for that Already!

- Segger J-Link
- ST-Link
- CMSIS-DAP
  - Many variants
- Blackmagic Probe
- ...

- Segger J-Link Tools
- ST-Link Utility
- stlink-tool
- pyOCD
- OpenOCD
  - Many variants
- Custom Scripts & Programs
- ...

There are Drawbacks!

- Very Stiff Ecosystem.
- Always Eclipse & GDB Based.
- Every Manufacturer with their own Flavor.
- Plenty of Tools with Awesome Features that are not Compatible with Each Other.
- Embedded Development should be Fun!

Host

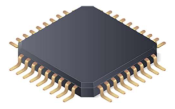
(probe-rs)

(probe-rs) USB $\updownarrow$ ETH Proprietary

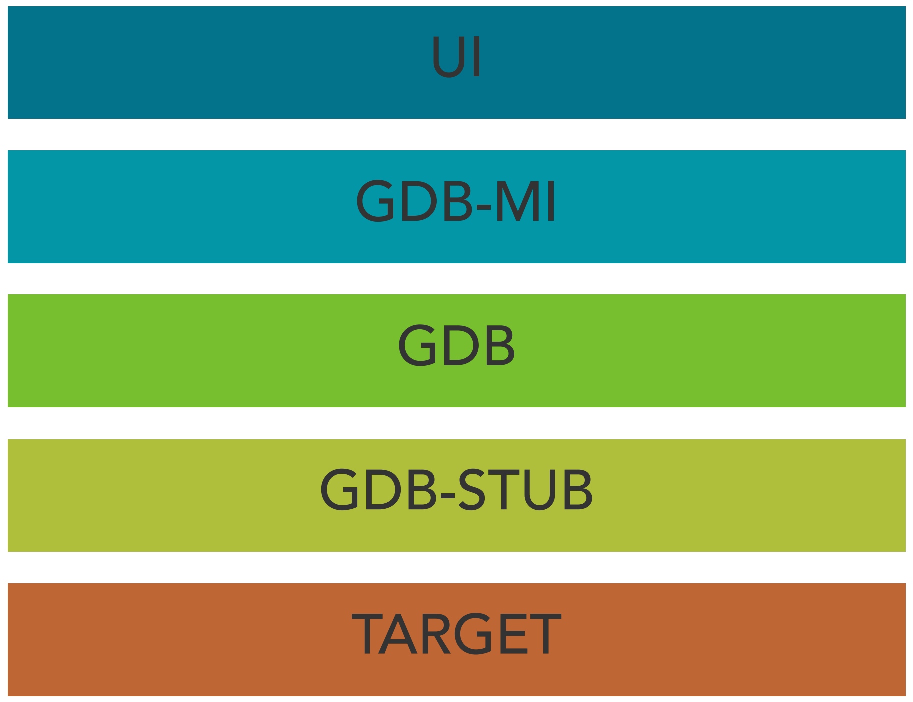
SWD/JTAG Peripheral
between
Host and Target

(probe-rs) SWD $\updownarrow$ JTAG Standardized

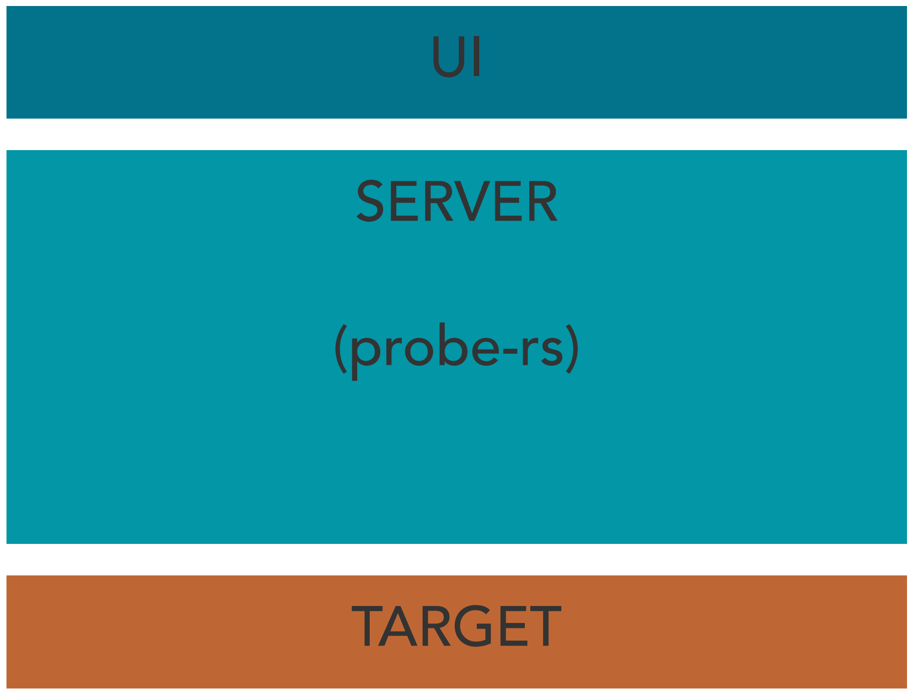
Target



The traditional way



The probe-rs way



The probe-rs benfits

- Editor of Your Choice
- Open, Streamlined & Documented API
- Simple Setup & Usage
- Less Friction with De-Facto Standards
- Rust Friendly but Language-Agnostic
- Project Encourages Contribution

A small Example.

Let us

- Flash a Program that Counts Up.
- Reset and Halt the CPU.
- Set a Breakpoint at the Counter Increment.
- Wait for the CPU to Halt.
- Read the Counter.
- Repeat the Last Two Points.

We open a new `Session`.

```
use probe_rs::Session;
use probe_rs::MemoryInterface;

let mut session = Session::auto_attach("nrf52")?;
```

Flash our ELF binary.

```rust
use probe_rs::Session;
use probe_rs::MemoryInterface;

let mut session = Session::auto_attach("nrf52")?;

probe_rs::flashing::download_file(
    &mut session,
    "./path/to/counter.elf",
    Format::Elf
)?;
```

Attach to `Core(0)`, reset the core and halt it.

```rust
use probe_rs::Session;
use probe_rs::MemoryInterface;

let mut session = Session::auto_attach("nrf52")?;

probe_rs::flashing::download_file(
    &mut session,
    "./path/to/counter.elf",
    Format::Elf
)?;

let timeout = std::time::Duration::from_millis(500);
let mut core = session.core(0)?;

core.reset_and_halt(timeout)?;
```

Set a breakpoint at the corresponding flash address.

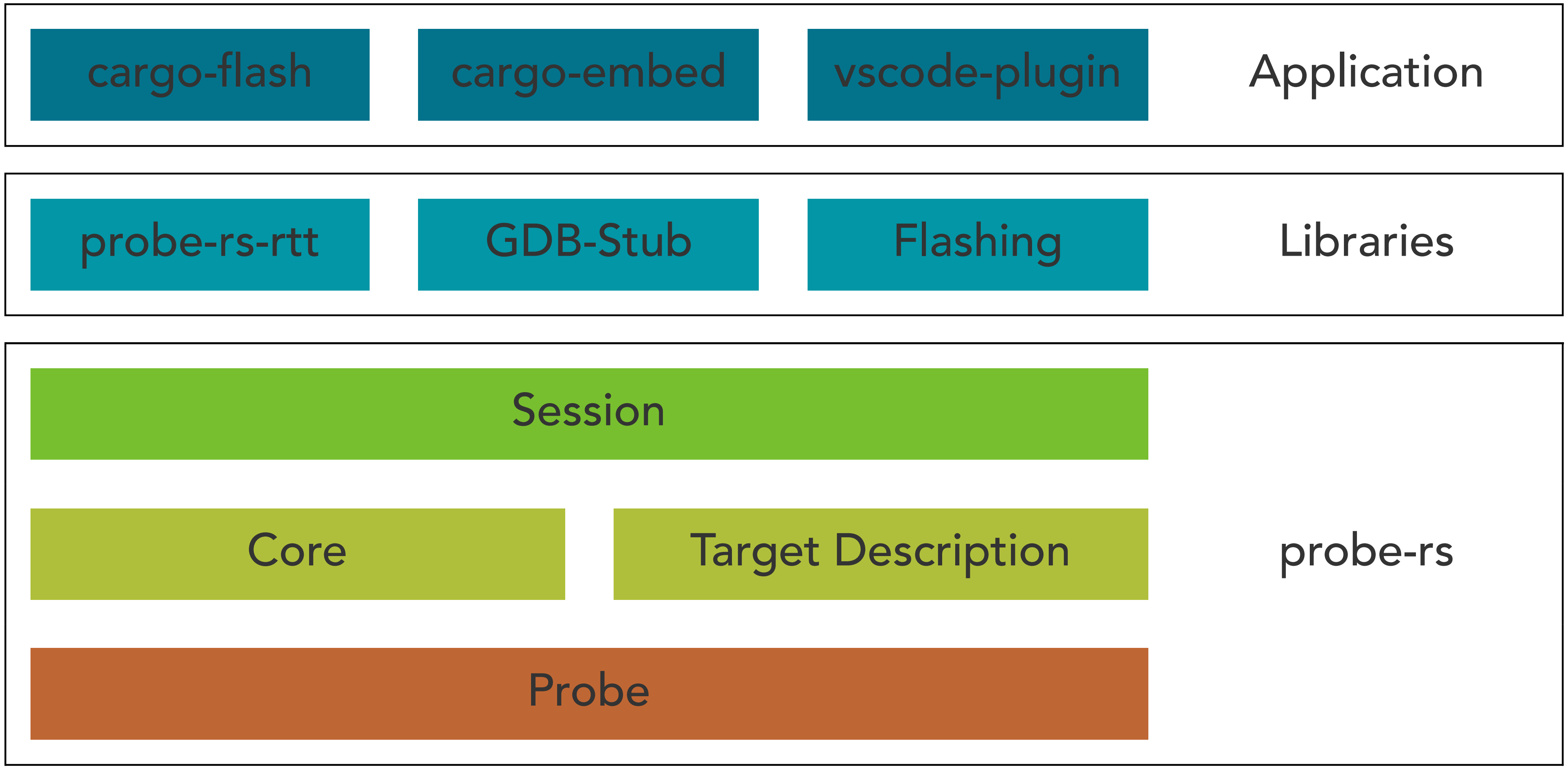
```rust
let timeout = std::time::Duration::from_millis(500);
let mut core = session.core(0)?;

core.reset_and_halt(timeout)?;

core.set_breakpoint(0x0000_0042)?;
```

Loop and read the counter.

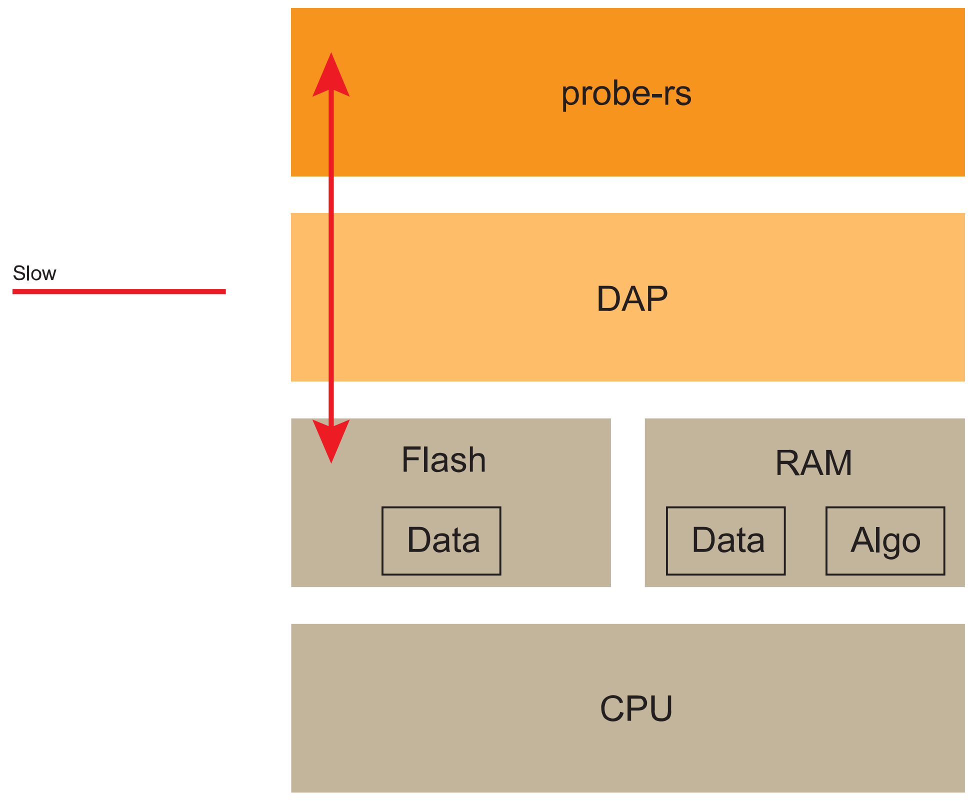
```rust
let timeout = std::time::Duration::from_millis(500);
let mut core = session.core(0)?;

core.reset_and_halt(timeout)?;

core.set_breakpoint(0x0000_0042)?;

loop {
    core.run()?;
    core.wait_for_core_halted(timeout)?;
    core.read_word_32(0x2000_1337)?;
}
```

We have Control over the Entire CPU!

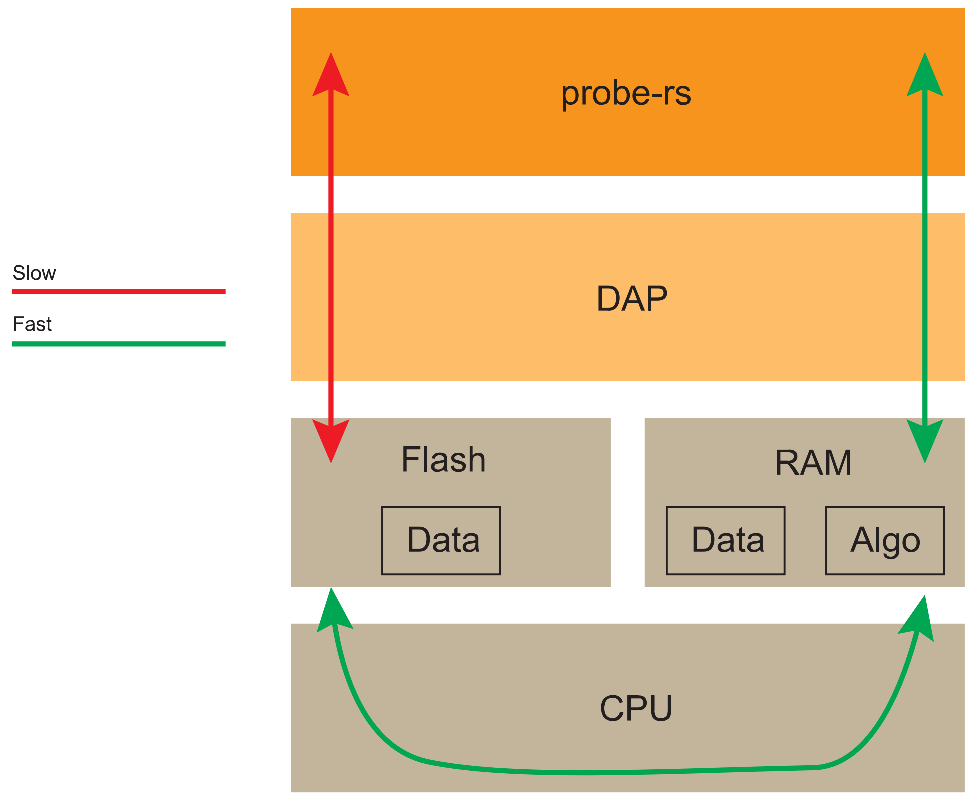
probe-rs Structure


cargo-flash
cargo-embed
vscode-plugin
Application
probe-rs-rtt
GDB-Stub
Flashing
Libraries
Session
Core
Target Description
probe-rs
Probe

Flashing



Flashing



CMSIS-Packs

- ARM Specification for Target Descriptions
- Contains Flash Algorithms
- Contains Custom Debug Sequences
- Implemented by Vendors!
- Thousands of Available Targets!

What's planned?

VSCode

- Plugin
- Microsoft DAP (Modern JSON API)
- No GDB!
- Status: Early Alpha
- https://github.com/probe-rs/vscode

ITM

- Powerful Data Streaming on ARM v7/v8
- ISR Event Tracking
- Memory Event Tracking
- Custom Binary Data
- Status: Around the Corner (Oxidize)
- https://github.com/probe-rs/probe-rs/pull/145

Custom Workflows

- ARM Debug Sequences
- Utilize CMSIS-Packs fully
- Unlocking, Attaching, Special Bytes, etc.
- Status: Concept Phase

More Stability

- The more Users the More Bugs ;)
- Speed Improvements
- Special Case Handling
- Status: Reoccuring

More Graphical Tools

- An ITM & RTT Tracer?
- ETM Tracer even?
- Flash Layout Visualization?
- Status: PoC in JS
- https://github.com/probe-rs/itm-tracer

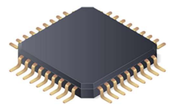
probe-rs-server

- Concurrent Use of probe-rs
- JSON-RPC
- Status: Being Worked on
- https://github.com/probe-rs/probe-rs/pull/293

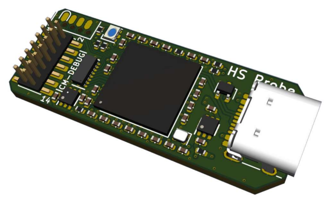
Demo time!

USB ↕ ETH

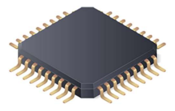
SWD ↕ JTAG



USB ↕ ETH



SWD ↕ JTAG



The hs_probe

- Open Source Probe
- Pure Rust Firmware
- Extremely Fast (up to 480 Mbit/s)
- Uses Standard ARM CMSIS-DAP
- Can Stream DAP, ITM and UART Data

Sign up for it

TODO: URL

Contribute!

- https://probe.rs
- https://github.com/probe-rs/probe-rs
- #probe-rs:matrix.org on Matrix
- Questions & Bugreports Very Welcome
- PRs even More Welcome ;)