A rusty way to store samples at liquid nitrogen temperatures

Reto Trappitsch

Why all of this?

Biological samples in a mass spectrometer: A problem with a solution!

- Biological samples are generally water
- We want to analyze samples in ultra high vacuum (UHV) ~10^{-10} mbar
- **Solution: Freeze the samples!**

Sample preparation

- High-pressure freezing vitrifies the sample
- Cutting, polishing, coating, imaging
- Immediate transfer to UHV prevents humidity crystallizing on sample

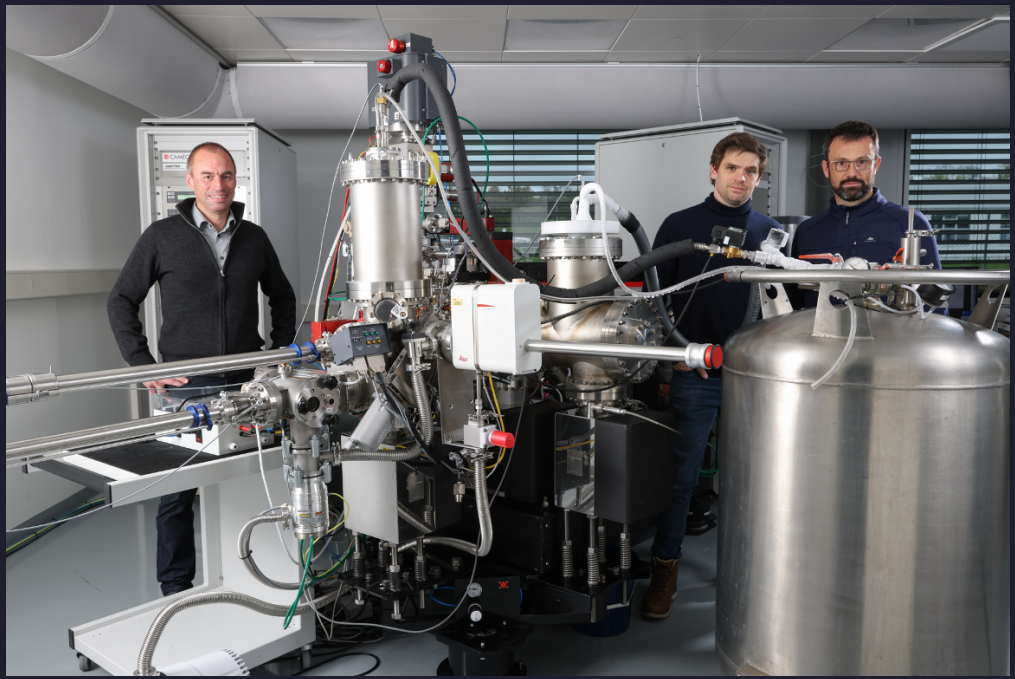
Sample analysis

- CryoNanoSIMS @EPFL
- Can hold 2 samples at liquid nitrogen temperatures for analysis

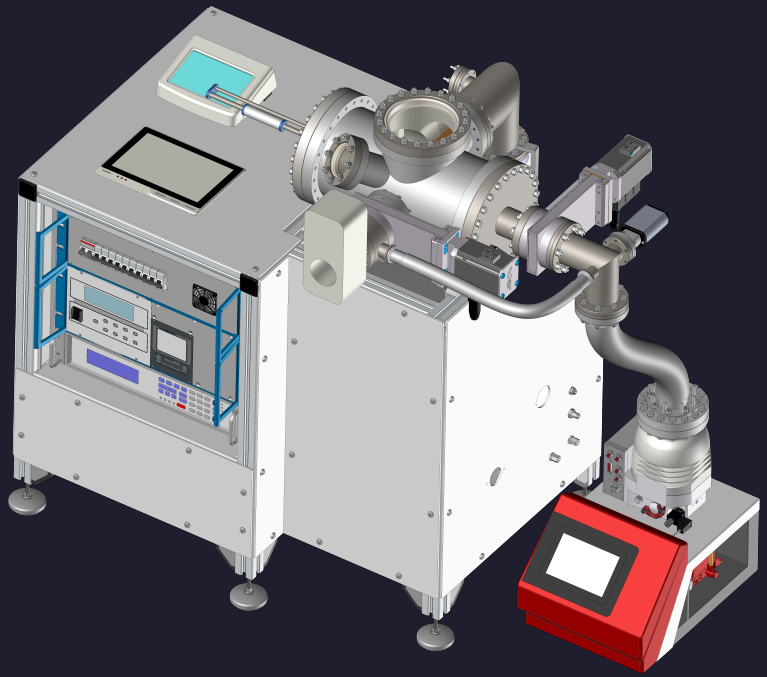
The current limitation

- Sample preparation can work on more than 2 samples at a time
- **We want to store samples in UHV at liquid nitrogen temperatures for days to weeks!**

The CryoNanoSIMS



The CryoStorage chamber



How can we make this work in a user friendly way?

Hardware - What we are dealing with

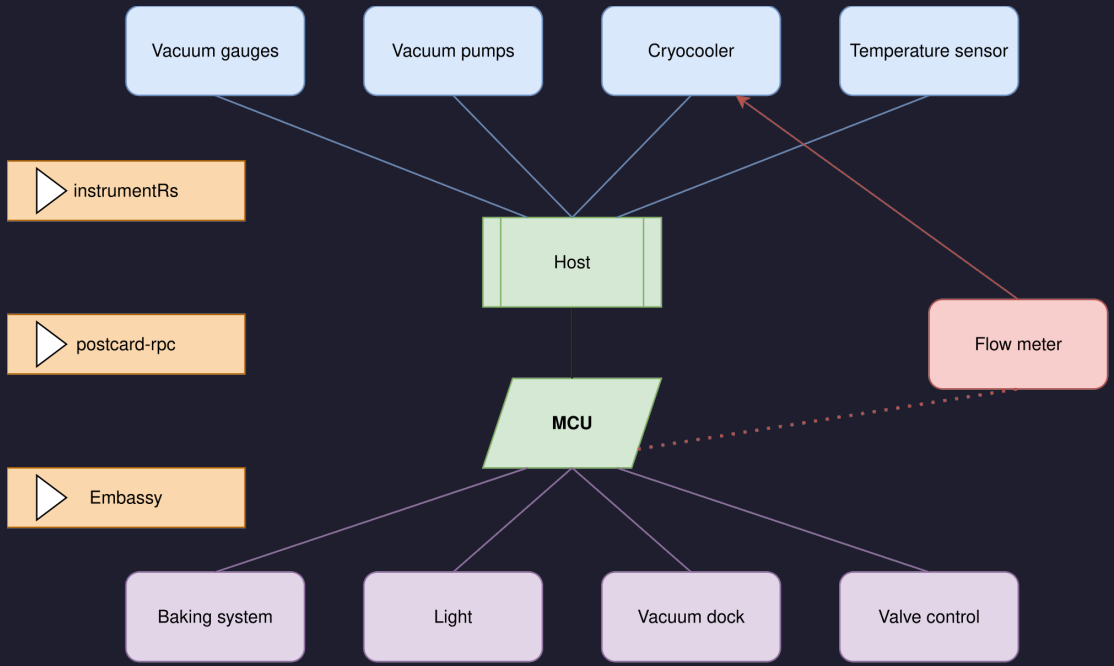
- 10" ReTerminal DM with touch screen (Raspberry Pi based)
- Various instrument controllers that communicate via RS-232, RS-485, TCP/IP
- Instruments that need to be controlled with digital signals (24 V)

What the user should see

- Touch UI on the ReTerminal
- Status of all instruments
- Control of all functions

The attack plan

How things hang together...

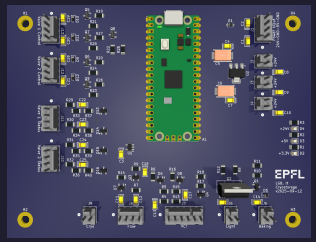


Oxidizing the CryoStorage chamber

Rust all the way down the stack

Digital IO instrument control

- Design electronics board to plug in a RaspberryPi Pico 2
  - Dual Cortex-M33 @150 MHz
  - 4 MB flash
  - 512 kB SRAM
- Firmware:
  - Embedded Rust
  - **Embassy**: EMBedded ASYnc



Communications MCU <-> Host

- postcard and postcard-rpc for RPC over USB
- RPC calls and broadcasts

Host level

- Poststation simplifies host communication
- slint UI for touch interface
- instrumentRs drivers for each additional instrument
  - Had to build this from scratch :)
- tokio async runtime on host

slint
GUI toolkit

The current status of our touch UI

Hello embedded world with embassy

- No heap -> no std!
- Access the hardware:
  - **PACs** (peripheral access crates)
  - **HALs** (hardware abstraction layer)
  - **BSCs** (board support crates)
- Embassy provides us with an async executor
- Various Rust embedded rooms on Matrix
  - Great community support and help!
- **Firmware can be developed in safe Rust**
  - We get all the benefits of Rust

```rust
#![no_std]
#![no_main]

use defmt::*;
use embassy_executor::Spawner;
use embassy_rp::gpio;
use embassy_time::Timer;
use gpio::{Level, Output};
use {defmt_rtt as _, panic_probe as _};

#[embassy_executor::main]
async fn main(_spawner: Spawner) {
    let p =
embassy_rp::init(Default::default());
    let mut led = Output::new(p.PIN_25,
Level::Low);

    loop {
        info!("led on!");
        led.set_high();
        Timer::after_millis(250).await;

        info!("led off!");
        led.set_low();
        Timer::after_millis(250).await;
    }
}
```

Take-aways: Why Rust and what's to come?

Embedded Rust

- Rust is great for embedded development
- Mature ecosystem, great community

Are we GUI yet?

- Slint: great for GUIs on limited hardware
- Full desktop applications... soon?
  - https://slint.dev/blog/making-slint-desktop-ready

Even Ferris likes the cold!

Future plans

Electronics

- Test our reworked second edition board
- Design and build third version

instrumentRs

- Some instruments we need are still missing
- instrumentRs has some weird design choices at the moment that need to be thought over
- It works... but could be better!

Host software

- Test and integrate with the system