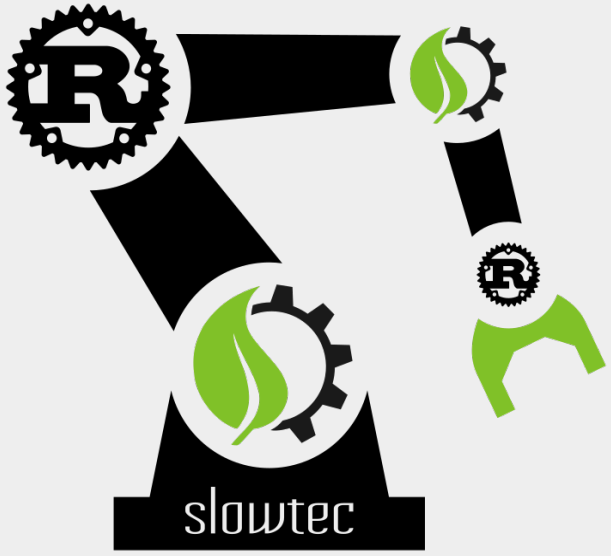
Industrial Automation



with Rust, Embedded Linux, and Open Hardware

Who we are...

Dipl.-Ing. Markus Kohlhase

Dipl.-Inform. Uwe Klotz

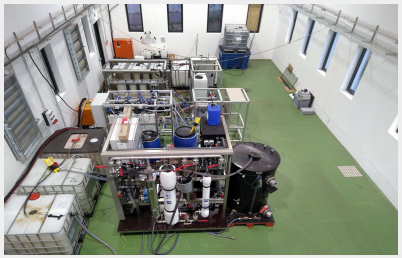
B.Sc. Enja Stein

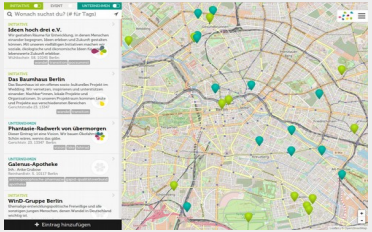
What we do...

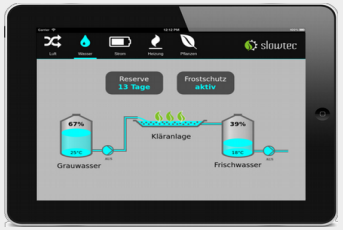
- Software Engineering
- Automation & Closed-loop Control
- Industrial 4.0
- (Web) App Development
- Product Prototyping
- Consulting

Some of our projects...

- Water treatment plants
  - Closed-loop control
  - HMI (Web app)
- Solar power plants
- Irrigation systems
  - Product design
  - Software development
  - I/O Systems engineering (Partner: Relumity)
- Laboratory software
- Off-grid house automation
- Geo. information system

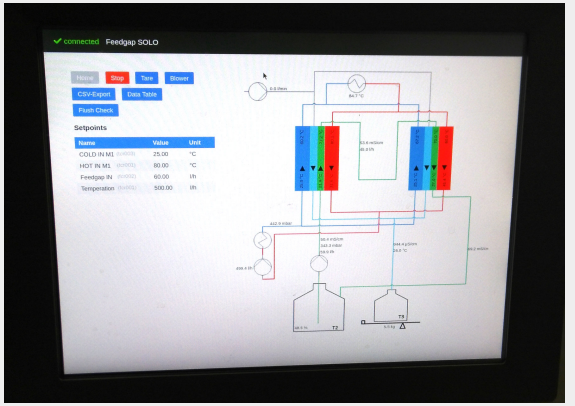




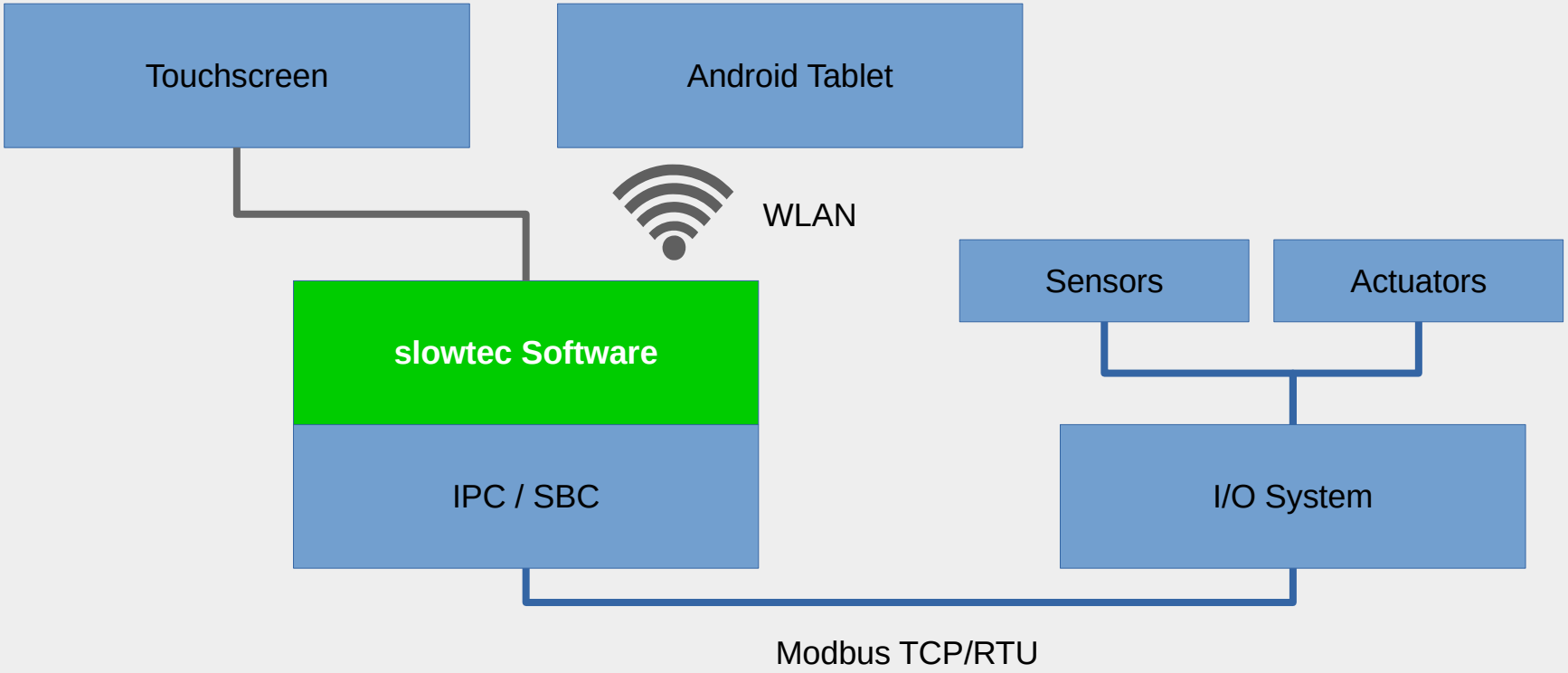


An industrial use case...

- ~ 40 Sensors
- ~ 15 Actuators



Architecture



Open Hardware
(...our experiences so far)

- I/O System
  - ~~IndustrialShields~~ ← no RTD, insufficient quality
  - ~~Kunbus~~ ← no AO/AI in 2016
- Industrial PC
  - ~~Raspberry Pi~~ ← unreliable
  - ~~Odroid~~ ← unreliable, depending on environmental conditions
  - ~~IndustrialShields~~ ← no Linux
  - Olimex ? ← no experiences so far

I/O System & Industrial PC
(Proprietary)

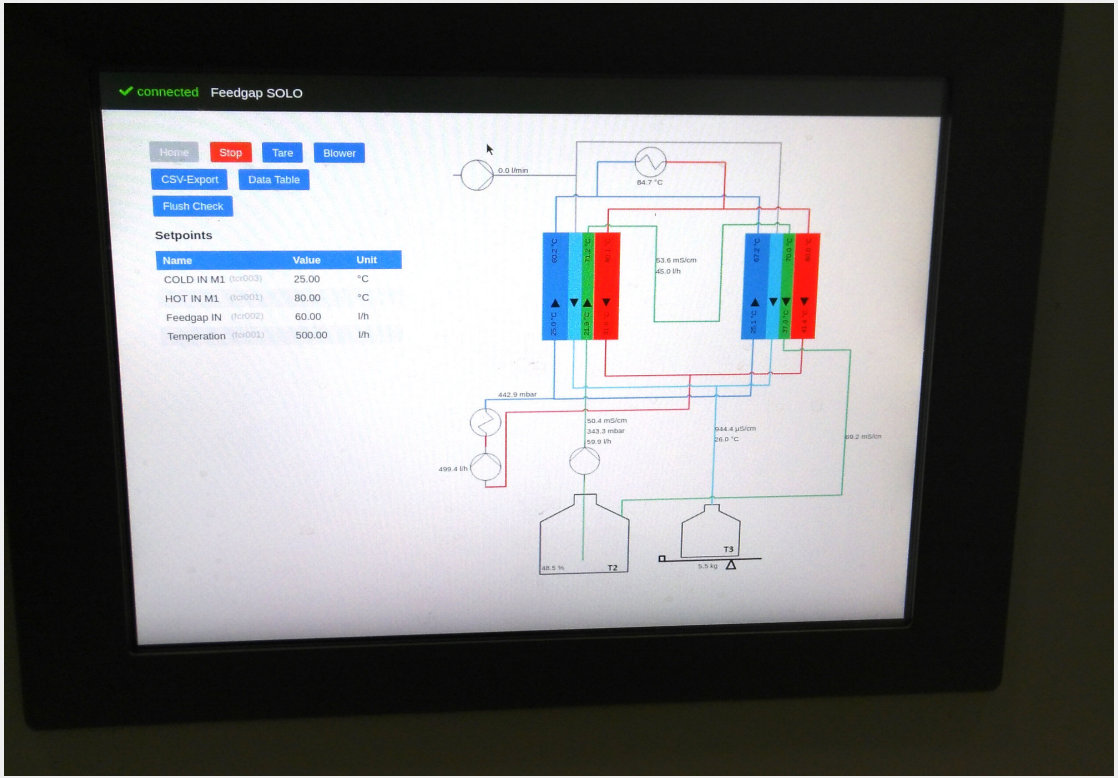
Our own open embedded industrial I/O board powered by Linux & Rust

- MIPS @ 1 GHz
- 128 MB RAM
- 32 MB NOR Flash
- versatile connectivity

Partner: Relumity, Stuttgart

Some insights...

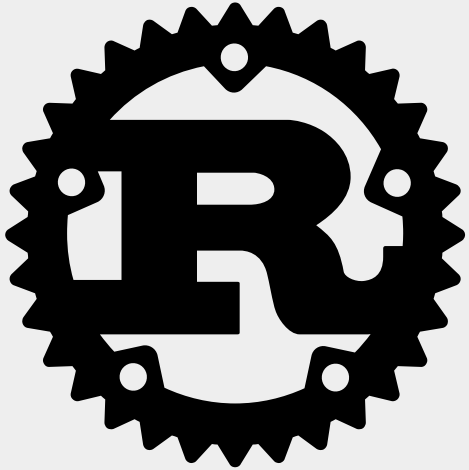
HMI (Touchscreen with Web App)

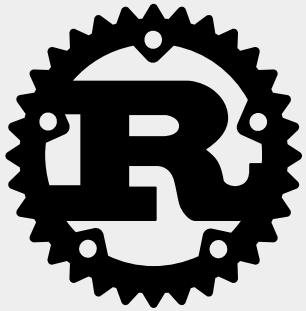


What is NixOS?

NixOS is a Linux distribution with a unique approach to package and configuration management.
It is **completely declarative**, makes upgrading systems **reliable.**

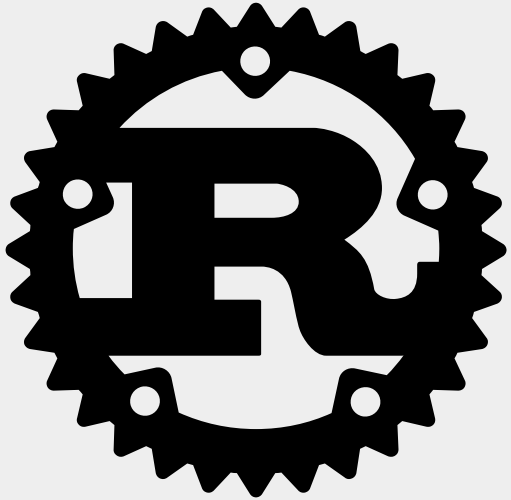
What is Rust?





Rust is an **open source** systems programming language with a focus on **safety**.

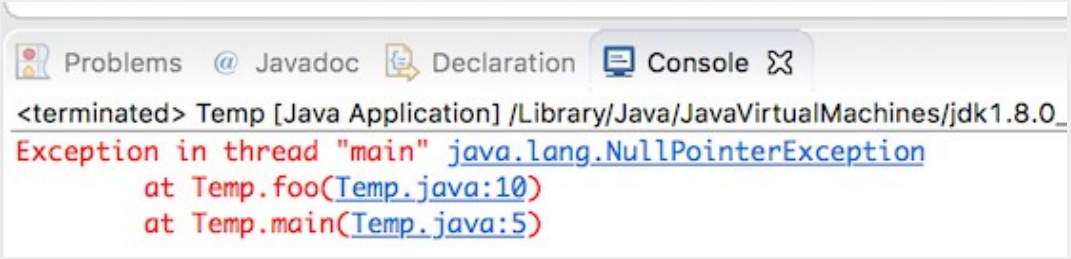
Why Rust?



“fast, reliable, productive – pick three”

Reliability

Why does this still happen in 2018?




```
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x20 pc=0x40142f]
```

...or even worse: Undefined behavior 😱

Rust Pros

- Safety & Reliability
- Security
- Predictability (hard-realtime)
- Efficiency & Performance
- Productivity
- Maintenance / Long Term Support
- Community-drive and Open Source
- Cross-platform / Embedded / Bare Metal
- Tooling
- Deployment

Language Goodies

- Powerful type system
  - Pattern matching a.k.a. *“switch on steroids”*
  - Product types (structs, tuples) with methods
  - Sum types (enums) with data and methods
  - Light-weight OO through *Traits*
- Zero-cost abstractions
  - Resolved at compile-time
- Explicit error handling, no runtime exceptions
- Borrow-checker
  - Many memory and concurrency errors become **impossible**
- Allows (some) functional programming patterns
- Interoperability with C/C++
  - FFI + bindgen
- Tooling – *Batteries included*

Rust Cons

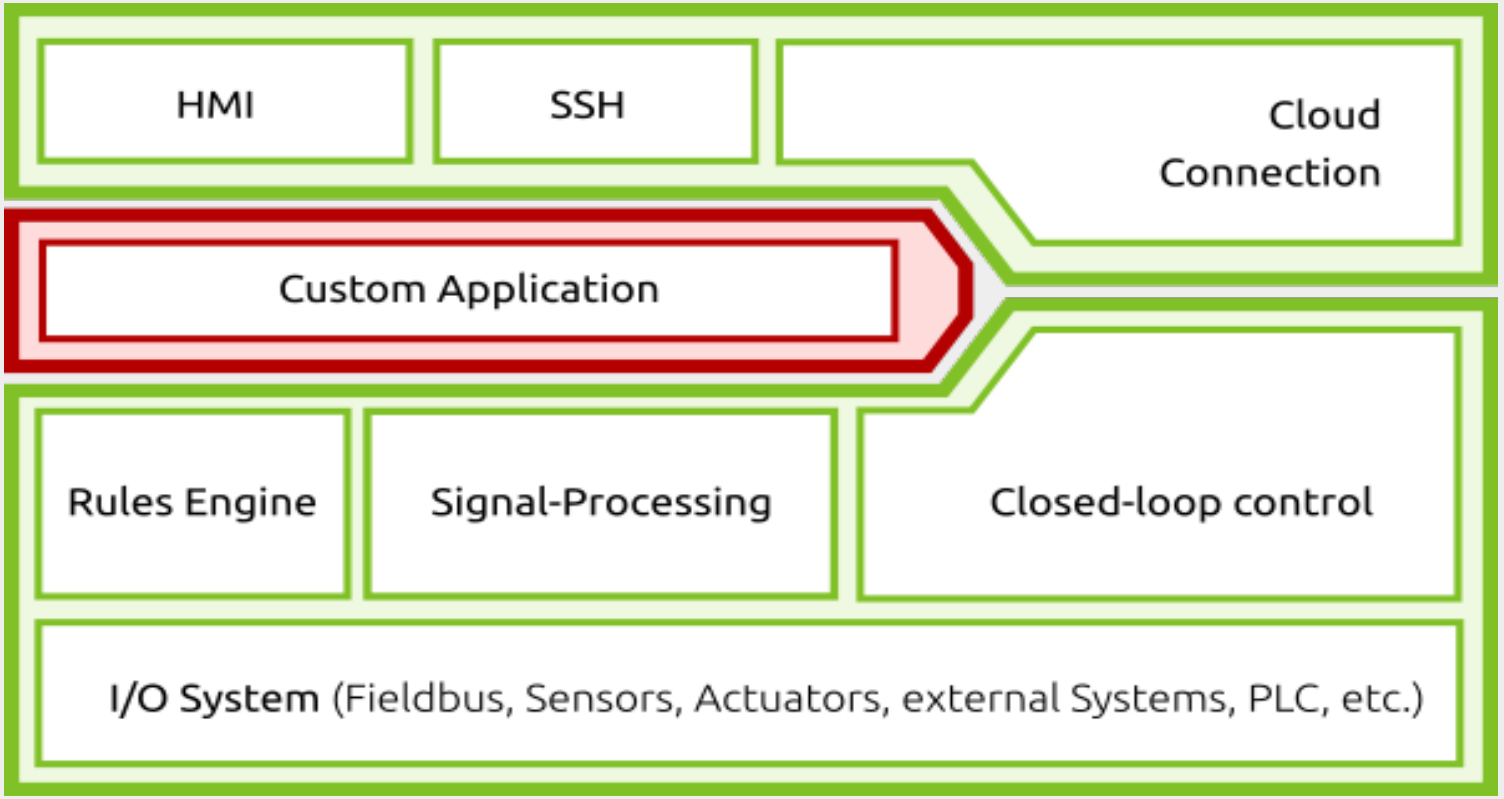
- Ecosystem is young and evolving
  - expect breaking changes when upgrading
- Pure Rust solutions are missing
  - use C wappers (FFI/bindgen)
- Async primitives are missing
  - still unhandy, i.e. requires some boilerplate
  - async/await announced for 2019
- No generators, coroutines, or higher-kinded types, yet
  - strict type system and borrow-checker take their toll
  - limits of the language design need to be pushed
- No ABI

Blazingly fast, compact, efficient

- Single, statically linked binary
- 6-10 MB in total (uncompressed)
- Various communication channels
  - HTTP
  - WebSocket
  - MQTT
  - ...
- Embedded Web Application (SPA)
- Embedded SQLite DB

How to build a controller without programming?

slowtec Open IIoT Stack



1. Configure your I/O system

```
[io_systems.main.ur20]
  ip = "192.168.0.222"
  cycle_time_ms = 600

  [io_systems.main.ur20.map]

    # Temperature
    tcr001 = [7, 0]
    tcr002 = [7, 2]

    # Pumps
    p1 = [11, 1]
    p2 = [11, 2]
```

2. Configure your sensors / actuators

```
[inputs.fcr001]
  title = "Temperation"
  unit = "l/h"
  crop = { low = 0.0 }
  [inputs.fcr001.scale]
    from = { low = 4.0, high = 20.0  }
    to   = { low = 0.0, high = 100.0 }

[outputs.p1]
  title = "Temperation pump P1"
  [outputs.p1.scale]
    from = { low = 0.0, high = 100.0 }
    to   = { low = 0.0, high = 5.0   }
```

3. Define your controllers

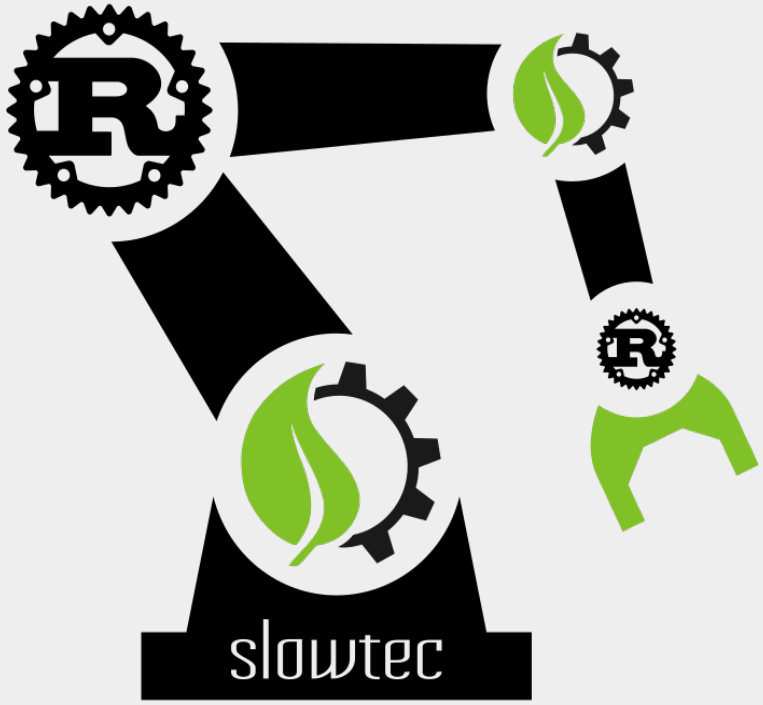
```
[controllers.condensor_temp]
  input  = "tcr003"
  output = "h1"

  [controllers.condensor_temp.pid]
    p = 2.0
    i = 0.003
    d = 0.1
    i_max = 80.0
    max = 90.0
    min = 30.0

  [controllers.condensor_temp.setpoint]
    constraint = { min = 20.0, max = 45.0 }
    Default = 25.0
```

4. Run & have fun!

```
$ slowtec-iot config.toml
```

Other features

- Rules / Actions
- State Machines
- Recording
- etc.



Thank you!

https://www.slowtec.de

https://github.com/slowtec