Tamme Dittrich

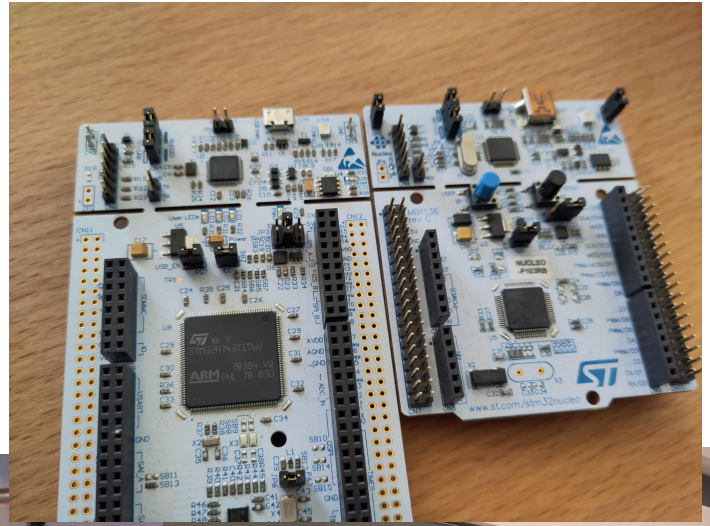
Chips don’t bite!

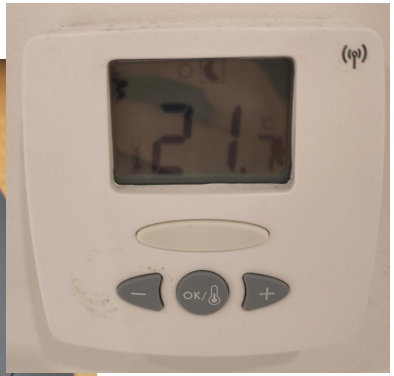
Who am I?

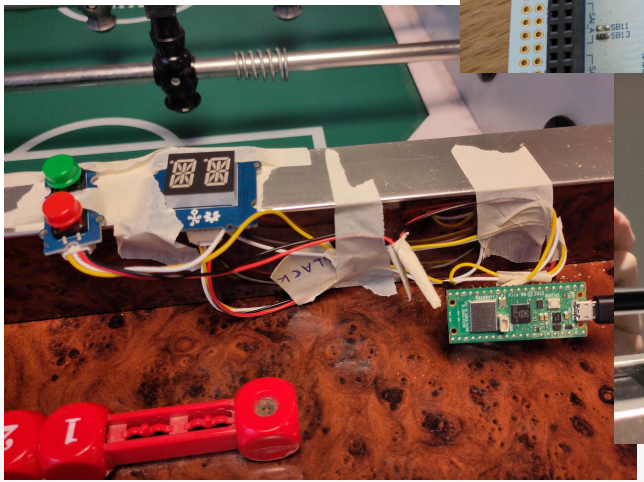
Tamme Dittrich

- Embedded software engineer @ Tweede golf
  - Soft- and Firmware in Rust
  - Rust trainer
- Getting people started with Rust
  - Training
  - Team augmentation
  - Mentoring, Reviewing, Pairing, ...

Chips



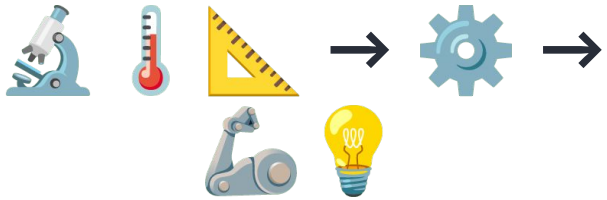

21
OK

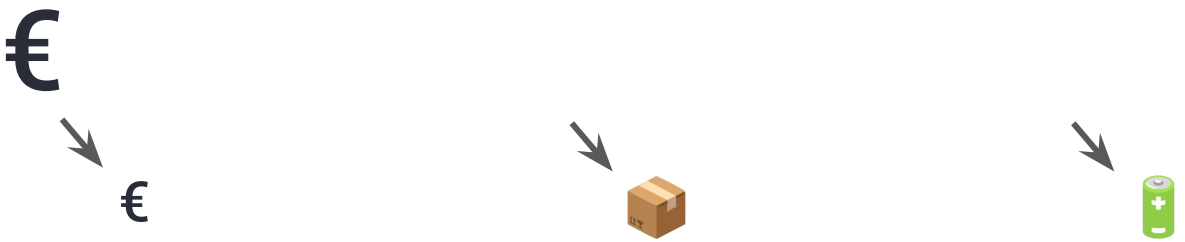



P
INDESIT
ON/OFF

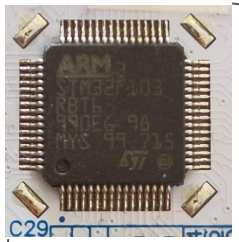

Espresso
Cappuccino
Kaffe
Melk-skum
jura

Why?





What's inside?



CPU

WiFi

Bluetooth

Flash

RAM

GPIO

Ethernet

LED

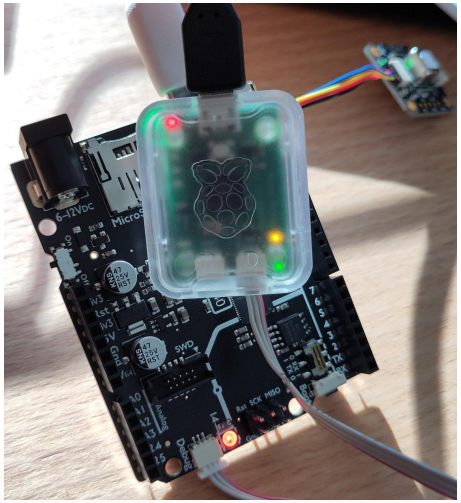
Button

Blink

```
let mut led = Output::new(p.PIN_25, Level::Low);

loop {
    for i: i32 in 1..=1000 {
        led.set_high().unwrap();
        Timer::after_millis(i).await;
        led.set_low()?;
        Timer::after_millis(i).await;
    }
}
```

Hello World



```rust
use rtt_target::{rprintln, rtt_init_print};

fn main() -> ! {
    rtt_init_print!();
    loop {
        rprintln!("Hello, world!");
    }
}
```

```
❯ cargo run
   Compiling metro-co2 v0.1.0 (/home/tamme/dev/metro-co2)
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.75s
     Running `probe-rs run --chip RP2040 target/thumbv6m-none-eabi/debug/metro-co2`
      Erasing ✔ [00:00:01] [########################################] 144.00 KiB/144.00 KiB @ 72.79 KiB/s
  Programming ✔ [00:00:04] [########################################] 144.00 KiB/144.00 KiB @ 34.73 KiB/s
    Finished in 6.158s
Hello, world!
```

Serverland: Build a HTTP client

Examples

```rust
use std::io::prelude::*;
use std::net::TcpStream;

fn main() -> std::io::Result<()
    let mut stream = TcpStream:

    stream.write(&[1])?;
    stream.read(&mut [0; 128])?
    Ok(())
} // the stream is closed here
```

Internet Engineering Task
Request for Comments: 91
Category: Standards Track
Published: June 2022
ISSN: 2070-1721

Abstract

The QUIC transport
in a transport for
control, and low-la
describes a mapping
identifies HTTP/2 f
how HTTP/2 extensio

Internet
Request fo
Category: Sta
Published: May
ISSN: 2070-1721

Abstract

Bisho

ngar, Ed.
Fastly
Thomson, Ed.
Mozilla

Task Force (IETF)
8446
061

E. Rescorla
Mozilla
August 2018

ity (TLS) Protocol Version 1.3

Abstrac

This doc es version 1.3 of the Transport Layer Security (TLS) prot S allows client/server applications to communicate over the Inte in a way that is designed to prevent eavesdropping, tampering, and message forgery.

This document updates RFCs 5705 and 6066, and obsoletes RFCs 5077, 5246, and 6961. This document also specifies new requirements for TLS 1.2 implementations.

We got cargo!

- `cargo add reqwest tokio …`
- Copy an example
- Done!*

Making a GET request

For a single request, you can use the `get` shortcut method.

```
let body = reqwest::get("https://www.rust-lang.org")
    .await?
    .text()
    .await?;

println!("body = {body:?}");
```

From: docs.rs/reqwest

Same on embedded

- `cargo add scd4x`
- Copy an example
- Done!*

```rust
let mut sensor = Scd4x::new(i2c, timer);

sensor.start_periodic_measurement().unwrap();
loop {
    timer.delay_ms(5000u32);

    let data = sensor.measurement().unwrap();

    rprintln!(
        "CO2: {0}, Temperature: {1:#.2} °C, Humidity: {2:#.2} RH",
        data.co2,
        data.temperature,
        data.humidity
    );
}
```

Based on: GH hauju/scd4x-rs/examples/linux.rs

Demo time

!!!

And now?

- Try it
- Ask Questions
- Tell others

Resources

- Discovery Book
- YouTube: Rusty Bits
- Matrix
- drive-rs

Thanks

Getting in touch

Contact me via e-mail or @tamme@fosstodon.org

Tamme Dittrich

Embedded software engineer
tamme@tweedegolf.nl

https://github.com/tdittr/rustlab-2024