Messomat 7K

mit Rust und Angular

```
38
39          // build our application with a single route
40          let app = Router::new()
41
42          .route("/all", get(get_all)
43          .with_state(sensordb))
44
45          .route("/status", get(get_status)
46          .with_state(command_sender.clone()))
47
48          .route("/set-fan", post(post_set_fan)
49          .with_state(command_sender.clone()))
50
51          .route("/set-send-mode", post(post_set_send_mode)
52          .with_state(command_sender.clone()))
53
54          .route("/set-on", post(post_set_on)
55          .with_state(command_sender.clone()))
56
57          .route("/set-off", post(post_set_off))
58          .with_state(command_sender.clone())
59
60          .layer(cors);
```

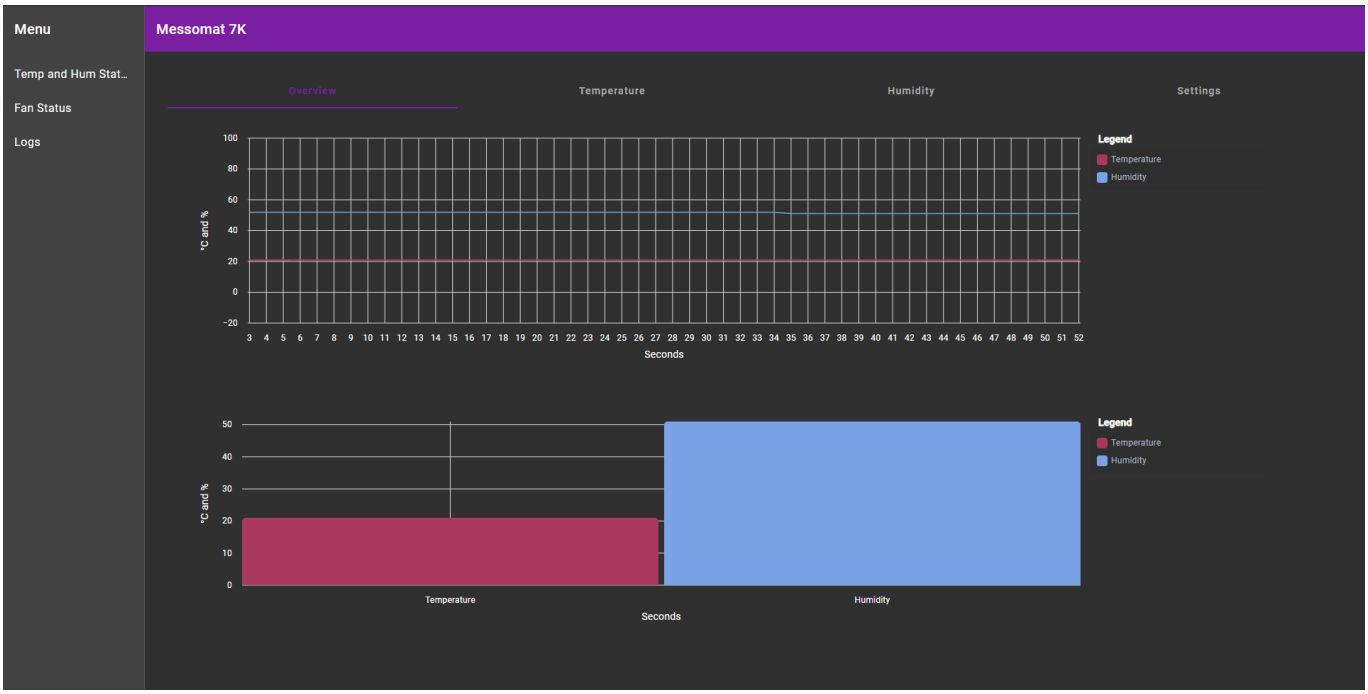
Backend

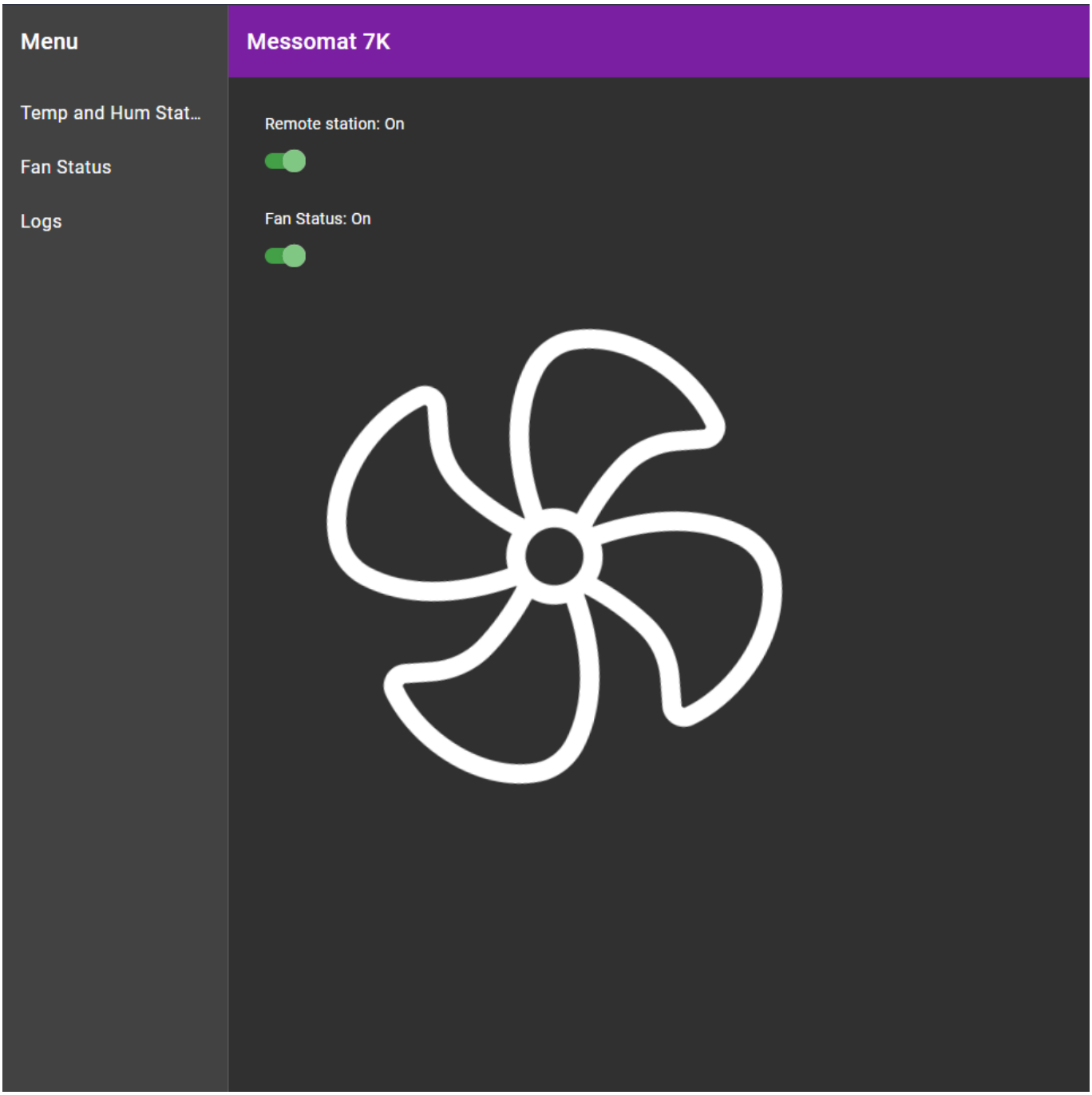
- Rust Nightly
- Tokio Async Runtime
- Tokio Serial (Serialport Library)
- parallel Request Handling
- Protocol fully mapped in Rust

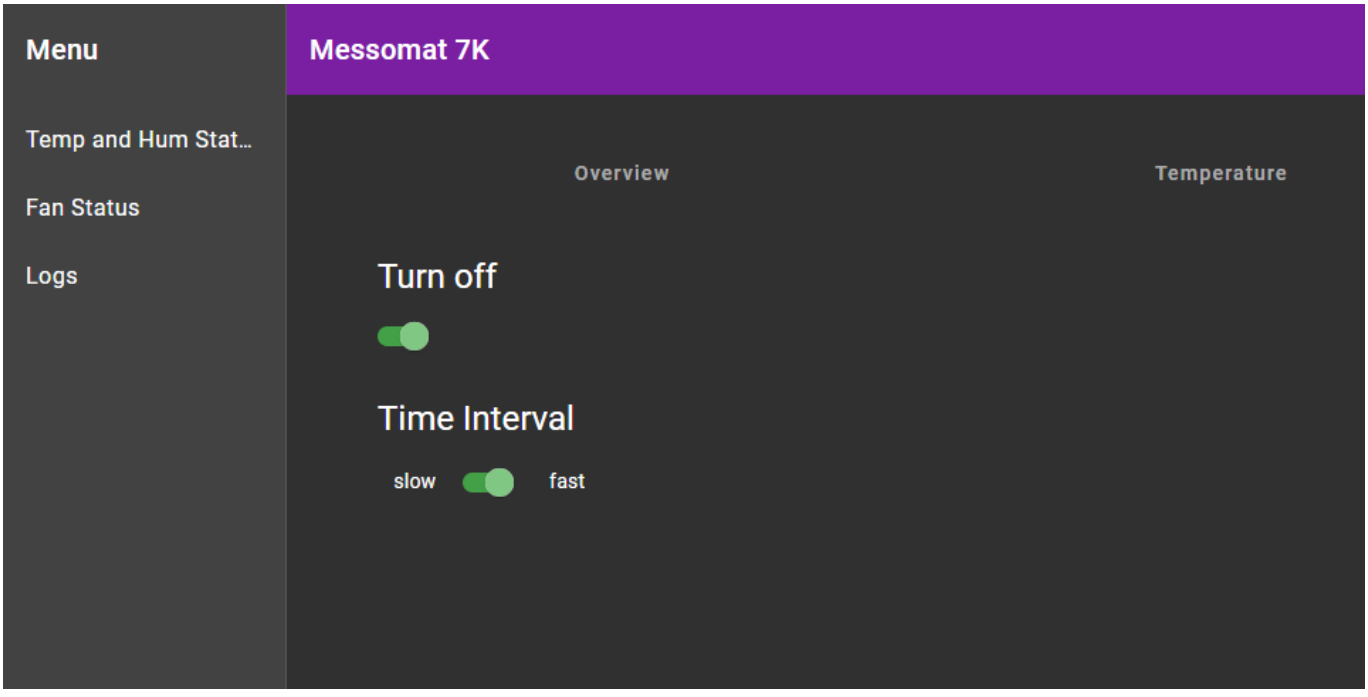
```rust
use anyhow::anyhow;
use tokio_serial::{SerialPortBuilderExt, SerialStream};

use std::time::Duration;

use tokio_serial::{DataBits, FlowControl, Parity, StopBits};

pub fn create_serialport(port_name: &str) -> Result<SerialStream, anyhow::Error> {
    tokio_serial::new(port_name, 9600)
        .data_bits(DataBits::Eight)
        .stop_bits(StopBits::One)
        .parity(Parity::None)
        .flow_control(FlowControl::None)
        .timeout(Duration::from_secs(1))
        .baud_rate(9600)
        .open_native_async()
        .map_err(|e| anyhow!(e))
}
```

Frontend

- Angular (JS Framework)
- ruft API auf
- nutzt Angular Material (UI-Library wie MudBlazor)
- nutzt NgxCharts für Diagramme (Line & Bar Charts)




Menu
Temp and Hum Stat...
Fan Status
Logs
Messomat 7K
Remote station: On
Fan Status: On


Menu
Temp and Hum Stat...
Fan Status
Logs
Messomat 7K
Overview
Temperature
Turn off
Time Interval
slow
fast

THX