Relatório 1

EEL7801

Aluno: Carlos Freitas

28 de Setembro de 2023

Desenvolvimento Geral

O projeto é subdividido em 3 blocos, firmware, software e hardware, todas tiveram avanços em relação a proposta do projeto. As próximas seções terão foco no desenvolvimento individual de cada bloco com mais detalhes. Em relação ao desenvolvimento geral do projeto tem-se uma pequena mudança na estrutura do software, especificamente no servidor, tal mudança se refere a exclusão do processamento de machine learning, tendo em seu lugar um visualizador de dados, o qual permitiria a observação das medidas coletados pelos sensores.

1 Software

O software é composto por dois blocos, a visualização dos dados e o servidor. Sendo a conexão entre o bloco embarcado e o "mundo de fora", bem como o responsável por armazenar os dados coletados, o servidor é a parte fundamental do software, ele é feito na linguagem de programação Rust, linguagem que vem ganhando espaço no mercado web e de embarcados por sua segurança e performance. O código em si, utiliza das "crates" axum, tokio e serde que permitem a criação de um servidor web assíncrono, de alta performance. Além disso, o servidor utiliza a "crate" sqlx para a integração com o banco de dados, o qual é criado pela "engine" Sqlite, escrita em C e amplamente utilizada em ambientes mais restritos como em um celular, tablet, etc.

main.rs

```rust
use anyhow::Result;
use axum::Router;
use std::net::SocketAddr;

mod handlers;
mod router;

#[tokio::main]
async fn main() -> Result<()> {
    let routes_all = Router::new().merge(router::routes());

    let addr = SocketAddr::from(([127, 0, 0, 1], 42068));

    axum::Server::bind(&addr)
        .serve(routes_all.into_make_service())
        .await?;
    return Ok(());
}
```

Figura 1: Arquivo fonte "main" do servidor

Com o servidor estando ligado é preciso definir como ele cuidará das requisões enviadas para ele, as quais, nesse caso, seriam de envio de dados provenientes do esp32 e requisição de dados tanto pelo esp32 como por requisições externas. Um esboço das funções que cuidam dessa função está

presente na figura 2. Nessa figura pode-se ver o protótipo de três dessas funções que manipulam as requisições "get_complete_data_handler", "post_measuring_handler" e "get_statistics_handler", bem como três structs que representam as struturas das mensagens, que teriam formato Json.

Figura 2: Arquivo fonte que processa as requisições

Quanto ao visualizador de dados, ele é um script feito com python para se emular uma requisição ao servidor e então utilizar os dados recebidos via http para se traçar gráficos do comportamento das medidas de cada sensor. O andamento do script pode ser visto na figura 3.

Figura 3: Arquivo fonte que processa as requisições

2 Firmware

O Firmware do projeto será feito integralmente no framework esp-idf da espressif, empresa responsável pelo esp32, usando C++ como linguagem de programação. Esse bloco do projeto será responsável por coordenar a leitura dos sensores, o envio para o servidor e a visualização por meio

de um display. Nessa parte serão desenvolvidos drivers para o controle dos sensores e a interface da comunicação sem fio.

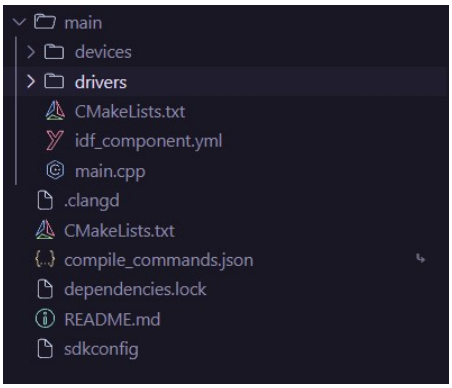

Figura 4: Estrutura de arquivos do Firmware

3 Hardware

O Hardware é bastante simples, sendo composto por um esp32, dois sensores de temperatura, um lm35 e um dht11, que funciona também como sensor de humidade, além disso há também um sensor de irradiância ultravioleta, gyml8511, e um sensor de qualidade do ar, cjmcu-811. Para visualização dos dados "localmente" tem-se um display.

Todos os dispositivos, display e sensores, serão gerenciados pelo esp32 levando em conta seus tempos de amostragem, formas de comunicação, etc. Infelizmente ainda não fiz o esquemático e realizei os testes físicos, pois nem todos os sensores foram entregues. No entanto, as conexões serão relativamente simples, com pouca ou nenhuma adição além de resistores de pull-up e semelhantes, isso se deve a inclinação da complexidade estar no software e firmware que manejariam a sincronia e a comunicação por código invés de por hardware.