MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
(Real Academia de Artilharia, Fortificação e Desenho - 1792)

Relatório Técnico: Monitoramento Inteligente de Carga com Sensor de Temperatura Analógica

Professor: 1º Ten Nicolas Oliveira

GRUPO:
Al Mikhael Pereira **Silveira**
Al **Marcos Vinícius** Rodrigues Silva
Al Rafael Santos **Sodré**

Resumo

Este relatório detalha o projeto e a implementação de um sistema de monitoramento de temperatura em tempo real, um componente crítico do sistema de monitoramento inteligente de carga. O sistema é composto por dois componentes principais: um software embarcado (C++), responsável pela leitura de um sensor de temperatura analógico num microcontrolador STM32MP1 DK1 para monitorar falhas ambientais que possam comprometer a integridade de uma carga sensível, e uma interface gráfica (Python/PyQt6), que atua como um dashboard centralizado. A comunicação entre os dois sistemas é realizada através do protocolo UDP, com dados serializados em formato JSON. O projeto cumpre todos os requisitos obrigatórios e bônus, incluindo alertas visuais, histórico gráfico e exportação de relatórios.

Conteúdo

1 Introdução

Este projeto visa solucionar um desafio logístico crítico: o monitoramento de uma carga sensível em trânsito, onde há risco de violação ou falha ambiental.

O objetivo da missão é que diferentes grupos desenvolvam módulos de sensores discretos que, ao final, serão integrados num único sistema de monitoramento centralizado. Este relatório foca no desenvolvimento do Módulo de Temperatura, cuja falha indica um comprometimento da integridade da carga.

A arquitetura segue o seguinte modelo:

- **O Cliente Embarcado (STM32):** O kit STM32MP1 DK1 lê o sensor de temperatura. O software (C++) formata a leitura num pacote JSON e o envia pela rede local via UDP.
- **O Monitor de Sensor UDP:** Uma aplicação robusta (Python/PyQt6) atua como o servidor central de monitoramento. Esta interface recebe os pacotes UDP do módulo, analisa o JSON e apresenta os dados ao operador de forma intuitiva, com gráficos, tabelas e alertas.

2 Descrição dos Componentes

2.1 Hardware: Sensor e Microcontrolador

Sensor:

- **Tipo**: NTC KY-013, um módulo sensor de temperatura baseado no sensor do tipo NTC de 10K;
- **Funcionamento**: O sensor monitora a condição ambiental da carga. Ele apresenta variação de resistência ôhmica em relação a temperatura submetida, isto é, quando aumenta a temperatura a resistência dimunui e por conseguinte a tensão enviada aumenta.

Microcontrolador:

- **Tipo**: Kit de desenvolvimento STM32MP1 DK1;
- **Funcionamento**: O kit juntamente com as aplicações obtém o valor analógico do sensor em forma de tensão, converte o valor lido para Graus Celsius e utiliza a classe `udp_client` para enviar os dados pela rede.

2.2 Software Embarcado (C++)

O software embarcado que está presente no repositório `leitura_sensor_temperatura` é o agente no local e contém os seguintes códigos:

- `embarcado/*.h`: Funções auxiliares para os arquivos .cpp;
- `embarcado/src/main.cpp` : O loop principal que lê o sensor *getTemperature()* e monta a estrutura de dados (pacote JSON);

- `embarcado/src/sensor.cpp` : Módulo que fornece a lógica para ler valores brutos de um sensor analógico e os converte para temperatura em ºC;
- `embarcado/src/main_embarcado.cpp` : Módulo que define a função *getTemperature()*, utilizada pelo main principal;
- `embarcado/src/udp_client.cpp` : Classe que abstrai a complexidade dos sockets C++ para envio de pacotes UDP;
- `embarcado/src/data_formatter.cpp` : Módulo que define a função responsável por preparar pacotes de dados de sensores no formato *UdpPacket*, incluindo a geração automática de timestamp e a associação de metadados como grupo, sensor, valor e unidade.

2.3 Software da Interface (Python)

O dashboard presente no repositório `interface_grafica_sensor_temperatura_analogica` funciona como a estação central de monitoramento e contém os seguintes scripts:

- `src/udp_listener.py` : A classe `UDPListener` corre em segundo plano, escutando a rede por pacotes UDP;
- `src/main_window.py` : A classe `MainWindow` é o "cérebro"da UI. Ela recebe o sinal do listener, e no slot `update_data` atualiza todos os componentes visuais, verifica os limites de alerta (configuráveis dinamicamente) e gere as funções de log;
- `main.py` Este script inicializa a QApplication do PyQt6, cria a instância da `MainWindow` (a interface gráfica principal) e inicia o loop de eventos da aplicação.

3 Protocolo de Comunicação

A comunicação entre o STM32 e a estação central é feita via UDP/IP.

- **Protocolo de Transporte UDP (User Datagram Protocol):** Essencial para a tarefa. A baixa latência é crítica para que alertas de violação ou falha ambiental sejam recebidos imediatamente na estação central. A perda de um único pacote de temperatura é aceitável em troca da velocidade em tempo real.

- **Protocolo de Dados JSON:** Permite que o sistema centralizado receba dados do sensor como grupo, sensor, valor e unidade.

- **Formato do pacote (Payload):** O Módulo de Temperatura envia o seguinte JSON:

```
{
"group": "grupo6",
"sensor_id": "SensorDeTemperatura",
"value": 23.5,
"unit": "ºC",
"ts": "2025-11-09T21:38:26Z"
```

4 Diagramas do Sistema

4.1 Fluxograma do Sistema

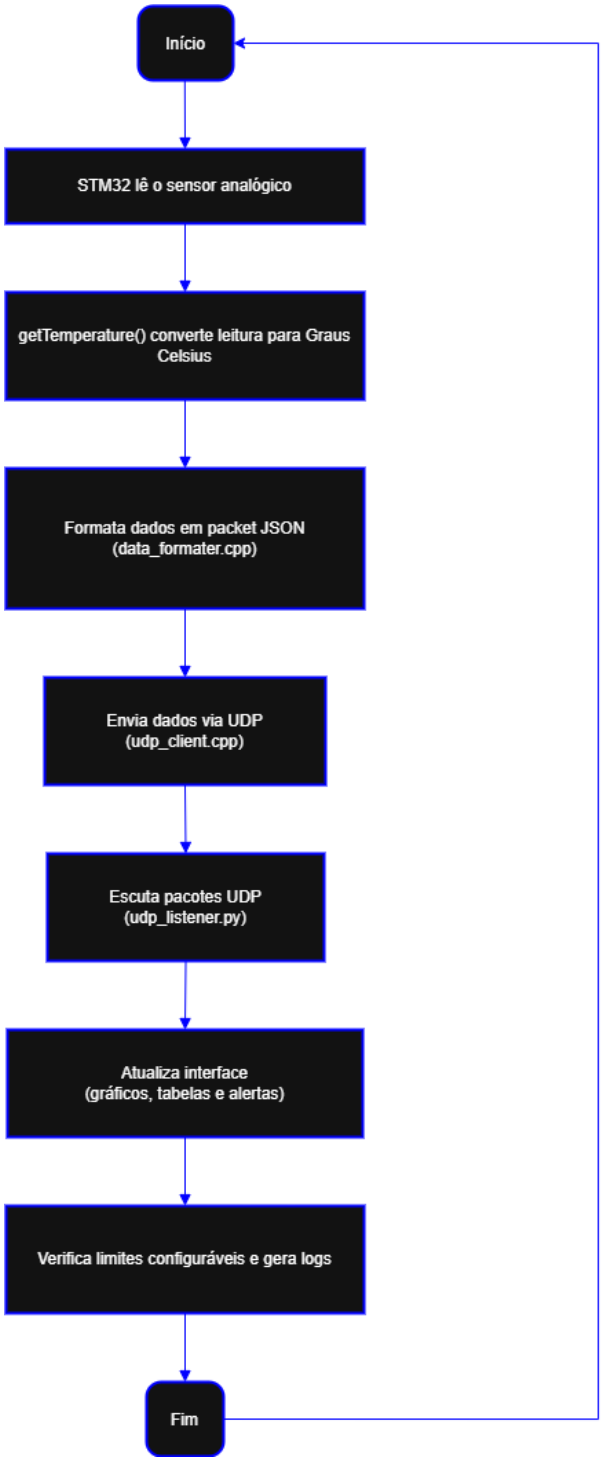

Figura 1: Fluxograma do Sistema

4.2 Diagrama de Classes (Software Embarcado C++)

O software embarcado (repositório `leitura_sensor_temperatura`) foi modularizado para encapsular a lógica de comunicação de rede, facilitando a reutilização e a integração.

A classe central desta arquitetura é a `UDPClient`, responsável por toda a abstração do socket de rede.

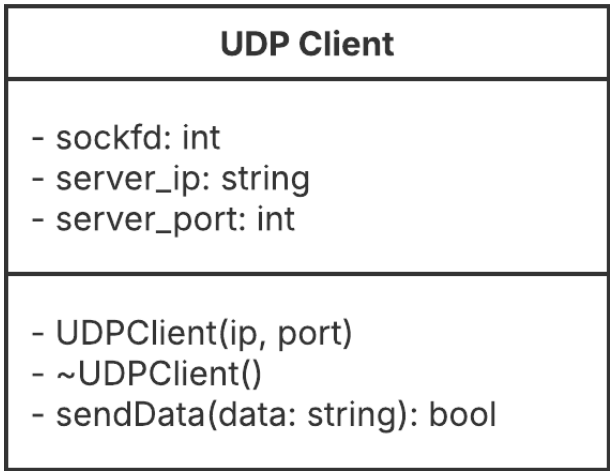

Figura 2: Diagrama UDPClient

O ficheiro `main.cpp` (loop principal) instancia 1 (um) objeto `UDPClient`. O main chama periodicamente a função `getTemperature()` (leitura do sensor) e, em seguida, utiliza o método `sendData()` do objeto `UDPClient` para transmitir o payload JSON.

4.3 Diagrama de Classes (Python UI)

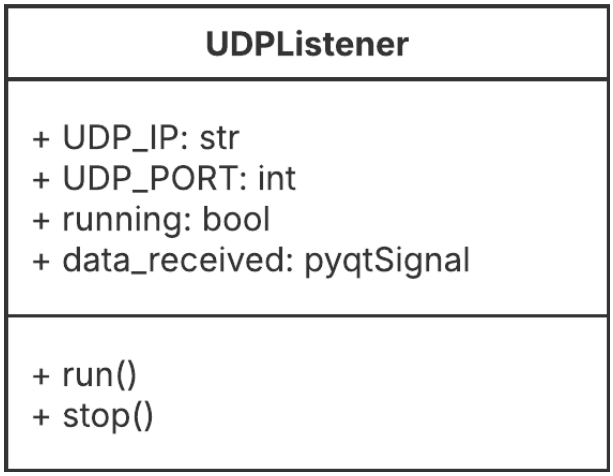

Figura 3: Diagrama UDPListener

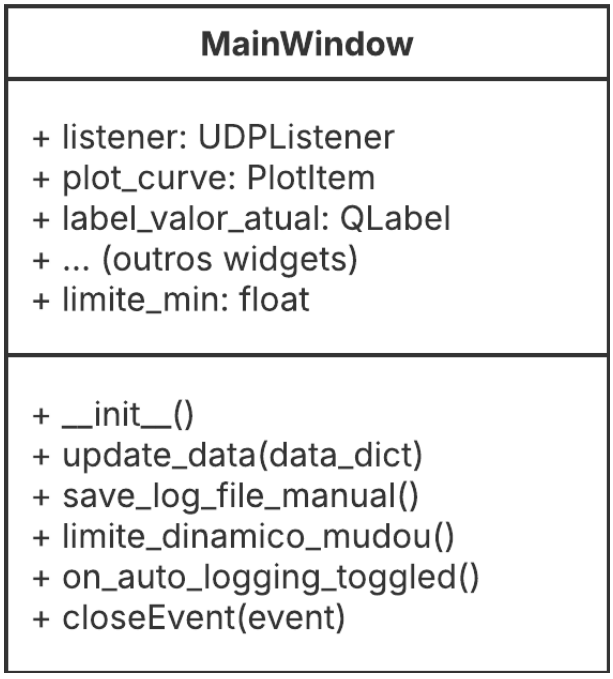

Figura 4: Diagrama MainWindow

5 Instruções de Compilação e Uso

5.1 Software Embarcado (C++) (Ambiente Linux)

As instruções abaixo assumem um ambiente de desenvolvimento Linux (nativo ou via VirtualBox) para a compilação cruzada, e SSH / TeraTerm para comunicação com a placa.

Clone o repositório

```
git clone https://github.com/MkDev21IA/
   leitura_sensor_temperatura.git
cd leitura_sensor_temperatura/embarcado
```

Compilação cruzada (Altere o PATH)

```
export CXX = "/ caminho / para / seu / toolchain / bin"
source ~/.bashrc

# Compilando:
$CXX ./src/*.cpp -I ./include -o ../build/sensor_temperatura
```

Acesso ao Console (SSH ou Serial):

1. **SSH:**

Via SSH

```
ssh root@<IP_DA_STM32>
```

2. **Serial:** Conecte o cabo USB (ST-Link) ao PC. Abra o TeraTerm, selecione a porta COM virtual correspondente ao ST-Link e configure o baud rate para 115200.

Transferência para a Placa

```
# Sintaxe: scp -O <ficheiro_local> <user>@<ip_da_placa>:<
    caminho_na_placa>
scp -O ./seu_executavel root@<IP_DA_STM32>:/home/root/
```

Execução na Placa

```
chmod +x /home/root/seu_executavel
/home/root/seu_executavel &
```

5.2 Interface Gráfica (Python)

Clone o repositório

```
git clone https://github.com/MkDev21IA/
    interface_grafica_sensor_temperatura_analogica.git
cd interface_grafica_sensor_temperatura_analogica
```

Crie e ative um ambiente virtual

```
python -m venv .venv
.\.venv\Scripts\Activate-ps1
```

Instale as dependências

```
pip install -r requirements.txt
```

Configure a rede. Abra o config.ini e garanta que a porta `UDP_PORT` é a mesma do C++ e que o IP `UDP_IP` é 0.0.0.0 (recomendado para escutar).

Execute a interface

```
python main.py
```

6 Resultados e Análise

6.1 Capturas de Tela da Interface

As capturas abaixo demonstram o monitor de sensor em operação.

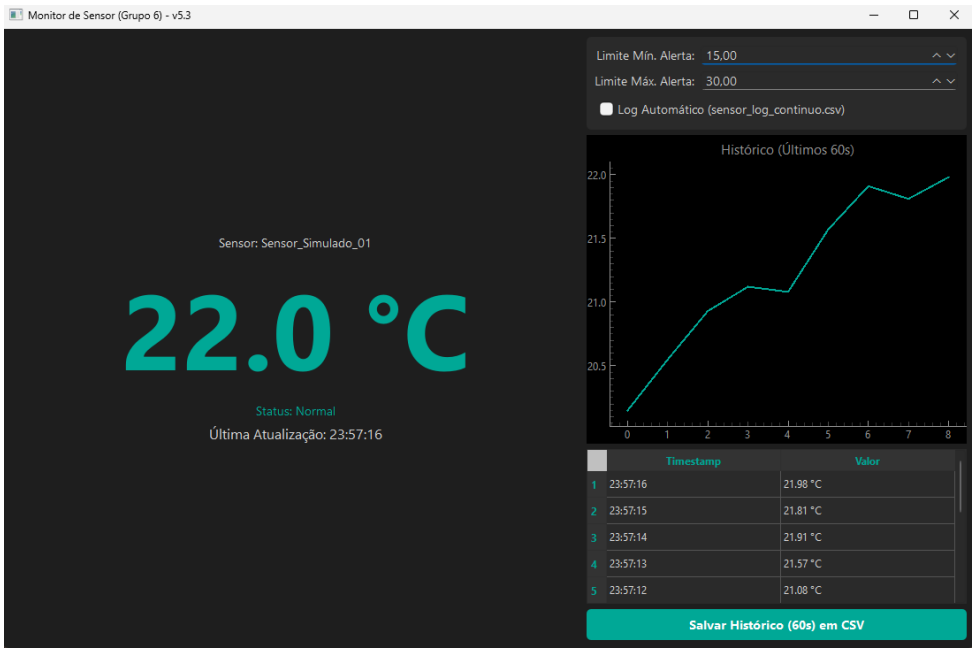

Figura 5: Interface monitorando uma carga em condição "Normal". A temperatura (ex: 22°C) está dentro dos limites de segurança, indicada pela cor verde. O gráfico e a tabela de histórico (com timestamps) confirmam a estabilidade.

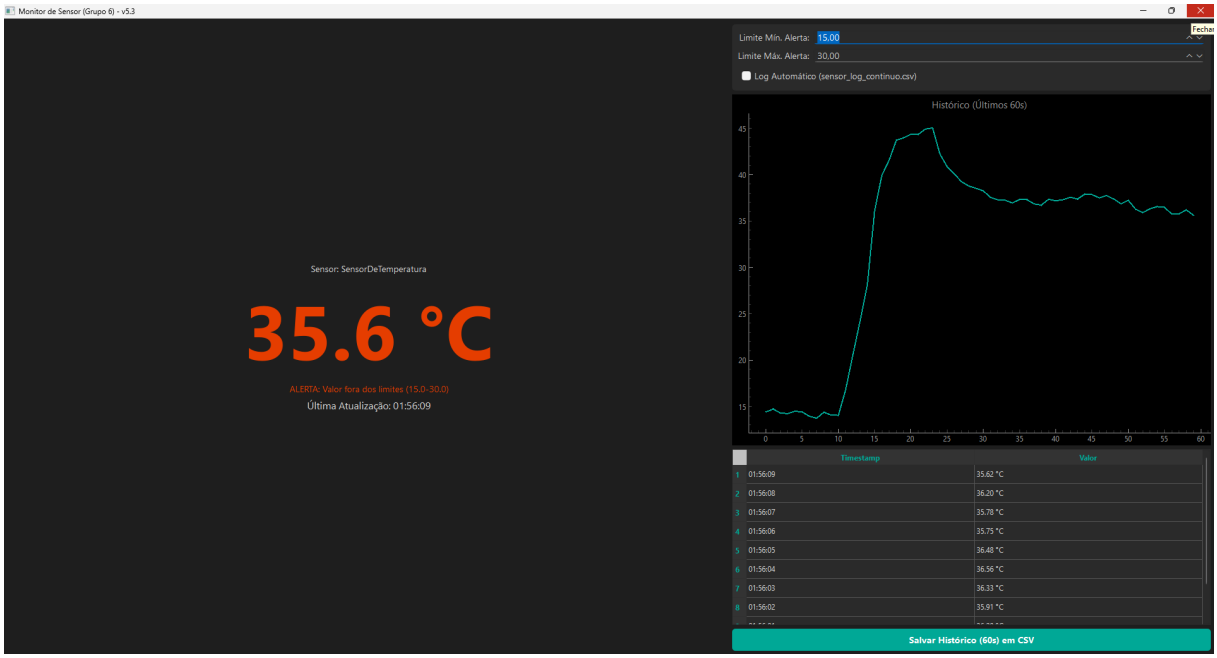

Figura 6: Simulação de uma falha ambiental. A temperatura (ex: 35.6°C) ultrapassou o limite máximo (definido dinamicamente em 30.0°C), indicando que a integridade da carga foi comprometida. O sistema responde imediatamente com um alerta visual vermelho.

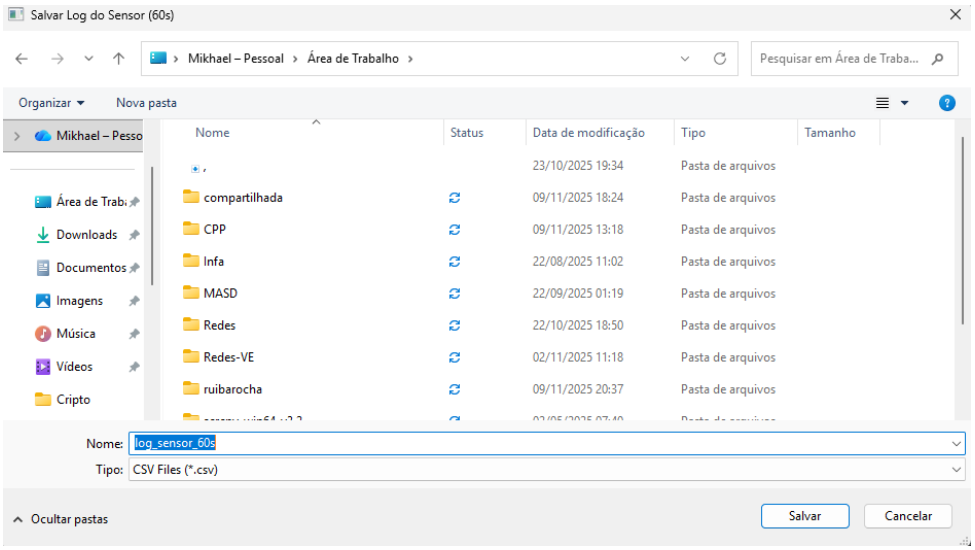

Figura 7: Demonstração da função de exportação de relatório. O operador pode salvar os últimos 60 segundos de dados para análise posterior à missão, ou ativar o log automático para um registro contínuo.

6.2 Análise dos Valores Medidos

A análise dos valores medidos foca na deteção de anomalias que indiquem comprometimento da carga.

- **Estado Estável:** O histórico gráfico permitiu observar o sensor reportando uma temperatura estável quando não estava em contato com objetos quentes ou frios, ou seja, na temperatura ambiente. Isto representa a linha de base da carga em condição segura.

- **Estado de Alerta:** A funcionalidade de "Alerta Visual"(Figura 6) foi validada para simular uma falha ambiental, aparecendo ao entrar em contato com um isqueiro, por exemplo. A interface respondeu instantaneamente quando o valor ultrapassou o limite configurado (30.0°C), provando a eficácia do sistema em notificar o operador sobre um comprometimento da carga.

- **Logs (Relatórios de Missão):** A capacidade de exportar relatórios em CSV (automática ou manual) permite a criação de um registro auditável, essencial para análises posteriores.

7 Documentação do Código

Para garantir a manutenibilidade, a integração e a reutilização futura dos módulos, ambos os códigos-fonte (C++ e Python) foram extensivamente documentados utilizando o formato Doxygen.

A documentação gerada automaticamente detalha todas as classes, métodos, parâmetros e ficheiros de ambos os projetos, criando um "manual do desenvolvedor"completo.

7.1 Documentação do Software Embarcado (C++)

A documentação do código C++, que detalha a classe `UDPClient` e o loop principal, está incluída no repositório `leitura_sensor_temperatura`.

* Como acessar: Abra o ficheiro `html/index.html` dentro do repositório do projeto C++.

7.2 Documentação da Interface Gráfica (Python)

A documentação da interface, detalhando as classes `MainWindow` e `UDPListener`, está incluída no repositório `interface_grafica_sensor_temperatura_analogica`.

* Como acessar: Abra o ficheiro `html/index.html` dentro do repositório do projeto Python.

8 Conclusão

Este projeto demonstrou com sucesso o desenvolvimento de um componente vital para o sistema de Monitoramento Inteligente de Carga. O Módulo de Temperatura (C++/STM32) e o Monitor visual (Python/PyQt6) comunicam de forma robusta, rápida e confiável.

Todos os requisitos obrigatórios e bônus da missão foram implementados. O sistema está funcional e pronto para utilização.