Trabalho Final – Sistema de Monitoramento via LoRa

Relatório Técnico – Entregável 1: Arquitetura e Protótipo Lógico

Disciplina: CIC0236 - TELEINFORMÁTICA E REDES 2

Título do Projeto: Monitoramento de Condições Ambientais em Sala de Equipamentos com LoRa

Grupo:

Weldo Gonçalves da Silva Junior - 222014133

HERCULES NATAN DANTAS DE ALMEIDA MEDEIROS - *222009760*

Data: 20 de Outubro de 2025

1. Introdução

A estabilidade das condições ambientais em salas de equipamentos, laboratórios e servidor, é crucial para garantir a integridade e a longevidade dos dispositivos eletrônicos. Variações de temperatura, umidade ou excesso de partículas podem levar a falhas, resultando em perdas de dados e custos elevados de manutenção. Sistemas de monitoramento contínuo são, portanto, essenciais para a detecção precoce de anomalias.

Este projeto propõe o desenvolvimento de um sistema de monitoramento distribuído, de baixo custo e alta eficiência energética, utilizando a tecnologia de comunicação LoRa (Long Range). O objetivo é coletar dados ambientais (temperatura, umidade, etc.) de múltiplos nós sensores localizados em pontos estratégicos e centralizá-los em um servidor para visualização e análise em tempo real.

2. Arquitetura do Sistema

A arquitetura foi projetada de forma modular para desacoplar as responsabilidades e facilitar a manutenção e escalabilidade futura. O sistema é composto por três componentes principais: Nós Sensores (Clientes LoRa), Gateway (LoRa → IP) e Servidor Central com Dashboard.

2.1. Diagrama da Arquitetura

O diagrama abaixo ilustra a interação entre os componentes e as camadas de comunicação envolvidas.

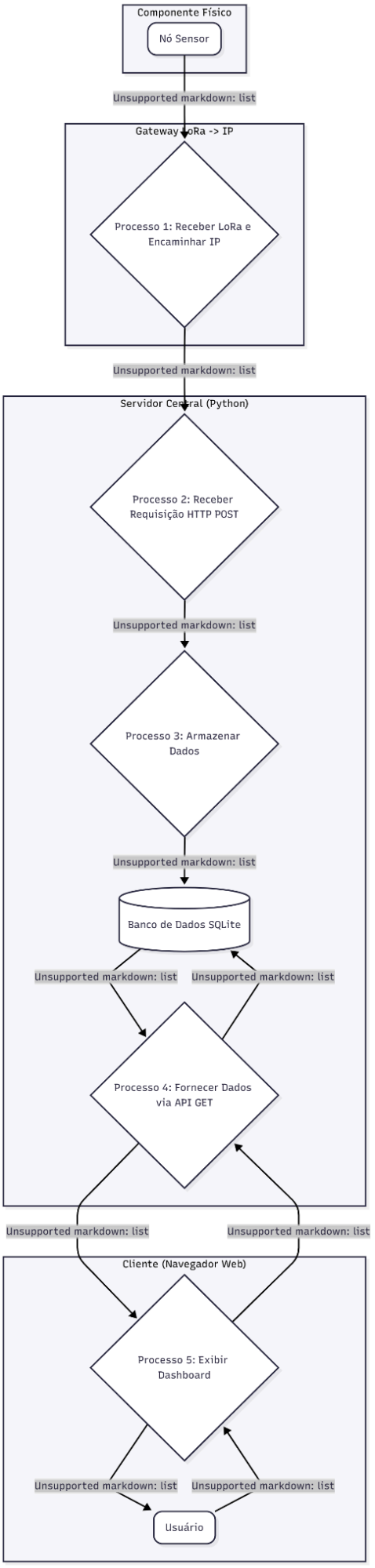
Componente Físico
Nó Sensor
Unsupported markdown: list
Gateway LoRa -> IP
Processo 1: Receber LoRa e Encaminhar IP
Unsupported markdown: list
Servidor Central (Python)
Processo 2: Receber Requisição HTTP POST
Unsupported markdown: list
Processo 3: Armazenar Dados
Unsupported markdown: list
Banco de Dados SQLite
Unsupported markdown: list
Unsupported markdown: list
Processo 4: Fornecer Dados via API GET
Unsupported markdown: list
Unsupported markdown: list
Cliente (Navegador Web)
Processo 5: Exibir Dashboard
Unsupported markdown: list
Unsupported markdown: list
Usuário

Figura 1: Diagrama de Fluxo de Dados do Sistema.

Fluxo Principal:

1. **Nós Sensores:** Coletam dados ambientais e os encapsulam em um payload.
2. **Comunicação LoRa:** Os dados são transmitidos via rádio (LoRa) para o Gateway.
3. **Gateway:** Recebe o pacote LoRa, extrai o payload e o retransmite pela rede IP local.
4. **Comunicação IP:** O Gateway envia os dados ao Servidor Central via uma requisição HTTP POST.
5. **Servidor Central:** Processa a requisição, armazena os dados em um banco de dados e os disponibiliza para o dashboard.
6. **Dashboard:** O usuário acessa uma página web que exibe os dados mais recentes de forma organizada.

2.2. Descrição dos Módulos

a) Nós Sensores (Clientes LoRa)

- **Responsabilidade:** Coletar dados do ambiente físico (ex: temperatura e umidade) em intervalos pré-definidos.
- **Implementação Lógica (Protótipo):** Para este entregável, os nós são simulados por um script Python (**`simulador.py`**). Este script utiliza a biblioteca `random` para gerar valores realistas de sensores e a biblioteca `urllib.request` para enviar esses dados diretamente ao servidor, simulando a ação de um gateway.
- **Camadas Envolvidas:** Aplicação (coleta de dados), Enlace/Física (transmissão LoRa).

b) Gateway (LoRa → IP)

- **Responsabilidade:** Atuar como uma ponte (bridge) entre a rede LoRa e a rede IP. Ele recebe pacotes de múltiplos nós sensores, decodifica os dados e os encaminha para um único ponto na rede IP (o Servidor Central).
- **Implementação Lógica (Protótipo):** A função do gateway é encapsulada no script de simulação. Ele é responsável por formatar os dados em JSON e realizar a requisição HTTP POST para o endpoint do servidor.
- **Camadas Envolvidas:** Enlace/Física (recepção LoRa), Rede/Transporte (encapsulamento em pacotes IP/TCP), Aplicação (requisição HTTP).

c) Servidor Central e Dashboard

- **Responsabilidade:** Receber, processar, armazenar e visualizar os dados.
- **Implementação Lógica (Protótipo):** Desenvolvido em Python, utiliza as seguintes bibliotecas padrão:
    - **`http.server`**: Para criar um servidor HTTP simples que responde a requisições GET (para o dashboard) e POST (para receber dados).
    - **`threading`**: Permite que o servidor trate múltiplas requisições simultaneamente sem bloqueio.

    - **`json`**: Para decodificar o corpo das requisições POST, que contém os dados dos sensores.
    - **`sqlite3`**: Para criar e gerenciar um banco de dados leve que armazena todos os dados recebidos, garantindo a persistência.
    - **`time`**: Para registrar o timestamp de cada medição recebida.
- **Camadas Envolvidas:** Aplicação (lógica de negócio e dashboard), Transporte/Rede (escuta em uma porta TCP/IP).

3. Fluxo de Dados e Formato da Mensagem

A padronização do formato de mensagem é fundamental para garantir a interoperabilidade entre os componentes do sistema.

3.1. Formato da Mensagem (Payload)

Foi escolhido o formato JSON (JavaScript Object Notation) para a comunicação entre o Gateway e o Servidor Central, devido à sua simplicidade, legibilidade e amplo suporte em diversas linguagens.

A estrutura da mensagem é a seguinte:

```
{
  "node_id": "sala_servidores_01",
  "timestamp": "2025-10-20T22:15:00Z",
  "data": {
    "temperatura": 24.5,
    "umidade": 45.8
  }
}
```

- **`node_id`** (string): Identificador único do nó sensor que originou a medição.
- **`timestamp`** (string): Data e hora da coleta no formato ISO 8601.
- **`data`** (object): Um objeto contendo os pares chave-valor das medições realizadas (ex: "temperatura", "umidade", "nivel_poeira").

3.2. Especificação do Fluxo de Comunicação

1. **Geração e Envio (Simulador):**
    - O script **`simulador.py`** gera um dicionário Python com a estrutura definida acima.
    - Os valores de **`temperatura`** e **`umidade`** são gerados aleatoriamente dentro de uma faixa pré-definida.
    - O dicionário é convertido para uma string JSON.

    - Utilizando **`urllib.request`**, uma requisição HTTP POST é enviada para o endpoint **`/data`** do servidor. O cabeçalho **`Content-Type`** é definido como **`application/json`**.
2. **Recepção e Processamento (Servidor):**
    - O servidor, implementado com **`http.server`**, recebe a requisição POST no endpoint **`/data`**.
    - O corpo (body) da requisição é lido e decodificado de JSON para um objeto Python.
    - Os dados extraídos (ID do nó, timestamp, temperatura, umidade) são inseridos em uma tabela no banco de dados SQLite.
    - O servidor responde com um status HTTP **`200 OK`** para confirmar o recebimento.
3. **Visualização (Dashboard):**
    - O usuário acessa o endereço do servidor (ex: **`http://localhost:8000`**) em um navegador.
    - O servidor responde à requisição GET servindo um arquivo HTML estático (**`index.html`**).
    - Este arquivo HTML contém código JavaScript que, periodicamente (a cada 5 segundos), faz uma requisição GET a um endpoint **`/api/latest_data`**.
    - O servidor, ao receber a requisição neste endpoint, consulta o banco de dados para obter a leitura mais recente de cada **`node_id`**, formata o resultado como JSON e o envia como resposta.
    - O JavaScript no lado do cliente atualiza a página dinamicamente com os novos dados, sem a necessidade de recarregar a página inteira.