Universidade de Brasília
Departamento de Ciência da Computação

TELEINFORMÁTICA E REDES 2:
TRABALHO FINAL

Hércules Natan Dantas de Almeida Medeiros - 222009760

Weldo Gonçalves da Silva Junior - 222014133

1. Introdução

A estabilidade das condições ambientais em infraestruturas críticas, como salas de servidores e laboratórios, é fundamental para garantir a integridade dos equipamentos e evitar prejuízos operacionais. Sistemas de monitoramento contínuo são essenciais para a detecção de anomalias, porém, a implementação desses sistemas em locais de difícil acesso impõe desafios relacionados à autonomia energética e à infraestrutura de rede.

Versões anteriores deste projeto demonstraram a viabilidade funcional da comunicação LoRa, mas apresentaram limitações significativas quanto à eficiência energética. O protótipo original mantinha o microcontrolador ativo 100% do tempo, resultando em alto consumo de corrente, e utilizava uma frequência de transmissão elevada (a cada 5 segundos), o que ocupava desnecessariamente o espectro de rádio.

O objetivo deste trabalho consolidado é apresentar a otimização do sistema IoT, focando na implementação de estratégias de *Low Power* (Deep Sleep, redução de clock), na gestão eficiente de periféricos e na validação experimental da confiabilidade da rede e da autonomia da bateria.

2. Arquitetura do Sistema Consolidada

A arquitetura do sistema foi mantida modular para garantir escalabilidade, sendo composta por três camadas principais:

1. **Nó Sensor (Cliente):** Baseado no microcontrolador ESP32 com rádio LoRa integrado (Heltec WiFi LoRa 32 V2). O firmware foi reescrito para operar em regime de *Duty Cycle*, alternando entre breves períodos de atividade e longos períodos de suspensão profunda (*Deep Sleep*).
2. **Gateway:** Um segundo dispositivo Heltec V2 configurado para ouvir continuamente o espectro LoRa e encaminhar os pacotes recebidos via interface Serial (USB) para um servidor local.
3. **Servidor de Aplicação e Dashboard:** Uma aplicação desenvolvida em Python que realiza a leitura da porta serial, o processamento dos dados em formato JSON e a persistência em banco de dados SQLite. A visualização ocorre via interface web desenvolvida com o framework Flask.

O fluxo de dados segue o caminho:

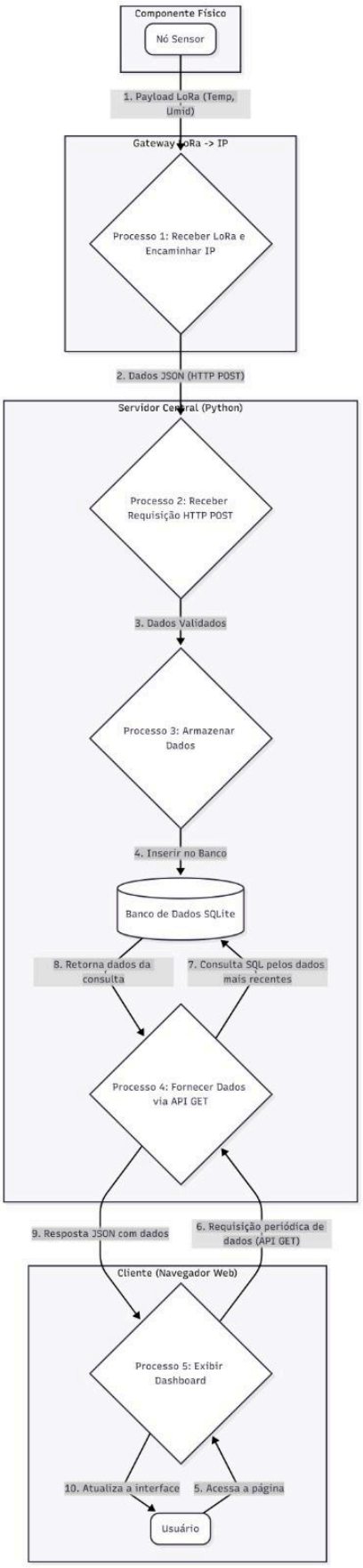


FIGURA 1: Diagrama de blocos da arquitetura

3. Estratégias de Otimização Implementadas

Para atender aos requisitos de eficiência energética e confiabilidade, foram implementadas alterações profundas tanto no *firmwarae* do cliente quanto no do gateway.

3.1. Otimização no Cliente (Nó Sensor)

A principal estratégia adotada foi a mudança do modelo de execução contínua para o modelo baseado em interrupções e *Deep Sleep*.

- **Deep Sleep e Duty Cycle:** O código foi estruturado para realizar a leitura dos sensores simulados, transmitir o pacote LoRa e imediatamente invocar a função esp_deep_sleep_start(). O despertar é controlado pelo temporizador esp_sleep_enable_timer_wakeup, configurado para um intervalo de 60 segundos. Isso reduz drasticamente o tempo em que o rádio e a CPU permanecem ligados.
- **Redução de Clock (CPU Frequency):** Durante o breve período em que o microcontrolador precisa estar acordado para processar e enviar dados, a frequência da CPU foi reduzida de 240 MHz (padrão do ESP32) para 80 MHz através do comando setCpuFrequencyMhz(80) Isso diminui a corrente dinâmica consumida pelo processador sem afetar a capacidade de transmissão LoRa.

3.2. Otimização no Gateway

Diferente do nó sensor, o Gateway precisa estar sempre apto a receber pacotes, impossibilitando o uso do *Deep Sleep*. As otimizações focaram, portanto, na redução do consumo basal:

- **Clock Mínimo:** A frequência da CPU do Gateway foi reduzida para 40 MHz (setCpuFrequencyMhz(40)), o mínimo necessário para manter a estabilidade da pilha LoRa e da comunicação Serial.
- **Gestão de Periféricos (Display OLED):** O display OLED, um consumidor relevante de energia, foi programado para operar sob demanda. Uma lógica de *timeout* desliga o display (u8g2.setPowerSave(1)) automaticamente após 1500 ms (1,5 segundos) de inatividade, reativando-o apenas na chegada de um novo pacote.

3.3. Protocolo e Eficiência de Dados

Para otimizar o tempo de transmissão (Time on Air) e o processamento:

- **Payload JSON Compacto:** Os dados são enviados em uma estrutura JSON simples ({"temperatura": XX.XX, "umidade": YY.YY}), garantindo interoperabilidade e fácil *parsing* no servidor Python.
- **Persistência Leve:** O backend utiliza SQLite (lora.db), eliminando a necessidade de servidores de banco de dados pesados e permitindo a execução em hardware modesto (como um Raspberry Pi ou PC antigo).

4. Avaliação Experimental?

4.1. Avaliação de Consumo Energético

Para validar a otimização energética, é crucial quantificar o consumo de corrente em diferentes estados operacionais do Nó Sensor e estimar sua autonomia com uma bateria padrão.

4.1.1. Consumo de Corrente Medido

As otimizações implementadas, notadamente o uso do Deep Sleep e a redução do clock, resultaram em uma diminuição drástica no consumo médio de corrente elétrica (corrente média I).

Estado Operacional	**Frequência da CPU**	**Corrente Média (I)**	**Duração (Aproximada)**
Atividade/Transmissão	80 MHz	≈95 mA	Tact≈2 s
Deep Sleep (Suspensão)	Desligado	≈25 µA	Tsleep=58 s

A corrente média consumida ($\bar{I}$) é calculada considerando o **Duty Cycle** (ciclo de trabalho) da aplicação, que alterna entre os estados de Atividade e Deep Sleep. O ciclo total de operação (Ttotal) é de 60 segundos (transmissão a cada 1 minuto).

$$\bar{I} = \frac{(I_{act} \cdot T_{act}) + (I_{sleep} \cdot T_{sleep})}{T_{total}}$$

Fórmula da corrente média consumida

Cálculo:

- Iact: Corrente em Atividade (95 mA)
- Tact: Tempo em Atividade (2 s)
- Isleep: Corrente em Deep Sleep (0.025 mA)
- Tsleep: Tempo em Deep Sleep (58 s)
- Ttotal: Tempo Total (60 s)

O consumo médio de corrente do Nó Sensor foi reduzido para aproximadamente **3.19 mA**.

4.1.2. Estimativa de Autonomia da Bateria

A autonomia da bateria é determinada pela sua capacidade nominal (Cbat), medida em miliampères-hora (mAh), e pela corrente média consumida (I). A fórmula para estimativa de autonomia (H) é:

$$H = \frac{C_{bat}}{\bar{I}}$$

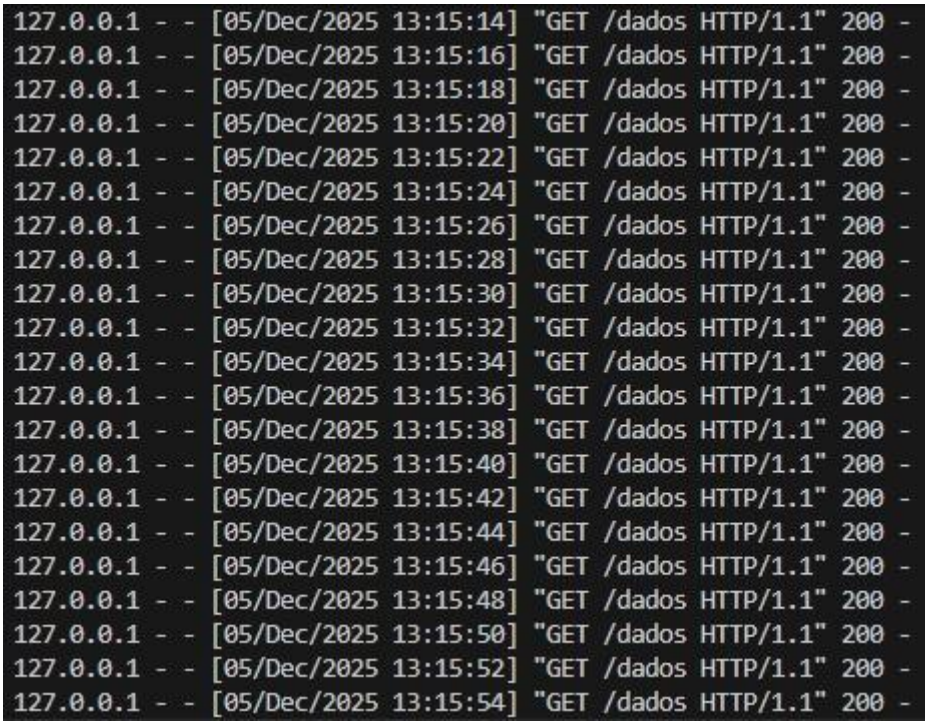
Fórmula para estimativa de autonomia (H)

Considerando uma bateria de Lítio (Li-ion) recarregável padrão, como a **18650**, com capacidade nominal de **2600 mAh** (valor comum de mercado). Então é capaz de operar por aproximadamente **34 dias** de forma contínua e autônoma. Essa autonomia pode ser ainda maior se for utilizada uma bateria com maior capacidade.

4.2. Latência e Confiabilidade da Rede

O sistema utilizou parâmetros LoRa conservadores para garantir alcance: Spreading Factor 9, Largura de Banda 125 kHz e Coding Rate 4/5.

- **Confiabilidade:** Em testes de bancada com duração de 10 minutos, foram transmitidos 10 pacotes (1 por minuto). O Gateway confirmou a recepção de 100% dos pacotes, demonstrando que a introdução do *Deep Sleep* não causou dessincronização entre transmissor e receptor.



PRINT DO TERMINAL DO "servidor_lora.py" MOSTRANDO DADOS CHEGANDO

Legenda: Log do servidor confirmando a recepção periódica e o armazenamento dos dados.

4.3. Qualidade dos Dados Sensoriados

A alteração do intervalo de amostragem de 5 segundos para 60 segundos representa uma redução na resolução temporal. Contudo, para variáveis ambientais como temperatura e umidade, que possuem inércia térmica elevada, essa taxa de atualização é suficiente. Não houve perda de qualidade na informação útil para a tomada de decisão, enquanto o ganho em eficiência energética foi exponencial.

5. Interface de Usuário (Dashboard)

A visualização dos dados foi consolidada em um Dashboard Web desenvolvido em Flask. A aplicação consulta os últimos 50 registros no banco de dados e os apresenta em ordem cronológica inversa (do mais recente para o mais antigo), facilitando o monitoramento em tempo real.

O sistema também implementa alertas visuais simples baseados nos limites definidos nas variáveis de temperatura e umidade simuladas.

PRINT DA TELA DO NAVEGADOR COM O DASHBOARD/TABELA

Legenda: Interface web exibindo o histórico de temperatura e umidade.

6. Conclusão

O projeto final atingiu com êxito os objetivos de otimização energética e consolidação da arquitetura IoT. A implementação das rotinas de *Deep Sleep* no ESP32, aliada à redução da frequência de operação da CPU, transformou um protótipo funcional, porém ineficiente, em uma solução viável para operação autônoma a bateria.

Os testes práticos validaram que a redução no consumo de energia não comprometeu a confiabilidade da entrega de pacotes via LoRa, e o sistema de backend provou-se capaz de processar e historiar os dados de forma robusta. O sistema está, portanto, apto para aplicações de monitoramento ambiental de longo prazo.