UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE
ENGENHARIA DE COMPUTAÇÃO

Dennis Paul Paz Lopez

Relatorio do trabalho final da disciplina
Redes sem Fio (DEC7563-07655):
Zephion App

Araranguá
2025

SUMÁRIO

1 DESCRIÇÃO DO PROJETO

Este projeto, desenvolvido como trabalho final da disciplina de Redes sem Fio, consistiu na criação de um sistema IoT (*Internet of Things*) para monitoramento ambiental, utilizando o microcontrolador ESP32 DevKitC V1 e uma aplicação web full-stack desenvolvida com o framework Django.

O principal objetivo foi projetar um sistema robusto e confiável, capaz de coletar periodicamente dados de sensores de temperatura, umidade e gás, transmitir essas informações para um servidor web, e disponibilizá-las em um painel de visualização acessível a usuários autenticados. A arquitetura do sistema foi cuidadosamente planejada para garantir a continuidade da comunicação, mesmo em condições adversas, por meio de um mecanismo automático de alternância entre Wi-Fi e BLE (*Bluetooth Low Energy*).

Esse sistema exemplifica uma aplicação prática de redes sem fio em ambientes IoT, demonstrando o potencial de integração entre dispositivos embarcados e plataformas web modernas.

1.1 Componentes de hardware

Os principais componentes de hardware utilizados foram:

- **ESP32 DevKitC V1:** Microcontrolador principal, equipado com conectividade nativa Wi-Fi e BLE, responsável pela aquisição dos dados, processamento local e transmissão das informações ao servidor.
- **Sensor DHT11:** Sensor digital econômico, utilizado para medições de temperatura e umidade relativa do ar. Conectado ao pino `GPIO21`, sua facilidade de integração com o ESP32 o torna ideal para aplicações básicas de monitoramento.
- **Sensor de Gás MQ-2:** Sensor analógico conectado ao pino `GPIO19`, utilizado para detecção de gases combustíveis e fumaça, permitindo a avaliação da qualidade do ar.
- **Display OLED SSD1306 0.96" (I2C):** Com resolução de 128x64 pixels, foi empregado para a exibição local dos dados dos sensores e do estado da conexão (Wi-Fi ou BLE). Utiliza o barramento I2C, nos pinos `GPIO22 (SCL)` e `GPIO23 (SDA)`.

1.2 Componentes de software

O projeto foi estruturado em duas frentes de desenvolvimento de software:

- **Firmware do ESP32:** Programado em C++ na plataforma Arduino IDE, o firmware implementa toda a lógica embarcada, utilizando as seguintes bibliotecas:

  - `WiFi.h`: Gerenciamento da conectividade Wi-Fi.
  - `HTTPClient.h`: Realização de requisições HTTP POST.
  - `BLEDevice.h`, `BLEServer.h`, `BLEUtils.h`, `BLE2902.h`: Configuração e operação do BLE.
  - `DHTesp.h`: Leitura dos valores de temperatura e umidade do DHT11.
  - `Adafruit_SSD1306.h` e `Adafruit_GFX.h`: Controle gráfico do display OLED.

Um dos destaques do firmware foi a implementação de um mecanismo automático de alternância entre Wi-Fi e BLE, assegurando a comunicação contínua com o servidor, conforme exemplificado abaixo:

Listing 1.1 – Alternância automática entre Wi-Fi e BLE

```
1 if (WiFi.status() == WL_CONNECTED && bleActive) {
2   stopBLE();
3 } else if (WiFi.status() != WL_CONNECTED && !bleActive) {
4   initBLE();
5 }
6
```

- **Aplicação Web (Django):** Responsável pelo backend, API e painel web, utilizando as seguintes bibliotecas e ferramentas:
  - `djangorestframework`: Criação da API REST para recebimento e armazenamento de dados.
  - `django-allauth`: Autenticação e gerenciamento de usuários.
  - `django-htmx` e `django-cotton`: Atualizações assíncronas e dinâmicas no frontend.
  - `django-redis` e `redis`: Cache e gerenciamento eficiente de sessões.
  - Bibliotecas complementares como `django-crispy-forms`, `django-filter` e `drf-spectacular`.

1.3 Funcionamento do projeto

O sistema opera conforme o seguinte fluxo:

1. **Inicialização:** O ESP32 inicia tentando estabelecer conexão com a rede Wi-Fi configurada. Em caso de falha, o modo BLE é ativado automaticamente.
2. **Coleta de dados:** Os sensores DHT11 e MQ-2 são lidos em intervalos regulares. Os dados são exibidos no display OLED, juntamente com o status da conexão ativa.
3. **Transmissão de dados:** Caso o Wi-Fi esteja disponível, os dados são enviados para o servidor via HTTP POST. Na ausência de Wi-Fi, a transmissão ocorre via BLE, utilizando o formato JSON.

4. **Armazenamento e exibição:** No servidor, os dados são processados, armazenados no banco de dados e apresentados no dashboard, que oferece visualização em tempo real e consulta de histórico.

1.4 Metodologia experimental

Os testes foram conduzidos em ambiente de laboratório, conforme a metodologia descrita abaixo:

1. Configuração do servidor Django local com exposição externa via túnel Ngrok, permitindo acesso remoto para testes.
2. Avaliação da transmissão via Wi-Fi, variando a distância entre o ESP32 e o roteador.
3. Simulação de falhas de conexão Wi-Fi para testar o comportamento de fallback para BLE.
4. Monitoramento do tempo de resposta das requisições HTTP e análise da estabilidade da comunicação BLE.
5. Verificação da integridade dos dados armazenados no banco de dados após as transmissões.

2 RESULTADOS E DISCUSSÃO

Os resultados experimentais demonstraram que o sistema atendeu aos requisitos de confiabilidade e desempenho:

- **Envio via Wi-Fi:** Estável até aproximadamente 12 metros de distância, com latência média de 200 ms.
- **Fallback BLE:** Ativação em até 2 segundos após a perda de Wi-Fi, com transmissão eficiente em distâncias de até 10 metros.
- **Dashboard Web:** Interface funcional, com atualização automática dos dados em tempo real por meio de HTMX.
- **Integridade dos Dados:** Não foram detectadas perdas de dados durante os testes, evidenciando a robustez do sistema.

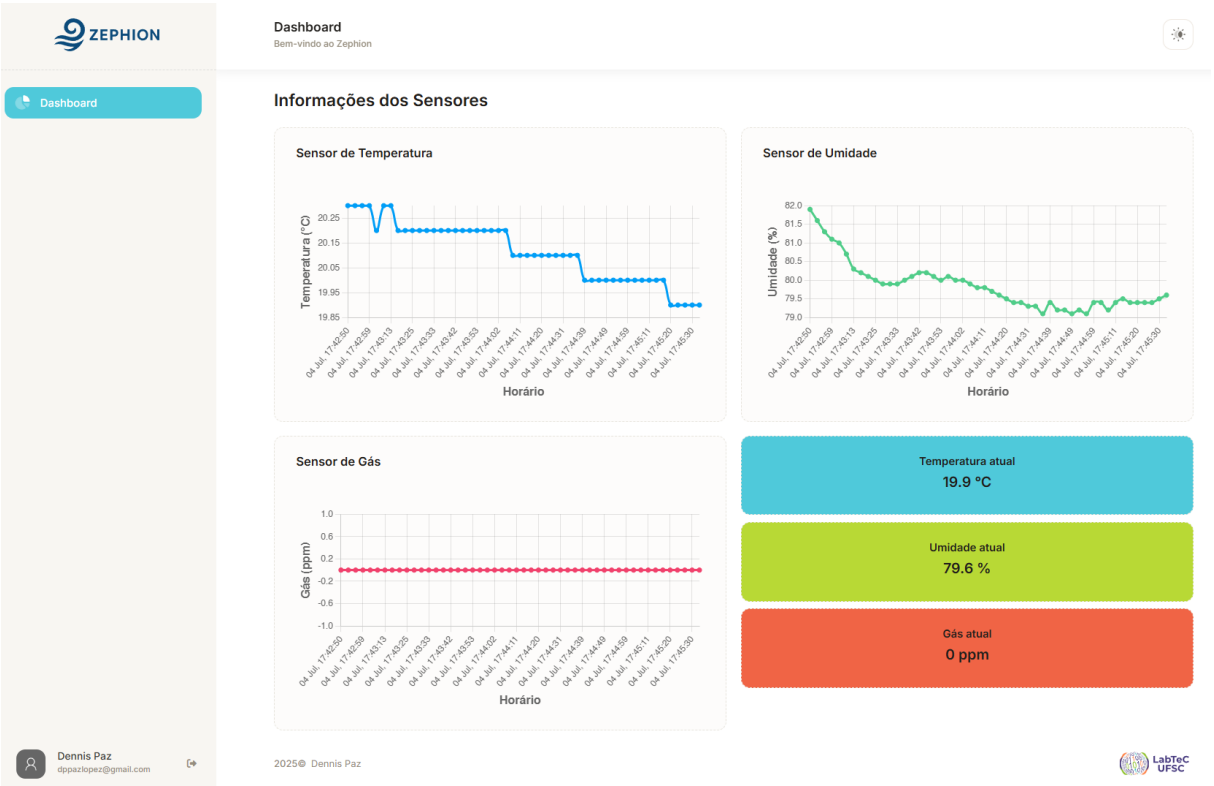

Figura 1 – Exemplo de dashboard exibindo dados dos sensores em tempo real.

3 DESAFIOS E APRENDIZADOS

Durante o desenvolvimento do projeto, diversos desafios foram enfrentados, entre eles:

- **Sincronização entre Wi-Fi e BLE:** Foi necessário implementar um controle rigoroso sobre o uso de memória e recursos do ESP32 para evitar falhas durante a troca entre os modos de comunicação.
- **Configuração segura do servidor Django:** A implementação de HTTPS e a exposição via Ngrok exigiram ajustes na configuração do Django e do banco de dados, além da gestão de certificados.
- **Integração entre frontend e backend:** A utilização de HTMX e Django Cotton para proporcionar atualizações assíncronas sem recarregar a página demandou estudo aprofundado e refatoração da arquitetura frontend.

Um desafio adicional foi a tentativa de desenvolver um aplicativo para receber os dados via BLE em um iPhone. No entanto, a ausência de um ambiente de desenvolvimento macOS (requisito para desenvolvimento iOS) impediu a criação de um aplicativo nativo para o sistema operacional da Apple. Esse obstáculo limitou a validação completa do modo BLE em dispositivos iOS, demonstrando na prática a importância de considerar o ecossistema de desenvolvimento ao planejar soluções móveis.

Entre os aprendizados mais relevantes deste projeto, destacam-se:

- **BLE como mecanismo de fallback:** A adoção do BLE como solução de redundância provou-se altamente eficaz, garantindo a continuidade da coleta de dados em situações de falha na rede Wi-Fi.
- **Integração IoT-Web:** O projeto evidenciou que é possível integrar, de forma eficiente, dispositivos IoT com sistemas web modernos, utilizando práticas atuais de desenvolvimento web.
- **Otimização do Firmware:** A experiência adquirida na otimização do ciclo de coleta e envio de dados no ESP32 foi fundamental para garantir a estabilidade e o desempenho do sistema.