UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO (2025)

DEVICE CONTROL SIM: UM SIMULADOR PARA CONTROLE DE DISPOSITIVOS BASEADO EM IOT

Ricardo Cézar Fernandes de Melo Júnior[1], Paulo Henrique Lopes Silva[2]

RESUMO

A crescente demanda por eficiência energética, economia de recursos e praticidade no cotidiano tem impulsionado o interesse pela automação de ambientes. Tecnologias como a IoT (*Internet of Things*, traduzido para Internet das Coisas), computação em névoa e protocolos de comunicação assíncrona tornaram possível o desenvolvimento de sistemas automatizados cada vez mais inteligentes e conectados. No entanto, esse tipo de aplicação ainda enfrenta desafios relevantes, como a complexidade na comunicação entre dispositivos distribuídos, tolerância a falhas, escalabilidade e a necessidade de resposta em tempo real. Diante dessas dificuldades, optou-se pelo desenvolvimento de um simulador, com o objetivo de testar e validar a viabilidade de um sistema de automação antes de sua implantação real. Este trabalho propõe a criação de um software de simulação voltado à automação de ambientes utilizando microcontroladores IoT. A aplicação simula a comunicação entre dispositivos e servidores, organizados em uma estrutura distribuída com centralização em um servidor na nuvem. O estudo de caso escolhido para a simulação é o Laboratório de Ciência da Computação (LCC) da UFERSA, com dois andares e diversas salas contendo equipamentos eletrônicos. O simulador permite controlar remotamente esses dispositivos, automatizando tarefas como o desligamento de equipamentos ao fim do expediente. Os resultados demonstram que a solução é eficaz no gerenciamento remoto e simultâneo de múltiplos dispositivos, revelando o potencial da aplicação para ambientes reais, tanto acadêmicos quanto corporativos.

Palavras-chave: automação; IoT; simulação; rede de dispositivos; *cloud computing*.

ABSTRACT

The growing demand for energy efficiency, resource savings, and everyday convenience has boosted interest in automating environments. Technologies such as IoT (Internet of Things), fog computing, and asynchronous communication protocols have enabled the development of increasingly intelligent and connected automated systems. However, this type of application still faces significant challenges, such as the complexity of communication between distributed devices, fault tolerance, scalability, and the need for real-time responses. In light of these difficulties, a simulator was developed to test and validate the feasibility of an automation system before its actual implementation. This work proposes the creation of simulation software focused on the automation of environments using IoT microcontrollers. The application simulates the communication between devices and servers, organized within a distributed structure with centralization on a cloud server. The chosen case study for the simulation is the Computer Science Laboratory (LCC) of UFERSA, which comprises two floors and several rooms housing electronic equipment. The simulator enables remote control of these devices, automating tasks such as shutting down equipment at the end of the workday. The results demonstrate that the solution is effective in the remote and simultaneous management of multiple devices, revealing the potential of the application for real-world settings in both academic and corporate environments.

Keywords: automation; IoT; simulation; device network; cloud computing.

---

[1] Graduando do Bacharelado em Ciência da Computação da UFERSA
[2] Professor Associado do Departamento de Computação da UFERSA/Mossoró

1 INTRODUÇÃO

Com o avanço acelerado da tecnologia, dispositivos eletrônicos tornaram-se cada vez mais essenciais no cotidiano das pessoas. Residências, empresas, universidades e diversos outros ambientes passaram a contar com uma gama de equipamentos como televisores, projetores, ar-condicionados, computadores e sensores inteligentes, todos desempenhando papeis cruciais na automatização de tarefas. Esse crescimento é impulsionado pela evolução da *Internet of Things* (IoT, traduzido para Internet das Coisas), que tem permitido uma comunicação eficiente entre dispositivos físicos conectados à rede, promovendo maior integração e automação em diferentes setores (Yaqoob et al., 2019; Laghari et al., 2025). Dessa forma, a interconexão de dispositivos amplia a eficiência na execução de tarefas rotineiras, reduzindo a necessidade de intervenção humana e otimizando recursos (Faccioni Filho, 2016).

A IoT é caracterizada pela interconexão de objetos físicos à Internet, permitindo que eles coletem, compartilhem e processem dados em tempo real, otimizando o funcionamento de sistemas diversos. A IoT oferece soluções mais inteligentes e autônomas para os desafios cotidianos, com o objetivo de integrar o mundo físico ao digital (Faccioni Filho, 2016). A sua aplicação vai desde ambientes industriais e urbanos até usos domésticos, sendo essencial para o desenvolvimento de cidades inteligentes e infraestruturas sustentáveis.

Nesse contexto, os aparelhos IoT apresentam vantagens significativas: são dispositivos de baixo consumo energético, compactos, com grande capacidade de sensoriamento e que podem ser instalados em locais remotos ou de difícil acesso. Essa flexibilidade possibilita a coleta de informações cruciais para tomadas de decisão e o acionamento remoto de sistemas, otimizando recursos e promovendo eficiência. A combinação com a tecnologia *fog computing* junto da IoT permite uma resposta rápida a eventos locais, reduzindo a latência e o congestionamento da rede (Yaqoob et al, 2019).

Além disso, a automação baseada em IoT tem potencial direto na redução de custos operacionais, sobretudo no consumo de energia. Estudos demonstram que a domótica pode proporcionar uma economia energética de até 40% em determinados contextos, ao permitir a integração e gerenciamento inteligente de dispositivos (Santos et al., 2019).

Com base nesses fatores, este trabalho propõe o desenvolvimento de um simulador de automação utilizando microcontroladores IoT para controlar remotamente os equipamentos do Bloco de Laboratórios de Ciência da Computação da UFERSA. Os dispositivos são conectados a servidores locais distribuídos por andar e se comunicam com um servidor central hospedado em nuvem, responsável por armazenar e processar todos os dados da rede de forma segura e acessível.

O objetivo central é facilitar o gerenciamento inteligente desses dispositivos, otimizando tarefas simples e recorrentes como o desligamento de equipamentos ao fim do expediente. Isso visa não apenas reduzir desperdícios e custos, mas também modernizar a infraestrutura do ambiente universitário com soluções tecnológicas avançadas.

2 DESENVOLVIMENTO

Nesta seção será apresentada toda a organização e o planejamento no desenvolvimento, assim como também explicar os conceitos e estudos utilizados na confecção desse projeto.

2.1 Método

A fim de organizar melhor a explicação detalhada do sistema, toda a construção foi separada em etapas, desde os estudos até o produto final, como mostra a seguir:

1. **Fundamentação teórica:** Esta etapa compreende a revisão e análise dos conceitos relevantes para o projeto. São abordados temas como Internet das Coisas (IoT), computação em névoa, protocolos de comunicação, entre outros, que fundamentam as decisões técnicas adotadas no desenvolvimento do simulador.

2. **Trabalhos relacionados:** Esta etapa apresenta estudos e projetos similares na área de automação com IoT, destacando abordagens utilizadas e tecnologias aplicadas. A análise serve de base para comparar soluções existentes e evidenciar os diferenciais do sistema desenvolvido neste trabalho.
3. **Tecnologias utilizadas:** Detalha-se aqui cada tecnologia empregada na implementação do sistema, com suas respectivas justificativas. A seleção de ferramentas, linguagens e *frameworks* é discutida com base em critérios como de aderência aos requisitos do projeto.
4. **Arquitetura geral do sistema:** Esta seção descreve a topologia adotada pelo simulador, especificando os componentes envolvidos, o fluxo de dados entre as entidades, e os mecanismos de controle utilizados. A arquitetura é apresentada de forma clara e hierárquica, com foco na distribuição e comunicação entre os nós da rede.
5. **Diagrama de sequência:** Apresenta-se os diagramas de sequências que representam de forma visual como os componentes do sistema interagem ao longo do tempo. Mostrando a ordem das mensagens trocadas entre objetos ou componentes durante a execução de funcionalidades específicas.

Essa divisão metodológica permite uma abordagem sistemática para o desenvolvimento do simulador, garantindo que cada etapa seja devidamente estruturada e documentada.

2.2 Fundamentação Teórica

Para construir um sistema que simula uma rede de conexão em sistemas heterogêneos como equipamentos IoT e servidores, foi necessário o levantamento e estudo sobre tais dispositivos, que ganhou muita notoriedade pela capacidade de realizar inúmeras funcionalidades. Desse modo, as próximas subseções são dedicadas no destaque e explicação de conceitos relacionados a este trabalho.

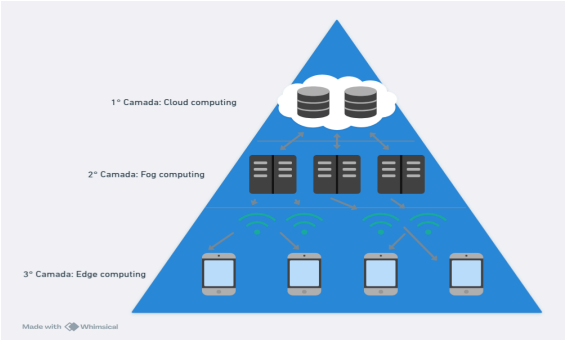
2.2.1 IoT

A princípio, essa nova tecnologia surgiu para dar uma nova visão sobre o uso da Internet, ao permitir conectar e abranger além dos computadores, também os objetos de uso diário. A Internet das coisas cria uma rede complexa e adaptativa ao conectar diversas "coisas", sendo estas físicas ou virtuais, e garantindo a interoperabilidade, que significa a eficácia em troca de mensagens em sistemas heterogêneos (Faccioni Filho, 2016; Yaqoob et al., 2019). Sendo assim, como citado anteriormente, tem capacidades para muitas funcionalidades, como sensoriamento de ambientes, podendo realizar análises de várias métricas existentes e conseguir armazenar toda essa informação, mesmo se estiver em lugares distantes, precisando de latência mínima para se manter conectado à rede.

2.2.2 *Fog Computing*

Em vista disso, ao criar uma infraestrutura com Internet das coisas, se forma um novo conceito chamado de *fog computing* (computação na neblina). Essa área surgiu após a *cloud computing* (computação em nuvem) e *edge computing* (computação na borda), combinando características de ambas para otimizar o processamento e a transmissão de dados (Laghari et al., 2025), conforme a Figura 1 abaixo.

Figura 1 – Comparação entre *Edge*, *Fog* e *Cloud Computing*



Fonte: Autoria Própria (2025)

A *cloud computing* (computação na nuvem) é um termo designado tanto para plataformas quanto para aplicações, em que pode ser um conjunto de recursos computacionais virtuais ou físicos centralizados, na qual possui diversas finalidades como: recursos sob demanda, armazenamento, banco de dados, redes e *software*. Porém, exige uma conexão estável para garantir um processamento eficiente dos dados transmitidos (Boss et al., 2007).

Em sequência, a *edge computing* (computação na borda) surge com uma nova utilidade de trazer essa nuvem para mais perto do usuário. Isso permite que operações críticas sejam executadas mais rapidamente, sem a necessidade de depender exclusivamente da nuvem para processamentos complexos (Yaqoob et al., 2019). Desse modo, os serviços na borda são os microcontroladores, servidores locais e também dispositivos móveis, desde que estejam próximos dos sensores ou atuadores responsáveis pela coleta ou execução de dados, na qual realizam a tarefa pesada, mesmo que limitada, garantindo a entrega rápida de dados.

Com o detalhamento dessas duas tecnologias, a *fog computing* (computação na névoa) reside no meio dessas outras duas, pois não muito diferente ela é uma extensão da nuvem, realizando armazenamento, processamento e análise de dados mais perto dos dispositivos em borda, como: roteadores, *gateways* e aparelhos IoT. Por conta disso, não existe centralização de dados, na qual permite maior eficiência na gestão dos dispositivos IoT, reduzindo a sobrecarga na nuvem e otimizando o tráfego de dados (Laghari et al., 2025).

2.2.3 Comunicação

Nesse sentido, para criar comunicação entre computadores junto da computação em névoa é necessário o uso dos conceitos de sistemas distribuídos, que consiste em serviços localizados em um nó distinto da rede, podendo ser geograficamente distante, e se comunica com os demais por meio da troca de mensagens através de uma rede de comunicação. Dessa forma, sendo capaz de monitorar ou até mesmo de enviar comandos para uma máquina que reside em outro local. Por conta disso, neste projeto foram analisados diferentes protocolos de comunicação.

O primeiro deles é a comunicação assíncrona, ocorre quando uma mensagem é enviada sem a necessidade de aguardar uma resposta imediata, permitindo que a aplicação continue seu fluxo de execução sem bloqueios (Yaqoob et al., 2019).

Em conseguinte tem a comunicação síncrona, que fundamenta-se na troca de dados, onde cada envio de mensagem necessita de uma resposta imediata, ou seja, criando métodos bloqueantes, que podem travar toda a aplicação. Esse tipo de comunicação é empregado em sistemas críticos, onde a consistência e a confiabilidade dos dados são prioridades, mas pode tornar-se um problema em sistemas distribuídos devido à necessidade de sincronização constante (Sato, 1995).

2.2.4 Paralelismo

Nessa perspectiva, ao utilizar comunicação síncrona, o método de esperar conexões é bloqueador, pois enquanto ele não recebe nenhuma conexão nova ele trava toda a aplicação, No entanto, a adoção da computação paralela pode mitigar esse problema, permitindo que diferentes tarefas sejam executadas simultaneamente, evitando bloqueios completos da aplicação (Senger; Guardia, 2025).

Tendo em vista um processo muito complexo, ou seja, contendo muitas instruções com decisões e ramificações, além de grande volume de dados. Pode ser dividido em problemas menores, na qual todas executam ao mesmo tempo e no final entrega a resposta de maneira muito mais rápida.

Dessa forma, esse modelo é amplamente adotado na arquitetura moderna dos processadores, que contam com múltiplos núcleos capazes de gerenciar múltiplas *threads* ao mesmo tempo (Lawrence Livermore National Laboratory, 2025). Cada núcleo pode criar e manipular *threads* que são unidades de execução mais leves que os processos convencionais, compartilhando o mesmo espaço de memória e contexto do processo principal. Essa característica permite que múltiplas tarefas sejam executadas em paralelo, otimizando o uso dos recursos do sistema.

Além disso, por conta do sistema operacional ser responsável pelo escalonamento e gerenciamento dessas *threads* convencionais, causando limitação na quantidade de criação delas. Foi utilizado as *threads virtuais*, por serem gerenciadas pela JVM, permitindo a criação de milhares de instâncias com consumo mínimo de recursos.

2.3 Trabalhos Relacionados

Diversas iniciativas têm sido desenvolvidas na área de automação residencial com IoT, evidenciando diferentes estratégias para controle, comunicação e integração entre dispositivos inteligentes. Um dos enfoques mais comuns envolve o uso de sistemas de baixo custo, como aqueles baseados em plataformas Arduino, que permitem o controle de dispositivos por meio de aplicativos móveis e conexão via redes Wi-Fi, priorizando a acessibilidade e a simplicidade na implementação (Wanzeler; Fülber; Merlin, 2016). Também há propostas com fins educacionais, como a criação de maquetes automatizadas que facilitam o ensino interdisciplinar de conceitos técnicos, aproximando teoria e prática com o uso de microcontroladores e ferramentas de controle remoto (Camargos et al., 2022).

Outros trabalhos exploram soluções mais robustas, adotando protocolos como o MQTT (*Message Queuing Telemetry Transport*) e integrando os sistemas a serviços em nuvem, visando à escalabilidade e ao controle remoto por meio de aplicações móveis (Kandauroff; Molina, 2019). A interoperabilidade entre dispositivos de diferentes fabricantes e padrões de comunicação também é um desafio recorrente, sendo abordada por propostas que utilizam plataformas centralizadoras, como o *Home Assistant*, para facilitar a integração entre tecnologias como Zigbee, Wi-Fi e outros protocolos heterogêneos (Rossi; Siqueira, 2022).

Além disso, há estudos que destacam o uso de simuladores e ambientes virtuais como ferramentas para validar previamente sistemas distribuídos, especialmente em contextos acadêmicos, permitindo testar a lógica de funcionamento sem necessidade de testes práticos com instalações físicas (Santos; Ferreira, 2022). Ainda no campo da acessibilidade e eficiência, algumas propostas utilizam aplicações *web* simples acopladas a microcontroladores para automatizar dispositivos domésticos, com destaque para a redução de custos operacionais e do consumo energético (Soares; Cerqueira, 2022). Complementarmente, pesquisas voltadas à realidade brasileira abordam a adaptação de soluções de domótica a edificações existentes, considerando fatores econômicos, estruturais e culturais, e evidenciando os ganhos em conforto, segurança e sustentabilidade (Messias, 2007).

Por fim, estudos mais recentes exploram a aplicação de *fog computing* para *smart homes* como uma alternativa à computação em nuvem, permitindo a otimização do consumo energético e a redução do tráfego de rede por meio de filtros preditivos aplicados localmente aos dispositivos (Stojkoska; Trivodaliev, 2017).

2.4 Tecnologias utilizadas

Esta seção foi dividida, para melhor explicação de todas as tecnologias selecionadas que estão presentes em alguma parte do projeto.

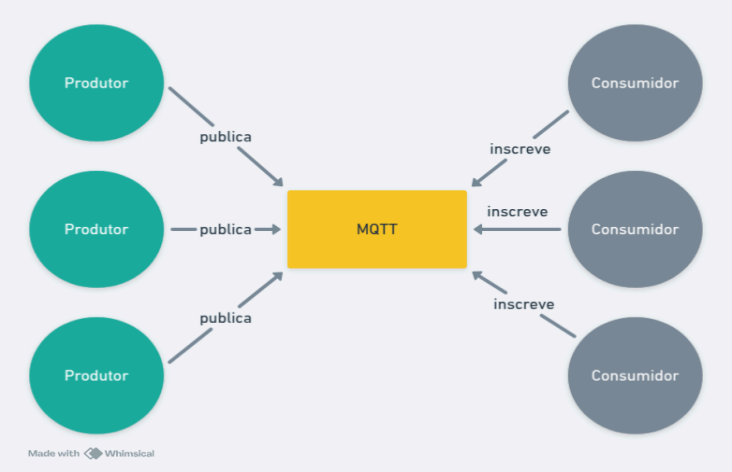
2.4.1 Java

A linguagem de programação Java, é compilada para um formato intermediário chamado bytecode, que é executado pela JVM(*Java Virtual Machine*). Com isso, este projeto foi desenvolvido utilizando o Java 21, que introduz oficialmente o suporte a threads virtuais por meio do Projeto Loom (Oracle, 2025).

2.4.2 MQTT e Eclipse Paho

Conforme dito anteriormente, a comunicação assíncrona foi utilizada neste simulador, com o uso do protocolo MQTT (Banks; Gupta, 2015) com suporte da biblioteca Eclipse Paho (Eclipse Foundation, 2025). Nesse sentido, esse protocolo é baseado na estrutura de fila, chamado de *broker*, pois o primeiro elemento que entra é também o primeiro a sair. Além disso, ele é um modelo do tipo publisher/subscriber.

Figura 2 - protocolo MQTT



Fonte: Autoria própria (2025)

Segundo a Figura 2, esse protocolo de mensagens é dividido pelos produtores que, neste caso, são os dispositivos IoT que enviam os dados para os tópicos na fila. Do outro lado, encontram-se os consumidores, que são os servidores na qual recebem as informações publicadas. Para conseguir ter a troca de mensagens, os consumidores devem estar inscritos no mesmo tópico, ou seja, os produtores irão enviar dados para o *broker* no canal específico e apenas os inscritos serão notificados e terão acesso a informação.

Portanto, na implementação do sistema foi utilizado a dependência do projeto *paho*, criada pela *eclipse foundation.* Essa biblioteca permite estabelecer conexão com o *broker* do projeto importado e conseguir publicar ou inscrever-se em algum tópico especificado.

2.4.3 JavaFx

A criação da interface do simulador foi realizada com a importação da biblioteca externa de código aberto chamada JavaFX 21 (OpenJFx, 2025), para ser compatível com a linguagem Java usada, que permite a criação de janelas, assim como também a estilização e funcionamento da página criada.

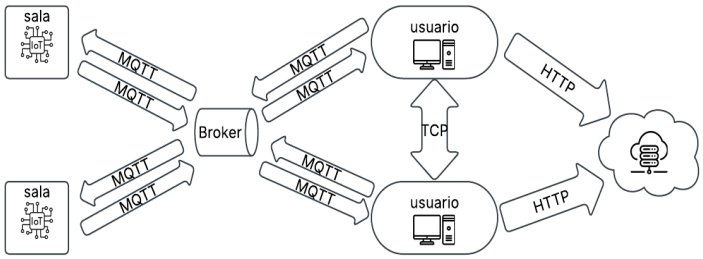
2.4.4 Programação *Web*

O servidor hospedado na nuvem, foi desenvolvido usando tecnologias comuns para *web* como *HTML5* e *CSS3* (W3C, 2014; W3C, 2023), que são linguagens de estruturação e estilização de páginas para navegadores. Além do mais, para a lógica e funcionamento deste servidor foi utilizado o *software* de código aberto multiplataforma chamado *node* (OpenJs Foundation, 2025), essa ferramenta é baseada no interpretador v8 do google, para permitir a execução de código *javascript* fora do navegador. Aliás a troca de mensagem entre os clientes e o servidor na nuvem, é realizada usando a biblioteca *WebSocket* (Faubel, 2011), um protocolo de comunicação em tempo real desenvolvido para navegadores, com as mensagens sendo recebidas no padrão de arquivo JSON (ECMA International, 2017), pois é um formato de texto que usa pares chave-valor para representar dados estruturados.

2.5 Arquitetura geral do sistema

Nessa seção é descrito como foi idealizado o planejamento e montagem da simulação do sistema.

Figura 3 - Estrutura do projeto



Fonte: Autoria própria (2025)

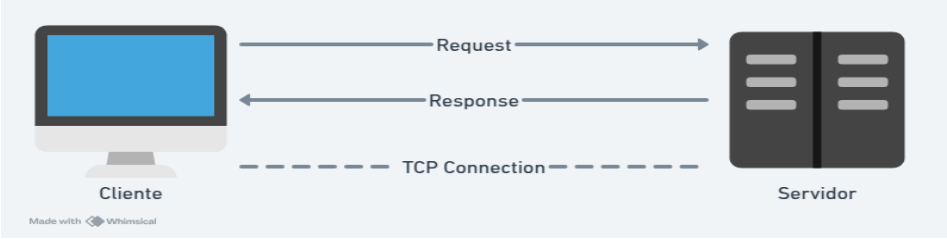
Conforme a Figura 3, é demonstrada toda a arquitetura do simulador, com cada sala possuindo um microcontrolador que controla os aparelhos do ambiente, como: luz, ar-condicionado, projetores e etc. Nesse sentido, os dispositivos IoT conectam-se aos usuários que são as máquinas através do *broker*, usando o MQTT como meio de transporte das mensagens. Para isso, cada microcontrolador define um tópico no *broker*, a partir disso é possível trocar mensagens. Além disso, cada cliente possui um servidor TCP próprio, assim permitindo conexões *peer-to-peer* (P2P), ou seja, é um tipo de comunicação sem a necessidade de um servidor central, como mostra a Figura 4 a seguir.

Figura 4 - Servidor TCP

Fonte: Autoria Própria (2025)

Como mostra a Figura 4, o servidor TCP é simples, pois o cliente envia o IP (*Internet Protocol* traduz-se para protocolo de Internet) próprio e o servidor estabelece uma conexão, com isso é possível trocar mensagens. Dessa forma, cada usuário da Figura 4 é um servidor TCP, assim, garantindo que um cliente conecte-se a outros e acesse mais microcontroladores, controlando mais equipamentos de outras salas. Por fim, cada computador dos usuários envia os dados das operações e conexões para um servidor na nuvem através do protocolo HTTP (*Hypertext Transfer Protocol*).

Figura 5 - Servidor HTTP



Fonte: Autoria Própria (2025)

Segundo a Figura 5, o servidor HTTP funciona de forma parecida com o servidor explicado anteriormente, a diferença consiste que toda requisição contém uma resposta logo em seguida, mesmo que a resposta seja vazia com código de bem sucedido. Nesse sentido, os clientes enviam todas as operações realizadas a estes servidores HTTP. Sendo assim é possível centralizar e criar um registro de históricos de comandos realizados em todo simulador.

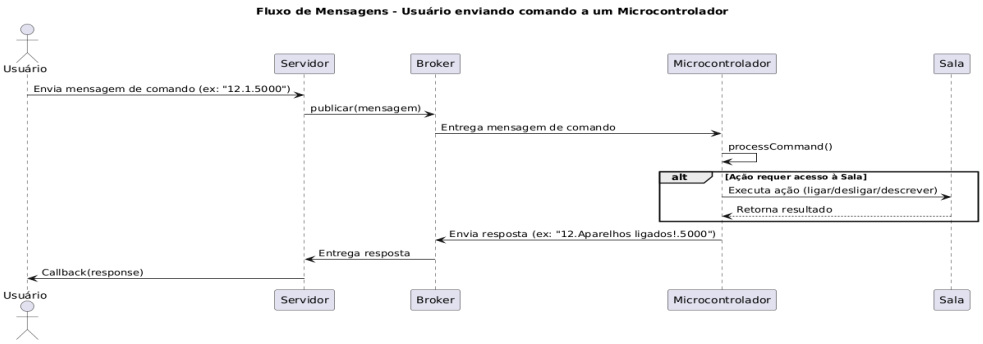
2.6 Diagrama de sequência

Para compreender o comportamento dinâmico do sistema proposto e a interação entre seus componentes ao longo do tempo, foram elaborados diagramas de sequência utilizando a notação UML. Estes diagramas representam de forma visual a troca de mensagens entre os atores e os elementos participantes durante a execução de funcionalidades específicas, permitindo analisar o fluxo de comunicação, o encadeamento de chamadas e os pontos de decisão envolvidos. A seguir, são apresentados dois cenários distintos: o primeiro descreve o envio direto de comandos do usuário a um microcontrolador. O segundo aborda a comunicação entre servidores distribuídos, ilustrando a interoperabilidade entre nós da rede e a escalabilidade da arquitetura desenvolvida.

2.6.1 Diagrama de Sequência: Comando direto do usuário ao microcontrolador

O primeiro diagrama representa o fluxo de mensagens quando um usuário envia um comando diretamente a um microcontrolador. A sequência de eventos ocorre conforme a Figura 6 abaixo.

Figura 6 - Diagrama de sequência 1



Fonte: Autoria própria (2025)

Conforme a Figura 6, o usuário publica um comando por meio de uma interface gráfica, que é transmitido ao componente Servidor responsável por preparar e encaminhar a mensagem.

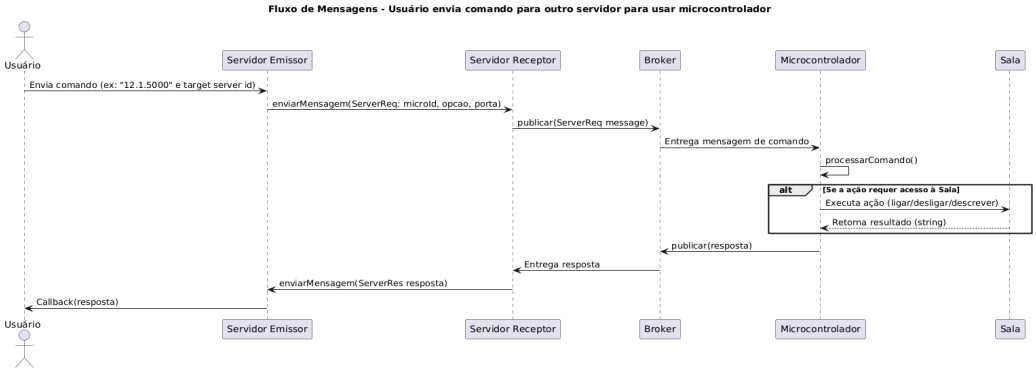
1. O comando é publicado no *broker*, que gerencia os tópicos de publicação/assinatura da arquitetura assíncrona.
2. O *broker* entrega a mensagem ao microcontrolador, que a interpreta via uma função de callback e executa o processamento do comando.
3. Caso o comando envolva controle de dispositivos, o microcontrolador acessa a instância da sala, que representa o ambiente controlado (ex: ligar luzes, desligar ar-condicionado etc.).
4. O resultado da ação é retornado para o microcontrolador.
5. Em seguida, o microcontrolador gera uma resposta, a qual percorre o caminho inverso: *broker* → Servidor → usuário.

Este diagrama demonstra claramente a eficiência da comunicação assíncrona com MQTT, permitindo o envio e recebimento de comandos e respostas de forma rápida e não bloqueante.

2.6.2 Diagrama de Sequência: Comando entre servidores distribuídos

O segundo diagrama representa um fluxo mais complexo, onde o usuário envia um comando para um microcontrolador que está vinculado a outro servidor na rede. A sequência de comunicação entre os componentes distribuídos é descrita de acordo com a Figura 7 a seguir.

Figura 7 - Diagrama de sequência 2



Fonte: Autoria Própria (2025)

Seguindo a Figura 7, abaixo está o passo a passo de como o fluxo de dados discorre nesse cenário.

1. O usuário envia um comando para o servidor emissor, que é o servidor local ao qual ele está conectado.
2. O servidor emissor identifica que o microcontrolador alvo pertence a outro servidor, e então realiza uma comunicação ponto a ponto (unicast) com o servidor receptor, enviando a requisição com o identificador do microcontrolador e os parâmetros da ação.
3. O servidor receptor processa a requisição e envia a mensagem ao *broker*.
4. O *broker* encaminha a mensagem ao microcontrolador correspondente.
5. O microcontrolador executa a ação solicitada (como ligar/desligar um aparelho), e se necessário, interage com o objeto sala para manipular os dispositivos.
6. Após processar o comando, o microcontrolador envia uma resposta, que retorna por meio do mesmo fluxo: microcontrolador → *broker* → Servidor Receptor.
7. O servidor receptor envia a resposta de volta ao servidor emissor por meio de unicast.
8. Finalmente, o servidor emissor repassa a resposta ao usuário, encerrando o ciclo da comunicação.

Esse diagrama evidencia o modelo de sistema distribuído e escalável, no qual múltiplos servidores e microcontroladores interagem de forma coordenada, mesmo estando em nós distintos da rede.

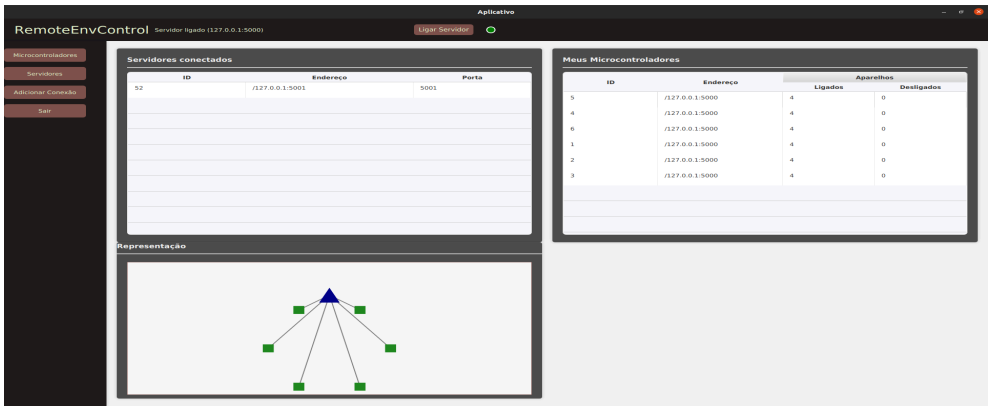
3 RESULTADOS

Nessa seção são descritos os resultados obtidos do simulador, mostrando o funcionamento do projeto com a interface gráfica, a simulação operando e o servidor na nuvem. Além do mais, vai ser detalhado o estudo de caso com a demonstração de três experimentos no cenário proposto.

3.1 Interface do Usuário

A interface do cliente, detalhando cada operação que pode ser realizada no sistema e como interpretar os dados recebidos durante o manuseio do simulador.

Figura 8 - Interface do Cliente



Fonte: Autoria própria (2025)

Segundo a Figura 8, a interface do usuário possui um painel vertical na lateral esquerda, um menu horizontal na parte superior e uma malha no centro da tela. Todas as operações possíveis do sistema estão presentes no menu vertical, que possui os *dialogs*, que é um controle gráfico no formato de uma pequena janela, na qual através dessa caixa de diálogo são enviadas as requisições do usuário, como mostra as imagens abaixo.

Figura 9 - Dialogs do simulador

Fonte: Autoria própria (2025)

Conforme a Figura 9, é perceptível todas as opções que podem ser escolhidas no simulador, como iniciar o servidor e especificar qual tópico ou sala quer inscrever-se. Além disso, é possível controlar microcontroladores ou servidores já conectados, na qual tem as ações de ligar, desligar ou descrever a sala escolhida. Outrossim, também é possível conectar a outra máquina se for necessário, apenas digitar endereço e porta de destino.

Além disso, observando a Figura 8, no *grid* encontram-se os principais componentes de visualização. Na primeira tabela são salvas todas as conexões com outros servidores, aparecendo os identificadores e o endereço destes servidores.

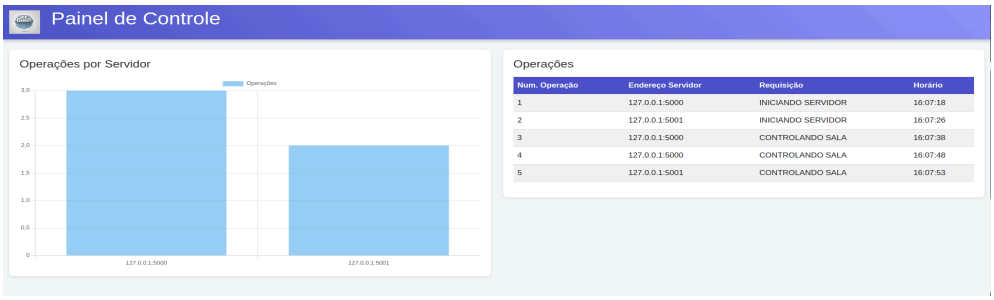
Na segunda tabela são mostrados todos os microcontroladores conectados, com os respectivos identificadores, endereços e os aparelhos pertencentes a ele. Estes aparelhos são divididos em ligados e desligados, dessa forma distinguindo os equipamentos eletrônicos que estão ou não operando.

Por fim, a última informação do *grid* é uma animação em tempo real demonstrando todas as conexões realizadas sendo desenhadas, onde o triângulo é a simbolização do servidor e os círculos os dispositivos IoT que estão conectados.

3.2 Interface do Servidor na Nuvem

A interface do servidor na nuvem é hospedada na plataforma especializada em computação em nuvem chamada *netlify* (Netlify, 2025). Assim, garantindo a centralização e armazenamento dos dados de todos os servidores. Nesse sentido, essa interface consiste em ser um *dashboard*, ou seja, um painel sem links externos com todas as informações necessárias acessíveis na janela principal.

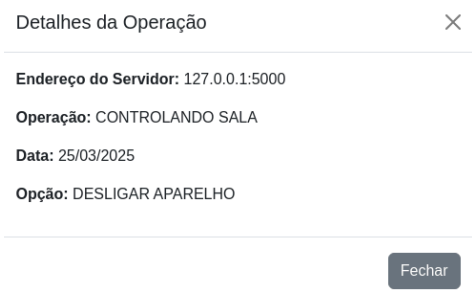
Figura 10 - Interface do Servidor em Nuvem



Fonte: Autoria própria (2025)

Conforme a Figura 10, no primeiro campo, tem um gráfico de todas as operações por servidor, em que é possível visualizar quais operações estão sendo realizadas em maior quantidade por servidor da simulação. No segundo campo tem uma tabela com todas as operações realizadas de cada servidor, mostrando o endereço do servidor que executa, a requisição realizada e o horário. Além disso, se clicar em alguma linha da tabela, abre uma janela detalhando mais informações sobre a operação realizada.

Figura 11 - Janela diálogo do servidor em nuvem



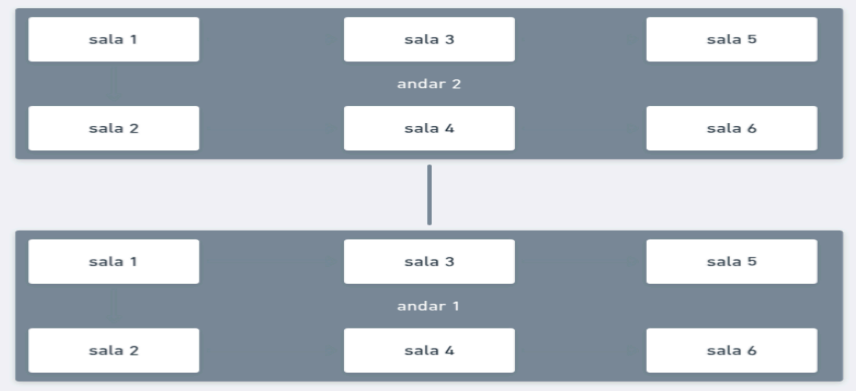
Fonte: Autoria Própria

A Figura 11 demonstra os detalhes das operações encontradas na tabela da Figura 10, pois nessa nova janela é possível ver exatamente que operação foi realizada e em qual servidor foi ativada a função. Por meio disso é possível registrar e salvar todos os acontecimentos do simulador e ter um controle sobre as operações realizadas.

3.3 Estudo de caso

A fim de realizar os testes do simulador e validar sua eficácia, é importante destacar o contexto e o problema que este projeto está tentando automatizar. Nesse aspecto, o ambiente de estudo de caso é o Bloco de Laboratórios de Ciência da Computação da UFERSA (Universidade Federal Rural do Semi-Árido), na qual contém dois andares e que cada um inclui seis salas, dentro dessas salas tem número variado de aparelhos. Para melhor entendimento, foi feita uma representação do local, conforme apresentado na figura 12.

Figura 12 - Representação do Departamento de Computação



Fonte: Autoria Própria (2025)

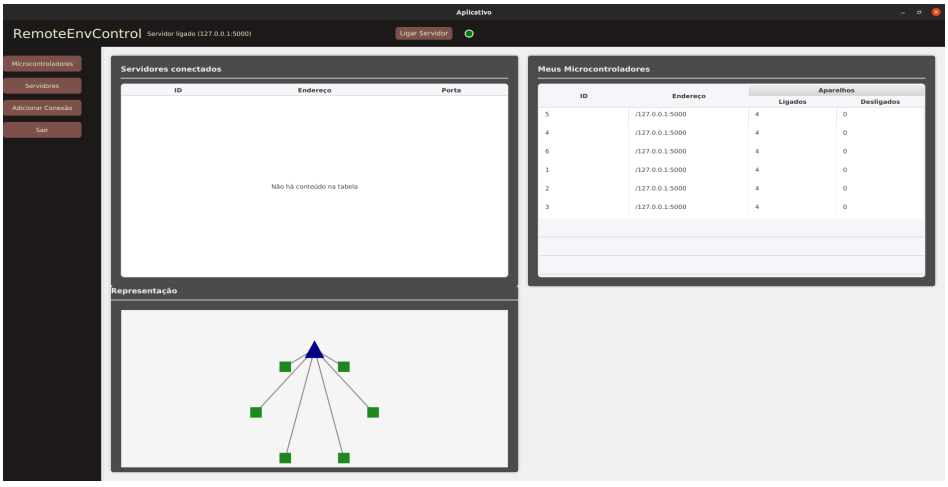
Baseando-se no modelo da Figura 12, o simulador vai atuar automatizando os processos de ligar e desligar os equipamentos básicos de uma sala do LCC, conectados a um microcontrolador simulado. Com isso, foram determinados três testes, sendo: o desligamento total dos aparelhos em dois andares, o desligamento em um andar apenas e gerenciar uma sala específica. Dessa forma, dois servidores serão ligados, sendo que cada um vai conter seis microcontroladores. Além disso, o estado inicial das salas vai ser de todos os aparelhos já ligados.

3.3.1 Primeiro Teste

O primeiro teste consiste em simular o desligamento de dois andares, para isso foram criados dois servidores, em que cada um vai conectar-se a seis salas e cada sala contendo dois equipamentos. Então, o objetivo é conseguir com apenas um dos servidores desligar os dois andares. Como mostra o passo a passo abaixo:

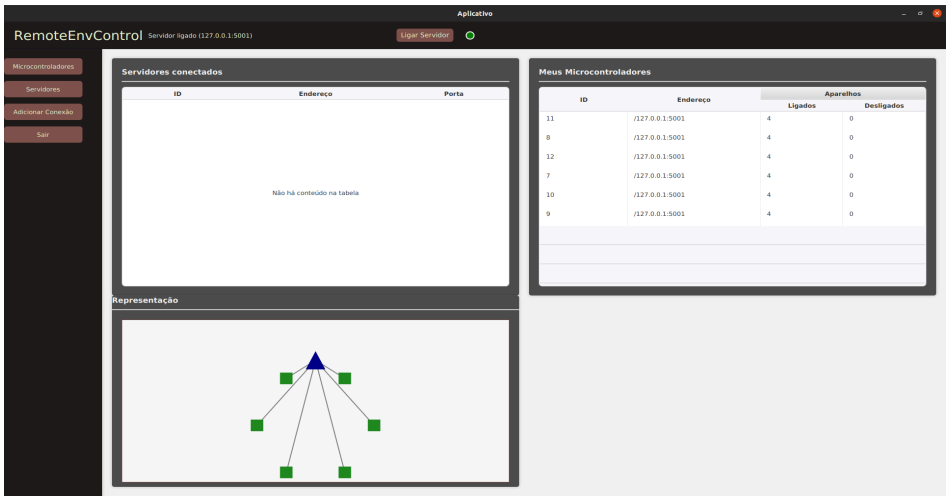
1. **Inicialização do sistema:** criação de dois simuladores com seis microcontroladores cada. Com isso, o primeiro simulador foi hospedado na porta 5000 e conectou-se ao *broker* através do tópico “andar1”, já o segundo foi inicializado na porta 5001 e inscrito no tópico “andar2”;

Figura 13 - Inicialização do primeiro servidor



Fonte: Autoria Própria (2025)

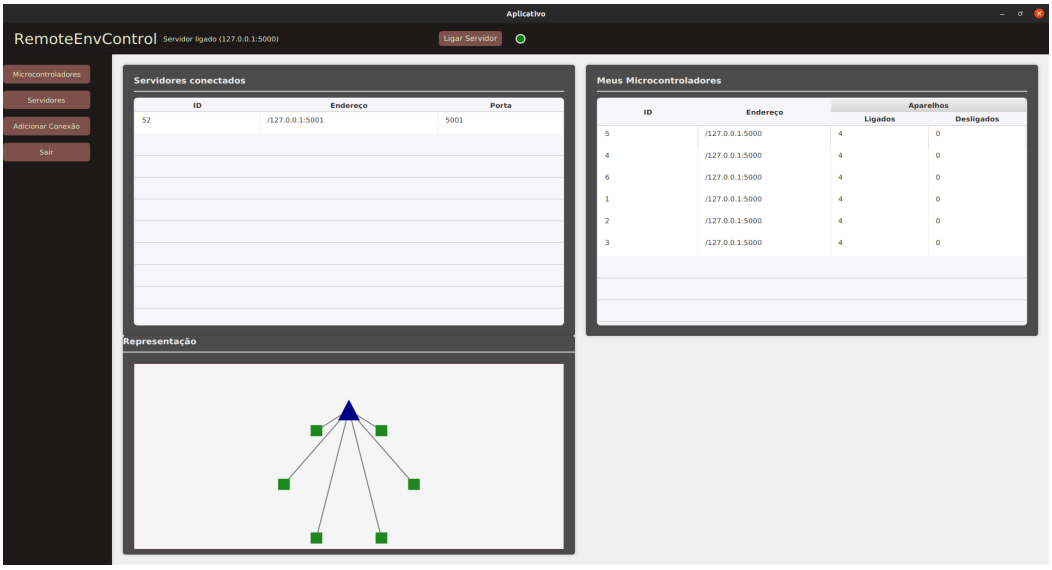
Figura 14 - Inicialização do segundo servidor



Fonte: Autoria Própria (2025)

2. **Conectar servidor:** Para que o servidor do primeiro andar consiga operar as máquinas do segundo piso, é preciso estabelecer a conexão entre eles. Desse modo, ao utilizar a opção "adicionar conexão" no menu lateral esquerdo, é possível realizar a conexão e viabilizar a operação dos dispositivos do segundo andar;

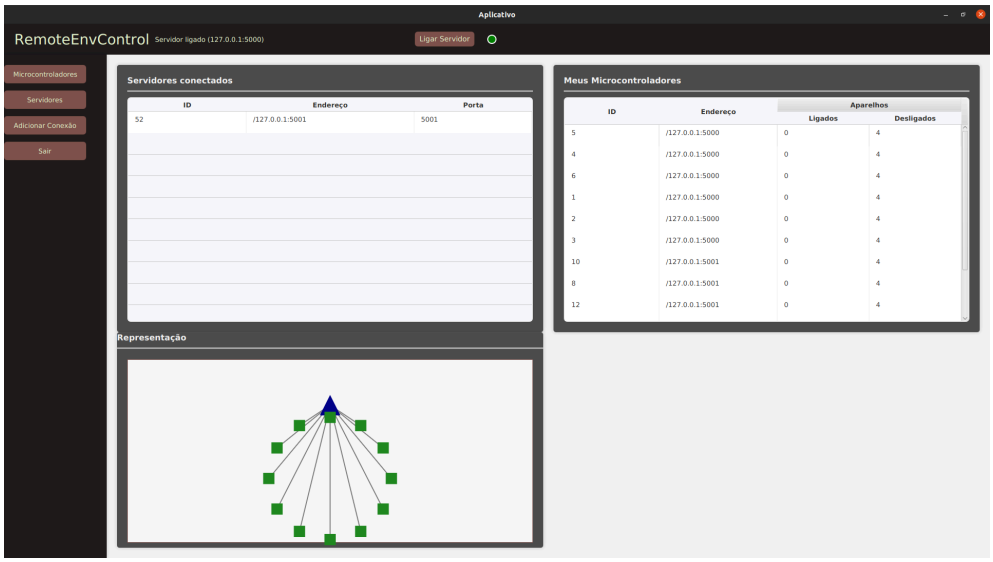
Figura 15 - Adição de Conexão



Fonte: Autoria Própria (2025)

3. **Desligamento das máquinas:** Com os servidores conectados, é possível desligar todos os equipamentos dos dois andares, primeiramente utilizando o controle de microcontroladores, aperta-se nas opções "desligar" e "todos". Depois, utilizando o controle de servidores, inscreve o identificador do servidor que deseja controlar e passa as mesmas opções de "desligar" e "todos".

Figura 16 - Desligamento dos blocos



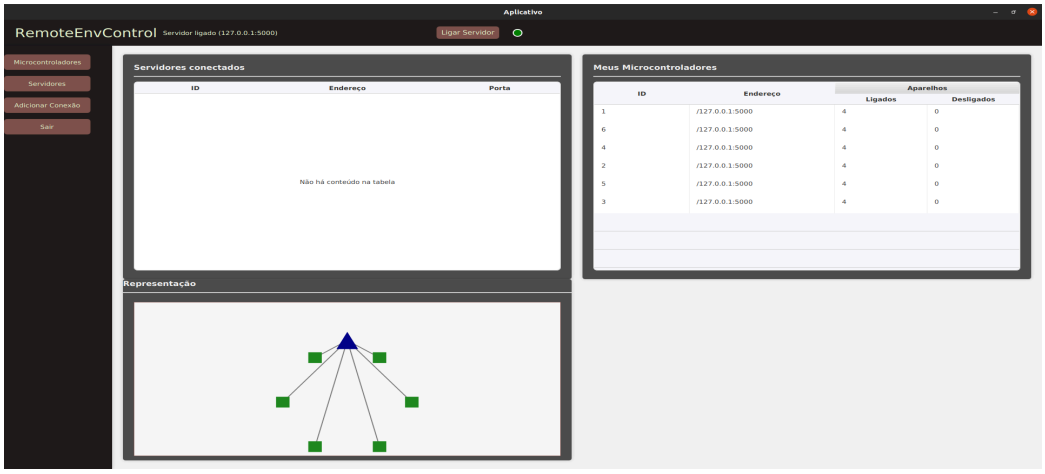
Fonte: Autoria Própria (2025)

Por fim, é visível que o servidor um agora contém os microcontroladores 1 a 12, ou seja, todas as salas e através da tabela é possível ver que todos os aparelhos estão desligados.

3.3.2 Segundo Teste

Para o segundo teste, o objetivo é desligar todos os aparelhos eletrônicos nas salas de um andar e cada sala contendo dois equipamentos. Para isso, foi criado um experimento com seis microcontroladores inseridos no tópico do *broker* “andar1”, os aparelhos foram inicializados ligados. Com isso, o teste segue o determinado passo a passo:

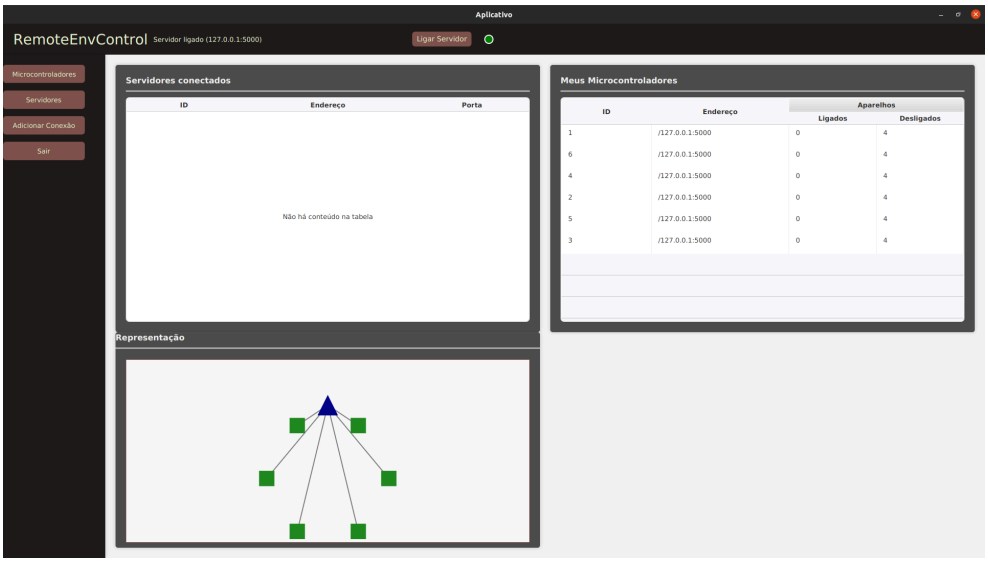
Figura 17 - Inicialização do segundo teste



Fonte: Autoria Própria (2025)

Em seguida, ao utilizar o controle de microcontroladores foi apertado as opções “desligar” e “todos”.

Figura 18 - Desligamento do primeiro andar

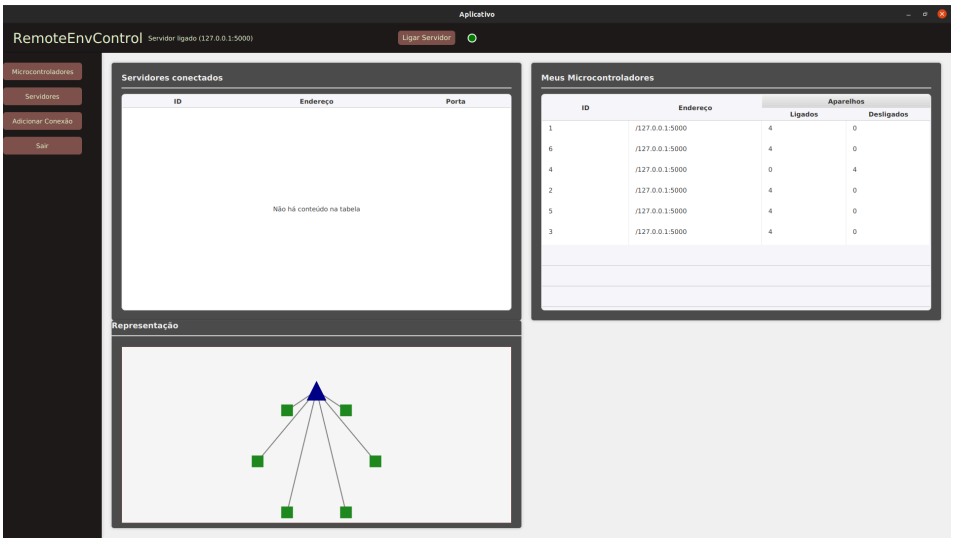


Fonte: Autoria Própria (2025)

3.3.3 Terceiro Teste

Para o terceiro teste, a meta é desligar os aparelhos em apenas uma sala, para isso foi usado as mesmas configurações dos testes anteriores. Com isso, a lógica segue praticamente a mesma, mas com a diferença que no controle de microcontroladores, passa o identificador da sala que deseja manusear. Neste caso, foi escolhida a sala quatro e desligar os aparelhos dela enquanto que todas as outras salas permaneçam ligadas.

Figura 19 - Desligamento de uma sala específica



Fonte: Autoria Própria (2025)

Portanto, todos os testes realizados serviram de validação para o simulador, mostrando o desempenho no gerenciamento de dispositivos IoT e capaz de automatizar o desligamento ou ligamento de equipamentos em

diferentes cenários. Com isso, os resultados mostraram que os objetivos foram atendidos, proporcionando um sistema confiável e escalável para automatização de atividades cotidianas.

4 CONCLUSÕES

De acordo com os objetivos definidos para este projeto, foi desenvolvida uma simulação de uma situação real, para automação de equipamentos comuns do dia a dia, na qual possui um foco maior na conexão e gerenciamento de dispositivos IoT, com a utilização de paralelismo e *fog computing*, para criação de um sistema mais eficiente e tolerante a falhas.

Nessa perspectiva, uma das principais tecnologias utilizadas para a eficiência na comunicação com dispositivos IoT, foi o uso do protocolo MQTT, pois, por ter a arquitetura baseada em *broker*, facilita a troca de mensagens de forma assíncrona e escalável. Além disso, o uso de tópicos permite um desempenho melhorado e otimizado, reduzindo o tráfego de rede e garantindo que apenas os serviços cadastrados sejam notificados nas informações. Ademais, a flexibilidade do MQTT concede integração com sistemas muito diferentes, encaixando de forma perfeita para a complexidade que é a Internet das coisas.

Outro aspecto importante foi a implementação do servidor na nuvem, que desempenha o papel de armazenar e centralizar os dados de todo simulador. Esse serviço foi feito para garantir o histórico das interações entre os dispositivos, armazenando as operações realizadas, como o desligamento ou ligamento de equipamentos. Mesmo que sendo uma simples aplicação, apresentou desafios na construção, como conseguir garantir uma comunicação eficiente e segura com o usuário. Com isso, a presença dele é essencial para a análise de dados e monitoramento contínuo do sistema.

Desta forma, um dos principais desafios do projeto foi garantir a interoperabilidade, pois cada componente do sistema é diferente e comunica-se de forma diferente, sendo assim um sistema totalmente heterogêneo. A utilização de dispositivos IoT, servidores locais e em nuvem, com cada um possuindo especificações e particularidades, desenvolve grandes desafios técnicos, para garantir sincronização e tratamento de falhas, pois qualquer falha pode comprometer o funcionamento do sistema como todo.

Nesse contexto, para trabalhos futuros dessa aplicação, seria interessante o desenvolvimento de interfaces *web* e *mobile*, a fim de tornar o sistema mais acessível, que permitirá o gerenciamento remoto de forma mais prática e intuitiva. Além disso, a implementação de algoritmos de balanceamento de carga e replicação de serviços, garantiria maior resiliência e tolerância a falhas. Outra melhoria é a integração com modelos de inteligência artificial para otimizar a gestão dos dispositivos IoT. Com essas melhorias, o simulador pode evoluir para cenários reais e, posteriormente, ser comercializado.

Por fim, este simulador mostrou-se bastante eficaz ao que foi proposto, na qual ao realizar uma amostra em um cenário real, pode se tornar um aplicativo para utilização em departamentos empresariais ou até mesmo residenciais, conseguindo automatizar tarefas simples de maneira fácil e eficaz.

REFERÊNCIAS

FACCIONI FILHO, Mauro. **Internet das Coisas**. Palhoça: UnisulVirtual, 2016.

SENGER, Hermes; GUARDIA, Hélio Crestana. ***Processos e Threads*.** Bacharelado em Sistemas de Informação – EaD UAB/UFSCar, 2025.

LAWRENCE LIVERMORE NATIONAL LABORATORY. **Introduction to Parallel Computing.** Disponível em: https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial. Acesso em: 12 mar. 2025.

SATO, Liria Matsumoto. **Ambientes de programação para sistemas paralelos e distribuídos.** 1995. Tese (Livre Docência) – Universidade de São Paulo, São Paulo, 1995. . Acesso em: 12 mar. 2025.

LAGHARI, Asif; HE, Haider; LAGHARI, Raheel; KHAN, Ahsan; MIN, Gwanggil. **Comparison of Fog Computing & Cloud Computing.** Disponível

em:https://www.researchgate.net/publication/330090457_Comparison_of_Fog_Computing_Cloud_Computing. Acesso em: 13 mar. 2025.

BOSS, Greg; MALLADI, Padma; QUAN, Dennis; LEGREGNI, Linda; HALL, Harold. **Cloud computing.** 2007. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/30844301/Cloud_computing_wp_final_8Oct-libre.pdf?1392220914. Acesso em: 13 mar. 2025.

YAQOOB, Ibrar; ABDELWAHAB, Salah; JAYARAJAH, Kasthuri; LARROUDE, Sophie; IMRAN, Muhammad Ali; GUO, Song; NGUYEN, Dinh Thai; DOU, Wanchun; NGUYEN, Trung Q. **Edge computing: A survey.** Disponível em: https://www.researchgate.net/publication/331362529_Edge_computing_A_survey. Acesso em: 13 mar. 2025.

ORACLE. **Java Platform, Standard Edition.** Versão 21. Disponível em: https://www.oracle.com/java/technologies/javase-downloads.html. Acesso em: 22 mar. 2025.

BANKS, Andy; GUPTA, Rahul. **MQTT Version 3.1.1 Plus Errata 01.** OASIS Standard. 2015. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Acesso em: 22 mar. 2025.

ECLIPSE FOUNDATION. **Eclipse Paho Project.** 2025. Disponível em: https://www.eclipse.org/paho/. Acesso em: 22 mar. 2025.

OPENJFX. **JavaFX Documentation.** Versão 21. 2025. Disponível em: https://openjfx.io. Acesso em: 22 mar. 2025.

WORLD WIDE WEB CONSORTIUM (W3C). **HTML5 – A vocabulary and associated APIs for HTML and XHTML.** 2014. Disponível em: https://www.w3.org/TR/html5/. Acesso em: 22 mar. 2025.

WORLD WIDE WEB CONSORTIUM (W3C). **CSS – Cascading Style Sheets Level 3. 2023.** Disponível em: https://www.w3.org/Style/CSS/. Acesso em: 22 mar. 2025.

OPENJS FOUNDATION. **Node.js.** 2025. Disponível em: https://nodejs.org. Acesso em: 22 mar. 2025.

FAUBEL, Michael. **The WebSocket Protocol (RFC 6455).** IETF, 2011. Disponível em: https://datatracker.ietf.org/doc/html/rfc6455. Acesso em: 22 mar. 2025.

ECMA INTERNATIONAL. Standard ECMA-404: **The JSON Data Interchange Format.** 2. ed. 2017. Disponível em: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/. Acesso em: 22 mar. 2025.

NETLIFY. **Netlify.** Disponível em: https://www.netlify.com/. Acesso em: 22 mar. 2025.

WANZELER, Tiago; FÜLBER, Heleno; MERLIN, Bruno. **Desenvolvimento de um sistema de automação residencial de baixo custo aliado ao conceito de Internet das Coisas (IoT).** XXXIV Simpósio Brasileiro de Telecomunicações – SBrT2016, Santarém, PA, 2016.

CAMARGOS, Ana Flávia Peixoto de et al. **Produto educacional: automação residencial com uso de Arduino e IoT.** Research, Society and Development, v. 11, n. 6, e8311628882, 2022. DOI: 10.33448/rsd-v11i6.28882.

KANDAUROFF, Andrey Viktor; MOLINA, Diego Luiz. **Sistema IoT de automação residencial via cloud e dispositivos móveis.** Trabalho de Conclusão de Curso (Engenharia Elétrica) – Universidade Federal do Paraná, Curitiba, 2019.

ROSSI, Augusto César dos Reis; SIRQUEIRA, Tassio Ferenzini Martins. **Integração de dispositivos IoT em automação residencial: desafios e soluções com protocolos heterogêneos.** Centro Universitário Academia – UniAcademia, Juiz de Fora, MG, 2022.

SANTOS, Diego; FERREIRA, Marcus Vinicius. **Simulação de sistemas de automação residencial via microcontroladores: uma proposta de arquitetura em ambientes educacionais.** Trabalho acadêmico, Universidade Federal do Pará, 2021.

SOARES, Lucas Silveira; CERQUEIRA, Jorsiele Damasceno. **Sistema de automação de baixo custo para domótica baseado em aplicação web.** Centro Universitário SENAI CIMATEC, 2022.

MESSIAS, Alan Fernandes. **Edifícios "inteligentes": a domótica aplicada à realidade brasileira.** Monografia (Graduação em Engenharia de Controle e Automação) – Universidade Federal de Ouro Preto, Ouro Preto, 2007.

STOJKOSKA, Biljana Risteska; TRIVODALIEV, Kire. **Enabling Internet of Things for Smart Homes Through Fog Computing.** In: 25th Telecommunications Forum (TELFOR), Belgrado, Sérvia. Anais [...]. IEEE, 2017. DOI: 10.1109/TELFOR.2017.8249316.