Universidade de Brasília – UnB
Campus Faculdade do Gama – FGA
Engenharia de Software

Ambiente Educacional de Acesso Remoto para Bancadas de Sistemas Embarcados

Autor: Igor Queiroz Lima
Orientador: Prof. Dr. Renato Coral Sampaio

Brasília, DF
2024

Igor Queiroz Lima

Ambiente Educacional de Acesso Remoto para Bancadas de Sistemas Embarcados

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Campus Faculdade do Gama – FGA

Orientador: Prof. Dr. Renato Coral Sampaio

Brasília, DF

2024

Igor Queiroz Lima
Ambiente Educacional de Acesso Remoto para Bancadas de Sistemas Embarcados/ Igor Queiroz Lima. – Brasília, DF, 2024-
70 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Renato Coral Sampaio

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Campus Faculdade do Gama – FGA , 2024.

1. Laboratório Remoto. 2. Sistemas Embarcados. I. Prof. Dr. Renato Coral Sampaio. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Ambiente Educacional de Acesso Remoto para Bancadas de Sistemas Embarcados

CDU 02:141:005.6

Igor Queiroz Lima

Ambiente Educacional de Acesso Remoto para Bancadas de Sistemas Embarcados

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 23 de setembro de 2024 – Data da aprovação do trabalho:

Prof. Dr. Renato Coral Sampaio
Orientador

Prof. Dr. Guillermo Alvarez Bestard
Convidado 1

Prof. Dr. John Lenon Cardoso Gardenghi
Convidado 2

Brasília, DF
2024

Resumo

No Campus Faculdade do Gama da Universidade de Brasília, os estudantes de Engenharia de Software enfrentam dificuldades ao tentar realizar práticas remotas em experimentos com dispositivos específicos da disciplina de Fundamentos de Sistemas Embarcados. A quantidade limitada de equipamentos disponíveis em relação ao número de alunos matriculados tem impulsionado a adoção de laboratórios de experimentação remota como uma solução, permitindo o acesso à distância a atividades práticas. Essa abordagem proporciona oportunidades adicionais de aprendizagem e promove uma educação mais prática e interativa, oferecendo maior flexibilidade no acesso aos equipamentos. Contudo, ainda há espaço para melhorias no sistema atual da disciplina. Com isso em mente, este trabalho tem como objetivo propor um novo Ambiente Educacional de Acesso Remoto para Bancadas de Sistemas Embarcados, visando aprimorar o formato atual e ampliando o acesso aos experimentos relacionados a sistemas embarcados para os alunos, ao mesmo tempo, em que proporciona aos professores um melhor controle dos recursos disponíveis.

Palavras-chave: Laboratório remoto. Sistemas embarcados.

Abstract

At the Gama Campus of the University of Brasília, Software Engineering students face challenges when conducting remote practices for the Fundamentals of Embedded Systems course using specific devices. The limited availability of equipment compared to the number of enrolled students has led to the adoption of remote experimentation laboratories as a solution, offering additional learning opportunities and promoting practical, interactive education with greater flexibility in equipment access. Despite these advancements, the current system presents room for improvement in several areas. Therefore, this paper proposes a new Educational Environment for Remote Access to Embedded Systems Benches, aiming to enhance the current system, expand access to embedded systems experiments for students, and provide teachers with better control over available resources.

Key-words: Remote laboratory. Embedded systems.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CDN	*Content Delivery Network*
ESP-IDF	*Espressif IoT Development Framework*
FGA	Faculdade do Gama
FPGA	*Field Programmable Gate Array*
FSE	Fundamentos de Sistemas Embarcados
GPIO	*General Purpose Input/Output*
GT-MRE	Grupo de Trabalho em Experimentação Remota
GUI	*Graphical User Interface*
HLS	*HTTP Live Streaming*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IHC	Interação Humano-Computador
LED	*Light-emitting diode*
MCU	*Microcontroller Unit*
OASIS	*Organization for the Advancement of Structured Information Standards*
OTA	*Over the Air*
P2P	*Peer-to-peer*
PD	Pesquisa e Desenvolvimento
RAL	Laboratório de Acesso Remoto
RExLab	Laboratório de Experimentação Remota
RF	Requisito Funcional
RNF	Requisito Não Funcional

RNP	Rede Nacional de Ensino e Pesquisa
RTC	*Real-Time Communication*
RTMP	*Real-Time Messaging Protocol*
SDK	*Software Development Kit*
SSH	*Secure Shell*
TCC	Trabalho de Conclusão de Curso
UI	*User Interface*
UnB	Universidade de Brasília
USB	*Universal Serial Bus*
USQ	*University of Southern Queensland*

Sumário

1 Introdução

Este capítulo expõe as motivações que fundamentam a realização deste trabalho, contextualizando a relevância do tema abordado ao apontar os problemas ou lacunas existentes que justificam sua execução.

1.1 Contextualização

A sociedade atual está completamente imersa em uma cultura cada vez mais digital, na qual as mudanças ocorrem em um ritmo acelerado (SILVA, 2023). Com a rápida evolução das tecnologias de informação e comunicação, tem ocorrido uma mudança significativa na realidade dos estudantes universitários nas últimas décadas (SILVA, 2023). Um exemplo disso é a facilidade com que os professores podem disponibilizar materiais didáticos pela internet, utilizando diversas ferramentas (BALAMURALITHARA; WOODS, 2009).

No contexto de constante inovação tecnológica, surge a possibilidade, já adotada por algumas universidades ao redor do mundo, dos laboratórios de experimentação remota. Esses laboratórios são ferramentas computacionais que apoiam a educação e são bem fundamentadas na literatura (TULHA; ANTONIO; COLUCI, 2019). Trata-se também de uma área de pesquisa que busca ampliar os limites da capacidade humana por meio do uso de recursos tecnológicos, permitindo o acesso remoto e o compartilhamento de recursos de qualquer localização geográfica (REXLAB. . . , 2023). Isso possibilita uma transformação e melhoria na forma como as atividades experimentais em sala de aula podem ser realizadas e disponibilizadas aos alunos.

Diante desse panorama de transformações, as instituições de ensino superior precisam estar preparadas para explorar as oportunidades proporcionadas pela experimentação remota, a fim de promover uma educação mais flexível, acessível e eficiente. A utilização dessas tecnologias permite a expansão dos limites do ensino e da aprendizagem, possibilitando a realização de experimentos e práticas mesmo à distância.

1.2 Justificativa

Os estudantes dos cursos de Engenharia Eletrônica e Engenharia de Software do Campus Faculdade do Gama da Universidade de Brasília (FGA/UnB) realizam práticas em laboratórios que possuem *hardware* específico, normalmente disponível apenas durante os horários de aula e limitado a um número restrito de equipamentos menor que a quanti-

dade de alunos matriculados nas disciplinas, tendo a necessidade de separação da turma em grupos para utilização.

A implementação de laboratórios remotos pode ampliar a capacidade dos laboratórios convencionais, permitindo que os estudantes realizem experimentos em mais horários e locais, além de possibilitar o acesso a mais alunos (BOCHICCHIO; LONGO, 2009, tradução nossa).

Embora a implementação de laboratórios online já estivesse sendo explorada e estudada como área de pesquisa, como demonstrado no estudo de Tulha, Antonio e Coluci (2019) intitulado "Uso de Laboratórios Remotos na Educação a Distância no Brasil: uma revisão sistemática", a pandemia da COVID-19 em 2020 acelerou significativamente esse processo. As universidades tiveram que se adaptar rapidamente ao ensino remoto, enfrentando um novo modelo de aprendizado (WERTH et al., 2021). Durante esse período, professores e alunos da FGA buscaram ativamente soluções para compartilhar os recursos do laboratório remotamente. Estudos anteriores propuseram as seguintes soluções:

- O trabalho de Bonifácio (2022), intitulado "Laboratório Remoto Baseado em FPGA Aplicado nas Disciplinas de Prática de Eletrônica Digital 1 e 2 na Faculdade UnB Gama", aborda a distribuição do kit de desenvolvimento Basys 3 para os alunos das disciplinas Prática de Eletrônica Digital 1 e 2, lecionadas em conjunto com as disciplinas de Teoria de Eletrônica Digital 1 e 2. O autor desenvolveu uma plataforma de acesso remoto que permite trabalhar remotamente com o kit, com visualização em tempo real e interação com os componentes da Basys 3.

- O trabalho de Souza (2023), intitulado "Plataforma Educacional de Acesso Remoto para Bancadas de Controle de Sistemas Embarcados", iniciou a criação de um ambiente para acesso agendado a simulações da disciplina de Fundamentos de Sistemas Embarcados (FSE), foi uma iniciativa valiosa para a disciplina, mas ainda há etapas a serem concluídas para torná-lo plenamente utilizável.

Atualmente, a disciplina de Fundamentos de Sistemas Embarcados (FSE) do curso de Engenharia de *Software* já possui experimentos realizados remotamente, porém, é necessário aprimorar o controle de acesso à plataforma de interações e visualização dos experimentos. Além disso, a visualização em tempo real do experimento não é eficiente, ocorrendo com atraso consideravelmente prejudicial a depender do tipo de experimento.

Considerando o trabalho de Souza (2023) como uma iniciativa promissora para a disciplina de FSE, torna-se fundamental avançar no desenvolvimento desse projeto e concluir a Plataforma Educacional de Acesso Remoto para Bancadas de Controle de Sistemas Embarcados. Oferecendo aos alunos o acesso remoto aos *hardwares* dos experimentos de maneira mais eficiente, facilitando sua realização e promovendo maior flexibi-

lidade e disponibilidade dos recursos. Além disso, com futuros trabalhos com proposta de aprimoramento desta plataforma, outras disciplinas que compartilham a necessidade de experimentações práticas em laboratórios remotos poderão se beneficiar.

Outra necessidade importante é a criação de novos experimentos compatíveis com a plataforma em desenvolvimento, considerando sua aplicação em outras disciplinas futuramente. É importante ressaltar que esses experimentos são utilizados como atividades avaliativas, implicando na necessidade de atualização ou criação de novos experimentos a cada semestre. Portanto, simplificar o processo de criação desses novos experimentos é uma tarefa que será realizada como parte deste trabalho.

Dessa forma, diante das limitações atuais do ambiente de laboratório remoto, da demanda por novos experimentos, justifica-se a realização deste trabalho para concluir o projeto existente, aprimorar a plataforma e facilitar a criação e disponibilização de experimentos.

1.3 Objetivos

1.3.1 Objetivos Gerais

O objetivo deste trabalho é concluir o desenvolvimento do projeto iniciado por Souza (2023) e aprimorar o ambiente de laboratório remoto para a disciplina de FSE, proporcionando aos alunos da disciplina de Fundamentos de Sistemas Embarcados o acesso remoto controlado aos *hardwares* utilizados nos experimentos, ao mesmo tempo, em que se busca facilitar a configuração e a criação de novos experimentos para a disciplina.

1.3.2 Objetivos Específicos

- Concluir o desenvolvimento da plataforma, a fim de possibilitar o agendamento e acesso ao painel interativo utilizado na matéria de sistemas embarcados.
- Desenvolver a disponibilização de vídeo em tempo real para a plataforma.
- Desenvolver código base para os microcontroladores utilizados para interatividade com o painel do experimento, a fim de dar mais agilidade para criação de novos experimentos.

1.4 Estrutura do Documento

Este documento está organizado em seis capítulos, conforme descrito a seguir:

- Capítulo 1 - Introdução: Apresenta uma breve introdução do contexto e das motivações que levaram à realização deste trabalho;
- Capítulo 2 - Fundamentação Teórica: Explora os conceitos teóricos necessários para a compreensão do trabalho a ser desenvolvido;
- Capítulo 3 - Metodologia: Descreve como o projeto será desenvolvido.
- Capítulo 4 - Desenvolvimento: Descreve o processo de execução da proposta, apresentando os resultados obtidos ao longo do desenvolvimento.
- Capítulo 5 - Considerações Finais: Resume os resultados alcançados e sugere direções para futuras melhorias e pesquisas.

2 Fundamentação Teórica

Neste capítulo, serão apresentados os conceitos fundamentais que embasam este estudo, esclarecendo as bases teóricas essenciais para uma compreensão adequada da elaboração do trabalho.

2.1 Laboratórios on-line

No contexto da educação e do ensino de ciências e engenharia, os laboratórios online têm se tornado uma ferramenta cada vez mais utilizada e consolidada. No entanto, é importante esclarecer algumas denominações semelhantes que podem causar confusão. De acordo com Balamuralithara e Woods (2009), os laboratórios online podem ser abordados de duas maneiras principais: laboratórios virtuais e laboratórios remotos, o que será discutido nas próximas seções. Além disso, também é possível encontrar laboratórios *online* híbridos que combinam as duas abordagens.

2.1.1 Laboratórios Virtuais

Os laboratórios virtuais são baseados em simulações, reproduzindo apenas uma parte da realidade e mantendo uma correlação constante entre entrada e saída de dados (TULHA; ANTONIO; COLUCI, 2019). Esse tipo de laboratório permite adaptar a realidade, os experimentos virtuais podem simplificar a aprendizagem destacando informações relevantes e removendo detalhes confusos, ou podem modificar características do modelo, como a escala de tempo, para facilitar a interpretação de certos fenômenos observáveis ou não, como reações químicas, termodinâmica ou eletricidade.(JONG; LINN; ZACHARIA, 2013).

2.1.2 Laboratórios Remotos

Os laboratórios remotos são experimentos reais influenciados por diversos fatores, como variações de temperatura e pressão no ambiente onde estão localizados (TULHA; ANTONIO; COLUCI, 2019). É nesse contexto que o presente trabalho se insere como parte de sua abordagem.

2.1.3 *Cases* de Laboratórios On-line

Hoje já é possível realizar experimentos reais em diversos campos científicos e tecnológicos. Plataformas como Relle, LabsLand e WebLab-Deusto contam com expe-

rimentos gratuitos reais que podem ser acessados remotamente pelas suas plataformas Web.

2.1.3.1 Relle

O Relle é um laboratório remoto desenvolvido pelo Grupo de Trabalho em Experimentação Remota (GT-MRE) do Laboratório de Experimentação Remota (RExLab), na Universidade Federal de Santa Catarina. O GT-MRE é um projeto financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e pela Rede Nacional de Ensino e Pesquisa (RNP), através dos Programas de P&D Temáticos. Oferece uma ampla gama de experimentos em ciências e engenharia. Seu objetivo é proporcionar aos usuários a oportunidade de conduzir experimentos em tempo real, utilizando equipamentos sofisticados e de última geração. Com uma interface intuitiva e recursos interativos, o Relle oferece uma experiência envolvente e enriquecedora para estudantes e pesquisadores (RELLE. . . , 2023).

2.1.3.2 Laboratório de Acesso Remoto da Universidade do Sul de Queensland

A Universidade do Sul de Queensland (USQ) na Austrália utiliza laboratórios de acesso remoto para permitir que estudantes externos realizem experimentos práticos remotamente, sistema de Laboratório de Acesso Remoto (RAL) oferece aos alunos, comunidades e indústria acesso externo a experimentos práticos e de laboratório — preenchendo a lacuna entre os espaços de aprendizado virtuais e da vida real (DICKMANN, 2013).

O RAL permite que os alunos, independentemente da localização, se envolvam ativamente no aprendizado orientado à ação contextual e alcancem os objetivos do curso com menos ênfase em participar de sessões de treinamento no campus (DICKMANN, 2013).

2.1.3.3 WebLab-Deusto

WebLab-Deusto situado na Espanha e fundado nos anos 2000 na Universidade de Deusto é uma equipe de pesquisa multidisciplinar com o objetivo de aumentar a aprendizagem experimental por meio do uso e desenvolvimento de laboratórios remotos. Para isso, vários laboratórios são oferecidos gratuitamente pela Internet, o software subjacente está disponível sob uma licença de código aberto e o equipamento pode ser replicado (WEBLAB-DEUSTO. . . , 2023, tradução nossa).

A história do projeto é marcada por um longo histórico de sucesso, oferecendo uma ampla gama de laboratórios acessíveis pela internet. Comprometidos em ampliar a aprendizagem experimental, eles desenvolveram uma infraestrutura sólida que permite que estudantes e pesquisadores conduzam experimentos reais em diversas áreas.

2.2 Sistemas Embarcados

Os *sistemas embarcados* são caracterizados pelo uso de hardware (eletrônica) e *firmware*[1] (software) incorporados em um dispositivo com um objetivo pré-definido. Eles estão presentes em diversos aspectos do nosso cotidiano, desde eletrodomésticos inteligentes, como geladeiras e máquinas de lavar, até sistemas complexos em automóveis, aviões e equipamentos médicos (PONTES, 2017).

A principal distinção entre um sistema embarcado e um computador de propósito geral está em sua finalidade. Enquanto os computadores e dispositivos similares são projetados para atender a uma ampla gama de funções, os sistemas embarcados são dimensionados com recursos direcionados a um domínio de objetivos mais limitado, muitas vezes focando em uma única função específica. Essa característica permite uma maior otimização e eficiência, pois os recursos do sistema são projetados para atender de forma precisa e eficaz às necessidades específicas do dispositivo (PONTES, 2017).

Os sistemas embarcados também são caracterizados por suas restrições de recursos, como memória limitada, capacidade de processamento restrita e disponibilidade limitada de energia. Essas restrições tornam esses sistemas diferentes de computadores de propósito geral e exigem uma abordagem cuidadosa no projeto e desenvolvimento do software embarcado. É necessário otimizar o uso dos recursos disponíveis e garantir que o dispositivo funcione corretamente dentro dessas restrições. Para isso, é fundamental que o *hardware* e o software sejam projetados em conjunto, de modo a garantir uma implementação eficiente que atenda aos objetivos de desempenho, custo e confiabilidade. Dessa forma, é possível criar sistemas embarcados eficazes, que ofereçam as funcionalidades desejadas e operem de forma confiável em seu ambiente específico (WOLF, 1994).

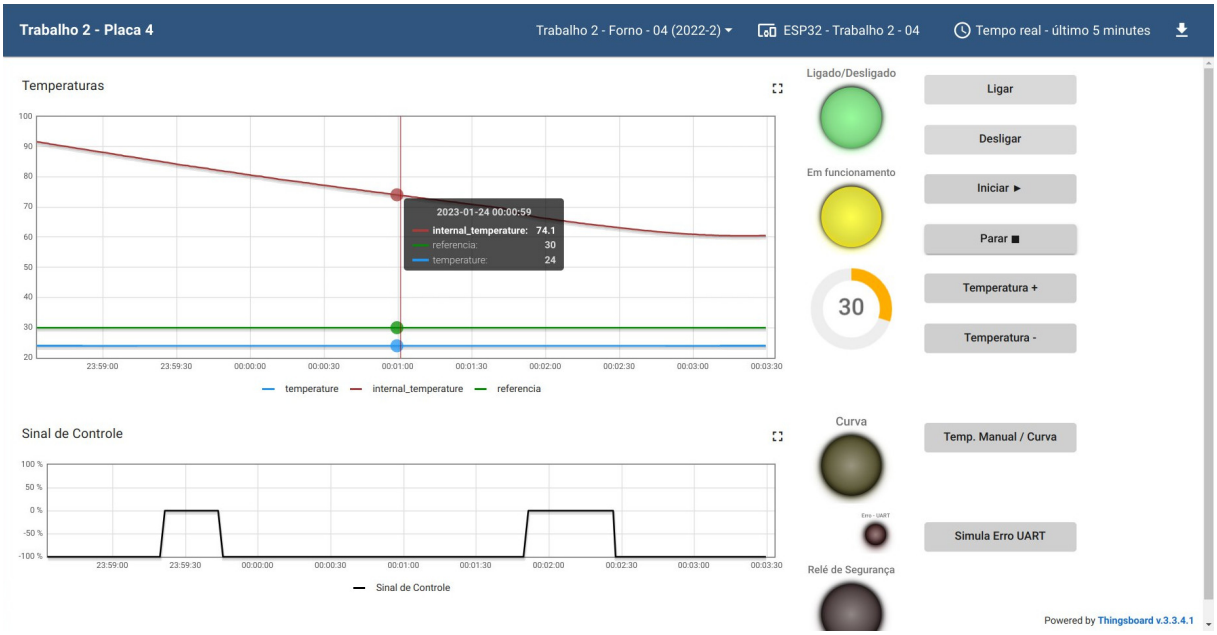
2.3 Disciplina de Fundamentos de Sistemas Embarcados

A disciplina de Fundamentos de Sistemas Embarcados (FSE) é oferecida aos alunos de engenharia do Campus Faculdade do Gama da Universidade de Brasília (FGA/UnB), os estudantes têm a oportunidade de aprender os conceitos fundamentais por trás dos sistemas embarcados e adquirir habilidades práticas no desenvolvimento de software para esses dispositivos. As aulas, ministradas duas vezes por semana com uma duração de uma hora e cinquenta minutos, visa proporcionar uma abordagem equilibrada entre a teoria e a prática, permitindo aos alunos compreenderem os princípios subjacentes e aplicá-los na resolução de problemas reais.

Durante as aulas, são propostos trabalhos práticos como método de avaliação, com enfoque especial nos dispositivos Raspberry Pi e ESP32. Nas próximas seções, será explicado sobre esses dispositivos.

Na disciplina, conta com um ambiente remoto para experimentação, que utiliza uma plataforma chamada ThingsBoard, detalhado na Seção 2.5.2. Essa plataforma é utilizada para gerenciar dispositivos de IoT. Através do ThingsBoard, os alunos têm acesso a um painel interativo como na Figura 1, conhecido como *dashboard*, onde podem visualizar informações e controlar componentes dos experimentos.

Figura 1 – Dashboard do ThingsBoard

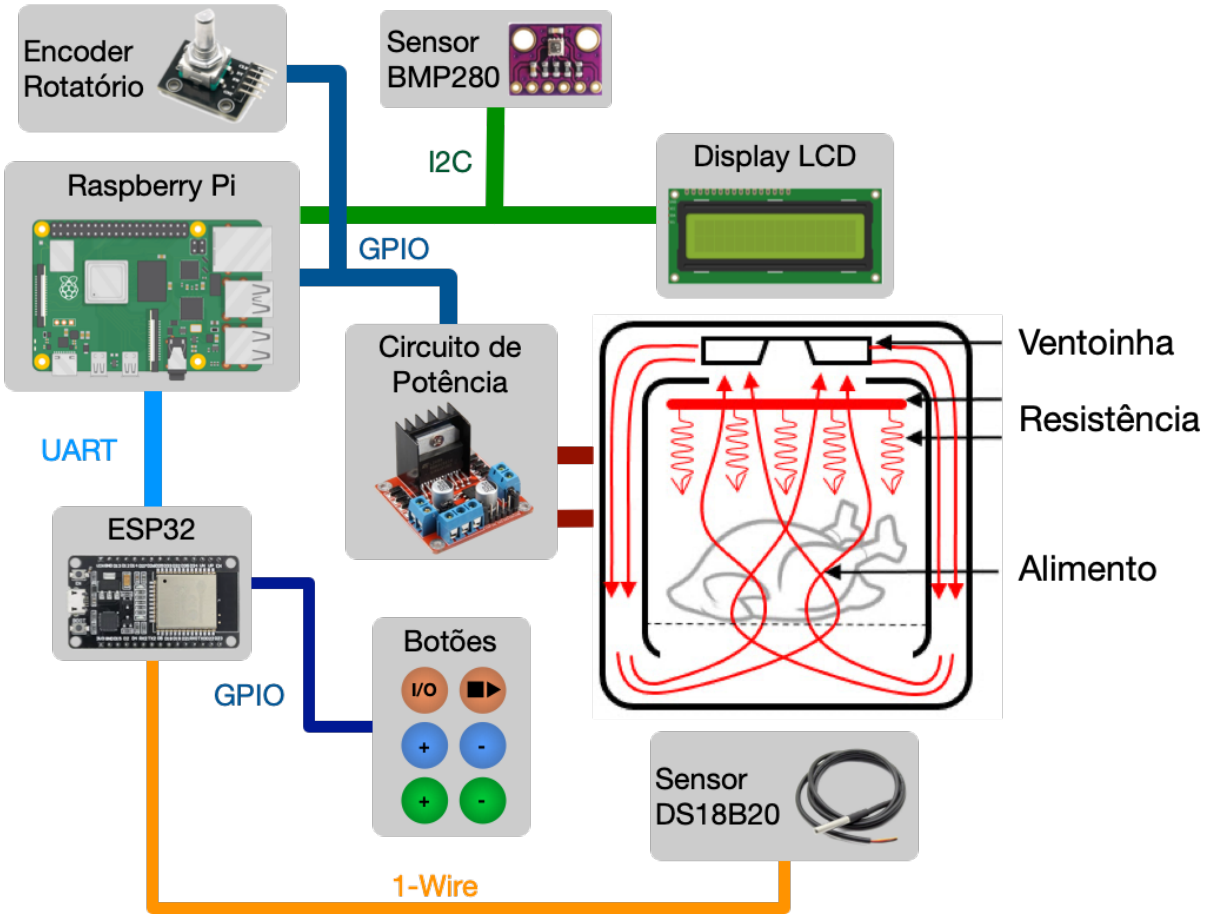


Fonte: autor

No *dashboard*, é possível ver o estado de alguns componentes, como botões, LEDs e sensores, e realizar ações relacionadas a eles. Isso é feito utilizando uma ESP32, que simula esses componentes do experimento e se comunica com o ThingsBoard. O objetivo é garantir que o *dashboard* funcione corretamente e mostre as informações do experimento, por exemplo, se uma lâmpada está ligada ou desligada, na Figura 2 podemos visualizar como são interligados componentes que fazem parte de um trabalho prático na matéria, o exemplo da figura trata-se de um sistema para *Air Fryer*.

Os alunos têm a possibilidade de programar a Raspberry Pi remotamente usando o SSH (*Secure Shell)* e interagir com eles por meio do *dashboard* do ThingsBoard. Além disso, a visualização de alguns componentes, como Displays LCD, é efetuada através da plataforma YouTube. Isso permite que os alunos acompanhem e visualizem os resultados dos experimentos.

Figura 2 – Composição do trabalho 2 da disciplina de FSE (2023/1)



Fonte: retirado do repositório da disciplina de FSE[2]

2.4 Hardwares

2.4.1 Raspberry Pi

O primeiro modelo de Raspberry Pi surgiu em 2012 e desde então cresceu o interesse por computadores *single-board*[3], por parte da engenharia e ciência da computação (ARIZA; BAEZ, 2022). O Raspberry Pi é uma placa de computador de baixo custo, pequena e portátil. Pode ser conectado a um monitor de computador ou televisão, teclado, mouse, pen drive, entre outros dispositivos como em um computador normal (ZHAO; JEGATHEESAN; LOON, 2015).

Uma das características poderosas destacadas na documentação de *hardware* da Raspberry Pi[4] é a fileira de pinos GPIO (entrada/saída de propósito geral) ao longo da borda superior da placa, conforme ilustrado na Figura 3. Todos os modelos atuais da

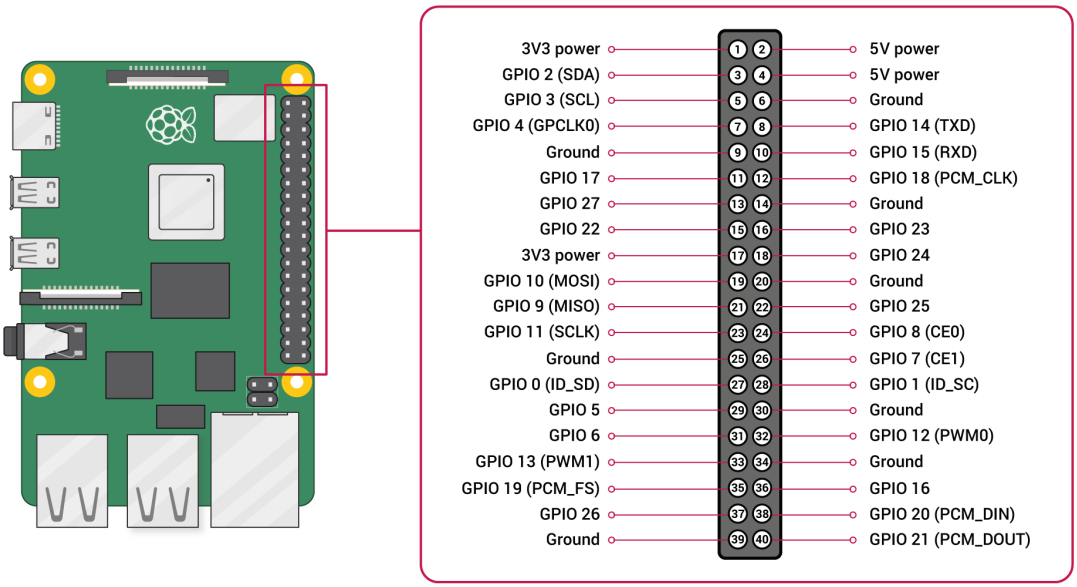
[2] Disponível em: <https://gitlab.com/fse_fga/trabalhos-2023_1/trabalho-2-2023-1>. Acesso em: 10 de jul. 2023.

[3] Computadores *single-board* são computadores em que seus componentes principais como memória e processador estão integrados em única placa (ARIZA; BAEZ, 2022).

[4] Disponível em: <https://www.raspberrypi.com/documentation>. Acesso em: 10 jul. 2023.

Raspberry Pi possuem um cabeçalho GPIO de 40 pinos. Esses pinos permitem interações com componentes em aplicações que abrangem robótica e Internet das Coisas (IoT)[5] (ARIZA; BAEZ, 2022).

Figura 3 – Raspberry Pi GPIOs



Fonte: retirado de Raspberry... (2023)

No geral, a Raspberry Pi é uma poderosa ferramenta para aprendizado, prototipagem e criação de projetos de eletrônica e computação, com uma ampla comunidade de suporte e recursos disponíveis.

2.4.2 Arduino

O Arduino é uma plataforma de *hardware* de código aberto que desempenhou um papel central na revolução no movimento *maker* e no desenvolvimento de *hardware* de código aberto. Trata-se de uma placa-mãe projetada para permitir que entusiastas, estudantes e profissionais desenvolvam projetos eletrônicos interativos de maneira acessível e fácil. A comunidade Arduino oferece um vasto conjunto de recursos, como placas de prototipagem, software de programação e documentação, todos disponíveis gratuitamente (BANZI, 2012).

Conforme descrito na página "O que é Arduino?"[6], trata-se de uma placa que oferece a capacidade de controlar dispositivos através do envio de instruções para o seu

[5] A IoT pode ser definida como a interconexão entre pessoas, animais ou objetos, com a capacidade de trocar dados por meio de uma rede sem a necessidade de interação entre humanos ou entre humanos e computadores (ZHAO; JEGATHEESAN; LOON, 2015).

[6] Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 10 de jul. 2023.

microcontrolador. A programação é realizada utilizando a linguagem de programação específica (baseada em Wiring[7]), juntamente com o software Arduino IDE. Essa plataforma tem sido amplamente adotada em uma variedade de domínios, desde projetos educacionais até projetos industriais. Com a acessibilidade e a flexibilidade, as pessoas podem criar uma ampla gama de dispositivos, como robôs, dispositivos de automação residencial, instrumentos musicais eletrônicos e muito mais.

Figura 4 – Arduino e Ethernet Shield

Fonte: retirado e adaptado de Arduino... (2022)

No entanto, embora seja adequado para algumas aplicações, o desenvolvimento de aplicações modernas para dispositivos conectados à Internet (Internet das Coisas, IoT) e comunicação Bluetooth pode ser desafiador, devido à necessidade de acessórios extras, conhecidos como *Shields* que pode ser visualizado um exemplar na Figura 4, que podem aumentar os custos do projeto (CARDOSO et al., 2020).

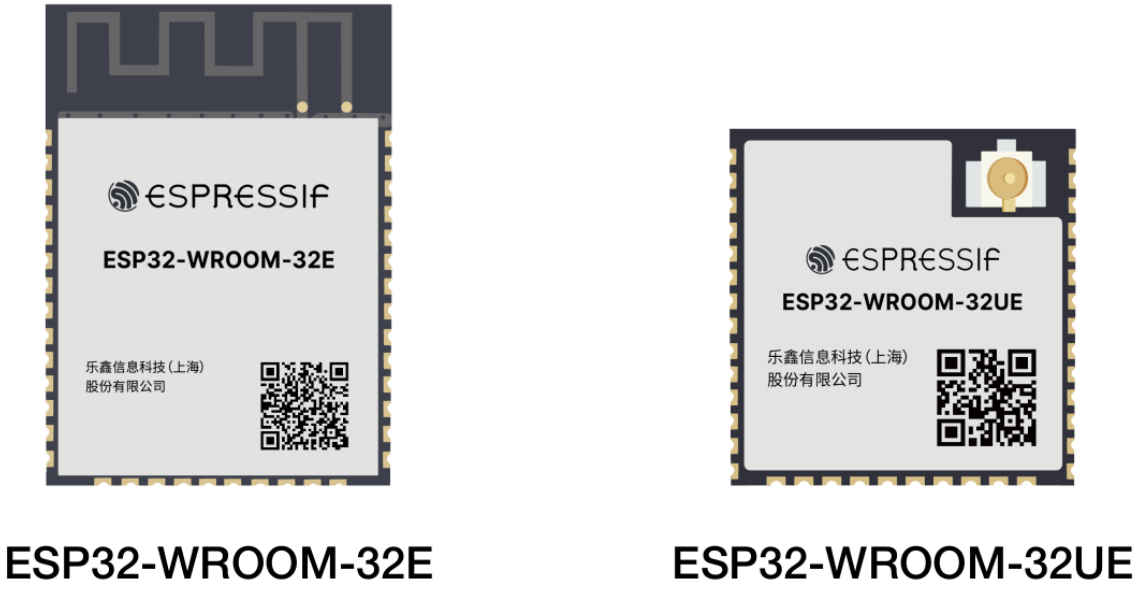
2.4.3 ESP32

Segundo as especificações da fabricante Espressif... (2023), o ESP32 é um circuito integrado dedicado ao desenvolvimento de aplicações portáteis e dispositivos IoT. O fabricante disponibiliza várias séries de módulos ESP32 programáveis, na Figura 5 é possível observar dois módulos, utilizados no kit de desenvolvimento ESP32-DevKitC que pode ser visualizado na Figura 6, onde é possível observar inserção do módulo na placa do kit.

[7] O *Wiring* é um *framework* de programação de código aberto para microcontroladores. O *Wiring* permite escrever software multiplataforma para controlar dispositivos conectados a uma ampla variedade de placas de microcontroladores, criando todo tipo de programação criativa, objetos interativos, espaços ou experiências físicas (WIRING, 2023).

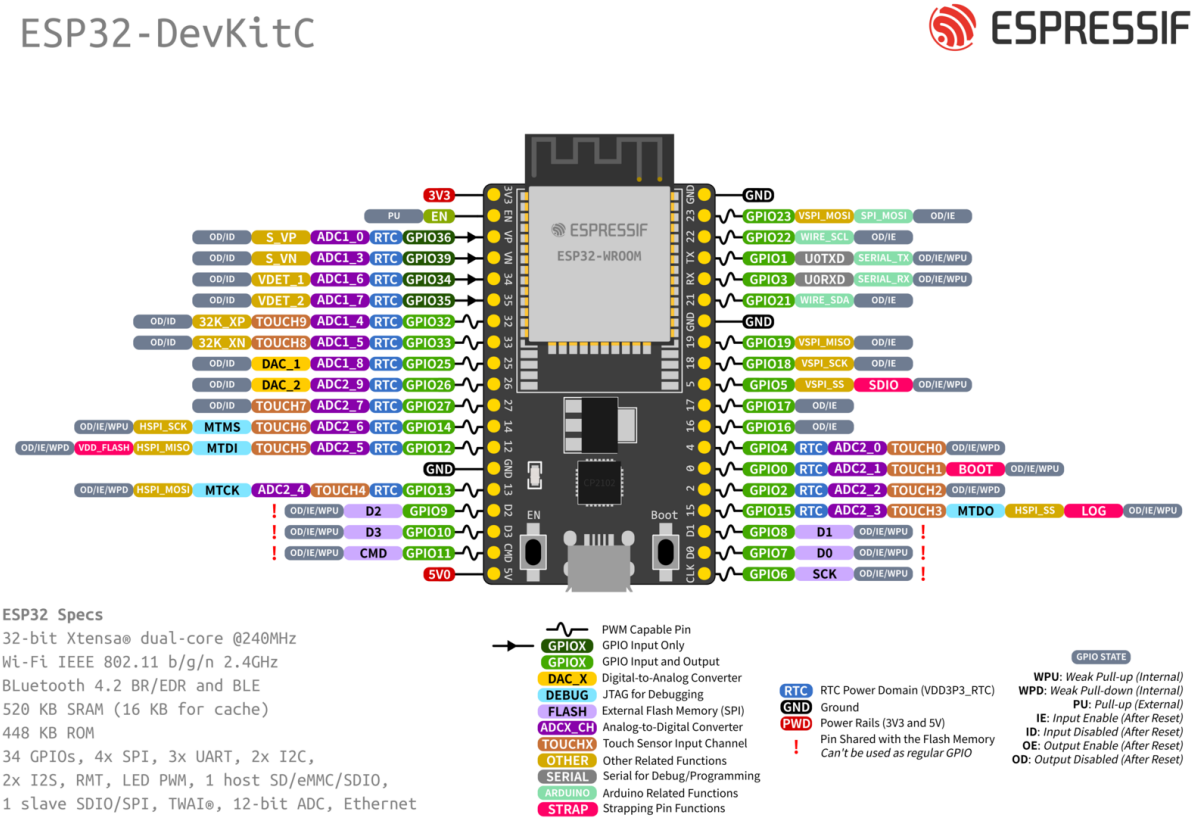
[8] Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf>. Acesso em: 5 de jul. 2023.

Figura 5 – Módulos ESP32



Fonte: retirado do *datasheet* dos módulos[8]

Figura 6 – Especificação e pinagem da ESP32-DevKitC



Fonte: retirado de Espressif... (2023)

Na disciplina de FSE, é utilizado a placa ESP32-DevKitC. Esse modelo é projetado de forma que todos os pinos do módulo ESP32 estejam expostos, permitindo um acesso e conexão fáceis com outros componentes eletrônicos. Além disso, a placa já possui os recursos básicos necessários para começar a usá-la. Basta conectar o cabo USB à placa e você está pronto para começar! Não é necessário adicionar componentes extras ou configurar de forma complexa. Essa facilidade simplifica o processo de aprendizado e possibilita a realização de experimentos e testes de maneira ágil e eficiente.

De acordo com Espressif... (2023), o ESP32 é um módulo MCU (Unidade de Microcontrolador, do inglês *Microcontroller Unit*) com recursos de Wi-Fi, Bluetooth e Bluetooth LE. Esses módulos têm como alvo uma ampla variedade de aplicações, desde redes de sensores de baixo consumo de energia até tarefas mais exigentes, como codificação de voz, streaming de música e decodificação de MP3 (ESPRESSIF..., 2023).

As especificações também destacam a possibilidade de suporte a atualizações seguras (criptografadas) por meio do ar (OTA, do inglês *Over-the-Air*), permitindo que os usuários atualizem seus produtos mesmo após o lançamento, com um custo mínimo e pouco esforço.

Embora as placas ESP32 possam ser utilizadas sem um sistema operacional (*bare-metal*), a abordagem mais conveniente e eficaz para programá-las é por meio do FreeRTOS, um sistema operacional de tempo real de baixo nível para dispositivos embarcados. Para a comunicação com a placa, é utilizada uma cadeia de ferramentas Python chamada ESP-IDF (MICHON et al., 2020).

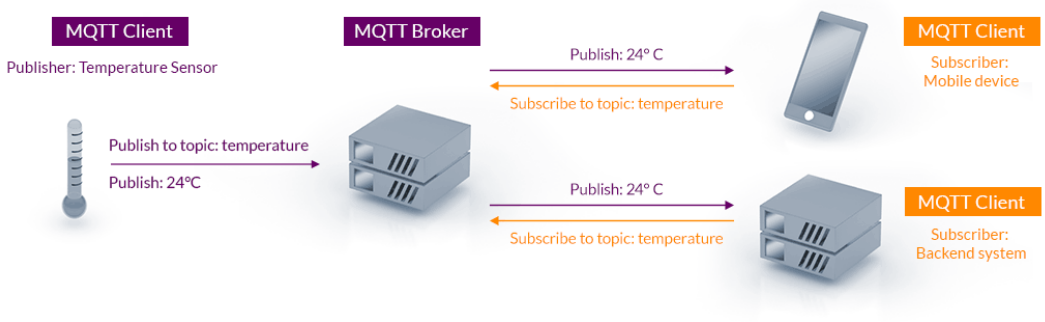
2.5 Softwares e Tecnologias

Nessa seção são abordados fundamentos dos softwares e Tecnologias utilizados pela disciplina de FSE já adotadas para a dinâmica da experimentação remota.

2.5.1 MQTT

O MQTT é um padrão OASIS (*Organization for the Advancement of Structured Information Standards*) para conectividade IoT. Trata-se de um protocolo de mensagens extremamente simples e leve, baseado em publicação/assinatura (*publish/subscribe*). Foi projetado para dispositivos com recursos limitados e redes com baixa largura de banda, alta latência ou pouca confiabilidade. Os princípios de design visam minimizar o uso de largura de banda da rede e os requisitos de recursos do dispositivo, ao mesmo tempo, em que buscam garantir confiabilidade e algum grau de garantia de entrega. Esses princípios também tornam o protocolo ideal para o mundo do Internet das Coisas, onde dispositivos estão conectados e para aplicativos móveis onde a largura de banda e a energia da bateria são valiosas (MQTT, 2022).

Figura 7 – Arquitetura MQTT Publish/Subscribe



Fonte: retirado de MQTT[9]

Um ambiente que utiliza comunicação MQTT segue uma arquitetura semelhante à mostrada na Figura 7. Nesse ambiente, dispositivos publicam dados para algum tópico. Esses dados são enviados para um servidor chamado de broker, com a função de rotear as mensagens publicadas para os assinantes. Eles se conectam ao broker e se inscrevem nos tópicos de seu interesse. Assim, quando uma nova mensagem é publicada no tópico, o broker encaminha essa mensagem para todos os assinantes correspondentes (HIVEMQ, 2015).

É importante ressaltar que os clientes do broker MQTT não são restritos a apenas a função de assinante ou publicador. Cada dispositivo pode atuar como publicador e/ou assinante, dependendo das necessidades de processamento de dados. Isso proporciona maior flexibilidade e adaptabilidade ao ambiente, permitindo que os dispositivos se comuniquem bidirecionalmente e troquem informações conforme necessário (HIVEMQ, 2015).

Essa arquitetura baseada em MQTT é amplamente utilizada em sistemas de Internet das Coisas (IoT) e outras aplicações que exigem uma comunicação eficiente e escalável entre dispositivos e servidores. Ela oferece uma solução robusta e flexível para troca de dados em tempo real (HIVEMQ, 2015).

2.5.2 ThingsBoard

ThingsBoard é uma plataforma de IoT de código aberto que permite o desenvolvimento e gerenciamento de projetos de IoT. Visando fornecer uma solução pronta para uso na nuvem ou local que viabilize a infraestrutura do lado do servidor para suas aplicações de IoT, o ThingsBoard oferece recursos como atualização de *firmware* de dispositivos, coleta e visualização de dados, análise de telemetria, controle remoto de dispositivos, criação de fluxos de trabalho, painéis dinâmicos e responsivos, recursos de cadeias de regras personalizáveis e integração com outros sistemas. Como plataforma de código aberto, o ThingsBoard beneficia-se de uma comunidade ativa de desenvolvedores, garantindo contí-

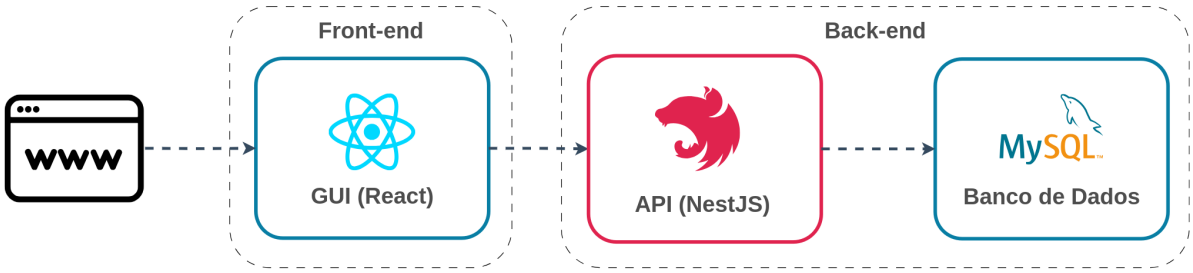
[9] Disponível em: <https://mqtt.org>. Acesso em: 5 de jul. 2023.

nua melhoria e suporte. No geral, o ThingsBoard simplifica e acelera o desenvolvimento e o gerenciamento de projetos de IoT e possui uma versão gratuita, chamada de ThingsBoard Community Edition (THINGSBOARD. . . , 2023).

2.5.3 Plataforma Educacional de Acesso Remoto para Bancadas de Controle de Sistemas Embarcados

A Plataforma Educacional de Acesso Remoto para Bancadas de Controle de Sistemas Embarcados, desenvolvida por Souza (2023), utiliza as tecnologias React, Material UI e NestJS. Essa plataforma web tem a proposta de permitir que alunos agendem e interajam remotamente com as bancadas dos experimentos propostos na disciplina de Fundamentos de Sistemas Embarcados (FSE).

Figura 8 – Diagrama de Arquitetura



Fonte: retirado e adaptado de Souza (2023)

O Diagrama de Arquitetura apresentado na Figura 8 ilustra o funcionamento do sistema como um todo. O front-end refere-se à interface gráfica do usuário (GUI), incluindo elementos de design e interação. É a parte visível com a qual os usuários interagem, como menus, botões e imagens. Já o back-end é responsável pela funcionalidade geral do aplicativo Web. Ele processa as solicitações enviadas pelo front-end e retorna as respostas adequadas. O back-end lida com o processamento e armazenamento de dados, além de executar a lógica de negócios (AWS, 2020).

2.5.3.1 React

As páginas de aplicativos da web devem carregar rapidamente para proporcionar uma boa experiência aos usuários. Com o React, é possível começar a mostrar o conteúdo da página mesmo enquanto os dados estão sendo buscados, preenchendo-o gradualmente antes mesmo do carregamento de qualquer código (JavaScript). Além disso, o React utiliza *APIs web* padrão para garantir que a interface do usuário permaneça responsiva mesmo durante o processo de renderização. Isso significa que os usuários podem interagir com a página sem atrasos, mesmo enquanto ela está sendo construída (REACT, 2023).

2.5.3.2 Material UI

Material UI é uma biblioteca de componentes React[10] de código aberto que oferece diversas vantagens para o desenvolvimento de interfaces. Contando com a contribuição de mais de 2.500 colaboradores que constantemente aprimoram seus componentes, acelera o processo ao permitir que você foque na lógica de negócios da aplicação. Além de ser visualmente padronizado, seguindo rigorosamente as diretrizes do Material Design (criado pelo Google) e oferecendo ampla opção de personalização. Essa ferramenta é amplamente confiável, utilizada por milhares de organizações e sustentada por uma comunidade ativa desde 2014 (MUI, 2024).

2.5.3.3 NestJS

O NestJS é um *framework* para construir aplicações no lado do servidor usando Node.js. Ele combina elementos de programação orientada a objetos, programação funcional e programação funcional reativa. O NestJS fornece uma arquitetura pronta para uso, inspirada pelo Angular, que permite criar aplicativos altamente testáveis, escaláveis e de fácil manutenção. Ele suporta TypeScript e pode ser integrado com os *frameworks* Express e Fastify. Com o NestJS, os desenvolvedores têm acesso a uma ampla gama de módulos de terceiros disponíveis para as plataformas subjacentes. O framework oferece uma solução abrangente para a arquitetura de aplicativos no lado do servidor, melhorando a produtividade e a confiabilidade do desenvolvimento (NESTJS, 2023).

2.5.3.4 WebRTC

Aplicativos baseados em Flash que utilizavam o Real-Time Messaging Protocol (RTMP) eram tradicionalmente eficazes para transmissões de baixa latência. No entanto, com a descontinuação do Flash e a redução gradual de suporte ou bloqueio do *plug-in* pelos navegadores, as Redes de Distribuição de Conteúdo (CDNs) também começaram a abandonar o RTMP, forçando os provedores de conteúdo a buscarem alternativas viáveis (AMAZON, 2024). Uma dessas alternativas é o WebRTC, uma estrutura que integra diversos protocolos bem estabelecidos, permitindo a transmissão em tempo real de vídeo, áudio e dados arbitrários por meio de conexões P2P (par-a-par, do inglês *peer-to-peer*), sem a necessidade de *plug-ins* (TANSKANEN, 2021). Embora o WebRTC ofereça a capacidade de transmitir vídeo com baixas latências, ele pode afetar a qualidade da transmissão ao vivo (TIDESTRÖM, 2019).

[10] Um componente React é uma unidade reutilizável de código que representa uma parte da interface de usuário (UI) (REACT, 2023).

3 Metodologia

3.1 Etapas do Desenvolvimento Teórico

O processo de desenvolvimento deste documento foi dividido em etapas que permitiram um avanço contínuo e organizado, garantindo uma evolução gradual do trabalho. Cada etapa foi essencial para o desenvolvimento coeso e estruturado deste trabalho de conclusão de curso.

3.1.1 Etapa 1: Definição do Escopo e Estrutura do Documento

Nesta etapa, foi realizada a definição do escopo do problema a ser abordado, bem como a organização da estrutura do documento. Os objetivos gerais e específicos foram estabelecidos, delimitando claramente o âmbito da pesquisa e suas metas. Essa etapa serviu como base para todo o desenvolvimento subsequente do trabalho.

3.1.2 Etapa 2: Pesquisa Bibliográfica

A segunda etapa consistiu na realização de uma pesquisa bibliográfica. O levantamento bibliográfico permitiu a identificação de conceitos-chave, teorias e trabalhos relacionados que contribuíram para fundamentar as abordagens e discussões presentes no trabalho. A revisão da literatura foi fundamental para embasar os argumentos e garantir a consistência das informações apresentadas.

3.1.3 Etapa 3: Definição da Proposta de Trabalho

Com uma base teórica estabelecida, a terceira etapa se concentrou na definição da proposta de trabalho. Nessa fase, foram delineados os métodos e procedimentos que serão adotados para alcançar os objetivos propostos. A definição da metodologia para condução adequada da pesquisa e a obtenção dos resultados esperados.

3.1.4 Etapa 4: Revisão

Após o desenvolvimento das etapas anteriores, o trabalho avançou para a etapa de revisão. Foram analisados em relação aos objetivos iniciais da pesquisa. Qualquer desvio ou ajuste necessário foi identificado e tratado nessa etapa para garantir a coerência e solidez do trabalho como um todo.

3.1.5 Etapa 5: Geração do Material de Apresentação

Por fim, com o trabalho revisado, foi gerado o material necessário para apresentação, para comunicar os principais aspectos da proposta à banca examinadora e demais interessados.

3.2 Etapas do desenvolvimento da plataforma web

Para dar continuidade a plataforma web, a arquitetura será mantida e a implementação da nova funcionalidade deverá ser compatível com as tecnologias que já foram previstas. Sobe este contexto, o de desenvolvimento da plataforma será composto por duas etapas, sendo a etapa de desenvolvimento e validação, seguindo o modelo de processo proposto na Seção 3.4.

3.2.1 Etapa de desenvolvimento

A etapa de desenvolvimento consistirá na implementação dos requisitos e funcionalidades definidos na proposta da plataforma web. Buscando sempre aplicar boas práticas de programação, design e arquitetura de *software* para garantir a qualidade, eficiência e escalabilidade do sistema.

3.2.2 Etapa de validação

Na etapa de validação será coletado o feedback dos alunos e professor que ministra a disciplina de Fundamento de Sistemas Embarcados (FSE), também os demais interessados, com o foco em identificar se a plataforma auxilia e trata o problema de acesso ao *dashboard* dos experimentos.

3.3 Etapas do desenvolvimento do código para ESP32

Para o desenvolvimento de uma base de código para geração de novos experimentos no laboratório e auxílio na atualização de *firmware* das ESP32, será feito em três etapas, sendo a etapa de definição da solução, desenvolvimento e validação.

3.3.1 Etapa de definição da solução

Nesta etapa, será realizada uma análise as funcionalidades desejadas para a geração de novos experimentos e a atualização do firmware das ESP32. Será feita uma pesquisa para identificar as melhores abordagens a serem utilizadas, garantindo a escolha de soluções eficientes e adequadas para o contexto do projeto.

3.3.2 Etapa de desenvolvimento

A etapa de desenvolvimento consistirá na implementação da estrutura de solução definida na etapa anterior. Buscando sempre aplicar boas práticas de programação, design e arquitetura de *software* para garantir a qualidade, eficiência e escalabilidade do sistema.

3.3.3 Etapa de validação

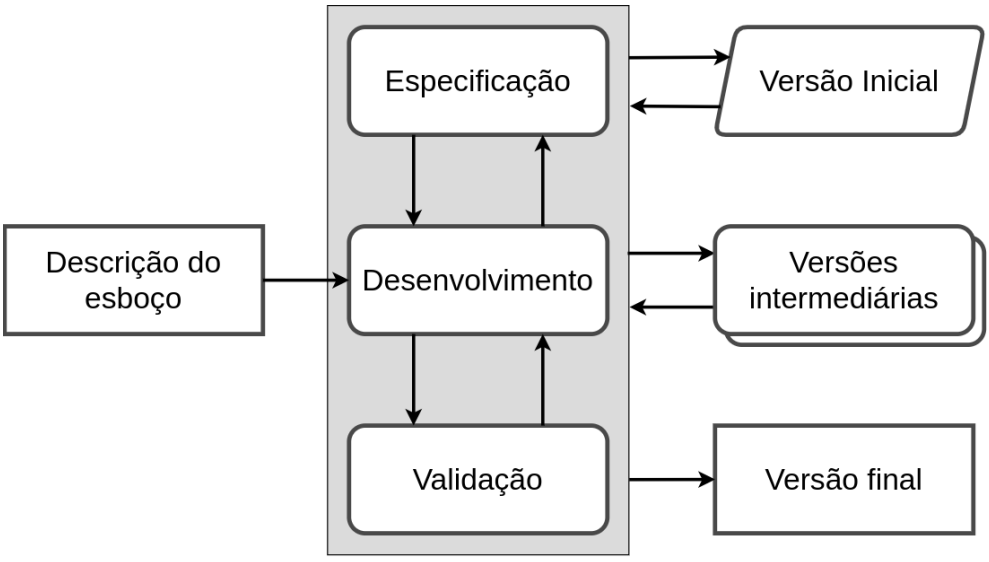
Na etapa de validação será coletado o feedback do professor que ministra a disciplina de Fundamento de Sistemas Embarcados (FSE), também os demais interessados, com o foco em identificar se a solução entrega facilitação relevante na geração de novos experimentos.

3.4 Modelo de Processo

Como plano de ação para a realização do trabalho proposto, será utilizado o modelo de processo de software incremental, especialmente considerando que o desenvolvimento será realizado por um único indivíduo e a natureza do projeto como trabalho de conclusão de curso.

A abordagem reflete nossa maneira cotidiana de resolver problemas, onde não buscamos uma solução completa antecipadamente, mas seguimos um caminho passo a passo, ajustando-nos conforme necessário. O modelo incremental baseia-se na ideia de desenvolver uma implementação inicial, submetê-la aos comentários dos usuários e, a partir disso, criar várias versões até alcançar um sistema adequado, conforme ilustrado na Figura 9 (SOMMERVILLE, 2011).

Figura 9 – Desenvolvimento Incremental



Fonte: retirado e adaptado de Sommerville (2011)

3.5 Requisitos

Requisitos são condições ou capacidades que um sistema deve atender ou possuir para cumprir um contrato, norma ou especificação. Esses requisitos podem ser classificados em funcionais (RF) e não funcionais (RNF). Um requisito funcional refere-se ao comportamento esperado do sistema, descrevendo os resultados que devem ser fornecidos por suas funções. Já um requisito não funcional aborda aspectos relacionados à qualidade do sistema, não contemplados pelos requisitos funcionais (POHL; RUPP, 2015).

A seguir estão os requisitos funcionais e não funcionais da plataforma web que devem ser atendidos:

- **RF01** O sistema deve permitir o cadastro de professores com acesso às funcionalidades relacionadas a turmas, experimentos e alunos. Isso inclui a criação de contas para os professores e a atribuição de permissões adequadas.
- **RF02** Os professores cadastrados devem poder manter turmas, informando detalhes como nome, horário e localização. Eles também devem conseguir associar turmas aos alunos e experimentos correspondentes.
- **RF03** Os professores cadastrados devem poder manter experimentos. Além disso, eles devem conseguir vincular esses experimentos às turmas correspondentes.
- **RF04** Os professores cadastrados devem poder manter alunos, fornecendo informações como nome, matrícula e turma a que pertencem. Isso permite que os alunos sejam associados às turmas adequadas.
- **RF05** Os alunos cadastrados devem poder agendar experimentos relacionados às suas turmas. Eles devem poder verificar a disponibilidade dos experimentos e escolher horários disponíveis para agendamento.
- **RF06** O sistema deve oferecer um recurso de streaming de vídeo ao vivo, permitindo que os usuários visualizem em tempo real os resultados dos experimentos que estão sendo realizados.
- **RF07** O sistema deve fornecer o painel do ThingsBoard embutido na plataforma, permitindo que os usuários interajam e acompanhem os experimentos em andamento.
- **RNF01** O sistema deve validar e informar erros nos campos dos formulários.
- **RNF02** O sistema deve incluir uma função de busca eficiente que permita aos usuários localizar rapidamente informações.

- **RNF03** O sistema deve fornecer feedback claro e imediato para todas as ações realizadas pelos usuários, incluindo mensagens de sucesso, erro e progresso.

A seguir estão os requisitos do código base para ESP32 que devem ser atendidos:

- **R01** O código deve permitir a conexão com a internet, garantindo que o dispositivo tenha acesso à rede para realizar as funcionalidades necessárias.
- **R02** O código deve permitir a comunicação com a plataforma ThingsBoard.
- **R03** O código deve suportar a atualização de firmware via OTA para dispositivos ESP32, permitindo a manutenção e melhoria do software remotamente, sem a necessidade de intervenção física no dispositivo.

3.6 Ferramentas

As ferramentas que serão utilizadas para realização do desenvolvimento para dar continuidade da plataforma Web desenvolvido por Souza (2023), será a biblioteca de interfaces de usuário web e nativas React, especificado anteriormente na Seção 2.5.3.1 juntamente com a biblioteca de componentes Material UI descrita na Seção 2.5.3.2. Em conjunto com o *framework* NestJS para construir aplicação no lado do servidor, especificado anteriormente na Seção 2.5.3.3 (NESTJS, 2023).

Para o desenvolvimento do código base para os microcontroladores ESP32 será utilizado o *framework* ESP-IDF (Framework de Desenvolvimento IoT da Espressif, do inglês Espressif IoT Development Framework), a qual é destinado à criação de aplicativos para a Internet das Coisas (IoT) com Wi-Fi, Bluetooth, gerenciamento de energia e várias outras funcionalidades do *hardware* da série ESP32 (ESPRESSIF. . . , 2023).

Os resultados do desenvolvimento de código para a ESP32, bem como da plataforma web, serão disponibilizados no Gitlab. O Gitlab é uma plataforma de código-aberto amplamente reconhecida e utilizada pela comunidade de desenvolvedores como um repositório de versionamento de código, permitindo o compartilhamento e rastreamento das alterações realizadas ao longo do tempo de maneira transparente aos interessados.

4 Desenvolvimento

O capítulo a seguir abordará o processo e os principais aspectos da solução desenvolvida. Conforme definido por Sommerville (2011, p. 3), a solução vai além do simples programa de computador:

> Muitas pessoas pensam que software é simplesmente outra palavra para programas de computador. No entanto, quando falamos de engenharia de software, não estamos nos referindo apenas ao programa em si, mas a toda a documentação associada e aos dados de configuração necessários para fazer esse programa operar corretamente. Um sistema de software desenvolvido profissionalmente é, com frequência, mais do que apenas um programa; ele normalmente consiste em uma série de programas separados e arquivos de configuração que são usados para configurar esses programas. Isso pode incluir documentação do sistema, que descreve sua estrutura; documentação do usuário, que explica como usar o sistema; e sites, para usuários baixarem informações atualizadas sobre o produto.

Para acomodar todas essas necessidades e garantir uma gestão eficiente do desenvolvimento, é essencial escolher uma plataforma adequada para hospedar e gerenciar o código-fonte. Conforme descrito na Seção 3.6, o GitLab será utilizado para esse fim. A plataforma oferece uma estrutura para a criação e gerenciamento de vários repositórios, permitindo a divisão em seções e subseções, facilitando o acesso a todos os códigos de um mesmo projeto. Dessa forma, todos os códigos desenvolvidos neste trabalho podem ser acessados através de um único link em: <https://gitlab.com/igorq937-tcc>.

4.1 Desenvolvimento da plataforma web

O desenvolvimento da plataforma web envolveu uma série de aprimoramentos no back-end e no front-end, visando melhorar a segurança, a usabilidade e a eficiência da aplicação. No back-end, foram reforçadas as rotinas de autenticação para controlar o acesso aos dados e funcionalidades, além da implementação de novos modelos de regras de negócio. Essas melhorias garantem que apenas usuários autenticados possam acessar e modificar informações, aumentando a proteção contra acessos não autorizados.

Simultaneamente, o front-end recebeu melhorias significativas, incluindo a reformulação e criação de várias páginas para alinhar com as mudanças implementadas no back-end. Essas atualizações visam proporcionar uma interface mais intuitiva e eficiente para todos os usuários da plataforma, seguindo as novas diretrizes de design detalhadas na Seção 4.1.1.

4.1.1 Processo de design

Para a avaliação do design das páginas da aplicação web, foi utilizada a avaliação heurística, um método de IHC (Interação Humano-Computador) desenvolvido para identificar problemas de usabilidade durante o processo de design iterativo. Este método consiste na inspeção sistemática da interface para detectar problemas que possam comprometer a usabilidade (BARBOSA et al., 2021).

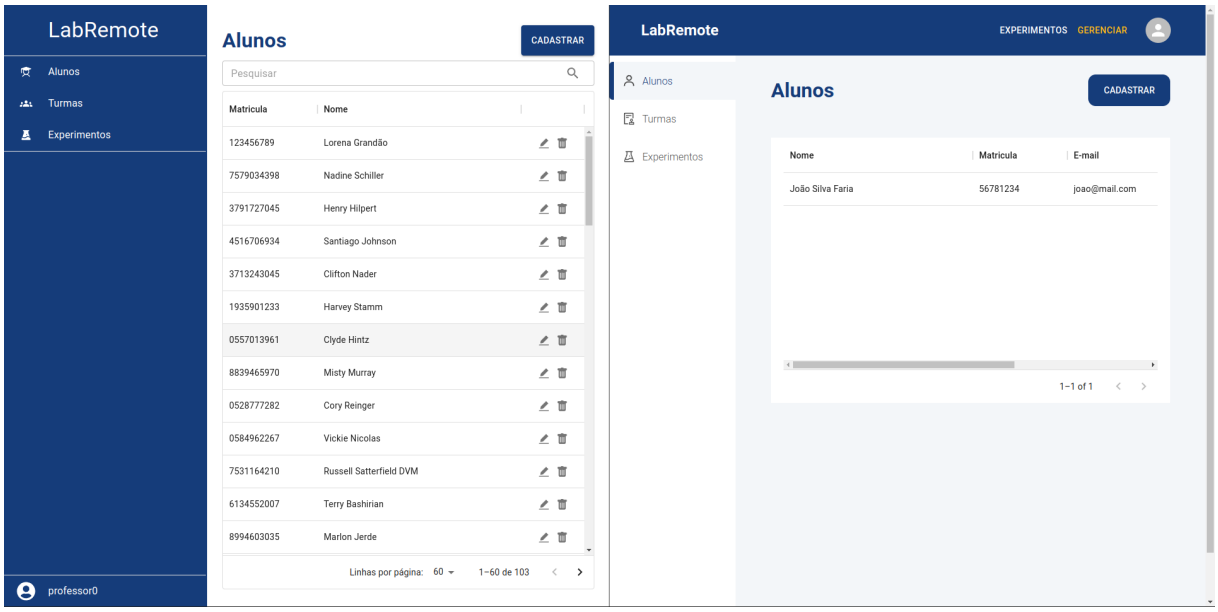
A avaliação heurística baseia-se em um conjunto de diretrizes de usabilidade, conhecidas como heurísticas, que descrevem características desejáveis tanto da interação quanto da interface. Essas heurísticas foram formuladas por Nielsen a partir da análise de mais de 240 problemas de usabilidade, conduzida ao longo de vários anos por especialistas experientes em IHC (NIELSEN, 1994 apud BARBOSA et al., 2021).

O método de avaliação heurística consiste em um conjunto inicial de dez heurísticas, que pode ser expandido para incluir novas diretrizes conforme necessário:

- **Visibilidade do estado do sistema:** o sistema deve informar o usuário sobre o que está acontecendo, por meio de feedback adequado e em tempo real.
- **Correspondência com o mundo real:** a interface deve usar linguagem familiar ao usuário, seguindo convenções e expectativas do mundo real.
- **Controle e liberdade do usuário:** o usuário deve ter formas claras de desfazer ações ou sair de estados indesejados.
- **Consistência e padronização:** elementos semelhantes devem ter significados e comportamentos consistentes em toda a interface.
- **Reconhecimento em vez de memorização:** a interface deve tornar visíveis as opções, ações e objetos, minimizando a necessidade de memorização.
- **Flexibilidade e eficiência de uso:** permitir atalhos e customizações para melhorar a eficiência dos usuários experientes sem prejudicar os iniciantes.
- **Projeto estético e minimalista:** evitar informações irrelevantes para não sobrecarregar o usuário e reduzir a complexidade visual.
- **Prevenção de erros:** projetar a interface para minimizar a ocorrência de erros antes que eles aconteçam.
- **Recuperação de erros:** fornecer mensagens de erro claras, que expliquem o problema e sugiram soluções.
- **Ajuda e documentação:** disponibilizar ajuda e documentação acessível, clara e focada nas tarefas do usuário.

Além disso, a biblioteca de componentes Material UI detalhada na Seção 2.5.3.2 foi adotada de acordo com as diretrizes do Material Design 3[1]. Todas as páginas da aplicação foram revisadas para maximizar o uso desses componentes, utilizando uma configuração de estilo padrão que assegura a consistência e a padronização da interface. Componentes customizados foram avaliados e sempre que possível, substituídos por alternativas disponíveis no Material UI. A Figura 10 ilustra o "antes e depois" da aplicação das diretrizes, mostrando telas mais responsivas, além de ter resultando na remoção de muitas linhas de estilização que agora são substituídas pelas propriedades padrão da biblioteca.

Figura 10 – Novo design à esquerda, antigo design à direita



Fonte: autor

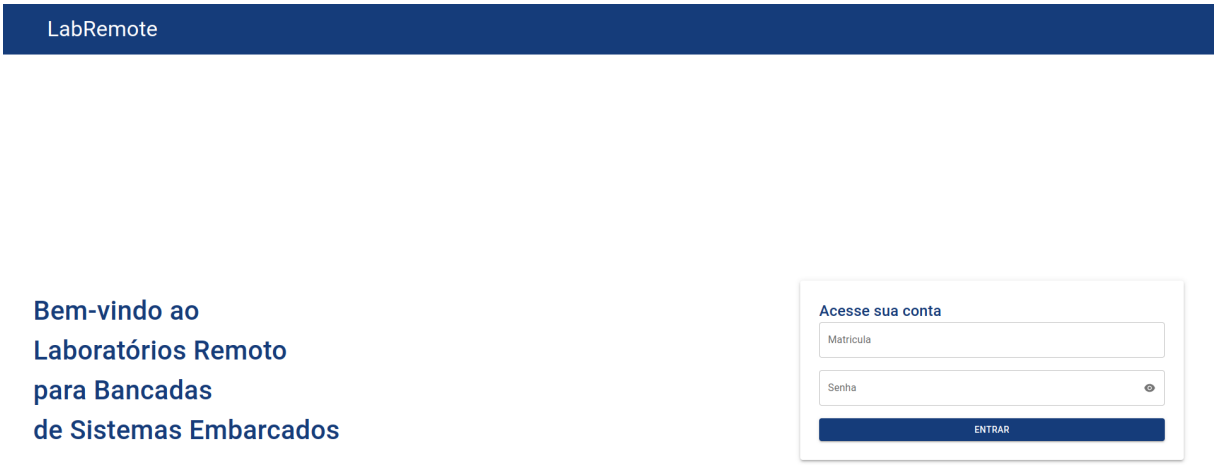
4.1.2 Telas da plataforma web

4.1.2.1 Tela de acesso

A tela de acesso é o ponto de entrada para todos os usuários da plataforma (usuário administrador, professores e alunos), que permite inserir suas credenciais para acessar a plataforma como pode ser visualizado na Figura 11. Para os alunos, o acesso só é permitido após o cadastro por parte do administrador ou de um professor designado, enquanto os professores devem ser cadastrados exclusivamente pelo administrador.

[1] Disponível em: <https://m3.material.io/>. Acesso em: 20 de mar. 2024.

Figura 11 – Tela de acesso

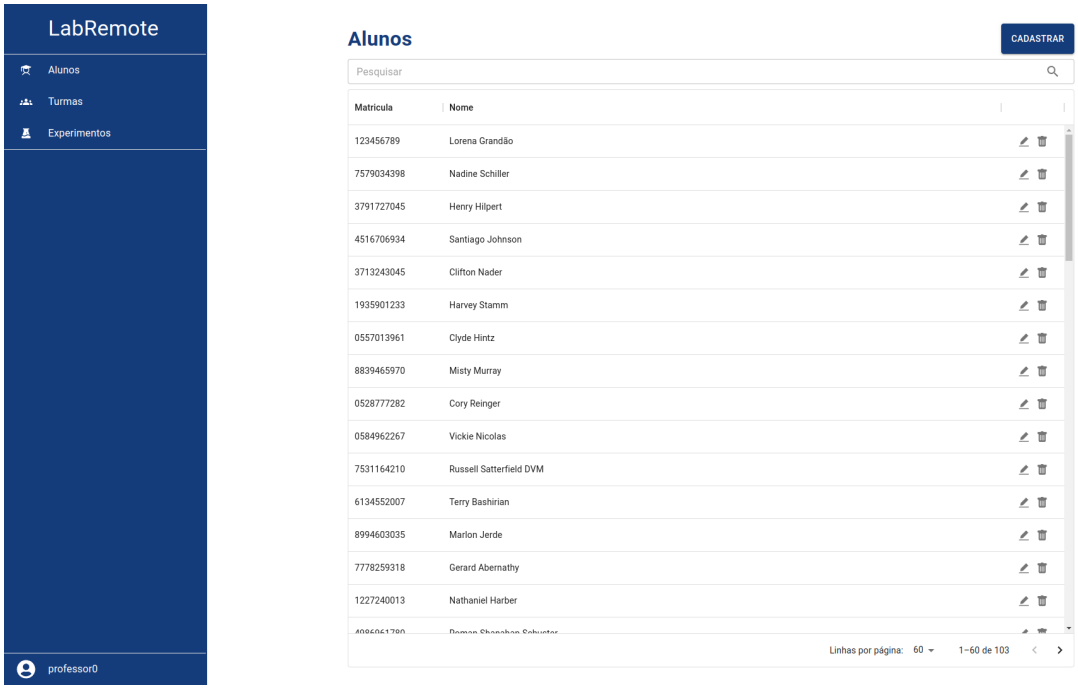


Fonte: autor

4.1.2.2 Tela de gerenciamento de alunos

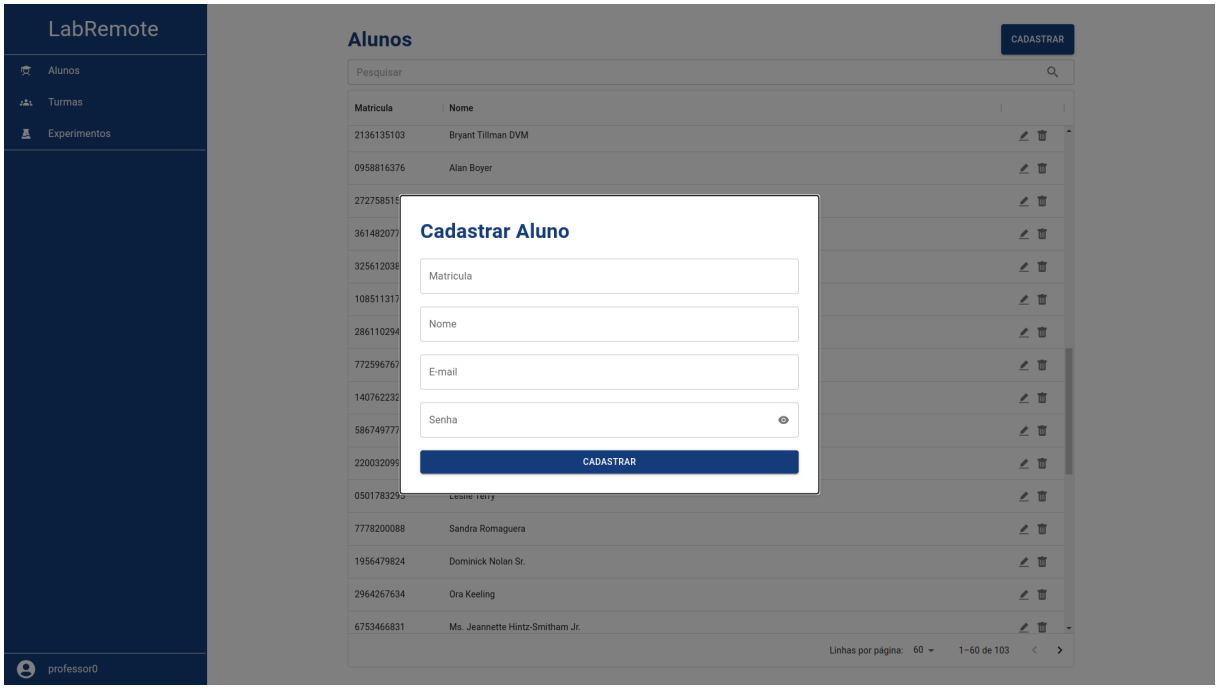
O administrador da plataforma e os professores podem gerenciar alunos na aplicação. A Figura 12 ilustra a tela de gerenciamento de alunos, onde é possível visualizar a lista de alunos cadastrados. A Figura 13 apresenta a tela de cadastro de novos alunos, permitindo a inserção de informações detalhadas de cada aluno. A Figura 14 mostra a tela de edição de alunos, que possibilita a atualização dos dados de alunos já cadastrados. A Figura 15 exibe a tela de confirmação de remoção de aluno.

Figura 12 – Tela de gerenciamento de alunos



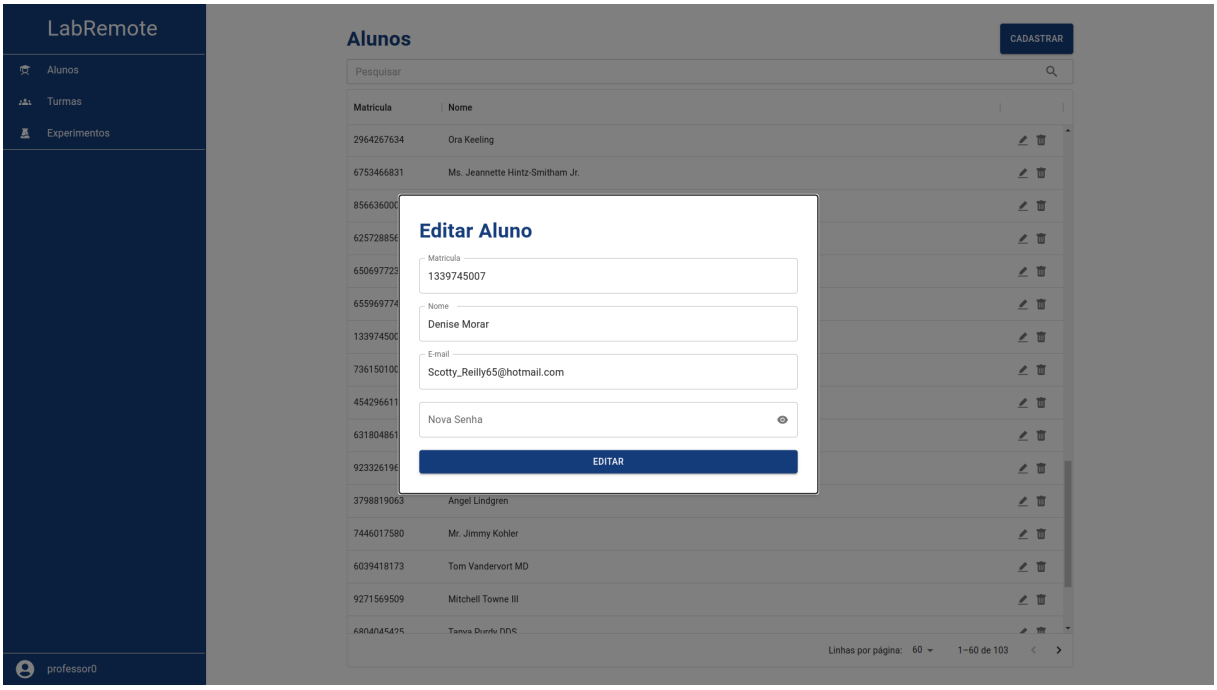
Fonte: autor

Figura 13 – Tela de cadastro de aluno



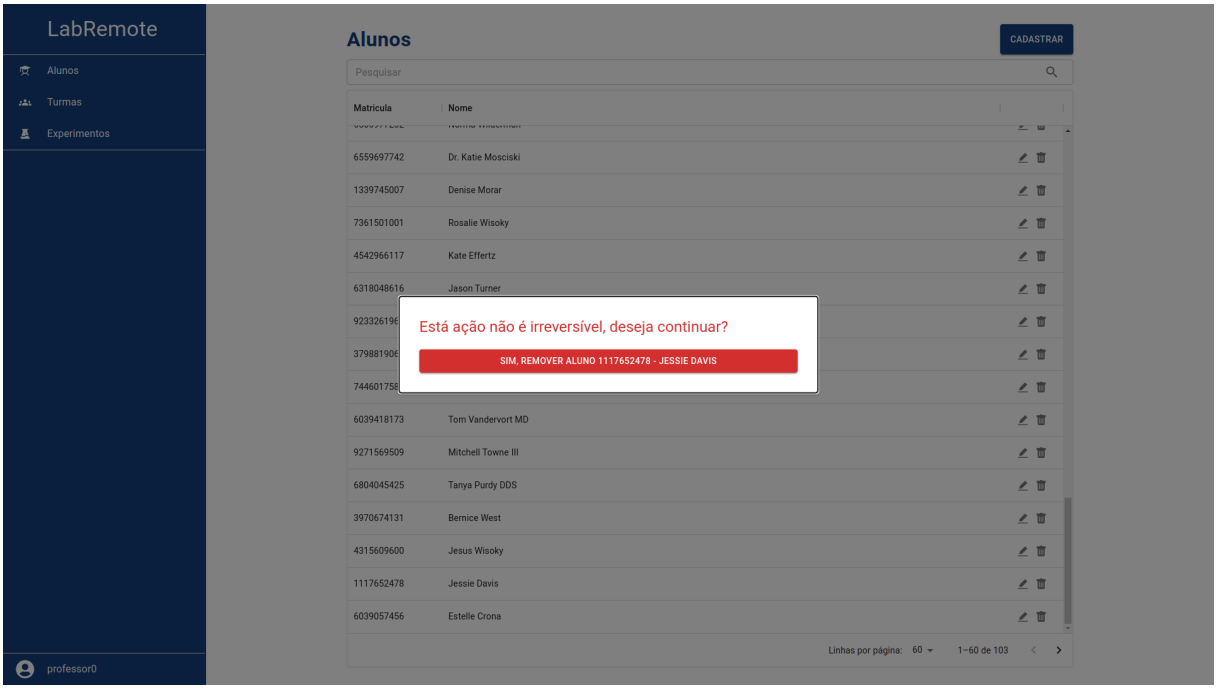
Fonte: autor

Figura 14 – Tela de edição de aluno



Fonte: autor

Figura 15 – Tela de remoção de aluno

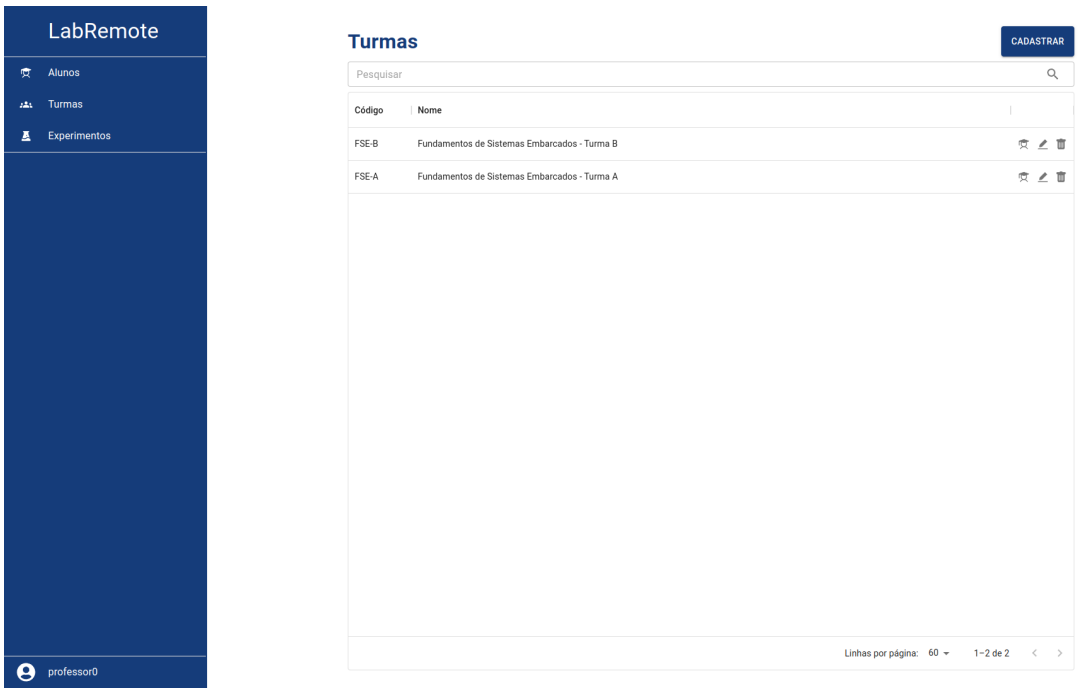


Fonte: autor

4.1.2.3 Tela de gerenciamento de turmas

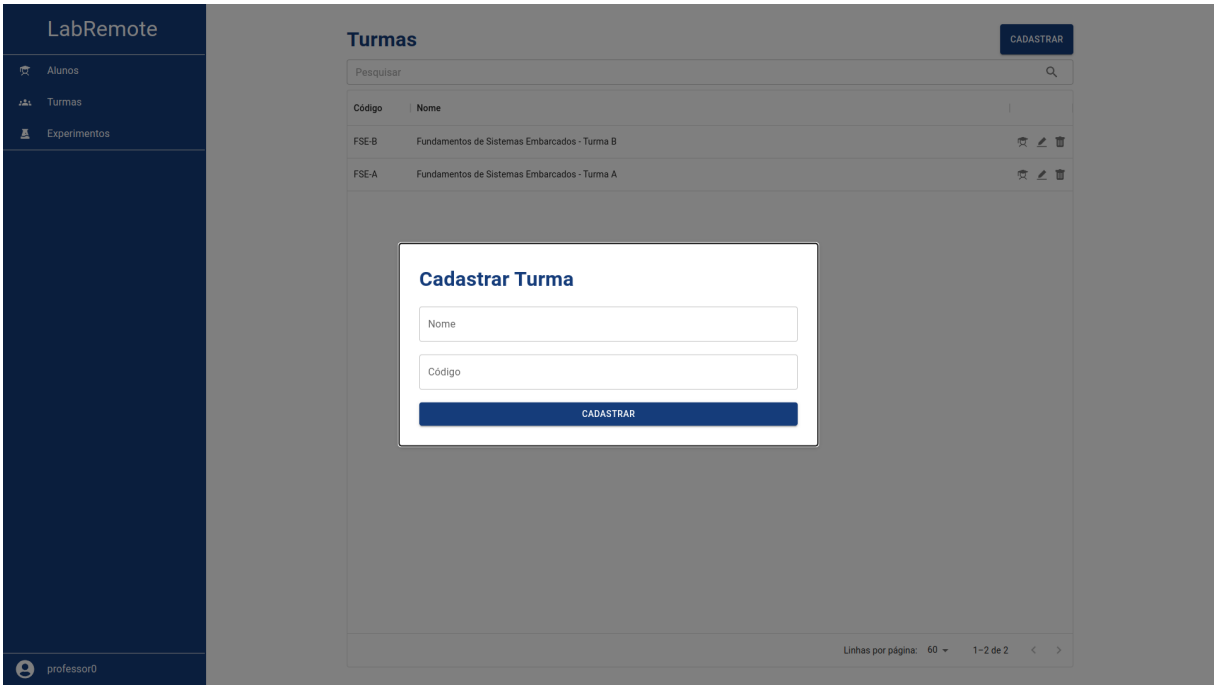
Professores cadastrados na aplicação têm a capacidade de gerenciar suas turmas, realizando operações como cadastro, edição e remoção. A Figura 17 apresenta a tela para o cadastro de novas turmas, onde o professor pode inserir informações detalhadas sobre cada turma. A Figura 18 ilustra a interface de edição, que permite modificar dados de turmas já registradas. A Figura 19 mostra a tela de confirmação para a remoção de uma turma. Finalmente, a Figura 20 exibe a tela de associação de alunos a uma turma, permitindo que o professor vincule alunos específicos às turmas existentes.

Figura 16 – Tela de gerenciamento de turmas



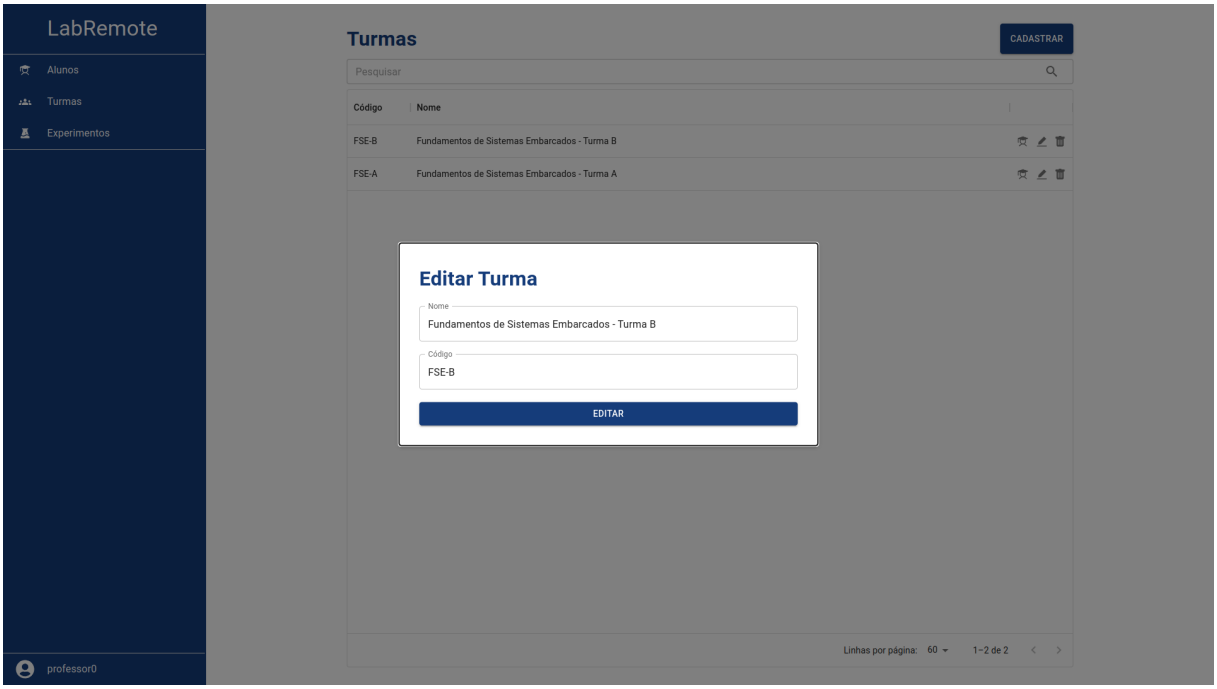
Fonte: autor

Figura 17 – Tela de cadastro de turma



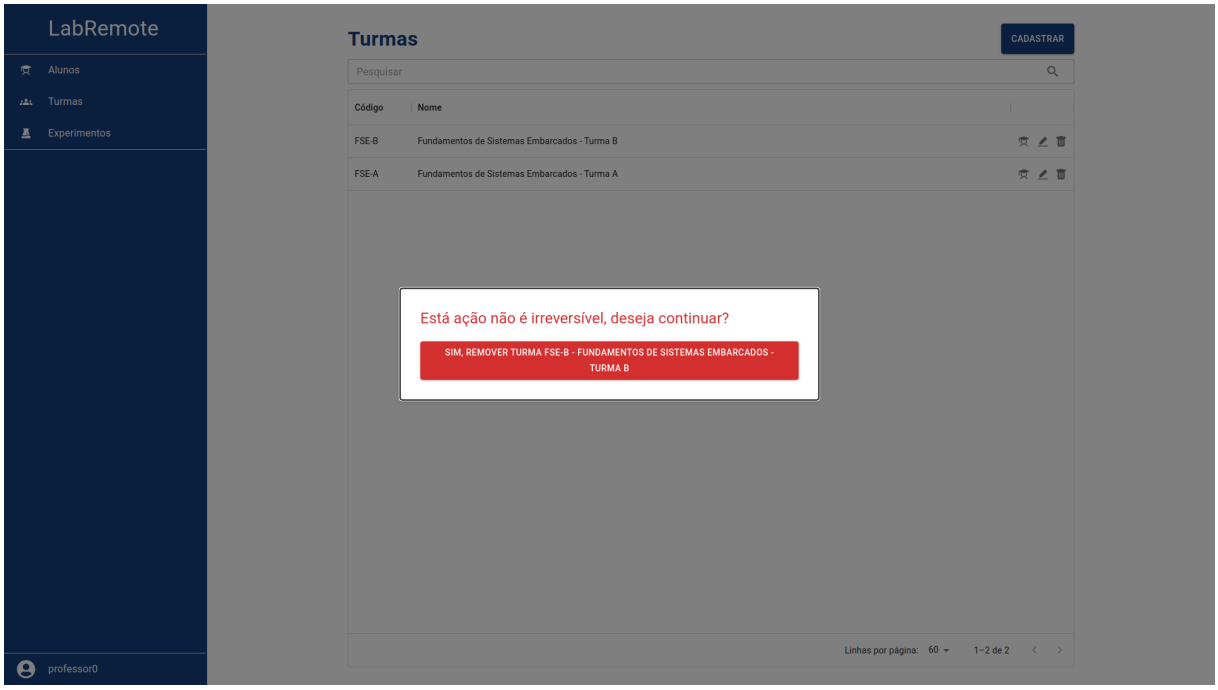
Fonte: autor

Figura 18 – Tela de edição de turma



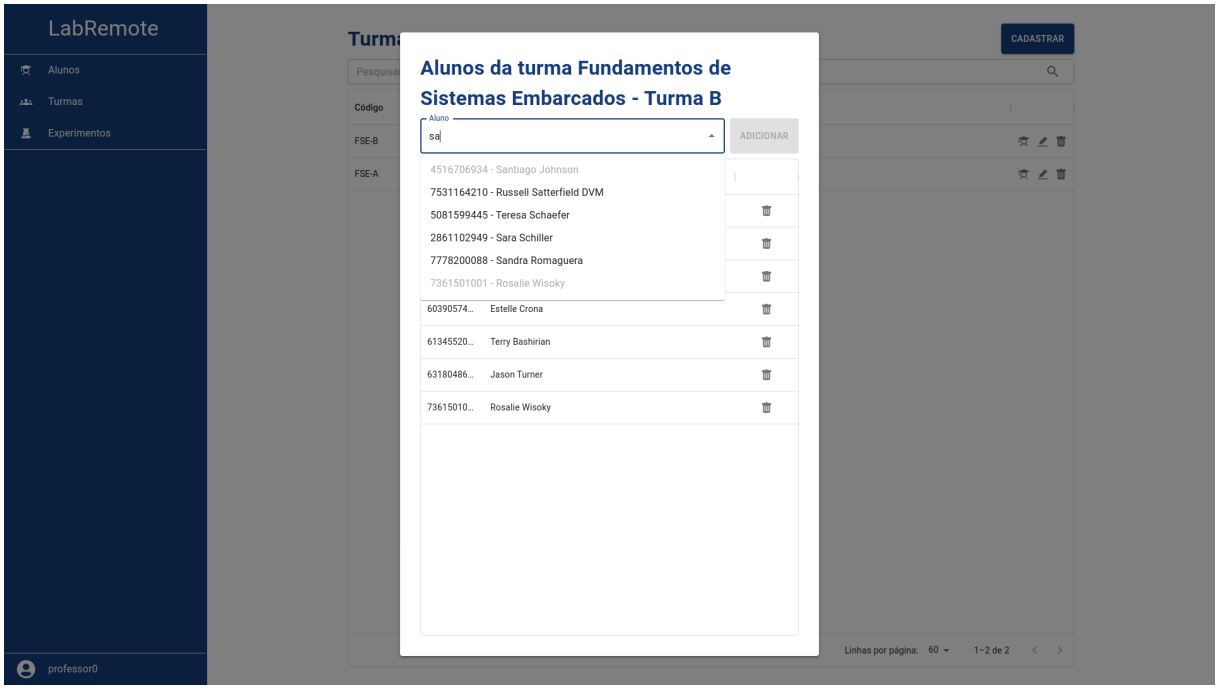
Fonte: autor

Figura 19 – Tela de remoção de turma



Fonte: autor

Figura 20 – Tela de associação de alunos a turma

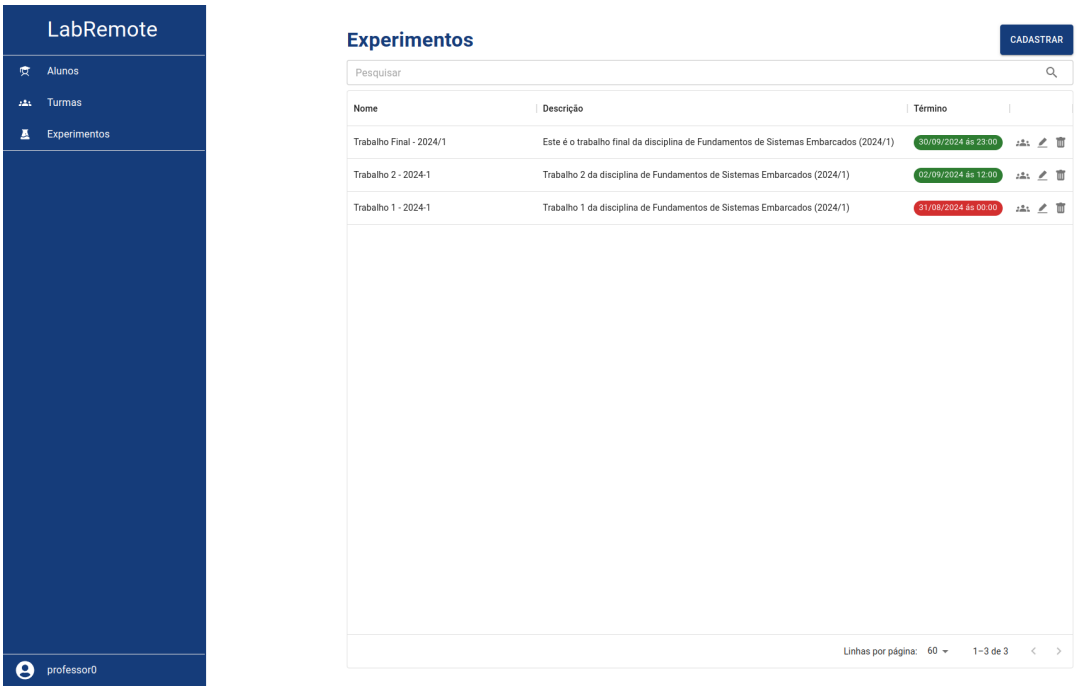


Fonte: autor

4.1.2.4 Tela de gerenciamento de experimentos

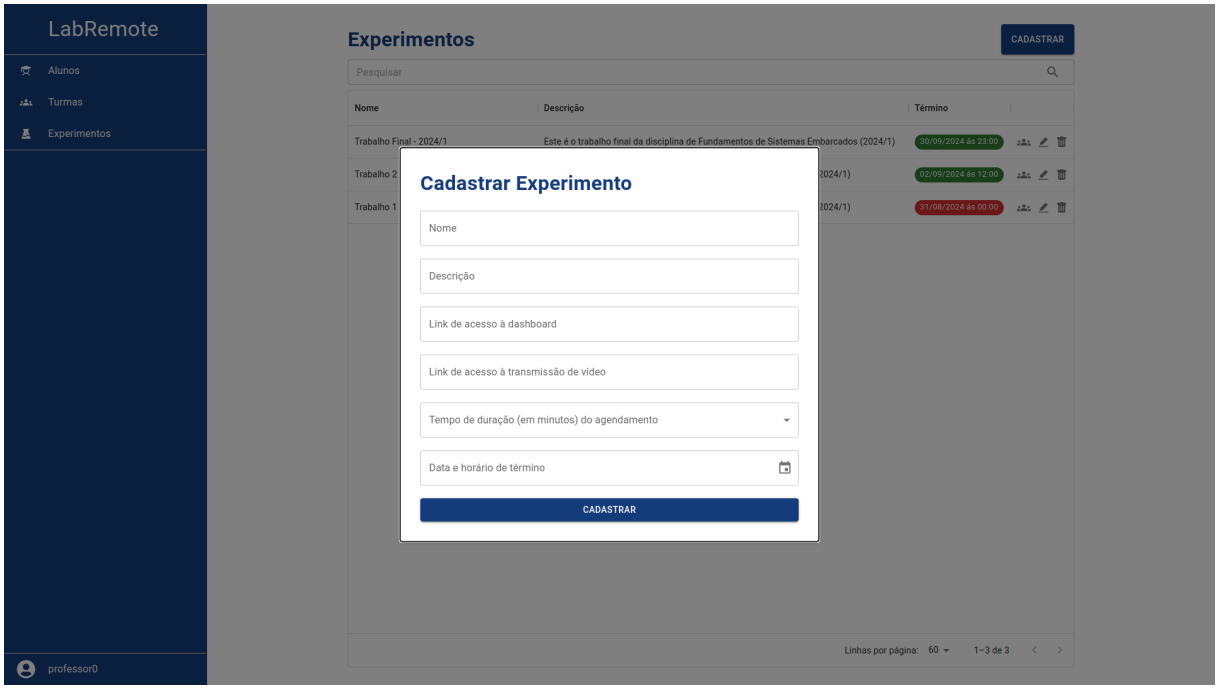
Professores cadastrados na aplicação têm a capacidade de gerenciar seus experimentos. A Figura 21 mostra a tela de gerenciamento de experimentos, onde o professor pode visualizar todos os seus experimentos cadastrados na plataforma. A Figura 22 apresenta a tela de cadastro de novos experimentos, permitindo ao usuário inserir detalhes e configurações específicas para cada experimento. A Figura 23 exibe a tela de edição de experimentos, que permite a atualização das informações e ajustes necessários nos experimentos existentes. A Figura 24 ilustra a tela de confirmação remoção de experimento. Por fim, a Figura 25 apresenta a tela de associação de turmas a um experimento, possibilitando o vínculo entre os experimentos cadastrados e as turmas específicas.

Figura 21 – Tela de gerenciamento de experimentos



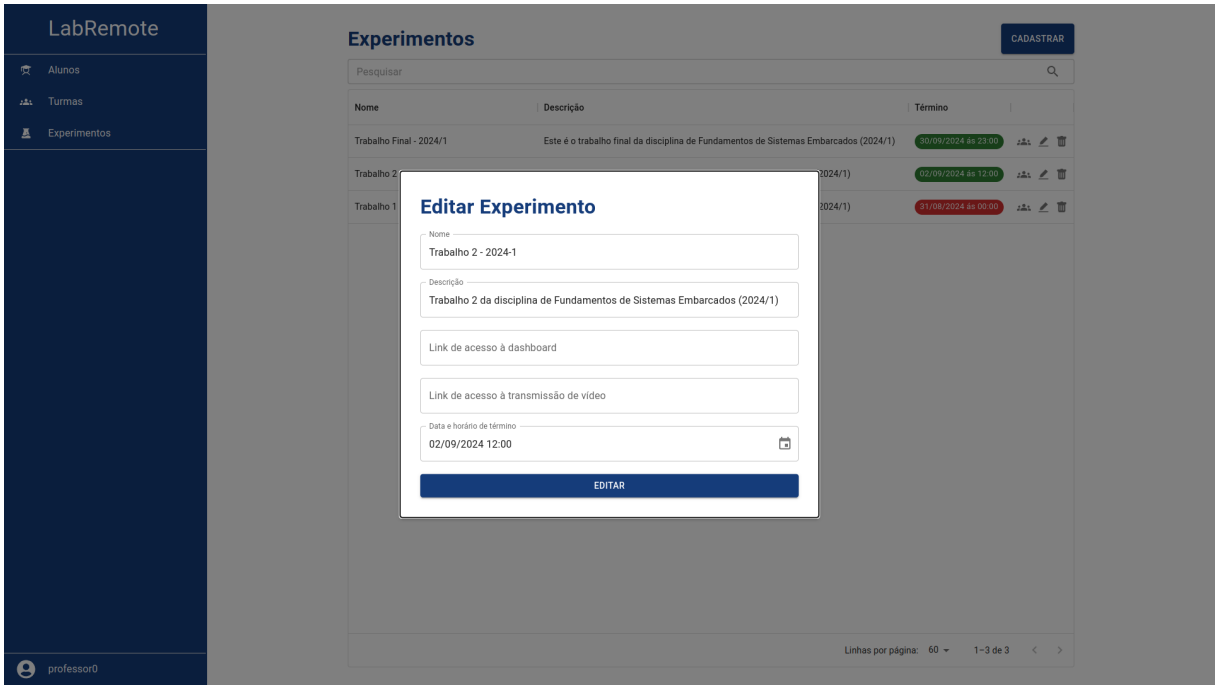
Fonte: autor

Figura 22 – Tela de cadastro de experimento



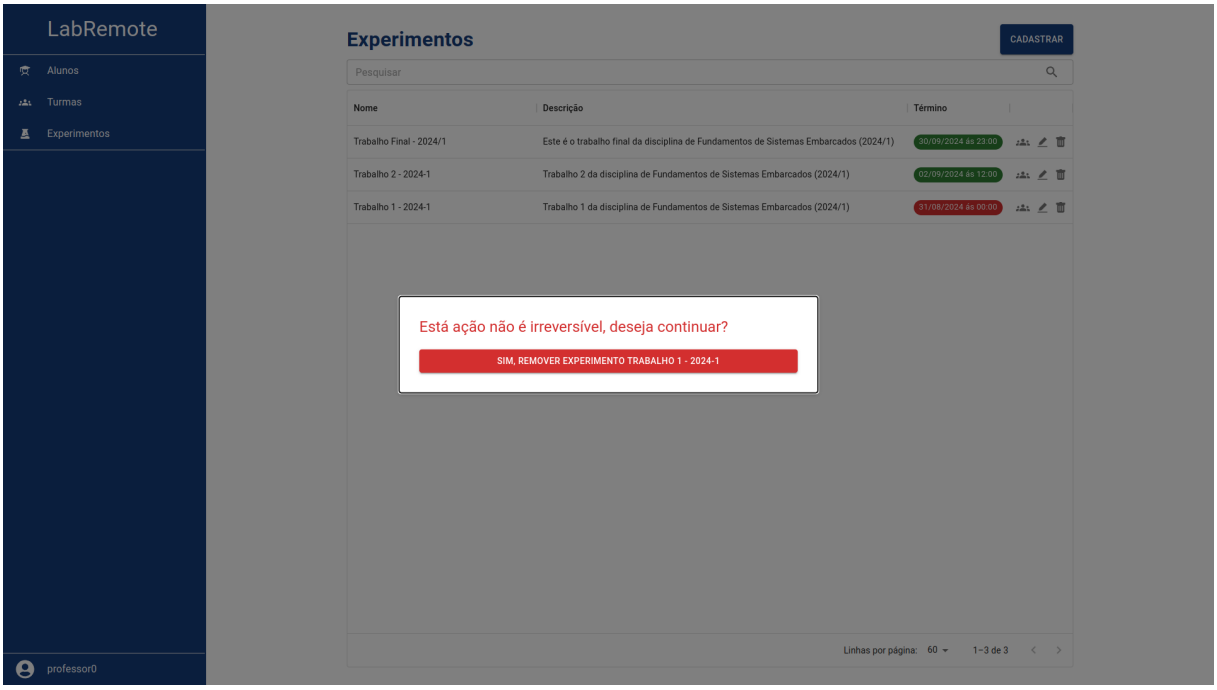
Fonte: autor

Figura 23 – Tela de edição de experimento



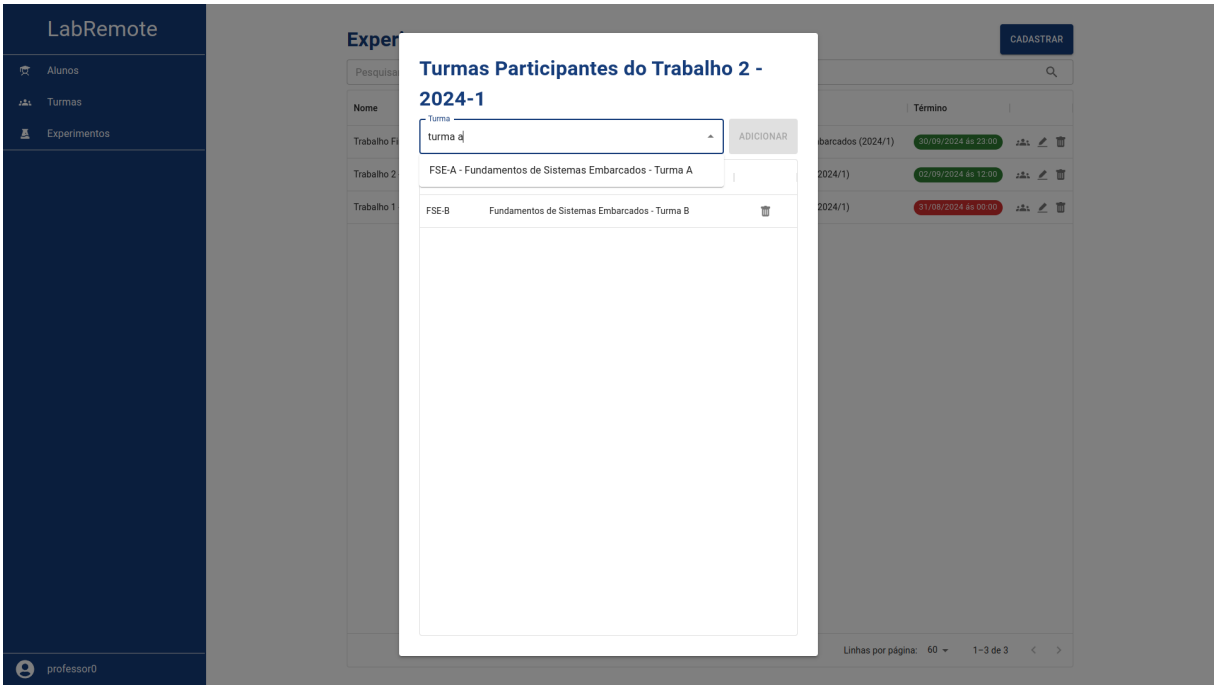
Fonte: autor

Figura 24 – Tela de remoção de experimento



Fonte: autor

Figura 25 – Tela de associação de turmas a experimento

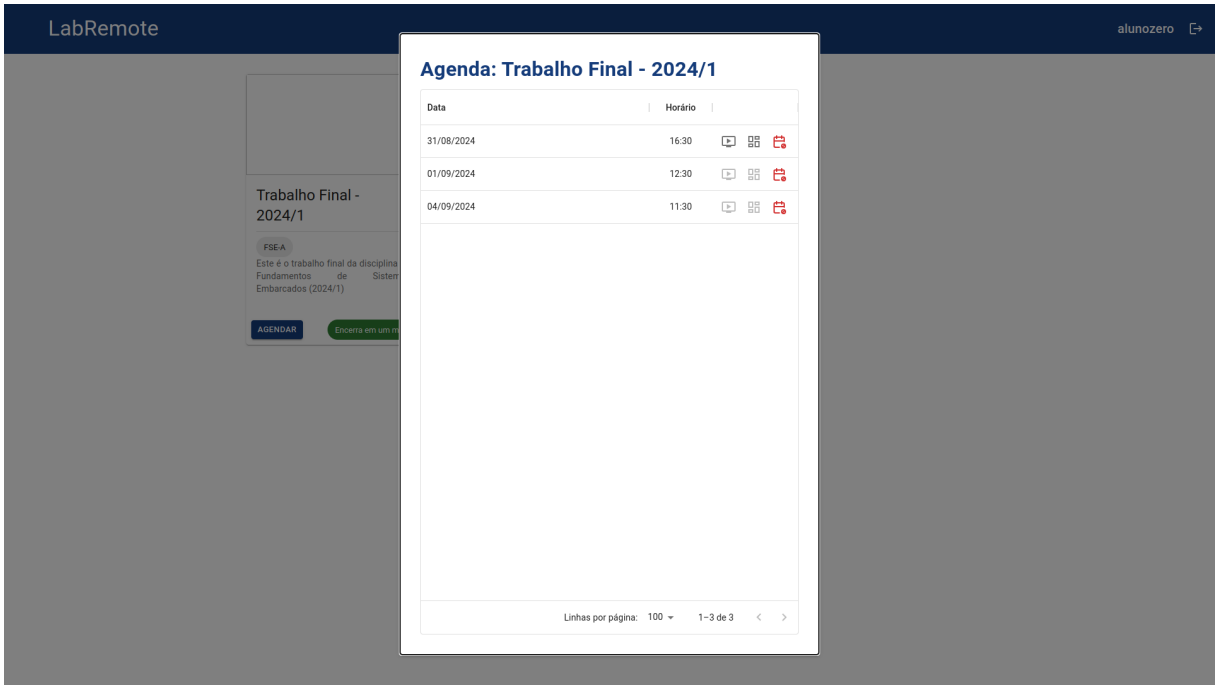


Fonte: autor

4.1.2.5 Tela de agendamento e acesso a *dashboard* do ThingsBoard

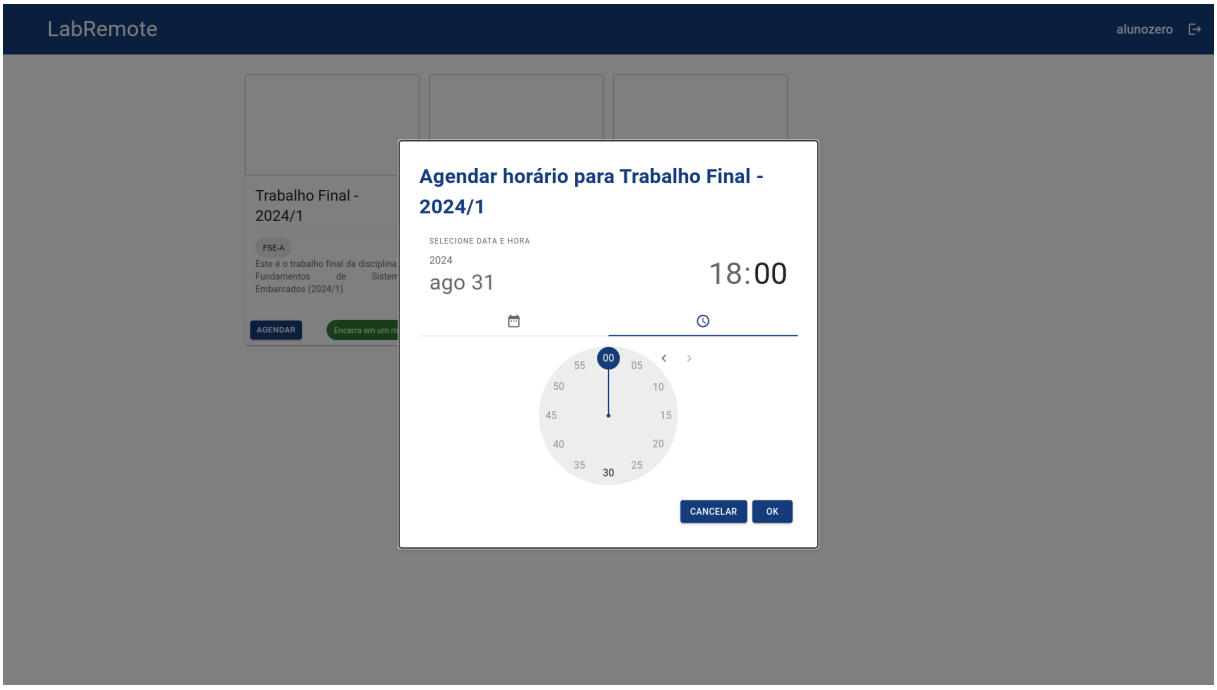
Os alunos podem visualizar os experimentos das turmas com as quais estão associados e gerenciar o agendamento para acessar o painel do ThingsBoard. A Figura 26 ilustra a tela de exibição dos experimentos disponíveis para agendamento, permitindo aos alunos visualizar todos os experimentos abertos para reserva que correspondem às turmas às quais estão associados. A Figura 27 mostra a tela de agendamento para um experimento específico, onde o aluno pode selecionar a data e o horário desejados. É permitido apenas um agendamento pendente por dia. Caso o aluno cancele ou conclua o agendamento, ele poderá realizar um novo agendamento no mesmo dia. A Figura 28 exibe a tela de listagem dos agendamentos de um experimento, permitindo ao aluno visualizar todos os horários já reservados para o experimento selecionado. Por fim, a Figura 29 apresenta a tela de acesso ao *dashboard*, onde o aluno pode visualizar o tempo restante para o seu próximo agendamento.

Figura 26 – Tela de exibição de experimentos para agendamento



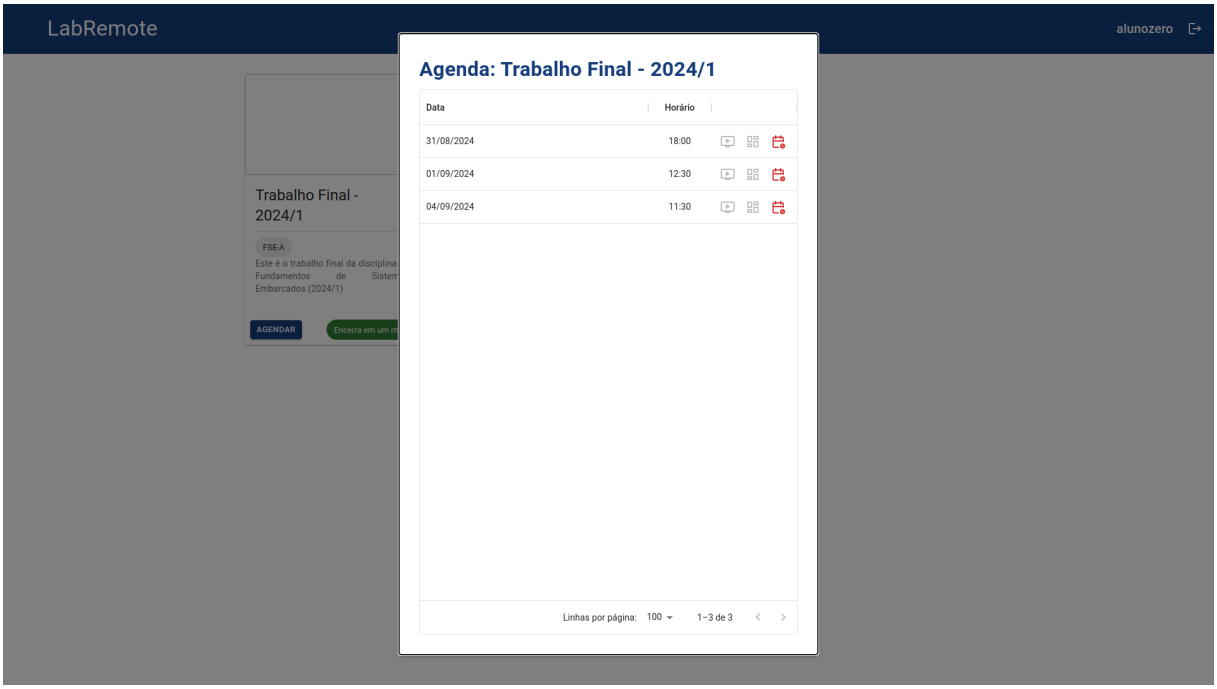
Fonte: autor

Figura 27 – Tela de agendamento para um experimento



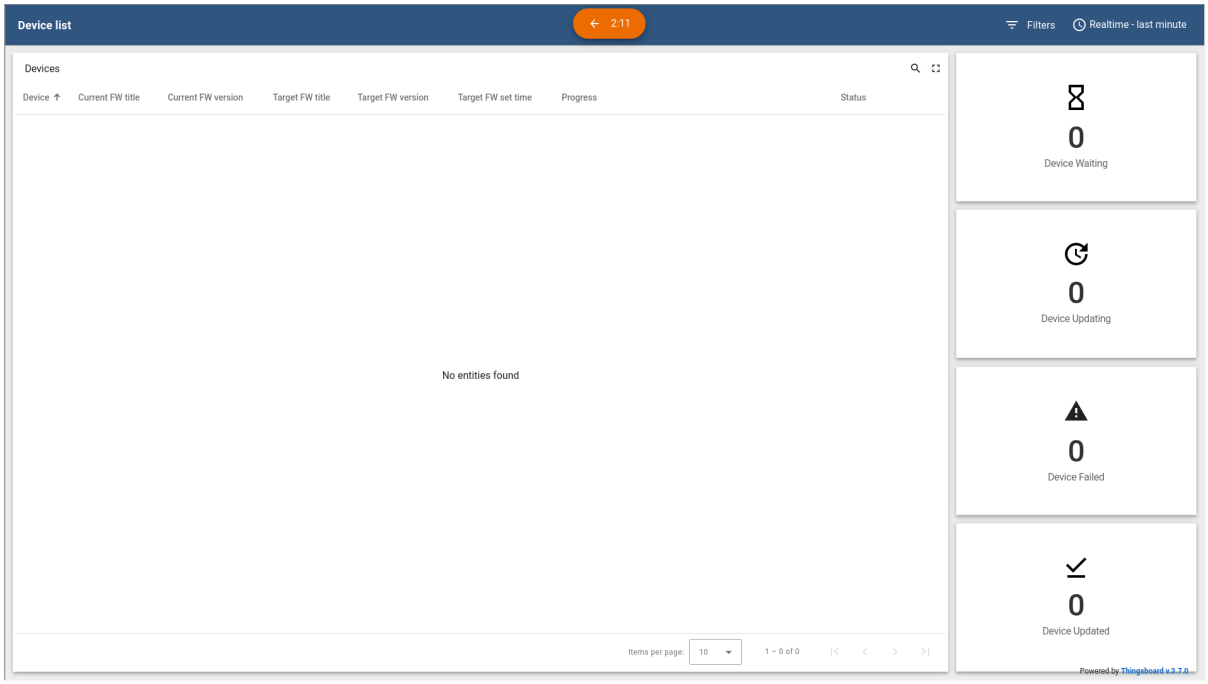
Fonte: autor

Figura 28 – Tela de listagem de agendamentos de um experimento



Fonte: autor

Figura 29 – Tela de acesso a *dashboard*



Fonte: autor

4.1.3 Sistema de streaming de vídeo ao vivo

Com a necessidade de streaming de vídeo ao vivo com baixa latência, foi analisado opções para realização do requisito **RF06**. Dentre eles Kurento, Janus WebRTC Server e MediaMTX que estão especificados nas Seções 4.1.3.1, 4.1.3.2 e 4.1.3.3 respectivamente.

4.1.3.1 Kurento

Kurento é uma plataforma de *media server* que oferece várias funcionalidades, incluindo transmissão em tempo real, processamento de mídia, e WebRTC. É frequentemente utilizado em aplicações que necessitam de manipulação avançada de mídia (KURENTO, 2023).

4.1.3.2 Janus WebRTC Server

Janus é um servidor WebRTC modular que facilita a criação de soluções de comunicação em tempo real. É leve e focado em ser uma ponte entre diferentes protocolos e o WebRTC (MEETECHO, 2014).

4.1.3.3 MediaMTX

MediaMTX (anteriormente rtsp-simple-server) é um servidor de mídia em tempo real e *proxy* de mídia pronto para uso e sem dependências, é um servidor projetado para

ser simples e eficiente, com foco específico em integração com dispositivos de câmera IP e outras fontes de mídia, incluindo WebRTC (BLUENVIRON, 2024).

4.1.4 Análise e escolha da solução para streaming de vídeo ao vivo

Após uma análise criteriosa dos pontos fortes e fracos de cada solução, o MediaMTX se destacou como a escolha mais adequada para o projeto. A simplicidade de configuração e a facilidade de integração do MediaMTX foram fatores decisivos, especialmente considerando o escopo do projeto e a necessidade de uma solução que pudesse ser rapidamente implementada e mantida com o mínimo de complexidade. Embora o Kurento ofereça funcionalidades avançadas e uma ampla gama de possibilidades para processamento de mídia, sua complexidade não se alinhava com os recursos e o cronograma disponíveis. Similarmente, o Janus, apesar de ser uma solução leve e modular. Portanto, o MediaMTX foi escolhido como a solução preferencial para atender ao requisito **RF06**.

4.1.5 Desenvolvimento da solução de streaming de vídeo ao vivo

De acordo com a página do repositório do MediaMTX, é possível utilizar o software de linha de comando FFmpeg ou GStreamer, bem como softwares com interface gráfica, como o OBS Studio, para a publicação de streaming de vídeo. Para o desenvolvimento de uma solução de streaming de vídeo de baixa latência, as alternativas de linha de comando, como FFmpeg e GStreamer, se mostraram mais versáteis.

O FFmpeg é uma solução poderosa que decodifica, codifica, transcodifica, multiplexa, demultiplexa, transmite, filtra e reproduz praticamente qualquer formato já criado, dos mais antigos aos mais avançados (FFMPEG, 2024). Já o GStreamer permite criar qualquer tipo de aplicação de streaming multimídia. Projetado para lidar com áudio, vídeo ou ambos, ele também pode processar outros tipos de fluxos de dados. Seu design de pipeline reduz a sobrecarga, tornando-o ideal para aplicações de áudio de alto desempenho que exigem baixa latência (GSTREAMER, 2024).

Após a análise das duas opções, tanto o FFmpeg quanto o GStreamer apresentam ótimos resultados, alcançando latências inferiores a 200 milissegundos em rede local com transmissão via WebRTC. Para efeito de comparação, também foi realizado a transmissão utilizando o protocolo HLS[2], disponível no MediaMTX, cuja latência ficou em torno de 4 segundos. O GStreamer demonstrou melhor desempenho em termos de consumo de CPU e memória em comparação com o FFmpeg. No entanto, o FFmpeg se destacou pela facilidade de uso, já que não requer a instalação de *plugins* adicionais, ao contrário do GStreamer. Os detalhes do teste podem se verificados no Apêndice A.

[2] O HTTP Live Streaming (HLS) é um dos protocolos de transmissão de vídeo mais utilizados, permitindo que dispositivos se ajustem dinamicamente a variações nas condições da rede, aumentando ou reduzindo a qualidade da transmissão conforme necessário (CLOUDFLARE, 2024).

Com a escolha do FFmpeg, foi configurado o MediaMTX para que uma câmera USB conectada ao computador que hospeda o MediaMTX seja acionada apenas quando há demanda pelo streaming. Com essa configuração, o streaming só é ativado ao acessar "<ip do servidor MediaMTX>:8889/mystream", o que ajuda a poupar o uso do *hardware* do servidor e da câmera. O arquivo de configuração, com todos os outros arquivos necessários para ser utilizado, está disponível em: <https://gitlab.com/igorq937-tcc/laboratorio-remoto-mediamtx-config>.

4.2 Desenvolvimento do código base para os microcontroladores ESP32

O desenvolvimento do código base para a ESP32, começou pela implementação da conexão com a internet. Em seguida, foi configurada a comunicação com a plataforma ThingsBoard e, finalmente, a funcionalidade de atualização Over-The-Air (OTA), que permite a atualização remota do firmware. Isso facilita a manutenção e evolução contínua dos dispositivos, eliminando a necessidade de intervenção física.

Como indicado na documentação do framework ESP-IDF, ao utilizar o OTA (Over-The-Air), é necessário configurar pelo menos duas partições para atualização, nomeadas como ota_0, ota_1, ota_2, e assim por diante, até ota_15. Essas partições são alternadas durante a atualização do *firmware*, onde todas, exceto uma, são usadas para a inserção de novos *firmwares*, enquanto apenas uma é utilizada pelo ESP32 para a inicialização. Além dessas partições, é necessário incluir uma partição adicional chamada otadata, que armazena o contador ota_seq, funcionando como um ponteiro para a partição que será selecionada para a inicialização (ESPRESSIF. . . , 2023).

Com o uso da biblioteca ThingsBoard Client SDK, o tamanho do *firmware* aumentou significativamente, excedendo a capacidade de armazenamento do ESP32. Para acomodar os requisitos do OTA e o aumento no tamanho do *firmware*, foi necessário remover a partição de fábrica (*factory*), destinada à restauração de fábrica, permitindo a criação de partições OTA maiores. Na ausência da partição de fábrica, o *bootloader* do ESP-IDF utilizará qualquer partição OTA disponível (geralmente a ota_0) para inicialização, normalmente o *factory* seria utilizado até que uma partição OTA fosse acionada. Como tanto atualizações (upgrades) quanto retrocessos (*downgrades*) do *firmware* podem ser realizados via OTA, a perda da funcionalidade de restauração de fábrica pode ser facilmente contornada. A configuração final das partições pode ser visualizada na Tabela 1.

Tabela 1 – Tabela de partições da ESP32

Nome	**Tamanho (*Bytes*)**	**Descrição**
nvs*	0x6000	Armazenamento de variáveis não voláteis
otadata*	0x2000	Dados de configuração OTA
phy_init*	0x1000	Inicialização dos parâmetros de *hardware* de rádio
coredump	64K	Armazenamento de informações de despejo de memória em caso de falha
ota_0*	1920k	Primeiro slot de aplicativo OTA
ota_1*	1920k	Segundo slot de aplicativo OTA
(*) Partições obrigatórias para o funcionamento adequado do sistema.		

4.3 Incrementos do desenvolvimento

O desenvolvimento seguiu o modelo de processo de software incremental, como mencionado na Seção 3.4. Cada incremento foi projetado para adicionar um conjunto específico de funcionalidades ou melhorar características existentes. Os incrementos seguiram uma ordem lógica de implementação, começando pelas funcionalidades básicas de autenticação e gerenciamento de usuários e evoluindo para aspectos mais complexos, como o desenvolvimento de uma solução de streaming de vídeo de baixa latência e a otimização da experiência do usuário nas interfaces web. E por fim o desenvolvimento do código base para os microcontroladores ESP32.

4.4 Verificação dos requisitos

Para garantir que os requisitos do software foram devidamente atendidos, foi elaborado uma lista de verificação, uma lista de verificação é composta por um conjunto de perguntas e/ou afirmações sobre uma determinada circunstância. As listas de verificação podem ser aplicadas sempre que muitos aspectos precisam ser considerados em um ambiente complexo e nenhum aspecto pode ser omitido (POHL; RUPP, 2015).

4.4.1 Verificação dos requisitos da plataforma web

- **RF01** O sistema permite o cadastro de professores com acesso às funcionalidades relacionadas a turmas, experimentos e alunos, incluindo a criação de contas e a atribuição de permissões adequadas? Sim, a plataforma possui um sistema de autenticação que regula o acesso a todas as funcionalidades.

- **RF02** Os professores cadastrados podem gerenciar turmas, informando detalhes como nome, horário e localização? Além disso, conseguem associar turmas a alunos

e experimentos correspondentes? Sim, os professores podem gerenciar turmas com todas as informações relevantes, além de associá-las a alunos e experimentos.

- **RF03** Os professores cadastrados podem gerenciar experimentos e vinculá-los às turmas correspondentes? Sim, os professores podem gerenciar experimentos, incluindo a vinculação às turmas adequadas.

- **RF04** Os professores cadastrados podem gerenciar alunos, fornecendo informações como nome, matrícula e turma a que pertencem, permitindo a associação desses alunos às turmas adequadas? Sim, os professores podem gerenciar alunos na plataforma, incluindo a associação às turmas desejadas.

- **RF05** Os alunos cadastrados podem agendar experimentos relacionados às suas turmas, verificando a disponibilidade dos experimentos e escolhendo horários disponíveis para agendamento? Sim, os alunos cadastrados na plataforma podem visualizar os experimentos associados às suas turmas e agendar horários disponíveis para acesso ao dashboard.

- **RF06** O sistema oferece um recurso de streaming de vídeo ao vivo, permitindo que os usuários visualizem em tempo real os resultados dos experimentos que estão sendo realizados? Sim, a plataforma permite o compartilhamento de streams ao vivo. Detalhes sobre a solução adotada para streaming de vídeo são discutidos na Seção 4.1.3.

- **RF07** O sistema fornece um painel do ThingsBoard embutido na plataforma, permitindo que os usuários interajam e acompanhem os experimentos em andamento? Sim, o dashboard do ThingsBoard está totalmente integrado à plataforma, oferecendo acesso direto e facilitando a interação e o monitoramento dos experimentos.

- **RNF01** O sistema informa erros nos campos dos formulários? Sim, todos os formulários da plataforma possuem validação e apresentam mensagens de erro claras para orientar o usuário.

- **RNF02** O sistema inclui uma função de busca eficiente que permite aos usuários localizar rapidamente informações? Sim, todos os dados listados na plataforma podem ser pesquisados de forma rápida e eficaz.

- **RNF03** O sistema fornece feedback claro e imediato para todas as ações realizadas pelos usuários, incluindo mensagens de sucesso, erro e progresso? Sim, a plataforma oferece feedback imediato com mensagens claras para todas as ações, garantindo uma melhor experiência do usuário.

4.4.2 Verificação dos requisitos do código base para ESP32

- **R01** O código permite a conexão com a internet? Sim, o código está configurado para estabelecer uma conexão com a internet.
- **R02** O código permite a comunicação com a plataforma ThingsBoard? Sim, o código garante comunicação contínua e eficaz com a plataforma ThingsBoard.
- **R03** O código permite a atualização por OTA da ESP32? Sim, o código suporta a atualização *Over-The-Air* (OTA) da ESP32, facilitando a manutenção e atualização remota do dispositivo.

5 Considerações Finais

Este capítulo tem como finalidade sintetizar os principais resultados alcançados ao longo do desenvolvimento deste trabalho, avaliando o cumprimento dos objetivos propostos. Por fim, serão apresentadas sugestões para trabalhos futuros, que possam expandir ou aprimorar as contribuições desta pesquisa.

5.1 Conclusão

Em suma, este trabalho propôs a criação de um Ambiente Educacional de Acesso Remoto para Bancadas de Sistemas Embarcados, voltado para a disciplina de Fundamentos de Sistemas Embarcados do Campus Faculdade do Gama da Universidade de Brasília. A proposta surgiu após uma avaliação detalhada do contexto da disciplina e a análise de soluções tecnológicas consolidadas, já amplamente exploradas em diferentes instituições ao redor do mundo.

O principal objetivo foi oferecer aos alunos um acesso remoto controlado aos *hardwares* utilizados nos experimentos da disciplina, criando um ambiente que permita a realização de atividades práticas de forma flexível e eficaz. Além disso, buscou-se simplificar a configuração e a criação de novos experimentos.

O desenvolvimento foi concluído, atendendo a todos os requisitos descritos na proposta de trabalho, que incluíam a conclusão de uma plataforma de agendamento para experimentos e sua integração com a *dashboard* de acesso aos experimentos já utilizada pela disciplina, além de um streaming de vídeo de baixa latência. Também foi elaborado um código base para os microcontroladores ESP32 utilizados na criação dos experimentos, visando facilitar a geração de novos experimentos. O resultado é um ambiente com potencial para superar os desafios associados à realização de atividades práticas.

5.2 Trabalhos Futuros

Embora o desenvolvimento do projeto tenha sido concluído, as etapas de validação especificadas nas Seções 3.2.2 e 3.3.3 não foram realizadas. Para dar continuidade a este trabalho, é essencial que, em estudos futuros, sejam conduzidos testes com usuários reais do sistema, a fim de validar a interface e os requisitos implementados.

Referências

AMAZON. *Video Latency in Live Streaming*. 2024. Disponível em: <https://aws.amazon.com/media/tech/video-latency-in-live-streaming/>. Citado 2 vezes nas páginas 34 e 69.

ARDUINO - Arduino Hardware. 2022. Disponível em: <https://www.arduino.cc/en/hardware>. Acesso em: 29 jun. 2023. Citado na página 29.

ARIZA, J. ; BAEZ, H. Understanding the role of single-board computers in engineering and computer science education: A systematic literature review. *Computer Applications in Engineering Education*, v. 30, n. 1, p. 304–329, 2022. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.22439>. Citado 2 vezes nas páginas 27 e 28.

AWS. *Qual é a diferença entre front-end e back-end no desenvolvimento de aplicações?* 2020. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-frontend-and-backend/>. Acesso em: 31 jul. 2023. Citado na página 33.

BALAMURALITHARA, B.; WOODS, P. Virtual laboratories in engineering education: The simulation lab and remote lab. v. 17, n. 1, p. 108–118, Mar 2009. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1002/cae.20186>. Citado 2 vezes nas páginas 19 e 23.

BANZI, M. *How Arduino is open-sourcing imagination*. TED Talks, 2012. Disponível em: <https://www.ted.com/talks/massimo_banzi_how_arduino_is_open_sourcing_imagination>. Acesso em: 03 jul. 2023. Citado na página 28.

BARBOSA, S. D. J. et al. *Interação Humano-Computador e Experiência do Usuário*. [S.l.]: Autopublicação, 2021. ISBN 978-65-00-19677-1. Citado na página 42.

BLUENVIRON. *GitHub - bluenviron/mediamtx: Ready-to-use SRT / WebRTC / RTSP / RTMP / LL-HLS media server and media proxy that allows to read, publish, proxy, record and playback video and audio streams.* 2024. Disponível em: <https://github.com/bluenviron/mediamtx>. Acesso em: 15 ago. 2024. Citado na página 56.

BOCHICCHIO, M. A.; LONGO, A. Hands-on remote labs: Collaborative web laboratories as a case study for it engineering classes. v. 2, n. 4, p. 320–330, Oct 2009. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5204073>. Citado na página 20.

BONIFáCIO, R. Laboratório remoto baseado em fpga aplicado nas disciplinas de prática de eletrônica digital 1 e 2 da faculdade unb gama. *Bdm.unb.br*, 2022. Disponível em: <https://bdm.unb.br/handle/10483/23653>. Citado na página 20.

CARDOSO, O. E. et al. Desenvolvimento de uma plataforma de aprendizagem de sistemas embarcados baseada em esp32. 2020. Citado na página 29.

CLOUDFLARE. *O que é HTTP live streaming (HLS)?* 2024. Disponível em: <https://www.cloudflare.com/pt-br/learning/video/what-is-http-live-streaming/>. Acesso em: 15 ago. 2024. Citado na página 56.

DICKMANN, K. Remote access laboratory design and installation. *Unisq.edu.au*, Oct 2013. Disponível em: <https://sear.unisq.edu.au/24655/>. Citado na página 24.

ESPRESSIF - ESP-IDF Programming Guide. 2023. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html>. Acesso em: 28 jun. 2023. Citado 5 vezes nas páginas 29, 30, 31, 39 e 57.

FFMPEG. *About FFmpeg*. 2024. Disponível em: <https://ffmpeg.org/about.html>. Acesso em: 15 ago. 2024. Citado na página 56.

GSTREAMER. *What is GStreamer?* 2024. Disponível em: <https://gstreamer.freedesktop.org/documentation/application-development/introduction/gstreamer.html?gi-language=c>. Acesso em: 15 ago. 2024. Citado na página 56.

HIVEMQ. *Introducing the MQTT Protocol - MQTT Essentials: Part 1*. 2015. Disponível em: <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>. Acesso em: 06 jul. 2023. Citado na página 32.

JONG, T. de; LINN, M. C.; ZACHARIA, Z. C. Physical and virtual laboratories in science and engineering education. v. 340, n. 6130, p. 305–308, Apr 2013. Disponível em: <https://www.science.org/doi/epdf/10.1126/science.1230579>. Citado na página 23.

KURENTO. *GitHub - Kurento/kurento: Kurento WebRTC Media Server*. 2023. Disponível em: <https://github.com/Kurento/kurento>. Citado na página 55.

MEETECHO. *GitHub - meetecho/janus-gateway: Janus WebRTC Server*. 2014. Disponível em: <https://github.com/meetecho/janus-gateway>. Acesso em: 15 ago. 2024. Citado na página 55.

MICHON, R. et al. A faust architecture for the esp32 microcontroller. In: *Sound and Music Computing Conference (SMC-20)*. [S.l.: s.n.], 2020. Citado na página 31.

MQTT. *MQTT -FAQ*. 2022. Disponível em: <https://mqtt.org/faq/>. Acesso em: 06 jul. 2023. Citado na página 31.

MUI. *Getting started/*. 2024. Disponível em: <https://mui.com/material-ui/getting-started/>. Acesso em: 15 ago. 2024. Citado na página 34.

NESTJS. *Documentation | NestJS - A progressive Node.js framework*. 2023. Disponível em: <https://docs.nestjs.com/>. Acesso em: 30 jul. 2023. Citado 2 vezes nas páginas 34 e 39.

NIELSEN, J. Enhancing the explanatory power of usability heuristics. In: *Conference Companion on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 1994. (CHI '94), p. 210. ISBN 0897916514. Disponível em: <https://doi.org/10.1145/259963.260333>. Citado na página 42.

POHL, K.; RUPP, C. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant*. 2nd. ed. [S.l.]: Rocky Nook, 2015. ISBN 978-1937538774. Citado 2 vezes nas páginas 38 e 58.

PONTES, E. R. Desenvolvimento de experimentos para o ensino de sistemas embarcados. *Ufcg.edu.br*, Universidade Federal de Campina Grande, 2017. Disponível em: <http://dspace.sti.ufcg.edu.br:8080/xmlui/handle/riufcg/18745>. Citado na página 25.

RASPBERRY Pi Documentation. 2023. Disponível em: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>. Acesso em: 24 jun. 2023. Citado na página 28.

REACT. *React*. 2023. Disponível em: <https://react.dev/>. Acesso em: 30 jul. 2023. Citado 2 vezes nas páginas 33 e 34.

RELLE - Sobre. 2023. Disponível em: <http://relle.ufsc.br/about>. Acesso em: 24 jun. 2023. Citado na página 24.

REXLAB - Sobre. 2023. Disponível em: <https://rexlab.ufsc.br/about/>. Acesso em: 24 jun. 2023. Citado na página 19.

SILVA, J. B. d. A utilização da experimentação remota como suporte para ambientes colaborativos de aprendizagem. *Ufsc.br*, Florianópolis, SC, 2023. Disponível em: <https://repositorio.ufsc.br/xmlui/handle/123456789/88357>. Citado na página 19.

SOMMERVILLE, I. *Engenharia de Sotware*. 9ª edição. ed. [S.l.]: Pearson Universidades, 2011. ISBN 978-85-7936-108-1. Citado 2 vezes nas páginas 37 e 41.

SOUZA, A. L. F. L. d. Plataforma educacional de acesso remoto para bancadas de controle de sistemas embarcados. *Bdm.unb.br*, Feb 2023. Disponível em: <https://bdm.unb.br/handle/10483/39138>. Citado 4 vezes nas páginas 20, 21, 33 e 39.

TANSKANEN, S. Latency contributors in webrtc-based remote control system. *Aaltodoc*, Mar 2021. Disponível em: <https://urn.fi/URN:NBN:fi:aalto-202103212337>. Citado na página 34.

THINGSBOARD - Open-source IoT Platform. 2023. Disponível em: <https://thingsboard.io/>. Acesso em: 24 jun. 2023. Citado na página 33.

TIDESTRÖM, J. *Investigation into low latency live video streaming performance of WebRTC*. 2019. Disponível em: <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1304486&dswid=-8987>. Citado na página 34.

TULHA, C. N.; ANTONIO, M.; COLUCI, V. R. Uso de laboratórios remotos no brasil: uma revisão sistemática. v. 22, n. 2, Oct 2019. Disponível em: <https://seer.ufrgs.br/InfEducTeoriaPratica/article/view/90543>. Citado 3 vezes nas páginas 19, 20 e 23.

WEBLAB-DEUSTO - About. 2023. Disponível em: <https://weblab.deusto.es/website/about.html>. Acesso em: 24 jun. 2023. Citado na página 24.

WERTH, A. et al. *Rapid Transition to Remote Instruction of Physics Labs During Spring 2020: Instructor Perspectives*. 2021. Disponível em: <https://www.proquest.com/working-papers/rapid-transition-remote-instruction-physics-labs/docview/2613417502/se-2>. Citado na página 20.

WIRING. 2023. Disponível em: <http://wiring.org.co/>. Acesso em: 29 jun. 2023. Citado na página 29.

WOLF, W. Hardware-software co-design of embedded systems. *Proceedings of the IEEE*, v. 82, n. 7, p. 967–989, 1994. Citado na página 25.

ZHAO, C. W.; JEGATHEESAN, J.; LOON, S. C. Exploring iot application using raspberry pi. In: . [S.l.: s.n.], 2015. Citado 2 vezes nas páginas 27 e 28.

Apêndices

APÊNDICE A – Apêndice

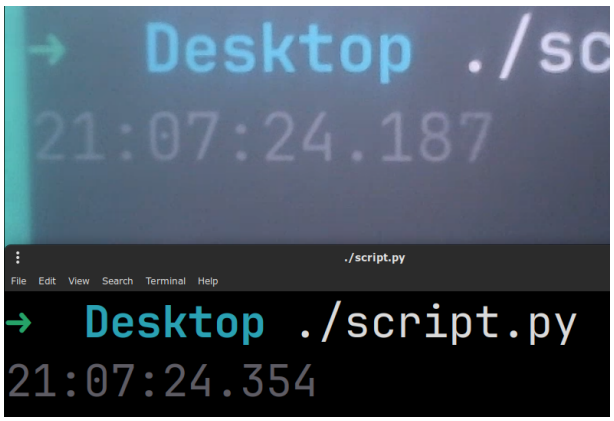
A.1 Teste do Streaming de Vídeo

Embora existam outros métodos, a forma mais simples de medir a latência de vídeo de ponta a ponta é a seguinte (AMAZON, 2024):

1. Utilize um tablet com um aplicativo de claquete.
2. Filme a tela com uma câmera conectada ao seu codificador de vídeo.
3. Publique o fluxo de vídeo em seu servidor de origem.
4. Transmita o vídeo ao seu reprodutor por meio de uma CDN (Content Delivery Network, ou Rede de Entrega de Conteúdo).
5. Coloque o reprodutor ao lado do tablet com a claquete.
6. Tire uma foto das duas telas.
7. Subtraia os códigos de tempo das duas imagens para determinar a latência.

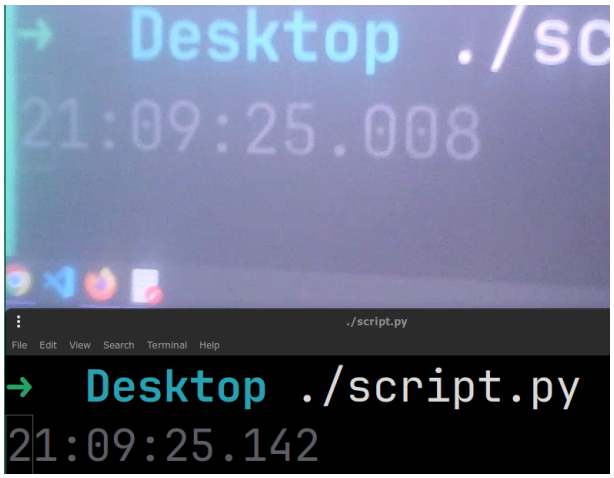
Para simplificar o processo, foi substituído o tablet por um monitor e optado por não utilizar uma CDN. A Figura 30 mostra que o FFmpeg alcançou uma latência de 167 milissegundos na transmissão WebRTC a partir do servidor MediaMTX. Por outro lado, a Figura 31 ilustra que o GStreamer obteve uma latência de 134 milissegundos na mesma configuração de transmissão WebRTC com o servidor MediaMTX. Em contraste na Figura 32, a transmissão com HLS codificada pelo GStreamer apresentou uma latência próxima a 4 segundos.

Figura 30 – Transmissão WebRTC com vídeo codificado por FFmpeg



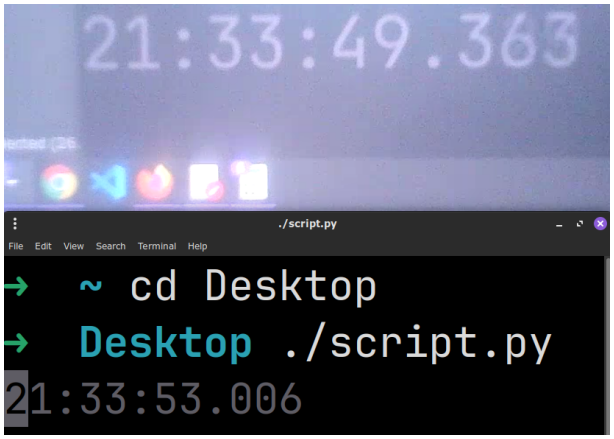
Fonte: autor

Figura 31 – Transmissão WebRTC com vídeo codificado por GStreamer



Fonte: autor

Figura 32 – Transmissão HSL com vídeo codificado por GStreamer



Fonte: autor