UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA

JOÃO MARCOS SOUZA FERREIRA

Aplicativo para Controle de Nível Baseado em Espaço de Estados utilizando MQTT como protocolo de comunicação

Uberlândia

2025

JOÃO MARCOS SOUZA FERREIRA

Aplicativo para Controle de Nível Baseado em Espaço de Estados utilizando MQTT como protocolo de comunicação

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Mecânica da Universidade Federal de Uberlândia como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica.

Área de concentração: Engenharia Mecatrônica

Orientador: Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares

Uberlândia

2025

JOÃO MARCOS SOUZA FERREIRA

Aplicativo para Controle de Nível Baseado em Espaço de Estados utilizando MQTT como protocolo de comunicação

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Mecânica da Universidade Federal de Uberlândia como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica.

Área de concentração: Engenharia Mecatrônica

Uberlândia, 26/09/2025

Banca Examinadora:

_____

Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares (UFU)

_____

Prof. Ms. Werley Rocherter Borges Ferreira (UFU)

_____

Prof. Ms. Nei Oliveira de Souza (IFTM)

Dedico este trabalho aos meus pais e meu irmão, pelo estímulo, carinho e apoio.

AGRADECIMENTOS

Agradeço ao professor e amigo Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares por me confiar o desenvolvimento deste projeto, me ajudando e orientando sempre que possível visando uma versão robusta e completa da solução proposta.

Ao colega de graduação, Lucas Farias Nogueira que me auxiliou durante as etapas de desenvolvimento de software.

Agradeço à minha família por todo suporte e apoio que recebi durante minha graduação, em especial aos meus pais e meu irmão que me incentivam a buscar minha melhor versão e me amparam sempre que não encontro forças para prosseguir.

“Ensinar não é transferir conhecimento, mas
criar as possibilidades para sua própria
produção ou a sua construção”
(Freire, 2002, p. 25)

RESUMO

A Quarta Revolução Industrial, também conhecida como Indústria 4.0, representa a integração de tecnológicas digitais em processos desde operações industriais até tarefas cotidianas. Esta revolução é suportada por pilares como a integração de sistemas, Big Data, robótica autônoma e Internet das Coisas (IoT). Dentre estes, a IoT se destaca devido a premissa de uso da internet com atividades cotidianas agilizando com segurança a coleta, análise e transferência de dados. Para garantir a integridade e eficiência dessa comunicação utiliza-se protocolos específicos de comunicação, como o MQTT (*Message Queuing Telemetry Transport*). Este é ideal para as aplicações de IoT pois é leve, consume pouca energia e possui rotinas robustas de segurança, garantindo uma transferência de informações segura e ágil entre dispositivos. Com o objetivo de aprimorar a experiência dos discentes com as tecnologias presentes na Industria 4.0 este trabalho propõe a modernização com IoT de uma bancada didática de controle de nível. A implementação de uma solução *wireless* em sistemas experimentais busca apresentar os aspectos, requisitos e contribuições que essas tecnologias oferecem para o ambiente prático de laboratório. A solução consiste em um aplicativo móvel desenvolvido na plataforma *Flutter* utilizando linguagem *Dart* que comunica com a bancada experimental através de dois modos: o modo remoto de operação utiliza a internet via protocolo MQTT, permitindo a configuração de parâmetros, acompanhamento do sistema em tempo real e coleta de dados através de um broker; e um modo de configuração e operação local via *Bluetooth Low Energy* (BLE) simplificando a configuração inicial do sistema da bancada, tornando o processo mais intuitivo permitindo que o docente envie as credenciais da rede Wi-Fi e do *broker* de forma intuitiva. Além de atuar como alternativa de conexão para o controle do experimento sem a necessidade de uma conexão com a internet, garantindo que as atividades de laboratório aconteçam mesmo em ambientes com rede instável ou inexistente. Na bancada um microcontrolador ESP 32 gerencia as operações utilizando sua arquitetura *dual-core* para otimizar as tarefas: o primeiro núcleo executa o processamento da malha de controle, à leitura do sensor e o acionamento da bomba, enquanto o segundo núcleo gerencia as comunicações MQTT e BLE. Como resultado observou-se uma conexão estável e de baixa latência entre o aplicativo e a bancada no modo remoto. Adicionalmente o processo de configuração via BLE mostrou-se eficiente e o modo de operação local demonstrou ser uma alternativa funcional, garantindo continuidade e versatilidade ao experimento. A fluidez na transmissão de dados foi uma característica notável em ambos os modos, permitindo a visualização instantânea do impacto das ações de controle no sistema. Conclui-se que a solução com arquitetura híbrida em bancadas experimentais de

controle de nível aprimora o conhecimento técnico em IoT e demonstra na prática a importância da redundância e dá flexibilidade em projetos de engenharia, tornando o aprendizado mais interativo, resiliente e alinhado com os desafios da Indústria 4.0.

Palavras-chave: Indústria 4.0; Internet das Coisas; MQTT; *broker*; microcontrolador ESP 32; Plataforma *Flutter*; Linguagem *Dart*.

ABSTRACT

The Fourth Industrial Revolution, also known as Industry 4.0, represents the integration of digital technologies into processes ranging from industrial operations to everyday tasks. This revolution is supported by pillars such as systems integration, Big Data, autonomous robotics, and the Internet of Things (IoT). Among these, IoT stands out due to its premise of using the internet in everyday activities, securely streamlining data collection, analysis, and transfer. To ensure the integrity and efficiency of this communication, specific communication protocols, such as MQTT (Message Queuing Telemetry Transport), are used. This is ideal for IoT applications because it is lightweight, consumes little energy, and has robust security routines, ensuring secure and agile information transfer between devices. To enhance students' experience with Industry 4.0 technologies, this work proposes the modernization of a level control teaching bench with IoT. The implementation of a wireless solution in experimental systems seeks to demonstrate the aspects, requirements, and contributions that these technologies offer to the practical laboratory environment. The solution consists of a mobile application developed on the Flutter platform using Dart language that communicates with the experimental bench in two modes: remote operation uses the internet via the MQTT protocol, allowing parameter configuration, real-time system monitoring, and data collection through a broker; and a local configuration and operation mode via Bluetooth Low Energy (BLE), simplifying the initial setup of the bench system and making the process more intuitive, allowing the instructor to easily send Wi-Fi network and broker credentials. It also acts as an alternative connection for experiment control without the need for an internet connection, ensuring that laboratory activities continue even in environments with unstable or nonexistent network access. On the bench, an ESP 32 microcontroller manages operations using its dual-core architecture to optimize tasks: the first core performs control loop processing, sensor reading, and pump activation, while the second core manages MQTT and BLE communications. As a result, a stable, low-latency connection was observed between the application and the bench in remote mode. Additionally, the configuration process via BLE proved efficient, and the local operation mode proved to be a functional alternative, ensuring continuity and versatility for the experiment. Fluid data transmission was a notable feature in both modes, allowing instant visualization of the impact of control actions on the system. The conclusion is that the hybrid architecture solution in level control experimental benches enhances technical knowledge in IoT and demonstrates in practice the importance of

redundancy and flexibility in engineering projects, making learning more interactive, resilient, and aligned with the challenges of Industry 4.0.
Keywords: Industry 4.0; Internet of Things; MQTT; Broker; ESP 32 microcontroller; Flutter Platform; Dart Language.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BLE	Bluetooth Low Energy
GND	Ground
GPIO	Entrada/Saída de Propósito Geral
IHM	Interface Homem-Máquina
IoT	Internet das Coisas
LTI	Linear e Invariante no Tempo
MQTT	Transporte de Filas de Mensagem de Telemetria
QoS	Qualidade de Serviço
SoC	Sistema em um Chip
UFU	Universidade Federal de Uberlândia
VCC	Tensão de corrente contínua

SUMÁRIO

1 INTRODUÇÃO

A Industria 4.0 é o marco de uma nova era da manufatura com forte presença de automação e integração das tecnologias digitais conectadas com o processo fabril. Com o avanço tecnológico e a crescente demanda por novas formas de consumir e transferir dados os conceitos da Indústria 4.0 se expandiram para diversas áreas de conhecimento alterando nosso cotidiano e a maneira como vivemos, nos relacionamos e trabalhamos.

Essa transformação se torna visível em interações homem-máquina mais colaborativa , em objetos do dia a dia que se conectam para transferir informações e em fábricas inteligentes com monitoramento autônomo de produção e manutenção. Para construir uma base sólida e bem definida a Industria 4.0 é apoiada em nove pilares tecnológicos: Cibersegurança, Realidade Aumentada, *Big Data*, Robótica Autônoma, Impressão 3D, Simulação, Integração de Sistemas, Computação na Nuvem e Internet das Coisas (IoT do inglês *Internet of Thing*).

A IoT é um dos principais pilares, definindo a interconexão de objetos e equipamentos por meio da internet para fornecer serviços inteligentes. Para viabilizar a comunicação eficiente nestes ecossistemas, o protocolo de comunicação MQTT (*Message Queuing Telemetry Transport*) se destaca por sua arquitetura leve e baixo consumo de energia, garantindo assim uma troca segura e ágil de dados.

Apesar dos avanços dessas tecnologias existe uma lacuna entre o conhecimento teórico e a aplicação prática, especialmente em ambientes de ensino de engenharia. Diante disso, o objetivo deste projeto é apresentar uma solução que integra as tecnologias da IoT a uma bancada de controle de nível experimental. A proposta centraliza a configuração, o controle do sistema físico e a telemetria do experimento em um único aplicativo móvel, buscando oferecer uma experiência de aprendizado que contemple os mecanismos de comunicação, as tecnologias envolvidas e os resultados práticos da aplicação de conceitos da Industria 4.0. O resultado observado foi uma maior clareza sobre o funcionamento prático de tais aplicações e um rápido retorno sobre ações de controle, permitindo ao aluno visualizar como a lei de controle afeta o comportamento do sistema físico.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Este projeto possui o objetivo de desenvolver uma solução IoT completa e robusta para efetuar o controle remoto de um tanque de nível de uma bancada didática de controle utilizada em laboratórios de engenharia.

1.1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral foram definidos os seguintes objetivos específicos para este projeto:

- Construção de uma bancada didática com um compartimento isolado para o sistema eletrônico protegendo os componentes de contato acidental e de respingos de água.
- Desenvolvimento de uma aplicação móvel multiplataforma utilizando o ambiente *Flutter* com a linguagem *Dart*, fornecendo uma interface de fácil manuseio para que o usuário configure, opere e monitore o experimento de forma completamente remota.
- Implementação do hardware do sistema utilizando um microcontrolador ESP32, responsável pela aquisição de dados do sensor ultrassônico, controle da bomba da água e gerenciamento da comunicação de rede.
- Estruturação da comunicação principal (modo remoto) entre o aplicativo móvel e o hardware através de um *broker* MQTT, permitindo a operação do sistema de qualquer localidade via conexão em nuvem.
- Desenvolvimento de uma forma de comunicação secundária via *Bluetooth Low Energy* (BLE) com dupla funcionalidade: para a configuração inicial do sistema e como um modo de operação local, garantindo a continuidade do experimento em caso de falha de conexão principal via MQTT.

1.1.3 JUSTIFICATIVA DO PROJETO

A justificativa do projeto fundamenta-se na necessidade de assimilar o conhecimento teórico das disciplinas de controle linear com as aplicações práticas demandadas pela Industria 4.0. A solução proposta visa modernizar uma bancada didática de controle de nível, um sistema clássico em estudo de controle em engenharias, aplicando tecnologias atuais de IoT e controle

remoto. A escolha de uma plataforma de baixo custo e alta eficiência como o microcontrolador ESP32 alinhada ao desenvolvimento de uma aplicação móvel intuitiva utilizando a linguagem *Dart* permitiu criar uma solução robusta, acessível e alinhada às mais recentes práticas de desenvolvimento de sistemas ciberfísicos.

Além do mais este projeto consolida o conhecimento teórico e prático adquirido ao longo de diversas disciplinas do curso de Engenharia Mecatrônica. A base para a modelagem e controle do sistema foi fundamentada em Controle de Sistemas Lineares onde conceitos de estados, alocação de polos e projeto de observadores foram estudados. A implementação do hardware foi viabilizada pelos conhecimentos de Sistemas Digitais, Eletrônica Básica e Eletrônica Digital que forneceram a base para a seleção do ESP32, o projeto do circuito de interface com os sensores e atuadores e a compreensão das conexões elétricas. Os conceitos de Redes Industriais e Internet das Coisas Industrial foram cruciais para o devido aprofundamento dos conhecimentos a respeito de Industria 4.0, IoT e protocolos de comunicação.

1.1.4 DESENVOLVIMENTO

Este documento apresenta a revisão bibliográfica e os fundamentos do trabalho desenvolvido através das fontes teóricas utilizadas durante a construção do projeto no capítulo 2. A metodologia contendo a descrição das técnicas utilizadas para a construção do aplicativo bem como do sistema de comunicação estabelecido entre o aplicativo, *broker* e ESP32 está apresentada no capítulo 3. Por fim serão apresentados os resultados encontrados durante o período de testes, seguido das conclusões e referências bibliográficas.

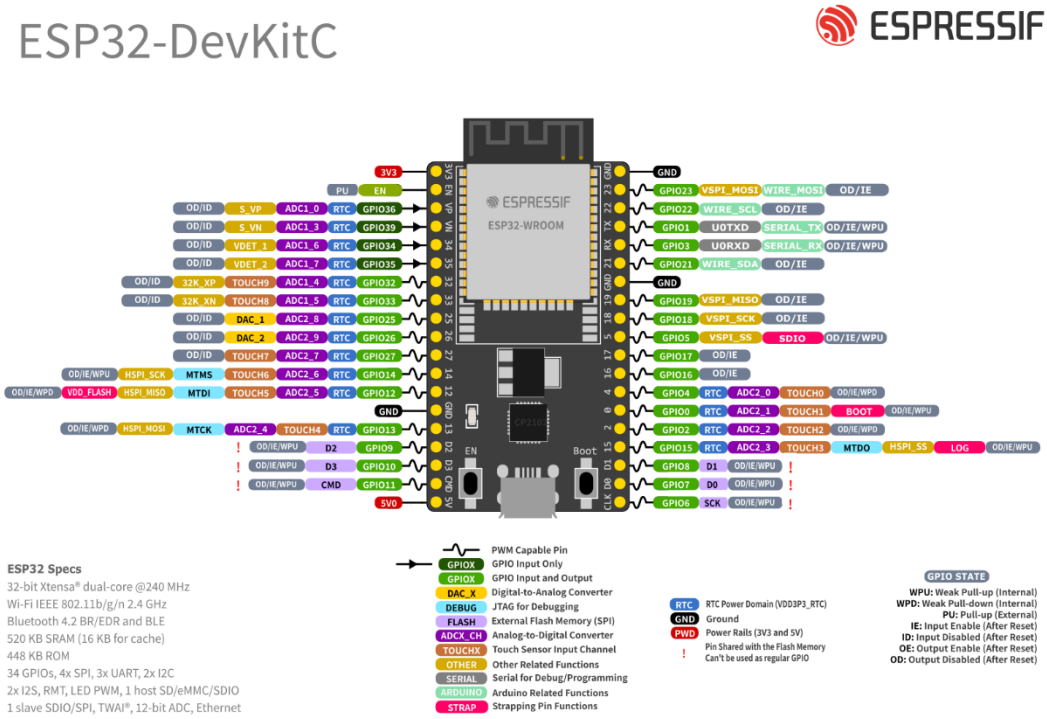
2 REVISÃO BIBLIOGRÁFICA

A primeira seção aborda os conceitos teóricos bem como os dispositivos envolvidos no projeto.

2.1 MICROCONTROLADOR ESP 32

O microcontrolador selecionado para a implementação do hardware do projeto foi o ESP32, especificamente a placa de desenvolvimento ESP32-DevKitC V4. É uma plataforma de baixo custo e alto desempenho desenvolvida pela empresa Espressif, que se consolidou como uma solução ideal para aplicações de Internet das Coisas(IoT) e automação.

Figura 1 - Layout de pinagem do ESP32



Fonte: https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32/esp32-devkitc/index.html

Ele possui um microprocessador dual-core Xtensa LX6 de 32 bits que oferece uma capacidade de processamento robusta para gerenciar múltiplas tarefas simultaneamente, no contexto deste projeto, a arquitetura permitiu que um dos núcleos fosse dedicado ao processamento da lógica de controle de nível enquanto o outro gerencia a comunicação de rede.

Uma característica marcante deste microprocessador está em sua conectividade integrada, a placa possui suporte nativo para Wi-Fi padrão 802.11b/g/n e Bluetooth v4.2 incluindo o modo de baixo consumo BLE.

Fundamental para a arquitetura do projeto, pois viabilizou tanto a comunicação principal via protocolo MQTT através da rede Wi-Fi quanto a comunicação secundária de configuração e contingência via BLE.

Ademais o ESP32 é um Sistema em um Chip (SoC) altamente integrado, possuindo componentes essenciais como antena de rádio frequência, amplificadores de potência e de baixo ruído, filtros e um módulo exclusivo para gerenciamento de energia. Simplificando o desenvolvimento de hardware e reduzindo a necessidade de componentes externos e o espaço físico ocupado pelo circuito eletrônico.

Devido à sua versatilidade, baixo consumo de energia, capacidade de processamento e principalmente à sua conectividade dual (Wi-Fi e *Bluetooth*) o ESP32 é a escolha ideal para atender a todos os requisitos de controle, monitoramento e comunicação da bancada didática de nível.

2.2 PROTOCOLO DE COMUNICAÇÃO MQTT

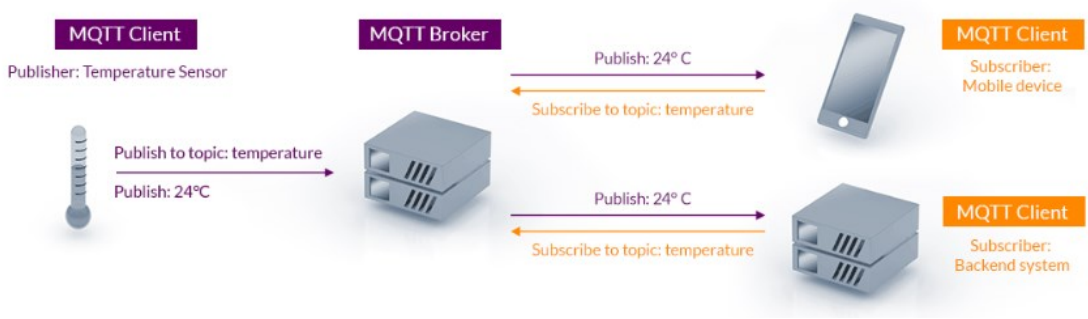
Entre os mais variados tipos de protocolos de comunicação da Internet das Coisas, o *Message Queue Telemetry Transport* (MQTT) demonstra uma característica notável de permitir um ótimo desempenho de transmissão em situações de recursos limitados (Maschietto, 2021).

Desenvolvido na década de 90 sobre a pilha de protocolos TCP/IP com objetivo de criar um meio de comunicação leve e eficiente para redes com baixa largura de banda e conectividade não confiável, características comuns em sistemas embarcados e aplicações de IoT. A arquitetura do MQTT é baseada em um modelo de comunicação assíncrono e desacoplado (Maschietto, 2021).

Essa arquitetura é sustentada por três componentes centrais:

- Publicador (*Publisher*): dispositivo ou cliente que envia as mensagens;
- Assinante (*Subscriber*): dispositivo ou cliente que registra para receber as mensagens de seu interesse
- *Broker*: um servidor central que recebe as mensagens dos publicadores e as encaminha para os assinantes correspondentes, filtrando-as com base em tópicos.

Figura 2 - Representação da arquitetura MQTT de publicador e assinante



Fonte: https://mqtt.org/

Diferente de um modelo cliente-servidor, no MQTT os dispositivos não se comunicam diretamente; um publicador envia uma mensagem para um tópico específico no *broker*, os assinantes que tiverem interesse na informação se inscrevem neste tópico e recebem a mensagem do *broker* assim que ela é publicada. Os tópicos são hierárquicos permitindo assim uma organização lógica e flexível dos fluxos de dados

Para garantir a confiabilidade na entrega das mensagens o MQTT implementa três níveis de Qualidade de Serviço (QoS). Isso permite ao desenvolvedor balancear entre velocidade e garantia de entrega (Maschietto, 2021).

Os três níveis QoS são:

- QoS 0 (*At most once* – No máximo uma vez): É o modo mais rápido, a mensagem é enviada apenas uma vez sem qualquer confirmação de recebimento. Ideal para dados de telemetria não críticos, onde é aceitável a perda ocasional de pacotes.
- QoS 1 (*At least once* – Pelo menos uma vez): Este modo garante que a mensagem seja entregue ao *broker* pelo menos uma vez, o remetente armazena a mensagem e a reenvia até receber uma confirmação. Pode acontecer a duplicação de mensagens em caso de falha na rede.
- QoS 2 *(Exactly once* – Exatamente uma vez): É o modo mais robusto e lento, utiliza um *handshake* de quatro etapas para garantir que a mensagem seja entregue ao destinatário exatamente uma vez, sem duplicadas. Usado em aplicações críticas onde a perda ou duplicação de dados é inaceitável.

O *broker* pode ser configurado para armazenar a última mensagem publicada em um tópico com a *flag* "*retain*", quando um novo cliente se inscreve neste tópico ele recebe imediatamente a última mensagem retida sem precisar esperar por uma nova publicação.

Além disso ele também possui sessões persistentes em que em caso de desconexão do cliente o *broker* armazena suas inscrições e as mensagens de QoS 1 e QoS 2 que não puderam ser entregues, ao se reconectar a sessão é retomada e o cliente recebe as mensagens pendentes, criando assim uma conexão durável e resiliente a falhas.

2.3 DESENVOLVIMENTO MÓVEL COM *FLUTTER* E *DART*

Para o desenvolvimento da interface homem-máquina (IHM) do projeto foi escolhido o ecossistema *Flutter* que utiliza a linguagem de programação *Dart*, ambas as tecnologias são desenvolvidas e mantidas pelo Google e representam uma abordagem moderna para a criação de aplicações multiplataformas. O *Flutter* se destaca por sua arquitetura que permite a criação de interfaces de usuário ricas e fluidas, compiladas nativamente para plataformas móveis, web e desktop a partir de uma única base de código (*Flutter*, 2025).

Flutter é um framework de UI (*User Interface*) de código aberto que permite o desenvolvimento de aplicações compiladas nativamente para mobile (tanto *Android* quanto para iOS), web, desktop (Windows, macOS e Linux) e sistemas embarcados a partir de uma única base de código. Ele compila o código *Dart* diretamente para o código de máquina nativo não dependendo de pontes de comunicação ou WebViews para fazer isso, desse modo garante que seu desempenho gráfico seja superior aos demais ecossistemas de desenvolvimento, além disso possui uma comunicação direta com os recursos do dispositivo ocasionando uma interface fluida e responsiva, essencial para o monitoramento de dados em tempo real.

Garantindo uma UI expressiva e flexível o *Flutter* faz uso do seu próprio motor de renderização de alta performance, o *Impeller*, que desenha cada pixel na tela concedendo aos desenvolvedores controle total sobre a interface, ademais ele possui uma funcionalidade denotada de *Hot Reload* (Recarregamento Rápido) onde os desenvolvedores podem injetar o código fonte modificado na máquina virtual em execução, atualizando a aplicação instantaneamente sem a necessidade de reiniciá-la.

Dart é a linguagem de programação otimizada para o cliente, sendo projetada especificamente para a construção rápida de interfaces de usuário em diversas plataformas. O *Dart* possui um sistema de tipos de segurança contra nulos verificado em tempo de compilação, eliminando uma classe inteira de exceções de ponteiro nulo. Ele possui suporte de primeira classe, características integradas da própria linguagem de programação, para programação assíncrona através das palavras-chave "*async*" e "*await*" que são utilizadas em caso do código lidar com operações futuras, para este projeto essa funcionalidade foi crucial para gerenciar a comunicação continua e não bloqueante com o *broker* MQTT.

2.4 FUNDAMENTOS DO SISTEMA DE CONTROLE

A estratégia de controle implementada na bancada didática baseia-se na moderna teoria de controle por Espaço de Estados, uma poderosa abordagem para efetuar análise e prototipagem de controladores para sistemas lineares com múltiplas entradas e saídas.

A principal estratégia da representação por espaço de estados é a de descrever um sistema dinâmico de ordem n por meio de um conjunto de n equações diferenciais de primeira ordem (Pedro Augusto, 2024). O conjunto mínimo de variáveis que define o estado interno do sistema em um instante de tempo é chamado de variáveis de estado, o vetor formado por essas variáveis constitui o vetor de estados. Um sistema linear e invariante no tempo (LTI) pode ser descrito matematicamente pela equação de estado:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$

Onde:

x(t) é o vetor de estados de dimensão *n x 1*.

u(t) é o vetor de entrada / sinal de controle de dimensão *r x 1*.

A é a matriz de estado *n x n* que descreve a dinâmica interna da planta.

B é a matriz de entrada *n x r* que relaciona os estados com a saída mensurável.

O LTI também pode ser descrito através da equação de saída:

$$y(t) = Cx(t) + Du(t) \tag{2}$$

Onde:

y(t) é o vetor de saída de dimensão *m x 1*.

x(t) é o vetor de estados de dimensão *n x 1*.

u(t) é o vetor de entrada / sinal de controle de dimensão *r x 1*.

C é a matriz de saída *m x n* que relaciona os estados com a saída mensurável.

D é a matriz de transmissão direta *m x r* que representa o acoplamento direto da entrada na saída.

Uma das técnicas mais eficazes para o projeto de controladores no espaço de estados é a alocação de polos por meio da realimentação de estados. Se todos os estados do sistema forem mensuráveis é possível realimentá-los para o sinal de controle (Ogata, 2010).

Podemos definir a lei de controle de realimentação como:

$$u(t) = -Kx(t) \quad (3)$$

Onde :

K é a matriz de ganhos de realimentação de estados *r x n*.

x(t) é o vetor de estados de dimensão *n x 1*.

Ao aplicar esta lei de controle a dinâmica do sistema em malha fechada se torna:

$$\dot{x}(t) = (A - BK)x(t) \quad (3)$$

Em que a estabilidade e a resposta transitória do sistema em malha fechada são determinadas pelos autovalores da matriz *(A – BK)* que são os polos da malha fechada.

Se o sistema dos completamente controlável é possível através da escolha adequada da matriz *k* alocar esses polos em qualquer posição desejada no plano complexo, garantindo assim o desempenho desejado para o sistema (Ogata, 2010).

Na prática não é possível medir todas as variáveis de estado, seja por custo ou por limitações físicas. Para solucionar isso projeta-se um observador de estados, um sistema auxiliar que fornece uma estimativa do vetor de estados real com base nas medições da entrada e da saída da planta.

O observador funciona como um modelo da planta, corrigindo continuamente sua estimativa com base no erro entre a saída medida e a saída estimada.

Podemos descrever a dinâmica do observador como:

$$\dot{x}(t) = (A - K_e C)\tilde{x}(t) + Bu(t) + K_e y(t) \quad (4)$$

Onde:

Matriz K_e é o ganho do observador.

No projeto desenvolvido, o objetivo é alocar os autovalores da matriz de erro do observador $(A - K_e C)$ em posições que garantam uma convergência rápida e estável da estimativa para o estado real onde $e(t) \to 0$.

A lei de controle é aplicada utilizando o estado estimado resultando em $u(t) = -K\tilde{x}(t)$, formando um sistema controlador-observador robusto.

A implementação do controle na bancada didática partiu de um modelo matemático de primeira ordem que relaciona a tensão aplicada na bomba, u(t), com a altura da coluna de água no tanque, h(t).

Este modelo, obtido experimentalmente através da resposta ao degrau, é descrito pela seguinte equação diferencial:

.

$$\dot{h}(t) = -0{,}006h(t) + 0{,}002u(t) \quad (5)$$

Para aplicar a teoria de controle por espaço de estados, o primeiro passo consiste em converter esta equação diferencial para a representação matricial $\dot{x}(t) = Ax(t) + Bu(t)$.

Sendo um sistema de primeira ordem, o vetor de estados x(t) possui uma única variável, a própria altura h(t).

Dessa forma, obtemos:

- Vetor de Estado: $x(t)=[h(t)]$
- Matriz de Estado (A): $A=[-0{,}006]$
- Matriz de Entrada (B): $B=[0{,}002]$

Com o sistema representado em espaço de estados, o próximo passo foi projetar o vetor de ganhos da realimentação de estados, K.

Para isso, utilizou-se a técnica de alocação de polos, cujo objetivo é posicionar os autovalores da malha fechada em locais pré-determinados que garantam a estabilidade e o desempenho desejado.

3 METODOLOGIA

Esta seção descreve detalhadamente os materiais, as tecnologias e os procedimentos empregados para o desenvolvimento da solução de controle remoto da bancada didática.

Fornecendo uma visão clara de arquitetura do projeto, implementação do hardware e do software, assim como os protocolos de comunicação utilizados, permitindo a compreensão e reprodutibilidade do trabalho.

3.1 ESTRUTURAÇÃO DO PROJETO

A solução consiste em uma arquitetura de sistema ciberfísico composta por três elementos centrais que interagem para permitir a operação remota e local: a bancada física (*hardware*), o aplicativo móvel (*front-end*) e a comunicação de rede.

A comunicação principal para operação do experimento é baseada no protocolo MQTT que utiliza um servidor intermediário (*broker*) para gerenciar a troca de mensagens, desacoplando o cliente (aplicativo) do dispositivo controlado (ESP32).

A comunicação secundária utiliza a tecnologia BLE para transferência de informações entre o aplicativo e a bancada através do *Bluetooth*, esta tecnologia também é responsável pela configuração inicial do sistema, onde são enviados os parâmetros de conexão da rede Wi-Fi e do *broker*.

No aplicativo será possível controlar a altura da coluna d´água em um tanque através da inserção de parâmetros na lei de controle de espaço de estados definida para este sistema, a referência e o valor de entrada que neste caso é a leitura efetuada pelo sensor ultrassônico da altura da coluna d´água.

O aplicativo possui ao todo quatro modos de uso classificados de acordo com o tipo de usuário e tipo de operação, onde um usuário aluno pode se comunicar com a bancada através do modo local ou remoto e um usuário professor pode acompanhar os experimentos no modo local ou remoto.

A montagem final da bancada incluindo a disposição dos componentes eletrônicos, o reservatório e o isolamento do sistema elétrico para proteção conta respingos de água é apresentada nas Figura 4 e 5.

Figura 3 - Diagrama da arquitetura geral do sistema

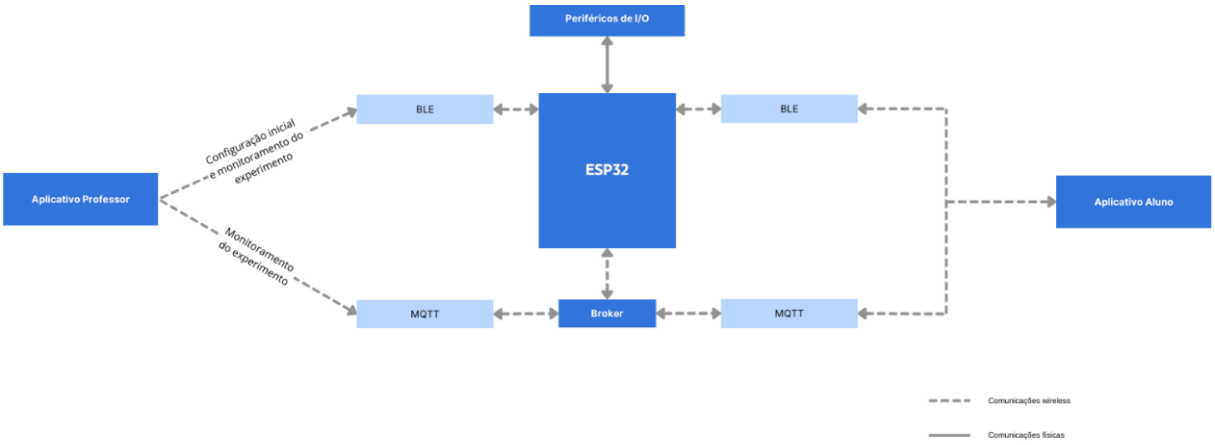


Figura 4 - *Overview* da bancada durante a fase de testes com o compartimento do circuito eletrônico aberto

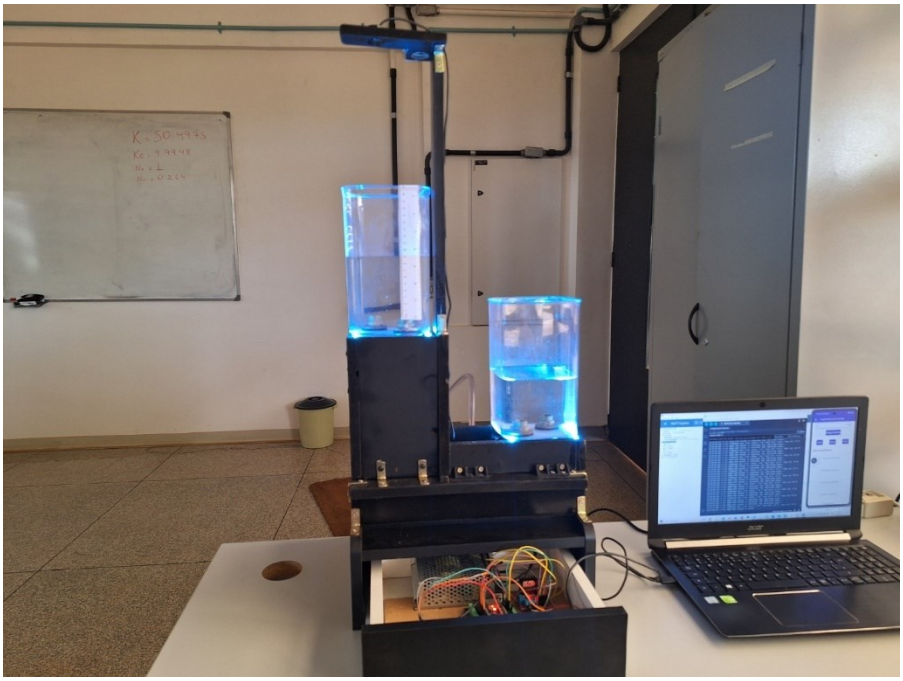
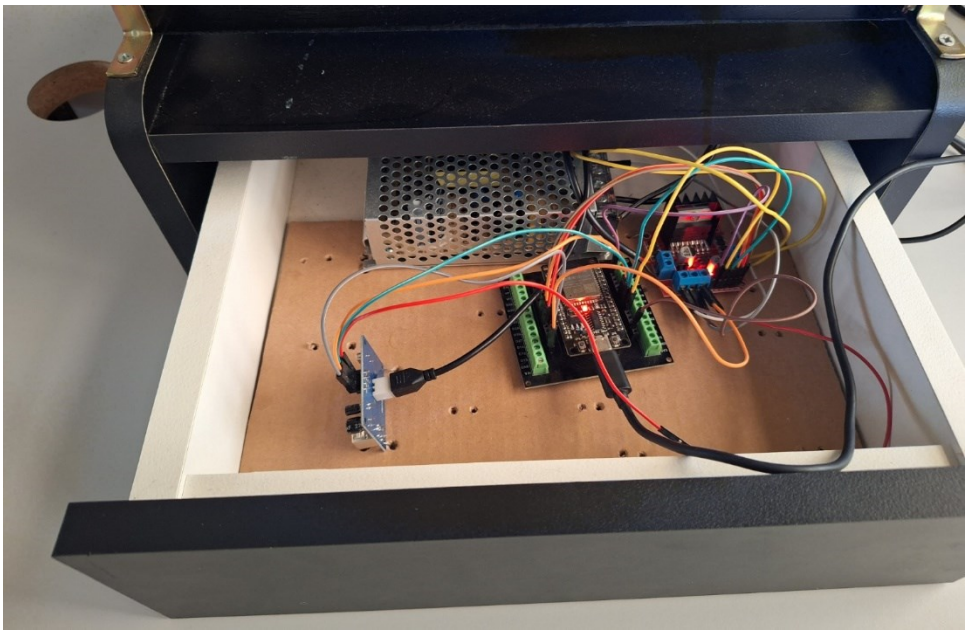


Figura 5 - Compartimento do sistema eletrônico composto em destaque



3.2 DESENVOLVIMENTO DO *HARDWARE*

O hardware da bancada foi montado utilizando componentes de baixo custo e alta disponibilidade, contendo como unidade central de processamento o microcontrolador ESP32.

3.2.1 COMPONENTES ELETRÔNICOS

Foram utilizados os seguintes componentes na montagem da bancada:

- Microcontrolador ESP32-DevKitC V4: Responsável por toda a lógica de controle, leitura de sensores e comunicação.
- Sensor Ultrassônico JSN-SR04T: Utilizado para medir a distância até a superfície da água, calcula o nível do tanque através da aplicação de um filtro que coleta 25 medições e efetua a média ponderada delas, entregando para o ESP32 valores mais precisos da altura da coluna de água.
- Driver de Motor Ponte H L298N: Módulo de potência utilizado para controlar a bomba de água (Saída A) e a fita LED (Saida B) a partir dos sinais de baixa corrente do ESP32.
- Bomba de Água Submersível 12V: Atuador responsável por encher o tanque de nível.

- Fita de LED 12V: Utilizada como feedback visual para indicar o estado atual do sistema.
- Fonte de Alimentação Externa 12V: Fornecimento de energia necessária para o ESP32,acionamento da bomba e da fita LED.

3.2.2 MONTAGEM E CONEXÕES

A interconexão do componentes foi realizada conforme o esquema elétrico detalhado, todos os componentes compartilham um referencial de terra (GND) comum para garantir a estabilidade dos sinais elétricos.

As tabelas a seguir apresentam as ligações feitas no projeto para transferência de dados e alimentação do sistema.

Tabela 1 - Tabela de Conexões de Hardware

PINO DO ESP32	COMPONENTE	PINO DO COMPONENTE	FUNÇÃO
5	Sensor Ultrassônico	Trig	Envia o pulso do sensor
18	Sensor Ultrassônico	Echo	Recebe o retorno do pulso do sensor
23	Driver do Motor (L298N)	ENA	Controle de velocidade da bomba (PWM)
16	Driver do Motor (L298N)	IN1	Controle de direção 1 da bomba
17	Driver do Motor (L298N)	IN2	Controle de direção 2 da bomba
25	Drive do Motor (L298N)	ENB	Controle de brilho da fita de LED (PWM)
26	Driver do Motor (L298N)	IN3	Controle de direção 1 da fita de LED
27	Driver do Motor (L298N)	IN4	Controle de direção 2 da fita de LED

Tabela 2 - Alimentação do sistema

COMPONENTE	CONEXÃO	DESCRIÇÃO
Bomba de Água	Saídas do Driver L298N	Conexão nos bornes OUT1 e OUT2
Fita de LED 12V	Saídas do Driver L298N	Conexão nos bornes OUT3 e OUT4
Driver do Motor L298N	Fonte de Energia Externa	Conexão entre o borne +12V no polo 12V e o borne GND do driver no GND da fonte.
Sensor Ultrassônico	Pino de Alimentação	Conexão do pino VCC do sensor no borne 5V e o pino do GND no borne GND do driver.
ESP32	Pino de Alimentação	Conexão do pino VCC do sensor no borne 5V e o pino do GND no borne GND do driver.

Uma característica importante a respeito das ligações foi o GND em comum entre todos os componentes eletrônicos, garantindo que todos tenham o mesmo referencial de 0 Volts evitando comportamentos estranhos.

3.2.3 *FIRMWARE* DO ESP32

O *firmware* embarcado no ESP32 foi desenvolvido utilizando linguagem C++ sobre o *framework* Arduino. Contando com o auxílio do sistema operacional de tempo real *FreeRTOS* para o gerenciamento de tarefas concorrentes, aproveitando a arquitetura *dual-core* do ESP32 para dividir as responsabilidades e otimizar o desempenho.

O núcleo *0* foi dedicado exclusivamente à tarefa *controlTask* que executa a malha de controle em intervalos precisos de 100ms, além disso realiza a leitura do sensor, aplica o filtro de média móvel para suavizar o sinal e calcula a ação de controle com base na lei de realimentação de estados.

O núcleo *1* foi dedicado à tarefa *commsTask* que gerencia todas as operações de comunicação e *feedback* visual. Esta segunda tarefa é responsável por mantes as conexões Wi-

Fi e MQTT ativas, publicar os dados de telemetria (nível, tensão, tempo, *status*) para o *broker* controlar a intensidade e os efeitos da fita de LED com base no estado atual do sistema.

As principais bibliotecas utilizadas foram as *PubSubClient* para a comunicação MQTT e a *BLEDevice* para a configuração inicial via *Bluetooth*.

3.3 DESENVOLVIMENTO DO APLICATIVO MÓVEL

A interface de usuário foi desenvolvida como um aplicativo móvel multiplataforma com uma experiência consistente e acessível.

3.3.1 TECNOLOGIAS UTILIZADAS

O aplicativo foi construído utilizando o ecossistema Google através do framework Flutter com linguagem Dart.

Foram utilizadas bibliotecas de código aberto para acelerar o desenvolvimento e implementar funcionalidades complexas, como a *mqtt_client* para a comunicação MQTT, *Flutter_blue_plus* para a comunicação BLE, *fl_chart* para a plotagem de gráficos em tempo real e *google_fonts* para a customização da tipografia.

Para validar a estrutura de tópicos e a integridade dos dados trafegados na rede foi utilizado o MQTT *Explorer*, um cliente MQTT gráfico de código aberto que monitora, depura e visualiza em tempo real as mensagens trocadas no *broker* MQTT.

3.3.2 ESTRUTURA E TELAS

O aplicativo foi estruturado para guiar o usuário através de um fluxo lógico com telas distintas para cada perfil e etapa do processo.

3.3.2.1 TELA DE BOAS-VINDAS

Tela de carregamento inicial que introduz o usuário ao aplicativo.

Figura 6 - Tela de boas-vindas do aplicativo

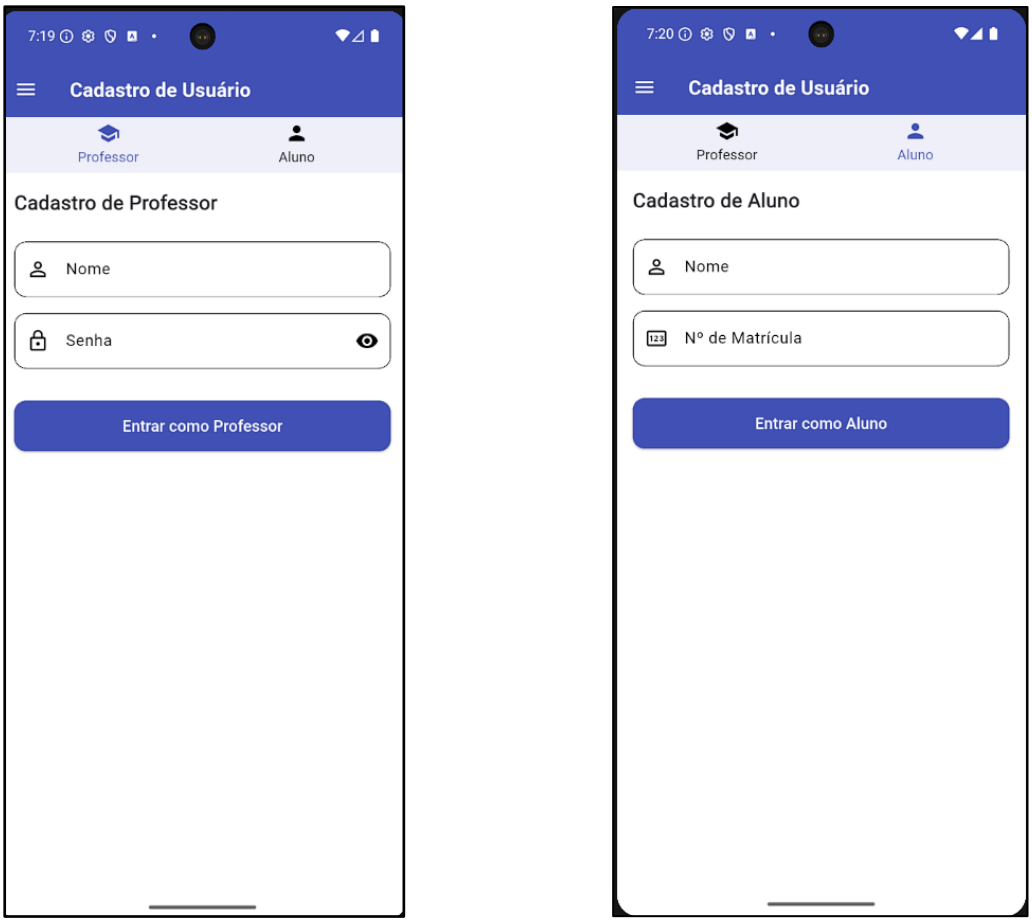
Nesta tela é feito o carregamento inicial do aplicativo que contempla as bibliotecas utilizadas e as funções que serão executadas em cada uma das telas.

Também é iniciado o processo de envio do *JSON* via *BLE* para a bancada, garantindo assim que o serviço esteja disponível para o usuário quando for requisitado.

3.3.2.2 TELA DE LOGIN

Ponto de entrada onde o usuário se identifica como professor ou aluno, direcionando-o para o fluxo correto.

Figura 7 - Telas de cadastro de usuário. a) Tela de *login* do professor. b) Tela de cadastro de aluno



Nesta tela temos duas opções de escolha de perfil, o professor ou o aluno. Para o professor será necessário um loguin com nome e senha que foram pré-definidos no código, garantindo assim que apenas pessoas autorizadas consigam configurar o sistema da bancada.

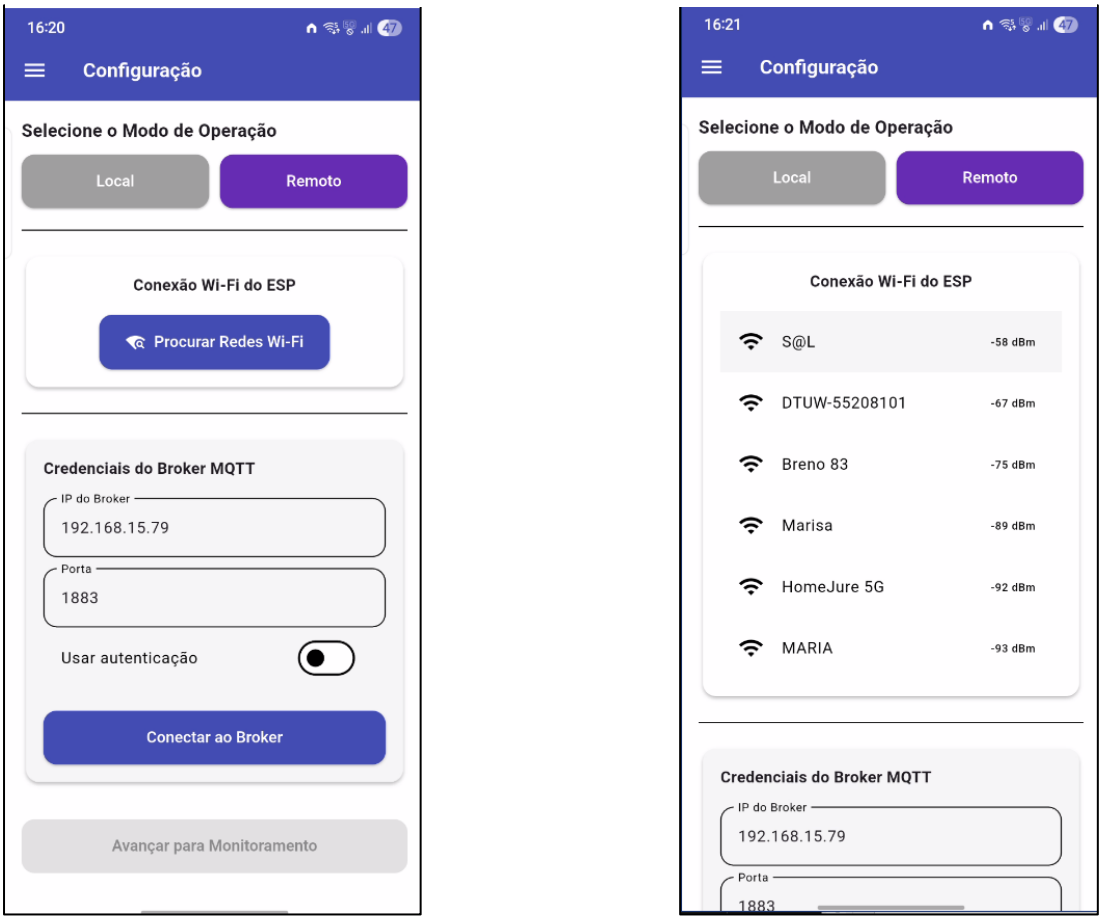
Para o perfil do aluno será requisitado o nome e o número de matrícula, este último será usado como um identificador no restante do aplicativo.

3.3.2.2.1 FLUXO DO PROFESSOR

- CONFIGURAÇÃO DA BANCADA

Tela onde o professor se conecta via BLE com a bancada para enviar as credenciais da rede *Wi-Fi* e do *broker*.

Figura 8 - Tela de configuração da conexão wireless. a)Tela após a seleção do modo remoto. b) Tela após o ESP32 procurar pelas redes disponíveis

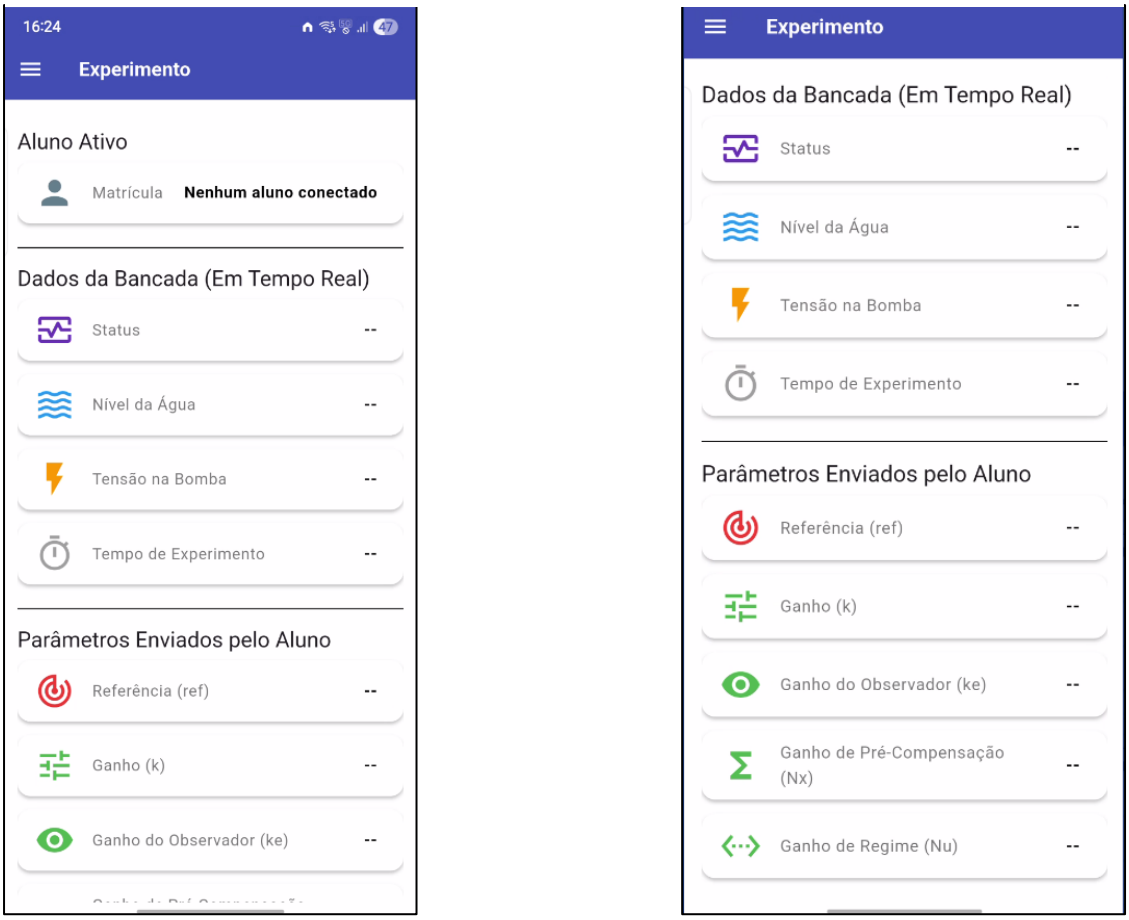


Nesta tela o professor irá requisitar que o ESP procure pelas redes *Wi-Fi* disponíveis na frequência de 2.4 Hz e retorne para o aplicativo uma lista com as redes encontradas através da comunicação *BLE*. Ao selecionar a rede desejada o professor insere a senha, em caso de sucesso ele deve agora inserir as credenciais do broker que por padrão são *IP* e porta mas também podem ser inseridos usuário e senha caso o broker esteja configurado com tais recursos. As credenciais do broker também são enviadas via *BLE* para o ESP.

- MONITORAMENTO

Tela que se conecta ao broker MQTT e se inscreve dinamicamente nos tópicos do aluno ativo no modo remoto, já no modo local o ESP utiliza o BLE para enviar um JSON com os dados do experimento para visualizar em tempo real todos os parâmetros e dados da execução atual.

Figura 9 - Tela de monitoramento do professor



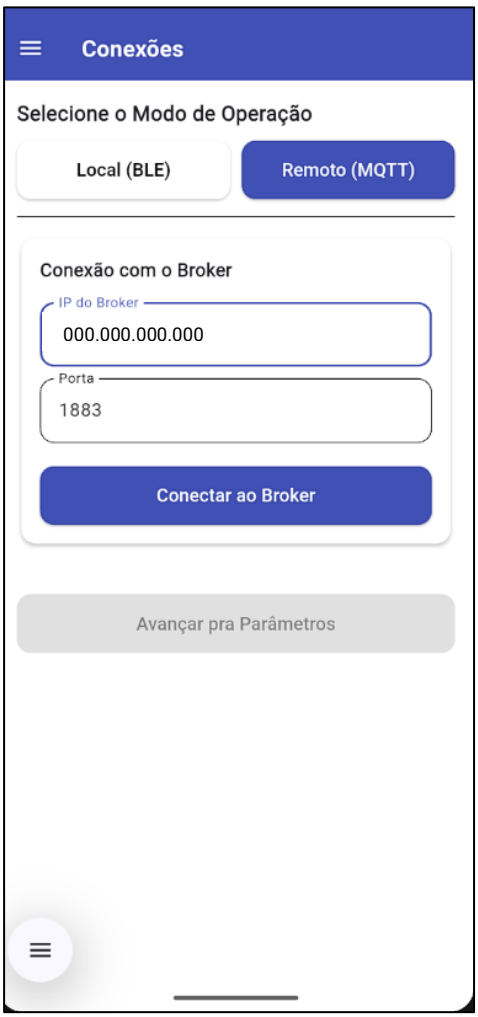
Ao chegar nesta tela a conexão do ESP32 com o broker foi estabelecida com sucesso, a bancada agora está em um estado de *Stand-by* aguardando o aluno publicar os valores de cada parâmetro em seu devido tópico. Graças ao identificador gerado através do número de matrícula fornecido pelo aluno na tela de cadastro a bancada irá utilizá-lo como guia e poderá efetuar o processo de subscrever aos tópicos corretos referentes ao aluno conectado no momento do experimento.

3.3.2.2.2 FLUXO DO ALUNO

- TELA DE CONFIGURAÇÃO DAS CONEXÕES

O aluno insere o endereço do *broker* MQTT para estabelecer uma conexão com a bancada.

Figura 10 - Tela de conexão com o broker no fluxo do aluno

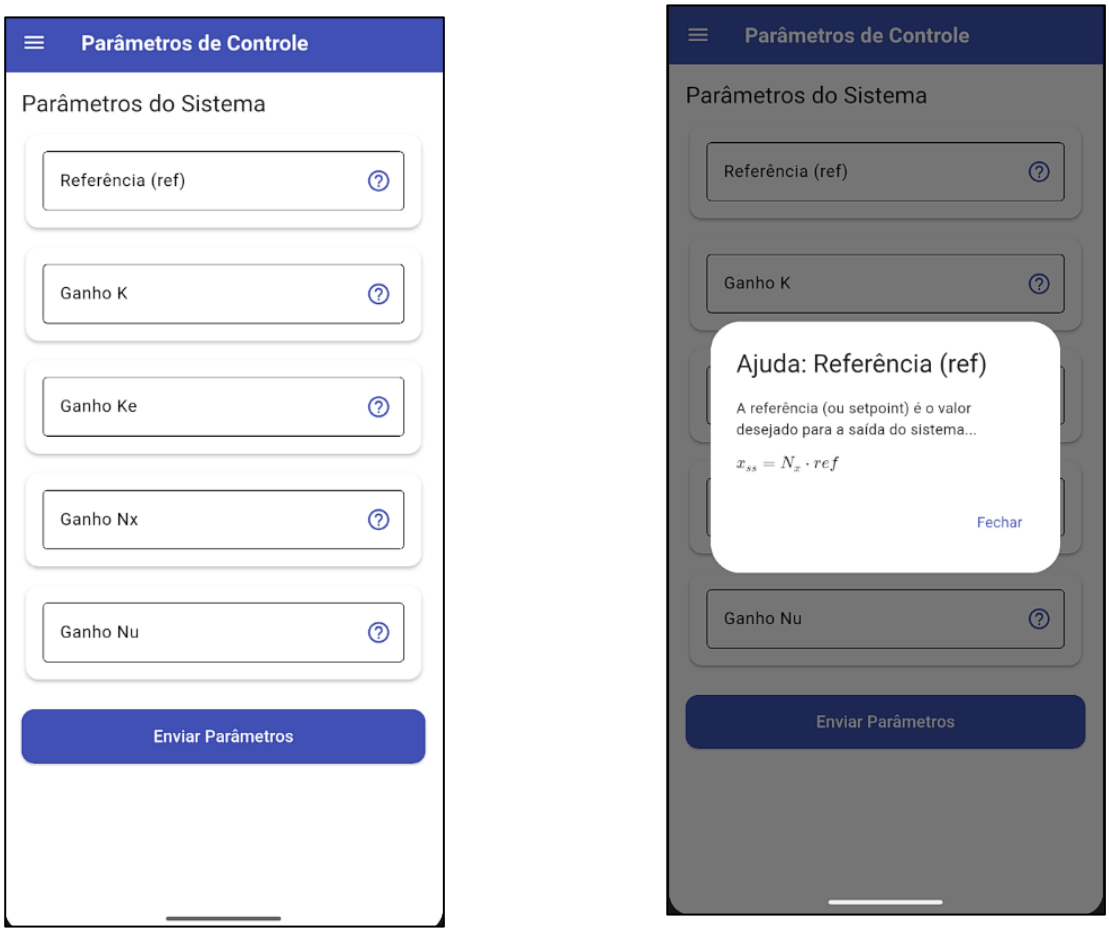


Como foi descrito no tópico acima o número de matrícula fornecido pelo aluno será guardado pelo aplicativo como uma variável global e será publicado em um tópico no broker denominado "/entradaid". A bancada ao receber essa mensagem com o número de matrícula irá efetuar o processo de inscrição nos tópicos referentes que pertencem ao aluno que irá enviar os parâmetros de controle, ou seja, ela ficara pronta para receber os dados do aluno atual.

- PARÂMETROS DE CONTROLE

Tela principal onde o aluno insere os ganhos de controle (*K*, *Ke*, *Nx*, *Nu*) e a referência. Na tela contém botões em cada *input* para que o usuário tenha um direcionamento de como calcular cada parâmetro.

Figura 11 - Tela de parâmetros de controle. a) Campos de *input* onde o aluno irá inserir cada uma dos valores. b) Pop-ups de ajuda em que o aluno pode consultar um breve texto explicativo sobre cada parâmetro.

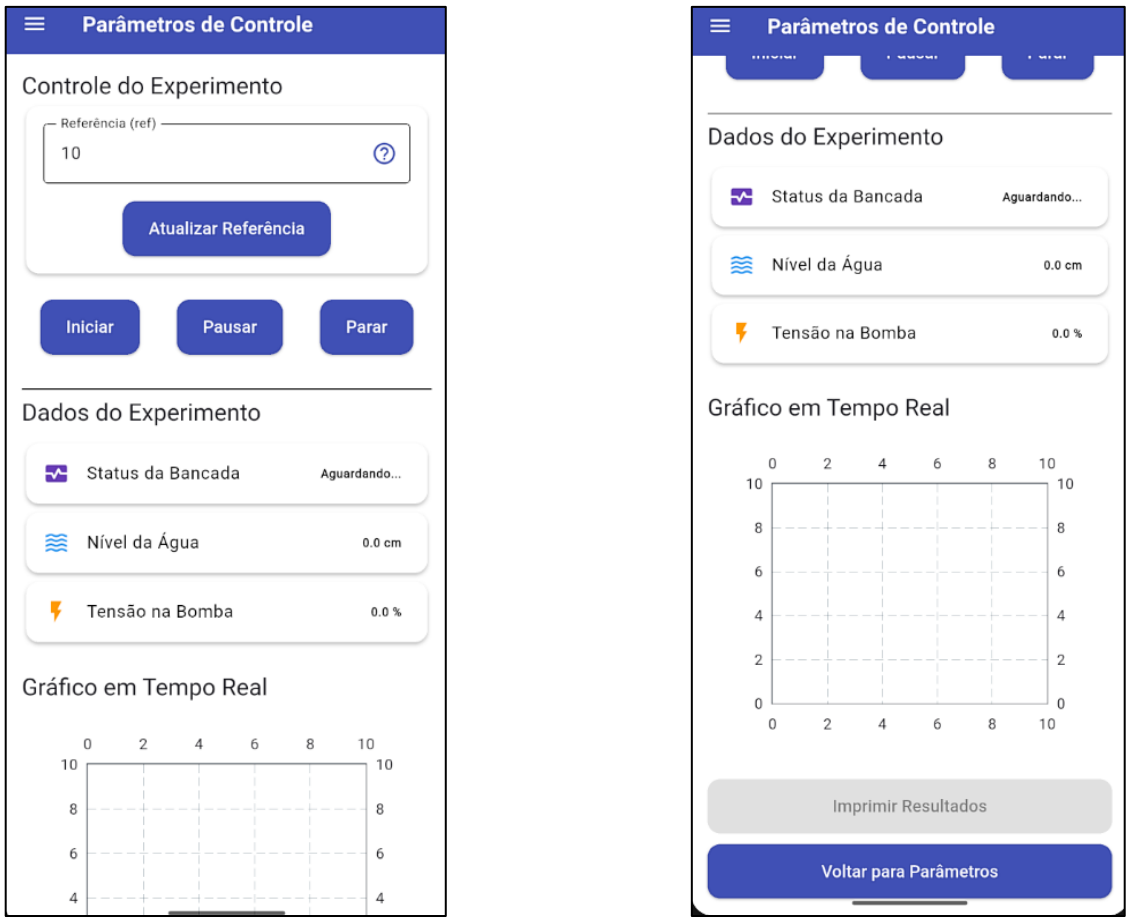


A interface principal do aluno detalhada na Figura 11 permite a inserção dos ganhos calculados, cada valor inserido é armazenado como um *float* para que seja publicado no formato esperado pelo ESP32. Após o envio o aluno é direcionado para a tela de controle onde o experimento é executado

- TELA DE EXPERIMENTO

Na tela de controle onde o aluno inicia, pausa e para o experimento e poderá visualizar também os dados recebidos de tensão, nível da água, tempo e tensão que a bancada publica em tópicos específicos durante o experimento. A tela também conta com um gráfico que é atualizado em tempo real e ilustra os dados recebidos de maneira didática e de fácil entendimento. Ao final da tela o aluno pode salvar os dados do experimento enviados pela bancada em um arquivo no formato txt.

Figura 12 - Tela de Experimentos. a) Parte superior onde estão os botões de controle de atualizar valor da referência, iniciar, pausar e parar. b) Parte inferior com informações em tempo real os valores do experimento.



Ao final da tela é possível imprimir os resultados, neste processo que está disponível após a finalização do experimento (ao pressionar o botão parar após execução bem sucedida do experimento) onde todos os valores de tensão, tempo e altura da coluna de água são colocados em um arquivo no formato *text file* e podem ser salvos no dispositivo.

4 RESULTADOS

Nesta seção são apresentados os resultados práticos obtidos a partir da implementação e dos testes da solução de controle, a validação do sistema foi divida em três etapas principais: a verificação da montagem e da comunicação do *hardware*, a confirmação dos fluxos de usuário no aplicativo móvel e a análise do desenvolvimento do sistema de controle de nível em malha fechada.

4.1 VALIDAÇÃO DA COMUNICAÇÃO E DA ARQUITETURA HÍBRIDA

A primeira etapa consistiu em validar a arquitetura de comunicação hibrida (BLE e MQTT), o fluxo de configuração realizado pelo perfil do professor. A validação acontece por meio de *feedbacks* em formato de notificação no aplicativo, logs no monitor serial do IDE do Arduino e do MQTT *Explorer*. Utilizando a comunicação BLE as credenciais da rede Wi-Fi e do broker MQTT foram enviadas ao ESP32, o microcontrolador estabeleceu a conexão com a rede e com o broker MQTT, confirmando a eficácia do método para o *setup* inicial do sistema de forma segura e intuitiva. Uma vez conectado o ESP32 iniciou a operar no modo de telemetria via MQTT, testes de conectividade demonstraram uma comunicação estável e com baixa latência entre a bancada e o *broker*, com a *commsTask* do firmware gerenciando eficientemente a reconexão automática em casos de instabilidade da rede, garantindo a robustez da operação remota.

Figura 13 - Tela de configuração da conexão do professor. a) Tela em seu estado natural. b) *Drawer* com retornos visuais dentro do *app*

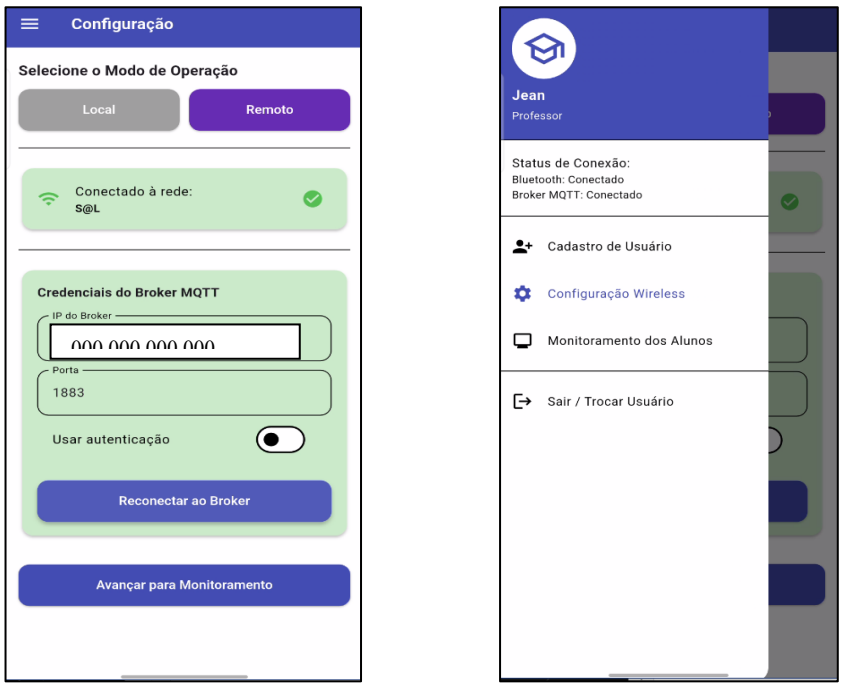
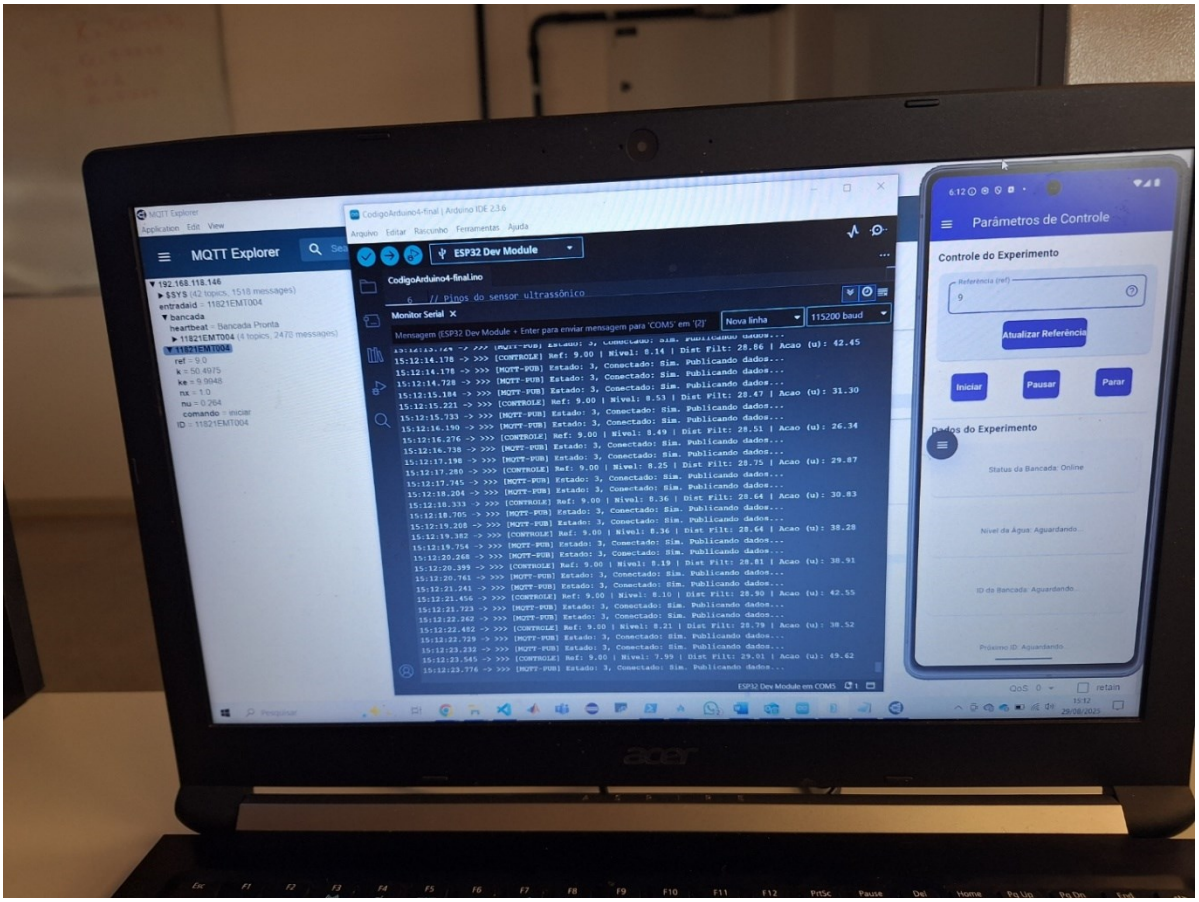


Figura 14 - Logs de confirmação do IDE do arduino com as mensagens de recebimento e envio de dados, assim como do MQTT *Explorer*.



4.2 ANÁLISE DE FLUXOS DE USUÁRIO E DA INTERFACE

Os fluxos de usuário tanto para o professor quanto para o aluno foram validados em sequência. Após a configuração da bancada pelo professor, em outro dispositivo foi efetuado o *login* do aluno para iniciar o experimento, o aplicativo do aluno se conectou com sucesso ao broker e publicou sua matrícula no tópico *entradaid*, imediatamente o ESP32 confirmou o recebimento e se inscreveu nos tópicos específicos daquele aluno publicando a confirmação no tópico bancada/*heartbeat*, o que permitiu ao aplicativo do aluno avançar para a tela de parâmetros como pode ser visualizado na Figura 14.

Figura 15 - Tela de experimentos com os retornos visuais das conexões

A tela de monitoramento do professor também foi validada, conectada ao mesmo *broker* ela detectou a matrícula publicada no tópico *entradaid* e se inscreveu dinamicamente em todos os tópicos associados àquele aluno, exibindo em tempo real tanto os parâmetros enviados pelo aluno quanto os dados de telemetria enviados pela bancada, como pode ser visualizado na Figura 13.b).

4.3 DESEMPENHO DO SISTEMA DE CONTROLE

Para a validação final um experimento de controle de nível foi conduzido, os parâmetros da lei de controle por realimentação de estados, previamente calculados, foram inseridos na tela de parâmetros do aplicativo do aluno e enviados para o ESP32.

Tabela 3 - Parâmetros de controle utilizados no experimento

PARÂMETRO	DESCRIÇÃO	VALOR ESPERADO
K	Ganho de regulador de estados	50.4975
Ke	Ganho do observador de estados	9.9948
Nx	Fator de escala de estado	1.0
Nu	Fator de escala de entrada	0.264

Após o envio dos parâmetros foi definida uma referência de 10cm para o nível da água e o comando *iniciar* foi enviado, o sistema respondeu prontamente acionando a bomba para elevar o nível do tanque ao mesmo tempo o aplicativo exibiu em tempo real a resposta do sistema através dos *cards* de telemetria e do gráfico.

Observou-se que o controlador foi capaz de levar o nível da água até o valor de referência e estabilizá-lo com sucesso. A implementação do filtro de média móvel no *firmware* do ESP32 mostrou-se crucial para suavizar as leituras do sensor ultrassônico resultando em uma ação de controle mais estável e eliminando os "trancos" na bomba que foram observados em testes preliminares. O sistema demonstrou ser robusto, mantendo o nível próximo à referência mesmo com as pequenas oscilações naturais do líquido.

Todos os códigos do projeto estão disponíveis no *Git*H*ub* do MAPL e podem ser acessados através do seguinte *link*:
https://github.com/MAPL-UFU/Didatic_Mobile_Solution_for_Remote_Water_Level_Control

Também foi produzido um vídeo durante a fase de testes, o mesmo pode ser acessado através deste segundo *link*: https://youtu.be/Eat6SGTJjog

5 CONCLUSÃO

O desenvolvimento deste projeto demonstrou com sucesso a viabilidade e a eficácia da aplicação de tecnologias da Industria 4.0, como a Internet das Coisas (IoT) e sistemas ciberfísico para a modernização de equipamentos didáticos em ambientes de ensino de engenharia. Os objetivos geral e específicos delineados no início deste trabalho foram alcançados, comprovando uma solução completa e robusta para o controle de uma bancada de nível por espaço de estados.

A arquitetura híbrida combina a comunicação *Bluetooth Low Energy* (BLE) para a configuração inicial e operação local e o protocolo MQTT para a operação remota provou ser uma excelente escolha. O uso do BLE simplificou o processo de *setup* da bancada, tornando-o intuitivo e seguro para o docente, enquanto o MQTT garantiu uma comunicação de baixa latência e alta confiabilidade para a transmissão de parâmetros de controle e dados de telemetria durante os experimentos. A implementação do *FreeRTOS* no *framework* no ESP32 para a separação de tarefas de controle e comunicação em núcleos distintos foi crucial para garantir a estabilidade do sistema, permitindo que a malha de controle operasse em tempo real sem ser afetada por flutuações na rede.

O aplicativo móvel desenvolvido em *Flutter* atendeu a todos os requisitos propostos, oferecendo uma interface de usuário rica e funcional, com fluxos distintos e adaptados para os perfis de professor e aluno. A capacidade de visualizar a resposta do sistema em tempo real através de gráficos e de exportar os dados coletados para análise posterior agrega um valor pedagógico significativo à ferramenta, aproximando a teoria de controle da prática experimental de forma interativa e moderna.

Importante ressaltar que sendo o foco do projeto puramente didático não foram implementados camadas avançadas de segurança na comunicação MQTT, como a criptografia de canal via TSL/SSL. Para a aplicação da solução em ambientes que exijam maior proteção de dados, recomenda-se a incorporação de protocolos de segurança adequados.

Conclui-se portanto que a integração de plataformas de baixo custo como o ESP32 e *frameworks* de desenvolvimento rápido como o *Flutter* viabiliza a criação de ferramentas de ensino poderosas e alinhadas aos paradigmas da Industria 4.0. Aprimorando a experiência de aprendizado e preparando os futuros engenheiros para os desafios tecnológicos do mercado.

6 TRABALHOS FUTUROS

6.1 INTEGRAÇÃO COM PLATAFORMAS DE *CLOUD*

Os dados de telemetria publicados no *broker* MQTT poderia ser integrados a uma plataforma de nuvem, como a AWS ou *Google Cloud* IoT e armazenados em um banco de dados. Possibilitando assim a criação de *dashboard* web para que os professores analisem o histórico de dados de todos os experimentos realizados, comparem resultados entre diferentes turmas e gerenciem as bancadas remotamente.

6.2 APRIMORAMENTOS DE SEGURANÇA E GERÊNCIA DE DADOS

Pensando em produzir uma aplicação mais robusta seria interessante implementar um sistema de autenticação completo no aplicativo, integrado a um banco de dados. Adicionalmente o *broker* poderia ser configurado com listas de controle de acesso (ACLs) e autenticação por cliente, além da implementação de comunicação criptografada garantindo que apenas usuários autorizados utilizem a bancada.

6.3 IMPLEMENTAÇÃO DE *STREAMING* DE VIDEO EM TEMPO REAL

Para criar uma experiência de laboratório remoto ainda mais imersiva, uma câmera poderia ser adicionada à bancada para transmitir ao vivo o experimento. O *stream* de vídeo poderia ser integrado diretamente ao aplicativo do aluno e à tela de monitoramento do professor, permitindo que eles tenham um *feedback* imediato do comportamento físico do sistema, complementando os dados numéricos e os gráficos. Esta funcionalidade poderia ser implementada com o uso de um microcontrolador como o Raspberry Pi e um módulo específico como o ESP32-CAM.

6.4 CRIAÇÃO DE UM GÊMEO DIGITAL

Utilizando os dados de telemetria em tempo real publicados via MQTT seria possível desenvolver um Gêmeo Digital da bancada construída em um ambiente de simulação como *Unity*, *Unreal Engine* ou mesmo um *dashboard web* avançado espelharia o estado da bancada física e os alunos poderiam primeiro testar seus parâmetros de controle no Gêmeo Digital de forma segura, prevendo comportamentos e só então aplicá-los ao sistema real.

6.5 GERENCIAMENTO DE FILAS COM IA

Para laboratórios com múltiplas bancadas e alta demanda, um sistema de gerenciamento de fila poderia ser implementado com o uso de algoritmos de Inteligência Artificial o sistema poderia otimizar o agendamento de experimentos, prever tempos de espera e até mesmo analisar os parâmetros enviados pelos alunos para sugerir sequencias de testes mais eficientes.

6.6 MELHORIA NO CONTINGENCIAMENTO

Para viabilizar uma estrutura mais robusta, pode ser adicionado ao ESP uma verificação do tipo de rede, para que caso a bancada e o telefone do aluno compartilhem da mesma rede, ou seja o experimento esteja acontecendo de maneira local através da internet, seja desenvolvido um procedimento em que caso de desconexão da bancada com a rede o experimento possa continuar a ser executado através do BLE sem a perda do status atual e de nenhuma informação já enviada/recebida através do MQTT.

REFERÊNCIAS

AUGUSTO, P. Laboratório 7 - Espaço de estados. Uberlândia, 2024.

Dart Overview, 2025. Disponível em < https://dart.dev/overview>

Documentação Flutter, 2025. Disponível em <https://docs.flutter.dev/>

Eclipse Mosquitto™ Um corretor MQTT de código aberto, 2025. Disponível em <https://mosquitto.org>

ESP32-DevKitC, 2025. Disponível em <https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32/esp32-devkitc/index.html>

ESP32 A feature-rich MCU with integrated Wi-Fi and Bluetooth connectivity for a wide-range of applications, 2025. Disponível em <https://www.espressif.com/en/products/socs/esp32>

MASCHIETTO, L. G.; VIEIRA, A. L. N.; TORRES, F. E.; et al. Arquitetura e Infraestrutura de IoT. Porto Alegre: SAGAH, 2021.

MORAIS, I. S. de; GONÇALVES, P. de F.; LEDUR, C. L.; et al. Introdução a Big Data e Internet das Coisas (IoT). Porto Alegre: SAGAH, 2018.

MQTT Explorer An all-round MQTT client that provides a structured topic overview, 2025. Disponível em <https://mqtt-explorer.com/>

MQTT: The Standard for IoT Messaging, 2025. Disponível em <https://mqtt.org/>

O que é QoS (Quality of Service) em redes?, 2025. Disponível em <https://www.fortinet.com/br/resources/cyberglossary/qos-quality-of-service>

OGATA, K. Engenharia de Controle Moderno. 5. ed. São Paulo: Pearson Prentice Hall, 2010.

QUINTINO, L. F.; SILVEIRA, A. M. da; AGUIAR, F. R. de; et al. Indústria 4.0. Porto Alegre: SAGAH, 2019.

SACOMANO, J. B.; GONÇALVES, R. F.; BONILLA, S. H. Indústria 4.0 : conceitos e fundamentos. São Paulo: Editora Blucher, 2018.

TAVARES, J. J.-P. Z. D. S. Arquitetura, Comunicação e Protocolos IoT e IIoT. Uberlândia, 2025.