Filipe Barros Vitorino

Monitoramento Inteligente de Equipamentos do Programa Água Doce com IoT:

Automação da Coleta de Dados com ESP32 e Aplicativo Flutter

Belo Horizonte, Minas Gerais
2025

Filipe Barros Vitorino

Monitoramento Inteligente de Equipamentos do Programa Água Doce com IoT:

Automação da Coleta de Dados com ESP32 e Aplicativo Flutter

Proposta de Pesquisa Tecnológica

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Thiago Ferreira de Noronha
Coorientadora: Anolan Milanés

Belo Horizonte, Minas Gerais
2025

Sumário

1 Introdução

O Programa Água Doce (PAD) constitui uma política pública do Governo Federal voltada para ampliar o acesso à água potável nas comunidades rurais do semiárido brasileiro, região historicamente marcada pela escassez hídrica, baixa pluviosidade e dependência de águas subterrâneas frequentemente salobras [11]. Coordenado pelo Ministério da Integração e do Desenvolvimento Regional, o programa utiliza sistemas de dessalinização baseados em osmose inversa para tornar a água subterrânea adequada ao consumo humano, representando uma ação estratégica de segurança hídrica e desenvolvimento social [5]. Sua atuação abrange estados como Bahia, Ceará, Pernambuco, Paraíba, Rio Grande do Norte, Piauí, Sergipe, Alagoas, Maranhão e Minas Gerais, beneficiando milhares de pessoas ao longo dos anos [14]. Desde sua criação, o PAD já recebeu investimentos superiores a R$ 250 milhões, destinados à implantação de sistemas, capacitação comunitária e manutenção da infraestrutura [5]. Em 2025, o programa superou a marca de mil sistemas entregues, consolidando-se como uma das maiores iniciativas de acesso à água potável no semiárido brasileiro [11].

Ao longo de sua implementação, o PAD estruturou um modelo de atuação que combina diagnóstico socioambiental, implantação de infraestrutura tecnológica e gestão comunitária descentralizada. As comunidades atendidas passam por avaliações técnicas que incluem análise da qualidade da água, vazão dos poços, vulnerabilidade social e disponibilidade de energia elétrica, garantindo que os recursos sejam destinados a locais com características compatíveis para a dessalinização [20]. Após a instalação dos sistemas, o programa promove a capacitação de operadores locais, fortalecendo a autonomia comunitária e assegurando o funcionamento contínuo dos equipamentos [5]. A distribuição da água potável é realizada por meio de pontos estruturados dentro das comunidades, permitindo que o acesso ao recurso seja organizado e equitativo, aspecto essencial em regiões marcadas pela escassez e irregularidade climática [11].

Além dos benefícios imediatos associados à melhoria da qualidade da água e à redução de doenças decorrentes do consumo inadequado, o PAD integra diretrizes de sustentabilidade ambiental por meio do manejo controlado do concentrado salino, resíduo gerado no processo de dessalinização [14]. Em diversas localidades, esse rejeito passa a ser utilizado em sistemas produtivos integrados, como piscicultura e irrigação de plantas halófitas, ampliando os impactos socioeconômicos positivos do programa e contribuindo para a geração de renda e fortalecimento da agricultura familiar [5]. Dessa forma, o PAD reforça seu papel não apenas como solução de acesso à água potável, mas como instrumento de desenvolvimento regional no semiárido.

Nesse contexto, a adoção de tecnologias acessíveis, como microcontroladores ESP32 e aplicativos móveis desenvolvidos em Flutter, surge como alternativa viável para aprimorar

o monitoramento dos sistemas, permitindo a automação da coleta de dados, o acompanhamento em tempo real e a identificação precoce de falhas ou de necessidades de manutenção [24, 10]. A integração dessas tecnologias de baixo custo e alta eficiência acrescenta inovação à política pública ao preencher lacunas técnicas relacionadas à gestão operacional dos dessalinizadores [26]. Assim, esta pesquisa assume relevância acadêmica e social ao propor uma solução tecnológica inclusiva para fortalecer a eficiência e a sustentabilidade do PAD, contribuindo para o desenvolvimento regional, para a melhoria da governança hídrica e para a ampliação da segurança no abastecimento das comunidades do semiárido brasileiro [4].

1.1 Objetivos Gerais

O presente trabalho tem como objetivo desenvolver uma solução tecnológica baseada em um aplicativo multiplataforma, construído com o framework Flutter, que permita a coleta, o monitoramento e a gestão dos dados operacionais provenientes dos sistemas de dessalinização implementados pelo Programa Água Doce. A proposta visa estabelecer uma comunicação eficiente e segura entre o aplicativo e os microcontroladores ESP32, responsáveis pela captação dos parâmetros técnicos dos equipamentos, como pressão, vazão, condutividade, entre outros indicadores essenciais para o funcionamento adequado dos sistemas.

Ao viabilizar a obtenção dos dados em tempo real, a solução proposta busca otimizar o acompanhamento e a manutenção dos sistemas de dessalinização, proporcionando maior agilidade na identificação de falhas, na tomada de decisões e na execução de ações corretivas ou preventivas. Dessa forma, pretende-se não apenas aprimorar a eficiência operacional, mas também fortalecer a sustentabilidade dos sistemas e garantir a continuidade do fornecimento de água de qualidade às comunidades rurais atendidas, contribuindo diretamente para a promoção da saúde, do bem-estar social e do desenvolvimento sustentável das populações em situação de vulnerabilidade hídrica.

1.2 Objetivos Específicos

- Desenvolver um aplicativo multiplataforma, utilizando Flutter, com interface intuitiva, para ser utilizado por operadores, técnicos e gestores do Programa Água Doce.

- Implementar a comunicação entre o aplicativo e o microcontrolador ESP32, por meio de protocolos como Bluetooth ou Wi-Fi, possibilitando a leitura de dados operacionais dos sistemas de dessalinização, como pressão, vazão, condutividade, tensão, entre outros.

- Projetar e desenvolver a arquitetura de backend, integrando banco de dados em nuvem para armazenamento e gestão dos dados coletados.
- Garantir que o aplicativo permita a visualização dos dados coletados, histórico de leituras e geração de relatórios para auxiliar na tomada de decisões técnicas e operacionais.
- Realizar testes funcionais e de comunicação para garantir o correto funcionamento da integração entre hardware (ESP32) e software (aplicativo Flutter).
- Proporcionar uma solução tecnológica que facilite a manutenção preditiva e preventiva dos sistemas de dessalinização, contribuindo para a continuidade e eficiência do fornecimento de água potável nas comunidades atendidas.

2 Impactos Sociais

Ao possibilitar o monitoramento eficiente dos equipamentos, a solução contribui para a maior disponibilidade de água de qualidade, reduzindo os riscos de falhas nos sistemas e garantindo o funcionamento contínuo dos dessalinizadores. Isso reflete diretamente na melhoria das condições de saúde da população, uma vez que o acesso à água potável é um direito básico e essencial para a prevenção de doenças de veiculação hídrica e promoção do bem-estar [27].

Além disso, o projeto fortalece a autonomia das comunidades, pois capacita operadores locais a utilizarem ferramentas tecnológicas para acompanhar e resolver problemas de forma mais ágil. A apropriação de ferramentas digitais por comunidades é um fator chave para promover a inclusão digital, gerar conhecimento local e ampliar as oportunidades de desenvolvimento socioeconômico [3].

O uso de tecnologias acessíveis, como o microcontrolador ESP32 e um aplicativo desenvolvido em Flutter, também favorece a sustentabilidade econômica do Programa. A principal vantagem econômica reside na substituição das inspeções manuais periódicas por um sistema de monitoramento contínuo e remoto. Estudos na área de monitoramento de recursos hídricos demonstram que essa abordagem automatizada reduz significativamente os custos associados ao deslocamento de equipes técnicas e à mão de obra especializada, otimizando os recursos públicos investidos [12].

Portanto, este trabalho não apenas propõe uma solução tecnológica, mas também contribui significativamente para a transformação social, promovendo saúde, dignidade, desenvolvimento e sustentabilidade para comunidades que enfrentam desafios históricos relacionados ao acesso à água potável.

3 Referencial Teórico

O desenvolvimento de soluções móveis baseadas no framework Flutter tem se consolidado como uma abordagem eficiente para a criação de aplicações multiplataforma, garantindo desempenho nativo a partir de um único código-fonte. Essa característica não apenas eleva a produtividade no ciclo de desenvolvimento, como também reduz custos operacionais e de manutenção [18]. Além disso, a adoção de princípios como a Clean Architecture tem sido amplamente recomendada na literatura técnica, assegurando maior escalabilidade, modularidade e separação clara de responsabilidades entre camadas lógicas [2].

No domínio da Internet das Coisas (IoT), o microcontrolador ESP32 surge como uma plataforma acessível e versátil, dotada de múltiplas interfaces de comunicação, o que a torna particularmente adequada para aplicações em ambientes remotos. Pesquisas como as conduzidas pela [16] demonstram a eficácia do protocolo WebSocket na transmissão de dados em tempo real com o ESP32, enquanto estudos como os publicados pela [17] exploram o protocolo ESP-NOW, viabilizando comunicação direta entre dispositivos mesmo na ausência de infraestrutura de rede convencional.

A integração entre Flutter e dispositivos embarcados já foi investigada em diversos contextos, reforçando sua viabilidade técnica. Trabalhos como os desenvolvidos pela [7, 15] destacam a utilização do Flutter para coleta e controle de dados via Bluetooth Low Energy (BLE), enquanto pesquisas como as da [25] comprovam sua aplicação no gerenciamento de sistemas IoT. Adicionalmente, estudos como [23, 9] propõem arquiteturas de monitoramento baseadas no ESP32 para aplicações ambientais e de saneamento, alinhando-se às demandas do Programa Água Doce (PAD), que requer soluções portáteis, econômicas e adaptáveis a regiões com limitações de conectividade.

Essa combinação de tecnologias móveis (Flutter) e dispositivos embarcados (ESP32) configura uma abordagem técnica promissora para o monitoramento automatizado de sistemas de dessalinização, possibilitando:

- Coleta e transmissão eficiente de dados em tempo real;
- Alertas automatizados sobre falhas ou necessidade de manutenção;

Dessa forma, a revisão da literatura sustenta a viabilidade técnica da proposta, evidenciando que a combinação dessas tecnologias pode contribuir significativamente para a otimização do PAD, assegurando maior eficiência operacional.

4 Tecnologias Utilizadas

O desenvolvimento deste projeto faz uso de tecnologias distribuídas em diferentes camadas da solução, permitindo integração eficiente entre o aplicativo móvel e o sistema embarcado responsável pela coleta dos dados dos sistemas de dessalinização. Para apresentar essas tecnologias de maneira organizada, elas são descritas a seguir conforme seu papel dentro da arquitetura.

4.1 Camada de Aplicação Móvel

O aplicativo é executado em dispositivos Android, plataforma amplamente utilizada pelos técnicos do Programa Água Doce. A escolha por Android se deve à sua acessibilidade, flexibilidade e ao suporte nativo a interfaces essenciais para comunicação sem fio, como Bluetooth Low Energy (BLE) e redes Wi-Fi.

4.2 Camada de Desenvolvimento do Aplicativo

A construção da interface móvel utiliza o framework Flutter, criado pelo Google, que permite o desenvolvimento de aplicações modernas e responsivas a partir de uma única base de código [10]. A linguagem adotada, Dart, oferece boa performance e é adequada para aplicações que exigem atualização rápida de informações.

O Flutter dispõe de extensa biblioteca de widgets, documentação abrangente e integração facilitada com recursos nativos dos dispositivos Android. Essas características tornam o framework adequado para o desenvolvimento de uma interface amigável e consistente para os operadores do Programa Água Doce.

4.3 Camada de Hardware

O sistema embarcado utiliza o microcontrolador ESP32, desenvolvido pela Espressif Systems e amplamente empregado em aplicações de Internet das Coisas (IoT) devido à sua versatilidade, conectividade e baixo consumo energético [6]. O dispositivo possui processador dual-core, suporte integrado a Wi-Fi e Bluetooth, além de interfaces que permitem a integração com sensores.

No contexto deste projeto, o ESP32 realiza a coleta e o processamento dos dados provenientes dos sensores dos sistemas de dessalinização, incluindo medições de pressão, vazão, temperatura e volume.

4.4 Camada de Desenvolvimento Embarcado

O firmware do ESP32 foi desenvolvido utilizando ferramentas amplamente reconhecidas no ecossistema embarcado. Entre elas destacam-se:

- o **Arduino Framework**, que oferece simplicidade e agilidade durante o desenvolvimento;
- o **ESP-IDF**, framework oficial da Espressif, que fornece maior controle e otimização do hardware;
- o **PlatformIO**, utilizado como ambiente de desenvolvimento e gerenciamento de dependências.

Essa combinação permite flexibilidade na implementação e organização eficiente do firmware.

4.5 Diferença entre Bluetooth Clássico e Bluetooth Low Energy (BLE)

O projeto faz uso doBluetooth Low Energy (BLE), uma evolução do Bluetooth clássico, projetada especificamente para dispositivos que exigem baixo consumo de energia, comunicação esporádica e transmissão de pequenos volumes de dados.

As principais diferenças entre Bluetooth Clássico e BLE são apresentadas na Tabela 1.

Tabela 1 – Comparativo entre Bluetooth Clássico e Bluetooth Low Energy (BLE) [1]

Característica	**Bluetooth Clássico**	**Bluetooth Low Energy (BLE)**
Consumo de energia	Alto	Baixo
Taxa de transmissão	Até 2-3 Mbps	Até 1 Mbps
Tempo de conexão	Conexões permanentes	Conexões rápidas e temporárias
Foco de uso	Áudio, chamadas, dados contínuos	IoT, sensores, dispositivos médicos
Topologia	Ponto a ponto	Ponto a ponto ou broadcast
Alcance	10-100 metros	10-100 metros

O BLE foi a escolha para este projeto devido à sua eficiência energética e à sua adequação para aplicações que envolvem monitoramento periódico e transmissão de pequenas quantidades de dados, como é o caso dos sistemas de dessalinização. Além disso, a maioria dos smartphones modernos é compatível com BLE, facilitando a comunicação entre o aplicativo desenvolvido em Flutter e o ESP32.

4.6 Integração das Tecnologias

A combinação do Flutter com o ESP32 proporciona uma solução robusta, moderna e de fácil manutenção. O aplicativo desenvolvido oferece uma interface amigável para operadores, enquanto o ESP32 realiza a coleta dos dados físicos dos sistemas. A comunicação entre ambos pode ocorrer tanto por Wi-Fi, quando há disponibilidade de rede, quanto por Bluetooth BLE, em situações onde não há infraestrutura de internet, garantindo versatilidade no uso.

5 Metodologia

A metodologia adotada neste trabalho foi estruturada em quatro etapas principais, que nortearam o desenvolvimento da solução proposta. Todas as etapas foram concluídas com êxito, resultando em um sistema funcional composto por hardware (ESP32) e software (aplicativo Flutter) integrados para coleta, sincronização e visualização de dados dos equipamentos do Programa Água Doce.

5.1 Levantamento de Requisitos

A definição dos requisitos foi realizada a partir da análise das rotinas operacionais das unidades do Programa Água Doce e das necessidades técnicas do monitoramento dos sistemas de dessalinização. Os requisitos a seguir orientaram o projeto, servindo como base para o desenvolvimento da arquitetura do ESP32, do aplicativo e dos mecanismos de comunicação.

Requisitos Funcionais (RF)

RF1. Coletar dados dos sensores (pressão, vazão, condutividade, tensão, volume) em tempo real.

RF2. Sincronizar arquivos de log armazenados no ESP32 para o aplicativo.

RF3. Exibir os valores dos sensores no aplicativo de forma atualizada.

RF4. Permitir a comunicação via Bluetooth Low Energy (BLE).

RF5. Permitir a comunicação via Wi-Fi em modo Access Point.

RF6. Enviar os dados coletados para o servidor remoto (nuvem) [8].

RF7. Armazenar localmente as leituras em banco de dados SQLite quando não houver internet.

RF8. Permitir ao operador visualizar histórico de coletas anteriores.

Requisitos Não Funcionais (RNF)

RNF1. O aplicativo deve funcionar mesmo sem internet (modo offline).

RNF2. A comunicação deve garantir integridade e ordem das mensagens transmitidas.

RNF3. A interface deve ser intuitiva e adequada ao perfil dos operadores do PAD.

RNF4. A transmissão dos dados deve ocorrer em tempo aceitável mesmo para grandes volumes.

RNF5. O sistema deve suportar reconexão automática quando houver perda temporária de comunicação.

5.2 Desenvolvimento do Protótipo com ESP32

O protótipo foi construído utilizando o microcontrolador ESP32, que integra as funções de leitura dos sensores e de comunicação sem fio. O dispositivo foi configurado para operar em dois modos distintos de transmissão: via Bluetooth Low Energy (BLE) e via Wi-Fi, permitindo avaliar o desempenho e a adequação de cada tecnologia às necessidades do projeto.

No modo BLE, o ESP32 atua como um GATT (Generic Attribute Profile), responsável por organizar serviços e características utilizadas na troca de dados entre cliente e servidor — conceito detalhado posteriormente no Capítulo 6.

Já no modo Wi-Fi, o ESP32 funciona como um Access Point e hospeda um servidor HTTP, que disponibiliza múltiplos endpoints. Cada endpoint retorna dados em formato JSON, de modo semelhante a uma API REST. Durante a transmissão, cada arquivo de log é tratado como uma página, e os dados são enviados em chunks JSON (partes sucessivas), suficientemente grandes para caber no buffer interno, permitindo a transmissão contínua e eficiente de grandes volumes de dados.

5.3 Desenvolvimento do Aplicativo com Flutter

O aplicativo foi desenvolvido de forma multiplataforma utilizando o framework Flutter, adotando a arquitetura MVVM (Model-View-ViewModel) para garantir melhor organização, testabilidade e manutenção do código.

A aplicação tem como principais funcionalidades monitorar os sensores em tempo real, exibir históricos de medições e gerar relatórios baseados nos dados coletados. Além disso, ela permite enviar os registros coletados para o servidor do projeto hospedado na nuvem [8], responsável por centralizar e armazenar as informações para análise posterior.

A comunicação entre o aplicativo e o microcontrolador ESP32 pode ocorrer de duas formas: via Bluetooth Low Energy (BLE) ou via Wi-Fi, dependendo da disponibilidade e das condições do ambiente.

O aplicativo também conta com um banco de dados local SQLite, que armazena temporariamente os dados quando não há conexão ativa, garantindo que nenhuma informação seja perdida. Assim que a conectividade é restabelecida, o aplicativo realiza a sincronização automática com o servidor remoto.

5.4 Testes e Validação

Na etapa final, foram realizados testes funcionais e de desempenho para validar o sistema. Os testes abrangeram tanto o correto funcionamento das leituras em tempo real e da sincronização de registros, quanto a comparação de desempenho entre os modos BLE e Wi-Fi.

5.4.1 Comparação de Desempenho entre BLE e Wi-Fi

Para comparar o desempenho das duas formas de comunicação, foram conduzidos experimentos controlados medindo o tempo total de transmissão de arquivos de log com diferentes quantidades de linhas.

Foram realizados testes variando o número de linhas transmitidas — de 144 até 1008 linhas, em incrementos regulares — com cada experimento repetido cinco vezes sob as mesmas condições de operação.

Os tempos médios de envio e recebimento foram registrados para ambos os modos de comunicação. Observou-se que o BLE apresentou maior latência à medida que o número de linhas aumentava, devido à necessidade de confirmação (ACK) a cada linha transmitida. Já o Wi-Fi manteve tempos significativamente menores, uma vez que envia os dados em blocos contínuos (chunks JSON) sem interrupção entre as partes.

Essa diferença de comportamento reflete as características intrínsecas de cada tecnologia: o BLE é ideal para transmissões leves e contínuas com baixo consumo de energia, enquanto o Wi-Fi se destaca em tarefas que exigem maior largura de banda e transferência de grandes volumes de dados.

Os resultados dessa comparação serão apresentados em forma de gráfico, evidenciando a diferença de tempo entre as transmissões BLE e Wi-Fi em função do número de linhas enviadas.

6 Análise dos Protocolos de Comunicação

6.1 Arquitetura e Comunicação BLE

A arquitetura de comunicação Bluetooth Low Energy (BLE) implementada neste projeto foi desenvolvida para permitir a interação direta entre o nó sensor, representado pelo microcontrolador ESP32, e o aplicativo móvel. O objetivo dessa camada de comunicação é fornecer uma alternativa de sincronização e monitoramento de dados independente de uma rede Wi-Fi, possibilitando que o usuário acesse leituras, configure parâmetros e realize o envio de registros históricos de forma local e de baixa potência.

O BLE é baseado em uma arquitetura denominada GATT (Generic Attribute Profile), que define como os dados são organizados e trocados entre dispositivos. Dentro dessa estrutura, um dos dispositivos assume o papel de GATT Server — no caso deste projeto, o ESP32 — enquanto o outro atua como GATT Client, papel desempenhado pelo aplicativo desenvolvido em Flutter. O servidor BLE é responsável por disponibilizar serviços e características que contêm os dados e comandos que o cliente pode ler, escrever ou assinar para receber notificações.

Cada unidade funcional dentro do GATT é identificada de forma única por meio de um UUID (Universally Unique Identifier). O UUID é uma sequência de 128 bits que garante a distinção inequívoca entre diferentes serviços e características BLE, mesmo em sistemas complexos com múltiplos dispositivos e sensores. No código implementado, definiram-se dois níveis principais de identificação: UUIDs de serviço e UUIDs de característica.

O UUID de serviço representa um agrupamento lógico de funcionalidades relacionadas — por exemplo, o serviço responsável pela comunicação com o hub central (denominado HUB Service) e o serviço que agrupa as características de cada sensor (Sensor Service). Cada serviço funciona como um "container" que organiza as características associadas a ele. Assim, o `HUB_SERVICE_UUID` identifica o serviço principal de controle e transmissão, enquanto o UUID dinâmico gerado para o Sensor Service é obtido a partir da configuração do hub, permitindo a criação flexível de novas características conforme os sensores conectados.

Dentro de cada serviço, encontram-se as características BLE, representadas também por UUIDs exclusivos. Cada característica contém um valor ou um conjunto de valores que podem ser lidos, escritos ou enviados como notificações para o cliente. No caso deste sistema, há duas características centrais associadas ao Hub Service: uma de recepção (RX), utilizada para o recebimento de comandos enviados pelo aplicativo, e outra de transmissão (TX), responsável pelo envio de dados, notificações e respostas do ESP32 ao cliente. Já no Sensor Service, cada sensor possui sua própria característica com UUID próprio, criada dinamicamente durante a inicialização do servidor, o que permite a transmissão

independente e paralela das leituras de cada dispositivo.

A comunicação entre servidor e cliente segue o modelo de eventos definido pelo GATT. Quando o aplicativo se conecta ao ESP32, o servidor BLE, através da classe `MyServerCallbacks`, reconhece o evento de conexão e inicia o processo de advertising, tornando o dispositivo disponível para emparelhamento e reconexão automática. Uma vez estabelecida a conexão, o cliente pode escrever na característica RX comandos específicos de controle, como iniciar a sincronização de registros (`0x02`), iniciar ou parar o envio de dados em tempo real (`0x03` e `0x05`), solicitar configurações (`0x20`), ou mesmo deletar arquivos de log (`0x06`).

Cada comando é tratado pela classe `MyCallbacks`, que interpreta o valor recebido e atualiza as variáveis de estado correspondentes. Essas variáveis, por sua vez, são monitoradas no laço principal da função `loopBLE()`, que executa rotinas específicas conforme o estado do sistema — por exemplo, `handleSyncProcess()` é chamado quando uma sincronização total é requisitada, e `notifySensorValue()` é utilizado para o envio periódico dos valores de sensores enquanto o fluxo em tempo real está ativo.

Para o envio dos dados, o servidor BLE utiliza a característica TX e o mecanismo de notificação. Quando o ESP32 atualiza o valor dessa característica e chama o método `notify()`, o cliente recebe instantaneamente o novo dado, sem necessidade de polling, o que garante eficiência energética e baixa latência. Como o BLE impõe um limite de tamanho por pacote, optou-se por transmitir os arquivos de log linha por linha, em vez de blocos maiores. A cada linha enviada, o ESP32 aguarda um sinal de confirmação (ACK) do aplicativo antes de prosseguir, assegurando que nenhuma informação seja perdida e que o processo de transmissão mantenha a integridade dos dados.

Um aspecto importante da arquitetura é o mecanismo de confirmação (ACK), implementado para assegurar a confiabilidade da comunicação. Após o envio de cada bloco de dados, o servidor aguarda um byte de confirmação enviado pelo cliente. Caso o ACK não seja recebido dentro de um intervalo de dois segundos, o envio é interrompido e o sistema registra o evento como falha de comunicação. Esse controle é essencial, pois o BLE, embora eficiente, não garante entrega de pacotes em nível de aplicação.

A função `handleSyncProcess()` coordena todo o fluxo de sincronização de múltiplos arquivos armazenados no sistema de arquivos interno (LittleFS). O processo inicia com o envio de uma mensagem de início (SOT), seguida pela transmissão de cada registro de log em formato JSON e finalizada com a mensagem de encerramento (EOT). Cada etapa é validada com a recepção de ACKs, assegurando a integridade e a ordem dos dados transferidos.

Em termos de estrutura lógica, a comunicação BLE implementada neste projeto estabelece um canal seguro, assíncrono e eficiente para transferência de dados e controle de dispositivos. A combinação entre o uso de UUIDs exclusivos, a criação dinâmica de serviços e características, o controle de estado por eventos e a fragmentação de mensagens

demonstra uma arquitetura BLE madura e escalável. Essa abordagem permite que o sistema seja facilmente expandido com novos sensores ou funcionalidades sem comprometer a estabilidade do canal de comunicação.

Por fim, vale destacar que o uso do modelo de GATT Server no ESP32 oferece total controle sobre os fluxos de notificação e resposta, além de permitir uma integração direta com o aplicativo cliente em Flutter, que atua como GATT Client. Dessa forma, o sistema BLE do projeto não apenas serve como meio de comunicação, mas também como um subsistema completo de sincronização, monitoramento e configuração, projetado para operar de forma confiável mesmo em ambientes sem conectividade de internet.

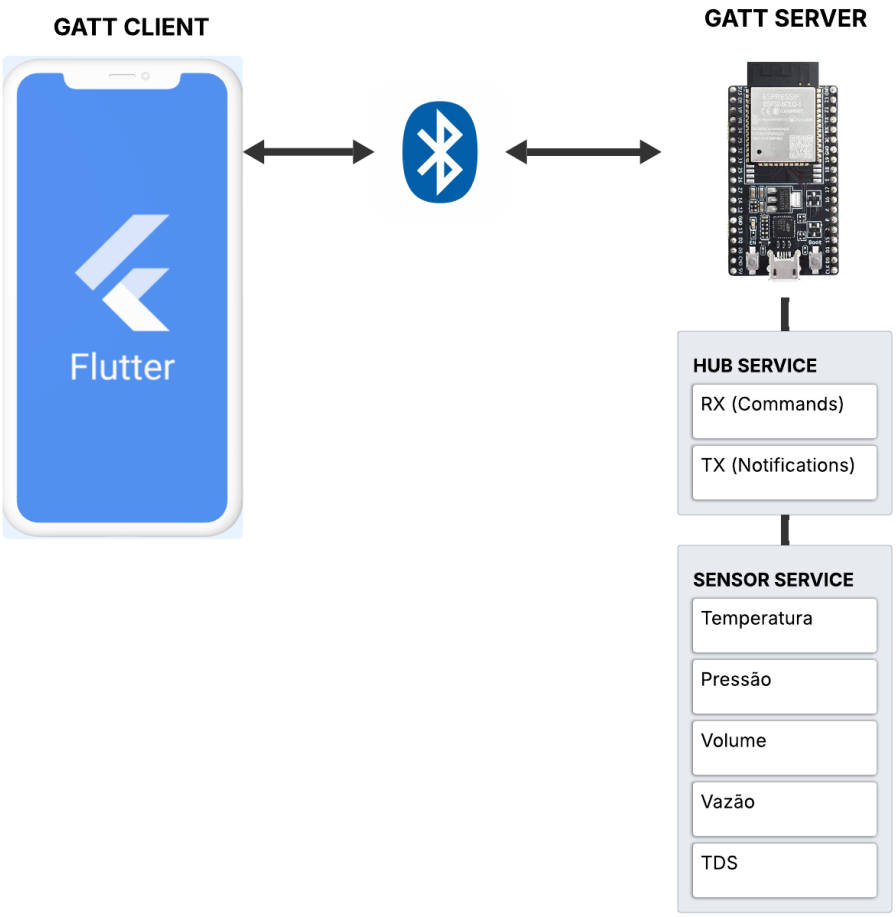

Figura 1 – Fluxo de comunicação BLE entre o aplicativo (GATT Client) e o ESP32 (GATT Server)

6.2 Arquitetura e Comunicação WI-FI

O módulo de Wi-Fi do ESP32 foi projetado para fornecer acesso à rede de forma confiável e eficiente, permitindo que o dispositivo funcione como um ponto de acesso (Access Point) para clientes conectarem-se diretamente. Ao inicializar o Wi-Fi, o ESP32 cria uma

rede própria com um SSID e senha definidos, fornecendo um endereço IP interno que possibilita a comunicação com o dispositivo através de qualquer navegador ou aplicativo capaz de realizar requisições HTTP.

Para gerenciar essa comunicação, foi implementado um servidor web que expõe múltiplos endpoints. Cada endpoint tem uma função específica: alguns fornecem informações de configuração do hub, outros permitem acessar os sensores em tempo real, limpar históricos de log ou consultar arquivos antigos. Esse modelo de interação se assemelha a uma REST API, onde cada recurso é identificado por uma URL e as operações são feitas por métodos HTTP como GET. Essa abordagem torna a integração com aplicativos móveis ou sistemas externos intuitiva e padronizada, permitindo que qualquer cliente realize requisições e receba respostas estruturadas em JSON.

Um dos desafios centrais é a transmissão dos arquivos de log, que podem ser grandes e, portanto, não caber integralmente na memória do ESP32. Para contornar essa limitação, cada arquivo de log é tratado como uma página, e a resposta é montada em chunks JSON. Nesse método, o servidor agrupa várias linhas do arquivo em blocos suficientemente grandes para caber no buffer e envia esses pedaços sequencialmente ao cliente. Diferentemente do BLE, onde cada linha é enviada individualmente e se espera um ACK, no Wi-Fi a comunicação é contínua, e o cliente processa os dados à medida que chegam. Esse formato permite consumir os arquivos de forma incremental, facilitando o carregamento de grandes históricos sem sobrecarregar a memória do dispositivo.

A páginação também é fundamental para a organização e navegação dos arquivos. Cada página representa um arquivo de log completo, e a resposta JSON inclui informações como número da página, total de arquivos, tamanho do arquivo e referências à página anterior e à próxima. Além disso, a transmissão em chunks garante que o conteúdo seja entregue de maneira robusta, mesmo em caso de interrupções temporárias na rede.

Para manter a compatibilidade com diferentes tipos de clientes, o servidor adiciona cabeçalhos CORS (Cross-Origin Resource Sharing). Isso é necessário porque navegadores e aplicações web seguem a política de same-origin, que impede, por padrão, que recursos hospedados em um domínio diferente acessem endpoints HTTP do ESP32. O CORS funciona como um mecanismo de segurança que informa explicitamente ao navegador que aquela origem externa tem permissão para consumir os dados oferecidos pela API local. Dessa forma, torna-se possível acessar o ESP32 tanto a partir do aplicativo Flutter quanto de ferramentas web ou clientes HTTP executados em domínios distintos.

O Wi-Fi handler também fornece endpoints para consultas imediatas aos sensores em tempo real. Quando solicitado, o servidor realiza uma leitura instantânea dos sensores e retorna os valores mais recentes em JSON. Isso garante que os aplicativos clientes possam obter dados atualizados de maneira rápida, sem esperar por agendamentos de coleta ou processos de sincronização.

Com essa arquitetura, o Wi-Fi handler do ESP32 consegue oferecer uma comunicação

eficiente e escalável, aproximando-se do comportamento de uma API REST completa, enquanto respeita as limitações de hardware do dispositivo, permitindo acesso incremental a grandes volumes de dados e garantindo que a experiência do usuário seja consistente e confiável.

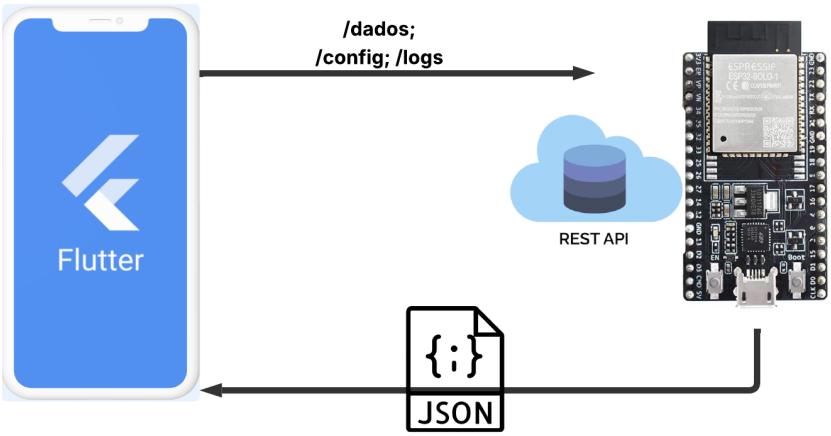

Figura 2 – Arquitetura de comunicação Wi-Fi entre o aplicativo e a API REST do ESP32.

6.3 Comparação entre BLE e Wi-Fi

Após a implementação das duas interfaces de comunicação sem fio no sistema, foi conduzida uma análise comparativa entre o Bluetooth Low Energy (BLE) e o Wi-Fi, com o objetivo de avaliar o desempenho prático de cada uma nas tarefas de monitoramento em tempo real e transmissão de arquivos de log. Ambas as tecnologias foram integradas ao ESP32 e desempenham as mesmas funções no sistema, diferenciando-se, contudo, pelo modo como realizam a comunicação e pelos resultados obtidos em termos de tempo de transmissão, confiabilidade e eficiência.

Nos experimentos realizados, o BLE apresentou comportamento consistente e confiável na transmissão de dados contínuos. Seu funcionamento, baseado em notificações do servidor GATT, permite que o microcontrolador envie atualizações imediatas aos dispositivos conectados, sem necessidade de consultas recorrentes por parte do aplicativo. Esse mecanismo é particularmente eficiente para o envio de medições em tempo real, mas também foi empregado no envio dos arquivos de log. Nesse caso, o envio ocorre linha a linha, e o aplicativo responde com uma confirmação (ACK) antes que a próxima linha seja transmitida. Esse método garante integridade total dos dados, embora resulte em maior tempo total de sincronização quando o volume de registros é elevado.

O Wi-Fi, por sua vez, também foi utilizado tanto para monitoramento em tempo real quanto para o envio de logs, mas adota uma estratégia de comunicação distinta. Nesse modo, o ESP32 atua como ponto de acesso e servidor web, disponibilizando endpoints HTTP que

seguem a lógica de uma API REST. O aplicativo requisita os dados ao microcontrolador, que envia as respostas no formato JSON, divididas em blocos (chunks) ajustados ao tamanho do buffer. Essa abordagem reduz a sobrecarga de controle e aproveita melhor a largura de banda, permitindo uma transmissão mais rápida e eficiente, especialmente quando há grande volume de informações a serem enviadas.

A avaliação de desempenho foi realizada a partir do envio de arquivos de log no formato JSONL, contendo diferentes quantidades de registros — variando de 144 até 1008 linhas. Cada teste foi repetido cinco vezes em condições idênticas, e a média dos tempos de transmissão foi utilizada como referência comparativa. Os resultados estão apresentados na Figura 3, que ilustra a variação do tempo de sincronização em função do número de registros transmitidos. Observa-se que, embora o tempo de transmissão cresça de forma aproximadamente linear para ambas as tecnologias, o BLE apresenta um crescimento mais acentuado. Para mil registros, o BLE atingiu cerca de 160 segundos, enquanto o Wi-Fi concluiu a mesma transmissão em aproximadamente 20 segundos.

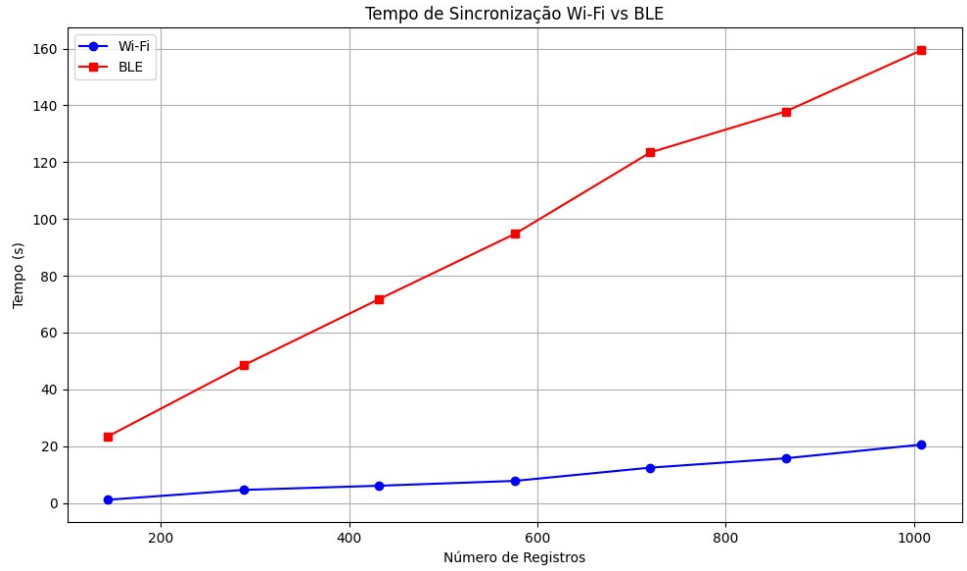

Figura 3 – Gráfico de comparação do tempo de transmissão BLE vs. Wi-Fi.

Essa diferença evidencia o impacto do protocolo de comunicação adotado. No BLE, o envio linha a linha com confirmação individual introduz um atraso cumulativo, o que torna o processo mais demorado à medida que o número de registros aumenta. Já no Wi-Fi, o envio em blocos contínuos e a maior largura de banda resultam em uma taxa de transferência significativamente superior, sem comprometer a integridade dos dados.

Em termos práticos, os resultados demonstram que ambas as interfaces são plenamente capazes de realizar as duas tarefas principais do sistema — o monitoramento em tempo real e a sincronização de registros —, mas cada uma apresenta vantagens específicas conforme o contexto de uso. O BLE mostra-se mais adequado em situações em que a estabilidade da conexão e o baixo consumo energético são prioritários, especialmente

durante o acompanhamento constante dos sensores. O Wi-Fi, por outro lado, destaca-se quando a demanda é pela transferência de grandes volumes de dados ou pela necessidade de reduzir o tempo total de sincronização.

Dessa forma, a comparação entre BLE e Wi-Fi não revela uma relação de substituição, mas de adaptação ao cenário operacional. O sistema pode explorar as vantagens de cada tecnologia conforme o contexto: utilizando BLE quando o foco é a comunicação contínua e energeticamente eficiente, e Wi-Fi quando o objetivo é realizar transmissões mais rápidas e de maior escala. Os resultados obtidos reforçam a flexibilidade da arquitetura proposta e demonstram que o uso combinado e dinâmico das duas interfaces representa uma solução robusta e eficiente para aplicações de monitoramento IoT.

7 Aplicativo

A coleta e o acompanhamento dos dados provenientes dos sensores instalados nas comunidades do Programa Água Doce exigem uma ferramenta prática e acessível, capaz de integrar-se diretamente ao hardware em campo. Para isso, foi desenvolvido um aplicativo móvel que atua como a principal interface de operação do sistema, permitindo ao operador visualizar medições em tempo real, sincronizar registros armazenados e gerir o funcionamento dos equipamentos de forma intuitiva e confiável.

Durante o desenvolvimento do sistema, três desafios centrais orientaram o projeto da comunicação: possibilitar a transmissão em tempo real das leituras, viabilizar a transferência completa do histórico de dados e, principalmente, conciliar as arquiteturas Wi-Fi e Bluetooth Low Energy (BLE) de modo que ambas pudessem executar as duas tarefas de forma independente. Assim, o operador tem liberdade para escolher o modo de comunicação mais adequado ao ambiente e à disponibilidade de rede — o BLE sendo ideal para conexões de curta distância e baixo consumo, e o Wi-Fi proporcionando maior velocidade e alcance para transmissões em lote.

O aplicativo foi desenvolvido em Flutter, adotando uma arquitetura moderna e modular, capaz de lidar com diferentes modos de operação e de adaptar-se dinamicamente às condições de conectividade. A lógica interna abstrai completamente a complexidade das camadas de rede, garantindo que o usuário tenha uma experiência fluida, independentemente do meio de comunicação utilizado.

A arquitetura funcional da aplicação é modularizada para gerenciar os distintos fluxos de trabalho do usuário, que englobam desde a autenticação e descoberta de dispositivos até a aquisição e sincronização de dados. O fluxo de interação inicia-se com a validação de acesso, progredindo para a seleção do modo de conectividade (BLE ou Wi-Fi AP) e, subsequentemente, para as operações de monitoramento em tempo real ou de transferência de dados históricos. As interfaces gráficas foram projetadas para expor esta lógica de forma intuitiva, e sua estrutura funcional é detalhada sequencialmente.

Tela de Autenticação

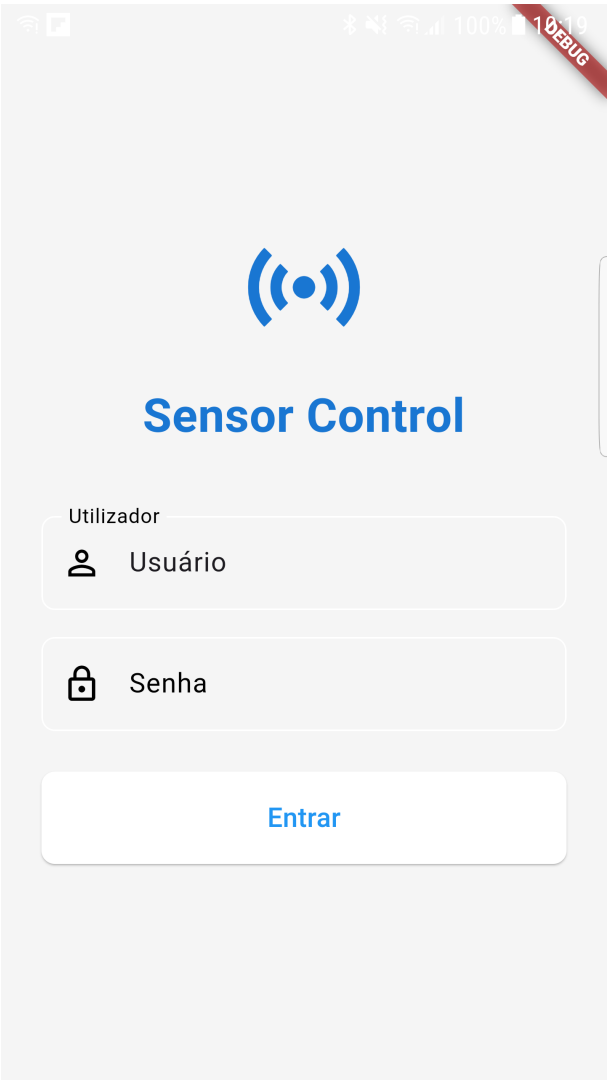

Figura 4 – Tela de Autenticação.

Esta interface é o ponto de entrada do sistema, responsável pela autenticação do usuário. Ela solicita a inserção de credenciais (Utilizador e Senha). A validação é iniciada pelo botão "Entrar", garantindo que apenas usuários autorizados acessem as funcionalidades subsequentes.

Menu Principal / Navegação

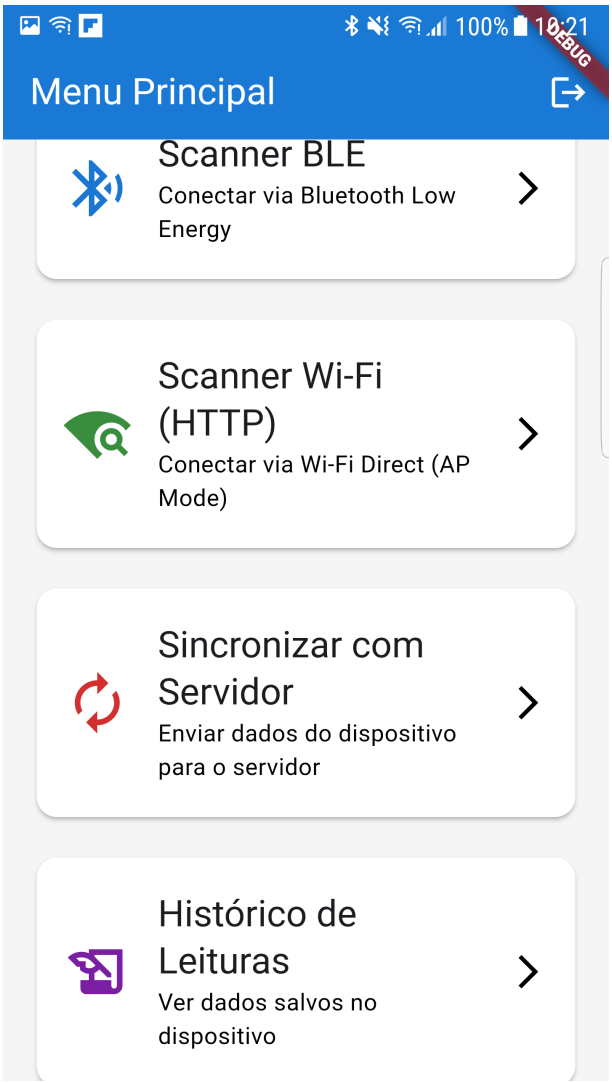

Figura 5 – Tela do Menu Principal.

Esta tela atua como o navegador central da aplicação. Ela direciona o usuário para os módulos primários do sistema:

- **Scanner BLE:** Inicia o fluxo de conexão com dispositivos via Bluetooth Low Energy.
- **Scanner Wi-Fi (HTTP):** Inicia o fluxo de conexão com dispositivos em modo *Access Point* (AP) via Wi-Fi.
- **Sincronizar com Servidor:** Gerencia o *upload* de dados armazenados localmente no dispositivo móvel para o servidor central (nuvem).
- **Histórico de Leituras:** Permite a visualização dos dados já coletados e salvos localmente.

Varredura de Dispositivos BLE

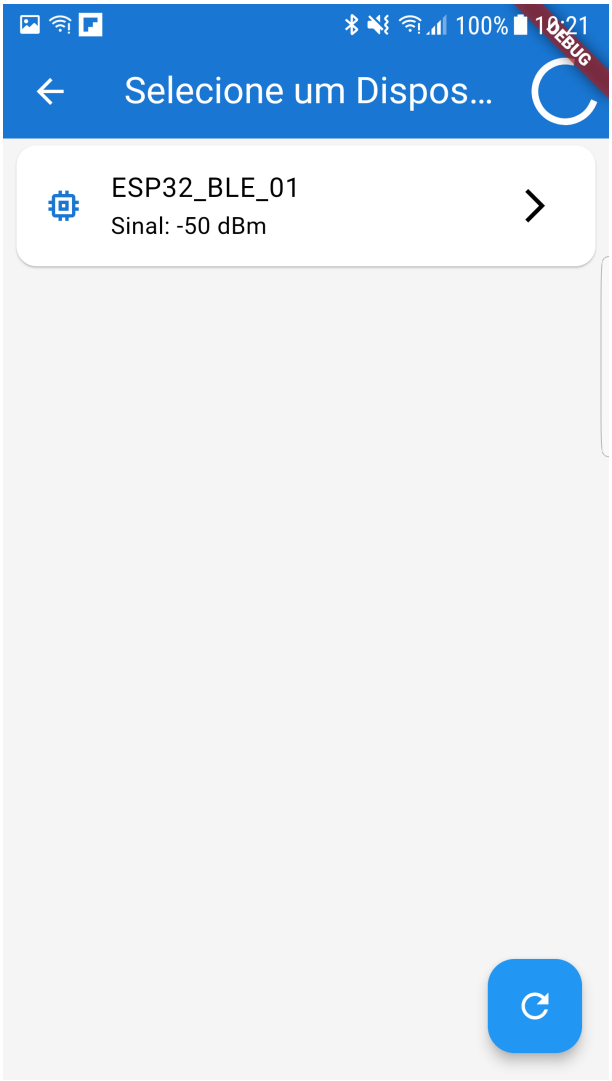

Figura 6 – Tela de Varredura de Dispositivos BLE.

Esta interface implementa a funcionalidade de varredura (*scan*) de Bluetooth Low Energy. Ela exibe uma lista de dispositivos BLE disponíveis ao alcance (ex: "ESP32_BLE_01"), juntamente com a intensidade do sinal recebido (RSSI, ex: -50 dBm), permitindo ao usuário selecionar o dispositivo ao qual deseja se conectar.

Varredura de Servidores Wi-Fi AP

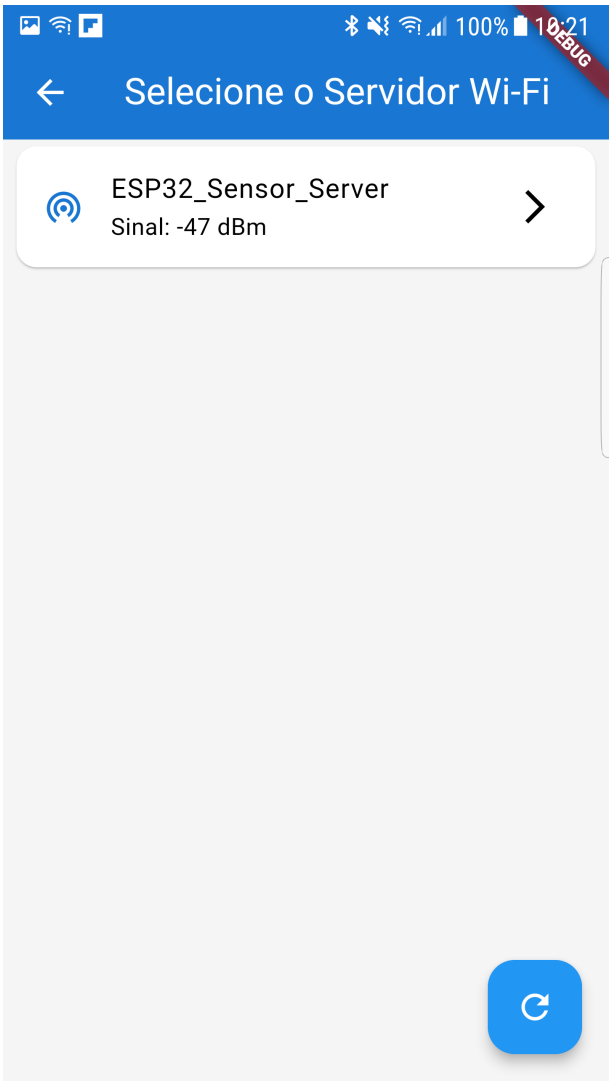

Figura 7 – Tela de Varredura de Servidores Wi-Fi AP.

De forma análoga à tela de BLE, esta interface realiza a varredura de redes Wi-Fi. O objetivo é identificar dispositivos embarcados (ex: "ESP32_Sensor_Server") operando em modo *Access Point*, permitindo uma conexão direta HTTP para coleta de dados.

Modal de Instrução de Conexão Wi-Fi

Figura 8 – Modal de Instrução de Conexão Wi-Fi.

Este é um diálogo modal instrutivo. Devido a restrições dos sistemas operacionais móveis, que não permitem a conexão programática a uma rede Wi-Fi específica, este modal orienta o usuário a realizar a conexão manualmente através das configurações do sistema antes de prosseguir.

Painel de Controle - Estado Ocioso

Figura 9 – Painel de Controle em Estado Ocioso.

Representa o *dashboard* de controle do dispositivo conectado (ex: "ESP32_BLE_01") em estado ocioso. A interface exibe as métricas que podem ser monitoradas (Vazão, Pressão, etc.), mas sem dados ("--"). Ela oferece duas ações principais: "Coletar Dados Salvos"(*download* do histórico do sensor) e "Iniciar Monitoramento"(*streaming* em tempo real).

Painel de Controle - Monitoramento Ativo

Figura 10 – Painel de Controle em Monitoramento Ativo.

Ilustra o *dashboard* em modo de monitoramento em tempo real. Os valores das métricas (ex: Vazão: 15.0 L/min) são recebidos e atualizados dinamicamente na interface. O estado do botão de controle foi alterado para "Parar Monitoramento", permitindo ao usuário encerrar o *streaming* de dados. As cores representam se os sensores estão dentro dos valores críticos.

Confirmação de Coleta de Dados

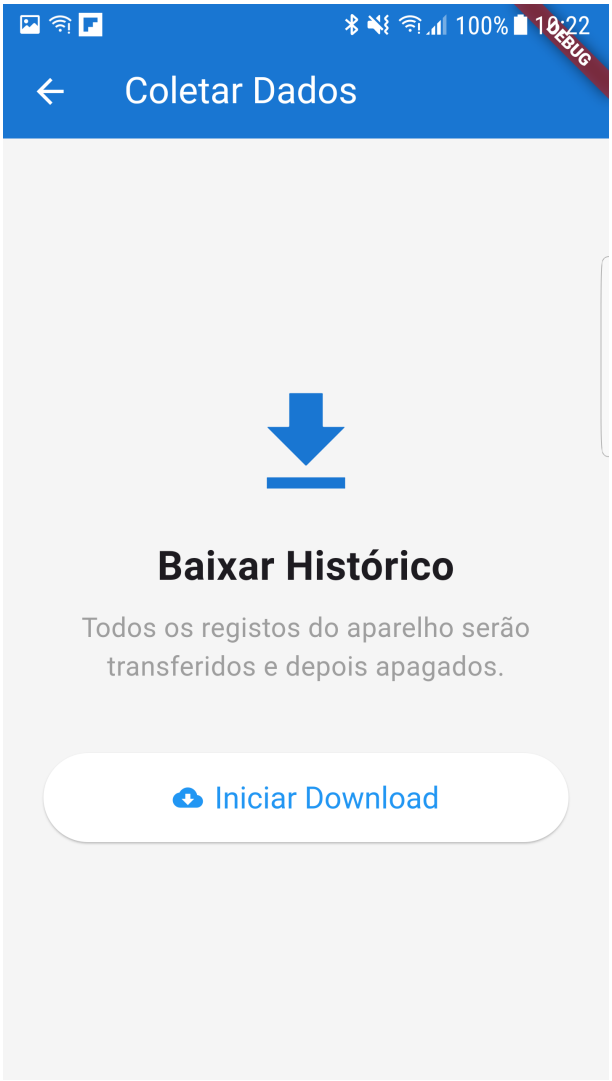

Figura 11 – Tela de Confirmação de Coleta de Dados.

Esta interface (Coletar Dados) atua como uma etapa de confirmação antes de iniciar a transferência de dados do dispositivo de hardware (ex: ESP32) para o aplicativo móvel. Ela informa ao usuário que a ação "Iniciar Download"irá transferir todos os registros armazenados na memória do sensor e, subsequentemente, apagá-los do dispositivo de origem. Esta operação, conhecida como "limpar após a leitura"(*clear-on-read*), é um mecanismo fundamental para gerenciar a memória limitada do dispositivo embarcado e garantir que apenas novos registros sejam coletados em futuras sessões, evitando a redundância de dados.

Interface de Progresso de Download

Figura 12 – Interface de Progresso de Download.

Esta tela fornece *feedback* visual ao usuário durante a transferência de dados do dispositivo embarcado para o aplicativo móvel (acionado pelo botão "Coletar Dados Salvos"). Ela exibe um indicador de progresso circular e um contador de registros (ex: 4 / 45).

Sincronização com Servidor Central [8]

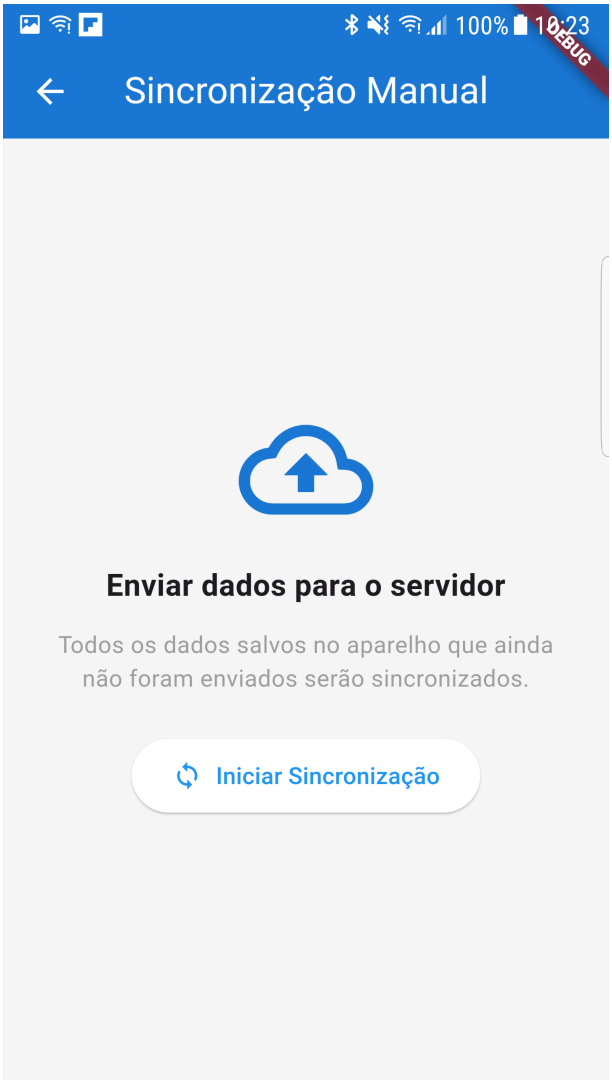

Figura 13 – Tela de Sincronização com Servidor Central.

Esta interface gerencia o *upload* dos dados. Ela é responsável por enviar todos os registros armazenados no banco de dados local do *smartphone* (coletados em sessões anteriores) para o servidor central (nuvem), garantindo a persistência e centralização dos dados.

Histórico de Leituras Locais

Figura 14 – Tela de Histórico de Leituras Locais.

Esta tela exibe uma lista de todas as sessões de coleta de dados que foram salvas no dispositivo móvel. Cada entrada no histórico detalha o dispositivo de origem, a data/hora da coleta, o número de registros baixados e os metadados de geolocalização (Latitude/Longitude) da operação.

Visualização de Registros Detalhados

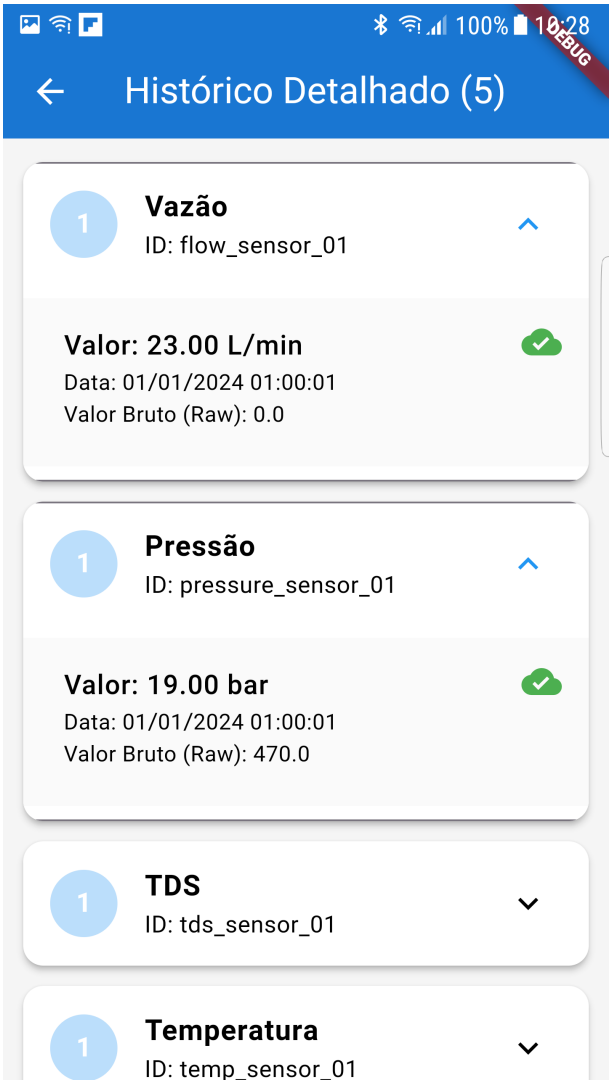

Figura 15 – Tela de Visualização de Registros Detalhados.

Esta tela (Histórico Detalhado) exibe o conjunto de registros individuais que compõem uma única sessão de coleta (previamente selecionada na tela `leitura.jpg`). Para cada métrica (Vazão, Pressão, etc.), a interface detalha:

1. O **Valor Processado** (ex: 23.00 L/min), que é a leitura já convertida para uma unidade de engenharia.
2. O **Valor Bruto (Raw)** (ex: 470.0), que representa o dado original do sensor (como um valor de ADC ou uma contagem), antes de qualquer calibração ou conversão.
3. O **Carimbo de Data/Hora (timestamp)** exato da medição.

8 Discussões

Durante o desenvolvimento das funcionalidades de comunicação via Bluetooth, foram identificados alguns desafios técnicos relacionados à compatibilidade entre o Flutter e a manipulação de conexões BLE.O principal desafio técnico do projeto reside na ausência de uma API nativa para a comunicação via Bluetooth no core do framework Flutter. Tal característica impõe a dependência de bibliotecas de terceiros, mantidas pela comunidade, o que introduz vulnerabilidades inerentes ao desenvolvimento, como a disparidade de comportamento entre as plataformas (Android e iOS) e o risco de descontinuidade ou abandono da manutenção desses pacotes.

Para mitigar esses problemas, optou-se pela utilização da biblioteca Flutter Blue Plus, que se destaca por ser uma das soluções mais robustas e atualizadas disponíveis para o desenvolvimento de aplicações Flutter com suporte a Bluetooth Low Energy. Essa biblioteca oferece uma interface relativamente estável e bem documentada, permitindo realizar operações como escaneamento de dispositivos, conexão, leitura e escrita de características BLE.

Apesar da adoção dessa solução, é importante ressaltar que persistem alguns desafios, como a necessidade de lidar com permissões específicas de cada sistema operacional, gerenciamento de desconexões inesperadas e possíveis interferências no ambiente, que afetam diretamente a qualidade da comunicação Bluetooth.

O desenvolvimento da funcionalidade de conexão Wi-Fi apresentou menos obstáculos, por se tratar de uma integração mais direta com o ESP32, especialmente quando se utiliza a comunicação via protocolo HTTP, que não depende de bibliotecas específicas no Flutter para redes Wi-Fi.

9 Trabalhos Futuros

Durante o desenvolvimento do aplicativo, diversas funcionalidades essenciais foram implementadas com sucesso, garantindo a comunicação eficiente entre o dispositivo móvel e os módulos sensores. No entanto, algumas melhorias e expansões podem ser exploradas em trabalhos futuros, de modo a tornar o sistema mais completo, automatizado e seguro.

Uma das principais evoluções previstas é a implementação de um sistema de autenticação de usuários. Embora o aplicativo já disponha de um mecanismo de login persistente, atualmente não há validação das credenciais dos operadores. Para um uso em escala institucional, será necessário integrar o sistema a uma base de dados contendo o cadastro de todos os operadores autorizados, permitindo o controle de acesso e a rastreabilidade das operações realizadas. É importante, contudo, considerar que, caso essa base de dados seja hospedada na nuvem, o aplicativo deverá possuir um modo de operação offline para situações em que não haja conectividade com a Internet.

Outro aprimoramento relevante é o desenvolvimento de um sistema de conexão automática. Na versão atual, o operador precisa escolher manualmente o tipo de comunicação (Wi-Fi ou BLE) e realizar todo o processo de seleção e pareamento. A implementação de um mecanismo de conexão inteligente — baseado em proximidade e disponibilidade do sinal — permitiria que o aplicativo selecionasse automaticamente o melhor modo de comunicação, reduzindo o tempo de configuração e tornando o processo mais transparente para o operador em campo.

Por fim, destaca-se a necessidade de aprimorar o sistema de alertas e notificações. Atualmente, o aplicativo exibe visualmente quando um sensor apresenta valores fora dos limites críticos, mas não emite notificações sonoras ou push. A adição de um mecanismo de aviso ativo, capaz de enviar alertas mesmo com o aplicativo em segundo plano ou com a tela bloqueada, seria de grande utilidade para garantir a segurança operacional dos equipamentos — especialmente em situações em que o monitoramento contínuo da pressão e do estado dos sensores é fundamental para evitar danos ao sistema

Bibliografia

[1] Bluetooth SIG, Inc. *Bluetooth Technology Overview*. Rel. técn. O documento aborda as topologias, os casos de uso e as diferenças fundamentais entre as tecnologias Bluetooth Clássico e Low Energy. Disponível em: `https://www.bluetooth.com/learn-about-bluetooth/tech-overview/`. Bluetooth Special Interest Group (SIG), 2022.

[2] Shady Boukhary e Eduardo Colmenares. "A clean approach to flutter development through the flutter clean architecture package". Em: *2019 international conference on computational science and computational intelligence (CSCI)*. IEEE. 2019, pp. 1115–1120.

[3] Comitê Gestor da Internet no Brasil (CGI.br). *Pesquisa sobre o Uso das Tecnologias de Informação e Comunicação nos Domicílios Brasileiros - TIC Domicílios 2023*. Rel. técn. A pesquisa anual do CGI.br analisa o acesso e o uso da internet no Brasil. As análises por área, como a rural, demonstram como a conectividade e as habilidades digitais são essenciais para o desenvolvimento de novas oportunidades econômicas e sociais. Disponível em: `https://cetic.br/pt/pesquisa/domicilios/`. Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação (Cetic.br), 2024.

[4] Renato Dagnino. "Tecnologia social: ferramenta para construir outra sociedade". Em: *Tecnologia social: ferramenta para construir outra sociedade*. 2010, pp. 297–297.

[5] Ministério do Desenvolvimento Regional. *Documento Base do Programa Água Doce*. Rel. técn. Acesso em: 17 nov. 2025. Governo Federal, 2012. URL: `https://www.gov.br/mdr/pt-br/assuntos/seguranca-hidrica/programa-agua-doce/ANEXO_I__PAD__Documento_Base_Final_2012.pdf`.

[6] Espressif Systems. *ESP32 Series Datasheet*. Rel. técn. Version 5.1. Disponível em: `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`. Espressif Systems, mai. de 2023.

[7] Amir Farmanesh. "Developing a cross-platform mobile application for health data integration and capture from multi devices using Flutter". Tese de dout. ETSI_Informatica, 2024.

[8] Gabriel Martins Medeiros Fialho. *Servidor para sistema de Monitoramento e Controle de Estações de Tratamento de Água*. Relatório Final de Projeto Orientado em Computação 2. Orientador: Thiago Ferreira de Noronha. Coorientadora: Anolan Milanés. Universidade Federal de Minas Gerais, jun. de 2025.

[9] Debjani Ghosh et al. “Smart saline level monitoring system using ESP32 and MQTT-S”. Em: *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE. 2018, pp. 1–5.

[10] Google. *Flutter: Build apps for any screen*. Documentação Oficial. Disponível em: `https://flutter.dev/`. 2025.

[11] Governo Federal do Brasil. *Programa Água Doce*. Folder Institucional - Ministério do Desenvolvimento Regional. Disponível em: `https://www.gov.br/mdr/pt-br/assuntos/seguranca-hidrica/programa-agua-doce/Folder_Institucional_AJUSTADO.pdf`. s.d.

[12] Mahmoodul Hasan et al. “A Review on the Legislative, Economic, and Technical Aspects of Unmanned Aerial Vehicles (UAVs) in Water Quality Monitoring”. Em: *Water* 14.20 (2022). O estudo analisa e confirma os benefícios econômicos de tecnologias de sensoriamento remoto para monitoramento de água, destacando a redução de custos com mão de obra e transporte em comparação com métodos tradicionais., p. 3332.

[13] Vinay Kumar. “Development of a control panel system and Android Bluetooth app for real-time integrated sensors.” Em: *Current Science (00113891)* 125.12 (2023).

[14] Secretaria do Meio Ambiente do Estado da Bahia. *Programa Água Doce (PAD) – Bahia*. `https://www.ba.gov.br/meioambiente/265/programa-agua-doce-pad`. Acesso em: 17 nov. 2025. s.d.

[15] Víctor Merino Rueda. “Aplicación basada en Flutter para el control de dispositivos BLE”. B.S. thesis. Universitat Autònoma De Barcelona, 2021.

[16] Nikola Mitrović et al. “Implementation and Testing of Websocket Protocol in ESP32 based IoT systems”. Em: *Facta Universitatis, Series: Electronics and Energetics* 36.2 (2023), pp. 267–284.

[17] Roberto Pasic, Ivo Kuzmanov e Kokan Atanasovski. “ESP-NOW communication protocol with ESP32”. Em: *Journal of Universal Excellence* 6.1 (2021), pp. 53–60.

[18] Rap Payne. “Developing in flutter”. Em: *Beginning App Development with Flutter: Create Cross-Platform Mobile Apps*. Springer, 2019, pp. 9–27.

[19] Programa Água Doce. *Site Oficial do Programa Água Doce*. `https://www.gov.br/mdr/pt-br/assuntos/seguranca-hidrica/programa-agua-doce`. Acessado em maio de 2025. 2025.

[20] Secretaria dos Recursos Hídricos do Ceará. *Programa Água Doce – PAD Ceará: Relatório de Gestão 2022*. Rel. técn. Acesso em: 17 nov. 2025. Governo do Estado do Ceará, 2022. URL: `https://www.srh.ce.gov.br/wp-content/uploads/sites/90/2022/11/V3-RELATORIO-GESTAO-2022_COMPLETO_compressed.pdf`.

[21] Phanuwut Rojnarong e Wanchalerm Pora. “Signal Strength and Energy Consumption on Various Internet of Things Communication Protocol Using Heltec ESP32 LoRa”. Em: *2024 21st International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE. 2024, pp. 626–630.

[22] Alexandre Saia. *O Acordo de Gestão do Programa Água Doce: Importância da Gestão Compartilhada dos Bens Comuns para o Desenvolvimento Local*. Rel. técn. Trabalho de Conclusão de Curso. Disponível em: `https://repositorio.enap.gov.br/jspui/handle/1/3479`. Escola Nacional de Administração Pública (ENAP), 2018.

[23] Borade Samar Sarjerao e Amara Prakasarao. “A low cost smart pollution measurement system using REST API and ESP32”. Em: *2018 3rd International Conference for Convergence in Technology (I2CT)*. IEEE. 2018, pp. 1–5.

[24] J. Silva, A. Santos e M. Costa. “Desenvolvimento de um sistema de monitoramento de nível e qualidade da água utilizando o microcontrolador ESP32”. Em: *Anais do Congresso Brasileiro de Automática (CBA)*. (Referência adaptada de exemplos práticos e artigos sobre o tema para ilustrar a aplicação). 2020.

[25] Luis Javier Soriano Suárez et al. “Aplicación en Flutter para gestión de equipo IOT”. B.S. thesis. Universidad de Granada, 2023.

[26] Universidade Federal da Bahia (UFBA). *Tecnologias Sociais de Convivência com o Semiárido como Estratégia de Mitigação/Adaptação às Mudanças Climáticas no Brasil*. Rel. técn. Disponível em: `https://repositorio.ufba.br/handle/ri/21859`. Repositório Institucional da UFBA, 2016.

[27] WHO and UNICEF. *Progress on household drinking water, sanitation and hygiene 2000-2022: Special focus on gender*. Rel. técn. Relatório do Joint Monitoring Programme (JMP). Disponível em: `https://washdata.org/reports/jmp-2023-wash-households`. World Health Organization (WHO) e United Nations Children’s Fund (UNICEF), jul. de 2023.