UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

Tiago Henrique Rodrigues Pedrosa Gonçalves

PilotStation: Estação de Controle de Múltiplos Drones Baseada em Web

Recife

2025

Tiago Henrique Rodrigues Pedrosa Gonçalves

PilotStation: Estação de Controle de Múltiplos Drones Baseada em Web

Trabalho apresentado ao Programa de Graduação em Engenharia da computação da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: Engenharia de Software e Robótica

Orientador (a): Edna Natividade da Silva Barros

Recife

2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Gonçalves, Tiago Henrique Rodrigues Pedrosa.
PilotStation: Estação de Controle de Múltiplos Drones Baseada em Web / Tiago Henrique Rodrigues Pedrosa Gonçalves. - Recife, 2025.
38 p. : il., tab.

Orientador(a): Edna Natividade da Silva Barros
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2025.
Inclui referências.

1. Engenharia de Software. 2. Robótica. 3. Drones . 4. Aplicações Web. I. Barros, Edna Natividade da Silva. (Orientação). II. Título.

000 CDD (22.ed.)

TIAGO HENRIQUE RODRIGUES PEDROSA GONÇALVES

PilotStation: Estação de Controle de Múltiplos Drones Baseada em Web

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Engenharia da Computação.

Aprovado em: 05 / 08 / 2025

BANCA EXAMINADORA

_____

Profa. Dra. Edna Natividade da Silva Barros (Orientadora)

Universidade Federal de Pernambuco

_____

Prof. Dr. Adrien Durand-Petiteville (Examinador Interno)

Universidade Federal de Pernambuco

Dedico este trabalho aos meus pais, meus maiores incentivadores, que sempre trabalharam com dedicação para que eu pudesse me dedicar aos estudos. Dedico também à minha família — irmãos, primos, tios e avós — por todo o amor e apoio constante. Dedico à minha namorada, pelo incentivo diário, pelos conselhos e por estar sempre ao meu lado. Por fim, dedico aos meus amigos que fiz durante o curso, vocês foram essenciais durante toda essa trajetória.

AGRADECIMENTOS

Agradeço especialmente a minha professora orientadora, Edna Natividade, que me orientou durante a elaboração desse trabalho, me auxiliando na criação do TCC, me mostrando pontos de melhoria e como eu poderia expressar minhas ideias de maneira mais clara.

Agradeço também ao meu amigo João Pedro Mafaldo pela imensa ajuda nos testes com o drone, por me ajudar na configuração e execução dos voos em todos os momentos que eu precisei.

Por fim, agradeço a todas as pessoas que contribuíram direta ou indiretamente no desenvolvimento deste presente trabalho.

RESUMO

A crescente utilização de Veículos Aéreos Não Tripulados (VANTs), especialmente em operações coordenadas com múltiplos drones, evidencia a necessidade de aplicações de Estações de Controle Terrestre mais flexíveis e acessíveis. As soluções *Ground Control Station* (GCS) tradicionais, como o *Mission Planner* e o *QGroundControl*, embora robustas, apresentam interfaces de alta complexidade e são baseadas em arquiteturas *desktop*, o que dificulta o uso por desenvolvedores em fases de prototipação e testes ágeis. Este trabalho apresenta o desenvolvimento da *PilotStation*, uma solução baseada em tecnologias web, projetada para preencher essa lacuna, oferecendo uma ferramenta leve e intuitiva para o controle e monitoramento de múltiplos drones. A solução foi desenvolvida sobre uma arquitetura cliente-servidor, com um *backend* em *Python* utilizando *FastAPI* e a biblioteca *pymavlink* para a comunicação direta com os veículos através do protocolo *Micro Air Vehicle Link* (MAVLink), e um *frontend* em *React* para a interface do usuário. A validação da plataforma foi realizada em ambientes de simulação com o *Gazebo*, onde foi comprovada a capacidade de gerenciar até 10 drones simultaneamente sem degradação significativa do desempenho, e em testes com um drone real, que corroboraram a consistência e a viabilidade da aplicação. Os resultados demonstram que o *PilotStation* cumpre seus objetivos, posicionando-se como uma ferramenta eficaz e de fácil utilização, otimizada para o ciclo de desenvolvimento e validação de projetos com veículos aéreos.

Palavras-chaves: Estação de Controle Terrestre, Drones, Múltiplos Veículos, MAVLink, Aplicações Web, Visualização 3D.

ABSTRACT

The increasing use of Unmanned Aerial Vehicles, especially in coordinated operations involving multiple drones, highlights the need for more flexible and accessible Ground Control Station applications. Traditional GCS solutions, such as *Mission Planner* and *QGroundControl*, while robust, feature highly complex interfaces and are based on desktop architectures, which hinders their use by developers during prototyping phases and agile testing. This work presents the development of *PilotStation*, a web-based solution designed to fill this gap by providing a lightweight and intuitive tool for controlling and monitoring multiple drones. The solution was developed on a client-server architecture, with a *Python* backend using *FastAPI* and the *pymavlink* library for direct communication with the vehicles via the MAVLink protocol, and a *React* frontend for the user interface. The platform was validated in simulation environments with *Gazebo*, where its capability to manage up to 10 drones simultaneously without significant performance degradation was demonstrated, and in tests with a real drone, which confirmed the consistency and feasibility of the application. The results show that *PilotStation* achieves its objectives, positioning itself as an effective and user-friendly tool optimized for the development and validation cycle of aerial vehicle projects.

Keywords:Ground Control Station, Drones, Multiple Vehicles, MAVLink, Web Applications, 3D Visualization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GCS	*Ground Control Station*
MAVLink	*Micro Air Vehicle Link*
ROS	*Robotic Operating System*
SITL	*Software In The Loop*
VANT	Veículo Aéreo Não Tripulado
VANTs	Veículos Aéreos Não Tripulados

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O uso de VANTs, ou drones, tem experimentado um crescimento exponencial nas últimas décadas, impulsionado por avanços tecnológicos e pela demanda crescente em setores como agricultura,infraestrutura e logística. Esses veículos oferecem soluções eficientes e precisas para tarefas que, de outra forma, exigiriam recursos significativos e tempo considerável.

Na agricultura, os drones têm se mostrado ferramentas valiosas para o monitoramento de culturas, detecção precoce de doenças, aplicação precisa de insumos e mapeamento de terrenos. Estudos demonstram que o uso de drones equipados com sensores multiespectrais permite identificar áreas afetadas por pragas ou deficiências nutricionais, resultando em intervenções mais eficazes e aumento na produtividade das culturas (MAKAM et al., 2024). Além disso, a utilização de drones fornece uma forma eficiente de detectar danos em áreas de difícil acesso, como pontes, linhas de transmissão e torres de comunicação (WOJCIECHOWSKI; WOJTOWICZ, 2023).

Uma abordagem com veículos aéreos que vem ganhando notoriedade é o uso de enxames, que são grupos coordenados de drones operando coletivamente para cumprir missões complexas. Essa estratégia permite ampliar a cobertura espacial, reduzir o tempo de operação e aumentar a resiliência do sistema frente a falhas individuais. Análises prospectivas destacam que o avanço de tecnologias como inteligência artificial embarcada e redes 5G tende a acelerar ainda mais essa adoção em contextos como agricultura de precisão, cidades inteligentes e missões de busca e resgate (SINGH; KUMAR, 2025).

Nesse cenário de crescente complexidade e escala das operações com múltiplos drones, soluções *web-based* têm se mostrado particularmente promissoras, como (POMA et al., 2024). Elas permitem que missões sejam planejadas, monitoradas e ajustadas em tempo real por meio de plataformas centralizadas acessíveis via navegador. A integração de algoritmos de coordenação de voo e interfaces gráficas intuitivas facilitam o gerenciamento simultâneo de vários VANTs por operadores humanos, mesmo em ambientes dinâmicos. Estudos recentes indicam que estações de controle terrestre distribuídas, com suporte a múltiplos veículos, são componentes estratégicos para operações coordenadas em larga escala (MOHAMMADI; ZHANG; LIU, 2023; SCHNEIDER et al., 2024).

No cenário atual, as GCS de código aberto mais consolidadas são o *Mission Planner*(OBORNE,

2024) e o *QGroundControl*(Dronecode Foundation, 2024). Ambas as plataformas oferecem soluções extremamente robustas e um conjunto completo de funcionalidades, desde o planejamento de missões até uma análise de voo completa. Apesar da completude ser muito útil para uso profissional, a grande quantidade de funcionalidades resulta em interfaces densas e de alta complexidade para desenvolvedores, principalmente nas fases de prototipação e testes.

Com isso, o presente trabalho se insere no contexto de oferecer uma aplicação leve e de fácil utilização, projetada para auxiliar desenvolvedores durante a validação de seus projetos com veículos aéreos. Diferentemente das aplicações citadas, que objetivam abranger diversos casos de uso e oferecer muitas funcionalidades, a proposta apresentada fornece uma interface simples que exibe informações essenciais de telemetria, envio de comandos básicos e visualização de logs, auxiliando no processo de desenvolvimento e testes.

1.2 OBJETIVOS

O objetivo desse projeto é desenvolver uma aplicação de Estação de Controle para drones baseada em *web*, com suporte à comunicação com múltiplos veículos utilizando o protocolo MAVLink, visando o monitoramento e envio de comandos em tempo real.

Além do objetivo geral, este projeto contempla os seguintes objetivos específicos:

- **Estabelecer um modelo de comunicação entre cliente e servidor eficiente**, por meio de uma arquitetura *web* que garanta a consistência na visualização dos dados de múltiplos veículos.
- **Implementar a comunicação com os drones por meio do protocolo MAVLink**, devido à sua ampla adoção na indústria de drones e compatibilidade com a maioria das controladoras de voo. O uso do protocolo MAVLink garante robustez, interoperabilidade e integração futura com outras controladores, *firmwares* e até mesmo outros tipos de veículo.
- **Desenvolver funcionalidades de visualização em tempo real**, permitindo que usuários acessem a interface da estação de controle para monitoramento de telemetria, posição, status do sistema e outros parâmetros.
- **Possibilitar o envio de comandos básicos**, para que o usuário possa enviar comandos básicos utilizando a interface web.

- **Implementar um sistema de armazenamento de logs**, para análise de telemetria e de comandos enviados durante o voo.

2 FUNDAMENTAÇÃO

2.1 FUNCIONAMENTO E ESTRUTURA DE MENSAGENS *MAVLINK*

O MAVLink é um protocolo de comunicação amplamente utilizado na robótica, principalmente no desenvolvimento de drones, suportado pelas principais controladoras como *ArduPilot* e *PX4* (KOUBâA et al., 2019). Ele define um conjunto extensivo de mensagens trocadas entre veículos não tripulados e estações de controle em solo GCS.

O protocolo foi projetado para ser leve e eficiente, empregando serialização binária para compactar as mensagens transmitidas (ARANTES, 2019). Cada mensagem MAVLink segue um formato de pacote bem definido que inclui campos como: byte de início (identifica a versão do protocolo), comprimento do dado (*payload*), número de sequência (*SEQ*), ID do sistema (*SYS_ID*), identificador do componente (*COMP_ID*), identificador da mensagem e os bytes de dados do *payload*, seguidos de um *checksum* de 2 bytes. O número de sequência (*SEQ*) varia de 0 a 255 e auxilia na detecção de pacotes perdidos, enquanto o checksum garante a integridade dos dados recebidos. Na figura 1, é possível ver como o pacote é dividido. As definições de cada mensagem (nomes, IDs e campos) são especificadas em arquivos *XML* – esses arquivos servem como referência para gerar bibliotecas em diversas linguagens (TEAM, 2025b). Vale ressaltar que cada *firmware* pode criar mensagens customizadas para se comunicar com aplicações ou outros veículos através da criação de arquivos *XML*.

Figura 1 – *Frame* de um pacote MAVLink

Fonte: WILLEE (2025)

Para garantir a confiabilidade na comunicação, o MAVLink implementa um mecanismo de confirmação de recebimento, *ACK*. Esse sistema é particularmente útil no envio de comandos que alteram o estado do veículo, como comandos de armar, decolar ou alterar modo de voo. O mecanismo de confirmação permite ao operador que enviou o comando saber se o comando foi aceito, rejeitado ou se a mensagem nem mesmo chegou ao veículo. Com isso, aplicações que se comunicam com drones podem implementar fluxos de entrega confiável, aguardando o recebimento do *ACK* após o envio de um comando, podendo reenviar o mesmo comando ou

notificar uma falha caso a confirmação não chegue em um tempo limite.

2.2 APLICAÇÕES DE ESTAÇÃO DE CONTROLE TERRESTRE PARA DRONES

Uma GCS é uma aplicação utilizada em solo para monitorar e controlar veículos não tripulados. Tipicamente, a GCS comunica-se com o drone por uma conexão de telemetria sem fio (rádio, *WiFi*, 4G, etc.) ou via cabo quando em solo. Ela apresenta dados em tempo real sobre o desempenho e posição do veículo, exibindo informações de voo, status de sensores, mapas com a localização do drone, entre outros dados. Além do monitoramento, a GCS permite enviar comandos ao drone durante o voo (por exemplo, alterar o modo de voo, armar ou decolar o veículo) e também é utilizada para configurar parâmetros do *firmware* da controladora e planejar missões antes do voo (TEAM, 2025a). Em resumo, a estação de solo é indispensável, em todas as fases do desenvolvimento de projetos envolvendo drones, desde a configuração inicial do veículo até o monitoramento de operações em tempo real e análise posterior, oferecendo ferramentas de *logs* e análise de dados, em alguns casos.

As aplicações GCS *open-source* mais consolidadas atualmente entre profissionais e entusiastas são o Mission Planner (OBORNE, 2024) e o QGroundControl (Dronecode Foundation, 2024). Elas são consideradas ferramentas completas, que oferecem funcionalidades robustas para operadores de VANTs, abrangendo desde o planejamento pré-voo até análise de dados após missões. A compatibilidade nativa com os principais *firmwares* de código aberto, como o Ardupilot e PX4, e comunidades ativas justificam seus usos e popularidade.

Essas GCS permitem a visualização em tempo real de um conjunto completo de dados de telemetria, como posição do veículo em um mapa, altitude, velocidade e estado da bateria. Além disso, oferecem controle direto sobre o Veículo Aéreo Não Tripulado (VANT) através do envio de comandos essenciais, como armar e desarmar os motores, iniciar uma decolagem autônoma e alterar o modo de voo do veículo. Para missões complexas, os usuários podem planejar rotas através da interface, definindo ações específicas a serem executadas a cada posição.

Apesar dos múltiplos benefícios, a densidade de funcionalidades introduz desafios significativos. Em termos de usabilidade, a grande quantidade de menus e configurações das aplicações podem tornar a interface intimidadora, principalmente para usuários iniciantes que não estão familiarizados com as particularidades desse tipo de solução. Além disso, embora as aplicações suportem a conexão com mais de um drone, a gestão simultânea de vários fluxos de telemetria

e controle de uma frota podem comprometer ainda mais a usabilidade, tornando a interface confusa quando o número de veículos conectados aumenta (POMA et al., 2024).

Adicionalmente, ambas as aplicações são desenvolvidas com tecnologias para aplicações *desktop*, o que exige instalação local e atrela o uso a sistemas operacionais específicos. Essa característica inerentemente limita a flexibilidade, impossibilitando o uso em nuvem e, consequentemente, o acesso remoto por meio de um servidor, além de impossibilitar a colaboração em tempo real entre usuários.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma análise de trabalhos relacionados no contexto de GCS baseadas na *web*. Serão considerados dois artigos que representam abordagens distintas e complementares para o desenvolvimento de tais sistemas. A análise destes artigos serviu como alicerce comparativo no desenvolvimento do presente trabalho.

3.1 CLOUDSTATION: A CLOUD-BASED GROUND CONTROL STATION FOR DRONES

O artigo "CloudStation: A Cloud-Based Ground Control Station for Drones"(HU et al., 2021) tem como objetivo o desenvolvimento de um sistema de GCS totalmente baseado em nuvem que permite o controle remoto de drones de qualquer lugar do mundo a partir de um navegador *web*.

A arquitetura proposta pelos autores se baseou em um modelo de cliente-servidor. O *backend* da solução foi desenvolvido em *Python* e hospedado em um provedor na nuvem, garantindo disponibilidade da solução por toda a internet. Além disso, o servidor conta com um banco de dados relacional para persistir dados dos veículos conectados. A comunicação foi feita utilizando o protocolo MAVLink entre o computador de bordo do drone e o servidor. O *frontend* da aplicação utiliza o protocolo HTTP e *WebSockets* para comunicação com o *backend*, exibindo informações de telemetria e localização do veículo.

3.1.1 Contribuições e funcionalidades

A principal contribuição do trabalho é a capacidade de operar um drone a partir de qualquer lugar do mundo utilizando apenas um navegador, eliminando a necessidade de instalação de um *software* específico. Ademais, o sistema conta com autenticação de usuários e possibilidade de controle de múltiplos drones, permitindo monitoramento e designação de missões.

3.1.2 Limitações

Os autores afirmam que não houve testes de latência das mensagens, o que pode ser um fator importante para casos de uso críticos e para controles mais precisos em tempo real. Outro ponto importante é que, pela proposta da solução, ela inviabiliza testes e desenvolvimento em

locais sem acesso à internet, o que pode impedir o uso em diversos ambientes.

3.2 OPEN-SOURCE WEB-BASED GROUND CONTROL STATION FOR LONG-RANGE INSPECTION WITH MULTIPLE UAVS

O artigo "Open-Source Web-Based Ground Control Station for Long-Range Inspection with Multiple UAVs"(POMA et al., 2024) também apresenta uma GCS baseada em *web*, com o foco principal no gerenciamento de frotas de drones em missões de longo alcance.

A arquitetura proposta utiliza o *Robotic Operating System* (ROS)(MACENSKI et al., 2022) para fazer a comunicação entre o sistema desenvolvido e os veículos, permitindo que a aplicação seja agnóstica ao tipo de drone e ao *firmware* adotado por ele. O uso do ROS serve como uma camada de abstração que permite a integração com qualquer veículo que possua um computador de bordo capaz de se comunicar utilizando as ferramentas do ROS. O *frontend* da aplicação foi desenvolvido em *React* e a comunicação com o servidor é feita através de *WebSockets* e *API REST*.

3.2.1 Contribuições e funcionalidades

A principal contribuição do trabalho é o uso do ROS para permitir o controle de um enxame heterogêneo de veículos, com diferentes *hardwares* e *firmwares*. Segundo os autores, a interface da aplicação foi pensada para ser minimalista e permitir a conexão de múltiplos drones sem sobrecarregar o operador da aplicação com muitas informações. O sistema projetado foi validado em simulações com até oito VANTs e em situações reais de inspeção de linhas de energia.

3.2.2 Limitações

O uso do ROS, apesar de ser importante para garantir uma compatibilidade com diversos tipos de veículos e sistemas, adiciona um grande acoplamento inerente ao sistema. A arquitetura altamente dependente dessa ferramenta, exige que todos os veículos possuam um computador de bordo com capacidade para executar ROS, como consequência, além de criar dependências de versões, aumenta a complexidade da configuração e manutenção da frota e do sistema como um todo. (CANELAS et al., 2024)

4 METODOLOGIA

4.1 VISÃO GERAL

O sistema proposto, PilotStation, é um sistema que implementa uma GCS desenvolvido com tecnologias *web*, que permite ao usuário conectar-se a um ou mais veículos aéreos que se comunicam com o protocolo MAVLink e recebem, em tempo real, informações sobre o estado operacional de cada um deles, por meio de uma interface simples e intuitiva.

O processo de concepção da interface e a definição das funcionalidades foram fortemente influenciados pela experiência prática do autor em atividades de desenvolvimento de soluções com drones. Essa vivência, adquirida ao longo de testes e validações em cenários reais e simulados, permitiu identificar quais informações de telemetria e comandos são mais relevantes durante as fases de prototipação e validação, bem como quais elementos de interface auxiliam o usuário de maneira eficiente. Assim, a organização das informações e nível de detalhamento dos dados apresentados na aplicação foram orientadas por demandas observadas em contextos reais, buscando oferecer uma solução simples, objetiva e adequada ao gerenciamento simultâneo de múltiplos veículos.

Dessa forma, a solução integra um mapa 3D interativo para visualização da posição relativa e orientação dos drones, proporcionando uma compreensão clara da situação de voo. Adicionalmente, o PilotStation possibilita a visualização e modificação dos parâmetros do *firmware* dos veículos conectados, com suporte atual para ArduPilot. Todas as sessões de voo são registradas em arquivos de log, contendo dados relevantes de telemetria e comandos emitidos pela aplicação. Esses registros podem ser acessados e analisados em uma página específica dedicada à leitura individual de logs por veículo, contribuindo para a identificação de falhas, otimizações e melhorias operacionais. Além disso, o sistema PilotStation está disponível publicamente no GitHub[1].

4.2 ARQUITETURA

A plataforma foi projetada com foco em conectividade com múltiplos drones, interface de fácil utilização e funcionalidades essenciais como visualização de parâmetros, envio de comandos e análise de *logs* de voo. Nessa seção, a proposta do sistema será detalhada, assim

[1] Disponível em: <https://github.com/TiagoHRPG/PilotStation>

como as funcionalidades implementadas. Além disso, será mostrada a arquitetura do projeto e como os módulos se conectam.

4.2.1 Tecnologias

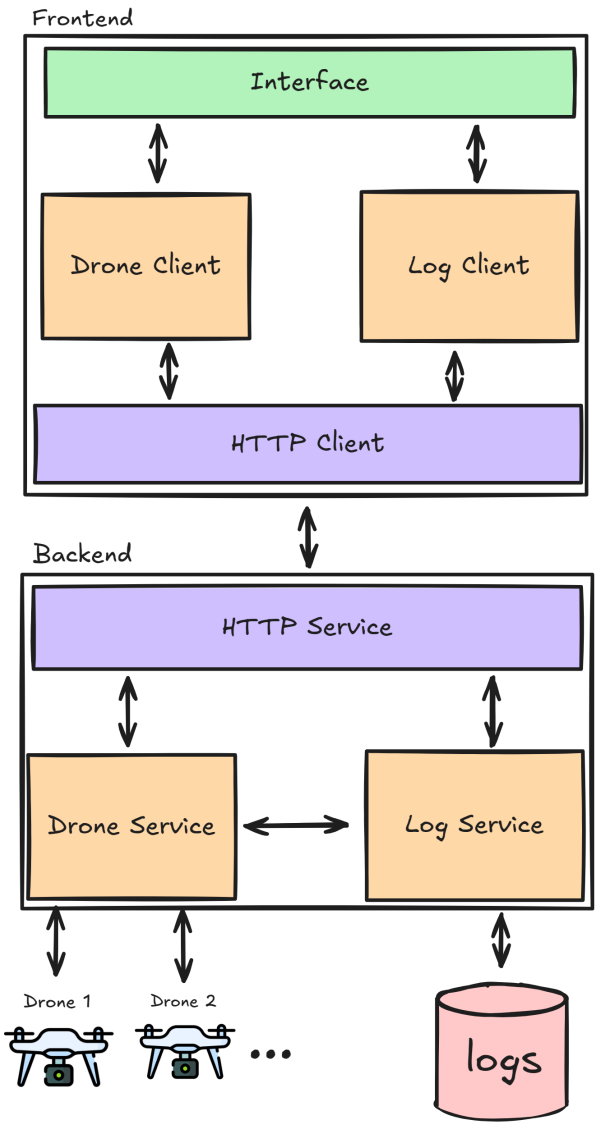
A escolha por uma arquitetura baseada em aplicações *web* foi feita devido à capacidade de instalação em diferentes sistemas operacionais e à facilidade na criação de interfaces fluidas. Além disso, esse tipo de arquitetura permite que diferentes dispositivos acessem simultaneamente as informações da aplicação, o que favorece uma visualização compartilhada do estado dos voos em tempo real. Essa funcionalidade pode ser particularmente útil em ambientes de teste, validação ou demonstração, onde múltiplos usuários podem acompanhar as informações transmitidas pelos drones em diferentes dispositivos.

Como dito anteriormente, a comunicação com o veículo é feita utilizando o protocolo de comunicação MAVLink. O *backend* da aplicação foi desenvolvido em *Python*, utilizando a biblioteca *pymavlink*(TEAM, 2025c) para a abstração da comunicação e facilitação do envio e recebimento de mensagens. Para a criação da *API*, foi utilizado o *framework FastAPI*(RAMíREZ, 2025), uma biblioteca robusta e de fácil utilização que cumpre bem o papel para o sistema, com simplicidade e eficiência. Foi adotado uma arquiteturano estilo *REST* devido à sua simplicidade e compatibilidade direta com o FastAPI. Como a comunicação na aplicação ocorre por meio de requisições pontuais, como envio de comandos e leitura de dados, o modelo *REST* atende de forma adequada, sem a complexidade adicional de protocolos orientados a *streaming* , como *WebSocket*. Para a criação de interfaces, o *framework* escolhido para o *frontend* foi o *React* (PLATFORMS, 2025). A escolha foi motivada pela capacidade da ferramenta em facilitar a criação de interfaces dinâmicas e proporcionar uma renderização eficiente de dados em tempo real, além de possuir uma ampla comunidade de usuários e vasto conteúdo disponível para aprendizado.

4.2.2 Arquitetura do sistema

A arquitetura é divida entre diversos módulos e componentes que desempenham funções específicas e se comunicam da forma representada na figura 2.

Figura 2 – Diagrama da arquitetura do sistema



Fonte: Elaborada pelo autor (2025)

4.2.2.1 Módulos do Backend

- ***Drone Service***: Responsável por orquestrar a comunicação com os drones conectados. Ele intercepta as mensagens MAVLink recebidas dos veículos e as converte em informações estruturadas para o restante da aplicação. Além disso, empacota os comandos gerados pela aplicação e envia para os drones correspondentes. Para garantir um tempo de resposta adequado mesmo com múltiplos drones conectados simultaneamente, o módulo foi implementado com o uso de uma *thread* dedicada para o recebimento das mensagens MAVLink de cada veículo. Essa *thread* realiza a leitura contínua dos pacotes recebidos, atualizando as informações internas do estado de cada drone. Em paralelo, a

thread principal do backend é responsável pelo envio de comandos aos veículos e pela exposição dos dados atualizados para o frontend. Essa separação entre leitura e escrita permite que o sistema mantenha a responsividade e o bom tempo de resposta, mesmo em cenários com múltiplas conexões ativas.

- ***Log Service***: Responsável por gerenciar o ciclo de vida dos *logs* de voo. Ele orquestra a criação, o registro e a recuperação dos dados de telemetria e eventos de cada veículo, garantindo a persistência das informações em arquivos para fins de análise pós-voo.
- ***HTTP Service***: É uma interface de comunicação via API entre o cliente e o servidor. Define as rotas de acesso para envio de comandos e leitura de dados.

4.2.2.2 Módulos do Frontend

- ***Drone Client***: Responsável por gerenciar os estados das instâncias dos veículos do lado do cliente. Ele faz chamadas de API ao servidor regularmente para atualizar as informações dos drones e é responsável por fazer chamadas de comandos solicitados pelo usuário ao servidor.
- ***Log Client***: Responsável por gerenciar as informações de *logs* que são visualizadas na interface, além de enviar para o servidor comandos de remoção de arquivos quando solicitados pelo usuário.
- ***HTTP Client***: É a interface de comunicação do lado do cliente com o servidor. Fazendo as chamadas da API para envio de comandos e leitura de dados.
- ***Interface***: Define as páginas da aplicação e todos os componentes visuais. Permite a interação gráfica com o usuário.

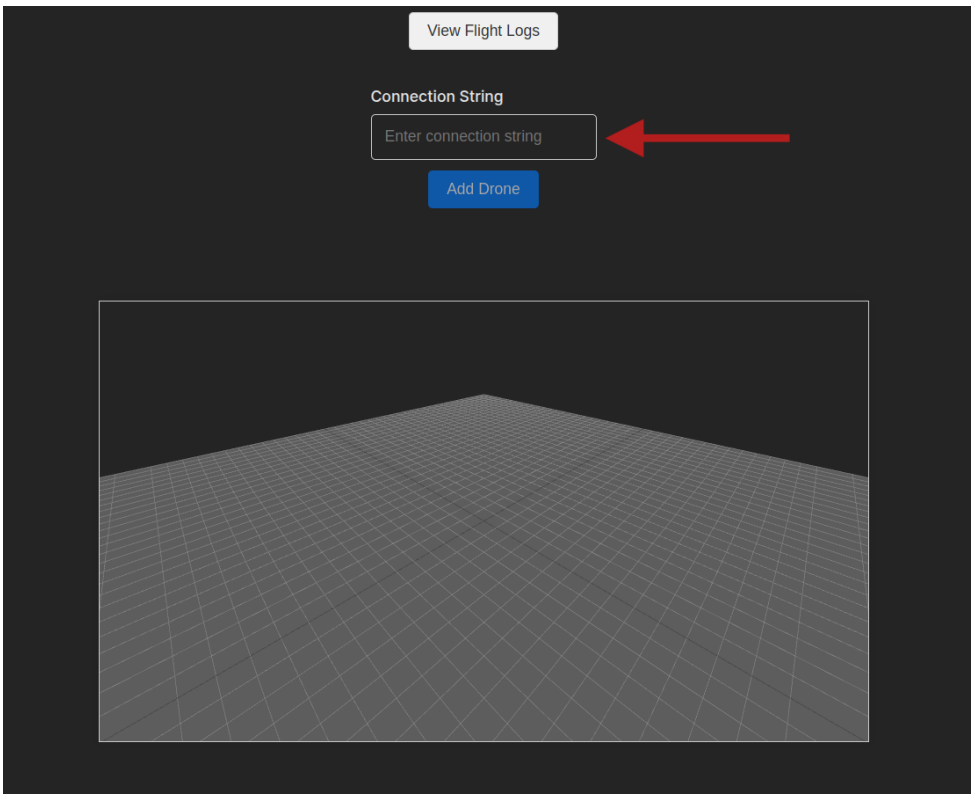
4.3 FLUXO DE UTILIZAÇÃO DA PLATAFORMA

Essa seção apresenta os fluxos básicos de utilização da aplicação.

4.3.0.1 Conexão com o drone

Na figura 3, é possível ver a tela inicial da aplicação, que permite a conexão com um drone através de um campo onde o usuário insere uma *string* de conexão. O formato exigido depende de como o veículo está conectado ao servidor: para uma conexão física via USB, o valor preenchido é o caminho da porta serial do dispositivo. Já para uma conexão de rede, como Wi-Fi, o usuário deve fornecer o endereço IP e porta que o drone está comunicando. Por fim, o usuário pode clicar em conectar e aguardar a conexão ser estabelecida.

Figura 3 – Tela para conexão com veículo



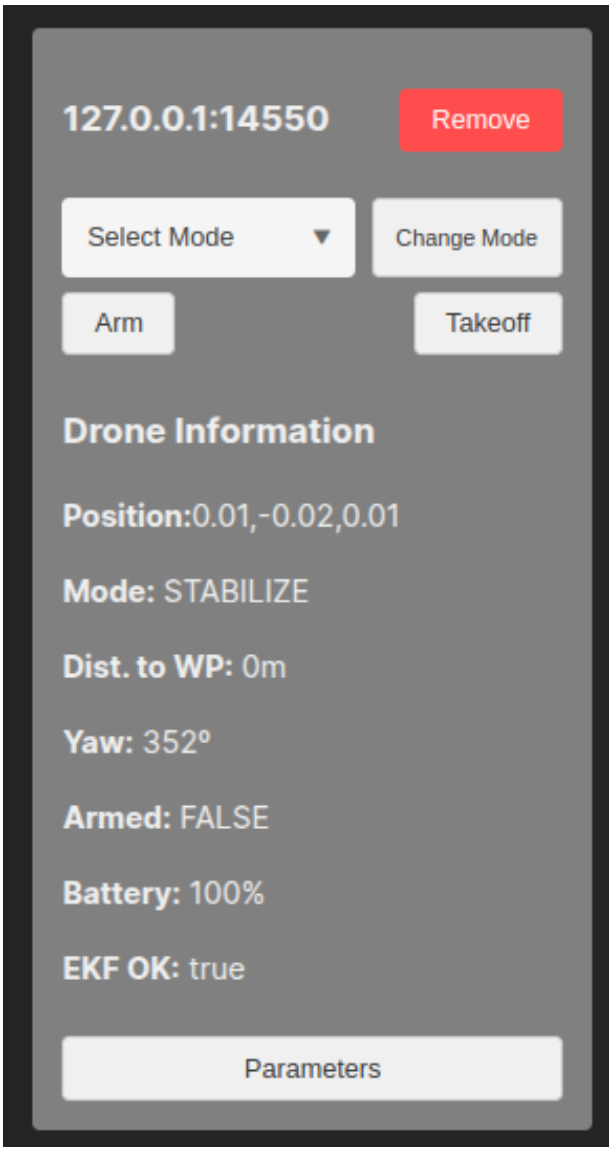
Fonte: Elaborada pelo autor (2025)

4.3.0.2 Visualização de informações

Se a conexão for estabelecida com sucesso, um componente com informações do drone conectado será exibido, bem como alguns botões para envio de comandos básicos como armar, decolar, mudar modo de voo e desconectar. Na figura 4, é possível ver o painel com as principais informações do veículo conectado, incluindo posição, modo de voo, distância até um ponto de interesse (quando aplicável), ângulo de orientação, estado de armamento, nível da bateria

e qualidade dos dados dos sensores. As informações são constantemente atualizadas com os dados enviados via MAVLink pelo veículo.

Figura 4 – Painel de informações do veículo



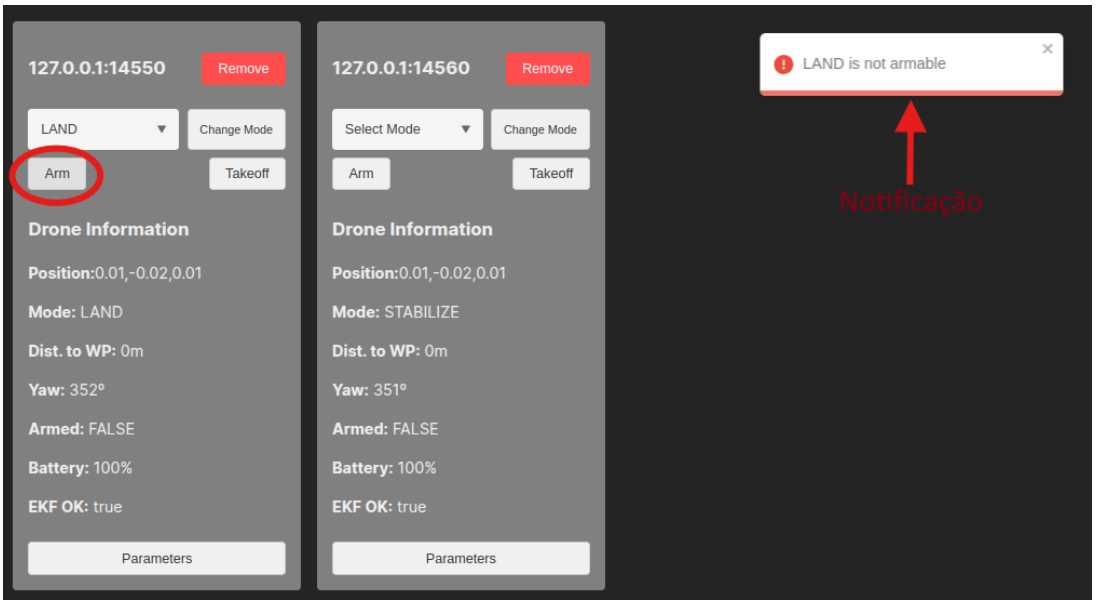
Fonte: Elaborada pelo autor (2025)

4.3.0.3 Envio de comandos e confirmação de recebimento

Com o drone conectado, caso o usuário clique em um comando e o servidor receba uma mensagem de confirmação que o comando não será executado, a aplicação mostra uma notificação ao usuário alertando que a solicitação falhou. A figura 5 exibe uma notificação de falha gerada no momento em que o usuário tenta armar o veículo enquanto este se encontra

no modo de voo "LAND". Como essa ação não é permitida nesse modo, o sistema exibe uma mensagem de erro informando que não é possível armar o drone na configuração de voo atual..

Figura 5 – Notificação de falha ao armar



Fonte: Elaborada pelo autor (2025)

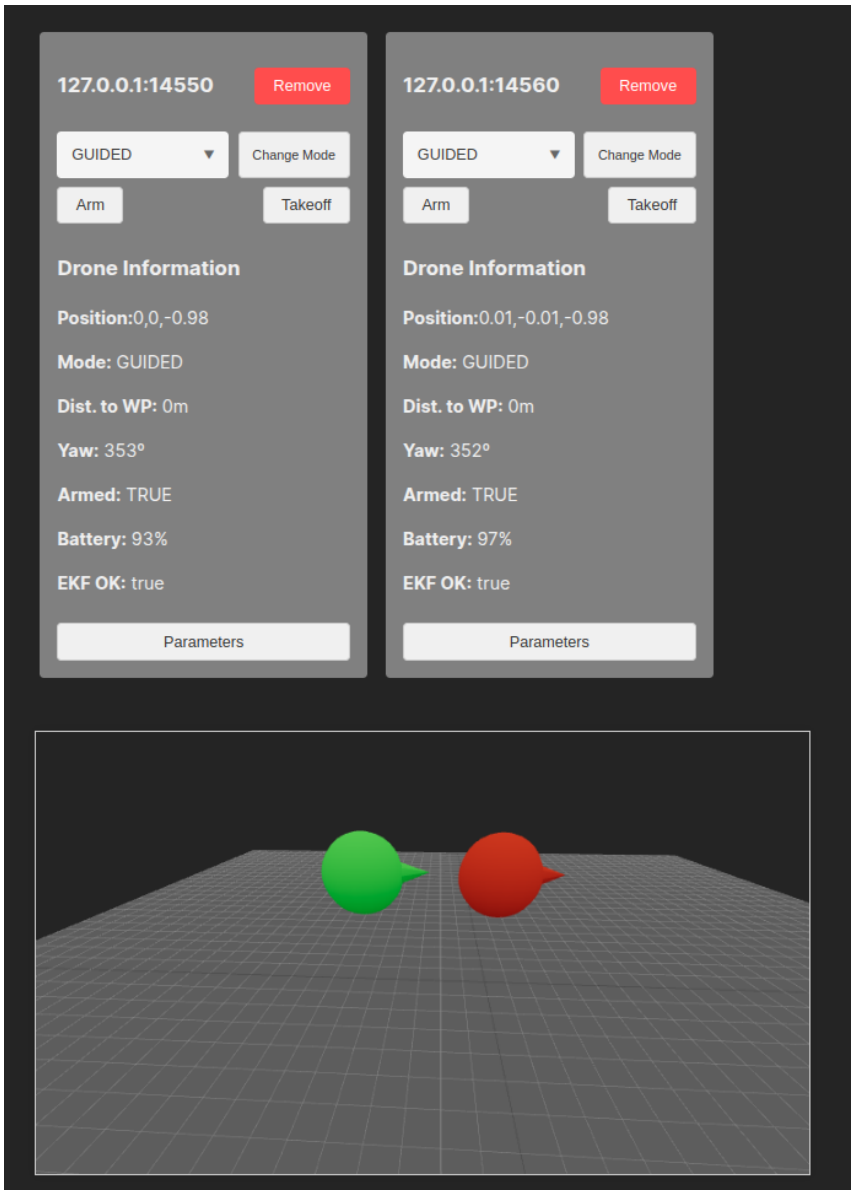
4.3.0.4 Visualização em Mapa 3D

A tela principal da aplicação exibe um mapa 3D que renderiza, em tempo real, a posição e a orientação de todos os drones conectados, com base nos dados de telemetria recebidos. A Figura 6 ilustra um cenário com dois veículos conectados simultaneamente. No mapa interativo, é possível visualizar os objetos que representam cada drone, cuja movimentação e orientação são atualizadas dinamicamente, facilitando o acompanhamento visual da frota durante o voo.

4.3.0.5 Gerenciamento de Parâmetros do Veículo

O sistema oferece uma interface dedicada para o gerenciamento de todos os parâmetros da controladora de voo do veículo. Esses parâmetros abrangem configurações relacionadas à navegação, controle de atitude, sensores, segurança e outros aspectos essenciais do funcionamento do drone. A Figura 7 apresenta a lista completa desses parâmetros, permitindo ao usuário visualizar, pesquisar e editar os valores conforme necessário para ajustar o comportamento do veículo.

Figura 6 – Mapa 3D com dois drones conectados



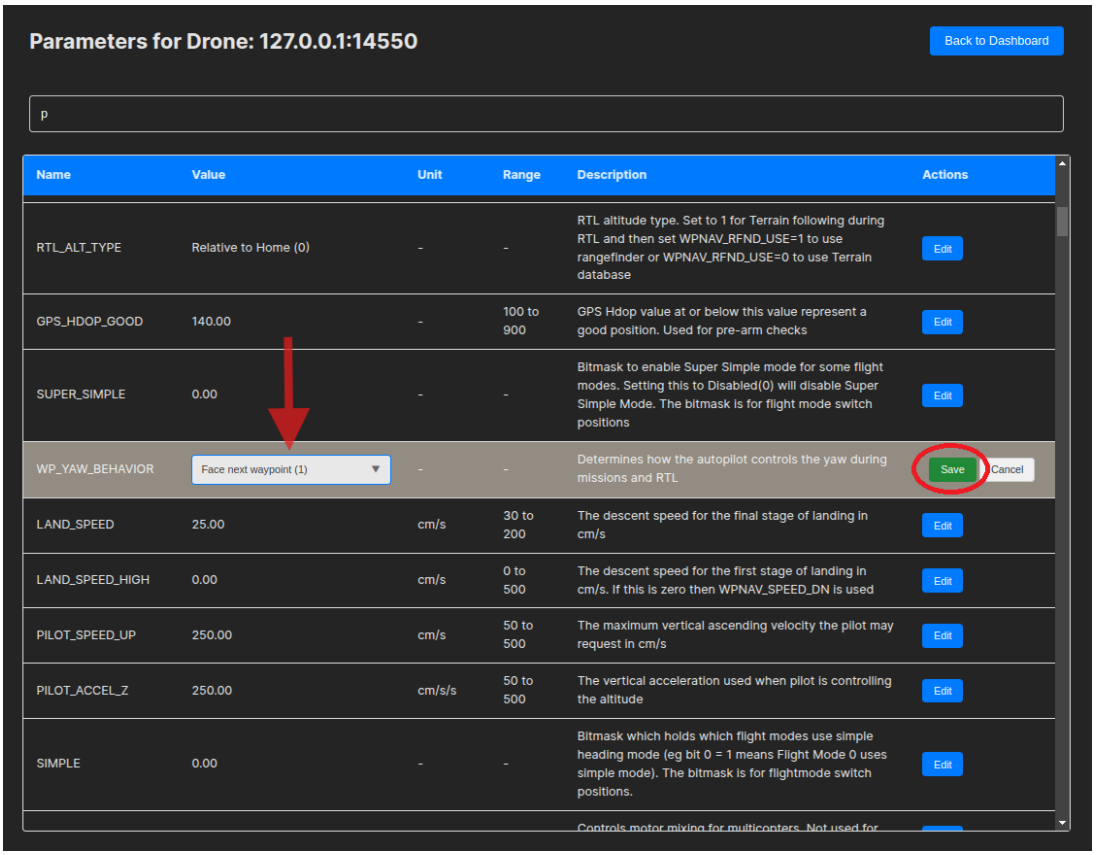
Fonte: Elaborada pelo autor (2025)

O usuário pode modificar o valor de qualquer parâmetro diretamente pela interface. Após a edição, o novo valor é enviado ao drone, e a aplicação exibe uma notificação de sucesso caso a alteração seja aplicada corretamente.

4.3.0.6 Visualização de logs

O programa possui uma seção dedicada à visualização de *logs* de voo, acessível a partir do menu principal. Esta página lista o histórico de todos os voos realizados, organizados por veículo.

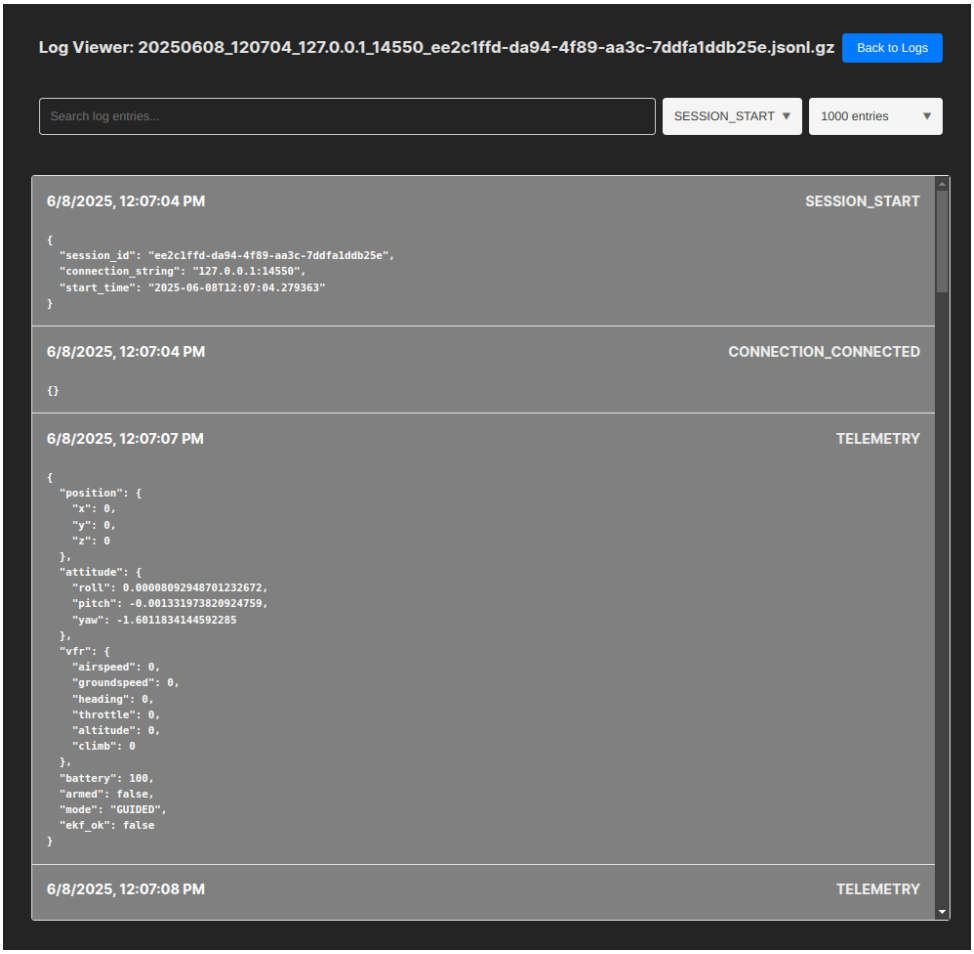
Figura 7 – Tela de gerenciamento de parâmetros



Fonte: Elaborada pelo autor (2025)

Ao selecionar um arquivo de *logs*, é possível visualizar informações de telemetria, comandos enviados e modificações de parâmetros, bem como os respectivos tempos que cada informação foi registrada. Isso permite a realização de análises pós-voo, possibilitando a verificação do estado do drone em cada instante. A figura 8 apresenta um exemplo de *log* extraído após um voo, no qual são exibidas informações sobre a conexão do drone, como a *string* de conexão utilizada e o registro do tempo em que a comunicação foi estabelecida. Também é possível observar dados de telemetria, incluindo posição do veículo, nível de bateria, modo de voo e estado geral do sistema, bem como os comandos transmitidos ao drone durante o voo.

Figura 8 – Tela de visualização de logs



Fonte: Elaborada pelo autor (2025)

5 RESULTADOS

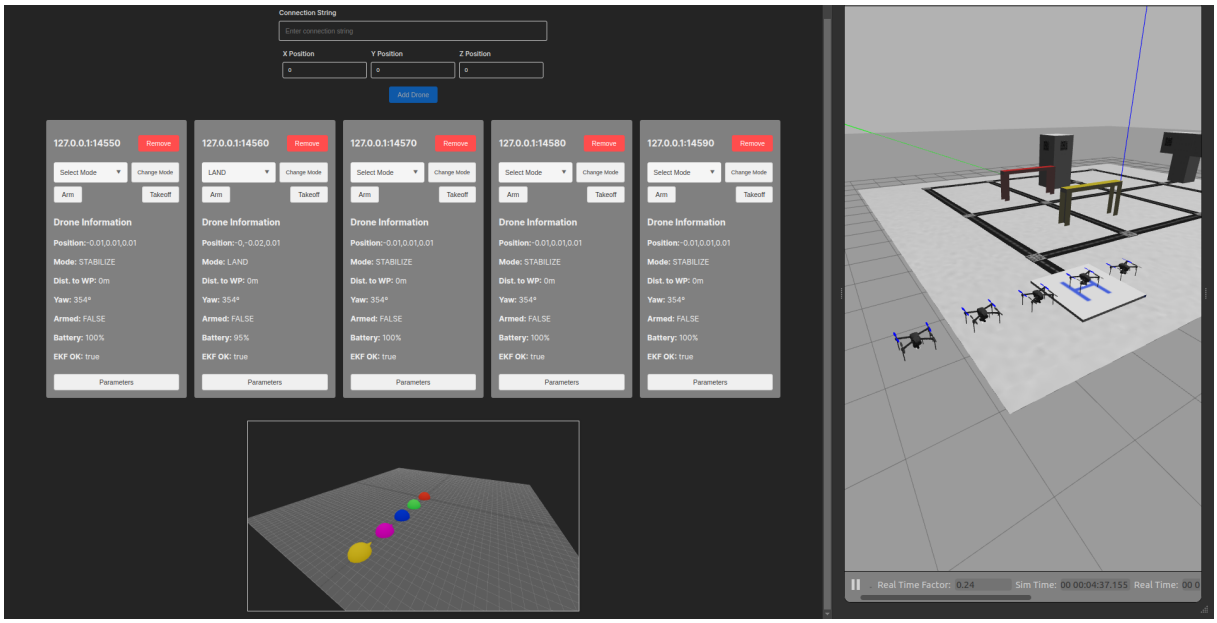
Nesse capítulo serão apresentados os resultados dos testes feitos na aplicação. A solução foi testada inicialmente na simulação e depois o teste foi estendido para drones reais.

5.1 SIMULAÇÃO

A simulação é um passo muito importante no teste de um programa, permitindo teste de novas funcionalidades e reduzindo significativamente o tempo de desenvolvimento, principalmente quando o testes reais de sistemas são difíceis e custosos (POMA et al., 2024). A aplicação desenvolvida foi inicialmente testada na simulação, que foi feita utilizando a plataforma Gazebo (KOENIG; HOWARD, 2004) e os drones foram simulados utilizando a técnica de *Software In The Loop* (SITL), que permite a execução do *firmware* do *Ardupilot* sem a necessidade de um *hardware* específico.

Para isso, uma bateria de testes foi feita, com o intuito de testar as capacidades de uso da solução com um e múltiplos drones, a bateria englobou testes com 1, 2, 5 e 10 drones simultaneamente. Foi constatado que a aplicação foi capaz de fazer o controle de mensagens, atualizando a interface corretamente com o estado dos veículos e sendo capaz de enviar comando para todos. Além disso, o mapa 3D presente na página inicial atualizou corretamente a posição dos drones ao longo do tempo. Na figura 9, é possível, na esquerda, ver a aplicação funcionando com 5 drones conectados e, na direita, o Gazebo com os veículos simulados. Para testar quantitativamente a capacidade da aplicação, foram realizados testes medindo o tempo entre a solicitação do comando de armar os motores por parte do usuário e a resposta recebida pela aplicação da mensagem de ACK, confirmando a execução do comando. A tabela 1 mostra os resultados para os 4 casos supracitados, o mesmo teste foi feito 5 vezes em cada situação e sua média e desvio padrão foram calculados para assegurar que o valor medido não foi devido a alguma outra condição externa intermitente. É notório que o aumento no número de veículos conectados não alterou de maneira significativa o tempo de resposta, o que pode ser justificado pela arquitetura adotada, que isola o recebimento contínuo das mensagens MAVLink em uma *thread* dedicada. Essa abordagem garante que o envio de comandos e o recebimento das mensagens ocorram sem bloqueios, contribuindo diretamente para a estabilidade dos tempos de resposta observados, mesmo com múltiplos drones conectados.

Figura 9 – PilotStation e Gazebo com 5 drones conectados



Fonte: Elaborada pelo autor (2025)

Tabela 1 – Resultado do tempo de resposta para o comando de armar em simulação (5 Amostras)

Estatística	**1 drone (ms)**	**2 drones (ms)**	**5 drones (ms)**	**10 drones (ms)**
Média	105	104	107	116,8
Desvio Padrão	0	1,09	1,67	14,78
Mínimo	105	103	105	105
Máximo	105	106	109	145

Além dos testes de desempenho, foram realizados testes para validar o funcionamento geral do sistema com mais de um veículo conectado simultaneamente. O teste iniciou com a conexão de dois veículos, seguido do envio de comandos individuais, validando a resposta de cada um deles. Durante o processo, o sistema exibiu corretamente as informações de telemetria e o estado do veículo em tempo real. Adicionalmente, a aplicação foi bem sucedida ao mostrar a lista completa de parâmetros da controladora dos veículos, comprovando a correta comunicação com os drones. Por fim, após desconectar com sucesso dos veículos, a página de visualização de *logs* exibiu todo o histórico do voo, incluindo informações de telemetria dos veículos e os comandos enviados ao longo do tempo. Esses resultados comprovam o correto funcionamento do programa no ambiente simulado. Um vídeo de demonstração do funcionamento do sistema pode ser acessado no link [1].

[1] PilotStation - Demonstração no Gazebo. Youtube. Disponível em <https://youtu.be/Ha2HMEKrRpo>

5.2 TESTES REAIS

Para os testes reais, analisamos o comportamento da aplicação com apenas um drone conectado pela dificuldade de adquirir e configurar uma frota de veículos. O foco principal foi verificar o comportamento da aplicação com a conexão do veículo real através de diferentes formas de conexão e checar se a aplicação continuaria com um tempo de resposta adequado. Na tabela 2, podemos ver o resultado dos tempos dos testes entre o envio de um comando para armar os motores e o recebimento da mensagem de confirmação, tanto para uma conexão USB quanto para uma conexão Wifi, testado em diferentes cenários com o drone a uma distância de 0,5 metro, 5 metros e 10 metros do roteador. Pelos resultados obtidos, podemos ver que não há alteração significativa entre o tipo do conexão, a não ser o atraso inato do Wifi pela distância e forma de propagação, o que, entretanto, não influenciou na experiência de uso da aplicação de maneira notável.

Tabela 2 – Resultado do tempo de resposta para o comando de armar em simulação (5 Amostras)

Estatística	**USB (ms)**	**Wifi 0,5 m (ms)**	**Wifi 5 m (ms)**	**Wifi 10 m (ms)**
Média	104,8	105,0	105,4	105,2
Desvio Padrão	0,45	1,79	1,36	1,94
Mínimo	104	103	104	104
Máximo	105	108	108	109

Adicionalmente, vale analisar que os tempos de resposta dos testes reais se aproximam muito dos testes simulados. O que permite inferir que a aplicação possui uma performance praticamente igual, seja para uso em drones simulados ou reais, desde que o veículo seja conectado de maneira correta e se comunique com o PilotStation através do protocolo MAVLink. Um vídeo de teste da solução em campo pode ser acessado no link[2].

[2] PilotStation - Teste de campo com drone. Youtube. Disponível em <https://youtu.be/AMsIpnV50Wo>

6 CONCLUSÃO E TRABALHOS FUTUROS

Este presente trabalho apresentou o desenvolvimento do *PilotStation*, um sistema de GCS baseado em *web*, motivado pelo crescimento de operações com veículos aéreos e pelas limitações de flexibilidade e usabilidade encontradas em GCS tradicionais. O objetivo principal foi o desenvolvimento de um sistema para monitoramento e envio de comandos em tempo real que suportasse múltiplos drones e que fosse uma ferramenta útil para usuários que desenvolvem projetos com VANTs.

É entendido que os objetivos propostos foram atingidos através do desenvolvimento de uma arquitetura cliente-servidor eficiente, capaz de gerenciar a comunicação com os veículos através do protocolo MAVLink. Além disso, a solução possui uma interface simples que exibe funcionalidades de visualização de informações, envio de comandos básicos, gerenciamentos de parâmetros e visualização de logs de voo.

A validação da aplicação foi realizada por meio de testes em ambientes simulados e com um drone real. Nos testes simulados com a plataforma *Gazebo*, foi possível avaliar a capacidade de gerenciamento de até 10 drones simultaneamente e verificar se a comunicação com os veículos seria eficiente mesmo em situações de múltiplas conexões, o que se provou verdade, já que não houve impacto significativo no desempenho. Os testes com o veículo real buscaram ver se o comportamento em uma situação real seria consistente com os testes feitos na simulação e observou-se que os tempos de resposta foram similares, comprovando sua viabilidade em cenários práticos.

Diante do exposto, o PilotStation se diferencia de (HU et al., 2021) por oferecer uma solução que não depende de acesso à internet para desenvolvimento local. Além disso, apresenta uma arquitetura mais simples e menos acoplada que (POMA et al., 2024), por não ter uma dependência com ROS, focando na simplicidade para o controle de veículos utilizando diretamente o protocolo MAVLink.

Como trabalhos futuros para a evolução da aplicação apresentada, acreditamos ser importante os seguintes aspectos:

- O desenvolvimento de uma interface para criação, envio e monitoramento de missões baseadas em *waypoints*, permitindo o usuário definir rotas e ações para múltiplos drones.
- Garantia de suporte a diferentes tipos de veículos, permitindo o monitoramento de uma frota heterogênea sem se limitar apenas a veículos aéreos. Apesar de ser possível esta-

belecer comunicação com múltiplos veículos utilizando o protocolo MAVLink, cada um pode possuir um conjunto distinto de mensagens, parâmetros e comandos. Isso significa que as informações exibidas na interface e os comandos enviados pelo sistema precisam ser adaptáveis de acordo com as características de cada veículo. Dessa forma, exigindo uma adaptação na arquitetura do sistema para detectar, abstrair e tratar corretamente essas particularidades, garantindo o bom funcionamento da aplicação.

- Introduzir visualizações customizáveis das informações presentes nos painéis, oferecendo personalização para os usuários e adaptando a aplicação às necessidades específicas de cada operação. Essa funcionalidade permitiria que o usuário definisse quais informações são mais relevantes para o seu caso de uso, melhorando a usabilidade e otimizando a apresentação dos dados. No entanto, essa flexibilidade exigiria que a solução fosse capaz de adaptar a interface conforme as configurações definidas, sem comprometer a estabilidade do sistema nem a consistência visual da aplicação.

- Adicionar camadas de segurança na aplicação, como um sistema de autenticação de usuário com permissões para diferentes operadores. Apesar da aplicação ser focada em utilização em um servidor local, é importante garantir que apenas usuários autorizados na rede possam se conectar e acessar informações dos veículos.

REFERÊNCIAS

ARANTES, J. *Sistema autônomo para supervisão de missão e segurança de voo em VANTs*. Tese (Doutorado) — Universidade de São Paulo, Escola de Engenharia de São Carlos, 2019. Universidade de São Paulo Sistema Integrado de Bibliotecas - SIBiUSP. Disponível em: <https://doi.org/10.11606/T.55.2019.TDE-10102019-091702>.

CANELAS, P.; SCHMERL, B.; FONSECA, A.; TIMPERLEY, C. S. Understanding misconfigurations in ros: An empirical study and current approaches. In: *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. New York, NY, USA: Association for Computing Machinery, 2024. (ISSTA 2024), p. 1161–1173. ISBN 9798400706127. Disponível em: <https://doi.org/10.1145/3650212.3680350>.

Dronecode Foundation. *QGroundControl*. 2024. Último acesso em 18-06-2025. Disponível em: <http://qgroundcontrol.com/>.

HU, L.; PATHAK, O.; HE, Z.; LEE, H.; BEDWANY, M.; MICA, J.; BURKE, P. J. "cloudstation:" a cloud-based ground control station for drones. *IEEE Journal on Miniaturization for Air and Space Systems*, v. 2, n. 1, p. 36–42, March 2021. ISSN 2576-3164.

KOENIG, N.; HOWARD, A. Gazebo: An open-source 3d robot simulator. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2004. v. 3, p. 2969–2974.

KOUBâA, A.; ALLOUCH, A.; ALAJLAN, M.; JAVED, Y.; BELGHITH, A.; KHALGUI, M. Micro air vehicle link (mavlink) in a nutshell: A survey. *IEEE Access*, v. 7, p. 87658–87680, 2019. ISSN 2169-3536.

MACENSKI, S.; FOOTE, T.; GERKEY, B.; LALANCETTE, C.; WOODALL, W. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, v. 7, n. 66, p. eabm6074, 2022. Disponível em: <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>.

MAKAM, S.; KOMATINENI, B. K.; MEENA, S. S.; MEENA, U. Unmanned aerial vehicles (uavs): an adoptable technology for precise and smart farming. *Discover Internet of Things*, Springer, v. 4, n. 1, p. 12, 2024. ISSN 2730-7239. Disponível em: <https://doi.org/10.1007/s43926-024-00066-5>.

MOHAMMADI, H.; ZHANG, W.; LIU, T. Web-based ground control systems for coordinated uav missions: Challenges and future directions. *International Journal of Advanced Robotic Systems*, v. 20, n. 3, 2023.

OBORNE, M. *Mission Planner*. 2024. Último acesso em 18-06-2025. Disponível em: <https://ardupilot.org/planner/>.

PLATFORMS, I. M. *React: A JavaScript library for building user interfaces*. 2025. Disponível em: <https://react.dev>. Acesso em: 24 may 2025.

POMA, A. R.; CABALLERO, A.; MAZA, I.; OLLERO, A. Open-source web-based ground control station for long-range inspection with multiple uav s. In: *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*. [S.l.: s.n.], 2024. p. 1385–1392. ISSN 2575-7296.

RAMíREZ, S. *FastAPI: Modern, fast (high-performance) web framework for building APIs with Python 3.7+*. 2025. Disponível em: <https://fastapi.tiangolo.com/>. Acesso em: 24 may 2025.

SCHNEIDER, B.; RUPP, T.; WEIGEL, T.; KRUEGER, J. Development of a ground station for swarm drone operations using ros and web technologies. In: *AIAA SciTech Forum*. [s.n.], 2024. Disponível em: <https://arc.aiaa.org/doi/10.2514/6.2024-0748>.

SINGH, R.; KUMAR, S. *A Comprehensive Insights into Drones: History, Classification, Architecture, Navigation, Applications, Challenges, and Future Trends*. 2025. Disponível em: <https://arxiv.org/abs/2501.10066>.

TEAM, A. D. *Choosing a Ground Station*. 2025. Disponível em: <https://ardupilot.org/copter/docs/common-choosing-a-ground-station.html>. Acesso em: 20 may 2025.

TEAM, A. D. *MAVLink Basics*. 2025. Disponível em: <https://ardupilot.org/dev/docs/mavlink-basics.html>. Acesso em: 20 may 2025.

TEAM, A. D. *pymavlink: MAVLink Micro Air Vehicle Communication Protocol - Python bindings*. 2025. Disponível em: <https://github.com/ArduPilot/pymavlink>. Acesso em: 24 may 2025.

WILLEE, H. *Packat Serialization - Mavlink*. 2025. Disponível em: <https://mavlink.io/en/guide/serialization.html>. Acesso em: 19 may 2025.

WOJCIECHOWSKI, P.; WOJTOWICZ, K. Detection of critical infrastructure elements damage with drones. In: *2023 IEEE 10th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. [S.l.: s.n.], 2023. p. 341–345. ISSN 2575-7490.