Breno Perricone Fischer

FlexStation - uma ferramenta para exibir, monitorar e controlar redes de dispositivos móveis e estacionários em testes de campo

RELATÓRIO DE PROJETO FINAL

DEPARTAMENTO DE INFORMÁTICA
Programa de graduação em Ciência da Computação

Rio de Janeiro
03 de 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Breno Perricone Fischer

FlexStation - uma ferramenta para exibir, monitorar e controlar redes de dispositivos móveis e estacionários em testes de campo

Relatório de Projeto Final

Relatório de Projeto Final, apresentado ao programa de Ciência da Computação da PUC-Rio como requisito parcial para a obtenção do titulo de Bacharel em Ciência da Computação.

Orientador: Prof. Markus Endler

Rio de Janeiro
03 de 2022

Resumo

Perricone Fischer, Breno; Endler, Markus. **FlexStation - uma ferramenta para exibir, monitorar e controlar redes de dispositivos móveis e estacionários em testes de campo**. Rio de Janeiro, 2022. 39p. Projeto de Graduação – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O framework FlexStation fornece as ferramentas necessárias para exibir, monitorar e controlar redes de dispositivos estacionários e móveis em testes de campo. O projeto é uma aplicação web arquitetada inicialmente para servir como estação base de experimentos em campo do projeto Ground-and-Air Dynamic sensors networkS (GrADyS), auxiliando no controle manual de quadricópteros (drones) e sendo ponto central para receber todas as informações relevantes para o monitoramento e análise dos testes. A FlexStation foi construída para ser reutilizável, podendo ser adaptada e configurada pelo usuário para se encaixar em outros projetos além do GrADyS. O objetivo é fornecer ao usuário operador da FlexStation uma interface gráfica que permita o envio de comandos e a visualização, em tempo real, do andamento dos experimentos em campo. A FlexStation também permite a persistência das informações do experimento, para uma análise posterior.

Palavras-chave

estação base, rede de dispositivos móveis, aplicação web

Sumário

1
Introdução

O uso de dispositivos interconectados, para busca inteligente de soluções de variados problemas, está cada vez mais presente. A chamada Internet das Coisas (IoT) é o conceito referente a redes interconectadas de dispositivos físicos, sensores, veículos, edificações, entre outros itens. De acordo com os pesquisadores Mohamed Ali Feki (1) e Shaila Patil (2), a IoT é reconhecida como um componente fundamental da próxima revolução tecnológica. Os benefícios obtidos com a IoT são amplos e englobam um uso mais eficiente dos recursos, melhora na qualidade de vida e aumento de produtividade (3). Por esses e outros benefícios, podemos observar a crescente demanda dessa tecnologia, com a amplitude de projetos de IoT, que vão desde cidades inteligentes até monitoramento ambiental.

Os benefícios são amplos, porém, o desenvolvimento e operação de aplicações IoT costumam envolver algumas dificuldades. As mais comuns a serem citadas são: o mal funcionamento de uma aplicação IoT devido a alta chance de falhas em redes sem fio (wireless), probabilidade de falhas nos dispositivos inteligentes que apoiam uma aplicação IoT (dispositivos IoT), como problemas mecânicos e esgotamento da bateria, e a mobilidade de dispositivos, variando rapidamente o alcance da comunicação wireless entre dispositivos e uma estação de controle com um operador (estação base).

Nesse cenário, ferramentas auxiliares que ajudam no desenvolvimento, controle e manutenção de redes de dispositivos interconectados, são de importância essencial. Controlar e visualizar a comunicação entre dispositivos são conceitos fundamentais para o sucesso de um projeto IoT (4).

Enquanto ferramentas de simulação auxiliam na depuração e desenvolvimento dessa comunicação, testes de campo revelam surpresas em campo. São fatores reais que precisam ser testados e levados em consideração. Um simulador de telecomunicação, por exemplo, dificilmente irá realizar simulações inserindo os efeitos que árvores, rochas ou edificações exercem na comunicação. A refração e reflexão do sinal, nos objetos citados, são exemplos de fatores reais possivelmente apenas testados em campo. Pode ser citado como exemplo, também, a influência da proximidade do solo no funcionamento de um rádio. Quanto maior a proximidade do rádio com o solo, menor o alcance do

sinal. Esse efeito pode ser de difícil reprodução e identificação em simulações. Já para testes de movimento e voos de um enxame de UAVs (drones), por exemplo, a simulação provavelmente não replicará a influência física da inércia e de ventos. Ao passo que os experimentos em campo aproximam os objetos de testes a esses e outras influências importantes a serem levadas em conta.

Uma pesquisa por ferramentas de monitoramento e controle, no contexto, principalmente, de IoT, revela poucas opções gratuitas e expansíveis. Nesse cenário, surgiu a ideia de uma estação base open-source para exibir, monitorar e controlar redes de dispositivos móveis em testes de campo. A FlexStation é uma aplicação web que foi desenvolvida pelo autor, com auxílio de membros do projeto Ground-and-Air Dynamic Sensors Networks (GrADyS) (5), do laboratório Laboratory for Advanced Collaboration (LAC) da PUC-Rio, que conta com recursos para facilitar o processo de teste em campo.

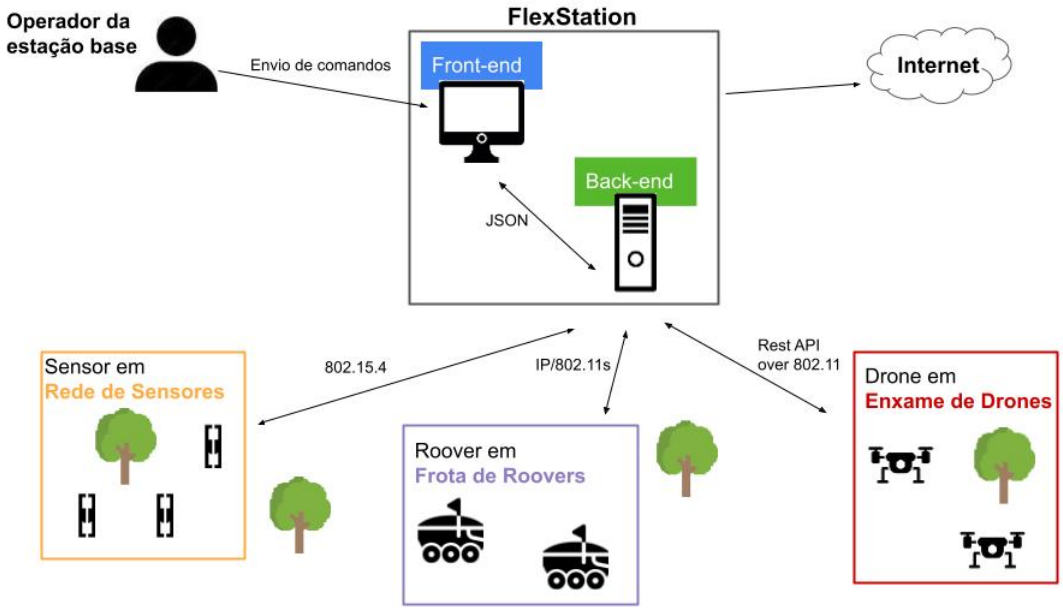

Figura 1.1: Visão geral da FlexStation

A Figura 1.1 representa o funcionamento geral da FlexStation e sua interação com dispositivos IoT. É apresentado um cenário de teste genérico, no qual a FlexStation serve como estação base para monitorar, analisar e controlar os alvos do teste de campo. A FlexStation desempenha um papel central no experimento, sendo a porta de entrada para dados que informam o andamento do teste e a porta de saída para comandos que interagem com os dispositivos IoT. O experimento exemplo envolve três conjuntos de dispositivos modelos possíveis de serem monitorados pela FlexStation: um enxame de drones, uma rede de sensores e uma frota de roovers. Os dispositivos IoT em questão devem ser capazes de enviar informações, como por exemplo a sua localização atual, e re-

ceber comandos, como por exemplo ir para um determinado ponto. A maneira que a comunicação entre a FlexStation e os dispositivos do experimento (comunicação externa) é feita, depende do usuário programador da FlexStation. Antes do teste de campo ser realizado, o usuário programador pré-configura a FlexStation, inserindo o código necessário para estabelecer uma comunicação externa eficiente durante os experimentos, definindo, assim, um protocolo de comunicação externa. A porta de comunicação externa está presente no módulo back-end, representado pela cor verde. Na Figura 1.1 foram exemplificados três formas de comunicação externa a partir dos padrões especificados pelo Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE): "802.15.4"(6) , que indica a utilização de uma rede wireless local de baixa taxa de transmissão (LR-WPAN); "Rest API over 802.11"(7) , retratando uma API para requisições HTTP que opera sob uma rede wireless local (WLAN); "IP/802.11s"(8) , que representa uma rede wireless mesh (WMN), sendo uma alteração no protocolo IEEE 802.11.

Nesse cenário exemplo, portanto, a FlexStation precisaria estar configurada para aceitar e se integrar a essas três redes wireless citadas, para estabelecer comunicações externas eficientes. Na seção de Arquitetura do projeto, será apresentado de forma mais detalhada a interação entre a FlexStation e as formas de comunicação externa exemplificadas na Figura 1.1. A primeira forma de comunicação, "802.15.4", apresenta o uso da FlexStation se comunicando com uma rede wireless (WSN), formada por microcontroladores ESP32. A FlexStation se comunica com um dos nós localmente, através de uma porta serial da máquina que ela está rodando. Um dos nós dessas rede WSN é usado como porta de entrada e saída, um ponto central para mediar a comunicação entre a rede e a FlexStation. A segunda forma de comunicação apresentada, "Rest API over 802.11", indica a comunicação entre a FlexStation e dispositivos IoT através requisições HTTP em cima de uma rede WLAN. Cada dispositivo IoT dessa rede contém um servidor com um endereço IP específico e uma API para rotas de comando. Por fim, a terceira forma de comunicação, "IP/802.11s", não foi alvo de implementação na FlexStation com o projeto GrADyS, porém apresenta a possibilidade de comunicação externa com a FlexStation se integrando com componentes de uma rede WMN.

A comunicação entre os módulos front-end e back-end (comunicação interna) é representado pela troca de mensagens no formato JavaScript Object Notation (JSON) (9) . O usuário operador da estação base pode acompanhar o decorrer do experimento a partir de uma interface gráfica, presente no módulo front-end, representado na Figura 1.1 pela cor azul. O controle manual também pode ser feito pelo usuário operador, através de botões presentes na interface.

O acesso à Internet é essencial para funcionamento correto do mapa virtual, também presente na interface gráfica.

Uma característica inicial planejada para a FlexStation foi torná-la utilizável em outros projetos além do GrADyS, projetando-a para ser extensível e reutilizável. A modularização da arquitetura foi um conceito adotado para tornar a aplicação reutilizável, facilitando inserções de novos protocolos de comunicação com dispositivos externos (dispositivos IoT), potencialmente podendo ser usado para monitorar e controlar dispositivos baseados em uma tecnologia de rede wireless diferente da concebida originalmente na ferramenta. De forma adicional, um exemplo que torna a FlexStation mais extensível é com a possibilidade e facilidade de inserção de novos botões de comandos na interface gráfica. A documentação da FlexStation contém as instruções detalhadas sobre o processo de inserção de novos botões e a explicação dos componentes envolvidos nas etapas de inserção. Como o intuito é tornar a FlexStation reutilizável para outros projetos que utilizem dispositivos IoT, a criação de uma documentação detalhada de como utilizá-la, configurá-la e como expandi-la foi um fator fundamental para melhor usabilidade.

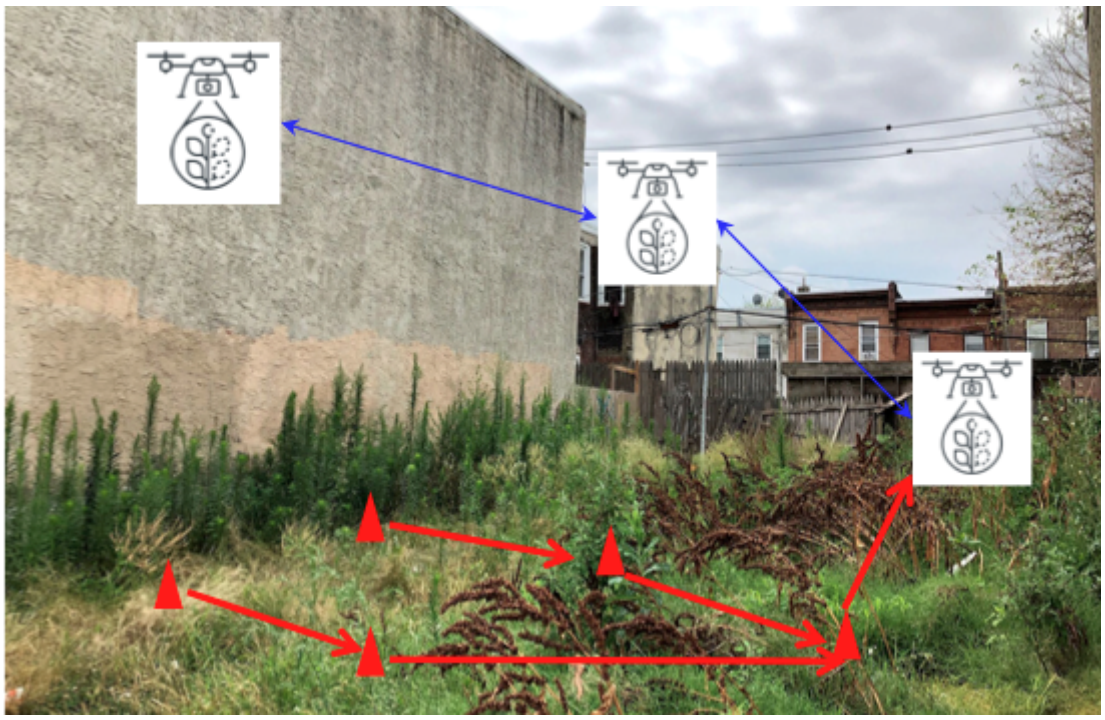

Figura 1.2: Ilustração da interação entre dispositivos IoT em um cenário exemplo

Na Figura 1.2 está ilustrado um cenário exemplo de um projeto que se beneficiaria utilizando a FlexStation. Nesse cenário uma frota de quadricópteros, popularmente conhecidos como drones, trocam dados constantemente entre si e com uma rede de sensores terrestres distribuídos por uma região. A

rede de sensores terrestres constantemente produzem dados a serem analisados, direcionando a informação para nós específicos. Já a frota de drones se organizam para coletar os dados dos nós da rede de sensores de forma eficiente. Com a estação base proposta, seria possível receber as informações coletadas pelos dispositivos externos (drones), avaliando a corretude geral do funcionamento do projeto. Essas informações recebidas são exibidas para o operador em tempo real e armazenadas em arquivos estruturados, para uma posterior inspeção.

A FlexStation poderia ser utilizada, por exemplo, para avaliar a altitude e a velocidade de voo da frota de drones ao longo do experimento. A altitude e velocidade de voo são fatores importantes no cenário exemplo dado, onde um drone precisará sobrevoar nós de uma rede de sensores, para a coleta de dados. Uma coleta de dados eficiente envolve determinar uma faixa de valores específicos para altitude de voo do drone, assim como para a velocidade do drone ao sobrevoar o nó sensor. As informações de voo são enviadas para a FlexStation ao longo do experimento e salvas em um arquivo, podendo ser manipulado para extrair a altitude e velocidade de voo nos momentos da coleta. Problemas durante a coleta podem ser observados com auxílio de todas informações salvas do experimento, comparando o momento da coleta com informações de voo do drone. As informações coletadas podem servir para realizar uma comparação direta com resultados obtidos em simulações do mesmo experimento. Uma diferença de resultados pode indicar uma interferência externa, um fator de campo que não foi levado em conta durante as simulações.

A FlexStation também fornece meios para monitorar a localização em tempo real dos diferentes dispositivos móveis, fazendo uso de um mapa virtual para uma melhor visualização. O estado de atividade de um dispositivo pode ser facilmente observado no mapa, auxiliando o operador na depuração. Um comportamento inesperado durante o experimento pode ser verificado visualmente, em tempo real, pelo operador de testes monitorando o mapa virtual. O desvio da rota de voo ou o mal funcionamento de um drone são exemplos de situações que a FlexStation facilitaria a observação. O controle manual é feito por meio de botões e itens, localizados na interface gráfica web, capazes de disparar comandos para um dispositivo ou conjunto de dispositivos externos. O controle de drones é uma ferramenta importante para o dinamismo e sucesso de um experimento. Um operador, por exemplo, pode disparar um comando que modifica a altitude do voo da frota de drones, adaptando o experimento de acordo com alguma necessidade observada. Outro uso de comandos é para iniciar uma missão específica, informando aos drones pontos com latitude e longitude a serem visitados.

O trabalho envolveu conceitos aprendidos desde o início do curso. O software foi projetado do zero, levando a necessidade de, primeiramente, estudar e escolher ferramentas de desenvolvimento. A elaboração inicial da estrutura do projeto, também foi realizada antes de qualquer código escrito. Disciplinas cursadas, como por exemplo, modelagem de software, interação humano-computador e design, tiveram papel fundamental nessa etapa do processo. A estruturação e escrita da documentação do projeto foi realizada ao longo de todo o processo de desenvolvimento. Um produto importante a se destacar, inserido na documentação, foi o Diagrama de Eventos, que deixa explícito toda a comunicação interna e externa do framework. Disciplinas como Programação Modular, Estruturas de dados avançadas, entre outras, serviram como base para a parte de desenvolvimento e programação do projeto. Finalmente, mostrou-se presente a importância de tópicos aprendidos ao longo do curso, como a persistência de dados, a estruturação de uma página web, métodos de requisição HTTP e grenciamento de dispositivos de uma rede.

2
Situação Atual

O framework foi desenvolvido de forma a auxiliar os testes de campo do projeto Ground-and-Air Dynamic sensors networkS (GrADyS). O processo de desenvolvimento da FlexStation teve participação ativa de integrantes do projeto GrADyS, com auxílio do Prof. Markus Endler, do Dr. Bruno Olivieri, do doutorando Marcelo Paulon e graduando em ciência da computação Thiago Lamenza. Vale ressaltar que, apesar de nascer com uso dentro do projeto GrADyS, o framework é reaproveitável e extensível, podendo ser integrado a diferentes projetos.

A FlexStation será uma adição ao conjunto de frameworks que auxiliam o controle e visualização de aparelhos IoT, em testes de campo, com uma abordagem para ser acessível, reutilizável e extensível. A motivação inicial para a criação da FlexStation surgiu a partir de experiências anteriores de membros do projeto GrADyS. Foi constatado que diversos projetos envolvendo o uso de dispositivos IoT exigiam ferramentas auxiliadoras para os experimentos de campo, seja para auxiliar na visualização dos dispositivos IoT durante o experimento a ser realizado em campo, para controlá-los ou para guardar dados do experimento. As ferramentas criadas para os fins citados costumam estar atreladas projetos específicos, funcionando apenas para o contexto de um projeto e caindo em desuso após o término deste. Sendo assim, é útil a criação de uma ferramenta de uso livre que pudesse ser reaproveitada e que fornecesse uma interface gráfica, auxiliando no monitoramento em tempo real e controle manual de dispositivos IoT, por envio de comandos.

Neste trabalho, foi realizada uma pesquisa, pelo autor, de soluções do mercado e ferramentas similares utilizadas em projetos acadêmicos. O foco dos frameworks encontrados, utilizados em projetos acadêmicos, são específicos para determinado contexto, possivelmente caindo em desuso e no esquecimento após a conclusão do projeto. Posso ressaltar a quantidade elevada de sistemas de controle e mapeamento de quadricópteros, se tratando de soluções de mercado. Algumas soluções de mercado encontradas e frameworks para uso acadêmico são descritos resumidamente abaixo:

– FlytBase (10) - uma plataforma corporativa de automação de drones, com soluções de software pagas, para implantação de drones totalmente

automáticos e conectados à nuvem. Os softwares são vendidos por planos customizáveis, via o canal de vendas. São produtos especializados para soluções com drones, focando em automação de voo, segurança e vigilância, delivery, resposta a emergências, controle de enxame de drones e mapeamento. A proposta principal é fornecer um produto completo para controlar, monitorar e automatizar voos de diferentes modelos de quadcopteros.

- QGroundControl (QGC) (11) - ferramenta open-source, com duas licenças, Apache 2.0 e GPL v3. Assim como no framework citado acima, o foco é em soluções com drones. Podemos destacar os principais recursos do QGC: a possibilidade de configuração completa com veículos aéreos que usam ArduPilot e PX4, automação de voo com planejamento de voo (mission planning), mapa virtual com as posições dos veículos, possibilidade de controle de múltiplos veículos e suporte multi-plataforma.
- GCS: A Ground Control Station for a Multi-UAV Surveillance System (12) - apresentado no paper, essa estação base foi desenvolvida para uma plataforma de planejamento de voos de vigilância, composta por múltiplos veículos aéreos não tripulados (UAVs). O foco da estação base é a validação de experimentos em campo. Esse é um exemplo de projeto que poderia utilizar a FlexStation, sem necessidade de desenvolvimento de uma estação base específica para testes. A estação base do artigo não está disponível para uso comercial.
- Paparazzi Ground Control Station (GCS) (13) - software semelhante ao proposto pelo autor, sendo open-source, com licenças GPL License Agreement e Creative Commons License. Ferramenta extensível, criado com foco principal em autonomia de voo para múltiplos UAVs, como multirotores, aeronaves de asa fixa, helicópteros e aeronaves híbridas. O controle manual de voo foi inserido como funcionalidade secundária Desde o início, a portabilidade foi um requisito funcional importante, assim como a possibilidade de controlar múltiplas aeronaves no mesmo sistema. O foco, como podemos perceber, é a possibilidade de criação fácil de planos de voos dinâmicos, complexos e autônomos. A ferramenta, porém, é pouco amigável para novos usuários (14). Um fator principal dessa complexidade, é que a GCS está inclusa e atrelada a outras funcionalidades do produto Paparazzi.

A FlexStation é uma proposta para a falta de ferramentas open-source, extensíveis, reutilizáveis, com boa usabilidade e que auxilie nas diferentes

necessidades de testes em campo, como visualização do experimento, controle manual de dispositivos e persistência da informação.

Para se ambientar com as funcionalidades essenciais, necessárias para a estação base a ser desenvolvida, os principais softwares semelhantes de mercado, foram avaliados e testados. O estudo das ferramentas Paparazzi e FlytBase foi feito pelo autor, levantando o que seria mais necessário e o que não é essencial na estação base proposta.

A experiência dos integrantes do projeto Gradys também auxiliou na busca por funcionalidades essenciais. A experiência adquirida pela equipe, com esse e outros projetos, envolvem, por exemplo, escrita de planos de voo automáticos, depuração e monitoramento de testes de campo com aparelhos IoT e redes interconectadas.

O controle de dispositivos externos, através da interface, costuma ser um fator importante durante a realização de testes de campo. A possibilidade de envio de comandos, através da estação base, aumenta o dinamismo dos testes. Para a FlexStation, um conjunto de botões foi implementado, conforme a demanda que os testes do GrADyS apresentaram. Para o controle de voo de drones, foi importante adicionar comandos de missões específicas, como retornar para o ponto de origem ou visitar pontos pré-determinados. Também foi necessário um botão capaz de enviar arquivos para um drone específico, via requisições HTTP, contendo um plano de voo, por exemplo.

Como elemento fundamental para o reuso da FlexStation, novos botões de comandos poderão ser adicionados, de acordo com as necessidades dos desenvolvedores de diferentes projetos. O comportamento de reação a um determinado comando, deverá ficar a cargo do desenvolvedor, que irá customizar a FlexStation, implementar. A FlexStation, porém, irá auxiliar no processo de inclusão do botão na interface e na comunicação entre os módulos, até o comando alcançar o protocolo de comunicação com dispositivos IoT.

O foco do projeto, então, torna-se fornecer uma ferramenta para controle e monitoramento de testes em campo, de redes móveis, assim como a possibilidade de controle e planejamento de voo durante os testes.

3
Objetivos

O objetivo da FlexStation é fornecer funções básicas para monitorar, visualizar e analisar de forma detalhada os experimentos realizados em campo envolvendo uma rede de dispositivos. O uso de uma estação base para monitorar e controlar é uma prática comum para atingir os objetivos citados. As informações da rede são redirecionadas para a estação base, fornecendo ao operador do experimento um panorama atualizado da situação de teste, seja em tempo real ou persistida para uma análise posterior.

Para o objetivo ser alcançado, uma aplicação IoT a ser testada que utilizar a FlexStation, deverá ser capaz de adaptá-la às suas demandas específicas. Uma característica presente na pluralidade de projetos envolvendo redes de dispositivos, é a troca de mensagens. Essas mensagens caracterizam o funcionamento de uma rede, sendo o alvo principal dos testes. Dessa forma, a estação base precisa fornecer meios para receber a informação. O protocolo de comunicação entre a estação base e os dispositivos da rede deverá ser implementado nos módulos que caracterizam o back-end da FlexStation.

Durante a implementação da FlexStation, viu-se a necessidade de inserir dois meios de comunicação entre a estação base e dispositivos externos (dispositivos IoT a serem testados). Sendo estes por requisições HTTP e com auxílio de um microcontrolador ESP32, conectado via Serial com a estação base. Esses métodos serão disponibilizados, assim como, por exemplo, a possibilidade de aceitar novos protocolos de conexão e troca de mensagens, desenvolvidos pelo usuário. Dessa maneira, a estação base se seria utilizável para uma variedade de projetos, com diferentes protocolos.

Uma ferramenta de teste de dispositivos IoT é a capacidade de persistir os dados sobre as mensagens trocadas na rede, para possibilitar uma posterior análise e depuração. Durante os experimentos, informações são geradas em tempo real, e isso dificulta uma análise imediata do que está acontecendo. Uma análise posterior pode ser realizada para determinar a razão de um erro ou a confirmação de uma hipótese sobre o funcionamento da rede de dispositivos, por exemplo. A FlexStation fornece meios de customizar a maneira que os dados são persistidos, quais eventos serão capturados e quando, assim como a estrutura dos arquivos gerados.

Para exibir qualquer informação que alcança a estação base, ocorre uma troca de mensagens interna, do módulo de conexões, responsável por receber as informações da rede, até a o módulo front-end. Essa comunicação é abstrata, independente do protocolo externo, e é feita através da troca de mensagens no formato JavaScript Object Notation (JSON). Então, independente do formato recebido, a mensagem deverá ser capaz de ser transformada para o formato JSON. Assim, existe a garantia de uma generalização na comunicação interna, sem a necessidade de maiores adaptações.

Algumas ferramentas são importantes para garantir fidelidade da informação recebida e exibida ao usuário, em tempo real. O módulo responsável por receber informações, dito back-end, repassa a mensagem em formato JSON para o módulo principal do front-end, escrito em linguagem Javascript. A conexão entre os módulos é efetuada a partir da tecnologia WebSocket, no qual uma seção dedicada full-duplex é criada. Múltiplas conexões WebSockets podem funcionar paralelamente, mantendo a garantia que não irá ocorrer bloqueio ou perda da informação.

Na interface é apresentado um mapa virtual, que ilustra o cenário de teste, com a localização dos dispositivos, seu tipo e seu status de atividade. O mapa presente na FlexStation utiliza a API do Google Maps, permitindo adicionar marcadores no mapa, atualizando a localização, a medida que a latitude e longitude dos dispositivos é passada para a estação base. A FlexStation armazena uma lista com os diferentes dispositivos que enviaram alguma informação, enquanto o servidor estiver rodando. Assim, é possível atualizar o mapa virtual da interface, periodicamente, com o status de atividade dos dispositivos ativos.

4
Atividades Preparatórias

Visando o desenvolvimento da FlexStation, foram estudadas as tecnologias e ferramentas necessárias. A princípio, o estudo se concentrou no mecanismo de desenvolvimento web, utilizando Python. Com a escolha do framework Django, para auxiliar o desenvolvimento, surgiu a necessidade de um aprofundamento nessa ferramenta. Foi estudada a estruturação de um projeto em Django, o funcionamento do servidor web Django, o mecanismo de renderização de templates (ou HTML) com Django, a criação de rotas e seu mapeamento para funções Python, chamadas de View, e toda configuração de projeto possível com o framework.

A estação base precisa fornecer um meio para receber a informação a ser analisada. Essa porta de comunicação foi um dos principais objetos de estudo, na busca de encontrar possibilidades de interação entre um dispositivo móvel e a estação base. O projeto GrADyS forneceu um papel importante nessa etapa, entregando conhecimentos e ferramentas necessárias pela busca de solução.

Uma opção de comunicação entre dispositivos da rede utilizada no projeto GrADyS, ocorreu através do uso de microcontroladores ESP32. Esse mecanismo de comunicação deverá ser suportado pela FlexStation. O microcontrolador é responsável pela lógica de receber/enviar mensagens para outros dispositivos, alvos dos testes. Portanto, seria apenas preciso que a estação base aceitasse uma conexão com um microcontrolador e como lidaria com as mensagens recebidas. Sendo assim, foi estudado o funcionamento de um microcontrolador ESP32, como estabelecer uma conexão via USB entre o microcontrolador e um microcomputador, através de uma interface UART com o PC, e como realizar a troca de mensagens.

Outro protocolo de comunicação, utilizado pelo projeto GrADyS, é na forma de requisições HTTP. Os dispositivos móveis, ou drones, servem como um servidor web, com IP próprio, capaz de realizar e aceitar requisições HTTP. Foi preciso adaptar a estrutura da arquitetura da FlexStation, já funcional para a comunicação externa via ESP32, para aceitar paralelamente mensagens via requisições HTTP. Os elementos responsáveis por repassar as mensagens recebidas, do back-end para o front-end, precisaram ser adaptados para o novo protocolo de comunicação externa, abrindo caminho para uma abstração

de comunicação interna da FlexStation. Ou seja, independente do protocolo externo, a FlexStation deverá manter seu funcionamento interno consistente.

A comunicação entre os módulos front-end e back-end da estação base utiliza a tecnologia WebSocket. A tecnologia precisa garantir que a comunicação seja extensível, permitindo múltiplas conexões simultâneas e independentes, ou seja, uma conexão funciona independente e sem afetar o funcionamento das outras e que não ocorra perda de nenhuma informação. O uso de conexões WebSocket satisfazem esses requisitos, uma vez possível a criação de múltiplas instâncias independentes de comunicação. Portanto, foi estudado como implementar as conexões, utilizando a biblioteca Django Channels e a API WebSocket nativa para Javascript, e se responderiam com as exigências propostas.

Um fator em comum, presente nas ferramentas similares apresentadas na seção "Situação Atual", é a visualização do experimento, em tempo real, com auxílio de um mapa virtual, informações, em texto, de envio e recebimento de mensagens e uma seção com conjuntos de botões, para envio de comandos. Seguindo como base os modelos de interface de ferramentas similares, a interface da FlexStation foi planejada.

Figura 4.1: Interface gráfica

A interface, como um todo, foi projetada e estudada para conter tudo que o experimento precisa passar ao operador do experimento. A informação precisa estar presente de forma clara, assim como as funcionalidades de controle acessíveis. Um mapa virtual, ocupando a maior parte da interface, exibe os dispositivos (UAV) em movimento, a troca de mensagens ou informações

pertinentes. O mapa utilizado é fornecido pela API do Google, integrado ao projeto com código Javascript. A interface também contém uma área com as últimas mensagens recebidas e uma coluna com os conjuntos de botões. Essa disposição de elementos na tela foi possível utilizando o sistema de layout CSS Grid, atribuindo a cada seção uma tamanho e localização na tela específico. Uma maior responsividade das seções pode ser possível atribuindo o redimensionamento de determinada seção com um botão, por exemplo.

A FlexStation utiliza uma biblioteca Python para auxiliar na persistência da informação, chamada logging. O funcionamento desse pacote foi estudado para tornar possível a criação de uma classe utilitária configurável, encarregada de salvar as informações que chegam a FlexStation em um arquivo tipo log.

5
Especificações do Projeto

5.1
Arquitetura

O projeto foi desenvolvido utilizando um computador portátil (notebook) com sistema operacional Windows 10 e o Subsistema Windows para Linux (WSL2), para portabilidade para sistemas Linux (Ubuntu LTS 20.04). Por ser uma aplicação web, foram utilizados três navegadores de rede para desenvolvimento e testes, sendo eles, o Google Chrome, o Mozilla Firefox e o Microsoft Edge.

A arquitetura da FlexStation foi pensada para torná-la adaptável e customizável a diferentes necessidades. As funcionalidades estão modularizadas, auxiliando na customização e inserção de novos elementos, como botões de comando e protocolos de comunicação com dispositivos IoT.

A primeira escolha analisada foram as linguagens de programação a serem utilizadas. Como o usuário terá contato direto com o código da FlexStation, para uma maior adaptação da estação base ao seu projeto, a linguagem de programação escolhida representa um importante fator. A inserção de um protocolo de comunicação externa, a inserção de novos botões de comando ou a manipulação da classe utilitária de persistência da informação são alguns exemplos no qual o usuário deverá realizar para configurar a estação base antes dos testes de campo.

O módulo back-end do projeto foi construído utilizando a linguagem de programação Python, em conjunto do uso do Django framework, uma ferramenta escrita em Python, para o desenvolvimento ágil de uma aplicação web. Esse framework fornece um servidor de desenvolvimento de fácil configuração, um mapeamento de rotas URL simples e extensível e uma arquitetura altamente modular.

O módulo front-end foi construído utilizando Linguagem de Marcação de Hipertexto (HTML), a linguagem de estilização Cascade Style Sheets (CSS) e a linguagem de programação mais popular em uso, Javascript.

As tecnologias citadas foram escolhidas em virtude ao seu uso amplo e popularidade, contribuindo para um framework com usabilidade mais extensível.

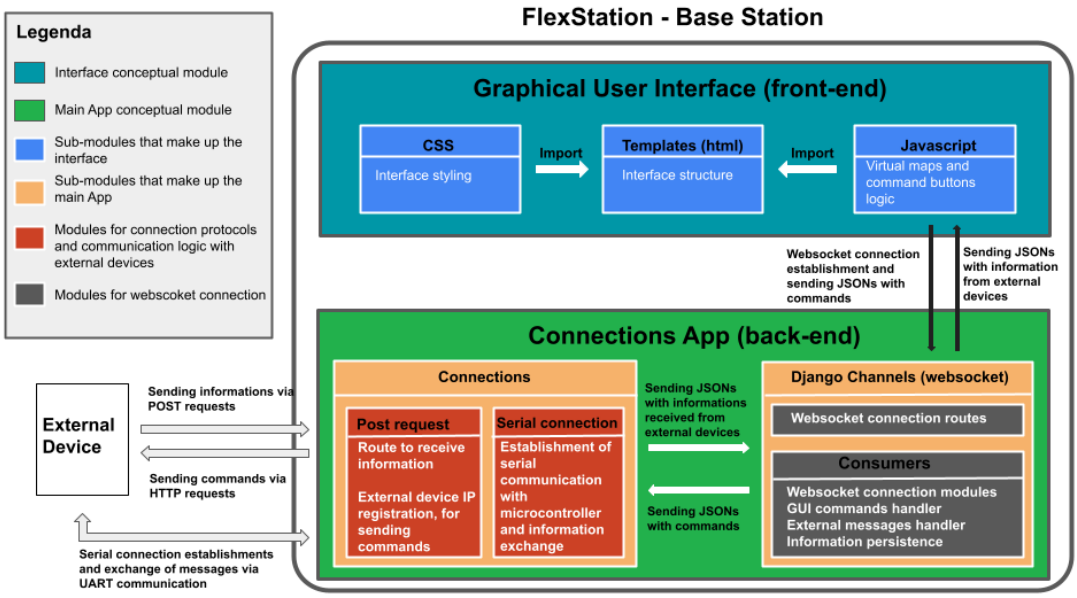

Figura 5.1: Representação da arquitetura da FlexStation

As linguagens de programação Python e Javascript também contam com um conjunto vasto de bibliotecas e pacotes, reflexo de uma comunidade extensa.

A arquitetura da FlexStation foi pensada e estruturada seguindo o conceito clássico de desenvolvimento web, com o módulo front-end responsável pela interface e visualização e o módulo back-end responsável pelo processamento de informação server-side e pela comunicação com dispositivos externos.

Como é apresentado na Figura 5.1, dispositivos externos podem enviar mensagens para a FlexStation através do submódulo Connections. O processamento da informação é feito no submódulo Django Channels, também responsável por repassar a informação para a interface, através de uma conexão WebSocket já estabelecida. As mensagens trocadas entre os módulos front-end e back-end seguem o formato JSON. Note que o caminho descrito, do dispositivo externo até a interface, também é possível no sentido inverso, quando um comando for ativado na interface.

5.2 Módulos

O front-end contém a interface principal, constituída por arquivos HTML e CSS. Os arquivos HTML foram escritos utilizando a linguagem Template, introduzida pelo Django. Com a linguagem Template é possível criar arquivos gerados em formato HTML, incluindo funcionalidades não nativas em HTML. Com a linguagem Template é possível utilizar variáveis, que são substituídas com valores quando o template é processado, e uso de tags. As tags permitem uso de operadores lógicos, laços de repetição, inserção de outros documentos

HTML, entre outras funcionalidades que fornecem ao desenvolvedor uma maior flexibilidade de desenvolvimento. A lógica da interface está presente nos arquivos codificados em linguagem de programação Javascript.

Os arquivos Javascript são responsáveis pelo funcionamento e renderização do mapa virtual, determinando como é realizada a inserção de marcadores no mapa, atualização ou remoção destes, assim como uma maior customização dos marcadores e do mapa em si. O mapa é renderizado assim que o arquivo Javascript é inicializado, iniciando com um zoom específico e uma localização pré-configurada pelo usuário. Os marcadores são renderizados a partir de mensagens que chegam até o arquivo Javascript, normalmente contendo a latitude e longitude, entre outras informações, do dispositivo. Cada dispositivo contém um identificador único, que é utilizado para criar uma lista de dispositivos. Essa lista fornece a possibilidade de direcionar comandos para dispositivos específicos, como por exemplo, retornar para o ponto de origem do experimento. A lista de opções disponíveis, contendo os dispositivos ativos, é mantida pelo arquivo Javascript, podendo ser usada para envio de comandos direcionados.

Os arquivos Javascript também são responsáveis pelo funcionamento dos botões de comando, presentes na interface. Uma vez o botão inserido no corpo do arquivo template, o arquivo Javascript pode registrar o seu comportamento ao ser ativado. Foi criado um protocolo relacionando um número inteiro com o comando em si. Portanto, cada botão está relacionado a um número que, ao ser enviado para o back-end, ocorre o mapeamento do número ao processamento da tarefa específica.

Outro componente principal da arquitetura é o módulo back-end, que fica encarregado da lógica de conexão e troca de mensagens com dispositivos externos, do processamento das informações recebidas e do envio para o front-end.

Para entender como foi estruturado e o seu funcionamento, primeiro é indispensável apresentar como o Django é estruturado e como ele opera.

O framework Django fornece um servidor web leve, para desenvolvimento, que pode ser usado via o arquivo manage.py. Por padrão, o servidor roda na porta 8000, no endereço de IP 127.0.0.1, o servidor local. Ao iniciar o servidor, e o navegador for acessado na URL, composta pelo endereço de IP com a porta onde o servidor está rodando, o Django irá chamar o método Python correspondente. Nesse caso, será chamado o método responsável por renderizar o arquivo template com o código da interface.

Criar um esquema URL com o Django é uma tarefa simples, graças ao mapeamento URL/View que o Django fornece. Quando um usuário acessa uma página presente no esquema URL, o Django faz um mapeamento para

uma função Python correspondente, chamada de View. Dentro do módulo back-end há o arquivo urls.py, responsável por associar os endereços de URL com as Views. Por sua vez, o arquivo views.py é responsável pela lógica de uma determinada View.

Uma View é uma função Python que recebe uma requisição web e retorna uma resposta. Essa resposta pode ser um conteúdo de uma página web, ou um JSON, ou qualquer outra resposta relevante. O corpo da função contém a lógica necessária para aquela View. Dessa forma, esse esquema URL, fornecido pelo Django, possibilita criar uma rota para receber requisições do tipo POST. Um dispositivo externo é capaz de realizar uma requisição, enviando no corpo da mensagem a informação útil, e a View correspondente é responsável pela lógica de como lidar com a mensagem. É possível persistir o conteúdo recebido, lidar com erros, retornar uma mensagem do tipo acknowledge, registrar o dispositivo em uma lista permanente para a sessão, formatar um JSON com as informações recebidas e redirecioná-lo para o módulo front-end. Qualquer lógica extra que uma aplicação necessite para essa etapa, pode ser incluído na View correspondente.

Outra função principal do módulo back-end é a persistência de todo evento ocorrido durante o experimento. Um arquivo log é gerado, quando a aplicação é iniciada, sendo nomeado pela data e hora da criação. Estes arquivos são preenchidos à medida que mensagens são recebidas, erros são apanhados, comandos são enviados e outros eventos de importância para o experimento.

Para gerar os arquivos log, o pacote 'logging', para Python, é utilizado. A classe utilitária Logger foi criada para conter toda a lógica de persistência de dados, podendo estender e copiar essa classe para outros módulos, modificando a lógica caso necessário. Para preencher o arquivo log, deve ser inserido no código chamadas para os métodos da classe Logger, de acordo com o evento. Por exemplo, o método público `log_info(data)`, irá registrar os dados enviados, assim como o método `except()` irá salvar uma exceção capturada. O formato de escrita no arquivo log é especificado dentro da classe Logger, usando a sintaxe aceita pela classe Formatting, do pacote logging.

Ambos módulos estão contidos em um mesmo diretório de projeto, em razão do funcionamento do Django. Como dito, o Django é encarregado de renderizar o template, contido no diretório templates, assim como os arquivos da pasta static, incluindo arquivos Javascript e CSS. A separação dos dois módulos, então, se configura apenas conceitualmente.

5.3 Comunicação entre módulos

Os dois módulos principais se comunicam, trocando mensagens JSON, via canais WebSocket. Uma conexão WebSocket é um canal dedicado full-duplex, baseado no Protocolo de controle de transmissão (TCP).

Esse projeto utiliza a biblioteca Django Channels para lidar com a comunicação WebSocket, no lado back-end. O Django Channels realiza um mapeamento similar ao mencionado esquema URL/View. A rota de uma conexão socket é inserida no arquivo routing.py, onde é fornecido o mapeamento para uma classe Consumer. Essa classe Python herdará da classe WebsocketConsumer, presente na biblioteca Django Channels, herdando assim os métodos necessários para aceitar uma conexão, enviar e receber através do canal estabelecido. Dessa maneira, as classes Consumer irão conter a lógica para lidar com a conexão do lado back-end.

O submódulo, contendo arquivos Javascript, inicia uma conexão socket com uma das rotas presentes dentro do submódulo Django Channels, assim que inicializados. Nativamente, a linguagem de programação Javascript apresenta uma API para a classe WebSocket, sendo suportado pela maioria dos navegadores modernos [https://datatracker.ietf.org/doc/html/rfc6455*]. A API WebSocket fornece meios para enviar mensagem via o canal estabelecido, consultar o status da conexão e reagir a uma mensagem recebida pelo socket.

5.4 Comunicação com dispositivos externos

A principal funcionalidade da FlexStation é a capacidade de trocar informações com outros dispositivos, os alvos dos testes. O portão de informação da estação base para dispositivos externos é através do submódulo Connections, que contém as rotas e a lógica para receber e enviar informação. Atualmente, estão implementados dois modos de conexão externa.

O primeiro modo é conectando um microcontrolador ESP32 na máquina da estação base. Esse microcontrolador deve ser capaz de detectar outros dispositivos, responsáveis por receber e enviar informações da rede. A FlexStation consegue estabelecer uma conexão UART com o microcontrolador ESP32, receber e enviar informações via serial, deixando o microcontrolador responsável para retransmitir o comando. Para aceitar a conexão com o microcontrolador, é necessário inserir a porta UART correta, assim como a velocidade de transmissão de informação da conexão (baud rate) no arquivo `config.ini`. A Classe encarregada de iniciar uma conexão com um microcontrolador é instanciada assim que o Javascript inicia a conexão WebSocket. O objeto instanciado

mantém uma rotina de verificação da porta UART. Uma vez que o microcontrolador é conectado, a interface indica esse evento, e a troca de informações se trona possível pelo microcontrolador ESP32.

Outra maneira de estabelecer uma comunicação entre a estação base e dispositivos externos é através de requisições do tipo POST. Um dispositivo, um UAV (drone) per se, deseja enviar a sua localização para a estação base. Esse objetivo pode ser alcançado com uma requisição POST para o endereço URL específico determinado pelo esquema URL do Django. Note que poderá ser anexado, no corpo da mensagem, o endereço IP e a porta do próprio dispositivo externo, caso tenha esse comportamento de comunicação. Dessa forma, a FlexStation é capaz de registrar o dispositivo com seu endereço único, podendo enviar comandos via requisições HTTP.

As classes Consumer são responsáveis por enviar uma mensagem para dispositivos externos. Quando um botão de comando é ativado na interface, o arquivo Jasvacript utiliza o método socket.send() para transmitir o comando diretamente para o Consumer (back-end). A mensagem recebida, é um JSON e contém qual dispositivo ou grupo de dispositivos deverá receber a mensagem, baseado em identificadores únicos. Além disso, contém o comando em si, que é um número inteiro. O primeiro passo é procurar na lista de dispositivos registrados, pelo endereço (IP e porta) a ser enviado, caso o protocolo de comunicação externa seja requisições POST. Com o endereço encontrado, o comando deve ser executado. Para o caso de requisições POST, além do endereço, deverá incluído um API endpoint, que o dispositivo irá interpretar internamente. Existe uma lista, no arquivo `config.ini`, onde ocorre o mapeamento de um código de comando para um API endpoint específico, que deverá ser adicionado ao endereço IP. Com o endereço completo, o back-end envia o comando através de uma requisição HTTP.

O diagrama de sequência, apresentado na Figura 5.2 detalha o fluxo de mensagens entre dispositivos externos e a FlexStation, assim como o fluxo de mensagens entre os módulos back-end e front-end da FlexStation. É detalhado graficamente o tipo de comunicação, com cada mensagem podendo ser síncrona ou assíncrona. É apresentado os seguintes fluxos de comunicação:

- Envio de localização de um dispositivo IoT externo para o submódulo "Comunicação Externa". O dispositivo é registrado em uma lista permanente por sessão, contendo seu endereço IP. A mensagem é salva no arquivo log. E a informação é encaminhado até o submódulo "Interface", exibindo-a na tela.
- Envio de um comando a partir do submódulo da interface, para um dispositivo IoT externo.

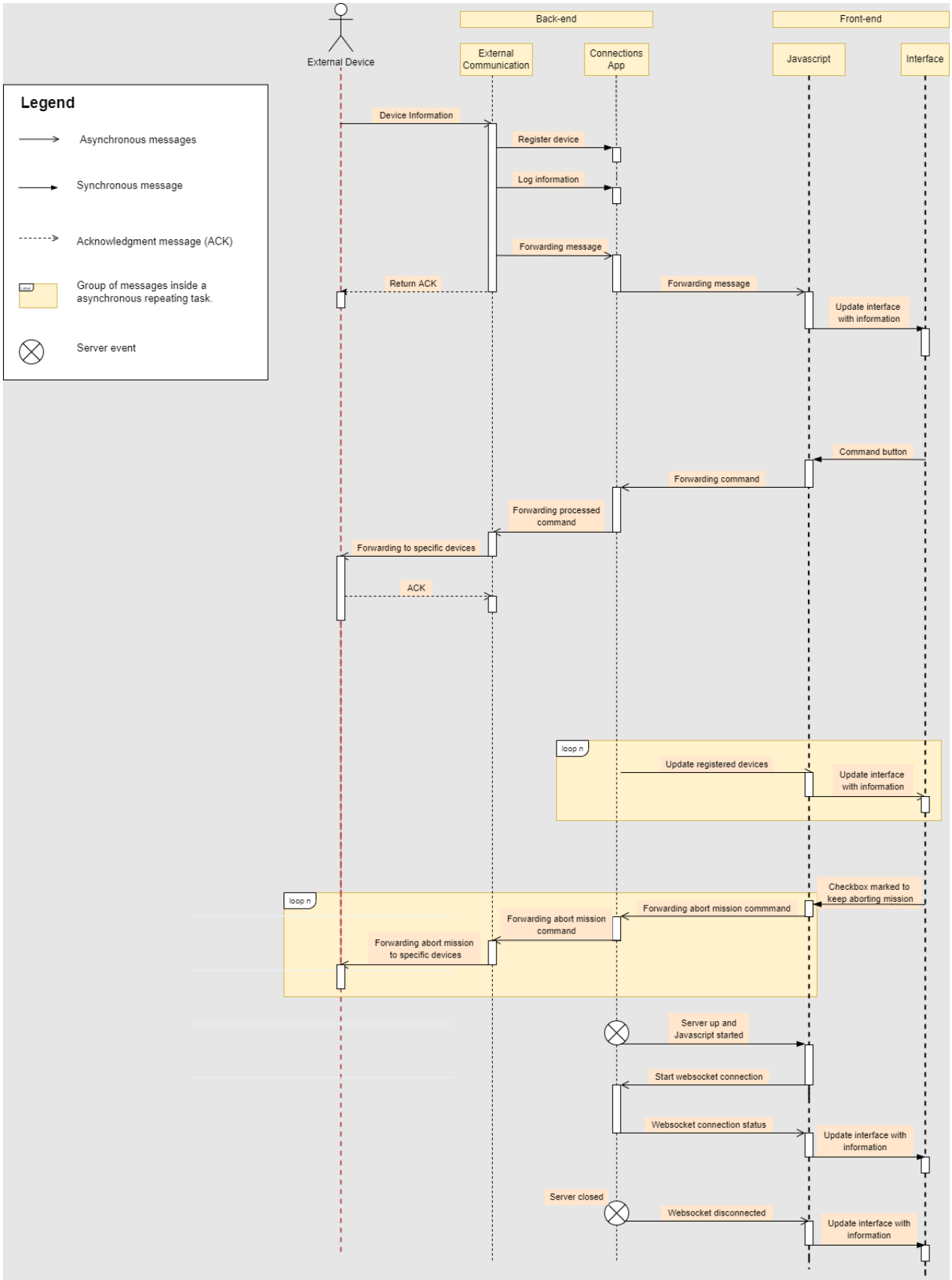

Figura 5.2: Diagrama de sequência

- Atualização periódica, a partir de um Consumer do submódulo "App Conexões". A lista permanente de dispositivos registrados é encaminhada para o submódulo "Interface", indicando o status de atividade de cada dispositivo. O intervalo de cada atualização é especificado no arquivo config.ini, sendo fornecido em segundos.
- O funcionamento de envio de comandos utilizando um botão do tipo "checkbox", no qual uma tarefa é criada no módulo back-end e fica sendo repetida até o botão ser desmarcado.
- A inicialização do servidor, resultando no estabelecimento das conexões WebSocket presentes no submódulo "Javascript". Assim como o encerramento do servidor, cancelando as conexões WebSocket ativas.

Caso seja necessário inserir uma comunicação customizada, a FlexStation fornece uma modularização do back-end e a fácil interação com o front-end. A abstração dos componentes, que caracterizam o back-end, encarregados de realizar as conexões com dispositivos externos e da lógica na troca de mensagens com dispositivos externos, é essencial para inserção de novos protocolos. Múltiplos protocolos podem funcionar simultaneamente, por conta de conexões WebSocket paralelas entre o back-end e a interface.

A proposta de tornar a estação base para testes de campo extensível, para diferentes projetos, cria desafios de modelagem e implementação. A implementação de toda a lógica para novos protocolos fica a cargo dos usuários que a forem utilizar. O desafio é tornar possível que o framework se adapte a maioria dos projetos do contexto alvo, enquanto fornece todos os meios necessários, sem perda de usabilidade, segurança e confiabilidade.

5.5 Motivações

As escolhas adotadas para a formação da arquitetura da FlexStation foram motivadas pelo estudo prévio de ferramentas similares, pelas próprias necessidades do projeto GrADyS e por experimentação durante o desenvolvimento.

Os estudos iniciais determinaram o tipo de aparelho alvo na qual a estação base poderia funcionar, podendo ser smartphones, tablets ou microcomputadores. No primeiro momento, foi considerado e testado utilização do framework Flutter, baseado na linguagem de programação Dart. O foco seria o desenvolvimento para aparelhos com o sistema operacional Android, especificamente para smartphones. A opção de fornecer os serviços da FlexStation na forma de aplicação web, surgiu da preferência pela linguagem de programação Python. A utilização de Python fornece uma melhor usabilidade para os

usuários do projeto, pela sua grande popularidade e extensão de bibliotecas. Além disso, a linguagem é a mesma utilizada pela tecnologia dos drones do projeto GrADyS e no geral. Essa similaridade fornece uma facilidade na integração. Com a linguagem de programação escolhida e o produto final sendo uma aplicação web, a escolha das outras tecnologias foram naturais. A escolha do framework Django foi consequência das facilidades fornecidas por este, para desenvolvimento do projeto, estruturação de código e acesso a um servidor web.

A interface foi construída com as ferramentas de desenvolvimento web clássicas, HTML, CSS e Javascript. Foi considerado utilizar uma biblioteca Javascript, como por exemplo React.js e Angular, para auxiliar na construção do módulo front-end. Como o projeto não apresentou um perfil de ser altamente escalável, não foi considerado que uma biblioteca Javascript seria útil. Além disso, iria fornecer um obstáculo extra para utilização do framework pelos usuários.

Durante a fase de implementação, surgiu a motivação pela escolha de comunicação interna utilizando a tecnologia WebSocket. Por conta da constante interação entre os módulos front-end e back-end, optou-se pelas facilidades oferecidas pelas conexões WebSocket. A troca de mensagens entre módulos é bi-direcional, ou seja, tanto mensagens precisam ser enviadas do back-end para o front-end, como a atualização de localização de um drone, por exemplo, como mensagens precisam ser enviadas do front-end para o back-end, como um comando disparado na interface. A conexão WebSocket permite essa comunicação full-duplex, sem riscos de colisão ou travamento do fluxo do programa. Foi considerada a criação e utilização de uma API RESTful, capaz de, a partir de um protocolo estabelecido, possibilitar a troca de mensagens entre módulos. Esse mecanismo de comunicação não se mostrou preferível para a FlexStation, devido, principalmente, a dificuldade de comunicação bi-lateral. Seria necessário criar o protocolo API REST para o sentido de mensagem front-end para back-end, diferindo, porém do mecanismo de comunicação sentido contrário.

5.6 Uso

A FlexStation está disponível para uso livre, na plataforma do Github, através do link https://github.com/BrenoFischer/gradys-gs.

Para apoiar o usuário da FlexStation, foi estruturada uma documentação e incluída no repositório do Github. A documentação explicita os pré-requisitos necessários, como instalar, como configurar e modificar localmente e apresenta os módulos e funcionamento geral da estação base.

Com o objetivo de utilizar a FlexStation, o usuário deverá ter instalado na máquina da estação base, o Python na versão 3.0 ou superior. Também, pip, um gerenciador de pacotes Python, é recomendado para gerenciar e automaticamente instalar pacotes necessários do projeto.

Com os pré-requisitos instalados, o projeto deverá ser clonado do repositório do Github. Visando manter o framework em uma ambiente separado, com seus pacotes próprios e versões específicas, é recomendado criar um ambiente virtual. O recomendado é utilizar o módulo para criar ambientes virtuais leves para Python, venv. A lista de pacotes necessários estão contidos no arquivo requeriments.txt, caso o sistema operacional seja Windows, e requerimentsz_linux.txt, caso o sistema operacional seja Linux. Os pacotes serão instalados pelo pip, que irá utilizar os arquivos texto, e instalar no ambiente virtual ativo.

O usuário conta com arquivos de configuração, para uma rápida configuração de parâmetros. Além de fornecer uma parametrização acessível e prática, auxilia na manutenibilidade do projeto, por criar um ambiente único de armazenamento da informação e de parâmetros utilizados em diferentes áreas do código. O principal arquivos de configuração, é o arquivo config.ini. Nele, é possível ajustar a frequência de atualização da interface, o IP padrão da estação base, o protocolo dos comandos do método de comunicação por requisições POST, ou qualquer parâmetro geral utilizado pela estação base.

O usuário alvo é incentivado a incluir funcionalidades necessárias, antes de qualquer experimento. Em razão do projeto ser Open Source, todos estão aptos a baixar o código fonte, modificar e criar novas funcionalidades. A arquitetura do projeto foi organizada com extensão em mente, especialmente para facilitar a inserção de novas formas de comunicação externa. Também é possível uma maior configuração dos botões de comando e incluir conjuntos novos de botões, caso necessário.

É possível registrar um novo botão, para enviar algum comando importante durante o experimento. Dentro do template, arquivo HTML, o novo botão poderá ser adicionado, modificando a estrutura da interface. O botão a ser inserido no HTML é um elemento <input> com seu parâmetro type sendo "button"ou "checkbox", dependendo do tipo de botão. O botão tipo "button"envia um comando disparado uma única vez, já o botão tipo "checkbox"envia comandos enquanto estiver marcado. Esse segundo tipo de botão, cria uma tarefa no back-end, que se manterá ativa até a informação que o botão foi desmarcado, cancelando, assim, a tarefa. Para incluir a lógica do botão, uma função callback onClick deverá estar contida no arquivo Javascript. Finalmente, o comando pode ser enviado em uma conexão WebSocket para

o back-end. Por fim, a lógica de tratamento do comando poderá ser feito no Consumer. Para o protocolo de comunicação de requisições POST, existe uma lista mapeando o código enviado pelo front-end para uma API endpoint. Caso o usuário esteja utilizando esse protocolo de comunicação, um novo mapeamento deverá ser incluído na lista.

Com o objetivo de adicionar um novo protocolo de comunicação externa, o usuário pode incluí-lo no módulo back-end. A lógica e código da nova comunicação fica de acordo com o desenvolvedor usuário. Como o módulo de comunicação da estação base foi construído utilizando Django, o novo código do protocolo deverá ser escrito usando a linguagem de programação Python. Um novo Consumer poderá ser adicionado, assim como um novo endereço de conexão, que serão responsáveis pela comunicação interna, enviando mensagens via conexão WebSocket, para o front-end. Como a mensagem irá alcançar o novo Consumer, também fica a cargo do desenvolvedor.

A interface contém um conjunto de botões responsáveis para carregar um arquivo. Um arquivo texto, por exemplo, pode conter um plano de voo, com as coordenadas e instruções que um drone deve seguir. Ao carregar um arquivo da máquina local, o front-end realiza uma requisição HTTP, direcionando para o dispositivo em questão. Para utilizar essa funcionalidade, portanto, o dispositivo a receber o arquivo deverá aceitar essa requisição e possuir um endereço IP válido.

Um vídeo apresentando o processo de instalação e uma demonstração de uso está disponibilizado na plataforma Youtube, através do link: https://www.youtube.com/watch?v=-4Jt4k6ftaAab_channel=BrunoOlivieri

5.6.1 Composição e uso da interface

A interface é separada em três seções principais: Mapa virtual, Botões de comando e Registro de mensagens recebidas.

Seção Mapa Virtual: O mapa virtual da Google ocupa a maior proporção da tela. É um mapa interativo, ajustável, capaz de fácil modificação da aproximação (zoom). Note que uma conexão com a internet é necessária para carregar o mapa, após a renderização da página web. Os marcadores, representando dispositivos externos, são inseridos e atualizados automaticamente no mapa, a partir de mensagens JSON que chegam do back-end para os arquivos Javascript, responsáveis pela lógica do mapa. Cada marcador possui um texto indicando o tipo de dispositivo e seu ID único. A cor do marcador indica o status de atividade, podendo ser verde, ativo; amarelo, atividade recente; vermelho, inativo. O tempo que determina cada estado pode ser configurado

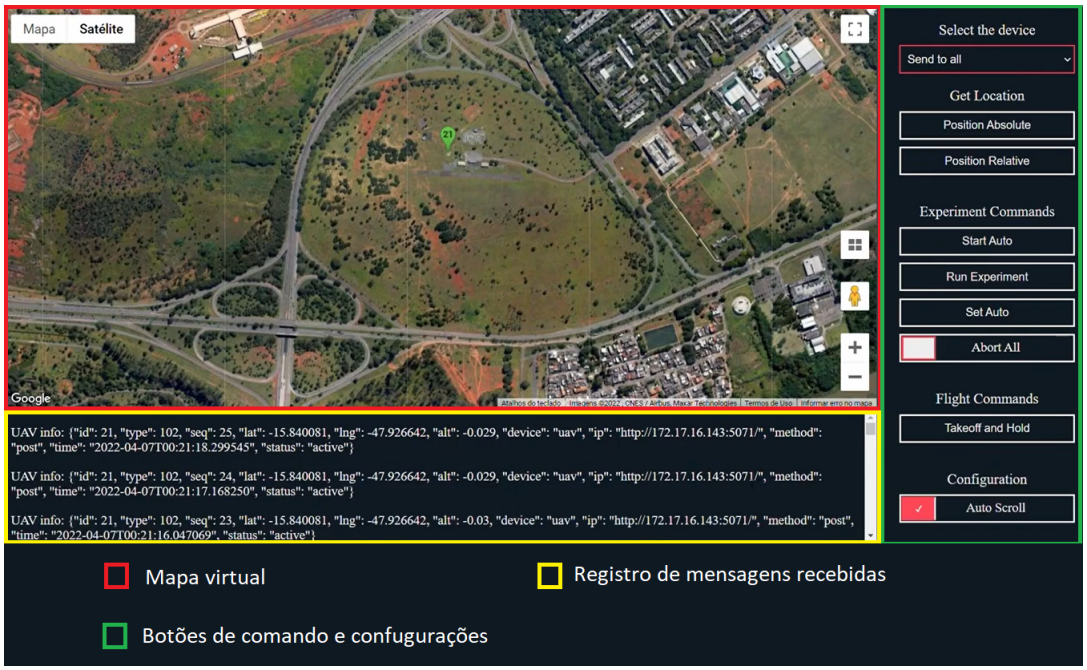

Figura 5.3: Diagrama de sequência

no arquivo config.ini, e determina o intervalo de tempo que um determinado dispositivo enviou informações para a estação base.

Seção botões de comando: A seção de botões de comando contém um campo de seleção, capaz de selecionar um grupo de dispositivos ou um dispositivo específico, já registrado. Todos os comandos enviados serão apenas para a opção selecionada. Essa seção também contém subgrupos de botões, com um título e os botões em si. Um botão pode ser do tipo checkbox ou simples, diferindo seus layouts. Na Figura 5.3 estão representados botões inseridos para auxiliar testes do projeto GrADyS. Os primeiros dois botões da seção "Get Location"enviam comandos para obter a latitude, longitude e altitude dos drones. A diferença entre os dois botões é no parâmetro altitude, onde o primeiro comando informa a altitude relativa ao nível do mar (MSL) e o segundo em relação à posição inicial de decolagem. Os três primeiros botões do conjunto seguinte auxiliam no controle de experimentos. Esses botões disparam uma série de comandos, decolando o drone, direcionando o voo para pontos pré-determinados e retornando à posição inicial após a visita completa dos pontos. A diferença principal entre os botões desse grupo é no conjunto de pontos a serem visitados, mudando latitude, longitude ou altitude. O quarto botão do conjunto "Experiment Commands"é um botão do tipo checkbox. Enquanto esse botão estiver pressionado, todos os drones serão ordenados a voltar para a posição incial de decolagem. O próximo conjunto "Flight Commands"contém um único botão, para ordenar a decolagem até determinada altura. Por fim, há a seção "Configuration", com botões para modificar alguma configuração

da interface, como o rolamento automático da seção em amarelo (Registro de mensagens recebidas).

Seção registro de mensagens: A seção de registro de mensagens mostra todas as mensagens recebidas pela estação base, em cor branca e todas mensagens enviadas pela estação base, em texto vermelho. Na Figura 5.3 estão representadas mensagens enviadas por um drone, contendo as seguintes informações: um identificador único para esse drone, o tipo da mensagem, na Figura 5.3 o tipo é "102"indicando que é uma mensagem do tipo "informação de posição", um sequencial, para auxiliar na depuração das mensagens, a altitude, longitude e latitude, o tipo de dispositivo, nesse caso sendo "uav", o campo para o endereço IP do drone, o método que foi enviado essa mensagem, indicando na figura que foi o méotdo POST Request, a hora que a mensagem foi recebida pela FlexStation e o status de atividade do drone.

A interface pode ser modificada para outras formas de controle e monitoramento. As seções de botões na área verde da Figura 5.3 podem ser customizadas. Cada título pode ser modificado, assim como os botões que pertencem àquela seção. Novos botões podem ser adicionados e o texto dentro de cada botão customizado. O estilo das mensagens da seção amarela da Figura 5.3 também podem ser customizados. A cor, tamanho da fonte e como serão exibidas podem ser modificadas para outras formas de monitoramento. Essas modificações são possíveis de serem feitas alterando o arquivo Template (HTML), Javascript e CSS. Maiores instruções estão disponíveis no repositório do Github

6
Implementação e Avaliação

A avaliação do sistema foi realizada em cada etapa do desenvolvimento. As avaliações auxiliaram para efetivar um conceito, uma ferramenta ou uma funcionalidade, assim como para perceber algum mal funcionamento ou inconsistência do projeto.

A primeira etapa se configurou no planejamento da arquitetura, linguagens a serem utilizadas e ferramentas que seriam incluídas. Para essa primeira fase, foi avaliado, principalmente, o funcionamento da ferramenta Django e da biblioteca Django Channels. O framework Django ilustrou um cenário de facilidades para o planejamento e implementação da arquitetura estruturada e o Django Channels forneceu a possibilidade de conexões WebSockets. Foi testada a capacidade de múltiplas conexões, simultâneas, e a consistência da informação trocada entre os módulos. Por análise de desempenho, foram incluídos Consumers capazes de enviar informação periodicamente. No outro lado, do front-end, foram iniciadas múltiplas conexões simultâneas, disparando os objetos Consumer. O fluxo da informação se mostrou satisfatório e, principalmente, não ocorreu o bloqueio do fluxo de código. A arquitetura interna foi planejada para funcionar com eventos assíncronos, utilizando os recursos necessários das linguagens de programação e bibliotecas.

A introdução de novas formas de comunicação entre a FlexStation e dispositivos IoT, também necessitava de uma avaliação de sua generalidade. O funcionamento da FlexStation deveria se manter consistente, independente do protocolo de comunicação com dispositivos externos. Dois protocolos foram implementados, sendo estes por requisição HTTP e utilizando microcontroladores ESP32. Os testes revelaram modificações pontuais para manter a comunicação entre os módulos front-end e back-end abstrata. Toda informação que chega no módulo back-end precisa ser convertida para um padrão, em JSON, para ser aceita pelo front-end.

Como experimentos de campo demandam um esforço grande, seja no preparo cuidadoso, na realização do experimento de fato, nas condições meteorológicas adequadas e na necessidade de equipamentos, foi implementado um módulo simulador dentro da FlexStation. O módulo uav_sim utiliza o simulador Software in the Loop (SITL)(15), possibilitando rodar o ArduPilot (16)

diretamente na máquina local da estação base. Por sua vez, o ArduPilot é o principal sistema de piloto automático de código aberto, que suporta uma variedade de veículos, incluindo multicópteros (drones), helicópteros tradicionais, rovers, submarinos, entre outros. É uma ferramenta versátil, com recursos para apoiar um controle de voos numerosos.

O SITL fornece a capacidade de simular o funcionamento de drones de forma fiel, uma vez que é executada a mesma tecnologia que um drone real utiliza, porém sem a necessidade de um hardware especial. Cada drone foi implementado para também funcionar como um servidor web, com endereço IP e porta própria. O servidor do drone foi escrito com a linguagem Python e a ferramenta Flask. Assim, a FlexStation pôde receber requisições POST com as informações simuladas dos drones. A FlexStation roda o servidor Django na rede local com IP 127.0.0.1 e uma porta específica. O endereço da requisição POST com as informações do drone é constituído, então, pelo IP do servidor Django, pela a porta que está rodando o servidor e uma rota específica para receber informações de drones, definido no esquema de rotas Django. O servidor do drone também permitiu enviar e testar comandos disparados da interface, sendo direcionados para o endereço IP de um drone ou conjunto de drones.

O primeiro teste com o simulador foi iniciar múltiplos servidores Flask, cada um representando um drone diferente, que por sua vez inicializa um SITL. Os servidores Flask possuem rotas internas para alguma tarefa a ser simulada, como por exemplo enviar a latitude e longitude do drone atualizadas. O esquema adotado é semelhante ao utilizado pelos drones do projeto GrADyS.

A FlexStation respondeu satisfatoriamente às múltiplas requisições dos drones, enviando as suas localizações periodicamente. O teste foi incremental, sendo testado com dois, três, cinco e dez drones, enviando informações a cada dez, cinco, um e meio segundo.

Com múltiplos drones ativos, foi possível testar o envio de comandos para drones específicos, a partir de um seletor na interface. No módulo back-end é verificado o identificador único do drone escolhido, na lista de dispositivos registrados, e direcionado o comando apenas para o endereço IP e porta do drone escolhido.

Um dos botões de comando criados foi para testar uma missão simples: decolar o drone, obter uma certa altitude de voo, visitar cinco pontos próximos e retornar para o ponto de início. O sucesso da missão foi possível de ser acompanhado pelo mapa virtual, assim como pelas informações enviadas pelos drones e persistidas em arquivos log.

Outro botão implementado é responsável para enviar o comando de

retornar um drone para o ponto de origem. Independente da sua posição atual, o drone deverá retornar em linha reta até o ponto que decolou. Esse comando foi testado em paralelo a missão em andamento, introduzida pelo botão mencionado anteriormente.

Uma missão denominada Espiral foi implementada, sendo disparada por um outro botão na interface. A ideia da missão Espiral é comparar o voo de um drone no simulador com o voo em campo. A missão envolve decolar o drone e fazê-lo visitar pontos específicos em espiral crescente. Cada ponto do espiral contém uma latitude, longitude e altitude específica. A altitude dos pontos varia de dez metros até sessenta metros, crescendo linearmente até metade e decrescendo para o restante de pontos. Foram testados fatores de campo, como a influência do vento durante o voo em espiral, e as diferenças obtidas utilizando o SITL. Esse cenário ilustra um exemplo possível de uso da FlexStation para análises posteriores, a partir dos resultados persistidos durante um teste em campo.

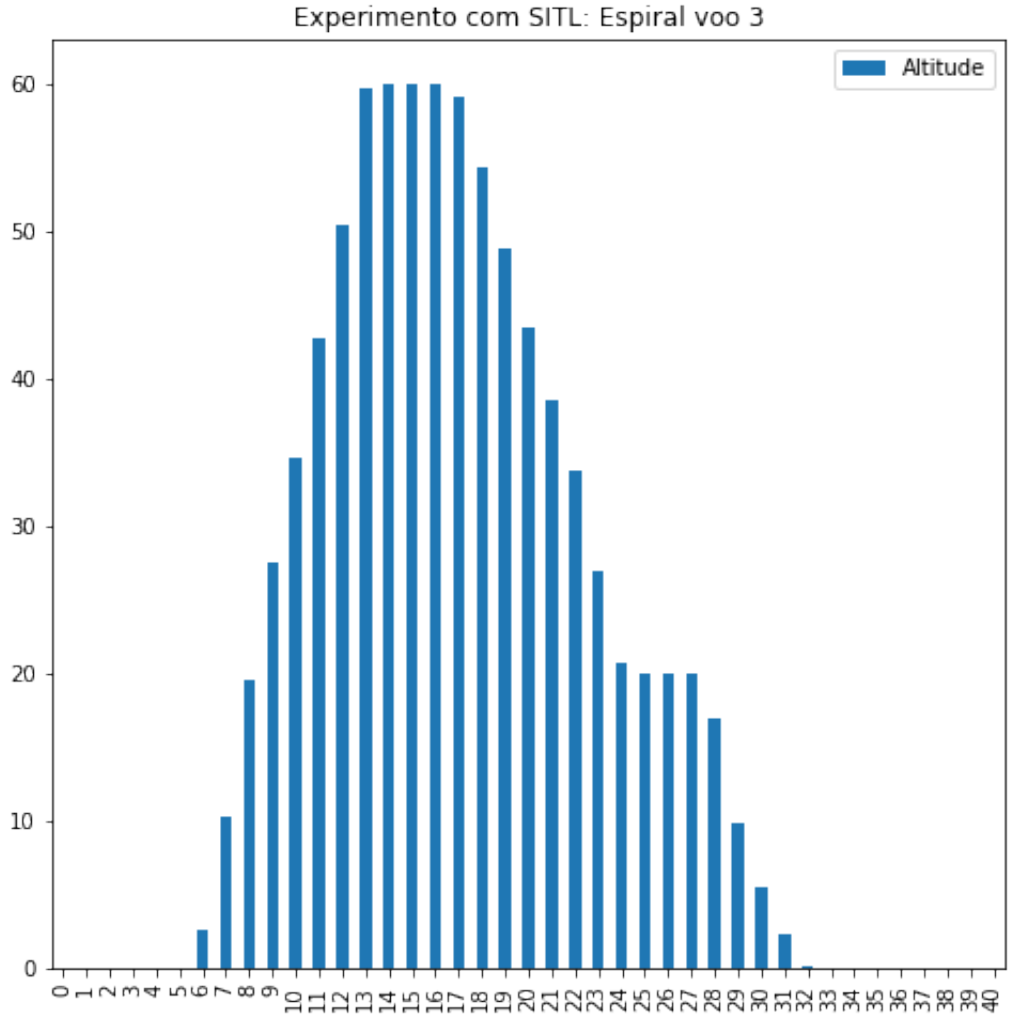

Figura 6.1: Variação de altitude de voo simulado ao longo da missão Espiral

A Figura 6.1 foi produzida após a realização de um voo simulado para os pontos da missão Espiral. O voo foi feito com o auxílio do SITL, e estão representadas as altitudes de voo ao longo do experimento. Durante a missão Espiral o drone enviou informações, contendo a altitude atual, em intervalos de um segundo. Cada informação recebida foi persistida em um arquivo log de sessão. Após a missão finalizada, as altitudes foram extraídas do arquivo log gerado, inseridas em uma estrutura de dados e utilizadas para montar o gráfico da Figura 6.1, com auxílio da biblioteca Pandas (17).

Foram realizados cinco testes como prova de conceito, cada teste representando um voo simulado pelo SITL conectado com a FlexStation. Cada teste gerou um conjunto de altitudes, semelhantes ao conjunto apresentado na Figura 6.1. Posteriormente, esses conjuntos de altitudes foram comparados servindo como uma verificação de conceito.

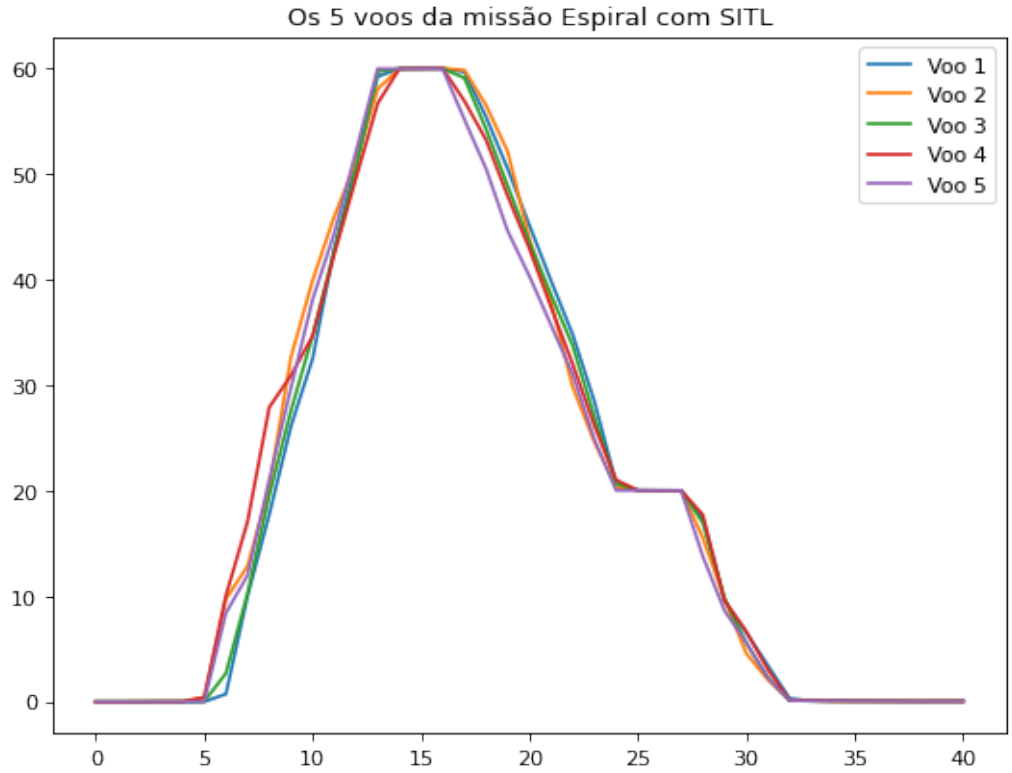

Figura 6.2: Os cinco testes de voos simulados (verificação)

A Figura 6.2 contém as altitudes dos cinco voo simulados, cada um representado por uma cor diferente. Como era esperado, os resultados apresentados na Figura 6.2 foram extremamente semelhantes. As pequenas diferenças apresentadas são em razão do funcionamento da própria simulação de mundo real, realizado pelo SITL, onde pequenas variações podem ocorrer, como por exemplo um intervalo maior para finalizar a decolagem.

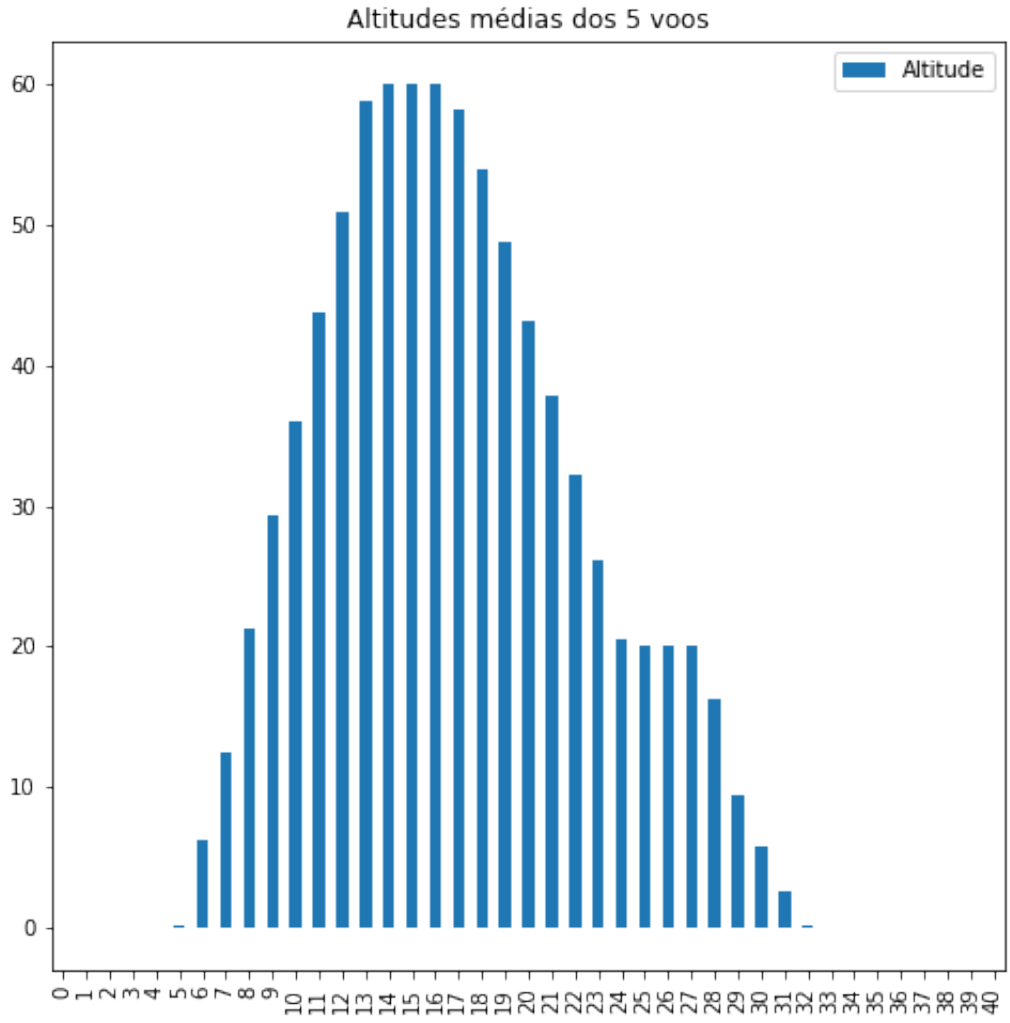

Figura 6.3: Altitudes médias dos cinco voos simulados

A partir das informações obtidas pelos cinco testes, foi gerado um conjunto de dados com as médias das altitudes. A Figura 6.3 representa esses valores obtidos. As médias das altitudes simuladas serão confrontadas com as altitudes obtidas pelo mesmo experimento Espiral, realizado em campo. Para este fim, um drone foi comandado com auxílio da FlexStation para percorrer os pontos determinados na missão Espiral e enviar sua altitude a cada segundo. Foi realizado o mesmo processo de extração utilizado anteriormente para os voos simulados, obtendo do arquivo log todas as altitudes registradas durante o voo em campo.

A Figura 6.4 ilustra as médias das altitudes dos voo simulados em comparação com as altitudes do voo em campo. Podemos perceber algumas discrepâncias entre os experimentos, como o intervalo maior pro drone em campo alcançar determinado ponto da missão. A calda mais alongada para o voo em campo, representando a perda gradual de altitude, indica justamente esse intervalo maior para percorrer os pontos finais. É possível perceber que

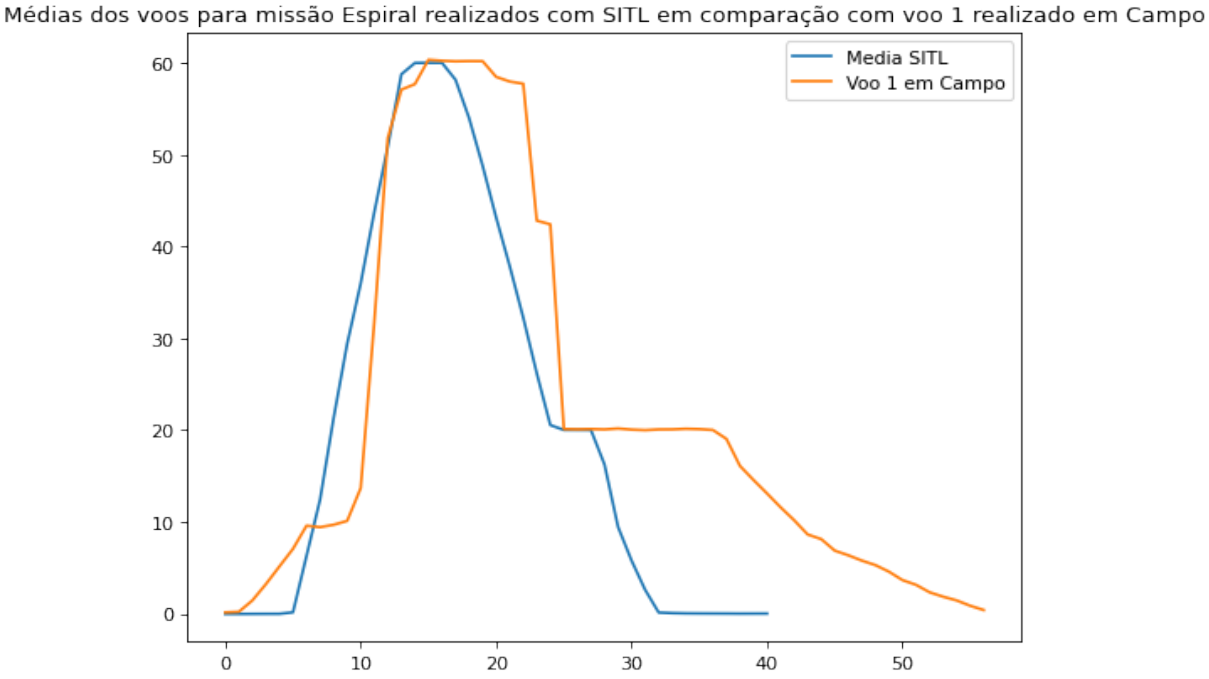

Figura 6.4: Comparação entre médias dos voo simulados com voo em campo (validação)

o tempo total de voo foi pelo menos dez segundo maior do que a média dos experimentos. Esse teste serviu como uma validação, em cima da verificação antes realizada. Outro ponto que podemos analisar é a linha mais arredondada para a média dos voos simulados, representando mudanças de altitudes menos bruscas. Esse fato ocorre por serem valores gerados a partir de uma média de cinco voos, as informações não são perdidas e a tendência natural é ocorrer essa suavização. Esse experimento final é justamente para ilustrar o auxílio da FlexStation no processo investigativo de uma prova de conceito e uma validação.

7
Conclusão

A FlexStation busca oferecer ferramentas auxiliares, para exibir, monitorar e depurar experimentos em redes de dispositivos móveis. O desenvolvimento desse framework busca auxiliar na integração da estação base com outros projetos, se propondo a fornecer serviço de uma estação base genérica, capaz de ser configurada e ajustada de acordo com requisitos necessários para os projetos em questão.

A futura adição de novas funcionalidades será bem-vinda, uma vez avaliadas como sendo fundamentais para uma variedade de projetos. É preciso, porém, manter a generalidade da FlexStation, sem incluir etapas desnecessárias de configuração ou dificultar esse processo.

As próximas etapas de desenvolvimento envolvem uma maior responsividade da interface e customização pelo usuário. É desejável fornecer ao usuário a capacidade de incluir ou excluir seções na interface, aumentar ou diminuir uma determinada seção, deixando da maneira pessoal para acompanhar um experimento. Também, busca-se continuar implementando alternativas de comunicação com dispositivos IoT, abrindo espaço para testar a FlexStation ao lidar com outros protocolos de comunicação externa.

Atualmente o módulo de simulação de drones está presente no diretório do projeto, com a intenção de testar funcionalidades da FlexStation, sem precisar fazer testes de campo frequentemente. É interessante separar esse módulo, uma vez que todos objetivos do projeto estejam implementados.

Referências bibliográficas

[1] FEKI, M.; KAWSAR, F.; BOUSSARD, M. ; TRAPPENIERS, L.. **Electronic publishing at ieee computer society: The internet of things: The next technological revolution**. IEEE Computer Society, 46(2):24–25, Fevereiro 2013.

[2] RAGHAVENDRA, K.; SHARANYA, P. ; PATIL, S.. **An iot based smart healthcare system using raspberry pi**. International Journal of Research and Scientific Innovation (IJRSI), 5:103–106, Junho 2018.

[3] GOMEZ, C.. **Internet of things for enabling smart environments: A technology-centric perspective**. Journal of Ambient Intelligence and Smart Environments, 11(1):23–43, Janeiro 2019.

[4] TONNEAU, A.-S.; MITTON, N. ; VANDAELE, J.. **A survey on (mobile) wireless sensor network experimentation testbeds**. In: 2014 IEEE INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING IN SENSOR SYSTEMS, p. 263–268, 2014.

[5] OLIVIERI, B.; PAULON, M. ; ENDLER, M.. **Gradys: Exploring movement awareness for efficient routing in ground-and-air dynamic sensor networks**. 2020.

[6] **Ieee standard for low-rate wireless networks**. `https://standards.ieee.org/ieee/802.15.4/7029/`. Acesso em: Maio/2022.

[7] **Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications**. `https://standards.ieee.org/ieee/802.11/3605/`. Acesso em: Maio/2022.

[8] **Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements part 11: Wireless lan**

medium access control (mac) and physical layer (phy) specifications amendment 10: Mesh networking. `https://standards.ieee.org/ieee/802.11s/4243/`. Acesso em: Maio/2022.

[9] **Introducing json**. ://www.json.org/json-en.html. Acesso em: Maio/2022.

[10] **Flytbase: Enterprise drone automation platform**. `https://flytbase.com/`. Acesso em: Setembro/2021.

[11] **Qgroundcontrol: Drone control**. `http://qgroundcontrol.com/`. Acesso em: Setembro/2021.

[12] PEREZ, D.; MAZA, I. ; CABELLERO, F.. **A ground control station for a multi-uav surveillance system**. J Intell Robot Syst, 69:119–130, Janeiro 2013.

[13] **Paparazzigcs**. `https://wiki.paparazziuav.org/wiki/GCS/`. Acesso em: Setembro/2021.

[14] HATTENBERGER, G.; BRONZ, M. ; GORRAZ, M.. **Using the paparazzi uav system for scientific research**. p. 247–252, Setembro 2014.

[15] **SITL simulator (software in the loop)**. `https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html`. Acesso em: Fevereiro/2022.

[16] **Ardupilot**. `https://ardupilot.org/`. Acesso em: Fevereiro/2022.

[17] **Pandas**. `https://pandas.pydata.org/`. Acesso em: Maio/2022.