UNIVERSIDADE DO VALE DO PARAÍBA

FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO

ENGENHARIA DA COMPUTAÇÃO

Plataforma de Recebimento e Análise de Dados de Nanossatélites

IAGO VINICIUS DA COSTA LUNA
URIEL GONÇALVES PAIVA DA CONCEIÇÃO

Orientador: Prof. Me. Hélio Lourenco Esperidião Ferreira
Orientador externo: Dr. Carlos Augusto Paiva Lameirinhas da Conceição

São José dos Campos, SP
2024

Resumo

Os nanossatélites estão ganhando popularidade nos tempos atuais devido ao seu baixo custo e eficiência para coleta de dados. Ademais, estes satélites de pequenos portes podem ser lançados junto com outros no payload de um foguete, assim reduzindo os custos de lançamento e apresentando resultados relevantes. Umas das tecnologias mais utilizadas para a comunicação destes dispositivos é o LoRa, uma tecnologia de transmissão a rádio por espalhamento de espectro em chirps. Esta tecnologia permite uma transmissão de baixa potência que pode alcançar longas distâncias. Neste trabalho foi construído um protótipo de uma estação de solo capaz de coletar dados em tempo real de um nanossatélite simulado e exibi-los para os usuários, armazenando as informações de umidade, temperatura, quantidade de CO, informações de rotação e de aceleração no banco de dados InfluxDb. A estação possibilita o cálculo de métricas como média, mediana, desvio padrão, variância, amplitude, moda e quartis para apoiar na análise, além de apresentar gráficos que permitem visualizar a variação dos dados coletados durante o lançamento. Também oferece a opção de acessar dados de lançamentos anteriores ou de realizar downloads em formato Excel, complementados por uma interface que exibe uma simulação 3D representando a movimentação do nanossatélite durante seu lançamento.

Palavras-chave: Nanossátelites, Tempo real, Estação de solo, LoRa.

Abstract

Nanosatellites are gaining popularity nowadays due to their low cost and efficiency in data collection. Moreover, these small satellites can be launched alongside others in a rocket's payload, thereby reducing launch costs and delivering relevant results. One of the most widely used technologies for communication in these devices is LoRa, a radio transmission technology based on chirp spread spectrum. This technology enables low-power transmission over long distances. In this project, a prototype of a ground station was built, capable of collecting real-time data from a simulated nanosatellite and displaying it to users while storing information such as humidity, temperature, CO levels, rotation, and acceleration data in the InfluxDB database. The station allows the calculation of metrics such as mean, median, standard deviation, variance, range, mode, and quartiles to support analysis, in addition to presenting graphs that visualize the variation of the data collected during the launch. It also offers the option to access data from previous launches or download them in Excel format, complemented by an interface that displays a 3D simulation representing the nanosatellite's movement during its launch.

Keywords: Nanosatellites, Real-time, Ground station, LoRa.

Lista de Figura

Lista de Abreviaturas e Siglas

LEO	*Low Earth Orbit*
IoT	*Internet of Things*
M2M	*Machine to Machine*
INPE	Instituto Nacional de Pesquisa e Espaciais
SBCD	Sistema Brasileiro de Coleta de Dados
PCD	Plataforma de Coleta de Dados Ambientais
CRC	Centro de Rastreamento e Controle de Satélites
OSI	*Open Systems Interconnections*
LASC	*Latin American Space Challenge*
UNIVAP	Universidade do Vale Paraíba
CO	Monóxido de Carbono
WEB	*World Wide Web*
LoRa	*Long Range*
MQTT	*Message Queing Telemetry Transport*
API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
OLED	*Organic Light-Emitting Diode*
WI-FI	*Wireless Fidelity*
CSS	*Cascading Style Sheet*
JSON	*JavaScript Object Notation*
SPI	*Serial Peripheral Interface*
UART	*Universal Asynchronous Receiver/Transmitter*
REST	*Representational State Transfer*
URL	*Uniform Resource Locator*

QoS	*Quality of Services*
HTML	*Hypertext Markup Language*
IIASA	*International Institute for Applied Systems Analysis*
GOES	*Geostationary Operational Environmental Satellite*
GNSS	*Global Navigation Satellite System*
GPS	*Global Positioning System*

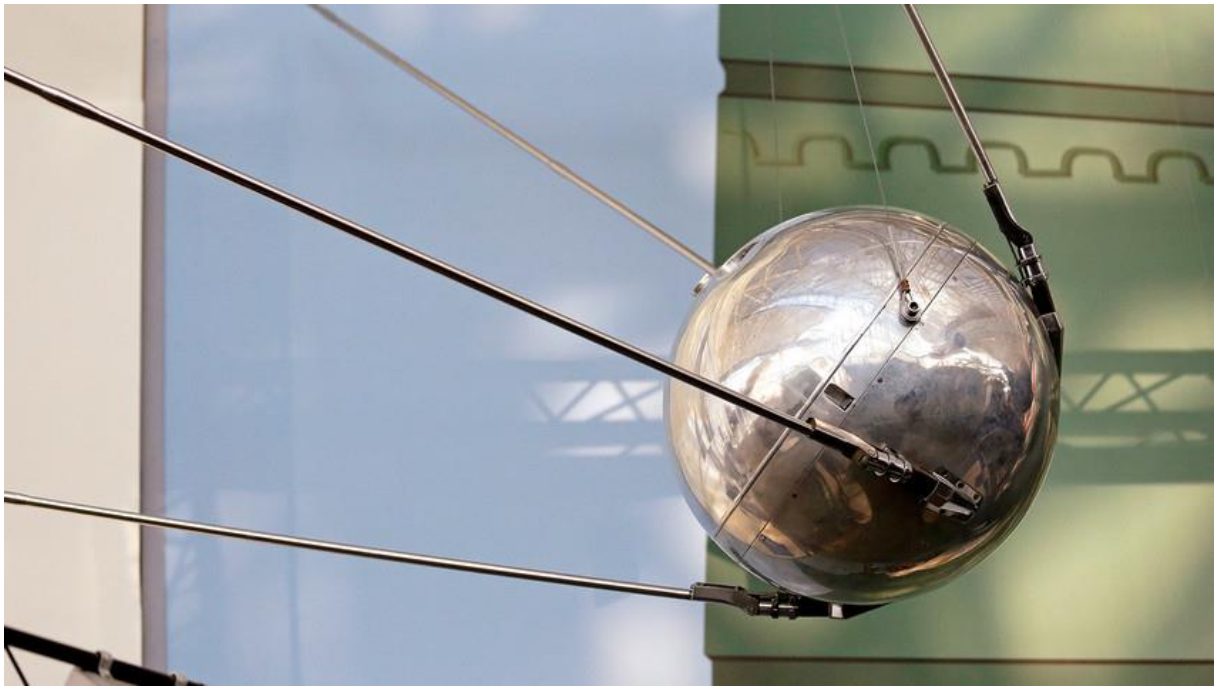
Sumário

1 Introdução

Um satélite pode ser classificado de duas maneiras, dependendo de qual perspectiva é analisada. Do ponto de vista da astronomia, é tudo o que órbita algo de maior tamanho, podendo também ser chamado de satélite natural, por exemplo a Lua é um satélite natural da Terra. Já no ponto de vista da astronáutica, um satélite artificial é um objeto construído pelos seres humanos e são lançados para a órbita de um planeta. [10]

O Sputnik 1 foi o primeiro satélite artificial a ser lançado para a órbita terrestre. Sua história teve início em 1952 quando o Conselho Internacional de Ciência declarou que na data 1 de julho de 1957 até dia 30 de dezembro de 1958, seria o Ano Internacional da Geofísica, devido a descoberta que os ciclos de atividade magnética solar estariam em alta nestas datas. Baseado neste fato os pesquisadores decidiram lançar um satélite artificial durante este período para mapear a superfície terrestre. O Sputnik 1 tem um tamanho de aproximadamente uma bola de basquete e pesa 83Kg como pode ser observado na Figura 1. Seu lançamento para órbita terrestre ocorreu no dia 4 de outubro de 1957 pela União Soviética, e demorou 98 minutos para orbitar a terra, marcando o início da era espacial [11]

Figura 1 - Primeiro satélite artificial



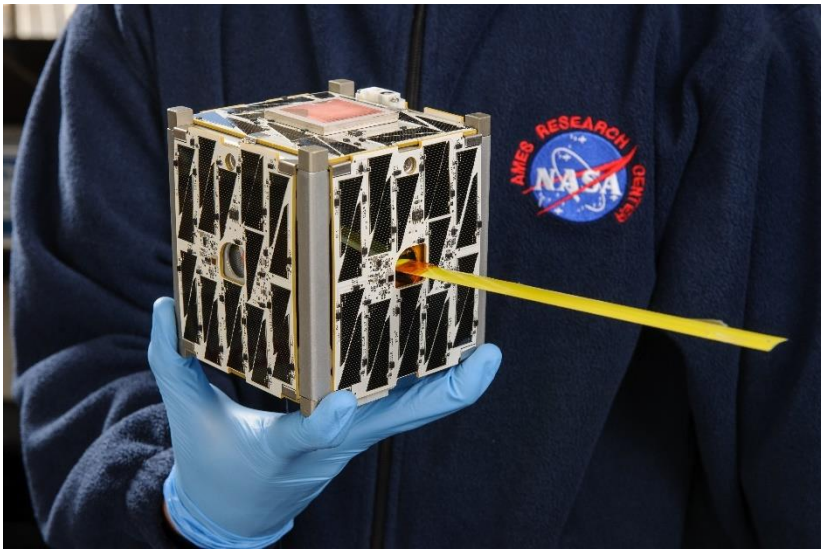
Fonte – Russia Beyond [12]

As estações de solo têm o papel ser o suporte às operações dos satélites e na utilização dos dados adquiridos por eles. São responsáveis por rastrear, monitorar e comandar os satélites, além de receber, processar e distribuir as informações coletadas para diversas finalidades, como monitoramento ambiental, navegação e previsão do tempo. Com o avanço da tecnologia, essas estações evoluíram de sistemas básicos de rastreamento para estruturas altamente automatizadas, equipadas com antenas de alta precisão e softwares especializados na análise de dados. Além disso, as estações de solo são fundamentais para enviar comandos aos satélites, como ajustes de órbita ou reconfigurações de sensores, visando o funcionamento contínuo das missões. Também desempenham um papel importante na colaboração internacional, integrando redes globais que permitem estudos climáticos, gestão de desastres naturais e pesquisas científicas.

No decorrer da corrida espacial (1957-1975), os satélites possuíam especificações únicas, sendo bem detalhados visualmente. No início da década de 1980, foi criado o conceito dos nanossatélites, os quais possuíam um *design* diferenciado quando comparado com seus antecessores, devido a intenção de reduzir os custos. [1]

Com o crescimento da produção de pequenos satélites deu início a uma categoria conhecida como nanosat (1-10kg). Outras categorias desta modalidade são: microssatélites (10-100kg), picosatélites (0.1-1kg) e femtosatélites (menos de 0.1kg). [2]. A denominação mais comum para um nanossatélite é *CubeSats*, os quais costumam ser construídos nas medidas 10x10x10cm, como pode ser observado na Figura 2. Possuem um sistema escalável ao longo de um dos eixos, por meio de incrementos de 1U até 12U, onde 1U significa um cubo nas medidas especificadas anteriormente. [1]

Figura 2 - CubeSat PhoneSat 2.5



Fonte: NASA [9]

Os CubeSats geralmente são lançados em *Low Earth Orbit* (LEO), localizada entre 160 e 2 mil km acima da superfície da Terra. Seu período orbital varia de 90 a 105 minutos, e sua vida útil vai de 8 meses a 5 anos. Normalmente, são colocados em órbita como carga secundária de veículos lançadores que transportam satélites maiores como carga principal.

Nanossatélites têm ganhado destaque devido ao baixo custo e simplicidade. São usados em aplicações como *Internet of Things* (IoT) e *Machine to Machine* (M2M). Exemplos incluem o AprizeSat, para monitoramento de combustível, oleodutos e rastreamento de contêineres, e o projeto brasileiro CONASAT, que utiliza CubeSats para coletar dados e imagens ambientais.

Esse projeto brasileiro foi idealizado pelo Instituto Nacional de Pesquisa e Espaciais (INPE). O CONASAT visa estabelecer uma comunicação de baixo custo e alta qualidade para assegurar a continuidade do Sistema Brasileiro de Coleta de Dados (SBCD), que tem como finalidade oferecer ao país um sistema de coleta de dados por satélites para várias aplicações, como o monitoramento de bacias hidrológicas, previsão meteorológica e climática, avaliação do potencial de energias renováveis, entre outras. O SBCD é composto, em sua grande maioria, por Plataformas de Coleta de Dados Ambientais (PCDs) e antenas, as quais estão distribuídas pelo território brasileiro e em algumas plataformas marítimas. Essas estações estão localizadas em pontos estratégicos para garantir a cobertura das áreas monitoradas, como regiões remotas próximas à Floresta Amazônica.

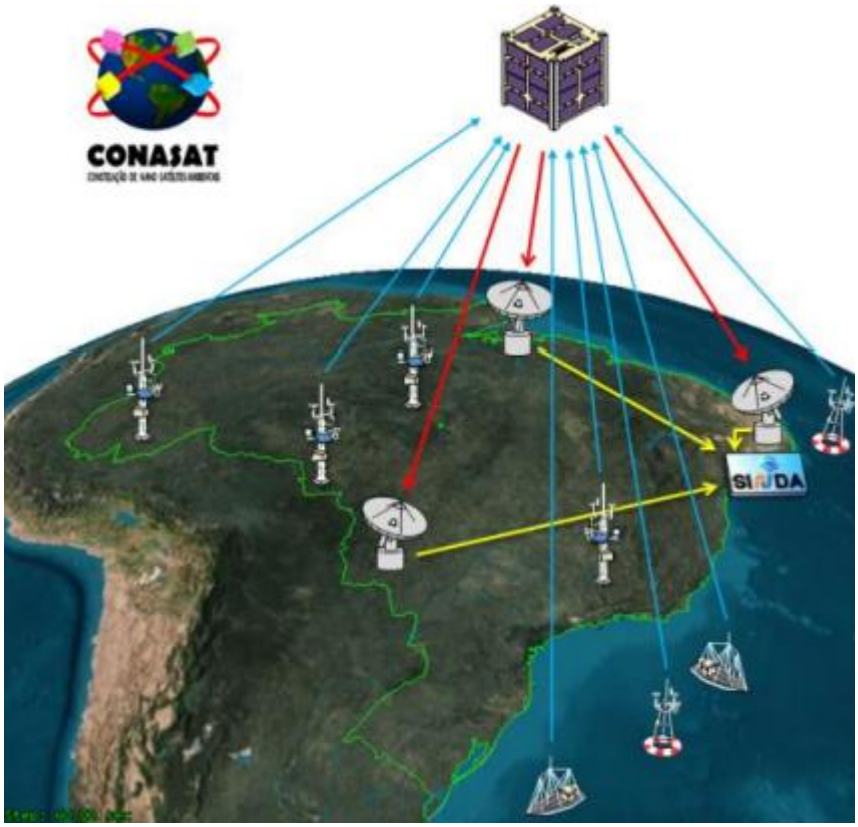
Cada PCD possui sensores específicos, dependendo da sua função, como medir a temperatura, umidade, vazão do rio, entre outros. As informações coletadas pelos sensores são transmitidas a cada 200 segundos por um transmissor na frequência de 401,62 MHz e recebidas

pelas estações de solo, responsáveis por processar e retransmitir os dados. Essas estações, equipadas com antenas de rastreamento, sistemas de comunicação e servidores para análise, também podem enviar comandos para configurar os sensores ou ajustar os satélites, garantindo o funcionamento adequado do sistema.

Além disso, essa comunicação conta com o protocolo AX.25, o qual pertence à camada de enlace do modelo OSI e foi projetado para ser utilizado por operadores de rádio amadores. Algumas estações brasileiras, como o Centro de Rastreamento e Controle de Satélites (CRC) em Cuiabá, também integram redes globais, contribuindo para pesquisas sobre mudanças climáticas e desastres naturais, reforçando a importância desse sistema.

A rede de grande abrangência territorial de PCDs teve sua origem no escopo da Missão Espacial Completa Brasileira. Quando o Satélite de Coleta de Dados 1 (SCD-1) foi lançado em 1993, existiam cerca de 60 PCDs. Este projeto foi crescendo e no ano de 2008, já havia mais de 800 PCDs instalados no território brasileiro.

Figura 3 - Arquitetura do SBCD



Fonte: CONASAT - Constelação de Nano Satélites para Coleta de Dados Ambientais [4]

Os subsistemas de telemetria (informação enviada para o satélite) e telecomando (comando enviado pelo satélite), componentes fundamentais do CONASAT e de qualquer sistema que envolve a troca de dados por satélite e sua base de solo, são essenciais para assegurar a comunicação entre o satélite e a Terra, permitindo a manutenção e o controle da plataforma. Esta comunicação é possível apenas durante os períodos em que o satélite está ao alcance de uma estação terrestre, permanecendo inativa no restante do tempo. O sistema de telemetria é responsável por transmitir, a cada passagem, os dados coletados por sensores a bordo, distribuídos entre todos os subsistemas, fornecendo informações sobre as condições internas do satélite (*housekeeping*). Essa funcionalidade garante que os dados ambientais (pressão atmosférica, precipitação, radiação solar, umidade do ar, temperatura, direção e velocidade do vento, ponto de orvalho, detecção da variação dos níveis de corpos de água) coletados pelas PCDs sejam enviados de forma eficaz para as estações terrestres, assegurando o monitoramento contínuo e preciso das condições ambientais no Brasil.

O projeto envolve um sistema integrado capaz de receber dados (umidade, temperatura, altitute aproximada, quantidade de CO no ambiente e a aceleração) de um prototipo de nanossatélite e apresentá-los em formato web (parte visual e interativa de um site) com uma interface gráfica. Esse sistema foi projetado para exibir os dados em tempo real, utilizando gráficos permitindo a visualização e interpretação das informações recebidas. Possui tambem uma interface 3D (permite ver o satélite em 3 dimensões) a qual simula a movimentação de voo do prototipo de nanossatélite, por meio dos dados coletados durante o lançamento. A transmissão dos dados é feita por meio de rádiofrequencia, aproveitando o longo alcance permitido pelos módulos Long Range Wide Area (LoRa), em conjunto com um microcontrolador de modelo ESP32.

Este projeto foi desenvolvido especialmente para a competição LASC onde o prototipo de nanossatélite será lançado e a estação irá exibir os dados, provando o funcionamento dos sensores e a capacidade de transmitir dados, como se fosse um satélite artificial real.

2 Materiais e Métodos

O projeto foi desenvolvido no laboratório da da Universidade do Vale do Paraíba (UNIVAP) e baseia-se na construção de um equipamento capaz de receber dados envidados por um protótipo de nanossátelite. Os controles dos parâmetros de funcionamento são realizados via web.

A construção do equipamento foi dividida em quatro partes: a construção do hardware; a comunicação entre a estação de solo e o nanossátelite; a manipulação e o armazenamento das informações coletadas em um banco de dados e o processo de criar a interface para exibir os dados coletados para o usuário final.

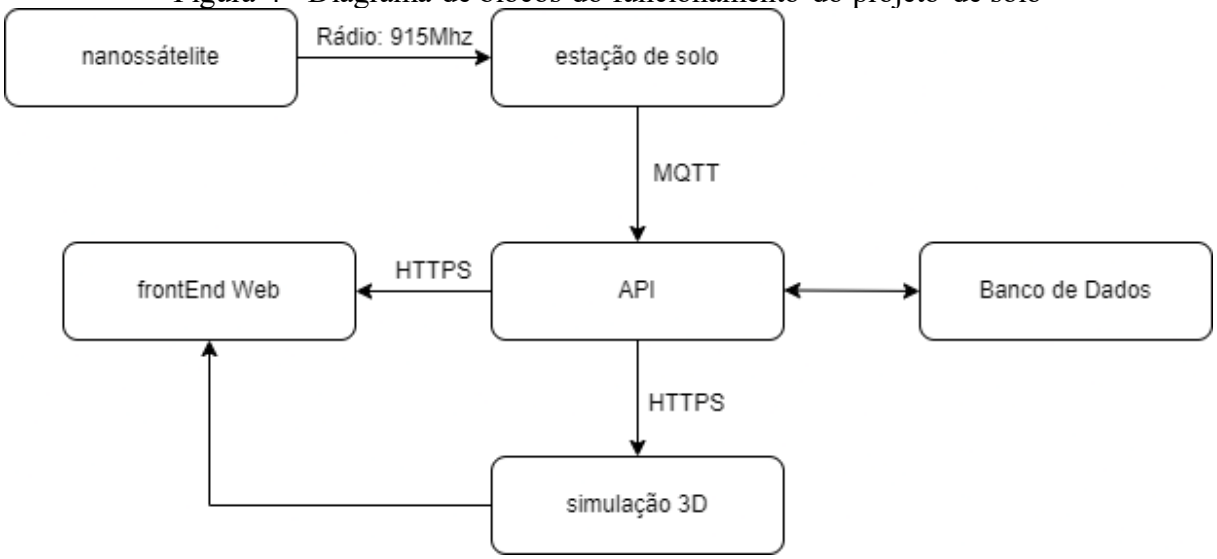
O diagrama de bloco apresentado na Figura 4 descreve o princípio de funcionamento do projeto, o qual envolve a comunicação entre a antena de recebimento de dados, e o software responsável por exibir e monitorar as informações coletadas.

O *backEnd* (programação dedicada à parte interna de uma aplicação, a parte não visível para o usuário) *web* recebe os conteúdos coletados pela estação de solo, e exibe no *frontEnd* (parte visual de um site ou aplicativo), criado por meio da linguagem HTML, JavaScript e CSS. O *back* também tem a função de guardar os dados adquiridos em um Banco de Dados programado em Flux (a linguagem de programação exclusiva do InfluxDB) através do sistema de gestão InfluxDB, além de fornecer os dados para a simulação desenvolvida em JavaScript por meio da biblioteca Three.js. O *backEnd web* é responsável por receber e processar os dados coletados pela estação de solo, transmitindo-os ao *frontEnd*, desenvolvido com HTML, JavaScript e CSS, para exibição em tempo real. As informações transmitidas pelo nanossatélite englobam altitude, umidade, temperatura e níveis de monóxido de carbono (CO), orientação e aceleração. Após receber essas informações, a estação de solo, realiza cálculos adicionais para produzir métricas derivadas das informações recebidas.

O protocolo de comunicação MQTT é utilizado para a transferência dos dados, estabelecendo uma comunicação entre a estação de solo e um broker (servidor responsável por gerenciar a comunicação MQTT). Um programa dedicado, inscrito no canal MQTT de envio ao broker, capta essas mensagens e atua como uma ponte para o banco de dados. Desta forma efetua o registro diretamente por meio de uma API (*Application Programming Interface*) específica. Assim, o sistema de *backEnd* guarda as informações no banco de dados, administrado pelo InfluxDB, empregando Flux para realizar consultas e manipular os dados. Adicionalmente, o *back* alimenta uma simulação em JavaScript, desenvolvida com a biblioteca Three.js, permitindo visualizar os movimentos dos satélites por meio das informações coletadas

atravéz de um sensor giroscópio, o qual é capaz de detectar a taxa de rotação em torno de seu eixo.

Figura 4 - Diagrama de blocos do funcionamento do projeto de solo



Fonte: O autor.

O LoRaWAN é o protocolo de comunicação entre o nanossatélite e sua estação de solo do projeto, usando o rádio LoRa como meio de transmissão sem fio. Este protocolo possibilita a construção de uma rede completa, a qual abrange o endereçamento de dispositivos finais (*end-devices*) ou seja o destino de uma mensagem transmitida pela rede, gateways (sistema ou equipamento que conecta redes diferentes) e até mecanismos para prevenir colisão de pacotes.

O HTTP (*Hypertext Transfer Protocol*), ou Protocolo de Transferência de Hipertexto, é encarregado de estabelecer um padrão de comunicação para a troca de mensagens entre clientes e servidores. Neste projeto, utilizou-se o HTTP para transferir os dados da API para o *frontEnd*, de maneira ágil e compreensível para o usuário.

2.1 Nanossatélite

O nanossatélite encarregado de transmitir dados para esta estação de solo faz parte de um projeto de Trabalho de Conclusão de Curso distinto. Trata-se de um *CubeSat* 1U com medidas 10x10x10cm, equipado com sensores para a coleta de informações de altitude (m), umidade (g/m³), temperatura (°C), concentração de CO (ppm), aceleração e rotação. Este satélite enviará os dados em tempo real para a estação de solo, que apresentará e criará gráficos dinâmicos. Adicionalmente, a estação disponibilizará

ferramentas para calcular métricas dos dados ao término da transmissão, bem como a possibilidade de consultar dados de lançamentos passados e realizar o download para análises mais aprofundadas.

2.2 Estação de Solo

A estação de solo é constituída de um microcontrolador ESP32 LoRa 868-915 MHz, e uma antena de 20cm, utilizada para ampliar o alcance de transmissão. O ESP32 LoRa (Espressif) é o microcontrolador utilizado no projeto e este já possui um módulo LoRa 868/915Mhz integrado junto com um display OLED 0.96, os componentes podem ser observados na Tabela 1. Também possui wi-fi e Bluetooth 4.2. Este dispositivo foi escolhido por conter maior quantidade de canais-analógicos e digitais, se comparado com o ESP8622.

Tabela 1 – Componentes Estação de Solo

Componentes Estação de Solo
ESP 32 LoRa
LoRa 868/915Mhz
display OLED 0.96

Fonte: Autor

O hardware contém um processador *dual-core* de 32 bits capaz de operar em uma frequência de até 240 MHz, o que permite que a placa execute tarefas de modo simultâneo sem que isso comprometa seu desempenho. Ideal para o projeto pois está ferramenta permite realizar uma comunicação WI-FI e ao mesmo tempo monitorar aplicações em tempo real. O ESP32 pode ser utilizado em diversas aplicações como dispositivos IoT, monitoramento remoto, automação residencial, aplicações industriais e agrícolas

2.2.1 Tecnologia LoRa

O módulo LoRa (*Long Range*) foi construído com a finalidade de oferecer uma comunicação sem fio de longa distância, com baixo consumo de energia. Isso permitiu que o dispositivo fosse amplamente utilizado e eficaz em projetos de Internet das Coisas (IoT). A tecnologia emprega um método de espalhamento espectral por *chirp* (CSS), ou seja, sua frequência aumenta ou diminui com o tempo, projetado para assegurar uma transmissão robusta com uma boa resistência a interferências, mesmo exposto a condições adversas. Esta

característica permite-lhe possuir uma conexão estável de transmissão tanto em áreas rurais quanto na cidade, onde a existência de obstáculos e ruídos pode representar um impasse para as comunicações via rádio frequência. [17]

O LoRa integra o protocolo LoRaWAN (Rede de Longa Distância e Área Ampla), um padrão de rede de baixa potência e ampla cobertura. Esta interação entre o módulo LoRa e a arquitetura LoRaWAN permite a formação de redes expansíveis que comportam milhares de aparelhos com requisitos de infraestrutura mínimos. As aplicações comuns deste protocolo abrangem a vigilância ambiental, monitoramento de agricultura, o rastreamento de ativos e a comunicação em regiões isoladas, como as atividades de satélites de pequeno porte e nanosatélites. [18]

Os pacotes enviados pelo LoRa são compostos de texto no formato JSON (JavaScript Object Notation), um padrão otimizado de notação de dados em JavaScript. Esse formato facilita a transmissão eficiente dos dados pelo protocolo HTTP, além de ser muito utilizado por sua simplicidade e compatibilidade com diversos sistemas. Um exemplo de como é o pacote transmitido pelo nanossatélite através do LoRa:


```
{
 "Medicao": 2,
 "Umid": 3,
 "Temp": 24,
 "AltAprox": 1.5,
 "CO": 2,
 "AcelX": -0.03,
 "AcelY": 0.15,
 "AcelZ": 9.86,
 "GyroX": 0.01,
 "GyroY": 0,
 "GyroZ": 0,
 "Lat": 0,
 "Lon": 0
}
```

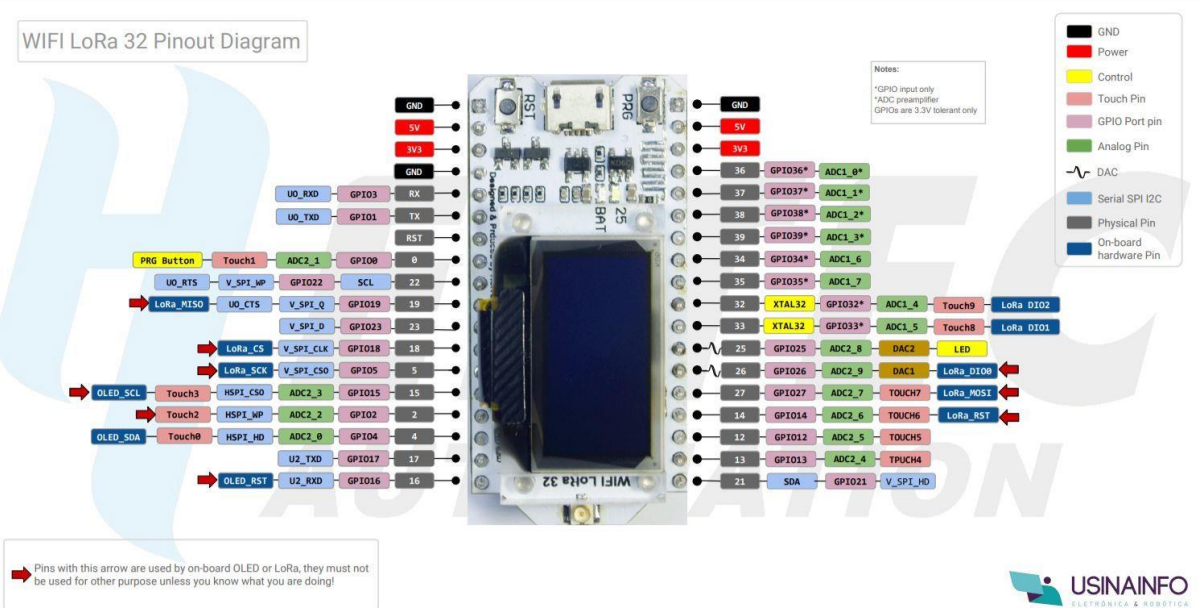

Este projeto emprega especificamente o chip SX1276, que está incorporado ao módulo ESP32 LoRa. Este modelo foi selecionado precisamente por estar integrado ao microcontrolador, dispensando a montagem de um circuito extra para ligar o ESP32 ao módulo

LoRa, o que torna a implementação simples. O SX1276 possui uma alta sensibilidade de -139 dBm, e uma potência de saída de +20 dBm, atributos que asseguram uma comunicação de longa distância e uma maior confiabilidade do sinal. Ademais, ele trabalha com uma comunicação *SPI half-duplex* (meio de comunicação que permite a troca de dados entre dispositivos transmissor e receptor) e tem uma taxa de bits programável de até 300 kbps.

2.2.2 Microcontrolador ESP32 LoRa

A Figura 5 apresenta as divisões de pinos que podem ser utilizados, tais como, conversores analógicos e digitais, protocolo UART, protocolo I2C, protocolo SPI, *touch* dentre outros.

Figura 5 - Definição de pinos do microcontrolador ESP32 LoRa



Fonte: USINAINFO [6]

O modulo LoRa integrado é o hardware responsável por realizar a comunicação de rádio frequência. Permite alcançar em torno 3 a 4km em uma área urbana e aproximadamente 12km em um ambiente rural.

Um satélite enquanto está sendo lançado, devido à alta velocidade do foguete ocorre o efeito Doppler, pois quando o foguete se afasta da estação, as ondas de rádio se alongam, resultando em uma frequência menor do que a emitida originalmente. Quando o foguete se aproxima, as ondas se comprimem, aumentando a frequência, essas mudanças na frequência

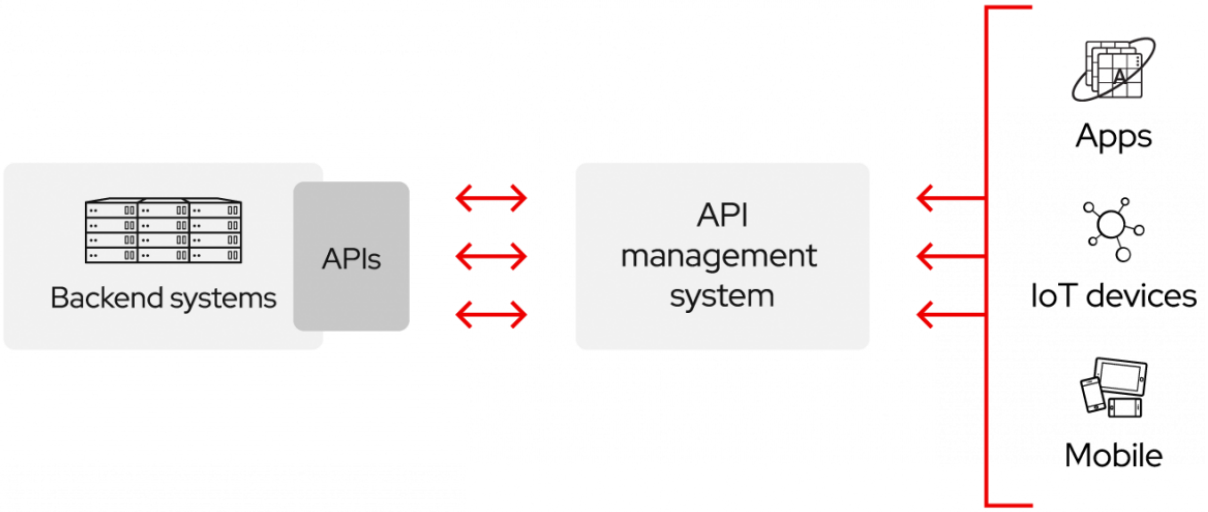
podem fazer com que o sinal saia da faixa de recepção das antenas. O LoRa conta com um espalhamento espectral, isso permite uma baixa sensibilidade integrada ao efeito Doppler. [8]

Os dados recebidos pelo nanossatélite serão criptografados em uma camada pelo protocolo HTTPs para respeitar o conceito de integridade da segurança da informação. Assim que essas informações forem interceptadas por um agente malicioso, o mesmo encontrara dificuldade para manipular a transmissão de dados e inserir dados falsos na comunicação.

2.3 REST API de Manipulação de dados

API é a sigla para *Application Programming Interface*, ou, em português, Interface de Programação de Aplicações. Trata-se de um conjunto estruturado de ferramentas, definições e protocolos destinados ao desenvolvimento de software. As APIs têm como objetivo integrar soluções e serviços, abstraindo detalhes de implementação, o que simplifica o processo de desenvolvimento de aplicações, bem como seu design, gerenciamento e utilização. Ao receber uma solicitação remota devidamente estruturada, a API define como a aplicação desenvolvida deve responder. A Figura 6 não apresenta a API específica utilizada neste projeto, mas demonstra o funcionamento de um sistema para gerenciamento de múltiplas APIs. A imagem foi empregada exclusivamente para ilustrar, de forma visual, o fluxo de dados coletados por dispositivos de Internet das Coisas para o *back-end*.

Figura 6 - Funcionamento de uma API



Fonte: RedHat. [15].

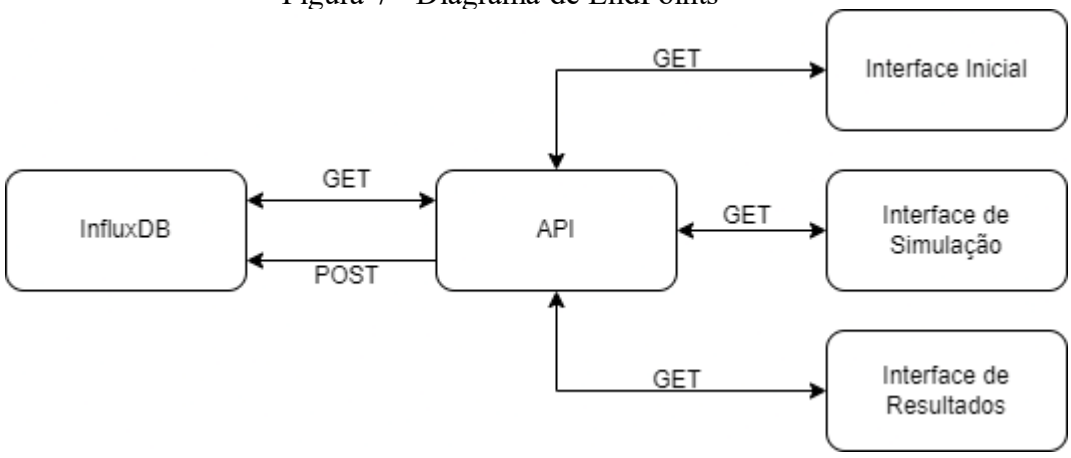
Representational State Transfer (REST) é uma arquitetura para a criação de serviços *web* escaláveis, que utiliza métodos HTTP padrão, como GET, PUT, POST e DELETE, para acessar

e manipular recursos em servidores. APIs RESTful são baseadas em uma estrutura simples, permitindo comunicação entre sistemas por meio de requisições e respostas de dados, utilizando URLs (*Uniform Resource Locator*, endereço de rede onde se encontra algum recurso) para identificar os recursos. As respostas são dados simples, sem a necessidade de uma interface gráfica. Utilizar a arquitetura REST implica seguir restrições que elaboram interfaces bem definidas e padronizadas, facilitando a comunicação entre cliente e servidor, comumente utilizando formatos como JSON ou XML. Isso torna as APIs RESTful ideais para integrar diversos sistemas de forma eficiente e flexível.

A linguagem de programação Python foi utilizada para o desenvolvimento daAPI do projeto. Esta escolha foi feita devido ao seu suporte para a integração do protocolo MQTT e por sua facilidade de aprendizado e uso.

O *microframework* (*frameworks* modularizado que possuem uma estrutura inicial mais simples que um *framework* convencional) para Python conhecido como Flask foi empregado para o desenvolvimento *web*, oferecendo algumas funcionalidades para o desenvolvimento de sites e APIs. As características deste *framework* apresentam uma curva de aprendizado suave, proporcionando uma base robusta para projetos que empregam uma solução personalizada possibilitando seu crescimento com o passar do tempo. Isso permite uma expansão simples do aplicativo e o acréscimo de novas funcionalidades conforme a necessidade futura. A Figura 7 a seguir apresenta um diagrama de *EndPoints* que representam extremidades de conexão API [21]

Figura 7 - Diagrama de EndPoints



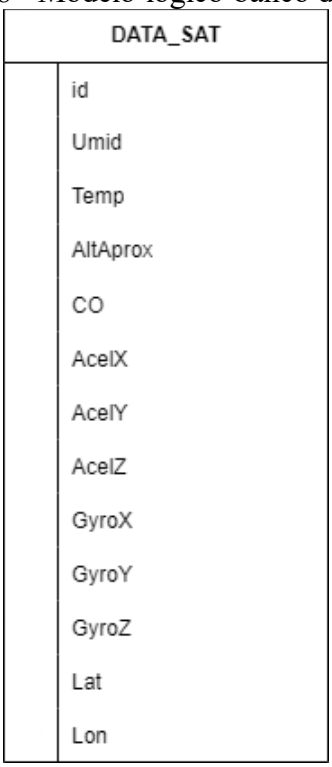
Fonte: Autor

2.4 Banco de Dados

O sistema de gestão de banco de dados não relacional InfluxDB foi implementado para armazenar os dados deste projeto. Este banco de dados, otimizado para séries temporais, permite registrar e consultar os dados contínuos ao longo do tempo, como os gerados pelo nanossatélite. A estrutura criada no InfluxDB para o projeto conta com uma coleção principal: a coleção "RegistroDeDados", que recebe as leituras de altitude, temperatura, pressão, umidade do ar, monóxido de carbono, acelerômetro nos três eixos (X, Y, Z); e os dados de um giroscópio nos três eixos (X, Y, Z). A relação entre os dados é organizada com base em *tags* (recursos que ajuda a monitorar aplicações em tempo real) e campos temporais, facilitando a busca e o agrupamento de registros para uma análise rápida e precisa. A escolha do InfluxDB se deve à sua alta performance para armazenamento e recuperação de grandes volumes de dados temporais, além de oferecer flexibilidade na manipulação de informações e escalabilidade em projetos voltados para análise de dados. O modelo lógico do banco de dados é exibido na Figura 8.

A estação de solo recebe os dados e os transmite ao servidor via protocolo MQTT. A API web é responsável por inserir esses dados no banco de dados e, posteriormente, extrair as informações para serem processadas e exibidas pela interface web.

Figura 8 - Modelo lógico banco de dados



Fonte: Autor

2.5 Codificação do sistema microcontrolado ESP32

O desenvolvimento da programação para integração do ESP32 com LoRa e MQTT foi desenvolvido na linguagem de programação C/C++, com a utilização de diversas bibliotecas específicas para garantir a funcionalidade do projeto. Foram importadas as bibliotecas WiFi.h para conexão à rede sem fio, PubSubClient.h para comunicação MQTT, SPI.h para suporte à comunicação SPI , além da LoRa.h para permitir a configuração e operação do módulo LoRa.

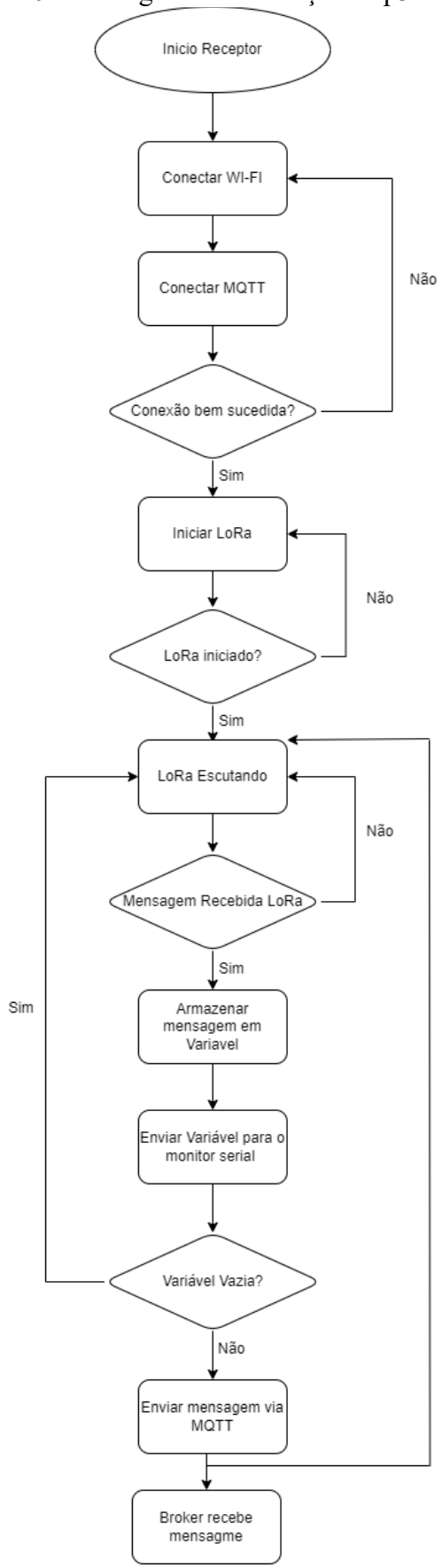
Primeiro, foram configurados os parâmetros básicos, como as credenciais da rede Wi-Fi (nome e senha) e as informações do servidor MQTT, que é usado para enviar e receber dados. Também foram definidos os números dos pinos que o módulo LoRa utiliza para funcionar, além da frequência de operação dele, que é de 915 MHz.

No método setup(), foi configurada a conexão Wi-Fi utilizando a função setup_wifi(), que faz uma tentativa de conexão e apresenta o endereço IP quando bem-sucedida. Em seguida, foi configurado o cliente MQTT por meio da função client.setServer(). O módulo LoRa foi inicializado, com a função LoRa.setPins() definindo os pinos corretos e a frequência estabelecida para a comunicação. A função LoRa.setSyncWord() foi utilizada para garantir que apenas dispositivos com a mesma palavra de sincronização pudessem se comunicar, melhorando a segurança e a organização da rede.

O método loop() é responsável por manter a conexão MQTT ativa, chamando a função reconnect() caso a conexão seja perdida. A função client.loop() garante que o ESP32 continue verificando e processando mensagens MQTT. A cada 5 (cinco) segundos, o código verifica se há novos pacotes recebidos pelo LoRa, e, se houver, a mensagem é armazenada na variável msg.

A mensagem recebida é então salva em uma variável que em seguida será serializado e enviada para o tópico MQTT configurado por meio de client.publish(), permitindo que outros dispositivos conectados ao servidor recebam os dados. O envio é confirmado no monitor serial com a exibição da mensagem enviada. O fluxograma do funcionamento do código citado acima pode ser observado por meio da Figura 9.

Figura 9 - Fluxograma codificação Esp32



Fonte: Autor

2.6 Integração Hardware e Software

É fundamental a conexão entre hardware e software para assegurar que as informações recolhidas pelo hardware sejam armazenadas de forma segura em um banco de dados, facilitando consultas futuras. Esta integração também permite que os dados sejam apresentados ao usuário em tempo real e possibilita o processamento para produzir métricas pertinentes. O principal objetivo dessa integração é dividir as responsabilidades: o hardware se dedica apenas à coleta e transmissão de dados, enquanto o software cuida do processamento e apresentação desses dados por meio de interfaces. Este sistema impede que o hardware perca desempenho ao realizar funções extras, reduzindo os atrasos na transmissão e evitando o desencaminhamento de informações. A integração é feita através do ESP32 que transmite as informações utilizando o protocolo MQTT, para um receptor do software, que será responsável por salvar as informações no banco de dados InfluxDB.

Para o desenvolvimento deste projeto foi utilizado a metodologia de engenharia de software Kanban, onde as atividades foram separadas entre os membros e criado um quadro de atividades sendo realizadas por cada um, atividades concluídas e atividades em andamento. Assim possibilitando um fluxo visual que permite ver o andamento do projeto.

2.6.1 MQTT

O protocolo MQTT (*Message Queuing Telemetry Transport*) é utilizado para transferir os dados da estação de solo para o banco de dados. Este protocolo de comunicação se fundamenta em normas de interação entre aparelhos, particularmente em contextos de Internet das Coisas (IoT). Foi desenvolvido para operar bem em redes com recursos e largura de banda restrita, facilitando a troca de informações entre sensores inteligentes, dispositivos acessórios e a nuvem. Permite implementação de forma simplificada e sendo altamente eficiente na comunicação de dados IoT, suportando tanto mensagens de dispositivos para a nuvem quanto da nuvem para dispositivos. Este protocolo é amplamente adotado devido à sua capacidade de operar em condições de rede adversas e sua eficiência em termos de consumo de energia e largura de banda. [16]

Para o funcionamento eficiente do protocolo MQTT é necessário a configuração de um Broker como o ponto central de comunicação de todas as mensagens recebidas por meio deste protocolo. O Broker atuará como um servidor, recebendo todas as mensagens e as encaminhando para os “tópicos” (canais no MQTT para categorizar e direcionar mensagens entre clientes) relevantes. Desta forma, os clientes (qualquer coisa que se comunique com o

Broker) poderão ter acesso apenas as informações dos “tópicos” nos quais estão “inscritos”, ou seja o cliente só irá receber a mensagem de um tópico específico. Este projeto utilizou o broker Mosquitto, que se destaca pela sua simplicidade de implementação e instalação, além de ter um código aberto que permite a visualização sobre sua codificação interna. O Broker configurado conta apenas com um tópico denominado TCC. As mensagens são textos formatados em JSON, e um script para capturar as informações e as cadastrarem diretamente no banco de dados.

QoS (*Quality of Services*) conhecido tambem como Qualidade de Serviço, é um componente importante do protocolo MQTT. Este serviço possui três níveis indicados por números de 0 a 2, onde quanto maior o nível, maior a garantia de entrega. Neste projeto foi utilizado o nível 0 de QoS. Este é o nível mais simples, a mensagem é enviada apenas uma vez e não haverá mais passos, ou seja, a mensagem não será armazenada e nem haverá uma confirmação de recebimento. Foi optado por utilizar este nível, devido a maior velocidade de transmissão. Quando se utiliza o QoS 0 pode ocorrer de dados serem perdidos no envio, porém a não confirmação de entrega possibilita uma rede de baixa latência aumentando a velocidade de envio, ou seja, se um dado for perdido não afetaria o projeto pois outros chegam de forma mais veloz.

3 Resultados e Discussão

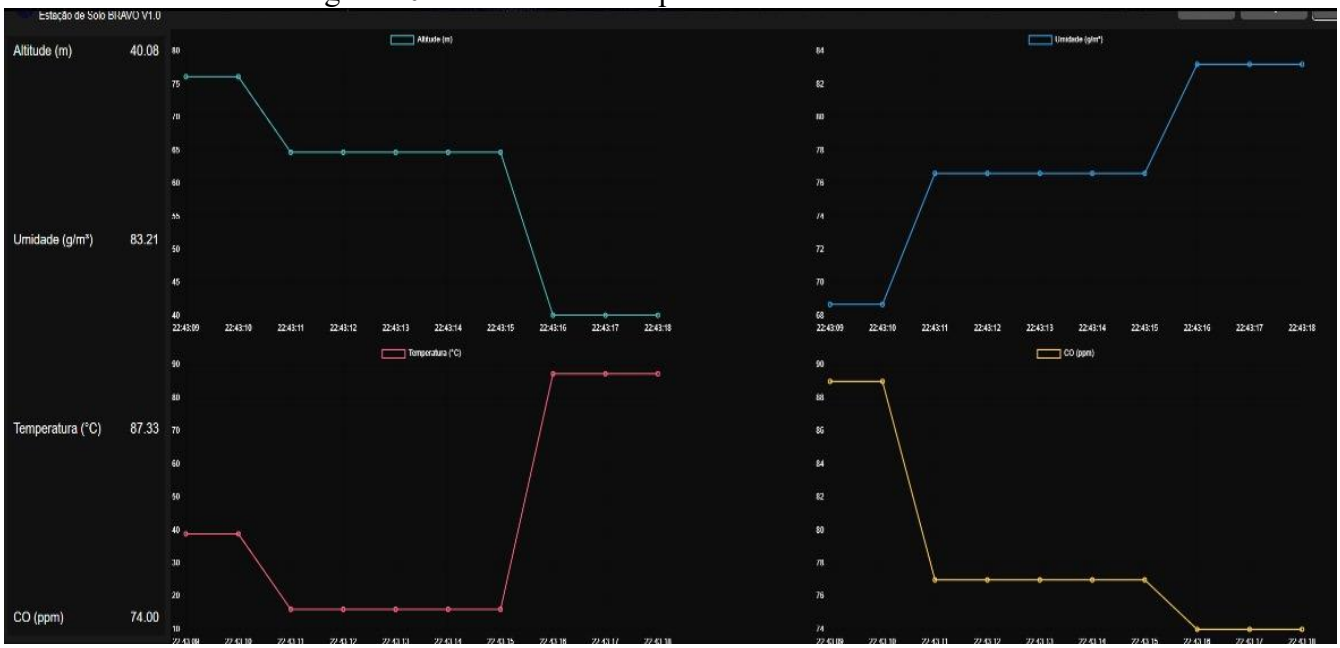
Nesta seção serão exibidos os resultados e as discussões encontradas durante o desenvolvimento deste projeto. Será iniciado pelo desenvolvimento das interfaces, em seguida o hardware desenvolvido, testes e finalizando com a análise de soluções disponíveis no mercado.

3.1 Interface WEB

Para a construção da interface WEB foi utilizado as linguagens JavaScript para o desenvolvimento e atualização dos gráficos e as funcionalidades dos botões. A parte visual do site foi elaborada em HTML (HyperText Markup Language) em conjunto com o CSS (Cascading Style Sheet) com a função de personalizar a aparência. Desta forma a apresentação da página fica visualmente atraente, possibilitando alterar as fontes, tamanho de texto e espaçamento de bordas, permitindo a interface ser de utilização simplificada.

A primeira interface do site consiste na captura e exibição dos dados em tempo real do protótipo de nanossatélite. Após o usuário selecionar o botão "Iniciar Monitoramento" localizado no canto superior direito da tela, o sistema irá começar a coletar e popular os gráficos. Na parte esquerda da tela serão exibidos os últimos dados recebidos pelo protótipo de estação de solo (altitude, umidade, temperatura e monóxido de carbono). A parte direita da tela possui os gráficos com as informações recebidas em tempo real, apresentando os diferentes dados recebidos ao longo do tempo. As informações exibidas na tela estão sendo extraídos do banco de dados (InfluxBD) por meio do *backend* programado em Python. A tela apresenta dois botões para direcionar o usuário para a próxima página desejada (Simulação, Resultados). A interface de coleta de dados em tempo real pode ser visualizada na Figura 10.

Figura 10 - Interface inicial para coleta de dados



Fonte: Autor

A interface da simulação 3D, exibe a maquete de um satélite, construído utilizando a ferramenta Three.js. Logo abaixo é apresentado os dados coletados pelos sensores de acelerômetro (aceleração x, aceleração y, aceleração z), e giroscópio (giro x, giro y e giro z) em tempo real. A maquete irá se mexer de acordo com os dados recebidos, para que a equipe consiga ver como foi o voo do lançamento de seu nanossátelite. A Figura 11 apresenta a Interface de Simulação 3D.

Figura 11 – Interface de Simulação 3D inicial

Fonte – Autor

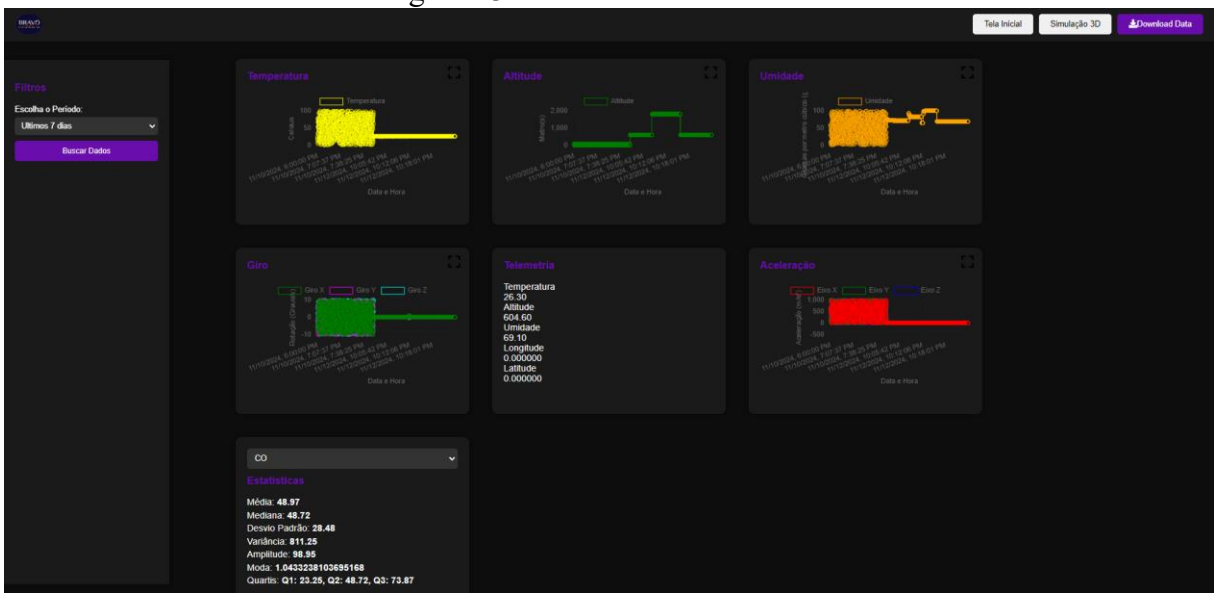
Esta interface também possibilita a visualização dos dados de lançamentos passados por meio dos filtros disponíveis que irão permitir o usuário customizar de qual datas os dados serão exibidos como pode ser observado na Figura 12

Figura 12 – Interface de Simulação com filtro customizado

Fonte - Autor

A Interface de resultados apresentada na Figura 13, tem o objetivo de mostrar uma visão geral sobre todos os dados coletados durante um lançamento ou sobre decolagens passadas através dos filtros selecionados na parte esquerda da tela. Esta tela possui os gráficos da primeira interface com a adição dos gráficos do giroscópio, do acelerómetro e das análises dos dados. Os gráficos possuem a função de expandir quando selecionados por um usuário para facilitar a leitura dos dados, além de serem interativos com click do mouse. O campo "Telemetria" exibe os últimos dados coletados pela estação de solo. Na parte "Estatísticas" possui uma *comboBox* com a opção do usuário de selecionar os dados da concentração de CO, temperatura e umidade, e com isso irá exibir: média, mediana, desvio padrão, variância, amplitude, moda e quartis; dos dados selecionados

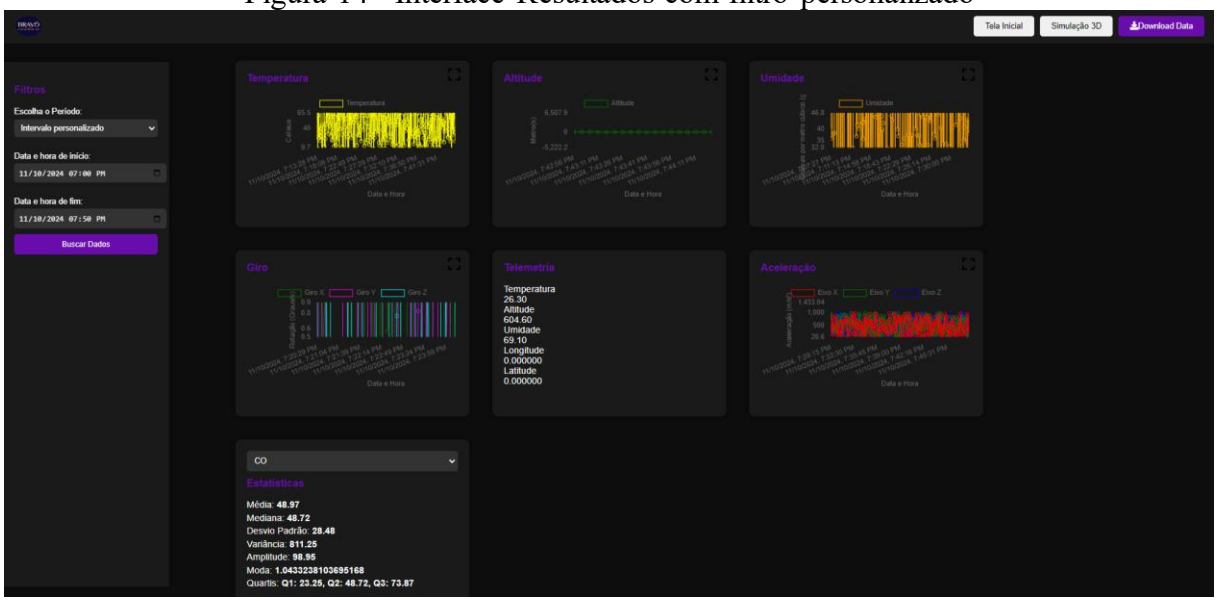
Figura 13 – Interface de Resultados



Fonte – Autor

Esta interface conta também com filtros para a visualização de lançamentos passados na parte direita, estes filtros podem variar de 7 dias, 30 dias ou até para filtros com data e hora personalizado pelo usuário, como pode ser observado na Figura 14.

Figura 14 - Interface Resultados com filtro personalizado

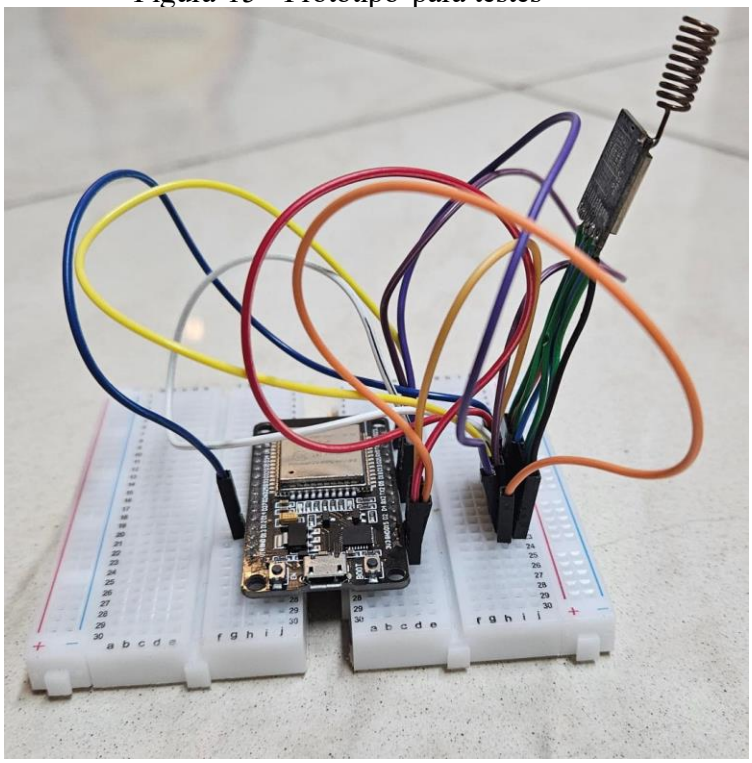


Fonte: Autor

3.2 Hardware desenvolvido

Durante os testes iniciais do projeto, o ESP32 LoRa queimou, e começou a apresentar dados corrompidos no monitor serial. Para que isso não atrasasse o prazo do projeto o microcontrolador foi substituído por um ESP32 normal e o modulo LoRa passou a ser um modelo: LoRa1276-915, da marca: G-NiceRF. Este módulo foi escolhido por estar disponível no laboratório da BRAVO. O protótipo pode ser observado na figura 15

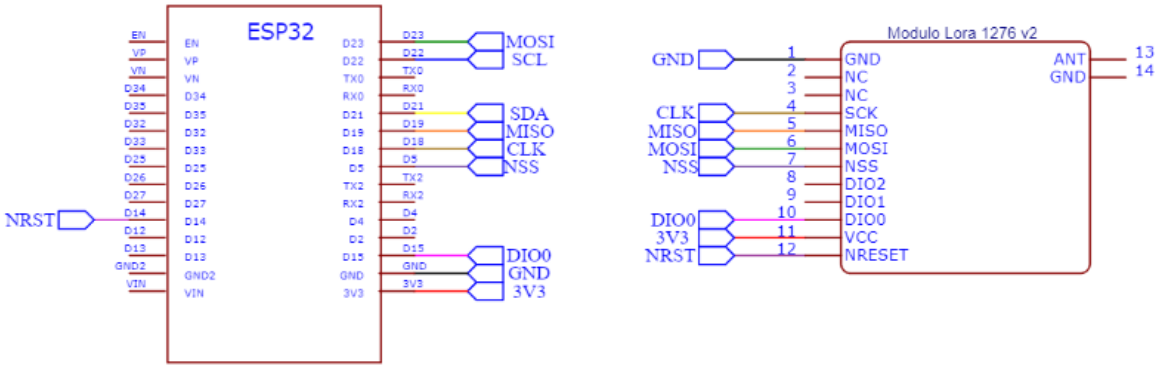
Figura 15 - Protótipo para testes



Fonte: Autor

Para que o módulo LoRa pudesse fornecer os dados para o ESP32, foi necessário utilizar cabos para conectar as entradas do LoRa nas portas seriais corretas do ESP32, para facilitar a visualização das conexões elétricas foi feito o diagrama elétrico deste circuito apresentado na Figura 16.

Figura 16 - Esquema eletrônico para o protótipo de teste



Fonte: Autor

3.3 Testes de bancada

As etapas de testes foram realizadas conforme o desenvolvimento do protótipo, buscando a identificação de algum erro antes de passar para a próxima etapa. Durante os testes finais da comunicação da estação de solo com o nanossatélite, foi notado que os aparelhos demoravam demasiadamente para iniciar a comunicação entre eles. Os aparelhos tinham que ficar em média 30 minutos ligados para haver a comunicação. Após uma investigação foi determinado que o motivo disso era o delay que estava programado no Esp32, quando foi configurado um delay menor, ambos começaram a se comunicar de prontidão.

Também foi realizado o teste de distância da transmissão, onde a estação de solo ficou parada em um lugar, enquanto o nanossatélite foi se distanciando da estação. Foram realizadas paradas para confirmar o recebimento nas distancias de 100 metros; 500m; 1km; 1,5km; 1,8km. Estes testes foram realizados em um ambiente urbano com interferência da infraestrutura e a distância máxima de transmissão foi de 1.8km, após isso os dados começaram a chegar corrompidos para a estação.

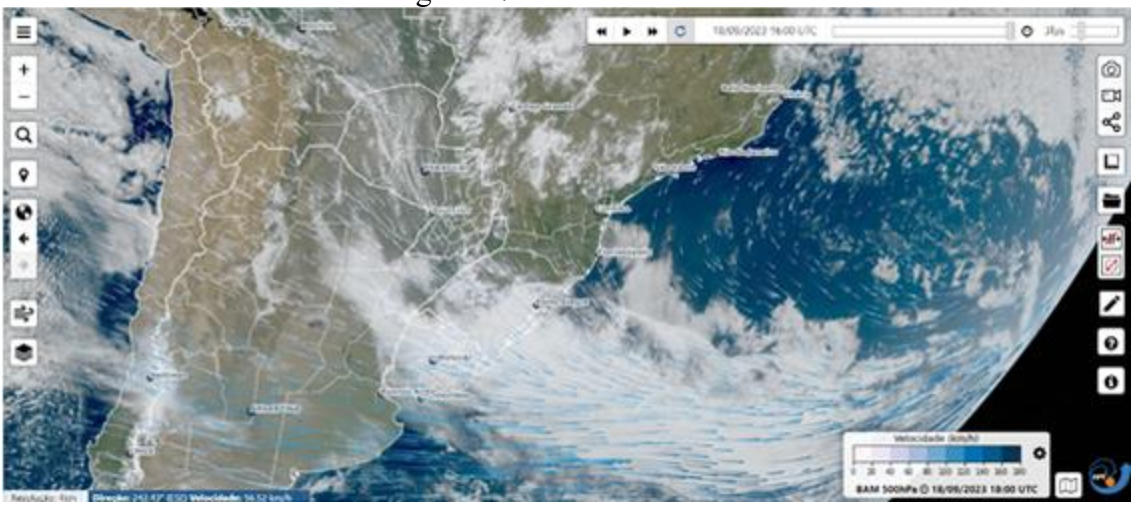
3.4 Soluções existentes no Mercado

Foi realizada uma pesquisa de mercado para identificar aplicações semelhantes ao projeto desenvolvido. Observou-se que atualmente não existem soluções comerciais disponíveis focadas em exibir e extrair dados de nanossatélites em tempo real. No entanto, existem algumas

aplicações acessíveis ao público para a visualização de dados de satélites em órbita, com a maioria delas voltada para imagens e monitoramento ambiental. Um exemplo é o DSAT (Aplicação para visualização interativa dos dados recebidos pela estação GOES-R do INPE), desenvolvido pelo INPE (Instituto Nacional de Pesquisas Espaciais), e outro é o aplicativo CAMALIOT, desenvolvido pela IIASA (*International Institute for Applied Systems Analysis*) em parceria com a ETH Zurich.

O DSAT é uma plataforma que auxilia no monitoramento ambiental quase em tempo real, fornecendo dados atualizados sobre sistemas precipitantes, condições atmosféricas e fenômenos meteorológicos. A interface gráfica do DSAT permite que os usuários visualizem todos os 16 canais espectrais do sensor ABI/GOES (*Geostationary Operational Environmental Satellite*), proporcionando uma visão detalhada de fenômenos meteorológicos, como tempestades e frentes frias. Esses dados são atualizados com uma frequência de 10 minutos, o que facilita o acompanhamento constante das mudanças atmosféricas. Essa interface vista é apresentada na Figura 17. [19]

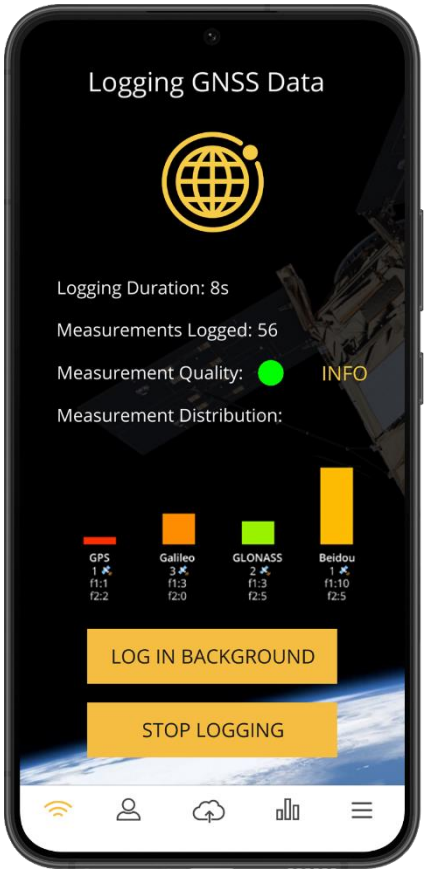
Figura 17 - Interface DSAT



Fonte – DSAT/INPE

Por outro lado, o aplicativo CAMALIOT, disponível na Google Play Store, é uma ferramenta voltada para o acesso e coleta de dados do sistema GNSS (Global Navigation Satellite System). Ele permite que os usuários obtenham informações sobre a localização atual e enviem esses dados para a instituição desenvolvedora, contribuindo para o desenvolvimento de modelos aprimorados de previsão climática e para o aumento da precisão dos sistemas de GPS em dispositivos móveis. Um exemplo da interface do aplicativo é apresentado na Figura 18. [20]

Figura 18 - Aplicativo Camaliot



Fonte – Camaliot.org

Ao comparar essas plataformas com a aplicação desenvolvida neste projeto, observa-se uma clara diferença em termos de funcionalidades. Não foram encontradas soluções disponíveis no mercado que ofereçam funcionalidades semelhantes às apresentadas no projeto desenvolvido neste trabalho. O sistema desenvolvido é exclusivo para a equipe BRAVO e foi projetado para exibir dados em tempo real do nanossatélite que será lançado na LASC. Além disso, a aplicação inclui a funcionalidade de exportação e armazenamento de dados em um banco de dados, permitindo que as informações coletadas sejam analisadas e comparadas com dados de lançamentos futuros.

4 Conclusão

Neste projeto, foi desenvolvido um protótipo de uma estação de solo para um protótipo de nanossatélites utilizando um ESP32 com tecnologia LoRa, implementando o protocolo MQTT para a transmissão dos dados recebidos para um computador. Essa estrutura permite que os dados capturados sejam armazenados em um banco de dados, elemento essencial para o funcionamento do projeto. As análises realizadas pela plataforma desenvolvida e a exibição dos dados dependem diretamente dessa funcionalidade, tornando a comunicação em tempo real um pilar do sistema. A interface simplificada e a capacidade de exportar e armazenar dados proporcionam uma solução prática e funcional para análises futuras, contribuindo para a eficiência do trabalho da equipe BRAVO.

O resultado foi um protótipo de estação de solo que permite a exibição de dados em tempo real de um protótipo de nanossatélite em uma interface web, com visualizações e métricas dos dados. Este projeto não apenas será utilizada na competição LASC, mas também se mostra versátil para testes de nanossatélites e análise de dados passados. Assim, o projeto entrega uma solução eficiente e adaptável, que apoia tanto o monitoramento de missões futuras quanto a exploração de dados anteriores, fortalecendo a posição da equipe BRAVO em suas iniciativas de pesquisa e inovação.

5 Referências

[1] CAMPS, Adriano. Nanosatellites and applications to commercial and scientific missions. Satell. Mission. Technol. Geosci, p. 145-169, 2020.

[2] LIDDLE, J. Douglas et al. Space science with CubeSats and nanosatellites. Nature Astronomy, v. 4, n. 11, p. 1026-1030, 2020.

[3] ABRASAT. Nano Satélites entram no radar do Brasil e do mundo. Disponível em: <https://abrasat.org.br/2021/03/05/nano-satelites-entram-no-radar-do-brasil-e-do-mundo/>. Acesso em: 30 maio. 2024.

[4] DE CARVALHO, Manoel Jozeane Mafra et al. CONASAT-constelaçao de nano satélites para coleta de dados ambientais. XVI Simpósio Brasileiro de Sensoriamento Remoto, p. 9108-9115, 2013.

[5] Nanossatélites Educacionais. Disponível em: <https://www.gov.br/aeb/pt-br/acoes-e-programas/aeb-escola-1/satelites-educacionais>. Acesso em: 30 maio. 2024.

[6] TEIXEIRA, G. ESP32 LoRa WiFi SX1278 Projeto Prático. Disponível em: <https://www.usinainfo.com.br/blog/esp32-lora-wifi-sx1278/>.

[7] ESP32-WROOM-32 Datasheet. [s.l: s.n.]. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>.

[8] VAZ, Lucas Saavedra. Estudo Experimental de um Sistema de Comunicação LoRa para Aplicação em Satélites. 2023.

[9] NASA Opens New CubeSat Opportunities for Low-Cost Space Exploration - NASA. Disponível em: <https://www.nasa.gov/news-release/nasa-opens-new-cubesat-opportunities-for-low-cost-space-exploration/>. Acesso em: 2 jun. 2024.

[10] O que é um Satélite (natural e artificial). Disponível em: <https://www.significados.com.br/satelite/>. Acesso em: 1 jun. 2024.

[11] LANIUS, Roger D.; LOGSDON, John M.; SMITH, Robert W. (Ed.). Reconsidering Sputnik: Forty years since the Soviet satellite. Routledge, 2013.

[12] ZAVIÁLOVA, V. 7 curiosidades sobre o Sputnik, o primeiro satélite artificial da Terra. Disponível em: <https://br.rbth.com/ciencia/79226-7-curiosidades-sobre-sputnik>. Acesso em: 2 jun. 2024.

[13] AEB explica importância dos satélites e uso científico para a humanidade. Disponível em: <https://www.gov.br/mcti/pt-br/acompanhe-o-mcti/noticias/2020/10/aeb-explica-importancia-dos-satelites-e-uso-cientifico-para-a-humanidade>.

[14] LIMA, JS dos S. et al. Documento de Requisitos Preliminares-Fase A (DRP). 2012.

[15] O que é API? Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>.

[16] O que é MQTT? – Explicação sobre o protocolo MQTT – AWS. Disponível em: <https://aws.amazon.com/pt/what-is/mqtt/#:~:text=O%20protocolo%20MQTT%20define%20um>.

[17] O que é o módulo Lora? Para que serve o módulo Lora? Disponível em: <https://www.mokolora.com/pt/what-is-a-lora-module/>.

[18] What is LoRaWAN® Specification - LoRa Alliance®. Disponível em: <https://lora-alliance.org/about-lorawan-old/>.

[19] DSAT, aplicação desenvolvida pelo INPE, é finalista no 27o Concurso Inovação no Setor Público 2023. Disponível em: <https://www.gov.br/inpe/pt-br/assuntos/ultimas-noticias/dsat-aplicacao-desenvolvida-pelo-inpe-e-finalista-no-27o-concurso-inovacao-no-setor-publico-2023>.

[20] CAMALIOT. Disponível em: <https://camaliot.org/>.

[21] CLEMENTE, P. Python: Desenvolvendo Aplicações Web com Flask. Disponível em: <https://blog.rocketseat.com.br/desenvolvendo-aplicacoes-web-com-flask/>.

[22] Aplicação exemplo: driver MQTT em comunicação com broker Mosquitto. Disponível em: <https://kb.elipse.com.br/aplicacao-exemplo-driver-mqtt-em-comunicacao-com-broker-mosquitto-mqtt/>.