Departamento de Engenharia Elétrica e de Computação
SEL0375 – Engenharia de Software
Prof. Ivan Nunes da Silva

Engenharia de Software

Entrega final

Pedro Zarzenon Jacomassi	N° USP 12547472
Victor Marcelo Riverete Monteiro Padial	N° USP 12547444
Ricardo Almeida de Souza Albuquerque	Nº USP 12547207
Maurício Garcia Di Mase	N° USP 12547152
Giovane Rodrigo Cavalcanti Moretto	Nº USP 12624252

1. Análise de Sistemas

1.1. Análise de Necessidade

O número de equipes universitárias dedicadas ao desenvolvimento de foguetes e satélites para competições tem crescido expressivamente no Brasil nos últimos anos, impulsionado pela nova corrida espacial e pelo interesse em tecnologias para viagens espaciais e transporte ultrarrápido na Terra. Com isso em mente, o projeto visa o auxílio de tais equipes na análise em tempo real de seus foguetes, de forma a dinamizar a interpretação dos dados essenciais para o sucesso do lançamento e sua recuperação. Existem opções Open Source com funções gráficas generalizadas para diferentes projetos aeroespaciais como foguetes, drones e satélites de equipes e empresas, porém o software a ser desenvolvido se adequa às necessidades específicas encontradas por inúmeras equipes das competições universitárias e similares.

1.2 Metas Globais do Sistema

O objetivo do software será receber, processar e exibir dados aerodinâmicos de um foguete de pequeno porte, transmitindo-os por telemetria a um terminal para serem visualizados pelo usuário em tempo real, auxiliando na análise e localização do foguete durante e após seu voo. O público alvo são equipes de competição de foguetes universitárias, que desenvolvem o próprio equipamento e frequentemente não têm os recursos disponíveis para realizar o monitoramento em tempo real dos dados de voo de maneira gráfica.

1.3 Estudo da Viabilidade

O software a ser criado será de uso específico, sendo voltado somente para aquisição e transmissão de dados coletados de aeronaves de pequeno porte, e portanto demandará baixos recursos técnicos e econômicos, garantindo-se sua viabilidade.

1.4 Análise Econômica

Para o projeto, será utilizado hardware de baixo custo que demanda pouca manutenção, como ESP32 e Arduino. Os demais equipamentos, como sensores e rádio transmissor, já se encontram integrados às aeronaves das equipes competidoras e poderão ser aproveitados para o software, minimizando seu custo de implementação.

1.5 Diagrama de Arquitetura

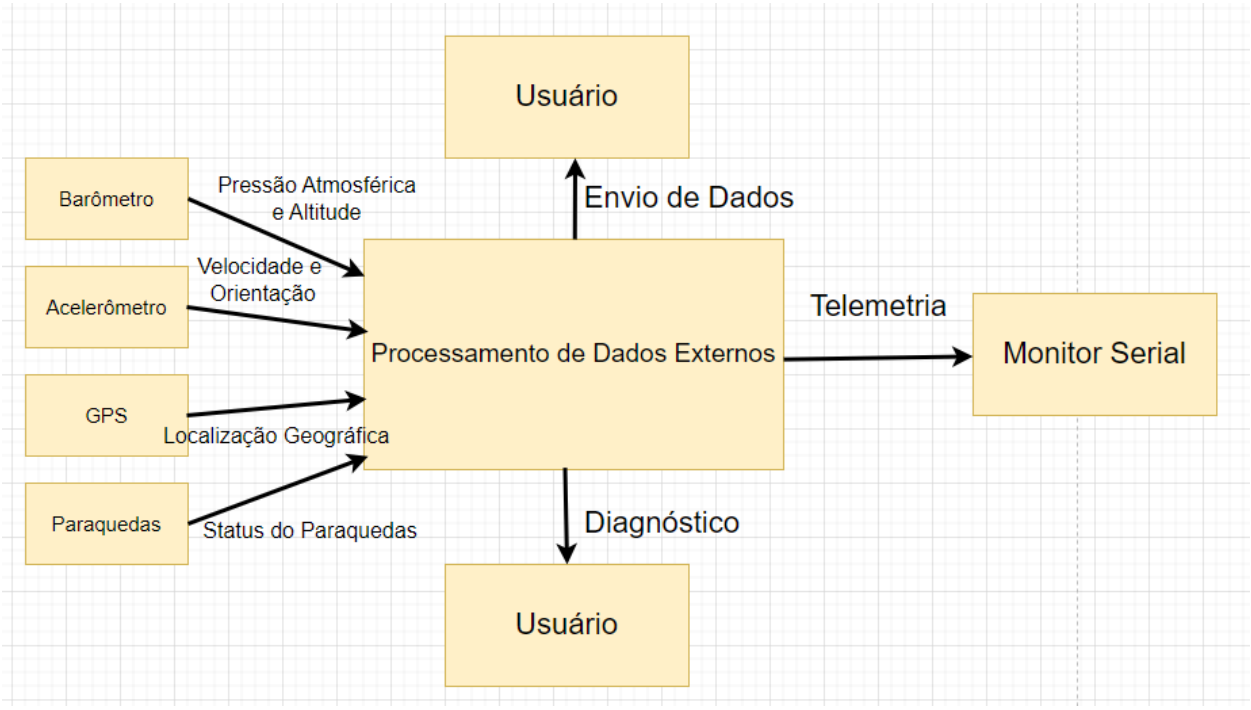

Figura 1: Diagrama de Arquitetura

2. Análise de Requisitos - ERS

2.1 Introdução

2.1.1. Propósito

Este documento tem como objetivo reunir as informações necessárias para o correto uso do software.

2.1.2. Escopo do Produto

O software é destinado a grupos estudantis de competição de aeronaves de pequeno porte de forma que possam monitorar os dados de voo de modo confiável com uma interface intuitiva.

2.1.3 Definições, Siglas e Abreviações

- **Terminal Serial**: Ferramenta de comunicação que permite enviar e receber dados através de uma interface serial, comum em dispositivos de hardware.
- **Monitor Serial**: Programa que exibe dados enviados e recebidos via interface serial, utilizado para depurar e monitorar sistemas embarcados.
- **LoRa**: Tecnologia de comunicação sem fio de longo alcance e baixa potência, usada principalmente em redes IoT.
- **ESP32 e Arduino**: ESP32 é um microcontrolador com conectividade Wi-Fi e Bluetooth, enquanto Arduino é uma plataforma de hardware e software open-source para prototipagem eletrônica.
- **Python**: Linguagem de programação de alto nível, versátil, amplamente utilizada em diversas áreas, como desenvolvimento web, automação e análise de dados.

2.1.4. Visão Geral

O restante do documento está organizado em “Descrições Gerais”, esclarecendo as funções, restrições e requisitos gerais para correta utilização do software, e “Requisitos Específicos”, que detalha as condições a serem cumpridas para que o software opere de modo otimizado.

2.2 Descrições Gerais

2.2.1. Perspectivas do Produto

O software funcionará aproveitando os equipamentos já existentes no foguete, conectando-os a um hardware de baixo custo e boa confiabilidade de modo a fornecer ao usuário dados de voo captados com alta precisão.

2.2.2. Funções do Produto

Como já mencionado, o principal papel do software é permitir à equipe responsável acompanhar o status de um foguete de pequeno porte a distância a partir do fornecimento de dados aerodinâmicos e outras informações. Tendo isso em mente, o produto final deverá ser capaz de traçar gráficos temporais de altitude e aceleração, informar localização do foguete e dados de pressão atmosférica, fornecer a orientação do foguete e mostrar o status de acionamento do paraquedas.

2.2.3. Características do Usuário

Tendo em mente que o software atuará exibindo informações referentes ao status de um foguete de competição, é necessário que o usuário possua um conhecimento mínimo sobre eletrônica e aerodinâmica, além de ser capaz de fazer a leitura de dados exibidos em um ambiente de sistema operacional.

2.2.4. Restrições Gerais

Mesmo garantindo-se a confiabilidade do software e hardware a ser instalado no foguete, a precisão e exatidão dos dados adquiridos ficarão limitados à precisão e exatidão que podem ser oferecidos pelos sensores instalados pela equipe em sua respectiva aeronave, o que tende a variar de projeto para projeto.

2.2.5. Superposições e Dependências

O software é destinado para computadores dotados de Windows 11. Os requisitos da máquina para instalação deste sistema operacional encontram-se em https://support.microsoft.com/pt-br/windows/requisitos-do-sistema-windows-11-86c11283-ea52-4782-9efd-7674389a7ba3

As bibliotecas de Python que serão utilizadas no projeto são as seguintes:

- AsyncIO - Para lidar com os dados em tempo real e a visualização em tempo real
- Pyserial (built in) - Para realizar a leitura dos dados seriais transmitidos pelo Arduino
- Plotly - Para a visualização de dados no Dashboard.

2.3 Requisitos Específicos

2.3.1. Requisitos Funcionais

a. Entradas:

- Barômetro (Altitude e Pressão)
- Acelerômetro (Orientação e Aceleração)
- GPS
- Status do paraquedas

b. Processamento

Concatenação de todos os dados para uma string a ser transmitido para base em solo via telemetria Lora e enviado para o software via USB que filtra e classifica os dados para sua exibição em tempo real.

c. Saída

- Gráfico temporal da Aceleração
- Gráfico temporal da Altitude
- Informar Medida da Pressão Atmosférica
- Informar Localização
- Orientação do Foguete
- Status do Paraquedas

2.3.2. Requisitos de Interface Externa

2.3.2.1 Interface com o Usuário

Serão exibidos ao usuário dados da telemetria do foguete, sendo eles: pressão atmosférica e altitude, velocidade e orientação, sinalização de abertura do paraquedas e localização geográfica,

2.3.2.2 Interface com o Hardware

Os dados obtidos pelo hardware serão enviados para o software através de um módulo Lora, de alto alcance.

2.3.3. Requisitos de Desempenho

Gráfico temporal da Aceleração	Desempenho	Confiabilidade
Excelente	**X**	
Bom		**X**
Mediano		

Gráfico temporal da Altitude	Desempenho	Confiabilidade
Excelente		**X**
Bom	**X**	
Mediano		

Informar localização	Desempenho	Confiabilidade
Excelente	**X**	**X**
Bom		
Mediano		

Orientação do foguete	Desempenho	Confiabilidade
Excelente		
Bom		**X**
Mediano	**X**	

Medida da Pressão Atmosférica	Desempenho	Confiabilidade
Excelente	**X**	**X**
Bom		
Mediano		

Status do Paraquedas	Desempenho	Confiabilidade
Excelente		
Bom		

Mediano	**X**	**X**

2.3.4. Limitações do Projeto

Sendo um conversor de dados enviados em formato de frases(string) para dados em formato gráfico(interface), o software possui limitações quanto à disposição dos dados na frase a serem interpretados pelo software, requerendo uma ordem pré estabelecida e específica na transmissão para sua interpretação pelo software. Ainda, o software ainda está sujeito à qualidade de transmissão de telemetria, necessitando de lidar com perdas de pacotes parcial ou completa dos dados durante o voo.

3. Diagrama do Fluxo de Dados

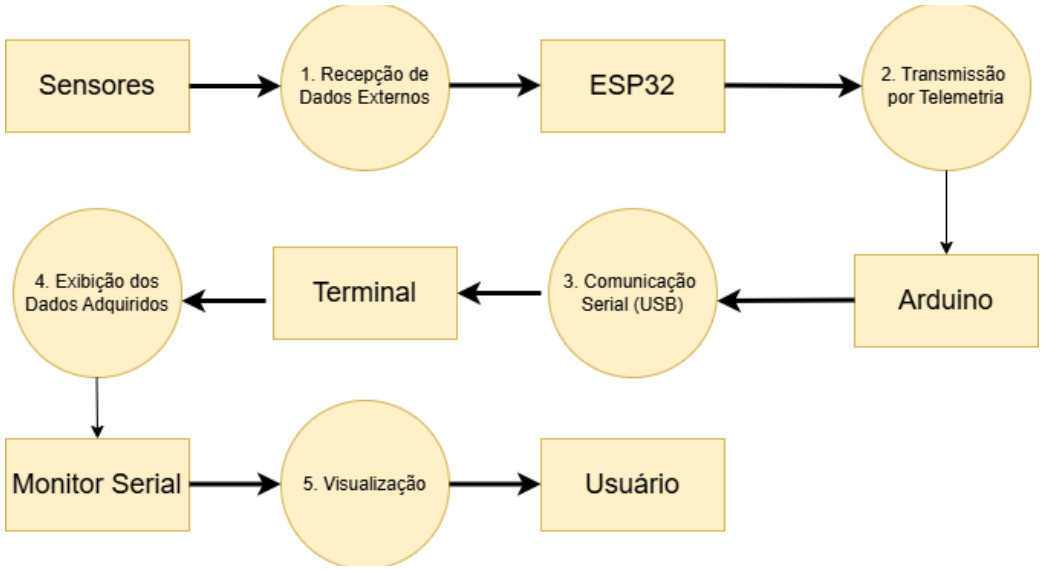

Figura 2: Diagrama de Fluxo de Dados

4. Dicionário de Dados

- foguete estar voando: variável booleana
- altitude: variável tipo float
- localização: vetor numérico tipo float
- rotação: vetor numérico tipo float
- status do paraquedas: variável booleana
- recebendo dados da telemetria: variável booleana
- recebendo dados do terminal: variável booleana
- dados: vetor numérico tipo float
- status instantâneo: vetor numérico tipo float

- gráfico: figura

5. **Especificação de Processos**
    a. **Sensoriamento e envio dos dados por telemetria(Sensores + ESP32 + Lora)**

```
enquanto(foguete estar voando)
Ler e armazenar altitude
Ler e armazenar localização
Ler e armazenar aceleração
Ler e armazenar rotação
Ler e armazenar status do paraquedas
Juntar todos os dados em uma string e enviar por telemetria
fim.enquanto
```

    b. **Recebimento dos dados e envio por serial(Arduino + Lora)**

```
enquanto(recebendo Dados da telemetria)
Ler e armazenar dados recebidos via telemetria
Enviar dados para monitor serial
Enviar dados para terminal via serial
fim.enquanto
```

    c. **Utilização gráfica dos dados da telemetria**

```
enquanto(recebendo dados do terminal)
Ler e armazenar dados recebidos via terminal serial
Plotar gráfico da altitude em relação ao tempo
Plotar gráfico da aceleração em relação ao tempo
Exibir a localização em todos os instantes de tempo
Exibir rotação instantânea do foguete
Exibir status instantâneo
fim.enquanto
```

```
enquanto(recebendo Dados da telemetria)
Ler e armazenar dados recebidos via telemetria
Enviar dados para monitor serial
Enviar dados para terminal via serial
fim.enquanto
```

6. Diagrama de entidade-relacionamento

Cada foguete de competição apresenta uma construção distinta, com sensores que diferem quanto à medida realizada, grau de precisão e valores limites. Portanto, é necessário que o software guarde informações de registro para que possam ser utilizadas configurações personalizadas para cada um dos foguetes.

Como a cada foguete é atribuído um nome, este campo pode ser usado como chave primária. Em adição, cada foguete pertence a uma única equipe e compete em apenas uma categoria, tornando esses campos próprios para atuarem como chaves secundárias.

Nome	Equipe	Categoria
Foguete A	Equipe 1	Categoria X
Foguete B	Equipe 2	Categoria Y
Foguete C	Equipe 1	Categoria Z
(...)	(...)	(...)

Figura 3: Relação entre foguetes, equipes e categorias

Já para o diagrama de entidade-relacionamento em si, é preciso considerar:

- Cada equipe pode registrar um ou mais foguetes
- Um foguete pertence a só uma equipe
- Cada foguete pertence necessariamente a uma, e somente uma categoria
- Uma categoria pode ter ou não um foguete competindo

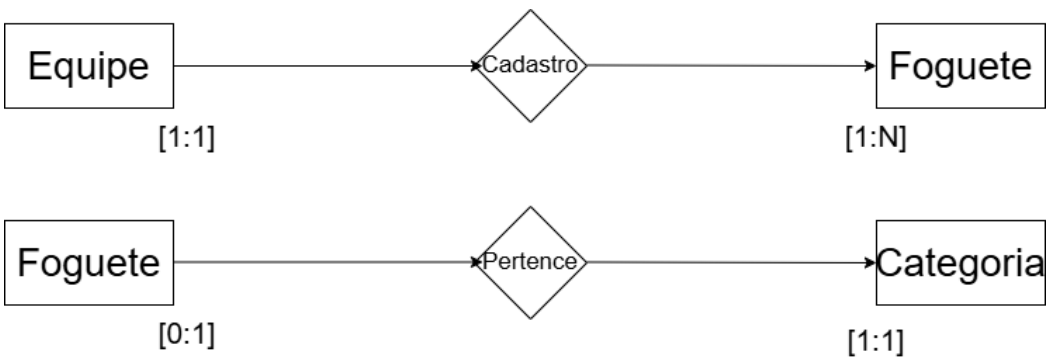

Figura 4: Entidade-relação

7. Codificação e Memorial de Procedimentos

7.1 Parte do Código Responsável pelos Cálculos Necessários

As bibliotecas `numpy` e `pandas` realizam os cálculos e manipulações necessárias, estruturando os dados em matrizes e vetores n-dimensionais. O código também realiza transformações nos dados provenientes do arquivo CSV, como limpeza, conversão de tipos e cálculo de ângulos de orientação. Essas transformações foram implementadas da seguinte forma:

Leitura e Decodificação de Dados:

- Utilizou-se a biblioteca `base64` para decodificar os arquivos enviados pelo usuário.
- Os dados são lidos em formato CSV com a biblioteca `pandas`, que permite organizar as informações em um *DataFrame*.

Tratamento e Validação:

- As colunas foram verificadas para garantir que os dados exigidos pelo sistema estavam presentes.
- Foram aplicadas funções de limpeza para lidar com valores inválidos ou ausentes utilizando `pd.to_numeric` e eliminação de nulos com `dropna`.

Cálculos de Orientação:

- A biblioteca `numpy` foi utilizada para calcular os ângulos de *Pitch*, *Roll* e *Yaw* a partir dos dados do giroscópio, por meio de funções trigonométricas como `arctan2`.

Gerenciamento de Dados Temporais:

- O código ajusta os dados de tempo e os parâmetros físicos como aceleração e altitude, organizando-os em matrizes compatíveis para posterior análise gráfica.

Os cálculos são essenciais para extrair informações úteis e construir visualizações significativas a partir dos dados do foguete.

```
tempo,aceleracao,altitude,pressao,latitude,longitude,gx,gy,gz,paraquedas
0,0.1,100,101325,-23.5489,-46.6388,0.01,-0.02,0.05,FECHADO
1,0.2,102,101310,-23.5485,-46.6387,0.02,-0.03,0.04,FECHADO
2,0.3,104,101295,-23.5481,-46.6386,0.03,-0.04,0.06,ABERTO
3,0.4,106,101280,-23.5477,-46.6385,0.04,-0.05,0.07,ABERTO
4,0.5,108,101265,-23.5473,-46.6384,0.05,-0.06,0.08,FECHADO
5,0.6,110,101250,-23.5469,-46.6383,0.06,-0.07,0.09,FECHADO
6,0.7,112,101235,-23.5465,-46.6382,0.07,-0.08,0.10,ABERTO
7,0.8,114,101220,-23.5461,-46.6381,0.08,-0.09,0.11,FECHADO
8,0.9,116,101205,-23.5457,-46.6380,0.09,-0.10,0.12,FECHADO
9,1.0,118,101190,-23.5453,-46.6379,0.10,-0.11,0.13,ABERTO
```

Figura 5: Exemplo de dados de entrada a serem processados

7.2 Parte do Código Responsável pela Interface

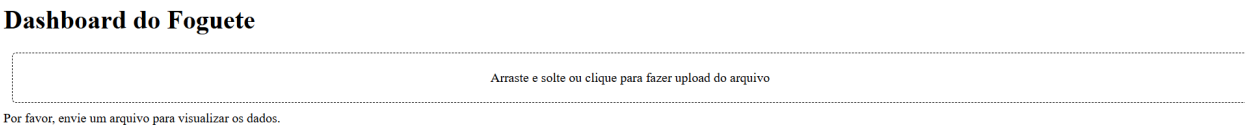

Figura 6: - Tela de carregamento dos dados

Para a criação da interface, foi utilizada a biblioteca `Dash`, que oferece uma plataforma interativa baseada em Python. O sistema permite o envio de arquivos e a exibição de gráficos e mapas interativos em um navegador. As etapas para a implementação da interface foram as seguintes:

Estruturação do Layout:

- A interface foi projetada utilizando elementos do Dash, como `html.Div` para estruturar o layout e `dcc.Upload` para o envio de arquivos.

- Botões, gráficos e mensagens dinâmicas foram organizados em contêineres flexíveis para uma apresentação clara e responsiva.

Interatividade com Callbacks:

- Os `callbacks` do Dash conectam os eventos de interação do usuário (como upload de arquivos) às funções de processamento e renderização dos gráficos.
- A cada upload, o sistema processa os dados enviados e atualiza os gráficos e informações exibidas.

Gráficos Interativos:

- Utilizou-se a biblioteca `plotly.graph_objects` para criar gráficos interativos, como os de aceleração, altitude e orientação (Pitch, Roll e Yaw).
- Foi implementado um mapa interativo com a biblioteca `Scattermapbox`, exibindo a trajetória geográfica baseada nas coordenadas de latitude e longitude.

Mensagens Dinâmicas:

- Informações textuais como pressão atmosférica, localização e orientação do foguete são exibidas de forma dinâmica, refletindo o estado mais recente dos dados.

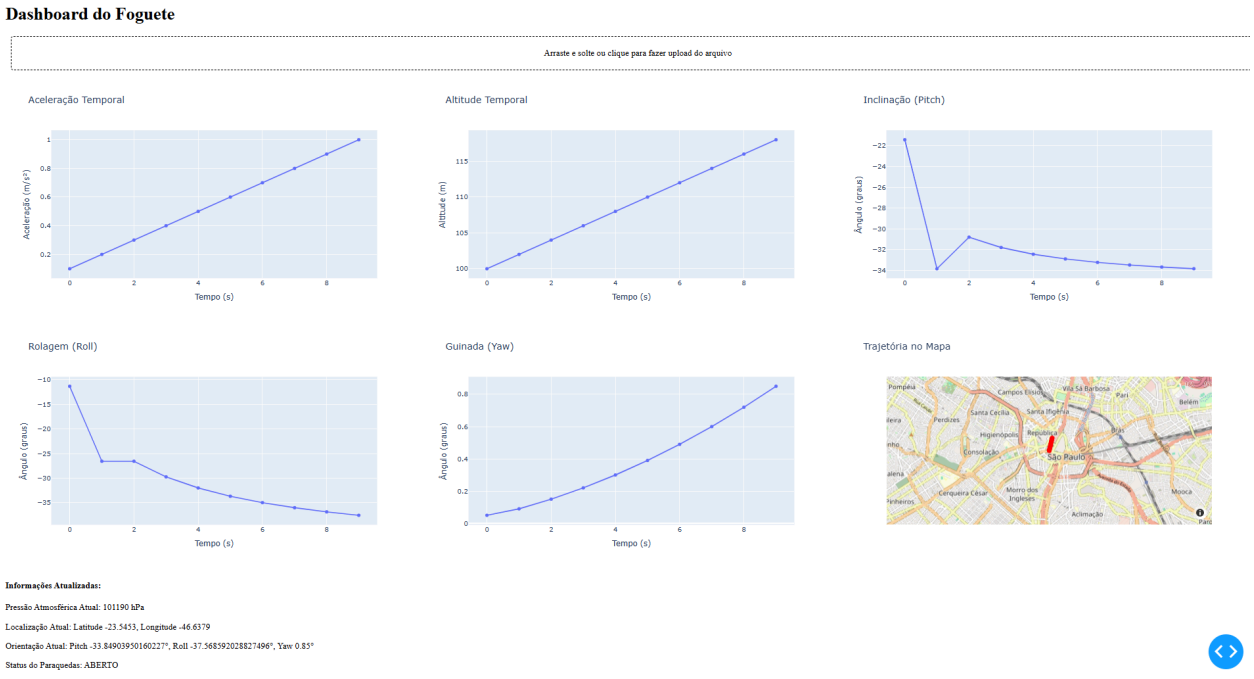

Figura 7: - Dados de voo na dashboard

8. Descrição das Estratégias de Testes Realizados no Software

Nos testes iniciais, serão utilizados *stubs* para garantir que o programa principal funcione como esperado. Será avaliado se o fluxo de entrada e saída de dados entre cada unidade e os demais módulos (representados pelos *stubs*) está conforme o esperado. Ao mesmo tempo, verificações de condições limites e caminhos independentes serão realizadas.

Em seguida, esses stubs serão gradativamente substituídos pelos módulos reais em uma estratégia *depth-first*, assegurando que os dados de sensores são lidos, processados e enviados corretamente por telemetria.

Para os testes de validação, uma versão software com interface será entregue a uma equipe de testes composta por membros de equipes de competição. Nessa etapa, o software deverá ser capaz de atender os requisitos de velocidade, robustez e demais necessidades levantadas durante a etapa de análise. Será também avaliado se o software é capaz de se recuperar de entradas inválidas e outros problemas que podem levar a interrupções indesejadas enquanto cumpre simultaneamente os requisitos já estabelecidos. Com a conexão do software a uma rede restrita, serão realizados também testes de segurança visando garantir que, mesmo após o acesso a internet, seja mantida a integridade dos dados captados e exibidos. Após a conclusão de todos esses passos, será testado o funcionamento do sistema completo, o que inclui o software, ESP32, Arduino, sensores e demais hardwares.

9. Documentação das Falhas Observados nos Testes

As falhas observadas nos testes podem ser dadas por diversos motivos, então, foram listadas os principais:

Falhas ao operar o modelo

- Esse tipo de falha, está relacionada principalmente ao uso de forma errada pelo usuário do software de telemetria, como carregamentos de dados inválidos ou acesso errado ao endereço HTML do software.

Falhas na comunicação entre o foguete o software

- Esse tipo de falha está ligada à comunicação entre o foguete e o software, que pode ter sido interrompida por problemas técnicos no foguete ou software, perda de sinal com a telemetria ou perda de pacotes.

Falhas na leitura de dados

- Esse tipo de falha pode ocorrer quando os dados inseridos no software estão com formatação e disposição divergente das aceitas pelo software

10. Análise dos Fatores e Métricas da Qualidade do Software

Após a realização dos testes, verificou-se que o software cumpriu com o requisito principal de receber e exibir os dados de aceleração e altitude temporais, inclinação, rolagem, guinada (Yaw) e também a trajetória no mapa como mostrado no dashboard da figura 9. A figura 8 mostra uma referência de valores de altitude e aceleração em função do tempo, com as quais se verifica a coerência dos mesmos dados obtidos pelo programa.

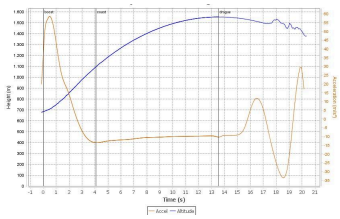

Figura 8: - Referências de aceleração e altitude

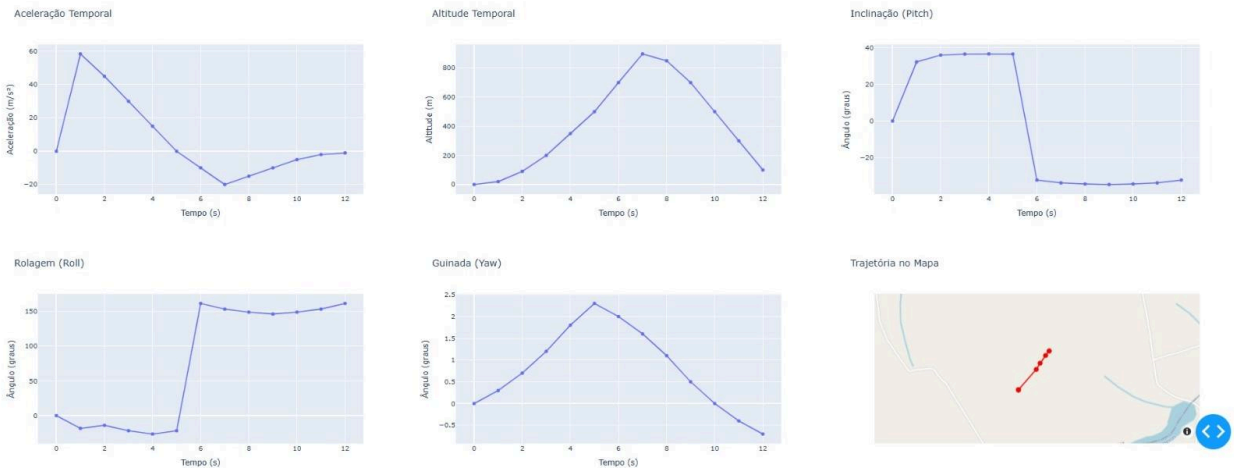

Figura 9: - Dashboard dos dados exibidos ao usuário

Por fim, as bibliotecas `plotly.graph_objects` e `Scattermapbox` se mostraram eficientes na construção dos gráficos interativos e na construção da trajetória geográfica mostrada no mapa.