Universidade Federal do ABC
Pró-reitoria de Pós-graduação da UFABC
Programa de Pós-Graduação em Engenharia Mecânica

Desenvolvimento de Laboratório Virtual: Um Ambiente para o Ensino de Modelagem e Controle de CubeSat

Gabriel Alves Silva

Número de Ordem : XXXX

Santo André - SP, Abril de 2024

Gabriel Alves Silva

Desenvolvimento de Laboratório Virtual: Um Ambiente para o Ensino de Modelagem e Controle de CubeSat

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Mecânica (área de concentração: Dinâmica de Sistemas), como parte dos requisitos necessários para a obtenção do Título de Mestre em Engenharia Mecânica.

Universidade Federal do ABC – UFABC

Pró-reitoria de Pós-graduação da UFABC

Programa de Pós-Graduação em Engenharia Mecânica

Orientador: Diego Paolo Ferruzzo Correa

Santo André - SP

Abril de 2024

Gabriel Alves Silva

Desenvolvimento de Laboratório Virtual: Um Ambiente para o Ensino de Modelagem e Controle de CubeSat/ Gabriel Alves Silva. – Santo André - SP, Abril de 2024-

69 p. : il. (algumas color.) ; 30 cm.

Orientador: Diego Paolo Ferruzzo Correa

Dissertação (Mestrado) – Universidade Federal do ABC – UFABC
Pró-reitoria de Pós-graduação da UFABC
Programa de Pós-Graduação em Engenharia Mecânica, Abril de 2024.

1. CubeSat. 2. Simulação. 3. Determinação e Controle de Atitude. 4. Plataforma Virtual. 5. Código Aberto. I. Diego Paolo Ferruzzo Correa. II. Universidade Federal do ABC. III. Programa de Pós-Graduação em Engenharia Mecânica. IV. Desenvolvimento de Laboratório Virtual: Um Ambiente para o Ensino de Modelagem e Controle de CubeSat.

CDU 02:141:005.7

Gabriel Alves Silva

Desenvolvimento de Laboratório Virtual: Um Ambiente para o Ensino de Modelagem e Controle de CubeSat

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Mecânica (área de concentração: Dinâmica de Sistemas), como parte dos requisitos necessários para a obtenção do Título de Mestre em Engenharia Mecânica.

Trabalho aprovado. Santo André - SP, 01 de janeiro de 2024:

Aos verme que roeu as frias carnes de meu cadáver.

Agradecimentos

Agradeço ao meu orientador, XXXXXXXXX, por todos os conselhos, pela paciência e ajuda nesse período.

Aos meus amigos ...

Aos professores ...

À XXXXXX pelo apoio financeiro para realização deste trabalho de pesquisa.

"Não sei o que,
não sei o que,
não sei o que lá."
(Autor Desconhecido)

Resumo

O presente projeto tem como objetivo principal o desenvolvimento de uma plataforma virtual *open-source* voltada para a simulação da dinâmica de um *CubeSat,* com especial ênfase nos elementos críticos do subsistema de determinação e controle de atitude, proporcionando uma interface gráfica intuitiva. Para alcançar essa meta, pretende-se elaborar roteiros detalhados partindo da instalação e configuração das aplicações computacionais, seguindo para a modelagem físico-matemática do *CubeSat*, sensores e atuadores seguindo *Unified Robotic Description Format (URDF)*, a aplicação de técnicas de determinação e controle de atitude com auxilio de *Python, C++* e *GNU Octave,* assim como a criação de um ambiente espacial virtual *Simulation Description Format (SDF)*, interligados pelo *Robot Operating System (ROS)*, e *Gazebo Simulator.*

Considerando a importância do estudo experimental na engenharia, a plataforma *CubeSat* foi concebida com a meta inicial de facilitar a interação dos estudantes universitários com o espaço e a tecnologia. No entanto, o desenvolvimento e lançamento de um *CubeSat* são tarefas desafiadoras, envolvendo recursos consideráveis. O custo associado, notadamente na casa dos milhões de reais para uma única missão, pode representar uma parcela significativa do orçamento institucional, exemplificado pelos 1% do orçamento total da Universidade Federal do ABC (UFABC) em 2020. Nesse contexto, o projeto propõe uma solução intermediária.

Guiado pela missão de despertar a vocação científica, inserir estudantes em atividades de pesquisa e aprimorar o desempenho dos alunos na engenharia, o projeto alinha-se às transformações educacionais aceleradas pela pandemia de COVID-19. Utilizando aplicações *open-source* para modelagem, programação e simulação, a proposta não apenas visa desenvolver uma ferramenta técnica avançada, mas principalmente cultiva uma abordagem educacional que promove a participação ativa dos estudantes na exploração do espaço e na engenharia aeroespacial.

Espera-se que o projeto, ao atingir os objetivos propostos, resulte em uma plataforma funcional para desenvolvimento colaborativo, contribuindo significativamente para a formação prática dos estudantes de engenharia aeroespacial, ao facilitar a exploração do espaço de maneira acessível e inovadora.

Palavras-chaves: CubeSat. Simulação. Determinação e Controle de Atitude. Plataforma virtual. Código Aberto.

Abstract

This project aims to develop an open-source virtual platform dedicated to simulating the dynamics of a CubeSat, with a particular focus on critical elements of the attitude determination and control subsystem, providing an intuitive graphical interface for this purpose. To achieve this goal, detailed scripts will be devised, encompassing the installation and configuration of computational applications, as well as the physical-mathematical modeling of the CubeSat, sensors, and actuators following the Unified Robotic Description Format (URDF). Additionally, it involves applying attitude determination and control techniques using Python, C++, and GNU Octave, and creating a virtual spatial environment with the Gazebo Simulator in the Simulation Description Format (SDF), interconnected through the Robot Operating System (ROS).

Recognizing the significance of experimental study in engineering, the CubeSat platform was initially designed to facilitate university students' interaction with space and technology. However, the development and launch of a CubeSat pose challenging tasks requiring substantial resources. The associated cost, potentially reaching millions of Brazilian reais for a single mission, can represent a significant portion of institutional budgets, as exemplified by 1% of the total budget of the Federal University of ABC (UFABC) in 2020. In this context, the project proposes an intermediary solution.

Guided by the mission to awaken scientific vocation, engage students in research activities, and enhance their performance in engineering, the project aligns with educational transformations accelerated by the COVID-19 pandemic. Utilizing open-source applications for modeling, programming, and simulation, the proposal not only aims to develop an advanced technical tool but primarily nurtures an educational approach that fosters active student participation in space exploration and aerospace engineering.

The project anticipates yielding a functional platform for collaborative development, significantly contributing to the practical education of aerospace engineering students by facilitating accessible and innovative space exploration.

Keywords: cubeSat. simulation. attitude determination and control. virtual platform. opensource.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADCS	Sistema de Determinação e Controle de Atitude (*Attitude Determination and Control System*)
CGEE	Centro de Gestão e Estudos Estratégicos
CSLI	Iniciativa de lançamento do CubeSat (*CubeSat Launch Initiative*)
EDO	Equação Diferencial Ordinária
IGRF	Campo de Referência Geomagnético Internacional (*International Geomagnetic Reference Field*)
INPE	Instituto Nacional de Pesquisas Espaciais
J2000	Data Juliana em 1 de Janeiro de 2000
JPL	Laboratório de Propulsão a Jato (*Jet Propulsion Laboratory*)
LEO	Órbita Terrestre Baixa (*Low Earth Orbit*)
MarsCo	(*Mars Cube One*)
NASA	Administração Nacional da Aeronáutica e Espaço (*National Aeronautics and Space Administration*)
NEO	Objeto Próximo da Terra (*Near-Earth Object*)
PID	Proporcional Integral Derivativo
REQ	Requisito
ROS	sistema operacional de robôs (*Robot Operating System*)
RPM	Rotações Por Minuto
SCFC	Sistema de Coordenadas Fixo no Corpo
SCGI	Sistema de Coordenadas Geocêntrico-Equatorial Inercial
SCO	Sistema de Coordenadas Orbital]
SDF	Formato de Descrição de Simulação (*Simulation Description Format*)
TRIAD	Determinação de Atitude Triaxial (*TRIaxial Attitude Determination*)
VE	Veículo Espacial
URDF	Formato de Descrição de Robôs Unificado (*Unified Robotics Description Format*)
UT	Horário Universal

Lista de símbolos

$\vec{a}$	Aceleração Linear
B	Base Ortonormal do SCFC
$\vec{b}$	Versor da Base B
C_j	Matriz de Cossenos Diretores para uma Rotação Elementar sobre o eixo j
$C^{B/A}$	Matriz de Transformação de Coordenadas de A para B
E	Matriz Antissimétrica do Eixo de Euler
$\vec{e}$	Eixo de Euler
$\vec{e}_r$	Eixo Radial
$\vec{e}_s$	Eixo Transversal
e_j	componente no eixo j do eixo de Euler
$\vec{F}$	Força
f	Coeficiente de Viscosidade
G	Constante Gravitacional Universal
$\vec{H}$	Momento Angular da Órbita
H_j	Componente j do Momento Angular
I	Base Ortonormal do SCGI; Matriz Identidade
$\vec{i}$	Versor da Base I
J	Matriz de Inercia, Momento de Inercia
K_{cr}	Ganho Crítico
K_P	Ganho Proporcional
m	Massa
N	Torque
N_{em}	Torque Eletromagnético
N_0	Torque Eletromagnético máximo
N_c	Torque de Coulomb
O	Base Ortonormal do SCO

$\vec{o}$	Versor da Base O
P	Período
P_{cr}	Período Crítico
Q	Matriz Antissimétrica do Quatérnio
q	Quatérnio
R	Matriz de Rotação, Raio
$\vec{r}$	vetor distância
r	Razão entre s e s_{max}
s	Velocidade de Rotação
s_{max}	Velocidade de Rotação Máxima
$\vec{v}$	Velocidade Linear
T	Torque
T_i	Período Integrativo
T_d	Período Derivativo
t	tempo
X_{ct}	Ciclo de Trabalho
α	Aceleração Angular
θ	Ângulo, Arfagem
μ	Parâmetro Gravitacional
ρ	Dimensão em um Eixo
Υ	Equinócio Vernal
ϕ	Rolamento
ψ	Guinada
$\vec{\omega}$	Velocidade angular
ω_0	Movimento Médio da órbita
$\vec{\omega}_{ib}^{b}$	Velocidade Angular de SCFC em relação ao SCGI visto no SCFC
$\vec{\omega}_{io}^{o}$	Velocidade Angular de SCO em relação ao SCGI visto no SCO
$\vec{\omega}_{ob}^{b}$	Velocidade Angular de SCFC em relação ao SCO visto no SCFC

Sumário

Introdução

A plataforma CubeSat emergiu como uma alternativa valiosa para países em desenvolvimento ampliarem seu acesso ao espaço. No caso do Brasil, desde 2014, a Agência Espacial Brasileira (AEB) tem colaborado com institutos de pesquisa e universidades na utilização de CubeSats, (**??**). Diferentemente do contexto dos Estados Unidos, onde os CubeSats são frequentemente empregados como projetos estudantis, no Brasil, o desenvolvimento desses satélites muitas vezes se torna uma parte significativa do programa espacial do país.

No entanto, o desenvolvimento e lançamento de um CubeSat não são tarefas trivialmente acessíveis, demandando recursos consideráveis. O custo associado, aproximadamente R$ 2.921.129,50 para uma única missão (**??**), pode representar uma parcela significativa do orçamento de uma instituição de ensino, como exemplificado pelos 1% do orçamento total da Universidade Federal do ABC (UFABC) em 2020 (**??**).

Essa realidade destaca a dificuldade em atingir os objetivos iniciais dos CubeSats, que incluíam proporcionar aos estudantes universitários a oportunidade de interagir com o espaço e a tecnologia relacionada. Diante desse cenário, surge a necessidade premente de encontrar uma solução intermediária que permita o treinamento, ensino e experimentação de forma eficaz para estudantes de engenharia aeroespacial e áreas afins.

Uma alternativa promissora para preencher essa lacuna prática é a utilização de aplicações de modelagem, programação e simulação. Contudo, a escolha entre softwares proprietários e open-source torna-se crucial, considerando os custos e a praticidade. Para contornar essas questões, exemplos de sucesso, como a Robotic Academy no ensino de robótica, demonstram a viabilidade de plataformas web de educação gratuita e aberta que facilitam o aprendizado prático (**??**).

Nesse contexto, o presente projeto de pesquisa propõe a criação de um laboratório virtual utilizando ferramentas open-source. Este laboratório abrangerá modelagem, programação, controle e simulação de sensores, ambiente espacial e atuadores, focando especificamente na plataforma CubeSat.

Além disso, o cenário global atual, marcado pela pandemia de COVID-19, acelerou a adoção de métodos de ensino remoto. Estratégias como o home office e o ensino a distância redefiniram as fronteiras entre o mundo digital e o real, proporcionando uma nova perspectiva para a educação. Este projeto se alinha a essas transformações, oferecendo uma solução educacional híbrida e acessível, que integra a visualização do comportamento mecânico de modelos de CubeSats, experimentação em ambientes virtuais e reais, direcionada a estudantes de engenharia aeroespacial, controle, mecânica, professores e entusiastas (**??**).

Motivação

Em diversas áreas da engenharia, o estudo experimental é fundamental para o desenvolvimento de habilidades práticas essenciais para a resolução eficiente de problemas futuros. Os laboratórios híbridos surgem como uma abordagem acessível, possibilitando

uma interação iterativa entre o mundo real e virtual, local e a distância.

Guiado pela missão de despertar a vocação científica e incentivar novos talentos entre estudantes de graduação, bem como contribuir significativamente para a formação e inserção desses estudantes em atividades de pesquisa, desenvolvimento tecnológico e inovação.

Esta proposta visa incorporar práticas e tecnologias de ensino híbrido para aprimorar o desempenho dos alunos na engenharia, proporcionando um contato prático abrangente que os laboratórios tradicionais muitas vezes não conseguem oferecer.

Não somente ao se desenvolver uma ferramenta técnica avançada, mas principalmente ao se cultivar uma abordagem educacional que promova a participação ativa dos estudantes na exploração do espaço e na engenharia aeroespacial, alinhando-se assim aos princípios fundamentais da formação acadêmica e científica.

Objetivos

Objetivo Geral

- Criar uma plataforma virtual open-source com capacidade de simulação da dinâmica de um CubeSat, enfocando os elementos cruciais do subsistema de determinação e controle de atitude, e proporcionando uma interface gráfica intuitiva.

Objetivos Específicos

- Redigir um Script de Instalação e Configuração das Aplicações Computacionais.
- Criar modelo físico-matemático e 3D de um CubeSat.
- Criar modelo de Sensores, Atuadores.
- Criar algorítimo de determinação e controle de atitude.
- Criar ambiente espacial.
- Criar Simulação do CubeSat em ambiente virtual.
- Redigir roteiros de modelagem e simulação como experimentos.

Parte I

Preparação da pesquisa

1 Uso de referências bibliográficas

O desenvolvimento de um laboratório virtual para o ensino de modelagem e controle de CubeSats é uma tarefa ambiciosa devido à sua extensão, complexidade e natureza multidisciplinar. Requer a integração de tópicos que abrangem desde a mecânica espacial até aplicações de robótica, visualização e simulação, ao mesmo tempo em que mantém um ambiente didático propício ao aprendizado.

O modelo de corpo-rígido em órbita se baseia nas leis de Euler para representação da atitude, sendo eles, matriz de cossenos diretores, ângulos de Euler, ângulo-eixo de Euler e parâmetros de Euler, *i.e.*, quatérnions. Para a visualização da rotação tanto em relação ao corpo quanto em relação a um referencial global são definidos três sistemas de coordenadas seguindo a referência temporal J2000, são eles, o sistema de coordenadas geocêntrico-equatorial inercial, o sistema de coordenadas orbital e o sistema de coordenadas fixo no corpo presentes da literatura **??**), **??**) e **??**).

Por sua vez o modelo dos Sensores Solar e Magnéticos e Rodas de reação em 3-eixos são adaptados da literatura **??**) e (**??**), a simplificação desconsiderando ruídos e abstraindo a aquisição dos vetores de posição do sol e o magnético, o atuador utiliza os conceitos de rotação de massa, torque eletromagnético e atrito presente, são arranjadas nos três eixos principais de inércia e se localizam no centro de massa do corpo. O algoritmo de TRIAD utiliza os valores de posição obtidos anteriormente, seguindo a metodologia proposta por **??**) enquanto a técnica de controle PID é implementada conforme descrito por **??**).

Seguindo a documentação presente em **??**), **??**), **??**) e **??**). A modelagem do veículo espacial, simplificado como um corpo rígido de configuração 6U, é realizada utilizando URDF (Unified Robotics Description Format) contemplando todos os subsistemas, como sensores solares e magnéticos, rodas de reação e câmera. O ambiente espacial, que inclui o veículo espacial e sua órbita, é obtido por meio de simulação computacional com GNU Octave, e esses valores são utilizados em um SDF (Simulation Description Format) para visualização no Gazebo Simulator. Essa integração dos modelos é facilitada pelo ROS (Robotic Operating System), permitindo a comunicação entre os códigos e os modelos.

2 Estado da Arte

Este capítulo fornece uma revisão dos trabalhos relacionados e resume a contribuição da presente dissertação de mestrado para o estado da arte.

A contribuição primária desta dissertação, como delineada na introdução, é o desenvolvimento de uma plataforma open-source para facilitar a aprendizagem de modelagem, simulação e controle de CubeSats.

Atualmente, os laboratórios virtuais para o ensino de engenharia, como robótica, modelagem, simulação e controle, alcançaram um estágio avançado, principalmente devido à adoção de ferramentas open-source como ROS (Robotic Operating System), Gazebo e Octave. O ROS oferece uma infraestrutura flexível e modular para o desenvolvimento de sistemas robóticos, permitindo a integração de diferentes componentes e algoritmos de controle. Por sua vez, o Gazebo é um simulador de robótica 3D que proporciona a criação de ambientes virtuais realistas para testes e experimentação. Complementarmente, o Octave oferece uma poderosa plataforma de computação numérica para análise e implementação de algoritmos de controle.

Essas ferramentas combinadas possibilitam aos estudantes e pesquisadores explorarem de maneira prática e eficaz os conceitos de modelagem, simulação e controle de CubeSats, contribuindo assim para o avanço da educação e pesquisa na área aeroespacial.

No entanto, é importante ressaltar que as ferramentas do ROS e Gazebo não são tão amplamente utilizadas pela comunidade aeroespacial devido à sua limitação em simular nativamente condições como microgravidade e ambientes sem fricção, elementos essenciais para a análise de CubeSats no espaço.

A seguir, serão discutidos alguns trabalhos relacionados ao uso de ferramentas open-source como ROS, Gazebo ou Octave para o ensino de modelagem ou controle de robótica ou CubeSats.

Trabalhos Relacionados

??) abordaram um estudo de caso que analisa os potenciais e as limitações das aplicações de código aberto, destacando a modelagem, simulação e análise de um Motor de Corrente Contínua. De acordo com o artigo, cerca de 90% dos estudantes recorrem ao GNU Octave ou ao Simulador Gazebo, com essa porcentagem reduzida para 70% entre os educadores, reafirmando a preferência da comunidade por essas duas ferramentas abertas. A conclusão dessa primeira parte ressalta que o GNU Octave oferece vantagens, como funções e bibliotecas úteis para controle, modelagem e análise numérica.

??) sintetizaram, em outro estudo de caso, os potenciais e as limitações das aplicações de código aberto na implementação do controle de um manipulador robótico com 2 graus de liberdade. Novamente, o GNU Octave é mencionado por apresentar resultados satisfatórios, mas a falta de um ambiente gráfico para simulações é destacada como uma limitação. Nesse estudo, essa lacuna é superada com o uso do Simulador Gazebo em conjunto com o ROS (Robotic Operating System), embora seja observado que essa ferramenta pode ser menos amigável para iniciantes.

??) descreveram o uso do ROS e Gazebo para a modelagem e simulação de robôs, destacando a utilidade da simulação cinética e dinâmica na avaliação da interação humana com construções robóticas. Embora o artigo se refira principalmente a construções terrestres, o conceito de construção robótica pode ser aplicado também a estruturas como a Estação Espacial Internacional.

Diante do aumento do interesse em manipuladores em órbita, **??**) identificaram uma lacuna nas técnicas atuais de modelagem e controle de manipuladores robóticos em ambiente espacial. Sugeriram, portanto, uma arquitetura e plataforma para a realização de estudos nessa área, apresentando uma arquitetura inteligente de GNC (Guiagem, Navegação e Controle) de espaçonave com componentes de IA de ponta para manipulação em órbita. Para treinar a plataforma, tornou-se necessário o acesso a dados confiáveis, os quais foram identificados como indisponíveis ou de custo elevado. Como solução, desenvolveram um simulador visual utilizando a ferramenta Unreal Engine 4, denominado OrVIS (Orbital Visualization). Para testar o sistema, utilizaram o ROS em conjunto com o Gazebo, ressaltando que, como um teste de bancada, não integrava visualização e dinâmica espacial.

??) também reconheceu o aumento na demanda por manipuladores e, consequentemente, por simulações envolvendo esses dispositivos. Considerando que o Gazebo e o ROS são plataformas consolidadas no mercado, ele propôs automatizar a geração de URDFs (Unified Robotic Description Format), o formato utilizado por essas aplicações, a partir dos graus de liberdade dos manipuladores. O URDF é um XML (extensible Markup Language) utilizado para descrever modelos robóticos em ambientes de simulação, especialmente no contexto do ROS. Esse tipo de arquivo contém informações detalhadas sobre a geometria, cinemática, dinâmica e outras propriedades físicas de um robô, incluindo links, juntas, sensores e suas relações espaciais. Esses arquivos desempenham um papel fundamental na modelagem e simulação de robôs em ambientes virtuais, possibilitando que os desenvolvedores visualizem, controlem e testem seus sistemas robóticos antes de implementá-los no mundo real.

??), é apresentado um projeto de localização e mapeamento simultâneo de um robô móvel utilizando o simulador Gazebo em conjunto com o ROS. É evidente que uma das grandes vantagens desse framework é a diversidade de bibliotecas, plugins e implementações disponíveis para sensores de distância e visualização, odometria, motores e modelagem de robôs.

??) é uma plataforma de acesso aberto projetada para a aprendizagem prática de robótica inteligente em cursos de engenharia. Conhecida como Robotics-Academy, ela compreende uma coleção de 18 exercícios apresentando diversos robôs, como carros e drones. A plataforma utiliza o middleware Robotic Operating System (ROS) e o simulador 3D Gazebo, juntamente com a linguagem de programação Python. Dada a natureza multidisciplinar da robótica, a Robotics Academy enfatiza mais os algoritmos do que o middleware.

Como mencionado anteriormente e aprofundado em **??**), destaca-se o curso prático à distância de drones, com ênfase na programação desses Veículos Aéreos Não Tripulados (VANTs). O hardware, que representa o próprio robô, é completamente simulado pelo Gazebo, enquanto os softwares dos drivers para a leitura das informações dos sensores e o controle dos atuadores são fornecidos pelo ROS. Os estudantes desenvolvem algoritmos a partir de templates disponibilizados. Os exercícios abordam uma variedade de tarefas, como navegação por posição, seguir objetos e pouso em um carro em movimento, entre outros.

Um aspecto que merece destaque na Robotics Academy, conforme abordado por **??**), é que em uma de suas atualizações mais recentes, foram oferecidas ainda mais opções para distribuição e utilização da plataforma. Inicialmente e até hoje, é possível obter a plataforma baixando seu código-fonte pelo GitHub. No entanto, percebeu-se que a instalação e configuração de pacotes extras eram uma barreira. Para superar esse obstáculo, foram apresentadas duas soluções: a instalação via Docker, que já inclui todas as configurações prévias e pode ser utilizada em sistemas Linux, Windows e MacOS; e também, para eliminar qualquer complicação de configuração por parte do usuário, foi disponibilizada a opção de um webserver pronto para uso, onde a computação ocorre de forma transparente para o usuário. Essas soluções visam maximizar a distribuição entre os interessados.

??) propõem várias tarefas de controle espacial para objetos em órbita complexos e com vários graus de liberdade. Apresentam uma framework unificada para simulações de robôs espaciais chamada OnOrbitROS, baseada no ROS. O desenvolvimento e teste de um sistema robótico em seus estágios iniciais são essenciais para reduzir custos, e as simulações estão se tornando cada vez mais comuns para isso. Assim a plataforma OnOrbitROS contribui para a comunidade aeroespacial, que tem sido relutante em adotar o ROS/Gazebo, ao oferecer uma solução que incorpora as qualidades de implementação e teste rápidos que essas ferramentas oferecem.

Esses são apenas alguns exemplos de trabalhos que exploram o uso de ferramentas open-source para o ensino de modelagem e controle de CubeSats. Existem muitos outros estudos e projetos interessantes nessa área, demonstrando o crescente interesse e importância dessas tecnologias na educação e pesquisa espacial.

3 Modelagem Físico Matemática

3.1 Sistemas de Coordenadas e Sistema de Medida de Tempo

É de fundamental importância para qualquer estudo da mecânica a determinação objetiva dos sistemas de referência, sendo eles o de coordenadas e o de tempo. O parâmetro para essa escolha é a facilitação da formulação, visualização e análise dos fenômenos e resultados.

3.1.1 Sistema de Coordenadas Geocêntrico-Equatorial Inercial (SCGI)

Por conta do tempo do fenômeno ser insignificante em relação à variação das direções e origem do sistema de coordenadas, é assumido como hipótese simplificadora que esse sistema é não rotacional e fixo no espaço, *i.e.*, inercial.

O sistema de coordenadas geocêntrico-equatorial inercial I tem como origem o centro de massa da Terra, *i.e.*, geocêntrico, e base composta pelos vetores $\{\vec{i_1}, \vec{i_2}, \vec{i_3}\}$. O eixo $\vec{i_1}$ tem como sentido o equinócio vernal Υ, o eixo $\vec{i_3}$ aponta para a normal do plano (fundamental) equatorial terrestre, *i.e.*, o polo Norte celeste e o eixo $\vec{i_2}$ completa a base ortogonal dextrogiro (**??**, p. 12-13). Mostrado na Figura 1.

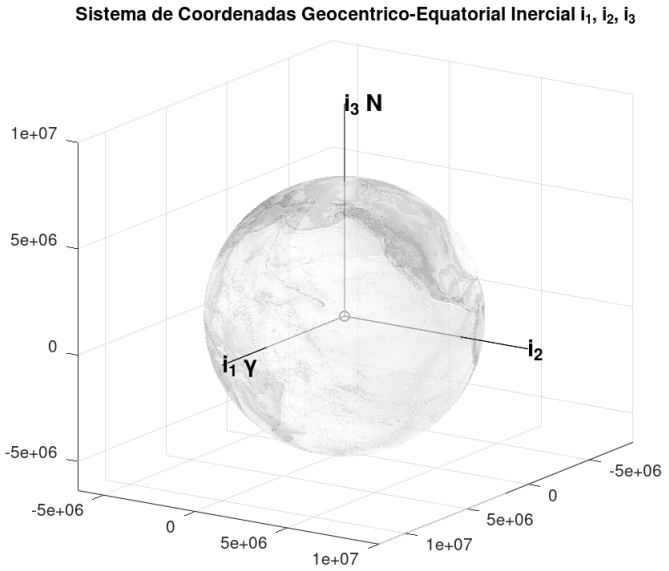


Figura 1 – Sistema de Coordenada Geocêntrico-Equatorial Inercial (SCGI)

3.1.2 Sistema de Coordenadas Orbital (SCO)

O sistema de coordenadas orbital O tem como origem o centro de massa do veículo espacial em órbita e base composta pelos vetores $\{\vec{o_1}, \vec{o_2}, \vec{o_3}\}$. O eixo $\vec{o_3}$ aponta em direção ao centro de massa da Terra ou, direção nominal do nadir. O eixo $\vec{o_2}$ tem como direção o oposto do vetor do plano fundamental de órbita do veiculo espacial (**??**, p. 26-29). E

o eixo $\vec{o}_1$ aponta para o vetor velocidade nominal ou, a direção nominal da trajetória da órbita. Representado na Figura 9.

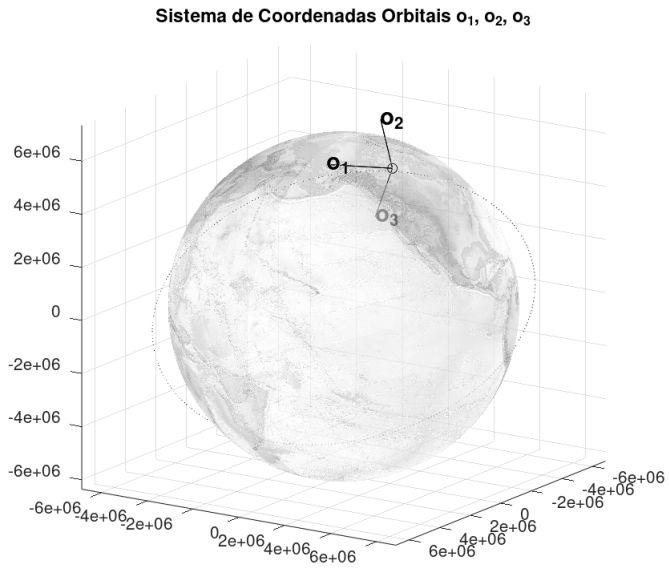


Figura 2 – Sistema de Coordenada Orbital (SCO)

3.1.3 Sistema de Coordenadas Fixo no Corpo (SCFC)

O sistema de coordenadas fixo no corpo, B, tem como origem o centro de massa do veículo espacial, sua base é composta pelos vetores $\{\vec{b}_1, \vec{b}_2, \vec{b}_3\}$, sendo escolhidos arbitrariamente. É interessante alinhar esses eixos com as direções dos eixos principais de inércia. Escolhendo eixo $\vec{b}_3$ como eixo de Spin, *i.e.*, o eixo de menor momento principal de inércia. Os eixos $\vec{b}_2$, $\vec{b}_1$ o segundo e o terceiro menores momentos principais de inércia (**??**, p. 26-29). Como representado na Figura 3.

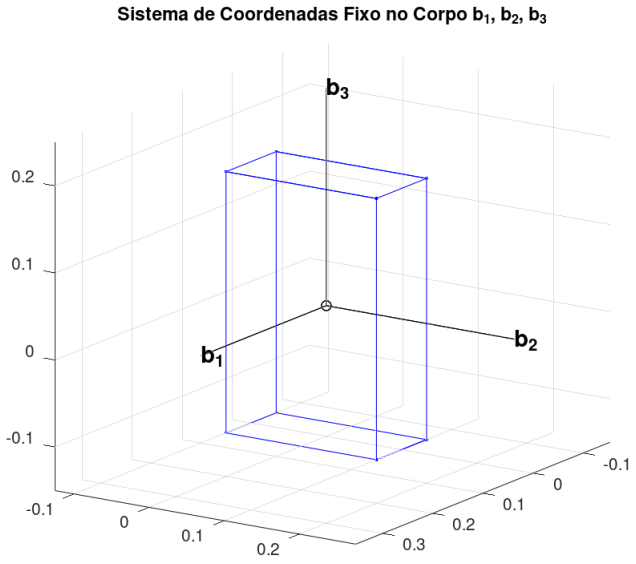


Figura 3 – Sistema de Coordenada Fixo no Corpo (SCFC)

3.1.4 Sistema de Referência de Tempo

A referência para os sistemas de coordenadas descritos acima é o equador médio e o equinócio vernal referente a data Gregoriana de 1º de Janeiro de 2000 às 12 horas U.T. (tempo universal) ou a data Juliana de 2451545. Esse sistema de referência é conhecido como J2000. É a escolha dessa referência que garante que o sistema de coordenadas Geocêntrico-equatorial possa ser considerado não rotacional (**??**).

3.2 Mecânica Rotacional

Nesta seção estuda-se a representação da cinemática rotacional de um corpo rígido. Entende-se que essa formulação é a mesma usada em um modelo de CubeSat. Sendo o principal objetivo descrever a orientação e rotação do corpo.

Existem várias maneiras de representar a atitude de um corpo rígido e a depender do propósito, visualização ou cálculo, umas são mais indicadas que outras. Temos entre elas a matriz de cossenos diretores, os ângulos de Euler, o Eixo-ângulo de Euler e os parâmetros de Euler (Quatérnions).

A formulações a seguir seguem a literatura (**??**, p. 323-486)

3.2.1 Matriz de Cossenos Diretores

Considerando o sistema A e B como dextrogiros e compostos por três vetores unitários ortonormais (versores) cada, sendo respectivamente $A = \{\vec{a_1}, \vec{a_2}, \vec{a_3}\}$ e $B = \{\vec{b_1}, \vec{b_2}, \vec{b_3}\}$.

Pode-se expressar os versores da base B em termos da base A como mostrado abaixo:

$$\begin{bmatrix} \vec{b_1} \\ \vec{b_2} \\ \vec{b_3} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} \vec{a_1} \\ \vec{a_2} \\ \vec{a_3} \end{bmatrix} = C^{B/A} \begin{bmatrix} \vec{a_1} \\ \vec{a_2} \\ \vec{a_3} \end{bmatrix} \tag{3.2.1.1}$$

$C^{B/A}$ é conhecido como a matriz de cossenos diretores. Descreve a orientação de B em relação a A.

Podendo ser escrita da forma a seguir:

$$C^{B/A} = \begin{bmatrix} \vec{b_1} \cdot \vec{a_1} & \vec{b_1} \cdot \vec{a_2} & \vec{b_1} \cdot \vec{a_3} \\ \vec{b_2} \cdot \vec{a_1} & \vec{b_2} \cdot \vec{a_2} & \vec{b_2} \cdot \vec{a_3} \\ \vec{b_3} \cdot \vec{a_1} & \vec{b_3} \cdot \vec{a_2} & \vec{b_3} \cdot \vec{a_3} \end{bmatrix} = \begin{bmatrix} \vec{b_1} \\ \vec{b_2} \\ \vec{b_3} \end{bmatrix} \cdot \begin{bmatrix} \vec{a_1} & \vec{a_2} & \vec{a_3} \end{bmatrix} \tag{3.2.1.2}$$

A matriz de cossenos diretores é conhecida como: matriz de rotação ou matriz de transformação de coordenadas de A para B. Tal transformação de coordenada é representada simbolicamente como $C^{B/A} : B \leftarrow A$

Por simplicidade, muitas vezes representa-se essa matriz simplesmente por C. Algumas propriedades da matriz C são:

$$C^{-1} = C^T \tag{3.2.1.3}$$

Que é equivalente a:

$$CC^T = I = C^T C \tag{3.2.1.4}$$

Isso ocorre pois C é uma matriz ortogonal. Consequentemente tem-se as seguinte relações entre $C^{A/B}$ e $C^{B/A}$:

$$\left[C^{A/B}\right]_{ij} = \vec{a}_i \cdot \vec{b}_j \tag{3.2.1.5}$$

$$\left[C^{B/A}\right]_{ij} = \vec{b}_i \cdot \vec{a}_j \tag{3.2.1.6}$$

$$\left[C^{A/B}\right]^{-1} = \left[C^{A/B}\right]^T = C^{B/A} \tag{3.2.1.7}$$

$$\left[C^{B/A}\right]^{-1} = \left[C^{B/A}\right]^T = C^{A/B} \tag{3.2.1.8}$$

Assim dados dois conjuntos de sistema de coordenadas, A e B, e um vetor arbitrário $\vec{H}$, esse pode ser expresso em termos dos vetores da base A e B:

$$\vec{H} = H_1\vec{a_1} + H_2\vec{a_2} + H_3\vec{a_3} \tag{3.2.1.9}$$

$$= H'_1\vec{b_1} + H'_2\vec{b_2} + H'_3\vec{b_3} \tag{3.2.1.10}$$

De onde se adquire:

$$\begin{bmatrix} H'_1 \\ H'_2 \\ H'_3 \end{bmatrix} = \begin{bmatrix} \vec{b_1}\cdot\vec{a_1} & \vec{b_1}\cdot\vec{a_2} & \vec{b_1}\cdot\vec{a_3} \\ \vec{b_2}\cdot\vec{a_1} & \vec{b_2}\cdot\vec{a_2} & \vec{b_2}\cdot\vec{a_3} \\ \vec{b_3}\cdot\vec{a_1} & \vec{b_3}\cdot\vec{a_2} & \vec{b_3}\cdot\vec{a_3} \end{bmatrix} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} = C^{B/A} \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} \tag{3.2.1.11}$$

Três rotações elementares respectivamente: no primeiro, no segundo e no terceiro eixos, da referência A, são descritas com as matrizes de rotação a seguir:

$$C_1(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_1 & sen\theta_1 \\ 0 & -sen\theta_1 & cos\theta_1 \end{bmatrix} \tag{3.2.1.12}$$

$$C_2(\theta_2) = \begin{bmatrix} cos\theta_2 & 0 & -sen\theta_2 \\ 0 & 1 & 0 \\ sen\theta_2 & 0 & cos\theta_2 \end{bmatrix} \tag{3.2.1.13}$$

$$C_3(\theta_3) = \begin{bmatrix} cos\theta_3 & sen\theta_3 & 0 \\ -sen\theta_3 & cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2.1.14}$$

Onde $C_i(\theta_i)$ denota a direção da matriz de cossenos diretores C para um rotação elementar sobre o eixo i de A com um ângulo de θ_i.

3.2.2 Ângulos de Euler

Existem 12 combinações para os ângulos de Euler. Elas são provindas performando rotações sucessivas sobre os eixos fixos de um corpo, onde cada rotação é efetuada em um eixo diferente daquele imediatamente anterior. Isso garante que é possível adquirir qualquer orientação arbitrária efetuando três rotações sucessivas sobre uma base fixa de referência.

Considere abaixo o exemplo da combinação de três rotações sucessivas nos eixos 321 fixo no corpo, que descrevem a orientação do sistema de referência B em relação a referência A.

Uma sequência particular é descrita abaixo:

$$C_3(\theta_3):\ A' \leftarrow A \tag{3.2.2.1}$$

$$C_2(\theta_2):\ A'' \leftarrow A' \tag{3.2.2.2}$$

$$C_1(\theta_1):\ B \leftarrow A'' \tag{3.2.2.3}$$

E cada rotação pode ser descrita como:

$$\begin{bmatrix}\vec{a_1'}\\ \vec{a_2'}\\ \vec{a_3'}\end{bmatrix} = \begin{bmatrix} cos\theta_3 & sen\theta_3 & 0\\ -sen\theta_3 & cos\theta_3 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\vec{a_1}\\ \vec{a_2}\\ \vec{a_3}\end{bmatrix} = C_3(\theta_3)\begin{bmatrix}\vec{a_1}\\ \vec{a_2}\\ \vec{a_3}\end{bmatrix} \tag{3.2.2.4}$$

$$\begin{bmatrix}\vec{a_1''}\\ \vec{a_2''}\\ \vec{a_3''}\end{bmatrix} = \begin{bmatrix} cos\theta_2 & 0 & -sen\theta_2\\ 0 & 1 & 0\\ sen\theta_2 & 0 & cos\theta_2\end{bmatrix}\begin{bmatrix}\vec{a_1'}\\ \vec{a_2'}\\ \vec{a_3'}\end{bmatrix} = C_2(\theta_2)\begin{bmatrix}\vec{a_1'}\\ \vec{a_2'}\\ \vec{a_3'}\end{bmatrix} \tag{3.2.2.5}$$

$$\begin{bmatrix}\vec{b_1}\\ \vec{b_2}\\ \vec{b_3}\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0\\ 0 & cos\theta_1 & sen\theta_1\\ 0 & -sen\theta_1 & cos\theta_1\end{bmatrix}\begin{bmatrix}\vec{a_1''}\\ \vec{a_2''}\\ \vec{a_3''}\end{bmatrix} = C_1(\theta_1)\begin{bmatrix}\vec{a_1''}\\ \vec{a_2''}\\ \vec{a_3''}\end{bmatrix} \tag{3.2.2.6}$$

A' e A'' são dois sistemas de referência intermediários. Os três ângulos θ_1,θ_2,θ_3 usados para a rotação são os chamados ângulos de Euler.

Combinando a sequência demonstrada acima obtém-se:

$$\begin{bmatrix}\vec{b_1}\\ \vec{b_2}\\ \vec{b_3}\end{bmatrix} = C_1(\theta_1)\begin{bmatrix}\vec{a_1''}\\ \vec{a_2''}\\ \vec{a_3''}\end{bmatrix} = C_1(\theta_1)C_2(\theta_2)\begin{bmatrix}\vec{a_1''}\\ \vec{a_2''}\\ \vec{a_3''}\end{bmatrix} = C_1(\theta_1)C_2(\theta_2)C_3(\theta_3)\begin{bmatrix}\vec{a_1}\\ \vec{a_2}\\ \vec{a_3}\end{bmatrix} \tag{3.2.2.7}$$

Expandindo a matriz de rotação de A para B mostrada acima tem-se:

$$C^{B/A} = C_1(\theta_1)C_2(\theta_2)C_3(\theta_3) = \begin{bmatrix} c_2c_3 & c_2s_3 & -s_2\\ s_1s_2c_3 - c_1s_3 & s_1s_2s_3 + c_1c_3 & s_1c_2\\ c_1s_2c_3 + s_1s_3 & c_1s_2s_3 - s_1c_3 & c_1c_2\end{bmatrix} \tag{3.2.2.8}$$

Onde $c \equiv cos(\theta)$ e $s \equiv sen(\theta)$.

Quando se tratando de veículos aeroespaciais é seguido a convenção, rotação no eixo longitudinal se chama rolamento ϕ, rotação no eixo transversal se chama arfagem θ e rotação no eixo vertical se chama guinada ψ.

3.2.3 Eixo-ângulo de Euler

O teorema de rotação do Eixo-ângulo de Euler afirma que ao se rotacionar um corpo-rígido em relação a um eixo arbitrário fixo no corpo e estacionário em relação à um referencial inercial, a atitude de tal corpo pode ser modificada de qualquer orientação para outra. (**??**, p. 329)

Supondo os vetores unitários $\vec{a_i}$ e $\vec{b_i}$com $(i = 1, 2, 3)$, fixos nos sistemas de referência A e B, respectivamente. A orientação de B em relação a A é caracterizado pelo vetor unitário $\vec{e}$ em conjunto ao eixo de Euler e o ângulo de rotação θ em relação a esse eixo, como a seguir:

$$\vec{e} = e_1\vec{a_1} + e_2\vec{a_2} + e_3\vec{a_3} = e_1\vec{b_1} + e_2\vec{b_2} + e_3\vec{b_3} \tag{3.2.3.1}$$

Onde e_i são os cossenos diretores do eixo de Euler em relação a ambos A e B, além disso $e_1^2 + e_2^2 + e_3^2 = 1$. Considerando que $C^{B/A} = C = [C_{ij}]$ é a matriz de cossenos diretores de B em relação a A, então o eixo-ângulo de Euler de rotação é caracterizado como:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{3.2.3.2}$$

O eixo-ângulo de Euler relativo à A e B é aquele que ao ser transformado por $C^{B/A}$ resulta nele mesmo.

Para parametrizar a matriz de cossenos diretores C em termos de e_i e θ_i uma sequencia de rotação deve ser utilizada, demostrada a seguir:

1. Rotacione o sistema de referência A, usando a matriz de rotação R, para alinhar o eixo $\vec{a_1}$ de A com a direção escolhida de $\vec{e}$. Sendo A' o novo sistema de referência encontrado após a rotação e A o sistema original, antes da rotação com os vetores base sendo $\{\vec{a_1}, \vec{a_2}, \vec{a_3}\}$, tem-se:

$$C^{A'/A} = R = \begin{bmatrix} e_1 & e_2 & e_3 \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3.2.3.3}$$

2. Rotacione ambos os sistemas A e A' como corpos-rígidos em torno da direção $\vec{e}$ um ângulo θ. Após essa rotação o sistema de referência A estará alinhado com o B, de vetores de base $\{\vec{b_1}, \vec{b_2}, \vec{b_3}\}$, assim A' será um novo sistema de referência chamado A'' pela a matriz de rotação:

$$C^{A''/A'} = C_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sen\theta \\ 0 & -sen\theta & cos\theta \end{bmatrix} \tag{3.2.3.4}$$

A orientação de B em relação a A é descrita pela matriz de cossenos diretores $C^{B/A}$. É importante notar que a orientação relativa entre A'' e B é a mesma que A' e A'', *i.e.*, $C^{A''/B} = C^{A'/A} = R$.

3. Por fim rotacione A'' pela matriz inversa $R^{-1} = R^T$, assim o sistema A'' estará alinhado com B, visto que $C^{B/A''} = R^{-1}$. Essas três rotações sucessíveis podem ser combinadas como:

$$\begin{bmatrix} \vec{b_1} \\ \vec{b_2} \\ \vec{b_3} \end{bmatrix} = C^{B/A} \begin{bmatrix} \vec{a_1} \\ \vec{a_2} \\ \vec{a_3} \end{bmatrix} \tag{3.2.3.5}$$

4. Onde:

$$C^{B/A} = C^{B/A''} C^{A''/A'} C^{A'/A} = C^{A/A'} C_1(\theta) C^{A'/A} = R^T C_1(\theta) R \tag{3.2.3.6}$$

Analogamente para caso $\vec{a_2}$ ou $\vec{a_3}$ forem alinhados com a direção $\vec{e}$, para esses casos se modifica, da equação acima, a matriz de rotação $C_1(\theta)$ por $C_2(\theta)$ ou $C_3(\theta)$, respectivamente, e a matriz $R(\theta)$ deverá, respectivamente, respeitar o seguinte arranjo:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ e_1 & e_2 & e_3 \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \text{ ou } \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ e_1 & e_2 & e_3 \end{bmatrix} \tag{3.2.3.7}$$

Ao se substituir as equações encontradas nos itens 1., 2. e 3. e definindo que $C = [C_{ij}] = C^{B/A}$, obtemos:

$$C_{11} = e_1^2 + (R_{21}^2 + R_{31}^2) cos\theta \tag{3.2.3.8}$$

$$C_{12} = e_1 e_2 + (R_{21} R_{22} + R_{31} R_{32}) cos\theta + (R_{21} R_{32} - R_{22} R_{31}) sen\theta \tag{3.2.3.9}$$

$$C_{13} = e_1 e_3 + (R_{21} R_{23} + R_{31} R_{33} cos\theta + (R_{21} R_{33} - R_{23} R_{31}) sen\theta) \tag{3.2.3.10}$$

$$\begin{matrix} . \\ . \\ . \end{matrix} \tag{3.2.3.11}$$

$$C_{33} = e_3^2 + (R_{23}^2 + R_{33}^2) cos\theta \tag{3.2.3.12}$$

Como cada elemento da matriz R encontrada no item 1. é igual ao sus cofatores, tem-se também:

$$e_1 = R_{22} R_{33} - R_{23} R_{32} \tag{3.2.3.13}$$

$$e_2 = R_{23} R_{31} - R_{21} R_{33} \tag{3.2.3.14}$$

$$e_3 = R_{21} R_{32} - R_{22} R_{31} \tag{3.2.3.15}$$

A partir dessa relação obtém-se $C = R^T C_1(\theta) R$, da forma matricial:

$$C = \begin{bmatrix} c\theta + e_1^2(1 - c\theta) & e_1 e_2 (1 - c\theta) + e_3 s\theta & e_1 e_3 (1 - c\theta) - e_2 s\theta \\ e_2 e_1 (1 - c\theta) - e_3 s\theta & c\theta + e_2^2 (1 - c\theta) & e_2 e_3 (1 - c\theta) + e_1 s\theta \\ e_3 e_1 (1 - c\theta) + e_2 s\theta & e_3 e_2 (1 - c\theta) - e_1 s\theta & c\theta + e_3^2 (1 - c\theta) \end{bmatrix} \tag{3.2.3.16}$$

Onde $c \equiv cos(\theta)$ e $s \equiv sen(\theta)$. Essa é parametrização da matriz de cossenos diretores C em termos de e_i e θ. Note que e_1,e_2 e e_3 não são independentes entre si, pois respeitam a condição $e_1^2 + e_2^2 + e_3^2 = 1$. Assim, definindo que:

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \text{ e } E = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{3.2.3.17}$$

É possível expressar a matriz de cossenos diretores parametrizada acima como:

$$C = cos\theta\ I + (1 - cos\theta)ee^T - sen\theta E \tag{3.2.3.18}$$

Sendo I a matriz identidade, *i.e.*, $C_{ij} = \delta_{ij}cos\theta + (1 - cos\theta)e_ie_j - sen\theta E_{ij}$.

Dados que a matriz de cosseno diretores $C = [C_{ij}]$, θ pode ser encontrado por:

$$cos\theta = \frac{1}{2}(C_{11} + C_{22} + C_{33} - 1) \tag{3.2.3.19}$$

E pode-se encontrar E, a partir de:

$$E = \frac{1}{2sen\theta}(C^T - C) \text{ se } \theta \neq 0, \pm\pi, \pm 2\pi, ... \tag{3.2.3.20}$$

E finalmente podemos encontra o eixo e por:

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \frac{1}{2sen\theta} \begin{bmatrix} C_{23} - C_{32} \\ C_{31} - C_{13} \\ C_{12} - C_{21} \end{bmatrix} \tag{3.2.3.21}$$

3.2.4 Parâmetros de Euler (Quatérnios)

Apesar dos ângulos de Euler serem de fácil visualização e cálculo, oferecem um contratempo na computação, devido aos problemas de singularidade. Assim é usado os parâmetros de Euler, também chamados de quatérnios, para contornar essas dificuldades.

Os quatérnios tem quatro dimensões, uma real e três imaginárias. Os quatérnios são escritos como $q = [q_1\ q_2\ q_3\ q_4]^T$, seus primeiros três componentes formam a parte vetorial $\vec{q} = [q_1\ q_2\ q_3]^T$e q_4 a parte escalar.

Considerando um eixo-ângulo de Euler $\vec{e}$ rotacionando em relação a um sistema de coordenadas fixo no corpo B e um sistema coordenadas inercial de referência A, definidos como:

$$\vec{e} = e_1\vec{a_1} + e_2\vec{a_2} + e_3\vec{a_3} \tag{3.2.4.1}$$

$$= e_1\vec{b_1} + e_2\vec{b_2} + e_3\vec{b_3} \tag{3.2.4.2}$$

Onde e_i é o cosseno diretor do eixo de Euler em relação a ambos A e B, e $e_1^2 + e_2^2 + e_3^2 = 1$. Pode-se definir os parâmetros de Euler como:

$$q_1 = e_1 sen(\theta/2) \tag{3.2.4.3}$$
$$q_2 = e_2 sen(\theta/2) \tag{3.2.4.4}$$
$$q_3 = e_3 sen(\theta/2) \tag{3.2.4.5}$$
$$q_4 = cos(\theta/2) \tag{3.2.4.6}$$

Onde θ é o ângulo de rotação em relação ao eixo de Euler,$e = (e_1, e_2, e_3)$ e definimos o vetor $\vec{q} = (q_1, q_2, q_3)$, como:

$$\vec{q} = e\ sen(\theta/2) \tag{3.2.4.7}$$

É importante ressaltar que os parâmetros não são independentes entre si, pois estão restringidos pela seguinte relação:

$$\vec{q}^T\vec{q} + q_4^2 = q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{3.2.4.8}$$

devido à $e_1^2 + e_2^2 + e_3^2 = 1$.

Agora que os quatérnios foram definidos, é possível relacionar os métodos de representação citados nas subsecções anteriores. Por exemplo a matriz de cossenos diretores, pode ser igualmente parametrizada usando quatérnios, como a seguir:

$$C^{B/A} = C(\vec{q}, q_4) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_2q_1-q_3q_4) & 1-2(q_1^2+q_3^2) & 2(q_2q_3+q_1q_4) \\ 2(q_3q_1+q_2q_4) & 2(q_3q_2-q_1q_4) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{3.2.4.9}$$

Onde (q_1, q_2, q_3, q_4) são os quatérnios associados a matriz $C^{B/A}$. Note que $sen\theta = 2sen(\theta/2)cos(\theta/2)$ e $cos^2(\theta/2) - sen^2(\theta/2) = 2cos^2(\theta/2) - 1 = 1 - 2sen^2(\theta/2)$.

Definindo o vetor $\vec{q}$ e a matriz antissimétrica Q como:

$$\vec{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix},\ Q = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{3.2.4.10}$$

A matriz de cossenos diretores pode ser escrita:

$$C = (q_4^2 - \vec{q}^t\vec{q})I + 2\vec{q}\vec{q}^T - 2q_4Q \tag{3.2.4.11}$$

Por outro lado, dado a matriz de cossenos diretores C pode-se determinar q_4 e $\vec{q}$, como mostrado a seguir:

$$q_4 = \sqrt{\frac{1}{2}(1 + C_{11} + C_{22} + C_{33})}\ \text{ para }\ 0 \leq \theta \leq \pi \tag{3.2.4.12}$$

$$\vec{q} = \frac{1}{4q_4}\begin{bmatrix} C_{23} - C_{32} \\ C_{31} - C_{13} \\ C_{12} - C_{21} \end{bmatrix}\ \text{ se }\ q_4 \neq 0 \tag{3.2.4.13}$$

3.2.5 Equações Diferenciais da Cinemática

A partir do estudo da descrição da orientação de um sistema de referência ou de um corpo rígido, apresentados nas subsecções anteriores, nessa subsecção é tratado sobre a cinemática, *i.e.*, a representação da orientação entre dois sistemas de coordenadas dependendo do tempo, essa relação é caracterizada por equações diferenciais da cinemática.

Para contornar singularidades e evitar eventuais contratempos em uma simulação computacional foi optado descrever as equações diferenciais da cinemática pela parametrização por quatérnios.

3.2.6 Equações Diferenciais Parametrização de Quatérnios

Considerando dois sistemas de referência A e B, os quais estão em movimento um em relação ao outro. O vetor da velocidade angular do sistema B em relação ao A é denotado como $\vec{\omega} \equiv \vec{\omega}^{B/A}$, pode ser escrito como:

$$\omega_1 = 2(\dot{q_1}q_4 + \dot{q_2}q_3 - \dot{q_3}q_2 - \dot{q_4}q_1) \tag{3.2.6.1}$$

$$\omega_2 = 2(\dot{q_2}q_4 + \dot{q_3}q_1 - \dot{q_1}q_3 - \dot{q_4}q_2) \tag{3.2.6.2}$$

$$\omega_3 = 2(\dot{q_3}q_4 + \dot{q_1}q_2 - \dot{q_2}q_1 - \dot{q_4}q_3) \tag{3.2.6.3}$$

Diferenciando, $\vec{q}^T\vec{q} + q^4 = q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$, tem-se:

$$0 = 2(\dot{q_1}q_1 + \dot{q_2}q_2 + \dot{q_3}q_3 + \dot{q_4}q_4) \tag{3.2.6.4}$$

Essas quatro equações são combinadas da forma matricial a seguir:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} = 2 \begin{bmatrix} q_4 & q_3 & -q_2 & -q_1 \\ -q_3 & q_4 & q_1 & -q_2 \\ q_2 & -q_1 & q_4 & -q_3 \\ q_1 & q_2 & q_3 & q_4 \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{bmatrix} \tag{3.2.6.5}$$

Como a matriz dessa equação é 4 x 4 e ortonormal, pode-se encontrar a equação diferencial dos quatérnios a seguir:

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} \tag{3.2.6.6}$$

Ou ainda, pode ser escrita como:

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{3.2.6.7}$$

Se definirmos os vetores $\vec{q}$ e $\vec{\omega}$ como:

$$\vec{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \ , \vec{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{3.2.6.8}$$

Rescreve-se a equação acima como:

$$\dot{\vec{q}} = \frac{1}{2}(q_4\vec{\omega} - \vec{\omega} \times \vec{q})\dot{q_4} = -\frac{1}{2}\vec{\omega}^T\vec{q} \tag{3.2.6.9}$$

Onde:

$$\vec{\omega} \times \vec{q} \equiv \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.2.6.10}$$

3.3 Dinâmica Rotacional

3.3.1 Dinâmica de Órbita Circular

A dinâmica orbital se baseia nas leis de Newton e de Kepler no estudo do problemas de 2 corpos. As leis de Kepler descrevem o movimento orbital sem pertubações enquanto as de Newton descrevem as relações físicas que regem sobre o movimento descrevendo as forças e momentos sobre o corpo.

Derivando a equação do movimento usando a segunda lei de Newton:

$$\vec{F} = m\vec{a} \tag{3.3.1.1}$$

Sendo $\vec{F}$ a somatória das forças resultantes externas, tem-se:

$$\vec{F} = \sum \vec{F}_{externas} = \vec{F}_{gravitacional} + \vec{F}_{arrasto} + \vec{F}_{magnética} + ... \tag{3.3.1.2}$$

Para esta modelagem é feito as seguintes simplificações:

1. É desconsiderado a atmosfera circundante e efeitos de arrasto.
2. Não há expelimento de matéria propulsiva ou de qualquer outra.
3. É desprezado qualquer interação com outros corpos de massa significativa.
4. Qualquer outra força é de intensidade de uma ordem de grandeza desprezível ou inexistente

Tem-se que a somatória das forças resultantes é de:

$$\vec{F} = \sum \vec{F}_{externas} = \vec{F}_{gravitaconal} \tag{3.3.1.3}$$

Considerando que a $\vec{F}$ pode ser determinada como a variação do momento linear $m\vec{v}$, pode-se escrever a seguinte relação:

$$\vec{F} = m\ \vec{a} = \frac{d}{dt}(m\ \vec{v}) = m\frac{d}{dt}\vec{v} \tag{3.3.1.4}$$

Introduzindo coordenadas polares tem-se que $\vec{e}_r$ é descrito como $\{cos\theta\ e_1, sen\theta\ e_2, 0\ e_3\}$ e $\vec{e}_a$ é descrito como $\{-sen\theta\ e_1, cos\theta\ e_2, 0\ e_3\}$ consequentemente:

$$\vec{v} = \frac{d}{dt}\dot{\vec{r}} = \dot{r}\vec{e}_r + r\dot{\vec{e}}_r = \dot{r}\vec{e}_r + r\theta\vec{e}_s \tag{3.3.1.5}$$

Como a órbita é circular a amplitude de r não varia, assim $\vec{v} = r\theta\vec{e}_s$, a aceleração fica:

$$\vec{a} = \frac{d}{dt}\vec{v} = -r\omega_0^2\ \vec{e}_r = \vec{r}\omega_0^2 \tag{3.3.1.6}$$

Agrupando as equações:

$$m_{corpo}\vec{r}\ w_0^2 = \vec{r}\frac{G\ m_{Terra}\ m_{corpo}}{r^3} \tag{3.3.1.7}$$

Simplificando:

$$\omega_0 = \sqrt{\frac{G\ m_{Terra}}{r^3}} = \sqrt{\frac{\mu}{r^3}} \tag{3.3.1.8}$$

A relação acimar permite encontrar a velocidade angular de um satélite em uma órbita circular revolvendo a Terra, onde G é o parâmetro gravitacional universal e r é a adição entre o raio médio da terra e a altitude da órbita.

3.3.2 Dinâmica Rotacional de Corpo Rígido

Considerando a equação (3.70) a $\vec{F}$ pode ser determinada como a variação do momento linear, analogamente o torque $\vec{T}$ é a variação do momento angular H, assim aplicando o produto vetorial entre o raio $\vec{R}$ e força $\vec{F}$, encontra-se:

$$\vec{T} = \vec{R} \times \vec{F} = \vec{r} \times m\vec{a} = \vec{r} \times m\frac{d}{dt}\vec{v} = \frac{d}{dt}\vec{H} \tag{3.3.2.1}$$

O momento angular de um corpo é dado por:

$$\vec{H} = J\ \vec{\omega} \tag{3.3.2.2}$$

Onde J é a matriz de inércia do corpo e ω a velocidade angular, chega-se em:

A somatória dos momentos de força aplicados externos ao corpo é igual a variação do vetor momento angular do corpo em relação ao seu centro de massa.

Sendo os eixos principais de inercia da matriz de inércia J coincidentes ao sistema de referencia fixa no corpo (super-escrito b), tem-se que:

$$J^b = \begin{bmatrix} J_1^b & 0 & 0 \\ 0 & J_2^b & 0 \\ 0 & 0 & J_3^b \end{bmatrix} \tag{3.3.2.3}$$

Como o corpo do estudo é um CubeSat, pode-se definir os componentes da matriz de inércia como a de um paralelepípedo de massa uniformemente distribuída, onde m é a massa e $\{\rho_1, \rho_2, \rho_3\}$,respectivamente, o comprimento a largura e a altura:

$$J_1^b = \frac{m}{12}(\rho_2^2 + \rho_3^2) \tag{3.3.2.4}$$

$$J_2^b = \frac{m}{12}(\rho_1^2 + \rho_3^2) \tag{3.3.2.5}$$

$$J_3^b = \frac{m}{12}(\rho_1^2 + \rho_2^2) \tag{3.3.2.6}$$

O vetor de momento angular $\vec{H}^b$ pode ser determinado como:

$$\vec{H}^b = J^b \cdot \vec{\omega}_{ib}^b \tag{3.3.2.7}$$

Onde $\vec{\omega}_{ib}^b$ é a velocidade angular do corpo (super-escrito b) com relação ao sistema inercial (sub-escrito i) escrita no sistema do corpo (sub-escrito b).

$$\vec{T^b} = J^b \cdot \dot{\vec{\omega}}_{ib}^b + \vec{\omega}_{ib}^b \times J^b \cdot \vec{\omega}_{ib}^b \tag{3.3.2.8}$$

Reagrupando tem-se que:

$$\dot{\vec{\omega}}_{ib}^b = {J^b}^{-1}(-\vec{\omega}_{ib}^b \times J^b \cdot \vec{\omega}_{ib}^b + \vec{T^b}) \tag{3.3.2.9}$$

Em módulo e dividida nos respectivos componentes, pode ser escrita como:

$$\dot{\omega}_{ib\,1}^b = \frac{1}{J_1^b}((J_2^b - J_3^b)\omega_2^b\omega_3^b + T_1^b) \tag{3.3.2.10}$$

$$\dot{\omega}_{ib\,2}^b = \frac{1}{J_2^b}((J_3^b - J_1^b)\omega_1^b\omega_3^b + T_2^b) \tag{3.3.2.11}$$

$$\dot{\omega}_{ib\,3}^b = \frac{1}{J_3^b}((J_1^b - J_2^b)\omega_1^b\omega_2^b + T_3^b) \tag{3.3.2.12}$$

3.3.3 Modelo de Corpo Rígido em Órbita Circular

Considerando que está-se interessado na alteração da atitude do satélite (corpo) em relação ao sistema de coordenadas geocêntrica-equatorial inercial ω_{ib}^b é necessário conectar a velocidade angular do corpo relativa ao sistema de orbita ω_{ob}^b com a velocidade angular do corpo ao redor da Terra, *i.e.*, ao referencial inercial. Nesse caso:

$$\vec{\omega}_{ib}^b = \vec{\omega}_{ob}^b + \vec{\omega}_{io}^b \tag{3.3.3.1}$$

$$\vec{\omega}_{ib}^b = \vec{\omega}_{ob}^b + C^{B/O}\,\vec{\omega}_{io}^o \tag{3.3.3.2}$$

Onde ω_{io}^o é o movimento médio da órbita. Que é obtido por:

$$\omega_{io}^o = \begin{bmatrix} 0 \\ -\omega_0 \\ 0 \end{bmatrix} \tag{3.3.3.3}$$

Recapitulando a seção anterior, tem-se que:

$$\omega_0 = \sqrt{\frac{\mu}{R_o^3}} \tag{3.3.3.4}$$

Onde, μ é a parâmetro gravitacional da Terra e R_o o raio da órbita circular.

3.4 Sistema de Determinação e Controle de Atitude (ADCS)

O sistema de determinação e controle de atitude é composto por sensores, e.g., sensor de campo magnético, sensor solar, por atuadores, e.g., torqueador eletromagnético, rodas-de-reação, e por algoritmos de determinação e controle, e.g., TRIAD, PID respectivamente, que tratam da orientação do veículo espacial.

Existe disponível no mercado uma ampla gama de sensores, atuadores e até de ADCS completos, como o XACT-50 da Blue Canyon Technologies (Figura **??**) que oferece de forma integrada a solução para a determinação e controle de atitude utilizando sensores de estrela e de rodas de reação.

3.4.1 Sensor Solar e Sensor Magnético

Os sensores são componentes com função de medir uma grandeza física e produzir um sinal capaz de alimentar o algoritmo de determinação de atitude. Os atuadores são componentes com função de alterar o estado do satélite da forma desejada a partir da produção de forças e torques, sua operação é definida por um controlador a partir do estado medido e de uma referencia. A seguir é explorado alguns exemplos de sensores e atuadores comuns.

O sensor solar determina os ângulos das faces do CubeSat em relação ao Sol, tendo um sensor em cada uma de suas faces é possível cobrir todo o céu. Pode ser utilizado em conjunto com sensores magnéticos para determinação de atitude, ou como modo de segurança caso falha de sistemas compostos por giroscópios, ou sensores de estrelas.

A opção disponível no mercado tem um acurácia por volta de 0.5 graus e campo de visão de 114 graus, massa de 5 gramas;

É possível usar as próprias placas fotovoltaicas como sensores solar visto que estas também podem medir os ângulo em relação ao Sol, o que é bem vindo ao se levar em conta as restrições de volume, massa, alimentação elétrica, etc, impostas à plataforma(**??**).

Esse ultimo exemplo pode ser modelado como como um sensor solar analógico, também conhecidos como detectores cossenoidal devido se basear pela variação da saída de corrente produzida pela placa fotovoltaica em relação ao ângulo do Sol.

Sendo E o fluxo de energia por uma superfície de área $d\mathbf{A}$ com a unidade normal $\hat{\mathbf{n}}$. Têm-se:

$$E = \vec{P} \cdot \hat{\mathbf{n}}\, d\mathbf{A} \tag{3.4.1.1}$$

$\vec{P}$ é o vetor de apontamento, que oferece a direção e magnitude do fluxo de energia de irradiação eletromagnética. Como energia é efeticamente depositada na célula fotovoltaica é gerado uma corrente I que é proporcional ao ângulo de incidência da radiação solar θ.

$$I\left(\theta\right) = I\left(0\right) cos\theta \tag{3.4.1.2}$$

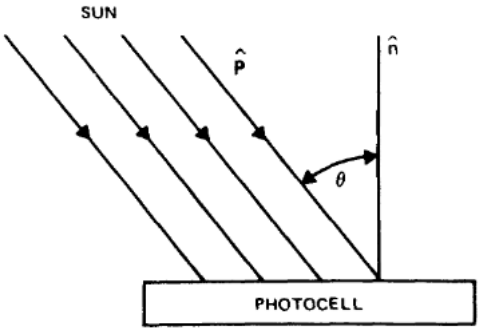


Figura 4 – Sensor Solar de Detecção Cossenoidal
Fonte: (**??**, p. 156)

Para esse modelo, as perdas devido a reflexão de Fresnel, a área efetiva da célula e a reflexão da interface célula-ar dependente do angulo são omitidas.

Para um cubeSat com as células fotovoltaicas encontradas nas seis faces do veículo espacial, $cos\theta$ pode ser encontrado pela relação do vetor normal do painel $\vec{n}$, com o vetor que aponta para o sol pelo regencial do corpo $\vec{s}_{ib}^{b}$, sendo ambos unitários.

$$I(\theta) = I(0)cos\theta \tag{3.4.1.3}$$

$$I(\theta) = I(0)\vec{n}^T\vec{s}_b \tag{3.4.1.4}$$

Utilizando a mesma metodologia para as seis faces tem-se:

$$\mathbf{I} = I(0)\mathbf{N}^T\vec{s}_b \tag{3.4.1.5}$$

Sendo $\mathbf{N}$:

$$\mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \tag{3.4.1.6}$$

Mas ainda $\vec{s}_{ib}^{b}$ pode ser encontrado por:

$$\vec{s}_b = C^{B/I}\vec{s}_i \tag{3.4.1.7}$$

Por fim tem-se:

$$\mathbf{I} = I(0)\mathbf{N}^T C^{B/I}\vec{s}_i \tag{3.4.1.8}$$

O sensor magnético ou magnetômetro, fornece a medida do campo magnético terrestre em três eixos. É necessário atenção para o tratamento de ruídos, já que correntes elétricas, campos magnéticos e temperatura, produzidas externas ou internas ao CubeSat interferem nas medições.

Análogo ao caso anterior as medidas do sensor magnético serão dadas utilizando o sistema de referência o de coordenadas do corpo.

Utilizando um modelo de dipolo inclinado para o campo magnético da terra, é possível escrever seus componentes no SCGI como:

$$m_i^* = \frac{R_T^3 H_0}{r^3}\left[3\mathbf{d}_i^T\hat{\mathbf{r}}_i\hat{\mathbf{r}}_i - \mathbf{d}_i\right] \tag{3.4.1.9}$$

Onde $\mathbf{d}_i$ é a direção unitária do dipolo no referencial inercial:

$$\mathbf{d}_i = \begin{bmatrix}\sin\theta'_m\cos\alpha_m \\ \sin\theta'_m\sin\alpha_m \\ \cos\theta'_m\end{bmatrix} \tag{3.4.1.10}$$

Substituindo acima:

$$m_i^* = \frac{R_T^3 H_0}{r^3}\begin{bmatrix}3(\mathbf{d}_i^T\hat{\mathbf{r}}_i)\hat{r}_1 - \sin\theta'_m\cos\alpha_m \\ 3(\mathbf{d}_i^T\hat{\mathbf{r}}_i)\hat{r}_2 - \sin\theta'_m\sin\alpha_m \\ 3(\mathbf{d}_i^T\hat{\mathbf{r}}_i)\hat{r}_3 - \cos\theta'_m\end{bmatrix} \tag{3.4.1.11}$$

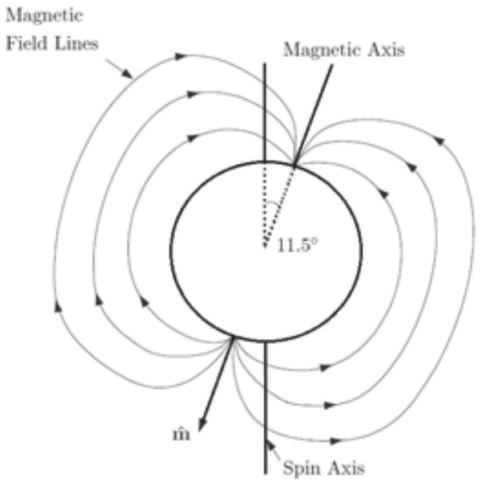


Figura 5 – Modelo Sensor Magnético
Fonte: (**??**, p. 13)

Sendo r o vetor posição do veículo espacial, R_T o raio da Terra, H_0 a constante característica do vetor magnético da Terra e θ'_m co-elevação.

$$\alpha_m = \theta_{go} + \omega_T t + \phi'_m \tag{3.4.1.12}$$

Sendo θgo a época sideral de greenwich, ω_T a rotação média da Terra, t o tempo desde a época e ϕ'_m a longitude leste do dipolo.

3.4.2 Determinação de Atitude: Metodo TRIAD

Pelo menos dois vetores são necessários para determinar a atitude. No entanto, é importante notar que, embora sejam necessários três parâmetros independentes para determinar a atitude, cada medição de vetor unitário na verdade fornece apenas dois parâmetros devido à restrição do vetor unitário. Portanto, são necessários três escalares para determinar completamente a atitude. Assim, a exigência é de mais de uma, mas

menos de dois medições de vetores. A determinação da atitude é única nesse sentido, pois uma medição não é suficiente, ou seja, o problema é subdeterminado, e duas medições são demais, ou seja, o problema é super-determinado. A principal implicação dessa observação é que todos os algoritmos de determinação de atitude são realmente algoritmos de estimação de atitude.

O algoritmo TRIAD é uma técnica fundamental na determinação da atitude de uma espaçonave. A atitude de uma espaçonave é essencialmente a descrição da matriz de rotação que descreve a orientação do referencial fixo à espaçonave, B, em relação a um referencial conhecido, como um referencial inercial, I.

Embora essa matriz de cossenos diretores possua nove números, como mostrado anteriormente, apenas três números são necessários para determiná-la completamente. Como cada vetor unitário medido fornece duas informações, são necessárias pelo menos duas medidas diferentes para determinar a atitude o que resulta em um problema super-determinado, pois temos três incógnitas e quatro quantidades conhecidas.

O algoritmo TRIAD parte de dois vetores de medição, como a direção para o sol e a direção do campo magnético terrestre. Denotamos os vetores reais como $\vec{s}$ e $\vec{m}$, respectivamente. As componentes medidas dos vetores, em relação ao referencial do corpo, são denotadas por $\vec{s}_b$ e $\vec{m}_b$, respectivamente. As componentes conhecidas dos vetores no referencial inercial são $\vec{s}_i$ e $\vec{m}_i$.

Idealmente, a matriz de rotação, ou matriz de atitude, $C^{B/I}$, satisfaz $\vec{s}_b = C^{B/I}\vec{s}_i$ e $\vec{m}_b = C^{B/I}\vec{m}_i$. Infelizmente, como o problema é super-determinado, geralmente não é possível encontrar tal $C^{B/I}$. O algoritmo de determinação de atitude determinística mais simples é baseado em descartar uma parte dessa informação no entanto, essa abordagem não se resume simplesmente a descartar uma das componentes de uma das direções medidas. O algoritmo é conhecido como algoritmo TRIAD, porque é baseado na construção de dois triângulos de vetores unitários ortogonais usando as informações vetoriais disponíveis. Os dois triângulos são as componentes do mesmo referencial, denotado por T, expresso nos referenciais do corpo e inercial. Este referencial é construído assumindo que um dos pares de vetores corpo/inercial está correto.

$$\begin{bmatrix} \hat{t}_1 \\ \hat{t}_2 \\ \hat{t}_3 \end{bmatrix} = \begin{bmatrix} \hat{s} \\ \hat{s} \times \hat{m} \\ \hat{t}_1 \times \hat{t}_2 \end{bmatrix} \tag{3.4.2.1}$$

$$\begin{bmatrix} \hat{t}_{1b} \\ \hat{t}_{2b} \\ \hat{t}_{3b} \end{bmatrix} = \begin{bmatrix} \hat{s}_b \\ \hat{s}_b \times \hat{m}_b \\ \hat{t}_{1b} \times \hat{t}_{2b} \end{bmatrix} \tag{3.4.2.2}$$

$$\begin{bmatrix} \hat{t}_{1i} \\ \hat{t}_{2i} \\ \hat{t}_{3i} \end{bmatrix} = \begin{bmatrix} \hat{s}_i \\ \hat{s}_i \times \hat{m}_i \\ \hat{t}_{1i} \times \hat{t}_{2i} \end{bmatrix} \tag{3.4.2.3}$$

$$C^{B/I} = C^{B/T}C^{B/I} = \begin{bmatrix} \hat{t}_{1b}\hat{t}_{2b}\hat{t}_{3b} \end{bmatrix} \begin{bmatrix} \hat{t}_{1i} \\ \hat{t}_{2i} \\ \hat{t}_{3i} \end{bmatrix} \tag{3.4.2.4}$$

3.4.3 Controle de Atitude: Técnica de Controle PID

Para o controle da atitude do CubeSat por rodas de reação se usa um controlador automático estes comparam o valor de saída, medido por um sensor, com o valor de entrada, de referência, desejado. A comparação desses dois valores determina um desvio chamado de erro, e esse sinal de erro é tratado pelo controlador que gera por fim um sinal de controle para os atuadores que agiram sobre o sistema na busca de diminuir esse desvio. A técnica aqui demostrada está presente em (**??**, p. 21).

O controle proporcional-integral-derivativo ou PID, é um controle automático que combina ações de natureza proporcional, integral e derivativa no tratamento do sinal de erro. Essa ação combinada oferece as vantagens induviais de cada um desses tipo de controle. A utilização de rodas de reação quando o controle é feito por velocidade, no presente caso aplica-se um controlador PID ao sinal de corrente. Dessa forma a equação do sinal de controle é dada por:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right) \tag{3.4.3.1}$$

Onde $e(t)$ é o erro entre a velocidade angular de referência e a observada, K_p é o ganho proporcional, T_i é o tempo integrativo e T_d é o tempo derivativo.

Para o caso que o modelo matemático da planta a ser considerada possa ser obtido, então é possível aplicar técnicas de projeto para auxiliar na determinação de parâmetros do controlador para atender as especificações do regime transitório e do regime permanente do sistema de malha fechada. Contudo se a planta for muito complexa é possível que a abordagem analítica seja não praticável, assim tem-se que recorrer para uma abordagem experimental de sintonia do controlador PID.

O método de sintonia de Ziegler e Nichols, (**??**, p. 524), sugere ajustar os parâmetros Kp, T_i e T_d a partir da resposta experimental ao aumentar o valor de K_p até resultar em uma estabilidade marginal, quando só a ação proporcional é considerada. Ou seja, definindo $T_i = \infty$ e $T_d = 0$, aumenta-se o valor de K_p de 0 até o valor crítico K_{cr}no qual a saída oscila de forma sustentada pela primeira vez, o período de oscilação encontrado é P_{cr}. Caso para qualquer valor de K_p não se encontre uma resposta oscilatória sustentada do sistema então o método não se aplica.

Para os valores de P_{cr} e K_{cr}encontrados Ziegle e Nichols sugere escolher os parâmetros Kp, T_i e T_d de acordo com a tabela 1 a seguir:

Tabela 1 – Regra de sintonia de Ziegler-Nichols baseada no ganho crítico K_{cr} e no período crítico P_{cr}.

Tipo de controlador	K_p	T_i	T_d
P	$0,5K_{cr}$	∞	0
PI	$0,45K_{cr}$	$\frac{1}{1,2}P_{cr}$	0
PID	$0,6K_{cr}$	$0,5P_{cr}$	$0,125P_{cr}$

3.4.4 Rodas de Reação em 3 Eixos

As rodas de reação (RDR) são dispositivos atuadores utilizados para o controle de atitude de veículos espaciais, possuindo, como a base de seu funcionamento, a transferência

de momento angular. Podendo ser arranjadas em conjuntos de três unidades para controlar a rotação nos três eixos, de quatro unidades para além dos controles dos três eixos oferecer redundância ao sistema, ou avulso.

Por permitirem a correção de atitude com precisão, estes dispositivos têm operação mais complexa e são mais custosos. Apenas em missões nas quais os requisitos do apontamento são mais sensíveis se utiliza esse tipo de atuador, sendo assim menos comuns em CubeSats.

Contudo, com o aumento de missões mais rebuscadas, estas que se encontram distante do campo magnético da Terra, faz-se mais necessário esses dispositivos, deste modo, se tornando comuns e até comercializados, que é o caso da SatBus 4RW0, um conjunto de 4 rodas de reação desenvolvidas pela NanoAvionics para seus CubeSats (M3P ao M16P), também podendo ser encomendadas, e a CubeWhell encontrada na CubeSat Shop e a Blue Canyon com sistemas completamente integrados de ADCS com rodas de reação acopladas, Figura **??**.

Uma roda de reação é uma massa rotativa motorizada que fornece torque para manobra e é usada para o apontamento de satélites. Podendo rotacionar tanto no sentido horário quanto no anti-horário, assim como acelerar ou desacelerar, criando torques e manobrando o satélite.

Esse dispositivo necessita de uma forma de dessaturar ao longo do tempo, assim um sistema adicional podendo ser torqueadores magnéticos ou propulsores cumprem essa função descarregando o momento angular.

Mais precisamente uma roda de reação consistem em cilindros de metal encaixadas em um motor sem escova achatado, é indispensável que esse motor tenha uma boa relação potência consumo e que funcione dentro da faixa de potência oferecida por esse sistema do satélite. Estas rodas de reação permitem um alto grau de acurácia e são essenciais em missões as quais um controle mais fino e robusto é necessário.

Variando entre 0,4 a 40 $\frac{kg\,m^2}{s}$, podendo ser uma roda pequena e rápida ou uma roda grande e lenta. Esses dispositivos se baseiam no principio da conservação de momento, *i.e.*, $\vec{H} = J^b_{RDR}\,\vec{\omega}_{RDR}$. Onde $\vec{\omega}_{RDR}$ é o vetor das velocidades angulares e J^b_{RDR} a matriz de inércia da roda-de-reação. O modelo utilizado para a caracterização física-matemática da roda de reação foi adaptado de (**??**, p. 270-271).

O torque da roda é oferecido por um motor de indução elétrica de corrente contínua, alimentado por pulsos quadrados provenientes da bateria controlados pelo computador de bordo ao nível do ciclo de trabalho, que está relacionado com a velocidade de rotação da roda.

O Ciclo de trabalho X_{ct} varia entre +1 e -1 a partir do sinal de controle, V, mostrado na Figura **??**.

O torque líquido é encontrado por:

$$N = X_{ct}N_{em} - N_{fricção} \tag{3.4.4.1}$$

Onde N_{em} é o torque eletromagnético produzido por indução, e $N_{fricção}$ contabiliza o atrito viscoso e de Coumlomb, descrito por:

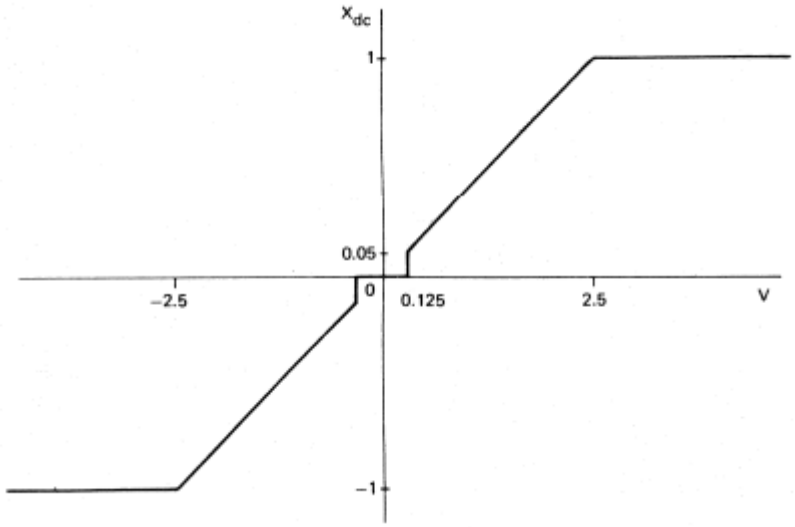


Figura 6 – Ciclo de Trabalho X_{CT}
Fonte: (**??**, p. 271)

$$N_{fricção} = N_c \left(\frac{s}{|s|} \right) + f\ s \tag{3.4.4.2}$$

Onde N_c é o coeficiente de fricção de Coulomb, f é o coeficiente de fricção viscosa e s a velocidade de rotação da roda.

Para encontrar o torque eletromagnético, N_{em}, pode-se se utilizar a seguinte aproximação:

$$N_{em} = \frac{2N_0\alpha\ r}{(\alpha^2 + r^2)} \tag{3.4.4.3}$$

Sendo, N_0 a magnitude máxima de N_{em}, α é o valor de r onde se encontra N_0 e r é encontrado pela seguinte relação:

$$r = 1 - \frac{s}{s_{max}} \text{ para } X_{ct} > 0 \tag{3.4.4.4}$$

$$r = 1 + \frac{s}{s_{max}} \text{ para } X_{ct} < 0 \tag{3.4.4.5}$$

Atente-se que s_{max} é conhecida pela velocidade de sincronização da roda-de-reação. O torque eletromagnético em relação a velocidade da roda é mostrado na Figura 7.

3.4.5 Modelagem das Rodas-De-Reação

A equação para uma roda-de-reação, controlada por uma corrente contínua $i(t)$ ao invés de uma tensão de controle, e, utilizando os modelos de zona ociosa, torque eletromagnético, fricção viscosa e o coeficiente do torque de Coulomb descritos acima, é dada por:

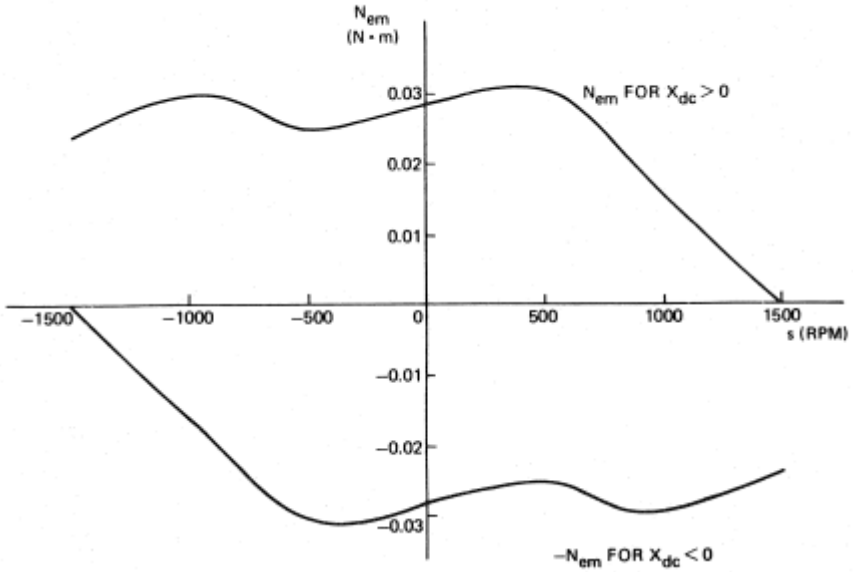


Figura 7 – Torque Eletromagnético N_{em} em relação a velocidade de rotação da roda-de-reação s

Fonte: (**??**, p. 271)

$$T_{aplicado} = J_{axial}\frac{d\omega_{rdr}}{dt} = Xct(i(t))\frac{2N_0\alpha\left(1-\frac{\omega_{rdr}}{\omega_{rdr\ max}}\right)}{\alpha^2\left(1-\frac{\omega_{rdr}}{\omega_{rdr\ max}}\right)^2} - (sign(Xct(i(t)))N_c + f\omega_{rdr}) \tag{3.4.5.1}$$

Como as três rodas-de-reação são idênticas, pode-se modela-las como cilindros de metal com massa uniformemente distribuída, onde m_{rdr} é a massa e $\{\rho_{rdr}\}$ o raio:

$$J_{axial} = \frac{m}{2}\rho_{rdr}^2 \tag{3.4.5.2}$$

Contabilizando somente as componentes axiais das rodas de reação e alinhando cada com eixos principais de inercia da matriz de inércia coincidentes ao sistema de referencia fixa no corpo, tem-se que J_{rdr}^b:

$$J_{rdr}^b = \begin{bmatrix} J_{axial1} & 0 & 0 \\ 0 & J_{axial2} & 0 \\ 0 & 0 & J_{axial3} \end{bmatrix} \tag{3.4.5.3}$$

E aplicando a modelagem para cada eixo, tem-se que o vetor de torques aplicado fica:

$$\vec{T}_{aplicado} = J_{rdr}^b\frac{d\vec{\omega_{rdr}^b}}{dt} \tag{3.4.5.4}$$

Parte II

Proposta

4 Implementando Laboratório Virtual

4.1 Robot Operating System 2 (ROS2) e Gazebo 11

O texto está bem escrito e claro, transmitindo informações importantes sobre o ROS e o ROS2. No entanto, sugiro fazer algumas pequenas alterações para melhorar a fluidez e a precisão:

O ROS, ou Robot Operating System, é uma plataforma de código aberto que fornece um conjunto de bibliotecas e ferramentas open-source para criar aplicações robóticas. Ele atua como um middleware entre a aplicação e os robôs, permitindo o controle de dispositivos em níveis baixos e a implementação de funcionalidades comuns, como troca de mensagens entre processos e gerenciamento de pacotes.

O ROS2 é a versão mais recente, projetada como uma evolução do ROS original, com melhorias significativas em áreas como segurança, escalabilidade, robustez e facilidade de uso. Ele suporta várias linguagens de programação, como C++, Python e outras, e oferece uma ampla gama de ferramentas e bibliotecas para auxiliar no desenvolvimento de software robótico.

Essas plataformas são fundamentais para o desenvolvimento de software robótico avançado, permitindo a criação de sistemas distribuídos e complexos, a reutilização de código e a integração com uma variedade de hardware e sensores.

No contexto do ROS, os conceitos de Node, Topic, Service e Action são essenciais para compreender como os sistemas robóticos são desenvolvidos e operam. Aqui está uma breve explicação de cada um:

Node (Nó): Um Node é um processo que executa uma função específica dentro de um sistema ROS. Os Nodes são unidades de processamento independentes que se comunicam uns com os outros por meio de mensagens. Cada Node pode realizar tarefas como controle de hardware, cálculos complexos, processamento de sensores, etc. Um sistema ROS geralmente consiste em vários Nodes que trabalham juntos para realizar tarefas complexas.

Topic (Tópico): Um Topic é um canal de comunicação assíncrono usado para enviar mensagens entre Nodes. Um Node pode publicar mensagens em um Topic e outro Node pode se inscrever para receber essas mensagens. Os Topics são usados para transmitir dados como leituras de sensores, comandos de controle, informações de estado, etc.

Service (Serviço): Um Service é uma forma de comunicação síncrona entre Nodes, onde um Node solicita um serviço a outro Node e aguarda a resposta. Os Services são usados para realizar operações que requerem uma resposta, como solicitar uma leitura de sensor específica, executar uma ação de controle, etc.

Action (Ação): As Actions são semelhantes aos Services, mas são usadas para operações que podem levar um tempo indeterminado para serem concluídas, como navegação de um robô até um determinado ponto. As Actions permitem que um Node envie uma solicitação, receba atualizações sobre o progresso da ação e, finalmente, receba uma resposta quando a ação for concluída.

Esses conceitos são fundamentais para o funcionamento do ROS, pois permitem que os Nodes se comuniquem de forma eficiente e realizem tarefas complexas de forma colaborativa em sistemas robóticos.

No ecossistema do ROS, RQT, RViz2 e Gazebo são componentes importantes que desempenham papéis distintos. Aqui está uma explicação de cada um:

RQT: O RQT é um framework de ferramentas gráficas para o ROS, que permite aos desenvolvedores visualizar e interagir com diferentes aspectos de um sistema ROS. Ele fornece uma variedade de plugins para tarefas como visualização de tópicos (*topic*) e serviços (*Service*), depuração, análise de dados, entre outros. O RQT é altamente extensível e pode ser personalizado para atender às necessidades específicas do desenvolvedor.

RViz2: O RViz2 é um visualizador 3D usado para exibir dados sensoriais, modelos de robôs, informações de navegação e outros dados relevantes em ambientes robóticos simulados ou reais. Ele permite que os desenvolvedores visualizem e interajam com o ambiente do robô de forma interativa, facilitando o desenvolvimento e depuração de algoritmos de controle e percepção.

GAZEBO: O Gazebo é um simulador robótico avançado que permite simular ambientes, sensores e atuadores com alta fidelidade. Ele é integrado ao ROS e é frequentemente usado para testar e validar algoritmos de controle em ambientes simulados antes de implantá-los em robôs reais. O Gazebo suporta a simulação de uma ampla variedade de sensores e atuadores, tornando-o uma ferramenta poderosa para o desenvolvimento de software robótico.

Essas ferramentas e conceitos são fundamentais para o desenvolvimento de sistemas robóticos com o ROS, fornecendo aos desenvolvedores as ferramentas necessárias para projetar, simular, visualizar e depurar seus sistemas robóticos de forma eficiente e eficaz.

No contexto do ROS (Robot Operating System) e do Gazebo, URDF (Unified Robot Description Format) e SDF (Simulation Description Format) são duas linguagens de marcação usadas para descrever modelos de robôs e ambientes de simulação, respectivamente. Aqui está uma explicação mais detalhada de cada um:

URDF (Unified Robot Description Format): O URDF é um formato de arquivo XML usado para descrever modelos de robôs em termos de suas estruturas, cinemática, visualização e outros aspectos relevantes para a simulação e controle. Ele é amplamente utilizado no ROS para descrever robôs em ambientes de simulação e no desenvolvimento de software de controle de robôs. O URDF permite especificar links, juntas, sensores, visualizações, entre outros componentes de um modelo de robô.

SDF (Simulation Description Format): O SDF é um formato de arquivo XML usado para descrever ambientes de simulação, incluindo modelos de robôs, objetos estáticos, luzes, terrenos e outras entidades. O SDF é usado principalmente no Gazebo, um simulador robótico popular, para definir o ambiente no qual os modelos de robôs são simulados. O SDF permite especificar a geometria, material, física e outras propriedades dos objetos no ambiente de simulação.

O URDF e o SDF são complementares e são frequentemente usados juntos no desenvolvimento de sistemas robóticos. O URDF descreve os modelos de robôs que serão controlados e simulados, enquanto o SDF descreve o ambiente no qual esses modelos serão colocados e interagirão. Juntos, esses formatos de arquivo permitem aos desenvolvedores criar simulações realistas de sistemas robóticos e testar seus algoritmos de controle em

ambientes seguros e controlados antes de implantá-los em robôs reais.

O Gazebo é um simulador robótico avançado que suporta uma variedade de sensores usados em robôs. Ele oferece modelos precisos e realistas de sensores, permitindo que os desenvolvedores simulem o comportamento sensorial de seus robôs em ambientes virtuais. Alguns dos sensores comumente suportados pelo Gazebo incluem:

Câmeras: O Gazebo pode simular câmeras de diferentes tipos, como câmeras RGB, câmeras de profundidade e câmeras de infravermelho. Esses sensores podem ser usados para capturar imagens do ambiente do robô e são úteis para tarefas como navegação, reconhecimento de objetos e mapeamento.

Lidar: Lidar, ou Light Detection and Ranging, é um sensor usado para medir distâncias e gerar nuvens de pontos tridimensionais do ambiente. O Gazebo suporta diferentes tipos de sensores lidar, que são essenciais para aplicações como mapeamento, localização e evasão de obstáculos.

GPS: O Gazebo pode simular um receptor GPS, permitindo que os desenvolvedores testem algoritmos de localização global em seus robôs. O GPS é essencial para a navegação autônoma em ambientes externos.

IMU (Inertial Measurement Unit): Um IMU é um sensor que mede a aceleração e a taxa de rotação de um objeto. O Gazebo suporta a simulação de IMUs, que são usados para estimar a orientação e a velocidade angular de um robô.

O ROS 2 Control é um conjunto de ferramentas e bibliotecas no ROS 2 que facilita o controle de robôs. Ele fornece uma estrutura para implementar controladores de baixo nível para manipuladores, veículos e outros tipos de robôs. O ROS 2 Control suporta diferentes tipos de controladores, como controladores de posição, velocidade e esforço, e oferece recursos para configurar, ativar e desativar esses controladores de forma programática. Essa flexibilidade torna o ROS 2 Control adequado para uma variedade de aplicações robóticas, desde robôs industriais até veículos autônomos.

4.2 Roteiro 0: Instalação e Configuração

O presente projeto propõe o desenvolvimento de um laboratório virtual como de ambiente para o ensino de modelagem e controle de CubeSats por meio de simulação.

Utilizando ferramentas aberta e livres, contornando o problema de custo.

E de roteiros de instalação, configuração e uso, contornando o problema da curva de aprendizado.

Para a realização do presente projeto está foi escolhido as aplicações a seguir:

- Ubuntu 22.04 LTS
- ROS 2 - Humble
- Gazebo 11
- GNU Octave

Para o funcionamento é previsto o hardware mínimo a seguir:

- Memória: 4 GB
- Disco: SSD 120 GB
- Placa de Vídeo: 1GB Dedicada
- Processador: QuadCore 2.1 GHz

4.2.1 Instalação ROS 2 Humble, Gazebo 11 Simulator e GNU Octave

É possível encontrar o sistema operacional de forma gratuita pela pagina oficial do Ubuntu na referência (**??**) e (**??**).

Faça a instalação mínima do sistema, abra o terminal pelo iniciar ou pelo comando de teclado, "CTRL+ALT+T". Selecione e cole a rotina abaixo para a instalação do ROS 2 Humble, o Gazebo 11 Simulator e o GNU Octave.

Listing 4.1 – bash version

```
#!/bin/bash

# Instala ROS

locale # check for UTF-8

sudo apt update -y
sudo apt upgrade -y
sudo apt --fix-broken install -y
sudo apt autoremove -y

sudo apt install locales -y

sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale # verify settings

sudo apt install software-properties-common -y
sudo add-apt-repository universe -y

sudo apt update -y
sudo apt upgrade -y
sudo apt --fix-broken install -y
sudo apt autoremove -y

sudo apt install curl -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key
    -o /usr/share/keyrings/ros-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture)
    signed-by=/usr/share/keyrings/ros-archive-keyring.gpg]
    http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo
```

```
    $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list >
    /dev/null

sudo apt update -y
sudo apt upgrade -y
sudo apt --fix-broken install -y
sudo apt autoremove -y

sudo apt install ros-humble-desktop -y
sudo apt install ros-dev-tools -y

source /opt/ros/humble/setup.bash
sudo apt install python3-pip -y
pip3 install -U argcomplete
pip3 install xacro
sudo apt install python3-colcon-common-extensions -y
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc


#ros2 run demo_nodes_cpp talker
#ros2 run demo_nodes_py listener

sudo apt install ros-humble-joint-state-publisher-gui
sudo apt install ros-humble-gazebo-ros-pkgs -y

sudo apt update -y
sudo apt upgrade -y
sudo apt --fix-broken install -y
sudo apt autoremove -y
```

Para testar a instalação abra duas abar no terminal e copie e cole:

Listing 4.2 – bash version

```
#!/bin/bash
ros2 run demo_nodes_cpp talker
```

Onde o resultado será:

[INFO] [1708993338.850590057] [talker]: Publishing: 'Hello World: 1'

[INFO] [1708993339.850601664] [talker]: Publishing: 'Hello World: 2'

Na primeira, e:

Listing 4.3 – bash version

```
#!/bin/bash
ros2 run demo_nodes_py listener
```

Na segunda, com o respectivo resultado sendo:

[INFO] [1708993347.861065131] [listener]: I heard: [Hello World: 10]

[INFO] [1708993348.851486479] [listener]: I heard: [Hello World: 11]

4.2.2 Configuração ROS Package

Abra o terminal, copie e cole a rotina abaixo:

Listing 4.4 – bash version

```
# Crie o diretorio
mkdir -p dev_ws/src
# Entre no diretorio
cd dev_ws
# Utilize o Colcon para construir seu Workspace
colcon build --symlink-install
# A flag symlink e usada por padrao
ls
#Verifice se as pastas build install log e src foram criadas, se sim, voce tem
    agora um workspace vazio.
# Os arquivos em ROS sempre sao organizados em packages, e eles devem estar no
    diretorio src
cd src
#Crie o package
ros2 pkg create --build-type ament_cmake my_package
# Crie a pasta Description Launcher e World
cd
mkdir -p dev_ws/src/my_package/description
mkdir -p dev_ws/src/my_package/launch
mkdir -p dev_ws/src/my_package/worlds
```

Com isso seu local de trabalho e pastas estão configuradas para trabalharmos com o ROS.

É recomendado, mas não obrigatório, usar um editor de código-fonte, por exemplo o GVim.

4.3 Roteiro 1: Descrevendo CubeSat - Unified Robot Description Format

No desenvolvimento de sistemas robóticos é desejado utilizar um linguagem comum para descrever o sistema. É interessante manter toda essa descrição no mesmo lugar onde outras aplicações fazem a referência, esse formato é o urdf. O arquivo urdf é formado por extruturas simples que se repetem ao longo do arquivo.

O primeiro passo é quebrar a estrutura do cubeSat em componentes separados que são chamados de *links*. Para a separação dos *links* tenha em mente o seguinte:

- Se duas partes de movem independentemente entre elas, as mesmas precisam ser *links* diferentes.
- Quando faz sentido a parte ser , ou faz algo além de se mover, *e.g.*, como sensores.

Na escolha dos *links* é necessário escolher sua origem, principalmente para partes que rotacionam pois esse será o ponto pivô.

Os links são definidos por três informações:

- Visual: É o que vai ser visto no Gazebo.
  - Geometria: Se refere ao formato
  - Material: Se refere a cor
- Collision: Usa os valores anteriores, mas determina a fronteira de interação entre links.
- Inertial: Todas as propriedades inerciais.
  - Massa: A massa do objeto
  - Inercia: A matriz de inercia do objeto

Como no exemplo a seguir de um paralelepípedo arbitrário a seguir:

Listing 4.5 – XML version

```
<link name="NOME_link">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <box size="${Aresta_X} ${Aresta_Y} ${Aresta_Z}" />
        </geometry>
        <material name="blue"/>
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <box size="${Aresta_X} ${Aresta_Y} ${Aresta_Z}" /
        </geometry>
    </collision>
    <inertial>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <mass value="${MASSA}" />
        <inertia
            ixx="Ixx" ixy="0.0" ixz="0.0"
            iyy="Iyy" iyz="0.0"
            izz="Izz" />
        </inertial>
    </link>
```

Agora devemos definir as relações entre esses *links*, essa relação é definida pelas *joints*, relacionando assim o movimento, rotação entre os sistemas de coordenadas de cada link, por isso em sua descrição é necessário dizer o tipo de movimento que está usando.

As *joints* são descritas por:

- Tipo:
  - Revolução, Continua, Fixa, Flutuante...
- *Parent Link*

- *Child Link*
- Origem, eixos de movimento, e limites.

Como no exemplo de uma junta continua, caracteriza o movimento de rotação entre dois *links*, em relação ao eixo z do *link pai*:

Listing 4.6 – XML version

```
<joint name="NOME_joint" type="continuous">
<origin xyz="0 0 0" rpy="0 0 0"/>
<parent link="Nome_link_Pai"/>
<child link="Nome_link_filho"/>
<axis xyz ="0 0 1" rpy = "0 0 0"/>
<limit effort="10000" velocity="10000"/>
<joint_properties damping ="0.0" friction="0.0"/>
</joint>
```

4.3.0.1 Descrição do CubeSat

Listing 4.7 – XML version

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="robot">

    <link name="world"></link>

    <joint name="1U_joint" type="floating">
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <parent link="world"/>
      <child link="1U_link"/>
    </joint>

    <xacro:property name="U_aresta" value="0.1" />
    <xacro:property name="U_massa" value="1" />

    <link name="1U_link">
        <visual>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
            </geometry>
    <material name="blue"/>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
            </geometry>
        </collision>
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <mass value="${U_massa}" />
            <inertia ixx="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                xiy="0.0" ixz="0.0"
                    iyy="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                        iyz="0.0"
```

```
                izz="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}" />
    </inertial>
</link>

<joint name="2U_joint" type="fixed">
    <origin xyz="0 0 ${U_aresta/2} " rpy="0 0 0"/>
    <parent link="1U_link"/>
    <child link="2U_link"/>
</joint>

<link name="2U_link">
    <visual>
        <origin xyz="0 0 ${U_aresta/2}" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
  <material name="orange"/>
    </visual>
    <collision>
        <origin xyz="0 0 ${U_aresta/2}" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
    </collision>
    <inertial>
        <origin xyz="0 0 ${U_aresta/2}" rpy="0 0 0"/>
        <mass value="${U_massa}" />
        <inertia ixx="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
            ixy="0.0" ixz="0.0"
                iyy="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                    iyz="0.0"
                izz="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}" />
    </inertial>
</link>

<joint name="3U_joint" type="fixed">
    <origin xyz="0 0 ${U_aresta}" rpy="0 0 0"/>
    <parent link="2U_link"/>
    <child link="3U_link"/>

</joint>

<link name="3U_link">
    <visual>
        <origin xyz="0 0 ${U_aresta/2}" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
  <material name="green"/>
    </visual>
    <collision>
        <origin xyz="0 0 ${U_aresta/2}" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
    </collision>
    <inertial>
        <origin xyz="0 0 ${U_aresta/2}" rpy="0 0 0"/>
        <mass value="1" />
```

```
        <inertia ixx="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
            ixy="0.0" ixz="0.0"
               iyy="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                   iyz="0.0"
               izz="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}" />
    </inertial>
</link>

<joint name="camera_joint" type="fixed">
    <origin xyz="${U_aresta} 0 ${U_aresta/2}" rpy="0 0 0"/>
    <parent link="3U_link"/>
    <child link="camera_link"/>
</joint>

<link name="camera_link">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
  <material name="white"/>
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
    </collision>
    <inertial>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <mass value="1" />
        <inertia ixx="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
            ixy="0.0" ixz="0.0"
               iyy="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                   iyz="0.0"
               izz="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}" />
    </inertial>
</link>

<joint name="5U_joint" type="fixed">
    <origin xyz="${U_aresta} 0 ${U_aresta/2}" rpy="0 0 0"/>
    <parent link="2U_link"/>
    <child link="5U_link"/>
</joint>

<link name="5U_link">
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
  <material name="gray"/>
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
            <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
        </geometry>
    </collision>
```

```
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <mass value="1" />
            <inertia ixx="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                ixy="0.0" ixz="0.0"
                    iyy="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                        iyz="0.0"
                    izz="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}" />
        </inertial>
    </link>

        <joint name="6U_joint" type="fixed">
        <origin xyz="${U_aresta} 0 0" rpy="0 0 0"/>
        <parent link="1U_link"/>
        <child link="6U_link"/>
    </joint>

    <link name="6U_link">
        <visual>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
            </geometry>
      <material name="gray"/>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <geometry>
                <box size="${U_aresta} ${U_aresta} ${U_aresta}" />
            </geometry>
        </collision>
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0"/>
            <mass value="1" />
            <inertia ixx="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                ixy="0.0" ixz="0.0"
                    iyy="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}"
                        iyz="0.0"
                    izz="${(1/12) * U_massa * (U_aresta*U_aresta+U_aresta*U_aresta)}" />
        </inertial>
    </link>
    <xacro:include filename="example_gazebo.xacro" />

</robot>
```

Itens para evitar repetição como propriedades e macros para incluir

Listing 4.8 – XML version

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" >

    <material name="white">
        <color rgba="1 1 1 1"/>
    </material>

    <material name="gray">
        <color rgba="0.3 0.3 0.3 1"/>
```

```
    </material>

    <material name="orange">
        <color rgba="1 0.3 0.1 1"/>
    </material>

    <material name="blue">
        <color rgba="0.2 0.2 1 1"/>
    </material>

    <material name="green">
        <color rgba="0.2 1 0.2 1"/>
    </material>

    <xacro:macro name="inertial_sphere" params="mass radius *origin">
        <inertial>
            <xacro:insert_block name="origin"/>
            <mass value="${mass}" />
            <inertia ixx="${(2/5) * mass * (radius*radius)}" ixy="0.0" ixz="0.0"
                   iyy="${(2/5) * mass * (radius*radius)}" iyz="0.0"
                   izz="${(2/5) * mass * (radius*radius)}" />
        </inertial>
    </xacro:macro>

    <xacro:macro name="inertial_box" params="mass x y z *origin">
        <inertial>
            <xacro:insert_block name="origin"/>
            <mass value="${mass}" />
            <inertia ixx="${(1/12) * mass * (y*y+z*z)}" ixy="0.0" ixz="0.0"
                   iyy="${(1/12) * mass * (x*x+z*z)}" iyz="0.0"
                   izz="${(1/12) * mass * (x*x+y*y)}" />
        </inertial>
    </xacro:macro>


    <xacro:macro name="inertial_cylinder" params="mass length radius *origin">
        <inertial>
            <xacro:insert_block name="origin"/>
            <mass value="${mass}" />
            <inertia ixx="${(1/12) * mass * (3*radius*radius + length*length)}"
                ixy="0.0" ixz="0.0"
                   iyy="${(1/12) * mass * (3*radius*radius + length*length)}" iyz="0.0"
                   izz="${(1/2) * mass * (radius*radius)}" />
        </inertial>
    </xacro:macro>

</robot>
```

Configurações adicionais para o gazebo

Listing 4.9 – XML version

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

    <gazebo reference="1U_link">
        <material>Gazebo/Blue</material>
    </gazebo>
```

```
    <gazebo reference="2U_link">
        <material>Gazebo/Orange</material>
    </gazebo>

    <gazebo reference="3U_link">
        <material>Gazebo/Green</material>
    </gazebo>

    <gazebo reference="5U_link">
        <material>Gazebo/Gray</material>
    </gazebo>

    <gazebo reference="ADCS_link">
        <material>Gazebo/Gray</material>
    </gazebo>

    <gazebo>
        <plugin name="gazebo_ros_joint_state_publisher"
            filename="libgazebo_ros_joint_state_publisher.so">
            <update_rate>20</update_rate>
            <joint_name>1U_joint</joint_name>
        </plugin>
    </gazebo>

    <gazebo>
   <plugin filename="gz-sim-imu-system"
        name="gz::sim::systems::Imu">
             <update_rate>20</update_rate>
            <joint_name>1U_joint</joint_name>
   </plugin>
    </gazebo>

    <gazebo>
        <plugin name="gazebo_ros_joint_pose_trajectory"
            filename="libgazebo_ros_joint_pose_trajectory.so">
            <update_rate>2</update_rate>
                <joint_name>1U_joint</joint_name>
        </plugin>
    </gazebo>

</robot>
```

4.4 Roteiro 2: Adicionando Sensores - Gazebo Pluggins

4.4.1 Sensor IMU

O IMU (Inertial Measurement Unit) Gazebo Sensor é uma ferramenta crucial na simulação de robôs em ambientes virtuais, como o Gazebo. Ele simula sensores que medem a orientação e a aceleração do robô, fornecendo dados importantes para a navegação e controle. Com o IMU, é possível testar algoritmos de fusão sensorial, odometria e controle de forma precisa e eficaz, sem a necessidade de hardware real. Isso torna o desenvolvimento e a depuração de sistemas robóticos mais acessíveis e econômicos.

Ele é adicionado da seguinte maneira:

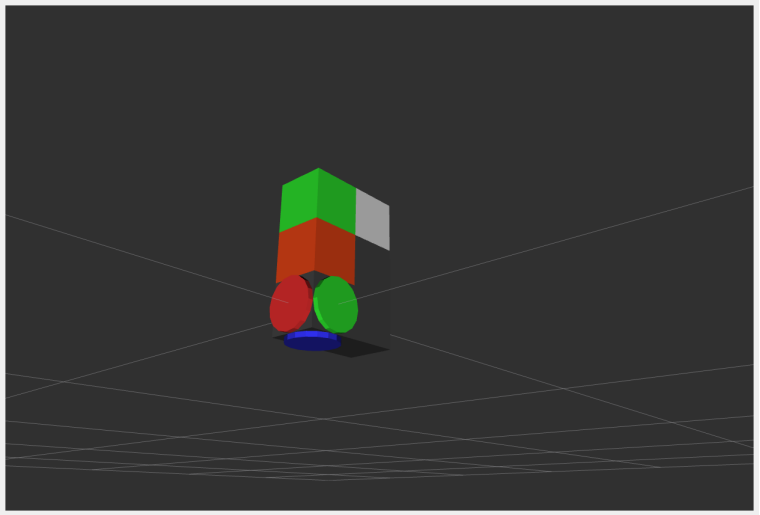


Figura 8 – Vizualização ROS2 do URDF do CUBESAT 6U

Listing 4.10 – XML version

```
    <joint name="6U_ADCS_joint" type="fixed">
       <origin xyz="0 0 0" rpy="-1.571 0 -1.571" />
       <parent link="6U_link" />
       <child link="6U_ADCS_link" />
    </joint>

    <link name="6U_ADCS_link"></link>

    <gazebo reference="6U_link">
   <sensor name="imu_sensor" type="imu">
        <always_on>1</always_on>
        <update_rate>1</update_rate>
        <visualize>true</visualize>
        <topic>imu</topic>
   </sensor>
    </gazebo>
```

Os dados do IMU são orientação, velocidade angular e aceleração linear. A saída das mensagens enviadas sobre o tópico /imu deve se assemelhar com o seguinte:

Listing 4.11 – Bash version

```
    ---
header:
  stamp:
    sec: 393
    nanosec: 424000000
  frame_id: 1U_link
orientation:
  x: 0.023725517648530682
  y: 0.04281485544486788
  z: -0.48399201528223695
  w: 0.8737023046258853
orientation_covariance:
- 0.0
```

```
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
angular_velocity:
  x: -0.05568477597832236
  y: -0.020611977578691595
  z: 0.2085619814357944
angular_velocity_covariance:
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
linear_acceleration:
  x: -0.3077501933465464
  y: -0.6085366879098204
  z: 10.27332940477441
linear_acceleration_covariance:
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
---
```


4.4.2 Sensor Câmera

A adição do plugin de câmera no Gazebo é fundamental para simulações que requerem visão computacional ou processamento de imagens. Esse plugin permite a simulação de câmeras em um ambiente virtual, gerando imagens realistas que podem ser processadas por algoritmos de visão computacional.

Ao adicionar o plugin de câmera, é possível configurar diversos parâmetros, como resolução da imagem, campo de visão, tipo de lente, entre outros. Além disso, o plugin permite que as imagens geradas sejam publicadas em tópicos ROS, facilitando a integração com outros componentes do sistema robótico virtual.

Com a adição do plugin de câmera, os desenvolvedores podem testar e depu-

rar algoritmos de visão computacional em um ambiente controlado e seguro, antes de implementá-los em hardware real. Isso acelera o desenvolvimento de sistemas robóticos que dependem de percepção visual, como robôs autônomos e sistemas de monitoramento.

Listing 4.12 – XML version

```
<joint name="camera_optical_joint" type="fixed">
    <origin xyz="0 0 0" rpy="-1.571 0 -1.571" />
    <parent link="camera_link" />
    <child link="camera_link_optical" />
</joint>

<link name="camera_link_optical"></link>

<gazebo reference="camera_link">
    <sensor type="depth" name="my_camera">
        <update_rate>20</update_rate>
        <visualize>true</visualize>
        <camera name="cam">
            <horizontal_fov>1.3962634</horizontal_fov>
            <image>
                <width>640</width>
                <height>480</height>
                <format>R8B8G8</format>
            </image>
            <clip>
                <near>0.02</near>
                <far>300</far>
            </clip>
            <noise>
                <type>gaussian</type>
                <mean>0.0</mean>
                <stddev>0.007</stddev>
            </noise>
        </camera>
        <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
            <frame_name>camera_link_optical</frame_name>
            <min_depth>0.1</min_depth>
            <max_depth>500</max_depth>
        </plugin>
    </sensor>
</gazebo>
```

4.5 Considerações Finais

Ao longo desta dissertação, exploramos o potencial do ROS 2 e do simulador Gazebo 11 para a modelagem e simulação de um CubeSat, assim como seu ambiente operacional em órbita baixa. A utilização destas ferramentas se mostrou altamente promissora, oferecendo um ambiente robusto e flexível para o desenvolvimento de sistemas espaciais.

A integração de sensores de câmera e IMU no simulador foi relativamente simples, permitindo a simulação de dados sensoriais realistas para o CubeSat virtual. Estes sensores são fundamentais para as operações de um satélite, incluindo orientação, navegação e controle.

Por outro lado, a implementação do ROS 2 Control tem sido um desafio significativo.

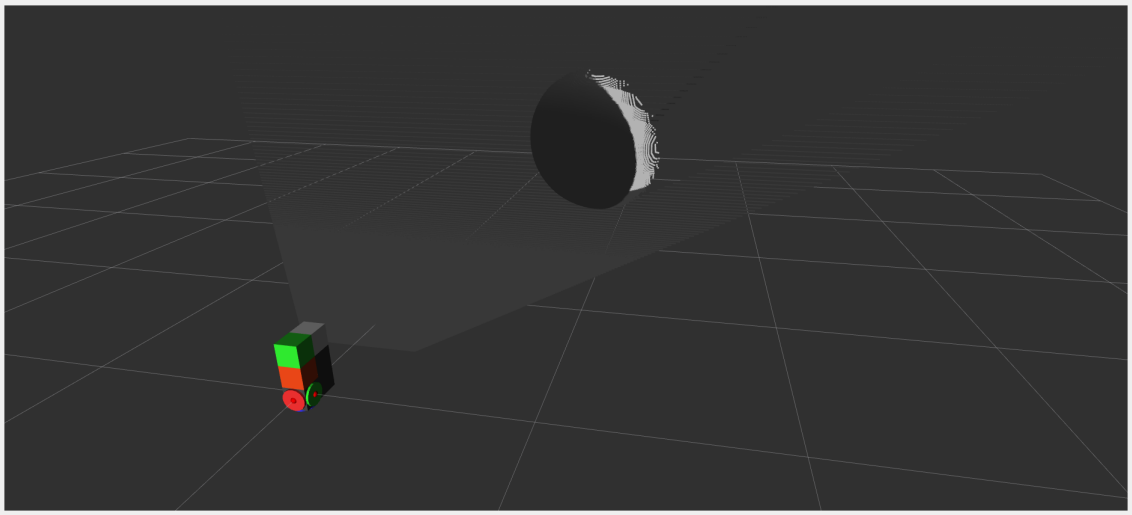


Figura 9 – Point Cloud do Sensor Câmera Visualização ROS2 do URDF do CUBESAT 6U

A transição do ROS 1 para o ROS 2 e a adaptação aos novos padrões de controle têm exigido um esforço adicional de desenvolvimento e depuração. No entanto, a utilização do ROS 2 Control oferece benefícios a longo prazo, como maior eficiência e suporte aprimorado para sistemas distribuídos.

A representação precisa do ambiente em órbita baixa também se mostra desafiadora. A modelagem de fenômenos como arrasto atmosférico, perturbações gravitacionais e interações com outros corpos celestes requer um esforço adicional de pesquisa e desenvolvimento. No entanto, a precisão deste ambiente é crucial para a validação dos sistemas de controle e navegação do CubeSat.

Em suma, o uso do ROS 2 e do Gazebo Simulator 11 para a modelagem e simulação de um CubeSat e seu ambiente operacional em órbita baixa é altamente promissor, apesar dos desafios encontrados. Estas ferramentas oferecem um ambiente rico e flexível para o desenvolvimento e teste de sistemas espaciais, contribuindo para o avanço da tecnologia espacial.

Parte III

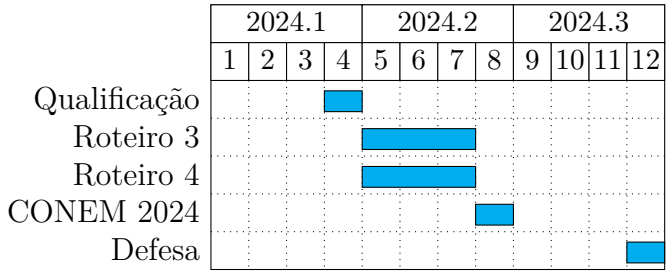
Parte Final

5 Cronograma de Atividades

5.1 Atividades Previstas

1. Exame de Qualificação de Mestrado do PPGMEC (Data 23/04/2024)
2. Roteiro 3: Adicionando Atuadores - ROS2 Control
3. Roteiro 4: Determinação e Controle de Atitude
4. CONEM 2024 (29/07/24 a 02/08/24)
5. Defesa da Dissertação de Mestrado (Previsão 2024.03)

5.2 Cronograma



Conclusões e Trabalhos Futuros

Proin non sem. Donec nec erat. Proin libero. Aliquam viverra arcu. Donec vitae purus. Donec felis mi, semper id, scelerisque porta, sollicitudin sed, turpis. Nulla in urna. Integer varius wisi non elit. Etiam nec sem. Mauris consequat, risus nec congue condimentum, ligula ligula suscipit urna, vitae porta odio erat quis sapien. Proin luctus leo id erat. Etiam massa metus, accumsan pellentesque, sagittis sit amet, venenatis nec, mauris. Praesent urna eros, ornare nec, vulputate eget, cursus sed, justo. Phasellus nec lorem. Nullam ligula ligula, mollis sit amet, faucibus vel, eleifend ac, dui. Aliquam erat volutpat.

Conclusões

Fusce vehicula, tortor et gravida porttitor, metus nibh congue lorem, ut tempus purus mauris a pede. Integer tincidunt orci sit amet turpis. Aenean a metus. Aliquam vestibulum lobortis felis. Donec gravida. Sed sed urna. Mauris et orci. Integer ultrices feugiat ligula. Sed dignissim nibh a massa. Donec orci dui, tempor sed, tincidunt nonummy, viverra sit amet, turpis. Quisque lobortis. Proin venenatis tortor nec wisi. Vestibulum placerat. In hac habitasse platea dictumst. Aliquam porta mi quis risus. Donec sagittis luctus diam. Nam ipsum elit, imperdiet vitae, faucibus nec, fringilla eget, leo. Etiam quis dolor in sapien porttitor imperdiet.

Cras pretium. Nulla malesuada ipsum ut libero. Suspendisse gravida hendrerit tellus. Maecenas quis lacus. Morbi fringilla. Vestibulum odio turpis, tempor vitae, scelerisque a, dictum non, massa. Praesent erat felis, porta sit amet, condimentum sit amet, placerat et, turpis. Praesent placerat lacus a enim. Vestibulum non eros. Ut congue. Donec tristique varius tortor. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nam dictum dictum urna.

Phasellus vestibulum orci vel mauris. Fusce quam leo, adipiscing ac, pulvinar eget, molestie sit amet, erat. Sed diam. Suspendisse eros leo, tempus eget, dapibus sit amet, tempus eu, arcu. Vestibulum wisi metus, dapibus vel, luctus sit amet, condimentum quis, leo. Suspendisse molestie. Duis in ante. Ut sodales sem sit amet mauris. Suspendisse ornare pretium orci. Fusce tristique enim eget mi. Vestibulum eros elit, gravida ac, pharetra sed, lobortis in, massa. Proin at dolor. Duis accumsan accumsan pede. Nullam blandit elit in magna lacinia hendrerit. Ut nonummy luctus eros. Fusce eget tortor.

Trabalhos Futuros

Ut sit amet magna. Cras a ligula eu urna dignissim viverra. Nullam tempor leo porta ipsum. Praesent purus. Nullam consequat. Mauris dictum sagittis dui. Vestibulum sollicitudin consectetuer wisi. In sit amet diam. Nullam malesuada pharetra risus. Proin lacus arcu, eleifend sed, vehicula at, congue sit amet, sem. Sed sagittis pede a nisl. Sed tincidunt odio a pede. Sed dui. Nam eu enim. Aliquam sagittis lacus eget libero. Pellentesque diam sem, sagittis molestie, tristique et, fermentum ornare, nibh. Nulla et tellus non felis imperdiet mattis. Aliquam erat volutpat.

Apêndices

APÊNDICE A – Primeiro Apêncice

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

APÊNDICE B – Segundo apêndice com título tão grande quanto se queira porque ele já faz a quebra de linha da coisa toda

Maecenas dui. Aliquam volutpat auctor lorem. Cras placerat est vitae lectus. Curabitur massa lectus, rutrum euismod, dignissim ut, dapibus a, odio. Ut eros erat, vulputate ut, interdum non, porta eu, erat. Cras fermentum, felis in porta congue, velit leo facilisis odio, vitae consectetuer lorem quam vitae orci. Sed ultrices, pede eu placerat auctor, ante ligula rutrum tellus, vel posuere nibh lacus nec nibh. Maecenas laoreet dolor at enim. Donec molestie dolor nec metus. Vestibulum libero. Sed quis erat. Sed tristique. Duis pede leo, fermentum quis, consectetuer eget, vulputate sit amet, erat.

Donec vitae velit. Suspendisse porta fermentum mauris. Ut vel nunc non mauris pharetra varius. Duis consequat libero quis urna. Maecenas at ante. Vivamus varius, wisi sed egestas tristique, odio wisi luctus nulla, lobortis dictum dolor ligula in lacus. Vivamus aliquam, urna sed interdum porttitor, metus orci interdum odio, sit amet euismod lectus felis et leo. Praesent ac wisi. Nam suscipit vestibulum sem. Praesent eu ipsum vitae pede cursus venenatis. Duis sed odio. Vestibulum eleifend. Nulla ut massa. Proin rutrum mattis sapien. Curabitur dictum gravida ante.

Phasellus placerat vulputate quam. Maecenas at tellus. Pellentesque neque diam, dignissim ac, venenatis vitae, consequat ut, lacus. Nam nibh. Vestibulum fringilla arcu mollis arcu. Sed et turpis. Donec sem tellus, volutpat et, varius eu, commodo sed, lectus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Quisque enim arcu, suscipit nec, tempus at, imperdiet vel, metus. Morbi volutpat purus at erat. Donec dignissim, sem id semper tempus, nibh massa eleifend turpis, sed pellentesque wisi purus sed libero. Nullam lobortis tortor vel risus. Pellentesque consequat nulla eu tellus. Donec velit. Aliquam fermentum, wisi ac rhoncus iaculis, tellus nunc malesuada orci, quis volutpat dui magna id mi. Nunc vel ante. Duis vitae lacus. Cras nec ipsum.

Anexos

ANEXO A – Nome do Primeiro Anexo

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

ANEXO B – Nome de Outro Anexo

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.