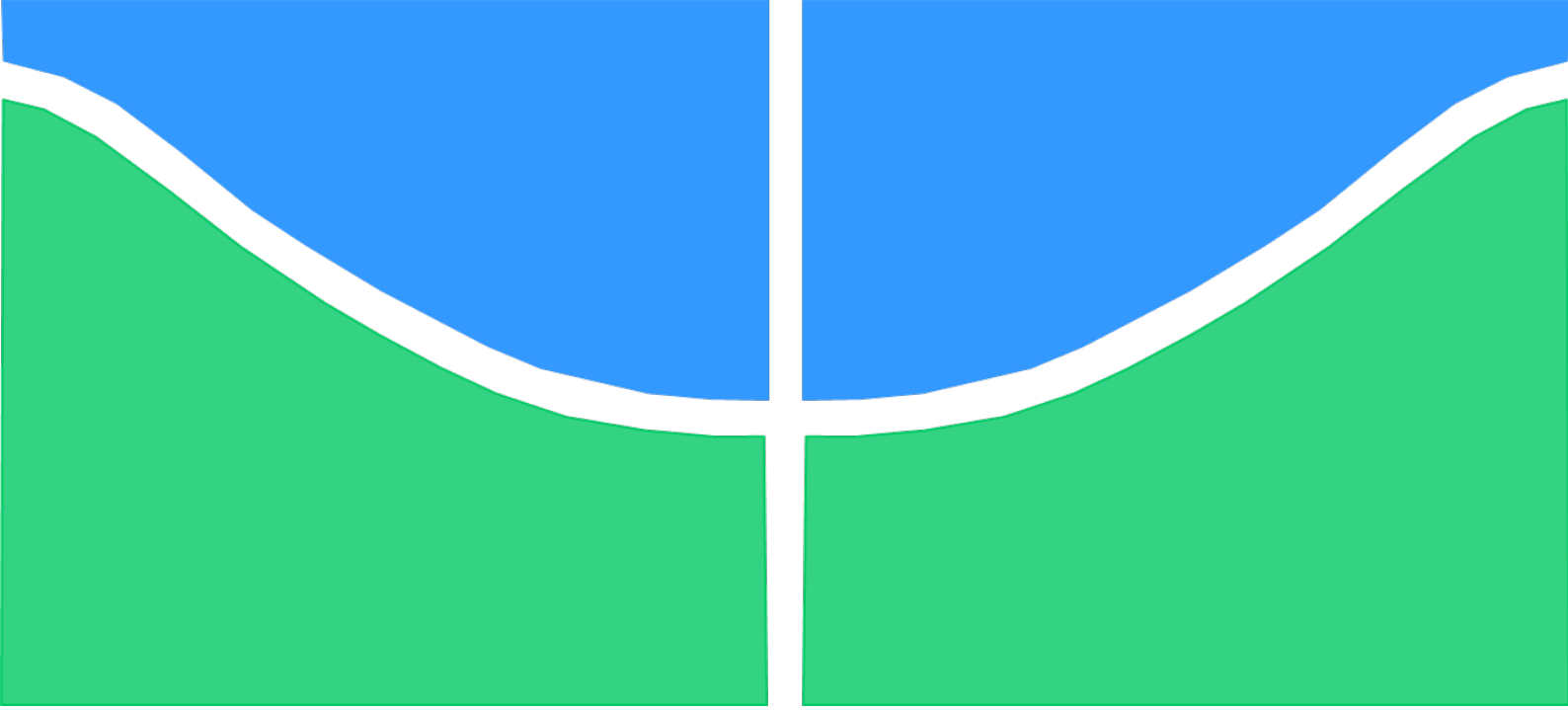


TRABALHO DE GRADUAÇÃO

SIMULADOR DE SISTEMA DE DETERMINAÇÃO E CONTROLE DE ATITUDE DE PEQUENOS SATÉLITES

Por,

Rodrigo Cardoso da Silva

Ulisses Alves Rodrigues

Brasília, Dezembro de 2015

ENGENHARIA MECATRÔNICA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

SIMULADOR DE SISTEMA DE DETERMINAÇÃO E CONTROLE DE ATITUDE DE PEQUENOS SATÉLITES

Por,
Rodrigo Cardoso da Silva
Ulisses Alves Rodrigues

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Dr. Renato Alves Borges , ENE/UnB _____
Orientador

Profa. Dra. Chantal Cappelletti, FGA/UnB _____
Coorientador

Prof. Dr. Simone Battistini, FGA/UnB _____
Examinador interno

Brasília, Dezembro de 2015

FICHA CATALOGRÁFICA

SILVA, RODRIGO CARDOSO DA. RODRIGUES, ULISSES ALVES.
Simulador de sistema de determinação e controle de atitude de pequenos satélites,

[Distrito Federal] 2015.

x, 112p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2015). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1. Plataforma de simulação
2. Balanceamento
3. Matrizes de rotação
4. Ângulos de Euler
5. Microssatélites

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

SILVA, R. C. da; RODRIGUES, U. A., (2015). Simulador de sistema de determinação e controle de atitude de pequenos satélites. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-n° 14/2015, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 112p.

CESSÃO DE DIREITOS

AUTORES: Rodrigo Cardoso da Silva. Ulisses Alves Rodrigues.

TÍTULO: Simulador de sistema de determinação e controle de atitude de pequenos satélites.

GRAU: Engenheiro ANO: 2015

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. Os autores reservam outros direitos de publicação e nenhuma parte deste Trabalho de Graduação pode ser reproduzida sem autorização por escrito de todos os autores.

Rodrigo Cardoso da Silva
Endereço de e-mail: rodrigocardoso2048@gmail.com

Ulisses Alves Rodrigues
Endereço de e-mail: ulisses.a.ro@gmail.com

Dedicatórias

Dedico este trabalho à minha mãe, Maria Rose Meiry Alves da Silva, cujo amor incondicional me dá forças para seguir meus objetivos sem medo. Dedico também àquele que é, ao mesmo tempo, meu melhor amigo, meu ídolo, meu campeão e meu pai, Ari Cardoso da Silva. E à minha irmã, Patrícia Cardoso da Silva. Amo vocês.

Rodrigo Cardoso da Silva

Dedico este trabalho à minha família, que nunca mediu esforços para que eu conseguisse completar minha trajetória na Universidade de Brasília

Ulisses Alves Rodrigues

Agradecimentos

Agradeço primeiramente aos meus pais, Ari e Rose, por não pouparem esforços em prol da minha felicidade e por serem o alicerce da minha vida. Agradeço ao meu amigo e companheiro neste projeto, Ulisses, pelo esforço, pelos momentos de descontração, pela cobrança e, principalmente, pela amizade. A todos os professores que tiveram participação significativa em minha formação acadêmica durante a vida universitária, em especial aos professores Lineu da Costa Araújo Neto, Celius Antônio Magalhães e Yuri Dumaresq Sobral, do departamento de Matemática; à professora Zilda Maria de Oliveira Shokranian, do Instituto de Física; aos professores Renato Alves Borges, João Yoshiyuki Ishihara, Gerson Henrique Pfitscher, Flávia Maria Guerra e Anésio de Leles Ferreira Filho, do departamento de Engenharia Elétrica; aos professores Antônio Francisco Parentes Fortes, Edson Paulo da Silva, Eugênio Liborio Feitosa Fortaleza e Guilherme Caribé Carvalho, do departamento de Engenharia Mecânica; ao professor Ricardo Lopes de Queiroz do departamento de Ciência da Computação; e aos professores Simone Battistini e Chantal Cappelletti, do campus do Gama. Reservo infindável admiração a todos esses professores. Ao professor Gerson Henrique Pfitscher, pelas valiosas histórias e conselhos de vida compartilhados comigo. Aos professores Renato, Chantal e Simone pelo apoio e pela orientação prestada durante a realização deste projeto. À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), fundação do Ministério da Educação (MEC), por ter fornecido os recursos sem os quais este projeto não teria se tornado uma realidade. Aos funcionários dos prédios SG-9 e SG-11 por toda a inestimável ajuda e por terem tido paciência com meus pedidos constantes, nomeadamente Marcos Roberto Pereira da Silva, Artur Alves Rocha, Valter e Cícero. Aos porteiros e porteiras do prédio SG-11, por terem sido obrigados a me ver entrar e sair dos laboratórios incontáveis vezes, mesmo em horários não-comerciais. Aos estudantes do campus do Gama, nomeadamente Pedro Beghelli, Mariana Sampaio, Sarah Gomes e Brenno Taylor, e ao professor Eugenio e seu orientando - e também amigo meu - Rédytton, por terem se envolvido e contribuído com este projeto. A todos os amigos que conheci durante a vida universitária e que a tornaram mais agradável. Aos meus melhores amigos, Gabriel Henrique, Flávio Carlos, Renato Carlos, Pablo Vinicius e Fábio Matos, que me mostraram ao longo dos últimos treze anos o valor da amizade verdadeira. E, por fim, a todos os familiares e amigos que tiveram de ser compreensíveis com minha ausência em diversas situações no período em que me dediquei a este trabalho, mas que mesmo assim continuaram me dando suporte.

Rodrigo Cardoso da Silva

Agradeço primeiramente aos meus pais, João e Neide, pelo amor, incentivo e apoio incondicionais. Agradeço ao meu irmão, Vinicius, que sempre foi um exemplo e um grande amigo. Agradeço aos meus companheiros de UnB, em especial ao Goiano e ao Velho, sem os quais não teria conseguido me formar. Agradeço ao Rodrigo por ter firmado comigo uma parceria de amizade e companheirismo na realização deste projeto. Agradeço ao nosso orientador, Renato Borges, cujas críticas e sugestões ajudaram a aprimorar a qualidade deste trabalho. Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por ter fornecido os recursos necessários para tornar este projeto uma realidade. Agradeço a todos os meus amigos, que me apoiaram nos momentos de dificuldade e celebraram comigo os tempos de sucesso.

Ulisses Alves Rodrigues

"Para ser grande, sê inteiro: nada
Teu exagera ou exclui.

Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.

Assim em cada lago a lua toda
Brilha, porque alta vive."

Fernando Pessoa

RESUMO

As peculiaridades do ambiente espacial, como a presença de microgravidade, do campo magnético da Terra e da ausência de atmosfera, influenciam no projeto do *hardware* e *software* utilizados para sistemas responsáveis pela determinação e pelo controle da atitude de satélites.

Para garantir o funcionamento adequado dos sistemas desenvolvidos, durante a fase de projeto, simulações utilizando *software* ou *hardware* específicos devem ser conduzidos. Almejando a simulação em terra das condições do ambiente espacial, o Laboratório de Aplicação e Inovação em Ciências Aeroespaciais (LAICA) da Universidade de Brasília (UnB), está desenvolvendo uma plataforma de testes dedicada à simulação do movimento de microssatélites. A plataforma é composta de uma mesa com rolamento a ar e de uma gaiola de Helmholtz.

A mesa com rolamento a ar é uma plataforma cujo propósito é de simular as condições de microgravidade de um microssatélite em órbita. A mesa utilizada como plataforma de simulação flutua por meio da atuação de um sistema pneumático que usa o rolamento a ar. A movimentação da mesa é controlada pelos sistemas mecânico e eletrônico de balanceamento, ambos embarcados no simulador. A parte mecânica da plataforma é composta pelo rolamento a ar, massas móveis e fixas, além de todo o equipamento que é embarcado. Já o sistema eletrônico é composto por motores, baterias, microcontroladores, sensores diversos (*e.g.* acelerômetros e magnetômetros) e dispositivos de comunicação sem fio. O deslocamento das massas móveis é alcançado por meio de motores de passo e, usando um algoritmo de balanceamento conhecido, almeja minimizar a distância entre o centro de massa e o centro de rotação da mesa, de modo a tornar o torque gravitacional presente desprezível.

Este trabalho descreve o projeto de toda a parte física e a implementação de todo o *software* utilizado nesse simulador, levando-se em conta todos os sistemas eletro-mecânicos e os procedimentos de calibração.

Palavras-chave: rolamento a ar, satélite, atitude, simulação, plataforma, balanceamento

ABSTRACT

The space environment peculiarities, as the presence of microgravity, the magnetic field of the Earth and the absence of atmosphere, influence the project of hardware and software used for systems responsible for determination and control of attitude of satellites.

To ensure proper functioning of the designed systems, during the project phase, simulations using software or specific hardware shall be conducted. Aiming the simulation at ground of the conditions of the space environment, the Laboratory of Aerospace Science and Innovation (LAICA) of the University of Brasília (UnB), is developing a dedicated test-bed for simulating spacecraft attitude motion. The test-bed is composed of an air bearing table and an Helmholtz cage.

The air bearing table is a hardware platform whose purpose is to simulate the microgravity conditions of a spacecraft in orbit. The table used as the simulation platform floats by means of actuation of a pneumatic system that uses an air bearing. The table movements are controlled by the mechanical and electronic systems of mass balancing, both embedded in the simulator. The mechanical part of the platform is composed by the air bearing, fixed and movable masses, besides all the embedded in the plate hardware. The electronic system is composed by motors, batteries, microcontrollers, various sensors (*e.g.* accelerometers and magnetometers) and wireless communication system. The displacement of the movable masses is achieved through the stepper motors and, using a well-known balancing algorithm, aims at minimizing the distance between the center of mass and the center of rotation of the table, so as to make the gravitational torque negligible.

This work describes the design of all the physical parts and the implementation of the whole software used into this simulator taking into account all the electro-mechanical systems and the calibration procedures.

Keywords: air bearing, satellite, attitude, simulation, platform, balancing

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

r	distância	[m]
d	distância	[m]
L	Comprimento do pêndulo	[m]
F	Força	[N]
P	Força-peso	[N]
I	Momento de inércia	[kg.m^2]
m	Massa do componente	[kg]
M	Massa do sistema	[kg]
g	Aceleração gravitacional	[m/s^2]
t	tempo	[s]
T	Período	[s]
v	Velocidade linear	[m/s]
V	Velocidade linear	[m/s]
a	Aceleração linear	[m/s^2]
H	Momento angular	[kg.m^2/s]

Símbolos Gregos

Ω	Velocidade angular	[rad/s]
ω	Velocidade angular	[rad/s]
α	Aceleração angular	[rad/s^2]
τ	Torque	[N.m]
ρ	Vetor-distância	[m]
θ	Ângulo arbitrário ou ângulo de arfagem	[rad]
ϕ	Ângulo de rolagem	[rad]
ψ	Ângulo de guinada	[rad]

Grupos Adimensionais

i	Contador
$\overrightarrow{i}$, $\overrightarrow{j}$, $\overrightarrow{k}$	Versores
S	Operador matriz antissimétrica
$\|\|\overrightarrow{a}\|\|$	Norma do vetor $\overrightarrow{a}$
Δ	Variação
R	Matriz de rotação
P	Ponto
E	Esperança de uma variável aleatória
det(A)	Determinante da matriz A

Subscritos

$\perp$	Componente perpendicular
N	Componente normal
T	Componente tangencial
i	Inercial (*inertial*) ou Referente a partícula
b	Corpo (*body*)
x	Referente ao eixo x
y	Referente ao eixo y
z	Referente ao eixo z
xyz	Referente ao sistema de referência xyz
XYZ	Referente ao sistema de referência XYZ
UMM_i	Referente à Unidade de Massa Móvel do eixo i
CR	Referente ao centro de rotação
CM	Referente ao centro de massa
G	Gravitacional - *Gravity*
$disp$	Dispositivos
$aero$	Aerodinâmico - *Aerodynamic*
xx	Referente ao momento principal de inércia do eixo x
yy	Referente ao momento principal de inércia do eixo y
zz	Referente ao momento principal de inércia do eixo z
xy, yx	Referente aos produtos de inércia dos eixos x e y
xz, zx	Referente aos produtos de inércia dos eixos x e z
yz, zy	Referente aos produtos de inércia dos eixos y e z

Sobrescritos

$\rightarrow$	Vetor
T	Matriz transposta
$\overline{A}$	Valor médio de A

Siglas

CR	Centro de Rotação
CM	Centro de Massa
UMM	Unidade de Massa Móvel
UMI	Unidade de Medição Inercial
UCP	Unidade Central de Processamento
ADI	Ambiente de Desenvolvimento Integrado

Capítulo 1

Introdução

Esta seção destina-se à contextualização do projeto, incluindo o histórico de pesquisas realizadas na área e a exposição do estado da arte.

1.1 Contextualização

Desde meados da década de 60, à época do início da corrida espacial, rolamentos a ar são utilizados para desenvolvimento de sistemas de controle e determinação de atitude de satélites e veículos espaciais [34].

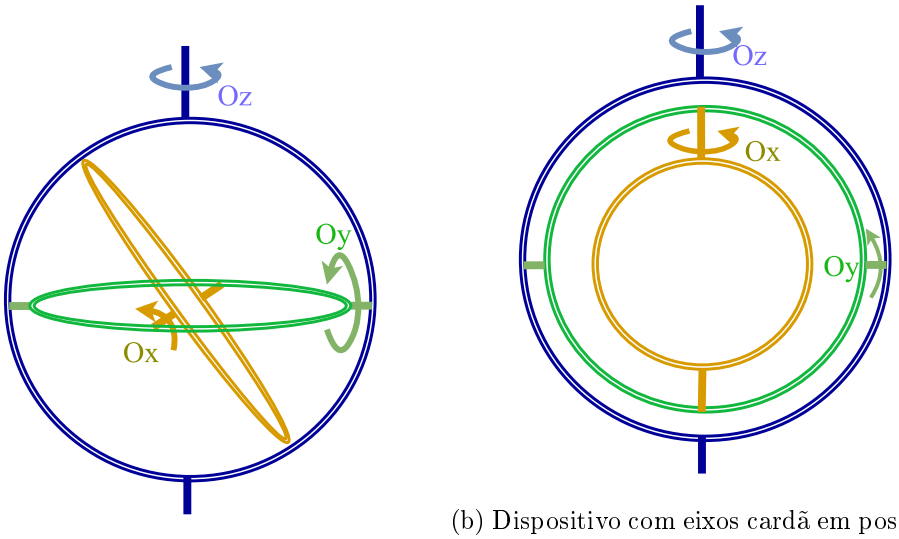
Existem diversas soluções para o problema de simular determinadas condições do ambiente espacial e os rolamentos a ar são apenas uma delas. Algumas técnicas podem ser mais aplicáveis em determinadas situações do que em outras: enquanto que, para o treinamento de astronautas, a utilização de tanques submersos mostra-se eficaz, para testes de pequenos satélites seu uso é indubitavelmente limitado.

Certamente, rolamentos a ar também possuem limitações e por isso não replicam com exatidão as condições encontradas no espaço. Dessa forma, esses dispositivos são ideais apenas para simular as condições dinâmicas existentes no espaço e são muito utilizados para a manipulação de equipamentos em plataformas com torque gravitacional reduzido. Plataformas baseadas em rolamentos a ar são capazes de simular a ausência de torque de maneira muito similar à encontrada no espaço e, consequentemente, são a solução mais utilizada nesse âmbito. Além disso, podem ser combinadas em sistemas de modo a permitir não só o movimento rotacional livre de torque, mas também o movimento translacional livre de forças.

Dentre os rolamentos a ar disponíveis, os mais utilizados são os rolamentos a ar rotacionais, que podem ser esféricos ou semiesféricos. Esses rolamentos são formados por componentes concêntricos separados apenas por uma fina camada de ar. Idealmente, tais rolamentos permitiriam o movimento rotacional 3D livre. Isso só não é possível devido à necessidade de se fixar equipamentos de teste ao rolamento, tal como a própria plataforma de simulação, o que implica que dificilmente a superfície de rotação abrangida por seu movimento alcançará 4π esferorradianos.

Outros dispositivos mecânicos, tais como juntas esféricas ou combinações de eixos cardã, tam-

bém permitem liberdade similar de movimento, mas apresentam desvantagens consideráveis. Enquanto a primeira apresenta atrito relevante, a segunda inclui singularidades conhecidas como *gimbal lock*, que são posições nas quais o dispositivo construído com eixos cardã apresenta perda de um ou mais graus de liberdade. A Figura 1.1 ilustra um dispositivo que faz uso desse tipo de eixo e suas possíveis configurações. Outro problema relacionado ao uso desses dispositivos é a necessidade de se levar em conta efeitos dinâmicos de natureza não-linear, o que tornaria simulações realistas mais difíceis [37].



(a) Dispositivo com eixos cardã em rotação.

(b) Dispositivo com eixos cardã em posição de *gimbal lock*.

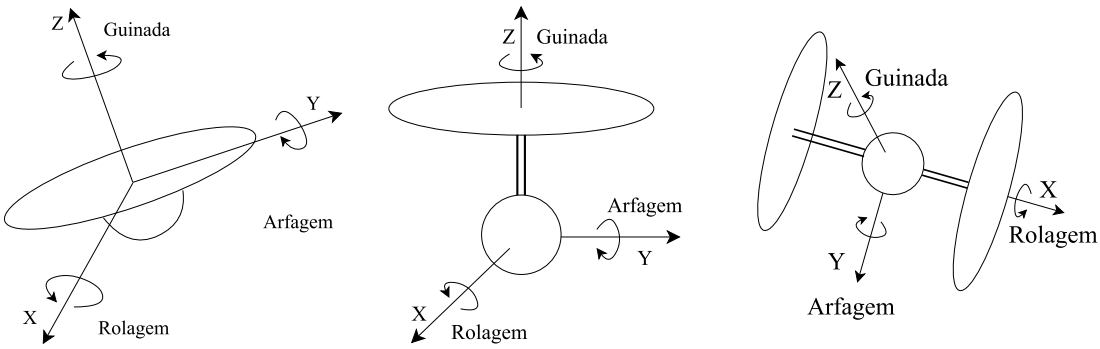
Figura 1.1: Configurações de um dispositivo com eixos cardã.

De todos os sistemas baseados em rolamentos a ar, existem dois principais conjuntos que merecem menção: sistemas planares e sistemas rotacionais.

Os sistemas planares são aqueles que apresentam dois graus de liberdade translacionais e um grau de liberdade rotacional perpendicular ao movimento planar. São especialmente úteis em simulações de manobras no espaço, tais como aproximação e atracamento de veículos espaciais. Geralmente são compostos por uma superfície polida e corpos que flutuam a partir de uma camada de ar gerada por eles mesmos.

As aplicações envolvendo sistemas planares são usualmente feitas com o intuito de validar técnicas de controle usando corpos de prova pequenos e genéricos, abrindo mão da utilização de cargas de vôo reais. Um exemplo de aplicação é o braço manipulador desenvolvido no laboratório de robótica aeroespacial da Universidade de Stanford. Esse projeto tinha como objetivo principal superar os desafios ligados às operações de construção, montagem e reparo realizadas por manipuladores em órbita [34].

Os sistemas rotacionais, por outro lado, são aqueles que apresentam três graus de liberdade rotacionais ortogonais entre si. Dentre as configurações mais conhecidas, estão: i) a configuração topo de mesa; ii) a configuração "umbela" e iii) a configuração "haltere". As figuras 1.2a, 1.2b e 1.2c ilustram essas configurações.



(a) Configuração topo de mesa. (b) Configuração umbela. (c) Configuração haltere.

Figura 1.2: Sistemas com rolamento a ar rotacionais.

A configuração topo de mesa é caracterizada por não apresentar deslocamento entre o centro de rotação do rolamento e a área de montagem do sistema. Tal configuração é formada, geralmente, por um hemisfério fixado a uma placa de montagem. A principal desvantagem dessa configuração com relação às outras duas é a de que a excursão dos ângulos de arfagem e rolagem é limitada pelos equipamentos montados no rolamento, fazendo com que tais ângulos não assumam valores superiores a $\pm 90°$.

Outra configuração usual é a configuração umbela, que leva esse nome por se assemelhar a um guarda-chuva. Nessa configuração, o hemisfério do rolamento dá lugar a um rolamento totalmente esférico separado da placa de montagem e ligado a ela por meio de uma haste. A vantagem dessa configuração está em extender o limite de movimentação dos ângulos de arfagem e rolagem.

Por fim, dentre as configurações mais usuais, encontra-se a configuração haltere, que se assemelha à configuração umbela com a exceção de que a mesa de montagem é replicada na extremidade diametralmente oposta do rolamento, ou seja, nessa configuração apresentam-se duas hastes, cada uma conectada a mesas de montagem distintas. A vantagem dessa configuração é que, diferentemente das outras duas, apenas um eixo possui limitações com relação à sua movimentação - nomeadamente, o eixo de arfagem.

Tipicamente, todas essas configurações não apresentam nenhuma limitação física capaz de restringir o movimento de guinada. Com relação às referências adotadas, em todas essas configurações o eixo de guinada é definido paralelamente ao vetor de gravidade. Nas configurações topo de mesa e umbela, os eixos de arfagem e rolagem são indistinguíveis, enquanto que, na configuração haltere, o eixo de rolagem é definido pelas hastes.

Como mencionado anteriormente, rolamentos a ar são capazes de fornecer ambientes de simulação com torque gravitacional quase nulo. Contudo, esse recurso não apresenta grande valia caso hajam outras fontes de torque no sistema. As possíveis fontes de torque foram agrupadas e são listadas a seguir.

1. Torques inerentes à plataforma, tais como desbalanceamentos estático e dinâmico, anisoelasticidade, instabilidades no material (*e.g.* tensões internas, temperatura, umidade, evaporação)

e variações de densidade.

2. Torques presentes no sistema de teste, incluindo fios presentes na plataforma, componentes mal-fixados, reposicionamento de componentes ou até mesmo descarregamento de baterias.

3. Torques no rolamento, como os provenientes da distribuição desigual da película de ar, de atritos causados por pressão insuficiente na alimentação pneumática e de sobrecarregamento da plataforma.

4. Torques provenientes do ambiente, como arrasto, correntes de ar, campos magnéticos, vibração e pressão de radiação.

As causas listadas nos itens 1 e 2 podem ser mitigadas através de uma boa elaboração do projeto do sistema. No caso dos torques no rolamento, listados no item 3, os atritos são amenizados escolhendo-se uma boa relação carga-pressão. Já a distribuição desigual na película de ar não é tão presente quanto antigamente, devido à evolução dos processos de fabricação utilizados na produção dos rolamentos. Finalmente, as causas citadas no item 4, apesar de serem dificilmente tratáveis, possuem soluções já implementadas. Em algumas instalações da NASA, por exemplo, experimentos são realizados em câmaras à vácuo, o que elimina efeitos térmicos e de arrasto. Outro exemplo é o LOACS[1], desenvolvido pela Boeing, no qual não foi possível implementar a câmara a vácuo, pois tal sistema era tripulado. Nesse caso, os efeitos ambientais foram reduzidos por meio do controle preciso da temperatura e da circulação de ar. Adicionalmente, o sistema foi isolado de distúrbios sísmicos mediante utilização de uma laje de concreto amortecida por molas pneumáticas.

Sistemas de simulação com rolamento a ar provaram ser, além de ferramentas pedagógicas valiosas, ferramentas importantes no desenvolvimento de tecnologias aeroespaciais desde os primórdios da exploração espacial. Enquanto rolamentos a ar planares são ferramentas ideais para simulação de dinâmica entre dois veículos, podendo simular técnicas de vôo, manobragem espacial, aproximação e atracamento, rolamentos a ar esféricos oferecem liberdade de experimentação de técnicas de controle, como rastreio de posição e orientação e compensação de dinâmica não-modelada.

1.2 Definição do problema e objetivos do projeto

O Simulador de Sistema de Determinação e Controle de Atitude de Pequenos Satélites desenvolvido pelo Laboratório de Aplicação e Inovação em Ciências Aeroespaciais (LAICA) é dotado de um rolamento a ar esférico. A configuração adotada no projeto foi a configuração topo de mesa, na qual o rolamento está diretamente ligado à placa de montagem do simulador.

O objetivo deste trabalho é realizar o desenvolvimento desse simulador desde sua construção até a implementação de um algoritmo de balanceamento que permita adequar o sistema às condições de simulação apropriadas.

[1]Sigla em inglês para *Lunar Orbiter Attitude Control Simulator*.

Capítulo 2

Fundamentos

Este capítulo destina-se à descrição do modelo dinâmico da mesa com rolamento a ar, das simplificações utilizadas, do método de solução das equações e, também, de todos os conceitos necessários para esse desenvolvimento.

2.1 Conceitos básicos

2.1.1 Derivada temporal de um vetor

Um conceito fundamental ao que será desenvolvido posteriormente neste capítulo diz respeito à derivada temporal de vetores quando existem dois ou mais sistemas de referência no sistema e há um movimento relativo entre eles, como no caso do sistema da mesa em que existem dois sistemas de referência: um inercial e outro fixo à mesa.

Considera-se, então, um sistema de referência inercial XYZ e outro sistema de referência em rotação xyz. O vetor $\overrightarrow{\Omega}$, medido em termos do sistema XYZ, descreve a rotação do sistema xyz. Imagina-se, em seguida, um vetor arbitrário $\overrightarrow{A}$, definido a partir do sistema xyz, e escreve-se tal vetor em termos dos versores componentes desse sistema

$$\overrightarrow{A} = A_x \overrightarrow{i} + A_y \overrightarrow{j} + A_z \overrightarrow{k} \tag{2.1}$$

em que A_x, A_y e A_z indicam as intensidades das componentes ortogonais do vetor A e $\overrightarrow{i}$, $\overrightarrow{j}$ e $\overrightarrow{k}$ representam os versores componentes do sistema de referência em rotação xyz. A Figura 2.1 ilustra essa situação.

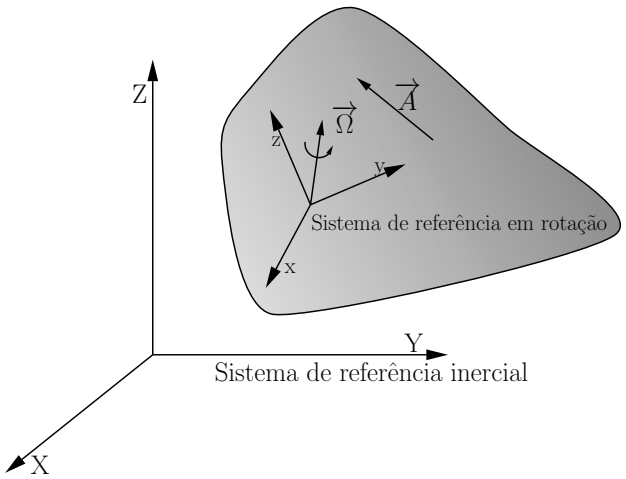


Figura 2.1: Vetor arbitrário representado em sistemas de referência inercial e em rotação.

Ao se calcular a derivada temporal de um vetor, é necessário atentar-se para a variação do mesmo com relação à direção e intensidade. No entanto, se essa derivada for tomada no sistema xyz, apenas a variação nas intensidades das componentes é considerada, pois é assumido que o vetor $\overrightarrow{A}$ é fixo ao sistema de referência xyz. Desse modo, tem-se

$$(\dot{\overrightarrow{A}})_{xyz} = \dot{A}_x \overrightarrow{i} + \dot{A}_y \overrightarrow{j} + \dot{A}_z \overrightarrow{k} \ . \tag{2.2}$$

No caso da derivada do vetor arbitrário, $\dot{\overrightarrow{A}}$, ser calculada em relação ao sistema de referência inercial fixo, há de se considerar tanto a variação em direção quanto em intensidade do vetor, conforme pode ser visto na Equação (2.3), onde nota-se o uso da regra de diferenciação do produto.

$$\dot{\overrightarrow{A}} = (\dot{A}_x \overrightarrow{i} + A_x \dot{\overrightarrow{i}}) + (\dot{A}_y \overrightarrow{j} + A_y \dot{\overrightarrow{j}}) + (\dot{A}_z \overrightarrow{k} + A_z \dot{\overrightarrow{k}}) \ . \tag{2.3}$$

Para se determinar a derivada temporal dos versores do sistema xyz, basta notar que tais diferenciações levam em conta apenas a variação na direção dos mesmos, visto que todos apresentam intensidade unitária. Define-se, então

$$\begin{aligned} \dot{\overrightarrow{i}} &= \overrightarrow{\Omega} \times \overrightarrow{i} \ , \\ \dot{\overrightarrow{j}} &= \overrightarrow{\Omega} \times \overrightarrow{j} \ , \\ \dot{\overrightarrow{k}} &= \overrightarrow{\Omega} \times \overrightarrow{k} \ . \end{aligned} \tag{2.4}$$

Utilizando o resultado das equações (2.4) e (2.2) na Equação (2.3), tem-se

$$(\dot{\overrightarrow{A}})_{XYZ} = (\dot{\overrightarrow{A}})_{xyz} + \overrightarrow{\Omega} \times \overrightarrow{A} \ . \tag{2.5}$$

O resultado da Equação (2.5) indica que a derivada de um vetor em um sistema inercial pode ser dada como a derivada desse vetor lida no sistema de referência em rotação ligado a ele somada a uma componente decorrente da rotação entre os dois sistemas de coordenadas.

2.1.2 Produto vetorial

Dentre as formas possíveis de se calcular o produto vetorial de dois vetores, duas são enfatizadas neste trabalho, conforme mostrado a seguir.

Sejam $\overrightarrow{v_1}$ e $\overrightarrow{v_2}$ dois vetores distintos de componentes tridimensionais. Esses vetores podem ser escritos da seguinte forma, em função dos versores que compõem o sistema de referência no qual estão descritos

$$\begin{aligned} \overrightarrow{v_1} &= x_1\,\overrightarrow{i} + y_1\,\overrightarrow{j} + z_1\,\overrightarrow{k}\ , \\ \overrightarrow{v_2} &= x_2\,\overrightarrow{i} + y_2\,\overrightarrow{j} + z_2\,\overrightarrow{k}\ . \end{aligned} \tag{2.6}$$

Para calcular o produto vetorial entre esses dois vetores, denotado por $\overrightarrow{v_1} \times \overrightarrow{v_2}$, pode-se utilizar o seguinte determinante

$$\begin{aligned} \overrightarrow{v_1} \times \overrightarrow{v_2} &= \left| \begin{bmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{bmatrix}_{3\times 3} \right| = \\ &= (y_1 z_2 - y_2 z_1)\,\overrightarrow{i} + (z_1 x_2 - z_2 x_1)\,\overrightarrow{j} + (x_1 y_2 - x_2 y_1)\,\overrightarrow{k}\ . \end{aligned} \tag{2.7}$$

Outra forma de se calcular o produto vetorial entre dois vetores é utilizando matrizes antissimétricas [7].

Definição 1 *Seja S uma matriz. Essa matriz é dita antissimétrica se obedece a seguinte relação:*

$$S + S^T = 0\ . \tag{2.8}$$

Essa restrição pode ser vista, também, da seguinte forma:

$$s_{ij} + s_{ji} = 0 \qquad\qquad \forall i, j\ . \tag{2.9}$$

Pela Definição 1, segue que uma matriz antissimétrica 3×3 possui elementos nulos em sua diagonal principal e, ainda, pode ser escrita da seguinte forma

$$S = \begin{bmatrix} 0 & -a & b \\ a & 0 & -c \\ -b & c & 0 \end{bmatrix}\ . \tag{2.10}$$

Logo, são necessários apenas três parâmetros para descrever completamente uma matriz antissimétrica. Define-se, então, o operador antissimétrico de um vetor. Sejam $\overrightarrow{v_{1m}} = \begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^T$ e $\overrightarrow{v_{2m}} = \begin{bmatrix} x_2 & y_2 & z_2 \end{bmatrix}^T$ as representações em vetor-coluna dos vetores $\overrightarrow{v_1}$ e $\overrightarrow{v_2}$. O operador antissimétrico do vetor $\overrightarrow{v_{1m}}$ é dado por

$$S(\overrightarrow{v_{1m}}) = S\left(\begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^T\right) = \begin{bmatrix} 0 & -z_1 & y_1 \\ z_1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} . \tag{2.11}$$

Usando esse operador, é possível escrever o produto vetorial entre $\overrightarrow{v_{1m}}$ e $\overrightarrow{v_{2m}}$ por meio da seguinte multiplicação matricial

$$\begin{aligned} \overrightarrow{v_{1m}} \times \overrightarrow{v_{2m}} = S(\overrightarrow{v_{1m}}) \cdot \overrightarrow{v_{2m}} &= \begin{bmatrix} 0 & -z_1 & y_1 \\ z_1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \\ &= \begin{bmatrix} (y_1 z_2 - y_2 z_1) \\ (z_1 x_2 - z_2 x_1) \\ (x_1 y_2 - x_2 y_1) \end{bmatrix} . \end{aligned} \tag{2.12}$$

Nota-se que, apesar de representado de forma diferente, o resultado do produto vetorial dado pela Equação (2.12) é o mesmo da Equação (2.7). Essas duas formas de se calcular o produto vetorial entre dois vetores serão utilizadas neste trabalho.

2.1.3 Modelo do pêndulo simples

A fim de validar os resultados obtidos com a execução do procedimento de balanceamento da mesa, utilizar-se-á, posteriormente, a modelagem da mesa por pêndulos simples. De maneira mais específica, o período de oscilação da mesa será observado. Nesse contexto, prossegue-se com a demonstração do período de oscilação do pêndulo simples.

Inicialmente, imagina-se um pêndulo simples como mostrado na Figura 2.2.

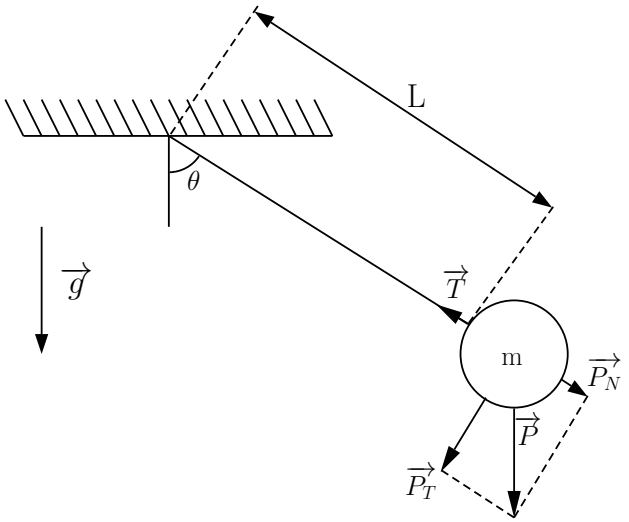


Figura 2.2: Pêndulo simples.

Nessa figura, L representa o comprimento do cabo que liga o ponto fixo - ou centro de rotação - do pêndulo à massa dele. m denota a massa pontual utilizada no pêndulo e θ é o ângulo que esse

pêndulo faz com a vertical. O conjunto é submetido à força gravitacional. As forças indicadas por $\overrightarrow{T}$ e $\overrightarrow{P}$ são a tração e o peso, respectivamente. $\overrightarrow{P}_N$ e $\overrightarrow{P}_T$ são as componentes normal e tangencial da força-peso atuante na massa no pêndulo.

Como pode ser observado na Figura 2.2, a componente tangencial da força peso gerará um torque restaurador em torno do centro de rotação, pois atuará sempre no sentido oposto ao deslocamento do pêndulo e em direção ao ponto central ($\theta = 0$), também chamado de posição de equilíbrio.

Esse torque pode ser escrito na forma

$$\tau = r_\perp \ F = -L \ |\overrightarrow{P_{\mathrm{T}}}| = -L(P \ sen\theta) \ , \tag{2.13}$$

em que τ representa o torque do pêndulo em torno do ponto fixo, $r_\perp$ representa o braço de alavanca da força F e o sinal negativo indica que o torque age no sentido de reduzir θ.

Aplicando a segunda lei de Newton para rotações, temos que

$$\tau = I\alpha \Rightarrow -L(P \ sen\theta) = I\alpha \ . \tag{2.14}$$

Donde é possível obter a expressão para a aceleração angular do pêndulo (α) como

$$\alpha = -\frac{mgL}{I} \ sen\theta \approx -\frac{mgL}{I} \ \theta \tag{2.15}$$

em que faz-se uso da aproximação $sen\theta \approx \theta$, válida para pequenos ângulos. A Tabela 2.1 mostra o erro associado a essa substituição para diferentes ângulos. No projeto da mesa com rolamento a ar os ângulos de arfagem e rolagem não ultrapassarão, por limitações físicas, $\pm 45°$. O erro associado a esse limite é de, aproximadamente, 10% (vide última linha da Tabela 2.1 em que $\theta = 45° \approx 0.7071 \ rad$).

θ (graus)	θ (rad)	$sin(\theta)$	Erro em relação a θ (%)
0°	0	0	-
5°	0.0873	0.0872	-0.13%
10°	0.1745	0.1736	-0.51%
15°	0.2618	0.2588	-1.14%
20°	0.3491	0.3420	-2.02%
25°	0.4363	0.4226	-3.14%
30°	0.5236	0.5000	-4.51%
35°	0.6109	0.5736	-6.10%
40°	0.6981	0.6428	-7.93%
45°	0.7854	0.7071	-9.97%

Tabela 2.1: Erro associado à substituição de sin(θ) por θ.

Como o pêndulo simples executa um movimento oscilatório e considera-se que não existem forças de arrasto ou atrito atuantes no sistema, pode-se modelar a posição do pêndulo como

$$s(t) = S \ cos(\omega t + \phi) \ , \tag{2.16}$$

que é uma cossenóide arbitrária de amplitude S, frequência angular ω e deslocamento de fase ϕ.

Derivando a Equação (2.16), temos que a velocidade e a aceleração são dadas por

$$\begin{aligned} v(t) &= \frac{d\ s(t)}{dt} = -\omega S\ sen(\omega t + \phi), \\ a(t) &= \frac{d\ v(t)}{dt} = -\omega^2 S\ cos(\omega t + \phi) \end{aligned} \tag{2.17}$$

e a relação entre a função de posição e a função de aceleração pode ser vista como

$$a(t) = -\omega^2 s(t)\ , \tag{2.18}$$

que, analogamente a movimentos angulares, é dada por

$$\alpha(t) = -\omega^2 \theta(t)\ . \tag{2.19}$$

Usando a Equação (2.19) para movimentos oscilatórios, a relação do período com a aceleração angular ($\omega = \frac{2\pi}{T}$) e a Equação (2.15), tem-se, finalmente, que o período do pêndulo simples pode ser dado por

$$T = 2\pi \sqrt{\frac{I}{mgL}}\ . \tag{2.20}$$

Como pode ser visto, a Equação (2.20) faz uso do momento de inércia do pêndulo. O momento de inércia para um sistema de partículas pode ser calculado da seguinte forma, para um eixo de rotação específico

$$I = \sum_i m_i \cdot d_i^2\ , \tag{2.21}$$

em que m_i é a massa de cada partícula do sistema e d é a distância da i-ésima partícula ao eixo de rotação.

Dessa forma, o momento de inércia do pêndulo é dado por

$$I = m\ L^2 \tag{2.22}$$

e a Equação (2.20) ainda pode ser simplificada para

$$T = 2\pi \sqrt{\frac{L}{g}}\ . \tag{2.23}$$

Mais detalhes podem ser vistos em [4].

2.1.4 Tensor de inércia e quantidade de momento angular

Um conceito importante relacionado à modelagem cinética de sistemas mecânicos é o do **tensor de inércia.**

Para ilustrar a origem desse conceito, prossegue-se com o desenvolvimento da formulação da quantidade de movimento angular para um corpo rígido tridimensional qualquer.

Considera-se, para isso, a Figura 2.3.

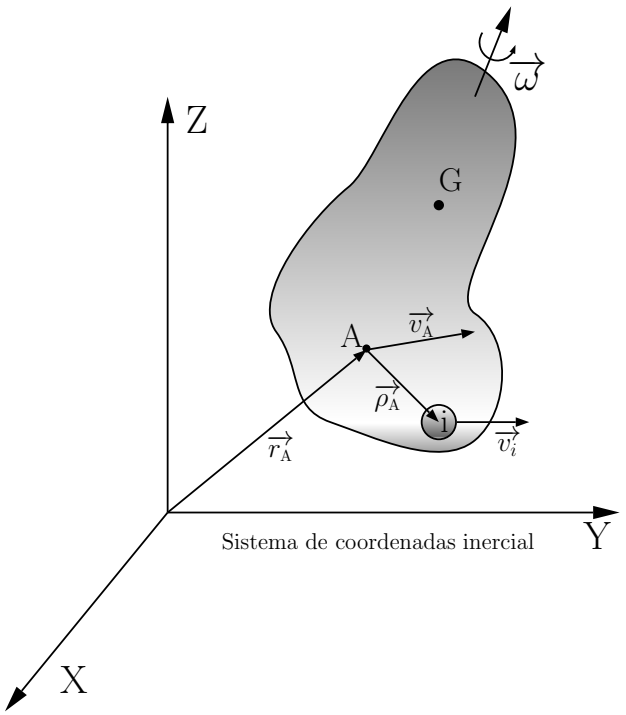


Figura 2.3: Representação de um corpo rígido em um sistema de referência inercial XYZ.

Na Figura 2.3, XYZ é um sistema de referência inercial, ou seja, seus eixos são fixos ou transladam à velocidade constante. Calcula-se o momento angular em relação a um ponto arbitrário desse corpo rígido, o qual se indica por A. O vetor $\overrightarrow{\rho_{\mathrm{A}}}$ representa o vetor-distância entre a i-ésima partícula do corpo e o ponto A, o vetor $\overrightarrow{r_{\mathrm{A}}}$ representa o vetor-distância entre a origem do sistema inercial e o ponto A, o ponto G representa a posição do centro de massa (CM) do corpo, $\overrightarrow{v_{\mathrm{A}}}$ representa a velocidade do ponto A, $\overrightarrow{\omega}$ representa o vetor velocidade angular do corpo e, por fim, $\overrightarrow{v_{\mathrm{i}}}$ representa o vetor-velocidade da i-ésima partícula do corpo.

Para a i-ésima partícula desse corpo, equaciona-se a quantidade de momento angular com relação ao ponto A

$$(\overrightarrow{H_{\mathrm{A}}})_{\mathrm{i}} = \overrightarrow{\rho_{\mathrm{A}}} \times m_{\mathrm{i}} \overrightarrow{v_{\mathrm{i}}} \ . \tag{2.24}$$

Utilizando análise de movimento relativo no corpo, é possível relacionar a velocidade da i-ésima partícula com a velocidade do ponto A da seguinte forma

$$\overrightarrow{v_{\mathrm{i}}} = \overrightarrow{v_{\mathrm{A}}} + \overrightarrow{\omega} \times \overrightarrow{\rho_{\mathrm{A}}} \tag{2.25}$$

Substituindo a Equação (2.25) na Equação (2.24), tem-se

$$(\overrightarrow{H_{\mathrm{A}}})_i = \overrightarrow{\rho_{\mathrm{A}}} \times m_{\mathrm{i}} \overrightarrow{v_{\mathrm{A}}} + \overrightarrow{\rho_{\mathrm{A}}} \times m_{\mathrm{i}} (\overrightarrow{\omega} \times \overrightarrow{\rho_{\mathrm{A}}}) \ . \tag{2.26}$$

Somando infinitesimalmente as componentes de momento angular para cada partícula do corpo, obtém-se

$$\overrightarrow{H_{\mathrm{A}}} = \left(\int_m \overrightarrow{\rho_{\mathrm{A}}}\, dm \right) \times \overrightarrow{v_{\mathrm{A}}} + \int_m \overrightarrow{\rho_{\mathrm{A}}} \times (\overrightarrow{\omega} \times \overrightarrow{\rho_{\mathrm{A}}})\, dm \ . \tag{2.27}$$

Existem duas situações particulares para a Equação (2.27). Uma delas é para o caso do ponto A ser um ponto fixo do corpo, tal como o centro de rotação, denotado como ponto O. Nesse caso, como a velocidade do ponto é nula, ou seja, $\overrightarrow{v_{\mathrm{A}}} = \overrightarrow{0}$, tem-se

$$\overrightarrow{H_{\mathrm{O}}} = \left(\int_m \overrightarrow{\rho_{\mathrm{O}}}\, dm \right) \times \cancelto{\overrightarrow{0}}{\overrightarrow{v_{\mathrm{O}}}} + \int_m \overrightarrow{\rho_{\mathrm{O}}} \times (\overrightarrow{\omega} \times \overrightarrow{\rho_{\mathrm{O}}})\, dm = \int_m \overrightarrow{\rho_{\mathrm{O}}} \times (\overrightarrow{\omega} \times \overrightarrow{\rho_{\mathrm{O}}})\, dm \ . \tag{2.28}$$

Outro caso é o de quando o ponto A é o centro de massa do corpo, ou seja, $\int_m \overrightarrow{\rho_{\mathrm{A}}}\, dm = \overrightarrow{0}$. Nesse caso, tem-se

$$\overrightarrow{H_{\mathrm{G}}} = \cancelto{\overrightarrow{0}}{\left(\int_m \overrightarrow{\rho_{\mathrm{G}}}\, dm \right)} \times \overrightarrow{v_{\mathrm{G}}} + \int_m \overrightarrow{\rho_{\mathrm{O}}} \times (\overrightarrow{\omega} \times \overrightarrow{\rho_{\mathrm{G}}})\, dm = \int_m \overrightarrow{\rho_{\mathrm{G}}} \times (\overrightarrow{\omega} \times \overrightarrow{\rho_{\mathrm{G}}})\, dm \ . \tag{2.29}$$

A Equação (2.27), que descreve o momento angular referente a um ponto arbitrário A, pode, ainda, ser simplificada em termos do momento angular relativo ao centro de massa da seguinte forma

$$\overrightarrow{H_{\mathrm{A}}} = \overrightarrow{\rho_{\mathrm{G/A}}} \times m\overrightarrow{v_{\mathrm{G}}} + \overrightarrow{H_{\mathrm{G}}} \ , \tag{2.30}$$

em que $\overrightarrow{\rho_{\mathrm{G/A}}}$ é o vetor-distância que representa a distância do ponto G com relação ao ponto A [5].

Ao analisar as equações (2.28) e (2.29), nota-se que ambas são da forma

$$\overrightarrow{H} = \int_m \overrightarrow{\rho} \times (\overrightarrow{\omega} \times \overrightarrow{\rho})\, dm \ . \tag{2.31}$$

Desenvolve-se a Equação (2.31) a fim de explicitar os termos que compõem o tensor de inércia. Para isso, utiliza-se a metodologia descrita por Donald T. Greenwood. [6]

Escreve-se o vetor posição $\overrightarrow{\rho}$ como uma decomposição cartesiana e faz-se o mesmo para o vetor de velocidade angular $\overrightarrow{\omega}$. Calcula-se, então, o produto vetorial $\overrightarrow{\omega} \times \overrightarrow{\rho}$ e, em seguida, o produto vetorial $\overrightarrow{\rho} \times (\overrightarrow{\omega} \times \overrightarrow{\rho})$

$$\overrightarrow{\rho} = x\,\overrightarrow{i} + y\,\overrightarrow{j} + z\,\overrightarrow{k} \ , \tag{2.32}$$

$$\overrightarrow{\omega} = \omega_x\,\overrightarrow{i} + \omega_y\,\overrightarrow{j} + \omega_z\,\overrightarrow{k} \ , \tag{2.33}$$

$$\overrightarrow{\omega} \times \overrightarrow{\rho} = \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ \omega_x & \omega_y & \omega_z \\ x & y & z \end{vmatrix} = (z\omega_y - y\omega_z)\,\overrightarrow{i} + (x\omega_z - z\omega_x)\,\overrightarrow{j} + (y\omega_x - x\omega_y)\,\overrightarrow{k} \ , \tag{2.34}$$

$$\overrightarrow{\rho} \times (\overrightarrow{\omega} \times \overrightarrow{\rho}) = \begin{vmatrix} \overrightarrow{i} & \overrightarrow{j} & \overrightarrow{k} \\ x & y & z \\ (z\omega_y - y\omega_z) & (x\omega_z - z\omega_x) & (y\omega_x - x\omega_y) \end{vmatrix} . \tag{2.35}$$

Resolvendo o determinante da Equação (2.35) e agrupando os termos convenientemente, tem-se

$$\begin{aligned} \overrightarrow{\rho} \times (\overrightarrow{\omega} \times \overrightarrow{\rho}) = & [(y^2 + z^2)\omega_x - xy\omega_y - xz\omega_z]\overrightarrow{i} \\ & + [-yx\omega_x + (x^2 + z^2)\omega_y - yz\omega_z]\overrightarrow{j} \\ & + [-zx\omega_x - zy\omega_y + (x^2 + y^2)\omega_z]\overrightarrow{k} \ . \end{aligned} \tag{2.36}$$

Aplicando a integral da Equação (2.31) no produto vetorial acima, tem-se

$$\begin{aligned} \int_m \overrightarrow{\rho} \times (\overrightarrow{\omega} \times \overrightarrow{\rho})\, dm = & \left[\int_m (y^2 + z^2)\omega_x\, dm + \int_m -xy\omega_y\, dm + \int_m -xz\omega_z\, dm\right]\overrightarrow{i} \\ & + \left[\int_m -yx\omega_x\, dm + \int_m (x^2 + z^2)\omega_y\, dm + \int_m -yz\omega_z\, dm\right]\overrightarrow{j} \\ & + \left[\int_m -zx\omega_x\, dm + \int_m -zy\omega_y\, dm + \int_m (x^2 + y^2)\omega_z\, dm\right]\overrightarrow{k} \ . \end{aligned} \tag{2.37}$$

Surge, neste momento, a definição dos momentos de inércia e dos produtos de inércia. Os momentos de inércia são dados como segue:

$$\begin{aligned} I_{xx} &= \textstyle\int_m (y^2 + z^2)\, dm \ , \\ I_{yy} &= \textstyle\int_m (x^2 + z^2)\, dm \ , \\ I_{zz} &= \textstyle\int_m (x^2 + y^2)\, dm \ . \end{aligned} \tag{2.38}$$

E os produtos de inércia são dados por

$$\begin{aligned} I_{xy} = I_{yx} &= -\textstyle\int_m xy\, dm \ , \\ I_{xz} = I_{zx} &= -\textstyle\int_m xz\, dm \ , \\ I_{yz} = I_{zy} &= -\textstyle\int_m yz\, dm \ . \end{aligned} \tag{2.39}$$

Partindo dessas definições e aplicando-as à Equação (2.37), é possível escrever o vetor de momento angular do corpo rígido da seguinte forma

$$\begin{aligned} \overrightarrow{H} = & (I_{xx} \cdot \omega_x + I_{xy} \cdot \omega_y + I_{xz} \cdot \omega_z)\overrightarrow{i} \\ & + (I_{yx} \cdot \omega_x + I_{yy} \cdot \omega_y + I_{yz} \cdot \omega_z)\overrightarrow{j} \\ & + (I_{zx} \cdot \omega_x + I_{zy} \cdot \omega_y + I_{zz} \cdot \omega_z)\overrightarrow{k} \ . \end{aligned} \tag{2.40}$$

Dessa forma, define-se o tensor de inércia como

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} . \tag{2.41}$$

Pode-se, então, representar a Equação (2.40) matricialmente como

$$\overrightarrow{H} = I \cdot \overrightarrow{\omega} , \tag{2.42}$$

em que I é o tensor de inércia definido na Equação (2.41) e $\overrightarrow{\omega} = [\ \omega_x \overrightarrow{i} \quad \omega_y \overrightarrow{j} \quad \omega_z \overrightarrow{k}\ ]^T$ é um vetor-coluna representando as componentes cartesianas da velocidade angular do corpo rígido.

O tensor de inércia, portanto, representa a inércia do corpo rígido a movimentos rotacionais levando-se em conta a rotação em quaisquer dos três eixos ortogonais (x, y, e z ou qualquer outra terna de versores ortogonais componentes de outro sistema de referência) [6].

2.1.5 Método dos mínimos quadrados

Ao se estabelecer sistemas lineares de equações entre variáveis que são calculadas sob condições específicas de medição ou tempo, é de se esperar que a solução desses sistemas varie um pouco, de condição para condição.

Uma forma de minimizar o erro do vetor-solução desses sistemas lineares é por meio do Método dos Mínimos Quadrados.

Seja, por exemplo, o seguinte sistema linear, representado matricialmente

$$[A]_{n\times n} \cdot [x]_{n\times 1} = [b]_{n\times 1} . \tag{2.43}$$

Sabe-se que esse sistema possui uma solução dada pelo vetor-coluna x, desde que a matriz A não possua determinante nulo. Para aplicar o método dos mínimos quadrados, expande-se esse sistema, sobreamostrando-o, de modo que o vetor-coluna x continua sendo determinado univocamente

$$[A_L]_{kn\times n} \cdot [x]_{n\times 1} = [b_L]_{kn\times 1} . \tag{2.44}$$

Para obter x, portanto, não é mais possível fazê-lo pré-multiplicando a equação pela inversa de matriz A em ambos os lados. Nesse caso, utiliza-se a pseudo-inversa de Moore-Penrose cuja definição é dada a seguir [9].

Definição 2 *Seja A uma matriz retangular $m \times n$. A sua pseudo-inversa de Moore-Penrose é definida como a matriz A^+ que satisfaz as seguintes condições*

$$\begin{aligned} AA^+A &= A \ , \\ A^+AA^+ &= A^+ \ , \\ (AA^+)^T &= AA^+ \ , \\ (A^+A)^T &= A^+A \ . \end{aligned} \tag{2.45}$$

No caso em que A é rank completo em colunas, ou seja, A^TA é invertível, a pseudo-inversa de Moore-Penrose pode ser obtida por

$$A^+ = (A^TA)^{-1}A^T \tag{2.46}$$

e diz-se que essa é a inversa à esquerda. Nesse caso, $A^+A = I$. Por outro lado, no caso em que a matriz A é rank completo em linhas, AA^T é invertível e a pseudo-inversa de Moore-Penrose, chamada de inversa à direita, é dada por

$$A^+ = A^T(AA^T)^{-1} \ . \tag{2.47}$$

Nesse caso, $AA^+ = I$.

Prosseguindo com a solução do sistema expandido dado pela Equação (2.44), basta pré-multiplicar ambos os lados da equação pela pseudo-inversa à esquerda

$$\begin{aligned} &(A_L^TA_L)^{-1}A_L^TA_L \cdot x = (A_L^TA_L)^{-1}A_L^Tb_L \Rightarrow \\ &\Rightarrow \cancel{(A_L^TA_L)^{-1}(A_L^TA_L)} \cdot x = (A_L^TA_L)^{-1}A_L^Tb_L \quad (I) \\ &\Rightarrow x = (A_L^TA_L)^{-1}A_L^Tb_L \ . \end{aligned} \tag{2.48}$$

O vetor-solução x obtido por esse método é, portanto, mais acurado, visto que parte da premissa de que os valores que compõem o sistema linear original variam, levando em conta essa variação.

2.1.6 Calibração de sensores

Sistemas de medição possuem características sistemáticas que afetam o desempenho geral do sistema. Características sistemáticas são aquelas que podem ser quantificadas graficamente ou analiticamente de forma exata.

Dentre estas, destacam-se três que são relacionadas à calibração de um sistema: faixa de operação, faixa de indicação e linha reta ideal [35].

A faixa de indicação é especificada pelos valores mínimos e máximos de entrada (E) e saída (S) de um sistema. Já a faixa de operação é dada pela máxima variação de entrada

$$E_{max} - E_{min} \ , \tag{2.49}$$

e pela máxima variação de saída

$$S_{max} - S_{min} \ . \tag{2.50}$$

A linha reta ideal é aquela que liga os pontos A (E_{min}, S_{min}) e B (E_{max}, S_{max}), mostrados na Figura 2.4, e é utilizada para quantificar as propriedades ideais de um elemento sensor, combinando as características explicadas anteriormente.

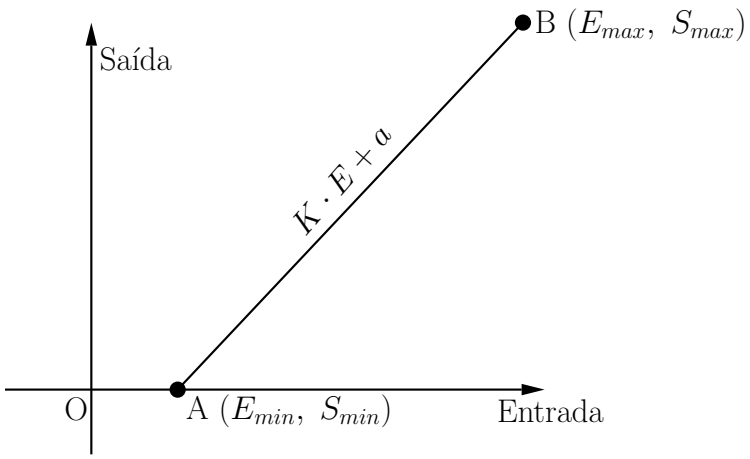


Figura 2.4: Reta de calibração ideal.

Essa reta é dada pela equação

$$S - S_{min} = \left(\frac{S_{max} - S_{min}}{E_{max} - E_{min}} \right) \cdot (E - E_{min}) \ , \tag{2.51}$$

donde obtém-se seu coeficiente angular, dado por

$$K = \frac{S_{max} - S_{min}}{E_{max} - E_{min}} \ , \tag{2.52}$$

e seu coeficiente linear, dado por

$$a = S(E_{min}) - K \cdot E_{min} \ . \tag{2.53}$$

Dessa forma, pode-se reescrever a equação da linha reta ideal na forma

$$S(E) = K \cdot E + a \ . \tag{2.54}$$

É importante ressaltar que, para que as equações descritas acima sejam válidas, assume-se que as variações nos valores de saída ocorram linearmente com as variações nos valores de entrada, ou seja, desconsidera-se qualquer não-linearidade no elemento sensor. Para o sensor utilizado neste trabalho, isso é válido e corresponde, dada certa tolerância, à realidade.

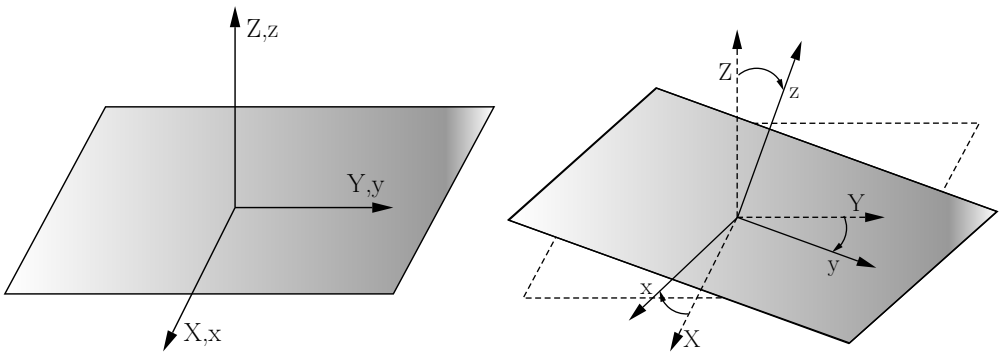
Essa reta foi utilizada para calibrar o sistema de medição presente neste trabalho e o procedimento utilizado para tal será descrito na Subseção 3.3.6.

2.2 Matrizes de rotação e sistemas inerciais

Uma das necessidades da modelagem do sistema da mesa é o de estabelecer sistemas inerciais para que se tenha uma interpretação correta das leituras dos sensores, assim como para se realizar as transformações das grandezas necessárias ao desenvolvimento dos modelos cinemático e cinético. Essa necessidade surge, em parte, da maneira como os sensores fornecem suas leituras: todas

as medições são feitas em relação ao sistema de referência próprio dos sensores e, por isso, um observador externo teria a interpretação dessas medidas dificultada.

Tendo isso em mente, são estabelecidos dois sistemas de referência para a mesa: i) uma referência inercial, fixa ao centro de rotação do rolamento a ar[1] e que não se move[2]; ii) e uma referência móvel, com origem também no centro do rotação do sistema, mas fixo à mesa e, portanto, mexendo com ela. Esses sistemas de referência são mostrados nas figuras 2.5a e 2.5b.



(a) Posição em que os sistemas de referência adotados coincidem.

(b) Posição dos sistemas de referência após rotação arbitrária da mesa.

Figura 2.5: Sistemas de referência utilizados na modelagem da mesa.

Outra necessidade ao utilizar-se dessa abordagem é a de definir uma matriz de rotação que estabeleça a relação de um sistema de coordenadas a outro por meio de alguns parâmetros. Esses parâmetros, neste caso, são os ângulos tridimensionais utilizados para se descrever a orientação do sistema móvel com relação ao sistema fixo.

Considere inicialmente o caso bidimensional, a partir do qual obtém-se o tridimensional por generalização imediata. Inicia-se, portanto, com o caso simples da rotação entre dois sistemas de referência de um ângulo qualquer, conforme pode ser visto na Figura 2.6.

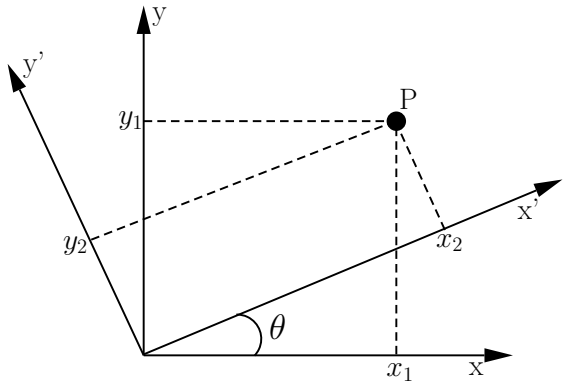


Figura 2.6: Rotação entre dois sistemas de referência.

[1]O centro de rotação da mesa com rolamento a ar está situado no ponto central da semiesfera que compõe o rolamento a ar.

[2]Na verdade, a referência inercial utilizada é fixa à Terra e essa se move. Considerar a referência fixa à Terra é, portanto, uma aproximação que, para o propósito deste trabalho, é válida.

O ponto P pode ser descrito nos dois sistemas de referência, tanto por $P = (x_1, y_1)$ no sistema xy (sistema de referência 1), quanto por $P = (x_2, y_2)$ no sistema x'y' (sistema de referência 2). As equações que descrevem as coordenadas x_2 e y_2 em termos de x_1 e y_1 são

$$\begin{aligned} x_2 &= cos(\theta) \cdot x_1 + sen(\theta) \cdot y_1 \ , \\ y_2 &= -sen(\theta) \cdot x_1 + cos(\theta) \cdot y_1 \ . \end{aligned} \tag{2.55}$$

As equações em (2.55) podem ser escritas na forma matricial como

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \underbrace{\begin{bmatrix} cos(\theta) & sen(\theta) \\ -sen(\theta) & cos(\theta) \end{bmatrix}}_{R(-\theta)} \cdot \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \ . \tag{2.56}$$

Cabe notar que a matriz R obtida é a $R(-\theta)$ e não a $R(\theta)$. Uma forma de perceber isso é notar que o ponto P no sistema de referência 2 é obtido de uma rotação no sentido **horário** do ponto P no sistema de referência 1. Por convenção, o sentido positivo das rotações é o sentido anti-horário.

Uma forma de se obter a matriz de rotação $R(\theta)$ é interpretando x_1, y_1, x_2 e y_2 como os versores que compõem os sistemas de referência representados na Figura 2.6. Assim, de forma geral, a matriz de transformação que rotaciona um vetor em θ graus pode ser obtida fazendo-se a projeção dos vetores da seguinte maneira

$$R_2^1 = \left[\ x_2^1 \ \middle| \ y_2^1 \ \right] = \begin{bmatrix} \overrightarrow{x_2} \cdot \overrightarrow{x_1} & \overrightarrow{y_2} \cdot \overrightarrow{x_1} \\ \overrightarrow{x_2} \cdot \overrightarrow{y_1} & \overrightarrow{y_2} \cdot \overrightarrow{y_1} \end{bmatrix} \ , \tag{2.57}$$

em que R_2^1 representa a rotação entre os sistemas de referência 1 e 2, x_2^1 indica o versor $\overrightarrow{x_2}$ representado no sistema de referência 1 e y_2^1 indica o versor $\overrightarrow{y_2}$ representado no sistema de referência 1.

Generalizando para o caso tridimensional, tem-se

$$R_2^1 = \left[\ x_2^1 \ \middle| \ y_2^1 \ \middle| \ z_2^1 \ \right] = \begin{bmatrix} \overrightarrow{x_2} \cdot \overrightarrow{x_1} & \overrightarrow{y_2} \cdot \overrightarrow{x_1} & \overrightarrow{z_2} \cdot \overrightarrow{x_1} \\ \overrightarrow{x_2} \cdot \overrightarrow{y_1} & \overrightarrow{y_2} \cdot \overrightarrow{y_1} & \overrightarrow{z_2} \cdot \overrightarrow{y_1} \\ \overrightarrow{x_2} \cdot \overrightarrow{z_1} & \overrightarrow{y_2} \cdot \overrightarrow{z_1} & \overrightarrow{z_2} \cdot \overrightarrow{z_1} \end{bmatrix} \ . \tag{2.58}$$

Para exemplificar o uso da Equação (2.58), demonstra-se a transformação que representa a rotação de um sistema de referência 1 a um sistema de referência 2 em θ graus em torno do eixo z do primeiro sistema. Isso é mostrado a seguir e ilustrado pela Figura 2.7.

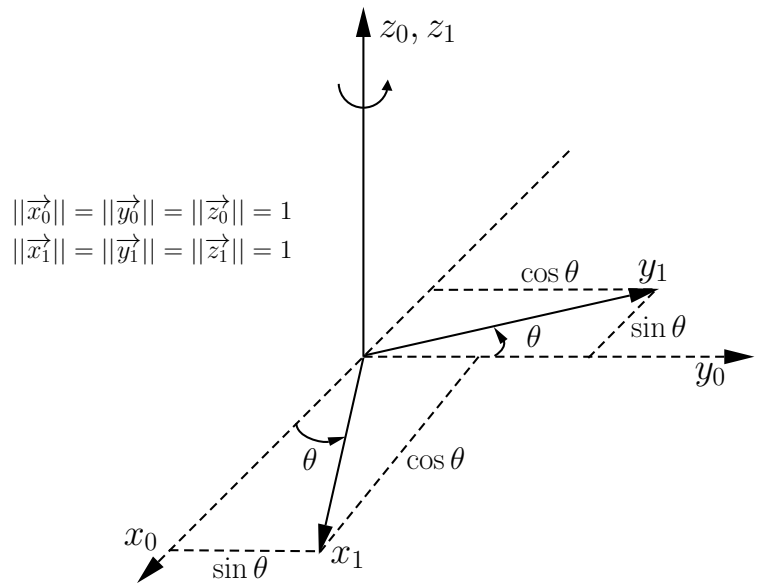


Figura 2.7: Rotação entre dois sistemas de referência tridimensionais em torno de um único eixo.

$$R_1^0 = \begin{bmatrix} \overrightarrow{x_1} \cdot \overrightarrow{x_0} & \overrightarrow{y_1} \cdot \overrightarrow{x_0} & \overrightarrow{z_1} \cdot \overrightarrow{x_0} \\ \overrightarrow{x_1} \cdot \overrightarrow{y_0} & \overrightarrow{y_1} \cdot \overrightarrow{y_0} & \overrightarrow{z_1} \cdot \overrightarrow{y_0} \\ \overrightarrow{x_1} \cdot \overrightarrow{z_0} & \overrightarrow{y_1} \cdot \overrightarrow{z_0} & \overrightarrow{z_1} \cdot \overrightarrow{z_0} \end{bmatrix} = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_{z,\theta} \ , \tag{2.59}$$

em que utilizam-se os operadores $s(\cdot) = sen(\cdot)$ e $c(\cdot) = cos(\cdot)$. Vale mencionar que se indica no subscrito da matriz de rotação o eixo de rotação e a variável que indica a magnitude da rotação. Dessa forma, $R_{z,\theta}$ denota uma rotação de θ graus em torno do eixo z.

Algumas propriedades importantes das matrizes de rotação são

1. $$\begin{cases} R^T = R^{-1} \\ R_1^0 = (R_0^1)^T \\ det(R) = 1 \end{cases} \tag{2.60}$$

2. A pós-multiplicação por uma matriz de rotação R indica uma rotação com relação ao sistema de referência atual. Uma pré-multiplicação por uma matriz de rotação R indica uma rotação com relação ao sistema de rotação inercial. Para melhor entendimento, consultar [7].

Encerra-se este tópico concluindo a utilidade das matrizes de rotação. Seja $\overrightarrow{P^0}$ um vetor-coluna representando um ponto no espaço dado em termos dos versores de um sistema de coordenadas de índice 0 e $\overrightarrow{P^1}$ um vetor-coluna representando o mesmo ponto no espaço dado em termos dos versores de um sistema de coordenadas de índice 1. Seja, ainda, R_1^0 a matriz de rotação que relaciona esses dois sistemas de coordenadas. A representação no espaço desse ponto no sistema de coordenadas de índice 0 pode ser obtida da seguinte forma

$$\overrightarrow{P^0} = R_1^0 \cdot \overrightarrow{P^1} \ , \tag{2.61}$$

ou, no sistema de coordenadas de índice 1, da seguinte forma

$$\overrightarrow{P^1} = R_0^1 \cdot \overrightarrow{P^0} = (R_1^0)^{-1} \cdot \overrightarrow{P^0} = (R_1^0)^T \cdot \overrightarrow{P^0} \ , \tag{2.62}$$

em que são utilizadas as propriedades enunciadas pelo conjunto de equações dado em (2.60) [7].

2.3 Determinação de atitude e ângulos de Euler

Existem diversas formas de representar a orientação de um corpo rígido em relação a um sistema de referência inercial, tais como a representação em ângulos de rolagem, arfagem e guinada[3], também conhecidos como ângulos de Euler e a representação eixo/ângulo. Neste texto dar-se-á enfoque na representação em ângulos de Euler [7].

No caso dessa representação, uma orientação qualquer de um corpo rígido pode ser obtida por meio de três rotações independentes em torno dos três eixos ortogonais do sistema de referência inercial. Na convenção utilizada, o ângulo de rolagem ou *roll*, ϕ, é dado por uma rotação em torno do eixo X, o ângulo de arfagem ou *pitch*, θ, é dado por uma rotação em torno do eixo Y e, por fim, o ângulo de guinada ou *yaw*, ψ, é dado por uma rotação em torno do eixo Z. Isso é ilustrado na Figura 2.8, mostrada a seguir.

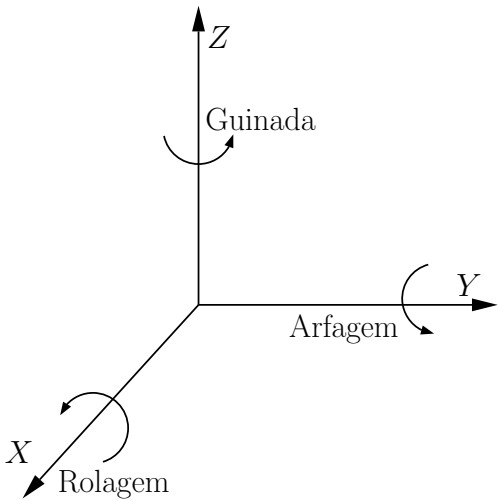


Figura 2.8: Representação em ângulos de rolagem, arfagem e guinada.

Dessa forma, uma orientação qualquer pode ser dada pela seguinte matriz de rotação:

$$\begin{aligned} R = R_b^i &= R_{Z,\psi} \cdot R_{Y,\theta} \cdot R_{X,\phi} = \\ &= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} = \\ &= \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} , \end{aligned} \tag{2.63}$$

[3] Do inglês, *roll, pitch and yaw*.

em que, novamente, denotam-se os operadores trigonométricos por sua forma abreviada dada por $s(\cdot) = s_{(\cdot)} = sen(\cdot)$ e $c(\cdot) = c_{(\cdot)} = cos(\cdot)$.

Cabe ressaltar que, como as rotações nessa representação são feitas em relação ao sistema de referência inercial, cada rotação subsequente é feita pré-multiplicando-se a matriz correspondente. Dessa forma, há duas interpretações para a matriz R dada pela Equação (2.63): rotações sucessivas nos eixos Z, Y e X com relação ao sistema de referência atual ou rotações sucessivas nos eixos X, Y e Z com relação ao sistema de referência inercial.

Seguindo o raciocínio desenvolvido na Seção 2.2, pode-se determinar a orientação de um ponto da mesa da seguinte forma

$$\overrightarrow{P^b} = R_i^b \cdot \overrightarrow{P^i} \ . \tag{2.64}$$

Para obter a matriz R_i^b, basta aplicar as propriedades descritas nas equações em (2.60) à matriz dada pela Equação (2.63). Tem-se, portanto

$$R_i^b = (R_b^i)^{-1} = (R_b^i)^T = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & c_\theta s_\phi \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\theta c_\phi \end{bmatrix} \ . \tag{2.65}$$

A matriz dada pela Equação (2.65) se mostrará muito útil no decorrer deste trabalho, visto que ela será utilizada para transformar vetores referidos ao sistema de referência inercial em vetores referidos ao sistema de referência móvel fixado à mesa.

2.4 Determinação dos ângulos de Euler através da matriz de rotação

Nesta seção explicar-se-á um dos métodos possíveis para obtenção dos ângulos de arfagem, rolagem e guinada através das matrizes de rotação e dos dados de aceleração e campo magnético de um sistema genérico. Esse método foi retirado de [30] e é chamado *tilt-compensated eCompass algorithm*.

Para esse fim, supõe-se, primeiramente, que o sistema possui um acelerômetro e um magnetômetro alinhados aos eixos inerciais, que foram representados anteriormente pela Figura 2.8. Além disso, assume-se que a única aceleração presente no sistema é a aceleração da gravidade, que possui a mesma direção e sentido oposto ao eixo Z inercial. Por fim, supõe-se que o eixo X aponta para o Polo Norte geomagnético.

Nessas condições, as medições do acelerômetro, G_r, e do magnetômetro, B_r, são

$$B_r = B \cdot \begin{bmatrix} c_\delta \\ 0 \\ -s_\delta \end{bmatrix} \ , \tag{2.66}$$

$$G_r = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} . \tag{2.67}$$

Onde g é a gravidade local, B é a intensidade do campo magnético da Terra e δ é o ângulo de inclinação do campo magnético medido. Este último varia de -90° do Polo Sul geomagnético, até 90° no Polo Norte geomagnético.

Vale ressaltar que o entendimento detalhado da variação do campo magnético na Terra não é relevante para esta demonstração, pois esses valores se cancelam ao longo do cálculo dos ângulos de Euler.

Aplicando-se a matriz de rotação R_i^b, proveniente da Equação (2.65), às medições G_r e B_r, obtém-se

$$B_p = R_i^b \cdot B_r = (R_{z,\psi})^{-1} \cdot (R_{y,\theta})^{-1} \cdot (R_{x,\phi})^{-1} \cdot B_r \ , \tag{2.68}$$

$$G_p = R_i^b \cdot G_r = (R_{z,\psi})^{-1} \cdot (R_{y,\theta})^{-1} \cdot (R_{x,\phi})^{-1} \cdot G_r \ . \tag{2.69}$$

O próximo passo do método consiste em calcular os ângulos de rolagem, ϕ, e arfagem, θ, através das medições do acelerômetro. Com esse fim, pré-multiplica-se a Equação (2.68) por $R_{x,\phi}$ e $R_{y,\theta}$, obtendo-se

$$\begin{aligned} R_{y,\theta} \cdot R_{x,\phi} \cdot G_p = R_{y,\theta} \cdot R_{x,\phi} \cdot \begin{bmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{bmatrix} = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \cdot \begin{bmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{bmatrix} = \\ = \begin{bmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \cdot \begin{bmatrix} G_{px} \\ G_{py} \\ G_{pz} \end{bmatrix} = (R_{z,\psi})^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} , \end{aligned} \tag{2.70}$$

em que G_{px}, G_{py} e G_{pz} são as componentes de G_p.

Utilizando a segunda linha da Equação (2.70), calcula-se o ângulo de rolagem

$$G_{py}c_\phi - G_{pz}s_\phi = 0 \Rightarrow \tan\phi = \frac{G_{py}}{G_{pz}} \ . \tag{2.71}$$

Além disso, pode-se calcular o ângulo de arfagem por meio da primeira linha da Equação (2.70),

$$G_{px}c_\theta + G_{py}s_\phi s_\theta + G_{pz}s_\theta c_\phi = 0 \Rightarrow \tan\theta = \frac{-G_{px}}{G_{py}s_\phi + G_{pz}c_\phi} \ . \tag{2.72}$$

Dessa forma, demonstrou-se como é possível calcular os ângulos de rolagem, ϕ, e arfagem, θ, por meio da Equação (2.70).

Com esses valores, é possível obter o ângulo de guinada, ψ, aplicando-se uma rotação inversa aos dados obtidos pelo magnetômetro, utilizando a Equação (2.68)

$$
\begin{aligned}
R_{z,\phi}^{-1} \cdot \begin{bmatrix} Bc_\delta \\ 0 \\ -Bs_\delta \end{bmatrix} &= \begin{bmatrix} c_\psi Bc_\delta \\ -s_\psi Bc_\delta \\ -Bs_\delta \end{bmatrix} = R_{y,\theta} \cdot R_{x,\phi} \cdot B_p = \\
&= \begin{bmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \cdot \begin{bmatrix} B_{px} \\ B_{py} \\ B_{pz} \end{bmatrix} = \begin{bmatrix} c_\psi Bc_\delta \\ -s_\psi Bc_\delta \\ -Bs_\delta \end{bmatrix} = \\
&= \begin{bmatrix} B_{px}c_\theta + B_{py}s_\theta s_\phi + B_{pz}s_\theta c_\phi \\ B_{py}c_\phi - B_{pz}s_\phi \\ -B_{px}s_\theta + B_{py}c_\theta s_\phi + B_{pz}c_\theta c_\phi \end{bmatrix} = \begin{bmatrix} B_{fx} \\ B_{fy} \\ B_{fz} \end{bmatrix} ,
\end{aligned}
\tag{2.73}
$$

em que as componentes B_{fx}, B_{fy} e B_{fz} são as componentes das medições do magnetômetro quando o sistema está em um estado no qual θ e ϕ são nulos.

Por fim, calcula-se o ângulo de guinada através da primeira e da segunda componentes da Equação (2.73)

$$
\begin{cases} c_\psi Bc_\delta = B_{fx}, \\ -s_\psi Bc_\delta = B_{fy} \end{cases} \Rightarrow \tan\psi = \frac{-B_{fy}}{B_{fx}} = \frac{B_{pz}s_\phi - B_{py}c_\phi}{B_{px}c_\theta + B_{py}s_\theta s_\phi + B_{pz}s_\theta c_\phi} . \tag{2.74}
$$

Assim, pode-se calcular o valor de ψ tendo como referência o pólo magnético terrestre.

2.5 Derivada temporal do momento angular em torno do centro de rotação

Um resultado que será utilizado posteriormente para estabelecer a modelagem dinâmica da mesa com rolamento a ar é o da derivada temporal do vetor momento angular em torno do centro de rotação para um corpo rígido arbitrário. Esse resultado é demonstrado a seguir. Para isso, utilizar-se-á o procedimento descrito na Subseção 2.1.1, que trata de derivadas vetoriais em contextos nos quais existem um sistema de referência inercial e outro fixado ao corpo em movimento.

Como enunciado na Equação (2.30), o momento angular com relação a um ponto arbitrário é dado por

$$
\overrightarrow{H_{\mathrm{A}}} = \overrightarrow{\rho_{\mathrm{G/A}}} \times M\overrightarrow{v_{\mathrm{G}}} + \overrightarrow{H_{\mathrm{G}}} . \tag{2.75}
$$

Tomando o centro de rotação do corpo como o ponto A em questão, tem-se

$$
\overrightarrow{H_{\mathrm{O}}} = \overrightarrow{\rho_{\mathrm{G/O}}} \times M\overrightarrow{v_{\mathrm{G}}} + \overrightarrow{H_{\mathrm{G}}} , \tag{2.76}
$$

em que utiliza-se o subscrito O para denotar o centro de rotação.

Realizando a diferenciação da Equação (2.76) em relação ao tempo, tem-se

$$
\frac{d\overrightarrow{H_{\mathrm{O}}}}{dt} = \frac{d}{dt}\left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right) + \frac{d\overrightarrow{H_{\mathrm{G}}}}{dt} , \tag{2.77}
$$

em que utiliza-se $\overrightarrow{r}$ para representar $\overrightarrow{\rho_{\mathrm{G/O}}}$, já que ambos, o vetor de desbalanceamento $\overrightarrow{r}$ e o vetor-distância $\overrightarrow{\rho_{\mathrm{G/O}}}$, representam o deslocamento do centro de massa (G) em relação ao centro de rotação (O). Nota-se, também, a substituição $\overrightarrow{v_G} = \dot{\overrightarrow{r}}$.

As derivadas envolvidas no cálculo são dadas por

$$\begin{aligned} \frac{d\overrightarrow{H_{\mathrm{G}}}}{dt} &= (\dot{\overrightarrow{H_{\mathrm{G}}}})_{\mathrm{xyz}} + \overrightarrow{\omega} \times \overrightarrow{H_{\mathrm{G}}} \ , \\ \frac{d}{dt}\left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right) &= \left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right)'_{xyz} + \overrightarrow{\omega} \times \left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right) \ . \end{aligned} \tag{2.78}$$

Assim, substituindo as equações em (2.78) na Equação (2.77), tem-se

$$\frac{d\overrightarrow{H_{\mathrm{O}}}}{dt} = \left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right)'_{xyz} + (\dot{\overrightarrow{H_{\mathrm{G}}}})_{\mathrm{xyz}} + \left[\overrightarrow{\omega} \times \left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right)\right] + \left(\overrightarrow{\omega} \times \overrightarrow{H_{\mathrm{G}}}\right) \ . \tag{2.79}$$

A fim de simplificar um pouco mais a Equação (2.79) calcula-se o termo $\left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right)'_{xyz}$

$$\begin{aligned} \left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right)'_{xyz} &= \frac{d}{dt}(\overrightarrow{r}) \times M\dot{\overrightarrow{r}} + \overrightarrow{r} \times \frac{d}{dt}(M\dot{\overrightarrow{r}}) = \\ &= M\underbrace{(\dot{\overrightarrow{r}} \times \dot{\overrightarrow{r}})}_{=\overrightarrow{0}} + \overrightarrow{r} \times M\ddot{\overrightarrow{r}} = \\ &= \overrightarrow{r} \times M\ddot{\overrightarrow{r}} \end{aligned} \tag{2.80}$$

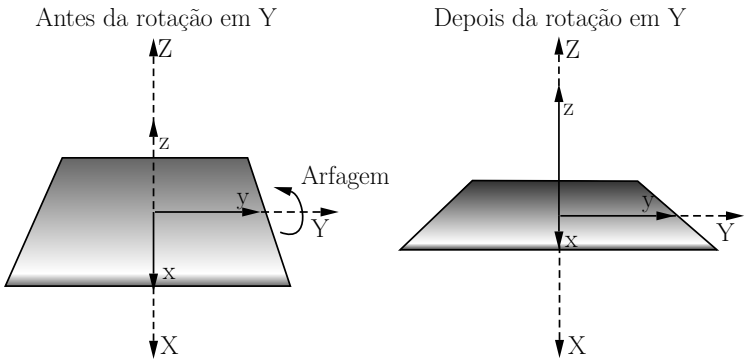
e a Equação (2.79) pode ser reescrita como

$$\frac{d\overrightarrow{H_{\mathrm{O}}}}{dt} = (\overrightarrow{r} \times M\ddot{\overrightarrow{r}}) + (\dot{\overrightarrow{H_{\mathrm{G}}}})_{\mathrm{xyz}} + \left[\overrightarrow{\omega} \times \left(\overrightarrow{r} \times M\dot{\overrightarrow{r}}\right)\right] + \left(\overrightarrow{\omega} \times \overrightarrow{H_{\mathrm{G}}}\right) \ . \tag{2.81}$$

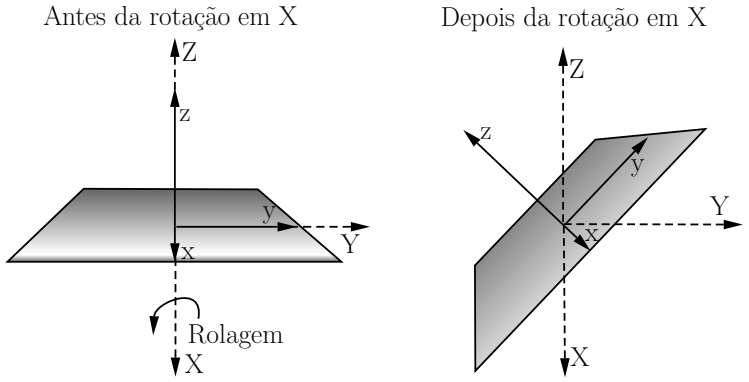
2.6 Equacionamento dinâmico da mesa

Trata-se, agora, do equacionamento dinâmico da mesa com rolamento a ar, levando-se em conta todas as variáveis envolvidas no processo, mensuráveis por meio dos sensores.

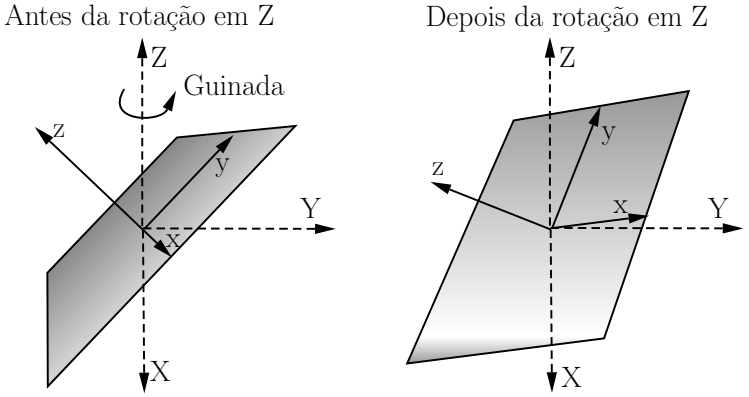
A mesa com rolamento a ar pode ser modelada como um corpo rígido com centro de massa (CM) situado a uma certa distância de seu centro de rotação (CR). Essa distância pode ser descrita pelo vetor $\overrightarrow{r}$, ao qual dar-se-á o nome de vetor de desbalanceamento. Como mostrado na Figura 2.5b, existem dois sistemas de referência no modelo da mesa. Um deles é inercial e fixo com relação à Terra, enquanto o outro rotaciona mantendo uma orientação fixa em relação à mesa. As figuras 2.9a, 2.9b e 2.9c mostram a representação desses eixos quando, a partir de uma posição inicial arbitrária, a mesa sofre rotações sucessivas nos eixos de arfagem, rolagem e guinada, nessa ordem.



(a) Orientação da mesa após um giro no eixo de arfagem (eixo Y inercial).



(b) Orientação da mesa após um giro no eixo de rolagem (eixo X inercial) partindo da configuração atingida na Figura 2.9a.



(c) Orientação da mesa após um giro no eixo de guinada (eixo Z inercial) partindo da configuração atingida na Figura 2.9b.

Figura 2.9: Sistemas de referência utilizados na modelagem.

A partir deste momento, abrevia-se o sistema de referência inercial com a letra "i" (referente, do inglês, a "*inertial frame*") e o sistema de referência fixo à mesa com a letra "b" (referente, do inglês, a "*body frame*". A orientação do sistema de referência b pode ser dada por meio dos ângulos de rolagem, arfagem e guinada ou ângulos de Euler 321, que é a forma enfatizada anteriormente neste capítulo.

Para iniciar o desenvolvimento das equações do movimento para a mesa, enuncia-se a fórmula

do momento angular da mesa em torno do centro de rotação,

$$\overrightarrow{H_{\mathrm{CR}}} = \overrightarrow{H_{\mathrm{O}}} = M(\overrightarrow{r} \times \overrightarrow{V}) \ , \tag{2.82}$$

em que M representa a massa total do sistema, $\overrightarrow{r}$ é a distância do CR ao CM e $\overrightarrow{V}$ é a velocidade linear do CM.

A equação que governa o sistema é a de que o momento externo resultante é igual à derivada do momento angular. Isso decorre da segunda lei de Newton aplicada ao movimento rotacional,

$$\overrightarrow{\tau_{\mathrm{CR}}} = \frac{d\overrightarrow{H_{\mathrm{CR}}}}{dt} \ , \tag{2.83}$$

em que $\overrightarrow{\tau_{\mathrm{CR}}}$ é o torque externo resultante aplicado ao sistema da mesa. O torque externo resultante pode ser escrito como a combinação de três torques externos, conforme a Equação (2.84),

$$\overrightarrow{\tau_{\mathrm{CR}}} = \overrightarrow{\tau_{\mathrm{disp}}} + \overrightarrow{\tau_{\mathrm{G}}} + \overrightarrow{\tau_{\mathrm{aero}}} \ , \tag{2.84}$$

em que $\overrightarrow{\tau_{\mathrm{disp}}}$ representa o torque externo aplicado por dispositivos que podem, eventualmente, estar ligados à mesa, tais como rodas de reação e propulsores a gás, $\overrightarrow{\tau_{\mathrm{G}}}$ representa o torque externo gravitacional, o qual se deseja anular para validar o uso da mesa, e $\overrightarrow{\tau_{\mathrm{aero}}}$ representa o torque de arrasto aerodinâmico.

A derivada do momento angular em torno do centro de rotação, como demonstrado na Seção 2.5, por sua vez, é dada por

$$\frac{d\overrightarrow{H_{\mathrm{O}}}}{dt} = (\overrightarrow{r} \times M\dot{\overrightarrow{r}}) + [\overrightarrow{\omega} \times (\overrightarrow{r} \times M\dot{\overrightarrow{r}})] + \dot{\overrightarrow{H_{\mathrm{G}}}} + (\overrightarrow{\omega} \times \overrightarrow{H_{\mathrm{G}}}) \ . \tag{2.85}$$

Os vetores utilizados nas equações (2.84) e (2.85) são definidos como

$$\overrightarrow{H_{\mathrm{G}}} = I \cdot \overrightarrow{\omega} = \begin{pmatrix} I_{xx}\omega_x + I_{xy}\omega_y + I_{xz}\omega_z \\ I_{yx}\omega_x + I_{yy}\omega_y + I_{yz}\omega_z \\ I_{zx}\omega_x + I_{zy}\omega_y + I_{zz}\omega_z \end{pmatrix} \ , \tag{2.86}$$

$$\overrightarrow{\tau_{\mathrm{disp}}} = \begin{bmatrix} \overrightarrow{\tau_{disp_x}} & \overrightarrow{\tau_{disp_y}} & \overrightarrow{\tau_{disp_z}} \end{bmatrix}^T \ , \tag{2.87}$$

$$\overrightarrow{\tau_{\mathrm{G}}} = M \cdot g \cdot \begin{bmatrix} 0 & -c_\phi c_\theta & s_\phi c_\theta \\ c_\phi c_\theta & 0 & s_\theta \\ -s_\phi c_\theta & -s_\theta & 0 \end{bmatrix} \cdot \overrightarrow{r} \ , \tag{2.88}$$

$$\overrightarrow{\tau_{\mathrm{aero}}} = \begin{pmatrix} -B_{\mathrm{X}}\omega_x^2 \\ -B_{\mathrm{Y}}\omega_y^2 \\ -B_{\mathrm{Z}}\omega_z^2 \end{pmatrix} \ . \tag{2.89}$$

Fazendo as devidas manipulações, pode-se escrever a Equação (2.85) como

$$A \cdot \dot{\overrightarrow{\omega}} + B = M \ , \tag{2.90}$$

em que as matrizes A, B e M são dadas por

$$A = \begin{bmatrix} Mr_y^2 + Mr_z^2 + I_{xx} & -Mr_x r_y + I_{xy} & -Mr_x r_z + I_{xz} \\ -Mr_x r_y + I_{xy} & Mr_x^2 + Mr_z^2 + I_{yy} & -Mr_y r_z + I_{yz} \\ -Mr_x r_z + I_{xz} & -Mr_y r_z + I_{yz} & Mr_x^2 + Mr_y^2 + I_{zz} \end{bmatrix}, \tag{2.91}$$

$$M = \begin{bmatrix} \overrightarrow{\tau_{disp_x}} \\ \overrightarrow{\tau_{disp_y}} \\ \overrightarrow{\tau_{disp_z}} \end{bmatrix} + \begin{bmatrix} -B_{\mathrm{X}}\omega_x^2 \\ -B_{\mathrm{Y}}\omega_y^2 \\ -B_{\mathrm{Z}}\omega_z^2 \end{bmatrix} + \begin{bmatrix} -Mgr_y c_\phi c_\theta + Mgr_z s_\phi c_\theta \\ Mgr_x c_\phi c_\theta + Mgr_z s_\theta \\ -Mgr_x s_\phi c_\theta - Mgr_y s_\theta \end{bmatrix}, \tag{2.92}$$

$$B = \begin{bmatrix} [(-2Mr_y r_z + I_{zy})\omega_y^2 + (2Mr_y r_z - I_{yz})\omega_z^2 + \\ +(-Mr_x r_z + I_{xz})\omega_x\omega_y + (Mr_x r_y - I_{xy})\omega_x\omega_z + \\ +(Mr_y^2 - Mr_z^2 - I_{yy} + I_{zz})\omega_y\omega_z] \\ \hline [(2Mr_x r_z - I_{zx})\omega_x^2 + (-2Mr_x r_z + I_{xz})\omega_z^2 + \\ +(Mr_y r_z - I_{zy})\omega_x\omega_y + (-Mr_x r_y + I_{xy})\omega_y\omega_z + \\ +(-Mr_x^2 - Mr_z^2 + I_{xx} - I_{zz})\omega_x\omega_z] \\ \hline [(-2Mr_x r_y + I_{xy})\omega_x^2 + (2Mr_x r_y - I_{xy})\omega_y^2 + \\ +(-Mr_y r_z + I_{yz})\omega_x\omega_z + (-Mr_x r_z + I_{xz})\omega_y\omega_z + \\ +(Mr_x^2 - Mr_y^2 - I_{xx} + I_{yy})\omega_x\omega_y] \end{bmatrix}_{3\times 1}. \tag{2.93}$$

Isolando $\dot{\overrightarrow{\omega}}$ na Equação (2.90), obtém-se a seguinte fórmula

$$\dot{\overrightarrow{\omega}} = (A)^{-1} \cdot (M - B) \ . \tag{2.94}$$

A Equação (2.94) pode ser integrada e resolvida simultaneamente com as variações dos ângulos de Euler [3].

2.7 Simplificação da equação do movimento da mesa

A Equação (2.94) descreve o movimento da mesa, porém apresenta muitos termos que podem ser desconsiderados se ela for submetida a uma condição de movimento específica. Um dos benefícios de se fazer isso é o de reduzir drasticamente o esforço computacional necessário para o processamento dos dados envolvidos. Cabe ressaltar, ainda, que alguns desses termos possuem magnitude desprezível e pouca precisão. Prossegue-se, então, com a simplificação dessa equação.

Para determinar o vetor-distância entre o CM e o CR da mesa, é necessário desenvolver uma forma pela qual, utilizando os dados obtidos pelos sensores, seja possível calcular esse vetor. A forma desenvolvida envolve a utilização das equações de movimento da mesa, tendo em mente que é possível obter os ângulos de orientação, as velocidades e as acelerações angulares da mesa. Inicia-se com a segunda lei de Newton no contexto de movimento rotacional, ou seja,

$$\tau = I \cdot \alpha = I \cdot \dot{\overrightarrow{\omega}} \ . \tag{2.95}$$

Em seguida, por motivos que se tornarão claros posteriormente, isola-se o vetor de velocidade angular, obtendo

$$\dot{\overrightarrow{\omega}} = I^{-1} \cdot \tau \ . \tag{2.96}$$

Considerando-se que, neste primeiro momento, não existem dispositivos acoplados à mesa capazes de gerar um torque externo e que o torque de arrasto aerodinâmico pode ser considerado desprezível, pois a mesa movimentar-se-á a velocidades suficientemente pequenas, simplifica-se a Equação (2.84), obtendo

$$\tau = \overrightarrow{\tau_{\mathrm{CR}}} = \cancelto{\overrightarrow{0}}{\overrightarrow{\tau_{\mathrm{disp}}}} + \overrightarrow{\tau_{\mathrm{G}}} + \cancelto{\overrightarrow{0}}{\overrightarrow{\tau_{\mathrm{aero}}}} = \tau_{\mathrm{G}} \ . \tag{2.97}$$

O cálculo do torque gerado em torno do centro de rotação da mesa é dado pelo seguinte produto vetorial

$$\tau_{\mathrm{G}} = \overrightarrow{r} \times \overrightarrow{F} = \overrightarrow{r} \times m\overrightarrow{g} \ . \tag{2.98}$$

Uma necessidade que surge, agora, é de representar todos os vetores no mesmo sistema de referência. Escolhe-se trabalhar no sistema de referência acoplado à mesa, ou seja, o sistema de referência b ("*body frame*"). Para isso, é necessário representar o vetor de aceleração gravitacional nesse sistema. Sabe-se que, no sistema inercial, esse vetor é dado por

$$(\overrightarrow{g})^i = g \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \ , \tag{2.99}$$

sendo que a utilização do sobrescrito i indica que esse vetor está referido no sistema de referência inercial.

Para representá-lo no sistema de referência da mesa, realiza-se a seguinte transformação

$$\begin{aligned} (\overrightarrow{g})^b &= R_i^b \cdot (\overrightarrow{g})^i = \\ &= \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & c_\theta s_\phi \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\theta c_\phi \end{bmatrix} \cdot g \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \\ &= g \cdot \begin{bmatrix} s_\theta \\ -c_\theta s_\phi \\ -c_\theta c_\phi \end{bmatrix} \ , \end{aligned} \tag{2.100}$$

em que utiliza-se a matriz de rotação dada pela Equação (2.65).

Calcula-se, então, o produto vetorial da Equação (2.98) utilizando o operador de matriz antis-

simétrica

$$\begin{aligned}\tau_{\mathrm{G}} &= \overrightarrow{r} \times m\overrightarrow{g} = S(\overrightarrow{r}) \cdot mg \begin{bmatrix} s_\theta \\ -c_\theta s_\phi \\ -c_\theta c_\phi \end{bmatrix} = \\ &= mg \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \begin{bmatrix} s_\theta \\ -c_\theta s_\phi \\ -c_\theta c_\phi \end{bmatrix} = \\ &= mg \begin{bmatrix} r_z c_\theta s_\phi - r_y c_\theta c_\phi \\ r_z s_\theta + r_x c_\theta c_\phi \\ -r_y s_\theta - r_x c_\theta s_\phi \end{bmatrix} .\end{aligned} \tag{2.101}$$

Antes de substituir esse resultado na Equação (2.96), é necessário determinar a inversa da matriz do tensor de inércia. Assumindo que os produtos de inércia são desprezíveis em relação as momentos principais de inércia, o tensor de inércia pode ser escrito da seguinte forma

$$I \approx \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} . \tag{2.102}$$

Como essa matriz é diagonal, sua inversa é dada invertendo-se os termos da diagonal principal. Logo, tem-se

$$I^{-1} = \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} . \tag{2.103}$$

Substituindo os resultados das equações (2.103) e (2.101) na Equação (2.96), obtém-se, finalmente

$$\begin{aligned}\dot{\overrightarrow{\omega}} &= mg \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} r_z c_\theta s_\phi - r_y c_\theta c_\phi \\ r_z s_\theta + r_x c_\theta c_\phi \\ -r_y s_\theta - r_x c_\theta s_\phi \end{bmatrix} = \\ &= \begin{bmatrix} \frac{mg}{I_{xx}}(-r_y c_\phi c_\theta + r_z s_\phi c_\theta) \\ \frac{mg}{I_{yy}}(r_x c_\phi c_\theta + r_z s_\theta) \\ \frac{mg}{I_{zz}}(-r_x s_\phi c_\theta - r_y s_\theta) \end{bmatrix} .\end{aligned} \tag{2.104}$$

2.8 Cálculos computacionais e aplicação do método dos mínimos quadrados

Para balancear a mesa com rolamento a ar, é necessário, primeiro, ter conhecimento do vetor-distância que localiza o CM da mesa com relação ao CR dela. Utiliza-se, portanto, da equação simplificada do movimento da mesa juntamente com os dados obtidos pelos sensores e a estimação do tensor de inércia para obter as componentes desse vetor. A Equação (2.104), assumindo que as acelerações angulares nos três eixos de rotação da mesa - XYZ - são, aproximadamente, constantes

em um curto intervalo de tempo, pode ser vista da seguinte forma

$$\dot{\overrightarrow{\omega}} = \frac{\Delta\overrightarrow{\omega}}{\Delta t} = \frac{1}{\Delta t}\begin{bmatrix}\Delta\omega_x\\ \Delta\omega_y\\ \Delta\omega_z\end{bmatrix} = \frac{1}{t_2-t_1}\begin{bmatrix}\omega_{x_{t2}}-\omega_{x_{t1}}\\ \omega_{y_{t2}}-\omega_{y_{t1}}\\ \omega_{z_{t2}}-\omega_{z_{t1}}\end{bmatrix}\ , \tag{2.105}$$

em que t_1 e t_2 são dois instantes de tempo consecutivos aos quais se atribuem conjuntos de medições tomados nesses instantes.

Sendo assim, é possível, utilizando os resultados das equações (2.104) e (2.105), escrever

$$\begin{aligned}
(\omega_x)_{t_2} - (\omega_x)_{t_1} &= \frac{-mg\Delta t}{2I_{xx}}\left\{\left[(c_\phi c_\theta)_{t_2} + (c_\phi c_\theta)_{t_1}\right] r_y - \left[(s_\phi c_\theta)_{t_2} + (s_\phi c_\theta)_{t_1}\right] r_z\right\}\ ,\\
(\omega_y)_{t_2} - (\omega_y)_{t_1} &= \frac{mg\Delta t}{2I_{yy}}\left\{\left[(c_\phi c_\theta)_{t_2} + (c_\phi c_\theta)_{t_1}\right] r_x + \left[(s_\theta)_{t_2} + (s_\theta)_{t_1}\right] r_z\right\}\ ,\\
(\omega_z)_{t_2} - (\omega_z)_{t_1} &= \frac{-mg\Delta t}{2I_{zz}}\left\{\left[(s_\phi c_\theta)_{t_2} + (s_\phi c_\theta)_{t_1}\right] r_x + \left[(s_\theta)_{t_2} + (s_\theta)_{t_1}\right] r_y\right\}\ ,
\end{aligned} \tag{2.106}$$

em que nota-se que a derivada do vetor $\overrightarrow{\omega}$ em tempo discreto é aproximada pela média aritmética do valor do lado direito da Equação (2.104) tomado em dois instantes de tempo consecutivos, t_1 e t_2.

É possível escrever a Equação (2.106) matricialmente como

$$\begin{pmatrix}\Delta\omega_x\\ \Delta\omega_y\\ \Delta\omega_z\end{pmatrix} = \begin{bmatrix}0 & \phi_{12} & \phi_{13}\\ \phi_{21} & 0 & \phi_{23}\\ \phi_{31} & \phi_{32} & 0\end{bmatrix}\cdot\begin{pmatrix}r_x\\ r_y\\ r_z\end{pmatrix}\ , \tag{2.107}$$

em que os termos ϕ_{ij} são dados por

$$\begin{cases}
\phi_{12} = -\dfrac{mg\Delta t}{2I_{xx}}\left((c_\phi c_\theta)_{t2} + (c_\phi c_\theta)_{t1}\right)\ ,\\
\phi_{13} = \dfrac{mg\Delta t}{2I_{xx}}\left((s_\phi c_\theta)_{t2} + (s_\phi c_\theta)_{t1}\right)\ ,\\
\phi_{21} = \dfrac{mg\Delta t}{2I_{yy}}\left((c_\phi c_\theta)_{t2} + (c_\phi c_\theta)_{t1}\right)\ ,\\
\phi_{23} = \dfrac{mg\Delta t}{2I_{yy}}\left((s_\theta)_{t2} + (s_\theta)_{t1}\right)\ ,\\
\phi_{31} = -\dfrac{mg\Delta t}{2I_{zz}}\left((s_\phi c_\theta)_{t2} + (s_\phi c_\theta)_{t1}\right)\ ,\\
\phi_{32} = -\dfrac{mg\Delta t}{2I_{zz}}\left((s_\theta)_{t2} + (s_\theta)_{t1}\right)\ .
\end{cases} \tag{2.108}$$

A Equação (2.107) pode ser escrita, ainda, de forma compacta, como

$$\Delta\Omega = \overrightarrow{\phi}\cdot\overrightarrow{r}\ . \tag{2.109}$$

Para resolver essa equação e obter o vetor de posição do centro de massa em relação ao centro de rotação da mesa, $\overrightarrow{r}$, utiliza-se o método dos mínimos quadrados. Nesse método, utiliza-se os valores dos $\Delta\omega_i$ e ϕ_{ij} para diversos instantes de tempo, de modo a ser possível obter um valor mais preciso e confiável desse vetor.

Coletando esses dados para diversos instantes, a Equação (2.107) toma a seguinte forma

$$\begin{pmatrix} (\Delta\omega_x)_{t0} \\ (\Delta\omega_y)_{t0} \\ (\Delta\omega_z)_{t0} \\ (\Delta\omega_x)_{t1} \\ (\Delta\omega_y)_{t1} \\ (\Delta\omega_z)_{t1} \\ \vdots \end{pmatrix} = \begin{bmatrix} 0 & (\phi_{12})_{t0} & (\phi_{13})_{t0} \\ (\phi_{21})_{t0} & 0 & (\phi_{23})_{t0} \\ (\phi_{31})_{t0} & (\phi_{32})_{t0} & 0 \\ 0 & (\phi_{12})_{t1} & (\phi_{13})_{t1} \\ (\phi_{21})_{t1} & 0 & (\phi_{23})_{t1} \\ (\phi_{31})_{t1} & (\phi_{32})_{t1} & 0 \\ \vdots & \vdots & \vdots \end{bmatrix} \cdot \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} . \tag{2.110}$$

Essa forma expandida pode ser escrita da seguinte maneira

$$\Delta\Omega_L = \overrightarrow{\phi_L} \cdot \overrightarrow{r} . \tag{2.111}$$

Por fim, pré-multiplicando-se a pseudo-inversa da matriz $\overrightarrow{\phi_L}$ na Equação (2.111), é possível determinar o vetor-distância entre o CR e o CM da mesa como

$$\overrightarrow{r} = [\phi_L^T \cdot \phi_L]^{-1} \cdot \phi_L^T \cdot \Delta\Omega_L . \tag{2.112}$$

2.9 Ajuste do centro de massa

Uma vez que o vetor de desbalanceamento entre o CR e o CM da mesa é determinado, deve-se determinar quanto cada Unidade de Massa Móvel (UMM) deve se mover para que esse vetor se anule.

A equação geral que define o CM de um sistema de partículas de massas concentradas é dada por

$$\overrightarrow{r_{\mathrm{CM}}} = \frac{1}{M} \sum_{i}^{N} m_{\mathrm{i}} \overrightarrow{r_{\mathrm{i}}} , \tag{2.113}$$

em que M é a massa total do sistema, m_i é a massa da i-ésima partícula que compõe o sistema, $\overrightarrow{r_i}$ é o vetor que descreve a distância entre a i-ésima partícula e a origem do sistema de referência adotado e a soma é feita para todas as N partículas que compõem a massa do sistema. Ou, na forma contínua,

$$\overrightarrow{r_{\mathrm{CM}}} = \frac{1}{M} \int_M \overrightarrow{r} dm , \tag{2.114}$$

em que as variáveis são análogas à equação discreta.

Uma simplificação que se pode adotar é a de considerar que o sistema da mesa é composto por apenas 4 componentes: a mesa desacoplada das UMMs e cada uma das 3 UMMs. Adicionalmente, considera-se que o desbalanceamento do CM do sistema completo é consequência apenas de um desbalanceamento das UMMs e que a mesa sem as UMMs estaria balanceada.

Considerando a mesa balanceada, seu centro de massa será dado por

$$\overrightarrow{r_{\mathrm{mesa}}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} . \tag{2.115}$$

E supõe-se que o centro de massa das UMMs possui apenas uma componente diferente de zero e essa componente é decorrente da forma como as mesmas são fixadas na mesa (uma no eixo x, outra no eixo y e a última no eixo z). Assim, os vetores dos centros de massa das UMMs seriam dados por

$$\overrightarrow{r_x} = \overrightarrow{r_{\mathrm{UMMx}}} = \begin{bmatrix} \Delta x \\ 0 \\ 0 \end{bmatrix}, \tag{2.116}$$

$$\overrightarrow{r_y} = \overrightarrow{r_{\mathrm{UMMy}}} = \begin{bmatrix} 0 \\ \Delta y \\ 0 \end{bmatrix}, \tag{2.117}$$

$$\overrightarrow{r_z} = \overrightarrow{r_{\mathrm{UMMz}}} = \begin{bmatrix} 0 \\ 0 \\ \Delta z \end{bmatrix}. \tag{2.118}$$

Combinando as equações (2.113), (2.115), (2.116), (2.117) e (2.118), tem-se

$$\begin{aligned} \overrightarrow{r_{\mathrm{CM}}} &= \frac{1}{M} \cdot \left(\cancelto{0}{m_{\mathrm{mesa}} \cdot \overrightarrow{r_{\mathrm{mesa}}}} + m_x \cdot \overrightarrow{r_x} + m_y \cdot \overrightarrow{r_y} + m_z \cdot \overrightarrow{r_z} \right) = \\ &= \frac{1}{M} \left(\begin{bmatrix} m_x \Delta x \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ m_y \Delta y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_z \Delta z \end{bmatrix} \right) \Rightarrow \\ \Rightarrow & \boxed{\overrightarrow{r_{\mathrm{CM}}} = \frac{1}{M} \cdot \begin{bmatrix} m_x \Delta x \\ m_y \Delta y \\ m_z \Delta z \end{bmatrix}}, \end{aligned} \tag{2.119}$$

em que m_x, m_y e m_z são as massas das UMMs dos eixos x, y e z, respectivamente.

Reescrevendo a Equação (2.119), tem-se

$$\overrightarrow{r_{\mathrm{CM}}} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \frac{1}{M} \cdot \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & m_z \end{bmatrix} \cdot \begin{bmatrix} r_{\mathrm{UMMx}} \\ r_{\mathrm{UMMy}} \\ r_{\mathrm{UMMz}} \end{bmatrix}, \tag{2.120}$$

em que nota-se a substituição de Δx, Δy e Δz pelos escalares r_{UMMx}, r_{UMMy} e r_{UMMz}, respectivamente. O subscrito UMMx, por exemplo, indica que o termo que o contém se refere à Unidade de Massa Móvel do eixo x.

Isolando o vetor de deslocamento das massas, tem-se

$$\begin{aligned}\begin{bmatrix} r_{\mathrm{UMMx}} \\ r_{\mathrm{UMMy}} \\ r_{\mathrm{UMMz}} \end{bmatrix} &= \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & m_z \end{bmatrix}^{-1} \cdot M \cdot \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \\ &= M \cdot \begin{bmatrix} \frac{1}{m_x} & 0 & 0 \\ 0 & \frac{1}{m_y} & 0 \\ 0 & 0 & \frac{1}{m_z} \end{bmatrix} \cdot \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \\ &= M \cdot \begin{bmatrix} \frac{r_x}{m_x} \\ \frac{r_y}{m_y} \\ \frac{r_z}{m_z} \end{bmatrix} .\end{aligned} \tag{2.121}$$

Deseja-se obter a variação do centro de massa que leva o vetor $\overrightarrow{r_{\mathrm{CM}}}$ a zero, a saber,

$$\overrightarrow{r_{\mathrm{UMM}}} + \Delta\overrightarrow{r_{\mathrm{UMM}}} = \overrightarrow{0} \Rightarrow \Delta\overrightarrow{r_{\mathrm{UMM}}} = -\overrightarrow{r_{\mathrm{UMM}}} = -M \cdot \begin{bmatrix} \frac{r_x}{m_x} \\ \frac{r_y}{m_y} \\ \frac{r_z}{m_z} \end{bmatrix} . \tag{2.122}$$

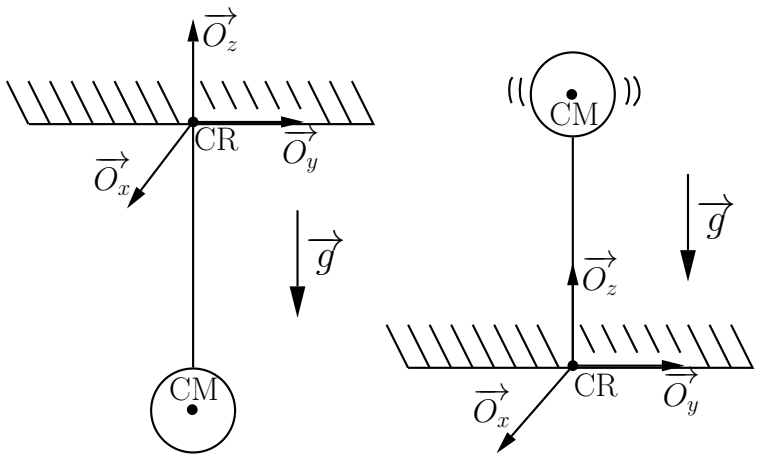
Assumindo que todas as UMMs possuam a mesma massa, ou seja, $m = m_x = m_y = m_z$, a Equação (2.122) toma a forma

$$\Delta\overrightarrow{r_{\mathrm{UMM}}} = -\frac{M}{m} \cdot \overrightarrow{r_{\mathrm{CM}}} . \tag{2.123}$$

2.10 Considerações sobre estabilidade

Por último, algumas considerações e discussões relacionadas ao funcionamento adequado do algoritmo de balanceamento e à estabilidade da mesa. Retornando à analogia feita na Subseção 2.1.3, considera-se que a mesa pode ser modelada de forma simplificada como um pêndulo.

Tipicamente, quando deixado em repouso, o pêndulo irá estabilizar em uma posição de equilíbrio em que seu centro de massa possuirá apenas a componente vertical diferente de zero, para o sistema de referência ilustrado na Figura 2.10.



(a) Posição de equilíbrio estável do pêndulo.

(b) Posição de equilíbrio instável do pêndulo.

Figura 2.10: Posições de equilíbrio do pêndulo.

Supondo que o algoritmo de balanceamento seja capaz de anular, além das componentes nos eixos x e y do vetor de desbalanceamento, a componente no eixo z, a massa do pêndulo ficaria totalmente concentrada no centro de rotação. Nesse caso, o pêndulo não teria uma posição de equilíbrio definida e poderia assumir qualquer orientação sem tender a uma posição diferente. Nessa condição, qualquer torque exercido na mesa faria com que ela se movimentasse livremente até atingir um de seus limites físicos de movimentação, dados pela forma como a plataforma está montada. Essa situação, no entanto, é impraticável, devido às limitações físicas da atuação do sistema.

Supõe-se agora que, após uma iteração qualquer do algoritmo de balanceamento, o CM passe a apresentar uma componente positiva no eixo z, mantendo as componentes dos eixos x e y nulas. Tal situação seria equivalente à situação de instabilidade do pêndulo ilustrada na Figura 2.10b. Qualquer pequeno distúrbio no pêndulo seria capaz de retirá-lo dessa posição de equilíbrio. Em outras palavras, nessa situação a mesa cairia para algum lado, sendo segurada apenas por seu suporte físico.

Nota-se, portanto, que deve-se atentar para a tolerância de balanceamento do eixo z: a componente em z do CM nunca assumirá, na realidade, o valor nulo, mas, caso atinja valores positivos acidentalmente, irá causar uma situação de desequilíbrio na plataforma. Logo, deve-se adotar uma tolerância que faça com que a componente z do CM esteja *sempre* abaixo do centro de rotação do sistema para que a mesa esteja sempre em condição estável, como ilustrado na Figura 2.10a.

Capítulo 3

Projeto

Este capítulo destina-se à descrição de todo o projeto, incluindo sua estrutura física, o hardware *utilizado e o* software *desenvolvido.*

3.1 Estrutura física

Durante o curso deste trabalho foi projetado e construído um sistema experimental para simulação da dinâmica de atitude de pequenos satélites. O sistema pode ser caracterizado por diversos subsistemas que serão discutidos neste capítulo. Para isso, dar-se-á atenção especial aos componentes utilizados e às funções desempenhadas por eles no simulador proposto.

O sistema de ensaio proposto permite testar o funcionamento do controle de atitude de microssatélites. Para este fim, é essencial que este seja capaz de simular um ambiente no qual as condições sejam as mais próximas do ambiente de operação de um satélite real. Por isso, é necessário reduzir todas as forças e binários que são ausentes, ou muito reduzidos, no ambiente espacial, mas que estão presentes no ambiente terrestre. Destes, os mais significantes para a aplicação proposta, são o atrito e o torque gravitacional.

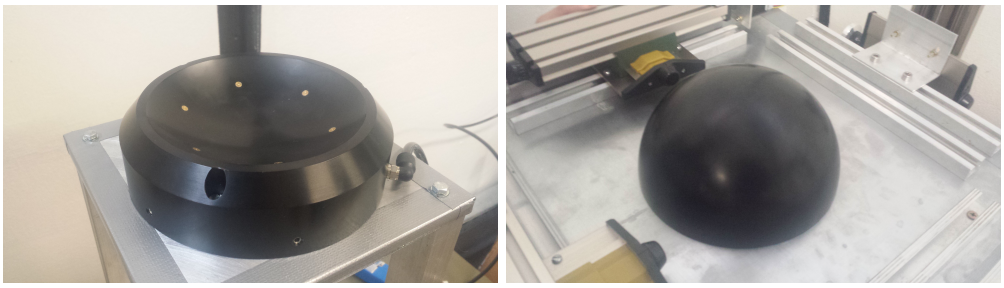
Como será discutido em detalhes posteriormente neste trabalho, a mesa necessita ser apoiada em um suporte físico. O principal problema de se utilizar um suporte físico é a existência de atritos que interferem com a atuação do microssatélite. Com o intuito de solucionar esse problema, propõe-se que seja utilizado um sistema que produza um colchão de ar entre a mesa e seu suporte, dessa forma reduzindo o atrito drasticamente. Isso pode ser ilustrado ao compararmos o coeficiente de atrito entre duas peças de alumínio, que é de cerca de 0,3 [19], com o do alumínio com um colchão de ar, que é de 0,01 [20]. No projeto foi utilizado um dispositivo capaz de gerar esse colchão de ar e um sistema pneumático para abastecê-lo, sendo que ambos serão discutidos em detalhes posteriormente. O dispositivo em questão é um *rolamento a ar* e foi utilizado na configuração topo de mesa, a qual foi apresentada no Capítulo 1.

Além disso, é necessário construir um sistema que seja capaz de balancear a mesa, para que o torque gravitacional seja diminuído drasticamente, a fim de simular de forma mais acurada o ambiente de operação de um pequeno satélite.

3.1.1 Estrutura mecânica

A estrutura mecânica da mesa pode ser dividida em duas partes principais, a parte superior, incluindo a mesa e os componentes que estarão acoplados a ela, e a parte inferior, composta pelo suporte da mesa. Entre as duas, está acoplado um rolamento a ar, cuja função é o de criar um colchão de ar entre essas duas partes para diminuir o atrito durante a movimentação da mesa, como discutido anteriormente nos requisitos, além de permitir movimentos rotacionais tridimensionais.

Utilizou-se um rolamento de ar esférico, fabricado pela Nelson Air Corp., cujas especificações são personalizadas e, por isso, não constam em domínio público. Este modelo é constituído por uma concavidade semiesférica com seis orifícios igualmente espaçados com relação a seu centro, por onde o ar comprimido é lançado, e por uma semiesfera de 175mm de diâmetro que flutua sobre essa concavidade, como mostrado nas figuras 3.1a e 3.1b, respectivamente. Essa semiesfera possibilita rotação de 360° em torno do eixo de *guinada* da mesa. Já nos eixos de arfagem e rolagem, esse conjunto possibilita uma rotação de 45°. A pressão de operação nominal do conjunto é de 80 *psi*, não podendo ultrapassar os 90 *psi*. De acordo com a especificação do *datasheet*, o rolamento a ar suporta 100kg ao ser alimentado com ar à pressão de 60 *psi* e aproximadamente 165.6kg a uma pressão de 80 *psi*. O consumo de ar do conjunto é menor que 1.0 $SCFM$[1]. Esse conjunto requer que o ar pressurizado utilizado para alimentar o sistema seja livre de óleo, seco (ponto de orvalho de $-15°C$) e limpo - com filtragem de partículas maiores que $10\mu m$ ou melhor. O conjunto é feito de alumínio AL6061 submetido ao processo de anodização dura [18].



(a) Base do rolamento a ar. (b) Semiesfera do rolamento a ar.

Figura 3.1: Componentes do rolamento a ar.

A parte inferior do sistema - o pedestal - é formada por duas bases ligadas por meio de 4 hastes maciças de alumínio de seção transversal quadrada. As bases são feitas de alumínio do tipo "*honeycomb*". O comprimento escolhido para as hastes permite que a mesa seja montada a uma altura confortável para a manipulação dos equipamentos e realização das simulações. A base do rolamento a ar - a concavidade semiesférica - é fixada na base superior. A Figura 3.2 ilustra esse conjunto.

[1] Do inglês *Standard Cubic Feet per Minute* ou "pé cúbico padrão por minuto". 1.0 $SCFM$ = 1.0 ft^3/min = 0.000472 m^3/s.

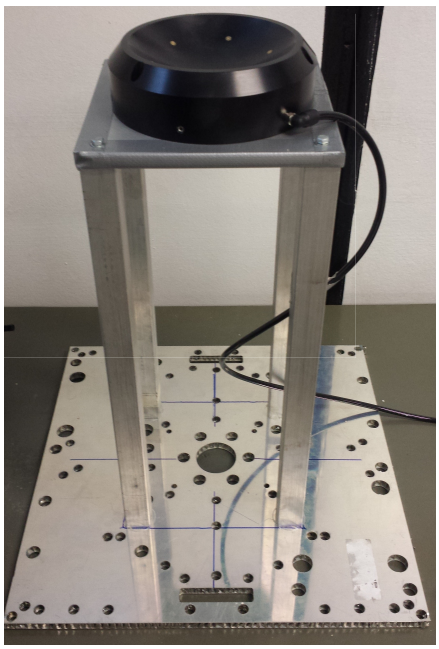


Figura 3.2: Parte inferior do sistema.

A parte superior do sistema é formada por uma placa de alumínio do tipo "*honeycomb*". O lado superior dessa placa acomoda os dispositivos de aquisição de dados necessários para a realização da simulação, que serão discutidos em detalhes na Seção 3.2. A Figura 3.3 ilustra a parte superior do sistema sendo segurada e mostrando o hemisfério que compõe o rolamento a ar.

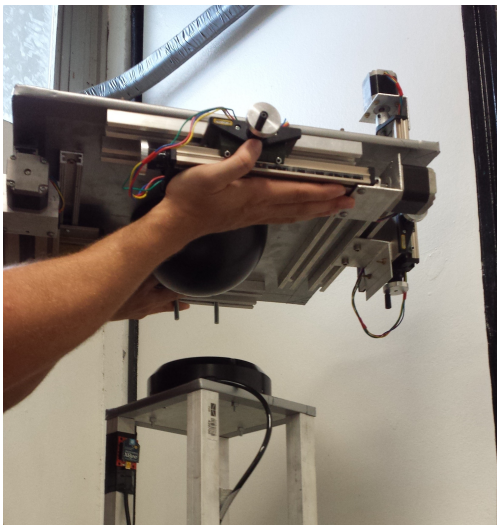


Figura 3.3: Parte superior do sistema.

Como foi explicitado nos requisitos, é necessário realizar o balanceamento da mesa com o intuito de diminuir a influência do torque gravitacional. Para isso, utiliza-se um conjunto de três UMMs[2] juntamente com um sistema de controle para modificar a localização do centro de massa da mesa.

As UMMs devem ser capazes de se movimentar de forma precisa. Para isso, utilizou-se a mesa

[2]UMM - Unidade de Massa Móvel.

coordenada de modelo KT70 fabricada pela Proxxon. Essa mesa coordenada possui dimensões de 200mmx70mm e permite movimentações de 134 mm na sua direção longitudinal (eixo x') e 46 mm ao longo da direção transversal (eixo y') (ver Figura 3.4). O dispositivo possui manivelas que permitem realizar deslocamentos relativos aos eixos citados acima, sendo que uma volta completa de uma das manivelas corresponde a um deslocamento de 1 mm no eixo correspondente. Adicionalmente, a menor divisão de escala de cada manivela corresponde a um deslocamento de 0.05 mm. A cada eixo do sistema de referência fixo à mesa corresponde uma UMM [22].

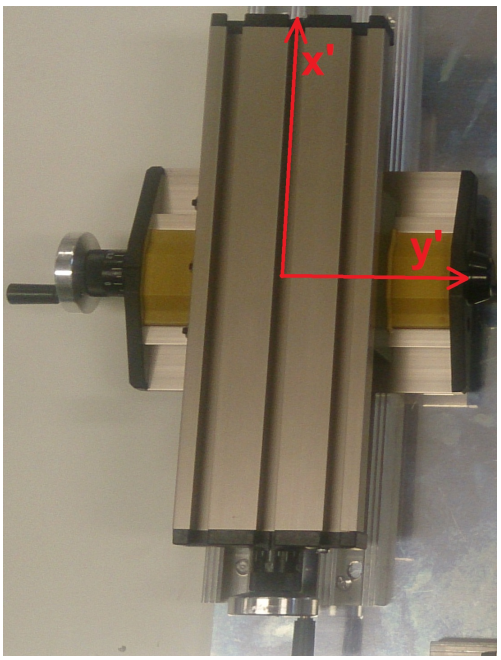


Figura 3.4: Mesa coordenada KT70.

Para acoplar os eixos dos motores ao parafuso sem-fim que causa o deslocamento no eixo longitudinal das UMMs, fez-se uso de um pequeno dispositivo cilíndrico de alumínio. Isso é ilustrado na Figura 3.5.

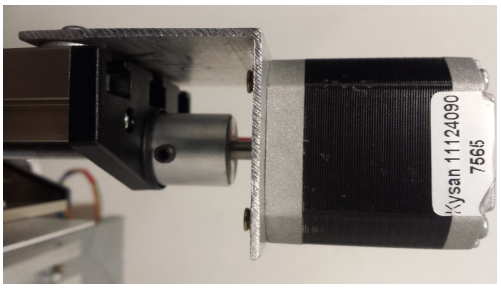
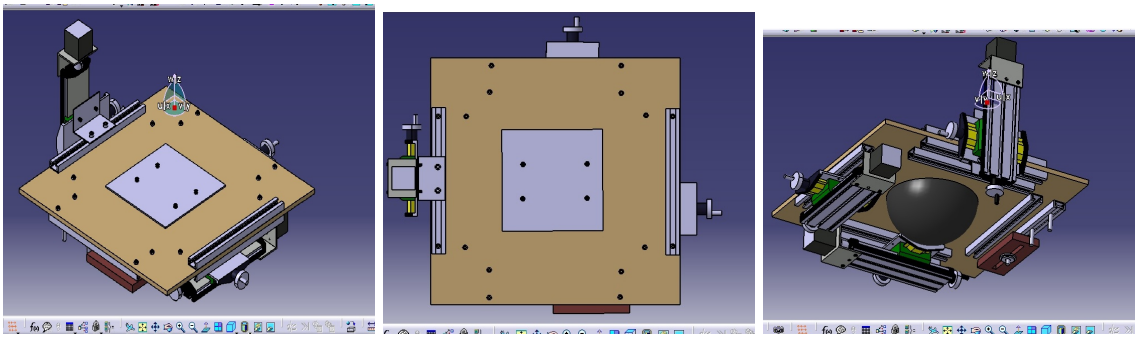


Figura 3.5: Acoplamento do motor à UMM.

A parte superior do sistema com os componentes citados anteriormente montados é mostrada nas figuras 3.6a, 3.6b e 3.6c, desenvolvidas no programa CATIA.

Figura 3.6: Desenho da mesa com rolamento a ar feita no programa CATIA.



(a) Vista isométrica. (b) Vista de cima. (c) Vista de baixo.

3.1.2 Estrutura pneumática

Para alimentar o rolamento a ar é necessário um sistema que permita que o colchão de ar seja gerado de forma ininterrupta. Para isso, utilizou-se um compressor de ar com as seguintes características

Fabricante	PUSKA
Modelo	COMBA 2100 R II
Pressão máxima de operação	10 bar
Vazão máxima	$0.3m^3/min$
Potência	1.5kW
Tensão de alimentação	230V
Frequência da rede de alimentação	50Hz

Tabela 3.1: Especificações do compressor.

Esse modelo foi escolhido porque atinge os requisitos necessários para o funcionamento do sistema de compressão a ar. A Figura 3.7a ilustra o modelo presente no laboratório.

(a) Compressor. (b) Filtro de ar

Figura 3.7: Componentes do sistema pneumático.

Cabe notar que a alimentação elétrica do compressor, como consta em suas especificações, difere da alimentação disponível no laboratório, que é de 220V com uma frequência de 60Hz. Essa diferença, no entanto, não causa maiores complicações com relação a seu funcionamento.

Outro componente do sistema pneumático é o filtro. Ele remete ao requisito de pureza do ar para funcionamento adequado do rolamento a ar. Na aplicação proposta, essas impurezas são acentuadas devido ao uso do compressor de ar, o qual, a uma pressão de 10 bar, aumenta a concentração de partículas de impureza em 11 vezes [24].

Para corrigir essas impurezas, foi utilizado um conjunto de filtros entre a tubulação do compressor e do rolamento a ar. Primeiramente, o ar sai do compressor e passa pelo filtro 4P-061-M04-DC, que possui a capacidade de filtrar partículas de até 0.01 micron [21]. Logo após passar pelo filtro, o ar pressurizado atinge a base do rolamento a ar. A Figura 3.7b mostra o filtro utilizado e a Figura 3.8 mostra um esquemático do sistema pneumático utilizado.

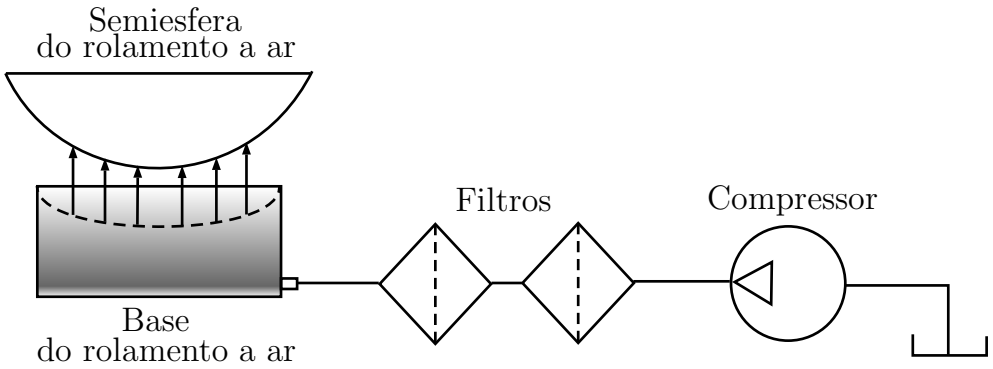


Figura 3.8: Esquemático do sistema pneumático.

3.2 Hardware

3.2.1 Unidade de Medição Inercial

Como sensor principal do sistema, utilizou-se uma unidade de medição inercial (UMI)[3] que possui a função de medir a orientação da mesa em relação a um eixo inercial através dos ângulos de Euler (arfagem, rolagem e guinada). Essa classe de sensores é comumente utilizada em sistemas de navegação inercial, nos quais a orientação de um objeto, como um satélite ou um navio, é calculada continuamente sem a necessidade de referência externa.

No simulador descrito neste trabalho, utilizou-se a UMI Adafruit 9DOF[4] da fornecedora Adafruit, o qual possui esse nome pois permite a realização de nove medidas distintas, que são: aceleração; orientação magnética; e velocidade angular, sendo que todas são realizadas nos 3 eixos da UMI, gerando um total de nove medidas distintas.

A UMI utilizada é formada pela fusão de dois dispositivos eletrônicos principais em uma mesma placa: o LSM303DLHC e o L3GD20.

O LSM303DLHC, combina um acelerômetro de três eixos (3D), como ilustrado na Figura 3.9, e um magnetômetro de três eixos (3D), como ilustrado na Figura 3.10, e pode ser utilizado em uma vasta gama de aplicações, desde detecção de posição de um objeto até na criação de dispositivos de realidade virtual. Além disso, possui capacidade de detecção de variações magnéticas de -1.3 até 1.3 Gauss e de acelerações entre -2g e +2g.

Vale mencionar que os limites de detecção citados podem ser ajustados através da modificação dos respectivos fundos de escala, todavia, para este trabalho, utilizou-se os fundos de escala padrão, pois eles estão nas faixas de valores esperados para a aplicação em questão.

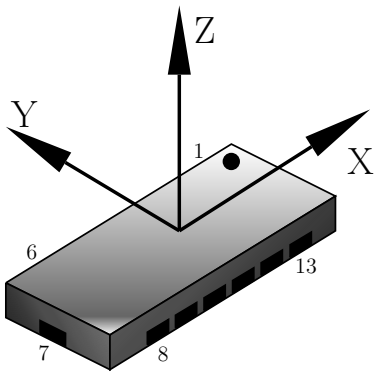


Figura 3.9: Acelerômetro 3D

[3] Do inglês *Inertial Measurement Unity* ou "Unidade de Medição Inercial".
[4] Do inglês *nine degrees of freedom* ou "Nove Graus de Liberdade".

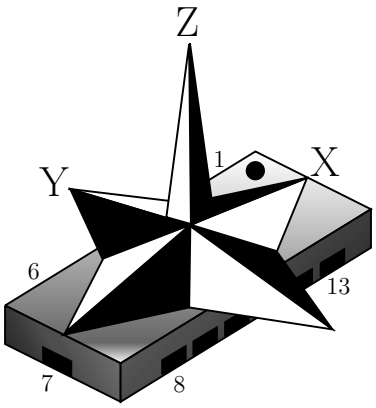


Figura 3.10: Magnetômetro 3D

Já o L3GD20 é formado por um único sensor, um giroscópio 3D, que mede a velocidade angular nos 3 eixos da UMI, como mostrado na Figura 3.11. Este dispositivo pode ser utilizado em diversas aplicações práticas, desde sistemas de navegação GPS até em controle de movimento em dispositivos de interface homem-máquina. Ele também possui fundos de escala ajustáveis [29], mas, para este trabalho, utilizou-se o fundo de escala padrão de 250 graus/segundo. Como o movimento esperado para a mesa é suave, considerou-se que esse fundo de escala é adequado para a aplicação.

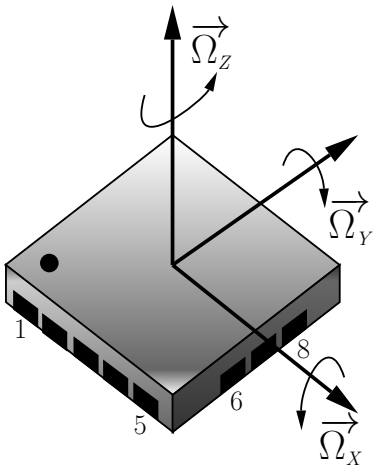


Figura 3.11: Giroscópio 3D

A combinação desses componentes, a qual chamamos de UMI, gera informações essenciais para a aplicação proposta, contudo não gera automaticamente os valores de atitude requeridos. Para obter esses valores, utilizou-se como base um algoritmo desenvolvido pela Adafruit [27] que simula um AHRS[5], que é um sistema de referência de atitude e rumo, muito utilizado em controle de satélites e aeronaves.

Para o cálculo dos ângulo de Euler, o algoritmo da Adafruit utiliza a biblioteca Adafruit_9DOF, desenvolvida em C++ no ambiente do Arduino. Nela, o cálculo dos ângulos é feito utilizando o método descrito na Seção 2.4.

[5]Do inglês *Attitude and Heading Reference System* ou "Sistema de Referência de Atitude e Rumo".

Por fim, para obtenção dessas orientações, deve-se conectar os sensores diretamente ao Arduino. A Figura 3.12 indica as ligações feitas. Nota-se que apenas 4 pinos são utilizados, dois de comunicação (SCL e SDA) e dois para a alimentação. Implementou-se a comunicação através do protocolo I2C[6]. Nele, o pino SDA[7] realiza a transferência de dados, enquanto o pino SCL[8] realiza a temporização entre o Arduino e a UMI.

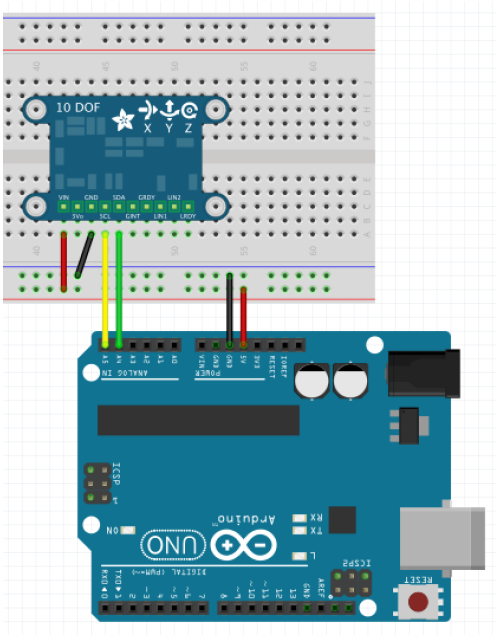


Figura 3.12: Ligação Arduino-UMI

3.2.2 Comunicação via XBee

XBee é o nome dado a uma família de componentes produzidos pela empresa Digi. Esses são dispositivos de rádio que podem ser utilizados em conjunto com a plataforma Arduino para prover comunicação sem fio utilizando o protocolo de comunicação ZigBee.

Eles foram criados com o intuito de suprir a demanda de dispositivos com baixo custo, consumo energético reduzido e com uma distância de transmissão razoável. Tipicamente são utilizados em sistemas com baixa taxa de transmissão de dados que requerem baterias que resistam por muito tempo. Sistemas nos quais ocorre transmissão intermitente de dados provindos de sensores são um exemplo comum desse tipo de aplicação [32].

Uma fonte de erro para a estimação do tensor de inércia da mesa com rolamento a ar seria a existência de cabos para a conexão ao computador remoto responsável pelo processamento dos dados de controle. Por isso, utilizaram-se dois módulos XBee: um para transmitir os dados da mesa até o computador e outro, conectado a esse computador, para receber esses dados.

Dessa forma, os sensores acoplados à mesa passam suas informações ao Arduino da mesa que,

[6] Do inglês *Inter-Integrated Circuit*.
[7] Do inglês *Serial Data*.
[8] Do inglês *Serial Clock*.

por sua vez, as envia para o módulo XBee. Esse módulo recebe as informações e as transmite em forma de pacotes para outro XBee ligado ao computador. Isso é ilustrado na Figura 3.13.

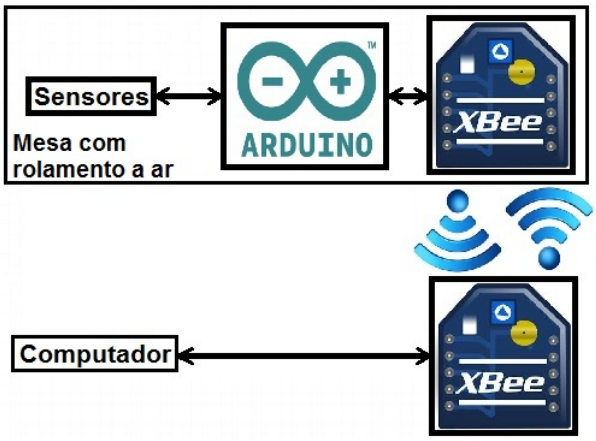


Figura 3.13: Tráfego de dados no sistema.

Utilizou-se, neste trabalho, o modelo xb24-aci-001 cujas características podem ser vistas na Tabela 3.2.

Característica	**Xbee**
Alcance máximo em ambientes fechados	30 m
Alcance máximo em ambientes abertos	90 m
Taxa de dados do pino RF	250.000 bps
Taxa de dados com interface serial	1200bps - 250kbps
Tensão de alimentação	2.8-3.4V

Tabela 3.2: Especificações do módulo Xbee utilizado no projeto.

Para utilizá-lo é necessário realizar uma configuração prévia. A qual será discutida posteriormente.

3.2.3 Motor e drivers

Para a movimentação das UMMs, foram utilizados motores de passo bipolares da empresa Kysan Electronics, modelo "SKU: 1124090" e "Mfg: 42BYGH4803-DC". As especificações referentes a esse modelo constam na Tabela 3.3.

Especificações elétricas	
Voltagem nominal	4.2V
Corrente por fase	1.5A
Resistência por fase	$2.8 \pm 10\%\Omega$
Indutância por fase	$4.8 \pm 20\% mH$
Número de fases	2
Especificações mecânicas	
Ângulo por passo	1.8°
Torque Estático[9]	$5.5 kg \cdot cm$
Outras especificações	
Classe de isolamento	B
Resistência do isolamento	$100 M\Omega$ (500 V DC)
Peso	375g
Tipo de conector	SKU 1160078
Tipo de pino	SKU 1160079

Tabela 3.3: Especificações do motor de passo utilizado no projeto.

Para controlar os motores, utilizou-se um driver de motor de passo baseado no circuito integrado (CI) A4988, fabricado pela Allegro MicroSystems, Inc. Existem vários drivers baseados nesse CI. O driver utilizado é conhecido, também, como StepStick. As especificações desse driver constam na Tabela 3.4 [25].

Driver de motor de passo A4988	
Tensão mínima de operação	8 V
Tensão máxima de operação	35 V
Corrente máxima por fase	1A
Tensão lógica mínima	3V
Tensão lógica máxima	5.5 V
Resoluções de micro-passo disponíveis	Passo completo, 1/2, 1/4, 1/8, e 1/16

Tabela 3.4: Especificações do driver de motor de passo A4988.

Observa-se que a faixa de tensão de operação do driver escolhido - 8V a 35V de alimentação para o motor - não abrange a tensão nominal do motor de passo utilizado. Em um primeiro momento, o driver utilizado pode parecer inadequado. No entanto, a tensão nominal presente nas especificações do motor - 4.2V - indica apenas que, se alimentado com essa tensão contínua, o motor irá apresentar a corrente nominal de 1.5A em suas fases. Utilizando a lei de Ohm, é possível verificar que esses dados são coerentes com a indicação da resistência por fase do motor

$$V = R \cdot i \Rightarrow 4.2V = 2.8\Omega \cdot 1.5A \ . \tag{3.1}$$

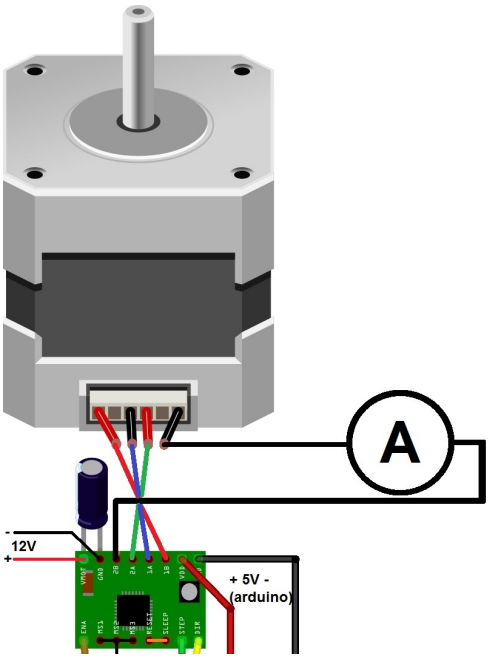
Na verdade, motores de passo são postos em operação a tensões tipicamente maiores para que

possam atingir maiores velocidades. Há de se atentar que o uso adequado de motores de passo requer apenas que a corrente nominal por fase não seja ultrapassada.

Observa-se na Tabela 3.4 que a corrente máxima que o driver pode fornecer é de $1.0A$. Esse valor é inferior à corrente nominal do motor, que é de $1.5A$ como consta na Tabela 3.3. Seria possível operar os motores com o driver em seu limite de corrente, porém sua operação nessa condição requereria um sistema de refrigeração ativo em seu circuito.

Dessa forma, nota-se que é necessário limitar a corrente fornecida às fases do motor de passo. Existem duas formas de se fazer isso. Uma delas é, enquanto se mede a corrente que se passa por uma das bobinas do motor sem atuá-lo, regular o potenciômetro presente no circuito do driver. As figuras 3.14a e 3.14b ilustram o procedimento.

(a) Placa do circuito do driver A4988. Circulado em azul está o potenciômetro.



(b) Conexão feita para medição de corrente em uma das bobinas do motor de passo.

Figura 3.14: Procedimento para limitar a corrente fornecida às fases do motor de passo.

Outra forma é a de, ajustando o potenciômetro mencionado anteriormente, verificar a tensão V_{REF} do circuito do driver tendo em mente a seguinte fórmula [26]

$$I_{lim} = \frac{V_{REF}}{8 \times R_S} \ , \tag{3.2}$$

em que I_{lim} é a corrente máxima que o driver irá fornecer e R_S é o valor de resistência do resistor de sensor presente no circuito do driver. Os drivers baseados no CI A4988 apresentam, normalmente, os valores 0.05Ω, 0.1Ω e 0.2Ω para as resistências R_S de seus circuitos. No caso do driver escolhido, olhando atentamente para o circuito é possível identificar um resistor do tipo SMD[10]

[10] Do inglês, "*Surface-Mount Device*". São componentes de circuito que são soldados diretamente na superfície de circuitos impressos.

com a identificação "R200"[11], o que indica que o valor de R_S do driver utilizado é de 0.2Ω. A fórmula que relaciona a corrente-limite e a tensão V_{REF} é dada por

$$I_{lim} = \frac{V_{REF}}{1.6} \ . \tag{3.3}$$

Essa tensão pode ser verificada medindo-se a tensão entre o pino GND do circuito do driver ou a alimentação lógica do circuito e a carcaça do potenciômetro, circulada na Figura 3.14a.

Neste trabalho, utiliza-se o driver do motor de passo no modo de passo completo. Nesse modo, a corrente fornecida às bobinas é limitada a 70% do limite de corrente ajustado ao driver em decorrência de um controle interno do circuito do driver. Dessa forma, almejando atingir o limite de corrente de $1.0A$, tem-se

$$1.0A = \frac{V_{REF}}{1.6} \ \Rightarrow \boxed{V_{REF} = 1.6V} \ , \tag{3.4}$$

mas, devido à limitação de *hardware* citada, a corrente observada nos enrolamentos dos motores de passo será de 70% do valor previsto em cálculo, ou seja, 0.7A. A Figura 3.15 mostra o circuito do driver StepStick.

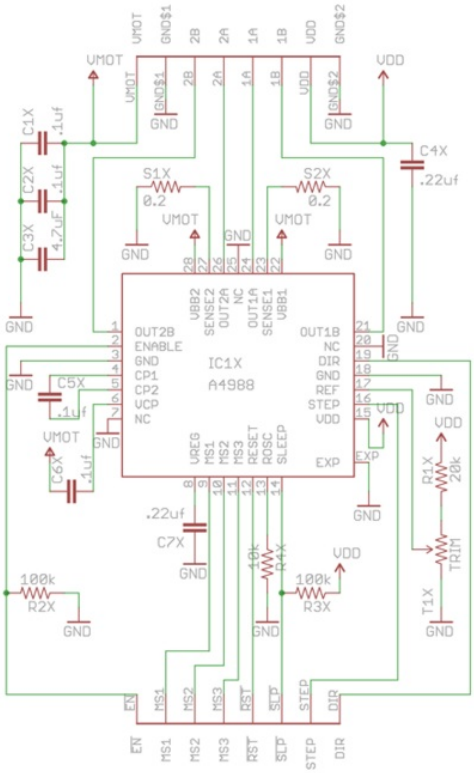


Figura 3.15: Diagrama do circuito do driver A4988 StepStick.

[11] Para resistores do tipo SMD utiliza-se usualmente um código de 3 dígitos para determinar a resistência, no qual os dois primeiros dígitos indicam os algarismos mais significativos e o terceiro é um multiplicador. Além disso, para resistências menores que 10Ω, utiliza-se um caractere R para indicar a posição do separador decimal. O código 1R0, por exemplo, indicaria um resistor de 1Ω.

Nota-se que o trimpot atinge uma resistência máxima de $10k\Omega$. Como V_{REF} é obtido de uma divisão de tensão entre o trimpot e um resistor de $20k\Omega$ submetidos à tensão VDD (aproximadamente 5V), então o valor máximo de V_{REF} é limitado a um valor próximo de 1.66V. Conclui-se, portanto, que a corrente máxima que se pode obter é de algo em torno de 0.7A com o driver disponível. Como os motores de passo deste projeto não serão submetidos a cargas elevadas, optou-se por regular V_{REF} para a metade do valor calculado: 0.8V.

A decisão de reduzir a corrente dos motores impacta diretamente em dois fatores cruciais: o limite de corrente escolhido reduz drasticamente o calor dissipado no circuito do driver, prevenindo o mesmo de eventuais danos; por se tratar de uma aplicação de eletrônica embarcada, qualquer consumo desnecessário de energia deve ser eliminado.

A Figura 3.16 explicita as conexões feitas entre o Arduino, o driver do motor 1, o motor 1 e as baterias presentes no sistema.

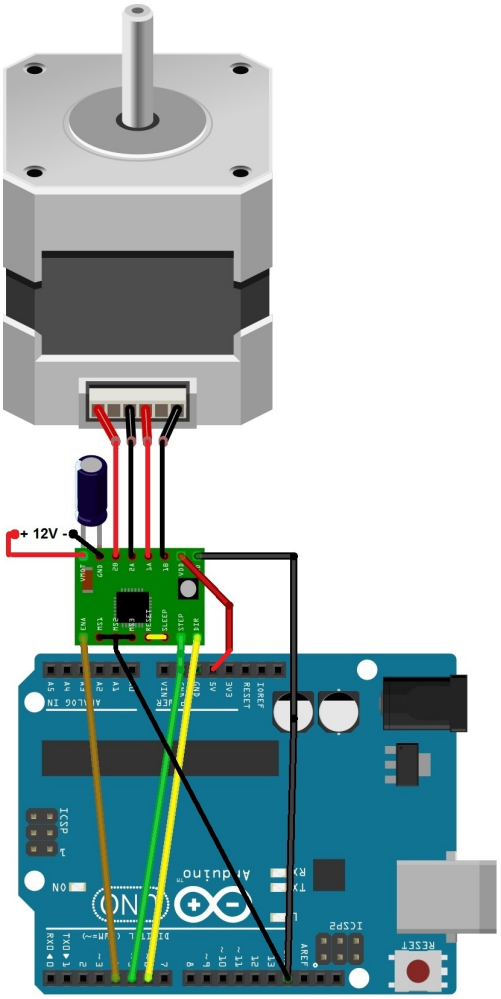


Figura 3.16: Diagrama de conexão elétrica entre o Arduino, o motor e seu driver.

Vale ressaltar que utilizam-se 3 pinos principais do driver: *Direction Input* (DIR); *Step Input* (STEP); e $\overline{ENABLE}$ ($\overline{EN}$). O primeiro é utilizado para determinar a direção de rotação do motor. Já o segundo é utilizado para controlar os passos do motor da seguinte maneira: para cada

pulso de tensão aplicado a esse pino o motor gira um passo completo. Por fim, o pino $\overline{EN}$ habilita as saídas do driver quando em nível lógico baixo e as desabilita quando em nível lógico alto.

Para os motores 2 e 3, utilizou-se conexão análoga à mostrada na Figura 3.16. A única diferença se dá nas conexões entre o Arduino e o driver do motores. No caso do motor 2, os pinos $\overline{EN}$, STEP e DIR são conectados aos pinos de sinal digital 7, 8 e 9 do Arduino, respectivamente. No caso do motor 3, os pinos $\overline{EN}$, STEP e DIR são conectados aos pinos de sinal digital 10, 11 e 12 do Arduino, respectivamente.

3.3 Software

Esta seção destina-se ao detalhamento da programação utilizada no projeto. Nela dar-se-á uma visão geral do funcionamento e aplicação das diversas funções criadas ao longo deste trabalho.

Implementou-se o projeto em dois ambientes de programação principais: o MATLAB e o ADI (Ambiente de desenvolvimento integrado) do Arduino.

Utilizou-se o primeiro para implementação da programação presente no computador externo à mesa, que, para este trabalho, será chamado de UCP (Unidade central de processamento), enquanto o segundo foi utilizado para implementar a programação embarcada a um Arduino fixo à mesa.

Os dois sistemas comunicam-se através de dois módulos XBee, cujo funcionamento será descrito a seguir.

3.3.1 Comunicação XBee

Como foi discutido na Seção 3.2.2, utilizaram-se dois módulos XBee para comunicação sem fio. Nesta subseção explicar-se-á sua configuração e a programação utilizada nesses dispositivos.

A configuração é realizada conectando-os a um computador e utilizando o software gratuito X-CTU da empresa Digi, desenvolvedora do XBee. A figura 3.17 ilustra esse processo.

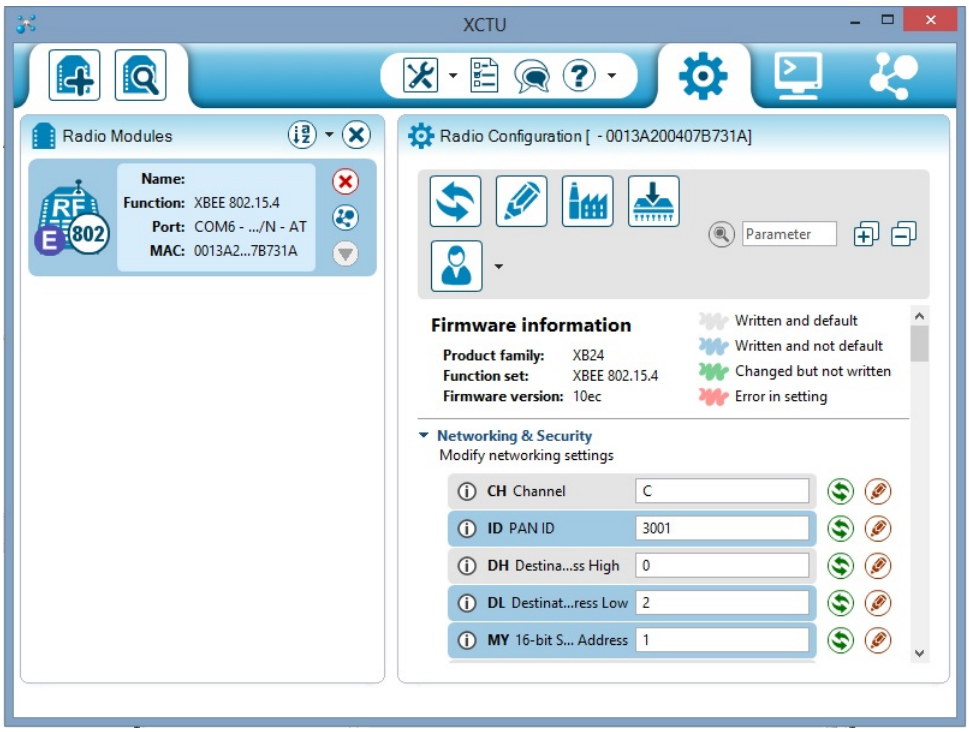


Figura 3.17: Tela do software XCTU.

A conexão do XBee utilizado na mesa é feita diretamente no Arduino, sem intermédio de qualquer circuito. A conexão feita é mostrada a seguir.

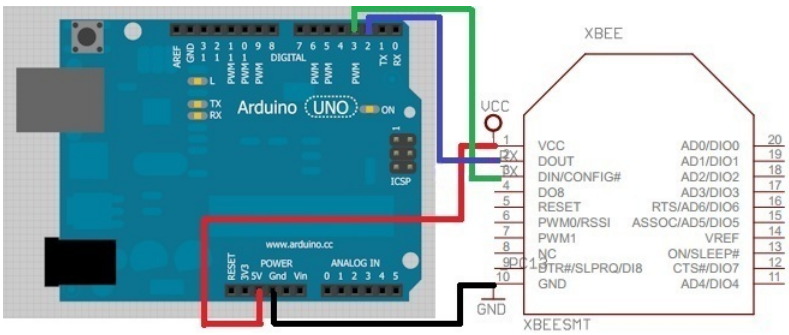


Figura 3.18: Conexões feitas entre o XBee e o Arduino.

Já no caso do XBee ligado ao computador, essa conexão é feita por meio de uma porta USB juntamente com um circuito adaptador chamado XBee Explorer. A figura 3.19 mostra a versão do XBee Explorer utilizado.

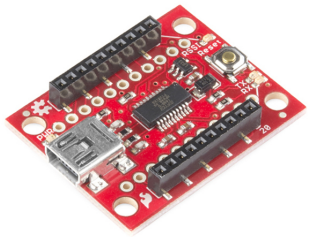


Figura 3.19: Adaptador Shield XBee Explorer.

Cabe ressaltar que, para que o módulo seja prontamente reconhecido pela porta USB, é necessário instalar o driver do chip FTDI presente no XBee Explorer [17].

Para realizar a comunicação entre os módulos XBee, é necessário indicar na configuração de cada um deles o endereço do dispositivo que receberá as mensagens enviadas. Como estão sendo utilizados módulos XBee série 1, não é necessário configurar uma hierarquia entre os módulos do tipo Mestre/Escravo. Ambos podem se comunicar livremente. Para realizar a comunicação, é necessário, também, criar uma identificação para a rede.

Os parâmetros que descrevem essas configurações são **IP PAN ID** (identificação da rede), **DH Destination Address High** (porção de bits mais significativos do endereço de envio de pacotes) e **DH Destination Address Low** (porção de bits menos significativos do endereço de envio de pacotes).

O endereço de cada módulo XBee pode ser obtido através do software X-CTU ou observando a etiqueta presente no verso do aparelho. A Tabela 3.5 indica os endereços dos XBees utilizados.

Localização do XBee	Endereço (High|Low)	IP PAN ID
Mesa *Air Bearing*	0013A200|407B731A	FF
Computador	0013A200|407B7313	FF

Tabela 3.5: Detalhes de configuração dos XBees.

O procedimento de teste foi basicamente o seguinte: carregou-se um programa no Arduino para que o mesmo se comunicasse com o XBee ligado a ele por meio do Monitor Serial (recurso disponível na ADI do Arduino). Através do software XCTU, fez-se com que o XBee ligado ao computador via USB enviasse mensagens para o módulo XBee ligado ao Arduino. Dessa forma, o teste procedeu de forma bem-sucedida.

Na imagem abaixo, as mensagens enviadas são mostradas no software XCTU, enquanto que as mensagens recebidas são mostradas pelo Monitor Serial.

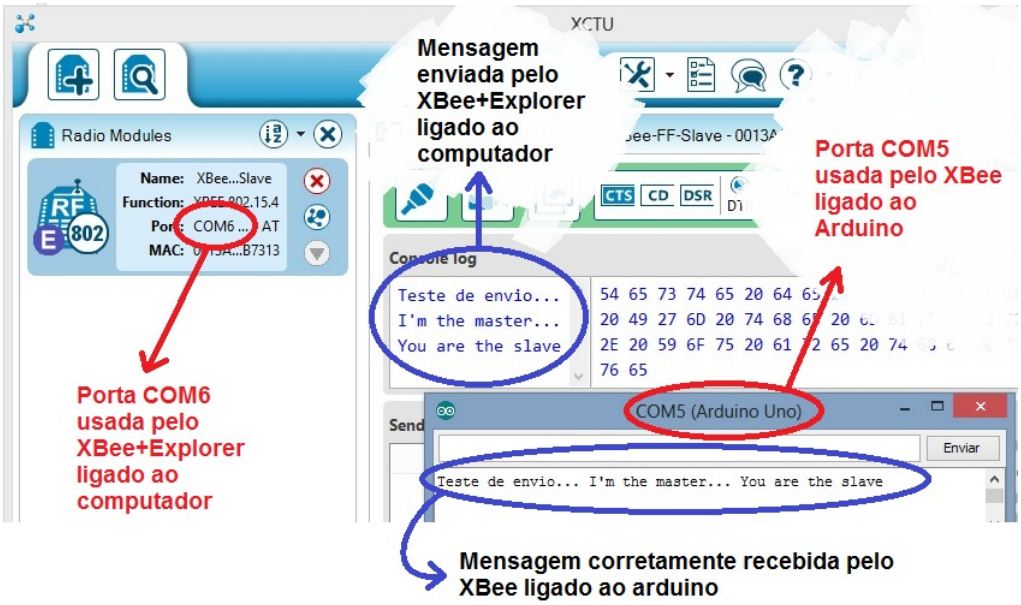


Figura 3.20: Teste de comunicação.

Nesse teste o código gravado no Arduino foi o seguinte:

```
#include <SoftwareSerial.h>
SoftwareSerial XBee(2, 3);

void setup()
{
  XBee.begin(9600);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available())
  {
    XBee.write(Serial.read());
  }
  if (XBee.available())
  {
   Serial.write(XBee.read());
  }
}
```

Cabe notar que não foram utilizados os terminais RX (receptor) e TX (transmissor) físicos do Arduino, mas sim terminais RX/TX simulados via software, por meio da biblioteca SoftwareSerial.h, nos pinos 2 e 3 do Arduino. Esta abordagem é mais prática, pois os pinos físicos citados são

utilizados no carregamento de programas para o Arduino. Dessa forma, sempre que se desejasse atualizar a programação desse microcontrolador seria necessário desfazer as conexões físicas do XBee para que não houvesse conflito.

3.3.2 Interfaceamento Arduino-MATLAB

Para a aquisição de dados através do Arduino e sua transmissão para o Matlab, seguiu-se um tutorial presente em um site especializado na comunicação serial dessas duas plataformas [12].

Criou-se um código no ADI do Arduino e outro no Matlab para estabelecer a comunicação entre ambos. O código do Arduino é dividido em 3 modos de operação principais: Estabelecimento da comunicação, aquisição de dados e controle dos motores. Para distinção entre esses modos é utilizada a variável "Mode" no código do Arduino. A seguir, explicar-se-á o modo de estabelecimento da comunicação.

Para esse modo, utiliza-se a função do Matlab *setup_serial_xbee*. Primeiramente, envia-se um caractere do Matlab por meio da função "fprintf(s,'%c','a');". Este caracter é enviado pela porta serial, lido pelo XBee acoplado à UCP e retransmitido para o XBee conectado ao Arduino embarcado. Para isso, utiliza-se o código abaixo retirado da função citada:

```
erro = 1;
    rate = 9600;

    %Criação da porta serial e configuração de seus parâmetros
    s=serial(comPort);
    set(s,'DataBits',8);
    set(s,'StopBits',1);
    set(s,'BaudRate', rate);
    set(s,'Parity','none');
    fopen(s);

    %Teste de comunicação
    %um caracter 'a' é enviado ao arduino e espera-se que um caracter 'a'
    %seja recebido de volta
    fprintf(s,'%c','a');
    a='b';
    while(a~='a')
        a=fscanf(s,'%s');
    end
    if(a=='a')
        disp('serial read')
        mbox = msgbox('Serial Communication setup'); uiwait(mbox);
        erro = 0; %operação bem sucedida, flag de erro falsa.
```

```
end
```

Nesse código, cria-se um objeto do tipo porta serial "s". Este é configurado e posteriormente habilitado para comunicação através da função "fopen()". Depois disso, utiliza-se a função "fread()" para ler um caractere até que seja identificado o caractere "a", que foi previamente enviado pelo Arduino. Por fim, coloca-se na tela uma mensagem de aviso para que o usuário saiba que a comunicação serial foi bem sucedida.

A seguir, está o código presente no Arduino que atua simultaneamente com o código do Matlab mostrado anteriomente.

```
...
Mode = Xbee.read();
...
switch(Mode)
{
...
case 'a':
   {
     Xbee.println('a');
     break;
   }

   default:
   {
     break;
   }
   ...
```

Nesse código nota-se que utilizou-se a função "Xbee.read()" para ler o caractere enviado através do XBee e a função "Xbee.println()" para enviar um caracter por meio do XBee para a UCP. Cabe ressaltar que alguns trechos do código foram suprimidos pois não são relevantes para esta seção.

Adicionalmente, cabe destacar que utiliza-se esta comunicação em diversas outras funções implementadas, sempre com a variável "Mode" para verificação do caracter enviado ao Arduino e seu respectivo modo de operação.

3.3.3 Teste dos sensores

Como citado anteriormente, para a determinação da atitude do sistema, utilizou-se uma UMI Adafruit 9DOF formada pelos sensores LSM303DLHC, que mede a aceleração e a velocidade angular e L3GD20, que mede o campo magnético.

Testou-se cada um desses sensores individualmente através da utilização dos exemplos e biblio-

tecas providos pela fabricante Adafruit. Fez-se o download dos mesmos através de um repositório no Github fornecido pela fabricante [10].

3.3.4 Procedimento de Teste

Primeiramente testou-se o acelerômetro através do uso do exemplo "accelsensor" que possui o recurso de coletar dados em tempo real da aceleração da UMI. O seguinte conjunto de dados foi obtido:

```
X: -0.55  Y: -0.39  Z: 9.85  m/s^2
X: -0.55  Y: -0.35  Z: 9.77  m/s^2
X: -0.63  Y: -0.31  Z: 9.85  m/s^2
X: -0.63  Y: -0.39  Z: 9.81  m/s^2
X: -0.63  Y: -0.35  Z: 9.77  m/s^2
X: -0.59  Y: -0.47  Z: 9.85  m/s^2
X: -0.59  Y: -0.39  Z: 9.89  m/s^2
X: -0.63  Y: -0.39  Z: 9.92  m/s^2
X: -0.63  Y: -0.35  Z: 9.85  m/s^2
X: -0.59  Y: -0.39  Z: 9.89  m/s^2
X: -0.63  Y: -0.39  Z: 9.89  m/s^2
X: -0.63  Y: -0.35  Z: 9.85  m/s^2
X: -0.63  Y: -0.39  Z: 9.77  m/s^2
X: -0.67  Y: -0.27  Z: 9.81  m/s^2
X: -0.63  Y: -0.39  Z: 9.81  m/s^2
```

Figura 3.21: Acelerômetro em repouso.

Esses dados foram coletados com a UMI em repouso. Nota-se que há uma descalibração nos 3 eixos, pois nos eixos X e Y a aceleração deveria ser nula e, no eixo Z, deveria ser em torno de $9.78 m/s^2$, que é a gravidade medida na estação geodésica número 80711435 do IBGE no Distrito Federal [11].

Posteriormente, movimentou-se a UMI e, com isso, notou-se que ela mede com sucesso a aceleração nos três eixos. Contudo, o erro de calibração continua presente.

Depois do acelerômetro, testou-se o giroscópio com o exemplo "sensorapi", com o qual obtiveram-se os seguintes valores em repouso:

```
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.03  Z: 0.02  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.00  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.03  Z: 0.01  rad/s
X: 0.00  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.02  Z: 0.02  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.00  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
X: 0.01  Y: 0.02  Z: 0.01  rad/s
```

Figura 3.22: Giroscopio em repouso.

Nota-se uma pequena descalibração nos 3 eixos, já que idealmente todos deveriam ter valores nulos. Também foram feitos testes com a UMI em movimento. Ao girá-la, variações da velocidade angular no eixo em que a rotação ocorria foram obtidas, como era esperado.

Por fim, foram realizados testes no magnetômetro através do uso do exemplo: "magsensor". Os seguintes dados foram obtidos:

```
X: 30.09  Y: 6.18  Z: 5.31  uT
X: 29.82  Y: 6.09  Z: 5.61  uT
X: 29.73  Y: 6.36  Z: 5.31  uT
X: 29.73  Y: 6.18  Z: 5.61  uT
X: 29.82  Y: 6.09  Z: 5.31  uT
X: 29.91  Y: 6.09  Z: 5.41  uT
X: 29.82  Y: 6.18  Z: 5.92  uT
X: 30.27  Y: 6.09  Z: 5.71  uT
X: 29.91  Y: 6.09  Z: 5.20  uT
X: 29.73  Y: 6.00  Z: 5.71  uT
X: 29.73  Y: 6.09  Z: 5.61  uT
X: 29.91  Y: 6.09  Z: 5.41  uT
X: 29.82  Y: 6.09  Z: 5.41  uT
X: 29.91  Y: 6.27  Z: 5.71  uT
X: 29.82  Y: 5.91  Z: 5.71  uT
```

Figura 3.23: Magnetometro com o eixo X apontando para o norte magnético da Terra.

Para obtenção dos dados da Figura 3.23, o eixo X do magnetômetro foi direcionado para o Polo Norte geomagnético e iniciou-se a coleta de dados. Nota-se que, para esse eixo, os valores medidos ficaram em torno de 30 μT, o que era esperado, já que a intensidade do campo magnético na América do Sul gira em torno desse valor [36].

3.3.5 Leitura dos Dados

Para a leitura dos dados da UMI criou-se uma função no MATLAB chamada *data_ acquisition.m* que interage com o Arduino embarcado. Ela recebe o objeto serial "s" e o número de dados que serão coletados. Este número corresponde ao número de iterações que serão utilizadas no método dos mínimos quadrados do algoritmo de balanceamento.

Esta função envia repetidamente o caracter "R" para o Arduino embarcado por meio de um laço. Para cada "R" recebido, o Arduino entra no modo de aquisição de dados, faz a leitura de 6 dados e os envia de volta para a UCP. Esses dados são: rolagem (*roll*), arfagem (*pitch*), guinada (*yaw*) e velocidade angular nos eixos x, y e z. Esses dados são armazenados em vetores para processamento posterior.

No Arduino, esses dados são coletados através dos sensores e concatenados como uma única *string*, que é enviada à UCP. Nela, a função de leitura de dados divide essa *string* em números e realiza uma verificação de erros em que verifica-se o número de informações recebidas e, caso este número esteja errado, requisita-se uma nova leitura de dados do Arduino com o envio de um novo caracter "R".

Adicionalmente, essa função também armazena os intervalos de tempo entre duas medições consecutivas, gerando um vetor com esses valores para todas as medições requisitadas. O valor desses intervalos pode variar devido à perda de dados por falhas de comunicação.

Por fim, a função *data_ acquisition.m* ainda possui o recurso de gerar gráficos em tempo real dos ângulos de Euler.

3.3.6 Calibração

Para a calibração dos sensores da UMI utilizou-se o método da determinação da curva de calibração discutido nos fundamentos teóricos.

Esse método foi implementado através da função do MATLAB *calibration_ IMU*. Ela utiliza-se da função de leitura de dados (*data_ acquisition*) para ler uma quantidade específica de dados dos sensores. Essa quantidade é escolhida pelo usuário.

Essa função é dividida em duas etapas. Primeiramente, lê-se dados da UMI em movimento. Para isso, o usuário deve rotacionar a UMI nos seus três eixos. É de suma importância que essa rotação seja feita de forma a abranger toda a excursão de cada um dos três eixos de rotação (rolagem, arfagem e guinada). Este procedimento deve ser realizado de forma lenta em torno do eixo do sensor, para que as acelerações medidas sejam fruto da gravidade e não de acelerações lineares. Esses dados são armazenados em vetores dos quais retiram-se os valores máximos e mínimos. Esses valores são utilizados para a determinação da reta de calibração através do cálculo da constante "K" e "a" para cada um dos eixos de rotação.

Para a segunda etapa, a UMI não deve ser movimentada pois almeja-se coletar os dados do giroscópio em repouso. Com esses dados calcula-se a média das velocidades angulares que serão

utilizadas como *offset* para a calibração.

Por fim, as constantes geradas por essa função são gravadas em um arquivo, que é utilizado pela função de balanceamento da mesa.

Cabe mencionar que, para a primeira etapa, utilizaram-se 2000 medições e, para a segunda, 500. Essas quantidades foram escolhidas arbitrariamente de modo a permitir uma coleta de dados por tempo suficiente.

No algoritmo de balanceamento utilizar-se-ão esses dados para calibrar os valores de arfagem, rolagem e as velocidades angulares. O trecho de código utilizado para tal é

```
event_gyro_x = event_gyro_x-omegax_offset;
        event_gyro_y = event_gyro_y-omegay_offset;
        event_gyro_z = event_gyro_z-omegaz_offset;
        roll = roll*roll_K + roll_a;
        pitch = pitch*pitch_K + pitch_a;
```

sendo que *roll* é a variável de rolagem, *pitch* é a variável de arfagem e as variáveis iniciadas por *event_ gyro* são referentes as velocidades angulares nos três eixos.

No código acima, cada um dos dados gerados pela UMI é calibrado. Nota-se que utilizam-se duas abordagens distintas para tal. Os valores das velocidades angulares são calibrados retirando-se os valores de *offset*, enquanto que, para os valores de rolagem e arfagem, utilizam-se as constantes da reta de calibração.

3.3.7 Algoritmo de Balanceamento

Nas subseções anteriores, explicou-se o método de aquisição dos dados e sua transmissão para o MATLAB. Nesta subseção focar-se-á nas funções criadas para fazer o tratamento desses dados, o cálculo da distância entre o centro de massa e o centro de rotação, que aqui chamaremos de "r", e a atuação nos motores.

A função *balancing_ algorithm* foi a função principal implementada para balanceamento da mesa e seu funcionamento será explicado a seguir. Além disso, nela várias outras funções são utilizadas, as quais também serão explicadas nesta subseção.

Para facilitar o entendimento, a atuação da função de balanceamento será dividida em 3 etapas: aquisição de dados, cálculo do vetor "r" e atuação dos motores.

3.3.7.1 Aquisição de dados

Primeiramente inicia-se a comunicação serial através da função *setup_ serial_ xbee*. Posteriormente, utiliza-se uma função de leitura de arquivo chamada *calibration_ data_ read*. Ela lê os dados de calibração presentes em um arquivo texto e os transforma em números, os quais serão utilizados

para a calibração dos dados. Todavia, caso esses dados não sejam encontrados, o algoritmo de balanceamento termina e a calibração dos sensores é solicitada ao usuário.

Por fim, utiliza-se a função *data_ acquisition* para ler os dados dos sensores, que são gravados em vetores.

O número de leituras realizados pode ser escolhido pelo usuário e corresponde ao número de iterações utilizadas no método dos mínimos quadrados. Nos testes, utilizaram-se 1500 leituras. Esse número foi escolhido com base na dissertação de Young [3].

Nela, realizou-se um estudo da relação entre a variância de "r" e o número de medições, no qual notou-se que, com mais de 60 segundos de medições (resultando, com uma frequência de coleta de dados de 25Hz, em 1500 medições), as variações em "r" são ínfimas. Os resultados desse estudo são discutidos em [3].

3.3.7.2 Cálculo do vetor de desbalanceamento

Para o cálculo do vetor "r" implementou-se a função *unbalance_ vector_ estimation*. Para isso, utilizam-se os vetores de dados das velocidades angulares nos três eixos, os vetores de arfagem e rolagem e o vetor que armazena os intervalos de tempo entre todas as amostras válidas.

Primeiramente, calcula-se a diferença entre os valores de velocidade angular em um instante de tempo t0 e um instante de tempo posterior t1. Isso é feito para todos os valores de velocidade angular coletados, que são armazenados na forma matricial, sendo que a diferença entre t0 e t1 é dada pelo vetor de intervalos de tempo chamado *intervals*.

Posteriormente, utiliza-se a função *Phi_ estimation* para calcular a matriz antissimétrica "Phi", que é necessária para o cálculo do vetor de desbalanceamento. Esse cálculo nada mais é do que uma aplicação direta da equação demonstrada na Seção 2.8.

Por fim, após a obtenção das matrizes, calcula-se o vetor de desbalanceamento "r" por meio da equação demonstrada na Seção 2.8.

Cabe destacar que os valores de guinada não são utilizados para o balanceamento da mesa, portando não são utilizados neste trabalho. Contudo, eles são armazenados e poderão ser utilizados em aplicações futuras.

3.3.7.3 Atuação nos motores

Para a atuação nos motores, realiza-se uma comparação entre os valores das componentes de "r" nos eixos x, y e z com valores de tolerância previamente definidos.

Se algum dos valores das componentes forem maiores que as tolerâncias estabelecidas, prossegue-se com a atuação dos motores. Caso contrário, considera-se que a mesa está balanceada e o algoritmo é terminado.

No caso da atuação dos motores, primeiramente calcula-se as direções para as quais as massas

devem ser movidas. Com esse intuito, criou-se a função *number2direction*, que converte os sinais de cada componente de "r" em direções de rotação de motor para cada eixo correspondente. Para este trabalho, convencionou-se a direção horária como direção 1 e a direção anti-horária como direção 2. Quando um dos motores gira na direção 1, a mesa coordenada acoplada a ele se move no sentido positivo de seu respectivo eixo e o contrário ocorre quando esse motor gira na direção 2. Tendo isso em mente, verifica-se o sinal de cada um dos componentes de "r" e move-se cada uma das mesas coordenadas na direção oposta para compensar o desbalanceamento.

Também é necessário calcular a distância que cada mesa coordenada deve se mover. Para isso, utiliza-se um ganho "k" aplicado ao desbalanceamento em cada eixo. Esse ganho é calculado por meio da relação entre a massa total do sistema e a massa de cada mesa coordenada, como descrito na Seção 2.9. As distâncias são calculadas multiplicando cada componente de "r" com seu respectivo ganho "k".

Por fim, realiza-se a atuação nos motores de passo através da função *motors_actuation*, que recebe como parâmetros as distâncias a serem percorridas por cada UMM e suas respectivas direções. Posteriormente, calcula-se o número de passos que cada motor deverá realizar e envia-se esta informação para os motores juntamente com um caracter "M". Este caracter é lido pelo Arduino embarcado, que entra em modo de atuação dos motores.

Vale mencionar que nesta etapa também ocorre uma verificação dos dados e, caso esses dados estejam em formato inválido, o Arduino embarcado envia uma mensagem de erro à UCP requisitando que ela reenvie o comando de atuação.

Após a atuação dos motores, o Arduino embarcado envia à UCP um caracter dizendo que a atuação foi finalizada com sucesso. Posteriormente, o ciclo se reinicia e novos dados são coletados dos sensores. Isso é feito até que as componentes de "r" nos três eixos estejam abaixo de suas respectivas tolerâncias.

3.3.8 Panorama

Finaliza-se esta seção dando uma perspectiva geral do funcionamento do programa desenvolvido. Com base na forma como o sistema embarcado, o qual possui um Arduino, interage com a UCP, a qual possui uma instância MATLAB executando as instruções implementadas e discutidas neste capítulo, pode-se entender o sistema como uma combinação cliente-servidor. O sistema embarcado representa o dispositivo servidor da rede, visto que verifica ininterruptamente a existência de requisições e, quando as recebe, responde de acordo com o que foi solicitado. A UCP, por outro lado, representa o dispositivo cliente, visto que faz requisições no decorrer da execução do algoritmo, tais como requisições de dados e requisições de atuação dos motores.

A Figura 3.24 mostra o fluxograma do lado cliente dessa interação de maneira mais ampla, ao passo que as figuras 3.25, 3.26, 3.27 e 3.28 mostram de maneira mais específica o funcionamento de alguns de seus blocos. Já a Figura 3.29, mostra o fluxograma do lado servidor.

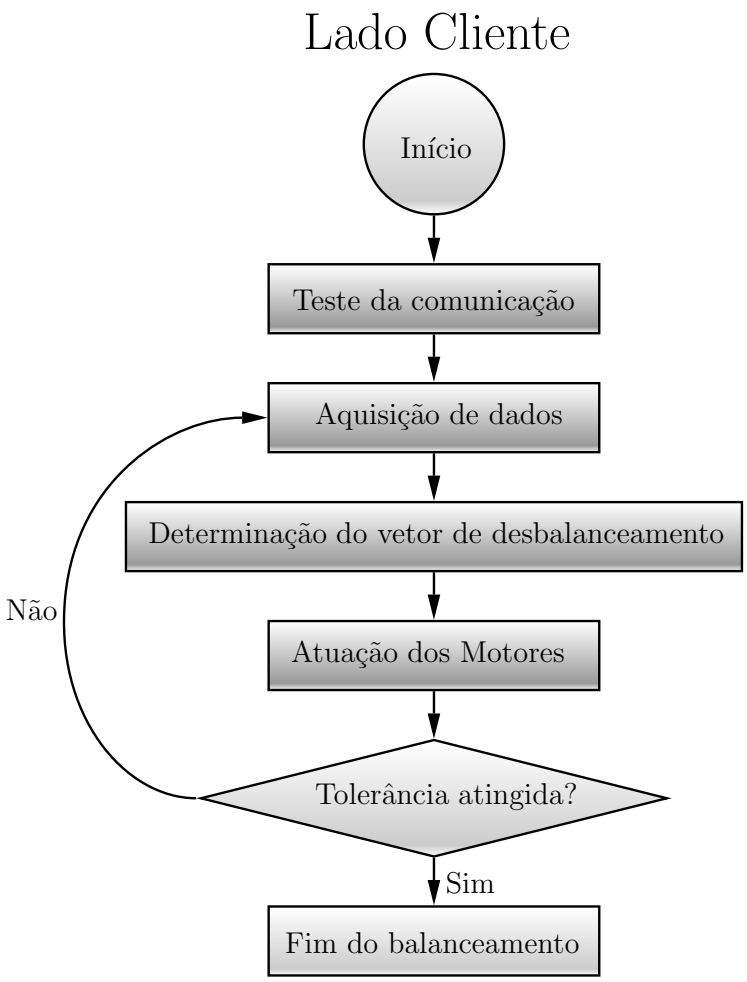


Figura 3.24: Fluxograma do lado cliente.

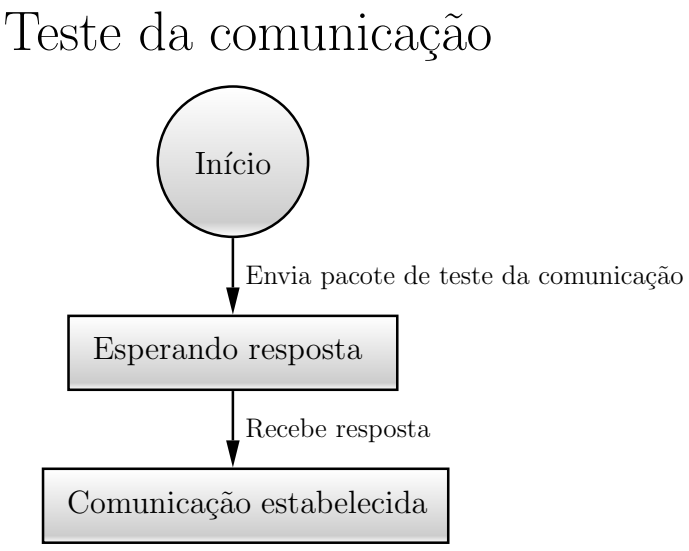


Figura 3.25: Fluxograma do módulo de teste da comunicação.

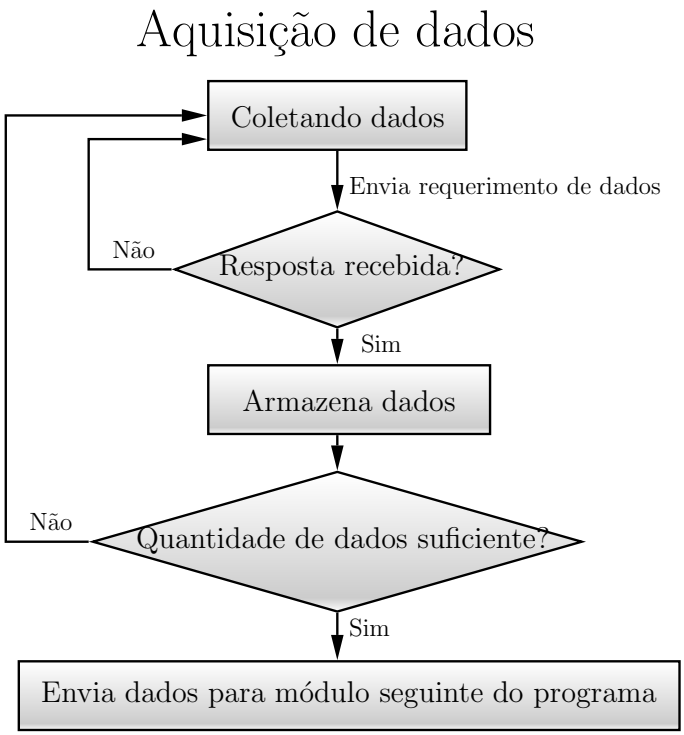


Figura 3.26: Fluxograma do módulo de aquisição de dados.

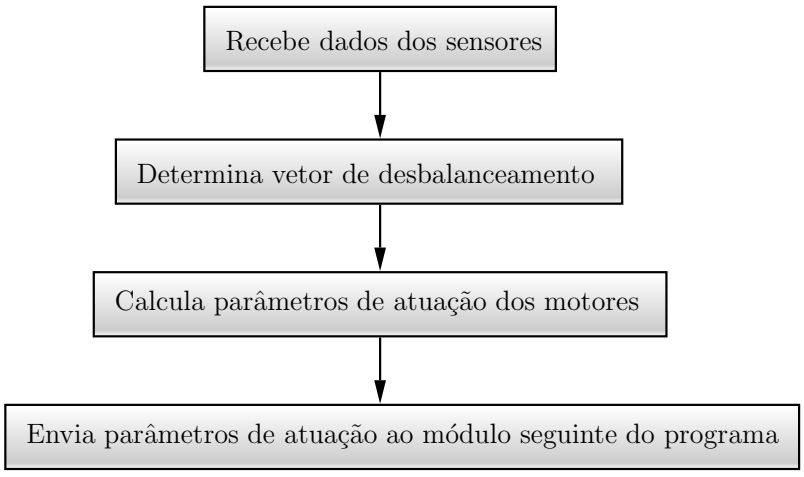


Figura 3.27: Fluxograma do módulo de cálculo do desbalanceamento.

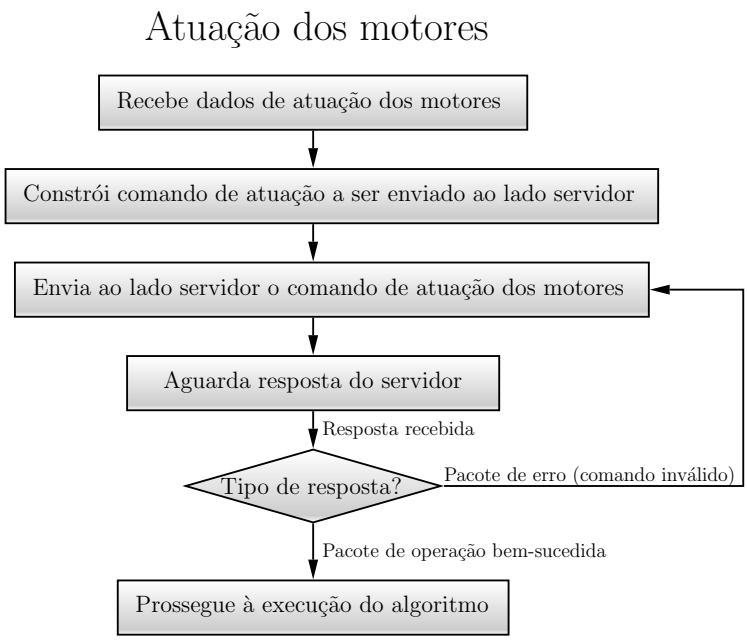


Figura 3.28: Fluxograma do módulo de atuação dos motores.

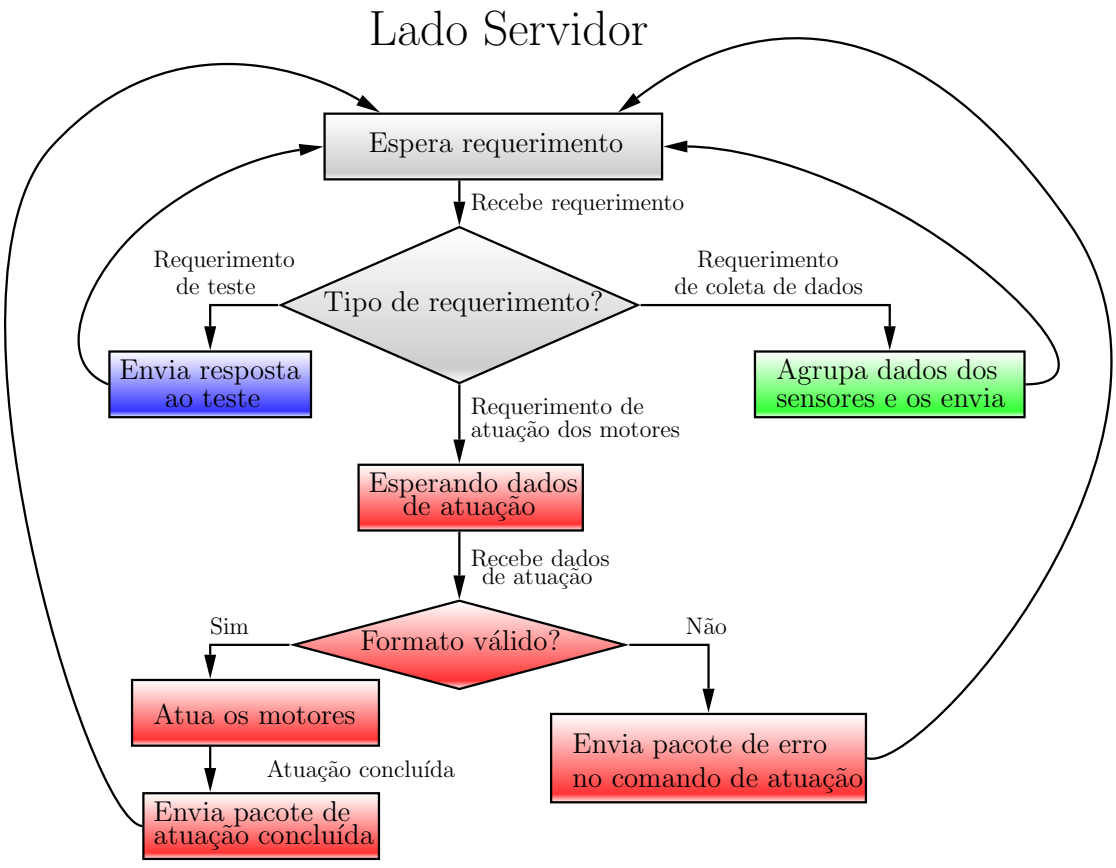


Figura 3.29: Fluxograma do lado servidor.

Capítulo 4

Testes e Resultados

Este capítulo destina-se à descrição da metologia utilizada para realização dos testes para validação do procedimento de balanceamento da mesa, assim como à exposição dos resultados obtidos.

4.1 Metodologia

Antes de descrever a metodologia utilizada para validar o procedimento de balanceamento desenvolvido, propõe-se um modelo análogo simples para a mesa.

Inicialmente, imagina-se a mesa como um pêndulo 3D com três graus de liberdade. Tal pêndulo seria capaz de oscilar em torno de dois eixos - nomeadamente, os eixos de arfagem e rolagem, simultaneamente ou não - por meio dos quais o pêndulo seria capaz de descrever uma área esférica de alcance, além de, também, poder realizar rotações em torno do eixo de guinada. A Figura 4.1 ilustra essa situação. Nela, o plano de movimentação do ângulo de rolagem está destacado.

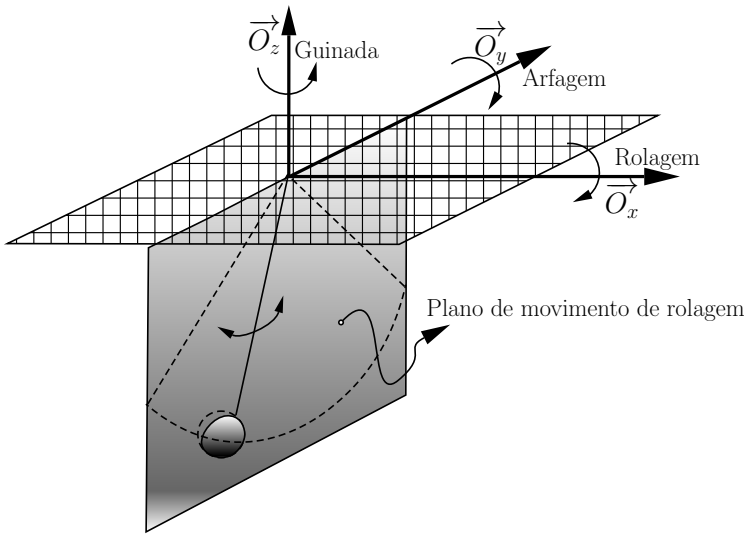


Figura 4.1: Movimento 3D do pêndulo.

Uma forma de simplificar esse modelo é ignorar o efeito de torção do pêndulo e, adicionalmente, considerar que ele se move com apenas um grau de liberdade em torno do ponto fixo do pêndulo.

Essa situação é equivalente à mostrada na Figura 2.2.

Como descrito na Subseção 2.1.3, o período de oscilação do pêndulo simples é dado pela Equação (2.20)

$$T = 2\pi \sqrt{\frac{I}{mgL}} \ . \tag{4.1}$$

Adaptando ao contexto deste trabalho, tem-se que o comprimento do fio do pêndulo, L, representa a distância r entre o CR e o CM da mesa, a massa do pêndulo, m, representa a massa total da mesa, M, e o momento de inércia do pêndulo, I, representa ou o momento de inércia principal referente ao eixo de rolagem, ou o momento de inércia principal referente ao eixo de arfagem. Dessa forma, a Equação (2.20) adaptada às variáveis de modelagem da mesa, pode ser reescrita como

$$T_{roll} = 2\pi \sqrt{\frac{I_{xx}}{Mg||\vec{r}||}} \ , \tag{4.2}$$

caso o eixo de rolagem seja escolhido e

$$T_{pitch} = 2\pi \sqrt{\frac{I_{yy}}{Mg||\vec{r}||}} \ , \tag{4.3}$$

caso o eixo de arfagem seja escolhido.

Cabe, neste momento, fazer uma observação com relação à simplificação utilizada. Erros consideráveis entre o período de oscilação da mesa e o período de oscilação previsto por esse modelo podem ser encontrados devido a dois fatores principais:

- **Determinação do desbalanceamento**
  Uma fonte de erro remete à própria utilização do algoritmo de balanceamento escolhido. O vetor de desbalanceamento, $\vec{r}$, possui intensidade muito pequena em relação aos outros termos da Equação (4.2). Além disso, a determinação do vetor $\vec{r}$ é sensível com relação à variação das medições obtidas por meio dos sensores. Outro fator que interfere na determinação do vetor de desbalanceamento diz respeito à eficácia da comunicação entre a mesa e o computador. Por exemplo, caso as baterias que alimentam o sistema embarcado na mesa apresentem baixa carga, a comunicação apresentará uma taxa elevada de perda de dados, influenciando diretamente na determinação do vetor de desbalanceamento.

- **Interação entre os eixos**
  O modelo utilizado supõe que a mesa estará oscilando em apenas um de seus eixos, no caso o eixo de arfagem ou o eixo de rolagem. No entanto, sabe-se que a mesa seria melhor descrita como um pêndulo complexo que, ao invés de possuir apenas um grau de liberdade, como é o caso do pêndulo simples, possui três e, além disso, troca energia entre esses graus.

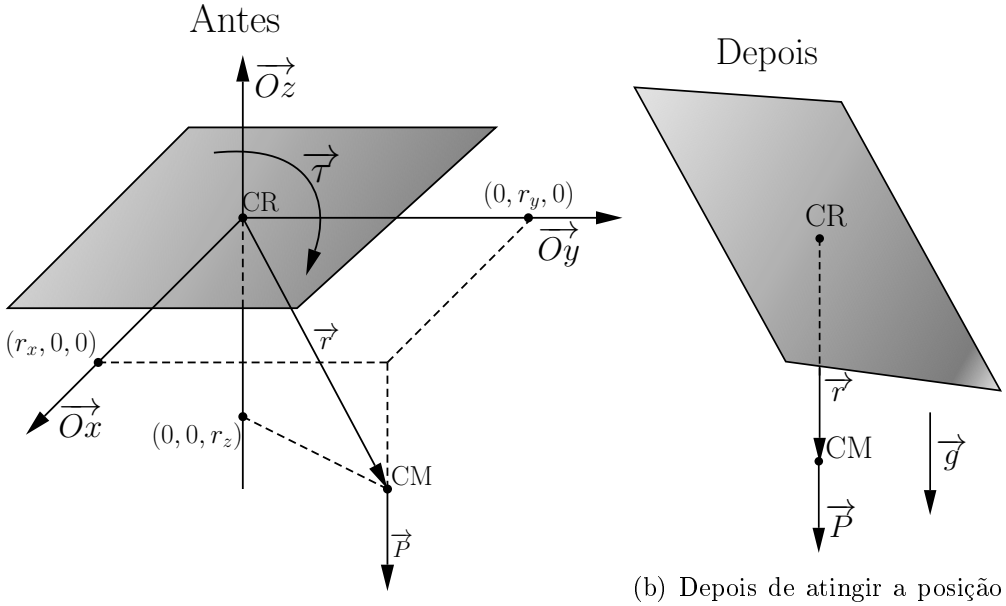
Apesar da modelagem utilizada apresentar erros consideráveis, para o propósito de validar a eficácia do procedimento de balanceamento basta que se note uma tendência: o período de oscilação

da mesa em torno do eixo de rotação em análise deve aumentar. Essa é uma consequência direta da observação de que o período, de acordo com a Equação (4.2), é inversamente proporcional ao módulo do vetor de desbalanceamento

$$T \propto \frac{1}{\sqrt{||\overrightarrow{r}||}} \, . \tag{4.4}$$

Tendo essa modelagem em vista, para validar o procedimento de balanceamento desenvolvido, buscou-se a utilização de duas abordagens. Uma delas, como citado anteriormente, é verificar a diminuição do período de oscilação da mesa em torno dos eixos de rolagem e arfagem. Outra é a de verificar se as angulações da mesa nos eixos de rolagem e arfagem se aproximam de valores nulos ou, em outras palavras, que a mesa se encontra na posição horizontal. Para ilustrar essa outra abordagem, utiliza-se a Figura 4.8, que será introduzida na Seção 4.4.

Na Figura 4.8, a posição de equilíbrio atingida pela mesa indica desbalanceamento nos três eixos do sistema de referência fixo à ela.[1] Dar-se-á atenção maior ao desbalanceamento que ocorre nos eixos de rolagem e arfagem. As figuras 4.2a e 4.2b ilustram a atuação do torque gravitacional na circunstância em que a mesa está desbalanceada.



(a) Antes de atingir a posição de equilíbrio.

(b) Depois de atingir a posição de equilíbrio.

Figura 4.2: Atuação do torque gravitacional em uma mesa desbalanceada.

Imagina-se que, caso a mesa esteja balanceada nos eixos de rolagem e arfagem, ou seja, o vetor de balanceamento apresente componentes r_x e r_y nulas, a posição de equilíbrio a ser atingida pela mesa será horizontal. Cabe enfatizar que a definição de "mesa equilibrada" adotada é a de condição na qual as componentes de desbalanceamento em x e y estão próximas de zero ou são desprezíveis e

[1] Na Figura 4.8, o desbalanceamento nos eixos de rolagem e arfagem são indiscutíveis. No caso do eixo de guinada, imagina-se primeiramente que não há desbalanceamento nesse eixo. Nesse caso, o desbalanceamento nos outros eixos faria com que a tendência da mesa fosse de permanecer completamente vertical, sendo impedida de atingir essa configuração apenas pelo suporte físico da plataforma, o que não se observa. Logo, conclui-se que o eixo de guinada também apresenta desbalanceamento.

a componente de desbalanceamento em z é, também, minimizada, porém estritamente negativa. O requisito dado ao desbalanceamento em z tem como objetivo evitar que a mesa assuma condições de equilíbrio instável.[2]

Definição 3 *A mesa com rolamento a ar é dita "balanceada" se apresentar as seguintes condições*

$$\begin{aligned} r_x &\approx 0 \ , \\ r_y &\approx 0 \ , \\ r_z &\approx 0 \ , \\ r_z &< 0 \ , \end{aligned} \tag{4.5}$$

em que r_x, r_y e r_z são as componentes cartesianas do vetor de desbalanceamento da mesa e valores negativos de r_z indicam um desbalanceamento no eixo z no mesmo sentido do vetor de gravidade.

Dado que no protótipo atual da mesa não é possível realizar deslocamentos muito longos no eixo z, o que impede que o desbalanceamento nesse eixo seja muito minimizado, é assumido que uma forma simplificada de verificar a eficácia do procedimento de balanceamento é a de verificar apenas os valores dos ângulos de rolagem e arfagem e das componentes do vetor de desbalanceamento, certificando-se de que esses valores se aproximam de zero a cada iteração do algoritmo.

Outro item importante a ser discutido é a respeito da calibração da UMI. A inclinação da mesa ao fim do processo de balanceamento será consequência direta das medições dos sensores. Porém, caso o ângulo nulo calculado pelos sensores não aparente ser, de fato, nulo, ou seja, não aparente estar de acordo com a referência inercial do sistema, a UMI requererá uma calibração mais precisa e robusta do que a discutida na Subseção 2.1.6.

Dentre as razões para esse tipo de descalibração, podem ser citadas as seguintes:

- Má fixação da IMU na placa eletrônica do sistema: a IMU deve estar perfeitamente paralela à placa eletrônica e deve ser montada nela de modo a permitir o alinhamento preciso dos eixos da UMI com os eixos de movimentação das massas móveis.
- Má fixação da placa eletrônica na mesa: a própria placa eletrônica deve ser fixada na mesa com o mesmo rigor que a UMI é fixada na placa eletrônica.
- Descalibração inerente aos sensores: por mais que os sensores utizados possuam calibração de fábrica, é necessário realizar nova calibração caso resultados mais precisos sejam desejados. Além disso a calibração de fábrica ocorre em condições diferentes das presentes no laboratório (*e.g.* a gravidade local é diferente, podendo gerar diferenças pequenas, porém perceptíveis).

Dessa forma, uma calibração adicional foi utilizada. A mesa foi posta na posição horizontal real, com auxílio de um nível de bolha, por *tentativa e erro* e os ângulos de arfagem e rolagem foram medidos durante um intervalo de tempo longo. A média dessas medidas foi tomada, resultando nos ângulos

[2] A Seção 2.10 deste trabalho discute as posições de equilíbrio instável e estável da mesa.

$$
\begin{aligned}
\textit{desbalanceamento em rolagem} = \overline{\phi} = E(\phi_1, \phi_2, \ldots, \phi_n) = 1.1533^\circ \ , \\
\textit{desbalanceamento em arfagem} = \overline{\theta} = E(\theta_1, \theta_2, \ldots, \theta_n) = 3.8159^\circ \ .
\end{aligned} \tag{4.6}
$$

Os valores de $\overline{\phi}$ e $\overline{\theta}$ foram considerados como *offsets* de medição e, por isso, foram subtraídos das medições feitas pela UMI. Ou seja,

$$
\begin{aligned}
\phi_{real} = \phi_{medido} - \overline{\phi} \\
\theta_{real} = \theta_{medido} - \overline{\theta} \ .
\end{aligned} \tag{4.7}
$$

Com isso, é garantido que a posição final do balanceamento da mesa seja, aproximadamente, horizontal com relação ao sistema de referência inercial.

A figura 4.3 ilustra a interface do programa de balanceamento durante a aquisição dos dados utilizados nesta seção.

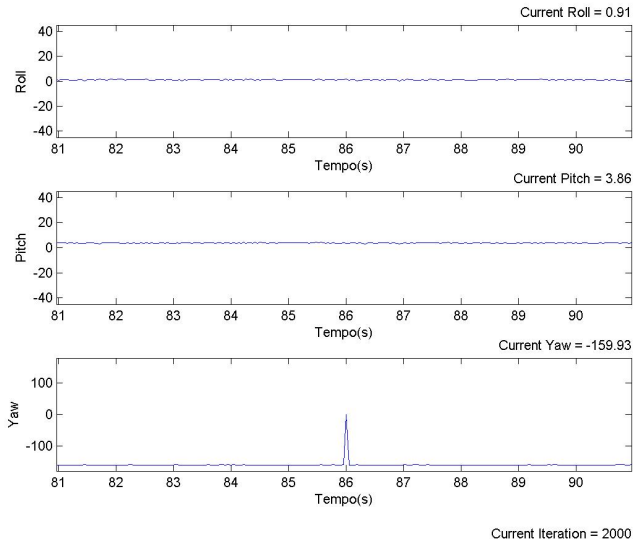


Figura 4.3: Coleta de dados de desbalanceamento.

4.2 Estimação do tensor de inércia

Um dos fatores essenciais para que o algoritmo de balanceamento demonstre um bom desempenho é ter uma boa aproximação do tensor de inércia do sistema. De maneira mais específica, é necessário que se tenha uma boa aproximação para os valores dos momentos principais de inércia, I_{xx}, I_{yy} e I_{zz}, já que esses valores são utilizados para se estimar o vetor de desbalanceamento do sistema.

Existem diversas formas de se estimar esses valores. Algumas delas são:

- Estimação analítica: por meio de simplificações do formato geométrico dos componentes do sistema e, sabendo a massa e estimando a posição do centro de massa desses componentes, é possível determinar uma aproximação analítica para o tensor de inércia.
- Estimação por programas de projeto do tipo CAD (do inglês, *Computer Aided Design* ou "Desenho Assistido por Computador"): por meio de programas do tipo CAD é possível desenhar de maneira precisa o sistema e, utilizando as ferramentas disponibilizadas por esse tipo de programa, é possível estimar várias de suas propriedades físicas e mecânicas (*e.g.* massa, volume, tensor de inércia).
- Por meio de experimentos que analisem a dinâmica da mesa. [2]

Neste trabalho, o tensor de inércia foi estimado por meio de uma estimação obtida no programa CATIA, que é do tipo CAD.

A Figura 4.4 ilustra a interface do programa e a estimação dos momentos de inércia principais é apresentada na Tabela 4.1.

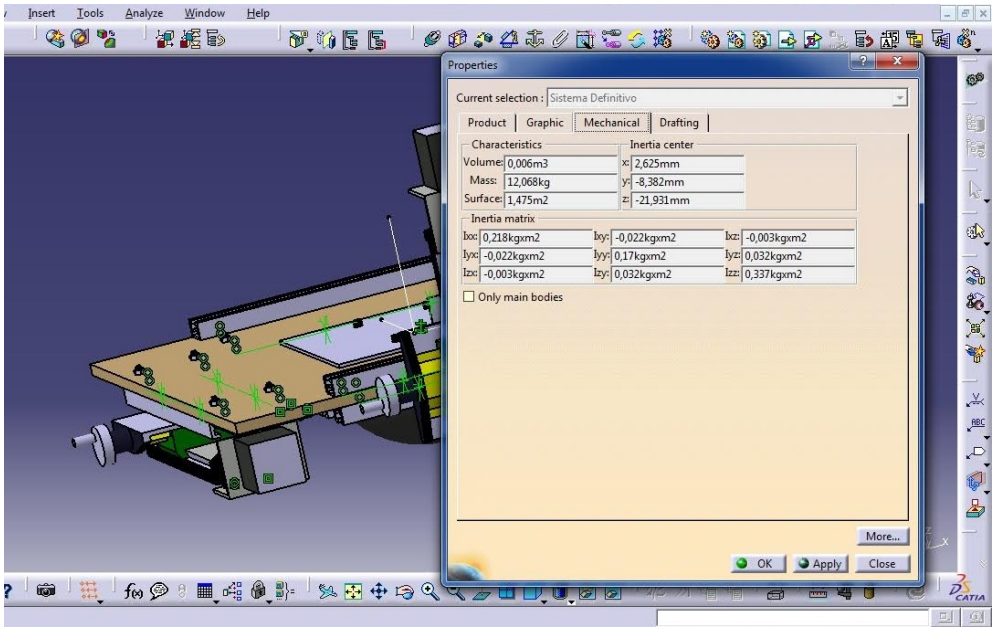


Figura 4.4: Interface do programa CATIA mostrando o tensor de inércia utilizado.

Componente do tensor de inércia	Valor estimado ($kg \cdot m^2$)
I_{xx}	0,218
I_{yy}	0,17
I_{zz}	0,337

Tabela 4.1: Momentos principais de inércia obtidos por meio do programa CATIA.

4.3 Avaliação do método de estimação do vetor de desbalanceamento

Antes de prosseguir com os resultados dos testes de balanceamento da mesa, uma avaliação da eficácia do método de estimação do vetor de desbalanceamento foi realizada. O propósito dessa avaliação é quantificar o quão precisas são as medidas do vetor de desbalanceamento.

Cabe comentar que, nesta seção, a acurácia do método não é avaliada, visto que essa depende da acurácia de outras medições como, principalmente, da medição do tensor de inércia. Como o tensor de inércia é obtido por aproximações, seja por meio do programa CATIA ou por meio de estimações analíticas, a acurácia do método não será estudada neste trabalho.

O método utilizado consiste em desligar a atuação dos motores e fazer inúmeras estimações do vetor de desbalanceamento. Como os motores estarão desligados, pressupõe-se que o vetor de desbalanceamento medido será sempre o mesmo e, portanto, variações em sua medida seriam decorrentes apenas de variações inerentes à medição.

Este item é subdividido em duas classes de testes. São elas

- Testes estáticos: são os testes realizados na condição em que a mesa está parada e com a atuação dos motores desligada; e
- Testes dinâmicos: são os testes realizados na condição em que a mesa está se movimentando livremente e a atuação dos motores está desligada. Para esses testes a mesa foi submetida a uma perturbação manual antes de se iniciarem as medições.

Foram realizados três testes: dois dinâmicos e um estático. Como o teste estático implica que as medições não variarão muito no decorrer do tempo, foram utilizadas 200 medições para realização de cada estimativa do vetor de desbalanceamento por meio do método dos mínimos quadrados. No caso dinâmico, para efeito de comparação com o teste estático, foi realizado um teste com a mesma quantidade de medições por estimativa. Além disso, um teste dinâmico adicional foi feito com 1500 medições por estimativa para analisar o efeito do aumento de medições na precisão das estimativas, totalizando assim os três testes.

A figura 4.5 ilustra a variação do vetor de desbalanceamento para várias medições sob a condição de teste estático, ao passo que as figuras 4.6 e 4.7 ilustram a variação desse vetor sob a condição de teste dinâmico com 200 e 1500 medidas por estimativa, respectivamente.

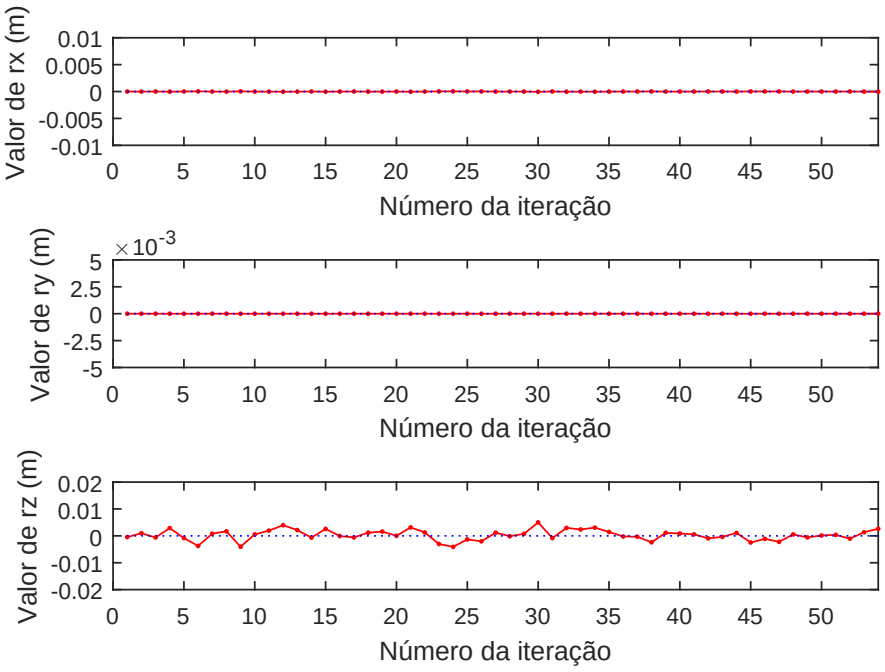


Figura 4.5: Estimativas do vetor de desbalanceamento (teste estático com 200 medições por estimativa).

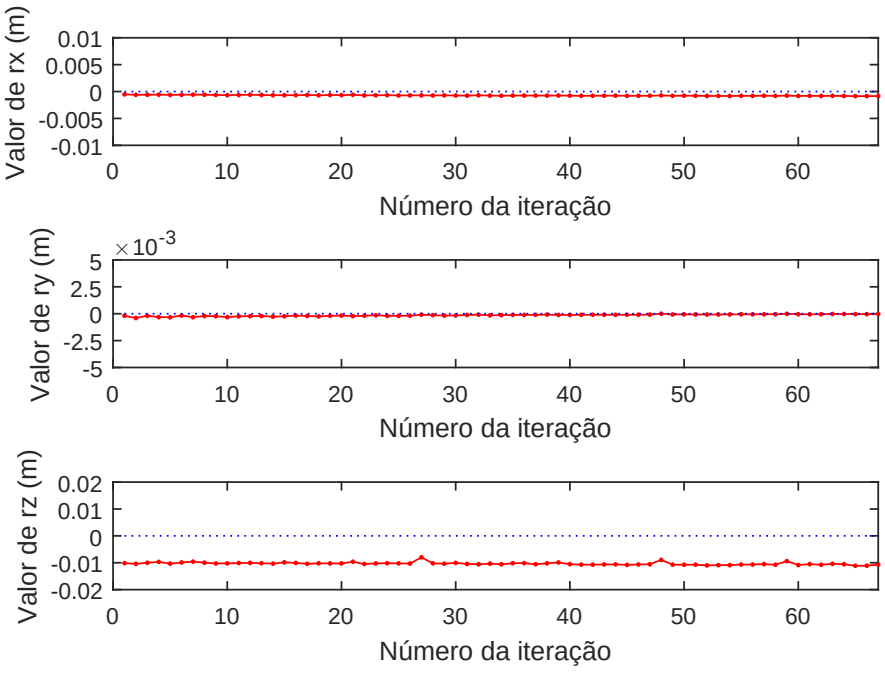


Figura 4.6: Estimativas do vetor de desbalanceamento (teste dinâmico com 200 medições por estimativa).

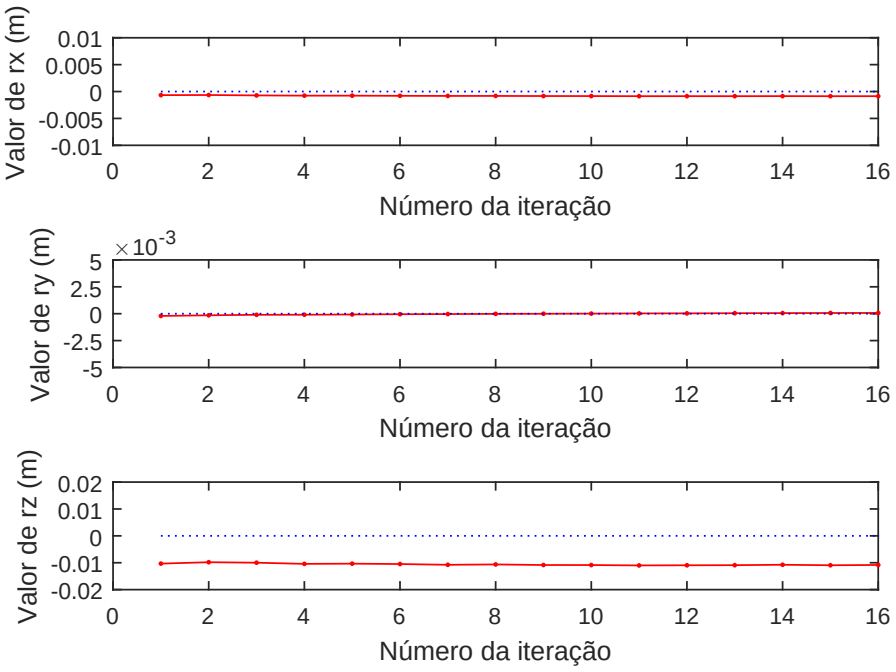


Figura 4.7: Estimativas do vetor de desbalanceamento (teste dinâmico com 1500 medições por estimativa).

A tabela 4.2 resume os desvios-padrão obtidos em cada eixo para cada teste.

Teste	Medidas por estimativa	Eixo	Desvio Padrão (m)
Estático	200	x	$1.8 \cdot 10^{-5}$
Estático	200	y	$1.8 \cdot 10^{-6}$
Estático	200	z	$2 \cdot 10^{-3}$
Dinâmico	200	x	$8.0 \cdot 10^{-5}$
Dinâmico	200	y	$8.9 \cdot 10^{-5}$
Dinâmico	200	z	$4.9 \cdot 10^{-4}$
Dinâmico	1500	x	$7.1 \cdot 10^{-5}$
Dinâmico	1500	y	$8.1 \cdot 10^{-5}$
Dinâmico	1500	z	$3.6 \cdot 10^{-4}$

Tabela 4.2: Desvios-padrão na medição do vetor de desbalanceamento.

Um fator cruxial cabe ser mencionado: a condição de teste influencia diretamente no funcionamento do algoritmo de balanceamento. No teste estático, todas as componentes do vetor de desbalanceamento obtidas estavam próximas de zero ou oscilando em torno desse valor. Para os eixos x e y isso é válido, pois os testes foram realizados com a mesa balanceada nos ângulos de arfagem e rolagem. No entanto, a componente em z apresenta comportamento **errôneo**, já que seu valor não apresentaria, para as condições de teste adotadas, valores tão baixos, tampouco positivos.

Já nos testes dinâmicos, o valor de desbalanceamento no eixo z é bem definido, mais preciso e

- o mais importante - coerente. Esse foi o motivo pelo qual optou-se por realizar o balanceamento da mesa em condições dinâmicas e não estáticas.

Quanto à precisão, pode-se dizer que as estimativas para a componente em z do vetor de desbalanceamento são muito mais imprecisas do que as obtidas nos eixos x e y. Isso resulta em curvas de balanceamento muito mais suaves nos eixos x e y, como poderá ser visto posteriormente neste capítulo. Nota-se, também que, no eixo z, as estimativas são muito mais precisas com a realização de testes dinâmicos do que com testes estáticos. Testes estáticos apresentaram menor variação nos eixos x e y, porém, devido à natureza errônea dessas estimativas, esse tipo de teste foi desconsiderado.

4.4 Resultados dos testes

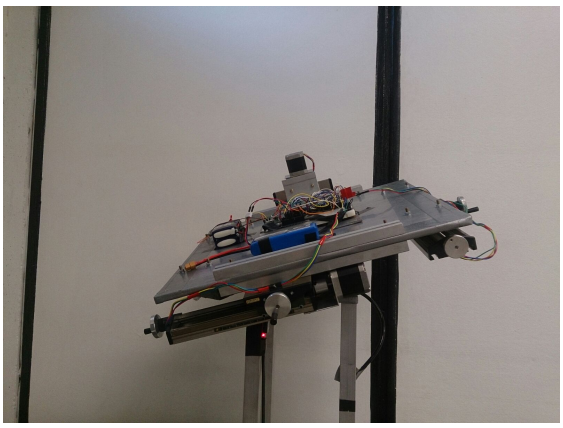
Para validar o procedimento de balanceamento, o comportamento do ângulo de arfagem, do ângulo de rolagem e das componentes do vetor de desbalanceamento foram verificados a cada iteração do algoritmo[3] com a mesa partindo de uma condição desbalanceada.

Cada iteração utilizou 1500 amostras de dados de orientação coletados. O procedimento adotado foi o seguinte:

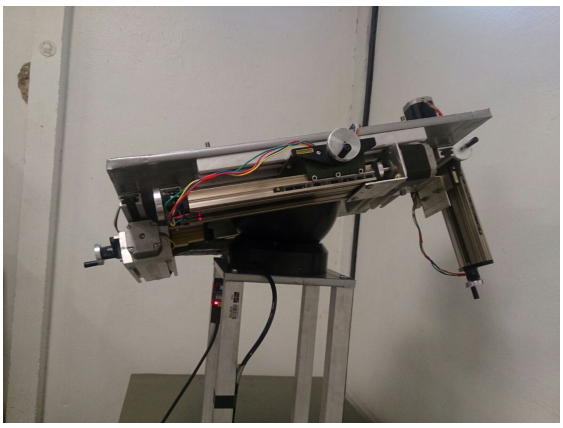
1. Uma perturbação é aplicada à mesa;
2. Inicia-se a coleta de dados;
3. Um vetor de desbalanceamento é calculado e os motores são atuados de maneira a minimizar o desbalanceamento;
4. Repete-se até que a mesa esteja balanceada. Cabe notar que entre cada iteração há uma pausa para perturbar a mesa e, então, iniciar a coleta de dados.

A condição inicial de desbalanceamento adotada foi atingida por meio de ajustes no peso de pré-balanceamento dela e da movimentação das UMMs até que a posição ilustrada pelas figuras 4.8a e 4.8b fosse observada.

[3]Entende-se por iteração do algoritmo a ação de: i) coletar dados de orientação da mesa; ii) a partir dos dados coletados, estimar um vetor de desbalanceamento; e iii) com base no vetor de desbalanceamento calculado, atuar nos motores para realizar a ação de correção.



(a) Configuração inicial da mesa desbalanceada (frente).

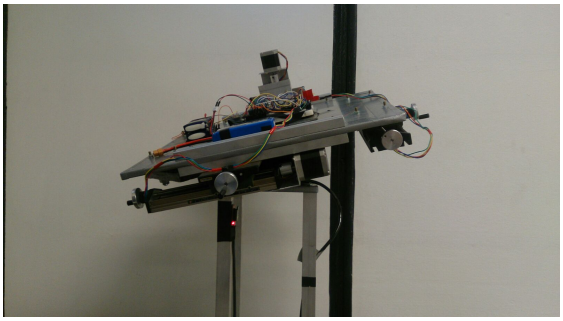


(b) Configuração inicial da mesa desbalanceada (lado).

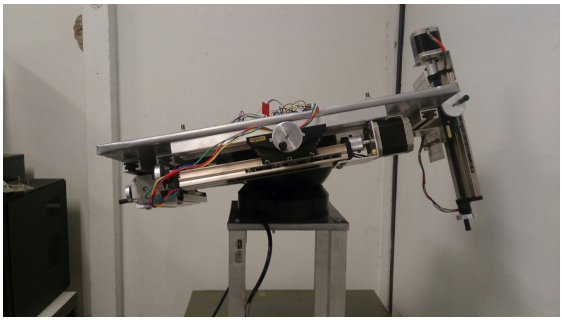
Figura 4.8: Configuração inicial da mesa.

Em seguida, iniciou-se o procedimento de balanceamento. As figuras 4.9a a 4.9d e 4.10a a 4.10f mostram as novas configurações atingidas pela mesa a cada iteração do algoritmo. Essas figuras permitem observar, além da orientação da mesa, também o deslocamento das UMMs. A partir da quinta iteração a configuração da mesa não foi mais registrada por meio de fotos, visto que desse ponto em diante não se pode perceber variações significativas da orientação da mesa a olho nu.

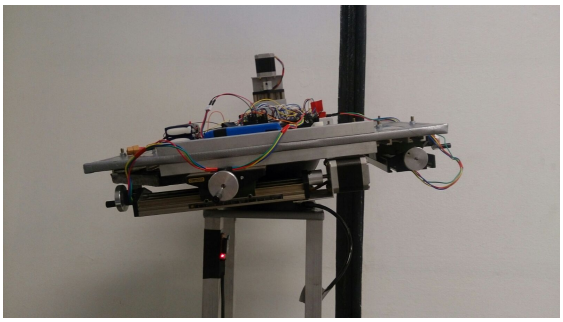
Figura 4.9: Configurações atingidas pela mesa a cada iteração do algoritmo de balanceamento.



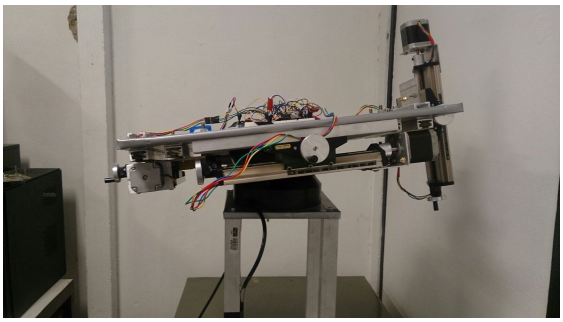
(a) Configuração da mesa após a primeira iteração do algoritmo (frente).



(b) Configuração da mesa após a primeira iteração do algoritmo (lado).

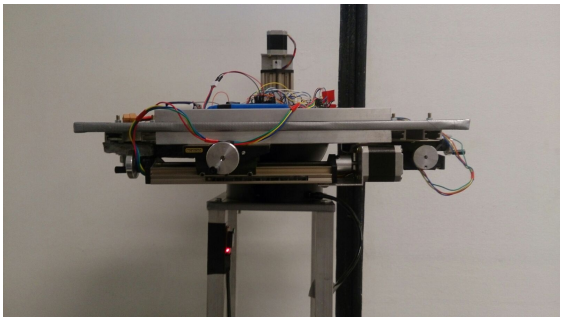


(c) Configuração da mesa após a segunda iteração do algoritmo (lado).

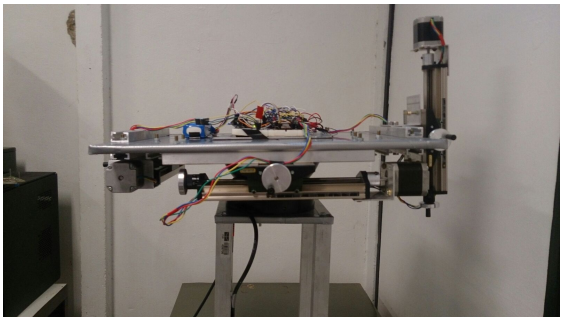


(d) Configuração da mesa após a segunda iteração do algoritmo (lado).

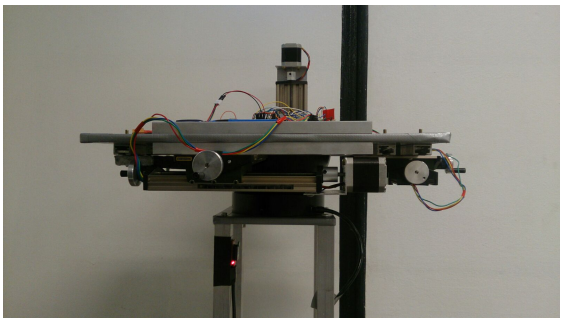
Figura 4.10: Configurações atingidas pela mesa a cada iteração do algoritmo de balanceamento (continuação).



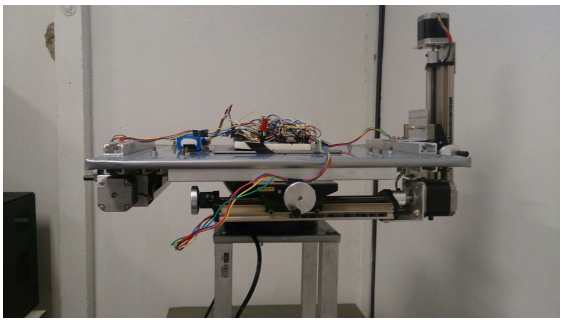
(a) Configuração da mesa após a terceira iteração do algoritmo (frente).



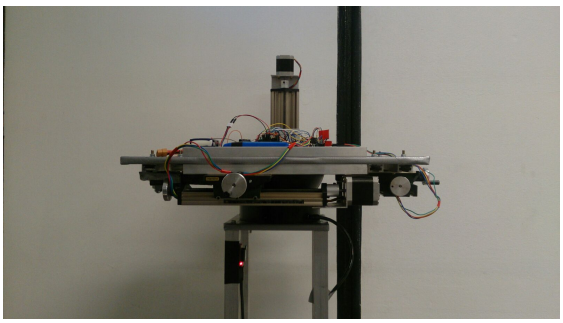
(b) Configuração da mesa após a terceira iteração do algoritmo (lado).



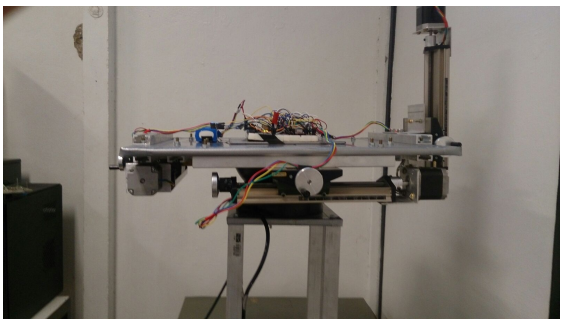
(c) Configuração da mesa após a quarta iteração do algoritmo (frente).



(d) Configuração da mesa após a quarta iteração do algoritmo (lado).



(e) Configuração da mesa após a quinta iteração do algoritmo (frente).



(f) Configuração da mesa após a quinta iteração do algoritmo (lado).

Os gráficos das figuras 4.11, 4.12 e 4.13 mostram a evolução das componentes do vetor de desbalanceamento no decorrer das iterações, ao passo que a Figura 4.14 exibe o comportamento dos ângulos de rolagem e arfagem da mesa.

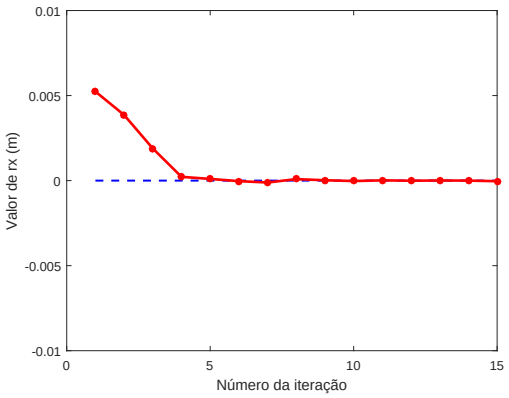


Figura 4.11: Histórico das componentes x do vetor de desbalanceamento.

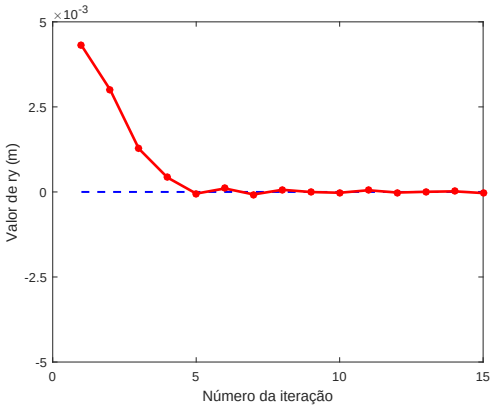


Figura 4.12: Histórico das componentes y do vetor de desbalanceamento.

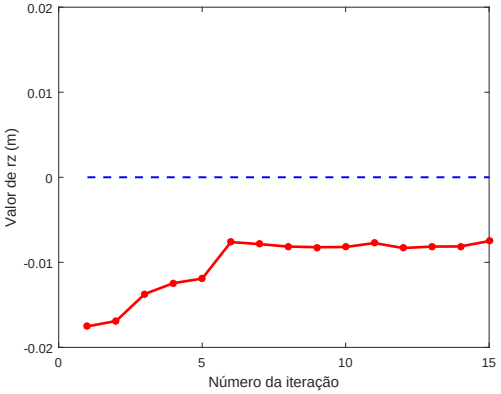


Figura 4.13: Histórico das componentes z do vetor de desbalanceamento.

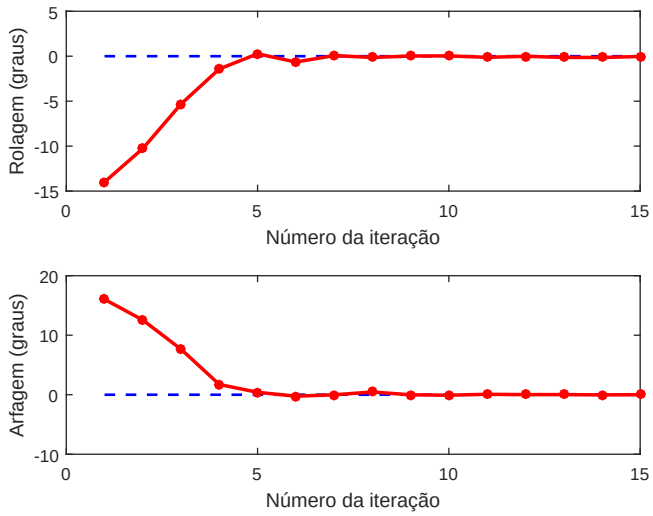


Figura 4.14: Histórico dos ângulos de arfagem e rolagem durante o balanceamento.

A partir da observação dessas figuras, nota-se que o procedimento de balanceamento revelou-se bem-sucedido, visto que os ângulos de arfagem e rolagem tendem a diminuir, assim como as componentes do vetor de desbalanceamento.

Cabe uma observação a respeito do comportamento da componente z do vetor de desbalanceamento visto na Figura 4.13: seu valor oscila em torno de um valor próximo de -0.008m. Isso é devido ao fato de que, após a quinta iteração do algoritmo, o motor da UMM do eixo z foi desligado a fim de preservá-lo, o que impede que essa componente continue sendo reduzida. Uma vez que a UMM atinge seu limite, a atuação do motor causaria uma sobrecarga do mesmo, levando-o a, eventualmente, queimar.

Outro fato notável é o de que a curva da componente z do vetor de desbalanceamento é menos suave do que as das componentes x e y. Isso era esperado devido ao maior desvio padrão inerente às medidas desse eixo. Para ilustrar a coerência dessa curva, a Figura 4.15 é mostrada. Nessa figura, as curvas somadas e subtraídas de três vezes o desvio-padrão associado a esse eixo foram representadas[4].

[4]O intervalo de $r_z - 3\sigma$ a $r_z + 3\sigma$, sendo σ o desvio-padrão, compreende uma faixa de valores correspondente a 99.7% das amostras de uma mesma medida.

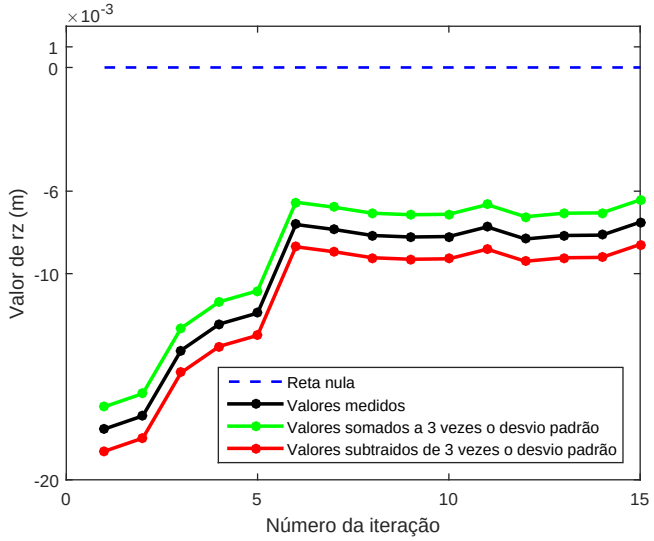


Figura 4.15: Evolução da componente z do vetor de desbalanceamento (desvio-padrão adicionado).

Para testar a outra abordagem de validação do procedimento de balanceamento, a evolução do período de oscilação dela no decorrer das iterações do algoritmo foi verificado, partindo de uma posição desbalanceada da mesa.

Nesse caso, foram realizados testes dinâmicos: uma perturbação inicial foi aplicada à mesa no início da execução do algoritmo. Cabe lembrar que a perturbação deve, na medida do possível, resultar em oscilações apenas no eixo de arfagem, pois esse foi o eixo escolhido para esse teste. Em seguida, o período de oscilação da mesa foi mensurado por meio da análise do gráfico do ângulo de rolagem. Esse procedimento foi aplicado ao sistema por mais 4 vezes. A Tabela 4.3 mostra os períodos de oscilação obtidos a cada iteração, além do período teórico previsto pela Equação (4.2) e o correspondente vetor de desbalanceamento. Os períodos foram medidos por meio dos dados obtidos dos sensores e armazenados na UCP.

	$\|\|\vec{r}\|\|$ (mm)	Período do pêndulo (calculado)	Período de oscilação da mesa (eixo de arfagem)
Condição inicial	12.9mm	1.8758s	1.845s
Iteração 1	11.7mm	1.9652s	1.8356s
Iteração 2	9.4mm	2.1985s	2.116s
Iteração 3	7.2mm	2.5085s	2.134s
Iteração 4	4.2mm	3.2645s	2.404s

Tabela 4.3: Evolução do período de oscilação no decorrer das iterações do algoritmo.

A Tabela 4.3 indica que, como previsto, o período de oscilação da mesa aumenta, o que reforça a coerência do procedimento de balanceamento. Era esperado que houvesse uma discrepância entre o período de oscilação teórico (calculado) e o período de oscilação real da mesa a cada iteração devido às dificuldades operacionais de se realizar esse teste e isso se confirmou. Porém, nessa abordagem a observação exclusiva do aumento dos períodos teórico e real é suficiente.

Conforme foi mencionado anteriormente, uma das causas desse erro é proveniente da dificuldade de se isolar o movimento de um eixo do sistema. Para ilustrar isso, verificam-se as figuras 4.16, 4.17 e 4.18.

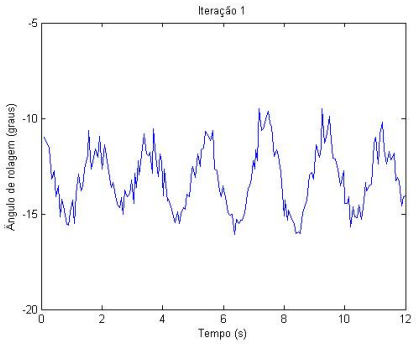


Figura 4.16: Oscilação do ângulo de rolagem da mesa após a primeira iteração.

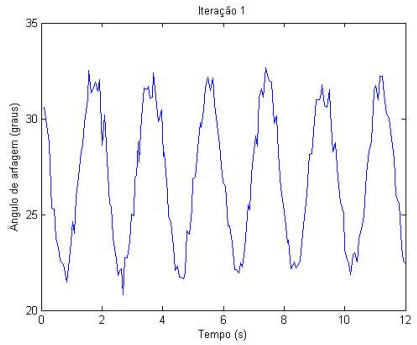


Figura 4.17: Oscilação do ângulo de arfagem da mesa após a primeira iteração.

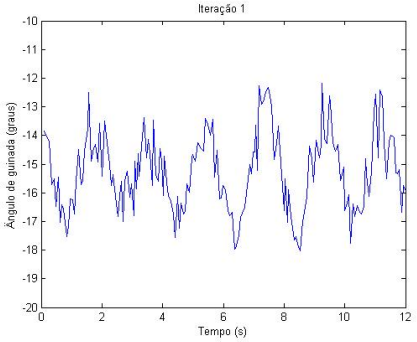


Figura 4.18: Oscilação do ângulo de guinada da mesa após a primeira iteração.

Como é observado nessas figuras, após a primeira ação de correção executada pelos motores das UMMs todos os eixos de orientação da mesa já apresentam oscilações, ou seja, não foi possível isolar o movimento do eixo de arfagem por meio do procedimento experimental adotado, já que eram esperadas oscilações menores nos eixos de rolagem e guinada.

É devido às dificuldades operacionais de se realizar esse tipo de teste com os aparatos disponíveis que somente a oscilação em arfagem foi analisada.

4.5 Desempenho do rolamento a ar

Por fim, cabe mencionar que, durante os testes, a plataforma apresentou um leve amortecimento. Esse fenômeno se observa devido à existência, no ambiente em que se encontra a plataforma, de ar e o atrito de arrasto aerodinâmico que ele causa na mesa.

Como dito anteriormente neste trabalho, o rolamento a ar reduz drasticamente o atrito experimentado pela mesa durante sua movimentação. Além disso, para os propósitos de equacionamento dinâmico da mesa relacionados a seu balanceamento, tal efeito é desprezado, o que não impede que se obtenha uma boa aproximação para o modelo da mesa. No entanto, ele é presente no sistema e, em uma exposição de longa duração, sua existência se torna perceptível. Esse efeito pode ser verificado observando-se a redução das velocidades angulares da mesa e a redução da amplitude de suas oscilações.

Para se ter uma dimensão desse amortecimento, foram feitos dois testes. Em um deles a mesa foi posta pra oscilar por meio de uma perturbação inicial, feita manualmente. Os ângulos de arfagem e rolagem da mesa foram coletados durante um longo intervalo de tempo e esse registro é mostrado nas figuras 4.19 e 4.20. Outro teste foi o de verificar o amortecimento nas velocidades angulares. Nesse caso, a mesa foi posta para girar em torno do eixo de guinada e os dados de velocidade angular no eixo z do sistema fixo à mesa foram coletados por um longo período de tempo. O gráfico resultante é mostrado na Figura 4.21.

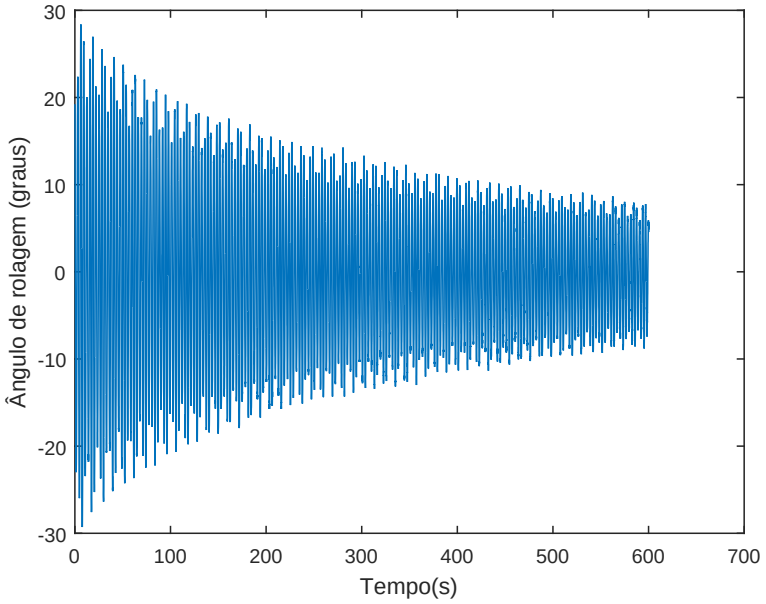


Figura 4.19: Amortecimento observado no ângulo de rolagem.

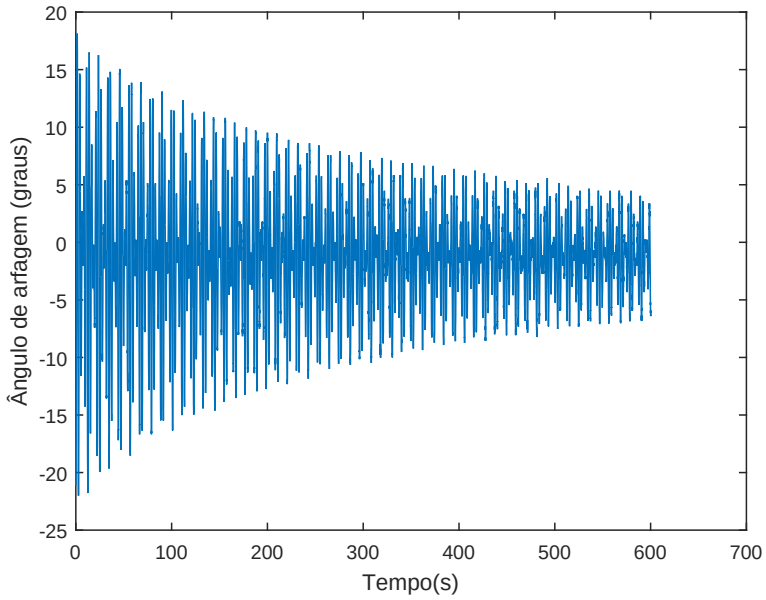


Figura 4.20: Amortecimento observado no ângulo de arfagem.

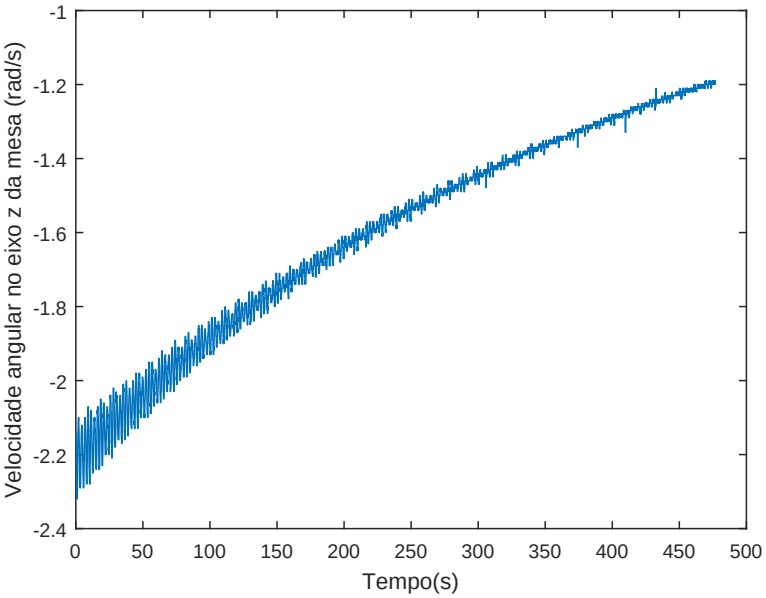


Figura 4.21: Diminuição da velocidade angular no eixo z do sistema de referência fixo à mesa.

As figuras 4.19 e 4.20 mostram que a amplitude de movimento da mesa nesses eixos decai levemente no decorrer do tempo. No caso da Figura 4.21, nota-se que a magnitude dessa velocidade diminui gradualmente.

Apesar de perceptível, o amortecimento é muito lento, já que, mesmo após um longo intervalo de tempo[5], a mesa ainda se movimentava consideravelmente.

[5]Os testes realizados para quantificar os amortecimentos do rolamento a ar duraram cerca de 10 minutos cada.

Capítulo 5

Conclusões

5.1 Considerações Finais

Como indicado nos capítulos 1 e 3, almejava-se construir uma plataforma que permitisse simular, dada certa tolerância, condições de torque gravitacional reduzido de modo a viabilizar, futuramente, testes com microssatélites. Esse objetivo foi alcançado, como foi apresentado no Capítulo 4.

Para a solução da problemática apresentada, fez-se uso de uma plataforma baseada na utilização de um rolamento a ar. Como todo sistema real, esse sistema também apresenta imperfeições, sendo uma delas o amortecimento na movimentação devido à presença de torques de arrasto que, apesar de terem sido desconsiderados no decorrer da modelagem, são presentes. Além disso, uma limitação principal foi encontrada: a incapacidade, dado o modelo atual do protótipo, de realizar balanceamentos significativos no eixo de guinada da mesa. Isso, no entanto, não interferiu na validação do sistema de balanceamento e, portanto, o projeto desenvolvido neste trabalho mostrou-se adequado para os propósitos estabelecidos.

Por fim, cabe mencionar que o projeto desenvolvido neste trabalho foi aceito para apresentação oral no *2nd IAA Latin American CubeSat Workshop*, organizado pela IAA (*International Academy of Astronautics*), pela Universidade de Brasília e pela Universidade Federal de Santa Catarina, e que será realizado em Florianópolis (Brasil) entre os dias 28 de fevereiro de 2016 e 3 de março de 2016.

5.2 Perspectivas Futuras

5.2.1 Sensores de fim de curso

Como foi discutido neste trabalho, utilizam-se motores de passo para controlar as UMMs, as quais possuem limitações físicas em seu deslocamento. A distância máxima que uma UMM pode percorrer é de 140mm, considerando-se que ela parta de uma de suas extremidades.

Entretanto, não há, na estrutura montada, nenhum dispositivo capaz de mensurar a distância restante que pode ser percorrida antes de se atingir a limitação física. Isso poderia gerar um problema, pois, caso essa limitação fosse atingida, o motor teria sua rotação impedida, o que poderia acarretar em uma corrente elevada e consequente danificação do equipamento.

A fim de corrigir esse problema, sensores de fim de curso poderiam ser acoplados ao equipamento, os quais seriam utilizados para impedir que os motores atingissem a limitação descrita, preservando os dispositivos utilizados.

5.2.2 Filtro de Kalman

Como pode-se notar neste trabalho, o algoritmo de balanceamento utilizado é relativamente lento devido a utilização do método dos mínimos quadrados, o qual exige muitas medições para gerar uma estimativa acurada do vetor de desbalanceamento, "r". O Filtro de Kalman poderia ser utilizado como uma alternativa a esse método. Com esse intuito, implementar-se-ia esse algoritmo para estimar "r" a cada medição dos sensores. Nota-se que, a princípio, essas predições seriam inacuradas, contudo estas se tornariam mais acuradas à medida que mais dados fossem coletados dos sensores.

5.2.3 Calibração precisa da UMI

Como mostrado neste trabalho, os sensores utilizados apresentam descalibração a qual foi parcialmente corrigida por meio de uma reta de calibração. Contudo, existem outros métodos mais acurados, porém mais complexos, que podem ser implementados para obtenção de resultados mais precisos.

Além disso, uma fonte de erro no algoritmo de desbalanceamento é a posição da UMI com relação à mesa. Idealmente esse sensor deve estar com os eixos de arfagem e rolagem alinhados aos respectivos eixos da mesa, para que suas medições reflitam fielmente as orientações da mesa. Por isso, um trabalho futuro remete ao desenvolvimento de um método de fixação e calibração precisa dos sensores.

5.2.4 Gaiola de Helmholtz

Uma adição ao projeto que tornaria as condições de teste mais semelhantes às condições espaciais seria a construção de uma gaiola metálica capaz de gerar um campo magnético similar ao encontrado na órbita terrestre. Essa gaiola já foi projetada, contudo, até o momento da redação deste trabalho, ainda não havia sido construída.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CAPPELLETTI, C. Project and manufacturing of a test bed for microsatellites attitude determination and control. 2008. Dissertação - Escola de Engenharia Aeroespacial - Universidade de Roma La Sapienza, Roma.

[2] OLSEN, T. A. Design of an adaptive balancing scheme for the small satellite attitude control simulator (SSACS). 1995. Dissertação - Departamento de Engenharia Mecânica e Aeroespacial, Universidade do Estado de Utah, Estados Unidos.

[3] YOUNG, J. S. Development of an automatic balancing system for a small satellite atitude control simulator. 1998. 153 páginas. Mestrado de Ciências em Engenharia Mecânica. Universidade do Estado de Utah. Logan, Utah.

[4] WALKER, Jearl;DAVID, Halliday; RESNICK, Robert **Fundamentos de Física, Volume 2**: gravitação, ondas e termodinâmica. 8ed. Rio de Janeiro: LTC Editora, 2009. 291p

[5] HIBBELER, Russel. **Dinâmica**: Mecânica para Engenharia. 10ed. Rio de Janeiro: LTC Editora, 2005.

[6] GREENWOOD, Donald T. Greenwood. **Principles of Dynamics**:Prentice-Hall international series in dynamics. 3ed. Michigan:Prentice-Hall.

[7] MARK, Spong. **Robot modeling and control**. 2ed. New York: John Wiley e Sons, 2006.

[8] BATE, R. Roger; MUELLER, D. Donald; WHITE, E. Jerry. **Fundamentals of Astrodynamics**:Dover books on Aeronautical Engineering. 1ed. Dover:Dover Publications, 1971.

[9] GHAOUI, L. El. **Hyper-Textbook**: Optimization Models and Applications. 1ed. California: EECS Department - UC Berkeley, 2014.
Disponível em: https://inst.eecs.berkeley.edu/ẽe127a/book/login/index.html Acesso em: 20/09/2015.

[10] Repositório GitHub com códigos de teste do IMU. Disponível em: <https://github.com/adafruit/Adafruit_LSM303DLHC>. Acesso em: 30 de Junho de 2015.

[11] Relatório de Estação Geodésica.
Disponível em: http://www.bdg.ibge.gov.br/bdg/pdf/relatorio.asp?L1=8071435.
Acesso em: 30 de Junho de 2015.

[12] Conectando o Arduino Uno com o Matlab.
Disponível em http://www.matlabarduino.org/serial-communication.html. Acesso em 30 de Junho de 2015.

[13] Repositório da Biblioteca Adafruit 9DOF.
Disponível em: https://github.com/adafruit/Adafruit_9DOF.
Acesso em: 30 de Junho de 2015

[14] Algoritmos de Calibração do IMU.
Disponível em: https://learn.adafruit.com/lsm303-accelerometer-slash-compass-breakout/calibration.
Acesso em: 2 de Junho de 2015.

[15] *Datasheet* do módulo de comunicação *wireless* XBee.
Disponível em: <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>
Acesso em: 30/06/2015.

[16] Tutorial de configuração do módulo de comunicação *wireless* XBee.
Disponível em: <https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu>
Acesso em: 30/06/2015.

[17] Guia de instalação do driver FTDI.
Disponível em: <https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers/windows-quick-and-easy>
Acesso em: 30/06/2015.

[18] *Datasheet* do rolamento a ar (*Air Bearing*).
Disponível em: <http://www.pi-usa.us/products/Air_Bearing_Stages/PIglide/(...)
(...)_HB_Spherical_Air_Bearing.pdf>
Acesso em: 14/11/2015.

[19] Coeficiente de atrito alumínio com anodização dura.
Disponível em: <http://aluminumsurface.blogspot.com.br/2011/02/coefficient-of-friction-between.html>
Acesso em: 14/11/2015.

[20] Coeficiente de atrito entre alumínio com anodização dura e o ar.
Disponível em: <http://www.nelsonair.com/NA_primer.htm>
Acesso em: 14/11/2015.

[21] Filtro extrator de óleo.
Disponível em: <http://catalog.jamiesonequipment.com/Asset(...)
(...)/RTi_i014_Oil_Extractor_Combo.pdf>
Acesso em: 14/11/2015.

[22] *Datasheet* mesa coordenada.
Disponível em: <https://www.proxtools.com/store/pc/catalog/manuals/(...)
(...)compound%20table%20kt70.pdf>
Acesso em: 14/11/2015.

[23] Informações técnicas compressor.
Disponível em: <http://ccompressor.com/equipment/compressors/puska/comba/comba-2100-r/comba-2100-r-ii/>
Acesso em: 14/11/2015.

[24] Informações sobre concentração de impurezas em um compressor.
Disponível em: <http://www.bosch.com.br/br/ferramentas_pneumaticas/(...)
(...)produtos/downloads/ManualPneumatica_ARComprimido.pdf>
Acesso em: 23/11/2015 às 23h31.

[25] Comentários sobre o driver StepStick.
Disponível em: <http://reprap.org/wiki/StepStick>
Acesso em: 16/11/2015 às 22h09.

[26] Datasheet do CI A4988.
Disponível em: <http://www.allegromicro.com/en/Products/Motor-Driver-And-Interface-ICs/Bipolar-Stepper-Motor-Drivers/A4988.aspx>
Acesso em: 18/11/2015 às 01h21.

[27] Software de simulação AHRS da Adafruit.
Disponível em: <https://learn.adafruit.com/ahrs-for-adafruits-9-dof-10-dof-breakout/software>
Acesso em: 17/11/2015 às 09h01.

[28] Datasheet do LSM303DLHS.
Disponível em: <http://www.adafruit.com/datasheets/LSM303DLHC.PDF>
Acesso em: 18/11/2015 às 09h27.

[29] Datasheet do L3GD20.
Disponível em: <http://www.adafruit.com/datasheets/L3GD20.pdf>
Acesso em: 18/11/2015 às 09h30.

[30] Artigo sobre a obtenção da atitute de um sistema através das matrizes de rotação.
Disponível em: <http://cache.freescale.com/files/sensors/doc/app_note/AN4248.pdf>
Acesso em: 22/11/2015 às 15h27.

[31] Artigo sobre a história do Xbee.
Disponível em: <https://en.wikipedia.org/wiki/XBee>
Acesso em: 23/11/2015 às 09h26.

[32] Artigo sobre o protocolo Zigbee
Disponível em: <https://en.wikipedia.org/wiki/ZigBee>
Acesso em: 23/11/2015 às 09h46.

[33] Datasheet Xbee modelo xb24-aci-001
Disponível em: <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>
Acesso em: 23/11/2015 às 10h08.

[34] SCHWARTS, J. L.; Peck, M. A.; Hall, C. D. *Historical Review of Air-Bearing Spacecraft Simulators*. Journal of Guidance, Control and Dynamics. Vol. 26. Número 4. Julho-Agosto de 2003.

[35] BENTLEY, John P. **Principals of Measurement Systems**. 1ed. Inglaterra: Pearson - Prentice Hall, 2005. 521p.

[36] Distribuição do campo magnético da Terra.
Disponível em: <http://web.ua.es/docivis/magnet/earths_magnetic_field2.html>
Acesso em: 06/12/2015 às 21h29.

[37] Explicação sobre o fenômeno *Gimbal Lock*.
Disponível em: <http://run.usc.edu/cs520-s12/quaternions/quaternions-cs520.pdf>
Acesso em: 09/12/2015 às 11h07.

ANEXOS

Anexo I: Descrição do conteúdo do CD.
Anexo II: Códigos das implementações dos programas utilizados.

I. DESCRIÇÃO DO CONTEÚDO DO CD

O CD anexo a este trabalho contém:

1. O arquivo digitalizado deste texto em formato ".pdf";
2. Todos os códigos de implementação dos programas utilizados neste trabalho, da mesma forma como constam na seção de anexos;
3. Um arquivo ".pdf" com o resumo deste trabalho; e
4. Um arquivo ".pdf" com o *abstract* deste trabalho.

II. PROGRAMAS UTILIZADOS

Antes de prosseguir à leitura dos códigos, cabe uma observação. No decorrer das páginas deste anexo serão vistas reticências em dois formatos distintos: sem parênteses ("...") e com parênteses ("(...)"). As reticências sem parênteses fazem parte da sintaxe do MATLAB e, portanto, sua presença não interferirá na execução dos códigos. Por outro lado, as reticências com parênteses, vistas tanto nos códigos em MATLAB, quanto nos códigos utilizados no Arduino, impedirão a execução dos programas e foram utilizadas **somente** com o propósito de organizar a exibição dos códigos no decorrer deste anexo.

Tendo dito isso, caso o leitor queira executar os códigos deste anexo, deverá retirar todas as reticências com parênteses presentes no código em questão.

II.1 Código utilizado no Arduino

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM303_U.h>
#include <Adafruit_L3GD20_U.h>
#include <Adafruit_Simple_AHRS.h>

// Create sensor instances.
Adafruit_LSM303_Accel_Unified accel(30301);
Adafruit_LSM303_Mag_Unified   mag(30302);

/* Assign a unique ID to this sensor at the same time */
Adafruit_L3GD20_Unified gyro = Adafruit_L3GD20_Unified(20);

// Create simple AHRS algorithm using the above sensors.
Adafruit_Simple_AHRS          ahrs(&accel, &mag);

//Initialise the Xbee
#include<SoftwareSerial.h>
SoftwareSerial Xbee(2,3);

// Pinos conectados ao Step e Dir do Easydriver
//M1
int pino_enable1 = 4;
int pino_passo1 = 5;
int pino_direcao1 = 6;
```

```
//M2
int pino_enable2 = 7;
int pino_passo2 = 8;
int pino_direcao2 = 9;

//M3
int pino_enable3 = 10;
int pino_passo3 = 11;
int pino_direcao3 = 12;

//Global Variables
//int time = 40;
char Mode = 'Z';
//int i = 0;

//Variáveis dos motores
int direcao1 = 0;
int direcao2 = 0;
int direcao3 = 0;
int passos_motor1 = 0;
int passos_motor2 = 0;
int passos_motor3 = 0;

void direcao(int direcao, int pino_direcao, int pino_enable){
  if (direcao == 1)
        {

          Serial.println(direcao);
          digitalWrite(pino_direcao, 1);
          digitalWrite(pino_enable, LOW);

        }
        if (direcao == 2)
        {
            Serial.println(direcao);
            digitalWrite(pino_direcao, 0);
            digitalWrite(pino_enable, LOW);
        }

}
```

```
//Função utilizada para fazer o Arduino aguardar ate que haja algum dado(...)
(...) disponível na porta serial
void wait_serial()
{
  while(Xbee.available()==0)
  {
  }
}

//Função que limpa o buffer de entrada do Xbee
void clear_buffer_xbee()
{
  while(Xbee.available())
  {
    char getData = Xbee.read();
  }
}

void displaySensorDetails(void)
{
  sensor_t sensor;
  gyro.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value);
  Serial.println(" rad/s");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value);
  Serial.println(" rad/s");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution);
  Serial.println(" rad/s");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

/*void   ResetBufferXBee(void)
{
  Xbeedevice.println(string);
  Xbee_device.println(string);
  Xbee_device.println(string);
```

```
}*/

void parada(int pino_passo, int pino_enable)
{
  digitalWrite(pino_enable, HIGH);
  digitalWrite(pino_passo, 0);
  delay(1);
}

void move_motor(int pino_passo)
{
  digitalWrite(pino_passo, 1);
  delay(1);
  digitalWrite(pino_passo, 0);
  delay(1);
}

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Gyroscope Test"); Serial.println("");

  /* Enable auto-ranging */
  gyro.enableAutoRange(true);

  /* Initialise the sensor */
  if(!gyro.begin())
  {
    /* There was a problem detecting the L3GD20 ... check your connections */
    Serial.println("Ooops, no L3GD20 detected ... Check your wiring!");
    while(1);
  }

  /* Display some basic information on this sensor */
  displaySensorDetails();

  // Initialize the sensors.
  accel.begin();
  mag.begin();

  //Xbee
  Xbee.begin(9600);
```

```
  clear_buffer_xbee();
  Xbee.flush();

  // Define os pinos como saida

  //M1

  pinMode(pino_passo1, OUTPUT);
  pinMode(pino_direcao1, OUTPUT);
  pinMode(pino_enable1, OUTPUT);

  //M2

  pinMode(pino_passo2, OUTPUT);
  pinMode(pino_direcao2, OUTPUT);
  pinMode(pino_enable2, OUTPUT);

  //M3

  pinMode(pino_passo3, OUTPUT);
  pinMode(pino_direcao3, OUTPUT);
  pinMode(pino_enable3, OUTPUT);
}

void loop(void)
{
  //Variavel de Modo de execucao
  char Mode = 'Z';
  //Variáveis dos motores
  int verify = 0; //Variavel de verificacao para o comando do motor
  int direcao1 = 0;
  int direcao2 = 0;
  int direcao3 = 0;
  int passos_motor1 = 0;
  int passos_motor2 = 0;
  int passos_motor3 = 0;

  Mode = Xbee.read();
  if(Mode==-1)
  {
    return;
  }
```

```
switch(Mode)
{
  case 'R':
  {
    sensors_vec_t   orientation;
    String string;
    // Use the simple AHRS function to get the current orientation.
    if (ahrs.getOrientation(&orientation))
    {
      /* 'orientation' should have valid .roll and .pitch fields */
      string=String(orientation.roll); string=string+" ";
      string=string+orientation.pitch; string=string+" ";
      string=string+orientation.heading; string=string+" ";
    }

    /* Get a new sensor event */

    sensors_event_t event;
    gyro.getEvent(&event);

    /* Display the results (speed is measured in rad/s)*/

    string=string+event.gyro.x; string=string+" ";
    string=string+event.gyro.y; string=string+" ";
    string=string+event.gyro.z;
    string+=" 1";

    Xbee.println(string);
    break;
  }

  case 'M':
  {
    Serial.println(Mode);
    delay(100);
   // Wait for data in the serial port
   // wait_serial();
   // if (Xbee.available() > 0){
      direcao1 = Xbee.parseInt();
      direcao2 = Xbee.parseInt();
      direcao3 = Xbee.parseInt();
```

```
  passos_motor1 = Xbee.parseInt();
  passos_motor2 = Xbee.parseInt();
  passos_motor3 = Xbee.parseInt();
  verify = Xbee.parseInt();
// } Update: condicional if desativado.

Serial.println(direcao1);
Serial.println(direcao2);
Serial.println(direcao3);
Serial.println(passos_motor1);
Serial.println(passos_motor2);
Serial.println(passos_motor3);
Serial.println(verify);

if(verify!=1)
{
  Xbee.println('E');
  // Se o dado verificador estiver errado, retorna(...)
  (...) erro e reinicia o loop do arduino.
  return;
}

direcao(direcao1, pino_direcao1, pino_enable1);
direcao(direcao2, pino_direcao2, pino_enable2);
direcao(direcao3, pino_direcao3, pino_enable3);

for (int p=0 ; (p < passos_motor1)|| (p < passos_motor2)||(...)
(...)(p< passos_motor3); p++)
{
  if(p<passos_motor1)
  {
    move_motor(pino_passo1);
  }
  if(p<passos_motor2)
  {
    move_motor(pino_passo2);
  }
  if(p<passos_motor3)
  {
    move_motor(pino_passo3);
  }
}
```

```
      passos_motor1 = 0;
      passos_motor2 = 0;
      passos_motor3 = 0;
      parada(pino_passo1, pino_enable1);
      parada(pino_passo2, pino_enable3);
      parada(pino_passo3, pino_enable3);
      Xbee.println('D');
      Serial.println('D');
      clear_buffer_xbee();
    }
    case 'a':
    {
      Xbee.println('a');
      break;
    }
    default:
    {
      break;
    }
  }
}
```

II.2 Código utilizado no Matlab

II.2.1 Funções utilizadas no Matlab

setup_serial_xbee.m

```
function [s,erro] = setup_serial_xbee(comPort)

    erro = 1; %flag de erro
    rate = 9600; %Escolher a taxa de transmissão de dados - padrão Xbee 9600

    %Criação da porta serial e configuração de seus parâmetros
    s=serial(comPort);
    set(s,'DataBits',8);
    set(s,'StopBits',1);
    set(s,'BaudRate', rate);
    set(s,'Parity','none');
    fopen(s);
```

```
    %Teste de comunicação
    %um caracter 'a' é enviado ao arduino e espera-se que um caracter 'a'
    %seja recebido de volta
    fprintf(s,'%c','a');
    a='b';
    while(a~='a')
        a=fscanf(s,'%s');
    end
    if(a=='a')
        disp('serial read')
        mbox = msgbox('Serial Communication setup'); uiwait(mbox);
        erro = 0; %operação bem sucedida, flag de erro falsa.
    end
end
```

data_acquisition.m

```
function [event_gyro_x,event_gyro_y,event_gyro_z,roll, pitch,(...)
(...) yaw, intervals] = data_acquisition(s, iterations)


% Essa função é utilizada para aquisição de dados do IMU através da
% comunicação sem fio com o Arduino

roll = zeros(iterations,1, 'double');
pitch = zeros(iterations,1, 'double');
yaw = zeros(iterations,1, 'double');
event_gyro_x = zeros(iterations,1, 'double');
event_gyro_y = zeros(iterations,1, 'double');
event_gyro_z = zeros(iterations,1, 'double');

intervals=zeros(iterations,1, 'double');
time = zeros(iterations,1, 'double');

i=1;
packets_lost=0;


    % Limpa o buffer do XBee lendo tudo o que está contido nele.
    if(s.BytesAvailable>0)
```

```
    display('Limpando dados do buffer do XBee.');
    display('Dados encontrados:');
    while(s.BytesAvailable>0)
        text_line=fgetl(s);
        display(text_line);
    end
    display('Buffer resetado com sucesso.');
end

tic;
while i<=iterations
    fprintf(s, '%c','R');%Envia-se um caracter 'R' para o arduino,
    %requisitando uma leitura de dados
    pause(0.02); %Espera 20ms aproximadamente (minimo que o MATLAB suporta)
    if(s.BytesAvailable==0) %Se não houve resposta ao requerimento,(...)
    (...) outra tentativa é feita
        %display('Data loss');
        packets_lost=packets_lost+1;
        continue;
    end
    intervals(i)=toc;
    tic;
    text_line=fgetl(s);
    ParsedLine=strread(text_line,'%s','delimiter',' ');
    if (size(ParsedLine,1)~=7)
        display('Dados recebidos em formato inválido(...)
        (...) (quantidade de informações diferente de 7)');
        continue; % Ignora o resto da iteração do laço e (...)
        (...)passa para a iteração seguinte
    elseif (strcmp(ParsedLine(7),'1')~=1)
        display('Dados recebidos em formato inválido(...)
        (...)(Dado de verificação inválido)');
        continue; % Ignora o resto da iteração do laço e (...)
        (...) passa para a iteração seguinte
    end
    TextData(i,:)=ParsedLine;
    event_gyro_x(i) = str2double(TextData(i,4));
    event_gyro_y(i) = str2double(TextData(i,5));
    event_gyro_z(i) = str2double(TextData(i,6));
    roll(i) = str2double(TextData(i,1));
    pitch(i) = str2double(TextData(i,2));
    yaw(i) = str2double(TextData(i,3));
```

```
time(i)=sum(intervals);

%Geração de gráficos em tempo real da atitude do sistema
if i==1
    handle=figure;
    subplot(3,1,1);
    roll_graph=plot(time,roll);
    axis([0 10 -45 45]);
    xlabel('Tempo(s)');
    ylabel('Roll');
    current_roll = text(10, 55,(...)
    (...) sprintf('Roll = %.2f',roll(i)),(...)
    (...)'horizontalAlignment','right');

    subplot(3,1,2);
    pitch_graph=plot(time,pitch);
    axis([0 10 -45 45]);
    xlabel('Tempo(s)');
    ylabel('Pitch');
    current_pitch = text(10,55,(...)
    (...)sprintf('Pitch = %.2f',pitch(i)),(...)
    (...)'horizontalAlignment','right');

    subplot(3,1,3);
    yaw_graph=plot(time,yaw);
    axis([0 10 -180 180]);
    xlabel('Tempo(s)');
    ylabel('Yaw');
    current_yaw = text(10,220, (...)
    (...)sprintf('Yaw = %.2f',yaw(i)),'horizontalAlignment','right');
    current_iteration = text(10,-380, (...)
    (...)sprintf('Current Iteration = %d',i),'horizontalAlignment',(...)
    (...)'right');
    current_packets_lost = text(10,-440, sprintf('Packets Lost =(...)
    (...) %d',packets_lost),'horizontalAlignment','right');

else
    subplot(3,1,1);
    set(roll_graph, 'XData', time(1:i));
    set(roll_graph, 'YData', roll(1:i));
    if time(i)>10
        set(current_roll,'string', (sprintf('Current Roll = (...)
```

```
(...)%.2f',roll(i))),'Position', [time(i), 55, 0]);
    axis([time(i)-10 time(i) -45 45]);
else
    set(current_roll,'string', (sprintf('Current Roll = (...)
(...)%.2f',roll(i))),'Position', [10, 55, 0]);
    axis([0 10 -45 45]);
end

subplot(3,1,2);
set(pitch_graph, 'XData', time(1:i));
set(pitch_graph, 'YData', pitch(1:i));
if time(i)>10
    set(current_pitch,'string', (...)
(...)(sprintf('Current Pitch = %.2f',pitch(i))),(...)
(...) 'Position', [time(i), 55, 0]);
    axis([time(i)-10 time(i) -45 45]);
else
    set(current_pitch,'string', (...)
(...)(sprintf('Current Pitch = %.2f',pitch(i))),(...)
(...) 'Position', [10, 55, 0]);
    axis([0 10 -45 45]);
end

subplot(3,1,3);
set(yaw_graph, 'XData', time(1:i));
set(yaw_graph, 'YData', yaw(1:i));
if time(i)>10
    set(current_yaw,'string', (...)
(...)(sprintf('Current Yaw = %.2f',yaw(i))),(...)
(...) 'Position', [time(i), 220, 0]);
    set(current_iteration, 'string', (...)
(...)(sprintf('Current Iteration = %d',i)),(...)
(...) 'Position', [time(i), -380, 0]);
    set(current_packets_lost, 'string', (sprintf('Packets Lost =(...)
(...) %d',packets_lost)), 'Position', [time(i), -440, 0]);
    axis([time(i)-10 time(i) -180 180]);
else
    set(current_yaw,'string', (...)
(...)(sprintf('Current Yaw = %.2f',yaw(i))),(...)
(...) 'Position', [10, 220, 0]);
    set(current_iteration, 'string', (...)
(...)(sprintf('Current Iteration = %d',i)),(...)
```

```
            (...) 'Position', [10, -380, 0]);
                set(current_packets_lost, 'string', (...)
            (...)(sprintf('Packets Lost = %d',packets_lost)),(...)
            (...) 'Position', [10, -440, 0]);
                axis([0 10 -180 180]);
            end
        end

        i=i+1;
    end
    close all;
end
```

close_ports.m

```
function [] = close_ports()

% Esta função fecha todas as portas seriais ativas e deleta suas
% respectivas instâncias

clc
clear all
if ~isempty(instrfind)
    fclose(instrfind);
    delete(instrfind);
end
close all
clc
disp('Serial Port Closed');

end
```

calibration_IMU.m

```
function [roll_K, roll_a,pitch_K, pitch_a, yaw_K, yaw_a, (...)
            (...)omegax_offset, omegay_offset, omegaz_offset]=calibration_IMU(s)

    clc
    iterations1=input('Passo 1: Calibração das orientações.\n O IMU deve(...)
            (...) atingir o máximo e o mínimo de todas as orientações(...)
            (...) possíveis durante a calibração.\n Insira o número de(...)
            (...) iterações.');
```

```
%[event_gyro_x,event_gyro_y,event_gyro_z,roll, pitch, yaw, intervals](...)
        (...) = data_acquisition(s,iterations1);
[~,~,~,roll, pitch, yaw, ~] = data_acquisition(s,iterations1);

roll_min=min(roll);
roll_max=max(roll);
pitch_min=min(pitch);
pitch_max=max(pitch);
yaw_min=min(yaw);
yaw_max=max(yaw);


iterations2=input('Passo 2: Calibração das velocidades angulares.\n(...)
        (...) O IMU deve permanecer PARADO.\n Insira o número de(...)
        (...) iterações.');
[event_gyro_x,event_gyro_y,event_gyro_z,~, ~, ~, ~] =(...)
        (...) data_acquisition(s, iterations2);

omegax_offset=mean(event_gyro_x);
omegay_offset=mean(event_gyro_y);
omegaz_offset=mean(event_gyro_z);

roll_K=(roll_max-roll_min)/(180-(-180));
pitch_K=(pitch_max-pitch_min)/(90-(-90));
yaw_K=(yaw_max-yaw_min)/(180-(-180));

roll_a=roll_min-roll_K*(-180);
pitch_a=pitch_min-pitch_K*(-90);
yaw_a=yaw_min-yaw_K*(-180);

%Os dados de calibração são gravados em arquivo para uso posterior.
fid = fopen('IMUCalibrationData.txt','w');
calibration_data=num2str(roll_K);
calibration_data=strcat(calibration_data, ',');
calibration_data=strcat(calibration_data, num2str(roll_a));
calibration_data=strcat(calibration_data, ',');
calibration_data=strcat(calibration_data, num2str(pitch_K));
calibration_data=strcat(calibration_data, ',');
calibration_data=strcat(calibration_data, num2str(pitch_a));
calibration_data=strcat(calibration_data, ',');
calibration_data=strcat(calibration_data, num2str(yaw_K));
calibration_data=strcat(calibration_data, ',');
```

```
    calibration_data=strcat(calibration_data, num2str(yaw_a));
    calibration_data=strcat(calibration_data, ',');
    calibration_data=strcat(calibration_data, num2str(omegax_offset));
    calibration_data=strcat(calibration_data, ',');
    calibration_data=strcat(calibration_data, num2str(omegay_offset));
    calibration_data=strcat(calibration_data, ',');
    calibration_data=strcat(calibration_data, num2str(omegaz_offset));
    fwrite(fid, calibration_data);
    fclose(fid);

    fclose(s);
    clear s;
    close all;
end
```

calibration_data_read.m

```
function [calibration_data, error] = calibration_data_read()

%Esta função lê dados armazenados em arquivo e retorna os dados de calibração
%correspondentes.

    error=0;
    calibration_data=zeros(9, 1, 'double');

    fid = fopen('IMUCalibrationData.txt','r+');

    if fid==-1
        error=1;
        return;
    end

    text_line=fgetl(fid);
    ParsedLine=strread(text_line,'%s','delimiter',',');
    if (size(ParsedLine,1)~=9)
        display('Dados de calibração insuficientes.');
        error=1;
        return;
    end

    roll_K = str2double(ParsedLine(1));
    roll_a = str2double(ParsedLine(2));
```

```
    pitch_K = str2double(ParsedLine(3));
    pitch_a = str2double(ParsedLine(4));
    yaw_K = str2double(ParsedLine(5));
    yaw_a = str2double(ParsedLine(6));
    omegax_offset = str2double(ParsedLine(7));
    omegay_offset = str2double(ParsedLine(8));
    omegaz_offset = str2double(ParsedLine(9));

    calibration_data=[roll_K, roll_a,pitch_K, pitch_a, yaw_K, yaw_a,(...)
            (...) omegax_offset, omegay_offset, omegaz_offset];
end
```

phi_estimation.m

```
function [Phi] = phi_estimation(m, g, Ixx, Iyy, Izz,(...)
            (...)roll, pitch, roll_anterior,pitch_anterior, Phi, time)

%Esta função calcula a matrix anti-simétrica Phi, utilizada para calcular o
%vetor de desbalanceamento

    P12 = -m*g*time*(cosd(roll)*cosd(pitch) + cosd(roll_anterior)*(...)
            (...)cosd(pitch_anterior))/(2*Ixx);
    P13 = m*g*time*(sind(roll)*cosd(pitch) + sind(roll_anterior)*(...)
            (...)cosd(pitch_anterior))/(2*Ixx);
    P21 = m*g*time*(cosd(roll)*cosd(pitch) + cosd(roll_anterior)*(...)
            (...)cosd(pitch_anterior))/(2*Iyy);
    P23 = m*g*time*(sind(pitch) + sind(pitch_anterior))/(2*Iyy);
    P31 = -m*g*time*(sind(roll)*cosd(pitch) + sind(roll_anterior)*(...)
            (...)cosd(pitch_anterior))/(2*Izz);
    P32 = -m*g*time*(sind(pitch) + sind(pitch_anterior))/(2*Izz);

    Phi = [Phi; 0 P12 P13;P21 0 P23;P31 P32 0];

end
```

unbalance_vector_estimation.m

```
function [r] = unbalance_vector_estimation(event_gyro_x, event_gyro_y,(...)
            (...) event_gyro_z, roll_reading, pitch_reading, intervals,(...)
            (...) iterations)
%Esta função retorna o vetor r de desbalanceamento entre o centro de massa
%e o centro de rotação a partir dos dados dos sensores

m = 10.8; % Massa total do sistema
g = 9.78; % Aceleração da gravidade local

Ixx = 0.21; % Momento principal de inércia do eixo x
Iyy = 0.121; % Momento principal de inércia do eixo y
Izz = 0.28; % Momento principal de inércia do eixo z

Phi = [];
Delta_w = [];

for i=2:1:iterations

  roll = roll_reading(i);
  pitch = pitch_reading(i);
  Phi = phi_estimation(m, g, Ixx, Iyy, Izz,roll, pitch,(...)
            (...) roll_reading(i-1),pitch_reading(i-1), Phi, intervals(i-1));

  delta_w_x = event_gyro_x(i) - event_gyro_x(i-1);
  delta_w_y = event_gyro_y(i) - event_gyro_y(i-1);
  delta_w_z = event_gyro_z(i) - event_gyro_z(i-1);

  Delta_w = [Delta_w; delta_w_x;delta_w_y;delta_w_z];
end

r = (inv(Phi'*Phi))*Phi'*Delta_w;
end
```

motors_actuation.m

```
function [] = motors_actuation(s,direction1,direction2,direction3,(...)
            (...)distance1,distance2,distance3)
% Esta função é utilizada para a atuação dos três motores de passo
% presentes no sistema.
    %Constantes
    distance_p_revolution = 0.001; %Distancia por revolução em metros.
    steps_p_revolution = 200; % Número de passos necessários para uma volta

    %Motor do eixo X
    revolutions1 = distance1/distance_p_revolution;
    number_steps1 = revolutions1*steps_p_revolution;
    number_steps1=round(number_steps1);
    number_steps1=abs(number_steps1)
    direction1
    if number_steps1>5000
        number_steps1=5000;
    end

    %Motor do eixo Y
    revolutions2 = distance2/distance_p_revolution;
    number_steps2 = revolutions2*steps_p_revolution;
    number_steps2=round(number_steps2);
    number_steps2=abs(number_steps2)
    direction2
    if number_steps2>5000
        number_steps2=5000;
    end

    %Motor do eixo Z
    revolutions3 = distance3/distance_p_revolution;
    number_steps3 = revolutions3*steps_p_revolution;
    number_steps3=round(number_steps3);
    number_steps3=abs(number_steps3)
    direction3
    if number_steps3>5000
        number_steps3=5000;
    end

    %Construção do comando de atuação a ser enviado para o arduino
    comando=num2str(direction1);
```

```
    comando=strcat(comando, ',');
    comando=strcat(comando, num2str(direction2));
    comando=strcat(comando, ',');
    comando=strcat(comando, num2str(direction3));
    comando=strcat(comando, ',');
    comando=strcat(comando, num2str(number_steps1));
    comando=strcat(comando, ',');
    comando=strcat(comando, num2str(number_steps2));
    comando=strcat(comando, ',');
    comando=strcat(comando, num2str(number_steps3));
    comando=strcat(comando, ',1');

    %Envia comando dos motores por meio da porta serial do Xbee
    while(1)
        verify='Z';
        fprintf(s, '%s', 'M');
        pause(0.03);
        fprintf(s, '%s', comando);

        while s.BytesAvailable==0
        end
        verify=fscanf(s, '%s');
        if verify == 'E'
            display('Erro na comunicação dos parâmetros de atuação(...
            (...) dos motores.');
        end
        if verify == 'D'
            break
        end
    end
end
```

number2direction.m

```
function [direction] = number2direction(number)
% Esta função traduz o número recebido em uma direção na qual a atuação
% deve ser realizada
    if number<0
        direction='1'; %Direção positiva - r < 0
    else
        direction='2'; %Direção negativa - r > 0
    end
end
```

balancing_algorithm.m

```
function [] = balancing_algorithm(iterations, serialport, tolerance, gain_vector)
%Função utilizada para executar o algoritmo de balanceamento da mesa

    %Fase de setup do hardware
    display('Fechando todas as portas seriais ativas . . .');
    close_ports();

    display('Inicializando porta serial do XBee . . .');
    s=setup_serial_xbee(serialport);

    clc
    display('Verificando dados de calibração dos sensores . . .');
    pause(1);
    [calibration_data, error]=calibration_data_read();
    if error==1
        display('Dados de calibração não encontrados.');
        display('Faça a calibração dos sensores.')
        return;
    end
    roll_K = calibration_data(1);
    roll_a = calibration_data(2);
    pitch_K = calibration_data(3);
    pitch_a = calibration_data(4);
    yaw_K = calibration_data(5);
    yaw_a = calibration_data(6);
    omegax_offset = calibration_data(7);
    omegay_offset = calibration_data(8);
    omegaz_offset = calibration_data(9);

    %Execução do algoritmo de balanceamento
    % display('Executando algoritmo de balanceamento em 5 . . .');
    % pause(1);
    % clc
    % display('Executando algoritmo de balanceamento em 4 . . .');
    % pause(1);
    % clc
    % display('Executando algoritmo de balanceamento em 3 . . .');
    % pause(1);
    % clc
    % display('Executando algoritmo de balanceamento em 2 . . .');
```

```
% pause(1);
% clc
display('Executando algoritmo de balanceamento em 1 . . .');
pause(1);
clc


display('Algoritmo de balanceamento iniciado.');
r=[inf;inf;inf];

%Considerou-se a mesma tolerância para todos os eixos
while(norm(r(1))>tolerance(1)||norm(r(2))>tolerance(2)||(...)
        (...)norm(r(3))>tolerance(3)||r(3)>0)
    %Fase de aquisição dos dados
    [event_gyro_x,event_gyro_y,event_gyro_z,roll, pitch, ~, intervals] = ...
        data_acquisition(s, iterations);

    %Calibração dos dados recebidos
    event_gyro_x = event_gyro_x-omegax_offset;
    event_gyro_y = event_gyro_y-omegay_offset;
    event_gyro_z = event_gyro_z-omegaz_offset;
    roll = roll*roll_K + roll_a;
    pitch = pitch*pitch_K + pitch_a;

    %Fase de determinação do vetor de desbalanceamento
    [r] = ...
        unbalance_vector_estimation(event_gyro_x, event_gyro_y, (...)
        (...)event_gyro_z, roll, pitch,...
        intervals, iterations);
    display('Vetor de desbalanceamento calculado. [rx; ry; rz]= ');
    display(r);

    %ZONA DE TESTES
    file=fopen('balanceamento_periodopendulo.csv', 'at');
    new_r=strcat(num2str(r(1)),',');
    new_r=strcat(new_r,strcat(num2str(r(2)),','));
    new_r=strcat(new_r,strcat(num2str(r(3)),','));
    new_r=strcat(new_r,strcat(num2str(mean(roll)),','));
    new_r=strcat(new_r,num2str(mean(pitch)));
    fprintf(file, '%s\n', new_r);
    fclose(file);
```

```
        %Fase de atuação dos motores
        %motors_actuation(s,number2direction(r(1)),number2direction(r(2)),...
        %    number2direction(r(3)), r(1)*gain_vector(1),r(2)*gain_vector(2)...
        %    ,r(3)*gain_vector(3))

        if (norm(r(1))<=tolerance(1) && norm(r(2))<=tolerance(2) &&(...)
            (...) (norm(r(3))<=tolerance(3)) && r(3) <= 0)
            display('Tolerância de desbalanceamento atingida.');
            display('Algoritmo concluído.');
            return;
        end
    end
end
```

II.2.2 Rotinas utilizadas no Matlab

script_calibration.m

```
%Script utilizado para a calibração do IMU

clc
clear all

close_ports();
%comPort='COM7'; %Rodrigo
comPort = '/dev/tty.usbserial-AH01D817'; %Ulisses
s = setup_serial_xbee(comPort);
[roll_K, roll_a,pitch_K, pitch_a, yaw_K, yaw_a, omegax_offset,(...)
            (...) omegay_offset, omegaz_offset]=calibration_IMU(s);
```

script_TG.m

```
clc
clear all

%Escolha da porta de comunicacao com o XBee. Varia de computador para computador.
comPort='COM7'; %Rodrigo
%comPort = '/dev/tty.usbserial-AH01D817'; %Ulisses
M = 10.8; %Massa do equipamento completo
m = 0.78; %Massa de uma "Massa móvel" medida
k = M/m;
gain=[k k k];
%tolerance=0.001;
tolerance_x=0.0001; %temporario
tolerance_y=0.0001; %temporario
tolerance_z=0.001; %temporario
iterations=500;

balancing_algorithm(iterations, comPort, tolerance, gain);
```