Universidade Federal de Juiz de Fora

Faculdade de Engenharia

Engenharia Elétrica - Habilitação em Robótica e Automação Industrial

Guilherme Marins Maciel

Compensação de Posicionamento Angular de um Manipulador Robótico Sobre uma Base com Três Graus de Liberdade

Juiz de Fora

2016

Guilherme Marins Maciel

Compensação de Posicionamento Angular de um Manipulador Robótico Sobre uma Base com Três Graus de Liberdade

Trabalho apresentado à disciplina de Trabalho de Conclusão de Curso do curso de graduação de Engenharia Elétrica, habilitação em Robótica e Automação Industrial, da Universidade Federal de Juiz de Fora.

Orientador: Prof. Exuperry Barros Costa

Coorientador: Prof. Leonardo Rocha Olivi

Juiz de Fora

2016

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Maciel, Guilherme Marins.
Compensação de posicionamento angular de um manipulador robótico sobre uma base com três graus de liberdade / Guilherme Marins Maciel. -- 2016.
83 f.

Orientador: Exuperry Barros Costa
Coorientador: Leonardo Rocha Olivi
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia, 2016.

1. Apontamento automático. 2. Manipulador robótico. 3. IMU. 4. Medição de orientação. 5. Filtro de Kalman. I. Costa, Exuperry Barros, orient. II. Olivi, Leonardo Rocha, coorient. III. Título.

Guilherme Marins Maciel

Compensação de Posicionamento Angular de um Manipulador Robótico Sobre uma Base com Três Graus de Liberdade

Trabalho apresentado à disciplina de Trabalho de Conclusão de Curso do curso de graduação de Engenharia Elétrica, habilitação em Robótica e Automação Industrial, da Universidade Federal de Juiz de Fora.

BANCA EXAMINADORA

Prof. Exuperry Barros Costa - Orientador
Universidade Federal de Juiz de Fora

Prof. Leonardo Rocha Olivi - Coorientador
Universidade Federal de Juiz de Fora

Prof. Lucas Corrêa Netto Machado
Universidade Federal de Juiz de Fora

RESUMO

O presente trabalho apresenta a fundamentação teórica e prática para a elaboração de um protótipo de um manipulador robótico antropomórfico, capaz de compensar rotações na base em 3 graus de liberdade no controle de orientação do *end-effector*, com o objetivo de apresentar um estudo introdutório sobre apontamento automático. Para a modelagem será apresentado a metodologia de cinemática direta proposta por Denavid-Hartenberg, obtendo-se a matriz de transformação homogênea que relaciona o *frame* inercial com o final. Para obtenção das variáveis de junta necessárias utilizou-se cinemática inversa diferencial, envolvendo análise jacobiana e otimização via gradiente descendente. Como a proposta envolve mudanças no *frame* inercial, será apresentado uma forma de identificar as rotações por meio da matriz de parametrização Euler ZYX, com os parâmetros obtidos de uma IMU(*INERTIAL MEASUREMENT UNIT*), utilizando leituras de acelerômetro, giroscópio e magnetômetro, fundidas e filtradas no filtro de Kalman. Para este experimento serão utilizados o Microcontrolador Arduino UNO, a placa IMU GY-87, e o software MATLAB.

Palavras-chave: Apontamento Automático. Manipulador Robótico. IMU. Medição de Orientação. Filtro de Kalman.

ABSTRACT

This paper presents a theoretical study for the development of a prototype of an anthropomorphic robotic manipulator, able to compensate rotations on the basis with 3 degrees of freedom for the end-effector orientation control, in order to present an introductory study on automatic appointment. For modeling will be presented the forward kinematics method proposed by Denavid-Hartenberg, getting the homogeneous transformation matrix that relates the inertial frame to the final frame. To obtain the necessary joint variables was used differential inverse kinematics, using jacobian analysis and gradient descent optimization technique. It will be presented a way to indentify the tilt through the Euler ZYX matrix with de parameters obtained by an IMU(INERTIAL MEASUREMENT UNIT) that contains accelerometer, gyroscope and magnetometer readings, the measures will be merged and filtered by Kalman Filter. For this prototype experiment will use the Microcontroller Arduino UNO, the IMU board GY-87, and MATLAB software.

Keywords: Robot Arm. IMU. Tilt Estimation. Kalman Filter

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Através dos séculos, a humanidade vem melhorando os processos de produção. A partir da metade do século XX foi concebida uma nova tecnologia para automatizar esses processos. Surgiu, então, o conceito de robô, uma máquina capaz de substituir os trabalhadores em suas tarefas. Qualquer máquina que opera com algum grau de autonomia, usualmente sobre controle computacional, pode ser chamada de robô [6].

Alguns dos fatores que mais impulsionaram a robótica na indústria foram o aumento do nível salarial da população (capacidade de consumo) e o desenvolvimento da microeletrônica, que barateou e melhorou a capacidade de processamento dos robôs. A vantagem desta evolução foi o aumento da produtividade e qualidade dos produtos. Em contrapartida, houve um decréscimo nos empregos de baixos salários na indústria [4].

De acordo com a IFR (*International Federation of Robots*), no ano de 2006 havia aproximadamente um milhão do robôs industriais no mundo, com destaque para países como o Japão, a Alemanha, os Estados Unidos e a Itália. O setor automotivo sempre foi o consumidor predominante de robôs industriais, mas atualmente a indústria química e eletrônica estão em destaque [3].

Uma categoria que se destaca proeminentemente na robótica são os manipuladores, que possuem uma base fixa e são constituídos por uma sequência de corpos rígidos denominados por elos (*links*) que são interconectados por partes móveis, denominadas articulações ou juntas. A estrutura fundamental do manipulador é dada pela sua cadeia cinemática aberta, que possui uma sequência de elos que conectam as duas extremidades da cadeia, sendo uma a base e a outra o efetuador (*end-effector*), no qual se encontra a ferramenta de trabalho para a execução da tarefa [3].

Existem aplicações em que se utiliza manipuladores robóticos em bases móveis. Nesse caso, necessita-se de uma instrumentação sobre a base móvel de modo a identificar as variações e possíveis erros estocásticos inerentes à robótica móvel, para que os mesmos sejam compensados. Um exemplo são antenas utilizadas na comunicação marítima, as quais estão instaladas sobre plataformas e navios. Essas antenas precisam apontar, com precisão, para um satélite geoestacionário. Dessa maneira, a inclinação e aproamento da embarcação devem ser compensados por um robô, sendo modelado como manipuladores robóticos para controle do apontamento do efetuador [2].

1.1 Revisão Bibliográfica

O controle de um manipulador envolve a posição e orientação da ferramenta. Para isso, determina-se um sistema de coordenadas ortonormais para cada uma das juntas [4], relacionando-se por meios de matrizes de transformações homogêneas [12].

Entre os anos de 1955 e 1956, Jacques Denavit e Richard Hartenberg propuseram uma forma particular de correlacionar referenciais consecutivos em uma cadeia cinemática, utilizando apenas 4 parâmetros, duas rotações e duas translações, resultando em uma matriz de transformação homogênea característica [14]. Os parâmetros de Denavit-Hartenberg [3, 4] são, até os dias atuais, uma convenção utilizada em cinemática direta de manipuladores robóticos.

Por meio da cinemática direta encontram-se funções que relacionam as variáveis das juntas com as posições e orientações do efetuador e seus elos. Utilizando o Jacobiano do modelo cinemático direto, é possível relacionar taxas de variação no espaço de juntas com as variações no espaço de trabalho, proporcionando métodos de cálculos dos parâmetros de junta relativos a movimentos desejados em espaço cartesiano [6].

A cinemática inversa, do inglês *Inverse Kinematics*, é um meio de calcular os parâmetros das juntas necessários para uma condição no espaço cartesiano ser atendida. Obter equações analíticas de cinemática inversa é, normalmente, uma tarefa sobremaneira complexa e, dependendo robô, não há solução exata conhecida. Assim, utilizam-se métodos numéricos [25] como alternativa para o cálculo da cinemática inversa. Existem diversos métodos de obtenção da cinemática inversa, os quais, em sua grande maioria, são métodos de otimização por meio de taxas de variações, que linearizam o modelo cinemático direto em um ponto de interesse utilizando matrizes Jacobianas e Hessianas, que fazem uso de pseudo-inversas dentre outras ferramentas matemáticas [29].

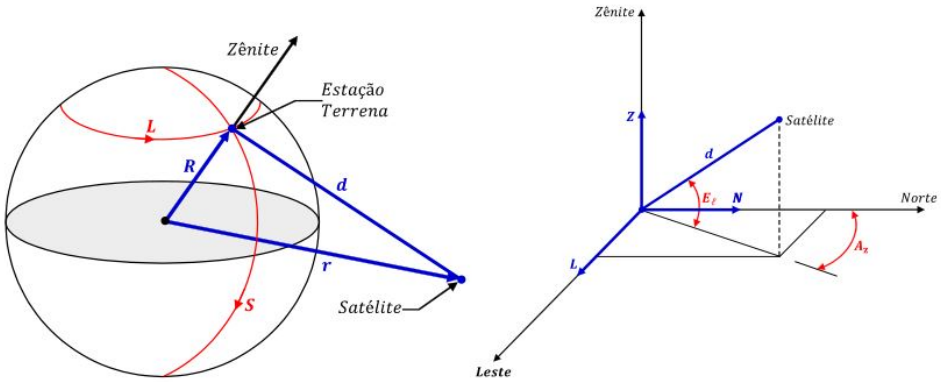
Uma técnica de otimização comumente usada na literatura é o método do Gradiente Descendente. Esta técnica tem sido utilizada com sucesso no cálculo da cinemática inversa, possibilitando encontrar soluções para posições e orientações de manipuladores industriais de diversos graus de liberdade. Com base no Jacobiano, este método busca recursivamente uma solução próxima do valor desejado por meio de uma função objetivo quadrática, até que o erro seja suficientemente pequeno [7].

Além de robôs industriais, um setor que utiliza controle de orientação em manipuladores robóticos é o de telecomunicação marítima, por meio de antenas automatizadas. Atualmente existem sistemas em navios que possibilitam grandes trocas de dados em alto-mar como o *VSAT (Very Small Aperture Terminal)* e televisão via satélite (*Television Receive-Only, TVRO*), em que ambos utilizam antenas direcionais que necessitam apontar para o satélite com grande precisão dentro de uma faixa angular bastante estreita [1].

Estas antenas, dotadas de motores precisos, são capazes de fazer o apontamento automático com base na leitura do GPS e de sensores que medem a inclinação e o aproamento da embarcação. Este conjunto é chamado de antena de apontamento automático com base inercialmente estabilizada. O sistema mantém o conjunto parábola/alimentador imóvel em relação ao satélite [2].

A direção na qual o manipulador deve apontar está de acordo com o ângulos de visada, que definem a posição do satélite em relação ao objeto em solo, os quais são usualmente dados em termo dos ângulos azimute A_z e elevação E_l, medidos em relação ao sistema de coordenadas horizontal-topocêntrico [28], como mostrado na Figura 1 .

Figura 1 – Sistema de coordenadas horizontal-topocêntrico.



Fonte: Imagem disponibilizada por [2].

As antenas de apontamento automático são manipuladores que possuem uma base móvel em relação ao sistema de coordenadas inercial. A orientação de um referencial em relação a outro pode ser obtida por meio de matrizes de rotação [6], em que apenas três parâmetros independentes são suficientes para representar a orientação de um corpo rígido em um espaço tridimensional. Estes parâmetros, denominados Ângulos de Euler [6], representam três rotações elementares e subsequentes em torno dos eixos, existindo doze possíveis combinações que satisfazem qualquer rotação [3]. Com relação aos ângulos de Euler, uma subcategoria muito utilizada é a *Tait–Bryan angles* (*Roll, Pitch e Yaw*) que são rotações subsequentes em 3 eixos cartesianos distintos, em destaque a sequência Euler ZYX, que são rotações no *frame* corrente (intrínsecas) na ordem Z (*Yaw*), Y (*Pitch*) e X (*Roll*) [13].

Para medir a orientação de um objeto pode-se usar uma IMU *(Inertial Measurement Unit)*, que é um dispositivo eletrônico que mede acelerações, velocidades rotacionais e, possivelmente, campos magnéticos, utilizando um acelerômetro, giroscópio e magnetômetro, todos com 3 eixos de medição [20]. Descrições completas destes sensores estão nas obras [26, 27]. Um exemplo é a GY-87, uma placa que contém o chip *MPU6050* com acelerômetro, giroscópio e chip *HMC5883L* com magnetômetro, ambos comunicam via rede I2C [9, 10].

Os ângulos *Roll* e *Pitch*, que serão definidos no Capítulo 3, são calculados pela fusão de medidas do acelerômetro, utilizando a aceleração da gravidade como referência [22]. Sabendo os dois ângulos mencionados previamente, o terceiro ângulo da tríade, *Yaw*, pode ser calculado por meio de fusão com medidas do magnetômetro, orientado pelo

campo magnético terrestre [24].

O envolvimento de sensores inerciais com manipuladores robóticos está presente na literatura em diversas aplicações, por exemplo, o uso de uma IMU no efetuador para estimar a orientação de um robô industrial [15], o uso de um manipulador de alta acurácia para calibrar o acelerômetro presente no efetuador [18], a utilização de uma IMU para teleoperação manual de um braço robótico [17] e para geração de trajetórias a serem seguidas pelo efetuador [16].

Sensores inerciais naturalmente produzem medidas contaminadas por ruídos estocásticos. Para descobrir o quanto um conjunto de medidas se aproxima de uma distribuição Gaussiana é o teste de Shapiro-Wilk [23]. A determinação de uma distribuição Gaussiana é útil pois existem diversas maneiras desenvolvidas na literatura para a filtragem desse tipo de ruído. Uma vez determinada a similaridade do ruído do sensor com uma distribuição Gaussiana, basta utilizar um filtro Gaussiano para reduzir drasticamente o ruído de maneira eficiente.

Em 1960 Rudolph Emil Kalman, propôs uma maneira iterativa e eficiente de amenizar ruídos Gaussianos, nominada Filtro de Kalman [8], detalhadamente explicada em [19]. Em termos da estimação da orientação utilizando uma IMU, o proposto por Kalman é uma das abordagens de filtragem mais populares [20]. No caso da aplicação proposta, o filtro de Kalman obtém um estado angular confiável de um sistema e, além disso, permite utilizar um modelo para estimar a velocidade *bias* do giroscópio, que é um *offset* na medição, e como este arraste varia constantemente, o procedimento de correção deve estar sempre atualizado [20, 21].

1.2 Objetivo e Organização do Trabalho

O presente trabalho tem como objetivo, por meio de integração de conhecimentos de manipuladores, sensores e filtros digitais, demonstrar um controle cinemático *online* de orientação de um manipulador robótico submetido a variações de orientação da base. As proposições serão demonstradas matematicamente e postas à prova por meio de resultados experimentais. A organização do trabalho segue a seguinte estrutura:

No Capítulo 2, encontra-se uma revisão acerca da cinemática de manipuladores robóticos, seu principais conceitos, modelagem via matrizes de transformação homogênea e controle cinemático utilizando a proposta de Denavid-Hartenberg, além de cinemática inversa envolvendo o Jacobiano inverso da matriz final do sistema e gradiente descendente.

No Capítulo 3, é apresentado a obtenção de valores confiáveis de orientação angular de um sistema, integrando medidas obtidas a partir de sensores inerciais de 3 eixos, sendo eles acelerômetro, magnetômetro e giroscópio. Para redução de ruídos é desenvolvido um filtro digital de Kalman para obtenção de medidas de rotações.

A apresentação da proposta teórica e o desenvolvimento do protótipo é demonstrado no Capítulo 4. Toda a parte de *hardware* e *software* utilizados, como os motores, placa de sensores, microcontrolador, integração com MATLAB no Windows, relatando suas funções para o sistema final.

No Capítulo 5 são apresentados os resultados deste trabalho, os modelos, simulações e experimentos, comprovando o arcabouço teórico utilizado. O Capítulo 6 apresenta as conclusões e ponderações sobre trabalhos futuros. Nos Anexos encontram-se todos os algoritmos utilizados no trabalho, como modelagem, simulação, calibração e protótipo.

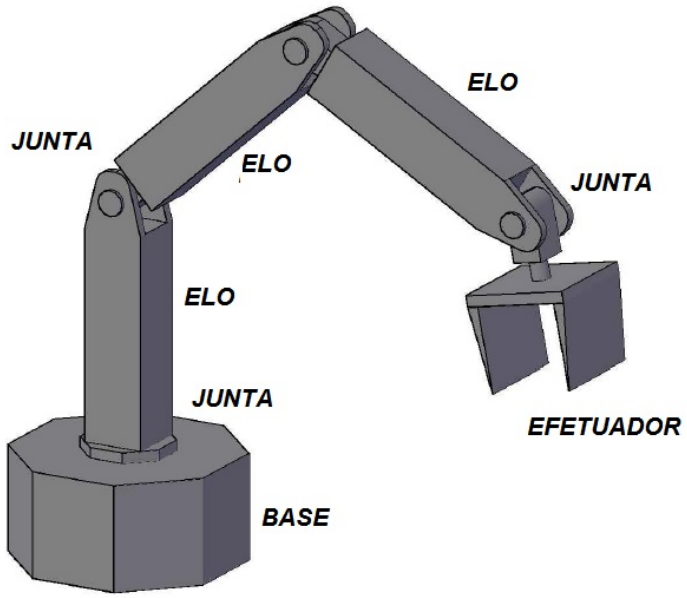
2 Cinemática de Manipuladores

2.1 Introdução

Para o estudo cinemático de manipuladores robóticos é necessário modelar matematicamente a localização e orientação de todos os seus elementos, incluindo suas juntas e elos. Juntas são conexões nas quais instalam-se dispositivos de movimento, como motores. As principais são as rotacionais (giram em torno de um eixo) e prismáticas (movimentos de translação sobre um eixo) [4].

O estudo de cinemática de manipuladores robóticos consiste em obter, a partir da posição da base, as propriedades geométricas e do valor das variáveis de juntas, que podem ser valores lineares ou angulares, dependendo do tipo da junta. Assim, é possível calcular a localização de todos os componentes do robô, procedimento definido por cinemática direta. Outra possibilidade é obtenção das variáveis de junta para uma determinada formação espacial desejada, que é por definição o problema inverso da cinemática direta e, portanto, denominado por cinemática inversa.

Figura 2 – Manipulador Robótico e seus aspectos construtivos



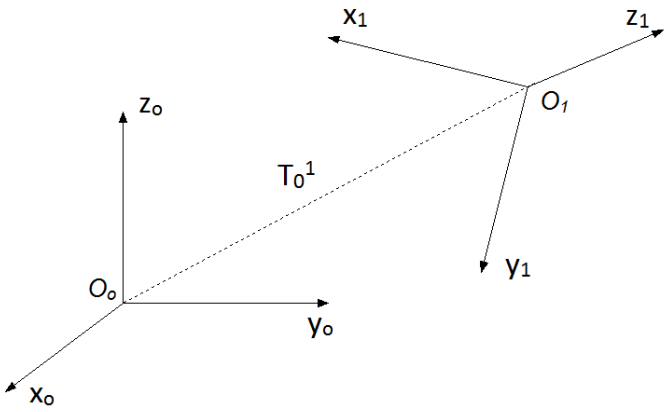
Fonte: Baseado em um bloco de Autocad gratuíto "Robotic Arm".

2.2 Transformações Homogêneas

Para representar a localização das juntas, utilizam-se os conceitos de sistemas de coordenadas (*frames*), que são os referenciais que contêm as coordenadas espaciais de interesse. Escolhe-se um referencial como o *frame* inercial para a base do robô, o qual

irá parametrizar os demais [7]. Por meio de matrizes de transformações homogêneas, modelam-se os *frames* em relação a outros dentro da cadeia cinemática, considerando a base como o primeiro e o efetuador como o último dos sistemas de referência.

Figura 3 – Exemplo transformação homogênea.



Fonte: Autoria Própria

Transformações homogêneas são representadas por matrizes 4×4, como mostrado na Equação (2.1), na qual a submatriz $\boldsymbol{R}_{3\times3}$ representa a matriz de rotação generalizada, que modifica a orientação dos eixos canônicos, e $\boldsymbol{p}_{3\times1}$ representa o vetor posição da origem do referencial atual em relação ao anterior [12].

$$
{}^{0}_{1}\boldsymbol{T} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & p_x \\ R_{21} & R_{22} & R_{23} & p_y \\ R_{31} & R_{32} & R_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{3\times3} & \boldsymbol{p}_{3\times1} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \tag{2.1}
$$

Relacionando a Equação (2.1) com a Figura 2.2, é possível identificar as coordenadas dos vetores canônicos no *frame* 1 com relação ao *frame* 0. O vetor coluna $[R_{11}\ R_{21}\ R_{31}]^T$ representa o vetor x_1, assim como a segunda, terceira e quarta colunas representam, respectivamente, y_1, z_1 e a origem do novo sistema 1.

2.3 Ângulos de Euler

Conforme pode ser visto na Equação (2.1), uma rotação em um sistema referencial pode ser representada por uma matriz de rotação $\boldsymbol{R}_{3\times3}$, e esta matriz pode ser modelada por uma sequência de 3 rotações elementares. Caso se utilize três eixos distintos, estas rotações podem ser chamadas de *Roll*, *Pitch* e *Yaw*, que representam rotações, positivas

no sentido anti-horário, nos eixos x, y e z, respectivamente. As matrizes referentes às rotações elementares podem ser vistas na Equações (2.2), (2.3) e (2.4) [6].

$$\boldsymbol{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix} \tag{2.2}$$

$$\boldsymbol{R}_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \tag{2.3}$$

$$\boldsymbol{R}_z(\alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

A parametrização dos ângulos de Euler depende da sequência em que as rotações forem realizadas, uma vez que diferentes combinações ordenadas produzem resultados finais diferentes. Outro fato a se considerar, é se as rotações podem ser intrínsecas ou extrínsecas. Rotações intrínsecas acontecem sobre os eixos atuais do *frame* corrente, sobre o sistema de coordenadas girante, que varia sua orientação conforme uma rotação é realizada. Extrínsecas são sobre os eixos de coordenadas fixo no espaço, normalmente o *frame* global, inercial.

Existem propriedades que relacionam os dois tipo de rotações. Toda sequência de rotações extrínsecas é equivalente a uma sequência de rotações intrínsecas dos mesmos ângulos, com a ordem das rotações colocada de forma inversa, e vice-versa. Por exemplo, uma sequência de rotações intrínseca $\boldsymbol{R}_z(\alpha), \boldsymbol{R}_y(\beta)$ *e* $\boldsymbol{R}_x(\gamma)$ é idêntica a uma sequência de rotações extrínsecas $\boldsymbol{R}_x(\gamma), \boldsymbol{R}_y(\beta)$ *e* $\boldsymbol{R}_z(\alpha)$ [6]. Outro ponto a se destacar é que ao realizar uma rotação intrínseca ${}^0_1\boldsymbol{R}$ no *frame* corrente, cuja orientação está parametrizada em relação ao referencial global pela matriz de rotação $\boldsymbol{R}^0_1$, a matriz de rotação do sistema final é resultado de uma pós-multiplicação, e o efeito de uma rotação extrínseca $\boldsymbol{R}_e$ no *frame* global é uma pré-multiplicação [6] conforme Equação (2.5).

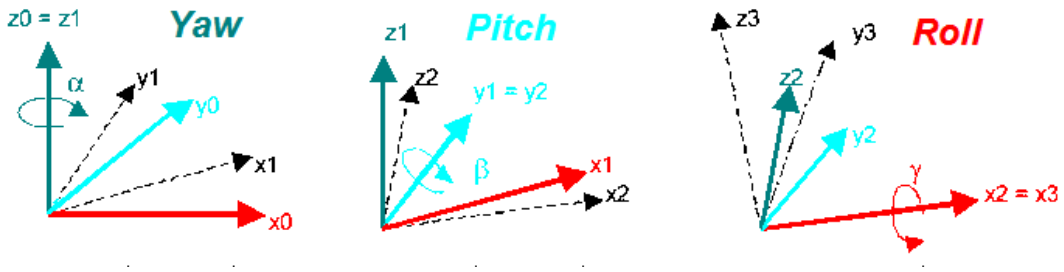
$$\begin{aligned} {}^0_2\boldsymbol{R} &= {}^0_1\boldsymbol{R}\ \boldsymbol{R}_i \\ {}^0_2\boldsymbol{R} &= \boldsymbol{R}_e\ {}^0_1\boldsymbol{R} \end{aligned} \tag{2.5}$$

2.3.1 Convenção de Euler tipo ZYX

Uma das mais utilizadas formas de parametrizar a rotação de um *frame* em relação a outro é a convenção Euler ZYX [13], consistindo em rotações intrínsecas *Yaw*, *Pitch* e

Roll em sequência no *frame* corrente, conforme Figura 4. Esta convenção é equivalente a RPY: rotações *Roll*, *Pitch* e *Yaw* extrínsecas no *frame* global.

Figura 4 – Composição da transformação rotacional ZYX.



Fonte: Autoria Própria

Baseando-se nas equações (2.2), (2.3) e (2.4), Define-se a matriz de rotação $\boldsymbol{R}_{zyx}$ conforme Equação 2.6.

$$\boldsymbol{R}_{zyx}(\alpha,\beta,\gamma) = \boldsymbol{R}_{z,\alpha}\boldsymbol{R}_{y,\beta}\boldsymbol{R}_{x,\gamma} = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{2.6}$$

Quando o problema é identificar os três parâmetros de Euler (α, β e γ), tendo a posse da matriz de transformação homogênea, utiliza-se a matemática inversa da equação (2.6). Tomando como base a submatriz $\boldsymbol{R}_{3\times3}$ contida na equação de transformação homogênea (2.1), calcula-se os ângulos de acordo com as Equações 2.7 [13].

$$\begin{aligned} &\textit{caso } (R_{11} = R_{21} = 0); \begin{cases} \alpha = 0, \\ \beta = \frac{\pi}{2}, \\ \gamma = \tan_2^{-1}(R_{12}, R_{22}) \end{cases} \\ &\textit{caso contrário}; \begin{cases} \alpha = \tan_2^{-1}(R_{21}, R_{11}), \\ \beta = \tan_2^{-1}(-R_{31}, \sqrt{R_{31} + R_{31}}), \\ \gamma = \tan_2^{-1}(R_{32}, R_{33}) \end{cases} \end{aligned} \tag{2.7}$$

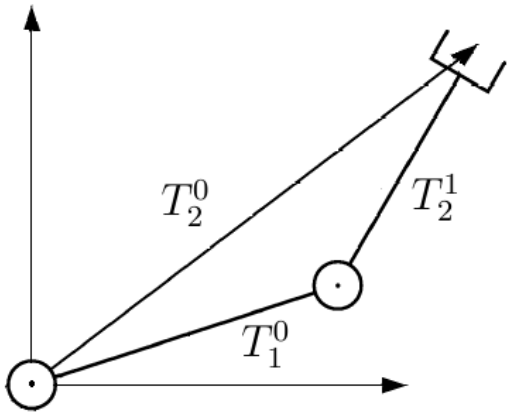
2.4 Cinemática Direta

Um robô manipulador é uma composição de elos (*links*) conectados entre si por meio de juntas. Cada junta é modelada por transformações homogêneas com relação

à junta anterior. Para uma melhor análise, todas as juntas são modeladas como uma transformação homogênea em relação ao *frame* inercial, conforme a propriedade matemática da cinemática de cadeia aberta mostrada na Equação (2.8), e a Figura 5 exemplifica a referenciada equação.

$$ {}^{0}_{n}\boldsymbol{T} = {}^{0}_{1}\boldsymbol{T}\ {}^{1}_{2}\boldsymbol{T} ... \ {}^{n-1}_{n}\boldsymbol{T} \tag{2.8} $$

Figura 5 – Ilustração exemplificando a Equação 2.8



Fonte: Autoria Própria

Neste tipo de aplicação, as matrizes de transformação homogênea apresentam elementos como funções matemáticas das variáveis de junta, $T(\theta_1, \theta_2, ..., \theta_n)$, em que θ pode ser um ângulo ou um deslocamento linear. Assim é possível obter toda a disposição do robô para uma configuração espacial em particular. A questão reside em como obter as matrizes de transformação homogênea conhecendo a configuração espacial do braço robótico. Para isso, uma convenção muito utilizada nas aplicações de manipuladores é a de Denavit-Hartenberg.

2.4.1 Parâmetros de Denavit-Hartenberg

A convenção de Denavit-Hartenberg é utilizada para calcular a matriz de transformação homogênea entre dois *frames* consecutivos na cadeia cinemática. O método dispõe de quatro operações em sequência, sendo duas translações e duas rotações. A matriz obtida na Equação (2.9) é resultado do produto dessas quatro operações na seguinte ordem: Rotação em z, translação em z, translação em x e rotação em x.

$$\boldsymbol{A}_i = \boldsymbol{R}_z(\theta_i)\boldsymbol{T}_z(d_i)\boldsymbol{T}_x(a_i)\boldsymbol{R}_x(\alpha_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & \alpha_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & \alpha_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Os valores das 4 variáveis da Equação (2.9) são obtidos geometricamente, conforme os passos abaixo [3], para uma transformação arbitrária de uma junta 0 para junta 1, a Figura 6 ilustra este procedimento, aonde o *frame* 0 é corrente e varia conforme o as etapas são realizadas.

- Escolher os vetores $\overrightarrow{z_0}$ e $\overrightarrow{z_1}$ das juntas sobre os eixo de rotação/translação. O vetor $\overrightarrow{x_1}$ deve ser posto na normal a $\overrightarrow{z_0}$ e $\overrightarrow{z_1}$. No caso especial em que os eixos Z são paralelos, a origem do *frame* subsequente é livre, pois a normal pode estar em qualquer lugar ao longo dos eixos Z.

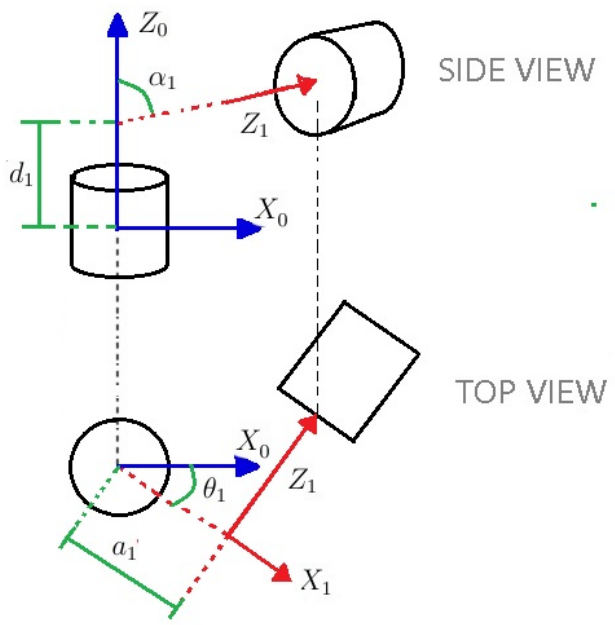
- $\boldsymbol{R}_z(\theta_1)$: Rotacionar o eixo $\overrightarrow{z_0}$ de θ_1 radianos de modo que o vetor $\overrightarrow{x_0}$ torne-se paralelo a $\overrightarrow{x_1}$

- $\boldsymbol{T}_z(d_1)$: Translação da origem O_0 na direção de $\overrightarrow{z_0}$ de d_1 unidades de comprimento, aonde d_1 é a diferença de altura entre as origem em relação a coordenada $\overrightarrow{z_0}$, desta forma as duas origens estão alinhadas sobre o eixo $\overrightarrow{x_1}$.

- $\boldsymbol{T}_x(a_1)$: Translação de O_0 na direção de $\overrightarrow{x_1}$ de a_1 unidades de comprimento, de modo que as origens estejam sobre o eixo $\overrightarrow{z_1}$.

- $\boldsymbol{R}_x(\alpha_1)$: Rotacionar o eixo $\overrightarrow{x_0}$ de α_1 radianos de modo que o vetor $\overrightarrow{z_0}$ esteja paralelo ao vetor $\overrightarrow{z_1}$

Figura 6 – Exemplo Denavit-Hartenberg



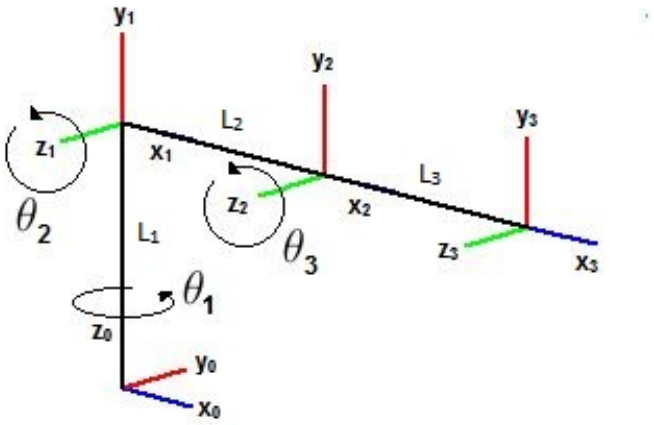
Fonte: Autoria Própria

É comum montar uma tabela em que cada linha contém os parâmetros da i-ésima transformação. Ao final do processo, de acordo com a propriedade da Equação (2.8), multiplica-se as matrizes para obter o modelo geométrico direto de transformação do *frame* inercial ao efetuador. Um exemplo será visto na Tabela 1.

2.4.1.1 Cinemática Direta Manipulador Antropomórfico

No caso do manipulador antropomórfico, sua cinemática direta pode ser analisada conforme a Figura 7, onde é apresentado uma possível orientação de *frames* para o valor das variáveis de junta nulos.

Figura 7 – Modelagem do Manipulador Antropomórfico RRR



Fonte: Autoria Própria

Como pode ser visto geometricamente, na transformação do *frame* 0 para o 1, há uma translação em z_0 de L_1 unidades de comprimento e uma rotação em x_0 de $\pi/2$ rad. Para os seguintes *frames* apenas translações em x_1 *e* x_2 *de* L_2 *e* L_3 unidades de comprimento. Por todas as juntas serem rotacionais, todas as variáveis são rotações em z de θ_i rad. A Tabela 1 apresenta os parâmetros de Denavit-Hartenberg.

Tabela 1 – Tabela DH para Manipulador Antropomórfico

Junta	θ_i	d_i	a_i	α_i
1	θ_1	L_1	0	$\pi/2$
2	θ_2	0	L_2	0
3	θ_3	0	L_3	0

O modelo geométrico de cinemática direta pode ser obtido multiplicando as matrizes de Denavit-Hartenberg em sequência, chegando na matrizes de transformação da base para o efetuador. Para simplificar a descrição, faz-se cosseno=C e seno=S:

$$
{}^0_3\boldsymbol{T} = {}^0_1\boldsymbol{A}\,{}^1_2\boldsymbol{A}\,{}^2_3\boldsymbol{A} =
$$

$$
\begin{bmatrix}
C(\theta_2+\theta_3)C(\theta_1) & -S(\theta_2+\theta_3)C(\theta_1) & S(\theta_1) & C(\theta_1)(L_3C(\theta_2+\theta_3)+L_2C(\theta_2)) \\
C(\theta_2+\theta_3)S(\theta_1) & -S(\theta_2+\theta_3)S(\theta_1) & -C(\theta_1) & S(\theta_1)(L_3C(\theta_2+\theta_3)+L_2C(\theta_2)) \\
S(\theta_2+\theta_3) & C(\theta_2+\theta_3) & 0 & L_1+L_3S(\theta_2+\theta_3)+L_2S(\theta_2) \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{2.10}
$$

2.5 Cinemática Inversa

A seção anterior apresentou a obtenção do modelo geométrico direto de um manipulador robótico. No entanto, para controlar o robô é necessário utilizar a cinemática inversa. Dado um estado cartesiano desejado para o robô, de posição e/ou orientação do efetuador, precisa-se obter o conjunto de variáveis de junta que é necessário para que a configuração desejada seja realizada. Dependendo da configuração construtiva do robô, esta solução pode não ser única.

Cada termo da matriz de transformação homogênea final do robô é uma expressão matemática em função das variáveis de junta. Utilizando métodos analíticos, é realizada a inversão destas equações, conduzindo à obtenção de todas as soluções possíveis. No entanto, a complexidade desta inversão é um desafio, e não há garantias de que seja possível fazê-la, servindo apenas para robôs simples [25]. O mais comum é utilizar métodos numéricos iterativos, que convergem para uma solução possível entre as existentes.

2.5.1 Cinemática Diferencial

Seja um vetor coluna com os estados no ambiente de trabalho do efetuador que se deseja controlar $\boldsymbol{x}_e = [x_{e1}, x_{e2}, \cdots, x_{en}]^T$, que são funções das variáveis de junta conforme a Equação 2.11. Caso queira-se controlar a posição e orientação de maneira completa, deve-se considerar $\boldsymbol{x}_e = [px, py, pz, \alpha, \beta, \gamma]^T$, que caracteriza o efetuador por meio da localização de sua origem no espaço de trabalho e sua orientação através de três parâmetros de Euler de rotações elementares.

$$\boldsymbol{x}_e = \boldsymbol{f}(\boldsymbol{\theta}) \tag{2.11}$$

Aplicando operações de diferenciação em ambos os termos da Equação (2.11) utilizando a regra da cadeia, obtém-se a taxa de variação de uma variável com relação à outra, ou seja, a velocidade no espaço de trabalho dada a velocidade do espaço das juntas.

$$\dot{\boldsymbol{x}}_e = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{2.12}$$

As duas velocidades se relacionam pelo Jacobiano geométrico, que é a relação diferencial de um vetor de n elementos, em que cada elemento é uma função de m variáveis de junta. O Jacobiano é definido pela Equação (2.13), que representa a direção de máximo

crescimento da função $\boldsymbol{f}(\boldsymbol{\theta})$.

$$\boldsymbol{J}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_m} \\ \frac{\partial f_2}{\partial \theta_1} & \ddots & & \\ \vdots & & & \vdots \\ \frac{\partial f_n}{\partial \theta_1} & & \cdots & \frac{\partial f_n}{\partial \theta_m} \end{bmatrix} \tag{2.13}$$

2.5.2 Método Numérico Recursivo Utilizando Gradiente Descendente

Seja o estado que se deseja posicionar o *end-effector* do robô um vetor da forma $\boldsymbol{x}_d = [x_{d1}, x_{d2}, \cdots, x_{dn}]^T$, e o vetor de erro entre a posição atual e a desejada para o robô dado por $\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}_e$. Com essas informações, pode-se calcular o erro quadrático médio pela expressão:

$$\boldsymbol{EQM} = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \tag{2.14}$$

Para minimizar o erro via gradiente descendente é necessário obter a direção de máximo crescimento por meio do gradiente, conforme Equação (2.15).

$$\nabla \boldsymbol{EQM} = -(\boldsymbol{x}_d - \boldsymbol{x}_e) = -\boldsymbol{e} \tag{2.15}$$

$$[\boldsymbol{x}_e]_{k+1} = [\boldsymbol{x}_e]_k + \eta\ \boldsymbol{e} \tag{2.16}$$

Para minimizar o erro, o sistema deve ser atualizado na direção oposta a de $\nabla \boldsymbol{EQM}$, que é o próprio vetor do erro ($\boldsymbol{e}$), de maneira iterativa a uma taxa de aprendizado η para garantir a convergência. No caso da cinemática inversa, este procedimento é modificado para atuar no espaço de juntas, já que são as variáveis de controle.

Considere a Equação 2.12, que relaciona as taxas de variação. Para uma variação pequena, é possível considerar a equação linearizada do modelo cinemático direto da seguinte maneira:

$$\Delta \boldsymbol{x}_e = \boldsymbol{J}(\theta)\Delta\boldsymbol{\theta} \tag{2.17}$$

O objetivo é descobrir a variação no espaço de juntas $\Delta\boldsymbol{\theta}$, que, a cada iteração atualizará o espaço cartesiano conforme a Equação (2.16). Portanto, manipula-se a Equação (2.17) substituindo $\Delta\boldsymbol{x}_e$ pela variação desejada no espaço de trabalho $\eta\boldsymbol{e}$. Para o caso da matriz $\boldsymbol{J}$ ser quadrada, tem-se que a solução é dada por

$$\Delta\boldsymbol{\theta} = \eta\boldsymbol{J}(\theta)^{-1}\boldsymbol{e} \tag{2.18}$$

No entanto, se a dimensão do vetor $\boldsymbol{x}_e$ for diferente do vetor $\boldsymbol{\theta}$, o Jacobiano não possui inversa. Isto ocorre quando o robô possui quantidade de variáveis de junta diferente do número de variáveis contidas no vetor objetivo. O procedente é usar a matriz pseudo-inversa, também conhecida como Matriz de *Moore–Penrose* $\boldsymbol{J}^{\dagger}$ (Equação (2.19)) [29], que utiliza decomposição em valores singulares para obter uma solução para o sistema linear (2.17).

$$\begin{aligned} \boldsymbol{J}(\theta)^{\dagger} &= (\boldsymbol{J}(\theta)^T \boldsymbol{J}(\theta))^{-1} \boldsymbol{J}(\theta)^T \\ \Delta\boldsymbol{\theta} &= \boldsymbol{J}^{\dagger} \Delta \boldsymbol{x}_e \end{aligned} \tag{2.19}$$

O ciclo é finalizado na atualização das juntas para um novo estado, conforme (2.20), o qual irá acompanhar o gradiente descendente no espaço de juntas. O algoritmo do gradiente descendente se interrompe quando o módulo do vetor de erro ($\boldsymbol{e}$) for suficientemente pequeno, considerando assim que o objetivo foi alcançado.

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \boldsymbol{J}(\theta)^{\dagger} \boldsymbol{e} \tag{2.20}$$

Ao usar mínimos quadrados, caso $det|J(\theta)^T J(\theta)| = 0$, não haverá inversa e deve-se tratar a matriz singular. Uma maneira de resolver é utilizar a estratégia de Levenberg-Marquardt, que consiste em fazer $J(\theta)^T J(\theta) + \delta I_{n(n\times n)}$, em que δ é um valor real pequeno, para se calcular uma solução aproximada para o sistema linear [7].

2.5.3 Controle de Orientação em uma Manipulador Antropomórfico

Uma possível metodologia para controlar a orientação do efetuador de um manipulador antropomórfico com base na seção anterior é utilizar os ângulos de Euler no vetor de estado do espaço de trabalho $\boldsymbol{x}_e = [\alpha, \beta, \gamma]^T$. Para obter estes parâmetros do efetuador em função dos ângulos das juntas, utiliza-se a Equação (2.7) sobre a matriz da Equação (2.10), para $\theta_2 + \theta_3$ entre $-\pi/2$ e $\pi/2$, obtendo a Equação (2.21).

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} atan2(C(\theta_2+\theta_3)S(\theta_1), C(\theta_2+\theta_3)C(\theta_1)) \\ atan2(-S(\theta_2+\theta_3), C(\theta_2+\theta_3)) \\ \pi/2 \end{bmatrix} \tag{2.21}$$

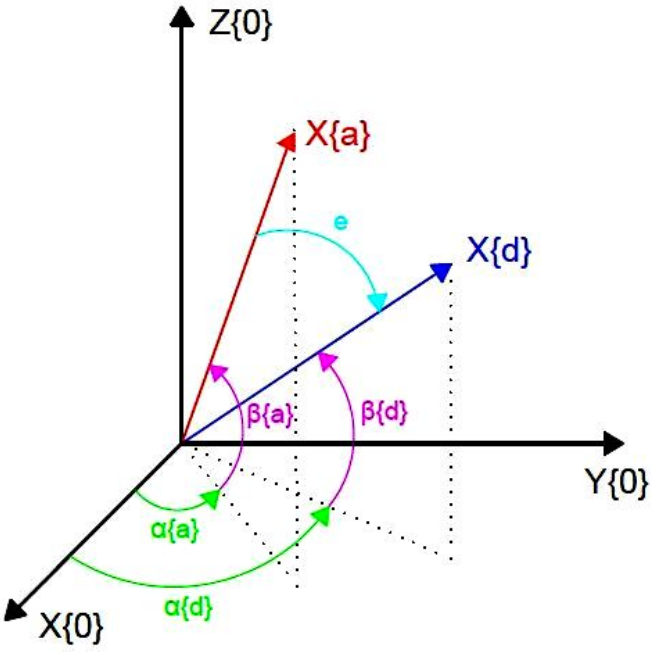
Como pode ser visto, o valor de *Roll* é fixo, e por isso não pode ser alterado pelas variáveis de junta neste tipo de manipulador, o que pode ocasionar complicações no cálculo da cinemática inversa pela inflexibilidade gerada no modelo. Se o objetivo é apenas a direção de apontamento do efetuador (eixo x_3), é suficiente utilizar $\boldsymbol{x}_e = [\alpha, \beta]^T$, uma vez

que na convenção de Euler ZYX, *Roll* é a última rotação intrínseca a ser realizada, e não interfere na direção do próprio vetor x_3. É o caso do robô desenvolvido neste trabalho e, portanto, a inflexibilidade não afeta negativamente o controle do manipulador.

A cinemática inversa da seção 2.5.2 pode, então, ser aplicada sem qualquer restrição. A Figura 8 ilustra e exemplifica este processo. Dada a atual posição do eixo x_a do efetuador e o destino desejado x_d, parametrizados por suas rotações α (*Yaw*) e β (*Pitch*), o algoritmo atualizará x_a em direção ao gradiente da função de erro quadrático médio, o vetor $\boldsymbol{e}$, que no caso é definido pela Equação (2.22). Estes Ângulos são equivalente aos ângulos azimute A_z e elevação E_l (Figura 1), utilizados como referências de apontamento geoespacial.

$$\boldsymbol{e} = \begin{bmatrix} \alpha_d - \alpha_a \\ \beta_d - \beta_a \end{bmatrix} \tag{2.22}$$

Figura 8 – Cinemática de Orientação RRR



Fonte: Autoria Própria

3 Estimação da Orientação de uma Plataforma

Este capítulo apresenta o processo de obtenção de medidas confiáveis da orientação de um dispositivo monitorado por uma unidade de medição inercial. Este processo precisa de nove medidas de sensores distintas na mesma iteração para se obter a orientação do dispositivo em relação a um *frame* inercial, que utiliza como referência a gravidade e o campo magnético terrestre. Os ruídos dessas medições são fontes de erros para o sistema, que devem ser filtrados.

3.1 Sensores Inerciais

Sensores inerciais medem aceleração e velocidade angular em aplicações de análise de movimento por meio de acelerômetros e giroscópios, sendo comum adicionar magnetômetros para se obter de maneira completa a análise das variáveis de orientação.

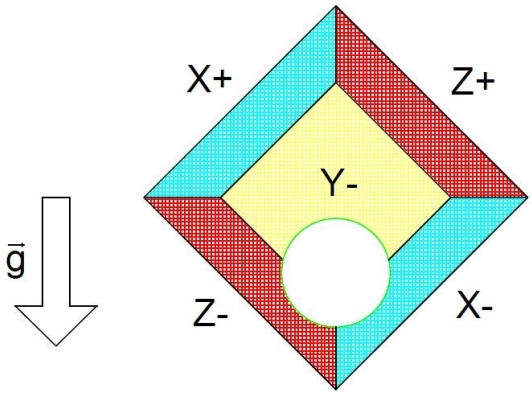
3.1.1 Acelerômetro

Acelerômetros são transdutores que produzem um sinal de saída proporcional a aceleração, vibração e choque que o dispositivo esta submetido. Tendo uma grande variedade de aplicações, e os mais comuns são os piezoelétricos [26].

Um material piezoelétrico produz um sinal de tensão proporcional à força mecânica aplicado sobre o mesmo. O princípio de funcionamento de um acelerômetro eletrônico é por meio de uma massa sobre um cristal piezoelétrico, que exercerá uma força sobre o cristal conforme a aceleração que o dispositivo sofre, seja por mudança inercial ou pela aceleração da gravidade.

Normalmente o aparelho é elaborado para medições em 3 eixos ortonormais, sendo que cada eixo mede a força e sentido da decomposição da aceleração sobre o mesmo. A Figura 9 ilustra este processo para um dispositivo submetido unicamente à força da gravidade.

Figura 9 – Princípio de funcionamento de um acelerômetro



Fonte: Autoria Própria

3.1.2 Magnetômetro

Os magnetômetros medem a grandeza de um campo magnético estático, e para tanto utilizam o princípio da interação eletromagnética [26]. Comumente são utilizados materiais magneto-resistentes, que alteram o valor da resistência elétrica com aplicação de um campo magnético.

Destaca-se a magnetorresistência anisotrópica (*anisotropic magnetoresistance, AMR*), que é a dependência resistiva entre o ângulo, a corrente elétrica e o campo magnético, devido ao espalhamento provocado pelo efeito Hall. Suas aplicações envolvem construção de bússolas digitais, por meio da captação do campo magnético terrestre, e medidores de intensidade de campo magnéticos externos.

3.1.3 Giroscópio

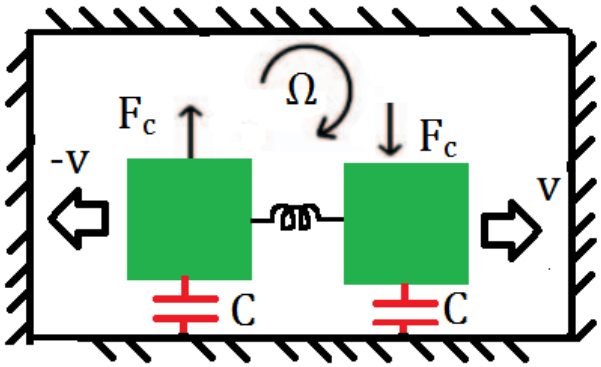
Giroscópio é um sensor que mede velocidade angular, eles podem ser divididos em vários tipos de acordo com o princípio de funcionamento. Um deles é o giroscópio *MEMS (Micro-Electro-Mechanical Systems)* de vibração linear [27]. O princípio de funcionamento é o efeito Coriolis, que é uma aceleração aplicada a um objeto que possui, simultaneamente, uma velocidade linear $\vec{v}$ e uma angular Ω.

$$\vec{A_c} = 2 \cdot \Omega \times \vec{v} \tag{3.1}$$

Seu princípio pode ser analisado como duas massas pareadas oscilando sempre em sentido opostos a mesma velocidade. Quando uma velocidade angular é aplicada, as massas movimentam, variando uma capacitância interna. A saída do sensor é proporcional à diferença entre as duas variações de capacitância. Acelerações lineares externas produzem

forças de mesma sentido nas duas massas, com isso consegue-se compensar os seus efeitos. A figura 10 ilustra este processo.

Figura 10 – MEMS Gyros



Fonte: Autoria Própria

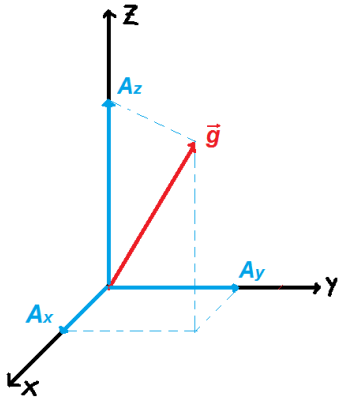
3.2 Cálculos dos Ângulos de Euler

Esta seção apresenta métodos de obter a inclinação e o ângulo azimute de uma plataforma planar com liberdade rotacional e translações brandas. Utiliza-se nesta análise um magnetômetro e um acelerômetro ideais, ambos com medidas em 3 eixos ortonormais.

3.2.1 *Roll* e *Pitch*

Para obter a inclinação de uma superfície com liberdade rotacional e que não esteja submetida a translações consideráveis, pode-se utilizar leituras de um acelerômetro com 3 eixos. Como a força da gravidade atua sobre os 3 sensores, e os mesmo são ortonormais, é possível obter o vetor aceleração gravitacional $\vec{g}$ pela soma vetorial das 3 medições, como na Figura 11, $|\vec{g}|^2 = A_x^2 + A_y^2 + A_z^2$.

Figura 11 – Decomposição do Vetor Gravidade



Fonte: Autoria Própria

Nesse caso, é importante considerar o *frame* inercial com o eixo z_o paralelo a força da gravidade. É analisado também o atual *frame* da base do acelerômetro. Dessa maneira parametriza-se o vetor da gravidade em relação ao *frame* inercial como ${}^{O}\boldsymbol{g} = [0, 0, |\vec{g}|]^T$ e ao *frame* corrente da base como ${}^{B}\boldsymbol{g} = [A_x, A_y, A_z]^T$. É necessário equacionar as matrizes de rotação que relacionam os dois referenciais, e com isto relacionar o vetor gravitacional de acordo com a equação (3.2) [22], em que ${}^{B}_{O}\boldsymbol{R}$ é a matriz de rotação que transforma o *frame* da base no *frame* inercial.

$$ {}^{B}\boldsymbol{g} = {}^{B}_{O}\boldsymbol{R}\ {}^{O}\boldsymbol{g} \tag{3.2} $$

Conforme visto na seção 2.3, a orientação de um corpo rígido em relação ao referencial global é representada por ângulos *Yaw, Pitch* e *Roll* (Euler ZYX), com a rotação sucessiva sobre os eixos z , y e x [22], resultando na matriz R_{zyx} (Equação (2.6)) que, no caso, será equivalente à ${}^{O}_{B}\boldsymbol{R}$, a qual transforma o *frame* inercial no *frame* da base.

$$ {}^{O}_{B}\boldsymbol{R} = \boldsymbol{R}_z(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\gamma) \tag{3.3} $$

Utilizando a propriedade das matrizes de rotação (${}^{B}_{O}\boldsymbol{R} = {}^{O}_{B}\ \boldsymbol{R}^T$), pode-se relacionar as Equações (3.2) e (3.3) da seguinte maneira:

$$ {}^{B}\boldsymbol{g} = {}^{O}_{B}\boldsymbol{R}^T\ {}^{O}\boldsymbol{g} = \boldsymbol{R}_x(\gamma)^T\boldsymbol{R}_y(\beta)^T\boldsymbol{R}_z(\alpha)^T\ {}^{O}\boldsymbol{g} \tag{3.4} $$

Finalmente substituindo as Equações 2.2, 2.3 e 2.4 e simplificando 3.4, encontra-se

uma relação entre os valores do acelerômetro e os ângulos de Euler ZYX.

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = |\vec{g}| \begin{bmatrix} -\sin(\beta) \\ \sin(\gamma)\cos(\beta) \\ \cos(\gamma)\cos(\beta) \end{bmatrix} \tag{3.5}$$

Conforme viseto, o ângulo α da rotação *Yaw* não pode ser obtido por meio de medidas de acelerômetro, provando que a convenção Euler ZYX não interfere no reconhecimento da inclinação da base. Manipulando a equação 3.5 obtém-se os ângulos *Pitch* e *Roll*.

$$\beta = atan2(-A_x, \sqrt{A_y^2 + A_z^2}) \tag{3.6}$$

$$\gamma = atan2(A_y, A_z) \tag{3.7}$$

3.2.2 *Yaw*

Para a obtenção do ângulo *Yaw*, é utilizado como referência o vetor do campo magnético da Terra. O magnetômetro possui medições de 3 eixos ortogonais, cuja saída é um vetor parametrizado na base do próprio magnetômetro como ${}^{B}\boldsymbol{B} = [M_x, M_y, M_z]^T$, que ao considerar apenas a atuação do campo magnético da Terra $\vec{B}$, representa a decomposição deste vetor no referencial do magnetômetro, $|\vec{B}|^2 = M_x^2 + M_y^2 + M_z^2$.

Considerando que os eixos x_o e y_o do *frame* inercial estão sobre o plano horizontal, com x_o apontando para o norte da Terra, podemos definir o campo magnético terrestre no referencial global ${}^{O}\boldsymbol{B}$ pela equação (3.8), em que δ é o ângulo entre x_o e $\vec{B}$, e que varia de acordo com a latitude do local.

$${}^{O}\boldsymbol{B} = |\vec{B}| \begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix} \tag{3.8}$$

Tal qual para *Roll* e *Pitch*, para calcular *Yaw* deve-se relacionar o vetor no referencial atual com o inercial por meio das matrizes de rotação. Como o objetivo é achar *Yaw* para a metodologia Euler ZYX da Seção 2.3.1, deve-se considerar a matriz ${}^{O}_{B}\boldsymbol{R} = \boldsymbol{R}_{zyx}$.

$$\begin{aligned} {}^{B}\boldsymbol{B} &= {}^{B}_{O}\boldsymbol{R}\ {}^{O}\boldsymbol{B} = {}^{O}_{B}\boldsymbol{R}^T\ {}^{O}\boldsymbol{B} \\ {}^{B}\boldsymbol{B} &= \boldsymbol{R}_x(\gamma)^T \boldsymbol{R}_y(\beta)^T \boldsymbol{R}_z(\alpha)^T\ {}^{O}\boldsymbol{B} \end{aligned} \tag{3.9}$$

Utilizando a propriedade das matrizes ortogonais: $\mathbf{R}\mathbf{R}^T = \boldsymbol{I}_{n(3\times3)}$, pode-se simplificar e isolar o parâmetro α, multiplicando os termos da equação por $\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\gamma)$. Esta

operação é equivalente a realizar, em ambos os termos da equação, rotações extrínsecas em x e y sucessivamente.

$$\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\gamma)\ {}^{B}\boldsymbol{B} = \boldsymbol{R}_z(\alpha)^T\ {}^{O}\boldsymbol{B}$$

$$\begin{bmatrix} \cos\beta & \sin\beta\sin\gamma & \sin\beta\cos\gamma \\ 0 & \cos\gamma & -\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \begin{bmatrix} Mx \\ My \\ Mz \end{bmatrix} = \begin{bmatrix} cos\alpha & sin\alpha & 0 \\ -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} |\vec{B}| \begin{bmatrix} \cos\delta \\ 0 \\ \sin\delta \end{bmatrix}$$

$$\begin{bmatrix} M_x\cos\beta + M_y\sin\beta\sin\gamma + M_z\sin\beta\cos\gamma \\ M_y\cos\gamma - M_z\sin\gamma \\ -Mx\sin\beta + M_y\cos\beta\sin\gamma + M_z\cos\beta\cos\gamma \end{bmatrix} = |\vec{B}| \begin{bmatrix} \cos\alpha\cos\delta \\ -\sin\alpha\cos\delta \\ \sin\delta \end{bmatrix} \tag{3.10}$$

Dividindo a segunda linha da Equação (3.10) pela primeira, é possível obter *Yaw* para o modelo proposto [24].

$$\alpha = atan2(M_z\sin\gamma - M_y\cos\gamma\ ,\ M_x\cos\beta + M_y\sin\beta\sin\gamma + M_z\sin\beta\cos\gamma) \tag{3.11}$$

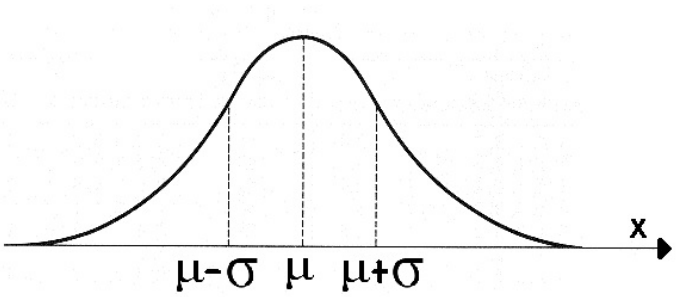
3.3 Filtragem de Erros

Um sistema real apresenta inúmeras imperfeições, seja por características de construção ou por fatores aleatórios. Há erros de origem determinística e não-determinística. Os erros determinísticos podem ser neutralizados durante a modelagem como por exemplo, valores de *offset* na medição, ou compensar algum erro construtivo por meio de calibração. Erros não-determinísticos não podem ser eliminados por calibração, pois têm origem aleatória. Para os sensores apresentados no trabalho, a maioria das medições estão poluídas por erros não-determinísticos. Uma maneira de amenizá-los é por meio de filtragens por processos estocásticos.

Fazendo uma análise estocástica de um conjunto de amostras de um mesmo estado do sistema, caso exista uma tendência para uma distribuição Gaussiana, é possível utilizar filtros Gaussianos e Bayesianos. A curva Gaussiana representa a função densidade de probabilidade de uma variável aleatória, cuja a função normalizada é dada pela Equação (3.12). O valor de μ é a média, o estado mais provável do sistema, σ é o desvio padrão, que representa o espalhamento das amostras, sendo que no intervalo de $(\mu - \sigma, \mu + \sigma)$ contém 68% da área da curva (probabilidade total).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{3.12}$$

Figura 12 – Distribuição Gaussiana



Fonte: Autoria Própria

3.3.1 Teste de Shapiro-Wilk

Para descobrir o quanto um conjunto de amostras se comporta como uma população normalmente distribuída, pode-se utilizar, dentre outros, o teste de Shapiro-Wilk [23], que é baseado na variável W dada por:

$$W = \frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})} \tag{3.13}$$

Onde x_i e a i-ésima amostra (posicionadas em ordem crescente em valor) e $\bar{x}$ é a média simples entre as amostras. Os valores de a_i podem ser calculados, mas o mais comum e usar valores tabelados nas *"Shapiro-Wilk Tables"*, presente no Anexo A.1, que fornecem o valor dos a_i para n amostras.

As *"Shapiro-Wilk Tables"* apresentam também o valor "p" (Anexo A.2), de acordo com o W calculado e o número de amostras, obtêm-se o valor aproximado de p que define o nível de significância, o quão parecida com uma Normal é a amostra de dados. Quando o "p" é inferior a 0,05 é comum considerar que o conjunto foi reprovado no teste.

3.3.2 Filtro de Kalman

O filtro de Kalman é um algoritmo recursivo utilizado para estimar o estado de um sistema a partir de medidas ruidosas de um sensor real. Para isso é necessário um modelo virtual do sistema, e que o ruído seja Gaussiano (passe no teste de Shapiro-Wilk). O filtro é divido em duas etapas: Predição e Atualização [19].

O algoritmo original do filtro de Kalman é eficiente quando o sistema a ser medido possuí um comportamento linear, acrescido de um ruído Gaussiano branco w_k (distribuição normal com covariância Q e de média zero) e que seu estado possa ser modelado conforme

a Equação (3.14).

$$x_k = f(x_{k-1}, u_k) + w_k \quad ; \quad w_k(N(0, Q_k)) \tag{3.14}$$

É considerado que este sistema está sendo observado, e que essa medição do estado $\boldsymbol{x}_k$ forma o vetor $\boldsymbol{z}_k$ descrito pela equação (3.15), que está corrompido com um ruído gaussiano v_k. A matriz $\boldsymbol{H}$, denominada matriz de observação, relaciona as variáveis do Filtro de Kalman com as medições reais. Das variáveis de estado do filtro, apenas as que estão sendo medidas participam do espaço observável.

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k \quad ; \quad \boldsymbol{v}_k(N(0, R_k)) \tag{3.15}$$

3.3.2.1 Processo de Predição

Tendo em vista um bom modelo matemático que simule o comportamento real do sistema, obtêm-se uma estimativa *a priori* do estado atual $\bar{\boldsymbol{x}}_k$. De maneira geral pode ser escrito conforme a Equação (3.16), denominada equação de predição de estado.

$$\bar{\boldsymbol{x}}_k = \boldsymbol{A}_k\boldsymbol{x}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k \tag{3.16}$$

Onde $\boldsymbol{x}$ é o vetor coluna com as variáveis de estado, sendo a saída do filtro de Kalman, $\boldsymbol{u}$ é a entrada de controle. $\boldsymbol{A}$ e $\boldsymbol{B}$ são as matrizes específicas da modelagem, a de modelo de transição de estado e modelo de entrada de controle respectivamente.

Nesta etapa também é formulada a equação de predição da covariância, Equação (3.17), na qual se estima, por meio da matriz de covariância do erro a priori $\bar{\boldsymbol{P}}_k$, a acurácia da medida, ou seja, o quão confiável é o estado predito do sistema. Utiliza-se a matriz de covariância do ruído do processo $\boldsymbol{Q}$, que é constituída a partir dos desvios padrão escolhidos para as variáveis do vetor $\boldsymbol{x}$, quando as variáveis são consideradas independentes a matriz $\boldsymbol{Q}$ é definida pela Equação (3.18).

$$\bar{\boldsymbol{P}}_k = \boldsymbol{A}_k\boldsymbol{P}_{k-1}\boldsymbol{A}_k^T + \boldsymbol{Q} \tag{3.17}$$

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \sigma_n^2 \end{bmatrix} \tag{3.18}$$

Basicamente, esta etapa se define em prever estado e covariância do erro para o próximo ciclo do algoritmo, baseado no modelo do sistema definido pelo usuário por meio dos parâmetros $\boldsymbol{A}$, $\boldsymbol{B}$ e $\boldsymbol{Q}$ [19].

3.3.2.2 Processo de Atualização

Nesta etapa a diferença entre predição e a medida real é compensada. Com os valores dos sensores reais, calcula-se o erro residual $\tilde{y}$, chamado de valor de inovação.

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\bar{\boldsymbol{x}}_k \tag{3.19}$$

Em seguida, é preciso calcular o ganho ótimo de Kalman, utilizando do mínimo erro quadrático para minimizar o valor esperado quadrático do erro entre o valor estimado e o real. Calcula-se primeiramente a matriz $\boldsymbol{S}$ dos sistema (Equação (3.20)), que informa a partir de $\bar{\boldsymbol{P}}_k$ a confiabilidade da medição real $\boldsymbol{z}_k$ em relação ao modelo, em que $\boldsymbol{R}$ é matriz de covariância do ruído dos sensores reais, equivalente da matriz $\boldsymbol{Q}$.

Então, é definido o ganho de Kalman $\boldsymbol{K}$ pela Equação (3.21), que otimizará a inovação da varáveis de estado, quanto maior $\boldsymbol{K}$, mais confiável é a medição real.

$$\boldsymbol{S}_k = \boldsymbol{H}_k\bar{\boldsymbol{P}}_k\boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{3.20}$$

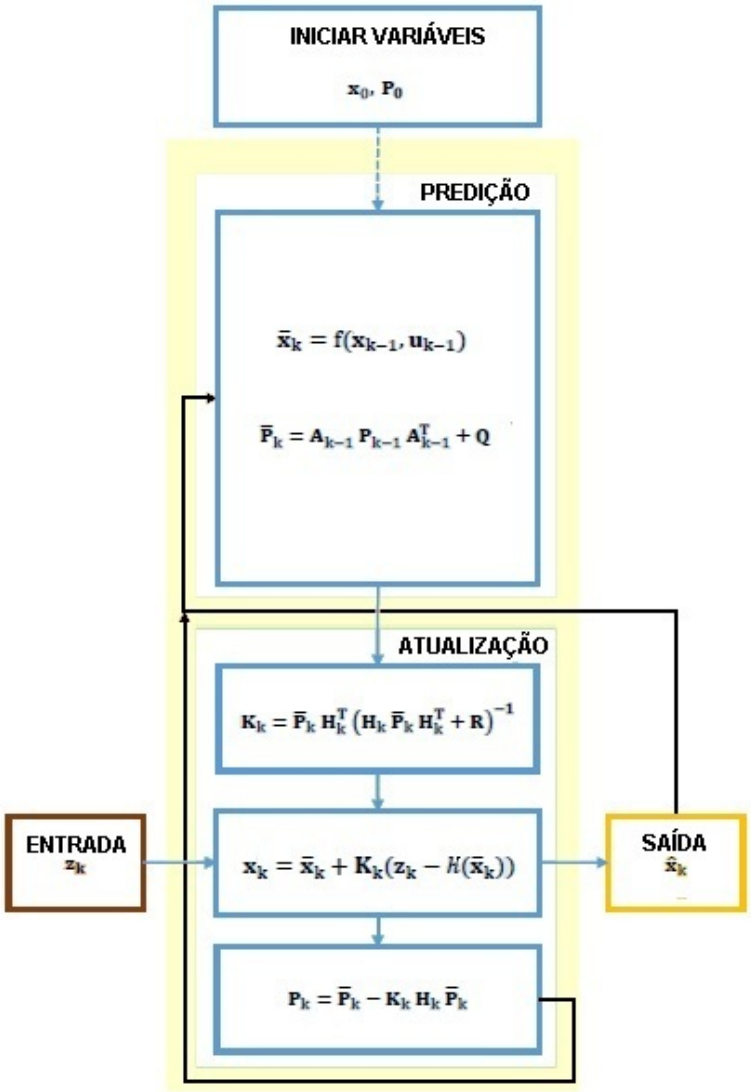
$$\boldsymbol{K}_k = \bar{\boldsymbol{P}}_k\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1} \tag{3.21}$$

Finalmente calcula-se a saída do filtro de Kalman, por meio da fusão entre o modelo matemático da predição e as medidas reais. O estado a *posteriori* é estimado, por meio da Equação (3.22). O novo valor de $\boldsymbol{x}_k$ estará entre $\bar{\boldsymbol{x}}_k$ e $\boldsymbol{z}_k$, e o ganho $\boldsymbol{K}_k$ definirá a proximidade em relação aos mesmos com base na probabilidade. Nesta etapa também é obtida a matriz de covariância do estado *a posteriori* pela Equação (3.23), em que $\boldsymbol{I}$ é a matriz identidade de mesma dimensão que $\boldsymbol{P}$. A Figura 13 apresenta um diagrama que resume o Filtro de Kalman.

$$\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k\tilde{\boldsymbol{y}} \tag{3.22}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\bar{\boldsymbol{P}}_k \tag{3.23}$$

Figura 13 – Algoritmo recursivo de Kalman



Fonte: Autoria Própria

3.3.3 Filtro de Kalman Aplicado a uma IMU

O filtro de Kalman é eficiente em uma unidade de medição inercial, conseguindo, por exemplo, dados mais confiáveis para as variáveis *Yaw*, *Pith* e *Roll* da convenção ZYX de Euler, a partir da fusão das medidas do acelerômetro, giroscópio e magnetômetro.

A Seção 3.2 apresenta a maneira de calcular estes ângulos. No entanto, utilizando sensores reais estas medidas estarão corrompidas por ruídos, que devem ser amenizados. Considere um eixo de um *frame* qualquer, a modelagem dinâmica de um sistema rotativo, envolve o ângulo θ que ele esta rotacionado e a velocidade angular atual $\dot{\theta}$, conforme a

Equação:

$$\theta_k = \theta_{k-1} + \dot{\theta}\Delta t \tag{3.24}$$

Para um melhor desempenho dinâmico do filtro a variável $\dot{\theta}$ é a saída real do giroscópio do mesmo eixo, o problema é que o giroscópio real apresenta *offset* na medida devido ao arraste, a velocidade angular *bias*, que é difícil de se modelar, uma vez que varia com o tempo [21].

A solução é introduzir este *offset* como mais uma variável do filtro de Kalman, $\dot{\theta}_b$ conforme a equação (3.25), deste modo a *bias* será estimada pelo filtro e compensada no próprio modelo dinâmico da predição [20], desta maneira a velocidade angular considerada no modelo será a diferença entre a medida do giro e a *bias*, que resulta em um sinal de média zero quando em repouso.

$$\begin{gathered}\bar{\boldsymbol{x}}_k = \boldsymbol{A}_k\boldsymbol{x}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k \\ \begin{bmatrix}\bar{\theta}_k \\ \bar{\dot{\theta}}_{bk}\end{bmatrix} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta_{k-1} \\ \dot{\theta}_{bk-1}\end{bmatrix} + \begin{bmatrix}\Delta t \\ 0\end{bmatrix}\dot{\theta}_k\end{gathered} \tag{3.25}$$

A matriz da covariância do erro *a priori* é definida pela Equação (3.26). A presença do Δt multiplicando a matriz $\boldsymbol{Q}$ torna a covariância do modelo simulado proporcional a duração do *loop* computacional, otimizando o algoritmo, uma vez que o erro do processo tende aumentar quando o tempo de amostragem cresce, aumentando a divergência entre o modelo simulado e o sistema real [21].

$$\begin{gathered}\bar{\boldsymbol{P}}_k = \boldsymbol{A}_k\boldsymbol{P}_{k-1}\boldsymbol{A}_k^T + \boldsymbol{Q} \\ \begin{bmatrix}\bar{P}_{00} & \bar{P}_{01} \\ \bar{P}_{10} & \bar{P}_{11}\end{bmatrix} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}P_{00_{(k-1)}} & P_{01_{(k-1)}} \\ P_{10_{(k-1)}} & P_{11_{(k-1)}}\end{bmatrix}\begin{bmatrix}1 & 0 \\ -\Delta t & 1\end{bmatrix} + \begin{bmatrix}Q_\theta & 0 \\ 0 & Q_{\dot{\theta}b}\end{bmatrix}\Delta t\end{gathered} \tag{3.26}$$

O erro residual é obtido pela equação (3.27), como a velocidade *bias* não observável a mesma não participa desta etapa, assim a matriz $\boldsymbol{H}$ a exclui. Assim o vetor $\boldsymbol{z}_k$ é apenas o valor do ângulo medido pelo sensor real.

$$\begin{gathered}\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\bar{\boldsymbol{x}}_k \\ \bar{\boldsymbol{y}}_k = \boldsymbol{z}_k - \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\bar{\theta}_k \\ \bar{\dot{\theta}}_{bk}\end{bmatrix}\end{gathered} \tag{3.27}$$

Os ganhos de Kalman são obtidos a partir das equações (3.28) e (3.29), a matriz $\boldsymbol{R}$ resume-se à covariância σ_θ^2 do único estado observável θ, assim são formulados os valores K_{0_k} e K_{1_k}, que estipulam os estados *a posteriori* das variáveis.

$$\boldsymbol{S}_k = \boldsymbol{H}_k \bar{\boldsymbol{P}}_k \boldsymbol{H}_k^T + \boldsymbol{R}$$
$$\boldsymbol{S}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \bar{P}_{00} & \bar{P}_{01} \\ \bar{P}_{10} & \bar{P}_{11} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \sigma_\theta^2 \tag{3.28}$$

$$\boldsymbol{K}_k = \bar{\boldsymbol{P}}_k \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1}$$
$$\begin{bmatrix} K_{0_k} \\ K_{1_k} \end{bmatrix} = \begin{bmatrix} \bar{P}_{00} & \bar{P}_{01} \\ \bar{P}_{10} & \bar{P}_{11} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \boldsymbol{S}_k^{-1} \tag{3.29}$$

Para completar o ciclo do algoritmo, atualiza-se o valor das variáveis de estado e da matriz covariância de erro. Baseado no erro residual e nos ganhos de Kalman, a estimativa *a posteriori* do ângulo θ_k, a velocidade $\dot{\theta}_{bk}$ e $\boldsymbol{P}_k$ são obtidas por meio das Equações (3.30) e (3.31)

$$\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$$
$$\begin{bmatrix} \theta_k \\ \dot{\theta}_{bk} \end{bmatrix} = \begin{bmatrix} \bar{\theta}_k \\ \bar{\dot{\theta}}_{bk} \end{bmatrix} + \begin{bmatrix} K_{0_k} \\ K_{1_k} \end{bmatrix} \tilde{\boldsymbol{y}}_k \tag{3.30}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \bar{\boldsymbol{P}}_k$$
$$\begin{bmatrix} P_{00_{(k)}} & P_{01_{(k)}} \\ P_{10_{(k)}} & P_{11_{(k)}} \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_{0_k} \\ K_{1_k} \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} \bar{P}_{00} & \bar{P}_{01} \\ \bar{P}_{10} & \bar{P}_{11} \end{bmatrix} \tag{3.31}$$

Este algoritmo pode ser utilizado com eficácia na obtenção de *Roll*, *Pitch* e *Yaw* de uma plataforma monitorada por sensores inerciais. Estes valores são importantes para o apontamento automático de um manipulador, já que a orientação da base faz parte da cadeia cinemática. No próximo capítulo será abordada a metodologia proposta para este processo.

4 Desenvolvimento do Projeto

Este capítulo apresenta a formulação do problema proposto por este trabalho, os materiais utilizados e os procedimentos realizados para o desenvolvimento do protótipo. Além da descrição do *hardware* e *software*, são apresentadas as funções de cada item para o projeto como um todo.

4.1 Proposta Teórica do Trabalho

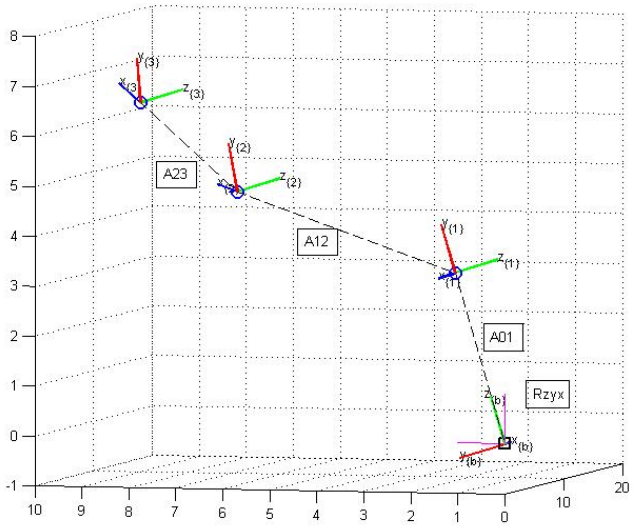
Como visto no Capítulo 2, para se controlar a direção de apontamento do efetuador utiliza-se a matriz de transformação homogênea que parametriza o *frame* final em relação à base. No caso do apontamento automático, a base está sujeita a rotações portanto ela não deve ser considerada o *frame* inercial como de costume na modelagem de robôs de base fixa. A solução é parametrizar a base e o restante da cadeia cinemática em relação ao sistema de coordenadas geográficas, abordadas no Capítulo 3, que, no caso, colocam o eixo z no sentido oposto ao da gravidade, x na direção Norte e y na direção Oeste.

Utilizando os Ângulos de Euler aplicados na matriz de coordenadas generalizadas da Equação (2.6), e as matrizes de Denavit-Hartenberg obtidas por meio da Tabela 1, obtêm-se o modelo de cinemática direta T_{final} obtida pela equação (4.1). De posse da orientação da base e dos ângulos das juntas, se conhece toda a disposição espacial do robô antropomórfico.

$$\mathbf{T}_{final} = \begin{bmatrix} & & & 0 \\ & \mathbf{R}_{zyx} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^{0}_{1}\mathbf{A}\ {}^{1}_{2}\mathbf{A}\ {}^{2}_{3}\mathbf{A} \tag{4.1}$$

Esta matriz homogênea é um dos pontos importantes do trabalho, sobre a qual as metodologias apresentadas nos Capítulos 2 e 3 serão aplicadas. A Figura 4.1 é uma representação virtual do processo de composição da equação (4.1), em que as matrizes de transformação homogênea de cada junta se encadeiam para obter o modelo final para o efetuador.

Figura 14 – Cadeia Cinemática do Manipulador Proposto.



Fonte: Autoria Própria

4.2 Materiais

Esta seção destina-se a apresentar os materiais utilizados na construção do robô manipulador utilizado no trabalho.

4.2.1 Arduino UNO

Figura 15 – Arduino UNO.

Fonte: https://eu.wikipedia.org/wiki/Arduino

O microcontrolador Arduino UNO (Figura 15) é um dos elementos mais importantes do projeto, sobre o qual recaem as funções de acionamento e controle dos motores das juntas, leitura e processamento dos dados dos sensores, aplicação do Filtro de Kalman e interface com o *Windows*. A Tabela 2 apresenta os dados técnicos.

Tabela 2 – Dados Técnicos Arduino UNO

Processador	ATmega328P
Input Voltage	7-12V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
DC Current per I/O Pin	20 mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

Fonte: https://www.arduino.cc/en/Main/ArduinoBoardUno

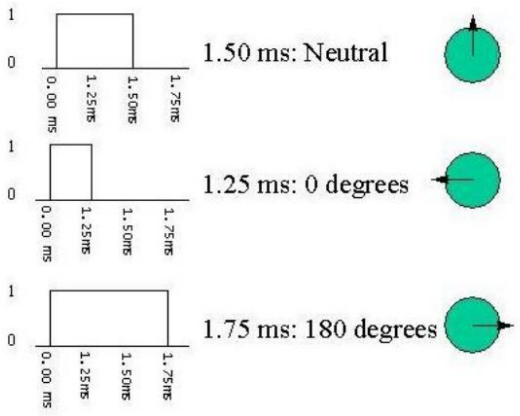
4.2.2 Servomotor HS-422

Figura 16 – HS-422

Fonte: http://www.myflymodelismo.com.br/

Para a atuação sobre as juntas do manipulador antropomórfico, foram utilizados três servomotores Hitec HS-422 (Figura 16), que apresenta controle de posição de 180 graus com 1 grau de resolução, e possui internamente um controlador em malha fechada com sinais de referência por meio de PWM (*Pulse Width Modulation*), conforme a Figura 17 exemplifica.

Figura 17 – Controle de posição do servomotor HS-422 por PWM



Fonte: http://www.seattlerobotics.org/guide/servos.html

Esse servomotor apresenta três fios, sendo dois de alimentação e um de comando, e pode ser facilmente utilizado com o Arduino por meio da biblioteca *Servo.h*. Outras informações podem ser vista na tabela abaixo:

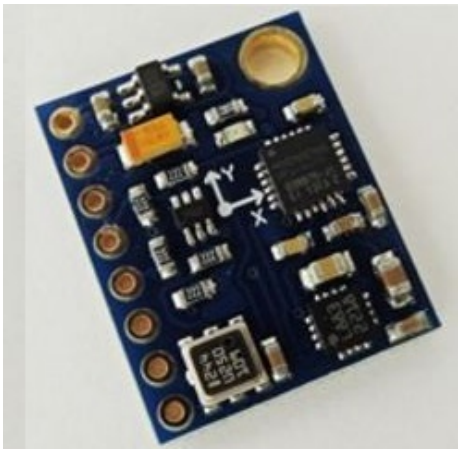
Tabela 3 – Dados Técnicos HS422

Tensão de operação	4.8 a 6V
Velocidade sem carga	0.21sec/60°(4.8V), 0.16sec/60°(6V)
Torque Nominal:	3.3kg.cm(4.8V), 4.1kg.cm(6V)
Corrente Elétrica	8mA parado ; 150mA sem carga.

Fonte: https://www.sparkfun.com/products/11884

4.2.3 IMU GY-87

Figura 18 – Módulo IMU GY-87

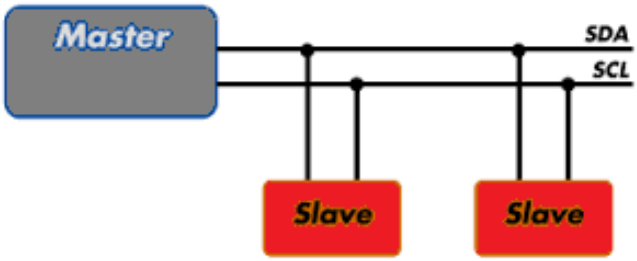


Fonte: Autoria Própria

A placa eletrônica IMU GY-87 possui instalados um acelerômetro, um magnetômetro e um giroscópio em uma única unidade de processamento, cada um com leituras em 3 eixos ortonormais, além de termômetro e barômetro. Os principais circuitos integrados deste dispositivo são: MPU6050, que contém um giroscópio e um acelerômetro, e o HMC5883L que contém o magnetômetro, ambos se comunicam por meio do protocolo de rede I2C.

O protocolo I2C descreve a comunicação serial a dois fios, SDA e SCL, patenteado pela empresa Philips. Esse protocolo é largamente utilizado para conectar periféricos de baixa velocidade. Cada dispositivo recebe um endereço no qual é conectado em paralelo ao barramento. O protocolo necessita de um dispositivo mestre para controlar os acessos e os processos de leitura e escrita na rede [10], papel desempenhado pelo Arduino neste projeto. A Figura 19 mostra um diagrama esquemático do protocolo I2C. Outras informações sobre o chip podem ser vista na Tabela 4.

Figura 19 – Rede I2C



Fonte: Autoria Própria

Tabela 4 – Informações dos Sensores

MPU6050	
Tensão de operação:	3-5V
Conversores ADC	16 bits
Comunicação	Protocolo padrão I2C
Faixa do Giroscópio:	±250, 500, 1000, 2000°/s
Faixa do Acelerômetro:	±2, ±4, ±8, ±16g
HMC5883L	
Tensão de operação:	2.16-3.6V
Conversores ADC	12 bits
Comunicação	Protocolo padrão I2C
Faixa do Magnetômetro	±1.16 gauss

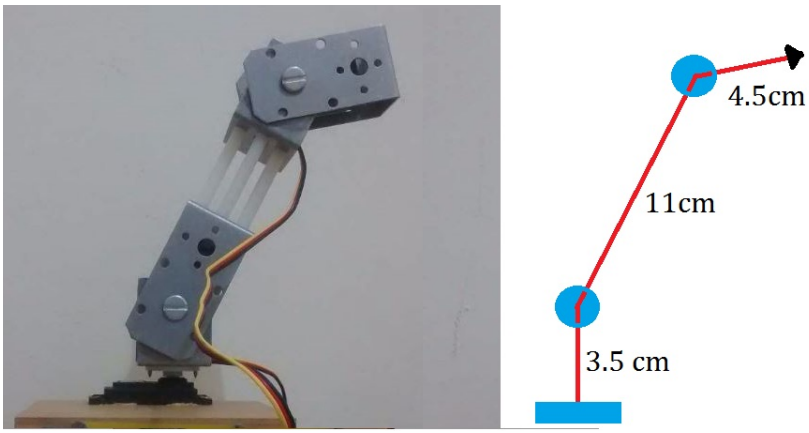
4.2.4 Protótipo Desenvolvido

Para a elaboração do protótipo foram utilizados 3 servos HS-422 conectados por suportes metálicos articulados formandos os elos (*links*) do robô. A posição central do

eixo de cada motor representa a posição da junta. Para estender os tamanhos dos elos, utilizaram-se prolongadores de PVC e a base foi confeccionada em madeira com formato cúbico e preenchida com areia para estabilizar o centro de gravidade do robô.

A Figura 20 apresenta uma visão de perfil do robô, ao lado de uma representação das distâncias entre as juntas. O sensor GY-87 foi afixado no centro da superfície do cubo, que será o local a ser considerado o *frame* da base.

Figura 20 – Protótipo de Manipulador RRR



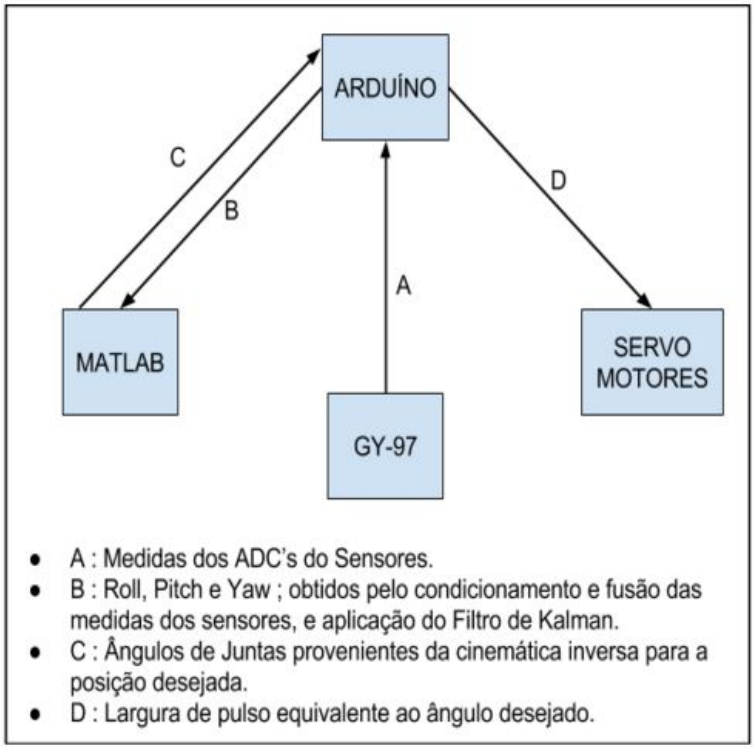
4.3 Descrição do Experimento

O objetivo central do experimento é controlar a posição e a orientação do efetuador com a base sujeita a rotações, com a restrição que as juntas se mantenham no intervalo entre -90 e 90 graus. O microcontrolador Arduino coleta as medidas dos sensores da placa GY-87 que está instalada no plano da base do manipulador: acelerômetro, giroscópio e magnetômetro, sendo 3 medidas de cada que correspondem aos 3 eixos canônicos (inerciais) da plataforma. As medidas são obtidas uma a uma por meio da rede I2C.

Por meio das equações apresentadas na Subseção 3.2, o Arduino calcula a orientação atual da base do robô. Com estes valores, e também com as medidas amostradas no giroscópio, efetuam-se três processos calculando filtros de Kalman simultâneos, descritos na Seção 3.3.3, para*Roll*, *Pitch* e *Yaw*, obtendo assim medidas confiáveis.

Um *software* desenvolvido em MATLAB® troca dados via comunicação serial com o Arduino e recebe a inclinação e aproamento da plataforma. Por meio da matriz T_{final} e MATLAB®, calculam-se os ângulos das juntas necessários para uma orientação desejada por meio da cinemática inversa apresentada na Seção 2.5.3. Ao final do cálculo dos ângulos das juntas, estes são enviados ao Arduino. O ciclo termina com o Arduino acionando os servomotores com o largura de pulso equivalente às posições angulares que as juntas necessitam atingir. A Figura 21 apresenta o diagrama resumido do processo descrito.

Figura 21 – Processo de atuação para apontamento automático



Fonte: Autoria Própria

5 Resultados

Neste capítulo serão apresentados os resultados experimentais dos estudos teóricos do trabalho por meio de simulações, análises e experimentos com o robô real (protótipo).

Antes de testar os algoritmos no protótipo, estes serão testados e calibrados por meio de simulação em MATLAB® na Seção 5.1, com o objetivo de analisar e otimizar o comportamento da cinemática envolvida. A Seção 5.2 apresenta os sensores utilizados, dentro da qual serão modelados e analisados estocasticamente. Na Seção 5.3 o filtro de Kalman será sintonizado e testado no sensor real. Para finalizar o capítulo, serão apresentados testes experimentais com o protótipo na Seção 5.4 .

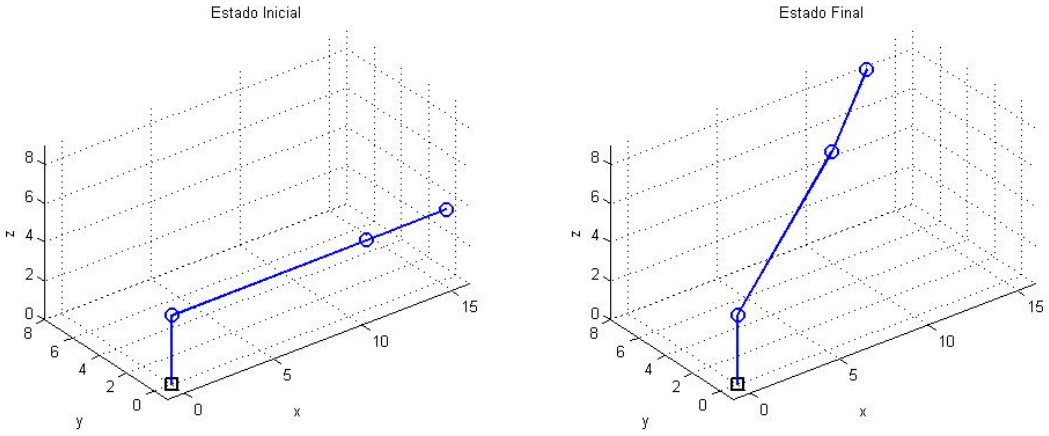
5.1 Simulação

Por meio do *software* MATLAB®, foi simulada a cinemática do protótipo, o qual está sujeito a mudanças rotacionais em sua base. O modelo da transformação homogênea é calculado pela equação (4.1). Substitui-se os valores simbólicos L_1, L_2 e L_3 pelas medidas reais dos elos do protótipo, dadas em centímetros, além da orientação angular da base, denominadas por α_b, β_b e γ_b, pelos valores angulares de interesse na simulação. O objetivo é levar o *frame* do efetuador a uma condição α_e e β_e pré-determinadas por meio da cinemática inversa proposta no trabalho.

5.1.1 Caso I

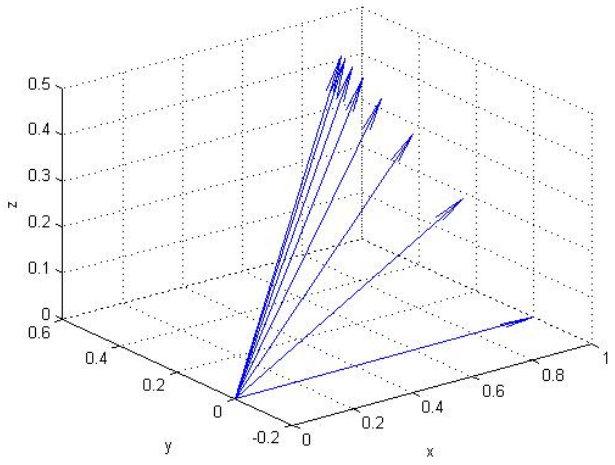
Com a base sem nenhuma alteração, o robô, a partir da posição inicial $\theta_1 = 0$, $\theta_2 = 0$ e $\theta_3 = 0$, teve colocada como objetivo a ser atingido a condição de apontamento $\alpha_e = \pi/4$ *e* $\beta_e = -\pi/4$. O critério de parada foi o Erro Quadrático Médio menor que 0,0005 radianos, e a taxa de aprendizagem foi calibrada como $\eta = 0.4$. O resultado para as variáveis de juntas foi, aproximadamente, $\theta_1 = 0.5$ rad, $\theta_2 = 0.25$ rad e $\theta_3 = 0.25$ rad, que produz as posições objetivo angulares com um erro menor que 0,05 radianos, e o resultado está exposto na Figura 22.

Figura 22 – Simulação 1



A Figura 23 apresenta a evolução da orientação do vetor x do *frame* do efetuador ao longo da simulação do caso I, demonstrando o funcionamento do método do gradiente descendente, no qual, com poucas iterações (apenas 8) o algoritmo convergiu, mostrando sua eficiência.

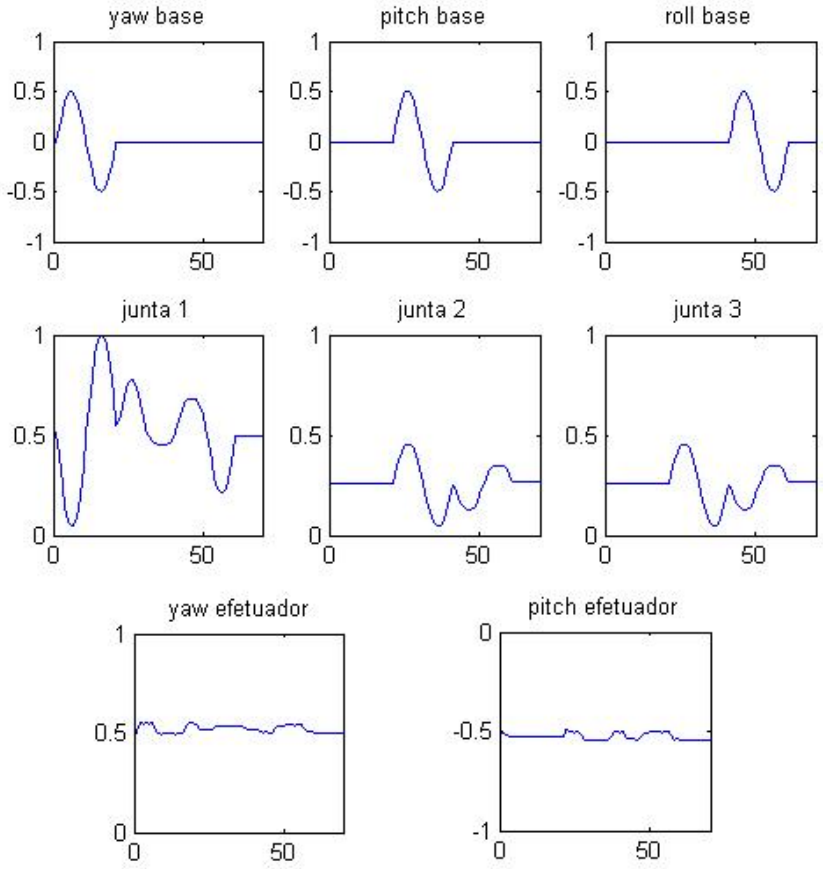
Figura 23 – Evolução do apontamento do *end-effector*



5.1.2 Caso II

Para testar o comportamento cinemático quando a base sofre variações angulares, o mesmo algoritmo do caso I foi utilizado, mas com os ângulos de Euler da base sofrendo oscilações em sequência. Por meio da Figura 24 que exibe os resultados do experimento, nota-se que o valor do apontamento sofre pequenas variações devido ao processo de otimização.

Figura 24 – Simulação com variação da base. Eixo x: iteração / Eixo y: radianos



O algoritmo escrito em MATLAB® que serviu de base para esta simulação está presente no Apêndice B. É possível se concluir que a proposta matemática do trabalho funcionou adequadamente, e a metodologia utilizada pode ser empregada em um teste prático. A otimização para o critério de parada com Erro Quadrado Médio ζ menor que 5×10^{-4} obteve acertos nas direções de *Pitch* e *Yaw* com acurácia superior a 0,05 radianos.

5.1.3 Otimização dos Parâmetros do Gradiente Descendente

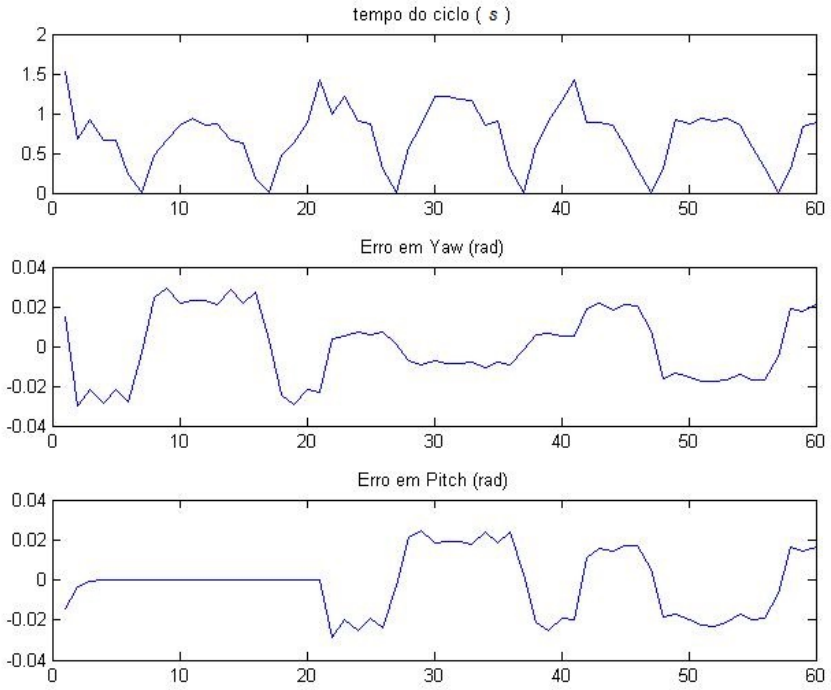
O algoritmo de cinemática inversa apresenta dois parâmetros que influenciam a dinâmica e a precisão do manipulador: o critério de parada do erro ζ e a taxa de aprendizado η.

Uma taxa de aprendizado com maiores intensidades proporcionam uma convergência mais rápida em direção ao destino, no entanto pode ocorrer um efeito "zigue-zague" em torno do ponto de mínimo da função de erro, não convergindo para a região menor que δ. Quanto menor η mais ciclos do algoritmo do gradiente descendente serão necessários, aumentando o tempo de convergência.

De acordo com o objetivo do projeto, estes parâmetros devem ser escolhidos apropriadamente, de modo que haja compromisso entre o tempo de convergência e erro máximo permitido. No caso do presente trabalho, o objetivo é acionar os servomotores para a configuração necessária de maneira rápida e com um erro satisfatório. Dessa maneira, foi feita uma calibragem, acompanhando o erro e o tempo de ciclo para diferentes parâmetros. As simulações realizam o mesmo procedimento do caso II.

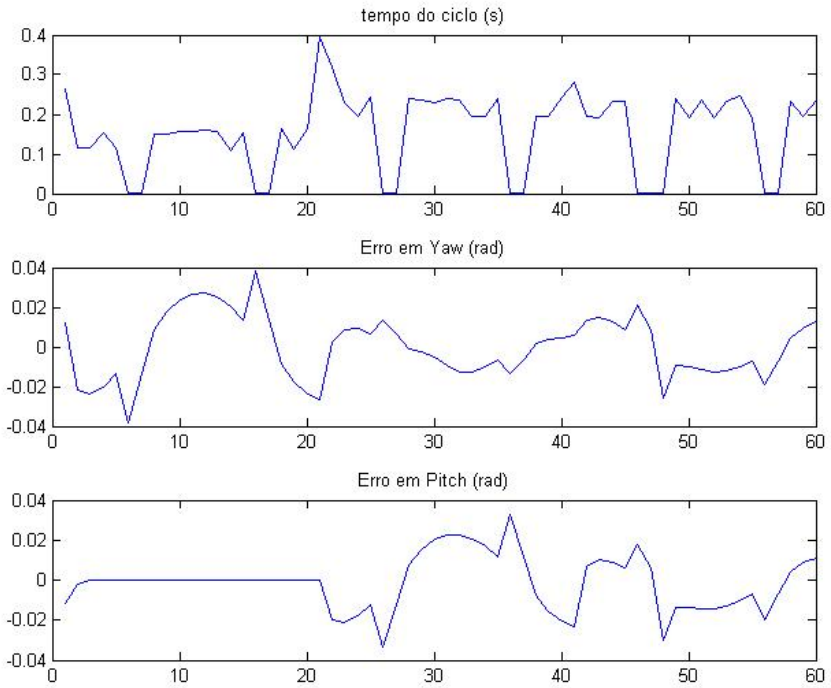
Por exemplo, para os parâmetros usados nos casos I e II, $\eta = 0.4$ e $\zeta = 5 \times 10^{-4}$ ocorreu a situação apresentada em 60 iterações na Figura 25, na qual o tempo de convergência passa, em alguns momentos de um segundo, e isto inviabiliza o controle em tempo real.

Figura 25 – Tempo e Erro para $\eta = 0.4$ *e* $\zeta = 5e^{-4}$



Um resultado melhor foi obtido ao aumentar a taxa de aprendizado para $\eta = 0.85$, e a tolerância do Erro Quadrado Médio, para $\zeta = 8 \times e^{-4}$. Os resultados são evidenciados na Figura 26.

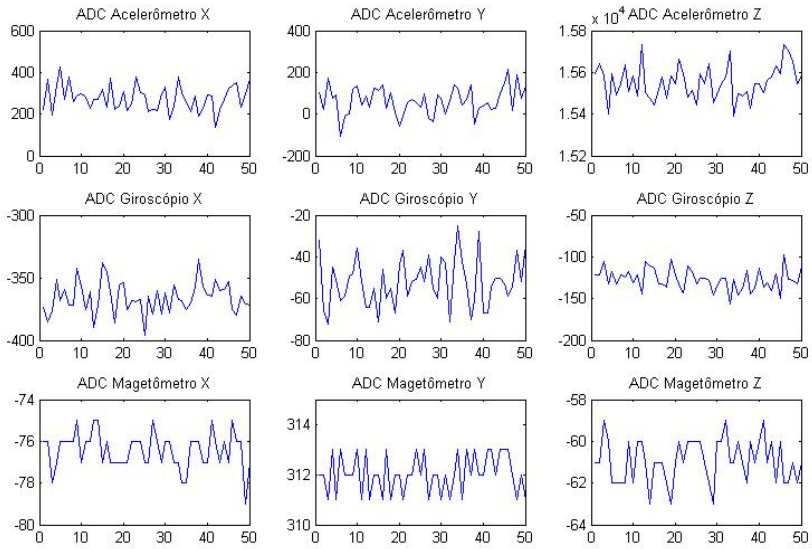
Figura 26 – Tempo e Erro para $\eta = 0.85$ e $\zeta = 8e^{-4}$



5.2 Modelagem dos Sensores

O Arduino recebe valores de tensão provenientes do IMU GY-87. A Figura 27 contém 50 amostras de cada medida de interesse desse sensores, os quais estão em repouso, de modo a evidenciar o ruído aleatório e a necessidade de compensação dos *offsets* via filtragem.

Figura 27 – Saídas ADC do GY-87



As Equações da Seção 3.2 utilizam as leituras do GY-87 para estimar ângulos, mas para usar as equações é necessário condicionar estas medidas angulares, compensando seus *offsets* e o ganho do conversor analógico digital (ADC).

Tabela 5 – Condicionamento dos Sensores

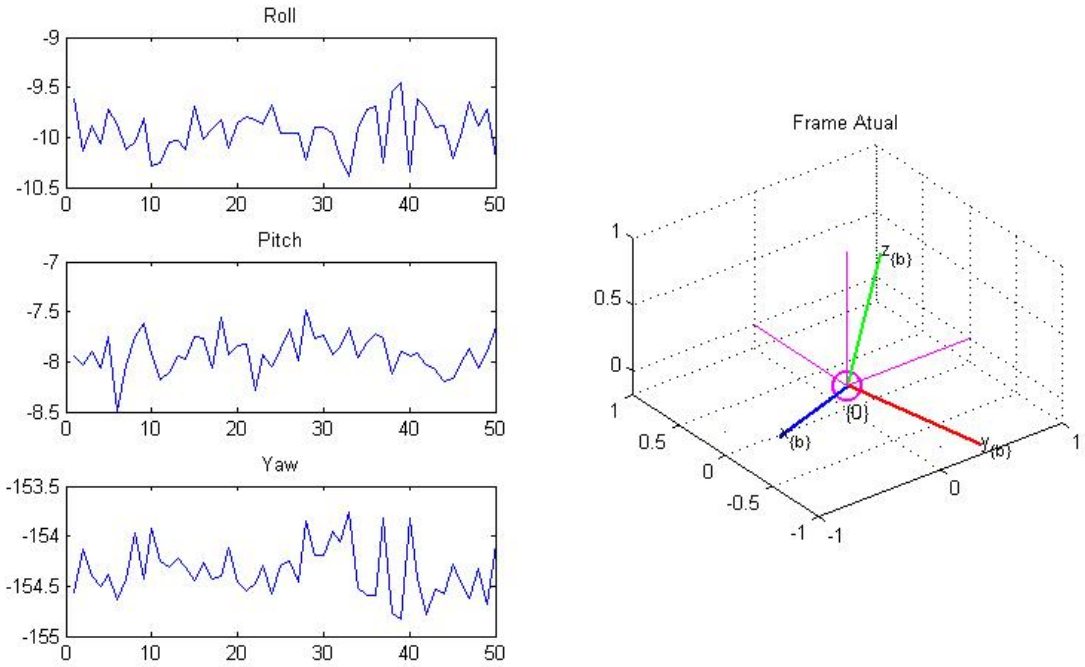
ADC	Vmax	Vmin	Vos	Ganho
AcX	16.6k	-16.2k	200	16.4k
AcY	16.7k	-16.3	200	16.5k
AcZ	15.8k	-17.4k	-800	16.6k
MgX	-74	-668	-371	297
MgY	615	5	305	310

A Tabela 5 contém os valores máximos e mínimos com relação à atuação exclusiva da força gravitacional e do campo magnético terrestre, e por meio deles são calculados os *offsets* (V_{os}) e o ganho do ADC. Os ganhos podem ser interpretados como sensibilidades, equivalentes a uma unidade de força gravitacional ou uma unidade do campo magnético terrestre, assim as medidas se tornam proporcionais aos ganhos. Os valores condicionados podem ser obtidos através da Tabela 5 e da Equação (5.1).

$$V_{final} = (V_{adc} - V_{os})/ganho \tag{5.1}$$

Os valores são aplicados nas fórmulas da Seção 3.2, de modo a calcular os ângulos γ, β e α das rotações *Roll*, *Pitch* e *Yaw*. A Figura 28 apresenta 50 amostras em uma determinada orientação em repouso.

Figura 28 – Teste da Estimação de Orientação



Antes de aplicar os métodos de filtragem, é necessário estudar estocasticamente essas medições. Pode-se definir a normalidade estatística dessa população pelo teste de Shapiro-Wilk, descrito na seção 3.3.1. Foi realizado a partir da função do MATLAB® "swtest.m", que de acordo com vetor de amostras retorna os valores W e p. Foram utilizadas as funções "hist.m" e "std.m" que efetuam um histograma e cálculo do desvio padrão respectivamente, a Figura 29 apresenta estes parâmetros descritos para as mesmas amostras da Figura 28.

De acordo com a Figura 29 os três ângulos obtidos de sensores reais possuem uma distribuição aproximadamente normal, os histogramas agruparam as amostras em 6 intervalos, e mesmo com apenas 50 amostras é possível perceber características da distribuição Gaussiana. O teste de Shapiro-Wilk foi bem sucedido para as 50 amostras.

O procedimento foi realizado novamente para 200 novas amostras com o GY-87 na mesma posição, o resultado apresentado na Figura 30 confirma as expectativas, deixando clara a tendência a Normalidade do ruído.

Figura 29 – Parâmetros Estocásticos de γ, β *e* α, 50 amostra

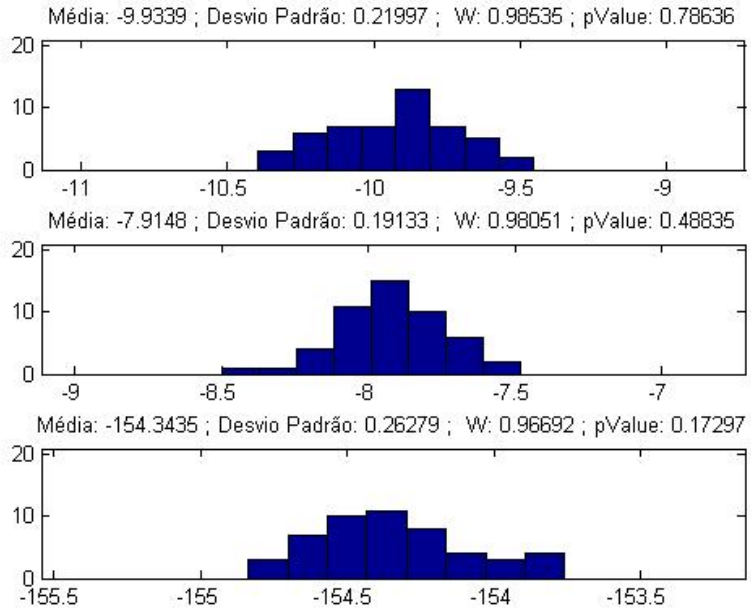
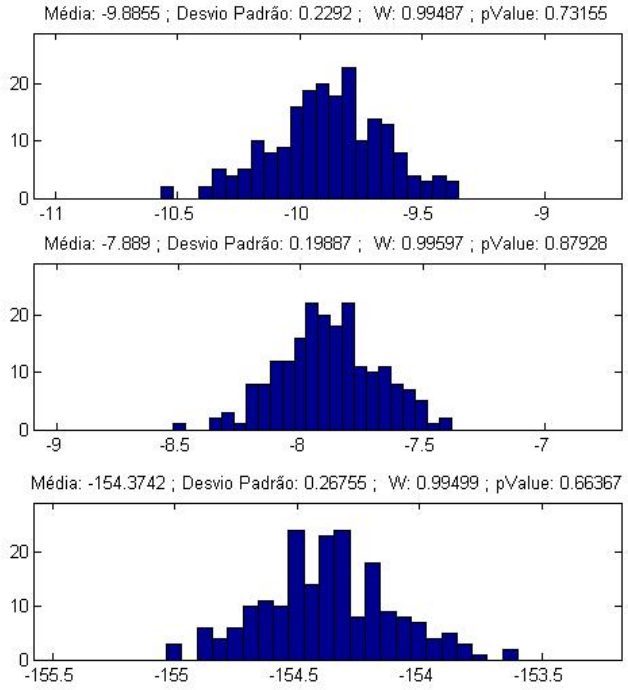


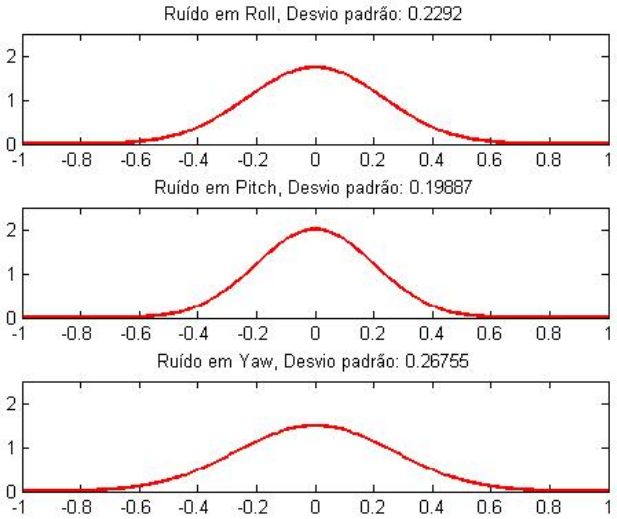
Figura 30 – Parâmetros Estocásticos de γ, β *e* α, 200 amostras



Pode-se concluir, a partir desses resultados, que a modelagem dos sensores fornece medidas de inclinação muito próximas da realidade na média, e as equações, que são

baseadas em medições reais, estão fornecendo valores contaminados por ruídos aproximadamente Gaussianos, que correspondem matematicamente às curvas de distribuição de probabilidade da Figura 31, utilizando a Equação (3.12), página 32.

Figura 31 – Erros das Medidas



5.3 Filtragem do Ruído

Conforme pode-se ver na Figura 28, é necessário amenizar o ruído para uma estimativa mais precisa dos ângulos e para melhorar o acionamento dos servomotores, já que de acordo com a posição da base serão gerados os valores das juntas para manter a orientação do *end-effector*. As equações do algoritmo recursivo do filtro de Kalman para uma IMU, apresentadas na seção 3.3.3, possuem duas matrizes a serem calibradas que são as covariâncias $\boldsymbol{R}$ e $\boldsymbol{Q}$.

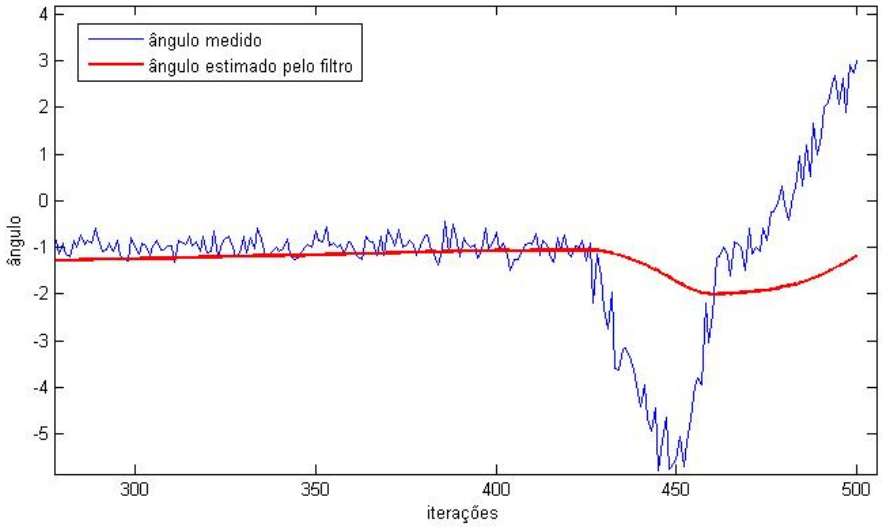
5.3.1 Calibração do filtro de Kalman

As covariâncias da medição real $\boldsymbol{R}$, de *Yaw, Pitch e Roll* já foram estimadas por meio dos desvios padrões calculados na seção 5.2. Para otimizar o filtro de Kalman, resta modificar os parâmetro da matriz $\boldsymbol{Q}$. As covariâncias do modelo virtual do sistema definirão o quanto o ruído será amenizado, e a velocidade que o filtro aprende a reconhecer mudanças do estado real do sistema.

Se a covariância do ruído do processo $\boldsymbol{Q}$ for pequena, isto significa que o modelo matemático representa bem o sistema e que os erros das medidas reais são pouco significantes, resultando em uma filtragem excessiva, o que pode tornar a resposta lenta. Por exemplo, digamos que o sistema não conta com giroscópios para quantificar as velocidades angulares do modelo estimado. Se o filtro possui a matriz $\boldsymbol{Q}$ pouco expressiva, a saída se

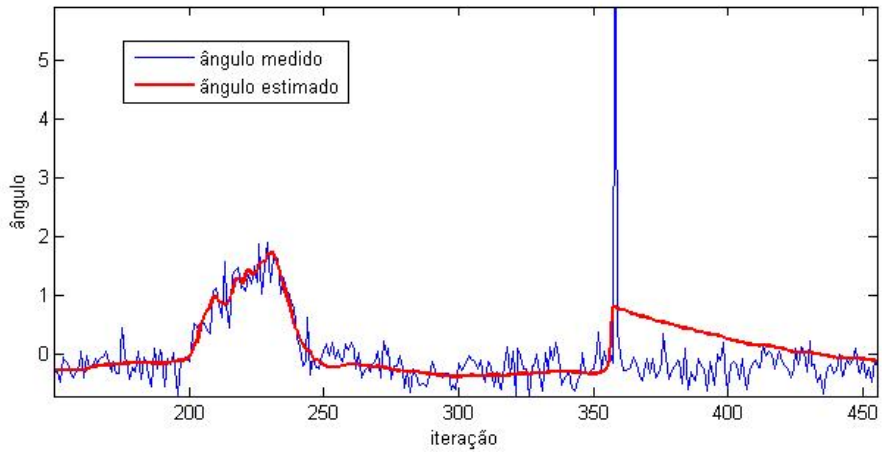
torna lenta e não acompanha a dinâmica real. A Figura 32 demonstra esta ocorrência, na qual foi utilizada a rotação *Roll* como experimento, com $\sigma_1 = 0.0003,\ \sigma_2 = 0.003$ e $u_k = 0$.

Figura 32 – Filtro com a matriz $\boldsymbol{Q}$ subdimensionada e sem uso do giroscópio



Utilizando a medida do giroscópio no eixo x da IMU, e os mesmo valores para $\boldsymbol{Q}$ na predição, obteve-se a característica apesentada na Figura 33, enfatizando a importância da velocidade angular como entrada de controle para reconhecimento das dinâmicas transitórias do sistema.

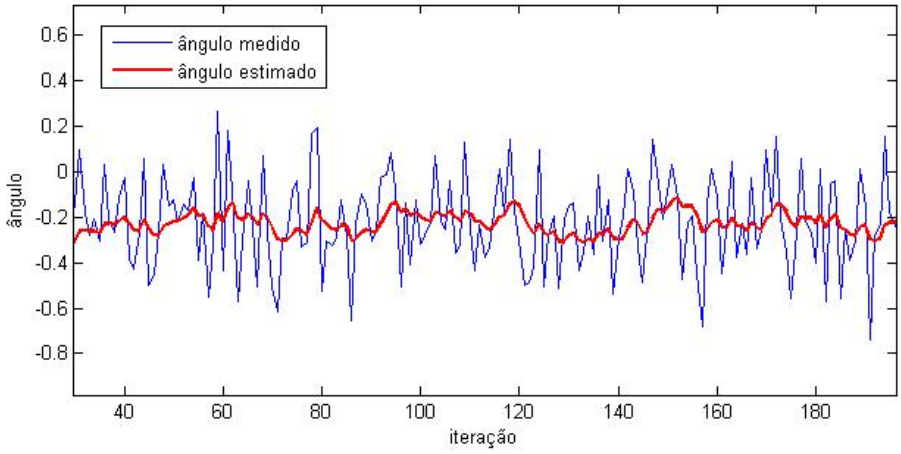
Figura 33 – Filtro com a matriz $\boldsymbol{Q}$ subdimensionada e com uso do giroscópio



O subdimensionamento da matriz de covariância ainda é um problema, pois, se o estado do modelo divergir da realidade, seja por qualquer motivo, haverá uma demora significativa para que convirja novamente, o que aconteceu próximo da iteração 350 da Figura 33, após a ocorrência de um breve distúrbio.

Se o projetista do filtro considerar um modelo da predição pouco confiável, fazendo a matriz $\boldsymbol{Q}$ proporcionalmente grande, o sensor real será considerado impreciso pois seus ruídos são significativos, e o filtro será altamente influenciado pela variação das medidas, o que irá abrandar pouco o ruído, como na Figura 34 aonde usou-se $\sigma_1 = 0.03,\ \sigma_2 = 0.3$.

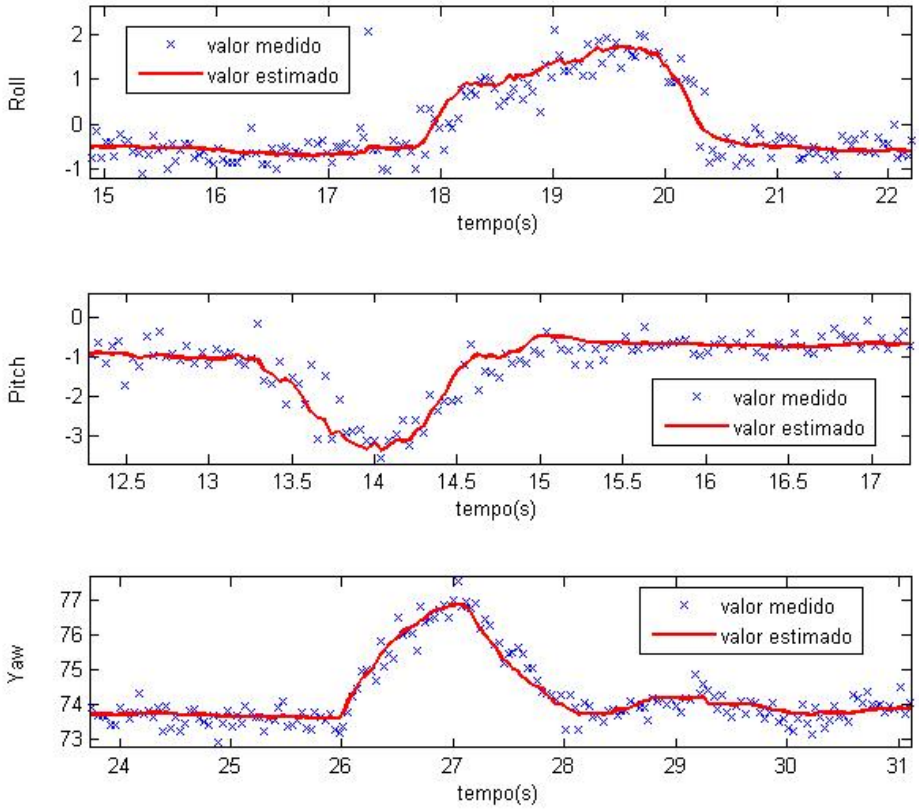
Figura 34 – Filtro com a matriz Q superdimensionada



A matriz $\boldsymbol{Q}$ deve ser escolhida de acordo com o interesse do projeto, dependendo do que realmente importa, se é identificar o valor com grande exatidão em regime estacionário ou acompanhar a dinâmica envolvida. No caso da IMU, o ideal é acompanhar o mais próximo possível a dinâmica do sistema mantendo uma boa filtragem do ruído.

Por se tratarem de testes visuais, e os três ângulos medidos possuírem desvios padrões próximos, para facilitar foi escolhido para *Roll, Pitch e Yaw* os valores dos desvios padrão para os modelos virtuais iguais, $\sigma_1 = 0.003,\ \sigma_2 = 0.03$, não havendo diferença perceptível na qualidade entre eles, conforme Figura 35.

Figura 35 – Resultado final do filtro

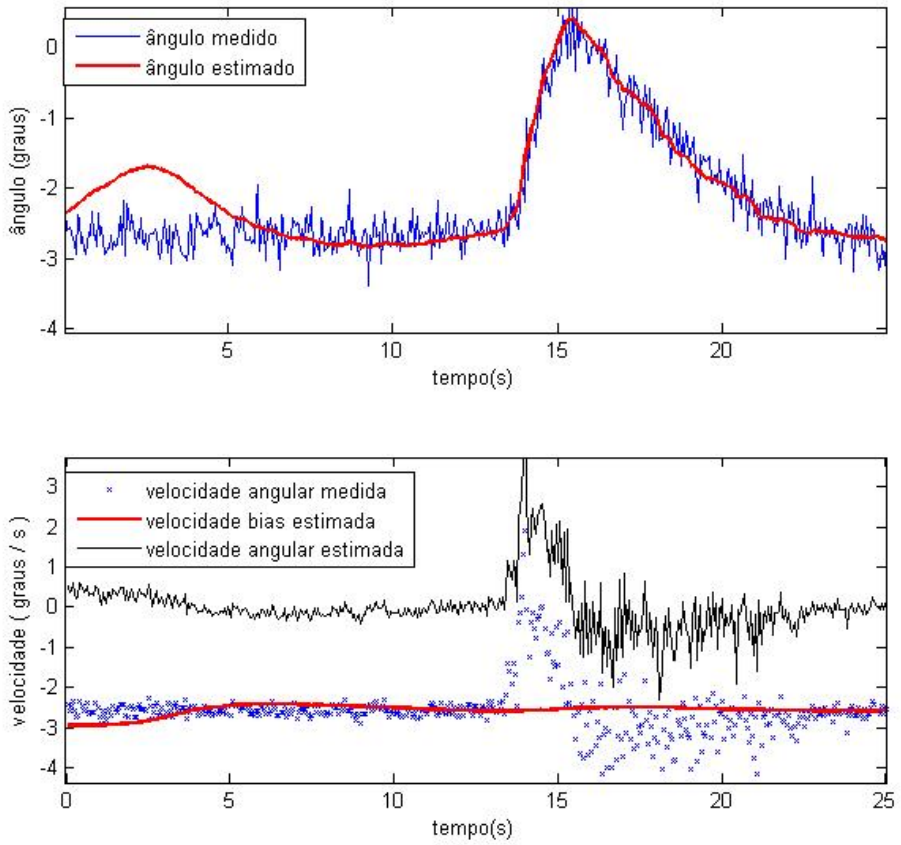


5.3.2 Reconhecimento de Bias da Velocidade

Conforme apresentado na subseção 3.3.3, este tipo de aplicação do filtro de Kalman possui como uma das variáveis de estado a velocidade do *bias* $\dot{\theta}_b$, referente ao *offset* do giroscópio. Sendo que a cada iteração o ganho do filtro de Kalman atualiza esta variável em direção ao valor real deste arraste.

A Figura 36 apresenta o efeito das fases deste processo. Com o GY-87 em repouso, no início o modelo simulado divergiu da realidade, pois o *bias* ainda estava sendo estimado e distante do real. Após aproximadamente 10 segundos, o *bias* pôde ser estabilizado, e a média da velocidade angular total utilizada na predição (diferença entre a medida do giroscópio e a *bias*), tendeu a zero.

Figura 36 – Dinâmica do *Bias*.



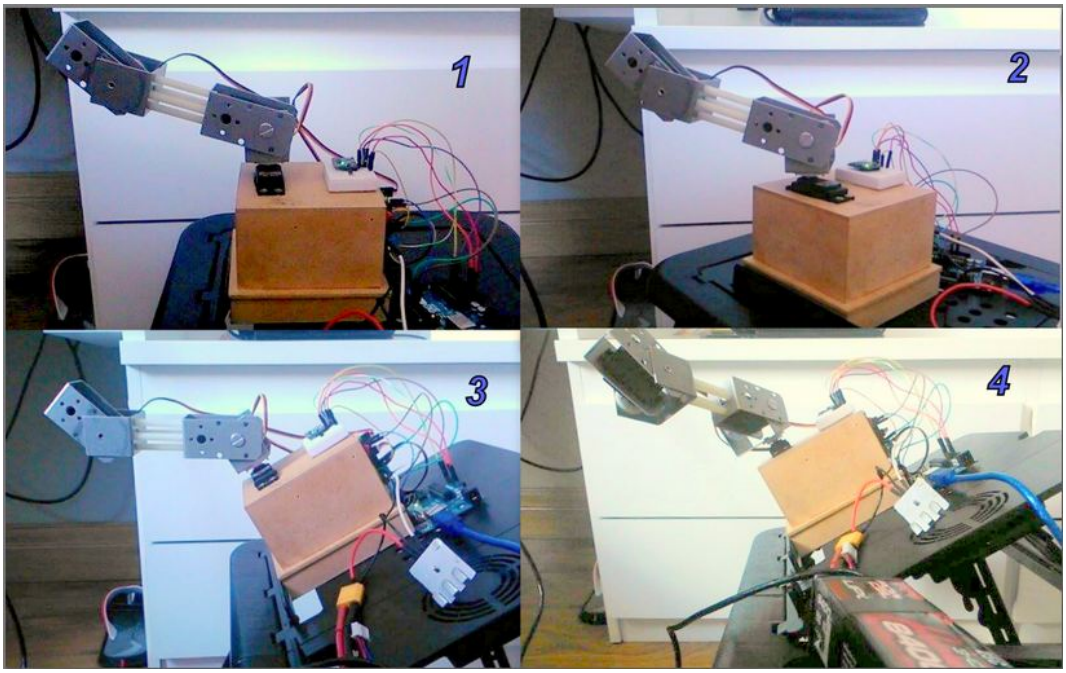
A solução para amenizar este efeito indesejado no início do algoritmo é iniciar o vetor $\bar{x}_0$ com as primeiras medidas do sensor real se deram com o sistema em repouso. Ao utilizar $\bar{\theta}_0$ como a medida angular real e $\bar{\dot{\theta}}_{b0}$ como a saída do giroscópio, isto posicionou estas variáveis próximas do estado final em regime permanente.

5.4 Resultados Experimentais

As características construtivas dos equipamentos utilizados limitaram a resposta temporal do robô. Por isso, não foi possível controlar o protótipo em tempo real, pois o tempo entre a aquisição do estado rotacional da base e o acionamento correspondente dos servos inviabilizou este processo.

A comprovação experimental da proposta do trabalho foi realizada por meio de avaliação visual e cinemática em regime permanente. Isto é, após uma mudança no *frame* da base, aguardou-se o atraso e regime transitório mecânico. O protótipo foi colocado em repouso sobre diversas condições angulares da base, conforme Figura 37, em cada posição o objetivo foi apontar a ferramentar o efetuador para a direção $Pitch_e = 60\,^\circ$ e $Yaw_e = 0\,^\circ$.

Figura 37 – Resultado Experimental



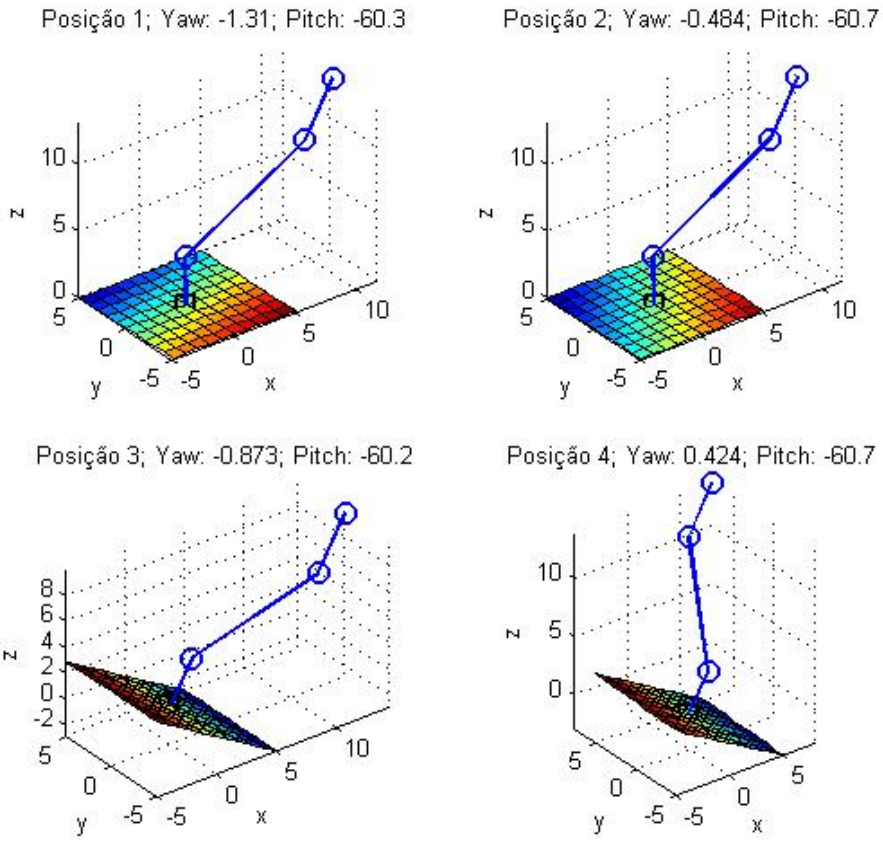
Além de um registro visual, foi coletado o estado os ângulos $Roll_b$, $Pitch_b$ e Yaw_b da base e os ângulos de acionamento do servos. Calculou-se $Pitch_e$ e Yaw_e por meio do modelo de cinemática direta, fundindo as Equações (4.1) e (2.7). Estas informações estão contidas na Tabela 6.

Tabela 6 – Apresentação dos Resultados

Posição	$Roll_b$	$Pitch_b$	Yaw_b	θ_1	θ_2	θ_3
1	-0.61°	-0.3°	-3.4°	1°	30°	30°
2	-0.8°	-0.2°	40.7°	-42°	30°	30°
3	-0.56°	29.8°	-2°	1°	45°	45°
4	11.5°	27.1°	20.4°	70°	44°	44°

A Figura 38 apresenta virtualmente as posições do experimento juntamente com as valores de *Yaw e Picth* do efetuador para estas configurações, mesmo com a baixa resolução dos servomotores, desprezando imperfeições mecânicas, o erro teórico foi satisfatório.

Figura 38 – Resultado Experimental - Comprovação



6 Conclusões e Trabalhos Futuros

A partir dos resultados obtidos e expostos no Capítulo 5, observa-se que a integração de conhecimentos teóricos na qual se baseou este trabalho foi adequada para a realização do apontamento automático do manipulador robótico. Mesmo com a utilização de equipamentos pouco precisos para a calibragem dos sensores e do robô, o método adotado foi capaz de corrigir em a inclinação da base para apontamento do efetuador, o que foi atestado por simulação, e de maneira visual com o robô real.

Com relação à comprovação experimental, objetivo foi parcialmente alcançado, pois mesmo com o sucesso obtido no apontamento em regime permanente, o protótipo não acompanhou variações rápidas da base em tempo real devido ao elevado tempo entre a aquisição de medidas dos sensores e o acionamento correspondente dos servomotores. O maior impacto nesse experimento foi tempo de processamento da cinemática inversa.

O tempo de processamento do algoritmo implementado em MATLAB®, juntamente com os processos de comunicação e filtragem no Arduino tornam o protótipo construído incapaz de reagir a mudanças bruscas da base. Para contornar esse problema, um algoritmo de cinemática inversa otimizado, estimação de orientação e de controle dos servomotores devem ser embarcados em um único aparelho dedicado. Os dispositivos poderiam ser um *Raspberry Pi II* ou uma *Tiva C Series*. Para comprovar os resultados futuros pode ser utilizado uma segundo dispositivo *IMU* no efetuador, medindo a orientação *frame* localmente.

REFERÊNCIAS

[1] DEBRUIN, J. Control systems for mobile satcom antennas. IEEE Control Systems Magazine, p. 86–101, Feb. 2008.

[2] FACINA. Alex Rodrigo. Estudos Preliminares sobre Antenas de apontamento automático e base estabilizada. Monografia. UFJF.

[3] SICILIANO, Bruno et al. Robotics: modelling, planning and control. Springer Science & Business Media, 2009.

[4] ROSÁRIO, João Maurício. Princípios de mecatrônica. Pearson Prentice Hall, 2006.

[5] CRAIG, John J. Introduction to robotics: mechanics and control. Upper Saddle River: Pearson Prentice Hall, 2005.

[6] SPONG, Mark W.; HUTCHINSON, Seth; VIDYASAGAR, Mathukumalli. Robot modeling and control. New York: Wiley, 2006.

[7] OLIVI, Leonardo Rocha. Manipuladores Robôtivos. Universidade Federal De Juiz De Fora. 2014.

[8] KALMAN, Rudolph Emil. A new approach to linear filtering and prediction problems. Journal of Fluids Engineering, v. 82, n. 1, p. 35-45, 1960.

[9] Honeywell 3-Axis Digital Compass IC HMC5883L datasheet

[10] INVENSENSE, MPU-6000 and MPU-6050: Product Specification, Revision 3.3 Disponível em: <http://www.gotronic.fr/pj-956.pdf> Acesso em: 12 de agosto de 2015.

[11] Costa, E. B. (2012). Algoritmos de controle aplicados a estabilização do voo de um quadrotor, Master's thesis, Universidade Federal de Juiz de Fora.

[12] LOPES, A. M. Robótica Industrial, Universidade do Porto, 2001.

[13] CHOCRON, Olivier. Handout No 2, Course 2.05, October. 2000.

[14] Denavit, Jacques; Hartenberg, Richard Scheunemann (1955). "A kinematic notation for lower-pair mechanisms based on matrices". Trans ASME J. Appl. Mech 23: 215–221.

[15] Karlsson, R.; Norrlof, M., "Bayesian position estimation of an industrial robot using multiple sensors,"in Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on , vol.1, no., pp.303-308 Vol.1, 2-4 Sept. 2004

[16] ORTIZ-SALAZAR, Manuel et al. IMU-Based Trajectory Generation and Modelling of 6-DOF Robot Manipulators. In: Mechatronics, Electronics and Automotive Engineering (ICMEAE), 2015 International Conference on. IEEE, 2015. p. 181-186.

[17] NETO, Pedro; PIRES, J. Norberto; MOREIRA, A. Paulo. Accelerometer-based control of an industrial robotic arm. In: Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on. IEEE, 2009. p. 1192-1197.

[18] Renk, E.L.; Rizzo, M.; Collins, W.; Lee, F.; Bernstein, D.S., "Calibrating a triaxial accelerometer-magnetometer - using robotic actuation for sensor reorientation during data collection,"in Control Systems, IEEE , vol.25, no.6, pp.86-95, Dec 2005

[19] KIM, Phil. Kalman filter for beginners: with MATLAB examples. CreateSpace, 2011.

[20] Pengfei Gui; Liqiong Tang; Mukhopadhyay, S., "MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion,"in Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference on , vol., no., pp.2004-2009, 15-17 June 2015 doi: 10.1109/ICIEA.2015.7334442

[21] LAUSZUS, Kristian. A practical approach to Kalman filter and how to implement it. 2012. APA

[22] TRIMPE, Sebastian; D'ANDREA, Raffaello. Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010. p. 2630-2636.

[23] SHAPIRO, Samuel Sanford; WILK, Martin B. An analysis of variance test for normality (complete samples). Biometrika, p. 591-611, 1965.

[24] OZYAGCILAR, Talat. Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. Freescale semiconductor, AN, v. 4248, 2012.

[25] Sá, C. E. A. de (1996). Implementação de Métodos Númericos para a Resolução do Problema Cinemático Inverso de Manipuladores Robóticos - Com Ênfase em Controle de Posição". Master Degree Thesis, Unicamp.

[26] WILSON, Jon S. Sensor technology handbook. Elsevier, 2004.

[27] Ming Wen; Zhang Luo; Weihui Wang; Sheng Liu, "A characterization of the performance of MEMS vibratory gyroscope in different fields,"in Electronic Packaging Technology (ICEPT), 2014 15th International Conference on , vol., no., pp.1547-1551, 12-15 Aug. 2014

[28] RODDY, D. Satellite Communications. [S.l.]: McGraw-Hill, 2006.

[29] BUSS, Samuel R. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation, v. 17, n. 1-19, p. 16, 2004.

A Tabelas de Shapiro-Wilk

A.1 Coeficientes a_i

Coeficientes necessários para a equação 3.13

Tabela 7 – Shapiro-Wilk: Coeficientes a_i: Parte I

n	5	6	7	8	9	10	11	12	13	14	15
i											
1	0.66	0.64	0.62	0.61	0.59	0.57	0.56	0.55	0.54	0.53	0.52
2	0.24	0.28	0.30	0.32	0.32	0.33	0.33	0.33	0.33	0.33	0.33
3	0.00	0.09	0.14	0.17	0.20	0.21	0.23	0.23	0.24	0.25	0.25
4			0.00	0.06	0.09	0.12	0.14	0.16	0.17	0.18	0.19
5					0.00	0.04	0.07	0.09	0.11	0.12	0.14
6							0.00	0.03	0.05	0.07	0.09
7									0.00	0.02	0.04
8											0.00

Tabela 8 – Shapiro-Wilk: Coeficientes a_i: Parte II

n	16	17	18	19	20	21	22	23	24	25	26	27
i												
1	0.51	0.50	0.49	0.48	0.47	0.46	0.46	0.45	0.45	0.45	0.44	0.44
2	0.33	0.33	0.33	0.32	0.32	0.32	0.32	0.31	0.31	0.31	0.30	0.30
3	0.25	0.25	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.25	0.25	0.25
4	0.19	0.20	0.20	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.22	0.22
5	0.14	0.15	0.16	0.16	0.17	0.17	0.18	0.18	0.18	0.18	0.18	0.18
6	0.10	0.11	0.12	0.13	0.13	0.14	0.14	0.15	0.15	0.15	0.16	0.16
7	0.06	0.07	0.08	0.09	0.10	0.11	0.12	0.12	0.12	0.13	0.13	0.13
8	0.02	0.04	0.05	0.06	0.07	0.08	0.09	0.09	0.10	0.10	0.11	0.11
9		0.00	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.08	0.09	0.09
10				0.00	0.01	0.03	0.04	0.05	0.05	0.06	0.07	0.07
11						0.00	0.01	0.02	0.03	0.04	0.05	0.05
12								0.00	0.01	0.02	0.03	0.04
13										0.00	0.01	0.02
14												0.00

Tabela 9 – Shapiro-Wilk: Coeficientes a_i: Parte III

n	28	29	30	31	32	33	34	35	36	37	38
i											
1	0.43	0.43	0.43	0.42	0.42	0.42	0.41	0.41	0.41	0.40	0.40
2	0.30	0.30	0.29	0.29	0.29	0.29	0.29	0.28	0.28	0.28	0.28
3	0.25	0.25	0.25	0.25	0.25	0.25	0.24	0.24	0.24	0.24	0.24
4	0.22	0.22	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
5	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
6	0.16	0.16	0.16	0.16	0.17	0.17	0.17	0.17	0.17	0.17	0.17
7	0.14	0.14	0.14	0.14	0.14	0.15	0.15	0.15	0.15	0.15	0.15
8	0.12	0.12	0.12	0.12	0.13	0.13	0.13	0.13	0.13	0.13	0.14
9	0.10	0.10	0.10	0.11	0.11	0.11	0.11	0.12	0.12	0.12	0.12
10	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10	0.10	0.11	0.11
11	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.09
12	0.04	0.05	0.05	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08
13	0.03	0.03	0.04	0.04	0.05	0.05	0.06	0.06	0.06	0.07	0.07
14	0.01	0.02	0.02	0.03	0.03	0.04	0.04	0.05	0.05	0.06	0.06
15		0.00	0.01	0.01	0.02	0.03	0.03	0.04	0.04	0.04	0.05
16				0.00	0.01	0.01	0.02	0.02	0.03	0.03	0.04
17						0.00	0.01	0.01	0.02	0.02	0.03
18								0.00	0.01	0.01	0.02
19										0.00	0.02

Tabela 10 – Shapiro-Wilk: Coeficientes a_i: Parte IV

n	39	40	41	42	43	44	45	46	47	48	49	50
i												
1	0.40	0.40	0.39	0,39	0.39	0.39	0.39	0.38	0.38	0.38	0.38	0.38
2	0.28	0.27	0.27	0.27	0.27	0.27	0.27	0.26	0.26	0.26	0.26	0.26
3	0.24	0.24	0.24	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23	0.23
4	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.20	0.20	0.20
5	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.18
6	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17
7	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.16	0.16	0.16	0.16
8	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
9	0.12	0.12	0.12	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
10	0.11	0.11	0.11	0.11	0.11	0.12	0.12	0.12	0.12	0.12	0.12	0.12
11	0.10	0.10	0.10	0.10	0.10	0.10	0.11	0.11	0.11	0.11	0.11	0.11
12	0.08	0.09	0.09	0.09	0.09	0.09	0.10	0.10	0.10	0.10	0.10	0.10
13	0.07	0.08	0.08	0.08	0.08	0.08	0.09	0.09	0.09	0.09	0.09	0.09
14	0.06	0.07	0.07	0.07	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.08
15	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.07	0.07	0.07	0.08
16	0.04	0.04	0.05	0.05	0.05	0.06	0.06	0.06	0.06	0.06	0.07	0.07
17	0.03	0.03	0.04	0.04	0.04	0.05	0.05	0.05	0.05	0.06	0.06	0.06
18	0.02	0.02	0.03	0.03	0.04	0.04	0.04	0.04	0.05	0.05	0.05	0.05
19	0.01	0.01	0.02	0.02	0.03	0.03	0.03	0.04	0.04	0.04	0.04	0.05
20	0.00	0.00	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.03	0.04	0.04
21			0.00	0.00	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.03
22					0.00	0.00	0.01	0.01	0.02	0.02	0.02	0.02
23							0.00	0.00	0.01	0.01	0.01	0.02
24									0.00	0.00	0.01	0.01
25											0.00	0.00

A.2 Valores Críticos p

Apresenta o nível de significância dado o número de amostras e W calculado

Tabela 11 – Shapiro-Wilk: p value

p	0,01	0,02	0,05	0,1	0,5	0,9	0,95	0,98	0,99
n									
5	0,686	0,715	0,762	0,806	0,927	0,979	0,986	0,991	0,993
6	0,713	0,743	0,788	0,826	0,927	0,974	0,981	0,986	0,989
7	0,730	0,760	0,803	0,838	0,928	0,972	0,979	0,985	0,988
8	0,749	0,778	0,818	0,851	0,932	0,972	0,978	0,984	0,987
9	0,764	0,791	0,829	0,859	0,935	0,972	0,978	0,984	0,986
10	0,781	0,806	0,842	0,869	0,938	0,972	0,978	0,983	0,986
11	0,792	0,817	0,850	0,876	0,940	0,973	0,979	0,984	0,986
12	0,805	0,828	0,859	0,883	0,943	0,973	0,979	0,984	0,986
13	0,814	0,837	0,866	0,889	0,945	0,974	0,979	0,984	0,986
14	0,825	0,846	0,874	0,895	0,947	0,975	0,980	0,984	0,986
15	0,835	0,855	0,881	0,901	0,950	0,975	0,980	0,984	0,987
16	0,844	0,863	0,887	0,906	0,952	0,976	0,981	0,985	0,987
17	0,851	0,869	0,892	0,910	0,954	0,977	0,981	0,985	0,987
18	0,858	0,874	0,897	0,914	0,956	0,978	0,982	0,986	0,988
19	0,863	0,879	0,901	0,917	0,957	0,978	0,982	0,986	0,988
20	0,868	0,884	0,905	0,920	0,959	0,979	0,983	0,986	0,988
21	0,873	0,888	0,908	0,923	0,960	0,980	0,983	0,987	0,989
22	0,878	0,892	0,911	0,926	0,961	0,980	0,984	0,987	0,989
23	0,881	0,895	0,914	0,928	0,962	0,981	0,984	0,987	0,989
24	0,884	0,898	0,916	0,930	0,963	0,981	0,984	0,987	0,989
25	0,888	0,901	0,918	0,931	0,964	0,981	0,985	0,988	0,989
26	0,891	0,904	0,920	0,933	0,965	0,982	0,985	0,988	0,989
27	0,894	0,906	0,923	0,935	0,965	0,982	0,985	0,988	0,990
28	0,896	0,908	0,924	0,936	0,966	0,982	0,985	0,988	0,990
29	0,898	0,910	0,926	0,937	0,966	0,982	0,985	0,988	0,990
30	0,900	0,912	0,927	0,939	0,967	0,983	0,985	0,988	0,990
31	0,902	0,914	0,929	0,940	0,967	0,983	0,986	0,988	0,990
32	0,904	0,915	0,930	0,941	0,968	0,983	0,986	0,988	0,990
33	0,906	0,917	0,931	0,942	0,968	0,983	0,986	0,989	0,990
34	0,908	0,919	0,933	0,943	0,969	0,983	0,986	0,989	0,990
35	0,910	0,920	0,934	0,944	0,969	0,984	0,986	0,989	0,990
36	0,912	0,922	0,935	0,945	0,970	0,984	0,986	0,989	0,990
37	0,914	0,924	0,936	0,946	0,970	0,984	0,987	0,989	0,990
38	0,916	0,925	0,938	0,947	0,971	0,984	0,987	0,989	0,990
39	0,917	0,927	0,939	0,948	0,971	0,984	0,987	0,989	0,991
40	0,919	0,928	0,940	0,949	0,972	0,985	0,987	0,989	0,991
41	0,920	0,929	0,941	0,950	0,972	0,985	0,987	0,989	0,991
42	0,922	0,930	0,942	0,951	0,972	0,985	0,987	0,989	0,991
43	0,923	0,932	0,943	0,951	0,973	0,985	0,987	0,990	0,991
44	0,924	0,933	0,944	0,952	0,973	0,985	0,987	0,990	0,991
45	0,926	0,934	0,945	0,953	0,973	0,985	0,988	0,990	0,991
46	0,927	0,935	0,945	0,953	0,974	0,985	0,988	0,990	0,991
47	0,928	0,936	0,946	0,954	0,974	0,985	0,988	0,990	0,991
48	0,929	0,937	0,947	0,954	0,974	0,985	0,988	0,990	0,991
49	0,929	0,938	0,947	0,955	0,974	0,985	0,988	0,990	0,991
50	0,930	0,939	0,947	0,955	0,974	0,985	0,988	0,990	0,991

B Algoritmo Simulação Cinemática

```
clc;clear;close all; %Limpa Tudo

syms l1 l2 l3 th1 th2 th3 g b a %Inicia variaveis simbolica

%Matriz R_zyx da base
Tf=[cos(a)*cos(b) cos(a)*sin(b)*sin(g)-sin(a)*cos(g)
cos(a)*sin(b)*cos(g)+sin(a)*sin(g) 0;
sin(a)*cos(b) sin(a)*sin(b)*sin(g)+cos(a)*cos(g)
sin(a)*sin(b)*cos(g)-cos(a)*sin(g) 0;
-sin(b) cos(b)*sin(g) cos(b)*cos(g) 0;
0 0 0 1];

%Cinempatica direta Robo RRR%%%
RRR1 =[cos(th1),-sin(th1)*0,sin(th1)*1,0  %Denavid Junta 0-1
sin(th1),cos(th1)*0,-cos(th1)*1, 0
0,1,0,l1
0,0,0,1];
RRR2 = [ cos(th2), -sin(th2), 0, l2*cos(th2) %Denavid Junta 1-2
sin(th2),  cos(th2), 0, l2*sin(th2)
0,0,1,0
0,0,0,1];
RRR3 = [ cos(th3), -sin(th3), 0, l3*cos(th3) %Denavid Junta 2-3
sin(th3),  cos(th3), 0, l3*sin(th3)
0,0,1,0
0,0,0,1];
THRRR0 = simplify(Tf); %Frame Base em relacao ao inercial
THRRR1 = simplify(Tf*RRR1); %Frame 1 em relacao ao inercial
THRRR2 = simplify(Tf*RRR1*RRR2); %Frame 2 em relacao ao inercial
THRRR3 = simplify(Tf*RRR1*RRR2*RRR3); %Frame 3 em relacao ao inercial

% Calculo do Angulos de Euler do End-Effector
R = THRRR3(1:3,1:3);
alpha=atan2(R(2,1),R(1,1)); % z
beta=atan2(-R(3,1),sqrt(R(3,2)^2+R(3,3)^2)); % y

%Defini-se o vetor de referencia da cinematica inversa
f_rot = [alpha ; beta];

% Vetor Z da base ( sera utilizado no plot da base ):
f_J0 = THRRR0(1:3,3);
% Equacoes da posicao do frame 1 (Para plot do robo)
f_J1 = THRRR1(1:3,4);
% Equacoes da posicao do frame 2 (Para plot do robo)
f_J2 = THRRR2(1:3,4);
```

```
% Equacoes da posicao do frame 3 (Para plot do robo)
f_J3 = THRRR3(1:3,4);

% Calculando o Jacobiano para a orientacao o end-effector:
Jf = jacobian(f_rot , [th1 th2 th3]);

%Posicao da base do robo
base.x = 0;
base.y = 0;
base.z = 0;

% Tamanho dos links do braco
L1 = 0.3567*10;%[cm]
L2 = 1.099*10;%[cm]
L3 = 0.45*10;%[cm]

% valor inicial das juntas
Theta = [0;0;0];

% -------- Taxa de aprendizado --------
n = 0.4;

% Bola de raio ksi (criterio de parada)
ksi = 5e-4;

% Obtenha o ponto-objetivo atual a ser alcancado: G = goal
G = [pi/6;
-pi/6];

% Vetores para plot
j1=[]; j2=[]; j3=[]; bx=[]; by=[]; bz=[]; ez=[]; ey=[];

%Inicia cinematica Inversa

for k = 0:70;

% Define os Angulos da base
angx =  0;
angy =  0;
angz =  0;
if (k >= 0) && (k <= 20)
angz =  0.5*sin(k*pi/10);
end
if (k >= 20) && (k <= 40)
angy =  0.5*sin(k*pi/10);
end
if (k >= 40) && (k <= 60)
```

```
    angx =  0.5*sin(k*pi/10);
    end

    % Posicao atual do end-effector para os valores de Theta atuais
    %Vetor Z Base
    P_J0 =  double(subs(f_J0 ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Posicao atual da Junta 1
    P_J1 = double(subs(f_J1 ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Posicao atual da Junta 2
    P_J2 = double(subs(f_J2 ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Posicao atual da Junta 3
    P_J3 = double(subs(f_J3 ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Direcao EndFactory
    P_rot = double(subs(f_rot ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Erro Quadratico Medio atual
    EQM = mean((G - P_rot).^2);

    % Enquanto a solucao nao atingir epsilon
    while EQM > ksi

    % Calcula o Gradiente Angular Atual, o erro angular:
    GradEQM = -(G - P_rot);

    % Calcula o Jacobiano para a posicao angular corrente
    J = double(subs(Jf , [l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % O Jacobiano pode sofrer de degeneracao (det(J2) = 0).
    J2=J'*J;
    if det(J2) == 0
    J2 = J2 + 0.00001 * eye(3);
    end

    %Calcula PseudoInversa de J
    psJ=inv(J2)*J';

    % O passo para alteracao de Theta na direcao contraria do Gradiente
```

```
    Theta = Theta - n *psJ *GradEQM;

    % Atualiza juntas para o novo Theta

    %Vetor Z Base
    P_J0 =  double(subs(f_J0 ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Posicao atual da Junta 1
    P_J1 = double(subs(f_J1 , [l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Posicao atual da Junta 2
    P_J2 = double(subs(f_J2 ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Posicao atual da Junta 3
    P_J3 = double(subs(f_J3 ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Direcao EndFactory
    P_rot = double(subs(f_rot ,[l1 l2 l3  th1 th2 th3  g b a] , [L1 L2 L3 Theta(1)
Theta(2)  Theta(3) angx angy angz]));

    % Erro Quadratico Medio atual
    EQM = mean((G - P_rot).^2);
    end

    %Atualiza vetor para plot das variaveis com o tempo.
    j1=[j1 Theta(1)];
    j2=[j2 Theta(2)];
    j3=[j3 Theta(3)];
    bx=[bx angx];
    by=[by angy];
    bz=[bz angz];
    ez=[ez P_rot(1)];
    ey=[ey P_rot(2)];


    %Plot dos graficos XY das variaveis
    subplot(3,3,3)
    plot(bx);
    title('roll base');
    axis([0 k -1 1]);

    subplot(3,3,2)
    plot(by);
```

```
title('pitch base');
axis([0 k -1 1]);

subplot(3,3,1)
plot(bz);
title('yaw base');
axis([0 k -1 1]);

subplot(3,3,4)
plot(j1);
title('junta 1');
axis([0 k 0 1]);

subplot(3,3,5)
plot(j2);
title('junta 2');
axis([0 k 0 1]);

subplot(3,3,6)
plot(j3);
title('junta 3');
axis([0 k 0 1]);
subplot(3,3,7)
plot(ez);
title('yaw efetuador');
axis([0 k 0 1]);

subplot(3,3,9)
plot(ey);
title('pitch efetuador');
axis([0 k -1 0]);

drawnow
end


%Plot do Robo Simulado na Ultima posicao
figure;

%Plot Juntas e links
plot3(base.x , base.y ,base.z, 'sk' , 'linewidth' , 2 , 'markersize' , 10)
hold on
plot3(P_J1(1) , P_J1(2) ,P_J1(3), 'ob' , 'linewidth' , 2 , 'markersize' , 10)
plot3([base.x P_J1(1)] , [base.y P_J1(2)] ,[base.z P_J1(3)], 'b' ,
'linewidth' , 2 , 'markersize' , 10)
plot3(P_J2(1) , P_J2(2) , P_J2(3), '*b' , 'linewidth' , 2 , 'markersize' , 20)
plot3([P_J1(1) P_J2(1)] , [P_J1(2) P_J2(2)],[P_J1(3) P_J2(3)] , 'b' ,
```

```
'linewidth' , 2 , 'markersize' , 10)
plot3(P_J3(1) , P_J3(2) , P_J3(3), 'ob' , 'linewidth' , 2 , 'markersize' , 10)
plot3([P_J2(1) P_J3(1)] , [P_J2(2) P_J3(2)],[P_J2(3) P_J3(3)] , 'b' ,
'linewidth' , 2 , 'markersize' , 10)

%Plot Plataforma
[xx,yy]=ndgrid(-5:1:5,-5:1:5);
z2 = (-P_J0(1)*xx - P_J0(2)*yy - 0)/P_J0(3);
surf(xx,yy,z2)
hold off
xlabel('x')
ylabel('y')
zlabel('z')
axis equal
grid on;
```

C Algoritmos do Experimento

C.1 Algoritmo no Microcontrolador

```
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"
#include "HMC5883L.h"
#include "math.h"
#include <Servo.h>


// Declaracao dos Sensores
// MPU 6050 - 0x68 - MPU6050_DEFAULT_ADDRESS
// HMC5883L - 0x1E - HMC5883L_DEFAULT_ADDRESS
MPU6050 mpu6050;
HMC5883L hmc5883l;

// Declaracao dos Servos
Servo myservo1;
Servo myservo2;
Servo myservo3;

//Declaracao das variaveis
int val1, val2, val3 ;
int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t mx, my, mz;
int  mode;
float AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ, MgX, MgY, MgZ, AX, AY, AZ,
GX, GY, GZ, MX, MY, MZ;
float Q_angle  =  0.003;
float Q_gyro   =  0.03;
float R_angle  =  0.23 * 0.23;
float x_bias = 0;
float y_bias = 0;
float z_bias = 0;
float XP_00 = 0.01, XP_01 = 0, XP_10 = 0, XP_11 = 0.01;
float YP_00 = 0.01, YP_01 = 0, YP_10 = 0, YP_11 =  0.01;
float ZP_00 =  0.01, ZP_01 = 0, ZP_10 = 0, ZP_11 =  0.01;
float KFangleX = 0.0;
float KFangleY = 0.0;
float KFangleZ = 0.0;
float DT = 0.05;
float rate_gyr_x = 0;
float rate_gyr_y = 0;
```

```
float rate_gyr_z = 0;
float AccZangle = 0;
float th1 = 0;
float th2 = 0;
float th3 = 0;


// Configurando mpu6050
// Configurando Parametros dos Escravos
void setSlaveControl(uint8_t slaveID) {
mpu6050.setSlaveEnabled(slaveID, true);
mpu6050.setSlaveWordByteSwap(slaveID, false);
mpu6050.setSlaveWriteMode(slaveID, false);
mpu6050.setSlaveWordGroupOffset(slaveID, false);
mpu6050.setSlaveDataLength(slaveID, 2);
}

void setup() {

// Definindo pino pwm da saida de pulso dos servos
myservo1.attach(9);
myservo2.attach(10);
myservo3.attach(11);

// Iniciando rede I2C
Wire.begin();

// Inciciando comunicacao serial
Serial.begin(9600);


//Aguarda permissao de inicializacao
Serial.println('a');
char a = 'b';
while (a != 'a')
{
a = Serial.read();
}


Serial.println("Initializing I2C devices...");

mpu6050.initialize();
if (mpu6050.testConnection()) {
Serial.println("MPU6050 connection successful");
}
else {
```

```
Serial.println("MPU6050 connection failed");
}

// configuratando magnetometro , para isso ativa Bypass
mpu6050.setI2CMasterModeEnabled(0);
mpu6050.setI2CBypassEnabled(1);
mpu6050.setSleepEnabled(0);
if (hmc5883l.testConnection()) {
Serial.println("HMC5883l connection successful");
hmc5883l.initialize();

I2Cdev::writeByte(HMC5883L_DEFAULT_ADDRESS,
HMC5883L_RA_MODE,
HMC5883L_MODE_CONTINUOUS);

$//Resolucao da Bussula
hmc5883l.setGain(HMC5883L_GAIN_1370);;

// Desativa Bypass
mpu6050.setI2CBypassEnabled(0);


// Configura endereco das medidas do magnetometro
// X axis word
mpu6050.setSlaveAddress(0, HMC5883L_DEFAULT_ADDRESS | 0x80);
mpu6050.setSlaveRegister(0, HMC5883L_RA_DATAX_H);
setSlaveControl(0);
// Y axis word
mpu6050.setSlaveAddress(1, HMC5883L_DEFAULT_ADDRESS | 0x80);
mpu6050.setSlaveRegister(1, HMC5883L_RA_DATAY_H);
setSlaveControl(1);
// Z axis word
mpu6050.setSlaveAddress(2, HMC5883L_DEFAULT_ADDRESS | 0x80);
mpu6050.setSlaveRegister(2, HMC5883L_RA_DATAZ_H);
setSlaveControl(2);

mpu6050.setI2CMasterModeEnabled(1);
} else {
Serial.println("HMC5883l connection failed");
}

// ativa sensor de tempertatura (nao usado neste algoritmo)
mpu6050.setTempSensorEnabled(true);

}

//Define as funcoes de filtro de kalman
```

```
float kalmanFilterX(float accAngleX, float gyroRate)
{
float  y, S;
float K_0, K_1;

KFangleX += DT * (gyroRate - x_bias);

XP_00 +=  - DT * (XP_10 + XP_01) + XP_11 * DT * DT + Q_angle * Q_angle;
XP_01 +=  - DT * XP_11;
XP_10 +=  - DT * XP_11;
XP_11 +=  + Q_gyro * Q_gyro * DT;

y = accAngleX - KFangleX;
S = XP_00 + R_angle;
K_0 = XP_00 / S;
K_1 = XP_10 / S;

KFangleX +=  K_0 * y;
x_bias  +=  K_1 * y;
XP_00 -= K_0 * XP_00;
XP_01 -= K_0 * XP_01;
XP_10 -= K_1 * XP_00;
XP_11 -= K_1 * XP_01;

return KFangleX;
}

float kalmanFilterY(float accAngleY, float gyroRate)
{
float  y, S;
float K_0, K_1;

KFangleY += DT * (gyroRate - y_bias);

YP_00 +=  - DT * (YP_10 + YP_01) + YP_11 * DT * DT + Q_angle * Q_angle;
YP_01 +=  - DT * YP_11;
YP_10 +=  - DT * YP_11;
YP_11 +=  + Q_gyro * Q_gyro * DT;

y = accAngleY - KFangleY;
S = YP_00 + R_angle;
K_0 = YP_00 / S;
K_1 = YP_10 / S;

KFangleY +=  K_0 * y;
y_bias  +=  K_1 * y;
```

```
YP_00 -= K_0 * YP_00;
YP_01 -= K_0 * YP_01;
YP_10 -= K_1 * YP_00;
YP_11 -= K_1 * YP_01;

return KFangleY;
}

float kalmanFilterZ(float accAngleZ, float gyroRate)
{
float  y, S;
float K_0, K_1;

KFangleZ += DT * (gyroRate - z_bias);

ZP_00 +=  - DT * (ZP_10 + ZP_01) + ZP_11 * DT * DT + Q_angle * Q_angle;
ZP_01 +=  - DT * ZP_11;
ZP_10 +=  - DT * ZP_11;
ZP_11 +=  + Q_gyro * Q_gyro * DT;

y = accAngleZ - KFangleZ;
S = ZP_00 + R_angle;
K_0 = ZP_00 / S;
K_1 = ZP_10 / S;

KFangleZ +=  K_0 * y;
z_bias  +=  K_1 * y;

ZP_00 -= K_0 * ZP_00;
ZP_01 -= K_0 * ZP_01;
ZP_10 -= K_1 * ZP_00;
ZP_11 -= K_1 * ZP_01;

return KFangleZ;
}

// Loop do algortimo
void loop() {

// Ve tempo atual, para calcular o delta temporal
float tempo = millis();

// Le valores atuais do magnetometro
mx = mpu6050.getExternalSensorWord(0);
my = mpu6050.getExternalSensorWord(2);
mz = mpu6050.getExternalSensorWord(4);
```

```
// Le valores atuais do acelerometro e giroscopio
mpu6050.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

//Transformando em variaveis float
AX = (float)ax;
AY = (float)ay;
AZ = (float)az;
GX = (float)gx;
GY = (float)gy;
GZ = (float)gz;
MX = (float)mx;
MY = (float)my;
MZ = (float)mz;

//Condicionamento do Sinais
AcX = (AX - 200) / 16400;
AcY = (AY - 200) / 16500;
AcZ = (AZ + 800) / 16600;
GyX = GX / 2500;
GyY = GY / 2500;
GyZ = GZ / 2500;
MgX = (MX + 390) / 354;
MgY = (MY - 302) / 365;
MgZ = (MZ + 302) / 348;


// Calculo dos Angulos de Euler em radianos
float AccXangle =  atan2(AcY, AcZ);
float AccYangle =   atan2( -AcX  , sqrt(AcZ * AcZ + AcY * AcY));
float AccZangle =   atan2(MgZ * sin(AccXangle) - MgY * cos(AccXangle),
MgX * cos(AccYangle) + MgY * sin(AccYangle) * sin(AccXangle) +
MgZ * sin(AccYangle) * cos(AccXangle));


// Chama as funcoes do Filtro de Kalman com as medidas atuais de Roll Pitch e Yaw
e suas velocidades angulares
float kalmanY = kalmanFilterY(AccYangle, GyY);
float kalmanX = kalmanFilterX(AccXangle, GyX);
float kalmanZ = kalmanFilterZ(AccZangle, GyZ);

//Comunicacao serial
if (Serial.available() > 0)
{
mode = Serial.read();
switch (mode)
{
```

```
// Envia valor atual dos Angulos de Euler Filtrados
Serial.print(kalmanX); Serial.print(" "); Serial.print(kalmanY);
Serial.print(" "); Serial.println(kalmanZ);
break;

// Recebe os angulos necessarios para os servos
case 'U':
th1 = Serial.parseFloat();
delay(10);
th2 = Serial.parseFloat();
delay(10);
th3 = Serial.parseFloat();
break;

}
}

//Acionamento dos servos

//limitando os angulos as condicoes do servo real.
val1 = int(th1);
if ( val1 < -90)
{
val1 = -90;
}
if ( val1 > 90)
{
val1 = 90;
}
val2 = int(th2);
if ( val2 < -90)
{
val2 = -90;
}
if ( val2 > 90)
{
val2 = 90;
}
val3 = int(th3);
if ( val3 < -90)
{
val3 = -90;
}
if ( val3 > 90)
{
val3 = 90;
}
```

```
//adequando o valor ao servo HS—422
val1 = map(val1, 90, —90, 0, 180);
val2 = map(val2, 90, —90, 0, 180);
val3 = map(val3, 90, —90, 0, 180);

//set angulos dos servos
myservo1.write(val1);
myservo2.write(val2);
myservo3.write(val3);

//Calcula tempo de duracao do loop em segundos
DT = (millis() — tempo) / 1000;

} //fim
```