DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Desenvolvimento e construção de protótipos robóticos educacionais

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Anelize Zomkowski Salvi

Florianópolis, Fevereiro de 2019

Desenvolvimento e construção de protótipos robóticos educacionais

Anelize Zomkowski Salvi

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Henrique Simas

Banca Examinadora:

Prof. Henrique Simas, UFSC
Orientador na Empresa

Prof. Henrique Simas
Orientador no Curso

Prof. Hector da Silveira
Responsável pela disciplina

Prof. Rodrigo Gesser, Avaliador

Renê Luiz dos Santos Baldissera, Debatedor

Gustavo dos Santos Gonçalves , Debatedor

Ao meu marido.

Agradecimentos

Agradeço aos meus amigos e professores por todo suporte durante estes anos de estudo. Ao meu orientador, Henrique Simas, pelo orientação neste trabalho e no de doutorado, pelas palavras de apoio e incentivo durante toda a jornada acadêmica.

Agradeço a minha mãe, irmão, ao amigo Antônio, aos meus sogros, aos meus avós e familiares, pelo suporte e compreensão durante estes anos de intenso trabalho.

Em especial, agradeço ao meu marido, que me impulsionou a cursar esta nova graduação, estando sempre ao meu lado. Este término de graduação e doutorado é uma conquista nossa e sempre serei grata por estes anos de companheirismo incondicional.

A resposta certa, não importa nada: o essencial é que as perguntas estejam certas.
Mario Quintana

Resumo

O presente trabalho foi desenvolvido como parte das atividades do projeto de pesquisa Triflex, do Laboratório Raul Gunther (LAR) da UFSC. Nele, foram desenvolvidos e validados protótipos de robôs Tripteron e Triflex II. Para tanto, foi desenvolvida uma plataforma modular para o acionamento destes robôs, sendo que o sistema Tripteron foi desenvolvido neste trabalho e o sistema Triflex II proveio da adaptação de um mecanismo existente no laboratório à nova plataforma de acionamento. Além disso, foram desenvolvidas as implementações em Arduino de duas rotinas que permitem o controle da movimentação das plataforma móveis do Tripteron e Triflex II por meio de dois joysticks, bem como a descrição de uma trajetória desejada, através da determinação dos parâmetros cinemáticos necessários. Assim, o sistema visa contribuir para o desenvolvimento de novos manipuladores paralelos e também apresenta finalidades educacionais, pois os mecanismos Tripteron e Triflex II apresentam equações cinemáticas inversas bem determinadas e desacoplamento de atuação. Logo, podem ser utilizados como exemplos didáticos para conceitos diversos do campo de robótica paralela.

Palavras-chave:. robôs paralelos, Tripteron, Triflex, autoalinhamento, cinemática.

Abstract

The present work was developed in the Raul Gunther Laboratory (LAR) of UFSC as part of the Triflex research project. In this work, prototypes of Tripteron and Triflex II systems were designed and validated. In order to construct the prototypes, a modular actuation platform was developed, to which the robots' legs and platform are attached. The Tripteron system was entirely designed in this work and the Triflex II system was adapted from an existing mechanism belonging to the laboratory. This mechanism was also attached to the novel modular actuation platform. In addition, this work also provides two Arduino routines, the first allows to control the robot platform by means of two joysticks and the second implements its inverse displacement equations. Thus, this work contributes to the development of new parallel manipulators and also presents educational purposes, since the robots Tripteron and Triflex II present simple kinematic equations and decoupling of actuation. Therefore, they can be used as didactic examples for different concepts in the area of parallel robotics.

Keywords:Parallel manipulator, Tripteron, Triflex, self-aligning, kinematics.

Lista de ilustrações

Lista de tabelas

=

Sumário

1 Introdução

O uso de robôs manipuladores está sendo mais frequente e está ligado diretamente a busca por inovações e melhora nos processos produtivos, buscando torná-los mais competitivos. Deste uso crescente, surge a necessidade do desenvolvimento de novos mecanismos e soluções robustas e adequadas ao ambiente industrial.

Neste contexto, o Laboratório Raul Gunther (LAR) da UFSC, no qual foi desenvolvido o presente trabalho, vem desenvolvendo pesquisas que busquem o aprimoramento do uso de robôs tanto no meio industrial quanto acadêmico.

Uma destas pesquisas desenvolvidas no laboratório é o projeto Triflex, coordenado pelo professor Henrique Simas. Nele, são estudadas as características estruturais e cinemáticas de robôs paralelos de três graus de liberdade, como por exemplo, o robô Tripteron (1), desenvolvido pela universidade de Laval.

Do projeto Triflex, foi desenvolvido um manipulador autoalinhante, derivado do robô Tripteron, o robô Triflex II (2). O robô Triflex II é composto por um mecanismo paralelo $PRRR + PRRU + PRRS$, em que P representa uma junta prismática, R, uma rotativa, S, uma esférica e U, uma universal, enquanto o robô Tripteron apresenta um mecanismo paralelo $PRRR + PRRR + PRRR$. Estes robôs são mostrados na Figura 1.

(a) (b)

Figura 1 – (a) Robô Tripteron (1), (b) Robô Triflex II (2).

O fato de o Triflex II ser autoalinhante confere-lhe características estruturais que serão discutidas a seguir.

Um mecanismo é dito autoalinhante se o mecanismo pode ser montado sem introduzir esforços e deformações mesmo que os corpos que formam o mecanismo possuam dimensões lineares e angulares diferentes das do projeto (12). Além disto, um mecanismo que possua esta propriedade é um mecanismo livre de restrições redun-

dantes, ou seja, de restrições que atuam no mesmo grau de liberdade de mecanismo.

Em geral, a presença de restrições redundantes em mecanismos e montagens implica vantagens como redundância nas conexões e redução de cargas concentradas. Porém, pode implicar em esforços na montagem, conferir ao mecanismo baixa tolerância a erros dimensionais, a deformações por impacto ou a variação térmica.

Assim, o projeto de mecanismos autoalinhantes pode apresentar vantagens práticas relevantes. Um exemplo dese tipo de projeto é a suspensão de um carro. A suspensão apresenta uma configuração autoalinhante, garantindo o funcionamento mesmo com pequenas deformações devidas aos impactos causados pelas irregularidade do manto estradal.

1.1 Motivação

Como mencionado, o presente trabalho foi desenvolvido no Laboratório Raul Gunther da UFSC, integrando o projeto de pesquisa Triflex, coordenado pelo professor Henrique Simas.

Como parte das pesquisas, o laboratório possui um modelo em ABS do robô Triflex II, porém seu sistema de acionamento não possibilita a validação de suas características cinemáticas e dinâmicas. Assim, surge a necessidade de desenvolver uma plataforma modular que permita não apenas avaliar as características do robô Triflex II, como também de outros robôs paralelos de três graus de liberdade, como o Tripteron, sua versão não autoalinhante.

O estudo de robôs como o Tripteron e o Triflex II podem ser utilizados como exemplos didáticos para conceitos diversos na área de robótica paralela. Como exemplos tem-se:

- conforme exposto no Capítulo 2, a comparação destes robôs permite a visualização dos conceitos de autoalinhamento e restrições redundantes, evidenciando a importância de se ponderar este tipo de características ao realizar o projeto de sistemas mecânicos,

- também conforme exposto no Capítulo 2, o robô Tripteron é um contra-exemplo conhecido para uma das fórmulas mais difundidas na literatura para o cálculo da mobilidade de um mecanismo,

- conforme exposto no Capítulo 3, estes robôs apresentam equações cinemáticas inversas bem determinadas, o que facilita suas implementações tanto em ambientes de simulação quanto no hardware escolhido para o protótipo proposto,

- também conforme exposto no Capítulo 3, dependendo da disposição de seus atuadores estes robôs podem apresentar acoplamento de atuação, bem como singularidades.

Assim, tem-se que a construção de protótipos deste tipo de robôs pode constituir uma plataforma de ensino e pesquisa relevante na área de robótica paralela.

1.2 Objetivos do trabalho

Nesta seção são apresentados os objetivos do presente trabalho.

1.2.1 Objetivo geral

O objetivo geral deste trabalho é o desenvolvimento e construção de protótipos dos robôs Tripteron e Triflex II, bem como implementação de suas equações cinemáticas.

1.2.2 Objetivos específicos

- Desenvolver um estudo dos mecanismos autoalinhantes que podem ser derivados do robô Tripteron;
- Levantar as restrições e detalhamento mecânico do sistema;
- Projetar o sistema eletrônico e de software;
- Desenvolver protótipos de robôs Tripteron e Triflex II utilizando um plataforma modular para o acionamento de suas pernas;
- Implementar as equações cinemáticas dos robôs;
- Testar e avaliar o sistema.

1.3 Organização do trabalho

O Capítulo 2 apresenta conceitos fundamentais da teoria de mecanismos, com foco na análise de restrições redundantes. O intuito deste capítulo é mostrar que o robô Tripteron e o robô Triflex II, que são os objetos de estudo deste trabalho, apresentam mecanismos similares.

O Capítulo 3 apresenta uma revisão de cinemática de robôs, focando-se na cinemática dos robôs Tripteron e Triflex II.

O Capítulo 4 apresenta o projeto da plataforma manipuladora proposta, utilizando a metodologia PRODIP (13).

O Capítulo 5 aborda os detalhes da implementação do projeto e os testes efetuados.

O Capítulo 6 apresenta as conclusões e as perspectivas de trabalhos futuros.

Os apêndices apresentam os detalhamentos dos desenhos mecânicos e os códigos implementados em Arduino, respectivamente.

2 Teoria de mecanismos e autoalinhamento

Neste capítulo são introduzidos conceitos fundamentais da teoria de mecanismos, com foco na análise de restrições redundantes. O intuito deste capítulo é mostrar que o robô Tripteron e o robô Triflex II, que são os objetos de estudo deste trabalho, apresentam mecanismos similares, sendo que o mecanismo do Triflex II pode ser obtido do mecanismo do robô Tripteron pela substituição adequada de juntas, tornando o Tripteron autoalinhante.

A motivação para esta exposição é o entendimento do mecanismo destes robôs, além do melhor entendimento dos protótipos de robô Tripteron e robô Triflex que são propostos nesse trabalho.

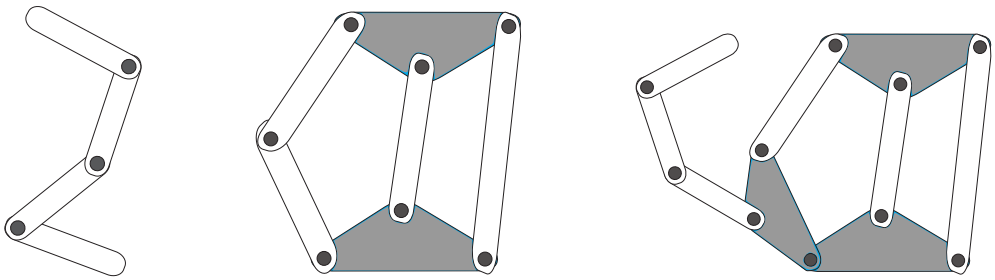
A terminologia utilizada neste capítulo encontra-se em concordância com a adotada pela "International Federation for the Promotion of Mechanism and Machine Science (IFToMM)" que pode ser encontrada em Ionescu. et. al. (14).

Um *mecanismo* é um sistema de corpos projetados para converter movimentos e forças de um ou vários corpos. Os corpos rígidos que fazem parte de um mecanismo são chamados de *elos*. Deste modo, pode-se também definir um mecanismo como uma cadeia cinemática em que um dos elos se comporta como uma base.

A superfície de contato de um elo em relação à outro é chamada de *elemento do par cinemático*. Dois elementos do par formam um *par cinemático*. Um par cinemático representa uma conexão entre dois elos. Uma *junta* é a realização física de um par cinemático, incluindo a ligação via elementos intermediários do mecanismo. Ao definir quais juntas do mecanismo são acionadas por um atuador, tem-se um *mecanismo atuado*.

Se cada elo em uma cadeia cinemática for conectado a outro elo por somente um caminho, a cadeia cinemática é chamada de *cadeia serial*. Por outro lado, se cada elo de uma cadeia cinemática for conectado a outro elo por no mínimo dois caminhos, a cadeia cinemática é denominada *cadeia fechada*. Se uma cadeia cinemática é formada por cadeias seriais e cadeias fechadas, ela é denominada *cadeia híbrida*. Na Figura 2 são ilustradas as cadeias cinemáticas serial, fechada e híbrida.

Um mecanismo é chamado de *mecanismo paralelo* se o elo escolhido como efetuador final está ligado à base pelo menos por duas cadeias cinemáticas independentes (15), (14), (16).



(a) (b) (c)

Figura 2 – Cadeia serial (a), cadeia fechada (b) e cadeia híbrida (c)

2.1 Mobilidade ou graus de liberdade

A *mobilidade M* ou *número de graus de liberdade* (*DoF*) de um mecanismo pode ser definida como o número de parâmetros independentes necessários para especificar completamente a configuração da cadeia cinemática no espaço, com respeito a um elo escolhido como referência.

Um corpo rígido deslocando-se no espaço pode apresentar seis movimentos distintos: três rotações e três translações, como mostra a Figura 3a . Enquanto no plano, pode apresentar três movimentos distintos: duas translações e uma rotação, como mostra a Figura 3b. Estes valores são os graus de liberdade associados ao espaço e ao plano, respectivamente.

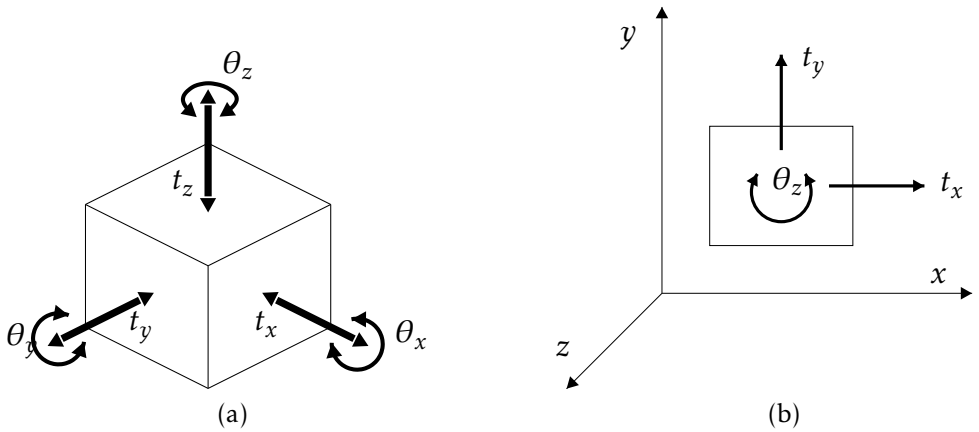


Figura 3 – Movimentos permitidos para um corpo rígido no espaço (a) e no plano (b).

Apesar de a mobilidade apresentar uma conceituação bem definida, diversas fórmulas para seu cálculo podem ser encontradas na literatura e apresentam aplicações específicas, sendo que uma revisão das diferentes abordagem para o cálculo da mobilidade pode ser encontrada em (17). Nas próximas seções é apresentada uma breve revisão bibliográfica das principais fórmulas para o cálculo da mobilidade encontradas na literatura.

2.1.1 O critério de Grübler-Kutzbach

O critério de mobilidade mais difundido na literatura é o critério de Grübler-Kutzbach (17).

$$M = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i \tag{2.1}$$

em que λ é a ordem do sistema de helicoides ($\lambda = 3$ no caso plano e $\lambda = 6$ no caso espacial), n é o número de elos do mecanismos, j é o número de juntas e f_i representa os graus de movimentos permitidos pela junta i.

Para um manipulador paralelo planar tipo 3RRR, tem-se um mecanismo com oito elos e nove juntas. Sua mobilidade pode ser calculada pelo critério de Grübler-Kutzbach:

$$\bar{l} = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i = 3(8 - 9 - 1) + 9 = 3 \tag{2.2}$$

A Figura 4 apresenta um contra-exemplo para o critério de Grübler-Kutzbach, o robô paralelo Tripteron (1), desenvolvido pela Universidade de Laval. Este robô é composto por um mecanismo paralelo contendo uma plataforma conectada à base por três pernas paralelas e apresenta mobilidade $M = 3$, porém, aplicando-se o critério de Grübler-Kutzbach tem-se $M = 0$, como será discutido a seguir.

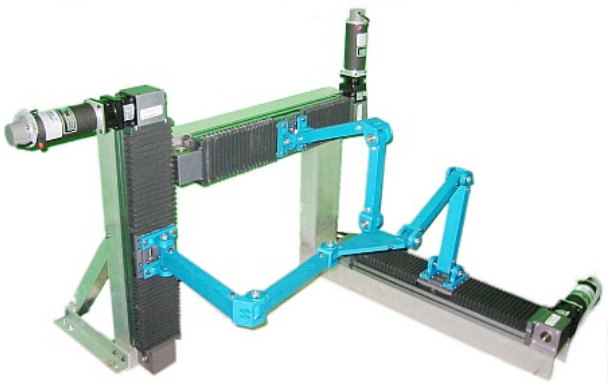
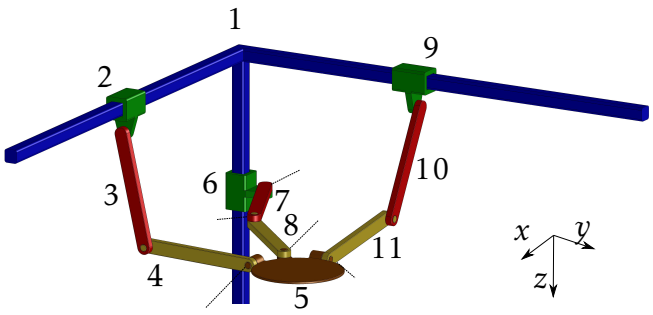


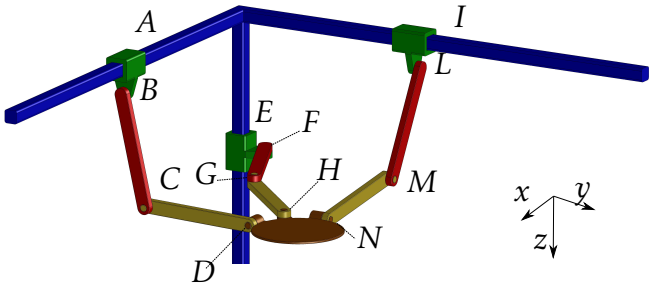
Figura 4 – Robô paralelo Tripteron (1).

A Figura 5 mostra o mecanismo do robô Tripteron. Sendo que Figura 5a mostra os $n = 11$ elos do Tripteron, enumerados de 1 a 11, e a Figura 5b mostra suas $j = 12$ juntas, nomeados de A a L. As juntas A, E e I são prismáticas, com translações relativas aos eixos x, z e y. As outras juntas são rotativas.

O Tripteron apresenta três graus de liberdade, as três translações ao longo dos eixos x, y e z, ou seja, mobilidade $M = 3$. Porém, aplicando o critério de Grübler-



(a)



(b)

Figura 5 – Robô Tripteron, seus elos (a) e juntas (b).

Kutzbach, tem-se:

$$M = \lambda(n-j-1) + \sum_{i=1}^{j} f_i = 6 \cdot (11-12-1) + 12 = 0 \tag{2.3}$$

Assim, aplicando o critério de Grübler-Kutzbach para o Tripteron obtém-se uma mobilidade $M = 0$, o que é um absurdo. Isto ocorre porque o Tripteron apresenta restrições redundantes, ou restrições que atuam sobre um mesmo grau de liberdade, e o critério de Grübler-Kutzbachm pode falhar nestes casos.

2.1.2 O critério de Ozol

O critério de Ozol (18) é uma reformulação do critério de Grübler-Kutzbach, considerando as restrições redundantes que atuam no mecanismo. Segundo o critério de Ozol:

$$M = \lambda(n-j-1) + \sum_{i=1}^{j} f_i + q \tag{2.4}$$

onde q representa o número de restrições que atuam em paralelo, ou seja, o número de restrições redundantes. Logo, o critério de Ozol efetua uma compensação dos graus de liberdade referentes às restrições redundantes que na fórmula anterior eram erroneamente retirados.

Por exemplo, o mecanismo Tripteron, apresentado na Figura 4, apresenta três restrições redundantes, logo:

$$M = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i = 6 \cdot (11 - 12 - 1) + 12 + 3 = 3 \tag{2.5}$$

Para determinar as restrições redundantes de um mecanismo, dois grupos de metodologias são encontradas na literatura: metodologias baseadas na análise topológica e metodologias baseadas na análise geométrica.

As baseadas na análise topológicas são: metodologia de Reshetov (12) e metodologia de Blanding (19). As baseadas na análise geométrica, e mais exatamente na formulação por helicoides, são: método de Davies (20) e método dos caminhos.

2.2 Mecanismos autoalinhantes

Um mecanismo é dito ser *autoalinhante* se pode ser montado sem introduzir esforços e deformações mesmo que os corpos que formam o mecanismo possuam dimensões lineares e angulares diferentes das do projeto (12). Além disto, um mecanismo que possua esta propriedade é um mecanismo livre de restrições redundantes.

A Figura 6 mostra um mecanismo de quatro barras que possui quatro juntas rotativas nomeadas A, B, C e D. Este mecanismo apresenta um grau de liberdade: a rotação em torno do eixo z.

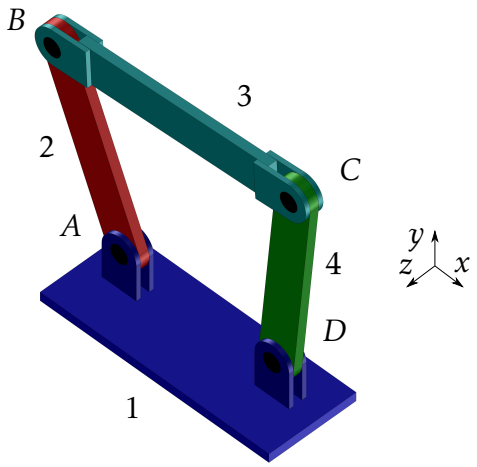


Figura 6 – Mecanismo de quatro com quatro juntas rotativas nomeadas A, B, C e D .

Imagine que o elo 1 da Figura 6 seja produzido com um defeito de fabricação de forma que as juntas rotativas A e D estejam desalinhadas, como mostra a Figura 7.

Para fechar o mecanismo em B serão necessárias deformações dos outros elos, portanto serão introduzidos esforços no mecanismo. Assim, pode-se concluir que o me-

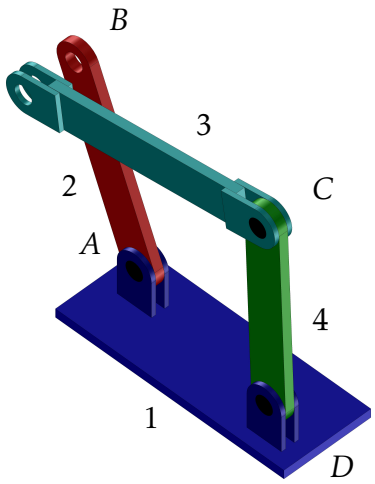


Figura 7 – Mecanismo de quatro barras com restrições redundantes

canismos de quatro barras com quatro juntas rotativas não é autoalinhante e, portanto, possui restrições redundantes.

2.3 Método de Reshetov

Esta seção descreve uma metodologia para análise da mobilidade de um mecanismo e da determinação das restrições redundantes. O objetivo desta metodologia é a eliminação das restrições redundantes através do aumento dos graus de liberdade de um conjunto de juntas, sem alterar a cinemática do mecanismo. O método, proposto por (12), é baseado na análise dos circuitos fundamentais, técnica que havia sido introduzida por Ozol (21) e Shamaidenko (12). O número de circuitos fundamentais de um mecanismo pode ser obtido através da equação:

$$v = j - n + 1. \tag{2.6}$$

A soma dos graus de liberdade das juntas F pode ser escrita em função das mobilidades relacionadas a cada eixo como:

$$F = f_{tx} + f_{ty} + f_{tz} + f_{rx} + f_{ry} + f_{rz} \tag{2.7}$$

sendo f_{tx}, f_{ty} e f_{tz} a soma das mobilidades lineares respectivamente ao longo dos eixos x, y e z e f_{rx}, f_{ry} e f_{rz} a soma das mobilidades angulares em torno dos mesmos eixos. A vantagem da formulação na equação 2.7 é a possibilidade de aplicação a diferentes sistemas de coordenadas. Podendo ser utilizado um sistema de coordenadas ortogonais ou oblíquas, sendo que, no último caso, os três eixos não podem pertencer a um mesmo plano.

Com base na formulação 2.7, Reshetov enuncia duas proposições que são expostas a seguir.

Proposição 1. *Dado um mecanismo formado por um circuito, a presença de todas as três mobilidades angulares $f_{rx} \geq 1$, $f_{ry} \geq 1$ e $f_{rz} \geq 1$ é condição necessária para que o circuito possa ser fechado sem deformações, ou seja, o mecanismo não possua restrições redundantes.*

Portanto, a falta de uma mobilidade angular implica a presença de restrições redundantes. Porém, a falta de uma mobilidade linear não sempre implica na presença de restrições redundantes. Isto porque uma mobilidade linear pode ser obtida não somente mediante de mobilidade linear de um par cinemático, mas também de uma rotação dos elos em torno de um eixo perpendicular à direção da mobilidade linear desejada.

Proposição 2. *Uma mobilidade linear pode ser substituída por uma mobilidade angular em torno de um eixo perpendicular à direção da mobilidade linear.*

A proposição 2 indica a possibilidade de compensar a falta de uma mobilidade linear por uma mobilidade angular extra. Porém é necessário verificar se no mecanismo existe um elo cuja rotação determine o movimento linear desejado, e que isto seja possível em todas as configurações do mecanismo.

Para mecanismos com número de circuitos $\nu > 1$, Reshetov apresenta uma metodologia completa (12) baseada nos seguintes passos fundamentais, como explicado em (21) :

1. Escolha qualquer conjunto de circuitos fundamentais com base na equação 2.6.
2. Prepare uma tabela com as mobilidade lineares e angulares para cada circuito fundamental.
3. Anote as mobilidades lineares e angulares na tabela para cada circuito. Em geral, uma mobilidade deve ser alocada no circuito onde eliminar uma restrição redundante, evitando circuito em que seja gerada mobilidade passiva.
4. Utilize as mobilidades angulares extras em cada circuito para compensar a falta de mobilidade linear. Se uma junta pertencer a dois circuitos, suas mobilidades angulares extras podem ser repartidas entre os dois circuitos para compensar a falta de mobilidade linear, sendo que cada mobilidade extra pode ser utilizada em apenas um circuito. Cada compensação é indicada com uma flecha apontando da mobilidade angular extra para a mobilidade linear faltante.

5. Compute todas as mobilidades que tiverem valor 0 como restrições redundantes e indique-as com uma flecha que aponte para baixo.

6. Compute as mobilidades restantes ≥ 2 como mobilidades do mecanismo, e indique-as com uma flecha que aponte para cima.

Porém, a metodologia de Reshetov pode apresentar falhas para mecanismos espaciais. Considere novamente o robô paralelo Tripteron, apresentado na Figura 4.

Para o cálculo das restrições redundantes, será aplicado o método de Reshetov (12). O número de circuitos fundamentais é $\nu = j - n + 1 = 12 - 11 + 1 = 2$. Escolhendo os circuitos $ABCDHGFE$ e $EFGHILMN$, o método de Reshetov é aplicado ao Tripteron, como mostrado na Figura 8, sendo que as linhas amarelas representam o circuito $ABCDHGFE$ e as linhas verdes, o circuito $EFGHILMN$.

translação	antes de compensar	após compensar	COMPENSAÇÃO	após compensar	antes de compensar	Rotação Mobilidade/ Restrição
f_x'	1 (A)	1(A)		2(CD)	3(BCD)	f_x"
f_y'	0	1(B)		0	0	f_y"
f_z'	1(E)	1(E)		2 (FG)	2(FG)	f_z"
f_x'	0	1 (L)		0	0	f_x"
f_y'	1(I)	1(I)		1(N)	3(LMN)	f_y"
f_z'	0	1(M)		1 (H)	1(H)	f_z"
translação	antes de compensar	após compensar		após compensar	antes de compensar	rotação

Figura 8 – Contra-exemplo para a metodologia de Reshetov.

O resultado apresentado na Figura 8 é independente do conjunto de circuitos fundamentais escolhido. A análise apresentada fornece um resultado incorreto: a mobilidade do Tripteron é $M = 2$ e o número de restrições redundantes é $q = 2$.

Note que para a análise usual, considerando dois circuitos fundamentais, tem-se apenas duas restrições redundantes aparentes, sendo estas relativas as restrições de rotação em x e y. Porém, analisando o mecanismo vê-se que ainda existe outra restrição

redundante relativa a rotação em z. Assim, a análise deve ser feita para os três circuitos, para que todas as restrições redundantes possam ser consideradas. Esta análise será efetuada na próxima seção que apresenta uma modificação ao método de Reshetov proposta por Carboni(22) e um vasto estudo das possibilidades de mecanismos autolinhantes derivados do Tripteron propostos pela estudante em conjunto com Estevan Murai.

2.4 Método de Reshetov modificado

Dada as limitações do método de Reshetov, principalmente devido à falha na identificação de restrições redundantes, Carboni(22) propôs a metodologia exposta a seguir para para avaliar as restrições redundantes de um mecanismo.

Dado um que se deseja avaliar, deve-se:

1. gerar todos os circuitos do mecanismo, sendo que qualquer conjunto de circuitos fundamentais do mecanismo pode ser utilizado para gerar todos os circuitos por combinações lineares. Desta forma, todos os percursos fechados do mecanismo são gerados. Um algoritmo que gera todos os circuitos de um mecanismo é proposto em (23).

2. Para cada circuito do mecanismo, deve-se:

   - montar uma tabela com a mobilidades lineares e angulares, atribuindo as mobilidades angulares e lineares proporcionadas pelas juntas do circuito. A diferença do método do Reshetov, na metodologia aqui apresentada, cada circuito é analisado de forma independente, e, para cada circuito, é verificada apenas a distribuição de mobilidades, com o objetivo de verificar a presença de restrições redundantes. As mobilidades das juntas compartilhadas por dois circuitos são atribuídas aos dois circuitos de forma independente.
   - Verificar se as mobilidades angulares são maiores que um, ou seja, se $f_{rx} \geq 1$, $f_{ry} \geq 1$ e $f_{rz} \geq 1$; caso contrário, cada mobilidade angular faltante é computada como restrição redundante.
   - Analisar as mobilidades lineares: as mobilidades lineares faltantes são compensadas, onde possível, com uma mobilidade angular extra ao longo de um eixo ortogonal ao eixo da mobilidade linear faltante.

3. O número de restrições redundantes do mecanismo é dado pela soma das restrições redundantes de cada circuito.

4. Dado o valor q das restrições redundantes, a mobilidade do mecanismo é calculada através do critério de Ozol:

$$M = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i + q$$

Como exemplo de aplicação de metodologia, a presente estudante em conjunto com o colega Estevan Murai desenvolveu um amplo estudo sobre as possibilidades de mecanismo autoalinhante derivados do Tripteron.

Considere os circuitos $ABCDEFGH$, $ILMNEFGH$ e $ABCDILMN$, linhas $1-3$, $3-6$ e $6-9$ da tabela na Figura 9, respectivamente.

translação	antes de compensar	após compensar	Compensação	após compensar	antes de compensar	Rotação Mobilidade/ Restrição
f_x'	1 (A)	1(A)		1(B)	2(BD)	f_x"
f_y'	0	1 (D)		0	0	f_y"
f_z'	1(E)	1(E)		1(H)	2 (GH)	f_z"
f_x'	0	1(G)		0	0	f_x"
f_y'	1(I)	1(I)		1(N)	2(MN)	f_y"
f_z'	0	1(M)		1(F)	1(F)	f_z"
f_x'	0	0		1(C)	1(C)	f_x"
f_y'	0	0		1(L)	1(L)	f_y"
f_z'	0	0		0	0	f_z"
translação	antes de compensar	após compensar		após compensar	antes de compensar	rotação

Figura 9 – Metodologia de Resehetov modificada.

Da análise dos três circuitos na Figura 9 tem-se $q = 3$, três restrições redundantes segundo as rotações em x, y e z, o que é compatível com o comportamento real do mecanismo. De fato:

- Para o primeiro circuito tem-se juntas rotativas apenas em x e z, assim aparece uma restrição redundante em relação a rotação em y;
- Para o segundo, as rotativas estão em y e z, logo a restrição surge em x;
- Para o terceiro, elas estão em x e y, portanto a restrição aparece em z.

Dessa forma, para que possam ser elencadas formas autoalinhantes por meio de substituição de juntas, deve-se compensar em cada circuito a restrição redundante de rotação relativa ao mesmo. Ou seja, determinar juntas que agreguem em cada circuito a rotação que lhe falta.

Em geral um mecanismo pode apresentar mobilidades gerais e locais (ou passivas) (12). A mobilidade que interessa a todos os elementos do mecanismo é chamada de *mobilidade geral*. E, a mobilidade que interessa a apenas um conjunto restrito de elementos do mecanismos é chamada de *mobilidade local*.

Logo, buscando evitar mobilidades passivas, algumas opções para a substituição de juntas para o Tripteron são apresentadas abaixo, em que R representa uma junta rotativa original, E uma esférica e D uma duplo-rotativa:

1. RR em braços distintos por EE. Esta substituição irá gerar duas mobilidades perigosas, logo não será considerada.
2. RR em braços distintos por uma E e uma D. Esta poderá ser uma posição autoalinhante, mas para isto a junta D deve ser orientada de modo a manter a rotação original de R e fornecer a rotação que falta ao circuito a que for incorporada.
3. RRR em braços distintos, por EEE também irá gerar mobilidade perigosa e será desconsiderada.
4. RRR em braços distintos por DDD, que pode ser uma forma autoalinhante desde que se observem as mesmas condições impostas em 2.
5. RRR em braços distintos por EDD pode ser uma forma autoalinhante se forem observadas as mesmas condições discutidas em 2, porém haverá a presença de mobilidades passivas. RRR em braços distintos por EED haverá mobilidades perigosas, portanto esta forma será desconsiderada.

Percebe-se que a substituição de um número crescente de juntas gera mobilidades passivas ou perigosas. Desta forma, a análise de Reshetov será feita apenas para os casos $2, 4$ e 5.

2.4.1 Análise de casos

Nesta seção os casos 2, 4 e 5 para possíveis substituições de juntas para o Tripteron são analisados em detalhes.

Para elucidar o segundo caso, considere a substituição das juntas:

- B por uma que esférica;
- L por uma duplo-rotativa orientada de modo a possibilitar uma rotação em y (a da junta rotativa) e uma em x (a rotação que falta ao circuito em que é incorporada).

Os circuitos considerados são: $ABCDEFGH$, linhas 1 – 3, e $ILMNEFGH$, linhas 3 – 6 na Figura 10, respectivamente.

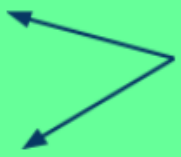
translação	antes de compensar	após compensar	Compensação	após compensar	antes de compensar	Rotação Mobilidade/ Restrição
f_x'	1 (A)	1(A)		3(BCD)	3(BCD)	f_x" ⇀ ⇀
f_y'	0	1(B)		1(B)	1(B)	f_y"
f_z'	1(E)	1(E)		2(FH)	3(BFH)	f_z" ⇀
f_x'	0	1 (L)		1(L)	1(L)	f_x"
f_y'	1(I)	1(I)		1(N)	3(LMN)	f_y"
f_z'	0	1(M)		1(G)	1(G)	f_z"
translação	antes de compensar	após compensar		após compensar	antes de compensar	rotação

Figura 10 – Metodologia de Resehetov modificada.

Da análise da tabela na Figura 10, verifica-se que para o caso 2 há o autoalinhamento e não há mobilidades passivas. Vale ressaltar que as substituições feitas para o caso 2 geram o mecanismo Triflex II (2), apresentado na Figura 11. Assim, tem-se que o Triflex II (2) é um mecanismo autoalinhante derivado do Tripteron (1).

Para elucidar o quarto caso, são feitas as seguintes substituições:

- junta B por uma duplo-rotativa que forneça rotações em x e y;

Figura 11 – Mecanismo Triflex II.

- junta F por uma duplo-rotativa que forneça rotações em z e x;
- junta L por uma duplo-rotativa que forneça rotações em y e z.

Os circuitos considerados são: $ABCDEFGH$, linhas 1 – 3, e $ILMNEFGH$, linhas 3 – 6 na Figura 12, respectivamente.

translação	antes de compensar	após compensar	Compensação	após compensar	antes de compensar	Rotação Mobilidade/ Restrição
f_x'	1 (A)	1(A)		2(CD)	3(BCD)	f_x"
f_y'	0	1(B)		1(B)	1(B)	f_y"
f_z'	1(E)	1(E)		3(FGH)	3(FGH)	f_z"
f_x'	0	1 (L)		1(F)	1(F)	f_x"
f_y'	1(I)	1(I)		1(N)	3(LMN)	f_y"
f_z'	0	1(M)		1(L)	1(L)	f_z"
translação	antes de compensar	após compensar		após compensar	antes de compensar	rotação

Figura 12 – Metodologia de Resehetov modificada.

Da análise da tabela na Figura 12, verifica-se que para o caso 3 há o autoalinhamento e não há mobilidades passivas.

Para elucidar o quinto e último caso, são feitas as seguintes substituições:

- junta B por uma esférica;
- junta F por uma duplo-rotativa que forneça rotações em z e x;
- junta L por uma duplo-rotativa que forneça rotações em y e z.

Novamente, os circuitos considerados são: $ABCDEFGH$ e $ILMNEFGH$, linhas 1 – 3 e 3 – 6 na Figura 13, respectivamente.

translação	antes de compensar	após compensar	Compensação	após compensar	antes de compensar	Rotação Mobilidade/ Restrição
f_x'	1 (A)	1(A)		2(CD)	3(BCD)	f_x"
f_y'	0	1(B)		1(B)	1(B)	f_y"
f_z'	1(E)	1(E)		4(BFGH)	4(BFGH)	f_z"
f_x'	0	1 (L)		1(F)	1(F)	f_x"
f_y'	1(I)	1(I)		1(N)	3(LMN)	f_y"
f_z'	0	1(M)		1(L)	1(L)	f_z"
translação	antes de compensar	após compensar		após compensar	antes de compensar	rotação

Figura 13 – Metodologia de Resehetov modificada.

Nesta caso há presença de uma mobilidade passiva, como comentado anteriormente.

2.5 Conclusões

Este capítulo apresentou conceitos fundamentais da teoria de mecanismos, com foco na análise de restrições redundantes. Foi mostrado, através do Método de Reshetov modificado, que o mecanismo do Triflex II e outros mecanismos, podem ser obtidos do mecanismo Tripteron pela substituição adequada de juntas, tornando o Tripteron autoalinhante.

O próximo capítulo apresenta uma revisão de cinemática de robôs, focando-se na cinemática dos robôs Tripteron e Triflex.

3 Conceitos básicos de cinemática de robôs manipuladores

Este capítulo apresenta uma revisão de cinemática de robôs, focando-se na cinemática dos robôs Tripteron e Triflex II, pois as equações da cinemática inversa foram implementadas no protótipo proposto.

A cinemática direta de robôs manipuladores visa determinar a posição do manipulador dado o acionamento das juntas. A cinemática inversa visa determinar qual deve ser o acionamento das juntas de modo a atingir uma dada posição do manipulador. Considere o robô manipulador representado na Figura 14, o problema de determinação da cinemática direta tem como dados de entrada as medidas dos ângulos α_1, α_2 e α_3 e visa determinar a posição do manipulador $P = (x_p, y_p)$, ou seja, é a transformação do sistema de coordenadas do espaço das juntas para o espaço cartesiano. O problema de determinação da cinemática inversa tem por objetivo determinar os parâmetros α_1, α_2 e α_3 para que o manipulador atinja o ponto $P = (x_p, y_p)$. A cinemática inversa é a transformação do sistema de coordenadas do espaço cartesiano para o espaço das juntas.

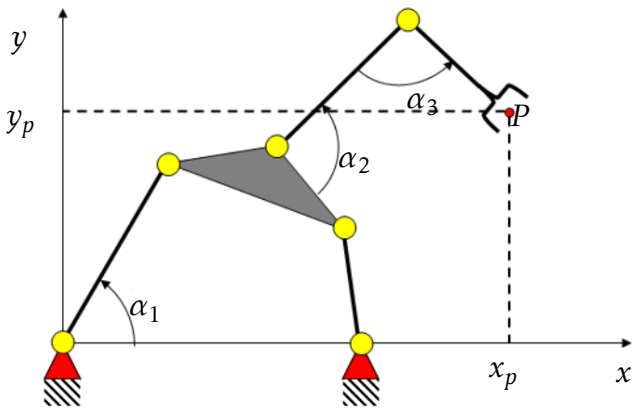


Figura 14 – Robô manipulador.

Na próxima seção, o problema de determinação das equações cinemáticas para robôs seriais é brevemente revisado, pois o foco principal do trabalho são os robôs paralelos Tripteron e Triflex II.

3.1 Cinemática de robôs seriais

Na próxima seção o problema de cinemática para robôs seriais é brevemente revisado, utilizando como ferramenta de simulação o Matlab R2017b da empresa Mathworks e Robotics Toolbox v8 (24).

Como o Robotics Toolbox v8 baseia-se na convenção de Denavit-Hartenberg, primeiramente, foi revisada esta convenção e após passou-se aos testes com a ferramenta em questão.

3.1.1 Convenção de Denavit-Hartenberg

Hartenberg e Denavit(3) propuseram uma notação sistemática para atribuir um sistema de coordenadas ortonormal com a regra da mão direita, um para cada elo numa cadeia cinemática aberta de elos. Uma vez que estes sistemas de coordenadas fixados ao elo são atribuídos, transformações entre sistemas de coordenadas adjacentes podem ser representadas por uma matriz de transformação de coordenadas homogêneas. Denavit e Hartenberg propõem uma convenção para a escolha do sistema de coordenadas de cada elo que será resumida a seguir, para tanto será considerado o sistema de juntas e elos apresentado na Figura 15.

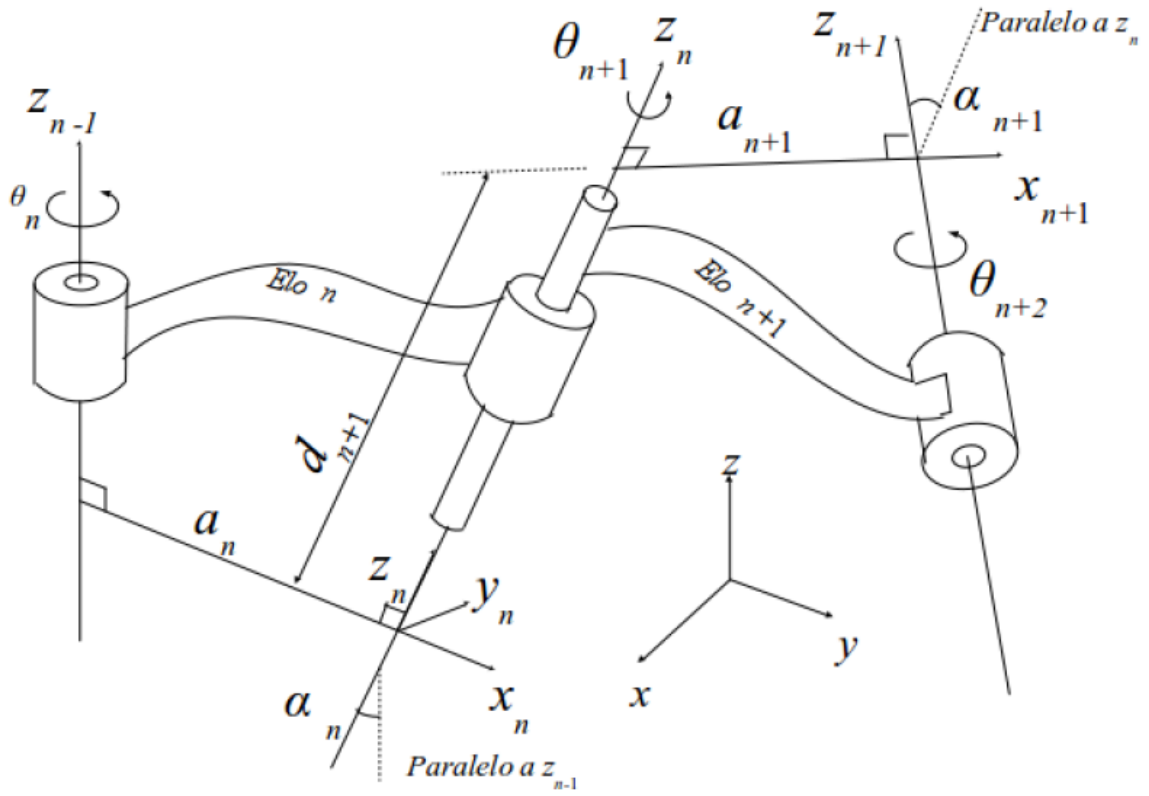


Figura 15 – Convenção de Denavit-Hartenberg (3).

De maneira geral, dado um robô serial, tem-se os seguintes passos para a determinação de seus parâmetros de Denavit-Hartenberg:

- **D1- Obtenção do sistema de coordenadas da base**: Estabelecer um sistema ortonormal de coordenadas (x_0, y_0, z_0) na base de suporte com o eixo Z_0 colocado ao longo do eixo de movimento da junta 1 apontando para o ombro do braço do robô. Os eixos X_0 e Y_0 podem ser convenientemente estabelecidos e são normais ao eixo Z_0.
- **D2- Inicialização e iteração**: Para cada $i = 1, ..., n$, efetuar passos D3 até D6.
- **D3 -Estabelecer o eixo das juntas**: Alinhar Z_i com o eixo de movimento (rotação ou translação) da junta $i + 1$. Para robôs tendo configurações de braço esquerdo-direito, os eixos Z_1 e Z_2 são apontados sempre para o ombro e o tronco do braço do robô.
- **D4- Estabelecer a origem do i-ésimo sistema de coordenadas**: Situar a origem do i-ésimo sistema de coordenas na interseção dos eixos Z_i e Z_{i-1} ou na interseção da normal comum entre os eixos Z_i e Z_{i-1} e o eixo Z_i.
- **D5- Estabelecer o eixo** X_i: Estabelecer X_i com vetor diretor $\frac{(Z_{i-1} \times Z_i)}{\|Z_{i-1} \times Z_i\|}$ ou ao longo da normal comum entre os eixos Z_i e Z_{i-1} quando eles forem paralelos.
- **D6 - Estabelecer o eixo** Y_i: Estabelecer Y_i com vetor diretor $\frac{(Z_{i-1} \times X_i)}{\|Z_{i-1} \times X_i\|}$.
- **D7 - Determinar os parâmetros das juntas e links**: Para cada $i = 1, ..., n$, efetuar os passos D8 a D11.
- **D8 - Determinar** d_i: d_i é a distância da origem do $(i-1)$-ésimo sistema de coordenadas até a interseção do eixo Z_{i-1} e o eixo X_i ao longo do eixo Z_{i-1}.
- **D9 - Determinar** a_i: a_i é a distância da interseção do eixo Z_{i-1} e o eixo X_i para a origem do i-ésimo sistema de coordenadas ao longo do eixo X_i.
- **D10 - Determinar** θ_i: θ_i é a medida do ângulo entre os eixos X_{i-1} e X_i sobre o eixo Z_{i-1}.
- **D11 - Determinar** α_i: α_i é o ângulo de rotação entre os eixos Z_{i-1} e Z_i sobre o eixo X_i.

Considere o robô planar, composto por duas juntas rotativas representado na Figura 16, seus parâmetros de Denavit-Hartenberg são mostrados na Tabela 1.

Uma vez que os sistemas de coordenadas Denavit-Hartenberg tenham sido estabelecidos, deve-se obter uma matriz de transformação homogênea relacionando o

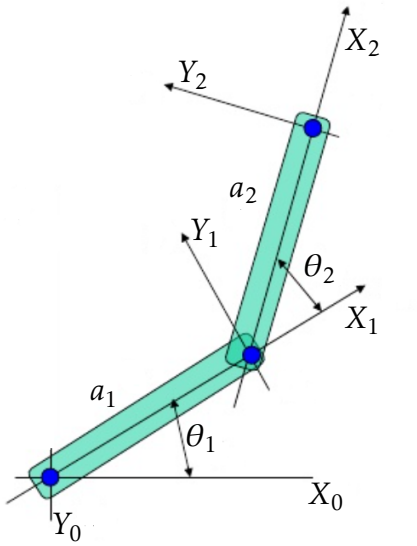


Figura 16 – Convenção de Denavit-Hartenberg para um robô planar com duas juntas rotativas.

i	a_i	α_i	d_i	θ_i
0	a_1	π	0	θ_1
1	a_2	0	0	θ_2

Tabela 1 – Parâmetros de Denavit-Hartenberg para um robô planar com duas juntas rotativas

i-ésimo ao $(i-1)$-ésimo sistema de coordenadas. Considerando o sistema de coordenadas como na Figura 15, um ponto r_i expresso no i-ésimo sistema de coordenadas pode ser expresso no $(i-1)$-ésimo sistema de coordenadas como r_{i-1} aplicando as transformações sucessivamente apresentadas a seguir:

- Rotação no eixo Z_{i-1}de um ângulo de θ_1 de modo a alinhar o eixo X_{i-1} com o eixo X_i.
- Translação uma distância de d_i ao longo do eixo Z_{i-1}.
- Translação ao longo do eixo X_i uma distância de a_i.
- Rotação do eixo X_i um ângulo de α_i.

Efetuando estas transformações tem-se a seguinte matriz de transformação homogênea relacionando o i-ésimo ao $(i-1)$-ésimo sistema de coordenadas:

$$A_i^{i-1} = \begin{bmatrix} cos\theta_i & -cos\alpha_i sen\theta_i & sen\alpha_i sen\theta_i & a_i cos\theta_i \\ sen\theta_i & -cos\alpha_i cos\theta_i & -sen\alpha_i sen\theta_i & a_i sen\theta_i \\ 0 & sen\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

Para o robô na Figura 16, tem-se:

$$A_1{}^0 = \begin{bmatrix} cos\theta_1 & sen\theta_1 & 0 & a_1 cos\theta_1 \\ sen\theta_1 & -cos\theta_1 & 0 & a_1 sen\theta_1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2{}^1 = \begin{bmatrix} cos\theta_2 & -sen\theta_2 & 0 & a_2 cos\theta_2 \\ sen\theta_2 & cos\theta_2 & 0 & a_2 sen\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplicando ambas as matrizes e tomando a última coluna, tem-se a posição do efetuador final que é dada por:

$$x = a_1 cos\theta_1 + a_2 cos(\theta_1 + \theta_2)$$

$$y = a_2 sen\theta_2 + a_2 sen(\theta_1 + \theta_2)$$

Para simular o comportamento cinemático deste robô foi utilizado o Robotics Toolbox v8. Este toolbox facilita a simulação da cinemática, dinâmica e a geração de trajetórias para robôs seriais, sendo baseado nos parâmetros de Denavit-Hartenberg do manipulador que são definidos como objetos do tipo *Link*.

Como primeira atividade durante o desenvolvimento deste trabalho, a estudante desenvolveu simulações de robôs manipuladores utilizando o Robotics Toolbox v8 (24) com intuito de fornecer exemplos simples para a compreensão da cinemática de robôs manipuladores serias pela convenção de Denavit-Hartenberg.

Primeiramente, a função Link é usada para criar cada elo, sendo que seus parâmetros são os parâmetros θ, d, a, α de Denavit-Hartemberg e um parâmetro auxiliar σ que determina o tipo de junta, sendo 0 para junta rotativa e 1 para prismática.

No caso do robô na Figura 16, como todas as juntas são rotativas, $\sigma = 0$. Assim, tomando $\theta_1 = \theta_2 = 0$, $a_1 = 1$, $a_2 = 2$ e os parâmetros d e α da Tabela 1, tem-se:

$$L(1) = Link([0\ 0\ 1\ \pi\ 0]);$$

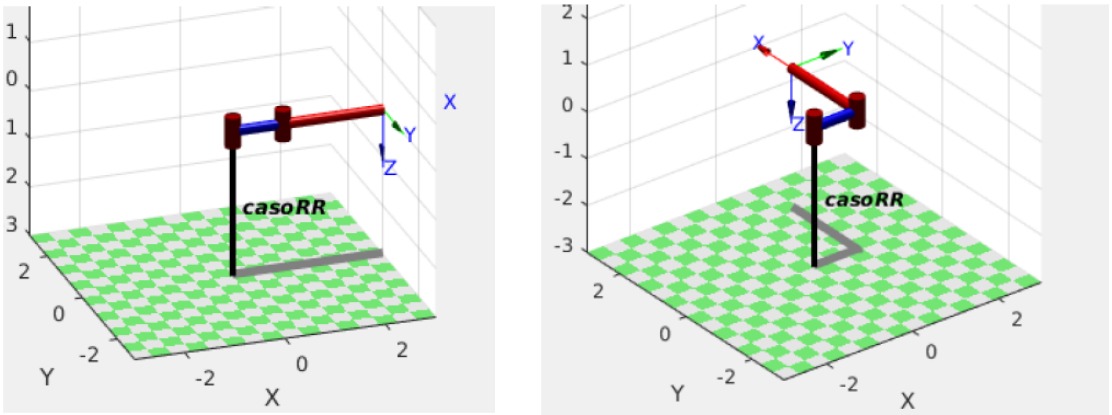
$$L(2) = Link([0\ 0\ 2\ 0\ 0]).$$

Criados os links, o robô é definido como uma cadeia serial dos mesmos, utilizando o comando SerialLink, no caso estudado, conferindo ao robô a nomenclatura casoRR, tem-se:

$$robo = SerialLink(L,'name','casoRR')$$

A função fkine é utilizada para implementar a cinemática direta, pois permite obter a matriz de transformação homogênea que relaciona o referencial da base com o do efetuador para uma dada configuração, sendo seu parâmetro, o deslocamento linear ou angular da junta.

Para exemplificar a cinemática direta, foram escolhidos os parâmetros $\theta_1 = \theta_2 = 0$ e $\theta_1 = 0$, $\theta_2 = -\pi/2$, sendo que os gráficos gerados com a função *plot*. Os resultados são ilustrados na Figura 17.



(a) (b)

Figura 17 – Robô planar composto de duas juntas rotativas nas posições (a) $\theta_1 = \theta_2 = 0$ e $\theta_1 = 0$, $\theta_2 = -\pi/2$.

Como segundo exemplo, considera-se o robô Puma. O Robotics Toolbox v8 apresenta diversos modelos de robô em sua biblioteca, sendo o robô Puma um deles, para utilizar o modelo do robô basta carregá-lo utilizando a função *mdl puma560* e os métodos utilizados são *fkine* para cinemática direta e *ikene6s*.

Inicialmente deve criar uma matriz de transformação que leve os eixos coordenados para uma posição desejada. Por exemplo, a função $T = transl(0.6, 0.1, 0) * rpy2tr(0, 180, 0, 'deg')$, translada os eixos pelo vetor $(0.6, 0.1, 0)$ e rotaciona em relação ao eixo y de um ângulo de 180^o. O resultado desta operação é mostrado na Figura 18a.

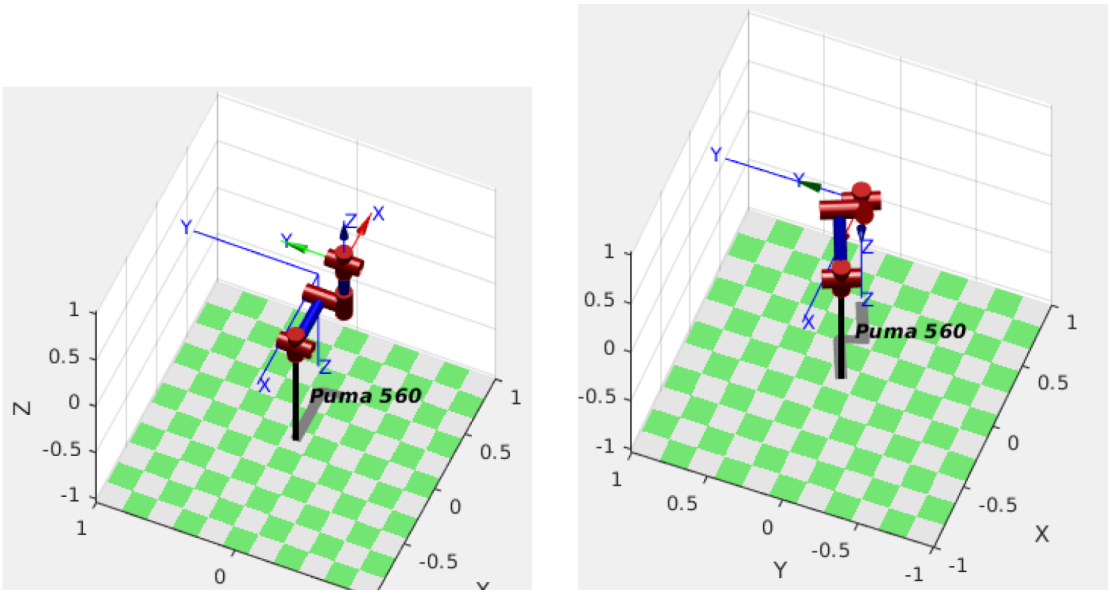
Após efetuada a transformação dos eixos, pode-se levar o robô a posição desejada utilizando o método *ikene6s* para determinar os ângulos de deslocamento e posterior método *fkine* para determinar a cinemática direta do robô:

$$q = p560.ikine6s(T)$$

$$p560.fkine(q).$$

Desta operação resulta a configuração na Figura 18b.

Na próxima seção o problema de determinação das equações cinemáticas para os robôs paralelos Tripteron e Triflex II será apresentado.



(a) (b)

Figura 18 – (a) Transformação dos eixos inicialmente associados ao robô; (b) Deslocamento do robô para a posição definida em (a).

3.2 Análise cinemática do robô Tripteron

Nesta seção, serão apresentadas as equações cinemáticas dos robôs Tripteron. Como discutido no capítulo anterior, estes robôs diferem apenas nos tipos de juntas que tornam o robô Triflex II autoalinhante. Neste capítulo será mostrado que a cinemática do Triflex II não difere da cinemática do Tripteron.

Considere a configuração do robô Tripteron na Figura 19.

A análise cinemática revisada neste capítulo foi apresentada por Kong e Gosselin(4). Para realizar esta análise, considere os sistemas de coordenadas $P - X_P Y_P Z_P$ e $O - XYZ$, ligados a plataforma e à base, respectivamente. Considere $i = 1, 2, 3$. Deixe B_i ser um ponto no eixo na junta rotacional i, A_i o ponto em que a junta prismática i conecta-se à base, A_{i0} o ponto inicial A_i e s_i o vetor unitário paralelo à junta prismática i.

Sem perda de generalização, considere cada um dos eixos X_P,Y_P e Z_P paralelos respectivamente à X,Y e Z e tome A_i e B_i tais que $A_i B_i$ seja perpendicular à s_i.

Deixe $-b_i^p$ denotar a posição do vetor B_i no sistema $P - X_P Y_P Z_P$ e a_i, a_{i0} as posições de A_i e de A_{i0} em relação ao sistema $O-XYZ$. Por fim, considere S_i o deslocamento de cada junta prismática em relação ao vetor s_i.

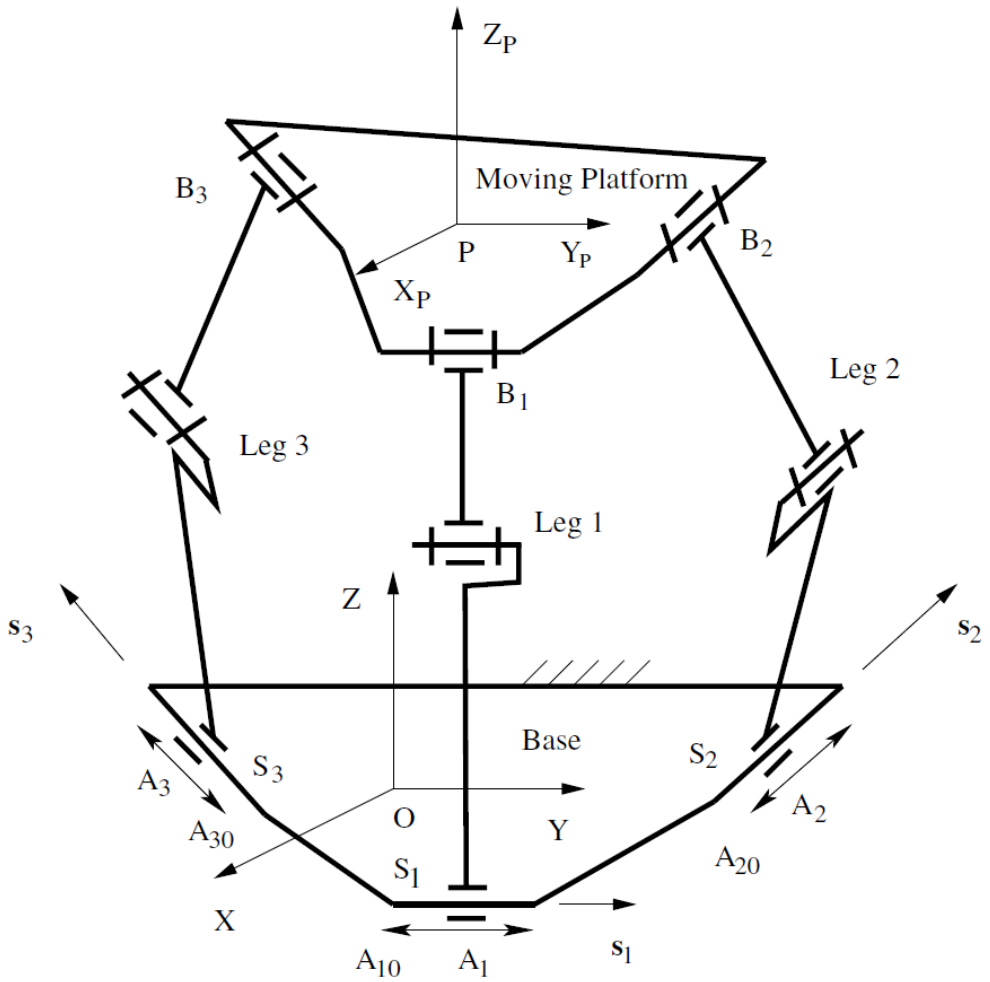


Figura 19 – Robô Tripteron (4).

3.2.1 Cinemática inversa do robô Tripteron

O problema de cinemática direta consiste em determinar os deslocamentos S_i, $i = 1, 2, 3$ das juntas prismáticas, dada uma posição p desejada para a plataforma móvel, em que p é o vetor de posição direcionado da origem O ao ponto P em coordenadas $O - XYZ$, conforme a Figura 19.

Como a plataforma móvel não pode rotacionar instantaneamente, a condição de perpendicularidade entre os vetores A_iB_i e o eixo da junta prismática i, com $i = 1, 2, 3$ é sempre mantida, logo:

$$s_i^T [p + b_i^P - (a_{i0} + S_i s_i)] = 0 \qquad i = 1, 2, 3. \tag{3.2}$$

Expandindo a Equação 3.2, obtem-se as equações da cinemática inversa do robô Tripteron.

$$S_i = s_i^T (p + b_i^P - a_{i0}) \qquad i = 1, 2, 3. \tag{3.3}$$

Para cada p no espaço de trabalho, a solução para os parâmetros S_i é única. No entanto, os parâmetros das juntas rotativas, que são passivas no processo, podem não ser únicos. Estes parâmetros podem ser determinados considerando cada perna do robô como um robô serial e aplicando a metodologia de Denavit-Hartenberg exposta na Seção 3.1.1.

3.2.2 Cinemática direta do robô Tripteron

O problema de cinemática direta para os robôs Tripteron consiste em determinar a posição p da plataforma móvel dados os deslocamentos S_i, $i = 1,2,3$ das juntas prismáticas.

Da Equação 3.2, tem-se:

$$s_i^T p = s_i^T (a_{i0} + S_i s_i - b_i^P) = 0 \qquad i = 1,2,3. \tag{3.4}$$

Reescreendo a Equação 3.4 na forma matricial, tem-se:

$$Jp = \begin{bmatrix} s_1^T (a_{10} + S_1 s_1 - b_1^P) \\ s_2^T (a_{20} + S_2 s_2 - b_2^P) \\ s_3^T (a_{30} + S_3 s_3 - b_3^P) \end{bmatrix} \tag{3.5}$$

em que a matriz Jacobiana J da manipulador é dado por

$$J = \begin{bmatrix} s_1^T \\ s_2^T \\ s_3^T \end{bmatrix} \tag{3.6}$$

Da Equação 3.5, tem-se:

$$p = J^{-1} \begin{bmatrix} s_1^T (a_{10} + S_1 s_1 - b_1^P) \\ s_2^T (a_{20} + S_2 s_2 - b_2^P) \\ s_3^T (a_{30} + S_3 s_3 - b_3^P) \end{bmatrix} \tag{3.7}$$

3.3 Análise cinemática do Triflex II

O robô Triflex II é composto por um mecanismo paralelo $PRRR+PRRU+PRRS$, em que P representa uma junta prismática, R, uma rotativa, S, uma esférica e U, uma universal. A Figura 20 mostra o projeto conceitual do Triflex II (2).

A análise cinemática revisada neste capítulo foi apresentada por Simas, Simoni e Martins(2). Para efetuar a análise, considere as pernas L_r, L_s, e L_u e o sistemas de coordenadas $(O_o - x_o y_o z_o)$ na Figura 20.

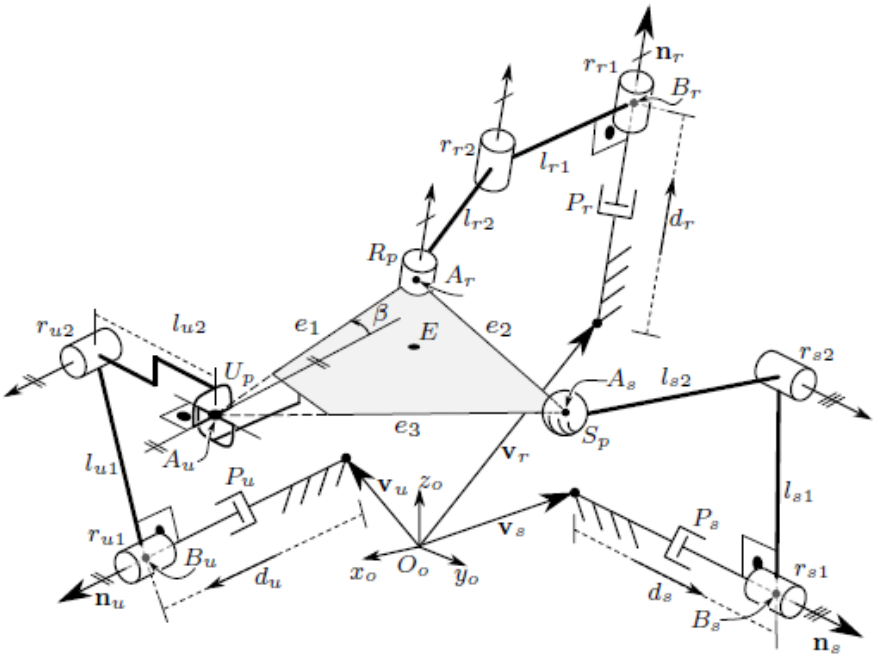


Figura 20 – Projeto conceitual do Triflex II (2).

Seja $i \in \{r, s, u\}$, considere v_i como o vetor de posição do ponto em que a junta prismática i conecta-se à base. Cada perna L_i apresenta dois links com comprimentos l_{i1} e e l_{i2}. Considere A_i como os pontos em que a perna L_i é ligada a plataforma móvel. Por fim, considere e_1 como a distância entre A_r e A_u, e_2 como a distância entre A_u e A_s e e_3 como a distância entre A_s e A_r, bem como um ponto E na plataforma móvel.

O mecanismo Triflex II apresenta a seguinte característica construtiva: A_u, A_s e A_r definem um plano cujo vetor normal é paralelo ao vetor $\vec{n}_r$ na Figura 20.

As coordenadas do ponto E na plataforma móvel podem ser escritas como:

$$E = K_u A_u + K_r A_r + K_s A_s \tag{3.8}$$

em que K_u, K_r and K_s são escalares conhecidos do projeto mecânico.

Sem perda de generalidade, considere z_o paralelo ao vetor $\vec{n}_r$ e $\vec{v}_r = 0$. Considere os parâmetros α, β, γ e δ como na Figura 21.

- α é a medida do ângulo entre x_o e a projeção do vetor $\vec{n}_u$ no plano $x_o y_o$ ($n_{u_{xy}}$) e pode ser obtido como $\alpha = Atan2(n_{u_y}, n_{u_x})$.
- β é a medida do ângulo entre a extensão da perna L_u e extensão do lado de comprimento e_1 da plataforma móvel.
- γ é a medida do ângulo entre x_0 e extensão do lado de comprimento e_1 da plataforma móvel, $\gamma = \alpha + \beta$.

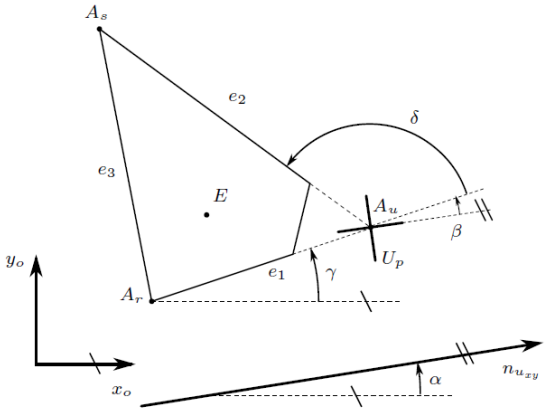


Figura 21 – Parâmetros auxiliares do Triflex II (2).

- δ é a medida do ângulo entre as extensões dos lados de comprimento e_1 e e_2 da plataforma móvel.

Considere os planos π_u, π_s e π_r na Figura 22, definidos respectivamente pelos vetores normais $\vec{n}_u$, $\vec{n}_s$ e $\vec{n}_r$ e os pontos B_u, B_s e B_r, respectivamente. Considere também $\vec{f}_u = \overrightarrow{B_u A_u}$, $\vec{f}_s = \overrightarrow{B_s A_s}$ e $\vec{f}_r = \overrightarrow{B_r A_r}$.

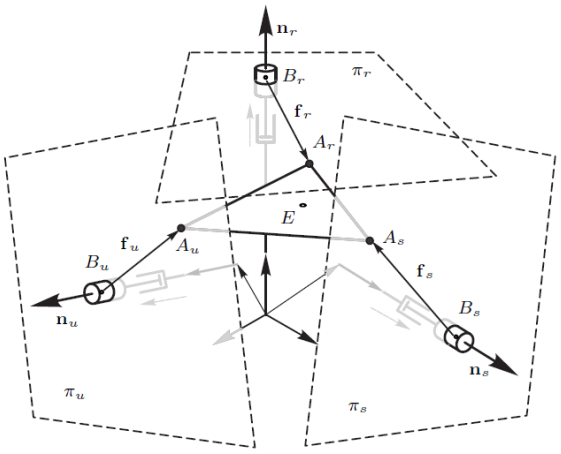


Figura 22 – Planos auxiliares do Triflex II (2).

3.3.1 Cinemática inversa do Triflex II

O objetivo da cinemática inversa é computar os deslocamentos d_u, d_s e d_r das juntas prismáticas em função das coordenadas de um ponto E na plataforma móvel.

Tem-se que A_u e A_s podem ser escritos em função de A_r como:

$$A_u = A_r + e_1 \begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \\ 0 \end{bmatrix} \tag{3.9}$$

e

$$A_s = A_r + e_1 \begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \\ 0 \end{bmatrix} + e_2 \begin{bmatrix} \cos(\gamma+\delta) \\ \sin(\gamma+\delta) \\ 0 \end{bmatrix} \tag{3.10}$$

Substituindo a Equação 3.9 e a Equação 3.10 na Equação 3.8, tem-se:

$$A_r = \frac{1}{K_u + K_s + K_r}(E - F_1 - F_2) \tag{3.11}$$

em que

$$F_1 = e_1 \begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \\ 0 \end{bmatrix} (K_u + K_s) \tag{3.12}$$

e

$$F_2 = e_2 \begin{bmatrix} \cos(\gamma+\delta) \\ \sin(\gamma+\delta) \\ 0 \end{bmatrix} K_s \tag{3.13}$$

Da Figura 22, tem-se:

$$\begin{aligned} \vec{n_u} \cdot \vec{f_u} &= 0 \\ \vec{n_s} \cdot \vec{f_s} &= 0 \\ \vec{n_r} \cdot \vec{f_r} &= 0 \end{aligned} \tag{3.14}$$

e

$$\vec{f_i} = A_i - B_i \text{ para } i = u, s, r \tag{3.15}$$

em que as coordenadas B_i são dadas por

$$B_i = \vec{v_i} + d_i \vec{n_i} \tag{3.16}$$

Substituindo as equações 3.15 e 3.16 na Equação 3.14, tem-se:

$$d_i = \vec{n_i} \cdot (A_i - \vec{v_i}) \tag{3.17}$$

em que $i \in \{r, s, u\}$.

Observe que estas equações cinemáticas são equivalentes às determinadas para o Tripteron. De fato, fazendo coincidir cada d_i ao adequado S_i, tem-se que:

- A_i na Equação 3.17 equivale a $p + b_i^P$ na Equação 3.3.
- $\vec{v_i}$ na Equação 3.17 equivale a a_{i0} na Equação 3.3.

.

Assim, os robôs Tripteron e Triflex II são cinematicamente equivalentes. Porém, no caso do robô Tripteron, as juntas prismáticas são mutuamente ortogonais enquanto no Triflex II elas podem estar numa configuração qualquer.

3.4 Análise de singularidade dos robôs Tripteron e Triflex II

Singularidades ocorrem quando a matriz Jacobiana J apresenta determinante nulo. Considere o robô Tripteron e a Equação 3.6, como os vetores s_1, s_2, s_3 são mutuamente ortogonais e não nulos, tem-se que o determinante de J é não nulo para todos os pontos do espaço, logo, o robô Tripteron não apresenta singularidades quando se consideram apenas os parâmetros s_1, s_2, s_3.

Porém, há restrições mecânicas que não são consideradas pelo Jacobiano e que são relativas à distância dos pontos A_i e B_i, uma vez que esta singularidade está relacionada ao espaço de trabalho das pernas passivas do robô. Para ilustrar este caso, considere a configuração do robô Triflex II na Figura 23.

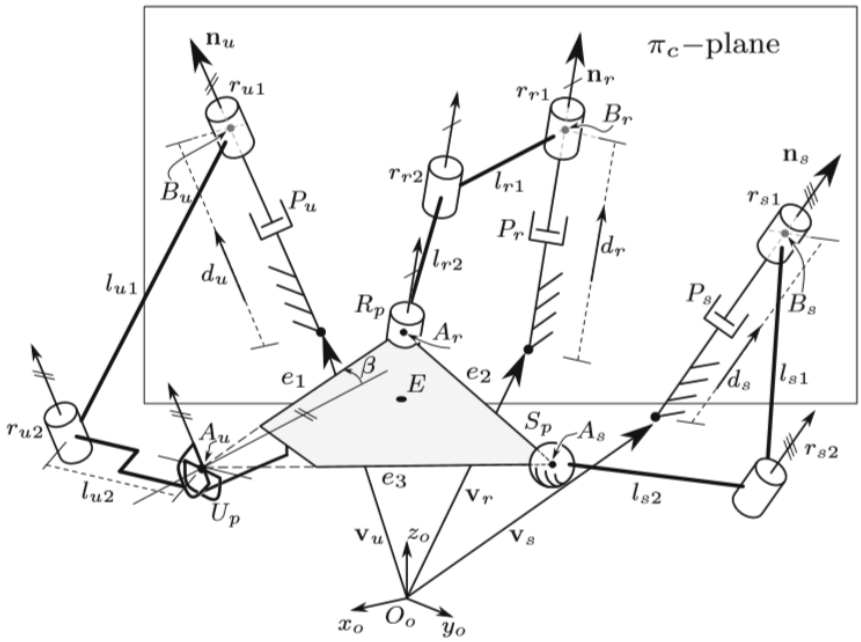


Figura 23 – Configuração singular do robô Triflex II (2).

Na Figura 23 considere os pontos A_i e B_i definidos como no caso do robô Tripteron tomando porém $i = u, s, r$. Considere também l_{ij} com $i = u, s, r$ e $j = 1, 2, 3$ os comprimentos das pernas conforme a Figura 23. Para que se respeitem as restrições mecânicas relativas à distância dos pontos A_i e B_i deve-se ter:

$$\|A_u - Bu\| < l_{u1} + l_{u2} \tag{3.18}$$

$$\|A_s - Bs\| < l_{s1} + l_{s2} \tag{3.19}$$

$$\|A_r - Br\| < l_{r1} + l_{r2} \tag{3.20}$$

Além disto, da Figura 23 pode-se concluir também que no caso do robô Triflex II podem ocorrer singularidades ligadas a matriz Jacobiana J, uma vez que os vetores diretores das juntas prismáticas não são necessariamente mutuamente ortogonais.

3.5 Conclusões

Este capítulo apresentou uma breve revisão de cinemática de manipuladores. O capítulo discutiu brevemente a cinemática de robôs seriais, utilizando como ferramenta de simulação o Robotics Toolbox v8 (24). Como o foco principal do trabalho são os robôs Tripteron e e Triflex II, o capítulo mostrou a equivalência entre suas equações cinemáticas e as diferenças nas condições de singularidade destes robôs.

4 Projeto da plataforma Tripteron/Triflex II

A metodologia utilizada para o desenvolvimento do projeto foi a do Projeto Integrado de desenvolvimento de Produtos [PRODIP] (13), desenvolvida no Núcleo de Desenvolvimento de Produtos [NEDIP] da Universidade Federal de Santa Catarina.

Na metodologia PRODIP, o desenvolvimento de produtos é idealizado em macrofases e fases. As fases são: planejamento, projetação e implementação.

O planejamento é dividido em planejamento do produto e do projeto, a projetação conta com fase de projeto informacional, conceitual, preliminar e detalhado. A implementação, por fim, conta com a preparação da produção, lançamento e validação. Sendo que métodos variados são sugeridos para a execução de cada etapa.

Ressalta-se porém que esta é uma metodologia de produto que não exige a execução de todas as etapas, buscando uma flexibilização e sendo conhecida como metodologia de referência, pois procura sistematizar e buscar meios para que a inovação ocorra sem perda de liberdade criativa.

Nas próximas seções é apresentado o projeto da plataforma manipuladora proposta, utilizando a metodologia PRODIP (13).

4.1 Planejamento do trabalho

O planejamento do trabalho centrou-se na definição das etapas do desenvolvimento de produtos que seriam adotadas. Foi definido que seriam apenas levantados os requisitos do sistema e as propostas de solução através de uma matriz morfológica, passando então para o projeto detalhado e prototipagem.

4.2 Projeto informacional

O projeto informacional aborda desde as definições do problemas às suas especificações. Não são apresentados as etapas de ciclo de vida e as especificações do usuário são breves, uma vez que se trata de um projeto centrado para o laboratório de robótica Raul Gunther que vem desenvolvendo diversos trabalhos na área, então estes requisitos estavam pré-definidos.

4.2.1 Definição do problema

O laboratório de robótica Raul Gunther desenvolveu uma série de pesquisas que culminaram no desenvolvimento dos mecanismos Triflex e Triflex II. Estes mecanismos são derivados do robô Tripteron e apresentam a característica de autoalinhamento. Destas pesquisas, surgiu a necessidade de se desenvolver um protótipo modular que auxiliasse nos diversos estudos conduzidos pelo laboratório, tanto na área de autoalinhamento, cinemática quanto no desenvolvimento de estratégias de controle.

Em vista desta necessidade, o projeto busca contribuir com o desenvolvimento de um manipulador modular, de fácil reconfigurabilidade, com boa repetibilidade, baixo custo e possibilidade de acoplamento de peças de outros protótipos desenvolvidos no laboratório. Vale ressaltar que durante o desenvolvimento do trabalho houve restrições quanto a materiais disponíveis no laboratório. Isto acabou por guiar o projeto mecânico e a implantação em hardware.

4.2.2 Requisitos do usuário

Tratando-se de um sistema didático para fins de ensino e pesquisa, o sistema desenvolvido deve:

- Não necessitar de um operador especializado em constante contato com o sistema;
- Apresentar baixo peso, para fins de transporte e operação;
- Possibilitar a visualização de conceitos como autoalinhamento e cinemática de manipuladores.

4.2.3 Requisitos de Projeto

O coordenador dos projetos envolvendo o sistema Triflex, professor Henrique Simas, estabeleceu os seguintes requisitos para o sistema mecânico e eletrônico do protótipo a ser construído:

- O sistema deve ser modular, sendo que as atuações em cada braço do robô devem ser simétricas;
- O sistema deve permitir o acoplamento de peças de outros protótipos desenvolvidos no laboratório;
- Os elos e juntas do manipulador devem ser fabricadas em ABS;
- O hardware deve ser simples e robusto, devendo-se utilizar motores NEMA disponíveis no laboratório;

- Deve ser operável por joystick e apresentar as implantações em hardware das equações cinemáticas.

4.3 Projeto Conceitual

A etapa de projeto conceitual visa gerar soluções que atendam as especificações levantadas no projeto informacional. Nela, buscaram-se as funções a ser desempenhada pelo produto com posterior busca de soluções, suas combinações e esboço de solução.

4.3.1 Estabelecimento de uma estrutura funcional

A Figura 24 apresenta um esquemático breve da estrutura funcional definida para o sistema. Nela, tem-se os sistemas: sistema embarcado, acionamento e transmissão, estrutura, fornecimento de energia e sensoriamento. Estes sistemas foram definidos para atender às necessidades do protótipo a ser construído, sendo que se partiu de funcionalidades gerais e estruturais do sistema, que são as mais à esquerda na Figura 24, passando ao seu refinamento e obtendo por fim a distribuição nos sistemas.

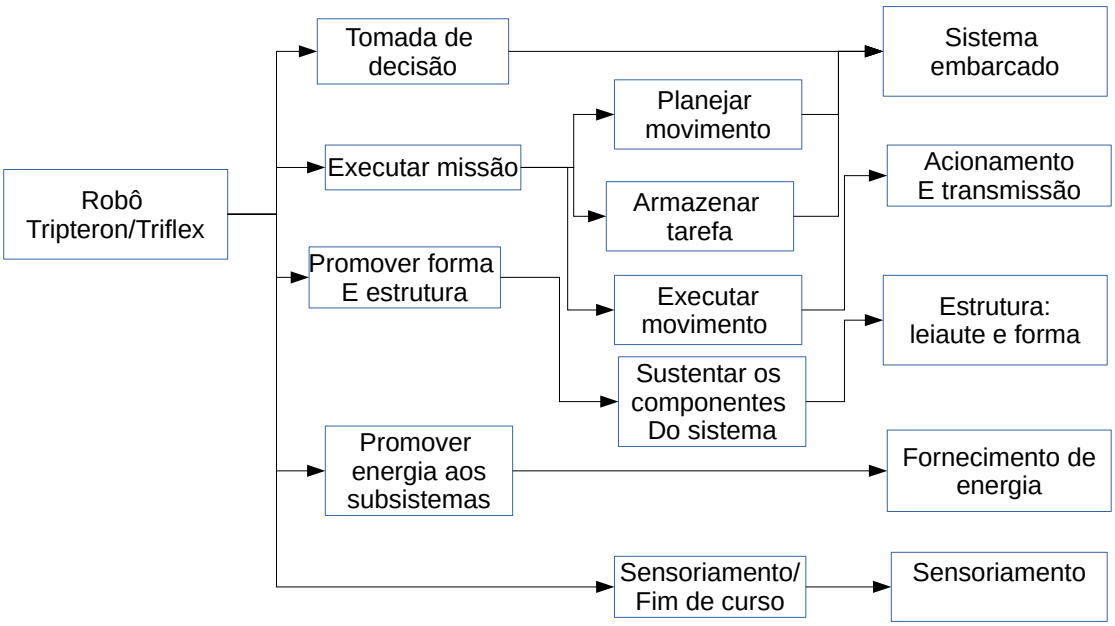


Figura 24 – Estrutura funcional para o sistema Tripteron/Triflex II.

Da Figura 24 tem-se que o sistema foi projetado para que o software pudesse ser embarcado, facilitando a operação por usuários menos experientes, os outros subsistemas são os usuais para este tipo de protótipo.

4.3.2 Pesquisa por princípio de solução

A matriz morfológica global do sistema é apresentada na Tabela 2.

Sistemas	Solução 1	Solução 2
Sistema embarcado	Arduino	Raspberry
Acionamento	servomotor	Nema 17, A4988 e Cnc shield
Transmissão	correia	fuso
Estrutura e leiaute	compensado	ABS
Fornecimento de energia	bateria	fonte CC
Sensoriamento	sensores de fim de curso	

Tabela 2 – Matriz morfológica

Da análise da matriz morfológica foram definidas as primeiras estruturas do sistema, conforme descrito a seguir.

4.4 Esboço de solução

4.4.0.1 Sistema embarcado

Após analisada as possibilidades, decidiu-se por um sistema utilizando o Arduino Uno. A escolha deveu-se principalmente pela familiaridade com o sistema e pela disponibilidade de bibliotecas e exemplos disponíveis na literatura.

4.4.0.2 Acionamento e transmissão

Foi escolhido o motor de passo Nema 17, o driver para motor de passo A4988 e o CNC shield v3. Esta escolha deveu-se ao fato de os motores de passo estarem disponíveis no laboratório e com o uso do driver e do CNC shield v3 o sistema eletrônico é compacto e adequado às necessidades do conjunto sem necessitar da fabricação de placas de circuito, o que facilita a manutenção do conjunto. Este conjunto é melhor explicado na Seção 4.5.2.

4.4.0.3 Estrutura e leiaute

O robô Tripteron e o Triflex II foram fabricados em ABS por um requisito de projeto, porém a plataforma a que o robô encontra-se fixado foi construída em compensado para diminuir custos e facilitar a prototipagem.

4.4.0.4 Fornecimento de energia

Para o fornecimento de energia aos motores foi utilizada uma fonte CC $12V$ com capacidade de corrente de $15A$. Para fornecer energia ao Arduino é utilizada uma bateria $9V$.

4.4.0.5 Sensoriamento

Para o sensoriamento utilizou-se apenas sensores fim de curso, um em cada braço para fornecer um sistema referencial para cada NEMA 17, possibilitando a implementação das equações cinemáticas.

4.5 Detalhamento do projeto

A visão geral do sistema projetado é apresentada na Figura 25. Foi optado por esboçar uma solução utilizando o sistema Tripteron na configuração clássica, ou seja, aquela em que as juntas prismáticas são mutuamente ortogonais.

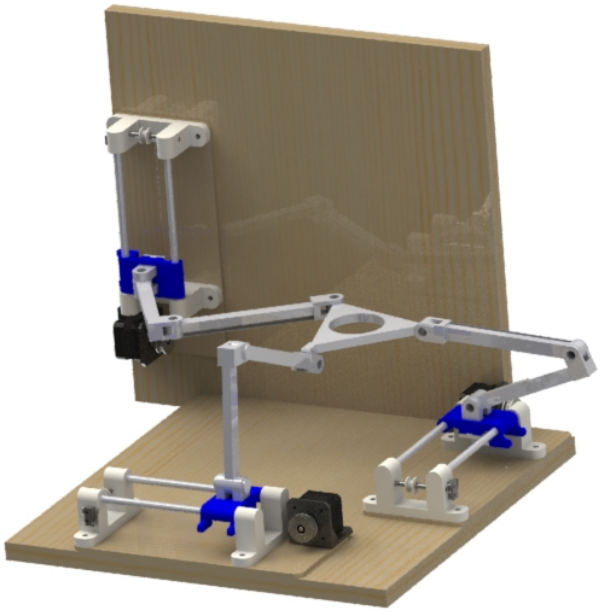


Figura 25 – Estrutura do sistema Tripteron/Triflex II proposto.

4.5.1 Detalhamento do sistema mecânico

Como os robôs Tripteron e Triflex II são robôs paralelos simétricos em sua atuação, modelou-se o sistema com três conjuntos intercambiáveis de acionamento e transmissão. Um desses conjuntos é mostrado na Figura 26.

Este conjunto é constituído por três elos, um dos elos, denominado slider, pertence ao sistema de transmissão linear e os outros dois elos pertencem a uma das pernas do robô. A junta prismática entre o slider e a base é implementada mediante a utilização de guias lineares em aço retificado com diâmetro de 8 mm, conforme mostrado na Figura 26. Essa junta é responsável pelo acionamento ou atuação, sendo, portanto, responsável pela movimentação de um grau de liberdade do robô.

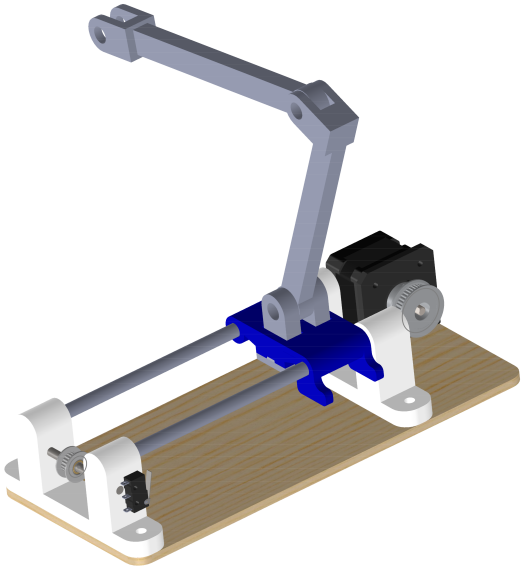
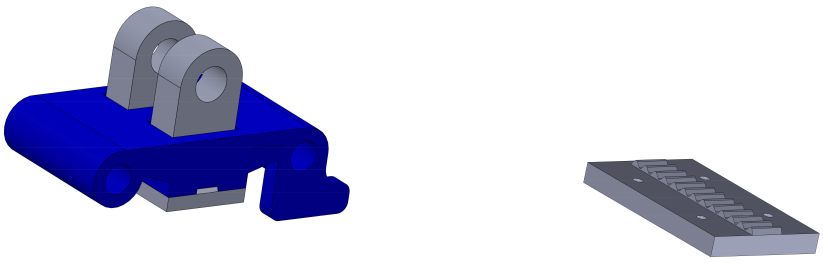


Figura 26 – Conjunto de acionamento mecânico base-plataforma.

A atuação é realizada da seguinte forma: uma correia dentada de 6.5 mm de largura movimenta o slider em relação à guia linear. A correia é acoplada através de uma polia dentada de 20 dentes a um motor de passo Nema, sendo que o tensionamento é realizado através de uma polia tensionadora, em ABS. Além disso, o slider foi projeto com um sistema integrado de fixação e regulagem da correia conforme mostrado na Figura 27.



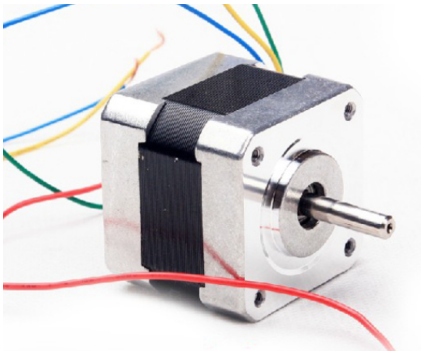
(a) (b)

Figura 27 – (a) Slider e seu sistema integrado de fixação e regulagem da correia (b).

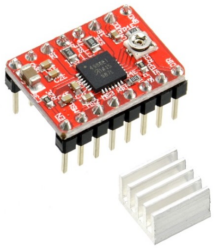
Os guias lineares são fixados à base por suportes em ABS cujo desenho é apresentado no Apêndice I, bem como o desenho detalhado dos demais componentes. Ressalta-se que a fabricação foi realizada por impressão $3D$, diretamente a partir do arquivo CAD. Logo, as cotas apresentadas são apenas para ilustrar as dimensões e não para fabricação.

4.5.2 Detalhamento do sistema eletrônico

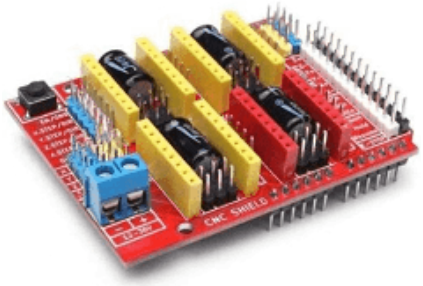
O sistema eletrônico conta com motor de passo Nema 17, driver para motor de passo A4988 e o CNC shield v3, um Arduino Uno, sensores fim de curso e dois joysticks mostrados na Figura 28.



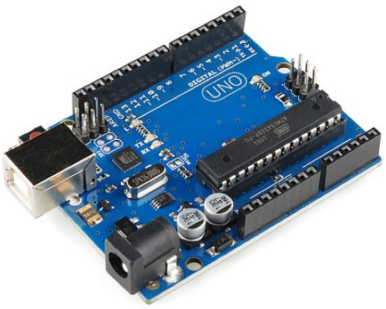
(a)



(b)



(c)



(d)

(e)

(f)

Figura 28 – (a) Nema 17 (5), (b) driver para motor de passo A498 (6), (c) CNC shield v3 (7), (d) Arduino Uno (8), (e) sensores fim de curso (9) (f) joystick (10).

4.5.3 Motor de passo

O motor de passo escolhido foi o Nema 17 (5), com tensão de operação de 3 – $20VDC$, corrente máxima de $0,95A$ e torque máximo de $2,7kg/cm$. Ele foi escolhido por estar disponível no laboratório.

4.5.4 Driver para motor de passo A4988

Para o acionamento do motor de passo foi escolhido o driver para motor de passo A4988 (6). Este driver opera em tensão lógica de $3-5V$, sendo adequados para motores que operem em tensão de $8-35V$, fornecendo até $2A$ por bobina.

4.5.5 CNC shield v3 para arduino

O CNC shield v3 (7) é adequado para Arduino Uno, apresenta entrada para alimentação externa, pinos para controle de sensores e soquetes para 4 drivers A4988. Ele foi escolhido por simplificar o sistema e eliminar a necessidade de placas de circuito impressos, facilitando a montagem e manutenção do sistema.

Como dito, a placa CNC shield v3 apresenta 4 soquetes para drivers A4988, sendo que cada um destes drivers pode controlar um motor de passo distinto. Porém, apenas três motores podem ser controlados independentemente, sendo associados a um sistema x, y ou z de variáveis da placa, como mostra Figura 29. O quarto motor, se utilizado, deve espelhar um dos movimentos. Como o sistema proposto apresenta apenas três eixos, foram utilizados três motores de passo, associados a cada um dos sistemas x, y ou z de variáveis da placa.

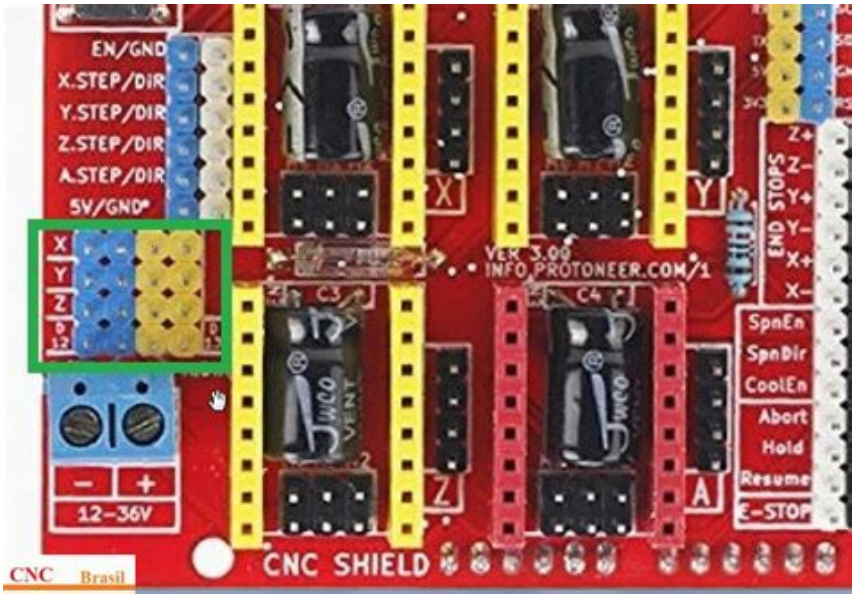


Figura 29 – Detalhes da placa CNC shield v3 (7).

A Figura 30 mostra a conexão entre os pinos do CNC shield v3 e o Arduino Uno.

As ligações realizadas entre as portas do Arduino Uno, CNC shield v3 e Joysticks (10) são esquematizadas na Tabela 3.

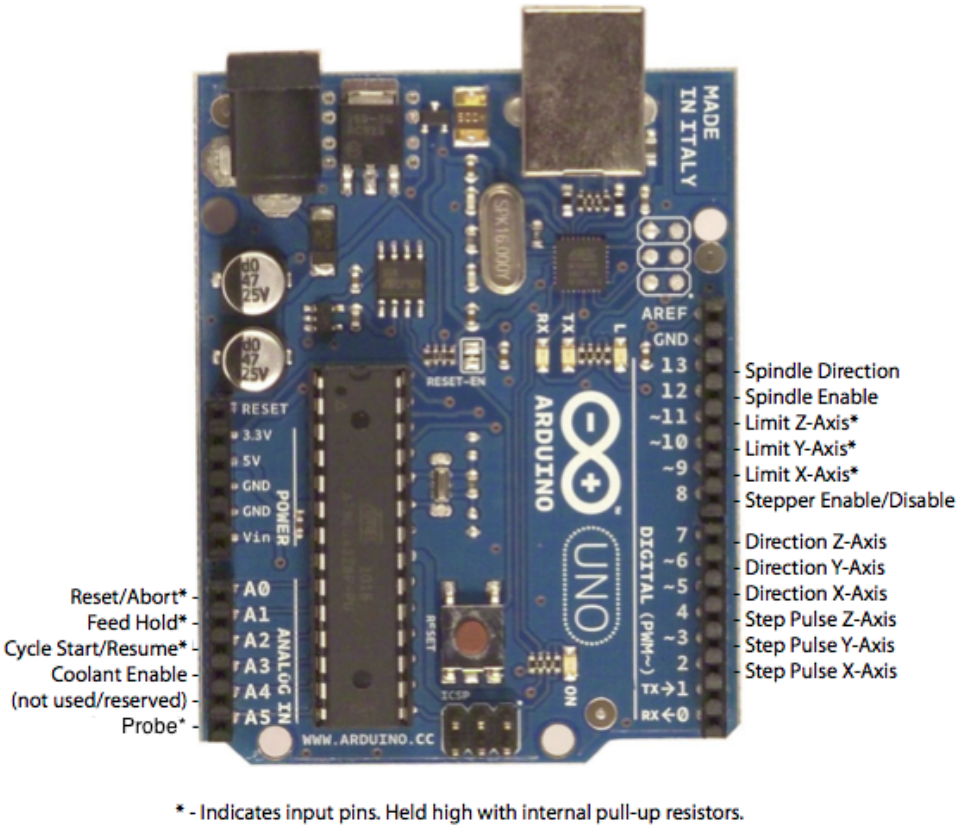


Figura 30 – Conexão entre os pinos do CNC shield v3 e o Arduino Uno (11).

Porta do Arduino	Porta do CNC shield v3	Função
D2	Step Pulse x	Ativação do motor x
D5	Direction x	Determinar a direção de giro do motor x
D9	Limit x	Ler sinal do sensor de fim curso x
A0	Abort	Ler sinal da porta VRx do primeiro Joystick
D3	Step Pulse y	Ativação do motor y
D6	Direction y	Determinar a direção de giro do motor y
D10	Limit y	Ler sinal do sensor de fim curso y
A1	Hold	Ler sinal da porta VRy do primeiro Joystick
D4	Step Pulse z	Ativação do motor z
D7	Direction z	Determinar a direção de giro do motor z
D11	Limit z	Ler sinal do sensor de fim curso z
A2	Resume	Ler sinal da porta VRx do segundo Joystick

Tabela 3 – Esquema das ligações elétricas entre arduino uno, cnc shield v3 e Joysticks.

4.5.6 Arduino Uno

A placa Arduino Uno (8) foi escolhida pela sua versatilidade, pela familiaridade de uso e por estar disponível no laboratório.

4.5.7 Chave fim de curso

A chave de fim de curso (9) escolhida por apresentar tensão máxima de operação de $125VAC$ e corrente máxima de $2A$.

4.5.8 Joystick

O Joystick (10) escolhido opera em tensão de $3-5V$, duas saídas analógicas para controle dos eixos, e uma saída digital.

4.6 Conclusões

O capítulo apresentou o projeto do manipulador Tripteron/Triflex II. O projeto foi desenvolvido utilizando a metodologia PRODIP com o intuito do desenvolvimento de um manipulador de fácil reconfigurabilidade, com boa repetibilidade e baixo custo. O detalhamento do sistema foi brevemente introduzido e os desenhos detalhados encontram-se no Apêndice I.

5 Implementação e resultados

Neste capítulo são abordados os detalhes da implementação do projeto, com ênfase na integração dos sistemas mecânicos propostos com os existentes no laboratório e no detalhamento da geração de trajetória para a plataforma móvel dos robôs Tripteron e Triflex II.

A Figura 31a e Figura 31b mostram os protótipos de robô Tripteron e Triflex II desenvolvidos.

(a) (b)

Figura 31 – (a) Robô Tripteron, (b) Robô Triflex II .

A configuração na Figura 31a foi totalmente desenvolvida durante este trabalho e não apresentou problemas de montagem. A configuração na Figura 31b utiliza o sistema de posicionamento e acionamento desenvolvidos neste trabalho e braços e plataforma móvel de um antigo modelo de Triflex II disponível no laboratório. Para montar esta configuração, foi acoplado seu antigo sistema de fixação aos novos sliders projetados, conforme Figura 32.

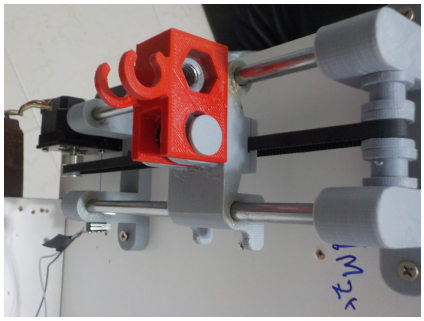


Figura 32 – Detalhe da conexão de um antigo sistema Triflex II aos novos sliders projetados neste trabalho.

5.1 Funções para movimentação da plataforma móvel

Nesta seção é discutida a implementação das funções de movimentação da plataforma pelos Joysticks e a implementação das equações cinemáticas inversas para descrição de uma trajetória desejada.

5.1.1 Movimentação da plataforma pelos Joysticks

Para efetuar a movimentação da plataforma por Joysticks, um dos requisitos de projeto levantado pelo professor Henrique Simas, foi implementada uma rotina em Arduino descrita brevemente a seguir.

Para se poder interpretar as movimentações do joystick como acelerações para os motores de passo e também como sentido de giro, utilizou-se a biblioteca *AccelStepper.h*. Esta biblioteca proporciona a utilização de métodos para a movimentação de até quatro motores de passo, bem como a biblioteca *Bounce2.h* para ativar a chave digital do Joystick.

Para acionar os motores foi criada uma função denominada acionaMotor, mostrada parcialmente na Figura 33. Esta função tem como parâmetros de entrada as leituras das portas ligadas aos Joysticks e é responsável pela conversão dos valores lidos para os limiares de velocidade estabelecidos e pela atuação dos motores. Para isto são utilizados os métodos *setSpeed* e *runSpeed* da biblioteca *AccelStepper.h*.

cnc_Joi_total_v1

```
void acionaMotor(long mapX, long mapY, long mapZ) {

 if (valX <= tresholdDown&& digitalRead(fimDeCursoX) == HIGH) {
    speedX = map(valX, 0, 1023, 400, 0);
    muoviX = true;
  } else if (valX>= tresholdUp ) {
    speedX = -map(valX, 0, 1023, 0, 400);
    muoviX = true;
  } else {
    speedX = 0;
    muoviX = false;
  }
```

Figura 33 – Função para acionar os motores.

A implementação completa em Arduino é mostrada no Apêndice II, Seção 8.1.

5.1.2 Implementação das equações cinemáticas inversas

Nesta seção é discutida a implementação em Arduino das equações cinemáticas inversas, conforme exposto na Seção 3.2.1 para descrição de uma trajetória desejada.

Para impor acelerações para os motores de passo foi novamente utilizada a biblioteca *AccelStepper.h*, porém para garantir a atuação e sincronização de todos os motores foi utilizada também a biblioteca *MultiStepper.h*.

Inicialmente serão discutidas as funções implementadas para auxiliar a determinação cinemática.

- Função *goHome*: leva os motores as coordenadas iniciais do referência da base, sendo que o sistema de referência da base é definido conforme descrito na Seção 3.2.1. A função *goHome* é chamada no ciclo setup e sua implementação é recursiva. Ela aciona os motores até que seja atingido um sinal alto nas portas ligadas aos sensores fim de curso, sinalizando que os sliders estão em sua posição zero. Após executada esta função é chamado o método *setCurrentPosition* da biblioteca *AccelStepper.h* para que seja estabelecido um referencial para os motores de passo.

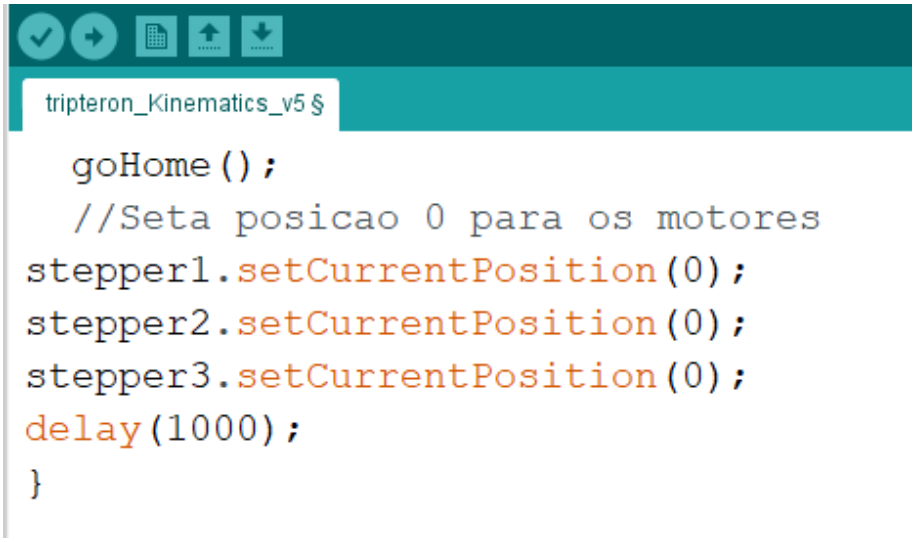


Figura 34 – Chamada da função que leva os sliders a posição inicial, seguida do comando *setCurrentPosition* para estabelecer um referencial para os motores de passo.

- Equações cinemáticas: as equações cinemáticas apresentadas na Seção 3.2.1 são implementadas considerando vetores diretores ortonormais para os sliders, uma vez que se montou o conjunto Tripteron/ Triflex II em uma configuração ortogonal. A Figura 35 também mostra a conversão entre o deslocamento necessário dos sliders para se atingir uma dada posição da plataforma móvel em número de passos necessários para cada motor. Para efetuar esta conversão, deve-se considerar o raio da polia e o número de passos por rotação de cada motor.

Na próxima seção são apresentados os testes efetuados para validar o sistema.


```
//Equacoes cinematicas
S1p1=(Px2+B1x-a10x);
S2p1=(Py2+B2y-a20y);
S3p1=(Pz2+B3z-a30z);

//Conversao de deslocamento para passos
dToSteps=(180/(ang*pi*radius));
  m1p1=S1p1*dToSteps;
  m2p1=S2p1*dToSteps;
  m3p1=S3p1*dToSteps;

//passos desejados
  long positions[3];
  positions[0] = m1p1;
  positions[1] = m2p1;
  positions[2] = m3p1;
  //move e aguarda todos os motores
  steppers.moveTo(positions);
  steppers.runSpeedToPosition();
}
```

Figura 35 – Implementação das equações cinemáticas.

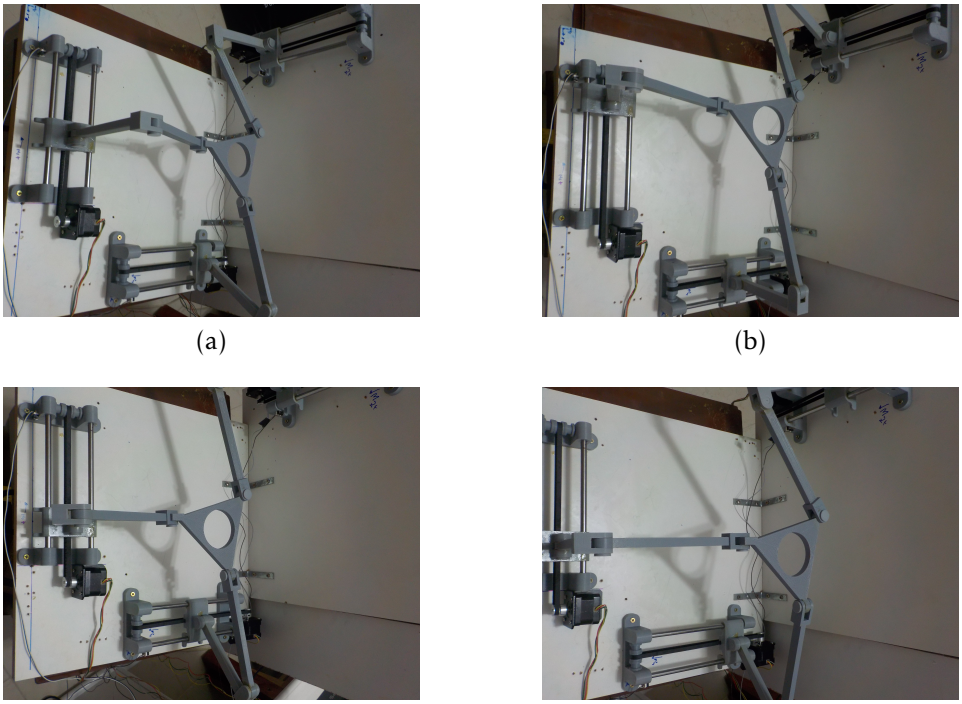
5.2 Testes e validação

Nesta seção são apresentados os testes efetuados com os robôs Tripteron e Triflex II. Estes testes tiveram por objetivo validar os acionamentos e a cinemática inversa dos robôs.

Os testes foram realizados para validar as funções implementadas que são: movimentação independente dos motores através dos joysticks e percorrer uma trajetória teste utilizando a equação da cinemática inversa para a determinação dos parâmetros do manipulador.

Os testes ocorreram dentro do esperado e a avaliação dos mesmos foi apenas qualitativa. Por exemplo, considerando os testes de validação da cinemática inversa, a Figura 36 mostra o robô Tripteron descrevendo uma trajetória helicoidal. Durante este teste, o robô apresentou uma boa repetibilidade, exerceu sua trajetória sem exceder a capacidade de torque dos motores ou atingir singularidades.

Para a validação da cinemática inversa, foram testadas outras trajetórias, como circular, descrição de um quadrado e de um triângulo, tanto para os robôs Tripteron como para o Triflex II. A validação da movimentação comandada por joystick ocorreu conforme o esperado para os dois robôs, sendo comprovado o desacoplamento dos



(a) (b)

(c) (d)

Figura 36 – Robô Tripteron descrevendo uma trajetória helicoidal.

movimentos lineares dos sistemas sliders.

Quanto ao sistema mecânico, apresenta boa repetibilidade tanto para o robô Tripteron quanto para o Triflex II. Porém, como as pernas do robô foram fabricadas em ABS o sistema apresenta folgas e dependendo da disposição dos sliders pode entrar em singularidades não previstas no detalhamento do projeto via software CAD. Porém, este problema não é encontrado para todas as disposições dos sliders, por exemplo, para a configuração na Figura 36 não há este problema.

5.3 Conclusão

Neste capítulo foram mostrados aspectos da implementação mecânica do sistema Tripteron e Triflex II propostos, bem como da implementação em arduino das equações necessárias para que o robô siga uma trajetória desejada ou mesmo possa ser controlado por Joystick. Foi também discutido que os testes efetuados apresentaram bons resultados. Mesmo que a análise destes testes seja apenas qualitativa, pode-se comprovar que o robô apresenta boa repetibilidade, não apresenta problemas na montagem mecânica e apresenta desacoplamento de acionamentos.

6 Conclusões

O presente trabalho foi desenvolvido como parte das atividades do projeto de pesquisa Triflex, do Laboratório Raul Gunther (LAR) da UFSC. Nele, foram desenvolvidos e validados protótipos de robôs Tripteron e Triflex II. Para tanto, foi desenvolvida uma plataforma modular para o acionamento destes robôs constituída por um sistema de transmissão composto de de guias lineares em aço retificado. A atuação do sistema é realizada por uma correia dentada que movimenta as pernas do robô em relação à guia linear.

O projeto detalhado do sistema mecânico foi apresentado na Seção 4.5.1 e os desenhos detalhados dos componentes encontram-se no Apêndice I.

O sistema eletrônico conta com motor de passo Nema 17, driver para motor de passo A4988 e o CNC shield v3, um Arduino Uno, sensores fim de curso e dois Joysticks, sendo que as conexões elétricas entre os componentes foram mostradas na Tabela 3, na Seção 4.5.2.

Para desenvolver o projeto foi adotada a metodologia Projeto Integrado de desenvolvimento de Produtos [PRODIP] (13), culminando num sistema de fácil reconfigurabilidade, com boa repetibilidade, baixo custo e que possibilita o acoplamento de peças de outros protótipos desenvolvidos no laboratório, conforme requisito de projeto.

Além disso, foram desenvolvidas as implementações em Arduino de duas rotinas que permitem o controle da movimentação das plataforma móveis do Tripteron e Triflex II por meio de dois joysticks, bem como a descrição de uma trajetória desejada, através da determinação dos parâmetros cinemáticos necessários. Estas implementações são discutidas na Seção 5.1 e apresentadas em suas totalidade no Apêndice II.

Foram realizados testes que comprovaram que o robô apresenta boa repetibilidade, não apresenta problemas na montagem mecânica e apresenta desacoplamento de acionamentos. Porém, a avaliação dos testes foi apenas qualitativa e visual.

Além disso, como parte dos trabalhos preliminares foi desenvolvido um estudo dos mecanismos autoalinhantes que podem ser derivados do robô Tripteron, deste estudo surgem novos manipuladores que poderão ser acoplados e testados na plataforma desenvolvida.

Deste modo, os resultados esperados foram alcançados em sua totalidade, sendo que a pesquisa encontra-se em evolução.

6.1 Trabalhos futuros

Como sugestões de trabalhos futuros tem-se:

- melhoria da estrutura mecânica ao qual se acoplam os robôs, pois atualmente é composta por duas peças de compensado que apresentam peso elevado;
- estudo e implantação de novos manipuladores à plataforma base;
- desenvolvimento de uma interface de software para o usuário.

Referências

1 LAVAL, L. de Robotique de l'Université. *Tripteron and Quadrupteron*. 2013. Disponível em: <http://robot.gmc.ulaval.ca/en/research/theme104.html>. Citado 4 vezes nas páginas 13, 19, 25 e 34.

2 SIMAS, H.; SIMONI, R.; MARTINS, D. Triflex ii: design and analysis of a self-aligning parallel mechanism with asymmetrical kinematic structure. *Meccanica*, v. 52, n. 11, p. 2991–3002, Sep 2017. ISSN 1572-9648. Disponível em: <https://doi.org/10.1007/s11012-017-0615-3>. Citado 7 vezes nas páginas 13, 19, 34, 45, 46, 47 e 49.

3 HARTENBERG, R.; DENAVIT, J. 1964, kinematic synthesis of linkages, mcgraw-hill, new york. Citado 2 vezes nas páginas 13 e 38.

4 KONG, X.; GOSSELIN, C. M. Kinematics and singularity analysis of a novel type of 3-crr 3-dof translational parallel manipulator. *The International Journal of Robotics Research*, v. 21, n. 9, p. 791–798, 2002. Disponível em: <https://doi.org/10.1177/027836490202100905 01>. Citado 3 vezes nas páginas 13, 43 e 44.

5 FILIPEFLOP. *Motor de Passo NEMA 17 3-20V Impressora 3D*. 2018. Disponível em: <https://www.filipeflop.com/produto/motor-de-passo-nema-17-3-20v-impressora-3d/>. Citado 2 vezes nas páginas 13 e 57.

6 FILIPEFLOP. *Driver Motor de Passo A4988*. 2018. Disponível em: <https://www.filipeflop.com/produto/driver-motor-de-passo-a4988/>. Citado 3 vezes nas páginas 13, 57 e 58.

7 FILIPEFLOP. *CNC Shield V3 para Arduino Impressora 3D*. 2018. Disponível em: <https://www.filipeflop.com/produto/cnc-shield-v3-para-arduino-impressora-3d/>. Citado 4 vezes nas páginas 13, 14, 57 e 58.

8 FILIPEFLOP. *Placa Uno R3*. 2018. Disponível em: <https://www.filipeflop.com/produto/placa-uno-r3-cabo-usb-para-arduino/>. Citado 3 vezes nas páginas 13, 57 e 59.

9 FILIPEFLOP. *Chave Fim de Curso para Impressora 3D com Cabo*. 2018. Disponível em: <https://www.filipeflop.com/produto/chave-fim-de-curso-para-impressora-3d-com-cabo/>. Citado 3 vezes nas páginas 13, 57 e 60.

10 FILIPEFLOP. *Joystick Arduino 3 Eixos*. 2018. Disponível em: <https://www.filipeflop.com/produto/joystick-arduino-3-eixos/>. Citado 4 vezes nas páginas 13, 57, 58 e 60.

11 V1.1, G. *Grbl's Pins*. 2018. Disponível em: <https://github.com/grbl/grbl/wiki/Connecting-Grbl>. Citado 2 vezes nas páginas 14 e 59.

12 RESHETOV, L. *Self-Aligning Mechanism*. 2. ed. [S.l.]: Mir, 1979. Citado 6 vezes nas páginas 19, 27, 28, 29, 30 e 33.

13 ROMANO, L. N. *Modelo de referência para o processo de desenvolvimento de máquinas agrícolas*. Tese (Doutorado) — Universidade Federal de Santa Catarina, Fevereiro 2003. Citado 3 vezes nas páginas 22, 51 e 67.

14 IONESCU, T. G. Iftomm definitions. *Mechanism and Machine Theory*, v. 38, p. 767–776, 2003. Citado na página 23.

15 TSAI, L. W. *The Jacobian Analysis of a Parallel Manipulator Using Reciprocal Screws*. [S.l.], 98. Citado na página 23.

16 GOGU, G. *Structural synthesis of parallel robots*. [S.l.]: Springer, 2008. Citado na página 23.

17 GOGU, G. Mobility of mechanisms: a critical review. *Mechanism and Machine Theory*, n. 40, p. 1068–1097, 2005. Citado 2 vezes nas páginas 24 e 25.

18 OZOL, O. New structural formula for mechanisms and its theoretical and practical importance. *Trans, of Latvian Agricultural Academy, Issue XI*, 1962. Citado na página 26.

19 BLANDING, D. K. *Exact Constraint: Machine Design Using Kinematic Principles*. New York, NY, USA: ASME Press,, 1999. Citado na página 27.

20 DAVIES, T. H. Self-aligning in mechanisms. *Advances in Mechanisms*, v. 1, 1970. Citado na página 27.

21 SZYDLOWSKI, W. M. *Self-Aligning Mechanisms, Forgotten Part of ME Curriculum*. [S.l.], 2000. Citado 2 vezes nas páginas 28 e 29.

22 CARBONI, A. P. *Análise de mecanismos com restrições redundantes através da aplicação da teoria de matroides*. Tese (Doutorado) — Universidade Federal de Santa Catarina, Outubro 2015. Citado na página 31.

23 MARTINS, D.; CARBONI, A. P. Variety and connectivity in kinematic chains. *Mechanism and Machine Theory*, n. 48, p. 1236–1252, 2008. Citado na página 31.

24 CORKE, P. *Robotics Toolbox*. 2018. Disponível em: <http://petercorke.com/wordpress/toolboxes/robotics-toolbox>. Citado 3 vezes nas páginas 38, 41 e 50.

7 Apêndice - projeto detalhado

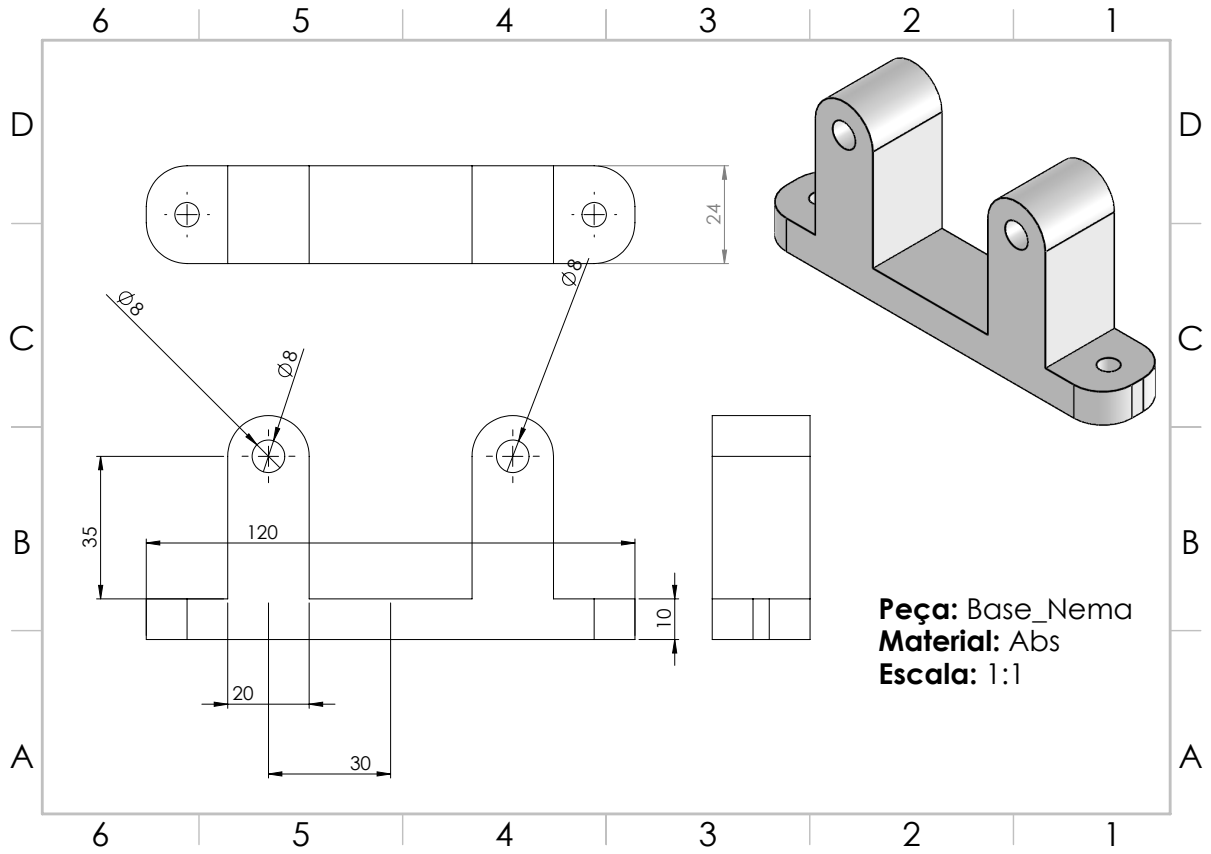


Figura 37 – Detalhamento do sistema de fixação dos conjuntos à base.

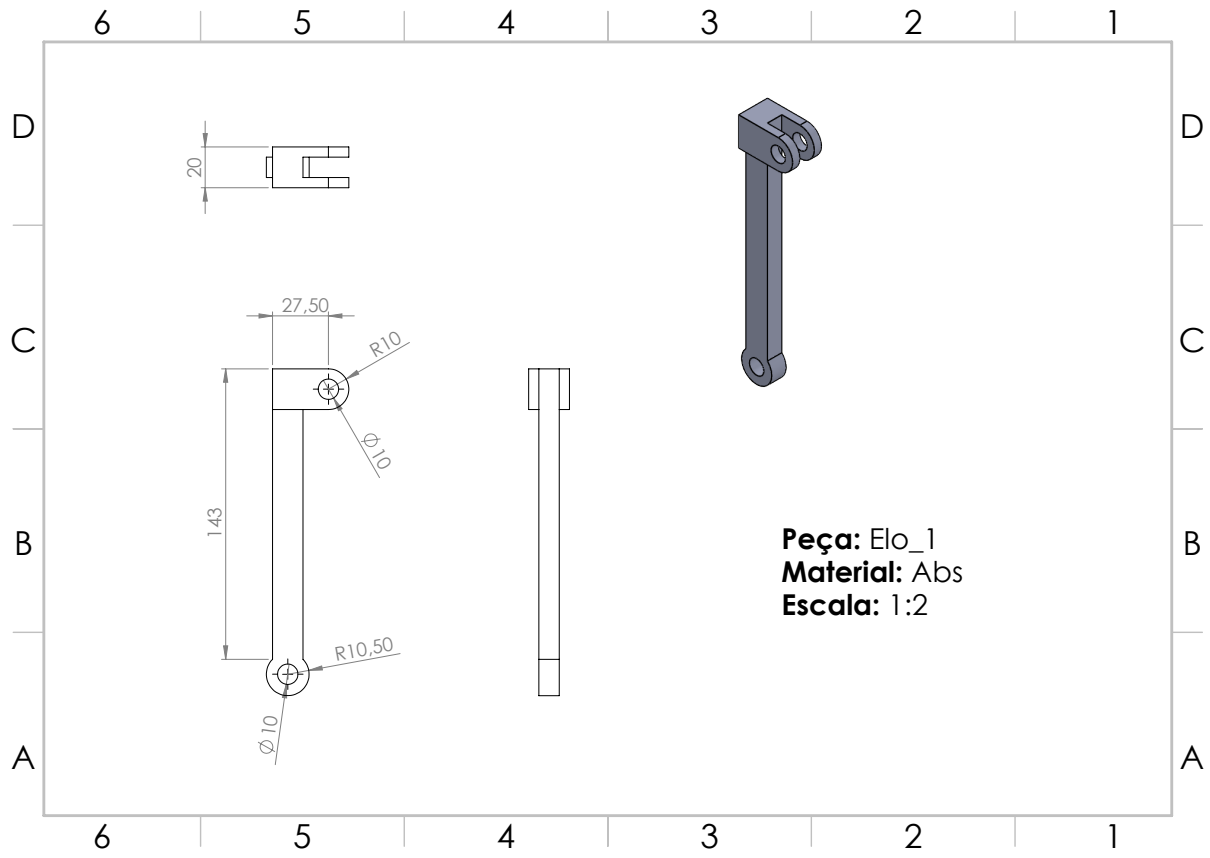


Figura 38 – Detalhamento do primeiro elo de cada perna.

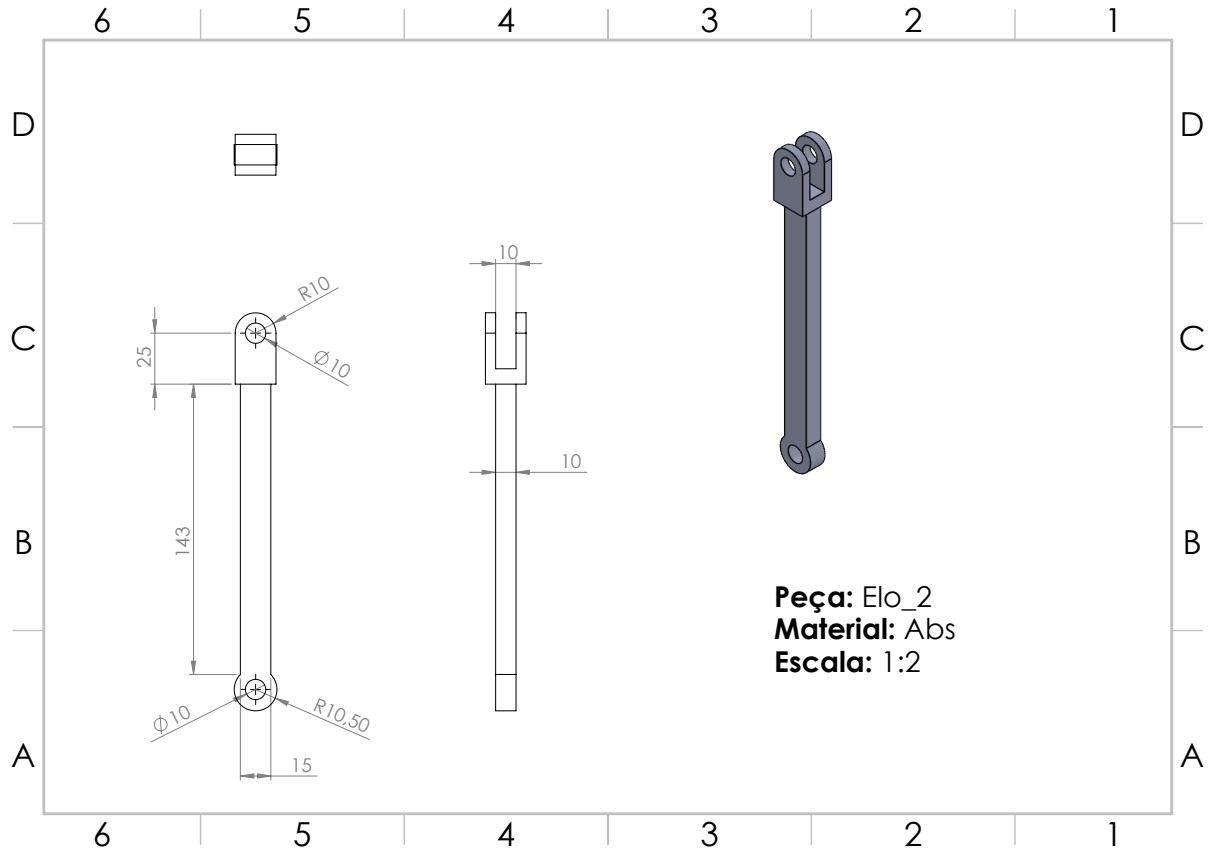


Figura 39 – Detalhamento do segundo elo de cada perna.

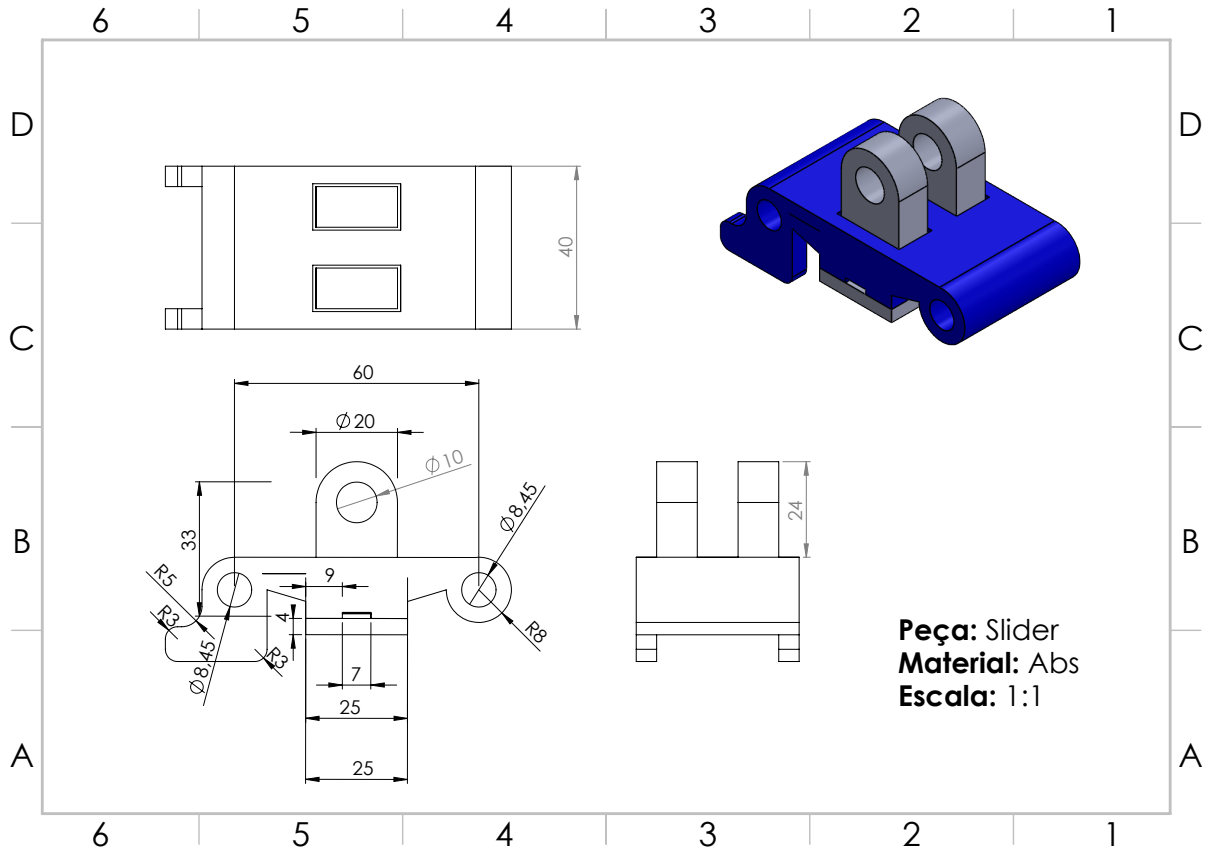


Figura 40 – Detalhamento do slider.

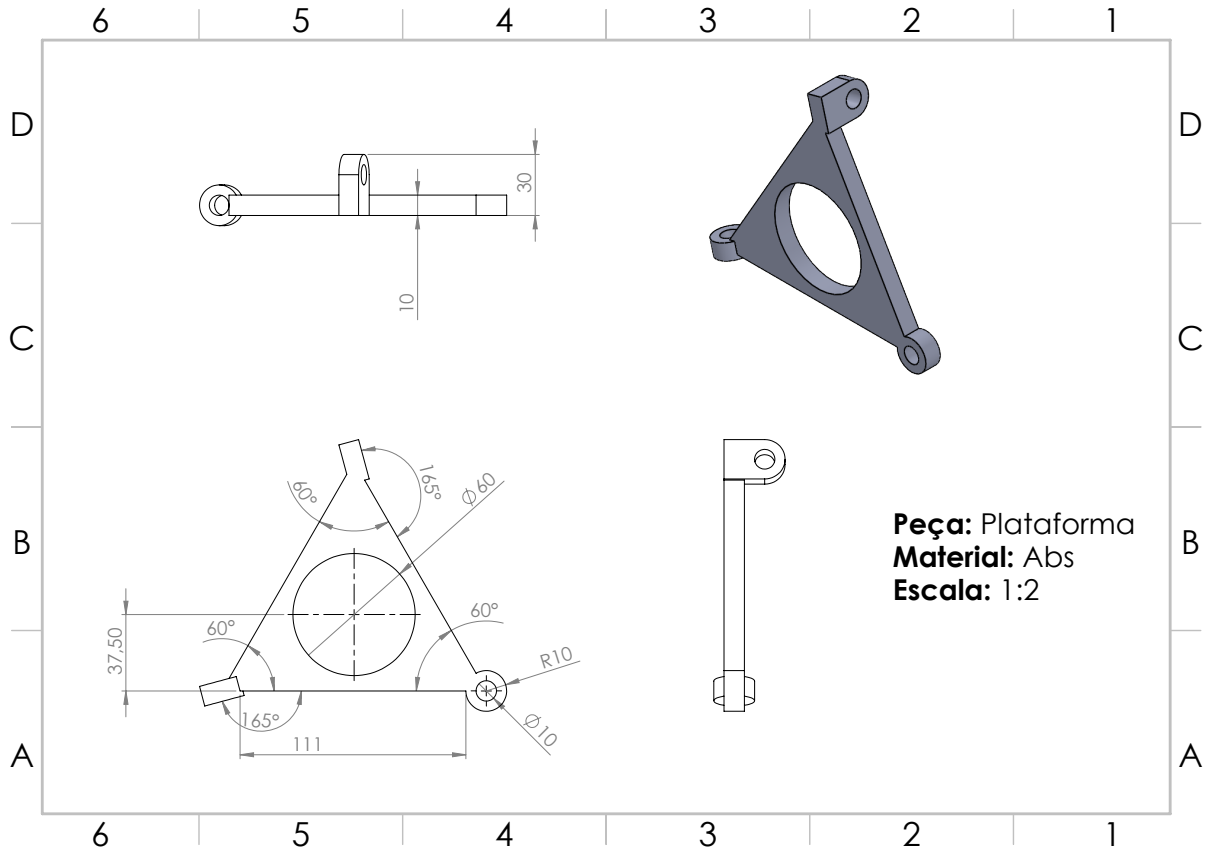


Figura 41 – Detalhamento da plataforma móvel.

8 Apêndice- códigos arduino

8.1 Código arduino para movimentação dos robôs por joystick

Esta seção apresenta o código arduino para movimentação dos robôs Tripteron e Triflex II por joystick, como explicado na Seção 5.1.1.

```
#include <AccelStepper.h>
#include <Bounce2.h>

const int pinEnable = 8;  //pino de ENABLE
const int fimDeCursoX = 9;
const int fimDeCursoY = 10;
const int fimDeCursoZ = 11;

unsigned long debounceDelay = 10; //auxiliar

//Variaveis do motor X
const int jX = A0;  //pino ligado ao joystick 1x
const int stepX = 2;  // STEP X
const int dirX = 5; //direção X
long speedX, valX, mapX;

//Variaveis do motor Y
const int jY = A1;   //pino ligado ao joystick 1y
const int stepY = 3;  // STEP Y
const int dirY = 6; //direção Y
long speedY, valY, mapY;

//Variaveis do motor Z
const int jZ = A2;  //pino ligado ao joystick 2y
const int stepZ = 4;  // STEP Z
const int dirZ = 7; //direção Z
long speedZ, valZ, mapZ;
```

```
//Parametros dos motores
const int maxSpeed = 200;  //velocidade maxima
const int minSpeed = 0; //velocidade minima
const float accelerazione = 50.0; //aceleracao

//parametros de sensibilidade do joystick
const int treshold = 30;
long tresholdUp, tresholdDown;

boolean abilitato, muoviX, muoviY, muoviZ, enable;

Bounce btnEnable = Bounce();

//Declacao dos motores
AccelStepper motoreX(AccelStepper::DRIVER, stepX, dirX);
AccelStepper motoreY(AccelStepper::DRIVER, stepY, dirY);
AccelStepper motoreZ(AccelStepper::DRIVER, stepZ, dirZ);

void setup() {
speedX = speedY = speedZ = 0;
Serial.begin(9600);
  enable = false;
  pinMode( fimDeCursoX, INPUT_PULLUP);
  pinMode( fimDeCursoY, INPUT_PULLUP);
  pinMode( fimDeCursoZ, INPUT_PULLUP);
  pinMode(pinEnable, OUTPUT);
  btnEnable.interval(debounceDelay);
  tresholdDown = (1023/ 2) - treshold;
  tresholdUp = (1023 / 2) + treshold;

  motoreX.setMaxSpeed(maxSpeed);
  motoreX.setSpeed(minSpeed);
  motoreX.setAcceleration(accelerazione);

  motoreY.setMaxSpeed(maxSpeed);
  motoreY.setSpeed(minSpeed);
  motoreY.setAcceleration(accelerazione);
```

```
  motoreZ.setMaxSpeed(maxSpeed);
  motoreZ.setSpeed(minSpeed);
  motoreZ.setAcceleration(accelerazione);
}

void loop() {
//leitura dos joysticks
  valX = analogRead(jX);
  valY = analogRead(jY);
  valZ = analogRead(jZ);
 acionaMotor(mapX, mapY, mapZ);
}

void acionaMotor(long mapX, long mapY, long mapZ) {

 if (valX <= tresholdDown&& digitalRead(fimDeCursoX) == HIGH) {
 //conversao dos dados
    speedX = map(valX, 0, 1023, 400, 0);
    muoviX = true;
  } else if (valX>= tresholdUp ) {
    speedX = -map(valX, 0, 1023, 0, 400);
    muoviX = true;
  } else {
    speedX = 0;
    muoviX = false;
  }

  if (valY <= tresholdDown&& digitalRead(fimDeCursoY) == HIGH ) {
    speedY =map(valY, 0, 1023, 400, 0);
    muoviY = true;
  } else if (valY >= tresholdUp) {
    speedY = -map(valY, 0, 1023, 0, 400);
    muoviY = true;
  } else {
    speedY = 0;
    muoviY = false;
  }

  if (valZ <= tresholdDown&& digitalRead(fimDeCursoZ) == HIGH) {
```

```
    speedZ = map(valZ, 0, 1023, 400, 0);
    muoviZ = true;
  } else if (valZ >= tresholdUp ) {
    //Z va su
    speedZ =  -map(valZ, 0, 1023, 0, 400);
    muoviZ = true;
  } else {
    speedZ = 0;
    muoviZ = false;
  }

  if (muoviX) {
  //pacionamento do motor
    motoreX.setSpeed(speedX);
    motoreX.runSpeed();
  } else {
    motoreX.stop();
  }

  if (muoviY) {
    motoreY.setSpeed(speedY);
    motoreY.runSpeed();
  } else {
    motoreY.stop();
  }

  if (muoviZ) {
    motoreZ.setSpeed(speedZ);
    motoreZ.runSpeed();
  } else {
    motoreZ.stop();
  }}

void checkEnable() {
  btnEnable.update();
  if (btnEnable.fell()) {
    enable = !enable;
  }
  }
```

8.2 Código da cinemática inversa

```
#include <AccelStepper.h>
#include <MultiStepper.h>

AccelStepper stepper1(AccelStepper :: DRIVER,  2, 5);
AccelStepper stepper2(AccelStepper :: DRIVER,  3, 6);
AccelStepper stepper3(AccelStepper :: DRIVER,  4, 7);

MultiStepper steppers;

//Chaves fim de curso
const int fimDeCursoX = 9;
const int fimDeCursoY = 10;
const int fimDeCursoZ = 11;

  //Sliders posicao inicial
  float a10x=0 ;
  float a10y=0 ;
  float a10z=0 ;

  float a20x=285 ;
  float a20y=-135 ;
  float a20z=0 ;

  float a30x=-85 ;
  float a30y=-365 ;
  float a30z=-51 ;

  //Plataforma posicao inicial

  float B1x=20.25;
  float B1y=-75.78;
  float B1z=15 ;

  float B2x=72.28 ;
  float B2y=-16.56 ;
  float B2z=0 ;
```

```
  float B3x=53.87 ;
float B3y=-49.69 ;
float B3z=0 ;

//variaveis do slider
float S1p1;
float S2p1;
float S3p1;

float S1p2;
float S2p2;
float S3p2;

float S1p3;
float S2p3;
float S3p3;

 // Numero de passos
 float m1p1;
 float m2p1;
 float m3p1;

float m1p2;
float m2p2;
float m3p2;

float m1p3;
float m2p3;
float m3p3;


// Desired Positions
float Px1=-100;
float Py1=-200;
float Pz1=-100;
float Px2;
float Py2;
float Pz2;
```

```
 //auxiliar conversao
float ang=1.8;
float pi=3.141516;
float radius=6.5;
float dToSteps;

boolean abilitato, muoviX, muoviY, muoviZ, enable;

void setup() {
 Serial.begin(9600);
 //Velocidades maximas dos motores
 stepper1.setMaxSpeed(400);
 stepper2.setMaxSpeed(400);
 stepper3.setMaxSpeed(400);
 stepper1.setAcceleration(50);
 stepper2.setAcceleration(50);
 stepper3.setAcceleration(50);
 pinMode( fimDeCursoX, INPUT_PULLUP);
 pinMode( fimDeCursoY, INPUT_PULLUP);
 pinMode( fimDeCursoZ, INPUT_PULLUP);
 pinMode(8, OUTPUT);
 steppers.addStepper(stepper1);
 steppers.addStepper(stepper2);
steppers.addStepper(stepper3);
//Leva sliders a posicao 0
goHome();
 //Seta posicao 0 para os motores
stepper1.setCurrentPosition(0);
stepper2.setCurrentPosition(0);
stepper3.setCurrentPosition(0);
delay(1000);
}

void loop() {
Px2=Px1;
Py2=Py1;
delay(100);
double t=0;
```

```
for(t=0;t<=2*pi;t=t+pi/1000){

//Trajetoria Helicoidal
Px2=20*cos(10*t)+Px1;
Py2=20*sin(10*t)+Py1;
Pz2=Pz1-10*t;

S1p1=(Px2+B1x-a10x);
S2p1=(Py2+B2y-a20y);
S3p1=(Pz2+B3z-a30z);

//Conversao de deslocamento para passos
dToSteps=(180/(ang*pi*radius));
  m1p1=S1p1*dToSteps;
  m2p1=S2p1*dToSteps;
  m3p1=S3p1*dToSteps;

//passos desejados
  long positions[3];
  positions[0] = m1p1;
  positions[1] = m2p1;
  positions[2] = m3p1;

  //move e aguarda todos os motores
  steppers.moveTo(positions);
  steppers.runSpeedToPosition();
}
}

void goHome(){
  if(digitalRead(fimDeCursoX) == HIGH){
     muoviX = true;
}
else{
    muoviX = false;
  }
  if(digitalRead(fimDeCursoY) == HIGH){
     muoviY = true;
}
```

```
else{
    muoviY = false;
  }
  if(digitalRead(fimDeCursoZ) == HIGH){
    muoviZ = true;
}
else{
    muoviZ = false;
  }
 if (muoviX) {
   stepper1.setSpeed(400);
   stepper1.runSpeed();
  } else {
     stepper1.stop();
  }
  if (muoviY) {
   stepper2.setSpeed(400);
   stepper2.runSpeed();
  } else {
     stepper2.stop();
  }
  if (muoviZ) {
   stepper3.setSpeed(400);
   stepper3.runSpeed();
  } else {
     stepper3.stop();
  }
  delay(10);
  if(muoviX||muoviY||muoviZ)
  {
     goHome();
  }}
```