UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FERNANDO RAFAEL ARNHOLD DA SILVA

PROJETO E CONSTRUÇÃO DE UM BRAÇO ROBÓTICO SCARA COM CONTROLE DA CINEMÁTICA

Recife
2019

FERNANDO RAFAEL ARNHOLD DA SILVA

PROJETO E CONSTRUÇÃO DE UM BRAÇO ROBÓTICO SCARA COM CONTROLE DA CINEMÁTICA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração: Processamento de Energia

Orientador: Prof. Dr. Fabrício Bradaschia

Coorientador: Prof. Dr. Marcio Evaristo da Cruz Brito

Recife

2019

Catalogação na fonte
Bibliotecária Maria Luiza de Moura Ferreira, CRB-4 / 1469

S586p Silva, Fernando Rafael Arnhold da.
Projeto e construção de um braço robótico SCARA com controle da cinemática / Fernando Rafael Arnhold Silva. - 2019.
139 folhas, il., tab.

Orientador: Prof. Dr. Fabrício Bradaschia.
Coorientador: Prof. Dr. Marcio Evaristo da Cruz Brito.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Elétrica, 2019.
Inclui referências e apêndices.

1. Engenharia Elétrica. 2. Ambientes virtuais. 3. Braços robóticos. 4. CAE. 5. Cinemática. 6. Construção. 7. SCARA. I. Bradaschia, Fabrício (Orientador). II. Brito, Marcio Evaristo da Cruz. III. Título.

UFPE

621.3 CDD (22. ed.) BCTG/2020-10

FERNANDO RAFAEL ARNHOLD DA SILVA

“PROJETO E CONSTRUÇÃO DE UM BRAÇO ROBÓTICO SCARA COM CONTROLE DA CINEMÁTICA”

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Aprovada em: 05 de julho de 2019.

BANCA EXAMINADORA

_____

Prof. Fabricio Bradaschia
(Orientador e Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. Dr. Gustavo Medeiros de Souza Azevedo
(Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. Dr. Marcio Evaristo da Cruz Brito
(Coorientador e Examinador Externo)
Universidade Federal de Pernambuco

_____

Prof. Dr. João Marcelo Xavier Natário Teixeira
(Examinador Externo)
Universidade Federal de Pernambuco

RESUMO

Sabe-se que a robótica esta cada vez mais presente nas indústrias do mundo todo, pois um robô é capaz de executar tarefas com precisão, com maior velocidade que seres humanos, não tem problemas de trabalhar em locais insalubres e não é afetados por tarefas repetitivas. Em suma, robôs industriais geram mais lucro, competitividade e maior qualidade nos produtos e serviços. Por outro lado, tais robôs possuem custo elevado e solicitam um alto grau de conhecimento para serem operados. Além disso, o Brasil é extremamente atrasado no uso da robótica na suas indústrias, o que se reflete no cenário atual de escassez de pesquisas acadêmicas em braços robóticos, em especial, o SCARA (*Selective Compliance Assembly Robot Arm* - Braço Robótico para Montagem com Conformidade Seletiva). O SCARA é um braço robótico com grande presença na indústria mundial, justificando sua importância. Assim, é de primordial importância que trabalhos focados no projeto e controle de braços robóticos SCARA sejam desenvolvidos no Brasil, servindo como motivação para que mais pessoas desenvolvam pesquisas na área. Seguindo este princípio, este trabalho tem como objetivo o projeto, controle e construção de um SCARA de forma precisa, didática e detalhada. Durante a pesquisa, são utilizados softwares CAE (*Computer Aided Engineering* - Engenharia Assistida por Computador) (SolidWorks) e ambientes virtuais (V-REP) para definir e validar os resultados encontrados em cada etapa. Durante o desenvolvimento da cinemática, foi realizado um conjunto de simulações que abordam toda uma metodologia que define a realização de movimentos simples até a realização de movimentos complexos. Com o protótipo construído, realizou-se vários experimentos, verificando a importância de cada uma das etapas executadas no trabalho até a obtenção do protótipo e validando a sua capacidade de realizar as mais diversas tarefas complexas com alta precisão.

Palavras-chave: Ambientes virtuais. Braços robóticos. CAE. Cinemática. Construção. SCARA.

ABSTRACT

It is known that robotics is increasingly present in industries round the world, because a robot is able to perform tasks accurately, faster than humans, has no problem working in unhealthy places and is unaffected by repetitive tasks. In short, industrial robots generate more profit, competitiveness and higher quality products and services. On the other hand, such robots have a high cost and require a high degree of knowledge to be operated. In addition, Brazil is extremely behind in the use of robotics in its industries, which is reflected in the current scenario of scarcity of academic research in robotic arms, in particular, the SCARA (Selective Compliance Assembly Robot Arm Selective). SCARA is a robotic arm with great presence in the world industry, justifying its importance. Thus, it is of paramount importance that works focused on the design and control of SCARA robotic arms be developed in Brazil, serving as motivation for more people to develop research in the area. Following this principle, this work aims at the design, control and construction of a SCARA in a precise, didactic and detailed way. During the research, CAE (Computer Aided Engineering) software (SolidWorks) and virtual environments (V-REP) are used to define and validate the results found in each step. During the development of the kinematics, a set of simulations was carried out, covering a whole methodology that defines the realization of simple movements until the accomplishment of complex movements. With the prototype built, experiments were carried out with several successful experiments, verifying the importance of each of the steps performed in the work until obtaining the prototype and validating its ability to perform the most diverse complex tasks with high precision.

Keywords: CAE. Construction. Kinematics. Robotic arms. SCARA. Virtual environments.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

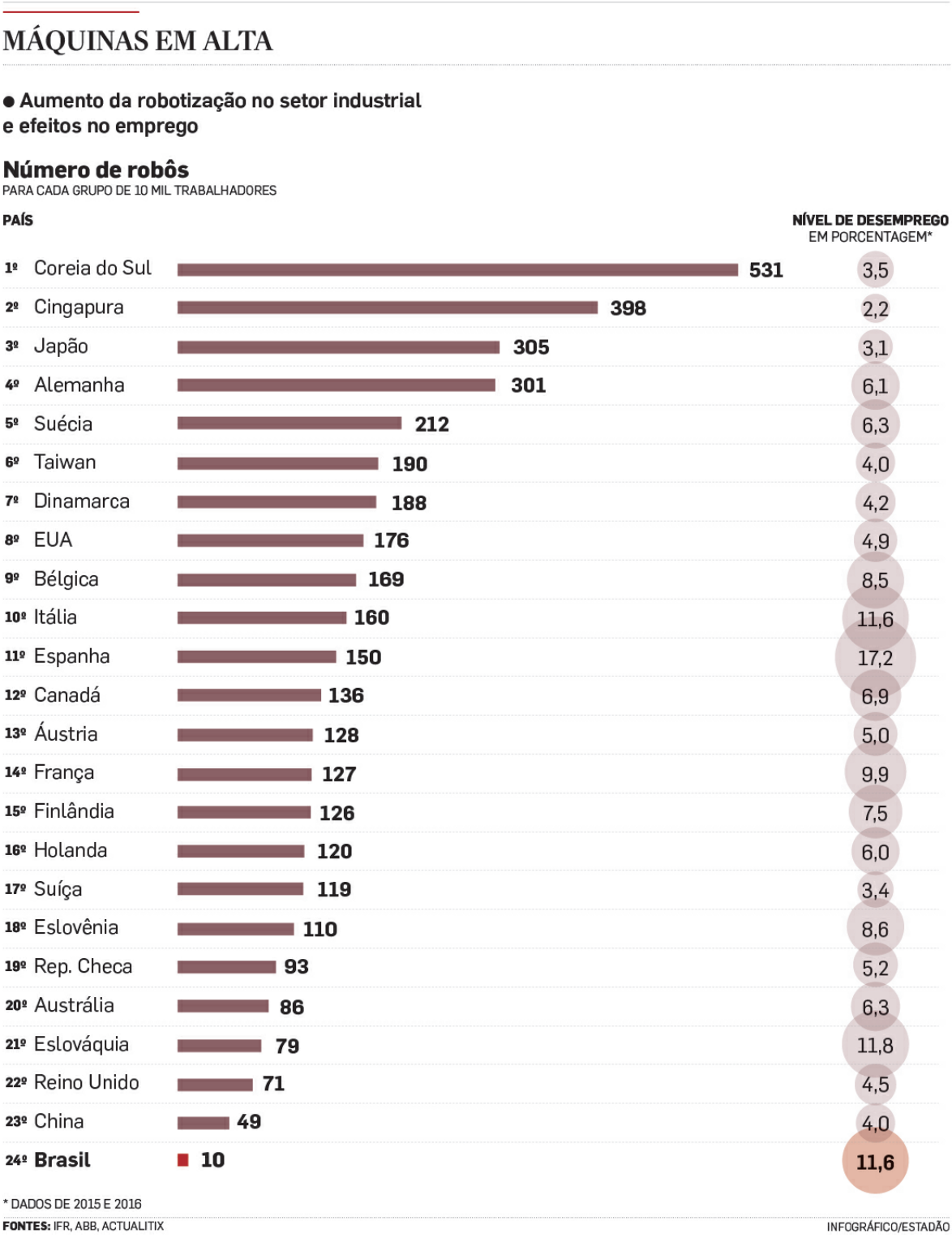
1 INTRODUÇÃO

A robótica se torna importante nas indústrias visto que os robôs trabalham incessantemente, têm uma alta precisão, realizam tarefas mais rápidas do que seres humanos e não são afetados por tarefas repetitivas ou ambientes insalubres. Por outro lado, tais robôs possuem um preço muito elevado e demandam um alto grau de conhecimento para colocá-los em operação. Além disso, o Brasil é extremamente atrasado no uso da robótica e um fato que justifica isto é que precisaríamos instalar cerca de 165.000 robôs industriais para nos aproximarmos da densidade robótica atual da Alemanha (FILHO, 2016). Para se ter uma ideia, o Brasil demoraria 100 anos para chegar nessa densidade, pois o ritmo atual de instalações é de apenas 1.500 robôs por ano. Na Figura 1 é evidenciado a importância da densidade robótica ao mostrar a densidade robótica dos países é a sua porcentagem de desemprego. Para minimizar estes problemas e elevar a eficiência do trabalho, é necessário o desenvolvimento de novas pesquisas, fazendo com que o uso destas novas tecnologias se tornem mais acessíveis, adequadas e otimizadas para as tarefas que são realizadas no Brasil (NIKU, 2015).

Os sistemas robóticos estão presentes nas indústrias, comércios, hospitais e em muitos outros ambientes. Estes sistemas podem ser classificados quanto ao controle, mobilidade da base, estrutura cinemática e espaço de trabalho. É importante ressaltar que estas classificações influenciam na estrutura mecânica, velocidade, precisão, regiões de trabalho, graus de liberdade, aplicação, complexidade e em muitas outras características (SANTOS *et al.*, 2016). Basicamente, o estudo da robótica é divido em robôs de base móvel e robôs de base fixa. A robótica móvel estuda os robôs que podem se movimentar livremente pelos ambientes aquáticos, aéreos e terrestres. Alguns exemplos de robôs moveis são seguidores de linha, drones e pequenos navios autônomos. A robótica fixa estuda robôs que têm sua mobilidade limitada e apenas podem interagir com o ambiente através da movimentação de seus elos. Alguns exemplos de robôs de mobilidade fixa são os manipuladores da Kuka e os manipuladores da ABB (KUKA, 2019). Na Figura 2 podem ser vistos algumas estruturas típicas de sistemas robóticos.

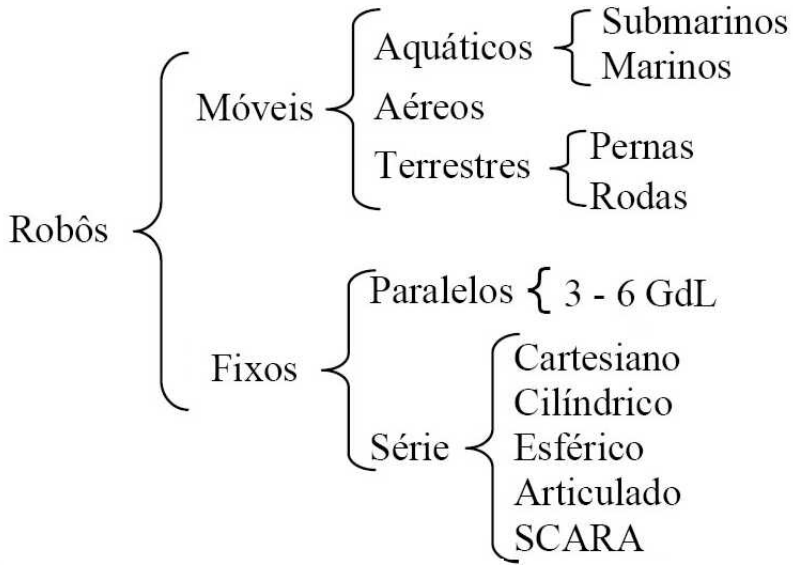
Sabe-se que, na indústria, os braços robóticos mais presentes são os de mobilidade fixa, nos quais se destacam os robôs cartesianos, cilíndricos, esféricos, articulados e o SCARA (*Selective Compliance Assembly Robot Arm* - Braço Robótico para Montagem

Figura 1 – Aumento na robotização no setor industrial e efeitos no emprego.



Fonte: ESTADÃO (2017).

Figura 2 – Estruturas típicas de sistemas robóticos.

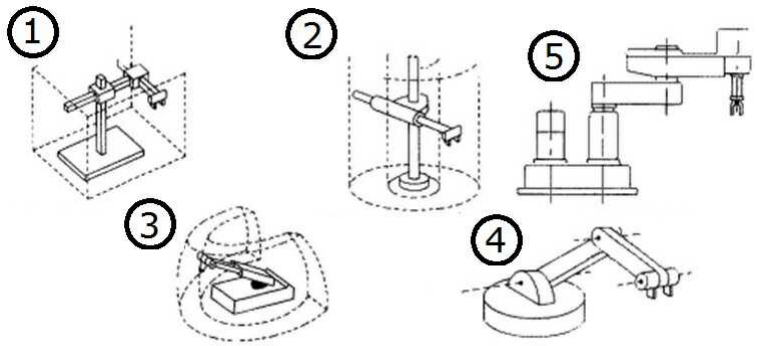


Fonte: Adaptado de RIASCOS (2019).

com Conformidade Seletiva). Cada um destes possui características únicas que definem sua estrutura, aplicação, precisão, etc. Na Figura 3, podem ser vistas algumas ilustrações dos braços robóticos citados.

Dentre os robôs de mobilidade fixa, SCARA é aquele que tem sua orientação paralela à região de trabalho, apresentando grande compactação e realizando movimentos com extrema agilidade. Este tipo de braço robótico tem a capacidade de modificar facilmente sua forma para realizar as mais diversas tarefas e é muito presente em tarefas de manipulação de objetos de um local para outro em fábricas de fármacos, circuitos impressos, automobilísticas, etc (YAMAHA, 2019).

Figura 3 – Ilustrações de braços robóticos.



Fonte: PLASCHKES (2019).
(1) Cartesianos; (2) Cilíndricos; (3) Esféricos; (4) Articulados; (5) SCARA.

1.1 ESTADO DA ARTE

Uma pesquisa bibliográfica mostra que existem diversos trabalhos internacionais sobre o SCARA que geram muitas contribuições acadêmicas. Existe um trabalho que estuda a repetibilidade do SCARA operando com uma caixa de engrenagens planetárias (MOUSAVI *et al.*, 2015), porém engrenagens planetárias são extremamente caras, dificultando seu uso em protótipos de pequeno porte em pesquisas no Brasil. Outro trabalho apresenta o projeto e construção de um robô SCARA com seis graus de liberdade (URREA; CORTÉS; PASCAL, 2016) cujo trabalho demanda uma alta complexidade e elevados custos para o desenvolvimento do protótipo. Em (PRADHAN; RAJARAJAN; SHETTY, 2018) é realizo a prototipação, emulação, implementação e avaliação do SCARA em um processo industrial cujo trabalho requisita de uma avaliação bastante trabalhosa e com custo bem elevado, visto que construir um braço robótico SCARA e avaliar seu desempenho em tarefas industrias reais demandam um grande número de aquisições de equipamentos e de matéria prima, confecções de peças e de muito estudo. Todos estes trabalhos demonstram o excelente nível da pesquisa internacional e que dificilmente seriam realizadas no Brasil.

Analisando diversos trabalhos nacionais, foi verificado um trabalho que estuda a modelagem matemática do SCARA com acionamentos pneumáticos cujo trabalho desenvolve excelentes modelos não-lineares (BEHNEN, 2017). Em (MELLO, 2016) é feito todo o desenvolvimento de um SCARA no qual realiza importantes experimentos seguindo a norma ISO 9283 e desenvolve um controle usando a lógica nebulosa (fuzzy) com bons resultados. Em (PEREIRA; SILVA; THÉ, 2014) é realizo o estudo do planejamento de trajetórias de um SCARA para não colisão usando câmeras cujo trabalho é pioneiro em usar câmeras para proporcionar um caminho livre de colisão para o SCARA e mostra resultados preliminares que podem ser melhorados. Em (PADOIN, 2011) é estudo a dinâmica da não-linearidade das folgas em uma junta rotativa no SCARA no qual realiza o desenvolvimento na modelagem das juntas rotativas. Em (VARGAS, 2015) é desenvolvido uma plataforma de testes de controladores para o SCARA cujo trabalho realiza uma análise matemática e obtém bons resultados com o controle de posição do servomotor. Em (SCHLUTER, 2018) é desenvolvido o estudo de um controle não-linear adaptativo com compensações de atrito para um SCARA com acionamento pneumático no qual é completo e extenso mas, não realiza experimentos com um protótipo.

Durante essa pesquisa bibliográfica, percebe-se que existem uma quantidade limitada de trabalhos acadêmicos que estudam sobre os braços robóticos SCARA, em especial, no Brasil. Após uma análise dos resultados obtidos nos trabalhos, percebe-se que vários deles possuem problemas, como imprecisões e falhas na operação, e limitações de operação e aplicação. Além disso, algumas tecnologias e temáticas importantes não são abordadas em suas pesquisas, como, por exemplo:

1. Uma pequena quantidade delas utilizou ambientes CAE (*Computer Aided Engineering* - Engenharia Assistida por Computador) e ambientes de simulação virtual para sustentar seus resultados;
2. Nenhuma delas realizou experimentos que verificam a execução de tarefas complexas pelo braço desenvolvido;
3. Em todas elas, faltou uma metodologia didática, detalhada e precisa para o desenvolvimento da cinemática de um SCARA (desde aplicações de cinemática direta e inversa à elaboração de trajetórias complexas);
4. Nenhuma delas elaborou um protótipo que tivesse uma precisão adequada para tarefas industrias e tampouco demonstrou a precisão experimentalmente.

1.2 OBJETIVO GERAL

Devido à escassez de pesquisas acadêmicas que abordem projeto, controle e construção do SCARA no Brasil de forma didática, precisa e detalhada, o objetivo deste trabalho é realizar o projeto e a construção de um braço robótico SCARA e de algoritmos de controle que permitam realizar a cinemática direta e inversa de forma otimizada, com grande precisão, satisfazendo os requisitos dinâmicos do robô e tendo uma interface amigável ao usuário. Para esse desenvolvimento, serão utilizados softwares CAE e ambientes de simulação virtual, como um passo fundamental para sustentar os resultados obtidos. Espera-se que este trabalho forneça conhecimentos suficientes para que outros pesquisadores possam evoluir no controle de braços robóticos SCARA, inclusive em aplicações industriais e comerciais, ampliando o conhecimento na área de robótica no Brasil.

1.3 MATERIAIS E MÉTODOS

O desenvolvimento desta pesquisa é realizado por meio de análise de revisões bibliográficas, acesso a livros, manuais de softwares e simulações em ambientes CAE. Estas simulações CAE são realizadas nos softwares MATLAB, SOLIDWORKS e V-REP. O objetivo destas simulações é analisar o comportamento da estrutura do braço robótico e, a partir destas, definir a estrutura mecânica do sistema, além de elaborar um sistema de controle eficiente e com custo computacional adequado para controlar todos os movimentos do braço robótico.

Com as simulações realizadas no MATLAB, é possível verificar as convergências e o custo computacional dos métodos de resolução da cinemática inversa. A partir da simulação no SOLIDWORKDS, é possível definir, de forma qualitativa, onde são concentrados os maiores esforços e deformações no braço robótico para que estes lugares possam receber os devidos cuidados. Com as simulações no V-REP, é possível verificar os efeitos do controle no movimento do sistema robótico, incluindo tarefas complexas. Assim, com todas as simulações realizadas, o protótipo será confeccionado com as especificações definidas. A partir do protótipo montado, é elaborado o sistema de controle, a IHM (Interface Homem-Maquina) e o protocolo que realiza a comunicação entre a IHM e o sistema de controle. Ao final, são realizados experimentos com o protótipo, de forma a comparar seu desempenho com as simulações realizadas no MATLAB e no ambiente virtual V-REP.

1.4 ORGANIZAÇÃO TEXTUAL

No Capitulo 2 são expostos alguns fundamentos teóricos relacionados à estrutura típica de um braço robótico, além de informações sobre o SCARA, sobre os métodos matemáticos relacionados ao cálculo da cinemática inversa e os conceitos das simulações de estrutura e virtualização.

No Capitulo 3 são realizadas a definição do desenho da estrutura, a análise qualitativa do estresse na estrutura robótica e as simulações no ambiente virtual V-REP.

No Capitulo 4 é exposta a metodologia empregada no processo de produção da estrutura do braço robótico, do sistema de controle, da comunicação realizada e da IHM.

No Capitulo 5 são apresentados os resultados experimentais com o protótipo do braço SCARA e seu desempenho é comparado com os resultados obtidos por simulação.

No Capitulo 6 são apresentadas as conclusões gerais do trabalho, além de sugestões de trabalhos futuros que possam dar continuidade ao desenvolvimento deste tipo de braço robótico no Brasil.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são expostos os conceitos fundamentais para o desenvolvimento do braço robótico, como a descrição da estrutura básica do SCARA, suas principais aplicações, os métodos matemáticos que descrevem seu movimento, a análise estrutural por elementos finitos e os principais ambientes de simulação virtual.

2.1 ESTRUTURA BÁSICA DE UM BRAÇO ROBÓTICO

Todo braço robótico é um sistema que contém a estrutura mecânica, os atuadores, efetuadores, sensores, sistema de controle e o software. Com relação à estrutura mecânica, apesar de existirem vários tipos de braços robóticos é possível verificar, em todos, um conjunto de corpos rígidos denominados elos, um conjunto de articulações denominadas juntas (articulações) e os sistemas de transmissão. Na Figura 4 podem ser vistos alguns desses elementos em uma ilustração do braço robótico do tipo SCARA.

Na Figura 4, os elos são as bases, os braços e quaisquer estruturas rígidas que estejam entre duas juntas. As juntas são as articulações que interligam os elos, dando liberdade de movimento para o braço.

Os robôs podem ter juntas lineares, rotativas ou esféricas.

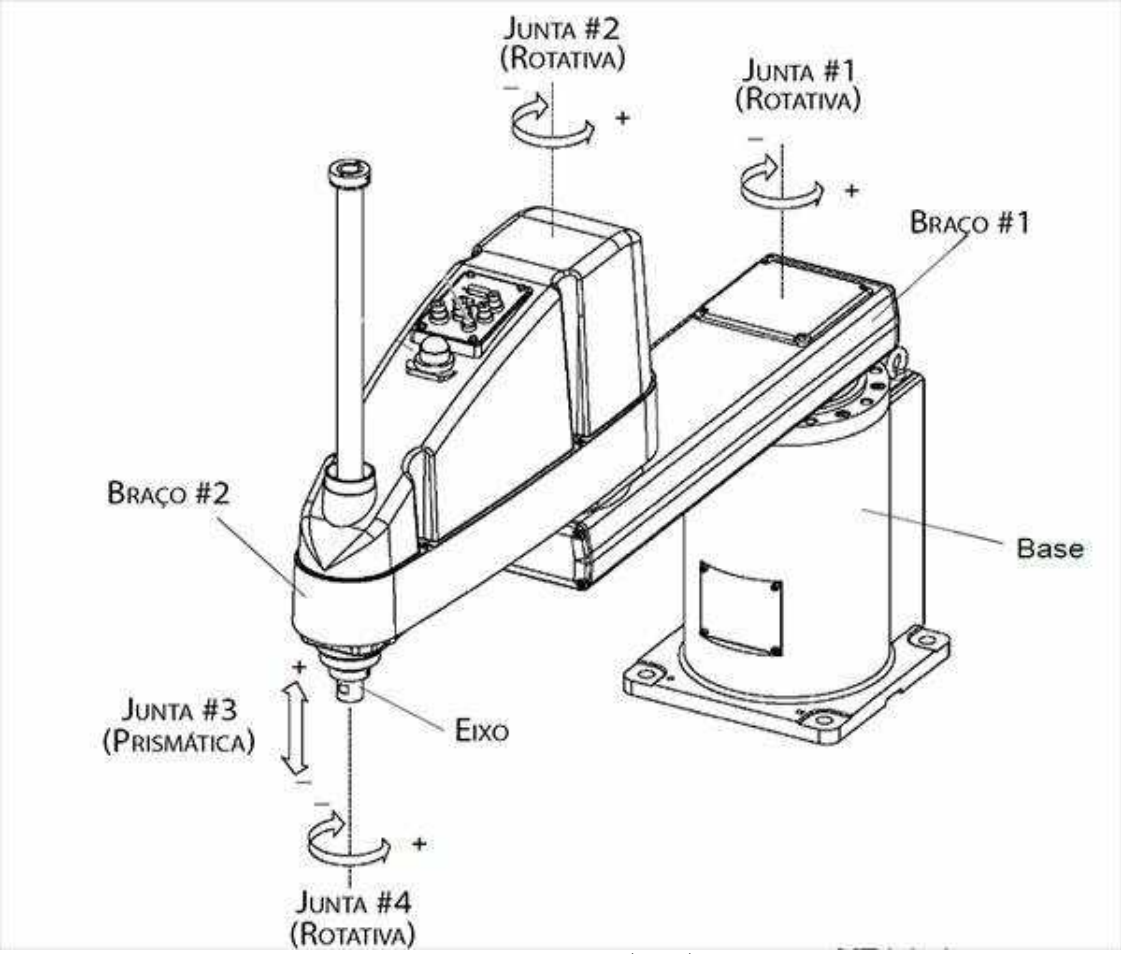
Juntas lineares ou juntas prismáticas são aquelas que permitem apenas o movimento linear entre dois elos. Essas articulações geralmente são acionadas por cilindros hidráulicos ou pneumáticos ou por atuadores elétricos lineares.

Juntas rotativas são aquelas que permitem apenas o movimento de dois elos adjacentes por meio de rotação em um único eixo. As articulações rotativas geralmente são acionadas por servomotores ou por motores de passo.

As juntas esféricas são aquelas que permitem movimentos rotativos em todos os eixos. As articulações esféricas geralmente não são acionadas visto que para serem acionadas seria necessário utilizar três atuadores independentes por junta, o que torna o sistema mais complexo e caro.

Com relação aos atuadores, estes representam os “músculos” dos braços robóticos. Sua função é realizar o movimento das articulações com base em comandos fornecidos pela unidade de controle. Os tipos mais comuns de atuadores em braços robóticos são

Figura 4 – Braço robótico SCARA com elos e juntas em destaque.



Fonte: ROSSINI (2018).

motores de passo, servomotores, atuadores pneumáticos e atuadores hidráulicos. Alguns desses motores podem ser vistos na Figura 5.

Com relação aos efetuadores, estes representam as "mãos" dos braços robóticos. Sua função é interagir com o ambiente. Os tipos mais comuns de efetuadores são os manipuladores, os soldadores, os pintores. Alguns exemplos de efetuadores podem ser vistos na Figura 6.

Com relação aos sensores, estes representam os "sentidos" do robô. É apenas a partir destes que o robô consegue extrair informações do ambiente para executar suas tarefas. Vale ressaltar que existem muitos braços robóticos que não possuem nenhum sensor e estes são intitulados como robôs que funcionam em malha aberta.

Com relação aos controladores, estes representam o "cérebro" do robô. Alguns exemplos de unidades de controle são os CLPs (Controladores Lógicos Programáveis), os DSPs (*Digital Signal Processors* - Processadores Digital de Sinais) e os microcontroladores. O objetivo de uma unidade de controle é receber parâmetros e sinais do ambiente e,

Figura 5 – Ilustração de alguns atuadores.

Fontes: TEIXEIRA; AMAZON; SILVA (2018).
(1) Motor de passo; (2) Servomotor; (3) Acionamento pneumático.

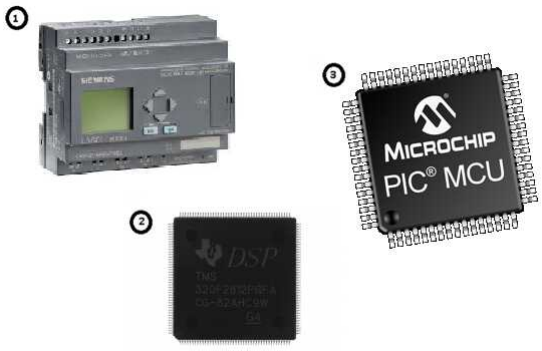
Figura 6 – Ilustração de alguns efetuadores.

Fontes: SMART; ROBOTICWELDINGSCELLS; YASKAWA(2019).
(1) Soldador; (2) Manipulador; (3) Pintor.

com base nestes, transmitir sinais para os atuadores (motores). Alguns exemplos de controladores podem ser vistos na Figura 7.

Os firmwares são os algoritmos executados pelos controladores. Fazendo uma analogia, podem ser considerados a forma como as ideias são organizadas no "cérebro" do robô. O objetivo destes firmwares é definir como os controladores vão processar as informações do ambiente e como estes vão transmitir a tomada de decisão para os atuadores.

Figura 7 – Ilustração de alguns controladores:



Fontes: MARCIO; AUTOELETRICA; CHIP1STOP (2018).
(1) CLP; (2) DSP; (3) Microcontrolador.

2.2 ROBÔ SCARA: DEFINIÇÕES, APLICAÇÕES E MODELO MATEMÁTICO

O braço robótico SCARA foi criado com a capacidade de modificar facilmente sua forma para realizar as mais diversas tarefas. Este braço robótico foi desenvolvido por Hiroshi Makino, um professor da Universidade de Yamanashi. Na sua concepção original, o SCARA tinha seu eixo Z (vertical) rígido e os eixos XY (plano horizontal) móveis, o que lhe tornou uma excelente solução para sistemas de montagem industriais (BANERJEE; ANANYA, 2017). Na Figura 8, é possível verificar um exemplo de um SCARA.

Figura 8 – Braço robótico SCARA com dois eixos móveis e um eixo fixo.

Fonte: EMOTIONSUPPLY (2018).

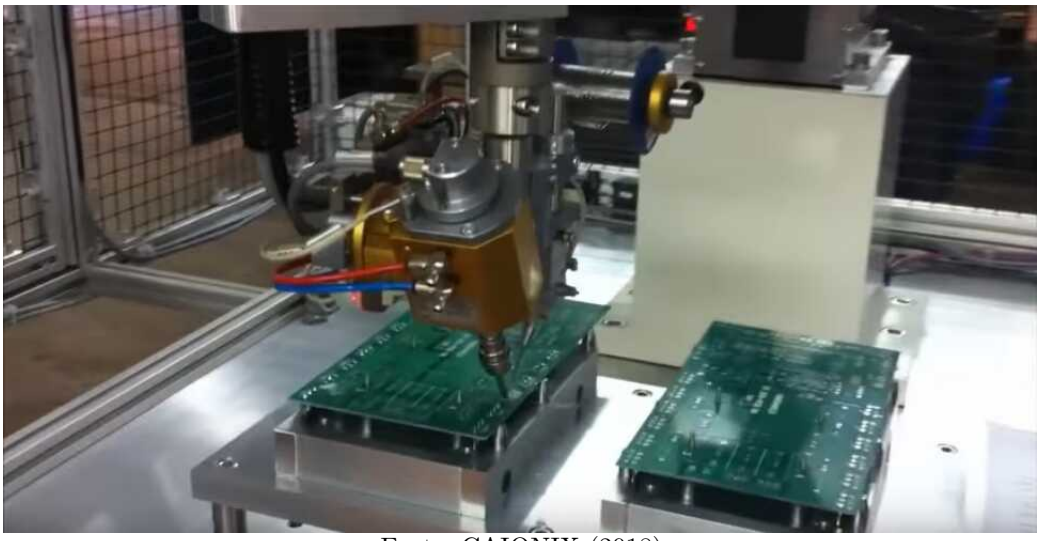
Os braços robóticos SCARA são amplamente aplicados em tarefas como pegar e colocar (*pick and place*), balançar misturas, girar objetos e manipular ferramentas. Aplicações típicas de pegar e colocar realizadas pelos robôs SCARA envolvem fixação de componentes em placas de circuito impresso e manipulação de objetos de um local ao outro. Tarefas de balançar são aplicadas em indústrias ou laboratórios que manipulam compostos químicos e precisam que o braço realize uma certa mistura de elementos. Tarefas

de girar são aplicadas a sistemas em que um objeto está em uma região em uma certa posição e precisa ser manipulado de tal forma a ter uma outra orientação. Manipulação de ferramentas envolve tarefas como soldagem de componentes, aplicações de depósito de resina para realização de peças em 3D, girar parafusos e muitas outras. O SCARA é um dos braços mais populares no setor industrial devido ao grande leque de aplicações, ao pouco espaço ocupado, ao custo mais baixo que braços semelhantes, além de ter uma estrutura robusta, realizar tarefas com uma velocidade impressionante e apresentar uma grande precisão de movimentos. Nas Figuras 9-12 são apresentadas algumas das aplicações do braço robótico SCARA.

Figura 9 – Aplicação do SCARA na produção de produtos hospitalares.

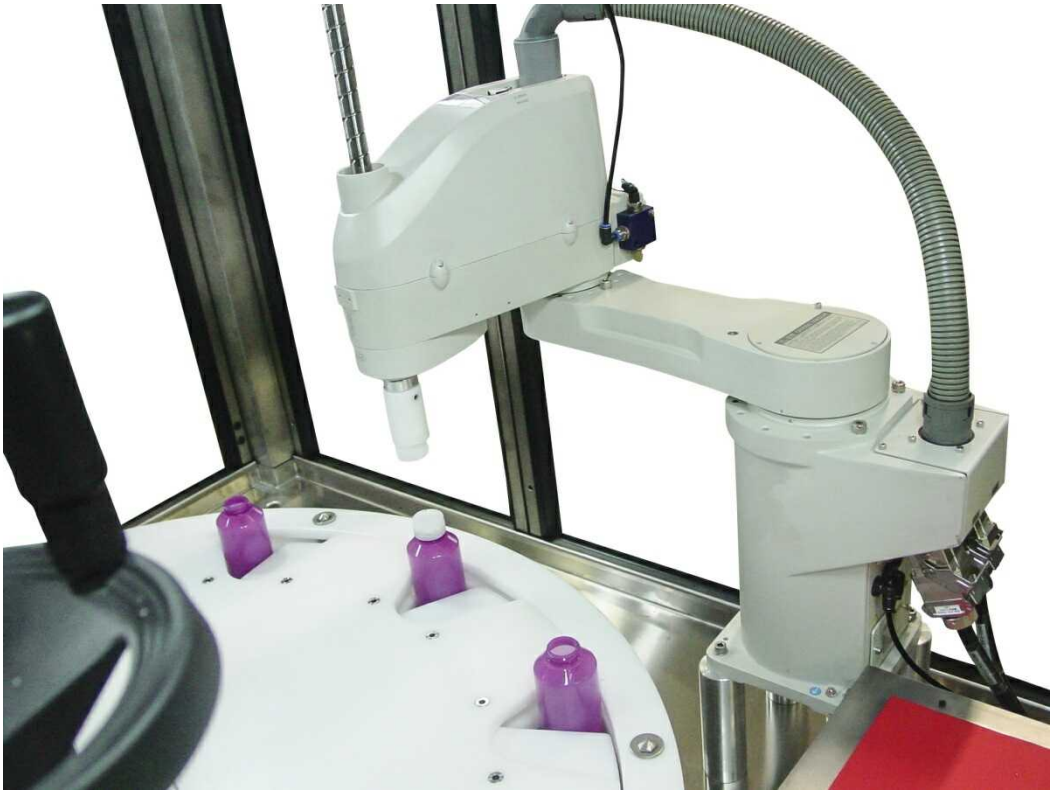
Fonte: ATBAUTOMATION (2018).

Figura 10 – Aplicação do SCARA em soldagem de componentes.



Fonte: CAIONIX (2018).

Figura 11 – Aplicação do SCARA no processo de produção de cosméticos.



Fonte: TECNOFLUSS (2018).

Figura 12 – Aplicação do SCARA na confecção de pequenos componentes com alta precisão.

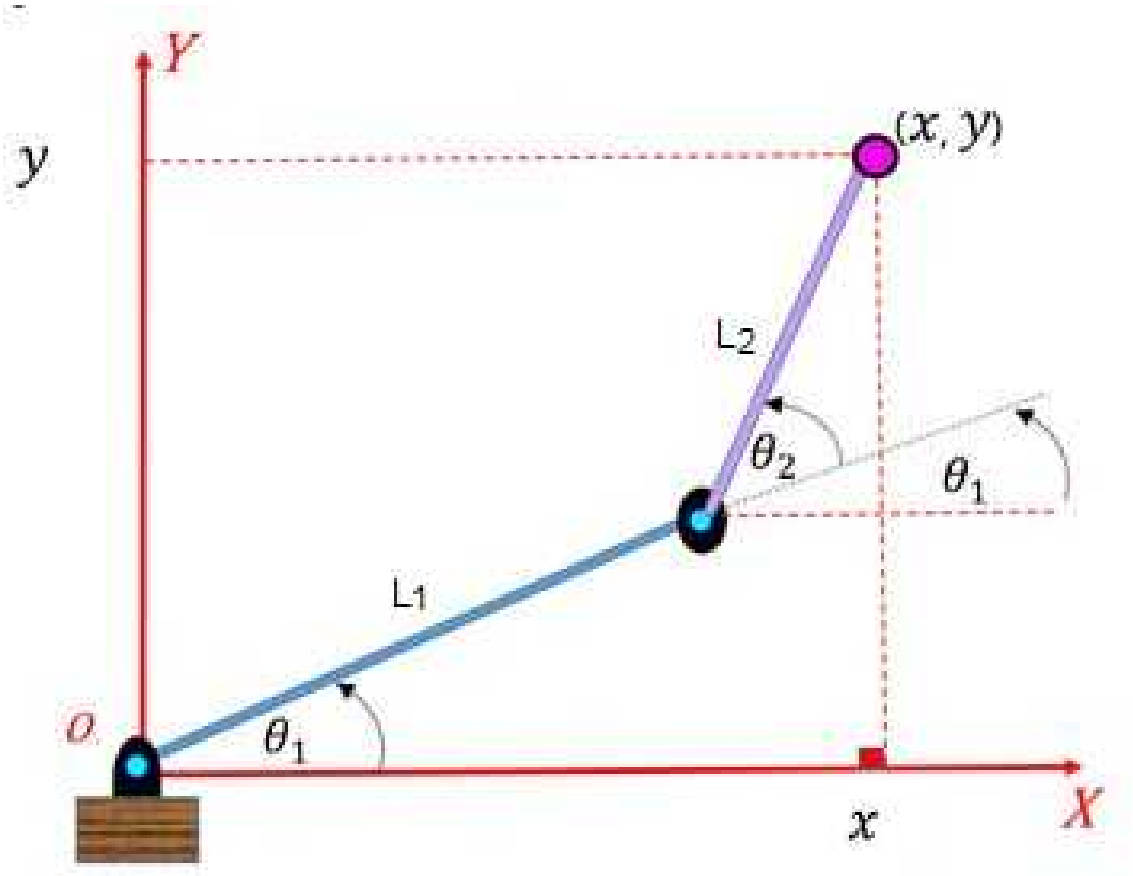
Fonte: AOETECH (2018).

O modelo matemático da posição nos eixos x, y e z do efetuador do SCARA é definido por

$$\begin{cases} x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \\ y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \\ z = Z \end{cases} \tag{1}$$

em que L_1 é o comprimento do primeiro elo robótico, L_2 é o comprimento do segundo elo robótico, θ_1 é o ângulo que o primeiro elo robótico faz com o eixo x e θ_2 é o ângulo que o segundo elo faz com a semirreta oriunda do prolongamento do primeiro elo robótico. É importante ressaltar que (1) pode ser facilmente obtida usando noções de trigonometria básica. Estes parâmetros podem ser identificados na Figura 9, em que é apresentado um esquema simplificado da vista superior de um SCARA. É importante ressaltar que os robôs SCARA que se movem no eixo z possuem um atuador específico que define a posição no eixo z de forma direta, sem necessitar de transformações trigonométricas. A equação (1) é chamada de equação da cinemática direta.

Figura 13 – Esquema simplificado da vista superior de um SCARA.



Fonte: O Autor (2019).

Baseado neste modelo matemático, é possível analisar os vários métodos matemáticos que definem a cinemática do SCARA, detalhados na seção seguinte.

2.3 MÉTODOS MATEMÁTICOS DA CINEMÁTICA DO SCARA

Em qualquer sistema robótico, deseja-se que este realize uma tarefa específica. Para isso, é necessário que os atuadores, localizados nas juntas, movimentem os elos de maneira adequada, seguindo uma programação definida. Sabe-se que o robô só consegue definir a posição angular das juntas, através dos seus atuadores, em outras palavras, os controladores conseguem somente comandar os motores nas juntas (posições angulares θ_1 e θ_2). Diante disso, há duas abordagens distintas que definem os movimentos do SCARA solicitados por um operador (NIKU, 2015) :

1. Dado um sistema robótico, faça o mesmo movimentar de tal forma a ter as juntas nas posições angulares definidas.
2. Faça o efetuador do SCARA se movimentar de sua posição atual (x; y; z) para uma nova posição (x_o; y_o; z_o) com uma orientação (direção para a qual o efetuador aponta) definida.

A primeira abordagem é conhecida como cinemática direta, pois a posição requisitada pelo operador (posição das juntas) corresponde de forma direta ao comando que o controlador enviará aos atuadores. Por exemplo, se o operador definiu que a junta que liga a base ao elo L1 deve sair de 20^o e ir para 30^o, o controlador vai comandar o motor dessa junta para incrementar 10^o. Observe que o comando é calculado diretamente pela posição requisitada pelo operador. Essa primeira abordagem é muito simples e não será dedicada mais explicações sobre a mesma.

A segunda abordagem é conhecida como cinemática inversa, pois a posição e orientação do efetuador requisitadas pelo operador não definem diretamente as posições das juntas, sendo necessário criar um método matemático eficaz que calcule a sequência adequada de posições das juntas que faça com que o efetuador saia de sua posição atual para a nova posição seguindo a orientação requisitada. Este tipo de cinemática é mais complicado de ser realizado visto que solicita cálculos com algoritmos complexos na sua execução. Neste trabalho foram abordados 3 métodos que podem ser utilizados para

realizar o movimento e estes métodos são o numérico, o analítico e o preditivo (BIANCHI, 2016).

O método numérico utiliza diversas iterações para tentar convergir para a solução adequada e este geralmente tem custos computacionais altos. O método analítico (solução fechada) é obtido a partir de manipulações algébricas das equações da cinemática direta em que pode-se encontrar várias soluções que as satisfazem. O método preditivo é baseado na exaustiva análise de todas as possibilidades de movimento e de uma contínua escolha daquele que leva a uma solução mais próxima da solução desejada (x_o,y_o,z_o) (PIMENTA, 2003). Este método chega ao fim quando a solução atual (θ_1,θ_2,Z) resulta na posição mais próxima da desejada (x_o,y_o,z_o).

Os três métodos que realizam a cinemática inversa visam achar a solução para o seguinte sistema de equações:

$$\begin{cases} 0 = L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) - x_o \\ 0 = L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) - y_o \\ 0 = Z - z_o \end{cases} \tag{2}$$

em que (x_o, y_o, z_o) é a posição desejada do efetuador.

Assim, um conjunto de valores (θ_1, θ_2, Z) que satisfaz (2) é uma solução possível de movimento dos atuadores capaz de deslocar o efetuador do SCARA da posição atual para a desejada. Cada método encontrará uma solução distinta para o sistema de equações a partir de uma abordagem própria.

2.3.1 Método Numérico

O primeiro método abordado é o numérico, que é baseado no cálculo iterativo dos argumentos que garantem a solução do sistema de equações em (2). Esse cálculo é orientado pelo gradiente das funções não lineares (SILVA, 2015).

Para encontrar a solução do sistema não linear de equações em (2) foi realizada a busca de uma solução orientada pelos zeros das funções do seguinte sistema de equações:

$$F(\theta_1,\theta_2,Z) = \begin{bmatrix} f_1(\theta_1,\theta_2,Z) \\ f_2(\theta_1,\theta_2,Z) \\ f_3(\theta_1,\theta_2,Z) \end{bmatrix} = \begin{bmatrix} L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) - x_o \\ L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) - y_o \\ Z - z_o \end{bmatrix} \tag{3}$$

Neste sistema, foi aplicado o método numérico de Newton-Raphson para encontrar sua solução. É necessário calcular a Jacobiana do sistema de equações não lineares, dado por:

$$J(\theta_1, \theta_2, Z) = \begin{bmatrix} -L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) & 0 \\ L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

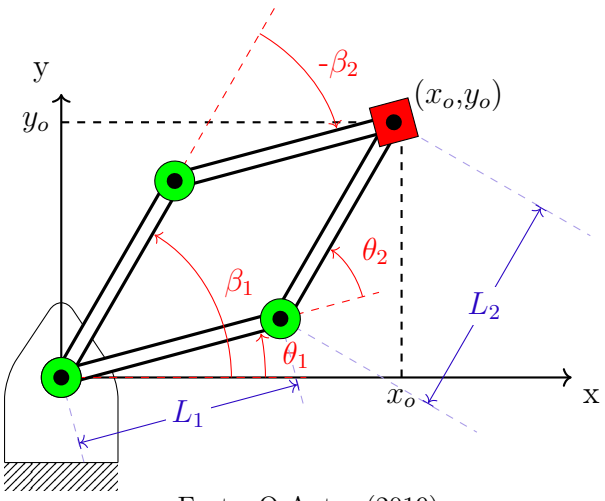
Com a Jacobiana literal bem definida, é necessário definir um ponto inicial no processo iterativo, denotado por ($\theta_{1(0)}$,$\theta_{2(0)}$,$Z_{(0)}$), e calcular a seguinte equação recursiva:

$$\begin{bmatrix} \theta_{1(k+1)} \\ \theta_{2(k+1)} \\ Z_{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_{1(k)} \\ \theta_{2(k)} \\ Z_{(k)} \end{bmatrix} - K.J^{-1}(\theta_{1(k)}, \theta_{2(k)}, Z_{(k)}).F(\theta_{1(k)}, \theta_{2(k)}, Z_{(k)}) \tag{5}$$

em que os índices k e k+1 denotam a k-ésima e k-ésima+1 iteração do método e a constante K é apenas um fator multiplicativo que acelera a convergência do método.

É importante ressaltar que diversas iterações são necessárias para encontrar a solução F(θ_1,θ_2,Z) $\cong \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$. Porém, esta solução não é única , já que, para cada posição desejada do efetuador (x_o, y_o, z_o), existem dois vetores $\begin{bmatrix} \theta_1 & \theta_2 & Z \end{bmatrix}$ de estados dos motores que satisfazem esta condição. Esta afirmação pode ser verificada na Figura 14, em que temos a vista superior de um braço SCARA a uma altura z_o = Z qualquer em que dois conjuntos de ângulos das juntas fazem o efetuador alcançar a mesma posição (x_0,y_0).

Figura 14 – Vista superior do SCARA exemplificando as múltiplas soluções do posicionamento do efetuador.



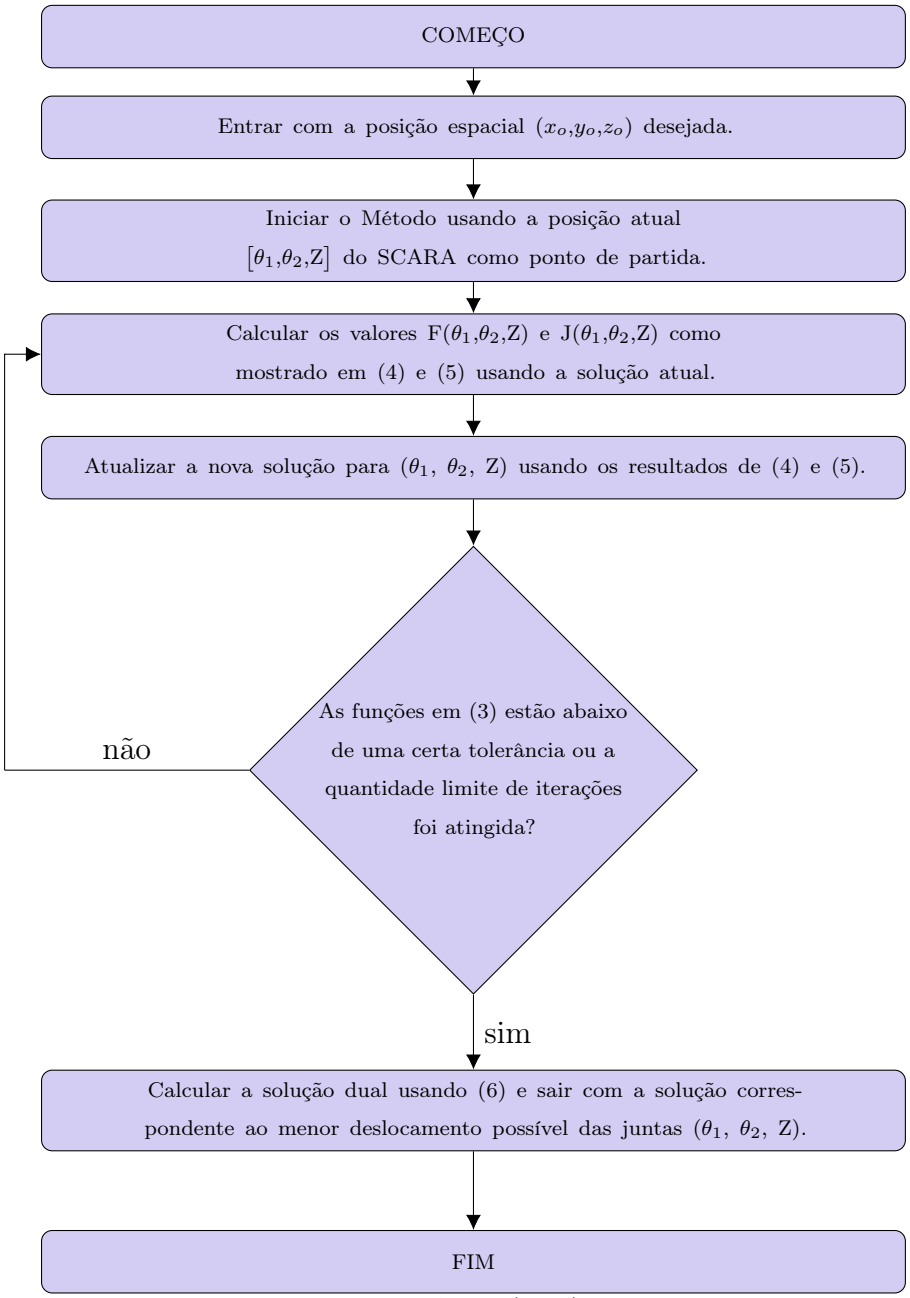
Fonte: O Autor (2019).

Pela Figura 14 é possível verificar que tanto o vetor $[\theta_1,\theta_2,Z]$ e o vetor $[\beta_1,\beta_2,Z]$ satisfazem a posição desejada (x_o,y_o,z_o). Na verdade, existe uma relação entre os ângulos θ_1 e θ_2 com β_1 e β_2, dada por:

$$\begin{cases} \beta_1 = 2\tan^{-1}(\frac{y_o}{x_o}) - \theta_1 \\ \beta_2 = \theta_2 \end{cases} \tag{6}$$

Na Figura 15 é mostrado o fluxograma de procedimentos necessários para a execução desse método numérico.

Figura 15 – Fluxograma dos procedimentos necessários para a execução do método numérico.



Fonte: O Autor (2019).

No apêndice B.1 é possível verificar o programa em .m (linguagem de programação do MATLAB) que resolve este sistema não linear pelo método numérico em poucos milissegundos. Este tempo depende do ponto inicial da busca, da constante K, da posição solicitada e do computador que executa esses cálculos. Caso o ponto inicial tenha uma distância grande do ponto desejado, o sistema demora mais para convergir para a solução ou pode não convergir, devido à existência de duas soluções possíveis. Vale lembrar que este tipo de situação praticamente não ocorre visto que a busca de soluções para a cinemática inversa é realizada usando um ponto inicial relativamente próximo ao ponto desejado e assim o sistema demora poucos milissegundos para obter a resposta desejada. Vale salientar que, após encontrada uma solução, é importante calcular a solução dual e aplicar a solução que provoca o menor deslocamento angular das juntas, reduzindo o esforço no SCARA.

2.3.2 Método Analítico

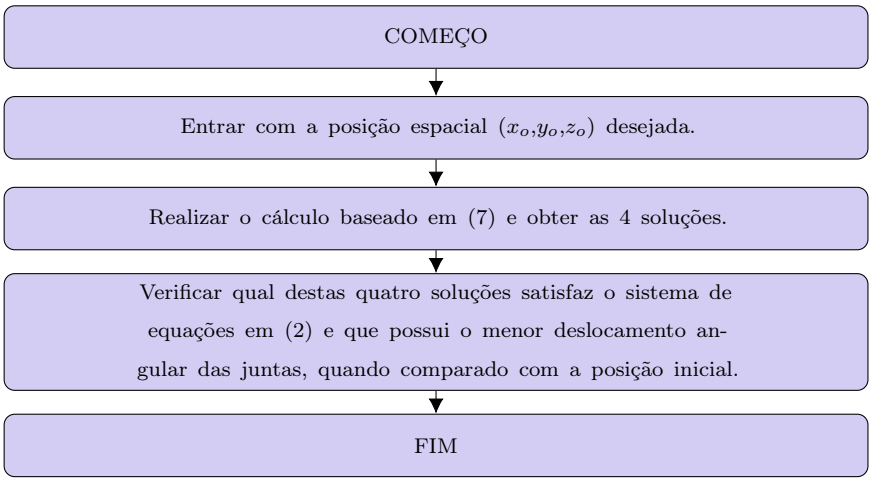
O segundo método avaliado é o analítico, que é baseado em resolver o sistema de equações em (2) algebricamente, considerando que (x_o,y_o,z_o) é a posição desejada do manipulador. Realizando operações aritméticas, que podem ser verificadas no Apêndice A, tem-se que os valores θ_1, θ_2 e Z que satisfazem o sistema de equações em (2) são:

$$\begin{cases} \theta_1 = \cos^{-1}\left(\frac{x_o^2+y_o^2+L_1^2-L_2^2}{2L_1\sqrt{x_o^2+y_o^2}}\right) + \tan^{-1}\left(\frac{y_o}{x_o}\right) \\ \theta_2 = \cos^{-1}\left(\frac{x_o^2+y_o^2-L_1^2-L_2^2}{2L_1L_2}\right) \\ Z = z_o \end{cases} \tag{7}$$

É importante ressaltar que as funções $\cos^{-1}$ resultam em duas possíveis soluções. Assim, temos duas soluções possíveis para θ_1 e para θ_2. Combinando as duas soluções, há 4 possíveis soluções para (7), sendo duas soluções inexequíveis, que devem ser desprezadas, e duas exequíveis, como mostradas na Figura 14. Uma forma adequada de escolher entre as duas exequíveis é selecionar aquela em que os deslocamentos angulares das juntas entre a posição inicial e a posição desejada sejam os menores possíveis.

Na Figura 16, é possível ver o fluxograma de procedimentos realizados para a execução desse método analítico.

Figura 16 – Fluxograma dos procedimentos necessários para executar o método analítico.



Fonte: O Autor (2019).

No Apêndice B.2, é possível verificar o programa em .m criado no MATLAB que resolve (2) pelo método analítico em poucos milissegundos. Este tempo é independente do ponto inicial da busca, da região solicitada e do ponto final.

2.3.3 Método Preditivo

O terceiro método avaliado é o preditivo, que se baseia em estimar a posição entre várias possibilidades de posições das juntas (incrementos, decrementos ou repouso) e escolher aquela posição que mais se aproxima da posição (x_o, y_o, z_o) desejada do efetuador. Este método utiliza a suposição que, em um passo, cada junta só pode incrementar, decrementar ou manter a posição das suas juntas. Assim, para cada uma das possibilidades de deslocamento do sistema, estima-se a nova posição do efetuador e verifica-se em qual destas possibilidades de movimentos a posição do efetuador que melhor se aproxima do ponto final solicitado.

Com relação à junta vinculada ao eixo z, a definição do movimento é bem simples, pois a solução para esta junta é direta (Z = z_o). Assim, a ação de incrementar, decrementar ou manter é definida para fazer z se aproximar de z_o. Entretanto, com relação às juntas associadas a θ_1 e θ_2, essa relação não é tão simples. Assim, por questões de simplicidade, é avaliado o método preditivo somente com as juntas rotativas vinculadas a θ_1 e θ_2. Desta maneira, os possíveis vetores que representam as possíveis combinações das juntas rotativas para um passo de incremento e decremento pré-definido são:

1. Manter e Manter $\Delta\theta$=[0,0]
2. Manter e Aumentar $\Delta\theta$=[0,+]
3. Aumentar e Manter $\Delta\theta$=[+,0]
4. Aumentar e Aumentar $\Delta\theta$=[+,+]
5. Aumentar e Diminuir $\Delta\theta$=[+,-]
6. Diminuir e Aumentar $\Delta\theta$=[-,+]
7. Diminuir e Diminuir $\Delta\theta$=[-,-]
8. Manter e Diminuir $\Delta\theta$=[0,-]
9. Diminuir e Manter $\Delta\theta$=[-,0]

Para cada combinação apresentada, calcula-se as duas primeiras funções definidas em (3). A combinação que apresentar os (x,y) que mais se aproximam de (x_o,y_o) é a solução escolhida para que o algoritmo seja executado novamente. O algoritmo só chega ao fim quando a posição (x,y) da solução escolhida é a mesma que a posição (x,y) da nova solução. É importante salientar que nenhum desses movimentos é executado passo por passo pelo braço robótico, pois o mesmo só executa uma trajetória depois que toda ela é definida. Esse método apresenta grande beneficio, em comparação com os outros métodos, com relação a simplicidade do método e por demandar menos conhecimento matemático e estrutural do braço robótico.

Na Figura 17 é possível verificar um diagrama que representa o fluxo de procedimentos realizados para a execução desse método preditivo.

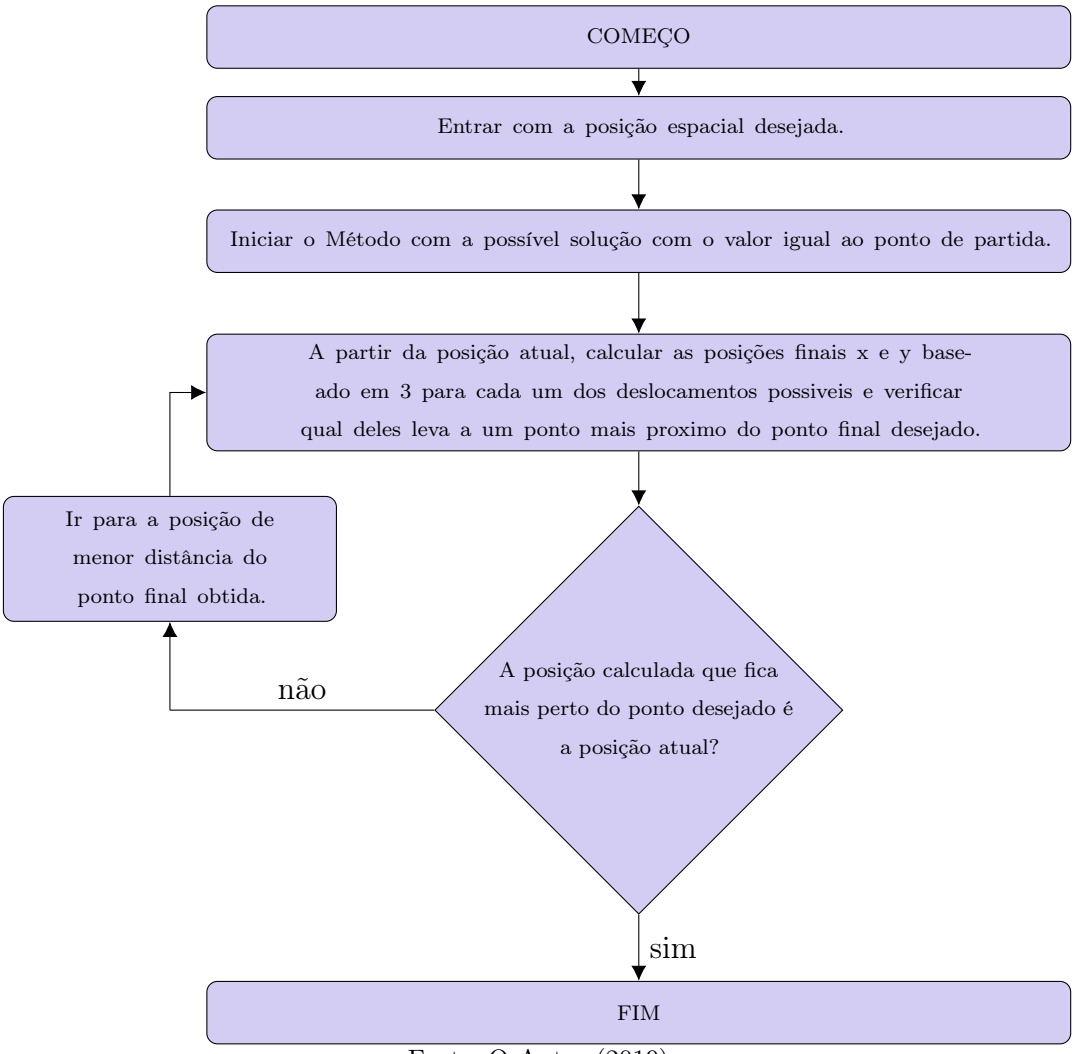
No Apêndice B.3 é possível verificar o programa em .m que aplica este algoritmo para desenvolver uma trajetória entre dois pontos e uma solução para (4).

2.4 ANÁLISE ESTRUTURAL DO ROBÔ SCARA

O método abordado para a análise estrutural do robô SCARA é realizado por um software de simulação baseado em elementos finitos que seja capaz de estimar forças de contato, deformação, torções que uma estrutura pode sofrer perante o acionamento de motores, o movimento de cargas pelos efetuadores, a sustentação da estrutura, etc.

Essa análise propõe que o número infinito de variáveis desconhecidas seja substituído por um número limitado de elementos de comportamento bem definido. Essas divisões podem apresentar diferentes formas, tais como a triangular, quadrilateral, entre outras,

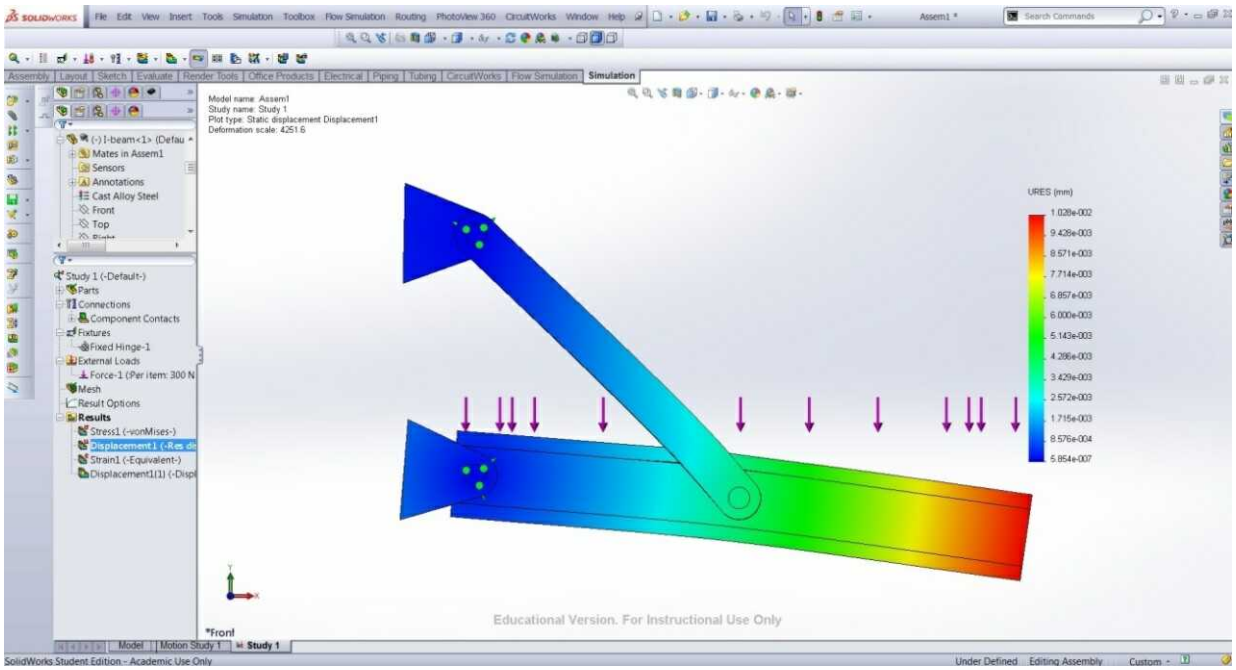
Figura 17 – Fluxograma dos procedimentos realizados pelo método preditivo.



Fonte: O Autor (2019).

em função do tipo e da dimensão do problema. Como são elementos de dimensões finitas, são chamados de "elementos finitos" – termo que nomeia o método. Este tipo de análise pode ser realizado por diversos softwares como SolidWorks, Ansys e Abaqus. Os conceitos e técnicas para simulação deste ambiente podem ser realizados de tal forma a ter uma grande previsão dos fenômenos mecânicos que a estrutura pode sofrer e são altamente recomentados no desenvolvimento de qualquer estrutura mecânica. Na Figura 18 é possível verificar um exemplo de uma simulações de deformação realizada no ambiente SolidWorks (SOLIDWORKS, 2018).

Figura 18 – Análise de deformações em uma peça, realizada no ambiente SolidWorks.



Fonte: O Autor (2019).

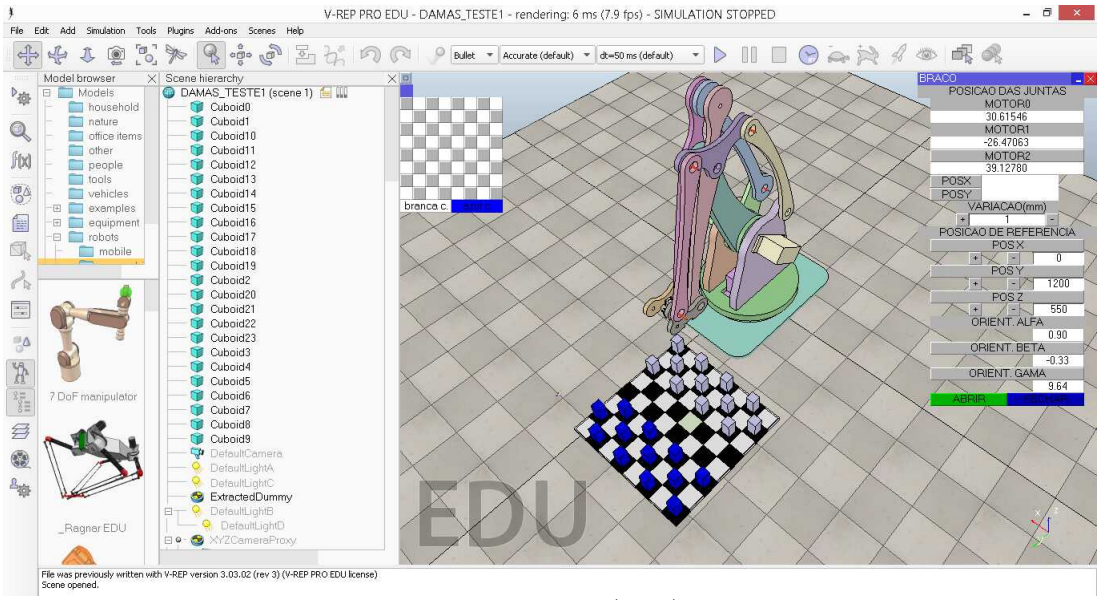
2.5 AMBIENTES DE SIMULAÇÃO VIRTUAL DO ROBÔ SCARA

Para testar e validar os métodos de cinemática direta e inversa antes da execução experimental do robô, diversas simulações foram feitas no ambiente 3D de simulação de robôs chamado V-REP. Este software foi escolhido por ser gratuito, aplicado na área educacional e ser um dos melhores em simular robôs com precisão. O simulador de robôs V-REP consiste em um ambiente de desenvolvimento integrado, baseado em um controle distribuído: cada objeto criado pode ser controlado individualmente por meio de um programa incorporado (V-REP, 2018). A interface gráfica do V-REP pode ser vista na Figura 19.

2.6 CONCLUSÃO

A análise da estrutura básica de um braço robótico é primordial para entender cada uma das suas partes e a funcionalidade de cada uma delas. Por consequência, entender o que é um SCARA, suas principais características, aplicações e seu modelo matemático é de fundamental importância para desenvolver qualquer temática no assunto.

Figura 19 – Interface gráfica do ambiente V-REP.



Fonte: O Autor (2019).

Com relação aos métodos matemáticos apresentados, da cinemática do robô SCARA, são necessários para definição das trajetórias simples e complexas que o efetuador realizará, como mostrado nos próximos capítulos.

Com relação à análise estrutural e ao ambiente virtual, é muito importante que seus conceitos e objetivos estejam muito claros para que, quando estes sejam citados nos próximos capítulos, não exista dúvida da importância dos mesmos no processo de desenvolvimento do trabalho.

Assim, tais conceitos serão utilizados nos desenvolvimentos dos próximos capítulos do trabalho.

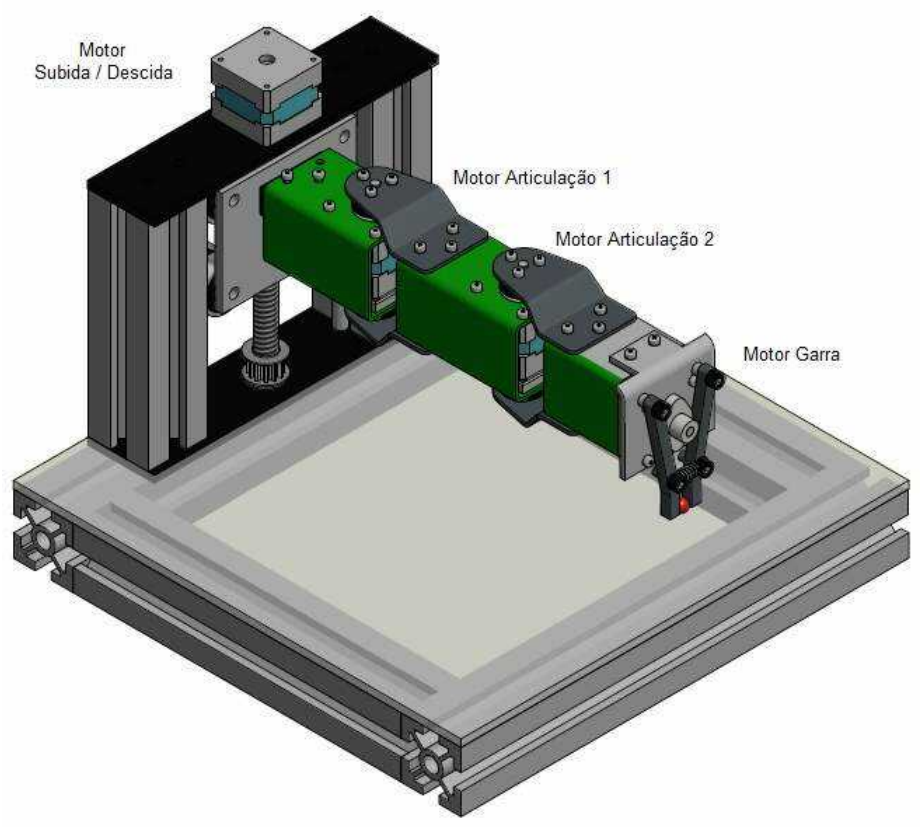
3 DEFINIÇÃO DA ESTRUTURA E SIMULAÇÕES EM AMBIENTES VIRTUAIS

Neste capítulo são apresentados os desenhos 3D da estrutura do robô SCARA, todos desenvolvidos no ambiente SolidWorks, os procedimentos e os resultados de simulações relacionados à estrutura mecânica, além de simulações relacionadas ao comportamento cinemático do SCARA em ambiente virtual.

3.1 DESENHO 3D E ANÁLISES MECÂNICAS NO SOLIDWORKS

O projeto mecânico da estrutura foi totalmente baseado no braço robótico comercial vendido pela TEKKNO (TEKKNO, 2018). Basicamente, o braço comercial é o conjunto de uma base de alumínio com um bloco móvel de eixo Z (semelhante aos usados em CNCs) acoplado a um conjunto de braços móveis feitos de aço galvanizado que utilizam um motor de passo do padrão nema 17 de 5kgf.cm e de 3kgf.cm para realizar o posicionamento dos elos móveis (WOTIOM, 2019). O projeto do braço robótico SCARA da TEKKNO pode ser visto na Figura 20.

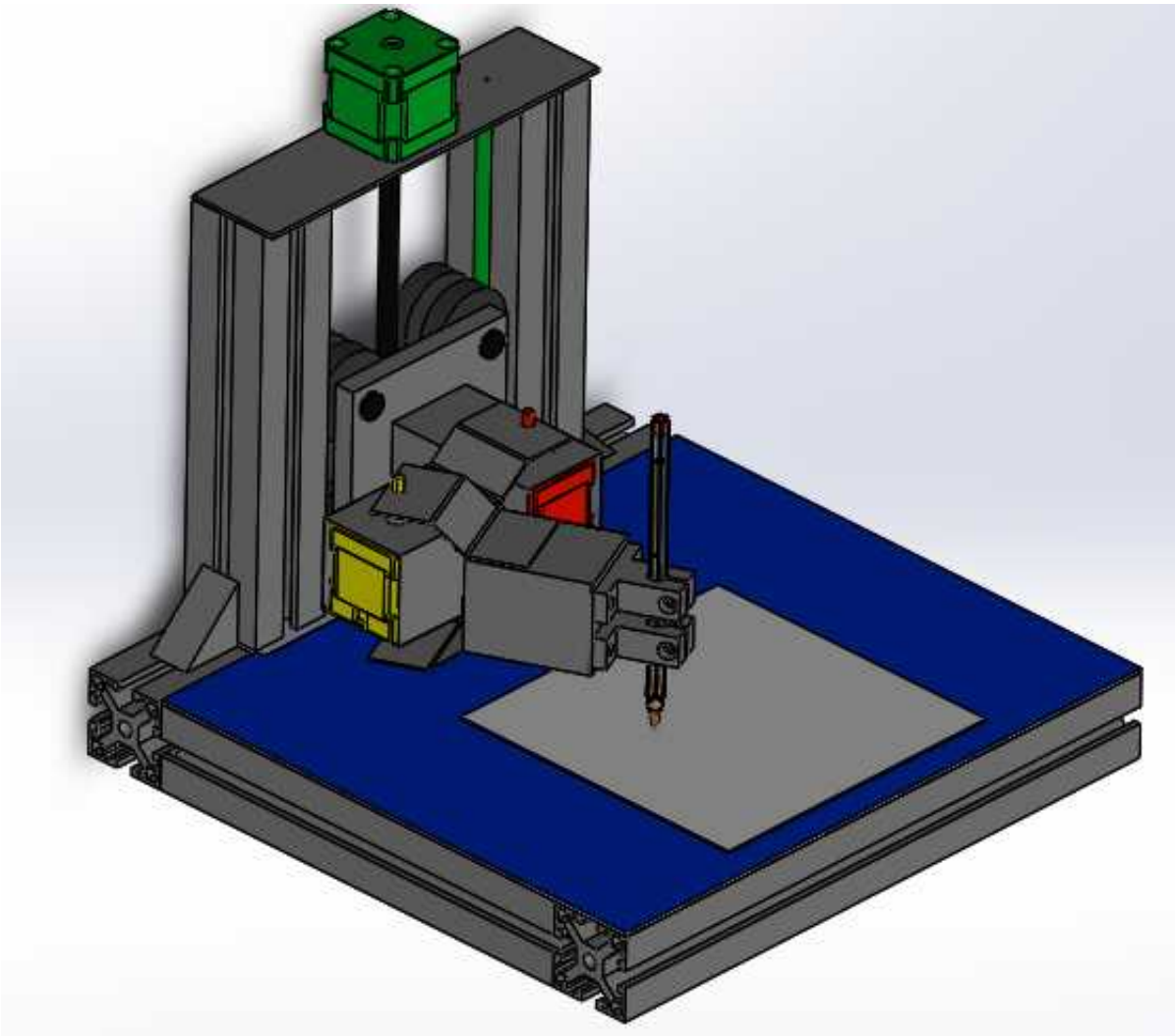
Figura 20 – Braço robótico oferecido pela empresa TEKKNO.



Fonte: TEKKNO (2018).

A partir da estrutura oferecida pela TEKKNO e de dados de projeto disponibilizados pela empresa, criou-se um projeto de sua estrutura mecânica no SolidWorks, em que se procurou ser o mais fiel possível ao projeto. As diferenças encontradas devem-se à modificação do efetuador, já que este deveria ser capaz de realizar uma tarefa que pudesse verificar a precisão de sua cinemática. Assim, definiu-se que a funcionalidade do robô seria desenhar ou traçar trajetórias em um papel. O esboço deste projeto realizado no SolidWorks pode ser verificado na Figura 21. Um desenho simplificado mostrando a visão superior do braço da Figura 21 pode ser visto na Figura 22. É importante ressaltar que os valores das dimensões que definem os elos do braço desenvolvido são $L_1 = 86$mm e $L_2 = 105$mm.

Figura 21 – Esboço do projeto do braço robótico desenvolvido no SolidWorks.

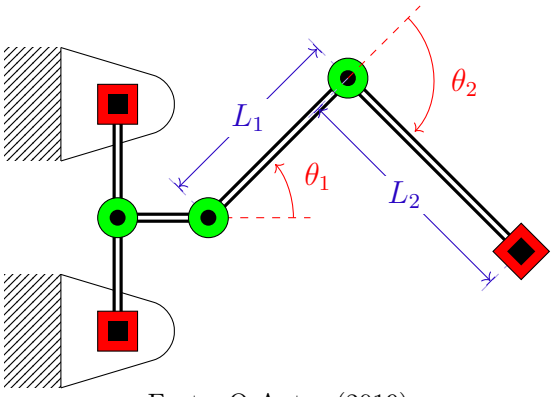


Fonte: O Autor (2019).

A partir desta estrutura, desenvolvida no SolidWorks, gerou-se um conjunto de peças 3D no formato .STL. Estes arquivos 3D são compatíveis com uma série de programas,

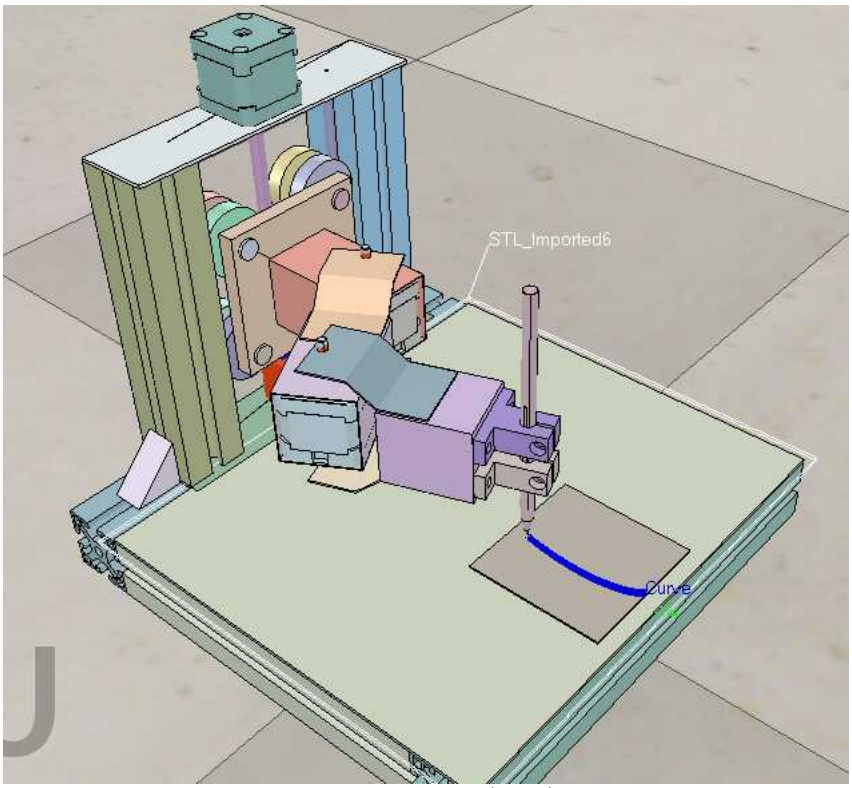
sendo um deles o ambiente de simulação 3D V-REP. Uma ilustração que mostra o resultado da importação dos arquivos .STL no ambiente V-REP pode ser vista na Figura 23.

Figura 22 – Desenho simplificado da visão superior do braço robótico SCARA desenvolvido.



Fonte: O Autor (2019).

Figura 23 – Ilustração do SCARA importado para o ambiente de simulação virtual V-REP.

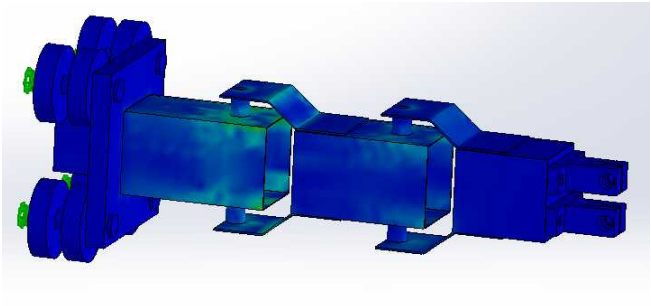


Fonte: O Autor (2019).

Para verificar o estresse mecânico das peças foi utilizado o ambiente de simulação SolidWorks. Neste ambiente, foram realizadas simulações referentes a esforços e deformações na estrutura devido ao peso do sistema e ao torque dos motores. O objetivo da simulação de esforços mostrada na Figura 24 é verificar, de forma qualitativa, regiões que sofrem um maior estresse mecânico e assim verificar quais regiões devem ter maiores cuidados durante

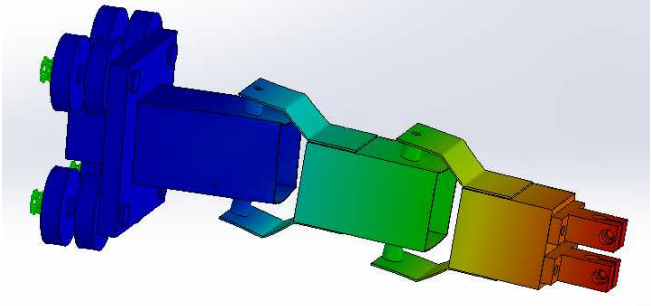
uma manutenção. Na Figura 24 são representados os estresses de menor intensidade com cores mais frias (azul escuro) e estresses de maior intensidade com cores mais quentes (vermelho). Deve-se atentar que existe uma maior quantidade de esforços nas regiões que se conectam com os eixos dos motores, mas estes esforços são muito pequenos para serem considerados problemas visto que o material utilizado para fazer este braço robótico é superdimensionado para os esforços que o mesmo pode sofrer. Com relação à simulação de deformações na Figura 25, o objetivo é conseguir verificar regiões que podem oferecer uma maior fonte de imprecisão devido a estas deformações e assim poder tomar medidas para minimizar seus efeitos. Na Figura 25 são representados as deformações de menores intensidades com cores mais frias (azul escuro) as deformações de maiores intensidades com cores mais quentes (vermelho). É possível verificar que a deformação na extremidade do braço robótico é a maior intensidade, mas este deslocamento está longe de ser algo preocupante pois a ordem de grandeza deste deslocamento é de centésimos de milímetro. Esta pequena deformação se deve ao fato de utilizar um material super dimensionado para construção do braço robótico.

Figura 24 – Simulação qualitativa dos esforços provocados na estrutura do braço robótico.



Fonte: O Autor (2019).

Figura 25 – Simulação qualitativa das deformações provocadas na estrutura do braço robótico.



Fonte: O Autor (2019).

3.2 SIMULAÇÕES DE POSICIONAMENTO POR CINEMÁTICA DIRETA

Para realizar as simulações de posicionamento por cinemática direta foi adotada a metodologia de solicitar aos atuadores que assumam as posições angulares desejadas e, assim, verificar o desempenho do braço robótico virtual no processo de deslocamento. Nesta etapa, é analisada a trajetória do efetuador para diferentes métodos e estes serão comparados. O primeiro método assume um controle de posição independente das juntas, ou seja, solicita que cada junta chegue à posição de referência no menor tempo possível. O segundo método assume um controle síncrono, de forma que todas as juntas devem terminar o movimento ao mesmo tempo. Assim, ao final das simulações, é verificado qual método é o mais adequado para realizar posicionamentos com cinemática direta respeitando uma velocidade angular máxima. Na Figura 26 é possível verificar um fluxograma que representa a lógica do método de controle independente por junta e na Figura 27 o fluxograma do método de controle síncrono.

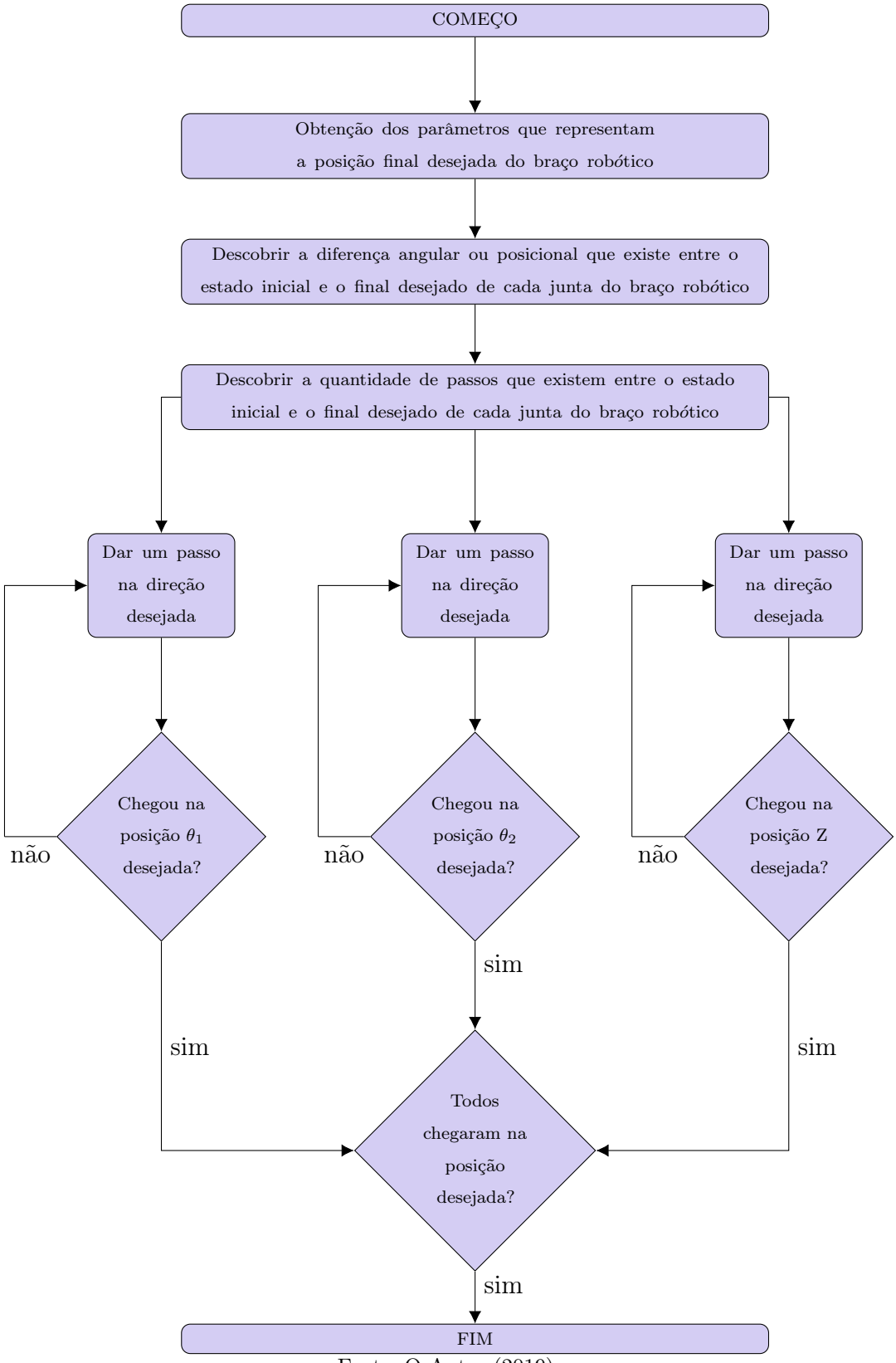
Durante estas simulações é verificada a trajetória realizada pelo efetuador e a velocidade que os dois métodos impõem ao efetuador. É importante ressaltar que estas simulações são realizadas de forma a obter dados comparativos de forma qualitativa, já que não se tem definido, a priori, a velocidade angular máxima proporcionada pelos atuadores. Para isso, é assumido que o atuador seja um motor de passo com acionamento de micro-passo cujo tempo que o sistema demora para dar um passo é de 1ms e que o conjunto consegue executar 25.600 passos por revolução (1/200 do motor e 1/128 do micro-passo), ou seja, uma resolução angular de $0,0140625^o$ por passo. Assim, com essas condições, é possível verificar qual dos métodos têm a melhor performance. Na Tabela 1, é possível verificar as posições solicitadas em duas simulações distintas. Nas Figuras 28 e 29 é possível ver os resultados do posicionamento e velocidade dos dois métodos em cada uma das simulações da Tabela 1. Deve ser observado que independente do método, a estrutura chega em sua posição requisitada utilizando o mesmo espaço de tempo. Isto ocorre porque quem determina o tempo de deslocamento em ambas as simulações é a junta que precisa realizar o maior deslocamento.

Tabela 1 – Posições inicial e final das duas simulações por cinemática direta.

Simulação N^o	Posição Inicial (x_o;y_o;Z)	Posição Requisitada (x_o;y_o;Z)
1	(0^o;0^o;1mm)	(40^o;10^o;1,5mm)
2	(0^o;0^o;1mm)	(25^o;90^o;2,5mm)

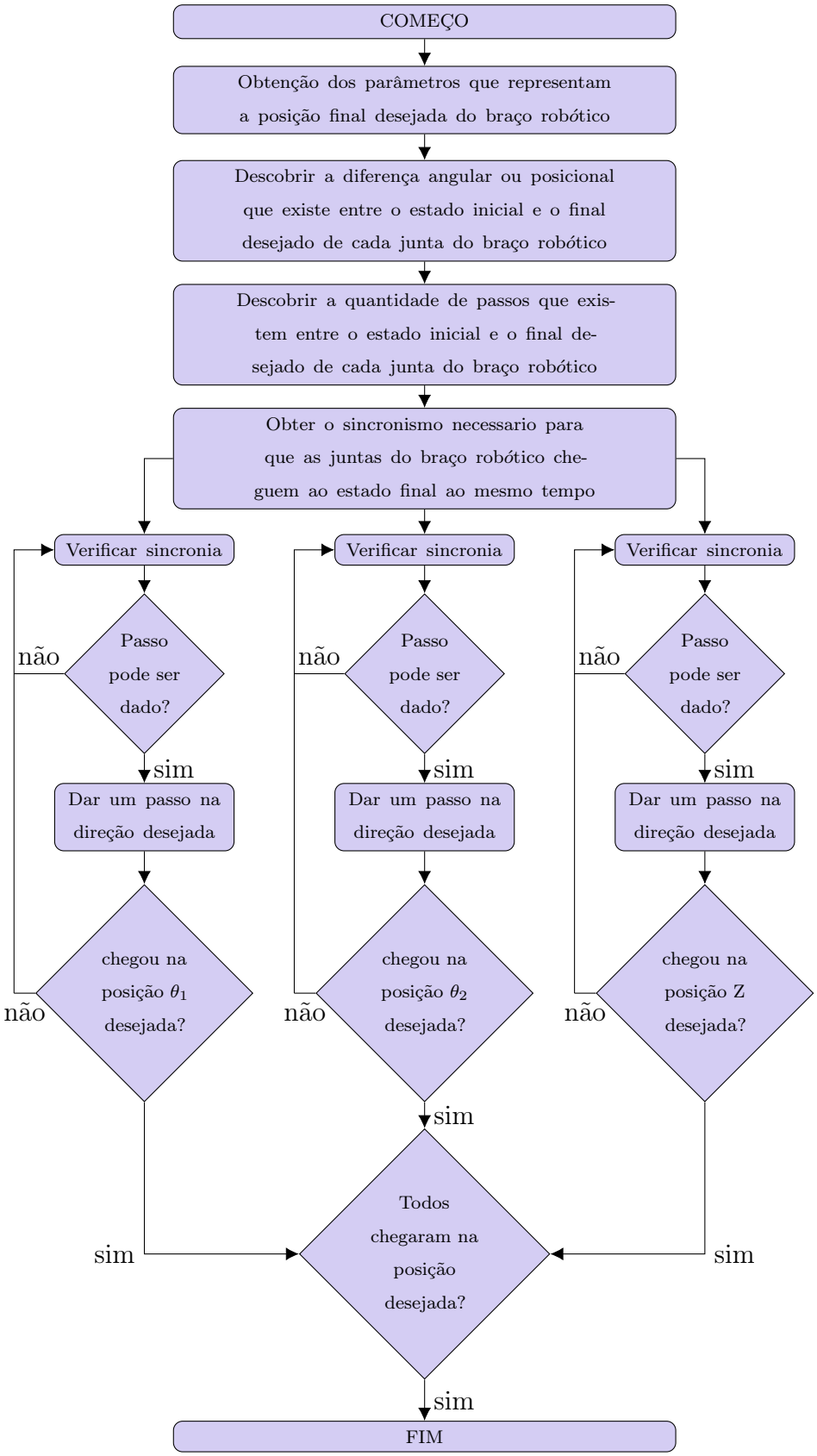
Fonte: O Autor (2019).

Figura 26 – Fluxograma do método de controle independente por junta.



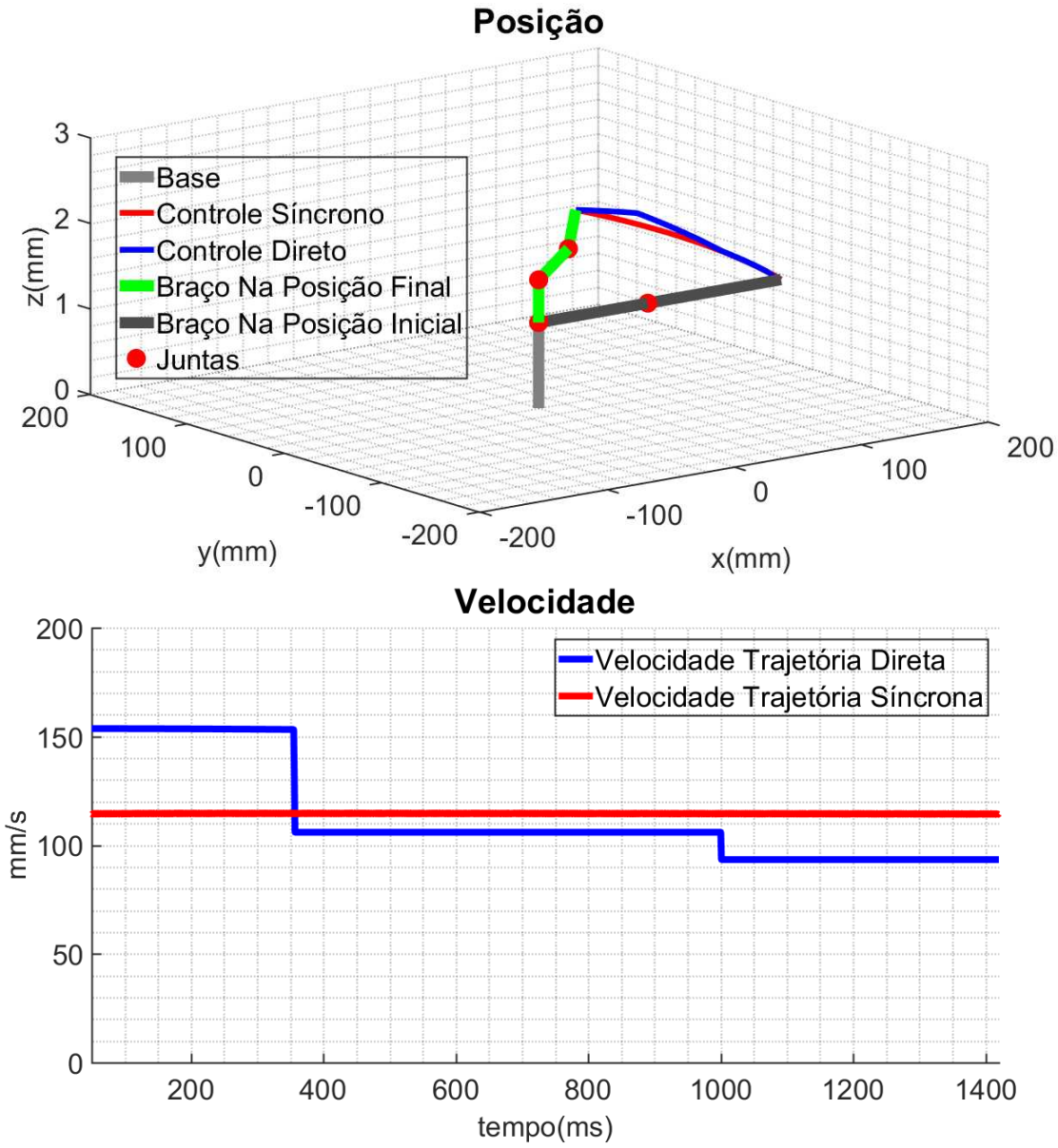
Fonte: O Autor (2019).

Figura 27 – Fluxograma do método de controle síncrono das juntas.



Fonte: O Autor (2019).

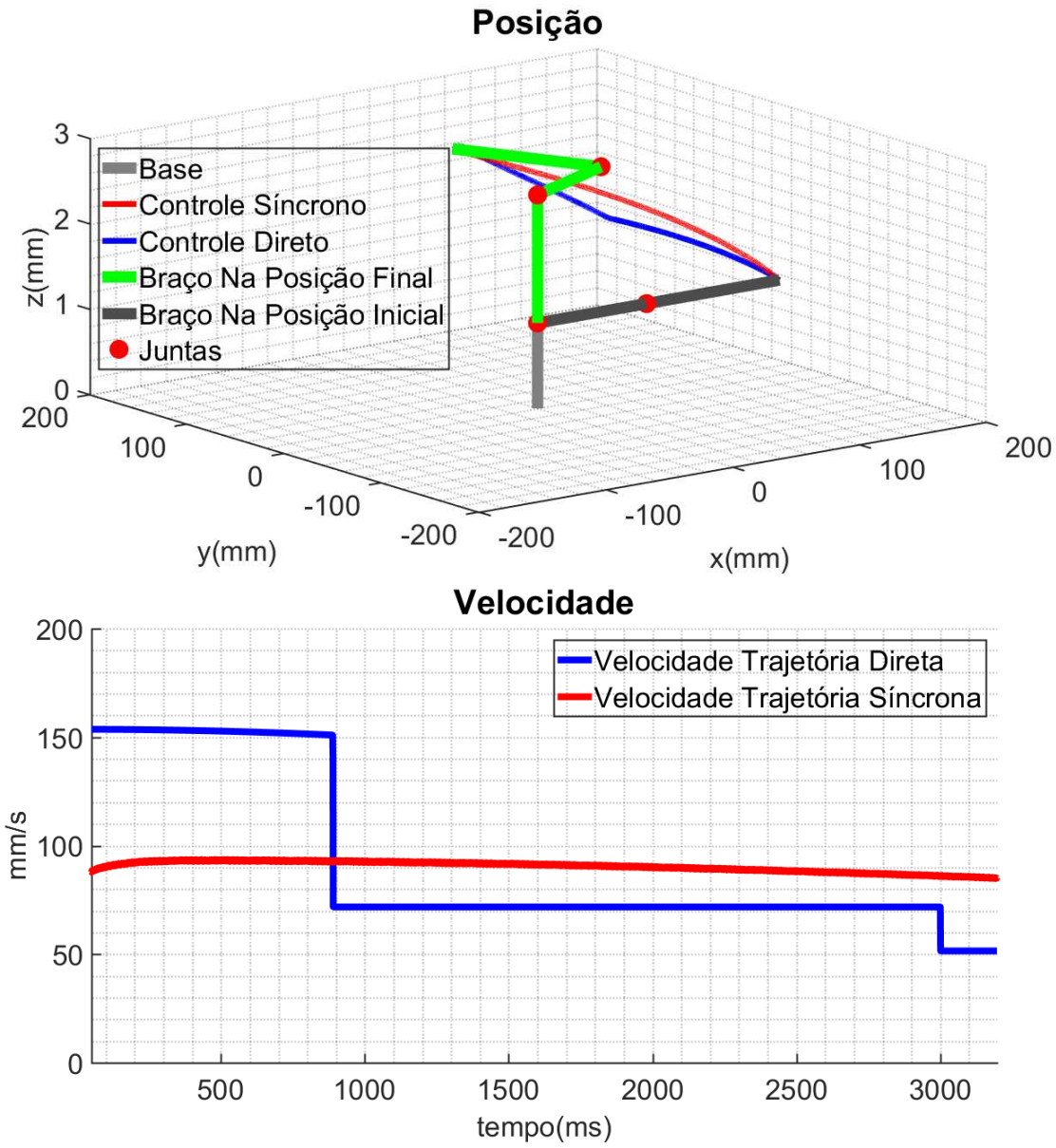
Figura 28 – Simulação do posicionamento do efetuador para o deslocamento solicitado na Simulação N^o 1.



Fonte: O Autor (2019).

Analisando os resultados das simulações das Figuras 28 e 29, é possível verificar que, em ambos os casos, o espaço deslocado do efetuador no método de controle independente é maior que o método de controle síncrono. É possível perceber também que o método independente tem uma grande variação na velocidade absoluta do efetuador, pois, no início do percurso, todos os motores das juntas estão operando e, no decorrer da trajetória, alguns destes motores param de ser acionados visto que a posição angular desta junta já foi atendida. Assim, verifica-se que o método síncrono é aquele que proporciona menores variações de velocidade, menores percursos deslocados e uma menor velocidade média.

Figura 29 – Simulação do posicionamento do efetuador para o deslocamento solicitado na Simulação Nº 2.

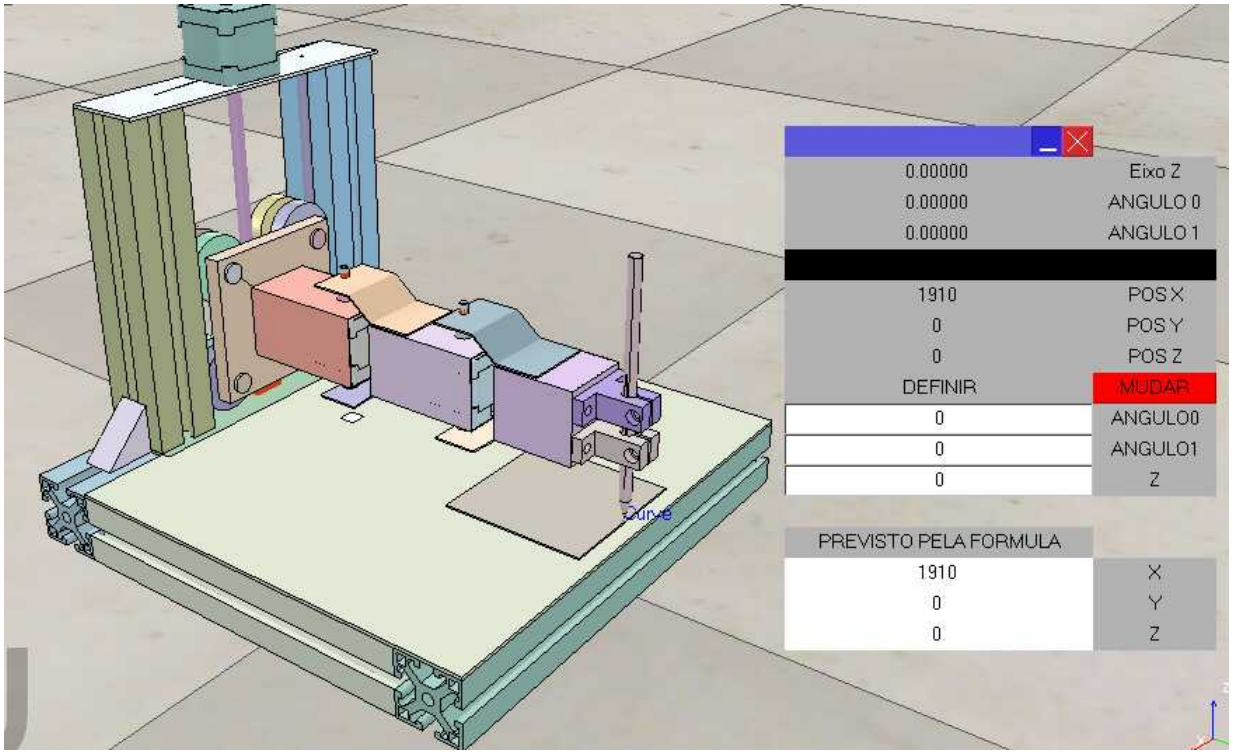


Fonte: O Autor (2019).

Assim, o método síncrono se mostra o mais adequado para executar trajetórias baseadas em cinemática direta.

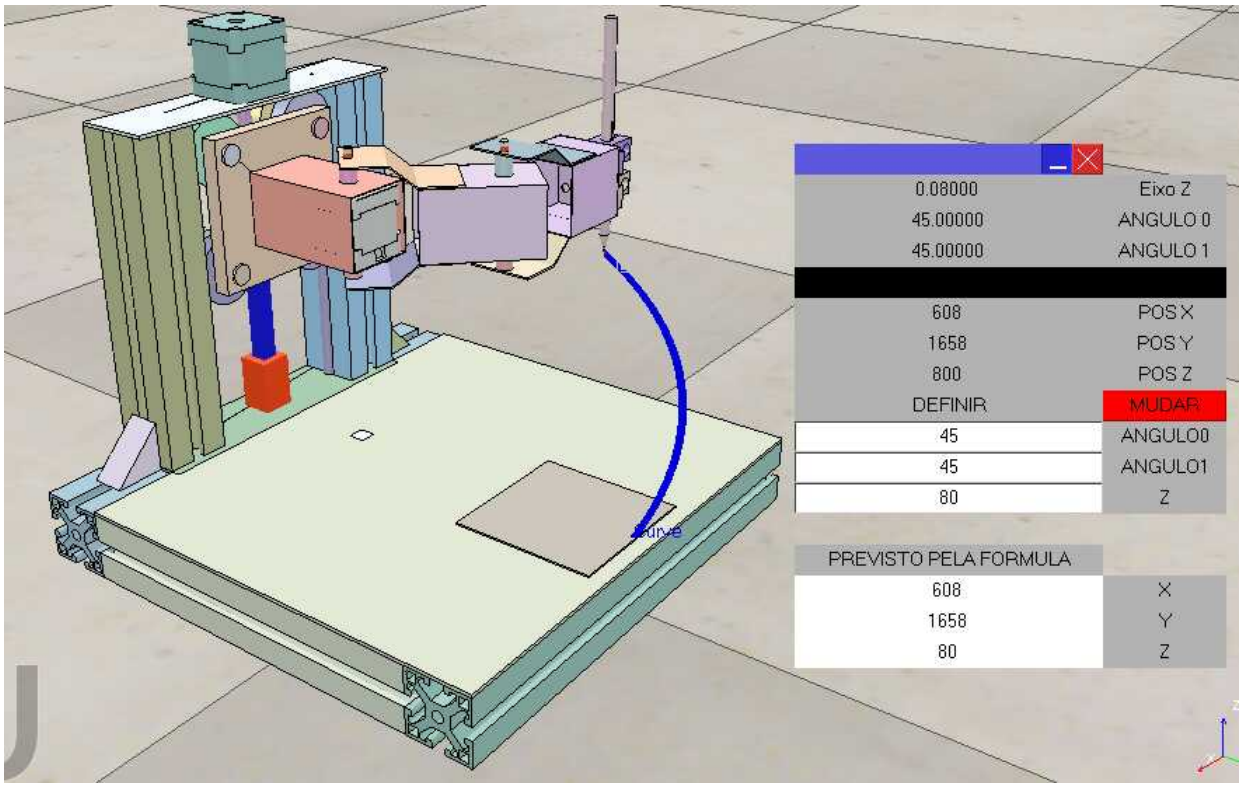
Para verificar a movimentação do efetuador durante o deslocamento, realizou-se uma simulação no ambiente V-REP em que o método do controle síncrono foi aplicado. Esta simulação pode ser verificada nas Figuras 30 e 31, em que a primeira mostra a posição inicial do movimento e a segunda a posição final do movimento, com trajetória em azul.

Figura 30 – Posição inicial da estrutura robótica durante a simulação do método síncrono no ambiente virtual de simulação V-REP.



Fonte: O Autor (2019).

Figura 31 – Posição final da estrutura robótica durante a simulação do método síncrono no ambiente virtual de simulação V-REP.



Fonte: O Autor (2019).

3.3 SIMULAÇÕES DE POSICIONAMENTO PELOS MÉTODOS DE CINEMÁTICA INVERSA

Nesta seção é verificada a capacidade de se obter o conjunto de posições angulares das juntas que garantam que o efetuador vá para uma dada posição desejada. A metodologia escolhida foi aplicar os três métodos matemáticos descritos na Seção 2.3 e verificar o desempenho dos mesmos em obter a posição angular que garante que o efetuador chegue à posição desejada. Nesta simulação não é avaliada a trajetória entre a posição inicial e a final nem tampouco a variação de velocidade do efetuador. Ao final das simulações, é esperado que seja definido qual método é o melhor para realizar a cinemática inversa em termos de precisão e tempo de execução. Nas Tabelas 2-4, é possível verificar, para cada método, a posição desejada do efetuador, as posições dos motores obtidas, o tempo de cálculo e a posição final alcançada pelo efetuador. O eixo z foi omitido por questões de simplicidade, pois este eixo é diretamente acionado da sua posição inicial para a final. Vale

lembrar que as comparações foram realizadas nas mesmas condições da cinemática direta, ou seja, 0,0140625^o por passo.

É sempre importante lembrar que as diferenças entre as posições desejadas e alcançadas são oriundas da resolução do braço robótico e não, necessariamente, do método matemático. Estas diferenças ocorrem porque não necessariamente a posição desejada é um ponto possível a ser alcançado pelas juntas. A posição inicial adotada nas três simulações e em todos os métodos é a posição em que θ_1=-45^o e θ_2=45^o, que define a posição do efetuador em aproximadamente x=165.81mm e y= -60.81mm.

Tabela 2 – Resultados das simulações com o Método Numérico.

Simulação N^o	1	2	3
Posição Desejada (xo ;yo)(mm)	(190;1)	(145;-85)	(166;-62)
Numérico			
Solução (θ_1;θ_2)	($-6,1664^o$;$11,7773^o$)	($-61,9875^o$;$57,0305^o$)	($-44,8242^o$;$44,0930^o$)
Tempo de Cálculo (ms)	0,4	0,2	0,1
Posição Alcançada (x_f;y_f) (mm)	(189,9993;1,0283)	(144,9984;-84,9976)	(165,9889;-61,9644)
Erro (mm)	0,0283	0,0028	0,0372
Analítico			
Solução (θ_1;θ_2)	($-6,1664^o$;$11,7773^o$)	($-61,9875^o$;$57,0305^o$)	($-44,8172^o$;$44,0578^o$)
Tempo de Cálculo (ms)	0,1	0,1	0,1
Posição Alcançada (x_f;y_f) (mm)	(189,9993;1,0283)	(144,9984;-84,9976)	(165,9957;-62,0084)
Erro (mm)	0,0283	0,0028	0,0094
Preditivo			
Solução (θ_1;θ_2)	($-6,3141^o$;$12,0305^o$)	($-61,9805^o$;$57,0164^o$)	($-44,8172^o$;$44,0648^o$)
Tempo de Cálculo (ms)	393,5	145,9	7,2
Posição Alcançada (x_f;y_f) (mm)	(189,9562;1,0004)	(145,0066;-85,0055)	(165,9959;-61,9855)
Erro (mm)	0,0438	0,0086	0,0157

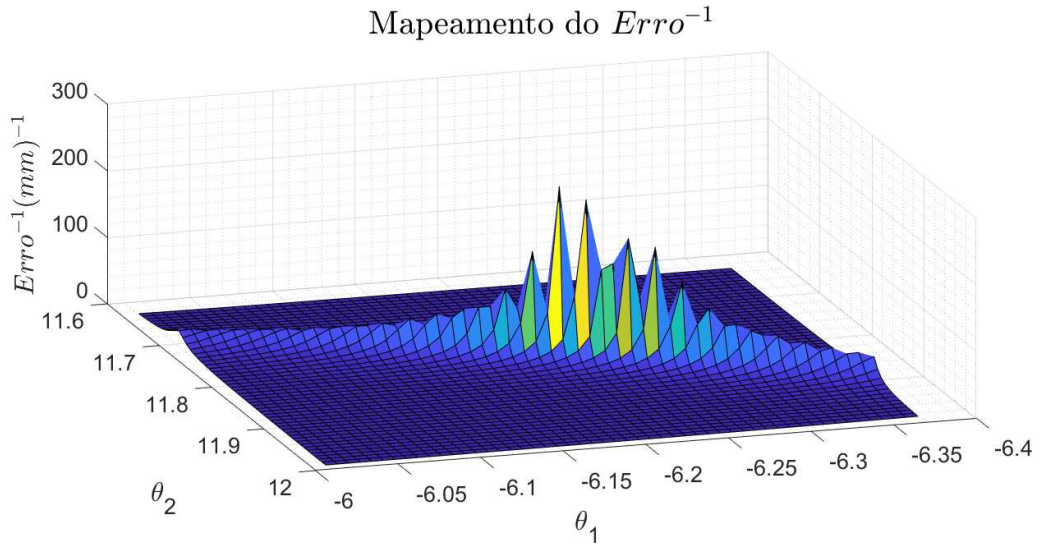
Fonte: O Autor (2019).

Analisando os resultados temporais apresentados na Tabela 2, percebe-se que:

- O método analítico é aquele que tem o seu tempo mais consistente e de menor valor. O fato da consistência é verificado pela característica deste método ter uma quantidade de cálculos que independente da posição inicial e final do efetuador;
- O método numérico tem uma dependência da distância entre o ponto inicial e o final visto que é baseado em uma convergência numérica em que o ponto inicial da busca é a distância entre o ponto inicial e o desejado, que impacta na quantidade de iterações do método, até sua convergência;
- O método preditivo não foi feito para calcular apenas o ponto final da trajetória, já que este calcula todo o conjunto de pontos que leva o ponto inicial ao ponto final no passo especificado. Assim, o tempo de cálculo deste método é extremante dependente da distância entre o ponto final e o inicial.

Com relação às analises das posições finais do efetuador, percebe-se que os métodos obtiveram respostas próximas o suficiente do ponto desejado. É importante ressaltar que a razão dos diferentes valores angulares se deve ao fato que o método numérico e o método preditivo estão sujeitos a ficarem presos em mínimos locais. Na Figura 32 é possível verificar a função inversa do erro (distância euclidiana entre a posição desejada e a posição alcançada do efetuador) para pontos próximos à posição desejada da simulação N^o 1. O maior pico representa o mínimo global e os outros picos representam mínimos locais.

Figura 32 – Análise da função inversa do erro na simulação N^o 1 para pontos próximos da posição desejada.



Fonte: O Autor (2019).

Assim, o melhor método de cinemática inversa, em termos de tempo e precisão na posição final, é o analítico. Porém, um movimento não é apenas definido pela posição das juntas no ponto desejado do efetuador. Para definir de forma completa um movimento, é importante estudar a trajetória que o efetuador irá realizar até chegar no ponto definido. Na próxima seção, estes detalhes são explicados.

3.4 SIMULAÇÕES DE TRAJETÓRIAS SIMPLES GUIADAS PELOS MÉTODOS DE CINEMÁTICA INVERSA

Nesta seção são avaliadas as possíveis trajetórias obtidas entre dois pontos quando são aplicados diferentes métodos de geração de trajetória. Uma possível solução para realizar esta tarefa é usar o controle independente ou síncrono para realizar o movimento das juntas, ambos definidos na Seção 3.2, juntamente com o método analítico ou numérico

de solução da cinemática inversa, ambos mostrados na Seção 3.3. Uma solução alternativa é usar apenas o método preditivo mostrado na Seção 3.3, já que este método define uma possível trajetória para o movimento do efetuador.

Nesta seção, dois métodos são comparados: 1) método analítico para solução da cinemática inversa em conjunto com o controle síncrono das juntas do braço robótico; 2) método preditivo, capaz de solucionar a cinemática inversa e definir o controle das juntas simultaneamente. Ambos métodos serão comparados pela sua trajetória, quantidade de passos e a distância do ponto atual do efetuador ao ponto final em cada passo (Erro). Na Tabela 3 é possível observar a posição inicial, a posição requisitada e o número de passos que cada um dos métodos realizou para chegar à posição desejada. As simulações descritas na Tabela 3 são mostrados nas Figuras 33-35, onde pode-se ver a trajetória espacial, trajetória angular e o gráfico do erro em cada caso. A posição inicial adotada nas três simulações e em todos os métodos é a posição em que θ_1=-45^o e θ_2=45^o, que define a posição do efetuador em aproximadamente x=165,81mm e y= -60,81mm.

Tabela 3 – Comparação do método preditivo e do método analítico com controle síncrono das juntas.

Simulação N^o	Posição Inicial (x_o,y_o)	Posição Desejada (x_o,y_o)	Passos Preditivos	Passos Analíticos
1	(165,81 mm;-60,81 mm)	(173,15 mm;-7,08 mm)	2408	2134
2	(165,81 mm;-60,81 mm)	(105,00 mm;-86,00 mm)	8852	6400
3	(165,81 mm;-60,81 mm)	(141,27 mm;121,78 mm)	12460	10240

Fonte: O Autor (2019).

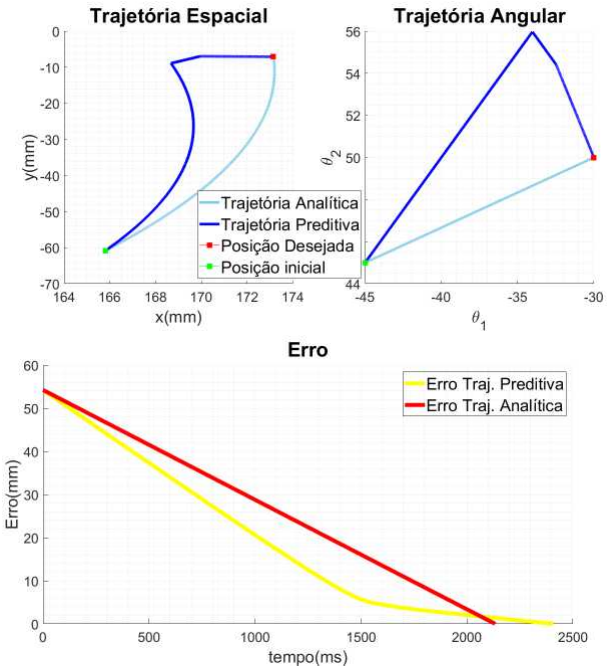
Com relação ao número de passos, vistos na Tabela 3, pode-se dizer que:

- O método analítico realiza a trajetória sempre com um menor número de passos quando comparado com o método preditivo;
- As diferenças destes métodos são devida à imprevisibilidade do método preditivo, pois apesar deste minimizar o erro em cada passo este não minimiza o erro da trajetória completa, pois o horizonte de análise é sempre de um passo por vez (otimização local e não global).

Assim, fica claro que o método analítico de solução da cinemática inversa em conjunto com o controle síncrono das juntas é a melhor solução quando se deseja o menor tempo para o efetuador chegar na posição desejada.

Com relação à trajetória descrita pelo efetuador, pode-se dizer que:

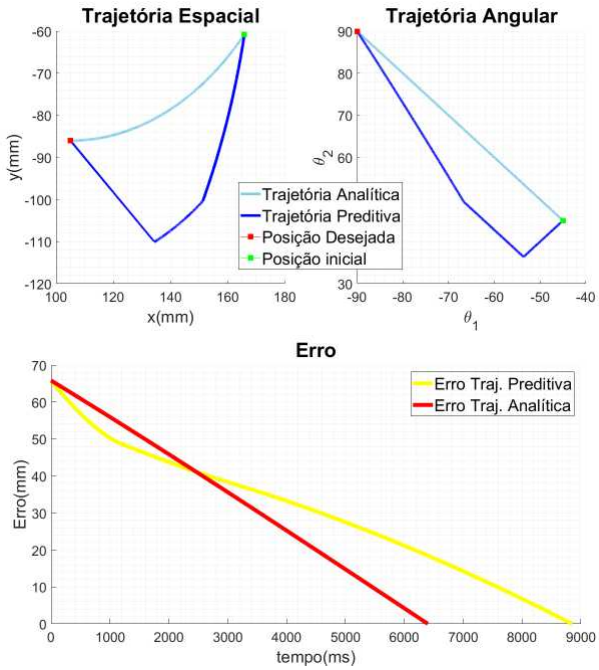
Figura 33 – Simulação N^o 1: trajetórias espacial e angular e erro para a posição desejada usando os métodos preditivo e analítico.



W

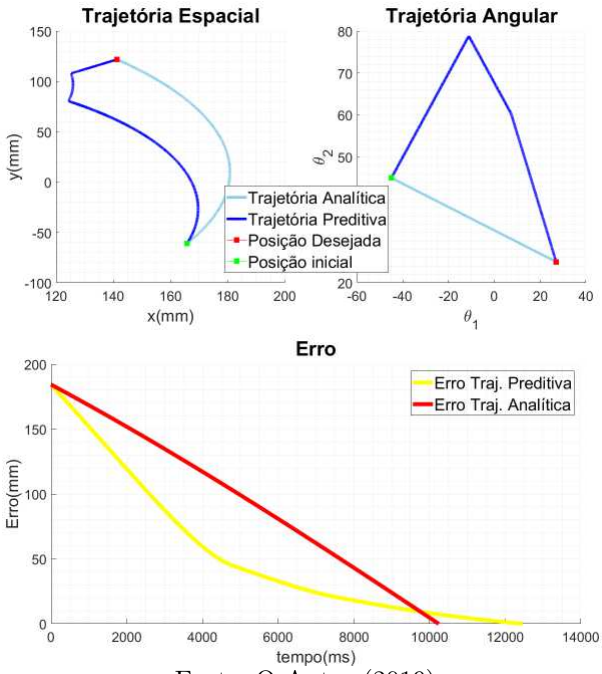
Fonte: O Autor (2019).

Figura 34 – Simulação N^o 2: trajetórias espacial e angular e erro para a posição desejada usando os métodos preditivo e analítico.



Fonte: O Autor (2019).

Figura 35 – Simulação N^o 3: trajetórias espacial e angular e erro para a posição desejada usando os métodos preditivo e analítico.



Fonte: O Autor (2019).

- O método preditivo segue uma trajetória imprevisível, definida por um segmento de reta e por um arco de parábola. Outro problema é que este método parece se aproximar do ponto desejado de uma forma completamente não intuitiva e variável;
- O método analítico com controle síncrono segue uma trajetória que parece muito com um arco de parábola e tem, como característica principal, a sua aproximação do ponto desejado com velocidade aproximadamente constante.

Assim, fica claro que o método analítico com controle síncrono é o mais adequado quando se deseja ter uma trajetória mais consistente entre dois pontos com um tempo reduzido de deslocamento.

3.5 SIMULAÇÕES DE TRAJETÓRIAS ESPECÍFICAS

Nesta seção são realizadas simulações com o objetivo de forçar uma trajetória específica. Uma solução possível para isso é definir um conjunto de coordenadas (x_o, y_o, z_o) próximas umas às outras, formando uma trajetória bem definida. Assim, quando um método qualquer for aplicado, o efetuador terá que gerar pequenas trajetórias entre cada par de pontos e assim definir uma trajetória mais específica. Apesar desta ser uma solução viável, é importante saber que existem outras maneiras de realizar esta tarefa e que,

para cada trajetória, existe uma maneira mais eficiente de ser realizada. Naturalmente, a melhor trajetória deveria ser aquela que minimiza uma dada função custo do tipo $\int_{P_o}^{P_f} F(\theta_1, \theta_2, Z)\text{ds}$, em que P_o e P_f são os pontos iniciais e finais da trajetória e F é a função-objetivo, que pode ser definida como a distância euclidiana do ponto desejado para a posição atual do efetuador. Porém, obter uma trajetória ótima baseado em algoritmo de minimização foge do escopo deste trabalho.

Desta maneira, a metodologia adotada é utilizar um conjunto de pontos próximos entre si para definir uma trajetória específica (no caso, um segmento de reta) e utilizar o método analítico em conjunto com o controle síncrono das juntas. Nesta simulação será fixado o ponto inicial e final da semirreta e o erro e o tempo da trajetória será verificado em função da quantidade de pontos intermediários que a trajetória terá que passar. Ao final desta simulação espera-se encontrar o número apropriado de pontos por mm de segmento de reta para que se tenha um bom desempenho ao se realizar uma reta. Em todas as trajetórias, foi usado, como ponto inicial, o ponto x=60.81mm e y=158.81mm e, como ponto final, o ponto x=60.81 e y=-158.81mm (distância de 316.62mm entre os pontos). Nas Figuras 36-40, podem ser verificadas as trajetórias espaciais do efetuador para diferentes quantidades de pontos internos que definem o segmento de reta. Os erros máximo e médio e o tempo total na realização da trajetória para diferentes quantidades de pontos internos que definem o segmento de reta podem ser vistos na Tabela 4.

Tabela 4 – Comparação dos erros e tempo de execução da trajetória para diferentes quantidades de pontos internos do segmento de reta.

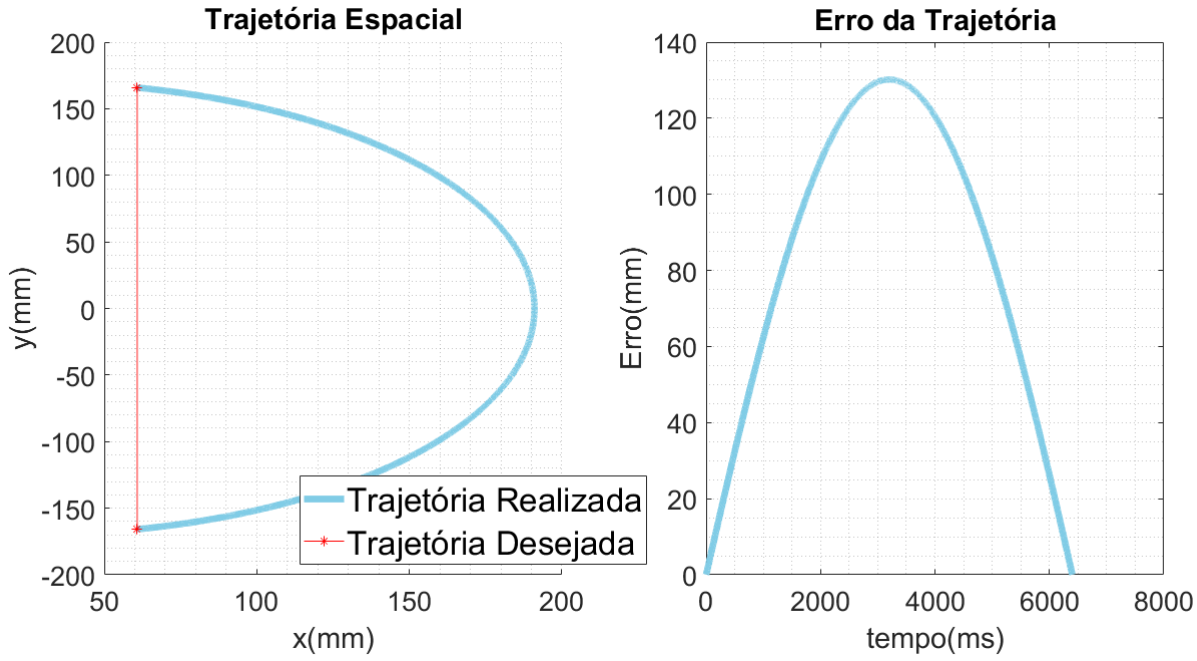
# Pontos Internos	Erro Máximo (mm)	Erro Médio (mm)	Tempo Médio (ms)
0	130,1888	83,4481	6.401
2	21,3640	11,9505	18.138
25	0,3664	0,1569	18.956
50	0,1448	0,0400	18.976
100	0,1322	0,0221	19.030

Fonte: O Autor (2019).

Diante das Figuras 36-40, é possível verificar que:

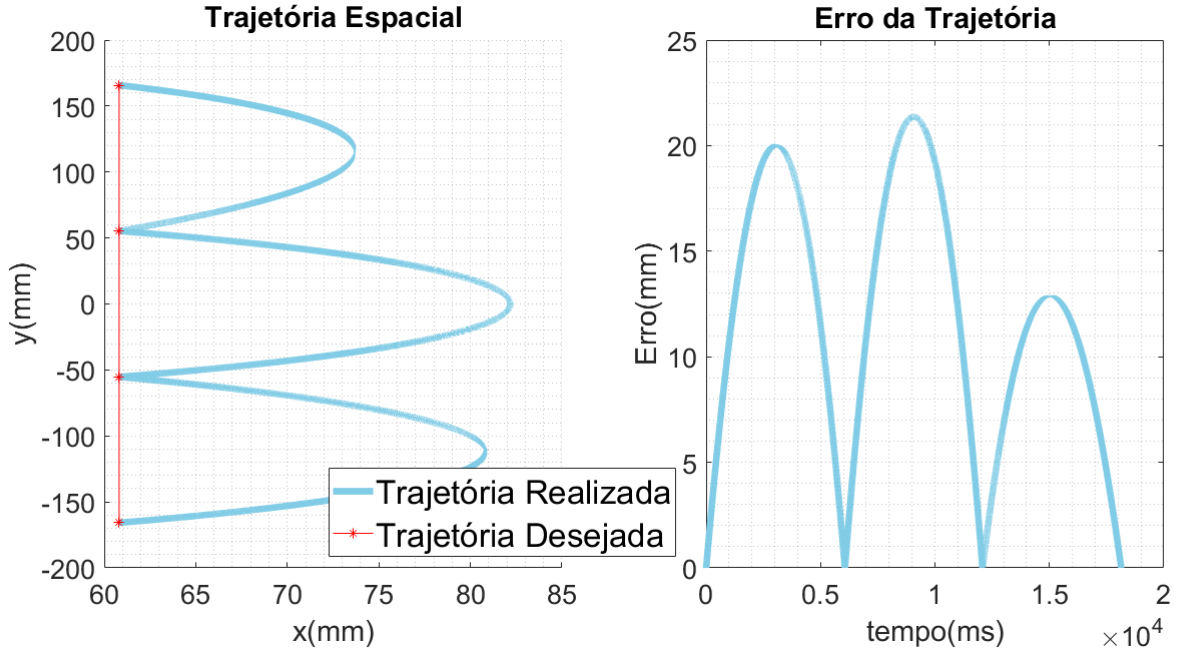
- Quanto menor o número de pontos intermediários, menor é o tempo (número de passos) para realizar a trajetória;
- Percebe-se que, a partir de uma certa quantidade de pontos (2 pontos), o aumento do número de pontos internos não afeta de forma apreciável o tempo de deslocamento.
- Quanto maior o número de pontos intermediários, menor é o erro durante a trajetória.

Figura 36 – Trajetória espacial e erro associado quando não há pontos internos no segmento de reta.



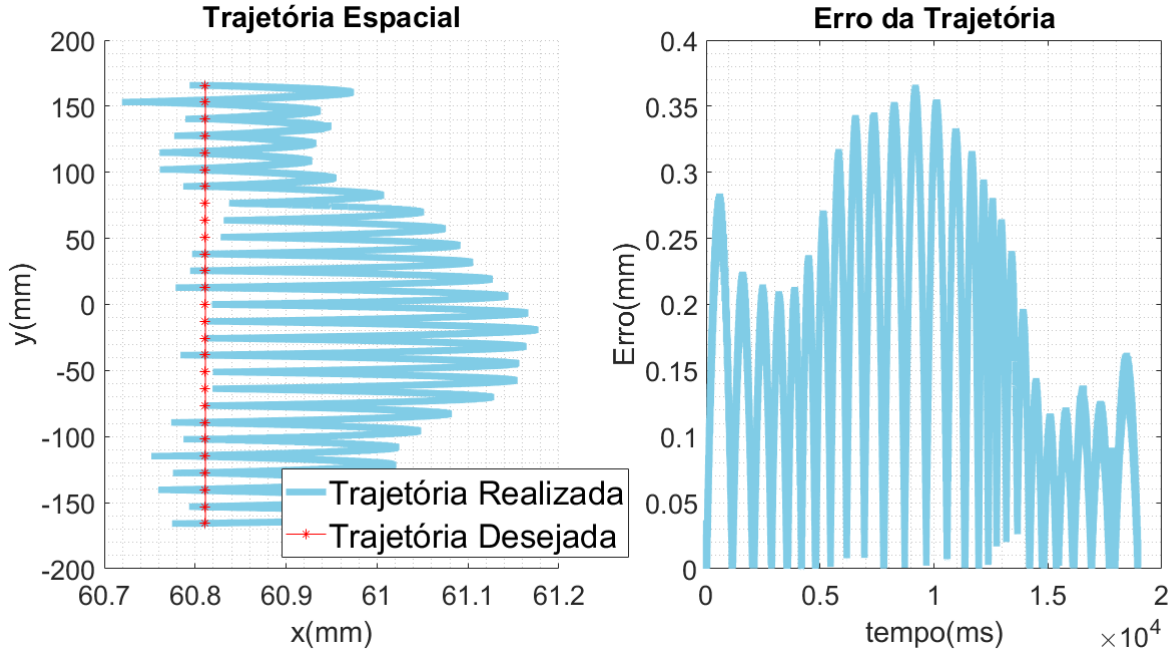
Fonte: O Autor (2019).

Figura 37 – Trajetória espacial e erro associado quando há 2 pontos internos no segmento de reta.



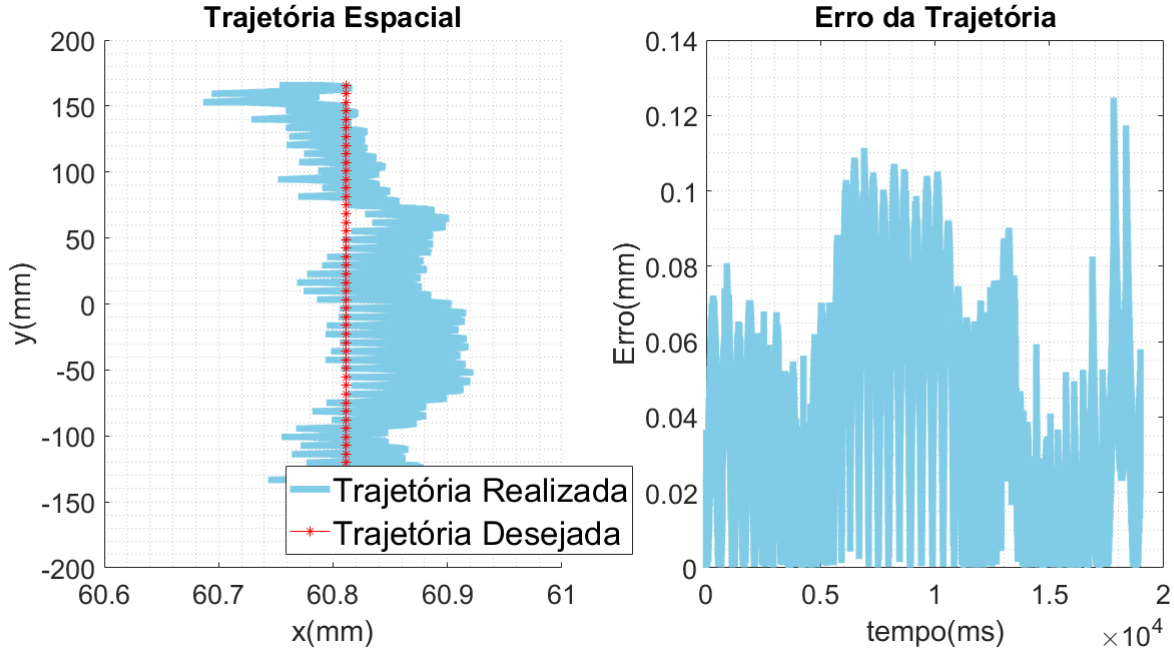
Fonte: O Autor (2019).

Figura 38 – Trajetória espacial e erro associado quando há 25 pontos internos no segmento de reta.



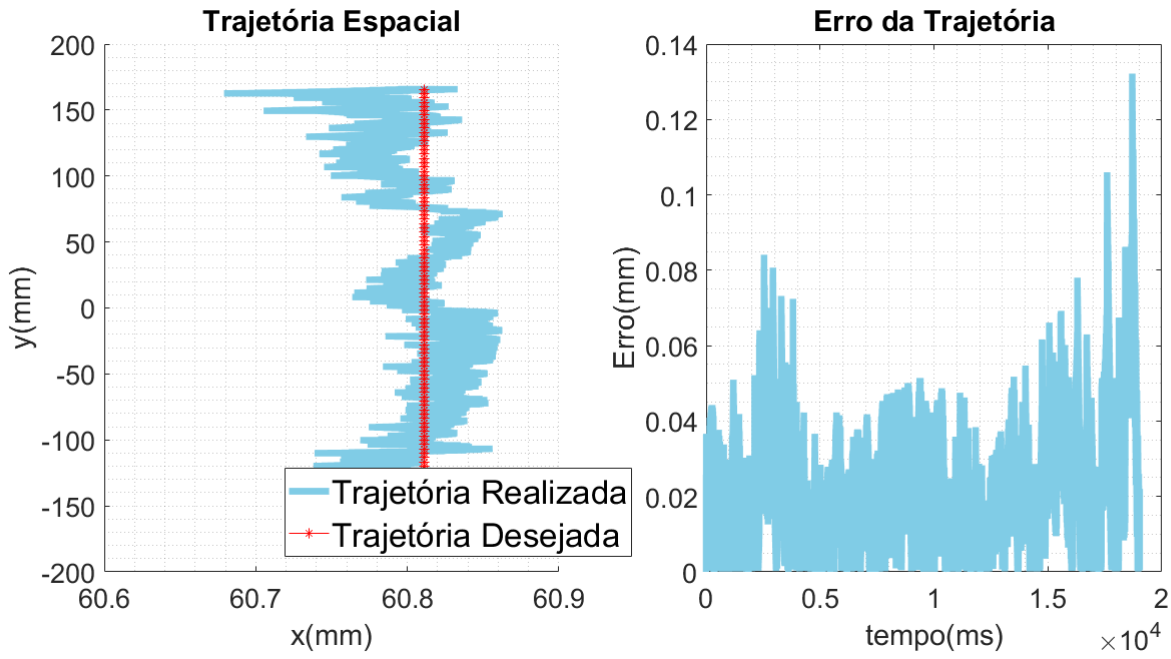
Fonte: O Autor (2019).

Figura 39 – Trajetória espacial e erro associado quando há 50 pontos internos no segmento de reta.



Fonte: O Autor (2019).

Figura 40 – Trajetória espacial e erro associado quando há 100 pontos internos no segmento de reta.



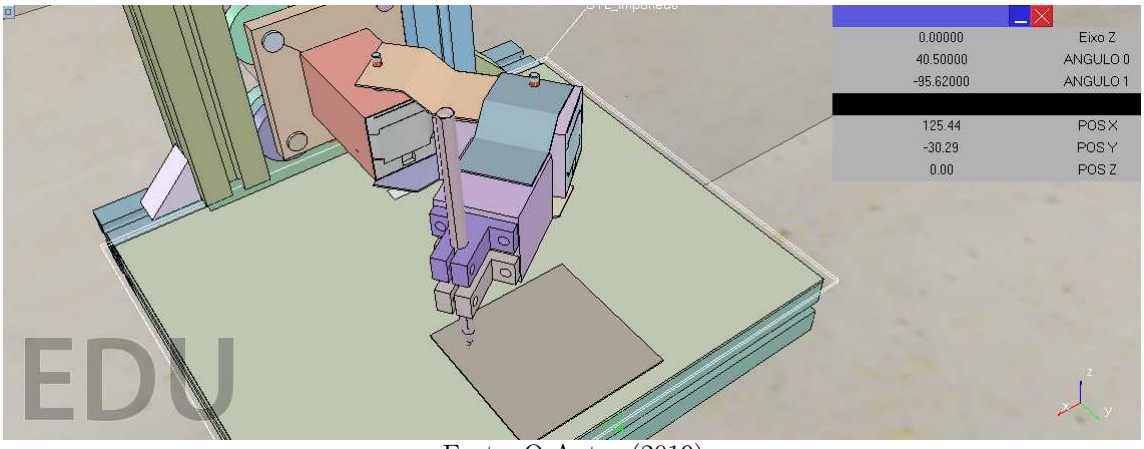
Fonte: O Autor (2019).

- Percebe-se que a partir de uma certa quantidade de pontos (50 pontos), não existe uma diminuição considerável do erro máximo e do erro médio das trajetórias.

Assim, fica claro que deve existir uma quantidade mínima de pontos intermediários que permita uma redução do erro quando o mesmo executa certos tipos de trajetórias. Assim, foi escolhida uma relação de 50 pontos internos para cada 317 mm de segmentos de reta a serem executados pelo efetuador. A partir de então, define-se uma densidade de um ponto intermediário a cada 6 mm de trajetória que demarca um segmento. Com esta definição de densidade o braço robótico passou de erro máximo de 40% para um erro de 0,04%. Assim, o sistema terá a capacidade de executar um conjunto de trajetórias baseadas em segmento de reta com um precisão da ordem de décimos de milimetro.

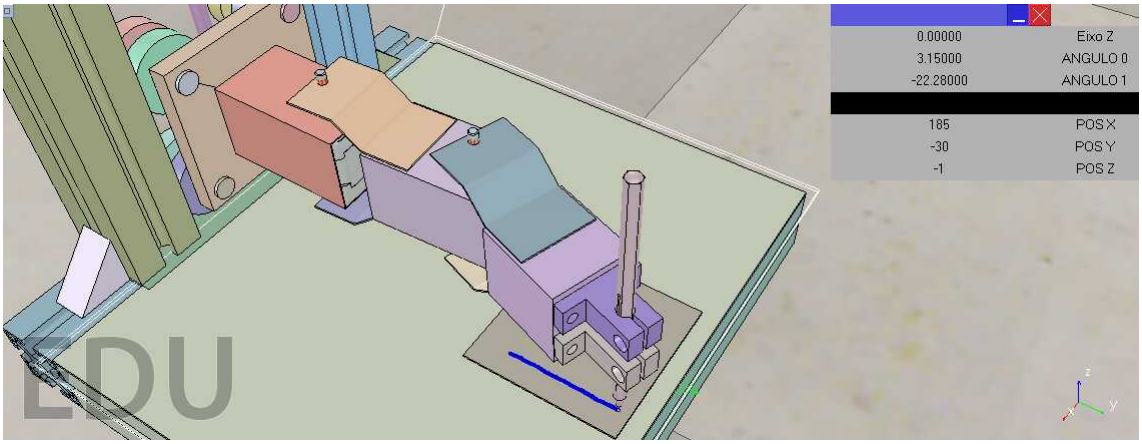
Para verificar melhor esta propriedade, foi executada uma simulação no ambiente virtual V-REP com a densidade mínima de pontos de um ponto intermediário a cada 6mm. Nesta simulação, foi solicitado ao sistema que ele simula-se o robô desenhar uma reta ao percorrer uma reta do ponto (x=120mm,y=-30mm) até o ponto (x=185mm,y=-30mm). Os resultados antes e depois desta simulação podem ser verificados nas Figuras 41 e 42, em que a reta está destacada em azul.

Figura 41 – Simulação que representa a posição inicial do braço robótico antes de desenhar a reta no ambiente V-REP.



Fonte: O Autor (2019).

Figura 42 – Simulação que representa a posição final do braço robótico após desenhar a reta no ambiente V-REP.



Fonte: O Autor (2019).

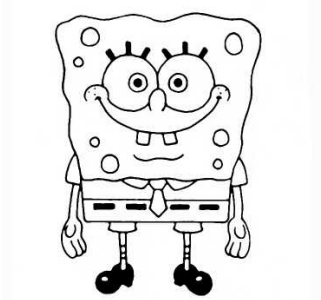
3.6 SIMULAÇÕES DE TAREFAS COMPLEXAS

Nesta etapa, é verificado o desempenho do braço robótico realizando tarefas complexas com a cinemática inversa sendo feita pelo método analítico com controle síncrono das juntas e uma densidade mínima de um ponto a cada 6mm de trajetória. Essa etapa é importante para verificar o desempenho do sistema em realizar um conjunto de trajetórias (retas) que definem algo desafiador a ser realizado por um braço robótico.

A primeira simulação consiste em realizar o desenho de um personagem de desenho animado como mostrado na Figura 43. Para realização da simulação, a figura foi vetorizada e filtrada fazendo assim com que ela fosse composta por um conjunto de vários segmentos de reta que formam o desenho. A simulação, realizada no ambiente MATLAB, pode

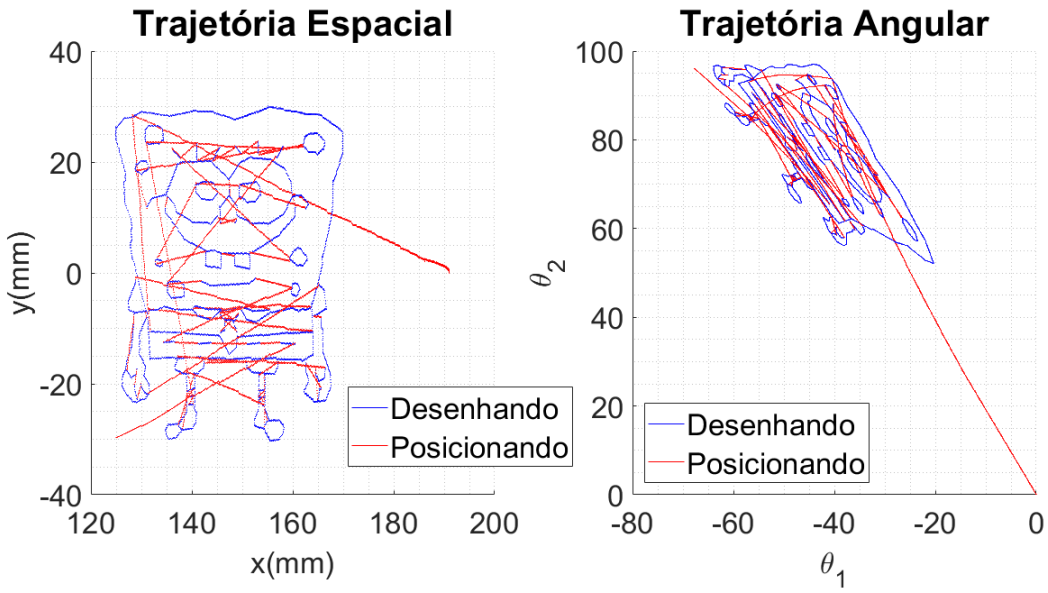
ser vista na Figura 44, em que a curva em amarelo representa o deslocamento com a caneta levantada e a curva em azul o deslocamento com a caneta tocando o papel. Na própria Figura 44, é possível verificar a sequência de posições angulares dos motores que propiciaram a execução do desenho.

Figura 43 – Primeiro desenho a ser realizado pelo braço robótico configurado com o método analítico e controle síncrono das juntas.



Fonte: SILYLANDIA (2018).

Figura 44 – Simulação em MATLAB do primeiro desenho, mostrando as trajetórias espaciais do efetuador e angulares das juntas do robô.



Fonte: O Autor (2019).

Analisando a Figura 44, percebe-se que a simulação foi um sucesso e foi obtido o desenho sem grandes deformações.

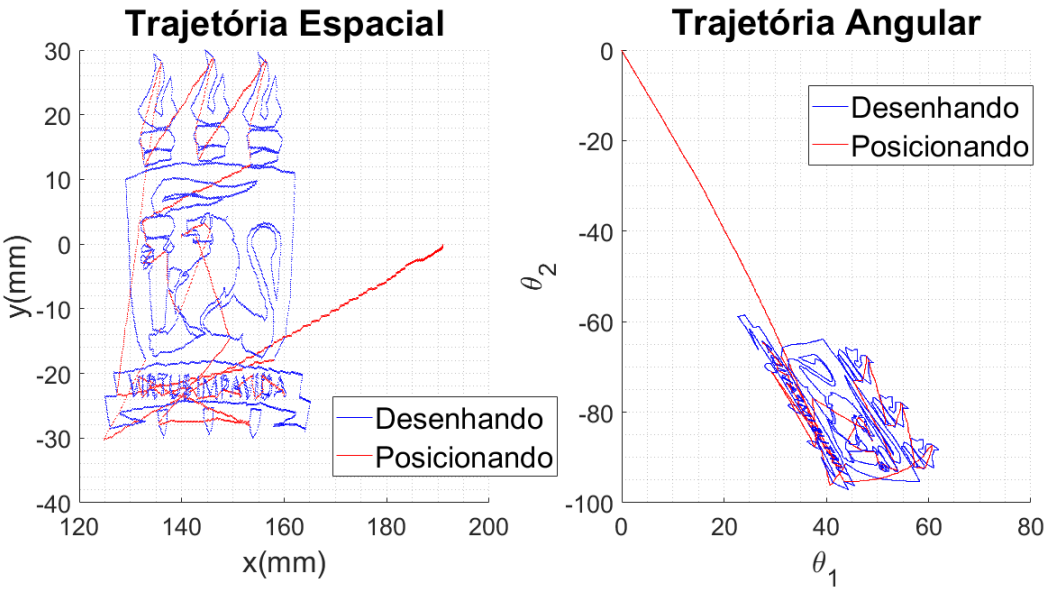
A segunda simulação consiste em realizar o desenho da logomarca da Universidade Federal de Pernambuco como mostrado na Figura 45. Para realização da simulação, a figura passou pelos mesmos processos de vetorização e filtragem da figura anterior. A simulação realizada no ambiente MATLAB pode ser vista na Figura 46, em que a curva em amarelo

representa o deslocamento com a caneta levantada e a curva em azul o deslocamento com a caneta tocando o papel. Na própria Figura 46 é possível verificar a sequência de posições angulares dos motores que propiciaram a execução do desenho.

Figura 45 – Segundo desenho a ser realizado pelo braço robótico configurado com o método analítico e controle síncrono das juntas.

Fonte: UFPE (2018).

Figura 46 – Simulação em MATLAB do segundo desenho, mostrando as trajetórias espaciais do efetuador e angulares das juntas do robô.

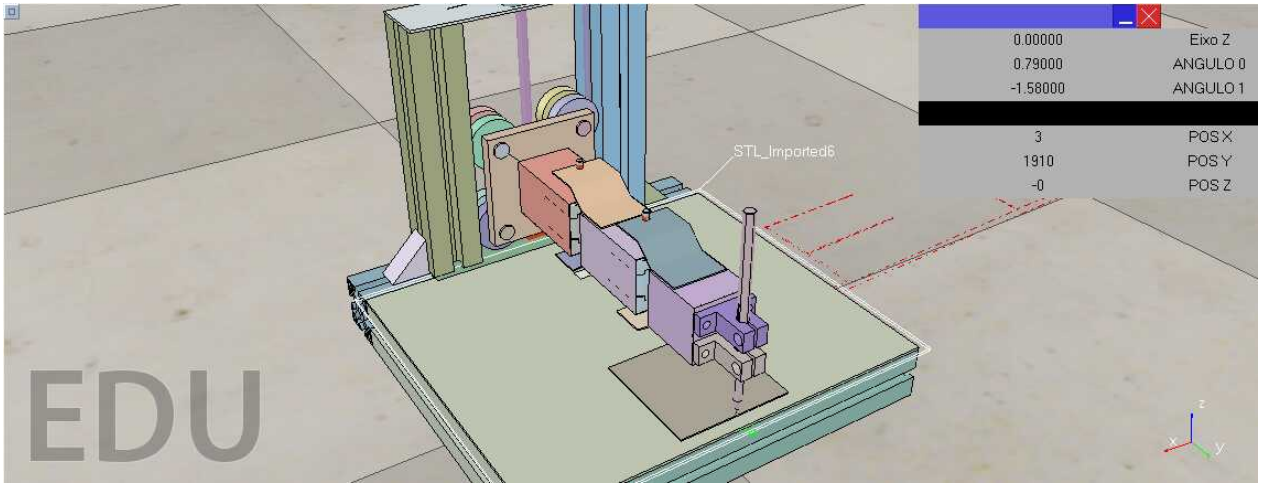


Fonte: O Autor (2019).

Analisando a Figura 46, percebe-se que novamente que a simulação foi um sucesso.

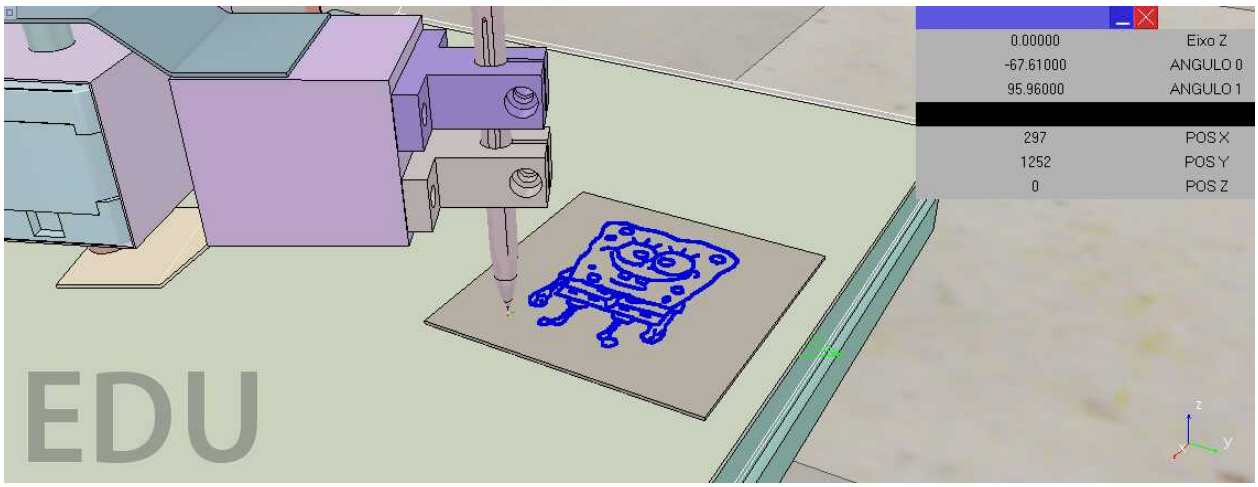
Para verificar melhor a viabilidade de se executar tais desenhos, ambas as simulações foram realizadas no ambiente virtual V-REP seguindo as mesmas trajetórias descritas nas Figuras 44 e 46. A posição inicial e as posições finais de cada simulação podem ser vistas nas Figuras 47-49.

Figura 47 – Posição inicial das simulações de desenhos complexos no ambiente V-REP.



Fonte: O Autor (2019).

Figura 48 – Posição final da simulação do primeiro desenho no ambiente V-REP.



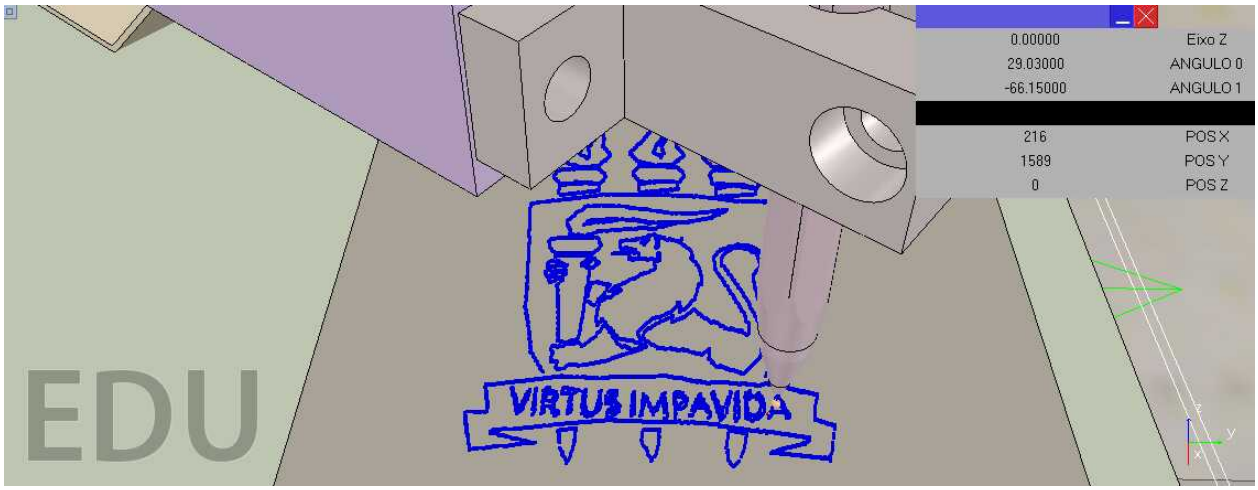
Fonte: O Autor (2019).

Diante das simulações apresentadas no V-REP, verifica-se que o método utilizado para a cinemática inversa e para o controle das juntas, além da densidade de pontos escolhida, garantem resultados satisfatórios para a execução de desenhos complexos pelo braço robótico simulado.

3.7 CONCLUSÃO

A análise dos processos de criação do desenho 3D e as análises estruturais foram de fundamental importância para definir características construtivas e utilizar este desenho

Figura 49 – Posição final da simulação do segundo desenho no ambiente V-REP.



Fonte: O Autor (2019).

3D para elaborar simulações em ambientes virtuais. Quanto às simulações, estas avaliaram a viabilidade de realizar métodos de cinemática direta e inversa, discutir diversas maneiras de realizar controle das juntas, determinar o numero de pontos intermediários que são necessários para realizar retas de forma satisfatória.

As conclusões das simulações em MATLAB, se tratando da metodologia mais adequada de controle para deslocamento em cinemática direta, é usar o controle síncrono visto que este contribui para um movimento com menores esforços mecânicos (quando comparado com o controle direto). Com relação ao método de solucionar a equação da cinemática inversa, o mais adequado foi utilizar o método analítico visto que este tem um tempo de solução pequeno e é independente da distância ao ponto desejado (quando comparado com os métodos numérico e preditivo). Com relação à realização de trajetórias entre dois pontos, percebe-se que o método analítico com controle síncrono é o mais adequado, quando comparado com o método preditivo, visto que o método preditivo assume trajetórias imprevisíveis enquanto o método analítico com controle síncrono assume trajetórias bem comportadas e que minimizam o tempo de deslocamento. Com relação à realização de uma reta entre dois pontos, verificou-se que a distância máxima adequada entre cada ponto intermediário deve ser de pelo menos 1 ponto a cada 6mm, visto que essa relação mantém um baixo erro durante a trajetória e realiza o movimento em um tempo adequado. Assim, verifica-se que a metodologia utilizada para executar toda e qualquer movimentação do SCARA deve ser realizada utilizando o método analítico com controle síncrono e pontos intermediários a cada 6mm.

Analisando a simulação de tarefa complexa, verificou-se um excelente desempenho em realizar a tarefa tanto no ambiente MATLAB quanto no ambiente V-REP, assim, a metodologia proposta foi um sucesso. A simulação com tarefa complexa reafirma que todas as simulações prévias chegaram em conclusões adequadas para o desenvolvimento da metodologia de controle do sistema. Assim, nos próximos capítulos, essas escolhas definirão a forma como o protótipo real irá se movimentar.

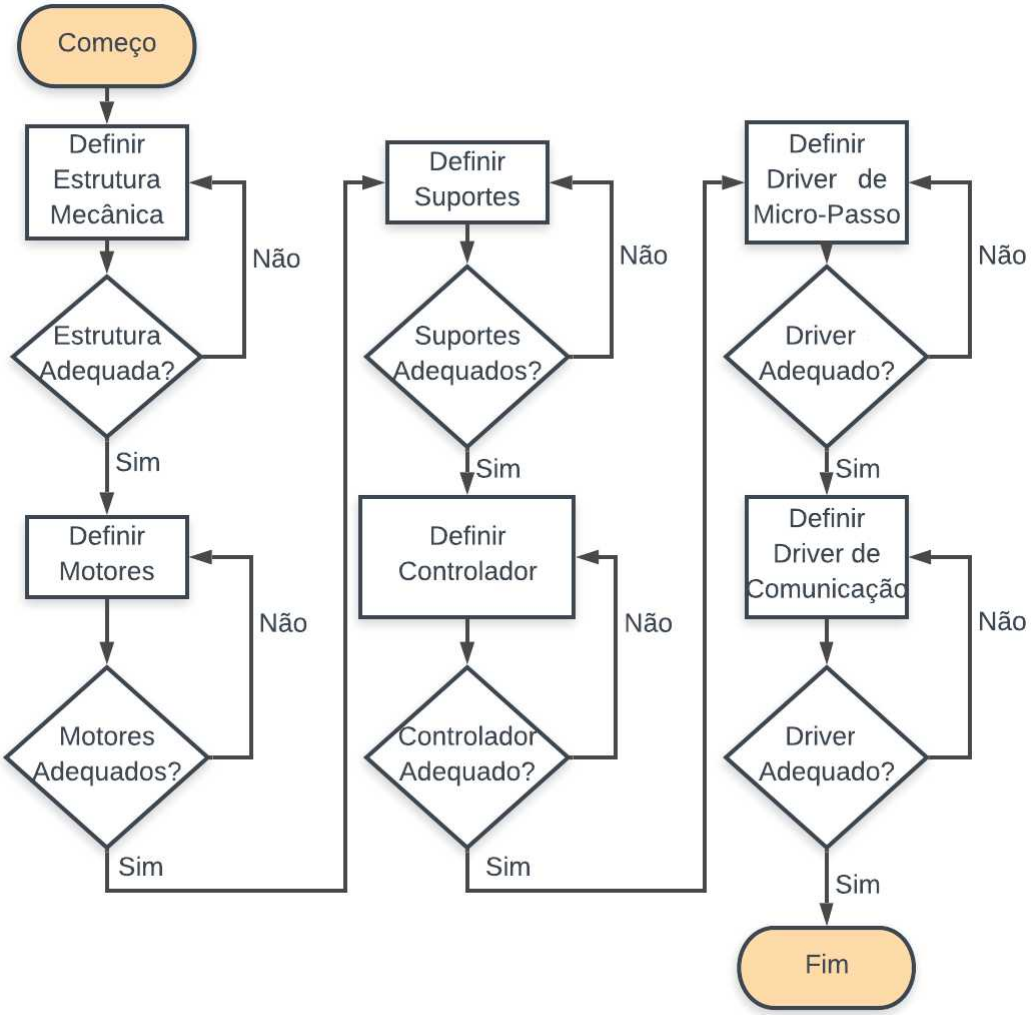
4 CONSTRUÇÃO DO BRAÇO SCARA E SEU SISTEMA DE CONTROLE

Neste capítulo serão apresentadas as etapas realizadas para desenvolvimento da estrutura, dos sistemas de controle, da comunicação entre os sistemas, do posicionamento inicial do braço robótico e da IHM.

4.1 DESCRIÇÃO DA ESTRUTURA ROBÓTICA E SEUS ATUADORES

Como dito na Seção 3.1, foi usado como referência o braço robótico mostrado na Figura 20. De forma simplificada, todas as escolhas estruturais e de hardwares seguiram o fluxograma mostrado na Figura 50.

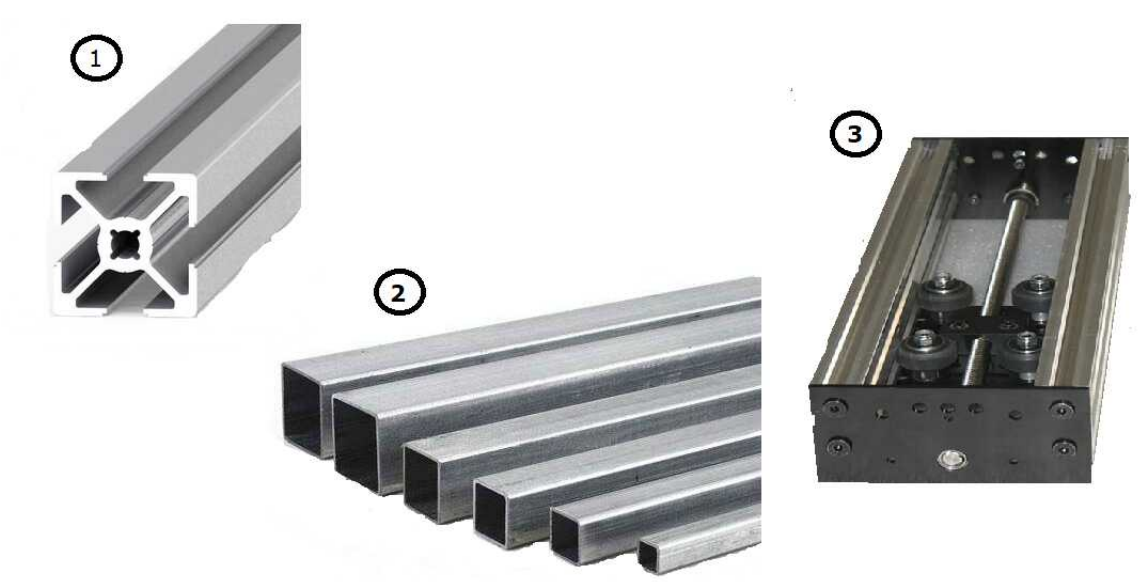
Figura 50 – Fluxograma do processo utilizado para definição das escolhas estruturais e de hardwares.



Fonte: O Autor (2019).

Basicamente, a estrutura deste braço robótico é composta de uma base que utiliza perfis de alumínio estrutural e uma articulação de aço galvanizado. O eixo móvel em Z é um produto comercial que é muito utilizado em CNCs. Na Figura 51 pode ser visualizado ilustrações destas estruturas presentes no SCARA.

Figura 51 – Ilustrações de estruturas presentes no SCARA.



Fonte: O Autor (2019).
1) Perfil de Alumínio Estrutural; 2) Peças de Aço Galvanizado; 3) Estrutura de um eixo móvel em Z.

Com relação aos atuadores, foram utilizados três motores de passo, padrão nema 17, sendo dois de 3kgf e um de 5kgf. A escolha dos motores foi baseada na necessidade da utilização de micro-passo com resolução de até 1/128 e torque suficiente para movimentar a estrutura. Por questões de disponibilidade, estes motores foram superdimensionados em relação ao torque, pois é difícil encontrar motores que realizem micro-passos com uma resolução tão alta e que tenham baixo torque. Na Figura 52 é possível verificar ilustrações dos motores de passo, padrão nema 17 utilizados no braço SCARA.

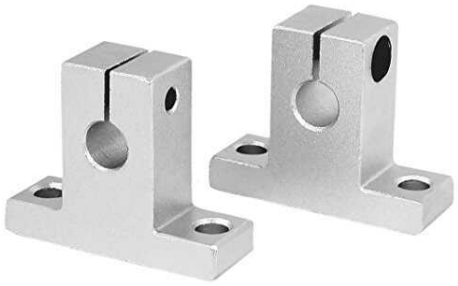
Com base na análise da Seção 3.2, foi verificado que a estrutura utilizada é adequada para a realização do experimento visto que os esforços são facilmente suportados pela estrutura. Com relação ao efetuador, é desejado que este seja capaz de manipular alguma ferramenta que possa qualificar o trabalho. Em outras palavras, um efetuador que consiga demonstrar que o braço robótico SCARA realizou uma trajetória especificada. Assim, modificou-se o projeto de referência para que este tenha a capacidade de portar uma caneta ou um lápis que tenha a capacidade de riscar em um papel a trajetória que foi

Figura 52 – Ilustração dos motores de passo, padrão nema 17 utilizados no SCARA.

Fonte: WOTIOM (2019).
1) 3kgf; 2) 5kgf.

realizada. Para isto, a estrutura escolhida foi usar um suporte sk8 ou um suporte sk12. Na Figura 53 são mostrados estes suportes.

Figura 53 – Representação dos suportes sk8 e sk10 usados para fixar o efetuador do braço SCARA.



Fonte: 4HOBBY (2018).

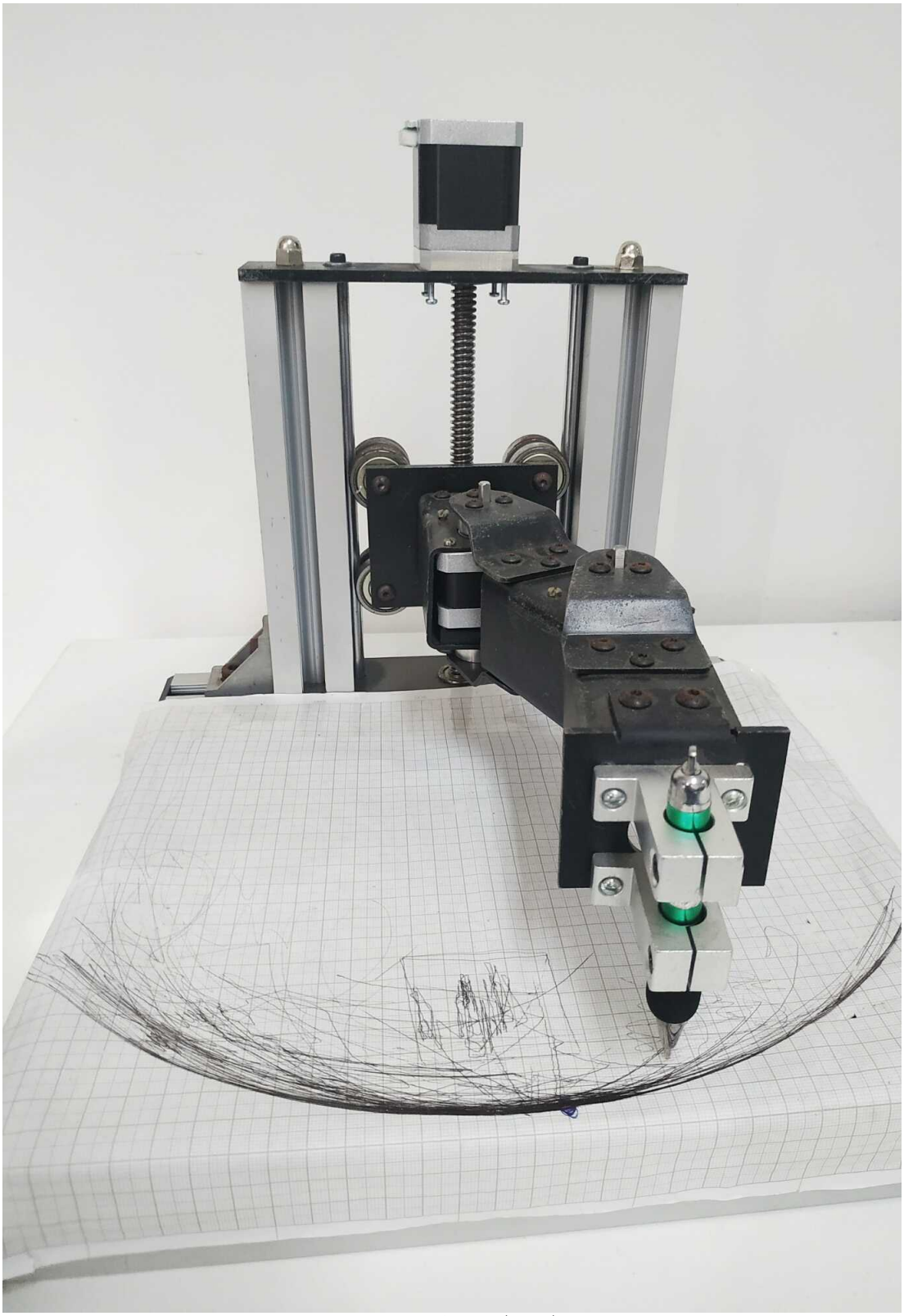
Ao final de todas estas adaptações, temos na Figura 54 o SCARA desenvolvido. Deve-se atentar à semelhança deste com o desenho projetado anteriormente na Figura 21.

Assim, com esta estrutura definida, pode-se começar a desenvolver os sistemas de controle que serão capazes de realizar o acionamento dos motores e receber os comandos da IHM baseados no protocolo de comunicação.

4.2 ELABORAÇÃO DO SISTEMA DE CONTROLE

Um dos primeiros passos na elaboração do sistema de controle foi escolher qual o controlador que será utilizado no sistema. As tarefas que este microcontrolador deve ser

Figura 54 – SCARA desenvolvido.



Fonte: O Autor (2019).

capaz de realizar envolvem receber comandos de uma IHM, interpretar as informações com o protocolo, e realizar o acionamento dos drivers de micro-passo. Existem diversos microcontroladores que conseguem realizar este tipo de tarefa a baixo custo e sem ter um processador. Assim, o escolhido para esta tarefa foi um PIC16F887, pois este tem um baixo custo ,é amplamente utilizado nos laboratórios da UFPE e o autor tem uma grande familiaridade com este. Na Figura 55 é possível verificar este microcontrolador.

Figura 55 – Microcontrolador PIC16F887 usado no sistema de controle.

Fonte: MICROCHIP (2018).

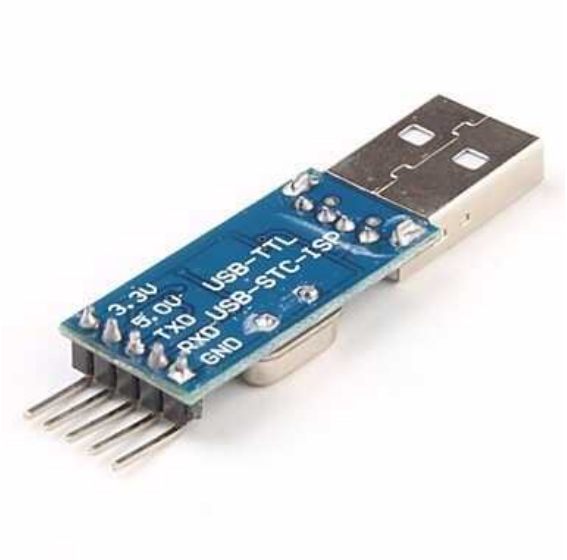
Como dito anteriormente, serão utilizados drivers de micro-passo. Estes drivers deverão ser capazes de realizar um aumento de resolução na escala de 1/128 e de fornecer uma corrente suficiente para o funcionamento dos motores. Assim, o driver de motor de passo escolhido para realizar estes acionamentos foi o DM542, que é capaz de operar com uma tensão de até 50V e injetar uma corrente controlada de até 4.2A. Na Figura 56 é possível verificar este driver de acionamento de micro-passo e suas conexões.

Figura 56 – Driver de micro-passo DM542 e suas conexões.

Fonte: TINEL (2018).

Para possibilitar a comunicação entre o microcontrolador e a IHM(computador) utilizou-se um conversor USB para Serial PI-2303HX. A escolha deste foi devido ao fato de existir muita documentação sobre ele e do mesmo ser muito popular e barato no mercado nacional. Na Figura 57 é possível verificar este conversor.

Figura 57 – Conversor USB-Serial PI-2303HX.



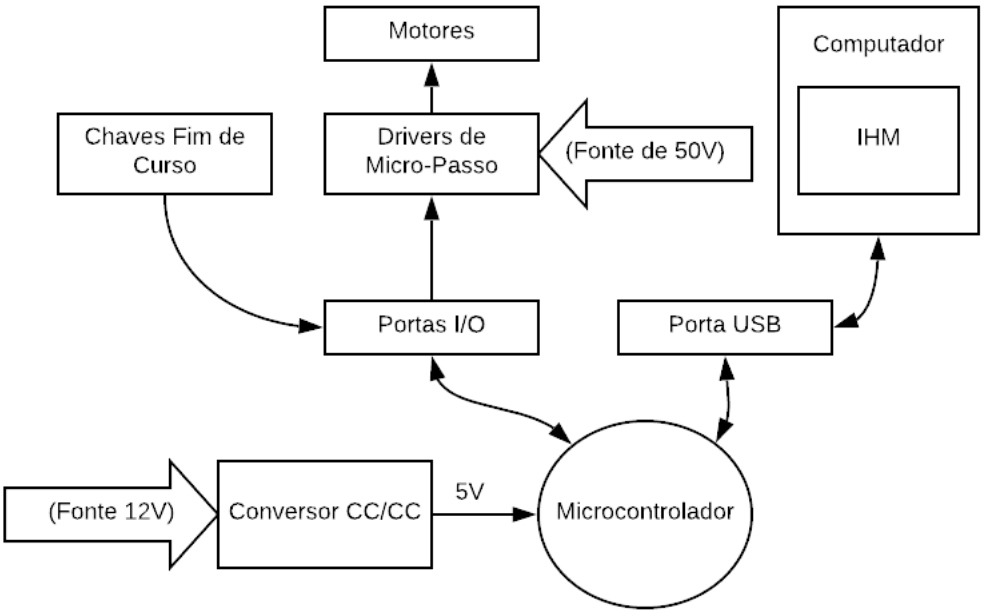
Fonte: MICROWAT (2018).

Com o microcontrolador, drivers de micro-passo e conversor USB-Serial definidos, foi projetada então uma placa de circuito impresso que se utiliza destes componentes de tal maneira a fornecer uma estrutura básica para o funcionamento do microcontrolador e tornar disponíveis os pinos do mesmo para conexão com outros componentes. Na Figura 58 pode ser visto um diagrama simplificado do sistema de controle. Nesta figura são mostradas todas as interações com o microcontrolador relacionadas a alimentação e fluxo de informação. Vale lembrar que neste projeto foram utilizados conversores CC-CC (convertendo 12V para 5V) do tipo BUCK.

A Figura 59 ilustra o diagrama esquemático do circuito do sistema de controle, desenvolvido no ambiente ISIS do PROTEUS (ambiente de desenvolvimento de circuitos). Através desse circuito foi elaborada uma placa de circuito impresso que foi utilizada para embarcar o microcontrolador e seus periféricos. Na Figura 60 é mostrado este circuito desenvolvido.

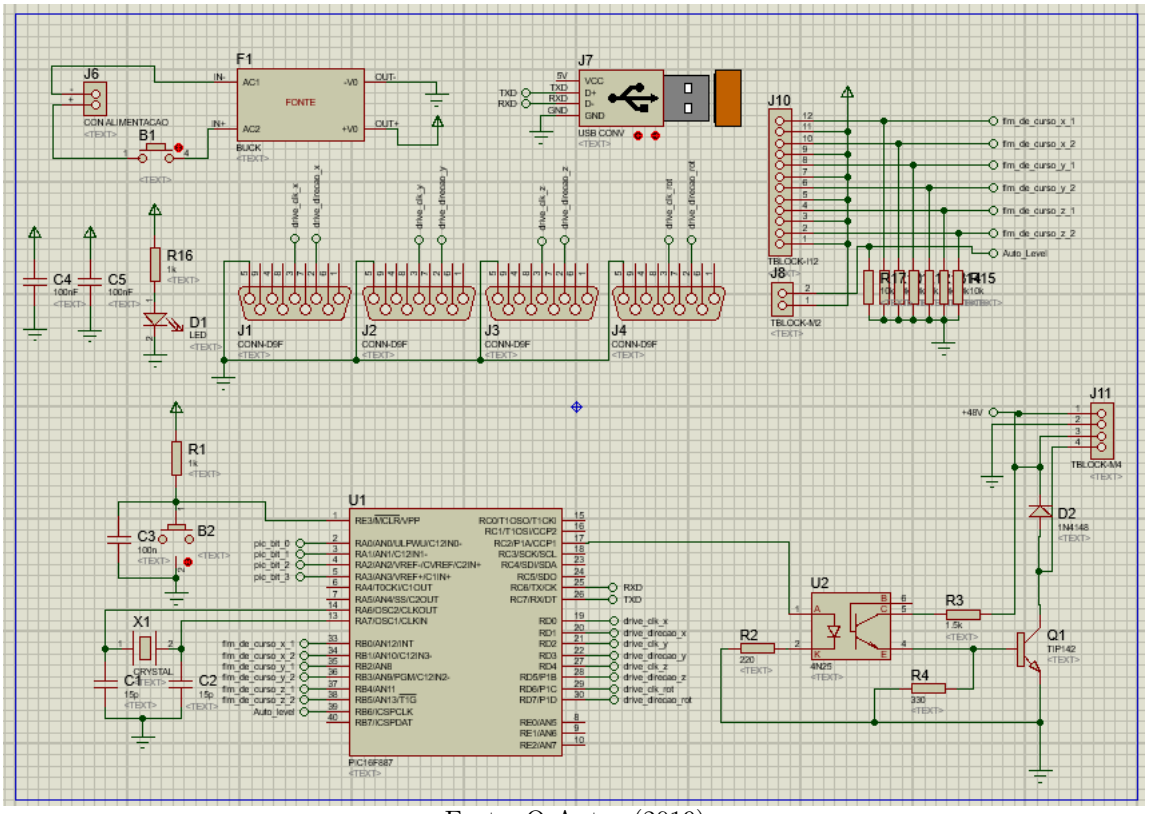
A programação do firmware embarcado no PIC16F877 pode ser verificada no Apêndice C. O objetivo deste código é verificar se existe alguma tentativa de comunicação entre a IHM e o microcontrolador e caso positivo, deve ser realizado um conjunto de

Figura 58 – Diagrama simplificado do sistema de controle.



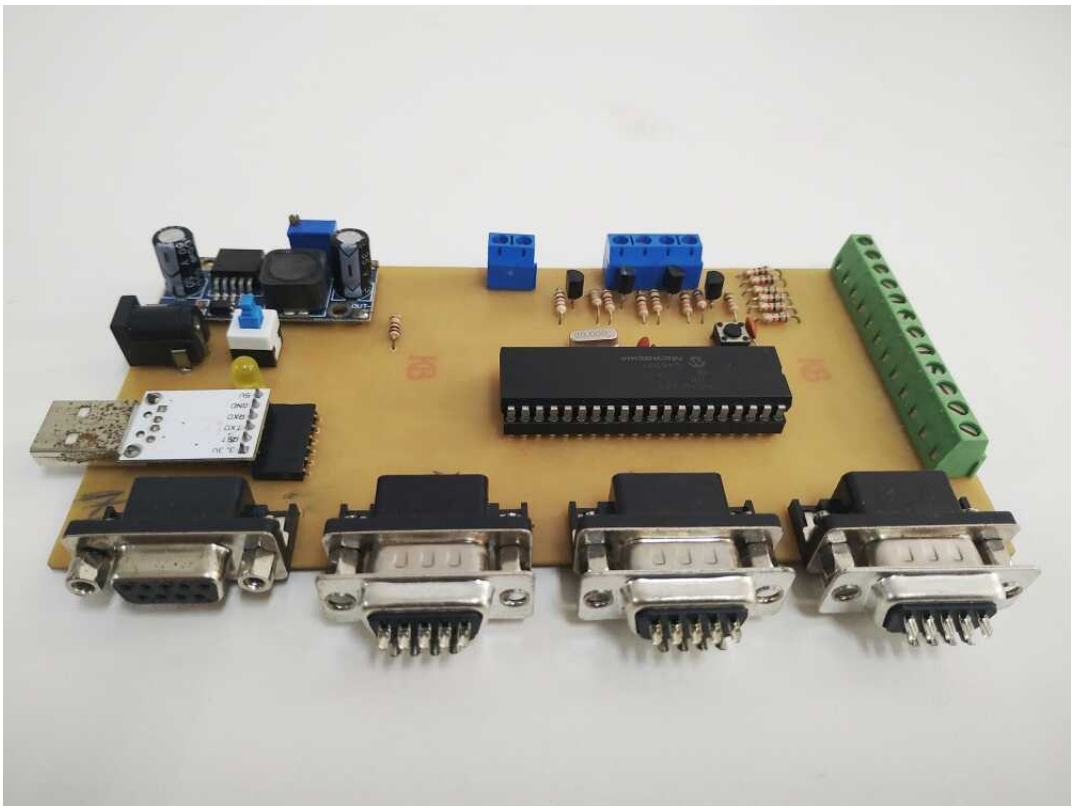
Fonte: O Autor (2019).

Figura 59 – Diagrama esquemático sistema de controle no ISIS-PROTEUS.



Fonte: O Autor (2019).

Figura 60 – Circuito desenvolvido.



Fonte: O Autor (2019).

atividades definidas pelo protocolo de comunicação desenvolvido. Basicamente, a função do microcontrolador é traduzir os comandos enviados pela IHM, comandar os drivers de micro-passo e detectar eventuais problemas que devem ser informados a IHM.

4.3 ELABORAÇÃO DO PROTOCOLO DE COMUNICAÇÃO

Para que o sistema de controle consiga realizar as tarefas demandadas pela IHM foi necessário criar um protocolo de comunicação dedicado, que estabelece as regras de comunicação entre o IHM e o microcontrolador.

A comunicação definida para este tipo de tarefa foi uma comunicação serial em que um cabo USB foi conectado do computador ao microcontrolador através de um conversor USB/Serial PI-2303HX. A configuração de comunicação utilizada foi de um *baud rate* de 115200bps, 8 bits de dados e um bit de stop ,em que, apenas 1 byte foi utilizado para realizar o controle dos motores.

Basicamente, a estrutura dos dados transmitidos segue um protocolo tal que para um dado byte uma ação será realizada. Primeiro, é verificado se o byte tem valor maior

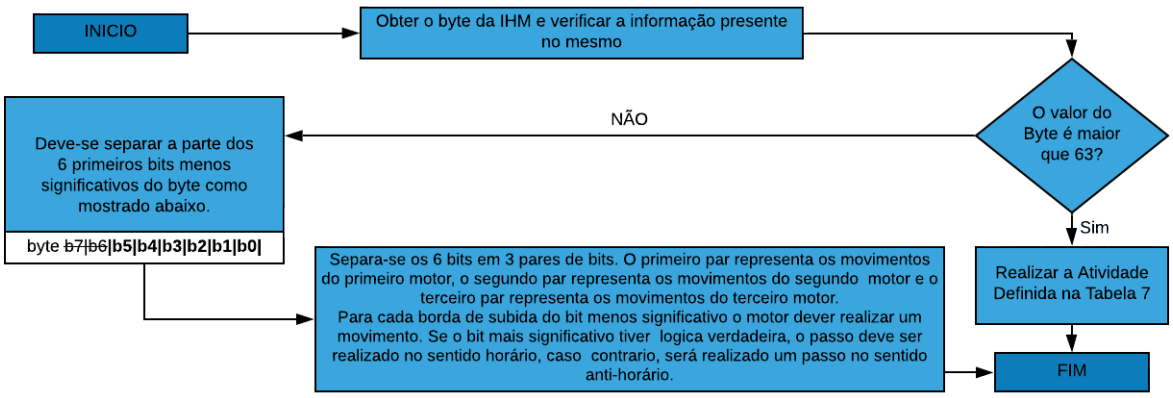
que 63 e caso esta afirmação seja verdade uma ação será realizada com base na Tabela 5. Caso contrário, será realizada uma operação de acionamento de motores, em que os movimentos têm relação com o posicionamento de cada um dos 6 bits menos significativos presentes neste byte. Todo o processo realizado durante a comunicação pode ser analisado no fluxograma mostrado na Figura 61.

Tabela 5 – Ação realizada para um dado byte.

Valor do byte	Ação Realizada
124	Realizar posicionamento automático
125	Liberar/Ligar motor0
130	Liberar/Ligar motor1
140	Liberar/Ligar motor2
150	Liberar todos os motores

Fonte: O Autor (2019).

Figura 61 – Fluxograma que representa o protocolo de comunicação realizado entre a IHM e o sistema de controle.



Fonte: O Autor (2019).

Assim, para qualquer operação realizada na IHM, um comando adequado será realizado por meio do envio do byte com os bits ordenados de tal forma a informar ao sistema de controle (microcontrolador) a tarefa que deve ser executada.

4.4 INICIALIZAÇÃO

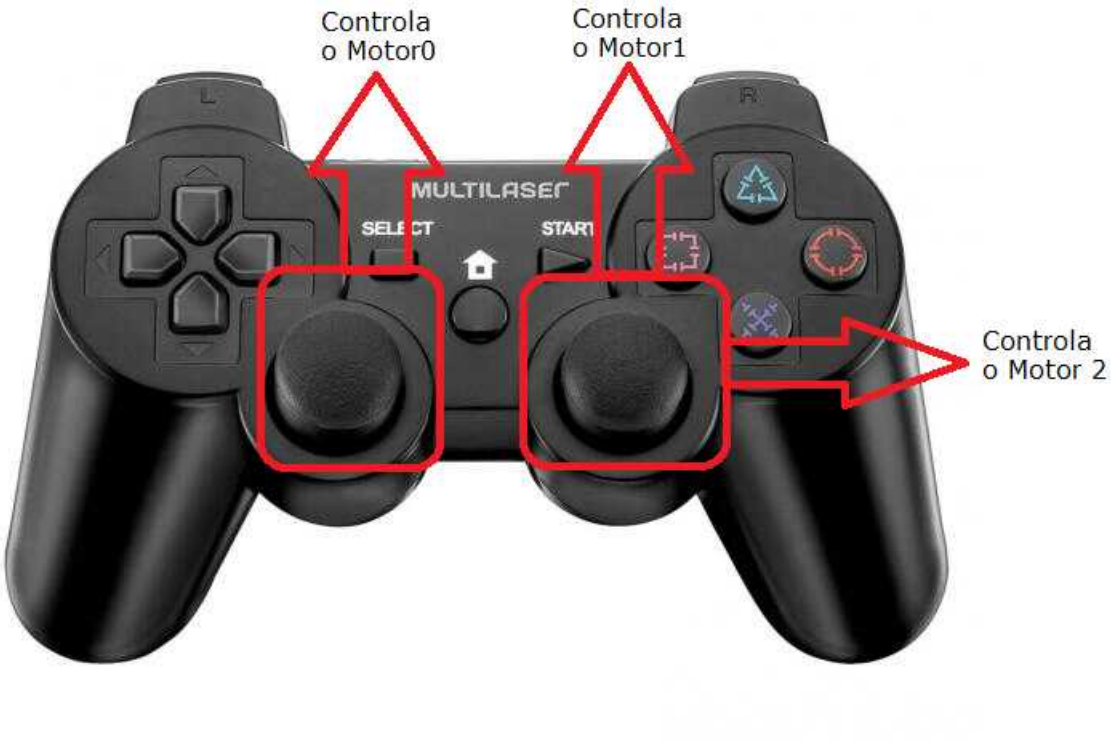
Nesta etapa foi adotado como objetivo desenvolver um algoritmo capaz de reconhecer o ponto de referência. A obrigatoriedade de uma inicialização ocorre porque sempre que o braço robótico é desligado extingue-se sua excitação elétrica e por consequência, perde-se sua posição final. Assim, é preciso implementar uma rotina que permita fazer com

que o braço robótico mova-se para uma posição conhecida no início das suas atividades. As metodologias propostas para obter uma posição conhecida dos motores envolvem as seguintes opções:

- Utilizar um joystick que consiga forçar o sistema a uma posição inicial conhecida.
- Utilizar chaves do tipo fim de curso em regiões específicas para que as informações destas possam ser utilizadas para definir uma posição escolhida.

Para a primeira opção, deve-se realizar uma programação na IHM que forneça a opção de controlar o sistema de modo manual utilizando um joystick. Assim, é necessário que na IHM existam opções que permitam capturar os comandos do joystick e utilizá-los para realizar o movimento dos motores. Desta forma, temos que uma parte do joystick será responsável por controlar o motor1 e a outra será responsável por controlar o motor0 e o motor2. Na Figura 62 é possível verificar a divisão destes controles no joystick

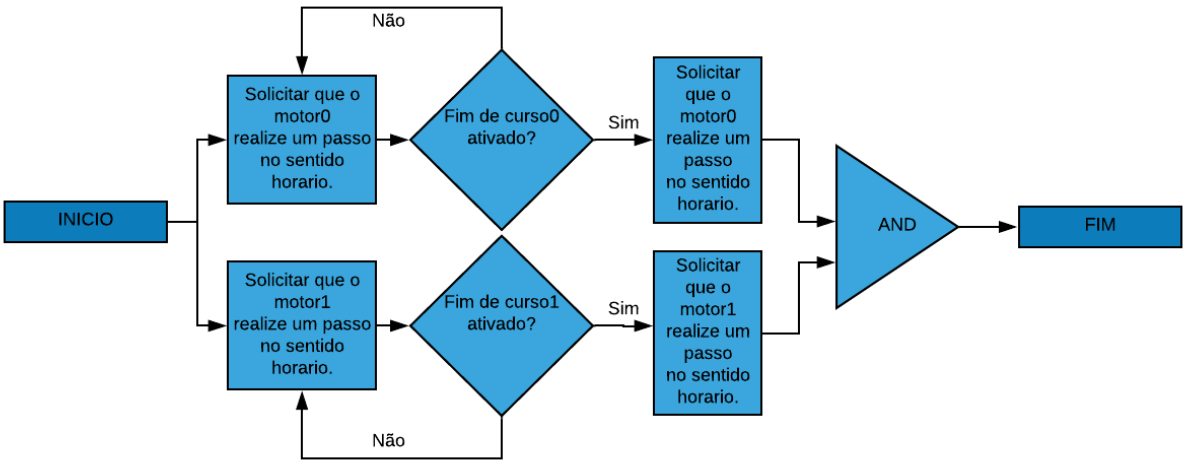
Figura 62 – Divisão dos controles realizada no joystick.



Fonte: MULTILASER (2019).

Para a segunda opção, utilizou-se chaves fim de curso e foi desenvolvida uma rotina no microcontrolador que segue o fluxograma mostrado na Figura 63. Este fluxograma demonstra de forma simplificada como pode ser realizado um posicionamento inicial com base nas informações obtidas da chave fim de curso.

Figura 63 – Fluxograma de funcionamento da calibração automática usando fim de curso.



Fonte: O Autor (2019).

Assim, para qualquer situação inicial dos motores, existe uma rotina que pode ser utilizada para que a estrutura possa ser posicionada em uma determinada posição específica.

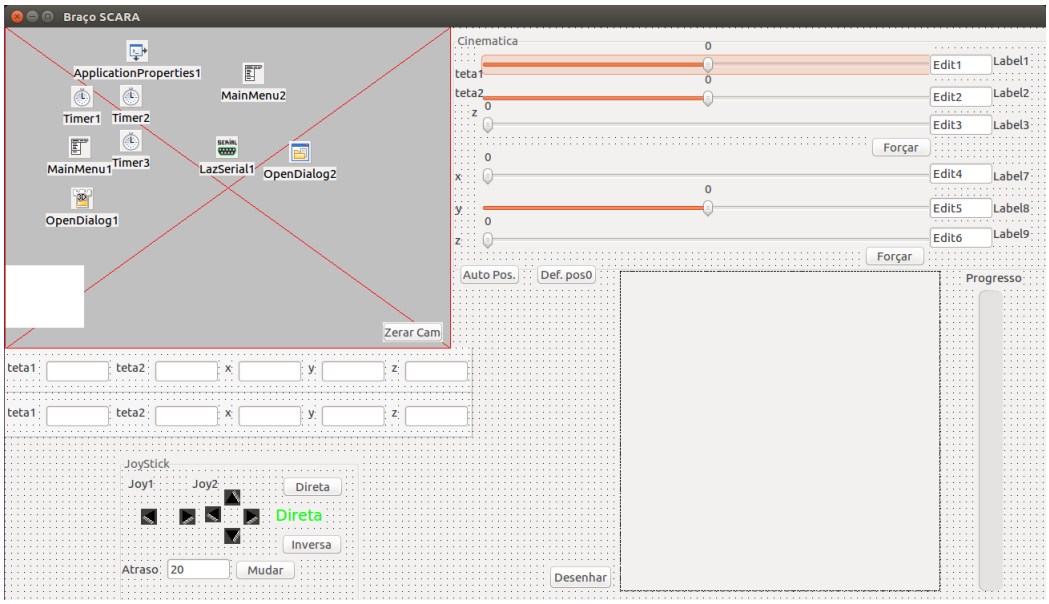
4.5 ELABORAÇÃO DA IHM

Para elaboração da IHM foi utilizado o ambiente de desenvolvimento LAZARUS (LAZARUS, 2019). Neste ambiente, foi desenvolvida uma IHM que contém elementos que sejam capazes de realizar as diversas tarefas solicitadas. A IHM foi desenvolvida utilizando bibliotecas, tais como, Castle Engine (ENGINE, 2019) para desenvolvimento de elementos 3D e LazSerial para estabelecer uma comunicação serial com o microcontrolador. A tela da IHM desenvolvida pode ser vista na Figura 64.

As funcionalidades desta IHM são:

- Realizar simulações com um braço robótico virtual usando joystick, barras de rolagem e trajetórias pré-programadas;
- Realizar movimentos no braço SCARA usando joystick;
- Solicitar que o braço robótico SCARA seja posicionado usando barra de rolagem;
- Solicitar que o braço robótico realize um conjunto de movimentos pré-programados.

Figura 64 – IHM desenvolvida no ambiente Lazarus.



Fonte: O Autor (2019).

É importante ressaltar que todas as contas relacionadas a cinemática inversa, calculo de trajetórias e posicionamento inicial são realizados pela IHM. Ou seja, a IHM é responsável por fazer todo o controle de posição, armazenar a posição atual, definir que tipo de trajetória será realizado, etc. Resumidamente, apenas o acionamento dos motores, conforme solicitado pela IHM, é realizado pelo microcontrolador.

Todas estas opções podem ser selecionadas através de um menu que fica na parte superior da tela. Opções deste menu podem ser vistas na Figura 65.

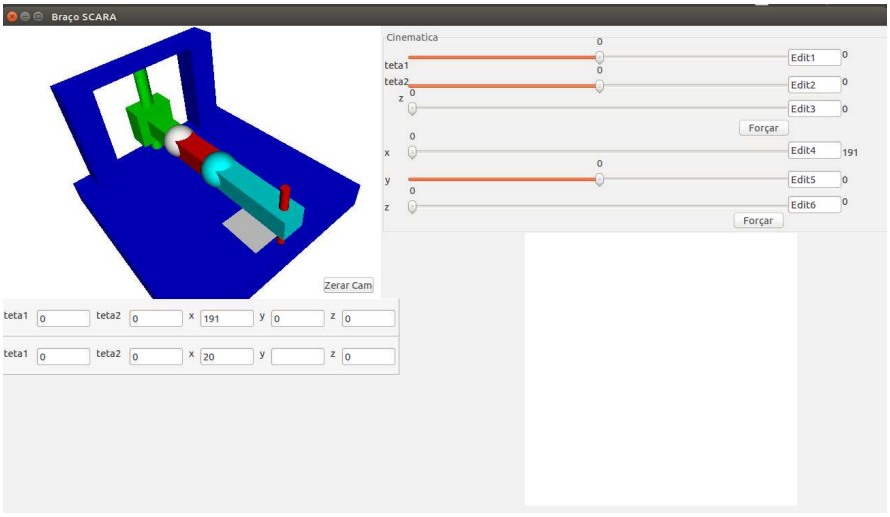
Figura 65 – Parte do menu presente na IHM.

Fonte: O Autor (2019).

É importante ressaltar que para cada uma das opções mostradas na Figura 65 teremos telas diferentes para realizar a interação com o usuário. Para o caso da opção "Barras"ser escolhida, teremos uma tela como mostrada na Figura 66. Para o caso da escolha da opção "Joystick", teremos uma tela como mostrado na Figura 67. Para o caso da escolha da opção de "Trajetória Prog.", teremos uma tela como mostrado na Figura 68.

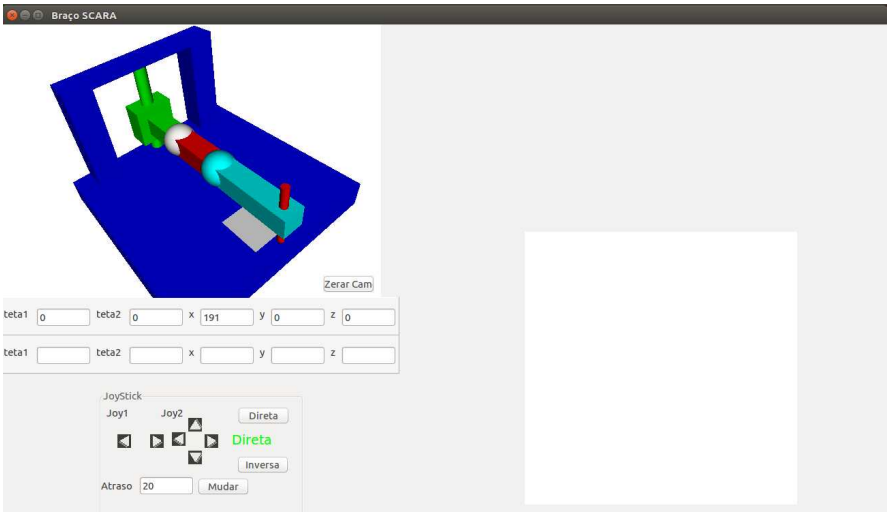
Na Figura 66 é possível observar que surgem barras de rolagem no campo superior direito da tela e que a partir dessa barra de rolagem é possível solicitar que o SCARA

Figura 66 – Tela com a opção barras selecionada.



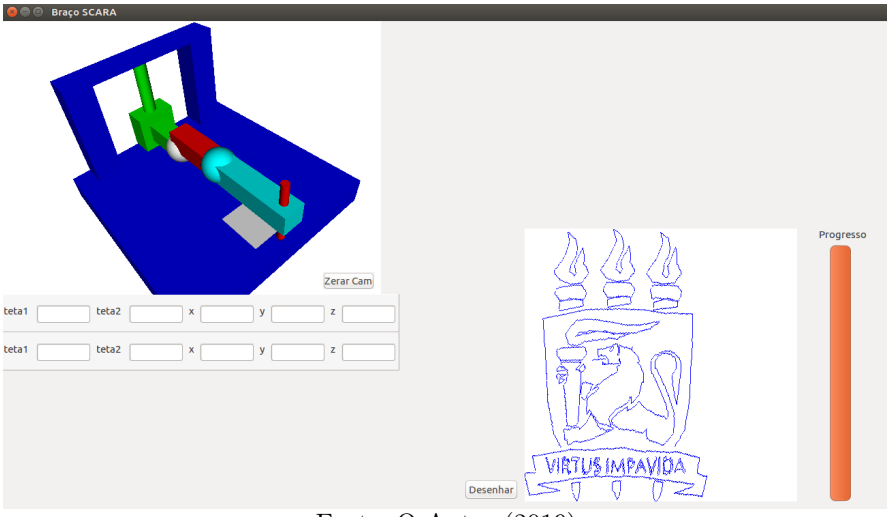
Fonte: O Autor (2019).

Figura 67 – Tela com a opção joystick selecionada.



Fonte: O Autor (2019).

Figura 68 – Tela com a opção de trajetória programada selecionada.



Fonte: O Autor (2019).

ocupe qualquer uma das posições possíveis. Repare que todo movimento solicitado será realizado no modelo virtual e no braço real, caso a opção controle real esteja ativada.

Na Figura 67 é possível observar que surge uma pequena região no canto inferior esquerdo em que pode ser escolhido o tipo de cinemática que será realizado (direta ou inversa). Cada uma das setas mostradas nesta imagem representa a posição pressionada pelo joystick e consequentemente o movimento realizado pelo SCARA.

Na Figura 68 é possível observar que o desenho selecionado aparece no canto inferior direito. Ao clicar no botão "desenhar", o braço robótico executará os comandos com a finalidade de realizar movimentos que façam o SCARA desenhar o desenho escolhido.

4.6 CONCLUSÃO

A descrição da estrutura robótica e de seus atuadores é de importe para justificar a escolha de cada um dos elementos utilizados no robô. As escolhas realizadas foram baseadas em resistência, peso, disponibilidade e características construtivas. Assim, foi escolhida uma estrutura que contenha grande similaridade com o projeto original, onde apenas pequenas modificações foram necessárias para dar-lhe a funcionalidade de portar canetas em seu efetuador. Quanto a escolha dos motores, escolheu-se aqueles que fossem capazes de fornecer torque adequado e a capacidade de serem acionados por micro-passos com resolução de até 1/128.

A elaboração do sistema de controle é uma parte fundamental do projeto é solicita a especificação de diversos componentes para a plena execução do sistema. O microcontrolador escolhido foi o PIC16f887 pois é um microcontrolador barato, com capacidade suficiente para realizar a tarefa, além disso, o autor tem grande familiaridade com este. Quanto à escolha dos drivers de micro-passo, escolheu-se aqueles que possuem a capacidade de executar micro-passos com resolução de até 1/128 e que possam fornecer corrente suficiente para alimentar os motores. Assim, foi escolhido o driver de micro-passo DM542. Quanto à escolha do periférico de comunicação, escolheu-se aquele que fosse capaz de se comunicar, utilizando uma interface serial, com a IHM (um computador), e que fosse um conversor usb/serial amplamente utilizado. Assim, o periférico de comunicação escolhido foi o PI-2303HX. Ao final, foi elaborado um programa que foi embarcado no microcontrolador e

construída uma placa de circuito impresso que contemplou todos os requisitos básicos de funcionamento do projeto.

O sistema de comunicação define como a comunicação foi realizada e o protocolo utilizado para que fosse possível uma integração entre o microcontrolador e a IHM. As velocidades de comunicação foram escolhidas de forma simplificada visto que a quantidade de dados que trafega é relativamente baixa (1 byte). Quanto ao protocolo, este foi definido para conseguir realizar tarefas específicas e controlar cada um dos motores como mostrado na Tabela 6 e no fluxograma da Figura 61.

Na Inicialização é demonstrada a problemática de não saber a posição inicial do SCARA e as duas possíveis soluções abordadas para solucionar este problema. Uma solução é usar um joystick para colocá-lo manualmente na posição desejada. Outra solução é usar chaves fim de curso para criar uma rotina automática de posicionamento. Esta rotina pode ser melhor compreendida pelo fluxograma na Figura 63.

Na elaboração da IHM foram demonstradas as funcionalidades da Tela criada no ambiente Lazarus. Esta IHM contempla uma virtualização do SCARA e funcionalidades tais como controle dos movimentos do SCARA por barras e por joystick. Outra funcionalidade é controlar o braço robótico de tal forma a realizar uma trajetória definida. A realização destas tarefas é de fundamental importância para verificar o desempenho do braço robótico nas mais variadas atividades.

Por fim, este capítulo é importante para entender como o braço robótico SCARA funciona e as etapas necessárias no desenvolvimento da estrutura, sistemas de controle, comunicação, inicialização do sistema e o ambiente de interface homem-máquina. Com estas etapas realizadas, agora é possível executar experimentos com o protótipo real e compará-lo com os resultados de simulação.

5 EXPERIMENTOS COM O PROTÓTIPO

Neste capítulo serão apresentados os experimentos realizados com o protótipo SCARA e os resultados obtidos. Espera-se que ao final deste capítulo seja possível verificar o desempenho do protótipo e compará-lo com os resultados das simulações.

5.1 EXPERIMENTO DE POSICIONAMENTO ANGULAR PELA IHM

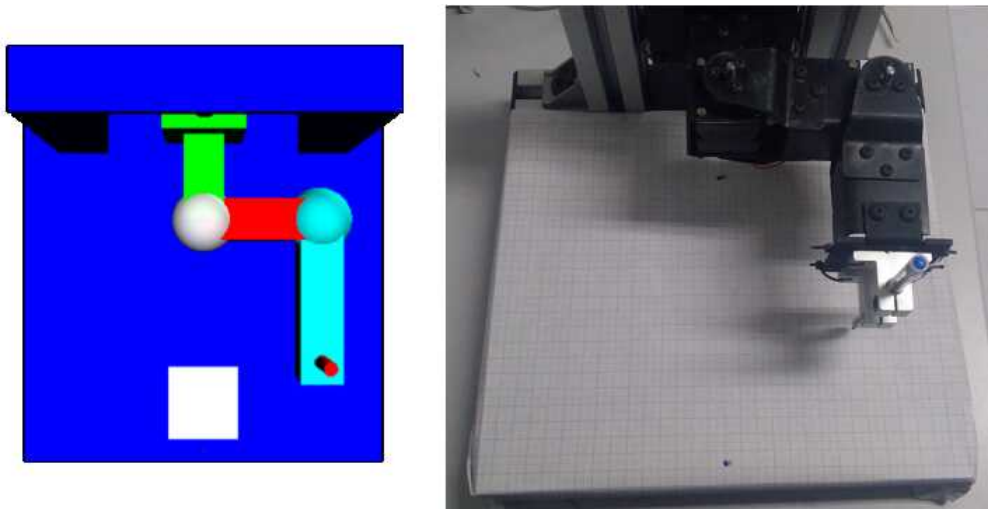
Para realizar o experimento de posicionamento angular foi adotada a metodologia de solicitar que o sistema seja posicionado em locais específicos (como na Seção 3.2). Esses pontos serão obtidos através de um papel milimetrado que será marcado por uma caneta (ferramenta do efetuador). O conjunto de posições angulares solicitadas ao sistema pode ser visto na Tabela 6. Nas Figuras 69-72 é possível observar, para cada simulação presente na Tabela 6, a posição final do SCARA ao lado de seu modelo virtual apresentado na IHM. É importante ressaltar que existe um erro nestas medições que é relacionado ao erro do papel milimetrado (+-0,5mm).

Tabela 6 – Resultados dos experimentos de posicionamento.

Experimento N^o	Posição Solicitada (θ_1;θ_2)	Posição Alcançada (θ_1;θ_2)	Erro Angular ($\Delta\theta_1$;$\Delta\theta_2$)
1	(90^o;-90^o)	(90^o;-90^o)	($0,0^o$;$0,0^o$)
2	(-90^o;90^o)	(-90^o;90^o)	($0,0^o$;$0,0^o$)
3	(45^o;-45^o)	(45;-45)	($0,0^o$;$0,0^o$)
4	(-45^o;45^o)	(-45^o;45^o)	($0,0^o$;$0,0^o$)

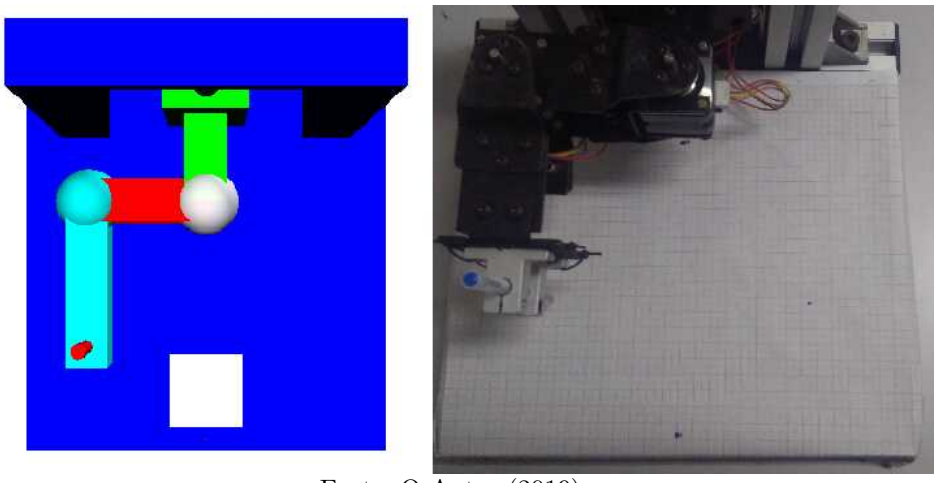
Fonte: O Autor (2019).

Figura 69 – Resultado do experimento N^o 1 com o SCARA virtual em conjunto com o SCARA real.



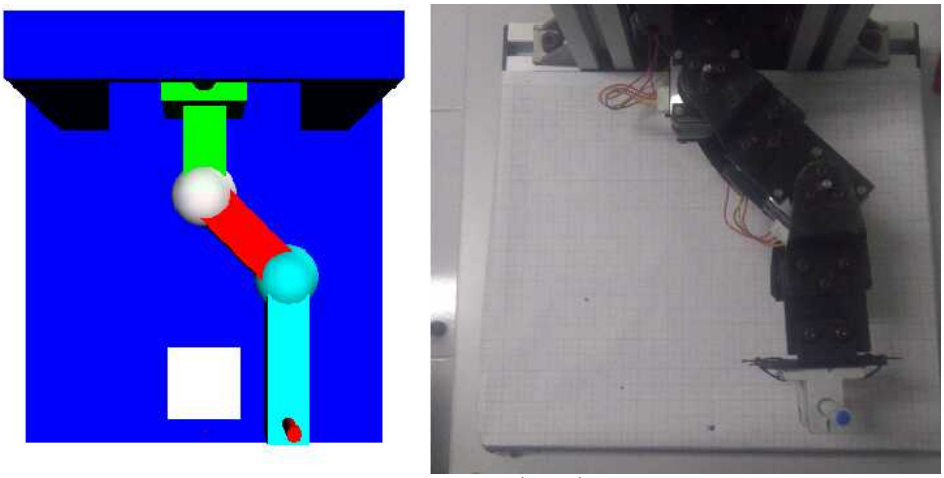
Fonte: O Autor (2019).

Figura 70 – Resultado do experimento Nº 2 com o SCARA virtual em conjunto com o SCARA real.



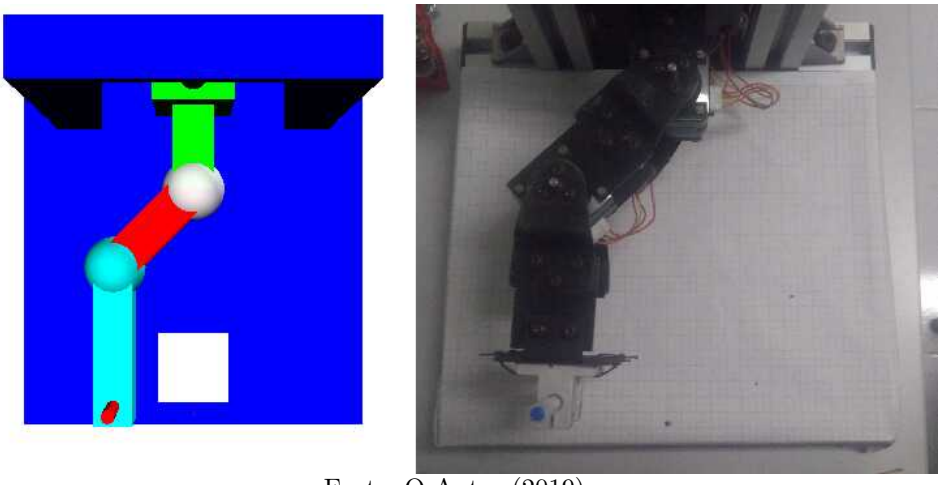
Fonte: O Autor (2019).

Figura 71 – Resultado do experimento Nº 3 com o SCARA virtual em conjunto com o SCARA real.



Fonte: O Autor (2019).

Figura 72 – Resultado do experimento Nº 4 com o SCARA virtual em conjunto com o SCARA real.



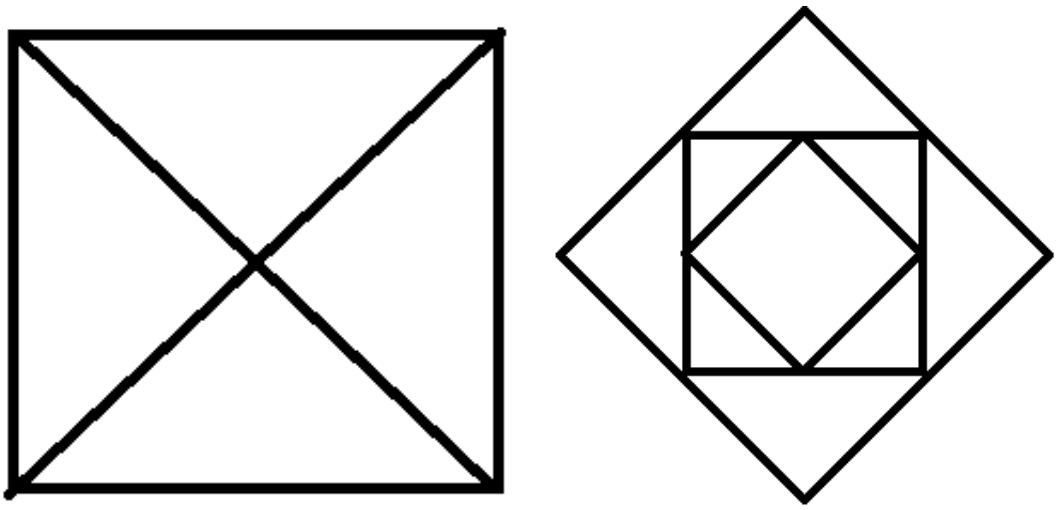
Fonte: O Autor (2019).

De acordo com os resultados mostrados na Tabela 6 e o posicionamento do SCARA mostrado nas Figuras 69-72 é possível perceber que o posicionamento solicitado era executado. Assim, esse experimento mostra a capacidade que o SCARA desenvolvido tem em realizar posicionamentos angulares.

5.2 EXPERIMENTO DE TRAJETÓRIAS ESPECÍFICAS PELA IHM

Para realizar o experimento de posicionamento pela IHM foi adotada a metodologia de solicitar que o sistema realizasse pequenos desenhos que demonstrem a capacidade de realizar um pequeno conjunto de retas. É importante ressaltar que durante a realização deste experimento foi utilizado o controle síncrono, o método analítico e uma quantidade mínima de 1 ponto para cada 6mm de trajetória. os desenhos escolhidos para demonstrar essa capacidade são mostradas na Figura 73. O resultado obtido destas trajetórias realizadas pelo SCARA em comparação com os resultados no MATLAB e V-REP podem ser vistos na Figura 74 e na Figura 75.

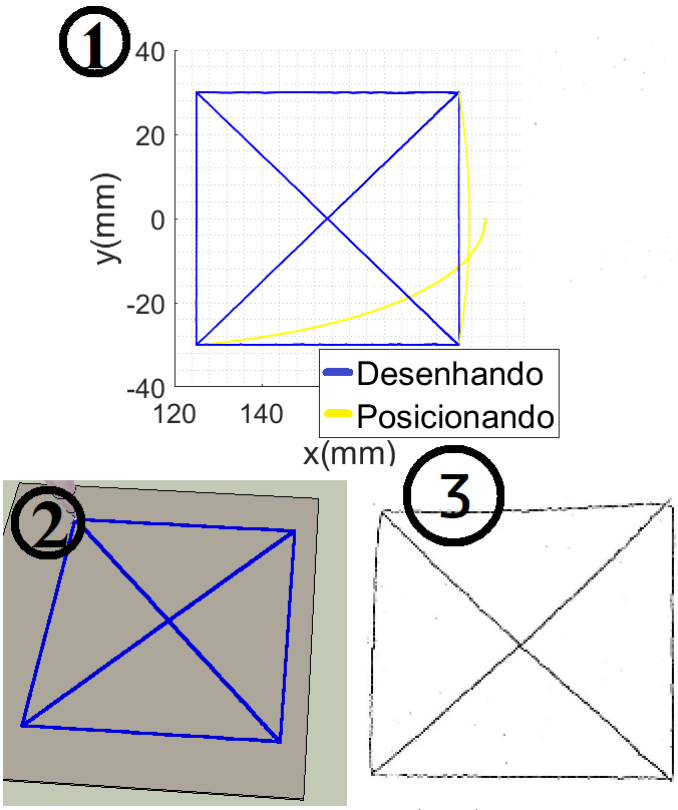
Figura 73 – Desenhos escolhidos para verificar a capacidade do SCARA desenhar um conjunto de retas.



Fonte: O Autor (2019).

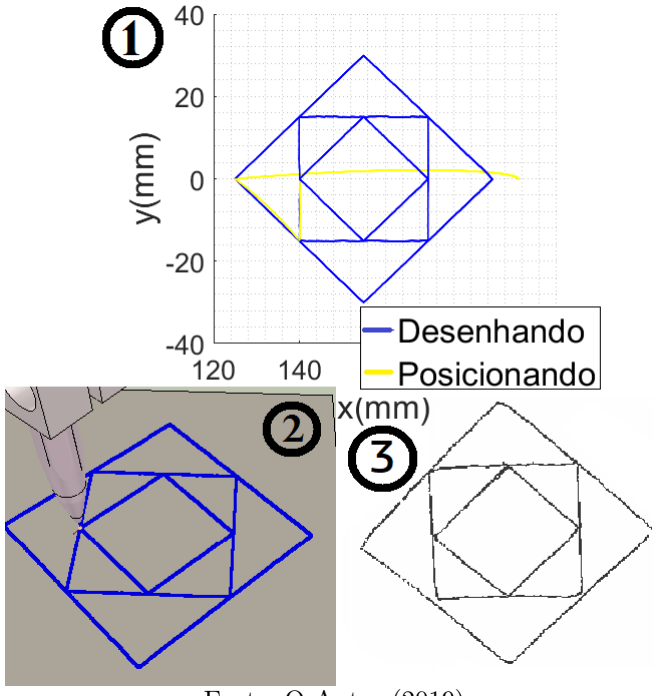
Nas Figuras 74 e 75 pode-se perceber que os resultados deste experimento demonstram a boa capacidade do SCARA em realizar um pequeno conjunto de retas.

Figura 74 – Resultado da primeira simulação da capacidade de realizar um pequeno conjunto de retas.



Fonte: O Autor (2019).
1) MATLAB; 2) V-REP; 3) SCARA.

Figura 75 – Resultado da segunda simulação da capacidade de realizar um pequeno conjunto de retas.

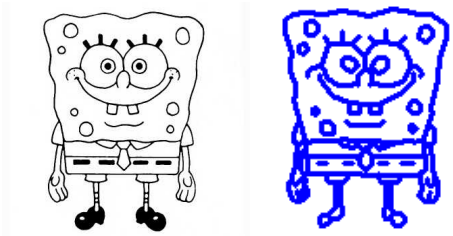


Fonte: O Autor (2019).
1) MATLAB; 2) V-REP; 3) SCARA.

5.3 EXPERIMENTO DE TAREFA COMPLEXA

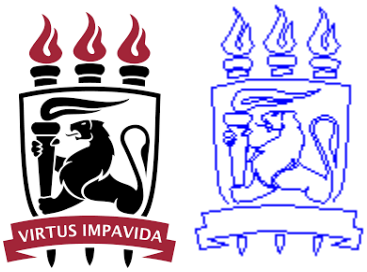
Para realizar o experimento de tarefas complexas foi solicitado que o SCARA realizasse movimentos de tal forma a executar desenhos mais complexos. Os desenhos escolhidos para realização do experimento podem ser vistos nas Figuras 76-79. Para realização da experimento, cada uma destas figuras foi vetorizada e filtrada fazendo assim com que ela fosse composta por um conjunto de pequenas trajetórias que formam o desenho. Cada uma dessas figuras vetorizadas pode ser vista nas Figuras 76-79. Os respectivos resultados obtidos pelo SCARA em comparação com os resultados no MATLAB e V-REP podem ser vistos nas Figuras 80-83. Os experimentos realizados demoram cerca de 3 a 10 minutos, cada um, para serem realizados.

Figura 76 – Desenho do personagem bob esponja e sua figura vetorizada.



Fonte: O Autor (2019).

Figura 77 – Desenho do logo da UFPE e sua figura vetorizada.



Fonte: O Autor (2019).

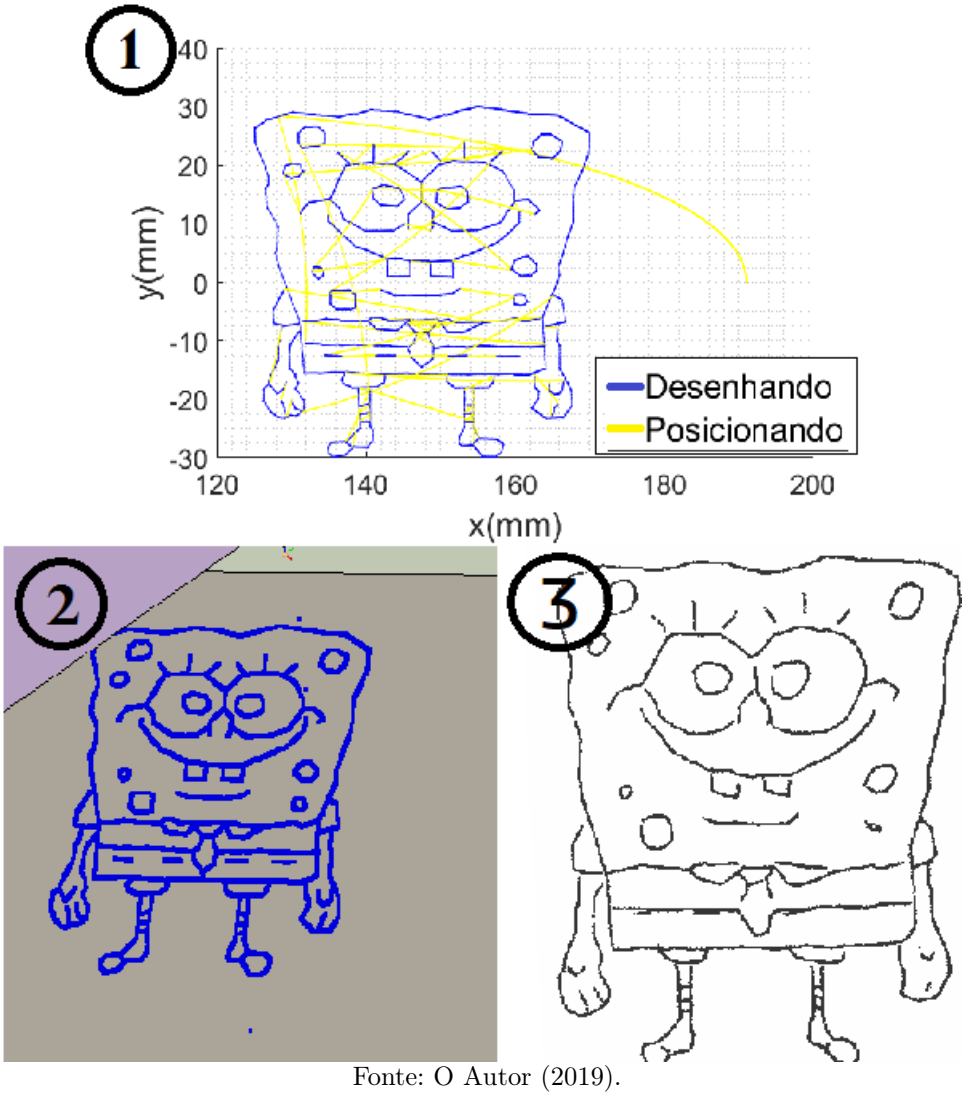
Figura 78 – Desenho da personagem docinho e sua figura vetorizada.

Fonte: O Autor (2019).

Figura 79 – Desenho da personagem florzinha e sua figura vetorizada.

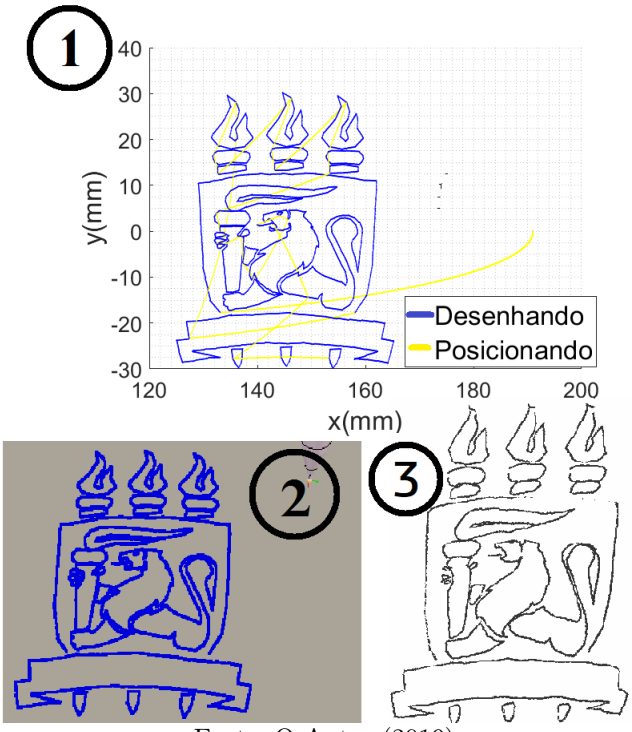
Fonte: O Autor (2019).

Figura 80 – Resultados da simulação em desenhar o personagem bob esponja.



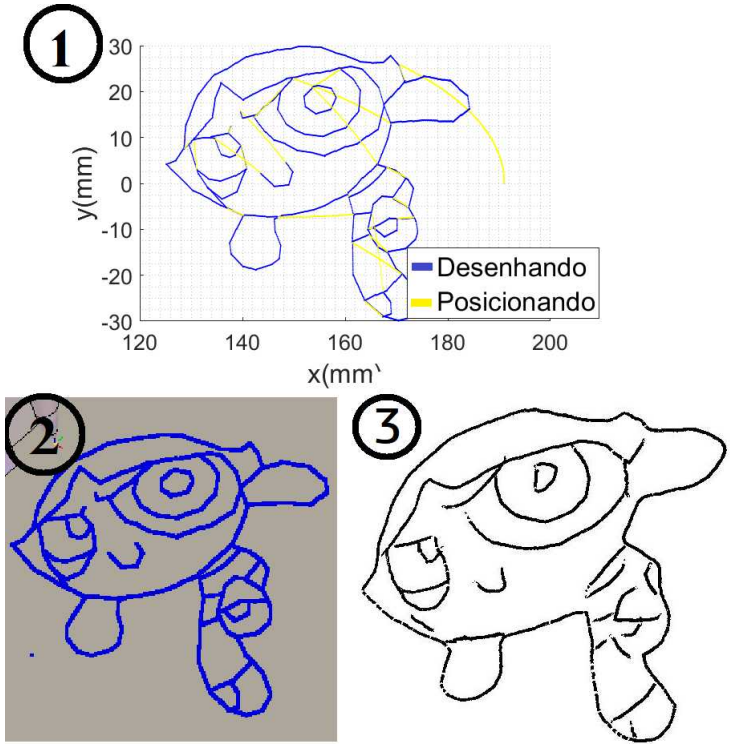
Fonte: O Autor (2019).
1) MATLAB; 2) V-REP; 3) SCARA.

Figura 81 – Resultados da simulação em desenhar o logotipo da UFPE



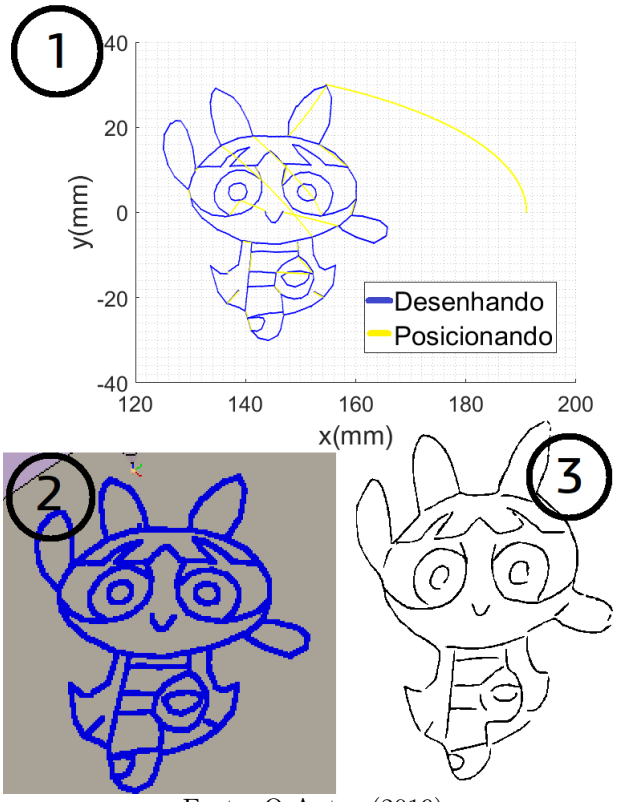
Fonte: O Autor (2019).
1) MATLAB; 2) V-REP; 3) SCARA.

Figura 82 – Resultados da simulação em desenhar a personagem docinho.



Fonte: O Autor (2019).
1) MATLAB; 2) V-REP; 3) SCARA.

Figura 83 – Resultados da simulação em desenhar a personagem florzinha.



Fonte: O Autor (2019).
1) MATLAB; 2) V-REP; 3) SCARA.

Pode-se verificar que o protótipo foi capaz de realizar os diversos experimentos com uma precisão adequada. Estes resultados demonstram uma grande similaridade com as simulações realizadas pelo software e ambientes virtuais na Seção 3.6.

5.4 CONCLUSÃO

Analisando o experimento de posicionamento percebe-se que o protótipo tem total capacidade de executar posicionamentos com exatidão e precisão. Os erros encontrados nesta simulação de posicionamento demonstra que, mesmo o sistema trabalhando em malha aberta, este realiza seus movimentos sem perder passos e com repetibilidade. Com relação ao experimento de trajetórias específicas, percebe-se que o SCARA realizou as trajetórias com precisão condizente com as encontradas nas simulações (precisão da ordem de décimos de milimetro). Estes resultados demonstram que o braço robótico é capaz de assumir tarefas mais complexas.

Os resultados dos experimentos de tarefa complexa demostram grande similaridade com as encontradas na simulação e reafirmam que o SCARA desenvolvido tem uma precisão

da ordem de décimos de milimetro. Certamente, este braço robótico poderia ser usado em diversas tarefas industriais, pois a precisão e exatidão que apresenta é mais do que adequada para a maioria das tarefas industrias (precisões de ordem milimétrica). Assim ficou demonstrado que o desenvolvimento das diversas etapas presentes neste trabalho foi adequado para desenvolver um protótipo com alta precisão.

6 CONCLUSÃO

Durante todo o trabalho, foi desenvolvido um braço robótico SCARA, abrangendo todas as etapas de projetos estrutural, simulação de estresse mecânico, desenvolvimento e simulação do posicionamento do robô, virtualização do braço robótico, construção do protótipo, criação de uma interface de controle e acionamento do protótipo. Durante todas essas etapas, foi verificado que cada uma delas tem uma grande importância para o sucesso do funcionamento do protótipo do braço SCARA, fazendo-o realizar as mais diversas tarefas complexas. Com base nos resultados obtidos, fica claro que o projeto foi um sucesso, visto que executou várias tarefas complexas com exatidão e precisão.

Sabendo da existência da escassez de pesquisas acadêmicas que abordem braços robóticos SCARA no Brasil, este trabalho foi realizado com o objetivo de aumentar a massa crítica de produções na área e motivar pessoas a contribuir com novas pesquisas na área. Este trabalho tem, como ponto especial, o uso de ferramentas de simulação virtual no V-REP e simulações estruturais no SolidWorks que contribuem para sustentar cada um dos resultados obtidos. Outra característica relevante é a existência de uma metodologia didática, detalhada e precisa para o desenvolvimento de cada uma das etapas da cinemática do SCARA (desde aplicação de cinemática direta, inversa e elaboração de trajetórias complexas). Por fim, elaborou-se um protótipo com alta precisão no qual foi verificada a importância da cada uma das etapas do projeto.

Durante o projeto e construção do braço SCARA, foram notados alguns problemas que poderiam ser resolvidos para obter um melhor desempenho do braço em executar tarefas específicas. Um dos problemas encontrados foi na utilização de um número grande de micro-passos (1/128), tornando o braço lento, visto que para executar um movimento equivalente a um passo inteiro sem redução, ele deveria executar 128 micro-passos. Uma possível solução para este problema seria usar drivers de micro-passo que suportam executar mais micro-passos por segundo. Outro possível problema é na utilização de reduções elétricas com drivers de micro-passo, pois não oferecem o benefício do aumento de torque. Este torque mais elevado pode ser necessário em tarefas que exijam mais da estrutura mecânica. Uma possível solução para esse problema seria o uso de reduções mecânicas como engrenagens e polias, que garantiriam uma elevação do torque durante a execução dos micro-passos.

Além do protótipo desenvolvido ser utilizado na elaboração dessa dissertação de mestrado, o braço SCARA e a pequisa desenvolvida será utilizada nos laboratórios do curso de graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco para auxiliar nas aulas e futuros projetos.

6.1 TRABALHOS FUTUROS

Diante do trabalho acadêmico desenvolvido, algumas ideias que podem ser desenvolvidas e contribuir para a evolução da pesquisa de braços SCARA são:

- Uso do protótipo para aplicações industriais, tais como manipular elementos, soldagem de componentes, pintura, etc;
- Verificar e buscar soluções para possíveis limitações de movimentação caso ocorra sobrecarga de peso no efetuador;
- Desenvolvimento de três novos graus de liberdade no efetuador (tornando-o um SCARA de 6 graus de liberdade);
- Desenvolvimento de engrenagens ou polias para o uso de reduções mecânicas para gerar aumento de torque e de precisão;
- Implementar um algoritmo de otimização que busque a menor trajetória incluindo restrições de movimento, como obstáculos, limites de torque e esforços nas juntas, etc;
- Desenvolvimento de um controle em malha fechada para evitar perdas de passo em tarefas que necessitem de altas velocidades;
- Utilização de câmeras para verificar, em tempo real, o espaço de trabalho e utilizar essas informações para definir trajetórias e tarefas mais complexas;
- Realizar desenhos com preenchimento.

REFERÊNCIAS

BANERJEE; ANANYA. Robotics in indian industry. **International Journal of Engineering Research and Development**, V6, 01 2017.

BEHNEN, V. L. **Modelagem matemática de um robô SCARA com acionamento pneumático e identificação experimental das principais não linearidades**. Dissertação (Mestrado) — Universidade Regional Do Noroeste Do Estado Do Rio Grande Do Sul, 2017.

BIANCHI, R. **Robótica**. [S.l.]: Centro Universitário FEI, 2016.

ENGINE castle. **Castle Game Engine**. 2019. Disponível em: <https://www.castle-engine.io>. Acesso em: 5 de agosto de 2018.

FILHO, J. R. H. A era da internet industrial e a indústria 4.0. **Producão em Foco**, v. 6, n. 3, 2016.

KUKA. **Robôs industriais da KUKA**. 2019. Disponível em: <https://www.kuka.com/pt-br/produtos-servi%C3%A7os/sistemas-de-rob%C3%B4/rob%C3%B4s-industriais>. Acesso em: 5 de agosto de 2018.

LAZARUS. **Lazarus**. 2019. Disponível em: <https://www.lazarus-ide.org/>. Acesso em: 5 de agosto de 2018.

MELLO, O. P. M. de. **Desenvolvimento de um robô manipulador SCARA**. 2016. Monografia (Engenheiro de Controle e Automação), UNB (Universidade de Brasília), DF, Brasil.

MOUSAVI, A. *et al.* Repeatability analysis of a scara robot with planetary gearbox. **IEEE**, p. 640–644, Oct 2015.

NIKU, S. benjamin. **Introdução a Robótica: Analise, Controle e Aplicações**. 2.ed. ed. [S.l.]: LTC, 2015.

PADOIN, E. **Modelagem matemática dinâmica da não linearidade de folga em uma junta rotativa de um robô SCARA com transmissão por engrenagens**. Dissertação (Mestrado) — Universidade Regional do Noroeste do Estado do Rio Grande do Sul, 2011.

PEREIRA, N.; SILVA, J. D.; THÉ, G. Planejamento de caminho livre de colisão para robô scara em ambiente monitorado por sensor de imagem. **Anais do XX Congresso Brasileiro de Automática**, 2014.

PIMENTA, K. B. **Controle de juntas robóticas usando controlador preditivo generalizado adaptativo direto**. Tese (Doutorado) — Universidade Estadual De Campinas, 3 2003.

PRADHAN, S.; RAJARAJAN, K.; SHETTY, A. S. Prototype, emulation, implementation and evaluation of scara robot in industrial environment. **Procedia Computer Science**, v. 133, p. 331 – 337, 2018. ISSN 1877-0509. International Conference on Robotics and Smart Manufacturing (RoSMa2018). Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050918309864>.

SANTOS, R. T. *et al.* Manipulador robótico scara na indústria:estudo de caso em sistema de paletização de fármacos. **Semana Acadêmica Carlos Drummond**, p. 17, 2016.

SCHLUTER, M. dos S. **Controle não linear adaptativo com compensação de atrito de um manipulador SCARA com acionamento pneumático**. Tese (Doutorado) — Universidade Federal Do Rio Grande Do Sul, 2018.

SILVA, E. C. N. **Métodos numéricos aplicados em problemas de otimização de engenharia**. [S.l.]: Universidade De São Paulo, 2015.

SOLIDWORKS. **Dassault Systemes**. 2018. Disponível em: <https://www.solidworks.com/>. Acesso em: 5 de agosto de 2018.

TEKKNO. **Robo Orbital 4 Articulações**. 2018. Disponível em: <https://www.tekkno.com.br/produto/3778/robo-orbital-4-articulacoes>. Acesso em: 5 de agosto de 2018.

URREA, C.; CORTÉS, J.; PASCAL, J. Design, construction and control of a scara manipulator with 6 degrees of freedom. **Journal of Applied Research and Technology**, v. 14, n. 6, p. 396 – 404, 2016. ISSN 1665-6423. Disponível em: <http://www.sciencedirect.com/science/article/pii/S166564231630093>.

V-REP. **Coppelia Robotics**. 2018. Disponível em: <http://www.coppeliarobotics.com/>. Acesso em: 5 de agosto de 2018.

VARGAS, R. M. **Desenvolvimento de uma plataforma de testes de controladores para os dois primeiros graus de liberdade de um robô SCARA**. 2015.

WOTIOM. **Stepper Motor model: WS17-0035-04-4**. 2019. Disponível em: <http://www.wotiom.com/wp-content/uploads/2018/09/datasheet-motor-WS17-0035-04-4-gr%C3%A1fico.pdf>. Acesso em: 5 de agosto de 2018.

YAMAHA. **SCARA**. 2019. Disponível em: <https://global.yamaha-motor.com/business/robot/lineup/application/ykxg/>. Acesso em: 5 de agosto de 2018.

APÊNDICE A – EQUAÇÃO ANALÍTICA DO SCARA

Inicialmente temos a equação (2):

$$\begin{cases} 0 = L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) - x_o \\ 0 = L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) - y_o \\ 0 = Z - z_o \end{cases} \tag{2}$$

A partir das duas primeiras equações em (2) e de pequenas manipulações algébricas temos:

$$\begin{cases} (x_o - L_1\cos(\theta_1))^2 = (L_2\cos(\theta_1+\theta_2))^2 \\ (y_o - L_1\sin(\theta_1))^2 = (L_2\sin(\theta_1+\theta_2))^2 \end{cases} \tag{A.1}$$

Somando as equações em (A.1):

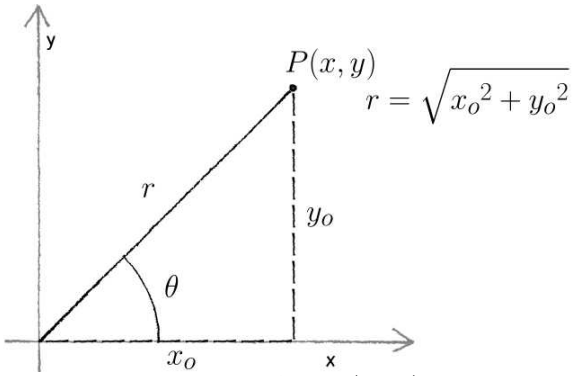
$${x_o}^2 + {y_o}^2 + {L_1}^2 - 2L_1(x_o\cos(\theta_1) + y_o\sin(\theta_1)) = {L_2}^2(\cos(\theta_1+\theta_2)^2 + \sin(\theta_1+\theta_2)^2) \tag{A.2}$$

Com pequenas manipulações algébricas em (A.2) tem-se:

$$\frac{{x_o}^2 + {y_o}^2 + {L_1}^2 - {L_2}^2}{2L_1\sqrt{{x_o}^2 + {y_o}^2}} = \frac{(x_o\cos(\theta_1) + y_o\sin(\theta_1))}{\sqrt{{x_o}^2 + {y_o}^2}} \tag{A.3}$$

Baseado na Figura A, que representa um triangulo retângulo formado pelo ponto P(x,y) (posição final do efetuador), temos:

Figura A – Triangulo retângulo formado pelo ponto(x,y).



Fonte: O Autor (2019).

$$\begin{cases} \theta = \tan^{-1}(\dfrac{y_o}{x_o}) \\ \dfrac{x_o}{\sqrt{{x_o}^2 + {y_o}^2}} = \cos(\theta) \\ \dfrac{y_o}{\sqrt{{x_o}^2 + {y_o}^2}} = \sin(\theta) \\ \cos(\theta_a)\cos(\theta_b) + \sin(\theta_a)\sin(\theta_b) = \cos(\theta_a - \theta_b) \end{cases} \tag{A.4}$$

Assim, usando (A.3) e as relações em (A.4) tem-se:

$$\frac{{x_o}^2 + {y_o}^2 + {L_1}^2 - {L_2}^2}{2L_1\sqrt{{x_o}^2 + {y_o}^2}} = \cos(\theta_1 - \tan^{-1}(\frac{y_o}{x_o})) \tag{A.5}$$

Fazendo pequenas manipulações algébricas em (A.5) para isolar θ_1 temos:

$$\theta_1 = \cos^{-1}(\frac{{x_o}^2 + {y_o}^2 + {L_1}^2 - {L_2}^2}{2L_1\sqrt{{x_o}^2 + {y_o}^2}}) + \tan^{-1}(\frac{y_o}{x_o}) \tag{A.6}$$

Para encontrar o valor de θ_2, deve-se elevar ao quadrado as duas primeiras equações em (2), obtendo:

$$\begin{cases} ({x_o}^2 = (L_2\cos(\theta_1 + \theta_2))^2 + (L_1\cos(\theta_1)^2 + 2(L_1L_2\cos(\theta_1 + \theta_2)\cos(\theta_1) \\ ({y_o}^2 = (L_2\sin(\theta_1 + \theta_2))^2 + (L_1\sin(\theta_1)^2 + 2(L_1L_2\sin(\theta_1 + \theta_2)\sin(\theta_1) \end{cases} \tag{A.7}$$

Depois, deve ser somadas as equações em (A.7) e usar a relação da quarta equação em (A.4) para simplificar, obtendo:

$${x_o}^2 + {y_o}^2 = {L_1}^2 + {L_2}^2 + L_1L_2\cos(\theta_2) \tag{A.8}$$

Isolando θ_2 em (A.9) tem-se:

$$\theta_2 = \cos^{-1}\left(\frac{x_o^2 + y_o^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \tag{A.9}$$

Assim, juntando (A.6), (A.9) e a última equação em (2), com uma pequena modificação, temos (7):

$$\begin{cases} \theta_1 = \cos^{-1}\left(\frac{x_o^2+y_o^2+L_1^2-L_2^2}{2L_1\sqrt{x_o^2+y_o^2}}\right) + \tan^{-1}\left(\frac{y_o}{x_o}\right) \\ \theta_2 = \cos^{-1}\left(\frac{x_o^2+y_o^2-L_1^2-L_2^2}{2L_1L_2}\right) \\ Z = z_o \end{cases} \tag{7}$$

APÊNDICE B – SIMULAÇÕES DE MÉTODOS MATEMÁTICOS

Neste Apêndice serão apresentados os códigos realizados em MATLAB que foram capazes de verificar e simular o comportamento dos métodos matemáticos.

Programa MATLAB para o Método Numérico

```
function [E,R,RR]=braco_scara_numerico(X1,Y1,X0,precisao)
x(:,1)=X0;
A1=86;
A2=105; N=50; Erro=zeros(2,N);
P_F=[X1 Y1];
for i=1:N
F(1,1)=A2*cos(x(1,i)+x(2,i))+A1*cos(x(1,i))-X1;
F(2,1)=A2*sin(x(1,i)+x(2,i))+A1*sin(x(1,i))-Y1;
dF(1,1)=-A2*sin(x(1,i)+x(2,i))-A1*sin(x(1,i));
dF(1,2)=-A2*sin(x(1,i)+x(2,i));
dF(2,1)=A2*cos(x(1,i)+x(2,i))+A1*cos(x(1,i));
dF(2,2)=A2*cos(x(1,i)+x(2,i));
Erro(:,i)=F;
%corecao de angulos
x(:,i+1)=x(:,i)-(dF\F);
while x(1,i+1)>pi
x(1,i+1)=x(1,i+1)-pi;
end
while x(1,i+1)<-pi
x(1,i+1)=x(1,i+1)+pi;
end
while x(2,i+1)>pi
x(2,i+1)=x(2,i+1)-pi;
end
while x(2,i+1)<-pi
x(2,i+1)=x(2,i+1)+pi;
end
```

```
end
R(1)=x(1,N+1);
R(2)=x(2,N+1);
R=R*180/pi;
E=Erro(:,N);
delta=1.8/precisao;
delta_T=zeros(4,2);
delta_T(1,:)=[0 0];
delta_T(2,:)=[delta 0];
delta_T(3,:)=[0 delta];
delta_T(4,:)=[delta delta];
%%Processamento da melhor resposta para uma precisao especifica...
RR=zeros(1,2);
RR(1)=floor(R(1)/((1.8/128)))*1.8/128
RR(2)=floor(R(2)/((1.8/128)))*1.8/128
MIN=norm(braco_scara_direto_degree(((RR+delta_T(1,:))-P_F)));
minimo=1;
for i=2:4
Erro=norm(braco_scara_direto_degree(((RR+delta_T(i,:))-P_F)));
if (MIN>Erro)
MIN=Erro;
minimo=i;
end
end
RR=RR+delta_T(minimo,:);
end
```

Programa MATLAB para o Método Analítico

```
function [R,RR]=braco_scara_analitico(X1,Y1,X0,precisao)
A1=86;
A2=105;
P_F=[X1 Y1];
P=X1^2+Y1^2;
```

```
6  P1=atan2(Y1,X1);
7  P2=(P+A1^2-A2^2)/(2*A1*sqrt(P));
8  P3=acos(P2);
9  P4=(P-A1^2+A2^2)/(2*A2*sqrt(P));
10 P5=acos(P4);
11 %
12 x_1(1)=P3+P1;
13 x_1(2)=-P3+P1;
14 x_2(1)=P5+P1;
15 x_2(2)=-P5+P1;
16 %
17 X(1,:)=[x_1(1) x_2(1)-x_1(1)];
18 X(2,:)=[x_1(1) x_2(2)-x_1(1)];
19 X(3,:)=[x_1(2) x_2(1)-x_1(2)];
20 X(4,:)=[x_1(2) x_2(2)-x_1(2)];
21 %
22 braco_scara_direto_rad(X(1,:));
23 braco_scara_direto_rad(X(2,:));
24 braco_scara_direto_rad(X(3,:));
25 braco_scara_direto_rad(X(4,:));
26 MIN=1;
27 X0=X0*pi/180;
28 F_MIN=norm(X(1,:)-X0)+100*norm(braco_scara_direto_rad(X(1,:))-P_F);
29 for i=2:4
30 if (F_MIN>norm(X(i,:)-X0)+100000*norm(braco_scara_direto_rad(X(i,:))-P_F)
      )
31 F_MIN=norm(X(i,:)-X0);
32 MIN=i;
33 end
34 end
35 R=X(MIN,:);
36 R=R*180/pi;
```

```
delta=1.8/precisao;
delta_T=zeros(4,2);
delta_T(1,:)=[0 0];
delta_T(2,:)=[delta 0];
delta_T(3,:)=[0 delta];
delta_T(4,:)=[delta delta];
%
RR=zeros(1,2);
RR(1)=floor(R(1)/((delta)))*delta;
RR(2)=floor(R(2)/((delta)))*delta;
MIN=norm(braco_scara_direto_degree(((RR+delta_T(1,:))-P_F)));
minimo=1;
for i=2:4
Erro=norm(braco_scara_direto_degree(((RR+delta_T(i,:))-P_F)));
if (MIN>Erro)
MIN=Erro;
minimo=i;
end
end
RR=RR+delta_T(minimo,:);
end
```

Programa MATLAB para o Método Preditivo

```
function [E,R,HISTORICO,HISTORICO_A]=braco_scara_preditivo(X1,Y1,X0,delta
   ,A1,A2)
P=X0;P_F=[X1 Y1];n_chegou=true;
delta_T(1,:)=[0 0]';delta_T(2,:)=[0  delta]';
delta_T(3,:)=[0  -delta]';delta_T(4,:)=[delta  0]';
delta_T(5,:)=[delta  delta]';delta_T(6,:)=[delta  -delta]';
delta_T(7,:)=[-delta  0]';delta_T(8,:)=[-delta  delta]';
delta_T(9,:)=[-delta  -delta]';
k=1;Pf(1,:)=braco_scara_direto_rad1(P,A1,A2);
MIN=norm(Pf(1,:)-P_F);
```

```
HISTORICO=[];
HISTORICO_A=[];
while n_chegou
minimo=1;
for i=2:9
Erro=norm(braco_scara_direto_rad1(P+delta_T(i,:),A1,A2)-P_F);
if (MIN>Erro)
MIN=Erro;
minimo=i;
end
end
P=P+delta_T(minimo,:);
Pf(1,:)=braco_scara_direto_rad1(P,A1,A2);
ERRO=(norm(Pf(1,:)-P_F));
if (minimo==1)
n_chegou=false;
else
%
HISTORICO(k,:)=Pf(1,:);
HISTORICO_A(k,:)=P';
k=k+1;
end
%
%
end
R=P;
E=ERRO;
end
```

APÊNDICE C – PROGRAMAÇÃO DO PIC16F887

```
1   program Aciona_passo_rs232_887_TB_GEPAE;
2   implementation
3   var rec, oldstate:byte;
4   local:boolean;
5   pos:word;
6   begin
7   TRISB:=255;
8   TRISC:=%10000000;
9   TRISE:=%111;
10  TRISA:=%00001100;
11  TRISD:=0;
12  PORTD:=0;
13  PORTB:=0;
14  PORTA:=0;
15  //ANSEL:=%00000000;
16  //ANSELH:=0;
17  local:=false;
18  delay_ms(500);
19  oldstate := 0;
20  UART1_Init(115200);
21  //Usart_Init(57600);
22  while true do
23  begin
24  {   if Button(PORTA, 3, 20, 1) then oldstate := 255;
25  if oldstate and Button(PORTA, 3, 20, 0) then
26  begin
27  local:=not local;
28  oldstate := 0;
29  end; }
```

```
30                  while UART1_Data_Ready() = 0 do rec :=0;
31                  rec := UART1_Read;
32                  if rec=250 then PORTC.1:=0;
33                  if rec=251 then PORTC.1:=1;
34                  if rec=255 then PORTB:=0;
35                  if rec=100 then
36                  if PORTA.0=1 then PORTA.0:=0 else PORTA.0:=1;
37                  if rec=150 then
38                  if PORTA.1=1 then PORTA.1:=0 else PORTA.1:=1;
39                  if rec=203 then
40                  if PORTA.4=1 then PORTA.4:=0 else PORTA.4:=1;
41                  if rec=101 then
42                  begin
43                  if PORTB.5=0 then UART1_write(0) else UART1_write(1);
44                  end;
45                  if rec>63 then rec:=0;
46                  if (rec>0) and (rec<63) then
47                  begin
48                  PORTD:=rec;
49                  delay_us(100);
50                  PORTD.0:=0;
51                  PORTD.2:=0;
52                  PORTD.4:=0;
53                  //Poderia dar apenas um delay...
54                  end;
55                  delay_us(1000);
56                  end;
57                  end.
```

APÊNDICE D – PROGRAMAÇÃO DO IHM

```
1  {
2  Copyright 2008-2017 Michalis Kamburelis.
3
4  This file is part of Castle Game Engine.
5
6  Castle Game Engine is free software; see the file COPYING.txt,
   included in this distribution, for details about the copyright.
8
9  Castle Game Engine is distributed in the hope that it will be useful,
10 but WITHOUT ANY WARRANTY; without even the implied warranty of
11 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
12
13 ----------------------------------------------------------------------
14 }
15
16 { Main form. }
17 unit mainf;
18
19 {$mode objfpc}{$H+}
20
21 interface
22
23 uses Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs,
```

```
24                OpenGLContext, Menus, CastleScene, CastleCameras,
                     CastleControl, CastleLog,
25                CastleLCLRecentFiles, CastleConfig, Buttons, ExtCtrls,
                     StdCtrls, ComCtrls,
26                ActnList, Arrow, LazSerial, CastleRecentFiles,
                     CastleSceneManager,
27                CastleDialogs, CastleControls, Math,CastleJoysticks, Types
                     ;
28                type
29
30                angulos = record
31                teta1: real;
32                teta2: real;
33                end;
34
35                TEventsHandler = class
36                procedure JoyAxisMove(const Joy: PJoy; const Axis: Byte;
                     const Value: Single);
37                end;
38
39
40                type
41
42                { TMain }
43
44                TMain = class(TForm)
45                ApplicationProperties1: TApplicationProperties;
46                Arrow1: TArrow;
47                Arrow4: TArrow;
48                Arrow5: TArrow;
49                Arrow6: TArrow;
50                Arrow7: TArrow;
```

```
51                  Arrow8: TArrow;
52                  Browser: TCastleControl;
53                  Button1: TButton;
54                  Button2: TButton;
55                  Button3: TButton;
56                  Button4: TButton;
57                  Button5: TButton;
58                  Button6: TButton;
59                  Button7: TButton;
60                  Button8: TButton;
61                  Button9: TButton;
62                  Edit1: TEdit;
63                  Edit10: TEdit;
64                  Edit11: TEdit;
65                  Edit12: TEdit;
66                  Edit13: TEdit;
67                  Edit14: TEdit;
68                  Edit15: TEdit;
69                  Edit16: TEdit;
70                  Edit17: TEdit;
71                  Edit2: TEdit;
72                  Edit3: TEdit;
73                  Edit4: TEdit;
74                  Edit5: TEdit;
75                  Edit6: TEdit;
76                  Edit7: TEdit;
77                  Edit8: TEdit;
78                  Edit9: TEdit;
79                  EditDirectionX1: TEdit;
80                  EditDirectionY1: TEdit;
81                  EditPositionX1: TEdit;
82                  EditPositionY1: TEdit;
```

```
83                EditPositionZ1: TEdit;
84                EditUpX: TEdit;
85                EditUpY: TEdit;
86                EditUpZ: TEdit;
87                GroupBox1: TGroupBox;
88                GroupBox2: TGroupBox;
89                Image1: TImage;
90                Label1: TLabel;
91                Label10: TLabel;
92                Label11: TLabel;
93                Label12: TLabel;
94                Label13: TLabel;
95                Label14: TLabel;
96                Label15: TLabel;
97                Label16: TLabel;
98                Label17: TLabel;
99                Label18: TLabel;
100               Label19: TLabel;
101               Label2: TLabel;
102               Label20: TLabel;
103               Label21: TLabel;
104               Label22: TLabel;
105               Label23: TLabel;
106               Label24: TLabel;
107               Label25: TLabel;
108               Label26: TLabel;
109               Label27: TLabel;
110               Label3: TLabel;
111               Label4: TLabel;
112               Label5: TLabel;
113               Label6: TLabel;
114               Label7: TLabel;
```

```
115                 Label8: TLabel;
116                 Label9: TLabel;
117                 LabelDirection1: TLabel;
118                 LabelPosition1: TLabel;
119                 LazSerial1: TLazSerial;
120                 ListBox2: TListBox;
121                 ListBox3: TListBox;
122                 ListBox4: TListBox;
123                 MainMenu1: TMainMenu;
124                 Arquivos: TMenuItem;
125                 MainMenu2: TMainMenu;
126                 MenuHelp: TMenuItem;
127                 MenuAboutOpenGL: TMenuItem;
128                 MenuItem1: TMenuItem;
129                 MenuItem10: TMenuItem;
130                 MenuItem11: TMenuItem;
131                 MenuItem12: TMenuItem;
132                 MenuItem13: TMenuItem;
133                 MenuItem14: TMenuItem;
134                 MenuItem2: TMenuItem;
135                 MenuItem3: TMenuItem;
136                 MenuItem4: TMenuItem;
137                 MenuItem5: TMenuItem;
138                 MenuItem6: TMenuItem;
139                 MenuItem7: TMenuItem;
140                 MenuItem8: TMenuItem;
141                 MenuItem9: TMenuItem;
142                 MenuWebsite: TMenuItem;
143                 MenuSep2: TMenuItem;
144                 OpenDialog1: TCastleOpen3DDialog;
145                 OpenDialog2: TOpenDialog;
146                 Panel1: TPanel;
```

```
147                 Panel2: TPanel;
148                 ProgressBar1: TProgressBar;
149                 Timer1: TTimer;
150                 Timer2: TTimer;
151                 Timer3: TTimer;
152                 ToggleBox1: TToggleBox;
153                 TrackBar1: TTrackBar;
154                 TrackBar2: TTrackBar;
155                 TrackBar3: TTrackBar;
156                 TrackBar4: TTrackBar;
157                 TrackBar5: TTrackBar;
158                 TrackBar6: TTrackBar;
159                 procedure Button1Click(Sender: TObject);
160                 procedure Button2Click(Sender: TObject);
161                 procedure Button3Click(Sender: TObject);
162                 procedure Button4Click(Sender: TObject);
163                 procedure Button5Click(Sender: TObject);
164                 procedure Button6Click(Sender: TObject);
165                 procedure Button7Click(Sender: TObject);
166                 procedure Button9Click(Sender: TObject);
167                 procedure ButtonNavigationTypeClick(Sender: TObject);
168                 procedure BrowserCameraChanged(Camera: TCamera);
169                 procedure FormCreate(Sender: TObject);
170                 procedure MenuAboutOpenGLClick(Sender: TObject);
171                 procedure ArquivosClick(Sender: TObject);
172                 procedure MenuFocusGLControlClick(Sender: TObject);
173                 procedure CarregarClick(Sender: TObject);
174                 procedure MenuItem10Click(Sender: TObject);
175                 procedure MenuItem11Click(Sender: TObject);
176                 procedure MenuItem12Click(Sender: TObject);
177                 procedure MenuItem13Click(Sender: TObject);
178                 procedure MenuItem14Click(Sender: TObject);
```

```
179                 procedure MenuItem1Click(Sender: TObject);
180                 procedure MenuItem2Click(Sender: TObject);
181                 procedure MenuItem3Click(Sender: TObject);
182                 procedure MenuItem4Click(Sender: TObject);
183                 procedure MenuItem8Click(Sender: TObject);
184                 procedure MenuItem9Click(Sender: TObject);
185                 procedure MenuQuitClick(Sender: TObject);
186                 procedure MenuShowConsoleClick(Sender: TObject);
187                 procedure MenuWebsiteClick(Sender: TObject);
188                 procedure ProgressBar1ContextPopup(Sender: TObject;
                       MousePos: TPoint;
189                 var Handled: Boolean);
190                 procedure RadioButton1Change(Sender: TObject);
191                 procedure Timer1Timer(Sender: TObject);
192                 procedure MenuMouseLookToggleClick(Sender: TObject);
193                 procedure FormDestroy(Sender: TObject);
194                 procedure RecentFilesOpenRecent(const URL: string);
195                 procedure Timer2Timer(Sender: TObject);
196                 procedure Timer3Timer(Sender: TObject);
197                 procedure ToggleBox1Change(Sender: TObject);
198                 procedure TrackBar4Change(Sender: TObject);
199                 procedure envia(dado:byte);
200                 private
201                 SceneURL: string;
202                 CameraChanged: boolean;
203                 ButtonsNavigationType: array [TNavigationType] of
                       TSpeedButton;
204                 CrosshairCtl: TCastleCrosshair;
205                 CrosshairActive: Boolean;    // there is something to
                       touch under the crosshair
206                 procedure OpenScene(const URL: string);
207                 procedure UpdateCaption;
```

```
208                 procedure UpdateCrosshairImage;
209                 procedure OnPointingDeviceSensorsChange(Sender: TObject);
210                 public
211                 { public declarations }
212                 end;
213
214                 var
215                 Main: TMain;
216
217                 var
218                 Serial: boolean;
219                 Direta: boolean;
220                 chegou: boolean;
221                 desenhando:boolean;
222                 verificou: boolean;
223                 valor0: real;
224                 eixo0: integer;
225                 valor1: real;
226                 eixo1: integer;
227                 valor2: real;
228                 eixo2: integer;
229                 valor3: real;
230                 eixo3: integer;
231                 posicao:integer=0;
232                 j:longint=0;
233                 teta1_desejado:real;
234                 teta2_desejado:real;
235                 z_desejado:real;
236                 teta1_real:real;
237                 teta2_real:real;
238                 z_real:real;
239                 bmp:Tbitmap;
```

```
240
241             implementation
242
243             uses LCLType, LCLIntf, CastleVectors, CastleBoxes,
                   X3DNodes, CastleRenderer,
244             CastleClassUtils, CastleUtils, X3DLoad, CastleURIUtils,
245             CastleGLUtils, CastleSceneCore, CastleFilesUtils,
                   CastleParameters,
246             CastleApplicationProperties, CastleTransform,
247             OpenGLInformation, CastleLCLUtils, ConsoleF, CastleImages,
                    CastleSoundEngine;
248
249             var
250             braco0,braco1,braco2, braco_fixo: TCastleScene;
251             bracotrans0,bracotrans,bracotrans1: TCastleTransform;
252             pontos: array [0..15000] of TCastleScene;
253             pontost: array [0..15000] of TCastleTransform;
254             procedure TMain.OpenScene(const URL: string);
255             begin
256
257             end;
258
259             procedure TMain.CarregarClick(Sender: TObject);
260             begin
261             end;
262
263             procedure TMain.MenuItem10Click(Sender: TObject);
264             var
265             F: TextFile;
266             s:char;
267             C: string;
268             i :integer;
```

```
269                 zoom: real;
270                 j:integer=1;
271                 x,y:real;
272                 teta1,teta2:real;
273                 begin
274                 MenuItem5.Enabled:=false;
275                 ProgressBar1.Visible:=True;
276                 ProgressBar1.enabled:=True;
277                 label27.Visible:=true;
278                 if OpenDialog2.Execute then
279                 begin
280                 AssignFile(F, OpenDialog2.FileName);
281                 Reset(F);
282                 c:='';
283                 while not eof(F) do
284                 begin
285                 Read(F, S);
286                 if s=',' then begin
287                 if j=1 then Listbox2.Items.Add(c);
288                 if j=2 then Listbox3.Items.Add(c);
289                 if j=3 then Listbox4.Items.Add(c);
290                 j:=j+1;
291                 if j>3 then j:=1;
292                 c:='';
293                 end
294                 else
295                 c:=c+s;
296                 end;
297                 end;
298                 //Canvas tem 600x600
299                 image1.Picture.Bitmap.Canvas.Pen.Width:=1;
300                 image1.Picture.Bitmap.Canvas.Brush.Style:=bsSolid;
```

```
301             image1.Picture.Bitmap.Canvas.fillrect(Rect(0,0,400,400));
302             image1.Picture.Bitmap.Canvas.Pen.Color:=clblue;
303             for i:=0 to listbox3.Items.Count-2 do
304             begin
305             teta1:=strtofloat(listbox3.items[i])*2*9.817477042468104e
                    -04;
306             teta2:=strtofloat(listbox4.items[i])*2*9.817477042468104e
                    -04;
307             x:=86.0*cos(teta1)+105.0*cos(teta1+teta2)-125;
308             y:=86.0*sin(teta1)+105.0*sin(teta1+teta2)+30;
309
310             if strtofloat(listbox2.items[i]) < 0.5 then
311             begin
312             image1.Picture.Bitmap.Canvas.MoveTo(trunc(x*20/3),400-
                    trunc(y*20/3));
313             end;
314             if strtofloat(listbox2.items[i]) > 0.5 then
315             begin
316             image1.Picture.Bitmap.Canvas.LineTo(trunc(x*20/3),400-
                    trunc(y*20/3));
317             end;
318             progressbar1.Position:=trunc((i*1000.0)/listbox3.Items.
                    Count);
319             end;
320             progressbar1.Position:=trunc(1000.0);
321             Button6.Enabled:=true;
322             button6.visible:=true;
323             end;
324
325             procedure TMain.MenuItem11Click(Sender: TObject);
326             begin
327             Serial:=True;
```

```
328                 GroupBox1.Visible:=True;
329                 MenuItem5.Enabled:=False;
330                 Timer3.Enabled:=True;
331                 Timer3.Interval:=1;
332                 end;
333
334                 procedure TMain.MenuItem12Click(Sender: TObject);
335                 begin
336                 Serial:=True;
337                 GroupBox2.Visible:=True;
338                 Timer2.Enabled:=True;
339                 Timer2.Interval:=1;
340                 Direta:=true;
341                 MenuItem5.Enabled:=False;
342                 Joysticks.Free;
343                 EnableJoysticks;
344
345                 Joysticks.OnAxisMove := @TEventsHandler(nil).JoyAxisMove;
346                 end;
347
348                 procedure TMain.MenuItem13Click(Sender: TObject);
349                 begin
350                 Serial:=True;
351                 menuitem10.Click;
352                 end;
353
354                 procedure TMain.MenuItem14Click(Sender: TObject);
355                 begin
356                 Close;
357                 end;
358
359                 procedure TMain.MenuItem1Click(Sender: TObject);
```

```
360                 begin
361
362                 end;
363
364                 procedure TMain.MenuItem2Click(Sender: TObject);
365                 begin
366                 LazSerial1.ShowSetupDialog;
367                 end;
368
369                 procedure TMain.MenuItem3Click(Sender: TObject);
370                 begin
371                 LazSerial1.Open;
372                 MenuItem6.Enabled:=false;
373                 MenuItem7.Enabled:=true;
374                 MenuItem4.Enabled:=true;
375                 button7.Enabled:=true;
376                 button8.Enabled:=true;
377
378                 button7.visible:=true;
379                 button8.visible:=true;
380                 end;
381
382                 procedure TMain.MenuItem4Click(Sender: TObject);
383                 begin
384                 LazSerial1.Close;
385                 close;
386                 end;
387
388                 procedure TMain.MenuItem8Click(Sender: TObject);
389                 begin
390                 GroupBox1.Visible:=True;
391                 MenuItem5.Enabled:=False;
```

```
392                 Timer3.Enabled:=True;
393                 Timer3.Interval:=20;
394                 end;
395
396                 procedure TMain.MenuItem9Click(Sender: TObject);
397                 begin
398                 GroupBox2.Visible:=True;
399                 Timer2.Enabled:=True;
400                 Timer2.Interval:=20;
401                 Direta:=true;
402                 MenuItem5.Enabled:=False;
403                 Joysticks.Free;
404                 EnableJoysticks;
405
406                 Joysticks.OnAxisMove := @TEventsHandler(nil).JoyAxisMove;
407                 end;
408
409                 procedure TMain.UpdateCaption;
410                 var
411                 S: string;
412                 begin
413                 if SceneURL <> '' then
414                 S := URICaption(SceneURL) else
415                 S := 'No Scene';
416                 S := S + ' — ' +  ApplicationName + ' — FPS: ' + Browser.
                       Fps.ToString;
417                 Caption := S;
418                 end;
419
420                 procedure TMain.MenuQuitClick(Sender: TObject);
421                 begin
422
```

```
423                 end;
424
425                 procedure TMain.MenuShowConsoleClick(Sender: TObject);
426                 begin
427                 end;
428
429                 procedure TMain.MenuWebsiteClick(Sender: TObject);
430                 begin
431                 if not OpenURL('http://castle-engine.sourceforge.net/')
                       then
432                 MessageDlg('WWW browser not found on your system.',
                       mtError, [mbClose], 0);
433                 end;
434
435                 procedure TMain.ProgressBar1ContextPopup(Sender: TObject;
                       MousePos: TPoint;
436                 var Handled: Boolean);
437                 begin
438
439                 end;
440
441                 procedure TMain.RadioButton1Change(Sender: TObject);
442                 begin
443
444                 end;
445
446                 procedure TMain.Timer1Timer(Sender: TObject);
447                 var
448                 Pos, Dir, Up: TVector3;
449                 begin
450                 UpdateCaption; { to update FPS }
451
```

```
452                 { Update edit boxes about camera only from time to time.
453                 Otherwise, updating edit controls
454                 on every move would cause refresh rate of OpenGL context
                       to suffer
455                 (e.g. when rotating object in Examine mode) }
456                 if CameraChanged then
457                 begin
458                 CameraChanged := false;
459
460                 Browser.Camera.GetView(Pos, Dir, Up);
461                 { Note that Dir, Up returned here are always normalized }
462                 end;
463                 end;
464
465                 procedure TMain.envia(dado:byte);
466                 begin
467                 Lazserial1.WriteBuffer(dado,1);
468                 end;
469
470                 procedure TMain.MenuMouseLookToggleClick(Sender: TObject);
471                 var
472                 Walk: TWalkCamera;
473                 begin
474                 Walk := Browser.SceneManager.WalkCamera(false);
475                 if Walk <> nil then
476                 begin
477                 Walk.MouseLook := (Sender as TMenuItem).Checked;
478                 UpdateCrosshairImage;
479                 Repaint;
480                 end;
481                 end;
482
```

```
483                 procedure TMain.OnPointingDeviceSensorsChange(Sender:
                        TObject);
484                 var
485                 OverSensor: Boolean;
486                 SensorList: TPointingDeviceSensorList;
487                 begin
488                 { check if the crosshair (mouse) is over any sensor }
489                 OverSensor := false;
490                 SensorList := Browser.MainScene.PointingDeviceSensors;
491                 if (SensorList <> nil) then
492                 OverSensor := (SensorList.EnabledCount>0);
493
494                 if CrosshairActive <> OverSensor then
495                 begin
496                 CrosshairActive := OverSensor;
497                 UpdateCrosshairImage;
498                 end;
499                 end;
500
501                 procedure TMain.UpdateCrosshairImage;
502                 var
503                 Walk: TWalkCamera;
504                 begin
505                 Walk := Browser.SceneManager.WalkCamera(false);
506
507                 CrosshairCtl.Exists := ((Walk <> nil) and Walk.MouseLook);
508
509                 if CrosshairActive then
510                 CrosshairCtl.Shape := csCrossRect
511                 else
512                 CrosshairCtl.Shape := csCross;
513                 end;
```

```
514
515             procedure TMain.MenuAboutOpenGLClick(Sender: TObject);
516             begin
517             TOpenGLInformation.Execute;
518             end;
519
520             procedure TMain.ArquivosClick(Sender: TObject);
521             begin
522             Close;
523             end;
524
525             procedure TMain.MenuFocusGLControlClick(Sender: TObject);
526             begin
527             Browser.SetFocus;
528             end;
529
530             function MyGetApplicationName: string;
531             begin
532             Result := 'model_3d_viewer';
533             end;
534
535             procedure TMain.FormCreate(Sender: TObject);
536             begin
537             Browser.Load(ApplicationData('bracocompleto.wrl'));
538             //Browser.MainScene.Spatial := [ssRendering,
                   ssDynamicCollisions];
539             //Browser.MainScene.ProcessEvents := true;
540
541             //Browser.MainScene.OnPointingDeviceSensorsChange :=
                   @OnPointingDeviceSensorsChange;
542
543             braco0:= TCastleScene.Create(Application);
```

```
544                 braco0.Load(ApplicationData('braco0.wrl'));
545                 braco0.ProcessEvents := true;
546                 bracotrans0 := TCastleTransform.Create(Application);
547                 bracotrans0.Translate(Vector3(0, 0, 1.9));
548
549                 braco1:= TCastleScene.Create(Application);
550                 braco1.Load(ApplicationData('braco1.wrl'));
551                 braco1.ProcessEvents := true;
552                 bracotrans := TCastleTransform.Create(Application);
553                 bracotrans.Translate(Vector3(0, 10, 9.4));
554
555                 braco2:= TCastleScene.Create(Application);
556                 braco2.Load(ApplicationData('braco2.wrl'));
557                 braco2.ProcessEvents := true;
558                 bracotrans1 := TCastleTransform.Create(Application);
559                 bracotrans1.Translate(Vector3(0, 18.6, 9.4));
560
561                 bracotrans0.Add(braco0);
562                 Browser.SceneManager.Items.Add(bracotrans0);
563
564                 bracotrans.Add(braco1);
565                 Browser.SceneManager.Items.Add(bracotrans);
566
567                 bracotrans1.Add(braco2);
568                 Browser.SceneManager.Items.Add(bracotrans1);
569
570                 Browser.Camera.SetView(
571                 Vector3(19.07, 43.35, 38.83),
572                 Vector3(  -0.39, -0.64,   -0.66),
573                 Vector3(  -0.48, -0.47,   +0.74)
574                 );
575                 teta1_real:=0;
```

```
576                 teta2_real:=0;
577                 z_real:=0;
578                 Serial:=False;
579                 bmp:=Tbitmap.Create;
580                 bmp.Width:=image1.Width;
581                 bmp.Height:=image1.Height;
582                 image1.Picture.Bitmap:=bmp;
583                 bmp.free;
584                 end;
585
586                 function cinematica_inversa(x,y,teta1,teta2:real):angulos;
587                 var
588                 teta11,teta12,teta21,teta22:real;
589                 x_aux,y_aux:real;
590                 aux,aux_tg,aux_sqrt,aux1,Dist_minima,Dist_aux:real;
591                 Minimo:integer;
592
593
594                 begin
595                 aux_sqrt:=sqrt(x*x+y*y);
596
597                 aux:=arccos((x*x+y*y+8.6*8.6-10.5*10.5)/(2*8.6*aux_sqrt));
598                 aux1:=arccos((x*x+y*y-8.6*8.6+10.5*10.5)/(2*10.5*aux_sqrt)
                       );
599                 aux_tg:=arctan2(y,x);
600
601                 teta11:= aux+aux_tg;
602                 teta12:= aux_tg-aux;
603                 teta21:= aux1+aux_tg;
604                 teta22:= aux_tg-aux1;
605
606                 if (IsNan(aux) or IsNan(aux1)) then
```

```
607                 begin
608                 aux_tg:=0;
609                 end
610                 else
611                 begin
612                 x_aux:=8.6*cos(teta11)+10.5*cos(teta21);
613                 y_aux:=8.6*sin(teta11)+10.5*sin(teta21);
614                 Dist_minima:=(x-x_aux)*(x-x_aux)+(y-y_aux)*(y-y_aux)
                       +0.0001*(teta11-teta1)*(teta11-teta1)+0.0001*(teta21-
                       teta2)*(teta21-teta2);
615                 Minimo:=1;
616                 x_aux:=8.6*cos(teta12)+10.5*cos(teta21);
617                 y_aux:=8.6*sin(teta12)+10.5*sin(teta21);
618                 Dist_aux:=(x-x_aux)*(x-x_aux)+(y-y_aux)*(y-y_aux)+0.0001*(
                       teta12-teta1)*(teta12-teta1)+0.0001*(teta21-teta2)*(
                       teta21-teta2);
619                 if (Dist_aux<Dist_minima) then
620                 begin
621                 Minimo:=2;
622                 Dist_minima:=Dist_aux
623                 end;
624                 x_aux:=8.6*cos(teta11)+10.5*cos(teta22);
625                 y_aux:=8.6*sin(teta11)+10.5*sin(teta22);
626                 Dist_aux:=(x-x_aux)*(x-x_aux)+(y-y_aux)*(y-y_aux)+0.0001*(
                       teta11-teta1)*(teta11-teta1)+0.0001*(teta22-teta2)*(
                       teta22-teta2);
627                 if (Dist_aux<Dist_minima) then
628                 begin
629                 Minimo:=3;
630                 Dist_minima:=Dist_aux
631                 end;
632                 x_aux:=8.6*cos(teta12)+10.5*cos(teta22);
```

```
633                 y_aux:=8.6*sin(teta12)+10.5*sin(teta22);
634                 Dist_aux:=(x-x_aux)*(x-x_aux)+(y-y_aux)*(y-y_aux)+0.0001*(
                       teta12-teta1)*(teta12-teta1)+0.0001*(teta22-teta2)*(
                       teta22-teta2);
635                 if (Dist_aux<Dist_minima) then
636                 begin
637                 Minimo:=4
638                 end;
639                 case Minimo of
640                 1:begin
641                 teta1:=teta11;
642                 teta2:=teta21-teta11
643                 end;
644                 2:begin
645                 teta1:=teta12;
646                 teta2:=teta21-teta12
647                 end;
648                 3:begin
649                 teta1:=teta11;
650                 teta2:=teta22-teta11
651                 end;
652                 4:begin
653                 teta1:=teta12;
654                 teta2:=teta22-teta12
655                 end;
656                 end;
657
658                 end;
659                 Result.teta1:=teta1;
660                 Result.teta2:=teta2;
661
662                 end;
```

```
663
664             procedure TMain.FormDestroy(Sender: TObject);
665             begin
666             end;
667
668             procedure TMain.RecentFilesOpenRecent(const URL: string);
669             begin
670             OpenScene(URL);
671             end;
672
673             procedure TMain.Timer2Timer(Sender: TObject);
674             var
675             T1: TVector4;
676             T2: TVector4;
677             D0,D1,D2: TVector3;
678             x_real,y_real:real;
679             ang:angulos;
680             begin
681             if valor0>0.5 then Arrow4.ArrowColor:=clRed else Arrow4.
                   ArrowColor:=clScrollBar;
682             if valor0<-0.5 then Arrow1.ArrowColor:=clRed else Arrow1.
                   ArrowColor:=clScrollBar;
683
684             if valor3>0.5 then Arrow7.ArrowColor:=clRed else Arrow7.
                   ArrowColor:=clScrollBar;
685             if valor3<-0.5 then Arrow6.ArrowColor:=clRed else Arrow6.
                   ArrowColor:=clScrollBar;
686
687             if valor2>0.5 then Arrow8.ArrowColor:=clRed else Arrow8.
                   ArrowColor:=clScrollBar;
688             if valor2<-0.5 then Arrow5.ArrowColor:=clRed else Arrow5.
                   ArrowColor:=clScrollBar;
```

```
689
690                 x_real:=8.6*cos(teta1_real)+10.5*cos(teta1_real+teta2_real
                        );
691                 y_real:=8.6*sin(teta1_real)+10.5*sin(teta1_real+teta2_real
                        );
692                 if Direta then
693                 begin
694                 if valor0<-0.5 then
695                 begin
696                 teta1_real:=teta1_real+0.0019635;
697                 end;
698                 if valor0>0.5 then
699                 begin
700                 teta1_real:=teta1_real-0.0019635;
701                 end;
702
703                 if valor2<-0.5 then
704                 begin
705                 teta2_real:=teta2_real+0.0019635;
706                 end;
707                 if valor2>0.5 then
708                 begin
709                 teta2_real:=teta2_real-0.0019635;
710                 end;
711
712                 if valor3>0.5 then
713                 begin
714                 z_real:=z_real-0.01;
715                 end;
716                 if valor3<-0.5 then
717                 begin
718                 z_real:=z_real+0.01;
```

```
719                 end;
720                 x_real:=8.6*cos(teta1_real)+10.5*cos(teta1_real+teta2_real
                        );
721                 y_real:=8.6*sin(teta1_real)+10.5*sin(teta1_real+teta2_real
                        );
722                 end
723                 else
724                 begin
725                 if valor0<-0.5 then
726                 begin
727                 x_real:=x_real+0.01;
728                 end;
729                 if valor0>0.5 then
730                 begin
731                 x_real:=x_real-0.01;
732                 end;
733
734                 if valor2<-0.5 then
735                 begin
736                 y_real:=y_real+0.01;
737                 end;
738                 if valor2>0.5 then
739                 begin
740                 y_real:=y_real-0.01;
741                 end;
742
743                 if valor3>0.5 then
744                 begin
745                 z_real:=z_real-0.01;
746                 end;
747                 if valor3<-0.5 then
748                 begin
```

```
749                 z_real:=z_real+0.01;
750                 end;
751                 ang:=cinematica_inversa(x_real,y_real,teta1_real,
                        teta2_real);
752
753                 teta1_real:=ang.teta1;
754                 teta2_real:=ang.teta2;
755                 end;
756
757                 Edit7.Text:=FloatToStr(teta1_real*180/3.14159265);
758                 Edit8.Text:=FloatToStr(teta2_real*180/3.14159265);
759                 Edit9.Text:=FloatToStr(x_real*10);
760                 Edit10.Text:=FloatToStr(y_real*10);
761                 Edit11.Text:=FloatToStr(z_real);
762
763                 D0:= Vector3(0, 0,1.9+z_real);
764                 D1:= Vector3(0,10,9.4+z_real);
765                 T1:=Vector4(0,0,1,teta1_real);
766                 T2:=Vector4(0,0,1,teta1_real+teta2_real);
767
768                 bracotrans0.Translation:=D0;
769
770                 bracotrans.Translation:=D1;
771                 bracotrans.Rotation:=T1;
772
773                 D2:= Vector3(-8.6*sin(teta1_real),10.0+8.6*cos(teta1_real)
                        ,9.4+z_real);
774                 bracotrans1.Translation:=D2;
775                 bracotrans1.Rotation:=T2;
776
777                 end;
778
```

```
779                 procedure TMain.Timer3Timer(Sender: TObject);
780                 var
781                 T1: TVector4;
782                 T2: TVector4;
783                 D0,D1,D2: TVector3;
784                 x_real,y_real:real;
785                 x,y:real;
786                 b:byte;
787                 begin
788                 b:=0;
789                 if desenhando then
790                 begin
791                 if (abs(teta1_desejado-teta1_real)<=0.0018) and (abs(
                       teta2_desejado-teta2_real)<0.0018) and (abs(z_desejado-
                       z_real)<0.008) then
792                 begin
793                 x:=86.0*cos(teta1_desejado)+105.0*cos(teta1_desejado+
                       teta2_desejado)-125;
794                 y:=86.0*sin(teta1_desejado)+105.0*sin(teta1_desejado+
                       teta2_desejado)+30;
795                 if strtofloat(listbox2.items[posicao]) < 0.5 then
796                 begin
797                 image1.Picture.Bitmap.Canvas.MoveTo(trunc(x*20/3),400-
                       trunc(y*20/3));
798                 end;
799                 if strtofloat(listbox2.items[posicao]) > 0.5 then
800                 begin
801                 image1.Picture.Bitmap.Canvas.LineTo(trunc(x*20/3),400-
                       trunc(y*20/3));
802                 end;
803
804                 if (posicao<listbox2.Items.Count-4) then
```

```
805                 begin
806                 if strtofloat(listbox2.items[posicao+1]) < 0.5 then
                        z_desejado:=0.4 else z_desejado:=0;
807
808                 if (abs(z_desejado-z_real)<0.008) then
809                 begin
810                 posicao:=posicao+1;
811                 teta1_desejado:=strtofloat(listbox3.items[posicao])
                        *0.0019635;
812                 teta2_desejado:=strtofloat(listbox4.items[posicao])
                        *0.0019635;
813                 progressbar1.Position:=trunc(posicao*1000/listbox3.Items.
                        Count);
814                 end;
815                 end
816                 else timer3.enabled:=false;
817                 end;
818                 end;
819
820                 if abs(teta1_desejado-teta1_real)>=0.00190 then
821                 if teta1_desejado>teta1_real then
822                 begin
823                 teta1_real:=teta1_real+0.0019635;
824                 b:=b or 1;
825                 end
826                 else
827                 begin
828                 teta1_real:=teta1_real-0.0019635;
829                 b:=b or 3;
830                 end;
831                 //else teta1_real:=teta1_desejado;
832                 if abs(teta2_desejado-teta2_real)>=0.00190 then
```

```
833                 if teta2_desejado>teta2_real then
834                 begin
835                 teta2_real:=teta2_real+0.0019635;
836                 b:=b or 4;
837                 end
838                 else
839                 begin
840                 teta2_real:=teta2_real-0.0019635;
841                 b:=b or 12;
842                 end;
843                 //else teta2_real:=teta2_desejado;
844                 if abs(z_desejado-z_real)>=0.08 then
845                 if z_desejado>z_real then
846                 begin
847                 z_real:=z_real+0.1;
848                 b:=b or 16;
849                 end
850                 else
851                 begin
852                 z_real:=z_real-0.1;
853                 b:=b or 48;
854                 end;
855                 x_real:=8.6*cos(teta1_real)+10.5*cos(teta1_real+teta2_real
                        );
856                 y_real:=8.6*sin(teta1_real)+10.5*sin(teta1_real+teta2_real
                        );
857                 if z_real < 0.1 then
858                 begin
859                 pontos[j]:= TCastleScene.Create(Application);
860                 pontos[j].Load(ApplicationData('bola.wrl'));
861                 pontos[j].ProcessEvents := true;
862                 pontost[j] := TCastleTransform.Create(Application);
```

```
863                 pontost[j].Translate(Vector3(-y_real, 10.0+x_real, 4));
864                 pontost[j].Add(pontos[j]);
865                 Browser.SceneManager.Items.Add(pontost[j]);
866                 j:=j+1;
867                 end;
868                 if groupbox1.Enabled then
869                 begin
870                 Label1.Caption:=FloatToStr(teta1_real*180/3.14159265);
871                 Label2.Caption:=FloatToStr(teta2_real*180/3.14159265);
872                 Label3.Caption:=FloatToStr(z_real);
873                 Label7.Caption:=FloatToStr(x_real*10);
874                 Label8.Caption:=FloatToStr(y_real*10);
875                 Label9.Caption:=FloatToStr(z_real);
876                 end;
877                 D0:= Vector3(0, 0,1.9+z_real);
878                 D1:= Vector3(0,10,9.4+z_real);
879                 T1:=Vector4(0,0,1,teta1_real);
880                 T2:=Vector4(0,0,1,teta1_real+teta2_real);
881                 bracotrans0.Translation:=D0;
882                 bracotrans.Translation:=D1;
883                 bracotrans.Rotation:=T1;
884                 D2:= Vector3(-8.6*sin(teta1_real),10.0+8.6*cos(teta1_real)
                        ,9.4+z_real);
885                 bracotrans1.Translation:=D2;
886                 bracotrans1.Rotation:=T2;
887                 Edit7.Text:=FloatToStr(teta1_real*180/3.14159265);
888                 Edit8.Text:=FloatToStr(teta2_real*180/3.14159265);
889                 Edit13.Text:=FloatToStr(teta1_desejado*180/3.14159265);
890                 Edit14.Text:=FloatToStr(teta2_desejado*180/3.14159265);
891                 Edit15.Text:=InttoStr(posicao);
892                 Edit17.Text:=FloatToStr(z_desejado*10);
893                 Edit9.Text:=FloatToStr(x_real*10);
```

```
894                 Edit10.Text:=FloatToStr(y_real*10);
895                 Edit11.Text:=FloatToStr(z_real*10);
896                 if Serial=True then
897                 begin
898                 if (b<>0) then
899                 envia(b);
900                 end;
901                 end;
902                 procedure TMain.ToggleBox1Change(Sender: TObject);
903                 begin
904                 timer2.Interval:=StrToInt(edit12.Text);
905                 end;
906                 procedure posicionar(teta1:real; teta2:real; z:real) ;
907                 begin
908                 z_desejado:=z;
909                 teta1_desejado:=teta1;
910                 teta2_desejado:=teta2;
911                 end;
912                 procedure TEventsHandler.JoyAxisMove(const Joy: PJoy;
                       const Axis: Byte;
913                 const Value: Single);
914                 begin
915                 // We showing axes position only for selected joystick.
916                 // If axes labels not initialized yet then exit.
917                 if Axis=0 then
918                 begin
919                 eixo0:=Axis;
920                 valor0:=Value;
921                 end;
922                 if Axis=1 then
923                 begin
924                 eixo1:=Axis;
```

```
925                 valor1:=Value;
926                 end;
927                 if Axis=2 then
928                 begin
929                 eixo2:=Axis;
930                 valor2:=Value;
931                 end;
932                 if Axis=3 then
933                 begin
934                 eixo3:=Axis;
935                 valor3:=Value;
936                 end;
937                 end;
938                 procedure TMain.TrackBar4Change(Sender: TObject);
939                 var
940                 teta1,teta2:real;
941                 x,y,z:real;
942                 ang:angulos;
943                 begin
944                 if (TrackBar1.Focused or TrackBar2.Focused or TrackBar3.
                       Focused) then
945                 begin
946                 teta1:=(TrackBar1.Position/2400)*(0.75*3.14159265);
947                 teta2:=(TrackBar2.Position/2400)*(0.75*3.14159265);
948                 z:=(TrackBar3.Position/1000);
949                 Edit1.Text:=FloatToStr(teta1*180/3.14159265);
950                 Edit2.Text:=FloatToStr(teta2*180/3.14159265);
951                 Edit3.Text:=FloatToStr(z);
952                 x:=8.6*cos(teta1)+10.5*cos(teta1+teta2);
953                 y:=8.6*sin(teta1)+10.5*sin(teta1+teta2);
954                 z:=z;
955                 Edit4.Text:=FloatToStr(x);
```

```
956                 Edit5.Text:=FloatToStr(y);
957                 Edit6.Text:=FloatToStr(z);
958                 TrackBar4.Position:=round(x*1000);
959                 TrackBar5.Position:=round(y*1000);
960                 TrackBar6.Position:=round(z*1000);
961                 posicionar(teta1,teta2,z);
962                 end;
963                 if (TrackBar4.Focused or TrackBar5.Focused or TrackBar6.
                       Focused) then
964                 begin
965                 x:=(TrackBar4.Position/1000);
966                 y:=(TrackBar5.Position/1000);
967                 z:=(TrackBar6.Position/1000);
968                 teta1:=(TrackBar1.Position/2400)*(0.75*3.14159265);
969                 teta2:=(TrackBar2.Position/2400)*(0.75*3.14159265);
970                 ang:=cinematica_inversa(x,y,teta1,teta2);
971                 teta1:=ang.teta1;
972                 teta2:=ang.teta2;
973                 //dteta1:=teta1-ang.teta1;
974                 //dteta2:=teta2-ang.teta2;
975                 Label7.Caption:=FloatToStr(x);
976                 Edit4.Text:=FloatToStr(x);
977                 Label8.Caption:=FloatToStr(y);
978                 Edit5.Text:=FloatToStr(y);
979                 Label9.Caption:=FloatToStr(z);
980                 Edit6.Text:=FloatToStr(z);
981                 TrackBar3.Position:=round(z*1000);
982                 TrackBar2.Position:=round((teta2*2400)/(0.75*3.14159265));
983                 TrackBar1.Position:=round((teta1*2400)/(0.75*3.14159265));
984                 Edit1.Text:=FloatToStr(teta1*180/3.14159265);
985                 Edit2.Text:=FloatToStr(teta2*180/3.14159265);
986                 Edit3.Text:=FloatToStr(z);
```

```
987                 posicionar(teta1,teta2,z);
988                 end;
989                 end;
990
991                 procedure TMain.ButtonNavigationTypeClick(Sender: TObject)
                        ;
992                 begin
993                 end;
994
995                 procedure TMain.Button2Click(Sender: TObject);
996                 var
997                 teta1,teta2:real;
998                 x,y,z:real;
999                 begin
1000                Browser.Camera.SetView(
1001                Vector3(19.07, 43.35, 38.83),
1002                Vector3(  -0.39, -0.64,   -0.66),
1003                Vector3(  -0.48, -0.47,   +0.74)
1004                );
1005
1006                teta1:=StrToFloat(Edit1.Text)*3.14159265/180;
1007                teta2:=StrToFloat(Edit2.Text)*3.14159265/180;
1008                z:=StrToFloat(Edit3.Text);
1009
1010                x:=8.6*cos(teta1)+10.5*cos(teta1+teta2);
1011                y:=8.6*sin(teta1)+10.5*sin(teta1+teta2);
1012                z:=z;
1013
1014                TrackBar3.Position:=round(z*1000);
1015                TrackBar2.Position:=round((teta2*2400)/(0.75*3.14159265));
1016                TrackBar1.Position:=round((teta1*2400)/(0.75*3.14159265));
1017                Label1.Caption:=FloatToStr(teta1*180/3.14159265);
```

```
1018                Edit1.Text:=FloatToStr(teta1*180/3.14159265);
1019                Label2.Caption:=FloatToStr(teta2*180/3.14159265);
1020                Edit2.Text:=FloatToStr(teta2*180/3.14159265);
1021                Label3.Caption:=FloatToStr(z);
1022                Edit3.Text:=FloatToStr(z);
1023
1024                Label7.Caption:=FloatToStr(x);
1025                Edit4.Text:=FloatToStr(x);
1026                Label8.Caption:=FloatToStr(y);
1027                Edit5.Text:=FloatToStr(y);
1028                Label9.Caption:=FloatToStr(z);
1029                Edit6.Text:=FloatToStr(z);
1030
1031                posicionar(teta1,teta2,z);
1032                end;
1033
1034                procedure TMain.Button3Click(Sender: TObject);
1035                var
1036                teta1,teta2:real;
1037                x,y,z:real;
1038                ang:angulos;
1039                begin
1040                x:=StrToFloat(Edit4.Text);
1041                y:=StrToFloat(Edit5.Text);
1042                z:=StrToFloat(Edit6.Text);
1043                teta1:=(TrackBar1.Position/2400)*(0.75*3.14159265);
1044                teta2:=(TrackBar2.Position/2400)*(0.75*3.14159265);
1045
1046                ang:=cinematica_inversa(x,y,teta1,teta2);
1047                teta1:=ang.teta1;
1048                teta2:=ang.teta2;
1049
```

```
1050                TrackBar3.Position:=round(z*1000);
1051                TrackBar2.Position:=round((teta2*2400)/(0.75*3.14159265));
1052                TrackBar1.Position:=round((teta1*2400)/(0.75*3.14159265));
1053                Label1.Caption:=FloatToStr(teta1*180/3.14159265);
1054                Edit1.Text:=FloatToStr(teta1*180/3.14159265);
1055                Label2.Caption:=FloatToStr(teta2*180/3.14159265);
1056                Edit2.Text:=FloatToStr(teta2*180/3.14159265);
1057                Label3.Caption:=FloatToStr(z);
1058                Edit3.Text:=FloatToStr(z);
1059
1060                Label7.Caption:=FloatToStr(x);
1061                Edit4.Text:=FloatToStr(x);
1062                Label8.Caption:=FloatToStr(y);
1063                Edit5.Text:=FloatToStr(y);
1064                Label9.Caption:=FloatToStr(z);
1065                Edit6.Text:=FloatToStr(z);
1066
1067                posicionar(teta1,teta2,z);
1068                teta1_real:=teta1;
1069                teta2_real:=teta2;
1070                z_real:=z;
1071                end;
1072
1073                procedure TMain.Button4Click(Sender: TObject);
1074                begin
1075                Label6.Caption:='Direta';
1076                Direta:=true;
1077                end;
1078
1079                procedure TMain.Button5Click(Sender: TObject);
1080                begin
1081                Label6.Caption:='Indireta';
```

```
1082                Direta:=false;
1083                end;
1084
1085                procedure TMain.Button6Click(Sender: TObject);
1086                var
1087                x,y:real;
1088                begin
1089
1090
1091                image1.Picture.Bitmap.Canvas.Pen.Color:=clred;
1092                teta1_desejado:=strtofloat(listbox3.items[0])*0.0019635;
1093                teta2_desejado:=strtofloat(listbox4.items[0])*0.0019635;
1094                x:=86.0*cos(teta1_desejado)+105.0*cos(teta1_desejado+
                       teta2_desejado)-125;
1095                y:=86.0*sin(teta1_desejado)+105.0*sin(teta1_desejado+
                       teta2_desejado)+30;
1096                verificou:=false;
1097                chegou:=false;
1098                timer3.Interval:=20;
1099                timer3.Enabled:=true;
1100                desenhando:=true;
1101                button6.enabled:=false;
1102                end;
1103
1104                procedure TMain.Button7Click(Sender: TObject);
1105                begin
1106                teta1_real:=0;
1107                teta2_real:=0;
1108                z_real:=0;
1109
1110                TrackBar1.Position:=round(0);
1111                TrackBar2.Position:=round(0);
```

```
1112                TrackBar3.Position:=round(0);
1113                TrackBar4.Position:=round(19.1*1000);
1114                TrackBar5.Position:=round(0);
1115                TrackBar6.Position:=round(0);
1116                Label1.Caption:=FloatToStr(0);
1117                Edit1.Text:=FloatToStr(0);
1118                Label2.Caption:=FloatToStr(0);
1119                Edit2.Text:=FloatToStr(0);
1120                Label3.Caption:=FloatToStr(0);
1121                Edit3.Text:=FloatToStr(0);
1122                Label7.Caption:=FloatToStr(19.1);
1123                Edit4.Text:=FloatToStr(19.1);
1124                Label8.Caption:=FloatToStr(0);
1125                Edit5.Text:=FloatToStr(0);
1126                Label9.Caption:=FloatToStr(0);
1127                Edit6.Text:=FloatToStr(0);
1128                Label3.Caption:=FloatToStr(0);
1129                Edit3.Text:=FloatToStr(0);
1130                end;
1131
1132                procedure TMain.Button9Click(Sender: TObject);
1133                begin
1134                Browser.Camera.SetView(
1135                Vector3(0, 15, 60),
1136                Vector3(  0, 0,   -1),
1137                Vector3(  0, -1,   0)
1138                );
1139                end;
1140
1141                procedure TMain.Button1Click(Sender: TObject);
1142                begin
1143                Browser.Camera.SetView(
```

```
                Vector3(19.07, 43.35, 38.83),
                Vector3(  −0.39, −0.64,   −0.66),
                Vector3(  −0.48, −0.47,   +0.74)
                );
                end;

                procedure TMain.BrowserCameraChanged(Camera: TCamera);
                begin
                CameraChanged := true;
                end;

                initialization
                {$I mainf.lrs}
                end.
```