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

Projeto de um braço robótico para fins didáticos

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Lucas Kenzo Kato

Florianópolis, Fevereiro de 2015

Projeto de um braço robótico para fins didáticos

Lucas Kenzo Kato

Esta monografia foi julgada no contexto da disciplina

DAS5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. *Rodrigo Antônio Marques Braga*

_____

Assinatura do Orientador

Banca Examinadora:

Prof. Rodrigo Antônio Marques Braga
Orientador na Empresa

Prof. Rodrigo Antônio Marques Braga
Orientador no Curso

Prof. Marcelo de Lellis Costa de Oliveira
Avaliador

Tiago Nunes Dalosto
Gustavo de Souza Satyro
Debatedores

Agradecimentos

Aos professores Rodrigo Braga e Ana V. Pazmino, pela orientação e oportunidade de participar deste projeto.

Ao Professor Henrique Simas, pela orientação em robótica.

Aos colegas de trabalho.

A minha família e amigos, pelo incentivo e inspiração.

Resumo

O desenvolvimento do presente trabalho está associado ao projeto de extensão “ações para o museu de ciência e tecnologia ufsc/jville”e foi realizado no grupo Multidesign do Laboratório Pronto3D/CCE/UFSC – Laboratório de Prototipagem e Novas Tecnologias orientadas ao 3D – cujo espaço é destinado ao ensino, pesquisa e extensão na área da materialização da forma por meio de equipamentos automatizados, tais como impressão 3D, corte a laser e fresadoras CNC.

Os principais objetivos do trabalho foram projetar e implementar um robô manipulador para fins didáticos, como uma forma de popularizar a tecnologia no espaço de ciência da UFSC/Joinville, buscando alcançar o público do ensino médio, técnico e superior. O projeto seguiu a metodologia de desenvolvimento de produto de Bonsiepe adaptada para o trabalho, a qual inclui: pesquisa e análise de produtos similares (análise sincrônica) presentes no mercado e de projetos no estado da arte; levantamento de requisitos estruturais, de *hardware* e *software*; desenvolvimento de uma interface de programação e de um *firmware* interpretador.

Como resultados, obteve-se um braço robótico de baixa complexidade com quatro graus de liberdade do tipo antropomórfico materializado via impressora 3D, além de possuir funções similares aos braços utilizados em indústrias, como a função *teach* e um tentativa de implementação de movimentos retilíneos, desenvolvida a partir de conceitos de cinemática de robôs manipuladores. A interface de programação possui um campo para comando direto do robô através de um terminal e outro, para a programação de códigos mais extensos, os quais são gravados no cartão SD e executados pelo *Firmware.*

Abstract

The present work development is associated to an extension Project called “Actions for the science and technology museum of Ufsc/jville” and was conducted at the Multidesign group of Pronto3D Laboratory – Prototyping and New Technologies Oriented to 3D Laboratory – a space intended for education, research and extension in the form materialization area through automated equipment such as 3D printing, laser cutting and CNC milling machines.

The main objectives of this Project were to design and implement a robotic arm for teaching purposes, as a way to popularize the technology in the Science Space at Ufsc/Joinville, aiming to reach the high school, technical and college students. The Project followed the Bonsiepe product development methodology, adapted to this work, which includes: Market research of similar products (Synchronic Analysis); analysis of such products to get structural, hardware and software requirements; development of a programming interface and a interpreter firmware.

As a result, we obtained a low complexity robotic arm with four degrees of freedom of anthropomorphic type manufactured by a 3D printer, besides having similar functions to the arms used in industries such as teaching functions and an attempt to have rectilinear motion implementation, developed from robotic kinematics concepts. The programming interface has a field to directly command the robot through a terminal and another to program extensive codes. These are recorded on the SD card and run in the firmware.

Sumário

Simbologia

ABS – Acrylonitrile Butadiene styrene

ACL – Advanced Control Language

ADC – Analog to Digital Converter

API – Application Program Interface

CAD – Computer Aided Design

DAC – Digital to Analog Converter

DH – Denavit-Hartenberg

Eeprom – Electrically-Erasable Programmable Read-Only Memory

GPIO – General Purpose Input/Output

IDE – Integrated Development Environment

IFR – International Federation of Robotics

LED – Light Emitting Diode

MDF – Medium Density Fiberboard

NA – Não aplicável

NC – Não contém

NI – Não informado

PLA – Polylactic acid

RAD – Rapid Application Development

SD – Secure Digital Card

USB – Universal Serial bus

VPL – Visual Programming Language

Lista de figuras

Lista de Quadros

Capítulo 1: Introdução

Nos últimos anos, com o desenvolvimento tecnológico, redução de custos de componentes eletrônicos e crescimento da mão de obra especializada [1], o campo da robótica vem se popularizando e se difundindo cada vez mais nos países em desenvolvimento, principalmente nas indústrias automobilísticas. Segundo a IFR (International Federation of Robotics) [2], em 2013, cerca de 70.000 unidades de braços robóticos foram vendidas no mundo e, em 2011, 60.000. Como reflexo dessa popularização, muitos grupos de pesquisadores e interessados no assunto começaram a investir nessa área através do uso dos robôs como instrumento de auxílio para suas atividades bem como para o ensino técnico focado no seu funcionamento.

Desse modo, o presente trabalho será direcionado para o ensino em robótica através de um projeto de um braço manipulador. Nesta seção são apresentadas as bases deste projeto, as quais incluem a motivação, os objetivos e o resumo de cada capítulo deste trabalho.

1.1: Motivação

Um dos grandes desafios no campo do ensino-aprendizagem é a aplicação prática de algum conceito ou teoria apresentada em sala de aula [3], este desafio ainda se verifica nos diversos níveis de ensino, seja no básico, técnico ou superior, principalmente nas engenharias.

Especificamente no campo das ciências exatas, existem algumas ferramentas tecnológicas que ajudam a superar esse desafio, seja através de programas de simulação numérica, os quais obtiveram um crescimento significativo com a evolução das Tecnologias de Informação e Comunicação e dos dispositivos eletrônicos, aproximando o conteúdo visto em aula com alguma aplicação virtual, como também através da prototipagem rápida de projetos, fazendo com que os alunos sintam-se ainda mais motivados no aprendizado de determinado assunto

com a observação e construção de algo “palpável”, sendo este último o grande propulsor da chamada Cultura *Maker*, também inspirada na teoria do Construtivismo que será abordada no capítulo 3.

Este trabalho está associado a um projeto de extensão de demanda interna: ações para o museu de ciência e tecnologia ufsc/jville. O público-alvo do projeto de extensão engloba cerca de 1000 crianças e adolescentes da rede pública estadual de Santa Catarina nas cidades de Joinville, Florianópolis e Araranguá, que farão a visita ao Espaço de Ciência e Tecnologia do Campus Joinville. O foco do projeto é popularizar a ciência e tecnologia para alunos do ensino fundamental II (9-12 anos), ensino médio (13-16 anos) e alunos do ensino superior.

Os resultados deste projeto, bem como outros projetos que procuram incentivar a ciência e tecnologia com jovens alunos do ensino médio são descritos no livro “Desafios da Educação em Engenharia: Vocação, Formação, Exercício Profissional, Experiências Metodológicas e Proposições” [4].Vários projetos com este intuito fazem parte de uma renovação pedagógica para popularizar a ciência e tecnologia aproximando as crianças e adolescentes.

O desenvolvimento do presente trabalho foi realizado no grupo Multidesign do Laboratório Pronto3D – Laboratório de Prototipagem e Novas Tecnologias orientadas ao 3D – cujo espaço é destinado ao ensino, pesquisa e extensão na área da materialização da forma por meio de equipamentos automatizados, tais como impressão 3D, corte a laser e fresadoras CNC. O laboratório, do curso de design da UFSC de Florianópolis, é composto por professores pesquisadores de diversas áreas, alunos de graduação e pós-graduação, além de profissionais envolvidos em projetos que se desdobram em aplicações de atividades de pesquisa e extensão. Além disso, este espaço faz parte de uma rede de laboratórios de prototipagem rápida e fabricação digital, denominada Rede Pronto3D, a qual tem como objetivo a estruturação de centros estrategicamente localizados no estado de Santa Catarina, atualmente nas cidades de Florianópolis, Lages, Criciúma e Chapecó. A Rede Pronto3D atende aos cursos de Design, Arquitetura, Engenharias e todas as outras áreas que envolvem criação, desenvolvimento e produção de modelos, protótipos,

maquetes e produtos em escala real, auxiliando as diferentes etapas do processo de projeto.

Com base nisso, a motivação deste projeto é ser uma alternativa para popularizar a tecnologia e suprir a lacuna existente entre o que se vê em sala de aula (teorias) e a parte prática no campo dos robôs manipuladores, enfatizando o processo de desenvolvimento de produto e a criação de uma interface de programação para o controle do robô.

1.2: Objetivo do projeto

1.2.1: Geral

O principal objetivo é projetar e materializar um kit de robótica industrial de baixo custo, composto por um braço robótico de pequenas dimensões para fins didáticos, auxiliando no processo de ensino e aprendizagem nas escolas de nível técnico e superior e no espaço de ciência e tecnologia UFSC/Jville. A finalidade do kit robótico é o ensino da programação, construção, modelagem e controle de robôs.

1.2.2: Específico

- **Levantar informações de robôs manipuladores e robótica educacional;**

  Parte deste objetivo será o fundamento do desenvolvimento de produto, analisando os possíveis modelos conceituais de braços robóticos, alguns modelos disponíveis no mercado e aqueles que estão no estado da arte. Quanto à parte educacional, será analisado como este ramo vem se desenvolvendo em âmbito nacional e como este projeto poderá suprir alguma lacuna existente na robótica educacional.

- **Especificar restrições e detalhes mecânicos para o modelo do braço robótico;**

Após a definição da estrutura mecânica básica do braço serão detalhados os componentes estruturais secundários relativos à rigidez e movimentação do braço. Além disso o robô manipulador será construído com juntas do tipo rotativa.

- **Projetar o sistema eletrônico e de software;**

  Nesta etapa avaliar-se-ão os controladores, plataformas de desenvolvimento, atuadores, interface de controle e linguagem de programação que são possíveis para o uso no projeto, justificando a escolha de cada um deles.

- **Materializar três modelos de braço robótico;**

  A materialização será feita através dos equipamentos disponíveis no laboratório Pronto3D, tais como impressora 3D, fresadora e máquinas de corte a laser. O primeiro modelo é o mais simples, cuja ideia é fazer com que o usuário monte a própria estrutura do braço pré-produzida em um material semelhante á cartolina, porém mais resistente através do recurso de modelamento de chapas disponível no software CAD do *Solidworks*. No segundo, busca-se um modelo mais robusto de pequenas dimensões a ser materializado via impressão 3D e o último, de dimensões maiores feito em MDF[1] e com acionamentos mais potentes que os anteriores.

- **Testar o produto;**

- **Avaliar os resultados;**

[1] Chapas de madeira comprimidas com resina, comumente usadas nas confecções de móveis e pequenos objetos [5]

1.3: Estrutura do trabalho

No capítulo 1 são apresentados os aspectos gerais do projeto, esclarecendo em qual área da robótica o trabalho está focado, quais as motivações que levaram ao início do mesmo, objetivos gerais e específicos e onde foi realizado.

Em seguida, no capítulo 2, discorre-se sobre a evolução histórica da robótica educacional, quando surgiram os primeiro projetos de grande relevância e finalizando com as contribuições que este trabalho oferece para a área em questão.

No capítulo 3 são abordadas as teorias e conceitos de robôs manipuladores fundamentais para este projeto, como os conceitos estruturais de um braço, os tipos existentes com suas vantagens e desvantagens e a teoria de movimentos de corpos rígidos, cinemática direta e inversa.

No capítulo seguinte, apresenta-se a metodologia de desenvolvimento de produto, no qual se inicia com a descrição do problema envolvido e segue com análises de produtos disponíveis no mercado e projetos no estado da arte. Com base nesta análise, geram-se as especificações de requisitos estruturais, tecnológicos e de software.

A descrição de como o projeto foi implementado é feita no capítulo 5, explicando como a interface de programação e o código do microcontrolador (*firmware*) foram feitos, apresentando as funções criadas e o detalhamento de uma função que engloba os conceitos e teoria de manipuladores robóticos, além de esclarecer os pontos relevantes na materialização do braço.

Os testes da interface e das funções do *firmware* são feitos no capítulo 6, incluindo os testes de movimentação do braço.

Por último, finaliza-se o trabalho com conclusões e resultados, apresentando, também, quais disciplinas do Curso de Engenharia de Controle e Automação foram contemplados e os futuros trabalhos que poderão ser feitos.

Capítulo 2: Robótica e Educação

O desenvolvimento tecnológico e de processos de produção das últimas décadas têm reduzido consideravelmente o custo de componentes eletrônicos, proporcionando, consequentemente, maior acessibilidade a determinados produtos para os mais diversos níveis sociais. Este processo também ocorre com a robótica. Cada vez mais empregada por indústrias de produção em larga escala e com condições de trabalho insalubres. A área da educação também segue o mesmo fluxo, pois com a rapidez do surgimento de novas ideias para o uso das tecnologias já existentes e amadurecidas, há a possibilidade de maior acesso à tecnologia robótica nas salas de aula de ensino superior, técnico e básico e, inclusive, para uso pessoal.

Além do aspecto tecnológico envolvido na robótica educativa, há a imensa contribuição da psicologia, que com seus fundamentos teóricos possibilitou grandes avanços na tecnologia educacional até chegar à robótica.

2.1: Bases da robótica educativa

Nas décadas de 50 e 60, apesar dos computadores ainda serem um aparato de dificílimo acesso à maioria das pessoas, eles foram o estopim de uma grande mudança na forma do ensino-aprendizagem tradicional. Nesse período a corrente “Behaviorista” estava em alta, cujo fundamento estava estruturado em análises de situações que se podiam apenas observar e medir, sem levar em consideração aspectos da mente ou da consciência. Um dos grandes estudiosos dessa corrente foi Burrhus Skinner, o qual formulou pressupostos no processo de ensino e aprendizagem que geram os seguintes passos: conteúdos a ensinar e aprendizagem gradual com objetivos bem específicos e com grau de dificuldade crescente. Décadas depois, por volta dos anos 80 e 90, estando os computadores mais difundidos, foram criados programas mais detalhados que buscavam adaptar-se ao ritmo de aprendizagem do aluno [6]. Assim, as famosas TICs – Tecnologias de

Informação e Comunicação – mostraram o seu grande potencial no ensino assistido por computador.

Com a dinâmica que o mundo da informática, mais especificamente com a internet, trouxe, as ideias “behavioristas” começaram a mostrar seus pontos fracos, já que seus fundamentos eram rígidos e consideravam o aluno como um instrumento, desprezando seus aspectos mentais, tão importantes no processo de aprendizagem. Isso proporcionou uma abertura para uma nova corrente da psicologia, chamada Construtivismo, em que considerava a aprendizagem como um processo ativo, ao contrário de só se obter um conhecimento através de uma leitura ou um discurso de um professor [7]. Sendo o principal objetivo construir o conhecimento e não, simplesmente, transmiti-lo.

Dessa forma, o foco das atividades educativas parte do princípio de manusear algo concreto com objetivos claros e que despertem o interesse dos alunos em determinado assunto. Isso é o que a robótica educacional em conjunto com as TICs tem como ponto forte, podendo ser utilizadas no ensino de diversos ramos do conhecimento, como também no desenvolvimento de aptidões dos alunos, como raciocínio lógico, trabalho em equipe, imaginação, criatividade e etc. No entanto, as TICs têm como principal fraqueza a necessidade de se ter instrutores capacitados, aptos a utilizar essas tecnologias de forma a sincronizar os conhecimentos que os alunos devem compreender com atividades motivadoras e que façam com que os alunos busquem por si só solucionar os problemas colocados a eles.

Um dos grandes passos dados nesse assunto foi o desenvolvimento da Linguagem LOGO no MIT – Massachusetts Institute of Technology – dirigido por Seymour Papert na década de 60 [8]. O Logo é uma linguagem derivada da linguagem de programação Lisp [9]. No início o projeto possuía um robô móvel que podia ser controlado via computador, sendo que mais tarde, esse robô foi substituído por uma tartaruga gráfica, vista na própria interface do programa, podendo ser controlada por crianças devido a sua linguagem de programação ser simples e intuitiva. Além disso, devido à sua ampla aplicação em assuntos distintos como matemática, música, robótica, telecomunicações, entre outros, ela foi bem aceita em

estudos mais sofisticados. Mais tarde, nos anos 80, foi desenvolvido um sistema com a interface de programação Logo utilizando motores, sensores e os blocos da famosa LEGO no MIT Media Lab. O projeto foi um sucesso entre professores e alunos, iniciando uma nova etapa na robótica educacional.

2.2: Contribuições do projeto na robótica educativa

Com base na evolução de recursos tecnológicos e inovações que contemplam essa modalidade de educação, o presente projeto busca criar um produto com as tecnologias mais atuais e difundidas no mercado, sendo sustentado pelas ideias Construcionistas, ou seja, aprender fazendo.

Existem diversas áreas de estudo quando se fala em robótica, como a mecânica, eletrônica e computação. Para o ensino em nível técnico o braço robótico desenvolvido auxilia na introdução à programação, possuindo uma interface com linguagem de programação simples, de alto nível e semelhante às que são utilizadas por alguns modelos reais, como o ACL e o *Scorbot ER-VII* [10]. Também o projeto busca aproximar a lógica de movimentação dos modelos industriais, como por exemplo, gravar inúmeras posições do órgão terminal do braço e fazê-lo seguir esses pontos de forma retilínea ou não.

Como parte do projeto é aberta (programa disponível), principalmente com respeito à programação do microcontrolador utilizado, o usuário pode alterar as funcionalidades do braço. Isto incentiva, inclusive, a construir outros modelos de manipuladores, tornando interessante o ensino superior em robótica, com a possibilidade de se verificar como os parâmetros físicos do braço podem dificultar a programação para a geração de trajetórias com a Cinemática Direta e Inversa, que será apresentada no capítulo seguinte.

Capítulo 3: Fundamentação teórica de manipuladores robóticos

Neste capítulo serão enfatizados os aspectos teóricos dos manipuladores robóticos que foram a base para o desenvolvimento deste projeto, apresentando os diversos tipos de braços existentes e a teoria matemática necessária para a geração de movimentos programados do robô, tais como: movimento de corpo rígido, cinemática direta e inversa.

3.1: Aspectos estruturais

A estrutura mecânica de um braço robótico consiste basicamente de elos (parte rígida que se movimenta de acordo com a junta), juntas (articulação), punho e um órgão terminal ou efetuador. Estes podem ser de diversos tipos, dependendo do que se deseja fazer, como garras, dispositivos de sucção, ponteiras para soldagem e etc. De um modo geral, o braço (elos e juntas) proporciona a mobilidade, o punho, a agilidade e o órgão terminal realiza a tarefa desejada [11]. Na Figura 1 é apresentado o modelo de um braço com suas partes estruturais. Onde se vê (*First arm* e *second arm*), ali estão os elos, e (*1st axis, 2nd axis...*), as juntas. O punho é composto pelo 4º, 5º e 6º eixo ou junta.

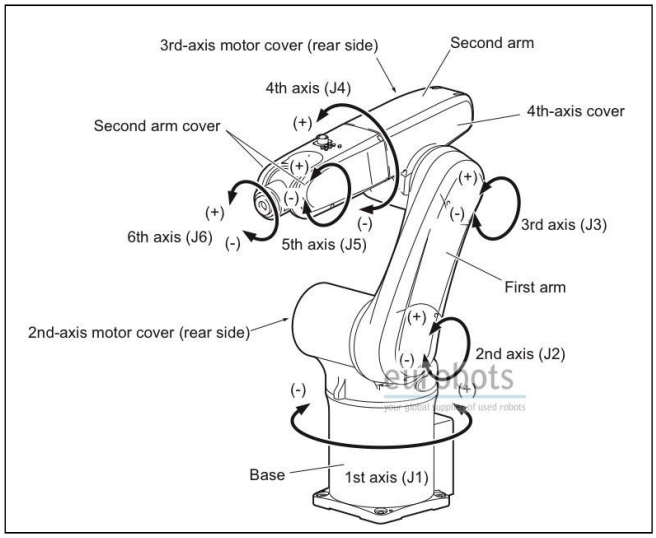


Figura 1 – Braço robótico
Fonte: [12]

Os braços também podem ser caracterizados como cadeias cinemáticas abertas ou fechadas. As abertas possuem uma única sequência de elos e juntas que ligam a base do robô até o manipulador, já as de cadeia fechada possuem mais de uma possibilidade.

A mobilidade do robô é determinada, principalmente, pelo tipo de junta utilizada, podendo ser rotativa ou prismática. A rotativa permite um movimento rotativo relativo a elos adjacentes à junta, enquanto que a prismática permite movimentos translacionais. Na cadeia cinemática aberta, cada junta representa um grau de liberdade (*degree of freedom*) ou DOF.

O volume de trabalho (Figura 5 e 6) caracteriza a região que o órgão terminal pode acessar.

No tópico seguinte são descritos os tipos de robôs manipuladores de acordo com o tipo de junta utilizada.

3.2: Tipos de robôs manipuladores

Existem modelos conceituais de robôs manipuladores que possuem características distintas, seja no volume de trabalho como em aspectos de robustez,

velocidade, exatidão nos movimentos e etc. Os principais modelos serão apresentados em seguida.

3.2.1: Cartesiano

O volume de trabalho deste tipo de robô manipulador é um paralelepípedo, com juntas do tipo prismáticas. Permite que o efetuador se movimente apenas com movimentos lineares, possuindo 3 graus de liberdade, como pode ser observado na Figura 2. Isso garante uma boa exatidão nos movimentos, porém, oferece pouca velocidade de atuação devido aos tipos de atuadores da junta que se utilizam atualmente. A estrutura oferece boa resistência podendo ser aplicado em atividades que necessitam de grande esforço, como no transporte de cargas pesadas [11].

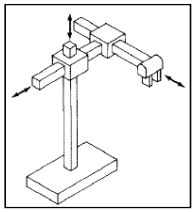


Figura 2 – Robô cartesiano
Fonte: [13]

3.2.2: Cilíndrico

Os robôs manipuladores que recebem esta denominação possuem seu volume de trabalho próximo a de um cilindro (Figura 3). A principal diferença entre este e o modelo anterior é a troca da junta da base por uma do tipo rotativa ao invés da prismática. Essa estrutura confere boa resistência mecânica, mas a exatidão do órgão terminal é inversamente proporcional ao alongamento da junta prismática horizontal.

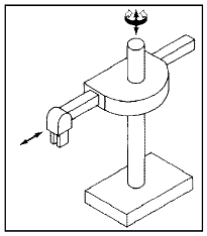


Figura 3 – Robô de geometria cilíndrica

Fonte: [13]

Este tipo de robô ajusta bem a atividades que necessitam carregar objetos não muito pesados. Porém, havendo necessidade de maior esforço, comumente se opta por utilizar acionamentos pneumáticos.

3.2.3: Esférico

Neste modelo, em relação ao cilíndrico, a segunda junta é substituída por uma do tipo rotacional, conferindo ao robô o volume de trabalho de uma semiesfera (Figura 4). Com essa mudança a resistência mecânica é prejudicada e, da mesma forma que o tipo anterior, a exatidão do órgão terminal diminui com o aumento da distância radial da última junta (prismática).

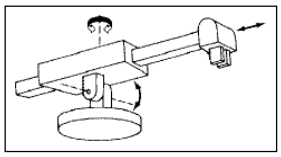


Figura 4 – Robô de geometria esférica

Fonte: [13]

A vantagem de se utilizar esse modelo é que ele confere rapidez nos movimentos com geração de trajetórias mais simples que o antropomórfico, que será visto adiante.

3.2.4: SCARA

Esta terminologia significa *Selective Compliance Assembly Robot Arm*. Tal modelo possui os mesmos tipos de junta do robô de geometria esférica, porém, os eixos de cada junta são paralelos entre si (Figura 5), garantindo alta resistência mecânica para movimentos com carga na direção vertical. Comumente se utiliza este tipo de braço para mover pequenos objetos.

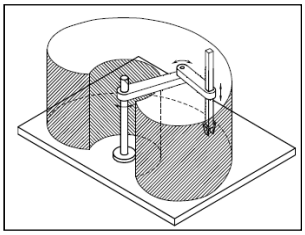


Figura 5 – Robô Scara
Fonte: [11]

3.2.5: Antropomórfico

Como o próprio nome diz, esse tipo de braço se assemelha ao do ser humano, possuindo apenas juntas rotativas. O eixo da junta da base é ortogonal aos demais (figura 6). Essa estrutura é uma das mais ágeis devido ao tipo de junta [11], mas a exatidão do posicionamento do efetuador varia dentro do volume de trabalho.

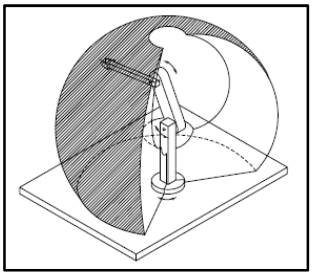


Figura 6 – Robô antropomórfico
Fonte - [11]

Até meados de 2005 essa estrutura era uma das mais utilizadas no ramo industrial. De acordo com a IFR, cerca de 59% dos robôs instalados até 2005 no mundo são deste tipo.

Os próximos itens apresentam a teoria para a cinemática de manipuladores robóticos, baseados em alguns livros e apostilas de robótica, tais como [11], [13], [14] e [15].

3.3: Movimentos de corpo rígido

Para o equacionamento da cinemática de braços robóticos, necessita-se, primeiramente, ter uma referência ou sistemas de coordenadas para representar as posições e orientações de corpos rígidos. Também é de grande relevância saber as transformações de coordenadas entre esses sistemas, para que vetores que representam posições, velocidades e acelerações, dados em um determinado sistema de coordenadas, possam ser representados em outros sistemas de coordenadas.

A seguir será apresentada a teoria básica de operações de rotação e translação entre dois sistemas de mesmo tipo, além da transformação homogênea.

3.3.1: Rotações

Dado um ponto P no espaço tridimensional, deseja-se relacionar as coordenadas deste ponto no sistema de coordenadas $O(x_1, y_1, z_1)$ em relação ao sistema fixo inercial $O(x_0, y_0, z_0)$, com representado na Figura 7.

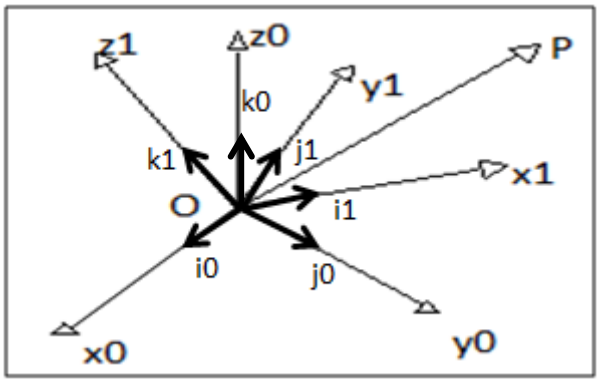


Figura 7 – Sistemas dextrogiros

Fonte – Elaborado pelo autor

Considerando os vetores unitários: (i_0, j_0, k_0) e (i_1, j_1, k_1), pode-se representar o ponto P em relação aos dois sistemas.

$$P_0 = P_{0x}i_0 + P_{0y}j_0 + P_{0z}k_0 \quad (1)$$

$$P_1 = P_{1x}i_1 + P_{1y}j_1 + P_{1z}k_1 \quad (2)$$

Como (1) e (2) representam o mesmo ponto, pode-se escrever:

$$P_{0x} = P_0 \cdot i_0 = P_1 \cdot i_0$$

$$P_{0y} = P_0 \cdot j_0 = P_1 \cdot j_0$$

$$P_{0z} = P_0 \cdot k_0 = P_1 \cdot k_0$$

Substituindo esse sistema por (2):

$$P_{0x} = P_{1x}i_1 \cdot i_0 + P_{1y}j_1 \cdot i_0 + P_{1z}k_1 \cdot i_0$$

$$P_{0y} = P_{1x}i_1 \cdot j_0 + P_{1y}j_1 \cdot j_0 + P_{1z}k_1 \cdot j_0$$

$$P_{0z} = P_{1x}i_1 \cdot k_0 + P_{1y}j_1 \cdot k_0 + P_{1z}k_1 \cdot k_0$$

ou

$$P_0 = R_0^1 P_1 \qquad (3)$$

Sendo que a matriz R_0^1 representa a matriz de rotação das coordenadas do ponto P do sistema $O(x_1, y_1, z_1)$ em relação ao sistema fixo $O(x_0, y_0, z_0)$, dada por:

$$R_0^1 = \begin{bmatrix} i_1 \cdot i_0 & j_1 \cdot i_0 & k_1 \cdot i_0 \\ i_1 \cdot j_0 & j_1 \cdot j_0 & k_1 \cdot j_0 \\ i_1 \cdot k_0 & j_1 \cdot k_0 & k_1 \cdot k_0 \end{bmatrix}$$

As colunas da matriz são os cossenos diretores do sistema 1 em relação ao fixo.

3.3.2: Composição de rotações

Na explicação anterior, apenas se considerou dois sistemas de coordenadas, como se o fixo estivesse na base do robô e o outro, no eixo da junta *shoulder* do braço. Só que um manipulador possui mais juntas, consequentemente, mais sistemas (x_n, y_n, z_n). Por isso, são necessárias as composições de matrizes de rotação.

Considerando um novo sistema $O(x_2, y_2, z_2)$, o mesmo ponto P pode agora ser representado de três formas:

$$P_0 = R_0^1 P_1$$

$$P_0 = R_0^2 P_2 \qquad (4)$$

$$P_1 = R_1^2 P_2 \qquad (5)$$

Substituindo (5) em (3):

$$P_0 = R_0^1 R_1^2 P_2$$

Desse modo, temos:

$$R_0^2 = R_0^1 R_1^2$$

Então, para se transformar as coordenadas do ponto P do sistema 2 ao fixo, deve-se primeiro transformá-lo para o sistema 1, através de R_1^2, para então, transformar para o sistema fixo com R_0^1.

Com vários sistemas, é possível usar uma forma generalizada de composições de rotações, em que:

$$R_0^n = R_0^1 R_1^2 \dots R_{n-1}^n$$

3.3.3: Transformações homogêneas

Primeiramente, considera-se uma transformação homogênea quando se agrega um parâmetro a uma matriz representativa de um vetor tridimensional [16].

Por exemplo: dado o vetor $v = (x, y, z)^T$, com o incremento de um parâmetro “d”, $\hat{v} = (x, y, z, d)^T$, $\hat{v}$ é uma representação homogênea de v.

Para completar as informações de um ponto em relação a outros sistemas de coordenadas são necessárias aquelas que dizem respeito à translação.

Considerando um sistema com apenas uma translação ao sistema fixo de origem, como na figura 8, sem rotações, cuja distância é representada pelo vetor d_0^1, sua representação homogênea será da seguinte forma:

$$P_0 = R_0^1 P_1 + d_0^1$$

ou

$$\begin{bmatrix} P_{0x} \\ P_{0y} \\ P_{0z} \end{bmatrix} = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \end{bmatrix} + \begin{bmatrix} d_{0x}^1 \\ d_{0y}^1 \\ d_{0z}^1 \end{bmatrix}$$

$$\begin{bmatrix} P_{0x} \\ P_{0y} \\ P_{0z} \end{bmatrix} = \begin{bmatrix} r11 & r12 & r13 & d_{0x}^1 \\ r21 & r22 & r23 & d_{0y}^1 \\ r31 & r32 & r33 & d_{0z}^1 \end{bmatrix} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \\ 1 \end{bmatrix}$$

Então se acrescenta mais uma linha para que a matriz seja quadrada.

$$\begin{bmatrix} P_{0x} \\ P_{0y} \\ P_{0z} \\ 1 \end{bmatrix} = \begin{bmatrix} r11 & r12 & r13 & d_{0x}^1 \\ r21 & r22 & r23 & d_{0y}^1 \\ r31 & r32 & r33 & d_{0z}^1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \\ 1 \end{bmatrix}$$

ou

$$\widetilde{P_0} = A_0^1 \widetilde{P_1}$$

Sendo que a matriz A (transformação homogênea) possui parâmetros de rotação e translação.

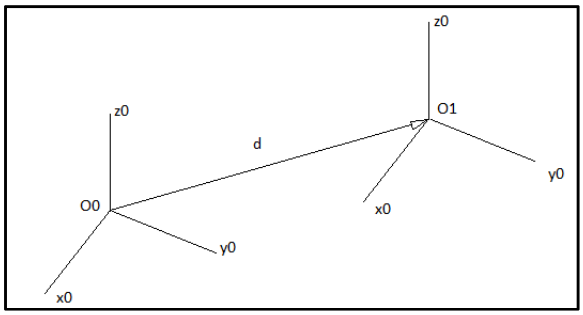


Figura 8 – Translação de coordenadas

Fonte – Elaborado pelo autor

Com esses conceitos, é possível discorrer sobre as cinemáticas de braços robóticos.

Considerando q_i como sendo a coordenada generalizada das posições de juntas (θ_i, d_i), é fundamental a relação entres esses parâmetros de juntas e o do órgão terminal. Isso é feito através da cinemática direta e inversa, representado na figura 9.

Figura 9 – Representação da cinemática direta e inversa

Fonte – Elaborado pelo autor

3.4: Cinemática Direta

O objetivo deste procedimento é encontrar os valores de posição e orientação do órgão terminal em função das variáveis de juntas. Para isso, utilizam-se as matrizes de transformação homogênea.

Um dos métodos mais conhecidos e utilizados é o de Denavit – Hartenberg, o qual fornece passos para gerar um modelo simplificado do braço com suas

dimensões e coordenadas de cada junta e do efetuador. Com base nisso, formula-se uma tabela, cujos parâmetros são utilizados para construir as matrizes homogêneas necessárias para encontrar os valores de orientação e posição.

O método é detalhado em seguida.

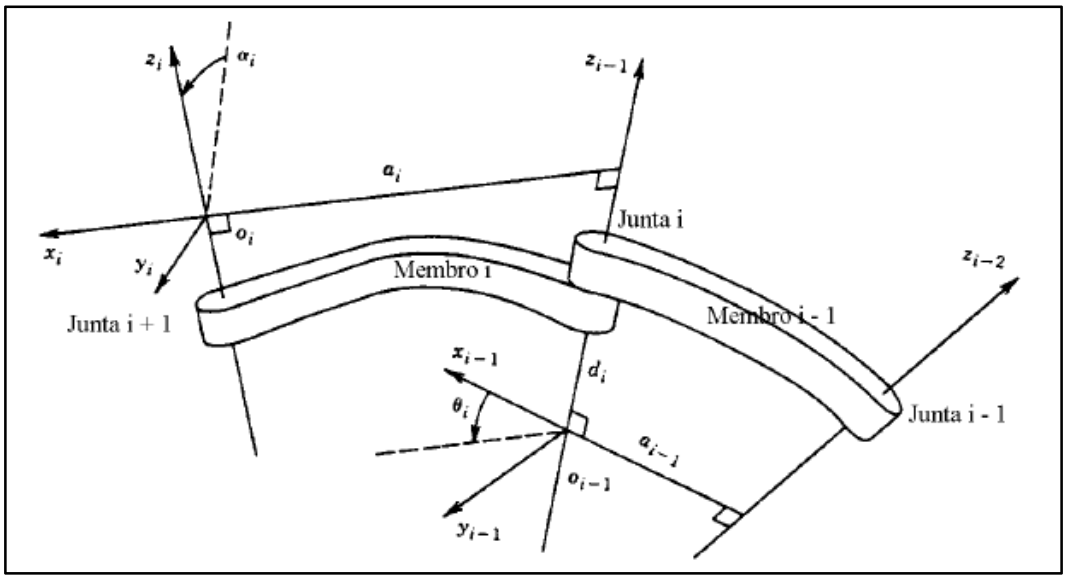


Figura 10 – Definição de coordenadas de juntas

Fonte – [15]

Considerando que z_{i-1} é o eixo da junta i, e o eixo x_i é perpendicular ao eixo z_{i-1}, apontando no sentido do afastamento desse último, interceptando o eixo z_{i-1}.

Com base na Figura 10, determina-se cada parâmetro da seguinte forma:

a_i = distância ao longo de x_i, de O_i à interseção dos eixos x_i e z_{i-1} ou a distância mais curta entre os eixos z_{i-1} e z_i;

d_i = distância ao longo de z_{i-1}, de O_{i-1} à interseção dos eixos x_i e z_{i-1};

α_i = ângulo do eixo z_{i-1} para o eixo z_i, medido em torno de x_i, com o sinal dado pela regra da mão direita;

θ_i = ângulo do eixo x_{i-1} para o eixo x_i, medido em torno de z_{i-1};

PASSO 1: Localizar e nomear os eixos das juntas, z_0, z_1, ... , z_{n-1}, podendo ser os mesmos de rotação (junta R) ou de translação (junta P). O eixo z_0 deverá apontar para o ombro. Para robôs com braço (e/ou antebraço) à esquerda ou à direita do ombro, os eixos z_1 (e/ou z_2) devem apontar no sentido do tronco para o braço.

PASSO 2: Estabelecer o sistema da base $O_0x_0y_0z_0$. Colocar O_0 em qualquer lugar sobre z_0. Os eixos x_0 e y_0 devem completar um triedro destrógiro. Colocar x_0 e y_0 em posições convenientes.

PARA i = 1, 2, ... , n-1, REALIZAR OS PASSOS 3 A 5:

PASSO 3: Localizar a origem O_i onde a normal comum a z_i e z_{i-1} (reta que contem a menor distância entre z_i e z_{i-1}) intercepta z_i. Se z_i intercepta z_{i-1}, localizar O_i nessa interseção. Se z_i e z_{i-1} são paralelos, localizar O_i na junta i.

PASSO 4: Estabelecer x_i ao longo do produto vetorial $\pm(\mathbf{z_i} \times \mathbf{z_{i-1}})$, através de O_i, ou da normal comum aos eixos z_i e z_{i-1}, quando eles forem paralelos.

PASSO 5: Estabelecer y_i de modo a completar um sistema destrógiro.

PASSO 6: Estabelecer o sistema do órgão terminal, $O_nx_ny_nz_n$, conforme fig. 3.3, onde:

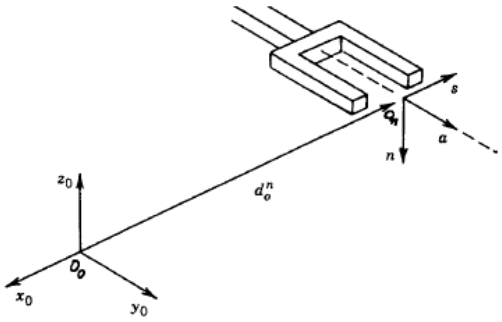


Fig. 3.3 Sistema do órgão terminal

O_n = centro do órgão terminal;
a = vetor unitário na direção de aproximação ao objeto (approach) z_{n-1};
s = vetor unitário na direção de abertura/fechamento da garra (sliding);
n = s x **a** = vetor unitário na direção normal, completa o sistema destrógiro.

Obs.: em caso de conflito da disposição acima com a condição DH2, seguir essa última.

Figura 11 – Algorítmo DH

Fonte – [15]

PASSO 7: Criar uma tabela com os parâmetros DH:

Corpo	a_i	α_i	d_i (*)	θ_i (**)
1				
2				
...				
n				

(*) variável, se junta P (**) variável, se junta R

PASSO 8: Formar as matrizes de transformação homogênea, substituindo os parâmetros DH na equação (3.5.2).

PASSO 9: Formar a matriz de transformação homogênea total, substituindo as matrizes obtidas no passo 8 na eq. (3.4.1).

PASSO 10: Formar a matriz de transformação homogênea total "direta", usando a eq. (3.4.3), a qual, após desenvolvimento, apresenta a forma

$$\mathbf{H}_0^n = \begin{bmatrix} \cos(x_n, x_0) & \cos(y_n, x_0) & \cos(z_n, x_0) & x_0 \\ \cos(x_n, y_0) & \cos(y_n, y_0) & \cos(z_n, y_0) & y_0 \\ \cos(x_n, z_0) & \cos(y_n, z_0) & \cos(z_n, z_0) & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6.1)$$

PASSO 11: Igualar as matrizes obtidas nos passos 9 e 10, obtendo as equações da cinemática direta de posição.

Figura 12 – Algorítmo DH (continuação)

Fonte – [15]

3.5: Cinemática Inversa

O processo da cinemática inversa é o contrário da direta, ou seja, a partir dos dados de posição e orientação do efetuador, deseja-se conhecer informações sobre as juntas.

Por exemplo, através da cinemática direta se obtém as seguintes equações para um determinado tipo de manipulador, conforme a Figura 13:

$$
\begin{aligned}
c_1[c_2(c_4c_5c_6-s_4s_6)-s_2s_5c_6]-s_1(s_4c_5c_6+c_4s_6) &= r_{11}\\
s_1[c_2(c_4c_5c_6-s_4s_6)-s_2s_5c_6]+c_1(s_4c_5c_6+c_4s_6) &= r_{21}\\
-s_2(c_4c_5c_6-s_4s_6)-c_2s_5c_6 &= r_{31}\\
c_1[-c_2(c_4c_5s_6+s_4s_6)+s_2s_5s_6]-s_1(-s_4c_5s_6+c_4c_6) &= r_{12}\\
s_1[-c_2(c_4c_5s_6+s_4s_6)+s_2s_5s_6]+c_1(-s_4c_5s_6+c_4c_6) &= r_{22}\\
s_2(c_4c_5s_6+s_4c_6)+c_2s_5s_6 &= r_{32}\\
c_1(c_2c_4s_5+s_2c_5)-s_1s_4s_5 &= r_{13}\\
s_1(c_2c_4s_5+s_2c_5)+c_1s_4s_5 &= r_{23}\\
-s_2c_4s_5+c_2c_5 &= r_{33}\\
c_1s_2d_3-s_1d_2-d_6(c_1c_2c_4s_5+c_1c_5s_2-s_1s_4s_5) &= d_x\\
s_1s_2d_3+c_1d_2+d_6(c_1s_4s_5+c_2c_4s_1s_5+c_5s_1s_2) &= d_y\\
c_2d_3+d_6(c_2c_5-c_4s_2s_5) &= d_z
\end{aligned}
$$

Figura 13 – Equações de cinemática direta

Fonte – [15]

As variáveis da direita da equação representam os valores de posição e orientação do órgão terminal. Os valores r_{ij} são elementos da matriz de rotação de uma transformação homogênea e d_x, d_y e d_z são elementos relativos à translação do sistema de coordenadas em relação ao sistema inercial (base).

As letras ‘c’ e ‘s’ representam os cossenos e senos, respectivamente. E os números de 1 a 6 representam a junta em questão.

Enquanto na forma direta sempre se obtém uma única solução, na inversa pode se encontrar uma, várias ou nenhuma solução, além de ser muito difícil de resolver o equacionamento, já que se trata de um sistema não linear.

Porém, existem algumas saídas. É possível solucionar numericamente ou geometricamente, sendo este último, o mais simples e intuitivo, o qual será utilizado neste trabalho e detalhado no capítulo 5.

O fato de poder existir múltiplas soluções implica na existência de várias configurações possíveis do braço para posicionar o terminal em um ponto. Quanto maior o número de graus de liberdade, maiores são as possibilidades.

Capítulo 4: Metodologia de desenvolvimento de produto

4.1: Problematização

Atualmente existem muitos produtos e projetos destinados à robótica educacional, porém poucos dão a devida ênfase aos métodos utilizados para mover um robô real (industrial), apresentando poucos detalhes de seu funcionamento ou de como é programado para realizar certos movimentos.

Em vista disso, este projeto busca contribuir nesse ramo com o desenvolvimento de um robô manipulador capaz de popularizar a ciência e a tecnologia, auxiliando no ensino fundamental, técnico e superior. Isto pode ocorrer pela utilização do robô como um meio facilitador para outras atividades educativas ou através da programação, incluindo funções similares aos braços robóticos industriais. É importante deixar claro em sua programação os detalhes conceituais de robótica fundamentais para certos movimentos, como os de tipo retilíneos.

O Trabalho se inicia de fato com uma pesquisa de mercado, detalhada no tópico seguinte. A seguir são realizadas análises dos produtos para o levantamento de requisitos e geração de soluções para o projeto, além da realização de testes e avaliação final do trabalho.

4.2: Análise sincrônica

Esta etapa faz parte da fase de análise da metodologia de Bonsiepe [17], cujo objetivo é levantar informações sobre os produtos similares ao que se pretende projetar, que estão disponíveis no mercado, e outros que são projetos acadêmicos. Desta análise são obtidos dados importantes para o desenvolvimento do produto no que tange à especificação de requisitos, destacando suas principais características, qualidades e defeitos. Para isso foram analisados três kits de robótica, com a possibilidade de se construir diversas estruturas, e três kits com estrutura bem definida – braço robótico.

Os seguintes kits foram analisados: Lego Mindstorms EV3 [18]; Linxmotion AL5D [19]; Modelix kit 411 [20]; RoboFácil [21]; Hajime [22] e EasyArm DS [23]. A seguir são apresentados os detalhes de cada um.

4.2.1: Lego Mindstorms EV3

O kit da *Lego* escolhido para a análise de produtos similares foi o Mindstorms EV3, que é a última geração desse modelo e é mundialmente conhecido devido à sua própria marca e a sua inovação no ramo de brinquedos, chamando a atenção de jovens e adultos interessados no tema da robótica. Ele vem com 601 peças, 4 portas de entrada para sensores e 4 de saída para motores, 3 motores, software de programação e inúmeras outras funcionalidades. O quadro 1 apresenta algumas características deste modelo.

Quadro 1 – Análise Kit Lego

Informações gerais
Nome: Lego Mindstorms EV3
Fabricação: Estrangeira
Material: Plástico
Preço: R$ 2799.99 (kit completo)
Informações técnicas
Arquitetura: Fechada
Linguagem de programação: Visual
Microprocessador: ARM 32 bits **Nº de portas (sinal de saída):** 4 **Nº de portas (sinal de entrada):** 4
Estrutura: Modular

Fonte - [18]

Apesar de ele vir com um número bem limitado de atuadores e transdutores, suas peças modulares permitem que se criem uma infinidade de estruturas, inclusive braços robóticos. A partir de um ambiente de programação visual, semelhante ao *LabView*[2], permite-se criar rotinas de movimentações bastante interessantes e até complexas.

Essas características o tornam uma ferramenta que pode ir muito além de um simples brinquedo, sendo inserido nas escolas e universidades como um meio facilitador de aprendizagem.

Como vantagens pode-se citar a facilidade de se programar, já que a linguagem é intuitiva e visual – ícones; possibilidade de se comunicar com outros módulos *EV3*; além de rodar o programa sem estar conectado a um computador.

Em contrapartida a arquitetura, tanto de hardware como de software, é fechada, o custo é relativamente alto em comparação aos outros produtos similares, e há uma grande limitação na quantidade de transdutores e motores que podem ser utilizados.

4.2.2: Modelix kit 411

Dentre os modelos nacionais, pode-se dizer que este produto é um dos maiores concorrentes da *Lego Mindstorms* devido à grande quantidade de peças, motores e sensores que são disponibilizados, além do software de programação que é bastante intuitivo.

O produto foi desenvolvido com foco nos alunos de ensino fundamental e médio, sendo interessante a sua utilização no ensino superior.

O quadro 2 apresenta algumas informações sobre este kit.

[2] LabView – Software da empresa *National Intruments*, muito utilizado na medição ou controle de sistemas eletroeletrônicos através de uma interface de programação visual.

Quadro 2 – Análise kit Modelix

Informações gerais
Nome: Modelix – kit 411
Fabricação: Nacional
Material: Alumínio e plástico
Preço: R$ 2390
Informações técnicas
Arquitetura: Fechada
Linguagem de programação: Visual - fluxograma
Microcontrolador: Atmel (modelo não disponibilizado) **Nº de portas (sinal de saída):** 14 digitais (mesma utilizada para entrada de sinal digial) **Nº de portas (sinal de entrada):** 6 analógicas; 14 digitais
Estrutura: Modular

Fonte - [20]

Como não há estrutura física pré-definida, considera-se esse produto com uma alta flexibilidade, sendo que suas peças devem ser conectadas com porcas e parafusos. Isto torna a sua montagem um pouco mais dificultosa em relação ao kit da *Lego* descrito anteriormente.

O software de programação possui plataforma fechada. A linguagem utilizada é do tipo visual, baseada em fluxograma, tornando-o bastante interessante para fins didáticos, uma vez que a base da construção de programas ou softwares pode aproveitar as informações dispostas em um fluxo de dados.

Além disso, o kit vem com um CD que inclui diversos cursos que ensinam a base para se utilizar microcontroladores, aprender alguns conceitos de robótica, e também, manuais para a utilização do software e de modelos de projetos que podem ser feitos com o kit.

Como vantagens, verifica-se: alta flexibilidade, bons recursos de apoio com manuais e cursos, software com linguagem de programação bastante intuitiva, possibilidade de se construir um braço robótico com os recursos disponíveis, entre outros

Como desvantagem: tanto a plataforma de hardware como a de software é fechada, o custo é relativamente alto e o software perde licença de uso após um ano.

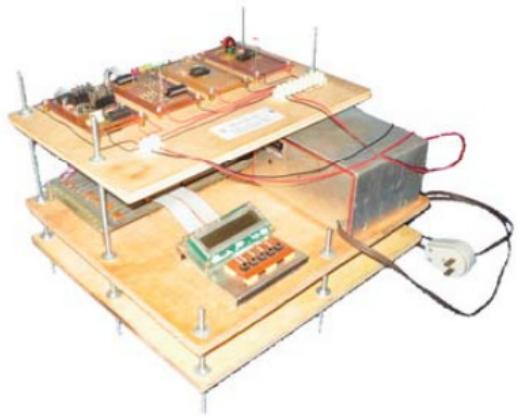
4.2.3: RoboFácil

Este kit de desenvolvimento não é comercializado, no entanto, por ser um projeto resultante de uma dissertação de mestrado, possui todo o seu desenvolvimento bem detalhado. Isto possibilita utilizá-lo como referência para a construção de projetos similares.

O kit RoboFácil não vem com atuadores e sensores, apenas com a placa de desenvolvimento (hardware) e um programa baseado em *VPL (Visual Programming Language)*. Nele estão incluídos alguns *plugins*, que são circuitos externos que permitem a comunicação da placa com o mundo externo. No total são quatro: controle de motores de passo, sensor de luminosidade, sensor de temperatura e *LEDs* .

Algumas de suas características podem ser vistas no quadro a seguir.

Quadro 3 – Análise projeto RoboFácil



Informações gerais
Nome: RoboFácil
Fabricação: Nacional
Material: NI (Não informado)
Preço estimado: R$ 313,30
Informações técnicas
Arquitetura: Aberta (hardware)
Linguagem de programação: Visual
Microcontrolador: Intel MCS-51 **Nº de portas (sinal de saída):** 16 **Nº de portas (sinal de entrada):** 16
Estrutura: NA (Não aplicável)
Observação: Produto não comercializado

Fonte - [21]

Pelo fato do projeto RoboFácil ser um protótipo, sua estrutura física ficou demasiado grande, não sendo interessante para aplicações em robótica embarcada.

Quanto ao software, a programação se assemelha bastante com o programa *LabView*, em que existem duas interfaces: uma é onde o usuário escolhe e dispõe os dispositivos a serem utilizados; a outra é onde se constrói o fluxo de atividades a serem executadas, tornando-o intuitivo e de fácil programação. Porém o software não é *open-source*[3].

[3] Open-source – Código aberto.

Esse kit de desenvolvimento tem como vantagem: arquitetura de hardware disponível; custo baixo; programação intuitiva e de fácil aprendizado; projeto do produto bem detalhado.

Como desvantagem verifica-se: estrutura física demasiado grande; não contém manual de instrução; não vem com materiais para montar a estrutura física de qualquer outro projeto nem com transdutores e atuadores.

4.2.4: Linxmotion AL5D

Uma das especificações para o projeto em questão é que ele deve ter a estrutura de um braço com juntas rotativas. Com isso, buscaram-se modelos com características semelhantes.

O quadro a seguir apresenta este modelo com suas características.

Quadro 4 – Análise do produto Lynxmotion

Informações gerais
Nome: Lynxmotion AL5D
Fabricação: Estrangeira
Material: Alumínio
Preço: R$ 1000,00 aproximadamente
Informações técnicas
Arquitetura: Fechada
Linguagem de programação: NA
Microcontrolador: NI **Nº de portas (sinal de saída):** 32 (digital) **Nº de portas (sinal de entrada):** 4 (digital ou analógico)
Estrutura:Não Modular

Fonte - [19]

A empresa *Lynxmotion* é especializada em fabricar produtos de alto valor agregado com especialização em robótica. O modelo em questão possui uma estrutura rígida sem flexibilidade, com 6 servo motores e 4 graus de liberdade, com a opção de adquiri-lo com ou sem placa de controle da própria empresa. A placa permite o controle de 32 atuadores.

Além disso, a empresa disponibiliza dois tipos de softwares que podem ser comprados separadamente: o primeiro, chamado de *FlowBotics Studio,* utiliza a programação de tipo visual, em que o usuário movimenta um braço virtual, podendo criar rotinas de repetição. O segundo simula um *Teach Pendant*[4], comumente usado no meio industrial, sendo possível gravar diversas posições e depois fazer com que o robô percorra a trajetória criada através desses pontos.

As vantagens observadas foram as seguintes: opção de software que permite controlar o robô de forma visual, sendo possível que crianças sem grandes conhecimentos de lógica de programação pudessem controla-lo; opção de software que simula um *Teach Pendant*, o qual é muito utilizado na indústria, além de ser possível visualizar no programa conceitos relevantes no que tange à robótica, como valores de torque nos atuadores, volume de trabalho, e etc.; robustez física; grande número de portas de saída da placa de controle da empresa, sendo possível utilizar mais de um robô com a mesma placa;

A desvantagem é que a arquitetura de hardware e software é fechada; o circuito eletrônico e os fios são expostos ao ambiente; o custo é relativamente alto e não há flexibilidade para montar outras estruturas.

4.2.5: Hajime

O kit Hajime faz parte de um projeto não comercializado, possuindo sua arquitetura de hardware aberta.

[4] *Teach Pendant* – Trata-se de um painel de acionamento e controle de robôs.

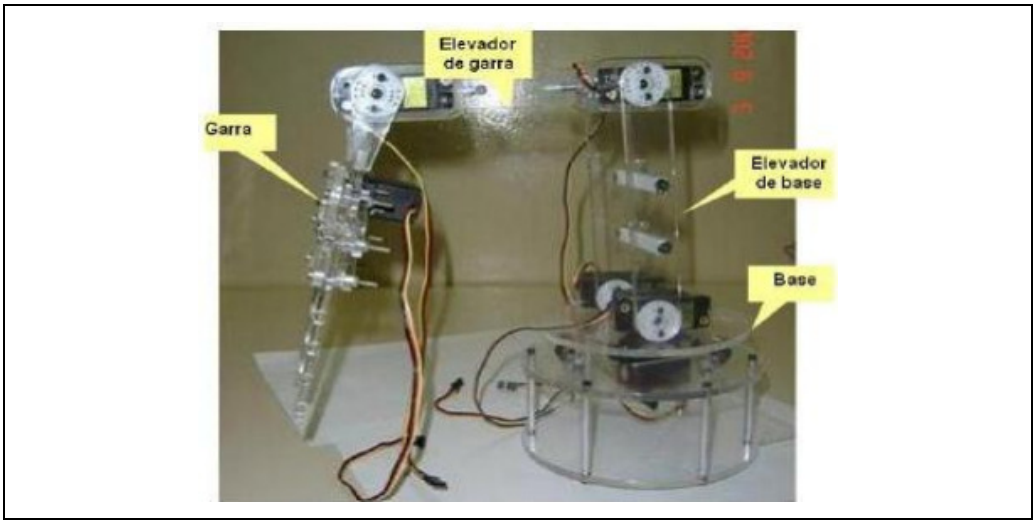
Trata-se de um braço robótico com seis juntas rotativas, tendo sua estrutura física construída a partir de materiais sucateados e não utiliza circuitos eletrônicos externos ao computador para comandar os motores. Toda a comunicação é feita via porta paralela.

Para o controle dos motores é utilizado um software com dois níveis de abstração. O primeiro foi planejado para que crianças pudessem manuseá-lo, tendo como linguagem do tipo icônica, em que é possível escolher quatro funções para cada atuador: adicionar, mover, parar ou carregar. O argumento dessas funções são valores de ângulos. No segundo nível de abstração também é possível movimentar os atuadores individualmente. Neste caso a resolução dos passos do motor é melhor (0,5º).

Como opção, ainda é possível controlar o robô via *joystick.*

Algumas informações sobre este robô são descritas no quadro seguinte.

Quadro 5 – Análise do projeto Hajime



Informações gerais
Nome: Hajime
Fabricação: Nacional
Material: Plástico (policarbonato)
Preço: NI
Informações técnicas
Arquitetura: Aberta (Hardware)
Linguagem de programação: Icônica
Microcontrolador: NA **Nº de portas (sinal de saída):** NA **Nº de portas (sinal de entrada):** NA
Estrutura: Não modular
Observação: Não utiliza microcontrolador, a comunicação é feita via porta paralela do conputador.

Fonte - [22]

Devido à limitação de funcionalidades do software, o mesmo deixa a desejar na questão do aprendizado de conceitos de robótica para jovens do ensino médio e superior. No entanto, o protótipo exige que os alunos tenham um conhecimento básico de eletrônica para poderem utilizar a porta paralela, a qual deve ser conectada com os motores.

O grupo desenvolvedor do projeto não divulgou os valores investidos.

Segundo Sasahara [22] Sasahara, L. R.; Cruz, S. M. S. Hajime: Uma Nova Abordagem em Robótica Educacional. Artigo. Rio de Janeiro: Anais do XXVII Congresso da SBC, 2007., as vantagens são: “arquitetura de hardware aberta; design e construção simples; materiais de fácil manuseio e baixo custo; não utiliza circuitos eletrônicos de processamento externo para executar os movimentos”. E ainda pode-se dizer que a interface com o usuário é bastante simples, podendo ser utilizada por crianças que não possuem conhecimentos de eletrônica.

As desvantagens observadas são: o software não é *open-source*; necessita-se de conhecimentos básicos de comunicação via porta paralela; limitação do software quanto às funcionalidades.

4.2.6: EasyArm DS

Este produto não se enquadra como um kit de robótica educacional, mas sua estrutura e componentes permitem a criação de um módulo com fins didáticos.

Sua estrutura geométrica é do tipo articulado ou antropomórfico com quatro graus de liberdade, sendo que os três primeiros motores, partindo da base, servem para posicionar o órgão terminal no espaço 3D e o último, orientar o efetuador no espaço tridimensional.

O conjunto vem com dois motores de passo que controlam a posição da base, três servomotores para o controle das outras juntas e a abertura e fechamento do órgão terminal e sensores ópticos para os motores de passo que servem para referenciá-los (*set point*).

Não está incluso placas controladoras nem software de programação dos atuadores.

Algumas informações sobre o produto são apresentadas no quadro seguinte.

Quadro 6 – Informações do braço EasyArm DS

Informações gerais
Nome: EasyArm DS
Fabricação: Nacional
Material: MDF
Preço: R$ 499
Informações técnicas
Arquitetura: NA

Linguagem de programação: NA
Microcontrolador: NC **Nº de portas (sinal de saída):** NA **Nº de portas (sinal de entrada):** NA
Estrutura: Não modular
Observação: Não inclui o microcontrolador. Toda estrutura eletrônica deve ser montada pelo usuário.

Fonte - [23]

Apesar de se utilizar motores mais robustos na base do robô, tornando-o capaz de levantar cargas mais pesadas em relação aos outros braços vistos anteriormente, o atuador do órgão terminal é um servomotor comumente utilizado em aeromodelismo. É necessário apenas um valor de referência – ângulo – para que ele posicione seu eixo. Ou seja, no modo como foi implantado, não há controle sobre a pressão exercida sobre um possível objeto que esteja neste órgão, criando uma grande limitação quanto à variação do peso da carga.

As vantagens observadas foram: estrutura física de material resistente, barato e de fácil fabricação; pequeno número de juntas, o que facilita o controle do braço como um todo; custo baixo em relação ao modelo *Lynxmotion*, sendo que este vem com mais recursos; possibilidade de utilizar diversos tipos de controladores a critério do usuário.

Como desvantagens verificou-se: modularidade nula; não inclui software para controle dos motores; são exigidos bons conhecimentos em eletrônica e software para utilizá-lo como um kit didático.

Com base nas características de cada modelo, é necessário compará-los através de um quadro contendo alguns atributos com seus respectivos pesos, já que cada um deles tem graus de relevância distintos.

Os seguintes atributos foram comparados:

- Fabricação: podendo ser nacional ou internacional. O maior peso vai para os produtos nacionais, já que os mesmos podem ter custo mais baixo e são de

grande importância para o desenvolvimento econômico e incentivo à pesquisa no país;

- Material: MDF, plástico(PLA) ou alumínio. O MDF possui maior importância devido ao seu custo (R$/m²) ser baixo em relação ao alumínio[5] e de fácil usinagem. Apesar de o PLA ser o mais caro, possui grande flexibilidade para a criação de peças complexas através da impressão 3D, ficando com o peso intermediário.

- Preço: baixo para valores menores que R$ 500,00, médio para valores entre R$ 501,00 e R$ 1500,00 e alto para valores acima de R$ 1501,00.

- Arquitetura: aberta ou fechada [24]. Somente foi levada em consideração a arquitetura de hardware, já que não se obtiveram informações sobre as de software para todos os modelos;

- Linguagem de programação: visual – fluxograma, icônica ou convencional. Neste caso considera-se icônica aquela em que não se utiliza fluxogramas e não é convencional – aquelas que necessitam digitar textualmente o comando;

- Estrutura: Modular ou não modular. Modular é quando o produto pode assumir várias configurações estruturais – formas [28];

Com esses atributos, montou-se um quadro conceitual que indica os pesos para as variações de cada atributo, apresentados em seguida. O valor 3 indica maior importância entre as possibilidades de cada atributo e o 1, o menor. Os produtos que não apresentam alguma informação são representados no quadro com o símbolo "---".

Quadro 7 – Quadro de análise de atributos

	Pesos	**3**	**2**	**1**

[5] Segundo [25], o valor do MDF de 3mm de espessura é de 11,59 R$/m², enquanto que, segundo [26] e [27], uma chapa de alumínio de mesma espessura é em torno de 32 R$/m².

Gerais	**Fabricação**	Nacional	Não comercializado	Importado
	Material	MDF	Plástico	Alumínio
	Preço	Baixo	Médio	Alto
Técnicos	**Arquitetura**	Aberta	---	Fechada
	Linguagem	Visual - fluxograma	Icônica	Convencional
	Estrutura	Modular	---	Não modular

Fonte – Elaborado pelo autor

Com base nesse quadro, montou-se outro que relaciona os produtos com os atributos e seus respectivos pesos, os quais indicam o grau de relevância de cada atributo.

Por haver kits com características bem distintas, resolveu-se separá-los em kits sem forma definida, que são os da *Lego, Modelix e RoboFácil*, e kits com forma de braços robóticos do tipo antropomórfico.

Na penúltima linha é calculada a soma dos valores para cada produto, considerando os pesos. Para os produtos que não contêm informação em determinado atributo, este não é levado em consideração na soma.

Por último é feita uma classificação, em que o melhor é aquele cuja pontuação é maior. O seguinte quadro apresenta essas relações de produtos, atributos e pesos.

Quadro 8 – Comparação dos produtos com a base da análise

		Kits sem forma definida				**Kits de braços robóticos**		
Gerais	Peso	Nome	Lego	Modelix	RoboFácil	Lynxm.	Hajime	EasyArm
	1	Fabricação	1	3	2	1	2	3
	1	Material	2	1	---	1	2	3
	3	Preço	1	1	3	2	---	3
Técnicos	3	Arquitetura	1	1	3	1	3	---
	3	Linguagem	2	3	2	---	2	---
	2	Estrutura	3	3	---	1	1	1
	-	Soma	21	25	26	13	21	17
Classificação			**3º**	**2º**	**1º**	**3º**	**1º**	**2º**

Fonte – [18], [20], [21], [19], [22], [23]

Nessa análise obtêm-se dois melhores classificados, um entre os kits sem forma definida e outro com a estrutura de um braço robótico antropomórfico.

Para cada um dos melhores é feita uma lista de verificação, cujo objetivo é pontuar os pontos positivos e negativos, para, posteriormente, levantar os requisitos do projeto.

4.3: Lista de verificação

Os quadros seguintes mostram as qualidade e fraquezas dos produtos/projetos com melhor classificação que foram os produtos RoboFácil e Hajime.

Os pontos positivos apresentam características que podem ser mantidas ou melhoradas no produto a ser desenvolvido e os pontos negativos, que devem ser evitados.

Quadro 9 – Qualidades e fraquezas do RoboFácil

RoboFácil	
Pontos positivos	**Pontos Negativos**
Arquitetura de hardware aberta	Estrutura física muito grande
Baixo custo	Sem manual de instrução
Programação intuitiva	Sem componente para montar o robô
Material do projeto bem detalhado e disponível	Sem transdutores
	Sem motores

Fonte – [21]

No projeto RoboFácil, o maior destaque está na disponibilização da arquitetura de hardware, com descrições de como foi implementado o circuito eletrônico principal, os *displays*, portas paralelas, porta serial, conversores DAC e ADC, plug-ins de controle dos motores de passo, *led* e sensores.

Em contrapartida, a estrutura possui grandes dimensões comparadas às outras e não possui um modelo de robô como demonstração que possa ser utilizado no sistema.

Quadro 10 – Qualidades e fraquezas do Hajime

Hajime	
Pontos positivos	**Pontos Negativos**
Arquitetura de hardware aberta	Software não é livre
Design e construção simples	Software com poucas funcionalidades
Materiais de fácil acesso	Necessidade de conhecimentos de comunicação paralela
Baixo custo	Material deve ser adquirido pelo usuário
Não utiliza circuitos eletrônicos de processamento externo	
Interface de controle simples	

Fonte – Elaborado pelo autor

No Hajime, por se tratar também de um projeto no estado da arte, o usuário deve buscar todo o material para a construção do braço robótico, além de utilizar a porta paralela, que vem entrando em desuso. Esses pontos são os que mais se destacam entre os negativos. Porém, como vantagem, o Hajime não possui circuito eletrônico e a interface é simples e intuitiva, podendo ser utilizada nos níveis de ensino mais básicos.

4.4: Lista de necessidades

Com base nos itens dos quadros anteriores, levanta-se uma lista de necessidades explicitando as características de cada modelo que podem ser mantidas e aquelas que podem ser aperfeiçoadas. Considerando-se, também, o acréscimo de novas funcionalidades para cada um dos melhores modelos de cada categoria – RoboFácil e Hajime.

Quadro 11 – Quadro de necessidades do RoboFácil

RoboFácil	
Manter	Arquitetura de hardware aberta
	Material do projeto bem detalhado e aberto
	Baixo custo
Aperfeiçoar	Estrutura física: torna-la compacta e leve
Acrescentar	Manual de instrução
	Disponibilidade de componentes para a montagem do braço
	Disponibilidade de motores
	Disponibilidade de sensores

Quadro 12 – Quadro de necessidades do Hajime

Hajime	
Manter	Arquitetura de hardware aberta
	Baixo custo
	Design e construção simples
Aperfeiçoar	Software com código aberto
	Encaixes da estrutura (eixo dos motores com os elos) com restrições mecânicas
Acrescentar	Placa de controle
	Funcionalidades específicas para movimento do robô através de linguagem de programação convencional
	Comunicação serial
	Material, tanto estrutural como atuadores
	Outras funcionalidades ao sistema

Fonte – Elaborado pelo autor

Com base nos quadros de necessidades formulam-se, primeiramente, os requisitos de forma mais abrangente, que são:

- Ter arquitetura de hardware/software aberta;
- Ter Baixo custo;
- Possuir estrutura física compacta, leve e resistente;
- Vir com manual de instrução;

- Ter funcionalidades para o aprendizado em programação e robótica;

4.5: Especificação de requisitos estruturais e técnicos

Nesta etapa, com base na lista de necessidades, analisam-se como aqueles pontos podem ser colocados em prática. Os resultados disso são apresentados nos requisitos de projeto conforme o quadro seguinte, através de especificações estruturais e técnicas.

Quadro 13 – Especificação de requisitos

	Requisitos	**Objetivo**	**Unidade**	**Classificação**
Estruturais	Juntas	3	nº	Obrigatório
	Material da junta	ABS		Desejável
	Elos	2	nº	Obrigatório
	Material do elo	MDF		Desejável
	Base	13x10x4	cm	Desejável
	Material da base	MDF		Desejável
	Garra (órgão terminal)	1	nº	Obrigatório
	Material da garra	ABS		Desejável
	Acoplamentos motor/elo/junta	5	nº	Obrigatório
	Material do acoplamento	ABS		Desejável
	Preço total desejável	300,00	R$	Desejável
Técnicos	Documentação com detalhes do projeto de hardware e software	1	nº	Desejável
	Plataforma de desenvolvimento	1	nº	Obrigatório
	Leitor cartão SD	1	nº	Obrigatório
	Servomotor	5	nº	Obrigatório
	Motor de passo	2	nº	Desejável
	Driver motor de passo	2	nº	Desejável
	Programa de escrita e envio de comandos	1	nº	Obrigatório
	Programa simulador de *Teach Pendant*	1	nº	Desejável

Fonte – Elaborado pelo autor

4.6: Especificação de requisitos de software

Para a especificação do software é utilizada parte da metodologia de Waslawick [29] referente à especificação de requisitos de sistemas de informação.

Os requisitos funcionais expressam as funções principais do sistema, já os suplementares declaram as restrições tecnológicas ou lógicas do sistema, sendo organizadas em atributos de implementação, interface e empacotamento.

Quadro 14 – Requisitos funcionais e suplementares

Requisitos funcionais

- Identificar função.
- Executar função.
- Controlar motores.

Requisitos suplementares

1. Usabilidade: “quais fatores humanos estão envolvidos no sistema? Que tipo de ajuda o sistema vai prover? Quais as formas de documentação ou manuais estarão disponíveis? Como esses manuais vão ser produzidos? Que tipo de informação eles vão conter? ”(Waslavick)

O sistema, se possível, virá com um manual de instrução, apresentando como se deve montar a estrutura física, as conexões eletrônicas e uma breve explicação das funções do software.

2. Implementação: “Qual linguagem dever ser usada? Por que motivo? Quais bibliotecas serão disponíveis? Quais bancos de dados serão acessíveis?”(Waslavick)

2.1. Linguagem ACL

O projeto do braço robótico foi inspirado no modelo Scorbot-ER VII [10], cuja linguagem de programação é o ACL que foi desenvolvida pela empresa *Eshed Robotec.*

Baseada na linguagem *Basic*, a qual foi desenvolvida com propósitos educa-

tivos, possui poucas funções e é de alto nível, tornando-se propícia para os fins desse projeto.

2.2. Linguagem C

Utilizada para programar o microcontrolador. A justificativa do uso desta linguagem é apresentada no capítulo seguinte.

3. Interface: "Como deve ser a interface? Vai ser seguida alguma norma ergonômica?" (Waslavick)

A interface conterá duas janelas, uma será o *Hyper Terminal*[6] e a outra, um editor de texto para digitar o código na linguagem ACL.

4. Empacotamento: "De que forma o software deve ser entregue ao usuário final?"(Waslavick)

Os arquivos executáveis e o código fonte do *firmware* devem estar disponíveis na internet.

A seguir são detalhados os requisitos funcionais.

[6] Hyper Terminal: antigo aplicativo de comunicações do Sistema Operacional Windows. Ele é utilizado para o envio de comandos para o microcontrolador via USB – Universal Serial Bus.

- **Identificar Função**
  Descrição:

O usuário digita no terminal ou o sistema lê do arquivo texto o comando desejado em ACL com seus comandos desejados. Ao clicar na tecla *Enter*, o sistema faz uma busca na memória do microcontrolador para verificar se a função existe e se foi escrita corretamente.

Usuário:

Não existem tipos diferentes de usuário, somente aquele que digita no terminal ou que escreveu no código texto.

Informações de entrada:

Nome da função e seu(s) argumento(s).

Informação de saída:

Um valor inteiro que identifica a função gravada na memória do sistema.

Restrições lógicas:

Enviar mensagem de erro ao digitar uma função inexistente ou incorreta.

- **Executar função**
  Descrição:

O sistema recebe o valor correspondente da função escolhida e a invoca passando os parâmetros fornecidos.

Usuário:

Não há.

Informações de entrada:

Valor correspondente da função e os parâmetros da mesma

Informação de saída:

Depende da função. Existem funções que apenas gravam algum dado, fornecem algum valor a alguma variável existente, listam no terminal as variáveis e seus valores, ativam algum motor, entre outras.

Restrições lógicas:

Enviar mensagem de erro caso seja digitado parâmetros inválidos, ou seja, que não estão de acordo com o que a função permite receber.

- **Mover motores**

  Descrição:

  O motor escolhido no projeto foi o servo-motor. A justificativa dessa escolha é descrita no capítulo seguinte.

  No caso do servo-motor, move-se o seu eixo referenciando-o com um valor de ângulo que varia de 0 a 180 graus.

  Usuário:

  Aquele de digitou no terminal ou escreveu no editor de texto.

  Informações de entrada:

  Ângulo de referência.

  Informações de saída:

  Não há.

  Restrições lógicas:

  O valor de referência não deve ser menor ou igual a 0 nem maior ou igual a 180 para não sobrecarregar o motor.

4.7: Análise funcional

Neste ponto, é feito o detalhamento de cada requisito funcional, conforme as figuras seguintes. Pelo fato de se apresentar como as funções irão funcionar, pode-se considerar essa etapa como sendo a de Anteprojeto segundo a metodologia de Bonsiepe.

Esta análise serve para reconhecer e compreender as características de uso de um produto, sendo feita, através de um diagrama sequencial, uma descrição das funções e procedimentos de uso geral do sistema, atentando para o objetivo global que é mover o braço robótico.

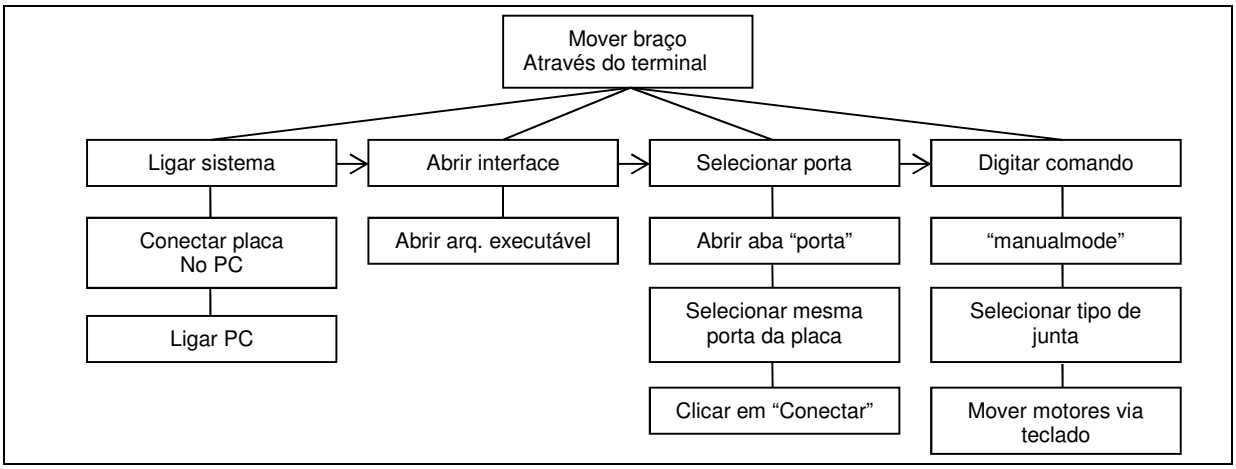


Figura 14 – Fluxo de ações e funções (mover via terminal)

Fonte – Elaborado pelo autor

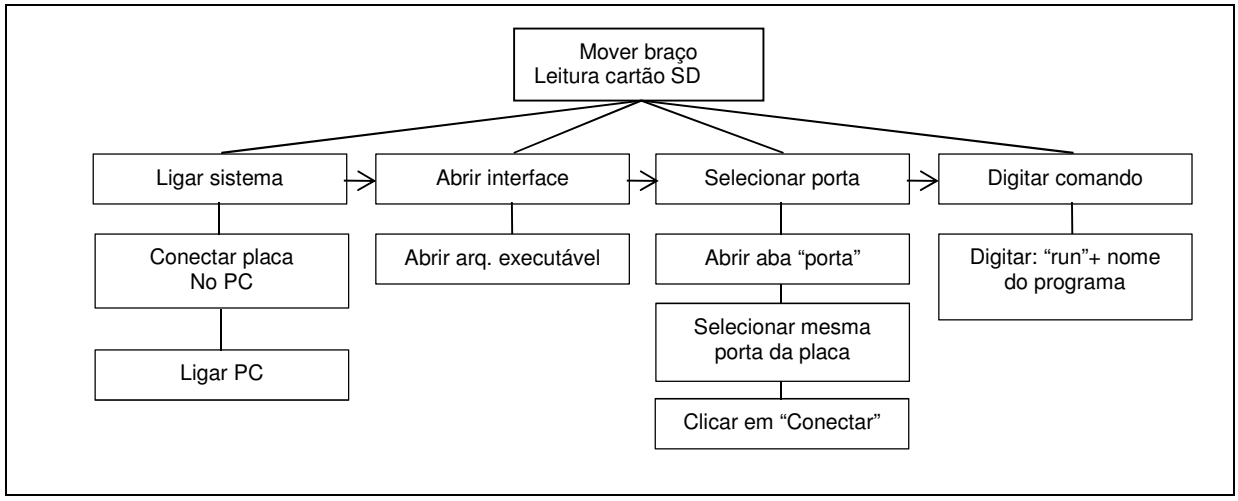


Figura 15 – Fluxo de ações e funções (mover via leitura SD)

Fonte – Elaborado pelo autor

Existem duas possibilidades para alcançar o principal objetivo: através de um terminal (Figura 14), em que o usuário move cada motor via teclado ou através da leitura de um código pré-definido em um arquivo de texto do software que é armazenado em um cartão SD (Figura 15).

4.8: Geração de alternativas

Da mesma forma que foram avaliados diferentes produtos e projetos de robótica educacional (análise sincrônica), torna-se fundamental avaliar as diferentes partes que irão compor os subsistemas do projeto em questão, levando em conta as características dos mesmos e os requisitos do produto.

Assim, foram analisadas as possibilidades para as quatro principais partes do braço manipulador: Plataforma de prototipagem eletrônica, que faz parte do robô; motor elétrico; IDE e linguagem de programação.

4.8.1: Placa microcontroladora do robô

Ao se falar em projetos de eletrônica, pensa-se de imediato em placas de circuito impresso com inúmeros componentes e conexões complexas, em que se necessita de um vasto conhecimento teórico para poder entender o mínimo do que

existe ali, podendo levar longos períodos para a materialização e elaboração de testes para um simples protótipo.

Em vista desses desafios, alguns grupos de pesquisadores buscaram criar meios que permitissem que usuários, leigos no assunto, pudessem criar seus projetos de forma mais simples e rápida a um custo relativamente baixo. Um dos grandes pioneiros nesse quesito foi a empresa norte-americana *Parallax*. Ela desenvolveu uma plataforma na década de 90 [30] com um interpretador em linguagem *Basic*, portas de entrada e saída e uma entrada para alimentação. Posteriormente, outros grupos aderiram à ideia, como a *Wiring* [31], *Beaglebone* [32], *Raspberry Pi* [33] e o tão conhecido *Arduino* [34].

Os modelos comparados no quadro seguinte possuem características distintas, porém, o *Arduino* leva uma grande vantagem devido a sua difusão no mercado brasileiro, com vários modelos nacionais, já que sua plataforma de hardware é aberta, sendo possível criar e vender placas semelhantes sem a necessidade de pagamentos pelo uso de patente.

Quadro 15 – Comparação de preços de plataformas de desenvolvimento

Nome	**Arduino**	**Beaglebone**	**Raspberry Pi**
Modelo	Mega 2560	Rev B	Model B+
Preço	U$ 40,00	U$ 45,00	U$ 30,00
Processador	ATmega2560	ARM Cortex – A8	ARM Cortex – A6
Freq. *Clock (max.)* *MHz*	16	800	400
RAM	8 KB	512 MB	512 MB
GPIO	70	65	40

A primeira versão do braço robótico não utilizará *encoders* ópticos, sendo desnecessária a utilização de microcontroladores com alta frequência de *clock*. Com isso, a Plataforma Arduino Mega se torna suficiente para a aplicação.

4.8.2: Acionamento das juntas

Parte indispensável que compõe os mecanismos robóticos são os motores, podendo ser de acionamento elétrico, hidráulico ou pneumático, cada um deles tem sua particularidade, com vantagens para determinadas aplicações.

Na imagem seguinte, comparam-se os acionamentos hidráulicos (H), pneumáticos (P), conjuntos de motor elétrico-fuso (M) e motor de passo (S), de acordo com a força e velocidade de atuação.

Neste projeto, por ter fins didáticos, não se necessita de grande esforço e alta velocidade para movimentar cargas, sendo, portanto, o mais adequado os motores de passo, de acordo com a Figura 16. Porém, dentre os motores elétricos, existem outros que podem se adequar ainda mais ao projeto.

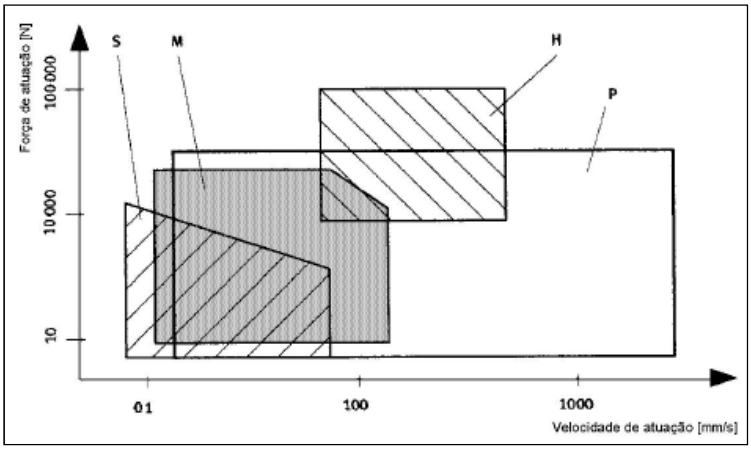


Figura 16 – Tipos de acionamentos para mecanismos de controle e automação

Fonte – [35]

O que se necessita é de um motor que receba como sinal de entrada um ângulo de referência e que o eixo do motor se mova para a referência recebida, ou seja, um servo-motor cumpre com o requisito. Tal dispositivo nada mais é que um motor de corrente contínua com um potenciômetro e um sistema de controle. No mercado existem modelos extremamente simples e baratos, como os chamados *micro servos* (Figura 17).

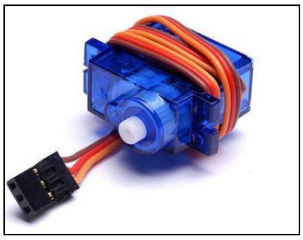


Figura 17 – Micro servo

Fonte – [36]

A desvantagem desse acionamento é o torque máximo, que de acordo com a especificação de [36], é de 1.3 kg.cm, sendo necessário o uso de um contrapeso em alguns elos do braço robótico para garantir a movimentação do mesmo ou motores mais potentes do mesmo tipo usado.

4.8.3: Interface de programação

O intermédio entre o mecanismo robótico e o usuário é feito através da interface de programação, que é a área onde o usuário comandará os motores através de comandos e funções pré-definidos. Essa interface foi construída com a RAD IDE Delphi, a qual tem como linguagem de programação o *Object Pascal*. Um dos maiores motivos que levou à escolha desse ambiente integrado de desenvolvimento foi a experiência dos membros do laboratório com o uso desta ferramenta, que apresenta grandes vantagens também, tais como:

- Linguagem de alto nível orientada a objeto.
- Integração com a API do Windows, o que permite a criação de programas que exploram ao máximo os recursos do sistema operacional.
- Compilador que gera arquivos executáveis nativos, ou seja, em código de máquina, tornando-o extremamente rápido e com proteção do código fonte.
- A IDE Delphi pode ser ampliada e personalizada com a adição de componentes e ferramentas criadas utilizando-se a linguagem *Object Pascal*.

Como desvantagens, essa interface fica muito restrita a aplicações, ou aos chamados “componentes”, de terceiros, sendo muitas vezes necessária a aquisição de um *plug in* [7] pago. Além disso, a própria IDE não é gratuita.

4.8.4: Linguagem de programação

Para o acionamento dos motores, utilizam-se funções em linguagem C que são utilizadas na programação do microcontrolador *Arduino*. Porém, devido à complexidade desta linguagem, para leigos no assunto da programação, torna-se a menos indicada para fins didáticos, sendo interessante o uso de uma linguagem mais simples e condizente com a realidade dos manipuladores robóticos.

O modelo que serviu de inspiração para o desenvolvimento deste trabalho foi o Scorbot – ER VII, já citado no item 5.6. Esse robô utiliza a linguagem ACL (*Basic*) pela qual o usuário realiza as operações com o braço.

Assim, torna-se necessária a implementação de um interpretador ACL para linguagem C.

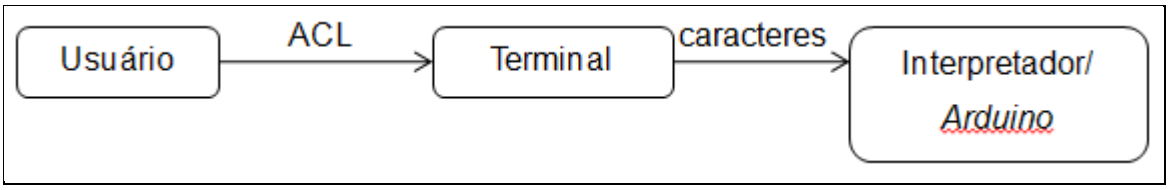


Figura 18 – Fluxo de informação usuário/arduino
Fonte – Elaborado pelo autor

Conforme a Figura 18, o usuário, através de um terminal, deve inserir os comandos em ACL. Em seguida a interface de programação (contém o terminal) envia os comandos digitados para o interpretador, presente no microcontrolador do *Arduino.* Os detalhes disso serão explicados no item seguinte.

[7] Plug in – permite a expansão de funções de um determinado sistema.

Capítulo 5: Integração dos recursos tecnológicos e materialização do braço

Neste capítulo são abordados os detalhes de implementação do projeto, com ênfase na criação da interface de programação, interpretador ACL para C e um detalhamento de como foi feita a geração de trajetória do órgão terminal.

Complementando o diagrama da figura 18, o Terminal é um campo da interface de programação que é usada para enviar comandos ao *Arduino* de forma imediata, podendo ou não movimentar o robô. O interpretador é construído na IDE do *Arduino* e nele gravado. A geração de trajetória é uma função criada nesse interpretador.

5.1: Interface de programação

Na IDE Delphi existem dois campos para a criação de programas, um é o chamado "*Form*", onde são inseridos os componentes visuais, o outro é um editor do código fonte na linguagem *Object Pascal.*

A interface é criada através de componentes pré-definidos do Delphi, sendo que cada um deles já possui campos para configuração de suas respectivas propriedades e funções. Por exemplo, um componente de botão (Figura 19) possui suas propriedades (Figura 20) e funções de evento. As propriedades podem ser: nome do objeto "botão", o nome que será visível no aplicativo, tamanho, visibilidade e etc.

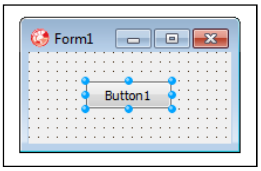


Figura 19 – Componente "Button"

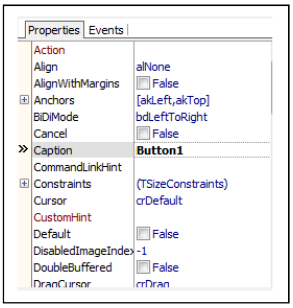


Figura 20 – Propriedades do componente “Button”

Já as funções de evento determinam quando uma determinada função (evento) deve ser executada. Por exemplo, ao escolher a opção “*OnClick*” e inserindo um nome para ela (Figura 21), automaticamente é criada no campo de edição do código-fonte uma função-padrão, a qual deve ser modificada de acordo com o que se deseja fazer quando o botão for clicado.

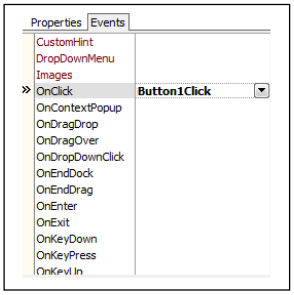


Figura 21 – Funções de eventos de “Button”

Com esse procedimento foi criada a interface de programação (Figura 22), a qual consiste de: botões para a configuração da porta serial – a qual é conectada à placa microcontroladora – e para a abertura da conexão; um terminal para a inserção de funções que vão para o microcontrolador; um editor de texto para a criação de um código que é gravado no cartão SD; botões para verificação da sintaxe e envio do código para o cartão e um campo para selecionar o programa do cartão e executá-lo.

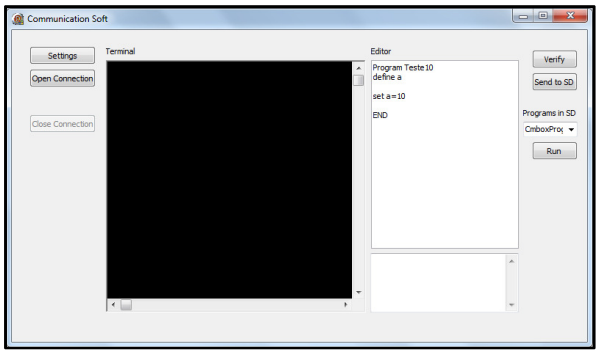


Figura 22 – Interface de Programação

Fonte – Elaborado pelo autor

5.1.1: Conexão

Primeiramente, para que a conexão seja feita o *Arduino* e a interface devem estar configurados na mesma porta serial para o envio e recebimento dos dados, e aquele deve estar conectado via USB com o computador da interface.

Feito isso, ao clicar em “open Connection” é chamado o método para “abrir” a porta serial do componente *TComPort* do Delphi. Assim, o aplicativo está apto para a recepção e envio de dados via terminal ou editor de código.

5.1.2: Terminal

O terminal é construído com o componente *TComTerminal*, semelhante ao *Hyper Terminal* do Windows. Estando a porta aberta, cada caracter inserido nele é imediatamente enviado pela porta serial ao microcontrolador, onde será feito o tratamento dos dados recebidos, o qual é detalhado no item 5.1.2. As mensagens que são recebidas do *Arduino* também são mostradas nesse terminal.

5.1.3: Editor de código

Os códigos criados neste campo devem iniciar com PROGRAM + [nome do programa] e terminar com END.

Para verificar a sintaxe, deve-se clicar no ícone “verificar”. Se não houver erros, pode-se enviar o programa para o cartão SD clicando no ícone “Enviar p SD”.

Havendo algum programa salvo no cartão, pode-se executá-lo clicando no ícone “Run” com a devida seleção do programa desejado no campo à esquerda desse ícone, chamado “*Programs in SD*”.

5.2: Interpretador ACL/C

A principal função do interpretador é invocar as funções da linguagem utilizada pelo microcontrolador através de funções em ACL, as quais são utilizadas pelo usuário e que contém outras funções não nativas, sendo um ACL adaptado.

Conforme se digita os comados no terminal da interface, os caracteres vão sendo enviados para o interpretador, o qual deve fazer um tratamento dos dados recebidos para poder identificar as funções e os seus respectivos argumentos quando for o caso, gerando, também, tratamento de exceção quando algo estiver incorreto.

Existem dois estados do interpretador:

1. Modo *echo* no qual os dados são transferidos a partir do terminal da interface de programação.

2. Modo *download*. Neste caso o usuário escreve um código no editor da interface e o envia para o *Arduino*. O código não é mostrado no terminal e é gravado em um cartão SD.

Na imagem seguinte se apresenta um fluxograma com o primeiro modo de operação (*echo*).

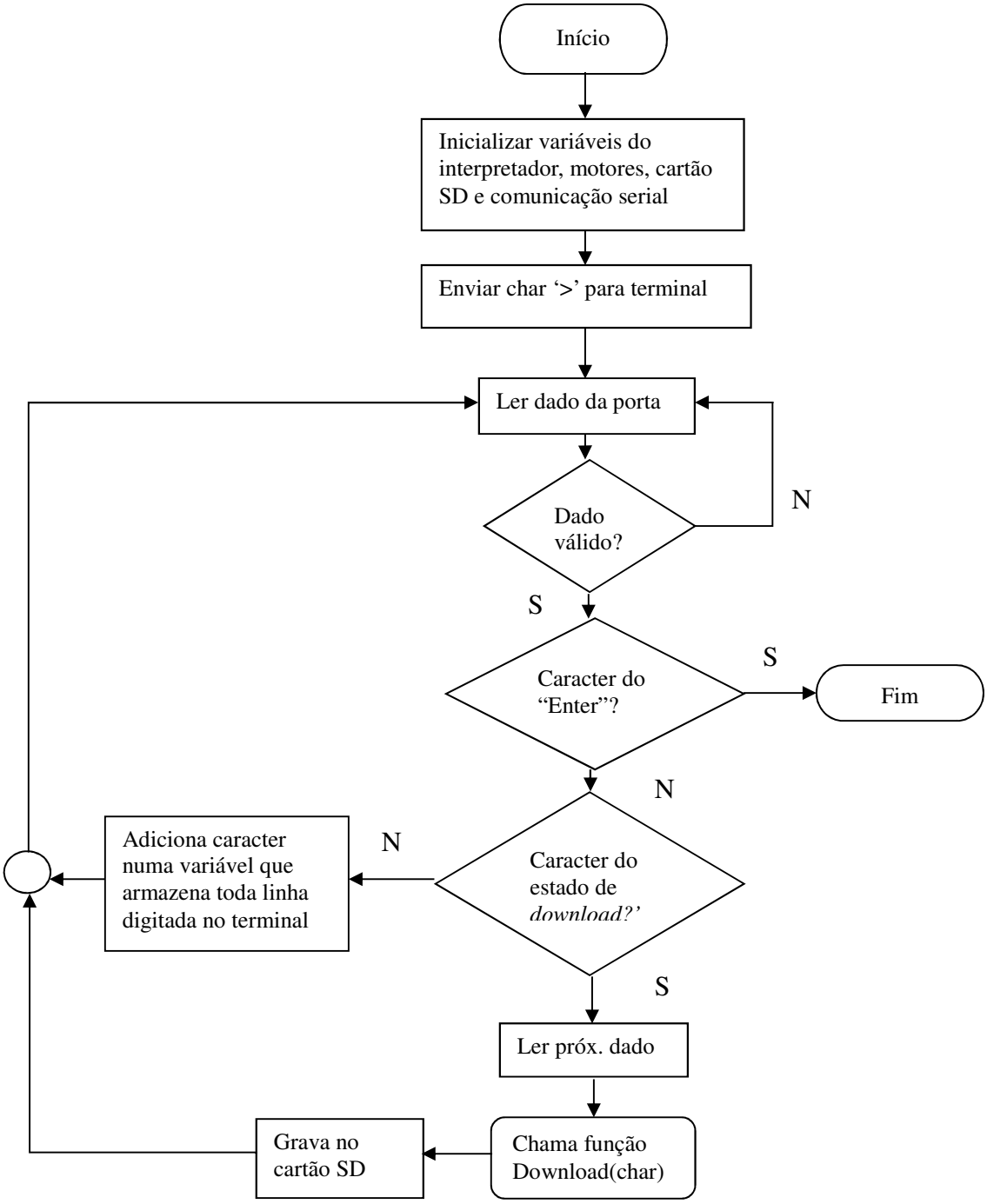


Figura 23 – Fluxograma do interpretador (modo *echo*)

Considerando o modo *echo*, após a identificação da linha digitada pelo usuário, necessita-se saber qual a função desejada com o seu respectivo parâmetro, quando houver. Como em ACL e em outras linguagens a primeira palavra geralmente é o nome de uma função. Este nome é verificado coletando-se os

caracteres até o primeiro espaço em branco. Feito isso, se não houver erros de sintaxe, coleta-se os dados dos argumentos, caso contrário, exibe-se uma mensagem de erro para o usuário. O mesmo tratamento é feito para a coleta dos argumentos.

A função *Download* do fluxograma anterior é detalhada em seguida.

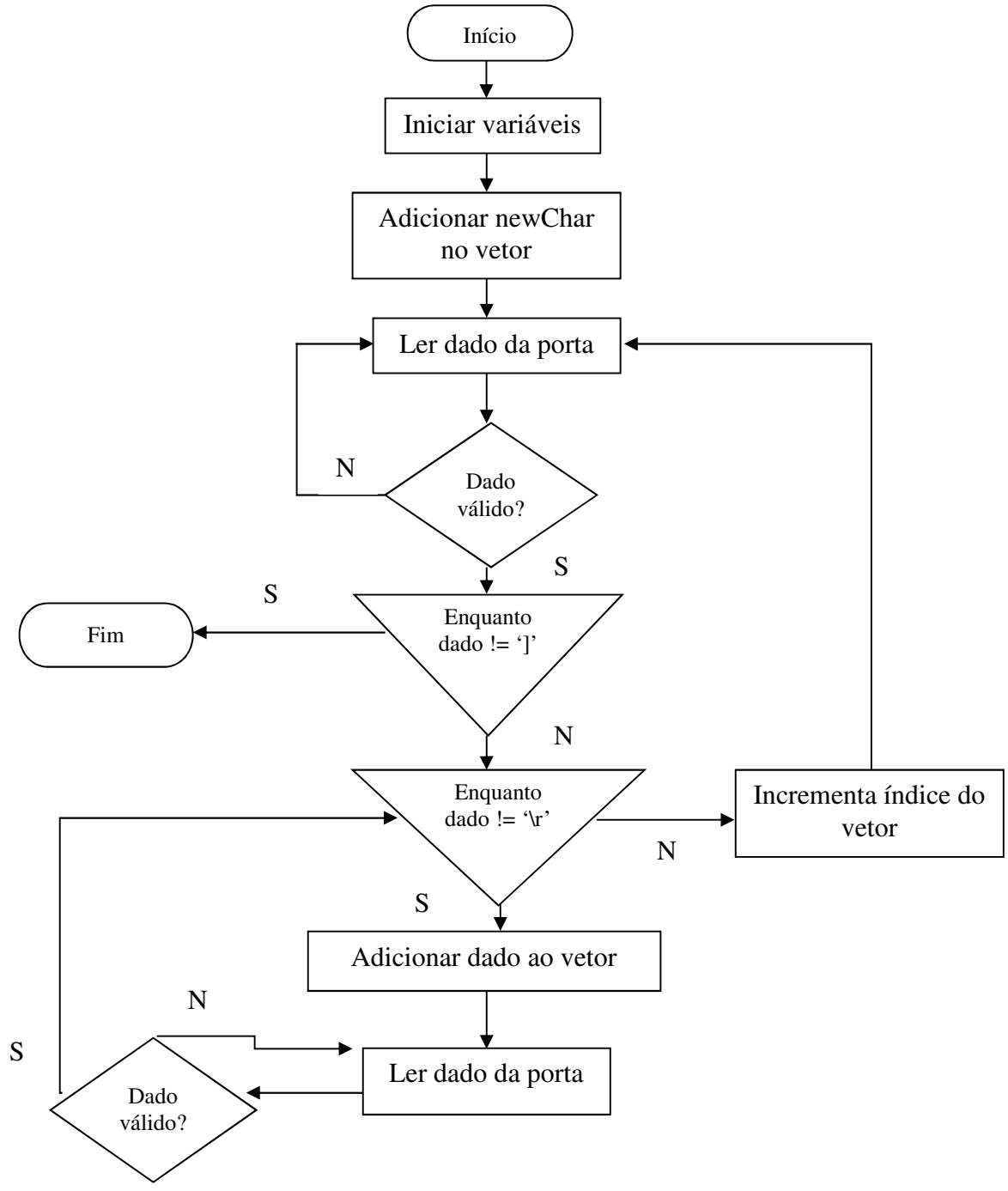


Figura 24 – Função Download

Após a conclusão da escrita do código no editor da interface e a execução da função para envio ao *Arduino*, o programa adiciona os caracteres especiais '[' e ']' no início e fim do código, respectivamente. Utiliza-se também um vetor do tipo *String* para armazenar o código digitado.

Existe uma função específica de uso interno do interpretador que invoca as funções de acordo com o estado (modo *echo* ou *download*).

As seguintes funções estão habilitadas para a utilização do usuário. Parte delas foram baseadas na apostila de ensino de robótica do curso de Engenharia Mecatrônica da PUCRS [14]:

- **Define**

  Esta função define apenas o nome de uma variável, alocando-a em um vetor de Structs, os quais têm dois campos (nome e valor). Os nomes devem iniciar com letras podendo ser seguidos de números. Caso se insira caracteres inválidos ou um nome já existente, uma mensagem de erro é apresentada ao usuário.

  Exemplo: DEFINE VAR

- **Set**

  Define um valor para a variável escolhida. Para atribuir o valor à *Struct* correta, faz-se uma busca pelo nome inserido e, então, atribui-se o valor ao campo "valor".

  Exemplo: SET VAR = 1

- **Print**

  Imprime no terminal o argumento da função, podendo ser de dois tipos: entre aspas e sem. O que estiver entre aspas é impresso no terminal, o que não estiver é considerado como o nome de uma variável e será impresso o valor da mesma.

Exemplo: PRINT “TESTE” → No terminal: TESTE.

PRINT VAR1 → No terminal: 5, se for este o valor alocado à VAR1.

- **If**

  Função condicional. Válida apenas para programas criados no editor da interface. O fluxograma desta função se encontra no Apêndice A.

  Exemplo:

```
IF VAR1 = VAR2
    Lista de comandos
ELSE
    Lista de comandos
ENDIF
```

- **For**

  Função de laço de repetição. Válida apenas para programas criados no editor da interface.

  Exemplo:

```
FOR I = 1 TO 10
    PRINT I
ENDFOR
```

- **Run**

  Esta função executa um código pré-programado e pode ser utilizada quando houver algum código no cartão SD e o mesmo estiver conectado ao *Arduino.* Ela pode ser executada tanto no terminal como no ícone “*Run*” da própria interface.

  Exemplo: RUN PROGRAMA1

- **Defp**

  Esta função define apenas o nome das variáveis de posição do robô. Essas variáveis são do tipo *Struct* as quais contêm campos de nome, eixo1, eixo2, eixo3, *roll* e *claw.* Os valores que vão nos campos, exceto no nome, são ângulos (em graus) que representam a posição do eixo de cada motor. Para a atribuição desses valores deve-se utilizar em conjunto as funções MANUALMODE e HERE que serão descritas adiante.

- **Listp**

  Imprime no terminal todas as variáveis de posição do robô, as quais contêm o nome, valores do eixos 1, 2, 3, do *roll* e *claw.*

- **Saveeprom**

  O processo para definir variáveis de posição e seus respectivos valores é um tanto desgastante quando se necessita de muitas variáveis desse tipo. Por isso, torna-se interessante gravá-las para que não se percam quando o microcontrolador for desenergizado. Desse modo, esta função salva esses dados na eeprom do *Arduino.*

- **Eraseeprom**

  “Apaga” os dados da eeprom. Na verdade os campos da memória são substituídos pelo caractere ‘0’, identificando que se pode utilizar a memória.

- **Manualmode**

  Esta função movimenta os motores de cada junta de modo não simultâneo. Primeiro seleciona-se o tipo de sistema de coordenada (no momento só existe o de juntas), podendo ser de “juntas” ou “xyz”. Depois se escolhe qual motor se deseja mover. Os seguintes caracteres realizam essa tarefa:

1 ou ‘q’ → movem o motor do eixo 1 (base).

2 ou ‘w’ → movem o eixo 2 (*shoulder*).

3 ou ‘e’ → movem o eixo 3 (elbow).

4 ou ‘r’ → movem o punho (*roll*).

5 ou ‘t’ → abre ou fecha a garra (*claw*).

Para sair deste modo, deve-se digitar o caracter ‘~’.

Cada clique move apenas um grau do eixo do motor.

- **Here**

  Uma das formas para a geração de trajetórias de robôs manipuladores é a utilização do modo *teach* (ensinar), que consiste em mover o braço através de um *teach pendant* até a posição desejada, gravando-a na memória do sistema. Neste projeto, ao invés de se utilizar um *teach pendant*, movimenta-se o braço com comandos via teclado com a função “Manualmode”. Para gravar os dados da posição deve-se usar a função “Here”, a qual tem como argumento o nome de uma variável que já deve existir.

  Exemplo:

  HERE POSICAO1

- **Open**

  Comando para abrir a garra ou órgão terminal.

- **Close**

  Comando para fechar a garra.

- **Move**

Para utilizar esta função necessita-se de ao menos uma variável de posição a qual entra como argumento para a mesma. O robô se move da posição atual até a desejada. Os motores se revezam para mover seus respectivos eixos de grau em grau e na ordem que inicia do acionamento da base até a junta *roll*, ou seja, o eixo 1 move um grau, o segundo, também, e assim sucessivamente até chegar na posição desejada em ângulo de cada eixo.

- **Movel**

  Esta função move o órgão terminal entre dois pontos através de uma reta. Apesar de ter praticamente a mesma funcionalidade da função anterior, o modo como é implementada difere bastante. Seus detalhes são apresentados no item seguinte (5.2.1).

- **Speed**

  Determina a velocidade de rotação do eixo do motor dado em porcentagem. A velocidade máxima sem carga é de aproximadamente 60°/0,12s ou 1°/ 2ms, de acordo com [36]. A velocidade é determinada com o tempo de espera na movimentação de 1° do eixo do motor. Em 100% incrementa-se 1 grau a cada 2 ms e em 1%, a cada 200 ms.

- **Motors2zero**

  Move os motores de modo que o braço fique na posição inicial padrão, também conhecida como “homes”.

As funções Define, Set, Print, If e For podem auxiliar no aprendizado básico sobre programação. Já as outras funções são exclusivas para a movimentação do robô. Os comandos Defp, Manualmode, Here, Move, Movel permitem que o usuário se familiarize com o procedimento de geração de trajetória chamado modo *Teach*, muito utilizado no meio industrial.

5.2.1: Geração de trajetória do órgão terminal

Este tópico refere-se à função *Movel* que realiza movimentos em linha reta do efetuador. Para realizar esta atividade é necessário aplicar conceitos de cinemática de robôs manipuladores, tais como Cinemática Direta e Inversa.

Um ponto importante a ser observado é que todos os cálculos devem ser feitos com base no sistema de coordenadas cartesianas. Como os campos da variável de posição que indicam os ângulos dos eixos de cada motor não estão neste sistema de coordenadas, os mesmos devem ser convertidos utilizando a Cinemática Direta com o método de Denavit-Hartenberg, o qual obtém as coordenadas cartesianas partindo de valores de juntas, que normalmente são do tipo angular. Este método pode fornecer parâmetros de posição e orientação de cada junta, mas o objetivo é obtê-los somente do efetuador os valores de posição.

Conforme já descrito na fundamentação teórica (Capítulo 3), tal método se baseia na estrutura física do braço, como é mostrado no apêndice D, no qual se cria um modelo simplificado do real, com suas dimensões e coordenadas cartesianas de cada junta.

Com esses dados pode-se montar a tabela de DH para a obtenção dos valores de posição do órgão terminal.

Tabela 1 – Parâmetros DH

Junta	θi	αi	ai	di
1	Θ1	90	5	35
2	Θ2	180	112	40
3	Θ3	-90	134	10

Fonte – Elaborado pelo autor

Os parâmetros da tabela estão definidos no capítulo 3 na parte da fundamentação teórica de Cinemática Direta. Eles representam valores angulares e de translação das coordenadas cartesianas de cada junta do braço robótico.

Com esses valores determinam-se as matrizes de transformação homogênea, as quais apresentam os parâmetros de posição e orientação das coordenadas cartesianas de duas juntas consecutivas:

$$A_{i-1}^{i} = R_{z,\theta_i} T_{z,d_i} T_{x,a_i} R_{x,\alpha_i}$$

ou

$$A_{i-1}^{i} = \begin{bmatrix} cos(\theta_i) & sen(\theta_i) & 0 & -a \\ -cos(\alpha_i)\, sen(\theta_i) & cos(\alpha_i)\, cos(\theta_i) & sen(\alpha_i) & -dsen(\alpha_i) \\ sen(\alpha_i) sen(\theta_i) & -sen(\alpha_i) cos(\theta_i) & cos(\alpha_i) & -dcos(\alpha_i) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A partir disso pode-se encontrar a matriz de transformação homogênea total indireta, que mostra a posição e orientação do órgão terminal em relação ao sistema de coordenadas na base.

$$H_0^n = A_0^1 A_1^2 ... A_{n-1}^n$$

Também é possível encontrar a mesma matriz através do modo direto, fazendo uma análise de projeção vetorial. Os elementos da matriz entre a 1ª e 3ª linha e coluna representam os cossenos diretores dos vetores do efetuador em relação ao sistema fixo inercial e na última coluna estão os valores de posição do efetuador.

$$H_0^n = \begin{bmatrix} \cos(x_n, x_0) & \cos(y_n, x_0) & \cos(z_n, x_0) & x_0 \\ \cos(x_n, y_0) & \cos(y_n, y_0) & \cos(z_n, y_0) & y_0 \\ \cos(x_n, z_0) & \cos(y_n, z_0) & \cos(z_n, z_0) & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Assim, iguala-se as duas equações para obter os componentes de posição da garra em relação à coordenada da base, os quais são x_0, y_0 e z_0.

Com isso, são feitos os seguintes passos:

a. Encontram-se as coordenadas cartesianas de duas variáveis de posição. Em cada variável estão os valores em graus de cada eixo de cada

junta do robô. Para isso é utilizado o método DH para fazer essa transformação.

Para encontrar essas coordenadas é necessário utilizar as equações com variáveis simbólicas fornecidas pelo método de DH que foram feitas a partir do Matlab aplicando a função *syms*.

b. Gera-se uma equação tridimensional para a reta que liga esses pontos.

Sendo $m = 1$ (m é um valor escalar que determina o módulo de um vetor v), P um vetor de coordenadas (x, y, z), $\vec{v}(a,b,c)$ um vetor paralelo a $\overrightarrow{P_1P_2}$ de mesmo módulo e $d = \left|\overrightarrow{P_1P_2}\right|$:

$$P_2 - P_1 = m\vec{v} \quad (1)$$

Considerando-se que já se tem as variáveis $P_1(x_1, y_1, z_1)$ (ponto inicial) e $P_2(x_2, y_2, z_2)$ (ponto final):

$$(x_2, y_2, z_2) - (x_1, y_1, z_1) = m(a,b,c)$$

$$(x_2, y_2, z_2) = (x_1 + ma, y_1 + mb, z_1 + mc)$$

Com isso obtém-se a seguinte equação paramétrica da reta:

$$\begin{cases} x_2 = x_1 + ma \\ y_2 = y_1 + mb \\ z_2 = z_1 + mc \end{cases}$$

Dividindo essa reta em 10 pontos (escolhido arbitrariamente), é possível obter as coordenadas do novo vetor $\vec{v}$ com módulo dividido pelo número de pontos desejado (neste caso 10).

$$a_{step} = \frac{a}{10}; b_{step} = \frac{b}{10}; c_{step} = \frac{c}{10}$$

$$\begin{cases} x_{n+1} = x_n + a_{step} \\ y_{n+1} = y_n + b_{step} \ \ com\ n = [1,9] \\ z_{n+1} = z_n + c_{step} \end{cases}$$

No algoritmo, as variáveis (x_n, y_n, z_n) iniciam com os valores de P_1. Ao final encontra-se o valor de P_2 e atribui-se a (x_n, y_n, z_n) os valores do último ponto encontrado e assim sucessivamente.

c. Para cada ponto, exceto os dos extremos da linha, realiza-se o procedimento inverso para encontrar os valores em ângulos necessários para cada junta. Isso é feito com a Cinemática Inversa pelo método geométrico.

O principal problema encontrado nesse ponto é a redundância, já que para cada coordenada pode existir pelo menos duas configurações possíveis (*elbow down e up*) para o manipulador, com exceção nas singularidades, em que o braço fica completamente “esticado”.

Outro fator limitante que deve ser considerado são as restrições do volume de trabalho do braço, ou seja, é possível ter duas variáveis de posição, mas devido à limitação desse volume, não é possível traçar uma linha reta com o órgão terminal.

Uma forma simples de contornar esse problema é fazer com que o robô atue apenas na posição *elbow down* ou cotovelo para baixo, dessa forma é excluída a redundância.

A determinação dos ângulos das juntas inicia com o do eixo *elbow* ou junta 3 (J3) denominado θ_3 como pode ser visto na figura 25, que representa a vista geométrica lateral do braço.

Já em posse dos valores das coordenadas P_x, P_y e P_z, calculados no passo anterior, os seguintes cálculos são feitos para encontrar o parâmetro θ_3:

$$L = \sqrt{(P_x - 5)^2 + (P_z - 35)^2}$$

Os valores 5 e 35 são as dimensões presentes na estrutura do eixo x e z, respectivamente, porém são excluídas para que a origem da coordenada fique na junta 2 ou *shoulder.*

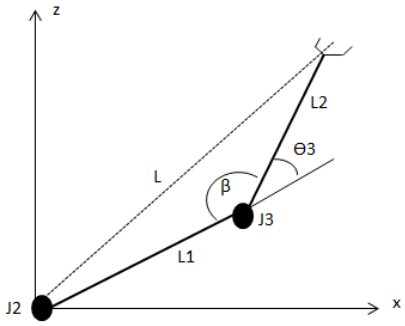


Figura 25 – Vista lateral do braço sem a base (θ_3)

Fonte – Elaborado pelo autor

A variável β é calculada com a lei dos cossenos:

$$\beta = cos^{-1}(\frac{-L^2+L_1^2+L_2^2}{2L_1L_2})$$

Portanto:

$$\boldsymbol{\theta_3 = \pi - \beta}$$

Observação: o motor do eixo 3 atua de 0 a 180 graus, mas geometricamente, θ_3 pode variar de -90 a 90 graus, sendo necessário fazer as devidas conversões no programa. Exemplo: se θ_3 for -90, o sinal que deve ser enviado ao motor é de 0 graus.

O processo é semelhante para encontrar o ângulo referente à junta 2 (θ_2). Com base na imagem seguinte, obtém-se:

$$\alpha = tan^{-1}(\frac{P_z - 35}{P_x - 5})$$

$\alpha = \theta_3 + \theta_2$, logo:

$$\boldsymbol{\theta_2 = \alpha - \theta_3}$$

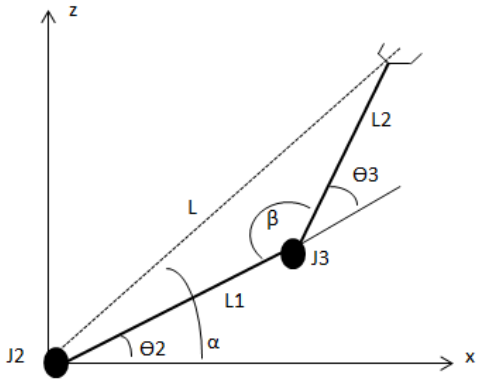


Figura 26 – Vista lateral do braço sem a base (θ_2)

Fonte – Elaborado pelo autor

Para o cálculo de θ_1, referente à base, utiliza-se a vista superior (Figura 23).

$$R = \sqrt{{P_x}^2 + {P_y}^2}$$

O ângulo α_2 é encontrado com a função “arco tangente 2” que retorna um valor entre $-\pi$ e π.

$$\alpha_2 = tan2^{-1}(\frac{P_y}{P_x})$$

$$\beta_2 = cos^{-1}(\frac{251}{R})$$

$$\boldsymbol{\theta_1 = \alpha_2 + \beta_2}$$

Cada ponto encontrado na reta possui seu correspondente em ângulos $(\theta_1, \theta_2, \theta_3)$ radianos, que devem ser convertidos em graus para serem usados como referência para os motores.

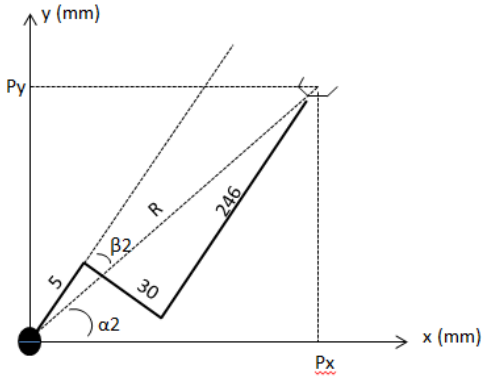


Figura 27 – Vista superior do braço (θ_1)

Fonte – Elaborado pelo autor

No *firmware* esses pontos são adicionados em um vetor de *Structs*, os quais contêm parâmetros de ângulos. O movimento de ponto a ponto é feito com a função *Move,* movendo os motores conforme já explicado, ou seja, quanto mais pontos, mais exato será o movimento.

5.3: Materialização do braço robótico

Um dos objetivos específicos do projeto tratava-se em criar três modelos de braço, sendo o primeiro deles feito em um material leve chamado cartão duplex, projetado através do recurso de modelamento de chapas (Figura 28). Este modelo, incluindo sua base, não era rígido o suficiente para manter o braço estável.

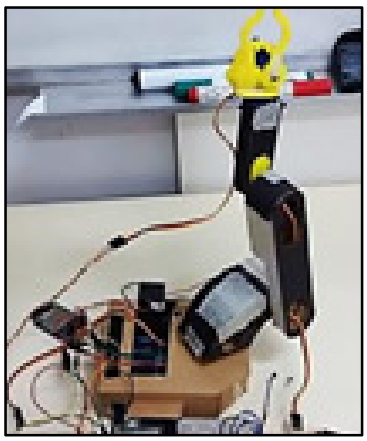


Figura 28 – Modelo 1

Fonte – Elaborado pelo autor

O segundo modelo foi criado em impressora 3D com material em ABS.

A primeira versão deste modelo ficou com uma estrutura um pouco mais rígida que a anterior (Figura 29), porém insuficiente para garantir que a estrutura ficasse estável durante a movimentação, sendo necessário criar uma nova versão materializada com impressão 3D, com o mesmo material, mas com ressaltos nas bordas dos elos para tornar a estrutura mais rígida e estável (Figura 30).

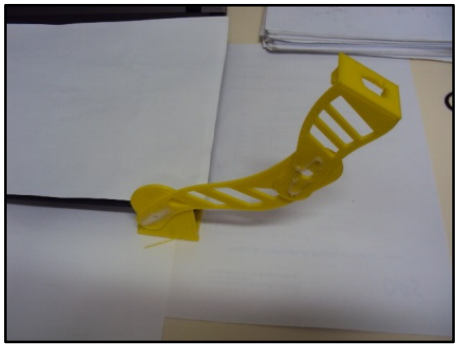


Figura 29 – Modelo 2 versão 1
Fonte – Elaborado pelo autor

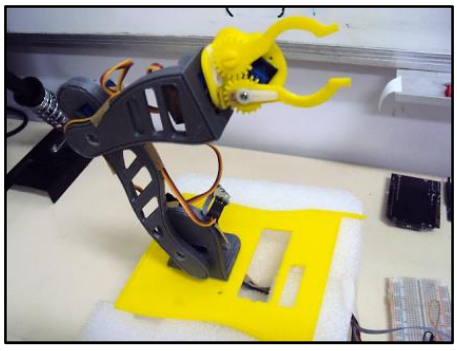


Figura 30 – Modelo 2 versão 2
Fonte – Elaborado pelo autor

No projeto estrutural o autor teve participação indireta na materialização. Sua contribuição nesse quesito foi fornecer especificações dimensionais para o encaixe dos motores e da placa de controle, montagem, testes do protótipo e geração de novas alternativas. O trabalho em equipe fez com que o processo de desenvolvimento fosse iterativo, com testes de prototipação e ajustes.

O custo das impressões 3D, incluindo o material utilizado e o custo de máquina/hora, foi de aproximadamente R$ 80,17, de acordo com o software de simulação da própria impressora.

O sistema de hardware (Figura 31) possui a placa *Arduino*, leitor de cartão SD e conexões com os servomotores. Para reduzir o número de conexões, tornando o sistema mais compacto, foi criada uma placa de circuito impresso a qual se conecta ao *Arduino* e aos demais componentes sem necessidade de *protoboard.*

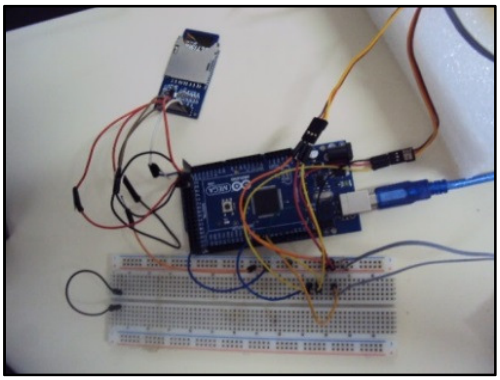


Figura 31 – Hardware
Fonte – Elaborado pelo autor

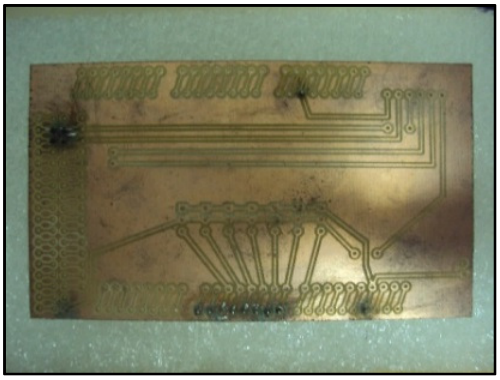


Figura 32 – Placa integradora (shield)
Fonte – Elaborado pelo autor

Com base em lojas especializadas em componentes eletrônicoso [37], a placa *Arduino Mega* de produção nacional custa em média R$ 89,00, o leitor de cartão SD, R$ 14,90 e os 5 motores utilizados, R$ 74,50. A placa de circuito impresso foi feita no Laboratório de Montagem Mecatrônica do Departamento de Automação e Sistemas da UFSC sem custo algum, totalizando R$ 178,40 somente para os componentes de hardware. Somando com a estrutura, o custo final fica em R$ 258,17.

Capítulo 6: Testes e avaliação

Neste capítulo serão apresentados os testes de software que incluem funções relacionadas direta e indiretamente ao robô. Será apresentado o objetivo de cada teste e seus resultados com imagens da interface de programação e do braço.

Para a avaliação do produto final serão utilizados os mesmos critérios da avaliação dos produtos na análise sincrônica (Capítulo 4): fabricação, material, preço, arquitetura, linguagem de programação, Microcontrolador, número de portas de sinal de saída e entrada, e estrutura. O capítulo se encerra com uma comparação entre os produtos melhores classificado naquela análise.

6.1: Testes de software

Os testes seguintes avaliam algumas funções implementadas no *firmware*. O objetivo principal é verificar o funcionamento das funções de acordo com as propostas apresentadas no capítulo 5. A figura 33 apresenta o teste com as funções que se relacionam de forma indireta com o robô, que são: *define, set* e *print.*

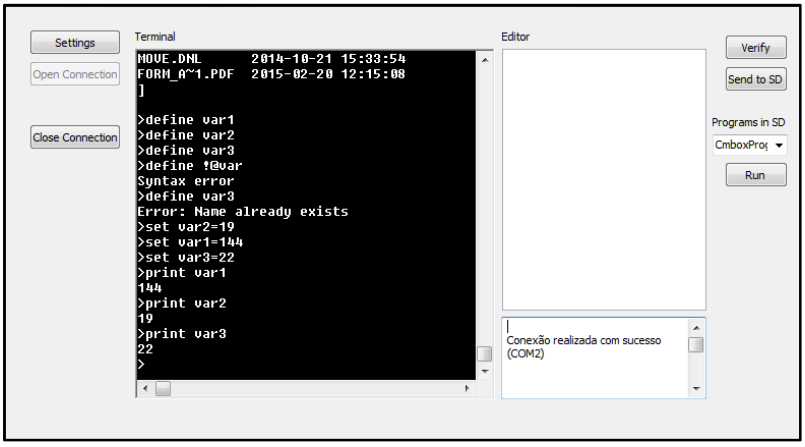


Figura 33 – Teste de criação e atribuição de variáveis

Fonte – Elaborado pelo autor

No programa são criadas três variáveis com a função *define*, seguidas de exemplos de tentativa de criação de nomes com erro de sintaxe e já existentes na

memória do programa. Após isso, utiliza-se a função *set* para atribuir os valores às variáveis, apresentando-os no terminal com a função *print.*

Na execução dessas funções não foram identificadas irregularidades, criando-se e atribuindo-se os valores às variáveis na memória local e gerando-se mensagens de erro coerentes com o que foi digitado.

Na figura 34, o resultado do teste com a função *run e if* são mostrados no terminal junto com sua programação, que deve, exclusivamente, ser feita no editor. Ao iniciar o programa da interface são apresentados ao usuário os arquivos presentes no cartão SD. Existem duas formas de se executar o programa do editor, uma é digitando a função *run* no terminal, ou então clicando-se no ícone com o nome desta função. Tanto um como outro invocam a mesma função no *firmware.*

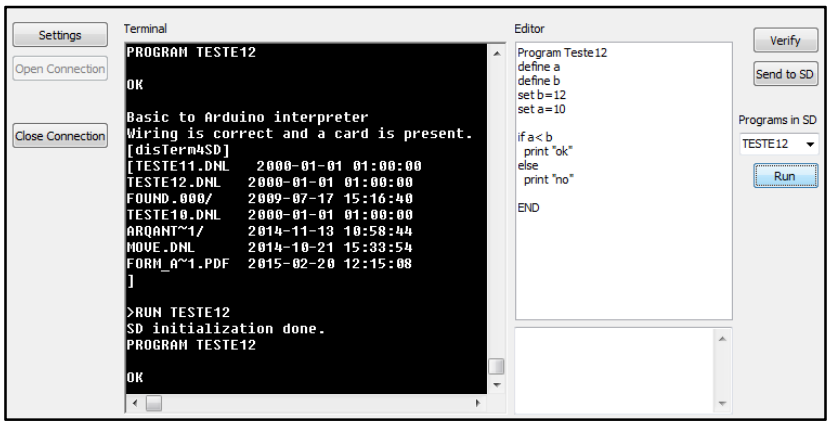


Figura 34 – Teste no editor de texto com função if

Fonte – Elaborado pelo autor

O próximo teste (Figura 35) é realizado com o mesmo procedimento do anterior, mas usando a função *for*. No momento ela pode ser utilizada apenas com as variáveis criadas com a função *define*, a qual é invocada internamente no programa deste exemplo, não sendo possível utilizá-la com as variáveis de posição do robô.

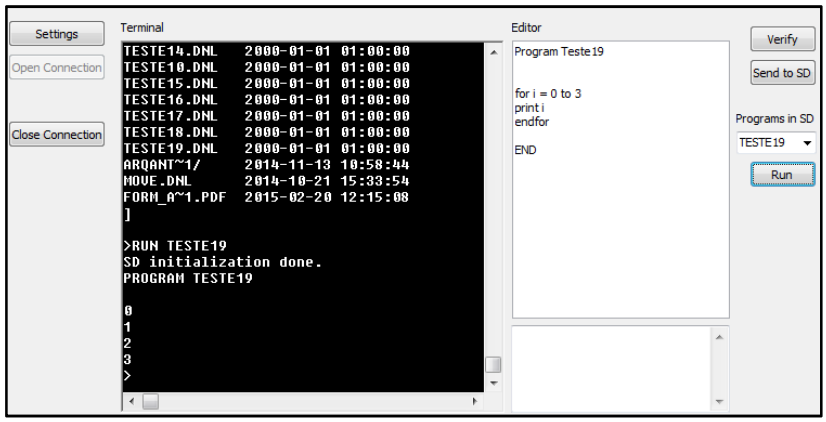


Figura 35 – Teste com função FOR

Fonte – Elaborado pelo autor

No teste seguinte (Figura 36) é feita a movimentação do braço através de comandos via terminal. Iniciando com *Manualmode, pressionado Ctrl+J* quando for solicitado para a escolha do sistema de coordenas mais o caractere relativo ao sentido da junta escolhida.

Devido a aspectos estruturais, verificou-se que o braço tem dificuldades de se movimentar a partir de determinadas posições. Porém, com influência externa, o mesmo conclui os movimentos de acordo com as instruções digitadas. Algumas soluções podem ser implantadas como já mencionado anteriormente, usando acionamentos de mesmo tipo só que mais potentes como o modelo DS821 com 5,2 kg.cm de torque.

O teste para a movimentação via código no cartão SD (Figura 37) também deve iniciar com o procedimento anterior, movendo-o via terminal. Na posição desejada utiliza-se o comando *here* para criar e gravar as posições de cada eixo em uma variável de posição e se repete esses passos até se atingir o número de pontos desejados. Com isso, cria-se o programa no editor. Neste caso, o teste foi feito com a função. A figura mostra o movimento de cada quadro da esquerda para a direita em cada linha, e de cima para baixo.

Figura 36 – Teste de movimento do robô via teclado e SD

Fonte – Elaborado pelo autor

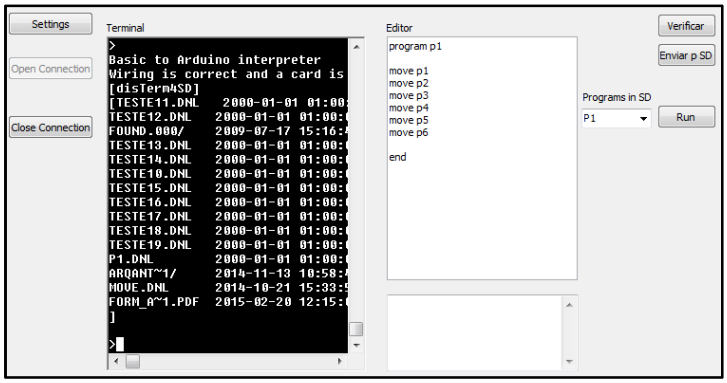


Figura 37 – Teste de movimentos via cartão SD

Fonte – Elaborado pelo autor

Os testes de movimentos retilíneos também foram realizados, porém, não cumprindo com o objetivo.

Primeiramente foi necessário escrever um programa no Matlab para gerar as equações simbólicas para obter as coordenadas cartesianas a partir dos ângulos com o método de Denavit-Hartenberg.

No teste foi criado o ponto P9 com os ângulos das juntas da base, *shoulder* e *elbow de* 172, 26 e 176 graus, respectivamente. E o ponto P10 com 172, 68 e 105 graus para as mesmas juntas. A partir do método DH geraram-se os dois pontos extremos da reta desejada. No quadro 17 são apresentadas as matrizes, com os pontos da reta considerando 10 pontos de interpolação gerados pelo Matlab e pelo *Firmware.* Já na figura 38, são mostrados as posições do braço referente a P9 e P10.

Quadro 16 – Resultados dos pontos Matlab e firmware em mm

MATLAB			***FIRMWARE***		
x	**y**	**z**	**x**	**y**	**z**
-166.80	53.73	-31.94	-98.49	141.44	206.70
-162.34	53.11	-4.16	-98.49	141.44	206.70
-157.88	52.48	23.61	-97.48	140.43	197.21
-153.42	51.85	51.39	-96.47	139.42	187.71
-148.95	51.22	79.17	-95.46	138.41	178.21
-144.49	50.60	106.95	-94.46	137.39	168.72
-140.03	49.97	134.73	-93.45	136.38	159.22
-135.57	49.34	162.51	-92.44	135.37	149.72
-131.10	48.72	190.29	-91.43	134.35	140.23
-126.64	48.09	218.08	-90.42	133.34	130.73

A simulação feita no Matlab foi coerente com o desejado, considerando-se a coordenada de referência adotada de acordo com DH, exceto para a coluna do eixo z.

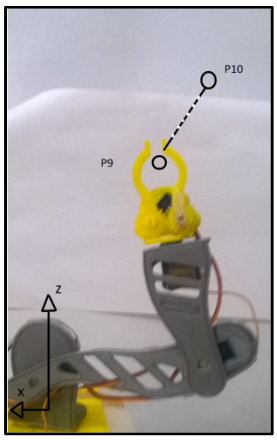


Figura 38 – Posição do braço nos pontos p9 e p10

Fonte – Elaborado pelo autor

Alguns fatores com grande possibilidade da causa do problema podem ser os seguintes:

- O erro da coordenada z vem diretamente do resultado da matriz de transformação homogênea do órgão terminal ao eixo da base, podendo ter sido causado por algum parâmetro que não foi bem escolhido no método.

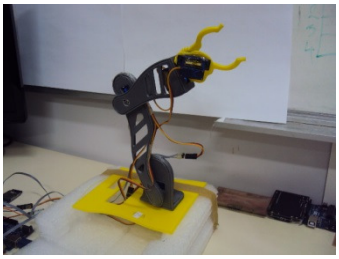
- Outra causa possível é a limitação do processador quanto ao número de bits. Eles determinam o quão exato é um número (número de casas decimais após a vírgula). Como esses valores são distintos tanto no *Arduíno* (8bits) quanto no Matlab (64 bits), os resultados finais nos equacionamentos também serão diferentes.
- No firmware foi utilizada exatamente a mesma equação simbólica do Matlab, porém os resultados foram completamente distintos, não se aproximando com o desejado. Sem os pontos corretos não é possível seguir para a Cinemática inversa.

Poder-se-ia realizar todos os cálculos necessários para encontrar essas equações no *firmware*, porém, o sistema terá resposta mais lenta proporcionalmente ao número de pontos da interpolação.

6.2: Avaliação do produto

Com base nos dados do produto final, gera-se um quadro de informações gerais e técnicas no mesmo padrão dos modelos da Análise Sincrônica (Quadro 15) que é comparado com os dois melhores modelos desta (RoboFácil e Hajime), quadro 3 e 5, respectivamente.

Quadro 17 – Informações do modelo do projeto



Informações gerais
Nome: ProntoArm
Fabricação: No Pronto3D/UFSC
Material: ABS
Preço: R$ 258,17 (aproximado)
Informações técnicas
Arquitetura: Aberta (Software)
Linguagem de programação: ACL modificado
Microcontrolador: **Nº de portas (sinal de saída):** Depende da placa utilizada.[8] **Nº de portas (sinal de entrada):** idem
Estrutura: Não Modular

Fonte – Elaborado pelo autor

[8] No *Arduino mega* existem 54 portas digitais e 16 analógicas, podendo configurá-las tanto como de entrada como de saída.

No quesito preço, apenas no projeto RoboFácil se apresenta um valor que é próximo ao do ProntoArm. Na arquitetura, tanto RoboFácil como Hajime disponibilizam a parte de hardware. Já o projeto deste trabalho, somente a arquitetura de software será disponibilizada, já que o circuito de montagem do sistema de hardware não é complexo. Foi necessário apenas verificar os documentos técnicos do leitor de cartão, da placa *arduino* e dos motores para se fazer a correta conexão dos fios. Na linguagem de programação, um dos projetos da análise tem linguagem do tipo icônica e o outro, visual. Neste, utiliza-se a convencional, no qual os comandos são textuais. No quesito de estrutura, tanto o ProntoArm como o Hajime não são modulares, ou seja, não é possível montar outras estruturas de braço robótico enquanto o RoboFácil não fornece estrutura pré-definida.

Seguindo o quadro 7, que contém os valores para cada atributo, gera-se um novo quadro comparando os modelos melhores classificados com o deste projeto.

Quadro 18 – Comparação dos modelos

			Braços robóticos		
	Peso	Nome	RoboFácil	Hajime	ProntoArm
Gerais	1	Fabricação	2	2	2
	1	Material	---	2	2
	3	Preço	3	---	3
Técnicos	3	Arquitetura	3	3	3
	3	Linguagem	2	2	1
	2	Estrutura	---	1	1
	-	Soma	26	21	27
Classificação			**2º**	**3º**	**1º**

Fonte – [21] e [22]

O resultado mostra que o ProntoArm possui a melhor classificação levando vantagem sobre os outros por possuir todas as informações do quadro. Porém no quesito da linguagem de programação teve a menor nota. Isto se dá porque embora a linguagem textual seja simples e intuitiva, dificilmente superará uma que também tem essas características e seja visual.

A lista de necessidades do item 4.4 contém quadros com informações do que poderia ser mantido, acrescentado e melhorado nos modelos avaliados. De um modo geral o projeto cumpriu os pontos mencionados, como:

- **Baixo custo**. O valo final estimado foi o menor dentre os modelos avaliados, podendo reduzi-lo ainda mais se forem comprados componentes em grande escala;

- **Estrutura física compacta e leve**. O material utilizado (PLA) para a construção dos elos do braço garantem essas características podendo ser modelados em formatos complexos. Entretanto o processo de sua produção é demasiado lento se for comparado com outros materiais, como o MDF e o alumínio.

- **Ampliação de funcionalidades do sistema**. O projeto possui funções interessantes para o ensino em robótica, com funções que auxiliam no aprendizado à programação, ao manuseio básico de manipuladores industriais e conceitos de cinemática de braços robóticos.

- **Disponibilidade de componentes para a montagem do braço**. Não há disponibilização, porém os componentes são descritos ao longo do trabalho, de modo que o leitor pode adquiri-los e realizar suas conexões de acordo com as especificações do fornecedor.

- **Manual de instrução**. Formalmente, não foi elaborado este documento, mas as informações sobre o funcionamento do produto são encontrados ao longo deste trabalho, mais especificamente no capítulo 5.

Capítulo 7: Conclusões e Perspectivas

O projeto desenvolvido buscou contribuir em um campo emergente da grande área da robótica. Trata-se trada da robótica educacional, fundamentada nas teorias de educação provenientes da psicologia, como o construtivismo. Na robótica, buscou-se atuar no campo dos robôs manipuladores, com base nos modelos industriais do tipo antropomórfico, resultando no desenvolvimento de um kit didático de um braço robótico de baixo custo. A meta era desenvolver um produto similar aos modelos industriais, porém simplificado, abaixo de R$ 500,00. Excepcionalmente, poderia-se considerar um valor maior, de aproximadamente R$ 1.000,00, para modelos que aproximem dos que são usados na indústrias, como o modelo da *Eshed Robotec*: ER-VII que custa em torno de R$ 10.000,00.

Para conhecer o que já está presente no mercado e alguns dos trabalhos desenvolvidos no meio acadêmico, foi realizada uma pesquisa bibliográfica para a coleta de informações pertinentes para o projeto. Desta forma pôde-se levantar os requisitos estruturais, funcionais e técnicos.

O conjunto desenvolvido pode ser dividido em: (1) *firmware,* (2) interface de programação, (3) estrutura física do braço e (4) componentes de hardware e acionamentos, com custo de R$ 258,17 somente em materiais, sem levar em conta o custo de desenvolvimento de software.

No aspecto educacional, com relação ao que foi proposto, os resultados obtidos foram satisfatórios, já que os testes para as funções do robô – descritos no capítulo anterior – mostraram resultados esperados. Uma exceção foi a função de geração de trajetória retilínea. Porém, o *firmware,* o qual estará disponível na internet, contém toda a estrutura conceitual para a execução desta função, permitindo o aprendizado básico sobre cinemática de robôs manipuladores. A solução encontrada para o problema da trajetória retilínea pode ser a seguinte: não utilizar as equações simbólicas obtidas no Matlab e fazer todos os cálculos da

cinemática direta no próprio *firmware*. No Matlab, os resultados foram coerentes com o que se esperava.

As outras funções auxiliam no aprendizado a programação além de permitir que o braço seja um instrumento secundário para atividades educacionais com outros focos além da robótica propriamente dita, como a elaboração de atividades que desenvolvam a criatividade e organização do usuário.

Com relação às atividades desenvolvidas, o projeto contemplou diversas áreas do Curso de Engenharia de Controle e Automação da UFSC, conforme o quadro seguinte:

Quadro 19 – Relação entre tópicos de projeto e disciplinas do curso

Tópico do projeto	**Disciplina**
Desenvolvimento de produto	DAS5411 – Introdução a Engenharia de Controle e Automação DAS5312 – Metodologia para Desenvolvimento de Sistemas EMC5301 – Introdução ao Projeto e Manufatura Assistidos por Computador
Desenvolvimento do *firmware e interface*	DAS5334 – Introdução a Informática para Automação DAS5305 – Informática Industrial I
Hardware	EEL7020 – Sistemas Digitais EEL5346 – Eletrônica Básica DAS5151 – Instrumentação em Controle
Aplicação de teoria de robótica	EGR5604 – Desenho Técnico I MTM5512 – Geometria Analítica MTM5245 – Álgebra Linear EMC5251 – Introdução à Robótica Industrial

Fonte – Elaborado pelo autor

- **Desenvolvimento de produto**

  Relacionado à aplicação de metodologias relativas à organização projetual, tais como pesquisa de mercado, análise de produtos similares ou sincrônica, levantamento de requisitos, geração de alternativas, implementação e testes. Tópicos estes presentes na disciplina DAS5411 e DAS5301. Em DAS5312 é abordado um tema relativo ao levantamento de requisitos de software de forma mais completa, o qual foi utilizado neste trabalho.

- **Desenvolvimento do *firmware* e interface de programação**.

  Foram necessários conhecimentos de programação em C e Delphi. Em DAS5334 e DAS5305 foram fornecidos conhecimentos necessários para a programação em C e iteração com o *firmware* e com a interface. A disciplina DAS5102 auxiliou no entendimento dos princípios gerais de concepção de um programa de computador, recursividade e passagem de parâmetros.

- **Aplicação de teoria de robótica**

  Neste quesito foram necessários conhecimentos básicos de desenho técnico, manipulação de matrizes, álgebra linear e cálculo presentes nas disciplinas EGR5604, MTM5161, MTM5512 e MTM5245. A teoria de cinemática inversa, direta e movimento de corpos rígidos foram apresentados na disciplina EMC5251.

- **Hardware**

  No projeto não houve desenvolvimento de hardware, além da placa de circuito impresso, porém, para a devida utilização dos componentes, foram necessários conhecimentos de sinais digitais e conceitos básicos de eletrônica fornecidos nas disciplinas EEL7020, EEL5346 e DAS5151.

Possibilidades no uso do kit no ensino:

- Utilização do braço como meio facilitador para atividades educativas, que desenvolvam aspectos de trabalho em equipe, raciocínio lógico, criatividade e etc.;
- Ensino de lógica de programação para nível técnico e superior;
- Ensino de conceitos de robótica industrial para nível técnico e superior, através da função de modo *teach* e conceitos de cinemática de manipuladores robóticos através da função “Movel” para movimentos retilíneos*;*

Com relação ao que pode ser realizado em trabalhos futuros, será divido em tópicos de estrutura física, *firmware* e interface:

Estrutura física:

- Gerar estrutura com menos atrito entre acoplamentos;
- Gerar um órgão terminal com variação de pressão sobre as partes que entram em contato com os objetos;
- Testar e avaliar o uso de servo motores com maior torque e outros tipos de acionamento com transdutores para realizar controle de posição mais sofisticado;
- Teste com modelos em maior escala em MDF e de estruturas modeladas através do processo de fabricação de modelamento de chapas já em desenvolvimento (Apêndice I);

***Firmware*:**

- Ampliar funções com base na linguagem ACL;
- Associa as funções de rotinas de repetição às variáveis de posição do braço robótico;
- Implementar outras funções de movimento para o braço robótico, como os circulares;

Interface de programação:

- Gerar um módulo de simulação de *Teach Pendant;*
- Permitir que na interface o usuário possa escolher os parâmetros do braço utilizado para os cálculos de cinemática direta e inversa;

- Criar na interface um campo de simulação visual de movimentos;

Bibliografia:

[1] Malone, T. W; Laubacher, R; Johns, T. The big idea: The age of hyperspecialization. Disponível em: <https://hbr.org/2011/07/the-big-idea-the-age-of-hyperspecialization/ar/1 >. Acesso em 25 de fevereiro de 2015.

[2] IFR. International Federation of Robotics. Disponível em: <ifr.org>. Acesso em 25 de Fevereiro de 2015.

[3] Valente, J.A. e Canhette C.C. LEGO-LOGO explorando o conceito de design. In J.A. Valente, (org.) Computadores e Conhecimento Repensando a Educação. Campinas, SP: NIED – UNICAMP, 1993. 580 p.

[4] Sacchelli, C. M. c; Garcia, T. R.; Delatorre, R. G.; Mikowski, A.; Bazzo, W. A.; Barros, A. A. C.; Chinelatto, A. L.; Chinelatto, A. S. A.; Peres, A.; Gobbi, A. M.; Furtado, C. M.; C.F. Junior, D.; Wiecheteck, G. K.; Siqueira, G. R.; Sandri, I. G.; Kruger, J. A.; Schwertl, S. L.; Villas-Boas, V. Potencial Social de Articulação Entre Ensino Médio e a Engenharia. In: Walter Antonio Bazzo; Adriana Maria Tonini; Valquíria Villas-Boas; Luiz Carlos de Campos; Liane Ludwing Loder. (Org.). Desafios da Educação em Engenharia: Vocação, Formação, Exercício Profissional, Experiências Metodológicas e Proposições. 1ed.Blumenau: EdFURB, 2012, v. , p. 13-36.

[5] Eleotério, J. R. Propriedades Físicas e Mecânicas de Painéis MDF de Diferentes Densidades e Teores de Resina. Dissertação de Mestrado. Piracicaba: Escola Superior de Agricultura, USP, 2000.

[6] Jelden, D.L. Operationalizing Learner-Crontrolled Education. International Conference on systems Research and Cybernetics, Baden – Baden, 1984.

[7] Coutinho, C. Percursos da Investigação em Tecnologia Educativa em Portugal: Uma abordagem temática e metodológica a publicações científicas (1985 - 2000). Dissertação de Doutoramento. Braga: Instituto de Educação e Psicologia da Universidade do Minho, 2005.

[8] Papert, S. Mindstorms: Children, Computers ans Powerful Ideas. New York: Basic Books, 1980.

[9] Logo Foundation. Disponível em: <http://el.media.mit.edu/logo-foundation/logo/index.html>. Acesso em: 20 de janeiro de 2015.

[10] Eshed Robotec. Scorbot-ER VII Users Manual. 2nd edition. Eshed Robotec, 1998.

[11] Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. Robotics: Modelling, Planning and Control. 1st edition. Springer Science & Business Media, 2009. 632 p.

[12] Eurobots. Fonte: <eurobots.net>. Acesso em 22 de janeiro de 2015.

[13] Fu, K.S.; Gonzalez, R.C.; Lee, C.S.G. Robotics: Control, Sensing, Vision and Intelligence. USA: McGraw-Hill, 1987.

[14] Braga, R. Programação de Robôs. Porto Alegre: DEM – PUCRS, 2005.

[15] Silva, R. M. Apostila: Introdução à Dinâmica e ao Controle de Manipuladores Robóticos. Porto Alegre: DEM – PUCRS.

[16] Lopes, A. M. Modelação Cinemática e Dinâmica de Manipuladores de Estrutura em Série. Dissertação de Mestrado. Porto: FEUP – DEMEGI, 2002.

[17] Bonsiepe, Guy. Metodologia Experimental: Desenho industrial. CNPq/Coordenação Editorial, Brasilia, 1984.

[18] Lego. Disponível em: <http://www.legobrasil.com.br/mindstorms>. Acesso em: 25 de agosto de 2014.

[19] Lynxmotion. Disponível em: < http://www.robotshop.com>. Acesso em 25 de agosto de 2014.

[20] Modelix. Disponível em: <http://www.wskits.com.br>. Acesso em 25 de agosto de 2014.

[21] Miranda, L. C. RoboFácil: Especificação e Implementação de Artefatos de Hardware e Software de Baixo Custo para um Kit de Robótica Educacional. Dissertação de Mestrado. Rio de Janeiro: IM – NCE – UFRJ, 2006.

[22] Sasahara, L. R.; Cruz, S. M. S. Hajime: Uma Nova Abordagem em Robótica Educacional. Artigo. Rio de Janeiro: Anais do XXVII Congresso da SBC, 2007.

[23] EasyArm. Disponível em: <http://www.labdegaragem.org/>. Acesso em 25 de agosto de 2014.

[24] Osier-Mixon, J. M. Hardware Aberto: Como e Quando Funciona. Disponível em: < http://www.ibm.com/developerworks/br/library/os-openhardware>. Acesso em 28 de janeiro de 2015.

[25] Leroy Merlin. Disponível em: < http://www.leroymerlin.com.br/?xdtoken=santa_catarina >. Acesso em janeiro de 2015.

[26] Shockmetais. Dados de chapas de alumínio. Disponível em: <www.shockmetais.com.br/produtos/aluminio/chapa >. Acesso em 28 de janeiro de 2015.

[27] Abal. Associação Brasileira do Alumínio. Disponível em: < http://www.abal.org.br/sustentabilidade/reciclagem/preco-da-sucata>. Acesso em 28 de janeiro de 2015.

[28] Oliveira, C. B. M. Estruturação, Identificação e Classificação de produtos em ambientes integrados de manufatura. Dissertação de Mestrado. São Carlos: EESC, USP, 1999.

[29] Waslavick, R. S. Análise e Projeto de Sistemas de Informação Orientados a Objetos. 2 ed. Rio de Janeiro: Elsevier, 2011.

[30] *Nutsvolts. Disponível em: <nutsvolts.texterity.com/nutsvolts/200705?pg=67&search_term=parallax#pg67>. Acesso em 9 de fevereiro de 2015.*

[31] Wiring. Disponível em: <wiring.org.co>. Acesso em 9 de fevereiro de 2015.

[32] Beaglebone. Disponível em: <http://beagleboard.org/bone>. Acesso em 9 de fevereiro de 2015.

[33] Raspberry Pi. Disponível em: <http://www.raspberrypi.org>. Acesso em 9 de fevereiro de 2015.

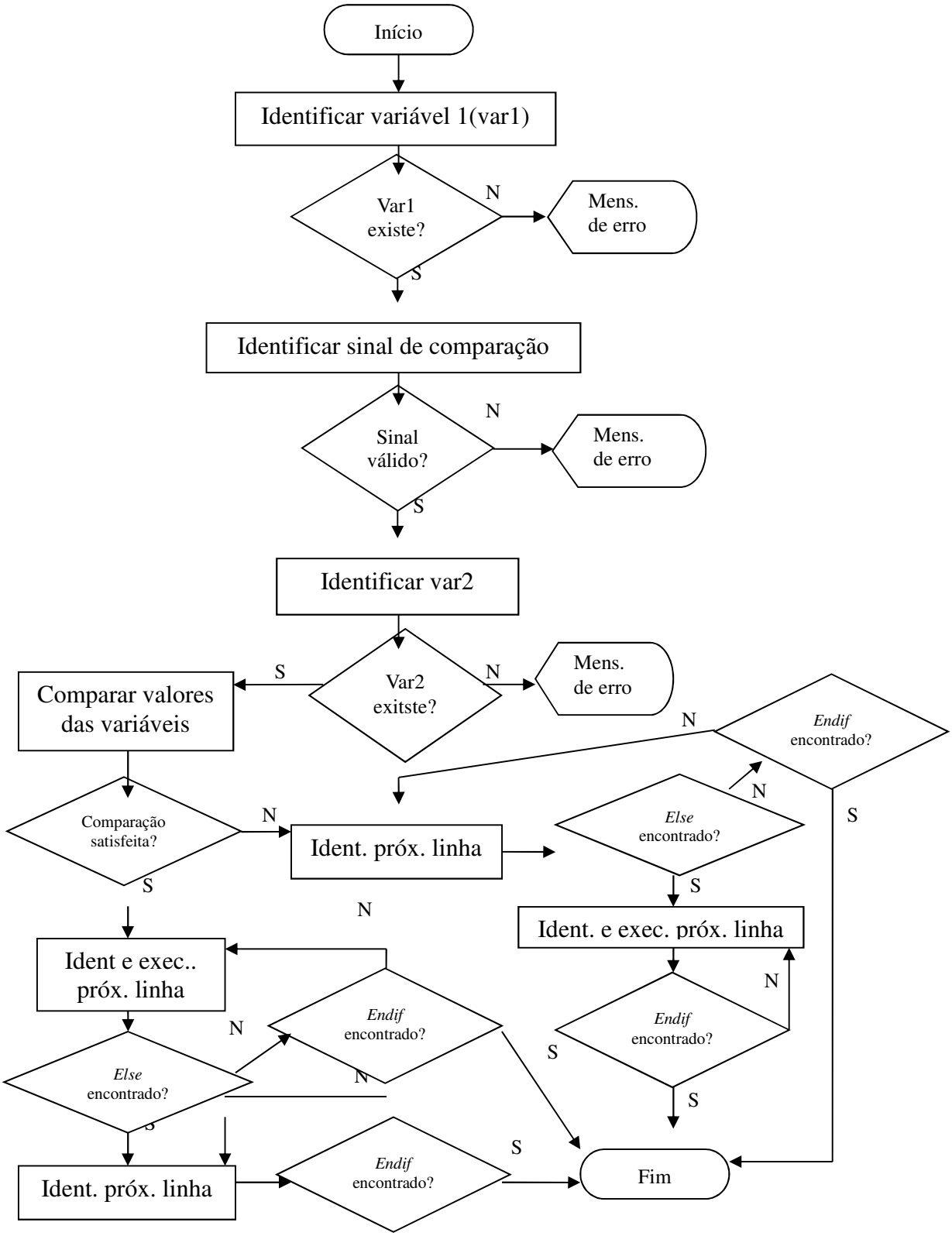
[34] Arduino. Disponível em: <arduino.cc>. Acesso em 9 de fevereiro de 2015.

[35] Negri, Victor J. Sistemas Hidráulicos e Pneumáticos para Automação e Controle: Parte I – Princípios Gerais da Hidráulica e Pneumática. Florianópolis: LASHIP/ EMC/ UFSC, 2001.

[36] Brasilrobotics. Disponível em: <brasilrobotics.blogspot.com.br>. Acesso em: 11 de fevereiro de 2015.

[37] FilipeFlop. Disponível em: <http://www.filipeflop.com/placa-ct-3d60d>. Acesso em 25 de fevereiro de 2015.

Apêndice A: Função *IF*



Apêndice B: Determinação dos parâmetros de Denavit-Hartenberg

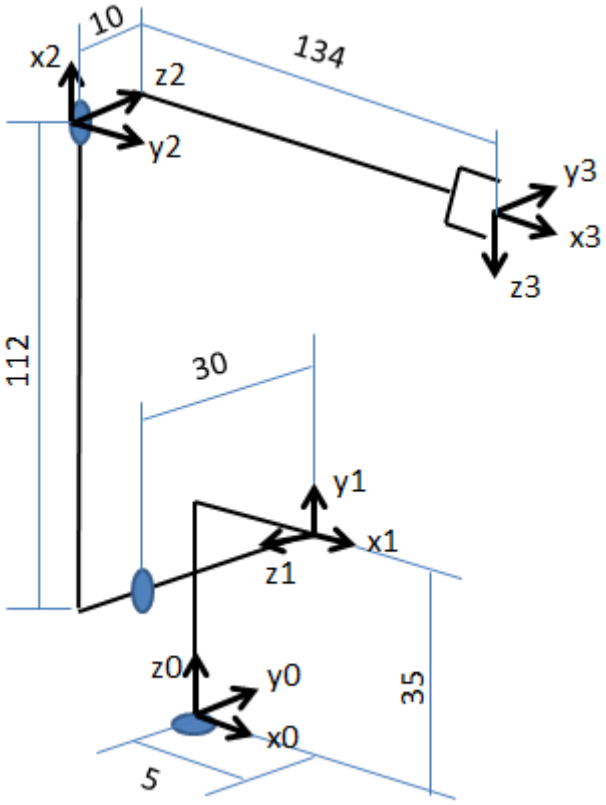


Figura 39 – Estrutura do braço segundo DH

Fonte – Elaborado pelo autor

Apêndice C: Alternativas de estrutura mecânica (Continua)

Nesta seção são apresentadas outras possibilidades estruturais para o manipulador robótico, como o desenvolvido através do método de fabricação de modelamento de chapas e em MDF.

Robô de Papel

Este modelo foi feito em modelamento de chapas, no qual o intuito é permitir que o usuário monte a estrutura do braço e a base. A vantagem é que a estrutura desmontada pode ocupar o mínimo de espaço em relação aos outros modelos (em PLA e MDF).

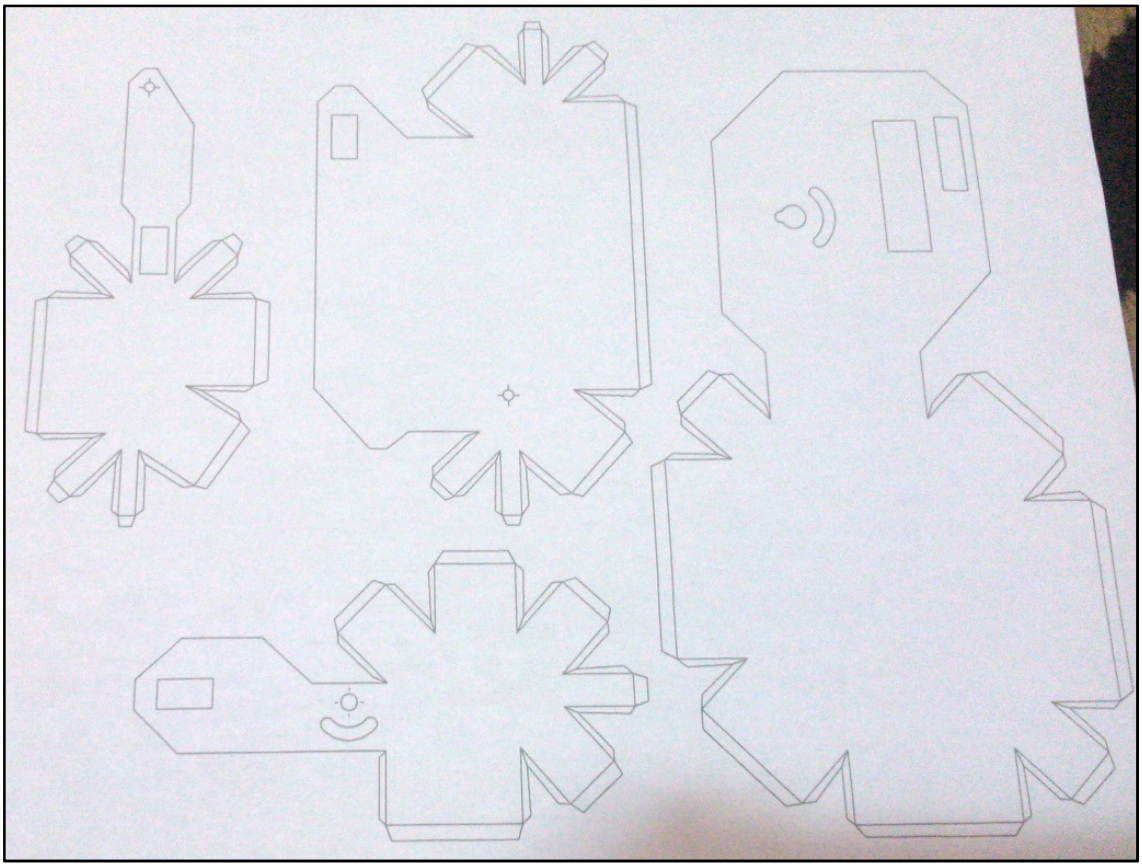


Figura 40 – Estrutura em modelamento de chapas

Apêndice C: Alternativas de estrutura mecânica (Continuação)

A montagem é mostrada nas imagens seguintes.

Figura 41 – Montagem do modelo em papel

Apêndice C: Alternativas de estrutura mecânica (Continuação)

Robô de maior porte em MDF

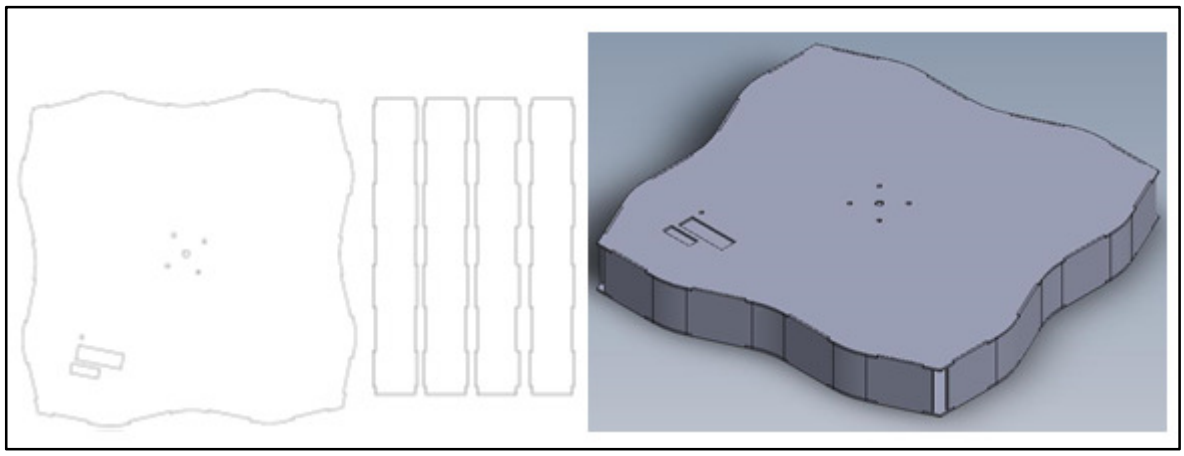


Figura 42 – Base

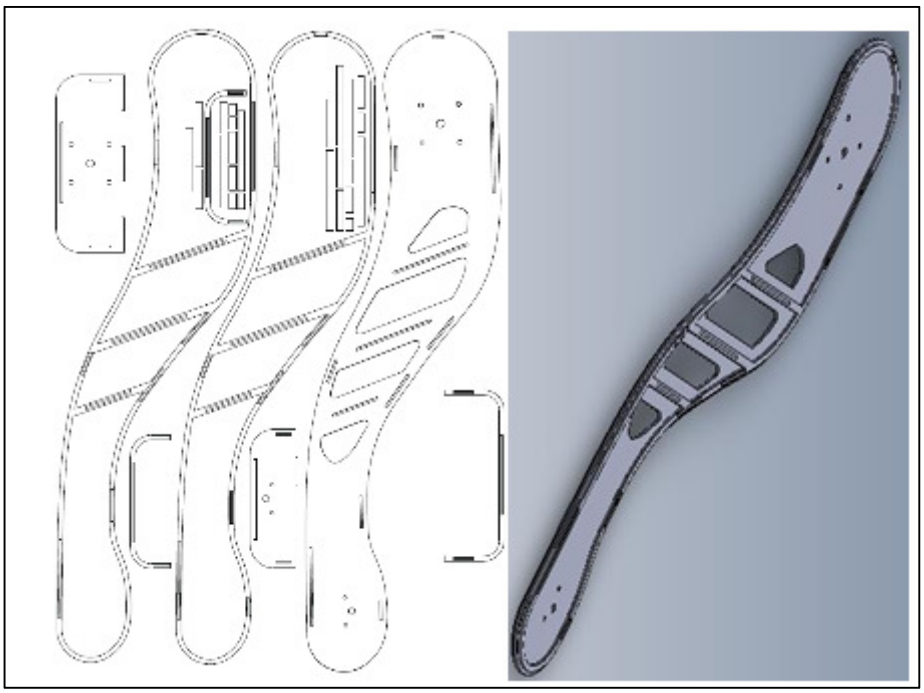


Figura 43 – *Shoulder*

Apêndice C: Alternativas de estrutura mecânica (Continuação)

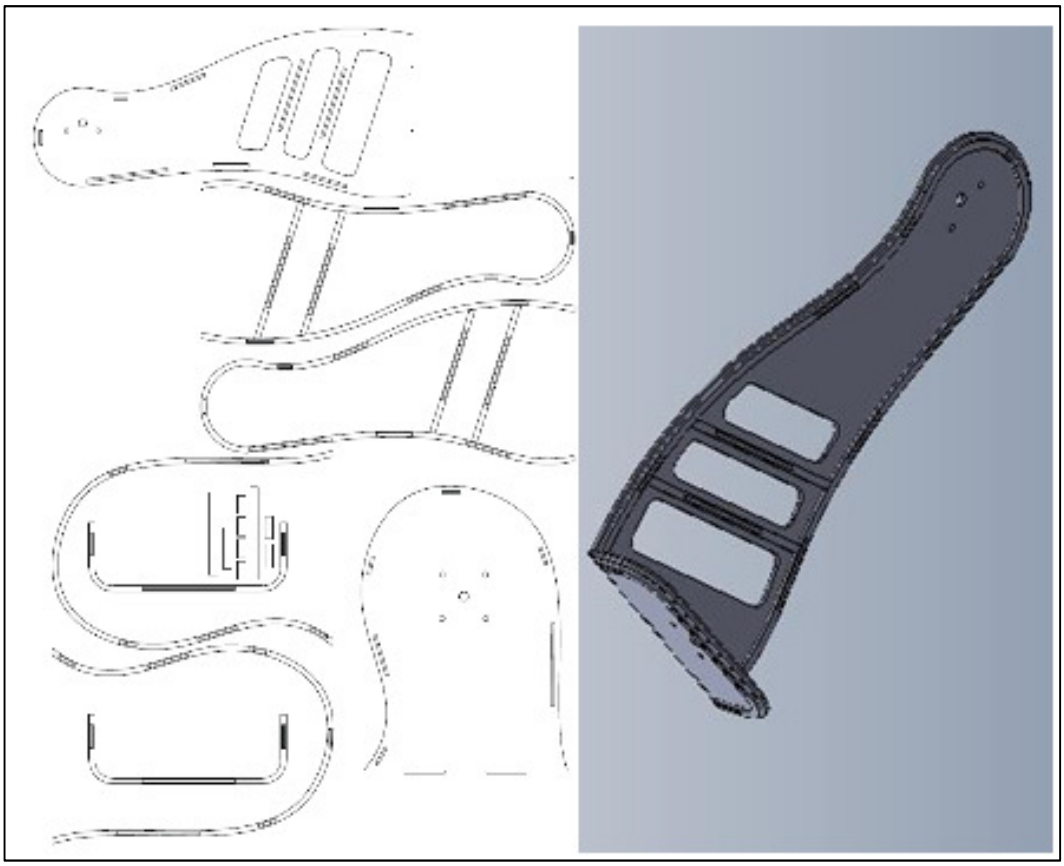


Figura 44 – Elbow

Nessa estrutura são usadas três chapas que se fixam através de encaixes, além de possuir filetes (ressaltos) para garantir maior rigidez do corpo.