

INSTITUTO FEDERAL DE GOIÁS
Engenharia de Controle e Automação

**CONSTRUÇÃO E CONTROLE DE UM MANIPULADOR
ROBÓTICO ACADÊMICO DE 5 GRAUS DE LIBERDADE**

Alexandre Alves Trindade
Denys Cezar Cabral

INSTITUTO FEDERAL DE GOIÁS
Engenharia de Controle e Automação

**CONSTRUÇÃO E CONTROLE DE UM MANIPULADOR
ROBÓTICO ACADÊMICO DE 5 GRAUS DE LIBERDADE**

Alexandre Alves Trindade
Denys Cezar Cabral

Trabalho de Conclusão de Curso (TCC) apresentada à Banca Examinadora como exigência parcial para a obtenção do título de Graduado em Engenharia de Controle e Automação pelo Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG), sob a orientação do Prof. Ildeu Lúcio Siqueira e coorientação do Prof. Énio Prates Vasconcelos Filho.

IFG
Goiânia - Goiás - Brasil
7 de dezembro de 2018

C1984s Trindade, A. A.; Cabral, D. C.

Construção e Controle de um Manipulador Robótico Acadêmico de 5 Graus de Liberdade/ Alexandre Alves Trindade

Denys Cezar Cabral. – Goiânia: Instituto Federal de Educação, Ciência e Tecnologia de Goiás, 2018.

96 f. : il.

Orientador: Prof. Dr. Ildeu Lúcio Siqueira.

Coorientador: Prof. Me. Énio Prates Vasconcelos Filho.

TCC (Trabalho de Conclusão de Curso) — Curso Superior em Engenharia de Controle e Automação, Departamento de Áreas acadêmicas IV, Instituto Federal de Educação, Ciência e Tecnologia de Goiás - IFG - Câmpus Goiânia.

Inclui apêndices.

1. Manipulador robótico — Comunicação — Cinemática Direta — Acionamento. I. Siqueira, I. L. (orientador). II. Vasconcelos Filho, É. P. (coorientador). III. Instituto Federal de Educação, Ciência e Tecnologia de Goiás. III. Construção e Controle de um Manipulador Robótico Acadêmico de 5 Graus de Liberdade.

CDD 004.6

Ficha catalográfica elaborada por Nome Bibliotecario CRB X/X.xxx

Biblioteca

Instituto Federal de Goiás, Goiânia - Goiás - Brasil

INSTITUTO FEDERAL DE GOIÁS
Engenharia de Controle e Automação

**CONSTRUÇÃO E CONTROLE DE UM MANIPULADOR ROBÓTICO
ACADÊMICO DE 5 GRAUS DE LIBERDADE**

Candidato: Alexandre Alves Trindade
Denys Cezar Cabral

Aprovado por:

Professor Ildeu Lúcio Siqueira , Dr.
Instituto Federal de Goiás
Orientador

Professor Carlos Roberto da Silveira Júnior, Dr.
Instituto Federal de Goiás
Examinador interno

Professor Cláudio Afonso Fleury, Dr.
Instituto Federal de Goiás
Examinador interno

Goiânia - Goiás - Brasil
7 de dezembro de 2018

*A todas as pessoas que apoiaram de alguma forma e tornaram
possível a realização deste trabalho.*

AGRADECIMENTOS

Ao nosso orientador professor Ildeu Lúcio Siqueira, pelo empenho e seriedade ao direcionar este trabalho da melhor forma possível .

Ao nosso coorientador professor Énio Prates Vasconcelos Filho, por todo apoio e disponibilidade para solucionar os desafios encontrados.

Ao engenheiro mecânico e técnico do laboratório de usinagem do IFG Paulo Vinícius da Silva Resende, pela ajuda na parte experimental do TCC.

Aos nossos familiares e amigos, pelo suporte e companheirismo.

RESUMO

Este trabalho apresenta a construção e o controle de manipulador robótico. É um projeto multidisciplinar, engloba conhecimentos das Engenharias de Controle e Automação, Mecânica e Elétrica. Este manipulador utiliza diversos *softwares* com o intuito de prever erros, simular movimentos, velocidade, acionamento e controle dos motores, e o quadro de comunicação desenvolvido. A metodologia consiste de: simulação, construção e testes de funcionamento. Os testes realizados são de posicionamento com *encoder*, acionamento e controle do sentido de giro do motor universal, responsável pela movimentação; comunicação serial entre supervisório e Proteus®. A técnica utilizada para realizar a comunicação resultou em economia de componentes, o motor universal CA utilizado proporcionou melhor custo-benefício e maior agilidade de movimentação. O algoritmo de cinemática do manipulador robótico acadêmico calcula posições com base em dados de entrada, que são pontos cartesianos, e através de recursos do simulador gráfico é estudado a trajetória de movimentação. O manipulador robótico construído realizou movimento gerado no ambiente de simulação, modifica a posição final do efetuador de acordo com os ângulos das juntas, dado que é enviado via supervisório. As incertezas de movimento decorrem da quantidade de peças e subconjuntos que foram acoplados, construção e montagem feita de forma manual, medidas auferidas com equipamentos analógicos que possuem determinado erro de medição, em contrapartida com métodos de fabricação que aplicam máquinas automáticas e instrumentos de medição com erro desprezível.

Palavras-chave: Manipulador robótico, comunicação, cinemática direta, acionamento de motores.

ABSTRACT

This work presents construction and control of robotic manipulator. It is a multidisciplinary project, encompasses knowledge of Control Engineering and Automation, Mechanical and Electrical. This manipulator uses several software in order to predict errors, simulate movements, speed, drive and control of the motors, and the protocol of communication adopted. The methodology simulation, construction and functional tests. These tests are of positioning with encoder; drive and control of the turning direction of the universal motor, responsible for the movement; serial communication between supervisory and Proteus®. The technique used to perform the communication has resulted in using less components, the universal AC motor used has provided better cost-benefit and greater agility of movement. The kinematics algorithm of the robotic manipulator calculates positions based on input data, which are Cartesian points, and using the resources of the graphic simulator, the movement trajectory is studied. The built robotic manipulator performed motion generated in the simulation environment, modifying the final position of the effector according to the angles of the joints, which is sent via supervisory. The uncertainties of movement arise from the number of parts and subassemblies that have been coupled, construction and assembly done manually, measures taken with analogical equipment that have a certain measurement error, counter to manufacturing methods that apply automatic machines and measuring instruments with despicable error.

Keywords: Robotic manipulator, communication, positioning, universal motor drive.

SUMÁRIO

	<u>Pág.</u>
LISTA DE FIGURAS	
LISTA DE TABELAS	
LISTA DE ABREVIATURAS E SIGLAS	
CAPÍTULO 1 - Introdução	1
1.1 Objetivo Geral	1
1.2 Objetivos Específicos	1
1.3 Estrutura do Trabalho	2
CAPÍTULO 2 - Fundamentação Teórica	3
2.1 Definição de Robô	3
2.2 Definição de Braço Robótico e Manipulador Robótico	4
2.3 Cinemática Direta	6
2.3.1 Posição e Orientação do Efetuador	7
2.3.2 Notação de Denavit-Hartenberg	11
2.3.3 Ângulos RPY	12
2.4 Acionamento e Controle do Motor Universal	14
2.4.1 Motor Universal	15
2.4.2 Ponte H	17
2.4.3 Controlador PID	19
2.5 Linguagem de Programação C#	21
CAPÍTULO 3 - Metodologia	23
3.1 Projeto Mecânico	23
3.2 Projeto Circuitos Eletroeletrônico	26
3.3 Desenvolvimento do Supervisório	30
3.3.1 Obtenção de Ângulos nas Articulações	33
3.4 Desenvolvimento de Sistema de Orientação e Posição do Efetuador	35
3.5 Construção do Manipulador Robótico	39
3.6 Testes de Funcionamento	41
CAPÍTULO 4 - Resultados e Discussões	46
4.1 Projeto Mecânico	46
4.2 Projeto do Circuito Eletrônico de Acionamento e Controle de Motores	52

4.3 Desenvolvimento do Supervisório	54
4.4 Cinemática do Manipulador Robótico Acadêmico	59
4.5 Construção do Manipulador Robótico	76
4.6 Testes de Funcionamento	78
4.6.1 Movimento do Manipulador Robótico Acadêmico	84
CAPÍTULO 5 - Conclusão	86
REFERÊNCIAS BIBLIOGRÁFICAS	88
APÊNDICE 1 - Placa Conversora RS232/TTL	91
APÊNDICE 2 - Placa de Controle com Microcontrolador	92
APÊNDICE 3 - Placa de Detecção de Passagem por Zero	93
APÊNDICE 4 - Placa de Controle de Potência com Triacs	94
APÊNDICE 5 - Circuito Eletrônico Controle dos Motores	96

LISTA DE FIGURAS

	<u>Pág.</u>
2.1 Esquema de manipulador robótico.	5
2.2 Esquema Estrutura de um Manipulador.	8
2.3 Sistema de coordenadas e parâmetros de ligamento.	8
2.4 Seis possíveis par inferior de juntas.	9
2.5 Sistemas de ligamentos são acoplados de forma que o sistema i é conectado rigidamente ao ligamento i	10
2.6 X - Y - Z ângulos fixos. Rotações na ordem $R_X (\gamma)$, $R_Y (\beta)$, $R_Z (\alpha)$	13
2.7 Ângulos Roll, Pitch, Yaw - Rotações Elementares.	14
2.8 Esquema motor CC excitação em série.	16
2.9 Conjulado <i>versus</i> velocidade de motor série universal	16
2.10 Circuito ponte H.	17
2.11 Corrente de estado constante para carga RL.	18
2.12 Controle pela tensão de terminal de armadura implementado com inversor de ponte H completa.	18
2.13 Diagrama de bloco de um sistema realimentado simples.	20
3.1 Exemplo de peças adequadas ao projeto.	24
3.2 Tela inicial do Solidwokrs®.	25
3.3 Principais etapas do desenvolvimento da estrutura mecânica.	25
3.4 Esquema do circuito eletrônico no Proteus® (versão de demonstração).	26
3.5 Circuito <i>snubber</i>	27
3.6 Circuito ponte H.	28
3.7 Circuito detector de passagem por zero.	29
3.8 Supervisório desenvolvido.	30
3.9 Vista geral do Visutal Studio C#.	31
3.10 Configuração da porta serial.	31
3.11 Validação da porta serial.	32
3.12 Quadro de envio de dados.	33
3.13 Relação matemática entre pontos e ângulos.	34
3.14 Disco <i>encoder</i>	34
3.15 Posição inicial do modelo.	36
3.16 Manipulador robótico acadêmico, vista:	37
3.17 Máquinas e equipamentos utilizados na construção do manipulador robótico acadêmico.	39
3.18 Fluxograma das etapas de testes de funcionamento.	41
3.19 Comunicação serial entre Proteus® (versão de demonstração) e o Supervisório.	42
3.20 Esquema inversão de rotação do giro do motor.	43

3.21	Esquema da comunicação feita para controle dos motores.	43
3.22	Vetor de comunicação (quadro).	44
3.23	Planta responsável pelo controle do ângulo de disparo do TRIAC.	45
4.1	Desenho do Projeto Mecânico, vista:	46
4.2	Primeira versão do projeto mecânico.	47
4.3	Segunda versão do projeto mecânico.	47
4.4	Submontagens e seus componentes:	48
4.5	Estudo dos ângulos dos eixos de movimentação.	49
4.6	Resultado apresentado no supervisório.	49
4.7	Formas de conversão de forças longitudinal para rotacional, vista:	50
4.8	Diâmetro máximo de movimento, vista superior.	51
4.9	Diâmetro máximo de movimento, vista lateral.	51
4.10	Esquema de controle do ângulo de disparo.	52
4.11	Painel elétrico de testes preliminares.	53
4.12	Supervisório desenvolvido.	54
4.13	Quadro de comunicação.	55
4.14	Painel elétrico.	55
4.15	Algoritmo de conversão de ângulos em pontos.	56
4.16	Visualização dos parâmetros de controle dos motores.	56
4.17	Modo de recepção dos dados e geração do gráfico.	57
4.18	PID sem torque.	58
4.19	PID com torque constante.	59
4.20	Ligamentos do manipulador robótico acadêmico (medidas em metro).	60
4.21	Posição inicial.	61
4.22	Posição após movimentação.	61
4.23	Fluxograma algoritmo de movimentação.	63
4.24	Resultado do teste de proximidade ao ponto x.	64
4.25	Giro de 180° na junta 1.	64
4.26	Resultados dos ajustes ao ponto objetivo.	65
4.27	Configuração da altura.	66
4.28	Fluxograma do código fonte.	67
4.29	Posição para x=-0,9.	70
4.30	Posição para x=1,29.	70
4.31	Posições 1-4, etapa da divisão de trajetória:	73
4.32	Ângulos da posição final.	75
4.33	Etapa de construção da garra.	76
4.34	Ligamento responsável pela sustentação do manipulador robótico acadêmico.	77
4.35	Manipulador robótico acadêmico montado.	78
4.36	Plataforma mecânica de testes preliminares de controle de potência, velocidade e inversão do sentido de giro do motor.	79
4.37	Plataforma mecânica, destinada a testes preliminares de posicionamento.	80

4.38	Disco <i>encoder</i> , e sensores ópticos.	81
4.39	Função linear de conversão de pontos para graus.	81
4.40	Interface de controle do manipulador.	82
4.41	Gráfico Velocidade x Tempo.	83
4.42	Testes preliminares com a garra.	84
4.43	Geração de trajetória, vista lateral.	85
1.1	Placa de conversão de sinais RS232/TTL.	91
1.2	Projeto eletroeletrônico de conversão de sinais RS232/TTL.	91
2.1	Placa de controle.	92
2.2	Projeto eletroeletrônico da placa de controle.	92
3.1	Detector de passagem por zero.	93
3.2	Projeto eletroeletrônico da placa detector de passagem por zero.	93
4.1	Placa de controle de potência e inversão de sentido de giro.	94
4.2	Projeto eletroeletrônico da placa de controle de potência e inversão de sentido de giro.	95
5.1	Projeto elétrico.	96

LISTA DE TABELAS

	<u>Pág.</u>
2.1 Matriz de transformada homogênea.	7
3.1 Legenda criação do objeto ligamento para o manipulador desenvolvido.	36
3.2 Parâmetros do PID.	45
4.1 Comandos para criar cada ligamento.	59
4.2 Comandos para criar e visualizar trajetória.	60
4.3 Matrizes transformadas homogêneas.	65
4.4 Resultado da transformação simultânea XY e XYZ.	66
4.5 Estudo do algoritmo aplicado ao ponto x.	68
4.6 Estudo do algoritmo aplicado aos pontos y e z.	69
4.7 Plano XY.	71
4.8 Plano XZ.	72
4.9 Plano YZ.	72
4.10 Sistema tridimensional XYZ.	73

LISTA DE ABREVIATURAS E SIGLAS

CA	- Corrente Alternada
CAD	- <i>Computer Aided Design</i> (Desenho Assistido por Computador)
CC	- Corrente Contínua
CLR	- <i>Common Language Runtime</i> (Tempo de Execução de Linguagem Comum)
CTS	- <i>Clear To Send</i> (Livre para Envio)
D-H	- Denavit-Hartenberg
DSR	- <i>Data Set Ready</i> (Conjunto de Dados Pronto)
DTR	- <i>Data Terminal Ready</i> (Terminal de Dados Pronto)
GC	- <i>Garbage Collector</i> (Coletor de Lixo)
GDL	- <i>Grau de Liberdade</i>
GND	- <i>Signal Ground</i> (Sinal Terra)
ISO	- <i>International Organization for Standardization</i> (Organização Internacional para Padronização)
IGBTs	- <i>Insulated Gate Bipolar Transistors</i> (Transistores Bipolares de Porta Isolada)
JIRA	- <i>Japanese Industrial Robot Association</i> (Associação de Robótica Industrial do Japão)
LED	- <i>Light Emitting Diode</i> (Diodo Emissor de Luz)
PIC	- <i>Programmable Interface Controller</i> (Interface de controle programável)
PID	- Proporcional-integral-derivativo
RIA	- <i>Robotics Industries Association</i> (Associação de Robótica Industrial)
RPY	- <i>Roll-Pitch-Yaw</i> (Rolagem-Arfagem-Guinada)
RS232	- <i>Recommended Standard 232</i> (Padrão Recomendado 232)
RTSS	- <i>Robotics Toolbox for Scilab/Scicos</i> (Ferramenta de Robótica para Scilab/Scicos)
RTS	- <i>Request To Send</i> (Requisição para Enviar)
RXD	- Recepção de dados
SISO	- <i>Single-Input Single-Output System</i> (Sistema Entrada-Simples-Saída-Simples)
TRIAC	- <i>Triode for Alternating Current</i> (Triodo para Corrente Alternada)
TTL	- <i>Transistor-Transistor Logic</i> (Lógica Transistor-Transistor)
TXD	- Transmissão de Dados

CAPÍTULO 1

Introdução

Este trabalho consiste em projetar, construir e controlar manipulador robótico com cinco graus de liberdade. A área de estudo e aplicação do presente TCC é robótica, especificamente robótica industrial. A construção do manipulador foi pautada pelo menor custo, utilizando materiais novos e usados encontrados na região metropolitana de Goiânia. É um trabalho multidisciplinar, e portanto, envolve conhecimentos de Engenharia de Controle e Automação, Mecânica e Elétrica.

A ideia nasceu da vontade dos integrantes deste trabalho em aplicar os conhecimentos adquiridos ao longo do curso no desenvolvimento de projeto do início ao fim. De forma semelhante aos Trabalhos de Conclusão de Curso de Engenharia de Controle e Automação do IFG Câmpus Goiânia, os quais citamos: "Desenvolvimento de um Sistema de Monitoramento de Movimento da Mão Humana e Reprodução do seu Movimento em Protótipo Robótico" e "Controle de uma Mão Robótica Articulada Controlada por um Microcontrolador". O desafio era construir um braço robótico, após alguns meses de imersão neste trabalho, foi notado o termo adequado pela bibliografia adotada de manipulador robótico acadêmico, protótipo de aplicação em indústria ou comércio no manuseio de objetos de pequeno e médio porte, para os propósitos definidos ao robô em construção.

A fase inicial consistiu em pesquisa de viabilidade econômica, previsão de tempo de conclusão do projeto, verificar ferramentas e máquinas disponíveis nos laboratórios do IFG - Câmpus Goiânia, e encontrar então professores com disponibilidade para agregar e orientar o desenvolvimento deste trabalho.

1.1 OBJETIVO GERAL

Construir e controlar manipulador robótico acadêmico com cinco graus de liberdade de baixo custo.

1.2 OBJETIVOS ESPECÍFICOS

- a) Projetar o manipulador robótico acadêmico de 5 (cinco) graus de liberdade em *software CAD*;
- b) Simular o circuito de acionamento e controle dos motores utilizados no robô desenvolvido;
- c) Desenvolver sistema de controle dos motores, testando diferenças entre simulação e prática;

- d) Realizar integração do projeto mecânico (robô construído) com o sistema computacional, para entrada de dados e leitura do *log* do sistema através de supervisório;
- e) Estudar e implementar recursos de manipulação e de cinemática direta para obter os parâmetros de posição e velocidade das juntas do protótipo construído.

1.3 ESTRUTURA DO TRABALHO

O presente Trabalho de Conclusão de Curso está dividido em cinco capítulos e mais cinco apêndices. O segundo capítulo, de fundamentação teórica, tem início com definições de robô, braço robótico e manipulador robótico. Apresenta notação adotada para posicionamento e movimentação de manipulador robótico, relações matemáticas utilizadas, finaliza no tópico da linguagem de programação C# utilizada para criar o supervisório.

Já o terceiro capítulo mostra a metodologia adotada neste trabalho. Como foi realizado o experimento proposto, especificações técnicas do *software* e do *hardware*.

O quarto capítulo traz apresentação e discussão dos resultados obtidos, com a realização da simulação de um robô com cinco graus de liberdade, desenvolvimento dos circuitos eletrônicos e criação do supervisório. O capítulo de conclusão, o quinto, de forma sucinta comenta os pontos principais da metodologia, e compara se os resultados apresentados são coerentes com os objetivos definidos no início do trabalho.

CAPÍTULO 2

Fundamentação Teórica

Neste capítulo serão apresentados os temas pertinentes ao projeto e construção do braço robótico, à cinemática direta de manipuladores robóticos; ao acionamento e controle de motor de corrente alternada - CA ou AC (do inglês, *Alternating Current*) - utilizando PIC (microcontrolador) (do inglês, *Programmable Interface Controller*) e comentário sobre a linguagem de programação escolhida, C#.

2.1 DEFINIÇÃO DE ROBÔ

O termo robô foi inicialmente usado em 1921, pelo dramaturgo checo Karen Capek, em sua peça teatral "Os Robôs Universais de Russum (R.U.R)". Em referência a um *autômato*. A palavra “robô” é de origem eslava, significa “trabalho forçado” ([ROMANO; DUTRA, 2002](#)).

Segundo a *Robotics Industries Association* (RIA), **robô industrial** é definido como: "manipulador multifuncional reprogramável projetado para movimentar materiais, partes, ferramentas ou peças especiais, através de diversos movimentos programados, para o desempenho de uma variedade de tarefas" ([RIVIN, 1988](#)).

A *Japanese Industrial Robot Association* (JIRA) define o termo “robô” como qualquer artefato que substitua o trabalho humano ([SOSKA, 1985](#)). Esta associação classifica 6 categorias de robôs:

- 1) Manipulados: possuem vários graus de liberdade, são operados manualmente;
- 2) De Sequência Fixa: manipuladores, desempenham tarefas sucessivas com métodos predeterminados;
- 3) De Sequência Variável: realizam tarefas sucessivas, facilmente modificados;
- 4) Repetitivos: controlados por operador humano, realizam tarefas gravadas;
- 5) De Controle Numérico: o programa de controle do manipulador é inserido no controlador, responsável por controlar suas movimentação;
- 6) Inteligentes: robôs com capacidade de compreender seu ambiente, e se adaptar a mudanças nas circunstâncias.

2.2 DEFINIÇÃO DE BRAÇO ROBÓTICO E MANIPULADOR ROBÓTICO

Na primeira geração de robôs, estes eram dotados apenas de sensores, requeriam ambientes estruturados, eram robôs repetidores e de pouco poder computacional. Denominados de sequência fixa, são programados para realizar determinada sequência de operações, para fazer outra operação devem ser reprogramados. A maioria dos robôs industriais em uso pertencem a essa geração ([ROSEN, 1985](#)).

O manipulador robótico é um dispositivo responsável por posicionar e orientar um objeto existente em sua extremidade. O objetivo é fixar adequadamente a ferramenta conforme a tarefa a ser executada ([ROSARIO, 2002](#)). A definição dada pela *International Organization for Standardization* (ISO), ISO 8373 para manipuladores robóticos: um sistema autônomo, reprogramável, de múltiplos propósitos e possuidor de três ou mais eixos de liberdade, que possa ser ou fixo ou móvel, para aplicações em automação industrial ([ISO, 2012](#)).

Robôs manipuladores, como o da Figura 2.1, tem sua estrutura mecânica composta por uma base, ligamentos e juntas; seus atuadores são motores ou cilindros pneumáticos, por exemplo, que forneçam força para movimentação; controlador que forneça interface com o usuário e ferramenta terminal que permita manipular objetos, ou tarefas específicas.

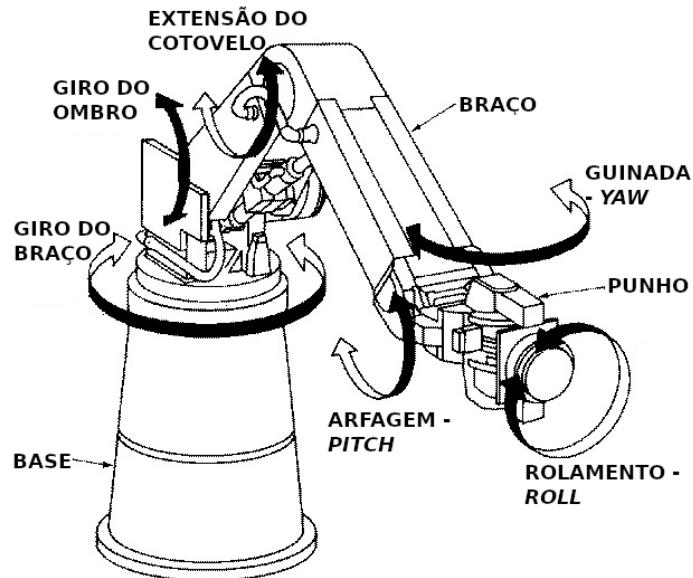
Uma extremidade da corrente é anexada à base enquanto a outra é livre e equipada com uma ferramenta para manipular objetos ou executar tarefas de montagem. Movimentos das juntas resulta em leve movimentação dos ligamentos. Mecanicamente, a composição inclui o braço, o subconjunto do pulso e a ferramenta. O *design* é feito de forma a alcançar uma peça na área de trabalho.

Geralmente, o conjunto do braço tem três graus de liberdade, a combinação desses movimentos posiciona o pulso no espaço de trabalho. O conjunto do pulso geralmente possui três movimentos rotacionais. A combinação desses movimentos orienta a ferramenta de acordo com o objeto. Estes últimos movimentos são denominados de rotação de: giro ou torção (*roll*); lateral ou para os lados (*yaw*); inclinação vertical ou pra cima e pra baixo (*pitch*) ([FU et al., 1987](#)).

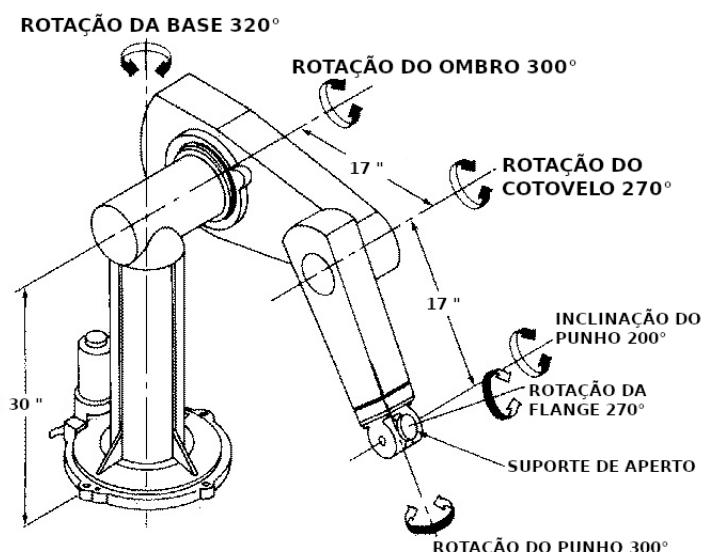
O braço robótico é constituído de no mínimo três graus de liberdade. As juntas dos braços do robô movem-se rapidamente se for utilizado motores elétricos. As máximas velocidades encontram-se entre 6.000 e 9.000 cm/min, velocidade essa muito superior à velocidade de soldagem dos processos a arco elétrico ([ROSARIO, 2002](#)).

Portanto, em um robô que possui seis graus de liberdade, o subconjunto do braço é o mecanismo de posicionamento enquanto o subconjunto do pulso é o mecanismo de orientação. Conceitos ilustrados com o robô Cincinnati Milacron T³, na Figura 2.1 (a) e o braço robótico Unimation PUMA, na Figura 2.1 (b).

Figura 2.1 - Esquema de manipulador robótico.



(a) Braço robótico Cincinnati Milacron T³.



(b) PUMA 560 series braço robótico.

Fonte: Adaptado de FU et al. (1987).

2.3 CINEMÁTICA DIRETA

Cinemática robótica se refere ao estudo analítico de movimentação de um robô manipulador. A cinemática direta não envolve complexidade em obter equações, sempre há solução de cinemática direta para manipulador ([KUCUK; BINGUL, 2006](#)).

Um manipulador robótico é composto de ligamentos em série, fixados um ao outro por meio de juntas de revolução ou prismática, da base até o efetuador (ferramenta). Calcular a posição final do efetuador, em termos das juntas é chamado de cinemática direta ([KUCUK; BINGUL, 2006](#)).

A Tabela 2.1 apresenta o produto da transformação de seis ligamentos. A Equação 2.1 constitui a cinemática do PUMA 560, especifica como calcular a orientação do efetuador em relação à origem do robô. São equações básicas de análise para esse manipulador ([CRAIG, 2005](#)).

$$\begin{aligned}
 r_{11} &= c_1[c_{23}(c_4c_5c_6 - s_4s_5) - s_{23}s_5c_5] + s_1(s_4c_5c_6 + c_4s_6), \\
 r_{21} &= s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] - c_1(s_4c_5c_6 + c_4s_6), \\
 r_{31} &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6, \\
 r_{12} &= c_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] + s_1(c_4c_6 - s_4c_5s_6), \\
 r_{22} &= s_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] - c_1(c_4c_6 - s_4c_5s_6), \\
 r_{32} &= -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6], \\
 r_{13} &= -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5, \\
 r_{23} &= -s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5, \\
 r_{33} &= s_{23}c_4s_5 - c_{23}c_5, \\
 p_x &= c_1[a_2c_2 + a_3c_{23} - d_4s_{23}] - d_3s_1, \\
 p_y &= s_1[a_2c_2 + a_3c_{23} - d_4s_{23}] + d_3c_1, \\
 p_z &= -a_3s_{23} - a_2s_2 - d_4c_{23}.
 \end{aligned} \tag{2.1}$$

Tabela 2.1 - Matriz de transformada homogênea.

r_{11}	r_{12}	r_{13}	p_x
r_{21}	r_{22}	r_{23}	p_y
r_{31}	r_{32}	r_{33}	p_z
0	0	0	1

Fonte: [CRAIG \(2005\)](#).

2.3.1 POSIÇÃO E ORIENTAÇÃO DO EFETUADOR

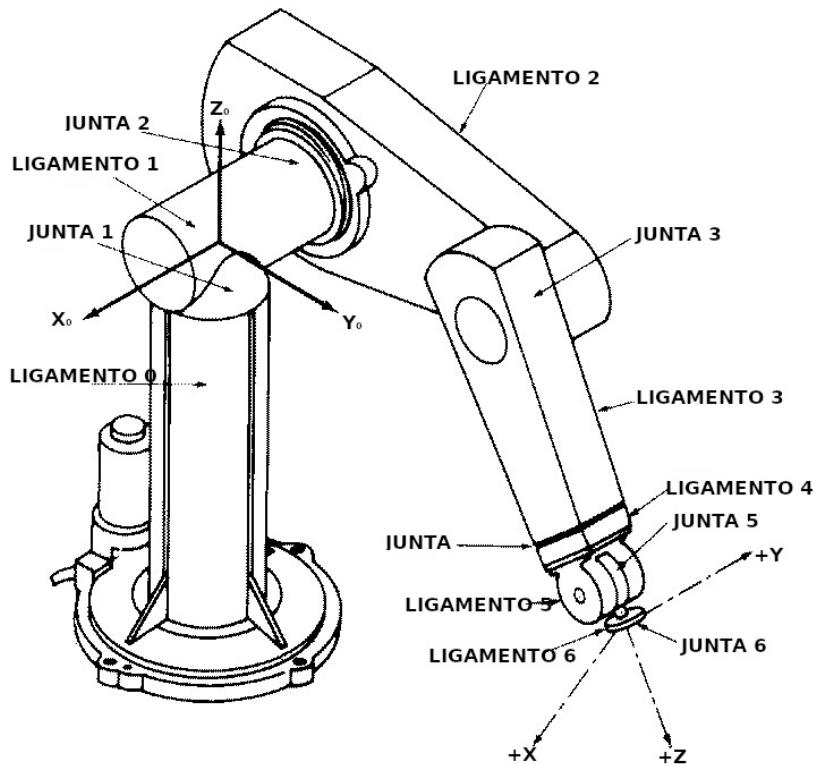
O manipulador mecânico consiste numa sequência de corpos rígidos, os ligamentos, conectados por juntas de revolução ou prismática (Fig. 2.2). O conjunto junta-ligamento constitui 1 grau de liberdade. Para um manipulador com N graus de liberdade existe N pares junta-ligamento, sendo o ligamento 0 (não é considerado parte do robô) fixado à base, onde um sistema de coordenadas inercial é geralmente estabelecido para este sistema dinâmico, no último ligamento é fixado uma ferramenta (FU et al., 1987).

De forma geral dois ligamentos são conectados por um par inferior de juntas, a qual possui duas faces deslizando sobre a outra enquanto em contato (FU et al., 1987). O termo **par inferior** é usado para descrever a conexão entre um par de corpos quando o movimento relativo é caracterizado por duas superfícies deslizando sobre a outra (CRAIG, 2005). Os seis tipos de par inferior de juntas são: revolução (rotação), prismático (deslizante), cilíndrico, esférico, rosca e plana, como apresentado na Figura 2.4. Apenas juntas de revolução e prismáticas são comuns em manipuladores (FU et al., 1987).

O eixo da junta é determinado na conexão de dois ligamentos (Fig. 2.3). Este eixo de junta possui duas normais conectadas a ele, uma para cada ligamento. A posição relativa de dois ligamentos conectados (ligamento $i-1$ e ligamento i) é dada por d_i , distância medida ao longo do eixo da junta entre as normais. O ângulo da junta θ_i entre as normais é medido entre plano normal ao eixo das juntas. Estes parâmetros determinam a posição relativa de ligamentos vizinhos (FU et al., 1987). A Figura 2.5 apresenta a localização dos sistemas $\{i-1\}$ e $\{i\}$ para manipulador genérico (CRAIG, 2005).

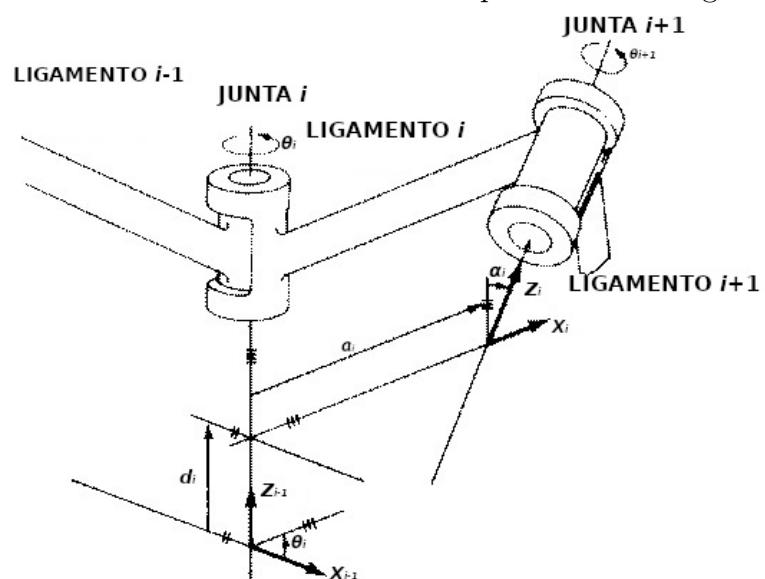
O parâmetro a_i é a menor distância medida ao longo da normal em comum entre eixos de juntas (eixos z_{i-1} e z_i das juntas i e $i+1$ por exemplo), α_i é o ângulo entre eixos de juntas medido no plano perpendicular a a_i . Estes parâmetros determinam a estrutura do ligamento i (FU et al., 1987).

Figura 2.2 - Esquema Estrutura de um Manipulador.



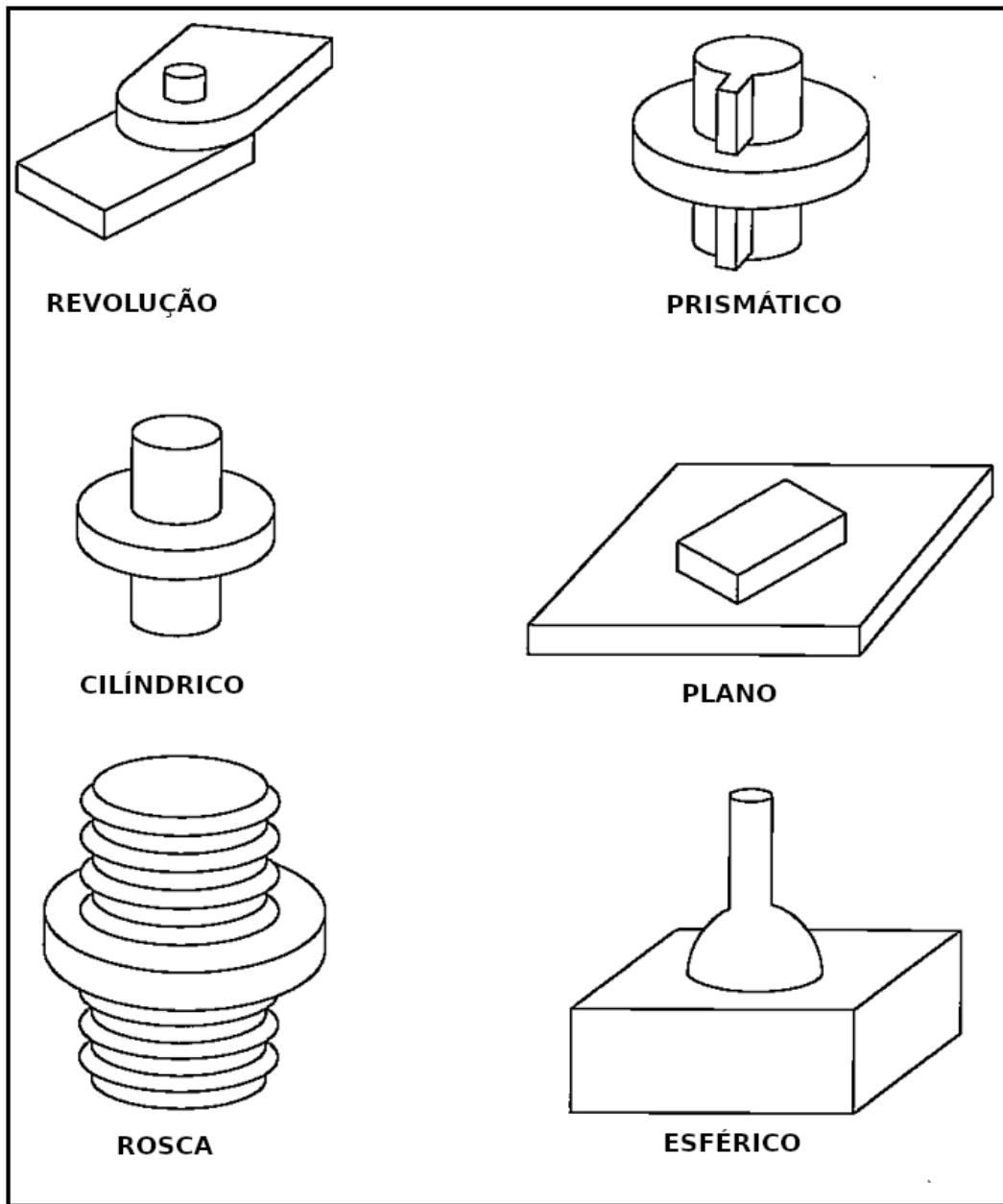
Fonte: [Adaptado de Fu et al \(1987\)](#).

Figura 2.3 - Sistema de coordenadas e parâmetros de ligamento.



Fonte: [Adaptado de Fu et al \(1987\)](#).

Figura 2.4 - Seis possíveis par inferior de juntas.



Fonte: Adaptado de CRAIG (2005).

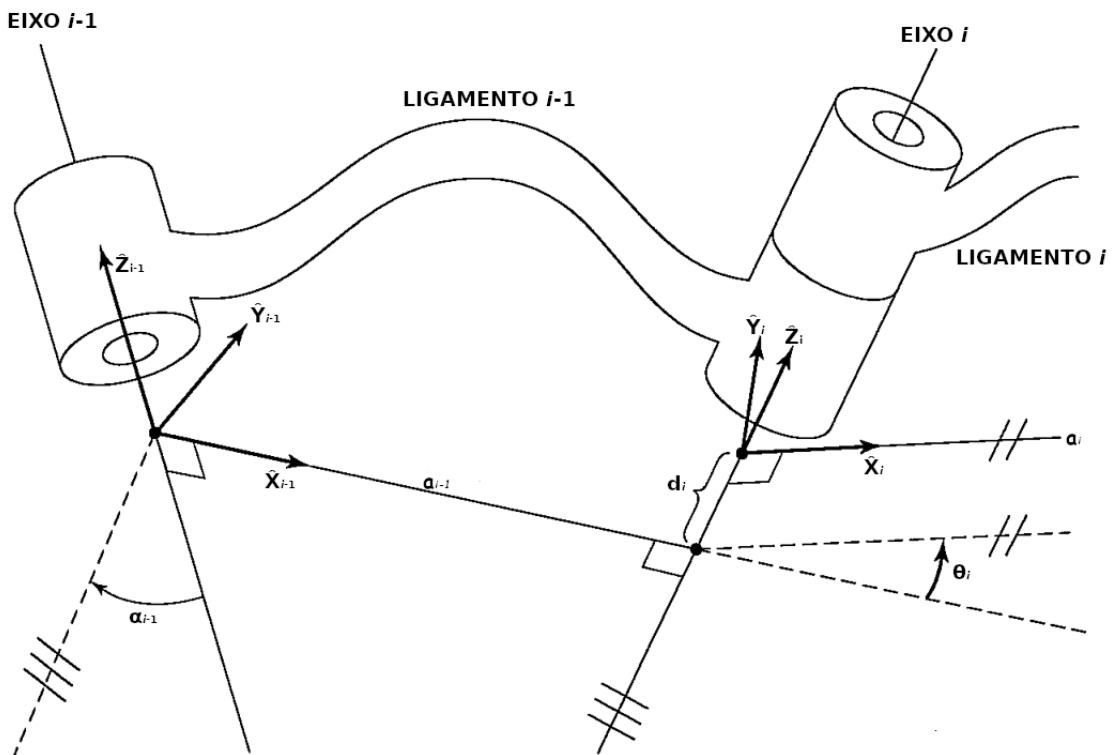
Craig (2005) afirma que os seguintes itens constituem resumo do procedimento a seguir para definir o sistema de ligamentos:

- 1) Identificar os eixos das juntas e imaginar (ou desenhar) linhas infinitas ao longo destes. Para os passos 2 a 5 abaixo, considerar duas dessas linhas vizinhas (nos eixos i e $i+1$).
- 2) Identificar a perpendicular em comum entre eles, ou o ponto de intersecção. No ponto de intersecção, ou no ponto onde a perpendicular em comum encontra o

i -ésimo eixo, defina a origem do sistema de ligamentos.

- 3) Atribuir o eixo \hat{Z}_i no sentido do i -ésimo eixo da junta.
- 4) Atribuir o eixo \hat{X}_i no sentido da perpendicular em comum, ou, se os eixos intersectam, atribuir \hat{X}_i normal ao plano que contém os dois eixos.
- 5) Atribuir o eixo \hat{Y}_i para completar o sistema de coordenadas (regra da mão direita).
- 6) Atribuir $\{0\}$ para corresponder $\{1\}$ quando a variável da primeira junta for zero. Para $\{N\}$, escolher um local de origem e direção de \hat{X}_N livremente, mas geralmente de forma a zerar o máximo de parâmetros do acoplamento que for possível.

Figura 2.5 - Sistemas de ligamentos são acoplados de forma que o sistema i é conectado rigidamente ao ligamento i .



Fonte: [Adaptado de CRAIG \(2005\)](#).

2.3.2 NOTAÇÃO DE DENAVIT-HARTENBERG

A notação Denavit-Hartenberg (D-H) é uma ferramenta utilizada para sistematizar a descrição de sistemas mecânicos articulados com N graus de liberdade. Denavit-Hartenberg propuseram método matricial para estabelecimento de sistema de coordenadas fixo para cada ligamento de uma cadeia cinemática articulada ([DENAVIT; HARTENBERG, 1955 apud ROSARIO, 2002](#)).

A representação D-H resulta na obtenção de matriz 4x4, de cada sistema de coordenadas do ligamento na junta, em relação ao ligamento anterior. Deste modo, através de transformações sucessivas podem ser obtidas as coordenadas do elemento terminal de um robô ([ROSARIO, 2002](#)).

Quando a junta i é acionada, o ligamento i move-se em relação ao ligamento $i - 1$. O i -ésimo sistema de coordenadas também se movimenta. O n -ésimo sistema de coordenadas se movimentará com o elemento terminal - ligamento n . Sistemas de coordenadas são estabelecidos obedecendo três regras ([ROSARIO, 2002](#)):

- 1) Eixo Z_{i-1} é colocado ao longo do eixo de movimento da junta i ;
- 2) Eixo X_i é normal ao eixo Z_{i-1} , apontando para fora dele;
- 3) Eixo Y_i completa o sistema utilizando a regra da mão direita.

Nesta convenção, cada transformada homogênea A_i é representada como produto de quatro transformações básicas (Equação 2.2) ([SPONG et al., 2006](#)).

Os quatro parâmetros, a_i , α_i , d_i , e θ_i são comumente denominados comprimento do ligamento, torção do ligamento, deslocamento do ligamento e ângulo da junta, respectivamente. A matriz A_i é uma função de uma variável, em que três dos quatro parâmetros mencionados são constantes para determinado ligamento, enquanto o quarto parâmetro θ_i para junta de revolução e d_i na junta prismática, é a variável da junta ([SPONG et al., 2006](#)).

Uma matriz de transformação homogênea arbitrária pode ser caracterizada por seis números, por exemplo, três números para especificar a quarta coluna, e três ângulos de Euler para especificar a parte superior esquerda 3x3, matriz de rotação ([SPONG et al., 2006](#)). Na representação D-H são necessários apenas quatro parâmetros. Isso se deve ao fato de enquanto o sistema i deve estar fisicamente conectado ao ligamento i , há liberdade em escolher a origem do sistema de coordenadas deste sistema. Por exemplo não é necessário a origem o_i do sistema i estar localizado no final do ligamento i . Portanto, com a escolha

inteligente da origem do sistema de coordenadas, é possível diminuir a quantidade de parâmetros de seis para quatro (SPONG et al., 2006).

$$\begin{aligned}
A_i &= \text{Rot}_{z,\theta_i} \text{Trans}_{z,d_i} \text{Trans}_{x,a_i} \text{Rot}_{x,\theta_i} & (2.2) \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

2.3.3 ÂNGULOS RPY

Esse método é utilizado para descrever a orientação do sistema B. A partir do início com sistema coincidente com sistema de referência A. Rotação de B primeiro em torno de \hat{X}_A ângulo de γ , em seguida, em torno de \hat{Y}_A ângulo de β , finalmente em torno de \hat{Z}_A ângulo de α .

Cada uma das três rotações ocorre em um eixo da referência fixa A. Esta convenção é denominada **X-Y-Z ângulo fixos**. As rotações são especificadas em torno do sistema de referência fixo (Figuras 2.6 e 2.7). Esta convenção também é denominada como **ângulos roll, pitch, yaw**.

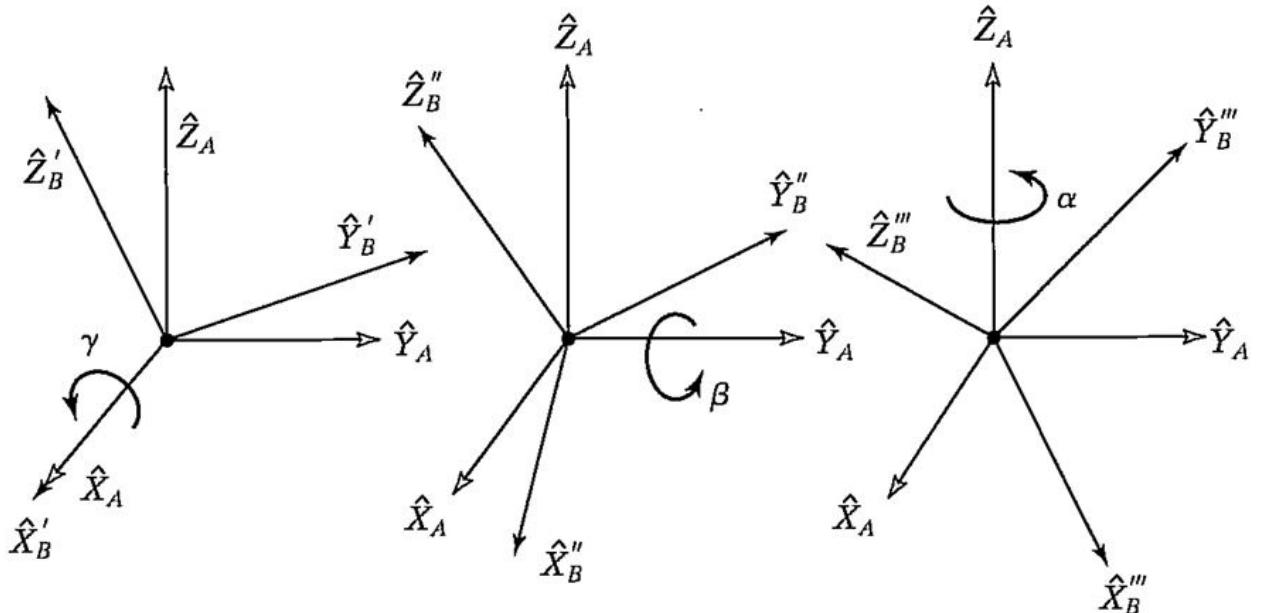
A derivação da matriz de rotação equivalente ${}^A_B R_{XYZ}(\gamma, \beta, \alpha)$, é direta, e todas rotações ocorrem em torno de eixos do sistema de referência (Equação 2.3). Aplicando rotações (a partir da direita) de R_X (γ), em seguida R_Y (β) e depois R_Z (α). Multiplicando as matrizes da Equação 2.3 resulta na Equação 2.4.

$${}^A_B R_{XYZ}(\gamma, \beta, \alpha) = R_Z(\alpha)R_Y(\beta)R_X(\gamma) \quad (2.3)$$

$$= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix}$$

$${}^A_B R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} c_\alpha c_\beta & c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \quad (2.4)$$

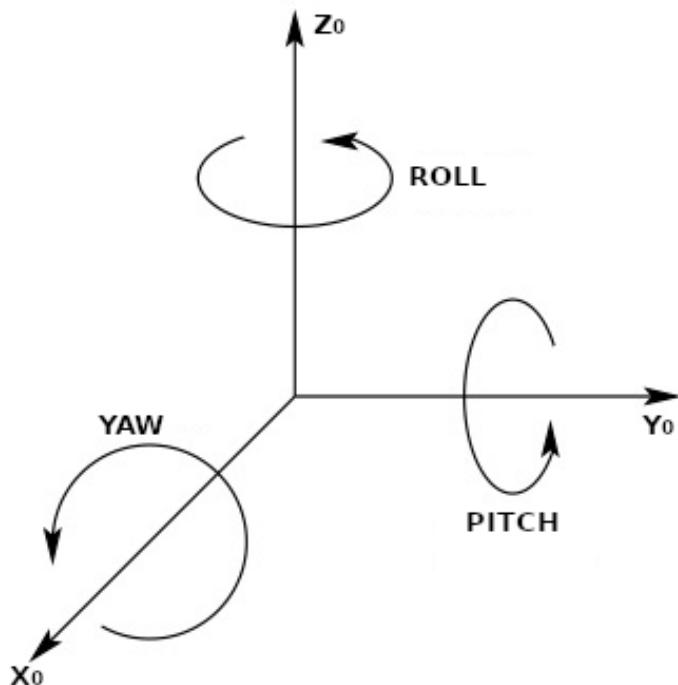
Figura 2.6 - X - Y - Z ângulos fixos. Rotações na ordem $R_X(\gamma)$, $R_Y(\beta)$, $R_Z(\alpha)$.



Fonte: [Craig. \(2005, p.42\)](#).

A definição apresentada nesta seção, especifica a ordem de três rotações. A Equação 2.4 é correta somente para rotações na seguinte ordem: \hat{X}_A ângulo de γ , \hat{Y}_A ângulo de β , \hat{Z}_A ângulo de α .

Figura 2.7 - Ângulos Roll, Pitch, Yaw - Rotações Elementares.



Fonte: Adaptado de Spong. (2006, p.50).

2.4 ACIONAMENTO E CONTROLE DO MOTOR UNIVERSAL

O motor série universal tem a capacidade de funcionar com correntes alternada ou contínua apresentando características similares. Pequenos motores universais são usados onde baixo peso é importante, como aspiradores de pó, eletrodomésticos e ferramentas portáteis, funcionando com velocidades elevadas (1.500 a 15.000 rpm) (KINGSLEY et al., 2014). Algumas modificações são necessárias para transformar um motor CC em um motor universal (GURU; HIZIROGLU, 2001).

Para controlar a velocidade de um motor universal pode ser feito pelo controle da tensão terminal de armadura com uso de sistemas inversores baseados em eletrônica de potência (KINGSLEY et al., 2014). O circuito ponte H é amplamente utilizado em servos amplificadores e inversores monofásicos, e tem propriedades semelhantes ao do conversor *buck* (abaixador), do qual é originado (ERICKSON; MAKISIMOVIC, 2004).

Para fazer a reversão de sentido do giro do motor foi utilizado uma ponte H. Uma das vantagens em utilizar a ponte H é não precisar de utilizar tensões negativas ou desconectar os terminais para que sejam trocados.

2.4.1 MOTOR UNIVERSAL

O motor universal é uma máquina CC com enrolamento de campo ligado em série (Fig. 2.8). Fluxo de eixo direto ϕ_d é proporcional à corrente de armadura. A tensão gerada E_a é proporcional ao produto da corrente de armadura pela velocidade do motor, e o conjugado será proporcional ao quadrado da corrente de armadura (KINGSLEY et al., 2014).

Uma característica típica de velocidade *versus* conjugado para esse motor quando está ligado em série e operando em condições CC é apresentado na Figura 2.9. Conjugado depende apenas do valor da tensão de armadura e não sua polaridade. Se a polaridade da tensão aplicada for invertida, o valor ou sentido do conjugado aplicado não serão alterados (KINGSLEY et al., 2014).

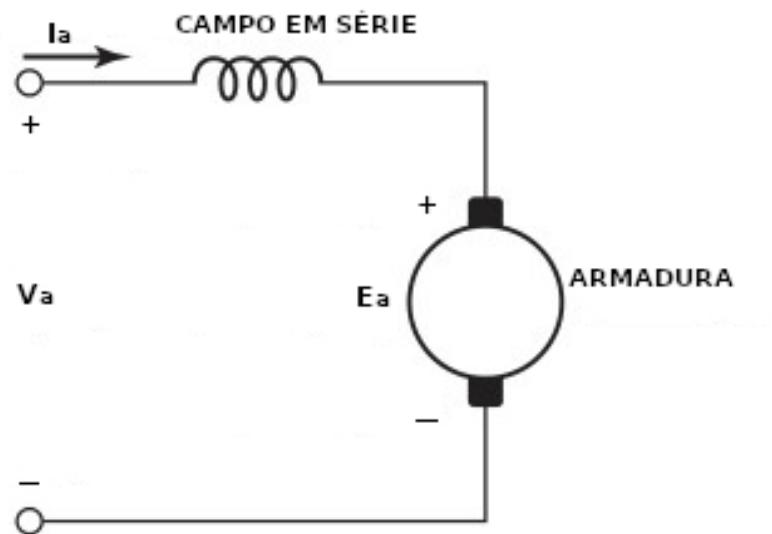
Para controlar a velocidade e o conjugado de um motor universal, pode-se variar a tensão CA aplicada usando elemento de chaveamento eletrônico denominado TRIAC. O ângulo de disparo do TRIAC pode ser ajustado de modo manual, ou comando por circuito de controle de velocidade (KINGSLEY et al., 2014).

Algumas das razões da utilização de motores universais são listadas a seguir (GURU; HIZIROGLU, 2001):

- 1) Quando há necessidade de operar com fonte CC e CA.
- 2) Operações em fonte CA em velocidades superior a 3600 rpm (motor de indução de dois polos 60Hz). A potência desenvolvida é proporcional à velocidade do motor, um motor de alta velocidade desenvolve mais potência para o mesmo tamanho do que um motor de baixa velocidade.
- 3) Quando é preciso um motor que ajuste velocidade automaticamente em relação a carga. Para o motor universal, a velocidade é alta quando a carga é leve, e baixa quando a carga é alta.

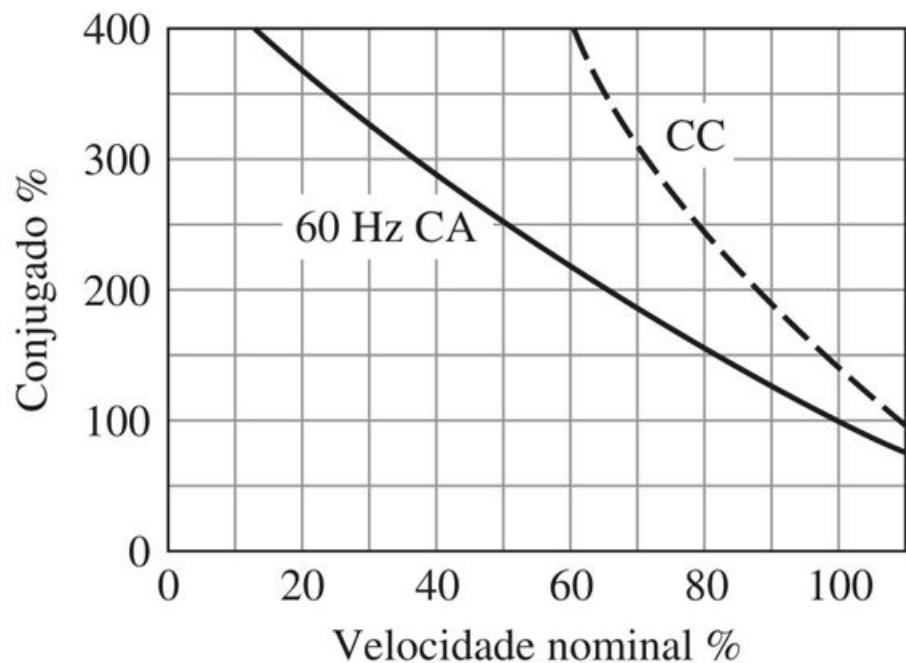
O motor universal é usado extensivamente na faixa de potência fracionária. São utilizados em aplicações que requerem variação da velocidade em relação a carga como motosserra, máquina de costura, máquinas portáteis e aspiradores de pó (GURU; HIZIROGLU, 2001).

Figura 2.8 - Esquema motor CC excitação em série.



Fonte: Adaptado de Kingsley et al. (2014).

Figura 2.9 - Conjugado *versus* velocidade de motor série universal .



Fonte: Kingsley et al. (2014).

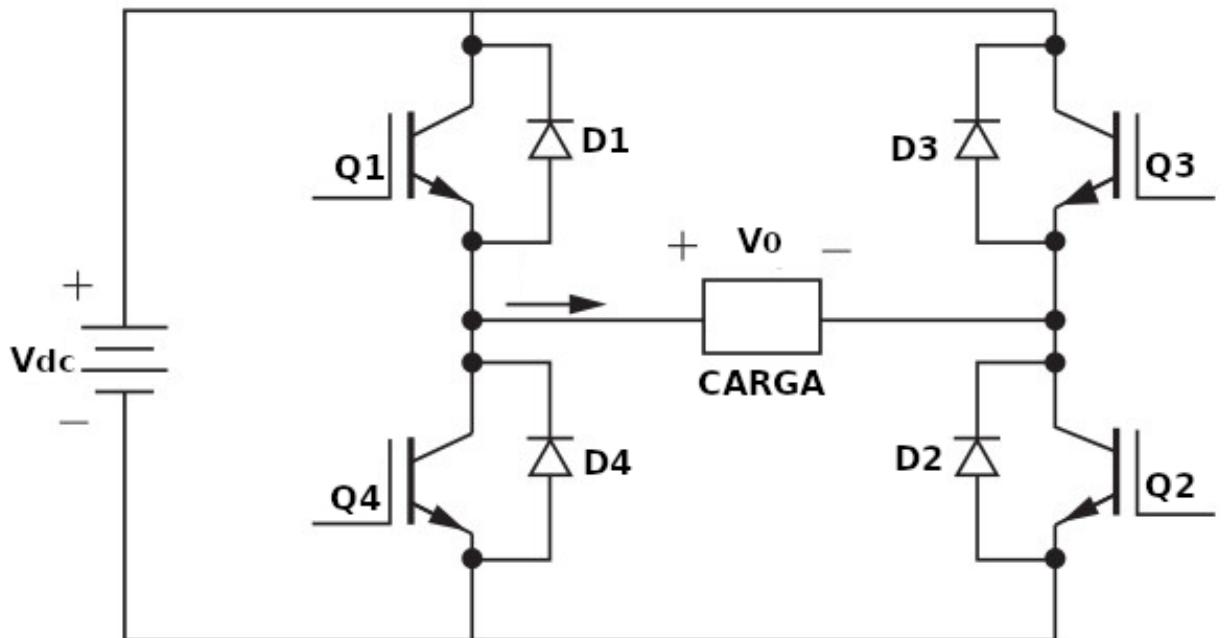
2.4.2 PONTE H

O circuito denominado ponte H, ou circuito ponte inversora, é apresentado na Figura 2.10. Os diodos são polarizados inversamente quando a corrente é positiva na chave. O chaveamento é implementado como *Insulated Gate Bipolar Transistors* (IGBTs) com diodos de *feedback*. Corrente do transistor e diodo para tensão de onda quadrada e carga RL é indicado na Figura 2.11 (HART, 2011).

Quando IGBTs Q_1 e Q_2 estão desativados, a corrente de carga deve ser contínua e irá transferir para os diodos D_3 e D_4 , fazendo com que a tensão de saída $-V_{dc}$, efetivamente acione o chaveamento 3 e 4 antes de Q_3 e Q_4 serem acionados. IGBTs Q_3 e Q_4 devem estar acionados antes da corrente de carga cair para zero (HART, 2011).

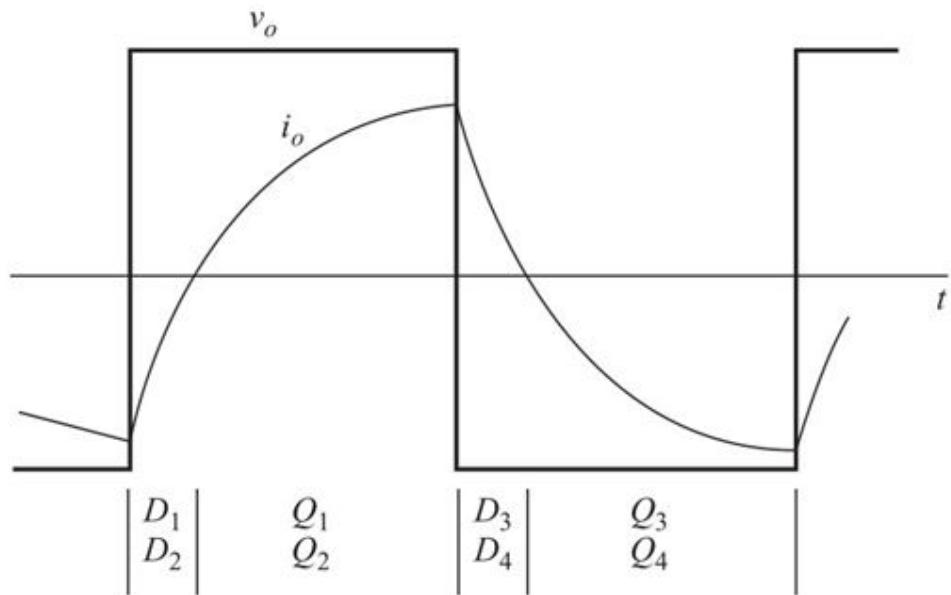
O controle da tensão de armadura (Fig. 2.12) tira vantagem do fato de que, a queda de tensão na resistência de armadura é pequena, em regime permanente, assim, variação na tensão de terminal de armadura em um motor em derivação irá resultar em variação substancial na tensão de velocidade. Com a queda de corrente de campo em derivação constante, fluxo é constante, a mudança na tensão de velocidade será acompanhada por variação proporcional na velocidade do motor. Portanto, a velocidade do motor pode ser controlada diretamente por meio da tensão terminal de armadura (KINGSLEY et al., 2014).

Figura 2.10 - Circuito ponte H.



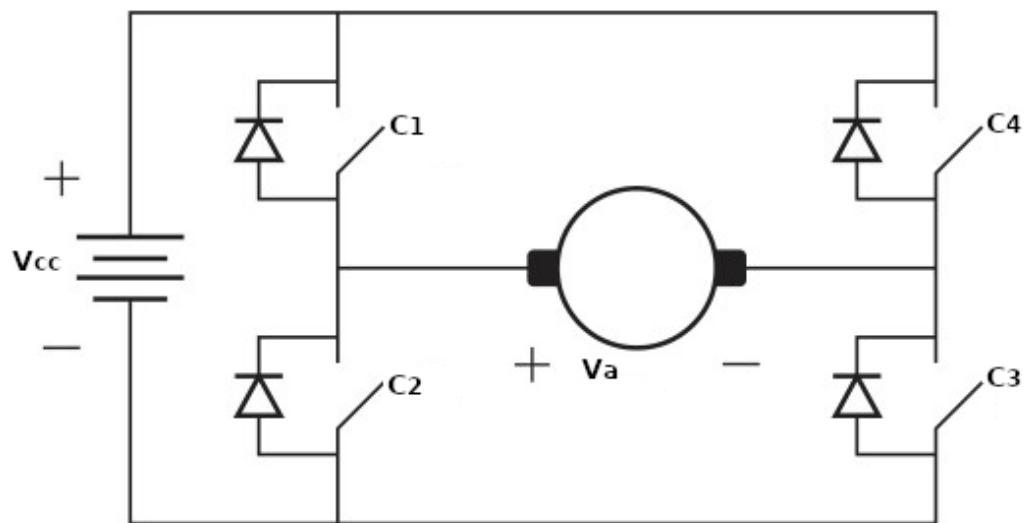
Fonte: Adaptado de Hart (2011).

Figura 2.11 - Corrente de estado constante para carga RL.



Fonte: [Hart \(2011\)](#).

Figura 2.12 - Controle pela tensão de terminal de armadura implementado com inversor de ponte H completa.



Fonte: [Adaptado de Kingsley et al. \(2014\)](#).

2.4.3 CONTROLADOR PID

O controlador PID é o algoritmo de controle mais popular. A maior parte dos sistemas realimentados são controlados por esse algoritmo ou pequenas variações dele. Pode ser implementado de diferentes formas, como controlador independente ou parte de pacote DDC (do inglês, *Direct Digital Control*) ou sistema de controle de processo distribuído hierárquico ([ÅSTRÖM; HÄGGLUND, 1995](#)).

Alguns dos aspectos importantes de implementação em computador digital de controlador PID são: questões como pré-processamento, diferentes aproximações digitais, filtro de ruído, e código computacional para implementação adequada. Aspectos operacionais, como transferência sem que ocorra problemas entre o modo manual e automático e entre diferentes definições de parâmetros ([ÅSTRÖM; HÄGGLUND, 1995](#)).

O controlador PID ideal no tempo contínuo para processo SISO (do inglês, *single-input single-output system*) é expresso no domínio de Laplace conforme a Equação 2.5. K_c é o ganho proporcional, T_i a constante de tempo integral e T_d constante de tempo derivativo. Se $T_i = \infty$ e $T_d = 0$ (controle proporcional), fica claro que o valor medido na malha fechada y sempre será menor do que o valor desejado r (para processos sem o termo integral, como erro positivo é necessário para manter o valor medido constante, e menor do que o valor desejado). A introdução da ação integral facilita atingir igualdade entre o valor medido e o valor desejado, conforme o erro constante produz aumento na saída do controlador. A introdução da ação derivativa significa que mudanças no valor desejado podem ser antecipados, e portanto correção apropriada pode ser adicionada antes da mudança real. Portanto, em termos simplificados, o controlador PID permite contribuições de entradas atuais, passadas e futuras ([O'DWYER, 2009](#)).

$$\begin{aligned} U(s) &= G_c(s)E(s) \\ G_c(s) &= K_c\left(1 + \frac{1}{T_i s} + T_d s\right) \end{aligned} \tag{2.5}$$

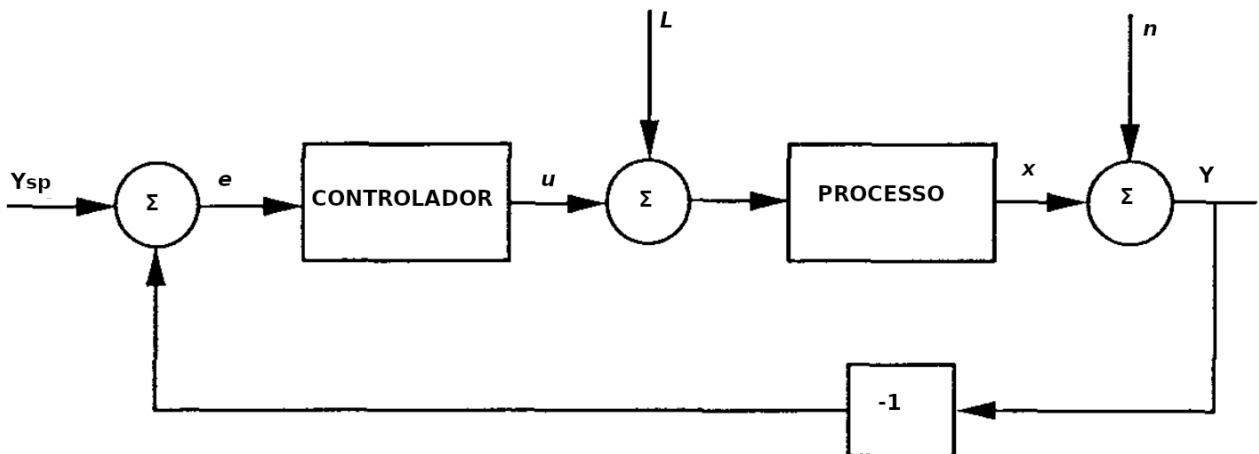
Os algoritmos de controle PID são geralmente mais complexos do que o controlador proporcional. Dentro da faixa proporcional o comportamento do algoritmo PID pode ser descrito conforme a Equação 2.6, em que u é a variável de controle, e a variável erro ($e = y_{sp} - y$). A variável de controle é portanto a soma de três termos: o termo P (proporcional ao erro), o termo I (proporcional à integral do erro) e o termo D (proporcional à derivada do erro) ([ÅSTRÖM; HÄGGLUND, 1995](#)).

$$u(t) = K(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt}) \quad (2.6)$$

A Figura 2.13 apresenta sistema simples realimentado, composto por um processo e controlador. Assumindo que o controlador tenha ação proporcional e que o modelo do processo é estático, e analisando o diagrama de blocos as Equações 2.7 são obtidas, x é a variável de processo, u a variável controlada, l o distúrbio, e K_p o ganho estático do processo (ÅSTRÖM; HÄGGLUND, 1995).

$$\begin{aligned} x &= K_p(u + l) \\ y &= x + n \\ x &= K_p(u + l) \\ u &= K(y_{sp} - y) + u_b \end{aligned} \quad (2.7)$$

Figura 2.13 - Diagrama de bloco de um sistema realimentado simples.



Fonte: Adaptado de ÅSTRÖM; HÄGGLUND (1995).

2.5 LINGUAGEM DE PROGRAMAÇÃO C#

A linguagem C# foi originalmente desenvolvida por um time de engenheiros da Microsoft, liderados por Anders Hejlsberg e Scott Wiltamuth ([LIBERTY; XIE, 2008](#)). C# é a linguagem mais utilizada da plataforma .NET, oferece o mesmo poder que C++ e é a linguagem nativa para a plataforma da Microsoft ([LIMA, 2002](#)). C# é uma linguagem moderna, orientada a objetos, e de alto nível. Os programas consistem de definições em classes que contêm métodos, os quais contêm a lógica do programa - instruções executadas ([NAKOV, 2013](#)).

A linguagem C# é distribuída com o ambiente no qual é executada, *Common Language Runtime* (CLR). Este ambiente faz parte da plataforma .NET Framework. Esta plataforma também inclui pacotes de bibliotecas provendo funcionalidades básicas, compiladores, *debuggers* e outras ferramentas de desenvolvimento. A estrutura CLR garante a portabilidade de programas escritos em C# para diferentes plataformas de *hardware* e sistemas operacionais. A plataforma .NET Framework consiste de um ambiente para desenvolvimento e execução de programas escritos em C# ou alguma outra linguagem compatível com .NET como VB.NET, *Managed C++*, J# ou F# ([NAKOV, 2013](#)).

Nakov (2013) afirma ainda que, a plataforma .NET Framework contém os seguintes elementos:

- Linguagens de programação .NET (C#, VB.NET e outras);
- Ambiente para execução de código gerenciado (CLR), que executa programas C# de forma controlada;
- Ferramentas de desenvolvimento, como o compilador csc, converte programas C# em código intermediário (denominado MSIL) em que o CLR consegue entender;
- Bibliotecas padrão, como ADO.NET permite acesso a banco de dados (como MS SQL Server ou MySQL) e WCF que conecta aplicações por meio de estruturas padrão de comunicação e protocolos como HTTP, REST, JSON, SOAP e TCP *sockets*.

Alguns dos benefícios de programar em C# são: possui muitas funções na biblioteca; elimina automaticamente funções e objetos sem valor atribuído, liberando memória na máquina; facilidade operacional; funciona em computadores com Windows e também outros sistemas como Mac, Linux e outros, desde que seja instalado o pacote .NET; é similar às linguagens C e C++ ([CLARK, 2016](#)).

Características de C# referentes ao desenvolvimento de programas: todos programas existem dentro de uma classe; esta linguagem de programação é *case sensitive* (diferencia maiúsculas de minúsculas); o compilador ignora caracteres espaços em branco, criados para facilitar a leitura do código; não é necessário que o nome do programa seja o mesmo da classe identificadora; pode-se usar a função Main() mais de uma vez no mesmo programa ([CLARK, 2016](#))f.

C# não requer ponteiros para alocar/desalocar áreas de memória no *heap*. Esse gerenciamento é feito pelo *Garbage Collector* (GC). C# suporta interfaces, sobrecarga, herança, polimorfismo, atributos, propriedades, coleções e outras características essenciais de uma linguagem orientada a objetos. Todo programa desenvolvido em C# é passível de reutilização a partir de qualquer outra linguagem de programação ([LIMA, 2002](#)).

Algumas das dificuldades encontradas no desenvolvimento do projeto para o sistema operacional Windows incluem complexidade associada à linguagem de programação, pouco reaproveitamento do código com linguagens de programação diferentes; falta de portabilidade do código executável. Para solucionar alguns desses problemas, as características de .NET incluem: simplicidade; sistemas auto-explicativos e controle de versões; tempo de execução compartilhado; independência da linguagem de programação ([LIMA, 2002](#)).

CAPÍTULO 3

Metodologia

Este trabalho é dividido em três etapas: a primeira projeto, a segunda construção, a terceira avaliação. A etapa de projeto é dividida em quatro subetapas: projeto mecânico utilizando SolidWorks®, projeto eletrônico, interface de controle (supervisório), e sistema de movimentação. Essas etapas demandam recursos computacionais, *softwares* e *hardware* para serem realizadas. Em seguida, após adquirido os materiais necessários (como aço inox, parafusos, motores universais, rolamentos, entre outros) é dado início à fase de construção do manipulador robótico. A etapa final terá os testes de funcionamento e integração do projeto. Etapas do presente trabalho:

- Projeto do manipulador robótico acadêmico;
- Construção do manipulador robótico;
- Testes de funcionamento (Avaliação).

3.1 PROJETO MECÂNICO

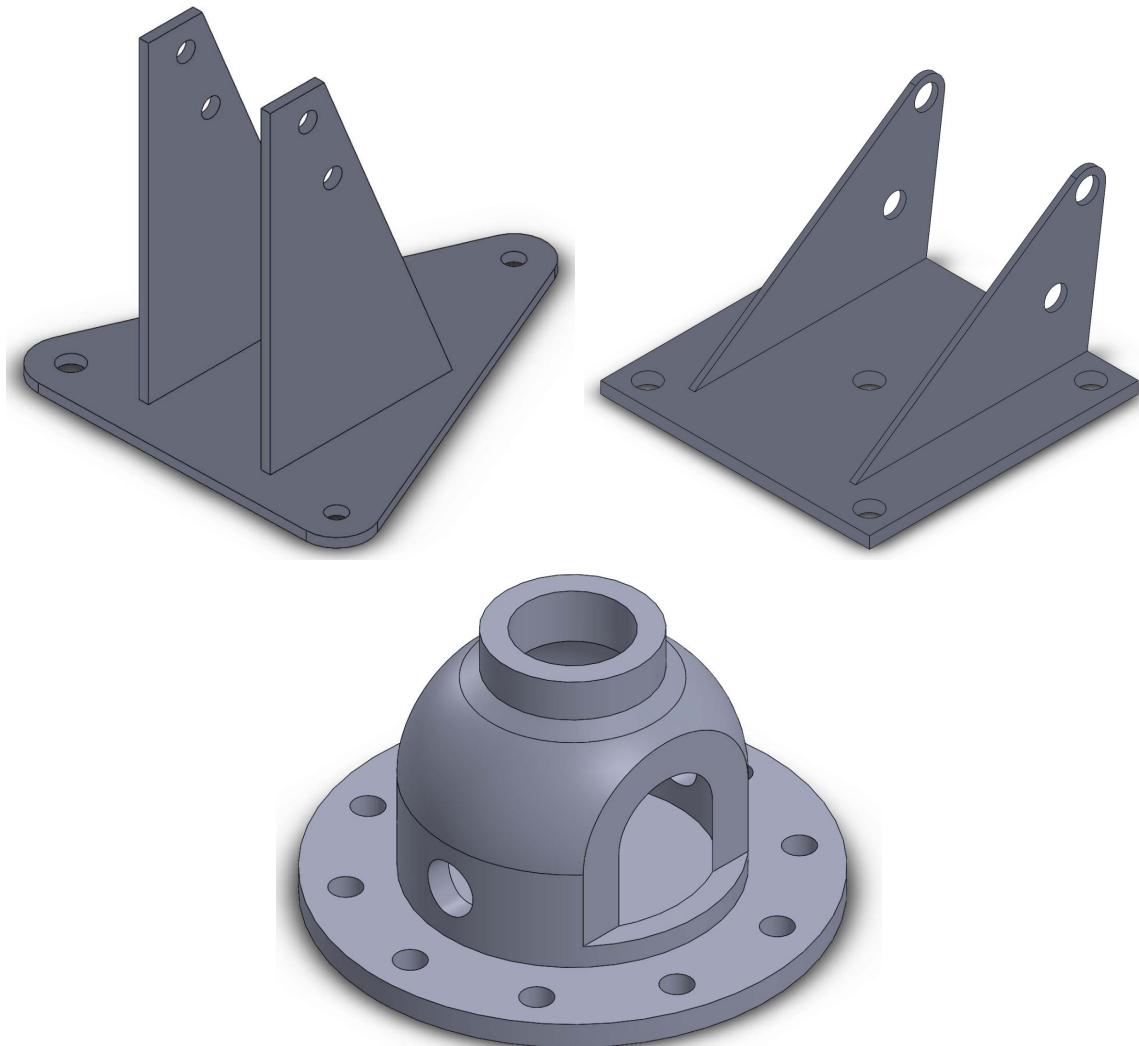
O projeto mecânico do manipulador robótico foi feito em *software CAD*. Este projeto foi modificado e atualizado conforme desenvolvimento do TCC, dessa forma novas ideias e ajustes foram incorporados ao projeto. A vantagem da simulação é eliminar possíveis erros existentes no projeto, e visualizar o funcionamento do mesmo. O Solidworks® é o *software* de computação gráfica utilizado para desenhos de máquinas, possui recursos como: criação de peças, montagens e simulação da própria estrutura física produzindo esforços e deformações. A Figura 3.2 mostra a tela inicial do Solidworks® com as três opções principais: desenvolvimento de peça, montagem ou desenho técnico.

A Figura 3.3 mostra as principais etapas para desenvolvimento da estrutura mecânica do manipulador. Inicialmente é criado desenho 2D selecionando a opção **Nova Peça** ou pode ser a própria reprodução de um desenho 2D feito manualmente em folha de papel. Em seguida é utilizado a opção **Extrusão** para visualizar a peça em 3D. Depois de finalizadas as peças, a opção **Nova Montagem** permite junção das partes desenvolvidas. Posteriormente para representação de desenho técnico em prancheta a opção **Novo Desenho Técnico** pode ser utilizada para montagem ou peça individual.

O projeto mecânico teve início em esboços desenhados no papel. Com base neste escopo, a utilização do *software* possibilitou criar peças e interagir uma com as outras através dos recursos de montagem e simulações avançadas.

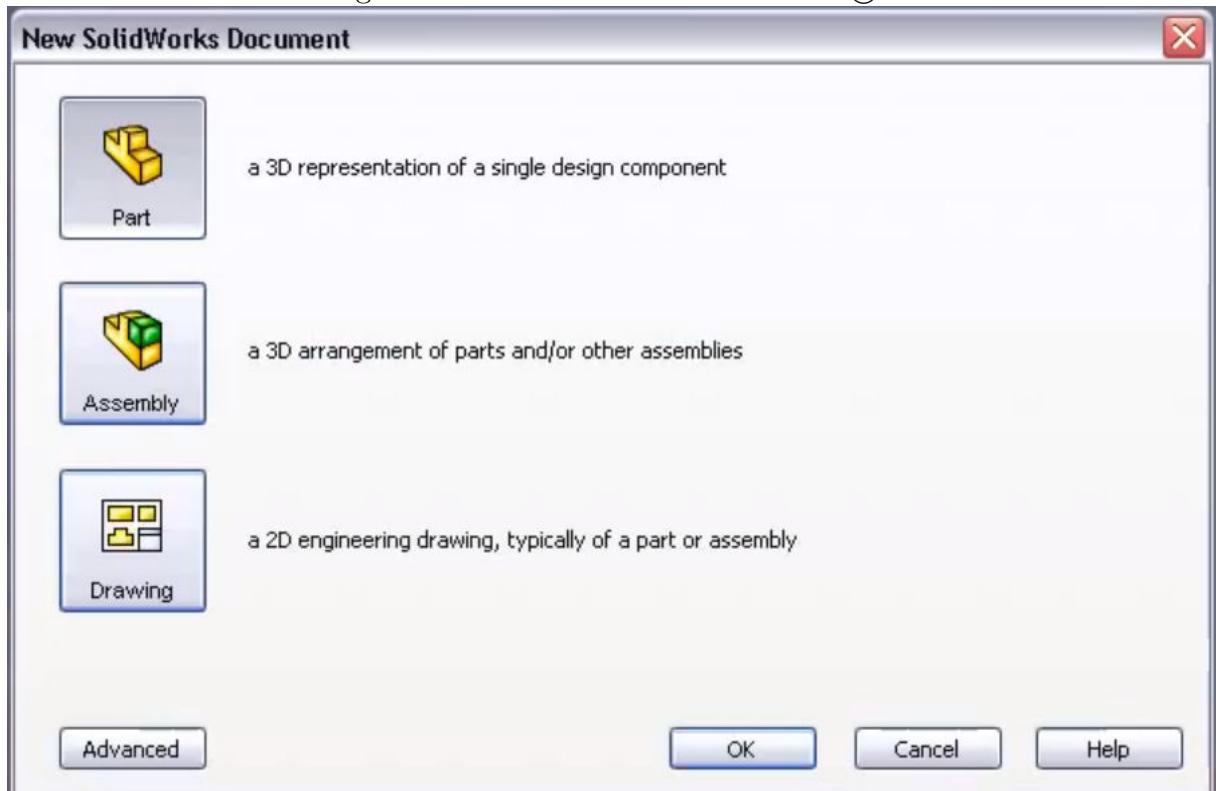
A evolução do projeto se deu também com o processo inverso. Em vez de desenhar e depois reproduzir a peça, à medida que se obtinha uma peça de sucata que poderia ser útil ao projeto, foram incorporadas durante o desenvolvimento, as medidas eram aferidas e inseridas ao projeto 3D. A Figura 3.1 consiste em exemplos de peças que foram obtidas, reproduzidas e adequadas ao projeto permitindo a mudança do projeto. Baseado nessas adequações sucessivas criou-se três versões do projeto mecânico até a versão final.

Figura 3.1 - Exemplo de peças adequadas ao projeto.



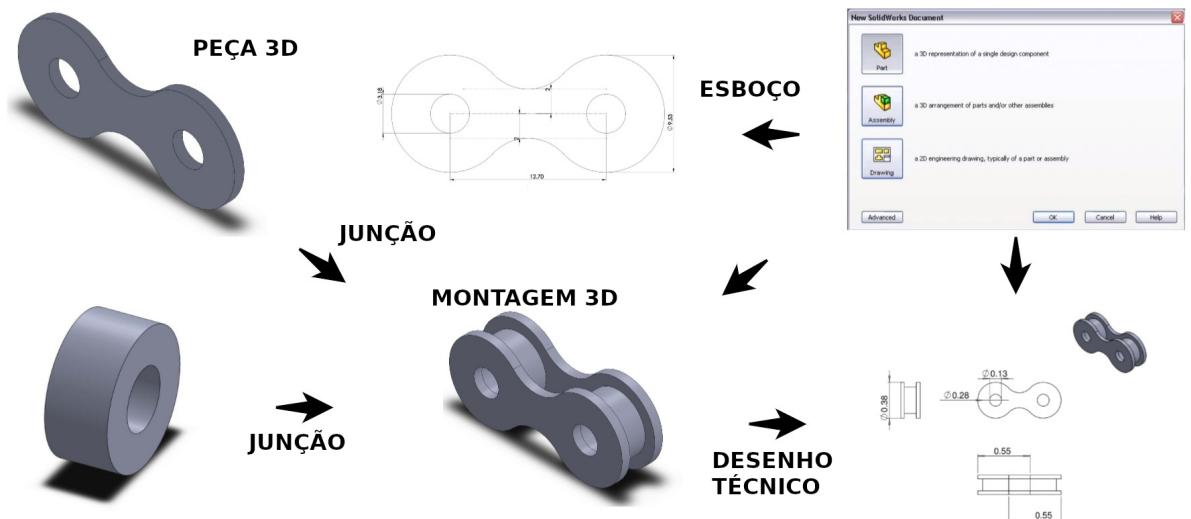
Fonte: Própria (2018).

Figura 3.2 - Tela inicial do Solidworks®.



Fonte: Própria (2018).

Figura 3.3 - Principais etapas do desenvolvimento da estrutura mecânica.

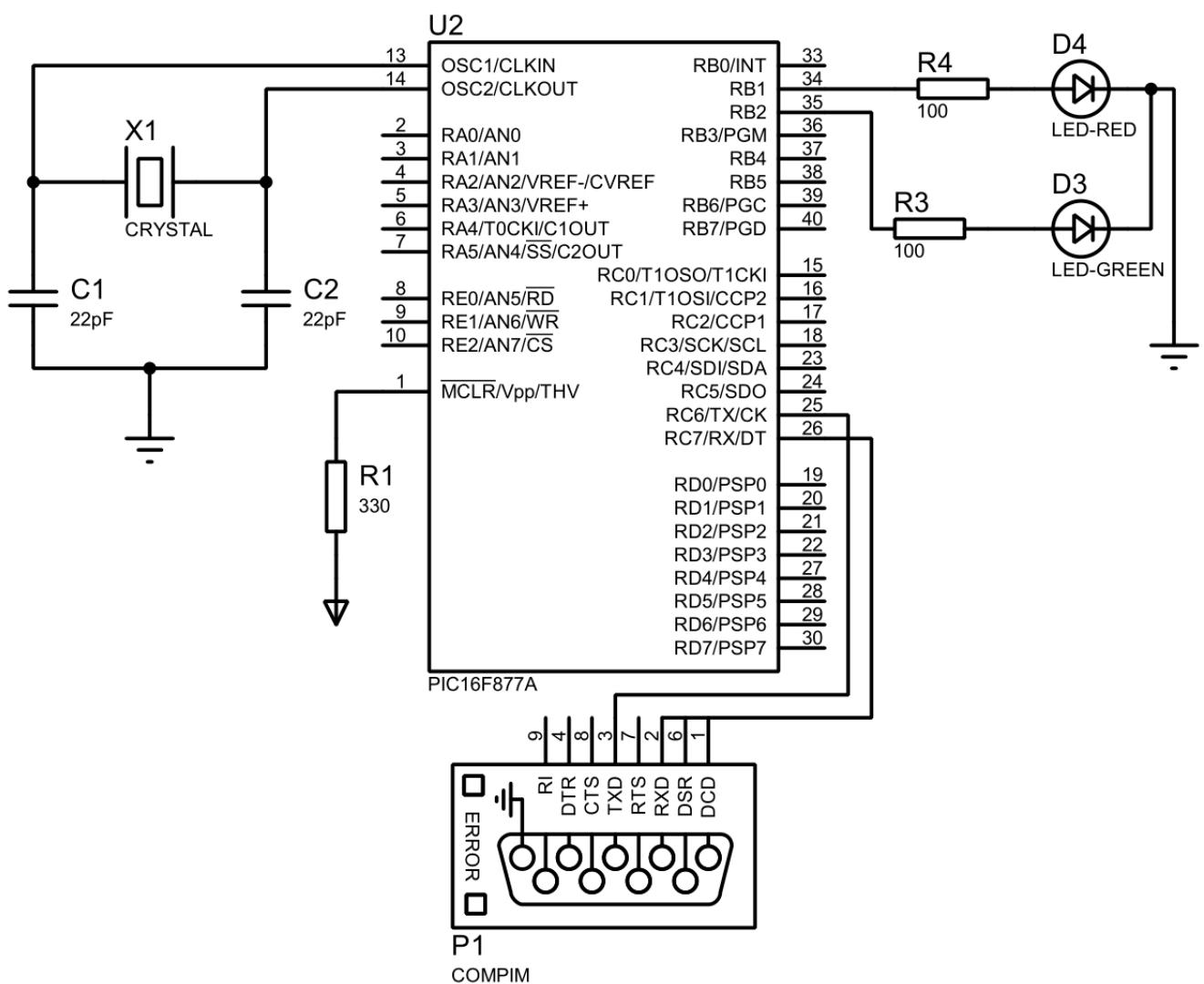


Fonte: Própria (2018).

3.2 PROJETO CIRCUITOS ELETROELETRÔNICO

A simulação do circuito eletrônico, responsável por acionar e inverter a rotação de cada motor do manipulador robótico, foi desenvolvida e modificada para utilizar o menor número de microcontroladores, preservando o funcionamento desejado. O circuito base desse projeto foi desenvolvido no Proteus® (versão de demonstração) - *software de design, simulação e testes de circuitos impressos* (Fig. 3.4). Contendo microcontrolador PIC16F877A, conversor RS232 e DB9 conectado aos pinos Tx (transmissor) e Rx (receptor) do PIC.

Figura 3.4 - Esquema do circuito eletrônico no Proteus® (versão de demonstração).



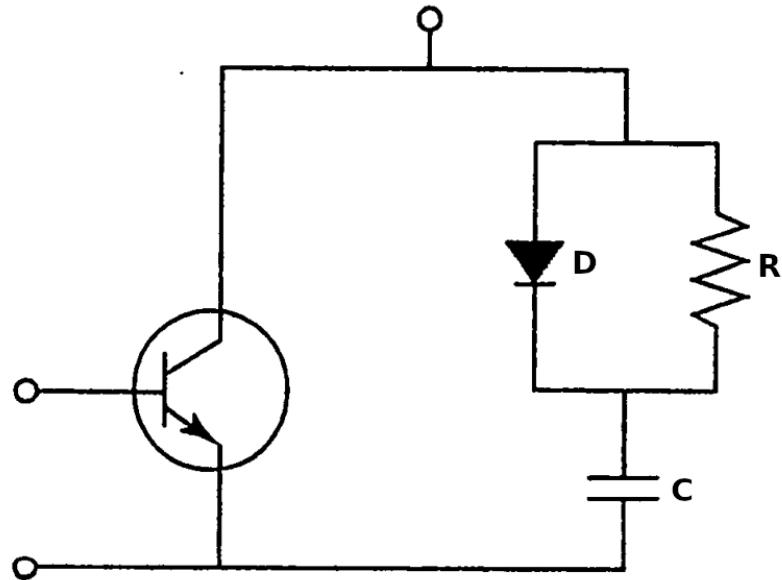
Fonte: Própria (2018).

Na montagem do circuito ponte H da Figura 3.6, primeiramente foi utilizado acoplador óptico, para isolar a etapa de controle da etapa de potência, em seguida foi desenvolvido circuito de proteção chamado *snubber*, responsável pelo amortecimento e prevenção de surtos abruptos de tensão da rede elétrica, provenientes de chaveamentos do TRIAC (BTA24) ou inversão de sentido de giro instantâneo do motor. A conexão dos pontos terminais dos motores universal estão identificados na Figura 3.20, ao acender ou desligar uma das lâmpadas indica inversão dos pólos da armadura do motor.

O circuito *snubber* é usado para limitar a tensão no dispositivo durante os transitórios de chaveamento. Circuito *snubber* típico para um BJT (transistor de junção bipolar) é apresentado na Figura 3.5. Ele é composto por um diodo e um resistor, um resistor e um capacitor (AHMED, 2000).

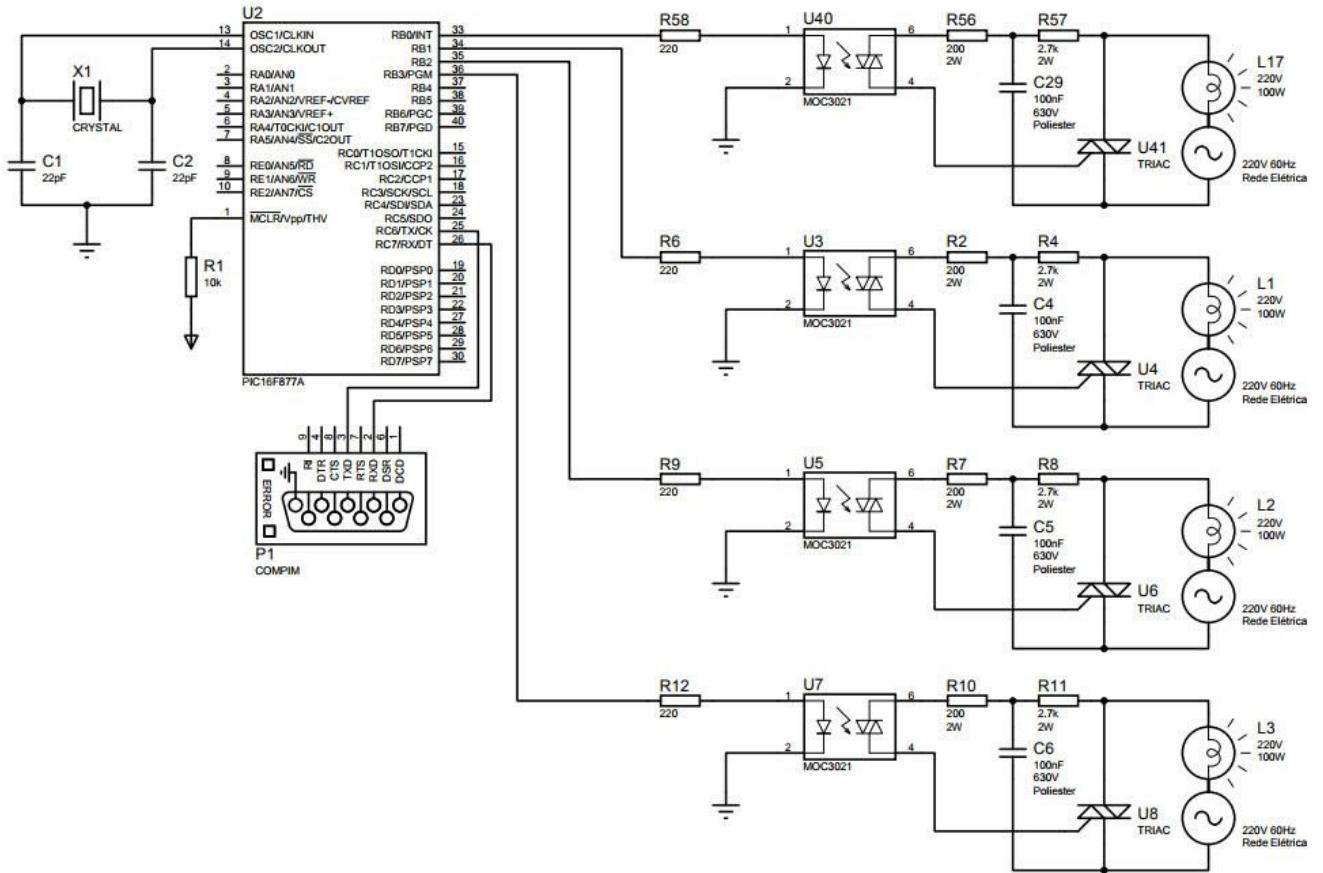
Em seguida, foi inserido no circuito o TRIAC, que é a chave bidirecional com uma porta de disparo. O mesmo permite tanto passagem do semicírculo positivo como negativo da rede elétrica. Para representar a passagem de corrente no circuito do Proteus® (versão de demonstração) foi utilizado lâmpada de 220V. O circuito contém 4 TRIACs caracterizando ponte H que foram responsável por inverter os pólos da armadura do motor universal.

Figura 3.5 - Circuito *snubber*.



Fonte: Ahmed (2000).

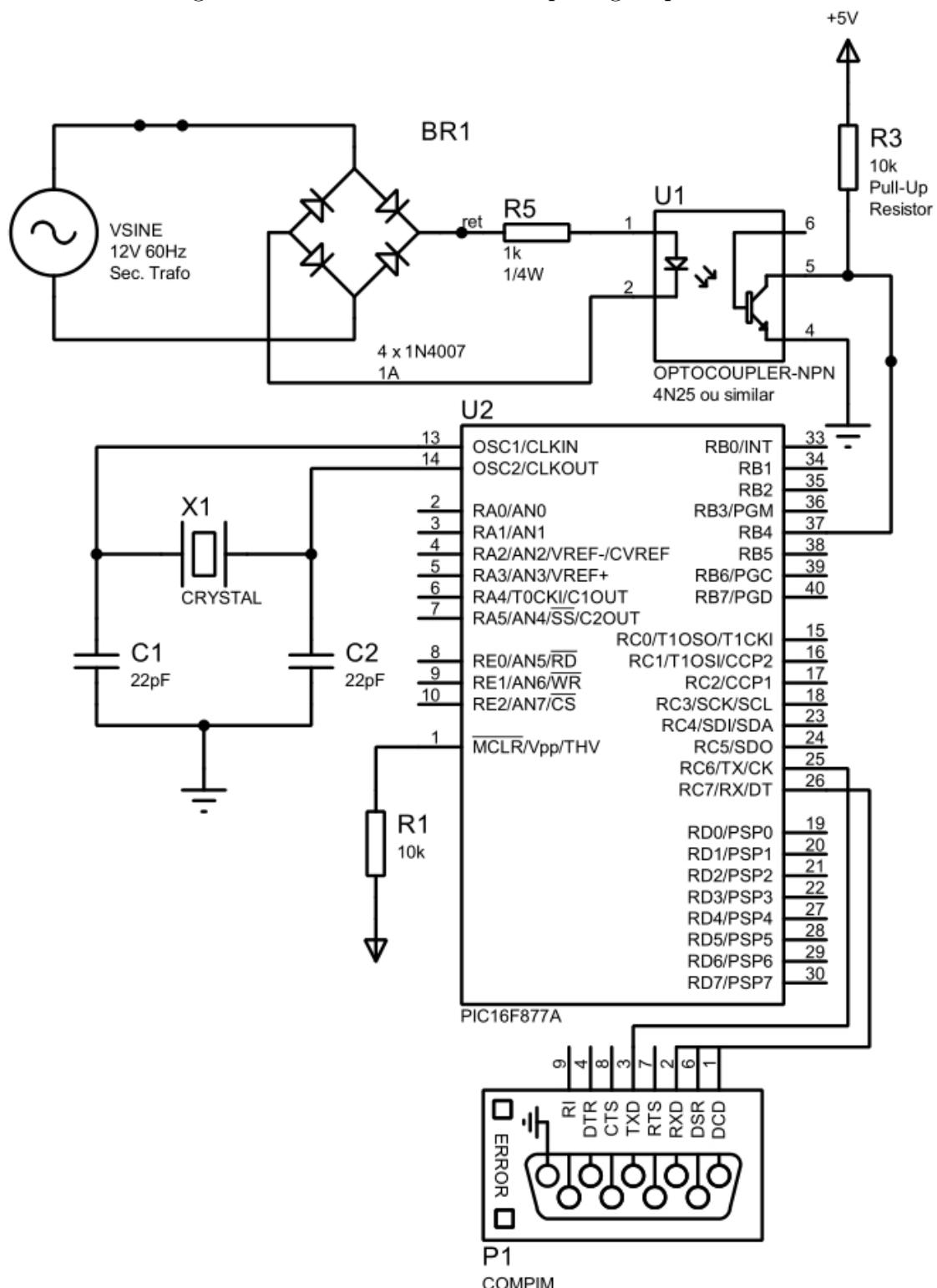
Figura 3.6 - Circuito ponte H.



Fonte: Própria (2018).

No esquema do circuito da Figura 3.7, detector de passagem por zero, primeiramente é utilizado transformador de 220 VCA para 12 VCA, em seguida, é feito a retificação de onda completa com 4 diodos, convertendo os semicírculos negativos em positivos da senoide. O sinal da rede elétrica é isolado através de acoplador óptico e um resistor de *pull – up* para setar cinco volts a cada vez que a tensão da rede elétrica passar por zero.

Figura 3.7 - Circuito detector de passagem por zero.



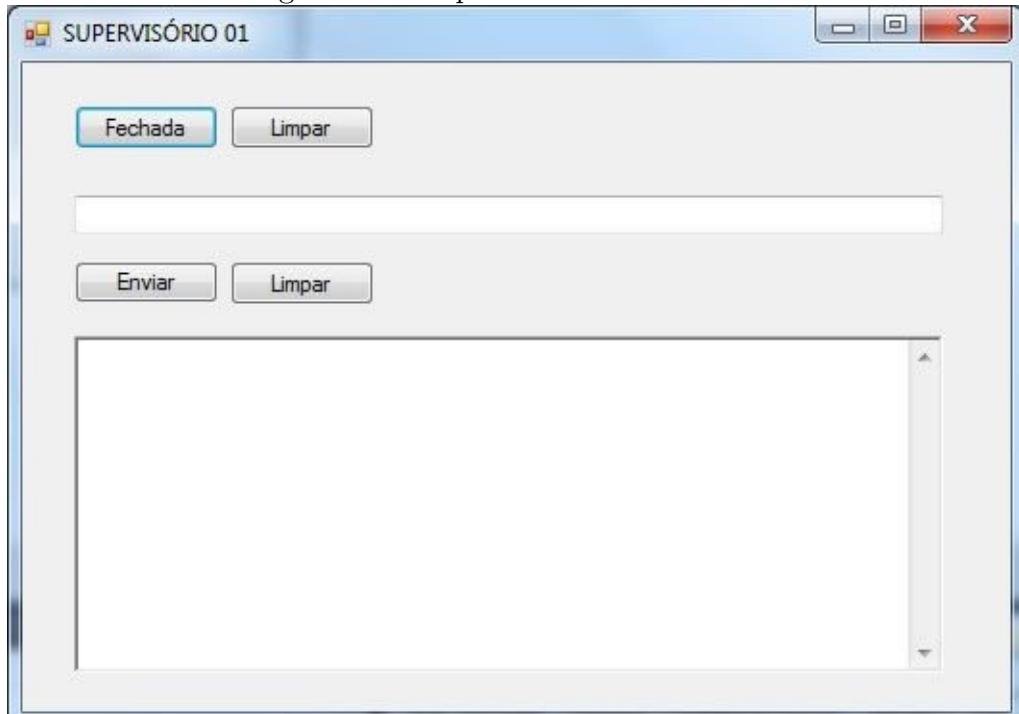
Fonte: Própria (2018).

3.3 DESENVOLVIMENTO DO SUPERVISÓRIO

A escolha em desenvolver sistema utilizando a plataforma Microsoft .NET, é baseado principalmente em sua capacidade de integração entre componentes desenvolvidos em linguagens diferentes. O supervisório foi desenvolvido como interface para o usuário e facilitar a entrada de dados e leitura do *log* de comandos.

Utilizando o Visual Studio C# foi desenvolvido o programa de comunicação. Contendo caixa de texto com botões para envio de dados, e outra caixa para recebimento de *logs* do microcontrolador, demonstrado na Figura 3.8.

Figura 3.8 - Supervisório desenvolvido.



Fonte: Própria (2018).

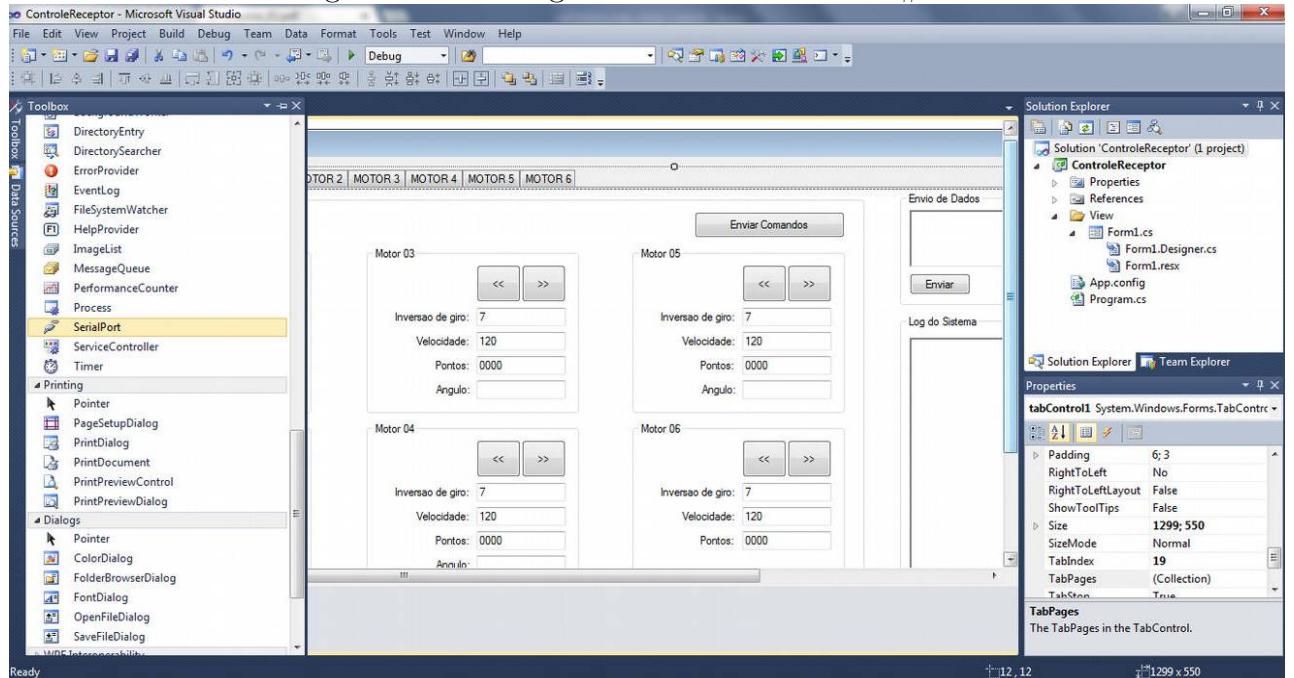
As principais funções utilizadas no Visual Studio C# estão representadas na Figura 3.9. Na parte lateral esquerda tem-se a **Tool Box** com os recursos como botões, **Group Box**, caixas de texto, gráficos e bibliotecas como **Serial Port (Porta Serial)** e **Timer (Temporizador)** entre outros.

No canto superior direito tem-se o **Solution Explorer** (Fig. 3.9), o qual possibilita acessar pastas, códigos e nomes dos arquivos presentes na árvore do projeto. Na parte inferior direita, tem-se **Propriedades** com todas as características dos componentes ao selecioná-los. Ao lado de **Propriedades** o campo que relaciona todos os objetos a algum tipo de

evento. Os eventos tornam o *software* flexível e inteligente para soluções de problemas.

A Figura 3.10 mostra como é feito a configuração da porta serial com a biblioteca **Serial Port**. O código da Figura 3.11 representa o modo de validação da porta serial.

Figura 3.9 - Vista geral do Visual Studio C#.



Fonte: Própria (2018).

Figura 3.10 - Configuração da porta serial.

```
porta.PortName = System.Configuration.ConfigurationManager.AppSettings["  
    ↪ PortName"];  
porta.BaudRate = int.Parse(System.Configuration.ConfigurationManager.  
    ↪ AppSettings["BaudRate"]);  
porta.Parity = Parity.None;  
porta.DataBits = 8;  
porta.StopBits = StopBits.One;
```

Fonte: Própria (2018).

O quadro de envio de dados, conforme Figura 3.12, é a junção dos dados presentes na tela do supervisório convertidos em *string*. Cada item é indexado sem valores nulos, caso haja, será completado com zeros. Isso garante que o microcontrolador realize varredura por índices coletando cada valor indexado corretamente. A presença do ">" e "*" representam o início e fim da comunicação. Estes caracteres filtram ruídos gerados na comunicação serial ou do circuito elétrico, e impedem erros leitura.

As informações de controle dos motores são passadas para o supervisório no seguinte formato: no início o caractere "@" e no final o caractere "&". Esses caracteres servem para identificar o tipo de informação enviados do microcontrolador para o *software*. Esse filtro impede passagem de ruídos pela porta serial, possibilita a criação de gráficos em tempo real com erro desprezível.

Figura 3.11 - Validação da porta serial.

```
if (!porta.IsOpen)
{
    try
    {
        porta.Open();
    }
    catch (Exception)
    {
        rtb_log_recebimento.AppendText("Porta Nao localizada");
    }
}
else
{
    porta.Close();
}
```

Fonte: Própria (2018).

Figura 3.12 - Quadro de envio de dados.

```
//##### QUADRO DE ENVIO
→ #####
PROTOCOLO = ">" + tb_pic_01_m01_inv.Text + pic_01_m01_vel_str + pic_01_m01_pos_str +
tb_pic_01_m02_inv.Text + pic_01_m02_vel_str + pic_01_m02_pos_str +
tb_pic_02_m01_inv.Text + pic_02_m01_vel_str + pic_02_m01_pos_str +
tb_pic_02_m02_inv.Text + pic_02_m02_vel_str + pic_02_m02_pos_str +
tb_pic_03_m01_inv.Text + pic_03_m01_vel_str + pic_03_m01_pos_str +
tb_pic_03_m02_inv.Text + pic_03_m02_vel_str + pic_03_m02_pos_str +
→ "*";
rtb_envia_dados.Text = PROTOCOLO;
porta.WriteLine(PROTOCOLO);
//##### FIM PROTOCOLO DE ENVIO
→ #####
```

Fonte: Própria (2018).

3.3.1 OBTENÇÃO DE ÂNGULOS NAS ARTICULAÇÕES

Os cálculos de obtenção de ângulos nas articulações dos 5 graus de liberdade estão relacionados com a quantidade de sinais enviados do sensor óptico para o microcontrolador, e o perímetro da roda dentada de cada eixo principal de rotação. A Figura 3.14 ilustra o disco *encoder* com 6 hastes e 6 espaços vazios interrompendo ou possibilitando a passagem do feixe de luz que sai do emissor para o receptor do sensor.

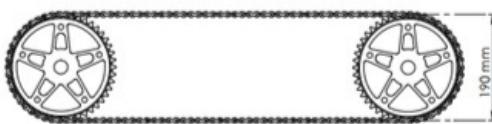
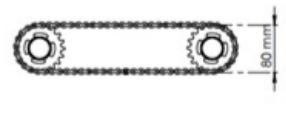
Esses sinais tanto de obstrução como de passagem do feixe são denominados **pontos**. Quando o nível lógico muda no pino do microcontrolador o algorítimo interno incrementa uma variável, quando o motor gira, no sentido horário ou decrementa, no sentido anti-horário. Assim cada vez que o disco *encoder* completa uma volta, o dispositivo fixado na barra rosada e na corrente (Fig. 4.4 (c) e Fig. 4.7 (d)) se move longitudinalmente produzindo movimento rotacional na roda dentada a cada passo de **1,5 mm** da rosca.

Portanto 1,5 mm/12 pontos equivale ao movimento longitudinal feito apenas por um ponto que corresponde a 0,125 mm. Então 0,125 mm corresponde a um sinal ou um ponto, a divisão do perímetro da roda dentada pela representação em milímetros de um sinal é obtido a quantidade de pontos ou sinais em determinado perímetro: $(Perímetro/.125) = Quantidade\ de\ Pontos$.

O perímetro de cada roda dentada é linearmente proporcional a 360 graus. A Figura 3.13 mostra a função desenvolvida e exemplos baseados nas relações lineares entre diâmetro,

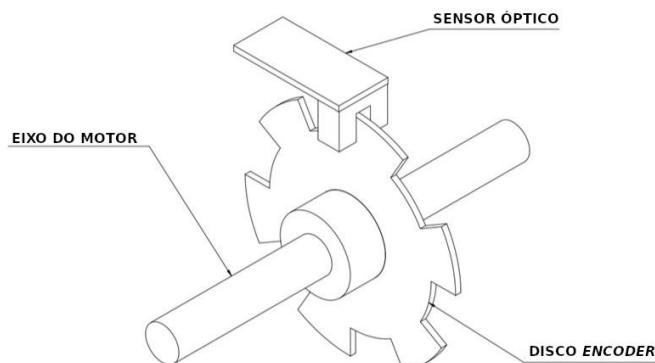
passo, quantidade de hastes do encoder e o ângulo a ser convertido em pontos.

Figura 3.13 - Relação matemática entre pontos e ângulos.

PASSO DA BARRA ROSCADA: P_B NÚMERO DE HASTES NO ENCODER: N_H DIÂMETRO RODA DENTADA: D_R ÂNGULO: A QUANTIDADE DE HASTES POR PERÍMETRO DA RODA DENTADA: Q_H_P	FÓRMULA DE CONVERSÃO DE ÂNGULOS PARA QUANTIDADE DE PONTOS DETECTADOS PELO SENSOR ÓPTICO. $Y = [(PI * D_R * N_H * A) / (P_B * 360)]$
	$Y = [(3.1416 * 190 * 12 * 30^\circ) / (1.5 * 360^\circ)]$ PARA 30° TEMOS Y = 397.93 PONTOS SENSOR ÓPTICO RECEBE 398 SINAIS
	$Y = [(3.1416 * 130 * 12 * 30^\circ) / (1.5 * 360^\circ)]$ PARA 30° TEMOS Y = 272.27 PONTOS SENSOR ÓPTICO RECEBE 272 SINAIS
	$Y = [(3.1416 * 80 * 12 * 30^\circ) / (1.5 * 360^\circ)]$ PARA 30° TEMOS Y = 167.55 PONTOS SENSOR ÓPTICO RECEBE 168 SINAIS

Fonte: Própria (2018).

Figura 3.14 - Disco encoder.



Fonte: Própria (2018).

O motor universal foi escolhido com base em suas características operacionais e por ser de fácil acesso (encontrado no comércio). Por estar presente na maioria dos aparelhos eletrodomésticos, possibilitando então seu aproveitamento neste trabalho diminuindo custo de aquisição de motores de outros tipos.

A velocidade do motor universal a vazio é excessivamente alta, e possui alto torque de partida. Estas características foram levadas em consideração para que o manipulador robótico tenha maior agilidade de movimento, em razão da robustez desejada para o mesmo.

3.4 DESENVOLVIMENTO DE SISTEMA DE ORIENTAÇÃO E POSIÇÃO DO EFETUADOR

O estudo da cinemática do manipulador significa analisar trajetórias, eliminar erros de movimentação de forma que o posicionamento final do efetuador, definido ao enviar as coordenadas cartesianas XYZ, tenha erro desprezível comparado com o valor encontrado pelo algoritmo desenvolvido.

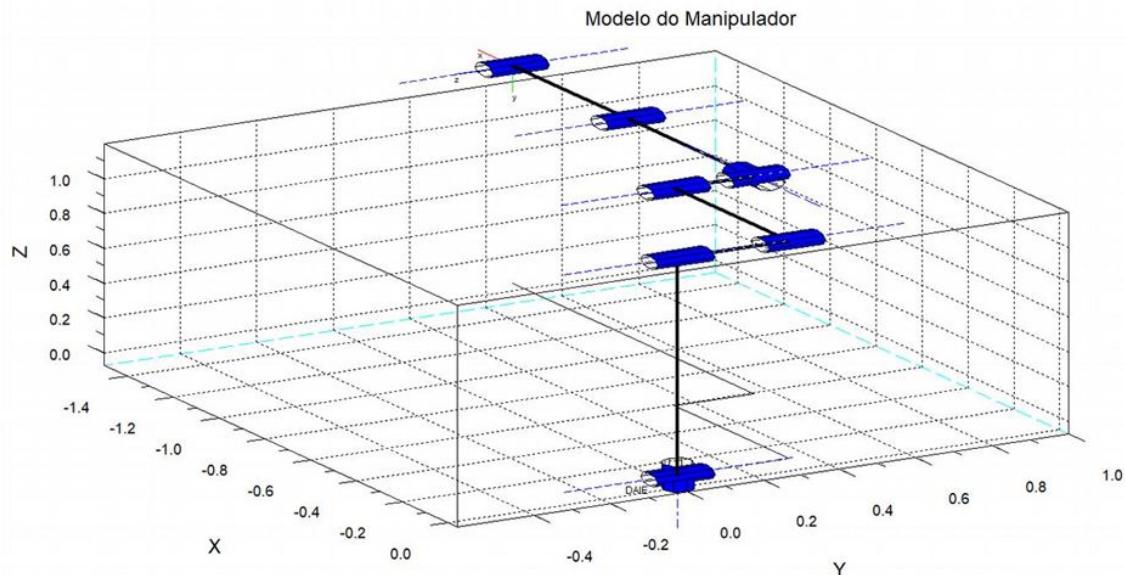
O *software* utilizado para modelar o manipulador robótico, nomeado DAIE (nome atribuído ao manipulador robótico no ambiente de simulação), foi o *Scilab* junto com o *RTSS - the Robotics Toolbox for Scilab/Scicos* ([CORKE, 2018a](#)). *Scilab* ([SCILAB.ORG, 2018](#)) é *software* livre, para computação numérica em aplicações de engenharia. RTSS disponibiliza funções da cinemática direta e inversa de manipuladores robóticos, modelos de manipuladores, todas funções disponíveis são documentadas com exemplos de aplicação. O RTSS é inspirado no *Robotic Toolbox* do Matlab®. A escolha de desenvolver este trabalho com os *softwares* citados se dá por dois motivos: serem robustos e gratuitos.

Para realizar simulações de movimentação e posicionamento, é necessário ter o modelo que represente os ligamentos do manipulador robótico, levando em consideração o tamanho, quantidade de ligamentos, tipo de juntas, prismáticas ou de revolução. O modelo desenvolvido é representado na Figura 3.15. A Figura 3.16 é a vista lateral e superior do manipulador, contendo informação dos comprimentos dos ligamentos e distâncias entre juntas. A versão estável do RTSS é **0.3.0** tem seu melhor funcionamento com *Scilab-4.1.2*. Para iniciar a utilizar as funções e modelos do RTSS, o primeiro passo é carregar para o *Scilab* os arquivos *builder.sce* e *loader.sce* contidos no arquivo *.zip* de instalação do RTSS.

O aspecto principal para entender o desenvolvimento do modelo é compreender o comando de criação de ligamentos, e os parâmetros deste comando, os quais incluem as medidas da Figura 3.16. O comando *rt_link* é utilizado para criar os ligamentos, o objeto armazena parâmetros, dinâmicos, cinemáticos, assim como de atuador e transmissão. O tipo do objeto criado é **link**. A Tabela 3.1 segue a descrição de parâmetros e comandos utilizados

para criar o modelo.

Figura 3.15 - Posição inicial do modelo.



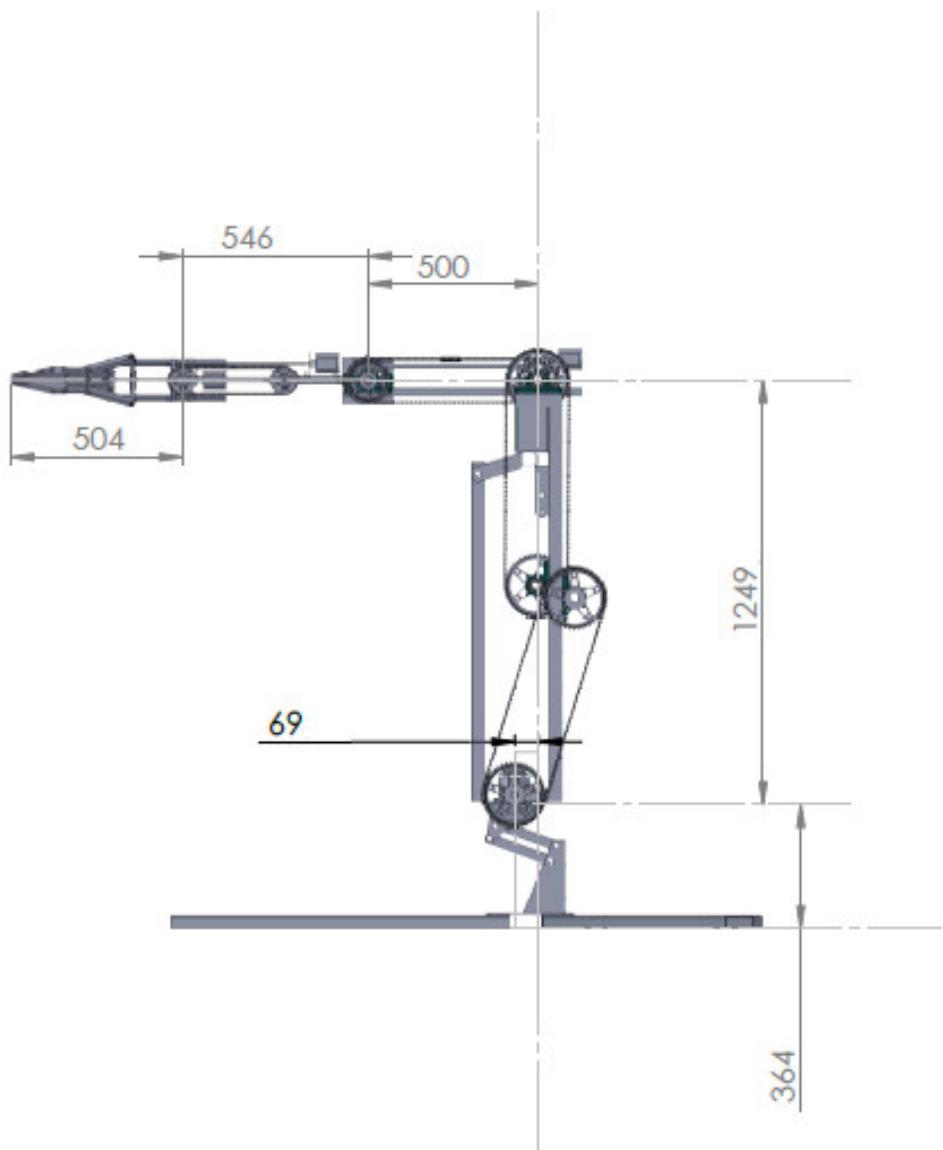
Fonte: Própria (2018).

Tabela 3.1 - Legenda criação do objeto ligamento para o manipulador desenvolvido.

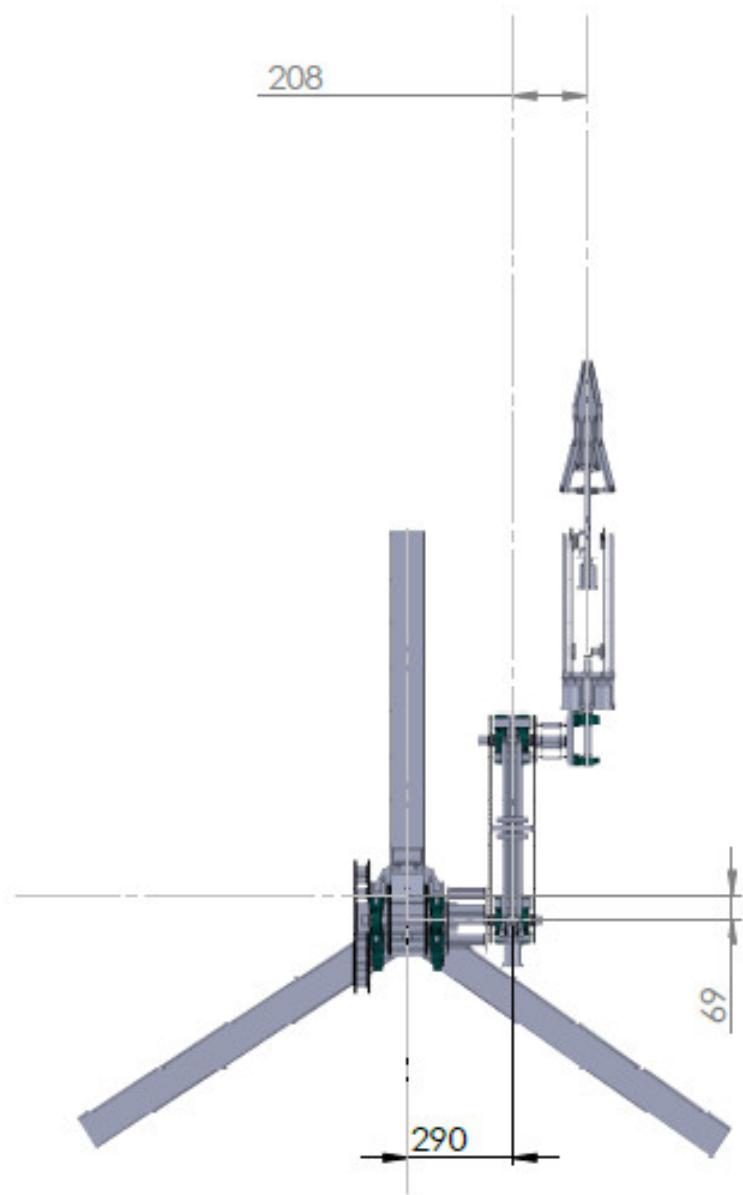
Parâmetro	Descrição
$l = rt_link()$	Cria objeto ligamento.
$l = rt_link(dh_row[, convention])$	Ligamento recebe os parâmetros Denavit Hartenberg como vetor de 5 posições.
$dh_row = [\alpha, a, \theta, d [, \sigma]]$	Vetor que armazena os Parâmetros Denavit Hartenberg.
α	Escalar. Parâmetro Denavit Hartenberg representa ângulo de torção do ligamento.
a	Escalar. Parâmetro Denavit Hartenberg representa comprimento do ligamento.
θ	Escalar. Parâmetro Denavit Hartenberg representa ângulo de rotação do ligamento.
d	Escalar. Parâmetro Denavit Hartenberg representa distância offset do ligamento.
σ	Escalar. Tipo da junta: 0 junta de revolução, valor não-nulo junta prismática.
$convention$	<i>String.</i> Parâmetros Denavit Hartenberg “standard” ou “modified”.

Fonte: [Modificado de rtss.sourceforge.net\(2018b\)](http://Modificado de rtss.sourceforge.net(2018b))

Figura 3.16 - Manipulador robótico acadêmico, vista:



(a) Lateral.



(b) Superior.
Fonte: Própria (2018).

3.5 CONSTRUÇÃO DO MANIPULADOR ROBÓTICO

Depois de planejado o manipulador robótico, foram definidas as peças e partes físicas essenciais para sua construção, e então a pesquisa por preços e condições de compra. O projeto foi financiado pelos integrantes do trabalho, as máquinas e ferramentas disponíveis no laboratório de mecânica do IFG - Câmpus Goiânia foram utilizadas na construção do manipulador robótico, e fabricação de certas peças não encontradas comercialmente tão facilmente.

A Figura 3.17 apresenta os principais equipamentos e máquinas, presentes no Laboratório de Mecânica que foram utilizados para construir o manipulador robótico. O processo de usinagem das peças foi realizado com o torno mecânico, serra, moto esmeril (esmerilhadeira) e furadeira. A bancada, Figura 3.17 (e) com a morsa e arco de serra foram usados para montagem, acoplamento de partes e ajustes feitos em partes específicas do manipulador robótico. Figura 3.17 (f) paquímetro, instrumento de medição em polegadas e milímetro.

Figura 3.17 - Máquinas e equipamentos utilizados na construção do manipulador robótico acadêmico.



(a) Torno mecânico.



(b) Serra.



(c) Moto esmeril (esmerilhadeira).



(d) Furadeira.



(e) Bancada.



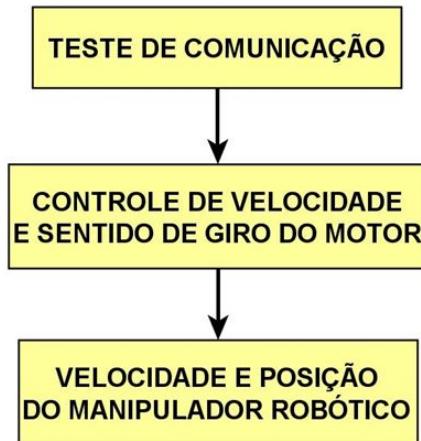
(f) Paquímetro.

3.6 TESTES DE FUNCIONAMENTO

Os testes de funcionamento/validação foram realizados diversas vezes ao decorrer do desenvolvimento do projeto. Foram realizados do início ao fim, presente não somente durante a construção do manipulador mas também nas etapas de simulação, e principalmente nos momentos que se faz necessário integrar o *software* com o manipulador construído.

A Figura 3.18 ilustra o fluxograma da lógica que foi seguida nos testes realizados. No primeiro momento foi notado a necessidade da comunicação feita entre o computador, os microcontroladores, utilizando protocolo RS232. Com o manipulador concluído, o estudo da cinemática direta foi utilizado para realizar os testes de velocidade e posicionamento do efetuador.

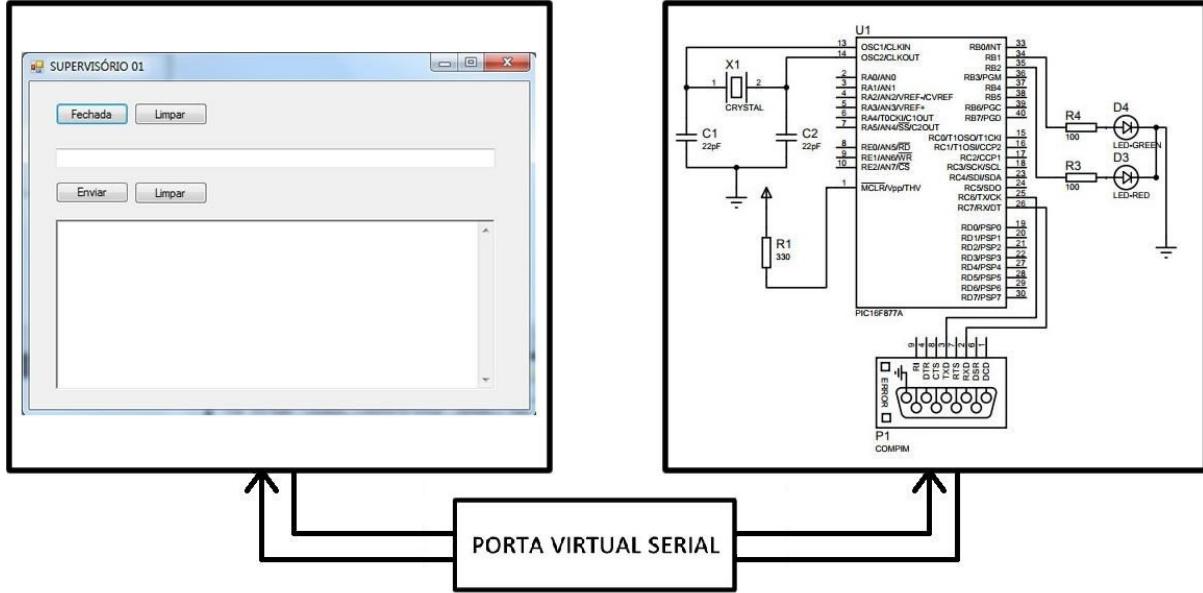
Figura 3.18 - Fluxograma das etapas de testes de funcionamento.



Fonte: Própria (2018).

Por meio de porta virtual serial de comunicação foi possível elaborar testes de envio de *strings* para o Proteus® (versão de demonstração), representado na Figura 3.19, com obtenção de resposta instantânea. Para analisar a capacidade de recebimento e envio de caracteres, testes com a maior *string* possível comprovaram a quantidade de caracteres possíveis sem que haja atraso (devido tempo de processamento) no envio ou recebimento dos dados. Logo após os testes preliminares, foram elaborados *strings* com múltiplos comandos para acender e desligar LEDs simultaneamente.

Figura 3.19 - Comunicação serial entre Proteus® (versão de demonstração) e o Supervisório.



Fonte: Própria (2018).

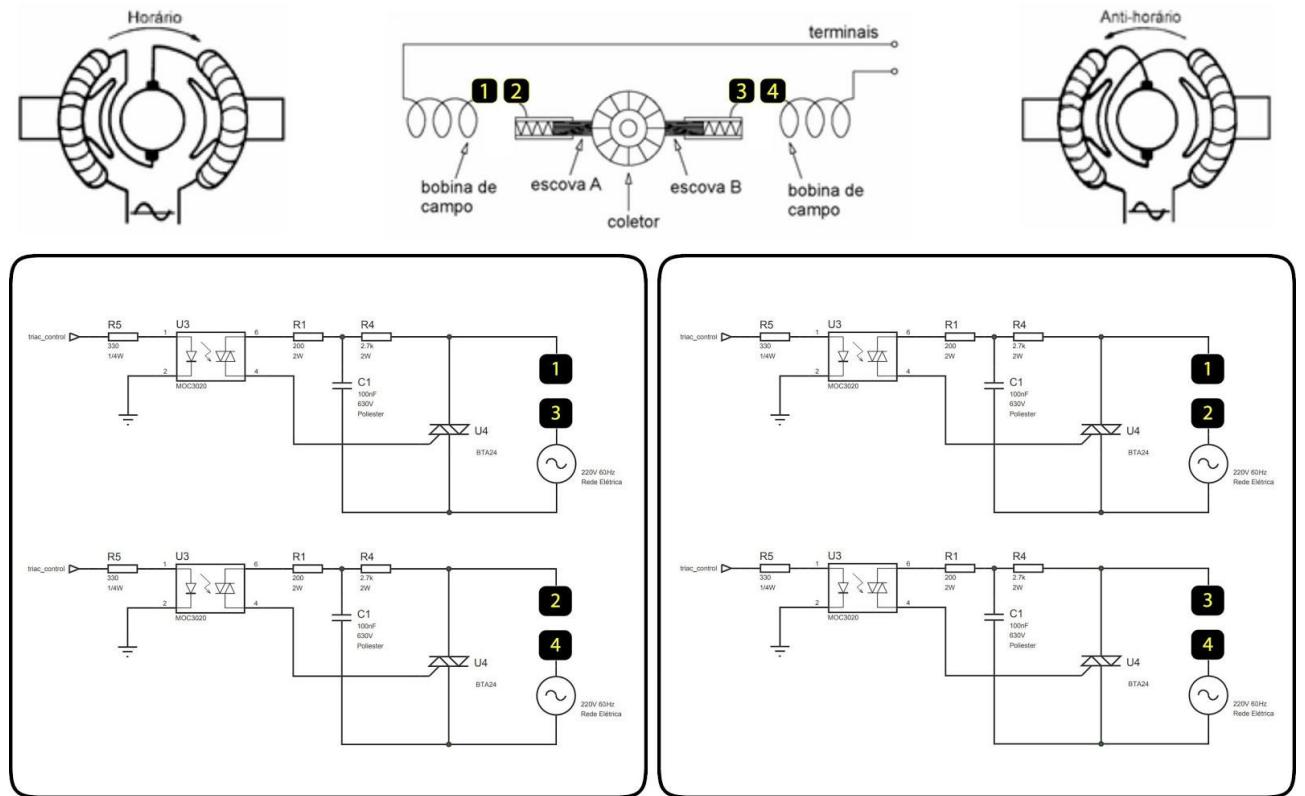
O supervisório envia uma *string* ao microcontrolador do Proteus® (versão de demonstração), contendo comandos para acionar os TRIACs, em consequência o PIC muda o estado de dois pinos para nível lógico alto e outros dois para nível lógico baixo (zero). Os sinais são recebidos pelo optoacoplador da placa de potência e transmitidos ao *gate* do TRIAC, onde libera a passagem de corrente no tempo em que o pino está em nível lógico alto. Assim, as lâmpadas são ligadas de acordo com a combinação elaborada pela *string* do supervisório, caracterizando a inversão do sentido de giro do motor universal (representa a inversão dos pólos da armadura do motor), esquema ilustrado na Figura 3.20.

O manipulador robótico industrial possui 5 graus de liberdade, contendo 5 motores universais para fazer o controle de posicionamento, logo para organizar o processamento e fazer a comunicação instantânea foram utilizados inicialmente três microcontroladores que receberam dados do computador via RS232. Foram usados três microcontroladores para distribuir o espaço de armazenamento de dados, evitar ocorrência de sobrecarga de processamento ou memória. Posteriormente, baseado nos testes realizados o código foi otimizado para apenas um microcontrolador.

Os dados são enviados por cabos e todos os microcontroladores recebem a *string* do supervisório. A *string* tem o endereçamento de cada microcontrolador e cada motor. No momento da recepção dos dados, o microcontrolador interpreta o vetor de informações e envia a seus motores. A Figura 3.21 apresenta um esquema desta comunicação, onde o supervisório (mestre) envia o comando e o microcontrolador (escravo) deve executá-lo (*listens*).

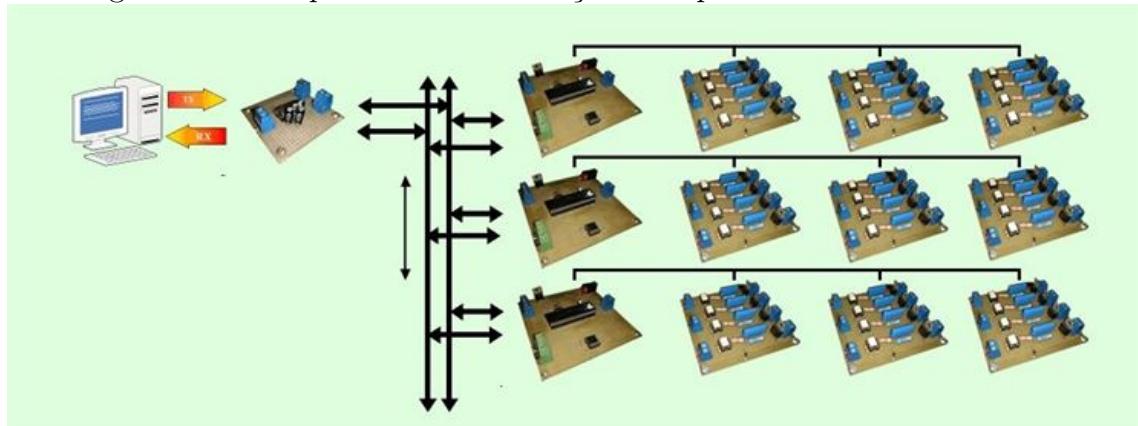
tening). Um dos microcontroladores recebe o sinal de passagem por zero e irá retransmitir para os outros dois, sincronizando todos com a rede elétrica.

Figura 3.20 - Esquema inversão de rotação do giro do motor.



Fonte: Própria (2018).

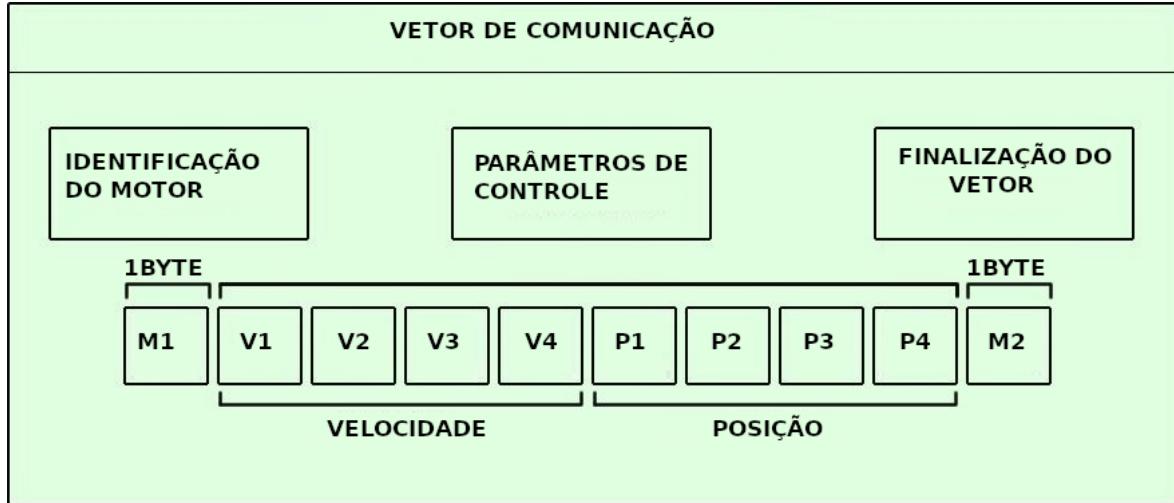
Figura 3.21 - Esquema da comunicação feita para controle dos motores.



Fonte: Modificado de Luiz Gustavo (2010).

A cada segundo os microcontroladores enviam vetor de informações para o supervisório, para que não haja colisão de pacotes, o sincronismo do envio da mensagem será feita de acordo com a ordem dos microcontroladores, ilustrado na Figura 3.22, sendo M1 a identificação do motor, V1 a V4 a velocidade e P1 a P4 a posição e M2 o fim do vetor. A sequência de envio de dados corresponde ao endereçamento 01, 02 e 03. A cada envio o microcontrolador seguinte é avisado até finalizar o envio do pacote.

Figura 3.22 - Vetor de comunicação (quadro).



Fonte: Própria (2018).

Com base em informações de ângulos passados do supervisório para o microcontrolador, o algoritmo de controle Proporcional Integral Derivativo (PID) - implementado no PIC - tem a função de aumentar a potência entregue ao motor até que o posicionamento angular seja atingido. Conforme mostrado na Figura 3.23 a planta seria o circuito de controle de ângulo de disparo da rede elétrica e o PID age diretamente em relação ao parâmetro de controle. Os parâmetros do controlador PID foram obtidos experimentalmente, a Tabela 3.2 apresenta os valores.

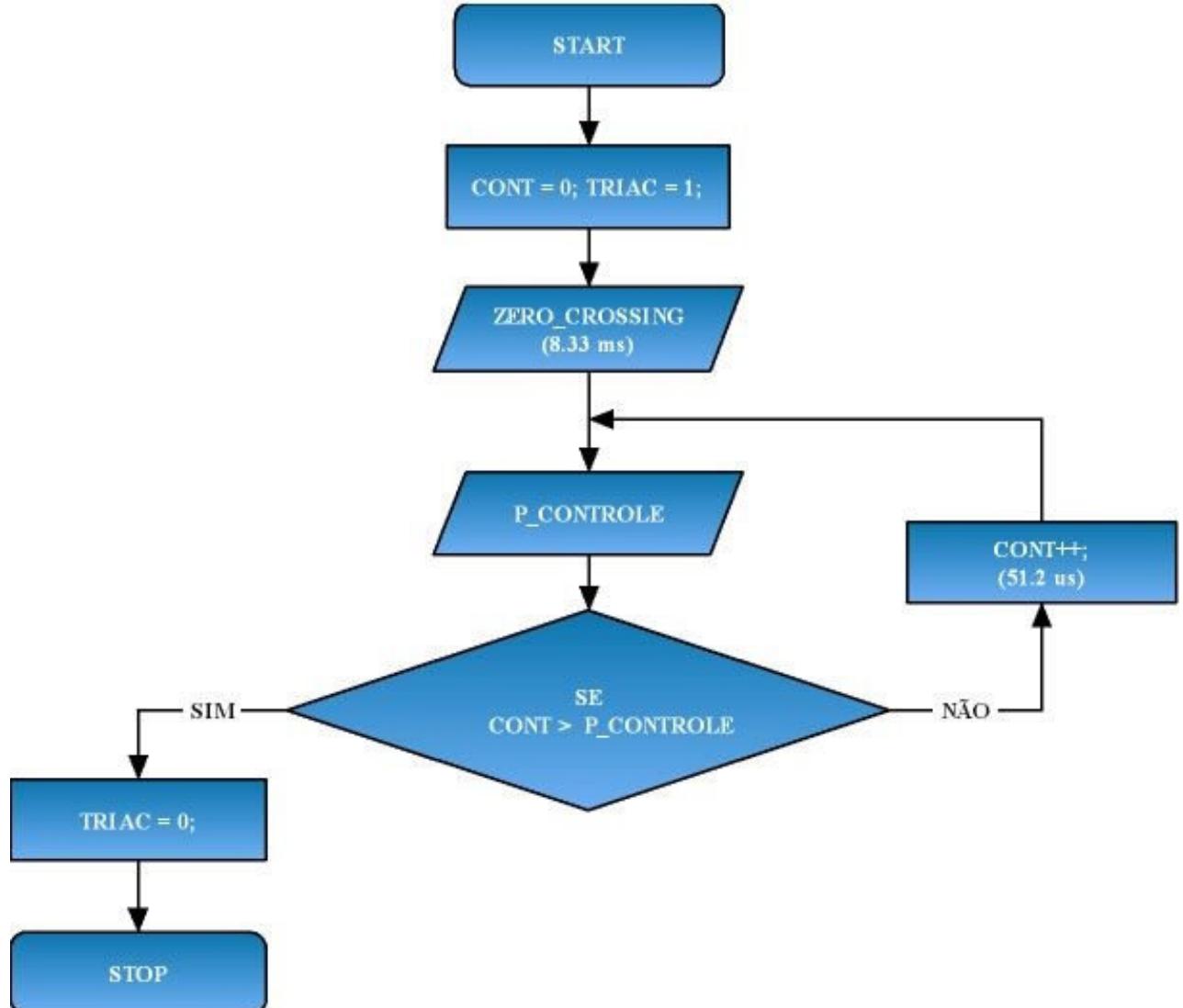
No fluxograma (Fig. 3.23) *start* inicializa o programa, em seguida é atribuído valor para variável *cont* (contador) e inicializa o TRIAC (passagem de corrente elétrica); *zero_crossing* faz a detecção de passagem por zero; *P_controle* é o parâmetro de controle enviado do supervisório ou do PID. Enquanto o contador for menor do que a variável de controle, ele é somado, se for maior, o TRIAC para de chavear.

Tabela 3.2 - Parâmetros do PID.

K_p	0,1
K_i	0,07
K_d	0,01

Fonte: Própria (2018).

Figura 3.23 - Planta responsável pelo controle do ângulo de disparo do TRIAC.



Fonte: Própria (2018).

CAPÍTULO 4

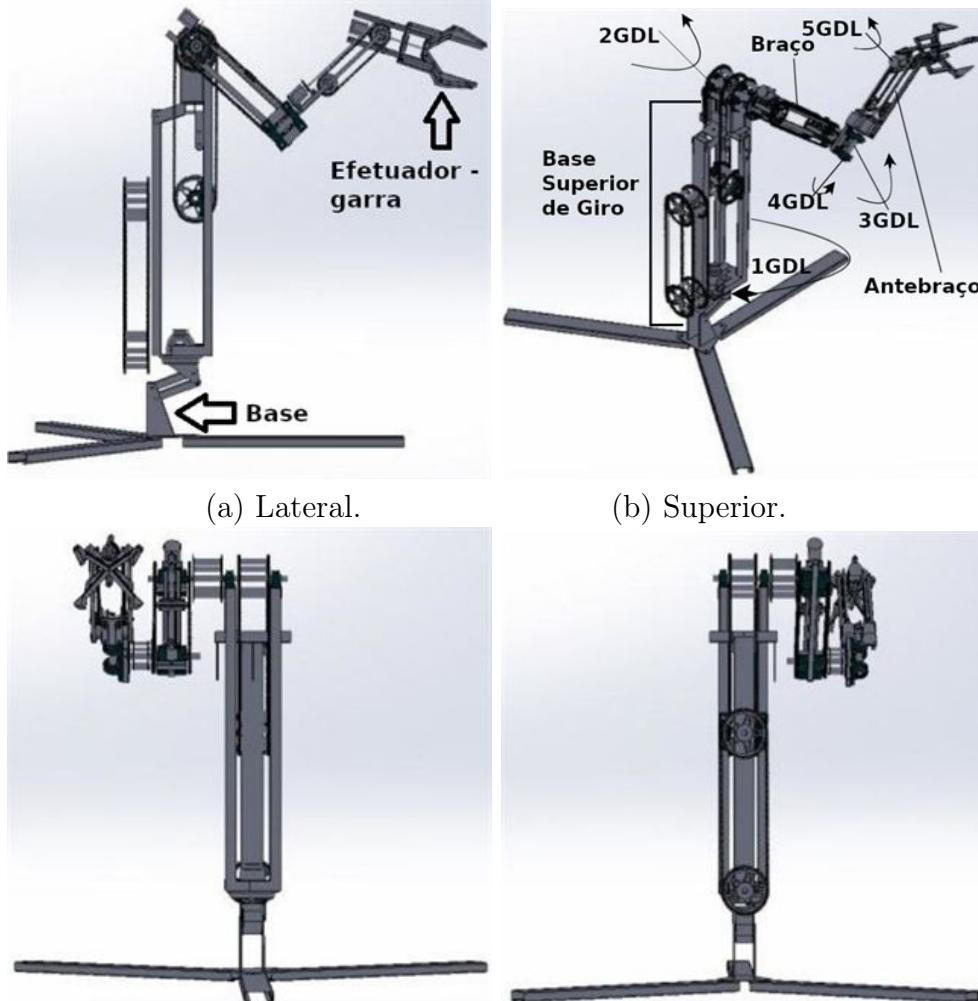
Resultados e Discussões

O presente capítulo apresenta a análise dos resultados obtidos (códigos fontes foram subidos para <https://github.com/AlexAlvTrin/TCC-ROBOTICA_IFG> e <<https://github.com/denysczr/MANIPULADOR>>).

4.1 PROJETO MECÂNICO

A Figura 4.1 demonstra o projeto mecânico do manipulador robótico. A base e o efetuador - garra, assim como cada posição das juntas responsáveis pelos cinco graus de liberdade são indicados na Figura 4.1 (a) e (b) respectivamente.

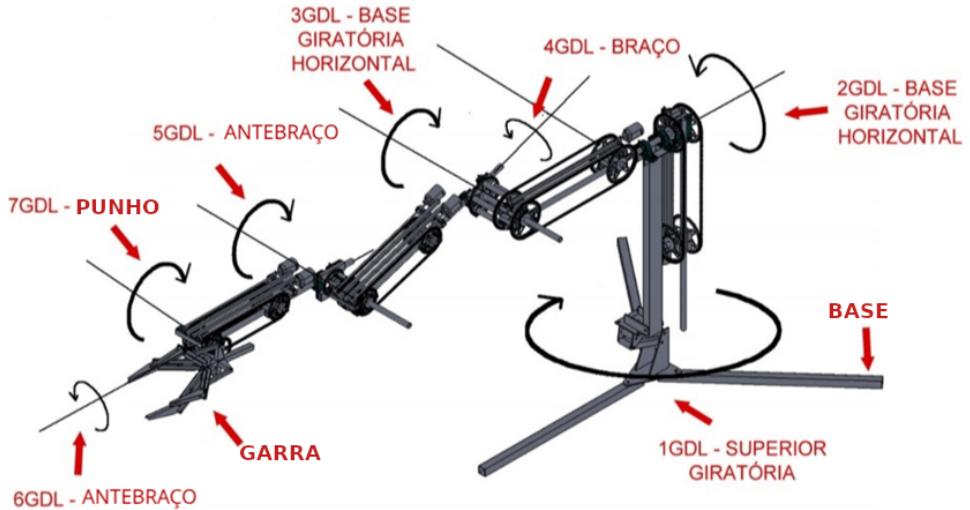
Figura 4.1 - Desenho do Projeto Mecânico, vista:



Fonte: Própria (2018).

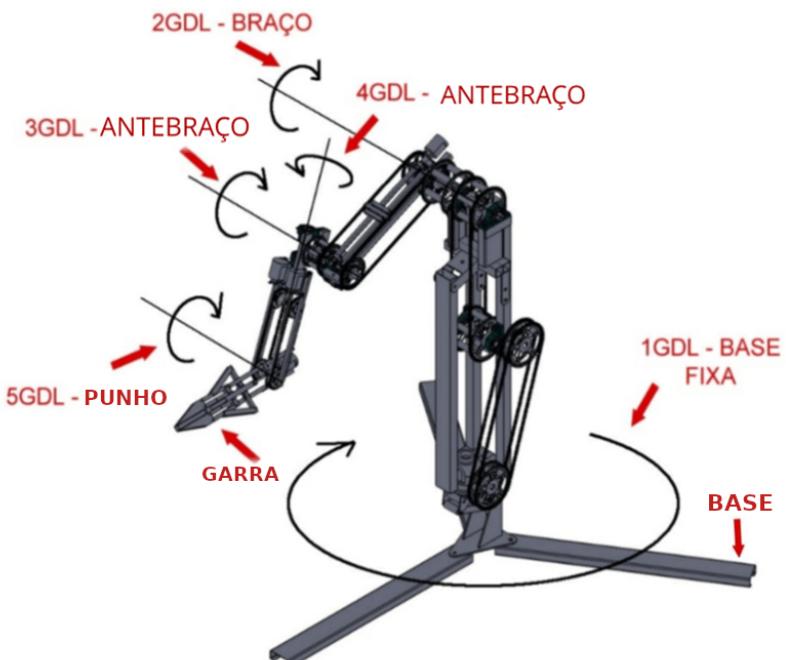
Devido as modificações aplicadas durante o desenvolvimento do manipulador robótico descrito na seção 3.1, foram definidas três versões do projeto mecânico, incluindo a versão final. As Figuras 4.2 e 4.3 mostram a evolução do projeto nas duas primeiras versões.

Figura 4.2 - Primeira versão do projeto mecânico.



Fonte: Própria (2018).

Figura 4.3 - Segunda versão do projeto mecânico.

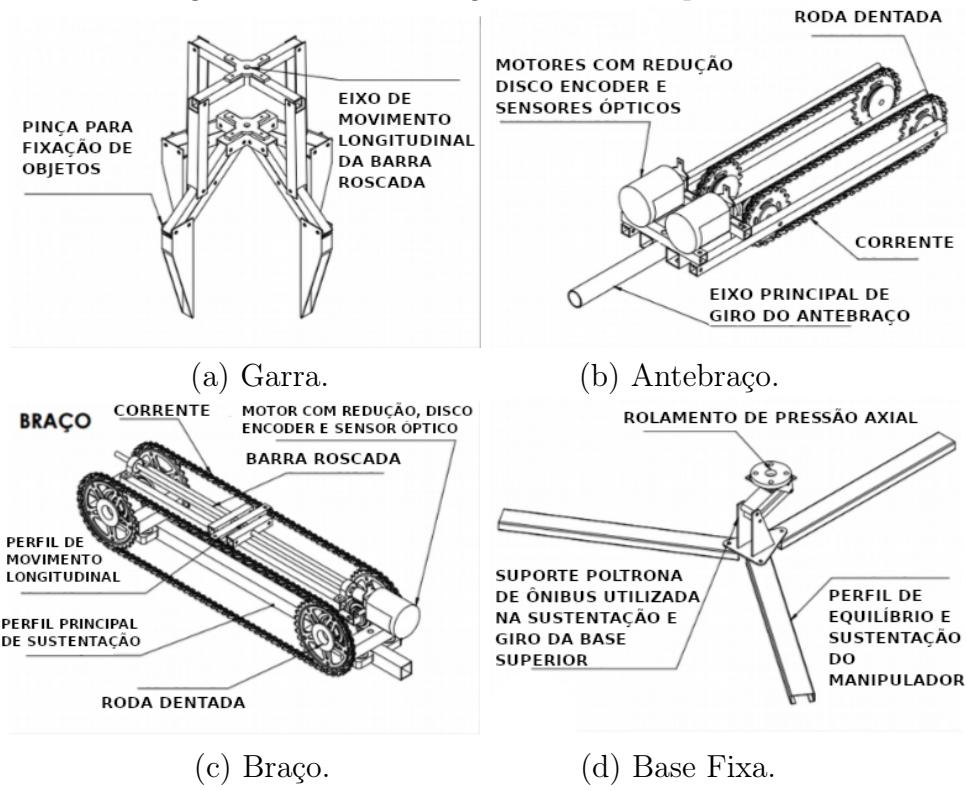


Fonte: Própria (2018).

A primeira versão do projeto mecânico possuia 7 GDL (graus de liberdade). A complexidade na realização de movimento era consideravelmente superior, pela maior quantidade de motores, peso, quantidade de peças necessitando de mais tempo de desenvolvimento.

Durante realização de pesquisas foi notado que para realizar os movimentos necessários seria suficiente 5 graus de liberdade. Com base nisso, e ponderada as dificuldades em construir a versão original do projeto mecânico, o projeto foi adaptado para 5 GDL. A Figura 4.4 apresenta detalhes de cada submontagem e descrição de seus componentes.

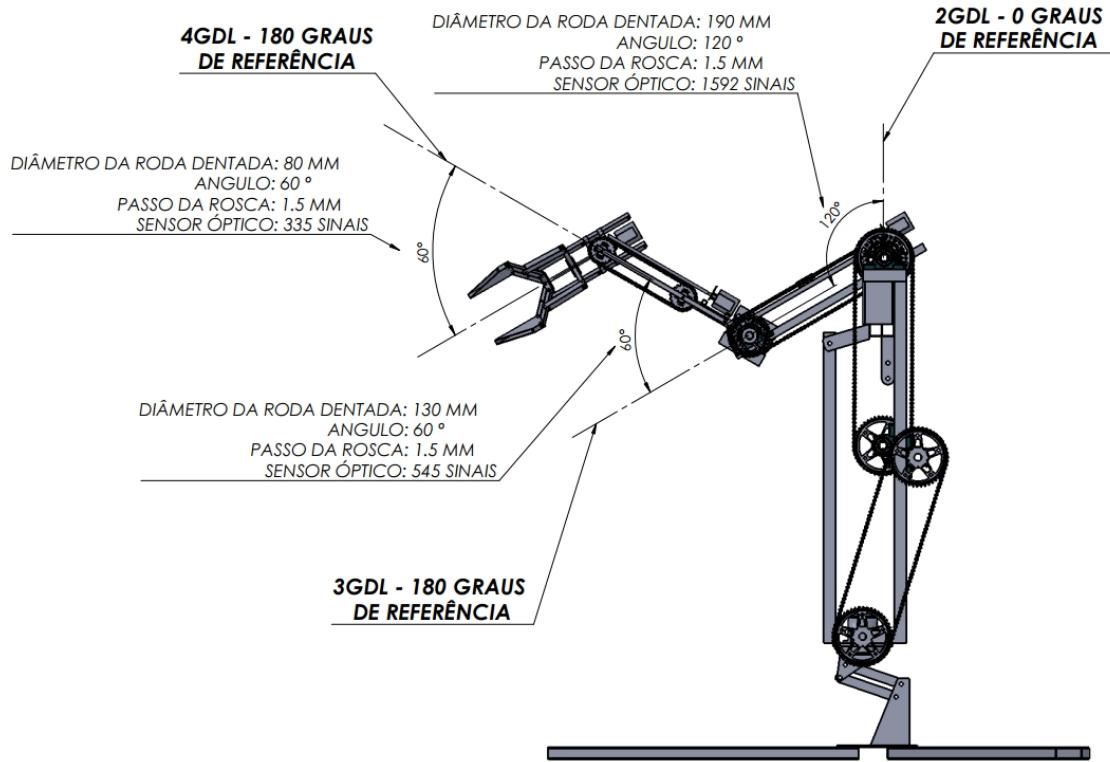
Figura 4.4 - Submontagens e seus componentes:



Fonte: Própria (2018).

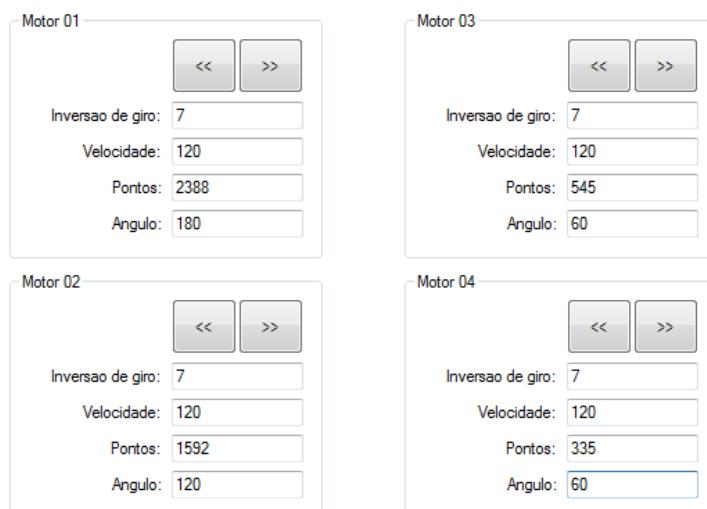
Cada grau de liberdade está relacionado a numeração do motor conforme visto na tela parcial do supervisório (Fig. 4.6) em comparação com o estudo dos ângulos dos eixos de movimentação (Fig. 4.5). Essas duas imagens mostram etapa de controle de movimento do manipulador com ângulos de 60 e 120° convertidos e enviados do supervisório. A função de conversão de ângulos para pontos é gerada na tela do supervisório, a medida em que o usuário digita valor imediatamente a conversão é mostrada no campo “Pontos” da tela (Fig. 4.39).

Figura 4.5 - Estudo dos ângulos dos eixos de movimentação.



Fonte: Própria (2018).

Figura 4.6 - Resultado apresentado no supervisório.

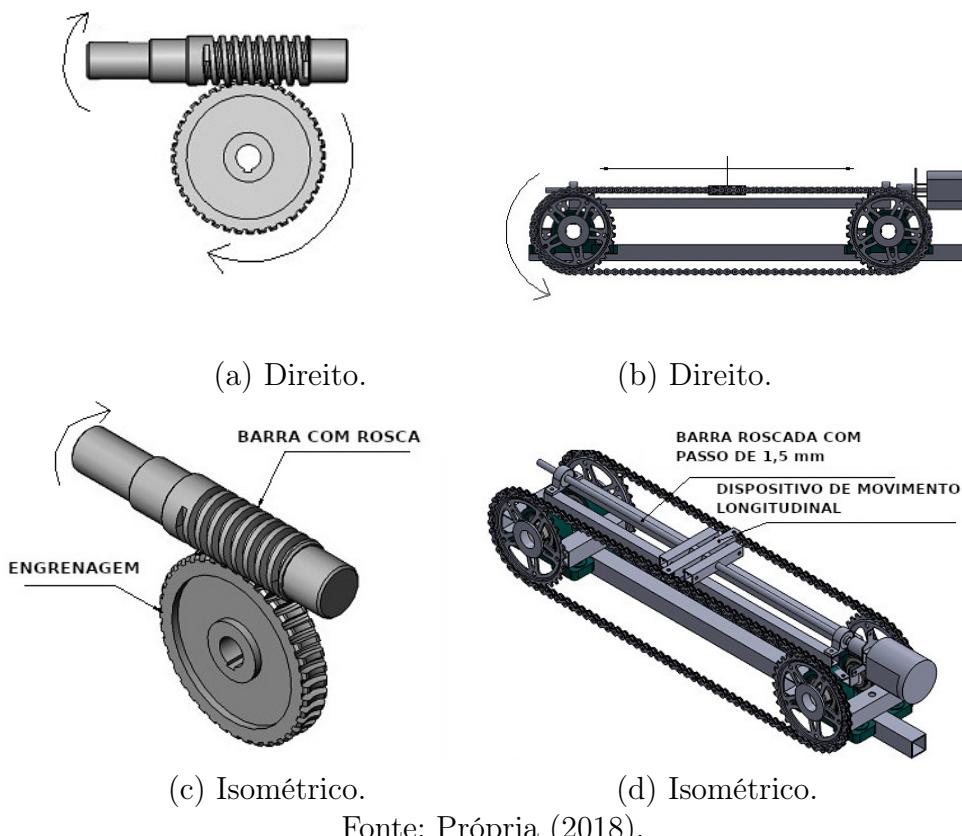


Fonte: Própria (2018).

A Figura 4.7 compara duas formas de conversão de forças longitudinal para rotacional. A forma da conversão dos movimentos e transmissão de potência nas articulações do manipulador foi baseada na redução por rosca "sem fim", Figura 4.7 (a).

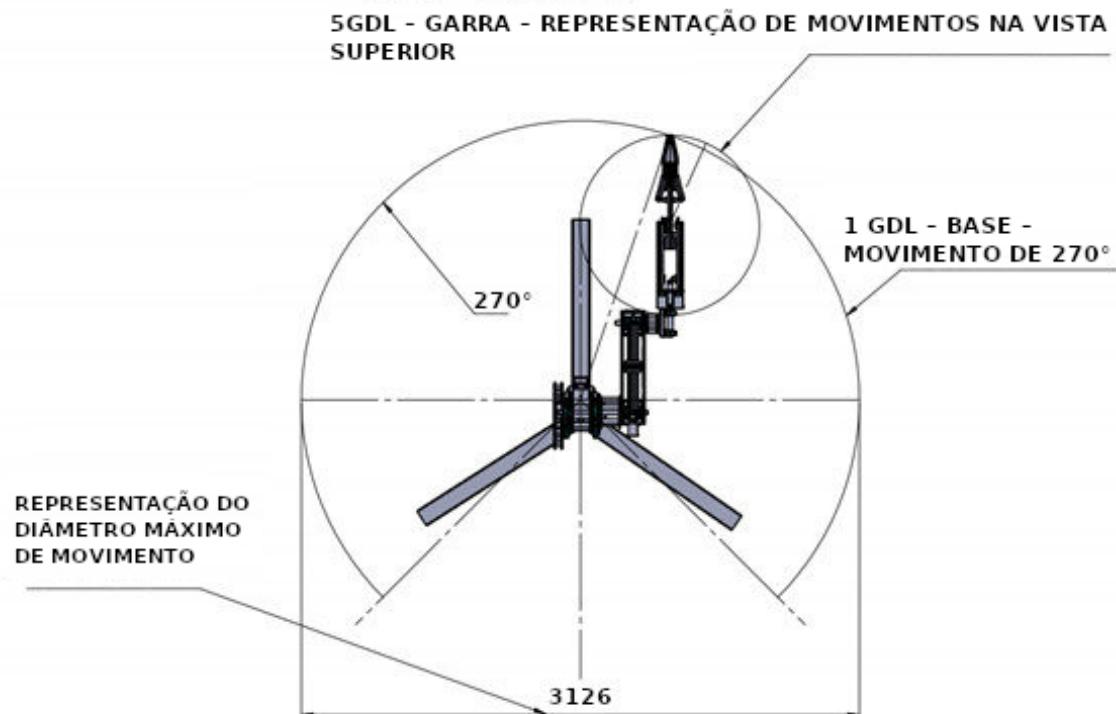
Foi desenvolvido método simplificado de custo reduzido para reproduzir os movimentos. A quantidade de dispositivos do sistema mecânico utilizados na transmissão de potência por corrente com rosca fixada, Figura 4.7 (b), causa maior erro na geração de ângulos, ocorre então cascateamento de erros na reprodução dos cálculos cinemáticos. A redução por rosca sem fim não foi utilizada (apenas baseado nesse método) no projeto por possuir alto custo, o que tornaria o projeto inviável, portanto o mesmo poderá ser utilizado em projetos futuros de maior precisão.

Figura 4.7 - Formas de conversão de forças longitudinal para rotacional, vista:



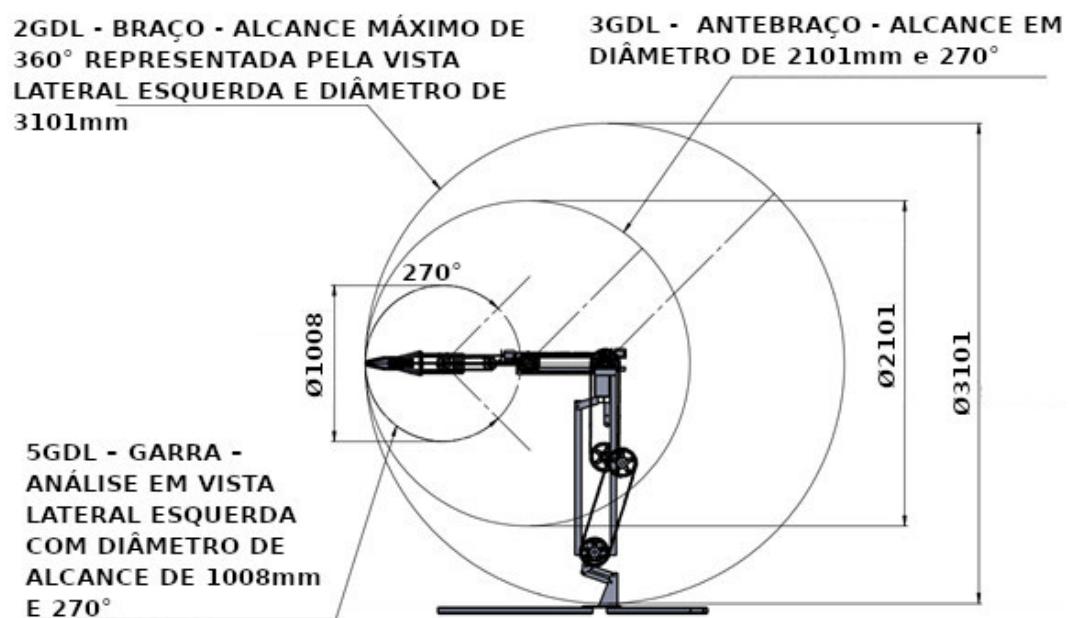
Fonte: Própria (2018).

Figura 4.8 - Diâmetro máximo de movimento, vista superior.



Fonte: Própria (2018).

Figura 4.9 - Diâmetro máximo de movimento, vista lateral.



Fonte: Própria (2018).

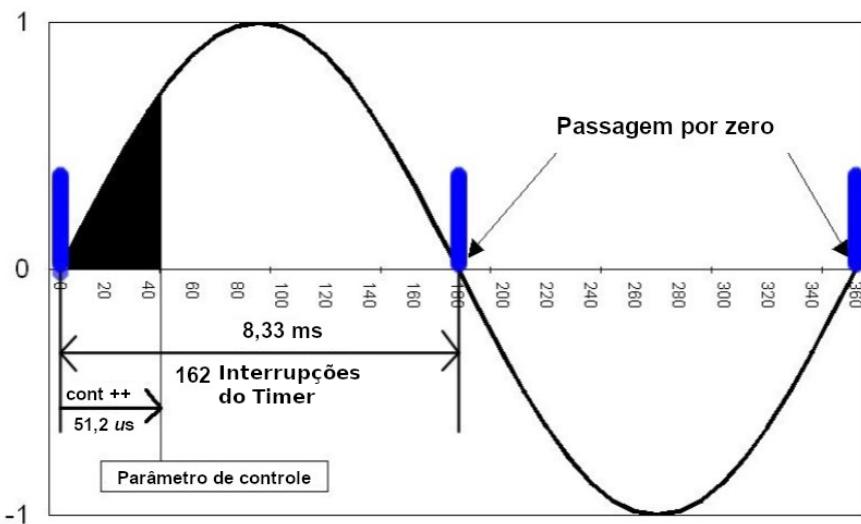
A Figura 4.8, apresenta a vista superior do projeto mecânico, o qual possui diâmetro máximo de 3126 mm, o movimento de giro (rotação) da base é limitado em até 270° de arco. Figura 4.9, vista lateral direita, as linhas circulares representam os limites dos eixos relacionados aos graus de liberdade. O limite do 5 GDL é de 1008 mm de diâmetro com máximo de arco de 270° ; o 3 GDL corresponde a 2010 mm e 360° de arco; o 5 GDL tem 3101 mm de diâmetro de limites e 360° de arco máximo.

4.2 PROJETO DO CIRCUITO ELETRÔNICO DE ACIONAMENTO E CONTROLE DE MOTORES

Fazendo uso da rede elétrica de 60 Hz, cujo período é $T=1/60$ correspondendo a cerca de 16,17ms equivalente a 360° (senóide), a passagem por zero da tensão ocorre a cada 180° , ou seja $T/2$ que é equivalente a aproximadamente 8,33 ms. Primeiramente foi configurado o *timer* do microcontrolador para 51,2 microsegundos, em que monitora o recebimento do disparo (*zero_crossing*). A cada vez que o pino muda seu estado de nível lógico 0 para 1 ocorre a passagem por zero da rede elétrica, nesse momento inicia-se a contagem da variável que representa o ângulo de disparo (incrementada) a cada 51,2 microsegundos.

Essa fragmentação do ângulo do semicírculo ocorre por divisão de 8,33 milisegundos por 51,2 microsegundos, produzindo aproximadamente 162 interrupções, que representa 162 ângulos de disparos, com $(180^\circ / 162 \text{ interrupções}) 1,11^\circ$ de variação por interrupção. O supervisório envia uma *string*, e em seu corpo contém informação de qual faixa de 0 a 162 preciso para controlar a potência entregue ao motor, esquema de funcionamento Figura 4.10.

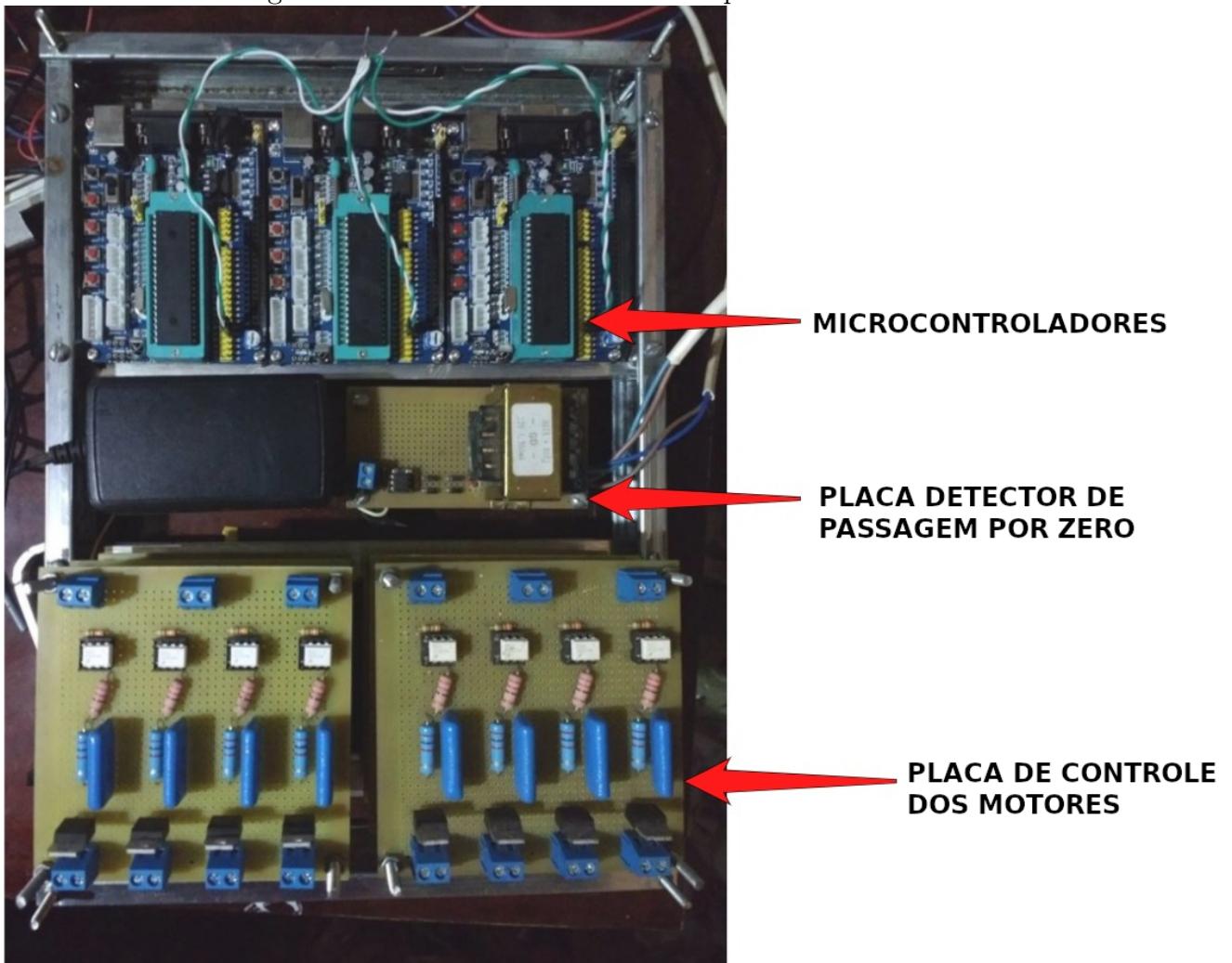
Figura 4.10 - Esquema de controle do ângulo de disparo.



Fonte: [Modificado de Laboratório de Garagem \(2014\)](#).

Para tornar o sistema mais seguro evitando mau contato ou curto-circuito, foi desenvolvido painel elétrico que integra todas as partes de controle e potência do manipulador. Conforme a Figura 4.11 é possível visualizar o barramento Tx/Rx entre as placas dos microcontroladores representados pelos fios branco e verde.

Figura 4.11 - Painel elétrico de testes preliminares.



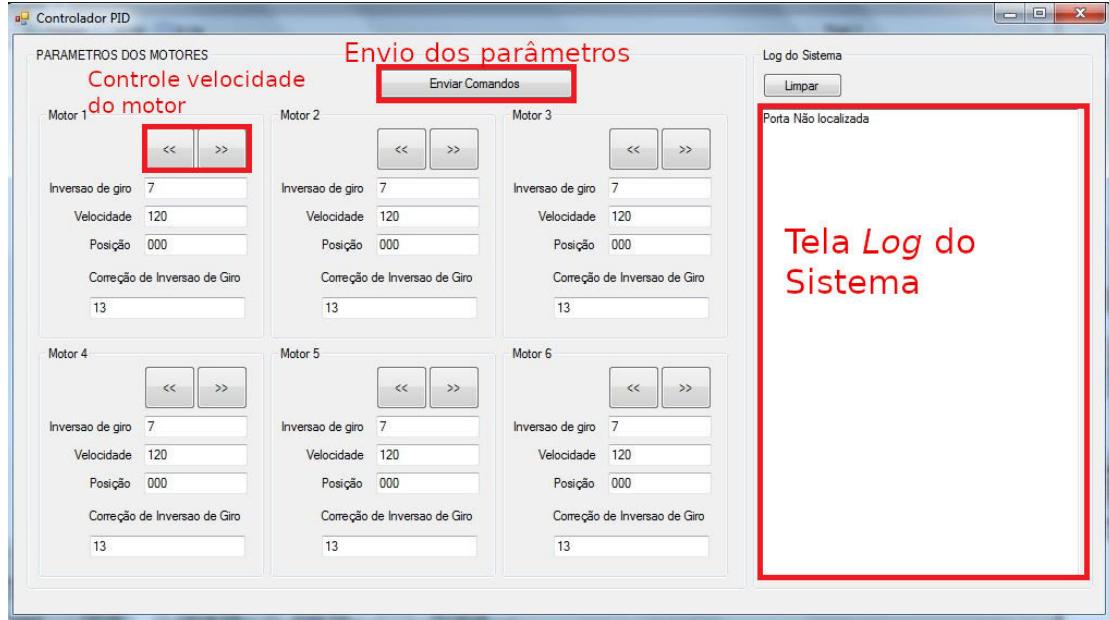
Fonte: Própria (2018).

Foram testados as placas de controle de potência com base na programação estabelecida no Proteus® (versão de demonstração), logo após, fez-se testes de detector de passagem por zero e controle do ângulo de disparo enviados do supervisório, resultando no controle da potência entregue ao motor universal. A interface entre a parte eletroeletrônica e mecânica está na seção 4.3.

4.3 DESENVOLVIMENTO DO SUPERVISÓRIO

O supervisório responsável pela entrada de dados é representado na Figura 4.12.

Figura 4.12 - Supervisório desenvolvido.



Fonte: Própria (2018).

O comando gerado pelo supervisório (Fig. 4.13) possui parâmetros para inverter o sentido de giro dos motores. O número **1** representa sentido horário, o número **0** sentido anti-horário. Os LEDs são usados como indicação no painel elétrico de que os TRIACs foram acionados e existe passagem de corrente elétrica AC no circuito de potência, é portanto um alerta ou medida de segurança.

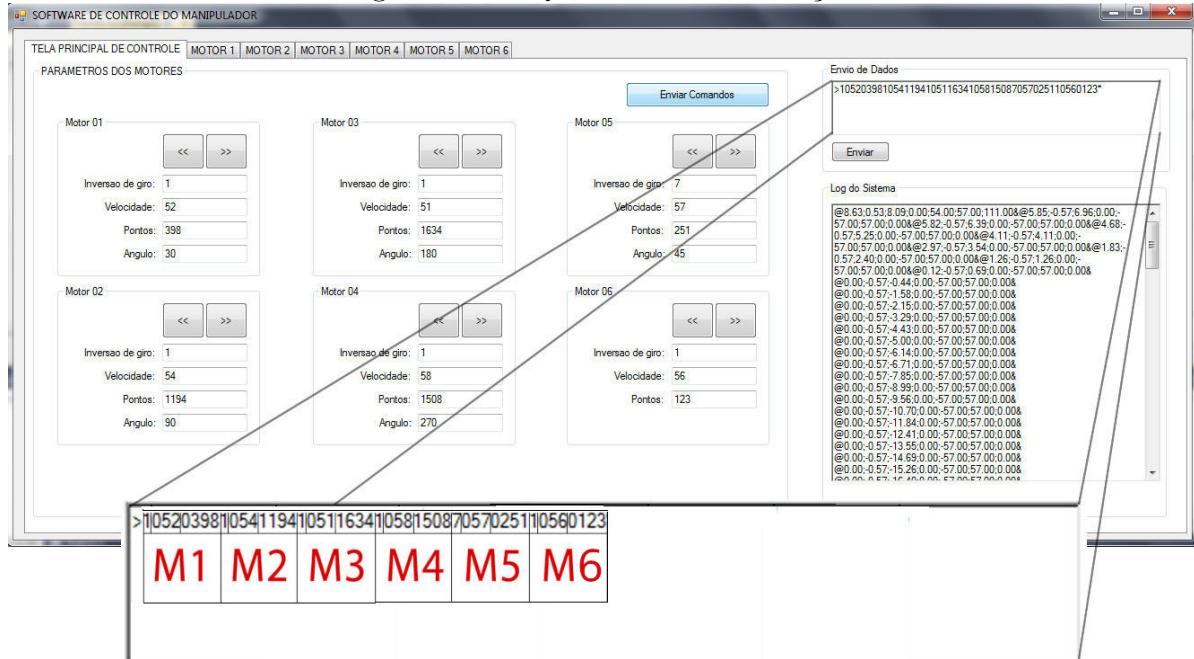
As cores verde e amarelo representam disparos de acionamento dos motores nos sentidos horário; os LEDs verde e vermelho no sentido anti-horário. No painel elétrico (versão final) os LEDs amarelo e verde estão acesos confirmando o comando para sentido horário de giro dos motores (Fig. 4.14).

O campo de inversão de sentido de giro foi desenvolvido apenas para testar os motores na ausência de envio de pontos. No *hardware*, a informação dos pontos passados do supervisório são comparados com os pontos anteriores, e com base nessa diferença de pontos maiores com menores o sentido de giro é definido.

A Figura 4.15 apresenta o algoritmo de conversão de ângulos para pontos em C#. O Visual Studio C# é orientado a objetos e a eventos, o que facilita o desenvolvimento de algoritmos complexos baseados em eventos. O evento da Figura 4.15 faz parte da escrita

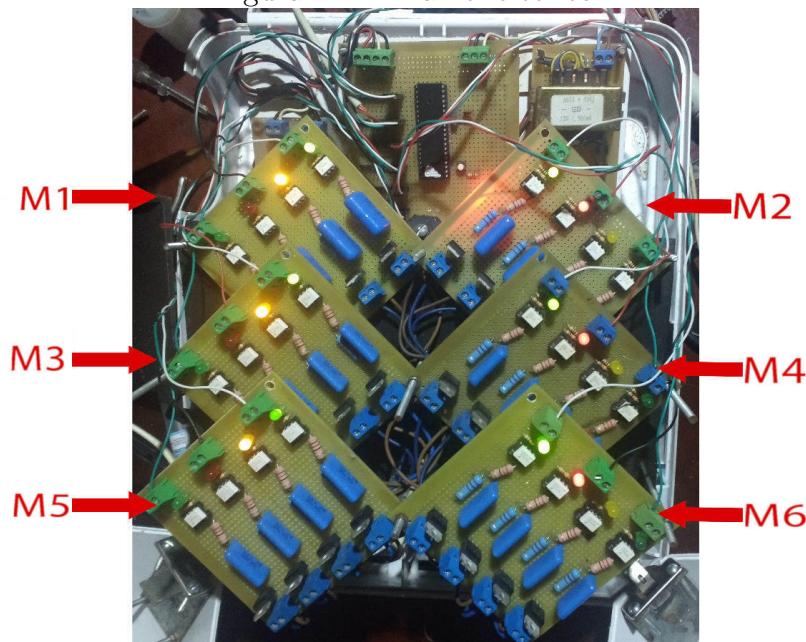
da função **TextChanged**, ao mudar o estado. Assim quando o texto é digitado no campo, executa o cálculo de conversão de ângulos para pontos.

Figura 4.13 - Quadro de comunicação.



Fonte: Própria (2018).

Figura 4.14 - Painel elétrico.



Fonte: Própria (2018).

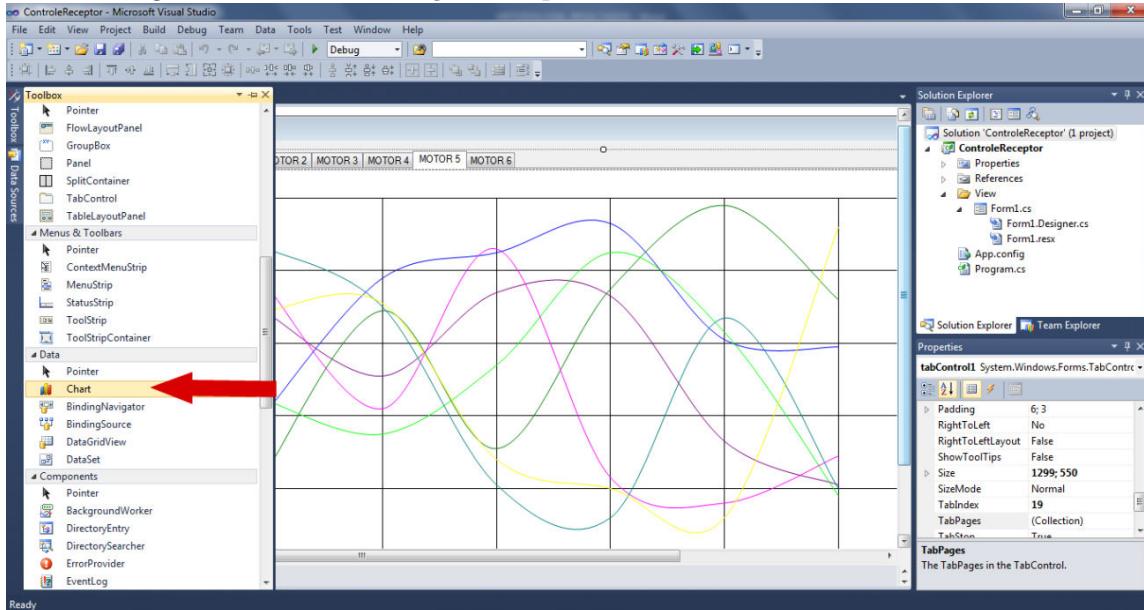
Figura 4.15 - Algoritmo de conversão de ângulos em pontos.

```
private void TB_M01_ANGULO_TextChanged(object sender, EventArgs e)
{
    if (TB_M01_ANGULO.Text != "")
    {
        double dado = (3.1416 * 190 * 12 * Convert.ToInt16(TB_M01_ANGULO.
            Text)) / (1.5 * 360);
        tb_pic_01_m01_pos.Text = Math.Round(dado).ToString();
    }
    else {
        tb_pic_01_m01_pos.Text = "";
    }
}
```

Fonte: Própria (2018).

A seta vermelha na Figura 4.16, indica opção de criação de gráficos. Para visualizar o gráfico apresentado, na aba lateral esquerda, opção **toolbox**, clicar em **Data -> Chart** e arrastar para a tela de trabalho. O código da Figura 4.17 é o modo de recepção, filtro dos dados obtidos pela serial e geração do gráfico.

Figura 4.16 - Visualização dos parâmetros de controle dos motores.



Fonte: Própria (2018).

Figura 4.17 - Modo de recepção dos dados e geração do gráfico.

```
string[] parts = dados.Split(new string[] { "@" }, StringSplitOptions.  
    ↪ None);  
if (parts.Length == 2)  
{  
    string[] parts2 = parts[1].Split(new string[] { "&" },  
        ↪ StringSplitOptions.None);  
    if (parts2.Length == 2)  
    {  
        char[] delimitador = { ';' };  
        string[] VETOR_DADOS = parts2[0].ToString().Split(delimitador)  
            ↪ ;  
  
        GRAFICO_M05.Series["PID"].Points.AddY(VETOR_DADOS[0]);  
        GRAFICO_M05.Series["PROPORCIONAL"].Points.AddY(VETOR_DADOS[1])  
            ↪ ;  
        GRAFICO_M05.Series["INTEGRATIVO"].Points.AddY(VETOR_DADOS[2]);  
        GRAFICO_M05.Series["DERIVATIVO"].Points.AddY(VETOR_DADOS[3]);  
        GRAFICO_M05.Series["ERRO"].Points.AddY(VETOR_DADOS[4]);  
        GRAFICO_M05.Series["SETPOINT"].Points.AddY(VETOR_DADOS[5]);  
        GRAFICO_M05.Series["SAIDA"].Points.AddY(VETOR_DADOS[6]);  
  
        TB_M05_PID.Text = VETOR_DADOS[0];  
        TB_M05_PROPORCIONAL.Text = VETOR_DADOS[1];  
        TB_M05_INTEGRAL.Text = VETOR_DADOS[2];  
        TB_M05_DERIVATIVO.Text = VETOR_DADOS[3];  
        TB_M05_ERRO.Text = VETOR_DADOS[4];  
        TB_M05_SETPOINT.Text = VETOR_DADOS[5];  
        TB_M05_SAIDA.Text = VETOR_DADOS[6];  
  
        GRAFICO_M05.Update();  
    }  
}
```

Fonte: Própria (2018).

Conforme os dados são passados do microcontrolador para o supervisório, a *string* é convertida em quadro de dados em que cada índice representa um parâmetro. Estes parâmetros são enviados a cada cálculo do algoritmo PID estimado em aproximadamente 200 ms.

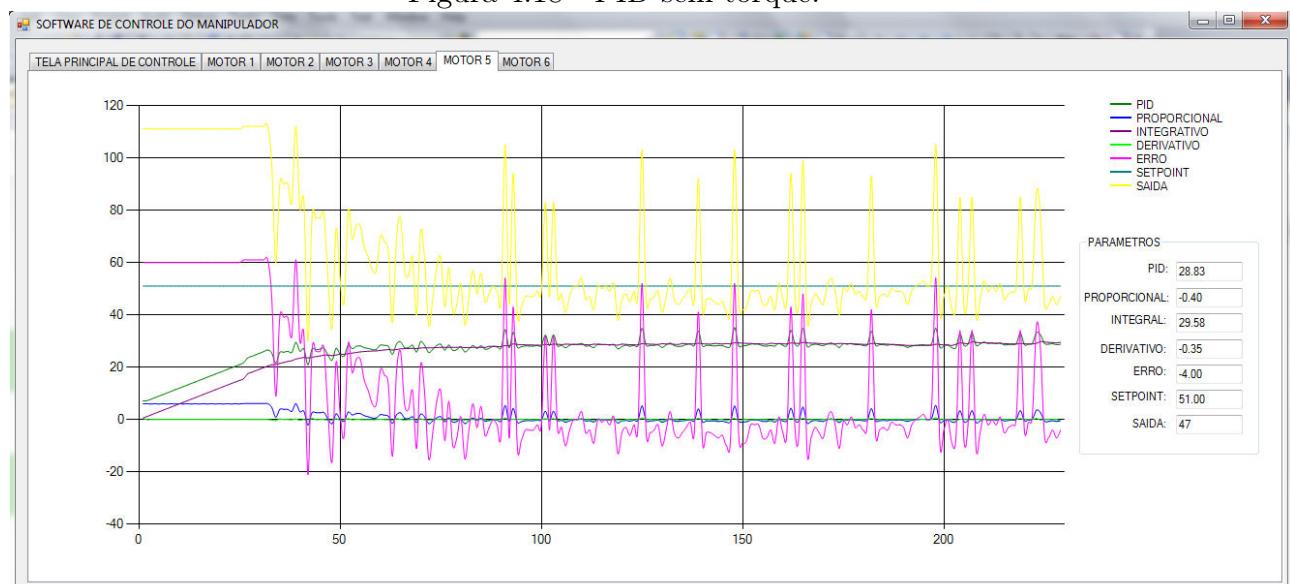
Quando o quadro é enviado do PIC para o supervisório, um pequeno distúrbio foi detectado no algoritmo de dimerização. Esse distúrbio é provocado quando os dados são enviados, a função de escrita **printf()** causa atraso no microcontrolador e altera o tempo da contagem do algoritmo. O atraso é o tempo de mostrar determinada mensagem com a

função citada (tempo de processamento). Mensagens consideradas longas (número excessivo de caracteres) foram identificadas nos testes como a causa de distúrbios no funcionamento do programa. O distúrbio também é ocasionado na recepção do quadro enviado do supervisório ao microcontrolador. O `printf()` foi utilizado apenas para gerar o gráfico do controle PID em tempo real, portanto não foi usado para mostrar mensagens, evitando assim que ocorra distúrbios (mensagens de *log* e de poucos caracteres não alteram o funcionamento do programa).

O gráfico da Figura 4.18 mostra o comportamento do controlador de velocidade sem presença de torque constante, assim o PID mantém determinada potência. De acordo com os movimentos feitos pelo manipulador, o momento gerado de determinado eixo de rotação pode variar, assim o torque exigido para gerar o movimento também pode aumentar ou diminuir, justifica portanto a necessidade do controlador. A quantidade de potência entregue ao motor deve ser proporcional mantendo a mesma velocidade angular.

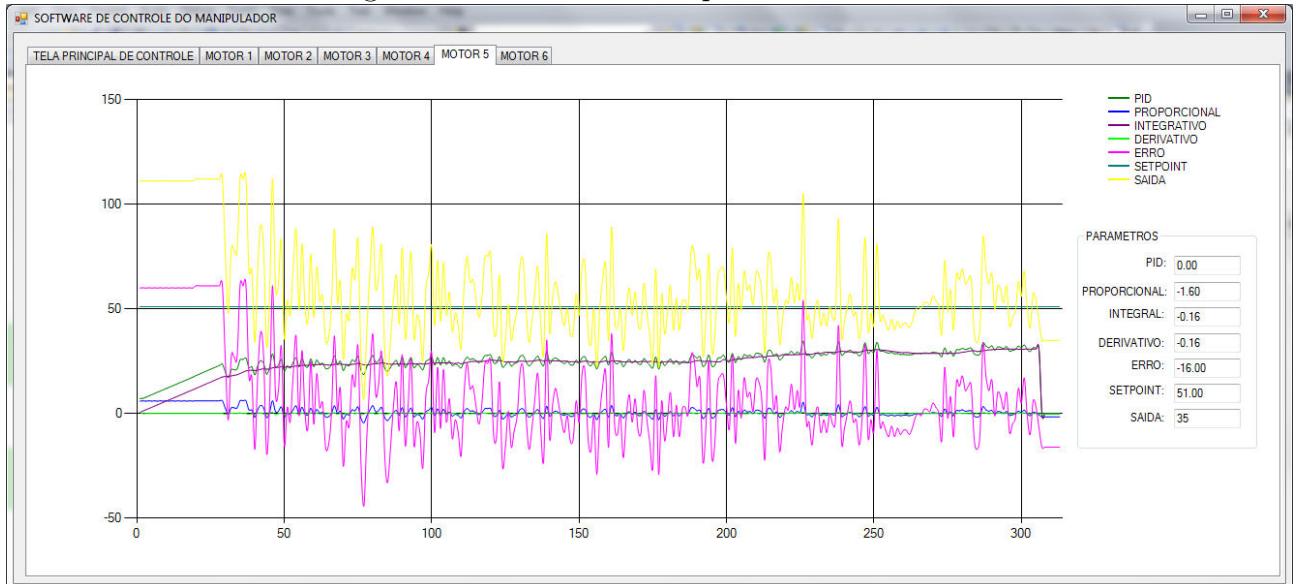
O gráfico da Figura 4.19 mostra o comportamento do controlador PID quando existe torque constante. Nota-se que a potência tem aumento gradativo até alcançar estado de regime permanente, apesar do período pequeno de execução do algoritmo. Porém, como o disco *encoder* com sensor óptico não está no eixo principal do motor, pode haver oscilação devido a margem da estabilização do erro. Ambos gráficos foram gerados em tempo real, os valores apresentados nas legendas são informações em tempo real, e não os ganhos (proporcional, integral e derivativo) definidos no algoritmo de controle PID implementado no microcontrolador.

Figura 4.18 - PID sem torque.



Fonte: Própria (2018).

Figura 4.19 - PID com torque constante.



Fonte: Própria (2018).

4.4 CINEMÁTICA DO MANIPULADOR ROBÓTICO ACADÊMICO

A garra é representada pelo L(9) (ligamento 9); o L(1) (ligamento 1) e L(6) (ligamento 6) tem o movimento de rotação em torno do próprio eixo (Fig. 4.20). A Tabela 4.1 apresenta os comandos para criar cada ligamento do modelo (identificação dos parâmetros dos ligamentos na Tabela 3.1).

Tabela 4.1 - Comandos para criar cada ligamento.

```

L(1) = rt_link([%pi/2 0 0 0 0], "standard");
L(2) = rt_link([0 1.24968 0 0 0], "standard");
L(3) = rt_link([0 0 0 -0.29039 0], "standard");
L(4) = rt_link([0 0.5 0 0 0], "standard");
L(5) = rt_link([-%pi/2 0 0 -0.20873 0], "standard");
L(6) = rt_link([%pi/2 0 0 0 0], "standard");
L(7) = rt_link([0 0.54679 0 0 0], "standard");
L(8) = rt_link([0 0.504 0 0 0], "standard");
L(9) = rt_link([0 0 0 0 0], "standard");

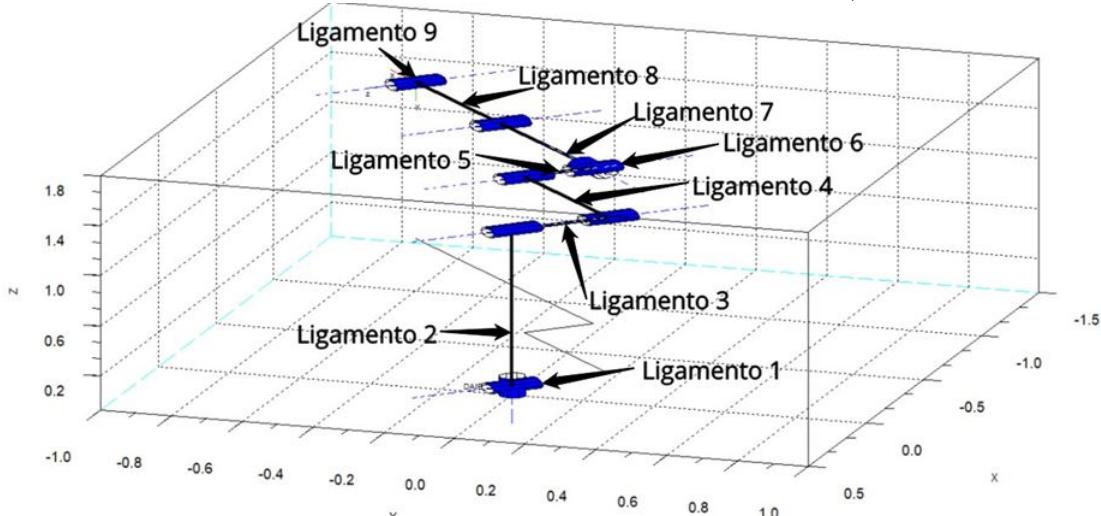
```

Fonte: Própria (2018).

O *software* adotado possibilita visualização de trajetórias, pois utiliza como variável de mudança de posição os ângulos das juntas em radianos. O comando *rt_drivebot(DAIE)* mostra qual a posição do efetuador no plano cartesiano, e também o valor do ângulo de

rotação de cada uma das juntas, como visto na Figura 4.21.

Figura 4.20 - Ligamentos do manipulador robótico acadêmico (medidas em metro).



Fonte: Própria (2018).

A posição inicial do manipulador robótico recebe como configuração das juntas o seguinte vetor de ângulos: $initial=[0 \%pi/2 \%pi/2 0 \%pi/2 0 -\%pi/2 0 0]$. Resulta na posição da Figura 4.21. O modo de visualizar a trajetória de movimento se dá principalmente por duas linhas de comando, Tabela 4.2, na qual t é o vetor tempo, seu ajuste resulta em movimentação mais lenta ou rápida, jt armazena matrizes que representam o cálculo da trajetória entre duas posições de juntas.

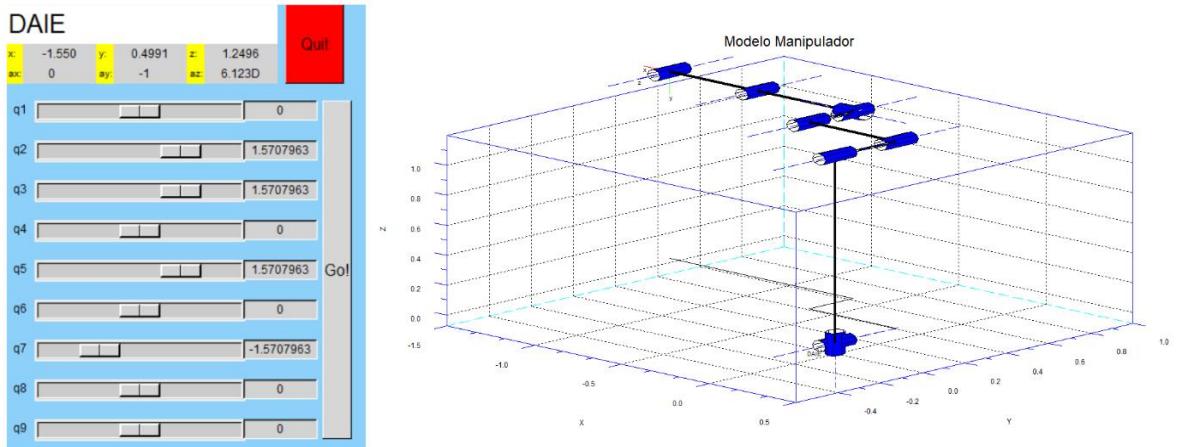
Como exemplo, realizando alterações com base no vetor inicial, nas juntas 1 e 3, a posição final é dada pela Figura 4.22. Desta forma, é possível realizar movimentos específicos, com o valor exato do ângulo da junta a qual deseja movimentar. Movimentação por juntas torna-se trabalhoso, tendo que realizar muitas alterações até encontrar de forma manual a posição desejada. Portanto, foi definido que o algoritmo deve ter como variável de entrada o ponto de chegada, por meio de coordenadas cartesianas XYZ.

Tabela 4.2 - Comandos para criar e visualizar trajetória.

Parâmetro	Legenda
$t = [0 : 0.056 : 10];$	Vetor tempo.
$jt = rt_jtraj(qready, qstretch, t);$	Calcula trajetória entre duas posições de juntas.
$rt_plot(DAIE, jt);$	Cria animação gráfica do movimento.

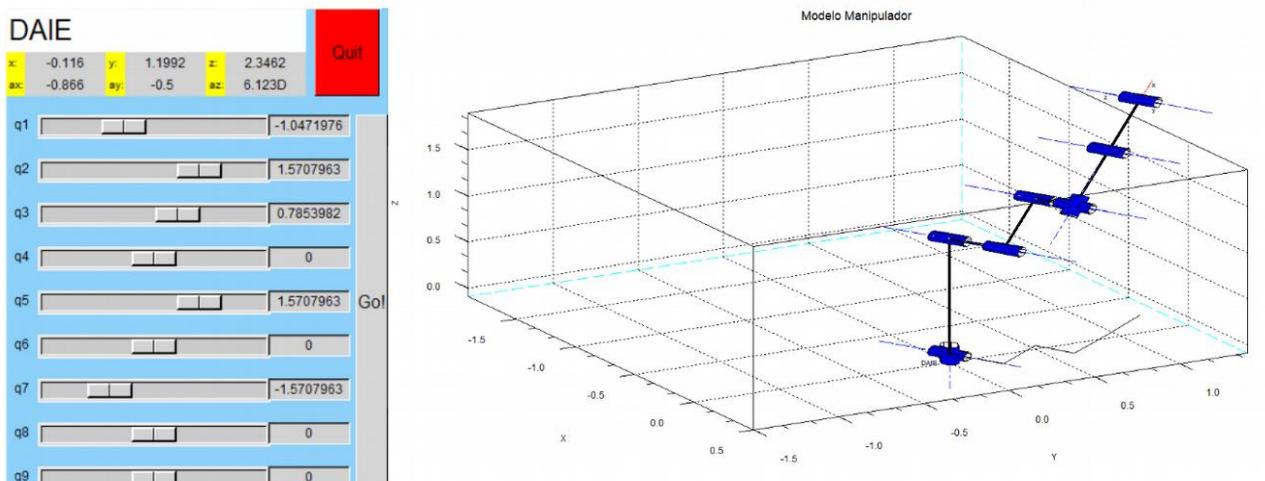
Fonte: Modificado de rtss.sourceforge.net(2018c)

Figura 4.21 - Posição inicial.



Fonte: Própria (2018).

Figura 4.22 - Posição após movimentação.



Fonte: Própria (2018).

O fluxograma da Figura 4.23 é referente ao algoritmo fundamental, criado durante desenvolvimento da solução da cinemática do robô. Intitulado algoritmo fundamental por aperfeiçoar o método de obter o posicionamento final em relação ao ponto inicial.

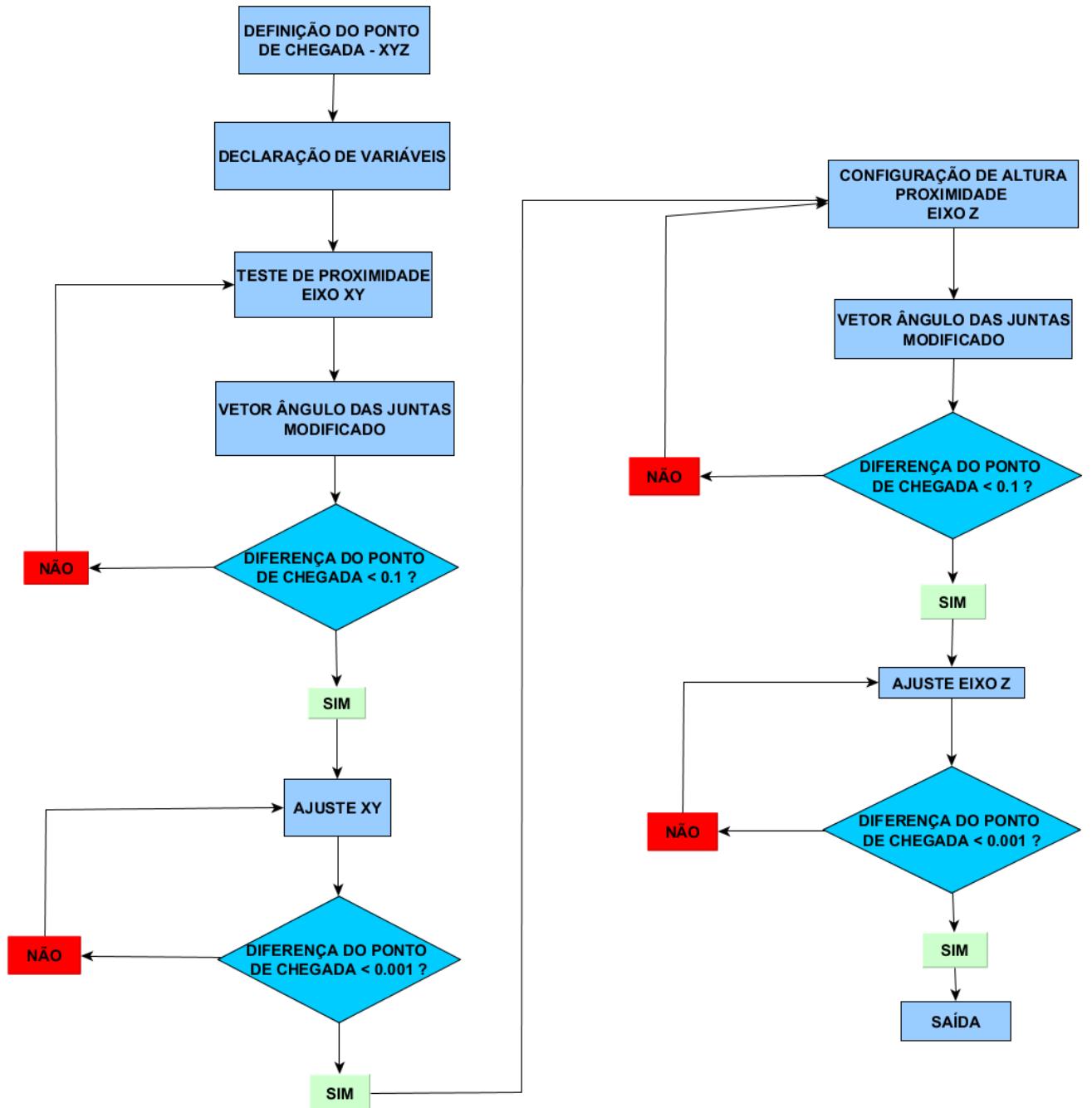
Primeiro é definido o ponto de chegada, eixos XYZ. Na etapa declaração de variáveis são criadas duas variáveis *ang* e *ang1* as quais recebem o vetor ângulo inicial das juntas. Os testes de proximidades são *loops* os quais percorrem 180° (ângulo da junta), a variável *ang* é modificada mais 1°(um grau) enquanto que *ang1* menos 1° (um grau). O comando T=r_fkine(DAIE,VETOR_ÂNGULO) – calcula a cinemática direta do manipulador, retorna matriz de transformada homogênea para determinado vetor de ângulo das juntas -

o segundo parâmetro é modificado com as variáveis de ângulo, com o objetivo de encontrar a posição final da garra, correspondente aos elementos p_x , p_y , p_z , como exemplificado na Tabela 2.1, da matriz encontrada. Ao encontrar pontos correspondentes com diferença menor do que 0,1 em relação ao ponto objetivo, o programa segue para ajustar os valores encontrados, também *loop*, percorre 180° , desta vez avançando $0,01^\circ$ a cada teste para achar valores com diferença menor que 0,001 do ponto objetivo, diferença de 0,1%.

Em relação ao eixo XY é modificado a junta 1, para aproximar a altura é modificado a junta 3. Para ajustar a solução final ao eixo XY é modificado as juntas 5, 6 e 8. Sendo a posição inicial da Figura 4.21, a coordenada do ponto objetivo é $(1.4, -0.5, 1.23)$. Com o intuito de demonstrar o funcionamento do algoritmo, é testado o ajuste somente ao eixo X. O algoritmo retorna aumento de 48° o valor de $x = -1.4085995$, Figura 4.24 enviando $+180^\circ$ para a junta 1 então o valor de $x = 1.4085995$, Figura 4.25.

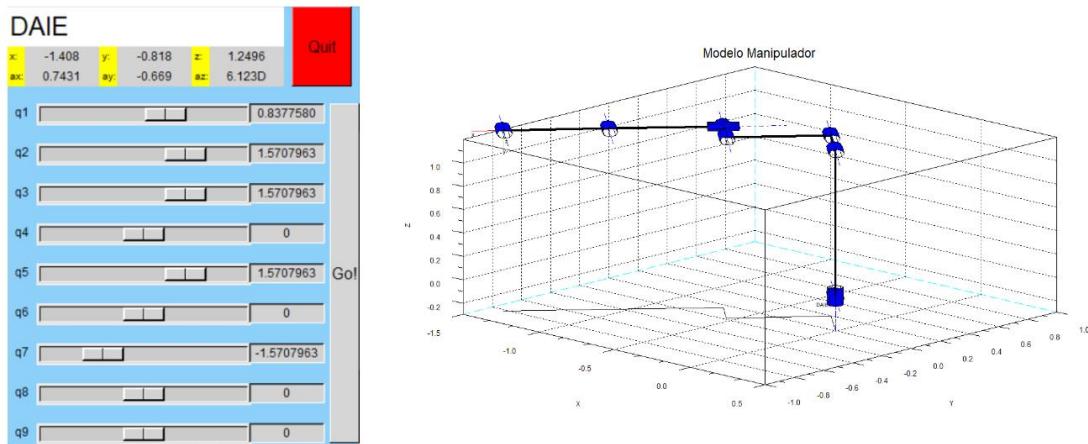
O primeiro ajuste é percorrido o ângulo a cada 0.001° , é encontrado a transformada X e $x = 1.4013888$. O teste de outro algoritmo desenvolvido, define diferença 0.000001 em relação ao ponto objetivo, ainda percorrendo 0.01° a cada iteração, o valor de $x = 1.3999949$. Resultados apresentados na Figura 4.26 e na Tabela 4.3.

Figura 4.23 - Fluxograma algoritmo de movimentação.



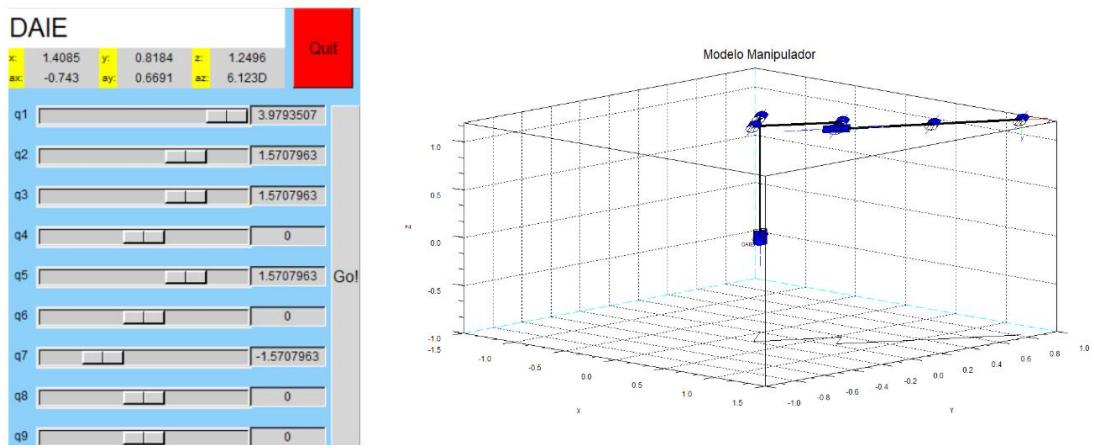
Fonte: Própria (2018).

Figura 4.24 - Resultado do teste de proximidade ao ponto x.



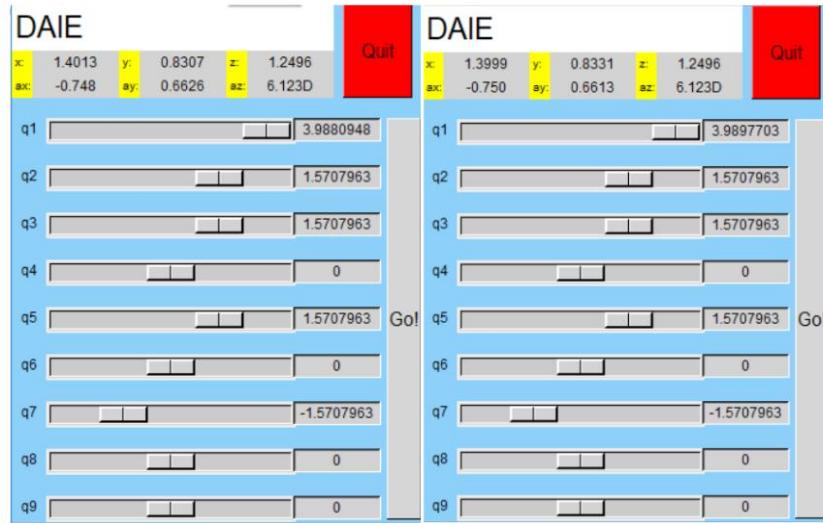
Fonte: Própria (2018).

Figura 4.25 - Giro de 180° na junta 1.



Fonte: Própria (2018).

Figura 4.26 - Resultados dos ajustes ao ponto objetivo.



Fonte: Própria (2018).

Tabela 4.3 - Matrizes transformadas homogêneas.

Vetor ângulo das juntas	[0.83775 1.57079 1,57079 0. 1,57079 0. -1,57079 0. 0.]
Aproximação do ponto x	$\begin{bmatrix} -0,66913 & 0. & 0,74314 & -1,40859 \\ -0,74314 & 0. & -0,66913 & -0,81848 \\ 0. & -1. & 0. & 1,24968 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Giro de 180°	$\begin{bmatrix} 0,66913 & 0. & -0,74314 & 1,40859 \\ 0,74314 & 0. & 0,66913 & 0,81848 \\ 0. & -1. & 0. & 1,24968 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Primeiro ajuste ao ponto x	$\begin{bmatrix} 0,66260 & 0. & -0,74896 & 1,40138 \\ 0,74896 & 0. & 0,66260 & 0,83077 \\ 0. & -1. & 0. & 1,24968 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Melhor ajuste alcançado	$\begin{bmatrix} 0,66135 & 0. & -0,75007 & 1,399 \\ 0,75007 & 0. & 0,66135 & 0,83311 \\ 0. & -1. & 0. & 1,24968 \\ 0. & 0. & 0. & 1. \end{bmatrix}$

Fonte: Própria (2018).

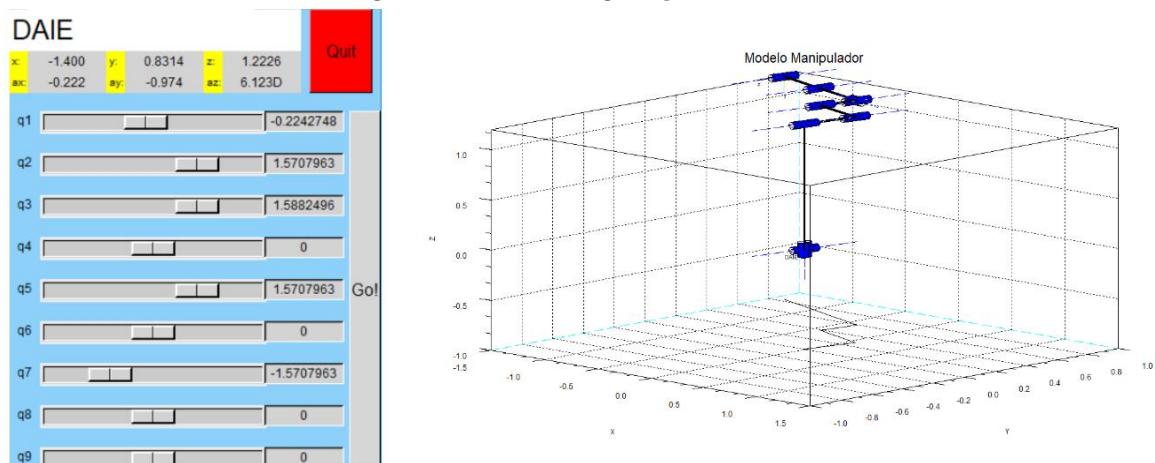
A Tabela 4.3 apresenta as transformadas homogêneas resultantes de cada etapa do algoritmo, os três primeiros elementos da quarta coluna são os pontos cartesianos XYZ. Na aproximação do ponto X é encontrado valor negativo, então é necessário girar a base do manipulador em 180° . O teste de proximidade e ajuste para um eixo retornou os resultados buscados, é então aplicado os mesmos algoritmos para o plano XY e XYZ. Para configuração da altura a junta 3 passa a ser modificada. O resultado da aproximação da altura, Figura 4.27 e Tabela 4.4.

Tabela 4.4 - Resultado da transformação simultânea XY e XYZ.

Plano XY	$\begin{bmatrix} 0,97495 & 0. & 0,22239 & 1,40094 \\ -0,22239 & 0. & 0,97495 & -0,83151 \\ 0. & -1, & 0. & 1,24968 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Plano tridimensional XYZ	$\begin{bmatrix} -0,97480 & 0,01701 & -0,22239 & -1,40071 \\ 0,22236 & -0,00388 & -0,97495 & 0,83146 \\ -0,01745 & -0,99984 & 0. & 1,2226 \\ 0. & 0. & 0. & 1. \end{bmatrix}$

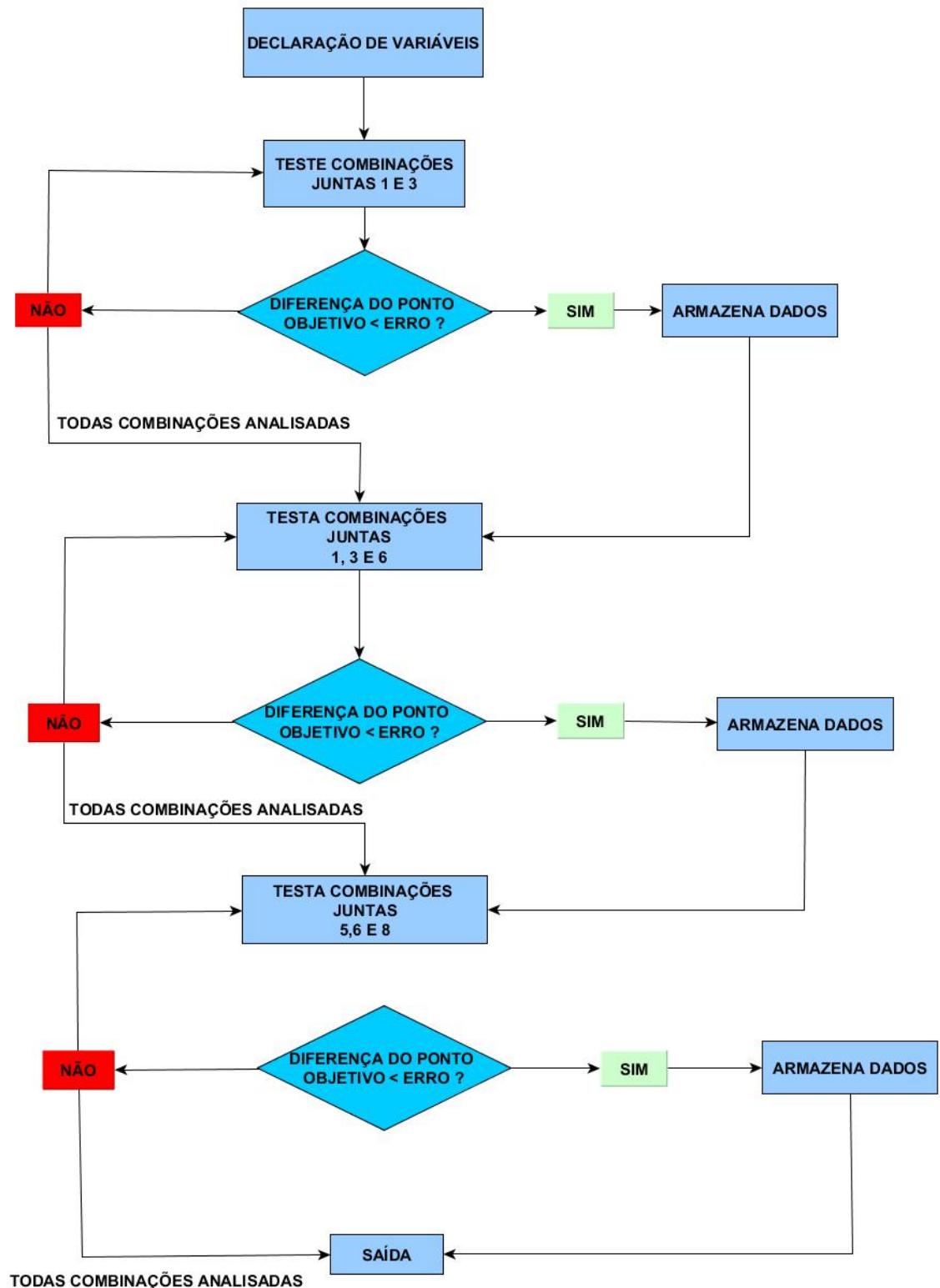
Fonte: Própria (2018).

Figura 4.27 - Configuração da altura.



Fonte: Própria (2018).

Figura 4.28 - Fluxograma do código fonte.



Fonte: Própria (2018).

O algoritmo de posicionamento busca o melhor posicionamento modificando as juntas 1 e 3. Configurando somente estas duas juntas obteve-se resultados satisfatórios, porém, não explora todas as possibilidades oferecidas pelo manipulador de cinco graus de liberdade. Com este pensamento, o desafio é criar lógica que considere todas as juntas de movimento – juntas 1, 3, 5, 6 e 8 – de forma ordenada, as trajetórias resultantes devem ser possíveis e não apresentar falhas como passar pelo meio do robô ou movimentar juntas que permanecerão estáticas – juntas 2, 4, 7, 9. O fluxograma do código que gera os resultados desta seção, Figura 4.28.

Para demonstrar os resultados obtidos com este algoritmo, primeiramente será apresentado análises individuais da coordenada x, y e z. Foi testado o ponto $x = -0,9$ e em seguida $x = 1,29$, Figuras 4.29 e 4.30 respectivamente. De forma semelhante os testes para a coordenada $y = 1$, $y = 0,76$; $z = 0,5$ e $z = 1,97$. O erro percentual ser menor do que 1% foi definido no código fonte, em que para análise de uma coordenada, no fluxograma há três testes condicionais em que o valor do erro menos a diferença do valor objetivo – sendo igual a 0,1 no primeiro bloco, o objetivo é identificar mais rapidamente uma combinação aproximada para então refinar os próximos dois blocos de testes para um valor do erro de 0,01 e 0,001. As Tabelas 4.5 e 4.6 contém os dados das transformadas homogêneas de cada movimento testado, e o erro percentual de cada resultado calculado.

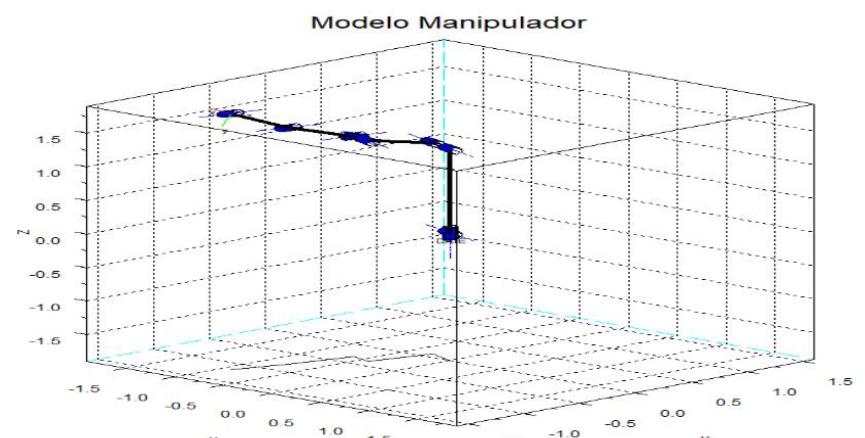
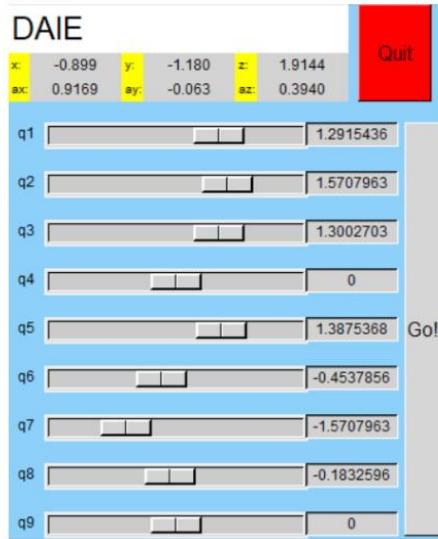
Tabela 4.5 - Estudo do algoritmo aplicado ao ponto x.

Matriz de movimentação ponto x	$\begin{bmatrix} -0,30059 & 0,26240 & 0,91694 & -0,89955 \\ -0,75846 & -0,64865 & -0,06301 & -1,18025 \\ 0,57824 & -0,71441 & 0,39400 & 1,91443 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
X	- 0,8995528
Erro	0,05%
Matriz de movimentação ponto x	$\begin{bmatrix} 0,92402 & 0,17871 & 0,33799 & 1,29539 \\ -0,26495 & -0,33799 & 0,90308 & -0,89068 \\ 0,27563 & -0,92402 & -0,26495 & 1,67713 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
X	1,29539
Erro	0,41%
Fonte: Própria (2018).	

Tabela 4.6 - Estudo do algoritmo aplicado aos pontos y e z.

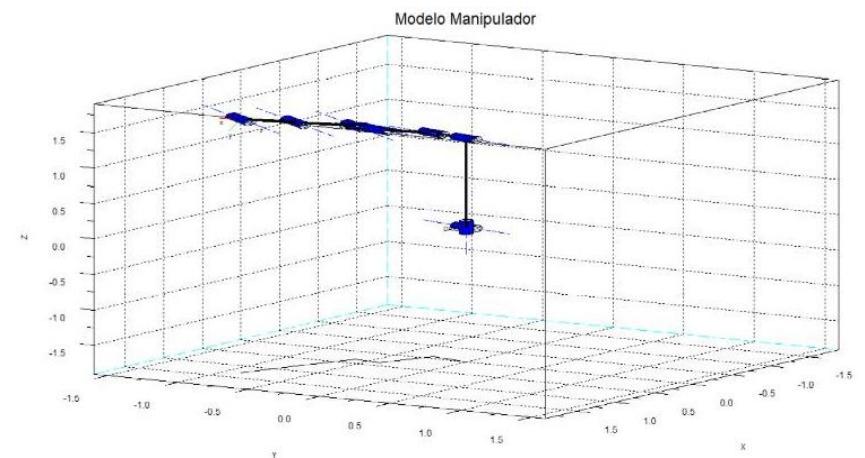
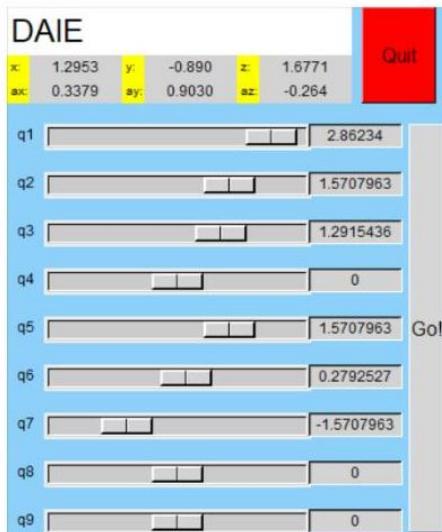
Matriz de movimentação ponto Y	$\begin{bmatrix} -0,96029 & 0,19030 & 0,20399 & -1,58699 \\ -0,18604 & 0,10805 & -0,97658 & 0,20029 \\ -0,20789 & -0,97576 & -0,06835 & 0,94048 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Y	0,20029
Erro	0,15%
Matriz de movimentação ponto Y	$\begin{bmatrix} -0,84200 & 0,15257 & -0,51744 & -1,30420 \\ 0,51744 & 0,49969 & -0,69465 & 0,75991 \\ 0,15257 & -0,85265 & -0,49969 & 1,7199 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Y	0,7599147
Erro	0,11%
Matriz de movimentação ponto Z	$\begin{bmatrix} -0,16635 & -0,25513 & -0,95248 & -0,20637 \\ -0,98338 & -0,02829 & 0,17933 & 0,08908 \\ -0,07269 & 0,96649 & -0,24618 & 0,50041 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Z	0,50041
Erro	0,083%
Matriz de movimentação ponto Z	$\begin{bmatrix} -0,28262 & 0,40485 & 0,86960 & -0,70768 \\ -0,52524 & -0,82389 & 0,21286 & -1,19284 \\ 0,80264 & -0,39660 & 0,44550 & 1,97043 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Z	1,97043
Erro	0,02%
Fonte: Própria (2018).	

Figura 4.29 - Posição para x=-0,9.



Fonte: Própria (2018).

Figura 4.30 - Posição para x=1,29.



Fonte: Própria (2018).

Ainda utilizando este mesmo algoritmo, que foi testado e demonstrou resultados coerentes, os próximos testes foram para os planos XY, XZ, YZ e XYZ. Para os planos XY e XZ a taxa de tolerância definida (erro em relação ao ponto objetivo) foi de 10%, para o YZ de 50% e para o sistema cartesiano tridimensional XYZ de 80%. Esta tolerância significa que nos cálculos os valores armazenados para estudo são considerados simultaneamente em relação aos respectivos planos, os testes de diferentes combinações das juntas geram pontos os quais são comparados com o ponto objetivo. Quando esta diferença for de 0,1,

0,5, 0,8, definido conforme a necessidade de cada simulação, a configuração de junta e a transformada homogênea resultante são armazenadas, o *loop* atual é finalizado e o programa segue para os próximos passos.

Os cálculos para um único eixo demonstram resultados com erros tendendo a zero em alguns casos, pela característica de análise simultânea faz-se necessário definição da tolerância aqui descrita.

O estudo detalhado nesta seção demanda não só cálculos computacionais e raciocínio para criar, testar e definir qual melhor solução para o problema de posicionamento, mas conhecimento do robô em estudo, principalmente noções espaciais. Ao ver rapidamente qualquer uma das fotos contidas neste trabalho é possível enxergar possibilidades de posicionamento, e rejeitar coordenadas incoerentes. As simulações e estudo do modelo desenvolvido auxiliam no conhecimento dos limites de posicionamento – pontos máximos e mínimos em cada um dos eixos.

Ao enviar coordenadas impossíveis para serem analisadas simultaneamente, o que acontece é a grande possibilidade das combinações não estarem dentro de nenhuma das combinações calculadas, e retornar resultado nulo. A Tabela 4.7 explicita os dados da análise de XY, Tabela 4.8 de XZ, Tabela 4.9 YZ e Tabela 4.10 sistema cartesiano XYZ.

Tabela 4.7 - Plano XY.

X - definido	-1.08
Y - definido	-0.89
Matriz de movimentação ponto XY	$\begin{bmatrix} -0.41642 & 0.21541 & 0.88328 & -1.08648 \\ -0.78318 & -0.57841 & -0.22817 & -0.98023 \\ 0.46174 & -0.78678 & 0.40957 & 1.96575 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
X - calculado	- 1,08648
Erro	0,60%
Y - calculado	- 0,98023
Erro	10,14%
Fonte: Própria (2018).	

Tabela 4.8 - Plano XZ.

X - definido	-1,12
Z - definido	0,43
Matriz de movimentação ponto XZ	$\begin{bmatrix} -0,34856 & -0,86447 & 0,36218 & -1,11236 \\ -0,36218 & 0,48063 & 0,79863 & 0,31658 \\ -0,86447 & 0,14720 & -0,48063 & 0,48491 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
X - calculado	- 1,11236
Erro	0,68%
Z - calculado	0,48491
Erro	12,8%
Fonte: Própria (2018).	

Tabela 4.9 - Plano YZ.

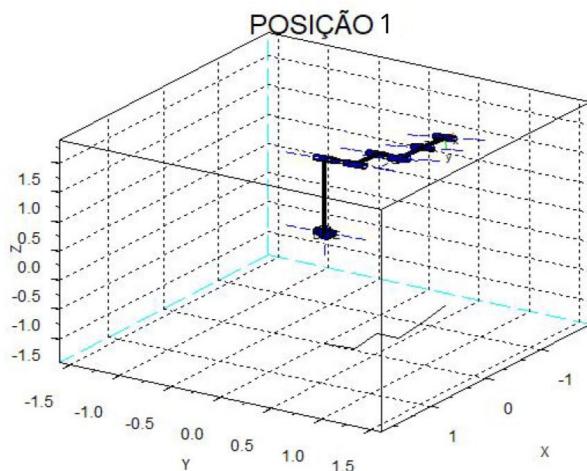
Y - definido	-1,6
Z - definido	0,99
Matriz de movimentação ponto YZ	$\begin{bmatrix} -0,20071 & 0,39364 & 0,89708 & 0,38928 \\ -0,97883 & -0,04331 & -0,20000 & -1,48910 \\ -0,03987 & -0,91824 & 0,39400 & 0,98988 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
Y - calculado	- 1,48910
Erro	6,93%
Z - calculado	0,98988
Erro	0,01%
Fonte: Própria (2018).	

Tabela 4.10 - Sistema tridimensional XYZ.

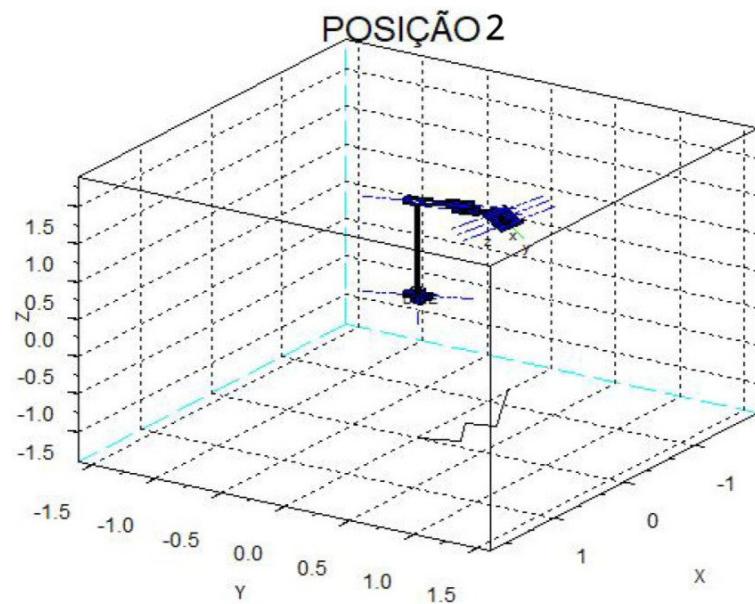
X - definido	-0,4
Y - definido	0,6
Z - definido	0,3
Matriz de movimentação ponto YZ	$\begin{bmatrix} 0,13432 & -0,99065 & -0,02372 & -0,25608 \\ 0,34099 & 0,02372 & 0,93976 & 0,71174 \\ -0,93041 & -0,13432 & 0,34099 & -0,19320 \\ 0. & 0. & 0. & 1. \end{bmatrix}$
X - calculado	-0,25608
Erro	35,9%
Y - calculado	0,71174
Erro	18,62%
Z - calculado	-0,19320
Erro	35,6%
Fonte: Própria (2018).	

O desenvolvimento do algoritmo leva em conta trajetórias possíveis, que não sejam incoerentes, como bater no próprio manipulador durante a simulação de movimentações. Pensando em segurança, foi aplicado divisão da trajetória, com o intuito de analisar cada posicionamento até o fim da movimentação. Na prática o programa espera pelo comando de entrada, para que possa continuar a exibir o trajeto, ou se melhor for, interromper a movimentação, parar no meio do caminho, ou voltar para posição original. Figura 4.31 retrata a divisão da trajetória da Tabela 4.10.

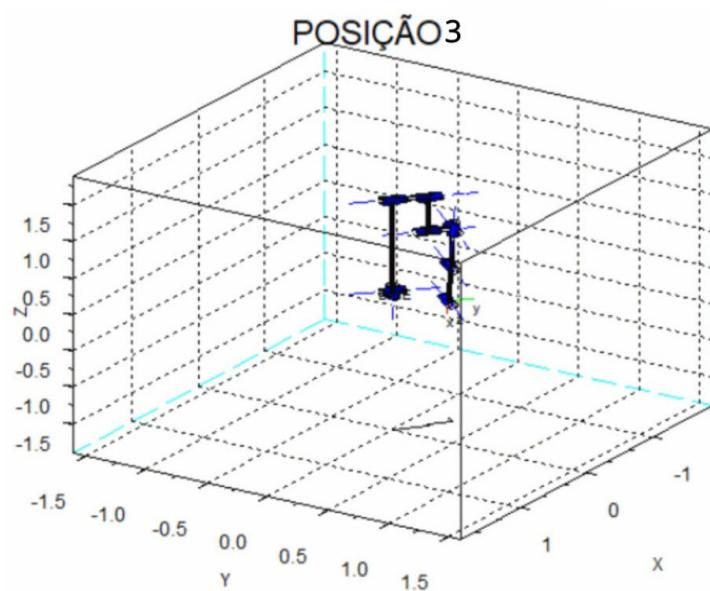
Figura 4.31 - Posições 1-4, etapa da divisão de trajetória:



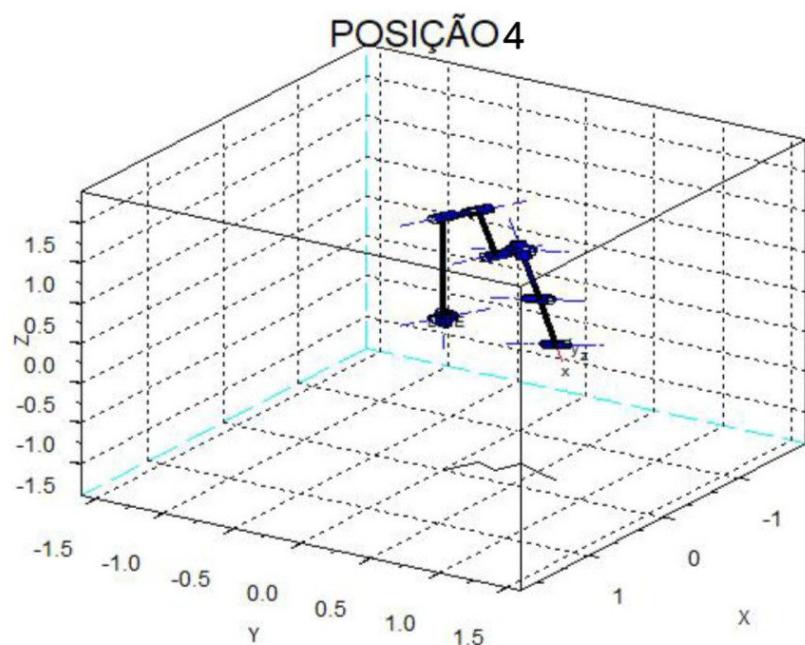
(a) Primeira.



(b) Segunda.



(c) Terceira.



(d) Quarta.

Figura 4.32 - Ângulos da posição final.

DAIE				Quif	
x:	-0.256	y:	0.7074	z:	-0.195
ax:	-0.017	ay:	0.9408	az:	0.3383
q1				1.1933268	
q2				1.5707963	
q3				3.5132187	
q4				0	
q5				1.5707963	
Gof					
q6				1.9424224	
q7				-1.5707963	
q8				0	
q9				0	

Fonte: Própria (2018).

4.5 CONSTRUÇÃO DO MANIPULADOR ROBÓTICO

Na seção 3.6 é apresentado as ferramentas e máquinas utilizadas para a construção do manipulador robótico. A primeira parte do robô a ser construído foi a garra, o material de aço inoxidável foi cortado e furado conforme as medidas definidas no projeto mecânico, ver dimensões dos ligamentos do manipulador (Fig. 3.16). A Figura 4.33 (a) apresenta o início da construção de partes específicas que compõem a garra, a Figura 4.33 (b) a parte mecânica da garra completa.

Figura 4.33 - Etapa de construção da garra.



(a) Partes da garra na furadeira.

(b) Garra montada.

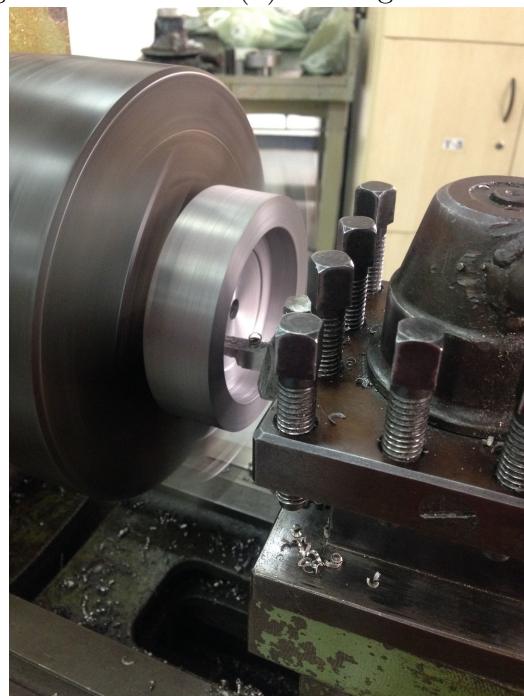
Fonte: Própria (2018).

O processo de usinagem, esmerilhamento e tratamento térmico foram aplicados em partes que fazem parte do ligamento da base, responsável pela sustentação do manipulador robótico. Figura 4.34 (a) especifica peças usinadas acoplada ao rolamento, o processo de usinagem dessa peça é apresentado na Figura 4.34 (c). A Figura 4.34 (b) é a montagem desse ligamento com o sistema mecânico de rotação da base, devido a dureza do material foi necessário técnica de tratamento térmico, recozimento, que consiste em submeter material a temperatura elevada por um determinado tempo e em seguida resfriamento lento e controlado (dentro de caixa de areia pré-aquecida), o que resultou na possibilidade de tornear, serrar e esmerilhar partes do ligamento da base.

Figura 4.34 - Ligamento responsável pela sustentação do manipulador robótico acadêmico.



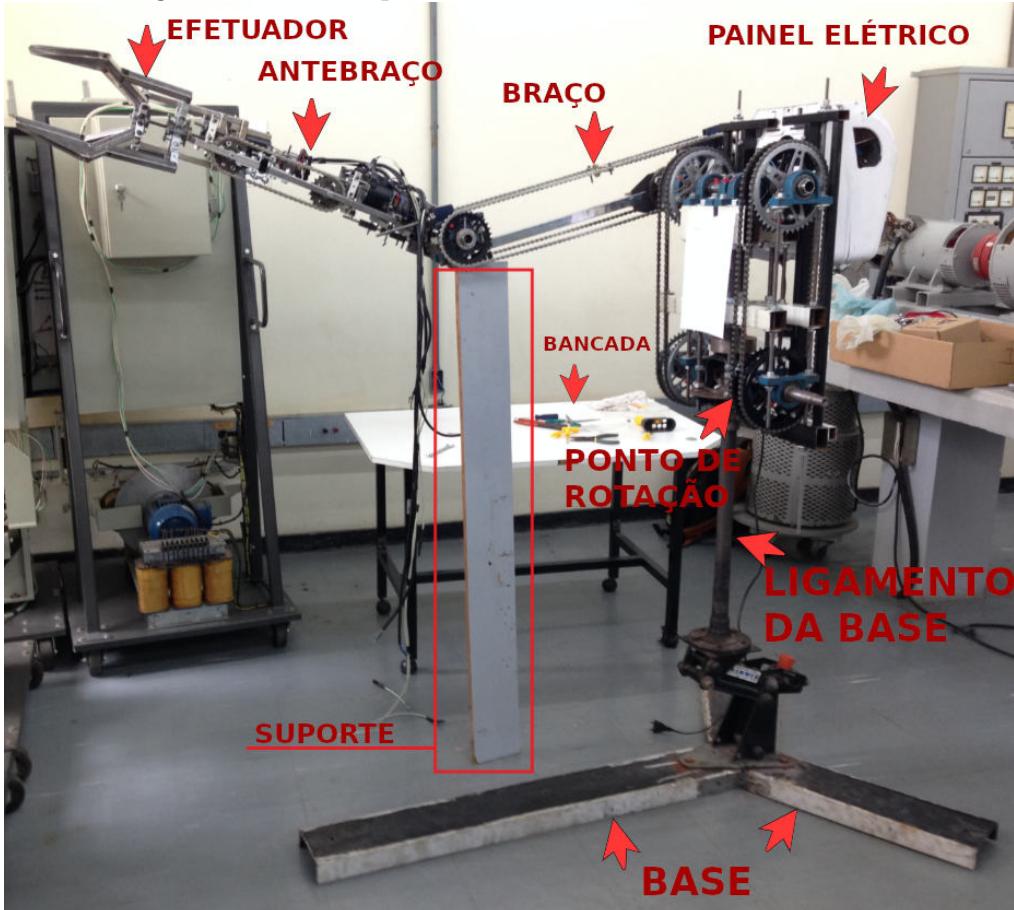
(a) Vista superior do ligamento da base. (b) Montagem do sistema de rotação da base.



(c) Usinagem de peça utilizada com o rolamento.
Fonte: Própria (2018).

O manipulador robótico construído foi montado conforme apresentado na Figura 4.35. O suporte foi usado como solução temporária para garantir estabilidade do robô. O ponto de rotação é responsável pelo movimento uniforme do manipulador robótico. O painel elétrico consiste de estrutura de proteção ao circuito de potência e controle dos motores.

Figura 4.35 - Manipulador robótico acadêmico montado.

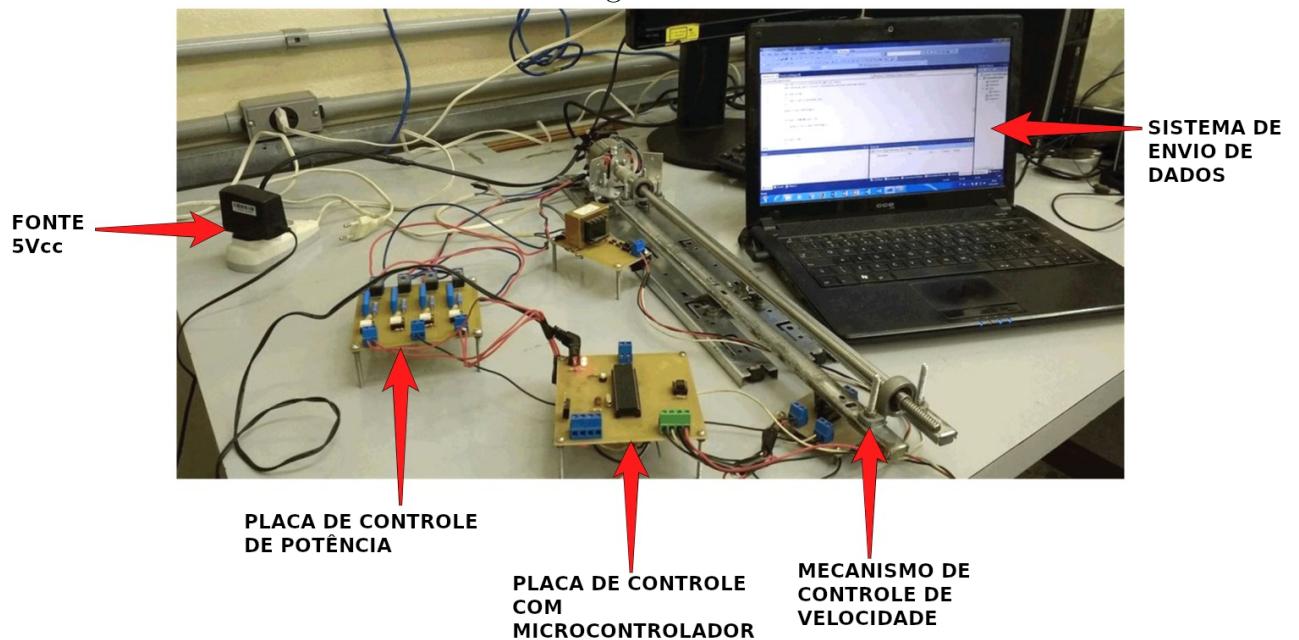


Fonte: Própria (2018).

4.6 TESTES DE FUNCIONAMENTO

Para a adaptação do motor universal a inversão do sentido de giro, os dois fios que ligam a armadura ao campo são separados, quando ocorre inversão dos pólos da armadura, é modificado a combinação original do sentido original para o sentido contrário. Esta combinação é feita pelos 4 TRIACs fixados na placa de controle de potência e inversão de sentido de giro. A lógica combinatória de inversão é passada pelo microcontrolador às entradas de controle da placa do circuito. Na Figura 4.36 é notado a estrutura metálica para fixação do motor universal, e em seu eixo é fixado uma rosca triangular que gira fixada em rolamento e mancais.

Figura 4.36 - Plataforma mecânica de testes preliminares de controle de potência, velocidade e inversão do sentido de giro do motor.

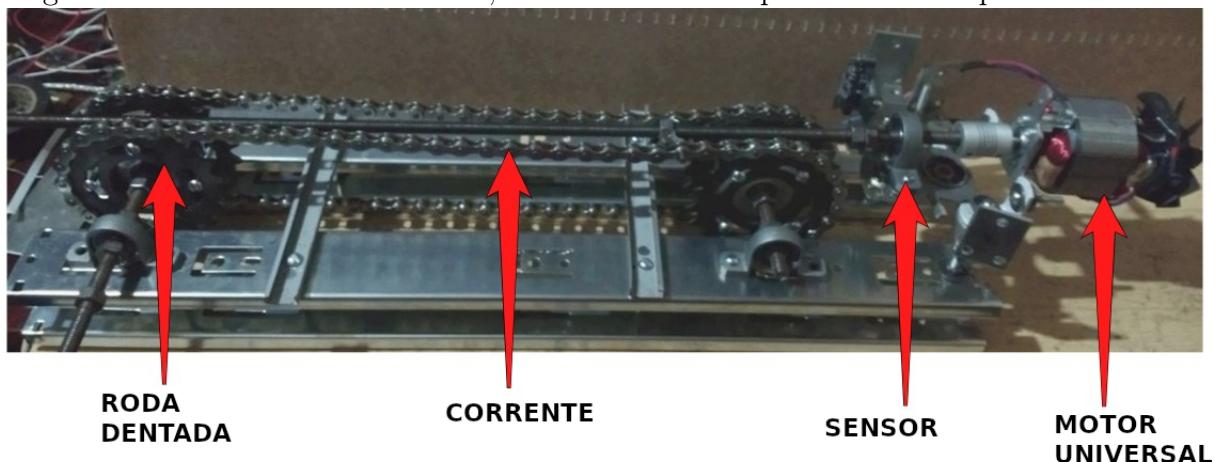


Fonte: Própria (2018).

A plataforma mecânica desenvolvida, Figura 4.37, possui roda dentada e corrente. A rosca triangular acoplada no eixo do motor possui porca de bronze TM23 fixada na corrente, ao ser rotacionada gera força axial movimentando a corrente de acordo com o sentido de rotação do motor, e assim produzindo movimento rotacional na roda dentada em torno de seu eixo no sentido horário e anti-horário.

O comprimento da região de trabalho do movimento da rosca engloba todo perímetro relacionado aos 360° da roda dentada. Para controlar o posicionamento foi colocado um potenciômetro linear e um disco *encoder* de impressora com 1200 pontos em 360° no eixo da roda. O sensor óptico fixado no disco de *encoder* envia pulsos ao microcontrolador no qual é feita uma contagem.

Figura 4.37 - Plataforma mecânica, destinada a testes preliminares de posicionamento.



Fonte: Própria (2018).

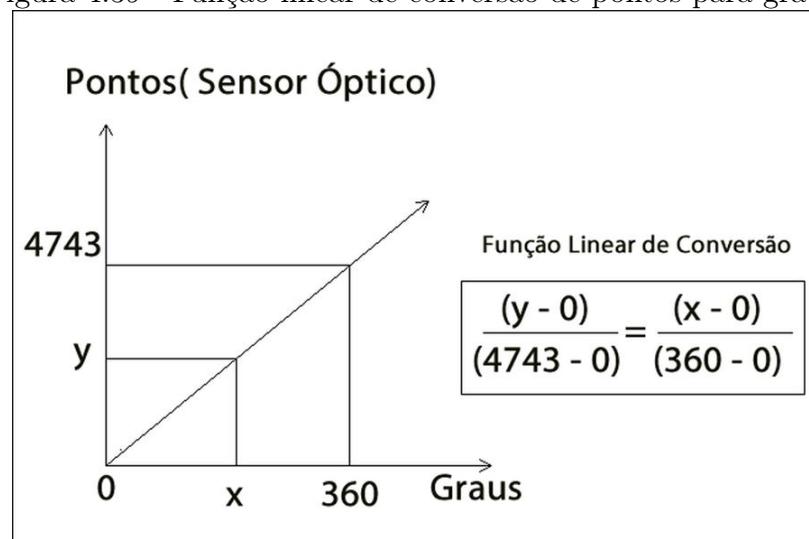
Com a utilização de dois sensores ópticos é possível verificar o estado atual do sentido de giro do motor, independente de comandos enviados do supervisório, Figura 4.38, logo o hardware é capaz de incrementar e decrementar pontos conforme o sentido adotado. Para melhor desempenho, o disco *encoder* possui 12 divisões, assim a precisão está relacionada ao furo da barra rosca dividida por 12 partes, e ao diâmetro da roda dentada. A conversão entre contagem de pontos para graus é caracterizado pela função linear descrita na Figura 4.39.

Figura 4.38 - Disco *encoder*, e sensores ópticos.



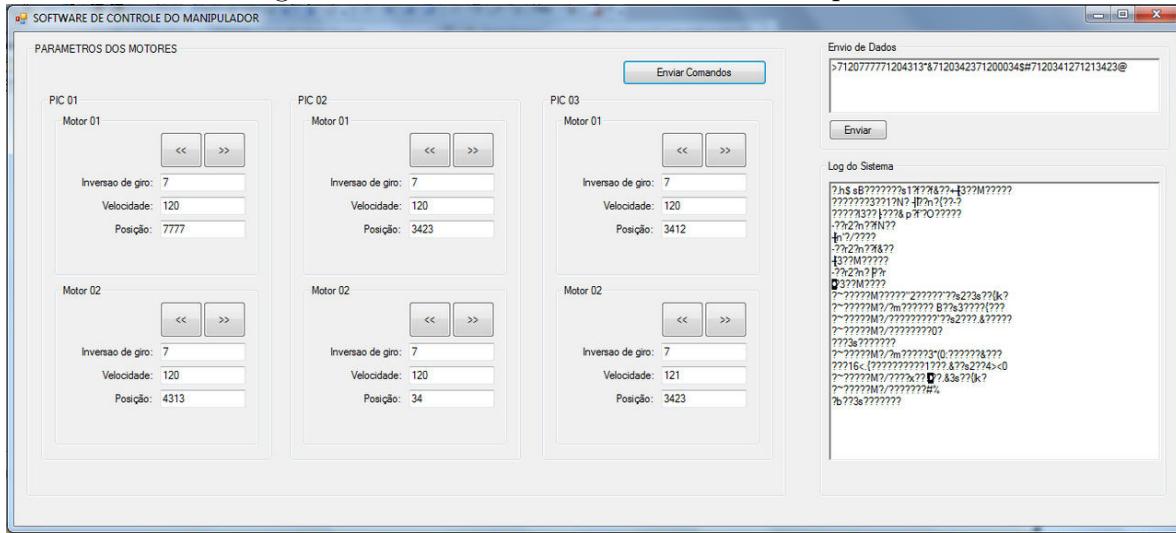
Fonte: Própria (2018).

Figura 4.39 - Função linear de conversão de pontos para graus.



Fonte: Própria (2018).

Figura 4.40 - Interface de controle do manipulador.



Fonte: Própria (2018).

Conforme mencionado anteriormente na seção 3.6, o quadro é dividido em três partes, cada um implementado por um microcontrolador. A diferença entre as partes é a identificação inicial e final presente no corpo do quadro de envio. O *software* faz o agrupamento de todos os dados, e exibe na tela o quadro para conferência e análise, conforme a Figura 4.40 do supervisório no campo “Envio de Dados”.

Dados de *log* enviados dos microcontroladores ao supervisório no mesmo cabo Tx/Rx, produz colisão de bytes produzindo caracteres aleatórios mostrados na parte designada a “Log de Sistema”. A solução mais simples e adequada foi produzir *logs* por PIC individualmente, uma vez que cada placa contém um conversor RS485.

A técnica de controle utilizada por meio do PID é fundamental para analisar e descrever o comportamento do manipulador robótico ao longo de suas trajetórias. O sensor óptico com o disco encoder presente no eixo de rotação da barra rosada envia sinais ao microcontrolador, estes dados interpretados pelo algoritmo, permite então controlar o fluxo de potência entregue ao motor mantendo uma rotação constante, independente da demanda de torque (controle PID).

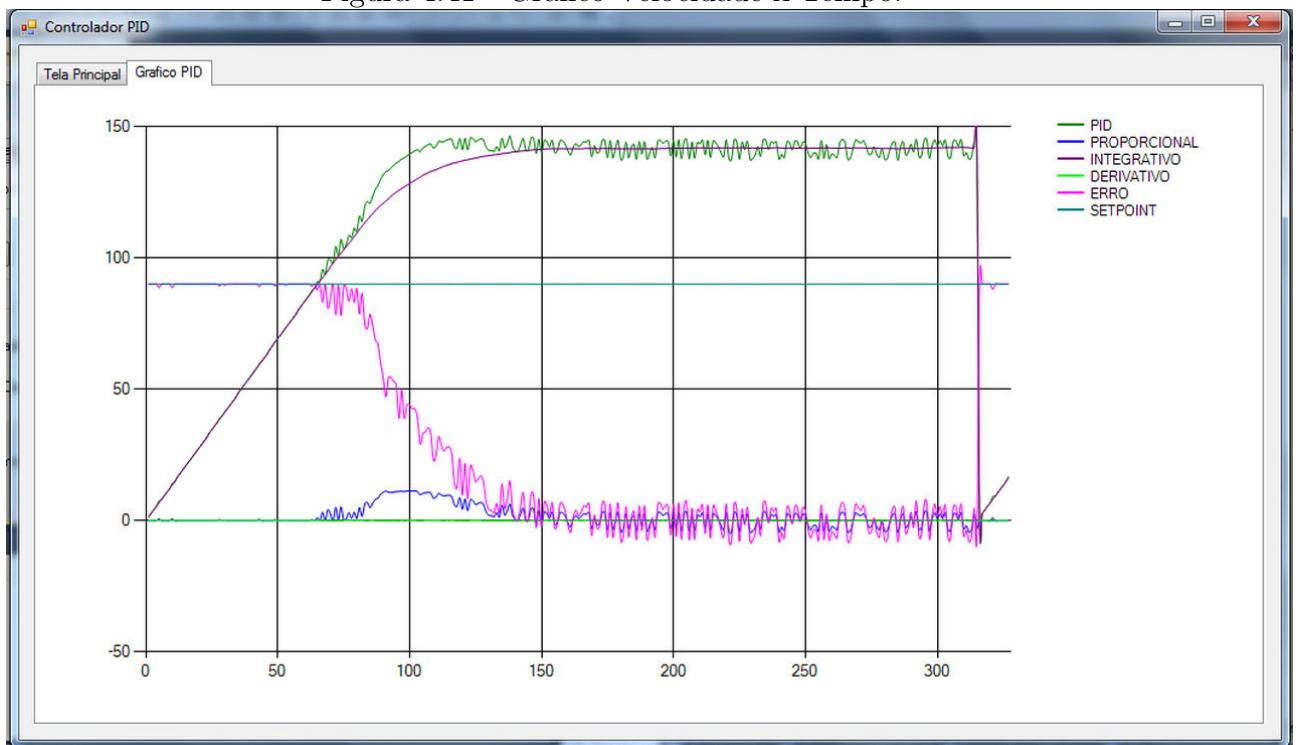
O supervisório envia um *SetPoint* que representa o período em *ms* de determinada rotação do eixo. Se o motor estiver sem demanda de torque o PID age normalmente; mas se o eixo do motor for exposto a um torque constante, para que seja mantido o período de rotação, o algoritmo age diretamente no *range* de controle de potência produzida, contrabalanceando entre torque e velocidade até que o *SetPoint* seja alcançado, ou seja o erro tender a zero.

Por meio dessa técnica abre-se possibilidades de análise de sensibilidade, como por exem-

plo: o sistema de controle do manipulador por meio do comportamento dos dados tem a capacidade de deduzir se está pegando um objeto e tomar uma decisão.

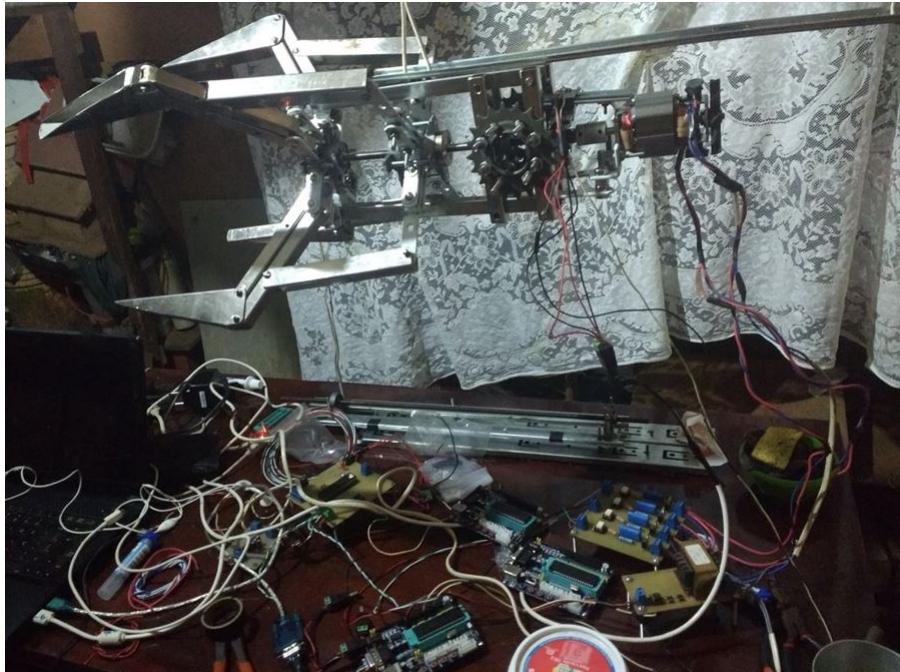
O gráfico PID gerado em tempo real pelo *software* a cada 150 ms, conforme apresentado na Figura 4.41, mostra a interação entre o proporcional, integral, derivativo, erro, PID e a correção do erro, estabilizando a velocidade de rotação do eixo. Os primeiros testes relacionados a garra do manipulador, (Fig. 4.42), evidenciam eficiência do controlador, gerando partida suave até que a demanda de contrabalanceamento de torque e velocidade seja atendida, e assim produzindo o movimento desejado.

Figura 4.41 - Gráfico Velocidade x Tempo.



Fonte: Própria (2018).

Figura 4.42 - Testes preliminares com a garra.



Fonte: Própria (2018).

4.6.1 MOVIMENTO DO MANIPULADOR ROBÓTICO ACADÊMICO

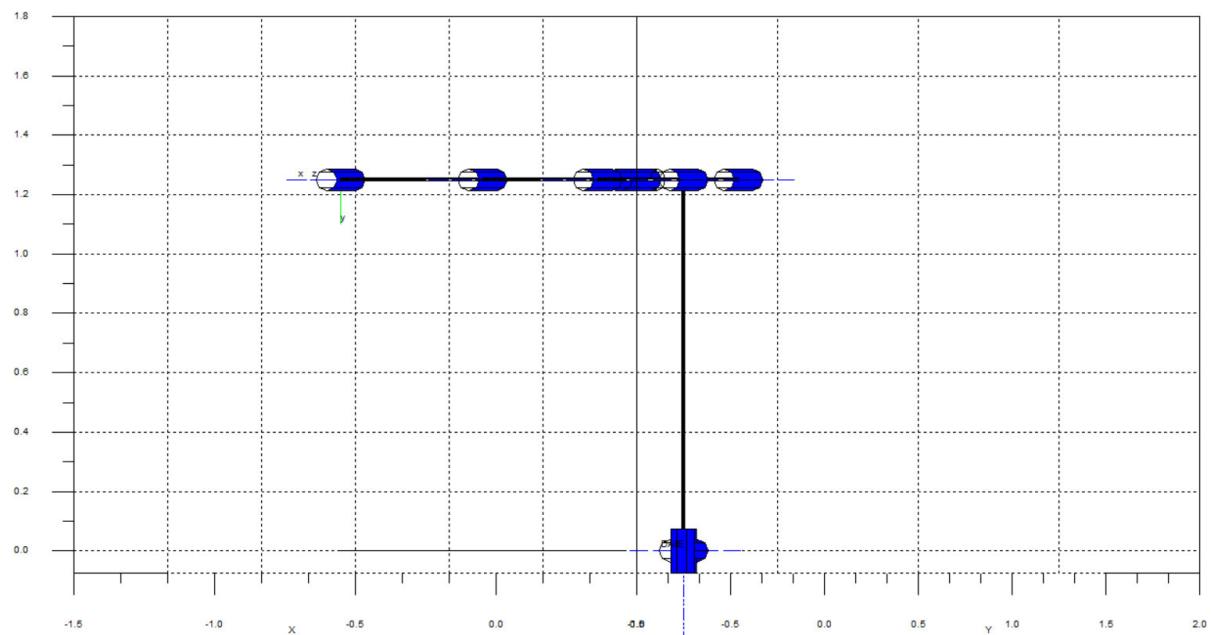
As Equações 4.1 e 4.2 são a configuração dos ângulos das juntas das posições mostradas na Figura 4.43 (a) e (b), respectivamente. As juntas que ocupam posição quinta e oitava do vetor de ângulos, representam no robô construído a junta do antebraço e do efetuador. Por comando do supervisório, os valores foram testados, assim como aplicação de segurar determinado objeto com a garra.

Depois de acoplado as partes do manipulador robótico foi identificado as seguintes necessidades: contrapeso para equilibrar o eixo do ligamento principal e garantir estabilidade nos movimentos; ocorre flambagem, devido utilização de porcas e parafusos (perdem fixação devido vibração mecânica) e algumas das superfícies dos materiais serem irregulares; peso excessivo dos materiais de aço inoxidável.

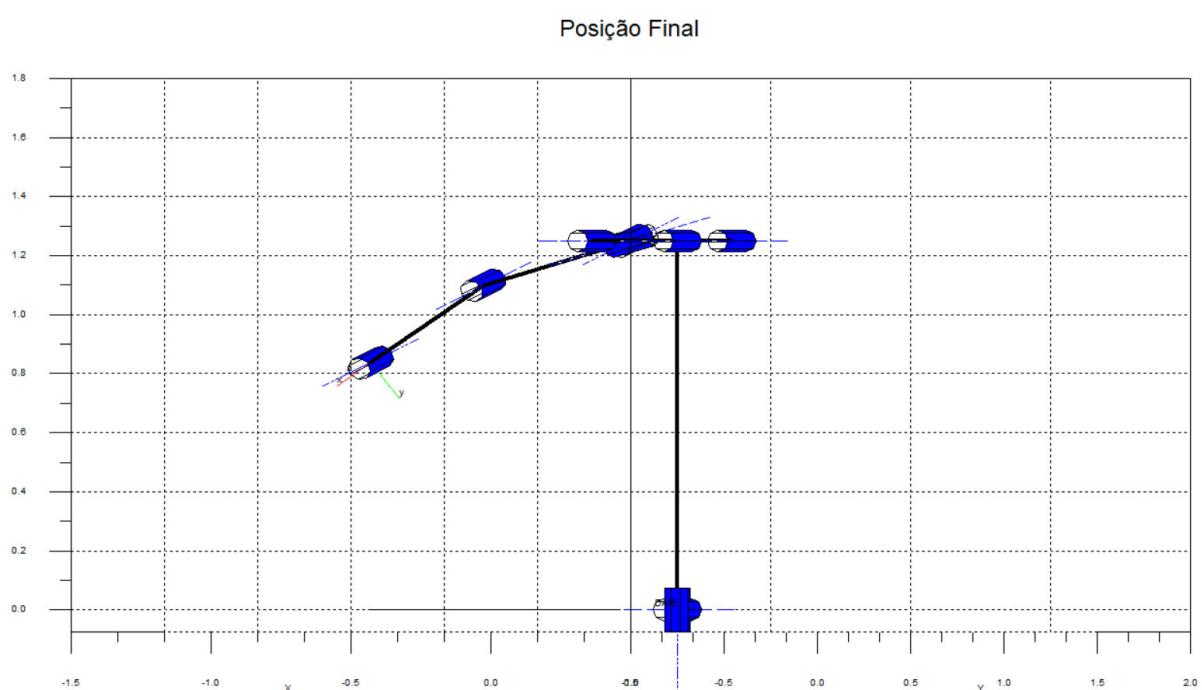
$$vetor_angulo_inicial = [0. \ 90. \ 90. \ 0. \ 90. \ 0. \ -90. \ 0. \ 0.] \quad (4.1)$$

$$vetor_angulo_final = [0. \ 90. \ 90. \ 0. \ 106. \ 16. \ -90. \ 16. \ 0.] \quad (4.2)$$

Figura 4.43 - Geração de trajetória, vista lateral.
Posição Inicial



(a) Posição inicial.



(b) Posição final.
Fonte: Própria (2018).

CAPÍTULO 5

Conclusão

A realização desse trabalho permitiu chegar às seguintes conclusões:

- Os testes realizados com o supervisório e o manipulador robótico construído obedeceu à lei de controle implementada no microcontrolador, demonstrou o funcionamento do protocolo de comunicação ao enviar determinado ângulo para uma das juntas do robô;
- O algoritmo de cinemática do manipulador robótico permitiu explorar trajetórias, ao gerar o vetor dos ângulos que é usado pelo supervisório (ângulo das juntas do manipulador);
- O orçamento limitado justifica a aplicação de materiais reciclados, ou que foram originalmente feitos para outras aplicações, algumas dessas peças são fonte de incerteza para a estabilidade do manipulador.
- A realização desse trabalho foi possível com a utilização dos seguintes equipamentos (principais): gravador de PIC K150, multímetro digital, ferro de solda. A construção e montagem foi feita no laboratório de usinagem do IFG Câmpus Goiânia. As principais máquinas para usinagem foram: o torno mecânico, furadeira, moto esmeril e serra. Foram realizados os métodos de tratamento térmico (cozimento) e soldagem.
- O erro dos equipamentos (análogicos) de medida, adicionados aos erros manuais (marcação no material a ser usinado), ocasionou em atraso na montagem do robô, em vista da necessidade de repetir os procedimentos necessários para correção dos erros citados.
- A escassez de trabalhos anteriores que utilizam os *softwares* e bibliotecas que esse trabalho foi baseado, no estudo da cinemática, constitui uma das dificuldades encontradas durante o desenvolvimento.
- Aprendizado desenvolvido: desenvolvimento de algoritmos vistos nas disciplinas do curso em aplicação de robótica; integração *hardware* e software; controle de motores simultaneamente; controle do ângulo de disparo da rede elétrica; controle PID em motores de corrente alternada; desenvolvimento de *driver* de controle; aperfeiçoamento do conhecimento de eletrônica digital e analógica; prática de soldagem e usinagem.

Os testes realizados foram: comunicação serial entre o supervisório e o Proteus® (versão de demonstração); controle da velocidade e sentido de rotação de giro dos motores; atualização e aperfeiçoamento do projeto mecânico e dos circuitos eletrônicos. Durante o desenvolvimento do TCC2, foi efetuado o acoplamento mecânico das placas descritas ao longo deste trabalho contendo os microcontroladores, responsáveis pela comunicação, controle de velocidade, potência e posicionamento.

Os testes realizados demonstram sucesso ao enviar comandos através do protocolo de comunicação desenvolvido; segurança foi considerada na implementação do sistema de movimentação do manipulador robótico, sendo que todas as placas de circuitos são embutidas de forma fixa em equipamentos desenvolvidos para este propósito específico. A divisão de trajetórias de movimento é implementado voltado pra segurança das pessoas, evitando assim movimentos bruscos sem que haja supervisão.

Sugestões para trabalhos futuros:

- Utilizar fuso de esfera recirculante com pré-carga para converter esforço longitudinal em rotacional, em contrapartida ao sistema de barra rosada que apresenta maior erro;
- Desenvolver projeto similar, com mínimo de peças e submontagens;
- Planejamento orçamentário, o intuito é agregar maior qualidade na construção do manipulador robótico;
- Desenvolver protótipo em escala reduzida para diminuir custo da construção, menor tempo de montagem e permitir que testes sejam realizados para então ampliar as medidas do robô;
- Ampliar pesquisa acadêmica sobre a utilização de motor AC - motor universal - na robótica, especificamente para manipuladores robóticos.

REFERÊNCIAS BIBLIOGRÁFICAS

- AHMED, A. *Eletrônica de potência*. São Paulo: Prentice Hall, 2000. 27
- CLARK, N. *C#: Programming Basics for Absolute Beginners*. [S.l.]: CreateSpace Independent Publishing Platform, 2016. 21, 22
- CORKE, P. I. 2018. Disponível em: <<http://rtss.sourceforge.net/>>. Acesso em: 9 jul. 2018. 35
- _____. **Função Scilab rt_link**. 2018. Disponível em: <http://rtss.sourceforge.net/docs/scifuncs/rt_link.htm>. Acesso em: 9 jul. 2018. 36
- _____. **Função Scilab rt_plot**. 2018. Disponível em: <http://rtss.sourceforge.net/docs/scifuncs/rt_plot-robot.htm>. Acesso em: 9 jul. 2018. 60
- CRAIG, J. J. *Introduction to Robotics: Mechanics and Control*. [S.l.]: Pearson Prentice Hall, 2005. 6, 7, 9, 10, 13
- DENAVIT, J.; HARTENBERG, S. *A kinematic notation for lower-pair mechanisms based on matrices*. [S.l.]: ASME J. on Applied Mechanics, 1955. 11
- ERICKSON, R. W.; MAKISIMOVIC, D. *Fundamentals of Power Electronics*. 2. ed. [S.l.]: Kluwer Academic Publishers, 2004. 14
- FU, K. S.; GONZALEZ, R. C.; LEE, C. S. G. *Robotics: Control, sensing, vision and intelligence*. [S.l.]: McGraw-Hill, 1987. 4, 5, 7, 8
- GURU, B. S.; HIZIROGLU, H. R. *Electric Machinery and Transformers*. 3. ed. [S.l.]: Oxford University Press, 2001. 14, 15
- GUSTAVO, L. **O que é RS232 RS485 RS422**. 2010. Disponível em: <<http://komoissofunciona.blogspot.com.br/2010/08/o-que-e-rs232-rs485-rs422.html>>. Acesso em: 25 fev. 2018. 43
- HART, D. W. *Power Electronics*. [S.l.]: McGraw-Hill, 2011. 17, 18
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 8373: Robots and robotic devices - Vocabulary*. [S.l.: s.n.], 2012. 4
- KINGSLEY, C.; FITZGERALD, A. E.; UMANS, S. D. *Electric Machinery*. 7. ed. [S.l.]: McGraw-Hill, 2014. 14, 15, 16, 17, 18
- KUCUK, S.; BINGUL, Z. *Industrial Robotics: Theory, Modelling and Control*. 1. ed. Germany: Pro Literatur Verlag, 2006. 6
- LABORATÓRIO DE GARAGEM. **Tutorial: Controlando a luminosidade de uma lâmpada incandescente com Dimmer Shield**. 2014. Disponível em: <<http://labdegaragem.com/profiles/blogs/controle-de-luminosidade-com-dimmer-shield>>. Acesso em: 25 fev. 2018. 52
- LIBERTY, J.; XIE, D. *Programming C# 3.0*. 5. ed. [S.l.]: O'Reilly, 2008. 21

LIMA, E. R. E. **C# e .NET - Guia do Desenvolvedor**. Rio de Janeiro: Editora Campus, 2002. [21](#), [22](#)

NAKOV, S. *Fundamentals of Computer Programming With C#*. [S.l.]: Svetlin Nakov Co, 2013. [21](#)

O'DWYER, A. **Handbook of PI and PID Controller Tuning Rules**. 3. ed. 57 Shelton Street, Covent Garden, London, WC2H 9HE: Imperial College Press, 2009. [19](#)

RIVIN, E. *Mechanical Design of Robots*. 1 ed. ed. New York.: Editora: McGraw-Hill Inc., 1988. [3](#)

ROMANO, V. F.; DUTRA, M. S. **CAPÍTULO 1: Introdução à Robótica Industrial**. In: Vitor Ferreira Romano (Editor). *Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos*. [S.l.]: Editora Edgar Blucher LTDA, 2002. [3](#)

ROSARIO, J. M. **CAPÍTULO 2: Modelagem e Controle de Robôs**. In: Vitor Ferreira Romano (Editor). *Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos*. [S.l.]: Editora Edgar Blucher LTDA, página 10., 2002. [4](#), [11](#)

ROSEN, C. A. "*Robots and Machine Intelligence*". In: Nof, S. Y. (ed), *Handbook of Industrial Robotics*. 1. ed. New York: John Wiley Sons, 1985. [4](#)

SCILAB.ORG. 2018. Disponível em: <<https://www.scilab.org/>>. Acesso em: 9 jul. 2018. [35](#)

SOSKA, G. *Third Generation Robots: Their Definition, Characteristics, and Applications. Robotics Age* 7. [S.l.: s.n.], 1985. [3](#)

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot Modeling and Control*. 1. ed. [S.l.]: John Wiley Sons, Inc, 2006. [11](#), [12](#), [14](#)

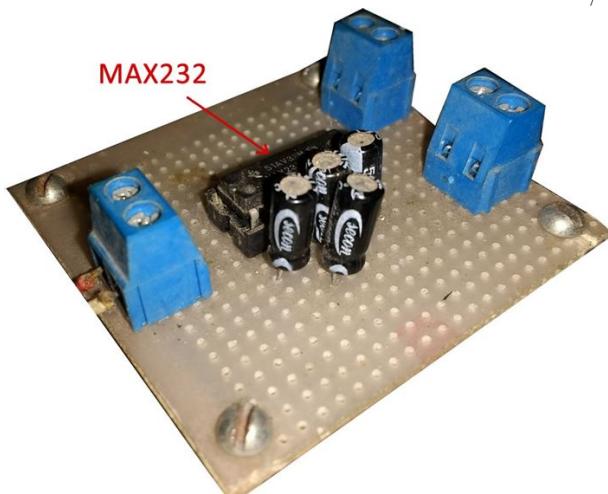
ÅSTRÖM, K. J.; HÄGGLUND, T. **PID Controllers: Theory, Design, and Tuning**. 2. ed. 67 Alexander Drive, Research Triangle Park, NC 27709: Instrument Society of America, 1995. [19](#), [20](#)

APÊNDICE 1

Placa Conversora RS232/TTL

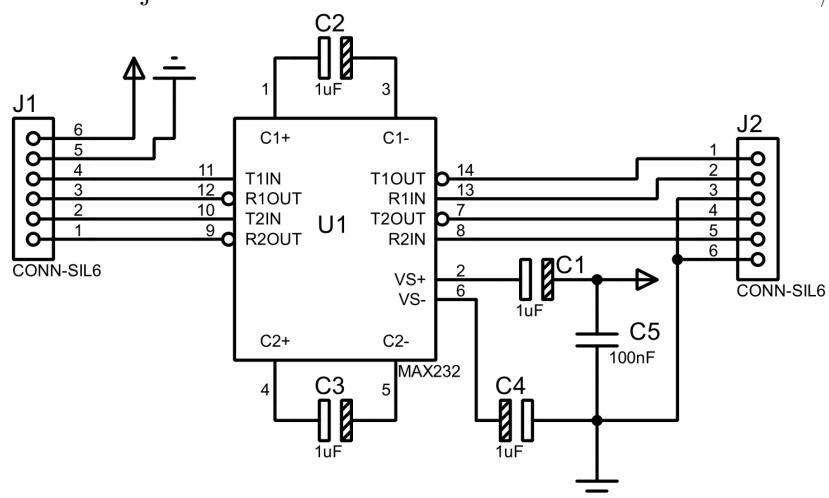
A placa conversora RS232/TTL, apresentado na Figura 1.1, tem a função de converter tensões provenientes de computadores (RS232) para sinais de microcontroladores com amplitude de 0 a 5v (TTL). Para conversão utiliza-se um CI MAX232 e capacitores. Projeto eletroeletrônico apresentado na Figura 1.2.

Figura 1.1 - Placa de conversão de sinais RS232/TTL.



Fonte: Própria (2018).

Figura 1.2 - Projeto eletroeletrônico de conversão de sinais RS232/TTL.



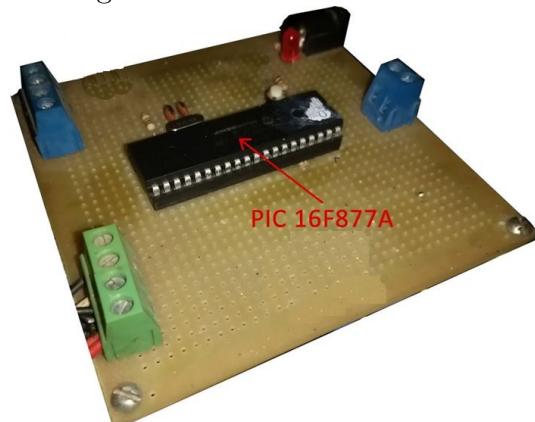
Fonte: Própria (2018).

APÊNDICE 2

Placa de Controle com Microcontrolador

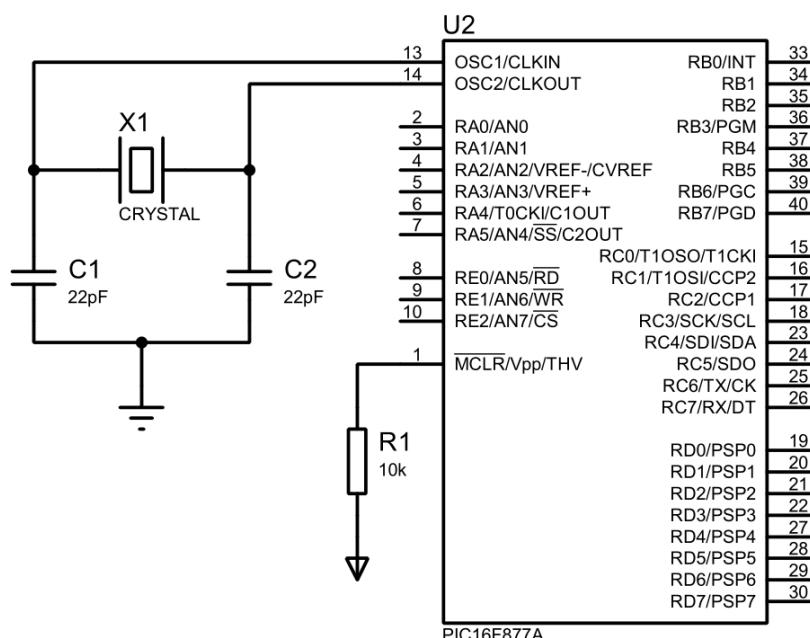
A placa de controle com microcontrolador, é utilizada para aplicação de comandos de controle simultâneos nas placas de potência, Figura 2.1. A placa é constituída de um microcontrolador PIC16F877A. Projeto eletroeletrônico apresentado na Figura 2.2.

Figura 2.1 - Placa de controle.



Fonte: Própria (2018).

Figura 2.2 - Projeto eletroeletrônico da placa de controle.



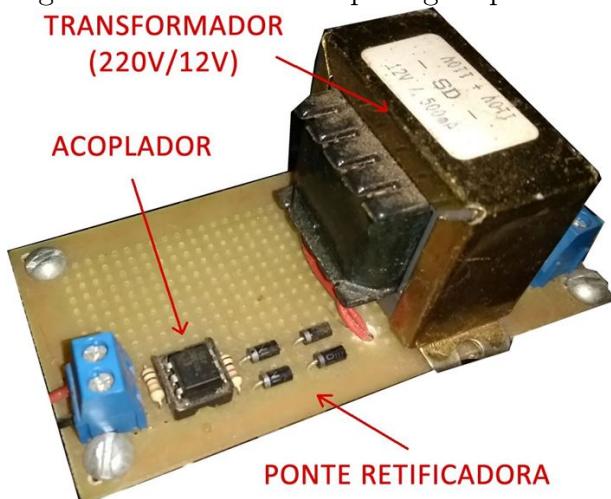
Fonte: Própria (2018).

APÊNDICE 3

Placa de Detecção de Passagem por Zero

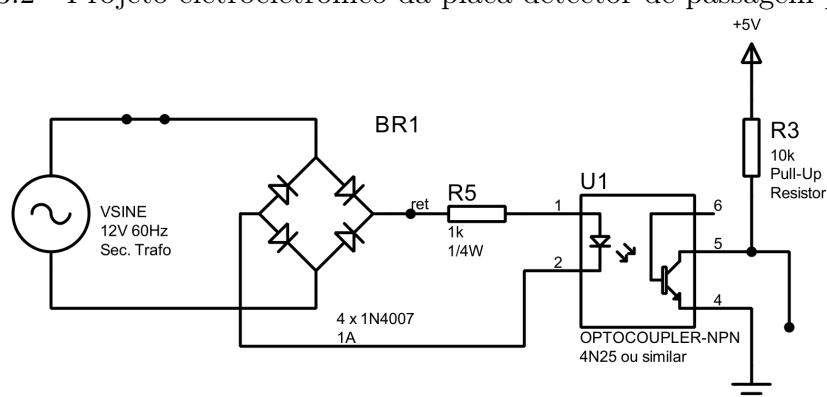
A placa detector de passagem por zero, Figura 3.1, é o circuito responsável por gerar pulsos a cada vez que a tensão da rede elétrica passa por zero. Este circuito é constituído por um transformador (220/12) VCA, uma ponte retificadora com 4 diodos e um acoplador óptico para isolar o sinal e enviar ao microcontrolador. Projeto eletroeletrônico apresentado na Figura 3.2.

Figura 3.1 - Detector de passagem por zero.



Fonte: Própria (2018).

Figura 3.2 - Projeto eletroeletrônico da placa detector de passagem por zero.



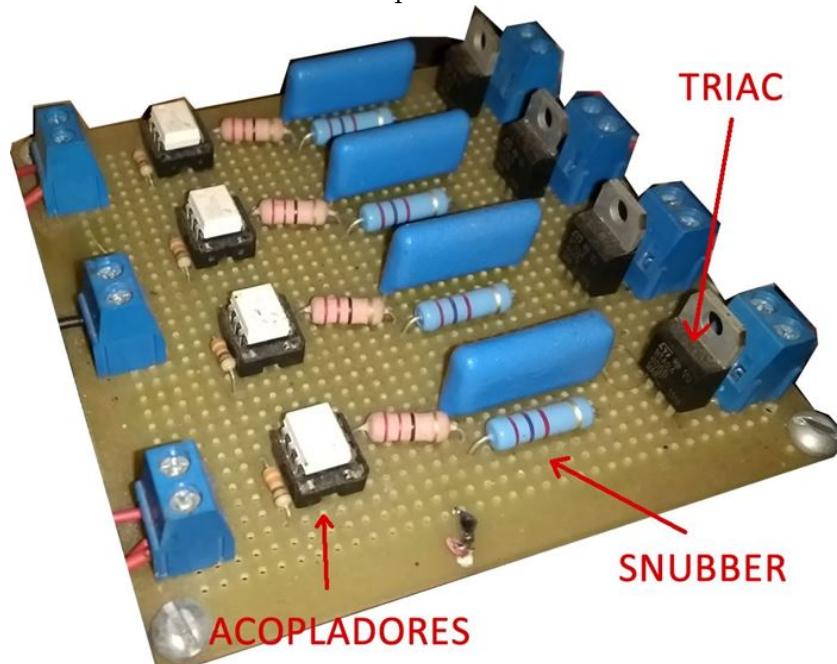
Fonte: Própria (2018).

APÊNDICE 4

Placa de Controle de Potência com Triacs

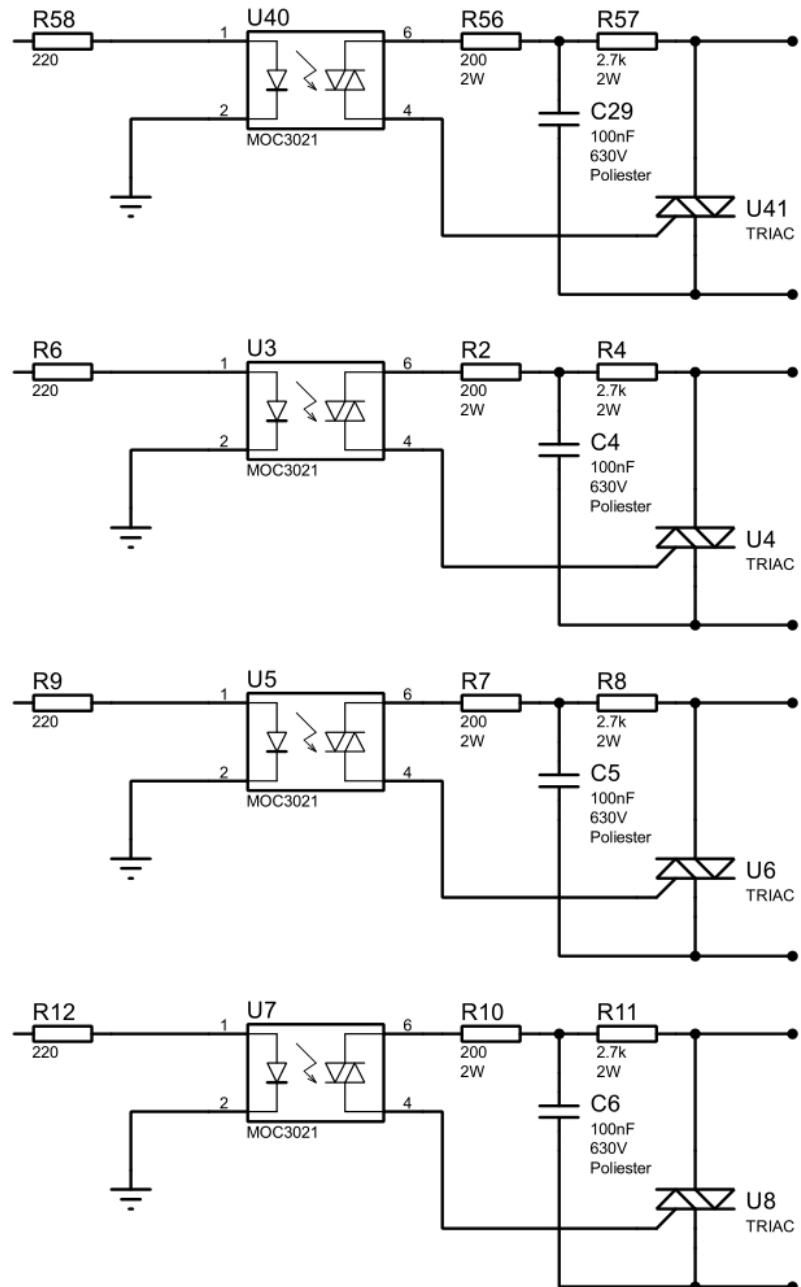
A placa de controle de potência com TRIACs, Figura 4.1, tem a função de inverter o sentido de giro dos motores por meio de quatro chaveamentos controlados (Ponte H), e promover fluxo de tensão, produzindo potência. A placa é constituída de acopladores ópticos para isolar o sinal de controle, um circuito de proteção de surtos de tensão (*Snubber*), e TRIACs que são tiristores de chaveamentos bidirecionais. Projeto eletroeletrônico apresentado na Figura 4.2.

Figura 4.1 - Placa de controle de potência e inversão de sentido de giro.



Fonte: Própria (2018).

Figura 4.2 - Projeto eletroeletrônico da placa de controle de potência e inversão de sentido de giro.



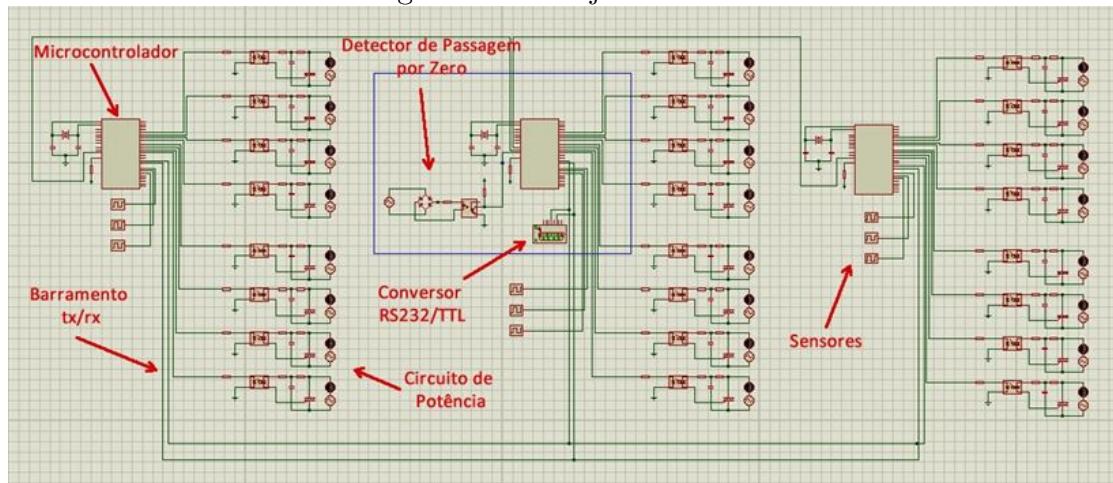
Fonte: Própria (2018).

APÊNDICE 5

Circuito Eletrônico Controle dos Motores

O projeto do circuito eletrônico de controle dos motores, Figura 5.1, proporciona visão geral da integração de todos os componentes, e o meio de comunicação entre eles. Os materiais incluem: microcontroladores; um barramento Tx/Rx; detector de passagem por zero; conversor RS232/TTL; circuito de potência e sensores de posicionamento.

Figura 5.1 - Projeto elétrico.



Fonte: Própria (2018).