FABIO JUNIOR JUBELLI
MARCELO DE LIMA
ROBSON LUIZ HORSTMANN

MANIPULADOR PARALELO EM CONFIGURAÇÃO DELTA

JOINVILLE, 2013

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS JOINVILLE
CURSO SUPERIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL

FABIO JUNIOR JUBELLI
MARCELO DE LIMA
ROBSON LUIZ HORSTMANN

MANIPULADOR PARALELO EM CONFIGURAÇÃO DELTA

Submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos de obtenção do título de Tecnólogo em Mecatrônica Industrial.

Orientador: Michael Klug, Msc.

JOINVILLE, 2013

Jubelli, Fabio J., Lima, Marcelo de., Horstmann, Robson L.
Manipulador Paralelo em Configuração Delta. Jubelli, Fabio J., Lima, Marcelo de., Horstmann, Robson L. – Joinville: Instituto Federal de Santa Catarina, 2013. 99 f.

Trabalho de Conclusão de Curso - Instituto Federal de Santa Catarina, 2013. Graduação. Curso Superior de Tecnologia em Mecatrônica Industrial. Modalidade: Presencial.
Orientador: Michael Klug, Msc.

1. Robô Delta 2. Cinemática Inversa 3. Microcontrolador 4. Servomotor 5. Pick-and-place
I.Título

MANIPULADOR PARALELO EM CONFIGURAÇÃO DELTA

FABIO JUNIOR JUBELLI
MARCELO DE LIMA
ROBSON LUIZ HORSTMANN

Este trabalho foi julgado adequado para obtenção do título de Tecnólogo em Mecatrônica Industrial e aprovado na sua forma final pela banca examinadora do Curso Mecatrônica Industrial do Instituto Federal de Educação, ciência e Tecnologia de Santa Catarina.

Joinville, 23 de Julho de 2013.

Banca Examinadora:

_____
Prof. Michael Klug, Mestre
Orientador

_____
Prof. Luiz Sérgio Barros Marques, Doutor
Avaliador

_____
Prof. Cláudio José Weber, Mestre
Avaliador

Não apenas este trabalho, mas todas nossas conquistas pessoais e profissionais são dedicadas a Deus e a tudo o que ele representa, às nossas famílias, professores e amigos.

AGRADECIMENTOS

Agradecemos em primeiro lugar a Deus por todos os acontecimentos em nossas vidas, pelas dificuldades que tivemos durante o desenvolvimento deste trabalho e que nos ensinaram que por mais difícil que seja uma situação, nunca se deve desistir, devemos sempre acreditar e ir atrás de nossos objetivos.

Aos nossos pais pelo amor, dedicação, educação e confiança, sentimentos que fizeram com que lutássemos com determinação.

Às nossas esposas e companheiras por todo o apoio e ajuda que obtivemos nos momentos de dificuldades.

Ao professor, orientador e amigo Michael Klug pelo apoio e compreensão pessoal neste trabalho.

A todos os professores, colegas e amigos que de alguma forma participaram de mais esta etapa que cumprimos de nossas vidas.

A todos fica aqui registrado o nosso muito OBRIGADO.

RESUMO

Este trabalho apresenta a construção do protótipo de um robô paralelo, em configuração Delta. Ao longo do trabalho foram projetados, implementados e testados componentes estruturais e mecânicos, circuitos eletrônicos e programas para controlar o robô. O processo de funcionamento do robô é tratado como uma resposta a um evento. A identificação de imagens é a entrada no qual um sinal é enviado diretamente a um computador. O controle é a etapa intermediária interpreta que e processa o evento, definindo uma ação a ser tomada. Por fim, representada pelo acionamento, a saída corresponde à resposta ao evento de acordo com a ação definida. O Robô Delta é alimentado através de fontes externas e apresenta autonomia de controle, pois é comandado por um microcontrolador Arduino Uno. Para a movimentação dos braços utilizam-se três servomotores. Para a manipulação da garra utilizam-se outros dois, sendo que todos são controlados pelo Arduino. O processo de movimentação se dá por algoritmos da cinemática inversa do robô, auxiliado pelo reconhecimento de imagens através de uma webcam, onde os objetos são representados por imagens de figuras geométricas planas e são identificados através de visão computacional. Para a implementação dos algoritmos e do reconhecimento de imagens foi utilizado o ambiente de programação do *MatLab.* Como resultado final deste trabalho, obteve-se um Robô Delta com capacidade de efetuar operações de *pick-and-place* no espaço bidimensional (plano).

Palavras-Chave: Robô Delta. Cinemática inversa. Microcontrolador. Servomotor. *Pick-and-Place.*

ABSTRACT

This work is about the construction of a parallel robot prototype in Delta configuration. Over the work mechanical components, electronic circuits and program to control the robot have been designed, implemented and tested structural. The operation process of the robot is treated as answer to an event. The identification of images is the input, in which a signal is sent directly to a computer. The control is the intermediate step that interprets and processes the event, defining an action to be taken. Finally, represented by the drive, the output corresponds to the answer to the event according to the action set. The Delta robot is powered for external sources and has autonomy control, because it is controlled by an Arduino Uno microcontroller. For the movement of the arms three servomotors are used. For manipulation of the gripper another two are used, all of which are controlled by Arduino. The moving process is given by the inverse kinematics algorithms of the robot, aided by the recognition of images using a webcam, where the objects are represented by images of flat geometric figures and are identified by computer vision. For the implementation of the algorithms and image recognition, had been used MatLab programming environment. As the final result of this work, it has been obtained a Delta robot with the capacity to make basic pick-and-place in two-dimensional space (plane).

Keywords: Delta Robot. Inverse Kinematics. Microcontroller. Servomotor. Pick-and-Place.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS, ABREVIATURAS E SÍMBOLOS

Sigla/Símbolo	Significado Original	Tradução
3D	Tridimensional	---
A	Ampere	---
arctan	Arcotangente	---
AVI	Audio Video Interleave	Intercalação Áudio e Vídeo
BMP	Bitmap	Mapa de bits
CAD	Computer Aided Design	Desenho Auxiliado por Computador
CAE	Computer Aided Engineering	Engenharia Auxiliado por Computador
CLP	Controlador Lógico Programável	---
cos	Cosseno	---
DC	Direct Current	Corrente Contínua
DFM	Design for Manufacturing	Projeto para Fabricação
E	Juntas Esféricas	---
EEPROM	Electrically-Erasable Programmable Read-Only Memory	Memória Somente de Leitura Programável Apagável Eletricamente
EPFL	École Polytechnique Fédérale de Lausanne	Escola Politécnica Federal de Lausana
GDL	Grau de Liberdade	---
GUI	Guide User Interface	Interface de Guia do Usuário
I/O	Input/Output	Entrada/Saída
IFSC	Instituto Federal de Santa Catarina	---
IHM	Interface Home-Máquina	---
ISO	International Organization for Standardiation	Organização Internacional para Normatização
kB	kilobyte	---

kg	kilograma	---
kg/cm	kilograma por centímetro	---
L_{ante}	Comprimento do Ante	---
L_{base}	Comprimento do Base	---
$L_{braço}$	Comprimento do Braço	---
LED	Light Emitter Diode	Diodo Emissor de luz
L_{mancal}	Comprimento do Mancal	---
L_{mesa}	Comprimento do Mesa	---
m/s	Metros por segundo	---
m/s^2	Metros por segundo ao quadrado	---
m^2	Metro quadrado	---
M8	Rosca métrica com diâmetro de 8mm	---
mA	miliampere, unidade de corrente elétrica	---
Matlab	MATrix LABoratory	
MDF	Medium Density Fiberboard	Placa de Fibra de Média Dendidade
MHz	Megahert, unidade de frequência	---
mm	Milímetro	---
ms	Milisegundos	---
P_A	Ponto do antebraço	---
P_{A1}	Ponto do antebraço da 1ª cadeia cinemática	---
P_B	Ponto do braço	---
P_{B1}	Ponto do braço da 1ª cadeia cinemática	---
PC	Personal Computer	Computador Pessoal
P_E	Ponto de Referência dos eixos	---
P'_E	Ponto de Referência dos eixos defasado em 120º	---
P''_E	Ponto de Referência dos eixos defasado em 240º	---
P_M	Ponto do mancal	---

P_{M1}	Ponto do mancal da 1ª cadeia cinemática	---
P_O	Ponto de Origem do Robô	---
PPM	Pulse Proportional Modulation	Modulação Parcial de Pulsos
PWM	Pulse Width Modulation	Modulação por Largura de Pulso
R	Juntas Rotativas	---
r	Raio da circunferência	---
R$	Real, moeda Brasileira	---
RAW	Formato de Vídeo	Cru
RIA	Robotic Industries Association	Associação das Indústrias Robóticas
RUR	Robôs Universais de Rossum	---
RZ	Matriz de rotação	---
sen	Seno	---
SRAM	Static Random Access Memory	Memória Estática de Acesso Aleatório
USB	Universal Serial Bus	Barramento Serial Universal
V	Volt, unidade de tensão elétrica	---
Vca	Volts em corrente alternada	---
Vcc	Volts em corrente contínua	---
X_{A1}	Valor do eixo x para o antebraço da 1ª cadeia cinemática	---
X_{B1}	Valor do eixo x para o braço da 1ª cadeia cinemática	---
X_{centro}	Valor de x no centro da circunferência = XM1	---
X_E	Valor de Referência do eixo x	---

X'_E	Valor de Referência do eixo x defasado em 120º	---
X''_E	Valor de Referência do eixo x defasado em 240º	---
X_{M1}	Valor do eixo x para o mancal da 1ª cadeia cinemática	---
X_{ponto}	XB1	---
Y_{A1}	Valor do eixo y para o antebraço da 1ª cadeia cinemática	---
Y_{B1}	Valor do eixo y para o braço da 1ª cadeia cinemática	---
Y_{centro}	Valor de y no centro da circunferência = YM1	---
Y_E	Valor de Referência do eixo y	---
Y'_E	Valor de Referência do eixo y defasado em 120º	---
Y''_E	Valor de Referência do eixo y defasado em 240º	---
Y_{M1}	Valor do eixo y para o mancal da 1ª cadeia cinemática	---
Y_{ponto}	YB1	---
Y_Z	Plano Cartesiano entre os eixos Y e Z	---
Z_{A1}	Valor do eixo z para o antebraço da 1ª cadeia cinemática	---
Z_{B1}	Valor do eixo z para o braço da 1ª cadeia cinemática	---
Z_{centro}	Valor de z no centro da circunferência = ZM1	---
Z_E	Valor de Referência do eixo z	---

Z'_E	Valor de Referência do eixo z defasado em 120º	---
Z''_E	Valor de Referência do eixo z defasado em 240º	---
Z_{M1}	Valor do eixo z para o mancal da 1ª cadeia cinemática	---
Z_{ponto}	ZB1	---
θ	Ângulo das Juntas Ativas	---
θ_1	Ângulo da Juntas Ativa da 1ª cadeia cinemática	---
θ_2	Ângulo da Juntas Ativa da 2ª cadeia cinemática	---
θ_3	Ângulo da Juntas Ativa da 3ª cadeia cinemática	---
MP	Megapixels	---

1. INTRODUÇÃO

Sabe-se que a arquitetura predominante em robôs manipuladores ainda é a do tipo serial. Entretanto pode-se notar que nos últimos anos existe uma tendência em favor da arquitetura paralela (MELLO, 2011).

O fato da arquitetura paralela ser constituída por vários braços atuando simultaneamente sobre uma plataforma móvel, em vez de um só braço como na arquitetura serial, algumas vantagens são alcançadas tais como: elevadas acelerações e velocidades, maior rigidez, maior capacidade de levantamento de carga condensado uma massa menor (SILVA et al., 2003). Suas principais desvantagens referem-se ao espaço de trabalho (região que o robô pode alcançar através de seus movimentos) inferior e à maior complexidade mecânica de seus componentes (FINOTTI, 2008).

Os mecanismos em configuração paralela (GOUGH et al., 1962) começaram a ser desenvolvidos na década de 60, por Gough e Whitehall numa máquina de teste de pneus. STEWART (1965) também utilizou esse mecanismo como simulador de voos.

A estrutura Delta foi proposta na década de 80, por Reymond Clavel da EPFL (Escola Politécnica Federal de Lausana - do francês *École polytechnique fédérale de Lausanne)*, Suíça (LOPES, 2002). Esse mecanismo possui 3 graus de liberdade (número de parâmetros independentes que são necessários para se definir a posição de um corpo no espaço em qualquer instante) e opera com a função de *Pick-and-Place* (OLSSON, 2009). Para executar tal função, e como diferencial, será utilizado controle de reconhecimento de objetos por imagem através de uma webcam.

1.1. Objetivo Geral

Este trabalho tem por objetivo projetar um robô de arquitetura paralela em configuração Delta, bem como a modelagem, implementação e controle de um protótipo.

1.2. Objetivos Específicos

Os principais objetivos específicos deste trabalho foram escolhidos da seguinte forma:

- Executar o projeto e construção do robô;
- Estudo da cinemática inversa da configuração Delta;
- Realizar o controle através de reconhecimento de imagem;
- Conciliar a fundamentação teórica com a prática;
- Definir trajetórias de tarefas de *pick-and-place*.

1.3. Justificativa

Algumas justificativas principais são dadas para sustentar os objetivos escolhidos, que são:

- Em comparação com os robôs de arquitetura serial os paralelos possuem vantagens quanto à velocidade de operação, exatidão, rigidez e capacidade de carga, tornando-os ideais em várias aplicações;
- A utilização de robôs de uma forma geral cresceu muito nos últimos anos e ainda está crescendo. Esse tipo de robô é visto em muitos casos como a nova geração em matéria de conceito mecânico, dando uma maior flexibilidade à manufatura atual e resolvendo diversos problemas. Academicamente a importância de trabalhos com o foco nesses robôs também cresceu;
- Ainda existem muitas questões mecânicas, eletrônicas e de modelagem a serem trabalhadas nesse tipo de arquitetura. Particularmente na questão de otimização ainda são encontrados poucos trabalhos.

1.4. Divisão do Trabalho

Já foram vistos neste Capítulo, os objetivos e a justificativa do projeto.

O Capítulo 2 tratará de uma revisão bibliográfica empregada neste trabalho.

No Capítulo 3 será desenvolvida a análise cinemática inversa.

No Capítulo 4, será apresentada a modelagem e implementação do projeto.

No Capítulo 5 realizar-se-á o controle de movimento do robô, bem como serão definidos os códigos de programação.

Por último, no Capítulo 6 têm-se a conclusão e algumas sugestões para futuros trabalhos.

2. REVISÃO BIBLIOGRÁFICA

2.1. Descrição de um Robô

Existem inúmeras definições sobre o que é um robô. Algumas delas são:

Segundo a *Robotic Industries Association* (RIA), robô industrial é definido como um 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).

Uma definição mais completa é apresentada pela norma ISO 10218 (*International Organization for Standardization*), como sendo: "uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial" (AGOSTINI, 2012).

2.2. Histórico dos Robôs

2.2.1. O Surgimento dos Termos Robô e Robótica

O escritor tcheco de peças teatrais Karel Capek foi o primeiro a utilizar o termo “robô”, como é conhecido atualmente. Esse fato concretizou-se em sua peça R.U.R. “Robôs Universais de Rossum”, que teve sua estreia em janeiro de 1921, na cidade de Praga (GROOVER, 1986). É importante salientar que a palavra robô significa “trabalhador escravo” ou “servo” (PIRES, 2002). Com seu tema inovador e mostrando desumanização das pessoas em uma sociedade avançada tecnologicamente, a peça foi um sucesso e logo estreou na Europa, Estados Unidos, mais tarde sendo difundida em todo o mundo. Por mais incrível que pareça, Capek imaginava que os robôs não seriam criados através de meios mecânicos e sim químicos. Existem algumas

evidências também que a palavra, na verdade foi sugerida pelo seu irmão Josef, também escritor da época (CARRARA, 2008).

Devido aos seus influentes trabalhos como escritor, Karel Capek foi indicado ao prêmio Nobel. Apesar de possuir um potencial consideravelmente expressivo perante os demais escritores de sua época, seus trabalhos sempre serão um mistério, já que ele foi morto 1938, após ter sido capturado pela Gestapo (polícia secreta alemã) devido a sua postura antinazista (KAREL CAPEK WEBSITE, 2013).

O escritor americano (de origem russa) Isaac Asimov popularizou o conceito de robô como sendo uma máquina de aparência antropomórfica, e que não possuía sentimento algum. O comportamento de tal máquina seria definido a partir de programações introduzidas por seres humanos, a fim de tornar possível o cumprimento de determinadas regras ou tarefas (TARTARI, 2006). O termo robótica foi criado por Asimov para designar a ciência que se dedica ao estudo dos robôs, no qual foi utilizado em uma de suas histórias: *Runaround*, publicada em 1942. Esse conceito se fundamenta pela observação de três leis básicas, sendo que posteriormente foi acrescentada à lei número zero (SCHIAVICCO, SICILIANO, 1995):

- **Lei Zero:** Um robô nunca deve causar mal à humanidade ou, por não agir, permitir que seja causado algum mal à humanidade;
- **Lei Um:** Um robô nunca deve ferir um ser humano, ou, por não agir, permitir que seja causado algum mal a um ser humano, a menos que isso viole uma lei de ordem mais alta;
- **Lei Dois:** Um robô deve obedecer às ordens dadas a ele por seres humanos, exceto quando essas ordens entrem em conflito com uma lei de ordem mais alta;
- **Lei Três:** Um robô deve proteger sua própria existência enquanto tal proteção não entre em conflito com uma lei de ordem mais alta.

Toda a base tecnológica para os atuais robôs industriais foi desenvolvida a partir de intensas pesquisas que tiveram início logo após a Segunda Guerra Mundial (1939 a 1945), quando foi desenvolvido um equipamento denominado tele operador "*master-slave*" empregado em atividades de manipulação de materiais radioativos (SOARES, 2007).

O sistema era formado de um manipulador "*master*", movido diretamente por um operador humano responsável pelas sequências de movimentos desejados, e um manipulador "*slave*" capaz de reproduzir os movimentos realizados remotamente pelo "*master*". Os vínculos entre os manipuladores "*master*" e "*slave*" eram realizados através de sistemas de transmissão mecânicos (FU et al., 1987).

2.2.2. O Primeiro Robô Industrial

No ano de 1956, logo após a ascensão tecnológica decorrente da Segunda Guerra Mundial, aconteceu uma reunião que entrou para a história no campo que tange a robótica. Nesse dia reuniram-se o bem-sucedido inventor George C. Devol e o famoso engenheiro Joseph F. Engelberger. Nesse encontro, debateram sobre alguns trabalhos relacionados aos robôs de Asimov (AZEVEDO et al., 2009) e atrelaram forças para desenvolver um manipulador robótico real que tivesse a capacidade de executar inúmeros trabalhos. Após tal encontro, Engelberger fundou uma empresa de manufatura que se dedicava à automação, chamada Unimation. Joseph foi então intitulado o "pai da robótica" devido ao primeiro robô desenvolvido pela Unimation, o Unimate (Figura 1). Esse robô foi primeiramente instalado numa fábrica da General Motors (TARTARI, 2006).

Figura 1 - O Unimate - primeiro robô industrial.
Fonte: www.electronicsteacher.com. Acesso em 18/05/2013.

2.2.3. O Primeiro Robô Paralelo

Um dispositivo patenteado por GWINNETT (1931), datado em 1928, é, historicamente, relacionado como o primeiro registro de um mecanismo paralelo (BONEV e RYU, 1998). Esse mecanismo é uma plataforma móvel que foi projetada para indústria do *show business*, conforme visto na Figura 2, a fim de movimentar-se durante a exibição de filmes, simulando algumas cenas e instigando as emoções dos espectadores. Apesar de ser uma ideia totalmente inovadora, o equipamento não chegou a ser construído.

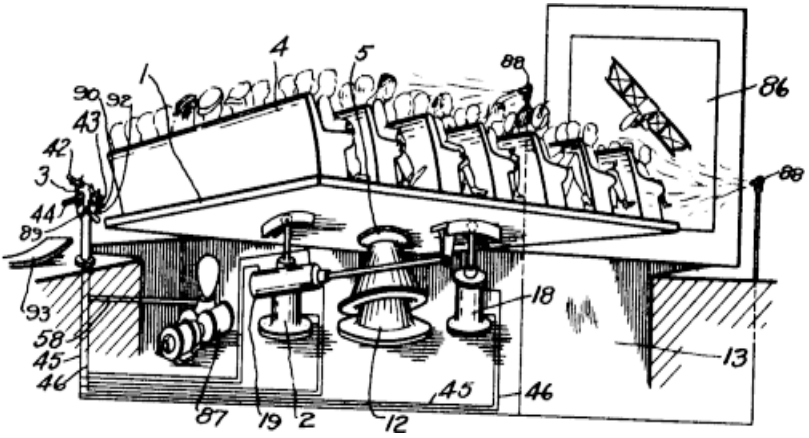


Figura 2 - Primeiro robô paralelo - Patenteado em 1931
Fonte: MELLO, 2011

Por volta de 1947, Gough desenvolveu um dispositivo que tinha o objetivo de testar o efeito da aterrissagem de aeronaves em pneus. Esse mecanismo era uma plataforma de seis “pernas” que permitia a aplicação de cargas não muito elevadas em direções distintas, imprimindo movimentos de rotação e translação, por meio do acionamento autônomo e individual de seus atuadores hidráulicos (MELLO, 2011). Embora essa plataforma seja, nos dias atuais, a arquitetura mais aceita para a

definição de manipuladores em configuração paralela, GOUGH (1962) menciona mecanismos hexápodes que já existiam antes do projeto de sua autoria. Tratam-se de estruturas com seis atuadores que se deslocam em trajetórias lineares (sendo três horizontais e três verticais). Essas estruturas, que eram capazes de promover deslocamentos consideráveis, porém não muito grandes, foram as precursoras de sua concepção.

Um dispositivo simétrico com cuja superfície é composta por oito faces (forma de um octaedro) foi a solução mais plausível para a elevada variedade de movimentos que Gough necessitava. Na Figura 3 pode-se visualizar o projeto de Gough, que apesar de ter sido construído em 1950, só foi totalmente finalizado, para que pudesse entrar em funcionamento, em 1954 (MELLO, 2011).

Figura 3 - Projeto original de Gough para testes de pneus de aeronaves (E) e versão moderna do aparato com atuadores elétricos(D).
Fonte: MELLO, 2011

Gough empenhou grandes esforços em seu projeto, porém apenas quando o primeiro simulador de voo foi arquitetado, a popularidade e utilização dos manipuladores de arquitetura paralela vieram à tona.

A indústria da aviação registrou um crescimento considerável nos anos 60. Assim, havia a necessidade de

empenhar grandes valores com o treinamento de pilotos. Esse fato, juntamente com a ânsia pela avaliação e execução de testes em novos dispositivos que não precisavam sair do lugar, ou seja, que não levantavam voo, instigaram vários pesquisadores com o estudo de sistemas com vários graus de liberdade capazes de possibilitar a simulação de uma plataforma similar à cabine de um avião, ou seja, que possuía uma dinâmica acelerada e dispunha da capacidade de suportar cargas elevadas (MELLO, 2011).

A figura 4 exibe um simulador de voo com seis graus de liberdade proposto por STEWART (1965). O pesquisador desenvolveu esse equipamento motivado pelos fatos decorrentes do crescimento na área aeronáutica e propôs profundas alterações se comparado à configuração alvitrada por GOUGH (1962).

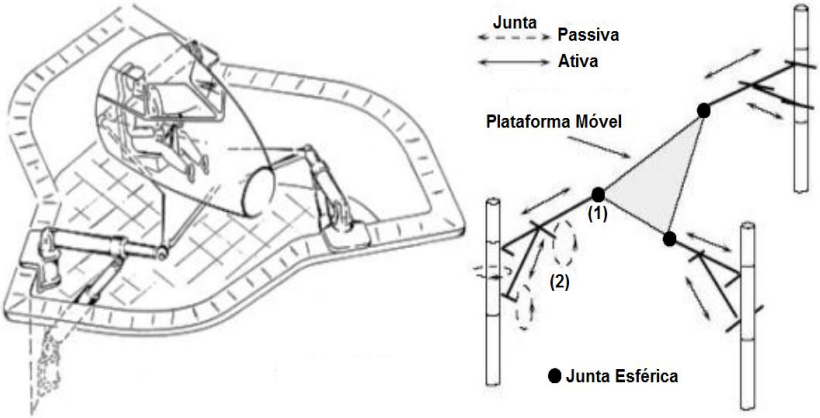


Figura 4 - Projeto de Stewart para simuladores de Voo.
Fonte: MERLET, 2006.

A plataforma de GOUGH (1962) foi utilizada para solucionar problemas em várias áreas da tecnologia, já o trabalho de STEWART (1965) parecia aplicável somente para a função que foi proposto. Apesar desses pressupostos, a plataforma de “seis pernas” (hexápode) é contraditoriamente conhecida como “plataforma de Stewart”.

2.3. Sistemas Robóticos

Um sistema robótico geralmente é constituído dos seguintes elementos (MELLO, 2011):

- Manipulador (sistema mecânico);
- Atuadores;
- Efetuador;
- Sistema de controle;
- Malha de sensoriamento.

O manipulador mecânico é o elemento constituinte da estrutura robótica. Esse elemento é composto por braços (*links* ou elos*)* conectados através de juntas articuladas. Na Figura 5, apresenta-se um manipulador robótico do tipo paralelo, onde as juntas entre os braços são atuadas (KLAFTER *et al*., 1989).

Os atuadores são os elementos responsáveis pelo movimento dos braços. Na área industrial pode-se encontrar uma infinidade de atuadores, tais como: motores elétricos (servomotores, motores de passo, etc.), pistões pneumáticos e hidráulicos, músculos artificiais, etc. A movimentação dos braços é feita através de motores com redução, onde há a aplicação de torques a fim de possibilitar movimentos imprescindíveis para executar determinada tarefa (MELLO, 2011).

O efetuador é o elemento designado para manipular peças, executar a função especificada ao manipulador robótico. O efetuador é propriamente dito, a interface entre o manipulador e o espaço de trabalho onde atuará (MELLO, 2011).

O controlador (microcontroladores, microprocessadores, CLP, etc.) possui a função de ler e manipular as entradas do sistema, de modo a alcançar um desempenho almejado (ROMANO *et al*., 1999).

Encoders, *resolvers* ou tacômetros, por exemplo, são alguns dos tipos de sensores utilizados para controlar os movimentos do robô, coma premissa de permitir a realimentação do sistema de controle, compondo assim, a malha de sensoriamento.

Figura 5 - Elementos de um sistema robótico.
Fonte: MOLINA, 2008.

2.4. Manipuladores Seriais

Na atualidade, quase que a totalidade da utilização de robôs, é referente à manipuladores que apresentam características antropomórficas, conforme pode ser visto na Figura 6, ou seja, que são semelhantes à verdadeiros braços humanos. Estes manipuladores são conhecidos como robôs seriais e são constituídos de uma série de elos conectados por meio de articulações. Podem possuir um ou mais graus de liberdade, que possibilitam movimentos translacionais dos elos e rotação em torno do próprio eixo (MELLO, 2011).

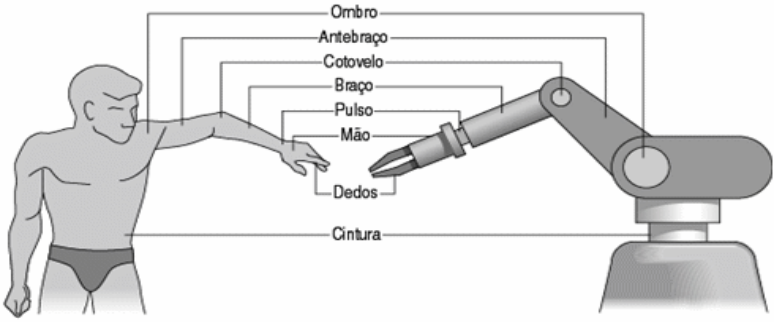


Figura 6 - Braço Humano X braço de robô serial.
Fonte: www.tegruposete7.wordpress.com. Acesso em 19/05/13.

Apesar do bom aspecto visual, o formato e forma de trabalho desses manipuladores não são convenientes para a aplicação em determinadas funções.

Esses robôs foram projetados e criados a fim de executar tarefas feitas por humanos, por isso precisam ter um bom espaço de trabalho e uma grande capacidade de efetuar manobras.

Devido às suas características, a relação entre capacidade de carga e peso é baixa. Outro problema desse tipo de robô, é que podem apresentar flambagem e vibrações de seus elos quando são submetidos ao efeito de cargas, podendo comprometer o desempenho dinâmico do sistema (MELLO, 2011). Em tais condições, a rigidez do sistema é comprometida, ocasionando uma menor precisão e menor capacidade de posicionamento. Por esse motivo, embora a arquitetura serial ainda seja predominante no âmbito robótico, os modelos paralelos vêm ganhando cada vez mais espaço nesse meio (GONÇALVES, 2009). Exemplo de um robô serial abaixo na Figura 7.

Figura 7 - Exemplos de Robô Serial.
Fonte: www.abb.com.br e www.panasonic.com.br. Acesso em 19/05/13.

2.5. Manipuladores Paralelos

Os autores KONG & GOSSELIN (2007) não apresentam uma definição referente à robôs paralelos, porém demarcam mecanismos paralelos como sendo sistemas de vários “n” graus de liberdade compostos por uma plataforma móvel conectada pelo menos à duas cadeias cinemáticas fechadas, que são chamadas elos.

Adentro nesta definição, se classificam mecanismos redundantes onde o número de atuadores é maior que o número de graus de liberdade que controlam o efetuador (*end effector*). Por conta destas características, MERLET (2006) sugere características que delimitam o estudo dos robôs paralelos, sendo elas:

- Ao menos duas cadeias cinemáticas suportam o efetuador. Cada uma dessas cadeias cinemáticas devem ter ao menos um atuador;
- O número de atuadores é igual ao número de graus de liberdade do efetuador;

- A mobilidade do manipulador é zero com os atuadores sem movimento;

Figura 8 - Exemplo de Robô Paralelo - Tricept (E) e Adept Quattro(D).
Fonte: www.pgene.ufabc.edu.br. Acesso em 19/05/13.

2.6. Comparação entre Manipuladores Paralelos e Seriais

Os robôs foram projetados, no princípio, para executar tarefas convenientes aos seres humanos. Portanto, existia a necessidade de possuírem uma alta capacidade de efetuar manobras e bom espaço de trabalho, mas a relação da capacidade de carga com o seu peso era baixa (DASGUPTA & MRUTHYUNJAYA, 2000). Alguns autores, dentre eles ANGELES e GOSSELIN (1988) e TANEV (2000), afirmam que o interesse pelos robôs paralelos deu-se como uma solução para os problemas de pouca capacidade de carga e rigidez que tinham os robôs de configuração serial.

Segundo MOLINA (2008), todos os atuadores são montados próximos à base nos manipuladores paralelos. Isso possibilita uma possível redução da massa nas partes móveis, implicando em melhores características dinâmicas em relação aos seriais.

Uma comparação das qualificações e características entre manipuladores com arquitetura serial e paralela (mecanismo e controle) é apresentada na Tabela 1. Nessa tabela podem-se observar as principais diferenças, assim como as vantagens e desvantagens na utilização e aplicação de cada uma das arquiteturas (MOLINA, 2008).

Nível	Características	Manipuladores	
		Serial	Paralelo
Mecanismo	Inércia	Grande	Pequena
	Volume de Trabalho	Grande	Pequeno
	Aparência	Antropomórfica	Base Estrutural
	Fabricação	Difícil	Fácil
Controle	Controle de Posição no Espaço de Trabalho	Difícil	Fácil
	Controle de Forças no Espaço de Trabalho	Fácil	Difícil
	Detecção de Forças	Difícil	Fácil
	Erro de Posição	Acumulado	Média
	Erro de Controle de Forças	Média	Acumulado
	Perto de Pontos Singulares	Degeneração no controle de força	Diminuição de Exatidão no Posicionamento
		Grande movimento no atuador	Grande Força no Atuador
	Dinâmica	Complicada	Muito Complicada

Tabela 1 - Comparação entre manipuladores seriais e paralelos.
Fonte: MOLINA, 2008.

2.7. Manipuladores Paralelos em Configuração Delta

A arquitetura Delta é considerada por muitos a configuração mais bem-sucedida de todas as arquiteturas paralelas já fabricadas (TARTARI, 2006). Isso deve-se à sua capacidade de trabalhar com aplicações *pick-and-place* com velocidade e precisão (MÜLLER, 2012). Essa configuração surgiu da necessidade constante de movimentar pequenas massas com alta velocidade (MELLO, 2011).

Esse tipo de arquitetura possui seis ligamentos passivos iguais, que ficam agrupados dois a dois e formam com as articulações esféricas e os corpos adjacentes, três paralelogramos (TARTARI, 2006).

Por projeto, as duas articulações esféricas de cada conjunto mais próximas dos motores (acopladas ao ligamento atuado do conjunto), formam uma linha reta que está sempre paralela ao plano da base fixa. Por estarem formando paralelogramos, garante-se que essa linha também estará sempre paralela à plataforma móvel.

Conclui-se assim que a plataforma móvel está sempre paralela à base fixa, demonstrando que a estrutura Delta apresenta apenas graus de liberdade de translação, conforme a Figura 9.

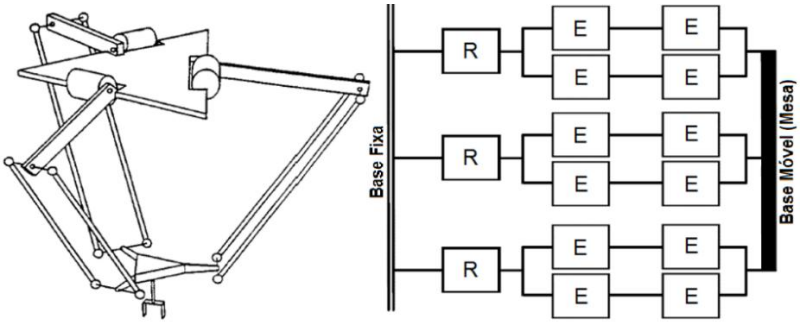


Figura 9 - Esquema do robô Delta (E) e a organização das juntas das cadeias cinemáticas da arquitetura (D).
Fonte: COMPANY e PIERROT, 1999.

As siglas das juntas seguem a classificação abaixo:

- R → Rotativa;
- E → Esférica;

2.8. Componentes e Softwares Utilizados no Projeto do Robô Delta

Para a realização deste projeto foi necessário primeiramente fazer a análise de componentes que melhor atendiam os requisitos e as especificações exigidas. Da mesma forma, procurou-se utilizar os softwares disponíveis no Campus, já que os mesmos dispunham de licenças, e atendiam as necessidades de modelagem e programação do projeto.

Dentre todas as ferramentas e componentes utilizados no projeto, os principais elementos podem ser observados a seguir:

2.8.1. Servomotor

Segundo Ottoboni (2002), o servomotor é um dispositivo eletromecânico que possui uma parte fixa (estator) e outra móvel (rotor), como muitas outras máquinas síncronas. O estator possui bastante semelhança ao de uma máquina elétrica convencional, porém com restrições quanto à alimentação. O rotor é composto por imãs permanentes que são posicionados alinhadamente sobre o rotor e com o controlador, ou gerador de sinais, chamado de resolver ou encoder. É possível verificar a estrutura de um servomotor na Figura 10.

Uma característica dos servomotores é que são dispositivos de malha fechada, ou seja, que recebem um sina de controle, verificam a posição atual e movem-se para a posição desejada. O eixo desse tipo de motor possui a liberdade de até 180 graus e precisão nos movimentos.

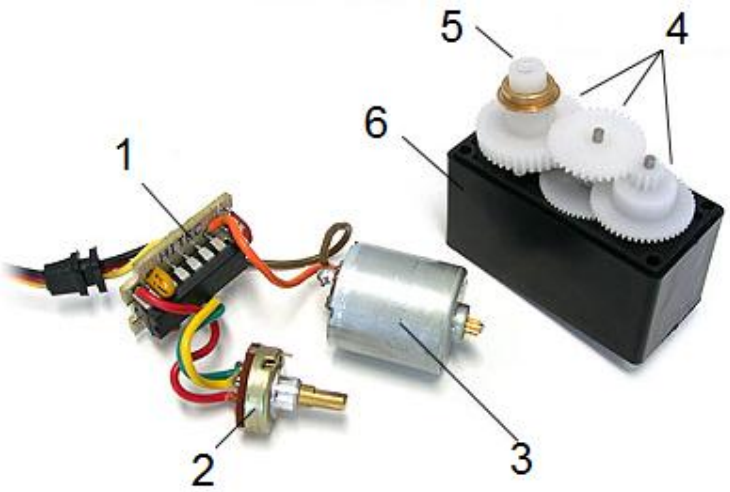


Figura 10 - Estrutura do Servomotor.
Fonte:www.pictronics.com.br. Acesso em 08/06/2013.

Abaixo seguem informações dos componentes do servomotor, referenciados na figura 10:

1) **Circuito de Controle:** Responsável por receber os sinais e energia do receptor, monitorar a posição do potenciômetro e controlar o motor de acordo com o sinal do receptor;
2) **Potenciômetro:** Ligado ao eixo de saída do servo monitora a posição do mesmo;
3) **Motor:** Movimenta as engrenagens e o eixo principal do servo;
4) **Engrenagens:** Reduzem a rotação do motor, transferem mais força ao eixo principal de saída e movimentam o potenciômetro junto com o eixo;
5) **Ponta do eixo de transmissão:** Responsável por transmitir o torque do motor para o dispositivo conectado à saída;
6) **Caixa do servo:** “Carcaça” do conjunto Servomotor.

Os servomotores utilizados no projeto possuem as seguintes características:

	Movimento dos Braços	Movimento da Garra
Quantidade Utilizada	3	2
Modelo	Turnigy TGY-1270HV	SO5NF STD
Tensão de Operação	6 V	6 V
Peso	170 g	20 g
Dimensões	59.5 X 29.5 X 54.3 mm	28.8 X 13.8 X 30.8 mm
Torque	35 Kg.cm	3.2 Kg.cm
Velocidade	0.2 seg/60°	0.18 seg/60°

Tabela 2 - Características dos Servomotores do Projeto

2.8.2. Microcontrolador – Arduino

Um microcontrolador é um sistema microprocessado encapsulado em um único chip (circuito integrado), com memórias, *clock* e periféricos mais limitados que um computador (TAVARES e GOMES, 2013). O uso desses circuitos integrados não somente reduz o custo da automação como também propicia maior flexibilidade (DESHMUKH, 2005). Dentre as plataformas de desenvolvimento que utilizam microcontroladores, o Arduino tem ganhado um grande destaque.

O Arduino consiste, por definição, em um microcontrolador de placa única e um conjunto de software para programá-lo (BANZI, 2011). O hardware é composto de um processador *Atmel* AVR, um cristal oscilador e um regulador linear de 5 volts. A placa expõe os pinos de entrada e saída em um encaixe padrão para que se possam conectar circuitos externos que agregam novas funcionalidades (TAVARES e GOMES, 2013). O *software* trata-se de uma linguagem de programação para desenvolvimento do *firmware* do microcontrolador e do gerenciador de inicialização que é executado na placa. O ambiente Arduino foi desenvolvido para usuários que não

possuem experiência com desenvolvimento de software ou eletrônica (MARGOLIS 2011).

Há uma diversidade de placas do tipo Arduino no mercado, inclusive implementações nacionais (SOUZA, 2011). Na Figura 11 pode-se visualizar a placa de um Arduino Uno.

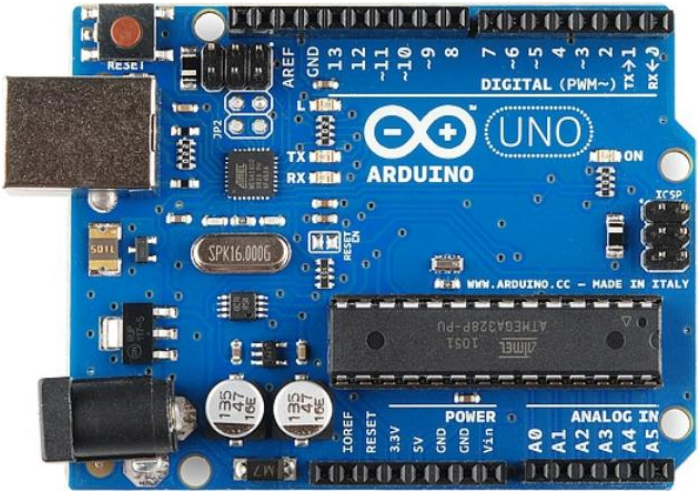


Figura 11 - Arduino Uno.
Fonte: www.dhgate.com. Acesso em: 14/06/2013.

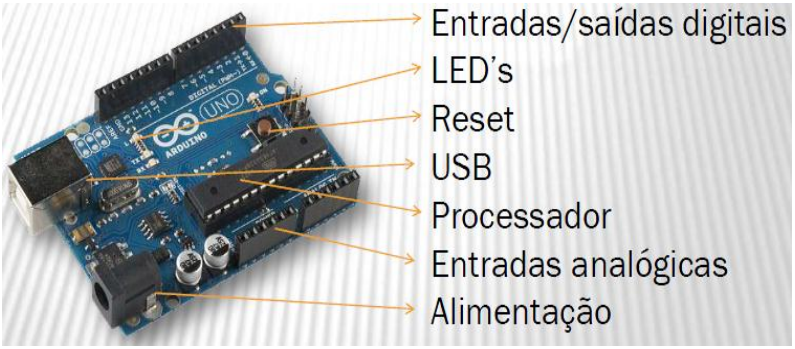


Figura 12 - Descrição das funções dos Componentes do Arduino
Fonte: www.dhgate.com. Acesso em: 14/06/2013.

Seguem os dados básicos referentes ao hardware do Arduino Uno:

Microcontrolador	ATmega328P
Tensão de Operação	5V
Tensão de Entrada (Recomendada)	7-12V
Tensão de Entrada (Limite)	6-20V
I/O Digitais	14 (6 fornecem PWM de saída)
Entradas Analógicas	6
Corrente DC por I/O	40 mA
Corrente DC por Pino (3.3V)	50 mA
Memória Flash	32 KB (ATmega328) no qual 0.5 KB são usadas para bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidade de Clock	16 MHz

Tabela 3 - Características do Arduino Uno.
Fonte: www.arduino.cc. Acesso em: 14/06/2013.

2.8.3. Sistema de Imagem – Webcam

Webcam é uma micro câmera (Figura 13) que é ligada a um computador ou outro dispositivo de controle. Foi criada para uso na internet e pode ser usada em videoconferências, produção de vídeos e imagens. Em sua grande maioria, são portáteis, e sua conexão acontece comumente através de um cabo USB (barramento serial universal, do inglês *Universal Serial Bus*).

Alguns itens são bastante importantes na hora de escolher uma webcam (CARVALHO, 2011):

- Possibilidade de usar altas taxas de gravação de vídeo;

- Alta qualidade de imagens, formatos de vídeo diversificados (AVI, sequência BMP, RAW, etc.) e boa resolução;
- Que tenha, ou permita, o uso de *softwares* de fácil manuseio com funções de ajuste dos parâmetros básicos da câmera (exposição, ganho e controle de cores) plenamente funcionais.

Figura 13 - Webcam Logitech C210.
Fonte:www.infotechnow.com. Acesso em 08/06/2013.

2.8.4. Software de Simulação e Controle – MatLab

O MatLab é um "*software*" destinado a fazer cálculos com matrizes (MATrix LABoratory), podendo funcionar como uma calculadora ou como uma linguagem de programação científica. Entretanto, os comandos do MatLab são mais próximos da forma como se escrevem expressões algébricas, tornando mais simples o seu uso.

Atualmente, o MatLab é definido como um sistema interativo e uma linguagem de programação para computação técnica e científica em geral, integrando a capacidade de fazer cálculos, visualização gráfica e programação (TONINI e COUTO, 1999).

Uso típico do MatLab:

- Cálculos matemáticos;
- Desenvolvimento de algoritmos;
- Modelagem, simulação e confecção de protótipos;
- Análise, simulação e confecção de dados;
- Gráficos científicos e de engenharia;
- Desenvolvimento de aplicações, incluindo a elaboração de interfaces gráficas com o usuário.

2.8.4.1. GUIDE

O MatLab fornece uma ferramenta para desenvolvimento de uma interface gráfica com o usuário (GUI, do inglês Graphical User Interface), que se chama GUIDE. A Figura 14 mostra um exemplo de interface de comando, criada para a interação com o operador do robô (primeira interface criada durante os testes).

As razões para a escolha desta ferramenta de desenvolvimento recaem sobre os seguintes pressupostos (FERREIRA, 2007):

- Muito do trabalho desenvolvido para este projeto estava em MatLab, entre os quais os algoritmos de cinemática, as definições e animações virtuais;
- Não havia necessidade de aprender linguagens de alto-nível vocacionadas para o desenvolvimento visual, como C#, Visual Basic, Java, etc.

Figura 14 - Exemplo de Interface de Comando.

2.8.5. Software de Modelagem de Partes Mecânicas – SolidWorks

O SolidWorks é um software para modelagem mecânica, onde através de esboços são modelados objetos sólidos. Além disso, permite fazer montagens, detalhamento e simulações. O mesmo permite a simulação de modelos 3D. Esse software é utilizado por estudantes, designers, engenheiros e outros profissionais para produzir componentes simples e complexos, conjuntos e desenhos (FONSECA e TAVARES, 2012).

Figura 15 - Logo do software SolidWorks.
Fonte: www.rocksolidperspective.com. Acesso em 08/06/13.

3. ANÁLISE CINEMÁTICA

Um manipulador é composto por uma série de corpos rígidos, conectados por pares cinemáticos ou juntas, conforme a Figura 16. A composição desses elementos forma uma cadeia cinemática, onde em um dos lados está a base e no outro extremo encontra-se o efetuador final, que pode ser uma garra ou outra ferramenta que permita a manipulação de objetos no espaço (HESS-COELHO, 2005). Conhecer a cinemática de um manipulador é de fundamental importância para o controle de posição e velocidade do efetuador final.

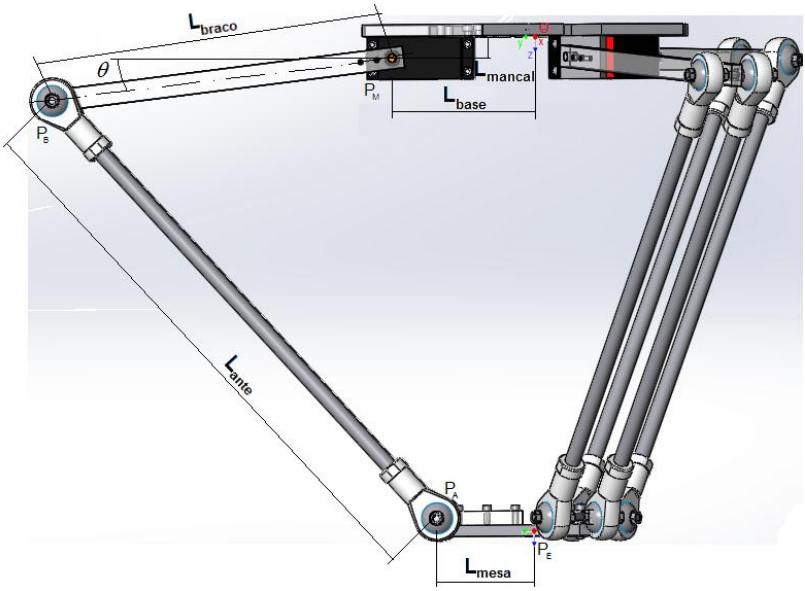


Figura 16 - Pontos e Dimensões do Manipulador.

Durante toda a análise cinemática faz-se necessário utilizar a nomenclatura dos pontos, bem como as dimensões de cada elo, também detalhada na Figura 16, sendo:

$$\begin{aligned} L_{base} &= 85\text{mm} \\ L_{mancal} &= 15\text{mm} \\ L_{braço} &= 240.5\text{mm} \\ L_{ante} &= 400\text{mm} \\ L_{mesa} &= 72.5\text{mm} \end{aligned} \tag{3.1}$$

Conforme descrito anteriormente, as três cadeias cinemáticas que compõem o manipulador são idênticas, e por este motivo todas as variáveis e coordenadas serão numeradas de 1 a 3, de acordo com a cadeia cinemática em questão.

3.1. Cinemática Inversa

O estudo da cinemática inversa consiste em estabelecer o valor das variáveis de junta do manipulador correspondente a uma dada posição do efetuador final. O cálculo da cinemática inversa é essencial para o controle de posição dos robôs paralelos.

Sendo assim, considera-se conhecida a posição do efetuador final que é representada pelas coordenadas cartesianas x_E, y_E e z_E do ponto de referência P_E que fica no centro da mesa móvel do manipulador.

Primeiramente analisar-se-á somente a cadeia cinemática 1, com objetivo de obter todas as variáveis das juntas (o ângulo da junta ativa θ_1).

Analisando a Figura 16, podem-se definir as coordenadas dos pontos P_{M1} e P_{A1} conforme as equações a seguir:

$$P_{M1} = \begin{cases} x_{M1} = 0 \\ y_{M1} = L_{base} \\ z_{M1} = L_{mancal} \end{cases} \quad (3.2)$$

$$P_{A1} = \begin{cases} x_{A1} = x_E \\ y_{A1} = y_E + L_{mesa} \\ z_{A1} = z_E \end{cases} \quad (3.3)$$

Uma vez conhecidos os pontos P_{M1} e $P_{A1,}$ conforme as equações (3.2) e (3.3) pode-se então definir a posição do ponto $P_{B1,}$ tomando como ponto de partida as restrições geométricas entre os elos.

Analisando a restrição imposta pelo primeiro elo vê-se que a distância entre os pontos P_{M1} e P_{B1} tem valor fixo equivalente ao comprimento do braço (L_{braco}), ou, em outras palavras, o ponto P_{B1} está localizado em uma circunferência de raio L_{braco} com centro no ponto $P_{M1,}$ situada no plano *Y Z*. Sendo assim, tem-se:

$$r^2 = (y_{ponto} - y_{centro})^2 + (z_{ponto} - z_{centro})^2 \quad (3.4)$$

$$L_{braço}^{\ 2} = (y_{B1} - y_{M1})^2 + (z_{B1} - z_{M1})^2 \quad (3.5)$$

Com os valores de P_{M1} encontrados na equação (3.2), ter-se-á:

$$L_{braço}^{\ 2} = (y_{B1} - L_{base})^2 + (z_{B1} - L_{mancal})^2 \quad (3.6)$$

Da mesma forma os pontos P_{B1} e P_{A1} tem distância fixa definida pelo comprimento do antebraço (L_{ante}). Como a união do segundo elo aos demais é feita através de duas juntas esféricas, pode-se afirmar que o ponto P_{B1} pertence à superfície de uma esfera com centro no ponto P_{A1} e raio L_{ante}. Portanto, aplicando a equação genérica de uma esfera no espaço, obtém-se:

$$r2 = (x_{ponto} - x_{centro})2 + (y_{ponto} - y_{centro})2 + (z_{ponto} - z_{centro})2 \tag{3.7}$$

$$L_{ante}{}^2 = (x_{B1}^{0} - x_{A1})2 + (y_{B1} - y_{A1})2 + (z_{B1} - z_{A1})2 \tag{3.8}$$

Pode-se simplificar a equação (3.8) sabendo que a coordenada do ponto P_{B1} no eixo X será sempre nula, pois por construção mecânica o primeiro elo não permite movimento nessa direção. Assim, eliminando x_{B1} e substituindo os valores de P_{A1} encontrados na equação (3.3), ter-se-á:

$$L_{ante}{}^2 = (-x_E)2 + \left[y_{B1} - (y_E + L_{mesa}) \right]^2 + (z_{B1} - z_E)2 \tag{3.9}$$

Desta forma, analisando as equações (3.4) e (3.5), pode-se perceber que temos apenas as incógnitas y_{B1} e z_{B1}. Com isso é possível montar o sistema de duas equações e duas incógnitas, conforme mostrado abaixo:

$$\begin{cases} L_{braço}{}^2 = (y_{B1} - L_{base})2 + (z_{B1} - L_{mancal})2 \\ L_{ante}{}^2 - (-x_E)2 = \left[y_{B1} - (y_E + L_{mesa}) \right]^2 + (z_{B1} - z_E)2 \end{cases} \tag{3.10}$$

Para encontrar a solução do sistema exposto foi utilizado o recurso de variáveis simbólicas do software MatLab, conforme código do programa listado a seguir na Figura 17:

```
clc
clear all

%Criando    todas   as  variábeis   simbólicas
syms Lmancal Lante Lbraco Lmesa Lbase;
syms xE yE zE;
syms zM1 yM1 zM1;
syms xB1 yB1 zB1;
syms xA1 yA1 zA1;
%Definindo  ponto PM1
xM1 = 0;
yM1 = Lbase;
zM1 = Lmancal;
%Definindo  ponto PA
xA1 = xE ;
yA1 = yE + Lmesa ;
zA1 = zE ;
%Definindo  coordenada  x   do  ponto PB1
xB1 = 0;
%Definindo  as  duas    equações    do  sistema
eq1 = (yB1-yM1)^2+(zB1-zM1)^2 - Lbraco^2;
eq2 = (xB1-xA1)^2+(yB1-yA1)^2+(zB1-zA1)^2-Lante^2;
%Executando a   função   para   cálculo da  solução
S = solve ( eq1 , eq2 , 'yB1 , zB1' );
```

Figura 17 - Variáveis Simbólicas.

Como resultado deste cálculo obtém-se duas soluções distintas, isso se dá porque o sistema matemático não leva em consideração as restrições mecânicas do manipulador. Na Figura 18 é mostrado um exemplo de duas possíveis soluções matemáticas encontradas para o posicionamento do ponto P_{B1}.

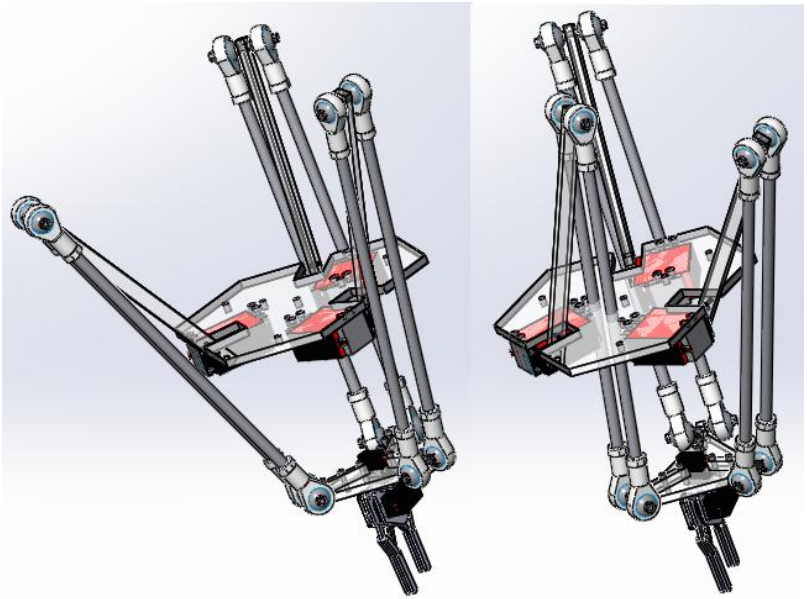


Figura 18 - Dois Possíveis Resultados do Cálculo da Posição.

Uma vez conhecidas as posições de todas as juntas do manipulador (pontos P_{M1}, P_{B1} e P_{A1}) pode-se, através de uma simples análise geométrica, calcular o ângulo da junta ativa θ_1.

$$\theta_1 = \arctan\left(\frac{z_{B1} - z_{M1}}{y_{B1} - y_{M1}}\right) \tag{3.11}$$

Depois de feita toda a análise para a primeira cadeia cinemática do manipulador, passa-se a buscar os ângulos de juntas das demais. Como as três cadeias cinemáticas são idênticas, a análise feita para a cadeia 1 se repetirá para as demais. Na primeira cadeia tinha-se o braço se deslocando no plano *YZ* o que simplificou bastante os cálculos. As três cadeias do manipulador estão posicionadas à 120° uma das outras, assim, utilizou-se uma matriz de transformação, denominada R_Z,

para rotacionar o sistema de coordenadas em 120° em torno do eixo *Z*:

$$R_Z = \begin{bmatrix} \cos\theta & -sen\theta & 0 \\ sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos 120 & -sen120 & 0 \\ sen120 & \cos 120 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.12)$$

Logo em seguida multiplicam-se os valores encontrados pelas coordenadas do ponto P_E, obtendo-se:

$$P'_E = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_E \\ y_E \\ z_E \end{bmatrix} \Rightarrow P'_E = \begin{cases} x'_E = -\dfrac{x_E + \sqrt{3}.y_E}{2} \\ y'_E = \dfrac{\sqrt{3}.x_E - y_E}{2} \\ z'_E = z_E \end{cases} \quad (3.13)$$

De maneira similar executa-se uma nova rotação do sistema de coordenadas obtendo o sistema $X^{"}Y^{"}Z^{"}$, sendo que o plano $Y^{"}Z^{"}$ fica posicionado no plano de trabalho do braço da terceira cadeia cinemática. Calculam-se então as novas coordenadas do ponto P_E através de uma nova multiplicação de R_Z pelas coordenadas de P'_E :

$$P''_E = \begin{bmatrix} -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x'_E \\ y'_E \\ z'_E \end{bmatrix} \Rightarrow P''_E = \begin{cases} x''_E = \frac{x_E - \sqrt{3}.y_E}{2} \\ y''_E = -\frac{\sqrt{3}.x_E + y_E}{2} \\ z''_E = z_E \end{cases} \quad (3.14)$$

Utilizando os pontos P'_E e P''_E, pode-se calcular, da mesma maneira que obteve-se o ângulo θ_1 à partir de P_E, os ângulos θ_2 e θ_3, respectivamente.

Para concluir a análise da cinemática inversa foi criado um código no MatLab e incluído no anexo B. Esta é uma função que tem como entrada as coordenadas do ponto de referência do efetuador final e, após executar os cálculos aqui demonstrados, retorna os ângulos de cada junta ativa do manipulador.

4. DESENVOLVIMENTO

O projeto do manipulador paralelo em configuração Delta é necessariamente interdisciplinar e envolve a utilização de conhecimentos de diferentes unidades curriculares, tais como:

- Projetos Mecânicos: a qual fornece metodologias para o estudo de estruturas e mecanismos em situações estáticas e dinâmicas;
- Controladores Lógicos Programáveis: fornecem técnicas para o projeto e integração de sensores, interfaces, atuadores e controladores;
- Teoria de controle: formula e avalia algoritmos ou critérios de inteligência artificial que controlam os movimentos desejados e gerenciam as interações entre o robô e o ambiente; e,
- Informática Industrial e programação: propicia ferramentas para a programação de robôs, capacitando-os à realização das tarefas específicas.

Neste tipo de projeto deve-se ainda considerar entre outros aspectos:

- Dimensionamento de atuadores, mecanismos, circuitos eletrônicos (hardware), unidades de controle e potência;
- Fabricação e montagem de peças;
- Seleção de materiais;
- Planificação dos movimentos;
- Simulação e modelagem;
- Desenvolvimento de técnicas de programação para o sistema de controle, sistema operacional, diagnose de sistemas/componentes e comunicação ao operador; e
- Testes de desempenho.

Os robôs são máquinas programáveis, projetadas para operar em diversas situações, logo, as especificações de operação relacionam-se a: volume de trabalho, capacidade de carga, velocidade máxima, precisão e repetitividade.

4.1. Pré-Protótipo do Robô Delta

Com a ideia formatada do projeto, antes de iniciar o desenvolvimento do protótipo original, decidiu-se primeiramente fazer um “pré-protótipo” (pois se dispunha de três servomotores com potência inferior à que seria usada, 13kg/cm) ou seja, uma forma miniaturizada do robô Delta para verificar a viabilidade de tal proposta e também para o desenvolvimento de toda a programação até que protótipo original ficasse inteiramente pronto.

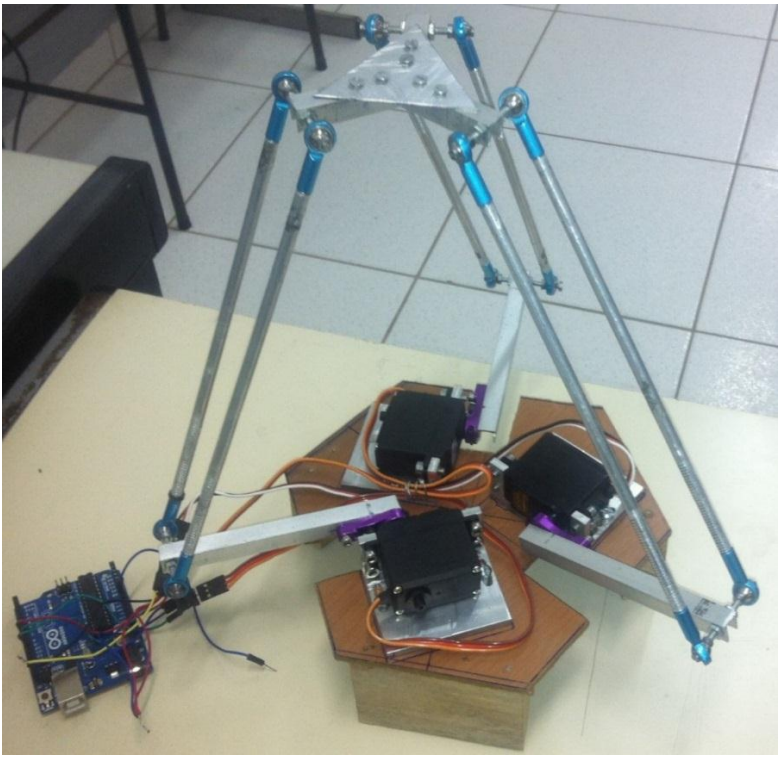


Figura 19 - Pré-protótipo do Manipulador.

4.2. Arquitetura

A estrutura do robô além de ser responsável pela rigidez, deve ter baixo peso, dimensões adequadas à aplicação do mesmo, deve ser resistente e de fácil construção.

O Robô Delta está disponível no mercado há algum tempo, por isso, para a construção do mesmo, houve a necessidade de analisar as características estruturais, sensoriamento e locomoção de alguns dos principais modelos comerciais, dentre eles:

- FLEXPICKER IRB 360 fabricado pela ABB;
- M-1*i*A fabricado pela FANUC;
- MPP3 fabricado pela MOTOMAN;
- YF03N fabricado pela KAWASAKI;
- ACCURAX G5 fabricado pela OMRON;
- POWERDELTA 250 fabricado pela ASYRIK;
- U10H fabricado pela VELTRU.

A partir desta análise, buscou-se implementar a melhor configuração, levando-se em conta as características de todos os modelos supra citados, a fim de definir qual seria a melhor estrutura com relação à massa, dimensão, capacidade de carga e velocidade.

Além de tal análise, uma pesquisa foi realizada tomando como princípio de que o robô Delta fora um projeto desenvolvido para consumo do mercado. Esse levantamento pode ser verificado no anexo A.

4.3. A Seleção dos Materiais

Apesar de todos os materiais terem sido adquiridos comercialmente, alguns encontravam-se em forma de matéria-prima bruta. Foi necessário executar um processo de manufatura (usinagem de modo geral), a fim de obter as peças que pudessem atender os requisitos de projeto. Dentre as atividades realizadas, podem-se citar:

- Corte, fresamento e furação dos braços;
- Corte e rosqueamento dos antebraços;

- Corte, fresamento e furação da base fixa e móvel;
- Furação das bases estruturais;
- Corte, faceamento (do topo) e rosqueamento dos perfis de alumínio;
- Corte, fresamento e furação dos suportes de fixação dos servomotores.

4.3.1. Aquisição de Itens Comerciais e Custo do Projeto

Com base na análise realizada e após a etapa de modelamento do protótipo, fez-se necessária a elaboração de uma lista de materiais que poderiam ser adquiridos no mercado, conforme Tabela 3:

Componente	Custo Total	Observações
Policarbonato (m²)	R$ 220,00	Para confecção das bases e dos braços
Barra Cilíndrica de Alumínio	R$ 12,00	Para confecção dos antebraços
Juntas Esféricas	R$ 240,00	12 juntas esféricas
Material para Estrutura	R$ 710,00	Perfis de alumínio 40 X 40 mm - Barra com 6 metros
		Componentes de fixação (porcas, parafusos, pinos, etc.)
		Chapas em MDF (espessura de 18 mm)
Placa Arduino	R$ 60,00	
Servomotores	R$ 320,00	3 para locomoção dos braços
		2 para movimentação da garra
Garra	R$ 55,00	
Fonte de Alimentação	R$ 150,00	3 fontes reguláveis 220Vca/5Vcc - 5A
Placa de Circuito Impresso	R$ 10,00	Placa para conexão dos servomotores ao Arduino
Total	R$ 1.777,00	

Tabela 4 - Custos do Projeto.

4.4. Construção

A modelagem mecânica do robô Delta pode ser resumida basicamente em três fases: o projeto mecânico em ambiente CAD (*Computer Aided Design* - desenho auxiliado por computador), a criação do modelo mecânico funcional em ambiente CAE (*Computer Aided Engineering* - engenharia auxiliada por computador), utilizado para a análise da cinemática e do seu movimento, e a manufatura da planta mecânica.

O mecanismo do robô Delta possui diversos componentes (MÜLLER, 2012):

- Uma base triangular equilátera com três elos primários, onde cada um é ligado ao centro de uma aresta da base servindo como junta ativa;
- Três pares de elos secundários, onde cada par é ligado a um dos elos primários através de juntas esféricas passivas;
- Uma base móvel que fecha uma cadeia cinemática paralela ao conectar cada centro de suas arestas com um dos pares dos elos secundários, utilizando juntas esféricas passivas.

Para projetar mecanicamente todos estes componentes em ambiente CAD, foi utilizada uma técnica que procura facilitar a prototipagem da mesma, ou seja, que permite que tudo o que foi modelado possa ser construído. Esta técnica se chama *Design For Manufacturing* (DFM) e é muito utilizada em projetos do gênero. O DFM promove o uso de peças mecânicas padrões e a modelagem mecânica de peças em software CAD, de tal forma que possam ser prototipadas utilizando materiais e processos de fabricação existentes (BACK, 2008).

Baseados nestas premissas, diversos componentes padrões foram adquiridos e outros fabricados, cuja união dos mesmos pode ser observada no exemplo na Figura 20, que apresenta uma vista da montagem dos componentes do robô Delta dentro do ambiente CAD (o software CAD utilizado para tal modelamento foi o SolidWorks - item 2.8.5).

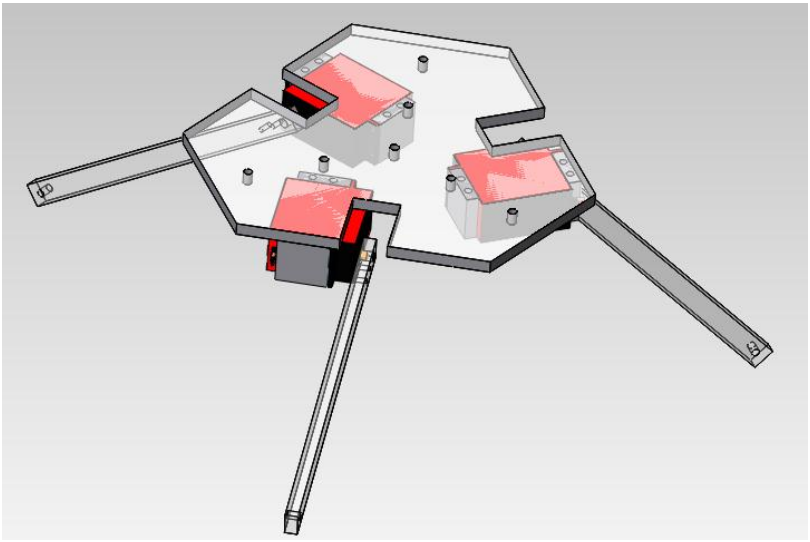


Figura 20 - Exemplo de Componentes Modelados no SolidWorks.

4.4.1. Estrutura de Sustentação para a Montagem do Manipulador

Para a construção estrutural, no qual o manipulador será fixo, optou-se pela escolha de perfis de alumínio por possuírem excelente acabamento, facilidade de montagem e desmontagem, caso seja necessária à locomoção da estrutura para fora do Campus. Tais perfis podem ser vistos na Figura 21. Decidiu-se usar perfis com dimensões de 40X40 mm embasados em modelos similares pesquisados e também para garantir uma melhor estabilidade estrutural. A estrutura foi construída utilizando nove barras (do perfil), sendo:

- Três utilizadas como colunas, cada uma com 1000mm;
- Três utilizadas como fixação superior, cada uma com 440mm; e,
- Três utilizadas como fixação inferior, cada uma com 400mm.

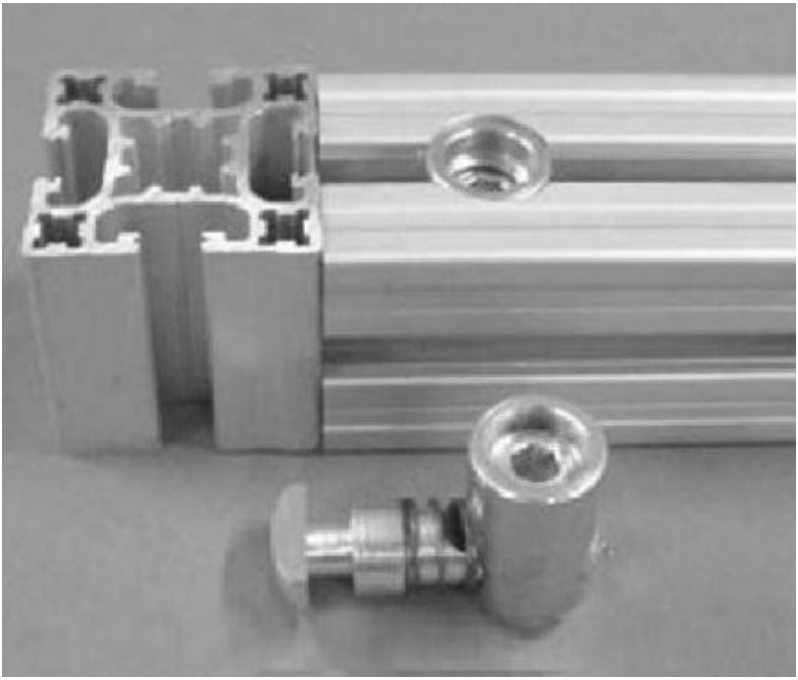


Figura 21 - Perfis de Alumínio e seus elementos de fixação.
Fonte: www.famak.com.br. Acesso em 14/06/2013.

A estrutura ainda conta com duas bases de MDF, uma superior e uma inferior, que foram escolhidas por possuir uma boa relação custo benefício e excelente aspecto visual. Ambas possuem formato hexagonal, sendo que a dimensão das arestas maiores é 700mm e das menores é 40mm.

A fixação dos perfis de alumínio e das bases de madeira foi feita através de parafusos, porcas e pinos adquiridos comercialmente.

Figura 22 - Estrutura de Sustentação Montada - Modelada no SolidWorks (E) e Imagem Real (D).

4.4.2. Estrutura do Manipulador

4.4.2.1. Base Fixa

A base fixa será montada à base de madeira superior, sendo que mesma foi fabricada utilizando lexan (resina termoplástica de policarbonato). Esse item tem formato hexagonal com três lados de 160mm e três de 100mm. Nos lados maiores ainda foram feitas aberturas de 45 X 30mm para possibilitar a movimentação dos braços, conforme a Figura 23.

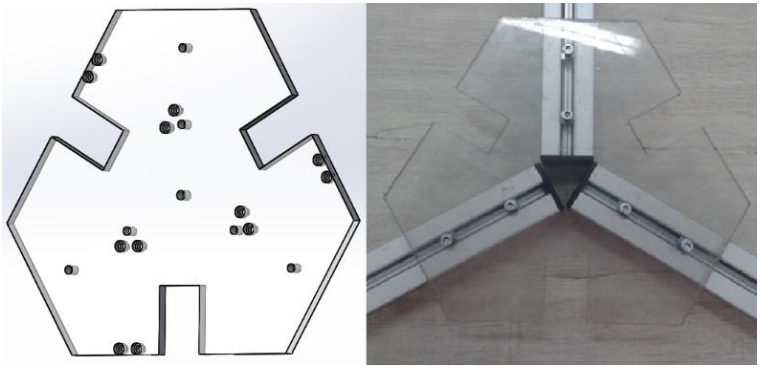


Figura 23 - Base Fixa onde serão montados os servomotores do manipulador - Modelada no SolidWorks (E) e Imagem Real (D).

4.4.2.2. Base Móvel

A Base móvel foi feita no mesmo material da fixa e também tem formato hexagonal, porém com dimensões menores, três lados com 110mm e três com 10mm. No centro desse item há uma abertura com o formato correto para o encaixe do servomotor responsável pela rotação da garra. A mesma será conectada nas extremidades dos três braços conforme a Figura 24.

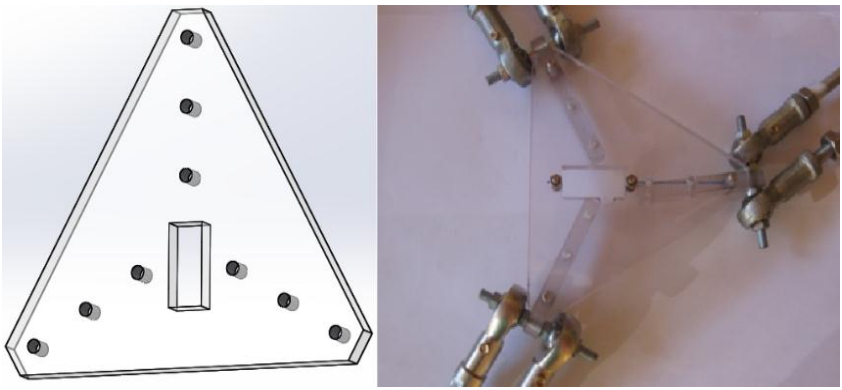


Figura 24 - Base móvel onde será montada a garra - Modelada no SolidWorks (E) e Imagem Real (D).

4.4.2.3.Braços

O braço foi feito em Lexan com formato retangular. Suas dimensões são 257x15x8mm. Esse item tem uma extremidade fixa ao antebraço e outra ao eixo do servomotor.

Figura 25 - Braço - Modelado no SolidWorks (C) e Imagem Real (B).

4.4.2.4.Antebraços

Cada antebraço do robô é constituído de cinco peças:

- Uma haste cilíndrica em alumínio com 360mm de comprimento e 8mm de diâmetro. A mesma ainda possui as extremidades roscadas;
- Duas juntas esféricas metálicas;
- Duas porcas M8 para fixação das juntas na haste.

As juntas esféricas metálicas foram selecionadas pela sua boa aplicabilidade no projeto, boa resistência mecânica e sobretudo pela facilidade de aquisição no mercado, através de um parceiro comercial de um dos membros da equipe.

Figura 26 - Antebraço - Modelado no SolidWorks (C) e Imagem Real (B).

4.4.2.5.Garra

Optou-se por um modelo comercial, o MK-II mostrado na Figura 27, devido à sua grande complexidade de construção (contém várias partes pequenas e várias junções de múltiplas peças) e ao baixo custo de mercado. Os servomotores, utilizados para sua movimentação e manipulação, foram direcionados pelo mesmo fornecedor da garra. O modelo é o DG S05NF STD com 3.2 kg.cm

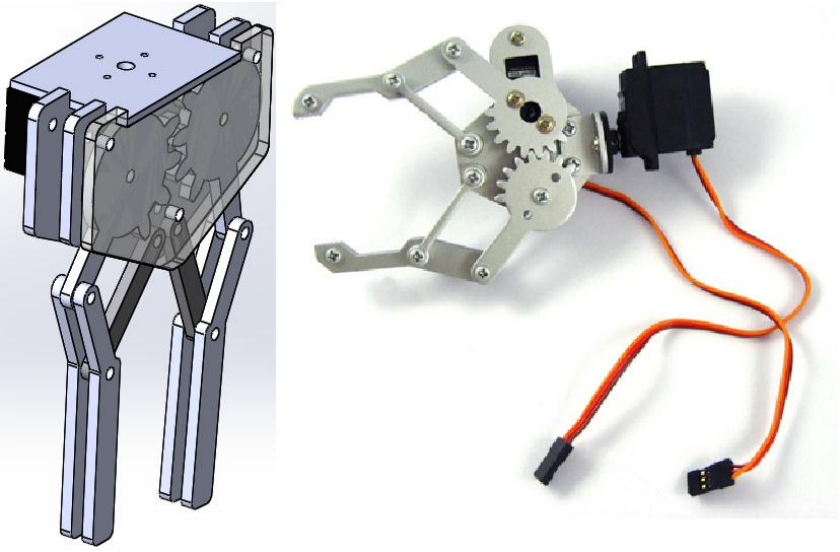


Figura 27 - Garra - Modelada no SolidWorks (E) e Imagem Real (D).

4.5. Adaptações do projeto

Durante a construção do manipulador robótico foi necessário fazer várias adaptações nas peças do projeto. Tais modificações surtiram grande impacto no decorrer da construção do robô, visto que foi necessário gastar um período considerável de tempo na tentativa de encontrar uma forma de ajustar o

projeto original. Tais modificações no projeto variam desde o encaixe de um simples parafuso até mesmo a complexa tarefa de ajustar corretamente alguns servomotores para a correta movimentação.

4.5.1. Servomotores

Os três servomotores dos braços, para os testes iniciais do pré-protótipo, tinham a capacidade de carga de 13 kg.cm.

Após os primeiros testes realizados com os servomotores do pré-protótipo, e em base nas pesquisas e com protótipos já montados, verificou-se que os servomotores deveriam ter a capacidade de aproximadamente 24~30 kg.cm (para uma estrutura com capacidade de levantamento de carga de aproximadamente 1kg), porém foram escolhidos servomotores de 35 kg.cm, devido à disponibilidade de mercado e à boa relação custo-benefício. O modelo do mesmo é o TURNIGY TGY-1270HV representado na Figura 28.

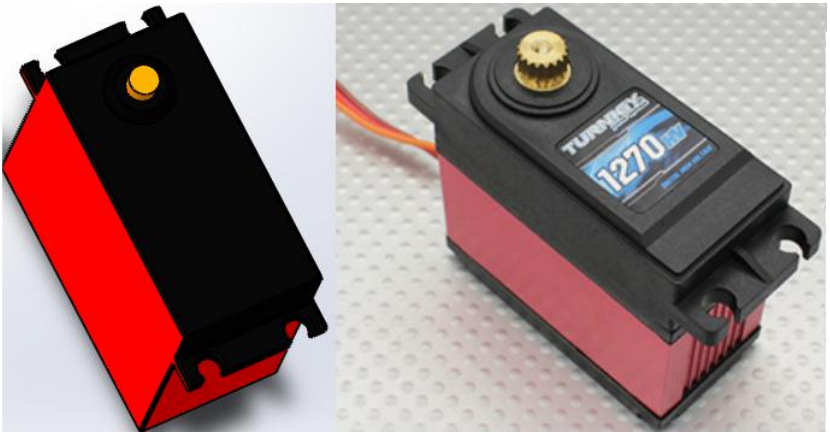


Figura 28 - Servomotor para acionamento dos braços – Modelado no SolidWorks (E) e Imagem Real (D).

4.6. Montagem do Protótipo

A mecânica do robô Delta foi previamente modelada no CAD SolidWorks, permitindo a observação dos detalhes dos seus componentes para sua posterior prototipagem. Seguindo padrões para um projeto otimizado com o DFM, muitos componentes foram fabricados utilizando os processos existentes dentro do laboratório do IFSC e outros foram adquiridos no mercado, conforme item 4.3.1. Após a obtenção de todos os componentes, o robô foi montado sob uma estrutura com perfis de alumínio responsável pela sua sustentação. Além disso, o sistema elétrico (cabos e conexões), de servo acionamento, entre outros, foram adicionados ao mesmo. A Figura 29 mostra a montagem final do protótipo de acordo com as características definidas no projeto.

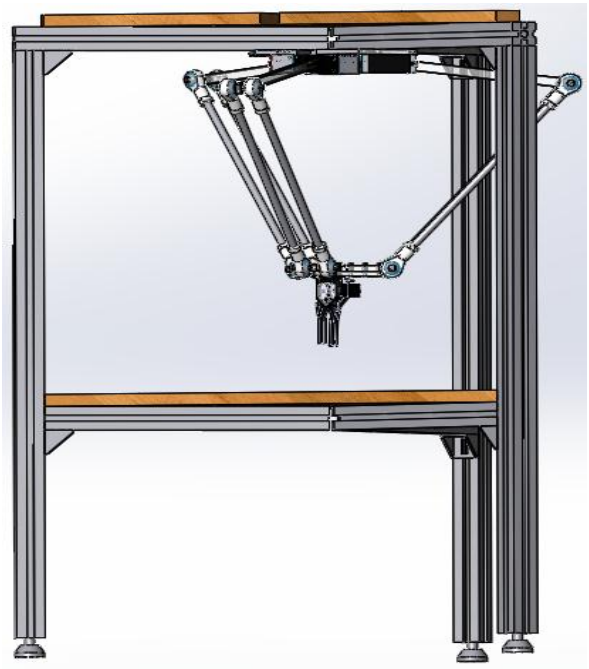


Figura 29 - Modelagem do Protótipo Finalizado.

4.7. Construção da Parte Elétrica

Para alimentação de todo o circuito serão utilizadas três fontes (comerciais) de 220Vca / 5Vcc – 5A com saída regulável para no máximo 6Vcc.

O circuito de comando dos servomotores (hardware elétrico) é basicamente constituído pela placa do Arduino Uno (item 2.8.2), sendo que a mesma é alimentada através da entrada USB do computador (PC).

O protótipo possui dois servomotores para a garra, mais três servomotores maiores para os braços, no qual são conectados ao circuito principal (Arduino). Ainda é necessária a utilização de uma webcam para a captura das imagens para localizar as peças no espaço de trabalho do robô.

Os servomotores foram conectados diretamente às saídas digitais (*Digital Output*) da placa do Arduino, conforme Figura 30, já a Webcam é conectada diretamente ao PC, através de uma porta USB.

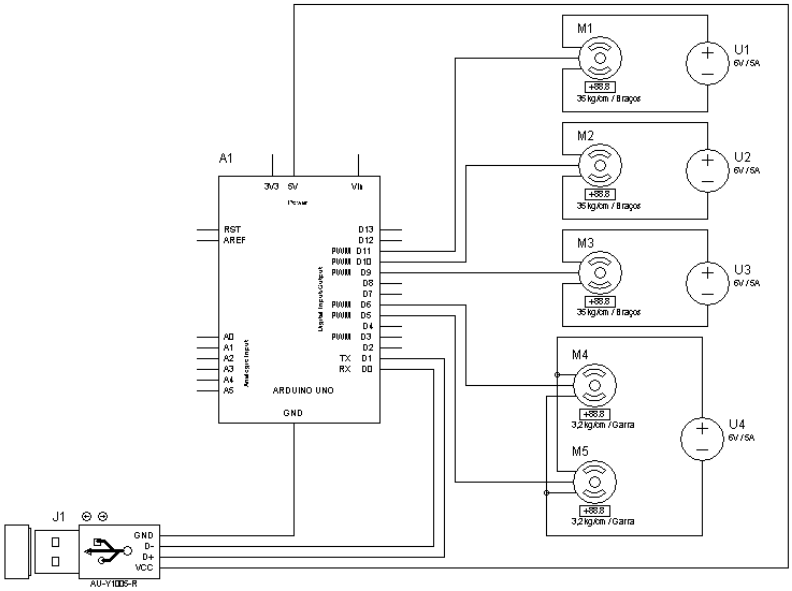


Figura 30 - Representação do diagrama de ligação do Arduino.

4.7.1. Placa de Circuito Impresso

Para facilitar a conexão dos servomotores à placa do Arduino, foi desenvolvida uma placa de circuito impresso, conforme Figura 31. A utilização dessa placa eliminou a necessidade de utilizar condutores elétricos e possíveis mau contatos decorrentes da utilização dos mesmos. A placa desenvolvida é conectada diretamente ao lado superior da placa Arduino e aos conectores dos servomotores.

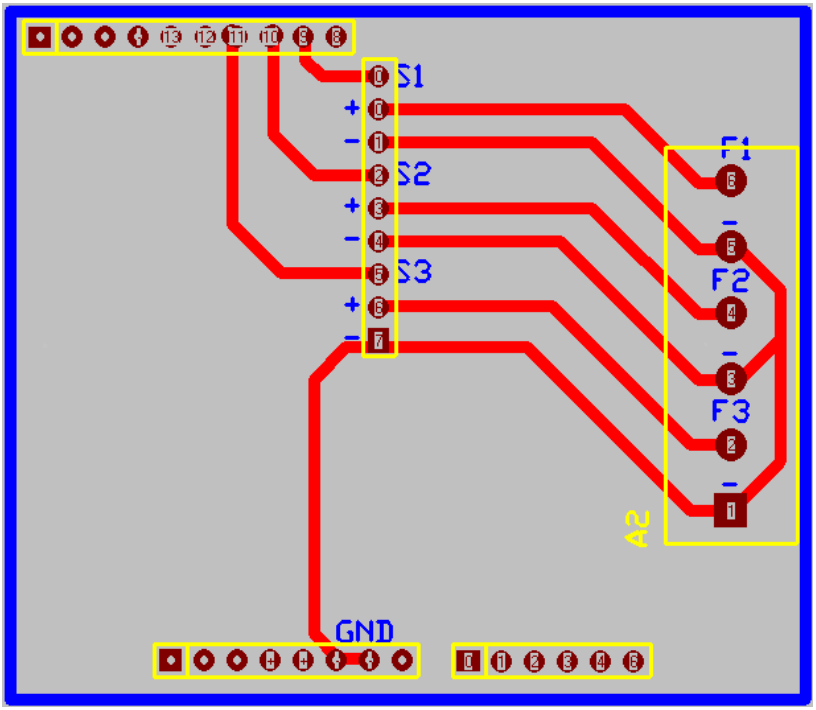


Figura 31 - Placa para Conexão dos Servomotores ao Arduino.

5. CONTROLE DE MOVIMENTO

O sistema de controle de qualquer robô é realizado por meio de um sistema de “software” e “hardware”. Este sistema processa os sinais de entrada e converte estes sinais em uma ação ao qual foi programado.

Para a implementação do *software* de controle de movimento foi utilizado o ambiente de desenvolvimento MatLab. Optou-se por esta ferramenta por dois principais motivos: domínio razoável das facilidades oferecidas pelo MatLab e o fato deste ambiente permitir a criação de uma interface, através da função GUIDE, que foi utilizada para controlar simultaneamente os servomotores e visualizar as imagens fornecidas pela webcam.

Para que o software pudesse enviar corretamente os respectivos comandos aos acionamentos, gerando assim as trajetórias, foi necessário desenvolver uma interface de comunicação. Desta forma, o Arduino troca informações com a porta USB do computador e com os servomotores, isso permite que os mesmos recebam os comandos vindos da interface projetada em ambiente GUIDE.

5.1. Programação e Software de Controle

O software de controle do robô, utilizado através da Interface Homem-Máquina (IHM) do projeto, foi desenvolvido em ambiente MatLab.

Logo abaixo é apresentada uma breve descrição de cada um dos principais módulos criados durante o desenvolvimento deste software:

- **Comunicação serial:** acesso direto ao microcontrolador da placa do Arduino;
- **Parâmetros de ajuste de posição dos servomotores:** são ajustes necessários que podem variar de acordo com as características físicas dos servomotores;

- **Controle dos servomotores:** ferramenta de controle simultâneo;
- **Interface de comando:** IHM para comando do robô;

A Figura 32 apresenta a tela do software desenvolvido e utilizado para realizar o controle remoto do robô.

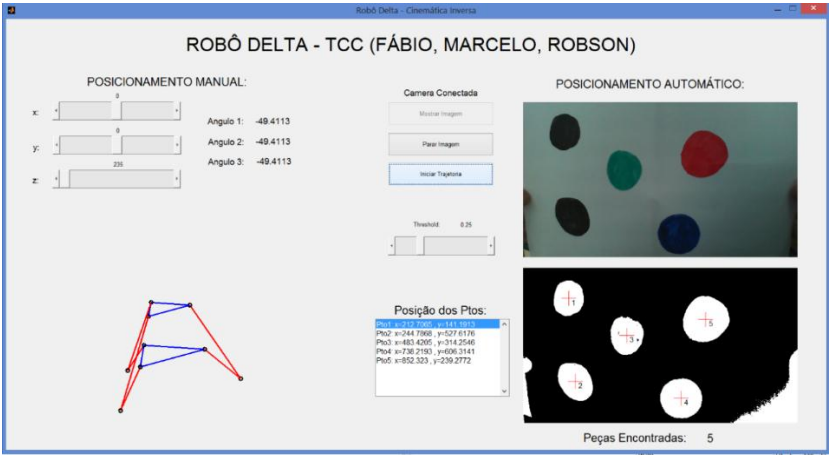


Figura 32 - Tela do Software.

A seguir serão abordados em maiores detalhes os principais módulos de controle utilizados neste software.

5.1.1. Comunicação Serial

O *MatLab Support Package* (também conhecido como "Pacote Arduino I/O") é um pacote desenvolvido pelo engenheiro Italiano Gianpiero Campa. O mesmo permite a utilização do MatLab para se comunicar com a placa Arduino através de um cabo USB. Este pacote é baseado em um programa servidor rodando na placa, que atende aos comandos que chegam via porta serial, executa-os e, se necessário, retorna um resultado.

O conjunto Arduino e MatLab permite:

- Iniciar a programação imediatamente, sem quaisquer caixas de ferramentas adicionais;
- Trabalhar no MatLab para o desenvolvimento interativo e depuração;
- Interativamente desenvolver programas de aquisição de dados digitais e para controlar servomotores, etc.

5.1.2. Parâmetros de ajuste de posição dos servomotores

Durante a fixação dos braços nos eixos dos servomotores, percebeu-se a necessidade de realizar o alinhamento individual em cada um, pois os mesmos deveriam estar todos alinhados em 90º em relação ao eixo Z do ponto de referência (P_O). Para tanto fez-se um incremento angular na posição inicial de cada servomotor, ou seja, ajustou-se no programa um valor de *offset* para cada um deles, para que todos estivessem, teoricamente, na mesma posição. Este ajuste foi realizado visualmente.

5.1.3. Controle dos Servomotores

Antes de explicar o conceito aplicado para melhorar a performance do robô com relação ao caminho percorrido, cabe aqui uma breve explicação do conceito utilizado com relação à forma de controle de posição dos servomotores.

O Arduino indica as posições ao servomotor através de um sinal conhecido como PPM (Modulação proporcional de pulsos, do inglês *Pulse Proportional Modulation*). Através de um condutor, a comunicação referente ao movimento angular desejado é feita com o servo. O ângulo é determinado pela duração do pulso aplicado ao condutor.

A PPM utiliza de 1 a 2ms de um período de tempo de 20ms para codificar essa informação. O comprimento dos pulsos determinará o quanto o servomotor irá girar. Um pulso de 1,5ms irá fazer com que este gire à posição de 90º (geralmente chamada de posição neutra). Se o pulso for menor que 1,5ms o

eixo do servomotor irá girar até 0º. Se o pulso for maior que 1,5ms o eixo do servomotor irá girar até 180º.

A Figura 33 explica graficamente o que foi comentado.

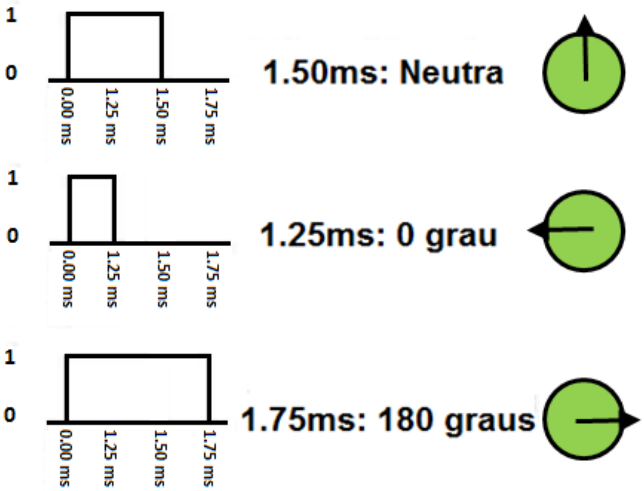


Figura 33 - Exemplo de PPM.
Fonte: www.wiki.openpilot.org. Acesso em 08/06/13.

5.1.4. Interface de Comando

A interface de comando foi implementada em ambiente GUIDE (MatLab). A mesma possui três barras de posicionamento manual (conhecida como "*Slider*"), ou seja, para ajuste das referências dos eixos x, y e z (a cinemática inversa fará com que o ângulo dos servomotores se ajuste através dos valores "setados"), assim como visualizadores das posições, possui ainda telas onde serão visualizados os tipos e quantidade de objetos detectados pela webcam.

A conexão com o Arduino já faz com que sejam geradas algumas linhas do código de programação que pode ser visto no anexo B, porém esta etapa não será abordada nesse trabalho.

Com exceção dessas etapas do programa, as linhas geradas “manualmente” podem ser vistas à seguir:

5.1.4.1.Inicialização

Este é o primeiro passo do programa. Essa etapa consiste em gerar um código para que algumas informações sejam lidas antes mesmo de se iniciar o programa, ou seja, antes do Delta se tornar visível. Dentre as etapas de tal código pode-se citar:

- Determina as funções do robô;
- “Limpa” as variáveis e informações existentes na porta serial;
- Desliga as mensagens ou avisos de erro;
- Cria variáveis;
- Delimita o ângulo máximo que os servomotores podem atingir;
- Seleciona as saídas digitais do Arduino para conexão dos servomotores;
- Informa o tamanho dos quadros onde serão verificados as imagens das peças. Este tamanho é dado em pixels;
- Habilita os botões de ativar, parar e capturar as imagens;
- Atualiza as imagens referente aos eixos;
- Configura o *threshold* (gatilho);
- Verifica dispositivos de vídeo conectados;
- Verifica se a câmera está conectada;
- Configura desenho do robô através da cinemática inversa;
- Seleciona os valores iniciais para os três eixos;
- Desenha robô;
- Cria a função de amostragens, pré-definindo um tempo entre elas (ts), etc.

Todas as etapas acima citadas podem ser verificadas no programa que está reproduzido no anexo B.

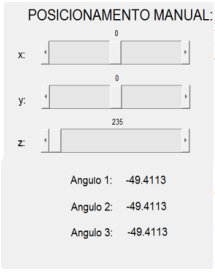
5.1.4.2.Ciclo de *Scan*

Nessa etapa há a criação de uma função de tempo. A mesma é utilizada no ciclo de *scan*, ou seja, pré-determina um tempo (ts) em que o programa para a movimentação dos servomotores será verificado, conforme abaixo:

```
function tempo(hObject, eventdata) %(source,eventdata)
global handles1 T a lim
a.servoWrite(9,round(T(1)+lim));
a.servoWrite(10,round(T(2)+lim));
a.servoWrite(11,round(T(3)+lim));
```

5.1.4.3.*Sliders*

Neste passo, cria-se e programa-se as barras de posicionamento manual (conhecidas como "*Sliders*"). As mesmas são utilizadas para ajustar as referências dos eixos x, y e z. Através desse ajuste, a cinemática inversa fará com que o ângulo de cada servomotor se altere com referência aos valores "setados" (também existem visualizadores das posições angulares). Para evitar possíveis problemas, existe uma verificação para impedir que os servomotores atinjam o ângulo limite (informado na inicialização). O programa é feito separadamente para cada eixo (x, y, e z), porém para ambos, a programação é idêntica, assim na Figura 34 pode-se verificar parte do código para o eixo x:




```
function posx_Callback(hObject, eventdata,
handles)
global vlrx vlry vlrz yB zB x y z T lim
vlrx=get(handles.posx,'Value');
set(handles.valorx,'String',num2str(vlrx));
% Cinematica Inversa
[Taux yB zB] = CalculaInversa(vlrx,vlry,vlrz);
% Verifica Limite de Angulo
if ((Taux(1)<=lim)&&(Taux(1)>=-
lim)&&(Taux(2)<=lim)&&(Taux(2)>=-
lim)&&(Taux(3)<=lim)&&(Taux(3)>=-lim))
T=Taux;
set(handles.espaco,'String','');
else
set(handles.espaco,'String','Fora do Range');
end
```

Figura 34 - Posicionamento dos Eixos.

5.1.4.4.Botões para Visualização de Imagens

Essa parte do código é utilizada para possibilitar a visualização das imagens detectadas pela webcam, isso se dá através de botões. Existe um botão para iniciar a captura das imagens, outro para parar e ainda outro para mostrar as imagens. Tanto os botões quanto o código podem ser vistos na Figura 35.

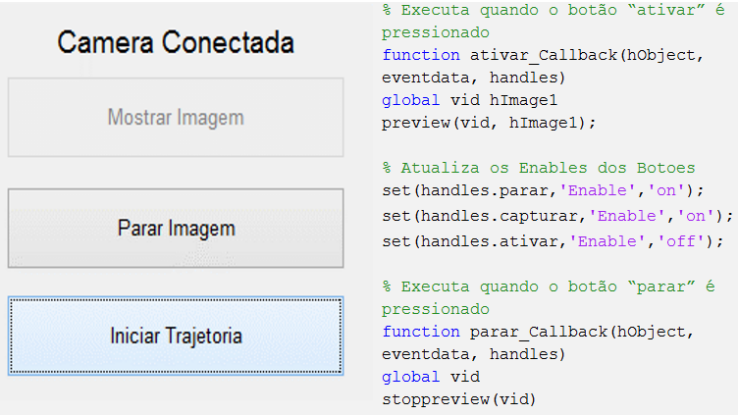


Figura 35 - Botões para Visualização de Imagens.

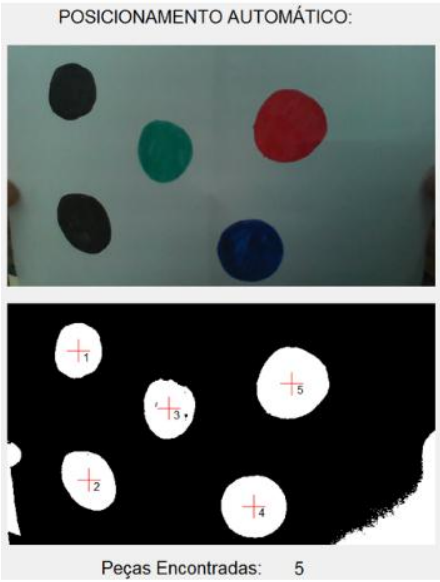
5.1.4.5.Captura de Imagens

As imagens capturadas podem ser visualizadas com essa etapa do código. Antes de se iniciar a trajetória é necessário transformar a imagem colorida (640X480X3 pixels) em uma imagem binária, ou seja, o valor de cada ponto é somente 0 ou 1 (preto ou branco). Para que se possa identificar corretamente os objetos, é conveniente que o contraste dos mesmos seja bem diferente do local (fundo) onde os mesmos estão dispostos, mesmo assim, ainda pode haver a detecção de ruído ou objetos

indesejados. Para evitar tais problemas algumas propriedades dos objetos são verificadas:

- Excentricidade;
- Área; e,
- Centroide.

Tal código ainda indica quantas peças foram identificadas e a posição cartesiana de cada uma.




```
% Executa quando o botão "capturar" é
pressionado
function capturar_Callback(hObject,
eventdata, handles)
global vid gatilho
quadro=getsnapshot(vid); % pegar um frame
axes(handles.camera2);
binquadro = im2bw(rgb2gray(quadro),
gatilho);
binquadro=~binquadro;
imshow(binquadro); hold on;
[label num]=bwlabel(binquadro,8);
graindata=regionprops(label,
'Eccentricity','Area','Centroid');
objeto=0; ptos='';
for i=1:num
if graindata(i).Eccentricity<0.7
if graindata(i).Area>1000
objeto=objeto+1;
plot(graindata(i).Centroid(1),graindata(i)
.Centroid(2),'r+','MarkerSize',20);
text(graindata(i).Centroid(1)+15,graindata
(i).Centroid(2)+20,num2str(objeto));

aux=strcat('Pto ',num2str(objeto),':
x=',num2str(graindata(i).Centroid(1)),' ,
y=',num2str(graindata(i).Centroid(2)));

ptos=strvcat(ptos,aux);
end
end
end
set(handles.qtdpecas,'String',num2str(obje
to));
set(handles.list_ptos,'String',ptos);

% Atualiza os Enables dos Botoes
set(handles.parar,'Enable','off');
set(handles.capturar,'Enable','off');
set(handles.ativar,'Enable','on');
```

Figura 36 - Telas e Programa para Captura de Imagens.

5.1.4.6.Threshold

Identificar partes de interesse em uma imagem é uma das etapas mais críticas no processamento de imagens. Uma das maneiras mais simples e eficazes de selecionar um determinado objeto ou parte em uma cena é através da operação de *threshold*.

Threshold é um valor limite que é usado em um critério de seleção. Todos os pixels de uma imagem são comparados a esse critério de seleção e são alterados conforme a necessidade. Um exemplo de aplicação de *threshold* é comparar todos os pixels a um valor limite e caso sejam maiores, igualar a um e caso sejam menores, igualar a zero. Desta forma destaca-se uma região da imagem. Comumente a operação de *threshold* gera imagens binárias ou em níveis de cinza.

Para tanto, na interface de comando há um *slider* para executar dessa função, sendo que o mesmo é descrito no programa como gatilho. O código e o S*lider* referente à esta parte do programa podem ser vistos na Figura 37.


```
function
slidergatilho_Callback(hObject,
eventdata, handles)
global gatilho
gatilho=get(handles.slidergatilh
o,'Value');
set(handles.vlrgatilho,'String',
num2str(gatilho));
```

Figura 37 - Gatilho de Imagens.

6. CONSIDERAÇÕES FINAIS

6.1. Conclusão

Neste trabalho de conclusão de curso foi desenvolvido um robô paralelo em configuração Delta com o intuito de executar funções de pick-and-place no espaço bidimensional (plano), através do reconhecimento de objetos por imagem.

Durante o desenvolvimento do robô Delta foram empenhados esforços a fim de cumprir os objetivos propostos na seção 1.1. Com o resultado final obtido, verificou-se que estes objetivos foram atingidos de forma satisfatória dentro das limitações do projeto, devido a fatores internos e externos.

Deste modo, pesquisando em vídeos, imagens e projetos de robôs já construídos, algumas ideias puderam ser aproveitadas e adaptadas às necessidades do protótipo que foi construído. Necessidades essas como, materiais e componentes disponíveis no mercado.

Por ser apenas um protótipo, o custo de fabricação de uma unidade torna-se elevado, mesmo assim, é menor que o custo dos modelos comerciais. Caso o protótipo Delta fosse transformado em um produto, os custos de desenvolvimento poderiam ser reduzidos com produção em série, com alterações de materiais e até mesmo de fornecedores e escala de produção.

Esta diferença de custo em relação aos modelos comerciais pode ser explicada pela variação da qualidade dos materiais utilizados, variabilidade na forma de sensoriamento e na forma de controle, bem como pelo valor de mercado e retorno do custo de desenvolvimento.

Diante da limitação de servomotores no mercado nacional e também pelo alto custo, optou-se por adquirir tais componentes no mercado internacional. Todavia, durante os testes, dois de quatro servomotores (três para o projeto e mais um reserva) apresentaram problemas, o que levou o grupo à solicitar novos exemplares, porém ficou-se à mercê de um novo prazo de entrega.

Diante de tal problemática, a equipe juntamente com o professor orientador optaram pela utilização do primeiro pré-

protótipo criado para verificação da viabilidade do projeto, sendo assim, todos os programas gerados para o projeto podem ser implementados e testados temporariamente.

Os resultados obtidos comprovam que mesmo em curtos prazos para execução de tarefas que exigem grande aplicação não só de conhecimento acadêmico, mas também de novos conceitos, é possível obter bons resultados.

6.2. Oportunidades de Melhoria

A cinemática diferencial não foi abordada neste trabalho devido ao fato de não ter sido feito o controle de velocidade. Para tal controle, é necessário o uso de *encoders* em todos os servomotores de movimentação. Tal feito pode ser implementado como uma oportunidade de melhoria em trabalhos futuros.

Da mesma forma, pode ser considerada como objeto de estudo futuro a cinemática direta, que é usada para o dimensionamento do espaço de trabalho. Tal implementação possibilitará a inclusão de proteções laterais para garantir a segurança do robô e de quem irá operá-lo.

Pode-se ainda realizar uma melhoria na rigidez do protótipo, substituindo peças confeccionadas em lexan por outras em alumínio ou similares.

REFERÊNCIAS

AGOSTINI, N. **Automação Robotizada**. Rio do Sul: Sibratec, 2012. 18 p..

ANGELES, J. GOSSELIN, C. M. **The optimum kinematic design of a planar three-degree-of-freedom parallel manipulator**. Journal of Mechanisms, Transmissions and Automation in Design, v. 110, nº1, p 35-41, 1988.

ATMEL CORPORATION., ***8 Bit AVR Microcontroller with 8K Bytes In-System Programmable Flash AT90S8515***. Disponível em: http://www.atmel.com. Acesso em: 14 jun. 2013.

AZEVEDO, S. AGLAÉ, A. PITTA. R. **Introdução à Robótica Educacional**. Natal: Roboeduc, 2009. 41p.

BACK, N. OGLIARI, A. DIAS, A. SILVA, J. C. **Projeto Integrado de Produtos.** São Paulo: Manole, 2008. 601 p.

BANZI, M. **Primeiros Passos Com o Arduino**. São Paulo: Novatec, 2011, 1 p.

BAPTISTA, L. F., **Manual de Introdução ao MatLab/SIMULINK**. Lisboa: Enidh - Escola Superior Náutica - Paço D'Arcos, 2008. 29 p.

BONEV I. A., GOSSELIN, C. M., **A Geometric Algorithm for the Computation of the Constant-Orientation Workspace of 6-RUS Parallel Manipulators**, ASME 2000, Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Canadá, 2000. 10 p.

BONEV I. A., RYU, J., **A new Method for Solving the Direct Kinematics of General 6-6 Stewart Platforms using three Linear extra Sensors**, Mechanism and Machine Theory, Elsevier Science, 1999. 14 p.

CARRARA, V., **Apostila de Robótica**. São Paulo: Universidade Braz Cubas, 2008. 81 p.

CARVALHO, F. **Astrofotografia com Webcams**. Disponível em: <http://www.cyberplocos.com.br/artigos/astrofotografia-com-webcams.html>. Acesso em: 08. Jun. 2013.

CHAPMAN, S.J.; **Programação em MatLab para engenheiros**. Tradução técnica: Flávio Soares Correa da Silva, São Paulo, Thomson Learning, 2006. 482 p.

CETINKUNT, S. **Mecatrônica**. Rio de Janeiro: LTC, 2008. 554 p.

COMPANY, O., PIERROT, F. **A New 3T-1R Parallel Robot**, In ICAR, Tóquio, 1999. 6 p.

DASGUPTA B., MRUTHYUNJAYA, T.S. The Stewart Platform: a Review, **Mechanical and Machinery Theory**, vol. 35, p. 15-40, 2000

DE MASI, D., **A Sociedade Pós-Industrial,** Editora SENAI, 4 ed., São Paulo, 2003. 447 p.

DESHMUKH, A. V. **Microcontrollers – Theory And Applications**. Noida, UP, India: Tata McGraw Hill, 2005. 4 p.

FINOTTI, G., **Cálculo explícito dos torques dos atuadores de um robô paralelo plano empregando o método de Kane**. 2008. 199 f.. Dissertação (Mestrado em engenharia) – Escola Politécnica da Universidade de São Paulo, USP, São Paulo.

FONSECA, J. O, TAVARES, J. M. R. S**., Introdução ao *SolidWorks*: Funcionalidades Básicas**, Universidade do Porto, 2012. 42 p.

FU, K.S., GONZALES, R.C., LEE, C.S.G., **Robotics - Control, Sensing, Vision and Intelligence**, McGraw-Hill Book Inc., International Edition, 1 ed., New York, 1987. 580 p.

GONÇALVES, R. S., **Estudo de rigidez de cadeias cinemáticas fechadas**. 2009. 263 f.. Tese (Doutorado em Engenharia Mecânica) – Faculdade de Engrenharia, Universidade federal de Uberlândia, Uberlândia.

GOSSELIN, C.; ANGELES, J. **Singularity analysis of closed-loop kinematic chains**. [S.l.]: IEEE Transactions on Robotics and Automation nº. 3, 281-290 p. 1990.

GOUGH, V.E., WHITEHALL, S.G., 1962, Universal Tire Test Machine. **9th Int. Technical Congress FISITA.**, v. 117, Maio, 1962. 21 p.

GROOVER, M. P. WEISS, M. NAGEL, R. N. ODREY, N. G. **Industrial Robotics: Technology, Programming, and Applications**. McGraw-Hill Higher Education, 1986. 546 p.

GWINNETT, J.E. Amusement Device, United States Patent Nº US1789680, 1931.

HESS-COELHO, T. A. **Topologia, síntese e análise de uma estrutura cinemática paralela**. 2005. 63 f. Dissertação (Mestrado em engenharia) – Escola Politécnica da Universidade de São Paulo, USP, São Paulo.

KAREL CAPEK WEBSITE, Disponível em http://capek.misto.cz/english/. Acesso em 18. Maio. 2013.

KLAFTER, R., CHMIELEWSKI, T., NEGIN, M., **Robotic Engineering, an Integrated Approach**, Prentice-Hall International Editions, London, 1989. 744 p.

KONG, X., GOSSELIN, C., **Type Synthesis of Parallel Mechanisms**, Springer Tracts in Advanced Robotics, vol. 33, 2007. 274 p.

LOPES, A. M., **Manipuladores de Estrutura Paralela**. Porto: Universidade do Porto, 2002. 131 p.

MARGOLIS, M. **Arduino Cookbook**. Sebastopol, CA, USA: O' RilleyMedia, 2011, 724 p.

MELLO, C. B., **Controle de Trajetória de uma Plataforma Stewart para Simulação de Transferência de Carga Fora de Porto**. 2011. 131 f.. Dissertação (Mestrado em Engenharia Mecânica) - Programa de Pós-graduação em Engenharia Mecânica, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

MERLET, J. P. **Parallel Robots (Solid Mechanics and Its Applications).** Netherlands: Springer, 2006. 380 p.

MOLINA, F. A. L., **Ambiente de Simulação de Manipuladores Paralelos: Modelagem, Simulação e Controle de uma Plataforma Stewart,** 2008. 171 f.. Dissertação (Mestrado em Engenharia Mecânica) - Universidade Estadual de Campinas, Campinas, SP.

MÜLLER, A. A., **O uso da Modelagem e da Cossimulação no Projeto de Sistemas Mecatrônicos com Ênfase no Desenvolvimento de um Robô**. 2012. 118 p.. Dissertação (Mestrado Profissional em Mecatrônica) - Instituto Federal De Educação Ciência e Tecnologia de Santa Catarina, Florianópolis.

OLSSON, A., **Modeling and control of a Delta-3 Robot**. Lund, Sweden: Department of Automatic Control, Lund University, 2009. 73 p.

OTTOBONI, A., **Servo - acionamentos.** Mecatrônica Atual, São Paulo, nº 6, p. 7- 14, out, 2002.

PIRES, J. N., **Das Máquinas Gregas à Moderna Robótica Industrial.** Coimbra, Portugal: Departamento de Mecânica, Universidade de Coimbra, 2002. 18 p.

RIVIN, E. I., ***Mechanical Design of Robots***, 1 ed., McGraw-Hill Inc., New York, 1988. 325 p.

ROMANO, V.F., DUTRA, M.S., **Introdução à Robótica Industrial**. Campinas: UNICAMP, 1999. 21 p.

SCHIAVICCO, L., SICILIANO, B., ***Robotica Industriale - Modellistica e Controllo di Manipolatori***, 1 ed., McGraw-Hill Inc., Milano, 1995. 393 p.

SICILIANO, B. et al. **Robotics: Modelling, Planning and Control**. Netherlands: Springer, 2009. 632 p.

SILVA, L. A. RAGUENES, N. SALTAREN, R. SEBASTIAN, J. M. ARACIL, R. **Diseño, Simulación, Análisis Cinemático y Dinámico de um robot paralelo para Control Visual de altas prestaciones**. Madri, Espanha: Universidad Politécnica de Madrid, •Escuela de Ingenieros Industriales, 2003. 41 p.

SOARES, B.F.F, SILVA, F.S., NIGRI, I., MELLO, C.B., MEGGIOLARO, M.A., **Master-Slave Servo-Bilateral Control of Direct Drive Electrical Manipulators**, 19th International Congress of Mechanical Engineering, Brasilia, DF, Brasil, 2007. 10 p.

SOUZA, A. R. de; et al. **A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC.** Revista Brasileira de Ensino de Física, São Paulo, v. 33, n. 1, 1702, fev, 2011.

STEWART, D., **A platform with 6 degrees of freedom**, Institution of Mechanical Engineers, Vol. 180, Pt. 1, n. 15, 16 p.

TANEV, T. K., **Kinematics of a Hybrid (Parallel-Serial) Robot Manipulator,** vol. 35, 2000. 1750 p.

TARTARI, S.C., **Modelagem e otimização de um robô de arquitetura paralela para aplicações industriais**. 2006. 227 f.. Dissertação (Mestrado em engenharia) – Escola Politécnica da Universidade de São Paulo, USP, São Paulo.

TAVARES, L.A., GOMES, E.L.B., **Uma solução com Arduino para controlar e monitorar processos industriais**. Instituto

Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais – Campus Pouso Alegre, 2013. 4 p.

TONINI, A.M., COUTO, B.R.G.M., **Ensinando Geometria Analítica com uso do MatLab**, Departamento de Ciências Exatas e Tecnologia do Centro Universitário de Belo Horizonte / DECET – UniBH, 2002. 49 p.

TSAI, L.W. **Robot Analysis: The Mechanics of Serial and Parallel Manipulators.** New York, USA: John Wiley and Sons, 1999. 505 p.

ANEXO A – ETAPAS DO PROJETO CONCEITUAL

O projeto conceitual deve ser definido quando algo é desenvolvido para consumo do mercado. Para tal feito, algumas etapas foram definidas:

a) Público Alvo

Primeiramente foi definido o público alvo à utilizar o Robô Delta, sendo:

- Manipulação de peças em uma esteira transportadora;
- Micro usinagem;
- Confecção de Placas de circuito Impressos;
- Medicina.

b) Requisitos

Em seguida, foram elencados os requisitos do projeto, como vê-se a seguir:

- Capacidade de Levantamento de carga;
- Alto nível de mobilidade;
- Compacto;
- Boa resolução no sistema de reconhecimento por imagem;
- Bom aspecto externo;
- Fácil interface de comando.

c) Especificações

Em seguida, algumas especificações foram definidas para atender aos requisitos:

- Carga de aproximadamente 1 kg;
- Três graus de liberdade para os braços mais um para a garra;
- Dimensões de 1000 x 780 x 780mm;
- Resolução de 1.3 MP.

d) Soluções

O próximo passo refere-se à definição de soluções para atender às especificações do projeto:

Requisitos	Soluções	Vantagens	Desvantagens
Capacidade de Levantamento de carga	Garra (elétrica)	Baixo Custo e fácil controle	Baixa capacidade de levantamento de cargas (na maioria dos casos)
	Ventosa	Ideal para superfícies lisas	Necessidade de um circuito pneumático
	Eletroímã	Ideal para materiais planos	Alto custo
Alto nível de mobilidade	Juntas esféricas	3 graus de liberdade	Menor capacidade de carga (na maioria das vezes)
	Juntas prismáticas	Ideal para movimentos ortogonais	Apenas 1 grau de liberdade
Altas acelerações e velocidades de operação	Servomotor	Elevado torque e precisão	Alto custo
	Motor CC	Bom torque e baixo escorregamento	Necessidade de manutenção de escovas
	Motor de Passo	Não usam escovas ou comutadores, pouco desgaste, dispensa realimentação	Pequeno porte, baixa velocidade comparada ao servomotor

Baixo peso	Estrutura de alumínio	Leve, custo médio, alta disponibilidade	
	Estrutura de policarbonato	Leve, disponibilidade	Custo elevado, transparência
	Estrutura de titânio	Leve	Custo muito elevado, difícil usinagem
	Estrutura de madeira	Leve, disponibilidade	Aspecto incompatível
Processamento	Arduino	Programação média, sem necessidade de construção de circuitos	Pouca versatilidade
	CLP	Programação simples	Alto consumo de energia, custo muito elevado, tamanho grande
	Microcontrolador	Custo e tamanho reduzido, facilidade de modificação	Saída serial muito limitada, rotinas matemáticas muito extensas
	Microprocessador (PIC)	Programação media	Necessidade de construção de circuitos e difícil programação
Boa resolução no sistema de reconhecimento por imagem	Webcam	Médio custo	Alcance limitado
	Câmera IP	Acesso remoto mais fácil, boa recepção wireless	Alto custo, requer banda larga
Bom aspecto externo	Perfis de Alumínio	Melhor aspecto externo, menor massa estrutural, facilidade de montagem	Alto custo
	Estrutura Tubular	Facilidade na montagem, médio preço	Pintura
	Estrutura Soldada	Boa rigidez, preço acessível	Dificuldade na montagem, pintura

Fácil interface de comando.	MatLab (GUIDE)	Programação média, grande versatilidade	Necessidade de aquisição de licenças
	Elipse Scada	Programação média	Pouca versatilidade

Tabela 5 - Soluções para o Projeto.

e) Estrutura Funcional

Para entender o funcionamento do Delta, a estrutura funcional foi desenvolvida:

Figura 38 - Estrutura Funcional – Visão global.

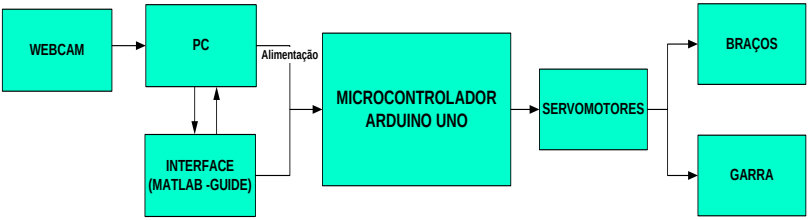


Figura 39 - Estrutura Funcional

As outras etapas do projeto, tais como detalhamento, seleção de componentes e seu dimensionamento, execução dos desenhos (modelagem, montagem, etc.) foram vistos no decorrer deste trabalho.

ANEXO B – PROGRAMAÇÃO DO ROBÔ DELTA

```
function varargout = delta(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @delta_OpeningFcn, ...
                   'gui_OutputFcn',  @delta_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% Executes just before delta is made visible.
function delta_OpeningFcn(hObject, eventdata, handles,
varargin)
clear a; delete(instrfind({'Port'},{'COM11'}));
warning off; clc;

%% Configuração Arduino
global a lim
lim=50;
a=arduino('COM11');
a.servoAttach(9); a.servoAttach(10);  a.servoAttach(11);
a.servoWrite(9,lim); a.servoWrite(10,lim);  a.servoWrite(11,lim);

%% Configuração Gráficos
% Desliga Eixos dos Graficos
global hImage1 hImage2
hImage1 = image(zeros(480, 640, 3), 'parent', handles.camera);
hImage2 = image(zeros(480, 640, 3), 'parent', handles.camera2);
```

```
axes(handles.camera); axis off;
axes(handles.camera2); axis off;

% Atualiza os Enables dos Botoes
set(handles.ativar,'Enable','off');
set(handles.parar,'Enable','off');
set(handles.capturar,'Enable','off');

% Atualiza a Figura
axis(handles.grafico,[-400 400 -600 600 -100 600]);
hold(handles.grafico);
axes(handles.grafico); axis off; %grid on;

%% Configuração Camera
global gatilho vid
gatilho=0.25;

% Verifica Dispositivos de Video Conectados
dispo=imaqhwinfo('winvideo');
ndispo=size(dispo.DeviceIDs,2);
connect=0;
for i=1:ndispo
   if (dispo.DeviceInfo(i).DeviceName(1:8)=='Logitech')
      connect=1;
      vid = videoinput('winvideo', i, 'RGB24_640x480');
%        vid = videoinput('winvideo', i, 'MJPG_640x480');
%        vid = videoinput('winvideo', i, 'YUY2_640x480');
%        vid = videoinput('winvideo', i, 'UYVY_640x480');
      break;
   end
end

if connect==1
   set(handles.camconect,'String','Camera Conectada');
   set(handles.ativar,'Enable','on'); % Habilita Preview
else
   set(handles.camconect,'String','Camera Não Conectada');
end

%% Configuração Desenho Robô
```

```
global handles1 ts t vlrx vlry vlrz T
global Lbase Lmancal Lbraco Lante Lmesa T1 T2 yB zB x y z
T(1)=0; T(2)=0; T(3)=0; yB=[357.7732  357.7732  357.7732]; zB=-
115.0492 -115.0492 -115.0492;
Lbase=180; Lmancal=-20; Lbraco=240; Lante=490; Lmesa=115;
T1=[cos(deg2rad(120))   -sin(deg2rad(120))  0   0;
  sin(deg2rad(120))    cos(deg2rad(120))   0   0;
      0                  0          1   0;
      0                  0          0   1];
T2=[cos(deg2rad(-120))   -sin(deg2rad(-120))   0   0;
  sin(deg2rad(-120))    cos(deg2rad(-120))   0   0;
      0                  0           1   0;
      0                  0           0   1];
yB=0; zB=0; x=0; y=0; z=0;

%Condição Inicial
vlrx=0; vlry=0; vlrz=235;

% Cinematica Inversa
[Taux yB zB] = CalculaInversa(vlrx,vlry,vlrz);
% Verifica Limite de Angulo
if ((Taux(1)<=lim)&&(Taux(1)>=-
lim)&&(Taux(2)<=lim)&&(Taux(2)>=-
lim)&&(Taux(3)<=lim)&&(Taux(3)>=-lim))
   T=Taux;
   set(handles.espaco,'String','');
else
   set(handles.espaco,'String','Fora do Range');
end

% Desenha Robo
desenha(hObject, eventdata, handles);
set(handles.ang1,'String',num2str(T(1)));
set(handles.ang2,'String',num2str(T(2)));
set(handles.ang3,'String',num2str(T(3)));

% Cria handles para função timer
handles1 = handles;
handles.output = hObject;
guidata(hObject, handles);
```

```
% Cria a função de Amostragem
ts=0.01;
t = timer('TimerFcn',@tiempo, 'ExecutionMode', 'fixedSpacing'
,'BusyMode', 'queue' ,'Period', ts);
start (t);

% --- Outputs from this function are returned to the command line.
function varargout = delta_OutputFcn(hObject, eventdata,
handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

 % Função de Tempo, executada a cada "ts" segundos
function tiempo(hObject, eventdata) %(source,eventdata)
global handles1 T a lim
a.servoWrite(9,round(T(1)+lim));
a.servoWrite(10,round(T(2)+lim));
a.servoWrite(11,round(T(3)+lim));

function posx_Callback(hObject, eventdata, handles)
global vlrx vlry vlrz yB zB x y z T lim
vlrx=get(handles.posx,'Value');
set(handles.valorx,'String',num2str(vlrx));

% Cinematica Inversa
[Taux yB zB] = CalculaInversa(vlrx,vlry,vlrz);
% Verifica Limite de Angulo
if ((Taux(1)<=lim)&&(Taux(1)>=-
lim)&&(Taux(2)<=lim)&&(Taux(2)>=-
lim)&&(Taux(3)<=lim)&&(Taux(3)>=-lim))
    T=Taux;
    set(handles.espaco,'String','');
else
    set(handles.espaco,'String','Fora do Range');
end

% Desenha Robo
desenha(hObject, eventdata, handles);
```

```
set(handles.ang1,'String',num2str(T(1)));
set(handles.ang2,'String',num2str(T(2)));
set(handles.ang3,'String',num2str(T(3)));

 % --- Executes on slider movement.
function posy_Callback(hObject, eventdata, handles)
global vlrx vlry vlrz yB zB x y z T lim
vlry=get(handles.posy,'Value');
set(handles.valory,'String',num2str(vlry));

% Cinematica Inversa
[Taux yB zB] = CalculaInversa(vlrx,vlry,vlrz);
% Verifica Limite de Angulo
if ((Taux(1)<=lim)&&(Taux(1)>=-
lim)&&(Taux(2)<=lim)&&(Taux(2)>=-
lim)&&(Taux(3)<=lim)&&(Taux(3)>=-lim))
   T=Taux;
   set(handles.espaco,'String','');
else
   set(handles.espaco,'String','Fora do Range');
end

% Desenha Robo
desenha(hObject, eventdata, handles);
set(handles.ang1,'String',num2str(T(1)));
set(handles.ang2,'String',num2str(T(2)));
set(handles.ang3,'String',num2str(T(3)));

 % --- Executes on slider movement.
function posz_Callback(hObject, eventdata, handles)
global vlrx vlry vlrz yB zB x y z T lim
vlrz=get(handles.posz,'Value');
set(handles.valorz,'String',num2str(vlrz));

% Cinematica Inversa
[Taux yB zB] = CalculaInversa(vlrx,vlry,vlrz);
% Verifica Limite de Angulo
if ((Taux(1)<=lim)&&(Taux(1)>=-
lim)&&(Taux(2)<=lim)&&(Taux(2)>=-
lim)&&(Taux(3)<=lim)&&(Taux(3)>=-lim))
```

```
    T=Taux;
    set(handles.espaco,'String','');
else
    set(handles.espaco,'String','Fora do Range');
end

% Desenha Robo
desenha(hObject, eventdata, handles);
set(handles.ang1,'String',num2str(T(1)));
set(handles.ang2,'String',num2str(T(2)));
set(handles.ang3,'String',num2str(T(3)));

 % --- Executes during object creation, after setting all properties.
function posx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to posx (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

 % --- Executes during object creation, after setting all properties.
function posy_CreateFcn(hObject, eventdata, handles)
% hObject    handle to posy (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
% --- Executes during object creation, after setting all properties.
function posz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to posz (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes during object deletion, before destroying
properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
global t
stop (t); delete(t); delete(instrfind({'Port'},{'COM11'}));

% --- Executes on button press in ativar.
function ativar_Callback(hObject, eventdata, handles)
global vid hImage1
preview(vid, hImage1);

% Atualiza os Enables dos Botoes
set(handles.parar,'Enable','on');
set(handles.capturar,'Enable','on');
set(handles.ativar,'Enable','off');

% --- Executes on button press in parar.
function parar_Callback(hObject, eventdata, handles)
global vid
stoppreview(vid)

 % --- Executes on button press in capturar.
function capturar_Callback(hObject, eventdata, handles)
global vid gatilho
quadro=getsnapshot(vid); % pegar um frame
axes(handles.camera2);
```

```
binquadro = im2bw(rgb2gray(quadro), gatilho);
binquadro=~binquadro;
imshow(binquadro); hold on;
[label num]=bwlabel(binquadro,8);
graindata=regionprops(label, 'Eccentricity','Area','Centroid');
objeto=0; ptos='';
for i=1:num
  if graindata(i).Eccentricity<0.7
     if graindata(i).Area>1000
        objeto=objeto+1;

plot(graindata(i).Centroid(1),graindata(i).Centroid(2),'r+','MarkerSi
ze',20);

text(graindata(i).Centroid(1)+15,graindata(i).Centroid(2)+20,num2
str(objeto));
        aux=strcat('Pto ',num2str(objeto),':
x=',num2str(graindata(i).Centroid(1)),' ,
y=',num2str(graindata(i).Centroid(2)));
        ptos=strvcat(ptos,aux);
     end
  end
end
set(handles.qtdpecas,'String',num2str(objeto));
set(handles.list_ptos,'String',ptos);

% Atualiza os Enables dos Botoes
set(handles.parar,'Enable','off');
set(handles.capturar,'Enable','off');
set(handles.ativar,'Enable','on');

% --- Executes on slider movement.
function slidergatilho_Callback(hObject, eventdata, handles)
global gatilho
gatilho=get(handles.slidergatilho,'Value');
set(handles.vlrgatilho,'String',num2str(gatilho));


 % --- Executes during object creation, after setting all properties.
function slidergatilho_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to slidergatilho (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

 % --- Executes on selection change in list_ptos.
function list_ptos_Callback(hObject, eventdata, handles)
% hObject    handle to list_ptos (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns list_ptos
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
list_ptos

% --- Executes during object creation, after setting all properties.
function list_ptos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to list_ptos (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: listbox controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

ANEXO C – IMAGENS DO PROTÓTIPO

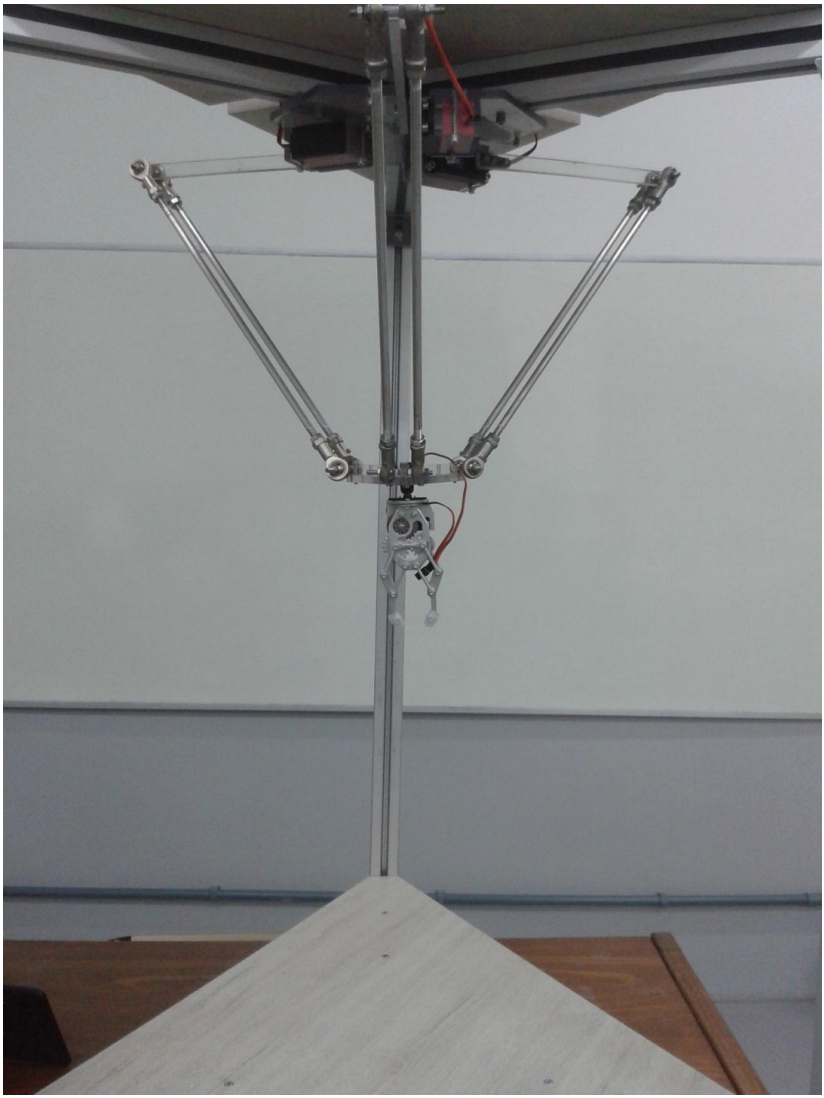


Figura 40 - Manipulador Robótico

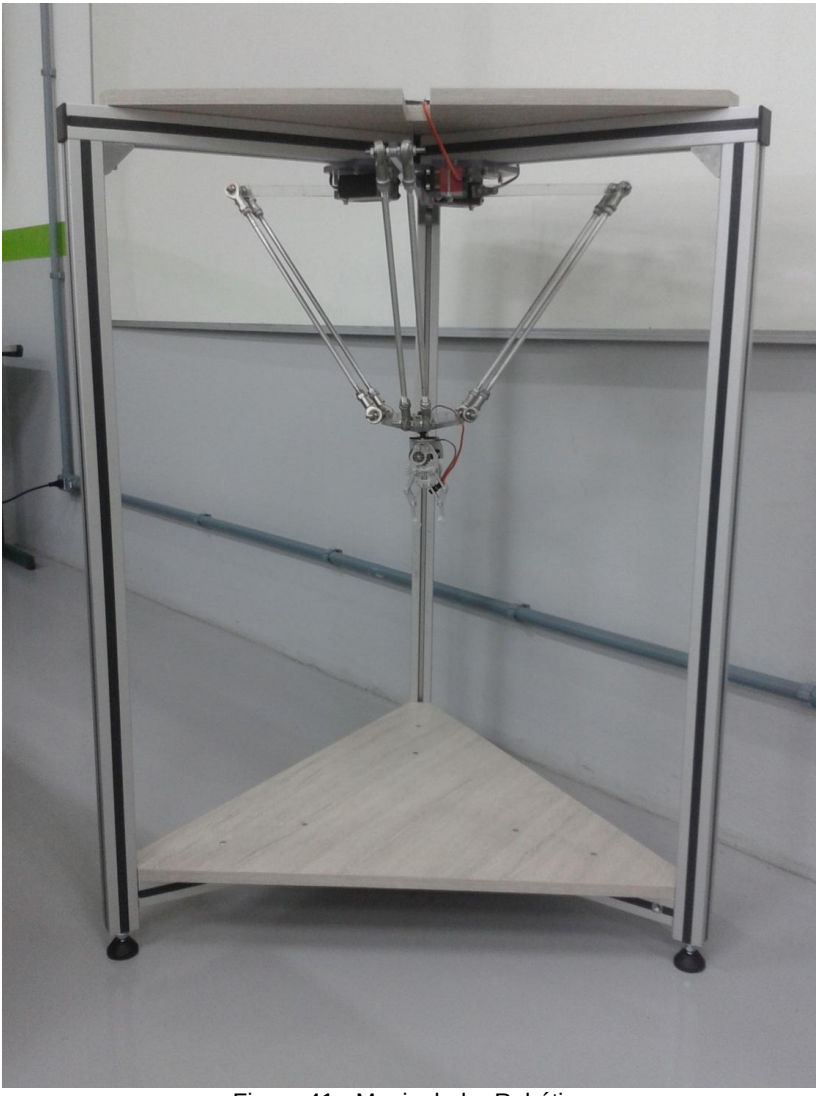


Figura 41 - Manipulador Robótico

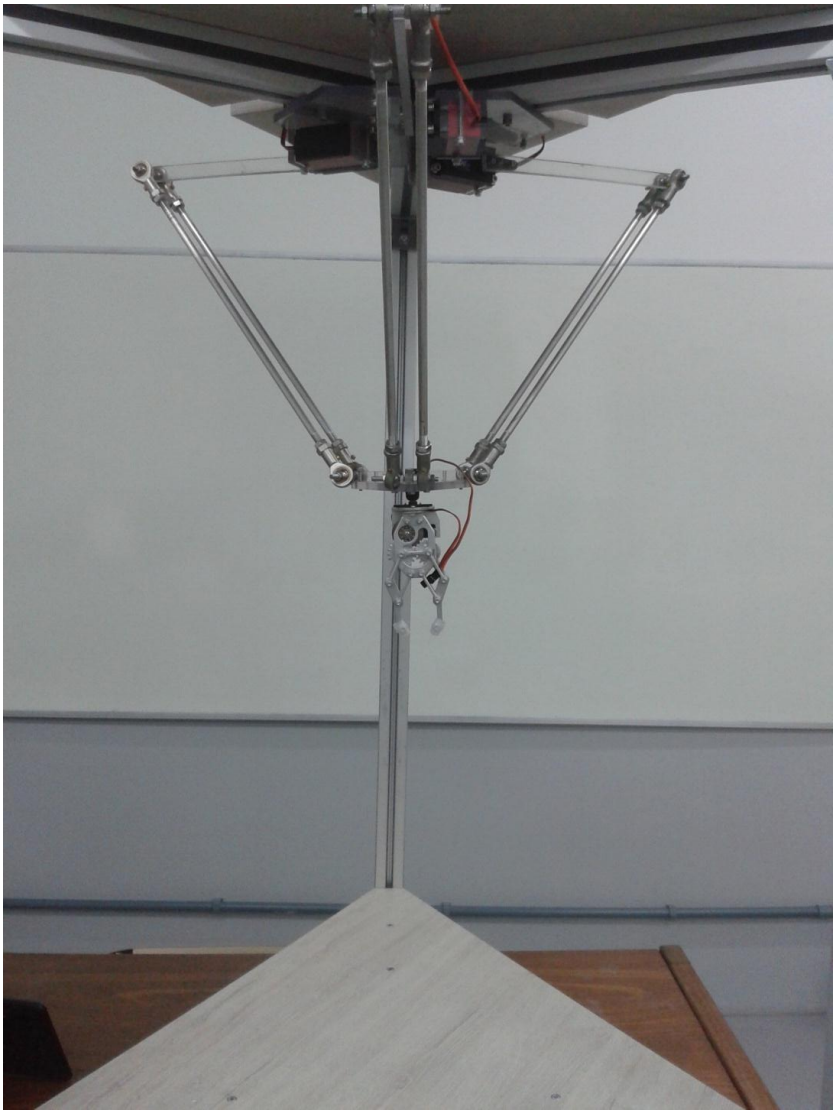


Figura 42 - Manipulador Robótico

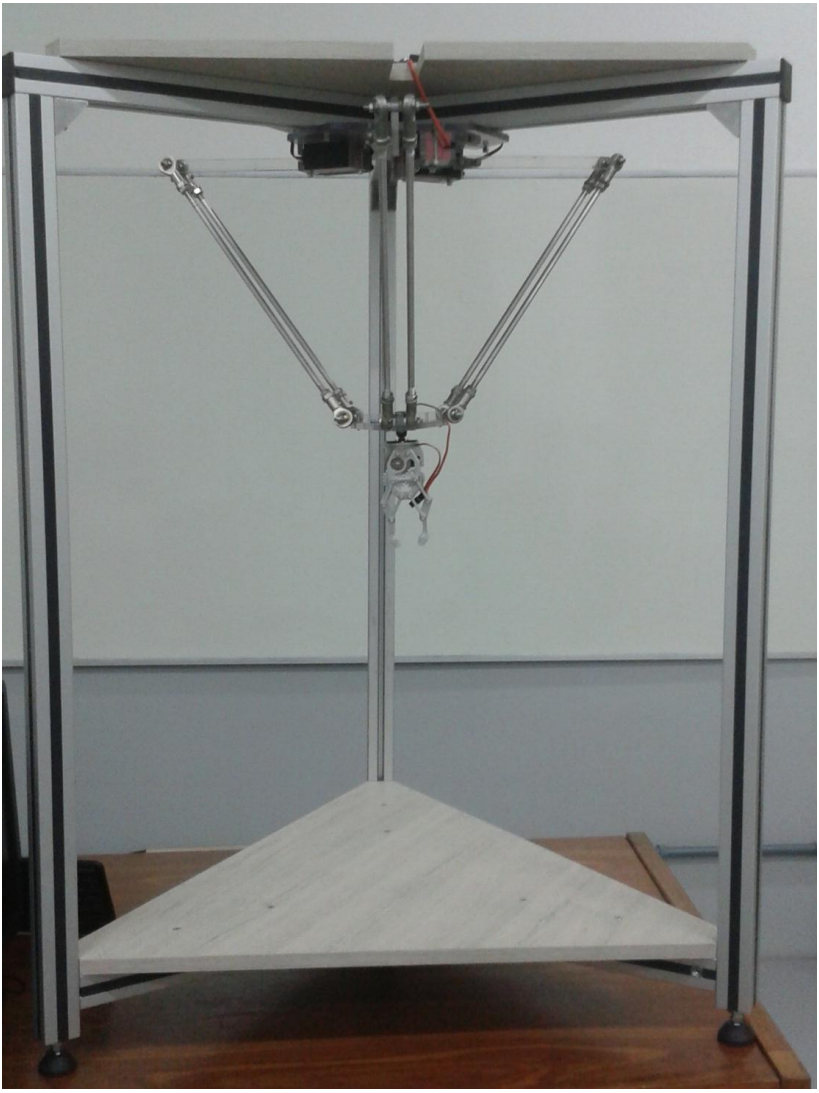


Figura 43 - Manipulador Robótico