UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

RAPHAEL SANTANA GALDINO

DESENVOLVIMENTO DE UM BRAÇO ROBÓTICO BASEADO NO ROBÔ SCARA UTILIZANDO O MODEL BASED DESIGN

Campina Grande, Paraíba
Dezembro de 2019

RAPHAEL SANTANA GALDINO

DESENVOLVIMENTO DE UM BRAÇO ROBÓTICO BASEADO NO ROBÔ SCARA UTILIZANDO O MODEL BASED DESIGN

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientador: Prof. Rafael Bezerra Correia Lima, D.Sc.

Campina Grande, Paraíba
Dezembro de 2019

RAPHAEL SANTANA GALDINO

DESENVOLVIMENTO DE UM BRAÇO ROBÓTICO BASEADO NO ROBÔ SCARA UTILIZANDO O MODEL BASED DESIGN

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Aprovado em _____/_____/_____

_____

Prof. George Aciole Junior, D. Sc.
Universidade Federal de Campina Grande
Avaliador, UFCG

_____

Prof. Rafael Bezerra Correia Lima, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Campina Grande, Paraíba
Dezembro de 2019

Dedico este trabalho a Deus. Sem Ele nada seria possível.

AGRADECIMENTOS

A Deus, sobre tudo e sobre todos, pois a Ele tudo devo e sou eternamente grato.

A minha família, meus pais Adriana e Erivanildo, e meus irmãos Gabriel e Edval, que fizeram o possível para me ajudar nessa jornada, e sempre estiveram ao meu lado não importando os problemas.

A minha namorada Amanda Villarim, que não mediu esforços para me ajudar e auxiliar nessa jornada, sempre me motivando e me apoiando, tornando esse momento mais leve.

A Vitor Rafael, um grande amigo que esse projeto me deu. Por todos os problemas que esse robô nos proporcionou e juntos conseguimos resolver.

Ao nosso professor orientador Rafael, que com sua paciência, sabedoria e conhecimento conseguiu nos ensinar sermos profissionais excelentes.

Aos meus amigos que a graduação me proporcionou, Jorge Luíz, Samuel de Melo, Vítor Ramos, Matheus Braga, Mylena Karla, Joyce Morais, Leonardo Magno, Arthur Felipe, Júlio César, Thiago Henriques, Leonardo Pereira, Anderson Wendel, Ariôsto Júnior, Marina Lua, Ulisses Gomes, Alexsandro Barros, Raphael Victor, Matheus Guerra aos meus amigos do LIEC e tantos outros que não couberam nessa lista. Sem vocês não teria conseguido, muito obrigado por todos os momentos compartilhados.

RESUMO

As necessidades por redução de custos e aumento na eficiência são uma realidade no setor produtivo, sendo a robótica uma área cada vez mais fundamental na indústria 4.0. A metodologia focando no modelo para o desenvolvimento de um projeto, o Model-Based Design (MBD), e o uso de simulações computacionais são alguns modos de testar, gerenciar e desenvolver sistemas complexos. Neste cenário, torna-se de extrema importância que o ensino em engenharia explore novas ferramentas de ensino. Desta forma, neste trabalho é abordado o emprego do MBD ao desenvolvimento de um robô Scara. São apresentados a modelagem da cinemática do robô, além de aspectos relacionados a simulação e projeto utilizando a metodologia Model-Based Design com a biblioteca Simscape do Simulink. Foi realizado ainda, comparativos entre o modelo e o robô construído foram feitos, além da criação de uma interface gráfica no ambiente App Designer. Ao fim do projeto, foi utilizado do software Inventor para criação da estrutura da simulação com o objetivo de tornar a simulação mais próxima do braço projetado.

Palavras-chave: Robótica. Model-Based Design. Simscape. MATLAB. Motor AX-12A. Scara.

ABSTRACT

The needs to reduce costs and increase efficiency are the reality in the productive sector, being robotic field increasingly fundamental in the industry 4.0. The methodology focused on the design model, Model-Based Design (MBD), and the use of computational simulations are some ways of testing, conducting and developing complex systems. In this scenario, it becomes extreme importance for engineering teaching to explore new teaching instruments. In this way, this work addresses the use of MBD with the development of a Scara robot. The modeling of the kinematics of the robot, in addition to aspects related to simulation and design using the Model-Based Design methodology with Simulink's Simulink library. It was also perfomed, comparatives between the model and the built robot were made, and a graphical interface was created in the App Designer environment. At the end of the Project, was used the Inventor software to create the simulation structure in order to make the simulation closer to the designed arm.

Keywords: Robotic. Model-Based Design. Simscape, MATLAB. AX-12A Engine. Scara.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

Lista de Abreviaturas e Siglas

RAM	*Random Access Memory*
GUI	*Graphical User Interface*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
ID	*Identity*
MBD	*Model-Based Design*

SUMÁRIO

1. INTRODUÇÃO

A globalização impõe novos desafios à indústria, pois acirra ainda mais a competição, aumenta a pressão por produtos mais baratos e individuais, que usam tecnologias cada vez mais complexas. A solução que a indústria 4.0 propõe é um sistema de produção mais flexível, com altos níveis de conectividade e automação (RUSSWURM, 2014). Tendo em vista este cenário, é incontestável que a formação em engenharia deve abordar as tendências de modernização dos sistemas de produção, objetivando formar e capacitar profissionais preparados para lidarem com os novos paradigmas, sobretudo no que diz respeito ao projeto, desenvolvimento e implementação de sistemas robóticos.

Sistemas robóticos são cada vez mais empregados no ambiente industrial, pois são capazes de reduzirem custos e aumentarem os níveis de produtividade. Contudo, robôs são máquinas complexas e muitas vezes caras, tornando muitas vezes inviável a construção de protótipos para testes e para o ensino em universidades e escolas de engenharia. Deste modo, o emprego de modelos computacionais, simulação e plataformas de ensino são de extrema importância para a formação do engenheiro em conformidade com o perfil do profissional da indústria 4.0.

Uma metodologia que pode ser aplicada para o desenvolvimento de novas tecnologias, incluindo robôs, é o *Model-Based Design* (MBD). O MBD é uma abordagem centrada no modelo para o desenvolvimento de sistemas dinâmicos e complexos. Por não necessitar de protótipos para o teste e implementação de um produto, esta abordagem reduz consideravelmente os custos, podendo ainda ser utilizado para fins didáticos (AARENSTRUP, 2015).

Com a visão de tornar o projeto mais didático e mais harmonioso, é utilizado do *software Autodesk Inventor*, que permite um maior detalhamento na construção da simulação em comparação com o *Simscape*.

1.1 Objetivos

O objetivo deste trabalho é descrever o desenvolvimento de uma plataforma de ensino em robótica baseado no robô de montagem Scara sob a perspectiva do *Model-Based Design*, com o uso de visão computacional. Para tanto, empregou-se as ferramentas disponíveis no ambiente *Simulink*, com ênfase na *toolbox Simscape*. O *Simscape* é uma biblioteca de blocos para modelagem de sistemas dinâmicos que fornece simulações por meio de redes físicas.

O conteúdo deste trabalho está dividido da seguinte forma. A seção 2 descreve a constituição do robô Scara construído, detalhando os componentes utilizados. A seção 3 introduz conceitos de visão computacional. A seção 4 conceitua brevemente *Model-Based Design*, citando ainda a modelagem cinemática do braço. A seção 5 detalha sobre o *V-Model*, informando como foi utilizada no projeto. A seção 6 explica como foi formada a interface do projeto e cita quais critérios foram utilizados para sua criação. A seção 7 detalha sobre o *Simscape*, visando explicar a união da parte de controle com a parte física. A seção 8 finaliza a fundamentação teórica com a utilização do *software Autodesk Inventor* no projeto. A seção 9 analisa os resultados obtidos experimentalmente, comparando-os com os resultados verificados no modelo. A seção 10 contém as considerações finais do trabalho

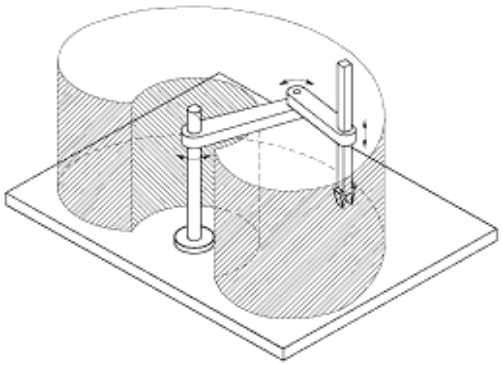
2. CONSTITUIÇÃO DO ROBÔ

2.1 Robo Scara

O robô *Scara* foi desenvolvido pela primeira vez no japão na década de 80. *Scara* é o acrônimo, em inglês, para *Selective Compliance Assembly Robot Arm*, e significa “braço robótico para montagem de conformidade seletiva”. Rápidos e compactos, os robôs *Scara* são versáteis, apresentando bom desempenho em operações de montagem de componentes de pequenas dimensões e transferência de peças entre células (MAKINO, 1982).

Geralmente, este tipo do robô possui um manipulador com duas juntas de revolução dispostas em paralelo, responsáveis pelo movimento de rotação, visando reproduzir o braço humano; podendo, ainda, dispor de uma junta prismática perpendicular a esse plano, responsável pelos movimentos lineares. Por tais características, equipamentos desta espécie possuem um volume de trabalho aproximadamente cilíndrico, conforme ilustrado na Figura 1.

Figura 1 - Volume de trabalho típico de um robô S*cara*



O robô *Scara* é mais utilizado em operações de *pick-and-place* ou montagem, pois essas aplicações necessitam de alta velocidade e alta precisão. Já a desvantagem do robô *Scara* é não possuir capacidade de suportar cargas elevadas, devido a sua configuração construtiva.

2.2 Características físicas

A estrutura física do robô *Scara* construído, assemelha-se à apresentada na Figura 2. A coluna central do braço é composta por um tripé de aço, fixada ao primeiro elo com o auxílio de um suporte plástico fornecido pelo fabricante dos motores. Cada elo é feito de duas placas de fibra de vidro, cada uma com 17 cm de comprimento e 2,5 cm de largura. Além disso, estão presentes nas juntas de rotação atuadores AX-12A. Em razão de aspectos construtivos do robô, os motores possuem limitações quanto ao seu movimento, ficando a primeira e a segunda junta restritas a rotacionar 102º e 150º, respectivamente.

O robô contém ainda uma câmera USB *Dlink* DSB C-120, posicionada acima do braço utilizando um suporte em L invertido em aço, com 25 cm de comprimento e 52 cm de altura. É mostrada na Figura 2 a estrutura física do robô *Scara.*

Figura 2 - Estrutura física do robô

2.3 Motores AX-12A

O servo motor AX-12A é um atuador inteligente, que integra um redutor de velocidade, um motor de precisão e um circuito de controle com a funcionalidade de rede, trabalhando em um único conjunto. De acordo com o manual fornecido pela ROBOTICS, empresa sul coreana que fabrica este dispositivo, o motor AX-12A possui um peso de 54.6g, com dimensões de 32mm x 50 mm x 40 mm. Estas e outras características principais do AX-12A encontram-se na Tabela 1.

Tabela 1 – Especificações sobre o motor AX-12A

Informação	Valor
Dimensões (mm)	32 x 52 x 40
Peso (g)	54.6
Taxa de redução	254:1
Torque máx. (N·m)	1.5
Tensão (V)	9 a 12
Faixa de operação (°)	0 a 300
Corrente máx. (mA)	900
Resolução (°)	0.293
Temperatura de operação (°C)	-5 a 85
Velocidade de comunicação (Mbps)	0.7343 a 1
Tipo de protocolo	*Half duplex Asynchronous Serial Communication*
Número de IDs	254 (0 a 253 ID)

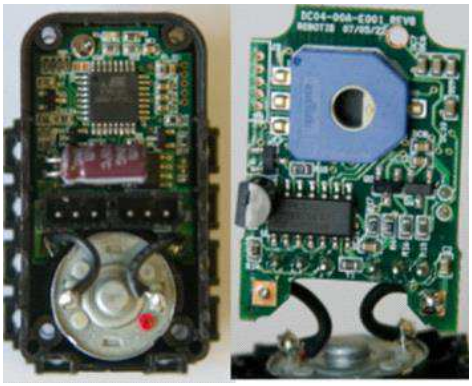
Fonte: ROBOTICS

Capaz de fornecer um torque elevado em comparação às suas proporções compactas, o motor é fabricado com materiais de alta qualidade,

conferindo-lhe a resistência e a robustez estrutural necessárias para suportar grandes forças externas para a sua dimensão. Além disso, detém a capacidade de detectar e reagir a condições internas, como mudanças na temperatura interna ou tensão de alimentação, fornecendo também uma leitura precisa do ângulo de rotação do seu eixo, com uma precisão de 0.293°, devido aos 1024 níveis de controle, variando o ângulo de 0° a 300°.

Desmontando o motor é possível observar 3 componentes essenciais, como pode-se observar na Figura 3:

Figura 3 – Componentes internos do motor AX-12A



-Motor: Responsável por gerar trabalho, conectado ao conjunto de engrenagens para aumentar o torque.

-Potenciômetro: Sensor necessário para realizar o monitoramento da posição.

-Circuito de controle: Responsável por acionar o motor quando necessário, depende da posição do potenciômetro.

Os três componentes acima explicados são dependentes um do outro, porque cada um possui função primordial no servo motor AX-12A.

O sensor mais utilizado para esse tipo de aplicação é o *murata* SV01, mostrado na Figura 4. Esse tipo de sensor de posição é um potenciômetro, com sua resistência até 10kΩ. Utilizando o valor de tensão entre os terminais do potenciômetro e aplicando sua proporcionalidade com a rotação do rotor, é possível encontrar a posição angular do eixo do motor. A tecnologia utilizada nesse componente está dominada e disseminada, sendo utilizada em diversas aplicações que necessitem do sensoriamento de posição.

Figura 4 - Potênciometro

Como mostrado na Tabela 1, o valor da faixa de atuação do AX-12A é de 0-300°, sensoriamento realizado pelo potenciômetro passando por um conversor A/D, ao converter o dado para o meio digital é possível por diferenciação encontrar a velocidade e a aceleração, abrindo possibilidades de diversas trocas de informações entre o usuário e o sistema, sendo necessário o armazenamento dessas informações, para isso, foi utilizado um conjunto de memórias.

O conjunto de memória é formado por uma RAM (*Random Access Memory*) e uma EEPROM (*Electrically-Erasable Programmable Read-Only Memory*), sendo responsáveis armazenar dados importantes para o usuário e por guardar parâmetros de controle. Nas Tabela 2 e Tabela 3 são mostrados os parâmetros da memória EEPROM e da memória RAM, respectivamente.

Tabela 2 - EEPROM

Endereço	Tamanho (Byte)	Nome do Dado	Acesso	Valor Inicial
0	2	Nº do Modelo	R	12
2	1	Versão do Firmware	R	-
3	1	ID	RW	1
4	1	Taxa de Transmissão	RW	1
5	1	Tempo de Atraso de Retorno	RW	250
6	2	Ângulo Limite (Horário)	RW	0
8	2	Ângulo Limite (Anti-Horário)	RW	1023
11	1	Temperatura Limite	RW	70
12	1	Tensão Mínima	RW	60
13	1	Tensão Máxima	RW	140
14	2	Torque Máximo	RW	1023

16	1	Nível de Retorno de Status	RW	2
17	1	Alarme LED	RW	36
18	1	Desligar	RW	36

Fonte: ROBOTICS

Onde, o ‘R’ significa somente leitura e o ‘RW’ significa leitura e escrita.

Esta tabela é importante por dois motivos: mostra quais são os parâmetros de controle do motor que podem ser modificados e qual é o valor inicial de cada parâmetro, permitindo prever possíveis problemas e adaptar os parâmetros para cada projeto.

Tabela 3 - RAM

Endereço	Tamanho (Byte)	Nome do Dado	Acesso	Valor Inicial
24	1	Ativar Torque	RW	0
25	1	LED	RW	0
26	1	Margem de Conformidade (Horário)	RW	1
27	1	Margem de Conformidade (Anti-Horário)	RW	1
28	1	Inclinação de Conformidade (Horário)	RW	32
29	1	Inclinação de Conformidade (Anti-Horário	RW	32
30	2	Posição Objetivo	RW	-
32	2	Velocidade de movimento	RW	-
34	2	Limite de Torque	RW	ADD 14&15
36	2	Posição Atual	R	-
38	2	Velocidade Atual	R	-
40	2	Carga Atual	R	-
42	1	Tensão Atual	R	-
43	1	Temperatura Atual	R	-
44	1	Registrado	R	0
46	1	Movendo	R	0
47	1	Trancado	RW	0
48	2	*Punch*	RW	32

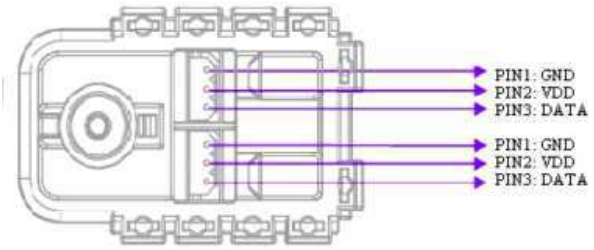
Fonte: ROBOTICS

Os valores da Tabela 3 são os dados que podem ser solicitados pelo controlador, e com eles é possível criar algoritmos para soluções dos mais diversos problemas. Neste projeto foram utilizados os parâmetros de velocidade atual, posição atual, posição objetivo, entre outros. Pode-se utilizar esses

parâmetros para realizar testes, como, por exemplo, o de carga atual e temperatura atual para analisar como o motor está se comportando diante da carga.

Para comunicação, é estabelecida uma conexão serial TTL do tipo *Half-Duplex,* ou seja, não consegue executar funções de enviar e receber dados simultaneamente. Cada motor AX-12A é identificado por um ID único na memória EEPROM, que recebe comandos e retorna respostas ao controlador externo (PC). A pinagem para conexão e alimentação dos motores é mostrado na Figura 5. Se acontecer de existirem dois motores com o mesmo ID (*Identity*), irá gerar problema na comunicação, porque nesse tipo de protocolo não é possível o envio simultâneo de informação. Para solucionar esse problema basta procurar o programa *"Wizard Dynamixel"* e efetuar a troca do ID na EEPROM.

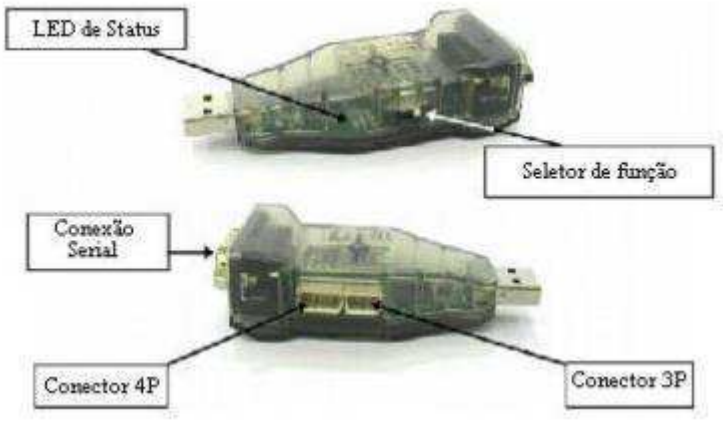
Figura 5 - Pinagem do motor AX-12A



Para a comunicação, é necessária a distinção dos tipos de pacotes de informação, pacote de instrução e pacote de status. Os pacotes de instrução são a comunicação oriunda do computador para os motores, já os pacotes de status são o contrário.

O dispositivo responsável por trafegar esses pacotes de informação e conectar os motores Dynamixel foi o USB2Dynamixel, a partir da porta USB de um computador. Tal dispositivo possui conexão 3P para motores da série AX, detém ainda conexão 4P para os dispositivos da série DX e RX, além disso, possui conexão serial, LED de status e um seletor de função. Na Figura 6 é mostrado o USB2Dynamixel.

Figura 6 - *USBDynamixel*



A função dos conectores e seus modos de comunicação estão listados abaixo:

Com 3 pinos: Comunicação TTL;

Com 4 pinos: Comunicação RS485;

Com cabo serial: Comunicação RS232.

A conexão do *USB2Dynamixel* com os motores AX-12A é feita como na Figura 7, dispondo os motores em série, conectando *USB2Dynamixel* ao primeiro motor, e o segundo terminal do primeiro conectando ao segundo motor, seguindo essa forma até o último motor, em que será conectado no terminal do último motor uma fonte de alimentação com a tensão contínua de 7-10 V, com finalidade de alimentar todos os outros motores. Quando selecionar a fonte utilizada no conjunto, atentar para a corrente que cada motor irá necessitar, e com a soma dessas correntes comprar ou fabricar a fonte de alimentação.

Figura 7 - Conexão dos motores AX-12A

Para realizar o controle dos motores AX-12A se faz necessária a utilização de uma biblioteca SDK, que fora escrita em na linguagem C. Apesar de possuir poucas funções, com a biblioteca é possível implementar o controle do robô *Scara.*

As funções utilizadas para realizar o controle do robô estão mostradas abaixo:

- MyDynamixel'MD' (port,baud): Iniciar os motores Dynamixel;

- Exit(MD): Encerrar a comunicação;

- viewsupportfcn(MD): Mostrar as funções existentes;

- adddevice(MD,id): Adicionar o dispositivo com o determinado ID;

- removedevice(MD,id): Remover o dispositivo com o ID selecionado;

- writeangle(MD, varargin): Movimenta o motor para o ângulo solicitado;

-setspeed(MD, varagin): Modifica a velocidade do motor;

-[presentPos] = readangle(MD,id): Lê o ângulo atual do motor;

-[status] = movendoouparado(MD,id): Informa se o motor está se movendo ou parado.

3. VISÃO COMPUTACIONAL E PROCESSAMENTO DE IMAGEM

A tecnologia de visão computacional é o processo de determinação e descrição de um espaço tridimensional no qual um agente está inserido, extraindo-se informações a partir de uma ou mais imagens, capturadas por uma ou mais câmeras, de modo a obter um modelo computacional do ambiente. Visão computacional pode ser entendida ainda como uma tentativa de fazer o processo inverso da formação de uma imagem, isto é, reconstruir suas propriedades e formas, além de obter parâmetros como iluminação e distribuição de cores (SZELISKI, 2010).

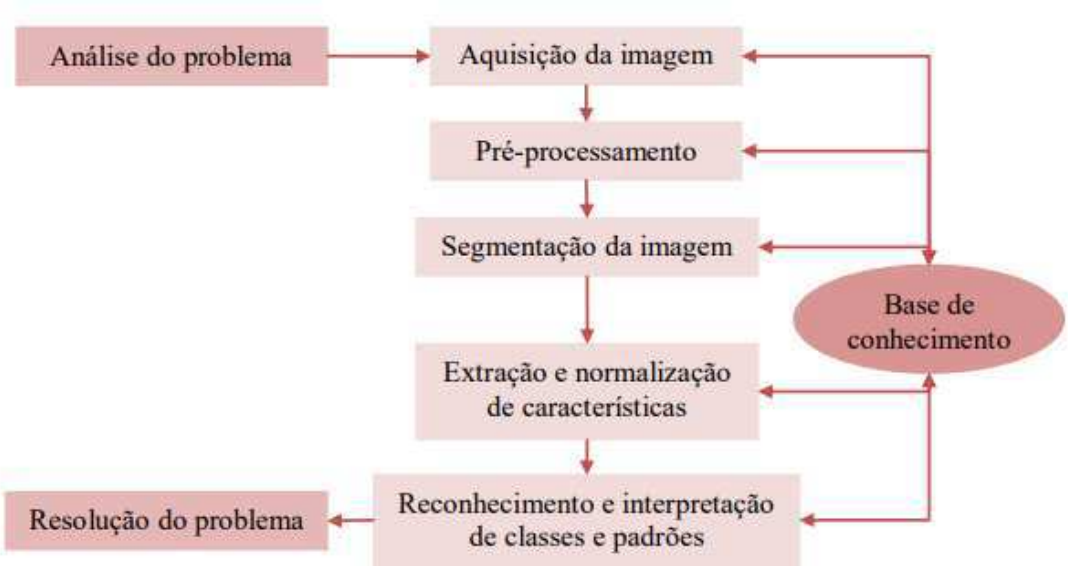
Sistemas de visão computacional são ferramentas de crescente aplicação industrial. Basicamente, estes sistemas envolvem a aquisição, processamento e interpretação informatizada de dados de imagem com vista a alguma aplicação (Groover, 2008).

De acordo com Gonzaga em 2010: Visão computacional é a ciência que caracteriza informações de interesse em uma imagem, a qual é estruturada na definição do problema visual a ser analisado e solucionado, organizado nas

etapas sequencias de aplicação: aquisição, pré-processamento, segmentação, extração de características e reconhecimento. Trata-se da integração de processos e representações gráficas para percepção visual.

Na imagem abaixo são apresentados os passos necessários para resolver um problema utilizando visão computacional.

Figura 8 - Sistema de visão computacional



De acordo com Forsyth e Ponce, as etapas sequenciais presentes em um problema de visão computacional se dividem ainda em três estágios: visão em baixo nível representando as etapas de aquisição e pré-processamento; visão em nível intermediário representando segmentação, extração e normalização de característica e visão em alto nível representando reconhecimento e interpretação de classes e padrões.

2.1 Transformada de *Hough*

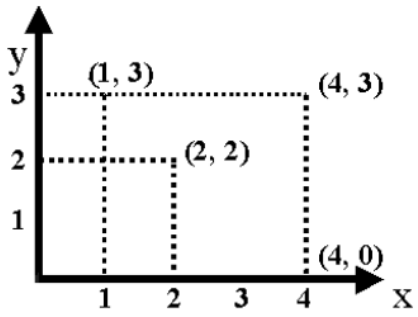
A transformada de Hough foi desenvolvida no ano de 1962 por Paul Hough e patenteada pela IBM. A transformada de Hough é uma ferramenta comumente utilizada no ramo da visão computacional para detectar curvas facilmente parametrizáveis em uma imagem, como retas, círculos e elipses. Está transformada fornece a relação entre o domínio da imagem e o domínio dos parâmetros, que consiste em uma matriz de acumuladores, com dimensão igual ao número de parâmetros a serem determinados.

É possível realizar várias parametrizações para o espaço de retas. Quando essa transformada foi criada por Hough, ele utilizou a equação de uma reta, y = mx + c, como uma representação paramétrica de uma linha, mas com essa equação existe um parâmetro ilimitado para as linhas que são paralelas a eixo y, para resolver isso, foi utilizada outra equação.

O algoritmo de Hough requer um acumulador com a dimensões iguais aos números de parâmetros desconhecidos da família de curvas que estão sendo buscadas, para as retas, a matriz de acumuladores possui dois parâmetros, o coeficiente angular e o coeficiente linear.

Considerando o exemplo abaixo é possível visualizar quatro pontos no gráfico, na Figura 9 é apresentado o espaço da imagem, ou seja, onde os estão os pixels.

Figura 9 - Exemplo da transformada de Hough



Com esses pontos é possível isolar os dois parâmetros desconhecidos da reta.

$$y = mx + c$$

Substituindo os valores de x e y no espaço da imagem e isolando c, encontra-se os valores da Tabela 4 - Equações do espaço de parâmetrosTabela 4:

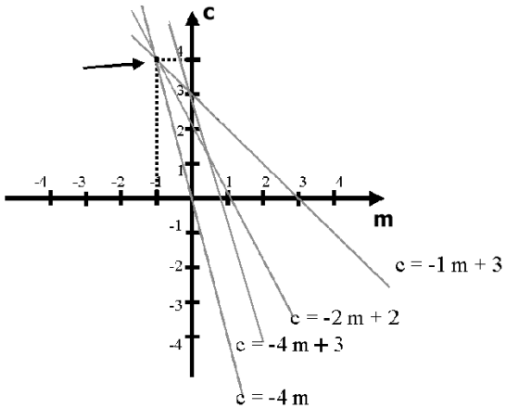
Tabela 4 - Equações do espaço de parâmetros

Pontos no espaço da imagem	Equações para o espaço de parâmetros
(1,3)	c = -1m + 3

(2,2)	c = -2m + 2
(4,3)	c = -4m + 3
(4,0)	c = -4m

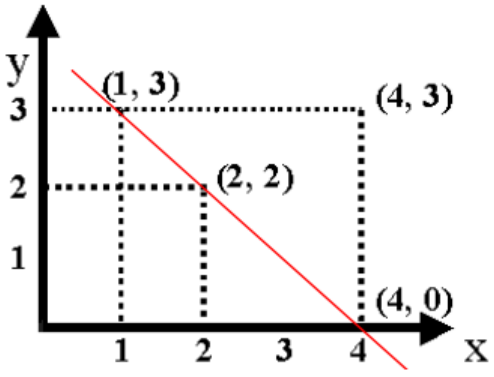
Plotando o espaço de parâmetros, chega-se à Figura 10:

Figura 10 – Espaço de parâmetros



Logo, para cada reta que cruza com outra no espaço de parâmetros significa que existe uma reta no espaço da imagem que consegue cruzar os pontos formadores dessas retas, e para cada cruzamento desse é somado uma unidade no acumulador daquele ponto A(m,c) . Com essa informação é possível ver na Figura 11, que existe um ponto que cruzam 4 retas, esse ponto possui um valor de c = 4 e de m = -1, então o Acumulador A(-1,4) é o de valor mais elevado, mostrando que a reta mais provável na imagem possui os y = mx + c, como mostrado na Figura 11.

Figura 11 – Reto com o maior número de pontos



Além dessa equação para encontrar retas em imagens, foi criado um método mais confiável, que transforma pontos no espaço da imagem em

solenoides no espaço dos parâmetros, a equação abaixo representa essa transformação:

$$\rho = x.cos\theta + y.sen\theta$$

Onde, ρ é à distância da origem até a reta, e o θ é o ângulo que a normal da origem até a reta forma com o eixo x.

Da mesma forma como na primeira equação, as curvas no espaço dos parâmetros que se cruzam são retas que passam por esses pontos representados pelas curvas. Então, como no exemplo anterior, quanto maior a quantidade de curvas que se cruzarem no mesmo ponto no espaço dos parâmetros, maior será o acumulador naquele ponto, e maior será a probabilidade daquele ponto ser uma reta.

Círculos são curvas parametrizáveis e, portanto, podem ser facilmente encontrados em uma imagem utilizando a transformada de Hough. Qualquer círculo pode ser descrito em termos de uma equação paramétrica que depende somente de três parâmetros: as coordenadas x e y do seu centro (a,b) e o seu raio R. A equação no espaço de parâmetros é dada por:

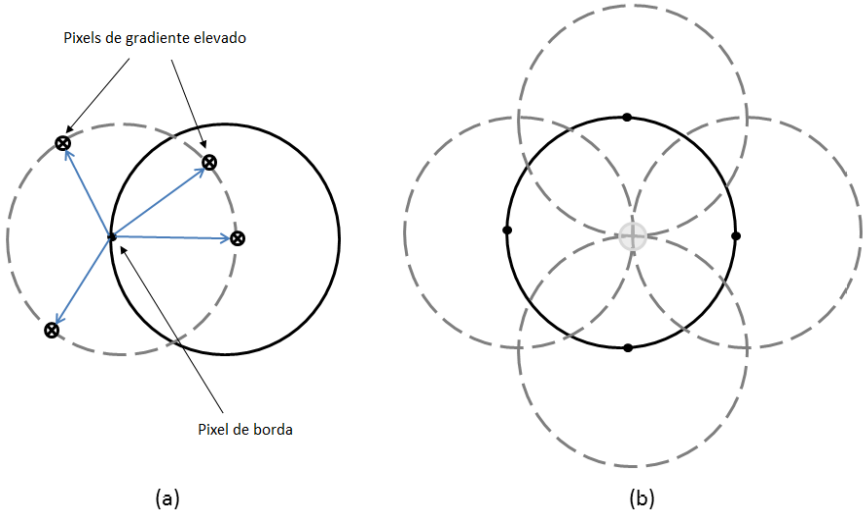
$$(x - a)^2 + (y - b)^2 = R^2$$

Há um conjunto de algoritmos para a detecção de círculos que empregam a transformada de Hough, tomando apenas implementações diferentes para isto. Contudo, há três etapas essenciais que são comuns: o cálculo da matriz de acumuladores, a estimação do centro do círculo e a estimação do raio do círculo.

Na primeira etapa, pixels do primeiro plano da imagem com gradiente elevado são aptos a incrementar a matriz de acumuladores, no qual cada um destes pixels forma um padrão circular, de raio fixo, centrado em um pixel de borda. A Figura 12a exemplifica este processo. Na fase de estimação do centro do círculo, as coordenadas do centro são obtidas aproximadamente buscando-se observar os máximos da matriz de acumuladores. Isso ocorre pois os pixels de gradiente elevado tendem a se acumular na região que corresponde ao centro do círculo a ser determinado no espaço dos parâmetros, como ilustrado na Figura 12b. Geralmente, o valor exato do raio do círculo não é conhecido, então é estabelecido um intervalo de busca para esse parâmetro. Logo, se esse intervalo

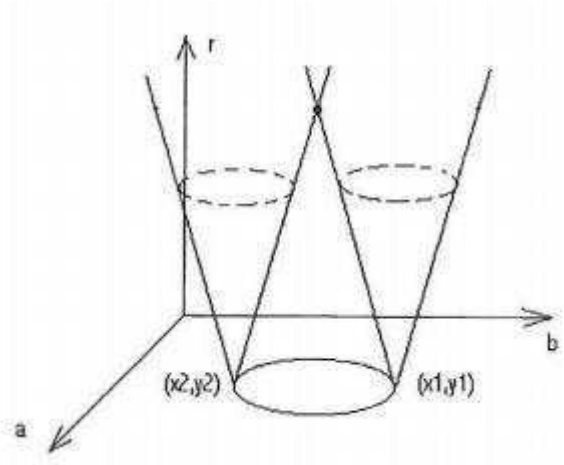
não for restrito, aumentará consideravelmente a quantidade de cálculos realizados. Portanto, é recomendável conhecer com antecedência a faixa de valores possíveis para o raio.

Figura 12 - Caracterização de um círculo utilizando a transformada de Hough



Quando se utiliza uma faixa de valores para o raio, o espaço de parâmetros deixa de ser planar e passa a ser espacial. Então, cada valor pesquisado passa a ser um dos planos do espaço de parâmetros. Logo, cada valor de raio forma um cone no espaço de parâmetros, como mostrado na Figura 13. O ponto que possuir maior quantidade de cruzamento desses cones está localizado o centro do círculo, ou seja, a cada ponto de cruzamento, a matriz de acumuladores daquele ponto é acrescentada em um.

Figura 13 - Cruzamento entre cones no espaço de parâmetros



2.2 *Toolbox Image Processing* e a função *imfindcircles*

A utilização de ferramentas computacionais já consolidadas é um passo importante para a aprendizagem e assimilação de conceitos básicos em visão computacional e processamento digital de imagens. Um dos softwares mais utilizados para isto é o MATLAB, que apresenta as toolboxes *Image Processing* e *Computer Vision*, que são pacotes adicionais do *software*.

Na toolbox *Image Processing* estão presentes diversos algoritmos, transformadas e funções que permitem o processamento, análise e visualização de imagens, facilitando e automatizando tarefas e processos comuns no processo digital de imagens. É possível ainda, melhoramento da imagem, redução de ruído e até mesmo processamento de imagem 3D. Uma das funcionalidades presentes neste pacote é a função *imfindcircles*, que utiliza a transformada de Hough para encontrar círculos em uma imagem digital.

A função *imfindcircles* tem como argumentos de saída *centers* e *radii*, que são as coordenadas dos círculos encontrados e os seus respectivos raios. Como argumentos de entrada, tem-se a imagem *Img*, o intervalo do raio dos círculos procurados chamado *radiusRange*. Há ainda pares de argumentos de entrada adicionais (*Name*, *Value*), que alteram especificações do algoritmo. Estes especificadores são descritos no. A sintaxe da função é: [*centers*, *radii*] = *infindcircles(Img, radiusRange, Name, Value)*.

Tabela 5 – Especificadores de *imfindcircles*

Name	Descrição	*Value*
'ObjectPolatiry'	Indica se os círculos são mais claros ou escuros que o fundo da imagem.	- *'bright'* (padrão) - *'dark'*
'Method'	Indica a técnica utilizada na matriz de acumuladores.	- '*PhaseCode'* (padrão) - '*TwoStage'*
'Sensitivity'	Indica a sensibilidade para encontrar círculos mais obscuros ou fracos.	- Valor entre 0 e 1 - 0.85 (padrão)
'EdgeThreshold'	Indica o limite do gradiente para determinar as bordas da imagem.	- Valor entre 0 e 1 - Valor calculado usando *graythresh.*

Fonte: MathWorks

4. MODEL BASED DESIGN

4.1 Introdução teórica ao Model Based Design

O *Model-Based Design* (MBD) é uma abordagem centrada no modelo para o desenvolvimento de sistemas de controle, processamento de sinais, comunicações e outros sistemas dinâmicos. O modelo inclui todos os componentes relevantes para o comportamento do sistema, algoritmos, lógica de controle e propriedade intelectual (AARENSTRUP, 2015).

Antes da utilização do *Model-Based Design*, para toda nova tecnologia, seria necessário a criação de um protótipo físico para realização de testes, com a finalidade de verificar e validar a tecnologia. O uso do MBD visa principalmente substituir os protótipos de plantas com valores elevados, tornando dessa forma o projeto mais viável, reduzindo custos, acelerando a detecção de erros e validando requisitos.

De acordo com Paternò (1999), alguns cientistas e engenheiros ainda têm uma reação negativa sobre o *Model-Based Design*, porque eles assumem que a simulação e modelagem estão próximo da teoria, distante da prática. Contudo, modelos são boas representações da realidade, e, para o ensino da engenharia são fundamentais, pois facilitam o entendimento e tornam didático o ensino de sistemas complexos.

Além de reduzir custos, o MBD apresenta outras vantagens, dentre elas: gerenciar sistemas complexos, automatizar tarefas complicadas e propensas a erros, explorar rapidamente novas ideias, criar uma linguagem comum que promova a comunicação e colaboração, capturar e reter propriedade intelectual, melhorar a qualidade do produto e reduzir os riscos (AARENSTRUP, 2015).

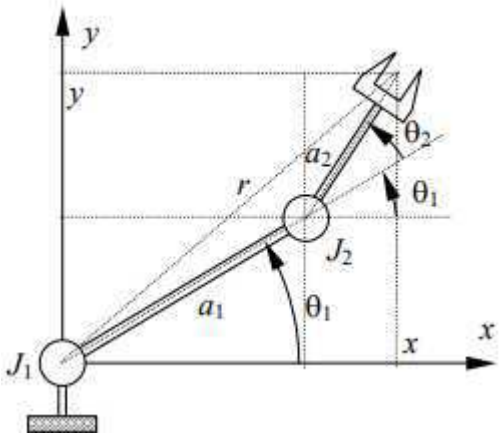
Para encontrar o modelo matemático do robô *Scara* é utilizado um robô com dois graus de liberdade como mostrado na Figura 14. Existem dois processos para determinação dos parâmetros: Cinemática Direta e Cinemática Inversa.

4.2 Cinemática Direta

A cinemática direta tem como objetivo encontrar os valores cartesianos, x e y, quando se possui os ângulos θ_1 e θ_2, as equações abaixo mostram essa relação:

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

Figura 14 - Cinemática Direta

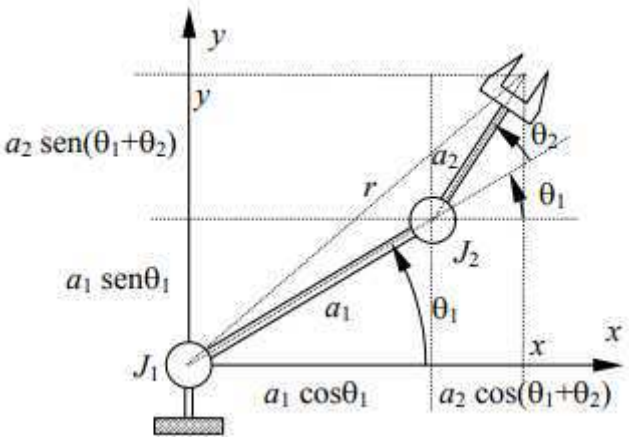


4.3 Cinemática Inversa

Já a cinemática inversa é o processo oposto, ou seja, encontrar os valores dos ângulos θ_1 e θ_2, utilizando os valores x e y.

Aplicando o teorema de Pitágoras na Figura 15, tem-se:

Figura 15 - Cinemática Inversa



$$r^2 = x^2 + y^2$$

Onde,

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

$$y = a_1 \operatorname{sen}\theta_1 + a_2 \operatorname{sen}(\theta_1 + \theta_2)$$

Elevando x e y ao quadrado, tem-se:

$$x^2 = {a_1}^2 \cos^2\theta_1 + 2a_1a_2 \cos\theta_1 \cos(\theta_1 + \theta_2) + {a_2}^2 \cos^2(\theta_1 + \theta_2)$$
$$y^2 = {a_1}^2 \operatorname{sen}^2\theta_1 + 2a_1a_2 \operatorname{sen}\theta_1 \operatorname{sen}(\theta_1 + \theta_2) + {a_2}^2 \operatorname{sen}^2(\theta_1 + \theta_2)$$

Aplicando no teorema de Pitágoras:

$$r^2 = {a_1}^2(cos^2\theta_1 + sen^2\,\theta_1) + 2a_1a_2\,[cos\,\theta_1\,cos(\theta_1 + \theta_2) + sen\,\theta_1\,sen(\theta_1 + \theta_2)] + {a_2}^2(cos(\theta_1 + \theta_2)^2 + sen(\theta_1 + \theta_2)^2)$$

Com a relação fundamental da trigonometria, chega-se:

$$r^2 = x^2 + y^2 = a_1^2 + a_2^2 + 2a_1a_2\,[\cos\theta_1 \cos(\theta_1 + \theta_2) + \operatorname{sen}\theta_1 \operatorname{sen}(\theta_1 + \theta_2)]$$

Aplicando identidade de somas e subtração de ângulos.

$$r^2 = x^2 + y^2 = a_1^2 + a_2^2 + 2a_1a_2cos\theta_2$$

Isolando θ_2:

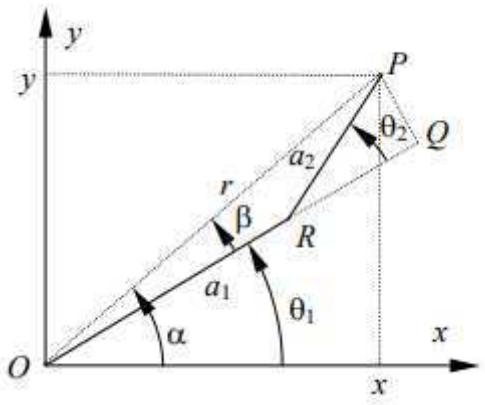
$$\theta_2 = \pm arccos\left(\frac{x^2 + y^2 - a_1^2 - a_2^2}{2\,a_1\,a_2}\right)$$

Onde, a utilização do arco cosseno gera dois valores, afirmando que existe duas posições do braço que alcançam o mesmo resultado, um deles o cotovelo do braço está para cima e o outro para baixo. Diante disso, o ângulo θ_2 é positivo quando o cotovelo está para baixo e negativo quando está para cima.

Para obter θ_1 é necessário utilizar o triângulo OPQ mostrando na Figura 16, aplicando a tangente no ângulo β, tem-se:

$$\tan\beta = \frac{a_2 sen\theta_2}{a_1 + a_2 cos\theta_2}$$

Figura 16 - Diagrama para encontrar θ_1



Calculando o ângulo α:

$$\tan \alpha = \frac{y}{x}$$

Aplicando a identidade trigonométrica de tan(α - β), tem-se:

$$\tan \theta_1 = \tan(\alpha - \beta) = \frac{\frac{y}{x} - \frac{a_2 sen\theta_2}{a_1 + a_2 cos\theta_2}}{1 + \frac{y}{x}\frac{a_2 sen\theta_2}{a_1 + a_2 cos\theta_2}}$$

Reduzindo a equação, chega-se em:

$$\tan \theta_1 = \frac{y(a_1 + a_2 cos\theta_2) - x\, a_2 sen\theta_2}{x(a_1 + a_2 cos\theta_2) + y\, a_2 sen\theta_2}$$

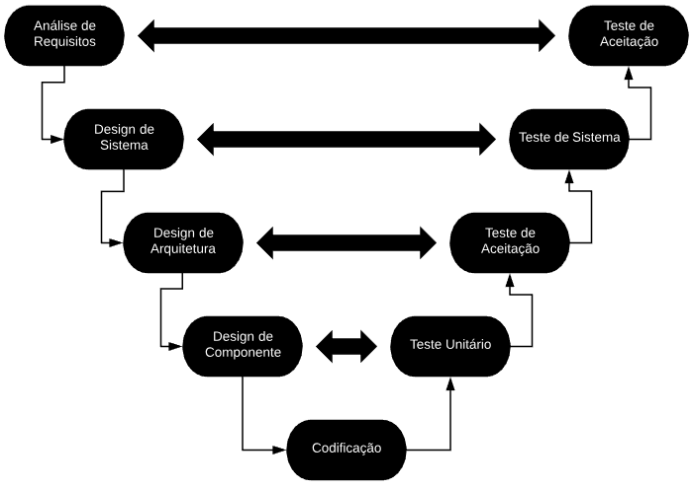
5. V-MODEL

O *V-model* é uma representação do desenvolvimento do sistema que destaca as etapas de verificação e validação no processo de desenvolvimento do sistema (MathWorks, 2018). Por conta dessas características, o *V-model* e o MBD são complementares, onde o MBD tem a função de implementar o protótipo, o *V-model* analisa essa implementação e conceitua os requisitos.

O lado esquerdo do "V" identifica as etapas que levam à geração de código, incluindo a especificação do sistema e o design detalhado do software. O lado direito do V concentra-se na verificação e validação dos passos citados no lado esquerdo, incluindo software e integração do sistema (MathWorks,2018). Todas as etapas do *V-Model* são bem definidas, diante disso, só é possível começar outra etapa quando a antecessor for completada.

O objetivo do *V-model* é melhorar a eficiência e eficácia do desenvolvimento de software e refletir a relação entre as atividades de teste e atividades de desenvolvimento, como mostrado na Figura 17 (Yadav,2012).

Figura 17 – Diagrama do V-model



O primeiro passo é realizar a análise de requisito, detalhar quais as funções que o projeto irá desempenhar. Após validar os requisitos com o teste de aceitação, é realizado o design do sistema, onde sua função é detalhar e compreender a configuração completa de *Hardware* e comunicação do produto em desenvolvimento. Completando a configuração do sistema, tem-se o design de arquitetura, onde será determinada por meio da viabilidade técnica e financeira, qual arquitetura será adotada. Para finalizar, é realizado o design de componente, é crucial determinar anteriormente os outros designs, pois esse é o design interno detalhado de todos os módulos.

Após o lado esquerdo do V implementado, a fase de codificação é iniciada, onde todo sistema é montado e codificado.

Como pode ser visto na Figura 17, nesse método existem 4 testes. Onde, o teste unitário avalia cada módulo individualmente, os testes de aceitação e de sistema servem para avaliar as funcionalidades do sistema, e a comunicação externa e interna. Para finalizar, é realizado o teste de aceitação, que tem como finalidade ponderar se o projeto está de acordo com os requisitos iniciais.

6. INTERFACE GRÁFICA

O termo interface possui uma série de significados, significados estes que fora se ampliando com o advento de novas tecnologias. Os usuários dessas

novas tecnologias possuem o controle de todas as funções disponibilizadas por meio da utilização das interfaces gráficas de usuário, mais conhecida como GUI (*Graphical User Interface*) De acordo com o dicionário, o significado da interface gráfica é “fronteira entre duas coisas”, ou seja, um ponto de interconexão entre o usuário e a máquina.

Os primeiros projetos que utilizavam interface gráfica eram designados para soluções de problemas nas áreas científicas. Ou seja, eram desenvolvidos e utilizados por engenheiros, em que todos já possuíam uma capacidade de entendimento do projeto considerável. Mas ao longo do tempo, essa prática foi mudando com a disseminação dos computadores, surgindo os mais variados tipos de usuários. Que agora utilizavam essa tecnologia para atividades rotineiras no trabalho ou para lazer. Atualmente, desde a infância o indivíduo já possui contato direto com os recursos da informática, prevendo-se que, praticamente todo ser humano irá utilizar computadores no futuro de uma ou de outra forma.

Diante disso, percebe-se que a interface é um dos componentes mais importantes do sistema, pois para o usuário à interface é o próprio sistema, ou seja, o contato mais direto que o usuário possui.

De acordo com CYBES (2000): “Pode-se definir Interface Humano-Computador como uma parte de um sistema interativo com a função de refletir para o usuário o estado do sistema, traduzir ações do usuário em pedidos de processamento (funcionalidades), e mostrar os resultados de forma adequada, além de coordenar a interação.”

Então para uma melhor experiência do usuário existem alguns métodos de design de interface. O utilizado no projeto foram as oito regras de ouro de Ben Shneiderman (Ben Shneiderman, 1986) em conjunto com as heurísticas de Nielsen:

1 - Consistência na interface: usar terminologia idêntica deve ser usado em prompts, menus, layout, letras maiúsculas e entre outros.

2 - Atender todas as necessidades: deve fazer uma interface para os mais diversos tipos de usuários.

3 - Oferecer *feedback* informativo: para cada informação do usuário, deve haver um *feedback* do sistema.

4 - Criar diálogos para gerar fechamento: sequência de ações deve ser organizada de forma a preparar o usuário para próxima sequência de tarefas.

5 - Prevenir erros: projetar o sistema de modo que os usuários não possam cometer erros graves.

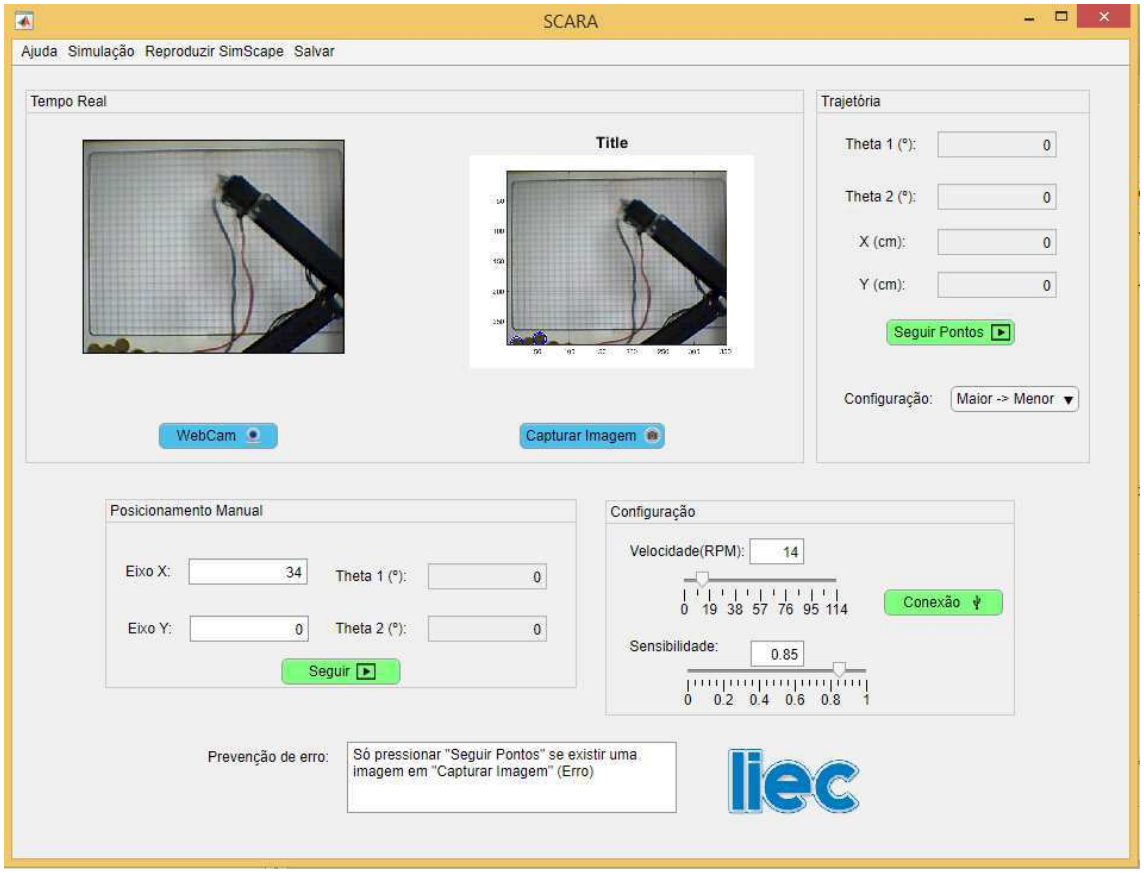
6 - Permitir reversão fácil de ação: permitir que o usuário possa corrigir alguma ação realizada equivocadamente.

7 - Oferecer sensação se controle para o usuário: dar a sensação ao usuário que ele está no controle, que a interface só obedece às ações solicitadas.

8 - Reduzir a carga de memória de curto prazo: fazer com que a necessidade de lembrar algum número ou ação precedente, seja a menor possível.

A interface criada para comandar o robô *Scara* foi embasada na definição de Cybes e nas oito regras de ouro de Bem Shneiderman, consequentemente, tornando-a mais didática. A interface realizada nesse projeto está apresentada na Figura 18.

Figura 18 - Interface gráfica do robô *scara.*



Com o objetivo de tornar mais simples o entendimento e a utilização da interface gráfica ao usuário é colocado no menu superior o botão de “Ajuda” que leva o usuário à uma tela com diversas instruções sobre a funcionalidade do projeto e auxilia na utilização da interface.

A interface possui um total de 5 *Buttons*, dois com a coloração azul responsáveis por realizar as funções da câmera e três com a coloração verde responsáveis por realizar ações no robô *Scara.* Possuindo ainda, dois *Slider*, sendo um responsável pela mudança de velocidade dos motores AX-12A, e o outro responsável pela alteração de sensibilidade da imagem capturada para reconhecimento de círculos. Para finalizar, uma *drop down* responsável pela escolha da configuração dos círculos presentes na imagem que guiará o robô até os itens.

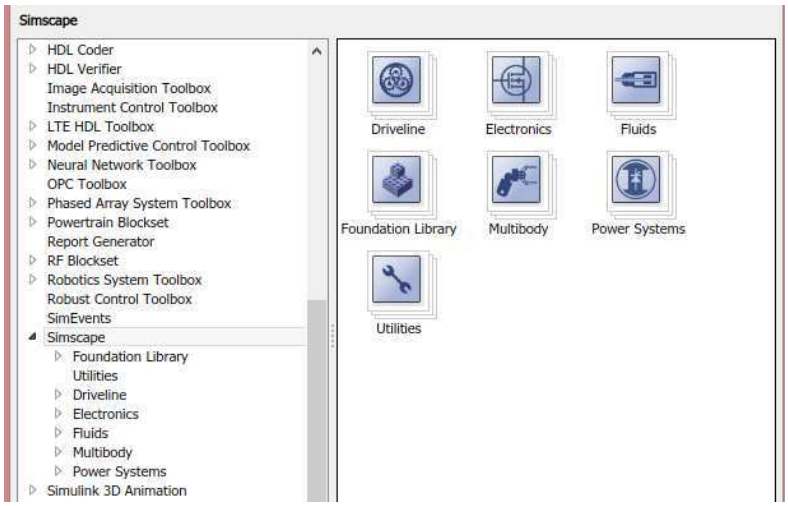
7. SIMSCAPE

Simscape é um conjunto de bibliotecas de blocos e recursos para modelagem de sistemas físicos no ambiente *Simulink*. Enquanto blocos tradicionais do *Simulink* representam operações matemáticas, os blocos do *Simscape* representam por si só diagramas dos modelos matemáticos do sistema em projeto. Esta diferença se deve a abordagem utilizando redes físicas, na qual cada sistema é representado por elementos funcionais que trocam energia entre si, de forma não direcional, por meio de portas conectadas. A conexão é feita de modo a se assemelhar às conexões entre componentes físicos reais.

Devido a facilidade e variedade de elementos que o *Simscape* fornece, empregá-lo para modelar problemas é relativamente simples, sendo possível ainda utilizá-lo como uma ferramenta auxiliar para o ensino de engenharia.

A biblioteca do *Simscape* pode ser dividida em duas grandes bibliotecas: a biblioteca de utilidades, que contém os blocos necessários para modelar as redes físicas; e a biblioteca de fundamentos, que contém os blocos que modelam os sistemas físicos. Esta última, por sua vez, é subdividida em bibliotecas menores, classificadas de acordo com o tipo de sistema que representam. A Figura 19 mostra em detalhes as subdivisões da biblioteca.

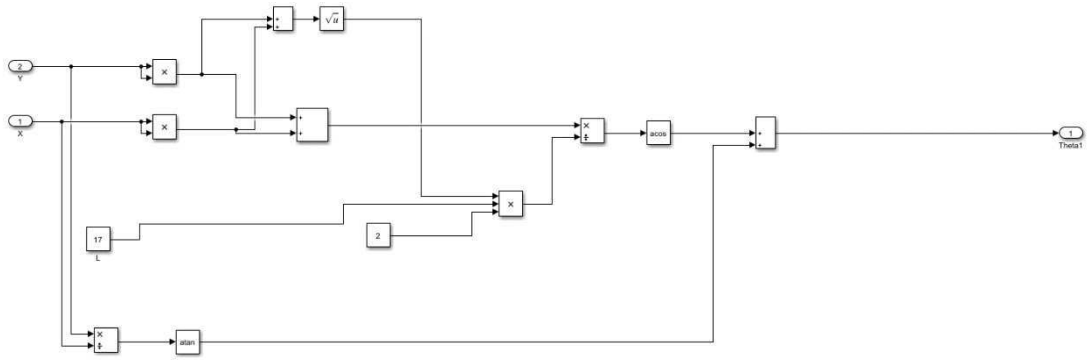
Figura 19 - Biblioteca *Simscape*



Para a modelagem do robô *Scara*, foi utilizada a biblioteca *Multibody*, que fornece um ambiente de simulação de sistemas com múltiplos corpos em 3D,

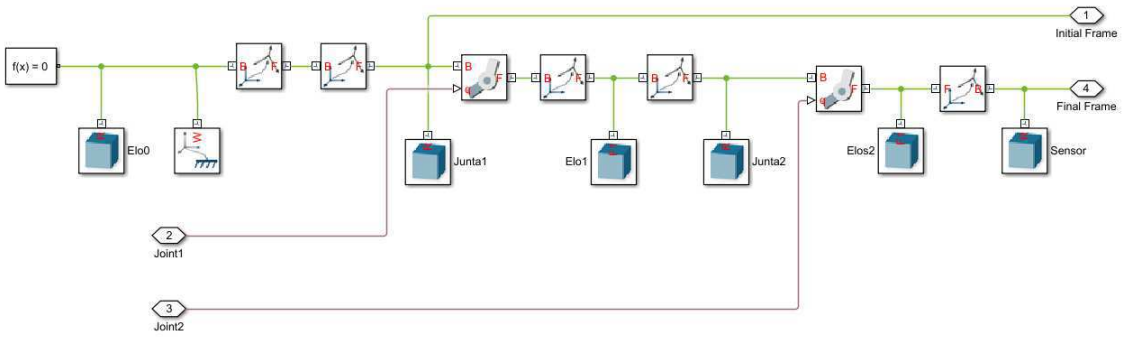
como suspensões de automóveis, equipamentos de construção e robôs. De acordo com a montagem feita, o *Simscape* formula e resolve as equações mecânicas que descrevem os corpos envolvidos, levando em conta os parâmetros fornecidos para a simulação. Devido a fácil integração com as demais ferramentas do *Simulink*, é possível integrar o sistema mecânico do *Simscape* com algum sistema de controle projetado, ou até fazer uso de outras bibliotecas do próprio *Simscape*, como pode ser visto o projeto do controlador no *Simulink* na Figura 20.

Figura 20 - Controlador utilizando o Simulink



Para a construção do modelo do robô *Scara*, foram usados os blocos *Solid*, que corresponde aos elos do braço e os blocos *Revolute Joint*, que representam as juntas de revolução. Além disso, foi necessário o emprego de um bloco de referencial inercial, o *World Frame*, e alguns blocos de transformação de referências, o bloco *Rigid Transform*, que é responsável por rotacionar e/ou transladar a referência. A Figura 21 apresenta o modelo desenvolvido.

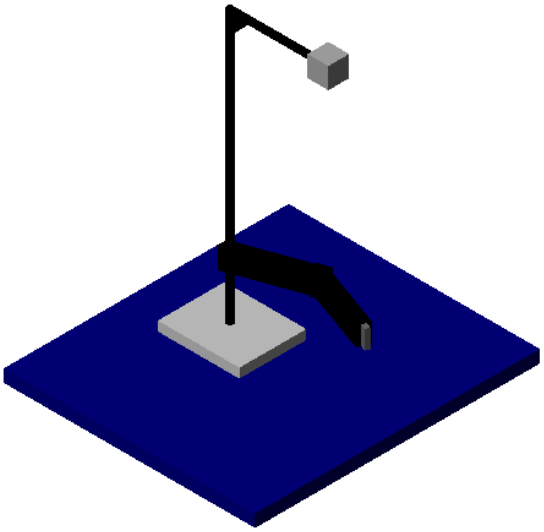
Figura 21 - Modelo físico no Simscape do robô Scara



Alguns outros blocos encontrados nesta figura são os blocos *Connection Port*, que servem para fazer a conexão do modelo com os demais conjuntos de blocos da simulação, e um bloco *Solver Configuration*, que é o bloco responsável por especificar os parâmetros do modelo antes da simulação de cada rede física presente no modelo.

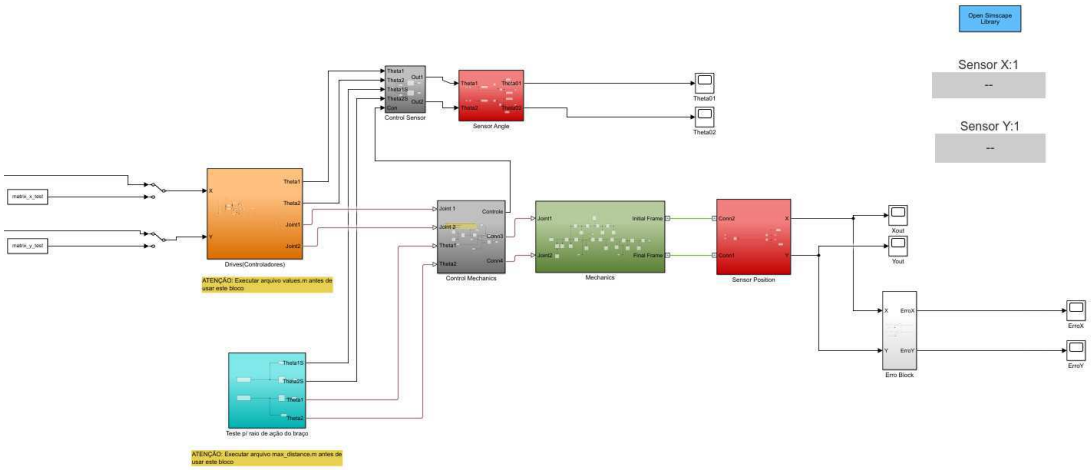
Com o modelo desenvolvido, foi possível avaliar se a modelagem cinemática de fato descrevia adequadamente o braço robótico, além de que tornou fácil explorar ideais para movimentação e trabalho do robô. A Figura 22 mostra o modelo na simulação.

Figura 22 - Simulação do robô *Scara*



Com a modularização do projeto torna-se mais simples a alteração e utilização do mesmo em outras aplicações, devido a isso, foi modularizado todos os sistemas utilizados, como: o bloco de controle, o físico e o de sensoriamento. Os blocos modularizados são mostrados na Figura 23.

Figura 23 - Blocos divididos em módulos da simulação



8. INVENTOR

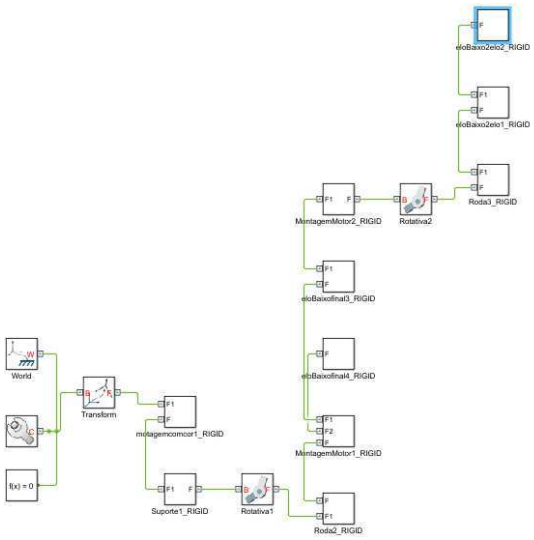
O *Autodesk Inventor* é um software utilizado na engenharia, com diversas funcionalidades, a principal delas é: facilitar a elaboração e dimensionamento das peças. O motivo da vasta utilização do software é sua interface didática e seus comandos e componentes autoexplicativos. Com o *Autodesk Inventor* é possível a criação de modelos 3D, realizar detalhamento em 2D e analisar o projeto tanto estaticamente quanto dinamicamente, prevenindo possíveis erros no projeto físico.

Segundo WAGUESPACK (2013), o *software Autodesk Inventor* é aplicado em projetos de engenharia e para tal possui diversos recursos, que auxiliam na modelagem, análise estrutural e detalhamento de conjuntos e seus componentes. Dentre suas principais funcionalidades está a modelagem em 3D, montar conjuntos e verificar as interferências entre componentes, mas com está mesma ferramenta é possível inserir restrições, regras e analisar a estrutura através do método dos elementos finitos e gerar os desenhos de detalhes em 2D.

Com a utilização do *Simscape* e do seu *plug-in Simscape Multibody Link* é possível transferir a simulação realiza no *Autodesk Inventor* para o *Simscape*. Para realizar essa transição é necessário baixar e instalar o plug-in, abrindo o

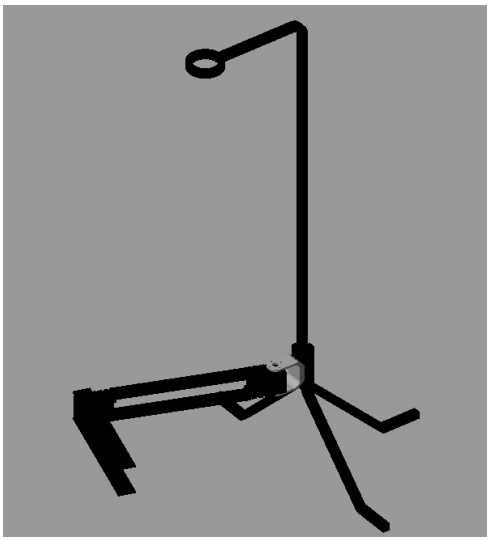
matlab, então executar o comando "*smlink_linkinv*". Após isso, no projeto do *Inventor Autodesk* é necessário entrar na aba do menu principal chamado "*add-ins*", se não possuir essa aba, na barra de pesquisa deve-se procurar *Simscape*, e exportar para o *Simscape*. Agora, no *matlab* é necessário entrar com o comando "smimport('projeto')", onde a pasta atual do matlab está aberta na exportação realizada pelo *Inventor* e o 'projeto' é o arquivo com extensão .xml. Um possível problema seria as vírgulas no arquivo .xml, para resolver, basta abrir o arquivo e substituir todos as vírgulas por ponto. Após isso, seu projeto será transferido para o *Simscape*, e todos os blocos físicos serão criados, como se pode ver na Figura 24.

Figura 24 - Blocos gerados pelo Simscape



O objetivo de transferir o projeto do *Autodesk Inventor* para o *Simscape* é, tornar a simulação o mais próximo da realidade e elevar a estética da simulação. Com esses aspectos o projeto se torna mais didático e mais chamativo, como se pode observar na figura.

Figura 25 - Simulação do braço robótico transferido do *Inventor*



9. RESULTADOS OBTIDOS

Os resultados obtidos com a simulação mostram que o modelo cinemático adotado descreve de forma bastante satisfatória o robô *Scara*, conforme pode ser visto na Tabela 5. A adoção do *Model-Based Design* com auxílio da biblioteca *Simscape* para modelagem e simulação no ambiente *Simulink* permitiu prever erros e limitações mecânicas do robô antes mesmo da sua fabricação, reduzindo o tempo e o material gasto no projeto.

Para realizar a simulação foi previamente selecionados alguns valores de 'x' e 'y'. Aplicando esse valor selecionado na simulação, encontrou-se o 'θ_1 simulação' e 'θ_2 simulação'. Então, aplicou-se a função *readAngle*, mostrada anteriormente, para encontrar o valor de 'θ_1 real' e 'θ_2 real'. Para finalizar, é aplicado a equação para encontrar o erro, dada pela equação abaixo:

$$Erro = \frac{|\theta\text{simulação} - \theta\text{real}|}{\theta\text{real}}$$

Tabela 6 - Comparação entre os valores de ângulo do robô físico real com os valores de ângulo da simulação

x (cm)	y (cm)	θ_1 real (°)	θ_2 real (°)	θ_1 simulação (°)	θ_2 simulação (°)	ErroX (%)	ErroY (%)
10,00	5,00	105,90	291,20	105,80	291,60	1.057	0.776
-4,00	12,00	190,30	285,60	190,30	286,30	0.222	1.727
20,00	-7,00	181,80	46,63	182,20	47,10	0.894	2.368
-7,50	-7,50	86,80	5,865	86,82	6,355	1.450	4.50
17,50	-19,50	141,60	70,09	141,50	70,82	1.045	0.123
-15,00	-15,00	66,28	46,63	66,40	47,21	0.069	1.353
-2,00	20,00	191,80	257,50	192,00	257,50	2.859	0.051

Analisando os valores experimentais obtidos é possível notar que o resultado é satisfatório, pois todos os valores de erro estão abaixo de 5%, mostrando que a modelagem cinemática descreve adequadamente a realidade.

10. CONSIDERAÇÕES FINAIS

O principal objetivo deste trabalho foi alcançado, tendo em vista que foi desenvolvida um braço robótico baseado no robô *Scara*. Por meio de uma interface gráfica feita no ambiente de desenvolvimento *App Designer*, o usuário pode acessar facilmente a simulação, criada utilizando a biblioteca *Simscape*, com foco na importância da abordagem de projeto baseada em modelo, o *Model-Based Design* (MBD). O usuário ainda é capaz de visualizar animações que ilustram o comportamento cinemático do robô e checar o modelo empregado.

Esse robô poderá ser implementado nos laboratórios de robótica. Como a plataforma permite que o modelo seja acessado com facilidade, pode-se abordar a simulação para verificar a modelagem cinemática. Após isto, é possível comparar as respostas obtidas no robô *Scara* real com o resultado obtido no *Simscape*. Aspectos ligados a cinemática, MBD e programação podem ser discutidos, levando à sala de aula uma forma mais prática de abordar temas que são de extrema importância na robótica.

Em trabalhos futuros, pretende-se aprimorar o robô para ele seja capaz de realizar atividades como o reconhecimento e organização de peças, etapa esta qual será empregado processamento de imagem em conjunto com uma rede neural. Essa melhoria visa tornar a plataforma mais didática e próxima de uma

aplicação real da indústria, encorajando e desmitificando o uso da robótica para os alunos.

11. REFERÊNCIAS

MathWorks (2018). Validation and verification for system development, https: //www.mathworks.com/help/ecoder/gs/ v-model-for-system-development.html# brufb98-9/.

Yadav S. R. Improvement in the V-Model. International Journal of Scientific & Engineering Research, Volume 3, Issue 2, February-2012.

Paternò, Fabio. Model-based design and evaluation of interactive applications. Springer, 1999.

Aarenstrup, R. (2015). Managing Model-Based Design, MathWorks, Inc.

BIS, Walter de Abreu ; NASCIMENTO JUNIOR, W. B. . Modelagem do Conhecimento Ergonômico para Avaliação da Usabilidade de Objetos de Interação. In: Workshop on Human Factors in Computer Systems, 2000, Gramado. Proceedings of the 3rd Workshop on Human Factors in Computer Systems. Porto Alegre: Sociedade Brasileira de Computação, 2000.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., & Elmqvist, N. (2005). Designing the user interface: Strategies for effective human-computer interaction. 4th Edition. Pearson Education Inc.

ROMANO, V. F. ; DUTRA, M. S. . Introdução à Robótica Industrial. In: Vitor Ferreira Romano. (Org.). Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos. 1ed. São Paulo: Edgard Blücher, 2002, v.1, p. 1-19.

SZELISKI, Richard. Introduction. In: Computer Vision: Algorithms and Applications. Springer, 2011, p. 1-28.

COSTA, Anna Helena Reali; OKAMOTO JR., Jun. Interação entre robô e ambiente. In: Vitor Ferreira Romano. (Org.). Robótica Industrial: aplicação na indústria de manufatura e de processos. 1ed. São Paulo: Editora Edgard Blücher, 2002, v. 1, p. 71-93.

The MathWorks, Inc. Image Processing Toolbox User's Guide. Disponível em: https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf.

The MathWorks, Inc. Simscape User's Guide. Disponível em: https://ww2.mathworks.cn/help/pdf_doc/physmod/simscape/simscape_ug.pdf.

PATERNÒ, Fabio. Model-based design and evaluation of interactive applications. Springer, 1999.

AARENSTRUP, R. Managing Model-Based Design. CreateSpace Independent Publishing Platform, 2015.

CYBIS, Walter de Abreu; NASCIMENTO JUNIOR, W. B.. Modelagem do Conhecimento Ergonômico para Avaliação da Usabilidade de Objetos de Interação. In: Workshop on Human Factors in Computer Systems, 2000, Gramado. Proceedings of the 3rd Workshop on Human Factors in Computer Systems. Porto Alegre: Sociedade Brasileira de Computação, 2000.

CARRARA, Valdemir. Apostila de Robótica. Universidade Braz Cubas.

MURATA Manufacturing Co., Ltd. Disponível em :https://docs-emea.rs online.com/webdocs/13f2/0900766b813f2fab.pdf

GONZAGA, A.(2010). Notas de Aula em visão computacional (Disciplina Pós-Graduação). Universidade de São Paulo, Escola de Engenharia de São Carlos, Departamento de Engenharia Elétrica – Laboratório de Visão Computacional (USP/EESC/SEL/LAVI) – São Carlos, São Paulo – Brasil.

FORSYTH, D. A. e PONCE, J. (2002). Computer Vision: A modern Approach. Prentice Hall – US ed.

Groover, M., 2008. “Automation, Production Systems and ComputerIntegrated Manufacturing – Third Edition”. Prentice-Hall, New Jersey.

Duda, R. and Hart, P. - Use of the Hough transformation to detect lines and curves in pictures.

Marroni, L. Aplicação da transformada de Hough para localização dos olhos em faces humanas (2002). Dissertação (Mestrado em Engenharia Elétrica) – Universidade de São Paulo.

Santaella, A., 2004 – Design de Interface (2004). Dissertação (Mestrado em Comunicação e Semiótica) – Universidade Católica de São Paulo.

Shneiderman, Ben. Designing the User Interface, 1986.