FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO
“ADIB MOISÉS DIB”

ANSELMO JOSÉ FERREIRA MACHADO
LEONARDO JÚLIO DA SILVA
MANOELA SANT´ANA DE SOUSA
PAULO ROBERTO GOULART

SISTEMA DE CORREÇÃO DE COORDENADAS DINÂMICAS PARA ROBÔS

São Bernardo do Campo - SP
Junho/2019

ANSELMO JOSÉ FERREIRA MACHADO
LEONARDO JÚLIO DA SILVA
MANOELA SANT´ANA DE SOUSA
PAULO ROBERTO GOULART

SISTEMA DE CORREÇÃO DE COORDENADAS DINÂMICAS PARA ROBÔS

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo (a) em Automação Industrial.

Orientador: Prof. Me. Marcelo George Griese

São Bernardo do Campo - SP
Junho/2019

ANSELMO JOSÉ FERREIRA MACHADO
LEONARDO JÚLIO DA SILVA
MANOELA SANT´ANA DE SOUSA
PAULO ROBERTO GOULART

SISTEMA DE CORREÇÃO DE COORDENADAS DINÂMICAS PARA ROBÔS

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de São Bernardo do Campo “Adib Moises Dib” como requisito parcial para a obtenção do título de Tecnólogo (a) em Automação Industrial.

Trabalho de Conclusão de Curso apresentado e aprovado em:01/06/2019

Banca Examinadora:

_____

Prof. Me. Marcelo George Griese, FATEC SBC - Orientador

_____

Prof. Me. Erasmo Assumpção, FATEC SBC - Avaliador

_____

Prof. Dr.Wellington de Sousa, FATEC SBC - Avaliador

Dedicamos esse trabalho aos nossos pais, companheiros, filhos e a todos os professores e colegas que de alguma forma nos ajudaram ao longo dessa caminhada acadêmica.

Agradecemos ao Prof. Me. Marcelo George Griese e ao Prof. Dr. Delcínio Ricci pela ajuda durante a elaboração deste trabalho.

“Há duas formas de viver a vida: uma é acreditar que não existem milagres, a outra é acreditar que todas as coisas são um milagre”

ALBERT EINSTEIN

RESUMO

Este trabalho propõe o desenvolvimento de um sistema de correção de coordenadas dinâmicas através de um sistema de visão para robôs manipuladores, capaz de identificar a posição atual da peça dentro de uma área de manipulação pré-determinada. Através do cálculo da diferença nos eixos x e y da posição atual para a posição programada no robô torna-se possível que o robô pegue uma peça em qualquer ponto dentro da área de manipulação, aumentando a flexibilidade e agilidade do processo. Com a aplicação deste projeto se obtém uma significativa diminuição de paradas indesejadas para manutenção e referenciamento de robôs resultantes de erros no manuseio de peças e colisões. Por meio de uma interface com o usuário, um procedimento prévio de calibração é realizado e a cada fotografia tirada da peça um par de coordenadas x e y é informado para que o robô possa manusear a peça em sua atual posição. O objetivo deste sistema é flexibilizar a célula de robótica da FATEC São Bernardo do Campo, com equipamentos de custos acessíveis. Na estrutura do trabalho são apresentados: o robô industrial, comunicação entre robôs e células de manufatura e de processos, utilização de sistemas de visão em robôs industriais, visão artificial e processamento de imagens, digitalização de imagens, representação das cores, propriedades e operações com imagens digitais, programação e biblioteca para processamento de imagens.

Palavras chave: Coordenadas. Imagem. Manipulação. Robô. Sistema de Visão.

ABSTRACT

This project proposes the development of a system of correction of dynamic coordinates through a vision system for manipulators robots, able to identifying the current position of the part within a predetermined area of manipulation. By calculating the difference in the x and y axes from the current position to the position programmed in the robot it becomes possible for the robot to pick up a part at any point within the manipulation area, increasing the flexibility and agility of the process. With the application of this project a significant reduction of unwanted stops for maintenance and referencing of robots resulting from errors in the handling of parts and collisions, is obtained. Through a user interface, a previous calibration procedure is performed and for each photograph taken from the part a pair of x and y coordinates is informed so that the robot can handle the part in its current position. The objective of this system is to make FATEC São Bernardo do Campo robotic cell more flexible, with affordable equipment. In the structure of the work are presented: the industrial robot, communication between robots and manufacturing cells and processes, use of systems of vision in industrial robots, artificial vision and image processing, image scanning, color representation, properties and operations with digital images, programming and library for image processing.

Keywords: Coordinates. Images. Manipulation. Robots. Vision System.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

CAD	Computer aided design
CCD	Dispositivo de carga acoplada (charge-coupled device)
CMOS	Complementary Metal Oxide Semiconductor
IDE	Integrated development environment
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
PLC	Programmable logic controller
RGB	Red,Green and Blue
RIA	Robotic Industries Association
SVA	Sistema de visão artificial

SUMÁRIO

INTRODUÇÃO

A automação em indústrias de diversos setores tem se tornado cada vez mais comum e necessária para a manutenção e aumento da competitividade econômica de uma empresa, uma vez que automatizar torna processos mais rápidos, flexíveis e padronizados, resultando em mais qualidade e eficiência, e menos desperdícios.

Por possuir características antropomórficas, a utilização de robôs pode ser uma alternativa vantajosa à mão de obra humana na indústria, principalmente em casos onde a atividade a ser realizada é extremamente repetitiva, com baixo conforto ergonômico, insalubre. A título de exemplo pode se citar atividades de soldagem, pintura, manipulação de peças pesadas, ou então atividades que requerem precisão de posicionamento, força, velocidade ou sequenciamento, como montagem de conjuntos mecânicos, aparafusamento, aplicação de cola e montagem de placas eletrônicas.

Como a utilização dos robôs tornam os processos padronizados, sua utilização requer um processo minimamente padronizado, principalmente no que tange ao posicionamento, uma vez que os pontos são programáveis e os mesmos são determinados pelas coordenadas espaciais de posicionamento X, Y e Z e orientação Rx, Ry e Rz. Tal aspecto se torna um problema, por exemplo, em sistemas em que um robô trabalha em conjunto com esteiras transportadoras, dispositivos alimentados manualmente ou sistemas com menor precisão de posicionamento do que o robô; onde não há garantia de que a posição da peça a ser manipulada é sempre a mesma e torna provável haver colisões do robô com a peça, provocando perda das referências de posicionamentos dos seus eixos e gerando uma grande parada do processo para manutenção e/ou correção dos pontos.

Este trabalho, que se intitula Sistema de Correção de Coordenadas Dinâmicas para Robôs tem por objetivo elaborar um sistema de visão aplicado à célula de manufatura robótica da Faculdade de Tecnologia Adib Moisés Dib para lhe passar correções de coordenadas dinamicamente. Justifica-se por tornar processos

industriais mais flexíveis, uma vez que a utilização de um robô com um sistema de correção de coordenadas admite variações de posicionamento de peças que são manipuladas, pintadas e soldadas.

Sua elaboração consiste em uma câmera fixa sobre uma mesa, que capta a variação no posicionamento de uma peça dentro de uma área de 300 mm x 300 mm. Após a tratativa da imagem capturada são gerados valores de correção das coordenadas x e y, de acordo com uma referência pré-estabelecida, a fim de que seja transmitido ao robô e o mesmo possa ter seus pontos reprogramados, pegando a peça independentemente de seu posicionamento dentro do campo de leitura.

Este trabalho é divido da seguinte forma:

Capítulo 1 – Fundamentação teórica: são abordadas teorias que dão sustentação ao desenvolvimento do projeto intitulado Sistema de Correção de Coordenadas Dinâmicas para Robôs;

Capítulo 2 – Metodologia: descreve a trajetória que se percorre para o desenvolvimento de uma pesquisa. São apresentados técnicas e métodos e fases da elaboração do trabalho;

Capítulo 3 – Desenvolvimento do projeto: encontra-se o passo a passo do seu desenvolvimento estrutural e representação de figuras para melhor compreensão.

E finalmente, as Considerações finais: encontram-se o objetivo e justificativa propostos na introdução, relação dos fatos verificados e teorias, conquistas alcançadas, pontos fortes e fracos e propostas de melhoria para novos projetos.

1 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordadas as teorias que dão sustentação ao desenvolvimento e construção do projeto Sistema de Correção de Coordenadas Dinâmicas para Robôs.

1.1 O robô industrial

Santos (2003) descreve que o robô industrial é um manipulador multifuncional reprogramável controlado por posição automático, com vários graus de liberdade, capaz de manipular materiais, peças, ferramentas ou dispositivos especializados por meio de movimentos programados para o desempenho de uma variedade de tarefas.

Muitas vezes, leva a aparência de um ou vários braços terminando em um pulso. Sua unidade de controle usa um dispositivo de memorização e algumas vezes pode usar aparelhos de sensoriamento e adaptação que levam em conta o ambiente e as circunstâncias.

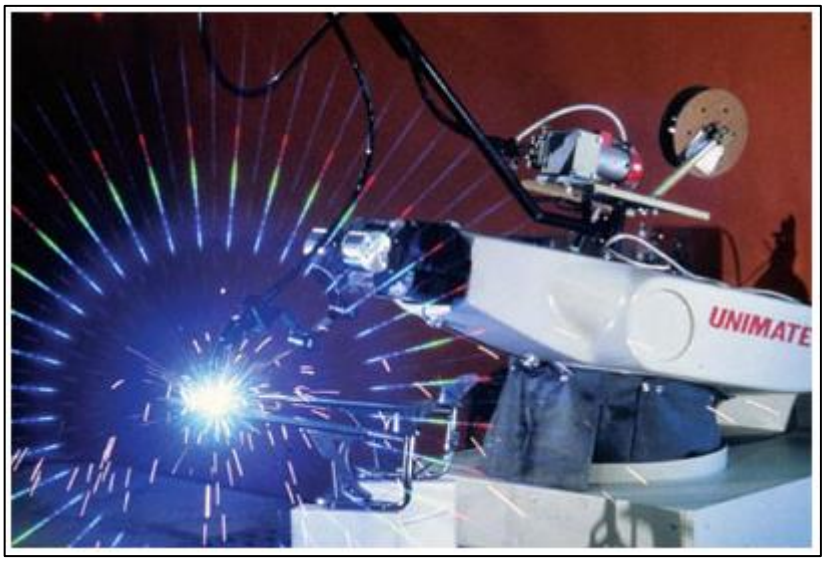
Os robôs, geralmente são projetados para realizar funções repetitivas e podem ser adaptadas para outras funções sem alteração permanente do equipamento. Uma máquina automatizada que faz apenas uma operação não é um robô, é simplesmente automação ou um manipulador. Um robô deve ter a capacidade de lidar com uma variedade de tarefas em uma fábrica.

A Robotic Industries Association (2015) destaca que o escritor Isaac Asimov foi o primeiro a utilizar a palavra robótica, por volta do ano 1950, mas quem realmente é reconhecido como o pai da robótica industrial é o inventor americano George Charles Devol Jr., que com a ajuda de Joseph Engelberger, construíram e desenvolveram o primeiro robô industrial chamado Unimate, cuja finalidade era manusear materiais.

Em 1954, o Unimate foi patenteado e em 1961 já era um braço robótico controlado digitalmente. Possuía um sistema de memória em estado sólido, servo controladores e fontes especiais de energia hidráulica e elétrica, e foi instalado

inicialmente em uma fábrica da General Motors em Nova Jersey nos Estados Unidos. Era utilizado para manusear metal em alta temperatura na fundição e também para realizar solda ponto nas carroçarias dos automóveis e a partir deste braço deu-se início ao processo que hoje é conhecido como robotização, revolucionando os processos industriais em todo o mundo. A Figura 1.1 ilustra o Unimate.

Figura 1.1 - Unimate o primeiro robô industrial



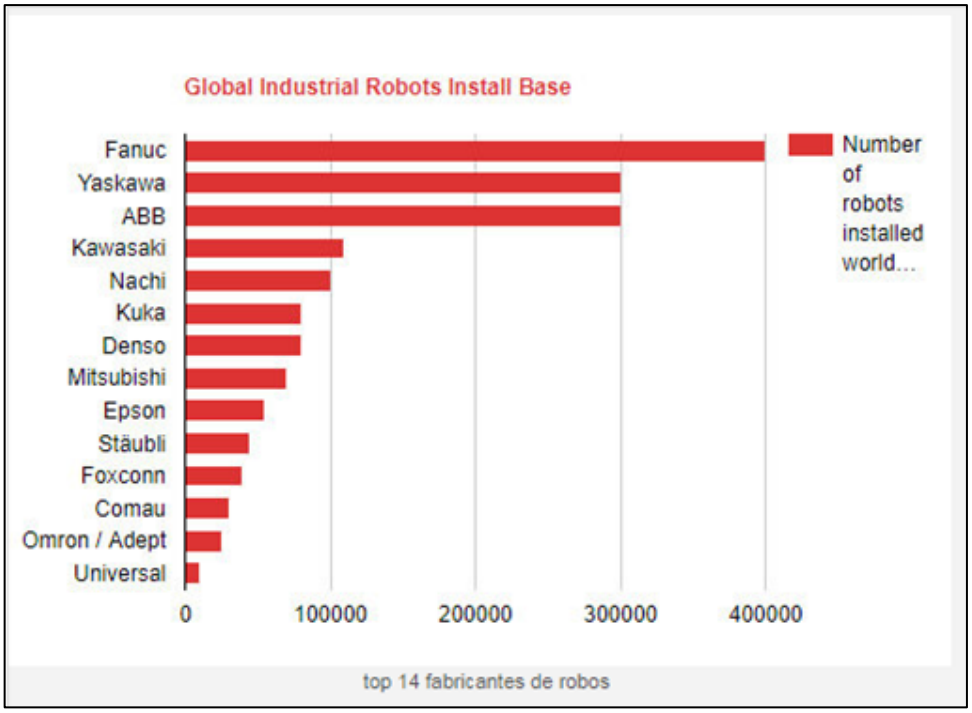
Fonte: www.robotics.org, 2018

A Robotic Industries Association (2015) destaca que o Unimate (robô) revolucionou a indústria automotiva, em um período muito curto, a General Motors havia pulado a frente de sua concorrência para se tornar a fábrica automotiva mais automatizada do mundo.

Em 1969 a General Motors reconstruiu sua planta fabril na cidade de Lordstown no estado de Ohio nos EUA, onde foram instalados os Unimate Spot Welding Robots no processo de solda ponto das carroçarias dos veículos, atingindo uma produção de 110 carros por hora, quantidade que superava em 100% suas concorrentes. Por este motivo os Europeus resolveram seguir o exemplo e montadoras como BMW, Volvo, Mercedes Benz, British Leyland e Fiat instalaram braços robóticos Unimate visando o aumento da produção nas mais diversas áreas de fabricação.

Montaqim (2015) enfatiza que foi no Japão onde os robôs foram aperfeiçoados, principalmente no setor automotivo e hoje o país está na vanguarda da avançada tecnologia robótica. Estima-se que atualmente existam cerca de 250 mil robôs trabalhando nas indústrias do país; que concentra os principais fabricantes de robôs no mundo, fornecendo 52% da oferta global, além disso o Japão também lidera a lista dos países mais robotizados do mundo, conforme Figura 1.2.

Figura 1.2 - Gráfico dos principais fabricantes de robôs e a quantidade instalada



Fonte: www.roboticsandautomationnews.com, 2015

O site Correio do Povo (2018) destaca os dados da Federação Internacional de Robótica, que mede o impacto do uso de robôs nos postos de trabalhos antes ocupados por humanos utilizando uma métrica de densidade, calculando a quantidade de robôs industriais instalados em cada nação comparadas com um grupo de dez mil operários empregados na indústria.

Em 2013 foram 178 mil robôs vendidos; em 2014, 221 mil; em 2015, 254 mil; em 2016, 294 mil e em 2017, 381 mil. Já em 2018 houve um crescimento sustentável na venda de robôs atingindo o montante de 381 mil comercializados e este volume de vendas representou um recorde para o setor com um aumento de 30% em relação ao

ano anterior quando foram vendidas 294 mil unidades. Os estudos apontam que até 2020 a quantidade de robôs industriais em uso espalhados por todo o mundo deve passar de 3 milhões.

A revista World Robotics (2017) aponta que os cinco maiores mercados na comercialização de robôs são China, Coréia do Sul, Japão, Estados Unidos e Alemanha, que representam 74% do mercado, com destaque para a China, que tem sido o maior mercado mundial desde 2013.

De acordo com o site Correio do Povo (2018), no cenário brasileiro, a comercialização de robôs em 2016 sofreu uma queda devido a situação política e econômica do país, que inviabiliza o investimento da indústria no setor, segundo fontes do Ministério da Indústria, comércio exterior e serviços, em 2016 a proporção de robôs industriais para 10 mil trabalhadores era de apenas 10, enquanto a média global era de 74 para esse mesmo número de empregados.

Freitas e Santos (2000) enfatizam que os robôs podem ser classificados em três gerações tecnológicas, na primeira eles podem repetir uma sequência de operações programadas específica para cada processo. A cada alteração de processo o robô é reprogramado. Nesta modalidade o ambiente de interação do robô deve ser completamente estruturado, pois as operações exigem o posicionamento preciso dos objetos a serem trabalhados, nos robôs desta primeira geração, não é possível a comunicação entre robôs.

Na segunda geração, o robô é capaz de se adaptar parcialmente a mudanças no ambiente através de sensores e recursos computacionais e consegue se comunicar com outros robôs. Na terceira geração faz uso intensivo de sensores, algoritmos de percepção e controle, utilizam inteligência artificial e possuem softwares que os tornam capazes de interagir com outros robôs, máquinas e pessoas, podem tomar decisões autônoma frente a situações não previstas quanto a seleção de peças entre outras funções.

1.2 Comunicação entre robôs e células de manufatura e de processos

Romano (2002) explica que por se tratar de uma máquina automática, o robô é capaz de reconhecer a célula de manufatura e de processos utilizando um conjunto de sensores, internos e externos, que possibilita a sua operação com precisão e segurança, dispensando o auxílio do operador quando em modo automático, como ilusta a Figura 1.3

Figura 1.3 - Interação de robôs com células de manufaturas

Fonte: www.todoproductividad.blogspot.com, 2013

O computador responsável pelo controle do robô recebe os sinais gerados por diversos sensores que se encontram instalados no mesmo. Ele é responsável pelo controle e análise das entradas e com os parâmetros de programação armazenados podem fornecer informações como: distâncias entre o robô e a peça a ser manipulada, o contato da garra com o objeto, a força que a garra exerce em contato com o mesmo, além de informações como a cor, textura e geometria da peça.

A malha de realimentação dos sensores interno da máquina requer um controle em tempo real da posição do robô que opera em períodos de amostragem na ordem de milissegundos. Os sensores externos ao robô utilizam uma malha de realimentação com as informações provenientes do meio externo e do processo, estas informações obtidas são utilizadas para a execução de uma tarefa num nível mais complexo.

Dentro de um período de amostragem da malha de realimentação externa devem existir muitos períodos de amostragem da malha de realimentação interna,

comprovando que a velocidade de processamento dos sensores internos é superior aos externos. Em contrapartida, os sensores externos possuem uma gama muito grande de aplicações e fornecem os mais variados tipos de informações dentro dos diversos processos.

Mandorsson (2013) diz que na maioria dos sistemas de automação avançados, as células robóticas precisam trocar dados com o meio externo e para que isso seja possível, são usados protocolos de comunicação industrial. Alguns fabricantes usam protocolos abertos e mais conhecidos como o DeviceNet, Profibus e EtherCAT, geralmente os dados enviados são de periféricos externos, por exemplo um sensor de proximidade, mas também podem ser informações de posição dos servos motores.

Segundo Schmitt et al. (2013), as empresas de automobilística estão investindo em células de soldagem robotizadas, para isto apresenta uma proposta de implementação usando ferramentas computacionais com foco na integração dos dispositivos com as redes de comunicação industrial. Para a troca dados entre o robô e a fonte de solda, o protocolo MODBUS é utilizado. Neste caso o robô atua como controlador master do sistema e não possui um protocolo MODBUS nativo, por isso são utilizadas as saídas digitais, que são capturadas por um computador, interpretadas e traduzidas para a fonte.

A Beckoff (2014) cita que em conjunto com uma fabricante de robôs desenvolveu bibliotecas de comandos para que os dados de um robô possam ser enviados diretamente para um PLC, onde não é necessário conhecimento de linguagem especial de programação de robôs. Com protocolo de comunicação EtherCAT, e o robô como mestre é utilizada uma interface para a tradução dos comandos de acionamento. O PLC recebe a informação de todo posicionamento do robô, através de um protocolo proprietário.

1.3 Utilização de sistemas de visão em robôs industriais

De acordo com Silveira (2017) a utilização de sistemas de visão em robôs industriais tem a finalidade de realimentação de sinais nos sistemas robóticos e tem despertado enorme interesse de pesquisadores e engenheiros desde a década de 70.

O sistema de visão aplicado à robótica utiliza comumente câmeras robustas, ideais para ambientes industriais, pois podem trabalhar com a orientação 2D ou 3D, proporcionam flexibilidade de programação de acordo com seu propósito de utilização, possuem a capacidade de detectar diferentes tipos de ondas eletromagnéticas, o que permite a identificação de objetos mesmo na ausência de luz visível. A Figura 1.4 mostra alguns modelos de câmeras utilizadas para este fim.

Figura 1.4 - Modelos de câmeras para uso industrial em robôs

(a) In-Sight 1020, da Cognex (b) NI-1764, da National Instruments (c) Iris GT1900, da Matrox

Fonte: https://revista-automacao.com

Steger e Markus (2007) citam que nos processos industriais automatizados, câmeras são usadas em diversas áreas e com diferentes finalidades. Em geral os sistemas de visão supervisionam, detectam e enviam informações críticas comunicando os resultados para o sistema de controle e monitoramento, que são processados e determinam a movimentação ou operação que o robô irá realizar.

A utilização deste tipo de sistema tem o objetivo de proporcionar um aumento de produção, melhoria da qualidade e agilidade dos processos e consequentemente a redução de custos, pois sua principal característica é o sistema de inspeção on-line no qual o robô pode realizar tarefas complexas e repetitivas em alta velocidade, com

alto grau de precisão e consistência de informações, garantindo também que nenhum tipo de colisão ocorra.

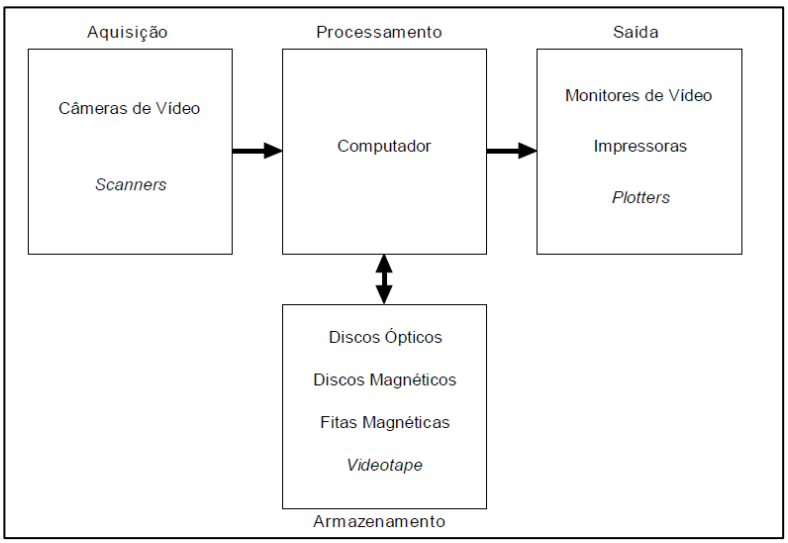
As indústrias automobilísticas, farmacêuticas, eletroeletrônicas e alimentícias são as que mais utilizam esta tecnologia, principalmente para inspeção visual automática, identificação de peças, controle de processos e células industriais e controle e direcionamento robótico.

1.4 Visão artificial e processamento de imagens

Segundo Marengoni e Stringhni (2009), uma linha muito tênue separa a visão computacional do processamento de imagens. No processamento de imagens são obtidos valores numéricos que podem ou não compor uma nova imagem. A visão computacional tem como objetivo emular a visão humana, interpretando a imagem de entrada parcialmente ou com um todo.

Em ambos os casos podem ser conceituados como um sistema computadorizado com capacidade de aquisição, processamento e interpretação de imagens correspondentes a imagens reais. A Figura 1.5 ilustra os elementos de um sistema de processamento de imagem.

Figura 1.5 - Elementos de um sistema de processamentos de imagem



Fonte: MARQUES FILHO e VIEIRA NETO, 1999, p. 2

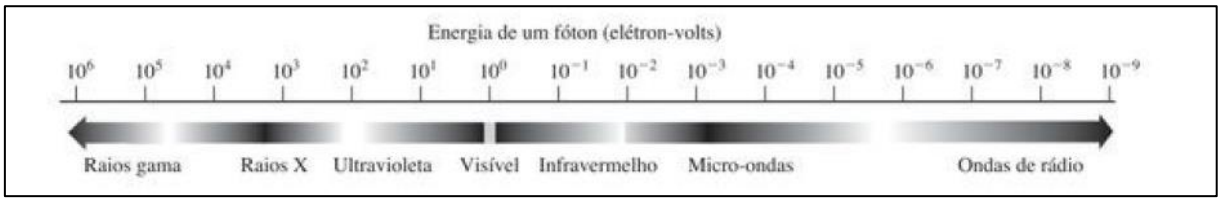
Marques Filho e Vieira Neto (1999) destacam a aquisição da imagem como processo de transdução optoeletrônica, ou seja, a conversão de uma imagem tridimensional em uma imagem analógica, onde um dispositivo capta uma faixa de energia do espectro eletromagnético e gera como saída um sinal elétrico proporcional ao nível de energia detectado.

De acordo com Jähne, Haußecker e Geißler (1999), para a observação, aquisição e processamento de uma imagem é necessária radiação, que pode ser emitida por uma fonte externa ou pelo próprio objeto de estudo.

Embora seja possível, a captação e processamento de imagens que utilizam fontes de radiação acústicas, ultrassônicas e eletrônica, que são feixes de elétrons utilizados em microscopia eletrônica, a fonte de radiação mais utilizada é a eletromagnética, que constiste em ondas que carregam energia e a propagam pelo espaço. Tais ondas também podem ser tratadas como um fluxo de partículas sem massa se deslocando em um padrão ondulatório na velocidade da luz com uma certa quantidade de energia chamada fotón.

Gonzalez e Woods (2010) apresentam o agrupamento das bandas do espectro de acordo com a energia de um fóton, onde os raios gama possuem a maior quantidade de energia e na outra extremidade as ondas de rádio possuem as menores, conforme mostra a Figura 1.6.

Figura 1.6 - Espectro eletromagnético organizado por energia por fóton



Fonte: GONZALEZ e WOODS, 2010, p. 5

As diferentes bandas do espectro podem gerar diferentes tipos de imagens para diferentes aplicações, conforme Tabela 1.1.

Tabela 1.1 - Aplicações das imagens das diferentes bandas do espectro eletromagnético

Banda espectral	Aplicação
Raios gama	Medicina nuclear, escaneamento ósseo, tomografias e observações astronômicas.
Raios X	Diagnóstico médicos, processos industriais e astronomia.
Ultravioleta	Microscopia fluorescente, imagens biológicas e até mesmo inspeções visuais.
Banda visível e infravermelho	Mais usual em captação de imagens, superando todas as outras em variação de aplicações, dentre elas microscopia ótica, sensoriamento remoto, fins industriais e de policiamento
Microondas	Radares.
Ondas de rádio	Astronomia e área da medicina, com ressonância magnética.

Fonte: GONZALEZ e WOODS, 2010, p. 20

Uma pequana faixa do espectro eletromagnético é visivel ao ser humano, que é totalmente dependente de luz para visualização e identificação de objetos. Esta luz pode ser provida pelos raios solares, luz da lua ou iluminação arificial, que o permite interagir com o ambiente ao seu redor.

Para Marques Filho e Vieira Neto (1999), três características dificultam aplicar aos robôs a capacidade visual de uma pessoa: uma enorme base de dados, uma altíssima velocidade de processamento e a capacidade de trabalhar em diversas situações.

Com os avanços tecnológicos, os meios de armazenamentos de massa possuem capacidades cada vez maiores e os processadores com velocidades mais rápidas. Porém, ainda há o desafio de fazer com que sistemas de visão artificial possam trabalhar com a mesma eficácia em diversos níveis de luminosidade e posição relativa de objetos sem que o contexto da imagem seja alterado.

Um sistema de visão artificial se mostra superior a visão humana no que diz respeito a detecção de todas as faixas do espectro eletromagnético, dos raios X ao infravermelho, na detecção de cores e com medições mais precisas através da contagem de pixels.

Por outro lado, a visão humana se mostra superior em tarefas 3D, mais sensível e flexível realizando tarefas em diferentes níveis de iluminação, enquanto a visão robótica apresenta um bom desempenho apenas para a tarefa ao qual foi projetado.

1.5 Digitalização de imagens

De acordo com Barelli (2018), as câmeras digitais seguem o mesmo principio das câmeras convencionais, onde um pequeno orifício com uma lente converge os feixes de luz da cena ou objeto para dentro de uma câmara escura, projetando uma imagem invertida na face contrária ao orifício. Nas câmeras analógicas, filmes químicos sensíveis a luz registravam a imagem.

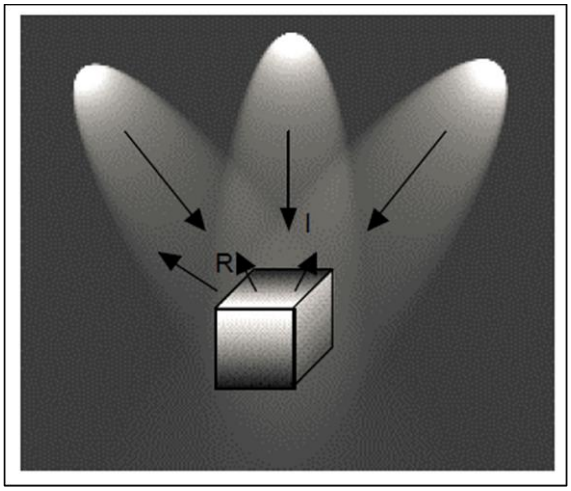
Marques Filho e Vieira Neto (1999) explicam que nas câmeras digitais a intensidade luminosa é convertida em sinais elétricos análogicos e em seguida circuitos eletrônicos chamados *frame grabbers* digitalizam tal informação.

O processo de digitaização inicia-se a partir da descrição de uma imagem monocromática pela função *f(x,y)* que é o resultado do produto da iluminância (i), que é a quantidade de luz que incide sobre o objeto e a refletância ou transmitância (r) característica própria do objeto que é a quantidade de luz refletida pelo mesmo, conforme mostra a Equação 1.1 a seguir:

$$f(x,y) = i(x,y).r(x,y) \qquad [1.1]$$

A Figura 1.7 ilustra os componentes iluminância (I) e refletância (R) de uma imagem.

Figura 1.7 - Componentes iluminância (I) e refletância (R) de uma imagem



Fonte: MARQUES FILHO e VIEIRA NETO, 1999, p. 20

A imagem obtida, convertida em sinal analógico passa por uma amostragem, que a divide em uma matriz de duas dimensões (M e N), onde M representa o número de linhas e N o número de colunas, cada ponto da matriz é denominado pixel. Desta forma, quanto maiores forem os valores de M e N, maior é a quantidade de pixels e consequentemente maior é a resolução da imagem.

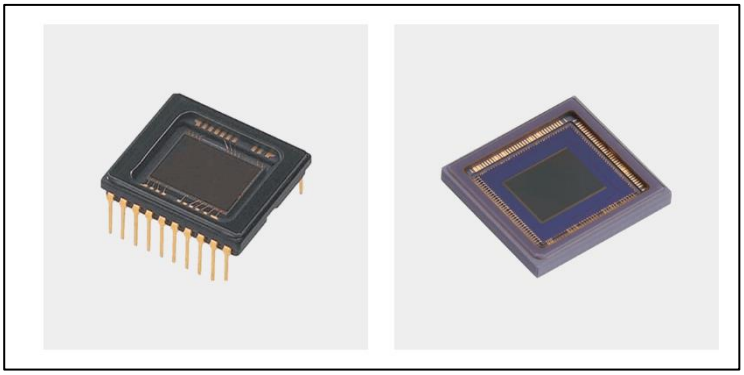
Outro processo ao qual o sinal analógico é submetido é a quantização, que atribui a cada pixel um número inteiro entre 0 a 2^n-1 que representa os níveis de cinza da imagem. Quanto maior for n, maior são os níveis de cinza da imagem. A quantidade de pixels e os níveis de cinza de uma imagem são diretamente proporcionais à quantidade de bytes necessários para seu armazenamento.

De acordo com Thomazini e Albuquerque (2007), dentre as tecnologias atuais mais utilizadas, destacam-se os sensores CCD, baseado em pastilhas semicondutoras dispostas em células que armazenam cargas elétricas proporcionais as luminosidade captada. Eles apresentam como desvantagem a falta de um sistema de controle integrado.

Os sensores CMOS captam a luz da mesma forma que os sensores CCD, porém apresentam como vantagem a integração do sistema de controle no mesmo

circuito, maior velocidade, baixa tensão de alimentação e menor custo de fabriação. A Figura 1.8 ilustra ambos os sensores.

Figura 1.8 - Sensores CCD e CMOS



Fonte: BARELLI, 2018, p. 26

1.6 Representação das cores

De acordo com Barelli (2018), a representação digital mais simples de uma imagem é a forma binária, onde o valor 0 corresponde a cor preta e o valor 1 corresponde a cor branca. Considerando que cada pixel representa a intensidade de luz captada pelo sensor, para imagens binárias é necessário determinar um valor analógico que serve de limite para determinar se o pixel é branco ou preto. A esse valor é dado o nome de limiar e o processo de conversão de imagens para preto e branco de acordo com este valor é chamado limiarização. As imagens binárias são muito utilizadas para segmentação de objetos e extração de características.

As imagens também podem ser representadas em tons de cinza, onde cada pixel pode ser representado por 8 bits, sendo possível 256 tons diferentes (de 0 a 255) para cada pixel. Imagens em tons de cinza são amplamente utilizadas para reconhecimento ou contagem de objetos onde a cor não é importante.

As imagens em cores reais são consideradas três matrizes bidimensionais sobrepostas, conhecidas como RGB (*Red* – vermelho, *Green* – verde e *Blue* – azul), onde uma matriz representa a intensidade da cor vermelha, a outra a intensidade da cor verde a última a cor azul, cada uma com 8 bits. Desta forma para representar um

pixel são necessários 24 bits, ou seja, são 16,8 milhões de cores diferentes. A Figura 1.9 apresenta tais sistemas de cores.

Figura 1.9 - Representação de cores

Fonte: BARELLI, 2018, p. 42

1.7 Propriedades e operações com imagens digitais

Segundo Marques Filho e Vieira Neto (1999), uma imagem monocromática adquirida e digitalizada pode ser vista como uma matriz, onde as linhas e colunas identificam um ponto *p* (pixel) de coordenada *(x,y)* na imagem. Os pixels que estão ao seu redor são denominados vizinhança.

Para a detecção de bordas ou de objetos de uma imagem, são analisados a vizinhança de um pixel e, se algum pixel adjacente a este tiver um nível de cinza dentro de uma faixa de similaridade, podemos dizer que eles estão conectados.

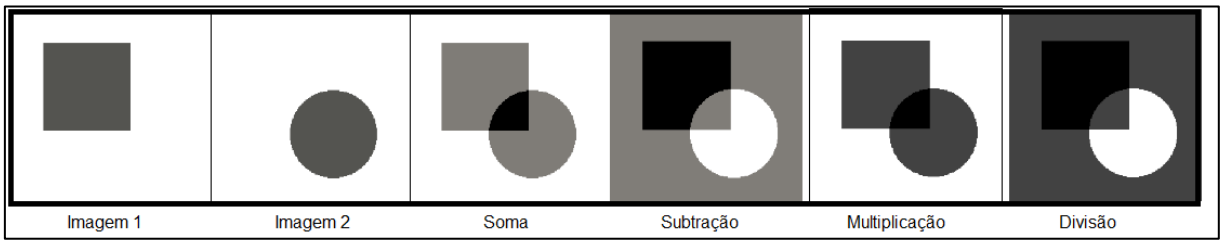
O cálculo de distância entre dois pixels, é comumente feito através do cálculo da distância euclidiana (D_e). Considerando dois pixels *p* e *q* de coordenadas *(x,y)* e *(s,t)* respectivamente, o cálculo da distância euclidiana será calculada conforme Equação 1.2 a seguir:

$$D_e = \sqrt{(x-s)^2 + (y-t)^2} \qquad [1.2]$$

Considerando ainda imagens digitalizadas como matrizes de números inteiros, é possível realizar operações lógicas ou aritméticas de duas imagens de tamanhos iguais gerando uma terceira cujo os pixels equivalem ao resultado da operação pixel a pixel das imagens originais.

Dentre as operações aritméticas, a soma é muito utilizada na normalização de brilhos e filtragem para remoção de ruídos; a subtração tem a função de detectar diferenças entre duas imagens da mesma cena, sejam elas consecutivas ou não; a multiplicação é utilizada na calibração de brilho, que adequa a faixa total dos níveis de cinza a uma faixa pré-determinada e a divisão é um recurso utilizado na normalização do brilho que pode adequar os diversos níveis de iluminância de uma cena. A Figura 1.10 apresenta as imagens originais 1 e 2 e os resultados das operações aritméticas.

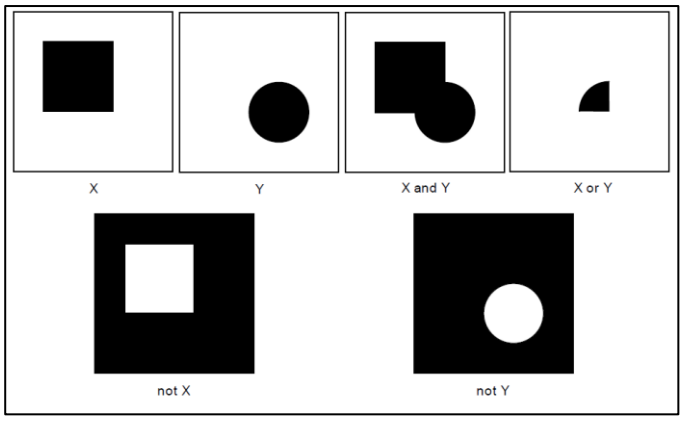
Figura 1.10 - Operações aritméticas



Fonte: MARQUES FILHO e VIEIRA NETO, 1999, p. 31

As operações lógicas ou booleanas podem ser utilizadas em imagens, porém a representação da imagem binária (em preto e branco) em detrimento da imagem em escala de cinza, nos permite uma melhor compreensão dos resultados. A Figura 1.11 apresenta as operações lógicas aplicadas nas imagens X e Y do exemplo.

Figura 1.11 - Operações lógicas



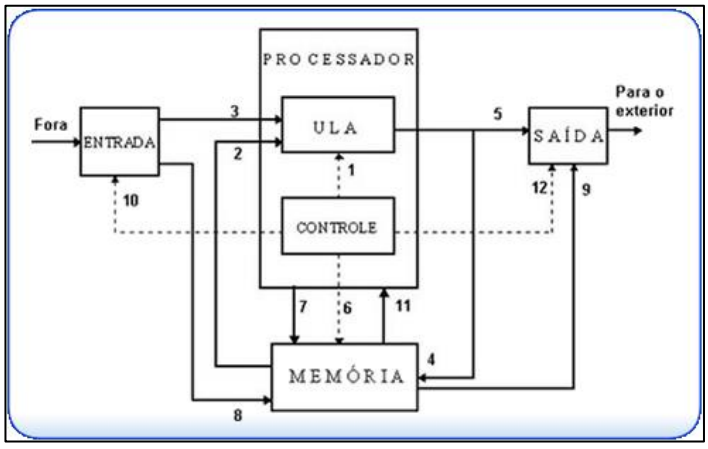
Fonte: MARQUES FILHO e VIEIRA NETO, 1999, p. 32

1.8 Programação

Manzano e Oliveira (2009) citam que o surgimento e desenvolvimento da computação está atrelado ao atendimento da necessidade de facilitar o uso e manuseio de operações de cálculos

O computador é um dispositivo eletrônico constituído por circuitos integrados embarcados em chips, controlado por um programa, capaz de receber, armazenar, tratar e produzir informações de forma automática, de maneira rápida e precisa. Sua organização básica é constituída por entrada (dados), processador, memória e saída (informação), conforme ilustrado na Figura 1.12.

Figura 1.12 - Organização básica de um computador



Fonte: www.uab.ifsul.edu.br, 2018

O processamento de dados é flexível, ou seja, é capaz de ser adequado às mais diversas necessidades de aplicação de um computador, através do desenvolvimento de diferentes sequências lógicas, que resultam em diferentes informações. A sequência lógica de ações a serem executadas para a realização de uma tarefa é chamada algoritmo.

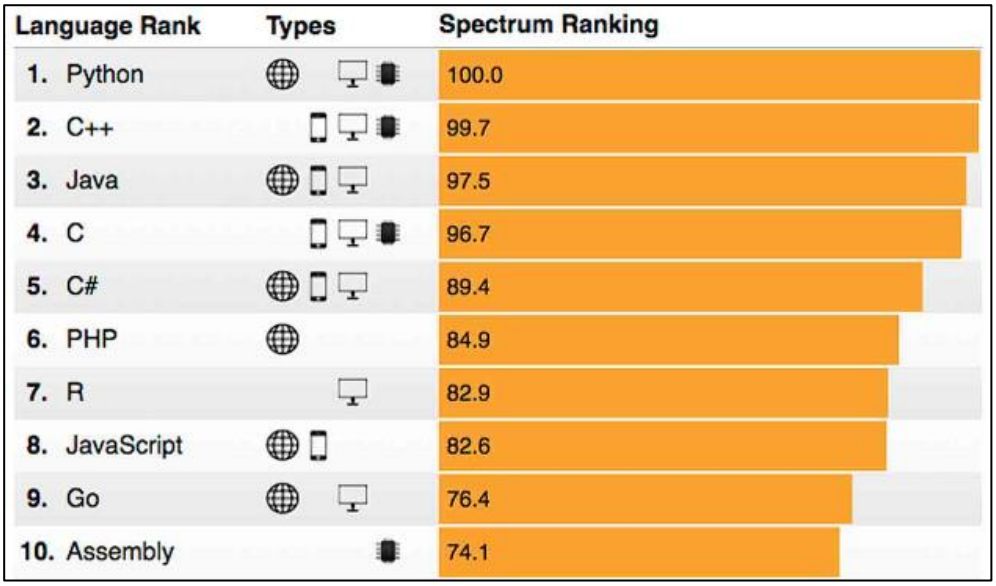
De acordo com Gudwin (1997), a implementação de um algoritmo em um computador necessita de uma prévia tradução do mesmo para que ele possa executar. Isso é feito através de uma linguagem de programação, que é um conjunto de símbolos e regras que permitem a execução de sequências lógicas por um computador.

Existem diversos tipos de linguagens de programação para utilização em ambiente doméstico, administrativo e industrial, sendo a divisão mais comum em relação ao seu nível de complexidade. Uma linguagem de baixo nível possui instruções que correspondem quase que diretamente ao código de máquina que é processado. A linguagem de alto nível é mais próxima à linguagem utilizada e compreendida pelo ser humano.

Segundo Sebesta (2018), as linguagens de programação podem ser implementadas de três maneiras, sendo a mais comum, a implementação baseada em compilação, onde o programa é traduzido para o código da máquina, chamada de linguagem-fonte do computador. Após a compilação, torna-se possível executar o programa.

Segundo o Instituto de Engenheiros Eletricistas e Eletrônicos - IEEE (2018), a linguagem mais utilizada em 2018 é a Python, superando a liguagem C. A Figura 1.13 mostra o ranking, das linguagens de programação em 2018.

Figura 1.13 - Ranking IEEE das linguagens de programação 2018



Fonte: www.spectrum.ieee.org, 2018

Labaki (2017) cita que o Python é uma linguagem de programação criada pelo holandês Guido Van Rossum sob o ideal de acessibilidade, por isso essa linguagem é de altíssimo nível, que tem como vantagem a legibilidade dos programas e sua fácil interpretação, e tem código aberto (gratuito), o que tornou-se um propulsor de sua utilização.

Segundo Menezes (2014), outras vantagens apresentadas por essa linguagem é a possibilidade de programar na forma procedural ou na forma orientada a objetos, que facilita o controle sobre a estabilidade de grandes projetos, e sua modularidade, possibilitando o fácil acesso e utilização de bibliotecas já criadas.

Atualmente, utiliza-se Python em praticamente qualquer sistema operacional como Linux, Microsoft Windowns ou Mac OS, e devido a essas vantagens, a utilização dessa linguagem vem crescendo em diversas áreas como computação, biotecnologia, economia, animação 3D, jogos, aplicativos móveis e automação.

1.9 Biblioteca para processamento de imagens

De acordo com Bradski e Kaehler (2008), uma das bibliotecas de visão computacional utilizadas com Python é a OpenCV, que pode ser usada com outras linguagens de programação e é compatível com diversos sistemas operacionais. Ela foi desenvolvida com o objetivo de possibilitar a construção de diversas aplicações de maneira simples, por isso uma de suas vantagens é o código aberto, ou seja, gratuito.

Com mais de 500 funções disponíveis, a OpenCV é indicada para aplicações de visão em tempo real em diferentes áreas como na medicina, no processamento de imagens e diagnósticos; na segurança, atuando no controle de acesso e rastreamento; na indústria, em inspeções e manipulação robótica de produtos.

Segundo Marengoni e Stringhni (2009), o OpenCV possui ferramentas aplicáveis no processamento inicial de uma imagem, quanto em aplicações de visão computacional a partir de um vídeo ou monitoramento instantâneo.

Na etapa de processamento, é possível a aplicação de diversos filtros, que de acordo com a aplicação, removem ruídos gerados no processo de captação e que atrapalham a interpretação ou reconhecimento de objetos da imagem.

Com o OpenCV também é possível realizar alterações no tamanho, e formato da imagem a ser utilizada, além de melhora de definicão, compressão, transformações de intensidade e binarização (conversão dos pixels em branco e preto), técnica muito utilizada para isolar objetos de interesse e componentes conectados.

Já na área de visão computacional, o OpenCV possui ferramentas para detecção de bordas, reconhecimento de padrões e consequentemente, reconhecimento de objetos, assim como diversas funções para rastreamento, onde além da identificação do objeto, é possível realizar um modelamento da trajetória para agilizar o processamento.

2 METODOLOGIA

Neste capítulo encontra-se a trajetória para o desenvolvimento e construção do projeto que se intitula Sistema de Correção de Coordenadas Dinâmicas para Robôs. Trata-se de uma pesquisa aplicada que é desenvolvida nas dependências da Fatec São Bernardo do Campo e nas residências dos integrantes do grupo.

Dentre os vários autores que tecem teorias sobre metodologia científica, Prodanov e Freitas (2013) enfatizam que a metodologia consiste em estudar, compreender e avaliar os métodos disponíveis para realização de uma pesquisa acadêmica, tornando necessário a aplicação de técnicas para a construção do conhecimento. Destaca que os métodos são procedimentos amplos do raciocínio, enquanto as técnicas são procedimentos mais restritos que operacionalizam os métodos mediante emprego de instrumentos adequados.

Severino (2016) aponta que a metodologia é a preparação metódica e planejada de um trabalho científico que se divide em momentos ou etapas como: o tema-problema e justificativa, levantamento bibliográfico referente ao tema, leitura e seleção dessa bibliografia para a construção da fundamentação teórica, construção e modelagem do projeto e redação do texto.

A redação do texto tem como base as normas da ABNT que se encontra no Manual de Normalização de Projeto de Trabalho de Graduação da Fatec São Bernardo do Campo (2017), que dá suporte para concretizar o projeto. A escrita do texto é uma linguagem simples, específica, concisa, com terminologia adequada, seguindo um raciocínio lógico.

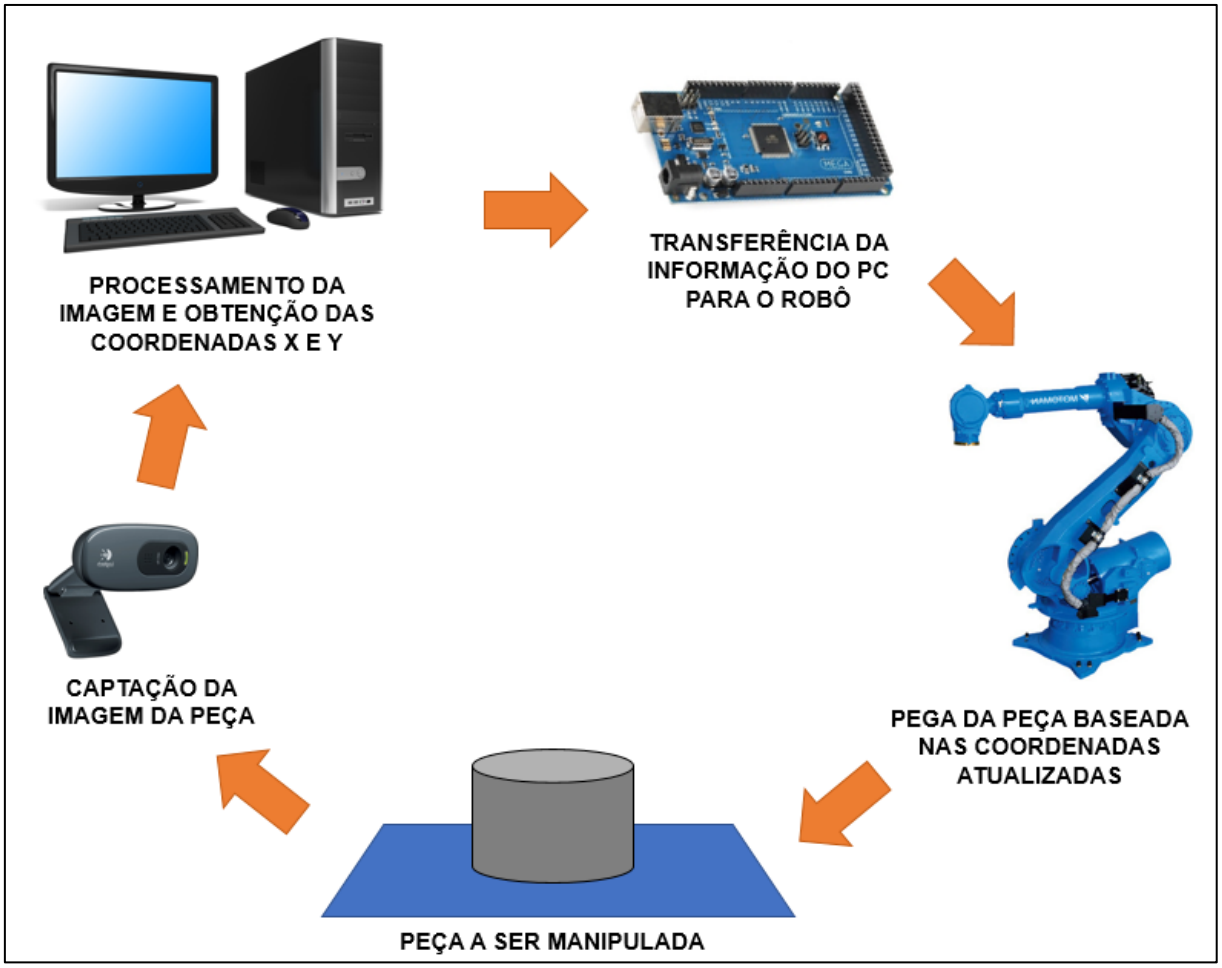
2.1 O tema-problema com justificativa e descrição do projeto

O objetivo deste trabalho que se intitula Sistema de Correção de Coordenadas Dinâmicas para Robôs é elaborar um sistema de visão aplicado à célula de manufatura robótica da Faculdade de Tecnologia Adib Moisés Dib para transmitir ao robô correções de coordenadas dinamicamente. Justifica-se por tornar processos

industriais mais flexíveis, uma vez que a utilização de um robô com um sistema de correção de coordenadas admite variações de posicionamento de peças que são manipuladas, pintadas e soldadas.

Sua elaboração consiste em uma câmera fixa sobre uma mesa, que capta a imagem de uma peça, localizada dentro de uma área pré-estabelecida de 300 mm x 300 mm. Através do processamento desta imagem por um computador, obtém-se os valores das coordenadas X e Y de posicionamento da peça. Tais valores são enviados a uma interface de comunicação, que converte a informação para o protocolo utilizado pelo robô e transmite para ele. Com esses dados, o ponto de pega da peça pelo robô tem suas coordenadas reprogramadas e a manipulação da peça ocorre sem nenhum tipo de colisão devido à alteração de seu posicionamento, como exemplificado na Figura 2.1.

Figura 2.1 - Diagrama de funcionamento

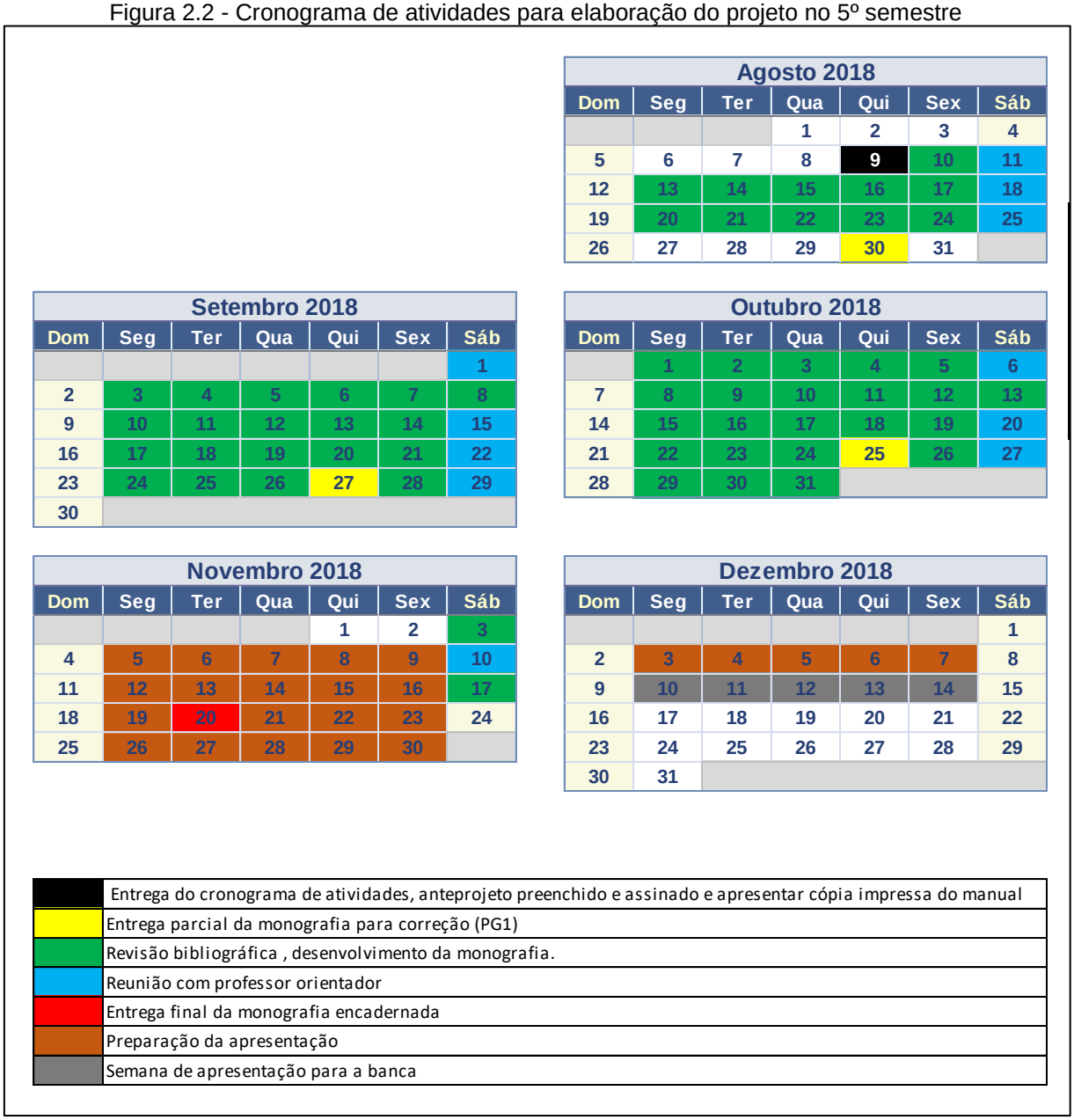


Fonte: Autoria própria, 2018

2.2 Etapas teóricas e práticas para elaboração do projeto

No quinto semestre o trabalho do grupo se focou nos elementos pré-textuais, fundamentação teórica, referencias e parte da metodologia cientifica. Além disso uma apresentação foi desenvolvida para uma apresentação sobre o desenvolvimento e funcionamento do projeto a ser construído. Durante este período o orientador acompanhou e contribuiu em todas as etapas do processo. A Figura 2.2 mostra o cronograma que o grupo seguiu durante o quinto semestre letivo.

Figura 2.2 - Cronograma de atividades para elaboração do projeto no 5º semestre

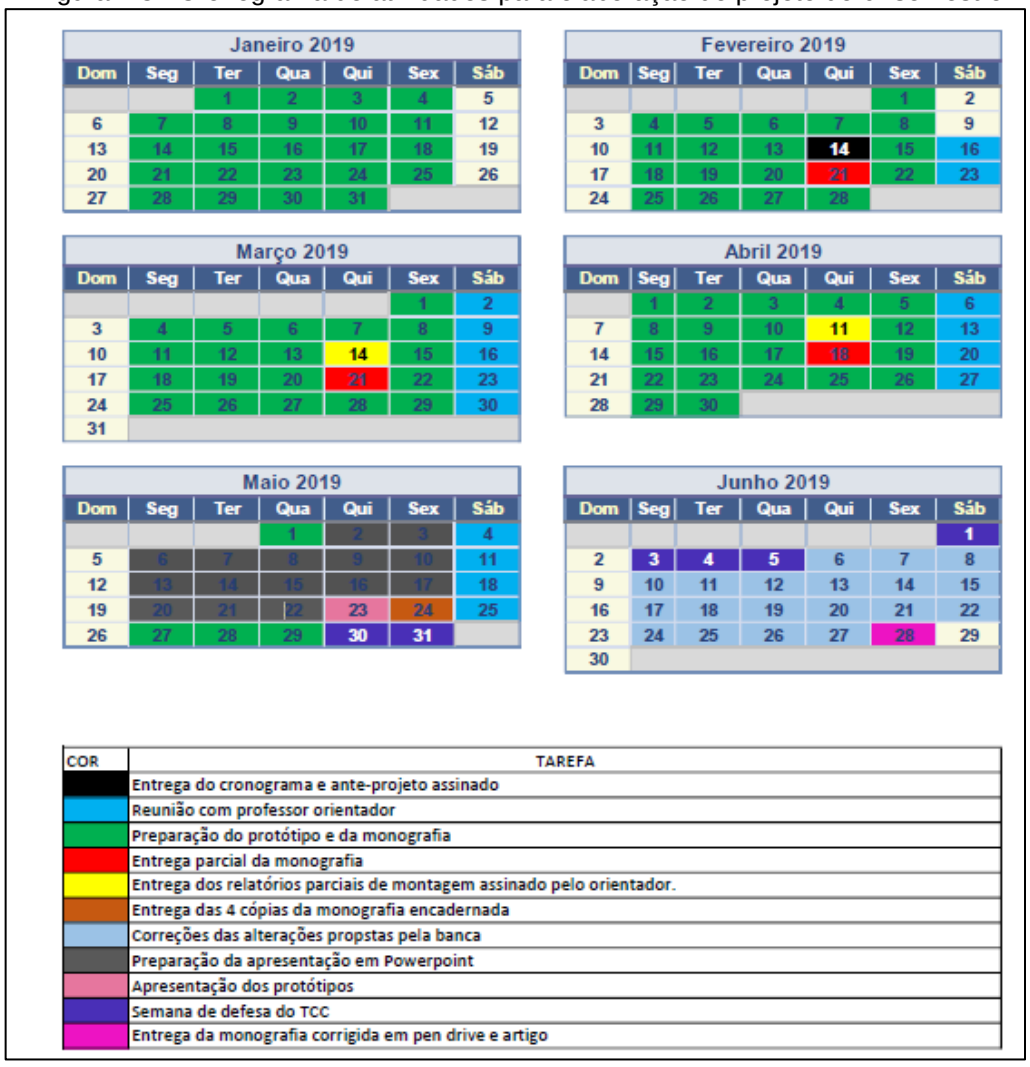


Fonte: Autoria própria, 2018

No sexto semestre, o grupo se dedicou a conclusão da metodologia científica, no desenvolvimento do projeto, incluindo o ambiente de captação da imagem, a programação do sistema de visão em linguagem Python, a construção e programação da interface para a comunicação entre o sistema de visão e o robô, os testes de funcionamento e integração das partes, as dificuldades e soluções propostas, além das considerações finais sobre o projeto.

Neste ponto, a apresentação de slides foi atualizada para a explanação sobre o desenvolvimento do sistema já concluído. Durante todo o processo, o orientador acompanhou e contribuiu em todas as etapas do projeto. A Figura 2.3 apresenta o cronograma seguido no sexto semestre.

Figura 2.3 - Cronograma de atividades para elaboração do projeto do 6º semestre



Fonte: Autoria própria, 2019

Após delimitar o tema-problema, justificativa e descrição do funcionamento do projeto parte-se para as seguintes etapas de acordo com o cronograma da Figura 2.2:

Primeira etapa: reunião dos integrantes do grupo com o orientador para traças as diretrizes de como efetuar as pesquisas. O orientador fez uma breve explanação sobre o tema indicou bibliografias e colocou-se a disposição para atender o grupo quando solicitado e marcou, obrigatoriamente um dia por semana para lhe apresentar o andamento da pesquisa;

Segunda etapa: o levantamento bibliográfico deu-se na biblioteca da FATEC São Bernardo do Campo, em sites especializados, manual de fabricante de robôs e pesquisas em trabalhos acadêmicos de outras universidades;

Terceira etapa: após leitura e releitura das bibliografias pesquisadas, fez se a seleção daquelas que se encontram mais próxima do tema proposto e constrói-se o Capítulo 1- Fundamentação teórica e as Referências;

Quarta etapa: levantamento dos materiais para serem usados na construção do projeto. Pesquisas de preços em sites e empresas especializadas do ramo. Estudo da viabilidade econômica. Aquisição conforme Tabela 2.1.

Tabela 2.1 - Materiais utilizado na confecção do projeto

Produto	Quantidade	Valores (R$)
Placa de perfil de alumínio	1	700,00
Trolley FESTO	1	500,00
Perfil Estrutural em Alumínio 30x30 Básico	1	35,00
Webcam HD Logitech C270 720p	1	150,00
Computador (com teclado, mouse, monitor sistema operacional Windows 10 32 bits	1	1000,00
Arduino Black Board Mega 2560	1	170,00
Placa de Relés com 16 reles	2	190,00
TOTAL		2.750,00

Fonte: Autoria própria, 2019

Quinta etapa: montagem do ambiente de captação com perfil estrutural, placa de perfil de alumínio e fixação da webcam;

Sexta etapa: programação do sistema de visão em Python para exibição da imagem digitalizada pela webcam, captura e processamento da imagem e extração dos valores de correção e envio para a interface de comunicação;

Sétima etapa: desenvolvimento de esquema elétrico e montagem do interface de comunicação, com conexão da placa Arduino Black Board com as placas de relés e conexão das saídas dos relés na borneira correspondente ao conector CN8 do robô;

Oitava etapa: programação do Arduino para conversão de dados recebidos via comunicação serial como *string* para envio de valores binários de forma paralela, e testes de comunicação com o robô;

Nona etapa: integração do sistema de visão com a interface de comunicação e o robô, com testes finais e descrição do funcionamento;

Décima etapa: finalizando o desenvolvimento do projeto faz-se as Considerações finais e o Resumo.

3 DESENVOLVIMENTO DO PROJETO

Neste capítulo encontra-se passo a passo a construção e desenvolvimento do projeto intitulado Sistema de Correção de Coordenadas Dinâmicas para Robôs.

Para melhor visualização e compreensão do projeto a Figura 3.1 ilustra-o finalizado.

Figura 3.1 - Projeto finalizado

Fonte: Autoria própria, 2019

Através de uma câmera posicionada sobre a área em que o robô busca pela peça a ser manuseada, é retirada uma fotografia e com o uso de algumas rotinas de processamento de imagens, as coordenadas atuais da peça são comparadas com as coordenadas de referência da posição de pega salva pelo robô.

A diferença dos valores das coordenas em X e Y são enviadas para o Robô através de uma interface microprocessada desenvolvida para o trabalho. Essa interface recebe os dados do software do sistema de visão e envia através de relés para as entradas digitais da interface de IO do robô MOTOMAN da FATEC São Bernardo do Campo, que recebe este sinal e corrige em seu programa os valores de pega da peça a ser manuseada.

A execução do desenvolvimento do projeto encontra-se amparado nos seguintes tópicos:

- ambiente de captação da imagem;
- sistema de visão;
- interface para comunicação entre o sistema de visão e o robô;
- testes de funcionamento e integração das partes;
- dificuldades encontradas e soluções.

3.1 Ambiente de captação da imagem

Para interação do projeto com o robô MOTOMAN da FATEC São Bernardo do Campo é necessário um ambiente de captação de imagens, onde as peças a serem manipuladas são dispostas para que o robô as pegue.

Todo o ambiente é construído sobre um *trolley* de 750 mm de altura, 350 mm de largura e 700 mm de profundidade, confeccionado em alumínio, com rodízios para que possa ser movimentado.

Sobre o *trolley* é fixado uma placa de perfil de alumínio, de 350 mm de largura e 700 mm de profundidade, com canaletas de alumínio para organização de cabos e ranhuras para fixação de componentes.

O suporte da câmera é composto por duas barras de perfil estruturado em alumínio de 30 mm por 30 mm, que são fixos entre si e na placa de perfil de alumínio com um conector universal 30, um sistema com pino, mola e parafuso próprio para tal aplicação. A Figura 3.2 ilustra o perfil estruturado e o conector universal em destaque:

Figura 3.2 - Perfil estruturado de alumínio e conector universal

Fonte: Autoria própria, 2019

No suporte é fixada uma webcam HD Logitec modelo C270, com resolução máxima de 1280 x 720 pixels, foco automático, campo de visão de 60º e cabo de 1,5 m para conexão USB 2.0.

Sobre a placa de perfil de alumínio utiliza-se uma placa de acrílico amarela, com o intuito de determinar a área útil, e ainda favorecer o contraste necessário para a detecção da peça pela câmera. O conjunto completo é constituído pelo *trolley*, a placa de perfil de alumínio, as barras de alumínio e a câmera, que forma o ambiente de captação de imagem, conforme ilustra a Figura 3.3.

Figura 3.3 - Ambiente completo de captação de imagem

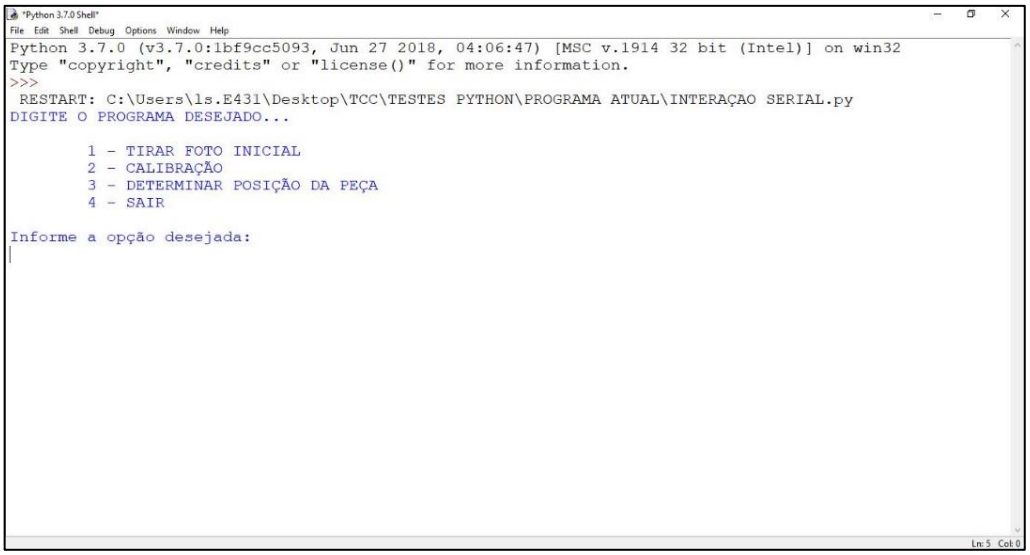
Fonte: Autoria própria, 2019

3.2 Sistema de visão

Após a construção do ambiente de captação de imagem, dá-se início o sistema de visão, que tem como objetivo determinar as coordenadas da peça a partir do processamento da captação da imagem. Para captação, digitalização e processamento de imagens, a webcam do ambiente de captação de imagens é conectada a um computador com sistema operacional Windows 10 de 32 bits. Neste estão instalados a IDE da linguagem de programação Python, versão 3.7.0 com suas diversas bibliotecas próprias, além das bibliotecas OpenCV versão 3.2, que possui funções específicas para tratamento de imagens, pyserial versão 3.4 para comunicação via serial e biblioteca Numpy para realização de cálculos em matrizes multidimensionais.

No ambiente de desenvolvimento do Python é criado o programa que se encontra no Apêndice A, com a finalidade de interagir com o usuário e realizar as funcionalidades do sistema de visão. Ao ser iniciado o programa carrega as bibliotecas acima citadas utilizando o comando *import*, que disponibiliza suas funções para o programa escrito. Em seguida, um menu para interface com o usuário é apresentado com as três funções disponíveis. A Figura 3.4 a seguir ilustra tal menu

Figura 3.4 - Interface de comunicação com o usuário



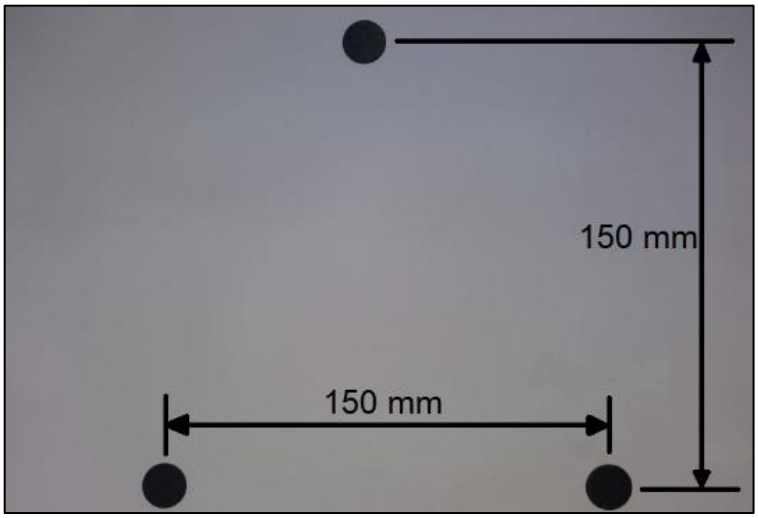
Fonte: Autoria própria, 2019

A primeira opção do menu (1- TIRAR FOTO INICIAL) é utilizada com o intuito de tirar uma foto da área de manipulação vazia. Esta foto é utilizada como imagem de referência para o processo de subtração de imagens descrita por Marques Filho e Vieira Neto (1999) e citada no capítulo 1.7 deste trabalho, por este motivo, este procedimento é repetido a cada alteração do ambiente de captação de imagem.

Através do comando *cv2.VideoCapture* o computador abre a conexão com a *webcam*, e exibe o vídeo em tempo real em uma janela chamada FOTO BASE e ao pressionar a tecla barra de espaço, a imagem apresentada na tela da área de manipulação vazia é captada e armazenada em uma variável de tipo *array* (matriz) na memória *RAM* do computador e em seguida a variável *array* é salva como uma foto com o nome FOTO 0.png no *Hard Disk* do computador. Ao pressionar a tecla ESC duas vezes, o programa retorna ao menu inicial.

A segunda opção do menu (2- CALIBRAÇÃO) realiza a calibração do sistema de visão, convertendo pixels em distância em milímetros. Este procedimento é realizado a cada inicialização do programa ou alteração física no ambiente de captação da imagem. Para isso é utilizado um padrão de calibração criado no *software* Inventor, que consistem em três círculos de 15 mm de diâmetro, posicionados nos vértices de um triângulo isósceles com 150 mm na base e 150 mm de altura e impresso em uma folha A4, conforme ilustra a Figura 3.4.

Figura 3.5 - Modelo do padrão de calibração



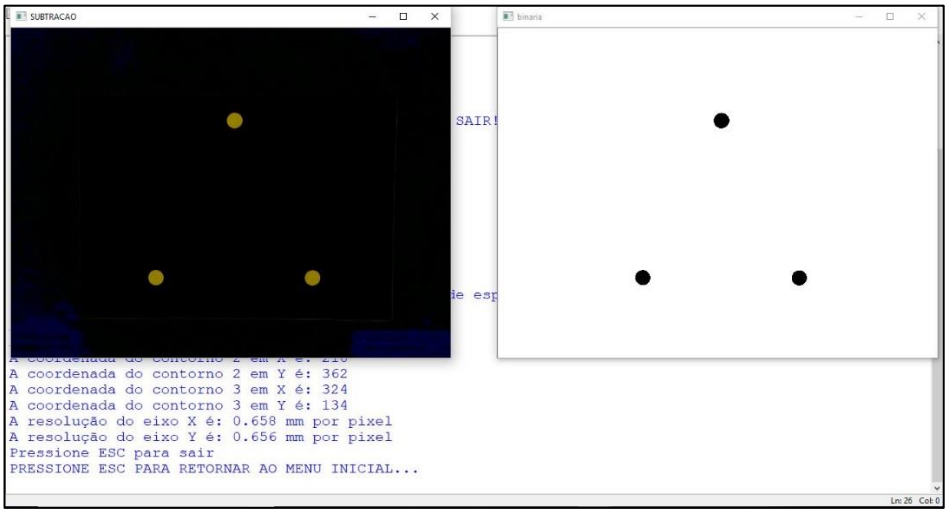
Fonte: Autoria própria, 2019

Após selecionar a opção 2 do menu, o computador novamente abre a conexão com a *webcam*, e exibe o vídeo em tempo real em uma janela chamada CALIBRAÇÃO. O padrão de calibração é posicionado na área de manipulação da peça e ao pressionar barra de espaço, a imagem apresentada na tela neste momento é captada e armazenada em uma nova variável de tipo *array* (matriz) na memória *RAM* do computador e em seguida esta variável *array* é salva com o nome FOTO 1.png no *Hard Disk* do computador.

Conforme descrito por Marques Filho e Vieira Neto (1999), o processo de subtração ressalta as diferenças entre as imagens, desta forma ao carregar as imagens armazenadas FOTO 0.png da área de manipulação vazia e FOTO1.png do padrão de calibração sobre a área de manipulação em duas matrizes e subtrair uma da outra, o padrão é destacado e salvo em uma nova matriz chamada *imgsub.* Esta nova matriz *imgsub* é convertida para tons de cinza e em seguida passa pelo processo de binarização, para facilitar o processamento, pois trabalha com uma matriz de uma dimensão ao invés de 3 como no sistema RGB.

A binarização avalia a matriz pixel a pixel e aqueles que estão acima de um valor de cinza determinado o converte em preto e abaixo deste valor determinado em branco. A Figura 3.6 a seguir ilustra o resultado dos processos de subtração e binarização:

Figura 3.6 - Foto capturada do padrão binarizada



Fonte: Autoria própria, 2019

Em seguida, utilizando a função cv2.findContours são determinados os contornos e as coordenadas cartesianas em pixel do centro de cada um dos círculos do padrão de calibração, tendo como origem do sistema cartesiano a extremidade superior esquerda da imagem, determinados da seguinte forma:

- D_{x1} – Coordenada em X do círculo 1 em pixels;
- D_{x2} – Coordenada em X do círculo 2 em pixels;
- D_{x3} – Coordenada em X do círculo 3 em pixels;
- D_{y1} – Coordenada em Y do círculo 1 em pixels;
- D_{y2} – Coordenada em Y do círculo 2 em pixels;
- D_{y3} – Coordenada em Y do círculo 3 em pixels.

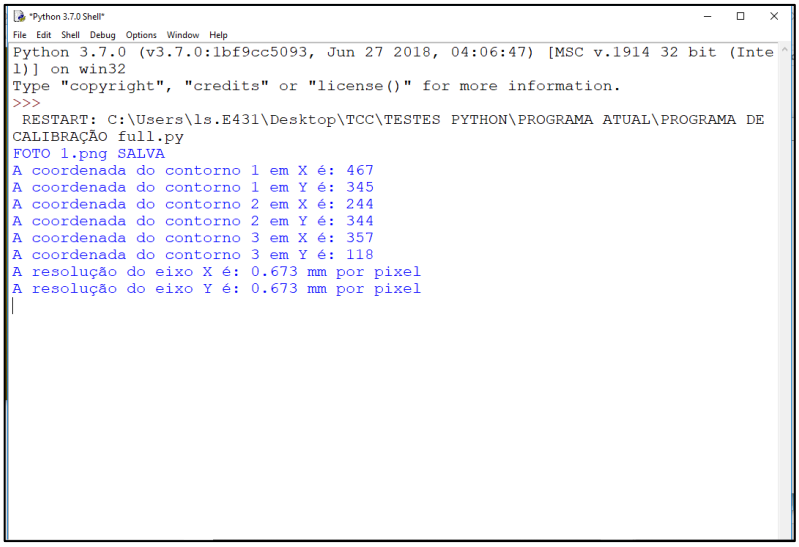
Com os valores das coordenadas em pixel e as distâncias reais entre os centros dos círculos conhecidos, através da regra de três se obteve as equações 3.1 e 3.2 que determinam as resoluções, valor de cada pixel em milímetros, Rx e Ry, dos eixos X e Y respectivamente do plano.

$$R_x = \frac{150}{|D_{x1} - D_{x2}|} \qquad [3.1]$$

$$R_y = \frac{150}{\left\{\frac{(|D_{y1} - D_{y2}|)}{2}\right\} - D_{y3}} \qquad [3.2]$$

A Figura 3.7 ilustra a determinação dos contornos e da resolução dos eixos X e Y.

Figura 3.7 - Padrão de calibração da câmera



Fonte: Autoria própria, 2019

Neste momento o programa assume os valores D_{x1} e D_{y1} como origem do plano cartesiano. O círculo que determina tal coordenada possui um ponto branco no centro que serve de referência para apoiar o dispositivo cônico e programar o ponto em que o robô originalmente busca pela peça. A Figura 3.8 a seguir apresenta o dispositivo:

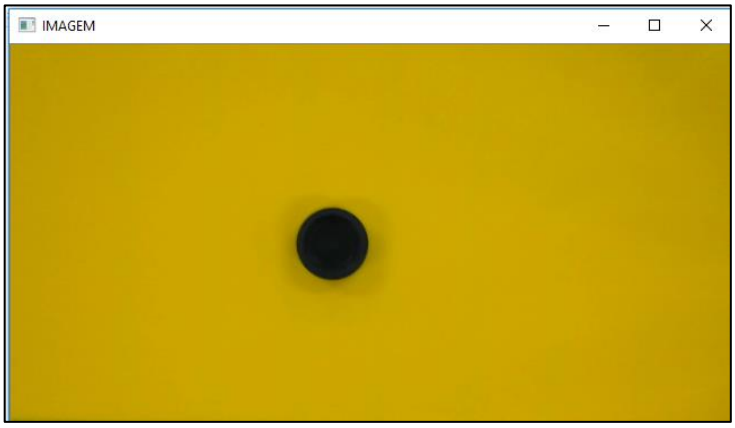
Figura 3.8 - Dispositivo cônico para programação do robô

Fonte: Autoria própria, 2019

Após a determinação da resolução, a tecla *ESC* é pressionada duas vezes para retornar ao menu inicial.

Selecionando a opção 3 do menu - DETERMINAR POSIÇÃO DA PEÇA, o programa reinicia a conexão com a câmera e abre uma janela com o título de IMAGEM, que exibe a área de manipulação da peça. A peça a ser manipulada é posicionada na área delimitada e ao pressionar a tecla barra de espaço, uma nova foto é salva com a peça sobre a área de manipulação, conforme ilustra a Figura 3.9 a seguir:

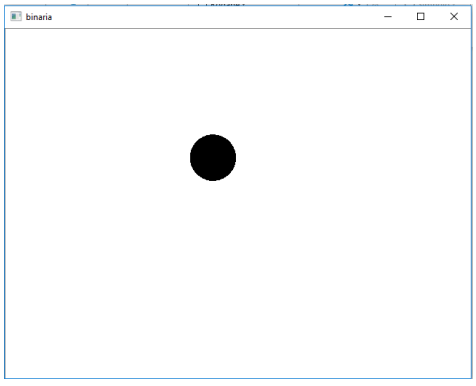
Figura 3.9 - Captura da foto



Fonte: Autoria própria, 2019

Esta nova imagem é carregada em uma variável do tipo matriz e subtraída de outra matriz já carregada com a imagem padrão da base vazia, gerando uma nova, que ressalta a diferença entre elas, no caso, a peça. Esta variável gerada é convertida para escala de cinza e posteriormente passa pelo processo de binarização, onde a representação da peça na imagem assume a cor preta e a base acrílica a cor branca, conforme ilustrado pela Figura 3.10 a seguir:

Figura 3.10 - Imagem da peça binarizada



Fonte: Autoria própria, 2019

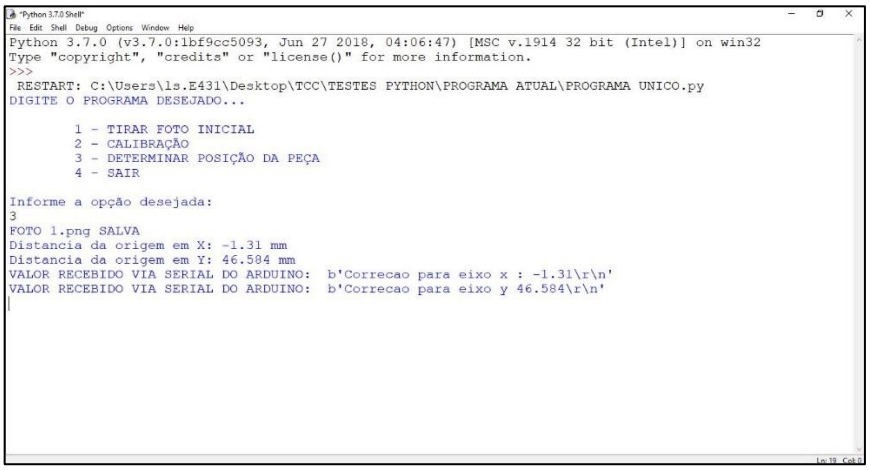
Em seguida, determina-se o contorno da peça e a partir deste, as coordenadas em pixel do centro de massa da peça. As equações 3.3 e 3.4 mostram como são determinadas a distância cartesiana da posição da peça em relação a coordenada de referência inserida no programa como a coordenada programada para o robô pegar a peça.

$$Pos_x = (C_{x1} - C_{refx}) * R_x \quad [3.3]$$

$$Pos_y = (C_{y1} - C_{refy}) * R_y \quad [3.4]$$

A Figuras 3.11 a seguir ilustram a captura da foto da peça, a imagem binarizada e a apresentação das distâncias em X e Y respectivamente.

Figura 3.11 - Apresentação das coordenadas



Fonte: Autoria própria, 2019

Em seguida os valores das coordenadas da peça Px e Py são enviados via comunicação serial à interface de comunicação com o robô.

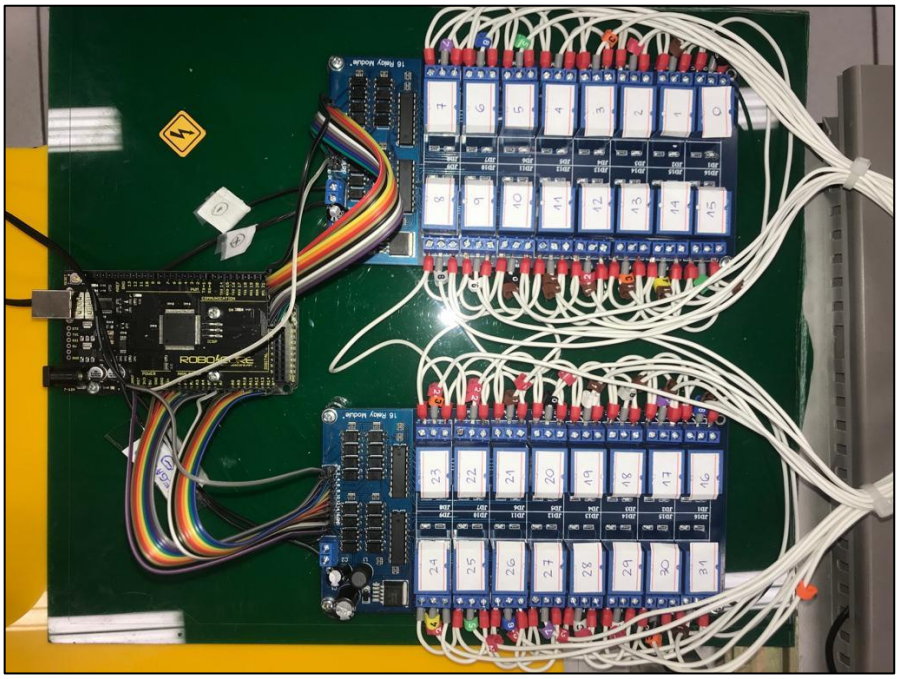
Pressionando a tecla ESC duas vezes, se retorna ao menu inicial e selecionando a opção 4 o programa é fechado. O programa completo encontra-se no Apêndice A.

3.3 Interface de comunicação entre o sistema de visão e o robô

Após o processamento da imagem através da programação do sistema de visão, parte-se para a elaboração de uma interface responsável pela comunicação entre o sistema de visão e o robô. Esta interface é necessária, pois o sistema de visão tem um canal de comunicação serial e o robô possui um canal de comunicação paralelo.

A interface de comunicação constitui-se de uma placa Arduino MEGA 2560 e duas placas Módulo Relé 16 canais, fixadas sobre uma placa de acrílico para facilitar sua visualização, organização, montagem e manutenção, conforme Figura 3.12.

Figura 3.12 - Interface de comunicação entre o sistema de visão e o robô



Fonte: Autoria própria, 2019

Para que o robô tenha seus valores de posição de x e y atualizados, é necessário transmitir, via comunicação paralela, informações como o comando a ser realizado e os dados adicionais de acordo com o comando escolhido. Essa informação é transmitida baseada na identificação dos sinais contida na Tabela 3.1.

Tabela 3.1 - Identificação dos sinais da interface de comunicação

INFORMAÇÃO ROBÔ	BORNE	RELÉ	PIN	PORT
DATA AREA 2 BIT0	19	6	29	A 7
DATA AREA 2 BIT1	20	7	28	A 6
DATA AREA 2 BIT2	21	4	27	A 5
DATA AREA 2 BIT3	22	5	26	A 4
DATA AREA 2 BIT4	23	2	25	A 3
DATA AREA 2 BIT5	24	3	24	A 2
DATA AREA 2 BIT6	25	0	23	A 1
DATA AREA 2 BIT7	26	1	22	A 0
DATA AREA 2 BIT8	27	9	30	C 7
DATA AREA 2 BIT9	28	8	31	C 6
DATA AREA 2 BIT10	29	11	32	C 5
DATA AREA 2 BIT11	30	10	33	C 4
DATA AREA 2 BIT12	31	13	34	C 3
DATA AREA 2 BIT13	32	12	35	C 2
DATA AREA 2 BIT14	33	15	36	C 1
DATA AREA 2 BIT15	34	14	37	C 0
DATA AREA 1 BIT0	11	17	A15	K 7
DATA AREA 1 BIT1	12	19	A14	K 6
DATA AREA 1 BIT2	13	21	A13	K 5
DATA AREA 1 BIT3	14	23	A12	K 4
DATA AREA 1 BIT4	15	25	A11	K 3
DATA AREA 1 BIT5	16	27	A10	K 2
DATA AREA 1 BIT6	17	29	A9	K 1
DATA AREA 1 BIT7	18	31	A8	K 0
PARIDADE	10	16	A7	F 7
STROBE	8	18	A6	F 6
VAZIO	F5	20	A5	F 5
VAZIO	F4	22	A4	F 4
COMMAND AREA 0 BIT0	35	24	A3	F 3
COMMAND AREA 0 BIT1	36	26	A2	F 2
COMMAND AREA 0 BIT2	37	28	A1	F 1
COMMAND AREA 0 BIT3	38	30	A0	F 0

Fonte: Autoria própria, 2019

Com base no Manual do Manipulador UPJ (2002) do robô Motoman, para esta aplicação é utilizado apenas um comando, escrever variável, por isso o valor atribuído para a escolha do comando, de 4 bits é fixo em 0101. Para determinar qual variável é escrita (*Data area 1*) têm-se 8 bits, que definem qual a coordenada corrigida. Para determinar o valor de correção escrito (*Data area 2*) utilizam-se 16 bits, que é o valor de correção da coordenada, obtido através da informação transmitida pelo sistema de visão.

Além destes, dois outros sinais são necessários, o sinal de *strobe*, que informa ao controlador do robô que os valores estão prontos para serem lidos e o bit de paridade, utilizado para detectar inconsistências na transmissão de dados. Para este sinal, são somados todos os bits 1 da informação. Se o resultado dessa soma for par (*even parity*) o bit de paridade tem o valor 1, se for ímpar (*odd parity*) este assume o valor 0.

O sistema de visão é conectado ao Arduino através de uma porta USB, estabelecendo entre ambos uma comunicação do tipo serial. Este canal é do tipo *half-duplex*, ou seja, os dois dispositivos podem enviar e receber dados, porém não simultaneamente.

A programação da interface é feita de forma que o Arduino inicia a comunicação serial e aguarda o recebimento da primeira *string* que é armazenada na variável x. Após uma pausa de 1 segundo, a interface de comunicação aguarda uma nova *string* que é armazenada na variável y.

Para que os valores recebidos pelo Arduino possam ser melhores trabalhados pela interface de comunicação a função *toInt()* é aplicada às variáveis x e y, transformando as variáveis do tipo *string* em variáveis do tipo *integer*. A partir disso as mesmas são identificadas como xInt e yInt.

Com os valores de correção das variáveis x e y armazenados, inicia-se a programação do Arduino para que esses valores, incluindo os dados que representam a variável e os dados do tipo do comando sejam transmitidos ao robô.

O robô Motoman possui um conector denominado CN8, que possui 50 pinos que representam os 50 bits simultâneos que o mesmo está apto a receber, por isso seu canal de comunicação é considerado paralelo.

Devido ao canal de comunicação do robô ser do tipo paralelo, todos os dados devem ser transmitidos pela interface utilizando uma saída digital para cada bit de informação. Baseado nisso desenvolve-se uma rotina no programa da placa Arduino. O programa completo encontra-se no Apêndice B.

Nesta etapa do programa a interface transmite todos os dados necessários para a correção do valor da posição do robô em x e após um intervalo de tempo determinado, há a transmissão dos dados necessários para a correção do valor da posição do robô em y. Essa rotina repete-se toda vez que há alguma alteração nos valores de correção de coordenadas enviados pelo sistema de visão.

As saídas digitais da interface são ligadas quando seu estado é igual a 1 e desligadas quando igual a 0. Quando ligadas, transmitem um sinal de 5 V, o que é insuficiente para acionar uma entrada do conector CN8 do robô Motoman, que necessita de 24 V. Por isso, faz-se necessário a utilização de relés, que são acionados com os 5 V da saída do Arduino, porém comutam os 24 V de uma fonte externa para a entrada do CN8.

Devido a quantidade de entradas que devem ser acionadas é necessário a utilização de no mínimo 30 relés. Optou-se então pela utilização de duas placas Módulo Relé com 16 canais cada, tendo no total 32 relés disponíveis.

Cada saída da interface de comunicação representa um bit da informação a ser transmitida. Essa saída aciona um relé que faz com que os dados cheguem ao robô em forma de tensão: 24 V para o bit igual a 0 e 0 V para o bit igual a 1, pois o controlador trabalha no sistema NPN.

3.4 Testes de funcionamento e integração das partes

Inicialmente o sistema de visão tem seu programa divido por suas funcionalidades, sendo elas a calibração e a determinação da posição da peça. Cada uma delas são testadas individualmente, facilitando o diagnóstico de falhas do sistema, e após obter os resultados esperados, o programa é unificado.

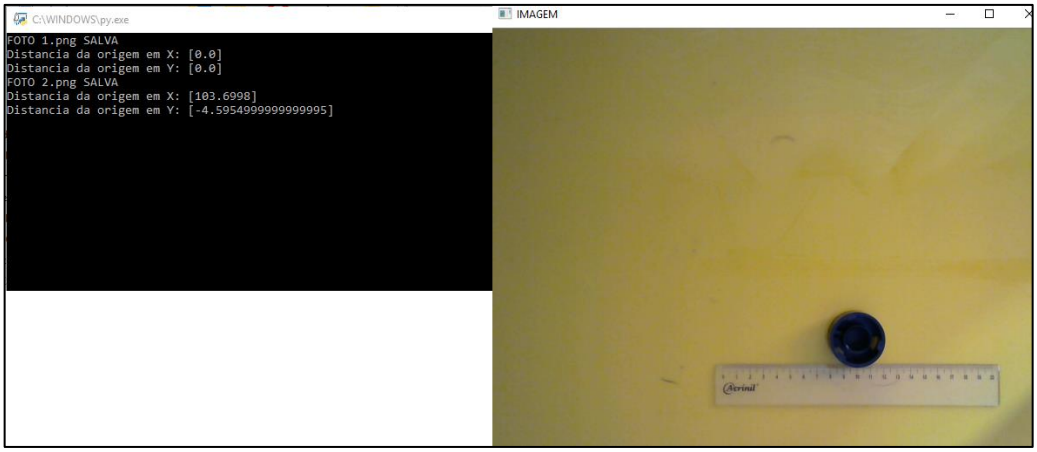
No processo de calibração a resolução obtida é de aproximadamente 0,6 mm por pixel, que permite a operação dentro da área inicialmente planejada de 300 mm x 300 mm. Para obter melhor resolução, é necessário diminuir a altura do posicionamento da câmera, o que limita a movimentação do robô, aumentando a possibilidade de colisões do mesmo com a estrutura.

O ponto branco criado no centro do círculo que determina da origem do plano cartesiano para a programação do ponto de pega da peça pelo robô, tem o diâmetro de 1mm determinado para que não influenciasse o processo de calibração, uma vez que em testes realizados com 2 mm de diâmetro o software o identifica como um novo contorno e desta forma o cálculo da resolução considera dois pontos com a mesma coordenada, apresentando resultados incoerentes.

Para a determinação da posição da peça inicialmente utiliza-se como referência um ponto qualquer dentro do campo de visão da câmera com coordenadas baseadas no plano cartesiano. Uma peça é colocada neste campo e é movida para outro ponto. Seu deslocamento, apresentado pelo software, mostrou-se próximo do real (medido com o auxílio de uma escala) em curtas distâncias, considerando o fato de que o controle com a escala e a movimentação não são precisas. Para deslocamentos maiores, na casa de 100 mm, o resultado apresentado pelo software apresenta diferenças significativas em relação ao real. Desta forma, torna-se necessário aprimorar a resolução do sistema.

A Figura 3.13 ilustra os testes iniciais de determinação da posição da peça:

Figura 3.13 - Testes inicias de determinação da posição da peça



Fonte: Autoria própria, 2019

O envio das informações obtidas com o sistema de visão para a interface de comunicação é feito de forma serial para um Arduino. Ao receber os valores das coordenadas atualizadas da peça, o Arduino armazena cada valor em uma variável e para controle, envia uma mensagem ao sistema de visão informando o valor recebido e a variável em que o mesmo é armazenado.

O sistema de visão apresenta os valores recebidos na tela de sua interface para assegurar que os dados são processado de maneira correta. Tempos de esperas (*delays*) foram padronizados para que não haja conflitos ou colisões no acesso ao meio de transmissão entre o sistema de visão e interface.

Para os testes da interface de comunicação, a conexão dos sinais de dados da interface de comunicação com o controlador do robô é feita com os cabos do terminal comum de cada relé conectados aos respectivos ,disponíveis na parte inferior do *trolley* do robô, que correspondem aos terminais do conector CN8. Os contatos normais fechados dos relés foram conectados aos 24 V da fonte do robô e os contatos normais abertos ao 0 V.

Uma vez conectada à borneira do robô, os testes iniciam-se forçando cada um dos sinais pela interface de comunicação, que são visualizados pela tabela de controle de variáveis, disponível no programa WINCAPS do robô. Os testes confirmam que o controlador é NPN, adotando o estado lógico HIGH quando recebe 0 V.

Durante os teste nota-se que os sinais do *Data area 1* e *Data area 2*, os sinais de comando, bits de *strobe* e paridade estão deslocados em relação ao bit mais significativo e o bit menos significativo. As alterações para resolução deste deslocamento são feitas na programação da própria interface de comunicação.

Para a verificação do funcionamento do sistema de visão, são realizados testes inserindo manualmente os valores atualizados das coordenadas da peça nas variáveis do robô pelo *pendant box,* assegurando que o robô movimenta-se de acordo com a nova posição da peça.

3.5 Dificuldades encontradas e soluções

Diversas dificuldades são encontradas ao longo do projeto, porém grande parte destas são solucionadas e proporcionam um grande aprendizado aos integrantes do grupo. Tratando-se de um projeto complexo, que faz uso de uma linguagem de programação não apresentada durante o curso, a trajetória do projeto mostra-se desafiadora e gratificante.

Uma das primeiras dificuldades encontradas na aplicação do sistema de visão mostra-se na manipulação de imagens capturadas e armazenadas no *Hard Disk*, que apresentam resolução de 640 pixels de largura e 480 pixels de altura. Apesar das tentativas para aumentar a resolução da imagem constata-se que essa é a resolução máxima obtida pelo modelo de webcam utilizada, o que delimita o processo. Porém, durante o trabalho a resolução da imagem mostra-se muito suficiente para a aplicação.

Um desafio do trabalho é a conversão em pixels para milímetros, cuja solução é a aplicação de uma regra matemática. Com a criação do padrão de calibração que fornece distâncias conhecidas e a aplicação da regra de três, pode-se obter as equações dos cálculos de distância com resultados satisfatórios.

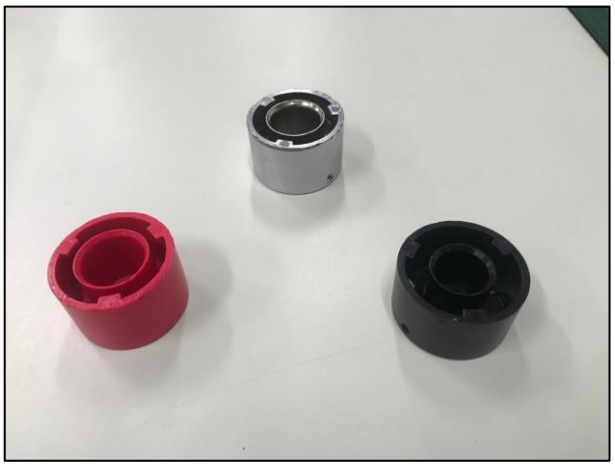
Apesar de o processo de subtração mostrar-se eficiente quanto à determinação de diferenças no campo visão da câmera, nota-se a influência da iluminação e,

consequentemente, do aparecimento de sombras na captação da imagem, corroborando a pouca flexibilização dos sistemas de visão conforme Marques Filho e Vieira Neto (1999) no capítulo 1.4 deste trabalho. Em ambientes com uma única fonte de iluminação é comum o aparecimento de sombras que não são eliminadas no processo de subtração e influenciam diretamente na determinação do contorno da peça.

Com soluções limitadas, adota-se que o processo de captura da imagem inicial deve ser repetido a cada alteração significativa do ambiente, que resultem no surgimento de reflexos ou sombras. Estes, desde que apareçam na imagem inicial e na calibração ou determinação da peça, não interferem no resultado final, sendo eliminados pelo processo de subtração.

Os testes de manipulação de peças são feitos com as peças didáticas utilizadas na aula de Robótica da FATEC São Bernardo do Campo, ilustradas na Figura 3.14 a seguir:

Figura 3.14 - Peças didáticas manipuladas pelo robô



Fonte: Autoria própria, 2019

Dado o formato cilíndrico da peça, nota-se que quando esta se encontra nas extremidades da área de captura determinada, a câmera reconhece parte da sua vista lateral e não apenas a vista superior como o esperado. Desta forma, ao invés de reconhecer o contorno e o centro de massa de um círculo, o programa reconhece uma

elipse, determinando o centro desta forma geométrica, porém os valores são incoerentes com os esperados.

Com relação a cor, o problema ocorre nas peças prateadas, que são claras e tem parte de sua forma apagada no processo de binarização. Testes realizados utilizando a base de cores mais escuras como vermelho e verde, apresentam o mesmo problema, com o agravante que com o fundo escuro, a influência de reflexos é muito maior no sistema de visão.

Com relação a interface de comunicação, há dificuldade para que o controlador do robô processe e assuma os valores recebidos. Seguindo as instruções do manual do mesmo, são inseridos os dados que informam o comando que é executado, a identificação da variável a ser escrita, o valor que deve ser escrito e a paridade da mensagem. Em seguida aciona-se o sinal do *strobe,* porém, o sinal de saída do robô de processamento de comando completo não é apresentado pelo controlador.

Porém, com o valor das variáveis inseridos na tabela pelo *pendant box* o robô localiza a peça na área de manipulação com uma precisão dentro do esperado.

Outra dificuldade encontrada é a falha de precisão causada pela não coincidência do plano cartesiano do robô com o plano do sistema de visão. A diferença angular apresentada entre os sistemas de coordenadas quando o padrão de calibração ou a câmera não estão alinhados com os eixos do robô causam erros de posicionamento significativos.

A solução aplicada é o alinhamento do padrão de calibração utilizando o robô para tal, posicionando o dispositivo cônico no centro de um dos círculos e movendo a garra 150 mm em X, ajustando o padrão para que o dispositivo fique no centro do segundo círculo. Em seguida, durante a calibração, é ajustado o ângulo do suporte da câmera até que sejam observados os mesmos valores de coordenada em Y para os dois círculos da base do triângulo.

Para maior aperfeiçoamento do projeto são aplicados os conceitos de mudança de coordenadas de geometria analítica para efetuar a correção angular do plano do sistema de visão, aplicando uma rotação matricial. As equações [3.3] e [3.4] apresentam o cálculo dos valores das coordenadas com correção angular, corrigido pelo ângulo α no momento da calibração:

$$posx = \left((C_{x1} - C_{refx}) * \cos\alpha + (C_{y1} - C_{refy}) * sen\ \alpha\right) * R_x \quad [3.3]$$

$$posy = \left((C_{x1} - C_{refx}) * \text{sen}\,\alpha + (C_{y1} - C_{refy}) * cos\ \alpha\right) * R_y \quad [3.4]$$

CONSIDERAÇÕES FINAIS

O objetivo do projeto é desenvolver um sistema automatizado para flexibilizar o processo de manipulação de peças que ocorrem em células robotizadas que não possuem tal sistema de fábrica. Justifica-se por permitir a interação com processos menos precisos de posicionamento, proporcionando redução de tempo de produção e de custos para fabricação de dispositivos para posicionamento de peças.

Para a construção e desenvolvimento fez-se o uso de estruturas de fixação em alumínio, webcam, um computador para programação e processamento de dados, uma placa Arduino Mega, placas de saída a relé, IDE de programação Arduino e Python.

As teorias pesquisadas para utilização no projeto foram fundamentais e proporcionou aos integrantes o conhecimento para a aplicação de sistemas de visão em células de manufatura, especialmente sobre robótica, processamento de imagens, meios de comunicação e sistemas de coordenadas. Dentre as fontes pesquisadas, os conteúdos que mais agregaram conhecimento foram os relativos ao processamento de imagens, comunicação serial e sistemas de coordenadas.

Os métodos e técnicas utilizados na metodologia científica foram de grande importância na organização e planejamento das etapas do projeto e ao objetivo proposto.

Destacam-se como vantagens do projeto a utilização em diversos setores industriais, sendo necessária somente uma adaptação para o fim ao qual este será utilizado; baixo custo de instalação; ausência de necessidade de manutenção e uma interface simples com o usuário.

Com uma linguagem de programação e conceitos de processamento de imagens não apresentados no decorrer do curso, o projeto proporcionou aos integrantes o conhecimento em mais uma das vertentes da automação industrial.

Como desvantagem, destaca-se a grande dependência da iluminação, inerente ao processamento de imagens, uma vez que alterações na iluminação do ambiente influenciam significativamente na detecção de objetos.

Como ponto forte destaca-se a flexibilidade no sistema de manipulação de peças a um custo relativamente baixo, levando em conta que dispositivos oferecidos para tais aplicações hoje no mercado possuem um alto valor, tornando sua utilização inviável, especialmente para empresas de pequeno e médio porte.

Durante o desenvolvimento algumas dificuldades foram encontradas como: comunicação direta com o robô e desenvolvimento da programação tanto da interface de comunicação quanto do sistema de visão. Tais dificuldades foram solucionadas através de consultas ao orientador, de pesquisas e conhecimentos breves dos integrantes do grupo.

Como sugestões de melhorias futuras, destaca-se a criação de uma interface de operação com um layout mais intuitivo ao usuário, a utilização de uma câmera com maior resolução, permitindo maior precisão de posicionamento. Ainda, para aprimorar o sistema, um algoritmo para detecção de objetos estranhos ao processo, que poderia ser implementado para parar o robô e evitar colisões e danos ao operador.

REFERÊNCIAS

BARELLI, F. **Introdução à visão computacional**. 1. ed. São Paulo: Casa do Código, 2018.

BECKHOFF. **PLC e controle de robô perfeitamente integrados**, 2014. Disponivel em: <www.download.beckhoff.com/2014/pr222014_Beckhoff_br.pdf>. Acesso em: 19 out. 2018.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**. 1. ed. Sebastopol: O´Reilly, 2008.

CORREIO DO POVO. **Robôs industriais tem recorde de vendas em 2017**, 2018. Disponivel em: <www.correiodopovo.com.br/Noticias/Economia/2018/6/654318/Robos-industriais-tem-recorde-de-vendas-em-2017>. Acesso em: 24 ago. 2018.

FREITAS, C. A.; SANTOS, P. B. **Robótica**. São Paulo: SENAI, 2000.

GONZALEZ, R. C.; WOODS, R. C. **Processamento digital de imagens**. 3. ed. São Paulo: Pearson Prentice Hall, 2010.

GUDWIN, R. R. **Linguagens de programação**. Faculdade de Engenharia Elétrica e de Computação. Campinas: UNICAMP. 1997.

INSTITUTO DE ENGENHEIROS ELETRICISTAS E ELETRÔNICOS - IEEE. IEEE Spectrum. **The 2018 top programming languages**, 2018. Disponivel em: <www.spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>. Acesso em: 20 set. 2018.

JÄHNE, B.; HAUßECKER, H.; GEIßLER, P. **Handbook of computer vision applications**. San Diego: Academic Press, v. 1 - Sensors and Imaging, 1999.

LABAKI, J. **Introdução a Python - Módulo A**. Ilha Solteira: UNESP, 2017.

MANDORSSON, A. Como conectar células de robôs a qualquer rede industrial. **Engenharia Brasil**, 2013. Disponivel em: <http://www.engenharia-brasil.com/2823-como-conectar-celulas-de-robos-a-qualquer-rede-industrial-.htm>. Acesso em: 18 out. 2018.

MANUAL DE NORMALIZAÇÃO DE PROJETOS DE TRABALHO DE GRADUAÇÃO FATEC SÃO BERNARDO DO CAMPO. **Material didático para utilização nos projetos de trabalho de graduação dos cursos de tecnologia em automação industrial e informática**. São Bernardo do Campo:Fatec, 2017.

MANUAL DO MANIPULADOR UPJ. **Manual do manipulador UPJ com controlador JRC Motoman**. Motoman Incorporated. Ohio. 2002.

MANZANO, J. A.; OLIVEIRA, J. F. **Algoritmos - lógica de desenvolvimento de programação de computadores**. 22. ed. São Paulo: Érica, 2009.

MARENGONI, M.; STRINGHINI,. Tutorial: Introdução à visão computacional usando OpenCV. **Revista de Informática teórica e aplicada**, Porto Alegre, v. 16, n. 1, p. 125-160, Julho 2009.

MARQUES FILHO, O.; VIEIRA NETO, H. **Processamento digital de imagens**. Rio de Janeiro: BRASPORT, 1999.

MENEZES, N. N. C**.. Introdução à programação com Python – algoritmos e lógica de programação para iniciantes**. 2. ed. São Paulo: Novatec, 2014.

MONTAQIM, A. Robotics & Automation News. **Top 14 industrial robot companies and how many robots they have around the world**, 2015. Acesso em: 24 ago. 2018.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do trabalho científico:** Métodos e técnicas da pesquisa e do trabalho científico. 2. ed. Rio Grande do Sul: Universidade Feevale, 2013.

ROBOTIC INDUSTRIES ASSOCIATION - RIA. **Unimate//The first industrial robot**, 2015. Disponivel em: <www.robotics.org/joseph-engelberger/unimate.cfm>. Acesso em: 24 ago. 2018.

ROMANO, V. F. **Robótica Industrial - aplicação na indústria de manufatura e processos**. 1. ed. São Paulo: Blucher , 2002.

SANTOS, V. M. F. Robótica Industrial. **ece.ufrgs**, 2003. Disponivel em: <www.ece.ufrgs.br/~rventura/RoboticaIndustrial.pdf>. Acesso em: 30 ago. 2018.

SCHMITT, Á. et al. Implementação de uma rede industrial para células de soldagem robotizadas utilizando o protocolo modbus. **Revista Ilha Digital**, Florianópolis, v. 4, p. 61-66, 2013.

SEBESTA, R. W. **Conceitos de linguagens de programação**. 11. ed. São Paulo: Bookman, 2018.

SEVERINO, A. J. **Metodologia do trabalho científico**. 24. ed. São Paulo: Cortez, 2016.

SILVEIRA, G. Avanços recentes em robótica guiada por visão. **A voz da indústria**, 2017. Disponivel em: <www.avozdaindustria.com.br/avancos-recentes-em-robotica-guiada-por-visao/>. Acesso em: 03 out. 2018.

STEGER, C.; MARKUS, U. **Machine vision algorithms and apllications**. 1. ed. Weinheim: Wiley-vch, 2007.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. D. **Sensores Indústriais- fundamentos e aplicações**. 3. ed. São Paulo: Érica, 2007.

UNIVESP - UNIVERSIDADE VIRTUAL DO ESTADO DE SÃO PAULO. **Geometria Analítica - Aula 12 - Mudança de Coordenadas**. 2016. Disponível em: < https://www.youtube.com/watch?v=uQPMW44RDpk>. Acesso em: 13 mai. 2019.

WORLD ROBOTICS. IFR - International federation of robots. **Executive Summary World Robotics 2018 Industrial Robots**, 2018. Disponivel em: <www.ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf>. Acesso em: 24 ago. 2018.

APÊNDICE A – PROGRAMA DO SISTEMA DE VISÃO EM PYTHON

```
import cv2
import numpy as np
import serial
import os
import time
opcao = 0
k = cv2.waitKey(1)

while opcao != 4:


    os.system('cls') or None
    print("""DIGITE O PROGRAMA DESEJADO...

        1 - TIRAR FOTO INICIAL
        2 - CALIBRAÇÃO
        3 - DETERMINAR POSIÇÃO DA PEÇA
        4 - SAIR
""")

    opcao = int(input("Informe a opção desejada: \n"))
    if opcao == 1:
        cam = cv2.VideoCapture(0 + cv2.CAP_DSHOW)
        cv2.namedWindow("FOTO BASE")
        img_counter = 0
        print("Pressione Barra de espaço para tirar a foto")
        while True:
            ret, frame = cam.read()
            cv2.imshow("FOTO BASE", frame)

            if not ret:
                break
            k = cv2.waitKey(1)

            if k%256 == 27:
                # ESC pressed
                print("Pressione ESC para retornar ao Menu Inicial")
                break
            elif k%256 == 32:
                # SPACE pressed
                imagem = "FOTO 0.png".format(img_counter)
                cv2.imwrite(imagem, frame)
                print("FOTO DE REFERENCIA ARMAZENADA, PRESSIONE ESC PARA
SAIR!".format(imagem))
        cv2.waitKey(0)
        cam.release()
```

```
        cv2.destroyAllWindows()


    elif opcao == 2:
        cam = cv2.VideoCapture(0 + cv2.CAP_DSHOW)
        cv2.namedWindow("CALIBRACAO")
        img_counter = 1
        FOTO0 = cv2.imread('FOTO 0.png')
        print("Posione o padrão de calibração e pressione Barra de espaço")
        while True:
            ret, frame = cam.read()
            cv2.imshow("CALIBRACAO", frame)
            if not ret:
                break
            k = cv2.waitKey(1)
            if k%256 == 32:
                # SPACE pressed
                img_name = "FOTO {}.png".format(img_counter)
                cv2.imwrite(img_name, frame)
                print("{} SALVA".format(img_name))
                img_counter += 1
                FOTO1 = cv2.imread(img_name)
                imgsub = cv2.subtract(FOTO0, FOTO1)
                cv2.imshow("SUBTRACAO", imgsub)
                gray = cv2.cvtColor(imgsub,cv2.COLOR_BGR2GRAY)
                ret,bin1 = cv2.threshold(gray,80,255,cv2.THRESH_BINARY_INV)
                cv2.imshow("binaria", bin1)
                _,contours,hierarchy = cv2.findContours(bin1,cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
                if len(contours) > 0:
                    cnt = contours[1]
                    M = cv2.moments(cnt)
                    # Processing here.
                    cx1 = int(M['m10']/M['m00'])
                    cy1 = int(M['m01']/M['m00'])
                    print( "A coordenada do contorno 1 em X é:",cx1 )
                    print( "A coordenada do contorno 1 em Y é:",cy1 )
                    cnt = contours[2]
                    M = cv2.moments(cnt)
                    # Processing here.
                    cx2 = int(M['m10']/M['m00'])
                    cy2 = int(M['m01']/M['m00'])
                    print( "A coordenada do contorno 2 em X é:",cx2 )
                    print( "A coordenada do contorno 2 em Y é:",cy2 )
                    cnt = contours[3]
                    M = cv2.moments(cnt)
                    # Processing here.
                    cx3 = int(M['m10']/M['m00'])
                    cy3 = int(M['m01']/M['m00'])
```

```
                print( "A coordenada do contorno 3 em X é:",cx3 )
                print( "A coordenada do contorno 3 em Y é:",cy3 )
                resx = float(150/abs(cx1-cx2))
                resy = float(150/(abs((cy1+cy2)/2)-cy3))
                print( "A resolução do eixo X é:", round(resx, ndigits=3) , "mm por pixel" )
                print( "A resolução do eixo Y é:", round(resy, ndigits=3) , "mm por pixel")
                print("Pressione ESC para sair")

            else:
                print ("Sorry No contour Found.")

        elif k%256 == 27:
            # ESC pressed
            print("PRESSIONE ESC PARA RETORNAR AO MENU INICIAL...")
            break
    cv2.waitKey(0)
    cam.release()
    cv2.destroyAllWindows()

elif opcao == 3:
    cam = cv2.VideoCapture(0 + cv2.CAP_DSHOW)
    cv2.namedWindow("IMAGEM")
    img_counter = 1
    FOTO0 = cv2.imread('FOTO 0.png')
    porta = 'COM5'
    velocidade = 9600
    conexao = serial.Serial(porta,velocidade)
    while True:
        ret, frame = cam.read()
        cv2.imshow("IMAGEM", frame)
        if not ret:
            break
        k = cv2.waitKey(1)

        if k%256 == 32:
            os.system('cls') or None
            # SPACE pressed
            img_name = "FOTO {}.png".format(img_counter)
            cv2.imwrite(img_name, frame)
            print("{} SALVA".format(img_name))
            img_counter += 1
            FOTO1 = cv2.imread(img_name)
            imgsub = cv2.subtract(FOTO0, FOTO1)
            cv2.imshow("SUBTRACAO", imgsub)
            gray = cv2.cvtColor(imgsub,cv2.COLOR_BGR2GRAY)
            ret,bin1 = cv2.threshold(gray,130,255,cv2.THRESH_BINARY_INV)
            cv2.imshow("binaria", bin1)
            _,contours,hierarchy = cv2.findContours(bin1,cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

```
        if len(contours) > 0:
            cnt1 = contours[1]
            M = cv2.moments(cnt1)
            # Processing here.
            if cx1 > cx2:
                ox = cx2
                oy = cy2
            else:
                ox = cx1
                oy = cy1

            cpx1 = int(M['m10']/M['m00'])
            cpy1 = int(M['m01']/M['m00'])
            posx = float((cpx1-ox)*resx*(-1))
            posy = float((cpy1-oy)*resy)
            Coordx= str(10*round(posx, ndigits=1))
            Coordy= str(10*round(posy, ndigits=1))
            print( "Distancia da origem em X:",Coordx ,"mm" )
            print( "Distancia da origem em Y:",Coordy ,"mm" )
            conexao.write(Coordx.encode());
            leitura_serialx = conexao.readline()
            print ("VALOR RECEBIDO VIA SERIAL DO ARDUINO: ",leitura_serialx)

            conexao.write(Coordy.encode());
            leitura_serialy = conexao.readline()

            print ("VALOR RECEBIDO VIA SERIAL DO ARDUINO: ",leitura_serialy)
        else:

            print ("Sorry No contour Found.")

    elif k%256 == 27:
        # ESC pressed
        print("PRESSIONE ESC PARA RETORNAR AO MENU INICIAL...")

        break
cv2.waitKey(0)

cam.release()
```

APÊNDICE B – PROGRAMA DA INTERFACE DE COMUNICAÇÃO

```
const int triggerType = LOW;// your relay type
//int loopDelay = 1000;// delay in loop
String x;
String y;
int xInt;
int yInt;
unsigned int XTESTE1, XTESTE2;
char byteRead;
String StringRead;
int intRead;
int mensagem;
int Valor[16];

void setup() {
  //declara os registradores como saída (saída=1, entrada=0);
  DDRF=B11111111; DDRK=B11111111; DDRA=B11111111; DDRC=B11111111;

  //reseta (desliga) todas as saídas na inicialização (desligada=0; ligada=1);
  PORTF=B11111111; //PORTF 0 A 3  4 MENOS SIGNIFICATIVOS 7 PARIDADE 6
STROBE
  PORTK=B11111111; //PORTK 0 A 7 DATA AREA 1
  PORTA=B11111111; //PORTA MENOS SIGNIFICATIVO 0 A 7 DATA AREA 2
  PORTC=B11111111; //PORTC MAIS SIGNIFICATIUVO 8 A 15 DATA AREA 2

  Serial.begin(9600);
}

void loop() {
char byteRead;
String StringReadx;
int intReadx;
```

```
String StringReady;
int intReady;
   while(Serial.available() > 0) {

   StringReadx = Serial.readString();         // recebe identificação do eixo
   delay(1000); // delay necessário
   Serial.print("Correcao para eixo x : ");// envio de texto para Python para teste
   Serial.println(StringReadx); // envio do valor armazenado para controle
   delay(1000); // delay necessário
   StringReady = Serial.readString();// Armazena o valor em x
   Serial.print("Correcao para eixo y ");// envio de texto para Python para teste
   Serial.println(StringReady); // envio do valor armazenado para controle

   mensagem = 1; //Flag de valor novo
   }

  if(mensagem == 1) {
   mensagem = 0;
      for(int i=0; i<16; i++) {Valor[i]=0;}
   intReadx = StringReadx.toInt();
   //Serial.println(intRead);
   if(intReadx < 0){Valor[15] = 1; intReadx = abs(intReadx)-1;}
   else {Valor[15] = 0;}

   intReadx = abs(intReadx);
   for(int i=0; intReadx>1; i++) {
     Valor[i] = intReadx % 2;
     intReadx = intReadx / 2;
     if(intReadx < 2){Valor[i+1] = 1;}
   }

   if (Valor[15] == 0){
   if (Valor[0] == 0) {digitalWrite(29, HIGH);} else {digitalWrite(29, LOW);}
```

```
if (Valor[1] == 0) {digitalWrite(28, HIGH);} else {digitalWrite(28, LOW);}
if (Valor[2] == 0) {digitalWrite(27, HIGH);} else {digitalWrite(27, LOW);}
if (Valor[3] == 0) {digitalWrite(26, HIGH);} else {digitalWrite(26, LOW);}

if (Valor[4] == 0) {digitalWrite(25, HIGH);} else {digitalWrite(25, LOW);}
if (Valor[5] == 0) {digitalWrite(24, HIGH);} else {digitalWrite(24, LOW);}
if (Valor[6] == 0) {digitalWrite(23, HIGH);} else {digitalWrite(23, LOW);}
if (Valor[7] == 0) {digitalWrite(22, HIGH);} else {digitalWrite(22, LOW);}

if (Valor[8] == 0) {digitalWrite(30, HIGH);} else {digitalWrite(30, LOW);}
if (Valor[9] == 0) {digitalWrite(31, HIGH);} else {digitalWrite(31, LOW);}
if (Valor[10] == 0) {digitalWrite(32, HIGH);} else {digitalWrite(32, LOW);}
if (Valor[11] == 0) {digitalWrite(33, HIGH);} else {digitalWrite(33, LOW);}

if (Valor[12] == 0) {digitalWrite(34, HIGH);} else {digitalWrite(34, LOW);}
if (Valor[13] == 0) {digitalWrite(35, HIGH);} else {digitalWrite(35, LOW);}
if (Valor[14] == 0) {digitalWrite(36, HIGH);} else {digitalWrite(36, LOW);}
if (Valor[15] == 0) {digitalWrite(37, HIGH);} else {digitalWrite(37, LOW);}
} else {
if (Valor[0] == 1) {digitalWrite(29, HIGH);} else {digitalWrite(29, LOW);}
if (Valor[1] == 1) {digitalWrite(28, HIGH);} else {digitalWrite(28, LOW);}
if (Valor[2] == 1) {digitalWrite(27, HIGH);} else {digitalWrite(27, LOW);}
if (Valor[3] == 1) {digitalWrite(26, HIGH);} else {digitalWrite(26, LOW);}

if (Valor[4] == 1) {digitalWrite(25, HIGH);} else {digitalWrite(25, LOW);}
if (Valor[5] == 1) {digitalWrite(24, HIGH);} else {digitalWrite(24, LOW);}
if (Valor[6] == 1) {digitalWrite(23, HIGH);} else {digitalWrite(23, LOW);}
if (Valor[7] == 1) {digitalWrite(22, HIGH);} else {digitalWrite(22, LOW);}

if (Valor[8] == 1) {digitalWrite(30, HIGH);} else {digitalWrite(30, LOW);}
if (Valor[9] == 1) {digitalWrite(31, HIGH);} else {digitalWrite(31, LOW);}
if (Valor[10] == 1) {digitalWrite(32, HIGH);} else {digitalWrite(32, LOW);}
if (Valor[11] == 1) {digitalWrite(33, HIGH);} else {digitalWrite(33, LOW);}
```

```
if (Valor[12] == 1) {digitalWrite(34, HIGH);} else {digitalWrite(34, LOW);}
if (Valor[13] == 1) {digitalWrite(35, HIGH);} else {digitalWrite(35, LOW);}
if (Valor[14] == 1) {digitalWrite(36, HIGH);} else {digitalWrite(36, LOW);}
if (Valor[15] == 0) {digitalWrite(37, HIGH);} else {digitalWrite(37, LOW);}
}

int COMANDO2 = B00001010;
PORTF = ~COMANDO2;

int MODESW = B00000000;
PORTK = ~MODESW;
delay(500);
PORTF = PORTF & B10111111;
delay(1000);
PORTF = PORTF | B01000000;
delay(1000);

//Outra
for(int j=0; j<16; j++) {Valor[j]=0;}
intReady = StringReady.toInt();
//Serial.println(intRead);
if(intReady < 0){Valor[15] = 1; intReady = abs(intReady)-1;}
else {Valor[15] = 0;}

intReady = abs(intReady);
for(int j=0; intReady>1; j++) {
  Valor[j] = intReady % 2;
  intReady = intReady / 2;
  if(intReady < 2){Valor[j+1] = 1;}
}

if (Valor[15] == 0){
```

```
if (Valor[0] == 0) {digitalWrite(29, HIGH);} else {digitalWrite(29, LOW);}
if (Valor[1] == 0) {digitalWrite(28, HIGH);} else {digitalWrite(28, LOW);}
if (Valor[2] == 0) {digitalWrite(27, HIGH);} else {digitalWrite(27, LOW);}
if (Valor[3] == 0) {digitalWrite(26, HIGH);} else {digitalWrite(26, LOW);}

if (Valor[4] == 0) {digitalWrite(25, HIGH);} else {digitalWrite(25, LOW);}
if (Valor[5] == 0) {digitalWrite(24, HIGH);} else {digitalWrite(24, LOW);}
if (Valor[6] == 0) {digitalWrite(23, HIGH);} else {digitalWrite(23, LOW);}
if (Valor[7] == 0) {digitalWrite(22, HIGH);} else {digitalWrite(22, LOW);}

if (Valor[8] == 0) {digitalWrite(30, HIGH);} else {digitalWrite(30, LOW);}
if (Valor[9] == 0) {digitalWrite(31, HIGH);} else {digitalWrite(31, LOW);}
if (Valor[10] == 0) {digitalWrite(32, HIGH);} else {digitalWrite(32, LOW);}
if (Valor[11] == 0) {digitalWrite(33, HIGH);} else {digitalWrite(33, LOW);}

if (Valor[12] == 0) {digitalWrite(34, HIGH);} else {digitalWrite(34, LOW);}
if (Valor[13] == 0) {digitalWrite(35, HIGH);} else {digitalWrite(35, LOW);}
if (Valor[14] == 0) {digitalWrite(36, HIGH);} else {digitalWrite(36, LOW);}
if (Valor[15] == 0) {digitalWrite(37, HIGH);} else {digitalWrite(37, LOW);}
} else {
if (Valor[0] == 1) {digitalWrite(29, HIGH);} else {digitalWrite(29, LOW);}
if (Valor[1] == 1) {digitalWrite(28, HIGH);} else {digitalWrite(28, LOW);}
if (Valor[2] == 1) {digitalWrite(27, HIGH);} else {digitalWrite(27, LOW);}
if (Valor[3] == 1) {digitalWrite(26, HIGH);} else {digitalWrite(26, LOW);}

if (Valor[4] == 1) {digitalWrite(25, HIGH);} else {digitalWrite(25, LOW);}
if (Valor[5] == 1) {digitalWrite(24, HIGH);} else {digitalWrite(24, LOW);}
if (Valor[6] == 1) {digitalWrite(23, HIGH);} else {digitalWrite(23, LOW);}
if (Valor[7] == 1) {digitalWrite(22, HIGH);} else {digitalWrite(22, LOW);}

if (Valor[8] == 1) {digitalWrite(30, HIGH);} else {digitalWrite(30, LOW);}
if (Valor[9] == 1) {digitalWrite(31, HIGH);} else {digitalWrite(31, LOW);}
if (Valor[10] == 1) {digitalWrite(32, HIGH);} else {digitalWrite(32, LOW);}
```

```
if (Valor[11] == 1) {digitalWrite(33, HIGH);} else {digitalWrite(33, LOW);}

if (Valor[12] == 1) {digitalWrite(34, HIGH);} else {digitalWrite(34, LOW);}
if (Valor[13] == 1) {digitalWrite(35, HIGH);} else {digitalWrite(35, LOW);}
if (Valor[14] == 1) {digitalWrite(36, HIGH);} else {digitalWrite(36, LOW);}
if (Valor[15] == 0) {digitalWrite(37, HIGH);} else {digitalWrite(37, LOW);}
}
MODESW = B10000000;
PORTK = ~MODESW;
delay(500);
PORTF = PORTF & B10111111;
delay(1000);
PORTF = PORTF | B01000000;
//comando automático
//LIGAR MOTOR
COMANDO2 = B00001110;
PORTF = ~COMANDO2;
MODESW = B10000000;
PORTK = ~MODESW;
delay(500);
PORTF = PORTF & B10111111;
delay(1000);
PORTF = PORTF | B01000000;
//Executar programa
COMANDO2 = B00001000;
PORTF = ~COMANDO2;
//numero programa
PORTA = B11111111;
PORTC = B11111111;
delay(500);
digitalWrite(29, LOW);
digitalWrite(A5, LOW);
delay(500);
```

```
    //start program
    MODESW = B01000000;
    PORTK = ~MODESW;
    delay(500);
    PORTF = PORTF & B10111111;
    delay(1000);

    for(int m=0; m<20; m++) {
    PORTF = PORTF | B01000000;
    delay(1000);
    PORTF = PORTF & B10111111;
    delay(3000);
    }
    digitalWrite(A5, HIGH);
  }
}
```