UNIVERSIDADE FEDERAL DO ABC

CMCC - Centro de Matemática, Computação e Cognição

Felipe Augusto Massari

RA: 11015812

Robô Repositor Guiado por Imageamento Externo

Prof. Dr. Rogério Perino de Oliveira Neves

Orientador

Santo André, SP

Agosto de 2013

Felipe Augusto Massari

Robô Repositor Guiado por Imageamento Externo

Trabalho submetido à Universidade Federal do ABC como parte dos requisitos do projeto de pesquisa do Jovens Talentos da Ciência.

Orientador: Professor Dr. Rogério Perino de Oliveira Neves

Santo André, SP
Agosto de 2013

RESUMO

A logística é de grande importância hoje nas indústrias e empresas tendo um papel fundamental na organização e fluxo tanto de informações quanto objetos, tal fluxo necessita de agilidade e organização para não afetar outros setores e muitas vezes precisam de um grande número de pessoas para que as tarefas sejam executadas. Embora apresente benefícios, a automação na logística é uma área pouco desenvolvida devido ao alto custo e em algumas vezes mão de obra bastante especializada. Tendo em vista a viabilidade da realização de tais tarefas, visa-se a implantação de um sistema autônomo de reposição e organização de peças de acordo com uma configuração prévia realizada pelo computador, a fim de evitar erros e tendo maior agilidade, controle e menos burocracia no transporte interno das mercadorias. Um projeto em escala reduzida de um robô foi desenvolvido com o intuito de realizar tarefas de organização com fácil manuseio e baixo custo. O protótipo robótico utiliza a linguagem Matlab, que recebe imagens do cenário e envia os comandos para a placa Arduino do robô, que controla as ações. O robô consiste em dois motores traseiros que controlam a movimentação e direção do mesmo, na parte frontal existe uma garra acionada por um motor de passo que faz a abertura e fechamento para captura do objeto, o robô realiza as funções de acordo com a captura da imagem do local e o processamento para verificação se algo necessita ser movido para outro local de acordo com a configuração estabelecida. Foram realizados testes iniciais com as partes separadas para verificar o funcionamento das garras e comando enviados da placa Arduino para o motor, posteriormente foram testadas todas as partes juntas.

Palavras-Chave: logística; robótica; automação; controle, arduino; matlab.

SUMÁRIO

Lista de Figuras

INTRODUÇÃO

A automação na logística visa facilitar o trabalho humano para que realizemos outras tarefas que sejam necessárias tomadas de decisão, deixando assim o trabalho manual e pesado para máquinas. Buscando sempre a melhoria e qualidade nos serviços, o ser humano faz implementação de novas tecnologias para facilitar e agilizar seu trabalho sempre com segurança, eficiência e baixos custos.

A logística segundo MELO & OLIVEIRA (2006), tem sofrido grandes transformações nos últimos 20 anos em todo mundo, cada dia surgem novas tecnologias e abordagens gerenciais que impulsionam e a tornam mais importante. Com isso, há um interesse em desenvolver tecnologia nesse ramo para implementação de serviços automatizados o que gera maior agilidade e precisão no processo de estocagem e reposição.

O avanço da tecnologia vem proporcionando as empresas executarem serviços que antes não eram possíveis de forma automática e a logística vem sendo beneficiada por tal processo. Com isso, as empresas podem reduzir custos e diminuir etapas operacionais, que segundo GARCIA (2008) essas ações se tornam fundamentais para o crescimento e sobrevivência em um mundo globalizado e competitivo.

Os robôs móveis, mesmo com toda a tecnologia, ainda apresentam limitações quanto à sua capacidade de navegação (Cazangi & Figueiredo, 2000), entretanto tem havido um grande interesse em pesquisas voltadas para automação de processos por meio de sistemas robóticos com intuito de promover a melhoria da qualidade dos produtos e otimização do tempo (Araújo et al., 2013). Esses sistemas utilizam várias técnicas de algoritmo para a resolução de problemas e tomadas de decisões a partir das informações que recebem.

O desenvolvimento de uma teoria relacionada ao projeto de robôs móveis autônomos traz consequências práticas importantes considerando-se as diversas aplicações a que podem ser empregados (Cohen et al., 2002). O desenvolvimento de um protótipo de robô autônomo pode ser aplicado a alguma situação específica o gera dificuldades dependendo do grau de complexidade e especificidade das suas funções.

O objetivo desse trabalho é desenvolver um protótipo, utilizando visão computacional e inteligência artificial para que possa ser implementado em diversos locais no setor da logística.

1. OBJETIVOS

1.1. Objetivo Geral

O principal objetivo deste projeto é a montagem de um protótipo inteligente capaz de organizar de maneia autônoma peças dispostas em uma arena de acordo com uma configuração prévia programada pelo operador. Para tanto, foi desenvolvido um robô de baixo custo utilizando uma placa open-source Arduino, motores para locomoção e uma câmera para coleta das imagens do cenário de trabalho.

1.2. Objetivo específico

Utilizar a linguagem de programação Matlab para realizar a programação e o processamento digital das imagens, detectando a falta de objetos e enviando comandos para os acionadores. Com o software criou-se um sistema autônomo de tomada de decisão que determine a melhor forma de organizar as peças, alterando sua disposição, e atingindo a configuração desejada.

O protótipo processa os dados coletados na imagem e determina a melhor rota que o robô utilizará até o objeto de captura. Ao capturar as imagens e localizar os objetos o software realiza uma série de cálculos para decidir qual o melhor caminho, levando em conta a distância e os objetos que por ventura estejam no caminho.

O projeto usou eletrônica digital para acionar o mecanismo, baseado principalmente da plataforma de desenvolvimento software-hardware Arduino, que será responsável pela tradução dos comandos em impulsos elétricos enviados aos motores e demais partes eletrônicas. Ele também conta com uma câmera digital que faz a leitura do local para identificar a disposição dos objetos, a comunicação entre robô-computador é feita através de comunicação sem-fim *bluetooth*.

O protótipo desenvolvido se movimenta em um plano através de dois motores ligados as suas rodas traseiras com movimentos independentes um do outro, que são tracionadas nos sentidos horário e anti-horário possibilitando sua locomoção em todos os sentidos. Para captura dos objetos o robô utiliza uma garra no estilo pinça de baixo torque, para não danificar os objetos.

2. REVISÃO BIBLIOGRÁFICA

2.1. Software Matlab

MATLAB é um aplicativo científico de alto desempenho especialmente indicado para cálculo numérico, vastamente empregado na realização de modelagens de sistemas e algoritmos. Esse software também integra análise numérica, cálculo com matrizes, processamento de sinais e imagens, além de construção de gráficos, entre outros recursos (DALCASTAGNÊ, 2008).

Os elementos básicos de informação do MATLAB são matrizes de dimensão arbitrária. Esse sistema permite a resolução de muitos problemas numéricos de maneira mais rápida do que escrever um programa semelhante em linguagens como Java ou C. Além disso, as soluções dos problemas são expressas no MATLAB quase exatamente como elas são escritas matematicamente.

O software, inicialmente desenvolvido para lidar com matrizes e vetores, atualmente dispõe de uma de uma vasta biblioteca de funções matemáticas, geração de gráficos e manipulação de dados. Ele ainda possui uma vasta coleção de bibliotecas denominadas *toolboxes* para áreas específicas, como equações diferencias ordinárias, estatística, processamento de imagens, processamento de sinais, finanças e muitas outras.

A linguagem e o ambiente de programação MATLAB permitem ainda que o usuário possa escrever suas próprias bibliotecas, podendo assim enriquecer a linguagem, incorporando a ela novas funções.

A *toolbox* para o processamento de imagem, *Image Processing Toolbox*, fornece um conjunto amplo de funções e aplicativos para o processamento de imagens podendo, por exemplo, executar o melhoramento da imagem através de filtros, redução de ruído e segmentação de imagens. A *toolbox* suporta um conjunto diversificado de tipos de imagem, incluindo alta gama dinâmica, além de funções de visualização permitem explorar uma imagem, região, ajustar o contraste, criar contornos ou histogramas e manipular regiões de interesse.

2.2. Comunicação Bluetooth

Com o surgimento de novas tecnologias, diversas alternativas ao uso de fios e cabos foram desenvolvidas, entre elas o *Bluetooth* que é considerado um padrão global de comunicação sem fio de curto alcance. Apresenta baixo consumo de energia, é robusto e barato;

permite a transmissão de dados no modo *full-duplex* entre dispositivos compatíveis com a tecnologia, tais como: celulares, computadores, fones de ouvido, teclado, mouse, etc (OLIVEIRA, 2003). A transmissão de dados é feita por ondas de rádio na frequência de micro-ondas, permitindo que um dispositivo detecte o outro independentemente de sua posição, desde que dentro do limite de alcance. Segundo ALECRIM (2008), a velocidade de comunicação é baixa, de até 1Mbps e há 3 classes de operação para o *Bluetooth*: Classe 1 (até 100 m); Classe 2 (até 10 m) e Classe 3 (até 1 m).

Ainda segundo ALECRIM (2008) o Bluetooth é uma tecnologia criada para operar no mundo inteiro, logo é necessário uma frequência de rádio aberta e padrão no mundo todo, uma frequência aberta em todos os países que vai de 2,4 GHz a 2,5 GHz. Como o Bluetooth utiliza uma faixa aberta de comunicação é necessário garantir a integridade da informação transmitida, portanto utiliza-se um esquema de transmissão denominado FH-CDMA (Frequency Hopping – Code-Division Multiple Acess) que consiste de modo simples, na divisão da frequência de comunicação em vários canais.

2.3. Plataforma Arduino

O arduino é uma plataforma aberta (open source) de hardware e software flexível e de fácil uso (Ref. ARDUINO), projetado com a finalidade de fácil entendimento e operação. O arduíno é baseado em uma placa microcontrolada por um microprocessador Atmel RISC de 8 bits (ATMEGA328), com entradas e saídas bi-direcionais. A plataforma tem sua programação simplificada pelo uso de uma linguagem similar ao C++, que gera executáveis diretamente transferidos para o microcontrolador ATMEGA328. Através das conexões na placa pode-se ler e controlar diversos dispositivos como sensores, servos motores e outros elementos eletrônicos.

3. METODOLOGIA

Com intuito de desenvolver um protótipo de baixo custo e fácil montagem foram utilizados materiais disponíveis visando também facilidade na aquisição das peças para sua construção. O projeto conta com uma placa Arduino que faz a comunicação entre computador e componentes eletrônicos, assim é possível passar todos comandos através de uma só placa reduzindo o espaço ocupado no robô. A comunicação entre a placa Arduino e o computador é feita através do código desenvolvido em Matlab, responsável pela análise dos dados e tomadas de decisões de acordo com a configuração dos objetos e posição do robô.

A parte de locomoção conta com dois motores com redutores e controles independentes, para que o protótipo tenha fácil locomoção e maior precisão na direção, que está na parte traseira. Na frente ele conta com apoio em uma esfera omnidirecional para que possa se locomover em qualquer direção com baixo atrito. Já a garra é acionada por um motor de passo com ângulo de movimentação de 180º, fixado em um eixo que faz o movimento de abertura e fechamento da garra, esta é compostas por duas pás que funcionam como uma pinça com a finalidade de agarrar os objetos e leva-los ao destino.

3.1. Programação

A parte de programação do protótipo foi dividida em módulos para maior facilidade no seu desenvolvimento, o esquema segue o fluxograma abaixo:

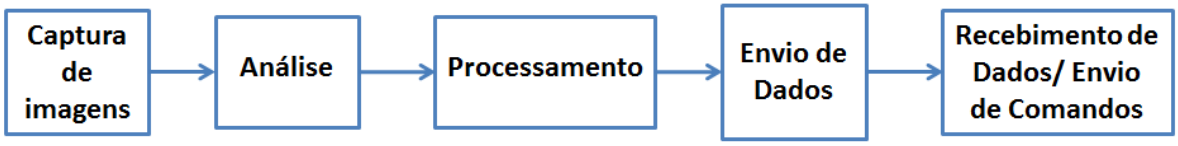


Figura 1: Resumo dos módulos de programação

O módulo captura de imagens é onde a aquisição das imagens é feita através de uma câmera externa conectada ao computador, que envia as imagens para análise. A câmera não necessita de uma alta velocidade de captura, já que os objetos não estão em movimento, apenas tem seu posicionamento alterado e volta novamente ao repouso.

Na análise dos dados, as imagens capturadas são enviadas quadro-a-quadro para o Matlab, que converte para uma imagem binária para melhor distinção dos objetos e fundo. Feita a conversão o software faz uma análise do posicionamento dos objetos a cada frame e em

seguida compara-os com uma disposição desejada. Após isso o software é responsável pelo processamento dos dados obtidos e tomadas de decisão. Caso não haja diferença na comparação das imagens o mesmo volta para a análise do frame seguinte.

O envio de dados se faz via comunicação *bluetooth* e só é enviado algum dado caso tenha alguma modificação entre a imagem capturada e a desejada. Caso seja preciso movimentação de algum objeto é enviado um comando do computador via *Bluetooth* contendo as informações de movimentação para que o robô execute. Os dados enviados são compostos por sete bytes, responsáveis pelas ações do robô, tal como girar o motor para um determinado lado, fechar ou abrir as garras ou acender o LED de identificação. O 1º byte encapsula o pacote e é sempre igual a 170, enquanto os demais são respectivamente:

1. Mapeamento dos 8 bits de controle
    a. Bits 1-4: configuração da ponte-h (motores)
    b. Bit 5: Estado da garra (0-Aberta, 1-Fechada)
    c. Bits 6-8: LED RGB
2. Velocidade do motor 1
3. Velocidade do motor 2
4. Numero pulsos para o motor
5. Tempo do pulso em estado-alto
6. Tempo do pulso em estado baixo

Os dados enviados pelo computador são recebidos pela placa Arduino que processa as informações e aciona leds, motores e de tração das rodas traseiras e de controle da garra. O Arduino trata os dados como comandos que ele executa de acordo com sua programação. O esquema abaixo mostra as etapas do processo da parte de inteligência, mostrando a maneira que o robô se comporta do ponto de vista computacional conforme a análise dos dados e as tomadas de decisão.

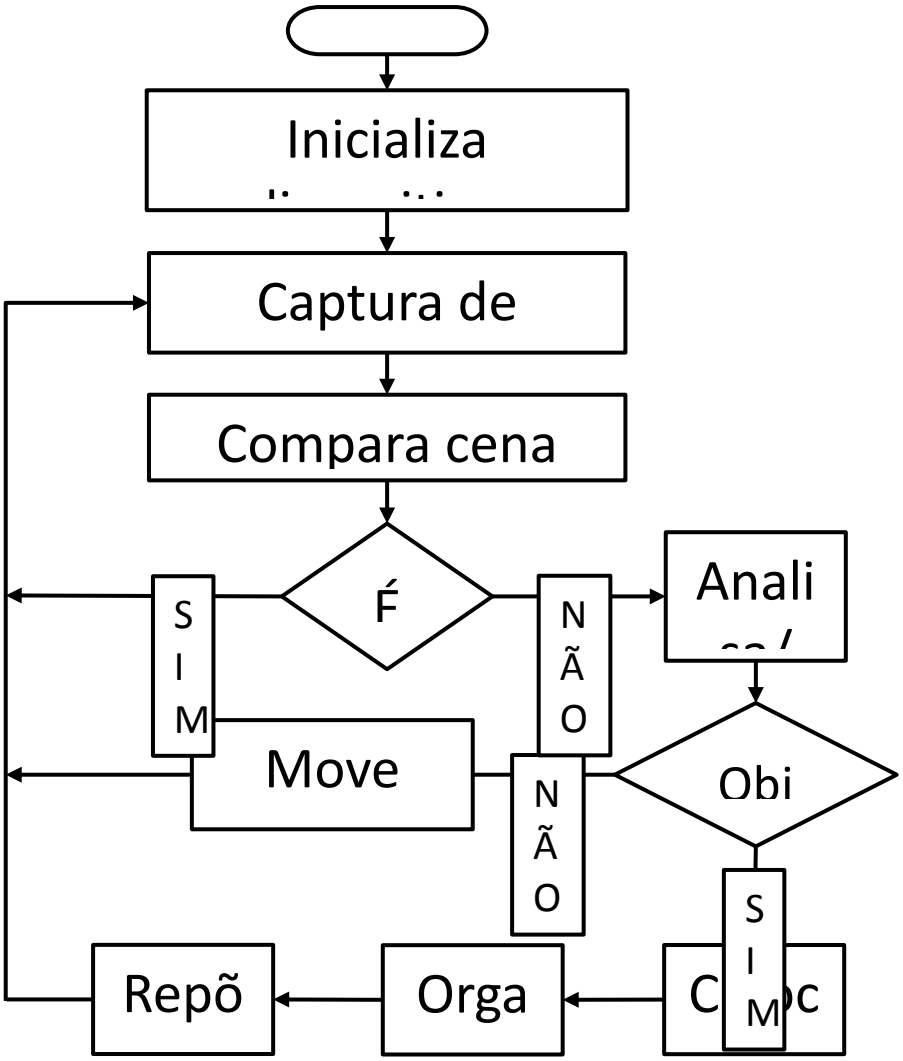


Figura 2: Fluxograma do processo de tomada de decisões

3.2. Montagem do Protótipo

O protótipo foi montado em partes separadas e depois agrupadas para realização de testes e ajustes. A mecânica utilizou material de acrílico que foi modelado na oficina multiusuário da UFABC usando ferramentas que garantissem boa precisão, principalmente na garra que exigia um ajuste fino.

A parte eletrônica foi montada em módulos independentes na parte acima da base de acrílico para que houvesse uma melhor disposição e assim fosse reduzido o tamanho do robô, o que facilitou seu deslocamento entre os objetos no espaço da arena. A foto abaixo mostra o protótipo já montado com os módulos e demais partes.

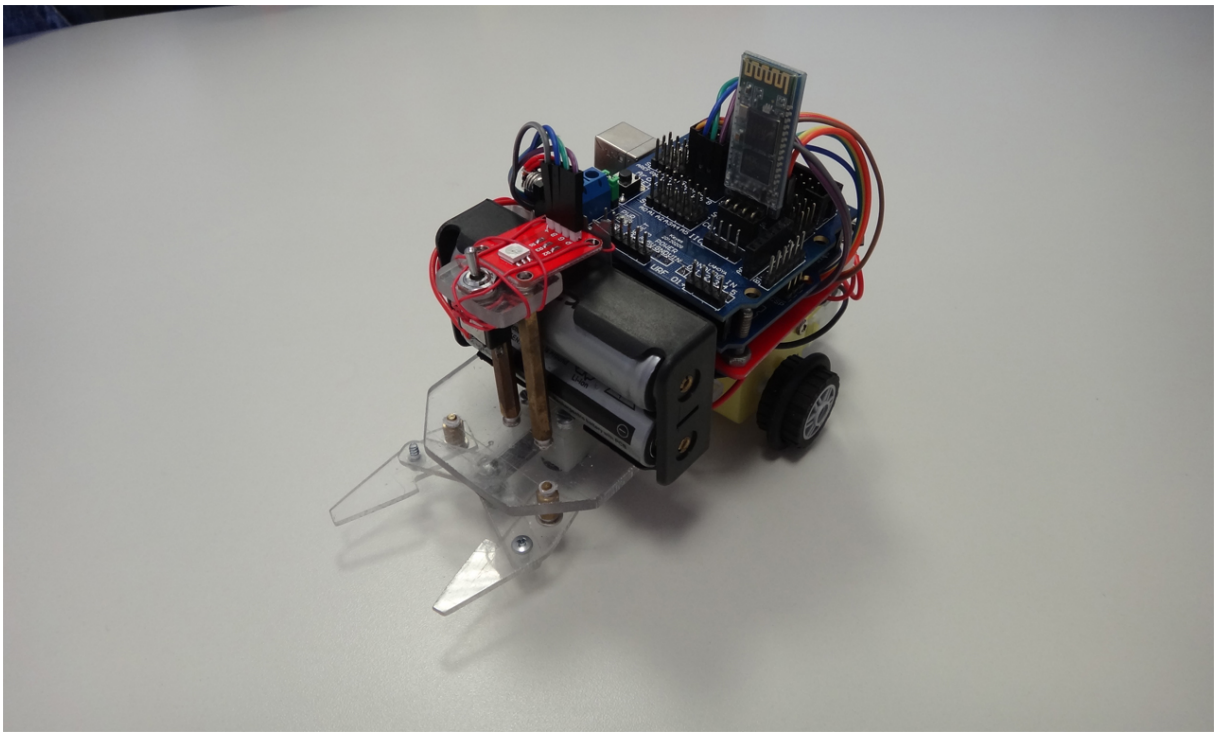


Figura 3: Protótipo montado com os módulos, bateria, motores e garra

Como parte do projeto, há uma base na qual o robô e as peças a serem organizadas estão e também onde fica um suporte para uma câmera posicionada acima da arena, que faz a captura das imagens para processamento.

3.3. Arena de Trabalho do Protótipo

A arena onde o protótipo fica exposto consiste em uma área retangular e plana com o fundo branco para maior distinção entre os objetos robô e arena. Tal contraste é importante para que o sistema de captura e análise das imagens consiga definir bem o limite dos objetos para uma aproximação correta.

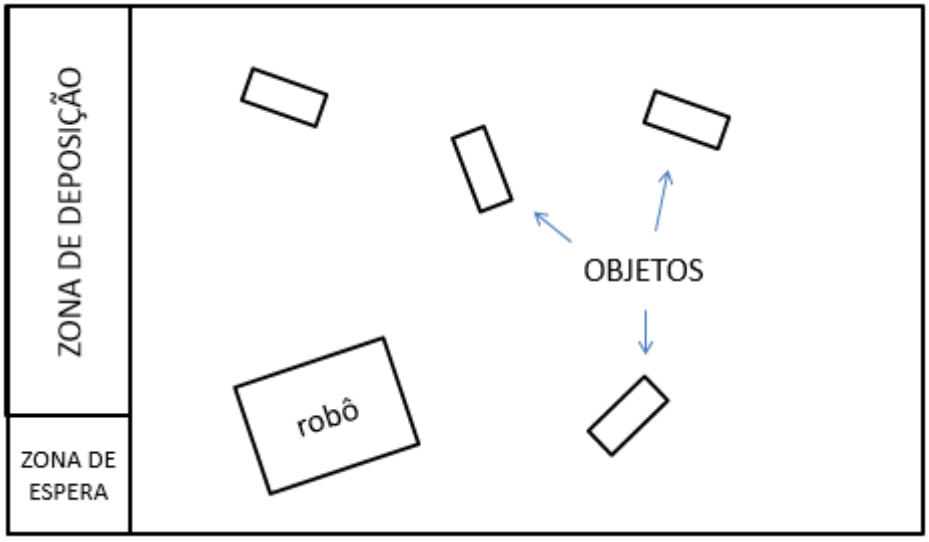


Figura 4: Modelo da arena contendo os objetos a serem organizados

No local onde o robô REPOUSA há também um espaço para que o mesmo possa colocar os objetos que estiverem atrapalhando sua passagem até o local de destino. Esse local de armazenamento é onde o protótipo permanece quando não estiver executando nenhuma função, tal espaço é identificado como “zona de espera” e fica na lateral da arena.

4. PARTES DO PROJETO

4.1. Sistema de Aquisição, Decisão e Controle

Os módulos de programação são responsáveis pelas funcionalidades do sistema, tais como: aquisição de dados, processamento, transmissão de dados e atuação mecânica. Eles foram divididos sub rotinas sendo cada uma delas responsável por executar uma função do robô.

4.1.1. Captura e processamento de imagem

As imagens do local serão capturadas por uma câmera RBG colocada sobre a arena, como os objetos no local não estão em constante movimentação, não é necessário utilizar uma câmera com alta taxa de amostragem (frames por segundo), uma vez que a função da imagem é apenas comparar o estado atual de disposição dos objetos com o estado final desejado.

Foi desenvolvido um módulo do software em Matlab para processar as imagens capturadas pela câmera. O mesmo ao receber as imagens capturadas faz a conversão da imagem RGB ou YUV para uma imagem binária para melhor distinção entre os objetos, robô e a arena. Ele trata a imagem com sendo uma matriz binaria, na qual os valores podem assumir “0” ou “1”, em que o zero significa ausência de elemento (robô ou objeto) e 1 significa presença de elemento.

4.1.2. Detecção dos objetos na imagem

Uma vez convertida a imagem o software localiza os objetos presentes utilizando segmentação, utilizando a área de cada objeto para distingui-los entre robô e objetos na arena. Para evitar ruído tanto na imagem convertida quanto nos dados processados foi utilizado uma arena que contraste com o robô e os objetos, sendo que o software trata a arena como ausência de cor, ou seja, zero na matriz binaria. Na localização o software retorna as suas coordenadas em vetores, assim como o ângulo que se encontram, em relação à base da arena.

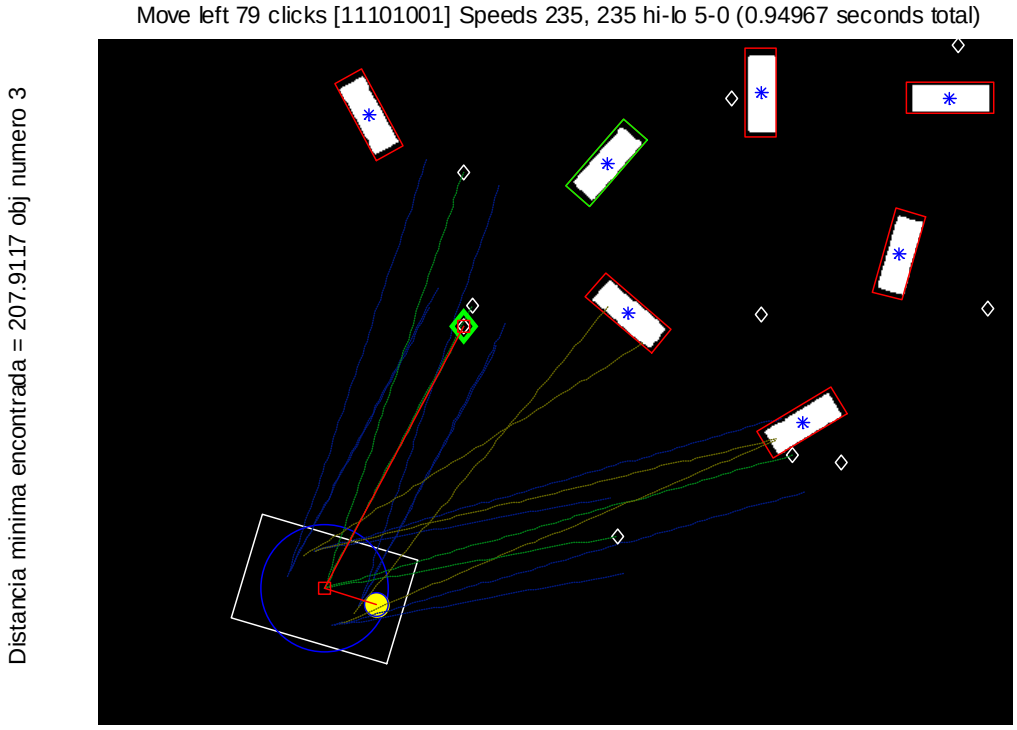


Figura 5: Robô e objetos detectados na imagem binária

As coordenadas são fundamentais para que o software possa saber a localização e assim poder definir uma rota viável até o objeto o de interesse.

4.1.3. Definição de rotas

Após a obtenção dos dados como: captura da imagem, conversão para imagem binária e detecção dos objetos o software define qual o melhor caminho de onde se encontra até o objeto mais próximo. Para isso é usado um algoritmo de otimização de rota para buscar o melhor caminho, assim o robô sabe qual percurso fazer, escolhendo o mais curto e desobstruído.

Para definir qual melhor rota o software faz análise da imagem e verifica quais objetos estão fora do local e se há algum obstáculo até tal objeto. Ao encontrar a rota mais eficiente o software envia os comandos a serem executados pelo robô que leva em conta a distância percorrida e obstáculos.

4.1.4. Movimentação e controle

A movimentação do robô é feita por duas rodas traseiras que são tracionadas cada uma por um motor com redução Alimentado com 7V. Também é utilizado um motor de passo 5V

para controlar o dispositivo responsável por pegar os objetos. O controle dos motores e servo é feito através da placa Arduino que recebe os controles via *bluetooth* enviados pelo computador.

A comunicação entre Arduino e motores é feita por uma placa de dupla ponte-H que tem como função amplificação e inversão das correntes, quando necessário. Nela é possível conectar os dois motores e o sinal recebido do Arduino faz com que ela envie tensão para um ou para os dois motores podendo inverter a polaridade, o que inverte o sentido de rotação deles.

Foi desenvolvido um programa para o Arduino com papel de comunicador, que interpreta os dados recebidos pela serial (módulo *Bluetooth*) executando as operações de configuração e acionamento dos motores, leds indicadores e servo, fazendo a interface de comunicação entre o Matlab e os motores de controle do protótipo.

Os dados enviados ao Arduino (8 bytes) ativam ou desativam os pinos e determinam velocidades e tempos. Dessa forma quando os comandos são passados aos motores eles executam tal movimento por um tempo de modo que ao final ele se encontra na posição desejada, aguardando pronto para executar o próximo comando. Na foto abaixo segue o esquema de conexões entre motores, Arduino, ponte-H e leds.

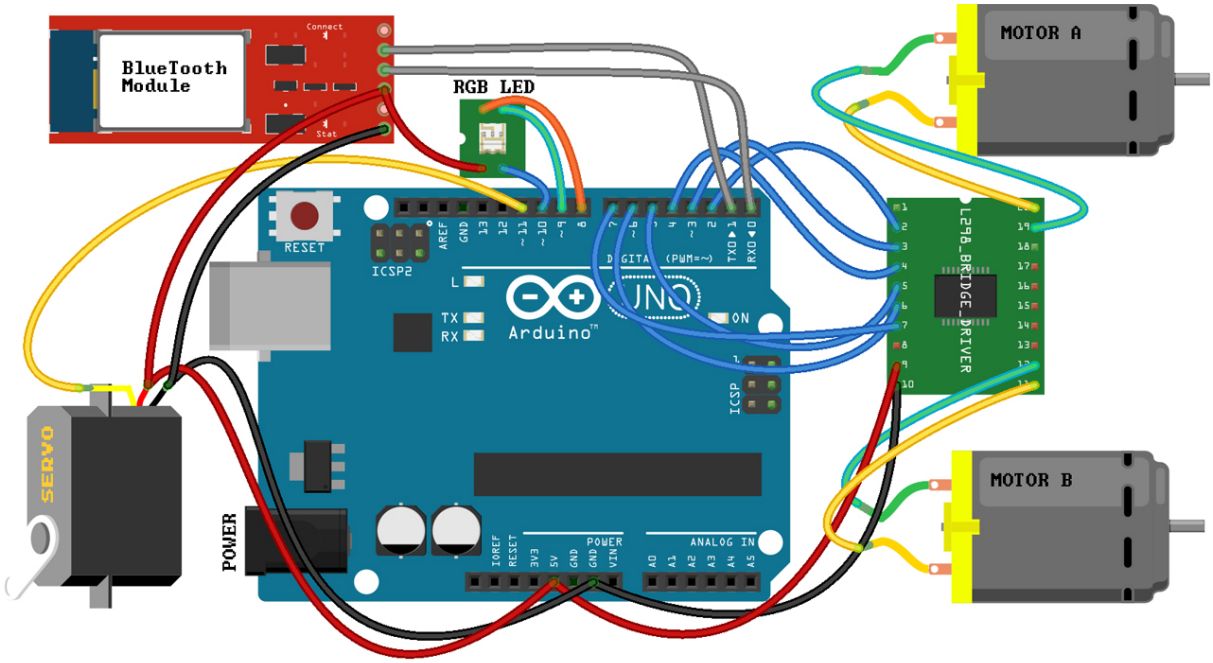


Figura 6: Esquema de conexão dos componenetes ao Arduino

4.1.5. Módulos de programação

A seção “Apêndice C” contém os códigos desenvolvidos para Arduino e Matlab utilizados no projeto. Seguem uma breve descrição das rotinas em Matlab desenvolvidas:

Abre

O módulo abre é o que inicializa a comunicação entre o programa e o arduíno. Quando inicializado ele verifica se há portas seriais e estabelece uma comunicação através dela.

Captura

Recebe a imagem e faz a conversão da imagem de RGB para uma imagem binária, transformando-a em uma matriz somente com os elementos 0 e 1.

Após a conversão da imagem foi usado a ferramenta `regionprops` que faz a separação e contagem dos elementos presentes na imagem e encontra sua orientação, centro de massa e área dos mesmos. O robô é identificado na imagem como área acima de um determinado número de pixels enquanto os objetos a serem organizados com área num certo intervalo, visando evitar a identificação de possíveis ruídos como objetos.

Detecta Robô

Utilizando os dados obtidos com o regionprops esse módulo identifica o robô encontrando seu eixo maior (a), eixo menor (b) e seu centro de massa. Feito isso é aplicado uma matriz de rotação para R para determinar a orientação do robô e é desenhado seu contorno na imagem, na sequência ele é removido da imagem binária para análise dos demais objetos.

Detecta Alvo

Esse módulo do software é responsável por identificar os objetos a organizar, essa parte funciona de maneira semelhante ao detecta robô, porem com algumas funções a mais. Pois além de identificar os objetos, encontrar sua orientação, centro de massa e seus eixos maiores e menores ele vai desenhar o contorno e também um ponto na parte frontal e posterior do objeto para que a sub-rotina chamada rota possa analisar os dados enviados e retornar para determinar qual o caminho desobstruído mais próximo do robô.

Rota

A função rota recebe os parâmetros referentes ao tamanho do robô, do objeto, dos pontos acima e abaixo do objeto, representados por diamantes na imagem, e retorna se o caminho está livre e a distância até eles. Esta sub-rotina é responsável por determinar se o percurso do robô

até o objeto está livre, ela representa isso com uma reta que vai do centro do robô ao centro do objeto, feito isso será delimitado uma largura por mais duas retas paralelas à linha central (uma acima e outra abaixo) para que o robô possa passar livremente. Após isso são retornadas as rotas livres para o módulo detecta alvo, para que o mesmo analise qual a menor distância.

O código que executa essa etapa está em anexo no apêndice C, 11.4.

Aproxima

Este módulo trata a aproximação do robô até os objetos através de um loop com ajuste fino a cada movimentação feita pelo robô. Primeiramente é calculado o ângulo entre os vetores que dão a orientação do robô e do objeto com relação ao eixo das abscissas e para corrigir os ângulos caso sejam negativos (já que o código retorna entre os valores de 0º à 180º e 0º à-180º) é somado 360º. Encontrando o ângulo e fazendo os ajustes é calculado o ângulo entre os vetores robô e objeto, fazendo com que o robô se vire para direita ou esquerda até ficar dentro do limite estabelecido de desvio (3º). Feito o alinhamento entre eles é analisada a distância entre ambos e caso ela ainda não esteja dentro do limite (2 pixels), é enviado o comando para seguir em frente por um tempo proporcional à distância (a velocidade de movimentação é conhecida). Então a rotina recomeça para determinar novamente os ângulos e posições.

Move

O modulo move é uma função que tem o papel de fazer a comunicação entre o Matlab e a placa Arduino. Na transmissão dos dados são enviados 7 bytes, responsáveis pelo funcionamento e controle dos motores para movimentação e giros do robô, abertura e fechamento das garras e acionamento dos leds de identificação.

Organiza

Esse último módulo é responsável pela organização dos objetos na “zona de depósito”. Uma vez atingido o objeto mais próximo e capturado o mesmo, esta parte do software irá delimitar uma faixa na vertical do lado esquerdo da imagem e fragmenta-la de modo que caiba um objeto em cada espaço. Feito isso será analisado qual desses espaços estão livres para receber o objeto agarrado pelo robô. Esta análise será semelhante aos módulos detecta robô e detecta alvo, com a diferença que será procurado o primeiro espaço livre (de cima para baixo) e levado até ele o objeto.

4.2. Protótipo Robótico

4.2.1. Estrutura do Protótipo

O protótipo foi construído utilizando placas de acrílico para que pudesse ter um robô leve e com estética aceitável. O chassi conta basicamente com duas partes, a primeira é a base do robô onde estão as placas eletrônicas motores e baterias e a segunda é o modelo de garra proposto para capturar os objetos.

As placas eletrônicas estão dispostas na parte superior da base alocadas em modulo de maneira a ocupar menos espaço possível e assim reduzir o tamanho do protótipo para que assim ele tenha maior facilidade de mobilidade entre os objetos.

As garras são formadas por duas pás (1 pinça) que se movem através de um eixo que transmite o movimento até elas. O apêndice A mostra o projeto da parte da garra desenvolvida em *Solid Works* onde constam as partes individuais e a montagem final dos mesmos na garra.

Foi adicionada também uma “carapaça” que cobre o robô escondendo seus dispositivos eletrônicos conforme a foto abaixo. Essa estrutura foi feita para evitar ruídos na imagem capturada e confeccionado na cor vermelha que facilita o software fazer seu reconhecimento distinguindo dos demais objetos.

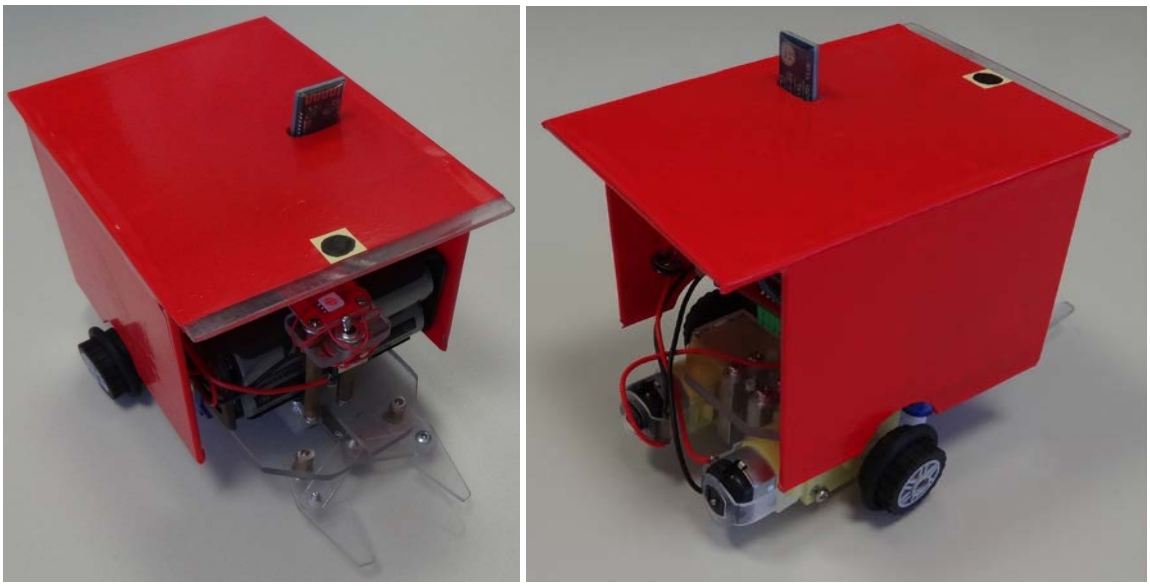


Figura 7: Imagens do Robô com a carapaça para facilitar localização na imagem

A estrutura do robô visa reduzir os custos e teve como objetivo principal a criação de um protótipo estruturalmente simples que execute as funções necessárias de maneira rápida e pratica. O anexo B mostra em o projeto da garra acoplado na base vista em vista explodida.

4.2.2. Sistema de Captura dos Objetos

O sistema de captura dos objetos foi desenvolvido através de estudo sobre qual seria a forma mais eficiente de realizar o movimento, porém sem uma alta complexidade e com espaço reduzido para não comprometer o tamanho da estrutura do protótipo, através disso chegou-se ao modelo de garra tipo pinça com movimento unidimensional.

Para a construção do mecanismo de captura foi utilizado acrílico, por ser um material de fácil acesso, fácil modelagem e com uma boa estética o que dispensa acabamentos posteriores. Foi utilizado material com as seguintes espessuras, 4 mm para confecção do guia por onde passará um eixo principal, além do próprio eixo. A parte das pinças e articulações da garra foi utilizado material na espessura de 2 mm, permitindo assim um melhor funcionamento do conjunto mecânico

O processo de captura dos objetos é feito através de uma garra frontal que consiste em duas pás que formam uma pinça e agarra o objeto e tem sua movimentação realizada por um eixo que transmite o movimento horizontalmente. Com os estudos realizados se conseguiu uma melhor eficiência na tarefa e o modelo proposto e o resultado foi um modelo constituído por um motor de passo que tem liberdade de rotação de 0º à 180º e transmite o movimento para que as pás façam os movimentos de abertura e fechamento como desejado.

Os estudos e desenvolvimento do modelo de garra utilizou o software *SolidWorks*, onde foram realizadas simulações para verificar o desempenho da garra quando em execução no protótipo. O resultado pode ser verificado no link que mostra a animação do resultado final: Ref. YOUTUBE1.

5. RESULTADOS

.

No teste da garra foi simulado no *SolidWorks*, um modelo com o objetivo de verificar a abertura e fechamento das pinças. Porém na construção observou-se a necessidade de um pequeno ajuste para aumentar a abertura da garra. Ainda nesse teste, foi analisado se o modo de transmissão de movimentos funcionaria de acordo com o projeto, o que de fato ocorreu.

Testes com os motores foram realizados com o objetivo de verificar seu funcionamento na ponte-H e o controle de velocidade dos mesmos. Eles foram ligados à placa de circuitos programada para fazê-los girar em sentido horário e anti-horário. E também foi testada a comunicação entre o computador e a placa que recebe os comandos para movimentação dos motores.

Na parte de software foram desenvolvidas e finalizadas as funções responsáveis pela captura, processamento, detecção de objetos, definição de rotas e envio de comandos, que foram testados com imagens e posteriormente foram realizados testes com fotos da arena com o robô e os objetos (foto abaixo).

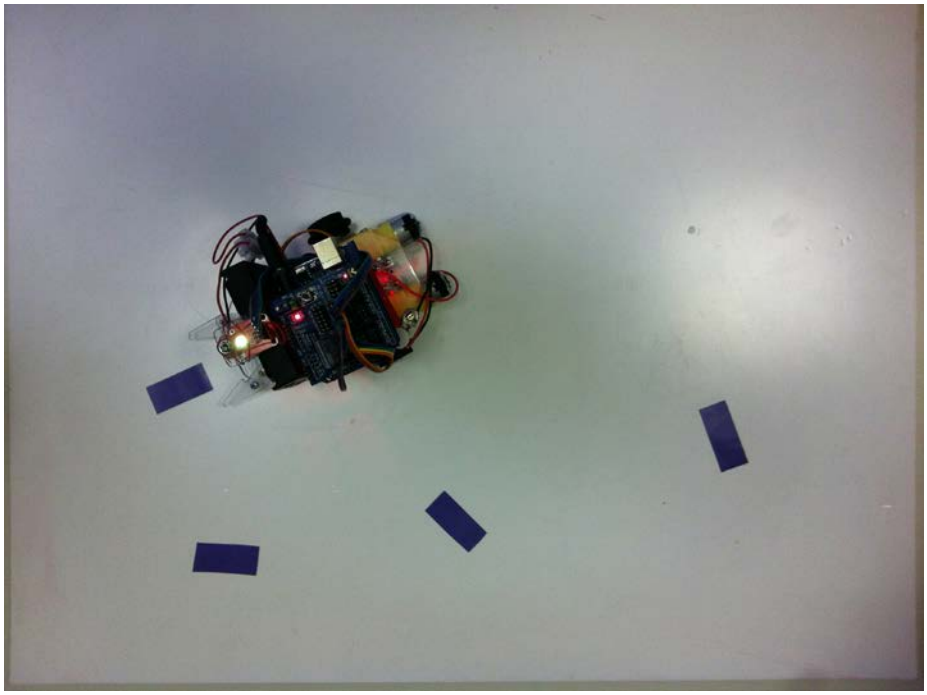


Figura 8: Imagem da arena com robô e objetos

Após a escrita do código e utilização de ferramentas do *Image Processing Toolbox,* com os ajustes os testes realizados obtiveram resultados satisfatórios. Quando testados com imagens capturadas da câmera obteve-se desempenho semelhante, porém ainda são necessários ajustes finos para a precisão do funcionamento completo do robô para a função desejada (trabalhos futuros).

De acordo com o que estava proposto no início do projeto e pelo cronograma, as tarefas foram concluídas de maneira satisfatório e o protótipo passou por todas etapas, superando os desafios encontrados durante sua execução. Segue o link do vídeo do protótipo em funcionamento, mostrando seu desempenho e funções: Ref. YOUTUBE2

6. CONCLUSÕES

Após testes realizados com o protótipo construído conclui-se que os objetivos foram alcançados e obtivemos sucesso no cumprimento das metas. Ao longo desse ano foram trabalhados diversos conceitos tanto na área de programação, quanto mecânica e eletrônica o que acrescentou bastante para o crescimento e desenvolvimento tanto acadêmico quanto profissional, pois também foi necessário superar limitações, trabalhar em grupo cumprir prazos e trabalhar seguindo os padrões acadêmicos.

No desenvolvimento do projeto foi abordado desde os conceitos básicos de programação até conceitos relativamente avançados, como o que ensinou a tratar com dados coletados e trata-los de maneira eficiente para que obtivéssemos os parâmetros desejados. Para conseguir esse resultado foram usados vários conceitos aprendidos com as disciplinas da graduação, o que acrescentou muito para o aprendizado e melhor entendimento dos assuntos.

Também foi necessário a busca de conhecimento além daqueles propostos pelo curso, como os de mecânica, para montagem e transmissão de movimentos e eletrônica, para o entendimento e desenvolvimento das ações realizadas pelo protótipo, o que fez enriquecer ainda mais o aprendizado.

Devido à falta de conhecimentos na área de programação, essa parte do projeto foi a que apresentou maior dificuldade e necessitou uma dedicação muito maior, até por esta ser o principal foco do projeto, pois todo o seu funcionamento dependia do sucesso dessa etapa. Embora tenha sido a maior dificuldade e exigindo maior dedicação a parte de programação foi concluída com sucesso e a grande dificuldade consistiu de usar conhecimentos matemáticos que ainda não tinham sidos abordados e traze-los para linguagem de programação.

Com desenvolvimento dos módulos separadamente, houve grande sucesso na integração das partes, pois os ajustes foram mínimos e o funcionamento foi correto desde o início. A execução da parte mecânica também obteve um rendimento considerável, pois foi a etapa em que houve maior aprendizado. O destaque maior foi o desenvolvimento da garra do protótipo, já que foi desenvolvido um modelo de movimentação através de transmissão angular em movimento linear.

As situações em onde ocorreram problemas ou algum desafio fundamental para a sequência do projeto foram os momentos mais difíceis, entusiasmante e motivador, pois havia a necessidade de resolvê-los para o progresso do projeto. Em sua maioria houve sucesso e nos

momentos de dúvida e dificuldade, a ajuda do professor foi fundamental para a superação dos obstáculos.

Apesar de todas adversidades o protótipo foi concluído dentro do prazo faltando apenas pequenos ajustes para que o robô tenha um funcionamento perfeito no cenário proposto, tais ajustes finos ainda serão realizados visando corrigir e adequar o modelo de acordo com o proposto inicialmente.

7. REFERÊNCIAS

Apostila de Matlab do Curso Engenharia Mecânica – UFRGS. Disponível em: http://www.mecanica.ufrgs.br/promec/alunos/download/matlab1.pdf (Acesso em 17/03/13)

ALVES DE ARAÚJO, Sidnei et al. Navegação Autônoma de Robôs: Uma Implementação Utilizando o Kit Lego Mindstorms. Anais SULCOMP, v. 2, n. 2, 2013.

Cazangi, R. R. & Figueiredo, M. F. (2000) "Sistema de Navegação Autônoma de Robôs Baseado em Sistema de Classificação com Aprendizado e Algoritmos Genéticos". In: Anais do II Fórum de Informática e Tecnologia de Maringá e V Mostra de Trabalhos em Informática da UEM, Maringá-PR, p. 29–29.

Cohen H. et al. (2002) "Vision based pursuing of moving vehicle from bird's view – PART I and PART II". The Vision and Image Sciences Laboratoy– TECHNION – Israel Institute Technology.

Componentes de um algoritmo genético. Disponível em: http://www.deti.ufc.br/~pimentel/disciplinas/ica_files/Documentos/Algorimos_Geneticos.pdf Acesso em 25/03/2013

DALCASTAGNÊ, A. L. Apostila Básica do Software Matlab, CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE SANTA CATARINA – CEFET-SC, Florianópolis, 2008. Disponível em: http://www.pessoal.utfpr.edu.br/richard/arquivos/digitais/matlab.pdf (Acesso em 21/03/13).

Image Processing Tolbox. Disponível em: http://www.mathworks.com/products/image/ (Acesso em 19/03/13)

Linden, Ricardo. *Algoritmos Genéticos - uma importante ferramenta da inteligência* computacional - 2ª Edição. BR: Brasport, 2008

Melo, I. H. B. S. & Oliveira, M. V. D. S. S. Automação da armazenagem: o caso da Multi Distribuidora. XIII SIMPEP - Bauru, SP, Brasil, 2006 acesso em http://antigo.feb.unesp.br/dep/simpep/anais/anais_13/artigos/547.pdf

MOORE, H. Matlab for Engineers. Prentice Hall; 3 edition, 2011

OLIVEIRA, R. A. R. Bluetooth e Multimídia. In *Anais do IV Workshop em Tratamento de Imagens, NPDI/DCC/ICEx/UFMG* (pp. 14-25), 2003. Disponível em: http://laplace.dcc.ufmg.br/npdi/uploads/c8962954-8be0-43ed.pdf (Acesso em 23/03/13).

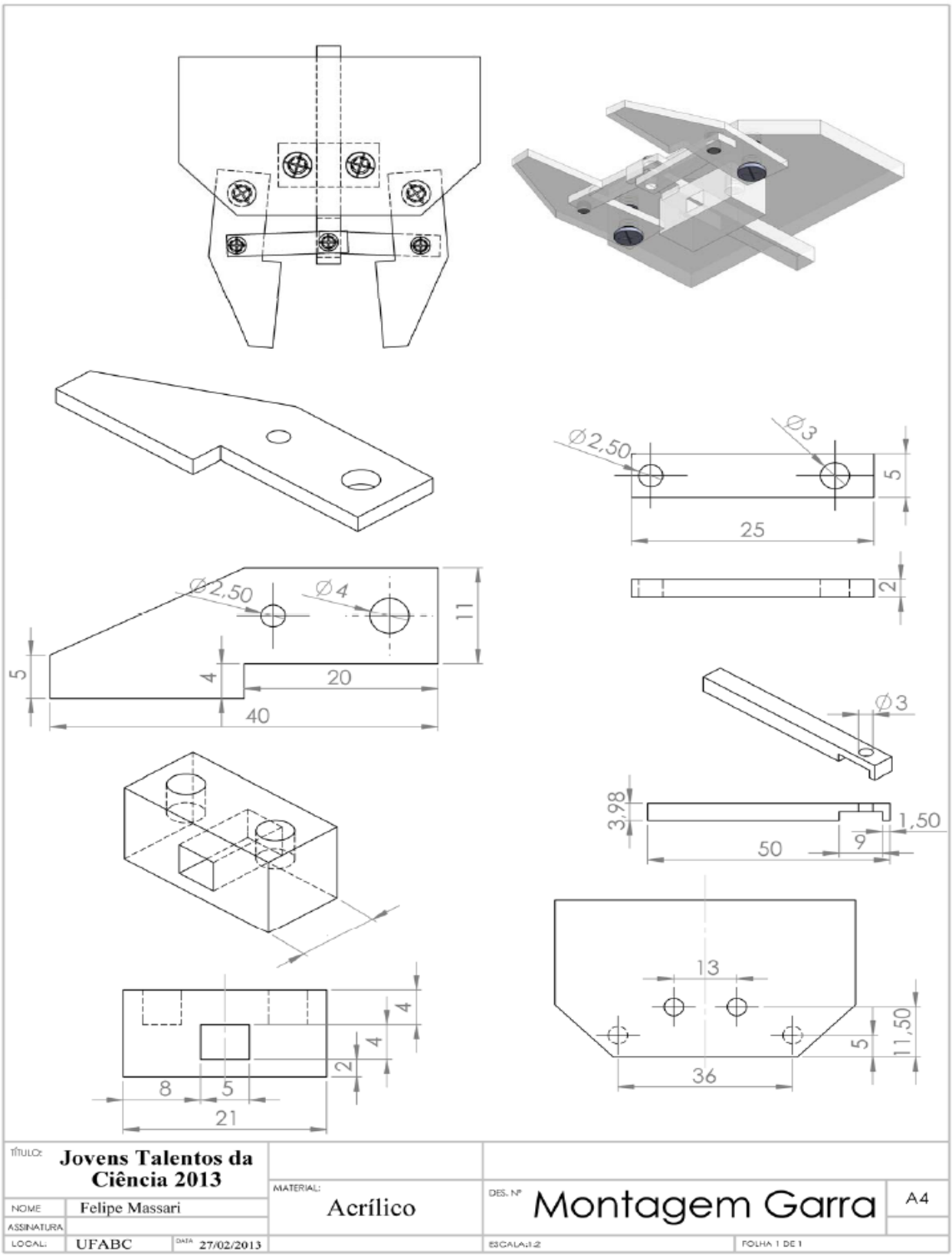
PACHECO, M. A. C. "Algoritmos genéticos: princípios e aplicações." ICA: Laboratório de Inteligência Computacional Aplicada. Departamento de Engenharia Elétrica. Pontifícia Universidade Católica do Rio de Janeiro, (1999). Disponível em: http://www.ica.ele.puc-rio.br/downloads/38/ce-apostila-comp-evol.pdf (Acesso em 23/03/13).

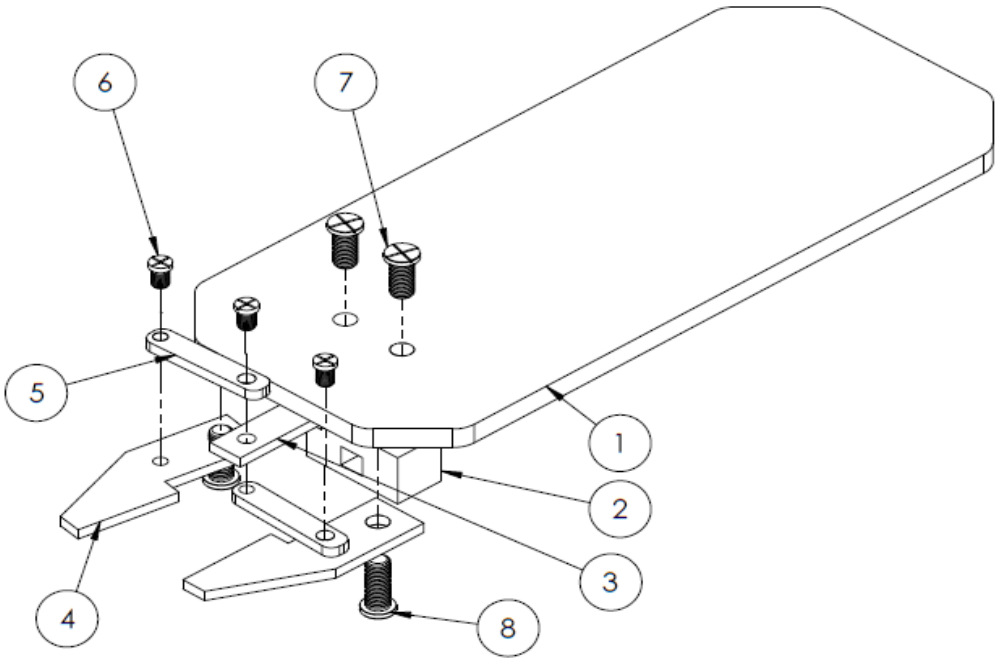
ARDUINO Home: http://arduino.cc/

YOUTUBE1: Modelo de operação da Garra http://youtu.be/7n9OYWml_5M

YOUTUBE2: Vídeo do robô em operação http://youtu.be/vMBWdECpnCI

8. APÊNDICE A – Construção da garra





Nº	ITEM			
1	Base	7	Parafuso M3 x 10mm	TÍTULO: VISTA EXPLODIDA DO PROTÓTIPO DE MONTAGEM
2	Guia para Eixo	8	Parafuso M3 x 12mm	
3	Eixo	9		
4	Garras	10		
5	Hastes de Apoio	11		
6	Parafuso M2 x 4mm	MATERIAL: ACRÍLICO		Montagem explodida — A4
NOME	Felipe Augusto Massari			
DATA	15/03/2013			ESCALA:1:5 — FOLHA 1 DE 1

9. APÊNDICE B – Montagem do chassi

10. APÊNDICE C – Códigos fonte

10.1. Código fonte do Arduino em C

```
#include <Servo.h>
//defines
#define LD 13
#define LDR 8
#define LDG 9
#define LDB 10
#define CLW 11
#define MA1 2
#define MA2 3
#define MB1 4
#define MB2 5
#define ENA 6
#define ENB 7
#define fecha 10
#define abre 60

Servo garra;
int a, s1, s2, t, t0, t1;
int i, b, val = 0; // serial port data

void claw(boolean state) {
  if (state) { garra.write(fecha); }
  else { garra.write(abre); }
}
void move(int speed1, int speed2, int cont, int d1, int d2) {
  if (speed1 >238) speed1 = 238;
  if (speed2 >238) speed2 = 238;
  for (i=0; i<cont; i++) {
    analogWrite(ENA, speed1);
    analogWrite(ENB, speed2);
    delay(d1);
    analogWrite(ENA, LOW);
    analogWrite(ENB, LOW);
    delay(d2);
  }
}
void pinblink(int pin, int cont, int d1, int d2) {
    for (i=0; i<cont; i++) {
      digitalWrite(pin, HIGH);
      delay(d1);
      digitalWrite(pin, LOW);
      delay(d2);
  }
}
void setup() {
  digitalWrite(LDR, LOW);
  digitalWrite(LDG, HIGH);
  digitalWrite(LDB, HIGH);
  for(i=2;i<13;i++) {
    pinMode(i, OUTPUT);
  }
  garra.attach(CLW);
  garra.write(fecha);
  Serial.begin(9600);
  pinblink(LD,10,100,100);
  garra.write(abre);
```

```
  digitalWrite(LDR, HIGH);
}
void loop () {
  val = Serial.read();
    if (val == 170) {
      a  = Serial.read();
      s1 = Serial.read();
      s2 = Serial.read();
      t  = Serial.read();
      t1 = Serial.read();
      t0 = Serial.read();
      digitalWrite(MA1, (a >> 0) & 1);
      digitalWrite(MA2, (a >> 1) & 1);
      digitalWrite(MB1, (a >> 2) & 1);
      digitalWrite(MB2, (a >> 3) & 1);
      claw((a >> 4) & 1);
      digitalWrite(LDR, !((a >> 5) & 1));
      digitalWrite(LDG, !((a >> 6) & 1));
      digitalWrite(LDB, !((a >> 7) & 1));
      if (s1>0||s2>0) move(s1,s2,t,t1,t0);
      delay(10);
    } else { pinblink(LD,2,100,100); }
}
```

10.2. Matlab: Abre (ponto de entrada no programa)

```
% Inicializa serial
if ~exist('ser','var')
    list = instrhwinfo('serial');
    disp([num2str(max(size(list.AvailableSerialPorts))) ' devices
found.']);
    disp([list.AvailableSerialPorts]);
    if ~exist ('porta', 'var')

porta=list.AvailableSerialPorts(size(list.AvailableSerialPorts,1));
    end
    ser=serial(porta,'BaudRate',9600);
    disp(['* Porta serial ' char(porta) ' conectada com sucesso *'])
end
if strcmp(ser.status,'closed')
    fopen(ser);
end
disp(['* Serial ' ser.port ' aberta *']);
disp(ser);
% Seleciona fonte = 'arquivo.png', 'winvideo', 'macvideo', url, etc, ex:
% url = 'http://blogs.mathworks.com/images/steve/2010/rice_binary.png';
if ~sum(strcmp(fonte, {'winvideo' 'macvideo' 'etc'}))
    % L? Imagem de arquivo
    im = imread(fonte);
    disp(['Imagem de ' fonte ' lida com sucesso'])
end
global bit
if size(bit,2)<8
    bit= [0 0 0 0 0 0 0 0];
end
move(ser,'G',1);
move(ser,'R',1);
move(ser,'B',1);
EmPos = 0;
```

```
while(1)
    if exist('v','var' )
        start(v);
        im=getdata(v);
        stop(v);
    end
    captura
end
```

10.3. Matlab: Captura

```
% sub-rotina do programa abre.m
roboarea = 2000;
objarea = 100;
% Imagem binaria
bimg=1-im2bw(im(:,:,2), .6);  % Gera imagem binaria
S=size(bimg);
imshow(bimg); % Mostra imagem
pause(1)
hold on; % Segura imagem
% set(gcf, 'Units', 'normalized', 'Position', [0,0,1,1],
'MenuBar','none','NumberTitle','off', 'Visible','on');S = size(bimg)'; %
Tamanho da imagem
ref=round(.12*S(2));
figure(1);
subim1=bimg;
subim1(1:size(bimg,1),1:ref)=0; % Retira objetos já posicionados da cena
Lmin = sqrt(S(1)^2+S(2)^2); % maior distancia  possivel, diagonal
[L, Ne]=bwlabel(subim1); % Identifica elementos conexos na imagem
% Extrai medidas dos objetos
k=regionprops(L,  'Orientation', 'MajorAxisLength', 'MinorAxisLength',
'Centroid', 'Area');
% detecta ponto de referência no robô
% Amarelo=(im(:,:,2)<50)&(im(:,:,1)<50)&(im(:,:,3)<10);  % ponto amarelo
% imshow(Amarelo); pause(2)
%[Amy, Amx]=find(Amarelo==1);
gr=rgb2gray(im);
px=min(gr(:)); % ponto mais escuro
[Amy, Amx]=find(gr==px);
Ax=mean(Amx); Ay=mean(Amy); obj= -1;
P = [1;1]; r=30; % posição padrão

detecta_robo
detecta_alvo
drawnow;

if exist('xx', 'var')
    title('Movendo');
    ylabel(['Distancia minima encontrada = ' num2str(Lmin) ' obj numero '
num2str(obj)])
    plot(xx, yy,'-g');
    plot(DD(1,1),DD(2,1),'dg', 'MarkerSize', 10, 'LineWidth', 2)
    plot(P(1,1),P(2,1), 'sr')
    plot(DD(1,1),DD(2,1), 'sr')
    plot([P(1,1) DD(1,1)], [ P(2,1) DD(2,1)], '-r')
    plot([P(1,1) Ax], [P(2,1) Ay], '-r')

    % Determina movimentacao do robo
```

```
    dr = [Ax;Ay]-P;
    do = DD-P;
    if ~EmPos
        EmPos=aproxima(dr,do,Lmin,ser);
    else
        % Catura
        fim = aproxima(dr,do,10,ser);
        if fim
            move(ser,'hold',0)
            % Determina a posicao destino
            organiza;
        end
    end
else
    xlabel(['Nada encontrado'])
end
drawnow;
pause(1);
title('Acquiring...');
```

10.4. Matlab: Detecta_robô

```
% Detecta o robo
for n=1:length(k)
    if k(n).Area > roboarea
        P = [k(n).Centroid(1) k(n).Centroid(2)]';
        % retira robô da imagem
        BW2 = zeros(size(subim1));
        BW2(floor(k(n).Centroid(2)), floor(k(n).Centroid(1)))= 1;
        subim1 = bwremove(subim1, BW2); % Remove robo da imagem bin?ria
        imshow(subim1);
        a=k(n).MajorAxisLength/2;
        b=k(n).MinorAxisLength/2;
        r=b/1.4; % Raio com margem de segurança
        theta = pi*k(n).Orientation/180;   % angulo
        R = [ cos(theta)   sin(theta)
            -sin(theta)   cos(theta)];
        Borda = R*[a -a -a a a; b b -b -b b]; %  bordas
        x = Borda(1,:) + P(1,1);
        y = Borda(2,:) + P(2,1);
        plot(x, y,'-w');
        plot(P(1,1),P(2,1),'ob', 'MarkerSize', r*2);
        plot(Ax,Ay,'ob', 'MarkerSize', 10, 'MarkerFaceColor','y');
    end
end
```

10.5. Matlab: Detecta_alvo

```
% Identifica alvos
for n=1:length(k)
    if k(n).Area > objarea && k(n).Area < roboarea
        C=[k(n).Centroid(1) k(n).Centroid(2)]';
        plot(C(1,1),C(2,1),'*');
        a=k(n).MajorAxisLength/2;
        b=k(n).MinorAxisLength/2;
```

```
        theta = pi*k(n).Orientation/180;   % angulo
        R = [ cos(theta)   sin(theta)
             -sin(theta)   cos(theta)];

        Borda = R*[a -a -a a a; b b -b -b b]; %  bordas
        x = Borda(1,:) + C(1,1);
        y = Borda(2,:) + C(2,1);
        plot(x, y,'-r');

        % Pontos de destino
        D1 = R*[  5*a; 0] + C;  % Ponto branco de baixo
        D2 = R*[ -5*a; 0] + C;  % Ponto branco de cima
        plot(D1(1,1), D1(2,1), 'dw');
        plot(D2(1,1), D2(2,1), 'dw');

        [livre1, L1]=rota(P, D1, r, subim1);
        [livre2, L2]=rota(P, D2, r, subim1);
        if (livre1&&L1<Lmin), Lmin = L1; obj = n; xx=x; yy=y; DD=D1; end
        if (livre2&&L2<Lmin), Lmin = L2; obj = n; xx=x; yy=y; DD=D2; end
    end
end
```

10.6. Matlab: Rota

```
function [livre, L]=rota(P, D, r, bimg)
passo = 5; livre = 0;
L= dist(P', D); % Distância
S = size(bimg); % <- A imagem binária é uma matriz deste tamanho

% Acha matriz de rotação
theta = -atan( (D(2, 1)-P(2,1)) / (D(1, 1)-P(1,1)));   % angulo
R = [ cos(theta)   sin(theta);
    -sin(theta)   cos(theta)];

% Rotas do ponto P até o ponto D e paralelas
R1x= round(linspace (P(1,1), D(1,1), L/passo)); % caminho pontilhado
R1y= round(linspace (P(2,1), D(2,1), L/passo));
P1 = R*[0; -r] + P; % Ponto acima
D1 = R*[0; -r] + D;
R2x= round(linspace (P1(1,1), D1(1,1), L/passo));
R2y= round(linspace (P1(2,1), D1(2,1), L/passo));
P3 = R*[0; +r] + P; % Ponto abaixo
D3 = R*[0; +r] + D;
R3x= round(linspace (P3(1,1), D3(1,1), L/passo));
R3y= round(linspace (P3(2,1), D3(2,1), L/passo));
% testa se a rota sai fora da imagem
if (sum( R1x>S(2)|R1y>S(1)|R2x>S(2)|R2y>S(1)|R3x>S(2)|R3y>S(1))) %
verifica se a rota sai da tela
    return;
end
if (sum( R1x<1|R1y<1|R2x<1|R2y<1|R3x<1|R3y<1)) % verifica se a rota sai
da tela
    return;
end

% Testa se a rota está obstruída
```

```
livre=1;
for i=1:L/passo
    if bimg(R1y(i),R1x(i))==1
        plot(R1x(1:i), R1y(1:i), ':','Color',[.4 .4 0]);
        livre = 0;
        return
    end
    if bimg(R2y(i),R2x(i))==1
        plot(R2x(1:i), R2y(1:i), ':','Color',[.4 .4 0]);
        livre = 0;
        return
    end
    if bimg(R3y(i),R3x(i))==1
        plot(R3x(1:i), R3y(1:i), ':','Color',[.4 .4 0]);
        livre = 0;
        return
    end
end
plot(R1x, R1y, ':','Color',[.0 .5 .1]);
plot(R2x, R2y, ':','Color',[.0 .1 .5]);
plot(R3x, R3y, ':','Color',[.0 .1 .5]);
end
```

10.7. Matlab: Aproxima

```
function EmPos=aproxima(R, O, L, ser)
EmPos = 0;
angulo = acos(dot(R,O)/(norm(R)*norm(O)))*180/pi;
tg=atan2([R(2,1) O(2,1)],[R(1,1) O(1,1)])*180/pi;
ar = tg(1);
ao = tg(2);
% deixa a1 e a2 = angulos positivos
if ar<0
    ar = ar+360;
end
if ao<0
    ao = ao+360;
end

if angulo>3&&L>2 % caso ainda n?o esteja alinhado ou na posi??o
    if ar>ao
        if (ar-ao)<180
            move(ser,'left', angulo);
        else
            move(ser,'right', angulo);
        end
    else  % a2>a1
        if (ao-ar)<180
            move(ser,'right', angulo);
        else
            move(ser,'left', angulo);
        end
    end
else if (angulo<=3)&&(L>2) % caso angulo = certo e pos = errada
        move(ser,'up', L);
    else if ~isnan(angulo)
        EmPos = 1;
        end
    end
```

```
end
```

10.8. Matlab: Move

```
function move(ser, o, d)

%move (orientation, diistance)
%   This functions sends command to the robot
%   to move in certain direction.

global bit; % control bits
s1=235;s2=235;
t=uint8(d);
thi=uint8(5);
tlo=uint8(0);
delay=d*12*.001; % d * (thi + tlo + 2)
switch o
    case 'up'
        bit(1:4)=[0 1 0 1];
    case 'back'
        bit(1:4)=[1 0 1 0];
    case 'left'
        bit(1:4)=[1 0 0 1];
    case 'right'
        bit(1:4)=[0 1 1 0];
    case 'left-up'
        bit(1:4)=[0 1 0 1]; s1=s1/2;
    case 'right-up'
        bit(1:4)=[0 1 0 1]; s2=s2/2;
    case 'hold'
        bit(5)=~bit(5); s1=0;s2=0;
    case 'R'
        bit(6)=~bit(6); s1=0;s2=0;
    case 'G'
        bit(7)=~bit(7); s1=0;s2=0;
    case 'B'
        bit(8)=~bit(8); s1=0;s2=0;
    otherwise
        bit(1:8)=0; s1=0;s2=0;
end
a=uint8(bin2dec(num2str(bit(8:-1:1))));
if isa(ser,'serial')
    if strcmp(ser.status,'open')
        try
            fwrite(ser, [170 a s1 s2 t thi tlo],'uint8');
            pause(delay);
            disp(['Move ' o ' ' num2str(t) ' clicks [' dec2bin(a) ']
Speeds '...
                num2str(s1) ', '  num2str(s2) ' hi-lo ' num2str(thi) '-
'...
                num2str(tlo) ' (' num2str(delay) ' seconds total)']);
        catch
            disp('Erro no envio dos dados pela serial!')
        end
    else
        disp('Serial not started!');
    end
end
title(['Move ' o ' ' num2str(t) ' clicks [' dec2bin(a) '] Speeds '...
```

```
            num2str(s1) ', '  num2str(s2) ' hi-lo ' num2str(thi) '-'...
            num2str(tlo) ' (' num2str(delay) ' seconds total)']);
```

10.9. Matlab: Organiza

```
%seleciona zona de espera
figure(3); close
figure(3);
alvo=bimg(1:size(bimg,1),1:.15*size(bimg,2));
area1=bwarea(alvo);
subplot(1,2,1);
imshow (alvo);
tam=round(.11*size(alvo,1));
pp =1;
for i=1:9
    box=alvo(pp:pp+tam,1:size(alvo,2));
    subplot(1,2,1);
    plot([1, 1, size(alvo,2)-1, size(alvo,2)-1],[pp, pp+tam-1, pp+tam-1,
pp],'-b');
    pp=pp+tam;
    subplot(1,2,2); imshow(box);
    hold on
    pause(2);
    boxpos = sum(sum((box)))>objarea
    if boxpos==0
        break
    end
end
```


```
num2str(s1) ', '  num2str(s2) ' hi-lo ' num2str(thi) '-'...
num2str(tlo) ' (' num2str(delay) ' seconds total)']);
```